



Documentation for Bitbucket Data Center and
Server 7.14

Contents

Bitbucket Data Center and Server documentation	6
Get started with Bitbucket Data Center and Server	7
Tutorials	8
Tutorial: Work with Bitbucket Server	9
Set up Sourcetree	10
Create a personal repository for the tutorial	12
Clone your repository and manage files locally	13
Commit and push changes	16
Git resources	18
Basic Git commands	20
Get started with Git	23
Use Bitbucket Data Center and Server	26
Importing code from an existing project	27
Creating projects	30
Creating repositories	31
Creating personal repositories	33
Using repository hooks	35
Permanently authenticating with Git repositories	38
Clone a repository	40
Personal access tokens	41
Controlling access to code	44
Allowing public access to code	46
Using project permissions	47
Using repository permissions	49
Using branch permissions	51
Branch permission patterns	55
Using SSH keys to secure Git operations	56
Creating SSH keys	57
SSH user keys for personal use	60
SSH access keys for system use	62
Using GPG keys	65
Workflow strategies	69
Branches	70
Automatic branch merging	76
Forks	79
Keeping forks synchronized	81
Pull requests	84
Create a pull request	86
Reviewing a pull request	88
Add default reviewers to pull requests	93
Reviewer groups for pull requests	94
Commenting on a pull request	95
Merge a pull request	98
Search for pull requests	100
Checks for merging pull requests	101
Pull request merge strategies	106
Code Insights	111
Enhancements to your code review workflow	114
Bitbucket search syntax	123
Manage webhooks	126
Event payload	130
If you use self-signed certificates	164
Notifications	168
Pull request notifications	169
Repository notifications	170
Markdown syntax guide	172
Requesting add-ons	176
Set the default time zone	177

Download a repository archive	178
Creating a Contributions guidelines file	181
Working with Git LFS Files	182
Compare branches, tags, and commits	185
Administer Bitbucket Data Center and Server	186
Users and groups	187
External user directories	192
Connect to an LDAP directory	195
Delegate user management to Jira	203
Delegate authentication to an LDAP directory	206
Connect Bitbucket to Crowd	212
Global permissions	215
Setting up your mail server	217
Integrate with Atlassian applications	221
Jira integration	223
Link Bitbucket with Jira	227
Use custom Jira issue keys	239
Use Smart Commits	240
Hipchat notifications	244
Bamboo integration	246
Connect Bitbucket to an external database	249
Connecting Bitbucket Server to MySQL	251
Connect Bitbucket to Oracle	255
Connect Bitbucket to PostgreSQL	258
Connect Bitbucket to SQL Server	261
Transitioning from jTDS to Microsoft's JDBC driver	266
Configuring Bitbucket Data Center to work with Amazon Aurora	268
Migrating Bitbucket Server to another server	271
Run Bitbucket in AWS	274
Launch Bitbucket in AWS manually	276
Administer Bitbucket in AWS	281
Recommendations for running Bitbucket in AWS	286
Secure Bitbucket in AWS	300
Using Bitbucket DIY Backup in AWS	304
Specify the Bitbucket base URL	306
Configuring the application navigator	307
Managing apps	308
Add a POST service webhook	309
View and configure the audit log	312
Audit log events	316
Audit log integrations	324
Push logs	327
Update your license key	329
Configuration properties	330
Change Bitbucket's context path	396
Data recovery and backups	397
Bitbucket Server Backup Client	400
Bitbucket DIY Backup	407
Bitbucket zero downtime backup	418
Running integrity checks in Bitbucket Data Center	423
Disable HTTP(S) access to Git repositories	426
Smart Mirroring	427
Set up a mirror	429
Set up and configure a mirror farm	434
Monitoring your mirror farm	438
Cloning LFS content from a mirror farm with more than one node fails	441
Export and import projects and repositories	443
Exporting	446
Importing	451
LFS Migration	454
Troubleshooting, canceling and cleanup	456
Error and warning messages	458
Git Large File Storage	462
Git Virtual File System (GVFS)	466

Enable SSH access to Git repositories	467
Setting up SSH port forwarding	470
Use diff transcoding	474
Change the port Bitbucket listens on	475
Lockout recovery process	476
Proxy and secure Bitbucket	477
Secure Bitbucket with Tomcat using SSL	480
Integrate Bitbucket with Apache HTTP Server	487
Secure Bitbucket with Apache using SSL	492
Secure Bitbucket behind nginx using SSL	496
Secure Bitbucket behind HAProxy using SSL	499
High availability for Bitbucket	502
Diagnostics for third-party apps	507
Event system	511
Upgrade tasks	514
Enabling JMX counters for performance monitoring	515
Enable debug logging	525
Scaling Bitbucket Server	528
Scaling Bitbucket Server for Continuous Integration performance	539
Bitbucket Server production server data	541
Add a shortcut link to a repository	545
Administer code search	546
How to Install and configure a remote Elasticsearch instance	550
Configure Bitbucket's code search index	554
Use an Elasticsearch cluster with Bitbucket Data Center	556
Adding additional storage for your repository data	560
Add a system-wide announcement banner	562
Configuring Project links across Applications	564
Improving instance stability with rate limiting	566
Use a CDN with Atlassian Data Center applications	569
Configure your CDN for Bitbucket Data Center	573
Managing personal access tokens	577
Connecting to a 3rd party application using Application Links	578
Setting a system-wide default branch name	580
Automatically decline inactive pull requests	581
Encrypt database password	582
Basic encryption	583
Advanced encryption	585
Custom encryption	589
Advanced repository management	592
Xcode for Bitbucket Server	595
Data pipeline	596
Data pipeline export schema	604
Integrated CI/CD	612
Integrate your CI/CD pipeline	613
Link your CI server	615
Configure your CI server	617
View builds information in Bitbucket	618
Perform build actions	619
View build logs	620
Download build artifacts	621
Link Bitbucket with Bamboo	622
Link Bitbucket with Jenkins	623
Atlassian Bitbucket Integration plugin triggers builds for multiple branches	624
Adding an additional logger for troubleshooting the Atlassian Bitbucket Integration plugin	625
Install or upgrade Bitbucket	626
Supported platforms	627
End of support announcements	632
Installing and upgrading Git	635
Bitbucket installation guide	638
Install a Bitbucket trial	640
Install Bitbucket Server on Windows	643
Install Bitbucket Server on Windows from a zip file	647
Install Bitbucket Server on Linux	652

Install Bitbucket Server on Linux from an archive file	657
Running Bitbucket Server with a dedicated user	662
Running Bitbucket Server as a Windows service	663
Run Bitbucket as a Linux service	666
Automated setup for Bitbucket	671
Start and stop Bitbucket	673
Install Bitbucket Server from an archive file	677
Run the Bitbucket installer	684
Bitbucket Server upgrade guide	687
How to update your add-on	720
Upgrade Bitbucket from an archive file	733
Migrate server.xml customizations to bitbucket.properties	765
Bitbucket Data Center upgrade guide	771
Use Bitbucket in the enterprise	774
Upgrade Bitbucket without downtime	779
Upgrade a Bitbucket cluster manually without downtime	782
Upgrade a Bitbucket cluster on AWS without downtime	786
Upgrade a Bitbucket cluster through the API without downtime	790
Bitbucket Data Center	795
Running Bitbucket Data Center on a single node	796
Moving back to Server	798
Clustering with Bitbucket	799
Set up a Bitbucket Data Center cluster	805
Bitbucket Data Center requirements	814
Install Bitbucket Data Center	817
Migrate from Server to Data Center	827
Deploy Bitbucket Data Center in Azure	829
Administer Bitbucket Data Center in Azure	833
Disaster recovery guide for Bitbucket Data Center	836
Adding and removing Data Center nodes	846
Deploy Bitbucket Data Center in AWS	850
Administer Bitbucket Data Center in AWS	855
Recommendations for running Bitbucket Data Center in AWS	860
Bitbucket Data Center and Server feature comparison	874
FAQ	877
Bitbucket rebrand FAQ	882
How do I change the external database password	884
Set the home directory	885
Raising a request with Atlassian Support	888
Support policies	890
Bug fixing policy	891
New features policy	896
Security Bugfix Policy	899
Find your Support Entitlement Number (SEN)	900
Contribute to our documentation	902
Change data collection settings	903
Bitbucket Server and Data Center GDPR support guides	904
Automated individual decision-making, including profiling in Bitbucket Server and Data Center	905
Communication of personal data breaches in Bitbucket Server and Data Center	906
Data protection by design and by default in Bitbucket Server and Data Center	907
Records of processing activities in Bitbucket Server and Data Center	909
Right of access by the data subject in Bitbucket Server and Data Center	910
Right to data portability in Bitbucket Server and Data Center	912
Right to erasure in Bitbucket Server and Data Center	913
Right to object in Bitbucket Server and Data Center	919
Right to rectification in Bitbucket Server and Data Center	920
Right to restriction of processing in Bitbucket Server and Data Center	922
Security of processing in Bitbucket Server and Data Center	923
Transfers of personal data to third countries or international organisations in Bitbucket Server and Data Center	925

Bitbucket Data Center and Server documentation

Bitbucket Data Center and Server is self-hosted Git repository collaboration and management for professional teams.

Get started

New to using Bitbucket? Get started with some introductory tutorials.

[Let's get started!](#)

Whats new in 7.14?

Read all about the latest and greatest changes in Bitbucket.

[View release notes](#)

Get started with Bitbucket Data Center and Server

Welcome to the Bitbucket Data Center and Server getting started documentation. Here you'll find tutorials and other information that will be useful for evaluating Bitbucket, and figuring things out when you're just starting.

If you haven't installed Bitbucket yet, but you'd like to try it out (for free!) you're better off starting at the page [Install a Bitbucket trial](#). Once you have your trial instance up and running, head back here to explore using Bitbucket.

Go ahead, dive in. Let us show you around Bitbucket and some of its handy features.

- [Tutorials](#)
 - [Tutorial: Work with Bitbucket Server](#)
- [Git resources](#)
 - [Basic Git commands](#)
 - [Get started with Git](#)

Tutorials

If you're just starting out with Bitbucket Data Center and Server, then this is the place for you. Come with us on a journey to discover all that Bitbucket has to offer using our Teams in Space scenario.

 Before you continue, make sure you've [installed Bitbucket Server](#) already.

Teams in Space



For each tutorial in this section, we'll use a fictional organization known as 'Teams in Space'. Their mission is to:

"Perform flight research and technology integration to revolutionize aviation and pioneer aerospace technology. Also, land the first humans on Mars by 2020."

You're an astronaut in the 'See Space EZ' team, which is working on the upcoming colonization of Mars.

Jump into a tutorial when you're ready:

Work with Bitbucket and SourceTree

[With Bitbucket running, learn how to get your work done](#)

More to come!

Tutorial: Work with Bitbucket Server

Teams in Space is a fictional company created by Atlassian that specializes in space travel for teams.

Welcome to the Teams in Space web team! You are joining us as a Bitbucket Data Center and Serverweb developer, and your first assignment is to update our company website to include a link to our Moon Itinerary so that our customers know what to expect on their day trip to the Moon.

Here's what you'll accomplish by the end of this tutorial:

1. [Set up Sourcetree to work with Bitbucket Server](#)
2. [Create a personal repository for the tutorial](#)
3. [Clone your repository and manage files locally](#)
4. [Commit and push changes](#)

For this tutorial we'll be using Sourcetree, a desktop Git client with a graphical interface, to work with Bitbucket. If you're already comfortable using Git from the command line we'll also include the Git command equivalent.

Time needed

5-10 minutes

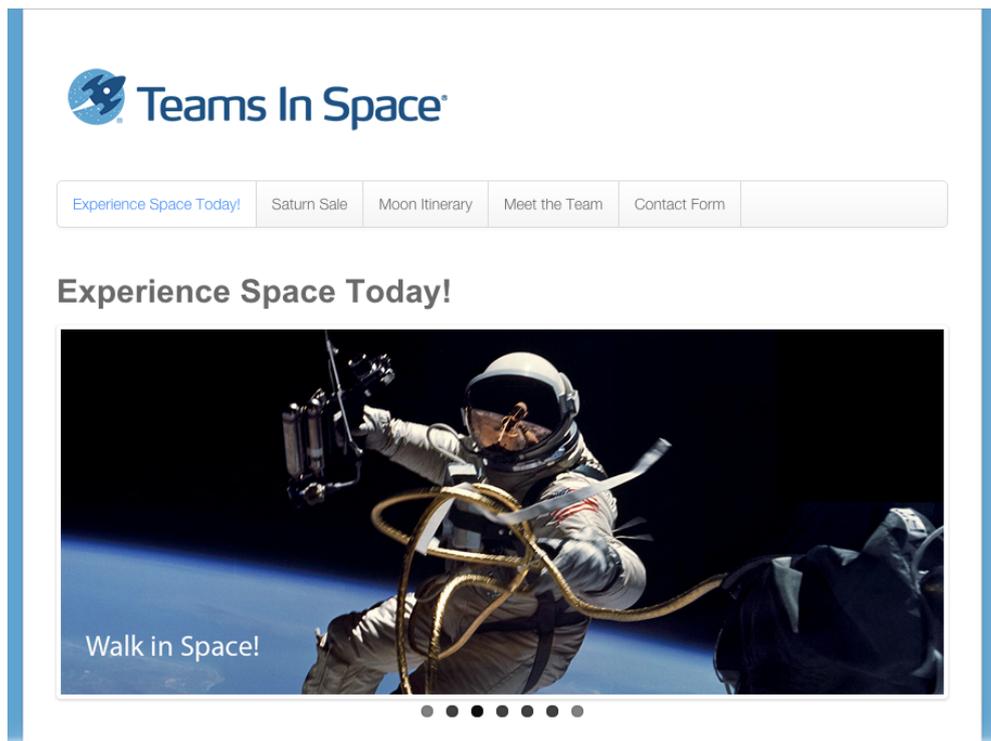
Audience

You're new to working with Bitbucket

Prerequisites

- [Bitbucket Server is installed](#)
- [You have login credentials](#)
- You have a [project](#) and [repo](#)

Here's what the final version of the HTML page will look like when you're finished (and we've got all the files you need to get this end result).



[Let's go!](#)

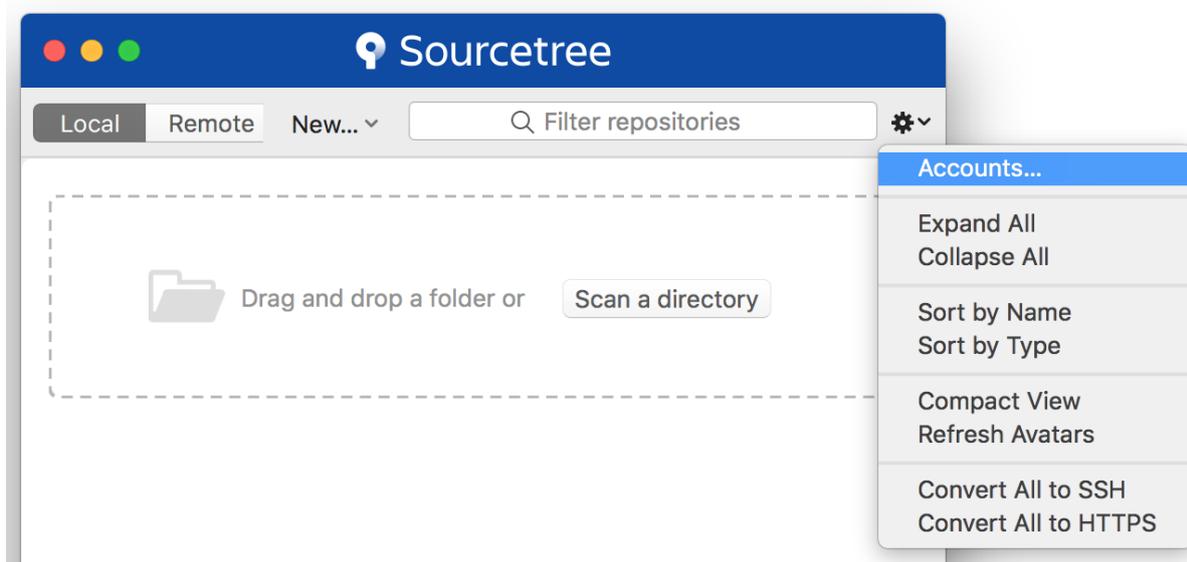
Set up Sourcetree

1. Set up Sourcetree
2. Create a personal repository for the tutorial
3. Clone your repository and manage files locally
4. Commit and push changes

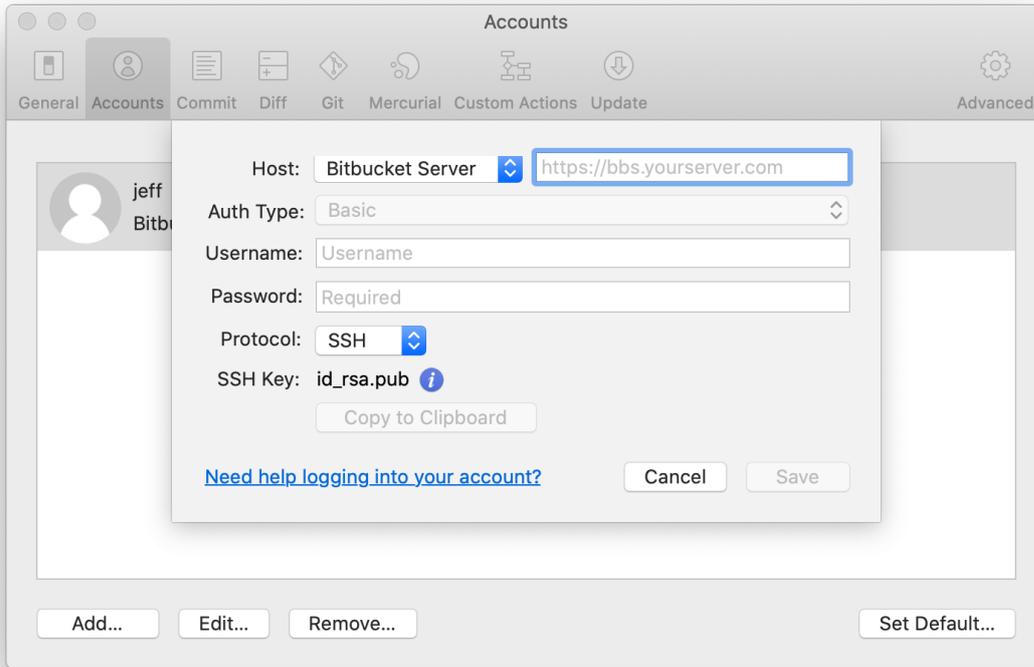
Sourcetree provides you with an interface that gives you the same capabilities you have with Git without the need to use the command line. If you prefer to use Git from the command line, feel free to [skip this step](#).

Install Sourcetree for Bitbucket Data Center and Server

1. Select the button for downloading Sourcetree from the [Sourcetree website](#).
2. Double-click the downloaded file to open it.
3. Install Sourcetree as you would any other installation.
4. Open Sourcetree, select the gear icon and then select **Accounts**.



5. Select **Add** from the **Accounts** tab.
6. After you select a **Host**, enter your hosting details. Keep the default **Auth Type** and select **Connect Account** to enter your Bitbucket credentials.



When you enter your account details, you can choose whether you prefer to connect with HTTPS or SSH. For information about setting up SSH for your account, see [Set up an SSH key](#).

7. Select **Skip Setup** from the **Clone your first repository** (you'll do this from within Bitbucket for this tutorial).

[Next step](#)

Create a personal repository for the tutorial

1. [Set up Sourcetree](#)
2. Create a personal repository for the tutorial
3. Clone your repository and manage files locally
4. Commit and push changes

In this step you will create a personal repository in Bitbucket Data Center and Server to use to keep track of your work for the Teams in Space website.

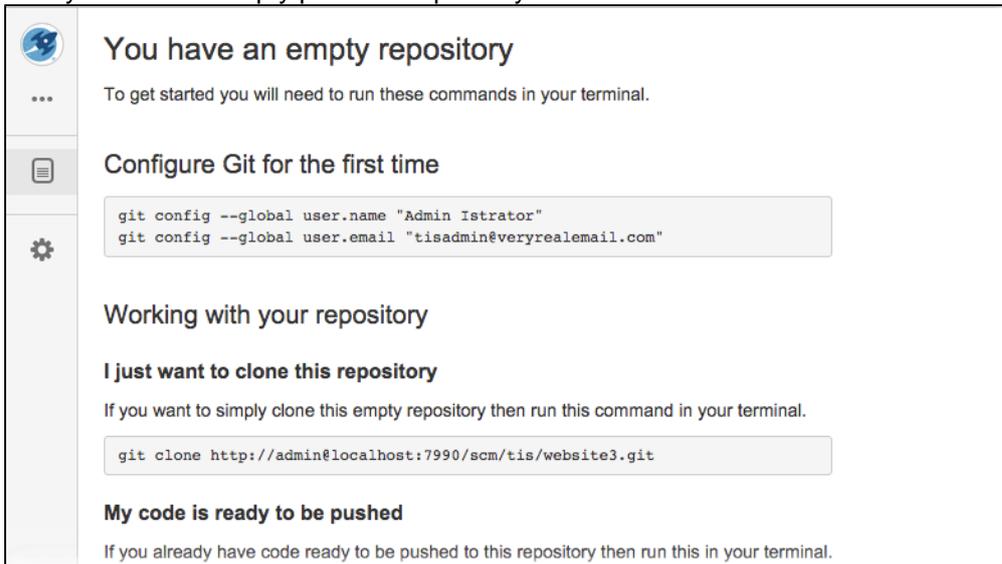
Personal repositories can be used for storing private files or starting your own project and are not visible to other users by default, but you can open access to these repositories whenever you want.

Create a personal repository in Bitbucket

1. From within a project, click **Create repository**.



2. Name your repository *Website*, then click **Create**.
Now you have an empty personal repository.



[Next step](#)

Clone your repository and manage files locally

1. [Set up Sourcetree](#)
2. [Create a personal repository for the tutorial](#)
3. Clone your repository and manage files locally
4. Commit and push changes

In this step you will clone your personal repository to your local computer. Cloning your repository locally creates a file directory on your computer that will keep in synch with your online repository.

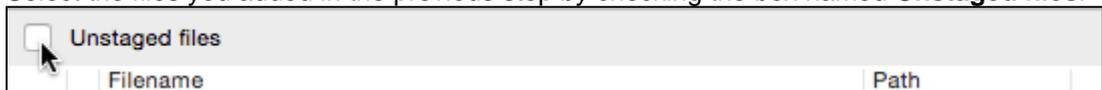
Making changes to live source files makes your website vulnerable to user errors. Since we all make mistakes, we instead clone the source files locally and make our changes on our own computer where we can first test that our changes won't break things in the process. Once we verify things are as they should be we then can push our changes to the live source files (usually a master branch). From there, others can pull in our changes to their local copy, and update files of the website.

1. Clone your personal repository using Sourcetree (or the [command line](#))
 - a. On the side navigation, click **Clone**, then **Clone in Sourcetree** to create a local directory where you can store the website files.



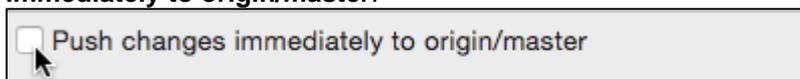
This opens the *Clone New* dialog in Sourcetree.

- b. Within Sourcetree, choose the appropriate destination for your personal repository, then click **Clone**.
You'll arrive at the empty directory in Sourcetree, and an empty directory named *website* was created on your local computer.
2. [Download the source files](#) and unzip them into the empty directory you just created.
 3. Add the files to your personal repository using Sourcetree (or the [command line](#)).
 - a. Select the files you added in the previous step by checking the box named **Unstaged files**.

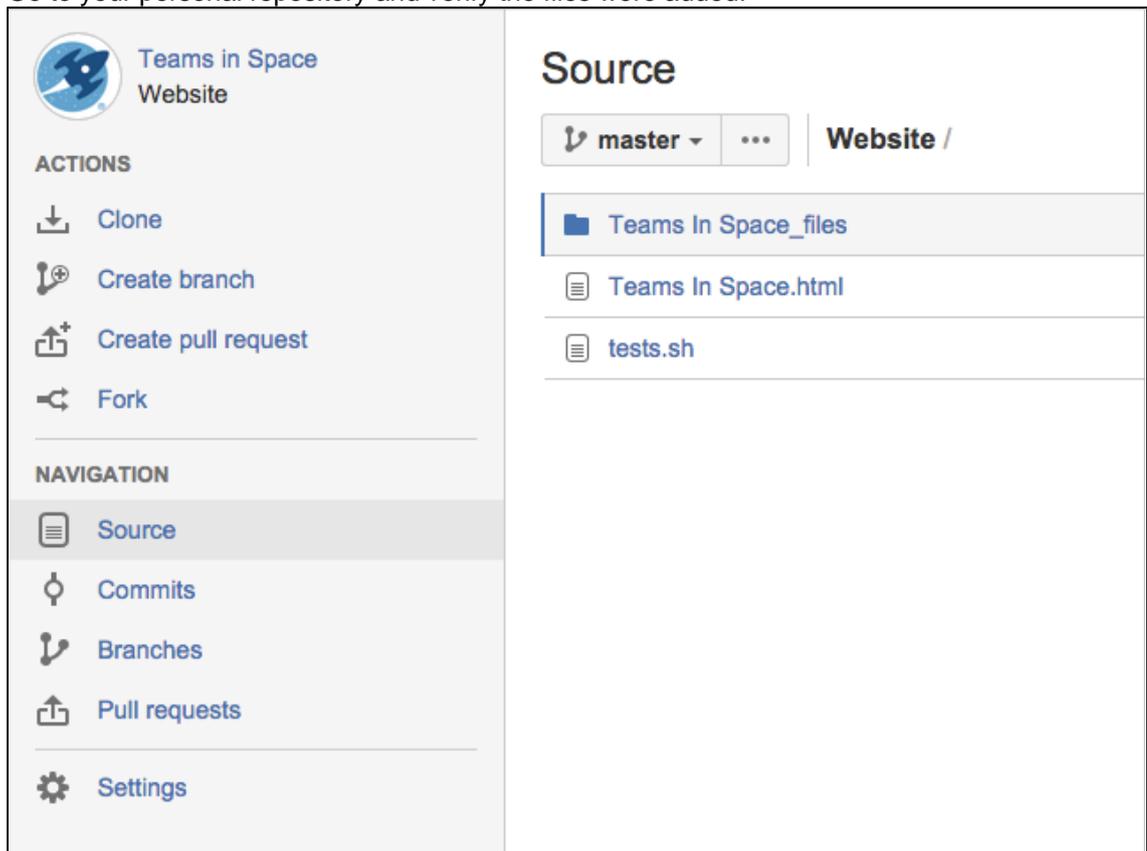


The files then appear in the Staged files pane.

- b. Click **Commit**, add a message in the comment box, and check the box **Push changes immediately to origin/master**.



- c. Go to your personal repository and verify the files were added.



[Next step](#)

Do it from the command line

1. Clone your personal repository from the command line. You can also copy the command directly from your empty repository. Look under **Working with your repository**. From a terminal window, run these commands

```
cd ~
git clone http://<username>@<Bitbucket Server URL>/scm/<project key>/website.git
```

cd ~	Change directory to your home directory
git clone	Command that copies the contents of the repository
<username>	Is the username you use to log in to the instance
<Bitbucket Server URL>	The URL for your Bitbucket Server instance
<project key>	The project key where your personal repository is
website.git	The name of your personal repository

- This creates an empty Git repository named *TISwebsite*
2. Add the files to your personal repository from the command line. From a terminal window

```
cd existing-project
git init
git add --all
git commit -m "Initial Commit"
git remote add origin http://<Bitbucket Server URL>/scm/tis/website.git
git push -u origin master
```

<code>cd existing-project</code>	Change to the directory where you unzipped the files
<code>git init</code>	Initialize the Git repository
<code>git add --all</code>	Adds the files to the repository
<code>git commit -m "Initial Commit"</code>	Adds a comment to the commit
<code>git remote add origin <url></code> <code>git push -u origin master</code>	Adds the remote repository and pushes your files to the master branch

[Next step](#)

Commit and push changes

1. [Set up Sourcetree](#)
2. [Create a personal repository for the tutorial](#)
3. [Clone your repository and manage files locally](#)
4. Commit and push changes

In this step you are going to make some changes to the HTML files added to your repository in Step 3. Once you make the changes and commit them you can add them to your repository on Bitbucket Data Center and Server. It's not enough to just make your changes, you have to share them with the world!

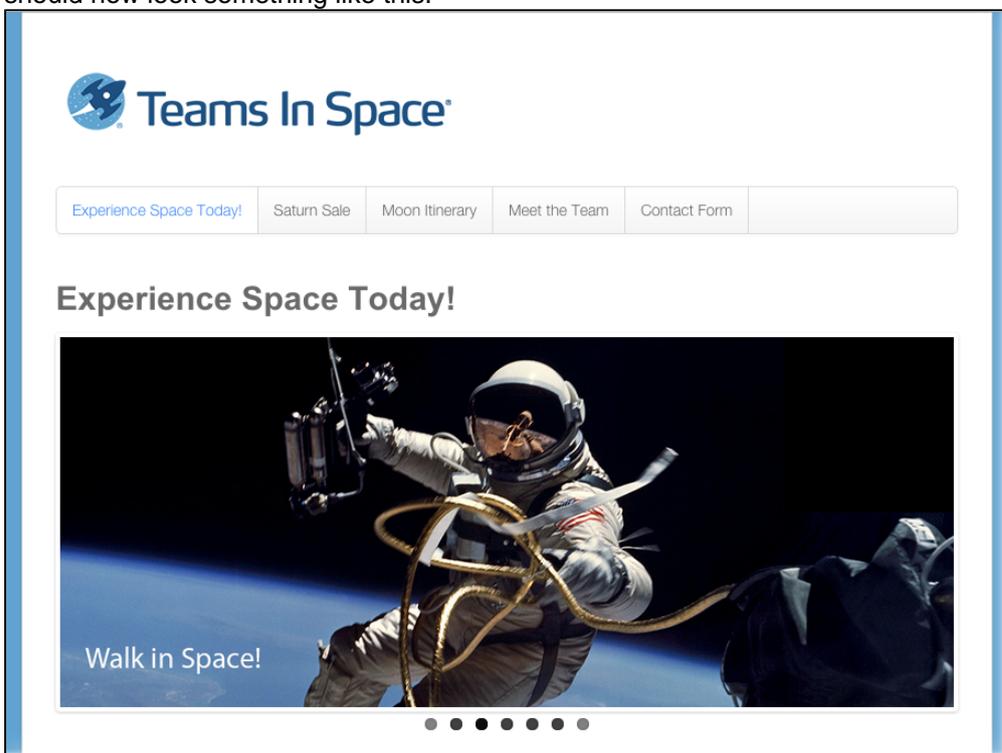
Make the changes

This step explains how to make a simple change in a source file that you'll then commit locally and push to your personal repository.

1. Open the `Teams in Space.html` file in a text editor.
2. Find the `<h3>Main Menu</h3>` line. Within that menu, add another list item to add a link to the itinerary on the main menu. (You can copy this code and add it in your HTML file).

```
<li id="menu-item-65" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-65"><a href="http://localhost:2431/?page_id=60">Moon Itinerary</a></li>
```

3. Save the file.
4. Check your work by going to the `Teams in Space.html` file and opening it in a web browser. It should now look something like this.



Commit and push the changes

Once you've made the changes and verified they work, you'll now commit the changes and push them to your repository.

1. Commit the changes using Sourcetree (or the command line).
 - a. From Sourcetree, click on **Working Copy** in the upper-left. In the Unstaged files pane on the bottom, you should see the `Teams in Space.html` file.
 - b. Select the checkbox to left of the file. The file moves to the **Staged files** pane.
 - c. Click Commit in the upper-left. The Commit dialog opens at the bottom.
 - d. Enter a commit message in the text field (something like "This is my first commit!" would do).
2. Push the changes to the repository.
 - a. There is now an indicator within the Branches field on the left that there are changes to push, as well as on the Push button on the top toolbar.



- b. Click the **Push** button, and make sure the master branch is selected, then click **Ok** to push the changes.
- c. You can verify the changes were pushed by going to the repository and clicking on **Commits**.

Do it from the command line

Commit the files you changed to your personal repository from the command line.
From a terminal window

```
cd website
git commit -m "Website changes"
git push -u origin master
```

Git resources

Get Git

See [Installing and upgrading Git](#)

On this page:

- [Get Git](#)
- [Learning Git](#)
- [Getting started](#)
- [Git cheat sheets and other resources](#)
- [Git .mailmap](#)

Learning Git

[Git Tutorials and Training](#)

[Basic Git commands](#)

Getting started

One "gotcha" when starting with Git is the way in which it pushes branches by default. On older versions of Git, pushing without arguments would push *all* branches that have the same name both locally and remotely. This can result in unexpected behavior if you have old branches that complain when the remote branch is updated. It can even be quite dangerous if you do a force push and it reverts changes on the server. You can see the current value by running:

```
git config push.default
```

If this value is blank or 'matching', it is our recommendation that you reconfigure it to use 'upstream'.

```
git config --global push.default upstream
```

There has been some [discussion](#) around changing the default behavior of Git.

Git cheat sheets and other resources

<http://rogerdudler.github.com/git-guide/>

<http://byte.kde.org/~zrusin/git/git-cheat-sheet-medium.png>

<http://nvie.com/posts/a-successful-git-branching-model/>

<http://zrusin.blogspot.com.au/2007/09/git-cheat-sheet.html>

<http://ndpsoftware.com/git-cheatsheet.html#loc=workspace;>

<http://blog.fournova.com/2011/06/git-cheat-sheet/>

<http://jan-krueger.net/development/git-cheat-sheet-extended-edition>

Git .mailmap

The Git `.mailmap` feature is useful locally, and in Bitbucket Data Center and Server repositories, to map multiple commit identities to the one Bitbucket user this can be used to tidy up your Git histories.

The [Git documentation](#) for `.mailmap` has configuration details (see the "MAPPING AUTHORS" section).

Basic Git commands

Here is a list of some basic Git commands to get you going with Git.

For more detail, check out the [Atlassian Git Tutorials](#) for a visual introduction to Git commands and workflows, including examples.

Git task	Notes	Git commands
Tell Git who you are	Configure the author name and email address to be used with your commits. Note that Git strips some characters (for example trailing periods) from <code>user.name</code> .	<pre>git config --global user.name "Sam Smith" git config--globaluser.email sam@example.com</pre>
Create a new local repository		<pre>git init</pre>
Check out a repository	Create a working copy of a local repository:	<pre>git clone /path/to/repository</pre>
	For a remote server, use:	<pre>git clone username@host:/path/to/repository</pre>
Add files	Add one or more files to staging (index):	<pre>git add <filename> git add *</pre>
Commit	Commit changes to head (but not yet to the remote repository):	<pre>git commit -m "Commit message"</pre>
	Commit any files you've added with <code>git add</code> , and also commit any files you've changed since then:	<pre>git commit -a</pre>
Push	Send changes to the master branch of your remote repository:	<pre>git push origin master</pre>
Status	List the files you've changed and those you still need to add or commit:	<pre>git status</pre>
Connect to a remote repository	If you haven't connected your local repository to a remote server, add the server to be able to push to it:	<pre>git remote add origin <server></pre>
	List all currently configured remote repositories:	<pre>git remote -v</pre>

Branches	Create a new branch and switch to it:	<code>git checkout -b <branchname></code>
	Switch from one branch to another:	<code>git checkout <branchname></code>
	List all the branches in your repo, and also tell you what branch you're currently in:	<code>git branch</code>
	Delete the feature branch:	<code>git branch -d <branchname></code>
	Push the branch to your remote repository, so others can use it:	<code>git push origin <branchname></code>
	Push all branches to your remote repository:	<code>git push --all origin</code>
	Delete a branch on your remote repository:	<code>git push origin :<branchname></code>
Update from the remote repository	Fetch and merge changes on the remote server to your working directory:	<code>git pull</code>
	To merge a different branch into your active branch:	<code>git merge <branchname></code>
	View all the merge conflicts: View the conflicts against the base file:	<code>git diff</code> <code>git diff --base <filename></code>
	Preview changes, before merging:	<code>git diff <sourcebranch> <targetbranch></code>
	After you have manually resolved any conflicts, you mark the changed file:	<code>git add <filename></code>
Tags	You can use tagging to mark a significant changeset, such as a release:	<code>git tag 1.0.0 <commitID></code>
	CommitID is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:	<code>git log</code>
	Push all tags to remote repository:	<code>git push --tags origin</code>
Undo local changes	If you mess up, you can replace the changes in your working tree with the last content in head: Changes already added to the index, as well as new files, will be kept.	<code>git checkout -- <filename></code>

	Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this:	<pre>git fetch origin git reset --hard origin/master</pre>
Search	Search the working directory for <code>foo()</code> :	<pre>git grep "foo()"</pre>

Get started with Git

Atlassian Bitbucket Data Center and Server is the Git repository management solution for enterprise teams. It allows everyone in your organization to easily collaborate on your Git repositories.

This page will guide you through the basics of Bitbucket. By the end you should know how to:

- [Create accounts for your collaborators, and organize these into groups with permissions.](#)
- [Create a project and set up permissions.](#)
- [Create repositories, and know the basic commands for interacting with them.](#)

Assumptions

This guide assumes that you don't have prior experience with Git. But we do assume that:

- You have Git version 1.7.6 or higher installed on your local computer.
- You are using a [supported browser](#).
- You have Bitbucket installed and running. If you haven't, see [Getting started](#).

Please read [Git resources](#) or check out our [Git tutorials](#) for tips on getting started with Git.

Add users to Bitbucket and grant permissions

The first thing you can do is add collaborators.

To add users

1. Go to the **Bitbucket** administration area by clicking the cog , then click **Users** in the Admin screen (under Accounts):
2. Click **Create user** to go directly to the user creation form.
3. Once you've created a user, click **Change permission** to set up their access permissions.

There are 4 levels of user authentication:

- **System Administrator** can access all the configuration settings of the instance.
- **Administrator** same as System Admins, but they can't modify file paths or the instance settings.
- **Project Creator** can create, modify and delete projects.
- **Bitbucket User** active users who can access **Bitbucket**.

See [Users and groups](#) for more information about authentication.

See [External user directories](#) if you have existing user identities you wish to use with **Bitbucket**.

Create your first project and share it with collaborators

Create your project

The next thing you do, is create a project. You'll add repositories to this project later.

Go to 'Projects' and click **Create project**. Complete the form and submit it to create your new project. See [Creating projects](#) for more information.

Open project access to others



Make the most of Bitbucket

[Automate your Bitbucket Server deployments](#)

[Bitbucket Data Center for enterprises](#)

[Run Bitbucket in AWS](#)

Learn Git

[Get started with Git](#)

[Git resources](#)

[Be a Git guru](#)

Bitbucket in action

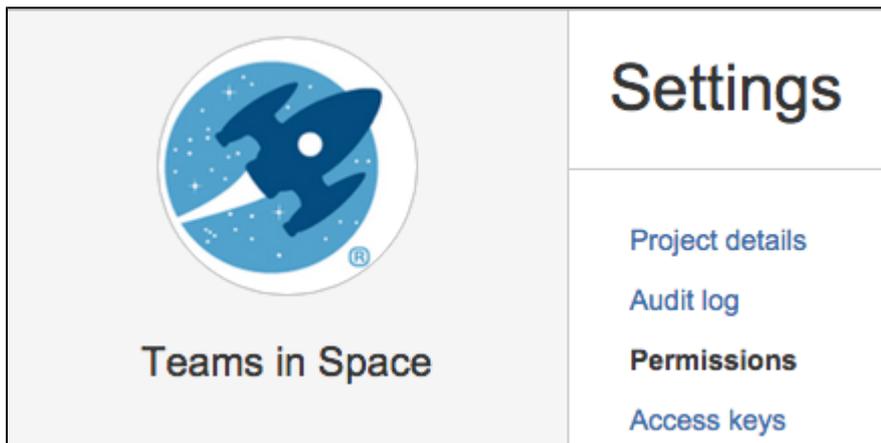
[Ecommerce speed](#)

[NASA rockets](#)

[Orbitz switches to Git](#)

If you are a project administrator, you can grant project permissions to other collaborators.

Click **Settings** then **Permissions** for the project:



The 'Project permissions' page allows you to add users and groups to a project you've already created.

There are 3 levels of project access:

- **Admin** can create, edit and delete repositories and projects, and configure permissions for projects.
- **Write** can push to and pull from all the repositories in the project.
- **Read** can only browse code and comments in, and pull from, the repositories in the project.

See [Using project permissions](#) for more information.

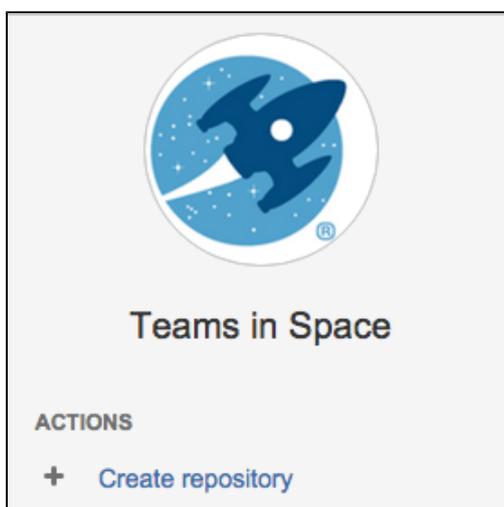
Create a repository and get your code into **Bitbucket**

Create a repository

If you are a project administrator, you can create repositories in the project.

Once a repository is created, the project permissions are applied to the repository. That means all repositories created in a project share the same access and permission settings. If you already have a Git project you'd like to use, see [Importing code from an existing project](#).

Click **Create repository** to open the repository creation form:



Once submitted you will be taken directly to your repository homepage. As there is no content in your repository yet, you'll see some instructions to help you push code to your repository. See [Creating repositories](#) for more information.

Clone and push

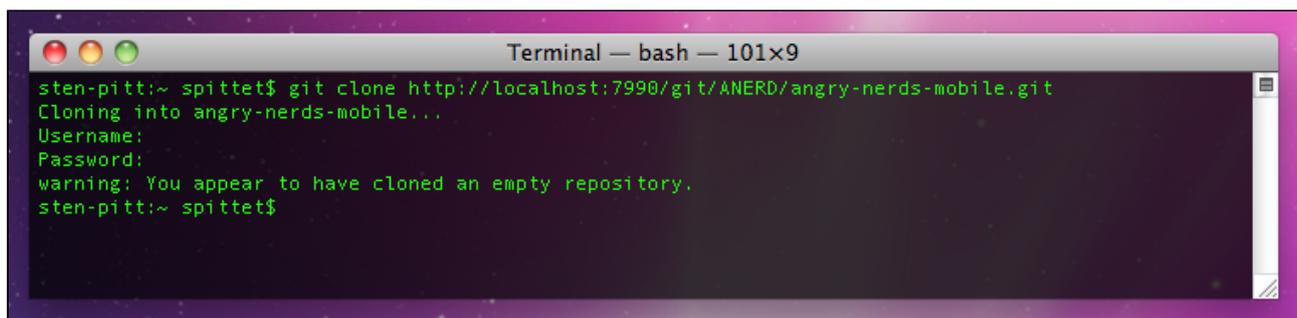
This section describes how to [clone the repository you just created](#) and then [push a commit](#) back to it. You can see the clone URL to use at the top right of the screen. [SSH access](#) may be available.

In a terminal, run the following command (replace `<bitbucketURL>` with the URL for your instance of **Bitbucket**):

```
git clone <bitbucketURL>/git/<projectname>/<reponame>.git
```

Use your **Bitbucket** username and password.

The result in your terminal should be similar to what you can see in the screenshot below.



You should now have a new empty directory tracked by Git, in the user space of your local machine. Let's add some content and push it back to **Bitbucket**.

In your `<reponame>` directory, create a text file named `helloworld.txt` and write "Hello World" in it.

Now run the following command in your terminal

```
cd <reponame>
git add .
git commit -m "My first commit"
git push origin master
```

If everything went fine, when you refresh the **Bitbucket** screen, you will see that the homepage of your repository has been replaced with a file browser showing you a link to `helloworld.txt`.

There you go, you're ready to get coding with your collaborators.

For more information about getting your code into **Bitbucket**, see [Importing code from an existing project](#).

Check out our [Git tutorials and training](#) for more information, and have a look at this list of [basic Git commands](#) that you will probably use often.

Use Bitbucket Data Center and Server

Bitbucket is the on-premises Git repository management solution for enterprise teams. It allows everyone in your organization to easily collaborate on your Git repositories.

This section describes the essentials of using Bitbucket.

If you are setting up Bitbucket see the [Install or upgrade Bitbucket](#) section. If you want to configure Bitbucket, see the [Administer Bitbucket Data Center and Server](#) section.

See [Get started with Git](#) for an overview of how to work with Bitbucket.

Related pages:

- [Git Tutorials and Training](#)
- [Git resources](#)
- [Administer Bitbucket Data Center and Server](#)
- [FAQ](#)

Working with projects

Bitbucket manages related repositories as projects. Find out how to [set up projects](#) and then [give your teams access](#) to those.

Working with repositories

If you have existing projects that you want to manage in Bitbucket, then you'll want to read [Importing code from an existing project](#).

See also:

- [Creating repositories](#)
- [Controlling access to code](#)
- [Pull requests](#)
- [Working with Git LFS Files](#)
- [Compare changes in Bitbucket Server](#)

Git resources

For those who are new to using Git:

- [Pull requests](#)
- [Basic Git commands](#)
- [Permanently authenticating with Git repositories](#)

Importing code from an existing project

This page describes the methods you can use to import code from existing projects into Bitbucket Data Center and Server. When creating a new repository, you can import code from an existing project into Bitbucket using either the terminal or the web interface.

If you're using Bitbucket Cloud, take a look at our [Import or convert code from an existing tool](#) page for more information.

Import code using the web interface

Introduced in [Bitbucket Server 4.9](#), you can import code and its version/branching history into Bitbucket from existing Git projects hosted with Bitbucket Cloud, GitHub, GitHub Enterprise, or a standalone Git repository using the web interface.

To start importing code

1. While viewing a project within Bitbucket click **Import repository** in the sidebar.
2. Select a source to import code from, provide the required information, then click **Connect**.
 - a. **For Bitbucket Cloud**, include the Username and [App password](#) for the account to import from, and ensure read access for account, team, project, and repository is enabled.
 - b. **For GitHub**, include the Username and [Personal access token](#) for the account to import from, and ensure the `repo` and `read:orgscopes` are enabled.
 - c. **For GitHub Enterprise**, provide the Server URL, Username, and [Personal access token](#) for the account to import from, and ensure the `repo` and `read:orgscopes` are enabled.
 - d. **For a single Git repository**, provide the Clone URL, and Username and Password (if required).
3. Choose which repositories to import.
 - a. **All repositories** imports all the repositories owned by the account provided.
 - b. **Select repositories** allows you choose specific repositories to import.
4. Click **Import**.

Once importing completes, you can refresh the project page to see the imported repositories.

Import code using the terminal

Import code from an existing project using the terminal by first cloning the repository to your local system and then pushing to an empty Bitbucket repository.

Import an existing, unversioned code project

If you have code on your local machine that is not under source control, you can put it under source control and import it into Bitbucket.

Assuming you have Git installed on your local machine, then:

1. Locally, change to the root directory of your existing source.
2. [Initialize the project](#) by running the following commands in the terminal:

```
git init
git add --all
git commit -m "Initial Commit"
```

3. Log into Bitbucket and [create a new repository](#).
4. Locate the clone URL in the nav panel on the left (for example: `https://username@your.bitbucket.domain:7999/yourproject/repo.git`).
5. Push your files to the repository by running the following commands in the terminal (change the URL accordingly):

On this page

- [Import code using the web interface](#)
- [Import code using the terminal](#)
- [Mirror an existing Git repository](#)

```
git remote add origin https://username@your.bitbucket.domain:7999/yourproject/repo.git
git push -u origin master
```

6. Done! Your repository is now available in Bitbucket.

Import an existing Git project

You can import your existing Git repository into an empty repository in Bitbucket. When you do this, Bitbucket maintains your commit history.

1. Check out the repository from your existing Git host. Use the `--bare` parameter:

```
git clone --bare https://username@bitbucket.org/exampleuser/old-repository.git
```

2. Log into Bitbucket and create a new repository (we've called it `repo.git` in this example).
3. Locate the clone URL in the nav panel on the left (for example: `https://username@your.bitbucket.domain:7999/yourproject/repo.git`).
4. Add Bitbucket as another remote in your local repository:

```
cd old-repository
git remote add bitbucket https://username@your.bitbucket.domain:7999/yourproject/repo.git
```

5. Push all branches and tags to the new repository:

```
git push --all bitbucket
git push --tags bitbucket
```

6. Remove your temporary local repository:

```
cd ..
rm -rf old-repository
```

Mirror an existing Git repository

You can mirror an existing repository into a repository hosted in Bitbucket.

1. Check out the repository from your existing Git host. Use the `--mirror` parameter:

```
git clone --mirror https://username@bitbucket.org/exampleuser/repository-to-mirror.git
```

2. Log into Bitbucket and create a new repository (we've called it `repo.git` in this example).
3. Locate the clone URL in the nav panel on the left (for example: `https://username@your.bitbucket.domain:7999/yourproject/repo.git`).
4. Add Bitbucket as another remote in your local repository:

```
git remote add bitbucket https://username@your.bitbucket.domain:7999/yourproject/repo.git
```

5. Then push all branches and tags to Bitbucket:

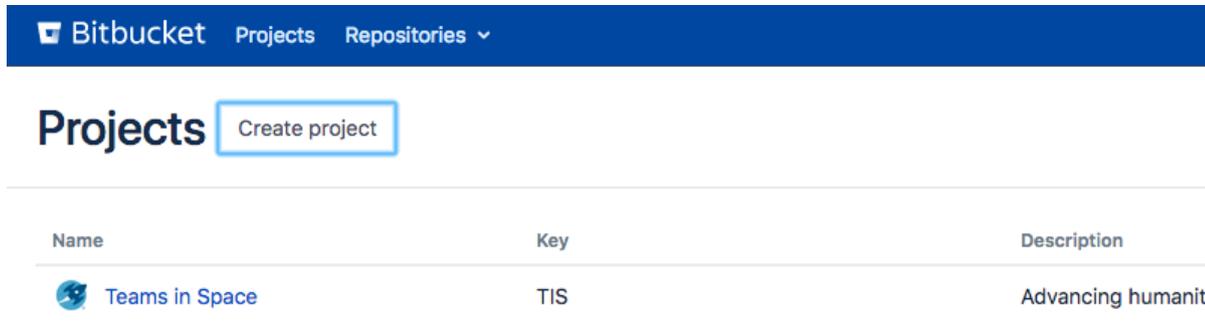
```
git push --all bitbucket
git push --tags bitbucket
```

6. Use `git fetch --prune origin` ('prune' will remove any branches that no longer exist in the remote) followed by the `git push` commands from step 5 to update the Bitbucket mirror with new changes from the upstream repository.

Creating projects

Projects in Bitbucket Data Center and Server allow you to group repositories and to [manage permissions](#) for them in an aggregated way.

To create a project, click on **Create project**:



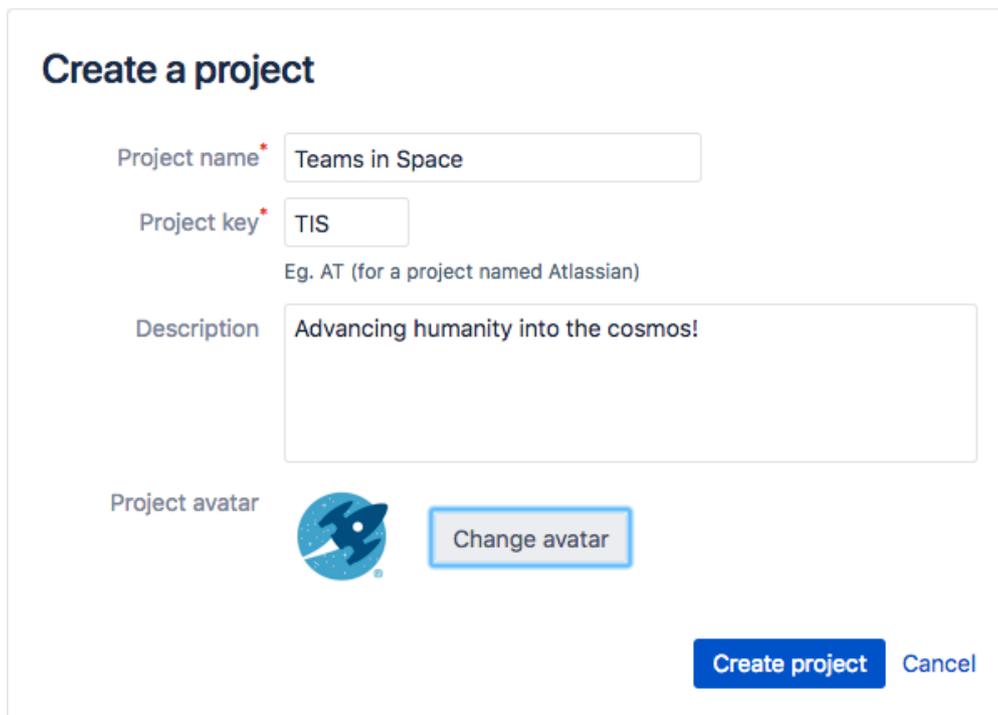
The screenshot shows the Bitbucket interface. At the top, there is a navigation bar with 'Bitbucket', 'Projects', and 'Repositories' with a dropdown arrow. Below this, the 'Projects' section is visible, featuring a 'Create project' button. A table below lists existing projects:

Name	Key	Description
 Teams in Space	TIS	Advancing humanity

Fill out the form. We recommend that you use a short project key. It will be used as an identifier for your project and will appear in the URLs.

Optionally, you can choose an avatar for the project. This is displayed throughout **Bitbucket** and helps to identify your project.

Click **Create project** when you're done.



The 'Create a project' form contains the following fields and elements:

- Project name***: Input field containing 'Teams in Space'.
- Project key***: Input field containing 'TIS'. Below it, a hint reads 'Eg. AT (for a project named Atlassian)'.
- Description**: Text area containing 'Advancing humanity into the cosmos!'.
- Project avatar**: A circular icon of a rocket ship. To its right is a 'Change avatar' button.
- At the bottom right, there are two buttons: 'Create project' (highlighted in blue) and 'Cancel'.

You'll want to add repositories to the project. See [Creating repositories](#) for details.

Creating repositories

Repositories in Bitbucket Data Center and Server allow you to collaborate on code with your team.

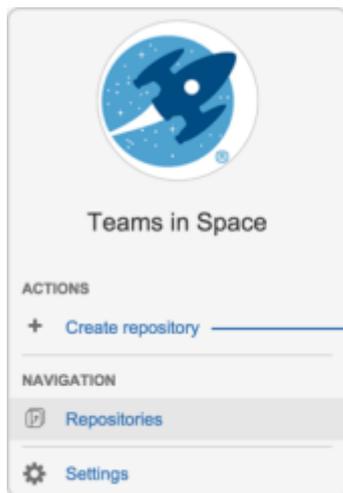
In order to create repositories you need to have [Project Admin permission](#) for the project to which you want to add a repository.

When a repository in Bitbucket is created, the project permissions are applied to the repository. That means all repositories created in a project share the same access and permission settings.

Go to the project and click **Create repository** to open the repository creation form:

Related pages:

- [Creating personal repositories](#)
- [Using repository permissions](#)
- [Creating projects](#)
- [Importing code from an existing project](#)



Get started

Once submitted you will be taken directly to your repository homepage.

There won't be any content in your repository yet, so you'll see some instructions to help you push code to your repository:

You have an empty repository

To get started you will need to run these commands in your terminal.

Configure Git for the first time

```
git config --global user.name "charlie"  
git config --global user.email "ccurti@atlassian.com"
```

Working with your repository

I just want to clone this repository

If you want to simply clone this empty repository then run this command in your terminal.

```
git clone https://charlie@bitbucketdc/scm/tp1/tp1.git
```

My code is ready to be pushed

If you already have code ready to be pushed to this repository then run this in your terminal.

```
cd existing-project  
git init  
git add --all  
git commit -m "Initial Commit"  
git remote add origin https://charlie@bitbucketdc/scm/tp1/tp1.git  
git push -u origin master
```

My code is already tracked by Git

If your code is already tracked by Git then set this repository as your "origin" to push to.

```
cd existing-project  
git remote set-url origin https://charlie@bitbucketdc/scm/tp1/tp1.git  
git push -u origin --all  
git push origin --tags
```

You will find your clone URL in the lefthand sidebar of the repository homepage. You can use this URL and share it with other people.

Let other people collaborate with you

In order to grant users access to this repository you have to set up permissions at the parent project level. More information is available on [Creating projects](#). You can also create a [Contributor's guidelines](#) file to add to your repository, where you can add info on what you'd like your contributors to do when contributing to your repository.

Creating personal repositories

Bitbucket Data Center and Server allows you to create personal repositories, unrelated to other projects, that you can use for such purposes as storing private snippets of work, kick-starting your own project, or contributing a bug-fix for a project you are not a member of.

By default, personal repositories are not visible to other Bitbucket users (unless they are a Bitbucket [system administrator](#)). However, you can:

- use [repository permissions](#) to open up access to other Bitbucket users and groups, for collaboration or review.
- allow [public access](#) (read-only) to your project, for anonymous users.

You can create personal repositories in two ways:

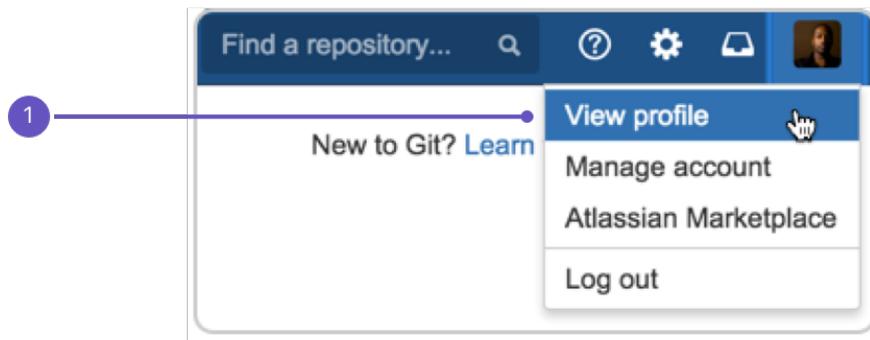
- [Directly](#), from your profile.
- By [forking](#) another repository.

Your personal repositories are listed on the **Repositories** tab of your profile page. Every Bitbucket user can see your profile page, but they can only see those repositories that you have given them permission to view.

Directly creating a personal repository

You can create a personal repository at any time from your profile:

1. Choose **View profile** from your user menu in the header.



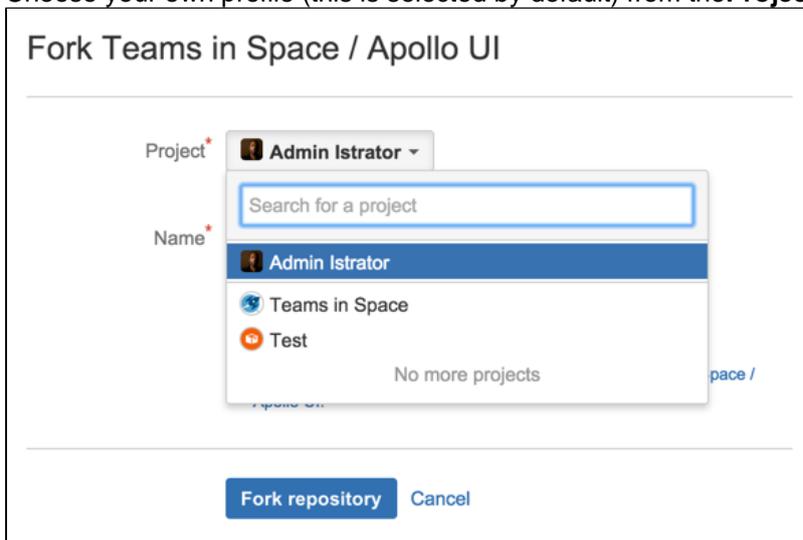
2. Click **Create repository**.
3. Set [repository permissions](#) on the new repository, if required.

Forking another repository

You can create a personal fork of any other repository for which you have [permission](#):

1. Go to the repository that you wish to fork.
2. Click **Fork** in the sidebar.

3. Choose your own profile (this is selected by default) from the **Project** list:



4. Click **Fork repository**.
5. Set [repository permissions](#) on the new repository, if required.

Using repository hooks

Hooks let you to extend what Bitbucket Data Center and Server does every time a repository changes, allow you to customize your team's workflow, and enable you to integrate with other systems.

You can configure a hook to run automatically every time a particular event occurs in a repository, for instance when code is pushed or a pull request is merged.

Bitbucket supports two types of hooks, [pre-receive](#) and [post-receive](#) hooks. Hooks are installed by system administrators and can be enabled for all repositories in a project, or for an individual repository.

On this page:

- [Pre-receive hooks](#)
- [Post-receive hooks](#)
- [Configure hooks for all repositories in a project](#)
- [Configure hooks for an individual repository](#)
- [Inherited hook configurations](#)
- [Add a new hook](#)
- [Create a hook](#)

Pre-receive hooks

Pre-receive hooks allow you to control which commits go into your repository before pushes are committed or pull requests are merged. For example, a pre-receive hook can reject pushes to the repository if certain conditions are not fulfilled. It could prevent force pushes to the repository or check whether all commits contain a valid Jira application issue key.

Default pre-receive hooks

Bitbucket comes with some pre-receive hooks installed by default that are disabled, but can be enabled at the project level for all repositories in a project, or for individual repositories.

The default hooks that come with Bitbucket are:

- **Reject Force Push**- rejects all force pushes to a repository.
- **Verify Commit Signature**- rejects commits and tags without a verified GPG signature.
- **Verify Committer**- rejects commits not committed by the user pushing to a repository.

Post-receive hooks

Post-receive hooks run after commits are processed, and are typically used to update external services or send notifications. For example, the [Web Post Hooks Plugin](#) can send a message to a chat client or notify a continuous integration server changes.

For releases prior to 5.0, Bitbucket supported two types of post-receive hook:

- [PostReceiveHooks](#) used to map to Git's `post-receive` hooks. They ran on the Bitbucket instance after a push.
- [AsyncPostReceiveRepositoryHooks](#), was executed by the Bitbucket instance.

From 5.0 onwards, these are now both deprecated and have been replaced by:

- [PostRepositoryHooks](#), which gets called for both PR merge and post-receive.

Configure hooks for all repositories in a project

Enabling (or disabling) hooks at the project level changes hooks for repositories set to inherit project settings. If you previously changed hooks for an individual repository, that repository's configuration will not change when configuring hooks at the project level.

To enable (or disable) hooks for repositories in a project (requires project admin permissions):

1. Go to **Project settings > Hooks**.
2. Click the toggle by the hook to enable (or disable) it.

Hooks for repositories set to [Inherited](#) in the project will now reflect this new configuration. Hooks explicitly configured at the repository level will not be affected.

Configure hooks for an individual repository

Configuring hooks at the repository level will override any checks configured at the project level. If you have not configured hooks for an individual repository it will inherit hooks configured at the project level.

To enable (or disable) hooks for a single repository (requires repository admin permissions):

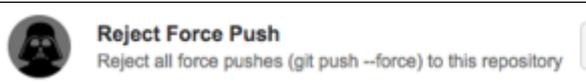
1. Go to **Repository settings > Hooks**.
2. Use the drop menu to the right of the hook to set it.
 - a. [Inherited](#) - uses the configuration set at the project level.
 - b. Enabled - enforces the conditions of the hook.
 - c. Disabled - ignores the conditions of the hook.

Once set, any changes made to hook configuration at the project level will be ignored for this repository because it was changed independent of the project configuration.

Inherited hook configurations

By default, Bitbucket comes with hooks disabled at the project and repository level. Unless hooks were configured at the repository level, enabling or disabling hooks at the project level inherits the configuration at the repository level.

For example, if you enabled the *Reject Force Push* hook for a project, and a repository hook configuration was unchanged, each repository would have the *Reject Force Push* hook enabled.

<p>Hook disabled, project level</p> 	<p>Hook disabled, repository level</p> 
<p>Hook enabled, project level</p> 	<p>Hook enabled, repository level</p> 

Now suppose you decide that the *Reject Force Push* hook isn't appropriate for one specific repository. You can change that individual repository's hooks independent of how it's configured at the project level. Any changes made to hook configuration at the project level for the *Reject Force Push* will be ignored for this repository, because it was changed independent of the project configuration.

<p>Hook enabled, project level</p> 	<p>Hook disabled, repository level</p> 
---	--

Add a new hook

Additional hooks can be installed by system administrators and can also be enabled for all repositories in a project, or for individual repositories.

To add hooks from the Atlassian Marketplace(requires system admin permission):

1. Go to **Project settings**>**Hooks**.
2. Click **Add hook**.
3. Search for a hook to add, and click **Install**.

Once you add a new hook, you can enable (or disable) it in the same way as the default hooks.

Create a hook

You can also write your own hooks. Here are some useful resources to help you get started.

- [Repository hooks](#)
- [Repository hook plugin module](#)
- [Blog post about hooks for Bitbucket](#)
- [Video demo on how to get started on Bitbucket hooks](#)

Permanently authenticating with Git repositories

In addition to SSH, Bitbucket Data Center and Server supports HTTP or HTTPS for pushing and pulling from managed Git repositories. However, Git does not cache the user's credentials by default, so you need to re-enter them each time you perform a clone, push, or pull.

This page describes two methods for permanently authenticating with Git repositories so that you can avoid typing your username and password each time you are pushing to or pulling from Bitbucket.

On this page

- [Using credential caching](#)
- [Using the .netrc file](#)

Using credential caching



You need Git 1.7.9 or above to use the HTTPS Credentials Caching feature.

Windows

On Windows you can use the application [git-credential-winstore](#).

1. [Download the software](#).
2. Run it.
3. You will be prompted for credentials the first time you access a repository, and Windows will store your credentials for use in the future.

Linux

On Linux you can use the 'cache' authentication helper that is bundled with Git 1.7.9 and higher. From the Git documentation:

This command caches credentials in memory for use by future git programs. The stored credentials never touch the disk, and are forgotten after a configurable timeout. The cache is accessible over a Unix domain socket, restricted to the current user by filesystem permissions.

Run the command below to enable credential caching. After enabling credential caching any time you enter your password it will be cached for 1 hour (3600 seconds):

```
git config --global credential.helper 'cache --timeout 3600'
```

Run the command below for an overview of all configuration options for the 'cache' authentication helper:

```
git help credential-cache
```

macOS

Follow these steps to use Git with credential caching on macOS:

1. Download the binary [git-credential-osxkeychain](#).
2. Run the command below to ensure the binary is executable:

```
chmod a+x git-credential-osxkeychain
```

3. Put it in the directory `/usr/local/bin`.
4. Run the command below:

```
git config --global credential.helper osxkeychain
```

Using the .netrc file

The `.netrc` file is a mechanism that allows you to specify which credentials to use for which server. This method allows you to avoid entering a username and password every time you push to or pull from Git, but your Git password is stored in plain text.

Warning!

- Git uses a utility called `cURL` under the covers, which respects the use of the `.netrc` file. Be aware that other applications that use `cURL` to make requests to servers defined in your `.netrc` file will also now be authenticated using these credentials. Also, this method of authentication is potentially unsuitable if you are accessing your Bitbucket instance via a proxy, as all `cURL` requests that target a path on that proxy server will be authenticated using your `.netrc` credentials.
- `cURL` will not match the machine name in your `.netrc` if it has a username in it, so make sure you edit your `.git/config` file in the root of your clone of the repository and remove the `user` and `'@'` part from any clone URL's (URL fields) that look like `https://user@machine.domain.com/...` to make them look like `http://machine.domain.com/...`

Windows

1. Create a text file called `_netrc` in your home directory (e.g. `c:\users\kannonboy_netrc`). `cURL` has problems resolving your home directory if it contains spaces in its path (e.g. `c:\Documents and Settings\kannonboy`). However, you can update your `%HOME%` environment variable to point to any directory, so create your `_netrc` in a directory with no spaces in it (for example `c:\curl-auth\`) then set your `%HOME%` environment variable to point to the newly created directory.
2. Add credentials to the file for the server or servers you want to store credentials for, using the format described below:

```
machine stash1.mycompany.com
login myusername
password mypassword
machine stash2.mycompany.com
login myotherusername
password myotherpassword
```

Linux or macOS

1. Create a file called `.netrc` in your home directory (`~/.netrc`). Unfortunately, the syntax requires you to store your passwords in plain text - so make sure you modify the file permissions to make it readable only to you.
2. Add credentials to the file for the server or servers you want to store credentials for, using the format described in the 'Windows' section above. You may use either IP addresses or hostnames, and you do *not* need to specify a port number, even if you're running Bitbucket on a non-standard port.
3. And that's it! Subsequent `git clone`, `git pull` and `git push` requests will now be authenticated using the credentials specified in this file.

Clone a repository

Cloning a repository

You can use Sourcetree, Git from the terminal, or any client you like to clone your Git repository. These instructions show you how to clone your repository using Git from the terminal.

1. From the repository, click **Clone** in the sidebar to display the Clone dialog. If there's [contributor's guidelines](#) available for the repository, have a look to make sure you're onboard with what's expected for contributions to the repository.
2. Copy the clone command (either the SSH format or the HTTPS).
If you are using the SSH protocol, ensure your public key is in Bitbucket Data Center and Server and loaded on the local system to which you are cloning.
3. From a terminal window, change to the local directory where you want to clone your repository.
4. Paste the command you copied from Bitbucket, for example:

Clone over HTTPS:

```
$ git clone https://username@bitbucket.org/teamsinspace/documentation-tests.git
```

Clone over SSH:

```
$ git clone git@bitbucket.org:teamsinspace/documentation-tests.git
```

If the clone was successful, a new sub-directory appears on your local drive. This directory has the same name as the Bitbucket repository that you cloned. The clone contains the files and metadata that Git requires to maintain the changes you make to the source files.

Cloning a mirror repository

You can use Sourcetree, Git from the terminal, or any client you like to clone your Git repository. These instructions show you how to clone a mirrored repository using Git from the terminal. Read more about [Smart Mirrors](#).

1. Navigate to the repository in Bitbucket Data Center and Server.
2. Click the **Clone** button.
3. If [mirrors are configured](#), use the **Clone from** dropdown to select the mirror closest to you. The clone URL changes.
4. Copy the clone URL (either SSH or HTTPS).
If you are using the SSH protocol, ensure your public key is correctly configured.
5. Launch a terminal window.
6. Change to the local directory where you want to clone your repository.
7. Enter `git clone` followed by the copied clone URL.

The command and clone URL together would look like this:

```
$ git clone ssh://git@bitbucket-au.example.com:7999/upstream/PROJ/repo.git
```

If the clone was successful, a new sub-directory appears on your local drive. This directory has the same name as the Bitbucket repository that you cloned. The clone contains the files and metadata that Git requires to maintain the changes you make to the source files.

Personal access tokens

Personal access tokens can be used in Bitbucket Data Center and Server in place of passwords for Git over HTTPS, or to authenticate when using the [Bitbucket REST API](#).

On this page:

- [Creating personal access tokens](#)
- [Using personal access tokens](#)

Related pages:

- [Managing personal access tokens](#)
- [Controlling access to code](#)
- [SSH access keys for system use](#)

Creating personal access tokens

To create a personal access token:

1. Go to **Profile picture** > **Manage account** > **Personal access tokens**.
2. Click **Create a token**.
3. Set the token name, [permissions](#), and [expiry](#).

Create a personal access token

Use personal access tokens in place of passwords for Git over HTTPS, or to authenticate when using the Bitbucket Server REST API. [Learn more](#).

Token details

Token name

Permissions

Tokens are like another password, so their permissions will default to the level of access you have. Because of this, it is recommended that you restrict the token's permission to the level it will need.

Projects

Repositories

This personal access token will allow the supplied third-party application to:

Pull and clone repositories

Expiry

For added security, you can set this token to automatically expire. If you set an expiry date, you won't be able to edit it once you've created the token.

Automatic expiry No Yes

Days until expiry

Permissions

Permissions restrict what a token can do. As tokens are like passwords, your tokens permissions will be set at your current level of access by default. We recommend, however, restricting your tokens permissions to only the level it will need.

Here are the permission combinations you can assign to a token:

i Repo permissions are inherited from the project permissions

A tokens repository permission must be as high as its project permission.

If you give a token *project write* permission, you cannot give it only *repository read* permissions (it must be write-level or higher).

	Project read	Project write	Project admin
Repository read	<ul style="list-style-type: none"> ✔ Pull and clone repositories 	<ul style="list-style-type: none"> ✘ Combination not possible 	<ul style="list-style-type: none"> ✘ Combination not possible
Repository write	<ul style="list-style-type: none"> ✔ Perform pull request actions ✔ Push, pull, and clone repositories 	<ul style="list-style-type: none"> ✔ Perform pull request actions ✔ Push, pull, and clone repositories 	<ul style="list-style-type: none"> ✘ Combination not possible
Repository admin	<ul style="list-style-type: none"> ✔ Perform pull request actions ✔ Update repository settings and permissions ✔ Push, pull, and clone repositories 	<ul style="list-style-type: none"> ✔ Perform pull request actions ✔ Update repository settings and permissions ✔ Push, pull, and clone repositories 	<ul style="list-style-type: none"> ✔ Perform pull request actions ✔ Update repository settings and permissions ✔ Update project settings and permissions ✔ Push, pull, clone, and fork repositories ✔ Create repositories

You can modify a tokens permissions, or revoke a token, by going to **Profile picture > Manage account > Personal access tokens**.

Expiry

For added security, when youre creating a token you can also set it to automatically expire. This is optional, but if your [administrator has made this a requirement](#) youll need to choose an expiry date thats within the limits theyve set.

Once a token has been created, its expiry date cannot be changed. You can see the expiry dates for all your tokens by going to **Profile picture > Manage account > Personal access tokens**.

Using personal access tokens

i Map one token per integration

Personal access tokens are a secure way to use scripts and to integrate external applications with Bitbucket. We recommend only mapping one token per integration. This way, if the system is compromised, you can simply revoke the token and not affect other integrations.

For Git operations, you can use your personal access token as a substitute for your password. For example, to clone using a personal access token you can enter:

```
> git clone https://bitbucketserver.com/scm/projectname/teamsinspace.git
Cloning into 'teamsinspace'...
Username for 'https://bitbucketserver.com':username
Password for 'https://username@bitbucketserver.com':MDM0MjM5NDc2MDxxxxxxxxxxxxxxxxxxxxxxxx
```

Or using Basic Auth:

```
git clone https://username:MDM0MjM5NDc2MDxxxxxxxxxxxxxxxxxxxxxxxx@bitbucketserver.com/scm/projectname/teamsinspace.git
```

In addition, for REST operations you can use Basic Auth:

```
curl -u username:MDM0MjM5NDc2MDxxxxxxxxxxxxxxxxxxxxxxxx https://bitbucketserver.com/rest/api/latest/resource/path
```

Or you can use Bearer Auth without your username:

```
curl -H 'Authorization: Bearer MDM0MjM5NDc2MDxxxxxxxxxxxxxxxxxxxxxxxx' https://bitbucketserver.com/rest/api/latest/resource/path
```

Controlling access to code

Bitbucket Data Center and Server provides the following types of permissions to allow fully customizable control of access to code.

Note that you can also:

- allow public (anonymous) access to projects and repositories. See [Allowing public access to code](#).
- use SSH keys to allow user accounts and other systems to connect securely to Bitbucket repositories for Git operations. See [Using SSH keys to secure Git operations](#).

Global permissions

Control user and group access to Bitbucket projects and to the Bitbucket instance configuration. For example, these can be used to control the number of user accounts that can access Bitbucket for licensing purposes.

See [Global permissions](#).

Project permissions

Apply the same access permissions to all repositories in a project. For example, these can be used to define the core development team for a project.

See [Using project permissions](#).

Repository permissions

Extend access to a particular repository for other, non-core, users. For example, these can be used to allow external developers or consultants access to a repository for special tasks or responsibilities.

See [Using repository permissions](#).

Branch permissions

Control commits to specific branches within a repository. For example, these can provide a way to enforce workflow roles such as the Release Manager, who needs to control merges to the release branch.

See [Using branch permissions](#).

Permissions matrix

The table below summarizes the cumulative effect of the permissions described above for anonymous and logged in users. In general, repository permissions override project permissions. A [personal project](#) can not be made public.

Key

Permission	Effect
BROWSE	Can view repository files, clone, pull to local
READ	Can browse, clone, pull, create pull requests, fork to a personal project
WRITE	Can merge pull requests
ADMIN	Can edit settings and permissions

Global (logged in)	Project	Repository	Branch	Effective permission
--------------------	---------	------------	--------	----------------------

✘	Personal	Personal	NA	No access
✘	Personal	Public access	NA	BROWSE just that repo
✘	No access	No access	NA	No access
✘	No access	Public access	NA	BROWSE just that repo
✘	Public access	Public access	NA	BROWSE all repos in project
✔	Personal	Personal	NA	No access
✔	Personal	Public access	NA	READ just that repo
✔	No access	No access	NA	No access
✔	No access	Public access	NA	READ just that repo
✔	Public access	No access	NA	READ all repos in project
✔	Public access	Public access	NA	READ
✔	Public access	Public access	For this user	READ that branch, no WRITE
✔	No access	READ	NA	READ just that repo
✔	Public access	READ	NA	READ just that repo
✔	READ	No access	NA	READ all repos in project
✔	READ	Public access	NA	READ all repos in project
✔	READ	READ	NA	READ all repos in project
✔	READ	No access	For this user	READ that branch, no WRITE
✔	No access	WRITE	NA	WRITE just that repo
✔	Public access	WRITE	NA	WRITE just that repo
✔	WRITE	No access	NA	WRITE all repos in project
✔	WRITE	WRITE	NA	WRITE all repos in project
✔	WRITE	WRITE	For other users	WRITE to other branches only
✔	ADMIN			Can edit settings and permissions

Allowing public access to code

You can open up public access for anonymous (unauthenticated) users to projects and repositories in Bitbucket Data Center and Server. This allows you to:

- Broadcast your repositories to a wider audience who generally don't have access to your source.
- Utilize unauthenticated cloning of repositories when setting up continuous integration servers to work with Bitbucket.
- Link from other systems, for example Jira applications or Confluence, to give users access to code without requiring authentication.
- Create opensource projects or repositories.

On this page:

- [Making a repository publicly accessible](#)
- [Making a project publicly accessible](#)
- [Viewing public repositories](#)
- [Disabling public access globally](#)

Related pages:

- [Using project permissions](#)
- [Using repository permissions](#)

Public access allows anonymous users to browse the files and commits for a specific repository or an entire project, and to clone repositories, without needing to login, or have an account in Bitbucket.

In Bitbucket, you can:

- configure a specific repository for public access.
- configure a project to allow public access to all repositories in the project.
- disable anonymous access by setting a global system property.

Making a repository publicly accessible

If you have **admin level permissions** for a specific repository, you can make it publicly available for **anonymous** access.

Go to the repository and click **Settings**, then **Repository** (under 'Permissions'). Check **Enable** (under 'Public Access') to allow users without a Bitbucket account to clone and browse the repository.

Making a project publicly accessible

If you have **admin level permissions** for a project, you can make the project publicly available for **anonymous** access.

This doesn't apply to private projects.

Go to the project and choose **Settings**, then **Permissions**. Check **Enable** (under 'Public Access') to allow users without a Bitbucket account to clone and browse any repository in the project.

Viewing public repositories

Bitbucket displays a list of repositories for which anonymous access has been enabled.

Anonymous and logged-in users can choose **Repositories > View all public repositories** to see these.

Disabling public access globally

Bitbucket provides a [system property](#) that allows you to turn off public access for the whole instance.

To do this, set the `feature.public.access` property to `false` in the `bitbucket.properties` file in your [Set the home directory](#).

Using project permissions

Project permissions in Bitbucket Data Center and Server allow you to manage access to repositories within a project in an aggregated way.

Related pages:

- [Creating projects](#)
- [Global permissions](#)
- [Using branch permissions](#)
- [Using repository permissions](#)
- [Allowing public access to code](#)

To modify permissions for a project

1. Go to **Settings > Permissions** for the project.
2. Click in the **Add Users** or **Add Groups** fields in the relevant section to search for, and bulk add, users or groups.
3. Choose a permission from the drop-down list, then click **Add**.

Once added, you can use the checkboxes to edit specific permissions for particular Bitbucket users or groups.

User access

Name	Admin ?	Write ?	Read ?
<input style="width: 80%;" type="text" value="Add Users"/> Read ▾ Add			
Admin Istrator	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Group access

Name	Admin ?	Write ?	Read ?
<input style="width: 80%;" type="text" value="Add Groups"/> Read ▾ Add			
confluence-users	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

There are three levels of project permission you can grant to a user or group for a project: *Admin*, *Write* and *Read*.

	Browse	Clone / Pull	Create, browse, comment on pull request	Merge pull request	Push	Create repositories	Edit settings / permissions
Project Admin	✓	✓	✓	✓	✓	✓	✓
Write	✓	✓	✓	✓	✓	✗	✗
Read	✓	✓	✓	✗	✗	✗	✗

Using repository permissions

Repository permissions in Bitbucket Data Center and Server allow you to manage access to a repository for an individual user or a user group beyond that already granted from [project permissions](#).

Related pages:

- [Using project permissions](#)
- [Using branch permissions](#)
- [Global permissions](#)
- [Allowing public access to code](#)

To modify permissions for a repository

1. Go to **Settings > Permissions** for the repository.
2. Click in the **Add Users** or **Add Groups** fields in the relevant section to search for, and bulk add, users or groups.
3. Choose a permission from the drop-down list, then click **Add**.

Once added, you can use the checkboxes to edit specific permissions for particular Bitbucket users or groups.

The screenshot shows the 'User access' and 'Group access' sections of the Bitbucket repository permissions interface. Each section has a search bar, a dropdown menu for permissions, and an 'Add' button. Below the search bar, there are rows for individual users or groups with checkboxes for 'Admin', 'Write', and 'Read' permissions.

User access			
Name	Admin ?	Write ?	Read ?
Add Users			Read ▾ Add
Admin Istrator	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Group access			
Name	Admin ?	Write ?	Read ?
Add Groups			Read ▾ Add
confluence-users	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

There are three levels of permissions you can grant to a Bitbucket user or group for a repository: *Admin*, *Write* and *Read*.

	Browse	Clone, fork, pull	Create, browse or comment on a pull request	Merge a pull request	Push	Edit settings and permissions
Admin	✓	✓	✓	✓	✓	✓
Write	✓	✓	✓	✓	✓	✗

Read						
------	---	---	---	---	---	---

Permissions regarding tasks

- Anyone with permission to browse a pull request can create a task on any comment, and can browse, resolve and reopen existing tasks in the pull request.
- Repository admins and pull request authors can edit and delete *any* task in the pull request. Reviewers and others can only edit or delete their *own* tasks.

Granting access to all repositories within a project

If you have a large number of repositories in a project, [project level permissions](#) provide a convenient way to grant access to *all* repositories within that project. For example you can grant a group, say "Team A", *Write* access at the project level, which will automatically give them *Write* access to all existing repositories in the project, as well as any repositories that are subsequently created in the project.

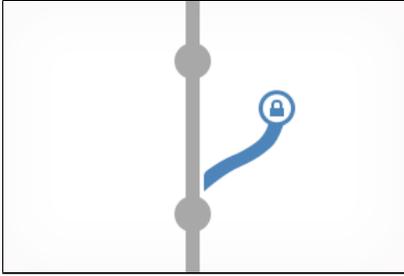
To modify permissions for a project, click the **Permissions** tab when viewing the project. You can add, or modify, permissions for individual users, and groups, in the same way as described above for a single repository.

Granting permission to create repositories

Only users with [project administration](#) permission can create *new repositories*.

Using branch permissions

Branch permissions in Bitbucket Data Center and Server control access to repository branches. This page describes branch permissions, and how to add branch permissions for an entire project or a single repository.



On this page:

- [About branch permissions](#)
- [Adding branch permissions](#)

Related pages:

- [Branches](#)
- [Automatic branch merging](#)
- [Branch permission patterns](#)

About branch permissions

Branch permissions provide another level of security in Bitbucket, with [user authentication](#) and [project, repository and global permissions](#), that together allow you to control or enforce your own workflow or process. With branch permissions you can control the actions users can perform on a single branch, branch type, or branch pattern within a repository or project.

If a user does not have commit access to the branch, an error message will be shown on the Git command line when they try to push a change to the branch. If no branch permissions are defined then anyone with commit access to the repository can push to any branch.

Good to know:

- Branch permissions are based on users or groups, and are actually restrictions, which are checked after project and repository level permissions, and prevent unauthorized pushing to or deleting a branch.
- They do not prevent branch creation. Branch permissions will only be enforced on updates to existing branches and tags.

Adding branch permissions

You can add branch permissions for all repositories in a project, or for individual repositories. Also, you can add multiple branch permissions for a project or repository. When creating a branch permission, you need to specify how to apply the permission, by either branch name, branch pattern, or branching model. You can also create exceptions for specific users, groups, or access key when creating or editing branch permissions.

Add branch permissions for all repositories in a project

To add branch permissions for all repositories in a project (requires project admin permission):

1. Go to **Project settings** > **Branch permissions**.
2. Click **Add permission**.
3. In the **Branches** field, specify which branches the permission applies to, either by **Branch name**, **Branch pattern**, or **Branching model**.
4. In the **Restrictions** field, select the type of actions to prevent and add exemptions for any of the selected restrictions.
Adding a user, group, or access key as an exemption means that restriction will *not* apply to them.

 Not adding any exemptions means the restriction will apply to everyone.

5. Click **Create** to finish.

Add a branch permission

Branches Branch name

Enter the branch name you want to restrict access to.

Branch pattern

Branching model

Restrictions Prevent all changes

Prevent deletion

Prevent rewriting history

Prevent changes without a pull request

Create Cancel

'Add a branch permission' dialog for a project

There are several ways to configure your branch permissions scheme when adding branch permissions for your project or repository. The section above outlines how to add branch permissions, however this section outlines the various options that are available from the **Add a branch permission** dialog.

'Branches' field options

For each branch permission, first determine which branch (or branches) to apply the permission to by either selecting a branch by name, branch pattern, or branching model.

Branch name

Enter the name of an existing branch to apply to restrict access to. Used to restrict access to a single branch.

Branch pattern

Use branch pattern syntax to select matching branches. Read more about [Branch permission patterns](#).

Branching model

Select a branch type to restrict access to multiple branches. Read more about [branching models](#).

'Restrictions' field options

Once you determine which branches a permission applies to, you then determine which actions to prevent, and optionally set exceptions to this permission.

Prevent all changes

Prevents pushes to the specified branch(es) and restricts creating new branches that match the branch (es) or pattern.

Prevent deletion

Prevents branch and tag deletion. Read [Branch permission patterns](#) for information about specifying tags.

Prevent rewriting history

Prevents history rewrites on the specified branch(es) - for example by a force push or rebase.

Prevent changes without a pull request

Prevents pushing changes directly to the specified branch(es); changes are allowed only with a pull request.

Add branch permissions for a single repository

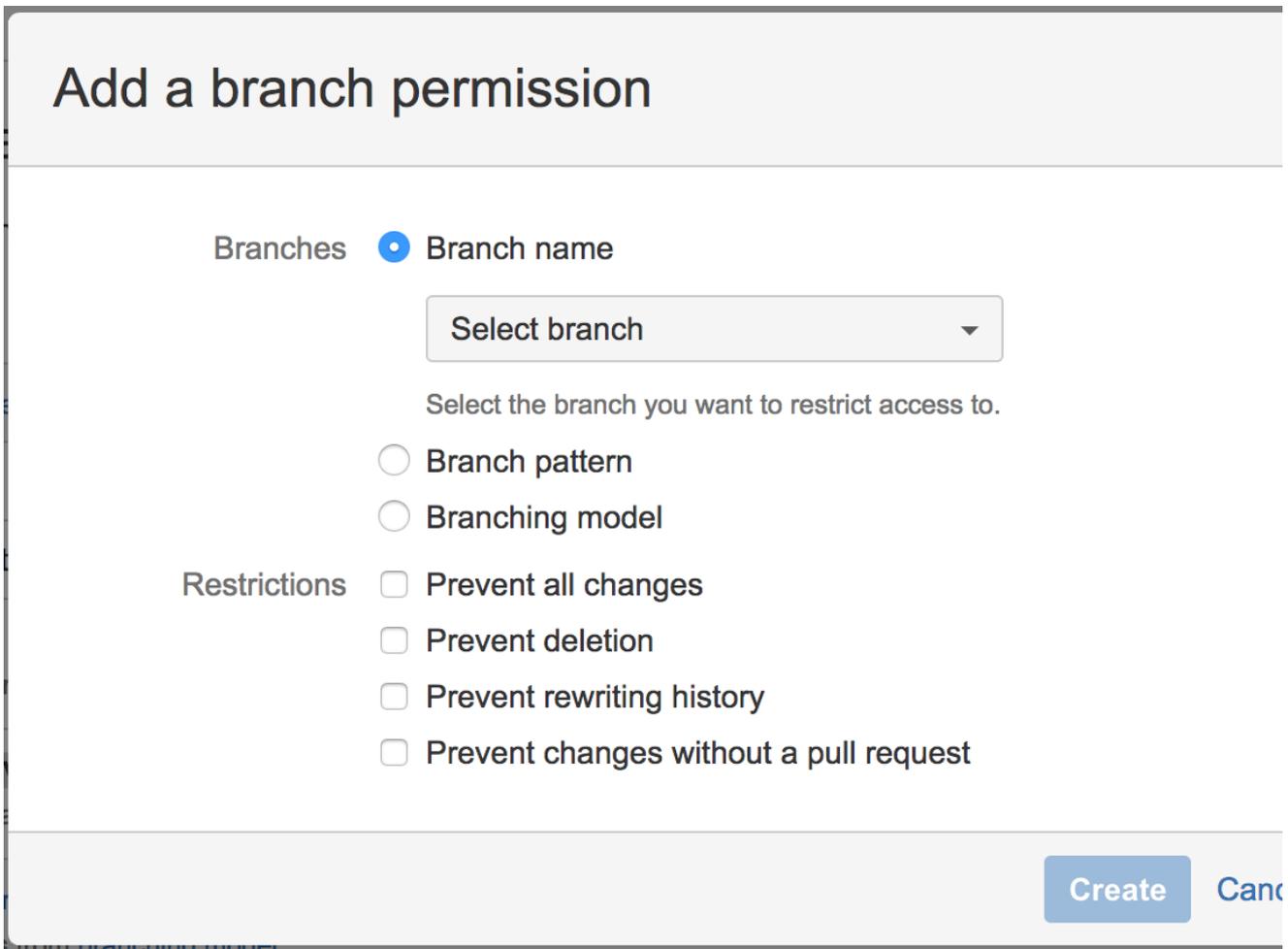
Adding branch permissions for an individual repository creates an additional branch permission. This means that the repository will have branch permissions from the project settings, and any additional branch permissions specifically set for a repository.

To add branch permissions for a single repository (requires repo admin permission):

1. Go to **Repository settings > Branch permissions**.
2. Click **Add permission**.
3. In the **Branches** field, specify which branches the permission applies to, either by **Branch name**, **Branch pattern**, or **Branching model**.
4. In the **Restrictions** field, select the type of actions to prevent and add exemptions for any of the selected restrictions.
Adding a user, group, or access key as an exemption means that restriction will *not* apply to them.

 Not adding any exemptions means the restriction will apply to everyone.

5. Click **Create** to finish.



'Add a branch permission' dialog for a repository

There are several ways to configure your branch permissions scheme when adding branch permissions for your project or repository. The section above outlines how to add branch permissions, however this section outlines the various options that are available from the **Add a branch permission** dialog.

'Branches' field options

For each branch permission, first determine which branch (or branches) to apply the permission to by either selecting a branch by name, branch pattern, or branching model.

Branch name

Enter the name of an existing branch to apply to restrict access to. Used to restrict access to a single branch.

Branch pattern

Use branch pattern syntax to select matching branches. Read more about [Branch permission patterns](#).

Branching model

Select a branch type to restrict access to multiple branches. Read more about [branching models](#).

'Restrictions' field options

Once you determine which branches a permission applies to, you then determine which actions to prevent, and optionally set exceptions to this permission.

Prevent all changes

Prevents pushes to the specified branch(es) and restricts creating new branches that match the branch(es) or pattern.

Prevent deletion

Prevents branch and tag deletion. Read [Branch permission patterns](#) for information about specifying tags.

Prevent rewriting history

Prevents history rewrites on the specified branch(es) - for example by a force push or rebase.

Prevent changes without a pull request

Prevents pushing changes directly to the specified branch(es); changes are allowed only with a pull request.

Branch permission patterns

Bitbucket Data Center and Serversupports a powerful type of pattern syntax for matching branch names (similar to pattern matching in Apache Ant). You can use branch permission patterns when [adding branch permissions](#) at the project or repository level to apply a branch permission to multiple branches.

Related pages:

- [Using branch permissions](#)
- [Branches](#)

These expressions can use the following wild cards:

?	Matches one character (any character except path separators)
*	Matches zero or more characters (not including path separators)
**	Matches zero or more <i>path segments</i> .

Pattern used in branch permissions match against all refs pushed to Bitbucket (i.e. branches and tags).

In Git, branch and tag names can be nested in a namespace by using directory syntax within your branch names, e.g. `stable/1.1`. The `'**'` wild card selector enables you to match arbitrary directories.

- A pattern can contain any number of wild cards.
- If the pattern ends with `/` then `**` is automatically appended - e.g. `foo/` will match any branches or tags containing a `foo` path segment
- Patterns only need to match a suffix of the fully qualified branch or tag name. Fully qualified branch names look like `refs/heads/master`, while fully qualified tags look like `refs/tags/1.1`.

Also see the [Ant documentation](#).

Examples

*	Matches everything
PROJECT-*	Matches and branch or tag named PROJECT-*, even in a name space. e.g. <code>refs/heads/PROJECT-1234</code> , <code>refs/heads/stable/PROJECT-new</code> or <code>refs/tags/PROJECT-1.1</code>
?.?	Matches any branch or tag of 2 characters separated by a <code>'.'</code> . e.g. <code>refs/heads/1.1</code> , <code>refs/heads/stable/2.X</code> or <code>refs/tags/3.1</code>
tags/ or tags/**	Matches all tags and any branches with 'tags' as a namespace. e.g. <code>refs/heads/stable/tags/some_branch</code> , <code>refs/tags/project-1.1.0</code>
heads/**/master	Matches all branches called master. e.g. <code>refs/heads/master</code> , <code>refs/heads/stable/master</code>

Using SSH keys to secure Git operations

Bitbucket Data Center and Server provides a simple way for user accounts and other systems to connect securely to Bitbucket repositories, using SSH keys, in order to perform Git operations. You can:

- add a personal key to a Bitbucket user account to allow a developer to easily authenticate when performing read operations from his or her local machine. A Bitbucket user can add any number of keys to their account. Read more at [SSH user keys for personal use](#).
- add an access key to a Bitbucket project or repository to allow other systems, such as build servers like Atlassian's [Bamboo](#), to authenticate for either read-only (pull, clone) or read-write (push, merge) operations, without the need to store user credentials. Read more at [SSH access keys for system use](#).

Related pages:

- [Creating SSH keys](#)
- [Enable SSH access to Git repositories](#)
- [Permanently authenticating with Git repositories](#)

Before you can use SSH keys to secure a connection with Bitbucket the following must have already been done:

- your Bitbucket administrator must have already [enabled SSH access](#) in Bitbucket.
- you need an SSH key! See [Creating SSH keys](#). Alternatively, you can use an existing key, if it isn't already being used as a repository or project access key in Bitbucket.

Note that:

- You can use the same SSH system access key for multiple repositories or projects.
- A Bitbucket user can add any number of keys to their account.
- Keys used for personal user accounts can't be re-used as a project or repository access key, and keys used as a project or repository access key can't be re-used for user accounts.
- Bitbucket supports DSA and RSA2 key types. RSA1 is not supported.

Creating SSH keys

This page describes how to create SSH keys.

SSH keys can be used to establish a secure connection with Bitbucket Data Center and Server for:

- when you are performing Git operations from your local machine
- when another system or process needs access to repositories in Bitbucket (for example your build server)

The SSH key needs to be added to Bitbucket, and your Bitbucket administrator must have [enabled SSH access](#) to Git repositories before you can make use of the key.

Bitbucket supports DSA, RSA2, and Ed25519 key types. RSA1 is not supported

You can use an existing SSH key with Bitbucket if you want, in which case you can go straight to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

On this page:

Related pages:

- [Using SSH keys to secure Git operations](#)
- [Enable SSH access to Git repositories](#)
- [Permanently authenticating with Git repositories](#)

Creating an SSH key on Windows

1. Check for existing SSH keys

You should check for existing SSH keys on your local computer. *You can use an existing SSH key with Bitbucket if you want, in which case you can go straight to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).*

Open a command prompt, and run:

```
cd %userprofile%\.ssh
```

- If you see "No such file or directory", then there aren't any existing keys: [go to step 3](#).
- Check to see if you have a key already:

```
dir id_*
```

If there are existing keys, you may want to use those: [go to either SSH user keys for personal use or SSH access keys for system use](#).

2. Back up old SSH keys

If you have existing SSH keys, but you don't want to use them when connecting to Bitbucket, you should back those up.

In a command prompt on your local computer, run:

```
mkdir key_backup  
copy id_rsa* key_backup
```

3. Generate a new SSH key

If you don't have an existing SSH key that you wish to use, generate one as follows:

1. Log in to your local computer as an administrator.
2. In a command prompt, run:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

Associating the key with your email address helps you to identify the key later on.

Note that the `ssh-keygen` command is only available if you have already installed [Git](#) (with Git Bash). You'll see a response similar to this:

```
C:\Users\ASUS>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/ASUS/.ssh/id_rsa):
```

3. Just press <Enter> to accept the default location and file name. If the `.ssh` directory doesn't exist, the system creates one for you.
4. Enter, and re-enter, a passphrase when prompted. The whole interaction will look similar to this:

```
C:\Users\ASUS>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/ASUS/.ssh/id_rsa):
Created directory 'C:/Users/ASUS/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:/Users/ASUS/.ssh/id_rsa.
Your public key has been saved in C:/Users/ASUS/.ssh/id_rsa.pub.
The key fingerprint is:
e6:99:c3:3c:52:fb:9c:e4:3f:df:4d:b2:80:11:a5:1e ASUS@ASUS-PC
C:\Users\ASUS>
```

5. You're done and you can now go to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

Creating an SSH key on Linux & macOS

1. Check for existing SSH keys

You should check for existing SSH keys on your local computer. *You can use an existing SSH key with Bitbucket if you want, in which case you can go straight to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).*

Open a terminal and run the following:

```
cd ~/.ssh
```

- If you see "No such file or directory, then there aren't any existing keys: [go to step 3](#).
- Check to see if you have a key already:

```
ls id_*
```

- If there are existing keys, you may want to use them; go to either [SSH user keys for personal use](#) or [SSH access keys for system use](#).

2. Back up old SSH keys

If you have existing SSH keys, but you don't want to use them when connecting to Bitbucket, you should back those up.

Do this in a terminal on your local computer, by running:

```
mkdir key_backup
cp id_rsa* key_backup
```

3. Generate a new key

If you don't have an existing SSH key that you wish to use, generate one as follows:

1. Open a terminal on your local computer and enter the following:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

Associating the key with your email address helps you to identify the key later on.

You'll see a response similar to this:

```
pwatsons-Mac-Pro: ~ pwatson$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/pwatson/.ssh/id_rsa):
```

2. Just press <Enter> to accept the default location and file name. If the `.ssh` directory doesn't exist, the system creates one for you.
3. Enter, and re-enter, a passphrase when prompted.

The whole interaction will look similar to this:

```
pwatsons-Mac-Pro: ~ pwatson$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/pwatson/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/pwatson/.ssh/id_rsa.
Your public key has been saved in /Users/pwatson/.ssh/id_rsa.pub.
The key fingerprint is:
47: 68: 04: f3: 93: 53: a3: af: bd: fc: 86: 60: 30: 47: cd: ea pwatson@pwatsons-Mac-Pro.local
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      o..oo          |
|      +.=o.         |
|      .O.           |
|      o.o=          |
|      =S o          |
|      E+            |
|      .....         |
|      ....          |
|      oo.           |
+-----+
pwatsons-Mac-Pro: ~ pwatson$
```

4. You're done and you can now go to either [SSH user keys for personal user](#) or [SSH access keys for system use](#).

SSH user keys for personal use

You can use SSH keys to establish a secure connection between your computer and Bitbucket Data Center and Server for when you are performing read-only (pull, clone) Git operations from your local machine. Personal keys are attached to your Bitbucket account they are bound by that account's permissions and use the account's identity for any operations.

Before you can use SSH keys to secure a connection with Bitbucket the following must have already been done:

- your Bitbucket administrator must have already [enabled SSH access](#) in Bitbucket.
- you need an SSH key! See [Creating SSH keys](#). Alternatively, you can use an existing key, if it isn't already being used as a repository or project access key.
- you need to have added your personal SSH key to your Bitbucket account see the following section.

Once you have an SSH key associated with your Bitbucket account, using it is easy! See [Use SSH keys to connect to Bitbucket repositories](#) below.

Note that:

- Bitbucket supports DSA, RSA2, and Ed25519 key types. RSA1 is not supported.
- A Bitbucket user can add any number of keys to their account.
- You can use the same SSH access key for multiple repositories or projects.
- Keys used for personal user accounts can't be re-used as a [project or repository access key](#), and keys used as a project or repository access key can't be re-used for user accounts.

Add an SSH key to your Bitbucket account

1. On Windows, in your command prompt, change directory to your .ssh directory, and copy the public key file to your clipboard by running:

Windows

```
cd %userprofile%\.ssh
clip < id_rsa.pub
```

On macOS or Linux simply run the following in a terminal:

Mac OS X

```
pbcopy < ~/.ssh/id_rsa.pub
```

Note that on Linux, you may need to download and install xclip, then use that, as shown in this code snippet:

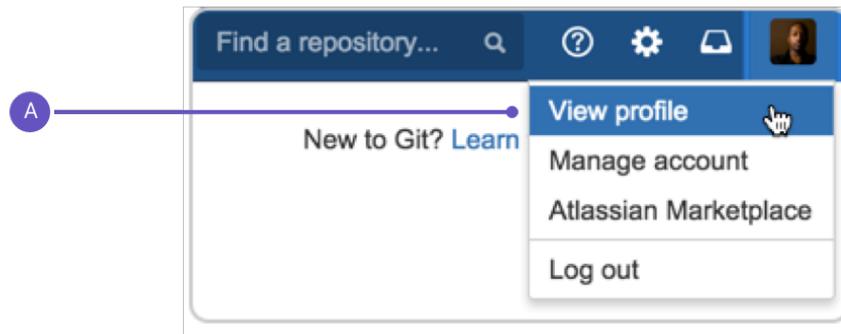
Linux

```
sudo apt-get install xclip
xclip -sel clip < ~/.ssh/id_rsa.pub
```

Related pages:

- [Creating SSH keys](#)
- [Enable SSH access to Git repositories](#)
- [Permanently authenticating with Git repositories](#)

2. In Bitbucket, go to your account:



- a. Select **View profile**.
3. Click on **SSH keys** and then **Add key**.
4. Paste the key into the text box:



5. Click **Add key**. You're done!

Use SSH keys to connect to Bitbucket repositories

SSH access needs to have been set up, as [described above](#). Once this is done, you can use SSH keys as follows:

1. Go to **Projects**, click a project, and choose a repository from the list.
2. Click **Clone** in the sidebar to see the clone URLs for the repository.
3. Choose the clone URL you want to use. SSH is available if you have already added an SSH key to your account. If you haven't done that yet, see [Add an SSH key to your Bitbucket account](#), above.

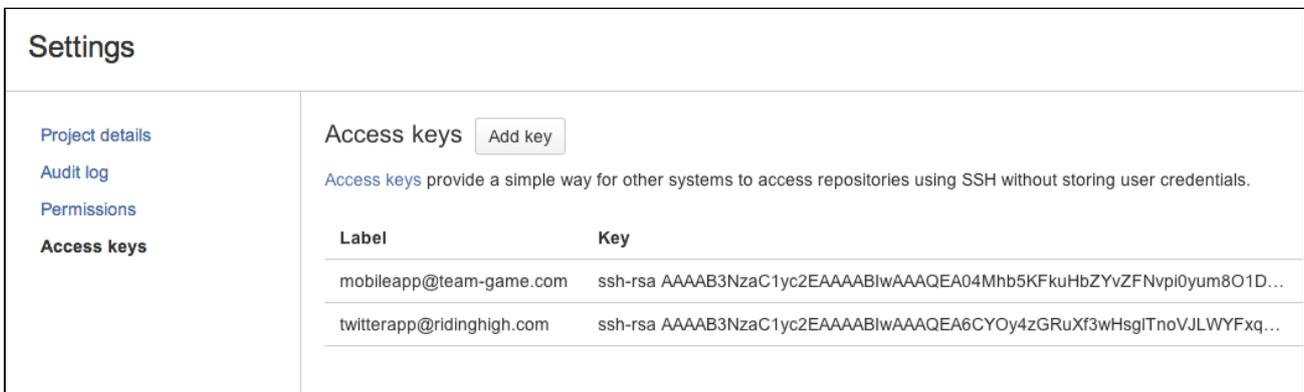
SSH access keys for system use

Bitbucket Data Center and Server administrators can set up SSH access keys to secure the Git operations that other systems perform on the repositories managed in Bitbucket. Using access keys avoids the need to store user credentials on another system, and means that the other system doesn't have to use a specific user account in Bitbucket. For example, access keys can be used to allow your build and deploy server to authenticate with Bitbucket to check out and test source code.

- [Project admins](#) can add and manage SSH access keys for a project. The keys apply to every repository in the project.
- [Repository admins](#) can add and manage SSH access keys for a particular repository.
- The access key can allow either *read-only* or *read-write* Git operations.
- Because they are designed to be used for system access, SSH access keys may push commits that are not signed with a GPG key even if the "Verify Commit Signature" hook is enabled.

Related pages:

- [Creating SSH keys](#)
- [Enable SSH access to Git repositories](#)
- [Permanently authenticating with Git repositories](#)



The screenshot shows the Bitbucket Settings page for a project. On the left is a sidebar with navigation links: Project details, Audit log, Permissions, and Access keys. The main content area is titled 'Access keys' and includes an 'Add key' button. Below the button is a descriptive sentence: 'Access keys provide a simple way for other systems to access repositories using SSH without storing user credentials.' A table lists existing keys with columns for 'Label' and 'Key'. Two keys are shown: one for 'mobileapp@team-game.com' and another for 'twitterapp@ridinghigh.com', both using 'ssh-rsa' type keys.

Label	Key
mobileapp@team-game.com	ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA04Mhb5KFkuHbZYvZFNvpi0yum8O1D...
twitterapp@ridinghigh.com	ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6CYOy4zGRuXf3wHsgITnoVJLWYFqx...

Note that Bitbucket supports DSA, RSA2, and Ed25519 key types. RSA1 is not supported.

Before you can use SSH keys to secure a connection with Bitbucket the following must have already been done:

- Your Bitbucket administrator must have already [enabled SSH access](#), on Bitbucket.
- You must have already created an SSL key. See [Creating SSH keys](#). Alternatively, you can use an existing key, if it isn't already being used for a personal account in Bitbucket.

Using SSH keys to allow access to Bitbucket repositories

To get the SSH key to work with your build, or other, system, you need to:

- Add the private key to that system. For Bamboo, see this page: [Shared credentials](#).
- Add the public key to Bitbucket as described here:

Add an SSH access key to either a Bitbucket project or repository

You simply copy the public key, from the system for which you want to allow access, and paste it into Bitbucket.

1. Copy the public key. One approach is to display the key on-screen using `cat`, and copy it from there:

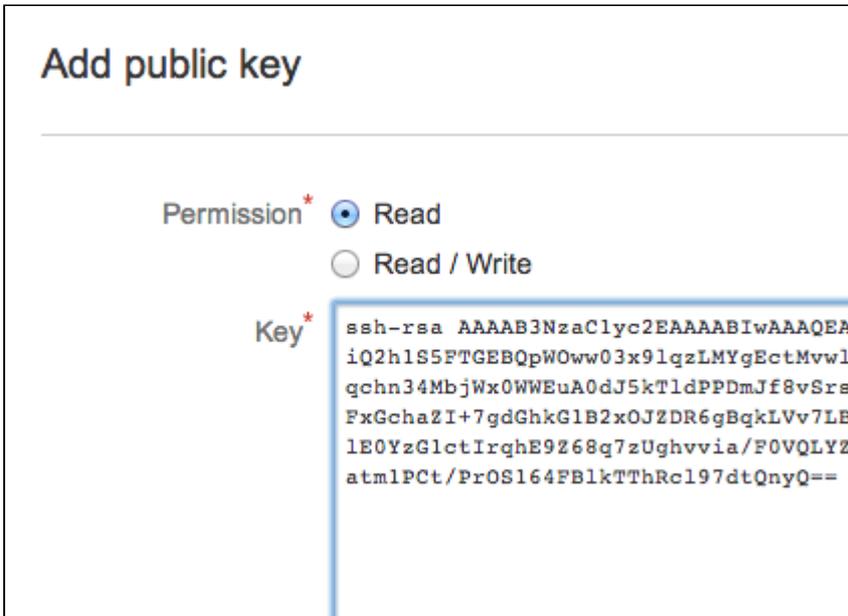
```
cat < ~/.ssh/id_rsa.pub
```

2. Now, in Bitbucket, go to the **Settings** tab for the project or repository.
3. Click **Access keys** and then **Add key**.
4. Choose the **Read** permission, for `git pull` or `git clone` operations for example, where you want to be sure that the system will *not* be able to write back to the Bitbucket repository.

Choose the **Read / Write** permission, for `git push` or `git merge` operations for example, where you may want your build system to mergesuccessful feature branch builds to the default branch in theBitbucket repository, or so that deployments can be tagged.

Note that if you attempt toadd a key already present on a project or repository but with a different permission to what it currently has, the permission will simply be updated.

5. Paste the key into the text box and click**Add key**.



Bitbucketlicense implications

- System access keys do not require an additional Bitbucketuser license.

Reusing access keys

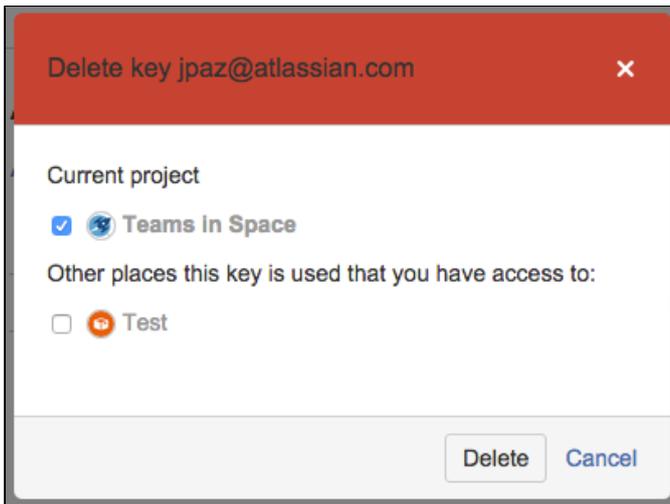
- You can use the same SSHaccess key for multiple repositories or projects.
- Keys used for [personal user accounts](#) can't be re-used as a project or repository system access key, and keys used as a project or repository access keycan't be re-usedfor user accounts.

Deleting an access key

You can delete an access key by going to**Settings >Access keys** for the repository, and clicking the cross for the key (the cross only appears when you move the mouse pointer there):



If the key is used for multiple projects or repositories, you can select the other places that you want the key to be deleted from:



Note that the dialog only displays the projects and repositories that you have permission to see. Be aware that the key may also be used in other places that are not listed in the dialog. To be 100% sure that all uses of the key are deleted, this operation must be performed by someone with the administrator or sysadmin [global permission](#).

Note that Bitbucket supports DSA, RSA2, and Ed25519 key types. RSA1 is not supported.

Using GPG keys

GPG keys are a way to sign and verify work from trusted collaborators. This page describes how to generate a GPG key to sign and verify commits and tags for use with Bitbucket Data Center and Server.



On this page:

- [About GPG keys](#)
- [Requiring GPG keys](#)
- [Install GPG](#)
- [Check for existing GPG keys](#)
- [Generate a new GPG key](#)
- [Add a GPG key](#)
- [Configure Git to use your GPG key](#)
- [Sign commits and tags with a GPG key](#)

About GPG keys

GPG is a command line tool used together with Git to encrypt and sign commits or tags to verify contributions in Bitbucket. In order to use GPG keys with Bitbucket, you'll need generate a GPG key locally, add it to your Bitbucket account, and also set it up for use with Git. If you already have a GPG key ready to go, you can jump straight to the [Add a GPG key to Bitbucket](#) section.

Administrators can also add GPG keys on behalf of their Bitbucket users, which can be useful if your organization manages public-key certificates with a keyserver.

Requiring GPG keys

Project and repository administrators can [enable the "Verify Commit Signature" hook](#) to require that commits are signed with GPG keys. When this hook is enabled, only [SSH access keys](#) are allowed to push unsigned commits.

Install GPG

If you don't already have GPG, you'll need to install it locally. You can install GPG manually using binaries for your operating system on the [GnuPG Download page](#), or use a package manager like [Homebrew](#).

Check for existing GPG keys

If you're not sure if you have a GPG key already, you can check for existing GPG keys locally.

To check if you have existing GPG keys:

1. In a terminal, use this command to list GPG keys you have access to:

```
gpg --list-secret-keys --keyid-format LONG
```

2. Check the output to see if you have a GPG key pair.
3. **If there are no GPG key pairs**, you'll need to [generate a new GPG key](#).

If there are GPG key pairs you want to use, you'll need to [add them to your Bitbucket account](#).

Generate a new GPG key

In order to generate a new GPG to sign commits and tags you need to have [GPG installed](#) already.

To generate a new GPG key:

1. In a terminal, use this command to generate a GPG key:

```
gpg --gen-key
```

2. Provide the information asked at the prompts.
 - a. Enter your identifying information.
 - b. Enter a secure passphrase.
3. Use this command to list your GPG keys.

```
gpg --list-secret-keys --keyid-format LONG
```

4. Copy the GPG key ID to use with Bitbucket. For example, below the GPG key ID is **7FFFC09ACAC05FD0**.

```
gpg --list-secret-keys --keyid-format LONG
/Users/bitbucketbot/.gnupg/pubring.gpg
-----
sec rsa2048/7FFFC09ACAC05FD0 2017-06-02 [SC] [expires: 2019-06-02]
5538B0F643277336BA7F0E457FFFC09ACAC05FD0
uid [ultimate] BitbucketBot <bitbucket@realaddress.com>
ssb rsa2048/95E8A289DFE77A84 2017-06-02 [E] [expires: 2019-06-02]
```

5. Get your public key you'll add to Bitbucket.
 - a. Paste the GPG key ID into this command to export the public key you will enter in Bitbucket.

```
gpg --armor --export 7FFFC09ACAC05FD0
```

- b. From the output, copy your public GPG key, which starts at -----BEGIN PGP PUBLIC KEY BLOCK----- and ends at -----END PGP PUBLIC KEY BLOCK-----.

You can now add your public GPG key to your Bitbucket account.

Add a GPG key

In order to use your GPG key with Bitbucket, you need to have [GPG installed](#), and have [generated a GPG key](#) to add.

To add your GPG key to:

1. From within Bitbucket, go to your account by clicking your profile picture in the upper-right, and select **Manage account**.
2. Click **GPG keys**. > **Add key**.
3. Copy your GPG key.

From a terminal, use this command to copy your GPG key to your clipboard:

```
gpg --armor --exportMY_KEY_ID | pbcopy
```

4. Paste your GPG key in the **Key** field, then click **Add key**.

Configure Git to use your GPG key

In order to use GPG keys with Bitbucket, you need to configure your local version of Git which GPG key to use.

To configure Git to use your GPG key:

1. Copy your GPG key ID.

To list your GPG keys, use this command:

```
gpg --list-secret-keys --keyid-format LONG
```

Copy the GPG key ID to use with Bitbucket. For example, below the GPG key ID is **7FFFC09ACAC05FD0**.

```
gpg --list-secret-keys --keyid-format LONG
/Users/bitbucketbot/.gnupg/pubring.gpg
-----
sec rsa2048/7FFFC09ACAC05FD0 2017-06-02 [SC] [expires: 2019-06-02]
5538B0F643277336BA7F0E457FFFC09ACAC05FD0
uid [ultimate] BitbucketBot <bitbucket@realaddress.com>
ssb rsa2048/95E8A289DFE77A84 2017-06-02 [E] [expires: 2019-06-02]
```

2. Paste your GPG key ID into this command to set your GPG key in Git.

```
git config --global user.signingkeyMY_KEY_ID
```

Sign commits and tags with a GPG key

In order to sign commits and tags with a GPG key in Bitbucket, you need to have:

- [installed GPG locally](#),
- [added a GPG key to your Bitbucket account](#), and
- [configured your local version of Git which GPG key to use](#).

To sign commits with your GPG key:

1. When committing changes to a local branch, use the `-s` flag to the `git commit` command:

```
git commit -s -m your commit message
```

2. Enter the passphrase for your GPG key.

To sign tags with your GPG key, add the `-s` flag to your `git tag` command:

```
git tag -S yourtag
```

You can verify a tag was signed using this command:

```
git tag -v yourtag
```

Workflow strategies

Various Git workflows are supported by Bitbucket Data Center and Server:

- [Centralized Workflow](#)
- [Feature Branch Workflow](#)
- [Gitflow Workflow](#)
- [Forking Workflow](#)

For information about setting up Git workflows in Bitbucket see [Branches](#) and [Forks](#).

Centralized Workflow

Like Subversion, the



Centralized Workflow uses a central repository to serve as the single point-of-entry for all changes to the project. Instead of `trunk`, the [default development branch can be customized](#) and all changes are committed into this branch. This workflow doesn't require any other branches besides the default branch.

[Read more about the Centralized Workflow...](#)

Feature Branch Workflow

The core idea behind the Feature Branch Workflow is that all feature development should take place in a dedicated branch instead of the default branch. This encapsulation makes it easy for multiple developers to work on a particular feature without disturbing the main codebase. It also means the default branch will never contain broken code, which is a huge advantage for continuous integration environments.

[Read more about the Feature Branch Workflow...](#)

Gitflow Workflow

The Gitflow Workflow defines a strict branching model designed around the project release. While somewhat more complicated than the Feature Branch Workflow, this provides a robust framework for managing larger projects.

[Read more about the Gitflow Workflow...](#)

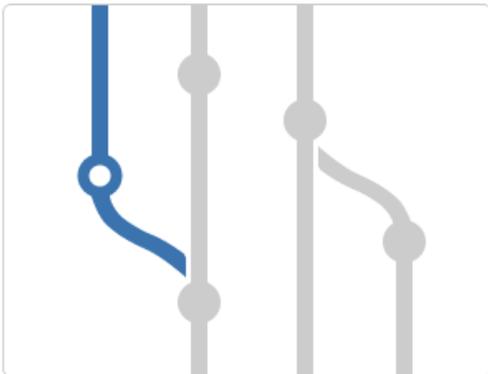
Forking Workflow

The Forking Workflow is fundamentally different than the other workflows discussed in this tutorial. Instead of using a single server-side repository to act as the central codebase, it gives every developer a server-side repository. This means that each contributor has not one, but two Git repositories: a private local one and a public server-side one.

[Read more about the Forking Workflow...](#)

Branches

Bitbucket Data Center and Server makes it easy to use a [branching workflow](#) for your Git development process. This page describes how to use to use branches with Bitbucket.



On this page:

- [Creating a branch](#)
- [Configuring branching models](#)
- [Automating the branch workflow](#)
- [Finding and managing branches](#)

Related pages:

- [Using branch permissions](#)
- [Automatic branch merging](#)
- [Branch permission patterns](#)

Creating a branch

You can create a new branch [from within JIRA Software](#) or [in Bitbucket](#). Bitbucket suggests the Branch type and Branch name based on where you are creating the branch from in the application. You can change these values depending on your branching model.

When creating a branch, you will provide this information for each branch:

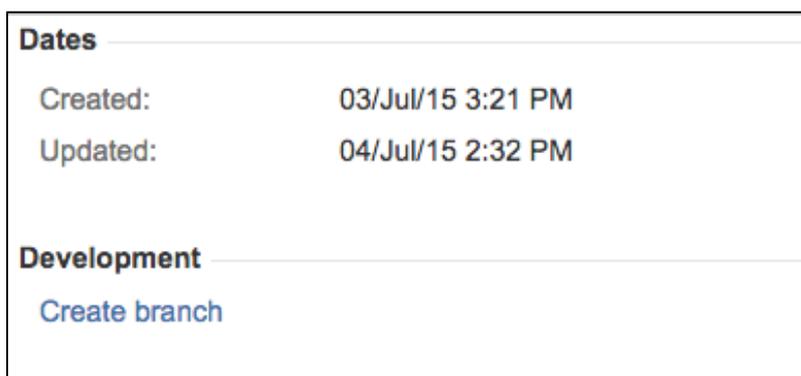
- the **Repository**
- the **Branch type**, if a [branching model](#) has been previously configured choose **Custom** if you need an *ad hoc* branch type
- the **Branch from point** you can choose either a branch or a tag.
- the **Branch name** the prefix is based on the branch type you selected, and as defined by the [branching model](#).

Create a branch from JIRA Software

To create a branch when viewing an issue in JIRA Software:

i *Must have JIRA Software version 6.1 or above, and it must be connected with Bitbucket by an [application link](#).*

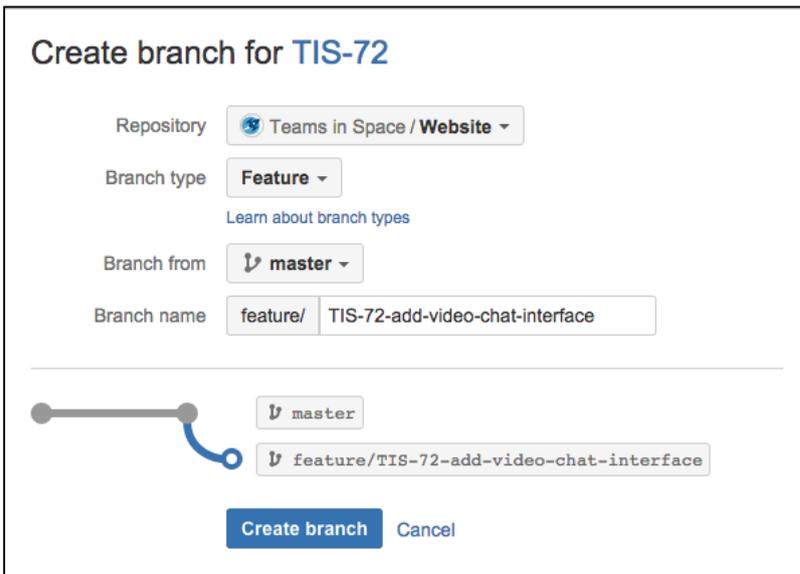
1. In the Development pane, click **Create Branch** (requires the View Development Tools project permission).



2. Choose the SCM, if more than one is available, where you want to create the branch.

3. Select the **Branch type** and **Branch name**, then click **Create branch**. Bitbucket suggests a **Branch type** based on the JIRA Software issue type, when a **branching model** is configured.

JIRA Software issue type	Bitbucket branch type
Bug	Bugfix
Story	Feature
New Feature	Feature

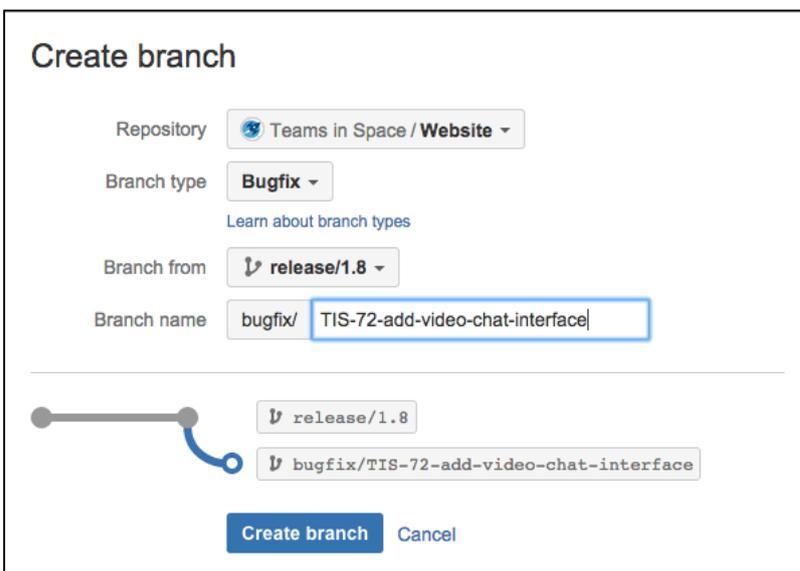


4. Once the new branch is created, Bitbucket takes you to the file listing for that. You can now pull to your local repository and switch to the new branch.

Create a branch from Bitbucket

To create a branch:

1. In Bitbucket, choose **Create branch** from the sidebar.
2. Select the **Branch type** and **Branch name**.



3. Click **Create branch**. Once the new branch is created, Bitbucket takes you to the file listing for that. You can now pull to your local repository and switch to the new branch.

Configuring branching models

With Bitbucket you can use branching models to define a branch workflow for repositories. When you map your workflow to repository branches with a branching model, admins can guide developers to name branches consistently by configuring which branch types to make available. There are a number of branch types available, and several branch types have default branch naming prefixes (described below). You can also specify your own naming convention for each branch type. A consistent naming convention makes it easier to identify branches by type.

Branch types

Bitbucket comes with several types of branches that are frequently used in software development. This section explains what each branch type is for, and has the typical naming convention for the prefix for each branch type. The prefix can be changed for each branch type.

	<p>Development branch</p> <p>Usually the integration branch for feature work and is often the default branch or a named branch. For pull request workflows, the branch where new feature branches are targeted.</p>	<p>va ries</p>
	<p>Production branch</p> <p>Used for deploying a release. Branches from, and merges back into, the development branch. In a Gitflow-based workflow it is used to prepare for a new production release.</p>	<p>va ries</p>
	<p>Feature branch</p> <p>Used for specific feature work or improvements. Generally branch from, and merge back into, the development branch using pull requests. See Feature branch workflow.</p>	<p>f e a t u r e/ /</p>
	<p>Release branch</p> <p>Used for release task and long-term maintenance versions. They branch from, and merge back into, the development branch. Merging into an older release branch can be configured to automatically merge to newer release branches, as well as the development branch.</p>	<p>r e l e a s e/ /</p>
	<p>Bugfix branch</p> <p>Typically used to fix Release branches.</p>	<p>b u g f i x/ /</p>

	<p>Hotfix branch</p>	<p>h o t f i x /</p>
<p>Used to quickly fix a Production branch without interrupting changes in the development branch. In a Gitflow-based workflow, changes are usually merged into the production and development branches.</p>		

You can configure a branching model for either:

- **a whole project**, so that repositories within a project can inherit the same branching model, or
- **an individual repository**, to override a project's branching model.

Good to know:

- New repositories will have the branching model enabled by default, and use the default branch prefixes.
- Enabled branch types can't have empty prefixes, have a 30 character limit, and can't overlap (for example PROD and PRODUCT would overlap).

Configure a project's branching model

To configure a branching model for a project (requires project admin permission):

1. Go to **Project settings > Branching model**.
2. Choose the details of branching model for repositories that inherit the project settings, then click **Save**.

Configure a repository's branching model

To configure the branching model for a repository (requires repository admin permission),

1. Go to **Repository settings > Branching model** for a repository
2. Under **Project settings inheritance**, select **Use custom settings**.
3. Choose the details of your repository branching model, then click **Save**.

Automating the branch workflow

Bitbucket can automate some merges in the branch workflow, based on the branching model for the repository. This allows merges to be cascaded to newer branches of the same parent, subject to a few conditions, so reducing the need for manual maintenance of branches.



To enable automatic merge for a project or repository (requires repo/project admin permissions):

1. Go to **Settings > Branching model** for a project or repository.
2. Under Automatic merge, check **Enable automatic merging**.
3. Click **Save**.

If Bitbucket cannot perform an automatic merge, perhaps because of a conflict, it creates a new pull request for that merge, and the automatic merge operation stops. You should then resolve the conflict locally before approving the new pull request, which may involve further cascading merges.

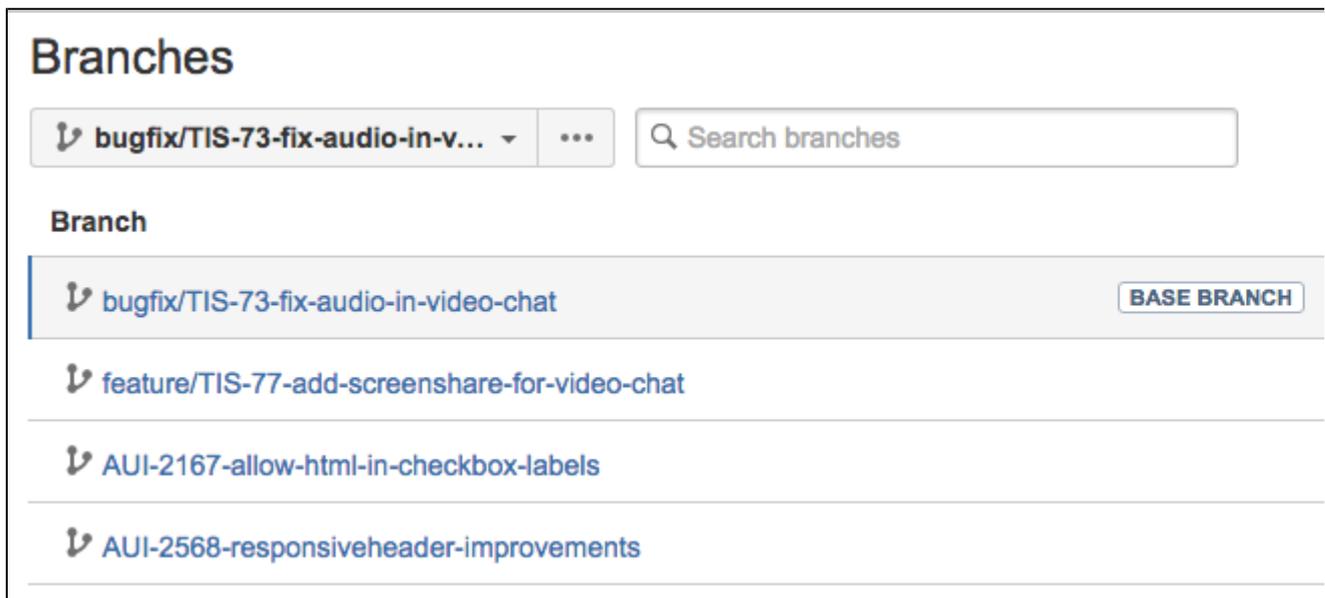
See [Automatic branch merging](#) for more information about the conditions for automatic merging, and how Bitbucket determines the ordering of branches.

Finding and managing branches

The branch listing pagemakes it easy to keep track of all the branches in your repository, to get there just select Branches from the side navigation bar.

The branch listing allows you to:

- See how many commits behind orahead your branch is compared to a chosen 'base branch'.
- Seethe latest status for pull requests originating from branches.
- See thebuild status of branches at a glance.
- Track the review and merge work that still needs to be done and can help with branch cleanup.
- Identify work in progress as well as stale branches. It is calculated for each branch against the base branch.



Behind /Ahead	Shows by how many commits a branch has diverged from the (default) 'base branch'. Use the branch selector to change the base branch.
Pull requests	Shows the relevant state of pull requests against each branch click the status to see detailed pull request information. <ul style="list-style-type: none"> • OPENif there is at least one open pull request. • MERGEDif there are no open pull requests, and at least onepull request has been merged. • DECLINEDif there are no open or mergedpull requests, and at least one pull request has been declined.
*Builds	Shows the status of the latest build results published to Bitbucket.The overall status is 'passed' if all the different builds (for example, unit tests, functional tests, deploy to staging) succeeded and 'failed' if at least one run failed for any of those.Click an icon to see details of the builds.

Actions	Menu that includes tasks for working with branches.
----------------	---

* Only if you have an [integrated build server](#).

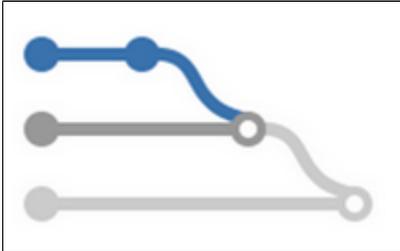
Search for branches

You can easily find branches by using the search at the top of the table on the Branches screen. If you're using a [branching model](#), you can filter by branch type simply by searching for the prefix for example, search for "feature/" to see all your feature branches.

You can find the feature and bugfix branches that haven't yet been merged into a particular release (for example, "release/2.10") by changing the 'base branch' just use the branch selector to change the base branch, and refer to the **Behind/Ahead** and **Pull requests** columns.

Automatic branch merging

Bitbucket Data Center and Server can automatically merge changes to newer release branches, reducing the need for manual maintenance of branches. This page describes best practices for establishing a branching strategy conducive for automatic branch merging, and how to enable it for individual repositories or all repositories in a project.



On this page:

- [Conditions for automatic merging](#)
- [Enable automatic branch merging](#)
- [Branch ordering algorithm](#)

Related pages:

- [Branches](#)
- [Using branch permissions](#)
- [Branch permission patterns](#)

Conditions for automatic merging

In order to use automatic branch merging, Bitbucket has to be able to determine the [ordering of branches](#), and relies on [semantic versioning](#) of branch names. For example, Bitbucket will order these branch names like this: 1.0.0 < 2.0.0 < 2.1.0 < 2.1.1. Read more about the [Branch ordering algorithm](#), including some examples of branch ordering, later on this page. Bitbucket expects that the 'development' branch, typically the default branch, is always ahead of any 'release' branches. The final merge in the automatic cascade will be to the 'development' branch.

These conditions must also be satisfied:

- [Branching model](#) must be configured.
- The 'release' branch type must be enabled or a 'production' branch must be set.
- The merge must be made using a pull request to a 'release' or 'production' branch type.
- There must be newer branches than the target branch of the pull request.

Good to know:

- Automatic merging is off by default.
- Commit messages will indicate a merge was automatic.
- There are audit log entries for automatic merges.
- [Notifications](#) are sent when merges succeed or fail.

What happens if an automatic merge fails?

When an automatic merge fails, Bitbucket creates a new pull request for the failed merge, and the automatic merge operation stops. You should resolve the conflict locally before approving the new merge, which may start a new series of cascading merges. Note that a pull request that gets automatically opened when a merge fails won't trigger the continuation of the initial merge chain if resolved locally (which is the approach that we recommend).

There are a few reasons an automatic merge could fail:

- There could be a conflict detected that prevents the merge.
- There is already an open pull request with the same source and target that the automatic merge would close.

Enable automatic branch merging

You can enable automatic branch merging for all repositories in a project, and choose to inherit or override this setting in individual repositories.

Enable automatic branch merging for all repositories in a project

All repositories in a project can inherit or override the branching model, including automatic branch merging.

To enable automatic branch merging for all repositories in a project (requires project admin permission):

1. Go to **Project settings > Branching model**.
2. Select **Enable automatic merging**, then click **Save**.

Enable automatic branch merging for a single repository

To enable automatic branch merging for a single repository:

1. Go to **Repository settings > Branching model**.
2. Select **Enable automatic merging**, then click **Save**.

Branch ordering algorithm

Bitbucket is able to automatically merge changes to newer release branches, as long as Bitbucket can determine the ordering of those branches. Ordering is based on [semantic versioning](#) in the naming pattern for branches.

Bitbucket uses the following ordering algorithm to determine the branches in the merge chain:

- Branches are selected and ordered on the basis of the name of the branch that started the cascade (i.e. the target of the pull request for the merge).
- Branch names are split into tokens using any of these characters: underscore '_', hyphen '-', plus '+', or period '.'.
- Only branches *matching* the name of the pull request target are added into the merge path. Matching means that every token before the first numeric token must be equal to the corresponding tokens of the target branch's name.
- Branches are ordered by number, if a given token is numeric. When comparing a numeric token with an ASCII token, the numeric is ranked higher (i.e. is considered as being a newer version).
- If both tokens are non-numeric, a simple ASCII comparison is used.
- In the unlikely case of the above algorithm resulting in equality of 2 branch names, a simple string comparison is performed on the whole branch name.
- There is a limit of 30 merges.

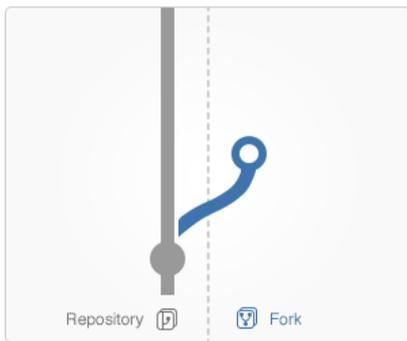
Branch ordering examples

The table below provides examples of branch naming patterns that Bitbucket is able, and not able, to order correctly:

<div style="background-color: #008000; color: white; padding: 2px 5px; border-radius: 3px; display: inline-block;">GOOD</div>	<ul style="list-style-type: none"> • release/1.0 • release/1.1-rc1 • release/1.1 • release/1.2 • release/2.0 	<p>Bitbucket tokenizes on the '.' and the '-' of '1.1-rc1' and is able to order these branch names correctly.</p>
---	---	---

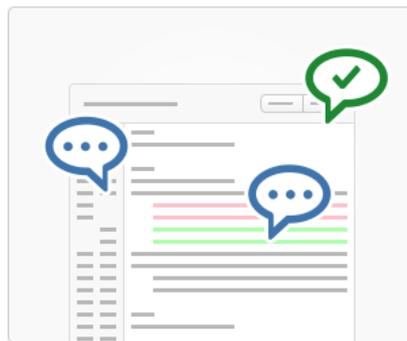
<p>GOOD</p>	<ul style="list-style-type: none">• release /bitbucket_1. 1• release /bitbucket_1. 2• release /bitbucket_2. 0	<p>Bitbucket tokenizes on the '.' and the '_' and orders the numeric parts of these branch names correctly.</p>
<p>BAD</p>	<ul style="list-style-type: none">• release/1.0• release /bitbucket_1. 1	<p>Bitbucket tokenizes on the '.' and the '_' but cannot recognize that 'bitbucket_1.1' should follow '1.0'.</p>

Forks



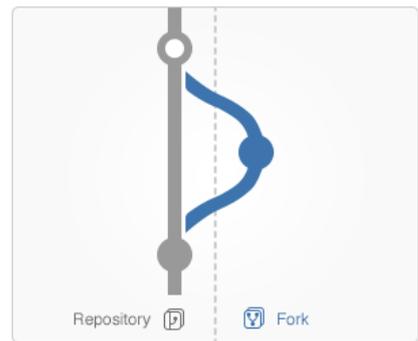
Fork

Develop features on a branch and create a pull request to get changes reviewed.



Discuss

Discuss and approve code changes related to the pull request.



Merge

Merge the branch with the click of a button.

Forks provide an alternative workflow to using branches, for where particular developers have restricted (read-only) access to a repository. See [Workflow strategies](#) for more information.

You can fork a repository into any other project in Bitbucket Data Center and Server for which you have admin access. You can also create [personal forks](#) and give other developers access to that using repository permissions.

Create a fork

You can create a fork for any repository that you can see in Bitbucket (that is, for which you have 'read' permission).

Simply click **Fork** in the sidebar. You can choose the location for the newly forked repository. Note that when a repository is forked into another project it will get that project's permissions, which may be less restrictive.

When creating the fork you can enable [fork syncing](#) to have Bitbucket automatically keep your fork up-to-date with changes in the upstream repository.

On this page:

- [Create a fork](#)
- [Delete forks to clear disk space](#)
- [Issue a pull request for a fork](#)
- [Merge a fork](#)
- [Synchronize with upstream](#)
- [Disable forking](#)
- [Pre-receive hooks and forks](#)

Related pages:

- [Workflow strategies](#)
- [Controlling access to code](#)
- [Creating personal repositories](#)

Delete forks to clear disk space

In order to clear any disk space on your Bitbucket Data Center instance, you should delete all related repositories, including any forks and upstreams. Head over to [Advanced repository management to learn more about related repositories](#), and to find out how to delete them all at once. (Data Center only)

Issue a pull request for a fork

Pull requests for forks in Bitbucket work just the way you'd expect. See [Pull requests](#).

When creating the pull request, you can choose the fork and the branch that contains the source to be pulled, as well as the destination fork and branch.

Merge a fork

Once a pull request has been approved by reviewers, it can be merged as usual. See [Pull requests](#).

Synchronize with upstream

Once you fork a repository, your fork can be kept up-to-date with changes in the upstream repo either automatically by Bitbucket or you can synchronize manually. You will still need to keep your remote working repository synced with your fork in Bitbucket yourself. See [Keeping forks synchronized](#) for more details.

Disable forking

The forking of repositories is available by default. However, you can turn off forking, on a per-repository basis, if this helps you to control your development process. You can do this on the **Repository details** tab of the repository settings.

Note that disabling forking on the parent repo doesn't delete any existing forks, and doesn't prevent those existing forks from being forked. Pull requests will still work from the existing forks. Furthermore, commits in the parent are viewable via the fork if the SHA1 hash is known to the user.

Pre-receive hooks and forks

Pre-receive hooks aren't copied with the fork and so are not run when code is merged in a pull request. This means that custom hooks are unable to prevent certain changes from being merged by pull requests from forks. Instead, the hook would have to also implement a merge check.

Keeping forks synchronized

Fork syncing helps you to keep your fork in Bitbucket Data Center and Server up-to-date with changes in the upstream repository. Bitbucket can do this automatically for all branches and tags you haven't modified in the fork.

If you have modified branches or tags in the fork, Bitbucket will offer syncing strategies. Bitbucket will never update your branch or tag in your fork if this means that your changes would be lost.

Note that syncing is about pulling recent upstream changes into your fork, whereas pull requests are about pushing your changes back to the upstream repository.

On this page:

- [Enabling automatic fork syncing](#)
- [What gets synced?](#)
- [Manual synchronization strategies](#)

Enabling automatic fork syncing

You can enable automatic fork syncing when you first fork the repository:

Fork Teams in Space / Apollo UI

Project* Admin Istrator

Where would you like to fork this repository into?

Name*

The repository's name will be used to create its URL
http://localhost:7990/scm/~admin/apollo-ui.git

Enable fork syncing

Keep branches and tags automatically in sync with Teams in Space / Apollo UI.

1. Enable fork syncing

You can also enable fork syncing at any later time by going to **Settings > Fork syncing** for the forked repository. Syncing is enabled by default.

What gets synced?

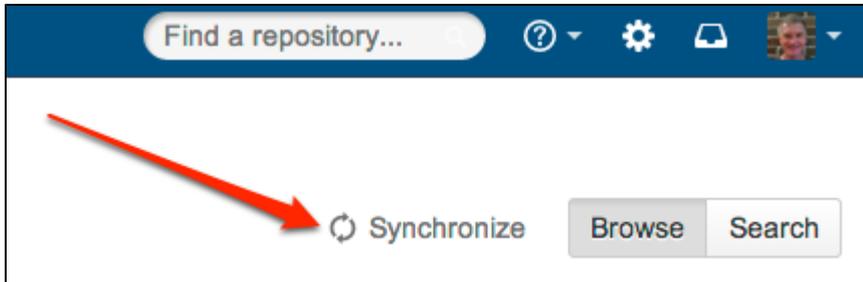
When performing automatic synchronization, Bitbucket updates the fork as follows:

- for branches - Bitbucket makes any fast-forward change, where there is no need to merge work and there is no risk of losing changes.
- for tags - Bitbucket makes updates only if the current state is the same as what upstream pointed to. So, a new tag in upstream will create a new tag in the fork, unless you have a tag of the same name, when the update will fail.

Manual syncing

If upstream and your fork have diverged, so that each has changes that are not in the other, Bitbucket will not perform a merge automatically. When you visit the branch, you have the option to manually synchronize the branch.

You can manually synchronize your branch at any time using **Synchronize** by going to the **Settings > Fork syncing** tab for the forked repository, or one of the **Source** or **Commits** tabs for a repository:

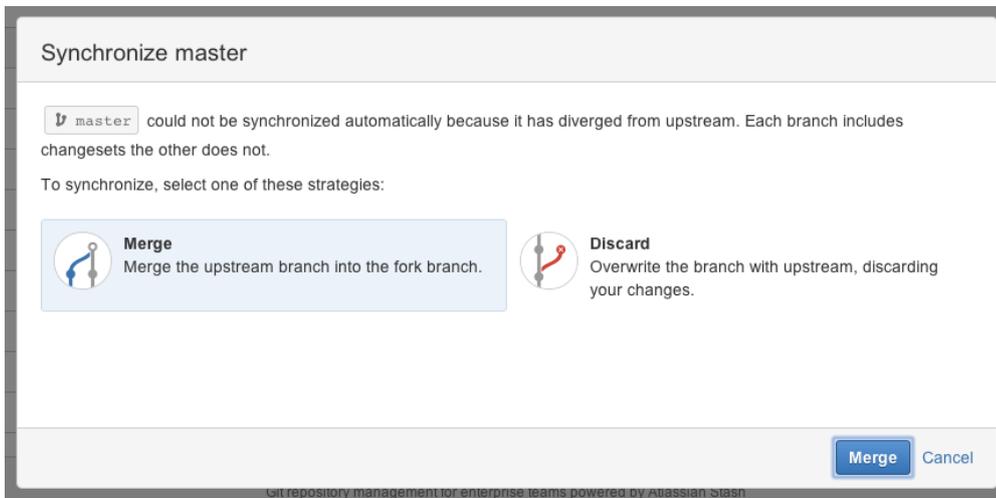


Manual synchronization strategies

When you initiate a manual synchronization, Bitbucket will ask you to choose one of the following synchronization strategies.

Merge strategy

Merge the upstream branch into the fork branch.



If Bitbucket detects conflicts when trying to perform the merge it will offer hints on how to resolve those:

Merge conflicts

Upstream changes could not be merged in automatically due to conflicts in the following file:

- CLI feature.txt

How to perform a manual merge:

Step 1: Fetch changes from the upstream repository (saving the upstream branch as FETCH_HEAD).

```
git fetch https://stash.dev.internal.atlassian.com/scm/dox/toolbox.git master
```

Step 2: Checkout the fork branch and merge in the changes from the upstream branch. Resolve conflicts.

```
git checkout master
git merge FETCH_HEAD
```

Step 3: After the merge conflicts are resolved, stage the changes accordingly, commit the changes and push.

```
git commit
git push https://stash.dev.internal.atlassian.com/scm/~pwatson/toolbox.git HEAD
```

Close

Once the merge is complete, your branch will have incorporated all the commits on the branch in the parent repository, but your branch will still be ahead of the parent (it has your changes on it). This means automatic synchronization for this branch will not occur until your changes are pushed to the parent repository.

Discard strategy

Overwrite your changes in your fork with the upstream branch. Your changes will be lost.

Synchronize master

`master` could not be synchronized automatically because it has diverged from upstream. Each branch includes changesets the other does not.

To synchronize, select one of these strategies:

 **Merge**
Merge the upstream branch into the fork branch.

 **Discard**
Overwrite the branch with upstream, discarding your changes.

 Discarding means your changes will be lost

Discard Cancel

Git repository management for enterprise teams powered by Atlassian Stash

Pull requests

Pull requests in Bitbucket Data Center and Server provide a quick and easy way for software teams to collaborate on code. A pull request is a dedicated forum for discussing a proposed feature. If there are problems with the changes, teammates can post feedback in the pull request and even tweak the feature by pushing follow-up commits. All of this activity is tracked directly inside of the pull request.

A pull request requires differences between two distinct branches. When you create a pull request you'll specify the branch to merge changes into.

Create a branch

Pull requests can be used with the [Feature Branch Workflow](#), the [Gitflow Workflow](#), or the [Forking Workflow](#). You can create branches from the Bitbucket UI, from the command line using Git, or from within a connected JIRA Software instance.

[Read about how to create a branch.](#)

Create a pull request

In their simplest form, pull requests allow a developer to notify team members that they've completed a feature. Once their feature branch is ready, the developer files a pull request via their Bitbucket account. This lets everybody involved know that they need to review the code and merge it into the `master` branch.

[Read about how to create a pull request.](#)

Review and discuss a pull request

As a reviewer of a pull request, your colleagues are counting on you to review changes to the code then provide feedback. Use reviewer status indicators to let a pull request author know you approve the changes or that changes need more work before you can approve.

To review a pull request, select either  **Approve** or  **Needs work** within the header of a pull request. Click the button again or click a different one to change your status.

Approving a pull request lets the author know you reviewed their changes and that you feel the work can be merged with the target branch.

Indicating a pull request **needs work** should accompany a comment to the author letting them know what should change before merging the pull request. Once the author pushes more changes to the pull request, revisit the pull request to review the new iteration. Bitbucket remembers what you've already reviewed and only shows you the changes made since your last visit. At any time you can choose to view the entire effective diff or individual commits and make comments there also.

[Read about how to review and discuss a pull request.](#)

Merge a pull request

Once you are ready to merge a pull request, and when the reviewers have approved it, click **Merge** at the top right of the pull request view. You can merge a pull request if you have write (or admin) [permission](#) on the project.

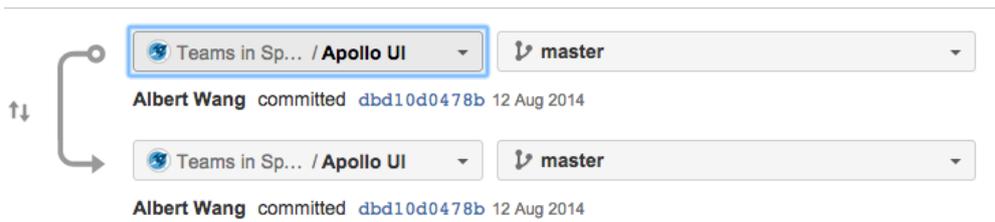
[Read about how to merge a pull request.](#)

Create a pull request

When you are ready to start a discussion about your code changes in Bitbucket Data Center and Server, it's time to create a pull request. A pull request is a dedicated forum for discussing proposed changes to a project. To create a pull request you'll need a feature branch that you've made changes to.

To create a pull request:

1. Select **Create pull request** in the sidebar.
2. Choose the source tag or branch and the destination branch. The source branch is where you made your code changes and the destination is the branch you want to merge to. The source and target branches may be located in different forks:



3. Use the Diff and Commits tabs to compare the source and destination branches before creating the pull request.
4. Select either **Create pull request** or **Continue**, and enter a title and description that will help people understand what your pull request is about. By default, the pull request title will contain the commit message or branch name. The description will populate with any existing commit history if no [description template](#) (Data Center only) is being used in the project. Use [mentions](#) (to notify another user), and [markdown](#) (to add formatting) in your description. If there are [contributor guidelines](#) available, have a look to make sure you're complying with the repository's owner's rules.
5. Add individual reviewers or [reviewer groups](#) users will receive a notification by email. Others who have [project permissions](#) can participate in the discussion if it interests them.
6. Select **Create**.

You will receive email notifications when your reviewers and other participants comment on the pull request or commit changes to it.

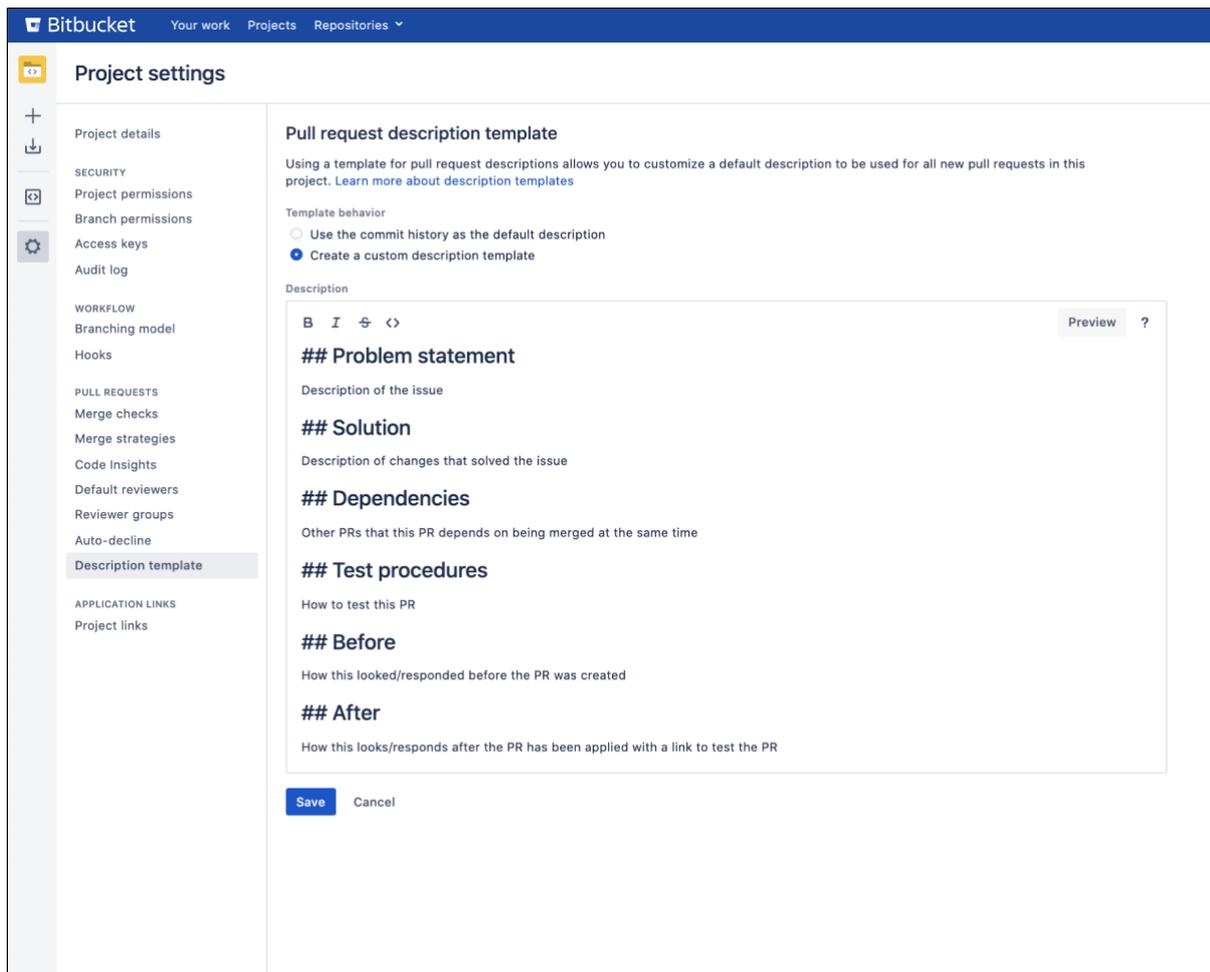
Edit a pull request

After creating a pull request, you can modify it by clicking **Edit** on the pull request's page. You can edit details such as the **Title**, **Description**, and **Reviewers**. In particular, you can change the **Destination** branch for the pull request you'll need [Read permission](#) on the branch you want to set.

Pull request description templates

 This feature is available with a Bitbucket Data Center license.

By default, pull request descriptions automatically populate with a list of relevant commit message history. The **Description** field is how pull request authors can set the context for a code review by adding images, links, or provide instructions for reviewers. Having a custom template for your pull request descriptions in Bitbucket Data Center will save time and help reviewers know what to expect while doing code reviews.



To create a description template:

1. Go to **Project settings** > **Description template** (under pull requests).
2. Select **Create a custom description template** for the **Template behavior**.
3. Fill out the **Description** field with the default information or leave it blank to set an empty description value for your template.
4. **Save** your changes.

The default description template will automatically be populated into the description field when a new pull request is created in the project. If you do not want to use a template for the description, you can set the **Template behavior** toggle to **Use the commit history as the default description**.

Reviewing a pull request

The review phase of a pull request in Bitbucket Data Center and Server typically involves reviewers making comments and the author pushing additional changes and commenting in response, until the pull request is ultimately approved. The pull request author usually starts by adding colleagues as reviewers. Reviewers then leave comments either on the entire pull request or on a specific part of the code changes, and then update their reviewer status to notify the author that they've completed their review.

Depending on the feedback provided by reviewers, the author may then update a pull request with new commits. This may be to clean up the code, resolve any outstanding tasks, or improve the quality of the code. They can also [apply suggestions](#) to quickly make code changes for minor issues like typos.

Once the cycles of reviewer feedback and new commits have reached a conclusion, a pull request can either be [merged to its target branch](#), or declined if the changes are not to be merged.

For details on how authors and reviewers can collaborate and discuss a pull request, see [Commenting on a pull request](#).

Review a pull request

Bitbucket allows you to add individual reviewers and [reviewer groups](#) (Data Center only) to a single pull request who can then approve or decline the request. Pull requests give those who have access to the repository, the ability to review the quality of the code that's specified in the pull request.

On this page:

- [Review a pull request](#)
- [Add a reviewer group to your pull request](#)
- [Create a Jira issue from a pull request](#)
 - [Overview](#)
 - [Diff view](#)
 - [Commits](#)
 - [View a single commit within a pull request](#)
 - [Find matching code in a pull request](#)
 - [Find files in a pull request](#)
 - [Watching and notifications](#)

The screenshot shows a Bitbucket pull request interface. At the top, it displays the source branch 'feature/BLUES-273-add-error-handling' and the target branch 'master', with an 'OPEN' status. There are 2 open tasks and a 'Merge' button. The pull request title is 'BLUES-273 - Handle errors when fetching dashboard details'. The overview tab is selected, showing the pull request was created by 'Matthieu Di Bernardino' yesterday. The 'Changes' section lists two items: 'Added new ServerError widget to show common errors with an illustration,' and 'Added handling of the error case when dashboard details can't be retrieved.' There are two screenshots showing a dashboard with a new error widget and a warning message. The 'Activity' section at the bottom has a text input field with the placeholder 'What do you want to say?'.

To review a pull request

1. **Access the pull request** by either following links from an email notification, selecting a notification within the pull request inbox (in the upper-right), or searching for a pull request by selecting **Pull requests** on the sidebar (read more about [searching for pull requests](#)).
2. **Review the changes** and comments left by your teammates within the pull request.
3. **Leave some feedback** about the changes, in any of the views, and use @ mentions to ask questions directly of your colleagues, who will receive a notification after you enter your comment. See [Commenting on a pull request](#) for more details about the various ways you can leave comments, including [pull request tasks](#).
4. **Finish your review** by indicating if you feel the pull request can be merged, or if the author of the pull request needs to make additional changes before you can provide your stamp of approval. Select either of the [status indicators](#) to let your team know you've reviewed the changes and the ball is now in their court:
 - **Approve** - indicates you've reviewed the changes and the code is ready to be merged.
 - **Needs work** - indicates you've reviewed the changes, but the code is not quite ready to be merged.

Add a reviewer group to your pull request

 This feature is available with a Bitbucket Data Center license.

You can add reviewer groups to your pull request by typing the name of an existing group into the **Reviewers** field. Each user that is a part of that group will then be added and display as individual users if they have project permissions. See [Reviewer groups for pull requests](#) for more information on creating and editing reviewer groups.

Create a Jira issue from a pull request

Bitbucket allows you to create a Jira issue directly from a pull request comment or task instead of being forced to leave Bitbucket and return with a created issue.

To create an issue from a pull request comment or task:

1. Create a new issue by clicking ... > **Create Jira issue**.
2. Select the **Project** and **Issue type**.
3. Add a summary, description, and any other fields as needed.
4. Click **Create issue**.

If a created issue is deleted inside Jira, it will continue to be counted as an issue in Bitbucket, but will no longer be visible on the pull request.

There are three main ways to view changes:

1. The **Overview tab** - lists all of the activity for a pull request since the pull request was created.
2. The **Diff view tab** - highlights which lines of code have been added, deleted, or modified.
3. The **Commits tab** - lists all the commits that will get merged. You can click to view a single commit.

Overview

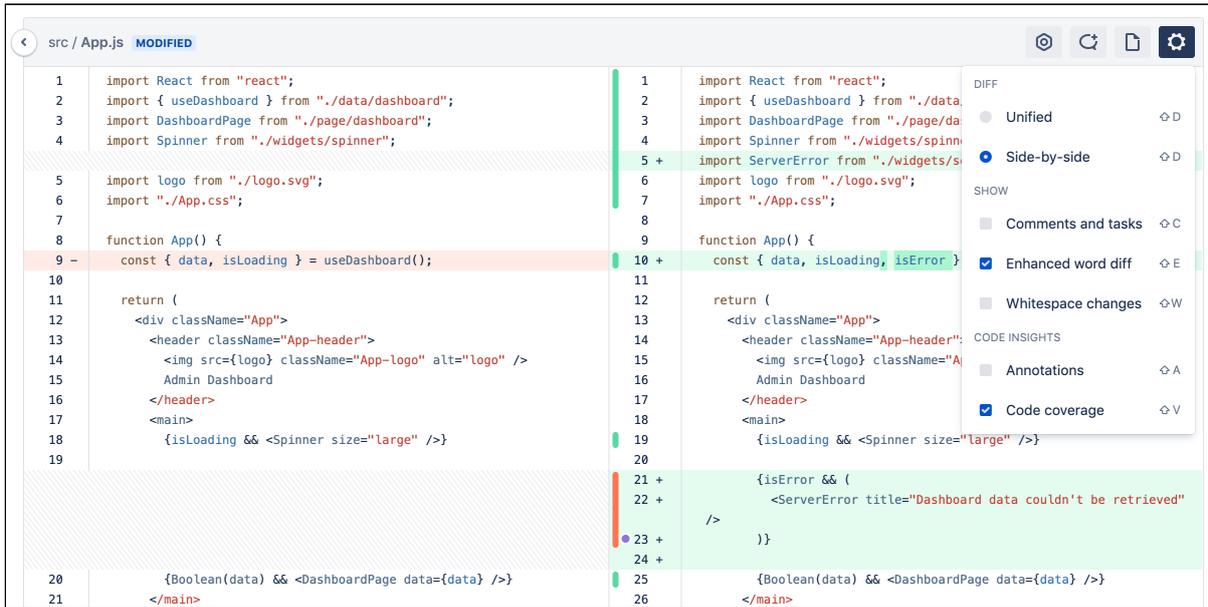
The overview tab captures all of the team's activity on the pull request in one place, right from the initial creation, through to when it is finally merged (or declined), with all the comments, replies, and commits that happen along the way.

Diff view

The diff view highlights the changes between the source and target branch. You can choose whether you want to highlight word level changes in the diff, or display details like comments and whitespace changes from the **Diff view settings**  icon.

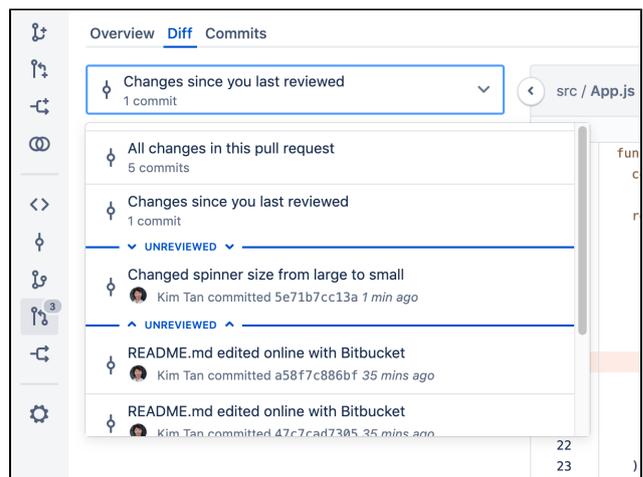
The **Unified diff** option shows the changes in one continuous column.

The **Side-by-side diff** lets you easily compare the changes by showing the original file on one side, highlighting removed and modified lines, and then changed files on the other side, highlighting added and modified lines.



Iterative reviews

Within the diff view, using the **Change Scope selector**, you can select a specific commit to review, or choose to view all changes within a pull request. If you return to a pull request that you previously reviewed, you'll only see the new commits added since you last reviewed the pull request.



Commits

The **Commit** tab lists all the commits that will get merged (those that are greyed out have already been merged). Clicking through to a commit leaves you inside the pull request context and the commit can be reviewed as part of the pull request.

When viewing a commit, you can comment on the whole file, or a particular line of code, for any file in the commit.

Participants can commit new changes to the branch. Bitbucket auto-updates the **Commit** tab of the pull request, so you can see exactly which commits will be merged. Bitbucket is smart about comments, moving them along when lines are added or removed. If a line with a comment gets removed, you can still view the comment in the activity, but Bitbucket marks the diff as *outdated* to let you know that this piece of code has been changed in recent commits. These hidden comments can also be viewed by selecting **other comments**.

View a single commit within a pull request

Author	Commit	Message	Commit date	Issues	Builds
Kim Tan	5e71b7cc13a	Changed spinner size from large to small	4 mins ago		
Kim Tan	a58f7c886bf	README.md edited online with Bitbucket	38 mins ago		
Kim Tan	47c7cad7305	README.md edited online with Bitbucket	38 mins ago		
Kim Tan	7ccf2b1a4c6	README.md edited online with Bitbucket	39 mins ago		
Kim Tan	078643dbdb1	README.md edited online with Bitbucket Modified readme file	39 mins ago		

Find matching code in a pull request

When you want to locate code in a pull request, you can search for it within changed files in the diff view.

To locate code within changed files, click **Search code** and enter your text into the search field. Only the files containing the match are then displayed expanded in the file tree with the number of occurrences highlighted per file. Easily navigate between occurrences by clicking through each line.

All changes in this pull request
1 commit

Filter file tree Search code

- App.js src 3
 - 12 <div className="App">
 - 13 <header className="App-header">
 - 14 <img src={logo} className="App-log...
- server-error.test.jsx.snap src/widgets/... 15
- server-error.jsx src/widgets/server-error 6
 - + 3 import classNames from 'classnames';
 - + 9 ...g: TitleTag = 'h4', className } =...
 - + 11 <div className={classNames('server...
 - + 14 <div className="server-error-messa...
 - + 15 <TitleTag className="server-error-...
 - + 26 className: string,
- server-error.test.jsx src/widgets/server-e... 1
 - + 27 ...hallow(<ServerError className="my...

```

src / widgets / server-error / server-error.jsx ADDED
1 + import React, { memo } from 'react';
2 + import { string } from 'prop-types';
3 + import classNames from 'classnames';
4 + import { I18n } from 'worm/i18n';
5 +
6 + import illustration from './error.svg';
7 + import './server-error.less';
8 +
9 + const ServerError = ( { title, message, ti
10 +   return (
11 +     <div className={classNames('serve
12 +       <img src={illustration} alt="
13 +
14 +     <div className="server-error-
15 +       <TitleTag className="serv
16 +         {title || I18n.getTex
17 +       </TitleTag>
18 +
19 +       <p>{message || I18n.getTe
20 +     </div>
21 +   </div>
22 +   );
23 + };
24 +

```

Find files in a pull request

To filter through changed files, click **Filter file tree** and enter the file name into the field. You can search through files in folders using wildcards.

Some common wildcards supported are *, ?, and **. For example; filtering on `**/test/**/*mock*`, reveals all files or folders within a directory called `test` that uses `mock` in their name. Note that matching is case-insensitive, so the filter will match both `text/one/my-mock-test.js` and `Test/two/MyMockTest.java`.

Watching and notifications

You automatically get added as a watcher of a pull request when you are added to the pull request as a reviewer, or when you perform an action related to the pull request (such as adding a comment):

Action	You're now a watcher
You are added as a reviewer	✓
You add yourself as a reviewer	✓
You comment on a pull request	✓
You reply to a comment	✓
You push to the source branch	✓

You can manually add yourself as a watcher by selecting **Watch** from the selection menu drop-down on the pull request screen.

You can stop watching a pull request by clicking the link in the email notification, or ... > **Unwatch** on the pull request screen. If you stop watching a pull request, you will not automatically be added as a watcher again if you subsequently perform an action that would otherwise have added you.

Bitbucket sends email notifications to watchers when certain [pull request events](#) occur. Email notifications are batched by default, but you can change your personal account settings (on the **Notification settings** tab) so that you get notifications immediately. The following notifications however, are always sent immediately to:

- the reviewers when a pull request is created
- all watchers when a pull request is deleted
- a user when they are added as a reviewer to a pull request
- a user when they are removed as a reviewer from a pull request
- a user when they are mentioned in the description of a pull request

See [Notifications](#) for details.

Add default reviewers to pull requests

Default reviewers allow you to automatically add one or more users as reviewers to pull requests in Bitbucket Data Center and Server. In addition, you can optionally specify how many of the specified default reviewers must approve a pull request prior to merging to ensure that a minimum level of review occurs.

Related pages:

- [Reviewing a pull request](#)
- [Pull requests](#)

You may have different roles in your team that should be automatically added to pull requests depending on the nature of the pull request. This can be achieved by assigning default reviewers for a specific repository, a specific branch, using a [branch pattern](#), or with a branch type from the [branching model](#). For example, you might specify a release manager be assigned to all pull requests targeting release branches in Bitbucket.

Once assigned in the repository settings, default reviewers will be pre-filled during pull request creation. At that time the set of reviewers can then be adjusted for each pull request.

Adding default pull request reviewers for all repositories in a project

To add default reviewers for pull requests(requires project admin permissions):

1. Go to **Project settings > Default reviewers**.
2. Click **Add default reviewers**.
3. For the **Source branch** and **Target branch** fields, select either **Branch name**, **Branch pattern**, **Branching model**, or **Any branch**.
 - a. Branch name - enter the name of an existing branch.
 - b. Branch pattern - use a [branch permission pattern](#) to match multiple branches.
 - c. Branching model - select the branch type to restrict access to. Read more about [branching models](#).
 - d. Any branch - add default reviewers for pull request coming from every branch (for Source branch) to any branch (for Target branch).
4. Enter the name(s) of who should be assigned as a reviewer in the **Default reviewers** field.
5. *Optional:* In the **Approvals required** field, select how many reviewers must approve pull requests that match this criteria before merging.
6. Click **Create**.

Adding default pull request reviewers for a single repository

To add default pull request reviewers for a single repository(requires repo admin permissions):

1. Go to **Repository settings > Default reviewers**.
2. Click **Add default reviewers**.
3. For the **Source branch** and **Target branch** fields, select either **Branch name**, **Branch pattern**, **Branching model**, or **Any branch**.
 - a. Branch name - select an existing branch by name.
 - b. Branch pattern - use a [branch permission pattern](#) to match multiple branches.
 - c. Branching model - select the branch type to restrict access to. Read more about [branching models](#).
 - d. Any branch - add default reviewers for pull request coming from every branch (for Source branch) to any branch (for Target branch).
4. Enter the name(s) of who should be assigned as a reviewer in the **Default reviewers** field.
5. *Optional:* In the **Approvals required** field, select how many reviewers must approve pull requests that match this criteria before merging.
6. Click **Create**.

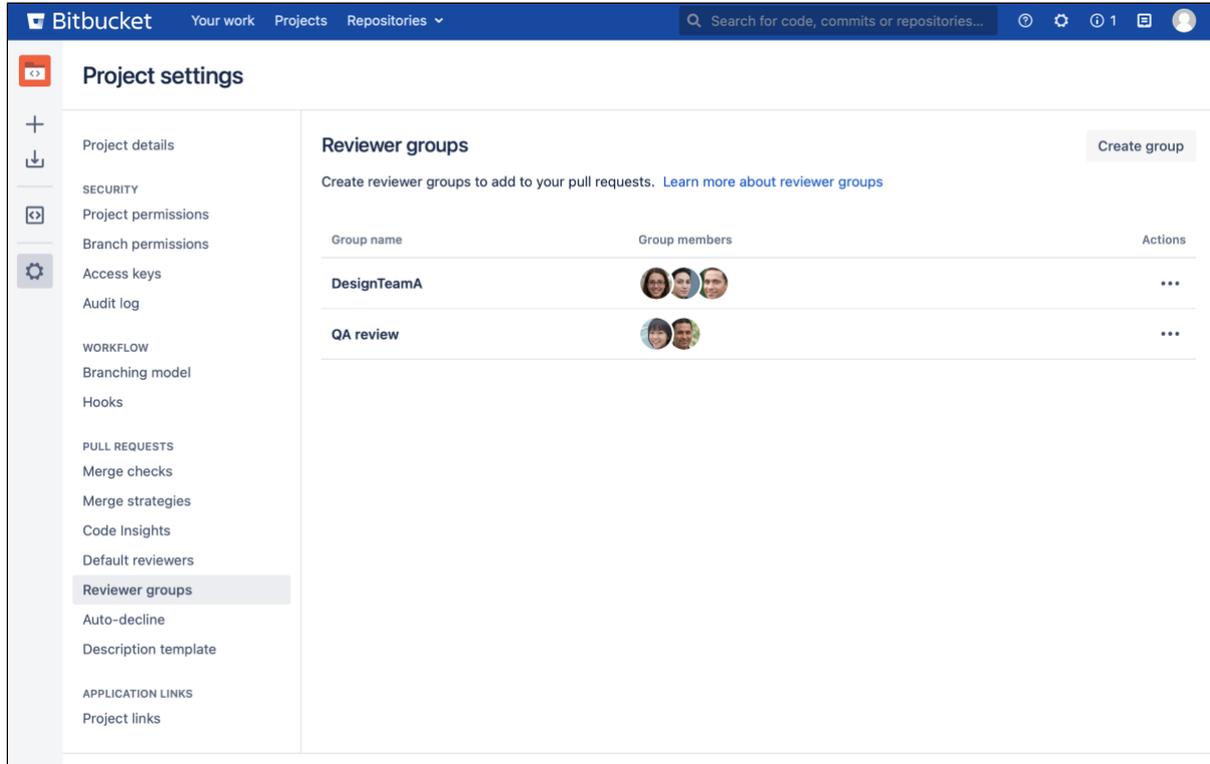
Reviewer groups for pull requests

 This feature is available with a Bitbucket Data Center license.

Related pages:

- [Reviewing a pull request](#)
- [Create a pull request](#)

Reviewer groups in Bitbucket Data Center, allow you to quickly add pre-defined groups of reviewers to pull requests. Project and repository admins can create and manage reviewer groups.



To create a reviewer group for a project or repository:

1. Select  **Reviewer groups** (under pull requests).
2. Select **Create group**.
3. Provide a name for your new reviewer group.
4. Search for users to add to the reviewer group.
5. Select **Create**.

To add a reviewer group to your pull request, start by typing in the name of the group into the **Reviewers** field. When you select a group, group members are added to the field as individual users.

Your browser does not support the HTML5 video element

Reviewer groups and permissions

Reviewer groups won't impact permissions. When a group member's permission is removed, they will still display as a member of the reviewer group, but will no longer have access to the associated code.

You can't add a user to the reviewer group if they don't have access to the project or repository.

When a repository with reviewer groups is moved to a new project, the reviewer group and all members that have access to the project are moved along with the repository. If no group members have access to the project, the reviewer group will still move across to the new project and will display in a search, but group members will not be added to the pull request.

Commenting on a pull request

The most important aspect of a pull request is the discussion it generates. You can comment on the entire pull request, a particular file, or on specific lines of code in a file to generate discussion relevant to your code review. You can also [create a task](#) or convert any comment to a task, so actions identified during the review can be easily tracked and resolved. Another efficient way to collaborate is to start a review, allowing you to leave multiple comments or tasks without sending them off to the author until you are ready to publish them.

For more on what you can do in a pull request and an overview to the code review workflow itself, see [Reviewing a pull request](#).

Comments

Places in Bitbucket Data Center and Serverwhere you can make comments:

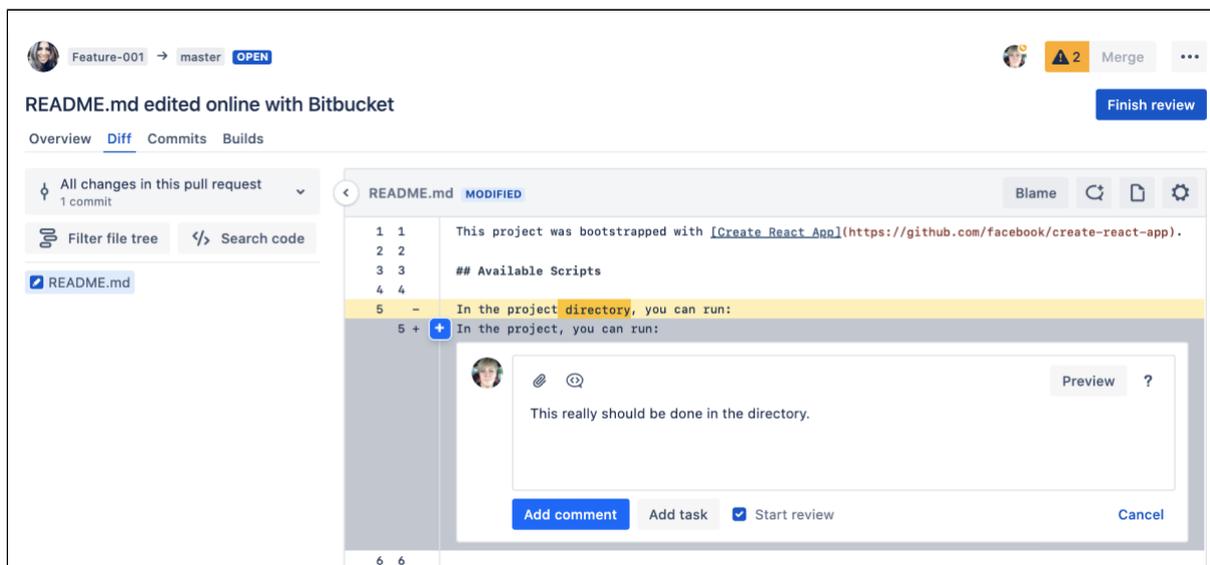
- **Overview tab** - you can add a comment on the Overview tab (just under 'Activity'), or reply to a previous comment. Use mentions to alert another Bitbucket user to your comment, and use Markdown to add formatting, for example, headings or lists.
- **Diff view tab** - display and create comments for a file directly on lines of code for commits and pull requests.

Draft multiple comments during a review process

When you are reviewing a pull request and you want to make multiple comments or tasks across various lines of code or files before the pull request is merged, you can manage your workflow more effectively by starting a review rather than adding your comments and submitting them one at a time.

To start a review:

1. From the pull request you are reviewing, begin by selecting the **Start review** button in the top, right side of the page. You can also start a review from the comment form by selecting the **Start review** checkbox.
2. Type your comment and then select **Add comment**.
3. Add any other additional comments or tasks to your review.

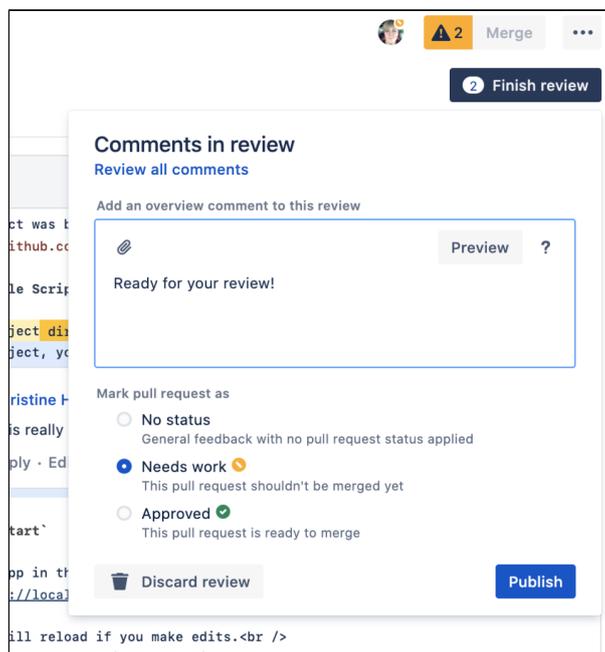


i While in a review:

- comments are pending and only visible to you
- all comments added will be part of the review
- the author will be notified only when you publish the review

To publish a review:

1. Select **Finish review** to open the **Comments in review** dialog.
2. From here, you can review all of your feedback and add an overview comment.
3. When you are ready, select **Publish** to send the review to the author.



i If you want to delete all of the pending comments and tasks without publishing, use **Discard review**.

Tasks

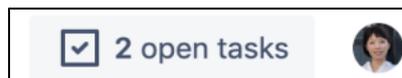
You can create a task on the entire pull request, a particular file, or on specific lines of code in a file to track required work identified during a code review. Convert a task into a comment, or a comment into a task. Anyone with repository read permission can convert any other user's comments and tasks (and vice versa). A repository administrator can delete other user's comments or tasks and can also enable a merge check that requires all tasks to be resolved before the pull request can be merged. See [Checks for merging pull requests](#).

To create a pull request task:

1. Select **Add a comment**  or the **+**  icon on a line of code
2. Add your text.
3. Select **Create task**.

When writing a task, you can use markdown to add formatting, images, and attachments to your tasks.

To see all the open and resolved tasks for a pull request:



1. Use **Shift+T** on the pull request page, or click the **Open tasks list** button.
2. Select the **'View on'** linked text in the **Tasks** window to see a task in the context of where it was created.

To resolve tasks for a pull request, **select the checkbox** next to the open task.

Suggestions

As a reviewer of a pull request in Bitbucket, you can suggest a small change to the code by leaving a suggestion right inside the comment or task itself. If you have write access to the source repository, you can commit the suggested change directly in the pull request without further action.

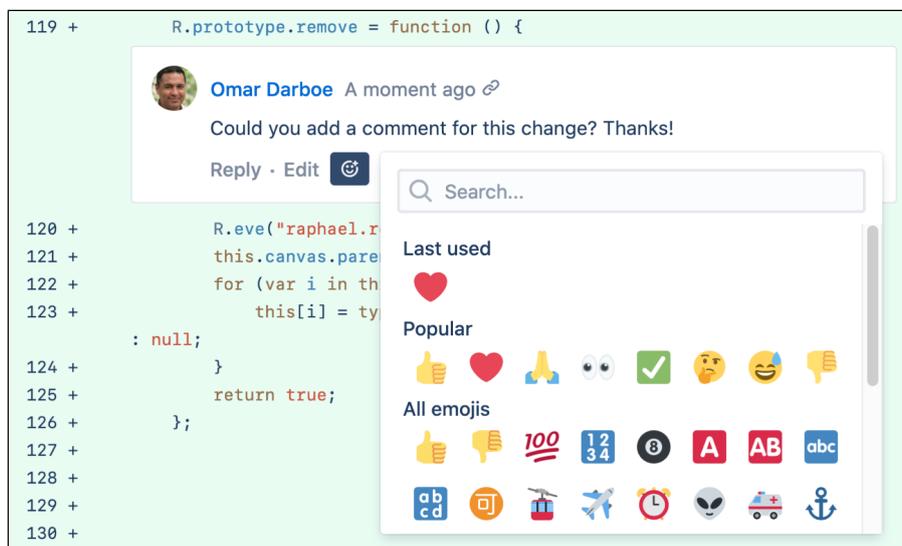
To create a pull request suggestion on a single line of code:

1. In the comment dialog, click .
2. Type your suggestion in the code block. You can also add any feedback outside of that code block.

To apply a pull request suggestion:

1. Select **Apply suggestion**.
2. Edit the commit message if required and then select **Commit changes** to add a new commit to the pull request.

React to comments



Sometimes there's nothing more to say, and a  or  is all you need when replying to a comment.

To react to a comment with an emoji, select the **Add reaction** button , then select an emoji.

To remove an emoji from a comment, select it and it will disappear.

Viewing other comments in Bitbucket

When updating a pull request, comments that are in older diffs or that have become outdated due to a pull request update will become hidden. To view them, click the **other comments** counter button at the top of the page to open a dialog box with more context as to why code has changed throughout a pull request. You'll be able to:

- see a file's activity stream showing comments that are outdated or appear on another diff
- distinguish which comments are actually outdated
- reply to, like, delete, or react to outdated comments the same way you can from the overview tab

Merge a pull request

Once you are ready to merge a pull request, and when the reviewers have approved it, click **Merge** at the top right of the pull request view. You can merge a pull request if you have write (or admin) [permission](#) on the project.

Bitbucket Data Center and Server does not enforce particular review workflows, so anyone with write permission on the repository can merge a pull request, including the person who opened it. This flexibility allows different teams to have different approaches to using Bitbucket. If your team requires stricter control, consider using [branch permissions](#) to restrict who can merge a pull request to particular users or groups. You might also want to consider using a plugin to enforce a particular workflow, for example to ensure that only approvals from members of your review team allow merging. See the page [Checks for merging pull requests](#) for details about enabling merge checks.

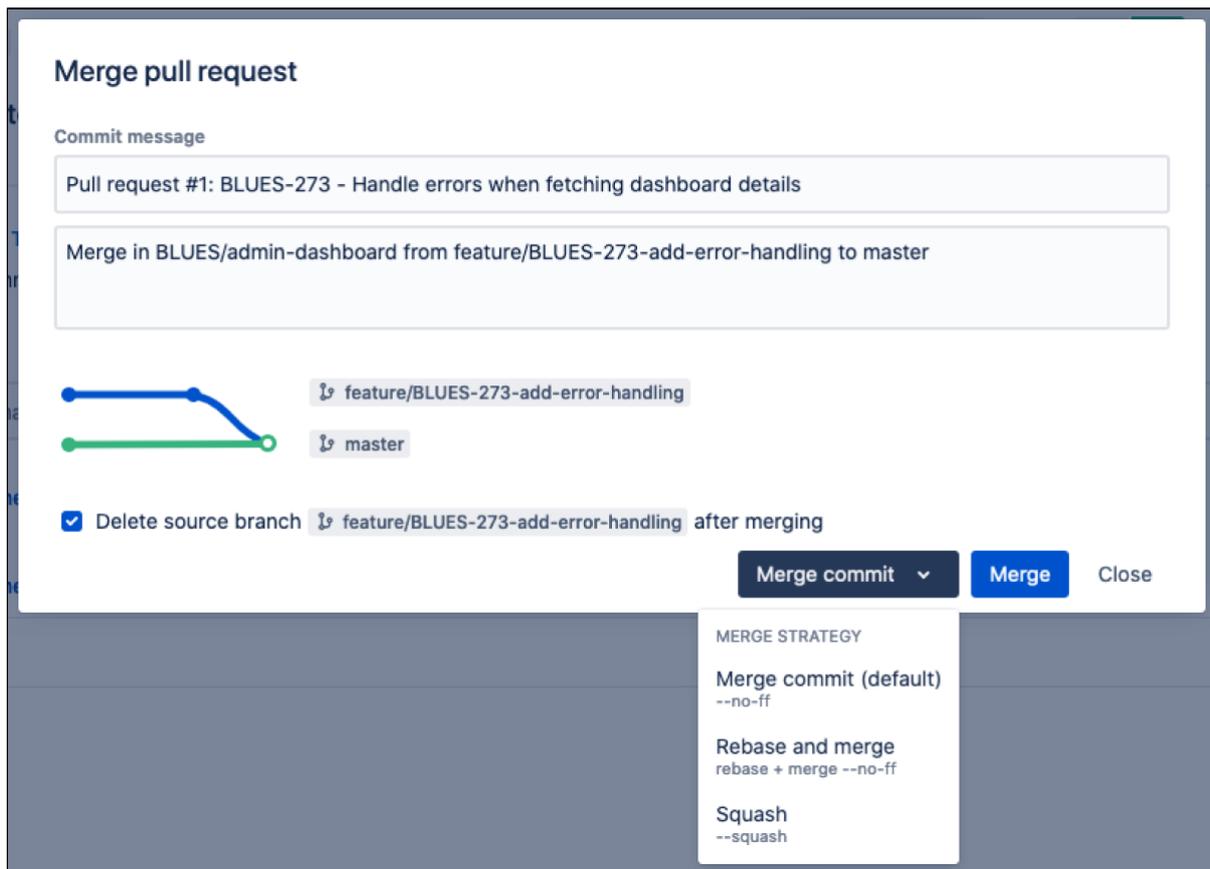
In the *Merge pull request* dialog, you can add information about the pull request in a comment. The text you add appears between the subject line and the log lines that Bitbucket and Git generate, and adds the text to the commit message for the merge.

On this page:

- [Delete source branch after merging](#)
- [Choose a merge strategy](#)

Related pages:

- [Pull request merge strategies](#)
- [Automatic branch merging](#)
- [Branches](#)



The *Merge pull request* dialog

Delete source branch after merging

When merging a pull request, you can choose to **Delete source branch after merging**. This setting is selected by default to help you keep your branch list clean. If this isn't your preferred setting, you can change it, and your preference will be saved for next time you merge a pull request.

Before deleting your source branch, Bitbucket will do some checks. The branch being merged will not be deleted if:

- The branch is the default repository branch.
- The user does not have permission to delete the branch.
- The branch is subject to an open pull request.

Once accepted, the pull request is marked as merged on the *Pull request* tab. If Bitbucket detects a conflict that prevents the merge, notifications are displayed on the *Overview* and *Diffs* tabs of the pull request. Click **More information** to see instructions for how to resolve the conflict in your local repository.

Choose a merge strategy

[Git merge strategies](#) affect the way the Git history appears after a merge has occurred. You can choose a merge strategy from the *Merge pull request* dialog. Administrators can configure which merge strategies are available and which merge strategy will be the default.

To change the merge strategy when merging a pull request, click the merge strategy in use (next to the *Merge* button), then select a new one.

The merge strategies available in Bitbucket are:

- **Merge commit** (`--no-ff`) DEFAULT: Always create a new merge commit and update the target branch to it, even if the source branch is already up to date with the target branch.
- **Fast-forward** (`--ff`): If the source branch is out of date with the target branch, create a merge commit. Otherwise, update the target branch to the latest commit on the source branch.
- **Fast-forward only** (`--ff-only`): If the source branch is out of date with the target branch, reject the merge request. Otherwise, update the target branch to the latest commit on the source branch.
- **Rebase and merge** (`rebase + merge --no-ff`): Commits from the source branch onto the target branch, creating a new non-merge commit for each incoming commit. Creates a merge commit to update the target branch. The PR branch is **not** modified by this operation.
- **Squash** (`--squash`): Combine all commits into one new non-merge commit on the target branch.
- **Squash, fast-forward only** (`--squash --ff-only`): If the source branch is out of date with the target branch, reject the merge request. Otherwise, combine all commits into one new non-merge commit on the target branch.

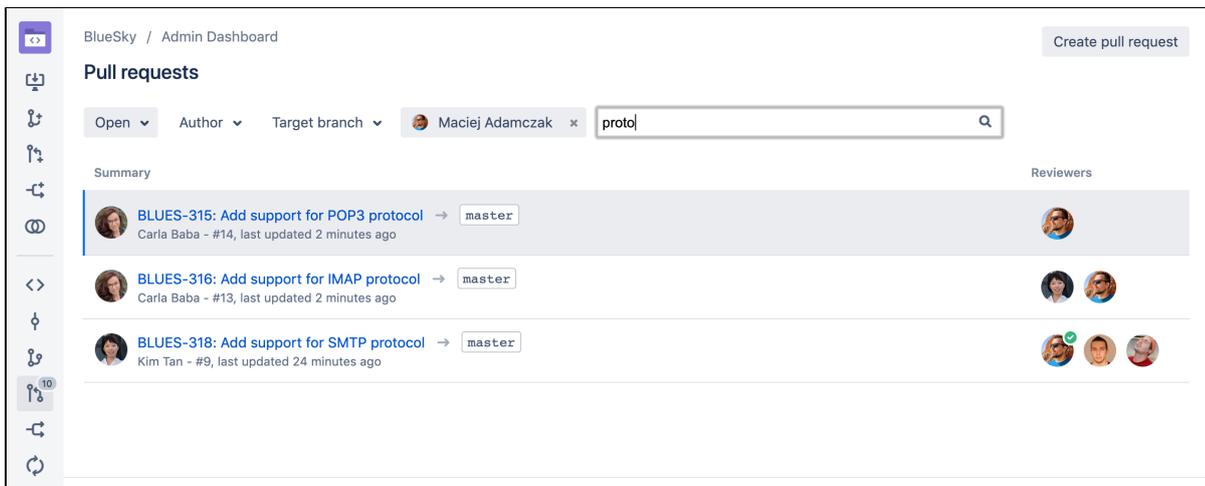
Search for pull requests

In Bitbucket Data Center and Server, you can easily find pull requests in a repository by using the following filters available on the pull request page:

- status
- author (the person who created the pull request)
- branch
- reviewer (the person who reviewed the pull request)
- text that is in the title or description

To find pull requests relating to your search criteria:

1. In the repository, click  on the sidebar to go to the Pull requests page.
2. Choose your criteria.



Checks for merging pull requests

Pull requests provide a way to do peer code reviews and merges as part of a branch-based development workflow. As your team grows, you may need to set restrictions about when pull requests can be merged to protect your production code and keep code quality high. Merge checks can help you do this.

On this page:

- [Merge checks](#)
- [Code Insights merge checks](#)
- [Required builds merge check](#)

Merge checks

Merge checks stop pull requests from being merged until they meet requirements that you've set. Your requirements can be based on a range of things, including the number of reviewers who have approved the pull request, or the [result of a Code Insights report](#). This ensures that pull requests are fully vetted before they're merged. It also helps to avoid the problem of code review blockages and the need to completely lock down a repository.

Default merge checks

Bitbucket Data Center and Server comes with some default merge checks. These can be enabled (or disabled) at the project level for all repositories in a project, or for individual repositories.

The default merge checks that come with Bitbucket are:

- **All reviewers approve** - requires all reviewers to approve a pull request before merging.
- **Minimum approvals** - requires at least the specified number of approvals before merging.
- **Minimum successful builds** - requires at least the specified number of successful builds before merging.
- **No incomplete tasks** - requires all tasks to be complete before merging.



In addition to **minimum approvals** for a pull request, you can also set **default reviewers**. If both of these checks have been configured, they both need to be met in order to merge. In other words, meeting the minimum number of approvals won't be enough to merge if the default reviewers have not approved the pull request.

Configure merge checks for all repositories in a project

Enabling (or disabling) merge checks at the project level changes merge checks for repositories set to inherit project settings. If you previously changed merge checks for an individual repository, that repository's configuration will not change when configuring merge checks at the project level.

To enable (or disable) merge checks for repositories in a project (requires project admin permissions):

1. Go to **Project settings > Merge checks**.
2. Click the toggle by the merge check to enable (or disable) it.

Merge checks for repositories set to **Inherited** in the project will now reflect this new configuration. Merge checks explicitly configured at the repository level will not be affected.

Configure merge checks for an individual repository

Enabling (or disabling) merge checks at the repository level enables a merge check for all pull requests created in that repository. Configuring merge checks at the repository level will override any checks configured at the project level. If you have not configured merge checks for an individual repository it will inherit merge checks configured at the project level.

To enable (or disable) merge checks for a single repository (requires repository admin permissions):

1. Go to **Repository settings > Merge checks**.
2. Use the drop menu to the right of the merge check to set it.

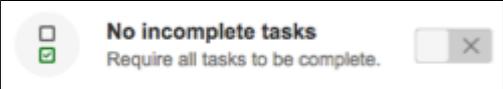
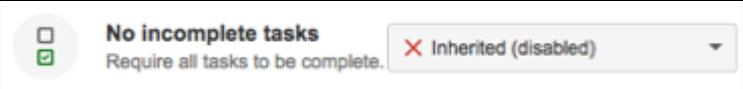
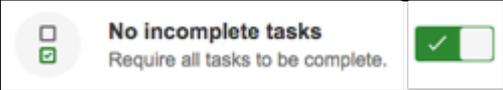
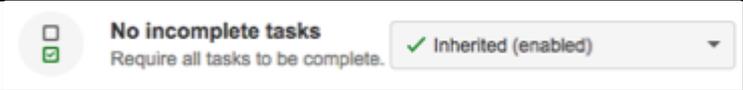
- a. **Inherited** - uses the configuration set at the project level.
- b. **Enabled** - enforces the conditions of the merge check.
- c. **Disabled** - ignores the conditions of the merge check.

Once set, any changes made to a merge check configuration at the project level will be ignored for this repository because it was changed independent of the project configuration.

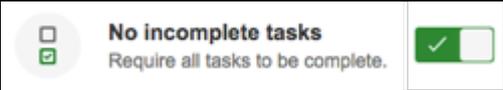
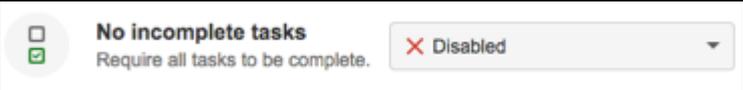
Inherited merge check configurations

By default, Bitbucket comes with merge checks disabled at the project and repository level. Unless merge checks were configured at the repository level, enabling or disabling merge checks at the project level inherits the configuration at the repository level.

For example, if you enabled the *No incomplete tasks* merge check for a project, and a repository merge check configuration was unchanged, each repository would have the *No incomplete tasks* merge check enabled.

<p>Merge check disabled, <i>project level</i></p> 	<p>Merge check disabled, <i>repository level</i></p> 
<p>Merge check enabled, <i>project level</i></p> 	<p>Merge check enabled, <i>repository level</i></p> 

Now suppose you decide that the *No incomplete tasks* merge check isn't appropriate for one specific repository. You can change that individual repository's merge checks independent of how it's configured at the project level. Any changes made to merge check configuration at the project level for the *No incomplete tasks* will be ignored for this repository, because it was changed independent of the project configuration.

<p>Merge check enabled, <i>project level</i></p> 	<p>Merge check disabled, <i>repository level</i></p> 
---	--

Add a new merge check

Additional merge checks can be installed by system administrators and can also be enabled for all repositories in a project, or for individual repositories.

To add merge checks from the Atlassian Marketplace (requires system admin permission):

1. Go to **Project settings > Merge checks**.
2. Click **Add merge check**.
3. Search for a merge check to add, and click **Install**.

Once you add a new merge check, you can enable (or disable) it in the same way as the default merge checks.

Create a merge check

You can also write your own [merge request check plugin](#).

Code Insights merge checks

You can block pull requests from being merged until their [Code Insights](#) reports meet your requirements. You can set these merge checks for all the repositories in a project, or a single repository, and your requirements can be based on whether:

- a specific report is present
- the report passes or fails
- the report adds annotations of a certain severity to the diff

 Annotations only block merging if they are on the diff. If they're somewhere else on a changed file, or on a file that hasn't changed they won't block the merge.

To add a Code Insights merge check for all the repositories in a project (requires project admin permissions):

1. Go to **Project settings > Code Insights**.
2. Enter the report key of your required report. You can find this on the report in the bottom right.
3. Enter its required status and its annotation requirements.
4. Click **add**.

To add a Code Insights merge check for a repository (requires repository admin permissions):

1. Go to **Repository settings > Code Insights**.
2. Enter the report key of your required report. You can find this on the report in the bottom right.
3. Enter its required status and its annotation requirements.
4. Click **add**.

Required builds merge check

 This feature is available with a Bitbucket Data Center license.

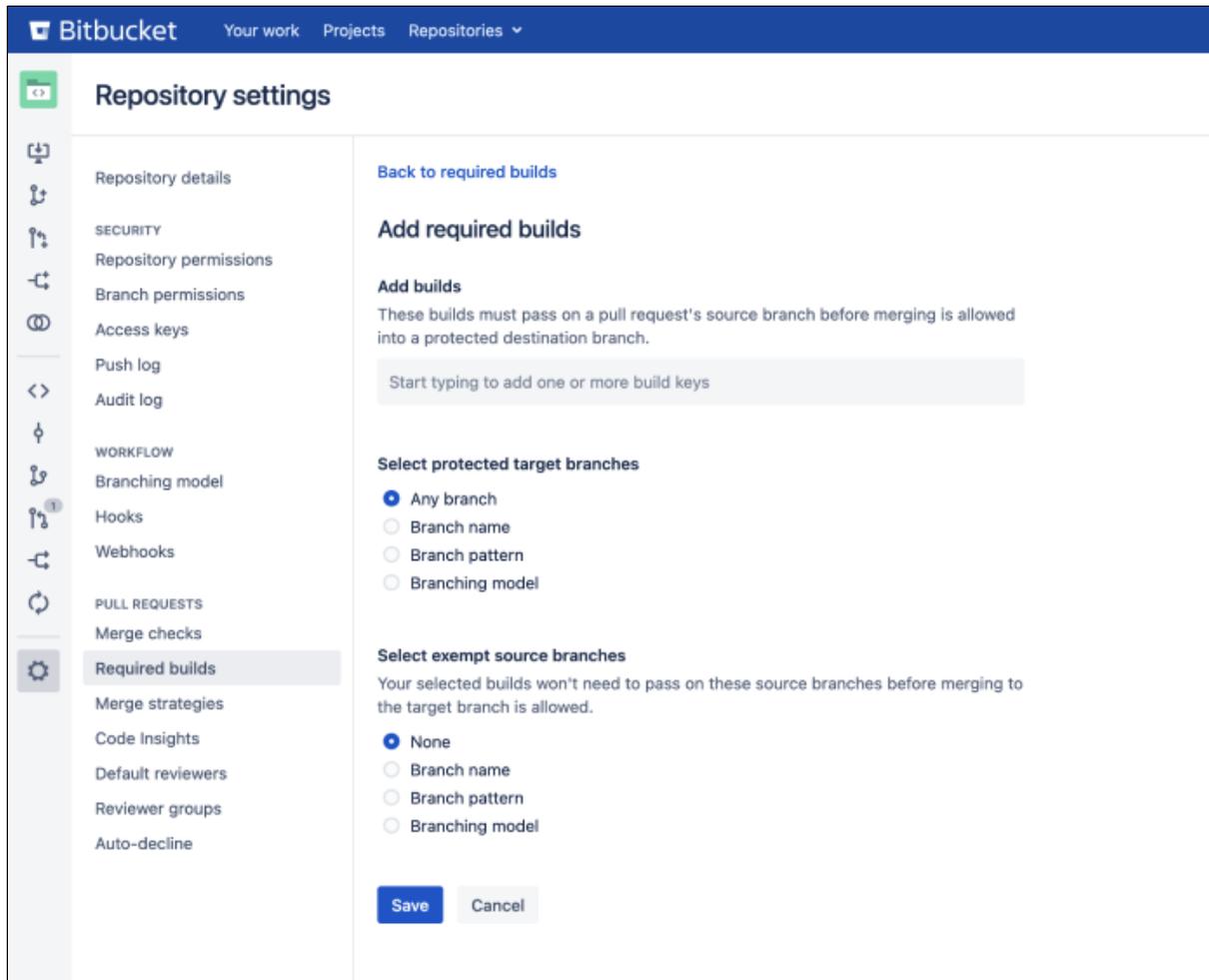
While the **minimum successful builds** merge check lets you require at least the specified number of successful builds, you can use the **required builds** merge check for requiring specific builds to successfully pass before a pull request can be merged into specified target branches.

Using required builds provides a simple but powerful interface to ensure code quality by protecting critical branches in your repository.

 To use required builds, you must have Bamboo 7.1+ or the Bitbucket Server Integration Plugin version 2.0.0+. See [Link your CI server for more details on integration guides](#).

To add a required build merge check for pull requests in a repository (requires repository admin permissions):

1. Go to **Repository settings**  **> Required builds**.
2. Select **Add required builds**.
3. From the **Add required builds** page, input the details as described in the table below.
4. Select **Save**.



Field	Description
Add builds	<p>Add any build keys here that will need to successfully pass before merging a pull request. The build key will depend on the CI tool used to generate the build.</p> <p>For example;</p> <ul style="list-style-type: none"> • In Bamboo, you may have a <i>project</i> key called PROJ, and a <i>plan</i> key called COREBUILDS, so then the build key in this field would be PLAN-COREBUILDS. • The build key of a Jenkins job is the name of each item in the path to the build, separated by a forward slash. For example; a folder called PROJ that contains a multibranch pipeline job called COREBUILDS, then the build key would be PROJ /COREBUILDS.
Select protected target branches	<p>Choose from a list of branches. Any pull request with a target branch that matches this setting will need to have successful builds on the latest commit before it can be merged.</p> <p>You can set this merge check to apply to Any branch in the repository, or you can be more specific by choosing:</p> <ul style="list-style-type: none"> • Branch name - a specific branch by name • Branch pattern - a pattern syntax for matching branch names • Branching model - any branch types that you have enabled will display here

Select exempt source branches	<p>In some cases, you may want to add specific source branches that wont need to pass in the selected required build(s).</p> <p>For example; you could allow urgent fixes to skip build checks by adding the Branch pattern <code>hotfix-*</code>, which would allow any branch starting with <code>hotfix-</code> to be merged without a successful build.</p> <p>Leave this option set to None if you dont want to add an exemption, otherwise choose from the following:</p> <ul style="list-style-type: none">• Branch name - a specific branch by name• Branch pattern - a pattern syntax for matching branch names• Branching model - any branch types that you have enabled will display here
--------------------------------------	---

Pull request merge strategies

Conflicts can happen with any of these merge strategies. When a conflict occurs, Bitbucket Data Center and Server will leave the repository as it was before attempting to apply the merge. To resolve such conflicts, check out the target branch locally and attempt to apply the rebase. The Git client can then facilitate resolution of these conflicts, finalize the local rebase and be pushed to the server. At that point you may wish to manually fast-forward the target branch, or simply attempt the pull request merge again using the web interface.

[Git merge strategies](#) affect the way the Git history appears after merging a pull request. With Bitbucket, you can choose which merge strategies to allow, and enable one or more merge strategies for all repositories in a project or for an individual repository. You can also allow users to choose a merge strategy from the merge dialog when they create a pull request. This page describes how to configure which merge strategies are available to your users, and briefly describes the merge strategies available.

On this page:

- [Merge strategies](#)
- [Using rebase in Bitbucket](#)
- [Change the merge strategy for a single pull request](#)
- [Configure merge strategies for all repositories in a project](#)
- [Configure merge strategies for an individual repository](#)

Merge strategies

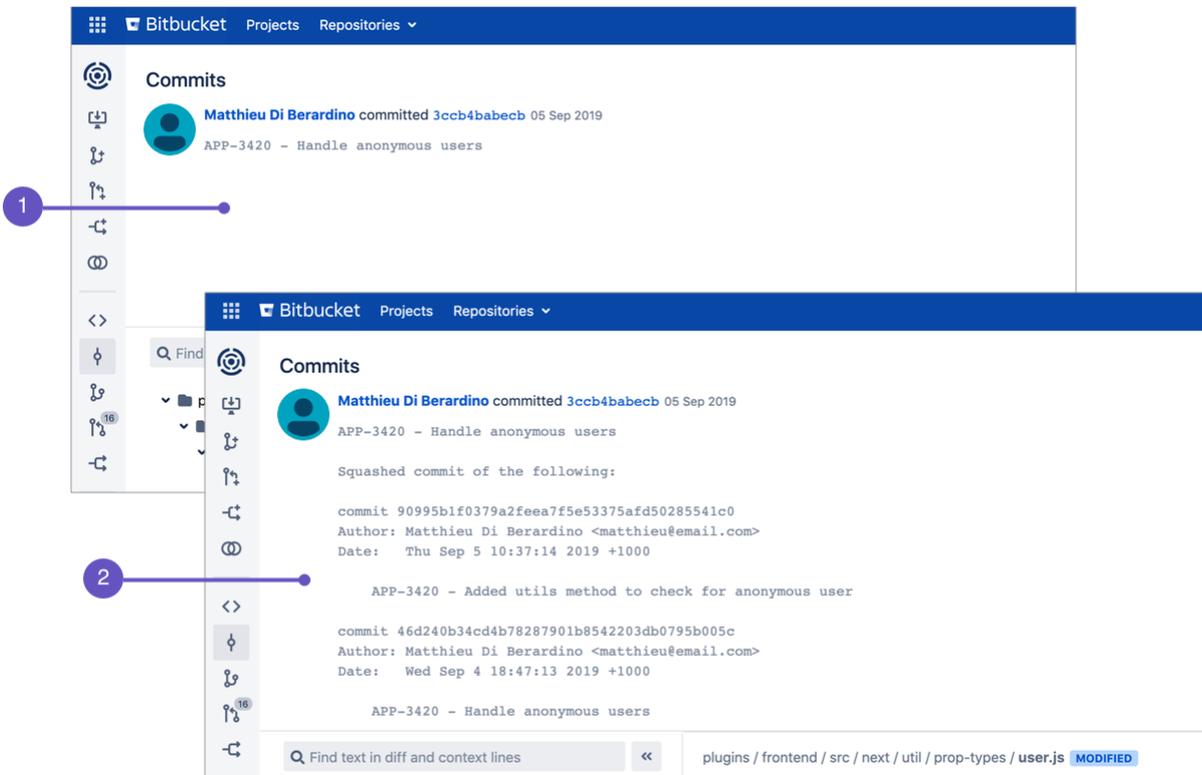
Bitbucket comes with only one merge strategy enabled by default, but all merge strategies can be enabled (or disabled) at the project level for all repositories in a project or for individual repositories, provided merge strategies for a repository are configured to inherit project settings.

The merge strategies available in Bitbucket are:

- **Merge commit** (`--no-ff`) DEFAULT: Always create a new merge commit and update the target branch to it, even if the source branch is already up to date with the target branch.
- **Fast-forward** (`--ff`): If the source branch is out of date with the target branch, create a merge commit. Otherwise, update the target branch to the latest commit on the source branch.
- **Fast-forward only** (`--ff-only`): If the source branch is out of date with the target branch, reject the merge request. Otherwise, update the target branch to the latest commit on the source branch.
- **Rebase, merge** (`rebase + merge --no-ff`): Commits from the source branch onto the target branch, creating a new non-merge commit for each incoming commit. Creates a merge commit to update the target branch. The PR branch is **not** modified by this operation.
- **Rebase, fast-forward** (`rebase + merge --ff-only`): Commits from the source branch onto the target branch, creating a new non-merge commit for each incoming commit. Fast-forwards the target branch with the resulting commits. The PR branch is **not** modified by this operation.
- **Squash** (`--squash`): Combine all commits into one new non-merge commit on the target branch.
- **Squash, fast-forward only** (`--squash --ff-only`): If the source branch is out of date with the target branch, reject the merge request. Otherwise, combine all commits into one new non-merge commit on the target branch.

Commit summaries

When you merge a pull request, you can control the number of commit summaries included in commit messages with the **Commit summaries** option. You can do this for a single repository or all repositories in a project. This setting is helpful if you don't want to see as many commit details in the commit message when merging or squashing a pull request.



1. Commit with no summaries.
2. Commit with two summaries.

Commit summaries

Set the maximum number of commit summaries to append to the commit message when a pull request is merged. Set to zero to exclude commit summaries.

Max summaries

To set the maximum number of commit summaries for a single repository(requires repository admin permissions):

1. Go to **Repository settings>Merge strategies**.
2. Enter the maximum number of commit summaries to include. Specify zero to display no summaries.

To set the maximum number of commit summaries for a project (requires project admin permissions):

1. Go to **Project settings>Merge strategies**
2. Enter the maximum number of commit summaries to include. Specify zero to display no summaries.

Using rebase in Bitbucket

Rebasing allows you to replay feature branch commits onto the tip of your target branch, creating a linear history. By rebasing your commits to the tip of the target branch, you retain existing commits, and simply add yours on top.

Potential issues

When using a rebase workflow, it's possible to encounter unexpected side effects if branches aren't carefully managed and tracked. For example:

- If you have a branch that uses a previously merged branch (using rebase) as a common ancestor, further merging (using rebase) can cause duplicate commits with empty content to be applied to a target branch.
- When running `git pull` locally after a pull request's source branch has been rebased using the UI, it can result in unexpected merges between the original commits (still present on the local branch), and their rebased replacements, (fetched from the server). When pulling, if the tracking branch has been updated on the remote, by default Git will merge the incoming changes with the local commit.

Conflicts

Conflicts can happen with any merge strategy. When a conflict occurs, Bitbucket will leave the repository as it was before attempting to apply the merge.

To resolve such conflicts, check out the target branch locally, and attempt to apply the rebase. The Git client can then facilitate resolution of these conflicts, finalize the local rebase, and be pushed to the server.

At that point you may wish to manually fast-forward the target branch, or simply attempt the pull request merge again using the web interface.

Transactional failures

A transactional failure occurs when a two-step process (ie rebase, then merge) is taking place and other changes are made at the same time that prevent the second step from applying as expected. The impact for you means on occasion, if you try to rebase, it will fail, and won't apply the merge. You will receive a transactional failure notification if this happens.



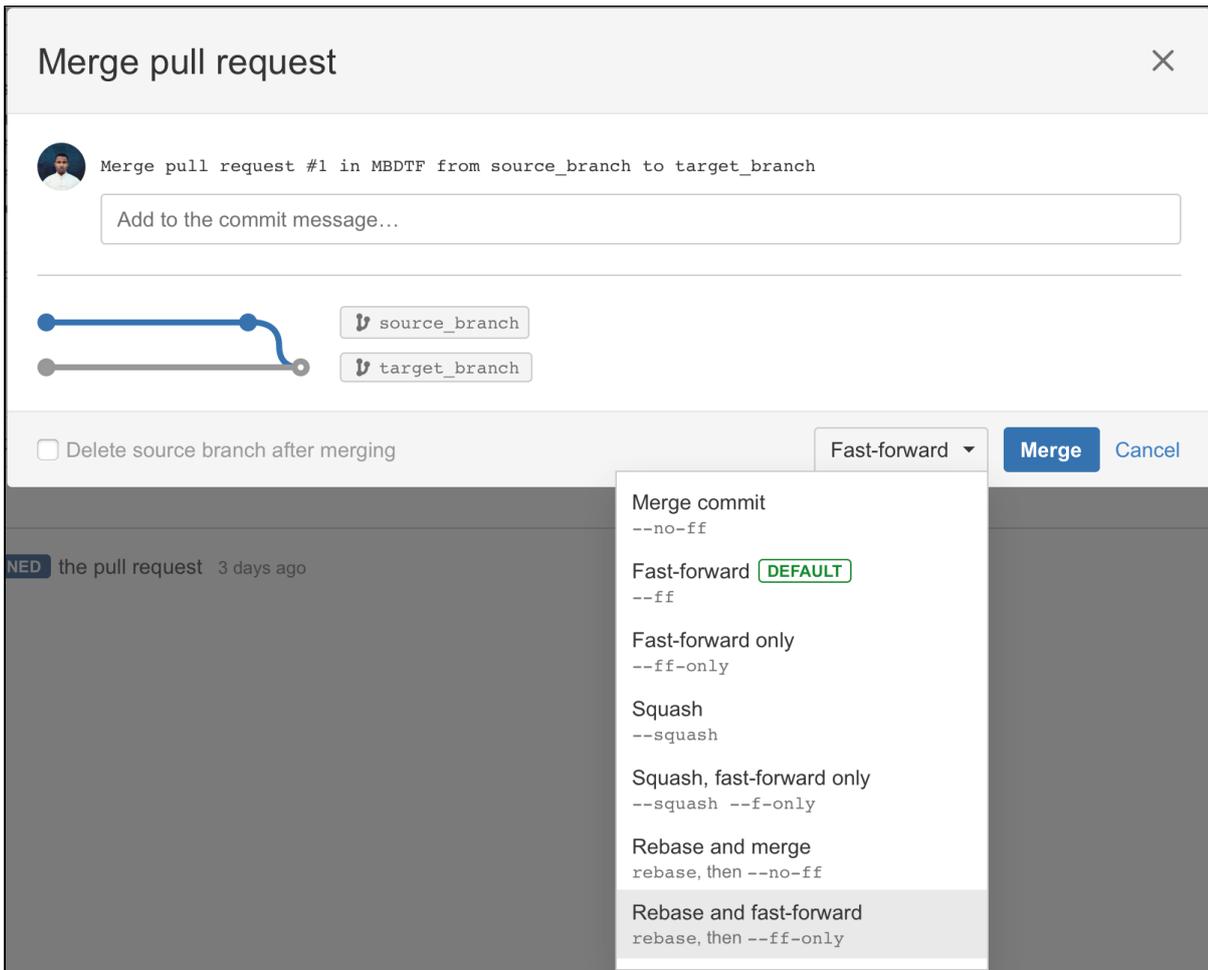
In the event there is a transactional failure, we recommend waiting a short time, refreshing the browser to load the latest pull request state, and trying again.

An **asynchronous** process updates pull requests when the source and target branches change, so multiple retries (or a longer wait) may be required before the system is in a consistent state to attempt the merge.

Need more context? Try our [merging versus rebasing tutorial](#).

Change the merge strategy for a single pull request

Which merge strategies can be used is determined by the merge strategies enabled by your project or repository admins.



To change the merge strategy for a specific pull request:

1. When [merging a pull request](#), click the default merge strategy in use, then select a new one.
2. Add a comment (if applicable), then click **Merge**.

Configure merge strategies for all repositories in a project

Enabling a merge strategy at the project level allows users to choose that merge strategy when merging pull requests for all repositories in a project, unless a repository is configured to use custom settings instead.

To enable (or disable) merge strategies for all repositories in a project (requires project admin permissions):

1. Go to **Project settings > Merge strategies**.
2. Click the toggle by a merge strategy to enable (or disable) it.
3. Click **Save**.

Users can now choose any of the merge strategies you enabled when they merge pull requests from the repositories in the project.

Configure merge strategies for an individual repository

Enabling (or disabling) a merge strategy at the repository level allows users to choose that merge strategy when merging pull requests created in that repository. Configuring merge strategies at the repository level requires you to override merge strategies configured at the project level. If you have not configured merge strategies for an individual repository it will inherit the merge strategies enabled at the project level.

To enable (or disable) merge strategies for a single repository (requires repository admin permissions):

1. Go to **Repository settings > Merge strategies**.
2. In the *Project settings inheritance* section, select **Use custom settings**.
3. Click the toggle by a merge strategy to enable (or disable) it.
4. (Optional) Once a merge strategy is enabled you can set it as the default merge strategy for that repository by hovering next to the toggle and clicking **Set as default**.



5. Click **Save**.

Once set, any changes made to merge strategies at the project level will be ignored for this repository because it was changed independent of the project configuration.

Code Insights

Code Insights is a feature added in [Bitbucket Server 5.15](#).

It surfaces information relevant to a pull request, so the author and reviewers are able to make better informed decisions. Information supplied could include:

- static analysis reports
- security scan results
- artifact links
- unit tests
- build status

It includes an API, and a UI so apps created by a third-party can supply data to be surfaced on the pull requests.

The information comprises two parts; **annotations and reports**.

Integrations

Integrations can be built to send data to pull requests. Integrations that have been built by third-parties can be found in the [Atlassian marketplace](#).

Building an integration: Resources

If you're looking to build your own integration, we have some tutorials available on Atlassian's developer documentation site.

If you are interested in **adding Code Insights as part of your CI pipeline**, here is a [very helpful resource](#).

If you're wanting a more comprehensive overview of the feature and how it works, try the [how to add code insights to pull requests](#) tutorial.

Quality reports and annotations

Quality reports

The generated reports show summary information sent by any integration to Bitbucket Data Center and Server for your branch. They can also be run against your branch without sending a report.

Reports can contain:

- a description
- a link back to the reporter's URL
- up to 6 fields for displaying data
- the annotations attached to the report (if there are any).

Annotations

Annotations show specific information on a given line in the pull request, and must be attached to a report.

Annotations are shown on the **unified diff view**, and on **reports**. They are not shown on side-by-side diff, commit diffs, or iterative diff.

You can see annotations on:

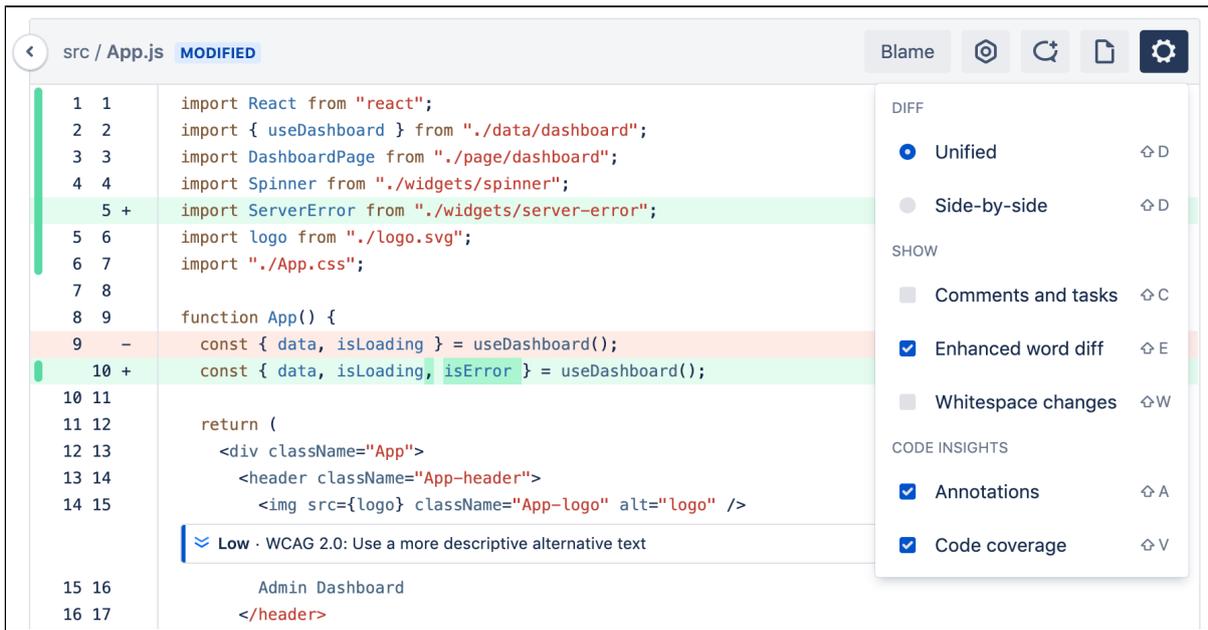
- changed lines, or
- lines added in the pull request.

This ensures that there is not too much noise from annotations that are not relevant to you.

Hiding annotations from the diff view

If you need to work without annotations enabled, you can hide them by using keyboard shortcut **Shift+ A** or:

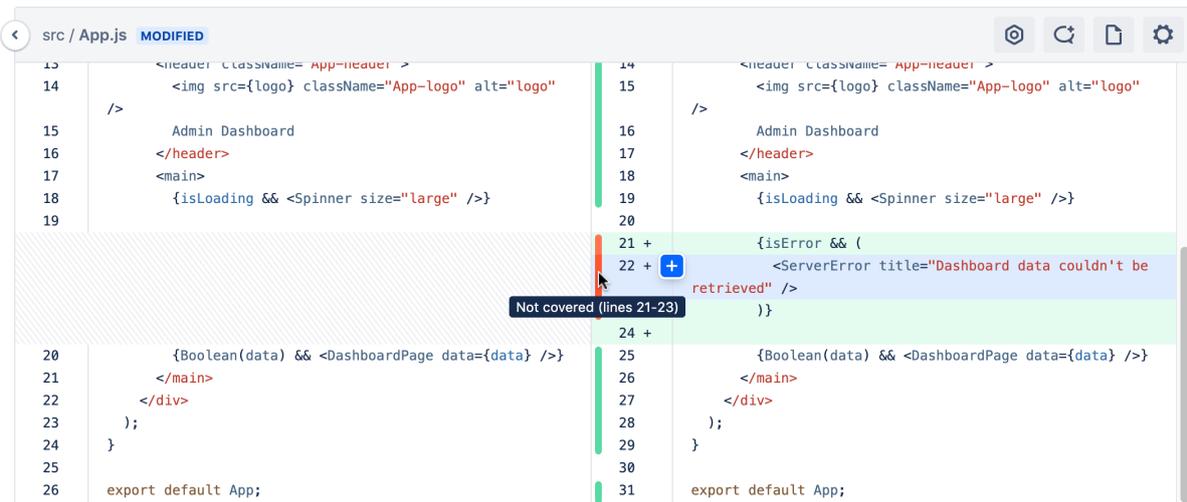
1. Click **Diff view settings** ⚙️.
2. Clear the **Annotations** check box option in the menu.



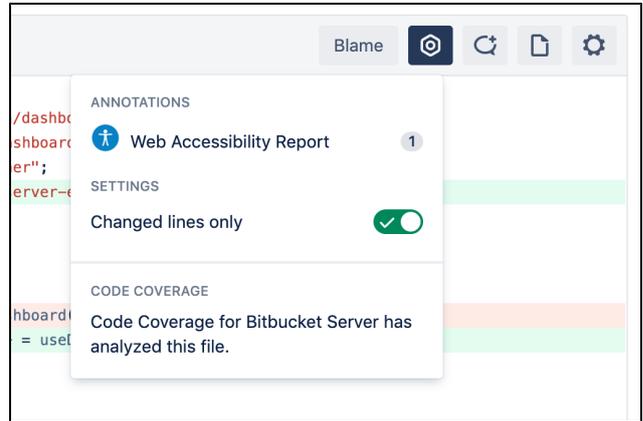
Code coverage

To ensure code that you are reviewing meets the quality requirements it takes to move on to the next stage of development, review code covered directly inside of a pull request without leaving Bitbucket by integrating your code coverage tool.

As part of your continuous integration (CI) workflow, code coverage finds aspects of the code which may not have been covered adequately by tests. The results are compiled and displayed in the diff view by color-coded blocks presenting which lines of code are covered **Fully**, **Partially**, or **Not covered**.



You can then also dig into a coverage report if its provided, to see those actual lines that may not have been covered and use that to identify critical parts of your application that still need to be tested.



Hiding code coverage from the diff view

If you need to work without code coverage displayed, you can hide it by using keyboard shortcut **Shift+ V**:

Click **Diff view settings** .

Clear the **Code coverage** check box option in the menu.

Head to Atlassian Marketplace for [Apps that support Code Insights](#). Here is the [link to API](#) if you want to integrate it with your own code coverage app.

Enhancements to your code review workflow

Since version 7.0 of Bitbucket Data Center and Server, we've been making big improvements to our pull request experience. This page collects them all in one spot for you.

Update	Released version
Reviewer groups for pull requests	7.13
Pull request description templates	7.13
Update pull requests to the latest version	7.10
New code review workflow	7.7
React to comments	7.7
Accessible colors for the diff view	7.7
View hidden comments	7.4
Locate pull requests by searching for text and selecting a reviewer	7.2
Pull requests get a makeover	7.0
Task improvements in a pull request	7.0
We've got your code covered	7.0

On this page:

- [Reviewer groups for pull requests \(released 7.13\)](#)
- [Pull request description templates \(released 7.13\)](#)
- [Update pull requests to the latest version \(released 7.10\)](#)
- [New code review workflow \(released 7.7\)](#)
- [React to comments \(released 7.7\)](#)
- [Accessible colors for the diff view \(released 7.7\)](#)
- [View hidden comments \(released 7.4\)](#)
- [Locate pull requests by searching for text and selecting a reviewer \(released 7.2\)](#)
- [Pull requests get a makeover \(released 7.0\)](#)
- [Task improvements in a pull request \(released 7.0\)](#)
- [We've got your code covered \(released 7.0\)](#)

Reviewer groups for pull requests (released 7.13)

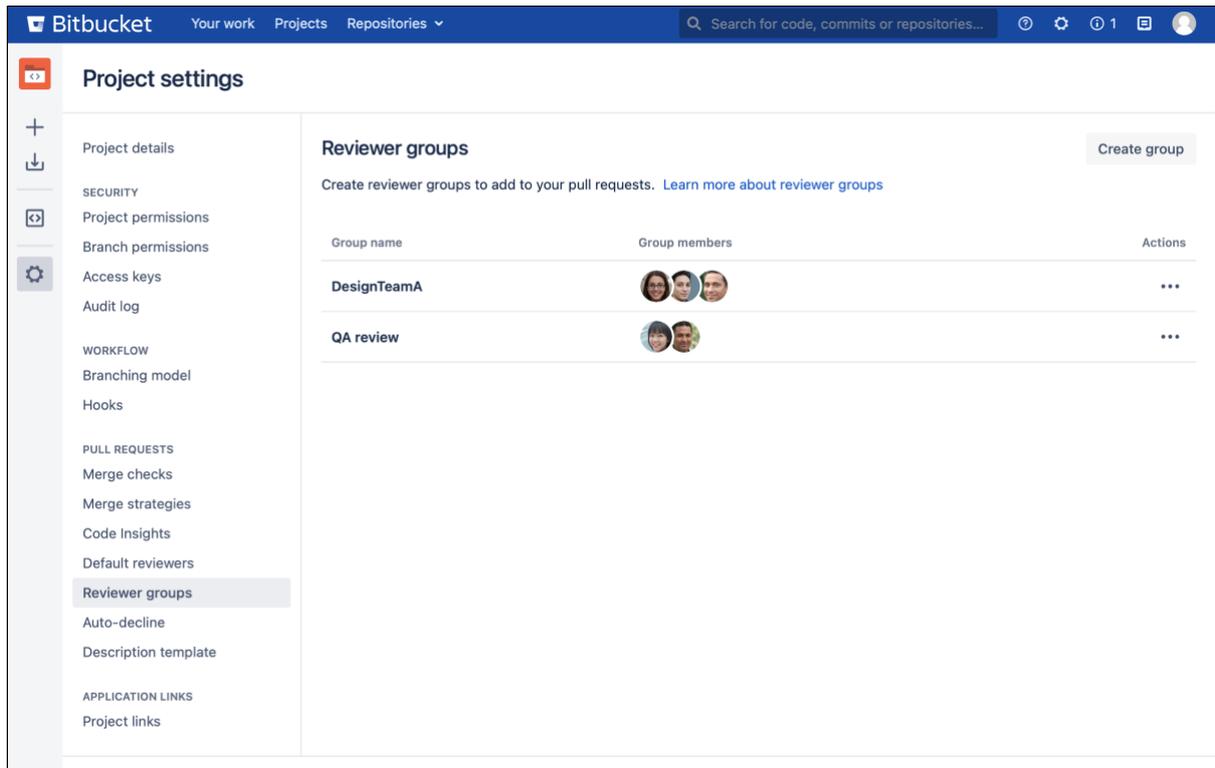
DATA CENTER

Spending more time than you should be, adding the same reviewers to your pull request, one user at a time? Reviewer groups are here to help you quickly add the right reviewers that need to be involved with your code review.

While creating a pull request, start by simply typing a group name into the Reviewers field, and then select a reviewer group to add.

Your browser does not support the HTML5 video element

Project and repository admins create and manage reviewer groups. This makes it easier for development teams to self-manage these groups. Select group members from existing Bitbucket users that have access to the project or repository.

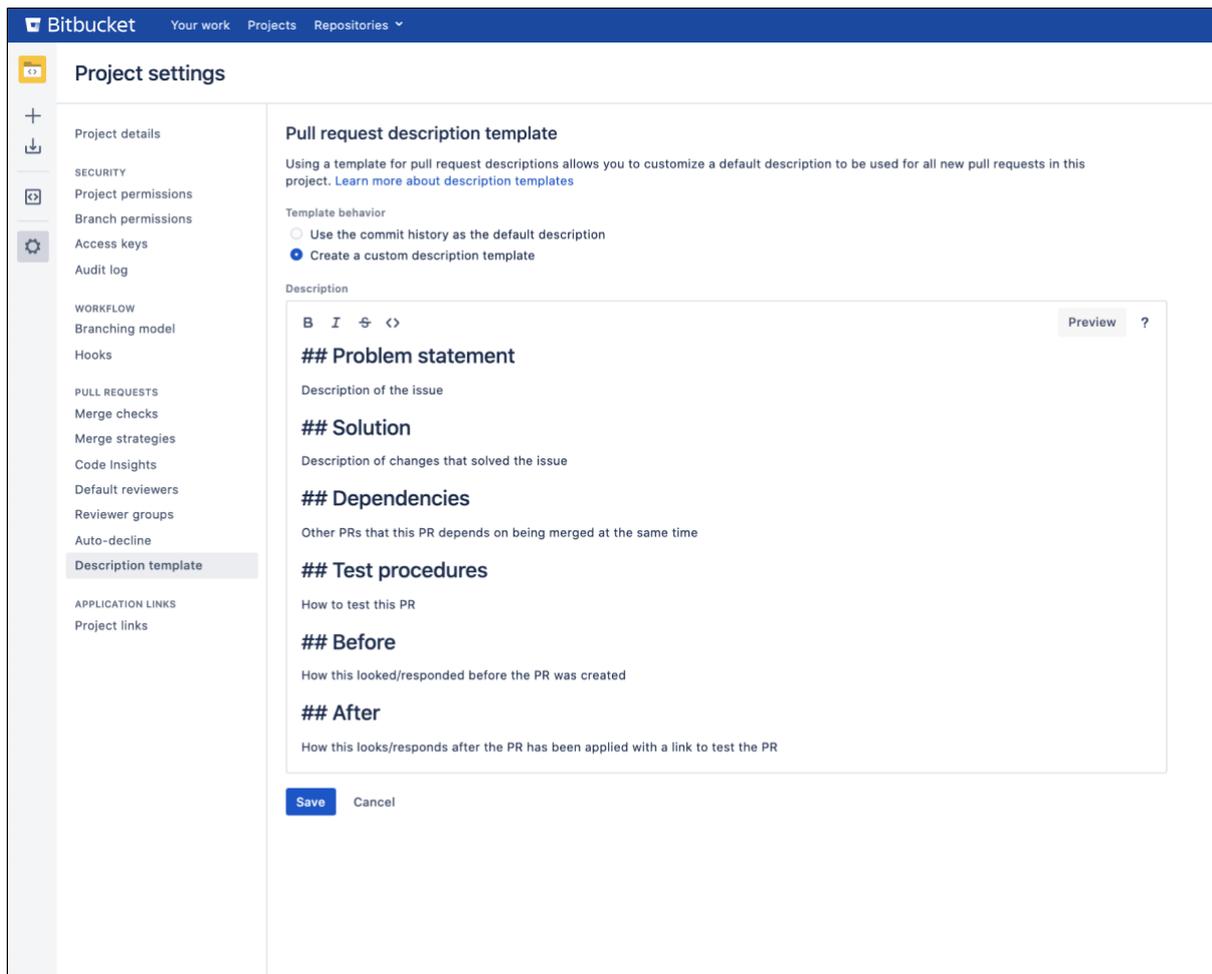


Get more information on how reviewer groups can help speed up your code review workflow.

Pull request description templates (released 7.13)

DATA CENTER

Having a template for your pull request descriptions will save time and help reviewers know what to expect while doing code reviews. Communicate review guidelines, or even remind authors about common tasks that should be done. When a new pull request is created, contents of the default description template are used and automatically applied.



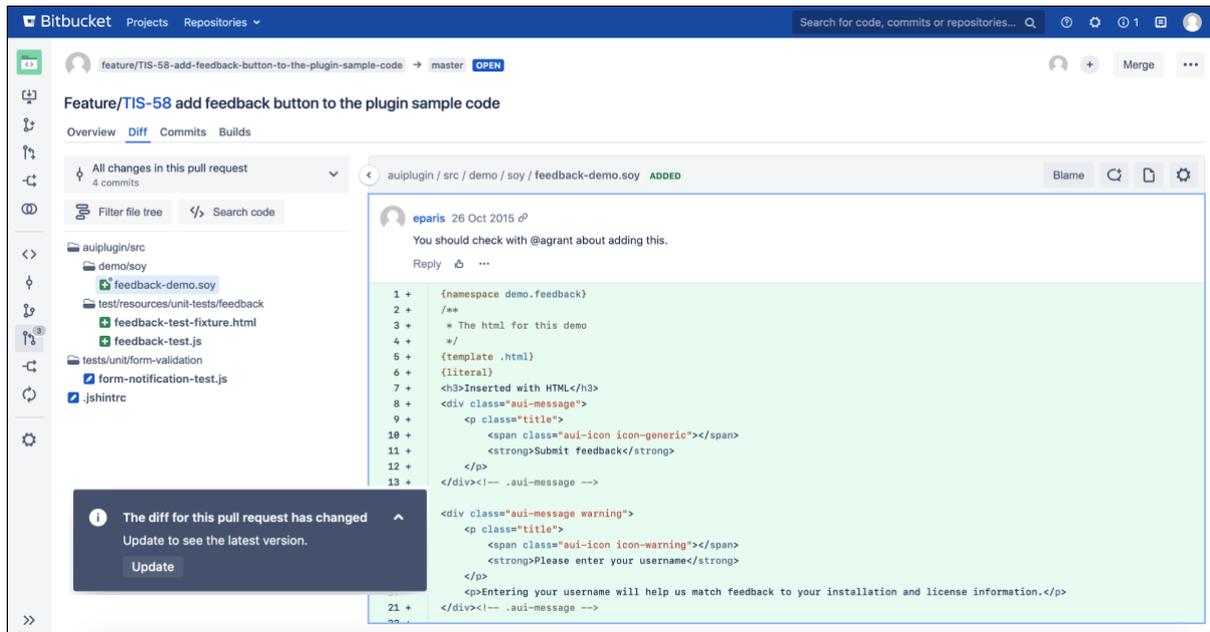
You can find the new description template for pull requests on the **Project settings** page.

Update pull requests to the latest version (released 7.10)

SERVER

DATA CENTER

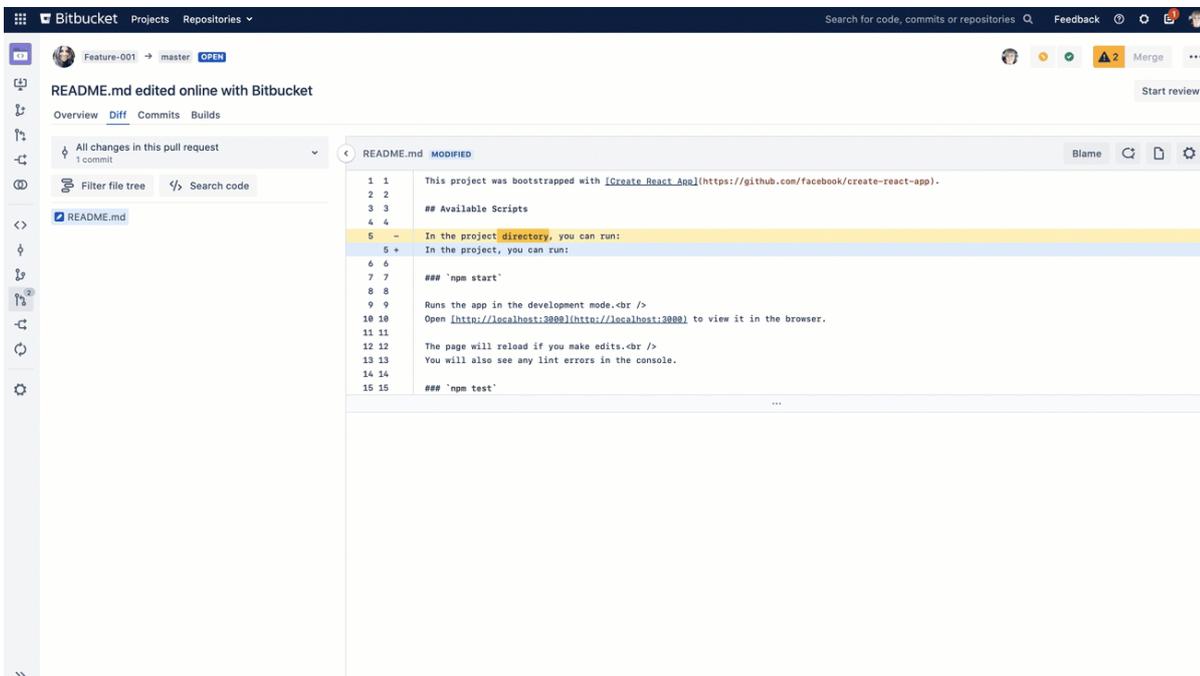
Now, when you're in the middle of reviewing code and a pull request is modified, Bitbucket will notify you so that you can update it to see the latest changes.



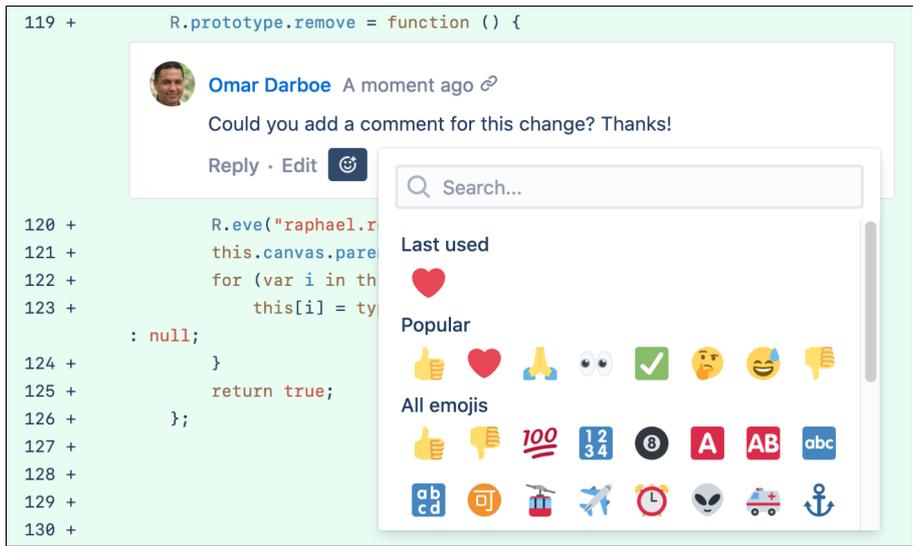
New code review workflow (released 7.7)



As a pull request reviewer, you can now draft multiple comments on files and code during a review process. Then when you are ready, send them all off at once, rather than just one at a time. This new code review workflow will save you time and unnecessary back and forth exchanges with the pull request author.



React to comments (released 7.7)



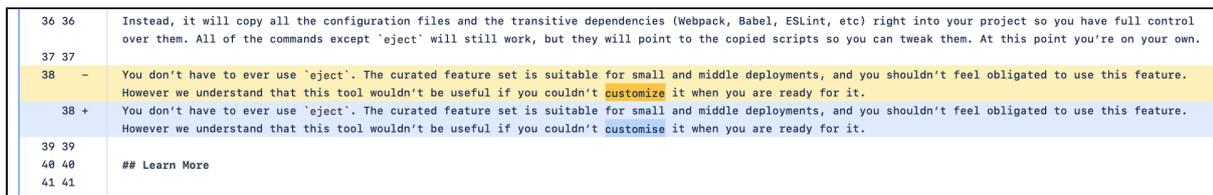
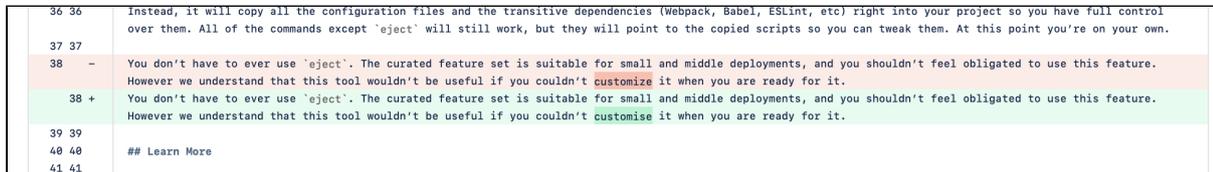
Responding to comments just got easier! With comment reactions you can now add emojis to any comment. So whether you want to give a quick , mark something as , or add a little , the right emoji is just a couple of clicks away.

Select the **Add reaction** button  to try it out, and see [Commenting on a pull request](#) for more information.

Accessible colors for the diff view (released 7.7)



We've been hard at work to make some important accessibility changes. This release includes a new setting for the diff view that lets you switch to an accessible color option that uses blue and yellow for added and removed lines respectively. Once enabled, the colors will change immediately in the diff view.

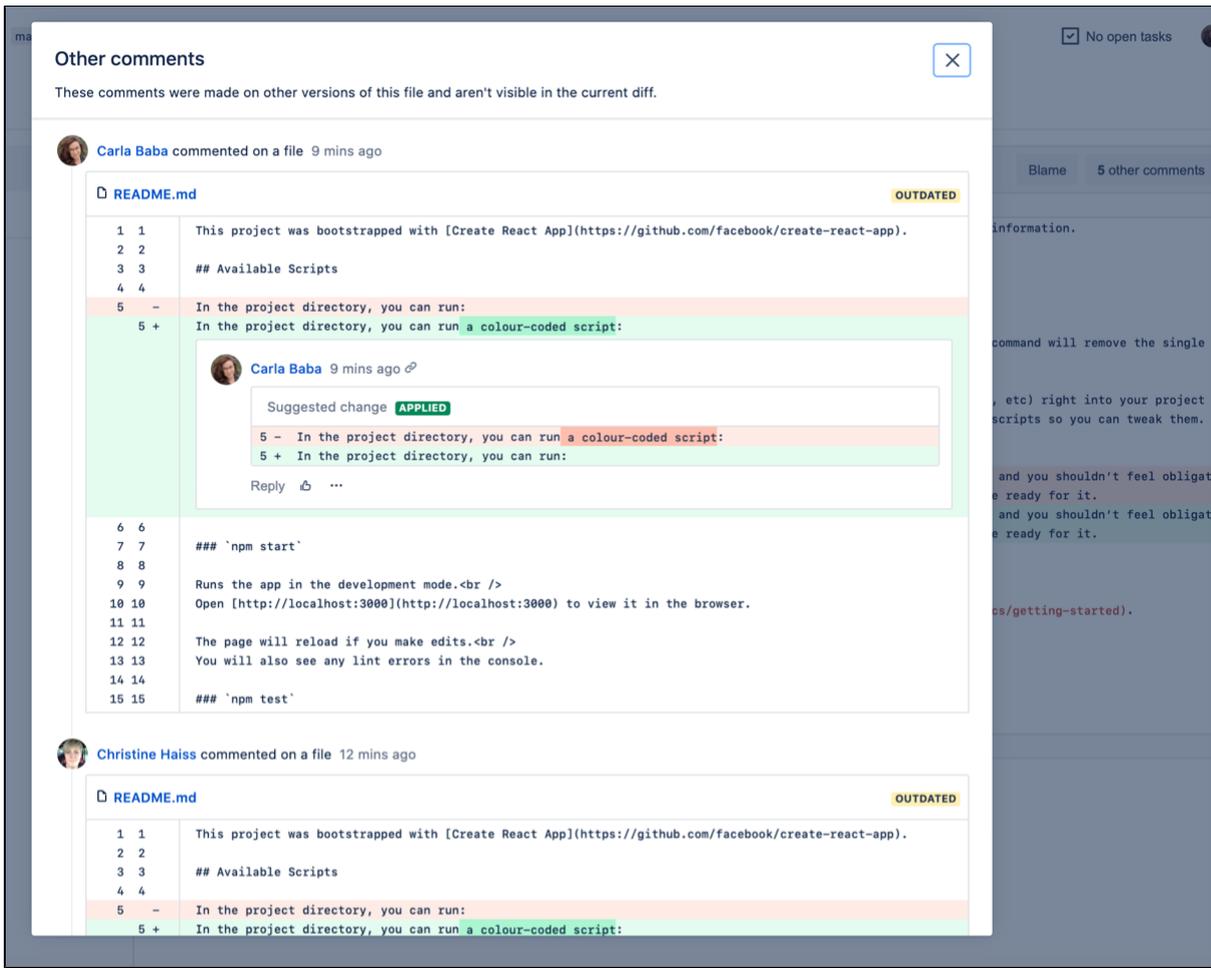


View hidden comments (released 7.4)



No more disappearing act for comments that are in older diffs or that have become outdated due to a pull request update. By selecting the other comment counter when it appears on your pull request, you'll have more context on why code has changed throughout a pull request by being able to:

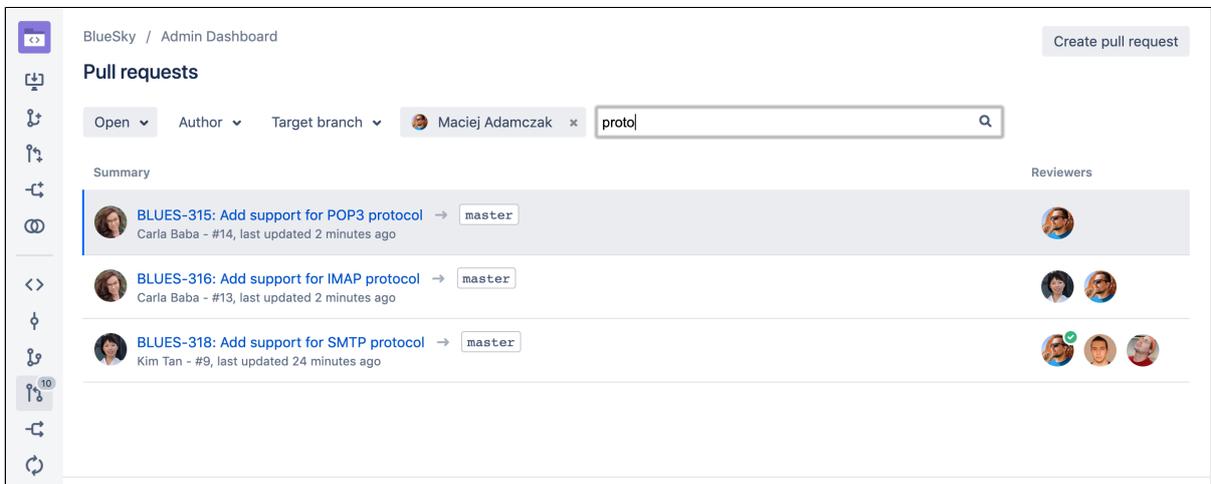
- See a file's activity stream showing comments that are outdated or appear on another diff.
- Distinguish which comments are actually outdated.
- Reply to, like, delete, or react to outdated comments the same way you can from the overview tab.



Locate pull requests by searching for text and selecting a reviewer (released 7.2)



It's not productive to have to sift through a large list of pull requests in order to locate a comment you've left or to see why a change was made, so we've made it easier to find a pull request by refining your search. Quickly filter on matches to pull request titles and their descriptions by entering some text into the new search field. We've also added the option to filter by any reviewer so that you can further narrow down the list of pull requests that you've reviewed.



For information on other pull request filtering options, see [Search for pull requests](#).

Pull requests get a makeover (released 7.0)

SERVER

DATA CENTER

The pull request page has been redesigned and packed full of improvements to make your code review experience smoother, more enjoyable, and less of a tedious task.

Faster navigation in pull requests

With twice-as-fast content loading while switching between diffs, improved file navigation performance, and smooth scrolling in a side-by-side diff, you just might find yourself having fun speeding through your code reviews.

Your browser does not support the HTML5 video element

A more intuitive design

We've improved the pull request workflow, giving you a more effective and efficient experience.

Better collaboration in the diff view

- Commenting
 - Comment anywhere in the diff, including on expanded lines of code that weren't changed as a part of the pull request.
 - Comment on files stored via [Git LFS](#).
- Syntax highlighting: Until now, syntax highlighting was only available on the side-by-side diff, but now its available in the unified diff too.

```
1 1 import React from "react";
2 2 import { useDashboard } from "../data/dashboard";
3 3 import DashboardPage from "../page/dashboard";
4 4 import Spinner from "../widgets/spinner";
5 + import ServerError from "../widgets/server-error";
6 6 import logo from "../logo.svg";
7 7 import "../App.css";
8 8
9 9 function App() {
10 - const { data, isLoading } = useDashboard();
10 + const { data, isLoading, isError } = useDashboard();
11 11
12 12 return (
13 13   <div className="App">
14 14     <header className="App-header">
15 15       <img src={logo} className="App-logo" alt="logo" />
16 16       Admin Dashboard
17 17     </header>
18 18     <main>
19 19       {isLoading && <Spinner size="large" />}
20 20
21 +     {isError && (
22 +       <ServerError title="Dashboard data couldn't be retrieved" />
23 +     )}
24 +
25 25     {Boolean(data) && <DashboardPage data={data} />}
26 26   </main>
27 27 </div>
28 28 );
29 29 }
30 30
31 31 export default App;
```

- Word wrap: No more horizontal scrolling to read lines of code in the diff.
- Expanding: Click Show more to expand all lines in the diff now, rather than just 10 at a time.

- Searching: You can now use your browser's native Ctrl+F action to search for code in the file you're viewing.
- Image thumbnails: Drag or attach an image into a comment and you'll see a thumbnail-sized preview of it rather than having the full-sized image rendered.

Do more with less clicking

- Copying: You can now copy code from just one side of the side-by-side diff view without also copying code from the other one. Also, select and copy branch names in one click.
- Pasting: When you paste code from a pull request into a comment, we'll automatically format it as a `code block` with the correct syntax highlighting.
- Editor: When you type into a comment or description editor in a PR, you'll see Markdown hints as you type, no need to preview until you're ready to see the final view.

Your browser does not support the HTML5 video element

File-tree improvements

- Filtering: Easily locate changed files in a pull request using the new **Filter file tree** button. Use wildcards to find what you need faster on pull requests that have a great deal of files to look through. You can find it just above the file-tree, along with the **Search code** button. To see an example, check out [Reviewing a pull request](#).

Your browser does not support the HTML5 video element

- Redesigned icons: Icons in the file tree view now have a more distinctive design with contrasting colors so that you can better recognize the relationship between them, and whether your file has been edited, added, removed, or modified.

Task improvements in a pull request (released 7.0)

SERVER

DATA CENTER

Create tasks in your pull requests without having to write a comment first. In fact, you can even convert comments into tasks. They also now support Markdown and rich content, like code snippets. Check the [Reviewing a pull request](#) page for more details.

```
15 16 Admin Dashboard
16 17 </header>
17 18 <main>
18 19   {isLoading && <Spinner size="large" />}
19 20
21 +   {isError && (
22 +     <ServerError title="Dashboard data couldn't be retrieved" />
23 +   )}
24 +
20 25   {Boolean(data) && <DashboardPage data={data} />}
21 26 </main>
22 27 </div>
23 28 );
24 29 }
25 30
26 31 export default App;
```

We've got your code covered (released 7.0)

[SERVER](#) [DATA CENTER](#)

Code Insights for Bitbucket offers a better way for your team to improve code quality by allowing continuous integration (CI) and other analysis tools to surface insights about code quality in pull requests. This is so issues that are related to code quality can be viewed and acted upon during a normal code review process. Now, you can access your code coverage results as a part of Code Insights.

Code coverage finds aspects of the code which may not have been covered adequately by tests. The results are compiled and displayed in the diff view using color-coded blocks which represent lines of code that are covered fully, partially, or not at all by tests. You can then also dig into a coverage report, if its provided, to see the actual lines that may not be covered and use that to identify critical parts of your application that still need to be tested.

```
13 <header className= App-header >
14 <img src={logo} className="App-logo" alt="logo"
15 />
16 Admin Dashboard
17 </header>
18 <main>
19   {isLoading && <Spinner size="large" />}
20
21 +   {isError && (
22 +     <ServerError title="Dashboard data couldn't be
23 +     retrieved" />
24 +   )}
25
26   {Boolean(data) && <DashboardPage data={data} />}
27 </main>
28 </div>
29 );
30 }
31 export default App;
```

Bitbucket search syntax

This page explains Bitbucket Data Center and Server's search syntax to help you find exactly what you're looking for.

Considerations

There are some restrictions on how searches are performed:

- Searches may include "." and "_", but all other punctuation characters are ignored. (e.g. !"#\$%&'()*+,-/:;<=>?@[\\]^`{|}~)
- To search for multiple words in exact order, put them in quotes (e.g. "find this phrase"). As with other queries, all punctuation except "." and "_" is ignored (e.g. "find-this-phrase" effectively searches for "find this phrase").
- Only the default branch is searchable (for most repositories the default branch will be master or main).
- Wildcard searches (e.g. qu?ck buil*) and regular expressions in queries are not supported.
- Case is not preserved, however search operators must be in ALL CAPS.
- Queries cannot have more than 9 expressions (e.g. combinations of terms and operators).
- The maximum length of a query cannot exceed 250 characters.
- Only files smaller than 512 KiB are searchable.
- Only code you have permission to view will appear in the search results.

A query in Bitbucket has to contain at least one search term, which can either be a single word or a phrase surrounded by quotes.

Operators

Search operators are words that can be added to searches to help narrow down the results. Operators must be in ALL CAPS. These are the search operators that can be used to search for files:

- AND
- OR
- NOT
- -
- ()

Multiple terms can be used, and they form a boolean query that implicitly uses the AND operator. So a query for "bitbucket server" is equivalent to "bitbucket AND server".

Term	Example query	Usage
AND	bitbucket AND server	Matches files that contain both "bitbucket" and "server".
OR	bitbucket OR server	Matches files that contain either "bitbucket" or "server" (or both).
NOT	bitbucket NOT jira	Matches files that contain "bitbucket" but don't contain "jira".
-	bitbucket -jira	Use before a term, matches files that contain "bitbucket" but don't contain "jira".
()	bitbucket AND (server OR cloud)	Matches files that contain "bitbucket" and either "server" or "cloud".

Modifiers

Modifiers can be used to further restrict search results. Use a modifier in the form "key:value". If there are multiple modifiers in a query they are implicitly combined using "AND" and apply to the whole search expression. These are the search modifiers that can be used to search for files.

Repository and Project modifiers

repo: <reponame> or repository: <reponame>

Search within a particular repository. Must be used with a project: modifier.

Term	Example query	Usage
repo:	<pre>jira repo: bitbucket project:atlassian or jira repository: bitbucket project:atlassian or jira repo: atlassian /bitbucket</pre>	Matches files within the "bitbucket" repository within the "atlassian" project that contain the term "jira".

project:<project key/name>

Search all repositories within a particular project for the search term.

Term	Example query	Usage
project:	<pre>jiraproject: atlassian</pre>	Matches files within the "atlassian" project that contain the term "jira".

Path modifier

path: <directory or file name>

Restrict search to only consider files with the search term in their path.

Term	Example query	Usage
path:	<pre>react path:frontend</pre>	Matches files within directories namedfrontend containing the term react.
	<pre>react path:frontend/* /package.lock</pre>	Matchespackage.lock files within directories namedfrontend containing the term react.

Language and file extension modifiers

lang: or language: <language>

Code search can be restricted to only consider a particular language or a particular file extension. For some languages adding `lang` criteria is equivalent to specifying the file extension. For example, `lang: java` is equivalent to `ext: java`. For other languages multiple file extensions are mapped to a single language. For example, `.hs`, `.lhs` and `.hs-boot` are file extensions used for the Haskell programming language, and will be matched when specifying `lang: haskell`.

Term	Example query	Usage
<code>lang:</code>	<code>jiralang: java</code> or <code>jiralanguage: java</code>	Matches files that contain the term "jira" within Java files, files with <code>.java</code> , <code>.class</code> , or <code>.jar</code> extension.

`ext:` or `extension:` <file extension>

Term	Example query	Usage
<code>ext:</code>	<code>jiraext: lhs</code> or <code>jiraextension: lhs</code>	Matches files that contain the term "jira" within Haskell files with the <code>.lhs</code> extension.

Fork modifier

`fork:`<true or false>

Exclude results from repositories based on whether or not they are forks.

Term	Example query	Usage
<code>fork:</code>	<code>fork: true</code>	Includes only results that are from repositories that are forks.
	<code>fork: false</code>	Includes only results that are from repositories that are not forks.

Manage webhooks

Webhooks provide a way to configure Bitbucket Data Center and Server to make requests to your server or another external service, whenever certain events occur. A webhook consists of:

- One or more events the default event is a repository push, but you can select multiple events to trigger the webhook.
- A URL the endpoint where you want Bitbucket to send the event payloads when a matching event happens.

There are two stages to the webhook: [Creating webhooks](#) and [Triggering webhooks](#). Once you've created a webhook for an event, every time that event occurs, Bitbucket sends a payload request that describes the event to the specified URL.

If you're having problems with a webhook, see [Troubleshooting webhooks](#).

When to use webhooks

Use webhooks to integrate applications with Bitbucket. For example:

- Every time a user pushes commits in a repository, you may want to notify your CI server to start a build.
- Every time a user pushes commits or creates a pull request, you may want to display a notification in your application.
- Every time a user pushes commits to a repository, and a mirror synchronizes those changes, you may want to notify your continuous integration system to start a build.

Advantages of webhooks

Without webhooks, you need to poll the API if you want to detect when events occur in Bitbucket. However, polling the API is inconvenient, inefficient, and error-prone.

Webhooks mean the API doesn't have to check for the same activity every minute.

Webhook secrets

Webhook secrets are what Bitbucket uses to authenticate the payload. The secret ensures that between Bitbucket and your endpoint, you're able to verify that the contents were not tampered with.

Combined with HTTPS, it helps ensure the message transmitted is the one that Bitbucket intended to send.

Securing your webhook

When you define a secret for a webhook, each request is signed via a Hash-based Message Authentication Code (HMAC).

The default for this algorithm is HMACSha256. The header X-Hub-Signature is defined and contains the HMAC.

To authenticate the validity of the message payload, the receiver can perform the HMAC algorithm on the received body with the secret as the key to the HMAC algorithm.

If the results do not match, it may indicate there was a problem with transmission that has caused the message payload to change.

Creating webhooks

On this page:

- [When to use webhooks](#)
- [Advantages of webhooks](#)
- [Webhook secrets](#)
 - [Securing your webhook](#)
- [Creating webhooks](#)
- [Triggering webhooks](#)
- [Troubleshooting webhooks](#)
 - [Circuit breaking](#)

You can create a webhook through Bitbucket, or with the API. Use the following steps to create a webhook on a repository in Bitbucket. Don't forget the administrators of a repository are the only users who can create a webhook on that repository.

1. Open the repository where you want to add the webhook.
2. From **Repository Settings**, click the **Webhooks** link.
3. Click the **Create webhook** button to create a webhook for the repository.
4. **Webhook events**: Select the event/s to trigger the webhook.
5. Enter a **Title** with a short description, and the **URL** of the application or server.
6. (optional) Enter a **Secret** string of up to 255 characters.
7. (optional) Use the **Test connection** button if required.
8. By default, the event for the webhook is a repository push, as demonstrated by the **Repository push** field.
9. (Optional) If you don't want the webhook to be active after you create it, remove the checkmark from **Active**.
10. Click **Create**.

To create a webhook using the API, you need to know the format of the HTTP request that Bitbucket expects and the format of the HTTP response that Bitbucket returns to your server.

Triggering webhooks

When an event associated with a webhook occurs, Bitbucket sends a request to the webhook URL containing the event payload.

You can create webhooks for the following events:

Repository events

- Push
- Modified
- Forked
- Comment added to commit
- Comment edited on commit
- Comment deleted on commit
- Mirror synchronized

Pull request events

- Opened
- Approved
- Unapproved
- Needs work
- Merged
- Declined
- Deleted
- Comment added
- Comment edited
- Comment deleted

Troubleshooting webhooks

When you perform an action to trigger a webhook and it doesn't work, you can use the **Eventlog** page to figure out what went wrong.

Under **Event details** in repository settings, you can click view details from the **Actions** column to access the event log for troubleshooting purposes.

[Back to webhooks](#)

test-me ACTIVE
<https://requestb.in/1mh28q11> ✎ Edit

Event log

The last success and failure for each even are recorded in the table below (up to the last 30 days).

Event type	Last success	Last failure	Successful calls
Pull request approved	36 mins ago	Never failed	2/2 (100%)
Pull request comment added	Never succeeded	Never failed	-
Pull request comment deleted	Never succeeded	Never failed	-
Pull request comment edited	Never succeeded	Never failed	-
Pull request declined	Never succeeded	Never failed	-
Pull request deleted	26 mins ago	Never failed	1/1 (100%)
Pull request marked needs work	36 mins ago	Never failed	3/3 (100%)
Pull request merged	35 mins ago	Never failed	1/1 (100%)
Pull request opened	34 mins ago	Never failed	2/2 (100%)

Click through the latest request results (such as Webhook event details) to troubleshoot the issue.

Webhook event details ✕

Request
Response

Response details

HTTP status: 200

Duration 139ms

Headers

```
Strict-Transport-Security: max-age=15552000
Date: Mon, 06 Nov 2017 00:33:00 GMT
Content-Type: text/html; charset=UTF-8
Via: 1.1 localhost (Apache-HttpClient/4.4.1 (cache))
Connection: keep-alive
X-Content-Type-Options: nosniff
Transfer-Encoding: chunked
```

Body

```
ok
```

Circuit breaking

To help protect your instance of Bitbucket, circuit breaking has been implemented in the Bitbucketwebhooks system. This means badly behaving webhooks are skipped for a period of time if they are consistently failing.

By default, when a webhook fails five times, it is considered unhealthy and is skipped.

Initially, it will only be skipped for a short period (10 seconds) but as it continues to fail it will gradually skip for longer periods, up to a max of 10 hours.

A webhook may also be skipped if there are too many webhooks in flight. If there are 250 webhooks being invoked, further requests will be skipped until the number in flight drops below 250.

These limits are entirely configurable if your instance has different requirements. For more information, see [Configuration properties](#).

If a webhook is being skipped, you can see so via the [JMX metrics](#) output by Bitbucket, or via the logs.

Event payload

When you have a webhook with an event, Bitbucket Data Center and Server sends the event request to the server URL for the webhook whenever that event occurs. This page describes the structure of these requests.

For Bitbucket to send event payload requests for a webhook with HTTPS endpoints, make sure your URL has a valid SSL certificate that a public certificate authority has signed.

The following payloads contain some of the five common entity types `User`, `Repository`, `Comment`, and `Pull Request` which have consistent representations in all the payloads where they appear. For example, the `actor` property in the `repo:refs_changed` payload is a representation of the event's user.

HTTP headers

All event payload requests have the following HTTP headers:

HTTP header	Description
X-Request-Id	A unique UUID for each webhook request
X-Event-Key	The event that kicked off this webhook. For example, a repository push will have <code>repo:refs_changed</code> .
X-Hub-Signature	See Webhook secrets

Repository events

You can create webhooks for the following events that occur in a repository.

Push

A user pushes one or more commits to a repository. This payload, with an event key of `repo:refs_changed`, provides the following fields:

Parameter	Description
<code>actor</code>	The user who pushed the commits.
<code>repository</code>	The repository with the commits.
<code>changes</code>	The details of the push.

```

{
  "eventKey": "repo:refs_changed",
  "date": "2017-09-19T09:45:32+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "repository": {
    "slug": "repository",
    "id": 84,
    "name": "repository",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    },
    "public": false
  },
  "changes": [
    {
      "ref": {
        "id": "refs/heads/master",
        "displayId": "master",
        "type": "BRANCH"
      },
      "refId": "refs/heads/master",
      "fromHash": "ecddabb624f6f5ba43816f5926e580a5f680a932",
      "toHash": "178864a7d521b6f5e720b386b2c2b0ef8563e0dc",
      "type": "UPDATE"
    }
  ]
}

```

Modified

A user updates the **Name** of a repository. This payload, with an event key of `repo:modified`, provides the following fields:

Parameter	Description
old	The details of the old version of the repository.
new	The defaults of the current version of the repository.
actor	The user who made the update.

```

{
  "eventKey": "repo:modified",
  "date": "2017-09-19T09:51:20+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "old": {
    "slug": "repository",
    "id": 84,
    "name": "repository",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    },
    "public": false
  },
  "new": {
    "slug": "repository2",
    "id": 84,
    "name": "repository2",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    },
    "public": false
  }
}

```

Fork

A user forks a repository. This payload, with an event key of `repo:fork`, provides the following fields:

Parameter	Description
<code>actor</code>	The user who forks the repository. This user is also the owner of the fork.
<code>repository</code>	The new repository.
<code>repository.origin</code>	The original repository that was forked.

```

{
  "eventKey": "repo:forked",
  "date": "2017-09-19T09:48:26+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "repository": {
    "slug": "repository2",
    "id": 86,
    "name": "repository2",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "origin": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",
      "forkable": true,
      "project": {
        "key": "PROJ",
        "id": 84,
        "name": "project",
        "public": false,
        "type": "NORMAL"
      },
      "public": false
    },
    "project": {
      "key": "~ADMIN",
      "id": 22,
      "name": "Administrator",
      "type": "PERSONAL",
      "owner": {
        "name": "admin",
        "emailAddress": "admin@example.com",
        "id": 1,
        "displayName": "Administrator",
        "active": true,
        "slug": "admin",
        "type": "NORMAL"
      }
    },
    "public": false
  }
}

```

Commit comment created

A user comments on a commit in a repository. This payload, with an event key of `repo:comment:added`, provides the following fields:

Parameter	Description
actor	The user who comments on the commit.
comment	The comment created.
repository	The repository with the commit.

commit	The hash.
--------	-----------

```
{
  "eventKey": "repo:comment:added",
  "date": "2017-09-19T09:53:06+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "comment": {
    "properties": {
      "repositoryId": 84
    },
    "id": 42,
    "version": 0,
    "text": "This is a great line of code!",
    "author": {
      "name": "admin",
      "emailAddress": "admin@example.com",
      "id": 1,
      "displayName": "Administrator",
      "active": true,
      "slug": "admin",
      "type": "NORMAL"
    },
    "createdDate": 1505778786337,
    "updatedDate": 1505778786337,
    "comments": [

    ],
    "tasks": [

    ],
    "permittedOperations": {
      "editable": true,
      "deletable": true
    }
  },
  "repository": {
    "slug": "repository",
    "id": 84,
    "name": "repository",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    },
    "public": false
  },
  "commit": "178864a7d521b6f5e720b386b2c2b0ef8563e0dc"
}
```

Commit edit on comment

A user edits a comment on a commit in a repository. This payload, with an event key of `repo:comment:edited`, provides the following fields:

Parameter	Description
-----------	-------------

<code>actor</code>	The user who edits the commit.
<code>comment</code>	The comment edited.
<code>previousComment</code>	The text of the comment prior to editing.
<code>repository</code>	The repository with the commit.
<code>commit</code>	The hash of the commit.

```

{
  "eventKey": "repo:comment:edited",
  "date": "2017-09-19T09:55:03+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "comment": {
    "properties": {
      "repositoryId": 84
    },
    "id": 42,
    "version": 1,
    "text": "This is a okay line of code!",
    "author": {
      "name": "admin",
      "emailAddress": "admin@example.com",
      "id": 1,
      "displayName": "Administrator",
      "active": true,
      "slug": "admin",
      "type": "NORMAL"
    },
    "createdDate": 1505778786337,
    "updatedDate": 1505778903525,
    "comments": [

    ],
    "tasks": [

    ],
    "permittedOperations": {
      "editable": true,
      "deletable": true
    }
  },
  "repository": {
    "slug": "repository",
    "id": 84,
    "name": "repository",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    },
    "public": false
  },
  "commit": "178864a7d521b6f5e720b386b2c2b0ef8563e0dc",
  "previousComment": "This is a great line of code!"
}

```

Comment deleted on commit

A user deletes a comment on a commit in a repository. This payload, with an event key of `repo:comment:deleted`, provides the following fields:

Parameter	Description
actor	The user who deletes the commit.

comment	The comment deleted.
repository	The repository with the commit.
commit	The hash of the commit.

```
{
  "eventKey": "repo:comment:deleted",
  "date": "2017-09-19T09:56:29+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "comment": {
    "id": 42,
    "version": 1,
    "text": "This is a okay line of code!",
    "author": {
      "name": "admin",
      "emailAddress": "admin@example.com",
      "id": 1,
      "displayName": "Administrator",
      "active": true,
      "slug": "admin",
      "type": "NORMAL"
    }
  },
  "createdDate": 1505778786337,
  "updatedDate": 1505778903525,
  "comments": [

  ],
  "tasks": [

  ]
},
"repository": {
  "slug": "repository",
  "id": 84,
  "name": "repository",
  "scmId": "git",
  "state": "AVAILABLE",
  "statusMessage": "Available",
  "forkable": true,
  "project": {
    "key": "PROJ",
    "id": 84,
    "name": "project",
    "public": false,
    "type": "NORMAL"
  },
  "public": false
},
"commit": "178864a7d521b6f5e720b386b2c2b0ef8563e0dc"
}
```

Mirror synchronized

A mirror has finished synchronizing this repository. This payload, with an event key of `mirror:repo_synchronized`, provides the following fields:

Parameter	Description
-----------	-------------

<code>mirrorServer</code>	The mirror which synchronized the changes. This JSON object contains both the <code>name</code> and the <code>id</code> of the <code>mirrorServer</code> which synchronized the changes.
<code>syncType</code>	The sync type the mirror used to synchronize the changes which are announced by this webhook. This value can be <code>snapshot</code> or <code>incremental</code> for mirrors 6.7 and higher. It defaults to <code>smartMirror</code> for mirrors before version 6.7.
<code>repository</code>	The repository.
<code>repository.links</code>	This JSON object contains the HTTP and SSH clone URLs of the primary server as well as the mirror that synchronized these changes. It also contains a link to view this repository in Bitbucket.
<code>changes</code>	The ref changes for this push.
<code>refLimitExceeded</code>	If this value is <code>true</code> , the list of <code>changes</code> will be empty because it exceeded the limit of refs that can be included.

```

{
  "eventKey": "mirror:repo_synchronized",
  "date": "2019-07-11T16:18:20+1000",
  "mirrorServer": {
    "id": "B88H-IR7J-5PV0-VCNS",
    "name": "Mirror Name"
  },
  "syncType": "INCREMENTAL",
  "refLimitExceeded": false,
  "repository": {
    "slug": "testrepo",
    "id": 1,
    "name": "testrepo",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "TP",
      "id": 1,
      "name": "testp",
      "public": false,
      "type": "NORMAL"
    },
    "public": false,
    "links": {
      "clone": [
        {
          "href": "ssh://git@example.com:7997/project/repository.git",
          "name": "ssh"
        },
        {
          "href": "https://example.com/scm/project/repository.git",
          "name": "http"
        }
      ],
      "self": [
        {
          "href": "https://example.com/projects/project/repos/repo/browse"
        }
      ]
    }
  },
  "changes": [
    {
      "ref": {
        "id": "refs/heads/master",
        "displayId": "master",
        "type": "BRANCH"
      },
      "refId": "refs/heads/master",
      "fromHash": "b5616b9",
      "toHash": "d055eca",
      "type": "UPDATE"
    }
  ]
}

```

Pull request events

You can create webhooks for the following events that occur on a pull request.

Opened

Parameter	Description
-----------	-------------

actor	The user who created the pull request.
pullrequest	Details of the pull request created.

```
{
  "eventKey": "pr:opened",
  "date": "2017-09-19T09:58:11+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "pullRequest": {
    "id": 1,
    "version": 0,
    "title": "a new file added",
    "state": "OPEN",
    "open": true,
    "closed": false,
    "createdDate": 1505779091796,
    "updatedDate": 1505779091796,
    "fromRef": {
      "id": "refs/heads/a-branch",
      "displayId": "a-branch",
      "latestCommit": "ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
      "repository": {
        "slug": "repository",
        "id": 84,
        "name": "repository",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "PROJ",
          "id": 84,
          "name": "project",
          "public": false,
          "type": "NORMAL"
        },
        "public": false
      }
    },
    "toRef": {
      "id": "refs/heads/master",
      "displayId": "master",
      "latestCommit": "178864a7d521b6f5e720b386b2c2b0ef8563e0dc",
      "repository": {
        "slug": "repository",
        "id": 84,
        "name": "repository",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "PROJ",
          "id": 84,
          "name": "project",
          "public": false,
          "type": "NORMAL"
        },
        "public": false
      }
    },
    "locked": false,
    "author": {
      "user": {
        "name": "admin",

```

```

    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "role": "AUTHOR",
  "approved": false,
  "status": "UNAPPROVED"
},
"reviewers": [
],
"participants": [
],
"links": {
  "self": [
    null
  ]
}
}
}

```

Source branch updated

Parameter	Description
actor	The user who created the pull request.
pullrequest	Details of the pull request created.
previousFromHash	Previous from-ref hash

```

{
  "eventKey": "pr:from_ref_updated",
  "date": "2020-02-20T14:49:41+1100",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL",
    "links": {
      "self": [
        {
          "href": "http://localhost:7990/bitbucket/users/admin"
        }
      ]
    }
  },
  "pullRequest": {
    "id": 2,
    "version": 16,
    "title": "Webhook",
    "state": "OPEN",
    "open": true,
    "closed": false,
    "createdDate": 1582065825700,
    "updatedDate": 1582170581372,
    "fromRef": {
      "id": "refs/heads/pr-webhook",
      "displayId": "pr-webhook",
      "latestCommit": "aab847db240ccae221f8036605b00f777eba95d2",
      "repository": {

```

```

    "slug": "dvcs",
    "id": 33,
    "name": "dvcs",
    "hierarchyId": "09992c6ad9e001f01120",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "GIT",
      "id": 62,
      "name": "Bitbucket",
      "public": false,
      "type": "NORMAL",
      "links": {
        "self": [
          {
            "href": "http://localhost:7990/bitbucket/projects/GIT"
          }
        ]
      }
    },
    "public": false,
    "links": {
      "clone": [
        {
          "href": "ssh://git@localhost:7999/git/dvcs.git",
          "name": "ssh"
        },
        {
          "href": "http://localhost:7990/bitbucket/scm/git/dvcs.git",
          "name": "http"
        }
      ],
      "self": [
        {
          "href": "http://localhost:7990/bitbucket/projects/GIT/repos/dvcs/browse"
        }
      ]
    }
  },
  "toRef": {
    "id": "refs/heads/master",
    "displayId": "master",
    "latestCommit": "86448735f9dee9e1fb3d3e5cd9fbc8eb9d8400f4",
    "repository": {
      "slug": "dvcs",
      "id": 33,
      "name": "dvcs",
      "hierarchyId": "09992c6ad9e001f01120",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",
      "forkable": true,
      "project": {
        "key": "GIT",
        "id": 62,
        "name": "Bitbucket",
        "public": false,
        "type": "NORMAL",
        "links": {
          "self": [
            {
              "href": "http://localhost:7990/bitbucket/projects/GIT"
            }
          ]
        }
      }
    }
  },
  "public": false,
  "links": {
    "clone": [
      {
        "href": "ssh://git@localhost:7999/git/dvcs.git",
        "name": "ssh"
      },
    ],
  },

```

```

        {
          "href": "http://localhost:7990/bitbucket/scm/git/dvcs.git",
          "name": "http"
        }
      ],
      "self": [
        {
          "href": "http://localhost:7990/bitbucket/projects/GIT/repos/dvcs/browse"
        }
      ]
    }
  },
  "locked": false,
  "author": {
    "user": {
      "name": "admin",
      "emailAddress": "admin@example.com",
      "id": 1,
      "displayName": "Administrator",
      "active": true,
      "slug": "admin",
      "type": "NORMAL",
      "links": {
        "self": [
          {
            "href": "http://localhost:7990/bitbucket/users/admin"
          }
        ]
      }
    }
  },
  "role": "AUTHOR",
  "approved": false,
  "status": "UNAPPROVED"
},
"reviewers": [],
"participants": [],
"links": {
  "self": [
    {
      "href": "http://localhost:7990/bitbucket/projects/GIT/repos/dvcs/pull-requests/2"
    }
  ]
}
},
"previousFromHash": "99f3ea32043ba3ecaa28de6046b420de70257d80"
}

```

Modified

Parameter	Description
actor	The user who created the pull request.
pullrequest	Details of the pull request created.
previousTitle	Previous title of the pull request, may not have changed
previousDescription	Previous description of pull request, may not have changed
previousTarget	Previous target of the pull request, may not have changed

```

{
  "eventKey": "pr:modified",
  "date": "2018-04-24T10:15:30+1000",
  "actor": {
    "name": "Administrator",
    "emailAddress": "example@atlassian.com",
    "id": 110653,

```

```

    "displayName": "Administrator",
    "active": true,
    "slug": "pathompson",
    "type": "NORMAL"
  },
  "pullRequest": {
    "id": 1,
    "version": 1,
    "title": "A new title",
    "description": "A new description",
    "state": "OPEN",
    "open": true,
    "closed": false,
    "createdDate": 1524528879329,
    "updatedDate": 1524528930110,
    "fromRef": {
      "id": "refs/heads/new-branch",
      "displayId": "new-branch",
      "latestCommit": "5a705e60111a4213da46839d9cbf4fc43639b771",
      "repository": {
        "slug": "example",
        "id": 12087,
        "name": "example",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "~ADMIN",
          "id": 8504,
          "name": "Administrator",
          "type": "PERSONAL",
          "owner": {
            "name": "Administrator",
            "emailAddress": "example@atlassian.com",
            "id": 110653,
            "displayName": "Administrator",
            "active": true,
            "slug": "admin",
            "type": "NORMAL"
          }
        }
      }
    },
    "public": false
  }
},
"toRef": {
  "id": "refs/heads/master",
  "displayId": "master",
  "latestCommit": "860c4eb4ed0f969b47144234ba13c31c498cca69",
  "repository": {
    "slug": "example",
    "id": 12087,
    "name": "example",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "~ADMIN",
      "id": 8504,
      "name": "Administrator",
      "type": "PERSONAL",
      "owner": {
        "name": "Administrator",
        "emailAddress": "example@atlassian.com",
        "id": 110653,
        "displayName": "Administrator",
        "active": true,
        "slug": "admin",
        "type": "NORMAL"
      }
    }
  }
},
"public": false
}
},
"locked": false,

```

```

"author": {
  "user": {
    "name": "Administrator",
    "emailAddress": "example@atlassian.com",
    "id": 110653,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "role": "AUTHOR",
  "approved": false,
  "status": "UNAPPROVED"
},
"reviewers": [
  {
    "user": {
      "name": "User",
      "emailAddress": "user@atlassian.com",
      "id": 36303,
      "displayName": "User",
      "active": true,
      "slug": "user",
      "type": "NORMAL"
    },
    "role": "REVIEWER",
    "approved": false,
    "status": "UNAPPROVED"
  }
],
"participants": [
]
},
"previousTitle": "A cool PR",
"previousDescription": "A neat description",
"previousTarget": {
  "id": "refs/heads/master",
  "displayId": "master",
  "type": "BRANCH",
  "latestCommit": "860c4eb4ed0f969b47144234ba13c31c498cca69",
  "latestChangeset": "860c4eb4ed0f969b47144234ba13c31c498cca69"
}
}

```

Reviewers Updated

Parameter	Description
actor	The user who created the pull request.
pullrequest	Details of the pull request created.
removedReviewers	Users that are no longer reviewers
addedReviewers	Users that have been added as reviewers

```

{
  "eventKey": "pr:reviewer:updated",
  "date": "2018-04-24T10:20:07+1000",
  "actor": {
    "name": "Administrator",
    "emailAddress": "admin@atlassian.com",
    "id": 110653,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "pullRequest": {

```

```

    "id": 1,
    "version": 2,
    "title": "A title",
    "description": "A description",
    "state": "OPEN",
    "open": true,
    "closed": false,
    "createdDate": 1524528879329,
    "updatedDate": 1524529207598,
    "fromRef": {
      "id": "refs/heads/new-branch",
      "displayId": "new-branch",
      "latestCommit": "5a705e60111a4213da46839d9cbf4fc43639b771",
      "repository": {
        "slug": "example",
        "id": 12087,
        "name": "example",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "~ADMIN",
          "id": 8504,
          "name": "Administrator",
          "type": "PERSONAL",
          "owner": {
            "name": "admin",
            "emailAddress": "example@atlassian.com",
            "id": 110653,
            "displayName": "Administrator",
            "active": true,
            "slug": "admin",
            "type": "NORMAL"
          }
        }
      },
      "public": false
    },
    "toRef": {
      "id": "refs/heads/master",
      "displayId": "master",
      "latestCommit": "860c4eb4ed0f969b47144234ba13c31c498cca69",
      "repository": {
        "slug": "example",
        "id": 12087,
        "name": "example",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "~ADMIN",
          "id": 8504,
          "name": "Administrator",
          "type": "PERSONAL",
          "owner": {
            "name": "Administrator",
            "emailAddress": "admin@atlassian.com",
            "id": 110653,
            "displayName": "Administrator",
            "active": true,
            "slug": "admin",
            "type": "NORMAL"
          }
        }
      },
      "public": false
    },
    "locked": false,
    "author": {
      "user": {
        "name": "Administrator",
        "emailAddress": "admin@atlassian.com",
        "id": 110653,
        "displayName": "Administrator",

```

```

    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "role": "AUTHOR",
  "approved": false,
  "status": "UNAPPROVED"
},
"reviewers": [
  {
    "user": {
      "name": "pathompson_admin",
      "emailAddress": "pathompson@atlassian.com",
      "id": 129659,
      "displayName": "Paul Thompson Admin",
      "active": true,
      "slug": "pathompson_admin",
      "type": "NORMAL"
    },
    "role": "REVIEWER",
    "approved": false,
    "status": "UNAPPROVED"
  }
],
"participants": [
]
},
"addedReviewers": [
  {
    "name": "new user",
    "emailAddress": "user2@atlassian.com",
    "id": 129659,
    "displayName": "New User",
    "active": true,
    "slug": "new_user",
    "type": "NORMAL"
  }
],
"removedReviewers": [
  {
    "name": "user",
    "emailAddress": "user@atlassian.com",
    "id": 36303,
    "displayName": "User",
    "active": true,
    "slug": "user",
    "type": "NORMAL"
  }
]
}

```

Approved

A user approves a pull request for a repository. This payload, with an event key of `pr:reviewer:approved`, provides the following fields:

Parameter	Description
<code>actor</code>	The user which made the approval.
<code>pullrequest</code>	Details of the pull request approved.
<code>participant</code>	Details of the PR participant status of the user making the change
<code>previousStatus</code>	The state of the approval before this change

```
{
```

```

"eventKey": "pr:reviewer:approved",
"date": "2017-09-19T10:10:01+1000",
"actor": {
  "name": "user",
  "emailAddress": "user@example.com",
  "id": 2,
  "displayName": "User",
  "active": true,
  "slug": "user",
  "type": "NORMAL"
},
"pullRequest": {
  "id": 1,
  "version": 1,
  "title": "a new file added",
  "description": "A new description, added a user",
  "state": "OPEN",
  "open": true,
  "closed": false,
  "createdDate": 1505779091796,
  "updatedDate": 1505779257496,
  "fromRef": {
    "id": "refs/heads/a-branch",
    "displayId": "a-branch",
    "latestCommit": "ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
    "repository": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",
      "forkable": true,
      "project": {
        "key": "PROJ",
        "id": 84,
        "name": "project",
        "public": false,
        "type": "NORMAL"
      }
    },
    "public": false
  }
},
"toRef": {
  "id": "refs/heads/master",
  "displayId": "master",
  "latestCommit": "178864a7d521b6f5e720b386b2c2b0ef8563e0dc",
  "repository": {
    "slug": "repository",
    "id": 84,
    "name": "repository",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    }
  },
  "public": false
}
},
"locked": false,
"author": {
  "user": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  }
},

```

```

    "role": "AUTHOR",
    "approved": false,
    "status": "UNAPPROVED"
  },
  "reviewers": [
    {
      "user": {
        "name": "user",
        "emailAddress": "user@example.com",
        "id": 2,
        "displayName": "User",
        "active": true,
        "slug": "user",
        "type": "NORMAL"
      },
      "lastReviewedCommit": "ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
      "role": "REVIEWER",
      "approved": true,
      "status": "APPROVED"
    }
  ],
  "participants": [
  ],
  "links": {
    "self": [
      null
    ]
  }
},
"participant": {
  "user": {
    "name": "user",
    "emailAddress": "user@example.com",
    "id": 2,
    "displayName": "User",
    "active": true,
    "slug": "user",
    "type": "NORMAL"
  },
  "lastReviewedCommit": "ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
  "role": "REVIEWER",
  "approved": true,
  "status": "APPROVED"
},
"previousStatus": "UNAPPROVED"
}

```

Unapproved

A user removes an approval from a pull request for a repository. This payload, with an event key of `pr:reviewer:unapproved`, provides the following fields:

Parameter	Description
<code>actor</code>	The user which removed the approval.
<code>pullrequest</code>	Details of the pull request unapproved.
<code>participant</code>	Details of the PR participant status of the user making the change
<code>previousStatus</code>	The state of the approval before this change

```

{
  "eventKey": "pr:reviewer:unapproved",
  "date": "2017-09-19T10:13:43+1000",
  "actor": {
    "name": "user",

```

```

    "emailAddress": "user@example.com",
    "id": 2,
    "displayName": "User",
    "active": true,
    "slug": "user",
    "type": "NORMAL"
  },
  "pullRequest": {
    "id": 1,
    "version": 1,
    "title": "a new file added",
    "description": "A new description, added a user",
    "state": "OPEN",
    "open": true,
    "closed": false,
    "createdDate": 1505779091796,
    "updatedDate": 1505779257496,
    "fromRef": {
      "id": "refs/heads/a-branch",
      "displayId": "a-branch",
      "latestCommit": "ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
      "repository": {
        "slug": "repository",
        "id": 84,
        "name": "repository",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "PROJ",
          "id": 84,
          "name": "project",
          "public": false,
          "type": "NORMAL"
        }
      },
      "public": false
    }
  },
  "toRef": {
    "id": "refs/heads/master",
    "displayId": "master",
    "latestCommit": "178864a7d521b6f5e720b386b2c2b0ef8563e0dc",
    "repository": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",
      "forkable": true,
      "project": {
        "key": "PROJ",
        "id": 84,
        "name": "project",
        "public": false,
        "type": "NORMAL"
      }
    },
    "public": false
  }
},
"locked": false,
"author": {
  "user": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "role": "AUTHOR",
  "approved": false,
  "status": "UNAPPROVED"
},

```

```

    "reviewers":[
      {
        "user":{
          "name":"user",
          "emailAddress":"user@example.com",
          "id":2,
          "displayName":"User",
          "active":true,
          "slug":"user",
          "type":"NORMAL"
        },
        "lastReviewedCommit":"ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
        "role":"REVIEWER",
        "approved":false,
        "status":"UNAPPROVED"
      }
    ],
    "participants":[
      ]
    },
    "participant":{
      "user":{
        "name":"user",
        "emailAddress":"user@example.com",
        "id":2,
        "displayName":"User",
        "active":true,
        "slug":"user",
        "type":"NORMAL"
      },
      "lastReviewedCommit":"ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
      "role":"REVIEWER",
      "approved":false,
      "status":"UNAPPROVED"
    },
    "previousStatus":"APPROVED"
  }
}

```

Needs work

A user marks a pull request as needs work. This payload, with an event key of `pr:reviewer:needs_work`, provides the following fields:

Parameter	Description
<code>actor</code>	The user which marked the PR as "Needs work".
<code>pullrequest</code>	Details of the pull request marked "Needs work".
<code>participant</code>	Details of the PR participant status of the user making the change.
<code>previousStatus</code>	The state of the approval before this change.

```

{
  "eventKey":"pr:reviewer:needs_work",
  "date":"2017-09-19T10:14:47+1000",
  "actor":{
    "name":"user",
    "emailAddress":"user@example.com",
    "id":2,
    "displayName":"User",
    "active":true,
    "slug":"user",
    "type":"NORMAL"
  },
  "pullRequest":{
    "id":1,

```

```

"version":1,
"title":"a new file added",
"description":"A new description, added a user",
"state":"OPEN",
"open":true,
"closed":false,
"createdDate":1505779091796,
"updatedDate":1505779257496,
"fromRef":{
  "id":"refs/heads/a-branch",
  "displayId":"a-branch",
  "latestCommit":"ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
  "repository":{
    "slug":"repository",
    "id":84,
    "name":"repository",
    "scmId":"git",
    "state":"AVAILABLE",
    "statusMessage":"Available",
    "forkable":true,
    "project":{
      "key":"PROJ",
      "id":84,
      "name":"project",
      "public":false,
      "type":"NORMAL"
    },
    "public":false
  }
},
"toRef":{
  "id":"refs/heads/master",
  "displayId":"master",
  "latestCommit":"178864a7d521b6f5e720b386b2c2b0ef8563e0dc",
  "repository":{
    "slug":"repository",
    "id":84,
    "name":"repository",
    "scmId":"git",
    "state":"AVAILABLE",
    "statusMessage":"Available",
    "forkable":true,
    "project":{
      "key":"PROJ",
      "id":84,
      "name":"project",
      "public":false,
      "type":"NORMAL"
    },
    "public":false
  }
},
"locked":false,
"author":{
  "user":{
    "name":"admin",
    "emailAddress":"admin@example.com",
    "id":1,
    "displayName":"Administrator",
    "active":true,
    "slug":"admin",
    "type":"NORMAL"
  },
  "role":"AUTHOR",
  "approved":false,
  "status":"UNAPPROVED"
},
"reviewers":[
  {
    "user":{
      "name":"user",
      "emailAddress":"user@example.com",
      "id":2,
      "displayName":"User",
      "active":true,
      "slug":"user",

```

```

    "type": "NORMAL"
  },
  "lastReviewedCommit": "ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
  "role": "REVIEWER",
  "approved": false,
  "status": "NEEDS_WORK"
}
],
"participants": [
]
},
"participant": {
  "user": {
    "name": "user",
    "emailAddress": "user@example.com",
    "id": 2,
    "displayName": "User",
    "active": true,
    "slug": "user",
    "type": "NORMAL"
  },
  "lastReviewedCommit": "ef8755f06ee4b28c96a847a95cb8ec8ed6ddd1ca",
  "role": "REVIEWER",
  "approved": false,
  "status": "NEEDS_WORK"
},
"previousStatus": "UNAPPROVED"
}
}

```

Merged

A user merges a pull request for a repository. This payload, with an event key of `pullrequest:fulfilled`, provides the following fields:

Parameter	Description
<code>actor</code>	The user who merged the pull request.
<code>pullrequest</code>	Details of the pull request merged.

```

{
  "eventKey": "pr:merged",
  "date": "2017-09-19T10:39:36+1000",
  "actor": {
    "name": "user",
    "emailAddress": "user@example.com",
    "id": 2,
    "displayName": "User",
    "active": true,
    "slug": "user",
    "type": "NORMAL"
  },
  "pullRequest": {
    "id": 9,
    "version": 2,
    "title": "file edited online with Bitbucket",
    "state": "MERGED",
    "open": false,
    "closed": true,
    "createdDate": 1505781560908,
    "updatedDate": 1505781576361,
    "closedDate": 1505781576361,
    "fromRef": {
      "id": "refs/heads/admin/file-1505781548644",
      "displayId": "admin/file-1505781548644",
      "latestCommit": "45f9690c928915a5e1c4366d5ee1985eea03f05d",
      "repository": {
        "slug": "repository",
        "id": 84,

```

```

    "name": "repository",
    "scmId": "git",
    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    },
    "public": false
  },
  },
  "toRef": {
    "id": "refs/heads/master",
    "displayId": "master",
    "latestCommit": "8d2ad38c918fa6943859fca2176c89ea98b92a21",
    "repository": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",
      "forkable": true,
      "project": {
        "key": "PROJ",
        "id": 84,
        "name": "project",
        "public": false,
        "type": "NORMAL"
      },
      "public": false
    }
  },
  },
  "locked": false,
  "author": {
    "user": {
      "name": "admin",
      "emailAddress": "admin@example.com",
      "id": 1,
      "displayName": "Administrator",
      "active": true,
      "slug": "admin",
      "type": "NORMAL"
    },
    "role": "AUTHOR",
    "approved": false,
    "status": "UNAPPROVED"
  },
  "reviewers": [
  ],
  "participants": [
    {
      "user": {
        "name": "user",
        "emailAddress": "user@example.com",
        "id": 2,
        "displayName": "User",
        "active": true,
        "slug": "user",
        "type": "NORMAL"
      },
      "role": "PARTICIPANT",
      "approved": false,
      "status": "UNAPPROVED"
    }
  ],
  "properties": {
    "mergeCommit": {
      "displayId": "7e48f426f0a",
      "id": "7e48f426f0a6e47c5b5e862c31be6ca965f82c9c"
    }
  }
}

```

```

    },
  }
}

```

Declined

A user declines a pull request for a repository. This payload, with an event key of `pr:declined`, provides the following fields:

Parameter	Description
actor	The user who declined the pull request.
pullrequest	Details of the pull request declined.

```

{
  "eventKey": "pr:declined",
  "date": "2017-09-19T11:14:43+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "pullRequest": {
    "id": 10,
    "version": 2,
    "title": "Commit message",
    "state": "DECLINED",
    "open": false,
    "closed": true,
    "createdDate": 1505783668760,
    "updatedDate": 1505783683969,
    "closedDate": 1505783683969,
    "fromRef": {
      "id": "refs/heads/decline-me",
      "displayId": "decline-me",
      "latestCommit": "2d9fb6b9a46eafb1dcef7b008d1a429d45ca742c",
      "repository": {
        "slug": "repository",
        "id": 84,
        "name": "repository",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "PROJ",
          "id": 84,
          "name": "project",
          "public": false,
          "type": "NORMAL"
        }
      },
      "public": false
    }
  },
  "toRef": {
    "id": "refs/heads/master",
    "displayId": "master",
    "latestCommit": "7e48f426f0a6e47c5b5e862c31be6ca965f82c9c",
    "repository": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",

```

```

    "forkable":true,
    "project":{
      "key":"PROJ",
      "id":84,
      "name":"project",
      "public":false,
      "type":"NORMAL"
    },
    "public":false
  }
},
"locked":false,
"author":{
  "user":{
    "name":"admin",
    "emailAddress":"admin@example.com",
    "id":1,
    "displayName":"Administrator",
    "active":true,
    "slug":"admin",
    "type":"NORMAL"
  },
  "role":"AUTHOR",
  "approved":false,
  "status":"UNAPPROVED"
},
"reviewers":[
  {
    "user":{
      "name":"user",
      "emailAddress":"user@example.com",
      "id":2,
      "displayName":"User",
      "active":true,
      "slug":"user",
      "type":"NORMAL"
    },
    "role":"REVIEWER",
    "approved":false,
    "status":"UNAPPROVED"
  }
],
"participants":[]
]
}
}

```

Deleted

A user deletes a pull request for a repository. This payload, with an event key of `pr:deleted`, provides the following fields:

Parameter	Description
actor	The user who deleted the pull request.
pullrequest	Details of the pull request deleted.

```

{
  "eventKey":"pr:deleted",
  "date":"2017-09-19T11:16:17+1000",
  "actor":{
    "name":"admin",
    "emailAddress":"admin@example.com",
    "id":1,
    "displayName":"Administrator",
    "active":true,
    "slug":"admin",
    "type":"NORMAL"
  }
}

```

```

},
"pullRequest":{
  "id":10,
  "version":3,
  "title":"Commit message",
  "state":"OPEN",
  "open":true,
  "closed":false,
  "createdDate":1505783668760,
  "updatedDate":1505783750704,
  "fromRef":{
    "id":"refs/heads/decline-me",
    "displayId":"decline-me",
    "latestCommit":"2d9fb6b9a46eafb1dcef7b008d1a429d45ca742c",
    "repository":{
      "slug":"repository",
      "id":84,
      "name":"repository",
      "scmId":"git",
      "state":"AVAILABLE",
      "statusMessage":"Available",
      "forkable":true,
      "project":{
        "key":"PROJ",
        "id":84,
        "name":"project",
        "public":false,
        "type":"NORMAL"
      },
      "public":false
    }
  },
  "toRef":{
    "id":"refs/heads/master",
    "displayId":"master",
    "latestCommit":"7e48f426f0a6e47c5b5e862c31be6ca965f82c9c",
    "repository":{
      "slug":"repository",
      "id":84,
      "name":"repository",
      "scmId":"git",
      "state":"AVAILABLE",
      "statusMessage":"Available",
      "forkable":true,
      "project":{
        "key":"PROJ",
        "id":84,
        "name":"project",
        "public":false,
        "type":"NORMAL"
      },
      "public":false
    }
  },
  "locked":false,
  "author":{
    "user":{
      "name":"admin",
      "emailAddress":"admin@example.com",
      "id":1,
      "displayName":"Administrator",
      "active":true,
      "slug":"admin",
      "type":"NORMAL"
    },
    "role":"AUTHOR",
    "approved":false,
    "status":"UNAPPROVED"
  },
  "reviewers":[
    {
      "user":{
        "name":"user",
        "emailAddress":"user@example.com",
        "id":2,
        "displayName":"User",

```

```

        "active":true,
        "slug":"user",
        "type":"NORMAL"
    },
    "role":"REVIEWER",
    "approved":false,
    "status":"UNAPPROVED"
  }
],
"participants":[
]
}
}
}

```

Comment added

A user comments on a pull request. This payload, with an event key of `pr:comment:added`, provides the following fields:

Parameter	Description
actor	The user that created the comment.
pullRequest	The pull request comment on.
comment	The comment created.
commentParentId	Id of the parent comment if one exists.

```

{
  "eventKey":"pr:comment:added",
  "date":"2017-09-19T11:21:06+1000",
  "actor":{
    "name":"admin",
    "emailAddress":"admin@example.com",
    "id":1,
    "displayName":"Administrator",
    "active":true,
    "slug":"admin",
    "type":"NORMAL"
  },
  "pullRequest":{
    "id":11,
    "version":1,
    "title":"A cool PR",
    "state":"OPEN",
    "open":true,
    "closed":false,
    "createdDate":1505783860548,
    "updatedDate":1505783878981,
    "fromRef":{
      "id":"refs/heads/comment-pr",
      "displayId":"comment-pr",
      "latestCommit":"ddc19f786996396d57e17c8f6d1d05d00318ad10",
      "repository":{
        "slug":"repository",
        "id":84,
        "name":"repository",
        "scmId":"git",
        "state":"AVAILABLE",
        "statusMessage":"Available",
        "forkable":true,
        "project":{
          "key":"PROJ",
          "id":84,
          "name":"project",
          "public":false,

```

```

        "type": "NORMAL"
      },
      "public": false
    }
  },
  "toRef": {
    "id": "refs/heads/master",
    "displayId": "master",
    "latestCommit": "7e48f426f0a6e47c5b5e862c31be6ca965f82c9c",
    "repository": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",
      "forkable": true,
      "project": {
        "key": "PROJ",
        "id": 84,
        "name": "project",
        "public": false,
        "type": "NORMAL"
      },
      "public": false
    }
  },
  "locked": false,
  "author": {
    "user": {
      "name": "admin",
      "emailAddress": "admin@example.com",
      "id": 1,
      "displayName": "Administrator",
      "active": true,
      "slug": "admin",
      "type": "NORMAL"
    },
    "role": "AUTHOR",
    "approved": false,
    "status": "UNAPPROVED"
  },
  "reviewers": [
  ],
  "participants": [
  ]
},
"comment": {
  "properties": {
    "repositoryId": 84
  },
  "id": 62,
  "version": 0,
  "text": "I am a PR comment",
  "author": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "createdAt": 1505784066751,
  "updatedAt": 1505784066751,
  "comments": [
  ],
  "tasks": [
  ]
},
"commentParentId": 43
}

```

Comment edited

This payload, with an event key of `pr:comment:edited`, provides the following fields:

Parameter	Description
<code>actor</code>	The user that edited the comment.
<code>pullRequest</code>	The pull request where the comment exists.
<code>comment</code>	The comment edited.
<code>commentParentId</code>	Id of the parent comment if one exists.
<code>previousComment</code>	Text of the previous comment.

```
{
  "eventKey": "pr:comment:edited",
  "date": "2017-09-19T11:24:19+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "pullRequest": {
    "id": 11,
    "version": 1,
    "title": "A cool PR",
    "state": "OPEN",
    "open": true,
    "closed": false,
    "createdDate": 1505783860548,
    "updatedDate": 1505783878981,
    "fromRef": {
      "id": "refs/heads/comment-pr",
      "displayId": "comment-pr",
      "latestCommit": "ddc19f786996396d57e17c8f6d1d05d00318ad10",
      "repository": {
        "slug": "repository",
        "id": 84,
        "name": "repository",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "PROJ",
          "id": 84,
          "name": "project",
          "public": false,
          "type": "NORMAL"
        }
      },
      "public": false
    }
  },
  "toRef": {
    "id": "refs/heads/master",
    "displayId": "master",
    "latestCommit": "7e48f426f0a6e47c5b5e862c31be6ca965f82c9c",
    "repository": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",

```

```

    "state": "AVAILABLE",
    "statusMessage": "Available",
    "forkable": true,
    "project": {
      "key": "PROJ",
      "id": 84,
      "name": "project",
      "public": false,
      "type": "NORMAL"
    },
    "public": false
  }
},
"locked": false,
"author": {
  "user": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "role": "AUTHOR",
  "approved": false,
  "status": "UNAPPROVED"
},
"reviewers": [
],
"participants": [
]
},
"comment": {
  "properties": {
    "repositoryId": 84
  },
  "id": 62,
  "version": 1,
  "text": "I am a PR comment that was edited",
  "author": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "createdDate": 1505784066751,
  "updatedDate": 1505784259446,
  "comments": [
],
  "tasks": [
]
},
"commentParentId": 43,
"previousComment": "I am a PR comment"
}

```

Comment deleted

A user deletes a comment on a pull request. This payload, with an event key of `pr:comment:deleted`, provides the following fields:

Parameter	Description
-----------	-------------

actor	The user that deleted the comment.
pullRequest	The pull request where the comment existed.
comment	The comment deleted.
commentParentId	Id of the parent comment if one exists.

```
{
  "eventKey": "pr:comment:deleted",
  "date": "2017-09-19T11:25:47+1000",
  "actor": {
    "name": "admin",
    "emailAddress": "admin@example.com",
    "id": 1,
    "displayName": "Administrator",
    "active": true,
    "slug": "admin",
    "type": "NORMAL"
  },
  "pullRequest": {
    "id": 11,
    "version": 1,
    "title": "A cool PR",
    "state": "OPEN",
    "open": true,
    "closed": false,
    "createdDate": 1505783860548,
    "updatedDate": 1505783878981,
    "fromRef": {
      "id": "refs/heads/comment-pr",
      "displayId": "comment-pr",
      "latestCommit": "ddc19f786996396d57e17c8f6d1d05d00318ad10",
      "repository": {
        "slug": "repository",
        "id": 84,
        "name": "repository",
        "scmId": "git",
        "state": "AVAILABLE",
        "statusMessage": "Available",
        "forkable": true,
        "project": {
          "key": "PROJ",
          "id": 84,
          "name": "project",
          "public": false,
          "type": "NORMAL"
        }
      },
      "public": false
    }
  },
  "toRef": {
    "id": "refs/heads/master",
    "displayId": "master",
    "latestCommit": "7e48f426f0a6e47c5b5e862c31be6ca965f82c9c",
    "repository": {
      "slug": "repository",
      "id": 84,
      "name": "repository",
      "scmId": "git",
      "state": "AVAILABLE",
      "statusMessage": "Available",
      "forkable": true,
      "project": {
        "key": "PROJ",
        "id": 84,
        "name": "project",
        "public": false,
        "type": "NORMAL"
      }
    },
    "public": false
  }
}
```

```
    }
  },
  "locked":false,
  "author":{
    "user":{
      "name":"admin",
      "emailAddress":"admin@example.com",
      "id":1,
      "displayName":"Administrator",
      "active":true,
      "slug":"admin",
      "type":"NORMAL"
    },
    "role":"AUTHOR",
    "approved":false,
    "status":"UNAPPROVED"
  },
  "reviewers":[

],
  "participants":[

]
},
"comment":{
  "id":62,
  "version":1,
  "text":"I am a PR comment that was edited",
  "author":{
    "name":"admin",
    "emailAddress":"admin@example.com",
    "id":1,
    "displayName":"Administrator",
    "active":true,
    "slug":"admin",
    "type":"NORMAL"
  },
  "createdDate":1505784066751,
  "updatedDate":1505784259446,
  "comments":[

],
  "tasks":[

]
},
"commentParentId":43
}
```

If you use self-signed certificates

 Atlassian applications allow the use of SSL within our products, however Atlassian Support does not provide assistance for configuring it. Consequently, Atlassian **cannot guarantee providing any support for it.**

- If assistance with conversions of certificates is required, please consult with the vendor who provided the certificate.
- If assistance with configuration is required, please raise a question on [Atlassian Answers](#).

In order to trust a self-signed certificate, the public certificate need to be imported in the Java keystore that Bitbucket Data Center and Server uses. In this example, it is `*.atlassian.com`, and we cover how to install it below.

If you're unable to install Portecle on the server or prefer the command line please see our [Command Line Installation](#) section below.

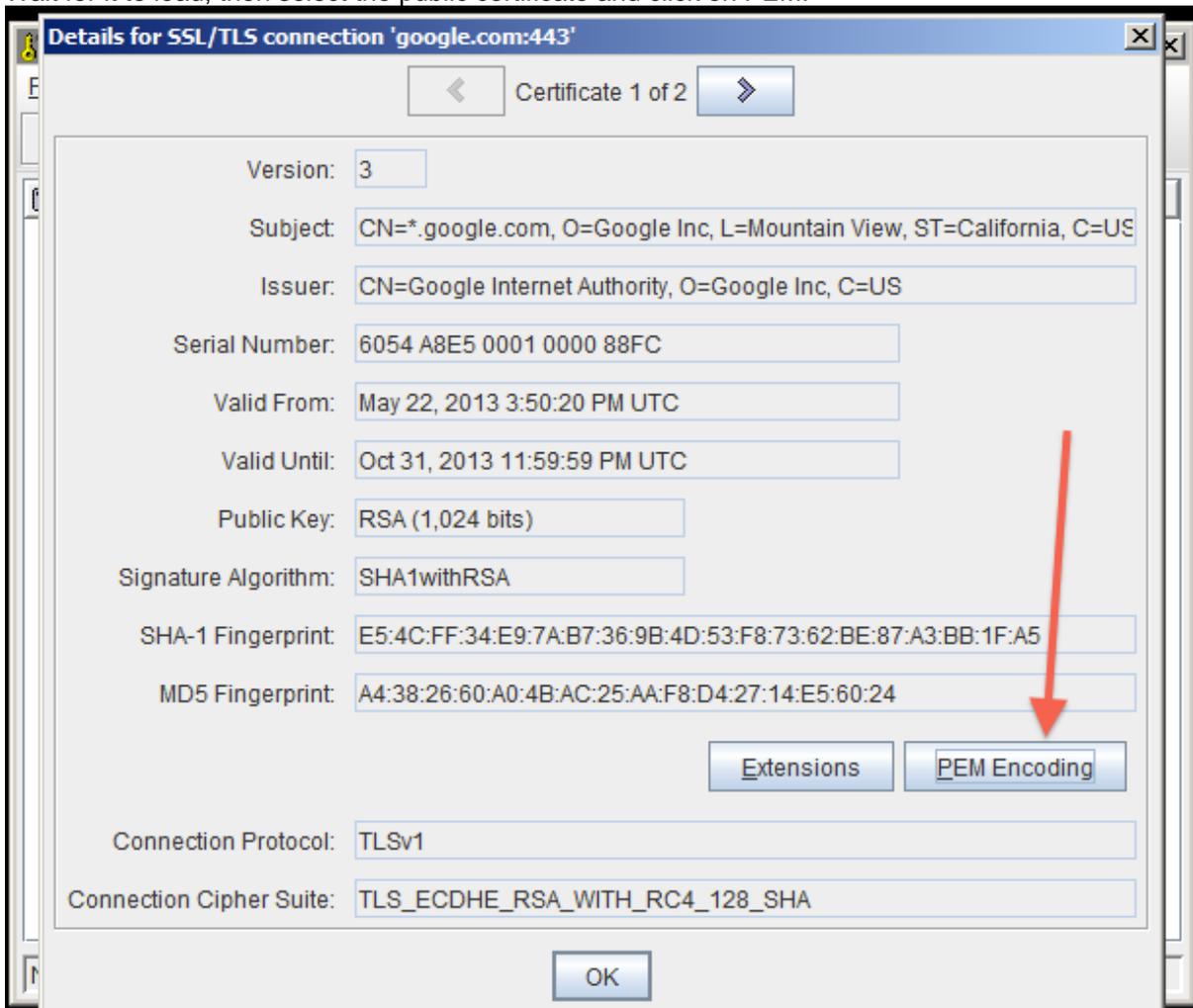
Obtain and import the endpoint's public certificate

1. Download and install the [Portecle](#) app onto the server that runs Bitbucket.
 This is a third-party application and not supported by Atlassian.
2. Ensure the `<JAVA_HOME>` variable is pointing to the same version of Java that Bitbucket uses.
 If running on a Linux/UNIX server, X11 will need to be forwarded when connecting to the server (so you can use the GUI), as below:

```
ssh -X user@server
```

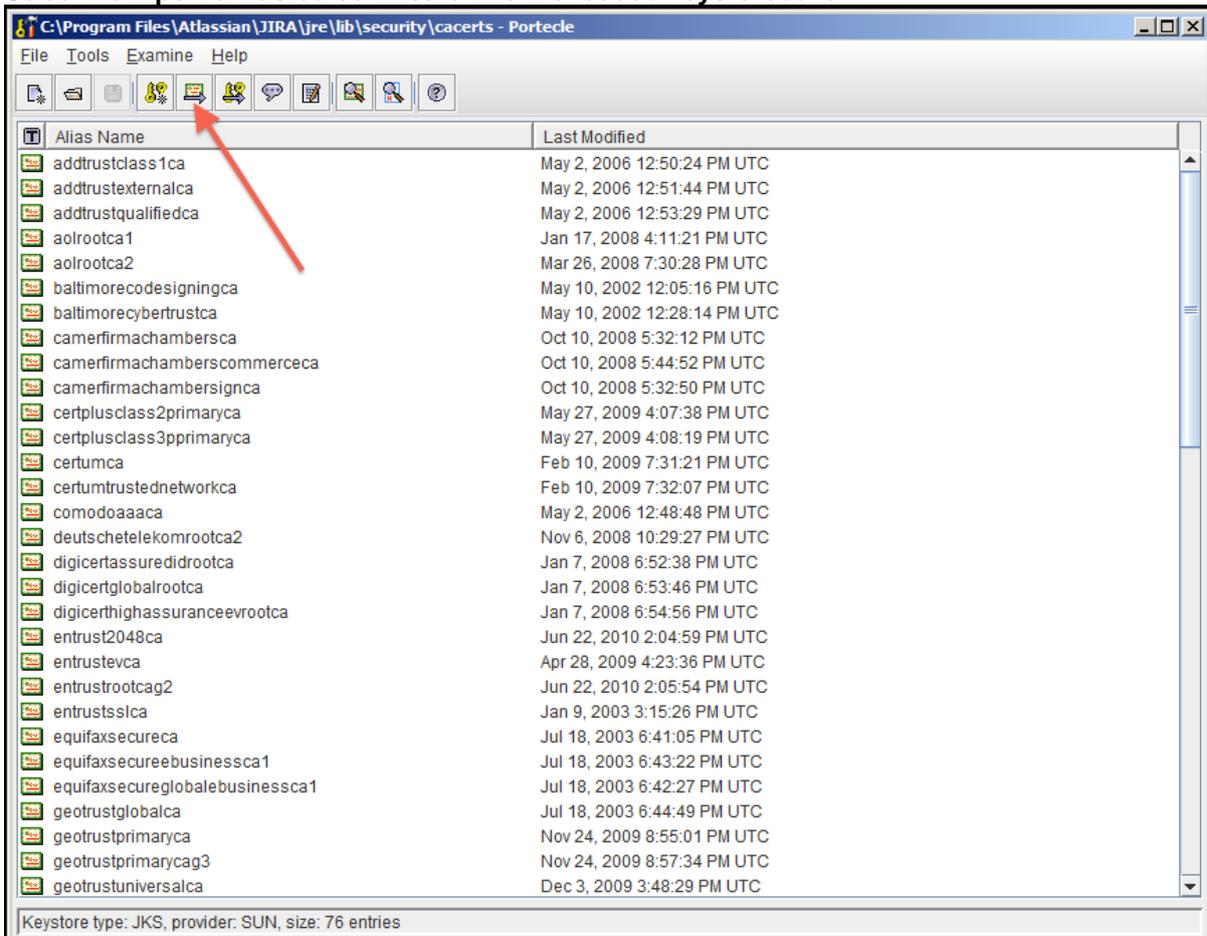
3. Select the **Examine** menu and then click **Examine SSL/TLS Connection**:
4. Enter the SSL Host and Port of the target system:

5. Wait for it to load, then select the public certificate and click on PEM:



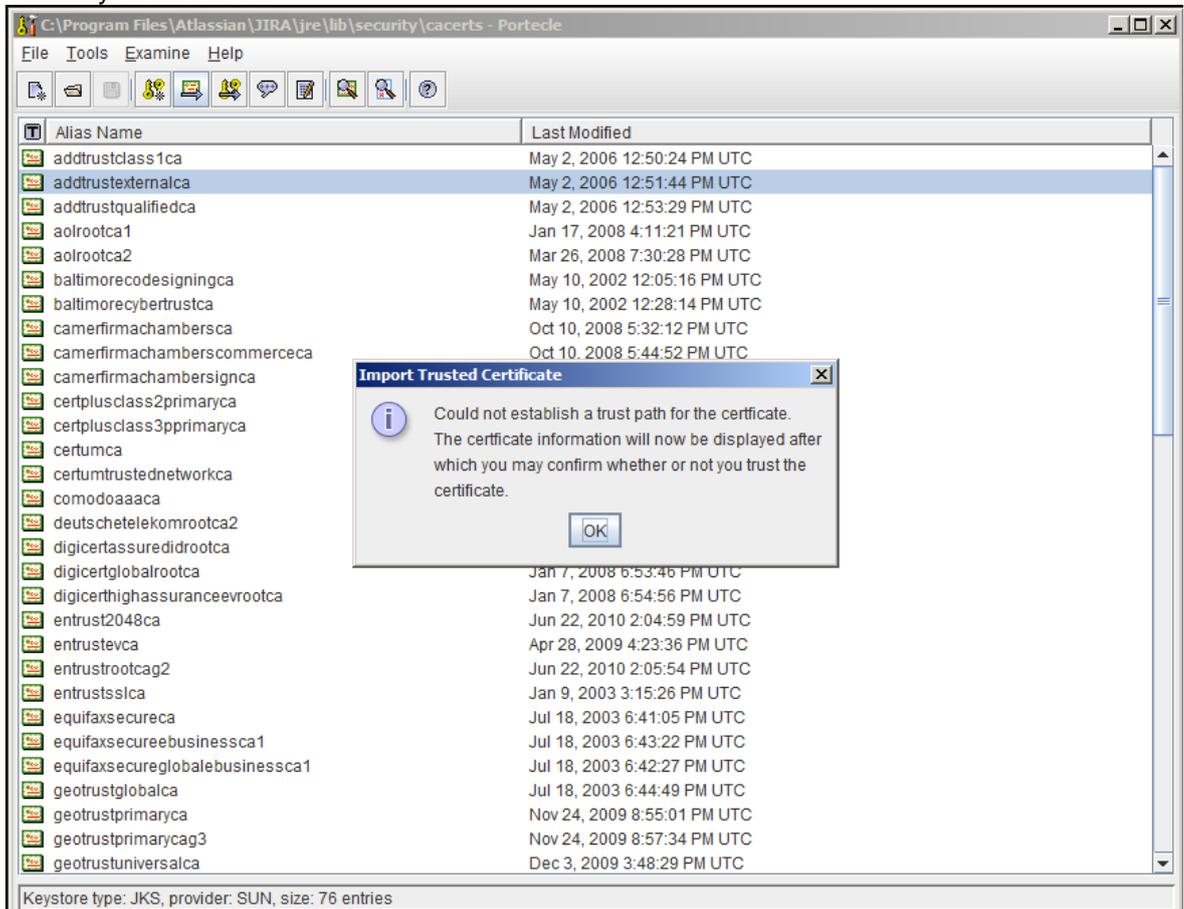
6. Export the certificate and save it.
7. Go back to the main screen and select the **Open an existing keystore from disk** option, select `cacerts` (for example `$JAVA_HOME/lib/security/cacerts`) then enter the password (the default is `changeit`).

8. Select the **Import a trusted certificate into the loaded keystore** button:



9. Select the certificate that was saved in step 6 and confirm that you trust it, giving it an appropriate alias (e.g.: confluence).

a. You may hit this error:



- b. If so, hit OK, and then accept the certificate as trusted.
10. Save the Key Store to disk:
11. Restart Bitbucket.
12. Test that you can connect to the host.

Command Line Installation

1. Fetch the certificate, replacing `google.com` with the FQDN of the server Bitbucket is attempting to connect to:

Unix:

```
openssl s_client -connect google.com:443 < /dev/null | sed -ne '/-BEGIN
CERTIFICATE-/,/-END CERTIFICATE-/p' > public.crt
```

Windows:

```
openssl s_client -connect google.com:443 < NUL | sed -ne '/-BEGIN CERTIFICATE-
/,/-END CERTIFICATE-/p' > public.crt
```

 The command above will only be executed if you have [Sed for Windows](#) as well as [OpenSSL](#) installed on your environment. If you don't have Sed or OpenSSL or you don't want to install it, use the instructions below as an alternative. Issue the following command:

```
openssl s_client -connect google.com:443
```

Save the output to a file called `public.crt`. Edit the `public.crt` file so it contains only what is between the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines. This is how your file should look like after you edited it:

```
-----BEGIN CERTIFICATE-----
< Certificate content as fetched by the command line.
Don't change this content, only remove what is before
and after the BEGIN CERTIFICATE and END CERTIFICATE.
That's what your Sed command is doing for you :- ) >
-----END CERTIFICATE-----
```

- Import the certificate:

```
<JAVA_HOME>/bin/keytool -import -alias <server_name> -keystore <JAVA_HOME>/jre
/lib/security/cacerts -file public.crt
```

Alternative KeyStore Locations

Java will normally use a system-wide keystore in `$JAVA_HOME/jre/lib/security/cacerts`, but it is possible to use a different keystore by specifying a parameter, `-Djavax.net.ssl.trustStore=/path/to/keystore`, where `'/path/to/keystore'` is the absolute file path of the alternative keystore.

However, setting this is **not recommended** because if Java is told to use a custom keystore (eg. containing a self-signed certificate), then Java will not have access to the root certificates of signing authorities found in `$JAVA_HOME/jre/lib/security/cacerts`, and accessing most CA-signed SSL sites will fail. It is better to add new certificates (eg. self-signed) to the system-wide keystore (as above).

Debugging

Problems are typically one of two forms:

- The certificate was installed into the incorrect keystore.
- The keystore does not contain the certificate of the SSL service you're connecting to.

Notifications

An email server must be configured in Bitbucket Data Center and Server for email notifications to be sent. See [Setting up your mail server](#). Note that if the mail server fails, notifications will be dropped. See also [Hipchat notifications](#).

Batched email notifications

Notifications are aggregated by user for each pull request and are emailed in a batch. The batch gets sent if things go quiet for a while (default is 10 minutes), or when the oldest notification gets 'stale' (default is 30 minutes), whichever comes first.

By default, email notifications are batched. However:

- You can change your personal account settings (on the **Notification settings** tab) so that you receive notifications immediately.
- Pull request notifications and repository notifications have separate settings for notifications to be batched or received immediately. E.g., you can opt to receive pull request notifications immediately, but receive repository notifications in a batch.
- A Bitbucket admin can configure the period of inactivity and the staleness timeout period in the [Bitbucket Server config properties file](#).
- A Bitbucket admin can change the notification mode for the Bitbucket instance to 'immediate' using a [system property](#), but users can still opt in for batched notifications.

Using 'mentions' to notify someone

From Bitbucket 2.0, 'mentions' can be used to notify another user about the pull request description or comment you are writing. Bitbucket sends an email to that person the emails are [batched](#) if they have opted for batching in their personal account settings.

To use mentions, simply start typing '@' and then the users display name, username or email address, and choose from the list that Bitbucket offers. You can use quotes for unusual names, for example if it has spaces. Use Control-Shift-P or Command-Shift-P to preview the mention.

Pull request notifications

Bitbucket Data Center and Server sends email notifications to the author, reviewers, and [watchers](#) of a pull request when the below events occur.

Pull request event	Notification recipients
A pull request is opened	The reviewers of the pull request
The pull request is merged	The watchers of the pull request
The pull request is declined	The watchers of the pull request
The pull request is reopened	The watchers of the pull request
The pull request is deleted	The watchers of the pull request
A reviewer is added	The added reviewer
A reviewer is removed	The removed reviewer
The reviewers are updated	The pull request author
The pull request is approved	The watchers of the pull request
The pull request is reviewed	The pull request author
The pull request is unapproved	The watchers of the pull request
A commit is made to the source branch	The watchers of the pull request
The destination branch is changed	The watchers of the pull request
A comment is added	The watchers of the pull request
A comment is replied to	The watchers of the pull request
A comment is liked	The pull request author and the comment author

The pull request author and reviewers are automatically added as watchers.

By default, pull request notifications are [batched](#). You can change your personal account settings in Bitbucket (from the **Notification settings** tab) so that you get all pull request notifications immediately.

However, in the following situations, notifications are always sent immediately:

- When a pull request is first opened, notifications are immediately sent to the reviewers.
- When a pull request is deleted, notifications are immediately sent to all watchers.
- When a pull request is reopened, notifications are immediately sent to the reviewers who have opted in for immediate notifications.
- When someone is added as a reviewer to a pull request, a notification is immediately sent to them.
- When someone is removed as a reviewer from a pull request, a notification is immediately sent to them.
- When someone is mentioned in the description of a pull request, a notification is immediately sent to them.

You don't receive notifications for events you initiate yourself. See also [Using pull requests in Bitbucket Server](#).

Repository notifications

You can watch a repository by visiting the source page of the repository and clicking the **watch** button. From here you can select which events to watch.

 The **watch** button will only be visible if an email server has been configured in Bitbucket Data Center and Server. For information on how to do this see [Setting up your mail server](#).

Bitbucket sends email notifications to the **watchers of a repository** when the following events occur.

Pull request event	Notification recipients
A pull request is opened	The watchers of the repository with a pull request notification scope of STATE CHANGES or ALL ACTIVITY
A pull request is merged	The watchers of the repository with a pull request notification scope of STATE CHANGES or ALL ACTIVITY
A pull request is declined	The watchers of the repository with a pull request notification scope of STATE CHANGES or ALL ACTIVITY
A pull request is reopened	The watchers of the repository with a pull request notification scope of STATE CHANGES or ALL ACTIVITY
A pull request is deleted	The watchers of the repository with a pull request notification scope of STATE CHANGES or ALL ACTIVITY
The pull request is approved	The watchers of the repository with a pull request notification scope of ALL ACTIVITY
The pull request is unapproved	The watchers of the repository with a pull request notification scope of ALL ACTIVITY
A commit is made to a pull request source branch	The watchers of the repository with a pull request notification scope of ALL ACTIVITY
The destination branch is changed	The watchers of the repository with a pull request notification scope of ALL ACTIVITY
A comment is added	The watchers of the repository with a pull request notification scope of ALL ACTIVITY
A comment is replied to	The watchers of the repository with a pull request notification scope of ALL ACTIVITY
A commit is pushed to the repository on the default branch	The watchers of the repository with a push notification scope of DEFAULT BRANCH
A commit is pushed to the repository on a non-default branch	The watchers of the repository with a push notification scope of DEFAULT BRANCH or ALL BRANCHES

Pull request notifications

The below notification scopes are available for **pull requests** when watching a repository.

Notification scope	Explanation
--------------------	-------------

All activity	Notifications will be sent for all pull request activity within the repository.
State changes	Notifications will only be sent for pull request state changes (opened, merged, declined, reopened, deleted) within the repository.
None	No notifications will be sent for pull requests within the repository (for pull requests you are not watching).

Note that if you're watching a particular pull request, this takes precedence over your repository pull request notification scope. For example, if you are watching a repository with the `state changes` pull request option selected, and are watching a particular pull request, you will receive all notifications for that pull request, and only state change notifications for other pull requests.

 Notifications for a pull request being *deleted* are only sent to **immediate** repository watchers (and watchers of the pull request). **Batched** repository watchers will not receive this notification.

Push notifications

The below notification scopes are available for **pushes** when watching a repository.

Notification scope	Explanation
All branches	Notifications will be sent for all pushes to the repository.
Default branch	Notifications will be sent for pushes to the default branch of the repository.
None	No notifications will be sent for pushes to the repository.

 A notification will only be sent if a push **adds or removes at least one commit**. If the push updates one or more refs without actually adding or removing a commit (e.g., pushing a new tag to a commit already in the repository on Bitbucket), then no push notification will be sent.

If you watch a repository *and* a specific pull request within it, you will receive **pull request notifications** for that pull request.

Your repository notifications will not include this pull request. Repository notifications about pull requests will **only** be sent for pull requests you are not watching.

You can manage all of your watched repositories in your Bitbucket personal account settings (on the **Watched repositories** tab).

By default, repository notifications are **batched**. You can change your personal account settings (on the **Notification settings** tab) so that you get all repository notifications immediately.

You don't receive notifications for events you initiate yourself.

Markdown syntax guide

Bitbucket Data Center and Serveruses Markdown for formatting text, as specified in [CommonMark](#) (with a few extensions). You can use Markdown in the following places:

- any pull request's descriptions or comments, or
- in [README](#) files (if they have the .md file extension).

Use *Control-Shift-P* or *Command-Shift-P* to preview your markdown.

Markdown syntax

The page below contains examples of Markdown syntax. For a full list of all the Markdown syntax, consult the [CommonMark help](#) or [specification](#).

On this page:

- [Markdown syntax](#)
 - [Headings](#)
 - [Paragraphs](#)
 - [Character styles](#)
 - [Unordered list](#)
 - [Ordered list](#)
 - [List in list](#)
 - [Quotes or citations](#)
 - [Inline code characters](#)
 - [Code blocks](#)
 - [Links to external websites](#)
 - [Linking issue keys to JIRA applications](#)
 - [Linking to pull requests](#)
- [Images](#)
 - [Tables](#)
- [Backslash escapes](#)
- [README files](#)

Headings

```
# This is an H1
## This is an H2
##### This is an H6

This is also an H1
=====

This is also an H2
-----
```

Paragraphs

Paragraphs are separated by empty lines. To create a new paragraph, press <return> twice.

```
Paragraph 1

Paragraph 2
```

Character styles

```
*Italic characters*
_Italic characters_
**bold characters**
__bold characters__
~~strikethrough text~~
```

Unordered list

```
* Item 1
* Item 2
* Item 3
  * Item 3a
  * Item 3b
  * Item 3c
```

Ordered list

1. Step 1
2. Step 2
3. Step 3
 1. Step 3.1
 2. Step 3.2
 3. Step 3.3

List in list

1. Step 1
2. Step 2
3. Step 3
 - * Item 3a
 - * Item 3b
 - * Item 3c

Quotes or citations

Introducing my quote:

```
> Neque porro quisquam est qui
> dolorem ipsum quia dolor sit amet,
> consectetur, adipisci velit...
```

Inline code characters

Use the backtick to refer to a `function()`.

There is a literal ```backtick (`)``` here.

Code blocks

Indent every line of the block by at least 4 spaces.

This is a normal paragraph:

```
    This is a code block.
    With multiple lines.
```

Alternatively, you can use 3 backtick quote marks before and after the block, like this:

```
```
This is a code block
```
```

To add syntax highlighting to a code block, add the name of the language immediately after the backticks:

```
```javascript
var oldUnload = window.onbeforeunload;
window.onbeforeunload = function() {
 saveCoverage();
 if (oldUnload) {
 return oldUnload.apply(this, arguments);
 }
};
```
```

Bitbucket uses CodeMirror to apply syntax highlighting to the rendered markdown in comments, READMEs and pull request descriptions. All the common coding languages are supported, including C, C++, Java, Scala, Python and JavaScript. See [Configuring syntax highlighting for file extensions](#).

Within a code block, ampersands (&) and angle brackets (< and >) are automatically converted into HTML entities.

Links to external websites

```
This is [an example](http://www.example.com/) inline link.

[This link](http://example.com/ "Title") has a title attribute.

Links are also auto-detected in text: http://example.com/
```

Linking issue keys to JIRA applications

When you use Jira application issue keys (of the default format) in comments and pull request descriptions Bitbucket automatically links them to the Jira application instance.

The default Jira application issue key format is two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example TEST-123.

Linking to pull requests

Introduced with Bitbucket 4.9, you can reference pull requests from comments and descriptions in other pull requests. The syntax for linking to pull request looks like this:

```
projectkey/repo-slug#pr_id
```

To link to a pull request in the *same project and repository*, you only need to include the pull request ID.

```
#123
```

To link to a pull request in the *same project but a different repository*, include the repository slug before the pull request ID.

```
example-repo#123
```

To link to a pull request in a *different project and repository*, include the project key and the repository slug before the pull request ID.

```
PROJ/example-repo#123
```

Images

Inline image syntax looks like this:

```
![Alt text](/path/to/image.jpg)
![Alt text](/path/to/image.png "Optional title attribute")
![Alt text](/url/to/image.jpg)
```

For example:

```
...
![Mockup for feature A](http://monosnap.com/image/b0cxxxxLGF.png)
...
```

Reference image links look like this:

```
![Alt text][id]
```

Where 'id' is the name of a previously defined image reference, using syntax similar to link references:

```
[id]: url/to/image.jpg "Optional title attribute"
```

An example using reference image links:

```
...
<!--Collected image definitions-->
[MockupA]: http://monosnap.com/image/b0cxxxxLGF.png "Screenshot of Feature A mockup"
...
<!--Using an image reference-->
![Mockup for feature A][MockupA]
...
```

You can specify image height and width as `key=value` pairs inside curly braces `{ }` after the applicable image node. For example:

```
![text](/url.png){width=640 height=480}
```

Tables

| Day | Meal | Price |
|---------|---------|-------|
| Monday | pasta | \$6 |
| Tuesday | chicken | \$8 |

Backslash escapes

Certain characters can be escaped with a preceding backslash to preserve the literal display of a character instead of its special Markdown meaning. This applies to the following characters:

- \ backslash
- ` backtick
- * asterisk
- _ underscore
- { } curly braces
- [] square brackets
- () parentheses
- # hash mark
- > greater than
- + plus sign
- minus sign (hyphen)
- . dot
- ! exclamation mark

README files

If your repository contains a `README.md` file at the root level, Bitbucket displays its contents on the repository's **Source** page if the file has the `.md` extension. The file can contain [Markdown](#) and a restricted set of HTML tags.

Requesting add-ons

The [Atlassian Marketplace](#) website offers hundreds of add-ons that your administrator can install to enhance and extend Atlassian Bitbucket Data Center and Server. If the add-on request feature is enabled for your Bitbucket instance, you can submit requests for add-ons from the Marketplace to your Bitbucket administrator.

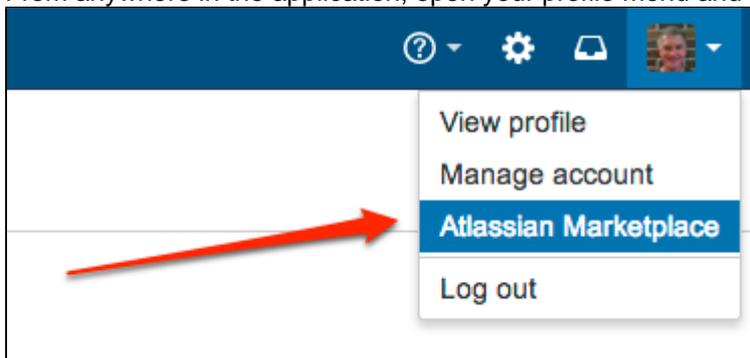
The 'Atlassian Marketplace for Bitbucket' page provides an integrated view of the Atlassian Marketplace from within your Bitbucket instance. The page offers the same features as the Marketplace website, such as searching and category filtering, but tailors the browsing experience to Bitbucket.

This in-product view of the Marketplace gives day-to-day Bitbucket users, not just administrators, an easy way to discover add-ons that can help them get work done. When you find an add-on of interest, you can submit a request to your administrator for the add-on with just a few clicks.

Submitting an add-on request

To browse for add-ons in the Atlassian Marketplace, follow these steps:

1. From anywhere in the application, open your profile menu and choose **Atlassian Marketplace**:



2. In the Atlassian Marketplace page, use the search box to find add-ons or use the category menus to browse or filter by add-ons by type, popularity, price or other criteria. You can see what your fellow users have requested by choosing the **Most Requested** filter.
3. When you find an add-on that interests you, click **Request** to generate a request for your administrator.
4. Optionally, type a personal message to your administrators in the text box. This message is visible to administrators in the details view for the add-on.
5. Click **Submit Request** when done.
6. Click **Close** to dismiss the 'Success!' message dialog box.

At this point, a notification appears in the interface your administrators use to administer add-ons. Also, your request message will appear in the add-on details view, visible from the administrator's 'Find New Add-ons' page. From there, your administrator can purchase the add-on, try it out or dismiss requests.

Updating an add-on request

After submitting the request, you can update your message at any time. Click the **Update Request** button next to the listing in the Atlassian Marketplace page to modify the message to your administrator.

The administrator is not notified of the update. However, your updated message will appear as you have modified it in the details view for the add-on immediately.

Set the default time zone

You can adjust the default time zone in Bitbucket Data Center and Server individually or for all users. Setting time zones gives remote users and distributed teams accurate, local timestamps in the application and notification emails. However, setting the default time zone for all users won't override individual user's time zone settings.

To set the default time zone for all users:

1. Log in to Bitbucket as an admin.
2. From the administration area, select **Server settings** (under 'Settings').
3. In the **Time Zone** field, specify the appropriate time zone.

To set the default time zone for a single user (yourself):

1. Click your profile picture (in the upper-right), then select **Manage account**.
2. In the **Time Zone** field, specify the appropriate time zone.

Download a repository archive

With Bitbucket Data Center and Server you can download an archive of source files at a particular point in time; you can download your source as a .zip file from the actions dropdown menu from the Source view, Commits list, and Branches list. You can also download the archive of individual branches, commits, and tags.

On this page:

- [What's the difference between downloading and cloning a repository?](#)
- [Download your source as a .zip file](#)
- [Download your source as a .tar or .tar.gz file](#)

What's the difference between *downloading* and *cloning* a repository?

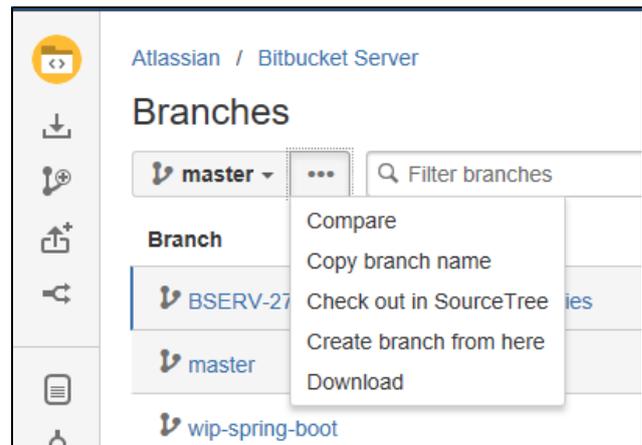
Cloning a repository copies the source files of a remote repository to your local machine, as well as the repository's Git history (branches, commits, tags, etc.). Cloning also creates a remote connection (usually called origin) pointing back to the cloned repository. You can only clone into a local directory that is a properly initialized Git repository (using the `git init` command).

Downloading a repository archive only copies a repository's source files from a specific point in time, depending on what was chosen to be copied. The biggest difference to downloading an archive is that you are not copying the repository history, or creating a connection to the remote repository. You are only getting the source files, and none of the Git metadata stored in the `.git` directory.

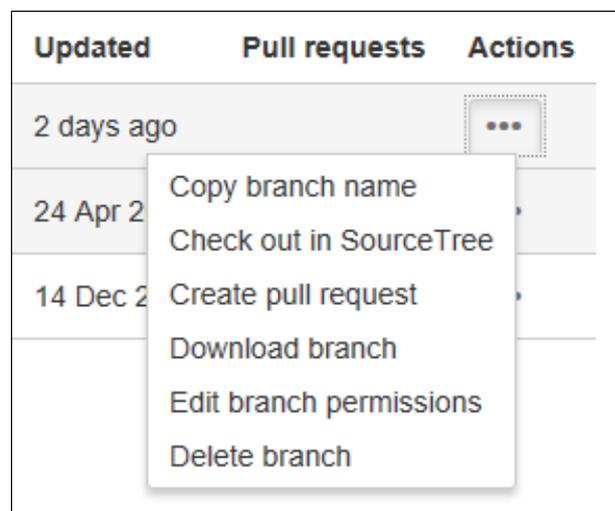
Download your source as a .zip file

To download a repository archive of a branch:

1. Go to either the Source view, Commit view, or Branches list of a repository.
2. Using the branch selector to choose a branch.
3. Click the actions dropdown next to the branch selector, then select **Download**.

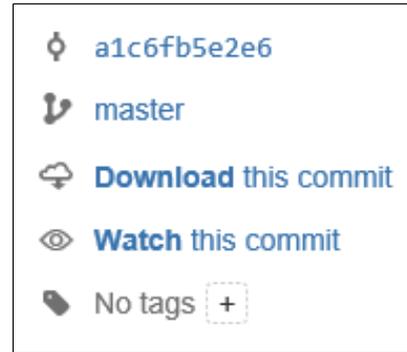


You can also download a repository archive of a specific branch in the actions menu of any branch, as viewed from the Branches list.



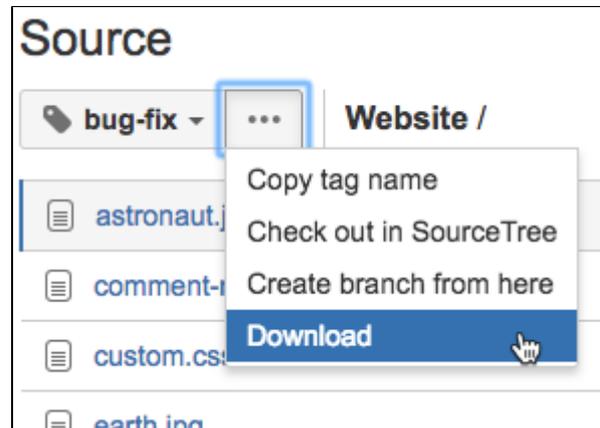
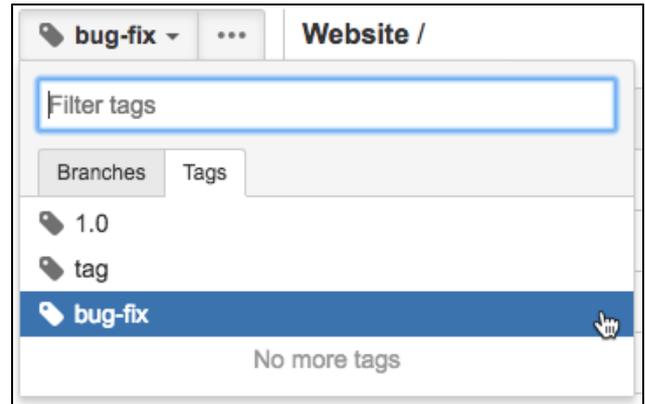
To download a repository archive from a commit:

1. Click on the hash number of a commit to view a single commit.
2. Click **Download this commit** in the upper-right panel.



To download a repository archive from a tag:

1. Go to either the Source view, Commit view, or Branches list of a repository.
2. Using the branch selector to choose a tag.
3. Click the actions dropdown next to the branch selector, then select **Download**.



Download your source as a .tar or .tar.gz file

When you download your source file from Bitbucket's UI, you are downloading the file in .zip format. However, it is possible to edit the URL and get the archive as other formats, like .tar, .gz, or .tar.gz.

To download your source as a file format other than .zip:

1. Follow any of the instructions [listed above](#), so that you are viewing the source you want to download.
2. Right-click the Download link and copy the URL address.
3. Paste the URL in the browser, it would look something like this:

```
.../projects/TIS/repos/website/archive?format=zip
```

4. Change **format=zip** to the format of your choice.

For **.tar.gz files**, change the argument to **format=tgz**.

For **.tar files**, change the argument to **format=tar**.

5. When you hit **Enter** the file will download from your browser.

Creating a Contributions guidelines file

Bitbucket Data Center and Serverlets you and your team create and share Git repositories, which allows you to collaborate with other developers within your organization. In most cases, individuals or teams will have some guidelines that they use when working on a repository, such as recommendations on who should review a pull request, info on best practices for coding, or desired file structures. You can add these guidelines to a file in your repo, and a link will be displayed to this file when:

- you access **Clone** in your sidebar (the link will direct you to the relevant page),
- you're about to create a pull request (the link will display the guidelines in a dialog), and
- you're viewing a pull request (the link will display the guidelines in a dialog).

To construct your file, it must:

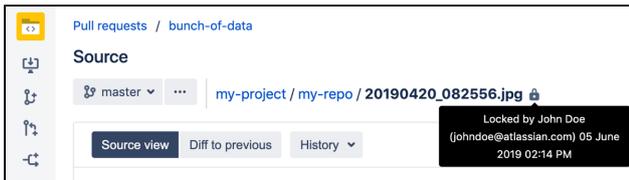
- be saved to the root directory of your repository,
- have the title **Contributing** which is not case sensitive,
- the contents of the file can be constructed using the [markdown syntax](#) in Bitbucket, and
- the file can have no extension, or the following extensions:
 - .md
 - .markdown
 - .mdown
 - .mkdn
 - .mkd
 - .text
 - .txt

Once you've created your file, you and your teammates can add and update the content, just as you would for any file in your repository.

Working with Git LFS Files

Git Large File Storage (LFS) is a Git extension that improves how large files are handled. It replaces them with tiny text pointers that are stored on a remote server instead of in their repository, speeding up operations like cloning and fetching in Bitbucket Data Center and Server.

Using Git LFS, you can also lock LFS files to stop others from editing them. This can be helpful if you work with large binary assets that can't be merged like images, videos, or 3D models.



When files are locked, an icon appears next to them. Hover over it to see who locked the file and when they did this.

To work with Git LFS files you need to know how to:

1. [Install the Git LFS client](#)
2. [Track Git LFS files and make them lockable](#)
3. [Lock and unlock Git LFS files](#)
4. [Unlock Git LFS files locked by other people](#)

i By default, Git LFS is disabled in Bitbucket for each repository and can only be turned on by repository admins. For more information, see [Git Large File Storage](#).

Installing the Git LFS client

To use Git LFS you first need to install the client on your local machine.

To install the Git LFS client:

1. Check that you have Git 1.8.2 or newer by running:

```
git --version
```

2. [Download](#) and run the Git LFS installer. If you use Linux or Mac OS X you can also use a package manager to install `git-lfs`.

3. Run this command in a terminal or command prompt:

```
git lfs install
```

LFS filters will be added to the `.gitconfig` file in your home directory and will be available for all your repositories. You only need to do this once.

Tracking Git LFS files and making them lockable

On this page:

- [Installing the Git LFS client](#)
- [Tracking Git LFS files and making them lockable](#)
- [Locking and unlocking Git LFS files](#)
- [Unlocking files locked by other people](#)
- [More information](#)

Related pages:

- [Git Large File Storage](#)
- [Getting Git Right](#)

Once you've installed the Git LFS client, you need to register the folder paths or file types you want it to handle using the `track` command. If you want to be able to lock these files, you must also include the `--lockable` flag. This needs to be done for each repository.

To track a folder path or file type:

1. Change to the repository directory.
2. Run this command. Note: you can remove the `--lockable` flag if you don't need it.

```
git lfs track '<pattern>' --lockable
```

Here `<pattern>` can be used to match:

- A folder path, or paths, such as `'path/to/some/folders/*'`.
- File names, or file types, such as `'*.bin,*.psd'` or `'*..*'`.
- Note: when using wildcards you need to include quotes to stop your shell from expanding them.

The pattern will then be added to the repository's `.gitattributes` file.

3. Commit and push the changes to `.gitattributes`:

```
git add .gitattributes
git commit -m "add Git LFS to the repo"
git push origin master
```

You should push the changes whenever you change what files are being tracked so everyone who clones the repo has the tracking patterns.

Now, when you add file names that match the pattern, they will automatically be handled by LFS. You can add multiple patterns this way, and you can see the patterns being tracked using the command `git lfs track`.

Locking and unlocking Git LFS files

Once a Git LFS file has been registered as lockable, it can be locked to stop others from editing it while you're working on it. This can be helpful if you work with large binary assets that can't be merged.

File locking follows a few rules:

- Each file can only be locked by one person at a time.
- Locked files can only be unlocked by the person who locked them (see below for how to force unlock files).
- If your push contains locked files that you didn't lock it will be rejected.
- If your merge contains locked files that you didn't lock it will be blocked.

To lock a Git LFS file:

Change to the repository directory and run the command:

```
git lfs lock <filename>
```

You can also lock a Git LFS file directly in source view by clicking the **Lock** button. A lock icon will then be displayed next to the file name.

To unlock a Git LFS file that you locked:

Change to the repository directory and run the command:

```
git lfs unlock <filename>
```

You can also unlock a Git LFS file directly in source view by clicking the **Unlock** button. The lock icon displayed next to the file is then removed to indicate an unlocked file.

Unlocking files locked by other people

Locked files in Bitbucket can only be unlocked by the person who locked them, so at times you may need to find out who locked a file so you can ask them for help.

When you try to lock, unlock, push, or merge a locked file that has been locked by another person, the error message will include the username and email address of the person who locked it so you can contact them. You can also run a command to see all locked files in a repository:

To see all locked files:

Change to the repository directory and run the command:

```
git lfs locks
```

There may also be times when you can't ask for a file to be unlocked. For example, the person who locked it may have left your organization. If you find yourself in a situation like this, you can use the force flag to unlock the file.

To force unlock a file:

Change to the repository directory and run the command:

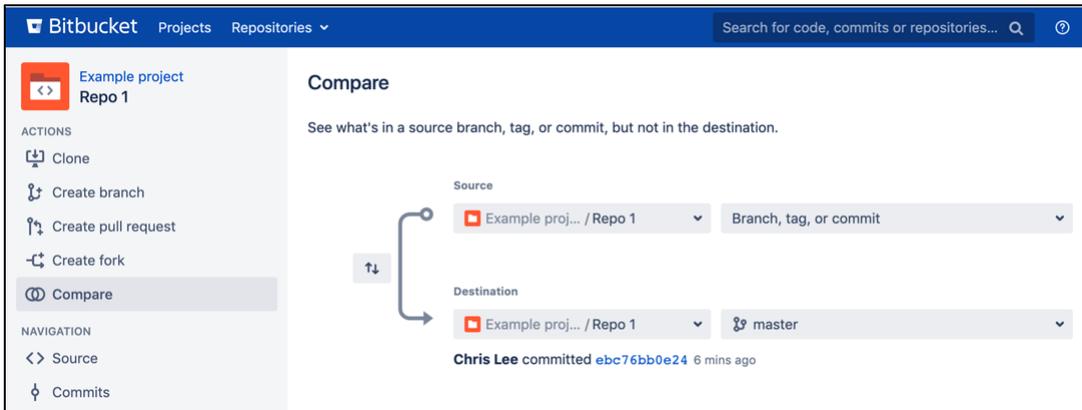
```
git lfs unlock <filename> --force
```

More information

Learn more about using Git LFS at our website, [Getting Git Right](#).

Compare branches, tags, and commits

You can quickly find revision changes using the **Compare** page.



Comparing revisions

To compare revisions in Bitbucket Data Center and Server:

1. From the sidebar, click **Compare**.
2. In the **Compare** page, from both the **Source** and **Destination** dropdown, select any combination of branches, tags, or commits.
3. Once selections are made, the comparison results display in a diff and a commits list tab.

i Bitbucket uses `git diff ...`

A three-dot diff is a comparison between the commit where the feature branch was last synched with the destination branch and the most recent version of the feature branch.

A two-dot diff is the direct comparison of two committish references such as SHAs.

Administer Bitbucket Data Center and Server

This section includes guidance on administration actions that can be performed for Bitbucket.

- [Users and groups](#)
- [External user directories](#)
- [Global permissions](#)
- [Setting up your mail server](#)
- [Integrate with Atlassian applications](#)
- [Connect Bitbucket to an external database](#)
- [Migrating Bitbucket Server to another server](#)
- [Run Bitbucket in AWS](#)
- [Specify the Bitbucket base URL](#)
- [Configuring the application navigator](#)
- [Managing apps](#)
- [View and configure the audit log](#)
- [Update your license key](#)
- [Configuration properties](#)
- [Change Bitbucket's context path](#)
- [Data recovery and backups](#)
- [Disable HTTP\(S\) access to Git repositories](#)
- [Smart Mirroring](#)
- [Export and import projects and repositories](#)
- [Git Large File Storage](#)
- [Git Virtual File System \(GVFS\)](#)
- [Enable SSH access to Git repositories](#)
- [Use diff transcoding](#)
- [Change the port Bitbucket listens on](#)
- [Lockout recovery process](#)
- [Proxy and secure Bitbucket](#)
- [High availability for Bitbucket](#)
- [Diagnostics for third-party apps](#)
- [Enabling JMX counters for performance monitoring](#)
- [Enable debug logging](#)
- [Scaling Bitbucket Server](#)
- [Add a shortcut link to a repository](#)
- [Administer code search](#)
- [Adding additional storage for your repository data](#)
- [Add a system-wide announcement banner](#)
- [Configuring Project links across Applications](#)
- [Improving instance stability with rate limiting](#)
- [Use a CDN with Atlassian Data Center applications](#)
- [Managing personal access tokens](#)
- [Connecting to a 3rd party application using Application Links](#)
- [Setting a system-wide default branch name](#)
- [Automatically decline inactive pull requests](#)
- [Encrypt database password](#)
- [Advanced repository management](#)
- [Xcode for Bitbucket Server](#)
- [Data pipeline](#)

Users and groups

Bitbucket Data Center and Server comes with an internal user directory already built-in that is enabled by default at installation. When you create the first administrator during the setup procedure, that administrator's username and other details are stored in the internal directory.

Bitbucket Admins and Sys Admins can manage users and groups in Bitbucket as described on this page. You can also set up Bitbucket to use [external user directories](#).

Note that:

- Even after users have been added to the user directory, they will not be able to log in to Bitbucket until they have been given [global access permissions](#).
- Permissions can also be applied separately at the level of [projects](#), [repositories](#) and [branches](#).

On this page

- [Creating a user](#)
- [Creating a group](#)
- [Adding users to groups](#)
 - [From the user account page](#)
 - [From the group page](#)
- [Changing usernames](#)
- [Deleting users and groups](#)
 - [Notes](#)
 - [Limitations](#)
- [Deleting a user versus anonymizing a user](#)
- [Anonymizing a user after deletion](#)
- [To anonymize a deleted user in Bitbucket](#)

Related pages:

- [Get started with Git](#)
- [External user directories](#)

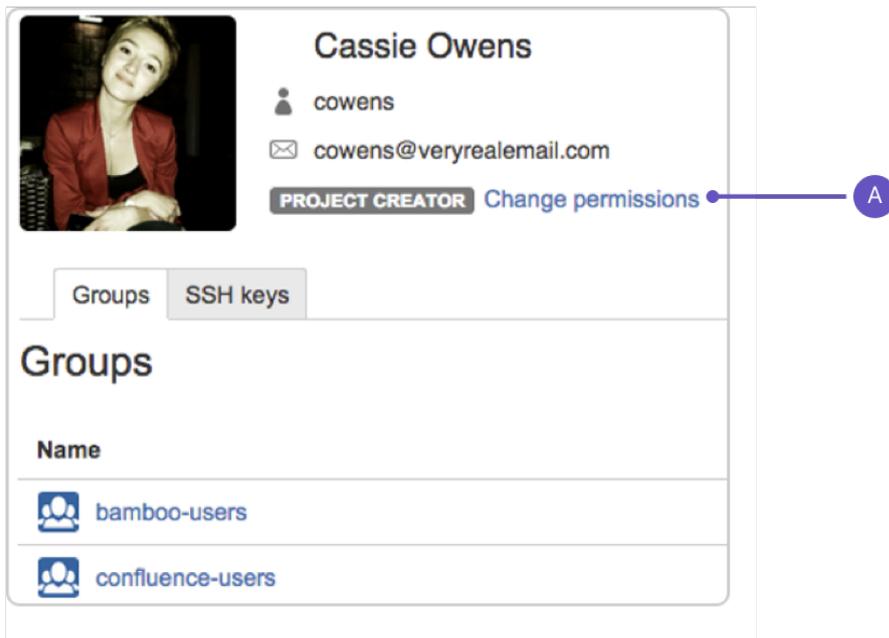


Managing 500+ users across Atlassian products? Find out how easy, scalable and effective it can be with Crowd! See [centralized user management](#).

Creating a user

To create a user:

1. In the administration area, click **Users** (under 'Accounts') and then **Create user** (on the 'Users' screen).
2. Complete the form. You can either set the user's password now, or have Bitbucket email the user with a link that they can use to set the password themselves:
3. Once you've created the user, click **Change permissions** to set up their access permissions. Note that a user doesn't have access to Bitbucket until global access permissions have been set.
 - a. **Set up user permissions**



See [Global permissions](#) for more information.

Creating a group

To create a group, from the administration area:

1. Click **Groups** (under 'Accounts') and then **Create group**.
2. Enter the name for the new group, and click **Create group** (again):

3. Now you can add users to your new group (see the next section).

Adding users to groups

You can add users to groups in two ways:

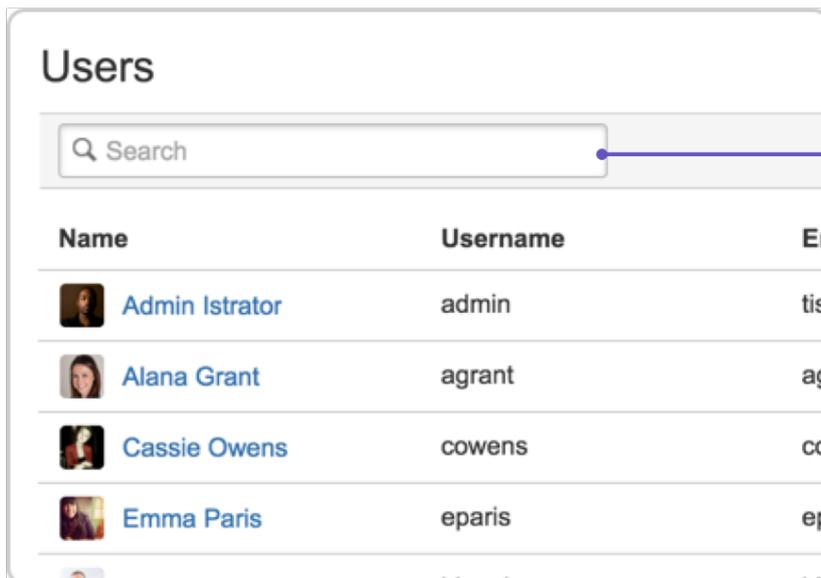
- add a particular user to multiple groups, [from the user's account page](#) in the admin area.
- add multiple users to a particular group, [from the group's page](#).

From the user account page

To add a user to a group from the user's account page,

1. Click **Users** in the Administration section, and then use the filter to find the user:

- a. **User search:** Filter users by name or email as you type.



2. On the account page for the user, use the filter to find a group to which you want to add the user.
3. Click **Add** for each group in turn.

From the group page

To add a user to a group from the group's page,

1. Click **Groups** (under "Accounts") in the administration area, and use the filter to find the group.
2. On the page for the group, use the filter to find a user to add to the group.
3. Click **Add** for each user you select, to make them a member of the group.

Changing usernames

You can change the username for a user account that is hosted in Bitbucket's internal user directory.

To change a user's username:

1. Go to **Users** in the Administration section, use the filter to find the user.
2. On the account page for the user, click **Rename**.

Deleting users and groups

You can delete a user or group from Bitbucket's internal user directory, or the external directory from which Bitbucket sources users, such as an LDAP, Crowd or Jira Software.

When a user or group is deleted from such a directory, Bitbucket checks to see if that user still exists in another directory:

- If the user or group *does exist* in another directory, Bitbucket assumes the administrator intended to *migrate* the user or group between directories and we leave their data intact.
- If the user or group *does not exist* in another directory, Bitbucket assumes the intent was to permanently delete them, and we delete the user's permissions, SSH keys and 'remember me' tokens.



When deleting users

In the case of users from an external directory (e.g. JIRA or LDAP) and internal users (from the internal directory), users or groups are preserved for **seven (7) days**.

This includes:

- SSH keys
- GPG Keys
- Access tokens
- All user related data stored by apps.

Notes

- If an entire directory is deleted, Bitbucket will preserve users and groups for **seven (7) days** before deleting.
- Content which might be of historical interest (comments, pull requests, etc.) is not deleted when a user or group is. Only authentication, authorization and data which serves no purpose to a user who can no longer log in is removed.
- In some situations, reordering the directories will change the directory that the current user comes from, if a user with the same username happens to exist in both. This behavior can be used in some cases to create a copy of the existing configuration, move it to the top, then remove the old one. Note, however, that duplicate usernames are not a supported configuration.
- You can enable or disable a directory at any time. If you disable a directory, your configuration details will remain but Bitbucket will not recognize the users and groups in that directory.

Limitations

- You cannot edit, disable or delete the directory that your own user account belongs to. This prevents administrators from locking themselves out of Bitbucket, and applies to internal as well as external directories.
- You cannot remove the internal directory. This limitation aligns with the recommendation that you always keep an administrator or sysadmin account active in the Bitbucket internal directory, so that you can troubleshoot problems with your user directories.
- You have to disable a directory before you can remove it. Removing a directory will remove the details from the database.

Deleting a user versus anonymizing a user

When someone leaves your organization, or no longer needs to use Bitbucket, you can **delete** their user account. Then if required, you can **anonymize** their username within Bitbucket.

Anonymizing a user means that any remaining personally identifiable information in Bitbucket after the user is deleted, is updated to be permanently non-attributable to that specific user.

Anonymizing a user after deletion

When you **anonymize** a username:

- The username is replaced with a non-attributable alias throughout Bitbucket.
- User mentions are replaced with a non-attributable alias throughout Bitbucket.
- If the user had a personal project, the personal project name and key is updated to a non-attributable alias.
- User cleanup for deletion happens immediately, if it hasn't already taken place (eg. deleting avatars, SSH keys, permissions).

The following data **will remain** after a username is **anonymized**:

- User content (such as comments and pull requests).
- User data in Git history.
- User data in third-party plug-ins may not be anonymized.

To anonymize a deleted user in Bitbucket



Be sure the user is deleted from Bitbucket prior to anonymization, including from any external directories that the user is a member of.

If the user is not deleted prior to anonymization, the anonymization will fail.

1. From the **Create user** menu on the user list page, select **Anonymize user** from the dropdown.
2. Enter the exact username to anonymize in the username field, and click continue.
3. Read through the details of the anonymization process and tick the box to confirm you wish to anonymize this user.
4. Click **Anonymize**.

Note that once the anonymize button is clicked, the process will continue even if the browser window is closed.

External user directories

You can connect Bitbucket Data Center and Serverto external user directories. This allows you to use existing users and groups stored in an enterprise directory, and to manage those users and groups in one place.

User management functions include:

- **Authentication:**determining which user identity is sending a request to Bitbucket.
- **Authorization:** determining the access privileges for an authenticated user.
- **User management:**maintaining profile information in user's accounts.
- **Group membership:**storing and retrieving groups, and group membership.

It is important to understand that these are separate components of a user management system. You could use an external directory for any or all of the above tasks.

There are several approaches to consider when using external user directories with Bitbucket, described briefly below:

- [LDAP](#)
- [Jira applications](#)
- [Crowd](#)
- [Multiple directories](#)

On this page

- [LDAP](#)
- [Jira applications](#)
- [Crowd](#)
- [Multiple directories](#)

Related pages

- [Connect to an LDAP directory](#)
- [Delegate authentication to an LDAP directory](#)
- [Connect Bitbucket to Crowd](#)
- [Delegate user management to Jira](#)
- [Users and groups](#)
- [External directory lockout recovery](#)



- Bitbucket provides a "read-only" connection to external directories for user management. This means that users and groups, fetched from *any external directory*, can only be modified or updated in the external directory itself, rather than in Bitbucket.
- Connecting Atlassian Bitbucket to your external directory is not sufficient to allow your users to log in. You must explicitly grant them access to Bitbucket in the [global permission screen](#).
- We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket will display a warning banner. See [this FAQ](#).
- Bitbucket comes with an internal user directory, already built-in, that is enabled by default at installation. When you create the first administrator during the setup procedure, that administrator's username and other details are stored in the internal directory.
- See also this [information about deleting users and groups](#) in Bitbucket.

LDAP

You should consider connecting to an LDAP directory server if your users and groups are stored in an enterprise directory.

There are two common ways of using an external LDAP directory with Bitbucket:

- For full user and group management, including for user authentication see [Connect to an LDAP directory](#) for instructions.
- For delegated user authentication only, while using Bitbucket's internal directory for user and group management see [Delegate authentication to an LDAP directory](#) for instructions.

Bitbucket is able to connect to the following LDAP directory servers:

- Microsoft Active Directory
- Apache Directory Server (ApacheDS) 1.0.x and 1.5.x
- Apple Open Directory (Read-Only)
- Fedora Directory Server (Read-Only Posix Schema)
- Novell eDirectory Server
- OpenDS
- OpenLDAP
- OpenLDAP (Read-Only Posix Schema)
- Generic Posix/RFC2307 Directory (Read-Only)
- Sun Directory Server Enterprise Edition (DSEE)
- Any generic LDAP directory server

Jira applications

You can delegate Bitbucket user and group management, as well as user authentication, to a [Jira application](#). This is a good option if you already use a Jira application in your organization. Note that Bitbucket can only connect to a Jira application server running Jira 4.3 or later.

You should consider using [Atlassian Crowd](#) for more complex configurations with a large number of users.

See [Connecting Bitbucket Server to Jira for user management](#) for configuration instructions.

Crowd

You can connect Bitbucket to [Atlassian Crowd](#) for user and group management, as well as for user authentication.

Crowd is an application security framework that handles authentication and authorization for your web-based applications. With Crowd you can integrate multiple web applications with multiple user directories, with support for single sign-on (SSO) and centralized identity management. See the [Crowd Administration Guide](#).

You should consider connecting to Crowd if you want to use Crowd to manage existing users and groups in multiple directory types, or if you have users of other web-based applications.

See [Connect Bitbucket to Crowd](#) for configuration instructions.

Multiple directories

When Bitbucket is connected directly to multiple user directories, where duplicate user names and group names are used across those directories, the effective group memberships that Bitbucket uses for authorization can be determined using either of these two schemes:

- 'aggregating membership'
- 'non-aggregating membership'.

See [Effective memberships with multiple directories](#) for more information about these two schemes.

Note that:

- Aggregating membership is used by default for new installations of Bitbucket.
- Authentication, for when Bitbucket is connected to multiple directories, only depends on the mapped groups in those directories the aggregation scheme is not involved at all.
- For inactive users, Bitbucket only checks if the user is active in the first (highest priority) directory in which they are found for the purpose of determining authentication. Whether a user is active or inactive does not affect how their memberships are determined.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.

- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed from the group in all directories the user exists in.

A Bitbucket admin can change the membership scheme used by Bitbucket using the following commands:

- To change to *aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d '{"membershipAggregationEnabled":true}' -u <username>:<password> <base-url>/rest/crowd/latest/application
```

- To change to *non-aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d '{"membershipAggregationEnabled":false}' -u <username>:<password> <base-url>/rest/crowd/latest/application
```

Note that these operations are different from how you make these changes in Crowd. Note also that changing the aggregation scheme can affect the authorization permissions for your Bitbucket users, and how directory update operations are performed.

Connect to an LDAP directory

You can connect Bitbucket Data Center and Serverto an existing LDAP user directory, so that your existing users and groups in an enterprise directory can be used in Bitbucket. The LDAP directory is used for both user authentication and account management.

Bitbucket is able to connect to the following LDAP directory servers:

- Microsoft Active Directory
- Apache Directory Server (ApacheDS) 1.0.x and 1.5.x
- Apple Open Directory (Read-Only)
- Fedora Directory Server (Read-Only Posix Schema)
- Novell eDirectory Server
- OpenDS
- OpenLDAP
- OpenLDAP (Read-Only Posix Schema)
- Generic Posix/RFC2307 Directory (Read-Only)
- Sun Directory Server Enterprise Edition (DSEE)
- Any generic LDAP directory server

On this page

- [License considerations](#)
- [Synchronization when Bitbucket is first connected to the LDAP directory](#)
- [Authentication when a user attempts to log in](#)
- [Connecting Bitbucket](#)
- [Server settings](#)
- [LDAP schema](#)
- [LDAP permission](#)
- [Advanced settings](#)
- [User schema settings](#)
- [Group schema settings](#)
- [Membership schema settings](#)

Related pages

- [Deleting Users and Groups](#)



Connecting Atlassian Bitbucket to your external directory is not sufficient to allow your users to log in. You must explicitly grant them access to Bitbucket in the [global permission screen](#).

We recommend that you use groups instead of individual accounts when granting permissions.

License considerations

When connecting Bitbucket to an external directory, be careful not to allow access by more users than your Bitbucket license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket will display a warning banner. See [this FAQ](#).

Synchronization when Bitbucket is first connected to the LDAP directory

When you first connect Bitbucket to an existing LDAP directory, the Bitbucket internal directory is synchronized with the LDAP directory. User information, including groups and group memberships, is copied across to the Bitbucket directory.

When we performed internal testing of synchronization with an Active Directory server on our local network with 10 000 users, 1000 groups and 200 000 memberships, we found that the initial synchronization took about 5 minutes. Subsequent synchronizations with 100 modifications on the AD server took a couple of seconds to complete. See the [option](#) below.

Note that when Bitbucket is connected to an LDAP directory, you cannot update user details in Bitbucket. Updates must be done directly on the LDAP directory, perhaps using a LDAP browser tool such as [Apache Directory Studio](#).

Option - Use LDAP filters to restrict the number of users and groups that are synchronized

You can use LDAP filters to restrict the users and groups that are synchronized with the Bitbucket internal directory. You may wish to do this in order to limit the users or groups that can access Bitbucket, or if you are concerned that synchronization performance may be poor.

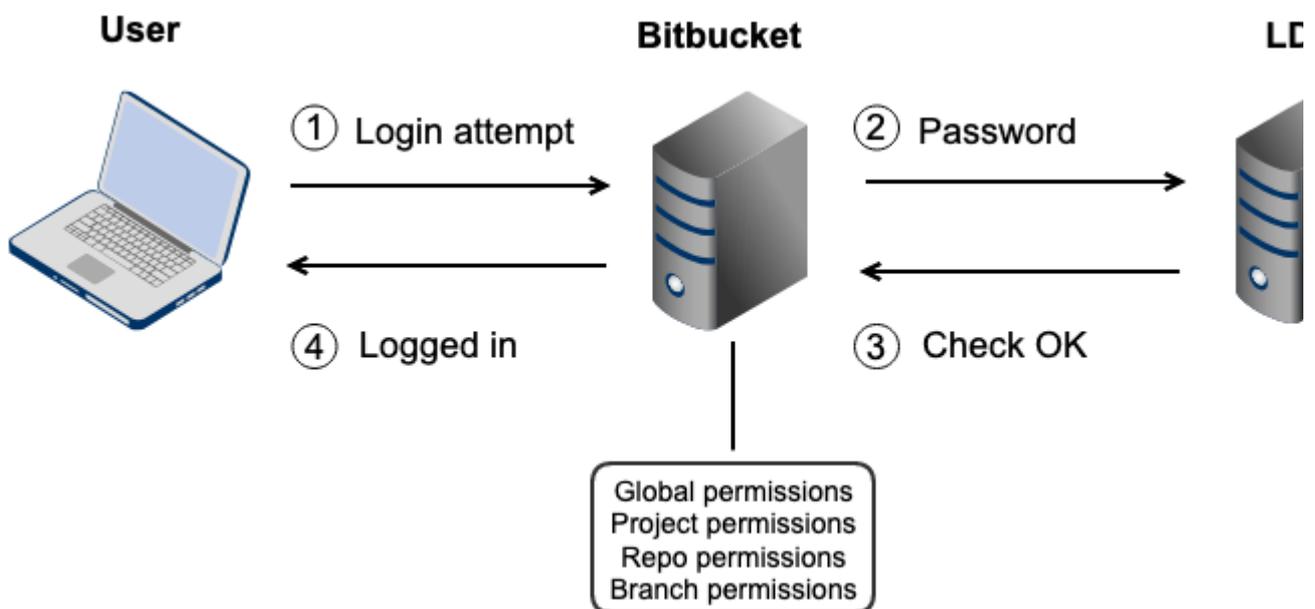
For example, to limit synchronization to just the groups named "bitbucket_user" or "red_team", enter the following into the **Group Object Filter** field (see [Group Schema Settings](#) below):

```
(&(objectClass=group)(|(cn=bitbucket_user)(cn=red_team)))
```

For further discussion about filters, with examples, please see [How to write LDAP search filters](#). Note that you need to know the names for the various containers, attributes and object classes in your particular directory tree, rather than simply copying these examples. You can discover these container names by using a tool such as [Apache Directory Studio](#).

Authentication when a user attempts to log in

When a user attempts to log in to Bitbucket, once synchronization has completed, Bitbucket confirms that the user exists in its internal directory and then passes the user's password to the LDAP directory for confirmation. If the password matches that stored for the user, LDAP passes a confirmation back to Bitbucket, and Bitbucket logs in the user. During the user's session, all authorizations (i.e. access to Bitbucket resources such as repositories, pull requests and administration screens) are handled by Bitbucket, based on permissions maintained by Bitbucket in its internal directory.



Connecting Bitbucket

To connect Bitbucket to an LDAP directory:

1. Log in as a user with 'Admin' permission.
2. In the Bitbucket administration area, click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select either **Microsoft Active Directory** or **LDAP** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows next to each directory on the 'User Directories' screen. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

| Setting | Description |
|----------------|--|
| Name | <p>Enter a meaningful name to help you identify the LDAP directory server. Examples:</p> <ul style="list-style-type: none"> • Example Company Staff Directory • Example Company Corporate LDAP |
| Directory Type | <p>Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for many of the options on the rest of screen. Examples:</p> <ul style="list-style-type: none"> • Microsoft Active Directory • OpenDS • And more. |
| Hostname | <p>The host name of your directory server. Examples:</p> <ul style="list-style-type: none"> • ad.example.com • ldap.example.com • opensds.example.com |
| Port | <p>The port on which your directory server is listening. Examples:</p> <ul style="list-style-type: none"> • 389 • 10389 • 636 (for example, for SSL) |
| Use SSL | <p>Check this if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.</p> |
| Username | <p>The distinguished name of the user that the application will use when connecting to the directory server. Examples:</p> <ul style="list-style-type: none"> • cn=admin, cn=users, dc=ad, dc=example, dc=com • cn=user, dc=domain, dc=name • user@domain.name <div style="border: 1px solid red; padding: 10px; margin-top: 10px;"> <p> By default, all users can read the uSNChanged attribute; however, only administrators or users with relevant permissions can access the Deleted Objects container. The specific privileges required by the user to connect to LDAP are "Bind" and "Read" (user info, group info, group membership, update sequence number, deleted objects), which the user can obtain by being a member of the Active Directory's built-in administrators group.</p> <p>Note that the incremental sync will fail silently if the Active Directory is accessed by a user without these privileges. This has been reported as CWD-3093.</p> </div> |
| Password | <p>The password of the user specified above.</p> <p>Note: Connecting to an LDAP server requires that this application log in to the server with the username and password configured here. As a result, this password cannot be one-way hashed - it must be recoverable in the context of this application. The password is currently stored in the database in plain text without obfuscation. To guarantee its security, you need to ensure that other processes do not have OS-level read permissions for this application's database or configuration files.</p> |

LDAP schema

| Setting | Description |
|---------------------|--|
| Base DN | <p>The root distinguished name (DN) to use when running queries against the directory server. Examples:</p> <ul style="list-style-type: none"> o=example,c=com cn=users,dc=ad,dc=example,dc=com For Microsoft Active Directory, specify the base DN in the following format: dc=domain1,dc=local. You will need to replace the domain1 and local for your specific configuration. Microsoft Server provides a tool called <code>ldp.exe</code> which is useful for finding out and configuring the the LDAP structure of your server. |
| Additional User DN | <p>This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example:</p> <ul style="list-style-type: none"> ou=Users |
| Additional Group DN | <p>This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example:</p> <ul style="list-style-type: none"> ou=Groups |

 If no value is supplied for **Additional User DN** or **Additional Group DN** this will cause the subtree search to start from the base DN and, in case of huge directory structure, could cause performance issues for login and operations that rely on login to be performed.

LDAP permission

| Setting | Description |
|------------------------------|---|
| Read Only | LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens. |
| Read Only, with Local Groups | LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens. However, you can add groups to the internal directory and add LDAP users to those groups. |

Advanced settings

 The **Manage User Status Locally** option, described below, will not work within Bitbucket. Do not enable this option.

 **BSERV-5129** - Disable "Manage User Status Locally" in Bitbucket Server GATHERING IMPACT

| Setting | Description |
|----------------------|--|
| Enable Nested Groups | Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called <i>nested groups</i> . Nested groups simplify permissions by allowing sub-groups to inherit permissions from a parent group. |

| | |
|------------------------------------|--|
| Manage User Status Locally | If true, you can activate and deactivate users in Crowd independent of their status in the directory server. |
| Filter out expired users | If true, user accounts marked as expired in ActiveDirectory will be automatically removed. For cached directories, the removal of a user will occur during the first synchronization after the account's expiration date.

Note: This is available in Embedded Crowd 2.0.0 and above, but not available in the 2.0.0 m04 release. |
| Use Paged Results | Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results. |
| Follow Referrals | Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referral</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error. |
| Naive DN Matching | If your directory server will always return a consistent string representation of a DN, you can enable naive DN matching. Using naive DN matching will result in a significant performance improvement, so we recommend enabling it where possible.

This setting determines how your application will compare DN's to determine if they are equal. <ul style="list-style-type: none"> • If this checkbox is selected, the application will do a direct, case-insensitive, string comparison. This is the default and recommended setting for Active Directory, because Active Directory guarantees the format of DN's. • If this checkbox is not selected, the application will parse the DN and then check the parsed version. |
| Enable Incremental Synchronization | Enable incremental synchronization if you only want changes since the last synchronization to be queried when synchronizing a directory.

⚠ Please be aware that when using this option, the user account configured for synchronization must have read access to: <ul style="list-style-type: none"> • The <code>uSNChanged</code> attribute of all users and groups in the directory that need to be synchronized. • The objects and attributes in the Active Directory deleted objects container. <p>If at least one of these conditions is not met, you may end up with users who are added to (or deleted from) the Active Directory not being respectively added (or deleted) in the application.</p> <p>This setting is only available if the directory type is set to "Microsoft Active Directory".</p> |
| Synchronization Interval (minutes) | Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes. |
| Read Timeout (seconds) | The time, in seconds, to wait for a response to be received. If there is no response within the specified time period, the read attempt will be aborted. A value of 0 (zero) means there is no limit. The default value is 120 seconds. |
| Search Timeout (seconds) | The time, in seconds, to wait for a response from a search operation. A value of 0 (zero) means there is no limit. The default value is 60 seconds. |

| | |
|------------------------------|---|
| Connection Timeout (seconds) | <p>This setting affects two actions. The default value is 0.</p> <ul style="list-style-type: none"> The time to wait when getting a connection from the connection pool. A value of 0 (zero) means there is no limit, so wait indefinitely. The time, in seconds, to wait when opening new server connections. A value of 0 (zero) means that the TCP network timeout will be used, which may be several minutes. |
|------------------------------|---|

User schema settings

| Setting | Description |
|-----------------------------|--|
| User Object Class | <p>This is the name of the class used for the LDAP user object. Example:</p> <ul style="list-style-type: none"> <code>user</code> |
| User Object Filter | <p>The filter to use when searching user objects. Example:</p> <ul style="list-style-type: none"> <code>(&(objectCategory=Person)(sAMAccountName=*))</code> <p>More examples can be found in our knowledge base. See How to write LDAP search filters.</p> |
| User Name Attribute | <p>The attribute field to use when loading the username. Examples:</p> <ul style="list-style-type: none"> <code>cn</code> <code>sAMAccountName</code> <p>NB: In Active Directory, the 'sAMAccountName' is the 'User Logon Name (pre-Windows 2000)' field. The User Logon Name field is referenced by 'cn'.</p> |
| User Name RDN Attribute | <p>The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example:</p> <ul style="list-style-type: none"> <code>cn</code> |
| User First Name Attribute | <p>The attribute field to use when loading the user's first name. Example:</p> <ul style="list-style-type: none"> <code>givenName</code> |
| User Last Name Attribute | <p>The attribute field to use when loading the user's last name. Example:</p> <ul style="list-style-type: none"> <code>sn</code> |
| User Display Name Attribute | <p>The attribute field to use when loading the user's full name. Example:</p> <ul style="list-style-type: none"> <code>displayName</code> |
| User Email Attribute | <p>The attribute field to use when loading the user's email address. Example:</p> <ul style="list-style-type: none"> <code>mail</code> |

| | |
|--------------------------|--|
| User Password Attribute | The attribute field to use when loading a user's password. Example: <ul style="list-style-type: none"> • unicodePwd |
| User Unique ID Attribute | The attribute used as a unique immutable identifier for user objects. This is used to track username changes and is optional. If this attribute is not set (or is set to an invalid value), user renames will not be detected they will be interpreted as a user deletion then a new user addition.

This should normally point to a UUID value. Standards-compliant LDAP servers will implement this as 'entryUUID' according to RFC 4530 . This setting exists because it is known under different names on some servers, e.g. 'objectGUID' in Microsoft Active Directory. |

Group schema settings

| Setting | Description |
|-----------------------------|---|
| Group Object Class | This is the name of the class used for the LDAP group object. Examples: <ul style="list-style-type: none"> • groupOfUniqueNames • group |
| Group Object Filter | The filter to use when searching group objects. Example: <ul style="list-style-type: none"> • (&(objectClass=group)(cn=*)) |
| Group Name Attribute | The attribute field to use when loading the group's name. Example: <ul style="list-style-type: none"> • cn |
| Group Description Attribute | The attribute field to use when loading the group's description. Example: <ul style="list-style-type: none"> • description |

Membership schema settings

| Setting | Description |
|---------------------------|--|
| Group Members Attribute | The attribute field to use when loading the group's members. Example: <ul style="list-style-type: none"> • member |
| User Membership Attribute | The attribute field to use when loading the user's groups. Example: <ul style="list-style-type: none"> • memberOF |

| | |
|--|--|
| <p>Use the User Membership Attribute, when finding the user's group membership</p> | <p>Check this if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.)</p> <ul style="list-style-type: none">• If this checkbox is selected, your application will use the group membership attribute on the user when retrieving the list of groups to which a given user belongs. This will result in a more efficient retrieval.• If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search.• If the Enable Nested Groups checkbox is selected, your application will ignore the Use the User Membership Attribute option and will use the members attribute on the group for the search. |
| <p>Use the User Membership Attribute, when finding the members of a group</p> | <p>Check this if your directory server supports the user membership attribute on the group. (By default, this is the 'member' attribute.)</p> <ul style="list-style-type: none">• If this checkbox is selected, your application will use the group membership attribute on the user when retrieving the members of a given group. This will result in a more efficient search.• If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search. |

Delegate user management to Jira

 This page does not apply to Jira Software Cloud; you can't use Jira Software Cloud to manage your Bitbucket Data Center and Server users.

You can connect Bitbucket to an existing Atlassian Jira Data Center and Server instance to delegate Bitbucket user and group management, and authentication. Bitbucket provides a "read-only" connection to Jira for user management. This means that users and groups, fetched from Jira, can only be modified or updated in that Jira server, rather than in Bitbucket.

Choose this option, as an alternative to Atlassian Crowd, for simple configurations with a limited number of users. Note that Bitbucket can only connect to an instance running Jira 4.3 or later.

Connecting Bitbucket and Jira is a 3-step process:

1. [Set up Jira to allow connections from Bitbucket](#)
2. [Set up Bitbucket to connect to Jira](#)
3. [Set up Bitbucket users and groups in Jira](#)

Also on this page:

- [Server settings](#)
- [Jira server permissions](#)
- [Advanced settings](#)

 You need to be an administrator in Jira and a system administrator in Bitbucket to perform the following tasks.

 Managing 500+ users across Atlassian products? Find out how easy, scalable, and effective it can be with Crowd! See [centralized user management](#).

1. Setup Jira to allow connections from Bitbucket

1. Log in as a user with the 'Jira Software Administrators' global permission.
2. For Jira 4.3.x, select **Other Application** from the 'Users, Groups & Roles' section of the 'Administration' menu.
For later versions, choose **Administration > Users > Jira User Server**.
3. Click **Add Application**.
4. Enter the **application name** (case-sensitive) and **password** that Bitbucket will use when accessing Jira.
5. Enter the **IP address** of your Bitbucket instance. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
6. Click **Save**.
7. Define the directory order, on the 'User Directories' screen, by clicking the blue up- and down-arrows next to each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

2. Setup Bitbucket to connect to Jira

1. Log in to Bitbucket as a user with 'Admin' permission.
2. In the Bitbucket administration area click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Atlassian Jira**.

4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the 'User Directories' screen, by clicking the arrows for each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

3. Set up Bitbucketusers and groups in Jira

In order to use Bitbucket, users must be a member of the `Bitbucket Server-users` group or have `Bitbucketglobal` permissions. Follow these steps to configure your `Bitbucketgroups` in Jira:

1. Add the `bitbucket-users` and `bitbucket-administrators` groups in Jira.
2. Add your own username as a member of both of the above groups.
3. Choose one of the following methods to give your existing Jira users access to Bitbucket:
 - Option 1: In Jira, find the groups that the relevant users belong to. Add those groups as members of one or both of the above `Bitbucketgroups`.
 - Option 2: Log in to Bitbucketusing your Jira account and go to the administration area. Click **Global permissions** (under 'Accounts'). Assign the appropriate permissions to the relevant Jira groups. See [Global permissions](#).

i Connecting Atlassian Bitbucket to Jira for user management is not sufficient, by itself, to allow your users to log in to Bitbucket. You must also grant them access to Bitbucket by using one of the above 2 options.

We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket will display a warning banner. See [this FAQ](#).

See also [this information about deleting users and groups](#) in Bitbucket.

Server settings

| Setting | Description |
|----------------------|--|
| Name | A meaningful name that will help you to identify this Jira server in the list of directory servers.
Examples: <ul style="list-style-type: none"> • Jira Software • My Company Jira |
| Server URL | The web address of your Jira server. Examples: <ul style="list-style-type: none"> • <code>http://www.example.com:8080</code> • <code>http://jira.example.com</code> |
| Application Name | The name used by your application when accessing the Jira server that acts as user manager. Note that you will also need to define your application to that Jira server, via the Other Applications option in the 'Users, Groups & Roles' section of the 'Administration' menu. |
| Application Password | The password used by your application when accessing the Jira server that acts as user manager. |

Jira server permissions

| Setting | Description |
|-----------|--|
| Read Only | The users, groups and memberships in this directory are retrieved from the Jira server that is acting as user manager. They can only be modified via that JIRA server. |

Advanced settings

| Setting | Description |
|------------------------------------|--|
| Enable Nested Groups | Enable or disable support for nested groups. Before enabling nested groups, please check to see if nested groups are enabled on the JIRA server that is acting as user manager. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups. |
| Enable Incremental Synchronization | Enable or disable incremental synchronization. Only changes since the last synchronization will be retrieved when synchronizing a directory. |
| Synchronization Interval (minutes) | Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes. |

Delegate authentication to an LDAP directory

You can configure Bitbucket Data Center and Server to use an LDAP directory for delegated user authentication while still using Bitbucket for user and group management.

You can either create new user accounts manually in the LDAP directory or use the option to automatically create a user account when the user attempts to log in, as described in the [Copy users on login](#) section below.

See also this [information about deleting users and groups](#) in Bitbucket.

To connect Bitbucket to an LDAP directory for delegated authentication:

1. Log in to Bitbucket as a user with 'Admin' permission.
2. Go to the Bitbucket administration area and click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Internal with LDAP Authentication** as the directory type.
4. Configure the directory settings, as described in the tables below.
5. Save the directory settings.
6. Define the directory order by clicking the arrows for each directory on the 'User Directories' screen. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

 Connecting Atlassian Bitbucket to your external directory is not sufficient to allow your users to log in. You must explicitly grant them access to Bitbucket in the [global permission screen](#).

We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket will display a warning banner. See [this FAQ](#).

On this page:

- [Server settings](#)
- [Manually creating users](#)
- [Copying users on login](#)
- [LDAP schema](#)
- [Advanced settings](#)
- [User schema settings](#)
- [Group schema settings](#)
- [Membership schema settings](#)

Server settings

| Setting | Description |
|---------|---|
| Name | A descriptive name that will help you to identify the directory. Examples: <ul style="list-style-type: none">• Internal directory with LDAP Authentication• Corporate LDAP for Authentication Only |

| | |
|----------------|---|
| Directory Type | Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for some of the options on the rest of screen. Examples: <ul style="list-style-type: none"> • Microsoft Active Directory • OpenDS • And more. |
| Hostname | The host name of your directory server. Examples: <ul style="list-style-type: none"> • <code>ad.example.com</code> • <code>ldap.example.com</code> • <code>opends.example.com</code> |
| Port | The port on which your directory server is listening. Examples: <ul style="list-style-type: none"> • 389 • 10389 • 636 (for example, for SSL) |
| Use SSL | Check this box if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting. |
| Username | The distinguished name of the user that the application will use when connecting to the directory server. Examples: <ul style="list-style-type: none"> • <code>cn=administrator,cn=users,dc=ad,dc=example,dc=com</code> • <code>cn=user,dc=domain,dc=name</code> • <code>user@domain.name</code> |
| Password | The password of the user specified above. |

Manually creating users

Move the delegated authentication directory to the top of the User Directories list and create the user manually (go to **Administration > Users > Create user**). Using this manual method you must currently create a temporary password when creating users. There is an improvement request to address this:

 **BSERV-3424** - Disable "Change password" field from admin and user page when delegated authentication is used CLOSED

 If you intend to *change* the authentication directory of your users from Bitbucket Server Internal Directory to Delegated LDAP Authentication you must select the option to "Copy User on Login" since you can't create a new user that has the same username as another user in another directory.

Copying users on login

The settings described in the table below relate to when a user attempts to authenticate with Bitbucket. This authentication attempt can occur either:

- when using the Bitbucketlogin screen.
- when issuing a Git clone or push command at the command line, for a repository managed by Bitbucket.

| Setting | Description |
|---------|-------------|
|---------|-------------|

| | |
|---------------------------------|---|
| Copy User on Login | <p>This option affects what will happen when a user attempts to log in. If this box is checked, the user will be created automatically in the internal directory that is using LDAP for authentication when the user first logs in and their details will be synchronized on each subsequent log in. If this box is not checked, the user's login will fail if the user wasn't already manually created in the directory.</p> <p>If you check this box the following additional fields will appear on the screen, which are described in more detail below:</p> <ul style="list-style-type: none"> • Default Group Memberships • Synchronize Group Memberships • User Schema Settings (described in a separate section below) |
| Update User attributes on Login | <p>Whenever your users authenticate to the application, their attributes will be automatically updated from the LDAP server into the application. After you select this option, you won't be able to modify or delete your users directly in the application.</p> <ul style="list-style-type: none"> • If you need to modify a user, do it on the LDAP server; it will be updated in the application after authenticating. • If you need to delete a user, do it on the LDAP server, but also in the application. If you delete the user only on the LDAP server, it will be rejected from logging in to the application, but it won't be set as inactive, which will affect your license. You'll need to disable the Update User attributes on Login option to delete the user, and then enable it again. |
| Default Group Memberships | <p>This field appears if you check the Copy User on Login box. If you would like users to be automatically added to a group or groups, enter the group name(s) here. To specify more than one group, separate the group names with commas. Each time a user logs in, their group memberships will be checked. If the user does not belong to the specified group(s), their username will be added to the group(s). If a group does not yet exist, it will be added to the internal directory that is using LDAP for authentication.</p> <p>Please note that there is no validation of the group names. If you mis-type the group name, authorization failures will result users will not be able to access the applications or functionality based on the intended group name.</p> <p>Examples:</p> <ul style="list-style-type: none"> • <code>confluence-users</code> • <code>bamboo-users, jira-administrators, jira-core-users</code> |
| Synchronize Group Memberships | <p>This field appears if you select the Copy User on Login checkbox. If this box is checked, group memberships specified on your LDAP server will be synchronized with the internal directory each time the user logs in.</p> <p>If you check this box the following additional fields will appear on the screen, both described in more detail below:</p> <ul style="list-style-type: none"> • Group Schema Settings (described in a separate section below) • Membership Schema Settings (described in a separate section below) |

LDAP schema

| Setting | Description |
|---------|-------------|
|---------|-------------|

| | |
|---------------------|---|
| Base DN | <p>The root distinguished name (DN) to use when running queries against the directory server. Examples:</p> <ul style="list-style-type: none"> • <code>o=example,c=com</code> • <code>cn=users,dc=ad,dc=example,dc=com</code> • For Microsoft Active Directory, specify the base DN in the following format: <code>dc=domain1,dc=local</code>. You will need to replace the <code>domain1</code> and <code>local</code> for your specific configuration. Microsoft Server provides a tool called <code>ldp.exe</code> which is useful for finding out and configuring the the LDAP structure of your server. |
| User Name Attribute | <p>The attribute field to use when loading the username. Examples:</p> <ul style="list-style-type: none"> • <code>cn</code> • <code>sAMAccountName</code> |

Advanced settings

| Setting | Description |
|----------------------|--|
| Enable Nested Groups | Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called <i>nested groups</i> . Nested groups simplify permissions by allowing sub-groups to inherit permissions from a parent group. |
| Use Paged Results | Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results. |
| Follow Referrals | Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referral</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error. |

User schema settings

Note: this section is only visible when **Copy User on Login** is enabled.

| Setting | Description |
|--------------------|--|
| Additional User DN | <p>This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example:</p> <ul style="list-style-type: none"> • <code>ou=Users</code> |
| User Object Class | <p>This is the name of the class used for the LDAP user object. Example:</p> <ul style="list-style-type: none"> • <code>user</code> |
| User Object Filter | <p>The filter to use when searching user objects. Example:</p> <ul style="list-style-type: none"> • <code>(&(objectCategory=Person)(sAMAccountName=*))</code> |

| | |
|-----------------------------|---|
| User Name RDN Attribute | The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: <ul style="list-style-type: none"> • <code>cn</code> |
| User First Name Attribute | The attribute field to use when loading the user's first name. Example: <ul style="list-style-type: none"> • <code>givenName</code> |
| User Last Name Attribute | The attribute field to use when loading the user's last name. Example: <ul style="list-style-type: none"> • <code>sn</code> |
| User Display Name Attribute | The attribute field to use when loading the user's full name. Example: <ul style="list-style-type: none"> • <code>displayName</code> |
| User Email Attribute | The attribute field to use when loading the user's email address. Example: <ul style="list-style-type: none"> • <code>mail</code> |

Group schema settings

Note: this section is only visible when both **Copy User on Login** and **Synchronize Group Memberships** are enabled.

| Setting | Description |
|-----------------------------|---|
| Additional Group DN | This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example: <ul style="list-style-type: none"> • <code>ou=Groups</code> |
| Group Object Class | This is the name of the class used for the LDAP group object. Examples: <ul style="list-style-type: none"> • <code>groupOfUniqueNames</code> • <code>group</code> |
| Group Object Filter | The filter to use when searching group objects. Example: <ul style="list-style-type: none"> • <code>(objectCategory=Group)</code> |
| Group Name Attribute | The attribute field to use when loading the group's name. Example: <ul style="list-style-type: none"> • <code>cn</code> |
| Group Description Attribute | The attribute field to use when loading the group's description. Example: <ul style="list-style-type: none"> • <code>description</code> |

Membership schema settings

Note: this section is only visible when both **Copy User on Login** and **Synchronize Group Memberships** are enabled.

| Setting | Description |
|---|---|
| Group Members Attribute | The attribute field to use when loading the group's members. Example: <ul style="list-style-type: none">• member |
| User Membership Attribute | The attribute field to use when loading the user's groups. Example: <ul style="list-style-type: none">• memberOf |
| Use the User Membership Attribute, when finding the user's group membership | Check this box if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.) <ul style="list-style-type: none">• If this box is checked, your application will use the group membership attribute on the user when retrieving the members of a given group. This will result in a more efficient retrieval.• If this box is not checked, your application will use the members attribute on the group ('member' by default) for the search. |

Connect Bitbucket to Crowd

You can configure Bitbucket Data Center and Server to use Atlassian Crowd for user and group management, and for authentication and authorization.

Atlassian Crowd is an application security framework that handles authentication and authorization for your web-based applications. With Crowd you can integrate multiple web applications and user directories, with support for single sign-on (SSO) and centralized identity management. See the [Crowd Administration Guide](#).

Connect to Crowd if you want to use Crowd to manage existing users and groups in multiple directory types, or if you have users of other web-based applications.

See also this [information about deleting users and groups](#) in Bitbucket.



Connecting Atlassian Bitbucket to your external directory is not sufficient to allow your users to log in to Bitbucket. You must explicitly grant them access to Bitbucket in the [global permission screen](#).

We recommend that you use groups instead of individual accounts when granting permissions. However, be careful not to add more users to those groups that your Bitbucket license allows. If the license limit is exceeded, your developers will not be able to push commits to repositories, and Bitbucket will display a warning banner. See [this FAQ](#).

On this page:

- [Server settings](#)
- [Crowd permissions](#)
- [Advanced settings](#)
- [Single sign-on \(SSO\) with Crowd](#)
- [Using multiple directories](#)



Managing 500+ users across Atlassian products? Find out how easy, scalable and effective it can be with Crowd Data Center! See [centralized user management](#).

To connect Bitbucket to Crowd:

1. Log in as a user with 'Admin' permission.
2. In the Bitbucket administration area, click **User Directories** (under 'Accounts').
3. Click **Add Directory** and select **Atlassian Crowd**.
4. Enter settings, as described below.
5. Test and save the directory settings.
6. Define the directory order, on the **Directories** tab, by clicking the blue up- and down-arrows next to each directory. The directory order has the following effects:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

Server settings

| Setting | Description |
|---------|-------------|
|---------|-------------|

| | |
|----------------------|--|
| Name | A meaningful name that will help you to identify this Crowd server amongst your list of directory servers. Examples: <ul style="list-style-type: none"> • Crowd Server • Example Company Crowd |
| Server URL | The web address of your Crowd console server. Examples: <ul style="list-style-type: none"> • <code>http://www.example.com:8095/crowd/</code> • <code>http://crowd.example.com</code> |
| Application Name | The name of your application, as recognized by your Crowd server. Note that you will need to define the application in Crowd too, using the Crowd administration Console. See the Crowd documentation on adding an application . |
| Application Password | The password which the application will use when it authenticates against the Crowd framework as a client. This must be the same as the password you have registered in Crowd for this application. See the Crowd documentation on adding an application . |

Crowd permissions

Bitbucket offers **Read Only** permissions for Crowd directories. The users, groups and memberships in Crowd directories are retrieved from Crowd and can only be modified from Crowd. You cannot modify Crowd users, groups or memberships using the Bitbucket administration screens.

For local Bitbucket directories, **Read Only** and **Read/Write** permissions are available.

Advanced settings

| Setting | Description |
|------------------------------------|---|
| Enable Nested Groups | Enable or disable support for nested groups. Before enabling nested groups, please check to see if the user directory or directories in Crowd support nested groups. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups. |
| Enable Incremental Synchronization | Enable or disable incremental synchronization. Only changes since the last synchronization will be retrieved when synchronizing a directory. Note that full synchronization is always executed when restarting the application. |
| Synchronization Interval (minutes) | Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes. |

Single sign-on (SSO) with Crowd

Once the Crowd directory has been set up, you can enable Crowd SSO integration by adding the following setting to `shared/bitbucket.properties` in the [home directory](#) (create this file if it doesn't exist yet):

bitbucket.properties

```
# Whether SSO support should be enabled or not. Regardless of this setting SSO authentication
# will only be activated when a Crowd directory is configured in Bitbucket that is configured
# for SSO.
plugin.auth-crowd.sso.enabled=true
```

Please note that you will need to correctly set up the domains of the applications involved in SSO. See [Crowd SSO Domain examples](#).

In addition to this property, Crowd SSO integration can be tuned using the system properties described on [Bitbucket config properties](#).

Using multiple directories

When Bitbucket is connected to Crowd you can map Bitbucket to multiple user directories in Crowd.

For Crowd 2.8, and later versions, there are two different membership schemes that Crowd can use when multiple directories are mapped to an integrated application, and duplicate user names and group names are used across those directories. The schemes are called 'aggregating membership' and 'non-aggregating membership' and are used to determine the effective group memberships that Bitbucket uses for *authorization*. See [Effective memberships with multiple directories](#) for more information about these two schemes in Crowd.

Note that:

- *Authentication*, for when Bitbucket is mapped to multiple directories in Crowd, only depends on the mapped groups in those directories the aggregation scheme is not involved at all.
- For inactive users, Bitbucket only checks if the user is active in the first (highest priority) directory in which they are found to determine *authentication*. The membership schemes described above are not used when Crowd determines if a user should have access to Bitbucket.
- When a user is added to a group, they are only added to the first writeable directory available, in priority order.
- When a user is removed from a group, they are only removed from the group in the first directory the user appears in, when non-aggregating membership is used. With aggregating membership, they are removed from the group in *all* directories the user exists in.

An administrator can set the aggregation scheme that Bitbucket uses when integrated with Crowd. Go to the **Directories** tab for the Bitbucket instance in Crowd, and check **Aggregate group memberships across directories** to use the 'aggregating membership' scheme. When the checkbox is clear 'non-aggregating membership' is used.

Note that changing the aggregation scheme can affect the authorization permissions for your users, and how directory update operations are performed.

Global permissions

User and group access

Bitbucket Data Center and Server uses four levels of account permissions to control user and group access to Bitbucket projects and to the Bitbucket instance configuration.

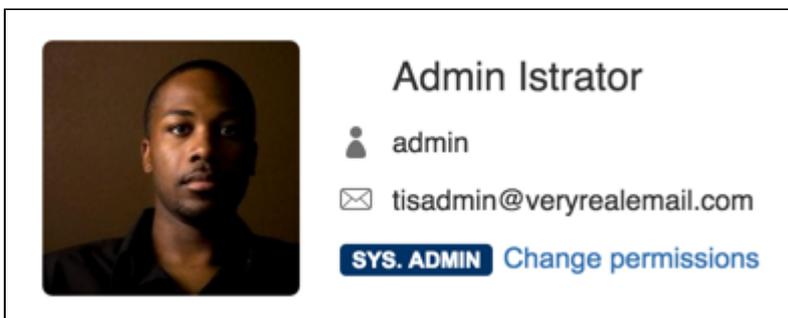
Related pages:

- [Users and groups](#)
- [Using project permission](#)

User accounts that have not been assigned "BitbucketUser" permission or higher, either directly or through group membership, will not be able to log in to Bitbucket. These users are considered unlicensed and do not count towards your Bitbucket license limit.

You can also [apply access permissions to projects](#).

A user's permission level is displayed on the user's page seen from the admin area.



The image shows a user profile card for 'Admin Istrator'. It includes a profile picture of a man, the name 'Admin Istrator', a user ID 'admin', and an email address 'tisadmin@veryrealemail.com'. At the bottom, there is a blue button labeled 'SYS. ADMIN' and a link 'Change permissions'.

| | Login / Browse | Create projects | Manage users / groups | Manage global permissions | Edit application settings | Edit server config |
|-----------------|----------------|-----------------|-----------------------|---------------------------|---------------------------|--------------------|
| Bitbucket User | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Project Creator | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Admin | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| System Admin | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

To edit the account permissions for an existing Bitbucket user or group:

1. In the admin area, select **Global permissions** (under 'Accounts').
2. Select, or clear, the permission checkboxes as required.
3. Select in the **Add Users** or **Add Groups** field to set permissions for additional users or groups.

You can remove all permissions for a user or group by clicking the X at the right-hand end of the row (when you hover there). This will remove that user or group.

Global Permissions

User access

| Name | System Admin [?] | Admin [?] | Project Creator [?] | Bitbucket User [?] |
|----------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| <input type="text"/> | | | | System Admin [?] Add |
| Admin Istrator | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Group access

| Name | System Admin [?] | Admin [?] | Project Creator [?] | Bitbucket User [?] |
|---------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| Add Groups | | | | Admin [?] Add |
| bitbucket-admins | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| bitbucket-users | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| jira-administrators | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Policies

Policies in Bitbucket Data Center are also set in Global Permissions and contain instance-wide permissions associated with user access to projects and repositories. Control who can delete a repository based on what level of permissions a user has including System Admin, Admin, Project Admin, and Repository Admin. By default, a user with Repository Admin access and higher can delete repositories.

To set the delete repository policy:

1. In **Global Permissions**, select the **Policies** tab.
2. Select from the options of permissions, who can delete a repository.
3. Select **Save**.

Global Permissions

Global permissions determine what a user can do based on their access and any additional set policies.

User and group access [Policies](#)

Policies contain enforceable, instance-wide permissions associated with user access to projects and repositories.

Select who can delete repositories.

System Admin, Admin, Project Admin, Repository Admin **DEFAULT**
 System Admin, Admin, Project Admin
 System Admin, Admin
 System Admin only

Save Cancel

Setting up your mail server

Setting up Bitbucket Data Center and Serverto use your SMTP mail server:

- allows Bitbucket to send [notifications](#) about events to do with commit discussions, [pull requests](#) and repositories. Note that if the mail server fails, notifications will be dropped.
- allows Bitbucket to email a link to a newly created user, which the user can use to generate their own password.
- allows a user to reset their password if they forget it.

See [Supported platforms](#) for the mail clients supported by Bitbucket.

Go to the administration area and click **Mail server** (under 'Settings'). Complete and save the form.

Mail server configuration

Mail settings

Hostname*

The hostname of the mail server (for example "localhost" or "192.168.1.15")

Port

The port of the mail server (if unspecified, the port 25 will be used). Typically port 25 or 587 for SMTP, and port 465 for SMTPS

Username

The username to use to connect to the mail server

Password

The password to use to connect to the mail server

Protocol

Select the protocol to use when connecting to the mail server

Use SSL/TLS if available

If the SMTP server supports the STARTTLS extension this will be used to encrypt mail with TLS, otherwise plaintext will be used. SMTPS servers always support SSL/TLS

Always use SSL/TLS

If the SMTP server does not support the STARTTLS extension mail will not be sent. SMTPS servers always support SSL/TLS

Email from*

Specifies the From: header in notification emails (for example: noreply@yourcompany.com)

Send a test email

Recipient

The email address to send the test message to

Hostname

The hostname of the mail server (for example "localhost" or "192.168.1.15").

Port

The port of the mail server (if unspecified, the port 25 will be used).

Username

The username to use to connect to the mail server.

Password

The password to use to connect to the mail server.

Protocol

Use either SMTP or SMTPS when connecting to the mail server.

When using SMTP, you can specify that:

- SSL/TLS is used if supported by the mail server, otherwise mail is sent in plaintext.
- mail should only be sent if the mail server supports SSL/TLS.

See [Securing email notifications](#) below.

Use SSL/TLS if available

If the SMTP server supports the STARTTLS extension this will be used to encrypt mail with SSL/TLS otherwise plaintext will be used. SMTPS servers always support SSL/TLS.

Always use SSL/TLS

If the SMTP server does not support the STARTTLS extension mail will not be sent. SMTPS servers always support SSL/TLS.

Email from

Specifies the 'From' header in notification emails (for example: noreply@yourcompany.com).

Send a test email

Enter an email address to send a test email to check that the mail server is configured correctly.

**Anonymous user**

If you wish to set up the outgoing mail server as an anonymous user, simply leave the username and password fields empty. However, in Chrome, these fields may be auto-populated, leading to an error as a workaround, try using a different browser.

Securing email notifications

Bitbucket 3.6 and later versions support the following protocols:

- SMTP, where mail is not encrypted.
- SMTP encrypted by SSL/TLS using the STARTTLS extension, where the protocol conversation is upgraded only if SSL/TLS is supported by the mail server, but otherwise remains as plaintext.
- SMTP, where STARTTLS support is required on the mail server, otherwise mail is not sent.
- SMTPS (where the whole protocol conversation uses SSL/TLS).

Note that if you use either SMTP with STARTTLS, or SMTPS, and connect to a self-signed mail server, you may need to import the server's certificate and set up a custom cacerts file for Bitbucket (just as you do for any outbound SSL/TLS connection to a self-signed server). See this [Bitbucket knowledge base article](#) for information about how to do that.

Configuring the mail server to use Gmail

If you wish to connect to a Gmail account for email notifications in Bitbucket, refer to the [Configuring the Mail Server to Use Gmail](#) guide.

In particular, note that Gmail won't show images in the email because of the way that Google loads images on their servers. For Google Apps, a Bitbucket administrator can solve the problem by adding the Bitbucket domain name to a whitelist see <https://support.google.com/a/answer/3299041?hl=en> for more information.

Integrate with Atlassian applications

When you integrate Bitbucket Data Center and Server with Atlassian applications, you get the following benefits:

| Application | Integration feature | Compatibility | | |
|---|--|---------------------------|--------------------------------|--|
|  | Using Smart Commits | Jira 7.1+ | Bitbucket 4.2+ | |
| | Related branches, commits and pull requests are all summarized in the Development panel in a Jira issue. | Jira 6.2+ | Stash 2.10+ | |
| | Create Git branches from within Jira and Jira Agile. | Jira 6.1+ | Stash 2.8+ | |
| | Transition Jira issues from within Bitbucket | Jira 5.0+ | Stash 2.7+ | |
| | See the Jira issues related to Bitbucket commits and pull requests | Jira 5.0+ | Stash 2.1+ | |
| | See all the code changes committed for the issue (on the Jira Source tab).

Click through to see a changed file , or the full commit , in Bitbucket. | Jira 5.0.4+ | Plugin version bundled in Jira | |
| | | Jira 5.05.0.3 | Jira Fisheye Plugin 5.0.4.1 | |
| | | Jira 4.4.x | Jira Fisheye Plugin 3.4.12 | |
| Jira 4.3.x | | Jira Fisheye Plugin 3.1.8 | | |
|  | When Bitbucket is integrated with Hipchat , notifications are sent to a Hipchat room whenever someone pushes to a repository in Bitbucket. | | Stash 2.2+ | |

| | | | |
|---|---|---------------------------------|----------------------------|
|  Bamboo | <p>Bamboo responds to repository events published by Bitbucket to:</p> <ul style="list-style-type: none"> • Trigger a plan build when a developer pushes to the connected repository. • Create or delete plan branches when a developer creates or removes a branch in the connected repository. <p>When you link a build plan to a Bitbucket repository, build notifications are automatically enabled.</p> <p>See Bamboo integration.</p> | Bamboo 5.6+ | Stash 3.1+ |
| | <p>See the latest build status for a commit when viewing Bitbucket commits and pull requests.</p> | Bamboo 4.4+ | Stash 2.1+ |
|  Sourcetree | <p>When you have Sourcetree installed, you can:</p> <ul style="list-style-type: none"> • clone a Bitbucket repository using Sourcetree. • check out a branch in Sourcetree, when viewing files, commits, or branches in a Bitbucket repository. | Sourcetree 1.7+ | Stash 2.7+ |
|  Crowd | <p>When Bitbucket is integrated with Crowd, you can:</p> <ul style="list-style-type: none"> • use Crowd for user and group management, and for authentication. | | |

Jira integration

Integrating Bitbucket Data Center and Server with Jira Software makes it easy to keep everyone up to date on code changes and helps to minimize switching between tools. Once they're integrated, you can link a Bitbucket commit, branch, or pull request to a Jira issue by including the issue key.

In Jira you can then:

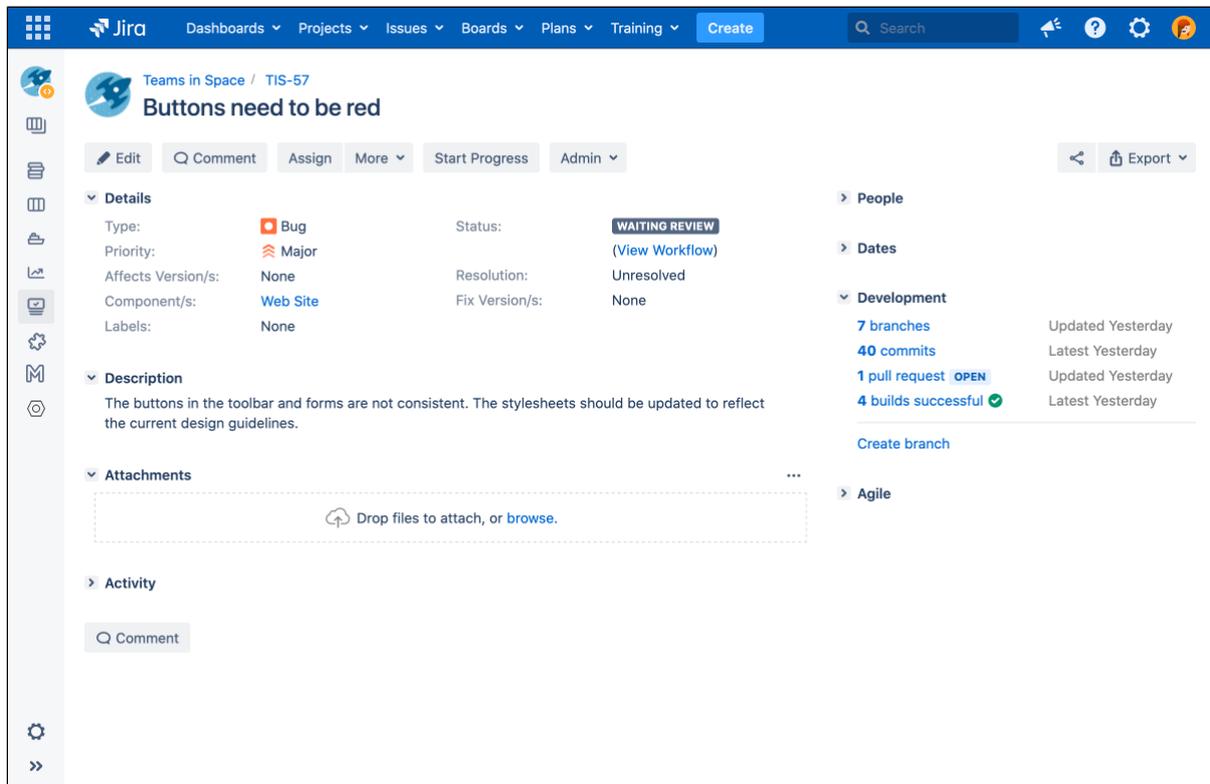
- see real-time development status updates within issues
- trigger issue transitions using Bitbucket events
- check development progress for a version
- see development information in Jira Software Cloud using OAuth

And in Bitbucket you can:

- see assigned Jira issues on the dashboard
- interact with Jira issues

You can also use Jira Software for delegated management of your Bitbucket users. To learn more, see [External user directories](#).

See real-time development status updates within Jira issues



Get visibility into development status from within the context of an issue. The development panel shows a summary of related branches, commits, pull requests, and builds. You can also select the links within this panel to see more information.

When it's time to pick up a new task, you can then create a branch directly from an issue. Using the issue type and summary, Jira will suggest the branch type and branch name, helping you get to work faster.

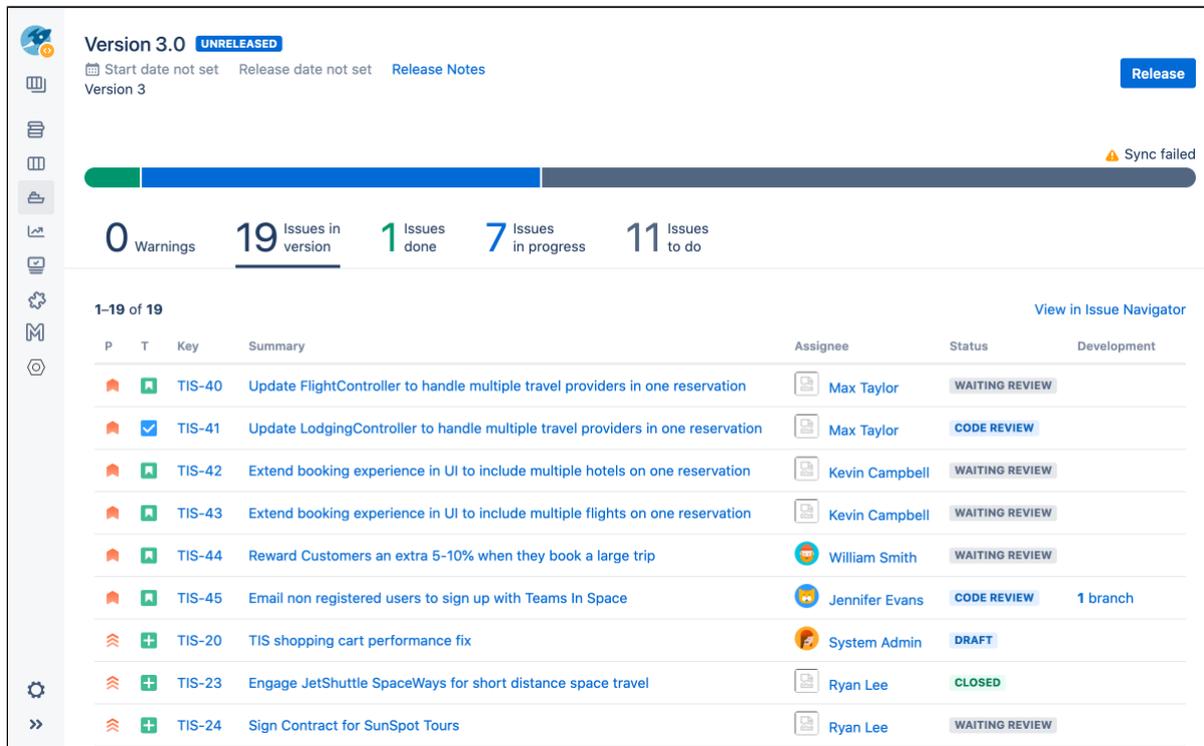
Trigger Jira issue transitions using Bitbucket events

With just a few clicks, you can configure your Jira workflow to respond to events in Bitbucket. For example, when a pull request is created, Jira can automatically transition the status of a linked issue. The events available in Bitbucket are:

- Branch created
- Commit created
- Pull request created
- Pull request merged
- Pull request declined

[Learn more about advanced workflow configuration](#) in the Jira Software documentation.

Check development progress for a version in Jira



The screenshot shows the Jira release hub for Version 3.0, which is currently UNRELEASED. The interface includes a progress bar at the top, a 'Release' button, and a summary of issues: 0 Warnings, 19 Issues in version, 1 Issue done, 7 Issues in progress, and 11 Issues to do. Below this is a table of issues with columns for Priority (P), Type (T), Key, Summary, Assignee, Status, and Development. The issues listed are:

| P | T | Key | Summary | Assignee | Status | Development |
|--------|------|--------|---|----------------|----------------|-------------|
| High | Task | TIS-40 | Update FlightController to handle multiple travel providers in one reservation | Max Taylor | WAITING REVIEW | |
| High | Task | TIS-41 | Update LodgingController to handle multiple travel providers in one reservation | Max Taylor | CODE REVIEW | |
| High | Task | TIS-42 | Extend booking experience in UI to include multiple hotels on one reservation | Kevin Campbell | WAITING REVIEW | |
| High | Task | TIS-43 | Extend booking experience in UI to include multiple flights on one reservation | Kevin Campbell | WAITING REVIEW | |
| High | Task | TIS-44 | Reward Customers an extra 5-10% when they book a large trip | William Smith | WAITING REVIEW | |
| High | Task | TIS-45 | Email non registered users to sign up with Teams In Space | Jennifer Evans | CODE REVIEW | 1 branch |
| Medium | Task | TIS-20 | TIS shopping cart performance fix | System Admin | DRAFT | |
| Medium | Task | TIS-23 | Engage JetShuttle SpaceWays for short distance space travel | Ryan Lee | CLOSED | |
| Medium | Task | TIS-24 | Sign Contract for SunSpot Tours | Ryan Lee | WAITING REVIEW | |

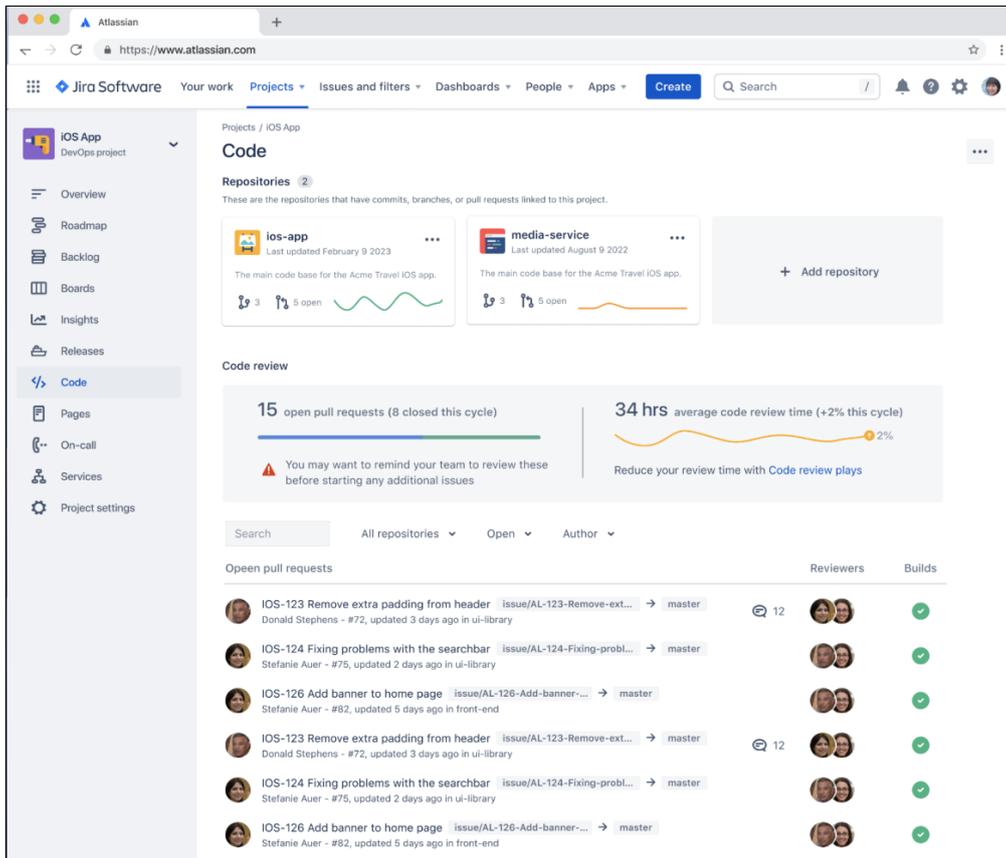
From the Jira release hub, you can see the progress of a version and determine which issues are going to ship at a glance. You can also see the commits related to each issue, helping you spot potential development issues that could cause problems.

[Learn more about checking the progress of a version](#) in the Jira Software documentation.

See development information in Jira Software Cloud using OAuth

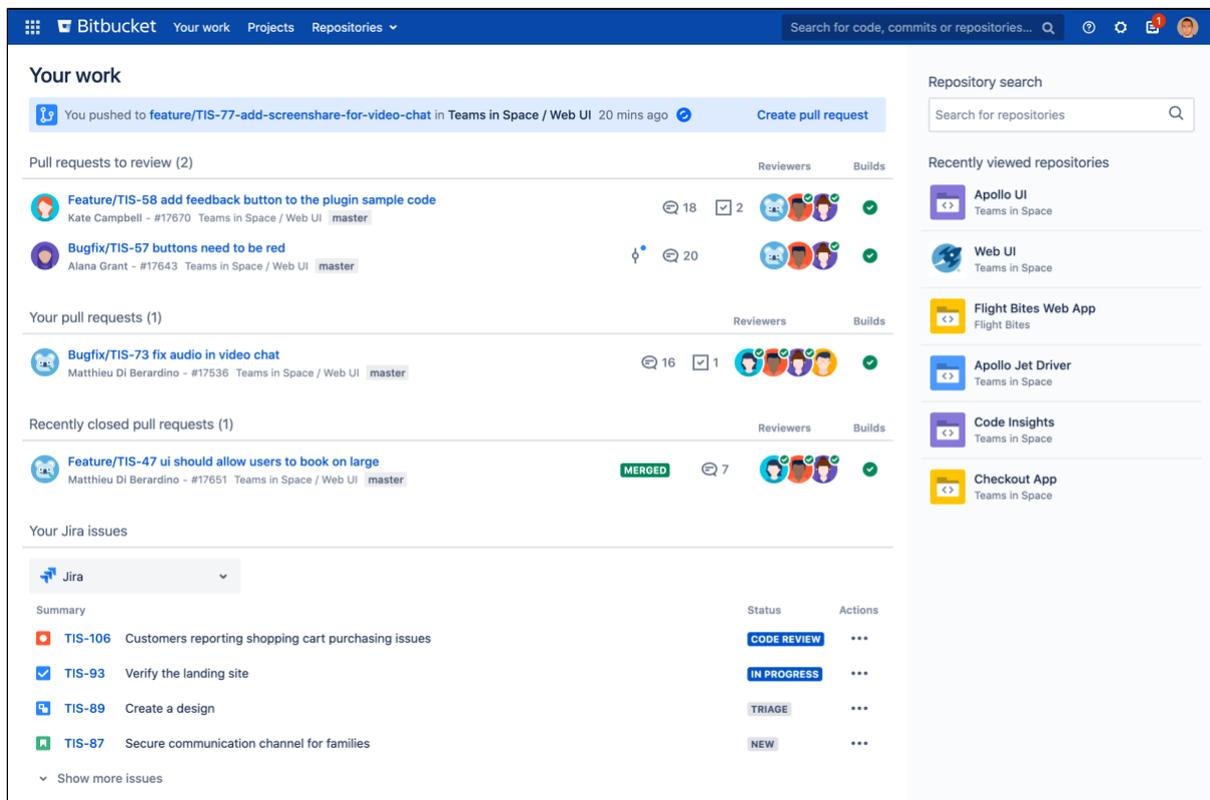
Integrating with Jira Software Cloud enables Bitbucket Data Center to send its development information using OAuth credentials. Integration not only sends development information such as branches, commits, and pull requests, but also give access to new automation and reporting features in Jira Software Cloud such as:

- [calculating Cycle time metrics](#)- the time that you take work to get from the first commit on a branch to production.
- linked Bitbucket repositories are displayed [in the Jira Software Cloud's code tab](#) for quick and easy access.
- triggering [Automation for Jira](#) events - these include Branch created, Commit created, Pull request created, Pull request declined, and Pull request merged. If you haven't linked your Jira and Bitbucket accounts, you'll be prompted to do so the first time you use these triggers.



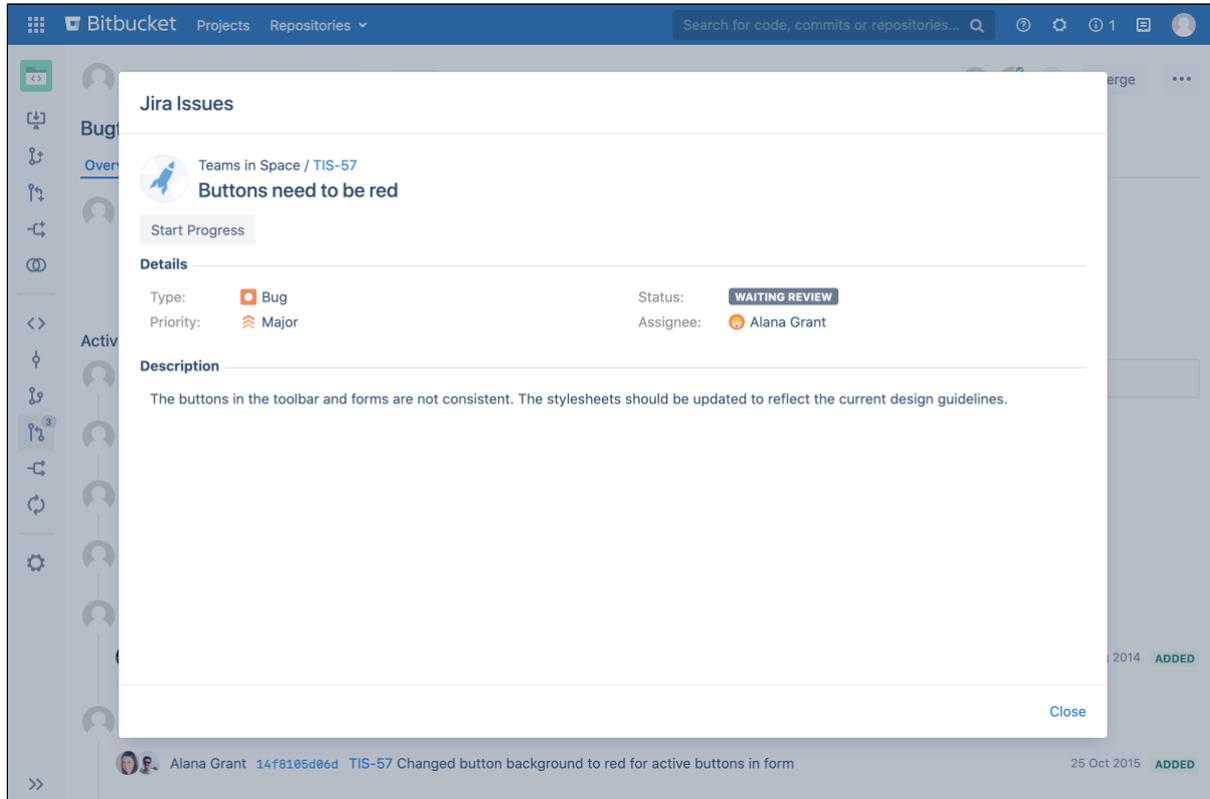
Learn more about integrating with Jira Software Cloud using OAuth.

See assigned Jira issues on the Bitbucket dashboard



On the Bitbucket **Your work** dashboard, you can see open Jira issues assigned to you. This makes it easy to see what's coming up at a glance without jumping between tools. When it's time to start a new task, you can then select the issue key to open the issue summary modal, or you can create a branch from the **Actions** menu and get started.

Interact with Jira issues in Bitbucket



When you mention a Jira issue key in a Bitbucket commit, branch, or pull request, the two are automatically linked. In Bitbucket you can then select the issue key to see more information or interact with the issue. Selecting the issue key in the modal will also take you straight to the issue in Jira.

To learn how to integrate them now, see [Link Bitbucket with Jira](#).

Link Bitbucket with Jira

See [Jira integration](#) for a description of all the integrations you get when Bitbucket Data Center and Server is linked with Jira.

You can also use Jira Software for delegated user management. See [External user directories](#).

This page describes how to link Bitbucket to Jira Data Center, Server, and Cloud using Application Links as well as connecting Bitbucket Data Center with Jira Software Cloud using OAuth.

On this page

- [Link Bitbucket with Jira](#)
- [Link Bitbucket with Jira Software Cloud](#)
- [Application Links and OAuth connections](#)
- [Integrate with Jira Software Cloud using OAuth](#)
- [Troubleshoot integration with Jira Software](#)

Link Bitbucket with Jira

You can integrate Bitbucket with one or more instances of Jira by means of 'application links'. You set up application links either:

- during the Bitbucket install process, using the [Setup Wizard](#), or
- at any time after installation, as described below.

To link Bitbucket to a Jira server:

1. Click **Application Links** (under 'Settings') in the Bitbucket admin area.
2. Enter the URL for the Jira site you want to link to and click **Create new link**.
3. Complete the application link wizard to connect Bitbucket to your Jira. You *must* make use of the automatic link-back from Jira to Bitbucket to get full integration (you'll need System Admin global permission for that).

Note that:

- Bitbucket only begins scanning commit messages for issue keys on the first push after you created the application link to Jira. The scan may take a short time.
- Jira permissions are respected, so a user who is not permitted to transition an issue will not see the transition buttons in Bitbucket.
- If Bitbucket is linked with multiple Jira sites and the projects happen to have the same key, only the issue from the instance marked as will be displayed. See [Making a primary link for links to the same application type](#).
- The following [system plugins](#) must be enabled in Bitbucket. These are bundled and enabled by default in Bitbucket:
 - Atlassian Navigation Links Plugin (com.atlassian.plugins.atlassian-nav-links-plugin)
 - Bitbucket Dev Summary Plugin (bitbucket-jira-development-integration-plugin).

See [Link Atlassian applications to work together](#) and [OAuth security for application links](#) for more details.

Link Bitbucket with Jira Software Cloud

There are port restrictions, and other considerations, when linking Bitbucket with Jira Software Cloud using Application Links.

Your local server must use a valid SSL certificate, and it must be accessible on port 80 or 443. For more information, see [this Atlassian Cloud documentation](#).

If you have an internet-facing firewall, make sure to allow the IP range used by Atlassian to reach your internal network. For up-to-date information on that, see [Atlassian cloud IP ranges and domains](#).

Application Links and OAuth connections

To fully benefit from how data is shared between Jira Software Cloud and Bitbucket, its helpful to configure both an Application Link and OAuth credential.Bitbucket requests data from Jira using AppLinks. Adding an OAuth connection brings moreBitbucket into Jira.

Integrate with Jira Software Cloud using OAuth

Integrate with Jira Software Cloud to enable Bitbucket Data Center to send it development info such as branches, commits, and pull requests, as well as to gain access to automation and reporting features in Jira Software Cloud, like:

- calculating Cycle time metrics- the time that you take work to get from the first commit on a branch to production.
- linked Bitbucketrepositories are displayed in the Jira Software Cloud's code tabfor quick and easy access.
- triggeringAutomation for Jira events - these include Branch created, Commit created, Pull request created, Pull request declined, and Pull request merged. If you haven't linked your Jira andBitbucketaccounts, you'll be prompted to do so the first time you use these triggers.

To begin, you must be a system admin forBitbucket as well as a site admin for Jira Software Cloud. To integrate, first, create OAuth credentials in Jira Software Cloud and then use them to register yourBitbucket instance.

Step 1: Create OAuth credentials in Jira Software Cloud

1. Navigate to **Jira home > Jira settings > Apps**.
2. Select **OAuth credentials** (under **Atlassian Marketplace** in the side navigator).
3. Select **Create new credentials**.
4. Provide the following details:

| Field | Description |
|-----------------|--|
| App name | Any name that describes your Bitbucket instance. This could be "Bitbucket" if your company has a single instance. If your company has multiple instances of Bitbucket, we suggest using the hostname of the server.
Example: bitbucket.mycompany.com |
| Server base URL | Your specified Bitbucket base URL . (Value used from the previous line)
Example: https://bitbucket.mycompany.com/ |

| | |
|----------------------------|---|
| Logo URL | A URL to the Bitbucket logo, which will be used as an icon in the list of credentials:BASE_URL/plugins/servlet/create-branch/icon.png
Example: https://bitbucket.mycompany.com/plugins/servlet/create-branch/icon.png |
| Permissions (required) | Allows creation of a key for development information.
Example: Select "Development information" |
| Create Branch URL Template | This URL allows the Create branch action shortcut. The template contains information that you provide while creating a branch between your Jira issue and the dev tool by inserting an issue key.
Example: https://bitbucket.mycompany.com/plugins/servlet/create-branch?issueKey={issue.key}&issueSummary={issue.summary} |

5. Select **Create**.

For more information on OAuth credentials in Jira Software Cloud, see their [Integrate with self-hosted tools using OAuth documentation](#).

Step 2: Register a Jira Software Cloud site in Bitbucket

To register a Jira Software Cloud site:

1. In the Bitbucket Data Center Administration area, select **Jira Cloud Integration** (under System).
2. Select **Register site**.
3. Complete the form using the OAuth credentials you created in Jira Software Cloud by providing the following details:

[Back to Jira Software Cloud integration](#)

Register a Jira Software Cloud site

Complete this form using the OAuth credentials you created in Jira Software Cloud. [Learn how to create OAuth credentials](#)

Site name

Site URL

Client ID

Secret

| Field | Description |
|-----------|--|
| Site name | Any name that describes your Jira Software Cloud site to users. This could be Jira if your company only uses a single site. If your company has multiple Jira sites, we suggest using the hostname of the site.
Example: mycompany.atlassian.net |
| Site URL | The site URL of your Jira Software Cloud site.
Example: https://mycompany.atlassian.net/ |
| Client ID | Copied from Jira Software Cloud OAuth credentials page. |
| Secret | Copied from Jira Software Cloud OAuth credentials page. |

4. Select **Submit**.

Troubleshoot integration with Jira Software

There are a few situations where the integration of Bitbucket with Jira can produce an error or may not function as expected:

Unable to see the Development panel within an issue

You must have the 'View Development Tools' permission in Jira to see the Development panel. See [Managing Global Permissions](#).

You don't have permission to access the project

If you don't have permission to access the project within Jira then Bitbucket will be unable to display issues.

The issue key is invalid

Bitbucket doesn't check for invalid issue keys, such as 'UTF-8'. An error will result if Bitbucket tries to connect to an issue that doesn't exist. See this issue:

 **BSERV-2470** - JIRA Integration: Check for issue validity before linking issues GATHERING INTEREST

The issue keys are of a custom format

Bitbucket assumes that issue keys are of the default format (that is, two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example TEST-123). By default, Bitbucket will not recognize custom issue key formats. See [Using custom Jira issue keys with Bitbucket Server](#) for details.

Having trouble integrating your Atlassian products with application links?

We've developed a [guide to troubleshooting application links](#), to help you out. Take a look at it if you need a hand getting around any errors or roadblocks with setting up application links.

Configuring Jira integration in the Setup Wizard

This page describes the 'Jira Software integration' screen of the Setup Wizard that runs automatically when you launch Bitbucket Data Center and Server for the first time.

The Setup Wizard guides you in configuring the Bitbucket connection with Jira Software using the most common options. You can also configure Jira Software integration from the Bitbucket administration screens at any time after completing the Setup Wizard.

There are two aspects to integrating Bitbucket with Jira Software:

- linking Jira Software and Bitbucket to enable the integration features. See [Jira integration](#).
- delegating Bitbucket user and group management to your Jira Software server. See [Connecting Bitbucket Server to Jira for user management](#).

Connecting to Jira Software in the Setup Wizard

To configure Jira Software integration while running the Bitbucket Setup Wizard:

1. Configure the following setting in Jira Software: [Configuring Jira Software application options](#).
2. Click **Integrate with Jira** and enter the following information when you get to the 'Connect to Jira' step of the setup wizard:

Jira base URL

The web address of your Jira server. Examples are:

`http://www.example.com:8080/jira/`

`http://jira.example.com`

Jira admin username

The credentials for a user with the 'Jira System Administrators' global permission in Jira.

Jira password

The credentials for a user with the 'Jira System Administrators' global permission in Jira.

Bitbucket Server base URL

Jira will use this URL to access your Bitbucket instance. The URL you give here will override the base URL specified in your Bitbucket administration console, for the purposes of the Jira connection.

3. Click **Connect**.
4. Finish the setup process.

JIRA integration

Use JIRA as a central server for user management or connect your issues and changesets simply by adding issue keys to your commit messages.

- Automatically import all your JIRA users.
- See what code changes are related to a specific JIRA issue.
- Quickly navigate to JIRA issues that are linked to commits.
- Keep track of bug-fixes.

Create JIRA connection

JIRA base URL* For example: <http://jira.atlassian.com>

JIRA administrator username*

This user must have system administrator rights in JIRA

JIRA password* The JIRA user's password

Stash base URL* JIRA will access Stash from this URL

Using JIRA as my user database

If you have JIRA 4.3 or later, Stash can use JIRA for user management. This is not recommended for more than 500 users. [Learn more about JIRA user management.](#)

Use JIRA as my user database

Troubleshooting

This section describes the possible problems that may occur when integrating your application with JIRA via the setup wizard, and the solutions for each problem.

| Symptom | Cause | Solution |
|---------|-------|----------|
|---------|-------|----------|

| | | |
|--|--|--|
| <p>The setup wizard displays one of the following error messages:</p> <ul style="list-style-type: none"> Failed to create application link from JIRA server at <URL> to this <application> server at <URL>. Failed to create application link from this <application> server at <URL> to JIRA server at <URL>. Failed to authenticate application link from JIRA server at <URL> to this <application> server at <URL>. Failed to authenticate application link from <application> server at <URL> to this JIRA server at <URL>. | <p>The setup wizard failed to complete registration of the peer-to-peer application link with JIRA. JIRA integration is only partially configured.</p> | <p>Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.</p> |
| <p>The setup wizard displays one of the following error messages:</p> <ul style="list-style-type: none"> Failed to register <application> configuration in JIRA for shared user management. Received invalid response from JIRA: <response> Failed to register <application> configuration in JIRA for shared user management. Received: <response> | <p>The setup wizard failed to complete registration of the client-server link with JIRA for user management. The peer-to-peer link was successfully created, but integration is only partially configured.</p> | <p>Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.</p> |

| | | |
|---|--|--|
| <p>The setup wizard displays the following error message:</p> <ul style="list-style-type: none"> • Error setting Crowd authentication | <p>The setup wizard successfully established the peer-to-peer link with JIRA, but could not persist the client-server link for user management in your <code>config.xml</code> file. This may be caused by a problem in your environment, such as a full disk.</p> | <p>Please investigate and fix the problem that prevented the application from saving the configuration file to disk. Then remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.</p> |
| <p>The setup wizard displays the following error message:</p> <ul style="list-style-type: none"> • Error reloading Crowd authentication | <p>The setup wizard has completed the integration of your application with JIRA, but is unable to start synchronizing the JIRA users with your application.</p> | <p>Restart your application. You should then be able to continue with the setup wizard. If this solution does not work, please contact Atlassian Support.</p> |
| <p>The setup wizard displays the following error message:</p> <ul style="list-style-type: none"> • An error occurred: <code>java.lang.IllegalStateException: Could not create the application in JIRA /Crowd (code: 500)</code>. Please refer to the logs for details. | <p>The setup wizard has not completed the integration of your application with JIRA. The links are only partially configured. The problem occurred because there is already a user management configuration in JIRA for this <code><application></code> URL.</p> | <p>Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup. Detailed instructions are below.</p> |
| <p>No users can log in after you have set up the application with JIRA integration.</p> | <p>Possible causes:</p> <ul style="list-style-type: none"> • There are no users in the group that you specified on the 'Connect to JIRA' screen. • For FishEye: There are no groups specified in the 'groups to synchronize' section of your administration console. • For Stash: You may not have granted any JIRA groups or users permissions to log in to Stash. | <p>Go to JIRA and add some usernames to the group.</p> <ul style="list-style-type: none"> • For FishEye: Go to the FishEye administration screens and specify at least one group to synchronize. The default is 'jira-users'. • For Stash: Grant the Stash User permission to the relevant JIRA groups on the StashGlobal permissions page. <p>If this solution does not work, please contact Atlassian Support.</p> |

Solution 1: Removing a Partial Configuration The Easiest Way

If the application's setup wizard fails part-way through setting up the JIRA integration, you may need to remove the partial configuration from JIRA before continuing with your application setup. Please follow the steps below.

Remove the partial configuration if it exists, try the 'Connect to JIRA' step again, and then continue with the setup wizard:

1. Log in to JIRA as a user with the **'JIRA System Administrators'** global permission.
2. Click the **'Administration'** link on the JIRA top navigation bar.
3. Remove the application link from JIRA, if it exists:
 - a. Click **Application Links** in the JIRA administration menu. The 'Configure Application Links' page will appear, showing the application links that have been set up.

- b. Look for a link to your application. It will have a base URL of the application linked to JIRA. For example:
 - If you want to remove a link between JIRA and FishEye, look for the one where the **Application URL** matches the base URL of your FishEye server.
 - If you want to remove a link between JIRA and Confluence, look for the one where the **Application URL** matches the base URL of your Confluence server.
 - If you want to remove a link between JIRA and Stash, look for the one where the **Application URL** matches the base URL of your Stash server.
 - c. Click **Delete** next to the application link that you want to delete.
 - d. A confirmation screen will appear. Click **Confirm** to delete the application link.
4. Remove the user management configuration from JIRA, if it exists:
 - a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click **'Other Applications'** in the **'Users, Groups & Roles'** section of the JIRA administration screen.
 - In JIRA 4.4: Select **'Administration' > 'Users' > 'JIRA User Server'**.
 - b. Look for a link to your application. It will have a name matching this format:

```
<Type> - <HostName> - <Application ID>
```

For example:

```
FishEye / Crucible - localhost - 92004b08-5657-3048-b5dc-f886e662ba15
```

Or:

```
Confluence - localhost - 92004b08-5657-3048-b5dc-f886e662ba15
```

If you have multiple servers of the same type running on the same host, you will need to match the application ID of your application with the one shown in JIRA. To find the application ID:

- Go to the following URL in your browser:

```
<baseUrl>/rest/applinks/1.0/manifest
```

Replace `<baseUrl>` with the base URL of your application.

For example:

```
http://localhost:8060/rest/applinks/1.0/manifest
```

- The application links manifest will appear. Check the application ID in the `<id>` element.
- c. In JIRA, click **'Delete'** next to the application that you want to remove.
5. Go back to the setup wizard and try the 'Connect to JIRA' step again.

Solution 2: Removing a Partial Configuration The Longer Way

If solution 1 above does not work, you may need to remove the partial configuration and then add the full integration manually. Please follow these steps:

1. Skip the 'Connect to JIRA' step and continue with the setup wizard, to complete the initial configuration of the application.
2. Log in to JIRA as a user with the **'JIRA System Administrators'** global permission.
3. Click the **'Administration'** link on the JIRA top navigation bar.
4. Remove the application link from JIRA, if it exists:
 - a. Click **Application Links** in the JIRA administration menu. The 'Configure Application Links' page will appear, showing the application links that have been set up.
 - b. Look for a link to your application. It will have a base URL of the application linked to JIRA. For example:
 - If you want to remove a link between JIRA and FishEye, look for the one where the **Application URL** matches the base URL of your FishEye server.
 - If you want to remove a link between JIRA and Confluence, look for the one where the **Application URL** matches the base URL of your Confluence server.

- If you want to remove a link between JIRA and Stash, look for the one where the **Application URL** matches the base URL of your Stash server.
 - c. Click **Delete** next to the application link that you want to delete.
 - d. A confirmation screen will appear. Click **Confirm** to delete the application link.
5. Remove the user management configuration from JIRA, if it exists:
- a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click **Other Applications** in the **Users, Groups & Roles** section of the JIRA administration screen.
 - In JIRA 4.4: Select **Administration** > **Users** > **JIRA User Server**.
 - b. Look for a link to your application. It will have a name matching this format:

```
<Type> - <HostName> - <Application ID>
```

For example:

```
FishEye / Crucible - localhost - 92004b08-5657-3048-b5dc-f886e662ba15
```

Or:

```
Confluence - localhost - 92004b08-5657-3048-b5dc-f886e662ba15
```

If you have multiple servers of the same type running on the same host, you will need to match the application ID of your application with the one shown in JIRA. To find the application ID:

- Go to the following URL in your browser:

```
<baseUrl>/rest/applinks/1.0/manifest
```

Replace `<baseUrl>` with the base URL of your application.

For example:

```
http://localhost:8060/rest/applinks/1.0/manifest
```

- The application links manifest will appear. Check the application ID in the `<id>` element.
- c. In JIRA, click **Delete** next to the application that you want to remove.
6. Add the application link in JIRA again, so that you now have a two-way trusted link between JIRA and your application:
- a. Click **Add Application Link**. Step 1 of the link wizard will appear.
 - b. Enter the **server URL** of the application that you want to link to (the 'remote application').
 - c. Click **Next**.
 - d. Enter the following information:
 - **Create a link back to this server** Check to add a two-way link between the two applications.
 - **Username and Password** Enter the credentials for a username that has administrator access to the remote application.
Note: These credentials are only used to authenticate you to the remote application, so that Application Links can make the changes required for the new link. The credentials are not saved.
 - **Reciprocal Link URL** The URL you give here will override the base URL specified in your remote application's administration console, for the purposes of the application links connection. Application Links will use this URL to access the remote application.
 - e. Click **Next**.
 - f. Enter the information required to configure authentication for your application link:
 - **The servers have the same set of users** Check this box, because the users are the same in both applications.
 - **These servers fully trust each other** Check this box, because you trust the code in both applications and are sure both applications will maintain the security of their private keys.
For more information about configuring authentication, see [Configuring authentication for an application link](#).
 - g. Click **Create**.

7. Configure a new connection for user management in JIRA:
 - a. Go to the JIRA administration screen for configuring the applications that have been set up to use JIRA for user management:
 - In JIRA 4.3: Click **'Other Applications'** in the **'Users, Groups & Roles'** section of the JIRA administration screen.
 - In JIRA 4.4: Select **'Administration' > 'Users' > 'JIRA User Server'**.
 - b. **Add** an application.
 - c. Enter the **application name** and **password** that your application will use when accessing JIRA.
 - d. Enter the **IP address** or addresses of your application. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
 - **Save** the new application.
8. Set up the JIRA user directory in the application.
 - For Confluence:
 - a. Go to the **Confluence Administration Console**.
 - b. Click **'User Directories'** in the left-hand panel.
 - c. **Add** a directory and select type **'Atlassian JIRA'**.
 - d. Enter the following information:
 - **Name** Enter the name of your JIRA server.
 - **Server URL** Enter web address of your JIRA server. Examples:


```
http://www.example.com:8080/jira/
http://jira.example.com
```
 - **Application name** and **Application password** Enter the values that you defined for Confluence in the settings on JIRA.
 - e. Save the directory settings.
 - f. Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the **'User Directories'** screen.
For details see [Connecting to Crowd or Jira for User Management](#).
 - For FishEye/Crucible:
 - a. Click **Authentication** (under 'Security Settings').
 - b. Click **Setup JIRA/Crowd authentication**. Note, if LDAP authentication has already been set up, you will need to remove that before connecting to JIRA for user management.
 - c. Make the following settings:

| | |
|---|--|
| Authenticate against | Select a JIRA instance |
| Application name and password | Enter the values that you defined for your application in the settings on JIRA. |
| JIRA URL | The web address of your JIRA server. Examples: <div data-bbox="663 1630 1450 1727" style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <pre>http://www.example.com:8080/jira/ http://jira.example.com</pre> </div> |
| Auto-add | Select Create a FishEye user on successful login so that your JIRA users will be automatically added as a FishEye user when they first log in. |
| Periodically synchronise users with JIRA | Select Yes to ensure that JIRA will synchronize all changes in the user information on a regular basis. Change the value for Synchronise Period if required. |

| | |
|-------------------------------------|--|
| When Synchronisation Happens | Select an option depending on whether you want to allow changes to user attributes from within FishEye. |
| Single Sign On | Select Disabled . SSO is not available when using JIRA for user management and if enabled will make the integration fail. |

- d. Click **Next** and select at least one user group to be synchronised from JIRA. If necessary, you could create a new group in JIRA, such as 'fisheye-users', and select this group here.
 - e. Click **Save**.
- For Stash:
 - a. Go to the Stash administration area.
 - b. Click **User Directories** in the left-hand panel.
 - c. **Add** a directory and select type **Atlassian JIRA**.
 - d. Enter the following information:
 - **Name** Enter the name of your JIRA server.
 - **Server URL** Enter web address of your JIRA server. Examples:


```
http://www.example.com:8080/jira/
http://jira.example.com
```
 - **Application name** and **Application password** Enter the values that you defined for Stash in the settings on JIRA.
 - e. Save the directory settings.
 - f. Define the directory order by clicking the blue up- and down-arrows next to each directory on the 'User Directories' screen.
For details see [Connecting Stash to JIRA for user management](#).

Notes

When you connect to Jira Software in the Setup Wizard, the setup procedure will configure *OAuth authentication* between Bitbucket and Jira Software. See [Configuring OAuth authentication for an application link](#) for more information.

Use custom Jira issue keys

Bitbucket Data Center and Server assumes that Jira Software issue keys are of the default format (that is, two or more uppercase letters ([A-Z][A-Z]+), followed by a hyphen and the issue number, for example TEST-123). By default, Bitbucket will not recognize custom issue key formats.

You can use custom issue key formats with Bitbucket, however note that integrations with Jira Software can depend on using the default issue key format in both applications. See [Integrating using custom Jira Software issue keys](#) for more details.

Configure Bitbucket to recognize custom issue key formats by editing `<Bitbucket installation directory>/bin/_start-webapp.sh` (on Windows, `_start-webapp.bat`).

To override the default issue key format, use the `JVM_SUPPORT_RECOMMENDED_ARGS` property, like this:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-Dintegration.jira.key.pattern=\"(<Some different regex>)\""
```

You'll need to restart Bitbucket.

For example, to use lowercase letters in issue keys, use a regex with the parameter like this:

```
'-Dintegration.jira.key.pattern=(?<=^[a-z]-|[\s\p{Punct}]&&[^-])([a-z]{1,10}-\d+)(?![^\W_])'
```

See also [Reindex Jira Software issue keys](#).

As always, be sure to back up your home directory (and perhaps the database) before performing any manual operation on Bitbucket. Consider testing this change on another copy of Bitbucket before using it in production.

Use Smart Commits

Smart Commits allow repository committers to process Jira Software Server issues using special commands in your commit messages.

You can:

- comment on issues
- record time tracking information against issues
- transition issues to any status defined in the Jira Software project's workflow.

A single Smart Commit command cannot span more than one line (you cannot use carriage returns in the commit message), but you can add multiple commands to the same line, or multiple commands on separate lines. See [this example](#) below.

Smart Commits work with Bitbucket Data Center and Server 4.2+ and Jira Software Server 7.1+.

On this page:

- [Smart Commit commands](#)
- [Advanced examples](#)
- [Get Smart Commits working](#)

Related pages:

- [Enabling DVCS Smart Commits](#)

Smart Commit commands

The basic command line syntax for a smart commit message is:

```
<ignored text><ISSUE_KEY><ignored text> #<COMMAND> <optional COMMAND_ARGUMENTS>
```

Any text between the issue key and the smart commit command is ignored.

There are three smart commit commands you can use in your commit messages:

- [comment](#)
- [time](#)
- [transition](#)

Comment

| | |
|--------------------|--|
| Description | Adds a comment to a JIRA Software issue. |
| Syntax | <code><ignored text> ISSUE_KEY<ignored text> #comment <comment_string></code> |
| Example | <code>JRA-34 #comment corrected indent issue</code> |
| Notes | <ul style="list-style-type: none">• The committer's email address must match the email address of a single JIRA Software user with permission to comment on issues in that particular project. |

Time

| | |
|--------------------|---|
| Description | Records time tracking information against an issue. |
| Syntax | <code><ignored text> ISSUE_KEY <ignored text> #time <value>w <value>d <value>h <value>m <comment_string></code> |
| Example | <code>JRA-34 #time 1w 2d 4h 30m Total work logged</code> |

| | |
|--------------|--|
| Notes | <p>This example records 1 week, 2 days, 4 hours and 30 minutes against the issue, and adds the comment 'Total work logged' in the Work Log tab of the issue.</p> <ul style="list-style-type: none"> • Each value for <i>w</i>, <i>d</i>, <i>h</i> and <i>m</i> can be a decimal number. • The committer's email address must match the email address of a single JIRA Software user with permission to log work on an issue. • Your system administrator must have enabled time tracking on your JIRA Software instance. |
|--------------|--|

Workflow transitions

| | |
|--------------------|--|
| Description | Transitions a JIRA Software issue to a particular workflow state. |
| Syntax | <code><ignored text> ISSUE_KEY<ignored text>#<transition_name><comment_string></code> |
| Example | JIRA-090 #close Fixed this today |
| Notes | <p>This example executes the close issue workflow transition for the issue and adds the comment 'Fixed this today' to the issue. Note that the comment is added automatically without needing to use the #comment command.</p> <p>You can see the custom commands available for use with smart commits by visiting the JIRA Software issue and seeing its available workflow transitions:</p> <ol style="list-style-type: none"> 1. Open an issue in the project. 2. Click View Workflow (near the issue's Status). <p>The smart commit only considers the part of a transition name before the first space. So, for a transition name such as <code>finish work</code>, then specifying <code>#finish</code> is sufficient. You must use hyphens to replace spaces when ambiguity can arise over transition names, for example: <code>#finish-work</code>.</p> <p>If a workflow has two valid transitions, such as:</p> <ul style="list-style-type: none"> • <code>Start Progress</code> • <code>Start Review</code> <p>A smart commit with the action <code>#start</code> is ambiguous because it could mean either of the two transitions. To specify one of these two transitions, fully qualify the transition you want by using either <code>#start-review</code> or <code>#start-progress</code>.</p> <ul style="list-style-type: none"> • When you resolve an issue with the <code>#resolve</code> command, you cannot set the Resolution field with smart commits. • If you want to add a comment during the transition, the transition must have a screen associated with it. • The committer's email address must match the email address of a single JIRA Software user with the appropriate project permissions to transition issues. |

Advanced examples

Multiple commands over multiple lines on a single issue

Syntax

```
<ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ...
#<COMMAND_n> <optional COMMAND_n_ARGUMENTS>
```

Commit message

```
JRA-123 #comment Imagine that this is a really, and I mean really, long comment
#time 2d 5h
```

Result

Adds the comment 'This is a really, and I' (but drops the rest of the comment) and logs 2 days and 5 hours of work against issue JRA-123.

Multiple commands on a single issue

Syntax

```
<ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ...
#<COMMAND_n> <optional COMMAND_n_ARGUMENTS>
```

Commit message

```
JRA-123 #time 2d 5h #comment Task completed ahead of schedule #resolve
```

Result

Logs 2 days and 5 hours of work against issue JRA-123, adds the comment 'Task completed ahead of schedule', and resolves the issue.

A single command on multiple issues

Syntax

```
<ISSUE_KEY1> <ISSUE_KEY2> <ISSUE_KEY3> #<COMMAND> <optional COMMAND_ARGUMENTS> etc
```

Commit message

```
JRA-123 JRA-234 JRA-345 #resolve
```

Result

Resolves issues JRA-123, JRA-234 and JRA-345. Multiple issue keys must be separated by whitespace or commas.

Multiple commands on multiple issues

Syntax

```
<ISSUE_KEY1> <ISSUE_KEY2> ... <ISSUE_KEYn> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2>
<optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS>
```

Commit message

```
JRA-123 JRA-234 JRA-345 #resolve #time 2d 5h #comment Task completed ahead of schedule
```

Result

Logs 2 days and 5 hours of work against issues JRA-123, JRA-234 and JRA-345, adds the comment 'Task completed ahead of schedule' to all three issues, and resolves all three issues. Multiple issue keys must be separated by whitespace or commas.

Get Smart Commits working

Smart Commits work with Bitbucket 4.2+ and Jira Software Server 7.1+.

To get Smart Commits working for Jira Software and Bitbucket:

1. Create an application link between Jira Software and Bitbucket. See [Linking Bitbucket Server with Jira](#).
2. Enable smart commits in Jira Software. See [Enabling DVCS Smart Commits](#).

Some limitations of Smart Commits:

- Smart Commits only support the default Jira Software issue key format (that is, two or more uppercase letters, followed by a hyphen and the issue number, for example BAM-123).
- Smart Commits don't provide for field-level updates in Jira Software issues.
- Note that elevated access rights in Jira Software Server can result from the way that Git (and Mercurial) allow commits to be attributed to a user other than the user pushing a change to the repository. If this seems like a risk for your situation, then you should consider disabling Smart Commits on the Jira Software Server instance.

Hipchat notifications

Bitbucket Data Center and Servercan send a notifications to a Hipchat room for activities performed in aproject or repository, and it works with Hipchat Server or Cloud.[Learn more about Hipchat.](#)

What kind of notifications can I get?

The Hipchat integration lets Bitbucket send the following notifications to your Hipchat rooms:

- Pull requestswhen they are created, commented, merged, and declined
- Commitswhen they are pushed and commented

Here's an example of what you might see (from an Atlassian Hipchat room):

| | |
|-----------|---|
| Bitbucket |  Branch bundle-HipChat-integration created by William in Bitbucket/bitbucket-server |
| Bitbucket |  One commit pushed to branch bundle-HipChat-integration by William in Bitbucket/bitbucket - BSERV-7845 Add tests for HipChat integration · 3fa9dcfb558 |
| Bitbucket |  Pull request Bundle HipChat integration created by William in Bitbucket/bitbucket-server |
| Bitbucket |  Pull request Bundle HipChat integration commented on by Alana in Bitbucket/bitbucket-server
Looks good to me!
See more |
| Bitbucket |  Pull request Bundle HipChat integration merged by William in Bitbucket/bitbucket-server |

Install and configure the Hipchat for Bitbucketintegration

To enable Bitbucketto send notifications in Hipchat you must connect Bitbucketto your Hipchat instance. In order to do this you need administration privileges for your Bitbucketinstance and for Hipchat.

To integrate Bitbucketand Hipchat

1. Select **Administration Settings**  > **Hipchat integration**
2. *If using Hipchat Cloud*, click the **Connect Hipchat** button, which takes you to hipchat.com sign up page.
If using Hipchat Server, click the link below the Connect Hipchat button, enter your Hipchat server URL, then click **Connect Hipchat**.
3. Log in to Hipchat with an account that has admin rights.
4. Click **Install** to finish installing the BitbucketHipchat Addon.
5. Select the repository that you want to send notifications from, and the Hipchat room where you want the notifications to appear.
 - You can choose multiple rooms to receive notifications from a repository but must add each connection separately.
 - Repeat this process for all the repositories where you want to send notifications.

In Hipchat, you will see a notification in your room telling you that notifications were enabled.

| | |
|-----------|---|
| Bitbucket |  Notifications from the repository Bitbucket/bitbucket-server will now appear in this room |
|-----------|---|

On this page

- [What kind of notifications can I get?](#)
- [Install and configure the Hipchat for Bitbucketintegration](#)
- [Enable Hipchat notifications for a repository](#)

Enable Hipchat notifications for a repository

Once Hipchat and Bitbucket are [integrated](#), you can enable and disable Hipchat notifications for a particular repo by going to **Settings > Hipchat**. You must have admin privileges for the project or repository you want to enable notifications for.

Initially, you will need to click the **Log in to Hipchat** button to connect your Bitbucket and Hipchat user accounts. If you don't see the **Log in to Hipchat** button you will need to talk to your Bitbucket administrator and get them to install the integration first.

Repository settings

Repository details

SECURITY

Repository permissions

Branch permissions

Access keys

Audit log

WORKFLOW

Branching model

Hooks

Webhooks

Hipchat integration

PULL REQUESTS

Merge checks

Merge strategies

Default reviewers

General

ADD-ONS

Branches and tags

Hipchat integration



Log in to get Hipchat and Bitbucket talking

Keep your team together with real time notifications. Life's easier with good communication.

[Log in to Hipchat](#)

Now you can add room links between your repository and rooms in Hipchat. Search for the room and press **Add** to get things going.

From repository: bitbucket-server

→

🔒

Development

▼

[Add](#)

You can change the notification types by unticking some of the checkboxes.

From repository: bitbucket-server

→

Select HipChat room

▼

[Add](#)

| Repository | Room | |
|--|---|--------|
| bitbucket-server | → Development | Done ✕ |
| Notify room with activity in these areas: | | |
| Pull requests <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Created <input checked="" type="checkbox"/> Commented <input checked="" type="checkbox"/> Merged <input checked="" type="checkbox"/> Declined | Commits <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Pushed <input checked="" type="checkbox"/> Commented | |

Bamboo integration

When you integrate [Bitbucket Data Center and Server](#) with Atlassian's [Bamboo](#) build and deployment server, commit, branch, build and deployment information is shared for users of both applications.

On this page:

- [Benefits of integration](#)
- [Configuration](#)

Benefits of integration

When Bamboo and Bitbucket Server are integrated, you and your team get all the following advantages:

Bitbucket Server tells Bamboo when to build

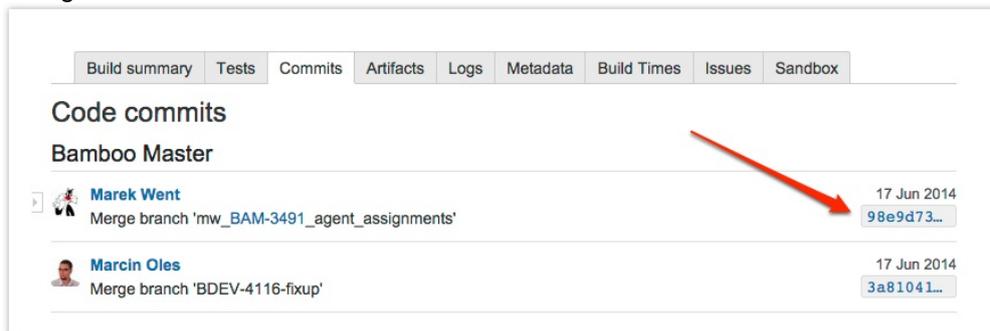
- When a developer pushes to a repository the build is automatically started.

Bitbucket Server tells Bamboo when to update plan branches to match changes in repository branches

- When a developer pushes a new branch to a repository a branch plan is automatically created.
- When a developer deletes a branch in a repository, the branch plan is automatically deleted or disabled.

Bitbucket Server commits are displayed in the relevant Bamboo builds

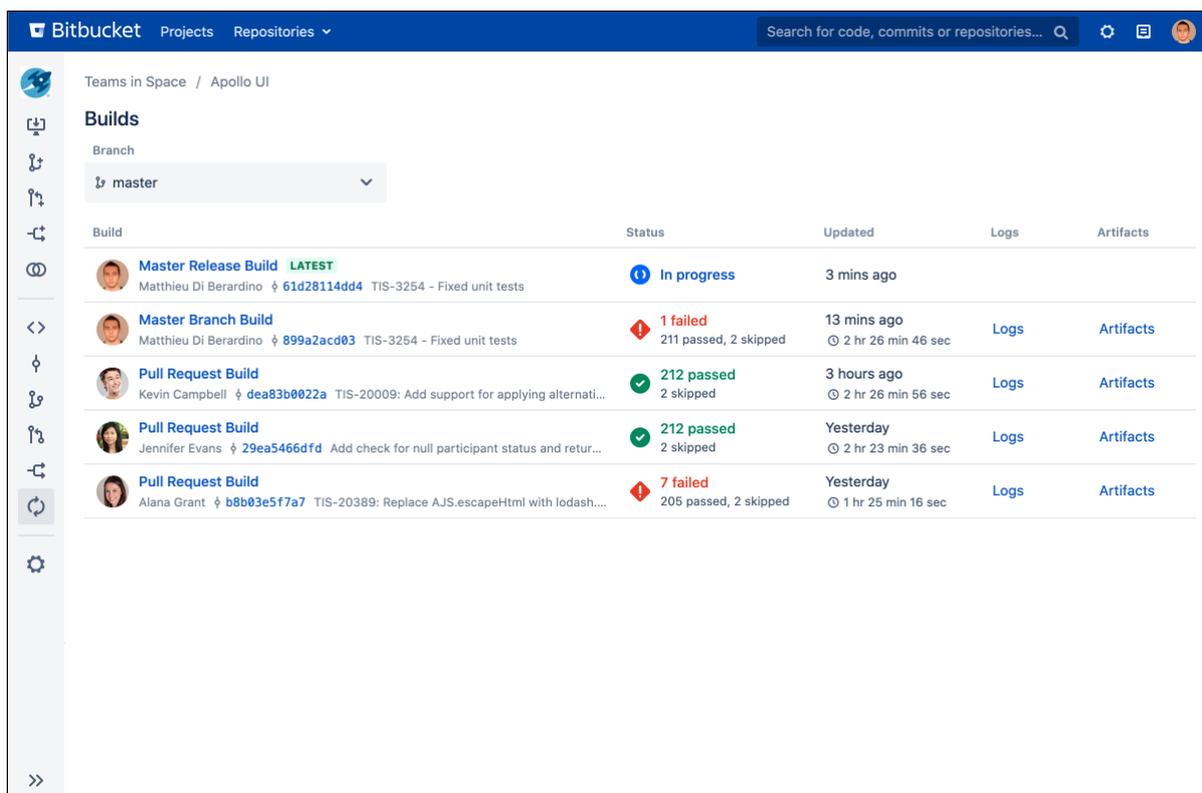
- In Bamboo, you can view all of the commits involved in the build, allowing you to accurately track changes:



- Simply click on a changeset to go to Bitbucket Server, where you can see the commit diff for all of the files that are part of the build.

Bamboo notifies Bitbucket Server automatically about build results

- Build notifications are automatically enabled when you link a build plan to a Bitbucket Server repository.
- Notifications are sent to all linked Bitbucket servers.
- You can see build results and other related information on the Builds, Pull request, Commits, and Branches pages so you can easily check the build status of a branch when deciding whether to merge change



| Build | Status | Updated | Logs | Artifacts |
|---|-----------------------------------|-----------------------------------|------|-----------|
| Master Release Build LATEST
Matthieu Di Bernardino <code>61d28114dd4</code> TIS-3254 - Fixed unit tests | In progress | 3 mins ago | | |
| Master Branch Build
Matthieu Di Bernardino <code>899a2acd03</code> TIS-3254 - Fixed unit tests | 1 failed
211 passed, 2 skipped | 13 mins ago
2 hr 26 min 46 sec | Logs | Artifacts |
| Pull Request Build
Kevin Campbell <code>dea83b0022a</code> TIS-20009: Add support for applying alternati... | 212 passed
2 skipped | 3 hours ago
2 hr 26 min 56 sec | Logs | Artifacts |
| Pull Request Build
Jennifer Evans <code>29ea5466dfd</code> Add check for null participant status and retur... | 212 passed
2 skipped | Yesterday
2 hr 23 min 36 sec | Logs | Artifacts |
| Pull Request Build
Alana Grant <code>b8b03e5f7a7</code> TIS-20389: Replace AJS.escapeHtml with lodash... | 7 failed
205 passed, 2 skipped | Yesterday
1 hr 25 min 16 sec | Logs | Artifacts |

Bitbucket Server displays the overall status of the build results. The status is 'passed' if all the different builds (for example, unit tests, functional tests, deploy to staging) have succeeded, and 'failed' if at least one run failed for any of those.

For example, when viewing the Commits tab for a Bitbucket Server project, you will see icons that indicate the status of the latest build results. The red 'fail' icon is displayed if there is at least one failed build run for the commit.

Note that the legacy Bitbucket Server notification type is deprecated. It is still available in Bamboo 5.6 but will be removed in Bamboo 5.7.

Bamboo provides support for Pull Request

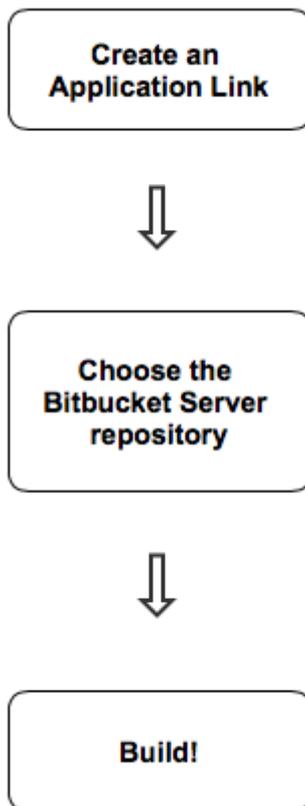
Starting from version 6.0, Bamboo can create plan branches by pull requests. Create a pull request when ready to share your work with teammates and the CI system. Bamboo will detect new pull requests and create plan branch.

Note that Bamboo doesn't provide pull request support for forked repositories yet.

Configuration

There are just a few simple configuration steps to get the integrations described above with Bamboo (versions 5.6 and later) and Bitbucket Server.

Bamboo will be automatically configured to respond to repository events published by Bitbucket Server, and to notify Bitbucket Server about build results you don't have to configure repository polling for new commits anymore in Bamboo, or set up dedicated web hooks in your Bitbucket Server instance.



1. Create an Application Link

You only need to do this once for each pair of Bitbucket Server and Bamboo instances.

See [Linking to another application](#).

Once linked, all the Bitbucket Server repositories are available to your plans in Bamboo.

2. Choose the Bitbucket Server repository for the Bamboo plan

Create a build plan (if necessary) and specify the repository in the plan (or job) configuration.

 To connect to a Bitbucket Server repository, select **Bitbucket Server / Stash** and provide the Bitbucket Server details.

You must [enable the SSH access](#) on Bitbucket Server, otherwise the integration features won't work and you will have to provide an alternative HTTP repository type to connect to the Bitbucket Server repository.

 **BAM-15464** - Provide HTTP(S) authentication method option for Bitbucket Server type repository
GATHERING INTEREST

See [Bitbucket Server](#) for more information about using Bitbucket Server source repositories in Bamboo.

3. Build!

You can also use the Bitbucket Rest API to automatically publish build status from Bamboo, Jenkins or any other build tool to Bitbucket. See the [Bitbucket developer documentation](#) to do with [updating build status](#).

Connect Bitbucket to an external database

This page provides information about using Bitbucket Data Center and Server with an external database.

Bitbucket ships with an embedded database that it uses straight out-of-the-box, with no configuration required. This is great for evaluation purposes, but for production installations, we recommend that you use one of the supported external databases.

Please refer to [Supported platforms](#) for the versions of external databases supported by Bitbucket.

If you just want to change the password for the external database, see [How do I change the external database password](#).

Instructions for connecting Bitbucket to the supported external databases:

- [Connect Bitbucket to PostgreSQL](#)
- [Connect Bitbucket to Oracle](#)
- [Connect Bitbucket to SQL Server](#)
- [Connecting Bitbucket Server to MySQL](#)

 MySQL is **not** supported for Bitbucket Data Center instances. MySQL is supported for Bitbucket Server (standalone) instances, but not recommended. See [Connecting Bitbucket Server to MySQL](#) for more information.

Why would I want to use an external database?

Bitbucket ships with an embedded database that is great for evaluation purposes, but for production installations we recommend that you make use of one of the [supported](#) external databases, for the following reasons:

- **Improved protection against data loss:** The Bitbucket Server built-in database, which runs [HSQLDB](#), is susceptible to data loss during system crashes. External databases are generally more resistant to data loss during a system crash. HSQLDB is not supported in production environments and should only be used for evaluation purposes.
- **Performance and scalability:** If you have a large number of users on your Bitbucket instance, running the database on the same server as Bitbucket may slow it down. We recommend that for large installations, Bitbucket and the DBMS are run on separate machines. When using the embedded database, the database will always be hosted and run on the same server as Bitbucket, which will limit performance.
- **Unified back-up:** Use your existing DBMS tools to back up your Bitbucket database alongside your organization's other databases.
- **Bitbucket Data Center support:** If you want to upgrade your instance to [Bitbucket Data Center resources](#), either now or in the future, to take advantage of the performance-at-scale and high availability benefits of running Bitbucket Server in clustered mode, then you **must** use an external database. HSQLDB is not supported in Bitbucket Data Center.

Using the Database Migration Wizard

 The Database Migration Wizard is not supported in Bitbucket Data Center instances while more than one cluster node is running. To migrate databases for a Bitbucket Data Center instance, you should perform the migration before starting multiple cluster nodes.

You can use the Database Migration Wizard to migrate the Bitbucket data:

- from the embedded database to a [supported](#) external DBMS.
- to another instance of the same DBMS.
- from one DBMS to another supported DBMS (for example, from MySQL to PostgreSQL).

You need to have created the DBMS (such as PostgreSQL) that you wish to migrate the Bitbucket data to before running the Migration Wizard.

To run the Database Migration Wizard:

1. Log in to Bitbucket.
2. In the administration area, click **Database** (under 'Settings').
3. Click **Migrate database** and follow the instructions for running the migration.

Notes about database migration

- **Back up the database and Bitbucket home directory:**
Before starting the database migration process you should back up your [Set the home directory](#). If you intend to migrate from one external database to another, you should also back up the existing database before proceeding. See [Data recovery and backups](#) for more information.
- **Bitbucket will be unavailable during the migration:**
Bitbucket will not be available to users during the database migration operation. In addition, running the migration when people are using Bitbucket can sometimes cause the migration to time out waiting for all activity in Bitbucket that uses the database to complete. For these reasons, we recommend that you run the database migration outside of normal usage periods.
- **Migration will usually take less than 30 minutes:**
The duration of the migration process depends on the amount of data in the Bitbucket database being migrated. For new installations, containing very little data, the migration process typically takes just a few seconds. If you have been using Bitbucket for some time, its database will contain more data, and the migration process will therefore take longer. If Bitbucket has been linked to a JIRA Software instance, and there are hundreds of thousands of commits in Bitbucket with issue keys in the commit messages, the migration may take tens of minutes.
- **We strongly recommend using a new clean database for the new Bitbucket database:**
In case of a migration failure, Bitbucket may have partially populated the target database. If the target database is new (therefore empty) and set aside for Bitbucket's exclusive use, it's very easy to clean up after a failed migration; just drop the target database and use a clean target database instance for the next attempt.
- **Ensure your [Set the home directory](#) is secured against unauthorized access:**
 - After the migration, the connection details (including the username and password) for the database are stored in the `bitbucket.properties` file.
 - The migration will create a dump file of the contents of your database in the [Bitbucket Server home export](#) directory. This is used during the migration and is kept for diagnostic purposes in the case of an error. You may remove this after migration but it may reduce Atlassian Support's ability to help you in the case of migration issues.
 - You can [edit the database password](#) if needed after migration.

Connecting Bitbucket Server to MySQL

This page describes how to connect Bitbucket Server to a MySQL or MariaDB database. The procedure for MySQL and MariaDB is the same, except where noted below. See [Connect Bitbucket to an external database](#) for general information.

MySQL / MariaDB performance issues

MySQL and MariaDB, while supported by Bitbucket Server, are currently *not* recommended especially in larger instances, due to inherent performance and deadlock issues that occur in this database engine under heavy load.

Affected systems may experience slow response times, deadlock errors and in extreme cases errors due to running out of database connections. These issues are intrinsic to MySQL and MariaDB (no other database engine in Bitbucket Server's [Supported platforms](#) shares this behavior) and are due to the way MySQL and MariaDB perform row-level locking in transactions. See <http://dev.mysql.com/doc/refman/5.0/en/innodb-deadlocks.html> for some general information on this.

Bitbucket Server does its best to work around the MySQL / MariaDB behavior - see issues [STASH-4517](#), [STASH-4701](#) and others for example. However, under very heavy load you will generally get better performance with any of the other database engines supported by Bitbucket Server (such as PostgreSQL which is also freely available) than you will with MySQL or MariaDB. Please see [Connect Bitbucket to an external database](#) for instructions on migrating your data to one of these other engines.

MySQL and MariaDB are not supported in Bitbucket Data Center

Bitbucket Data Center does not support any version of MySQL or MariaDB. With Bitbucket Data Center you must use one of the other database engines [supported](#) by Bitbucket Server (such as PostgreSQL which is also freely available). Please see [Connect Bitbucket to an external database](#) for instructions on migrating your data to one of these other engines.

MySQL 5.6.x compatibility

Note that Bitbucket Server is not compatible at all with versions of MySQL 5.6 earlier than 5.6.16 because of bugs in its query optimizer ([#68424](#), [#69005](#)). Please watch [STASH-3164](#) for further updates on this. Bitbucket Server does support versions of MySQL 5.6 from 5.6.16 on.

JDBC driver support

Bitbucket Server only supports MySQL JDBC version 5.1.

See [Supported platforms](#) for the versions of MySQL and MariaDB supported by Bitbucket Server.

The overall process for using a MySQL or MariaDB database with Bitbucket Server is:

1. Install MySQL or MariaDB where it is accessible to Bitbucket Server. It is assumed here that you already have MySQL or MariaDB installed and running. See the MySQL documentation at <http://dev.mysql.com/doc/>.
2. [Create the database and user](#) on the MySQL / MariaDB server for Bitbucket Server to use.
3. [Download and install the JDBC driver](#).
4. [Migrate Bitbucket Server to the MySQL / MariaDB database](#).

Create the Bitbucket Server database

Before you can use Bitbucket Server with MySQL or MariaDB, you must set up the MySQL or MariaDB server as follows:

Create database

Create a database on MySQL or MariaDB for Bitbucket Server to use.

Create database user

Create a Bitbucket Server user on the database.

Character encoding

Configure the database to use `utf8` character set encoding.

Note that Bitbucket Server on MySQL and MariaDB does not support 4 byte UTF-8 characters.

Collation

Configure the database to use `utf8_bin` collation (to ensure case sensitivity).

Logging format

If MySQL or MariaDB is using binary logging, configure the database to use a binary logging format of either `MIXED` or `ROW`.

Refer to the [MySQL documentation](#). Note that Bitbucket Server sets the MySQL / MariaDB transaction isolation level to `READ-COMMITTED` when it connects to the database.



Packages of MySQL or MariaDB in some Linux distributions may be configured with `binlog_format=statement` by default. Before using such packages with Bitbucket Server you must change this to either `mixed` or `row`. See [this KB article](#) for more information.

Connection timeout

Bitbucket Server requires the database to keep idle connections alive for at least 10 minutes.

If the database is configured with less than a 10 minute connection timeout, there will be [seemingly random connection errors](#).

Here is an example of how to do that. When Bitbucket Server and MySQL / MariaDB run on the same physical computer (accessible through `localhost`), run the following commands (replacing `bitbucketuser` and `password` with your own values):

```
mysql> CREATE DATABASE bitbucket CHARACTER SET utf8 COLLATE utf8_bin;
mysql> CREATE USER 'bitbucketuser'@'localhost' IDENTIFIED BY 'password';
mysql> GRANT ALL PRIVILEGES ON bitbucket.* TO 'bitbucketuser'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> QUIT
```

This creates an empty MySQL / MariaDB database with the name `bitbucket`, and a user that can log in from the host that Bitbucket Server is running on who has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

If the MySQL / MariaDB database and Bitbucket Server instances are on the same physical computer, you can use `localhost` and *not set a password* by omitting `IDENTIFIED BY 'password'` from the 2nd MySQL statement above (if you trust the security *within* this computer).

If the MySQL / MariaDB database and Bitbucket Server instances are on different computers, just replace the `localhost` part of the `GRANT ALL` statement above with the hostname of the machine that Bitbucket Server is running on. See the documentation at <http://dev.mysql.com/doc/refman/5.1/en/account-names.html>.

Note that Bitbucket Server will generally require about 2530 connections to the database. The maximum number of connections is a configurable system property see [Database pool](#).

Download and install the JDBC driver

The JDBC drivers for MySQL / MariaDB are *not* bundled with Bitbucket Server (due to licensing restrictions). You need to download and install the driver yourself, after you have installed Bitbucket Server.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).

i The MariaDB Java Client is not compatible with Bitbucket Server

The MySQL Connector/J must be used for both MySQL and MariaDB. The MariaDB Java Client is **not** compatible with Bitbucket Server and is not supported.

2. Expand the downloaded zip/tar.gz file.
3. Copy the `mysql-connector-java-5.1.XX-bin.jar` file from the extracted directory to your `<Bitbucket home directory>/libdirectory` (for Bitbucket Server 2.1 or later).
4. Stop, and then restart Bitbucket Server. See [Start and stop Bitbucket](#).

Migrate Bitbucket Server to the MySQL / MariaDB database

You can migrate Bitbucket Server to the MySQL or MariaDB database created above, either from the embedded database or from another external database.

The migration process makes a backup of your existing Bitbucket Server database in `exports` under the Bitbucket Server [home directory](#). See [Data recovery and backups](#) for further information about backing up Bitbucket Server.

Run the migration as follows:

1. In the administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **MySQL** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

See [these notes](#) about database migration.

Migrate Database

Bitbucket will be unavailable to users while a migration is in progress. See our [documentation](#) for more information.

| | |
|--------------------|---|
| Database Type | <input type="text" value="PostgreSQL"/> |
| Hostname* | <input type="text"/> |
| | Hostname or IP address of the database server |
| Port* | <input type="text" value="5432"/> |
| | TCP port number for the database server |
| Database name* | <input type="text"/> |
| Database username* | <input type="text"/> |
| Database password | <input type="password"/> |

Hostname

The hostname or IP address of the computer running the database server.

Port

The TCP port with which Bitbucket Server can connect to the database server. The default value is the default port that MySQL or MariaDB runs against. You can change that if you know the port that your MySQL or MariaDB instance is using.

Database name

The name of the database that Bitbucket Server should connect to.

Database username

The username that Bitbucket Server should use to access the database.

Database password

The password that Bitbucket Server should use to access the database.

Connect Bitbucket to Oracle

This page describes how to connect Bitbucket Data Center and Server to an Oracle database.

The overall process for using an Oracle database with Bitbucket is:

- Install Oracle where it is accessible to Bitbucket.
- Create a database and user on the Oracle server for Bitbucket to use.
- Install Bitbucket on Windows, or on Linux or Mac. See [Getting started](#).
- Either:
 - at Bitbucket install time, run the Setup Wizard to connect Bitbucket to the Oracle database, or
 - at a later time, migrate Bitbucket to the Oracle database. See [Using the Database Migration Wizard](#).

It is assumed here that you already have Oracle installed and running. For information about installing Oracle and creating Oracle databases, see the [Oracle documentation pages](#). For the versions of Oracle supported by Bitbucket see [Supported platforms](#).

Prerequisites

Backup

If you are migrating your data from the internal Bitbucket database, back up the [home directory](#).

If you are migrating your Bitbucket data from a different external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See [Data recovery and backups](#).

Create the Bitbucket database

Before you can use Bitbucket with Oracle, you must set up Oracle as follows:

- Ensure that you have a database instance available for Bitbucket (either create a new one or use an existing one)
The character set of the database must be set to either `AL32UTF8` or `UTF8`, to support storage of Unicode data as per the [Oracle documentation](#).
Note that it is important to the proper operation of Bitbucket that the database store its data in a case-sensitive manner. By changing the values of the `NLS_COMP` and/or `NLS_SORT` variables, it is possible to cause Oracle to perform its searches in a case-insensitive manner. We therefore strongly recommend that those variables be left at their default values.
- Create a user that Bitbucket will connect as (e.g. `bitbucket`).
 - ✓ Remember the database user name; it will be used to configure Bitbucket's connection to the database in subsequent steps.
 - ⓘ When you create a user in Oracle, a schema is automatically created.
It is strongly recommended that you create a new database user for use by Bitbucket rather than sharing one that is used by other applications or people.
- Grant the Bitbucket user `connect` and `resource` roles only. The `connect` role is required to set up a connection, while `resource` role is required to allow the user to create objects in its own schema.
- Note that Bitbucket requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be [seemingly random connection errors](#).

The format of the command to create a user in Oracle is:

```
CREATE USER <user>
  IDENTIFIED BY <password>
  DEFAULT TABLESPACE USERS
  QUOTA UNLIMITED ON USERS;
GRANT CONNECT, RESOURCE to <user>;
```

Here is a simple example, using SQL*Plus, of how one might create a user called `bitbucket` with password `jdHyd6Sn21` in tablespace `users`, and grant the user a minimal set of privileges. When you run the command on your machine, remember to replace the username, password and tablespace names with your own values.

```
CREATE USER bitbucket
  IDENTIFIED BY jdHyd6Sn21
  DEFAULT TABLESPACE USERS
  QUOTA UNLIMITED ON USERS;
GRANT CONNECT, RESOURCE to bitbucket;
```

This creates an empty Oracle schema with the name `bitbucket`, and a user that can log in from the host that Bitbucket is running on and who has full access to the newly created schema. In particular, the user is allowed to create sessions and tables.

Bitbucket will generally require about 2530 connections to the database. The maximum number of connections is a configurable system property see [Database pool](#).

Connect Bitbucket to the Oracle database

You can now connect Bitbucket to the Oracle database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate to Oracle, either from the embedded Bitbucket database or from another external database.

When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **Oracle** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Bitbucket Setup Wizard.

When migrating to Oracle

1. In the Bitbucket administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **Oracle** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Migrate Database

Stash will be unavailable to users while a migration is in progress. See our [documentation](#) for more info

| | |
|--------------------|--|
| Database Type | <input type="text" value="Oracle"/> |
| Hostname* | <input type="text"/> |
| | Hostname or IP address of the database server |
| Port* | <input type="text" value="1521"/> |
| | TCP port number for the database server |
| Database name* | <input type="text"/> |
| Database username* | <input type="text"/> |
| Database password | <input type="password"/> |
| | <input type="button" value="Start Migration"/> <input type="button" value="Test"/> <input type="button" value="Cancel"/> |

Hostname

The hostname or IP address of the computer running the database server.

Port

The TCP port with which Bitbucket can connect to the database server. The default value is the default port that MySQL or MariaDB runs against. You can change that if you know the port that your MySQL or MariaDB instance is using.

Database name

The name of the database that Bitbucket should connect to.

Database username

The username that Bitbucket should use to access the database.

Database password

The password that Bitbucket should use to access the database.

Related pages:

- [Connect Bitbucket to an external database](#)
- [Connecting Bitbucket Server to MySQL](#)
- [Connect Bitbucket to PostgreSQL](#)
- [Connect Bitbucket to SQL Server](#)

Connect Bitbucket to PostgreSQL

This page describes how to connect Bitbucket Data Center and Server to a PostgreSQL database.

The overall process for using a PostgreSQL database with Bitbucket is:

- install PostgreSQL where it is accessible to Bitbucket
- create a database and user on the PostgreSQL server for Bitbucket to use
- install Bitbucket on Windows, or on Linux or Mac. See [Getting started](#).
- Either:
 - at Bitbucket install time, run the Setup Wizard to connect to the PostgreSQL database, or
 - at a later time, migrate Bitbucket to the PostgreSQL database. See [Using the Database Migration Wizard](#).

It is assumed here that you already have PostgreSQL installed and running. For more information about PostgreSQL installation and operation, refer to the [PostgreSQL documentation](#). For additional information review this page on [tuning](#).

PostgreSQL has the idea of schemas. When you create a PostgreSQL database, a 'public' schema is created and set as the default for that database. It is possible to create a different schema (e.g. 'bitbucket') and set that as the default schema. Bitbucket will use whatever schema is set as the default for the logged-in user. Bitbucket does not provide a way for a user to nominate the schema to use; it uses a schema that is set as the PostgreSQL default.

See [Supported platforms](#) for the versions of PostgreSQL supported by Bitbucket.

Prerequisites

Backup

If you are migrating your Bitbucket data from the HSQL internal database, back up the [home directory](#).

If you are migrating your Bitbucket data from another external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See [Data recovery and backups](#).

Create the Bitbucket database

Before you can use Bitbucket with PostgreSQL, you must:

- Create a role for Bitbucket to use when it connects to the database. We strongly recommend that this role be established for Bitbucket's use exclusively; it should not be shared by other applications or people.
- Create a database in which Bitbucket can store its data. The database must be configured to use the UTF-8 character set. During normal operation, Bitbucket will acquire 2530 connections to the database. The maximum number of connections is a configurable system property see [Database pool](#).
- Note that Bitbucket requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10-minute connection timeout, there will be [seemingly random connection errors](#).

Here is an example of how to create a user called `bitbucketuser` with password `jellyfish`, and a database called `bitbucket`, which is configured for use by `bitbucketuser`. Using a PostgreSQL client application like `psql` or `pgAdmin`, run the following commands, replacing the user name, password, and database name with your own values.

```
CREATE ROLE bitbucketuser WITH LOGIN PASSWORD 'jellyfish' VALID UNTIL 'infinity';

CREATE DATABASE bitbucket WITH ENCODING='UTF8' OWNER=bitbucketuser CONNECTION LIMIT=-1;
```

If the server that is hosting the PostgreSQL database is not the same server as Bitbucket, then please ensure that the Bitbucketserver can connect to the database server. Please also refer to the [PostgreSQL documentation on how to set up pg_hba.conf](#). If the `pg_hba.conf` file is not set properly, remote communication to the PostgreSQL server will fail.

Connect Bitbucketto the PostgreSQL database

You can now connect Bitbucketto the PostgreSQL database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate to PostgreSQL, either from the embedded HSQL database or from another external database.

When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **PostgreSQL** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow theinstructions in the BitbucketSetup Wizard.

When migrating to PostgreSQL

1. In the Bitbucketadministration area, click**Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **PostgreSQL** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Migrate Database

Stash will be unavailable to users while a migration is in progress. See our [documentation](#) for more infori

| | |
|--------------------|--|
| Database Type | <input type="text" value="PostgreSQL"/> |
| Hostname* | <input type="text"/> |
| | Hostname or IP address of the database server |
| Port* | <input type="text" value="5432"/> |
| | TCP port number for the database server |
| Database name* | <input type="text"/> |
| Database username* | <input type="text"/> |
| Database password | <input type="password"/> |
| | <input type="button" value="Start Migration"/> <input type="button" value="Test"/> <input type="button" value="Cancel"/> |

Hostname

The hostname or IP address of the computer running the database server.

Port

The TCP port with which Bitbucket can connect to the database server. The default value is the default port that MySQL or MariaDB runs against. You can change that if you know the port that your MySQL or MariaDB instance is using.

Database name

The name of the database that Bitbucket should connect to.

Databaseusername

The username that Bitbucket should use to access the database.

Databasepassword

The password that Bitbucket should use to access the database.

Related pages:

- [Connect Bitbucket to an external database](#)
- [Connecting Bitbucket Server to MySQL](#)
- [Connect Bitbucket to Oracle](#)
- [Connect Bitbucket to SQL Server](#)

Connect Bitbucket to SQL Server

This page describes how to connect Bitbucket Data Center and Serverto a Microsoft SQL Server database. You can connect Bitbucket to a SQL server either at install time using the Setup Wizard, or later using the [Using the Database Migration Wizard](#). These instructions assume that you already have SQL Server installed and running.

The overall process for using a SQL Server database with Bitbucket is:

- install SQL Server where it is accessible to Bitbucket
- create a database and user on the SQL Server instance for Bitbucket to use
- install Bitbucket on Windows, or on Linux or Mac. See the [Bitbucket installation guide](#)

See [Supported platforms](#) for the versions of SQL Server supported by Bitbucket.

Prerequisites

Back up your current database

If you are migrating your data from the internal Bitbucket database, back up the [home directory](#).

If you are migrating your Bitbucket data from a different external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See [Data recovery and backups](#).

Create the Bitbucket database

Before you can use Bitbucket with SQL Server, you must set up SQL Server as follows:

| Step | Notes |
|-------------------------|--|
| Create a database | e.g. <code>bitbucket</code> . Remember this database name for the connection step below. |
| Set the collation type | This should be case-sensitive, for example, 'SQL_Latin1_General_CP1_CS_AS' (CS = Case Sensitive). |
| Set the isolation level | Configure the database to use the isolation level, Read Committed with Row Versioning. |
| Create a database user | e.g. <code>bitbucketuser</code> . This database user should not be the database owner, but should be in the <code>db_owner</code> role. It needs to be in this role during setup <i>and</i> at all points when Bitbucket is running due to the way it interacts with the database. See SQL Server Startup Errors . Remember this database user name for the connection step below. |

On this page:

- [Prerequisites](#)
- [Connect Bitbucket to the SQL Server database](#)
- [Use Integrated Authentication or 'Windows Authentication Mode' \(Optional\)](#)
- [Install the JDBC driver](#)

Related pages:

- [Transitioning from jTDS to Microsoft's JDBC driver](#)
- [Connect Bitbucket to an external database](#)
- [SQL Server documentation](#)
- [JDBC documentation](#)
- [Supported platforms - External Databases: SQL](#)

| | |
|-------------------------------|--|
| Set database user permissions | The Bitbucket database user has permission to connect to the database, and to create and drop tables, indexes and other constraints, and insert and delete data, in the newly-created database. |
| Enable TCP/IP | Ensure that TCP/IP is enabled on SQL Server and that SQL Server is listening on the correct port (which is 1433 for a default SQL Server installation). Remember this port number for the connection step below. |
| Check the authentication mode | Ensure that SQL Server is operating in the appropriate authentication mode. By default, SQL Server operates in 'Windows Authentication Mode'. However, if your user is not associated with a trusted SQL connection, 'Microsoft SQL Server, Error: 18452' is received during Bitbucket startup, and you will need to change the authentication mode to 'Mixed Authentication Mode'.

<i>Bitbucket instances running on Windows are also able to support SQL Server databases running in 'Windows Authentication Mode'. This is described at the bottom of this page and it has to be manually configured: Use Integrated Authentication (Optional)</i> |
| Check that SET NOCOUNT is off | Ensure that the SET NOCOUNT option is turned off. You can do that in SQL Server Management Studio as follows:

<ol style="list-style-type: none"> 1. Navigate to Tools > Options > Query Execution > SQL Server > Advanced. Ensure that the SET NOCOUNT option is cleared. 2. Now, go to the Server > Properties > Connections > Default Connections properties box and clear the no count option. |

Note that Bitbucket will generally require about 2530 connections to the database.

Note also that Bitbucket requires the database to keep idle connections alive for at least 10 minutes. If the database is configured with less than a 10 minute connection timeout, there will be [seemingly random connection errors](#).

Here is an example of how to create and configure the SQL Server database from the command line. When Bitbucket and SQL Server run on the same physical computer (accessible through `localhost`), run the following commands (replacing `bitbucketuser` and `password` with your own values):

```
SQL Server> CREATE DATABASE bitbucket
SQL Server> GO
SQL Server> USE bitbucket
SQL Server> GO
SQL Server> ALTER DATABASE bitbucket SET ALLOW_SNAPSHOT_ISOLATION ON
SQL Server> GO
SQL Server> ALTER DATABASE bitbucket SET READ_COMMITTED_SNAPSHOT ON
SQL Server> GO
SQL Server> ALTER DATABASE bitbucket COLLATE SQL_Latin1_General_CP1_CS_AS
SQL Server> GO
SQL Server> SET NOCOUNT OFF
SQL Server> GO
SQL Server> USE master
SQL Server> GO
SQL Server> CREATE LOGIN bitbucketuser WITH PASSWORD=N'password', DEFAULT_DATABASE=bitbucket,
CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
SQL Server> GO
SQL Server> ALTER AUTHORIZATION ON DATABASE::bitbucket TO bitbucketuser
SQL Server> GO
```

This creates an empty SQL Server database with the name `bitbucket`, and a user that can log in from the host that Bitbucket is running on who has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

Connect Bitbucket to the SQL Server database

You can now connect Bitbucket to the SQL Server database, either:

- when you run the Setup Wizard, at install time,
- when you wish to migrate to SQL Server, either from the embedded database or from another external database.

When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **SQL Server** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Bitbucket Setup Wizard.

When migrating to SQL Server

1. In the Bitbucket administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **SQL Server** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Migrate Database

Bitbucket will be unavailable to users while a migration is in progress. See our [documentation](#) for more

Database Type

Hostname*

Hostname or IP address of the database server

Port*

TCP port number for the database server

Database name*

Database username*

Database password

| | |
|----------------------|---|
| Hostna
me | The hostname or IP address of the computer running the database server. |
|----------------------|---|

| | |
|--------------------------|--|
| Port | The TCP port with which Bitbucket can connect to the database server. The default value of 1433 is the default port that SQL Server runs against. You can change that if you know the port that your SQL Server instance is using. |
| Database name | The name of the database that Bitbucket should connect to. |
| Database username | The username that Bitbucket should use to access the database. |
| Database password | The password that Bitbucket should use to access the database. |

Named Instances

If you have a named instance on your server, you will need to manually edit the `bitbucket.properties` file as described on the [Connecting to named instances in SQL Server from Bitbucket Server](#) Knowledge Base article.

Use Integrated Authentication or 'Windows Authentication Mode'(Optional)

Windows authentication is only available for Bitbucket instances running on Windows. It cannot be used on Linux because Microsoft does not provide shared objects for it. You will either need to run Bitbucket on Windows, allowing you to use Windows security, or you will need to enable mixed-mode authentication for SQL Server if you are running Bitbucket on Linux. Unfortunately, there are no other options at this time.

Integrated authentication uses a native DLL to access the credentials of the logged-in user to authenticate with SQL Server. The native DLLs for both 32- and 64-bit systems are included in the distribution; there is no need to download the entire package from Microsoft.

Bitbucket does not currently support configuring the system to use integrated authentication from the UI (Vote for it!)

 **BSERV-3035** - Add support for integrated authentication for Microsoft SQL Server GATHERING INTEREST

). This means you can't currently migrate to SQL Server with integrated authentication, nor can you configure Bitbucket to use SQL Server with integrated authentication during initial setup. However, if Bitbucket has already been configured to use SQL Server (for example, when the Setup Wizard was run at first use), you can enable integrated authentication by directly modifying Bitbucket's configuration, as follows:

1. Based on the JVM being used to run Bitbucket, rename either the `x64` or `x86` DLL to `sqljdbc_auth.dll` in `lib/native`. Note that running on Windows x64 does *not* require the use of the `x64` DLL; you should only use the `x64` DLL if you are also using a 64-bit JVM.
2. In `start-webapp.bat`, a `JVM_LIBRARY_PATH` variable has already been defined. Simply remove the leading `rem`. Note that if you are putting the native DLL in an alternative location, you may need to change the value to point to your own path. The value of the `JVM_LIBRARY_PATH` variable will automatically be included in the command line when Tomcat is run using `start-bitbucket.bat`.
3. Edit the `%BITBUCKET_HOME%\shared\bitbucket.properties` file to include `integratedSecurity=true` in the `jdbc.url` line. Note that `jdbc.user` and `jdbc.password` will no longer be used to supply credentials *but they must still be defined* Bitbucket will fail to start if these properties are removed.
4. Ensure the Bitbucket process or service is running as the correct user to access SQL Server. (Note that this user is generally a Windows Domain User Account, but should not be a member of any administrators groups, that is local, domain, or enterprise.)

 It is also possible to configure integrated authentication over Kerberos, rather than using the native DLLs. Details for that are included in the [JDBC documentation](#).

Install the JDBC driver

This section is only relevant to some distributions of Bitbucket, for example if you are running Bitbucket via the Atlassian Plugin SDK, or have built Bitbucket from source.

If the SQL Server JDBC driver is *not* bundled with Bitbucket, you will need to download and install the driver yourself.

1. Download the appropriate JDBC driver from the [Microsoft download site](#).
2. Install the driver file to your `<Bitbucket home directory> /libdirectory` (for Bitbucket 2.1 or later).
3. Stop, then restart, Bitbucket. See [Start and stop Bitbucket](#).

If Bitbucket was configured to use Microsoft SQL Server by manually entering a JDBC URL, please refer to [this guide](#).

Transitioning from jTDS to Microsoft's JDBC driver

This page describes how to change from using jTDS to using the Microsoft SQL Server JDBC driver to access Microsoft SQL Server.

What do I have to do?

If Bitbucket Data Center and Server was configured to use Microsoft SQL Server by following the steps outlined in [Connect Bitbucket to SQL Server](#), *no change is necessary*. However, if Bitbucket was configured to use Microsoft SQL Server by **manually entering a JDBC URL**, the system will lock on startup if the driver class and URL are not manually updated.

How to proceed

In the [home directory](#), `bitbucket.properties` must be edited to change the JDBC driver and URL. The existing configuration should look similar to this:

```
jdbc.driver=net.sourceforge.jtds.jdbc.Driver
jdbc.url=jdbc:jtds:sqlserver://localhost:1433;databaseName=stash;
jdbc.user=stashuser
jdbc.password=secretpassword
```

⚠ The JDBC URL above is in the format constructed by Bitbucket when connecting Bitbucket to SQL Server and will automatically be updated to a URL compatible with Microsoft's driver, with no change required on the administrator's part. If the URL contains additional properties, such as `domain=`, it will need to be manually updated.

To use Microsoft's SQL Server driver, the settings above would be updated to this:

```
jdbc.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
jdbc.url=jdbc:sqlserver://localhost:1433;databaseName=stash;
jdbc.user=stashuser
jdbc.password=secretpassword
```

The exact values to use in the new URL are beyond the scope of this documentation; they must be chosen based on the jTDS settings they are replacing.

Additional Information

The new JDBC driver class is: `com.microsoft.sqlserver.jdbc.SQLServerDriver`

The JDBC URL format for the jTDS driver is documented on SourceForge at <http://jtds.sourceforge.net/faq.html#urlFormat>.

The JDBC URL format for Microsoft's SQL Server driver is documented on MSDN at <http://msdn.microsoft.com/en-us/library/ms378428.aspx>, with documentation for additional properties at <http://msdn.microsoft.com/en-us/library/ms378988.aspx>.

Why change drivers?

Recent releases of Hibernate, which Bitbucket uses to simplify its persistence layer, have introduced a requirement that the JDBC drivers and connection pools used be JDBC4-compliant. JDBC4 was introduced with Java 6.

The jTDS driver used by releases prior to Bitbucket 2.1 is a JDBC3 driver, compatible with Java 1.3, and therefore cannot be used with newer versions of Hibernate. While jTDS 1.3.0 and 1.3.1 *claim* to implement JDBC4, and JDBC4.1, they actually don't. The new methods have been "implemented", but their implementations all throw new `AbstractMethodError()`, which means they can't actually be *used*. (See an example [here](#), on [GitHub](#).)

Since jTDS 1.3.1 does not provide a functioning JDBC4 implementation, the decision was made to replace jTDS with Microsoft's own SQL Server driver. Microsoft's driver is actively maintained, where jTDS hasn't been updated since 2014 (and prior to the small round of updates done in 2014 it hadn't been updated for multiple years). Microsoft offers a full JDBC4.2 (Java 8) driver and supports all the features of SQL Server, including SQL Server 2016.

Bitbucket attempts to automatically update jTDS JDBC URLs to values compatible with Microsoft's JDBC driver. However, for installations using custom JDBC URLs for example, to use domain authentication such automatic updating is not possible; the URL, which was manually entered, must be manually updated.

Configuring Bitbucket Data Center to work with Amazon Aurora

These instructions will help you connect Bitbucket Data Center to an *existing* Amazon Aurora PostgreSQL database.

⚠ Amazon Aurora is only supported on a Data Center license

Bitbucket Data Center supports the use of a single-writer, PostgreSQL-compatible Amazon Aurora clustered database. A typical production-grade cluster includes one or more readers in a different availability zone. If the writer fails, Amazon Aurora will automatically promote one of the readers to take its place. For more information, see [Amazon Aurora Features: PostgreSQL-Compatible Edition](#).

The overall process for using a PostgreSQL-compatible Amazon Aurora database with an existing Bitbucket deployment is:

1. Deploy Amazon Aurora (preferably, through the [Modular Architecture for Amazon Aurora PostgreSQL Quick Start](#)).
2. Configure PostgreSQL on the Amazon Aurora database.
3. Connect your Bitbucket deployment to the Amazon Aurora database.

See [Supported platforms](#) for the versions and configurations of Amazon Aurora supported by Bitbucket Data Center.

Backups (for migrations)

If you are migrating Bitbucket Server data from the HSQL internal database, back up the [Set the home directory](#).

If you are migrating Bitbucket Server data from another external database, back up that database by following the instructions provided by the database vendor before proceeding with these instructions.

See [Data recovery and backups](#).

1. Deploy Amazon Aurora

✅ Use the Quick Start for new deployments

If you are deploying a new Bitbucket Data Center with Amazon Aurora from scratch, we recommend that you use the [AWS Quick Start for Bitbucket](#). This Quick Start lets you configure a PostgreSQL-compatible Amazon Aurora cluster with one writer and two readers (preferably in separate availability zones). See [Deploy Bitbucket Data Center in AWS](#) for more information.

Bitbucket Data Center specifically supports the use of an Amazon Aurora cluster with the following configuration:

- It must have only one writer, replicating to one or more readers.
- Your PostgreSQL engine must be version 9.6 or higher.

See [Supported platforms](#) for more details.

AWS documentation

AWS has some helpful guides for setting up an Aurora database and migrating to it:

- [Modular Architecture for Amazon Aurora PostgreSQL](#): a Quick Start that guides you through the deployment of a PostgreSQL-compatible Aurora Database cluster. This cluster is similar to the one set up by the [AWS Quick Start for Jira](#) (one writer and two readers, preferably in different availability zones).
- [Upgrading the PostgreSQL DB Engine for Amazon RDS](#): shows you how upgrade your database engine to a supported version before migrating it to Amazon Aurora.

- [Migrating Data to Amazon Aurora PostgreSQL](#): contains instructions for migrating from Amazon RDS to a PostgreSQL-compatible Amazon Aurora cluster.
- [Best Practices with Amazon Aurora PostgreSQL](#): contains additional information about best practices and options for migrating data to a PostgreSQL-compatible Amazon Aurora cluster.

Amazon also offers an [AWS Database Migration Service](#) to facilitate a managed migration. This service offers minimal downtime, and supports migrations to Aurora from a wide variety of source databases.

i When migrating from a non-clustered PostgreSQL database into an Amazon Aurora database, we recommend you use `pg_dump` and `pg_restore`. For more information, see [Backup and Restore](#) (from the PostgreSQL documentation).

2. Connect Bitbucket to the Amazon Aurora database

You can now connect Bitbucket to the PostgreSQL database, when you either:

- Run the Setup Wizard (at install time)
- Want to migrate Bitbucket to PostgreSQL (either from the embedded HSQL database or from another external database)

When running the Setup Wizard at install time

1. Select **External** at the 'Database' step.
2. Select **PostgreSQL** for **Database Type**.
3. Complete the form. See the table below for details.
4. Click **Next**, and follow the instructions in the Setup Wizard.

When migrating to Amazon Aurora

1. In the Bitbucket administration area, click **Database** (under 'Settings').
2. Click **Migrate database**.
3. Select **PostgreSQL** for **Database Type**.
4. Complete the form. See the table below for details.
5. Click **Start Migration**.

Migrate Database

Stash will be unavailable to users while a migration is in progress. See our [documentation](#) for more information.

| | |
|--------------------|--|
| Database Type | <input type="text" value="PostgreSQL"/> |
| Hostname* | <input type="text"/> |
| | Hostname or IP address of the database server |
| Port* | <input type="text" value="5432"/> |
| | TCP port number for the database server |
| Database name* | <input type="text"/> |
| Database username* | <input type="text"/> |
| Database password | <input type="password"/> |
| | <input type="button" value="Start Migration"/> <input type="button" value="Test"/> <input type="button" value="Cancel"/> |

Hostname

The hostname or IP address of your Amazon Aurora database.

Port

The TCP port with which Bitbucket can connect to the database server.

Database name

The name of your Amazon Aurora database.

Databaseusername

The username that Bitbucket should use to access the database.

Databasepassword

The password that Bitbucket should use to access the database.

Related pages:

- [Connect Bitbucket to an external database](#)
- [Connecting Bitbucket Server to MySQL](#)
- [Connect Bitbucket to Oracle](#)
- [Connect Bitbucket to SQL Server](#)

Migrating Bitbucket Server to another server

This page describes how to move your Bitbucket Server installation from one physical machine to a different machine. For most scenarios, the overall procedure involves the following 4 steps, although your situation may not require all of these:

1. Prepare for the migration.
2. Move the Bitbucket Server data.
3. Move the Bitbucket Server installation to the new location, and update the value of the `BITBUCKET_HOME` environment variable.
4. Update the Bitbucket Server `bitbucket.properties` file. This will be necessary if you were unable to use the Migration Wizard in Step 2.

See also the [Bitbucket Server upgrade guide](#). You can upgrade Bitbucket Server either before or after you migrate Bitbucket Server. This page *does not* describe any aspect of the upgrade procedure.

1. Prepare for the migration

In preparation for migrating Bitbucket Server to another server, check that you have done the following:

- Confirm that the operating system, database, other applicable platforms and hardware on the new machine will comply with the [requirements](#) for Bitbucket Server.
- Check for any known migration issues in the [Bitbucket Server Knowledge Base](#).
- Alert users to the forthcoming Bitbucket Server service outage.
- Ensure that users will not be able to update existing Bitbucket Server data during the migration. You can do this by temporarily changing the access permissions for Bitbucket Server.
- Make sure you have [created a user in Bitbucket Server](#) (not in your external user directory) that has System Administrator [global permissions](#) so as to avoid being locked out of Bitbucket Server in case the new server does not have access to your external user directory.

2. Move the Bitbucket Server data to a different machine

This section gives a brief overview of how to move the Bitbucket Server data to a different machine. You do not need to do anything in this section if you will continue to use the embedded database - the Bitbucket Server data is moved when you move the Bitbucket Server installation.

The Bitbucket Server data includes the data directories (including the Git repositories), log files, installed plugins, temporary files and caches.

You can move the Bitbucket Server data:

- from the embedded database to a [supported](#) external DBMS.
- to another instance of the same DBMS.
- from one DBMS to another supported DBMS (for example, from MySQL to PostgreSQL).

You can also move the actual DBMS. Atlassian recommends that for large installations, Bitbucket Server and the DBMS run on separate machines.

There are 2 steps:

1. Create and configure the DBMS in the new location. Please refer to [Connect Bitbucket to an external database](#), and the relevant child page, for more information.
2. Either:
 - If the new location is currently visible to Bitbucket Server, use the Bitbucket Server Database Migration Wizard. Please refer to [Connect Bitbucket to an external database](#), and the relevant child page, for more information.

On this page:

- [1. Prepare for the migration](#)
- [2. Move the Bitbucket Server data to a different machine](#)
- [3. Move Bitbucket Server to a different machine](#)
- [4. Update the Bitbucket Server configuration](#)

Related pages:

- [Supported platforms](#)
- [Connect Bitbucket to an external database](#)
- [Bitbucket Server upgrade guide](#)

- If the new location is not currently visible to Bitbucket Server (perhaps because you are moving to a new hosting provider), you need to perform a database export and then import the backup to the new DBMS. Please refer to the vendor documentation for your DBMS for detailed information.
You will also need to update the `bitbucket.properties` file in the `<Bitbucket home directory>` as described below.

3. Move Bitbucket Server to a different machine

This section describes moving the Bitbucket Server installation to a different machine.

1. Stop Bitbucket Server. See [Start and stop Bitbucket](#).
2. Make an archive (such as a zip file) of the Bitbucket home directory. The home directory contains data directories (including the Git repositories), log files, installed plugins, SSH fingerprints, temporary files and caches. The home directory location is defined:
 - on Windows, by the `BITBUCKET_HOME` environment variable, or by the `BITBUCKET_HOME` line of `<Bitbucket Server installation directory>/bin/set-bitbucket-home.bat`.
 - on Linux and Mac, by the `BITBUCKET_HOME` line of `<Bitbucket Server installation directory>/bin/set-bitbucket-home.sh`.
3. Copy the archive of the Bitbucket home directory to the new machine and unzip it to its new location there.
 - For production environments the Bitbucket Server home directory should be secured against unauthorized access. See [Bitbucket home directory](#).
 - When moving the Bitbucket Server home directory from Windows to Linux or Mac, make sure that the files within `<Bitbucket home directory>/git-hooks` and `<Bitbucket home directory>/shared/data/repositories/<repoID>/hooks` directories have the executable file permission set.
4. Set up an instance of Bitbucket Server in the new location by doing one of the following:
 - Make an archive of the old Bitbucket Server installation directory and copy it across to the new machine.
 - Install the same version of Bitbucket Server from scratch on the new machine.
5. Redefine the value for `BITBUCKET_HOME`, mentioned in Step 2. above, in the new `<Bitbucket Server installation directory>`, using the new location for your copied home directory. See [Set the home directory](#) for more information.
6. If you are continuing to use the Bitbucket Server embedded database, or you used the Migration Wizard to move the Bitbucket Server data, you should now be able to start Bitbucket Server on the new machine and have all your data available. See [Start and stop Bitbucket](#). Once you have confirmed that the new installation of Bitbucket Server is working correctly, revert the access permissions for Bitbucket Server to their original values.
7. If you moved the Bitbucket Server data by performing a database export and import, carry on to Step 4. below to update the `bitbucket.properties` file in the `<Bitbucket home directory>`.

Using rsync to move your installation

Another way to move your Bitbucket Server installation is to use `rsync` (learn more at ss64.com). This option may be helpful if you want to avoid the system downtime that using an archive file will require.

If you use `rsync`, make sure to use the `--delete` option. Without it, if you run `rsync` more than once, files removed from the source machine will not be removed from the target. As a result, the two filesystems will not match, and this can lead to problems with your installation.

4. Update the Bitbucket Server configuration

If you moved the Bitbucket Server data by performing a database export, you must update the `bitbucket.properties` file within `<Bitbucket home directory>/shared` with the changed configuration parameters for the database connection.

The configuration parameters are described in [Configuration properties](#).

Once the configuration parameters are updated, you should be able to start Bitbucket Server on the new machine and have all your data available. See [Start and stop Bitbucket](#). Once you have confirmed that the new installation of Bitbucket Server is working correctly, revert the access permissions for Bitbucket Server to their original values.

Run Bitbucket in AWS

This page provides an overview of the options available for running self-managed Bitbucket Data Center and Bitbucket Server instances on Amazon Web Services.

Running Bitbucket on [Amazon Web Services \(AWS\)](#) gives you scalable computing capacity without the need to invest in hardware up front, while retaining control over where and how your code is hosted within your organization.

To this end, Atlassian provides:

- a reference deployment in the form of an AWS Quick Start, which launches, configures, and runs Bitbucket Data Center and required services in a matter of minutes, using AWS best practices for security and availability
- an Amazon Machine Image (AMI) that can be used for running Bitbucket on EC2 as an application server building block in more heavily customised deployments
- tools and guidelines for manually deploying, backing up, restoring, sizing, and administering Bitbucket Server and Bitbucket Data Center instances on AWS

✔ This page describes deploying a *single-node* instance of Bitbucket Server on AWS. For a deployment that is better suited to the architectural principles of AWS, we recommend [deploying a clustered Bitbucket Data Center instance](#), which offers greater performance at scale, high availability, and elastic scalability.

Installing Bitbucket Server in AWS

To install Bitbucket Server in AWS, you'll need to launch the Atlassian Bitbucket Server AMI in AWS. You can do this directly from the [AWS Console](#), or by running the EC2 launch wizard. See [Launch Bitbucket in AWS manually](#) for more information.

Performance guidelines

To get the best performance out of your Bitbucket deployment in AWS, it's important to not under-provision your instance's CPU, memory, or I/O resources. See [Recommendations for running Bitbucket Server in AWS](#) for more information.

Whether you choose to deploy Bitbucket Data Center, which offers performance gains via horizontal scaling, or a single node Bitbucket Server instance, we have specific recommendations on choosing AWS EC2 and EBS settings for best performance per node.

Backing up Bitbucket in AWS

Atlassian also provides utilities suitable for backing up and restoring your Bitbucket instance using native AWS snapshots. This provides a number of advantages:

- Performance: AWS snapshots occur asynchronously resulting in shorter backup downtime for your instances
- Durability: The underlying storage of AWS snapshots is in Amazon S3, which is stored redundantly and with high durability
- Availability: AWS snapshots are available across an entire AWS region, and are available to restore even in the event of an outage affecting an entire Availability Zone (AZ)

To learn more about how to back up and restore a Bitbucket instance in AWS, see [Using Bitbucket DIY Backup in AWS](#).

Administering Bitbucket in AWS

See [Administer Bitbucket in AWS](#) for information about performing administration tasks on a Bitbucket instance within AWS, including:

- configuring variables when launching Bitbucket in AWS
- maintaining, resizing, upgrading, migrating, and customizing your Bitbucket deployment in AWS
- additional details about the components within the Bitbucket AMI

Securing Bitbucket within AWS

AWS is accessed over the public Internet, so it is important to apply appropriate security measures when running Bitbucket Server in AWS. See [Best practices for securing Bitbucket in AWS](#) for security guidance on a range of security topics, including Amazon Virtual Private Cloud (VPC), Security Groups, and SSL.

Launch Bitbucket in AWS manually

This page describes how to launch the Bitbucket Amazon Machine Image (AMI) manually, giving you complete control over the components enabled in the AMI, and over AWS-specific configuration, network and security settings. If you are looking for an automated way to launch Bitbucket we recommend deploying a multi-node Bitbucket Data Center instance, using [the AWS Quick Start guide](#).

You can launch the Atlassian Bitbucket Server AMI directly from the [AWS Console](#), or by running the EC2 launch wizard. See [Launching EC2 Instances](#) for detailed instructions.

The Bitbucket AMI

The Atlassian Bitbucket Server AMI provides a typical deployment of Bitbucket Server in AWS. It bundles all the components used in a typical Bitbucket Server deployment (reverse proxy, external database, backup tools, data volume, and temporary storage) pre-configured and ready to launch.

You can use the Atlassian Bitbucket Server AMI as a "turnkey" deployment of a Bitbucket Server instance in AWS, or use it as the starting point for customizing your own, more complex Bitbucket Server deployments.

Finding the Atlassian Bitbucket Server AMI

You can find the Atlassian Bitbucket Server AMI by clicking [Community AMIs](#) and searching for **Atlassian Bitbucket**.

Be sure to use the correct AMI ID for your specific region.

To find the latest AMI using the Amazon CLI you can use the following command:

```
aws ec2 describe-images --owners 098706035825 --filters "Name=name,Values=Atlassian Bitbucket*" --query 'sort_by(Images, &CreationDate)[-1].{ID:ImageId, "AMI Name":Name}' --output table
```

 The Bitbucket AMI is not currently available in some regions, such as `sgovcloud-us` (AWS GovCloud (US)) and `cn-north-1` (China (Beijing)).

Components of the Bitbucket Server AMI

An instance launched from the Atlassian Bitbucket Server AMI contains the following components:

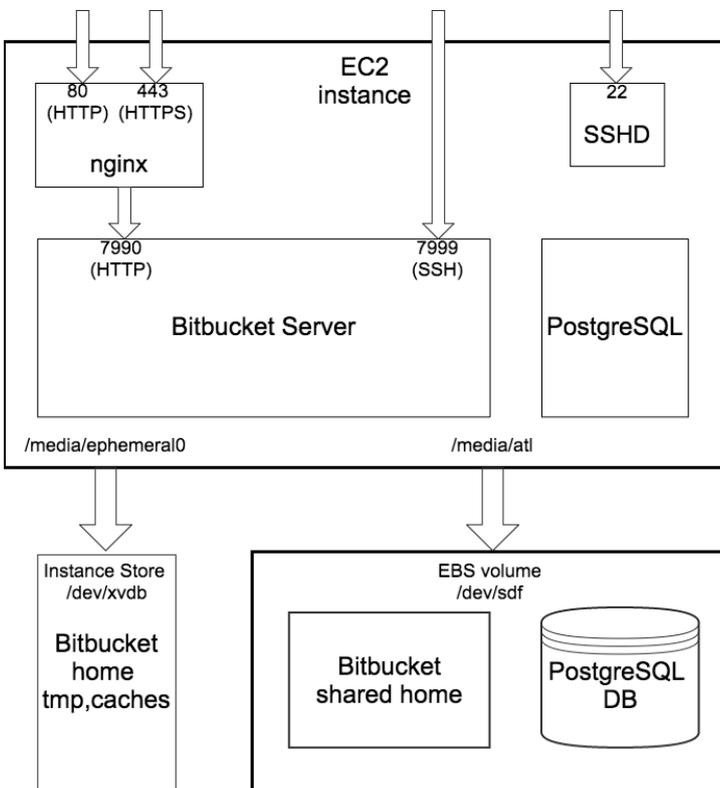
- Bitbucket Server (either the latest version or a version of your choice),
- an external PostgreSQL database,
- nginx as a reverse proxy,
- the Bitbucket Server DIY Backup utilities pre-configured for native AWS snapshots,
- an EBS Volume and Instance Store to hold the data.

| | |
|-----------------------------|---|
| Operating system | Amazon Linux 64-bit, 2016.09.0 |
| Bitbucket Server | Bitbucket Server (latest public version or a version of your choice) is downloaded and installed on launch. |
| Administrative tools | atlassian-bitbucket-diy-backup pre-installed and configured for AWS native backup, accessible over SSH. |

On this page:

- [The Bitbucket AMI](#)
- [Choosing an instance type](#)
- [Configure instance details](#)
- [Add storage](#)
- [Configure your Security Group](#)
- [What's next?](#)

| | |
|----------------------|--|
| Reverse proxy | <p>nginx, configured as follows:</p> <ul style="list-style-type: none"> • listens on port 80 and (optionally) 443, • (optionally) terminates SSL (HTTPS) and passes through plain HTTP to Bitbucket Server, • displays a static HTML page when the Bitbucket Server service is not running. |
| Database | PostgreSQL 9.3 |
| Block devices | <ol style="list-style-type: none"> 1. An EBS volume (/dev/xvdf, mounted as /media/at1), that stores: <ul style="list-style-type: none"> • the Bitbucket Server shared home directory, containing all of Bitbucket Server's repository, attachment, and other data, • PostgreSQL's data directory. 2. An EC2 Instance Store (/dev/xvdb, mounted on /media/ephemeral0) to store Bitbucket Server's temporary and cache files. |



Choosing an instance type

When choosing an EC2 instance type, see [Infrastructure recommendations for enterprise Bitbucket instances on AWS](#) for recommended instance sizing.

⚠ Minimum hardware requirements

The default t2.micro (Free tier eligible), small, and medium instance types don't meet Bitbucket Server's [minimum hardware requirements](#), and aren't supported for production deployments. See [Infrastructure recommendations for enterprise Bitbucket instances on AWS](#) for the EC2 instance types supported by Bitbucket Server.

Configure instance details

When configuring your EC2 instance there are some important details to consider:

Identity and Access Management(IAM) Role

! IAM Role must be configured at launch time

An IAM Role can only be configured for your EC2 instance **during initial launch**. You can't associate an IAM role with a running EC2 instance **after** launch. See [IAM Roles](#) for more information.

It is recommended to launch your instance with an IAM Role that allows native AWS DIY Backup to run without explicit credentials. See [IAM Roles for Amazon EC2](#) for more information.

While configuring instance details in the EC2 Launch wizard, you can create a new IAM Role by clicking **Create new IAM role**. The role should contain at least the following policy:

```
{
  "Statement": [
    {
      "Resource": [
        "*"
      ],
      "Action": [
        "ec2:AttachVolume",
        "ec2:CreateSnapshot",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVolumes",
        "ec2:DetachVolume"
      ],
      "Effect": "Allow"
    }
  ],
  "Version": "2012-10-17"
}
```

Advanced Details

The Atlassian Bitbucket Server AMI can be configured in a number of different ways at launch time:

- The built-in PostgreSQL and Nginx components (enabled by default) can be disabled
- Self-signed SSL certificate generation (disabled by default) can be enabled

You can control these options supplying User Data to your instance under **Advanced Details** in **Step 3: Configure Instance Details** of the EC2 launch wizard. All user-configurable behavior in the Atlassian Bitbucket Server AMI can be controlled by creating a file `/etc/at1` containing shell variable definitions. On first boot, the Atlassian Bitbucket Server AMI will source the file `/etc/at1` (if it exists), allowing its built-in default variable definitions to be overridden.

For example, to enable self-signed SSL certificate generation (and force all Web access to Bitbucket Server to use HTTPS), you can add User Data (**As text**) as follows:

```
#!/bin/bash
echo "ATL_SSL_SELF_CERT_ENABLED=true" >>/etc/at1
```

For a complete list of variables that can be overridden in User Data at launch time, see [Administer Bitbucket in AWS](#).

User Data is flexible and allows you to run arbitrary BASH commands on your instance at launch time, in addition to overriding variables in `/etc/at1`. See [Running Commands on Your Linux Instance at Launch](#) for more information.

⚠ Security considerations

See [Securing Bitbucket in AWS](#) for more details about enabling HTTPS and self-signed certificates in the Atlassian Bitbucket Server AMI.

Add storage

When attaching EBS volumes, use these storage device settings for your instance.

| Type | Device | Purpose | Size (GiB) | Volume Type | IOPS | Delete on Termination |
|----------------|---------------|---|------------|--|--------|-----------------------|
| Root | /dev
/xvda | Linux root volume | 50 | General Purpose (SSD) | N/A | Yes |
| EBS | /dev
/xvdf | Bitbucket Server data: repositories, attachments, avatars, etc. | 100+ | General Purpose (SSD) / Provisioned IOPS * | 300+ * | Yes |
| Instance Store | /dev
/xvdb | Bitbucket Server temporary files and caches | N/A | N/A | N/A | N/A |

* Provisioned IOPS with at least 500 1000 IOPS is recommended for instances with more than 500 active users. See [Infrastructure recommendations for enterprise Bitbucket instances on AWS](#) for more information.

The Atlassian Bitbucket Server AMI won't use any other block devices attached to the instance. The EBS volume for /dev/xvdf will be initialized and formatted at launch time, unless a snapshot id is provided (see the screen capture below), in which case it will only format it if it's not already formatted. See [Managing EBS Volumes](#) for more information about storage options in Amazon EC2.

Attach an existing EBS snapshot

You can also attach an existing EBS volume based on a snapshot during launch. To attach an existing EBS volume, within the *Device* field, change the EBS volume device to /dev/sdf and enter the Snapshot ID of the snapshot.

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

| Type | Device | Snapshot | Size (GiB) | Volume Type | IOPS | Delete on Termination | Encrypted |
|------------------|--|---------------|------------|-----------------------|------------|--------------------------|--|
| Root | /dev/sdb | snap-1b4bed30 | 10 | General Purpose (SSD) | 30 / 3000 | <input type="checkbox"/> | Not Encrypted |
| EBS | <input checked="" type="checkbox"/> /dev/sdf | snap-735d3027 | 100 | General Purpose (SSD) | 300 / 3000 | <input type="checkbox"/> | Not Encrypted <input type="checkbox"/> |
| Instance Store 0 | /dev/sdh | N/A | N/A | N/A | N/A | N/A | Not Encrypted <input type="checkbox"/> |

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

See [Administering Bitbucket Server in AWS - Moving your Bitbucket Server data volume between instances](#) for more details.

Configure your Security Group

When configuring your Security Group, you must allow allow incoming traffic to all the following ports. For more information, see [Using Security Groups](#).

| Type | Protocol | Port | Description |
|-----------------|----------|------|---|
| SSH | TCP | 22 | SSH port, allowing access to administrative functions |
| HTTP | TCP | 80 | |
| HTTPS | TCP | 443 | |
| Custom TCP Rule | TCP | 7999 | Bitbucket Server SSH port for Git hosting operations |

What's next?

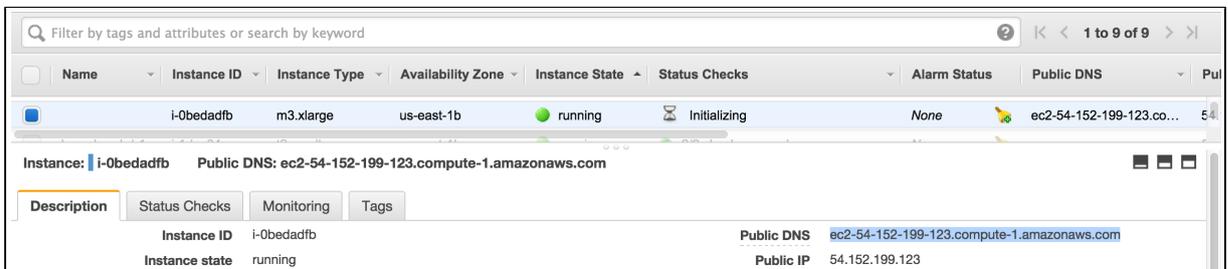
Now you're ready to configure your AWS version of Bitbucket Server.

View your new instance

Once your new EC2 instance has launched, find it within the EC2 console and navigate to the URL provided so you can continue configuring Bitbucket Server.

To find the URL of your new EC2 instance

1. From within the EC2 Console, in the Description tab of your new instance, copy the **Public DNS**.



2. Paste the URL into a browser window to view start using Bitbucket Server.

Set up your AWS deployment of Bitbucket Server

Once you've followed the URL of the EC2 instance you are presented with the [Bitbucket Server Setup Wizard](#). When you've completed the setup, you can use your instance like any other Bitbucket Server instance. So be sure to check out the rest of the [Getting Started with Bitbucket Server documentation](#).

Administer Bitbucket in AWS

This page describes administering a *single-node* instance of Bitbucket Server on AWS. For a deployment that is better suited to the architectural principles of AWS, we recommend [deploying a clustered Bitbucket Data Center instance](#), which offers greater performance at scale, high availability and elastic scalability.

This page describes the Atlassian Bitbucket Amazon Machine Image (AMI), what's inside it, how to launch it, and how to perform administration tasks in the Amazon Web Services (AWS) environment.

The Bitbucket AMI

The Atlassian Bitbucket Server AMI provides a typical deployment of Bitbucket Server in AWS. It bundles all the components used in a typical Bitbucket Server deployment (reverse proxy, external database, backup tools, data volume, and temporary storage) pre-configured and ready to launch.

You can use the Atlassian Bitbucket Server AMI as a "turnkey" deployment of a Bitbucket Server instance in AWS, or use it as the starting point for customizing your own, more complex Bitbucket Server deployments.

On this page:

- [The Bitbucket AMI](#)
- [Components of the Bitbucket Server AMI](#)
- [Launching your instance](#)
- [Connecting to your instance using SSH](#)
- [Installing an SSL certificate in your Bitbucket Server instance](#)
- [Backing up your instance](#)
- [Upgrading your instance](#)
- [Stopping and starting your EC2 instance](#)
- [Migrating your existing Bitbucket Server or Bitbucket Data Center instance into AWS](#)
- [Resizing the data volume in your Bitbucket Server instance](#)
- [Moving your Bitbucket Server data volume between instances](#)

Components of the Bitbucket Server AMI

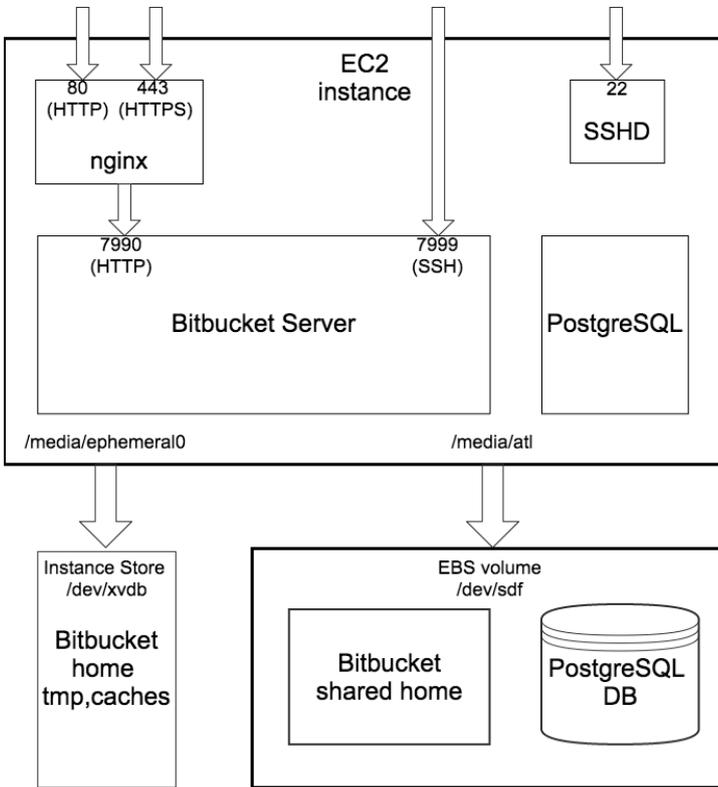
An instance launched from the Atlassian Bitbucket Server AMI contains the following components:

- Bitbucket Server (either the latest version or a version of your choice),
- an external PostgreSQL database,
- nginx as a reverse proxy,
- the Bitbucket Server DIY Backup utilities pre-configured for native AWS snapshots,
- an EBS Volume and Instance Store to hold the data.

| | |
|-----------------------------|--|
| Operating system | Amazon Linux 64-bit, 2016.09.0 |
| Bitbucket Server | Bitbucket Server (latest public version or a version of your choice) is downloaded and installed on launch. |
| Administrative tools | atlassian-bitbucket-diy-backup pre-installed and configured for AWS native backup, accessible over SSH. |
| Reverse proxy | nginx , configured as follows: <ul style="list-style-type: none">• listens on port 80 and (optionally) 443,• (optionally) terminates SSL (HTTPS) and passes through plain HTTP to Bitbucket Server,• displays a static HTML page when the Bitbucket Server service is not running. |
| Database | PostgreSQL 9.3 |

Block devices

1. An [EBS volume](#) (/dev/xvdf, mounted as /media/at1), that stores:
 - the Bitbucket Server shared home directory, containing all of Bitbucket Server's repository, attachment, and other data,
 - PostgreSQL's data directory.
2. An [EC2 Instance Store](#) (/dev/xvdb, mounted on /media/ephemeral0) to store Bitbucket Server's temporary and cache files.



Launching your instance

The Atlassian Bitbucket Server AMI can be launched by either

- using a CloudFormation template which automates creation of the associated Security Group and IAM Role. For more info, see [Quick Start with Bitbucket Server and AWS](#).
- manually using the AWS Console, which gives finer control over the optional components in the instance and AWS-specific network, security, and block device settings. For more info, see [Launch Bitbucket in AWS manually](#).

On first boot, the Atlassian Bitbucket Server AMI reads the file /etc/at1 (if any), which can override variables that enable each of the installed components. So for example to enable a self-signed SSL certificate, you can supply user data to the instance at launch time like this:

```
#!/bin/bash
echo "ATL_SSL_SELF_CERT_ENABLED=true" >>/etc/at1
```

The following variables can be configured:

| Variable name | Default value | Description |
|-------------------|---------------|---|
| ATL_NGINX_ENABLED | true | Set to false to disable the Nginx reverse proxy, and leave Bitbucket Server's server.xml configured to listen on port 7990 with no proxy. |

| | | |
|---------------------------|--------|---|
| ATL_POSTGRES_ENABLED | true | Set to <code>false</code> to disable the PostgreSQL service, and leave Bitbucket Server configured with its internal H2 database. |
| ATL_SSL_SELF_CERT_ENABLED | false | Set to <code>true</code> to enable a self-signed SSL certificate to be generated at launch time, and for Bitbucket Server's <code>server.xml</code> and Nginx's <code>nginx.conf</code> to be configured for HTTPS.

Requires <code>ATL_NGINX_ENABLED</code> also to be <code>true</code> . |
| ATL_BITBUCKET_VERSION | latest | Must be a valid Bitbucket version number (for example, <code>5.5.0</code>), or <code>latest</code> for the latest public release. |

See [Proxy and secure Bitbucket](#) for more information about Bitbucket Server's `server.xml` configuration file.

Connecting to your instance using SSH

When connecting to your instance over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

The `ec2-user` has sudo access. The Atlassian Bitbucket Server AMI does not allow SSH access by `root`.

Installing an SSL certificate in your Bitbucket Server instance

If launched with a self-signed SSL certificate (you selected **SSL Certificate > Generate a self-signed certificate** in [Quick Start with Bitbucket Server and AWS](#) or you set `ATL_SSL_SELF_CERT_ENABLED=true` in [Launch Bitbucket in AWS manually](#)), Bitbucket Server will be configured to force HTTPS and redirect all plain HTTP requests to the equivalent `https://` URL.

It's highly recommended to replace this self-signed SSL certificate with a proper one for your domain, obtained from a Certification Authority (CA), at the earliest opportunity. See [Secure Bitbucket in AWS](#). Once you have a true SSL certificate, install it as soon as possible.

To replace the self-signed SSL certificate with a true SSL certificate

1. Place your certificate file at (for example) `/etc/nginx/ssl/my-ssl.crt`
2. Place your `password-less` certificate key file at `/etc/nginx/ssl/my-ssl.key`
3. Edit `/etc/nginx/nginx.conf` as follows:
 - a. Replace references to `/etc/nginx/ssl/self-ssl.crt` with `/etc/nginx/ssl/my-ssl.crt`
 - b. Replace references to `/etc/nginx/ssl/self-ssl.key` with `/etc/nginx/ssl/my-ssl.key`
4. Append the contents of `/etc/nginx/ssl/my-ssl.crt` to the default system PKI bundle (`/etc/pki/tls/certs/ca-bundle.crt`) to ensure scripts on the instance (such as DIY backup) can `curl` successfully.
5. Restart `nginx`.

Backing up your instance

The Atlassian Bitbucket Server AMI includes a complete set of Bitbucket Server DIY Backup scripts which has been built specifically for AWS. For instructions on how to backup and restore your instance please refer to [Using Bitbucket DIY Backup in AWS](#).

Upgrading your instance

To upgrade to a later version of Bitbucket Server in AWS you first must [connect to your instance using SSH](#), then follow the steps in the [Bitbucket Server upgrade guide](#).

Stopping and starting your EC2 instance

An EC2 instance launched from the Atlassian Bitbucket Server AMI can be stopped and started just as any machine can be powered off and on again.

When stopping your EC2 instance, it is important to first

1. Stop the `atlbitbucket`, `atlbitbucket_search`, and `postgresql93` services.
2. Unmount the `/media/at1` filesystem.

If your EC2 instance becomes unavailable after stopping and restarting

When starting your EC2 instance back up again, if you rely on Amazon's automatically assigned [public IP address](#) (rather than a fixed private IP address or Elastic IP address) to access your instance, your IP address may have changed. When this happens, your instance can become inaccessible and display a "The host name for your Atlassian instance has changed" page. To fix this you need to update the hostname for your Bitbucket Server instance.

To update the hostname for your Bitbucket Server instance

1. Restart the Bitbucket service on all application nodes by running this command, which will update the hostname

```
sudo service atlbitbucket restart
```

2. Wait for Bitbucket Server to restart.
3. If you have also set up Bitbucket Server's Base URL to be the public DNS name or IP address be sure to also [update Bitbucket Server's base URL in the administration screen](#) to reflect the change.

Migrating your existing Bitbucket Server or Bitbucket Data Center instance into AWS

Migrating an existing instance to AWS involves moving consistent backups of your `/${BITBUCKET_HOME}` and your database to the AWS instance.

To migrate your existing instance into AWS

1. Check for any known migration issues in the [Bitbucket Server Knowledge Base](#).
2. Alert users to the forthcoming service outage.
3. [Create a user](#) in the Bitbucket Server Internal User Directory with `SYSADMIN` permissions to the instance so you don't get locked out if the new server is unable to connect to your User Directory.
4. Take a backup of your instance with either the [Bitbucket Server Backup Client](#) (Bitbucket Server only) or the [Bitbucket Server DIY Backup](#) (Bitbucket Server or Data Center).
5. Launch Bitbucket Server in AWS using the [Quick Start instructions](#), which uses a CloudFormation template.
6. Connect to your AWS EC2 instance with SSH and upload the backup file.
7. Restore the backup with the same tool used to generate it.
8. If necessary, update the JDBC configuration in the `/${BITBUCKET_HOME}/shared/bitbucket.properties` file.

Resizing the data volume in your Bitbucket Server instance

By default, the application data volume in an instance launched from the Atlassian Bitbucket Server AMI is a standard Linux ext4 filesystem, and can be resized using the standard Linux command line tools.

To resize the data volume in your Bitbucket Server instance

1. Stop the `atlbitbucket`, `atlbitbucket_search`, and `postgresql93` services.
2. Unmount the `/media/at1` filesystem.
3. Create a snapshot of the volume to resize.
4. Create a new volume from the snapshot with the desired size, in the same availability zone as your EC2 instance.
5. Detach the old volume and attach the newly resized volume as `/dev/sdf`.
6. Resize `/dev/sdf` using `resize2fs`, verify that its size has changed, and remount it on `/media/at1`.

7. Start the `atlbucket`, `atlbucket_search`, and `postgresql93` services.

For more information, see [Expanding the Storage Space of an EBS Volume on Linux](#), [Expanding a Linux Partition](#), and the Linux manual pages for `resize2fs` and related commands.

Moving your Bitbucket Server data volume between instances

Occasionally, you may need to move your Bitbucket Server data volume to another instance for example, when setting up staging or production instances, or when moving to an instance to a different availability zone.

There are two approaches to move your Bitbucket Server data volume to another instance

1. Take a backup of your data volume with Bitbucket Server DIY Backup, and restore it on your new instance. See [Using Bitbucket DIY Backup in AWS](#) for this option.
2. Launch a new instance from the Atlassian Bitbucket Server AMI with a snapshot of your existing data volume.

i A Bitbucket Server data volume may only be moved to a Bitbucket Server instance of the same or higher version than the original.

To launch a new instance from the Bitbucket Server AMI using a snapshot of your existing Bitbucket Server data volume

1. Stop the `atlbucket`, `atlbucket_search`, and `postgresql93` services on your existing Bitbucket Server instance.
2. Unmount the `/media/atl` filesystem.
3. Create a snapshot of the Bitbucket Server data volume (the one attached to the instance as `/dev/sdf`).
4. Once the snapshot generation has completed, launch a new instance from the Atlassian Bitbucket Server AMI as described in [Launch Bitbucket in AWS manually](#). When adding storage, change the EBS volume device to `/dev/sdf` as seen below and enter the id of the created snapshot.

Step 4: Add Storage
 Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

| Type | Snapshot | Size (GiB) | Volume Type | IOPS | Delete on Termination | Encrypted |
|------------------|---------------|------------|-----------------------|------------|--------------------------|---------------|
| Root | snap-1b4bed30 | 10 | General Purpose (SSD) | 30 / 3000 | <input type="checkbox"/> | Not Encrypted |
| EBS | snap-735d3027 | 100 | General Purpose (SSD) | 300 / 3000 | <input type="checkbox"/> | Not Encrypted |
| Instance Store 0 | N/A | N/A | N/A | N/A | N/A | Not Encrypted |

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

5. If the host name (private or public) that users use to reach your Bitbucket Server instance has changed as a result of moving availability zones (or as a result of stopping an instance and starting a new one) you will need to SSH in and run


```
sudo /opt/atlassian/bin/atl-update-host-name.sh <newhostname>
```

 where `<newhostname>` is the new host name.
6. Once Bitbucket Server has restarted your new instance should be fully available.
7. If the host name has changed you should also update the JDBC URL configuration in the `bitbucket.properties` file (typically located in `/var/atlassian/application-data/bitbucket/shared/`), as well as Bitbucket Server's base URL in the [administration screen](#) to reflect this.

Recommendations for running Bitbucket in AWS

Knowing your load profile is useful for planning your instance's growth, looking for inflated metrics, or simply keeping it at a reasonable size. In [Bitbucket Data Center load profiles](#), we showed you some simple guidelines for finding out if your instance was *Small*, *Medium*, *Large*, or *XLarge*. We based these size profiles on Server and Data Center case studies, covering varying infrastructure sizes and configurations.

A single node can be adequate for most Small or Medium size deployments, especially if you don't require high availability. If you have an existing Server installation, you can still use its infrastructure when you upgrade to Data Center. Many features exclusive to Data Center (like [SAML single sign-on](#), [self-protection via rate limiting](#), and [CDN support](#)) don't require clustered infrastructure. You can start using these Data Center features by simply upgrading your Server installations license.

✔ For more information on whether clustering is right for you, check out [Atlassian Data Center architecture and infrastructure options](#).

As your load grows closer to Large or XLarge, you should routinely evaluate your infrastructure. Once your environment starts to experience performance or stability issues, consider migrating to a clustered (or cluster-ready) infrastructure. When you do, keep in mind that it may not be always clear how to do that effectively for example, adding more application nodes to a growing Medium-sized instance doesn't always improve performance (in fact, the opposite might happen).

To help you plan your infrastructure set-up or growth, we ran a series of performance tests on typical Medium, Large, and XLarge instances. We designed these tests to get useful, data-driven recommendations for your clustered deployment's application and database nodes. These recommendations can help you plan a suitable clustered environment, one that is adequate for the size of your projected content and traffic.

ⓘ Note that large repositories might influence performance.

We advise that you [monitor performance](#) on a regular basis.

Approach

We ran all tests in AWS. This allowed us to easily define and automate multiple tests, giving us a large (and fairly reliable) sample.

Each part of our test infrastructure was provisioned from a standard AWS component available to all AWS users. This allows for easy deployment of recommended configurations. You can also use [AWS Quick Starts for deploying Bitbucket Server Data Center](#).

It also means you can look up specifications in [AWS documentation](#). This helps you find equivalent components and configurations if your organization prefers a different cloud platform or bespoke clustered solution.

Some things to consider

To effectively benchmark Bitbucket on a wide range of configurations, we designed tests that could be easily set up and replicated. Accordingly, when referencing our benchmarks for your production environment, consider:

- We didn't install apps on our test instances, as we focused on finding the right configurations for the *core* product. When designing your infrastructure, you need to account for the impact of apps you want to install.
- We used RDS with default settings across all tests. This allowed us to get consistent results with minimal setup and tuning.
- Our test environment used dedicated AWS infrastructure hosted on the same subnet. This helped minimize network latency.
- We used an internal testing tool called Trikit to simulate the influx of git packets. This gave us the ability to measure git request speeds without having to measure client-side git performance. It also meant our tests didn't unpack git refs, as the tool only receives and decrypts git data.

- The performance (response times) of git operations will be affected largely by repository size. Our test repositories averaged 14.2MB in size. We presume that bigger repositories might require stronger hardware.
- [Due to limitations in AWS, we initialized EBS volumes](#) (storage blocks) on the NFS servers before starting the test. Without disk initializations, there is a significant increase in disk latency, and test infrastructure slows for several hours.

We enabled **analytics** on each test instance to collect usage data. For more information, see [Collecting analytics for Bitbucket Server](#).

Methodology

Each test involved applying the same amount of traffic to a Bitbucket data set, but on a different AWS environment. We ran three series of tests, each designed to find optimal configurations for the following components:

- Bitbucket application node
- Database node
- NFS node

To help ensure benchmark reliability, we initialized the EBS volumes and tested each configuration for three hours. We observed stable response times throughout each test. Large instance tests used **Bitbucket Data Center 5.16** while XLarge used **Bitbucket Data Center 6.4**. We used a custom library (Trikit) running v1 protocol to simulate Git traffic.

Data sets

We created a Large-sized Bitbucket Data Center instance with the following dimensions:

| Metric | Value (approximate) |
|-----------------------------------|---------------------|
| Repositories | 52,000 |
| Active users | 25,000 |
| Pull requests | 850,000 |
| Traffic (git operations per hour) | 40,000 |

Content and traffic profiles are based on [Bitbucket Data Center load profiles](#), which put the instances overall load profile at the highest level of *Large* profile. We believe these metrics represent a majority of real-life, Large-sized Bitbucket Data Center instances.

| Metric | Value (approximate) |
|-------------------------------|---------------------|
| Users | 25,000 |
| Groups | 50,000 |
| Projects (including personal) | 16,700 |
| Comments on pull requests | 3,500,000 |

| Metric | Total | Component | Value (approximate) |
|---------------------|---------|----------------------|---------------------|
| Total repositories | 52,000 | Regular repositories | 26,000 |
| | | Public forks | 9,000 |
| | | Private repositories | 17,000 |
| Total pull requests | 859,000 | Pull requests open | 8,500 |
| | | Pull requests merged | 850,000 |

| | | | |
|---------------------------|--------|---------|--------|
| Traffic | 40,000 | Clones | 16,000 |
| (git operations per hour) | | Fetches | 14,000 |
| | | Pushes | 10,000 |

We created an XLarge-sized Bitbucket Data Center instance with the following dimensions:

| Metric | Value (approximate) |
|-----------------------------------|---------------------|
| Repositories | 110,000 |
| Active users | 50,000 |
| Pull requests | 1,790,000 |
| Traffic (git operations per hour) | 65,000 |

Content and traffic profiles are based on [Bitbucket Data Center load profiles](#), which put the instances overall load profile at the XLarge profile. We believe these metrics represent a majority of real-life, XLarge-sized Bitbucket Data Center instances.

| Metric | Value (approximate) |
|-------------------------------|---------------------|
| Users | 25,000 |
| Groups | 3,000 |
| Projects (including personal) | 52,000 |
| Comments on pull requests | 8,700,000 |

| Metric | Total | Component | Value (approximate) |
|--------------------------------------|-----------|----------------------|---------------------|
| Total repositories | 105,000 | Regular repositories | 52,000 |
| | | Public forks | 17,000 |
| | | Private repositories | 35,000 |
| Total pull requests | 1,790,000 | Pull requests open | 130,000 |
| | | Pull requests merged | 1,660,000 |
| Traffic
(git operations per hour) | 70,000 | Clones | 18,700 |
| | | Fetches | 25,300 |
| | | Pushes | 26,000 |

Benchmark

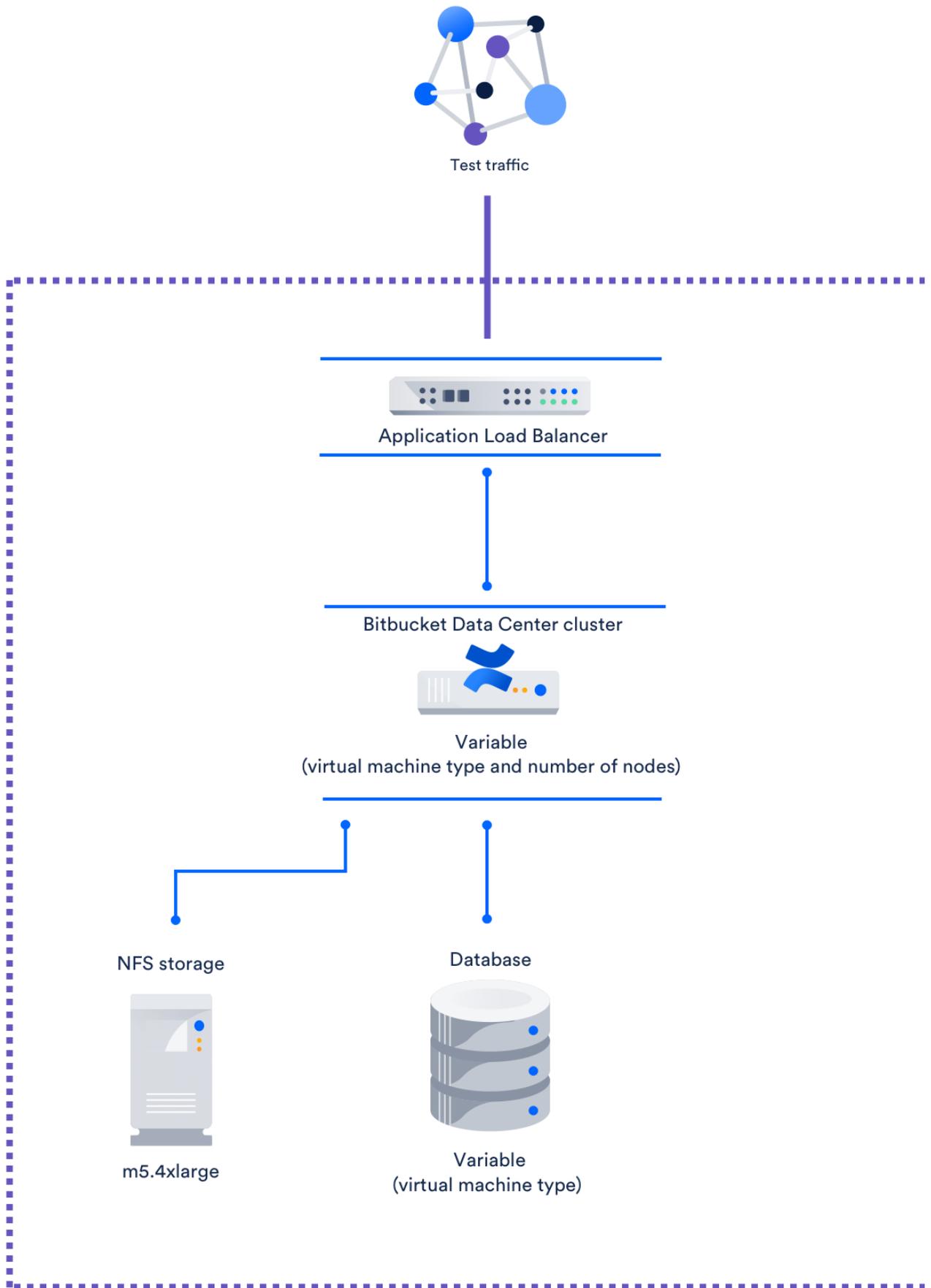
We used the following benchmark metrics for our tests.

| Benchmark metric | Threshold | Reason |
|------------------|-----------|--------|
|------------------|-----------|--------|

| | | |
|---|--|---|
| Git throughput , or the number of git hosting operations (fetch/clone/push) per hour | 32,700 (Minimum) for Large and
65,400 (Minimum) for XLarge ,
the higher the better | These thresholds are the upper limits of traffic defined in Bitbucket Data Center load profiles . We chose them due to the spiky nature of git traffic. |
| Average CPU utilization (for application nodes) | 75% (Maximum) , the lower the better | When the application nodes reach an average of CPU usage of 75% and above, Bitbucket's adaptive throttling starts queuing Git hosting operations to ensure the responsiveness of the application for interactive users. This slows down Git operations. |
| Stability | No nodes go offline | When the infrastructure is inadequate in handling the load it may lead to node crashes. |

 The test traffic had fixed sleep times to modulate the volume of git hosting operations. This means the benchmarked git throughput doesn't represent the maximum each configuration can handle.

Architecture



We tested each configuration on a freshly-deployed Bitbucket Server Data Center instance on AWS. Every configuration followed the same structure:

| Function | Number of nodes | Virtual machine type | Notes |
|------------------|-----------------|---|--|
| Application node | Variable | m5.xlarge
m5.2xlarge
m5.4xlarge
m5.12xlarge
m5.24xlarge | <p>When testing m5.xlarge (16GB of RAM), we used 8GB for JVM heap. For all others, we used 12GB for JVM heap. Minimum heap (Xms) was set to 1G for all the tests.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p> Weve observed that using a smaller JVM heap (2-3GB) is enough for most instances.</p> </div> <p>Also note that Git operations are expensive in terms of memory consumption and are executed outside of the Java virtual machine. See more on Scaling Bitbucket Server.</p> <p>Each Bitbucket application used 30GB General Purpose SSD (gp2) for local storage. This disk had an attached EBS volume with a baseline of 100 IOPS, burstable to 3,000 IOPS.</p> |
| Database | 1 | m5.xlarge
m5.2xlarge
m5.4xlarge | We used Amazon RDS Postgresql version 9.4.15, with default settings. Each test only featured one node. |
| NFS storage | 1 | m5.4xlarge
m5.2xlarge
m5.xlarge | <p>Our NFS server used a 900GB General Purpose SSD (gp2) for storage. This disk had an attached EBS volume with a baseline of 2700 IOPS, burstable to 3,000 IOPS. As mentioned, we initialized this volume at the start of each test.</p> <p>For more information on setting up Bitbucket Data Center's shared file server, see Step 2. Provision your shared file system (in Install Bitbucket Data Center). This section contains the requirements and recommendations for setting up NFS for Bitbucket Data Center.</p> |
| Load balancer | 1 | AWS Application Load Balancer (ELB) | We used AWS Elastic Load Balancer. Application Load Balancer at the time of performance testing doesn't handle SSH traffic. |

We ran several case studies of real-life Large and XLarge Bitbucket Data Center instances to find optimal configurations for each component. In particular, we found many used m5 series virtual machine types ([General Purpose Instances](#)). As such, for the application node, we focused on benchmarking different series' configurations.

 Refer to the [AWS documentation on Instance Types](#) (specifically, [General Purpose Instances](#)) for details on each virtual machine type used in our tests.

Recommendations for Large-sized instances

We analyzed our benchmarks and came up with the following optimal configuration:

Best-performing and most cost-effective configuration

| Component | Recommendation |
|-----------|----------------|
|-----------|----------------|

| | |
|-------------------|----------------------|
| Application nodes | m5.4xlarge nodes x 4 |
| Database node | m5.2xlarge |
| NFS node | m5.2xlarge |

Performance of this configuration

- Git throughput: 45,844 per hour
- Cost per hour ¹: \$4.168
- Average CPU utilization: 45%



¹ In our recommendations for Large-sized profiles, we quoted a *cost per hour* for each configuration. We provide this information to help inform you about the comparative price of each configuration. This cost only calculates the price of the nodes used for the Bitbucket application, database, and NFS nodes. It does not include the cost of using other components of the application like shared home and application load balancer.

These figures are in USD, and were correct as of July 2019.

We measured performance stability in terms of how far the instances average CPU utilization is from the 75% threshold. As mentioned, once we hit this threshold, git operations start to slow down. The further below the instance is from 75%, the less prone it is to slow due to sudden traffic spikes.

However, there are no disadvantages in using larger-size hardware (m5.12xlarge, for example), which will provide better performance.

Low-cost configuration

We also found a low-cost configuration with acceptable performance at **\$2.84** per hour:

| Component | Recommendation |
|-------------------|----------------|
| Application nodes | m5.4xlarge x 3 |
| Database node | m5.xlarge |
| NFS node | m5.xlarge |

This low-cost configuration offered a lower Git throughput of 43,099 git hosting calls per hour than the optimal configuration. However, this is still above our minimum threshold of 32,700 git hosting calls per hour. The trade-off for the price is fault tolerance. If the instance loses one application node, CPU usage spikes to 85%, which is above our maximum threshold. The instance will survive, but performance will suffer.

The following table shows all test configurations that passed our threshold, that is, above 32,500 git hosting operations per hour and below 75% CPU utilization, with no node crashes. We sorted each configuration by descending throughput.

| Application nodes | Database node | NFS node | Git throughput | Cost per hour |
|-----------------------|-------------------|-------------------|----------------|---------------|
| m5.4xlarge x 6 | m5.4xlarge | m5.4xlarge | 46,833 | 6.800 |
| m5.12xlarge x 2 | m5.4xlarge | m5.4xlarge | 45,848 | 6.792 |
| m5.4xlarge x 4 | m5.4xlarge | m5.4xlarge | 45,844 | 5.264 |
| m5.2xlarge x 8 | m5.4xlarge | m5.4xlarge | 45,626 | 5.264 |
| m5.4xlarge x 3 | m5.4xlarge | m5.4xlarge | 44,378 | 4.496 |

| | | | | |
|-----------------------|------------------|------------------|---------------|--------------|
| m5.4xlarge x 3 | m5.2xlarge | m5.4xlarge | 43,936 | 3.784 |
| m5.2xlarge x 6 | m5.4xlarge | m5.4xlarge | 43,401 | 4.496 |
| m5.4xlarge x 3 | m5.xlarge | m5.xlarge | 43,099 | 2.840 |
| m5.4xlarge x 3 | m5.xlarge | m5.4xlarge | 43,085 | 3.428 |

As you can see, the configuration m5.4xlarge x 4 nodes for the application doesn't provide the highest git throughput. However, configurations with higher throughput cost more and provide only marginal performance gains.

Recommendations for XLarge instances

We analyzed our benchmarks and came up with the following optimal configuration:

Best-performing configuration

| Component | Recommendation |
|-------------------|-----------------|
| Application nodes | m5.12xlarge x 4 |
| Database node | m5.2xlarge |
| NFS node | m5.2xlarge |

Performance of this configuration

- Git throughput: 75,860 per hour
- Cost per hour¹: \$10.312
- Average CPU utilization: 65%

We measured performance stability in terms of how far the instances average CPU utilization is from the 75% threshold. As mentioned, once we hit this threshold, git operations start to slow down. The further below the instance is from 75%, the less prone it is to slow due to sudden traffic spikes.

 ¹In our recommendations for Extra Large-sized profiles, we quoted a *cost per hour* for each configuration. We provide this information to help inform you about the comparative price of each configuration. This cost only calculates the price of the nodes used for the Bitbucket application, database, and NFS nodes. It does not include the cost of using other components of the application like shared home and application load balancer.

These figures are in USD, and were correct as of July 2019.

Low-cost configuration

We also found a low-cost configuration with good performance at **\$7.02** per hour:

| Component | Recommendation |
|-------------------|----------------|
| Application nodes | m5.8xlarge x 4 |
| Database node | m5.2xlarge |
| NFS node | m5.xlarge |

This low-cost configuration offered a lower Git throughput of 74,275 git hosting calls per hour than the optimal configuration. However, this is still well above the defined threshold of 65,400 git hosting calls per hour. The trade-off for the price is fault tolerance. There were timeouts and errors observed on the m5.8xlarge x 3 nodes, so performance degradation may be encountered if the an application node goes down.

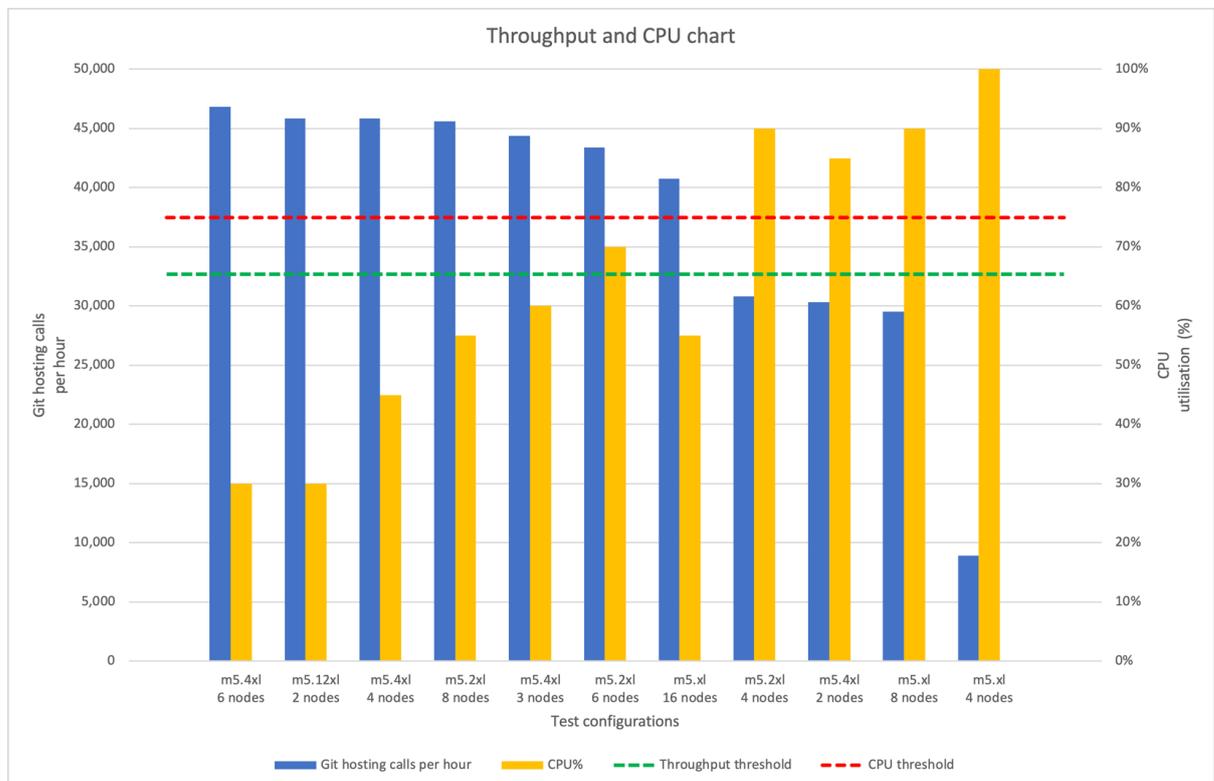
The following table shows all test configurations that passed our threshold, that is, above 32,500 git hosting operations per hour and below 75% CPU utilization, with no node crashes. We sorted each configuration by descending throughput.

| Application nodes | Database node | NFS node | Git throughput | Cost per hour |
|------------------------|-------------------|-------------------|----------------|-----------------|
| m5.12xlarge x 4 | m5.2xlarge | m5.2xlarge | 75,860 | \$ 10.31 |
| m5.4xlarge x 8 | m5.2xlarge | m5.2xlarge | 73,374 | \$ 7.24 |
| m5.8xlarge x 4 | m5.2xlarge | m5.xlarge | 74,275 | \$ 7.02 |
| m5.4xlarge x 6 | m5.2xlarge | m5.2xlarge | 71,872 | \$ 5.70 |
| m5.12xlarge x 3 | m5.2xlarge | m5.2xlarge | 66,660 | \$ 8.01 |

Application node test results

Our first test series focused on finding out which AWS virtual machine types to use (and how many) for the application node. For these tests, we used a single **m4.4xlarge** node for the database and single **m4.4xlarge** node for the NFS server.

Benchmarks show the best git throughput came from using m5.4xlarge (16 CPUs) and m5.12xlarge nodes (46 CPUs). You will need at **least three nodes for m5.4xlarge and two nodes for m5.12xlarge**.



CPU is underutilized at 30% for the following application node configurations:

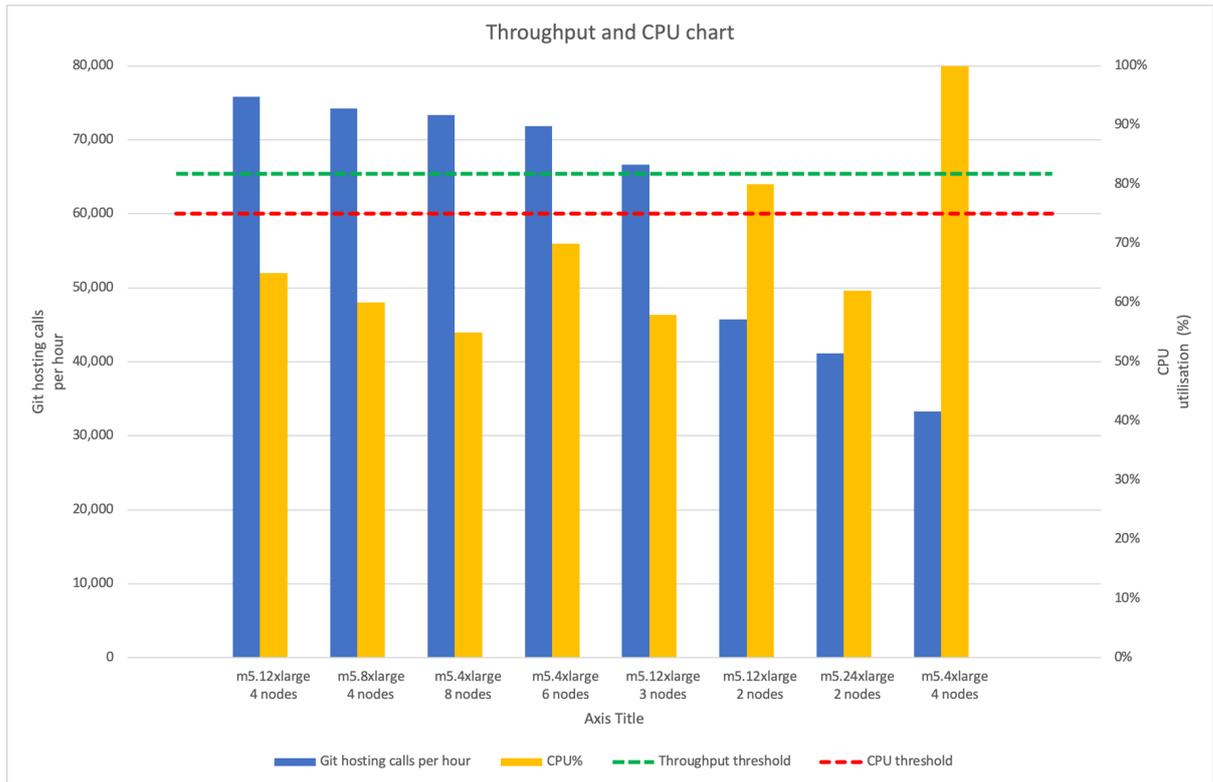
- m5.4xlarge x 6
- m5.12xlarge x 2

This demonstrates both configurations are overprovisioned. It would be more cost-effective to use **three or four m5.4xlarge nodes** for the application.

However, on the three-node m5.4xlarge set-up, the CPU usage would be at ~85% if one of the nodes failed. For this reason, we recommend the **four-node m5.4xlarge** set-up for better fault tolerance.

Our first test series focused on finding out which AWS virtual machine types to use (and how many) for the application node. For these tests, we used a single **m4.2xlarge** node for the database and single **m4.2xlarge** node for the NFS server.

Benchmarks show the best git throughput came from using **m5.12xlarge (48 CPUs)** and **m5.8xlarge nodes (32 CPUs)**. You will need four nodes for both instance types.



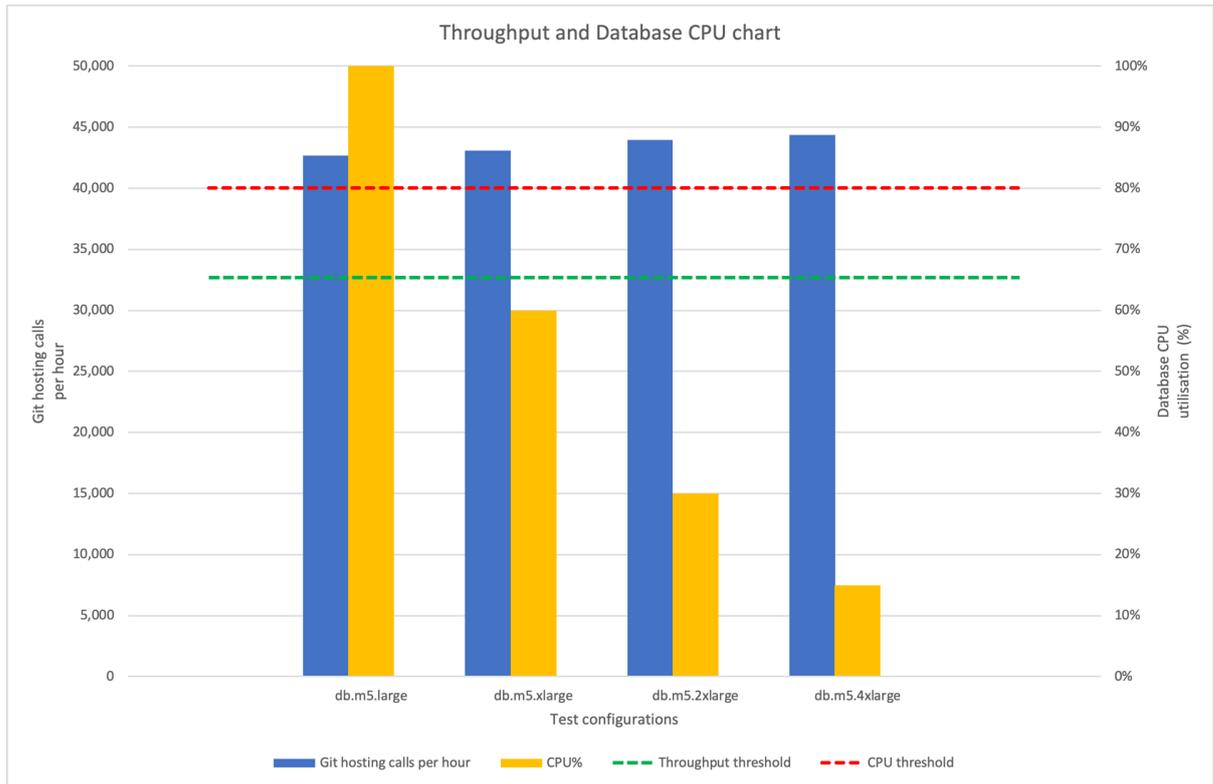
We have also carried out performance testing on 2 nodes (96 CPUs), but this resulted in poor performance, not meeting the threshold. Test results showed that 2 node deploys are not suitable for xlarge load. During the 2 node tests, the time spent on kernel was very high, which was not evident on 4+ nodes.

Database node test results

From the application node test series, we found using **three m5.4xlarge** nodes for the application yielded optimal performance (even if it wasn't the most fault tolerant). For our second test series, we tested this configuration against the following virtual machine types for the database:

- m4.large
- m4.xlarge
- m4.2xlarge
- m4.4xlarge

As expected, the more powerful virtual machine used, the better the performance. We saw the biggest gains in CPU utilization. Git throughput also improved, but only marginally.

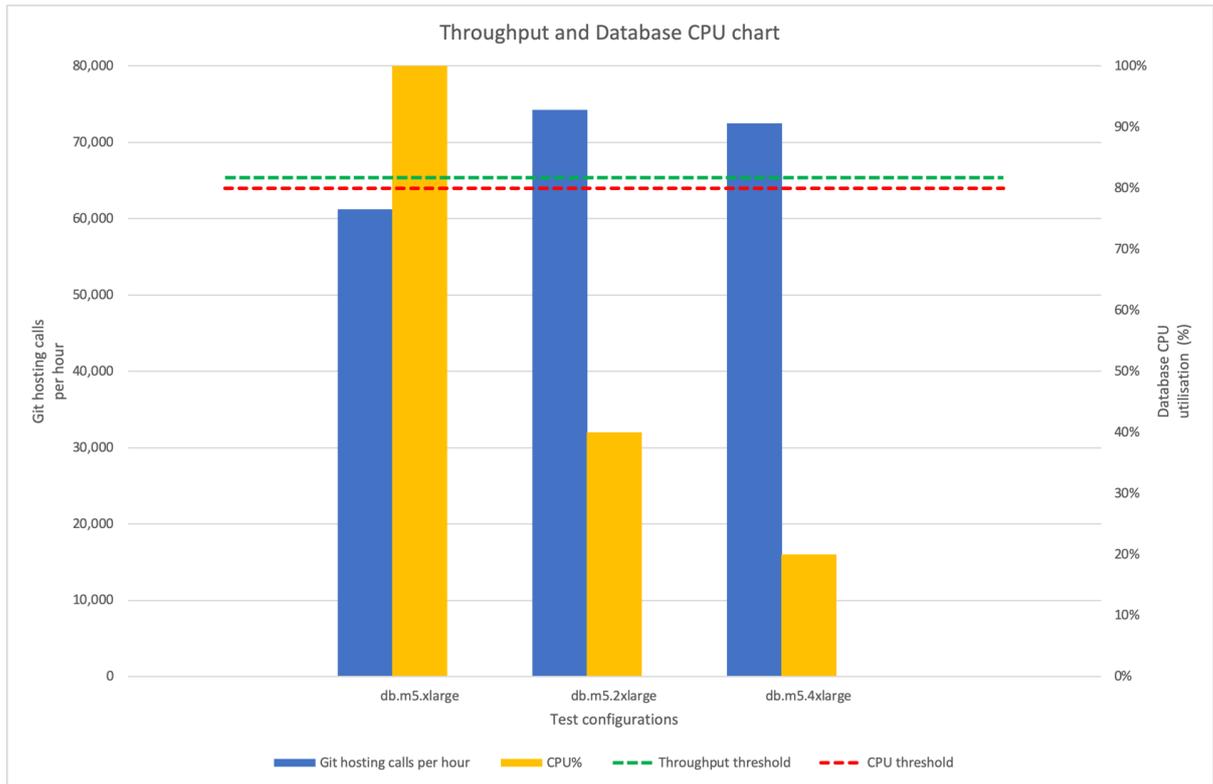


Only **m5.large** failed the CPU utilization threshold. All other tested virtual machine types are acceptable, although, **m5.xlarge** is pretty close to our CPU utilization threshold at 60%.

From the application node test series, we found using **four m5.12xlarge** nodes for the application yielded optimal performance. For our second test series, we tested this configuration against the following virtual machine types for the database:

- m4.xlarge
- m4.2xlarge
- m4.4xlarge

The m4.xlarge was saturated on CPU at 100%, and db.m4.4xlarge did not result in improvements in performance. For this reason, m4.2xlarge remains the recommended instance type for the extra-large load. The CPU utilisation was at ~ 40% on m4.2xlarge.



NFS node test results

In previous tests (where we benchmarked different application and database node configurations), we used **m5.4xlarge** for the NFS node (NFS protocol v3). During each of those tests, NFS node CPU remained highly underutilized at under 18%. We ran further tests to see if we could downgrade the NFS server (and, by extension, find more cost-effective recommendations). Results showed identical git throughput, using the downsized m5.xlarge NFS node. This led to our low-cost recommendation.

| Component | Recommendation |
|-------------------|----------------|
| Application nodes | m5.4xlarge x 3 |
| Database node | m5.xlarge |
| NFS node | m5.xlarge |

As mentioned, this recommendation costs \$3.044 per hour but offers lower fault tolerance.

Based on other test results, we recommend that, for the NFS node, use at least **m5.xlarge** with **IOPs higher than 1500**.

Benchmarks for the extra-large tests all used **m5.2xlarge** for the NFS instance. During each of those tests, the NFS node CPU remained highly underutilized at 25%. We ran further tests to see if we could downgrade the NFS server (and, by extension, find more cost-effective recommendations). Results showed identical git throughput, using the downsized **m5.xlarge** NFS node with CPU utilisation at 60%.

This led to our low-cost recommendation.

| Component | Recommendation |
|-------------------|----------------|
| Application nodes | m5.8xlarge x 4 |

| | |
|---------------|------------|
| Database node | m5.2xlarge |
| NFS node | m5.xlarge |

Disk I/O

Disk I/O performance is often a limiting factor, so we also paid attention to disk utilization. Our tests revealed the disk specifications we used for the NFS node were appropriate to our traffic:

- 900GB [General Purpose SSD \(gp2\)](#) for storage
- IOPS:
 - Baseline of 2700 IOPS
 - Burstable to 3,000 IOPS.

As mentioned, we [initialized](#) this volume at the start of each test.

 Please be aware this information is only a guideline, as IOP requirements will depend on usage patterns.

The table below shows the I/O impact of our tests on the NFS nodes disk:

| Metric | Value |
|--|------------|
| Total throughput (Read + Write throughput) | 1,250 IOPS |
| Read throughput | 700 IOPS |
| Write throughput | 550 IOPS |
| Read bandwidth | 100 MB/s |
| Write bandwidth | 10 MB/s |
| Average queue length | 1.3 |
| Average read latency | 1.5 ms/op |
| Average write latency | 0.6 ms/op |
| Disk utilization | 45% |

Disk I/O performance is often a limiting factor, so we also paid attention to disk utilization. Our tests revealed the disk specifications we used for the NFS node were appropriate to our traffic:

- 1800GB [General Purpose SSD \(gp2\)](#) for storage
- IOPS: baseline of 4500 IOPS

As mentioned, we [initialized](#) this volume at the start of each test.

 Please be aware this information is only a guideline, as IOP requirements will depend on usage patterns.

| Metric | Value |
|--|------------|
| Total throughput (Read + Write throughput) | 9,00 IOPS |
| Read throughput | 2,700 IOPS |
| Write throughput | IOPS |

| | |
|-----------------------|------------|
| Read bandwidth | 113 MB/s |
| Write bandwidth | 15 MB/s |
| Average queue length | 3.5 |
| Average read latency | 1.0 ms/op |
| Average write latency | 0.70 ms/op |
| Disk utilization | 80 % |

Although the average disk utilisation is high at 80%, the read and write latency was low at < 1ms/op. It is recommended that the NFS server disk to have 4500 IOPs or more to ensure that it does not become the bottleneck.

Secure Bitbucket in AWS

This page describes security best practices for running and maintaining Bitbucket in AWS.

Amazon Virtual Private Cloud (VPC) and Subnets

Amazon VPC enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. See [Amazon EC2 and Amazon Virtual Private Cloud](#) for more information.

A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a subnet that you select. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet.

See Amazon's article called [Your VPC and Subnets](#) for a general overview of VPCs and subnets.

To bolster the security of your VPC you may wish to enable one or more of the following:

- Secure your VPC with a firewall virtual appliance / AMI to defend against unauthorized network activity
- Configure a site-to-site VPN to ensure information is transferred securely between Bitbucket Server and its users
- Configure an intrusion prevention or intrusion detection virtual appliance to detect when unauthorized network activity has occurred
- Enable [Amazon CloudTrail](#) to log VPC API operations and keep an audit trail of network changes

Atlassian Standard Infrastructure

If you deployed Bitbucket through the [AWS Quick Start](#), it will use the Atlassian Standard Infrastructure (ASI). The ASI is a virtual private cloud (VPC) that contains the components required by all Atlassian Data Center applications. For more information, see [Atlassian Standard Infrastructure \(ASI\) on AWS](#).

Security Groups

A security group acts as a virtual firewall that controls the traffic for one or more instances. The security group(s) that apply to newly launched instances depend on your launch method:

- If you launched instance(s) via the AWS console or API, the EC2 launch process gives you the opportunity to either create a new security group or associate one or more existing security group(s) with the instance. We recommend allowing inbound access to Bitbucket only on ports 22, 80, 443, and 7999, and only allowing access from the tightest possible IP address range.
- If launched via **BitbucketServer.template** or **BitbucketDataCenter.template**, AWS CloudFormation creates and manages a security group as part of the stack, allowing inbound access on ports 22, 80, 443, and 7999 from the **Permitted IP range** of addresses you specify. We recommend specifying the tightest possible **Permitted IP range** and not adding unnecessary inbound access to the security group after launch.

We recommend using security groups to restrict incoming traffic to your Bitbucket instance to the absolute minimum required.

See [Amazon EC2 Security Groups for Linux Instances](#) for more information.

Configure SSL to enable HTTPS

On this page:

- [Amazon Virtual Private Cloud \(VPC\) and Subnets](#)
- [Security Groups](#)
- [Configure SSL to enable HTTPS](#)
- [Keeping your system up-to-date](#)

In order to enable HTTPS, you need a valid SSL certificate. SSL certificates are issued by a trusted third party Certificate Authority (CA), such as [VeriSign](#), [DigiCert](#) or [Thawte](#), which provide such services on a commercial basis. Atlassian does not provide such services.

Once you have a proper SSL certificate from a CA, you can [import it to AWS Certificate Manager](#). This will enable you to use the certificate later on when deploying Bitbucket on AWS. To do that (for example, when deploying Bitbucket through the [Quick Start](#)), you'll need to provide the your certificate's Amazon Resource Number (ARN).

 Until you install a valid SSL certificate, new Bitbucket Server and Data Center instances are configured to serve requests over plain HTTP, not HTTPS. This means all passwords and data will be sent unencrypted over the public internet (unless users are connected to AWS via a [Virtual Private Gateway](#)).

Load balancer settings with SSL enabled or disabled

When you deploy Bitbucket with SSL enabled, your load balancer's listeners will be set up as follows:

| Load Balancer Protocol | Load Balancer Port | Instance Protocol | Instance Port |
|------------------------|--------------------|-------------------|---------------|
| HTTP | 80 | HTTP | 7991 |
| HTTPS | 443 | HTTP | 7990 |

But if you didn't have an SSL certificate set up at initial launch time, your ELB will be configured with only one HTTP listener as follows:

| Load Balancer Protocol | Load Balancer Port | Instance Protocol | Instance Port |
|------------------------|--------------------|-------------------|---------------|
| HTTP | 80 | HTTP | 7990 |

Replace any self-signed SSL certificates (for Bitbucket Server)

When you use the [Quick Start](#) to deploy Bitbucket Data Center, you can supply a proper CA certificate to your deployment immediately at launch time. However, if you deploy Bitbucket Server via [Bitbucket Server template](#) (or manually as described in [Launching Bitbucket Server in AWS](#)), there is currently no way to install your own SSL certificate at initial launch time.

If you intend for your Bitbucket Server instance to be internet facing, we recommend setting `ATL_SSL_SELF_CERT_ENABLED=true` to enable HTTPS to your instance at launch time. This setting will generate a self-signed certificate for your Bitbucket Server instance.

Self-signed certificates do not offer the same security as proper CA certificates. If your Bitbucket Server instance uses a self-signed certificate:

- most browsers will display security warnings that must be ignored before proceeding to the Bitbucket Server Web interface
- git clients will refuse to connect to Bitbucket Server over HTTPS unless configured to ignore the self-signed certificate with `git config --global http.sslVerify false`
- application links and/or integrations with other applications that use Bitbucket Server's REST API and don't accept self-signed certificates may fail

Because of this, we recommend that you replace any self-signed SSL certificate with a proper CA certificate for your domain. To do this:

1. Place your certificate file at (for example) `/etc/nginx/ssl/my-ssl.crt`
2. Place your `password-less` certificate key file at `/etc/nginx/ssl/my-ssl.key`
3. Edit `/etc/nginx/nginx.conf` as follows:
 - a. Replace references to `/etc/nginx/ssl/self-ssl.crt` with `/etc/nginx/ssl/my-ssl.crt`

- b. Replace references to `/etc/nginx/ssl/self-ssl.key` with `/etc/nginx/ssl/my-ssl.key`.
4. Append the contents of `/etc/nginx/ssl/my-ssl.crt` to the default system PKI bundle (`/etc/pki/tls/certs/ca-bundle.crt`) to ensure scripts on the instance (such as DIY backup) can curl successfully.
5. Restart nginx.

Installing or changing your SSL certificate after deploying

When you deploy Bitbucket Data Center for the first time through the [Quick Start](#), we recommend that you supply a proper CA certificate. To do this, you need to [import your certificate to AWS Certificate Manager](#) and then specify its Amazon Resource Number through the **SSL Certificate ARN** field.

If you didn't do this during deployment through the [Quick Start](#) (or if you used a self-signed certificate), you can still install or change your SSL certificate anytime *after* initial deployment:

1. In the AWS console, go to **Services > CloudFormation**, select your stack, and click **Update Stack**.
2. Specify your certificate's Amazon Resource Number (ARN) in the **SSL Certificate ARN** field. See [Importing Certificates into AWS Certificate Manager](#) for more information.
3. Configure the HTTP to HTTPS redirect manually in the `bitbucket.properties` file, located in the `<Bitbucket home directory>/sharedirectory`, as described in [Redirect HTTP Requests to HTTPS](#).
4. Restart the Bitbucket service by running the following command on all application nodes

```
sudo service atlbitbucket restart
```

 If the hostname of your Bitbucket instance has changed you will need to update Bitbucket's base URL as described in the page [Specifying the base URL for Bitbucket Server](#).

Keeping your system up-to-date

It is essential to keep your Bitbucket Server instance up-to-date with patches and updates to maximize security and minimize opportunity for exploits and misadventure. On first boot a Bitbucket Server AMI instance will download the latest official release of Bitbucket Server at that time so you are assured of having the very latest version of Bitbucket Server when you first start using Bitbucket Server in AWS.

 Please be sure to always perform a backup of your instance before attempting any update.

Amazon Linux Security Updates

The Bitbucket Server AMI is based on Amazon Linux and the latest version of this is used whenever we cut a new release of the Bitbucket Server AMI. Occasionally vulnerabilities in libraries and utilities used in Amazon Linux will be detected and updates posted in the Amazon Linux AMI yum repository. Atlassian will issue new versions of the Bitbucket Server AMI where necessary to ensure new Bitbucket Server AWS instances start with these updates but if you are managing an existing instance you may need to apply these updates yourself. By default, Amazon Linux applies all security updates on reboot. Alternatively you can run "yum update --security".

You may wish to apply other updates from the Amazon Linux AMI yum repository to your Bitbucket Server instance. You must ensure that any updated packages are supported by the version of Bitbucket Server you are running. Bitbucket Server version requirements can always be found on the [Supported platforms](#) page.

Bitbucket Server Updates

The Atlassian Bitbucket Server team have a strong release cadence and routinely issue releases including new features, performance and security fixes. It is strongly recommended you keep Bitbucket Server as up to date as possible. To update Bitbucket Server in an existing instance please follow the [Bitbucket Server Upgrade Guide](#).

Using Bitbucket DIY Backup in AWS

This page describes considerations for DIY Backup and Restore of Bitbucket instances deployed in an Amazon Web Services (AWS) environment.

About Bitbucket Server DIY Backup for AWS

The [Bitbucket Server DIY Backup](#) tools fully support backup and restore of Bitbucket instances deployed on Amazon Elastic Block Store (EBS) volumes and/or Amazon Relational Database Service (RDS) instances.

The tools work by taking EBS and/or RDS snapshots of Bitbucket's database and shared home directory volume. These snapshots can later be launched as new EBS volumes and/or RDS instances, and attached to your running instance, thus restoring Bitbucket to a specific point in time.

The benefits of using native Amazon EBS and RDS support in Bitbucket Server DIY Backup are:

- taking AWS native snapshots are faster than filesystem level copying
- backup-associated downtimes can be eliminated
- the snapshots are stored with the redundancy and durability of S3
- it makes it easy to relocate an instance to a different Region or Availability Zone in the future

The scripts use the [AWS CLI](#) toolset, which is included in all instances launched from the AMI, regardless of the method you used to launch it.

If you launched your Bitbucket instance via **BitbucketServer.template** or **BitbucketDataCenter.template**, your EC2 node(s) should already belong to an [IAM role](#) with a policy granting the permissions required to backup and restore the EBS volume and/or RDS instance. If you did not launch your Bitbucket instance in AWS via one of these CloudFormation templates, then see the [Setting up the instance role](#) section below for an example policy with similar permissions.

Configuring the Bitbucket Server DIY Backup in AWS

If you launched your Bitbucket instance via **BitbucketServer.template** or **BitbucketDataCenter.template**, there is already a copy of the [Bitbucket Server DIY Backup](#) tools pre-installed and pre-configured on your instance. Just SSH to your instance (the shared file server node in the case of Bitbucket Data Center), and run:

```
cd /opt/atlassian/bitbucket-diy-backup
git pull
./bitbucket.diy-backup.sh
```

If you launched your Bitbucket instance in AWS manually, you need to clone the [Bitbucket Server DIY Backup](#) repository and configure the variables manually.

```
git clone git@bitbucket.org:atlassianlabs/atlassian-bitbucket-diy-backup.git
cd atlassian-bitbucket-diy-backup
cp -i bitbucket.diy-backup.vars.sh.example-aws bitbucket.diy-backup.vars.sh
```

Then edit `bitbucket.diy-backup.vars.sh` appropriately for your environment.

Once you have configured `bitbucket.diy-backup.vars.sh` correctly, you can then run your backups by typing:

```
./bitbucket.diy-backup.sh
```

On this page

- [About Bitbucket Server DIY Backup for AWS](#)
- [Configuring the Bitbucket Server DIY Backup in AWS](#)

See [Bitbucket zero downtime backup](#) for more information.

Setting up the instance role

The DIY backup and restore scripts use the [AWS CLI](#) toolset to do their job. These tools need to authenticate with AWS in order to gain access to your resources (EBS volumes, snapshots, etc). The recommended way of providing credentials to the instance is by launching it with an instance role that has a suitable policy attached. If you are using the [Bitbucket Server CloudFormation template](#), it'll take care of creating a policy for you and attach it to the instance at launch time.

If you need to create your own policy, you can use this JSON object as an example of the minimum permissions required for an instance:

```
{
  "Statement": [
    {
      "Resource": [
        "*"
      ],
      "Action": [
        "ec2:AttachVolume",
        "ec2:CreateSnapshot",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVolumes",
        "ec2:DetachVolume"
      ],
      "Effect": "Allow"
    }
  ],
  "Version": "2012-10-17"
}
```

For other ways of configuring the AWS CLI toolset, please refer to the [documentation](#).

Specify the Bitbucket base URL

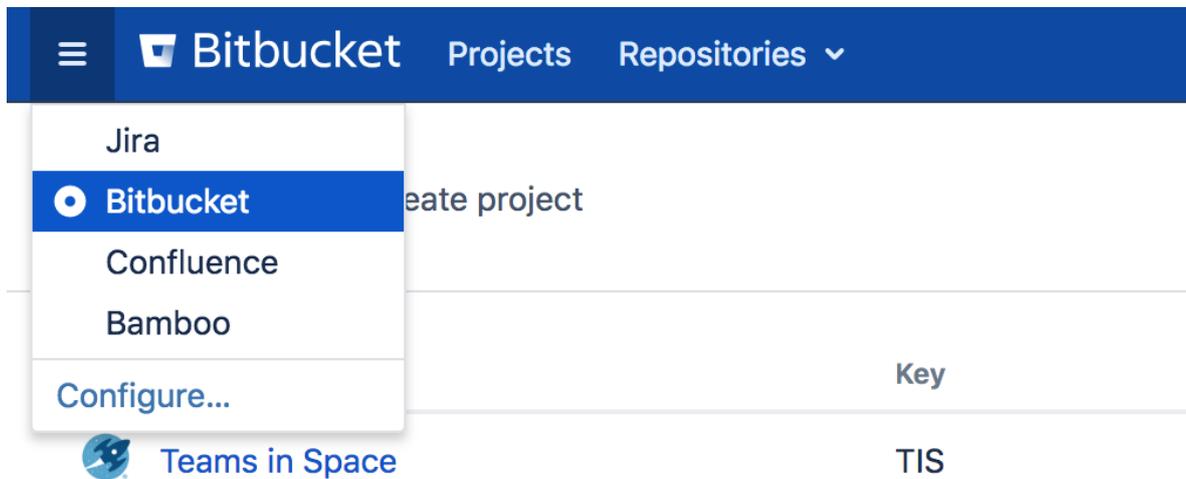
The base URL is the root URL for your installation of Bitbucket Data Center and Server. All links *which are not from a web request* (for example in Bitbucket email notifications), will be prefixed by this URL. If you are experiencing trouble with setting an `https` base URL, ensure you configured [Tomcat with SSL](#) correctly.

To specify Bitbucket's base URL

1. In the administration area, click **Server settings** (under 'Settings').
2. In the **Base URL** field, type the URL address of your Bitbucket instance (for example, "https://bitbucket.mycompany.com").
3. Click **Save**.

Configuring the application navigator

The application navigator, on the left of the Bitbucket Data Center and Serverheader, allows you to switch to your other applications, such as JIRA Software and Bamboo or any other web application all from the Bitbucketheader:



Users only see the application navigator when links are set up if there are no links, only administrators can see it.

Bitbucket administrators can configure which apps appear in the navigator just click **Configure** in the application navigator, or go to the Bitbucket admin area and click **Application Navigator**:

- **Linked applications** are automatically configured in the application navigator, and can't be deleted. Click **Manage** to configure those in the source application.
- Specify new links, as required by your users, by entering a **Name** and **URL**.
- Restrict the visibility of links to particular user groups, or hide the link completely. Click in a row, under the **Groups** column header, to edit those properties for existing rows.
- Use the 'handles' at the left to change the link order when seen in Bitbucket.

Application Navigator

The application navigator appears in the top left corner of the header so users can quickly access other applications. Linked applications are automatically configured in the application navigator for you, and you can't delete them. Add your own links for users by providing the name and URL of the destination below.

| Name | URL | Hide | Restricted to Groups | |
|----------------------|--|--------------------------|----------------------|---------------------------------------|
| <input type="text"/> | <input type="text"/> | <input type="checkbox"/> | <input type="text"/> | <input type="button" value="Add"/> |
| JIRA | http://jira.teamsinspace.com:8080/ | Manage | | |
| Confluence | http://confluence.teamsinspace.com:8090/ | Manage | | |
| FishEye + Crucible | http://fecru.teamsinspace.com:8060/ | Location | | |
| Bitbucket | https://bitbucket.dev.com:7990/ | | | <input type="button" value="Delete"/> |

Managing apps

An app is an installable component that supplements or enhances the functionality of Bitbucket Data Center and Server in some way. For example, the [Custom Navigation Plugin](#) enables you to configure custom navigation tabs specific to a repository. Other apps are available for adding graphs to Bitbucket, importing SVN source control projects, and accessing Atlassian support.

Bitbucket comes with many pre-installed apps (called system apps). You can install more apps, either by acquiring the app from the [Atlassian Marketplace](#) or by uploading it from your file system. This means that you can install apps you have developed yourself. For information about developing your own apps for Bitbucket, see [Bitbucket Developer Documentation](#).

About the Universal Plugin Manager (UPM)

You administer apps for Bitbucket using the Universal Plugin Manager (UPM). The UPM is itself an app that exposes app administration pages in the Bitbucket Administration Console. UPM works across Atlassian applications, providing a consistent interface for administering apps in Bitbucket, Crucible, Confluence, Fisheye, Jira applications, and Bamboo.

UPM comes pre-installed in recent versions of all Atlassian applications, so you do not normally need to install it yourself. However, like other apps, the UPM software is subject to regular software updates. Before administering apps in Bitbucket, therefore, you should [verify your version](#) of the UPM and update it if needed.

Administering apps in Bitbucket

You can update UPM, or any app, from the UPM's own app administration pages. Additionally, you can perform these tasks from the UPM administration pages:

- Install or remove apps
- Configure app settings
- Discover and install new apps from the [Atlassian Marketplace](#)
- Enable or disable apps and their component modules

It shows only those plugins that are supported in your version of the product, so that you do not install incompatible plugins.

If the app request feature is enabled in your Atlassian application, non-administrative users can also discover apps on the Atlassian Marketplace. Instead of installing the apps, however, these users have the option of requesting the apps from you, the administrator of the Atlassian application.

For more information on administering the app request feature or performing other common app administration tasks, see the [Universal Plugin Manager documentation](#). For an end-user's view of requesting apps in Bitbucket, see [Requesting apps](#).

Apps for Bitbucket Data Center

Installing, and managing, apps for Bitbucket Data Center is done in the same way, as described above. The only requirement is that the app is Data Center-compatible see [Bitbucket Data Center Apps](#) for compatibility information.

You can install an app from any cluster node. The app is stored on the [shared file system](#) for the Bitbucket Data Center, and made available to all nodes in the cluster.

On this page

- [About the Universal Plugin Manager \(UPM\)](#)
- [Administering apps in Bitbucket](#)
- [Apps for Bitbucket Data Center](#)

Add a POST service webhook

Repository administrators can add a POST service to a repository. Bitbucket Data Center and ServerPOSTs to the service URL you specify.

You can use anURL with the following format:

```
https://server:port/path/
```

The service receives a POST whenever the user pushes to the repository.

The content type header of the POST has an`application/json`type. The content is a JSON payload that represents the repository push.

Setting up the POST service

You can either set up the POST service manually or you can write a service to automate this. You would write a service if you are integrating an application with Bitbucket.

Set up in the repository settings

1. Go to **Repository Settings > Hooks**
2. Click **Add hook**
3. Install the **Web Post Hooks for Bitbucket Server** app
4. Go back to **Repository Settings > Hooks**
5. Find **Post-receive WebHooks** and choose **Enabled**
6. Add the URLs where Bitbucket should send update messages. You can add 1 to 5 URLs.
7. Click **Enable**

POST data

When a user pushes to a repository, BitbucketPOSTs to the URL you provided. The body of the POST request contains information about the repository where the change originated, a list of recent commits, and the name of the user that made the push.

Example of payload

This is an example of a push that contains one commit that changes 2 files (*pom.xml*) in folders *iridium-common* and *iridium-magma*.

JSON Payload

```
{
  "repository":{
    "slug":"iridium-parent",
    "id":11,
    "name":"iridium-parent",
    "scmId":"git",
    "state":"AVAILABLE",
    "statusMessage":"Available",
    "forkable":true,
    "project":{
      "key":"IR",
      "id":21,
      "name":"Iridium",
      "public":false,
      "type":"NORMAL",
      "isPersonal":false
    },
    "public":false
  },
  "refChanges":[
    {
      "refId":"refs/heads/master",
```

```

    "fromHash": "2c847c4e9c2421d038fff26ba82bc859ae6e20",
    "toHash": "f259e9032cdeble28d073e8a79a1fd6f9587f233",
    "type": "UPDATE"
  }
],
"changesets": {
  "size": 1,
  "limit": 100,
  "isLastPage": true,
  "values": [
    {
      "fromCommit": {
        "id": "2c847c4e9c2421d038fff26ba82bc859ae6e20",
        "displayId": "2c847c4"
      },
      "toCommit": {
        "id": "f259e9032cdeble28d073e8a79a1fd6f9587f233",
        "displayId": "f259e90",
        "author": {
          "name": "jhocman",
          "emailAddress": "jhocman@atlassian.com"
        },
        "authorTimestamp": 1374663446000,
        "message": "Updating poms ...",
        "parents": [
          {
            "id": "2c847c4e9c2421d038fff26ba82bc859ae6e20",
            "displayId": "2c847c4"
          }
        ]
      },
      "changes": {
        "size": 2,
        "limit": 500,
        "isLastPage": true,
        "values": [
          {
            "contentId": "2f259b79aa7e263f5829bb6e98096e7ec976d998",
            "path": {
              "components": [
                "iridium-common",
                "pom.xml"
              ],
              "parent": "iridium-common",
              "name": "pom.xml",
              "extension": ".xml",
              "toString": "iridium-common/pom.xml"
            },
            "executable": false,
            "percentUnchanged": -1,
            "type": "MODIFY",
            "nodeType": "FILE",
            "srcExecutable": false,
            "link": {
              "url": "/projects/IR/repos/iridium-parent/commits/f259e9032cdeble28d073e8a79a1fd6f9587f233#iridium-common/pom.xml",
              "rel": "self"
            }
          },
          {
            "contentId": "2f259b79aa7e263f5829bb6e98096e7ec976d998",
            "path": {
              "components": [
                "iridium-magma",
                "pom.xml"
              ],
              "parent": "iridium-magma",
              "name": "pom.xml",
              "extension": ".xml",
              "toString": "iridium-magma/pom.xml"
            },
            "executable": false,
            "percentUnchanged": -1,
            "type": "MODIFY",
            "nodeType": "FILE",
            "srcExecutable": false,

```

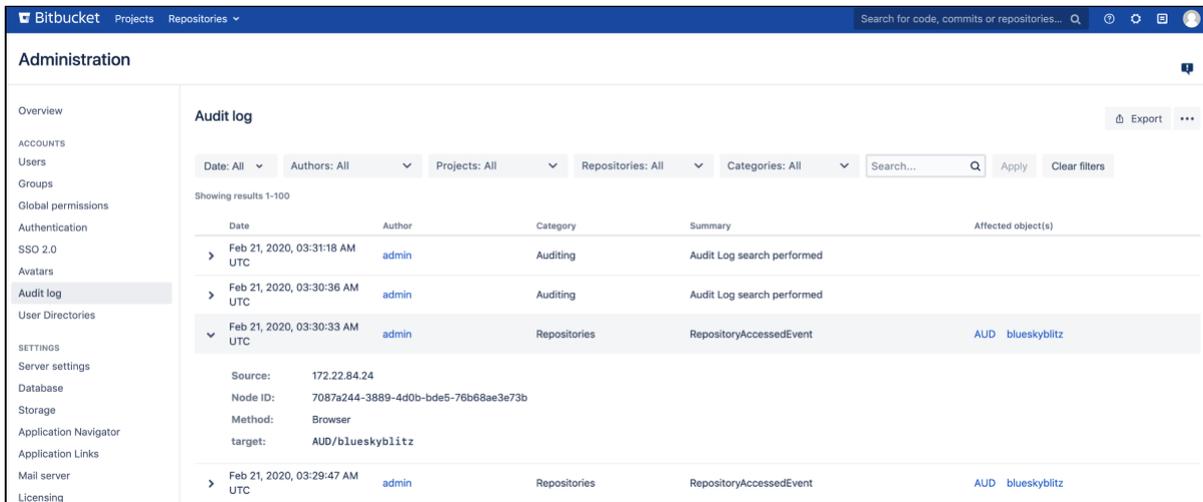
```
        "link":{
          "url":"/projects/IR/repos/iridium-parent/commits
/f259e9032cdeble28d073e8a79a1fd6f9587f233#iridium-magma/pom.xml",
          "rel":"self"
        }
      },
      "start":0,
      "filter":null
    },
    "link":{
      "url":"/projects/IR/repos/iridium-parent/commits
/f259e9032cdeble28d073e8a79a1fd6f9587f233#iridium-magma/pom.xml",
      "rel":"self"
    }
  },
  "start":0,
  "filter":null
}
```

Properties

Some of the system-wide properties for the Webhook Plugin can be overridden in the Bitbucketconfiguration file. The available properties are listed in [Bitbucket Server config properties](#).

View and configure the audit log

The auditing feature tracks key activities in Bitbucket Data Center and Server, allowing administrators to get an insight into the way Bitbucket is being used. The audit system can be used to identify authorized and unauthorized changes, or suspicious activity over a period of time. The audit log experience lets you search and filter the log for details, along with utilizing grouped coverage areas for clarity.



| Date | Author | Category | Summary | Affected object(s) |
|-------------------------------|--------|--------------|----------------------------|--------------------|
| Feb 21, 2020, 03:31:18 AM UTC | admin | Auditing | Audit Log search performed | |
| Feb 21, 2020, 03:30:36 AM UTC | admin | Auditing | Audit Log search performed | |
| Feb 21, 2020, 03:30:33 AM UTC | admin | Repositories | RepositoryAccessedEvent | AUD blueskyblitz |
| Feb 21, 2020, 03:29:47 AM UTC | admin | Repositories | RepositoryAccessedEvent | AUD blueskyblitz |

Expanded event details:

Source: 172.22.84.24
Node ID: 7087a244-3889-4d0b-bde5-76b68ae3e73b
Method: Browser
target: AUD/blueskyblitz

Viewing the audit logs for your instance

To view the global audit page:

1. In the administration area, go to **Audit log** (under Accounts).
2. **Expand** any event to get more details.



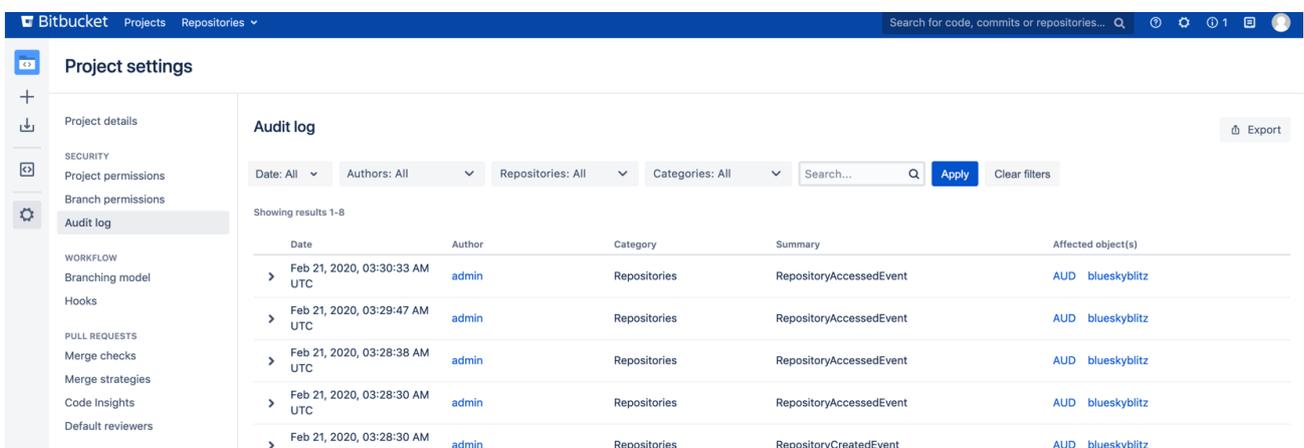
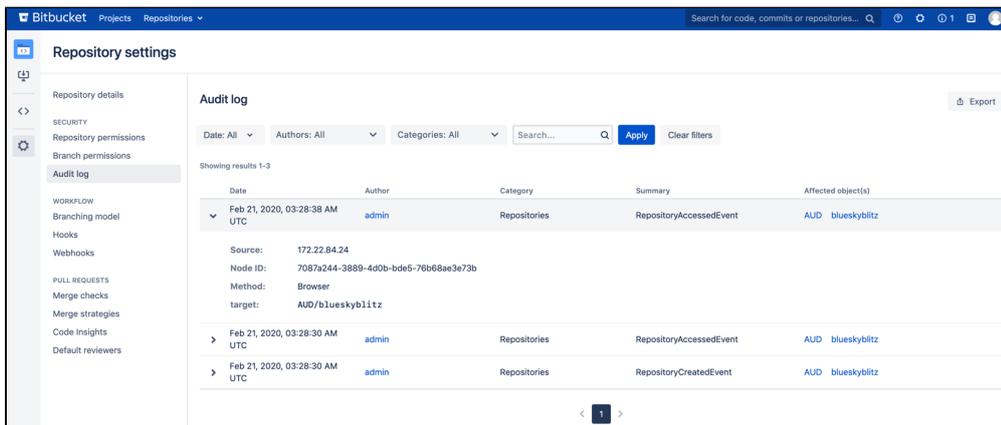
| | | | | |
|--|-------|--------------|--------------------------------------|------------------|
| Feb 21, 2020, 03:28:30 AM UTC | admin | Repositories | RepositoryCreatedEvent | AUD blueskyblitz |
| Source: 172.22.84.24 | | | | |
| Node ID: 7087a244-3889-4d0b-bde5-76b68ae3e73b | | | | |
| Method: Browser | | | | |
| details: {"project": "AUD", "repository": "blueskyblitz"} | | | | |
| target: AUD/blueskyblitz | | | | |
| Feb 21, 2020, 03:28:18 AM UTC | admin | Projects | BranchModelConfigurationCreatedEvent | AUD |
| Source: 172.22.84.24 | | | | |
| Node ID: 7087a244-3889-4d0b-bde5-76b68ae3e73b | | | | |
| Method: Browser | | | | |
| details: {"development": {"refId": null, "useDefault": true}, "types": [{"id": "BUGFIX", "displayName": "BUGFIX", "enabled": true, "prefix": "bugfix/"}, {"id": "FEATURE", "displayName": "FEATURE", "enabled": true, "prefix": "feature/"}, {"id": "HOTFIX", "displayName": "HOTFIX", "enabled": true, "prefix": "hotfix/"}, {"id": "RELEASE", "displayName": "RELEASE", "enabled": true, "prefix": "release/"}, {"scope": {"type": "PROJECT", "resourceId": 27}} | | | | |
| target: AUD | | | | |

Information for each event may include:

- **IP address** - IP address of the user who performed the action (though not recorded for system-generated events) Can also show the node IP address.
- **Node ID** - unique ID of the node where the action was performed
- **Method** - depending on how the action was performed, will be either Browser (end user) or System (system process)
- **Target** - a legacy attribute that represents the target of an action
- **Details** - a legacy attribute containing additional information about event details
- **Load balancer/proxy** - shown while using a load balancer or proxy

 Some of the information in each event is not available for events logged by Bitbucket 6.x.

In addition to viewing all events in the global audit page, administrators, system administrators, and delegated administrators can also see a list of events for each project and repository by going to the settings and selecting **Audit log** (under Security). These audit logs display a subset of the events recorded in the log file.



Accessing audit logs

You can find the log file in the `<home directory>/log/audit` directory. On clustered Bitbucket Data Center deployments, each application node will have its own log in the local `<home directory>/log/audit` directory. The audit log file is used primarily for integrating with third-party logging platforms.

Refer to [Audit log integrations](#) for detailed information about the log file.

i All audit log events are stored in the database. There is a limit of 10 million events logged in the database. When that limit is reached, the oldest records will be deleted as necessary.

Audit log events from previous versions of Bitbucket

Any events that were logged before you upgraded to Bitbucket 7.x:

- wont be visible until after the migration task completes in the background
- will appear as two separate entries in the list
- wont contain details like Source, Node ID, and Method

Adjusting data retention and selecting which events to log

In the audit log settings you can decide how long you want to retain the logged events in the database and the areas from which you want to collect the logs.

Setting the database retention period

You can decide to retain the data in the database for a maximum of 99 years, however, setting long retention periods can increase the size of your DB and affect performance.

To set the retention period:

1. In the administration area, go to > **Settings**.
2. Adjust the **Database retention period**.
3. **Save** your changes.

i If you limit the retention period, all the events that exceed the newly set period will be deleted from the database and from the UI, however, they will be retained in the audit log file.

Selecting events to log

The events that are logged are organized in categories that belong to specific coverage areas. For example, mirror-related events are logged in the Global administration category that belongs to the Global configuration and administration coverage area. For all coverage areas and events logged in each area, see [Audit log events](#).

Audit log settings

Audit log database storage

Your retention period sets the length of time we store logs in your database, with a maximum of 10,000,000 records. Check our documentation to learn how and where logs are stored. [Learn more](#)

Database retention period

Years ⚠ Keep an eye on your database size

Audit log file retention

Each file has a size limit of 100 MB. A new file is created every 24 hours or when size limit is reached. You should allocate enough disk space to store the files.

Number of files stored

files per node

Coverage

Select the areas you want to log. [Learn about logged events](#)

| Coverage area | Coverage level i |
|---|-------------------------|
| Global configuration and administration
Log instance or system admin actions around instance administration or configuration such as platform changes or upgrades to global settings. | Base ⌵ |
| User management
Log actions around users, groups, memberships, and roles such as adding and removing users and groups. | Base ⌵ |
| Permissions
Log actions around local and global permissions and configurations such as changing to anonymous access or update group permissions. | Base ⌵ |
| Local configuration and administration
Log admin actions around spaces, projects or repos such as creating or deleting a project or space, or updates to a repository. | Base ⌵ |
| Security
Log user actions related to security such as authentication, granted site access or created group. | Base ⌵ |

To adjust the coverage:

In the administration area, go to > **Settings**.

In the **Coverage level** drop-down, choose **Base** to log the most important events or **Off** to stop collecting events from a particular area.

Coverage levels reflect the number and frequency of events that are logged.

Off: Turns off logging events from this coverage area.

Base: Logs low-frequency and some of the high-frequency core events from selected coverage areas. Note that the base level is the only logging level available for Bitbucket Server.

The following coverage levels are only available in Bitbucket Data Center:

Advanced: Logs everything in Base, plus additional events where available.

Full: Logs all the events available in Base and Advanced, plus additional events for a comprehensive audit.

Exporting audit log events

You can export up to 100,000 events as a CSV file. If you have more events than that, only the 100,000 newest events are included in the export. In Bitbucket Data Center, you can also export up to 100k filtered events based on your current search.

To export audit log events:

1. On the **Audit log** page, select **Export**.
2. Select **Filtered results** (Data Center only) or the **Latest 100k events**.
3. Select **Export**.

Change the audit log file retention

You can choose how many audit log files to store in the local home directory on each node. By default, we store 100 files. Make sure you've provisioned enough disk space for these files, especially if you have set the logging level to Advanced or Full.

To change the file retention setting:

1. On the **Audit log** page, select ... **Settings**.
2. Enter the maximum number of files to be stored and select **Save**.

Once a node reaches the log file retention limit, the oldest one is deleted. If you need to keep these logs, for example for compliance purposes, you may want to manually back up the files in this directory on a regular basis, or send them to a third party logging platform. See Audit log integrations in Bitbucket.

Audit log events

The auditing component of Bitbucket Data Center and Server will log many different events that occur when being used. Events have been assigned a coverage level to reflect the number and frequency of events that are logged these levels can be used to control how much information is added to the audit log file. For example, if you have an instance under high load and no need for auditing in certain coverage areas, you may wish to turn audit logging off by selecting **Off** in the Audit log settings page. Learn more about these settings in [View and configure the audit log](#).

Coverage levels available with both a Server and Data Center license:

- **Off:** Turns off logging events from this coverage area.
- **Base:** Logs low-frequency and some of the high-frequency core events from selected coverage areas.

Coverage levels where events are only available with a Data Center license:

- **Advanced:** Logs the core events as well as the low and medium frequency events from the coverage areas.
- **Full:** Logs all the events available in Base and Advanced, plus additional events for a comprehensive audit.

 The events generated by external apps that call Jira REST API that fall into the Apps coverage area are not listed here because they are app-dependent.

The following tables provide lists of new and legacy event summaries for all coverage levels and categories.

On this page:

- [Global configuration and administration coverage area](#)
- [Global administration category](#)
- [Apps category](#)
- [User management coverage area](#)
- [Users and groups category](#)
- [Permission coverage area](#)
- [Permissions category](#)
- [Local configuration and administration coverage area](#)
- [Projects category](#)
- [Repositories category](#)
- [System category](#)
- [Security coverage area](#)
- [Auditing category](#)
- [Authentication category](#)
- [Security category](#)
- [End user activity coverage area](#)
- [Repositories category](#)
- [Pull requests category](#)
- [Search category](#)
- [Apps category](#)

Global configuration and administration coverage area

Global administration category

| | |
|-----------------|--|
| Base | <p>Instance setup completed (ApplicationSetupEvent)
 System backup cancelled (BackupCanceledEvent)
 System backup failed (BackupFailedEvent)
 System backup started (BackupStartedEvent)
 System backup succeeded (BackupSucceededEvent)
 Base URL changed (BaseUrlChangedEvent)
 Server name changed (DisplayNameChangedEvent)
 Elasticsearch settings changed (ElasticsearchConfigurationChangeAuditEvent)
 Elasticsearch full sync started (ElasticsearchFullSynchronisationAuditEvent)
 HTTP access to SCM hosting changed (HttpScmHostingChangedEvent)
 Product license changed (LicenseChangedEvent)
 Mail server changed (MailHostConfigurationChangedEvent)
 Database migration failed (MigrationFailedEvent)
 Database migration started (MigrationStartedEvent)
 Database migration succeeded (MigrationSucceededEvent)
 Database migration cancelled (MigrationCanceledEvent)
 Mirror disabled (MirrorDisabledEvent)
 Mirror enabled (MirrorEnabledEvent)
 Mirroring request rejected (MirroringRequestRejectedEvent)
 Mirroring request accepted (MirroringRequestAcceptedEvent)
 Mirroring request received (MirroringRequestCreatedEvent)
 Mirror installed (MirrorInstalledEvent)
 Mirror uninstalled (MirrorUninstalledEvent)
 Server email address changed (ServerEmailAddressChangedEvent)
 Server time zone changed (ServerTimeZoneChangedEvent)
 Resource throttled (TicketRejectedEvent)
 Jira site config created (JiraSiteConfigCreatedEvent)
 Jira site config updated (JiraSiteConfigUpdatedEvent)
 Jira site config deleted (JiraSiteConfigDeletedEvent)</p> |
| Advanced | <p>Default rate limiting settings changed (DefaultRateLimitSettingsModifiedEvent)
 Rate limiting disabled (RateLimitingDisabledEvent)
 Rate limiting enabled (RateLimitingEnabledEvent)
 User rate limiting settings created (UserRateLimitSettingsCreatedEvent)
 User rate limiting settings deleted (UserRateLimitSettingsDeletedEvent)
 User rate limiting settings changed (UserRateLimitSettingsModifiedEvent)
 SSH settings changed (SshScmHostingChangedEvent)
 LFS enabled (GitLfsFeatureEnabledEvent)
 LFS disabled (GitLfsFeatureDisabledEvent)
 Announcement banner created (AnnouncementBannerCreatedEvent)
 Announcement banner deleted (AnnouncementBannerDeletedEvent)
 Announcement banner updated (AnnouncementBannerUpdatedEvent)
 Logging settings enabled (LoggingSettingsEnabledEvent)
 Logging settings disabled (LoggingSettingsDisabledEvent)
 Profiling settings disabled (ProfilingSettingsDisabledEvent)
 Profiling settings enabled (ProfilingSettingsEnabledEvent)
 Application link created (ApplicationLinkAddedEvent)
 Application link edited (ApplicationLinkUpdatedEvent)
 Application link deleted (ApplicationLinkDeletedEvent)</p> |
| Full | No additional events available |

Apps category

| | |
|-----------------|---|
| Base | <p>Plugin disabled (PluginDisabledEvent)
 Plugin enabled (PluginEnabledEvent)
 Plugin uninstalled (PluginUninstalledEvent)
 Plugin upgraded (PluginUpgradedEvent)</p> |
| Advanced | No additional events available |

| | |
|-------------|--|
| Full | Plugin container unavailable (PluginContainerUnavailableEvent)
Plugin module disabled (PluginModuleDisabledEvent)
Plugin module enabled (PluginModuleEnabledEvent)
Plugin module available (PluginModuleAvailableEvent)
Plugin module unavailable (PluginModuleUnavailableEvent)
Plugin framework started (PluginFrameworkStartedEvent) |
|-------------|--|

User management coverage area

Users and groups category

| | |
|-----------------|---|
| Base | Personal access token created (AccessTokenCreatedEvent)
Personal access token deleted (AccessTokenDeletedEvent)
Personal access token changed (AccessTokenModifiedEvent)
User group automatically created (AutoGroupCreatedEvent)
User automatically deleted from user group (AutoGroupMembershipDeletedEvent)
User automatically created (AutoUserCreatedEvent)
User directory created (DirectoryCreatedEvent)
User directory deleted (DirectoryDeletedEvent)
GPG key added (GpgKeyCreatedEvent)
GPG key deleted (GpgKeyDeletedEvent)
User group created (GroupCreatedEvent)
User group deleted (GroupDeletedEvent)
User group updated (GroupUpdatedEvent)
User deleted from user group (GroupMembershipDeletedEvent)
User added to user group ((GroupMembershipsCreatedEvent)
Group added to user group (GroupMembershipsCreatedEvent)
User created (UserCreatedEvent)
User created from directory sync (UserCreatedFromDirectorySynchronisationEvent)
User password changed UserCredentialUpdatedEvent)
User deleted (UserDeletedEvent)
User erased (UserErasedEvent)
Username changed (UserRenamedEvent) |
| Advanced | No additional events available |
| Full | No additional events available |

Permission coverage area

Permissions category

| | |
|-----------------|--|
| Base | Global permission granted (GlobalPermissionGrantedEvent)
Global permission requested (GlobalPermissionGrantRequestedEvent)
Global permission change request (GlobalPermissionModificationRequestedEvent)
Global permission changed (GlobalPermissionModifiedEvent)
Global permission remove request (GlobalPermissionRevocationRequestedEvent)
Global permission removed
Project permission granted (ProjectPermissionGrantedEvent)
Project permission requested (ProjectPermissionGrantRequestedEvent)
Project permission change request (ProjectPermissionModificationRequestedEvent)
Project permission changed (ProjectPermissionModifiedEvent)
Project permission remove request (ProjectPermissionRevocationRequestedEvent)
Project permission removed (ProjectPermissionRevokedEvent)
Repository permission granted (RepositoryPermissionGrantedEvent)
Repository permission requested (RepositoryPermissionGrantRequestedEvent)
Repository permission change request (RepositoryPermissionModificationRequestedEvent)
Repository permission changed (RepositoryPermissionModifiedEvent)
Repository permission remove request (RepositoryPermissionRevocationRequestedEvent)
Repository permission removed (RepositoryPermissionRevokedEvent)
Repository auto-merge configuration changed (RepositoryAutoMergeConfigUpdatedEvent) |
| Advanced | No additional events available |
| Full | No additional events available |

Local configuration and administration coverage area

Projects category

| | |
|-------------|---|
| Base | Hook configuration removed (HookScriptConfigurationRemovedEvent)
Hook configuration set (HookScriptConfigurationSetEvent)
Hook created (HookScriptCreatedEvent)
Hook deleted (HookScriptDeletedEvent)
Hook deleted by app (HookScriptsDeletedByPluginKeyEvent)
Hook changed (HookScriptUpdatedEvent)
Project avatar changed (ProjectAvatarUpdatedEvent)
Project created (ProjectCreatedEvent)
Project creation requested (ProjectCreationRequestedEvent)
Project deleted (ProjectDeletedEvent)
Project deletion requested (ProjectDeletionRequestedEvent)
Project settings change requested (ProjectModificationRequestedEvent)
Project imported (ProjectImportedEvent)
Project settings changed ((ProjectModifiedEvent)
SSH access key added to project (SshAccessKeyGrantedEvent)
SSH access key deleted from project (SshAccessKeyRevokedEvent)
Project branch model created (ProjectBranchModelConfigurationCreatedEvent)
Project branch model deleted (ProjectBranchModelConfigurationDeletedEvent)
Project branch model updated (ProjectBranchModelConfigurationUpdatedEvent)
Project branch permission added (ProjectRefRestrictionAddedEvent)
Project branch permission deleted (ProjectRefRestrictionDeletedEvent)
Project branch permission updated (ProjectRefRestrictionUpdatedEvent)
Project code insight condition added (ProjectInsightReportConditionAddedEvent)
Project code insight condition deleted (ProjectInsightReportConditionDeletedEvent)
Project code insight condition changed (ProjectInsightReportConditionUpdatedEvent)
Project auto-merge configuration changed (RepositoryAutoMergeConfigUpdatedEvent)
Project default reviewers added (ProjectPullRequestConditionCreatedEvent)
Project default reviewers deleted (ProjectPullRequestConditionDeletedEvent)
Project default reviewers updated (ProjectPullRequestConditionUpdatedEvent)
Project auto-decline settings created (ProjectAutoDeclineSettingsCreatedEvent)
Project auto-decline settings deleted (ProjectAutoDeclineSettingsDeletedEvent)
Project auto-decline settings changed (ProjectAutoDeclineSettingsUpdatedEvent) |
|-------------|---|

| | |
|-----------------|--|
| Advanced | <ul style="list-style-type: none"> Project pull request merge config deleted (ProjectPullRequestMergeConfigDeletedEvent) Project pull request merge config updated (ProjectPullRequestMergeConfigUpdatedEvent) Pull request description template created (PullRequestTemplateCreatedEvent) Pull request description template deleted (PullRequestTemplateDeletedEvent) Pull request description template updated (PullRequestTemplateUpdatedEvent) Pull request reviewer group created (ReviewerGroupCreatedEvent) Pull request reviewer group deleted (ReviewerGroupDeletedEvent) Pull request reviewer group updated (ReviewerGroupUpdatedEvent) |
| Full | No additional events available |

Repositories category

| | |
|-------------|---|
| Base | <ul style="list-style-type: none"> Repository webhook created (InternalWebhookCreatedEvent) Repository webhook deleted (InternalWebhookDeletedEvent) Repository webhook changed (InternalWebhookModifiedEvent) Repository created (RepositoryCreatedEvent) Repository failed to create (RepositoryCreationFailedEvent) Default branch changed (RepositoryDefaultBranchModifiedEvent) Repository deleted (RepositoryDeletedEvent) Repository deletion requested (RepositoryDeletionRequestedEvent) Repository forked (RepositoryForkedEvent) Repository fork failed (RepositoryForkFailedEvent) Repository hook deleted (RepositoryHookDeletedEvent) Repository hook disabled (RepositoryHookDisabledEvent) Repository hook enabled (RepositoryHookEnabledEvent) Repository hook settings changed (RepositoryHookSettingsChangedEvent) Repository imported (RepositoryImportedEvent) Repository change requested (RepositoryModificationRequestedEvent) Repository settings changed (RepositoryModifiedEvent) Elasticsearch repository sync completed (ElasticsearchRepositorySynchronisationAuditEvent) Repository branch model created (RepositoryBranchModelConfigurationCreatedEvent) Repository branch model deleted (RepositoryBranchModelConfigurationDeletedEvent) Repository branch model updated (RepositoryBranchModelConfigurationUpdatedEvent) Repository branch permission added (RepositoryRefRestrictionAddedEvent) Repository branch permission deleted (RepositoryRefRestrictionDeletedEvent) Repository branch permission updated (RepositoryRefRestrictionUpdatedEvent) Repository code insight condition added (RepositoryInsightReportConditionAddedEvent) Repository code insight condition deleted (RepositoryInsightReportConditionDeletedEvent) Repository code insight condition changed (RepositoryInsightReportConditionUpdatedEvent) Repository default reviewers added (RepositoryPullRequestConditionCreatedEvent) Repository default reviewers deleted (RepositoryPullRequestConditionDeletedEvent) Repository default reviewers updated (RepositoryPullRequestConditionUpdatedEvent) Repository auto-decline settings created (RepositoryAutoDeclineSettingsCreatedEvent) Repository auto-decline settings deleted (RepositoryAutoDeclineSettingsDeletedEvent) Repository auto-decline settings changed (RepositoryAutoDeclineSettingsUpdatedEvent) Repository required build merge check created (RequiredBuildConditionCreatedEvent) Repository required build merge check deleted (RequiredBuildConditionDeletedEvent) Repository required build merge check updated (RequiredBuildConditionUpdatedEvent) |
|-------------|---|

| | |
|-----------------|---|
| Advanced | Repository LFS enabled (GitLfsEnabledEvent)
Repository LFS disabled (GitLfsDisabledEvent)
Repository pull request merge config deleted (RepositoryPullRequestMergeConfigDeletedEvent)
Repository pull request merge config updated (RepositoryPullRequestMergeConfigUpdatedEvent)
Repository transcode diff setting enabled (GitTranscodeDiffEnabledEvent)
Repository transcode diff setting disabled (GitTranscodeDiffDisabledEvent)
Pull request reviewer group created (ReviewerGroupCreatedEvent)
Pull request reviewer group deleted (ReviewerGroupDeletedEvent)
Pull request reviewer group updated (ReviewerGroupUpdatedEvent) |
| Full | No additional events available |

System category

| | |
|-----------------|--|
| Base | No additional events available |
| Advanced | SCM pull request merge config deleted (ScmPullRequestMergeConfigDeletedEvent)
SCM pull request merge config updated (ScmPullRequestMergeConfigUpdatedEvent) |
| Full | No additional events available |

Security coverage area

Auditing category

| | |
|-----------------|---|
| Base | Audit log search performed
Audit log exported
Audit log configuration updated |
| Advanced | No events available |
| Full | No events available |

Authentication category

| | |
|-----------------|--|
| Base | No events available |
| Advanced | User login failed (AuthenticationFailureEvent)
User logged in (AuthenticationSuccessEvent)
User login failed (SSH) (SshAuthenticationFailureEvent)
User logged in (SSH) (SshAuthenticationSuccessEvent)
User logged out (LogoutSuccessEvent) |
| Full | No events available |

Security category

| | |
|-----------------|---|
| Base | No events available |
| Advanced | Unauthorized access to a resource (AuthorizationFailureEvent) |
| Full | No events available |

End user activity coverage area

Repositories category

| | |
|-----------------|--|
| Base | Repository accessed by user (RepositoryAccessedEvent)
Run build (AnalyticsActionRunEvent) |
| Advanced | Repository watcher added (RepositoryWatcherAddedEvent)
Repository watcher removed (RepositoryWatcherRemovedEvent)
Repository notification settings updated (RepositoryNotificationSettingsUpdatedEvent)
Branch created (BranchCreatedEvent)
Git archive created (ContentArchiveSuccessfulEvent)
Git archive failed (ContentArchiveFailedEvent)
Branch deleted (BranchDeletedEvent)
Diff succeeded (DiffSuccessfulEvent) |
| Full | Changes read from repository (RepositoryOtherReadEvent)
Repository written to (RepositoryOtherWriteEvent)
Repository pulled (RepositoryPullEvent)
Changes pushed to repository (RepositoryPushEvent)
Git hook activity (RepositoryHookEvent)
Repository cloned (RepositoryCloneEvent) |

Pull requests category

| | |
|-----------------|--|
| Base | Auto merge failed (AutomaticMergeStoppedEvent)
Auto merge succeeded (AutomaticMergeSucceededEvent)
Pull request approved by participant (PullRequestParticipantApprovedEvent)
Pull request reviewed by participant (PullRequestParticipantReviewedEvent)
Pull request unapproved by participant (PullRequestParticipantUnapprovedEvent)
Pull request participants changed (PullRequestParticipantsUpdatedEvent)
Pull request declined (PullRequestDeclinedEvent)
Pull request deleted (PullRequestDeletedEvent)
Pull request merged (PullRequestMergedEvent)
Pull request opened (PullRequestOpenedEvent)
Pull request reopened (PullRequestReopenedEvent)
Pull request reviewers changed (PullRequestReviewersUpdatedEvent)
Pull request open request (PullRequestOpenRequestedEvent) |
| Advanced | Pull request rebased (PullRequestRebasedEvent)
Pull request watcher added (PullRequestWatcherAddedEvent)
Pull request watcher removed (PullRequestWatcherRemovedEvent)
Pull request filters used (PullRequestFilterEvent) |
| Full | Pull request comment reply added (PullRequestCommentRepliedEvent)
Pull request comment created (PullRequestCommentAddedEvent)
Pull request comment deleted (PullrequestCommentDeletedEvent)
Pull request comment changed (PullRequestCommentEditedEvent)
Pull request task created (PullRequestTaskAddedEvent)
Pull request task deleted (PullRequestTaskDeletedEvent)
Pull request task changed (PullRequestTaskEditedEvent) |

Search category

| | |
|-----------------|---------------------|
| Base | No events available |
| Advanced | No events available |

| | |
|-------------|--|
| Full | Repository search succeeded (RepositoriesSearchSuccessfulEvent)
Code search succeeded (CodeSearchSuccessfulEvent) |
|-------------|--|

Apps category

This category is for auditing events generated by third-party apps.

 Bitbucket Server customers can set the [configuration property](#) `audit.legacy.events.logging.forced=true` to move the following events from **Full** to **Base** level:

- Plugin container unavailable, Plugin module disabled, Plugin module enabled, Plugin module available, Plugin framework started
- User log in failed, User logged in, User logged in (SSH)
- Repository read event, Repository write event, Repository pull event, Repository push event, Git hook activity, Repository cloned

Note that adding these events to **Base** can significantly increase the size of the audit log.

Audit log integrations

Bitbucket Data Center and Server writes audit logs to the database and a log file. By itself, the log file saves you the effort of periodically exporting your audit logs from the database for long-term storage. However, the main purpose of the file is to easily integrate Bitbucket to a third-party logging platform.

Selecting which events to log and adjusting data retention

The [Audit log settings](#) menu controls the coverage of audit logs in both database and log file.

The log file's retention is ultimately controlled by [log rotation](#). We use basic log rotation to manage the volume of logs. We automatically archive the audit logfile when:

- the node's time reaches 12:00 midnight, or
- the audit log file reaches 100MB.

Once a node reaches the log file retention limit, the oldest one is deleted. By default the limit is 100 log files (the current audit log file + 99 archives). Make sure you allocate enough disk space for these log files on each application node. For the default setting of 100 files, you should allow 10GB.

To customize the log rotation rules (and, ultimately, the retention rules), use the following `bitbucket.properties` file parameters:

- `com.atlassian.audit.file.max.file.size` controls the maximum size (in MB) for the current audit log file before it is archived.
- `com.atlassian.audit.file.max.file.count` controls the maximum number of audit log files (counting the current audit log file and all archived log files).

Both default to 100. If you adjust either of these values, make sure you allocate the right amount of space on each application node. For example, if you set `com.atlassian.audit.file.max.file.count=150`, you should allocate at least 15GB just for log files on each application node.

For more information on using the `bitbucket.properties` file, [click here](#).

Log file details

Bitbucket Server writes audit logs in real time to the home directory. Specifically, these logs are written to the audit logfile. On clustered Bitbucket Data Center deployments, each application node will produce its own logfile in its local home directory.

Location

To integrate the audit logfile with a third-party logging platform, you'll need to know its exact location. This may vary, depending on how you configured your home directory. For more information about the local home directory, [click here](#).

On a clustered Bitbucket Data Center deployment, the audit logfile's directory should be the same on all nodes.

See [CloudWatch Logs Agent Reference](#) for more information. If you want to see how we automate this via Ansible, check out our deployment playbooks on <https://bitbucket.org/atlassian/dc-deployments-automation/src/master/>.

File name

The audit log file name uses the following naming convention:

```
YYYYMMDD-XXXXX.audit.log
```

The `XXXXX` portion is a 5-digit number (starting with `00000`) tracking the number of audit log files archived in the same day (`YYYYMMDD`). For example, if there are 5 archived log files today (January 1, 2020), then:

- the oldest archived log file is `20200101.00000.audit.log`
- the current audit log file is `20200101.00005.audit.log`

Format

Each audit log is written as a JSON entry to the audit log file. Every line in the file represents a single event, allowing you to use [regular expressions](#) to do simple searches if needed.

Integrating with logging agents

Most enterprise environments use a third-party logging platform to aggregate, store, and otherwise manage logs from all hosts. Logging platforms like AWS CloudWatch and Splunk use *agents* to collect logs from every host in the environment. These agents are installed on each host, collecting local logs and sending them back to a centralized location to be aggregated, analyzed, audited, and/or stored.

If your logging platform uses agents this way, you can configure each node's agent to monitor the audit log file directly. Logging agents from most major platforms (including AWS CloudWatch, Splunk, ELK, and Sumo Logic) are compatible with the audit logfile.

Amazon CloudWatch Agent

We provide [Quick Starts for Bitbucket Data Center](#) for easy deployments on AWS. This Quick Start lets you deploy Bitbucket Data Center along with an Amazon CloudWatch instance to monitor it.

To set up Amazon CloudWatch, use the **Enable CloudWatch Integration** parameter's default setting (namely, `Metrics and Logs`). The Quick Start will then configure the [Amazon CloudWatch Agent](#) to collect the logs from each node's audit log files. The agent will send these logs to a separate log group named `bitbucket-<aws-stack-name>-audit`.

Our Quick Start also sets up a default dashboard to help you read the collected data, including logs from each audit logfile. Refer to [Working With Log Groups and Log Streams](#) for related information.

Manual configuration

If needed, you can also manually configure the Amazon CloudWatch agent to collect the audit log files. To do this, set the following parameters in the Agent Configuration File:

- `file`: set this to `<local home directory>/log/audit/*`. Don't forget to set the absolute path to the [home directory](#).
- `log_group_name` and `log_stream_name`: use these to send Bitbucket Data Center's audit logs to a specific log group or stream.

Splunk Universal Forwarder

For Splunk Enterprise or Splunk Cloud, you can use the [Splunk Universal Forwarder](#) as your logging agent. This will involve installing the universal forwarder on each application node.

You'll also need to define each node's audit log directory as one of the forwarder's inputs. This will set the forwarder to send all logs from the audit log directory to a pre-configured [receiver](#). One way to define the forwarder's inputs is through the Splunk CLI. For Linux systems, use the following command on each application node:

```
./splunk add monitor <local home directory>/log/audit/*audit.log
```

Refer to the following links for detailed instructions on configuring the Splunk Universal Forwarder on each node:

- [How to forward data to Splunk Enterprise](#)

- [How to forward data to Splunk Cloud](#)

Filebeat (for the ELK stack)

Within the [ELK stack](#), you can use the [Filebeat](#) plugin to collect logs from each node's audit log files. Each time a log is written to the current audit log file, Filebeat will forward that log to Elasticsearch or Logstash.

To set this up, [install Filebeat](#) first on each application node. Then, set the audit log file directory as a [Filebeat input](#). To do that, add its directory as a path in the `filebeat.inputs` section of each node's `filebeat.yml` configuration file. For example:

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
  - <local home directory>/log/audit/
```

Sumo Logic installed collectors

If you have a Sumo Logic instance, you can use [installed collectors](#) to collect logs from each node's audit log files. To do this, [install a collector](#) on each node first. Then, add `<local home directory>/log/audit/*` as a [Local File Source](#) to each node's collector.

Deprecated audit log file format

Previous releases of Bitbucket Server also generated an audit log file, but this file used a different format. This format is now deprecated. If you require logs generated in this format, you can configure Bitbucket to also generate the legacy format alongside the current one. However, we recommend that you use the current audit log file, as we will remove the legacy format in Bitbucket 8.0.

The legacy audit log file will have the same set of defaults and settings as Bitbucket Server releases before 7.0, such as:

- The file will rotate at 25 MB.
- There will be a 100-file limit to the number of the legacy audit log files that Bitbucket keeps. When the limit is reached, the oldest file is deleted each day.

To enable the legacy audit log file, set `audit.legacy.log=true` in the `bitbucket.properties` file. For more information on using this file, click [here](#).

Push logs

Bitbucket Data Center and Server retains logs of important audit events with the auditing feature. However, when it comes to knowing who pushed certain commits and who was responsible for creating or deleting branches, a push log can be used by anyone that can administer the repository.

Viewing a push log

1. In the Repository settings, go to **Push log** (under Security).
2. Review any push events, created branch events, deleted branch events, modified files, and merge events.

| Push date | Pushed by | From | To | Action |
|-------------|--------------|-------------|-------------|---------|
| 06 Mar 2020 | Kim Tan | 0ff514c6073 | 55ccd6eda64 | PUSH |
| 21 Aug 2019 | Kim Tan | 11eea8070fc | 0ff514c6073 | MERGE |
| 05 Aug 2019 | Kim Tan | cf22c452356 | 11eea8070fc | MERGE |
| 24 May 2019 | Zoe Trakosas | f00e1cb13d9 | cf22c452356 | MERGE |
| 24 May 2019 | Kim Tan | cc1fc49bc88 | f00e1cb13d9 | MERGE |
| 09 Apr 2019 | Zoe Trakosas | a38f43dc79b | cc1fc49bc88 | PUSH ⓘ |
| 22 Jan 2019 | Zoe Trakosas | e6f54de0e42 | a38f43dc79b | MERGE |
| 22 Jan 2019 | Zoe Trakosas | c9f55976308 | e6f54de0e42 | PUSH ⓘ |
| 15 Nov 2018 | Zoe Trakosas | fdab7da736c | c9f55976308 | MERGE |
| 30 Jul 2018 | Kim Tan | 841feb0a05f | fdab7da736c | PUSH ⓘ |
| 23 Apr 2018 | Kim Tan | Created | 841feb0a05f | UNKNOWN |

Each row in the push log represents:

- **Push date** a date for the push
- **Pushed by** who performed the push
- **From/To** what range of commits were affected
 - Created the first time a user pushes their branch
 - Deleted a branch has been deleted
- **Action** trigger types for each event, which can be any of the following:
 - Merge
 - Push
 - Forced push
 - Create branch
 - Delete branch
 - File edit
 - Unknown ([automatic branch merges](#), [imported repository](#), [pull request rebase](#))

Tracking push events

Bitbucket will track whether updates are forced or not and display these events. Previously, tracking was not available, so we may not know whether they were forced. In these cases, there will be an ⓘ with a tooltip next to the action trigger to indicate this status.

Filtering branches

By default, you'll see the repository's default branch (typically `master`). You can search for history and details by another branch (even a deleted one) in the push log. For example; see what commit a branch pointed to before it was deleted, or who created a branch and when.

To view how and when a branch was deleted:

1. In the **Push log** page, from the **Branch** dropdown, select any deleted branch.
2. Once selected, the events results are displayed.

| Push date | Pushed by | From | To | Action |
|------------|---|----------------------------|----------------------------|-------------------------------|
| 2 days ago |  Alexandre Carlton | 1c0ddb6ad1 | Deleted | DELETE BRANCH |
| 2 days ago |  Alexandre Carlton | Created | 1c0ddb6ad1 | CREATE BRANCH |

Update your license key

When you upgrade or renew your Bitbucket Data Center and Server license, you'll receive a new license key. To update your license key:

1. Go to  > **Licensing**.
2. Select **Edit license**.
3. Enter your new license.
4. Select **Save**.

 You can access your license key at <http://my.atlassian.com>

For more information about licensing or to find out more about starter licenses, see our [licensing FAQ](#).

Configuration properties

This page describes the configuration properties that can be used to control behavior in Bitbucket Server. Create the `bitbucket.properties` file, in the shared folder of your [home directory](#), and add the system properties you need, use the standard format for Java properties files.

Note that `bitbucket.properties` is created automatically when you perform a [database migration](#).

Bitbucket Server must be restarted for changes to become effective.

Default values for system properties, where applicable, are specified in the tables below.

On this page:

- [Analytics](#)
- [Application mode](#)
- [Attachments](#)
- [Audit](#)
- [Authentication](#)
- [Avatars](#)
- [Backups](#)
- [Branch Information](#)
- [Branch Utils](#)
- [Builds](#)
- [Clustering](#)
- [Code Insights](#)
- [Comment Likes and Comment Reactions](#)
- [Commit Indexing](#)
- [Commit graph cache](#)
- [Commits](#)
- [Content](#)
- [Data Center Migration](#)
- [Database](#)
- [Database Pool](#)
- [Database Schema](#)
- [Diagnostics](#)
- [Display](#)
- [Downloads](#)
- [Elasticsearch](#)
- [Events](#)

- [Executor](#)
- [Features](#)
- [File editor](#)
- [Footer](#)
- [Graceful shutdown](#)
- [Hibernate](#)
- [Hook Scripts](#)
- [Importer](#)
- [JMX](#)
- [Jira](#)
- [Jira Automatic Transition Trigger Events](#)
- [Jira cloud development information events](#)
- [Job Scheduler](#)
- [Liquibase](#)
- [Logging](#)
- [Migration](#)
- [Mirroring](#)
- [Notifications](#)
- [POST service webhook](#)

- [Paging](#)
- [Password Reset](#)
- [Process execution](#)
- [Profiling](#)
- [Pull Request - Reviewer Groups](#)
- [Pull Request - Suggestions](#)
- [Pull Request Commit Indexing](#)
- [Pull Requests](#)
- [Ref metadata](#)
- [Ref searching](#)
- [Repository shortcuts](#)
- [Resource Throttling](#)
- [Rest](#)
- [SCM - Cache](#)
- [SCM - Misc](#)
- [SCM - git](#)
- [SMTP](#)
- [SSH](#)
- [SSH command execution](#)
- [SSH security](#)
- [Search](#)
- [Server](#)
- [Server Connectors](#)
- [Server busy banners](#)
- [Setup automation](#)
- [Syntax highlighting](#)
- [Tasks](#)
- [Topic](#)
- [Webhooks](#)
- [Zero downtime backup /disaster recovery](#)

Analytics

| Default value | Description |
|-----------------------|--|
| analytics.aws.enabled | |
| true | Controls whether AWS instance analytics events are published. This setting only has an effect if analytics is enabled. |

Application mode

| Default value | Description |
|------------------|---|
| application.mode | |
| default | Controls what mode Bitbucket is in - currently "mirror" and "default" are supported |

Attachments

| Default value | Description |
|----------------------------|-------------|
| attachment.upload.max.size | |

| | |
|----------|--|
| 10485760 | <p>Defines the largest single attachment the system will allow. Attachments larger than this will be rejected without ever being stored locally.</p> <p>This setting applies to all uploaded data, including avatars. Some types may have their own further restriction they apply on top of this. However, for such types, the file <i>will</i> be stored locally prior to that secondary limit being applied. If that secondary limit is higher than this limit, this limit is the one that will be applied.</p> <p>This value is in bytes.</p> |
|----------|--|

Audit

These properties control the auditing feature, determining the number of audit entries logged, or stored in the database, and the size of those entries. Changing these settings will only affect new audit entries.

Increasing the amount of auditing done may have an adverse effect on performance.

| Default value | Description |
|---------------|---|
| | <code>audit.details.max.length</code> |
| 1024 | Defines the number of <i>characters</i> that can be can stored as details in a single audit entry. This property is only applicable to legacy (pre 7.0) audit log file. This property is effective only when <code>audit.legacy.log=true</code> |
| | <code>audit.legacy.events.logging.forced</code> |
| false | This property controls whether ADVANCED or FULL level logging is enforced for actions that were audited prior to application version 7.0. If this property is set as true, those actions will be audited regardless of coverage settings. |
| | <code>audit.legacy.log</code> |
| false | In version 7.0, a new format of audit log file was introduced. This property controls whether audit events are logged to the legacy (pre 7.0) audit log file, in addition to the new format. This property exists to ease the transition from the legacy audit log file to the new format and will be removed in 8.0. |
| | <code>plugin.audit.search.max.concurrent.nontext.requests</code> |
| 10 | Maximum number of concurrent non-freetext search requests allowed, defaults to 10 per node |
| | <code>plugin.audit.search.max.concurrent.text.requests</code> |
| 5 | Maximum number of concurrent freetext search requests allowed, defaults to 10 per node |
| | <code>plugin.audit.search.query.timeout</code> |
| 30 | Timeout in seconds for a queued search request, defaults to 30 seconds |
| | <code>plugin.audit.db.limit.rows</code> |
| 10000000 | Maximum number of audit event rows stored in DB, events exceeding the limit get deleted in time order, defaults to 10M checked on hourly basis |
| | <code>plugin.audit.db.limit.buffer.rows</code> |
| 1000 | Buffer to accommodate new audit events, defaults to 1000 rows |
| | <code>plugin.audit.db.delete.batch.limit</code> |

| | |
|--|--|
| 10000 | maximum number of events to be deleted per database transaction used when enforcing retention limits, defaults to 10,000 rows |
| plugin.audit.log.view.sysadmin.only | |
| false | Only allows the the system admin (but not admin) to see the global audit log |
| plugin.audit.schedule.db.limiter.interval.mins | |
| 60 | Database size check, running every 60 minutes |
| plugin.audit.broker.exception.loggedCount | |
| 3 | Maximum number of audit events written to system log file in case of error, defaults to 3 |
| plugin.audit.retention.interval.hours | |
| 23 | Database retention check, which deletes events exceeding retention period, running every day at midnight, and only runs if the last run is more than 23 hours. |
| plugin.audit.file.max.file.size | |
| 100 | Size limit in megabytes for individual audit file, file rotates when limit is reached, defaults to 100MB |
| plugin.audit.file.max.file.count | |
| 100 | Maximum number of audit files, the earliest file will be deleted when limit is reached, defaults to 100 |
| plugin.audit.consumer.buffer.size | |
| 10000 | Maximum number of audit events kept in buffer waiting to be consumed, defaults to 10,000 |
| plugin.audit.broker.default.batch.size | |
| 3000 | Maximum number of audit events dispatched to consumer, defaults to 3,000 per batch |
| plugin.audit.coverage.cache.read.expiration.seconds | |
| 30 | How long the coverage cache is valid, defaults to 30 seconds |
| plugin.audit.distinct.categories.and.actions.cache.refresh.seconds | |
| 900 | How long the distinct categories and actions cache is refreshed, defaults to 900 seconds |
| plugin.audit.retention.file.configuration.expiration.seconds | |
| 300 | How frequently the retention file configuration cache (i.e. count) is refreshed, defaults to 300 seconds |

Authentication

See also [Connecting Bitbucket Server to Crowd](#).

| Default value | Description |
|----------------|-------------|
| auth.cache.tti | |

| | |
|--|---|
| 5 | <p>Controls the time-to-idle for entries in the authentication cache. A short TTI (5 to 10 seconds) helps narrow the window for malicious users authenticating with outdated credentials and is recommended. Setting this to a value less than 1 will default it to the configured TTL (<code>auth.cache.ttl</code>).</p> <p>This value is in seconds.</p> |
| <code>auth.cache.ttl</code> | |
| 30 | <p>Controls the time-to-live for entries in the authentication cache. Setting this to a value less than 1 will disable the cache. The maximum allowed value is 300 seconds (5 minutes). Longer TTLs pose a security risk, as potentially out-of-date credentials can still be used while they remain in the cache. A short TTL, like the default, can reduce burst load on remote authentication systems (Crowd, LDAP) while keeping potential exposure to outdated credentials low, especially when paired with a shorter TTI.</p> <p>This value is in seconds.</p> |
| <code>auth.remember-me.enabled</code> | |
| optional | <p>Controls whether remember-me authentication is disabled, always performed or only performed when a checkbox is checked on the login form. The 'Remember my login' checkbox is only displayed when set to 'optional'. Possible values are:</p> <ul style="list-style-type: none"> • <code>always</code>
No checkbox, remember-me cookie is always generated on successful login. • <code>optional</code>
Checkbox is displayed on login form. Remember-me cookie is only generated when checkbox is checked. • <code>never</code>
Remember-me authentication is disabled completely. |
| <code>auth.remember-me.cookie.name</code> | |
| <code>_atl_bitbucket_remember_me</code> | Defines the cookie name used for the remember-me authentication |
| <code>auth.remember-me.token.expiry</code> | |
| 30 | <p>How long remember-me tokens are valid. Note that once a remember-me token is used for authentication, the token is invalidated and a new remember-me token is returned.</p> <p>This value is in days.</p> |
| <code>auth.remember-me.token.grace.period</code> | |
| 60 | <p>How long a token can be re-used for authentication <i>after</i> it has been used to authenticate . This setting allows a grace period to allow parallel authentication attempts to succeed. This commonly happens when a browser is started and opens multiple tabs at once.</p> <p>This value is in seconds</p> |
| <code>auth.remember-me.token.cleanup.interval</code> | |
| 300 | <p>Controls how frequently expired remember-me tokens are cleaned up.</p> <p>This value is in minutes</p> |

| | |
|--|---|
| plugin.auth-crowd.sso.enabled | |
| false | Whether SSO support should be enabled or not. Regardless of this setting SSO authentication will only be activated when the system is connected to a Crowd directory that is configured for SSO. |
| plugin.auth-crowd.sso.config.ttl | |
| 15 | The auth plugin caches the SSO configuration that is retrieved from the remote Crowd server. This setting controls the time to live of that cache.

This value is in minutes . |
| plugin.auth-crowd.sso.config.error.wait | |
| 1 | If an error occurs while retrieving the SSO configuration from the remote Crowd server, the system will wait this long before retrying. The wait time between subsequent attempts is incremented exponentially (1s -> 1.5s -> 2.3s -> 3.4s, etc). The wait time is capped at the configured TTL.

This value is in seconds . |
| plugin.auth-crowd.sso.http.max.connections | |
| 20 | The maximum number of HTTP connections in the connection pool for communication with the Crowd server. |
| plugin.auth-crowd.sso.http.proxy.host | |
| | The name of the proxy server used to transport SOAP traffic to the Crowd server. |
| plugin.auth-crowd.sso.http.proxy.port | |
| | The connection port of the proxy server (must be specified if a proxy host is specified). |
| plugin.auth-crowd.sso.http.proxy.username | |
| | The username used to authenticate with the proxy server (if the proxy server requires authentication). |
| plugin.auth-crowd.sso.http.proxy.password | |
| | The password used to authenticate with the proxy server (if the proxy server requires authentication). |
| plugin.auth-crowd.sso.http.timeout | |
| 5000 | The HTTP connection timeout used for communication with the Crowd server. A value of zero indicates that there is no connection timeout.

This value is in milliseconds . |
| plugin.auth-crowd.sso.socket.timeout | |
| 20000 | The socket timeout. You may wish to override the default value if the latency to the Crowd server is high.

This value is in milliseconds . |
| plugin.auth-crowd.sso.session.validationinterval | |
| 3 | The number of minutes to cache authentication validation in the session. If this value is set to 0, each HTTP request will be authenticated with the Crowd server. |
| plugin.auth-crowd.sso.session.lastvalidation | |

| | |
|--|--|
| atl.
crowd.
sso.
lastva
lidati
on | The session key to use when storing a Date value of the user's last authentication. |
| plugin.auth-crowd.sso.session.tokenkey | |
| atl.
crowd.
sso.
tokenk
ey | The session key to use when storing a String value of the user's authentication token. |

Avatars

| Default value | Description |
|------------------------|--|
| avatar.max.dimension | |
| 1024 | <p>Controls the max height <i>and</i> width for an avatar image. Even if the avatar is within the acceptable file size, if its dimensions exceed this value for height <i>or</i> width, it will be rejected. When an avatar is loaded by the server for processing, images with large dimensions may expand from as small as a few kilobytes on disk to consume a substantially larger amount of memory, depending on how well the image data was compressed. Increasing this limit can <i>substantially</i> increase the amount of heap used while processing avatars and may result in OutOfMemoryErrors.</p> <p>This value is in pixels.</p> |
| avatar.max.size | |
| 1048576 | <p>Controls how large an avatar is allowed to be. Avatars larger than this are rejected and cannot be uploaded to the server, to prevent excessive disk usage.</p> <p>This value is in bytes.</p> |
| avatar.url.default | |
| mm | <p>Defines the fallback URL to be formatted into the "avatar.url.format.http" or "avatar.url.format.https" URL format for use when a user does not have an acceptable avatar configured. This value may be a URL or, if using Gravatar, it may be the identifier for one of Gravatar's default avatars.</p> |
| avatar.url.format.http | |

| | |
|---|--|
| <pre>http://www.gravatar.com/avatar/%1\$s.jpg?s=%2\$d&d=%3\$s</pre> | <p>Defines the default URL format for retrieving user avatars over HTTP. This default uses any G-rated avatar provided by the Gravatar service</p> <p>The following format parameters are available:</p> <ul style="list-style-type: none"> • %1\$s
The user's e-mail address, MD5 hashed, or "00000000000000000000000000000000" if the user has no e-mail. • %2\$d
The requested avatar size. • %3\$s
The fallback URL, URL-encoded, which may be defined using "avatar.url.default". • %4\$s
The user's e-mail address, not hashed, or an empty string if the user has no e-mail. |
| <p>avatar.url.format.https</p> | |
| <pre>https://secure.gravatar.com/avatar/%1\$s.jpg?s=%2\$d&d=%3\$s</pre> | <p>Defines the default URL format for retrieving user avatars over HTTPS. This default uses any G-rated avatar provided by the Gravatar service</p> <p>The following format parameters are available:</p> <ul style="list-style-type: none"> • %1\$s
The user's e-mail address, MD5 hashed, or "00000000000000000000000000000000" if the user has no e-mail. • %2\$d
The requested avatar size. • %3\$s
The fallback URL, URL-encoded, which may be defined using "avatar.url.default". • %4\$s
The user's e-mail address, not hashed, or an empty string if the user has no e-mail. |

Backups

| Default value | Description |
|---------------------------------|--|
| <p>backup.drain.scm.timeout</p> | |
| <p>60</p> | <p>Controls how long the backup should wait for outstanding SCM operations to complete before failing.</p> <p>This value is in seconds.</p> |
| <p>backup.drain.db.timeout</p> | |

| | |
|--|---|
| 90 | <p>Draining database connections during backup happens in two stages. Stage 1 passively waits a set amount of time for all connections to be returned to the pool. If connections are still leased when <code>backup.drain.db.timeout</code> seconds has elapsed then stage 2 begins and will interrupt the owning threads, wait <code>backup.drain.db.force.timeout</code> seconds and finally attempt to roll back and close any remaining connections.</p> <p>In stage 1 of draining connections during backup, this setting controls how long the backup should wait for outstanding database operations to complete before moving to stage 2. See <code>backup.drain.db.force.timeout</code></p> <p>This value is in seconds.</p> |
| <code>backup.drain.db.force.timeout</code> | |
| 30 | <p>In stage 2 of draining connections during backup, this property controls how long the backup process should wait (after interrupting the owning threads) for those threads to release the connections before forcibly rolling back and closing them. Note if all connections have been returned to the pool stage 2 is skipped and so this property has no effect. A negative value will skip stage 2 completely. See <code>backup.drain.db.timeout</code></p> <p>This value is in seconds.</p> |

Branch Information

| Default value | Description |
|---|--|
| <code>plugin.bitbucket-branch-information.timeout</code> | |
| 5 | <p>Controls timeouts for retrieving branch information, which for large repositories can be quite slow and consume a single Git process.</p> <p>This value is in seconds.</p> |
| <code>plugin.bitbucket-branch-information.max.branches</code> | |
| 1000 | Controls the maximum number of branches that are shown the user |

Branch Utils

| Default value | Description |
|--|---|
| <code>plugin.bitbucket-auto-merge.limit</code> | |
| 30 | <p>Defines the maximum number of merges allowed along in single automatic merge chain. If the number of merges included in the chain exceeds this limit, the <i>entire chain</i> will be skipped. Setting this limit to 0 means automatic merging is effectively disabled, since the merge chain must always be empty.</p> |
| <code>plugin.bitbucket-auto-merge.timeout</code> | |
| 300 | <p>Defines the maximum amount of time any command used to perform a single automatic merge from the chain is allowed to execute <i>or</i> idle. Since an automatic merge may require a series of different commands at the SCM level, this timeout does <i>not</i> define the upper bound for how long the overall merge process might take; it only defines the duration allotted to any single command.</p> <p>This value is in seconds.</p> |
| <code>plugin.bitbucket-branch-model.version.separator</code> | |

| | |
|--|--|
| [_\\- . +] | The version component separator for branch model. This value is a regular expression pattern that is used to split branch names parsed as version strings into separate components for comparison. The default value results in any of the following 4 characters being used for comparison: _ - + . |
| plugin.bitbucket-branch-model.validation.prefix.length | |
| 30 | The maximum allowed length of branch prefixes in branch model |

Builds

| Default value | Description |
|---------------------------------|--|
| build.actions.threads.max | |
| 1*\${scaling.concurrency} | When performing actions on a build status, this sets the upper limit to the number of concurrent threads that can be performing actions at the same time.

There is limited support for mathematical expressions; +, -, *, / and () are supported. |
| build.actions.cache.expiry | |
| 5 | The maximum number of minutes to wait for a response from a remote CI system.

This value is in minutes |
| build.actions.cache.max.pending | |
| 5000 | The maximum number of concurrent requests to a remote CI system. |

Clustering

| Default value | Description |
|---------------------------------------|--|
| hazelcast.node.authentication.enabled | |
| true | Enable node authentication. When this is enabled the group name and password are verified before any other checks are run |
| hazelcast.enterprise.license | |
| | Specifies a Hazelcast enterprise license. An enterprise license is <i>not</i> required, but supplying one will unlock additional Hazelcast functionality, such as the ability to use the Hazelcast Management Center with clusters containing more than 2 nodes, which may be useful in production environments. |
| hazelcast.managementcenter.url | |
| | Specifies the URL where the Hazelcast Management Center is running. When a URL is configured each node in the cluster will connect to the Management Center at that URL and broadcast its status. |
| hazelcast.group.name | |
| \${user.name} | Specifies the cluster group the instance should join. This can be used, for example, to partition development and production clusters. |
| hazelcast.group.password | |
| \${user.name} | The password required to join the specified cluster group. |

| | |
|--|--|
| <code>hazelcast.http.sessions</code> | |
| <code>local</code> | <p>Specifies how HTTP sessions should be managed.</p> <p>The following values are supported:</p> <ul style="list-style-type: none"> <code>local</code> <p>HTTP sessions are managed per node. When used in a cluster, the load balancer MUST have sticky sessions enabled. If a node fails or is shut down, users that were assigned to that node need to log in again.</p> <ul style="list-style-type: none"> <code>sticky</code> <p>HTTP sessions are distributed across the cluster with a load balancer configured to use sticky sessions. If a node fails or is shut down, users do not have to log in again. In this configuration, session management is optimized for sticky sessions and will not perform certain cleanup tasks for better performance.</p> <ul style="list-style-type: none"> <code>replicated</code> <p>HTTP sessions are distributed across the cluster. The load balancer does not need to be configured for sticky sessions.</p> <p><code>local</code> is the recommended setting for standalone installations. For clustered installations <code>local</code> is the most performant option, followed by <code>sticky</code> and <code>replicated</code>.</p> |
| <code>hazelcast.local.public.address</code> | |
| | In most environments it should not be necessary to configure this explicitly. However, when using NAT, such as when starting cluster nodes using Docker, it may be necessary to configure the public address explicitly to avoid request binding errors. |
| <code>hazelcast.network.aws</code> | |
| <code>false</code> | A boolean flag to indicate whether Hazelcast has AWS EC2 Auto Discovery enabled. When setting this property to <code>true</code> , either <code>hazelcast.network.aws.iam.role</code> or (<code>hazelcast.network.aws.access.key</code> and <code>hazelcast.network.aws.secret.key</code>) become required properties. |
| <code>hazelcast.network.aws.iam.role</code> | |
| | If <code>hazelcast.network.aws</code> is <code>true</code> , then you must either set this property to your AWS role or set <code>hazelcast.network.aws.access.key</code> and <code>hazelcast.network.aws.secret.key</code> in order to discover your cluster node instances via the AWS EC2 API. |
| <code>hazelcast.network.aws.access.key</code> | |
| | If <code>hazelcast.network.aws</code> is <code>true</code> and <code>hazelcast.network.aws.iam.role</code> is not set, then you must set this property to your AWS account access key, in order to discover your cluster node instances via the AWS EC2 API. |
| <code>hazelcast.network.aws.secret.key</code> | |
| | If <code>hazelcast.network.aws</code> is <code>true</code> and <code>hazelcast.network.aws.iam.role</code> is not set, then you must set this property to your AWS account secret key, in order to discover your cluster node instances via the AWS EC2 API. |
| <code>hazelcast.network.aws.region</code> | |
| | The AWS region to query. If empty, Hazelcast's default ("us-east-1") is used. If set, it will override any value for "hazelcast.network.aws.host.header" (see below). |
| <code>hazelcast.network.aws.host.header</code> | |

| | |
|---|--|
| | The Host: header to use when querying the AWS EC2 API. If empty, then Hazelcast's default (" ec2.amazonaws.com ") is used. If set, then "hazelcast.network.aws.region" shouldn't be set as it will override this property. |
| hazelcast.network.aws.port.range | |
| | Port to connect to on instances found via the AWS discovery mechanism. Can be a port range in the format 5701-5703 or single port. Providing a range with more than 3 ports may impact the application startup time. By default this is set to the value of the hazelcast.port property |
| hazelcast.network.aws.security.group.name | |
| | There are 2 mechanisms for filtering out AWS instances and these mechanisms can be combined (AND). <ul style="list-style-type: none"> • If hazelcast.network.aws.security.group.name is set, only instances within that security group will be selected. • If hazelcast.network.aws.tag.key and hazelcast.network.aws.tag.value are set, only instances with that tag key/value will be selected. |
| hazelcast.network.aws.tag.key | |
| | The AWS tag key to use to filter instances to form a cluster with. |
| hazelcast.network.aws.tag.value | |
| | The AWS tag value to use to filter instances to form a cluster with. |
| hazelcast.network.azure | |
| false | A boolean flag to indicate whether Hazelcast has Azure Auto Discovery enabled. When setting this property to true, hazelcast.network.azure.cluster.id , hazelcast.network.azure.group.name and hazelcast.network.azure.subscription.id become required properties. If Azure system-assigned managed identity is not enabled for the Bitbucket VMs then hazelcast.network.azure.client.id , hazelcast.network.azure.client.secret and hazelcast.network.azure.tenant.id are also required properties. |
| hazelcast.network.azure.cluster.id | |
| | If hazelcast.network.azure is true, then you must set this property to the name of the tag on the hazelcast vm or scale set resources, the value of the tag should be the port that hazelcast will use to communicate |
| hazelcast.network.azure.group.name | |
| | If hazelcast.network.azure is true, then you must set this property to the Azure resource group name of the cluster |
| hazelcast.network.azure.subscription.id | |
| | If hazelcast.network.azure is true, then you must set this property to your Azure subscription ID |
| hazelcast.network.azure.client.id | |
| | If hazelcast.network.azure is true and Azure system-assigned managed identity is not enabled for the Bitbucket VMs, then you must set this property to your Azure Active Directory Service Principal client ID |
| hazelcast.network.azure.client.secret | |
| | If hazelcast.network.azure is true and Azure system-assigned managed identity is not enabled for the Bitbucket VMs, then you must set this property to your Azure Active Directory Service Principal client secret |

| | |
|---|--|
| hazelcast.network.azure.tenant.id | |
| | If <code>hazelcast.network.azure</code> is <code>true</code> and Azure system-assigned managed identity is not enabled for the Bitbucket VMs, then you must set this property to your Azure Active Directory tenant ID |
| hazelcast.network.kubernetes | |
| <code>false</code> | A <code>boolean</code> flag to indicate whether Hazelcast has Kubernetes Auto Discovery enabled. No other properties are required for configuration of Kubernetes. |
| hazelcast.network.multicast | |
| <code>false</code> | A <code>boolean</code> flag to indicate whether Hazelcast has multicasting enabled. |
| hazelcast.network.multicast.address | |
| | The multicast address for the cluster, used to locate members when multicast discovery is enabled. If no value is set here, Hazelcast's default (224.2.2.3) is used. This value should not need to be configured explicitly on most networks. |
| hazelcast.network.multicast.port | |
| <code>\${hazelcast.port}</code> | The multicast port to bind to. By default, this will be the <code>hazelcast.port</code> (5701, unless configured otherwise), and updating that setting will also update this one unless this port is configured explicitly. |
| hazelcast.network.tcpip | |
| <code>false</code> | A <code>boolean</code> flag to indicate whether Hazelcast has TCP/IP enabled. |
| hazelcast.network.tcpip.members | |
| <code>localhost:5701,localhost:5702</code> | List of members that Hazelcast nodes should connect to when TCP/IP is enabled. These nodes function as root nodes, allowing cluster nodes to discover each other. This comma-separated list does <i>not</i> need to include every node in the cluster. When new nodes join they will use the connected node to find the other cluster nodes. |
| hazelcast.port | |
| <code>5701</code> | The network port where Hazelcast will listen for cluster members. If multiple instances are run on the same server Hazelcast will automatically increment this value for additional nodes. |

Code Insights

| Default value | Description |
|---|---|
| plugin.bitbucket-code-insights.reports.expiry.days | |
| <code>60</code> | Controls how long code insight cards are kept in the database.

This value is in days . |
| plugin.bitbucket-code-insights.pullrequest.changedlines.cache.max | |
| <code>500</code> | Controls the number of pull request diffs kept in the insights diff cache |
| plugin.bitbucket-code-insights.pullrequest.changedlines.cache.ttl | |
| <code>7200</code> | Controls the number of seconds for which the diff cache is kept.

This value is in seconds |

Comment Likes and Comment Reactions

| Default value | Description |
|--|--|
| plugin.bitbucket-comment-likes.max.resources | |
| 500 | The maximum number of comment likes associated with a single comment. The system will allow to create more likes than this number, but the results of API calls to get likes will be <i>strictly limited</i> to the value configured here, with appropriate warnings recorded in the application logs. |
| plugin.bitbucket-comment-likes.max.page | |
| 100 | The maximum size of a page of comment likes |
| plugin.bitbucket-comment-likes.max.reactions | |
| 50 | The maximum number of specific comment reactions for a single comment. The system will not allow you to add more of the same reaction. |

Commit Indexing

These properties control how commits are indexed when after pushes or pull request merges.

| Default value | Description |
|------------------------------------|---|
| indexing.max.threads | |
| 4 | Controls the maximum number of threads which are used to perform indexing. The resource limits configured below are not applied to these threads, so using a high number may negatively impact server performance. |
| indexing.import.max.threads | |
| 1 | Controls the maximum number of threads which are used to perform indexing during Data Center migration imports. The resource limits configured below are not applied to these threads, so using a high number may negatively impact server performance. |
| indexing.job.batch.size | |
| 250 | Defines the number of commits which will be indexed in a single database transaction. |
| indexing.job.queue.size | |
| 150 | Defines the maximum number of pending indexing requests. When this limit is reached, attempts to queue another indexing operation will be rejected. |
| indexing.queue.timeout.poll | |
| 60 | Controls how long indexing processes are allowed to wait for the next commit to be made available in the commit queue before assuming the process that retrieves the commits is stuck and giving up.

This value is in seconds . |
| indexing.queue.size | |
| 1024 | Defines the size of the queue that will be used for indexing. When the limit is reached the program will block until there is space in the queue to add any required new items. |
| indexing.process.timeout.execution | |

| | |
|--|---|
| 3600 | Controls how long indexing processes are allowed to execute before they are interrupted, even if they are producing output or consuming input.

This value is in seconds . |
| indexing.snapshot.timeout.execution | |
| 120 | Controls how long snapshot generation, which captures the state of a repository's branches and tags, is allowed to execute before it is interrupted. This timeout is applied whether the process is producing output or not.

This value is in seconds . |
| indexing.snapshot.timeout.idle | |
| <code>\${indexing.snapshot.timeout.execution}</code> | Controls how long snapshot generation, which captures the state of a repository's branches and tags, is allowed to run without producing output before it is interrupted.

This value is in seconds . |

Commit graph cache

| Default value | Description |
|-----------------------------------|--|
| commit.graph.cache.min.free.space | |
| 107374
1824 | Controls how much space needs to be available on disk (specifically under <code><BITBUCKET_HOME>/caches</code>) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk.

This value is in bytes . |
| commit.graph.cache.max.threads | |
| 2 | Defines the number of threads that will be used to create commit graph cache entries. |
| commit.graph.cache.max.job.queue | |
| 1000 | Defines the maximum number of pending cache creation jobs. |

Commits

| Default value | Description |
|-----------------------------|--|
| commit.diff.context | |
| 10 | Defines the number of context lines to include around diff segments in commit diffs. |
| commit.lastmodified.timeout | |
| 120 | Defines the timeout for streaming last modifications for files under a given path, applying a limit to how long the traversal can run before the process is canceled. This timeout is applied as both the execution and idle timeout.

This value is in seconds . |
| commit.list.follow.renames | |

| | |
|-------------------------|---|
| true | Defines whether file history commands in the UI should follow renames by default. This feature can be disabled as it may cause significant load for repositories with long commit logs. |
| commit.message.max | |
| 262144 | Controls the maximum length of the commit message to be loaded when retrieving one or more commits from the SCM. Commit messages longer than this limit will be truncated. The default limit is high enough to not affect processing for the general case, but protects the system from consuming too much memory in exceptional cases. |
| commit.message.bulk.max | |
| 16384 | Controls the maximum length of the commit message to be loaded when bulk retrieving a commits from the SCM. Commit messages longer than this limit will be truncated. The default limit is high enough to not affect processing for the common case, but protects the system from consuming too much memory when many commits have long messages. |

Content

| Default value | Description |
|-------------------------|---|
| content.archive.timeout | |
| 1800 | Defines the timeout for archive processes, applying a limit to how long it can take to stream a repository's archive before the process is canceled. This timeout is applied as both the execution and idle timeout.

This value is in seconds . |
| content.patch.timeout | |
| 1800 | Defines the timeout for patch processes, applying a limit to how long it can take to stream a diff's patch before the process is canceled. This timeout is applied as both the execution and idle timeout.

This value is in seconds . |

Data Center Migration

| Default value | Description |
|---------------------------|--|
| migration.threadpool.size | |
| 2 | Maximal size of thread pool the maximum number of concurrent migrations. |

Database

Database properties allow explicitly configuring the database the system should use. They may be configured directly in `bitbucket.properties`, or they may be specified during setup. Existing systems may be migrated to a new database using the in-app migration feature.

If no database is explicitly configured, an internal database will be used automatically. Which internal database is used is not guaranteed.

If the `jdbc.driver`, `jdbc.url`, `jdbc.password` and `jdbc.user` properties are specified in `bitbucket.properties` when the Setup Wizard runs after installation, then those values will be used, and the Setup Wizard will not display the database configuration screen.

Warning: `jdbc.driver` and `jdbc.url` are available to plugins via the `ApplicationPropertiesService`. Some JDBC drivers allow the username and password to be defined in the URL. Because that property is available throughout the system (and will be included in support requests), that approach should not be used. The `jdbc.username` and `jdbc.password` properties should be used for these values instead.

| Default value | Description |
|--|---|
| <code>jdbc.driver</code> | |
| <code>org.h2.Driver</code> | <p>The JDBC driver class that should be used to connect to the database.</p> <p>The system uses an internal database by default, and stores its data in the home directory.</p> <p>The following JDBC drivers are bundled with the distribution:</p> <ul style="list-style-type: none"> <code>com.microsoft.sqlserver.jdbc.SQLServerDriver</code> (more info) <code>oracle.jdbc.OracleDriver</code> (more info) <code>org.postgresql.Driver</code> (more info) <p><i>The JDBC driver for MySQL is not bundled, due to licensing restrictions; administrators will need to download and install that driver manually. See Connecting to MySQL for instructions.</i></p> |
| <code>jdbc.url</code> | |
| <code>jdbc:h2:\${bitbucket.shared.home}/data/db;MVCC=TRUE;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;TRACE_LEVEL_FILE=4</code> | <p>This is the JDBC URL that will be used to connect to the database. This URL varies depending on the database you are connecting to. Please consult the documentation for your JDBC driver to determine the correct URL.</p> |
| <code>jdbc.user</code> | |
| <code>sa</code> | <p>This is the user that will be used to authenticate with the database. <i>This user must have full DDL rights.</i> It must be able to create, alter and drop tables, indexes, constraints, and other SQL objects, as well as being able to create and destroy temporary tables.</p> |
| <code>jdbc.password</code> | |
| | <p>The password that the user defined by <code>jdbc.user</code> will connect with. This will be decrypted by the class mentioned in the <code>jdbc.password.decrypter.classname</code> property if defined.</p> |
| <code>jdbc.password.decrypter.classname</code> | |
| | <p>Fully qualified name of the class that's used for decrypting <code>jdbc.password</code>. This class must implement the <code>com.atlassian.db.config.password.Cipher</code> interface, and must be available on the classpath. Do not specify this parameter if <code>jdbc.password</code> isn't encrypted.</p> |

Database Pool

These properties control the database pool. The pool implementation used is HikariCP. Documentation for these settings can be found on the [HikariCP configuration section](#).

To get a feel for how these settings really work in practice, the most relevant classes in HikariCP are:

- `com.zaxxer.hikari.HikariConfig`
Holds the configuration for the database pool and has documentation for the available settings.
- `com.zaxxer.hikari.pool.HikariPool`
Provides the database pool and manages connections.
- `com.zaxxer.hikari.util.ConnectionBag`

Holds references to open connections, whether in-use or idle.

| Default value | Description |
|--|--|
| <code>db.pool.rejectioncooldown</code> | |
| 10 | When a connection cannot be leased because the pool is exhausted, the stack traces of all threads which are holding a connection will be logged. This defines the cooldown that is applied to that logging to prevent spamming stacks in the logs on every failed connection request.

This value is in minutes |
| <code>db.pool.size.idle</code> | |
| 5 | Defines the number of connections the pool tries to keep idle. <i>The system can have more idle connections than the value configured here.</i> As connections are borrowed from the pool, this value is used to control whether the pool will eagerly open new connections to try and keep some number idle, which can help smooth ramp-up for load spikes. |
| <code>db.pool.size.max</code> | |
| 80 | Defines the maximum number of connections the pool can have open at once. |
| <code>db.pool.timeout.connect</code> | |
| 15 | Defines the amount of time the system will wait when attempting to open a new connection before throwing an exception. The system may hang, during startup, for the configured number of seconds if the database is unavailable. As a result, the timeout configured here should <i>not</i> be generous.

This value is in seconds . |
| <code>db.pool.timeout.idle</code> | |
| 1750 | Defines the maximum period of time a connection may be idle before it is closed. In general, generous values should be used here to prevent creating and destroying many short-lived database connections (which defeats the purpose of pooling).

Note: If an aggressive timeout is configured on the database server, a <i>more aggressive</i> timeout must be used here to avoid issues caused by the database server closing connections from its end. The value applied here should ensure the system closes idle connections before the database server does. This value needs to be less than <code>db.pool.timeout.lifetime</code> otherwise the idle timeout will be ignored.

This value is in seconds . |
| <code>db.pool.timeout.leak</code> | |
| 0 | Defines the maximum period of time a connection may be checked out before it is reported as a potential leak. <i>By default, leak detection is not enabled.</i> Long-running tasks, such as taking a backup or migrating databases, can easily exceed this threshold and trigger a false positive detection.

This value is in minutes . |
| <code>db.pool.timeout.lifetime</code> | |

| | |
|----|---|
| 30 | <p>Defines the maximum <i>lifetime</i> for a connection. Connections which exceed this threshold are closed the first time they become idle and fresh connections are opened.</p> <p>This value is in minutes.</p> |
|----|---|

Database Schema

These properties control aspects of how the database schema is managed.

| Default value | Description |
|-----------------------------|--|
| db.schema.lock.maxWait | |
| 30 | <p>Defines the maximum amount of time the system can wait to acquire the schema lock. Shorter values will prevent long delays on server startup when the lock is held by another instance or, more likely, when the lock was not released properly because a previous start was interrupted while holding the lock. This can happen when the system is killed while it is attempting to update its schema.</p> <p>This value is in seconds.</p> |
| db.schema.lock.pollInterval | |
| 5 | <p>Defines the amount of time to wait between attempts to acquire the schema lock. Slower polling produces less load, but may delay acquiring the lock.</p> <p>This value is in seconds.</p> |

Diagnostics

| Default value | Description |
|--|--|
| diagnostics.alert.dispatcher.max.threads | |
| 5 | <p>The maximum number of alert dispatcher threads. The number of dispatcher threads will only be increased when the alert queue is full and this configured limit has not been reached.</p> |
| diagnostics.alert.dispatcher.queue.size | |
| 250 | <p>The number of events that can be queued. When the queue is full and no more threads can be created to handle the events, events will be discarded.</p> |
| diagnostics.issues.event.dropped.threaddump.cooldown.seconds | |
| 60 | <p>Configures how often thread dumps should be generated for alerts relating to dropped events. Taking thread dumps can be computationally expensive and may produce a large amount of data when run frequently.</p> <p>This value is in seconds</p> |
| diagnostics.issues.event.slow.listener.time.millis | |
| 15000 | <p>Configures when an alert is raised for a slow event listener. If an event listener takes longer than the configured time to process an event, a warning alert is raised and made visible on the System Health page.</p> <p>This value is in milliseconds</p> |
| diagnostics.issues.event.slow.listener.overrides | |

| | |
|--|---|
| | <p>Configures overrides for specific event listeners and/or specific plugins. This setting can be used to suppress 'slow event listener detected' alerts for specific event listeners or plugins. The value should be comma-separated list of configurations of individual triggers, where a trigger is either the plugin-key, or the plugin-key followed by the event listener class name.</p> <p>Overrides are <i>only</i> considered if they specify more tolerant limits than the value specified in the <code>diagnostics.issues.event.slow.listener.time.millis</code> value. Setting a shorter override (e.g. 1000 when the default is 15000) will have no effect.</p> <p>The following example sets the trigger for the <code>com.company.example-plugin</code> to 60s and sets the limit for the <code>com.company.RepositoryCreatedListener</code> event listener in the same plugin to 30s.</p> <pre>com.company.example-plugin:60000, com.company.example-plugin.com. company.RepositoryCreatedListener:30000</pre> <p>Configured values are in milliseconds</p> |
| <code>diagnostics.issues.hookscript.slow.time.seconds</code> | |
| 30 | <p>Defines the maximum amount of time an individual hook script is allowed to execute <i>or</i> idle before a warning would be logged in the diagnostics plugin.</p> <p>This value is in seconds. The default is 30 seconds, with a 10 second minimum</p> |
| <code>diagnostics.alert.retention.time.minutes</code> | |
| 43200 | <p>Configures the minimum amount of time alerts are kept in the database before being periodically truncated. The default (43200 minutes) is 30 days.</p> <p>This value is in minutes</p> |
| <code>diagnostics.alert.truncation.interval.minutes</code> | |
| 1440 | <p>Configures the interval at which alerts are truncated from the database; in case of a fresh instance (or full cluster) (re-)start, this is also the initial offset until the truncation is executed for the first time. The default (1440 minutes) is 24 hours.</p> <p>This value is in minutes</p> |

Display

These properties control maximum limits of what will be displayed in the web UI.

| Default value | Description |
|--|--|
| <code>display.max.source.lines</code> | |
| 20000 | <p>Controls how many lines of a source file will be retrieved before a warning banner is shown and the user is encouraged to download the raw file for further inspection. This property relates to <code>page.max.source.lines</code> in that up to $(display.max.source.lines/page.max.source.lines)$ requests will be made to view the page.</p> |
| <code>display.max.jupyter.notebook.size</code> | |
| 2097152 | <p>Controls the size of the largest jupyter notebook that will be automatically loaded and rendered in the source view. Users will be prompted to manually trigger loading the notebook for files larger than this.</p> <p>This value is in bytes.</p> |

Downloads

| Default value | Description |
|--------------------------|---|
| http.download.raw.policy | |
| Smart | <p>Controls the download policy for raw content.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> • Insecure
Allows all file types to be viewed in the browser. • Secure
Requires all file types to be downloaded rather than viewed in the browser. • Smart
Forces "dangerous" file types to be downloaded, rather than allowing them to be viewed in the browser. <p>These options are case-sensitive and defined in <code>com.atlassian.http.mime.DownloadPolicy</code></p> |

Elasticsearch

Bitbucket Server 4.6+ ships with an embedded instance of Elasticsearch. These properties enable admins to configure the base URL of the Elasticsearch instance, and enable basic security measures in the form of a username and password for accessing the Elasticsearch instance.

Warning: If an Elasticsearch parameter is set in the properties file, it cannot be edited later from the admin UI. Any changes that need to be made to the Elasticsearch configuration must be made within the bitbucket.properties file.

| Default value | Description |
|--|--|
| plugin.search.elasticsearch.baseurl | |
| | Sets the base URL of an Elasticsearch instance. |
| plugin.search.elasticsearch.username | |
| | Username for connecting to an Elasticsearch instance. |
| plugin.search.elasticsearch.password | |
| | Password for connecting to an Elasticsearch instance. |
| plugin.search.elasticsearch.aws.region | |
| | AWS region Bitbucket is running in. When set, enables request signing for Amazon Elasticsearch Service. |
| plugin.search.indexing.max.batch.size | |
| | <p>Maximum size of indexing batches sent to Elasticsearch. This value should be less than the maximum content length for http request payloads supported by Elasticsearch. The default value is 15728640 (=15MB). (Note that for Elasticsearch instances running on AWS, this value must be less than the Network Limit size of the Elasticsearch instance).</p> <p>This value is in bytes.</p> |

Events

These properties control the number of threads that are used for dispatching asynchronous events. Setting this number too high can decrease overall throughput when the system is under high load because of the additional overhead of context switching. Configuring too few threads for event dispatching can lead to events being queued up, thereby reducing throughput. These defaults scale the number of dispatcher threads with the number of available CPU cores.

| Default value | Description |
|--|--|
| event.dispatcher.core.threads | |
| 0.8 * <code>{scaling.concurrency}</code> | The minimum number of threads that is available to the event dispatcher. The <code>{scaling.concurrency}</code> variable is resolved to the number of CPUs that are available. |
| event.dispatcher.max.threads | |
| <code>{scaling.concurrency}</code> | The maximum number of event dispatcher threads. The number of dispatcher threads will only be increased when the event queue is full and this configured limit has not been reached. |
| event.dispatcher.queue.size | |
| 4096 | The number of events that can be queued. When the queue is full and no more threads can be created to handle the events, events will be discarded. |
| event.dispatcher.queue.rejectioncooldown | |
| 10 | When an event cannot be dispatched because the queue is full, the stack traces of all busy event processing threads will be logged. This defines the cooldown that is applied to that logging to prevent spamming the stacks in the logs on every rejected event.

This value is in minutes |
| event.dispatcher.keepAlive | |
| 60 | The time a dispatcher thread will be kept alive when the queue is empty and more than core. threads threads are running.

This value is in seconds . |

Executor

Controls the thread pool that is made available to plugins for asynchronous processing.

| Default value | Description |
|------------------------------------|---|
| executor.max.threads | |
| <code>{scaling.concurrency}</code> | Controls the maximum number of threads allowed in the common <code>ExecutorService</code> . This <code>ExecutorService</code> is used by for background tasks, and is also available for plugin developers to use. When more threads are required than the configured maximum, the thread attempting to schedule an asynchronous task to be executed will block until a thread in the pool becomes available. By default, the pool size scales with the number of reported CPU cores. Note: A minimum of 4 is enforced for this property. Setting the value to a lower value will result in the default 4 threads being used. |

Features

These properties control high-level system features, allowing them to be disabled for the entire instance. Features that are disabled at this level are disabled *completely*. This means instance-level configuration for a feature is overridden. It also means a user's permissions are irrelevant; a feature is still disabled even if the user has the system admin permission.

| Default value | Description |
|------------------------------|---|
| feature.attachments | |
| true | Controls whether users are allowed to upload attachments to repositories they have access to. If this feature is enabled and later disabled, attachments which have already been uploaded <i>are not automatically removed</i> . |
| feature.auth.captcha | |
| true | <p>Controls whether to require CAPTCHA verification when the number of failed logins is exceeded. If enabled, any client who has exceeded a set limit for failed logins using either the web interface or Git hosting will be required to authenticate in the web interface and successfully submit a CAPTCHA before continuing. Disabling this will remove this restriction and allow users to incorrectly authenticate as many times as they like without penalty.</p> <p>Warning: It is strongly recommended to keep this setting enabled. Disabling it has the following ramifications:</p> <ul style="list-style-type: none"> • Users may lock themselves out of any underlying user directory service (LDAP, Active Directory etc) because the system will pass through all authentication requests (regardless of the number of previous failures) to the underlying directory service. • For installations where Bitbucket Server is used for user management or a directory service with no limit on failed login attempts is used, the system will be vulnerable to brute-force password attacks. |
| feature.commit.graph | |
| true | Controls whether a commit graph is displayed to the left of the commits on the repository commits page. |
| feature.diagnostics | |
| true | Controls whether diagnostics is enabled. Diagnostics looks for known anti-patterns such as long-running operations being executed on event processing threads and identifies the responsible plugins. Diagnostics adds a little bit of overhead, and can be disabled if necessary. |
| feature.file.editor | |
| true | <p>Controls whether users can edit repository files in the browser and via REST.</p> <p>When set to <code>false</code>, the edit UI and REST interface are disabled globally.</p> |
| feature.forks | |
| true | Controls whether repositories can be forked. This setting <i>supersedes and overrides</i> instance-level configuration. If this is set to <code>false</code> , even repositories which are marked as forkable cannot be forked. |
| feature.getting.started.page | |
| true | Controls whether new users are redirected to a getting started page after their first login. |
| feature.git.rebase.workflows | |

| | |
|--------------------------------------|--|
| true | <p>Controls whether rebase workflows are enabled for Git repositories. This can be used to fully disable all of the Git SCM's built-in rebase support, including:</p> <ul style="list-style-type: none"> • "Rebase and fast-forward" and "Rebase and merge" merge strategies - "Rebase" action for pull requests - "Rebase" action for ref sync <p>When this feature is disabled, repository administrators and individual users cannot override it. However, third-party add-ons can still use the Java API to rebase branches or perform rebase "merges".</p> |
| feature.data.center.migration.export | |
| true | Controls whether Data Center migration archives can be generated on this instance. |
| feature.data.center.migration.import | |
| true | Controls whether Data Center migration archives can be imported into the instance. Note that this feature is only available for Data Center installations |
| feature.jira.cloud.devinfo | |
| true | Controls whether the system can send development information to Jira Cloud. Note that this feature is only available for Data Center installations |
| feature.personal.repos | |
| true | <p>Controls whether personal repositories can be created.</p> <p>When set to <code>false</code>, personal repository creation is disabled globally.</p> |
| feature.public.access | |
| true | Public access allows anonymous users to be granted access to projects and repositories for read operations including cloning and browsing repositories. This is controlled normally by project and repository administrators but can be switched off system wide by setting this property to <code>false</code> . This can be useful in sensitive environments. |
| feature.pull.request.auto.decline | |
| true | <p>Controls whether the process of automatically declining inactive pull requests is enabled for the system.</p> <p>When this feature is enabled, all pull requests that are inactive (no recent comments, pushes etc.) are declined. Individual projects or repositories are still able to opt-out or configure a different inactivity period. When this feature is disabled, it overrides any configuration applied to individual projects or repositories, and prevents any pull request from being automatically declining due to inactivity.</p> |
| feature.pull.request.deletion | |
| true | Controls whether the system allows pull requests to be deleted via the UI and REST. Disabling this feature will prevent pull request deletion in <i>all</i> repositories, including by admins and sysadmins, and will override any settings applied to individual repositories. |
| feature.pull.request.suggestions | |
| true | Controls whether the system allows users to add pull request suggestions through inline comments via the UI. |
| feature.pull.request.templates | |
| true | Controls whether the system allows users to create and manage pull request templates via REST or UI |
| feature.rate.limiting | |

| | |
|-----------------------------------|--|
| true | Controls whether HTTP requests will be rate limited per user. If this is enabled, repeated HTTP requests from the same user in a short time period may be rate limited. If this is disabled, no requests will be rate limited. |
| feature.repository.delete.policy | |
| true | Controls whether a user can delete a repository by checking their permission level against the repository delete policy. Note that this feature is only available for Data Center installations |
| feature.repository.management | |
| true | Controls whether admins have access to the Repositories view in the UI or the repository-management endpoint in REST. Note that this feature is only available for Data Center installations. |
| feature.required.builds | |
| true | Controls whether admins have access to the Required Builds view in the UI or the required-builds endpoint in REST. Note that this feature is only available for Data Center installations. |
| feature.reviewer.groups | |
| true | Controls whether a user can manage reviewer groups. Note that this feature is only available for Data Center installations |
| feature.rolling.upgrade | |
| true | Controls whether rolling upgrade can be performed for bug-fix versions. Note that this feature is only available for Data Center installations. |
| feature.smart.mirrors | |
| true | Controls whether smart mirrors can be connected to the instance. Note that this feature is only available for Data Center installations |
| feature.user.time.zone.onboarding | |
| true | Controls whether users with mismatching time zones are shown an alert prompting them to change their user time zone. |

File editor

| Default value | Description |
|-------------------------|---|
| content.upload.max.size | |
| 5242880 | Controls the maximum allowed file size when editing a file through the browser or file edit REST endpoint

This value is in bytes . Default is 5 MiB |

Footer

| Default value | Description |
|------------------------------|-------------|
| footer.links.contact.support | |

| | |
|--|--|
| | <p>Controls whether the Contact Support link is displayed in the footer. If this is not set, then the link is not displayed. Otherwise, the link will redirect to the URL or email that is provided.</p> <p>Example formats:</p> <ul style="list-style-type: none"> • https://support.example.com/ • mailto: support@example.com |
|--|--|

Fork Syncing (Ref Syncing)

| Default value | Description |
|--|---|
| <code>plugin.bitbucket-repository-ref-sync.fetch.timeout</code> | |
| 300 | <p>Defines the maximum amount of time fetch commands used to synchronize branches in bulk are allowed to execute <i>or</i> idle. Because fetch commands generally produce the majority of their output upon completion, there is no separate idle timeout. The default value is 5 minutes.</p> <p>This value is in seconds.</p> |
| <code>plugin.bitbucket-repository-ref-sync.merge.timeout</code> | |
| 300 | <p>Defines the maximum amount of time any command used to merge upstream changes into the equivalent branch in a fork is allowed to execute <i>or</i> idle. Since merging branches may require a series of different commands at the SCM level, this timeout does <i>not</i> define the upper bound for how long the overall merge process might take; it only defines the duration allotted to any single command.</p> <p>This value is in seconds.</p> |
| <code>plugin.bitbucket-repository-ref-sync.rebase.timeout</code> | |
| 300 | <p>Defines the maximum amount of time any command used to rebase a fork branch against upstream changes is allowed to execute <i>or</i> idle. Since merging branches may require a series of different commands at the SCM level, this timeout does <i>not</i> define the upper bound for how long the overall rebase process might take; it only defines the duration allotted to any single command.</p> <p>This value is in seconds.</p> |
| <code>plugin.bitbucket-repository-ref-sync.threads</code> | |
| 3 | <p>Controls the number of threads used for ref synchronization. Higher values here will help synchronization keep up with upstream updates, but may produce noticeable additional server load.</p> |

Graceful shutdown

| Default value | Description |
|------------------------------|-------------|
| <code>server.shutdown</code> | |

| | |
|--|--|
| graceful | <p>Controls whether Tomcat, SSH server and job scheduler should be shutdown gracefully or immediately. When graceful shutdown is enabled, this is how different components would behave after shutdown is initiated - * Tomcat stops accepting new HTTP connections but active connections are not interrupted. * Ssh server stops accepting new SSH connections but active connections are not interrupted. * Job scheduler does not start any new jobs but currently running jobs are not cancelled.</p> <p>System will allow above components to shutdown gracefully for some time which can be controlled by property <code>graceful.shutdown.timeout</code>. If value of this property is set as <code>immediate</code>, then HTTP/SSH requests and scheduled jobs will not be waited for completion and will be terminated abruptly in their usual shutdown lifecycle.</p> |
| <code>graceful.shutdown.timeout</code> | |
| 30 | <p>This is the minimum time for which system is guaranteed to wait for currently active HTTP/SSH requests and scheduled jobs to finish before terminating them. If you are using <code>stop-bitbucket.sh</code> to stop Bitbucket, the value of this property is effective only if it is less than or equal to the value of <code>BITBUCKET_SHUTDOWN_TIMEOUT</code> environment variable. It is recommended to set the value of this property at least 10 seconds less than <code>BITBUCKET_SHUTDOWN_TIMEOUT</code> environment variable.</p> <p>This value is in seconds</p> |

Hibernate

| Default value | Description |
|--|--|
| <code>hibernate.format_sql</code> | |
| <code>\${hibernate.show_sql}</code> | When <code>hibernate.show_sql</code> is enabled, this flag controls whether Hibernate will format the output SQL to make it easier to read. |
| <code>hibernate.jdbc.batch_size</code> | |
| 20 | Controls Hibernate's JDBC batching limit, which is used to make bulk processing more efficient (both for processing and for memory usage). |
| <code>hibernate.show_sql</code> | |
| <code>false</code> | Used to enable Hibernate SQL logging, which may be useful in debugging database issues. This value should generally only be set by developers, not by customers. |

Hook Scripts

| Default value | Description |
|-------------------------------------|--|
| <code>hookscripts.output.max</code> | |
| 32768 | <p>Applies a limit to how many bytes a single hook script can write to <code>stderr</code> or <code>stdout</code>. The limit is enforced <i>separately</i> for each, so a limit of 32768 allows for a maximum of 65536 bytes of combined output from a single script. Output beyond the configured limit is truncated, and a message is included to indicate so.</p> <p>This value is in bytes. The default is 32K, with a 16K minimum.</p> |
| <code>hookscripts.path.shell</code> | |
| | Defines the location of <code>bash.exe</code> , which is used when invoking hook scripts on Windows. This property is ignored on other platforms; only Windows requires using a shell to invoke hook scripts. |

| | |
|----------------------|---|
| hookscripts.size.max | |
| 10485760 | The size a script can be when it is uploaded.

This value is in bytes . |
| hookscripts.timeout | |
| 120 | Defines the maximum amount of time an individual hook script is allowed to execute <i>or</i> idle. Since multiple hook scripts can be registered, this timeout does not define an upper bound for how long overall execution can take; it only defines the duration allotted to any single hook script.

This value is in seconds . The default is 120 seconds, with a 30 second minimum |

Importer

| Default value | Description |
|--|--|
| plugin.importer.external.source.request.socket.timeout | |
| 30 | Controls how long requests to external repository source servers can continue without producing any data before the importer gives up.

This value is in seconds . |
| plugin.importer.external.source.request.timeout | |
| 30 | Controls how long requests to external repository source servers can proceed before the importer gives up.

This value is in seconds . |
| plugin.importer.import.repository.thread.max | |
| 8 | Maximal size of thread pool the maximum number of concurrent repository imports. |
| plugin.importer.repository.fetch.timeout.execution | |
| 360 | Defines the execution timeout for fetch processes, applying a hard limit to how long the operation is allowed to run even if it is producing output or reading input. The default value is 6 hours.

This value is in minutes . |
| plugin.importer.repository.fetch.timeout.idle | |
| 60 | Defines the idle timeout for fetch processes, applying a limit to how long the operation is allowed to execute without either producing output or consuming input. The default value is 60 minutes.

This value is in minutes . |

JMX

See [Enabling JMX counters for performance monitoring](#).

| Default value | Description |
|---------------|-------------|
| jmx.enabled | |

| | |
|-------|--|
| false | Controls whether JMX management interfaces for the system and its libraries are registered.

Note: Some libraries may register their JMX management interfaces regardless of this setting. |
|-------|--|

Jira

| Default value | Description |
|--|--|
| <code>plugin.jira-integration.comment.issues.max</code> | |
| 500 | Controls the maximum result size allowed when retrieving Jira issues linked to comments. |
| <code>plugin.jira-integration.pullrequest.attribute.commits.max</code> | |
| 100 | Controls the maximum number of commits to retrieve when retrieving attributes associated with commits of a pull-request. This value must be between 50 and 1000, which are imposed as lower and upper bounds on any value specified here. |
| <code>plugin.jira-integration.remote.page.max.issues</code> | |
| 20 | Controls the maximum number of issues to request from Jira. This value must be between 5 and 50, which are imposed as lower and upper bounds on any value specified here. |
| <code>plugin.jira-integration.remote.timeout.connection</code> | |
| 5000 | The connection timeout duration in milliseconds for requests to Jira. This timeout occurs if the Jira server does not answer. e.g. the server has been shut down. This value must be between 2000 and 60000, which are imposed as lower and upper bounds on any value specified here.

This value is in milliseconds . |
| <code>plugin.jira-integration.remote.timeout.socket</code> | |
| 10000 | The socket timeout duration in milliseconds for requests to Jira. This timeout occurs if the connection to Jira has been stalled or broken. This value must be between 2000 and 60000, which are imposed as lower and upper bounds on any value specified here.

This value is in milliseconds . |
| <code>plugin.jira-development-integration.reindex.pullrequests.commit.message.max</code> | |
| <code>\${commit.message.bulk.max}</code> | Controls the maximum length of the commit message to be loaded when reindexing pull requests. Any commit messages longer than this limit will be truncated.

Setting this value less than or equal to 0 will prevent indexing commits when reindexing pull requests, which may be useful for sysadmins if including commits results in excessive load on the server. |
| <code>plugin.jira-development-integration.reindex.pullrequests.command.timeout</code> | |
| 300 | Defines the timeout for streaming commits when reindexing pull requests. Pull requests are reindexed in batches, with a separate process for each batch. If a batch's process times out, subsequent batches will be skipped.

This value is in seconds , and is applied as both the execution and idle timeout. |

Jira Automatic Transition Trigger Events

These properties control whether events should be converted into Remote Events so they can trigger automatic issue transitions in Jira.

| Default value | Description |
|--|---|
| <code>plugin.dev-summary.pr.commits.threshold</code> | |
| <code>\${plugin.jira-integration.pullrequest.attribute.commits.max}</code> | Limit the number of commits to be scanned per pull request |
| <code>plugin.dev-summary.pr.events.enabled</code> | |
| <code>true</code> | Controls whether pull request events should be published |
| <code>plugin.dev-summary.branch.events.threshold</code> | |
| <code>10</code> | Limit the number of branch events sent per synchronization If set to zero then no branch events will be published |
| <code>plugin.dev-summary.commit.events.threshold</code> | |
| <code>100</code> | Limit the number of commit events sent per synchronization If set to zero then no commit events will be published |
| <code>plugin.dev-summary.issuechanged.events.threshold</code> | |
| <code>100000</code> | Limit the number of issue changed events sent per synchronization. Issue changed events are based on the issue keys within the commit messages for commit events or within branch names for branch events. For branches, this value is the maximum of branches considered. For commits, this value is the maximum number of commits considered. If set to zero then no issue changed events will be published |
| <code>plugin.dev-summary.issue.commits.threshold</code> | |
| <code>100</code> | Limit the number of commits to be returned per issue |
| <code>plugin.dev-summary.repository.trigger.settings.enabled</code> | |
| <code>false</code> | Controls whether the Jira triggers page is visible on repository settings |

Jira cloud development information events

These properties control how many branches, commits and pull requests are sent to Jira Cloud.

| Default value | Description |
|--|--|
| <code>plugin.jira-development-integration.cloud.devinformit.files.max</code> | |
| <code>10</code> | Limit the number of files on each Commit object. The API documentation specifies this should not be set above 10 |
| <code>plugin.jira-development-integration.cloud.devinformentity.issuekeys.max</code> | |
| <code>100</code> | Limit the number of issue keys per entity (commit, branch, pull request) The API documentation specifies this should not be set above 100 |
| <code>plugin.jira-development-integration.cloud.devinformevent.branch.max</code> | |
| <code>100</code> | Limit the number of branches we process per event This ensures Bitbucket doesn't process too many branches at once and should only be triggered on the initial push of a large, existing repository. |

| | |
|--|--|
| <code>plugin.jira-development-integration.cloud.devinfo.event.commit.max</code> | |
| 200 | Limit the number of commits we process per event This ensures Bitbucket doesn't process too many commits at once and should only be triggered on the initial push of a large, existing repository. |
| <code>plugin.jira-development-integration.cloud.devinfo.repository.entity.max</code> | |
| 100 | Limit the number of entities (branches, commits, pull requests) we put on each repository object. The API documentation specifies this should not be set above 400 |

Job Scheduler

Controls the scheduler that processes background jobs submitted by the system and by plugins.

| Default value | Description |
|---|---|
| <code>scheduler.history.expiry.days</code> | |
| 30 | <p>Controls how long job history is remembered after it is last updated before it expires. Job history in the "RunDetails" class available to plugins from atlassian-scheduler is only valid for the last run of the job on the same cluster node. Calls to retrieve RunDetails for the same job on different cluster nodes may return different results.</p> <p>This value is in days. The value cannot be negative.</p> |
| <code>scheduler.refresh.interval.minutes</code> | |
| 1 | <p>Controls the frequency at which the scheduler will automatically poll for changes to clustered jobs. If the value is positive, then the scheduler will refresh its queue of clustered jobs at the specified interval in minutes. If the value is 0 or negative, then jobs submitted on one cluster node will never be scheduled on other cluster nodes. It is not recommended to modify this setting unless recommended by Atlassian Support.</p> <p>This value is in minutes. Using 0, or a negative value, disables clustering of background jobs completely.</p> |
| <code>scheduler.shutdown.timeout</code> | |
| 15 | <p>The number of seconds to wait for running jobs to end after calling 'shutdown' on the scheduler service.</p> <p>This property is deprecated and, as of 7.8, it no longer does anything. It will be removed in 8.0.</p> <p>This value is in seconds.</p> |
| <code>scheduler.worker.threads</code> | |
| 4 | Controls the number of worker threads that will accept jobs from the queue on each cluster node. If the value is 0 or negative, then the scheduler's default of 4 threads will be used. |

Liquibase

| Default value | Description |
|--|-------------|
| <code>liquibase.commit.block.size</code> | |

| | |
|-------|--|
| 10000 | The maximum number of changes executed against a particular Liquibase database before a commit operation is performed. Very large values may cause DBMS to use excessive amounts of memory when operating within transaction boundaries. If the value of this property is less than one, then changes will not be committed until the end of the change set. |
|-------|--|

Logging

Logging levels for any number of loggers can be set in `bitbucket.properties` using the following format:

```
logging.logger.{name}={level}
```

To configure all classes in the `com.atlassian.bitbucket` package to DEBUG level:

```
logging.logger.com.atlassian.bitbucket=DEBUG
```

To adjust the ROOT logger, you use the special name ROOT (case-sensitive):

```
logging.logger.ROOT=INFO
```

Migration

Draining database connections during database migration happens in two stages. Stage 1 passively waits a set amount of time for all connections to be returned to the pool. If connections are still leased when `migration.drain.db.timeout` seconds has elapsed then stage 2 begins and will interrupt the owning threads, wait `migration.drain.db.force.timeout` seconds and finally attempt to roll back and close any remaining connections.

| Default value | Description |
|--|--|
| <code>migration.drain.db.timeout</code> | |
| <code>\${backup.drain.db.timeout}</code> | In stage 1 of draining connections during migration, this setting controls how long the migration should wait for outstanding database operations to complete before moving to stage 2. See <code>migration.drain.db.force.timeout</code> .

This value is in seconds . |
| <code>migration.drain.db.force.timeout</code> | |
| <code>\${backup.drain.db.force.timeout}</code> | In stage 2 of draining connections during migration, this property controls how long the migration process should wait (after interrupting the owning threads) for those threads to release the connections before forcibly rolling back and closing them. Note if all connections have been returned to the pool stage 2 is skipped and so this property has no effect. A negative value will skip stage 2 completely. See <code>migration.drain.db.timeout</code> .

This value is in seconds . |

Mirroring

| Default value | Description |
|---|--|
| <code>plugin.mirroring.farm.operation.callback.timeout</code> | |
| 300 | Defines how long a distributed operation in a mirror farm will wait for responses from other farm members before timing out.

This values is in seconds . |

| | |
|---|--|
| plugin.mirroring.farm.operation.max.inflight | |
| 1000 | Defines the maximum number of in-flight operations to keep track of per operation type before evicting the eldest. |
| plugin.mirroring.farm.operation.refchange.chunk.timeout | |
| 14400 | Defined how long to wait before timing out an operation to distribute and fetch refs This value is in seconds |
| plugin.mirroring.farm.operation.update.refs.timeout | |
| 1800 | Defined how long to wait before timing out a update ref command This values is in seconds |
| plugin.mirroring.farm.max.lock.acquisition.attempts | |
| 1800 | Defines the number of times an attempt is made to acquire a lock on a repository during initial synchronization before giving up. |
| plugin.mirroring.farm.max.chunk.size | |
| 5000 | Defines the maximum number of ref-changes before they will be split into a new chunk. |
| plugin.mirroring.farm.operation.threads | |
| 5 | Defines the maximum thread pool size for executing distributed operations. |
| plugin.mirroring.farm.operation.initial.retry.delay | |
| 1 | Defines how long the system should wait before retrying an operation that has failed the first time.

This values is in seconds . |
| plugin.mirroring.farm.operation.retry.attempts | |
| 5 | Defines the maximum number of times an operation is attempted before giving up and the failure is propagated. |
| plugin.mirroring.farm.vet.threads | |
| 5 | Defines the maximum thread pool size for executing operations to fix repository inconsistencies. |
| plugin.mirroring.http.write.enabled | |
| true | For <i>mirror instances</i> , this controls whether SCM write operations should be allowed for HTTP requests.

If enabled, write requests are redirected to the upstream server.

If disabled, an error message will be displayed when attempting a push, etc. |
| plugin.mirroring.ifs.download.upstream | |
| false | For <i>mirror instances</i> , this controls whether Large File Support (LFS) downloads are always requested from the upstream.

If enabled, LFS downloads are always requested from the upstream

If disabled, LFS downloads are served from the mirror |
| plugin.mirroring.ssh.upstream.proxying.enabled | |

| | |
|---|---|
| true | <p>For <i>upstream/primary instances</i>, this controls whether SCM commands proxied by a mirror over SSH should be allowed.</p> <p>If enabled, such proxied commands are performed on the upstream with the same user identity, rights and privileges as the user executing the command on the mirror.</p> <p>If disabled, an error message will be sent when a user attempts to execute a command that must be proxied on the upstream (such as a push).</p> |
| plugin.mirroring.ssh.proxy.enabled | |
| true | <p>For <i>mirror instances</i>, this controls whether compatible SCM commands are proxied to the upstream on behalf of the user.</p> <p>If enabled, such commands are performed on the upstream by the mirror acting with the same user identity, rights and privileges as the user executing the command on the mirror.</p> <p>If disabled, an error message will be sent to the user unless the command can be safely executed on the mirror (e.g. <code>whoami</code>).</p> <p>Note that SSH push proxying is not supported for mirrors of bitbucket.org and this flag is ignored.</p> |
| plugin.mirroring.ssh.proxy.parseconfig | |
| false | <p>This controls whether the SSH client created to proxy SCM commands is configured with the runtime user's SSH config file (usually found at <code>~/.ssh/config</code>)</p> <p>This is disabled by default but may be enabled if features like host aliasing are required.</p> |
| plugin.mirroring.ssh.proxy.upstream.timeout.execution | |
| 86400 | <p>Defines a hard limit for the amount of time that a proxied push to an upstream over SSH can run for even if it is producing input/output</p> <p>This values is in seconds.</p> |
| plugin.mirroring.ssh.proxy.upstream.timeout.idle | |
| 1800 | <p>Defines an idle timeout when proxying pushes over SSH to an upstream server. If no data is sent or received in this amount of time the connection will be terminated.</p> <p>This values is in seconds.</p> |
| plugin.mirroring.strict.hosting.status | |
| false | <p>Controls whether the mirror http status endpoint will return a 200 status code only when the mirror is synchronized. <i>NOTE</i> Use with caution as at least one mirror node must always be accessible from the upstream server. This is intended for deployments using advanced load balancer configurations.</p> |
| plugin.mirroring.upstream.url | |
| | <p>Defines where the mirror server should be mirroring from, this is also referred to as the base URL of the upstream server. Only define this property for mirror servers.</p> |
| plugin.mirroring.upstream.namespace | |
| | <p>Defines the namespace of the upstream server at initial registration, to be used in the clone URL for repositories mirrored from this server. Only used for mirror servers.</p> |
| plugin.mirroring.capabilities.refresh.interval | |
| 10 | <p>Controls how frequently the mirror will attempt to refresh the capabilities from the upstream server.</p> <p>This value is in minutes</p> |

| | |
|--|---|
| <code>plugin.mirroring.local.command.timeout.idle</code> | |
| 180 | <p>Defines the idle timeout for local commands, applying a limit to how long the operation is allowed to execute without either producing output or consuming input. The default value is 3 minutes.</p> <p>This value is in seconds.</p> |
| <code>plugin.mirroring.remote.command.timeout.idle</code> | |
| 3660 | <p>Defines the idle timeout for remote commands, applying a limit to how long the operation is allowed to execute without either producing output or consuming input. The default value is 61 minutes. This value should be greater than the <code>throttle.resource.mirror-hosting.timeout</code> value set on the upstream.</p> <p>This value is in seconds.</p> |
| <code>plugin.mirroring.state.refresh.interval</code> | |
| 1 | <p>Controls how frequently the mirror will attempt to refresh its state from the upstream server.</p> <p>This value is in minutes</p> |
| <code>plugin.mirroring.synchronization.delay.initial</code> | |
| 15 | <p>Controls how long after startup the first full synchronization should be attempted.</p> <p>This value is in seconds</p> |
| <code>plugin.mirroring.synchronization.fetch.timeout.execution</code> | |
| 172800 | <p>Defines the execution timeout for fetch processes, applying a hard limit to how long the operation is allowed to run even if it is producing output or reading input. The default value is 48 hours.</p> <p>This value is in seconds.</p> |
| <code>plugin.mirroring.synchronization.interval</code> | |
| 3 | <p>Controls how frequently a full synchronization with the upstream server should run</p> <p>This value is in minutes</p> |
| <code>plugin.mirroring.synchronization.ls-remote.timeout</code> | |
| 15 | <p>Controls how long to wait for an ls-remote command to the upstream server before it times out.</p> <p>This value is in minutes</p> |
| <code>plugin.mirroring.synchronization.max.failures</code> | |
| 3 | <p>Controls the number of consecutive failed synchronizations on a repository before the mirror gives up.</p> |
| <code>plugin.mirroring.synchronization.repository.page.size.max</code> | |
| 100 | <p>Controls how large the pages should be when requesting upstream repositories</p> |
| <code>plugin.mirroring.upstream.auth.cache.ttl</code> | |

| | |
|--|--|
| 300 | <p>Controls how long to cache the result of authentication requests made against the primary server such that if the same credentials are provided to the mirror within that period, the same result is used to calculate the outcome of the authentication and a remote (network) calls for the authentication request is avoided.</p> <p>Note: this caches both authentication successes (valid username/password, SSH key registered to user) but also failures (incorrect username/password pairs, unknown SSH keys).</p> <p>Also note: that cache entries expire after they are inserted, not after the last access. Repeatedly making authentication requests to the mirror with the same credentials will not prevent expiry of the result from the cache.</p> <p>This value is in seconds.</p> <p>A positive value will cache an authentication result for the configured period. A non-positive value will disable the cache and cause all authenticated requests to be executed remotely every time. Defaults to 5 minutes.</p> |
| <code>plugin.mirroring.upstream.auth.cache.max</code> | |
| -1 | <p>Configures the maximum number of authentication cache entries. Cache entries for Bitbucket Server mirrors can be heavy for certain permission schemes so this setting can control how much memory is devoted to this cache.</p> <p>For Bitbucket Server mirrors, there is one cache entry consumed for each combination of {user credentials, authentication method (HTTP basic/SSH)}</p> <p>For bitbucket.org mirrors, there is one cache entry consumed for each combination of {user credentials, authentication method (HTTP basic/SSH), repository}</p> <p>A negative value indicates an unlimited cache. A zero value indicates the cache should be disabled. A positive value indicates a specific cache limit. Defaults to unlimited.</p> |
| <code>plugin.mirroring.upstream.auth.cache.fallback.ttl</code> | |
| 1800 | <p>Controls the expiry of values in a secondary cache (separate to <code>plugin.mirroring.upstream.auth.cache.expiry</code>) which caches the result of authentication requests to be used to recover from remote authentication request fails for environmental or connectivity reasons. The types of problems this cache will try to overcome are network partition, request timeout, socket timeout, thread interruption, invalid HTTP response codes or entities from the primary.</p> <p>Note: this caches both authentication successes (valid username/password, SSH key registered to user) but also failures (incorrect username/password pairs, unknown SSH keys).</p> <p>Also note: that cache entries expire after they are inserted, not after the last access. Repeatedly making authentication requests to the mirror with the same credentials will not pause expiry of the result from the cache.</p> <p>This value is in seconds.</p> <p>A positive value will cache the authentication results for the configured period. If the value is positive and smaller than <code>plugin.mirroring.upstream.auth.cache.expiry</code> it will be adjusted upwards. A non-positive value will disable the cache and cause any failing authentication requests to not attempt recovery using a previously changed result thus failing any related authentication requests by clients. Defaults to half an hour.</p> |
| <code>plugin.mirroring.upstream.auth.cache.fallback.max</code> | |

| | |
|--|---|
| -1 | <p>Configures the maximum number of fallback authentication cache entries. Cache entries for Bitbucket Server mirrors can be heavy for certain permission schemes so this setting can control how much memory is devoted to this cache.</p> <p>For Bitbucket Server mirrors, there is one cache entry consumed for each combination of {user credentials, authentication method (HTTP basic/SSH)}</p> <p>For bitbucket.org mirrors, there is one cache entry consumed for each combination of {user credentials, authentication method (HTTP basic/SSH), repository}</p> <p>A negative value indicates an unlimited cache. A zero value indicates the cache should be disabled. A positive value indicates a specific cache limit. Defaults to unlimited.</p> |
| plugin.mirroring.upstream.event.ref.change.max.count | |
| 25 | <p>Defines the maximum number of ref-changes that will be published to the upstream server as part of the RepositorySynchronizedEvent. This prevents event objects from occupying unbounded amounts of memory while they are queued to be published or processed.</p> |
| plugin.mirroring.upstream.request.socket.timeout | |
| 15 | <p>Controls how long active requests to upstream servers can continue without producing any data before the mirror gives up. Must be a value smaller than plugin.ssh.auth.timeout by a decent margin if auth fallback caching is to be effective for socket timeouts</p> <p>This value is in seconds.</p> |
| plugin.mirroring.upstream.request.timeout | |
| 15 | <p>Controls how long requests to upstream servers can proceed before the mirror gives up. Must be a value smaller than plugin.ssh.auth.timeout by a decent margin if auth fallback caching is to be effective for request timeouts</p> <p>This value is in seconds.</p> |

Notifications

| Default value | Description |
|---|--|
| plugin.bitbucket-notification.mail.max.comment.size | |
| 2048 | Controls the maximum allowed size of a single comment in characters (not bytes). Extra characters will be truncated. |
| plugin.bitbucket-notification.mail.max.description.size | |
| 2048 | Controls the maximum allowed size of a single description in characters (not bytes). Extra characters will be truncated. |
| plugin.bitbucket-notification.mentions.enabled | |
| true | Controls whether mentions are enabled |
| plugin.bitbucket-notification.max.mentions | |
| 200 | Controls the maximum number of allowed mentions in a single comment |
| plugin.bitbucket-notification.sendmode.default | |
| BATCHED | <p>Default mode for sending notifications for users who have not set an explicit preference.</p> <p>This value is either BATCHED or IMMEDIATE.</p> |

| | |
|---|--|
| <code>plugin.bitbucket-notification.batch.min.wait.minutes</code> | |
| 10 | The minimum time to wait for new notifications in a batch before sending it (inactivity timeout).
This value is in minutes . |
| <code>plugin.bitbucket-notification.batch.max.wait.minutes</code> | |
| 30 | The maximum time to wait since the first notification of a batch before sending it (staleness avoidance timeout).
This value is in minutes . |
| <code>plugin.bitbucket-notification.batch.notification.flush.limit</code> | |
| 40 | The maximum number of notifications collected in a batch before the batch is sent automatically. Once this number of notifications is reached, the batch will be sent regardless of the time settings. |

POST service webhook

See [POST service webhook for Bitbucket Server](#) for more information.

| Default value | Description |
|--|--|
| <code>plugin.com.atlassian.stash.plugin.hook.threadPoolCoreSize</code> | |
| 2 | Core size of thread pool the default number of concurrent hooks notifications. |
| <code>plugin.com.atlassian.stash.plugin.hook.threadPoolMaxSize</code> | |
| 3 | Maximal size of thread pool the maximum number of concurrent hooks notifications. |
| <code>plugin.com.atlassian.stash.plugin.hook.queueSize</code> | |
| 1024 | The maximum size of the queue which holds queued requests that are yet to be sent.
When this size is exceeded the oldest unsent message will be dropped and a warning message logged. |
| <code>plugin.com.atlassian.stash.plugin.hook.connectionTimeout</code> | |
| 10000 | Connection timeout for hook request.
When the connection times out a warning message will be logged.
This value is in milliseconds . |
| <code>plugin.com.atlassian.stash.plugin.hook.changesetsLimit</code> | |
| 500 | Limit of maximum count of changesets that will be sent in the POST data for a single ref change. |
| <code>plugin.com.atlassian.stash.plugin.hook.changesLimit</code> | |
| 100 | Limit of maximum count of changes for a single changeset in the POST data. |

Paging

These properties control the maximum number of objects which may be returned on a page, regardless of how many were actually requested by the user. For example, if a user requests `Integer.MAX_INT` branches on a page, their request will be limited to the value set for `page.max.branches`.

This is intended as a safeguard to prevent enormous requests from tying up the server for extended periods of time and then generating responses whose payload is prohibitively large. The defaults configured here represent a sane baseline, but may be overridden by customers if necessary.

| Default value | Description |
|---------------------------------------|--|
| page.max.attachments | |
| 500 | Maximum number of attachments per page. |
| page.max.branches | |
| 1000 | Maximum number of branches per page. |
| page.max.changes | |
| 1000 | Maximum number of changes per page. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when the number of changes in a commit exceeds this size. |
| page.max.commits | |
| 100 | Maximum number of commits per page. |
| page.max.diff.lines | |
| 10000 | Maximum number of segment lines (of any type, total) which may be returned for a single diff. Unlike other page limits, this is a hard limit; subsequent pages cannot be requested when a diff exceeds this size. |
| page.max.directory.children | |
| 1000 | Maximum number of directory entries which may be returned for a given directory. |
| page.max.directory.recursive.children | |
| 100000 | Maximum number of file entries which may be returned for a recursive listing of a directory. A relatively high number as this is used by the file finder which needs to load the tree of files upfront. |
| page.max.groups | |
| 1000 | Maximum number of groups per page. |
| page.max.granted.permissions | |
| 1000 | Maximum number of granted permissions per page. |
| page.max.hooksripts | |
| 100 | Maximum number of hook scripts per page. |
| page.max.index.results | |
| 50 | Maximum number of commits which may be returned from the index when querying by an indexed attribute. For example, this limits the number of commits which may be returned when looking up commits against a Jira issue. |
| page.max.projects | |
| 1000 | Maximum number of projects per page. |
| page.max.pullrequest.activities | |
| 500 | Maximum number of pull request activities per page. |

| | |
|---------------------------------------|--|
| page.max.pullrequests | |
| 1000 | Maximum number of pull requests per page. |
| page.max.repositories | |
| 1000 | Maximum number of repositories per page. |
| page.max.reviewergroups | |
| 100 | Maximum number of reviewer groups per page. |
| page.max.source.length | |
| 5000 | Maximum length for any line returned from a given file when viewing source. This value truncates long lines. There is no mechanism for retrieving the truncated part short of downloading the entire file. |
| page.max.source.lines | |
| $\{\text{display.max.source.lines}\}$ | Maximum number of lines which may be returned from a given file when viewing source. This value breaks large files into multiple pages. This property relates to <code>display.max.source.line</code> in that up to $(\text{display.max.source.lines}/\text{page.max.source.lines})$ requests will be made to view the page. |
| page.max.tags | |
| 1000 | Maximum number of tags per page. |
| page.max.users | |
| 1000 | Maximum number of users per page. |
| page.scan.pullrequest.activity.size | |
| 500 | The page size to use when searching activities to find a specific one. |
| page.scan.pullrequest.activity.count | |
| 4 | The number of pages of activities to scan, when searching for a given activity, before giving up. |

Password Reset

| Default value | Description |
|--------------------------------|--|
| password.reset.validity.period | |
| 4320 | Controls how long a password reset token remains valid for. Default period is 72 hours.
This value is in minutes . |

Process execution

Controls timeouts for external processes, such as `git`.

| Default value | Description |
|---------------------------|-------------|
| process.timeout.execution | |

| | |
|----------------------|---|
| 120 | Configures a hard upper limit on how long the command is allowed to run even if it is producing output.

This value is in seconds . Using 0, or a negative value, disables the timeout completely. |
| process.timeout.idle | |
| 60 | The idle timeout configures how long the command is allowed to run without producing any output.

This value is in seconds . Using 0, or a negative value, disables the timeout completely. |

Profiling

| Default value | Description |
|--------------------------------|--|
| atlassian.profile.mintime | |
| 1 | Controls the threshold time below which a profiled section should not be reported.

This value is in milliseconds . |
| atlassian.profile.mintotaltime | |
| 0 | Controls the threshold time below which a profiled request should not be reported.

This value is in milliseconds . |

Pull Request - Reviewer Groups

| Default value | Description |
|-------------------------------------|--|
| pullrequest.reviewergroups.max.size | |
| 100 | Defines the maximum number of users a reviewer group may contain |

Pull Request - Suggestions

| Default value | Description |
|---------------------------------------|---|
| pullrequest.suggestions.commit-author | |
| AUTHOR | Defines whether the author of the commit that results when applying a suggestion should be the suggestion author or the user who applied the suggestion. Valid values are: - AUTHOR the suggestion commit author should be the suggestion author - ACTOR the suggestion commit author should be the user who applied the suggestion Regardless of this value, the merge commit committer will always be the user who merged the pull request. |
| pullrequest.suggestions.drift.timeout | |
| 30 | Defines the maximum amount of time SCM commands used to drift suggestions back to the source branch are allowed to execute before they are terminated. In most cases a user will be waiting whilst this happens, so we need to balance between applying the suggestion and user experience. If this timeout is exceeded, the user is advised to apply the suggestion manually.

This value is in seconds . |

Pull Request Commit Indexing

Controls how commits are associated with pull requests.

| Default value | Description |
|---|---|
| <code>pullrequest.commit.indexing.maximum.commits</code> | |
| 1000 | Defines the maximum number of commits that will be associated with a pull request when a pull request is created or when new commits are pushed to a pull request. A larger number will impact pull request creation and rescoping time. Pull requests that have more than this number of commits may not appear linked in the commit screen. |
| <code>pullrequest.commit.indexing.backfill.maximum.processed</code> | |
| 5000 | Defines the maximum number of pull requests to process at once while backfilling pull request commits. A lower limit will mean less data needs to be stored in memory at once, but processing will take longer as a result. |
| <code>pullrequest.commit.indexing.backfill.batch.size</code> | |
| 250 | Defines the maximum number of pull request commit relationships to commit to the database in a single transaction. |
| <code>pullrequest.commit.indexing.backfill.process.timeout</code> | |
| 600 | Defines the maximum amount of time SCM commands used to backfill pull request commits are allowed to execute before they are terminated.

This value is in seconds . |

Pull Requests

| Default value | Description |
|---|---|
| <code>pullrequest.bulk.rescope.timeout</code> | |
| 300 | Defines the maximum amount of time any command used to analyse the effects of ref changes to open pull requests (rescoping) is allowed to execute. For these commands, the standard idle timeout for processes applies.

This value is in seconds |
| <code>pullrequest.deletion.role</code> | |
| AUTHOR | Determines which users can delete pull requests. Valid values are: - AUTHOR pull request authors and repository admins can delete pull requests - REPO_ADMIN repository admins can delete pull requests |
| <code>pullrequest.diff.context</code> | |
| 10 | Defines the number of context lines to include around diff segments in pull request diffs. By default, git only includes 3 lines. The default is 10, to try and include a bit more useful context around changes, until the ability to "expand" the context is implemented. |
| <code>pullrequest.diff.context.expand.size</code> | |
| 25 | Defines the number of context lines to show at a time when clicking the expand buttons around segments in the pull request diff. |
| <code>pullrequest.merge.commit-author</code> | |

| | |
|--|--|
| AUTHOR | Defines whether the author of the merge commit that results when merging a pull request should be the pull request author or the user who merged the pull request. Valid values are: - <code>AUTHOR</code> the merge commit author should be the pull request author - <code>ACTOR</code> the merge commit author should be the user who merged the pull request Regardless of this value, the merge commit committer will always be the user who merged the pull request. |
| pullrequest.merge.timeout | |
| 300 | Defines the maximum amount of time any command used to merge a pull request is allowed to execute or idle. Since merging a pull request may require a series of different commands at the SCM level, this timeout does <i>not</i> define the upper bound for how long the overall merge process might take; it only defines the duration allotted to any single command.

This value is in seconds . |
| pullrequest.rescope.commits.display | |
| 5 | Defines the maximum number of commits per type (either added or removed) to display in a rescope activity. |
| pullrequest.rescope.commits.max | |
| 1000 | Defines the absolute maximum number of commits that will be evaluated when attempting to determine, for a given rescope activity, which commits were added to or removed from a pull request. Adjusting this setting can have significant memory footprint impact on the system. It is not recommended to be changed, but the option is provided here to support unique use cases. |
| pullrequest.rescope.cleanup.interval | |
| 30 | Controls how frequently empty rescope activities are cleaned up. Because pull requests rescope very frequently it is important to remove empty rescopes from the database to keep activity queries performant.

This value is in minutes . |
| pullrequest.rescope.detail.threads | |
| 2 | Defines the maximum number of threads to use for precalculating rescope details. These threads perform the requisite processing to determine the commits added and removed when a pull request is rescope, where most rescopes do not add or remove any commits. Such "dead" rescopes are deleted during processing. The primary goal is to ensure all details have already been calculated when users try to view a pull request's overview. |
| pullrequest.rescope.drift.commandtimeout | |
| 180 | Defines the maximum amount of time SCM commands used to perform comment drift are allowed to execute before they are terminated. Aggressive timeouts should <i>not</i> be used here. Commands which timeout may result in comments which are still present in the diff being orphaned or drifted incorrectly.

This value is in seconds . |
| pullrequest.rescope.drift.maxattempts | |
| 5 | Controls how many times comment drift will be retried when it fails. Certain failure types are considered to be unrecoverable and will not be retried regardless of this setting. Unrecoverable failures are logged. |
| pullrequest.rescope.drift.threads | |
| 4 | Defines the maximum number of threads to use when processing comment drift for a pull request during rescope. Higher numbers here do <i>not</i> necessarily mean higher throughput! Performing comment drift will usually force a new merge to be created, which can be very I/O intensive. Having a substantial number of merges running at the same time can significantly <i>reduce</i> the speed of performing comment drift. |

pullrequest.rescope.threads

| | |
|---|---|
| 1 | Defines the maximum number of threads to use for rescoping pull requests. |
|---|---|

Ref Restrictions (Branch permissions)

| Default value | Description |
|--|---|
| plugin.bitbucket-ref-restriction.case.insensitive | |
| true | Controls whether refs are matched case insensitively for ref restrictions. |
| plugin.ref-restriction.feature.ascii.art | |
| true | Controls whether ASCII art is displayed when a push is rejected. |
| plugin.ref-restriction.feature.splash | |
| true | Controls whether new users are shown a splash page when first viewing ref restrictions. |
| plugin.bitbucket-ref-restriction.max.resources | |
| 100 | The maximum number of ref restrictions per repository. This count refers to each permission item, not each branch. High numbers of branch permissions will adversely impact the speed of pushes to the repository, so increasing this limit is not recommended. |
| plugin.bitbucket-ref-restriction.max.resource.entities | |
| 50 | The maximum number of access grants per ref restriction. |

Ref metadata

| Default value | Description |
|--------------------------------|--|
| ref.metadata.timeout | |
| 2 | Controls timeouts for retrieving metadata associated with a collection of refs from all metadata providers collectively.

This values is in seconds . |
| ref.metadata.max.request.count | |
| 100 | Maximum number of refs that can be used in a metadata query. |

Ref searching

| Default value | Description |
|-------------------------------|---|
| ref.search.boost.branches.max | |
| 1000 | Maximum number of refs we can guarantee exact and prefix matching in correct ordering |

Repository shortcuts

| Default value | Description |
|---|--|
| <code>plugin.repository.shortcut.url.scheme.extended.whitelist</code> | |
| | The extended whitelist for allowed URL schemes. The URL for a repository shortcut must begin with one of the schemes in the default whitelist or a scheme in this property. Schemes should be comma separated, e.g. 'scheme1:,scheme2:'. |

Resource Throttling

These properties define concurrent task limits for the ThrottleService, limiting the number of concurrent operations of a given type that may be run at once. This is intended to help prevent overwhelming the server hardware with running processes. Two settings are used to control the number of processes that are allowed to process in parallel: one for the web UI and one for 'hosting' operations (pushing and pulling commits, cloning repositories).

When the limit is reached for the given resource, the request will wait until a currently running request has completed. If no request completes within a configurable timeout, the request will be rejected.

When requests to the UI are rejected, users will see either a 501 error page indicating the server is under load, or a popup indicating part of the current page failed to load.

When SCM hosting commands (pull/push/clone) are rejected, it is messaged in multiple ways:

- An error message is returned to the client, which the user will see on the command line: "Bitbucket is currently under heavy load and is not able to service your request. Please wait briefly and try your request again"
- A warning message is logged for every time a request is rejected due to the resource limits, using the following format: "A [scm-hosting] ticket could not be acquired (12/12)"
- For five minutes after a request is rejected, a red banner will be displayed in the UI to warn that the server is reaching resource limits.

The underlying machine-level limits these are intended to prevent hitting are very OS- and hardware-dependent, so you may need to tune them for your instance. When hyperthreading is enabled for the server's CPUs, for example, it is likely that the default settings will allow sufficient concurrent operations to saturate the I/O on the machine. In such cases, we recommend starting off with a less aggressive default on multi-cored machines; the value can be increased later if hosting operations begin to back up.

Additional resource types may be configured by defining a key with the format 'throttle.resource.<resource-name>'. When adding new types, it is strongly recommended to configure their ticket counts explicitly using this approach.

| Default value | Description |
|--|---|
| <code>scaling.concurrency</code> | |
| <code>cpu</code> | <p>Allows adjusting the scaling factor used. Various other CPU/throttling dependent properties are defined in terms of this setting, so adjusting this value implicitly adjusts those. Some examples:</p> <ul style="list-style-type: none"> • <code>event.dispatcher.core.threads</code> • <code>event.dispatcher.max.threads</code> • <code>executor.max.threads</code> • <code>throttle.resource.scm-hosting</code> <p>The default value, <code>cpu</code>, resolves to the number of <i>detected</i> CPU cores. On hyperthreaded machines, this will be <i>double</i> the amount of physical cores.</p> |
| <code>throttle.resource.git-lfs</code> | |

| | |
|--|--|
| 80 | Limits the number of Git LFS file transfer operations which may be running concurrently. This is primarily intended to prevent Git LFS requests from consuming all available connections thereby degrading UI and Git hosting operation. This should be a fraction of the maximum number of concurrent connections permitted by Tomcat. |
| throttle.resource.git-lfs.timeout | |
| 0 | Controls how long threads will wait for Git LFS uploads/downloads to complete when the system is already running the maximum number of concurrent transfers. It is recommended this be set to zero (i.e. don't block) or a few seconds at most. Since waiters still hold a connection, a non-zero wait time defeats the purpose of this throttle.

This value is in seconds . |
| throttle.resource.mirror-hosting | |
| 2*\${scaling.concurrency} | Limits the number of SCM hosting operations served to mirrors, which may be running concurrently. This limit is intended to protect the system's CPU and memory from being consumed excessively by mirror operations.

Note that dynamic throttling is not supported for mirror hosting operations. |
| throttle.resource.mirror-hosting.timeout | |
| 3600 | Controls how long threads will wait for SCM mirrors hosting operations to complete when the system is already running the maximum number of SCM commands.

This value is in seconds . |
| throttle.resource.scm-command | |
| 40 | Limits the number of SCM commands, such as: <code>git diff</code> , <code>git blame</code> , or <code>git rev-list</code> , which may be running concurrently. This limit is intended to prevent the operations which support the UI from preventing push/pull operations from being run. |
| throttle.resource.scm-command.timeout | |
| 2 | Controls how long threads will wait for SCM commands to complete when the system is already running the maximum number of SCM commands.

This value is in seconds . |
| throttle.resource.scm-hosting.timeout | |
| 300 | Controls how long threads will wait for SCM hosting operations to complete when the system is already running the maximum number of SCM hosting operations.

This value is in seconds . |
| throttle.resource.scm-hosting.strategy | |

| | |
|--|---|
| adaptive | <p>Specifies the strategy for throttling SCM hosting operations. Possible values are 'adaptive' and 'fixed'.</p> <p>If 'fixed' is specified, <code>throttle.resource.scm-hosting.fixed.limit</code> is used as the fixed upper limit on the number of concurrent hosting operations.</p> <p>If 'adaptive' is specified, the maximum number of hosting operations will vary between <code>throttle.resource.scm-hosting.adaptive.min</code> and <code>throttle.resource.scm-hosting.adaptive.max</code> based on how many hosting operations the system believes the machine can support in its current state and given past performance.</p> <p>If any configured adaptive throttling setting is invalid and reverts to a default but this conflicts with other correctly configured or default settings, the throttling strategy will revert to 'fixed'. E.g. this will occur if <code>throttle.resource.scm-hosting.adaptive.min</code> is set to the same value as <code>throttle.resource.scm-hosting.adaptive.max</code></p> |
| <code>throttle.resource.scm-hosting.adaptive.limit.min</code> | |
| 1* <code>{scaling.concurrency}</code> | <p>When the adaptive strategy is enabled for throttling SCM hosting operations, this sets the lower limit on the number of SCM hosting operations, meaning pushes and pulls over HTTP or SSH, which may be running concurrently.</p> <p>Setting a lower limit provides a way to specify a minimum service level for SCM hosting operations regardless of what the adaptive throttling heuristic believes the machine can handle.</p> <p>There is limited support for mathematical expressions; +, -, *, / and () are supported.</p> |
| <code>throttle.resource.scm-hosting.adaptive.limit.max</code> | |
| 4* <code>{scaling.concurrency}</code> | <p>When the adaptive strategy is enabled for throttling SCM hosting operations, this sets the upper limit on the number of SCM hosting operations, meaning pushes and pulls over HTTP or SSH, which may be running concurrently. This is intended primarily to prevent pulls, which can be very memory-intensive, from exhausting a server's resources. Note that if the machine does not have sufficient memory to support this default value or an explicitly configured value, a smaller value will be chosen on startup.</p> <p>Adaptive throttling will vary the total tickets There is limited support for mathematical expressions; +, -, *, / and () are supported.</p> |
| <code>throttle.resource.scm-hosting.adaptive.cpu.target</code> | |
| 0.75 | <p>When the adaptive strategy is enabled for throttling SCM hosting operations, this sets the target CPU utilisation for the machine (across all processors) which the system takes into consideration when calculating the current throttling limit.</p> <p>This value represents a trade-off: higher numbers may boost raw throughput of hosting operations, but at the expense of overall system responsiveness for all users. Increasing the target too high or too low brings diminishing returns.</p> <p>This must be a value between 0.0 and 1.0 and is a percentage of the total available CPU power across all cores</p> |
| <code>throttle.resource.scm-hosting.fixed.limit</code> | |
| 1.5* <code>{scaling.concurrency}</code> | <p>When the fixed strategy is enabled for throttling SCM hosting operations, this limits the number of SCM hosting operations, meaning pushes and pulls over HTTP or SSH, which may be running concurrently. This is intended primarily to prevent pulls, which can be very memory-intensive, from exhausting a server's resources. There is limited support for mathematical expressions; +, -, *, / and () are supported.</p> |
| <code>throttle.resource.scm-refs</code> | |

| | |
|---|---|
| <code>8*\${scaling.concurrency}</code> | Limits the number of ref. advertisement operations, which may be running concurrently. Those are throttled separately from clone operations as they are much more lightweight and much shorter so we can and should allow many more of them running concurrently. |
| <code>throttle.resource.scm-refs.timeout</code> | |
| 120 | Controls how long threads will wait for ref. advertisement operations to complete when the system is already running the maximum number of ref. advertisement operations.

This value is in seconds . |

Rest

| Default value | Description |
|---|--|
| <code>plugin.rest.raw.content.markup.max.size</code> | |
| <code>\${plugin.bitbucket-readme.max.size:5242880}</code> | Controls the maximum allowed file size when downloading a marked-up file through the raw content REST endpoint

This value is in bytes . Default is 5 MiB |

SCM - Cache

See [Scaling for Continuous Integration performance](#) for more information about configuring the SCM Cache.

Note: The settings controlled by these properties can be configured via REST. The REST configuration *takes precedence* over the configuration in `bitbucket.properties`.

| Default value | Description |
|---|--|
| <code>plugin.bitbucket-scm-cache.expiry.check.interval</code> | |
| 30 | Controls how frequently expired cache entries are invalidated and removed from the cache.

This value is in seconds . |
| <code>plugin.bitbucket-scm-cache.eviction.hysteresis</code> | |
| 107374
1824 | When eviction is triggered the amount of disk space requested for eviction is calculated as $(\text{eviction.hysteresis} + \text{eviction.trigger.free.space} - \text{free space under <Bitbucket home directory>/caches})$

This value is in bytes . |
| <code>plugin.bitbucket-scm-cache.eviction.trigger.free.space</code> | |
| 644245
0944 | Controls the threshold at which eviction is triggered in terms of free space available on disk (specifically under <code><Bitbucket home directory>/caches</code>)

This value is in bytes . |
| <code>plugin.bitbucket-scm-cache.minimum.free.space</code> | |
| 536870
9120 | Controls how much space needs to be available on disk (specifically under <code><Bitbucket home directory>/caches</code>) for caching to be enabled. This setting ensures that the cache plugin does not fill up the disk.

This value is in bytes . |

| | |
|--|---|
| plugin.bitbucket-scm-cache.protocols | |
| HTTP ,
SSH | Controls which protocols caching is applied to. The HTTP value includes both <code>http</code> and <code>https</code> .

This property value is a comma-separated list. Valid values are: HTTP and SSH. |
| plugin.bitbucket-scm-cache.capabilities.enabled | |
| true | Controls whether git v2 capabilities advertisement is cached. |
| plugin.bitbucket-scm-cache.capabilities.maxCount | |
| 1 | The maximum number of git v2 capabilities advertisement to retain <i>per repository</i> . If there are more than this configured limit, the least recently accessed entry will be invalidated. |
| plugin.bitbucket-scm-cache.capabilities.ttl | |
| 3600 | Controls the 'time to live' for git v2 capability advertisement caches.

This value is in seconds . |
| plugin.bitbucket-scm-cache.upload-pack.enabled | |
| true | Controls whether caching is enabled for git-upload-pack (clone operations). |
| plugin.bitbucket-scm-cache.upload-pack.maxCount | |
| 20 | The maximum number of upload-pack cache entries to retain <i>per repository</i> . If there are more than this configured limit, the least recently accessed entry will be invalidated. |
| plugin.bitbucket-scm-cache.upload-pack.ttl | |
| 14400 | Controls how long the caches for clone operations are kept around when there no changes to the repository.

Caches are automatically invalidated when someone pushes to a repository or when a pull request is merged.

This value is in seconds . |

SCM - Misc

| Default value | Description |
|---------------------------------|---|
| http.cloneurl.includeusername | |
| false | Controls whether the HTTP clone URL should contain the currently authenticated user's username |
| http.scmrequest.async.enabled | |
| true | Controls whether asynchronous process is enabled for HTTP SCM requests. Asynchronous processing can significantly increase the server's ability to service UI users while under heavy HTTP hosting load |
| http.scmrequest.async.keepalive | |
| 300 | Controls how long asynchronous request threads are allowed to idle before they're terminated. Aggressive timeouts may reduce the number of idle threads, but may also reduce the server's ability to respond to load spikes.

This value is in seconds . |
| http.scmrequest.async.queue | |

| | |
|--|---|
| 0 | Do not use. This property is deprecated and, as of 7.4.2, 7.5.1 and 7.6.0 (or newer), it no longer does anything. |
| <code>http.scmrequest.async.threads</code> | |
| 250 | Controls how many threads are used to process HTTP SCM requests asynchronously. Asynchronous processing frees up servlet container threads to allow them to handle other requests, like page requests for UI users. |

SCM - git

| Default value | Description |
|--|---|
| <code>plugin.bitbucket-git.path.executable</code> | |
| <code>git</code> | <p>Defines the default path to the git executable. On Windows machines, the <code>.exe</code> suffix will be added to the configured value automatically if it is not present. In general, "git" should be an acceptable default for every platform, here, assuming that it will be available in the runtime user's PATH.</p> <p>With the new path searching performed by <code>DefaultGitBinaryHelper</code>, setting a default value here is unnecessary, as the plugin will quickly discard the value. This is left here purely for documenting how to set an explicit path.</p> |
| <code>plugin.bitbucket-git.path.libexec</code> | |
| | <p>Defines the path to the git libexec directory (containing the git-core directory). This path is hard-coded into the git executable and is used for forking processes like <code>git-http-backend</code>. If this value is set, those processes will be forked directly. This eliminates an unnecessary fork (<code>git -> git-http-backend</code>) and may improve scalability.</p> |
| <code>plugin.bitbucket-git.author.name.type</code> | |
| <code>displayname</code> | <p>Defines whether commits created by the system should use the username (jdoe) or the display name (John Doe) to specify the Git author/commmitter. By default, the display name will be used.</p> <p>This value can be either <code>displayname</code> or <code>username</code>.</p> |
| <code>plugin.bitbucket-git.diff.renames</code> | |

| | |
|---|--|
| copies | <p>Defines whether copy and/or rename detection should be performed. By default, both rename <i>and</i> copy detection are performed. Only files modified in the same commit are considered as rename or copy origins, to minimize overhead.</p> <p>The possible settings are:</p> <ul style="list-style-type: none"> • "copy" or "copies"
Applies <code>--find-copies</code>. • "rename" or "renames"
Applies <code>--find-renames</code>. • "off"
Disables rename <i>and</i> copy detection. <p>When using "copy" or "copies", the value may optionally be suffixed with a "+" to use <code>--find-copies-harder</code>. This setting should be used with caution, as it can be very expensive. It considers every file in the repository, even files not modified in the same commit, as possible origins for copies. When copy and/or rename detection is enabled <code>plugin.bitbucket-git.diff.renames.threshold</code> can be used control the similarity index required for a change to be identified as a copy or rename. This configuration can also be applied at repository level with <code>plugin.bitbucket-git.diff.renames.KEY.slug</code> or at project level with <code>plugin.bitbucket-git.diff.renames.KEY</code></p> |
| <code>plugin.bitbucket-git.diff.renames.threshold</code> | |
| 50 | <p>Defines the threshold, as a percentage, for a file to be detected as a rename or a copy. This setting is only applied if copy and/or rename detection is enabled. The default threshold applied is 50% similarity (defined in git itself).</p> <p><code>git diff</code> and <code>git diff-tree</code> <i>do not honor</i> 100 (identical files only) for the threshold. They ignore the threshold and apply the default 50% threshold instead. A configured threshold of 100 will be applied as 99. Similarly, a configured threshold that is 0, or negative, will be applied as 1. This configuration can also be applied at repository level with <code>plugin.bitbucket-git.diff.renames.threshold.KEY.slug</code> or at project level with <code>plugin.bitbucket-git.diff.renames.threshold.KEY</code></p> |
| <code>plugin.bitbucket-git.environment.commandsizesize</code> | |
| 32000 | <p>Defines the maximum number of characters that can be added to a single command. Different operating systems (and even different versions of the same operating system) have different hard limitations they apply to command line lengths. This default is intended to be low enough to work on all supported platforms out of the box, but still high enough to be usable. It is configurable in case it proves to be too high on some platform. This default is based on Windows, which has a limit of 32768 characters. Testing on Linux (Ubuntu 12.04 running a 3.2 kernel) found that its limit is at least 32x the limit imposed by Windows.</p> |
| <code>plugin.bitbucket-git.environment.variablesizesize</code> | |
| 16000 | <p>Defines the maximum number of characters that can be added to a single environment variable. Different operating systems (and even different versions of the same operating system) have different hard limitations they apply to environment variables. This default is intended to be low enough to work on all supported platforms out of the box, but still high enough to be usable. It is configurable in case it proves to be too high on some platform.</p> |
| <code>plugin.bitbucket-git.hosting.allow-filter</code> | |
| true | <p>Controls whether partial clones, using <code>--filter</code>, are allowed. Partial clones are not cached, and some of the filters offered by Git can be very resource-intensive for the server to apply, so it can sometimes be more efficient to use a normal clone (or a shallow one) instead. Partial clones are enabled by default.</p> |
| <code>plugin.bitbucket-git.hosting.http.bufferbuffersize</code> | |

| | |
|--|--|
| 8192 | Defines the buffer size in bytes which is used when marshaling data between the git process and the HTTP socket. The default is 8K, with a 1K minimum. |
| plugin.bitbucket-git.hosting.ssh.buffersize | |
| 4096 | Defines the buffer size in bytes which is used when marshaling data between the git process and the SSH socket. The default is 4K, with a 1K minimum. |
| plugin.bitbucket-git.hosting.timeout.execution | |
| 86400 | Defines the execution timeout for push/pull processes, applying a hard limit to how long the operation is allowed to run even if it is producing output or reading input. The default value is 1 day, with a 5 minute minimum.

This value is in seconds . |
| plugin.bitbucket-git.hosting.timeout.idle | |
| 1800 | Defines the idle timeout for push/pull processes, applying a limit to how long the operation is allowed to execute without either producing output or consuming input. The default value is 30 minutes, with a 2 minute minimum.

This value is in seconds . |
| plugin.bitbucket-git.pullrequest.merge.auto.timeout | |
| <code>\${pullrequest.merge.timeout}</code> | Defines the maximum amount of time any command used to calculate a pull request's effective diff, or check for conflicts, is allowed to execute <i>or</i> idle. Because the commands used generally do not produce output there is no separate idle timeout.

This setting is deprecated. Use <code>plugin.bitbucket-git.pullrequest.operation.timeout</code> instead. See the documentation for that property for additional details about what operations this timeout applies to.

This value is in seconds . |
| plugin.bitbucket-git.pullrequest.operation.timeout | |
| <code>\${plugin.bitbucket-git.pullrequest.merge.auto.timeout}</code> | Defines the maximum amount of time any command used to calculate a pull request's effective diff, or check for conflicts, is allowed to execute <i>or</i> idle. Because the commands used generally do not produce output there is no separate idle timeout.

This timeout applies to operations that are run in the background. "Foreground" operations, like showing the file tree or an individual file's diff, do not use this timeout. However, in certain cases, such foreground operations may be blocked if a relevant background operation has not yet completed. For example, if a pull request's effective diff has not been calculated, displaying the file tree will block while the effective diff is calculated.

Using an aggressive timeout here may result in pull requests becoming unviewable. For example, if the system times out calculating a pull request's effective diff, it will not be possible to load the file tree and the pull request cannot be reviewed.

This value is in seconds . |
| plugin.bitbucket-git.ssh.binary | |
| ssh | Defines the SSH binary to use for <i>outgoing</i> SSH commands, i.e. git commands that fetch from or push to external repositories over SSH. This setting does <i>not</i> affect incoming SSH requests. |
| plugin.bitbucket-git.worktree.expiry | |
| 30 | Defines the amount of time after which a temporary work tree expires and can be cleaned up. A minimum value of 2 minutes and a maximum value of 24 hours is enforced.

This value is in minutes . The default is 30 minutes. |

SMTP

| Default value | Description |
|---------------------------|--|
| mail.timeout.connect | |
| 60 | Controls the timeout for establishing an SMTP connection.

This value is in seconds . |
| mail.timeout.send | |
| 60 | Controls the timeout for sending an e-mail.

This value is in seconds . |
| mail.test.timeout.connect | |
| 30 | Controls the timeout for establishing a test SMTP connection. Shorter timeouts should be applied for when sending test e-mails, as the test occurs in user time.

This value is in seconds . |
| mail.test.timeout.send | |
| 30 | Controls the timeout for sending a test e-mail. Shorter timeouts should be applied for when sending test e-mails, as the test occurs in user time.

This value is in seconds . |
| mail.error.pause.log | |
| 300 | Controls how frequently messages will go to the standard log file about mail sending errors. All errors are logged to <code>atlassian-bitbucket-mail.log</code> , but warnings will be added to the standard log periodically if there are errors sending messages.

This value is in seconds |
| mail.error.pause.retry | |
| 5 | Controls how long to wait before retrying to send a message if an error occurs.

This value is in seconds |
| mail.threads | |
| 1 | Controls the number of threads to use for sending emails. Higher values may result in higher mail server load, but may also allow the system to work through its internal queue faster. |
| mail.max.message.size | |
| 1048576 | Controls the maximum allowed size of a single mail message in bytes, which is the sum of the subject and body sizes.

This value is in bytes . |
| mail.max.queue.size | |
| 157286400 | Controls the maximum allowed size for the mail queue in bytes (any new message will be rejected if the mail queue reaches that size)

This value is in bytes . |

| | |
|-------------------------------------|--|
| mail.max.shutdown.wait | |
| 5 | Controls the maximum time to wait for the mail queue to empty on shutdown. Once this time elapses, any mail remaining in the queue will be logged as rejected in the mail log and dropped. A value of 0 or less means no waiting will occur, and all mail in the queue will be dropped on shutdown.

This value is in seconds . |
| mail.crypto.protocols | |
| TLSv1
TLSv1.
1
TLSv1.
2 | Space-separated list of crypto protocols to use when sending encrypted email. This default value is POODLE-safe. Order does not matter - JavaMail always tries the latest supported protocol first. An empty value causes the product to use all protocols supported by the shipped version of JavaMail which may not be POODLE-safe |
| mail.crypto.ciphers | |
| | Space-separated list of ciphers to use when connecting via SSL or TLS. An empty value causes the product to use all ciphers supported by the JVM |

SSH

| Default value | Description |
|--------------------|---|
| plugin.ssh.address | |
| | Sets the address where the application will listen for SSH connections. By default the application will accept SSH connections on all of its network interfaces. This property can be used to restrict that to specific interfaces, with multiple addresses separated by commas (without spaces). |
| plugin.ssh.port | |
| | Sets the port where the application will listen for SSH connections, 7999 by default. This value and the SSH base URL's(<code>plugin.ssh.baseurl</code>) port don't need to match, but they often should match.

If the SSH base URL and SSH port configurations are modified in the global Server settings page, the configurations specified in the properties file will no longer be used. |
| plugin.ssh.baseurl | |
| | Sets the URL on which SSH is accessible, this is used as the base for SSH clone URLs. If SSH is running on a non-standard port, the base URL must start with <code>ssh://</code> or clones will fail because the port syntax is ambiguous when paired with a path (e.g. <code>host:port/path</code> vs. <code>host:path</code>) and Git does not apply the port.

If the SSH base URL and SSH port configurations are modified in the global Server settings page, the configurations specified in the properties file will no longer be used. |

SSH command execution

| Default value | Description |
|---------------------------------|-------------|
| plugin.ssh.command.timeout.idle | |

| | |
|---------------------------------------|--|
| 7200 | <p>Controls timeouts for all SSH commands, such as those that service git and hg operations over SSH. The idle timeout configures how long the command is allowed to run without writing any output to the client. For SCM commands, the <code>plugin.*.hosting.timeout.idle</code> properties defined above will be applied to the underlying process. The default value is 2 hours.</p> <p>This values is in seconds.</p> |
| plugin.ssh.nio.workers | |
| -1 | <p>Controls the maximum number of NIO worker threads to process incoming SSH commands. The default value is -1, which will use a dynamic maximum based on the number of CPU cores + 1 (the default for the Apache SSHD library).</p> |
| plugin.ssh.auth.timeout | |
| 30 | <p>Controls the timeout when trying to authenticate the incoming SSH command. If authentication does not complete within this period the SSH command will fail. Using a shorter timeout hastens the rejection SSH commands under heavy load and can help keep open socket/file count down.</p> <p>This values is in seconds.</p> |
| plugin.ssh.session.pending-writes.max | |
| 3840 | <p>Controls the number of pending writes the throttling mechanism will allow SSH sessions. Throttling works as a flow control mechanism to prevent the system from writing too much data to Apache MINA's WriteRequestQueue. This is particularly helpful when clients (such as TortoiseGit) setup session channels with extremely large window sizes (2GBs) which means Apache MINA's own flow control mechanism will not stop the command from writing data.</p> <p>A default value of 512 means that at any given time the system will only be responsible for, at most, 4 MBs per session (some more data may be written to the queue as part of Apache MINA's own handling of the SSH protocol).</p> <p>Rate limiting will be applied to any SSH session which establishes a channel with a remote window size larger than $\{\text{plugin.ssh.session.pending-writes.max}\} * 8197$ (packet size optimized for Git).</p> <p>A value of 0 or less effectively disables session io rate limiting.</p> |
| plugin.ssh.session.max | |
| 250 | <p>Controls the maximum number of concurrent SSH sessions allowed. If this property is removed the system will set the limit at 250. If this property is configured below 100 the system will set the limit at 100. Increasing this will result in additional memory usage during peak load and can lead to out-of-memory errors.</p> |

SSH security

| Default value | Description |
|---|--|
| plugin.ssh.disabled.ciphers | |
| arcfour128,
arcfour256, aes128-cbc,
aes192-cbc, aes256-cbc,
3des-cbc, blowfish-cbc | <p>Controls which default ciphers are disabled when executing all SSH commands. Non existent ciphers are ignored. Names are case sensitive. If you override this property, the default values will NOT be appended, and should be included explicitly.</p> <p>Example value: <code>arcfour128,3des-cbc</code></p> <p>To enable additional ciphers see the KB article Disable default SSH algorithms.</p> |
| plugin.ssh.disabled.key.exchanges | |

| | |
|--|--|
| | <p>Controls which default key exchange algorithms are disabled when executing all SSH commands. Non existent key exchange algorithms are ignored. Names are case sensitive. If you override this property, the default values will NOT be appended, and should be included explicitly.</p> <p>Example value: <code>ecdh-sha2-nistp256,ecdh-sha2-nistp384</code></p> <p>To enable additional key exchange algorithms see the KB article Disable default SSH algorithms.</p> |
| <code>plugin.ssh.disabled.macs</code> | |
| <code>hmac-md5, hmac-sha1-96, hmac-md5-96</code> | <p>Controls which default macs are disabled when executing all SSH commands. Non existent macs are ignored. Names are case sensitive. If you override this property, the default values will NOT be appended, and should be included explicitly.</p> <p>Example value: <code>hmac-sha1-96,hmac-md5-96,hmac-md5</code></p> <p>To enable additional macs see the KB article Disable default SSH algorithms.</p> |

Search

| Default value | Description |
|--|---|
| <code>plugin.search.codesearch.indexing.enabled</code> | |
| <code>true</code> | Controls whether code indexing is enabled. Setting this to <code>false</code> prevents repository content from being indexed going forward. Existing repository content will <i>not</i> be un-indexed automatically. |
| <code>plugin.search.indexing.event.shutdown.timeout</code> | |
| <code>2</code> | Controls how long an elasticsearch indexing process is given to complete when Bitbucket is shutting down, before it is stopped forcefully. This value is in seconds . |
| <code>plugin.search.codesearch.indexing.exclude</code> | |
| | <p>Allows configuring strategies for excluding certain repositories from code search indexing. This can be used to reduce disk space requirements for the Elasticsearch index.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <code>all-forks</code> excludes all forks from code search indexing <code>personal-forks</code> excludes personal forks from code search indexing <code>synced-forks</code> excludes forks which have ref synchronization enabled from code search indexing <code>undiverged-forks</code> excludes forks from code search indexing which have ref synchronization enabled, and have not had their default branch updated |
| <code>plugin.search.pageSize.primary</code> | |
| <code>25</code> | Controls the page size of primary search results (eg. code for code search) |
| <code>plugin.search.pageSize.secondary</code> | |
| <code>10</code> | Controls the page size of secondary search results (eg. sidebar PRs/repos/etc) |

Server

| Default value | Description |
|---------------------------------|---|
| server.context-path | |
| / | Controls the context path where the application should be published, / by default. |
| server.display-name | |
| Atlassian Bitbucket | Controls the application's display name. |
| server.session.cookie.http-only | |
| true | Controls whether the session cookie should include an HttpOnly restriction, for those browsers that honor it. HttpOnly is enabled by default. |
| server.session.cookie.max-age | |
| 1209600 | Controls the value of "Max-Age" attribute in session cookie, for those browsers that honor it. This value is in seconds and the default value is 1209600 seconds i.e. 2 weeks. "Expires" attribute is also set based on this value which is absolute date and time when cookie will expire. |
| server.session.cookie.name | |
| BITBUCKETSESSIONID | Controls the name used for the session cookie, which is "BITBUCKETSESSIONID" by default. Note that most servlet containers use "JSESSIONID" by default. That is not used here to facilitate installations where multiple applications have the same hostname (typically via reverse proxying) with distinct context paths. In such a setup, using "JSESSIONID" can result in unexpected behavior where logging into one application logs users out of others. |
| server.session.timeout | |
| 1800 | Controls the session's timeout, which defaults to 30 minutes. Session timeout <i>may not</i> result in a user having to login again; they may automatically receive a new session via a separate remember-me token.

This value is in seconds . |
| server.session.tracking-modes | |
| cookie | Controls which mechanisms may be used for tracking sessions. By default, only "cookie" tracking is enabled. Other options are "ssl", to use the SSL session ID, and "url", to append the session ID to the URL after a semi-colon. Multiple values may be specified separated by commas (e.g. "cookie,url") |

Server Connectors

These properties control the primary server connector. Additional connectors can be configured using the prefix `server.additional-connector.#`, where # is the connector number. For example: to set a port on the first additional connector, the property would be `server.additional-connector.1.port=7991`. Numbers 1 to 5 are supported, allowing for 5 connectors in addition to the primary connector.

| Default value | Description |
|------------------------|-------------|
| server.ajp.packet-size | |

| | |
|--|---|
| 0 | <p>Controls the AJP packet size. When using the default value (0), Tomcat's default packet size is applied. The minimum value is 8192, and the maximum is 65536. If this property is overridden, the value set should match the <code>max_packet_size</code> directive for <code>mod_jk</code>.</p> <p>Note: Packet size can only be configured when <code>server.connector-protocol</code> is set to AJP/1.3; otherwise it will be ignored.</p> <p>This value is in bytes.</p> |
| server.ajp.secret | |
| | <p>Controls the secret that is used to secure the AJP connector. By default no secret is used or required. If a secret is configured, by default it will be required.</p> |
| server.ajp.secret-required | |
| | <p>Controls whether the configured secret is required when connecting via AJP. If no secret is configured, any value configured here is ignored and the property is always set to <code>false</code>. If a secret is configured and this value is not set, it defaults to <code>true</code>.</p> |
| server.address | |
| | <p>Controls the network address the application will bind to, 0.0.0.0 by default.</p> |
| server.compression.enabled | |
| true | <p>Controls whether the server will attempt to compress data to reduce network costs. By default, compression is enabled on all HTTP/1.1 connectors.</p> <p>Note: Compression cannot be used when <code>server.connector-protocol</code> is set to AJP/1.3, and it will be ignored if enabled.</p> |
| server.compression.excluded-user-agents | |
| | <p>Controls the list of user-agents to exclude from compression.</p> |
| server.compression.mime-types | |
| text/css, text/html, text/javascript, text/json, text/plain, text/xml, text/x-javascript, application/javascript, application/json, application/x-javascript, application/vnd.git-lfs+json | <p>Controls which MIME types are compressed, when compression is enabled. CSS, HTML, JavaScript and JSON are compressed by default.</p> |
| server.compression.min-response-size | |
| | <p>Controls the minimum response length for the compression to be performed. Only the mime-types specified as part of the <code>server.compression.mime-types</code> will be compressed.</p> <p>This value is in bytes.</p> |
| server.connection-timeout | |

| | |
|-----------------------------|---|
| 20000 | <p>Controls the connection timeout, which defines how long the server will wait for a request after a connection is established.</p> <p>This value is in milliseconds.</p> |
| server.connector-protocol | |
| HTTP/1.1 | <p>Controls the wire protocol used by the primary connector, which is HTTP/1.1 by default.</p> <p>The following values are supported: * HTTP/1.1 or org.apache.coyote.http11.Http11NioProtocol (Default): A standard HTTP 1.1 connector, which can be configured with or without SSL. All of the various settings are supported for HTTP 1.1. * AJP/1.3 or org.apache.coyote ajp.AjpNioProtocol: An AJP connector. Several settings, such as max-http-header-size and all of the compression and SSL settings, are not honored when using AJP.</p> <p><i>Apache Portable Runtime (APR) and NIO2-based connectors are not supported and may not be used. Attempting to configure either type will result in the system failing during startup.</i></p> |
| server.max-http-header-size | |
| 0 | <p>Controls the maximum size of the HTTP message header. When using the default value (0), Tomcat's default limit is applied.</p> <p>Note: The max HTTP header size cannot be configured when server.connector-protocol is set to AJP/1.3, and it will be ignored if set; server.packet-size may be the property you're looking for instead.</p> <p>This value is in bytes.</p> |
| server.max-http-post-size | |
| 0 | <p>Maximum size of the HTTP post or put content. When using the default value (0), Tomcat's default limit is honored.</p> <p>This value is in bytes.</p> |
| server.packet-size | |
| 0 | <p>Controls the AJP packet size. When using the default value (0), Tomcat's default packet size is applied. The minimum value is 8192, and the maximum is 65536. If this property is overridden, the value set should match the max_packet_size directive for mod_jk.</p> <p>Note: Packet size can only be configured when server.connector-protocol is set to AJP/1.3; otherwise it will be ignored. Deprecated in 7.14. Use server.ajp.packet-size instead.</p> <p>This value is in bytes.</p> |
| server.port | |
| 7990 | <p>Controls the port where the application will listen for connections, 7990 by default.</p> |
| server.proxy-name | |
| | <p>The proxy name, used to construct proper redirect URLs when a reverse proxy is in use.</p> |
| server.proxy-port | |

| | |
|------------------------|--|
| | The proxy port, used to construct proper redirect URLs when a reverse proxy is in use. If a proxy port is not set, but a name is, the connector's port will be used as the default. |
| server.redirect-port | |
| | The redirect port to use when redirecting from non-SSL to SSL. Defaults to 443, the standard SSL port. |
| server.require-ssl | |
| false | Require an SSL connection when connecting to the server. Setting this to "true" will automatically redirect insecure connections to the configured "redirect-port" for their connectors. |
| server.scheme | |
| http | The connector scheme, either "http" or "https". Defaults to "http" unless "secure" is set to "true"; then it defaults to "https". In general, this property should not need to be set.

Note: The scheme cannot be configured when <code>server.connector-protocol</code> is set to AJP/1.3, and it will be ignored if set. |
| server.secure | |
| false | Whether the connector is secured. Note that setting this to "true" does <i>not</i> enable SSL; SSL is enabled using <code>server.ssl.enabled</code> instead. One use case for setting this to "true" is when the system is behind a reverse proxy and SSL is terminated at the proxy.

Note: The secure flag cannot be configured when <code>server.connector-protocol</code> is set to AJP/1.3, and it will be ignored if set. |
| server.server-header | |
| | Controls the value to use for the Server response header.

If empty, which is the default, no header is sent. |
| server.ssl.ciphers | |
| | Controls the supported SSL ciphers for the connector.

This property value is a comma-separated list. |
| server.ssl.client-auth | |
| | Controls whether the client authentication is wanted ("want") or needed ("need"), this setting directly relates to the <code>clientAuth</code> option for Tomcat. Requires a configured trust store.

Default is empty, which corresponds to the Tomcat's default value of false. |
| server.ssl.enabled | |
| false | Controls whether SSL is enabled. By default, SSL is disabled and the primary connector uses unsecured HTTP.

Note: SSL cannot be used when the <code>server.connector-protocol</code> is set to AJP/1.3, and attempting to enable it will prevent the server from starting. |
| server.ssl.key-alias | |

| | |
|--|--|
| tomcat | Controls the alias used when selecting the key to use in the keystore. For compatibility with keys setup for Bitbucket Server 4.x and earlier, the default is "tomcat". |
| server.ssl.key-password | |
| changeit | Controls the password used to access the key within the keystore. (<code>server.ssl.key-store-password</code> is used to control the password for accessing the keystore itself.) For compatibility with keys setup for Bitbucket Server 4.x and earlier, the default is "changeit". |
| server.ssl.key-store | |
| <code>\${bitbucket.shared.home}/config/ssl-keystore</code> | Controls the location of the keystore, "\$BITBUCKET_HOME/shared/config/ssl-keystore" by default. |
| server.ssl.key-store-password | |
| changeit | Controls the password used to access the keystore. (<code>server.ssl.key-password</code> is used to control the password for accessing the specific key within the keystore.) For compatibility with keys setup for Bitbucket Server 4.x and earlier, the default is "changeit". |
| server.ssl.key-store-type | |
| <code>\${keystore.type:jks}</code> | Controls the keystore type. The JVM's default keystore type as returned by <code>KeyStore.getDefaultType()</code> , typically "jks", is used by default. |
| server.ssl.enabled-protocols | |
| all | <p>Controls which SSL protocols the connector will allow when communicating with clients. This value can be a comma-separated list (without spaces) to allow multiple protocols.</p> <p>Possible values include: * SSLv2Hello * SSLv3 * TLSv1 * TLSv1.1 * TLSv1.2 * TLSv1.3 (Requires a supporting JVM) * all (Default; equivalent to "SSLv2Hello,TLSv1,TLSv1.1,TLSv1.2,TLSv1.3")</p> <p>Note that SSLv2 and SSLv3 are inherently unsafe and should not be used.</p> |
| server.ssl.protocol | |
| TLS | <p>Controls the SSL protocol the connector will use by default.</p> <p>To control which protocols are <i>supported</i>, set <code>server.ssl.enabled-protocols</code> instead. Even if the connector defaults to a given protocol, clients can use any <i>enabled</i> protocol.</p> |
| server.ssl.trust-store | |
| | Controls which trust store holds the SSL certificates |
| server.ssl.trust-store-password | |
| | Controls the password to be used to access the trust store. |
| server.ssl.trust-store-provider | |
| | Controls the provider for the trust store. |
| server.ssl.trust-store-type | |
| | Controls the type for the trust store. |

Server busy banners

| Default value | Description |
|--|---|
| <code>server.busy.on.ticket.rejected.within</code> | |
| 5 | Controls how long a warning banner is displayed in the UI after a request is rejected due to excessive load.

This value is in minutes . Using 0, or a negative value, disables displaying the banner. |
| <code>server.busy.on.queue.time</code> | |
| 60 | Controls how long requests need to be queued before they cause a warning banner to appear.

This value is in seconds . Using 0, or a negative value, disables displaying the banner. |

Setup automation

If these properties are specified in `bitbucket.properties`, when the Setup Wizard runs after installing Bitbucket Server their values will be used and the Setup Wizard will not display the corresponding configuration screens.

You can use these properties to automate setup and remove the need to interact with the Setup Wizard when provisioning a new server.

See our [automated setup documentation](#) for more details.

| Default value | Description |
|--|---|
| <code>setup.displayName</code> | |
| Bitbucket | The display name for the instance. |
| <code>setup.baseUrl</code> | |
| | The base URL for the instance. |
| <code>setup.license</code> | |
| AAABa1evaA4N... | The license. |
| <code>setup.sysadmin.username</code> | |
| admin | The username for the system admin account. |
| <code>setup.sysadmin.password</code> | |
| password | The password for the system admin account. |
| <code>setup.sysadmin.displayName</code> | |
| John Doe | The display name for the system admin account. |
| <code>setup.sysadmin.emailAddress</code> | |
| sysadmin@yourcompany.com | The email address for the system admin account. |

Syntax highlighting

See [Configuring syntax highlighting for file extensions](#) for more information.

| Default value | Description |
|---|---|
| syntax.highlighter.<MIME type>.executables | |
| exe1 , exe2 | Controls the language highlighter used for a given set of hashbang executables.
The '<MIME type>' refers to the MIME type CodeMirror uses. |
| syntax.highlighter.<MIME type>.extensions | |
| ext1 , ext2 | Controls the language highlighter used for a given set of file extensions.
The '<MIME type>' refers to the MIME type CodeMirror uses. |
| syntax.highlighter.application/json.extensions | |
| ipynb | |
| syntax.highlighter.text/x-sh.executables | |
| sh , bash , zsh | |
| syntax.highlighter.text/x-erlang.executables | |
| escript | |
| syntax.highlighter.text/javascript.executables | |
| node | |
| syntax.highlighter.text/x-perl.executables | |
| perl | |
| syntax.highlighter.text/x-python.executables | |
| python | |
| syntax.highlighter.text/x-ruby.executables | |
| ruby | |
| syntax.highlighter.text/x-sh.extensions | |
| makefile , Makefile | |
| syntax.highlighter.text/velocity.extensions | |
| vm | |
| syntax.highlighter.text/x-objectivec.extensions | |
| m | |

Tasks

| Default value | Description |
|------------------------------|---|
| task.max.anchors.per.request | |
| 500 | Maximum number of anchors that can be used when counting or searching tasks |

| | |
|-------------------------------|--|
| task.max.contexts.per.request | |
| 100 | Maximum number of contexts that can be used when counting or searching tasks |
| task.max.tasks.per.request | |
| 500 | Maximum number of tasks that can be retrieved when searching tasks |
| task.query.disjunction.size | |
| 100 | Sets the maximum size of the disjunction clause when querying tasks by anchors or contexts |

Topic

| Default value | Description |
|---------------------------------|--|
| topic.default.message.max.queue | |
| 1024 | Controls the default size of a topic's message queue. Messages that arrive for the topic when the message queue is full will be dropped (and an error logged). This default will only apply when the topic is created (by a plugin) without specifying the message queue size. |
| topic.dispatch.core.threads | |
| 1 | Controls the minimum number of threads to keep alive for dispatching topic messages to topic subscribers |
| topic.dispatch.max.threads | |
| 3 | Controls the maximum number of threads to dispatch topic messages to the topic subscribers |

Webhooks

| Default value | Description |
|--|--|
| plugin.webhooks.http.backoff.interval.exponent | |
| 1.2 | In the event that a webhook is considered unhealthy and is marked for backoff, it will start by backing off at the initial backoff delay. If the webhook continues to fail, it will back off at a rate of $(\text{initial backlog interval}) * (\text{backoff exponent}^{\text{number of failures}})$ up to a configured maximum |
| plugin.webhooks.http.backoff.interval.initial | |
| 10 | The initial delay given to a webhook when it fails more than the configured maximum number of times. While in this backoff state, the specific webhook will be skipped

This value is in seconds |
| plugin.webhooks.http.backoff.interval.max | |
| 7200 | The maximum delay given to a webhook when it continually fails. While in this backoff state, the specific webhook will be skipped

This value is in milliseconds , default 2 hours. |
| plugin.webhooks.http.ip.blacklist | |

| | |
|---|--|
| 169.254.169.254 | <p>This value controls a banned set of IPs that webhooks cannot connect to. Doing so will cause an exception. By default the AWS metadata endpoint is banned to avoid leaking data about the machine that the application is running on</p> <p>This value is a comma separated list of IPv4/6 addresses or CIDR ranges.</p> |
| <code>plugin.webhooks.http.backoff.trigger.count</code> | |
| 5 | <p>The maximum number of failures in a row before a webhook is considered unhealthy and will be skipped. The time skipped will continue to rise as per the delay exponent until the webhook is healthy once again. A single success will clear all instances of failures from the count.</p> |
| <code>plugin.webhooks.signature.algorithm</code> | |
| sha256 | <p>The algorithm to use for signing the outgoing request body, e.g. sha1 or sha256. The resulting signature will be sent in the X-Hub-Signature header for webhooks that have a secret configured.</p> |
| <code>plugin.webhooks.connection.timeout</code> | |
| 20000 | <p>The timeout given for a single webhook request to receive a TCP connection</p> <p>This value is in milliseconds</p> |
| <code>plugin.webhooks.socket.timeout</code> | |
| 20000 | <p>Controls how long requests to external services can continue without producing any data before webhooks gives up.</p> <p>This value is in milliseconds</p> |
| <code>plugin.webhooks.dispatch.queue.size</code> | |
| 250 | <p>The maximum size of the queue used to transform webhook publishing events into HTTP events.</p> |
| <code>plugin.webhooks.dispatch.queue.timeout</code> | |
| 250 | <p>The maximum amount of time allowed to be placed on the dispatch queue. This time is only utilised if the dispatch queue is filled. The application will wait this amount of time to attempt to be placed onto the queue.</p> <p>If unsuccessful, the webhook will be skipped.</p> <p>This value is in milliseconds</p> |
| <code>plugin.webhooks.io.threads</code> | |
| 3 | <p>Number of threads to be used to process HTTP IO.</p> |
| <code>plugin.webhooks.callback.thread.count</code> | |
| 10 | <p>Number of threads used to process callbacks to Bitbucket with results of the HTTP requests.</p> |
| <code>plugin.webhooks.http.connection.max</code> | |
| 200 | <p>Maximum number of concurrent outgoing connections. Further connections will be placed upon the dispatch queue until they are able to be processed.</p> |
| <code>plugin.webhooks.http.connection.host.max</code> | |
| 5 | <p>Maximum number of connections to the same HTTP host.</p> |
| <code>plugin.webhooks.statistics.flush.interval</code> | |

| | |
|---|--|
| 30 | The interval in seconds at which webhooks statistics are written to the database. A longer interval is more efficient, but results in slightly outdated invocation details in the webhooks administration pages. This value should not be 0

This value is in seconds |
| plugin.webhooks.response.http.body.size.max | |
| 16384 | Maximum size in bytes for a webhook response body. Any larger bodies will be truncated at this length before being provided to callbacks and/or persisted for historical tracking.

This value is in bytes |
| plugin.webhooks.dispatch.inflight.max | |
| 500 | Maximum number of webhooks requests able to be either - Currently in progress - waiting for a HTTP connection any further connection requests will be skipped |

Zero downtime backup/disaster recovery

| Default value | Description |
|--|---|
| disaster.recovery | |
| false | When set to true repair jobs are triggered on startup |
| integrity.check.pullRequest.batchSize | |
| 1000 | Maximum number of results returned in each database call used by integrity checking tasks |
| integrity.check.pullRequest.updatedSinceDays | |
| 7 | Controls the date range used to filter merged pull requests for integrity checking. This property defines the number of days to go back since the most recent pull request update |
| integrity.check.repository.batchSize | |
| 1000 | Maximum number of results returned in each database call used by repository check tasks |

Change Bitbucket's context path

There are various reasons why you might need to change the context path for Bitbucket Data Center and Server. Two of those are:

- You are running Bitbucket behind a proxy.
- You have another Atlassian application, or Java web application, available at the same hostname and context path as Bitbucket, and are experiencing login problems (see [Login and session conflicts with multiple Atlassian applications](#)).

Related pages

- [Integrate Bitbucket with Apache HTTP Server](#)
- [Login and session conflicts with multiple Atlassian applications](#)

To change the context path:

1. Navigate to your [home directory](#).
2. Stop Bitbucket. See [Start and stop Bitbucket](#).
3. Open the `shared/bitbucket.properties` file, and add the `server.context-path` property. Set it to reflect the context path that you want Bitbucket to be accessible at. For example, to set the context path to `/bitbucket` you would add:

```
server.context-path=/bitbucket
```

Then save the file.

4. Start Bitbucket. See [Start and stop Bitbucket](#).

Bitbucket is now available at the same host as before, but under the new context path. For example, a server that was at `http://localhost:7990` will now be reachable at `http://localhost:7990/bitbucket`.

5. Once started, go to  > **Server settings**.
6. Append the new context path to your base URL:

```
https://localhost:7990/bitbucket
```

7. Select **Save**.

Important considerations

If you're running Bitbucket behind Apache,

- You need to make sure the host or context path that Bitbucket is exposed on is not also being used by another web application that is listening on a different port.
- If you updated the Bitbucket context path using the steps outlined above, you need to also update your Apache configuration, as described in [Integrating Bitbucket Server with Apache HTTP Server](#).

If you had Application Links set up before changing the context path in Bitbucket, you have to recreate those using the new Bitbucket URL. See [Link Bitbucket with Jira](#).

If you use SSH, the context path does not affect the URL at which SSH operations occur. After changing the context path so that Bitbucket is accessible at `https://localhost:7990/bitbucket`, SSH operations occur without the context path at `ssh://my-bitbucket-hostname:7999`.

Data recovery and backups

This page provides an overview of the backup and restore options available for use with Bitbucket Data Center and Server:

- [Bitbucket backup essentials](#)
- [The importance of being consistent](#)
- [Bitbucket backup strategies](#)
- [Further information](#)

Questions? Check out [FAQ - Data recovery and backup](#).

Bitbucket backup essentials

An effective backup strategy is essential:

- for avoiding data loss in the event of any system breakdown,
- for restoring Bitbucket after any system breakdown,
- as part of the Bitbucket upgrade process.

We highly recommend that you establish a data recovery plan that is aligned with your company's policies. At the very least, you should consider these aspects:

- How frequently should Bitbucket be backed up? We recommend that backups are made at least daily.
- How much downtime is acceptable? When using a backup strategy with any downtime we recommend scheduling backups at a time of day that minimizes impact on users, e.g., out of office hours.
- How long should backups be retained for? We recommend that backups be retained for at least one month.
- Where should the backups be stored? We recommend that backups are stored at an offsite location.
- How quickly and easily can you restore your data in an emergency? We recommend restoring your backups in a staging environment on a regular basis to ensure that your backup strategy works in the event of a *real* emergency scenario.

The importance of being consistent

All backup strategies for Bitbucket need to capture the state of two fundamental data sources:

- The **home directory** on the file system, which contains your repository data, log files, plugins, and so on (see [Set the home directory](#) for more detail).
- The **database**, which contains data about pull requests, comments, users, groups, permissions, and so on.

These two data sources hold the entire state of a Bitbucket instance. To backup the complete state of your instance, you need to take *consistent* snapshots of both using one of the strategies below. If you attempt to restore snapshots containing inconsistencies then there is a risk of corruption or data loss in your repositories and pull requests.

(In addition, if you have a [remote Elasticsearch instance](#) then search indexes are maintained in Elasticsearch's data directory, but you don't have to include this in your backup as it can be completely rebuilt if necessary from the data in your home directory and database.)

Bitbucket backup strategies

Bitbucket provides multiple strategies for taking backups free of inconsistencies, and each are summarized in the table below. Each option has tradeoffs between technical requirements and the amount of downtime involved. Which strategy you choose depends on the scale of your instance, your file system and database technologies, your recovery point objective, and your users' tolerance of downtime when backups are taken.

Related pages

- [Connect Bitbucket to an external database](#)
- [Supported platforms](#)
- [Bitbucket Server Backup Client](#)
- [Bitbucket DIY Backup](#)
- [Bitbucket zero downtime backup](#)
- [Disaster recovery guide for Bitbucket Data Center](#)

| | Zero Downtime Backup | DIY Backup | Backup Client |
|--------------------------------|---|---|---|
| Summary | A technique that eliminates downtime completely using internally consistent database snapshots and block-level filesystem snapshots | A technique that minimizes downtime using incremental copy or vendor-specific snapshot technology | Simple but unsuitable for use in high availability environments.

An external utility which locks a Bitbucket Server instance and backs up its entire home directory and database in a vendor-independent format. |
| Downtime | ✔ Zero at backup time. | ⚠ Low. Only needs to lock Bitbucket briefly to create a consistent snapshot. Downtime can be as low as a few seconds. | ✘ High. Bitbucket is locked for the entire duration of the backup process, which may take several minutes or longer, especially in large organizations. |
| Minimum product version | Bitbucket 4.8+ | Stash 2.12+

Bitbucket 4.0+ | Stash 2.7+

Bitbucket 4.0+ |
| Bitbucket Server | ✔ Supported. Bitbucket tolerates (but does not attempt to resolve) any inconsistencies between the home directory and database after restoring. | ✔ Supported | ✔ Supported |
| Bitbucket Data Center | ✔ Supported. Bitbucket can perform an integrity check at restore time to scan for inconsistencies between the home directory and database, and resolve them. | ✔ Supported | ✘ Not supported, even if you are running on one cluster node. |
| Minimum requirements | <ul style="list-style-type: none"> Requires you to use the snapshot tools of your file system and database vendor. Example scripts are provided. Requires your home directory to be on a file system volume capable of atomic (block level) snapshots, for example, Amazon EBS, LVM, NetApp, XFS, or ZFS. Minimizing the time between database and filesystem snapshots (or using vendor-specific point-in-time recovery) reduces the chances of inconsistencies occurring | Requires you to use the snapshot tools of your file system and database vendor. Example scripts are provided. | No special requirements. "Just works" out of the box. |
| Backup format | Vendor-specific database snapshot and block level file system snapshot of the entire disk volume. | Vendor-specific database dump and file system snapshot. | Database vendor-independent. |

| | | | |
|----------------------|--|--------------------------------------|--|
| Documentation | Bitbucket zero downtime backup | Bitbucket DIY Backup | Bitbucket Server Backup Client |
|----------------------|--|--------------------------------------|--|

Further information

[Bitbucket zero downtime backup](#)

[Bitbucket DIY Backup](#)

[Bitbucket Server Backup Client](#)

Bitbucket Server Backup Client

The backup client is not compatible with Bitbucket Data Center

The Bitbucket Server Backup Client is not compatible with clustered Bitbucket Data Center instances (even if you switch to a single node).

To back up a clustered Bitbucket Data Center instance, you need to use the [Bitbucket Server DIY Backup](#) instead.

This page describes using the Bitbucket Server Backup Client. We highly recommend that you establish a data recovery plan that is aligned with your company's policies.

To start using the Backup Client:

1. [Download](#) the Bitbucket Server Backup Client.
2. Unzip the client into a directory on the server where your Bitbucket Server instance resides.

Questions? Check out [FAQ - Data recovery and backup](#).

How it works

The Backup Client implements a common and universal way to backup a Bitbucket Server instance, and does the following:

1. Locks access to the Bitbucket Server application, the repositories managed by Bitbucket Server and the Bitbucket Server database for the entire duration of the back up. This state is called 'maintenance mode'.
2. Checks that all Git and database operations have completed.
3. Performs an application-specific backup of the [Set the home directory](#) and the Bitbucket Server database. The backup is generic and does not depend on the server or database configuration.
4. Stores the backup as a single tar file on the local filesystem in the specified location.
5. Unlocks Bitbucket Server from maintenance mode.

You will get an error message if you try to access the Bitbucket Server web interface, or use the Bitbucket Server hosting services, when Bitbucket Server is in maintenance mode.

The client supports Windows and Linux platforms, and Bitbucket Server versions 4.0 and higher, but does not provide ways to integrate with your organizations IT policies or processes.

As an indication of the unavailability time that can be expected when using the Bitbucket Server Backup Client, in our testing and internal use we have seen downtimes for Bitbucket Server of 78 minutes with repositories totalling 6 GB in size. For comparison, using [Bitbucket Server DIY Backup](#) for the same repositories typically results in a downtime of less than a minute.

What is backed up

The Backup Client backs up all the following data:

- the database Bitbucket Server is connected to (either the internal or external DB)
- managed Git repositories
- the Bitbucket Server audit logs
- installed plugins and their data

On this page:

- [How it works](#)
- [What is backed up](#)
- [Backing up Bitbucket Server using the client](#)
- [Canceling the client backup](#)
- [Restoring Bitbucket Server to use the existing DB](#)
- [Restoring Bitbucket Server to use a newly created DB](#)
- [Debug logging](#)
- [Related pages](#)

The backup does NOT include the following files and directories:

- `export/`*
- `log/`*
- `shared/data/db`* (HSQL data in the DB is backed up, but the files on disk are not)
- `tmp`
- the `plugins` directory (except for the `installed-plugins` directory)

Backing up Bitbucket Server using the client

Before you begin backing up Bitbucket Server using the client, you should first ensure you are using [a release of the Bitbucket Server Backup Client that is compatible with your Bitbucket Server instance](#).

The Backup Client is a command line tool and not a plugin (no application UI) so it must be run externally from somewhere with access to the Bitbucket Server home directory. Usually, you will run the Backup Client directly on the Bitbucket Server server. Run the client with the following commands:

```
cd <path/to/backup-config.properties file>
java -jar <path/to/bitbucket-backup-client.jar>
```

Configuration options are kept in the `backup-config.properties` file, an example of which is included with the client. This file is automatically read from the directory you were in when the `bitbucket-backup-client` is run. The properties are fully documented in the `backup-config.properties` file, but include:

`bitbucket.home`

Defines the location of the home directory of the Bitbucket Server instance you wish to back up or restore to. **REQUIRED**

If omitted here it will be taken from the `BITBUCKET_HOME` environment variable or the Java system property of the same name if supplied to the Backup and Restore Client on the command line. As a required value, backup and restore will fail if it is not supplied through one of these mechanisms.

`bitbucket.user`

Defines the username of the Bitbucket Server user with administrative privileges you wish to perform the backup. **REQUIRED**

If omitted here it will be taken from the Java system property of the same name if supplied to the Backup Client on the command line. As a required value, backup will fail if it is not supplied through one of these mechanisms.

`bitbucket.password`

Defines the password of the Bitbucket Server user with administrative privileges you wish to perform the backup. **REQUIRED**

If omitted here it will be taken from the Java system property of the same name if supplied to the Backup Client on the command line. As a required value, backup will fail if it is not supplied through one of these mechanisms.

`bitbucket.baseUrl`

Defines base URL of the Bitbucket Server instance you wish to back up. **REQUIRED**

E.g. `http://localhost:7990/bitbucket` or `http://bitbucketserver/`.

If omitted here it will be taken from the Java system property of the same name if supplied on the command line to the Backup Client. As a required value, backup will fail if it is not supplied through one of these mechanisms.

`backup.home`

Defines where the Backup Client will store its own files, such as backup archives.

If not specified, these files are stored beneath the working directory for the Backup Client. Backup files will be stored in a `backup` subdirectory and logs will be stored in a `logs` subdirectory.

Note that on Windows, you must use two backslashes between paths. E.g. `C:\\path\\to\\folder` or instead use the forward slash e.g. `C:/path/to/folder`.

The location defined by `backup.home` must not be located in the directory defined by `bitbucket.home`. If that is the case, the Backup Client will fail.

Alternatively, these properties can be given on the command-line, when they need to be prefixed with `"-D"`, and be placed before the `"-jar"` parameter. For example:

```
java -Dbitbucket.password="admin" -Dbitbucket.user="admin" -Dbitbucket.baseUrl="http://localhost:7990" -Dbitbucket.home=path/to/bitbucket/home -Dbackup.home=path/to/backup-home -jar bitbucket-backup-client.jar
```

Canceling the client backup

You can cancel the running client backup operation if necessary.

To cancel the backup:

1. Copy the cancel token echoed by the client in the terminal (or the Command Prompt on Windows):

```
$ ./bitbucket.diy-backup.sh
[http://localhost:7990/bitbucket] INFO: Prepared backup of DB bitbucket in /bitbucket-backup
/bitbucket-db/
building file list ... done.
sent 4.17M bytes received 484 bytes 2.78M bytes/sec
total size is 121.12M speedup is 29.06
[http://localhost:7990/bitbucket] INFO: Prepared backup of /bitbucket-home to /bitbucket-backup
/bitbucket-home/
[http://localhost:7990/bitbucket] INFO: locked with '7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a'
[http://localhost:7990/bitbucket] INFO: backup started with
'82c73f89e790b27fef3032e81c7071388ae4e371'
[http://localhost:7990/bitbucket] INFO: Waiting for DRAINED state..... done
[http://localhost:7990/bitbucket] INFO: db state 'DRAINED'
[http://localhost:7990/bitbucket] INFO: scm state 'DRAINED'
```

2. Go to the Bitbucket Server interface in your browser. Bitbucket Server will display this screen:

Performing Backup

Backup is currently underway and shouldn't take long.

Backing up Bitbucket home

[Cancel backup](#)

3. Click **Cancel backup**, and enter the cancel token:

Performing Backup

Backup is currently underway and shouldn't take long.

Backing up Bitbucket home

Authentication token

82c73f89e790b27fef3032e81c7071388ae4e371

To cancel, please enter an authentication token. This can be found in the Bitbucket logs

Cancel backup

4. Click **Cancel backup**.

Restoring Bitbucket Server to use the existing DB

This section applies if you are restoring Bitbucket Server to fix a corrupted installation, but are able to use the existing DB that Bitbucket Server was backed up from. This scenario assumes that Bitbucket Server is to be restored to the same server from which Bitbucket Server was originally backed up.

The Restore Client must be run on the machine that Bitbucket Server should be restored to. To ensure restores do not accidentally delete existing data, the Restore Client will only restore into an empty home directory and an empty database.

The Restore Client will use the JDBC connection configuration contained in the backup you are restoring from.

Follow this process:

1. Stop your Bitbucket Server instance.
2. Delete the content of the current home directory, so that it is empty.
3. Drop the existing tables in your database so it is empty. The database still needs to exist with the same user/password, and it should have the configuration described in the 'Create the Bitbucket Server database' section of the relevant page here:
 - [MySQL](#)
 - [Oracle](#)
 - [PostgreSQL](#)
 - [SQL Server](#)
4. Run the Restore Client using the following command (replacing 'path/to/bitbucket/home' and '/path/to/original/backup/file' with your own values):

```
java -Dbitbucket.home="path/to/bitbucket/home" -jar bitbucket-restore-client.jar /path/to/original/backup/file
```

5. If you are restoring Bitbucket Server to fix a corrupted installation, now follow Steps 4 to 6 of the [Bitbucket Server upgrade guide](#). Note that you should use the same version of Bitbucket Server that was used to back up Bitbucket Server.

Restoring Bitbucket Server to use a newly created DB

This section applies if you intend to perform a restore into a newly created DB. This scenario assumes the restore is done to a server different from the one from which Bitbucket Server was originally backed up.

The restore process is database agnostic, meaning the database you are restoring your backup to could be of a different configuration or type from the originally backed up database.

When restoring Bitbucket Server, the Restore Client must be run on the machine that Bitbucket Server should be restored to. To ensure restores do not accidentally delete existing data, the Restore Client will only restore into an empty home directory and an empty database.

Follow this process:

1. Create a new empty home directory using the user account that will be used to run Bitbucket Server.
2. Create the database. It should have the configuration described in the 'Create the Bitbucket Server database' section of the relevant page here:
 - [MySQL](#)
 - [Oracle](#)
 - [PostgreSQL](#)
 - [SQL Server](#)
3. Run the Restore Client. [See the following section](#) for details.
4. Install Bitbucket server on the new server by following the instructions on [Run the Bitbucket installer](#). Point the installation to an empty directory and install it as a service in case you used to have that in your previous server.
5. Make sure your new installation is up and running - refer to [Start and stop Bitbucket](#). At this stage you should see Bitbucket server guiding you for a configuration from scratch (i.e. it will ask you for license details, admin user etc.) and that's expected.
6. Stop the newly installed instance.
7. Delete the contents from the *newly installed* `BITBUCKET_HOME`.
8. Copy the contents of the *restored* `BITBUCKET_HOME` into the *newly installed* `BITBUCKET_HOME`.
9. Start the *newly installed* Bitbucket server.
10. At this stage, the following will happen:
 - a. If you installed a binary at the same release as it was on your previous server, everything will be like before and you will need to perform an upgrade in a separate step. This is the recommended approach.
 - b. If you installed a binary at a release above the one it was installed on your previous server, an upgrade will be performed at this stage.

Running the Restore Client from the command line

You can run the Restore Client from the command line. In this scenario, as you will have created a new database, you need to specify the JDBC connection parameters that should be used.

The Restore Client uses the JDBC connection configuration specified in the `jdbc.driver`, `jdbc.url`, `jdbc.user` and `jdbc.password` parameters used in the command to run the Restore Client (see below). Once the database backup is successfully restored, the client will write the specified parameters to the `bitbucket-config.properties` file in the newly restored Bitbucket Server home directory, allowing the new instance to connect to the restored database once the steps outlined below are followed.

Before performing a restore to a MySQL database, note that the JDBC driver for MySQL is *not* bundled with it (due to licensing restrictions). You need to download and install the driver yourself.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).
2. Expand the downloaded zip/tar.gz file.
3. Copy the `mysql-connector-java-5.1.XX-bin.jar` file from the extracted directory to your `<path/to/backup/client>/jdbc` directory.

In this example, you can follow the restore into a newly created PostgreSQL database:

```
java -Djdbc.override=true -Djdbc.driver=org.postgresql.Driver -Djdbc.url=jdbc:postgresql://HOSTNAME:PORT/DATABASE -Djdbc.user=bitbucketuser -Djdbc.password=password -Dbitbucket.home="/path/to/bitbucket/home" -jar /path/to/bitbucket-restore-client.jar /path/to/original/backup/file
```

Alternatively, you can configure these parameters on the `backup-config.properties` file make sure the file exists in the current working directory. A sample file is shipped with the client. The properties are fully documented in the `backup-config.properties` file and more details are described below:

`bitbucket.home`

The full path to a directory that the Restore Client will populate with the Bitbucket Server home data. This directory must be empty. On Windows, you must use two backslashes (`\\`) or a single forward slash (`/`) to separate paths.

`jdbc.override`

By default, the Restore Client will restore into the same database that was backed up. If `jdbc.override` is set to `true`, the Restore Client will restore into the database specified by the `jdbc` properties in the table below. The database must be empty.

`jdbc.driver`

The driver class that Bitbucket Server should use to log in to the new database. See examples below.

`jdbc.url`

The connection details for the new database, formatted as a JDBC URL. See examples below.

`jdbc.user`

The username that Bitbucket Server should use to log in to the new database.

`jdbc.password`

The password that Bitbucket Server should use to log in to the new database.

`bitbucket.home.restore.whitelist`

Defines a comma-separated list of files and directories that may be present in the target home *and* shared directories when restoring a backup. Files other than those matching these entries will result in a failure.

By default files `.snapshot`, `lost+found`, `.DS_Store` are white listed

Example use of JDBC properties

Example `jdbc.driver` and `jdbc.url` properties are shown below:

| Database | <code>jdbc.driver</code> | <code>jdbc.url</code> |
|------------|---|---|
| MySQL | <code>com.mysql.jdbc.Driver</code> | <code>jdbc:mysql://HOSTNAME:PORT/DATABASE?autoReconnect=true&characterEncoding=utf8&useUnicode=true&sessionVariables=storage_engine%3DInnoDB</code> |
| Oracle | <code>oracle.jdbc.driver.OracleDriver</code> | <code>jdbc:oracle:thin:@//HOSTNAME:PORT/SERVICE</code> |
| PostgreSQL | <code>org.postgresql.Driver</code> | <code>jdbc:postgresql://HOSTNAME:PORT/DATABASE</code> |
| SQL Server | <code>com.microsoft.sqlserver.jdbc.SQLServerDriver</code> | <code>jdbc:sqlserver://HOSTNAME:PORT;databaseName=DATABASE;</code> |

Debug logging

Debug logging can be turned on by adding the following to the `logback.xml` file in the working directory from where you're running the backup client. Create this file if it does not already exist.

logback.xml

```
<included><logger name="com.atlassian.bitbucket" level="DEBUG"/></included>
```

Related pages

- [Data recovery and backups](#)
- [Bitbucket DIY Backup](#)
- [Scheduling tasks on Linux](#)
- [Scheduling tasks on Windows](#)
- [Debug logging for the Bitbucket Server Backup Client](#)
- [Bitbucket Server - FAQ - Data recovery and backups](#)
- [How to establish staging server environments for Bitbucket Server](#)

Bitbucket DIY Backup

i This article explains use of the Bitbucket Server DIY Backup scripts for use with Bitbucket Server 4.x+. If you are running an earlier version of this product, formerly known as Stash, please see [Using Stash \(3.11\) DIY Backup](#).

The Bitbucket Server DIY Backup is an alternative strategy to using the [Bitbucket Server Backup Client](#). It allows you to:

- significantly reduce the downtime needed to create a consistent backup,
- use the vendor-specific database backup tool appropriate to your back end database, for example:
 - `pg_dump` if your back end database is PostgreSQL, or
 - `sqlcmd` with an appropriate command for differential backup, if your back end database is MS SQL Server,
- use the optimal file system backup tool for your Bitbucket Server home directory, for example:
 - an LVM snapshot logical volume if your Bitbucket Server home directory uses LVM,
 - a SAN-based backup if your Bitbucket Server home directory uses a Storage Area Network, or
 - `rsync`, if available.
- take backups of [Bitbucket Data Center](#) instances without having to bring nodes down manually.

On this page:

- [How it works](#)
- [What is backed up](#)
- [DIY Backups using Bash scripts](#)
- [Restoring a DIY Backup](#)
- [Canceling the backup](#)
- [Advanced - writing your own DIY Backup using the REST APIs](#)

Related pages:

- [Data recovery and backups](#)
- [Bitbucket Server Backup Client](#)
- [Scheduling tasks on Linux](#)
- [Scheduling tasks on Windows](#)

Download the worked example scripts from Bitbucket:

[Download](#)

The key to reducing downtime is the use of optimal, vendor-specific database and file system backup tools. Such tools are generally able to take snapshots (though sometimes in a vendor-specific format) in much less time than the generic, vendor-neutral format used by the Bitbucket Server Backup Client.

Bitbucket Server DIY Backup does require you to write some code in a language of your choice to perform the required backup steps, using the REST API available for Bitbucket Server 4.0.

DIY Backup supports Windows and Linux platforms, and Bitbucket Server version 4.0 and higher. DIY Backup supports both Bitbucket Server and Bitbucket Data Center instances equally - any DIY Backup solution that works on one should work on the other without modification.

For information about other backup strategies for Bitbucket Server, see [Data recovery and backups](#). That page also discusses the tight coupling between the Bitbucket Server file system on disk and the database that the application uses.

i Please note that the examples on this page are provided as guidance for developing a DIY Backup solution. As such, the third-party tools described are for example only - you will need to choose the tools that are appropriate to your own specific installation of Bitbucket Server.

Consult the vendor documentation for the third-party tools you choose - unfortunately, Atlassian can not provide support for those tools.

This page:

- Describes a complete DIY Backup solution for a PostgreSQL database and local filesystem, using `bash` shell scripts.
- Provides background information about how the Bitbucket Server REST API can be used for DIY Backups.

You can use this solution directly if your Bitbucket Server instance has the same or similar configuration, or use this as a starting point to develop your own DIY Backup solution tailored to your hardware configuration.

How it works

When you use DIY Backup instead of the Bitbucket Server Backup Client, you have complete control over the backup steps, and can implement any custom processes you like in the language of your choice. For example, you can use your database's incremental or fast snapshot tools and/or your file server's specific tools as part of a DIY Backup.

The DIY Backup works in a similar way to the [Bitbucket Server Backup Client](#) and does the following:

1. Prepares the Bitbucket Server instance for backup. This happens before Bitbucket Server is locked, so we want to do as much processing as possible here in order to minimize downtime later. For example, we can take an initial snapshot using incremental database and filesystem utilities. These do not have to be 100% consistent as Bitbucket Server is still running and modifying the database and filesystem. But taking the initial snapshot now may reduce the amount of work done later (while the application is locked), especially if the amount of data modified between backups is large. The steps include:
 - Taking an initial backup of the database (if it supports progressive/differential backups).
 - Doing an initial `rsync` of the home folder to the backup folder.
2. Initiates the backup, which will:
 - Lock the Bitbucket Server instance.
 - Drain and latch the connections to the database and the filesystem.
 - Wait for the drain/latch step to complete.
3. Once the instance is ready for backup we can start with the actual DIY Backup. This will include steps to:
 - Make a fully consistent backup of the database, using `pg_dump`.
 - Make a fully consistent backup of the filesystem, using `rsync`.
4. Notify the Bitbucket Server instance once the backup process finishes and unlock it.
5. Archive all files created during the backup into one big archive.

A user will get an error message if they try to access the web interface, or use the hosting services, when the application is in maintenance mode.

As an indication of the unavailability time that can be expected, in Atlassian's internal use we have seen downtimes for Bitbucket Server of 78 minutes with repositories totalling 6 GB in size when using the Bitbucket Server Backup Client. For comparison, using Bitbucket Server DIY Backup for the same repositories typically results in a downtime of less than a minute.

What is backed up

The Bitbucket Server DIY Backup backs up all the same data as the Bitbucket Server Backup Client:

- the database the instance is connected to (either the internal or external DB)
- managed Git repositories
- the Bitbucket Server logs
- installed plugins and their data

DIY Backups using Bash scripts

This section presents a complete DIY Backup solution that uses the following tools:

- `bash` - for scripting
- `jq` - an open source command line JSON processor for parsing the REST responses from Bitbucket Server
- `pg_dump` (or `sqlcmd`) - for backing up a PostgreSQL database
- `rsync` - for backing up the filesystem

- `tar` - for making a backup archive

This approach (with small modifications) can be used for running DIY Backups on:

- Linux and Unix
- macOS
- Windows with cygwin (note that cygwin Git is *not* supported by Bitbucket Server).

Bash scripts

You can download the example scripts from [Bitbucket](#) or simply clone the repository.

Running the Bash script

Once you have downloaded the Bash scripts, you need to create one file:

- `bitbucket.diy-backup.vars.sh` (you can copy [bitbucket.diy-backup.vars.sh.example](#) to start)

For example, here's how you might configure `bitbucket.diy-backup.vars.sh`:

- your Bitbucket Server server is called `bitbucket.example.com`, uses port 7990, and has its home directory in `/bitbucket-home`
- you want to generate the backup in `/bitbucket-backup`, and store your `.tar.gz` backups in `/bitbucket-backup-archives`,
- you have a System Administrator in Bitbucket with the username "admin" and password "admin", and you run Bitbucket (and the backup scripts) as the OS user "atbitbucket"

bitbucket.diy-backup.vars.sh

```
#!/bin/bash

CURL_OPTIONS="-L -s -f"
INSTANCE_NAME=bitbucket

BITBUCKET_URL=http://bitbucket.example.com:7990
BITBUCKET_HOME=/bitbucket-home/
BITBUCKET_UID=atlbitbucket
BITBUCKET_GID=atlbitbucket

BACKUP_HOME_TYPE=rsync
BACKUP_DATABASE_TYPE=postgresql
BACKUP_ARCHIVE_TYPE=tar

BITBUCKET_BACKUP_USER=admin
BITBUCKET_BACKUP_PASS=admin
BITBUCKET_BACKUP_EXCLUDE_REPOS=()

BITBUCKET_DB=bitbucket
POSTGRES_HOST=localhost
POSTGRES_USERNAME=dbuser
export PGPASSWORD=dbpass
POSTGRES_PORT=5432

# Make use of PostgreSQL 9.3+ options if available
psql_version="$(psql --version | awk '{print $3}')"
psql_majorminor="$(printf "%d%03d" $(echo "${psql_version}" | tr "." "\n" | head -n 2))"
if [[ ${psql_majorminor} -ge 9003 ]]; then
PG_PARALLEL="-j 5"
PG_SNAPSHOT_OPT="--no-synchronized-snapshots"
fi

BITBUCKET_BACKUP_ROOT=/bitbucket-backup
BITBUCKET_BACKUP_DB=${BITBUCKET_BACKUP_ROOT}/bitbucket-db/
BITBUCKET_BACKUP_HOME=${BITBUCKET_BACKUP_ROOT}/bitbucket-home/

BITBUCKET_BACKUP_ARCHIVE_ROOT=/bitbucket-backup-archives

# Used by the scripts for verbose logging. If not true only errors will be shown.
BITBUCKET_VERBOSE_BACKUP=TRUE

HIPCHAT_URL=https://api.hipchat.com
HIPCHAT_ROOM=
HIPCHAT_TOKEN=

KEEP_BACKUPS=0
```

The supplied `bitbucket.diy-backup.vars.sh` is written to use PostgreSQL, rsync, and tar by default. But if you want to use different tools, you can also customize the top section of this file:

Example usage:

```
# Strategy for backing up the Bitbucket home directory:
# - amazon-ebs - Amazon EBS snapshots of the volume containing the home
directory
# - rsync - "rsync" of the home directory contents to a temporary location.
NOTE: This can NOT be used
# with BACKUP_ZERO_DOWNTIME=true.
BACKUP_HOME_TYPE=rsync

# Strategy for backing up the database:
# - amazon-rds - Amazon RDS snapshots
# - mysql - MySQL using "mysqldump" to backup and "mysql" to restore
# - postgresql - PostgreSQL using "pg_dump" to backup and "pg_restore" to
restore
# - postgresql93-fslevel - PostgreSQL 9.3 with data directory located in the
file system volume as home directory (so
# that it will be included implicitly in the home volume snapshot)
BACKUP_DATABASE_TYPE=postgresql

# Strategy for backing up Elasticsearch:
# - <leave blank> - No separate snapshot and restore of Elasticsearch state
(default).
# - s3 - Amazon S3 bucket - requires the Elasticsearch Cloud plugin to be
installed.
# - fs - Shared filesystem - requires all data and master nodes to mount a
shared file system to the same mount point.
BACKUP_ELASTICSEARCH_TYPE=
```

You also need to create two directories for DIY Backup to work:

1. `${BITBUCKET_BACKUP_ROOT}` is a working directory (`/bitbucket-backup` in our example) where copies of Bitbucket Server's home directory and database dump are built during the DIY Backup process.
2. `${BITBUCKET_BACKUP_ARCHIVE_ROOT}` is the directory (`/bitbucket-backup-archives` in our example) where the final backup archives are saved.

The Bash scripts may be run on any host, provided it has:

- read/write access to the above `${BITBUCKET_BACKUP_ROOT}` and `${BITBUCKET_BACKUP_ARCHIVE_ROOT}` directories,
- read access to the `${BITBUCKET_HOME}` directory,
- read access to the database, and
- network access to run `curl` commands on the Bitbucket Server server.

This is true regardless of whether you have an instance of Bitbucket Server or Bitbucket Server Data Center. It doesn't matter whether the filesystem access is direct or over NFS, or whether the network access is direct to a Bitbucket Server node or to a load balancer / reverse proxy.

Once your `bitbucket.diy-backup.vars.sh` is correctly configured, run the backup in a terminal window:

```
$ ./bitbucket.diy-backup.sh
```

The first time you run the backup, `rsync` will do most of the work since the `/bitbucket-backup` working directory is initially empty. This is normal. Fortunately, this script performs one `rsync` before locking Bitbucket Server, followed by a second `rsync` while Bitbucket Server is locked to minimize downtime.

On second and subsequent backup runs, `/bitbucket-backup` is already populated so the backup process should be faster. The output you can expect to see looks something like this:

```
$ ./bitbucket.diy-backup.sh
[http://localhost:7990/bitbucket] INFO: Prepared backup of DB bitbucket in /bitbucket-backup/bitbucket-
db/
building file list ... done.
sent 4.17M bytes received 484 bytes 2.78M bytes/sec
total size is 121.12M speedup is 29.06
[http://localhost:7990/bitbucket] INFO: Prepared backup of /bitbucket-home to /bitbucket-backup
/bitbucket-home/
[http://localhost:7990/bitbucket] INFO: locked with '7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a'
[http://localhost:7990/bitbucket] INFO: backup started with '82c73f89e790b27fef3032e81c7071388ae4e371'
[http://localhost:7990/bitbucket] INFO: Waiting for DRAINED state..... done
[http://localhost:7990/bitbucket] INFO: db state 'DRAINED'
[http://localhost:7990/bitbucket] INFO: scm state 'DRAINED'
[http://localhost:7990/bitbucket] INFO: Performed backup of DB bitbucket in /bitbucket-backup/bitbucket-
db/
[http://localhost:7990/bitbucket] INFO: Backup progress updated to 50
building file list ... done.
sent 4.87M bytes received 484 bytes 3.25M bytes/sec
total size is 121.82M speedup is 24.99
[http://localhost:7990/bitbucket] INFO: Performed backup of /bitbucket-home to /bitbucket-backup
/bitbucket-home/
[http://localhost:7990/bitbucket] INFO: Backup progress updated to 100
[http://localhost:7990/bitbucket] INFO: Bitbucket instance unlocked
[http://localhost:7990/bitbucket] INFO: Archiving /bitbucket-backup into /bitbucket-backup-archives
/bitbucket-20150917-082818-498.tar.gz
[http://localhost:7990/bitbucket] INFO: Archived /bitbucket-backup into /bitbucket-backup-archives
/bitbucket-20150917-082818-498.tar.gz
```

Restoring a DIY Backup

When restoring Bitbucket Server, you must run the `bitbucket.diy-restore.sh` script on the machine that Bitbucket Servers should be restored to. In order to ensure accidental restores do not delete existing data, you should never restore into an existing home directory.

The new database should be configured following the instructions in [Connect Bitbucket to an external database](#) and its sub-page that corresponds to your database type.

To see the available backups in your `/${BITBUCKET_BACKUP_ARCHIVE_ROOT}` directory, just type:

```
$ ./bitbucket.diy-restore.sh
```

You should see output similar to this:

```
$ ./bitbucket.diy-restore.sh
Usage: ./bitbucket.diy-restore.sh <backup-file-name>.tar.gz
Available backups:
bitbucket-20150917-082818-498.tar.gz bitbucket-20150918-083745-001.tar.gz
```

To restore a backup, run `bitbucket.diy-restore.sh` with the file name as the argument:

```
$ ./bitbucket.diy-restore.sh bitbucket-20150917-082818-498.tar.gz
```

You should see output like this:

```
$ ./bitbucket.diy-restore.sh bitbucket-20150917-082818-498.tar.gz
[http://localhost:7990/bitbucket] INFO: Extracted /bitbucket-backup-archives/bitbucket-20150917-082818-498.tar.gz into /tmp/bitbucket.diy-restore.dQsbzU
[http://localhost:7990/bitbucket] INFO: Performed restore of /tmp/bitbucket.diy-restore.dQsbzU/bitbucket-db to DB bitbucket2
[http://localhost:7990/bitbucket] INFO: Performed restore of /tmp/bitbucket.diy-restore.dQsbzU/bitbucket-home to /bitbucket-home2
```

Canceling the backup

You can cancel the running backup operation if necessary.

To cancel the backup:

1. Copy the cancel token echoed in the terminal (or the Command Prompt on Windows). Look for the line "backup started with token"

```
$ ./bitbucket.diy-backup.sh
[http://localhost:7990/bitbucket] INFO: Prepared backup of DB bitbucket in /bitbucket-backup/bitbucket-db/
building file list ... done.
sent 4.17M bytes received 484 bytes 2.78M bytes/sec
total size is 121.12M speedup is 29.06
[http://localhost:7990/bitbucket] INFO: Prepared backup of /bitbucket-home to /bitbucket-backup/bitbucket-home/
[http://localhost:7990/bitbucket] INFO: locked with '7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a'
[http://localhost:7990/bitbucket] INFO: backup started with '82c73f89e790b27fef3032e81c7071388ae4e371'
[http://localhost:7990/bitbucket] INFO: Waiting for DRAINED state..... done
[http://localhost:7990/bitbucket] INFO: db state 'DRAINED'
[http://localhost:7990/bitbucket] INFO: scm state 'DRAINED'
```

E.g. use "82c73f89e790b27fef3032e81c7071388ae4e371"

2. Go to the Bitbucket Server interface in your browser. Bitbucket Server will display this screen:

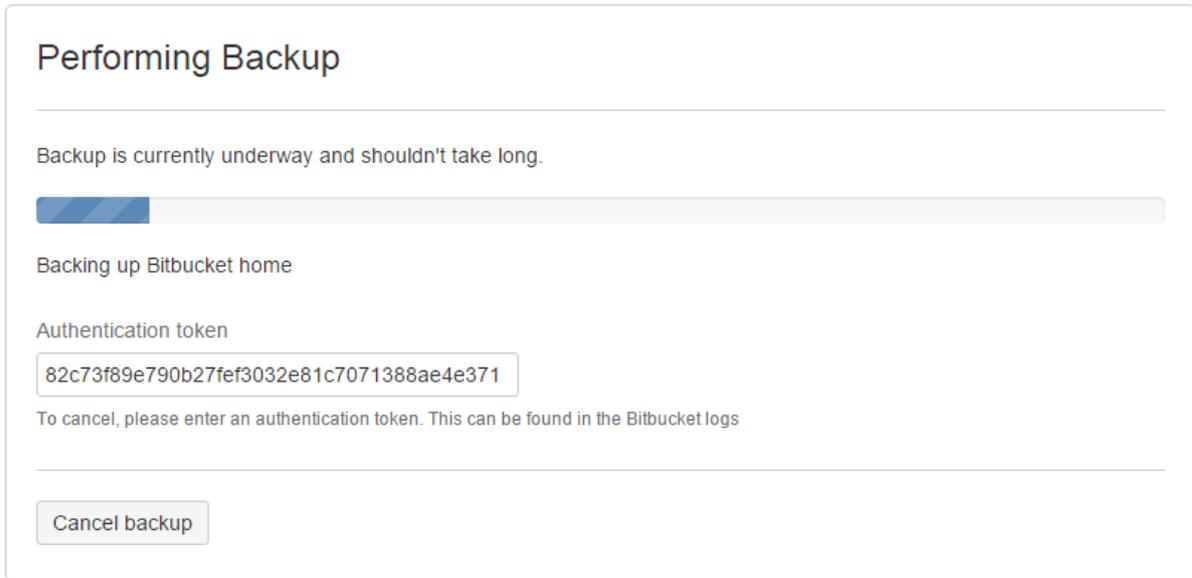
Performing Backup

Backup is currently underway and shouldn't take long.

Backing up Bitbucket home

[Cancel backup](#)

3. Click **Cancel backup**, and enter the cancel token:



4. Click **Cancel backup**.

Note that Bitbucket Server will still be locked in maintenance mode. Repeat these steps using the "locked with" token (e.g. "7187ae1824ce1ede38a8e7de4bccf58d9a8e1a7a") to exit maintenance mode as well, and unlock Bitbucket Server.

Advanced - writing your own DIY Backup using the REST APIs

i This section is optional and provides background information about how you might use the Bitbucket Server REST APIs if you need to rewrite the DIY Backup scripts described above in your preferred language or to customize them heavily.

Note that this discussion shows `curl` commands in Bash, however you can use any language.

The following steps are involved:

Preparation

Before you lock Bitbucket Server you can perform any preparation you like. It makes sense to perform as much processing as possible before you lock the application, to minimize downtime later. For example, you could perform an `rsync`:

```
rsync -avh --delete --delete-excluded --exclude=/caches/ --exclude=/data/db.* --exclude=/export/ --
exclude=/log/ --exclude=/plugins/.*/ --exclude=/tmp --exclude=/.lock ${BITBUCKET_HOME}
${BITBUCKET_BACKUP_HOME}
```

Lock the Bitbucket Server instance

The next step in making a backup of a Bitbucket Server instance is to lock the instance for maintenance. This can be done using a POST request to the `/mvc/maintenance/lock` REST point (where `BITBUCKET_URL` points to the Bitbucket Server instance, `BITBUCKET_BACKUP_USER` is a Bitbucket Server user with backup permissions, and `BITBUCKET_BACKUP_PASS` is this user's password).

| REQUEST | RESPONSE |
|---|---|
| <pre>curl -s \ -u \${BITBUCKET_BACKUP_USER}:\${BITBUCKET_BACKUP_PASS} \</pre> | <pre>{ "unlockToken": " 0476adeb6cde3a41aa0cc19fb394779191f5d306", "owner": { "displayName": "admin", "name": "admin" } }</pre> |

```
-X POST \
-H "Content-type: application/json" \
"${BITBUCKET_URL}/mvc/maintenance/lock"
```

```
}
}
```

If successful, the Bitbucket Serverinstance will respond with a 202 and will return a response JSON similar to the one above. The `unlockToken` should be used in all subsequent requests where `BITBUCKET_LOCK_TOKEN` is required. This token can also be used to manually unlock the instance.

Start the backup process

Next, all connections to both the database and the filesystem must be drained and latched. Your code must handle backing up of both the filesystem and the database.

At this point, you should make a POST request to `/mvc/admin/backups`. Notice that the `curl` call includes the `?external=true` parameter:

| REQUEST |
|---|
| <pre>curl -s \ -u \${BITBUCKET_BACKUP_USER}:\${BITBUCKET_BACKUP_PASS} \ -X POST \ -H "X-Atlassian-Maintenance-Token: \${BITBUCKET_LOCK_TOKEN}" \ -H "Accept: application/json" \ -H "Content-type: application/json" \ "\${BITBUCKET_URL}/mvc/admin/backups? external=true"</pre> |

| RESPONSE |
|---|
| <pre>{ "id": " d2e15c3c2da282b0990e8efb30b4bffbcbf09e04", "progress": { "message": "Closing connections to the current database", "percentage": 5 }, "state": "RUNNING", "type": "BACKUP", "cancelToken": " d2e15c3c2da282b0990e8efb30b4bffbcbf09e04" }</pre> |

If successful the instance will respond with 202 and a response JSON similar to the one above will be returned. The `cancelToken` can be used to manually cancel the back up process.

Wait for the instance to complete preparation.

Part of the back up process includes draining and latching the connections to the database and the filesystem. Before continuing with the back up we have to wait for the instance to report that this has been done. To get details on the current status we make a GET request to the `/mvc/maintenance` REST point.

| REQUEST |
|---|
| <pre>curl -s \ -u \${BITBUCKET_BACKUP_USER}:\${BITBUCKET_BACKUP_PASS} \ -X GET \ -H "X-Atlassian-Maintenance-Token: \${BITBUCKET_LOCK_TOKEN}" \ -H "Accept: application/json" \ -H "Content-type: application/json" \ "\${BITBUCKET_URL}/mvc/maintenance"</pre> |

| RESPONSE |
|---|
| <pre>{ "task": { "id": " 0bb6b2ed52a6a12322e515e88c5d515d6b6fa95e", "progress": { "message": "Backing up Bitbucket home", "percentage": 10 }, "state": "RUNNING", "type": "BACKUP" }, "db-state": "DRAINED", "scm-state": "DRAINED" }</pre> |

This causes the Bitbucket Serverinstance to report its current state. We have to wait for both `db-state` and `scm-state` to have a status of `DRAINED` before continuing with the backup.

Perform the actual backup

At this point we are ready to create the actual backup of the filesystem. For example, you could use `rsync` again:

```
rsync -avh --delete --delete-excluded --exclude=/caches/ --exclude=/data/db.* --exclude=/export/ --
exclude=/log/ --exclude=/plugins/.*/ --exclude=/tmp --exclude=/.lock ${BITBUCKET_HOME}
${BITBUCKET_BACKUP_HOME}
```

The `rsync` options shown here are for example only, but indicate how you can include only the required files in the backup process and exclude others. Consult the documentation for `rsync`, or the tool of your choice, for a more detailed description.

When creating the database backup you could use your vendor-specific database backup tool, for example `pg_dump` if you use PostgreSQL:

```
pg_dump -Fd ${BITBUCKET_DB} -j 5 --no-synchronized-snapshots -f ${BITBUCKET_BACKUP_DB}
```

While performing these operations, good practice is to update the instance with progress on the backup so that it's visible in the UI. This can be done by issuing a `POST` request to `/mvc/admin/backups/progress/client` with the token and the percentage completed as parameters:

REQUEST

```
curl -s \
  -u ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \
  "${BITBUCKET_URL}/mvc/admin/backups/progress/client?token=${BITBUCKET_LOCK_TOKEN}
&percentage=${BITBUCKET_BACKUP_PERCENTAGE}"
```

Bitbucket Server will respond to this request with an empty 202 if everything is OK.

When displaying progress to users, Bitbucket Server divides the 100 percent progress into 90 percent user DIY Backup, and 10 percent application preparation. This means, for example, if your script sends `percentage=0`, Bitbucket Server may display up to 10 percent progress for its own share of the backup work.

Inform the Bitbucket Server instance that the backup is complete

Once we've finished the backup process we must report to the Bitbucket Server instance that progress has reached 100 percent. This is done using a similar request to the progress request. We issue a `POST` request to `/mvc/admin/backups/progress/client` with the token and 100 as the percentage:

REQUEST

```
curl -s \
  -u ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \
  -X POST \
  -H "Accept: application/json" \
  -H "Content-type: application/json" \
  "${BITBUCKET_URL}/mvc/admin/backups/progress/client?token=${BITBUCKET_LOCK_TOKEN}
&percentage=100"
```

Bitbucket Server will respond with an empty 202 if everything is OK. The backup process is considered completed once the percentage is 100. This will unlatch the database and the filesystem for this Bitbucket Server instance.

Unlock the Bitbucket Server instance

The final step we need to do in the back up process is to unlock the instance. This is done with a DELETE request to the `/mvc/maintenance/lock` REST point:

REQUEST

```
curl -s \  
  -u ${BITBUCKET_BACKUP_USER}:${BITBUCKET_BACKUP_PASS} \  
  -X DELETE \  
  -H "Accept: application/json" \  
  -H "Content-type: application/json" \  
  "${BITBUCKET_URL}/mvc/maintenance/lock?token=${BITBUCKET_LOCK_TOKEN}"
```

The Bitbucket Server instance will respond to this request with an empty 202 if everything is OK, and will unlock access.

Bitbucket zero downtime backup

This page describes how to back up Bitbucket Data Center or Server without downtime, and restoring the backups correctly.

About zero downtime backup

Zero downtime backup is a technique introduced in Bitbucket 4.8 which backs up a Bitbucket instance without requiring it be locked for maintenance. It requires:

1. your shared home directory to be on a file system volume capable of atomic (block level) snapshots, and
2. your database to be capable of either taking atomic snapshots or restoring a snapshot at the same point in time as the home directory snapshot was taken.

Use of these technologies allows you to take backups as often as you need (e.g., hourly), without inconveniencing your users and build agents with frequent downtimes.

Prerequisites

Block level file system snapshots

Your shared home directory must be on a file system volume capable of atomic (block level) snapshots, for example, [Amazon EBS](#), [LVM](#), [NetApp](#), [XFS](#), or [ZFS](#). These technologies are becoming increasingly common in modern operating systems and storage solutions. If your shared home directory volume pre-dates these technologies, then you must first move your shared home directory onto one that supports block level snapshots before using zero downtime backup. You also need to script the steps to snapshot the volume in the backup process. The [atlassian-bitbucket-diy-backup](#) script does not include fully worked examples for every vendor technology. If you are unable to create such a snapshot, please consider [another backup strategy](#).

Block level snapshots ensure your repository data will have full internal consistency when restored, even if taken without a maintenance lock under heavy load. Which block level snapshot technology you use depends on your choice of infrastructure for your shared home directory volume:

- **File server which provides block level snapshots** (for example, [NetApp Snapshots](#)): Refer to your vendor's documentation on how to script the snapshot and restore process. Atlassian does not provide examples or support for the use of these vendor tools.
- **Linux file system: LVM, XFS, and ZFS** are all capable of taking block level snapshots. Refer to the documentation with your Linux distribution on how to script the snapshot and restore process. Atlassian does not provide examples or support for their use.
- **Amazon Web Services (AWS) Elastic Block Store (EBS) volume:** You can use [Amazon EBS Snapshots](#) to take block level snapshots. The EBS volume may be formatted with any file system type, but note that Linux typically requires the filesystem to be "frozen" with `fsfreeze` while the snapshot is taken, for its own internal consistency. The [atlassian-bitbucket-diy-backup](#) script has a worked example of using EBS Snapshots.

Whichever file system snapshot technology you ultimately choose, you will need to refer to your vendor's documentation to script the snapshot and restore process. The only fully worked examples currently included in the [atlassian-bitbucket-diy-backup](#) script are for Amazon EBS and ZFS.

Database snapshot technology

Your database must be capable of restoring a snapshot close to the same point in time as the home directory snapshot. The easiest way to do this is by taking database snapshots close to the home directory backup time. Alternatively, some databases support a vendor-specific "point-in-time recovery" feature at restore time. All database vendors supported by Bitbucket provide tools for taking fast snapshots and point-in-time recovery.

Which technology you choose is a tradeoff between how much work you need to do at backup and restore time.

On this page

- [About zero downtime backup](#)
- [Prerequisites](#)
- [Configure the example DIY Backup script for zero downtime backup](#)
- [Back up your instance](#)
- [Restore from a backup](#)

- **Vendor dump and restore utilities:** This option needs you to dump your database to a file at backup time, in parallel with the home directory snapshot. This can generally achieve a database snapshot that is within a few seconds of your home directory snapshot. The [atlassian-bitbucket-diy-backup](#) script includes a worked example of using PostgreSQL's `pg_dump` and `pg_restore` with zero downtime backup.
- **Point-in-time recovery:** Alternatively if you enable the point-in-time recovery feature of your database vendor, then you don't have to dump the database explicitly at backup time. Instead you can recover a snapshot of the database state at the same time as the home directory snapshot, at restore time. This saves you from having to do database dumps during backup, at the cost of a slightly longer restore time. The [atlassian-bitbucket-diy-backup](#) script does *not* include a worked example of using point-in-time recovery, you need to refer to your vendor's documentation on how to script this process.
- **Amazon Web Services (AWS) Relational Database Service (RDS) instance:** You can use the [RDS Backup and Restore](#) feature to take snapshots at backup time. The [atlassian-bitbucket-diy-backup](#) script includes a worked example of using RDS snapshots with zero downtime backup.
- **Block level snapshot of the database's data volume:** If your database's data directory is located on a file system volume that is capable of block level snapshots, then most databases support backup and restore at the file system level. The [atlassian-bitbucket-diy-backup](#) script includes a worked example of using block level snapshots where the database is on the same volume as your shared home directory.

Whichever database backup technology you choose, you will need to refer to your vendor's documentation to script the snapshot and restore process.

Bitbucket Server 4.8 or later

You must be running Bitbucket 4.8 or higher to use zero downtime backup. See the [Bitbucket Server upgrade guide](#).

Configure the example DIY Backup script for zero downtime backup

With the above prerequisites, you can create backups with any method that performs home directory and database snapshots simultaneously. Atlassian provides an example [atlassian-bitbucket-diy-backup](#) script which can automate the process, and can be used as a starting point for you to configure and customize your own backup procedures.

Step 1: Get the script

Clone or pull the latest version of the sample [atlassian-bitbucket-diy-backup](#) script, for example:

```
git clone https://bitbucket.org/atlassianlabs/atlassian-bitbucket-diy-backup.git
cd atlassian-bitbucket-diy-backup
```

Step 2: Configure the script

Create the file `bitbucket.diy-backup.vars.sh` by copying the appropriate `bitbucket.diy-backup.vars.sh.example`, and edit it to match your environment. For example, to use Amazon EBS snapshots of the volume containing your shared home directory and Amazon RDS snapshots of your database, you might configure it as follows:

```
BACKUP_DATABASE_TYPE=amazon-rds
BACKUP_HOME_TYPE=amazon-ebs
BACKUP_ZERO_DOWNTIME=true
BITBUCKET_URL=https://your-bitbucket-url
```

If your shared home directory is not on Amazon EBS, then you must define a new `BACKUP_HOME_TYPE` for your chosen file system technology. Give it a name (e.g., `myhome`) that will identify the snapshot script to run in the next step.

```
BACKUP_HOME_TYPE=myhome
```

 `BACKUP_HOME_TYPE=rsync` is *not* a valid option to use with zero downtime backup.

Refer to [Bitbucket DIY Backup](#) and [Using Bitbucket DIY Backup in AWS](#) for more information on the configurable options in the DIY Backup example script.

Step 3: (If necessary) Script the snapshot process

 If you have chosen `BACKUP_HOME_TYPE` and `BACKUP_DATABASE_TYPE` values that both already have worked examples included in the [atlassian-bitbucket-diy-backup](#) repository (e.g., `ebs-home` and `rsync`), then you can just configure the appropriate variables in `bitbucket.diy-backup.vars.sh` to match your environment, and skip this step.

If your shared home directory is not one of the worked examples provided in [atlassian-bitbucket-diy-backup](#), then you must script the process of taking and restoring snapshots of your home directory volume.

Create a script called `bitbucket.diy-backup.myhome.sh` (where `myhome` is the value of `BACKUP_HOME_TYPE` that you set in the previous step), defining the following BASH functions:

```
#!/bin/bash

function bitbucket_prepare_home {
# you can optionally do any backup-time validation you need to do here
}

function bitbucket_backup_home {
# this is where you freeze, snapshot, and unfreeze your home directory
volume
}

function bitbucket_restore_home {
# this is where you restore a snapshot of your home directory volume
}
```

Similarly if you wish to use the point-in-time recovery feature of your database with zero downtime backup (or some other functionality that does not have a fully worked example provided in [atlassian-bitbucket-diy-backup](#)), then you must script the process.

Create or modify a script `bitbucket.diy-backup.mydb.sh` (where `mydb` is the value of `BACKUP_DATABASE_TYPE` that you set in the previous step), defining the following BASH functions:

```
#!/bin/bash

function bitbucket_prepare_db {
# you can optionally do any backup-time validation you need to do here
}

function bitbucket_backup_db {
# this is where you can snapshot your database, if necessary
}

function bitbucket_prepare_db_restore {
# this is where you can do any restore-time validation, if necessary
}

function bitbucket_restore_db {
# this is where you restore a snapshot of your database
}
```

Back up your instance

Once your `bitbucket.diy-backup.vars.sh` is correctly configured, SSH to the instance (or the appropriate node in a Data Center instance, typically the file server) and run the backup script from the `atlassian-bitbucket-diy-backup` directory.

```
./bitbucket.diy-backup.sh
```

This script can also be run on a regular schedule (e.g., hourly), from `cron`.

Restore from a backup

Step 1: Stop Bitbucket

The restore process only works while Bitbucket is stopped. Stop the Bitbucket services (on all nodes if your instance is a clustered Data Center instance).

```
sudo service atlbitbucket stop
sudo service atlbitbucket_search stop
```

See [Start and stop Bitbucket](#) for more information.

Step 2: Run the restore script

To restore, SSH to the instance (or the appropriate node in a Data Center instance, typically the file server) and run the restore script from the `atlassian-bitbucket-diy-backup` directory.

```
./bitbucket.diy-restore.sh
```

Step 3: Start Bitbucket

 Data Center customers at this point may enable integrity checking to scan for potential inconsistencies between the database and home directory, and resolve them if necessary so that your pull request and repository state are completely consistent with each other. See [Running integrity checks in Bitbucket Data Center](#) for further information.

Start the Bitbucket services (on all nodes if your instance is a clustered Data Center instance).

```
sudo service atlbitbucket start
sudo service atlbitbucket_search start
```

See [Start and stop Bitbucket](#) for more information.

Running integrity checks in Bitbucket Data Center

This feature is only for customers with an active [Bitbucket Data Center resources](#) license.

On this page

- [About integrity checks](#)
- [Running integrity checks](#)
- [What integrity checks look for](#)
- [Feedback from the integrity check process](#)

This page describes how to run integrity checks in a Bitbucket Data Center instance, for example, after restoring from backups.

About integrity checks

Bitbucket Data Center allows you to perform an integrity check that scans for potential inconsistencies between the database and home directory, and resolves them if necessary so that your pull request and repository state are completely consistent with each other. You can perform an integrity check in any situation where you suspect your database and home directory may contain inconsistencies, for example, after restoring from a backup.

Running integrity checks

To run integrity checks, add this line to your `{BITBUCKET_HOME}/shared/bitbucket.properties`:

```
disaster.recovery=true
```

Then [start Bitbucket](#). You can start Bitbucket on all cluster nodes if you wish.

After starting, Bitbucket will run integrity checks on one cluster node only. Integrity checks may take several minutes to complete, but run in the background. While integrity checks are running users can still log in, interact with the system, and perform hosting operations on repositories.

Disabling integrity checks

After you have restored Bitbucket, integrity checks have run, and you have resumed normal operation, turn off the `disaster.recovery` property in your `bitbucket.properties` file so integrity checks won't run unnecessarily the next time your instance is restarted.

```
disaster.recovery=false
```

What integrity checks look for

Integrity checks (which run when `disaster.recovery` is set to `true`) scan your instance for inconsistencies between the database and home directory that can occur when snapshots of your database and file system were taken at slightly different times.

Why integrity checks are needed

When Bitbucket is running it is constantly modifying its database and home directory, but under almost all circumstances the two data sources will be consistent with each other (even if the UI is slow to catch up).

However, when database and home directory snapshots are taken independently, and updates that affect the database and home directory happen between the two snapshots, integrity checks may find inconsistencies. An example of when this could happen is if a pull request is merged between snapshots. When snapshots of your database and home directory are taken close enough together the chance of inconsistencies arising are small.

What integrity checks cannot detect

Inconsistencies in Git: It's important to note that integrity checks *only detect inconsistencies between your database and home directory*, not internal inconsistencies within the repositories themselves.

If you suspect repositories in your Bitbucket instance have become corrupted in some other way, you may need to manually run `git fsck` to diagnose and restore individual repositories. See [Recommended action plan if a repository becomes corrupted on a Bitbucket Server](#) for more information, or contact Atlassian Support.

Information not in your database/home directory when the backup was taken: The Integrity Checker can detect mismatches between the state of a repository or pull request in the database and file system and make adjustments to restore integrity, but it cannot reconstruct information not in your database or home directory when the backup was taken.

This means if your backups are taken hourly, when restoring from your latest backup your users may lose up to an hour worth of work. In addition, if your latest database and file system snapshots were taken a minute apart, changes to pull requests made in this time may be lost and cannot be reconstructed by the Integrity Checker.

The best way to ensure inconsistencies don't occur in your backups is to ensure your file system and database snapshots are taken as close together in time as possible, or use the "point-in-time recovery" feature of your database vendor to restore the database to when the file system snapshot was taken.

Feedback from the integrity check process

A warning  or information  banner will be displayed to system administrator to indicate the state/outcome of the integrity check process. This banner can be in one of four states.

- Integrity checks are running, no inconsistencies have been found 
- Integrity checks are running, at least one inconsistency has been found 
- Integrity checks complete, no inconsistencies found 
- Integrity checks complete, inconsistencies found 

When an integrity check finds an inconsistency

If an integrity check finds an inconsistency between the database and home directory, it will automatically perform adjustments to restore integrity between the two. For example, if during your backup process someone merges a pull request *after* a database snapshot, but *before* the file system snapshot, and that backup is restored. In this case the integrity checks will find the pull request is in an inconsistent state and adjust the pull request in the database to match the actual state on disk. When this happens the adjustment is shown in the Activity tab of the pull request, and is attributed to the Integrity Checker service user. Any activity to your pull requests performed by the Integrity Checker also generates the usual notifications.

Activity

 What do you want to say?

 **Integrity Checker** remotely merged the pull request from `feature/BSERVDEV-13144-ssh-jmx` to `master` 1 hour ago

 **Ben Humphreys** marked the pull request as APPROVED 6 days ago

An example of an adjustment made by the Integrity Checker on the Activity tab

The Integrity Checker will write a message to the application log whenever it encounters an inconsistency. Filtering the `atlassian-bitbucket.log` for `DefaultIntegrityCheckReporter` will return all relevant log entries.

You should read the Integrity Checker log entries to understand why the inconsistency occurred. Inconsistency error messages will read:

```
The repository PROJ/repo[1] exists but the directory /repositories/1 is missing. To restore integrity,
an empty repository directory was created.
```

or

```
PROJ/repo[1]: Pull request #1 is marked merged but the merge commit could not be found on the target ref. Trying to restore integrity by reopening
```

or

```
PROJ/repo[1]: Pull request #1 could not be reopened, declining instead. (Reason: REASON)
```

Where *REASON* can be one of

- *an open pull request with the same to and from refs already exists*
- *unexpected missing commit*
- *fromRef could not be resolved*

If you find many inconsistencies from a larger range of time, this may indicate that your database and home directory snapshots were taken further apart in time than you intended. To ensure these inconsistencies don't arise, **test your disaster recovery plan regularly**, and ensure that your backup and restore processes capture database and home directory snapshots as close together in time as possible.

Pull requests updated by rescoping.

The standard Bitbucket server rescoping process will normalize a large number of pull request inconsistencies in these cases the integrity check reporter will *not* log a message to the application log but rather send a notification to all pull request collaborators.

Example Scenarios

Here are a few example scenarios that the 'Integrity Checker' can detect and resolve:

| Integrity Check | Filesystem state | Database state | Result |
|--------------------------------------|------------------------------|------------------------------------|--|
| Recently merged pull requests | Pull request is merged | Pull request is marked as 'open' | 'Integrity Checker' will mark pull request as remotely merged.

<i>Note: only the merge activity will be attributed to the 'Integrity Checker' user, the merge commit will remain authored by the original merger.</i> |
| | Pull request is not merged | Pull request is marked as 'merged' | 'Integrity Checker' will re-open the pull request. |
| Repository creation | Repository #2 does not exist | Repository #2 exists | An empty repository will be created on the filesystem. |

Disable HTTP(S) access to Git repositories

Administrators can disable HTTP(S) access to Git repositories in Bitbucket Data Center and Server. This removes the ability to push to or clone Git repositories over HTTP(S).

To disable HTTP(S) access:

1. Go to  > **Server settings**.
2. Uncheck **HTTP(S) enabled**.
3. Select **Save**.

Related pages:

- [Enable SSH access to Git repositories](#)

Smart Mirroring



A Data Center license is required to use this feature. Get an [evaluation license](#) to try it out, or [purchase a license](#) now.

Smart Mirroring can greatly improve Git clone speeds for distributed teams working with large repositories. Large repositories that take hours to clone from a Bitbucket instance over the Internet from the other side of the world can take minutes when cloned from a local mirror on a fast network.

As your team grows and becomes more distributed, so will the need to clear up congestion resulting from an increasing number of continuous integration (CI) builds. Mirror farms can handle this issue by taking mirrors and clustering them into farms to reduce the time spent waiting for those build results.

If you're ready to start mirroring your repositories, you can jump straight to the [Set up a mirror](#) or [Set up and configure a mirror farm](#) page and follow the steps there, or read on to learn more about the benefits of using a mirror. If you have mirrors already configured, you might be searching for instructions on how to [Clone a mirror repository](#).

On this page:

- [About Smart Mirroring](#)
 - [How it works](#)
 - [Self-healing](#)
- [Ready to get started setting up a mirror?](#)

Related pages:

- [Set up a mirror](#)
- [Set up and configure a mirror farm](#)

Also:

Check out the latest on-demand webinar, [How to support your geo-distributed teams with Atlassian Data Center](#).

In this webinar, learn how Atlassian Data Center provides performance at scale for your distributed teams and get access to:

- Best practices on instance setup and configuration for distributed team work
- A deep dive on some of the latest geo-performance features like CDN and Mirror Farms
- Tips for scaling teamwork globally

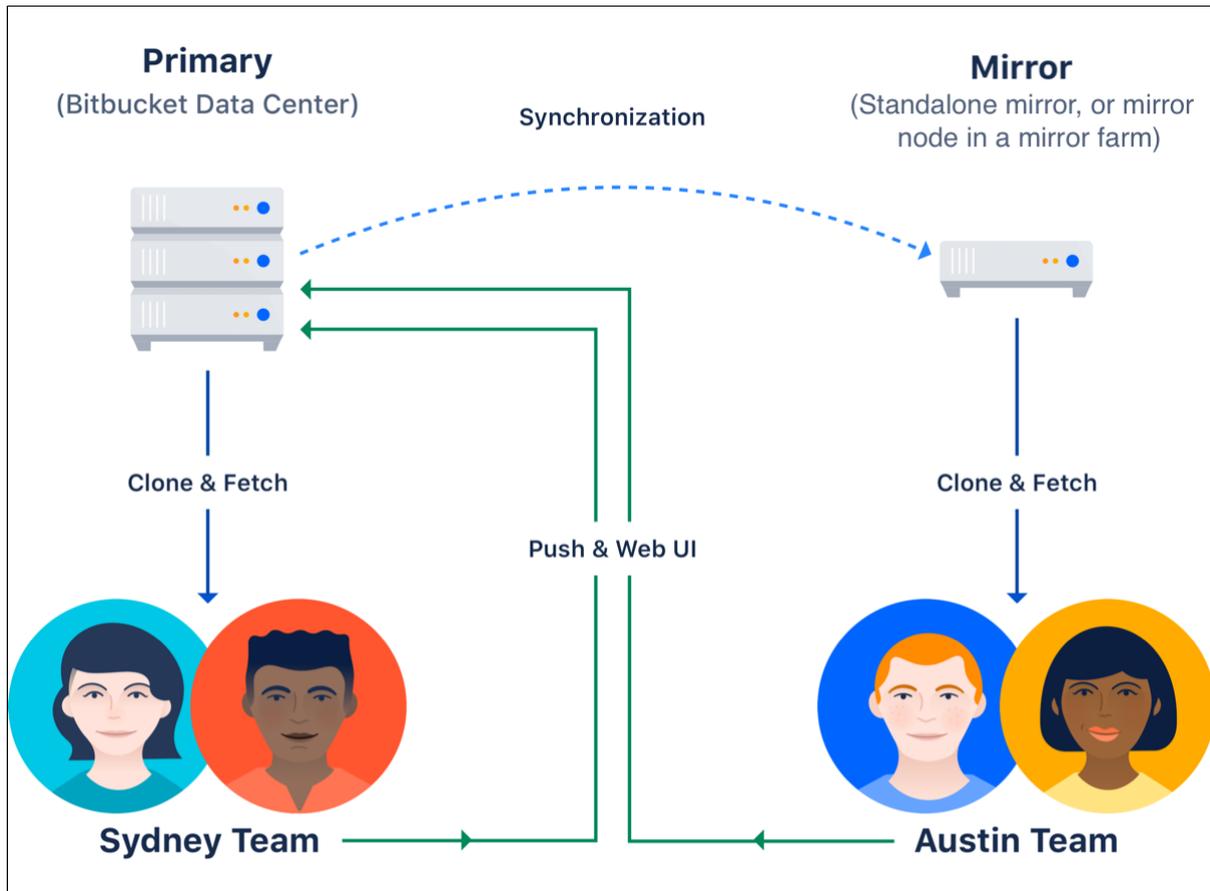
About Smart Mirroring

Many software development teams using Git have large repositories. This is a result of either storing lots of historical information, using monolithic repositories, or storing large binary files (sometimes all three). Companies with distributed software development teams often have little control over the network performance available to them between sites. In combination, this leads to a loss of development time when developers have to wait long periods to clone a large repository from across the world.

Smart mirroring prevents this lost development time by allowing you to set up live mirrors that host copies of repositories in remote locations. These mirrors automatically keep any repository hosted on them in sync with the primary Bitbucket Data Center instance. Users in those remote locations may clone and fetch repositories from the mirror and get identical content, only faster. You can choose to mirror a selection of projects, or mirror *all* repositories in *all* projects from the primary instance.

How it works

Mirrors run the same application as a full Bitbucket Server instance, but they are configured to mirror a primary Bitbucket Data Center instance, where the primary copy of all your repositories is hosted.



When a user clones or fetches from a mirror, the mirror automatically delegates the authentication and authorization of their credentials back to the primary server. No extra user management is required on standalone mirrors or mirror nodes in a farm. All the users, groups, and permissions of the primary Bitbucket instance (whether provided by the built-in user directory and permission system or by your own user directories and/or custom extensions) are always replicated *exactly* on all mirrors.

Self-healing

Self-healing is one of the main design principles for smart mirroring. Mirrors have the ability to detect and recover from a number of error scenarios, while All operations retry with exponential backoff. Smart mirroring also includes an anti-entropy system (the farm vet) which verifies the consistency of a mirror against the primary every 3 minutes. For more information and help with monitoring the health of your mirror farm, see [Monitoring your mirror farm](#).

Ready to get started setting up a mirror?

Be sure to read [Set up a mirror](#) or [Set up and configure a mirror farm](#) for detailed instructions on installing a mirror or mirror farm.

Don't have Bitbucket Data Center yet? [Purchase a Data Center license](#), or [get an evaluation license to try it out](#).

Set up a mirror

 A Data Center license is required to use this feature. Get an [evaluation license](#) to try it out, or [purchase a license](#) now.

Smart Mirroring can drastically improve Git clone speeds for distributed teams working with large repositories. This page describes how to install a mirror and troubleshoot any issues you might encounter.

For an overview of the benefits of using mirrors and how mirror farms can help your organization, see [Smart Mirroring](#). These instructions assume you already have a fully licensed Bitbucket Data Center instance up and running.

On this page

- [Before you start](#)
- [1. Install Bitbucket on the mirror](#)
- [2. Set up HTTPS on the mirror](#)
- [3. Start the mirror](#)
- [4. Set up the mirror](#)
- [5. Approve a mirror request](#)
- [6. Decide which projects to mirror](#)
- [Troubleshooting](#)

Before you start

You must also meet the following requirements:

- **Your primary Bitbucket instance must be a fully licensed Bitbucket Data Center instance**- You do not have to run your Bitbucket Data Center instance as a multi-node cluster to use smart mirroring, but you must have an up-to-date Data Center license.
- **The primary instance and all mirror(s) must have HTTPS with a valid (i.e., signed by a Certificate Authority anchored to the root and not expired) SSL certificate**- This is a strict requirement of smart mirroring on both the primary instance and all mirror(s), and cannot be bypassed. The mirror setup wizard will not proceed if either the mirror or the primary instance does not have a valid SSL certificate.
- **The primary Bitbucket instance must have SSH enabled**- Mirrors keep their repositories synchronized with the primary instance over SSH and cannot use HTTP or HTTPS for this. See [Enable SSH access to Git repositories](#) for instructions on enabling SSH access on your primary instance.
- **The platform the mirrors are running on must meet the same minimum requirements for Bitbucket Data Center**- Check the [Supported platforms](#) for detailed requirements, including those for Java and Git, that apply to each mirror.
- **The platform the mirrors are running on must have an OpenSSH client installed**- The mirror will need an OpenSSH client installed to fetch changes from the primary Bitbucket Server.

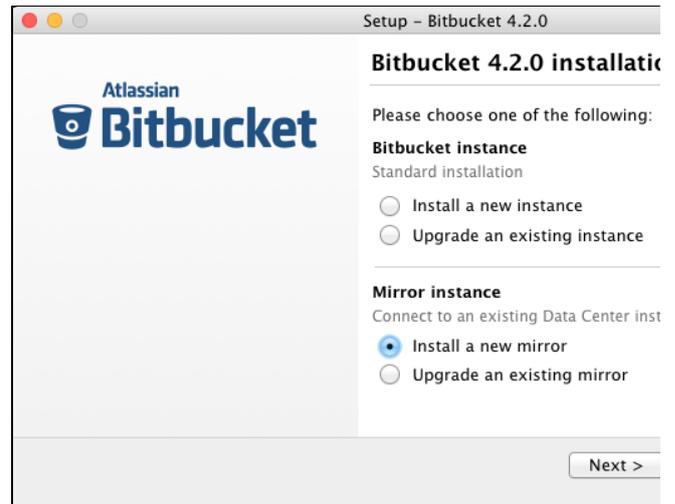
 If you are using Bitbucket version 7.9 or higher, then you must use OpenSSH version 7.2 or higher to avoid sync failures for repositories.

- **The platform the mirrors are running on should not be under-provisioned**- Just as with your primary Bitbucket Data Center instance, your mirrors need to be provisioned with enough CPU, memory, and I/O resources to handle their peak workload. See [Scaling Bitbucket Server](#) for more information.
- **Running your mirror behind a reverse proxy server and terminating SSL at the proxy is highly recommended** - See [Proxy and secure Bitbucket](#).

1. Install Bitbucket on the mirror

The easy way to install a mirror is to download and run the Bitbucket installer and select the **Install a new mirror** option.

i This is the same installer that you use for standard Bitbucket Server and Data Center instances. The mirroring functionality is included in the same distribution.



1. Go to [Download Bitbucket Server](#) and download the latest version of the Bitbucket installer to the server where the mirror will run.
2. Run the installer, and be sure to select **Install a new mirror**.
3. Complete the rest of the installation wizard.

Installing from an archive file

Alternatively, If you prefer not to use the Bitbucket installer, you can:

1. [Install Bitbucket Server from an archive file](#) but **do not start it**.
2. Add the following line to your `${BITBUCKET_HOME}/shared/bitbucket.properties`:

```
application.mode=mirror
```

2. Set up HTTPS on the mirror

You *must* configure your mirror to use HTTPS with a valid SSL certificate, and make secure access mandatory.

SSL certificates are issued by a trusted third party Certificate Authority (CA), such as [VeriSign](#), [DigiCert](#) or [Thawte](#), which provide such services on a commercial basis. Atlassian does not provide such services or support their use.

Once you have an SSL certificate, you can configure your mirror to use it by following one of the options in [Securing a reverse proxy using HTTPS](#).

You can choose whether to install your SSL certificate in a reverse proxy server or directly into Tomcat, either approach will work. But for performance, running your mirror behind a reverse proxy server such as nginx or HAproxy and terminating SSL at the proxy is highly recommended. See [Proxy and secure Bitbucket](#) for more information.

! Both your mirror *and* your primary (upstream) instance must be configured for HTTPS. The mirror setup wizard will not proceed if either the mirror or the primary instance does not have a valid SSL certificate.

3. Start the mirror

Start your mirror.

```
sudo service atlbitbucket start
```

See [Start and stop Bitbucket](#) for more information on starting and stopping mirrors with the `service` command.

4. Set up the mirror

Once your mirror has started, you can go to the setup wizard by navigating to `https://<mirror-full-name>` in your browser, where `<mirror-full-name>` is your mirror's fully qualified name in the DNS. This name will be pre-filled as the **Mirror base URL** and cannot be changed after you click **Submit configuration**, so make sure your primary Bitbucket instance is able to reach the mirror with this name.

Configure mirror settings

Mirror name*
Users see this name when cloning a repository

Mirror base URL*
URL where users will access this mirror

Primary server URL*
The Bitbucket instance you want to mirror

Descriptions of the fields:

| | |
|---------------------------|---|
| Mirror name | The human-readable name of the instance. Users will see this name when selecting a mirror to clone from.
Choose a name that your users will recognize and understand which name is the closest and fastest one for them. |
| Mirror base URL | The URL where the mirror can be accessed. |
| Primary server URL | The URL of the primary Bitbucket Data Center instance. |

After configuring your details, click **Submit configuration**.

 The setup wizard fields cannot be changed after you click **Submit configuration**, so fill them in carefully. If you do make a mistake and need to change a mirror's setup after it has been submitted, you need to stop the mirror, delete its home directory completely, and go back to step 2, [Set up SSL on the mirror](#).

Advanced: Automated setup

As an alternative to the setup wizard, you can automate the setup process without needing to navigate to the mirror in a browser. You can supply the <Mirror name>, <Mirror base URL>, and <Primary server URL> to your new mirror by adding the following properties to your `{BITBUCKET_HOME}/shared/bitbucket.propertiesfile`. These properties can only be applied on the initial start of a new mirror, not after the mirror has already been set up.

```
setup.displayName=<Mirror name>
setup.baseUrl=<Mirror base URL>
plugin.mirroring.upstream.url=<Primary server URL>
plugin.mirroring.upstream.type=server
```

This can be useful if you deploy new mirrors using an automated process such as Puppet. See [Automated setup for Bitbucket](#) for more information.

Once you have successfully set up your mirror, it will guide you to log into the primary instance as an administrator to approve it.

5. Approve a mirror request

Once a mirror has been installed and configured, a request is sent to the primary Bitbucket Data Center instance.



An approved mirror will have access to all projects and repositories in the primary Bitbucket Data Center instance. Smart Mirroring has been designed to require secure communication throughout and restrict all functionality to the appropriate privilege level (e.g., system administrator) to ensure the security of all your sensitive information. It is still *your* responsibility to set up mirrors with the same stringent security practices as your primary Bitbucket Data Center instance. See the FAQ [How secure is Smart Mirroring?](#) for more information.

To approve a mirror request

1. In your primary Bitbucket Data Center instance, go to **Admin > Mirrors**. The authorization request will appear.
2. Click **Authorize** to approve the mirror request and start syncing the projects and repositories of the primary Bitbucket Data Center instance.

6. Decide which projects to mirror

Once a mirror instance is approved you need to decide which projects to mirror. Go to **Admin > Mirrors** and type in the name of a project in the search box. Do this for each project you want to mirror. You can also choose to mirror all projects.



If you choose to mirror all projects, you will not be able to select individual projects. This can't be undone. If you decide you no longer want to mirror all projects, you'll need to completely remove and then reinstall the mirror.

Troubleshooting

This section contains guidance on troubleshooting problems with installing a mirror. For further reference, read the [Bitbucket Data Center FAQ](#), which covers the most frequently asked questions about Bitbucket Data Center.

No valid HTTPS configuration

- Your primary Bitbucket Data Center instance and the mirror server must be configured for HTTPS access in order to use a mirror instance. See [Proxy and secure Bitbucket](#) for more information.
- The value for the `Primary server URL` field must include the "https://" prefix.

Synchronizing projects stays stuck on "Retrieving project information..."

- Ensure [SSH is enabled](#) and usable on the primary instance. See [Enable SSH access to Git repositories](#).
- Ensure your SSH base URL is set correctly in Server settings on the primary instance. See [Enable SSH access to Git repositories#SSHbaseURL](#).
- If appropriate, see [Setting up SSH port forwarding](#).

Incorrectly set Mirror base URL

- The property `Mirror base URL` can only be set once. If configured incorrectly you will need to delete your mirror and set it up again.

"No response was received from the URL you entered" when adding an application link to Bitbucket

Currently, when Bamboo connects to Bitbucket Server's instance, it uses the Base URL to perform any Git operations. To make the URL available also for mirrored instances, you must use the Generic Git for mirrored repositories.

Set up and configure a mirror farm

 A Data Center license is required to use this feature. Get an [evaluation license](#) to try it out, or [purchase a license](#) now.

Mirror farms allows you to set up a cluster of mirrors to cover the load in a local region. This page walks you through how to install and configure a mirror farm. Included are requirements you should meet before starting as well as the information you'll need for your load balancer configuration.

For an overview of the benefits of using mirrors and how mirror farms can help your organization, see [Smart Mirroring](#). These instructions assume you already have a fully licensed Bitbucket Data Center instance up and running.

On this page

- [Before you start](#)
- [1. Set up the load balancer for your mirror farm](#)
- [2. Install Bitbucket on the mirror node](#)
- [3. Start the mirror](#)
- [4. Approve the mirror farm request](#)
- [5. Decide which projects to mirror](#)

Before you start

We recommend you use real-time synchronization on all mirrors to keep clocks in sync. Also, while mirror farms can be in different locations, they must be able to talk to their peers quickly.

You must also meet the following requirements:

- **Your primary Bitbucket instance *must* be a fully licensed Bitbucket Data Center instance-** You do not have to run your Bitbucket Data Center instance as a multi-node cluster to use smart mirroring, but you must have an up-to-date Data Center license.
- **Your mirror farm has to be accessible over HTTPS with a valid certificate.** This configuration uses the same properties as the primary Bitbucket instance. See [Proxy and secure Bitbucket](#).
- **You should have one dedicated load balancer per mirror farm.** Running your mirrors behind a load balancer is necessary and terminating SSL at the proxy is highly recommended. See [Proxy and secure Bitbucket](#).
- **You need to have set up a host name for your mirror farm (this relates to the load balancer and the SSL certificate).** This is so users can connect to the mirror farm. Configurations for your mirror farm and load balancer must be in sync, having the same hostname.
- **The primary Bitbucket Data Center instance *must* have SSH enabled.**
- **You must have a minimum version of Git 2.11.1 installed.** See [Supported platforms](#) for more details.
- **The platform the mirrors are running on *must* meet the same minimum requirements for Bitbucket Data Center-** Check the [Supported platforms](#) for detailed requirements, including those for Java and Git, that apply to each mirror.
- **The platform the mirrors are running on *must* have an OpenSSH client installed-** The mirror will need an OpenSSH client installed to fetch changes from the primary Bitbucket Server.

 If you are using Bitbucket version 7.9 or higher, then you must use OpenSSH version 7.2 or higher to avoid sync failures for repositories.

- **The platform the mirrors are running on *should not* be under-provisioned-** Just as with your primary Bitbucket Data Center instance, your mirrors need to be provisioned with enough CPU, memory, and I/O resources to handle their peak workload. See [Scaling Bitbucket Server](#) for more information.

- **Minimum version of Bitbucket**-The primary instance and mirrors do not need to be running identical Bitbucket versions, but a mirror running Bitbucket 6.7 or later, can only point to a primary running Bitbucket 6.7 or later. An existing mirror running Bitbucket 6.6 or earlier can point to a primary running any supported Bitbucket version.

1. Set up the load balancer for your mirror farm

Your load balancer must:

- run on a dedicated machine
- have a high-speed LAN connection to the Bitbucket cluster nodes (that is, high bandwidth and low latency)
- support **both** HTTPS (for web traffic) **and** TCP (for SSH traffic)
- use HTTPS with a valid SSL certificate (signed by a Certificate Authority anchored to the root and not expired)
- have the mirrors running behind it all belong to the same mirror farm



You can use either a load balancer that supports session affinity ("sticky sessions") using the `BITBUCKETSESSIONID` cookie or you can set the `hazelcast.http.sessions=replicated` and sticky sessions are not required.

We also recommend that you terminate SSL (HTTPS) at your load balancer and use a HTTP connection from the load balancer to Bitbucket for better performance.



For more details on installing and configuring your load balancer, head to the [Install Bitbucket Data Center](#) page. If you don't have a preference for your load balancer, we also provide instructions here for `haproxy`, a popular open source software load balancer.

2. Install Bitbucket on the mirror node

The easy way to install a mirror is to download and run the Bitbucket installer and select the **Install a new mirror** option.



This is the same installer that you use for standard Bitbucket Server and Data Center instances. The mirroring functionality is included in the same distribution.

1. Go to www.atlassian.com/software/bitbucket/download and download the latest version of the Bitbucket installer.
2. Run the installer, and be sure to select **Install a new mirror**.
3. Do not start Bitbucket.
4. Add the following properties to your `{BITBUCKET_HOME}/shared/bitbucket.properties` file:

```
application.mode=mirror
setup.displayName=<Mirror name>
setup.baseUrl=<Mirror base URL>
plugin.mirroring.upstream.url=<Primary server URL>
```

5. Configure Hazelcast so that your mirror nodes talk to one another. For details, follow the instructions in the step, 4. *Start the first cluster node* in the [Install Bitbucket Data Center](#) page.

6. To add additional mirror nodes, repeat the above steps for installing a new mirror. There is no need to authorize the additional mirrors on your primary (upstream) instance if the properties match.

3. Start the mirror

Start your mirror:

```
sudo service atlbitbucket start
```

See [Start and stop Bitbucket](#) for more information on starting and stopping mirrors with the `service` command.

4. Approve the mirror farm request

Once a mirror farm has been installed and configured, a request is sent to the primary Bitbucket Data Center instance. From this page, its recommended that you verify the mirror farm ID against the one displayed at the mirrors base URL.

To approve a mirror farm request:

1. In your primary Bitbucket Data Center instance, go to **Admin > Mirrors**.
2. Click **Authorize** to approve the mirror request and start syncing the projects and repositories of the primary Bitbucket Data Center Instance.

 If the authorization request is declined, you'll return to the Mirrors page and there will be no pending mirror.

If the properties of additional mirrors don't match, it could stop your farm from starting. The new mirror could also start its own farm if it can't find other mirrors.

 An approved mirror farm will have access to all projects and repositories in the primary Bitbucket Data Center instance. Smart Mirroring has been designed to require secure communication throughout, and to restrict all functionality to the appropriate privilege level (such as system administrator) to ensure the security of all your sensitive information. It is still *your* responsibility to set up mirrors with the same stringent security practices as your primary Bitbucket Data Center instance. See the FAQ [How secure is Smart Mirroring?](#) for more information.

5. Decide which projects to mirror

Once a mirror farm is approved, you need to decide which projects to mirror. In the **Mirrors** settings page, type in the name of a project in the search box. Do this for each project you want to mirror. You can also choose to mirror all projects.

 If you choose to mirror all projects, you will not be able to select individual projects. This can't be undone. If you decide you no longer want to mirror all projects, you'll need to completely remove and then reinstall the mirror.

The screenshot shows the Bitbucket Administration interface. The top navigation bar includes the Bitbucket logo, 'Projects', 'Repositories', and a search bar. The main content area is titled 'Administration' and shows the configuration for a 'Production mirror farm'. The URL is 'https://production-mirror-farm.internal.website.com' and the ID is 'ABCD-EFGH-IJKL-MNOP'. A 'Delete' button is visible in the top right. Below this, there are tabs for 'Mirror nodes' and 'Projects'. The 'Mirror nodes in this farm' section contains a table with the following data:

| ID | Name * | Address | Status | Last synchronized |
|----------------------------|---------------------|---------------|--------|-------------------|
| c1234c12-b123-4123-8c32... | production-farm1... | /11.123.12... | ● | A moment ago |
| d1234c12-b123-4123-8c32... | production-farm2... | /11.123.12... | ● | A moment ago |
| 31234c12-b123-4123-8c32... | production-farm3... | /11.123.12... | ● | A moment ago |
| 41234c12-b123-4123-8c32... | production-farm4... | /11.123.12... | ● | A moment ago |
| g1234c12-b123-4123-8c32... | production-farm5... | /11.123.12... | ● | A moment ago |
| h1234c12-b123-4123-8c32... | production-farm6... | /11.123.12... | ● | A moment ago |
| i1234c12-b123-4123-8c32... | production-farm7... | /11.123.12... | ● | A moment ago |
| k1234c12-b123-4123-8c32... | production-farm8... | /11.123.12... | ● | A moment ago |
| z1234c12-b123-4123-8c32... | production-farm9... | /11.123.12... | ● | A moment ago |

For further reference and troubleshooting, read the [Bitbucket Data Center FAQ](#), which covers the most frequently asked questions about Bitbucket Data Center.

Monitoring your mirror farm

There are a number of helpful tools and techniques you can use to monitor the health of your mirror farm.

- [JMX metrics](#)
- [Endpoints](#)
- [Webhook](#)
- [Load balancer configurations](#)

Performance monitoring using JMX metrics

Java Management eXtensions (JMX) is a technology used for monitoring and managing Java applications. JMX can be used to determine the overall health of each mirror node and the mirror farm. The following statistics are most important to monitor:

- Hosting tickets on mirror nodes
- Mirror hosting tickets on the primary
- Incremental sync time on mirror nodes
- Snapshot sync time on mirror nodes
- As always its important to monitor your nodes for disk space, CPU and memory.

For more information and a complete list of JMX metrics, see [Enabling JMX counters for performance monitoring](#).

Synchronization and consistency

A **repo-hash** endpoint is provided on both the mirror farm and the primary server. Its used to check the consistency of a mirror farm and nodes with respect to the primary. This is the same endpoint that Mirror farm vet uses to repair any inconsistencies that come up, such as the result of a missing webhook. There are some important considerations to keep in mind when using this endpoint:

- The endpoint, `rest/mirroring/latest/repo-hashes`, is available on both the primary and the mirror nodes. It returns a stream of JSON containing a `content` and `metadata` hash for each repository. The `content` hash is a digest of the Git repository itself, while the `metadata` hash is a digest of the metadata that Bitbucket holds concerning the repository, such as the repository name.
- Content hashes or just metadata hashes are individually requested by calling `rest/mirroring/latest/repo-hashes/content` or `rest/mirroring/latest/repo-hashes/metadata`.
- This is what the payload looks like:

```
{
  "projects": [
    {
      "id": 1,
      "public": false,
      "repositories": [
        {
          "id": 1,
          "hashes": {
            "content": "082a2ffa1520447bb6c0072f9f9d850c76f111c0ff9a08cca8838b12b0ccc31a",
            "metadata": "b8fae6cb4704174f8dafae601355279950f921ba55b7620f4bdaa1280e735d14"
          }
        },
        {
          "id": 2,
          "hashes": {
            "content": "0000000000000000000000000000000000000000000000000000",
            "metadata": "e80aeaf459a69e7000b9e785eb39640a5d929f7ec4f09512a9ab6fabf4a0c80a"
          }
        }
      ]
    }
  ]
}
```

- The process to generate content hashes while reasonably fast needs to run against every repository on the instance, for larger instances this could take quite some time so we make an optimisation. When a upstream is first upgraded to a mirror farm capable version a empty content hash is generated for each repository this appears as 00 as can be seen in the `content` attribute of the second repository above. When the farm vet encounters a repository with a content hash of 00 it considers that repository up to date.
- A mirror will only return entries for the project or repository its mirroring. While the content returned from a mirror and the primary will be the same, the order of entries could be different. One way to sort the order consistently for diffing is to use the JQ query `jq '.projects | sort | .[]'. repositories |= sort_by(.id)'`

Webhook

The mirror synchronized webhook can be used to trigger builds as soon as the mirror has finished synchronizing. Its also useful for monitoring the repository in your mirror farm. Details of this repository event can be found in the [Event payload](#) page.

Monitoring the status of your mirrors

You can configure your load balancer to check the nodes status using the `/status` endpoint. A response code of 200 is returned if the mirror node is in a `SYNCHRONIZED` state. If there are no nodes in the `SYNCHRONIZED` state, a 200 response code will be returned for any mirror that is in one of the following states:

- `BOOTSTRAPPED`
- `BOOTSTRAPPING`
- `METADATA_SYNCHRONIZED`

 For customers who want a strict status endpoint we provide a `plugin.mirroring.strict.hosting.status` configuration property that when set to true, the `/status` endpoint returns a 200 response code only if the mirror is in the `SYNCHRONIZED` state. The setup for this configuration is outside the scope of the document. It is important to note that at least one mirror node should be accessible from the upstream server.

The table below displays each state and its description:

| State | Description |
|-----------------------|--|
| STARTING | A Bitbucket application is starting. |
| STOPPING | A Bitbucket application is stopping. |
| BOOTSTRAPPING | The mirror component is started. |
| BOOTSTRAPPED | The mirror has joined the cluster. If this is the first time the mirror farm has been connected to a primary, this is the state the application will wait in until it has been authorized. |
| METADATA_SYNCHRONIZED | Project or repository metadata has been synchronized from the primary and Git repositories have started synchronization. |
| SYNCHRONIZED | The mirror farm has synchronized all Git repositories from the primary.

If new projects or repositories are added to the mirror farm this state will not change. It indicates that the initial set of projects or repositories that were configured at startup time have been synchronized. |
| ERROR | There was an error starting the application node. |

 When performing a GET operation against the `/status` endpoint, the returned data is made up of JSON with two properties, `status` and `nodeCount`.

For example; `{ "state" : "SYNCHRONIZED" , "nodeCount" : "4" }`

Cloning LFS content from a mirror farm with more than one node fails

If you have only one mirror farm node or one mirror node, there is no issue and therepo is cloned successfully with the LFS files. However, once you deploy additional mirror nodes, you might encounter an issue with cloning LFS content from the mirror farm:

```
An "Authentication required: Authorization error" error is returned.
Error downloading object: Cute-Animals-that-Are-Not-Pets-812x464.jpg (8033694): Smudge error: Error
downloading Cute-Animals-that-Are-Not-Pets-812x464.jpg
(8033694255cebd1948f815f199d0091d2ea9804c6db9eea8f9043a2d037fb52e): Authentication required: Authorization
error: https://bbmirrorfarm2:8443/rest/git-lfs/storage/BITBUCKETDC2/DC4LFS/dc-lfs-repol
/8033694255cebd1948f815f199d0091d2ea9804c6db9eea8f9043a2d037fb52e
Check that you have proper access to the repository
```

The following message is logged in `$BB_HOME/log/atlassian-bitbucket.log`:

```
2020-09-10 00:09:18,327 WARN [http-nio-7990-exec-6] *1UUSRYCx9x1195x0 10.255.0.1,10.255.2.3 "POST /scm
/bitbucketdc2/dc4lfs/dc-lfs-repol.git/info/lfs/objects/batch HTTP/1.1" c.a.j.i.s.
DefaultAuthenticationResultHandler Signature mismatch during JWT authentication, issuer: com.atlassian.
bitbucket.server.bitbucket-git-lfs
com.atlassian.jwt.exception.JwtSignatureMismatchException: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
eyJzdWUiOiIxIiwicmVhbyI6MTIsImVzcyI6ImNvbS5hdGxhc3NpYW4uYml0YnVja2V0LnNlcnZlci5iaXRidWNrZXQtZ210LWxmcyIsImNv
bnRleHQiOiOnsidXNlciI6eyJoaWdoZXN0UGVybWlzc2l2b2IiOiI6ImlNZU19BRElJTIiImRpc3BsYXl0YWllIjojQWRtaW4iLCJzbHVnIjojYWRt
aW4iLCJlc2VyS2V5IjojMSIsInVzZXJyZWllIjojYWRtaW4ifX0sImV4cCI6MTU5OTY5Njg1OCwiaWF0IjojNTk5Njk2NTU4fQ.
b2hGyy1RF1tee3X5zXdwelQfqh1XlyR18pCXsAhBv1E
    at com.atlassian.jwt.core.reader.NimbusJwtReader.verify(NimbusJwtReader.java:164)
    at com.atlassian.jwt.core.reader.NimbusJwtReader.read(NimbusJwtReader.java:74)
    at com.atlassian.jwt.core.reader.NimbusJwtReader.readAndVerify(NimbusJwtReader.java:57)
    at com.atlassian.jwt.internal.DefaultJwtService.verifyJwt(DefaultJwtService.java:49)
    at com.atlassian.jwt.internal.sal.JwtAuthenticatorImpl.verifyJwt(JwtAuthenticatorImpl.java:62)
    at com.atlassian.jwt.core.http.auth.AbstractJwtAuthenticator.verifyJwt(AbstractJwtAuthenticator.
java:118)
    at com.atlassian.jwt.core.http.auth.AbstractJwtAuthenticator.authenticate(AbstractJwtAuthenticator.
java:71)
    at com.atlassian.jwt.internal.sal.JwtAuthenticatorImpl.authenticate(JwtAuthenticatorImpl.java:30)
    at com.atlassian.jwt.internal.servlet.JwtAuthFilter.mayProceed(JwtAuthFilter.java:79)
    at com.atlassian.jwt.internal.servlet.JwtAuthFilter.doFilter(JwtAuthFilter.java:35)
    at com.atlassian.stash.internal.spring.lifecycle.LifecycleJohnsonServletFilterModuleContainerFilter.
doFilter(LifecycleJohnsonServletFilterModuleContainerFilter.java:42)
    at com.atlassian.stash.internal.web.auth.BeforeLoginPluginAuthenticationFilter.
doBeforeBeforeLoginFilters(BeforeLoginPluginAuthenticationFilter.java:90)
    at com.atlassian.stash.internal.web.auth.BeforeLoginPluginAuthenticationFilter.doFilter
(BeforeLoginPluginAuthenticationFilter.java:73)
    at com.atlassian.stash.internal.request.DefaultRequestManager.doAsRequest(DefaultRequestManager.
java:87)
    at com.atlassian.stash.internal.hazelcast.ConfigurableWebFilter.doFilter(ConfigurableWebFilter.java:
38)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
    at java.lang.Thread.run(Unknown Source)
... 68 frames trimmed
```

Workaround 1:

Upgrade your Bitbucket to at least:

- Bitbucket **6.10.7**
- Bitbucket **7.3.2**

Workaround 2:

The issue happens due to the way LFS authentication is being handled between the mirror farm nodes. If for some reason you can not upgrade your instance, the only option left is **removing additional mirror** farm nodes leaving only one.

Please be advised you may still use multiple mirrors, just not as a mirror farm.

Export and import projects and repositories

Data Center Migration is a tool for admins that can be used to:

- consolidate multiple Bitbucket instances
- move from a Bitbucket Server to a Bitbucket Data Center instance
- selectively export/import projects and repositories from one Bitbucket Data Center instance to another

Git data can be imported or exported into Bitbucket Data Center from another Bitbucket Server or Data Center deployment, along with pull requests, comments and attachment history.

This feature is only available to customers with a Bitbucket Data Center instance.

Before starting a migration, review the below information carefully. There is also information around [troubleshooting, canceling and cleanup](#), and [error and warning messages](#).

Before you start

This is a high-level description of considerations and tasks to review and complete before performing a migration. Read through the following sections carefully before starting a migration.

User Interface

There's no graphical user interface in the initial ([Bitbucket Server 5.14](#)) release of this feature. Instead, REST calls can be made to control the migration process.

To perform the migration you will need to either:

- use your favorite tool to make REST calls, or
- refer to the provided [sample cURL commands for import](#) and [sample cURL commands for export](#).

If you plan to use cURL commands, we recommend you [setup a .netrc file](#) (sample commands use the `-nfl ag`) and install [jq](#) on your machine.

Using the preview REST endpoint

You can use the `preview` function to review the scope of the export, prior to a migration.

The preview takes into account fork hierarchies, which are always migrated fully.

Disk space requirements

Always make sure you have enough disk space for the files you're migrating. The space required on disk during export is roughly the size of the Git data and attachments being exported.

Additional space is required for ElasticSearch, and for your database server.

If you are unsure what space you have available, the following knowledge base article tells you how to [identify a repository ID](#) in Bitbucket, which you can then use to check the available disk space.

Issues with project name conflicts

Before migration, check the names of projects on both the source and destination instances. If a project name is in use **on the destination instance**, the project being **imported** will be renamed.

To avoid this either:

- check all project names and rename **before** migration, or
- **after** the migration, review the warnings on the import job, and rename projects manually.

The same logic applies to repository names in personal projects.

Time needed for export

There is no real way to predict the time required for the export, it is impacted by your data set, individual load, and traffic.

Some tests have shown that an export of 200GB can take anywhere between 2 - 4 hours, but this can be higher.

Time needed for import

There is no real way to predict the time required for import, it is impacted by the size of the export archive being imported, the number of pull requests, and the number of repositories.

An import **will take longer than an export** (roughly four times as long). This is because indexing and validation (performed during import) doesn't occur during export, and remains dependent on your data set, individual load, and traffic.

User and license management recommendations

Non-active users are added to a generated list of "deleted users" on **export**, and do not take up licenses on migration.

We strongly recommend that both the source and destination instances are connected to the **same user directory** prior to migration.

Even though users will not be migrated, their permissions on projects and repositories will, and permissions are matched by *username*. If users are missing on the destination instance, permissions might not be migrated and a warning will be logged.

If user directories are different on both instances, it is possible that users with the same *username* exist on the destination instance, but are actually different users in the source instance. This would result in incorrect permissions being granted to those users on import.

Third party plugins

It's important to know that third party plugins **will not be migrated, neither will their data**.

Plugins can implement their own migrations using [this API](#).

Pull request comments

If a pull request comment references an attachment that was uploaded to a different repository, there is a possibility that the attachment will not be successfully imported.

E.g if a user has uploaded an attachment to a comment on a pull request in one repository, then copies and pastes the link into a **different** comment on a pull request, in another repository.

This will not impact the import of the rest of the comment, only the attachment.

For more information see the [troubleshooting, canceling and cleanup page](#).

Entities included in migration

The below table lists the entities that will be imported and exported during migration.

 **Git LFS objects** migration needs to be run separately. You can use this [Git LFS migration process](#) to export these objects.

| Category | Item | Import | Export |
|----------|------|--------|--------|
|----------|------|--------|--------|

| | | | |
|----------------------------|--|---|---|
| Git | Git repositories on file system | ✓ | ✓ |
| Git LFS | LFS objects on file system | ✗ | ✗ |
| Pull requests | Pull request metadata | ✓ | ✓ |
| | PR user info | ✓ | ✓ |
| | PR comments | ✓ | ✓ |
| | PR attachments | ✓ | ✓ |
| | PR File comments | ✓ | ✓ |
| Project Settings | Description | ✓ | ✓ |
| | Project permissions (user access) | ✓ | ✓ |
| | Project permissions (group access) | ✓ | ✓ |
| | Project permissions (public) | ✓ | ✓ |
| | Project permissions (default permission) | ✓ | ✓ |
| Repository Settings | Default branch | ✓ | ✓ |
| | Forks | ✓ | ✓ |
| | Transcode diffs | ✓ | ✓ |
| | LFS enabled | ✓ | ✓ |
| | Repo permissions (user access) | ✓ | ✓ |
| | Repo permissions (group access) | ✓ | ✓ |
| | Repo permissions (public access) | ✓ | ✓ |
| | Git hooks on disk | ✓ | ✓ |
| | Repository hooks | ✗ | ✗ |

Exporting

Before you start your export

Always carefully review the [migration prerequisites](#) before starting the export process.

Decide which project or repositories you want to migrate.

You will require project keys and repository slugs, which you can get from the URL of the repository.

Visiting an URL for a repository

The format is `https://bitbucket.example.com/projects/PROJECT/repos/repository`, where **PROJECT** is the project key and **repository** is the repository slug.

Previewing the export

Previewing allows you to experiment with the export request and returns a list of repositories that would be included with a given request.

You can perform a preview using the below command:

```
curl -s -n -X POST -H 'Content-type: application/json' -d '{"repositoriesRequest":{"includes":[{"projectKey":"**","slug":"**"}]}}' http://localhost:7990/rest/api/1.0/migration/exports/preview | jq .
```

The following request values are required for preview ([REST Documentation](#)):

| Description | Value |
|----------------|--|
| URL | <code>/rest/api/1.0/migration/exports/preview</code> |
| HTTP verb | POST |
| HTTP header | <code>Content-type: application/json</code> |
| Authentication | Basic |

To preview all repositories:

```
Request body

{
  "repositoriesRequest": {
    "includes": [
      {
        "projectKey":"**",
        "slug":"**"
      }
    ]
  }
}
```

To preview one full project and a specific repository:

Request body

```
{
  "repositoriesRequest": {
    "includes": [
      {
        "projectKey": "PROJECTKEY",
        "slug": "*"
      },
      {
        "projectKey": "PROJECTKEY2",
        "slug": "repository-slug"
      }
    ]
  }
}
```

If needed, you can specify the project key and slug pair as many times as necessary.

 As soon as you select one repository of a fork hierarchy, then every repository of that fork hierarchy will be exported, including personal forks and origins of the repository.

Performing the export

Once the export is started, an archive file will be written to disk containing all necessary Git data and database entities.

 The archive file can be found after the migration is finished, in `$BITBUCKET_SHARED_HOME/data/migration/export/Bitbucket_export_<Job ID>.tar`

You can perform the export using the below command:

```
curl -s -n -X POST -H 'Content-type: application/json' -d '{"repositoriesRequest":{"includes":[{"projectKey":"*","slug":"*"}]}}' http://localhost:7990/rest/api/1.0/migration/exports | jq .
```

The following values are required for export([REST Documentation](#)):

| Description | Value |
|----------------|--|
| URL | <code>/rest/api/1.0/migration/exports</code> |
| HTTP verb | POST |
| HTTP header | <code>Content-type: application/json</code> |
| Authentication | Basic |

Note: exactly the same request body as with previewing the export can be used.

To export all repositories:

Request body

```
{
  "repositoriesRequest": {
    "includes": [
      {
        "projectKey": "*",
        "slug": "*"
      }
    ]
  }
}
```

To export one full project and a specific repository:

Request body

```
{
  "repositoriesRequest": {
    "includes": [
      {
        "projectKey": "PROJECTKEY",
        "slug": "*"
      },
      {
        "projectKey": "PROJECTKEY2",
        "slug": "repository-slug"
      }
    ]
  }
}
```

You can specify the `projectKey` and `slug` pair as many times as necessary.



As soon as you select one repository of a fork hierarchy, then every repository of that fork hierarchy will be exported, including personal forks and origins of the repository.

The response will contain the job ID, which is important to query the status of the job later.

During the export

You can query the following specific REST endpoint to see progress:

```
watch -n30 'curl -s -n -X GET http://localhost:7990/rest/api/1.0/migration/exports/<Job ID> | jq .'
```

Replace `<Job ID>` with the `"id"` value that was returned from request that started the export job.

The following values are required to query the progress ([REST Documentation](#)):

| Description | Value |
|-------------|---|
| URL | <code>/rest/api/1.0/migration/exports/<Job ID></code> |

| | |
|----------------|-------|
| HTTP verb | GET |
| Authentication | Basic |

After the export

On completion, the job state will change to `COMPLETED` and the export file can be found in `$BITBUCKET_SHARED_HOME/data/migration/export/Bitbucket_export_<Job ID>.tar`. If any errors occurred, the job state will change to `ABORTED` for fatal errors or `FAILED`.

To check for warnings or errors:

```
curl -s -n -X GET http://localhost:7990/rest/api/1.0/migration/exports/<Job ID>/messages | jq .
```

Replace `<Job ID>` with the "id" value that was returned from request that started the export job.

The following values are required to check for warnings or errors ([REST Documentation](#)):

| Description | Value |
|----------------|--|
| URL | <code>/rest/api/1.0/migration/exports/<Job ID>/messages</code> |
| HTTP verb | GET |
| Authentication | Basic |

Warning messages should be dealt with on a case by case basis, it generally means that some data was not exported but the archive can be used to import into another instance.

Error messages generally mean that the archive that was generated is not usable. Please see the [troubleshooting section](#) for more information.

Canceling the export

You may see a small delay between accepting the request and the actual job cancelation. In most cases, the delay will be inconsequential.

In the unlikely case of communication issues across a cluster, it may take a few seconds to cancel a job.

You should always actively query the job status to confirm that a job has been successfully canceled.

The export archive will remain in place, but can not be used to perform an import and should only be used to diagnose problems.

To cancel the export:

```
curl -s -n -X POST -H 'Content-type: application/json' http://localhost:7990/rest/api/1.0/migration/exports/<Job ID>/cancel
```

Replace `<Job ID>` with the "id" value that was returned from request that started the export job.

The following values are required to cancel the export ([REST Documentation](#)):

| Description | Value |
|-------------|-------|
|-------------|-------|

| | |
|----------------|--|
| URL | <code>/rest/api/1.0/migration/exports/<Job ID>/cancel</code> |
| HTTP verb | POST |
| Authentication | Basic |

Importing

Before you start your import

Always carefully review the [migration prerequisites](#) before starting the import process.

Copy your import file to the target instance by copying into `$(BITBUCKET_HOME)/shared/data/migration/import` and specifying the path relative to the import directory in the REST call. Your import file must be copied to this directory or your import will not work.

After copying the file, it is recommended to verify its integrity and that it has been copied without errors. You can do this for example by calculating the export archive's MD5 sum (e.g. by using the `md5` tool or similar) on both the source and the target instance and checking that the result is the same.

Performing the import

Once you call the REST endpoint, the importer will begin to populate the database and file system.

⚠ You will not be prompted to resolve project-key collisions, they will be resolved automatically and a warning message will be added to the job. See section [After the import](#) below for how to query job messages.

You can perform the import using the below command:

```
curl -s -n -X POST -H 'Content-type: application/json' -d '{"archivePath":"Bitbucket_export_422.tar"}' http://localhost:7990/rest/api/1.0/migration/imports | jq .
```

In the above example, the file '**Bitbucket_export_422.tar**' was imported from the `home/shared/data/migration/import` directory. The `archivePath` parameter can either be relative or absolute.

Example

Relative (to `$(BITBUCKET_HOME)/shared/data/migration/import`):

```
{"archivePath":"Bitbucket_export_422.tar"}
```

The following request values are required for import ([REST Documentation](#)):

| Value | Description |
|----------------|--|
| URL | <code>/rest/api/1.0/migration/imports</code> |
| HTTP verb | POST |
| HTTP header | <code>Content-type: application/json</code> |
| Authentication | Basic |

The response will contain the job ID, which is important to query the status of the job later.

During the import

You can query the following specific REST point to see progress:

```
watch -n1 'curl -s -n -X GET http://localhost:7990/rest/api/1.0/migration/imports/<Job ID> | jq .'
```

Replace<Job ID>with the"**id**"value that was returned from request that started the import job.

The following request values are required to query the progress ([REST Documentation](#)):

| Description | Value |
|----------------|--|
| URL | /rest/api/1.0/migration/imports/<Job ID> |
| HTTP verb | GET |
| Authentication | Basic |

During the import process, no modifications are made to the source instance, so you can roll backto it if anything goes wrong.

After the import

On successful completion, the job state will change to "completed". If any errors occurred, the job state will change toABORTEDfor fatal errors orFAILED.

To check for warnings or errors:

```
curl -s -n -X GET http://localhost:7990/rest/api/1.0/migration/imports/<Job ID>/messages | jq .
```

Replace<Job ID>with the"**id**"value that was returned from request that started the import job.

The following values are required to check for warnings or errors ([REST Documentation](#)):

| Description | Value |
|----------------|---|
| URL | /rest/api/1.0/migration/exports/<Job ID>/messages |
| HTTP verb | GET |
| Authentication | Basic |

Warning messages should be dealt with on a case-by-case basis, it generally means that some data was not completely imported, or that name collisions were resolved.

Error messages mean that the import was not successful and some data was only partially imported or not at all. Please see the [troubleshooting section](#) for more information.

Canceling the import

Import jobs are not canceled instantaneously like export jobs. During import, once the cancel request has been accepted, the following checkpoints will apply, then the job will stop:

- after the current fork hierarchy has been imported and verified.
- before the next repository is imported.
- before the next pull request is imported.

This means that some repositories might be missing from a fork hierarchy, or some pull requests might be missing from repositories. Git data in a repository will always be consistent if the job was canceled cleanly.

To cancel the import:

```
curl -s -n -X POST -H 'Content-type: application/json' http://localhost:7990/rest/api/1.0/migration/imports/<job ID>/cancel
```

Replace<Job ID>with the"**id**"value that was returned from request that started the import job.

The following request values are required to cancel the import ([REST Documentation](#)):

| Description | Value |
|----------------|---|
| URL | /rest/api/1.0/migration/imports/<job ID>/cancel |
| HTTP verb | POST |
| Authentication | Basic |

LFS Migration

Git LFS data and objects are not migrated using the Data Center Migration tool, and/or cloned or re-pushed manually.

LFS enabled is a per-repository setting. This means that after the import if you have it enabled on the source, **LFS enabled** will be enabled on the target.

Before you start

The LFS migration process requires read access to all affected repositories on the **source** instance, and write access to the same repositories, on the **target** instance.

Preparing the export

 **Export outputs generated when using Data Center Management do not contain LFS objects.**
This process is required for LFS objects only.

Clones can be done before, during or after the export. If you discard the source after exporting, your export archive does not contain the LFS objects. These exist only in the source or in clones that you've made separately.

To prepare for migrating LFS objects you will need Git LFS installed on the client. Refer to [Git Large File Storage](#) or the process if you are unsure.

1. Clone the relevant repositories that include the LFS objects you want to migrate:

- For each Git LFS enabled repository, a clone has to be created on a machine that can access both the source and target instance of a migration.
- Otherwise, clone from the source instance first, and transfer the clone later.
- You may have existing clones as part of your development workflow that you can use. If you can't or don't want to rely on existing clones, it's best to create dedicated clones for the migration process.

To migrate the full history of LFS objects, you need full clones: use `git clone $source_repo_url` for each LFS-enabled repository.

If a recent history will be sufficient, you can use a shallow clone: `git clone $source_repo_url --depth 1` or similar, for each LFS-enabled repository.

2. Run `git lfs fetch --all` on each individual clone to retrieve the LFS objects.

Exporting LFS objects/data

You can find the [full export process](#) here if needed.

1. Export relevant repositories using the existing data center migration process.
2. Maintain respective clones with LFS objects separately, as described above.

Importing LFS objects/data



Separate clones are created prior to export, as only the **setting** that enables LFS will be restored, not the objects.

1. Import relevant repositories using the existing [data center import process](#).

These repositories will have LFS enabled as when they were exported.

2. Once the import is completed, add the new origin to each of your clones:
`git remote add $target_instance_name $target_repo_url`

3. To complete the process, push LFS objects:
`git lfs push --all $target_instance_name`

Troubleshooting, canceling and cleanup

Exporting forks

When selecting individual repositories for exporting, the following rules apply:

- Once you select a fork in any hierarchy, the entire hierarchy will be included.
- If you only select the root repository, all forks will still be included.

Conflict resolution

Conflict resolution is automatic. If a project with the same name or key already exists, the imported project is renamed. After the import has finished, imported projects and repositories can be renamed or moved around as required.

For personal repositories, if a repository with the same name already exists, the imported repository is renamed.

Cancellation

If you cancel during export: The migration will stop immediately, and the archive is not usable.

If you cancel during import: The migration will complete importing the current repository, then stop. It's important to understand that if you cancel midway through the import, the fork hierarchy may not be complete and subsequent imports will not reuse already existing fork parents.



If you cancel while importing pull requests, the import will finish importing the current pull request, then stop.

If you cancel while importing Git data, no pull requests will be imported.

Import complete with errors

Every job (both import and export) has messages that you can query from the REST API.

Warnings:

If a job is completed but has warnings: you should read the warnings and decide if any action is required.

Errors:

Errors mean something is broken or will fail.

The job will not complete, and you will need to identify and mitigate any errors you receive.

Repositories that remain "Initializing":

Whenever there is an error, there is a chance repositories will stay in a state that says "initializing". This are a technical error, and will not resolve on their own. **Currently you will need to log a support case if you see this behaviour.**

Rollback and Delete

There is no automatic rollback, you will need to manually go through and delete the data you do not want, before trying again.

Will source or target be down during migration?

Source and target will both remain available during a migration, during both import and export. This may negatively impact performance, and it should be run during low usage periods if possible.

Export archives may become bigger or less efficient if the instance is being used during migration. Because of this it is not recommended to use an instance during a migration.

What to do if your migration is corrupted

The export can be corrupted: When the job is completed and has no errors in the job messages, you can use the archive.

Import: there is a chance when the import fails, repos will stay in an initializing state. This is a fatal error, and cannot be remediated. You have to go and delete any initializing repos.

There is a [REST endpoint](#) which can be used to find all repos that are in state initializing.

Attachments in comments on pull requests

Attachments are uploaded and stored in the repository of the comment they added in. If a pull request comment references an attachment that was uploaded to **adifferent** repository, there is a possibility that the attachment will not be successfully imported. The following will be logged (and added as a warning to the import job) if an attachment is not successfully imported:

```
WARN c.a.s.i.m.DefaultImportContext Warning registered for job 1 of type com.atlassian.bitbucket.migration.import (RUNNING) against pr attachment: Could not find repository created with export ID '1,234'
```

Workaround: On the source instance, [find the repository with the export ID](#) that was logged when the attachment link failed to resolve on import (see example above).

If this repository was exported and imported onto the target instance, find the repository ID on the target instance, and update the repository ID in the attachment link in the comment text of any comments referencing it.

If this repository was not exported, replace the attachment link by downloading the attachment from the source instance and reattaching it to the comment on the target instance.

Error and warning messages

Import and export jobs may have associated messages after a migration is completed.

The messages range in severity from low (warnings) to high (errors), and if a job is associated with error messages, there is a chance it has not completed successfully.

Warnings

Warnings contain information that is important for the admin to consider before using an export archive or after importing archives.

Errors

A job is not considered successful if at least one error message is registered with it. More information about an error, such as the full stack trace for exceptions, is available in the server logs.

Export Warnings

 Number of repositories / projects has changed since the export was first initiated.

Meaning: Repositories may have been created, deleted, renamed or moved while the export job was executing. This has no effect on the export, except for repositories not being exported if they don't match the repository selector anymore, or projects being exported if they now match the repository selector.

Recommended action: None.

Export errors

 Could not serialize object with id: '<ID>' to JSON

Meaning: An exception occurred while exporting pull request data. This means that the export archive will be missing some data related to pull requests.

Recommended action: Delete export archive and retry export.

 Could not export attachments for '<ID>': an unexpected error occurred: <MESSAGE>

Meaning: An exception occurred while exporting repository attachments. This means some attachments for the repository mentioned will be missing.

Recommended action: Delete export archive and retry export.

 Exception executing callback '<METHOD>' on '<EXPORTER>': '<MESSAGE>'

Meaning:

An exception occurred while executing a request to an exporter interface. Data Center Migrations uses a collection of exporters to migrate data. If this message is logged, some data of their related entities might be missing:

- **GitRepositoryExporter:** Git data might be corrupt or missing. The resulting export archive will most likely contain corrupt Git data.
- **GitLfsSettingsExporter:** The setting whether Git LFS is enabled was not exported.
- **MetadataExporter:** Project or repository metadata, such as name, description, owner, fork hierarchy or public accessibility setting, may not have been exported correctly.
- **PullRequestExporter:** Data related to pull requests, such as title, description, reviewers, and other activity, may not have been exported correctly.
- **RefSyncStatusExporter:** For forked repositories that have fork synchronization enabled, the synchronization status may not have been exported correctly.
- **PermissionExporter:** Permissions for projects or repositories may not have been exported completely.

Recommended action:Delete export archive and retry export.

Import warnings

```

❗ Failed to delete file: <FILE>.
Failed to delete temp directory: <DIRECTORY>.
Following errors happened while deleting temp directory:
<ERRORS>

```

Meaning:During import, some temporary files and directories are created and then cleaned up. This message means that a file or directory could not be cleaned up and may have to be cleaned up manually.

Recommended action:Delete these files after import manually.

```

❗ Custom hooks have been imported and stored in directory:
<DIRECTORY>.

```

Meaning:Git repositories on disk may contain custom hooks that an administrator has placed there manually. These hooks are not migrated as-is, but are placed in a different directory so Git does not use them.

Recommended action:To enable these custom hooks again, they have to be moved out of this directory and into the corresponding 'hooks' directory of the repository.

```

❗ Unknown entry '<ENTRY>' found in archive

```

Meaning:This message is registered when the export archive contains an entry that the import job did not handle. This message can have a multitude of causes, and means that the data related to this entry is not imported. The most common cause is when an export archive has been tampered with.

Recommended action:Verify that repositories have been imported correctly.

```

❗ Detected inconsistency with repository '<REPOSITORY>'

```

Meaning: Integrity checks are run on repositories after they have been imported. It is possible for these checks to find and fix inconsistencies, which are logged as warnings.

Recommended action: None.

Import errors

❗ Could not import comment activity '<ACTIVITY>' for pull request '<PULL REQUEST>': the referenced comment could not be found. This may be due to an earlier import error.

Meaning: A pull request activity (such as replies to comments) is referencing a comment that could not be found. This means that some activity stream data will be missing from the mentioned pull request.

Recommended action: Check the server logs and verify that the pull request has been imported correctly.

❗ Could not import activity '<ACTIVITY>' for pull request '<PULL REQUEST>': an unexpected error occurred and the activities in the current batch will be rolled back: <MESSAGE>

Meaning: A pull request activity could not be imported due to the mentioned error message. Some data related to this pull request will be missing.

Recommended action: Check the server logs and verify that the pull request has been imported correctly.

❗ Error performing an integrity check. Further checks will continue.

Meaning: Integrity checks are run on repositories after they have been imported. It is possible for these checks to find errors that can not be fixed.

Recommended action: Check the server logs. Log messages preceding this one may give more information about why this check failed.

❗ Checking the integrity of imported repository hierarchy <HIERARCHY ID> failed with an unexpected error: <MESSAGE>

Meaning: Integrity checks have failed with an unexpected error.

Recommended action: Check the server logs. Log messages preceding this one may give more information about why this check failed.

❗ Checking the integrity of imported repositories failed with an unexpected error: <MESSAGE>

Meaning: Integrity checks have failed with an unexpected error.

Recommended action: Check the server logs. Log messages preceding this one may give more information about why this check failed.

 The importing of repository '<REPOSITORY>' failed to complete: an unexpected error occurred while attempting to make it available to the system: <MESSAGE>

Meaning: The repository state could not be set to AVAILABLE.

Recommended action: These repositories failed to import and may have to be cleaned up manually. Delete these repositories using the REST API.

 Exception executing callback '<METHOD>' on '<IMPORTER>': '<MESSAGE>'

Meaning:

An exception occurred while executing a request to an importer interface. Data Center Migrations uses a collection of importers to migrate data. If this message is logged, some data to their related entities might be missing:

- **GitRepositoryImporter:** Git data was not imported completely. The repository is most likely not usable and should be deleted.
- **GitLfsSettingsImporter:** The setting whether Git LFS is enabled was not imported.
- **MetadataImporter:** Project or repository metadata, such as name, description, owner, fork hierarchy or public accessibility setting, may not have been imported correctly.
- **PullRequestImporter:** Data related to pull requests, such as title, description, reviewers, and other activity, may not have been imported correctly.
- **RefSyncStatusImporter:** For forked repositories that have fork synchronization enabled, the synchronization status may not have been imported correctly.
- **PermissionImporter:** Permissions for projects or repositories may not have been imported completely.

Recommended action: The affected repositories were most likely not imported correctly. They should be deleted using the REST API and another attempt at importing them should be made. If repeated attempts fail, it might help to export again.

Git Large File Storage

Git Large File Storage (LFS) is a [Git extension](#) that improves how large files are handled. It replaces them with tiny text pointers that are stored on a remote server instead of in their repository, speeding up operations like cloning and fetching.

Bitbucket Data Center and ServerShip with Git LFS enabled at an instance level, but disabled for each repository. It also includes an embedded LFS object store, removing the need for an external one.

On this page

- [Important Notices](#)
- [Enable Git LFS for a repository](#)
- [Install and use the Git LFS command line client](#)
- [Disable Git LFS](#)
- [Embedded Object Store](#)
- [Repository Forking](#)
- [Smart Mirroring](#)
- [Limit Disk Space Usage](#)
- [Limit Network Connections](#)

Important Notices

HTTP(S) must be enabled

Git LFS supports SSH remotes, but downloading and uploading objects is done via HTTP(S). If SCM connections to Bitbucket over HTTP(S) are not enabled, Git LFS will not work.

The Bitbucket base URL should also be configured to a HTTP(S) URL.

We recommend configuring SSL

If your instance is available over the internet, usernames, passwords, and other data may be at risk. See [Proxy and secure Bitbucket](#) for more info.

The Data Center Migration Tool does not export Git LFS

For more information see [LFS Migration](#).

Enable Git LFS for a repository

Git LFS is disabled by default.

To enable Git LFS:

1. Go to **Repository settings > Large file storage (LFS)**.
2. Select **Allow LFS**.
3. Select **Save**.

If you cannot select **Allow LFS**, this is because the instance administrator has disabled LFS support on the instance.

Install and use the Git LFS command line client

Git LFS aims to integrate with the standard Git workflow as seamlessly as possible.

To push your first Git LFS files to an existing repository:

1. Enable Git LFS support for the repository ([see above](#)).
2. Download and install the [git-lfs command line client](#).
3. Install the Git LFS filters:

```
git lfs install
```

This adds the following lines to the `.gitconfig` file located in your home directory:

```
[filter "lfs"]
  clean = git-lfs clean %f
  smudge = git-lfs smudge %f
  required = true
```

The above change applies globally, so it is not necessary to run this for each repository you work with.

4. Choose the file types you would like LFS to handle by executing the `git lfs track` command. The `git lfs track` command creates or updates the `.gitattributes` file in your repository. Change to your cloned repository, then execute `git add` to ensure updates to the `.gitattributes` are later committed:

```
git lfs track "*.jpg"
git add .gitattributes
```

5. Add, commit, and push your changes as you normally would:

```
git add image.jpg
git commit -m "Added an image"
git push
```

When pushed, the Git tree updates to include a pointer to the actual file content. This pointer will include the SHA256 hash of the object and its size in bytes. For example:

```
oid sha256:4fa32d6f9b1461c4a53618a47324e243e36ce7ceae72ad440cc811a7e6881be1
size 1580060
```

The object itself will be uploaded to a separate location via the [Git LFS Batch API](#). Specifically, the object will be uploaded to an embedded LFS object store within your Bitbucket instance.

Disable Git LFS

Git LFS support is enabled by default. It can, however, be disabled. This will prevent upload or download of LFS objects for all repositories, but it won't delete existing LFS objects stored within Bitbucket.

To disable Git LFS:

1. Log in with sysadmin permissions.
2. Go to  > **Server settings**.
3. Untick **Git LFS enabled**.
4. Select **Save**.

Embedded Object Store

Bitbucket includes an embedded LFS object store. Uploaded Git LFS objects are stored in the directory `$BITBUCKET_HOME/shared/data/git-lfs/storage`. This storage location cannot be changed, as for clustered deployments it is critical that LFS object storage reside on the `shared` filesystem so it is globally visible to all cluster nodes.

Repository Forking

LFS enabled repositories support forking and a full fork-based workflow including [pull requests](#) between forks and [fork synchronization](#). Forking is implemented through sharing of objects between fork and origin repositories and as such forks are lightweight. This means creating a fork is faster because objects are not duplicated when a fork is created.

Resolution of merge conflicts is slightly different from the non-LFS case, specifically where a merge conflict exists between two LFS objects. Since the Git repository simply contains an LFS pointer, conflict resolution must be performed with the conflicted pointer file.

Such a conflict might look like this:



Conflict: Added file to Source, Added same file to Target

This file is in a conflicted state. You will need to resolve the conflict manually before you can merge this pull request.

| | | |
|---|-----|---|
| 1 | 1 | version https://git-lfs.github.com/spec/v1 |
| | 2 + | <<<<<<< |
| 2 | 3 | oid sha256:4fa32d6f9b1461c4a53618a47324e243e36ce7ceae72ad440cc811a7e6881be3 |
| 3 | 4 | size 1580067 |
| | 5 + | ===== |
| | 6 + | oid sha256:6bb99acfa40fe1c259397afea7fb18055cb25a8d4f0ae7dee9924fb76ec5ca72 |
| | 7 + | size 1083955 |
| | 8 + | >>>>>> |

Users are still required to resolve the merge conflict manually using the command line Git client, as they normally would. The merge strategy, since this is only a pointer, can only be an *ours* or *theirs* based strategy, keeping one or the two pairs (oid/size). When resolving the conflict, if the pointer file becomes corrupted it will not be recognized as an LFS file.

Smart Mirroring

[Smart Mirroring](#) supports mirroring of Git LFS objects as of Bitbucket 4.5. Mirroring of Git LFS objects is performed on-demand; that is, when a client requests download of a Git LFS object from the mirror node, the object will be streamed from the upstream node if it is not already available on the mirror. Subsequent downloads of the same object will be downloaded directly from the copy stored on the mirror.

Limit Disk Space Usage

By default Bitbucket won't permit Git LFS uploads if the amount of free disk space in the filesystem that hosts the storage directory has less than 100 megabytes of free disk space. When a client attempts to upload (via a Git push) an LFS file that would result in the free disk-space threshold being exceeded, an error message indicating as such is sent to the client:

```
$ git push
Git LFS: (0 of 1 files, 1 skipped) 0 B / 348.59 MB, 348.59 MB skipped
[aa39717817507d0ae3434a36347159e4970aec061c8c506f197c0eeadd2e8efe2] Insufficient free space in store
error: failed to push some refs to 'https://bitbucket.example.com/bitbucket/scm/myproject/myrepo.git'
```

A warning will also be logged in the `atlassian-bitbucket.log` file. For example:

```
2016-01-01 18:00:00 WARN [http-nio-7990-exec-2] user @G6I7R6x969x556x0 0:0:0:0:0:0:1 "POST /scm
/myproject/myrepo.git/info/lfs/objects/batch HTTP/1.1" c.a.b.i.s.g.l.s.e.EmbeddedStoreAccessor Upload
rejected due to insufficient free space in store - Required: 1073741824 Free: 10485760
```

The free disk-space threshold can be tuned by adding the following property to the [config properties](#) file (this example tunes the threshold to one gigabyte, and note that this value is in bytes):

```
plugin.bitbucket-git-lfs.minimum.free.space=1073741824
```

Setting the value to zero will disable the free disk space check and will allow files to be uploaded even if doing so would exhaust all available disk space. This however is not encouraged, it can lead to situations where the system exhausts all disk space but is unable to log error messages stating the same (because the disk is full), leading to difficulties debugging problems that are caused by disk space exhaustion.

```
plugin.bitbucket-git-lfs.minimum.free.space=0
```

Bitbucket must be restarted for the change to take effect.

Limit Network Connections

Large Git LFS download or upload operations over very slow network links could take many minutes, or even hours. Bitbucket supports a finite number of HTTP connections (by default 200). If Git LFS were permitted to exhaust this connection pool then the user interface, Git hosting, and REST API access could be impacted. For this reason a default limit of 80 concurrent Git LFS connections are permitted. This limit can be increased or decreased if necessary by overriding the `throttle.resource.git-lfs=setting` in the [config properties](#) file. Bitbucket must be restarted for the change to take effect.

When a client makes a request to upload or download an object, and the request would exceed the maximum number of concurrent connections, a HTTP status 503 is returned, and this error is sent to the client:

```
The requested resource is busy and cannot service your request. Please try again later
```

The `atlassian-bitbucket.log` file will also contain an associated warning:

```
2016-01-01 18:00:00 WARN [http-nio-7990-exec-3] username @1HJ10F1x1115x417x1 0:0:0:0:0:0:1 "GET /rest /git-lfs/storage/myproject/myrepo/3a9219fde5bc436a2fc37cdd38bdb8478a210c7a49405dd9603ccdc95ed39613 HTTP /1.1" c.a.s.i.t.SemaphoreThrottleService A [git-lfs] ticket could not be acquired (0/80)
```

Git Virtual File System (GVFS)

Microsoft has developed an [open source project](#) called GVFS (Git Virtual File System) that helps with developer productivity when it comes to very large monolithic codebases in Git.

On this page:

GVFS virtualizes the file system underlying Git repositories on developers machines, making more efficient use of bandwidth, space and computation by only transferring and working with files that have been accessed.

The Atlassian Marketplace provides an [experimental app](#) for Bitbucket Data Center and Server that adds the server-side support needed to serve GVFS-enabled Git clients. It can be installed on any [supported server platform](#) and once installed, Bitbucket will handle GVFS protocol requests for any repository without any additional setup, from properly configured Git clients.

For more information about setting up GVFS on the client side, please see the [blog post and linked resources from Microsoft](#). As at October 2017, only Windows client support was available. [Sourcetree 2.0 and above for Windows is also GVFS aware](#) when used with a GVFS enabled client.

Like GVFS itself, the experimental app is relatively new, and unsupported. It is an experimental technology preview not intended for production use, though being a distinct code path it is unlikely to introduce side effects for non-GVFS users on a production instance. If you work on large repositories, we'd love for you to [take it for a spin](#) and send us your [feedback](#).

Enable SSH access to Git repositories

Administrators can enable SSH access to Git repositories in Bitbucket Data Center and Server. This allows users to:

- add their own SSH keys to Bitbucket
- use those SSH keys to secure Git operations between their computer and the Bitbucket instance.

Each user must [add their own SSH key pairs](#) to their account to be able to use SSH to access repositories.

Supported key types are DSA, RSA2, and Ed25519. Note that RSA1 is not supported. We've tested key sizes of 768, 1024, 2048, 4096 and 8192 bytes.

On this page:

- [Enabling SSH access](#)
- [SSH base URL](#)
- [When running Bitbucket behind a proxy](#)

Related pages:

- [Setting up SSH port forwarding](#)
- [Creating SSH keys](#)

Performance

Using SSH has performance implications. When users connect to Bitbucket using SSH the encryption of data adds to overall CPU usage. See [Scaling Bitbucket Server](#) for more information.

Security

To implement SSH authentication support, Bitbucket bundles a version of the [Apache Mina SSHD](#) server. The Bitbucket SSH server is not integrated with the SSH server on the host Bitbucket is running on, and it doesn't consider the users on the host when authenticating Bitbucket users.

To prevent security issues, the embedded SSH server has been locked down to allow execution of a small set of commands for Git hosting. The only commands that are supported are `git upload-pack`, `git receive-pack`, `git archive-pack` and `whoami` (a custom `whoami` implemented in Bitbucket, not the `whoami` command that exists on Linux). It is not possible to open an SSH shell using the embedded server to execute arbitrary commands on the server.

Enabling SSH access

To enable SSH access:

1. Go to  > **Server settings**.
2. Select **SSH enabled**.
3. Enter values for **SSH port** and **SSH base URL**, according to the information in the sections below.
4. Select **Save**.

 These options will only be available if the "Bitbucket Server - SSH" app is enabled. For instructions on how to enable this app on your instance, please refer to [Disabling and enabling apps](#).

 For Data Center installations, a load balancer setup is required for SSH. For instructions on how to install and configure your load balancer, refer to [Install Bitbucket Data Center](#).

SSH base URL

The **SSH base URL** is the base URL with which users can access the SSH push/pull/clone functionality of Bitbucket.

This is the base URL that Bitbucket will use when displaying SSH URLs to users. If you do not set this, it will default to the host that is set in **Bitbucket base URL**, with the port that SSH is listening on. See [Specify the Bitbucket base URL](#).

i For example, if the **SSH base URL** is not set and the **Bitbucket base URL** is `https://bitbucket.atlassian.com` and the SSH port is 7999, the SSH URL for the repository `Jira` in the project `Atlassian` will be `ssh://git@bitbucket.atlassian.com:7999/ATLASSIAN/jira.git`

If you set up [port forwarding](#), you will need to set the **SSH base URL** to the machine and port that is being forwarded to Bitbucket. However, you do not need to specify the port portion of the URL if the default SSH port (port 22) is being forwarded to Bitbucket.

i If the **SSH base URL** and **SSH port** configurations are modified in the global Server settings page, the configurations specified in the properties file will no longer be used.



Client



Bitbucket

| Port forwarding | SSH base URL | Bitbucket base URL | SSH port | Resulting SSH URL for a repo |
|-----------------|--|--|----------|--|
| Not set | Not set | <code>https://bitbucket.atlassian.com</code> | 7999 | <code>ssh://git@bitbucket.atlassian.com:7999/<projectname>/<reponame>.git</code> |
| Port 22 > 7999 | <code>https://bitbucket.atlassian.com</code> | <code>https://bitbucket.atlassian.com</code> | 7999 | <code>ssh://git@bitbucket.atlassian.com/<projectname>/<reponame>.git</code> |

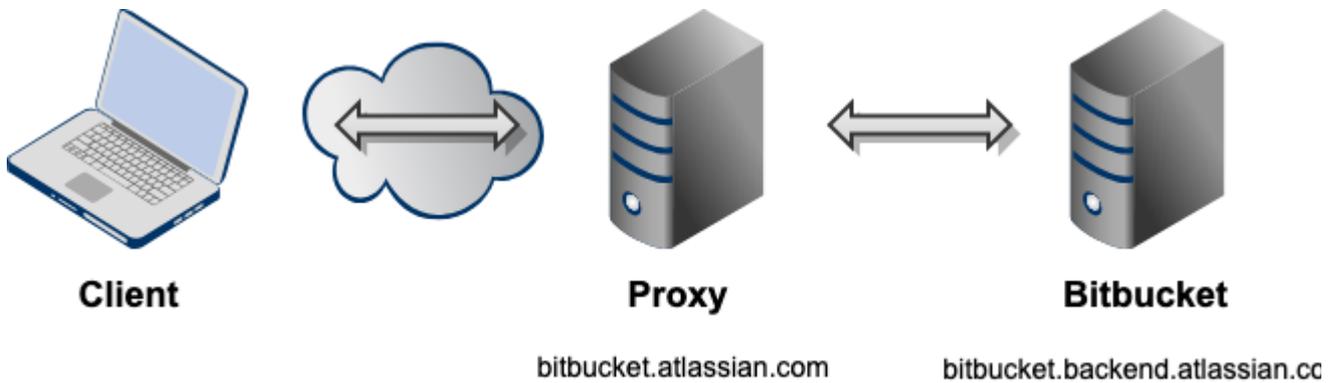
When running Bitbucket behind a proxy

If you run Bitbucket behind a http proxy such as Apache (e.g. as per our [instructions](#)), and if Apache runs on a different host, SSH will not be available on that host. Instead, you will need to set the SSH base URL to the machine Bitbucket is actually running on (and the URL should include the SSH port Bitbucket is serving from).

i For example, if the **SSH base URL** is set to `ssh://bitbucket.backend.atlassian.com:7999`, the SSH URL for the repository `Jira` in the project `Atlassian` will be `ssh://git@bitbucket.backend.atlassian.com:7999/ATLASSIAN/jira.git`

If you set up [port forwarding](#), you will need to set the **SSH base URL** to the proxy machine and port that is being forwarded to Bitbucket. However, you do not need to specify the port portion of the URL if the default SSH port (port 22) is being forwarded to Bitbucket.

i For example, if you set up port forwarding from your http proxy host, `bitbucket.atlassian.com`, port 22, to `bitbucket.backend.atlassian.com` port 7999, set the **SSH base URL** to `ssh://bitbucket.atlassian.com`. Then, the SSH URL for the repository `Jira` in the project `Atlassian` will be `ssh://git@bitbucket.atlassian.com/ATLASSIAN/jira.git`



| Port forwarding | SSH base URL | SSH port | Bitbucketbase URL | Resulting SSH URL for a repo |
|--|---|----------|--|--|
|  | <code>ssh://bitbucket.backend.atlassian.com:7999</code> | 7999 | <code>https://bitbucket.backend.atlassian.com</code> | <code>ssh://git@bitbucket.backend.atlassian.com:7999/<projectname>/<reponame>.git</code> |
|  Port 22>7999 | <code>ssh://bitbucket.atlassian.com</code> | 7999 | <code>https://bitbucket.backend.atlassian.com</code> | <code>ssh://git@bitbucket.atlassian.com/<projectname>/<reponame>.git</code> |
|  Port 44>7999 | <code>ssh://bitbucket.atlassian.com:44</code> | 7999 | <code>https://bitbucket.backend.atlassian.com</code> | <code>ssh://git@bitbucket.atlassian.com:44/<projectname>/<reponame>.git</code> |

Setting up SSH port forwarding

There are two scenarios where you might want to set up port forwarding in Bitbucket Data Center and Server: to remove port numbers from your SSH URLs or if Bitbucket is running behind a reverse proxy on a separate machine

On this page:

- [Remove port numbers from your SSH URLs](#)
- [Bitbucket is running behind a reverse proxy on a separate machine](#)

How to set up port forwarding

- [HAProxy](#)
- [Installing HAProxy](#)
- [Configuring HAProxy](#)

Configuring the SSH base URL

Remove port numbers from your SSH URLs

Bitbucket listens for SSH connections on port 7999 by default.

Your users will need to include the port in the URL they use to clone from Bitbucket, for example:

```
git clone ssh://git@bitbucket.mycompany.com:7999/PROJECT/repo.git
```

Rather than have the port number in the URL, you may wish to set up port forwarding so that connections to the default SSH port are forwarded to the port Bitbucket is listening on (e.g. you could forward port 22 to port 7999).

This would allow your users to use a URL without a port number in it, like this:

```
git clone ssh://git@bitbucket.mycompany.com/PROJECT/repo.git
```

Bitbucket is running behind a reverse proxy on a separate machine

You may be following our instructions for [setting up Bitbucket behind an Apache front-end](#).

In this case, your users may not be able to access Bitbucket directly for SSH connections, or if they can, you may wish to make the SSH and HTTPS URLs consistent.



For example, if you have the above topology, without port forwarding (and assuming the default port of 7999), your users will need to clone Bitbucket directly from the backend, like this:

```
git clone ssh://git@bitbucket.backend.atlassian.com:7999/PROJECT/repo.git
```

In your network, the `bitbucket.backend.atlassian.com` machine may not be accessible directly, or you may want the URL to be consistent with the HTTPS URL of `https://bitbucket.atlassian.com/scm/PROJECT/repo.git`.

In this case, you need to set up port forwarding on the `bitbucket.atlassian.com` machine to accept connections and forward them to port 7999 on the `bitbucket.backend.atlassian.com` machine.

How to set up port forwarding

HAProxy

Atlassian recommends the use of [HAProxy](#) for forwarding SSH connections through to Bitbucket.

HAProxy is [supported](#) on Linux, Solaris and FreeBSD.

i HAProxy is not an Atlassian product, so Atlassian does not guarantee to provide support for its configuration. This section is provided for your information only use it at your own risk. We recommend that you refer to the [HAProxy documentation](#).

Installing HAProxy

Your operating system may support installing HAProxy using its system package manager, such as `apt-get`, `yum` or `rpm`. This will be the easiest way.

Alternatively, you may build HAProxy yourself and install it.

1. Download the latest version of HAProxy from <http://haproxy.1wt.eu/#down>.
2. Extract the archive and cd into the directory:

```
tar xzvf haproxy-1.4.21.tar.gz
cd haproxy-1.4.21
```

3. Read the instructions in the README for how to build on your system. This is generally quite simple - on a Linux 64 bit 2.6 Kernel, the command is:

```
make TARGET=linux26 ARCH=x86_64
```

4. If it completes successfully, install it following the instructions in the README:

```
sudo make install
```

Configuring HAProxy

HAProxy is extremely powerful - it is designed as a HTTPS load balancer, but also can serve as a port forwarder for `ssh`.

The full documentation for version 1.4 is [here](#). More documentation is available on the [HAProxy website](#).

An example simple configuration is as follows:

```
global
    daemon
    maxconn 10000

defaults
    timeout connect 500s
    timeout client 5000s
    timeout server 1h

frontend sshd
    bind *:7999
    default_backend ssh
    timeout client 1h

backend ssh
    mode tcp
    server localhost-bitbucket-ssh 127.0.0.1:7999
```

The above configuration will listen on port 7999 (indicated by the `bind` directive) on all network interfaces. As indicated by the `server` directive, traffic is forwarded to 127.0.0.1, port 7999. You will need to replace 127.0.0.1 with the IP address of the machine running Bitbucket.

You can check your configuration by running:

```
haproxy -f haproxyconf.txt -c
```

To run haproxy, simply start it using

```
haproxy -f haproxyconf.txt
```



If you use HAProxy to additionally proxy HTTP traffic, ensure that the running mode configuration is set to `http`:

```
backend http
    mode http
    bind *:80
    server localhost-bitbucket-http 127.0.0.1:7990
```

Using the default SSH port

You can configure HAProxy to listen on the default SSH port instead, so that the port does not need to be specified in the clone URL.

By default, the normal ssh daemon is running on port 22. You have several options:

- Configure HAProxy to listen on an alternate port as in the previous example.
- Configure multiple network interfaces on the physical machine and force the default ssh daemon to listen on all but the interface for accessing Bitbucket. Configure HAProxy to only listen on that interface.
- Move the default ssh daemon to listen on another port and let HAProxy bind on port 22.

We do not provide instructions on the last two options, except for how to configure HAProxy.

Use the same configuration as the last example, but change the bind port to 22, e.g.

```
...
frontend sshd
    bind *:22
...
```

You will have to run this configuration as the `root` user, using `sudo`, because it specifies a port to listen on that is less than 1024.

```
sudo haproxy -f haproxyconf.txt
```

Configuring the SSH base URL

Once port forwarding is set up, you will need to configure the SSH base URL in Bitbucket so that the clone urls presented in Bitbucket indicate the correct host and port to clone from. See the [SSH base URL](#) section in [Enable SSH access to Git repositories](#).

Use diff transcoding

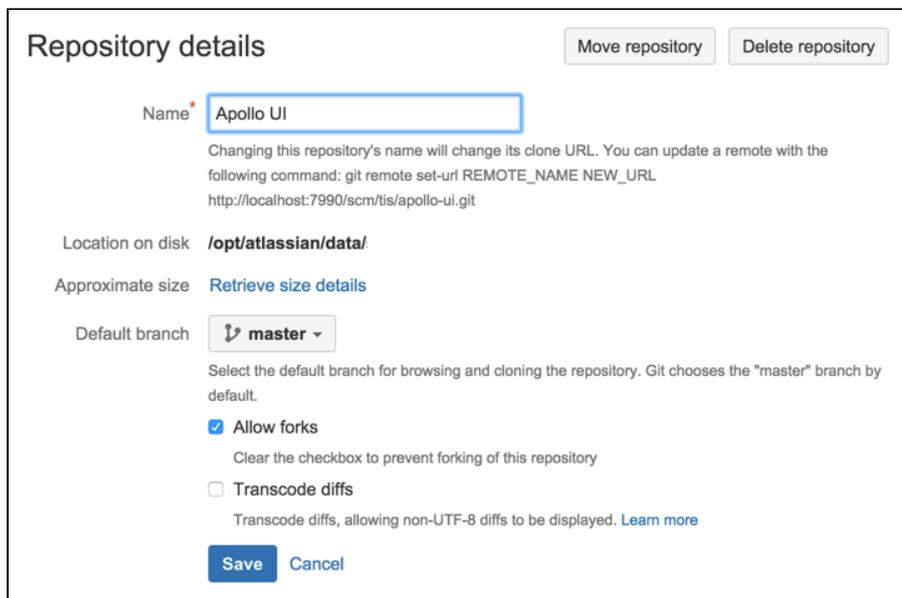
Bitbucket Data Center and Serversupports transcoding for diffs. This allows it to convert files in encodings like EUC-JP, GB18030 and UTF-16 to UTF-8, so they are processed correctly by `git diff`, which only supports UTF-8. Similar transcoding has been applied to the Bitbucket source view since it was released, so this change brings the diff view in line with the source view. Diff transcoding is applied to commit and pull request diffs, as well as the diff-to-previous view.

 Git for Windows, formerly known as msysgit, has known issues with Unicode paths. Diff transcoding works on all supported versions of Git for Windows, but 1.8.0 or higher is required to support Unicode paths.

Enabling diff transcoding

Diff transcoding must be explicitly enabled for each repository (unlike source view transcoding, which is always performed).

Repository administrators can enable diff transcoding on the repository settings page:



The screenshot shows the 'Repository details' page for a repository named 'Apollo UI'. The page includes buttons for 'Move repository' and 'Delete repository'. The 'Name' field is set to 'Apollo UI'. Below the name, there is a note: 'Changing this repository's name will change its clone URL. You can update a remote with the following command: `git remote set-url REMOTE_NAME NEW_URL` `http://localhost:7990/scm/tis/apollo-ui.git`'. The 'Location on disk' is '/opt/atlassian/data/'. The 'Approximate size' is 'Retrieve size details'. The 'Default branch' is 'master'. There are two checkboxes: 'Allow forks' (checked) and 'Transcode diffs' (unchecked). The 'Transcode diffs' checkbox has a tooltip that says: 'Transcode diffs, allowing non-UTF-8 diffs to be displayed. [Learn more](#)'. At the bottom, there are 'Save' and 'Cancel' buttons.

Performance and scaling

There's a performance consideration with transcoding. It is implemented using Git's `textconv` support, so using it adds overhead to displaying diffs. Where possible, the best approach, given `git` only supports UTF-8 content, is to use UTF-8 encoding so that transcoding is not necessary. In repositories without non-UTF-8 content, diff transcoding should be left disabled. Other encodings are often a necessity, however, and for repositories containing such content enabling diff transcoding allows using the full range of Bitbucket features.

When transcoding is enabled, `git diff` writes the before and after blobs to temporary files and invokes the `textconvscript` once for each file. The script Bitbucket installs uses Perl to send a request back to Bitbucket with the path to each temporary file. Bitbucket then opens each file, detects the encoding using the same algorithm the source view uses, converts the file to UTF-8 and streams it out for `git diff` to use. After `git diff` has invoked the `textconvscript` the temporary files it created are deleted.

Writing the blobs to disk, starting Perl and calling back into Bitbucket are all overhead processing compared to performing a diff without transcoding. How much overhead that is varies by the size of the diff. When nominally-sized files containing two or three thousand lines or less are being compared the overhead is miniscule, under 50 milliseconds on an average server. However, when comparing larger files the overhead can result in a noticeable delay displaying the diff.

Change the port Bitbucket listens on

You can change the port that Bitbucket Data Center and Server listens on from the default '7990' to a different value if another application is already running on that port.

You can use [netstat](#) to identify free ports on your machine.

Related pages:

- [Specify the Bitbucket base URL](#)
- [Proxy and secure Bitbucket](#)
- [Bitbucket Server config properties](#)

To change the port Bitbucket listens on

1. Navigate to your [home directory](#).
2. Open the `shared/bitbucket.properties` file, add the `server.port` property (or edit it if a line for the `server.port` property already exists), and set the value to the port number Bitbucket will run on. For example, to set to the port to 8080 you would add:

```
server.port=8080
```

Then save the file.

3. Restart Bitbucket so the change takes effect. See [Start and stop Bitbucket](#).

Important considerations

If you are using a firewall, ensure that it is configured to allow HTTP or HTTPS traffic over the connector port you have chosen.

If you are running Bitbucket on a Linux server and want to bind to privileged ports (those below 1024, for example port 80), you will need to start Bitbucket as root to successfully bind to the port. Or, you can bind Bitbucket to a port over 1024 and then configure iptables to redirect traffic from port 80 to the higher port.

Lockout recovery process

This page describes how to recover administrator access for Bitbucket Data Center and Server 5.X+, and later. For releases prior to that, please refer to the documentation specific to that version.

As an administrator, you may find yourself locked out of Bitbucket and unable to log in. This situation can arise when all users are managed externally from Bitbucket, and it becomes unable to access those user directories for some reason, including:

- The external user directory server is not accessible (because the network is down, or the directory is down, or the directory has been moved to another IP address).
- Users are managed within a Jira application and the Application Link from Bitbucket to a Jira application has been accidentally deleted.
- The admin password has been forgotten or lost.
- The admin account is shaded by a remote account in an LDAP or Jira application that is connected to Bitbucket but which is unavailable.

The lockout recovery process is:

1. Edit the `<Bitbucket installation directory>\bin_start-webapp.sh` file (or `_start-webapp.bat` on Windows) and add the `"-Datlassian.recovery.password=temporarypassword"` value to the `JVM_SUPPORT_RECOMMENDED_ARGS` property. The property value must be non-blank, and should look like this when you've done that:

```
# Occasionally Atlassian Support may recommend that you set some specific JVM arguments.
# You can use this variable to do that. Simply uncomment the below line and add any required
# arguments. Note however, if this environment variable has been set in the environment of the
# user running this script, uncommenting the below will override that.
#
JVM_SUPPORT_RECOMMENDED_ARGS=-Datlassian.recovery.password=temporarypassword
```

Here we are using "temporarypassword", but you should use your own value.

2. Start Bitbucket **manually** by running `<Bitbucket installation directory>\bin\start-bitbucket.sh` (or `start-bitbucket.bat` on Windows).
3. Log in using the 'recovery_admin' username and the temporary password specified in Step 1.
4. Repair your configuration. We strongly recommend that you do not perform other actions while Bitbucket is in recovery mode.
5. Confirm your ability to log in with your usual admin profile.
6. Shut down Bitbucket, remove the `atlassian.recovery.password` argument from `_start-webapp.sh` (or `_start-webapp.bat` on Windows), and restart Bitbucket as usual.

Proxy and secure Bitbucket

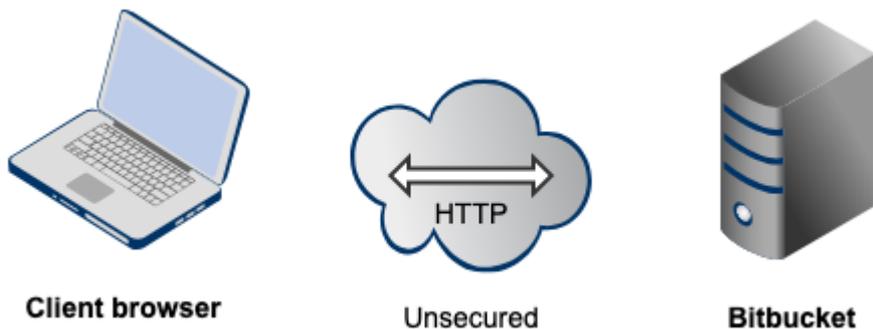
This page provides an overview of some common network topology options for running Bitbucket Data Center and Server, including running Bitbucket behind a reverse proxy and securing access to Bitbucket by using HTTPS (HTTP over SSL).

Note that Bitbucket does not need to run behind a web server - it is capable of serving web requests directly using the bundled Tomcat application server. On this page, 'connecting to Bitbucket' really means connecting to Tomcat, which is used to serve Bitbucket content.

Connecting to Bitbucket directly over HTTP

Connecting directly to Bitbucket (that is, Tomcat) is the default install configuration, as described on [Getting started](#).

When set up this way, the user accesses Bitbucket directly over HTTP, without using SSL - all communication between the user's browser and Bitbucket will be unsecured.



You may also wish to consider the following:

- Bitbucket, by default, will listen for requests on port 7990 - this [port can be changed](#) if required.
- The address with which to access Bitbucket, by default, will be `http://<computer name>:7990`. Change the [base URL](#) if required.
- You can [set the context path](#) for Bitbucket if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).

Securing access to Bitbucket using HTTPS

Access to Bitbucket can be secured by enabling HTTPS (HTTP over SSL) for the Tomcat application server that is bundled with Bitbucket. You should consider doing this, and making secure access mandatory, if Bitbucket will be internet-facing and usernames, passwords and other proprietary data may be at risk.

When set up in this way, access to Bitbucket is direct, and all communication between the user's browser and Bitbucket will be secured using SSL.

See [Secure Bitbucket with Tomcat using SSL](#) for configuration details.

Using a reverse proxy for Bitbucket

On this page

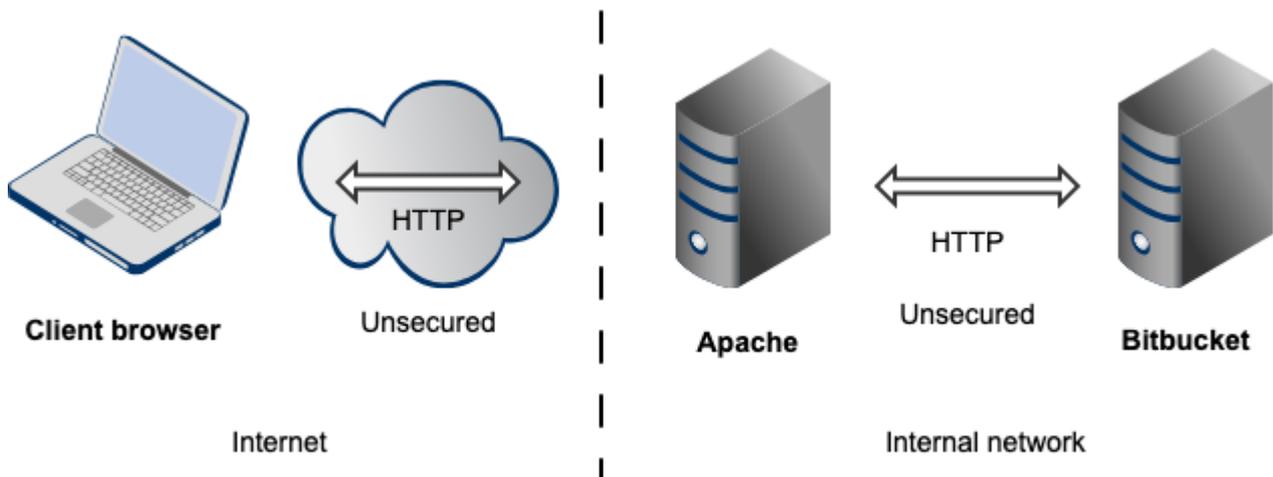
- [Connecting to Bitbucket directly over HTTP](#)
- [Securing access to Bitbucket using HTTPS](#)
- [Using a reverse proxy for Bitbucket](#)
- [Securing a reverse proxy using HTTPS](#)

You can run Bitbucket behind a reverse proxy, such as Apache HTTP Server. You may wish to do this if you want to:

- use a different port number to access Bitbucket, particularly if you are [Integrating Jira Cloud with Bitbucket](#).
- use a different context path to access Bitbucket

When set up this way, external access to Bitbucket is via a reverse proxy, without using SSL. All communication between the user's browser and Apache, and so Bitbucket, will be unsecured, but users do not have direct access to Bitbucket. An example scenario is where Apache provides a gateway through which users outside the firewall can access Bitbucket.

See [Integrate Bitbucket with Apache HTTP Server](#) for configuration details.



Note that:

- Bitbucket, by default, will listen for requests on port 7990 this port can be changed if required.
- Bitbucket (Tomcat) needs to know the URL (proxy name) that Apache serves.
- The address with which to access Bitbucket will be `http://<proxy name>:7990`. Change the [base URL](#) if required.
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket.
- You can [set the context path](#) for Bitbucket if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).

Securing a reverse proxy using HTTPS

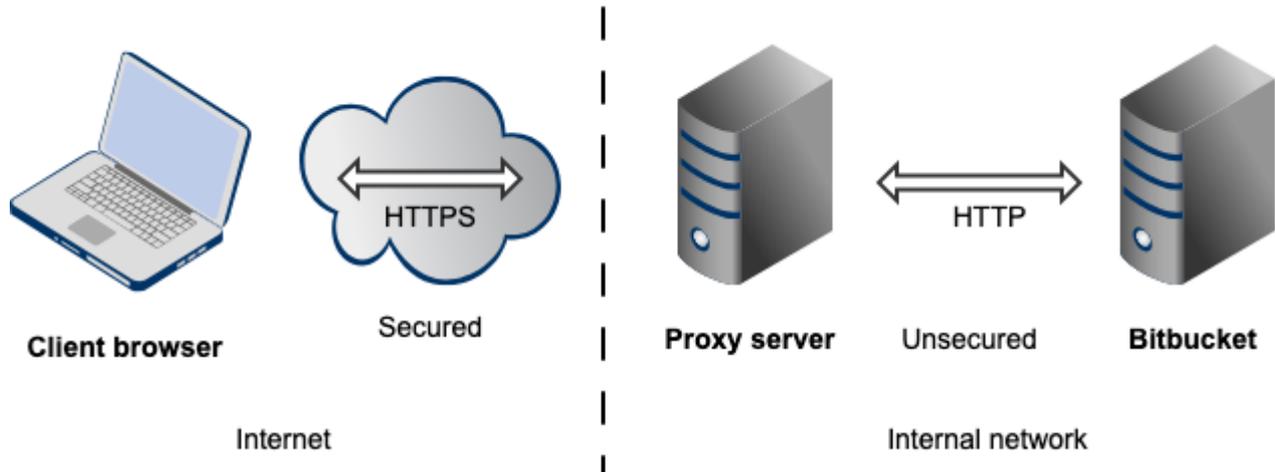
You can run Bitbucket behind a reverse proxy, such as Apache HTTP Server or nginx, that is secured using HTTPS (HTTP over SSL). You should consider doing this, and making secure access mandatory, if usernames, passwords and other proprietary data may be at risk. An example scenario is where Apache HTTP Server provides a gateway through which users outside the firewall can access Bitbucket.

When set up in this way, external access to Bitbucket is via a reverse proxy, where external communication with the proxy uses HTTPS. All communication between the user's browser and the reverse proxy will be secured, whereas communication between the proxy and Bitbucket will not be secured (it doesn't use SSL).

See the following pages for configuration details:

- [Secure Bitbucket with Apache using SSL](#)
- [Secure Bitbucket behind nginx using SSL](#)

- [Secure Bitbucket behind HAProxy using SSL](#)



Note that:

- The reverse proxy (for example, Apache) will listen for requests on port 443.
- Bitbucket, by default, will listen for requests on port 7990. Bitbucket (Tomcat) needs to know the URL (proxy name) that the proxy serves.
- The address with which to access Bitbucket will be `https://<proxyName>:<proxyPort>/<context path>`, for example `https://mycompany.com:443/bitbucket`
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket.
- Bitbucket (Tomcat) should be configured to refuse requests on port 7990 and to redirect those to the proxy on port 443.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).
- It would be possible to set up an SSL connection between the proxy server and Tomcat (Bitbucket), but that configuration is very unusual, and not recommended in most circumstances.
- Incidentally, note that Bitbucket 4.0 and later versions do not support `mod_auth_basic`.

Secure Bitbucket with Tomcat using SSL

This page describes how to enable HTTPS (HTTP over SSL) access for Apache Tomcat, the application server shipped with Bitbucket Data Center and Server.

Why should I secure Bitbucket using SSL?

You should consider doing this, and making secure access mandatory, if Bitbucket will be internet-facing, which could potentially put usernames, passwords and other proprietary data at risk.

There are other network topology options for running Bitbucket, including running Bitbucket behind a reverse proxy. For an overview of some common options, see [Proxy and secure Bitbucket](#).

When Bitbucket is set up using these instructions, access to Bitbucket is direct, and all communication between a user's browser and Bitbucket will be secured using SSL.

Before you begin

The configuration described here results in a scenario where:

- Bitbucket would listen for requests on port 8443; [the port can be changed](#), if required.
- Bitbucket would be available at `https://<computer name>:8443`; [the base URL can be changed](#), if required.
- As a result, any existing [links with other applications](#) would need to be reconfigured using this new URL for Bitbucket.
- You can [set the context path](#) for Bitbucket if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).

These instructions are for reference use only

Atlassian applications allow the use of SSL within our products, however Atlassian Support does not provide assistance for configuring it. Consequently, Atlassian **cannot guarantee providing any support for it**.

If you require assistance with conversions of certificates is required, consult the vendor who provided the certificate. Atlassian Support will refer SSL-related support to the issuing authority for the certificate.

Also, these instructions are intended for administrators setting up Bitbucket for a small team.

On this page:

- [Before you begin](#)
- [Generate a self-signed certificate](#)
- [Configure HTTPS in bitbucket.properties](#)
- [Export the self-signed certificate](#)
- [Request a CA certificate](#)
- [Troubleshooting](#)

Related pages:

- [Integrate Bitbucket with Apache HTTP Server](#)
- [Secure Bitbucket with Apache using SSL](#)

Generate a self-signed certificate

Self-signed certificates are useful if you require encryption but do not need to verify the identity of the requesting website. In general, you might use a self-signed certificate on a test environment and on internal corporate networks (intranets).

Because the certificate is not signed by a certificate authority (CA), users may receive a message that the site is not trusted and may have to perform several steps to accept the certificate before they can access the site. This usually occurs the first time they access the site.

This approach to creating a certificate uses Java's [keytool](#), there are other tools for generating certificates are available.

 If you are setting up a production instance of Bitbucket, we strongly recommend [using a CA certificate](#) (described below), rather than using a self-signed certificate.

To generate a self-signed certificate using Java's `keytool` utility

1. Create a new `.keystore` file in the Bitbucket home directory.
 - a. Log in with the user account that has access to Bitbucket home directory (or use `sudo` alternatively).
 - b. Run the following command:

| | |
|---------------------|--|
| Windows | <pre>"%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA -keystore <Bitbucket home>\shared\config\ssl-keystore</pre> |
| Linux, macOS | <pre>\$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA -keystore <Bitbucket home>/shared/config/ssl-keystore</pre> |

 The default keystore Bitbucket uses is `<Bitbucket home directory>/shared/config/ssl-keystore`. You can specify different keystore location using the `keystore` parameter. But, in this case, you will have to include it in the `bitbucket.properties` file as described in the [configuration](#) section.

2. When asked for **password** (the first of two passwords you will set, which must be identical):
 - Specify the password you want to use for the certificate (private key).
 - The default password Tomcat expects is 'changeit'. If you use any other value you must specify it in the `<Bitbucket home directory>/shared/bitbucket.properties` file.
 - Make a note of the password as you will need it again in the next step.

3. Follow the prompts to specify your **name**, **organization** and **location**. This information is used to construct the X.500 Distinguished Name (DN) of the entity.

The CN ("What is your first and last name?") *must match the fully-qualified hostname of the server running Bitbucket*, otherwise Tomcat will not be able to use the certificate for SSL. This is the name you would type in your web browser after 'http://' (no port number) to access your Bitbucket installation.

4. When asked for the **password** for 'tomcat' (the alias you entered in the `keytool` command above), press the 'Enter' key. This specifies that your keystore entry will have the **same password** as your private key. If needed, you can specify different key password.

Configure HTTPS in `bitbucket.properties`

The default connection for Bitbucket uses an unsecured HTTP connection. Depending on the type of access you require, you can change Bitbucket's default connector (or add additional connectors) to require HTTPS-only access to your instance.

- To configure HTTPS-only access, you would replace the default connector to require HTTPS-only access.
- To redirect HTTP requests to a secure HTTPS connection, you would configure the default connector to redirect to an additional, secured connector.

There are a number of ways to configure secure connections depending on your requirements. These instructions describe how to replace Bitbucket's default connector with a secure, HTTPS-only connection by adding configuration properties to the [bitbucket.properties](#) file.

To configure HTTPS-only access to Bitbucket

1. Navigate to your [home directory](#).

a. Open the `shared/bitbucket.properties` file and add these properties:

```
server.port=8443
server.ssl.enabled=true
server.ssl.key-store=/path/to/keystore/bitbucket.jks
server.ssl.key-store-password=<password value>
server.ssl.key-password=<password value>
```

By adding these properties to your `bitbucket.properties` file you override the default connector values.

| Property | What it does |
|---|---|
| <code>server.port=8443</code> | Enables SSL access on port 8443 (the default for HTTPS is 443, but 8443 is used here instead of 443 to avoid conflicts) |
| <code>server.scheme=https</code> | Specifies a required URL scheme. |
| <code>server.ssl.enabled=true</code> | Specifies whether SSL enabled (true) or not (false). Setting this to true <i>automatically</i> defaults <code>server.scheme=https</code> and <code>server.secure=true</code> ; they do not need to be set explicitly. |
| <code>server.ssl.client-auth=need</code> | Specifies whether client authentication is wanted ("want") or needed ("need"). Requires a trust store. |
| <code>server.ssl.protocol=TLSv1.2</code> | We recommend requiring TLS 1.2. If you have clients that don't support TLS 1.2, don't include this property. The default is "TLS". |
| <code>server.ssl.key-alias=<alias></code> | Specifies SSH key values. Only required if non-default alias is used (default value is "tomcat"). |

| | |
|---|--|
| <pre>server.ssl.key-store=/path/to/keystore/bitbucket.jks</pre> | <p>Specifies where the keystore is on your filesystem. Only required if you created the keystore in non-default location (other than <Bitbucket home directory>/shared/config/ssl-keystore).</p> <p>On Windows you should use forward-slashes in the path, for example:</p> <pre>server.ssl.key-store=c:/Atlassian/ApplicationData/Bitbucket/bitbucket.jks</pre> <p>To use backslashes, they must be doubled (e.g. "C:\\Atlassian\\...")</p> |
| <pre>server.ssl.key-store-password=<password value></pre> | <p>Specifies your keystore password. Only required if your keystore password is anything other than the default ("changeit").</p> |
| <pre>server.ssl.key-password=<password value></pre> | <p>Specifies your key password. Only required if your key password is anything other than the default ("changeit").</p> |
| <pre>server.ssl.key-store-type=jks</pre> | <p>Specifies keystore type. Only required if your keystore is anything other than the default ("jks"). Could be "pkcs12" for certificates provided by CA.</p> |

If you created the keystore somewhere else on the filesystem, add the `keystoreFile` attribute to the connector tag as well. If your keystore password is anything other than "changeit", add `addKeystorePass="<password value>"` to the `bitbucket.properties` file.

2. Then save the file.
3. [Start \(or re-start\) Bitbucket.](#)
4. Access Bitbucket at `https://localhost:8443/` in your browser.

Export the self-signed certificate

If you need to export a certificate from the keystore (for example, to be able to import it into another keystore), you should use the following commands.

To export the self-signed certificate, run this command

| | |
|---------------------|---|
| Windows | <pre>"%JAVA_HOME%\bin\keytool" -export -alias tomcat -file file.cer -keystore <Bitbucket home directory>\shared\config\ssl-keystore</pre> |
| Linux, macOS | <pre>\$JAVA_HOME/bin/keytool -export -alias tomcat -file file.cer -keystore <Bitbucket home directory>/shared/config/ssl-keystore</pre> |

Request a CA certificate

Digital certificates are issued by a Certification Authority (CA) to verify your website represents your company. When running Bitbucket in a production environment, you need a certificate issued by a CA, such as [VeriSign](#), [DigiCert](#) or [Thawte](#). The instructions below are adapted from the [Tomcat documentation](#).

First, you will generate a local certificate and create a certificate signing request (CSR) based on that certificate. You then submit the CSR to your chosen certificate authority. The CA will use that CSR to generate a certificate for you.

1. Use Java's `keytool` utility to generate a local self-signed certificate, as described in the [section above](#).
2. Use the `keytool` utility to generate a CSR. If needed, replace the keystore file name with the path to and filename of the `.keystore` file you generated for your local certificate:

| | |
|---------------------|---|
| Windows | <code>"%JAVA_HOME%\bin\keytool" -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <Bitbucket home>\shared\config\ssl-keystore</code> |
| Linux, macOS | <code>\$JAVA_HOME/bin/keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore <Bitbucket home>/shared/config/ssl-keystore</code> |

3. Submit the generated file called `certreq.csr` to your chosen certificate authority. Refer to the documentation on the CA's website to find out how to do this.
4. The CA will send you a certificate.
5. Import the new certificate into your local keystore. Assuming your certificate is called `file.cer`, run this command to add the certificate to the keystore:

| | |
|---------------------|---|
| Windows | <code>"%JAVA_HOME%\bin\keytool" -import -alias tomcat -file file.cer -keystore <Bitbucket home>\shared\config\ssl-keystore</code> |
| Linux, macOS | <code>\$JAVA_HOME/bin/keytool -import -alias tomcat -file file.cer -keystore <Bitbucket home>/shared/config/ssl-keystore</code> |

Troubleshooting

Here are some troubleshooting tips if you are using a self-signed key created by `keytool`, or a CA certificate, as described above.

When you enter "https://localhost:8443/" in your browser, if you get a message such as "Cannot establish a connection to the server at localhost:8443," look for error messages in your `<Bitbucket home>/shared/logs/atlassian-bitbucket.logfile`. Here are some possible errors with explanations.

SSL + Apache + IE problems

Some people have reported errors when uploading attachments over SSL using Internet Explorer. This is due to an IE bug, and can be fixed in Apache by setting:

```
BrowserMatch ".MSIE." \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
```

[Google](#) has plenty more on this.

Can't find the keystore

```
java.io.FileNotFoundException: /var/atlassian/application-data/bitbucket/shared/config/ssl-keystore (No such file or directory)
```

This indicates that Tomcat cannot find the keystore. By default Bitbucket expects keystore to be available in the file `<Bitbucket home directory>/shared/config/ssl-keystore`. If you use non-default location, you will need to specify where the keystore file is in `<Bitbucket home directory>/shared/bitbucket.properties`. Add the following attribute to the connector tag you uncommented:

```
server.ssl.key-store=${bitbucket.shared.home}/config/ssl-keystore
```

Additionally, please, make sure you are running Bitbucket as the user who has permission to read the keystore file.

Incorrect password, or passwords don't match

```
java.io.IOException: Keystore was tampered with, or password was incorrect
```

You likely used a different password for your keystore file than the default, which is "changeit", or specified incorrect key password in the `bitbucket.properties` file.

```
java.io.IOException: Cannot recover key
```

You likely used a different password for your key than the default, which is "changeit", or specified incorrect key password in the `bitbucket.properties` file.

To check if either of these problems is the case, locate the `<Bitbucket home directory>/shared/bitbucket.properties` file, and verify these two properties have matching values.

```
server.ssl.key-store-password=<password value>  
server.ssl.key-password=<password value>
```

Wrong certificate

```
javax.net.ssl.SSLException: No available certificate corresponds to the SSL cipher suites which are enabled.
```

By default, keytool creates certificates that use DSA public key. Make sure you are using `-keyalg RSA` option when generating certificates.

If the Keystore has more than one certificate, Tomcat will use the "tomcat" certificate unless otherwise specified in the `<Bitbucket home directory>/shared/bitbucket.properties` file.

Verify this attribute is set in the `bitbucket.properties` file.

```
server.ssl.key-alias=tomcat
```

Using Apache Portable Runtime (APR)

Bitbucket does not support APR at the moment.

Enabling client authentication

To enable client authentication in Tomcat, ensure that the value of the `server.ssl.client-auth` attribute in your `bitbucket.properties` file is set to "need".

```
server.ssl.client-auth=need
```

Wrong certificate type

If the certificate from the CA is in PKCS12 format, add the `server.ssl.key-store-type` attribute in your `bitbucket.properties` file with the "PKCS12" value.

```
server.ssl.key-store=${bitbucket.shared.home}/config/ssl-keystore
server.ssl.key-store-password=changeit
server.ssl.key-store-type=pkcs12
```

Alternatively, you can import pkcs12 certificate into default JCS keystore using the following command:

```
keytool -importkeystore -srckeystore <Bitbucket home>/shared/config/certificate.pfx -srcstoretype pkcs12
-srcstorepass exportpass -srcalias <Source keystore alias> -destkeystore <Bitbucket home>/shared/config
/ssl-keystore -deststoretype jks -deststorepass changeit -destalias tomcat
```

Certificate chain is incomplete

If the root certificate and intermediary certificate(s) aren't imported into the keystore before the entity/domain certificate, you will see the following error:

```
[root@dev atlas]# $JAVA_HOME/bin/keytool -import -alias tomcat -file my_entity_cert.crt
Enter keystore password:
keytool error: java.lang.Exception: Failed to establish chain from reply
```

Most likely, the CA sent a compressed file containing several certificates. The import order matters so you must import the root certificate first, followed by one or many intermediate certificates, followed lastly by the entity/domain certificate. There are many resources online that provide guidance for [certificate installation for Tomcat \(Java-based\) web servers using keytool](#).

Integrate Bitbucket with Apache HTTP Server

When opened in a viewport, the user will be redirected to: [Proxying Atlassian server applications with Apache HTTP Server \(mod_proxy_http\)](#).

This page explains how to establish a network topology in which Apache HTTP Server acts as a [reverse proxy](#) for Bitbucket Data Center and Server. Typically, such a configuration would be used when Bitbucket is installed in a protected zone 'behind the firewall', and Apache HTTP Server provides a gateway through which users outside the firewall can access Bitbucket. You may wish to do this if you want to:

- use a different port number to access Bitbucket
- use a different context path to access Bitbucket

Be aware that Bitbucket does not need to run behind a web server, since it is capable of serving web requests directly; to secure Bitbucket when run in this way see [Secure Bitbucket with Tomcat using SSL](#). For an overview of other network topology options, see [Proxy and secure Bitbucket](#). Otherwise, if you want to install Bitbucket in an environment that incorporates Apache HTTP Server, this document is for you.

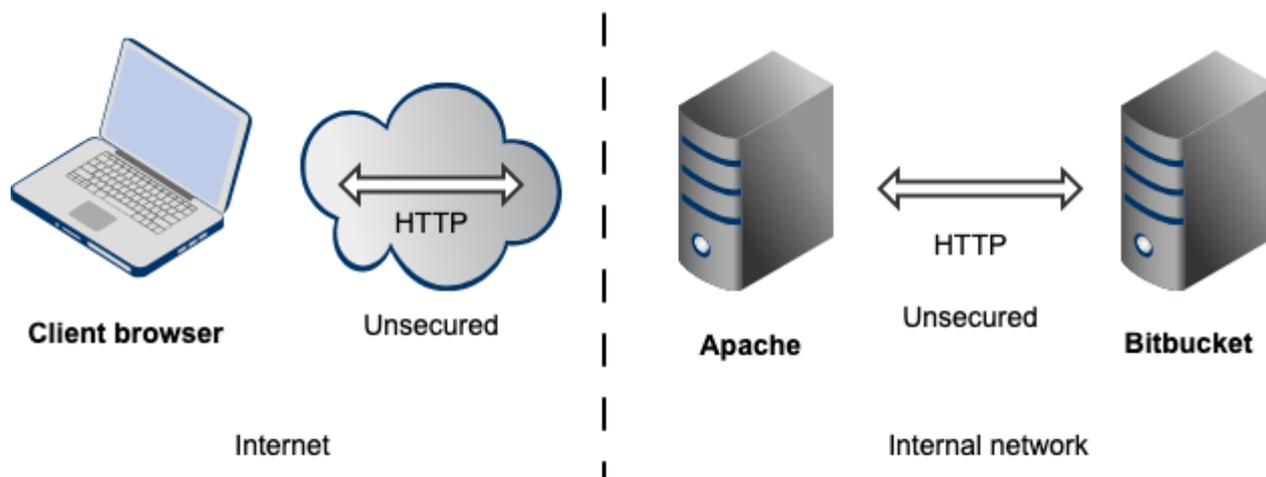
When Bitbucket is set up following the instructions on this page, external access to Bitbucket uses a reverse proxy, without using SSL. All communication between the user's browser and Apache, and so Bitbucket, will be unsecured, but users do not have direct access to Bitbucket.

On this page:

- [About using Apache software](#)
- [Step 1: Configure the Tomcat Connector](#)
- [Step 2: Change Bitbucket's base URL](#)
- [Step 3 \(optional\): Set a context path for Bitbucket](#)
- [Step 4: Enable mod_proxy and mod_proxy_http in Apache HTTP Server](#)
- [Step 5: Configure mod_proxy to map requests to Bitbucket](#)
- [Step 6: Configure mod_proxy to disable forward proxying](#)
- [Step 7: Allow proxying to Bitbucket from everywhere](#)
- [Step 8 \(optional\): Configure Apache HTTP Server for SSL](#)
- [A note about application links](#)
- [Troubleshooting](#)

Related pages:

- [Secure Bitbucket with Apache using SSL](#)
- [Secure Bitbucket with Tomcat using SSL](#)



Note that:

- Bitbucket, by default, will listen for requests on port 7990 this port can be changed if required.
- Bitbucket (Tomcat) needs to know the URL (proxy name) that Apache serves.
- The address with which to access Bitbucket will be `http://<proxy name>:7990`. Change the [base URL](#) if required.
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket.
- You can [set the context path](#) for Bitbucket if you are running another Atlassian application, or Java web application, at the same hostname and context path as Bitbucket.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).

About using Apache software

This section has general information pertaining to the use of [Apache HTTP Server](#) and [Apache Tomcat](#). It is important that you read this section before proceeding to the steps that follow.

Configuring Tomcat 7

The Bitbucket distribution includes an instance of Tomcat 7, the configuration of which is determined by the contents of the `<Bitbucket home directory>/shared/server.xml` file. Note that any changes that you make to the `server.xml` file will only be effective upon starting or re-starting Bitbucket.

You may find it helpful to refer to the [Apache Tomcat 7.0 Proxy Support HowTo](#) page.

Configuring Apache HTTP Server

⚠ Since Apache HTTP Server is not an Atlassian product, Atlassian does not guarantee to provide support for its configuration. You should consider the material on this page to be for your information only; use it at your own risk. If you encounter problems with configuring Apache HTTP Server, we recommend that you refer to the [Apache HTTP Server Support](#) page.

Note that Bitbucket 2.10 and later versions do not support `mod_auth_basic`.

You may find it helpful to refer to the [Apache HTTP Server Documentation](#), which describes how you can control Apache HTTP Server by changing the contents of the `httpd.conf` file. The section on [Apache Module mod_proxy](#) is particularly relevant. Note that any changes you make to the `httpd.conf` file will only be effective upon starting or re-starting Apache HTTP Server.

This document relates to Apache HTTP Server version 2.4.2; the configuration of other versions may differ.

Step 1: Configure the Tomcat Connector

Find the normal (non-SSL) `Connector` directive in Tomcat's `<Bitbucket home directory>/shared/server.xml` file, and add the `scheme`, `proxyName`, and `proxyPort` attributes as shown below. Instead of `mycompany.com`, set the `proxyName` attribute to your domain name that Apache HTTP Server will be configured to serve. This informs Bitbucket of the domain name and port of the requests that reach it via Apache HTTP Server, and is important to the correct operation of the Bitbucket functions that construct URLs.

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="8443"
  compression="on"
  compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,application/javascript,
application/x-javascript"
  scheme="http"
  proxyName="mycompany.com"
  proxyPort="80" />
```

Note: Apache HTTP Server's `ProxyPreserveHost` directive is another way to have the hostname of the incoming request recognized by Bitbucket instead of the hostname at which Bitbucket is actually running. However, the `ProxyPreserveHost` directive does not cause the scheme to be properly set. Since we have to mess with Tomcat's `Connector` directive anyway, we recommend that you stick with the above-described approach, and don't bother to set the `ProxyPreserveHost` in Apache HTTP Server.

For more information about configuring the Tomcat Connector, refer to the [Apache Tomcat 7.0 HTTP Connector Reference](#).

Step 2: Change Bitbucket's base URL

After re-starting Bitbucket, open a browser window and log into Bitbucket using an administrator account. Go to the Bitbucket administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the proxy URL (the URL that Apache HTTP Server will be serving).

Step 3 (optional): Set a context path for Bitbucket

By default, Bitbucket is configured to run with an empty context path; in other words, from the 'root' of the server's name space. In that default configuration, Bitbucket is accessed at:

```
http://localhost:7990/
```

It's perfectly fine to run Bitbucket with the empty context path as above. Alternatively, you can set a context path by changing the `Context` directive in Tomcat's `<Bitbucket home directory>/shared/server.xml` file:

```
<Context path="/bitbucket" docBase="${catalina.home}/atlassian-bitbucket" reloadable="false" useHttpOnly="
true">
  ....
</Context>
```

If you do set a context path, it is important that the same path be used in [Step 5](#), when setting up the `ProxyPass` and `ProxyPassReverse` directives. You should also append the context path to Bitbucket's base URL (see [Step 2](#)).

Step 4: Enable `mod_proxy` and `mod_proxy_http` in Apache HTTP Server

In the [mod_proxy](#) documentation, you will read that `mod_proxy` can be used as a forward proxy, or as a reverse proxy (gateway); you want the latter. Where the `mod_proxy` documentation mentions '*origin server*', it refers to your Bitbucket instance. Unless you have a good reason for doing otherwise, load `mod_proxy` and `mod_proxy_http` dynamically, using the [LoadModule directive](#); that means un-commenting the following lines in the `httpd.conf` file:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

Experienced administrators may be aware of the Apache Connector module, `mod_jk`. Atlassian does not recommend use of the `mod_jk` module with Bitbucket, since it has proven itself to be less reliable than `mod_proxy`.

Step 5: Configure `mod_proxy` to map requests to Bitbucket

To configure `mod_proxy` for use with Bitbucket, you need to use the [ProxyPass](#) and [ProxyPassReverse](#) directives in Apache HTTP Server's `httpd.conf` file as follows:

```
ProxyPass / http://localhost:7990/ connectiontimeout=5 timeout=300
ProxyPassReverse / http://localhost:7990/
```

Suppose Apache HTTP Server is configured to serve the `mycompany.com` domain; then the above directives tell Apache HTTP Server to forward web requests of the form `http://mycompany.com/*` to the Tomcat connector (Bitbucket) running on port 7990 on the same machine.

The `connectiontimeout` attribute specifies the number of seconds Apache HTTP Server waits for the creation of a connection to Bitbucket.

The `timeout` attribute specifies the number of seconds Apache HTTP Server waits for data to be sent to Bitbucket.

If you set up a context path for Bitbucket in [Step 3](#), you'll need to use that context path in your `ProxyPass` and `ProxyPassReverse` directives. Suppose your context path is set to `/bitbucket`, the directives would be as follows:

```
ProxyPass /bitbucket http://localhost:7990/bitbucket connectiontimeout=5 timeout=300
ProxyPassReverse /bitbucket http://localhost:7990/bitbucket
```

If Bitbucket is to run on a different domain and/or different port, you should use that domain and/or port number in the `ProxyPass` and `ProxyPassReverse` directives; for example, suppose that Bitbucket will run on port 9900 on `private.mycompany.com` under the context path `/bitbucket`, then you would use the following directives:

```
ProxyPass /bitbucket http://private.mycompany.com:9900/bitbucket connectiontimeout=5 timeout=300
ProxyPassReverse /bitbucket http://private.mycompany.com:9900/bitbucket
```

Step 6: Configure `mod_proxy` to disable forward proxying

If you are using Apache HTTP Server as a reverse proxy only, and not as a forward proxy server, you should turn forward proxying off by including a [ProxyRequests](#) directive in the `httpd.conf` file, as follows:

```
ProxyRequests Off
```

Step 7: Allow proxying to Bitbucket from everywhere

Strictly speaking, this step is unnecessary because access to proxied resources is unrestricted by default. Nevertheless, we explicitly allow access to Bitbucket from any host so that this policy will be applied regardless of any subsequent changes to access controls at the global level. Use the [Proxy](#) directive in the `httpd.conf` file as follows:

```
<Proxy *>
  Order Deny,Allow
  Allow from all
</Proxy>
```

The [Proxy](#) directive provides a context for the directives that are contained within its delimiting tags. In this case, we specify a wild-card url (the asterisk), which applies the two contained directives to all proxied requests.

The [Order](#) directive controls the order in which any [Allow](#) and [Deny](#) directives are applied. In the above configuration, we specify "Deny,Allow", which tells Apache HTTP Server to apply any [Deny](#) directives first, and if any match, the request is denied unless it also matches an [Allow](#) directive. In fact, "Deny,Allow" is the default; we include it merely for the sake of clarity. Note that we specify one [Allow](#) directive, which is described below, and don't specify any [Deny](#) directives.

The [Allow](#) directive, in this context, controls which hosts can access Bitbucket via Apache HTTP Server. Here, we specify that all hosts are allowed access to Bitbucket.

Step 8 (optional): Configure Apache HTTP Server for SSL

If you want to set up SSL access to Bitbucket, follow the instructions on [Secure Bitbucket with Apache using SSL](#). When you are finished, users will be able to make secure connections to Apache HTTP Server; connections between Apache HTTP Server and Bitbucket will remain unsecured (not using SSL). If you don't want to set up SSL access, you can skip this section entirely.

Note: It would be possible to set up an SSL connection between Apache HTTP Server and Tomcat (Bitbucket), but that configuration is very unusual, and not recommended in most circumstances.

A note about application links

When an [application link](#) is established between Bitbucket and another Atlassian product (for example Jira), and Bitbucket is operating behind Apache HTTP Server, the link from the other product to Bitbucket must be via the proxy URL; that is, the 'reciprocal URL' from, say Jira, to Bitbucket must match the proxy name and port that you set at [Step 1](#).

Troubleshooting

In general, if you are having problems:

1. Ensure that Bitbucket works as expected when running directly from Tomcat on `http://localhost:7990/bitbucket`.
2. Watch the log files (usually in `/var/log/httpd/` or `/var/log/apache2/`). Check that you have a `LogLevel` directive in your `httpd.conf`, and turn up logging (`LogLevel debug`) to get more info.
3. Check out the [Knowledge Base](#).

In particular:

- On **Fedora Core 4** people have reported 'permission denied' errors when trying to get `mod_proxy` (and `mod_jk`) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.
- Some users have reported problems with user sessions being hijacked when the `mod_cache` module is enabled. If you have such problems, disable the `mod_cache` module. Note that this module is enabled by default in some Apache HTTP Server version 2 distributions.

Secure Bitbucket with Apache using SSL

When opened in a viewport, the user will be redirected to: [Securing your Atlassian applications with Apache using SSL](#).

This guide for integrating Apache and SSL is now obsolete

We've created a much better guide for [Securing your Atlassian applications with Apache using SSL](#).

You may also be interested in our other Reverse Proxy Setup Guides:

- [Proxying Atlassian server applications with Apache HTTP Server \(mod_proxy_http\)](#)
- [Proxying Atlassian server applications with Apache HTTP Server \(mod_proxy_ajp\)](#)
- [Reverse Proxy and Application Link Troubleshooting Guide](#)

You can run Bitbucket Data Center and Server behind a reverse proxy, such as Apache HTTP Server or nginx, that is secured using HTTPS (HTTP over SSL). You should consider doing this, and making secure access mandatory, if usernames, passwords, and other proprietary data may be at risk.

There are other network topology options for running Bitbucket; for an overview of some common options, see [Proxy and secure Bitbucket](#).

When Bitbucket is set up following the instructions on this page, external access to Bitbucket is via Apache HTTP Server as a reverse proxy, where external communication with the proxy uses HTTPS. All communication between the user's browser and Apache will be secured, whereas communication between Apache and Bitbucket will not be secured (it doesn't use SSL).

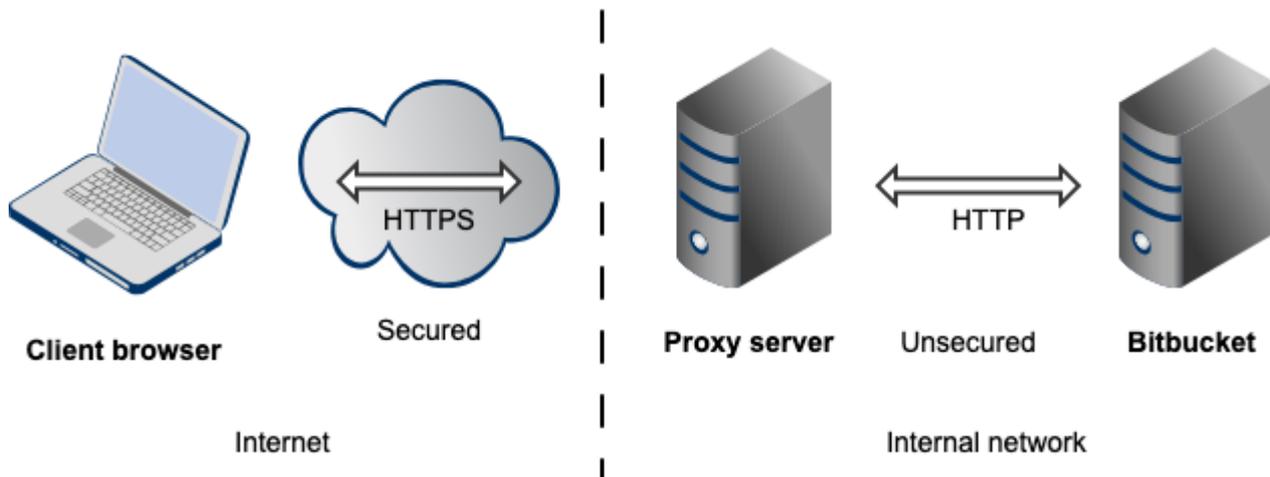
- The steps on this page would normally be performed after [Integrate Bitbucket with Apache HTTP Server](#).

On this page:

- [This guide for integrating Apache and SSL is now obsolete](#)
- [Step 1: Configure the Tomcat Connector for SSL](#)
- [Step 2: Set up a virtual host in Apache HTTP Server](#)
- [Step 3: Create SSL certificate and key files](#)
- [Step 4: Update the base URL to use HTTPS](#)
- [Using a self-signed certificate](#)

Related pages:

- [Integrate Bitbucket with Apache HTTP Server](#)
- [Secure Bitbucket with Tomcat using SSL](#)
- [Secure Bitbucket behind nginx using SSL](#)



Note that:

- The reverse proxy (for example, Apache) will listen for requests on port 443.
- Bitbucket, by default, will listen for requests on port 7990. Bitbucket (Tomcat) needs to know the URL (proxy name) that the proxy serves.
- The address with which to access Bitbucket will be `https://<proxyName>:<proxyPort>/<context path>`, for example `https://mycompany.com:443/bitbucket`
- Any existing [links with other applications](#) will need to be reconfigured using this new URL for Bitbucket.
- Bitbucket (Tomcat) should be configured to refuse requests on port 7990 and to redirect those to the proxy on port 443.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).
- It would be possible to set up an SSL connection between the proxy server and Tomcat (Bitbucket), but that configuration is very unusual, and not recommended in most circumstances.
- Incidentally, note that Bitbucket 4.0 and later versions do not support `mod_auth_basic`.

Step1: Configure the Tomcat Connector for SSL

Find the normal (non-SSL) `Connector` directive in Tomcat's `<Bitbucket home directory>/shared/server.xml` file, and change the `redirectPort`, `scheme`, `proxyName` and `proxyPort` attributes as follows:

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="443"
  compression="on"
  compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,application/javascript,
application/x-javascript"
  secure="true"
  scheme="https"
  proxyName="mycompany.com"
  proxyPort="443" />
```

The `redirectPort` directive causes Tomcat-initiated redirections to secured resources to use the specified port. Right now, the Bitbucket configuration of Tomcat does not involve Tomcat-initiated redirections, so the change to `redirectPort` is redundant. Nevertheless, we suggest that you change it as directed above for the sake of completeness.

Start, or restart, Bitbucket.

Step 2: Set up a virtual host in Apache HTTP Server

Un-comment the following LoadModule directive in Apache HTTP Server's `httpd.conf` file:

```
LoadModule ssl_module modules/mod_ssl.so
```

Add the following directives to the `httpd.conf` file:

```
Listen 443
<VirtualHost *:443>
    SSLEngine On
    SSLCertificateFile "/usr/local/apache2/conf/server.crt"
    SSLCertificateKeyFile "/usr/local/apache2/conf/server.key"
    SSLCertificateChainFile "/usr/local/apache2/conf/server.crt"
    ProxyPass / http://localhost:7990/ connectiontimeout=5 timeout=300
    ProxyPassReverse / http://localhost:7990/
</VirtualHost>
```

The `Listen` directive instructs Apache HTTP Server to listen for incoming requests on port 443. Actually, we could omit that directive in this case, since Apache HTTP Server listens for `https` requests on port 443 by default. Nevertheless, it's good to make one's intentions explicit.

The `VirtualHost` directive encloses a number of child directives that apply only and always to requests that arrive at port 443. Since our `VirtualHost` block does not include a `ServerName` directive, it inherits the server name from the main server configuration.

The `SSLEngine` directive toggles the use of the SSL/TLS Protocol Engine. In this case, we're using it to turn SSL on for all requests that arrive at port 443.

The `SSLCertificateFile` directive tells Apache HTTP Server where to find the PEM-encoded certificate file for the server.

The `SSLCertificateKeyFile` directive tells Apache HTTP Server where to find the PEM-encoded private key file corresponding to the certificate file identified by the `SSLCertificateFile` directive. Depending on how the certificate file was generated, it may contain a RSA or DSA private key file, making the `SSLCertificateKeyFile` directive redundant; however, Apache strongly discourages that practice. The recommended approach is to separate the certificate and the private key. If the private key is encrypted, Apache HTTP Server will require a pass phrase to be entered when it starts up.

The `SSLCertificateChainFile` is optional. Please consult with the CA vendor to verify if this is required. This directive sets the optional all-in-one file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate.

The `ProxyPass` and `ProxyPassReverse` directives should be set up in the manner described in [Step 5 of the Integrate Bitbucket with Apache HTTP Server](#) page. In particular, if Bitbucket is to run on a separate machine from Apache, you should use that domain (and perhaps the port number and context path) in the `ProxyPass` and `ProxyPassReverse` directives.

For more information about the support for SSL in Apache HTTP Server, refer to the [Apache SSL/TLS Encryption](#) manual. In addition, you will find lots of relevant information in the `<apache directory>/conf/extra/httpd-ssl.conf` file, which is included in the standard Apache distribution.

Start, or restart, Apache.

Step 3: Create SSL certificate and key files

In [Step 2](#), you specified `server.crt` and `server.key` as the certificate file and private key file respectively. Those two files must be created before we can proceed. This step assumes that [OpenSSL](#) is installed on your server.

Generate a server key file:

```
openssl genrsa -des3 -out server.key 2048
```

You will be asked to provide a password. Make sure that the password is strong because it will form the one real entry point into the SSL encryption set-up. *Make a note of the password because you'll need it when starting Apache HTTP Server later.*

If you don't wish to specify a password, don't use the `-des3` option in the command above.

Generate a certificate request file (`server.csr`):

```
openssl req -new -key server.key -out server.csr
```

Generate a self-signed certificate (`server.crt`):

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

The above command generates a self-signed certificate that is valid for one year. You can use the certificate signing request to purchase a certificate from [a certificate authority](#). For testing purposes though, the self-signed certificate will suffice. Copy the certificate file and private key file to the locations you specified in [Step 2](#).

```
cp server.key /usr/local/apache2/conf/  
cp server.crt /usr/local/apache2/conf/
```

Step 4: Update the base URL to use HTTPS

Open a browser window and log into Bitbucket using an administrator account. Go to the Bitbucket administration area and click **Server settings** (under 'Settings'). Change **Base URL** to use HTTPS, for example, "https://bitbucket.mycompany.com").

Using a self-signed certificate

There are two implications of using the self-signed certificate:

- When you access Bitbucket in a web browser, you can expect a warning to appear, alerting you that an un-trusted certificate is in use. Before proceeding you will have to indicate to the browser that you trust the certificate.
- When you perform a Git clone operation, SSL verification will fail.

The SSL verification error message will look something like this:

```
error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify  
failed while accessing https://justme@mycompany/git/TP/test.git
```

It's easy to fix. Turn SSL verification off for individual Git operations by setting the `GIT_SSL_NO_VERIFY` environment variable. In Unix, you can set the variable in-line with Git commands as follows:

```
GIT_SSL_NO_VERIFY=true git clone https://justme@mycompany/git/TP/test.git
```

In Windows you have to set the variable in a separate shell statement:

```
set GIT_SSL_NO_VERIFY=true  
git clone https://justme@mycompany/git/TP/test.git
```

Once you have purchased and installed a signed certificate from a certificate authority, you will no longer have to include the `GIT_SSL_NO_VERIFY` modifier.

Secure Bitbucket behind nginx using SSL

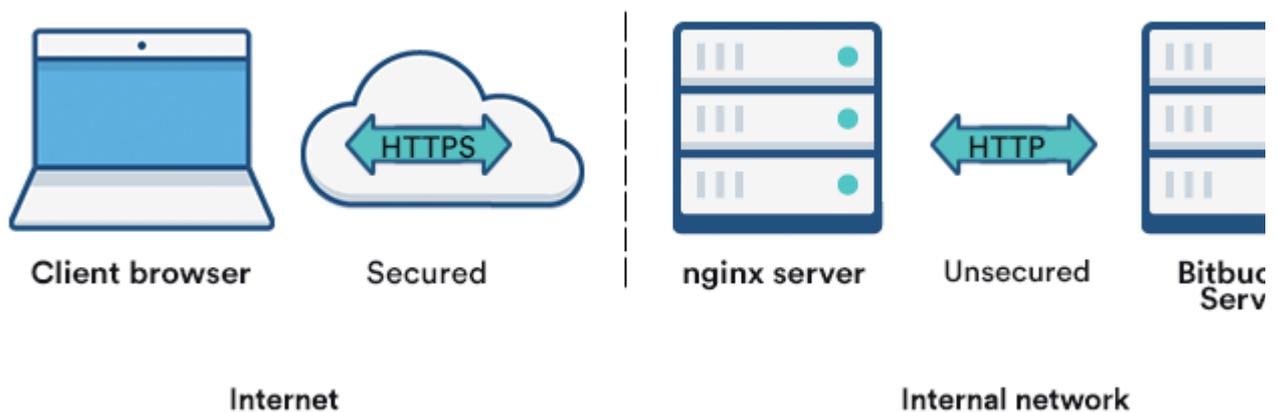
This page describes how to establish a network topology in which the nginx server acts as a [reverse proxy](#) for Bitbucket Data Center and Server. Typically, such a configuration would be used when Bitbucket is installed in a protected zone 'behind the firewall', and nginx provides a gateway through which users outside the firewall can access Bitbucket.

The configuration described on this page results in a scenario where:

- External client connections with nginx are secured using SSL. Connections between nginx and Bitbucket are unsecured.
- Bitbucket and nginx run on the same machine.
- Bitbucket is available at `https://mycompany.com:7990/bitbucket`.

On this page

- [Step 1: Configure the Embedded Tomcat Connector](#)
- [Step 2: Set a context path for Bitbucket](#)
- [Step 3: Change Bitbucket's base URL](#)
- [Step 4: Configure nginx](#)
- [Resources](#)



Also note that:

- We assume that you already have a running instance of nginx. If not, refer to the [nginx documentation](#) or instructions on downloading and installing nginx.
- SSL certificates must be installed on the server machine.
- Any existing [links with other applications](#) will need to be reconfigured using the new URL for Bitbucket.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).

Be aware that Bitbucket does not need to run behind a web server, since it is capable of serving web requests directly; to secure Bitbucket when run in this way see [Secure Bitbucket with Tomcat using SSL](#). Otherwise, if you want to install Bitbucket in an environment that incorporates nginx, this document is for you. (You can of course run Bitbucket behind nginx without securing client connections to nginx using SSL we don't describe this option on this page.)

Note that the [Atlassian Support Offering](#) does not cover nginx integration. Assistance with nginx may be obtained through the [Atlassian community](#) or from an [Atlassian Partner](#).

Step 1: Configure the Embedded Tomcat Connector

Find the Bitbucket configuration properties file `<Bitbucket home directory>/shared/bitbucket.properties`, creating it if necessary and add the properties as shown below. Instead of `mycompany.com`, set the `server.proxy-name` property to your domain name that the nginx server will be configured to serve. This informs Bitbucket of the domain name and port of the requests that reach it via nginx, and is important for the correct operation of the Bitbucket functions that construct URLs.

```
server.port=7990
server.secure=true
server.scheme=https
server.proxy-port=443
server.proxy-name=mycompany.com
```

Step 2: Set a context path for Bitbucket

By default, Bitbucket is configured to run with an empty context path; in other words, from the 'root' of the server's name space. In that default configuration, Bitbucket would be accessed at:

```
http://mycompany.com:7990/
```

For the example configuration on this page, we want Bitbucket to be accessed at:

```
https://mycompany.com/bitbucket
```

In Bitbucket's configuration properties file `<Bitbucket home directory>/shared/bitbucket.properties` file, set the context path to `/bitbucket` by adding the following property:

```
server.context-path=/bitbucket
```

If you use a context path, it is important that the same path is:

- appended to the context path of Bitbucket's base URL ([Step 3](#)).
- used when setting up the location for the `proxy_pass` directive ([Step 4](#)).

Step 3: Change Bitbucket's base URL

Before re-starting Bitbucket, open a browser window and log into Bitbucket using an administrator account. Go to the Bitbucket administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the proxy URL (the URL that the nginx server will be serving).

For this example, use `http://mycompany.com/bitbucket` (Note the context path included with this).

Step 4: Configure nginx

Edit `/etc/nginx/nginx.conf`, using the example server configuration below, to configure nginx as a proxy server.

Put the `proxy_pass` directive in the location block, and specify the protocol, name and port of the proxied server in the parameter (in our case, it is `http://localhost:7990`):

<https://confluence.atlassian.com/bitbucketserverkb/git-push-fails-client-intended-to-send-too-large-chunked-body-779171802.html>

```
http {
...
...
    client_max_body_size 0;
...
...
    server {
        listen            443;
        server_name       mycompany.com;

        ssl               on;
        ssl_certificate    <path/to/your/certificate>;
        ssl_certificate_key <path/to/your/certificate/key>;
        ssl_session_timeout 5m;
        ssl_protocols     TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers        HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        # Optional optimisation - please refer to
        # http://nginx.org/en/docs/http/configuring_https_servers.html
        # ssl_session_cache shared:SSL:10m;
        location /bitbucket {
            proxy_pass          http://localhost:7990;
            proxy_set_header    X-Forwarded-Host $host;
            proxy_set_header    X-Forwarded-Server $host;
            proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header    X-Real-IP $remote_addr;
            proxy_redirect      off;
        }
    }
...
...
}
```

Refer to http://nginx.org/en/docs/http/nginx_http_proxy_module.html.

Changes made in the configuration file will not be applied until the command to reload configuration is sent to nginx or it is restarted. To reload the configuration, execute:

```
nginx -s reload
```

This command should be executed under the same user that started nginx.

i Notice that we added `client_max_body_size 0;` to the `http` block of the nginx configuration because of [Git push fails - client intended to send too large chunked body](#).

Resources

You may find the following resources helpful in setting up Bitbucket behind nginx:

- http://nginx.org/en/docs/http/configuring_https_servers.html
- <http://www.cyberciti.biz/tips/using-nginx-as-reverse-proxy.html>
- <https://mywushublog.com/2012/08/atlassian-tools-and-nginx/>

Secure Bitbucket behind HAProxy using SSL

This page describes how to establish a network topology in which the HAProxy server acts as a reverse proxy for Bitbucket Data Center and Server. Typically, such a configuration would be used when either when:

1. Bitbucket is installed in a protected zone 'behind the firewall', and HAProxy provides a gateway through which users outside the firewall can access Bitbucket.
2. Bitbucket needs to be served on protected ports (e.g. ports < 1024 on Linux). Bitbucket cannot access these ports directly as *it must not* be run as a privileged user (e.g. root). In this case HAProxy can bind to these ports and forward the requests to Bitbucket.

On this page:

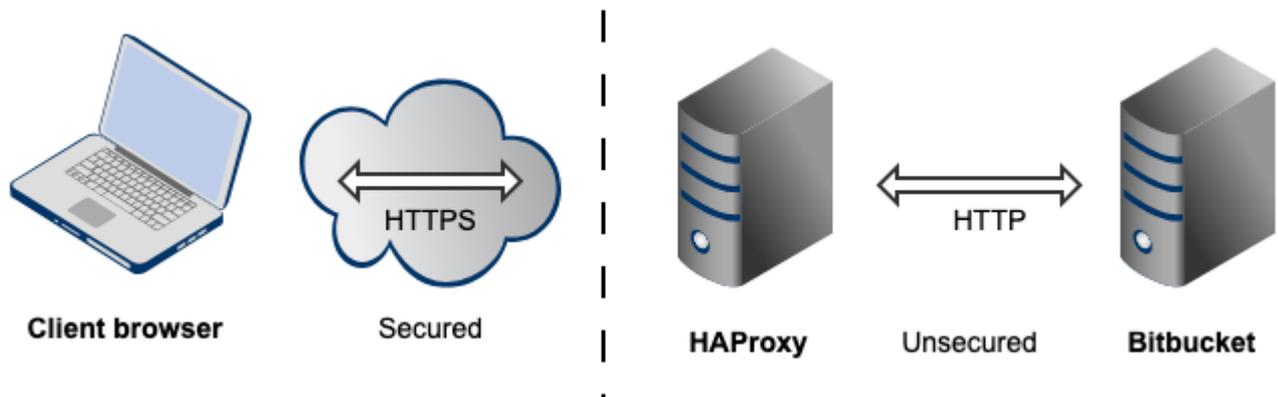
- [Step 1: Set a context path for Bitbucket](#)
- [Step 2: Change Bitbucket's base URL](#)
- [Step 3: Configure the Tomcat Connector](#)
- [Step 4: Configure HAProxy](#)
- [\(Optional\) Step 4: Redirect SSH connections](#)

Related pages:

- <http://www.haproxy.org/#docs>
- [Setting up SSH port forwarding](#)

The configuration described on this page results in a scenario where:

- External client connections with HAProxy are secured using SSL. Connections between HAProxy and Bitbucket are unsecured.
- Bitbucket and HAProxy run on the same machine.
- Bitbucket is currently available at `http://mycompany.com:7990/`.
- Bitbucket is to be made available at `https://mycompany.com/bitbucket`.



Important considerations

- We assume that you already have a running instance of HAProxy.
- SSL certificates must be installed on the server machine.
- Any existing [links with other applications](#) will need to be reconfigured using the new URL for Bitbucket.
- Securing Git operations between the user's computer and Bitbucket is a separate consideration - see [Enabling SSH access to Git](#).
- It is also possible to get Bitbucket to directly use SSL without the help of a proxy as documented in [Secure Bitbucket with Tomcat using SSL](#).

Note that the [Atlassian Support Offering](#) does not cover HAProxy integration, but you can get assistance with HAProxy from the Atlassian community on answers.atlassian.com, or from an [Atlassian Expert](#).

Step 1: Set a context path for Bitbucket

Bitbucket and HAProxy need to be serving from the same context. Bitbucket is currently accessed at `http://mycompany.com:7990`. It needs to be changed to serve from `http://mycompany.com:7990/bitbucket` to match context `https://mycompany.com/bitbucket`.

1. Locate the `bitbucket.properties` file in the shared directory of your `<Bitbucket home directory>`.
2. Change the context path for Bitbucket by adding

```
server.context-path=/bitbucket
```

3. Save the file.

Good to know

- If you use a context path, it is important that the same path is appended to the context path of Bitbucket's base URL ([Step 2](#)).
- The context path for serving from the `root` context is `path=""` (i.e. **not** `path="/"`).

Step 2: Change Bitbucket's base URL

1. Open a browser window and log into Bitbucket using an administrator account.
2. Go to the Bitbucket administration area and click **Server settings** (under 'Settings'), and change **Base URL** to match the URL HAProxy will be serving. For this example, use `https://mycompany.com/bitbucket`.

Step 3: Configure the Tomcat Connector

Following our example, you need to configure these attributes that tell Tomcat how HAProxy is serving Bitbucket so it can generate correct URLs.

Locate the `<Bitbucket home directory>/shared/bitbucket.properties` file, and add the following:

```
server.secure=true
server.scheme=https
server.proxy-port=443
server.redirect-port=443
server.proxy-name=mycompany.com
```

What these attributes do

- `proxyPort` is set to 443 to indicate that HAProxy is accepting connections over on the standard HTTPS port 443.
- `proxyName` and `scheme` are set to the values that HAProxy is serving Bitbucket over.
- `secure` attribute is also set to `true` to tell Bitbucket that the connection between the client and HAProxy is considered secure.
- `redirectPort` is set to 443 so that Tomcat knows how to send a user to a secure location when necessary (this is not really necessary in this example because this connector is already secure).

For more information about configuring the Tomcat Connector, refer to the [Apache Tomcat 7.0 HTTP Connector Reference](#).

Step 4: Configure HAProxy

Merge the example below into your HAProxy configuration (e.g. `/etc/haproxy/haproxy.cfg`). This is a complete HAProxy 1.5.x configuration. Note that HAProxy 1.5.x or greater is required for SSL support. You can just take the bits that fit your needs. The important configuration is in the `bitbucket_http_frontend` and `bitbucket_http_backend`.

```
global
    log /dev/log local0
    log /dev/log local1 notice
    user haproxy
    group haproxy
    daemon
    ssl-default-bind-options no-sslv3
    maxconn 1000

defaults
    log          global
    mode http
    option httplog
    option dontlognull
    timeout connect 5000
    timeout client  50000
    timeout server  50000

# Tells HAProxy to start listening for HTTPS requests. It uses the SSL key
# and certificate found within certAndKey.pem. All requests will be routed
# to the bitbucket_http_backend.
frontend bitbucket_http_frontend
    bind *:443 ssl crt /etc/haproxy/certAndKey.pem ciphers HIGH:!aNULL:!MD5
    default_backend bitbucket_http_backend
    # This is an optional rule that will redirect all requests to https://mycompany.com
    # to https://mycompany.com/bitbucket.
    redirect location /bitbucket if { path -i / }

# The bitbucket_http_backend simply forwards all requests onto http://mycompany.com:7990/.
# It will only allow 50 concurrent connections to the server at once.
backend bitbucket_http_backend
    mode http
    option httplog
    option forwardfor
    option http-server-close
    option httpchk
    server bitbucket01 mycompany.com:7990 maxconn 50
```

(Optional) Step 4: Redirect SSH connections

HAProxy also has the ability to proxy all Bitbucket SSH traffic. See [Setting up SSH port forwarding](#) for details.

High availability for Bitbucket

This page describes how to set up a single Bitbucket Server instance in a highly available configuration.

For production installs, we highly recommend that you first read [Use Bitbucket in the enterprise](#).

For Active/Active high availability with Bitbucket Server, see [Bitbucket Data Center resources](#) instead.

For guidance on using Bitbucket Data Center as part of your disaster recovery strategy, see the [Disaster recovery guide for Bitbucket Data Center](#).

If Bitbucket Server is a critical part of your development workflow, maximizing application availability becomes an important consideration. There are many possible configurations for setting up a HA environment for Bitbucket Server, depending on the infrastructure components and software (SAN, etc.) you have at your disposal. This guide provides a high-level overview and the background information you need to be able to set up a single Bitbucket Server in a highly available configuration.

Note that Atlassian's [Bitbucket Data Center resources](#) product uses a cluster of Bitbucket Server nodes to provide Active/Active failover. It is the deployment option of choice for larger enterprises that require high availability and performance at scale, and is fully supported by Atlassian. Read about [Failover for Bitbucket Data Center](#).

High availability

[High availability](#) describes a set of practices aimed at delivering a specific level of "availability" by eliminating and/or mitigating failure through redundancy. Failure can result from unscheduled down-time due to network errors, hardware failures or application failures, but can also result from failed application upgrades. Setting up a highly available system involves:

Proactive Concerns

- Change management (including staging and production instances for change implementation)
- Redundancy of network, application, storage and databases
- Monitoring system(s) for both the network and applications

Reactive Concerns

- Technical [failover](#) mechanisms, either automatic or scripted semi-automatic with manual switchover
- Standard Operating Procedure for guided actions during crisis situations

This guide assumes that processes such as change management are already covered and will focus on **redundancy / replication and failover procedures**. When it comes to setting up your infrastructure to quickly recover from system or application failure, you have different options. These options vary in the level of uptime they can provide. In general, as the required uptime increases, the complexity of the infrastructure and the knowledge required to administer the environment increases as well (and by extension the cost goes up as well).

Understanding the availability requirements for Bitbucket Server

Central version control systems such as Subversion, CVS, ClearCase and many others require the central server to be available for any operation that involves the version control system. Committing code, fetching the latest changes from the repository, switching branches or retrieving a diff all require access to the central version control system. If that server goes down, developers are severely limited in what they can do. They can continue coding until they're ready to commit, but then they're blocked.

Git is a distributed version control system and developers have a full clone of the repository on their machines. As a result, most operations that involve the version control system don't require access to the central repository. When Bitbucket Server is unavailable developers are not blocked to the same extent as with a central version control system.

As a result, the availability requirements for Bitbucket Server *maybe* less strict than the requirements for say Subversion.

| Consequences of Bitbucket Server unavailability | |
|--|---|
| ✔ Unaffected | ✘ Affected |
| <p>Developer:</p> <ul style="list-style-type: none"> • Commit code • Create branch • Switch branches • Diff commits and files • ... • Fetch changes from fellow developers | <p>Developer:</p> <ul style="list-style-type: none"> • Clone repository • Fetch changes from central repository • Push changes to central repository • Access Bitbucket Server UI - create/do pull requests, browse code <p>Build server:</p> <ul style="list-style-type: none"> • Clone repository • Poll for changes <p>Continuous Deployment:</p> <ul style="list-style-type: none"> • Clone repository |

Failover options

High availability and recovery solutions can be categorized as follows:

| Failover option | Recovery time | Description | Possible with Bitbucket Server |
|--------------------------------|---|--|--------------------------------|
| Automatic correction / restart | 2-10 min (application failure)
hours-days (system failure) | <ul style="list-style-type: none"> • Single node, no secondary server available • Application and server are monitored • Upon failure of production system, automatic restarting is conducted via scripting • Disk or hardware failure may require reprovisioning of the server and restoring application data from a backup | ✔ |
| Cold standby | 2-10 min | <ul style="list-style-type: none"> • Secondary server is available • Bitbucket Server is NOT running on secondary server • Filesystem and (optionally) database data is replicated between the 'active' server and the 'standby' server • All requests are routed to the 'active' server • On failure, Bitbucket Server is started on the 'standby' server and shut down on the 'active' server. All requests are now routed to the 'standby' server, which becomes 'active'. | ✔ |

| | | | |
|----------------|----------|---|---|
| Warm standby | 0-30 sec | <ul style="list-style-type: none"> • Secondary service is available • Bitbucket Server is running on both the 'active' server and the 'standby' server, but all requests are routed to the 'active' server • Filesystem and database data is replicated between the 'active' server and the 'standby' server • All requests are routed to the 'active' server • On failure, all requests are routed to the 'standby' server, which becomes 'active' •  This configuration is currently not supported by Bitbucket Server, because Bitbucket Server uses in-memory caches and locking mechanisms. At this time, Bitbucket Server only supports a single application instance writing to the Set the home directory at a time. |  |
| Active /Active | < 5 sec | <ul style="list-style-type: none"> • Provided by Bitbucket Data Center resources, using a cluster of Bitbucket Server nodes and a load balancer. • Bitbucket Server is running, and serving requests, on all cluster nodes. • Filesystem and database data is shared by all cluster nodes. Clustered databases are not yet supported. • All requests are routed to the load balancer, which distributes requests to the available cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds. • Bitbucket Data Center is the deployment option of choice for larger enterprises that require high availability and performance at scale. |  |

Automatic correction

Before implementing failover solutions for your Bitbucket Server instance consider evaluating and leveraging automatic correction measures. These can be implemented through a monitoring service that watches your application and performs scripts to start, stop, kill or restart services.

1. A Monitoring Service detects that the system has failed.
2. A correction script attempts to gracefully shut down the failed system.
 - a. If the system does not properly shut down after a defined period of time, the correction script kills the process.
3. After it is confirmed that the process is not running anymore, it is started again.
4. If this restart solved the failure, the mechanism ends.
 - a. If the correction attempts are not or only partially successful a failover mechanism should be triggered, if one was implemented.

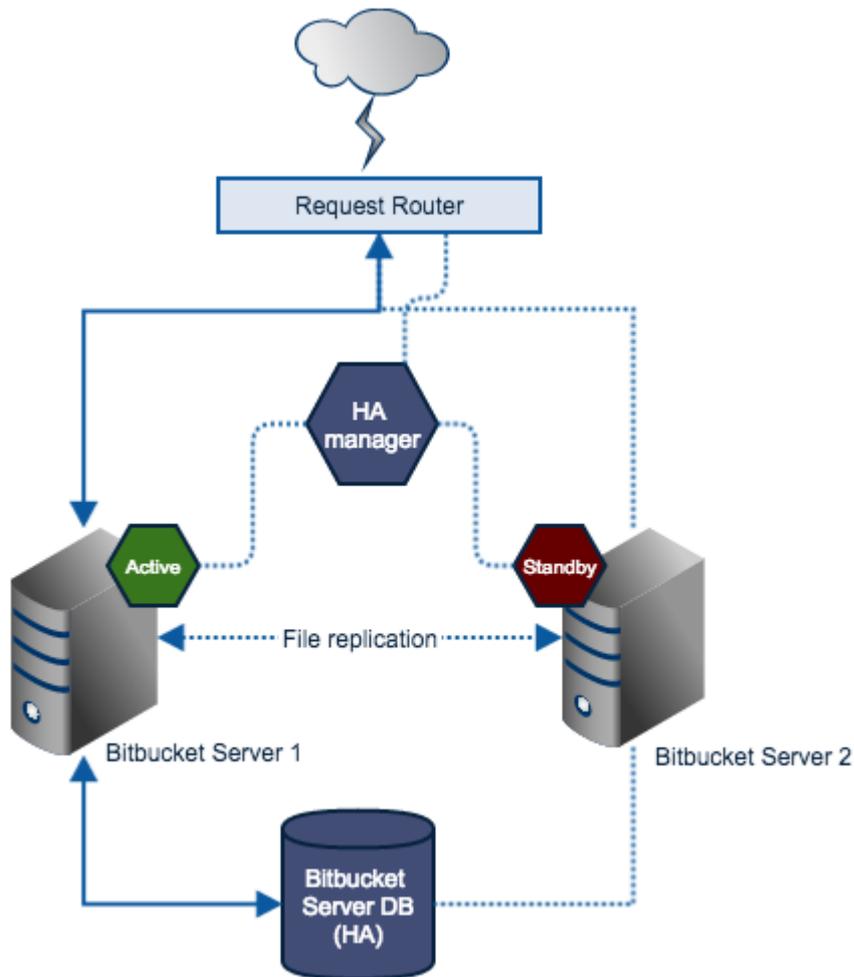
Cold standby

The cold standby (also called Active/Passive) configuration consists of two identical Bitbucket Server instances, where only one server is ever running at a time. The Bitbucket home directory on each of the servers is either a shared (and preferably highly available) network file system or is replicated from the active to the standby Bitbucket Server instance. When a system failure is detected, Bitbucket Server is restarted on the standby server. If the system failure persists, a failover mechanism is started that shuts down Bitbucket Server on the active server and starts Bitbucket Server on the standby server, which is promoted to 'active'. At this time, all requests should be routed to the newly active server.

For each component in the chain of high availability measures, there are various implementation alternatives. Although Atlassian does not recommend any particular technology or product, this guide gives options for each step.

System setup

This section describes one possible configuration for how to set up a single instance of Bitbucket Server for high availability.



Components

Request Router

Forwards traffic from users to the active Bitbucket Server instance.

High Availability Manager

- Tracks the health of the application servers and decides when to fail over to a standby server and designate it as active.
- Manages failover mechanisms and sends notifications on system failure.

Bitbucket Server instance

- Each server hosts an identical Bitbucket Server installation (identical versions).
- Only one server is ever running a Bitbucket Server instance at any one time (known as the active server). All others are considered as standbys.
- Resides on a replicated or shared file system visible to all application servers.

- Must never be modified when the server is in standby mode.

Bitbucket Server DB

The production database, which should be highly available. How this is achieved is not explored in this document. See the following database vendor-specific information on the HA options available to you:

- **Postgres**
<http://www.postgresql.org/docs/9.2/static/high-availability.htm>
- **MySQL**
<http://dev.mysql.com/doc/refman/5.5/en/ha-overview.html>
- **Oracle**
<http://www.oracle.com/technetwork/database/features/availability/index.html>
- **SQLServer**
<http://technet.microsoft.com/en-us/library/ms190202.aspx>

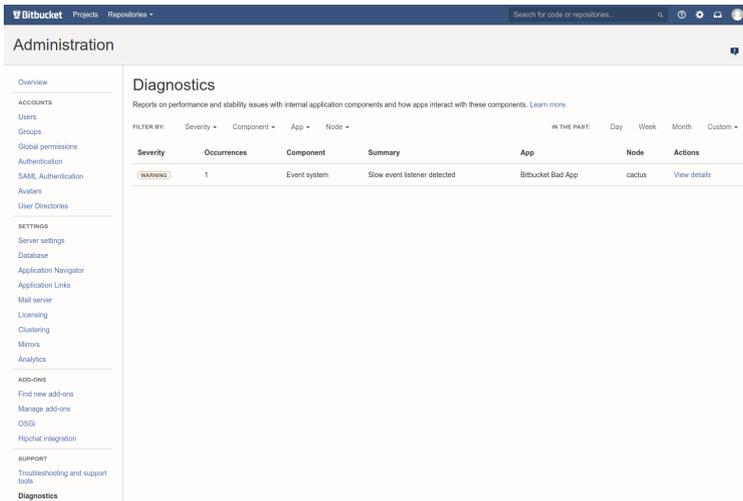
Licensing

Developer licenses can be used for non-production installations of Bitbucket Server deployed on a cold stand-by server. For more information see [developer licenses](#).

Diagnostics for third-party apps

Overview

The Bitbucket Data Center and Server diagnostics tool displays a summary of the alerts that have been raised on the instance in the past 30 days, grouped by issue and app combination.



The screenshot shows the Bitbucket Administration interface. The main content area is titled 'Diagnostics' and contains a table of alerts. The table has columns for Severity, Occurrences, Component, Summary, App, Node, and Actions. A single alert is shown with a severity of 'WARNING', 1 occurrence, and the component 'Event system'. The summary is 'Slow event listener detected', the app is 'Bitbucket Bad App', and the node is 'cactus'. There is a 'View details' link in the Actions column.

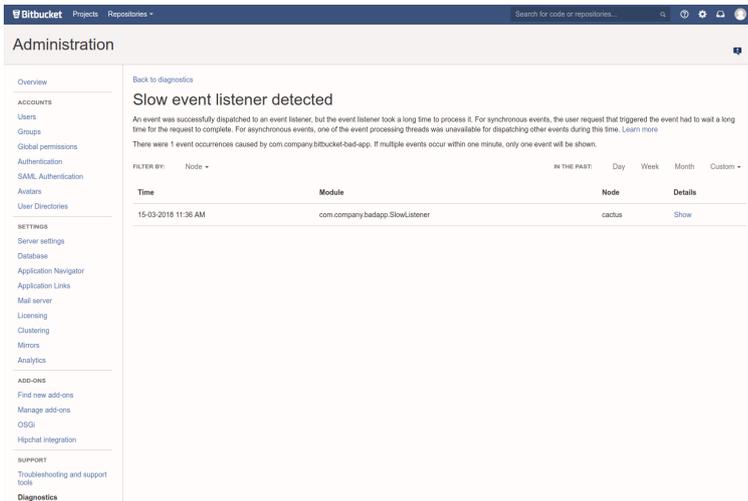
| Severity | Occurrences | Component | Summary | App | Node | Actions |
|----------|-------------|--------------|------------------------------|-------------------|--------|------------------------------|
| WARNING | 1 | Event system | Slow event listener detected | Bitbucket Bad App | cactus | View details |

There are three levels of severity:

- **Error**: a serious problem has occurred that impacts system stability and/or availability
- **Warning**: an issue has been detected that impacts performance or can lead to more serious problems in the future
- **Info**: something worth noting has happened.

The alerts can be filtered by severity, component, app, node and time.

Alert Details

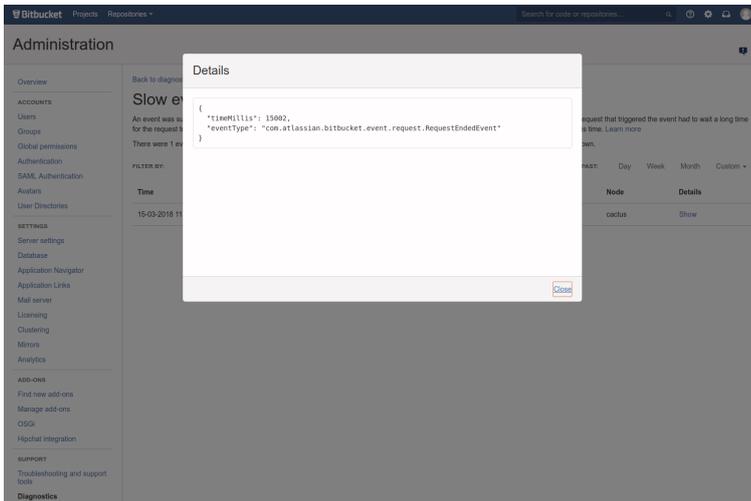


The screenshot shows the Bitbucket Administration interface. The left sidebar contains a navigation menu with categories: Overview, ACCOUNTS (Users, Groups, Global permissions, Authentication, SAML, Authentication, Avatars, User Directories), SETTINGS (Server settings, Database, Application Navigator, Application Links, Mail server, Licensing, Clustering, Minors, Analytics), ADD-ONS (Find new add-ons, Manage add-ons, OSGi, Hipchat integration), and SUPPORT (Troubleshooting and support tools, Diagnostics). The main content area is titled 'Administration' and shows a 'Back to diagnostics' link. The primary heading is 'Slow event listener detected'. Below this, a paragraph explains the issue: 'An event was successfully dispatched to an event listener, but the event listener took a long time to process it. For synchronous events, the user request that triggered the event had to wait a long time for the request to complete. For asynchronous events, one of the event processing threads was unavailable for dispatching other events during this time. Learn more'. A note states: 'There were 1 event occurrences caused by com.company.bitbucket.badapp.SlowListener. If multiple events occur within one minute, only one event will be shown.' Below the text is a table with columns: Time, Module, Node, and Details. The table contains one row: Time: 15-03-2018 11:56 AM, Module: com.company.badapp.SlowListener, Node: caduce, Details: Show. Above the table, there is a filter section: 'FILTER BY: Node' and a 'IN THE PAST:' section with options: Day, Week, Month, Custom.

The alert details page is accessed by clicking into an alert, and displays individual alerts for the selected issue and app combination.

When many alerts are raised in a short time window, just one representative alert is displayed per 1 minute window.

Clicking **Show** displays information relating to the issue, that was collected when the alert was raised.



You can also set up alerts using [JMX metric details](#).

Event system

Atlassian apps have an internal event system that allows core functionality and apps to respond to events such as user actions. When a user performs an action, an event is raised. The event system then dispatches the event to all registered event listeners.

There are two types of event dispatches:

- **Synchronous:** the event is dispatched to event listeners immediately and the request is blocked until all event listeners have processed the event.
- **Asynchronous:** the event is put on the event queue and is dispatched to event listeners in the background. The request that raised the event does not wait for the event listeners to complete their processing of the event.

Many core features make use of the event system as event producers, event listeners or both. In addition, many third-party apps rely on the event system to integrate with the app, or to respond to important app events.

Disruption or degradation of the event system can impact system performance, stability and availability.

Diagnostics monitors the event system for these acute problems, but also for symptoms that could cause problems under increased load.

The following issues are monitored by the diagnostics tool:

- [EVENT-1001](#): **An event was dropped because the event queue was full.**
- [EVENT-2001](#): **A slow event listener was detected.**

Refer to the full event descriptions below.

EVENT-1001 An event was dropped because the event queue was full

ERROR

Problem

An asynchronous event could not be dispatched to registered event listeners because the event queue is full - too many events are awaiting dispatch. As a result, the event has not been processed and (core) functionality may be degraded. What functionality is degraded depends on the event type.

 this only affects asynchronous dispatch of events.

Alert Details

When **EVENT-1001** is raised, the following data is collected to help diagnose the problem:

- `eventType`: the type of event that was dropped
- `queueLength`: the capacity of the event queue
- `threadDumps`: a list of thread dumps of all event processing threads that were busy when the event was dropped.

Cause

The event queue can fill up if events are being raised faster than they can be processed. This is most commonly caused by one or more event listeners taking too long to process an event, or getting completely stuck.

If this is the case, one or more **EVENT-2001** (slow event listener detected) alerts have likely been raised. These **EVENT-2001** alerts can help identify the offending event listener, including the app that provided it.

If **EVENT-2001** alerts do not clearly identify the cause, the thread dumps in the alert details will identify what the event processing threads were doing at the time of the alert.

The event queue may fill up if a component produces too many events in a short time.

Mitigation

If a slow or blocked event listener has been detected and the event listener is provided by a non-critical app, that app could be (temporarily) disabled. For marketplace apps, report the problem with the vendor.

For Atlassian-provided apps, contact support at <https://support.atlassian.com>. For apps that were developed in-house, contact the relevant developer.

Configuration options

Thread dump cool-down period: Controls how often thread dumps should be generated for alerts relating to dropped events. Taking thread dumps can be computationally expensive and can produce a large amount of data when run frequently.

Add the following to the `<BITBUCKET_HOME>/shared/bitbucket.properties` configuration file:

```
# time is in seconds, default is 60s
diagnostics.issues.event.dropped.threaddump.cooldown.seconds=60
```

EVENT-2001 A slow event listener was detected WARNING

Problem

An event was dispatched to an event listener, but the event listener took a long time processing an event. For synchronous events, this means that the user request that raised the event had to wait a long time for the request to complete. For asynchronous events, this means that one of the event processing threads was unavailable for dispatching other events during this time.

EVENT-2001 does not imply that there is an acute problem. Rather, it's a warning that performance may be degraded (for **synchronous events**).

If EVENT-2001 happens frequently or if events are being raised at a high frequency (heavy load), asynchronous event processing will fall behind and the event queue can fill up. This can ultimately lead to EVENT-1001.

Alert Details

When EVENT-2001 is raised, the following data is collected to help diagnose the problem:

- `trigger`: the specific event listener that was slow, including the app that registered it
- `timeMillis`: the time in milliseconds that the event listener spent processing the event
- `eventType`: the type of event that was processed (slowly).

Cause

Event listeners can be slow for a variety of reasons. Some of the most common causes are:

- The event listener performed a long running operation on the event thread: e.g. some type of indexing or automated analysis.
- The event listener performed blocking I/O: e.g. sent a HTTP request to an external system without configuring a sufficiently short timeout.
- The event listener was blocked trying to acquire a lock: e.g. a cluster lock, a local app lock, a database lock, etc.
- A system-wide issue affected the event listener, causing it to be slow:

- The system had a long gc pause.
- The system has run out of database connections.

Mitigation

Some of the causes listed in the previous section indicate that the event listener itself is the cause (long running operation, blocking I/O, acquiring locks without a short timeout). If this is the case, the problem should be reported to the app vendor.

For marketplace vendors, the marketplace listing contains a 'Support' section with instructions on how to best report issues.

For apps provided by Atlassian, please raise an [issue](#) or contact [Atlassian support](#) if the problems are acute.

For apps that have been developed in-house, please contact the app developer (See [Guidelines for Data Center App Development](#) for guidelines on how to avoid slow event listeners).



If the triggering event listener is the cause of the issue, and the event listener is provided by a non-critical app, consider (temporarily) disabling the app.

Configuration options

Slow event listener limit: Controls when an alert is raised for a slow event listener. If an event listener is slower than the configured limit, an EVENT-2001 alert is raised.

Add the following to the `<BITBUCKET_HOME>/shared/bitbucket.properties` configuration file:

```
# time is in milliseconds, default is 15s
diagnostics.issues.event.slow.listener.time.millis=15000
```

Slow event listener limit overrides: Configures overrides for specific event listeners and/or specific apps. This setting can be used to suppress 'slow event listener detected' alerts for specific event listeners or plugins, which have been determined to not be problematic. The value should be a comma-separated list of configurations of individual triggers, where a trigger is either the plugin-key, or the plugin-key followed by the event listener class name. Overrides are only considered if they specify more lenient limits than the value specified by `diagnostics.issues.event.slow.listener.time.millis`.

Add the following to the `<BITBUCKET_HOME>/shared/bitbucket.properties` configuration file:

```
# The following example sets the trigger for com.company.example-app to 30s and sets the
# the limit for the com.company.RepositoryCreatedListener event listener in the same app to 60s.
#
# Configured values are in milliseconds
diagnostics.issues.event.slow.listener.overrides=\
  com.company.example-app:30000,\
  com.company.example-app.com.company.RepositoryCreatedListener:60000
```

Upgrade tasks

UPGRADE TASK-2001 Invalid content in pull request commit backfill upgrade marker file

WARNING

Problem

With the introduction of [pull request links on commit pages in 5.11](#), Bitbucket Data Center and Server must run an upgrade task to index existing pull request commit relationships. The upgrade task processes a single repository at a time, and persists progress by writing an upgrade marker file with the ID of the latest repository processed. Once all repositories have been processed, the upgrade task writes `-1` to the file.

The UPGRADE TASK-2001 issue indicates that the contents of this marker file are not valid. This means progress of the upgrade task cannot be confirmed, and as a result, some pull request commit relationships may not have been indexed. The contents of the file are invalid if the file is empty or the file contents cannot be converted to an integer. A missing upgrade marker file will not trigger this issue.

Alert Details

When **UPGRADE TASK-2001** is raised, the following data is collected to help diagnose the problem:

`content`: the content of the marker file, truncated to 10 characters.

Cause

The upgrade marker file at `<PATH_TO_SHARED_HOME>/config/upgrade/core-backfill-pull-request-commits` is corrupted. The contents of the file are considered invalid if the file is empty or the file contents cannot be converted to an integer.

Mitigation

If this issue is encountered, the marker file can be deleted in order to run the upgrade task again next time Bitbucket is started.

Alternatively, the contents of the file can be set to `-1` in order to mark the upgrade task as completed.

However, this may result in some or all pull request links missing from commits. If the issue occurs repeatedly after manual intervention, contact support at <https://support.atlassian.com>.

Enabling JMX counters for performance monitoring

This article describes how to expose JMX MBeans within Bitbucket Data Center and Server for monitoring with a JMX client.

What is JMX?

JMX ([Java Management eXtensions](#)) is a technology for monitoring and managing Java applications. JMX uses objects called MBeans (Managed Beans) to expose data and resources from your application.

Why would I want to enable JMX monitoring within Bitbucket?

For large Bitbucket instances, enabling JMX allows you to more easily monitor the consumption of application resources. This enables you to make better decisions about how to maintain and optimize machine resources.

What can I monitor with JMX?

It is possible to monitor various statistics using JMX counters within Bitbucket. Below are some examples of some statistics that can be monitored.

Mail statistics (com.atlassian.bitbucket:name=MailStatistics)

| Name | Description |
|---------------------|---|
| AverageMessageSize | Average size (in bytes) of messages sent |
| LargestMessageSent | Largest message that has been sent (in bytes) |
| LastMessageFailure | Date of the last failure to send a message |
| LastMessageSuccess | Date of the last successful message send operation |
| LastQueueFullEvent | Last time the message queue was full |
| QueueFullEventCount | Number of times the message queue was full |
| QueueUsage | Queue usage as a fraction, 0.0d indicates empty and 1.0d indicates full |
| QueuedMessagesCount | Current count of queued (unsent) messages |
| QueuedMessagesSize | Current size (in bytes) of the queued (unsent) messages |
| TotalMailDataSent | Total size (in bytes) of messages sent |
| TotalMessagesFailed | Total number of messages that failed to send |
| TotalMessagesSent | Total number of messages sent |

Diagnostics

Alerts Total (com.atlassian.diagnostics:type=Alerts,name=Total)

| Name | Description |
|----------------------|------------------------------------|
| LatestAlertTimestamp | Timestamp of the most recent alert |

On this page:

- [What is JMX?](#)
- [Expose JMX MBeans within Bitbucket](#)
- [Expose JMX MBeans when Bitbucket is run as a Windows service](#)
- [Verify JMX is configured correctly](#)

Related reading:

- [Understanding JMX \(Oracle\)](#)

| | |
|--------------|--|
| TotalCount | Total number of alerts since the JVM was started |
| ErrorCount | Number of alerts of severity ERROR since the JVM was started |
| InfoCount | Number of alerts of severity INFO since the JVM was started |
| WarningCount | Number of alerts of severity WARNING since the JVM was started |

Plugin (com.atlassian.diagnostics:type=Alerts,Category=Plugin,name=\${PLUGIN_NAME})

| Name | Description |
|----------------------|--|
| LatestAlertTimestamp | Timestamp of the most recent alert |
| TotalCount | Total number of alerts since the JVM was started |
| ErrorCount | Number of alerts of severity ERROR since the JVM was started |
| InfoCount | Number of alerts of severity INFO since the JVM was started |
| WarningCount | Number of alerts of severity WARNING since the JVM was started |
| PluginName | Plugin name, if available |

Issue (com.atlassian.diagnostics:type=Alerts,Category=Issue,name=\${ISSUE_ID})

| Name | Description |
|----------------------|--|
| LatestAlertTimestamp | Timestamp of the most recent alert |
| Component | Component the issue is defined for |
| Count | Number of alerts for the issue since the JVM was started |
| Severity | Issue's severity |
| Description | Issue's description |

Hosting statistics

| Protocol | Object name |
|----------|--|
| SSH | com.atlassian.bitbucket:name=SshHostingStatistics |
| HTTP | com.atlassian.bitbucket:name=HttpHostingStatistics |

Hosting statistic attributes

All the hosting statistics attributes are monotonically increasing since the JVM was restarted

| Name | Description |
|------------------|--|
| CloneCacheBypass | Clone requests that have bypassed the scm-cache |
| CloneCacheHit | Clone requests served from the scm-cache |
| CloneCacheMiss | Clone requests that could not be served from the scm-cache |
| CloneRead | Bytes read from clients during clone operations |

| | |
|-------------------|---|
| CloneRequestCount | Number of clone requests served |
| CloneWritten | bytes written to clients during <i>clone</i> operations |
| FetchRead | Bytes read from clients during <i>fetch</i> operations |
| FetchRequestCount | Number of fetch requests served |
| FetchWritten | Bytes written to clients during fetch operations |
| Requests | Total number of requests served |
| TotalBytesRead | Total bytes read from clients |
| TotalBytesWritten | Total bytes written to clients |

Webhooks statistics (com.atlassian.webhooks:name=Webhooks)

| Name | Description |
|-------------------------------|---|
| PublishCount | A count of the total number of events that could trigger webhooks (A publish may create many dispatches) |
| DispatchSuccessCount | Total number of webhooks to fire successfully with a successful HTTP response |
| DispatchRejectedCount | A count of the number of webhook dispatches that were rejected for execution |
| DispatchLastRejectedTimestamp | The last time a webhook was rejected, either from circuit breaking, or due to too many webhooks being in flight |
| DispatchInFlightCount | Total number of dispatches that have been triggered and are awaiting resolution |
| DispatchFailureCount | Total number of webhooks that fired successfully, but the HTTP response indicates a failure (non 2xx code) |
| DispatchErrorCount | Total number of webhooks to have had an error while they were being dispatched |
| DispatchCount | Total number of webhooks to have been dispatched |

Thread pools

| Thread pool | Description | Object name |
|------------------------|--|--|
| BuildActionsThreadPool | Threads that handle Integrated CI/CD build actions | com.atlassian.bitbucket.thread-pools:name=BuildActionsThreadPool |
| EventThreadPool | Threads that dispatch events to @EventListenermethods | com.atlassian.bitbucket.thread-pools:name=EventThreadPool |
| IoPumpThreadPool | Threads that handle blocking process I/O | com.atlassian.bitbucket.thread-pools:name=IoPumpThreadPool |
| NioPumpThreadPool | Threads that handle nonblocking process I/O | com.atlassian.bitbucket.thread-pools:name=NioPumpThreadPool |
| ScheduledThreadPool | Thread pool that takes care of several miscellaneous scheduled tasks | com.atlassian.bitbucket.thread-pools:name=ScheduledThreadPool |

Thread pool attributes

| Name | Description |
|--------------------|--|
| Active Count | Returns the approximate number of threads that are actively executing tasks |
| MaximumPoolSize | Returns the maximum allowed number of threads |
| PoolSize | Returns the current number of threads in the pool |
| Queue Length | The number of tasks awaiting execution by the thread pool |
| LargestPoolSize | The largest number of threads that have ever been simultaneously in the pool |
| CompletedTaskCount | The approximate total number of tasks that have completed execution. Because the states of tasks and threads may change dynamically during computation, the returned value is only an approximation, but one that does not ever decrease across successive calls |

Repositories (com.atlassian.bitbucket:name=Repositories)

| Name | Description |
|-------|---|
| Count | Number of repositories currently configured across all projects |

Scm Statistics (com.atlassian.bitbucket:name=ScmStatistics)

| Name | Description |
|--------|---|
| Pulls | Number of scm pulls serviced by this instance since it was started |
| Pushes | Number of scm pushes received by this instance since it was started |

Ticket statistics

Bitbucket uses 'tickets' as a mechanism for creating back-pressure to prevent the system from being overloaded with requests. There are two types of tickets, hosting tickets and command tickets.

Hosting tickets (com.atlassian.bitbucket:name=HostingTickets): Limits the number of SCM hosting operations, meaning pushes and pulls over HTTP or SSH, which may be running concurrently.

Command tickets (com.atlassian.bitbucket:name=CommandTickets): Limits the number of SCM commands, such as: ``git diff``, ``git blame``, or ``git rev-list``, which may be running concurrently.

Bitbucket supports the following metrics for each ticket type.

| Name | Description |
|---------------------|---|
| Available | The number of tickets available for acquisition (lower number means higher load) |
| LastRejection | The timestamp of the last rejected ticket, or null if no tickets have been rejected |
| Name | The name of the ticket bucket either 'scm-command' or 'scm-hosting' |
| OldestQueuedRequest | The timestamp at which the oldest queued request started waiting, or null if there are no queued requests |
| QueuedRequests | The number of requests currently waiting for an available ticket |

| | |
|-------|---|
| Total | The maximum number of tickets that can be acquired concurrently before back-pressure is applied |
| Used | The number of tickets that have been acquired (higher number means higher load) |

Event Statistics (com.atlassian.bitbucket:name=EventStatistics)

| Name | Description |
|------------------------|---|
| DispatchedCount | Total number of listener callbacks that have been performed. An event that is delivered to 10 listeners counts as 10 dispatches |
| LastRejection | Date of the last event being rejected, or <i>null</i> if no event has been rejected |
| PublishedCount | Total number of events delivered. An event that is delivered to 10 listeners counts as 1 event |
| QueueCapacity | Maximum number of event callbacks that can be queued before events are rejected |
| QueueLength | Number of event callbacks that have been queued but haven't been dispatched yet |
| RejectedCount | Total number of events that were not dispatched because the event queue was full |
| RemainingQueueCapacity | Remaining number of event callbacks that can be queued before events are rejected |

Cluster Lock Statistics (com.atlassian.bitbucket:name=ClusterLocks)

| Name | Description |
|------------------------|--|
| LockedCount | Number of cluster locks that are currently held by this node |
| QueuedThreadCount | Number of threads on this node that are currently blocked waiting for a lock |
| TotalAcquiredCount | Total number of times a cluster lock was acquired on this node since startup |
| TotalAcquireErrorCount | Number of times an exception was thrown while trying to acquire a cluster lock on this node since startup |
| TotalAcquireTimeMillis | Total time in milliseconds that any thread on this node has spent acquiring a lock (including time blocked waiting for a lock to become available) |
| TotalReleasedCount | Total number of times a cluster lock was released on this node since startup |
| TotalReleaseErrorCount | Total number of times an exception was thrown while releasing a cluster lock on this node since startup |

SSH Session Statistics (com.atlassian.bitbucket:name=SshSessions)

| Name | Description |
|-----------------------|---|
| ActiveSessionCount | Number of currently active SSH sessions |
| MaxActiveSessionCount | Highest number of concurrently active SSH sessions since the last startup |
| SessionClosedCount | Total number of SSH sessions that have been closed since the last startup |

| | |
|-----------------------|---|
| SessionCreatedCount | Total number of SSH sessions that have been created since the last startup |
| SessionExceptionCount | Total number of SSH sessions that have been terminated because an exception was thrown from the SSH command run |

Rate limiting statistics (com.atlassian.bitbucket:name=RateLimitStatistics)

| Name | Description |
|----------------------|---|
| RejectedRequestCount | The number of rate limited requests |
| UserMapSize | The number of token buckets currently in memory |

Interesting 3rd party library attributes

Bitbucket exposes the JMX attributes from number of third party libraries. Listed below is a sample of the attributes that are particularly interesting from an operations perspective.

HikariCP -(com.zaxxer.hikari:type=Pool (bitbucket))

| Name | Description |
|---------------------------|--|
| ActiveConnections | Active Connections (in use) |
| IdleConnections | Idle Connection count |
| ThreadsAwaitingConnection | The number of threads waiting for a connection (when all available connections are in use) |
| TotalConnections | Total Connections |

Hibernate - (org.hibernate.core:sessionFactory=bitbucket.core,serviceRole=org.hibernate.stat.Statistics,serviceType=org.hibernate.stat.internal.ConcurrentStatisticsImpl)

| Name | Description |
|---------------------------|--|
| QueryCacheHitCount | Global number of cached queries successfully retrieved from cache |
| QueryCacheMissCount | Global number of cached queries <i>not</i> found in cache |
| SecondLevelCacheHitCount | Global number of cacheable entities/collections successfully retrieved from the cache |
| SecondLevelCacheMissCount | Global number of cacheable entities/collections <i>not</i> found in the cache and loaded from the database |

Expose JMX MBeans within Bitbucket

To enable Bitbucket to publish specific statistics using JMX:

1. Locate and open the `bitbucket.properties` file in the `<Bitbucket home directory>/shared` directory.
 - a. Add this property to the file.

```
jmx.enabled=true
```

- b. Save and close the file.

2. Create a JMX password file for secure access to JMX monitoring.

3. Modify the `set-jmx-opts.sh` file to enable Bitbucket to expose JMX Mbeans.

These changes will not take effect until Bitbucket is restarted.

Set up the JMX password file

To set up a JMX password file to secure access to JMX monitoring

1. Create a file named `jmx.access`.

 This file will contain password information. Ensure the file is only readable by the secure user Bitbucket will run under. However, note that if the user cannot read the file Bitbucket will fail to start.

2. Edit the `jmx.access` file to include this property and save the file.

```
monitorRole <password>
```

If you wish to use a username other than `monitorRole` or `controlRole` you will need to modify the `jmremote.access` file located in the `/lib/management/` directory of the installed Java.

3. Change ownership of `jmx.access` file,

```
chown bitbucket:bitbucket <path>/jmx.access
```

where `bitbucket` is the user that runs Bitbucket service.

4. Change file permissions of `jmx.access` file.

```
chmod 600 <path>/jmx.access
```

Modify the Bitbucket environment file

To modify the `set-jmx-opts.sh` (for Windows `set-jmx-opts.bat`) files to enable JMX monitoring:

1. Within the `bin` directory, locate the file `set-jmx-opts.sh` (for Windows `set-jmx-opts.bat`) and change these properties.

```
JMX_REMOTE_AUTH=password
JMX_REMOTE_PORT=3333
RMI_SERVER_HOSTNAME=-Djava.rmi.server.hostname=<hostname>
JMX_PASSWORD_FILE=<path>/jmx.access
```

2. Restart Bitbucket.

Expose JMX MBeans when Bitbucket is run as a Windows service

To expose JMX MBeans when Bitbucket is run as a Windows service

1. Stop the Bitbucket service.
2. Open the command line prompt and enter.

```
cmd
```

3. Navigate to the Bitbucket `bin` directory.

```
cd <Bitbucket installation dir>\bin
```

4. Run this command.

```
tomcat8w //ES//AtlassianBitbucket Server
```

5. In the window that appears, click on the Java tab to see the list of current startup options. Under "Java Options:" form, input the value

```
-Dcom.sun.management.jmxremote.port=<JMX_REMOTE_PORT>  
-Djava.rmi.server.hostname=<hostname>  
-Dcom.sun.management.jmxremote.ssl=false  
-Dcom.sun.management.jmxremote.password.file=<JMX_PASSWORD_FILE>
```



Ensure the owner of this password file is the secure user Bitbucket will run as. If the user cannot read the file, Bitbucket will fail to start.

6. Replace the values within the < > characters.

```
JMX_REMOTE_PORT=3333  
JMX_PASSWORD_FILE=<path>\jmx.access
```

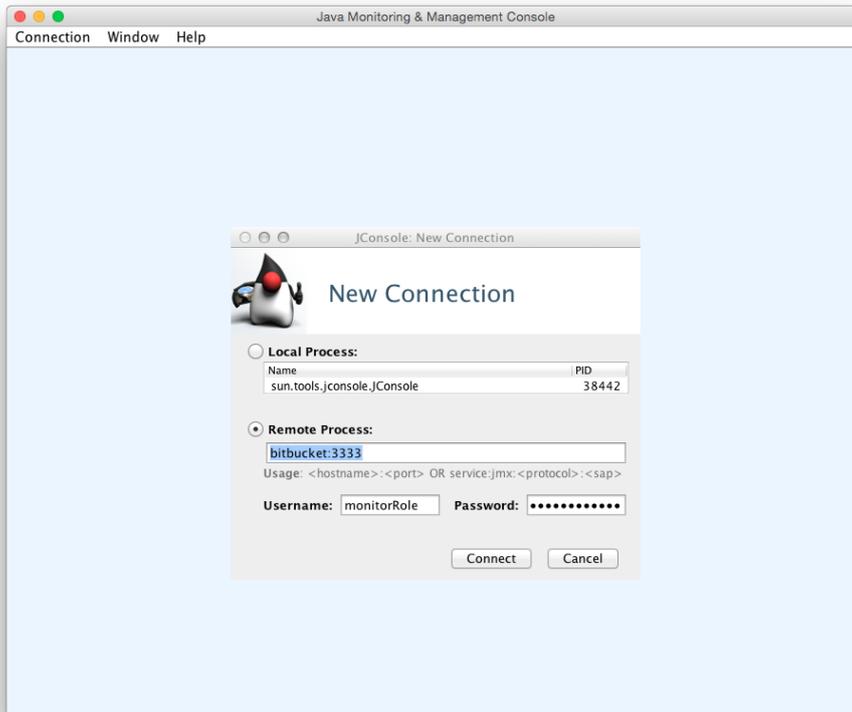
7. Restart the Bitbucket service.

Verify JMX is configured correctly

These steps use JConsole to test that JMX has been configured correctly. JConsole is a utility that ships with the Oracle JDK.

1. To start the jconsole utility, from a command line prompt enter

```
jconsole
```



2. Create a new JConsole connection with similar connection settings.

| | |
|---------------------------|--|
| bitbucket | the hostname of the instance to monitor |
| 3333 | the JMX port number previously configured. |
| <i>username, password</i> | values configured within the JMX password file <code>jmx.access</code> . |

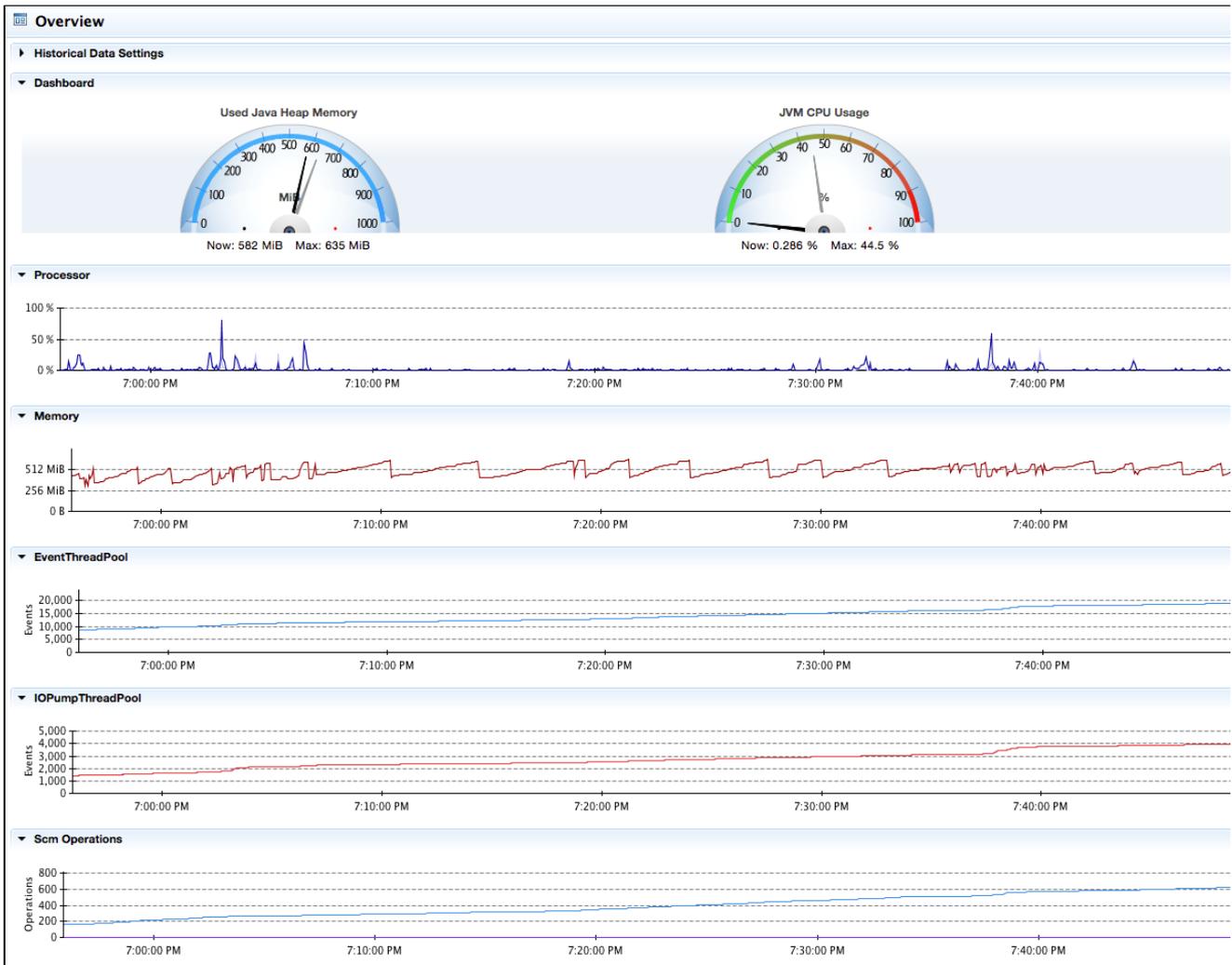
3. Click **Connect**.

When configured correctly, you will see these properties.

| | |
|--------------------------------------|--|
| com.atlassian.bitbucket | <ul style="list-style-type: none"> • CommandTickets • HostingTickets • Projects • Repositories • ScmStatistics • Tickets • EventStatistics • ClusterLocks • SshSessions |
| com.atlassian.bitbucket.thread-pools | <ul style="list-style-type: none"> • EventThreadPool • IoPumpThreadPool • ScheduledThreadPool |

Example performance dashboard

This dashboard was generated using [Java Mission Control](#) that ships with the Oracle JDK (since 1.7u40). See the documentation that comes with your JMX client of choice for more information.



Configuring JMX to use SSL

You can find information about the options for configuring JMX to use SSL in the `set-jmx-opts` files. Comprehensive documentation is [available from Oracle](#).

Enable debug logging

This page describes how to enable various types of debug-level logging in Bitbucket Data Center and Server.

Logs can be found in `<Bitbucket home directory>/log`.

On this page

- [Debug logging for your instance](#)
- [Profiling logging for your instance](#)
- [Debug logging for Git operations on the client](#)
- [Debug logging for your Backup Client](#)

Debug logging for your instance

This section describes how to enable debug level logging in Bitbucket.

Enabling debug logging via the UI

To enable debug logging:

1. Go to the  > **Logging and Profiling**.
2. Select **Enable debug logging**.

Enabling debug logging on startup

To enable debug logging whenever Bitbucket. is started, edit the `<Bitbucket home directory>/shared/bitbucket.properties` file (if this file doesn't exist then you should create it) and add these two lines:

```
logging.logger.ROOT=DEBUG
logging.logger.com.atlassian.bitbucket=DEBUG
```

If your instance is earlier than version 3.2, the `bitbucket.properties` file is at the top level of your [home directory](#).

Enabling debug logging at runtime

To enable debug logging for the root logger once Bitbucket. has been started, run these two commands in your terminal:

```
curl -u <ADMIN_USERNAME> -v -X PUT -d "" -H "Content-Type: application/json" <BASE_URL>/rest/api/latest/logs/rootLogger/debug
curl -u <ADMIN_USERNAME> -v -X PUT -d "" -H "Content-Type: application/json" <BASE_URL>/rest/api/latest/logs/logger/com.atlassian.bitbucket/debug

# e.g.
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json" http://localhost:7990/rest/api/latest/logs/rootLogger/debug
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json" http://localhost:7990/rest/api/latest/logs/logger/com.atlassian.bitbucket/debug
```

To enable debug logging for a specific logger, run this command in a terminal:

```
curl -u <ADMIN_USERNAME> -v -X PUT -d "" -H "Content-Type: application/json" <BASE_URL>/rest/api/latest/logs/logger/<LOGGER_NAME>/debug

# e.g. embedded Crowd debug log
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json" http://localhost:7990/rest/api/latest/logs/logger/com.atlassian.crowd/debug

# e.g. LDAP debug log
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json" http://localhost:7990/rest/api/latest/logs/logger/com.atlassian.crowd.directory.SpringLDAPConnector/DEBUG
# e.g. email debug log
curl -u admin -v -X PUT -d "" -H "Content-Type: application/json" http://localhost:7990/rest/api/latest/logs/logger/bitbucket.mail-log/debug
```

Profiling logging for your instance

This section describes how to enable profiling in Bitbucket. This log is essential when troubleshooting performance issues.

Enabling profiling logging via the UI

To enable profiling:

1. Go to the  > **Logging and Profiling**.
2. Select **Enable profiling**.

Debug logging for Git operations on the client

Atlassian Support might request debug logs for Git operations (on the client) when troubleshooting issues. You can enable debug logging on the Git client by setting the following variables. If you are using HTTP/S, remove the `Authorization` header from the output, as it contains your Basic-Auth information. Atlassian provides a set of [scripts](#) that simplify the collection of Git client debug information.

On Linux

Execute the following on the command line before executing the Git command:

```
export GIT_TRACE_PACKET=1
export GIT_TRACE=1
export GIT_CURL_VERBOSE=1
```

To measure the length of time a command takes to complete, use the `timeutility/command` prior to the actual git command. For example, to test a push to branch "master":

```
time git push origin master
```

On Windows

Execute the following on the command line before executing the Git command:

```
set GIT_TRACE_PACKET=1
set GIT_TRACE=1
set GIT_CURL_VERBOSE=1
```



Setting `GIT_CURL_VERBOSE` is only useful for connections over HTTP/S since SSH doesn't use the `libcurl` library.

Debug logging for your Backup Client

Atlassian Support might request debug logs for the Backup client when troubleshooting issues.

To enable debug logging for the Backup client, add a file named `logback.xml` to your working directory(`pwd`) with the following content:

logback.xml

```
<included><logger name="com.atlassian.bitbucket" level="DEBUG"/></included>
```

Scaling Bitbucket Server

This page discusses performance and hardware considerations when using Bitbucket Server.

Note that [Bitbucket Data Center resources](#), not discussed on this page, uses a cluster of Bitbucket Server nodes to provide Active/Active failover, and is the deployment option of choice for larger enterprises that require high availability and performance at scale.

Hardware requirements

The type of hardware you require to run Bitbucket Server depends on a number of factors:

- The count and concurrency of clone operations, which are the most resource-intensive operation Bitbucket Server performs. One major source of clone operations is continuous integration. When your CI builds involve multiple parallel stages, Bitbucket Server will be asked to perform multiple clones concurrently, putting significant load on your system.
- The size of your repositories. There are many operations in Bitbucket Server that require more CPU, memory and I/O when working with very large repositories. Furthermore, huge Git repositories (larger than a few GBs) are likely to impact the performance of Git clients as well as Bitbucket Server.
- The number of users.

Here are some rough guidelines for choosing your hardware:

- Estimate the number of concurrent clones that are expected to happen regularly (look at continuous integration). Add one CPU for every 2 concurrent clone operations.
- Estimate or calculate the average repository size and allocate 1.5 x number of concurrent clone operations x min(repository size, 700MB) of memory.
- If you're running Bitbucket Data Center, check your size using the [Bitbucket Data Center load profiles](#). If your instance is Large or XLarge, take a look at our infrastructure recommendations for [Bitbucket Data Center AWS deployments](#).

See [Scaling Bitbucket Server for Continuous Integration performance](#) for some additional information about how Bitbucket Server's SCM cache can help the system scale.

Understanding Bitbucket Server's resource usage

Most of the things you do in Bitbucket Server involve both the Bitbucket Server instance and one or more Git processes. For instance, when you view a file in the web application, Bitbucket Server processes the incoming request, performs permission checks, creates a Git process to retrieve the file contents and formats the resulting webpage. The same is true for the 'hosting' operations like pushing commits, cloning a repository, or fetching the latest changes.

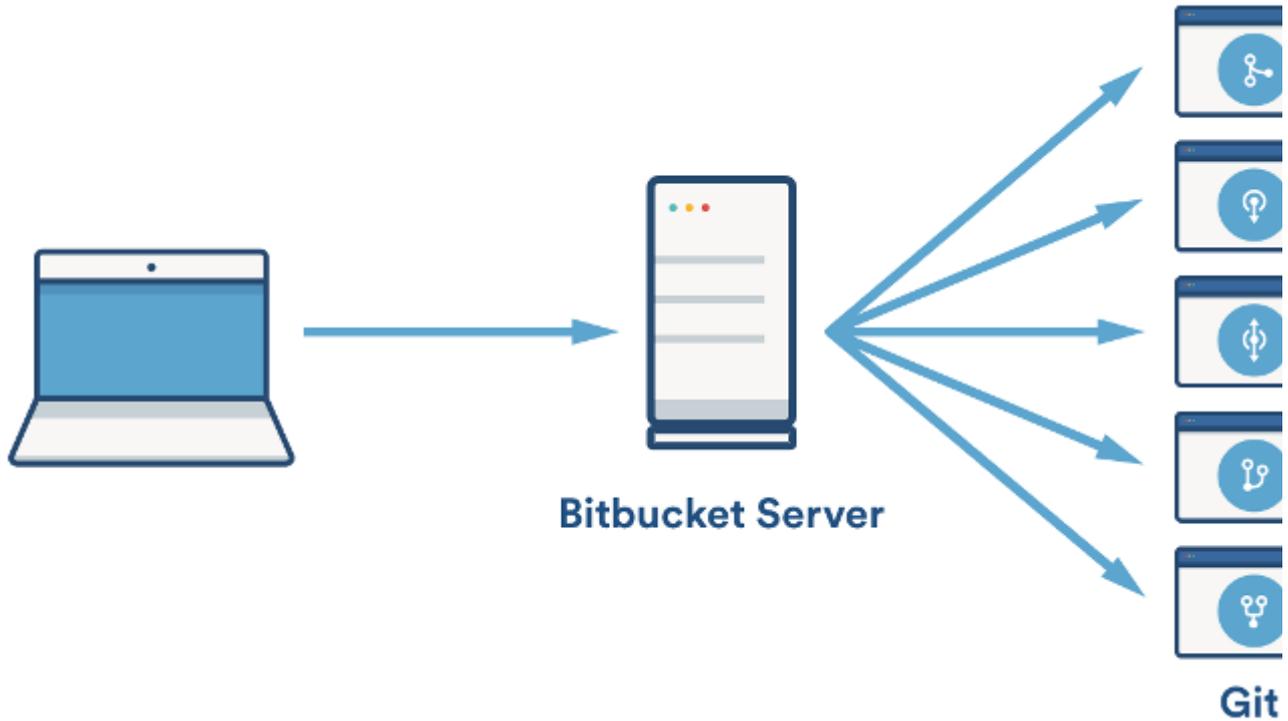
As a result, when configuring Bitbucket Server for performance, CPU and memory consumption for both Bitbucket Server *and* Git should be taken into account.

On this page:

- [Hardware requirements](#)
- [Understanding Bitbucket Server's resource usage](#)
- [Clones examined](#)
- [Monitoring](#)
- [Tickets and throttling](#)
- [Caching](#)
- [HTTPS and SSH](#)
- [Configuring Bitbucket Server scaling options and system properties](#)

Related pages:

- [Use Bitbucket in the enterprise](#)
- [Resources for migrating to Git](#)
- [Bitbucket Server production server data](#)
- [Scaling Bitbucket Server for Continuous Integration performance](#)
- [Potential performance impact of embedded Crowd directory ordering](#)
- [Configuration properties](#)



CPU

In Bitbucket Server, much of the heavy lifting is delegated to Git. As a result, when deciding on the required hardware to run Bitbucket Server, the CPU usage of the Git processes is the most important factor to consider. Cloning repositories is the most CPU intensive Git operation. When you clone a repository, Git on the server side will create a *pack file* (a compressed file containing all the commits and file versions in the repository) that is sent to the client. Git can use multiple CPUs while compressing objects to generate a pack, resulting in spikes of very high CPU usage. Other phases of the cloning process are single-threaded and will, at most, max out a single CPU.

Encryption (either SSH or HTTPS) may impose a significant CPU overhead if enabled. As for whether SSH or HTTPS should be preferred, there's no clear winner. Each has advantages and disadvantages as described in the following table:

| | HTTP | HTTPS | SSH |
|-----------------------|---|---|---|
| Encryption | No CPU overhead for encryption, but plain-text transfer and basic authentication may be unacceptable for security. | Encryption has CPU overhead, but this can be offloaded to a separate proxy server (if the SSL/TLS is terminated there). | Encryption has CPU overhead. |
| Authentication | Authentication is slower it requires remote authentication with the LDAP or Crowd server. | | Authentication is generally faster, but may still require an LDAP or Crowd request to verify the connecting user is still active. |
| Cloning | Cloning a repository is slightly slower over HTTP. It requires at least 2 separate requests and potentially significantly more each performing its own authentication and permission checks. The extra overhead is typically small, but depends heavily on the latency between client and server. | | Cloning a repository takes only a single request. |

Memory

When deciding on how much memory to allocate for Bitbucket Server, the most important factor to consider is the amount of memory required for Git. Some Git operations are fairly expensive in terms of memory consumption, most notably the initial push of a large repository to Bitbucket Server and cloning large repositories from Bitbucket Server. For large repositories, it is not uncommon for Git to use hundreds of megabytes, or even multiple gigabytes, of memory during the clone process. The numbers vary from repository to repository, but as a rule of thumb 1.5x the repository size on disk (contents of the `.git/objects` directory) is a reasonable initial estimate of the required memory for a single clone operation. For large repositories, or repositories that contain large files, memory usage is effectively only bounded by the amount of RAM in the system.

In addition to being the most CPU-intensive, cloning repositories is also the most memory intensive Git operation. Most other Git operations, such as viewing file history, file contents and commit lists are lightweight by comparison. Clone operations also tend to retain their memory for significantly longer than other operations.

Bitbucket Server has been designed to have fairly stable memory usage. Pages that could show large amounts of data (e.g. viewing the source of a multi-megabyte file) perform incremental loading or have hard limits in place to prevent Bitbucket Server from holding on to large amounts of memory at any time. In general, the default memory settings (`-Xmx1g`) should be sufficient to run Bitbucket Server. Installing third-party apps may increase the system's memory usage. The maximum amount of memory available to Bitbucket Server can be configured in `_start-webapp.sh` or `_start-webapp.bat`.



The memory consumption of Git is not managed by the memory settings in `_start-webapp.sh` or `_start-webapp.bat`. Git processes are executed outside the Java virtual machine, so JVM memory settings do not apply.

Allocating a large heap for Bitbucket Server's JVM may constrain the amount of memory available for Git processes, which may result in poor performance. A heap of 1-2GB is generally sufficient for Bitbucket Server's JVM.

Disk

Git repository data is stored entirely on the filesystem. Storage with low latency/high IOPS will result in significantly better repository performance, which translates to faster overall performance and improved scaling. Storage with high latency is generally unsuitable for Git operations, even if it can provide high throughput, and will result in poor repository performance. Filesystems like Amazon EFS are not recommended for Bitbucket Server's home or shared home due to their high latency.

Available disk space in `$BITBUCKET_HOME/caches`, where Bitbucket Servers SCM cache stores packs to allow them to be reused to serve subsequent clones, is also important for scaling. The SCM cache allows Bitbucket Server to trade increased disk usage for reduced CPU and memory usage, since streaming a previously-built pack uses almost no resources compared to creating a pack. When possible, `$BITBUCKET_HOME/caches` should have a similar amount of total disk space to `$BITBUCKET_HOME/shared/data/repositories`, where the Git repositories are stored.

Network

Cloning a Git repository, by default, includes the *entire history*. As a result, Git repositories can become quite large, especially if they're used to track binary files, and serving clones can use a significant amount of network bandwidth.

There's no fixed bandwidth threshold we can document for the system since it will depend heavily on things like; repository size, how heavy CI (Bamboo, Jenkins, etc.) load is, and more. However, it's worth calling out that Bitbucket Servers network usage will likely far exceed other Atlassian products like Jira or Confluence.

Additionally, when configuring a Data Center cluster, because repository data must be stored on a shared home which is mounted via NFS, Bitbucket Data Center's bandwidth needs are even higher -and its performance is far more sensitive to network latency. Ideally, in a Data Center installation, nodes would use separate NICs and networks for client-facing requests (like hosting) and NFS access to prevent either from starving the other. The NFS network configuration should be as *low latency* as possible, which excludes using technologies like Amazon EFS.

Database

The size of the database required for Bitbucket Server primarily depends on the number of repositories the system is hosting and the number of commits in those repositories.

A very rough guideline is: $100 + ((\text{total number of commits across all repositories}) / 2500)$ MB.

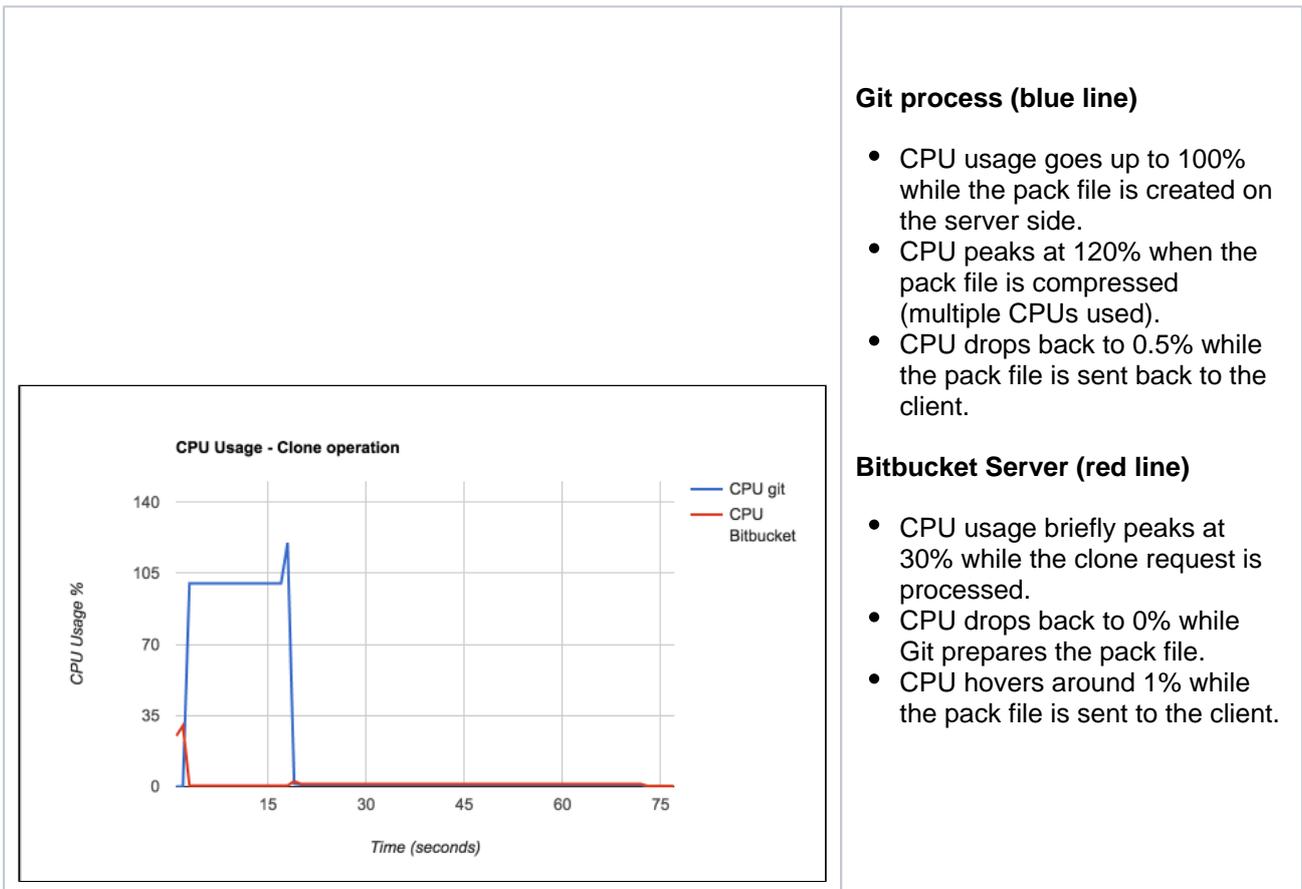
So, for example, for 20 repositories with an average of 25,000 commits each, the database would need $100 + (20 * 25,000 / 2500) = 300$ MB.

Note that repository data is *not* stored in the database; its stored on the filesystem. As a result, having multi-gigabyte repositories does not necessarily mean the system will use dramatically more database space.

Where possible, it is *preferable* to have Bitbucket Servers database on a separate machine or VM, so the two are not competing for CPU, memory and disk I/O.

Clones examined

Since cloning a repository is the most demanding operation in terms of CPU and memory, it is worthwhile analyzing the clone operation a bit closer. The following graphs show the CPU and memory usage of a clone of a 220 MB repository:

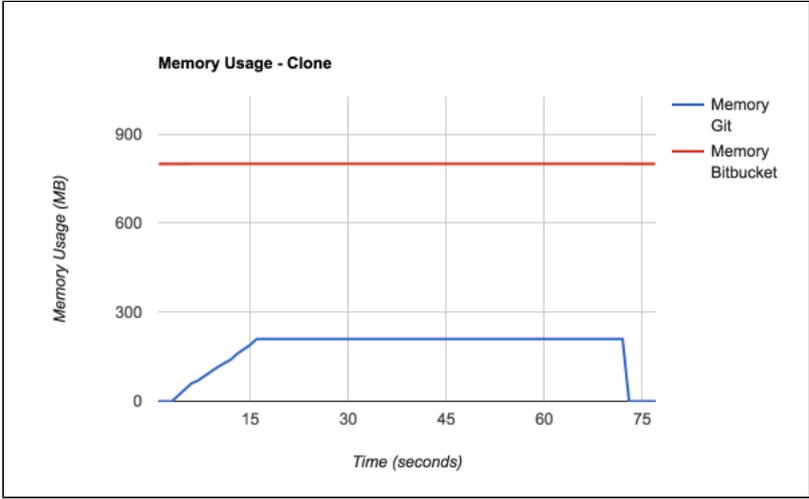


Git process (blue line)

- CPU usage goes up to 100% while the pack file is created on the server side.
- CPU peaks at 120% when the pack file is compressed (multiple CPUs used).
- CPU drops back to 0.5% while the pack file is sent back to the client.

Bitbucket Server (red line)

- CPU usage briefly peaks at 30% while the clone request is processed.
- CPU drops back to 0% while Git prepares the pack file.
- CPU hovers around 1% while the pack file is sent to the client.

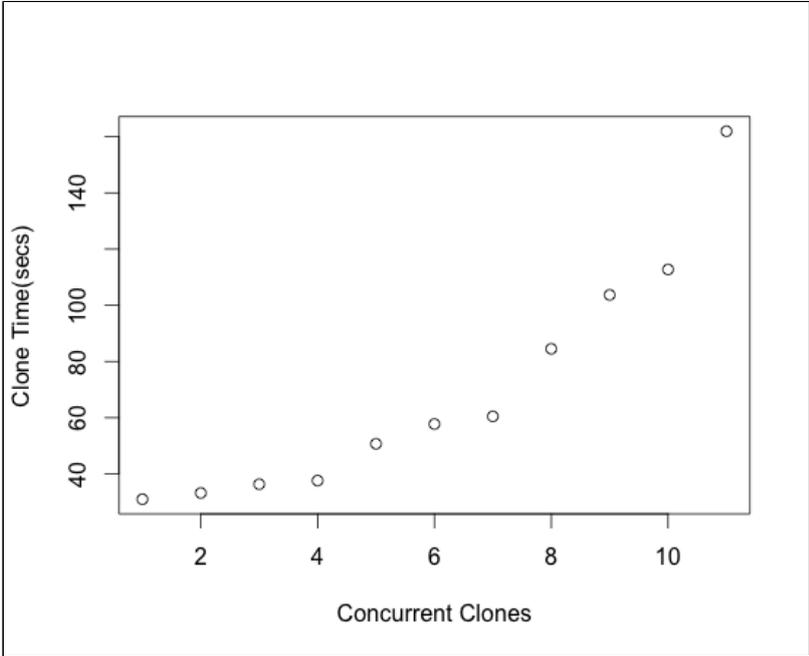


Git process (blue line)

- Memory usage slowly climbs to 270 MB while preparing the pack file.
- Memory stays at 270 MB while the pack file is transmitted to the client.
- Memory drops back to 0 when the pack file transmission is complete.

Bitbucket Server (red line)

- Memory usage hovers around 800 MB and is not affected by the clone operation.



This graph shows how concurrency affects average response times for clones:

- Vertical axis: average response times.
- Horizontal axis: number of concurrent clone operations.

The measurements for this graph were done on a 4 CPU server with 12 GB of memory. Response times become exponentially worse as the number of concurrent clone operations exceed the number of CPUs.

Monitoring

In order to effectively diagnose performance issues and tune Bitbucket Servers scaling settings, it is important to configure monitoring. While exactly how to set up monitoring is beyond the scope of this page, there are some guidelines that may be useful:

- At a minimum, monitoring should include data about CPU, memory, disk I/O (for any disks where Bitbucket Server is storing data), free disk space, and network I/O

- Monitoring free disk space can be very important for detecting when the SCM cache is nearing free space limits, which could result in it being automatically disabled
- When Bitbucket Server is used to host large repositories, it can consume a large amount of network bandwidth. If repositories are stored on NFS, for a Data Center cluster, bandwidth requirements are even higher
- Bitbucket Server exposes [many JMX counters](#) which may be useful for assembling dashboards to monitor overall system performance and utilization
 - In particular, tracking used/free tickets for the various buckets, described below, can be very useful for detecting unusual or escalating load
- Retaining *historical data* for monitoring can be very useful for helping to track increases in resource usage over time as well as detecting significant shifts in performance
 - As users create more repositories, push more commits, open more pull requests and generally just use the system, resource utilization will increase over time
 - Historical averages can be useful in determining when the system is approaching a point where additional hardware may be required or, for Data Center installations, when it may be time to consider adding another cluster node

Tickets and throttling

Bitbucket Server uses a ticket-based approach to throttling requests. The system uses a limited number of different ticket buckets to throttle different types of requests independently, meaning one request type may be at or near its limit, while another type still has free capacity.

Each ticket bucket has a default size that will be sufficient in many systems, but as usage grows, the sizes may need to be tuned. In addition to a default size, each bucket has a default timeout which defines the longest a client request is allowed to wait to acquire a ticket before the request is rejected. Rejecting requests under heavy load helps prevent cascading failures, like running out of Tomcat request threads because too many requests are waiting for tickets.

Ticket buckets

The following table shows ticket buckets the system uses, the default size and acquisition timeout, and what each is used for:

| Bucket | Size | Timeout | Usage |
|-------------|------------|-----------|---|
| scm-command | 40 (Fixed) | 2 seconds | <p>scm-command is used to throttle most of the day-to-day Git commands the system runs. For example:</p> <ul style="list-style-type: none"> • <code>git diff</code>, used to show commit and pull request diffs • <code>git rev-list</code>, used to show commit lists and information about specific commits • <code>git merge</code>, used to merge pull requests <p>scm-command tickets are typically <i>directly connected</i> to web UI and REST requests, and generally have very quick turnaround - most commands typically complete in tens to hundreds of milliseconds. Because a user is typically waiting, scm-command tickets apply a very short timeout in order to favor showing users an error over displaying spinners for extended periods.</p> |

| | | | |
|-------------|-----------------------------|-----------|---|
| scm-hosting | 1x-4x (Adaptive; see below) | 5 minutes | <p>scm-hosting is used to throttle <code>git clone</code> and <code>git fetch</code>. During a clone or fetch, after sending a ref advertisement, the server generates a pack file (on the fly) which contains the objects the client has requested. For large repositories, this can be a very CPU, memory and I/O-intensive operation. Servicing clones and fetches produces the majority of the load on most Bitbucket Server instances.</p> <p>For SSH only, scm-hosting is also used to throttle <code>git upload-archive</code> requests. <code>git upload-archive</code> does not currently support HTTP(S) remotes. Generating an archive is less resource-intensive than generating a pack, but is likely to still use more resources (and for longer) than serving a ref advertisement or push.</p> <p>scm-hosting uses an adaptive throttling mechanism (described in detail below) which allows the system to dynamically adjust the number of available tickets in response to system load. The default range is proportional to a configurable scaling factor, which defaults to the number of CPUs reported by the JVM. For example, if the JVM reports 8 CPUs, the system will default to $1 \times 8 = 8$ tickets minimum and $4 \times 8 = 32$ tickets maximum.</p> |
| scm-refs | 8x (Fixed proportionally) | 1 minute | <p>scm-refs is used to throttle ref advertisements, which are the first step in the process of servicing both pushes and pulls.</p> <p>Additionally, because most of the CPU and memory load are client side, <i>pushes</i> are throttled using the scm-refs bucket. Unlike a clone or a fetch, the pack for a push is generated using the clients CPU, memory and I/O. While processing the received pack does produce load on the server side, its minimal compared to generating a pack for a clone or fetch.</p> <p>The default size for the scm-refs bucket is proportional to a configurable scaling factor, which defaults to the number of CPUs reported by the JVM. For example, if the JVM reports 8 CPUs, the system will default to $8 \times 8 = 64$ scm-refs tickets.</p> <p>Ref advertisements are generally served fairly quickly, even for repositories with large numbers of refs, so the default timeout for scm-refs is shorter than the default for scm-hosting.</p> |
| git-lfs | 80 (Fixed) | Immediate | <p>git-lfs is used to throttle requests for large objects using Git LFS. LFS requests are much more similar to a basic file download than a pack request, and produce little system load. The primary reason they're throttled at all is to prevent large numbers of concurrent LFS requests from consuming all of Tomcat's limited HTTP request threads, thereby blocking access to users trying to browse the web UI, or make REST or hosting operations.</p> <p>Because LFS is predominantly used for large objects, the amount of time a single LFS ticket may be held can vary widely. Since its hard to make a reasonable guess about when a ticket might become available, requests for git-lfs tickets timeout immediately when the available tickets are all in use.</p> |

| | | | |
|----------------|---------------------------|--------|---|
| mirror-hosting | 2x (Fixed proportionally) | 1 hour | <p>"mirror-hosting" is <i>Data Center-only</i> and is used to throttle <code>git clone</code> and <code>git fetch</code> requests from smart mirrors and mirror farms. Using a separate bucket for mirror requests allows different configuration, like using a longer timeout. No user would wait an hour to acquire a ticket, but mirrors will. A separate bucket ensures busy mirrors don't consume all of the system's "scm-hosting" tickets and prevent users or CI from being able to push or pull.</p> <p>Unlike "scm-hosting", "mirror-hosting" is <i>not</i> adaptive. It uses a fixed number of tickets based on the number of CPUs reported by the JVM. For example, if the JVM reports 8 CPUs, the system will default to $2 \times 8 = 16$ "mirror-hosting" tickets. The default limit is generally sufficient, but for instances with a large number of mirrors, or large mirror farms, it may be necessary to increase it. Administrators will need to balance the number of "mirror-hosting" tickets they allow against the number of "scm-hosting" tickets they allow to prevent excessive combined load between the two.</p> |
|----------------|---------------------------|--------|---|

Earlier versions

Prior to Bitbucket Server 7.3, "scm-hosting" tickets were used to throttle all parts of hosting operations, including ref advertisements and pushes. This meant that the "scm-hosting" bucket often needed to be sized very generously to prevent fast-completing ref advertisements from getting blocked behind slow-running clones or fetches when competing for tickets. However, when the "scm-hosting" limit was very high, if a large number of clone or fetch requests were initiated concurrently, it could result in a load spike that effectively crippled or even crashed the server. "scm-refs" tickets were introduced to combat that risk. With "scm-refs", administrators can configure the system to allow for heavy polling load (typically from CI servers like Bamboo or Jenkins) without necessarily increasing the number of available "scm-hosting" tickets.

Adaptive throttling

Adaptive throttling uses a combination of total physical memory, evaluated once during startup, and CPU load, evaluated periodically while the system is running, to dynamically adjust the number of available "scm-hosting" tickets within a configurable range.

During startup, the total physical memory on the machine is used to determine the maximum number of tickets the machine can safely support. This is done by considering how much memory Bitbucket Server and Elasticsearch need for their JVMs and an estimate of how much memory each Git hosting operation consumes on average while running, and may produce a safe upper bound that is lower (but never higher) than the configured upper bound.

To illustrate this more concretely, consider a system with 8 CPU cores and 8GB of physical RAM. With 8 CPU cores, the default adaptive range will be 8-32 tickets.

- The total is reduced by 1GB for Bitbucket Server's default heap: 7GB
- The total is reduced by 512MB for Elasticsearch: 6.5GB
- The remainder is divided by 256MB: $6656 / 256 = 26$

In this example, the actual upper bound for the adaptive range will be 26 tickets, rather than the 32 calculated from CPU cores, because the system doesn't have enough RAM to safely handle 32 tickets.

While the system is running, Bitbucket Server periodically samples CPU usage (every 5 seconds by default) and increases or decreases the number of available tickets based on a target load threshold (75% by default). A smoothing factor is applied to CPU measurements so the system doesn't overreact by raising or lowering the number of available tickets too aggressively in response to bursty load.

[Adaptive throttling](#) is enabled by default, but the system may automatically revert to [fixed throttling](#) if any of the following conditions are met:

- A non-default fixed number of tickets has been set; for example `throttle.resource.scm-hosting=25`

- A fixed throttling strategy is configured explicitly; for example `throttle.resource.scm-hosting.strategy=fixed`
`throttle.resource.scm-hosting.fixed.limit=25`
- The adaptive throttling configuration is invalid in same way
- The total physical memory on the machine is so limited that even the minimum number of tickets is considered unsafe

Adaptive throttling is only available for the "scm-hosting" ticket bucket. Other buckets, like "scm-refs", do not support adaptive throttling; they use fixed limits for the number of tickets. This prevents high CPU usage from `git clone` and `git fetch` requests from reducing the number of tickets available in other buckets, which generally don't use much CPU.

Caching

Building pack files to serve clone requests is one of the most resource-intensive operations Bitbucket Server performs, consuming significant amounts of CPU, memory and disk I/O. To reduce load, and allow instances to service more requests, Bitbucket Server can cache packs between requests so they only need to be built once. When a pack is served from the cache, no "scm-hosting" ticket is used. This can be particularly beneficial for systems with heavy continuous integration (CI) load, from systems like Bamboo or Jenkins, where a given repository may be cloned several times either concurrently or in short succession.

Cached packs are stored in `$(BITBUCKET_HOME)/caches/scm`, with individual subdirectories for each repository. On Data Center nodes, packs are cached *per node* and are not shared. If free space on the disk where `$(BITBUCKET_HOME)/caches/scm` is stored falls below a configurable threshold, pack file caching will be automatically disabled until free space increases. Cached packs will automatically be evicted using a least-recently-used (LRU) strategy to try and free up space when free space approaches the threshold.

Considerations

- Because clones typically include the full history for the repository, cached packs are often close to the same size as the repository being cloned. This means cached packs can consume a significant amount of disk space often more than the repository itself consumes if multiple packs are cached
 - It may be desirable to use a separate disk or partition mounted at `$(BITBUCKET_HOME)/caches` to allow for more disk space and to ensure cached packs don't fill up the same disk where other system data is stored
- Using single-branch clones (e.g. `git clone --single-branch`) can result in a large number of distinct cached packs for a single repository. In general, for maximizing cache hits, it's better to use full clones

Limitations

- Pack files for `git fetch` requests are not cached. Unlike clones, where it's likely the same clone will be executed multiple times, fetches tend to be much more unique and are unlikely to produce many cache hits

Earlier versions

Prior to Bitbucket 7.4 the system supported caching ref advertisements as well as packs. Unlike pack file caching, enabling ref advertisement caching did little to reduce system load. Instead, the primary benefit of ref advertisement caching was reduced contention for "scm-hosting" tickets. Ref advertisement caching was disabled by default because it could result in advertising stale refs. This meant administrators had to balance the risk of stale data against the reduction in "scm-hosting" ticket usage.

Bitbucket Server 7.3 introduced a new "scm-refs" bucket for throttling ref advertisements, eliminating contention for "scm-hosting" tickets, and Bitbucket Server 7.4 introduced significant reductions in the number of threads used to service HTTP and SSH hosting operations. The combination of those improvements eliminate the benefits of ref advertisement caching, leaving only its downsides, so support for ref advertisement caching has been removed.

HTTPS and SSH

Bitbucket Server can serve hosting operations via HTTPS and SSH protocols. Each has its pros and cons, and it's possible to disable serving hosting operations via either protocol if desired.

HTTPS

Serving hosting operations via HTTPS requires 2 or more requests to complete the overall operation. By default Git will attempt to reuse the same connection for subsequent requests, to reduce overhead, but it's not always possible to do so. Additionally, Git does not support `upload-archive` over HTTP(S); it's only available over SSH. One advantage of HTTPS for hosting is that encryption overhead can be offloaded to a proxy to reduce CPU load on the Bitbucket Server instance.

Git's HTTPS support offers 2 wire protocols, referred to as "smart" and "dumb". Bitbucket Server only supports the "smart" wire protocol, which has 2 versions: v0 and v2 (v1 was a transitional protocol and is not generally used). The v0 "smart" wire protocol is always supported; v2 is only supported when Git 2.18+ is installed on both Bitbucket Server *and on clients*.

By default, Tomcat allows up to 200 threads to process incoming requests. Bitbucket Server 7.4 introduced the use of asynchronous requests to move processing for hosting operations to a background threadpool, freeing up Tomcat's threads to handle other requests (like web UI or REST requests). The background threadpool allows 250 threads by default. If the background threadpool is fully utilized, subsequent HTTPS hosting operations are handled directly on Tomcat's threads. When all 200 Tomcat threads are in use, a small number of additional requests are allowed to queue (50 by default) before subsequent requests are rejected.

SSH

Hosting operations via SSH are handled using a single request, with bidirectional communication between the client and server. SSH supports the full range of hosting operations: `receive-pack` (push), `upload-archive` (archive) and `upload-pack` (pull). One disadvantage of SSH is that its encryption overhead cannot be offloaded to a proxy; it must be handled by the Bitbucket Server JVM.

Git's SSH support only offers a single wire protocol, which is roughly equivalent to HTTPS's "smart" wire protocol. As with the HTTPS "smart" wire protocol, Git's SSH wire protocol supports 2 versions: v0 and v2. The v0 wire protocol is always supported; v2 is only supported when Git 2.18+ is installed on both Bitbucket Server *and on clients*.

By default, Bitbucket Server allows up to 250 simultaneous SSH sessions, and sessions over that limit are rejected to prevent backlogs.

Earlier versions

Prior to Bitbucket Server 7.4, hosting operations via HTTPS and SSH used 5 threads per request. One of these was the actual HTTPS or SSH request thread, and the rest were overhead related to using blocking I/O to communicate with the `git` process that was servicing the request. In Bitbucket Server 7.4, that blocking I/O approach was replaced with a non-blocking approach which eliminated the 4 overhead threads, allowing HTTPS and SSH hosting operations to be serviced by a single thread.

Configuring Bitbucket Server scaling options and system properties

The sizes and timeouts for the various ticket buckets are all configurable; see [Configuration properties](#).

When the configured limit is reached for the given resource, requests will wait until a currently running request has completed. If no request completes within a configurable timeout, the request will be rejected. When requests while accessing the Bitbucket Server UI are rejected, users will see either a 501 error page indicating the server is under load, or a popup indicating part of the current page failed to load. When Git client 'hosting' commands (pull/push/clone) are rejected, Bitbucket Server does a number of things:

- Bitbucket Server will return an error message to the client which the user will see on the command line: "Bitbucket is currently under heavy load and is not able to service your request. Please wait briefly and try your request again."

- A warning message will be logged for every time a request is rejected due to the resource limits, using the following format:
"A [scm-hosting] ticket could not be acquired (0/12)"
 - The ticket bucket is shown in brackets, and may be any of the available buckets (e.g. scm-command, scm-hosting, scm-refs or git-lfs)
- For five minutes after a request is rejected, Bitbucket Server will display a red banner in the UI *for all users* to warn that the server is under load.
 - This period is also configurable.

The hard, machine-level limits throttling is intended to prevent hitting are very OS- and hardware-dependent, so you may need to tune the configured limits for your instance of Bitbucket Server. When hyperthreading is enabled for the server CPU, for example, the default number of scm-hosting and scm-refs tickets may be too high, since the JVM will report double the number of physical CPU cores. In such cases, we recommend starting off with a less aggressive value; the value can be increased later if hosting operations begin to back up and system monitoring shows CPU, memory and I/O still have headroom.

Scaling Bitbucket Server for Continuous Integration performance

If you've got CI or other automatic tooling set up to poll Bitbucket Server for changes, you can end up with high load on your Bitbucket Server instance. Consider for instance a CI server that has a number of builds set up for a given repository. Each of those builds polls Bitbucket Server for changes and when it detects a change, it starts a new build. If your CI server supports parallel and/or chained build steps, each of these builds typically results in multiple clone operations of the same repository. The result: lots of polling for changes, and bursts of clones of a repository.

Caching

CI results in highly repetitive calls to Bitbucket Server: polling for changes typically results in the same response 90% of the time and a build triggers a number of identical clone calls to Bitbucket Server. Both operations can benefit greatly from caching when you experience repetitive load from CI.

On this page:

- [Limitations](#)
- [Considerations](#)
- [Configuration](#)
- [REST API](#)

Related pages:

- [Scaling Bitbucket Server](#)
- [Configuration properties](#)

Limitations

- Git 2.18 introduced version 2 of the Git wire protocol, which is fully supported by Bitbucket Server's caching. Please note, that due to differences in the protocols, Git clients using version 1 of the wire protocol cannot use caches generated for clients using version 2 of the wire protocol, and vice versa. As a result, when there's a mix of clients using version 1 and 2 of the wire protocol, disk usage for caching can be higher and cache efficiency can be lower, compared to when all clients use the same version of the wire protocol.

Considerations

Cache data is stored on disk for clone operations for a configurable period of time. Since large instances can potentially consume an entire disk the SCM Cache Plugin monitors remaining disk space and is automatically disabled when the configured minimum free disk space is reached.

See [Bitbucket Server config properties](#) for descriptions of the available system properties.

Configuration

The SCM Cache Plugin for Bitbucket Server can be configured using either the REST endpoint (some settings, not all) or the system properties in `<BITBUCKET_HOME>/bitbucket.properties`. Settings configured through REST are considered *before* the settings in `bitbucket.properties`.

REST API

| Method | Url | Description |
|--------|-----|-------------|
|--------|-----|-------------|

| | | |
|--------|--|---|
| GET | <code>/rest/scm-cache/latest
/config/protocols</code> | Retrieve the protocols for which caching has been enabled. |
| PUT | <code>/rest/scm-cache/latest
/config/protocols/
{protocol}</code> | Enable caching of hosting requests for the protocol (HTTP or SSH). |
| DELETE | <code>/rest/scm-cache/latest
/config/protocols/
{protocol}</code> | Disable caching of hosting requests for the protocol (HTTP or SSH). |
| GET | <code>/rest/scm-cache/latest
/config/upload-pack
/enabled</code> | Retrieve whether clone caching is enabled (true) or disabled (false). |
| PUT | <code>/rest/scm-cache/latest
/config/upload-pack
/enabled/{status}</code> | Enable (status = true) or disable (status = false) clone caching. |
| GET | <code>/rest/scm-cache/latest
/config/upload-pack/ttl</code> | Retrieve the expiry in seconds for the clone caches. |
| PUT | <code>/rest/scm-cache/latest
/config/upload-pack/ttl/
{expiryInSec}</code> | Set the expiry in seconds for the clone caches. |
| GET | <code>/rest/scm-cache/latest
/caches</code> | Retrieve information about the current caches: size, number of cache hits and misses, etc. |
| DELETE | <code>/rest/scm-cache/latest
/caches</code> | Clear all caches. |
| GET | <code>/rest/scm-cache/latest
/caches/{projectKey}/
{repoSlug}</code> | Retrieve information about the caches for the repository identified by projectKey and repoSlug: size, number of cache hits and misses, etc. |
| DELETE | <code>/rest/scm-cache/latest
/caches/{projectKey}/
{repoSlug}</code> | Clear the caches for the repository identified by projectKey and repoSlug. |

Bitbucket Server production server data

This page provides some data around the Bitbucket Server production instance that we run internally at Atlassian. We're providing this to give some idea of how Bitbucket Server performs in a production environment. Please realise that this information is entirely specific to this particular instance the details of your own installation may result in different performance data.

This data was collected with New Relic in February 2013, when the server was running a pre-release version of Bitbucket Server 2.2.

On this page:

- [Hardware](#)
- [Load](#)
- [Server load](#)
- [Git operations](#)

Hardware

The performance data below was gathered from the Atlassian Bitbucket Server production server running on:

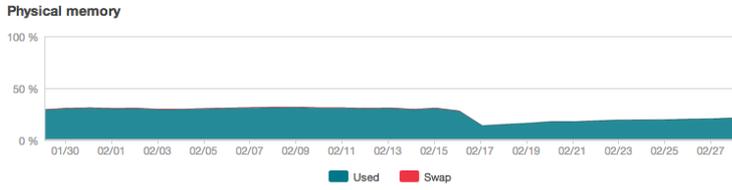
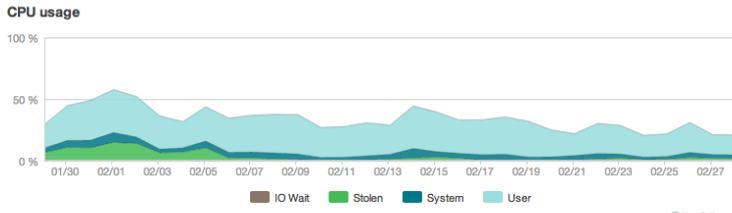
- Virtualized hardware
- 4 Hyper-threaded cores
- 12 GB RAM

Load

Load data summary for February 2013:

| Type | Load |
|--|---|
| CPU usage | less than 30% on average |
| Load average | less than 3 on average |
| Physical Memory | peaked at 31% |
| Processes | Git: 17.3% CPU
Java: 18.8% CPU |
| Clones | on average less than 300ms |
| Git operations/hour | peaking at 11,000 with an average of about 3,500 |
| Concurrent connections/hour | peaking at 100 connections with an average of about 40 concurrent connections |
| CI running against Bitbucket Server instance | 3 build servers with approximately 300 agents |

Server load



stash-prod.private.atlassian.com

8 CPUs
12 GB RAM
Intel QEMU

Scientific Linux release 6.2 (Carbon)
Linux 2.6.32-220.7.1.el6.x86_64
x86_64
New Relic agent 1.1.2.124

Apps

| | Response time | Throughput | Errors |
|---------------------------------------|---------------|------------|--------|
| j2ee_stash-prod.private.atlassian.com | 1,290 ms | 570 rpm | 1.46 % |

Processes >

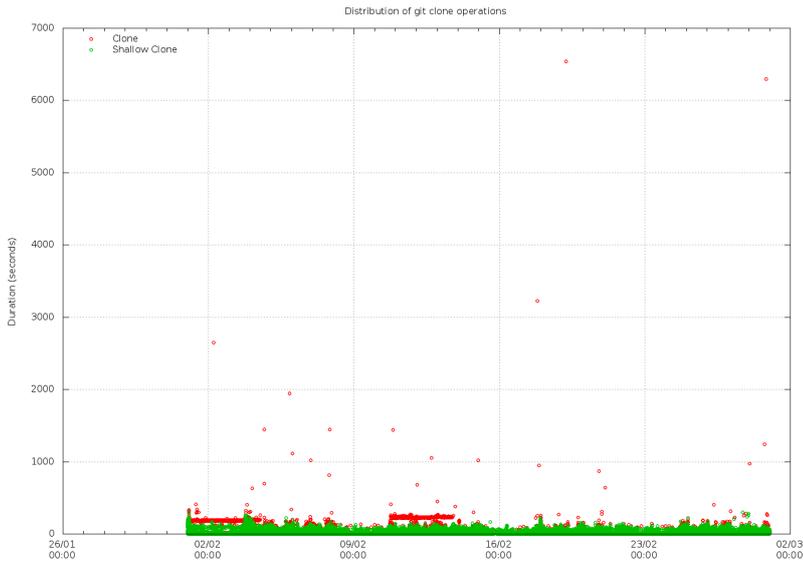
| User | Count | CPU | Memory |
|-------------|-------|-------|---------|
| git | 5 | 17.3% | 226 MB |
| java | 1 | 13.8% | 836 MB |
| bzip2 | 1 | 7.0% | 7.01 MB |
| portreserve | 2 | 6.2% | 393 MB |
| rsync | 1 | 6.2% | 90.8 MB |

65 more...

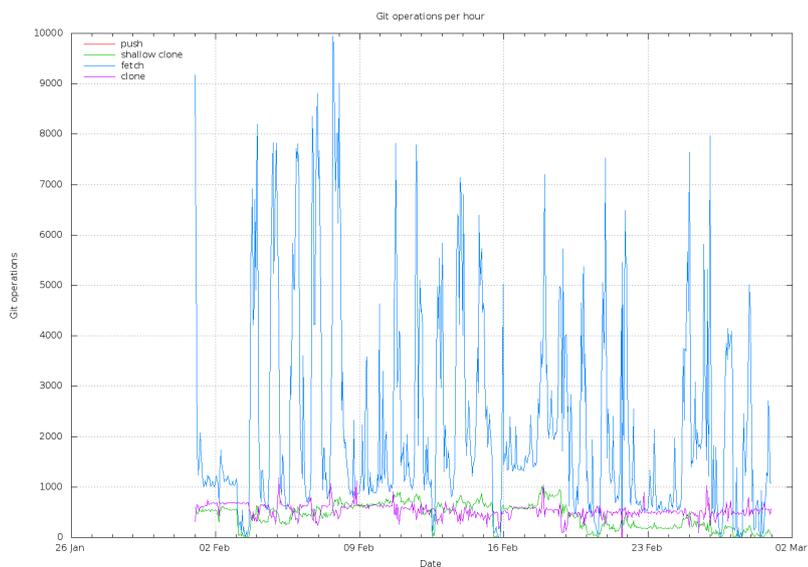
Recent stash-prod.private.atlassian.com events

Git operations

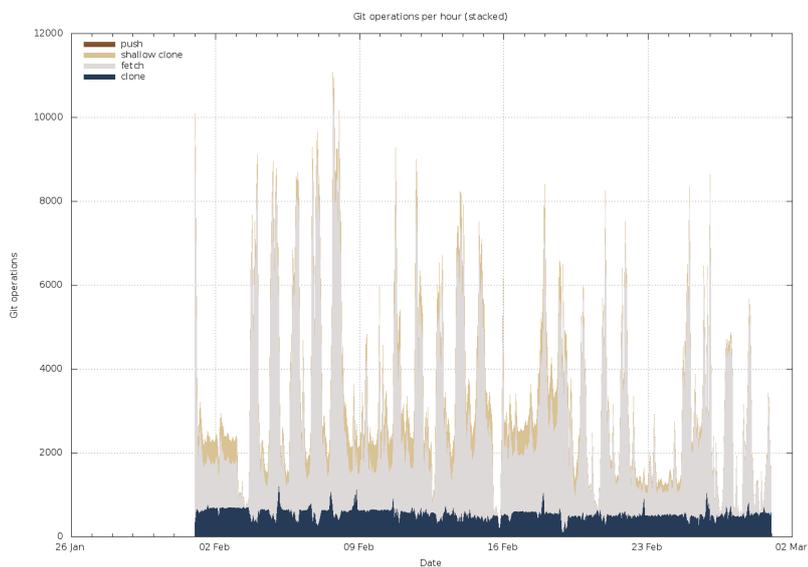
Git clone operations



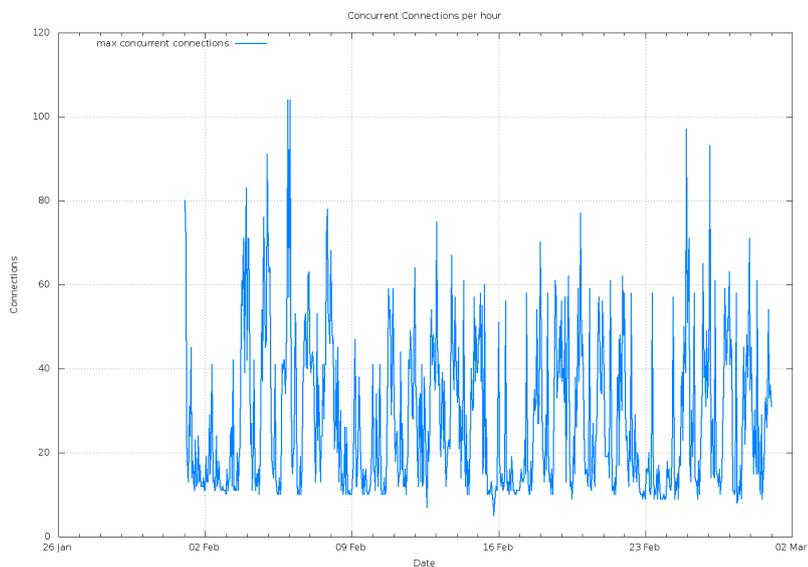
Git operations per hour



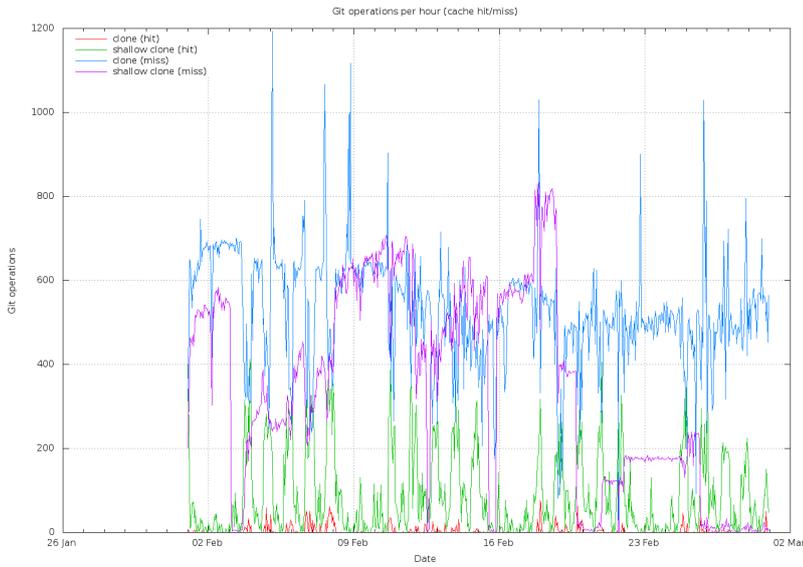
Git operations per hour (stacked)



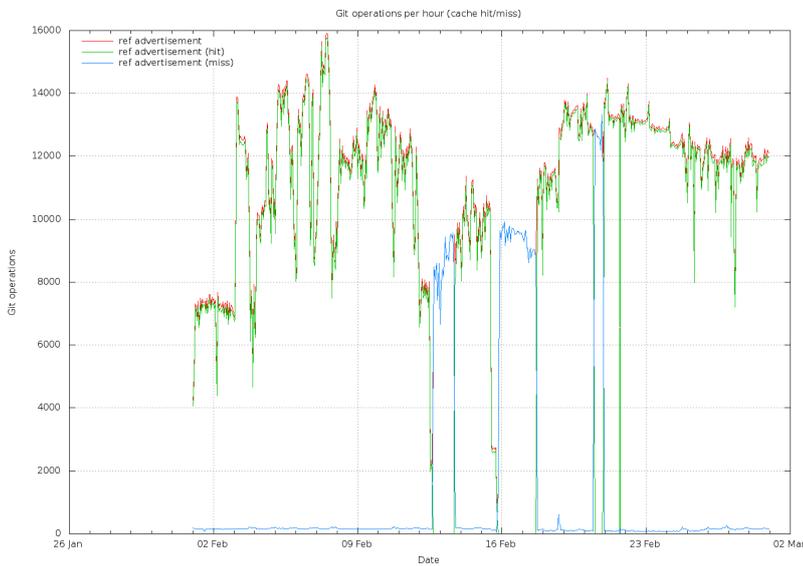
Concurrent connections per hour



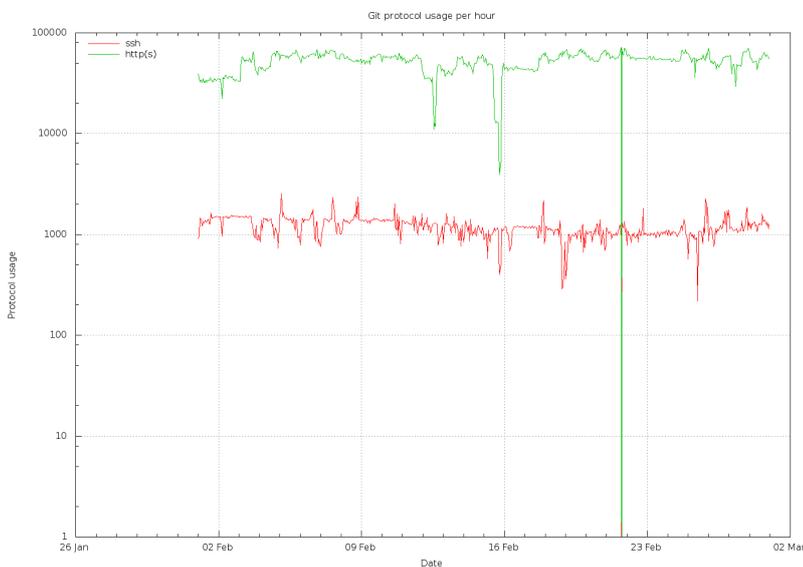
Git operations - cache hit/miss



Git operations - cache hit/miss



Git protocol usage per hour



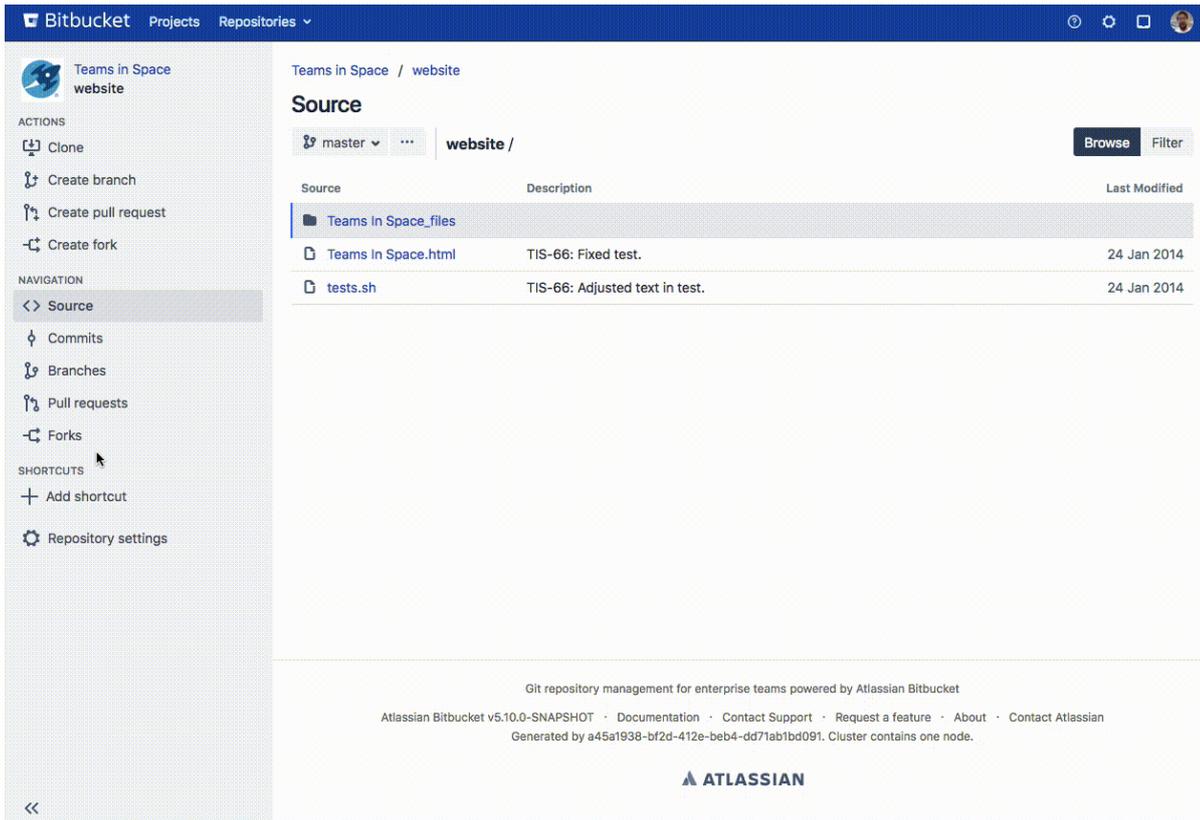
Add a shortcut link to a repository

Repository admins can add, edit, and delete shortcuts from the sidebar of a Bitbucket Data Center and Server repository. A shortcut is a link to a site outside of Bitbucket.

To add a new shortcut link

1. From within a repository, open the sidebar.
2. Select **Add shortcut**.
3. Add the **URL** and **Label**, then select **Add**.

To delete or edit an existing shortcut link, hover over the shortcut and select **Editor** or **Delete**.



Administer code search

Bitbucket Data Center and Server allows you to search through your code to find exactly what you're looking for right from the search bar. You can restrict your search results to a specific project or repository using search filters. You can also search for code in a particular language (e.g., lang:java) or with a particular file extension (e.g., ext:css). [Learn more about Bitbucket search syntax.](#)

This page explains how Bitbucket Server and Data Center uses Elasticsearch, how to configure Elasticsearch to fit the needs of your organization, and provides some tips for troubleshooting common problems.

About Elasticsearch

Code search is powered by Elasticsearch, which indexes the content of Bitbucket Server repositories in real time. Elasticsearch comes bundled with Bitbucket Server and is managed by Bitbucket Server automatically. Under most circumstances it should not need to be managed manually by an administrator.

Important details and requirements about Bitbucket and Elasticsearch

This section outlines important considerations for using Bitbucket and Elasticsearch. Read the section appropriate for your installation.

For *all* Bitbucket Server/Data Center installations

- Elasticsearch is required to run Bitbucket Server and Data Center (although [code search can be disabled entirely](#) if needed).
- For information on supported versions of Elasticsearch and Bitbucket Server, please refer to [Supported Platforms](#).
- You can configure what is indexed for code search to [exclude various types of forked repositories to save disk space](#).

 Data Center will continue support for both Elasticsearch 2.3 and 5.5, **only until the release of Bitbucket 6.0** to allow an upgrade to be planned. At this time, Elasticsearch 2.3 will no longer be supported for Data Center.

For *Bitbucket Server installed using the installer*

- If you install Bitbucket Server *using the installer* Elasticsearch is configured to work out of the box.
- Bitbucket Server's *bundled Elasticsearch instance* requires ports 7992 and 7993 to be available. This is not configurable when using the bundled Elasticsearch instance. If these ports are not available, you will need to [How to Install and configure a remote Elasticsearch instance](#) to work with Bitbucket.

For *Bitbucket Server installed using an archive file (.zip/.tar.gz)*

- After unzipping the archive files, you need to copy the contents of the `<Bitbucket installation directory>/elasticsearch/config-template` directory to the `<Bitbucket home>/shared/search` directory, or Bitbucket won't start correctly. Details can be found in the archive installation instructions for [Windows](#) and [Linux](#).

For *Bitbucket Data Center installations*

On this page:

- [About Elasticsearch](#)
- [Change the username and password for accessing Elasticsearch](#)
- [Use a remote Elasticsearch instance with Bitbucket Data Center](#)
- [Start Bitbucket Data Center with a remote Elasticsearch instance](#)
- [Troubleshooting and frequently asked questions about Elasticsearch](#)

Related pages:

- [How to Install and configure a remote Elasticsearch instance](#)
- [Bitbucket Server config properties - Elasticsearch](#)
- [Bitbucket Server config properties - Search](#)

- Bitbucket Data Center requires a remote Elasticsearch instance, as it is not bundled or installed for Bitbucket Data Center. You must install and configure a remote Elasticsearch instance, as described in the [installation instructions for Bitbucket Data Center](#).
- Bitbucket Data Center can have *only one* remote connection to Elasticsearch for your cluster. This may be a standalone Elasticsearch installation or a clustered installation behind a load balancer.
- Code search is not critical for high availability, however it is possible run a cluster of Elasticsearch nodes to achieve high availability, although Atlassian cannot guarantee support for such a setup in event you encounter problems.

Change the username and password for accessing Elasticsearch

There are two ways you can change the user credentials Bitbucket uses to access Elasticsearch.

1. Configure the details of your Elasticsearch instance within the Bitbucket Server UI, or
2. Configure the details of your Elasticsearch instance within the `bitbucket.properties` file.

If an Elasticsearch parameter is set from within the `bitbucket.properties` file, it cannot be edited later from the admin UI. Any changes that need to be made to the Elasticsearch configuration must be made within the `bitbucket.properties` file.

Go to  > **Server settings** > **Search**.

Here you can see the current configuration of Elasticsearch, including the port and user credentials for accessing Elasticsearch.

Search

| | |
|------------------------|--|
| Elasticsearch URL * | <input type="text" value="http://localhost:7992/"/> |
| | Base URL of the Elasticsearch instance (defaults to http://localhost:7992) |
| Elasticsearch username | <input type="text" value="bitbucket"/> |
| | Username for connecting to the Elasticsearch instance |
| Elasticsearch password | <input type="password" value="*****"/> |
| | Password for connecting to the Elasticsearch instance |

1. Locate the `bitbucket.properties` file in the `<Bitbucket home directory>/shared` directory.
2. Add the details of your Elasticsearch instance.

```
<Bitbucket home directory>/shared/bitbucket.properties

plugin.search.elasticsearch.baseurl=http://localhost:7992/
plugin.search.elasticsearch.username=<username>
plugin.search.elasticsearch.password=<password>
```

*`<username>` and `<password>` are values you provide.

 If a parameter is set in the `bitbucket.properties` file, it cannot be edited later from the admin UI. Any changes that need to be made to the Elasticsearch configuration must be made within the `bitbucket.properties` file.

Use a remote Elasticsearch instance with Bitbucket Data Center

Bitbucket Data Center requires a remote Elasticsearch instance, as it is not bundled or installed for Bitbucket Data Center. This may be a standalone Elasticsearch installation or a clustered installation behind a load balancer.

For instructions on setting up a single remote instance of Elasticsearch to use with Bitbucket Data Center, see the [Bitbucket Data Center installation instructions](#).

For guidance on using a clustered Elasticsearch installation with Bitbucket Data Center, see the page [How to Install and configure a remote Elasticsearch instance](#), but note that Atlassian cannot guarantee support for such a setup in event you encounter problems.

Start Bitbucket Data Center with a remote Elasticsearch instance

Because Bitbucket Data Center requires a remote Elasticsearch instance, you need to start Bitbucket Data Center with an argument that won't try to start the bundled Elasticsearch that comes with Bitbucket Server.

To start Bitbucket when using a remote Elasticsearch instance

| | |
|--------------------|---|
| For Linux | <code>start-bitbucket.sh --no-search</code> |
| For Windows | <code>start-bitbucket.bat /no-search</code> |

Troubleshooting and frequently asked questions about Elasticsearch

I can't configure Elasticsearch settings from the UI because the fields are grayed out

Your Elasticsearch configurations were set within the `bitbucket.properties` file. If a parameter is set in the `bitbucket.properties` file, it cannot be edited later from the admin UI. Any changes that need to be made to the Elasticsearch configuration must be made within the `bitbucket.properties` file.

My remote Elasticsearch instance is using up a lot of disk space

With Bitbucket 5.1+ you can configure the Elasticsearch search indexes to exclude certain types of forked repositories, which can save disk space when your company uses a fork-based workflow for development. See the page [Bitbucket Server config properties - Search](#) for the options available to change what is indexed for code search. Note that excluding content from being indexed can prevent such content from ever appearing in code search results, but will not prevent the repositories themselves from appearing in search results.

I need to disable code search entirely

If you need to turn off code search for all of your Bitbucket Server (or Data Center) users, you can disable code search entirely from within the `bitbucket.properties` file. See the page [Bitbucket Server config properties - Search](#) for details.

Code search is unavailable

There are a few ways Elasticsearch may not have been configured properly. To troubleshoot the bundled version of Elasticsearch that comes with Bitbucket Server, the page [Troubleshooting steps for Bitbucket Server Code Search](#) is a good place to start. If you're still stuck after looking at the tips on that page, you

could post a question in the [Atlassian Community](#) (in case your configuration is not typical or unsupported), or [contact Atlassian Support](#) for assistance (you'll need to log in to raise a support request).

Is Elasticsearch/code search required to run Bitbucket Server/Data Center?

Yes. Elasticsearch is required to run Bitbucket Server and Data Center (although code search can be disabled entirely if needed).

Can I use the bundled instance of Elasticsearch with Bitbucket Data Center?

No. Bitbucket Data Center requires a remote Elasticsearch instance, as it is not bundled or installed for Bitbucket Data Center.

Can I use a clustered installation of Elasticsearch with Bitbucket Data Center?

Yes, however, Bitbucket Data Center can have *only one* remote connection to Elasticsearch for your cluster. This may be a standalone Elasticsearch installation or a clustered installation behind a load balancer. For guidance on using a clustered Elasticsearch installation with Bitbucket Data Center, see the page [How to Install and configure a remote Elasticsearch instance](#), but note that Atlassian cannot guarantee support for such a setup in event you encounter problems.

How do I secure my remote Elasticsearch instance?

Atlassian strongly recommends you secure access to your remote Elasticsearch instance with a username and password, and a minimum of basic HTTP authentication. You can secure your remote Elasticsearch instance with a security plugin that requires anyone connecting to it provides authentication credentials. Atlassian provides a free plugin called Buckler for this purpose.

For instructions on how to secure your remote Elasticsearch instance with Buckler, see [Install and configure a remote Elasticsearch instance - Secure your remote Elasticsearch instance](#)

How to Install and configure a remote Elasticsearch instance

This page describes how to provision a remote Elasticsearch instance to work with *Bitbucket Data Center*.

Bitbucket Data Center requires a remote Elasticsearch instance, as it is not bundled or installed for Bitbucket Data Center.

Bitbucket Data Center can have *only one* remote connection to Elasticsearch for your cluster. This may be a standalone Elasticsearch installation or a clustered installation behind a load balancer.

For details of about how Bitbucket uses Elasticsearch, including troubleshooting tips and frequently asked questions, see [Administer code search](#).

On this page

- [Step 1: Install Elasticsearch on a remote machine](#)
- [Step 2: Configure Elasticsearch](#)
- [Step 3: Secure Elasticsearch](#)
- [Step 4: Connect Elasticsearch to Bitbucket](#)

Related pages

- [Administer code search](#)
- [Bitbucket Server config properties - Elasticsearch](#)
- [Bitbucket Server config properties - Search](#)

Step 1: Install Elasticsearch on a remote machine

We don't provide specific instructions for installing Elasticsearch, but a good place to start is the [Elasticsearch \(6.8.6\) installation instructions](#). Elastic provides installation packages in several different formats [here](#). Note that the authentication plugin Buckler, described within the [Secure Elasticsearch](#) section only supports versions 6.8.6 of Elasticsearch. Refer to the [Supported platforms - Additional Tools](#) section to see the current Elasticsearch release we support.

Step 2: Configure Elasticsearch

The `elasticsearch.yml` file contains configuration details for your Elasticsearch instance.

To configure your remote Elasticsearch instance

1. Locate the `elasticsearch.yml` file within the [configuration directory of your Elasticsearch \(6.8.6\) instance](#).
2. Add these parameters to your `elasticsearch.yml` file

```
action.auto_create_index: ".watches,.triggered_watches,.watcher-history-*"
network.host: 0.0.0.0
xpack.security.enabled: false
```



Third party plugins, such as Elastic's Shield plugin, may require [specific exceptions to be set for `action.auto_create_index`](#). Consult your provider's documentation for more information.

Step 3: Secure Elasticsearch

You need to secure access to your remote Elasticsearch instance with a username and password. We recommend securing your remote Elasticsearch instance with a security plugin that requires anyone connecting to it provides authentication credentials. Atlassian provides a free plugin called Buckler for this purpose. Bitbucket Server also supports authentication to Elasticsearch through other plugins that provide basic authentication, like Elastic's [Shield plugin](#).

To secure your remote Elasticsearch instance with the Buckler plugin

Copy the Buckler plugin URL to your clipboard:

| Elasticsearch version | Buckler plugin |
|-----------------------|-------------------------------|
| Elasticsearch 7.9.3 | Buckler 2.14 |
| Elasticsearch 7.5.2 | Buckler 2.1.3 |
| Elasticsearch 6.8.6 | Buckler 2.1.2 |
| Elasticsearch 6.6.1 | Buckler 2.1.1 |
| Elasticsearch 6.5.3 | Buckler 2.0.1 |

Install the plugin onto your remote Elasticsearch instance using the plugin helper in the `Elasticsearch/bin` directory:

```
./elasticsearch-plugin install -b <link from table above>
```

Configure basic authentication for Bitbucket to access your remote Elasticsearch installation. We strongly suggest enabling basic HTTP authentication, at minimum, for a remote Elasticsearch instance working with Bitbucket Data Center.

1. Create a directory called `buckler` within the `elasticsearch/config` directory.
2. Within the `elasticsearch/config/buckler` directory, create a file named `buckler.yml`.



When installing using rpm or deb file

The location of your configuration directory varies depending on how you installed Elasticsearch. For rpm/deb installations the location is typically in `/etc/elasticsearch`

if the file is placed in the wrong location Buckler does not enforce any permissions, so your instance/cluster is not secure.

3. Enable Buckler: at this point there are no configuration properties within `buckler.yml`, so the features of the Buckler plugin are disabled.

To enable the Buckler for basic HTTP authentication, you add these properties to the file, creating a username and password that Bitbucket will use to access Elasticsearch (configured in a later step).

elasticsearch/config/buckler/buckler.yml

```
auth.basic.http.enabled: true
auth.basic.username: <username>
auth.basic.password: <password>
```

```
auth.basic.http.enabled: true
auth.basic.tcp.enabled: true
auth.basic.username: admin
auth.basic.password: basicpassword
tls.http.enabled: true
tls.tcp.enabled: true
tls.keystore.path: /path/to/keystore
tls.keystore.password: keystorepassword
```

| Parameter | Value | Description |
|---------------------------------------|---------------------------------------|--|
| <code>auth.basic.http.enabled:</code> | <code>true</code> | Enables basic authentication for HTTP. |
| <code>auth.basic.tcp.enabled:</code> | <code>true</code> | Enables basic authentication for TCP. |
| <code>auth.basic.username:</code> | <code><username></code> | Username to access Elasticsearch instance. |
| <code>auth.basic.password:</code> | <code><password></code> | Password to access Elasticsearch instance. |
| <code>tls.http.enabled:</code> | <code>true</code> | Enables TLS for HTTP. |
| <code>tls.tcp.enabled:</code> | <code>true</code> | Enables TLS for TCP. |
| <code>tls.keystore.path:</code> | <code><path/to/keystore></code> | Absolute filesystem path to the keystore. |
| <code>tls.keystore.password:</code> | <code><keystorepassword></code> | Password for accessing the keystore. |

If you enable TLS, the following block will need to be added to `elasticsearch.yml`

```
http.type: buckler
transport.type: buckler
```

This is relevant for Buckler version 1.0.1 onwards.

4. Start your remote Elasticsearch instance. See [Elasticsearch \(6.8.6\) documentation](#) for specific instructions.

Step 4: Connect Elasticsearch to Bitbucket

Once you've configured your Elasticsearch instance you then need to connect it to Bitbucket.

To configure your remote Elasticsearch instance using the `bitbucket.properties` file

i Once a parameter is set in the `bitbucket.properties` file, *it cannot be edited later from the admin UI*. Any changes that need to be made to the Elasticsearch configuration must be made within the `bitbucket.properties` file.

1. Locate the `bitbucket.properties` file in the `<Bitbucket home directory>/shared/directory`.
2. Add the details of your Elasticsearch instance (created in [step 3.3](#) above):

```
<Bitbucket home directory>/shared/bitbucket.properties

plugin.search.elasticsearch.baseurl=http://localhost:9200/
plugin.search.elasticsearch.username=<username>
plugin.search.elasticsearch.password=<password>
```

3. Save and close the file.
4. Start Bitbucket Server without starting the bundled Elasticsearch instance.

| | |
|--------------------|---|
| For Linux | <code>start-bitbucket.sh --no-search</code> |
| For Windows | <code>start-bitbucket.bat /no-search</code> |

Your remote Elasticsearch instance is now configured to work with Bitbucket Server.

Configure Bitbucket's code search index

This page describes how to configure what is indexed within the Bitbucket Server (or Data Center) code search index. This could be beneficial for your instance if you need to reduce the amount of disk space used by Elasticsearch, which provides the functionality for Bitbucket's code search.

There are various options you can use to optimize disk usage. Which configuration you choose depends on your needs, however organizations with a fork-based workflow are the most likely to see the benefits of changing code search index properties.

To change what is indexed for Bitbucket's code search

1. Locate the `bitbucket.properties` file in the `<Bitbucket home directory>/shareddirectory`.
2. In the `bitbucket.properties` file, set a value for the property `plugin.search.codesearch.indexing.exclude=`

These are the options for configuring what code is indexed. You can only set one value at a time.

| Value | Description |
|-------------------------------|--|
| <code>all-forks</code> | Code in forked repositories will not appear in code search results. |
| <code>personal-forks</code> | Code in forked repositories of personal projects will not appear in code search results. |
| <code>synced-forks</code> | Code in forked repositories with syncing enabled will not appear in code search results. |
| <code>undiverged-forks</code> | Same as <code>synced-forks</code> , except if the default branch is ever changed, the entire forked repository will be indexed |

3. Restart Bitbucket.
4. If you've changed to a more restrictive setting (eg no exclusions, to `forks` exclusion) then you'll need to kick off a manual reindex. We will not automatically remove your data from the index, and only by running the search sync job will data be removed. Elasticsearch should be running when you execute this command. This will iterate through your repositories and remove all those from Elasticsearch where they would otherwise not be indexed with your new setting. In our example case this would mean all repositories that were forks would have their repository content removed from the Elasticsearch instance

```
curl -XPOST -H 'Content-Type: application/json' -H 'Accept: application/json' -u <adminUsername> http://<BitbucketURL>/rest/indexing/latest/sync
```

Disable code search

You can also choose to disable code search entirely by changing the value of `plugin.search.codesearch.indexing.enabled=` to `false`. When code search is disabled, no repository content is indexed from the time at which the property is set. This does not disable Bitbucket's search box, or delete any existing indexes. Repository and project metadata will also still be indexed, allowing you to use the dashboard and search box's repository quick search.

To remove the data from the index after disabling code search you must run the search sync job. Bitbucket will not remove data from the index until you specify you're ready via the following command.

```
curl -XPOST -H 'Content-Type: application/json' -H 'Accept: application/json' -u <adminUsername> http://<BitbucketURL>/rest/indexing/latest/sync
```



You still must have Elasticsearch to use Bitbucket Server

Use an Elasticsearch cluster with Bitbucket Data Center

Bitbucket Data Center requires a connection to a remote Elasticsearch installation to enable code search. Although code search is not critical for high availability, it is possible to run a cluster of Elasticsearch nodes to achieve high availability for the Bitbucket's code search index. This page provides guidance on deploying a cluster of Elasticsearch nodes that run behind a load balancer connected to Bitbucket application nodes.

Bitbucket Data Center can have *only one* remote connection to Elasticsearch for your cluster. This may be a standalone Elasticsearch installation or a clustered installation behind a load balancer.



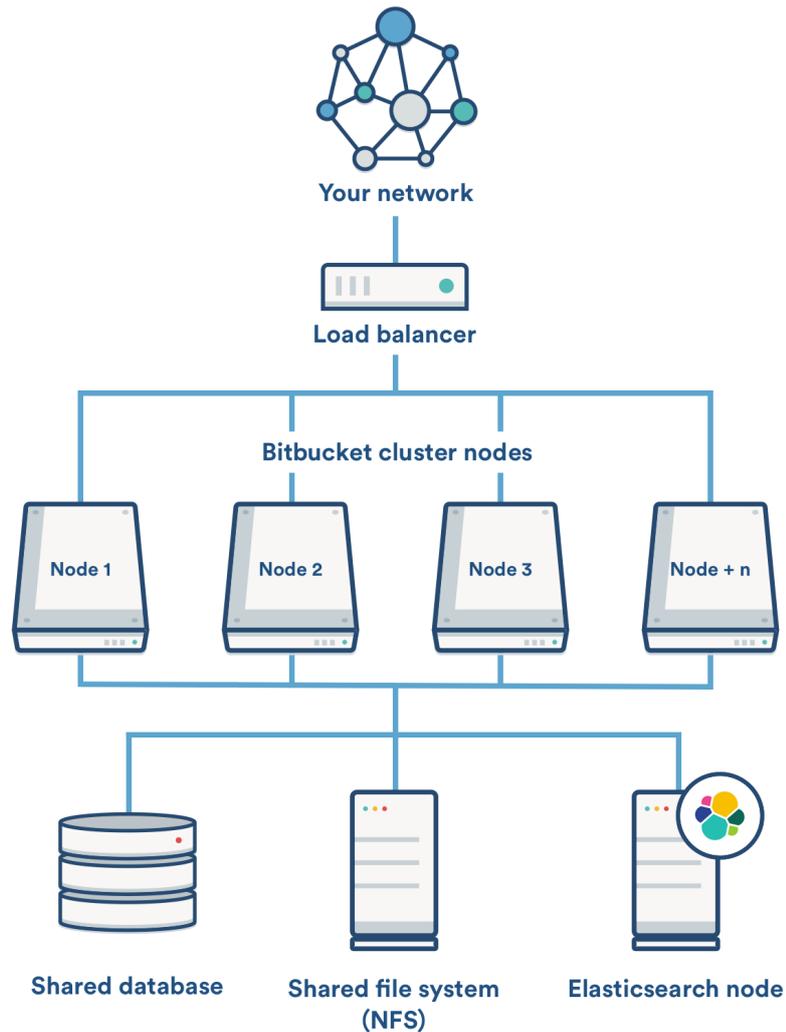
Bitbucket Data Center allows the use of clustered Elasticsearch installation, however Atlassian Support does not provide assistance for configuring them. Consequently, Atlassian **cannot guarantee providing any support for them**. If assistance with configuration is required, raise a question in the [Atlassian Community](#).

Standard Bitbucket Data Center component diagram

Before thinking about deploying an Elasticsearch cluster to use with Bitbucket Data Center, it helps to understand what a standard Bitbucket Data Center installation looks like. The page [Bitbucket Data Center requirements](#) outlines the detailed requirements.

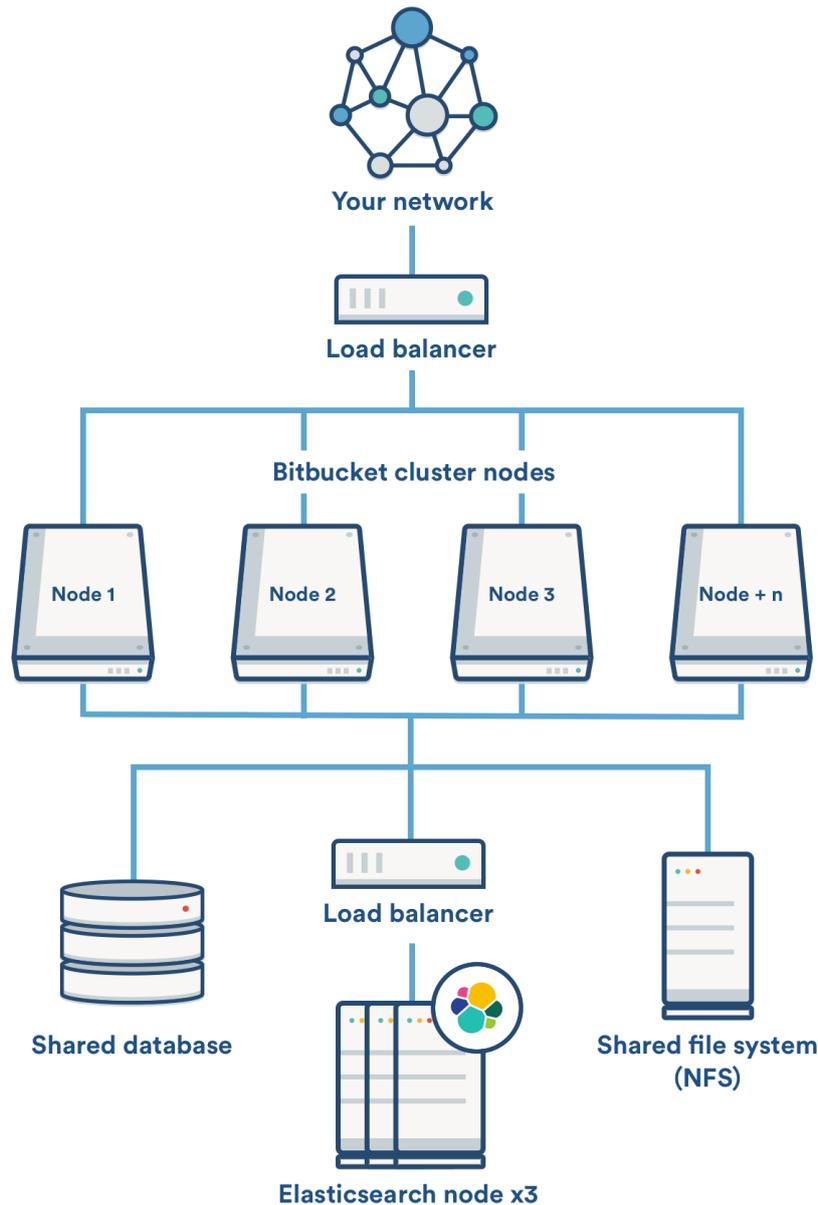
At minimum, a standard Bitbucket Data Center installation must have these components, each on a dedicated machine, and connected by a high speed LAN (click the links to view detailed requirements for each component):

- [Bitbucket applications nodes](#), all running the same version of Bitbucket Data Center.
- [Load balancer](#) that supports session affinity (or, "sticky sessions").
- [Shared database](#), able to take block-level snapshots.
- [Shared filesystem](#), accessible via NFS as a single mount point.
- [A remote Elasticsearch node](#), with a single connection to the Bitbucket application nodes.



Bitbucket Data Center component diagram with an Elasticsearch cluster

This component diagram provides an example configuration of a Bitbucket Data Center installation deployed with a cluster of Elasticsearch nodes behind a load balancer. In this example, we assume you'll run Elasticsearch in its own cluster.



Using Amazon's Elasticsearch service with Bitbucket Data Center

The easiest way to set up and deploy an Elasticsearch cluster for Bitbucket Data Center is to use the Amazon's Elasticsearch service. Atlassian cannot provide direct support for setting up your AWS Elasticsearch cluster, but a good place to start is with Amazon's documentation: [Amazon Elasticsearch Service \(documentation\)](#).

i Be sure to use the correct version

Data Center will continue support for both Elasticsearch 2.3 and 5.5, **only until the release of Bitbucket 6.0** to allow an upgrade to be planned. At this time, Elasticsearch 2.3 will no longer be supported for Data Center.

Setting up a standalone Elasticsearch cluster with Bitbucket Data Center

If you're not using AWS's Elasticsearch service, and instead are setting up a cluster of Elasticsearch nodes yourself, you generally want to set up a cluster of *at least two* Elasticsearch nodes. Two nodes allows each node to have a replica on it, meaning that if one node goes down for a short period, you'll still have all of your search results available to you while you repair/replace the other node. If you require more fault tolerance, increase the number of nodes in your cluster.

- Elastic's official documentation for setting up an Elasticsearch cluster: [Elasticsearch - Installation \(5.5\)](#).

Securing your Elasticsearch cluster

After setting up your Elasticsearch cluster, it's important you secure your cluster. You can use a variety of techniques to secure Elasticsearch in a cluster. To get started with basic authentication you'll need to use a plugin with Elasticsearch that will block attempts to access the cluster without a valid username and password. Atlassian created a small plugin named Buckler to do just that.

Secure Elasticsearch with Atlassian's Buckler plugin

For instructions on how to configure Buckler, see the page [Install and configure a remote Elasticsearch instance - Step 3: Secure Elasticsearch](#). Buckler needs to be installed on every node in the cluster. In any case, if you're going to use Amazon's Elasticsearch Service, you must set the AWS region in the `bitbucket.properties` file to enable request signing.

```
<Bitbucket home directory>/shared/bitbucket.properties
```

```
plugin.search.elasticsearch.aws.region=
```

Secure Elasticsearch with Elastic's Shield plugin

Bitbucket also supports authentication to Elasticsearch through other plugins that provide basic authentication, like Elastic's [Shield plugin](#). This plugin isn't directly supported by Atlassian, but Bitbucket can still interoperate with it via utilising its basic authentication system.

Secure Elasticsearch with Amazon's request signing

Basic authentication and AWS request signing are supported by Bitbucket (AWS request signing Bitbucket 4.10+). AWS Request signing allows you to use the AWS Elasticsearch service to serve as your Elasticsearch cluster. This will allow you to secure your Elasticsearch cluster to only allow requests from the AIM user that the node Bitbucket is running on inside of AWS ec2 has.

Adding additional storage for your repository data

 This feature is available with an active [Bitbucket Data Center](#) license.

As your repository data grows it may become too large to hold in a single storage location. When this happens, you'll need to add additional storage space. There are two ways to do this:

1. Increase the storage space allocated to your `sharedhomedirectory`
2. Add additional data stores

Increasing the storage space allocated to your `sharedhomedirectory` is the most straightforward option, and we recommend it for most customers. If this is not an option for you (for example, because of a company policy limiting the size of disks or partitions), you should add a data store.

Before adding a data store, read through the considerations below.

Data store considerations

The ability to add data stores is aimed at organizations with limitations on disk partition size. It's not intended as a way to increase disk performance. In addition:

- Data stores **cannot** be moved or removed, once added.
- Any nodes that don't have access to the data store won't join the cluster until it's mounted and the node is restarted.
- Repository data cannot be moved between data stores. Once a repository has been created in the shared home directory or on a data store, all of its data will stay there, including the data in its forks, any pull request attachments, and [Git LFS data](#).

Adding a data store using Storage Management

You don't need to stop `Bitbucket` to add a data store.

To add a new data store:

1. [Create a new shared filesystem](#).
2. Mount the shared filesystem on all nodes making sure that:
 - the filesystem is writeable by the user running the `Bitbucket` process.
 - the data store is not a subdirectory or parent directory of the shared home or another data store.
 - the data store is mounted on all nodes in the same location using NFS.
3. Go to  **Storage > Add a data store**.

On this page:

- [Data store considerations](#)
- [Adding a data store using Storage Management](#)
- [Shared home use after adding a data store](#)

4. Enter the data store directory path, and select **Check path**.

Add a data store

Data stores cannot be removed once created. The path must be a directory and cannot be in the shared home or in another data store.

Path

Add the data store directory path. [Learn more](#)

5. Select **Add data store**.

- If you get an error use the IP address included below the message to find the affected node and fix the problem.

Shared home use after adding a data store

After one or more data stores have been added to Bitbucket, the system will not use the shared home directory to store new repositories, except in a few situations.

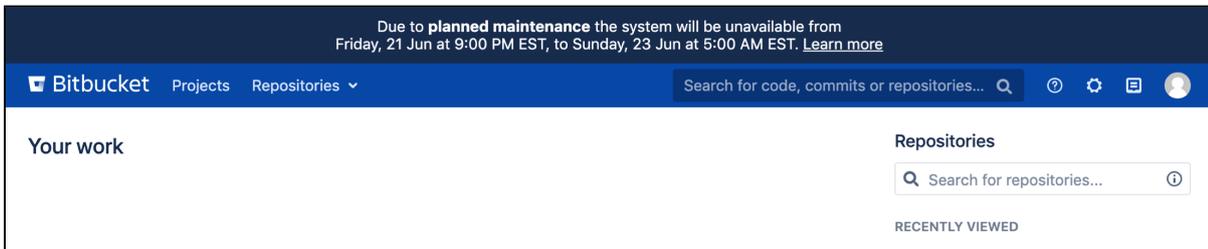
For example, new forks of repositories stored in the shared home directory will still be created in this directory. This is because Bitbucket stores forks in the same location as their origin repository to save disk space. New attachments and new LFS objects for repository hierarchies stored on the shared home will also continue to be added to the shared home. Lastly, the shared home will continue to hold data that is not related to a single repository, such as user avatars.

Add a system-wide announcement banner

An announcement banner is a great way to communicate important information like scheduled downtime or upcoming maintenance periods. When turned on, it appears in Bitbucket Data Center and Server at the top of every page until you turn it off, helping to make sure you get your message out.

On this page:

- [Turning on the announcement banner](#)
- [Turning off the announcement banner](#)

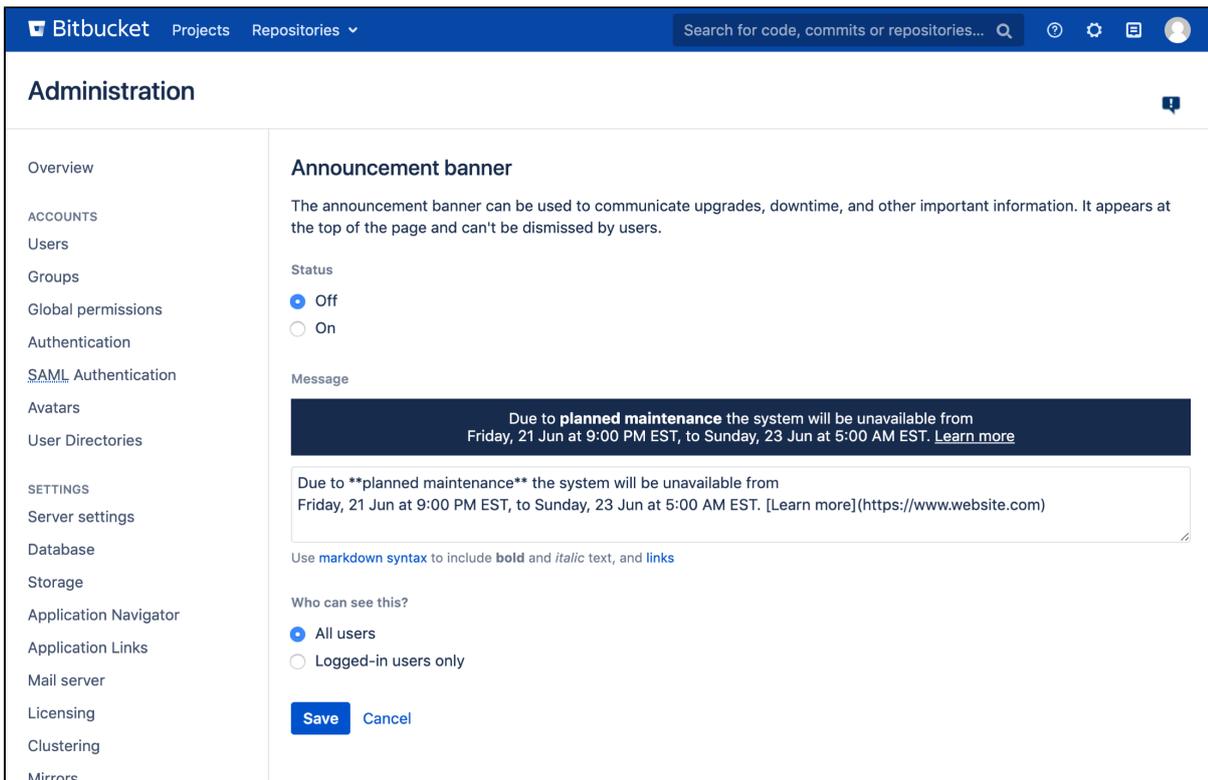


When setting up your banner, you can:

- use [markdown syntax](#) to add **bold** and *italic* text, and to add a link to where users can find more information
- control whether all users or only logged-in users can see it

Turning on the announcement banner

You can set up and turn on the announcement banner through the Bitbucket settings page.



To turn on the announcement banner:

1. Go to  > **Announcement Banner**
2. Change the status to **On**
3. Write your message
4. Select who should see it
5. Select **Save**

Your announcement banner will appear immediately.

 When writing your message, use the banner preview to make sure it fits inside the banner. If it doesn't, you won't be able to save it.

Turning off the announcement banner

Once you turn on your announcement banner it will be shown to users until you turn it off.

To turn off the announcement banner:

1. Go to  > **Announcement Banner**
2. Change the status to **Off**
3. Select **Save**

Configuring Project links across Applications

After connecting your Atlassian applications using [Application Links](#), you can also connect the areas of those applications that contain information relating to your project or team using Project Links.

Below are some selection priority advantages you'll get from Project Links that connect Jira and Bitbucket Data Center and Server:

- When you create a feature branch from Jira, repositories from a linked Jira project will be prioritized in the list of repositories.
- When you create a Jira issue from a comment in Bitbucket, the linked Jira project is pre-selected from the list of projects.
- When you check the details of a Jira issue in Bitbucket, issues from linked Jira projects are shown higher in the issue view dialog. This is helpful when there are multiple issues with the same issue-key from different Jira instances.
- When you check development details in Jira, commits and branches are shown higher in the development details dialog.

On this page:

- [Creating an Application Link](#)
- [Creating a Project Link](#)
- [Making a Project Link the Primary link](#)
- [Deleting a Project Link](#)

Related pages:

- [Creating projects](#)

Creating an Application Link

You won't be able to create a Project Link until you link Bitbucket to another application using Application Links.

1. Go to  > **Application Links**.
2. Enter the URL of the application you want to link to and then click **Create new link**.
3. Finish configuring the link.

For more information on creating and using Application Links, see [Link Atlassian applications to work together](#).

Creating a Project Link

There are two types of Project Links:

- 2-way Project Links are reciprocal links that provide authorized access for Project Links to be created or deleted from both linked applications.
- 1-way Project Links do not share authorized access to delete or create Project Links between linked applications.

To link a Bitbucket project to a project or space in another application when there are Application Links (requires [project admin permissions](#)):

1. Go to **Project settings > Project links**.
2. Click **Create Project Link**.
3. The instructions for creating a Project Link will vary depending on whether Bitbucket has authorization:
 - If Bitbucket has authorization to create the Project Link:
 1. Choose the application that contains the project you want to link to.
 2. Choose a project to link to. If there are no projects listed, then all possible projects have already been linked.

3. Select the **Make this a 2-way Project Link** checkbox if you want to create a reciprocal link from the target application to Bitbucket.

4. Click **Create link**.

- If Bitbucket doesn't have authorization to create the Project Link:
 1. Choose the application that contains the project you want to link to.
 2. Choose one of the following options:
 - **Dont authorize Bitbucket Server** - Click this option if you only need to create links to public projects.
 - **Authorize Bitbucket Server** - Click this option if you want to allow Bitbucket to create links to all projects on this application.

Making a Project Link the Primary link

Making a project link the primary link gives it priority over other links. If you've set up Project Links to more than one project in the same application, for example you have linked your Bitbucket project to two Jira projects, then one of the Project Links will be marked as *Primary*.

To make a Project Link the Primary link (requires [project admin permissions](#)):

1. Go to **Project settings > Project links**.
2. Next to the Project Link that you want to make *Primary*, select ***** > Make primary link**.

Deleting a Project Link

Deleting a project link stops the two projects from sharing information.

To delete a project link (requires [project admin permissions](#)):

1. Go to **Project settings > Project links**.
2. Next to the Project Link that you want to delete, select ***** > Delete**. If authorization is required, choose one of the following:
 - **Authorize** - Granting Bitbucket authorization to access to the linked application allows you to delete the 2-way Project Links between those applications.
 - **Delete** - Deleting without granting Bitbucket authorization to access to the linked application will restrict you to being able to delete only the 1-way link from Bitbucket to the linked applications. Links back to Bitbucket will need to be deleted in the linked applications Administration settings.
3. (Optional) Select the option to delete the reciprocal link from the linked application to Bitbucket.
4. Click **Delete** to delete the project link.

Improving instance stability with rate limiting



A Data Center license is required to use this feature. [Learn more](#)

When automated integrations or scripts send requests to Bitbucket in huge bursts, it can affect its stability, leading to drops in performance or even downtime. Rate limiting allows your instance to self-protect. It keeps it stable by giving you control over how many requests automations can make, and how often they can make them.

On this page:

- [How rate limiting works](#)
- [How to turn on rate limiting](#)
- [How rate limiting works in a cluster](#)
- [Identifying users who have been rate limited](#)
- [Adding and editing rate limiting exemptions](#)
- [How to help users avoid being rate limited](#)

Related pages:

- [Enabling JMX counters for performance monitoring](#)

How rate limiting works

Rate limiting targets HTTP requests with basic or bearer authentication (such as REST API requests). It operates using the token bucket algorithm. When its turned on:

- Each user is given a token bucket containing the maximum number of tokens they can hold (this is their token bucket size).
- Every time they make an HTTP request one token is taken out of their bucket.
- New tokens are added into their bucket at a constant rate until it is full (this is their token bucket refill rate)
- If their bucket is empty, they cannot make requests.

When rate limiting is on and set up correctly for your instance, your users should experience improved stability and performance, and those using it reasonably wont experience any change in their ability to send it requests. The only time users will be rate limited is when they act in a way that will negatively impact Bitbucket, such as when they make requests in large bursts or when they make requests too frequently.

Example

An admin has turned on rate limiting and has set a token bucket size of 60 and a refill rate of 5.

One of their developers sends Bitbucket 60 requests in a single burst. As this developer starts with a full token bucket, all 60 requests will be successful. Their token bucket, however, will now be empty and it will begin refilling at a speed of 5 tokens per second. As it refills, they can send more requests and they wont be rate limited as long as they have enough tokens in their bucket.

Another developer sends Bitbucket 100 requests in a single burst. As their bucket can only hold 60 tokens, only their first 60 requests will be successful and then theyll be rate limited. Their token bucket will now be empty and it will begin refilling at 5 tokens per second. If they try to send the remaining 40 requests and they dont have enough tokens in their bucket theyll be rate limited again. To send all their requests successfully they'll have to rewrite their script.

How to turn on rate limiting

You need to be a System Admin or Admin to turn on rate limiting. The first time you do this youll see default values for:

1. Token bucket size - This is the number of tokens that are available to a user. It determines the maximum number of requests they can send in a burst. By default, this is set to 60
2. Token bucket refill rate - This is the number of tokens added to a users bucket per second. It determines their throughput. By default, this is set to 5

These values should be suitable for most instances and we recommend starting with them before adjusting up or down.

To turn on rate limiting:

1. Go to  > **Rate limiting**.
2. Change the status to **On**.
3. If necessary, change the **token bucket size** and **token bucket fill rate**.
4. Click **Save**.

Rate limiting can be disabled instance-wide

If required, System Admins can disable rate limiting instance-wide through the Bitbucket properties file. To find out how to do this, look for `feature.rate.limiting` in the [config properties](#) documentation.

How rate limiting works in a cluster

If your instance consists of a cluster of nodes behind a load balancer, each of your users will have a separate token bucket on each node. In other words, if you have three nodes, your users will have three token buckets.

In addition, the global settings for *token bucket size* and *refill rate* apply separately to each of these buckets. In other words, if you've set them to 60 and 5, they'll be 60 and 5 on each bucket, unless the user has an exemption.

This means that each user's ability to send requests won't change, and Bitbucket will remain protected and stable regardless of which node their requests are routed to.

Identifying users who have been rate limited

When a user is rate limited, they'll know immediately as they'll receive an HTTP 429 error message (too many requests).

There are also two ways you can identify which users have been rate limited:

1. You can look at the *Users rate limited in the past 24 hours table* on the rate limiting settings page.
2. For more detailed information you can look in the Bitbucket access log. Every time a user is rate limited an event with the label `rate-limited` will be recorded.

Lastly, you can use [JMX metrics](#) for an aggregated view of how many users have been rate limited. To allow you to do this, metrics for the number of rate limited requests and current user map size have been exposed.

Adding and editing rate limiting exemptions

To allow users to send Bitbucket a different number of requests than the global settings allow, you can add an exemption. There are two types of exemptions:

- Allow custom settings - Use this to give a user a different token bucket size and refill rate.
- Allow unlimited requests - Use this to remove users from rate limiting altogether.

To add an exemption:

1. Open the **Exemptions** tab.
2. Click **Add exemption**.
3. Find the user and choose their new settings.
4. Click **Save**.

It will be applied to the user, and will show up in the list of exemptions.

To edit or remove an exemption:

1. Open the **Exemptions** tab.

2. Find the user in the list of exemptions
3. Click the
4. Choose what youd like to do.

You can also add, edit, and remove exemptions using our [REST API](#).

How to help users avoid being rate limited

If rate limiting is set up correctly, users who send Bitbucket requests in a reasonable way wont experience any change besides improved system stability and performance. As for those users who are being rate limited, there are simple changes they can make to stop this from happening, and at the same time help out their team. They can:

- Avoid tight loops by writing scripts that wait for each REST request to finish before a new one is fired. For example, if these REST requests are being used to populate a wallboard, they can consider sending them one at a time, and only sending a new request after the previous one has finished.
- Avoid spawning multiple threads all making requests.

Use a CDN with Atlassian Data Center applications

On this page:

- [Get started with CDN](#)

[How it works](#)

[How to determine whether a CDN will help your users](#)

[What is cached?](#)

[Planning your CDN implementation](#)

- [Infrastructure requirements](#)
- [Considerations for private instances](#)
- [Marketplace apps and third party customizations](#)

If your users are distributed across the world and experience poor performance when using Data Center products, you may be able to improve their experience by using a Content Delivery Network (CDN). Common CDNs include AWS CloudFront, Cloudflare, Akamai, and others.

CDN support is available in **Data Center** editions of:

- Jira Software 8.3
- Jira Service Management (formerly Jira Service Desk) 4.3
- Confluence 7.0
- Bitbucket 6.8.

Get started with CDN

Here's a quick summary of what's involved to enable your CDN in Bitbucket Data Center:

1. Use [our template](#) to spin up an AWS CloudFront distribution, or create an account with the CDN vendor of your choice.
2. Update your load balancer and firewall to allow the CDN to reach your site.
3. In Bitbucket Data Center, provide the CDN URL, and enable CDN support.

As end users access your site, static assets will be cached on the edge server closest to them, and served from there until they expire. This means it might take some time before you can start measuring the impact of the CDN, depending on when your users are online and accessing the site in each location. We don't provide the ability to preload the cache, so assets will be cached as they are served for the first time.

See [Configure your CDN for Bitbucket Data Center](#) for the full step-by-step guide.

As always, we recommend testing this on your staging environment, before making any changes to your production site.

How it works

Static assets (such as JavaScript, CSS, and fonts) are cached on edge servers provided by a CDN vendor that are geographically closer to the user. This means when someone views a page, some of the assets needed to display the page are delivered by a server in their region, rather than from your server, known as the origin server. This can speed up page load times.



For example, if your server (known as the origin) is in Germany, a CDN can improve page load time by as much as 50% for users located in Rio de Janeiro, as static assets can be served from an edge server in Brazil. If you're new to CDNs and would like to learn more about how they work, CloudFlare provides a great introduction, see <https://www.cloudflare.com/learning/cdn/performance/>.

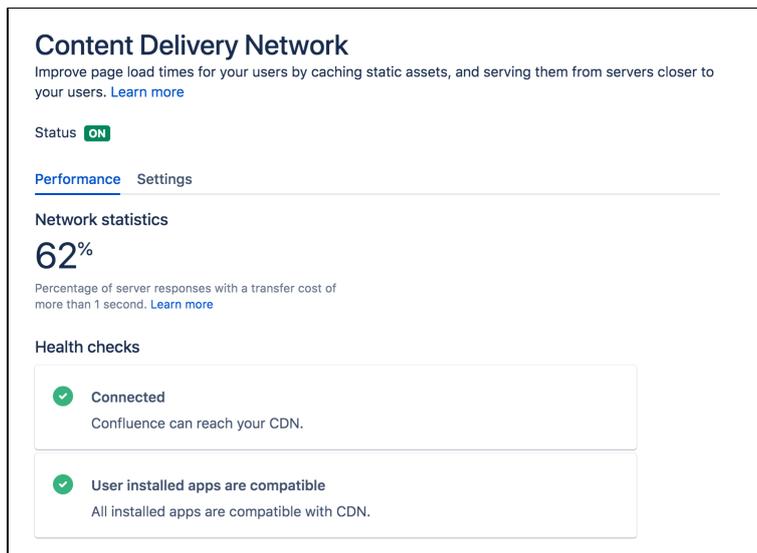
It's important to note that using a CDN will not make your application inherently faster, what it will do is reduce the load on your cluster, and reduce the latency experienced by some users, which should result in faster page load times for users.

Tests on our internal dogfooding instances located in Gdask, Poland have shown the response time for the View Issue action in Jira Data Center is ~50% faster for people accessing from US East, when CDN is enabled.

How to determine whether a CDN will help your users

A good starting point when assessing whether a CDN will help your users, is to take a look at the network overhead experienced in your site.

Go to **Content Delivery Network** in the admin console of your Data Center application. On the **Performance** tab you'll see the percentage of requests that had a transfer cost of more than one second. Put simply, the higher the percentage, the more likely it is that your users requests are being affected by network conditions, such as latency and connection quality.



Content Delivery Network
Improve page load times for your users by caching static assets, and serving them from servers closer to your users. [Learn more](#)

Status **ON**

[Performance](#) [Settings](#)

Network statistics
62%
Percentage of server responses with a transfer cost of more than 1 second. [Learn more](#)

Health checks

- Connected**
Confluence can reach your CDN.
- User installed apps are compatible**
All installed apps are compatible with CDN.

This network statistic is a useful indicator of the network conditions your users experience when using the product. If the percentage is high, it's likely that using a CDN will benefit your users in these conditions.

As users access pages in your site (for example a Confluence page, Jira issue, or Bitbucket pull request page), we measure the amount of time the browser has to wait to get the content of that page. We then subtract the time required to render the page on the server. This leaves us with the time it took to send the request and retrieve the response.

This time is dependent mostly on the latency between the server and the browser, but also includes things like SSL connection setup time.

This metric is collected on requests that don't use CDN, so it will continue to provide consistent statistics on your network, even after you enable CDN.

You should also consider where your users are geographically located. For example, if your servers are located in Frankfurt, and the majority of your teams are located in Germany and Austria, your team based in Malaysia may be suffering from high latency, resulting in slow page load times.

Network diagnostic tools such as `tracert`, `ping`, and `mtr` can be helpful to determine the amount of latency being experienced.

In these examples we'll use `tracert` to display some basic network statistics, including latency information. Remember to replace `yoursite.com` with your base URL.

In Windows, open Command Prompt and enter the following:

```
> tracert yoursite.com
```

In Linux or Mac OS, open Terminal and enter the following:

```
$ traceroute yoursite.com
```

This will display the number of hops, and three latency times, in milliseconds, for each server. Average the three figures to get the latency for that server.

The `mtr` command (my traceroute) is a useful combination of `ping` and `tracert`. You will need to install `mtr` to be able to use it in MacOS or Windows.

What is cached?

We only cache static assets served by a Data Center application or Marketplace app. These are things that are only going to change when you upgrade your Data Center application or app. Dynamic content is not cached.

Here's a summary of what will be cached when you enable CDN:

| Cached | Not cached |
|--|---|
| <ul style="list-style-type: none"> • CSS • JavaScript • Fonts | <ul style="list-style-type: none"> • attached files • pages or issues • personal information, including avatars • assets that are part of a theme |

You shouldn't need to ever manually invalidate the cache, as we handle this when you upgrade your Data Center product, or an app.

Planning your CDN implementation

Infrastructure requirements

You can use any origin pull CDN. You're responsible for any costs associated with your CDN.

We've prepared a CloudFormation template that you can use to configure Amazon CloudFront with minimal effort. You can find all our AWS deployment resources in this repository <https://bitbucket.org/atlassian/atlassian-aws-deployment/src/master/templates/cdn/>.

There are some other infrastructure requirements that you need to be aware of before you start:

- **HTTP/2 is highly recommended**
Your load balancer, firewall, or proxy should allow HTTP/2 traffic. Using HTTP/2 will provide the best performance for your end users. Check the documentation for your particular provider to find out how to do this.
- **Firewall considerations**
Your CDN must be able to access and cache static assets. If your instance is not publicly accessible will you need to make some changes to your firewall to allow requests from the CDN to pass through. We recommend using application firewalls instead of standard IP range filtering, as CDN IP ranges can change without notice.

Considerations for private instances

If your site is publicly accessible on the internet, you should be able to enable CDN without any problems.

If your site is not publicly accessible you can:

- configure your firewall to allow requests from your CDN to pass through. More information on how to do this is provided in our step-by-step guides below.
- set up your own caching servers closer to your users which will not require opening any traffic to the internet, instead of using a CDN vendor. See [How to configure Apache for caching and HTTP/2](#) to learn more about this workaround.

Marketplace apps and third party customizations

Some marketplace apps or customizations may not be compatible with the CDN feature. A health check, on the Content Delivery Network admin screen will let you know if any of your apps are not compatible.

See [User-installed apps health check fails in Data Center when configuring CDN](#) to find out what to do if any of your apps are incompatible.

Configure your CDN for Bitbucket Data Center

On this page:

- [Configure an internet facing load balancer \(optional\)](#)
 - [Add an internet-facing load balancer](#)
 - [Update your firewall rules for the internet-facing load balancer](#)
- [Configure your CDN to cache assets](#)
- [Enable CDN in Bitbucket Data Center](#)
 - [Configure CDN in Bitbucket via REST](#)
- [Troubleshooting](#)
 - [Frequently asked questions](#)

If your users are distributed across the world and experience high latency when using Bitbucket Data Center, you may be able to improve their experience by using a Content Delivery Network (CDN). Common CDNs include AWS CloudFront, Cloudflare, Azure CDN, Akamai, and others.

Head to [Use a CDN with Atlassian Data Center applications](#) to learn about our CDN capabilities, and how to assess whether it will improve your users' experience.

Once you're ready to start using a CDN, there are three main steps:

1. Configure an internet-facing load balancer (optional)
2. Configure your CDN.
3. Enable the CDN feature in Bitbucket Data Center.

Configure an internet facing load balancer (optional)

If your site is not publicly accessible, you'll need to make sure that your CDN can reach it, but only to access and cache static assets. The way you do this depends on your particular load balancer and web application firewall. Refer to the documentation for your load balancer and firewall for detailed guidance.

Add an internet-facing load balancer

Add an internet-facing load balancer to your setup. This is in addition to your primary load balancer. Your CDN is the only entity that will interact with this load balancer. We recommend you:

- Enable HTTPS - the traffic from this load balancer will be sent over the public internet and should be encrypted.
- Enable HTTP/1.1 - currently, the caching proxies and CDNs do not handle HTTP/2 well (or at all) on the way to the origin.
- For AWS deployments, you would set up an internet-facing application load balancer.

Update your firewall rules for the internet-facing load balancer

Unlike your primary load balancer, this internet-facing load balancer must be locked down to ensure that your CDN can only pull data it is allowed to cache. When configuring your firewall rules we recommend:

1. The configuration should only allow requests for paths that start with `"/s/"`. If your application is deployed with a context path (for example `yoursite.com/wiki` or `yoursite.com/jira`) you will need to include it in the path. All other requests must be blocked.
2. You can also choose to limit the allowed HTTP methods to GET, HEAD, OPTIONS.

For AWS deployments, you will configure a Web Access Control List (WebACL) in the Web Application Firewall attached to your application load balancer. The condition to use is a "string match condition" applied to "URI".

To check that your setup is secure, perform the following manual tests:

1. A GET on `https://internet-facing-proxy/` should return "403 FORBIDDEN".
2. A GET on `https://internet-facing-proxy/s` should return "403 FORBIDDEN".

3. A GET on `https://internet-facing-proxy/s/` should return "404 NOT FOUND".
4. A GET on `https://internet-facing-proxy/s/.` should return "403 FORBIDDEN".
5. A GET on `https://internet-facing-proxy/s/./s/` should return "404 NOT FOUND".

Configure your CDN to cache assets

You'll need an account with a CDN provider. You're responsible for all costs associated with your CDN. We only support serving static assets from a CDN at this time. This means page content, attached files, and personally identifiable information, including things like user avatars, won't be cached by your CDN.

We've prepared a [CloudFormation template](#) that you can use to configure Amazon CloudFront with minimal effort. You can find all our AWS deployment resources in this repository <https://bitbucket.org/atlassian/atlassian-aws-deployment/src/master/templates/cdn/>.

If you choose not to use our template, define the following in your CDN configuration. This example is based on AWS CloudFront.

| | |
|--|---|
| Origin domain name | This is your Atlassian application base URL, including the context path if you've configured one.
For example: <code>mycompany.com/confluence</code> |
| Origin path | Leave blank. There is no need to specify a path. |
| Allowed HTTP methods | Optionally limit to: GET, HEAD, OPTIONS |
| Viewer protocol policy | redirect HTTP to HTTPS |
| Object caching | Use origin cache headers |
| Forward cookies | None
This is important to make sure static assets are cached without the user context. |
| Query String Forwarding and Caching | Forward all, cache based on all |
| HTTP protocols | Must include HTTP/2 |
| Error pages/Error Caching Minimum TTL (seconds) | The default error page caching time for CloudFront is 5 minutes. Consider lowering it to a value in the range of 10-30 seconds to decrease the time required to recover from an outage. |
| Compress Objects Automatically | Yes |

Using the default should be fine for most of the other settings.

You will need to adapt this information for your particular CDN provider. You should refer to the documentation for your CDN for details, as we've found that terminology differs between CDNs.

Enable CDN in Bitbucket Data Center

Once you've configured your CDN, you can enable the CDN option in Bitbucket Data Center.

To turn on CDN:

1. Go to **Admin > Content Delivery Network**.
2. Navigate to the **Settings** tab.
3. Set the status to **On**
4. Paste the URL generated by your CDN into the URL field and hit **Validate**.

5. If successful, save your changes.

Content Delivery Network

Improve page load times for your users by caching static assets, and serving them from servers closer to your users. [Learn more](#)

Status **OFF**

Performance [Settings](#)

Status *

On

Off

CDN URL

✔ **Connected**

Your browser can reach your CDN.

As end users access Bitbucket in their browser, static assets will be cached on the edge server closest to them, and served from there until they expire. This means it might take some time before you can start measuring the impact of the CDN, depending on when your users are online and accessing the site in each location.

Configure CDN in Bitbucket via REST

You can also interact with the CDN feature using the following REST endpoint: `<base-url>/rest/static-asset-caching/configuration`

- **GET** -returns the current CDN status, and URL.
- **DELETE** -deletes the existing configuration and reverts to the default state (CDN disabled, no URL). This is useful if you can't access the UI because of a caching problem.
- **PUT** -sets the CDN URL and status to the values passed in the body of the request as follows:

```
{
  enabled: true,
  url: https://yourcdnurl.com
}
```

Troubleshooting

Here are some common problems that you may encounter.

- **We only accept HTTPS CDN URLs**
This is particularly important if you're using Azure CDN, as Azure CDN will mirror the same protocol as the originating request, which means your Data Center application will need to be provisioned with HTTPS.
- **Data Center application UI is inaccessible or not functional**
Although unlikely, a misconfiguration of your CDN or a CDN service outage may mean your application's UI is not accessible. If this happens, you will need to disable the CDN feature using the REST API, as follows.

```
curl -v -u <admin username>:<admin password> -X DELETE http://<your-base-url>/rest/static-asset-caching/configuration
```

This example uses Curl, but you can use any language. Don't forget to replace the username, password, and base URL placeholders with your own details.

- **HTTP/2 disabled**

Your load balancer, firewall, or reverse proxy should allow HTTP/2 traffic. Using HTTP/2 will provide the best performance for your end users. See [HTTP/2 health check fails in Data Center when configuring CDN](#) for more information.

- **User-installed apps may not be compatible**

This warning is displayed when we detect that a Marketplace or other user-installed app is using a deprecated method, which may result in assets being cached incorrectly. See [User-installed apps health check fails in Data Center when configuring CDN](#) for more information on what to do if you see this warning.

Frequently asked questions

Can I control what static assets are cached?

No, the application controls this. All requests for static assets are routed to the CDN. Requests for non-static assets are routed directly to your product.

Is personally identifiable information cached?

User created content, usernames, mentions, avatars etc are not static assets, so are not cached. Your CDN should also be configured to pull content from your product with cookies stripped to make sure it operates without user context.

Is dynamic content such as `batch.js` cached?

Although dynamically generated, `batch.js` is considered static content, so is cached.

Managing personal access tokens

Users can [create personal access tokens](#) and use them in place of passwords for Git over HTTPS, or to authenticate when using the Bitbucket Data Center and Server REST API . As an administrator, you can edit and revoke tokens, and set global token settings.

On this page:

- [Editing and revoking tokens](#)
- [Require token expiry](#)

Related pages:

- [Personal access tokens](#)
- [Controlling access to code](#)
- [SSH access keys for system use](#)

Editing and revoking tokens

As an administrator, you cant create tokens for users. However, once a user has created a token, you can edit or revoke it.

To edit or revoke a token:

1. Go to  > **Users**.
2. Search for the user and click on them.
3. Open the **Personal access tokens** tab.
4. Click **Edit** or **Revoke**.

Clicking Edit will allow you to change a tokens name or its permissions. If it has an expiry date, however, you will not be able to modify it. Once a tokens expiry date has been set, it cant be changed.

Require token expiry

By default, when a user is creating a personal access token, they can choose whether they want it to expire. As a system administrator, for added security you can make setting a token expiry a requirement.

To require token expiry:

1. Go to  > **Personal access tokens**.
2. Choose Yes for **Expiry required**.
3. Enter the **Max days until expiry**.
4. Click **Save**.

Connecting to a 3rd party application using Application Links

Application Links is a bundled app that allows you to connect Bitbucket Data Center and Server and other Atlassian tools to set up links, share information, and provide access to resources or functionality.

Application Links can also be used to connect Bitbucket (and other Atlassian tools) with 3rd party applications. We've done this ourselves with Jenkins: we enabled an outgoing Application Link from Bitbucket to Jenkins, allowing users to perform Jenkins actions from within Bitbucket.

On this page we use [Jenkins](#) as an example to explain how to:

- Make your 3rd party application compatible with Application Links.
- Create an outgoing Application link from Bitbucket to your 3rd party application.



Application Links is based on the OAuth 1.0 a protocol. This page explains how to set up a 3rd party application as an OAuth provider. It doesn't explain how to set it up as an OAuth consumer as this can be done easily using existing libraries.

On this page:

- [Terms used in this article](#)
- [Application link options](#)
 - [Outgoing/incoming, or both](#)
 - [Automatically created or manually created](#)
- [Making your 3rd party application compatible with Application Links](#)
 - [Configuring your 3rd party application as an OAuth provider](#)
 - [Configuring your 3rd party application as an OAuth consumer](#)

Related pages:

- [Related page](#)

Terms used in this article

- **OAuth 1.0a** - [OAuth](#) is an [open standard](#) for access delegation. It's commonly used as a way for Internet users to grant websites or applications access to their information on other websites without giving them their passwords. Application Links is based on OAuth 1.0a, and all references to OAuth in this article relate to this protocol.
- **Resource owner** - An end user who owns the resource. For example, a Jenkins Job.
- **Client** - An application that makes OAuth calls on behalf of a resource owner.
- **OAuth provider** - An application that allows a client to create a token that will be used later for authentication.
- **OAuth consumer** - A client that is an OAuth consumer. The OAuth consumer will ask the OAuth provider for various tokens. These tokens will be explained later.
- **3-legged OAuth dance** - This process is explained on Github in [these diagrams](#).

Application link options

Your Application Link can be:

- Outgoing/incoming, or both.
- Automatically created or manually created.

Outgoing/incoming, or both

If an Application Link is outgoing/incoming (outgoing for one application, incoming for the other application), only one party can perform actions on the other. If it's both outgoing and incoming, like in Atlassian products, both parties can perform actions on each other.

In our Jenkins example, the Application Link is outgoing/incoming. It is:

| | |
|---|--|
| Outgoing from Bitbucket, the OAuth consumer | Incoming for Jenkins, the OAuth provider |
|---|--|

In this arrangement:

- Jenkins *provides* access tokens to Bitbucket. Jenkins users are resource owners while the Jenkins server acts as an OAuth provider.
- Bitbucket *consumes* these tokens and uses them to perform actions on Jenkins.
- Jenkins, however, cannot perform actions on Bitbucket.

This also means that Jenkins needs to manage token creation and handling, and it needs to expose endpoints to generate request tokens, to generate access tokens, and for authorization.

Automatically created or manually created

When connecting Atlassian applications, Application Link creation is automatic. How this works is explained in the [Jira documentation](#).

When connecting Atlassian applications with 3rd party applications, the Application Link must be created manually. An example of how to do this is explained in our plugin [readme](#).

In our Jenkins example, Jenkins is an OAuth provider only, so the Application Link needs to be created manually. This process can be tedious, but it only needs to be done once.

Making your 3rd party application compatible with Application Links

To make your 3rd party application compatible with Application Links:

1. Decide if the Application Link from Bitbucket needs to be outgoing, incoming, or both.
2. Implement OAuth:
 - For an outgoing link from Bitbucket, your 3rd party application needs to be configured as an OAuth provider.
 - For a bidirectional link from Bitbucket, your 3rd party application needs to be configured as an OAuth provider and also needs to act as an OAuth Consumer.

See below for how to do this:

Configuring your 3rd party application as an OAuth provider

To be an OAuth provider, your 3rd party application needs to:

1. Do token handling.
2. Have exposed endpoints:
 - To generate request tokens
 - To generate access tokens
 - For authorization to generate authorization code.]

For a more concrete example of how this can be done, take a look at the [OAuth provider guide](#) that we've written for our [integration plugin for Jenkins](#). It explains how you can potentially copy and reuse the Application Link module to quickly configure your 3rd party application as an OAuth provider.

Configuring your 3rd party application as an OAuth consumer

There are many libraries that work out of the box for configuring your 3rd party application as an OAuth consumer. For example, [Scribe](#).

Once your 3rd party application is an OAuth provider and consumer, you can manually create a bidirectional Application Link between it and Bitbucket and perform actions on both ends.

Setting a system-wide default branch name

The default branch for a repository is its integration branch for work. In pull request workflows, it's where feature branches are targeted. It's also the first branch created in a new repository.

As a Bitbucket Data Center and Server administrator, you can select how this branch is named by going to  > **Default branch name**. On this page your options are:

On this page:

Related pages:

- [Branches](#)

| Option | Description |
|---------------------------|--|
| Use the Git default name | The name given to the default branch is based on the installed Git version: <ul style="list-style-type: none">• In Git 2.27 and below, the default name is hard-coded as master.• In Git 2.28+, the default name is master, but can be configured. |
| Set a custom default name | You can set an instance-wide custom default name for this branch. This custom name will be used instead of the Git name, and when a user creates a new repository they'll see it prefilled in the default branch name field. They can then use this name, or enter a branch name of their own. |

 When creating a new repository, users can leave the Default branch name field empty. If they do, the name given to the default branch will be the default branch name that has been configured for the instance.

Automatically decline inactive pull requests

Bitbucket Data Center and Server can automatically decline pull requests that are inactive, reducing the number of pull requests in the list that can add up. For example, if an older pull request has not had any activity (like commits, comments or approvals) in a set amount of time, it will be declined. By default, the option is on and set to 4 weeks. This period of inactivity can be overwritten for projects and repositories.

On this page:

- [Automatically decline inactive pull requests for all repositories in a project](#)
- [Automatically decline inactive pull requests for a repository](#)

There is an instance-wide configuration property for this feature in the `bitbucket.properties` file that allows admins to disable or turn the feature off.

```
feature.pull.request.auto.decline=false
```

Automatically decline inactive pull requests for all repositories in a project

To configure the automatic declining of inactive pull requests in a project:

1. Select **Project settings** > **Auto-decline**.
2. Select the **Auto-decline pull requests** check-box to enable it.
3. Set the amount of time before a pull request will be automatically declined from 1, 2, 4, 8, or 12 weeks.
4. Select **Save**.

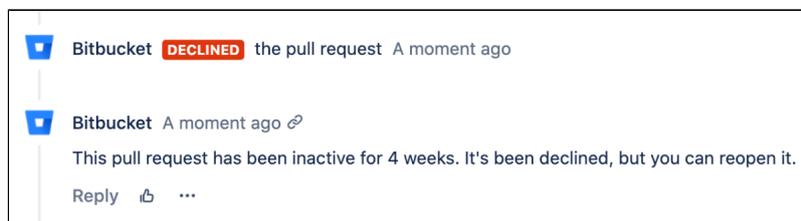
Automatically decline inactive pull requests for a repository

 Configuring this setting at the repository level, will override the setting for it at the project level.

To configure the automatic declining of inactive pull requests in a single repository:

1. Select **Repository settings** > **Auto-decline**.
 - a. By default, settings are inherited from the project settings.
2. Select **Use custom settings**.
3. Select the **Auto-decline pull requests** check-box to enable it.
4. Set the amount of time before a pull request will be automatically declined from 1, 2, 4, 8, or 12 weeks.
5. Select **Save**.

Once the configuration is saved, inactive pull requests will become declined by the Bitbucket system user. This will add a comment indicating that the reason for the decline is due to inactivity.



Encrypt database password

To add extra security to your Bitbucket Data Center and Server instance, you can encrypt the database password that is stored in the `bitbucket.properties` file. There are different encryption methods for both basic and advanced users. Additionally, you can create your own encryption mechanism.



Your Bitbucket instance needs a plain-text password to connect to the database, so the configuration file will contain all the information needed to decrypt the password. It is necessary to protect access to Bitbucket and database servers to reach the required level of security.

Basic encryption

This method uses a Base64 cipher, which is simple obfuscation. It is a straightforward solution for users who don't want to store database passwords in plain-text.

[Learn more about basic encryption](#)

Advanced encryption

This method allows you to choose an algorithm to encrypt a database password. It provides more security as you don't have to store the encrypted password anywhere in the configuration file, which makes it difficult for unauthorized parties to find and decrypt it.

[Learn more about advanced encryption](#)

Custom encryption

If you have special requirements for database password encryption, you can create your own encryption mechanism based on our examples.

[Learn more about custom encryption](#)

Basic encryption

Basic encryption is one of the three ways to encrypt database passwords in Bitbucket Data Center and Server. See also, [Advanced encryption](#) and [Custom encryption](#).

For this method, we'll use Base64Cipher, which is a way to achieve simple obfuscation of the database password.

Step 1. Encrypt the password

1. Go to `<Bitbucket-installation-directory>/tools/atlassian-password`.
2. Run the following command to encrypt your password:

```
java -cp ".*" com.atlassian.db.config.password.tools.CipherTool
```

- **-- silent -s**: limits logging to a minimum
- **-- help -h**: prints a help message with all parameters
- **-- mode -m**: defines what to do with the password, either **encrypt** or **decrypt**. If omitted, 'encrypt' will be used.
- **-- password -p**: plain text password. If omitted, you'll be asked to enter it. We recommend that you omit this parameter so that your password is not stored in the history.

```
main DEBUG [db.config.password.DefaultCipherProvider] Initiate cipher provider class: com.atlassian.db.config.password.ciphers.base64.Base64Cipher
main DEBUG [password.ciphers.base64.Base64Cipher] Initiate Base64Cipher
main DEBUG [password.ciphers.base64.Base64Cipher] Encrypting data...
main DEBUG [password.ciphers.base64.Base64Cipher] Encryption done. Success!
Success!
For Jira, put the following lines in database config xml file:

<atlassian-password-cipher-provider>com.atlassian.db.config.password.ciphers.base64.Base64Cipher<
/atlassian-password-cipher-provider>
<password>ZGZnZGZn</password>

and restart the instance.

For Bitbucket, set the following properties in bitbucket.properties:

jdbc.password.decrypter.classname=com.atlassian.db.config.password.ciphers.base64.Base64Cipher
jdbc.password=ZGZnZGZn

and restart the instance.
```

Step 2. Add the encrypted password to the properties file

1. Go to the Bitbucket home directory and back up the `bitbucket.properties` file. Move the backup to a safe place outside of your instance.
2. In the `bitbucket.properties` file, replace the `jdbc.password` property with the password generated by CLI. Also, specify the `jdbc.password.decrypter.classname` property. For example:

```
jdbc.password.decrypter.classname=com.atlassian.db.config.password.ciphers.base64.Base64Cipher
jdbc.password=put-the-obfuscated-password-generated-by-the-cli-tool-here
```

3. Restart Bitbucket.

Decrypting the password

To decrypt the password, extend the command with the **-m decrypt** parameter:

```
java -cp ".*" com.atlassian.db.config.password.tools.CipherTool -m decrypt
```

When asked for a password, provide the encrypted one from `yourbitbucket.propertiesfile`.

Troubleshooting

This means that Bitbucket couldn't connect to the database to access your configuration, most likely because of an error with decrypting your password.

To solve this problem, open `<Bitbucket_home_directory>/log/atlassian-bitbucket.log`, and check for `DataSourcePasswordDecryptionException`. For example:

```
com.atlassian.stash.internal.jdbc.DataSourcePasswordDecryptionException: java.lang.
IllegalArgumentException: Illegal base64 character 25
```

The exception contains details about the error. If the error is `java.lang.IllegalArgumentException`, you will need to encrypt the password again.

To investigate this problem, open `<Bitbucket_home_directory>/log/atlassian-bitbucket.log`, and check for `JdbcSQLExceptions`. The messages should be pretty clear as to what went wrong.

You'll probably see the following messages:

```
Wrong user name or password [28000-176]
```

This means that Bitbucket decrypted the password successfully, but the password itself is incorrect. You can verify that by completing these steps:

1. Open the `bitbucket.propertiesfile`, and copy the encrypted password.
2. [Decrypt the password](#).
3. Check if the decrypted password is the same as the one in your backup `bitbucket.propertiesfile`.

To disable database password encryption, remove the `jdbc.password.decrypter.classname` property from the `bitbucket.propertiesfile`, and change the value of `jdbc.password` to the unencrypted in your backup.

Advanced encryption

In this method, you'll use `AlgorithmCipher` that allows you to choose the algorithm used to encrypt your password in Bitbucket Data Center and Server.

Before you begin: Prepare the JSON object

You'll need to provide all arguments required to encrypt your password in a JSON object. Prepare beforehand by using the information and examples below.

| Field | Description |
|--------------------------------|---|
| <code>plainTextPassword</code> | Password in plain text. |
| <code>algorithm</code> | You can choose one of the following algorithms: <ul style="list-style-type: none">• AES/CBC/PKCS5Padding• DES/CBC/PKCS5Padding• DESede/CBC/PKCS5Padding |
| <code>algorithmKey</code> | The algorithm key must correspond with the algorithm chosen above: <ul style="list-style-type: none">• AES• DES• DESede |

```
{ "plainTextPassword": "yourPassword", "algorithm": "AES/CBC/PKCS5PADDING", "algorithmKey": "AES" }
```

Step 1. Encrypt the password

1. Go to `<Bitbucket-installation-directory>/tools/atlassian-password`.
2. Run the following command to encrypt your password:

```
java -cp ".*" com.atlassian.db.config.password.tools.CipherTool -c com.atlassian.db.config.password.ciphers.algorithm.AlgorithmCipher
```

- **-- silent -s**: limits logging to a minimum
- **-- help -h**: prints a help message with all parameters
- **-- mode -m**: defines what to do with the password, either **encrypt** or **decrypt**. If omitted, 'encrypt' will be used.
- **-- password -p**: JSON object with required arguments. If omitted, you'll be asked to enter a password. We recommend that you omit this parameter so that your password is not stored in the terminal history.

3. After running the command, you'll be asked to provide the required arguments in a JSON object in a single line. Prepare it based on the information from [Before you begin](#).

When encrypting your password, the encryption tool generates three files and prints the output JSON object that you'll later add to the `bitbucket.properties` file. The next step discusses how to secure those files.

Step 2. Secure the generated files

Move the files generated by the tool to a secure place. Change them to read-only and accessible only to the user running Bitbucket. Note that if a multi-node cluster is in use, then the files should be available on the same path for all nodes. Bitbucket needs to be able to access and read those files to decrypt your password and connect to the database.

The following files are generated:

- `javax.crypto.SealedObject_[timestamp]`
File with the encrypted password.
- `javax.crypto.spec.SecretKeySpec_[timestamp]`
Key used to encrypt your password. You will need this file to decrypt your password.
- `java.security.AlgorithmParameters_[timestamp]`
Algorithm parameters used to encrypt your password. You will need this file only if you want to [recreate an encrypted password](#).

Step 3. (Optional) Store file paths as environment variables

This step is optional. You can store paths to the generated files as environment variables. If the paths aren't present in the `bitbucket.properties` file and the `jdbc.password` is an empty JSON object (`{}`), `AlgorithmCipher` will look for them in the environment. This way, file paths are not stored in the file, making it difficult to locate the files used for encryption.

1. Store the two generated files as environment variables. You don't need to add the file with algorithm parameters, because `AlgorithmCipher` does not use it to decrypt the password. You must set the following environment variables to the correct values in any of the scripts used for launching your Bitbucket instance:

```
com_atlassian_db_config_password_ciphers_algorithm_javax_crypto_spec_SecretKeySpec
com_atlassian_db_config_password_ciphers_algorithm_javax_crypto_SealedObject
```

2. Edit the output from the first step, [Encrypt the password](#), and remove paths to the files. The decrypter class name and password should match the following:

```
jdbc.password.decrypter.classname=com.atlassian.db.config.password.ciphers.algorithm.
AlgorithmCipher
jdbc.password={}
```

Step 4. Add the encrypted password to bitbucket.properties

1. Go to the Bitbucket home directory and back up the `bitbucket.properties` file. Move the backup to a safe place, ideally outside your instance.
2. In the `bitbucket.properties` file, replace the `jdbc.password` property with the output JSON object. Depending on whether you're using environment variables or not, adjust the JSON object to one of the following examples:

- If you're storing file paths as environment variables, remove the paths from the output. The properties should look like the following:

```
jdbc.password.decrypter.classname=com.atlassian.db.config.password.ciphers.algorithm.
AlgorithmCipher
jdbc.password={}
```

- If you're not using environment variables and want to stick to file paths in the `bitbucket.properties` file, make sure you update their paths after moving them to a secure place. The properties should look like the following:

```
jdbc.password.decrypter.classname=com.atlassian.db.config.password.ciphers.algorithm.
AlgorithmCipher
jdbc.password={"sealedObjectFilePath":"/home/bitbucket/javax.crypto.
SealedObject_123456789","keyFilePath":"/home/bitbucket/javax.crypto.spec.
SecretKeySpec_123456789"}
```

- **WINDOWS** You shouldn't use backslashes in the path to avoid JSON parsing errors. The paths should look like the following example:

```
jdbc.password.decrypter.classname=com.atlassian.db.config.password.ciphers.algorithm.
AlgorithmCipher
jdbc.password={"sealedObjectFilePath":"C:/bitbucket/javax.crypto.SealedObject_123456789", "
keyFilePath":"C:/bitbucket/javax.crypto.spec.SecretKeySpec_123456789" }
```

3. Restart Bitbucket.

Decrypt the password

To decrypt an encrypted password, extend the command used earlier with the **-m decrypt** parameter:

```
java -cp ".*" com.atlassian.db.config.password.tools.CipherTool -c com.atlassian.db.config.password.
ciphers.algorithm.AlgorithmCipher -m decrypt
```

When asked for the password, provide the JSON object from your `bitbucket.properties` file.

Sample JSON object when using file paths

```
{"sealedObjectFilePath":"/home/bitbucket/javax.crypto.SealedObject_123456789", "keyFilePath":"/home
/bitbucket/javax.crypto.spec.SecretKeySpec_123456789" }
```

Sample JSON object when using environment variables

```
{}
```

Recreate an encrypted password

If you lose an encrypted password and try to encrypt the plain text password once again, the new encrypted password will look different. This is not an issue, as it will still represent the same plain text password. However, in some cases, you might want to keep it consistent, for example by having the same encrypted password when a Bitbucket instance is migrated to another server.

To encrypt the password in the exact same way as you did before, you will need the key used to encrypt the original password and the algorithm parameters. Both of these were generated by the encryption tool and saved in the following files:

- **Key:** `javax.crypto.spec.SecretKeySpec_[timestamp]`
- **Algorithm parameters:** `java.security.AlgorithmParameters_[timestamp]`

Once you've located these files, you can point the encryption tool to their location by using two extra fields in the JSON object.

| Field | Description |
|--|---|
| <code>keyFilePath</code> | Path to a file that contains the key used to encrypt your original password, e.g. <code>javax.crypto.spec.SecretKeySpec_[timestamp]</code> .

If you stored the file path as environment variable, you can omit this parameter. |
| <code>algorithmParametersFilePath</code> | Path to a file that contains the algorithm parameters used to encrypt your original password, e.g. <code>java.security.AlgorithmParameters_[timestamp]</code> . |

```
{"plainTextPassword":"yourPassword", "algorithm":"AES/CBC/PKCS5PADDING", "algorithmKey":"AES",
"algorithmParametersFilePath":"java.security.AlgorithmParameters_123456789", "keyFilePath":"javax.
crypto.spec.SecretKeySpec_123456789" }
```

To encrypt the password, follow the steps in the first step, [Encrypt the password](#), and use the JSON object with the key and algorithm parameters.

Troubleshooting

This means that Bitbucket couldn't connect to the database to access your configuration, most probably because of an error with decrypting your password.

To solve this problem, open `<Bitbucket_home_directory>/log/atlassian-bitbucket.log`, and check for `DataSourcePasswordDecryptionException`

For example:

```
com.atlassian.stash.internal.jdbc.DataSourcePasswordDecryptionException: java.lang.  
IllegalArgumentException: <>
```

This exception contains details about the error. If the error is `java.lang.
IllegalArgumentException`, you will need to encrypt the password again.

- If the error is related to missing files, there might be a problem with your environment variables. They could have been deleted, or have not been set correctly. To verify that, try adding file paths to the JSON object in the `thebitbucket.properties` file.
- If you're seeing some Bouncy Castle errors, you will need to encrypt the password again.

To investigate this problem, open `<Bitbucket_home_directory>/log/atlassian-bitbucket.log`, and check for `JdbcSQLExceptions`. The messages should be pretty clear as to what went wrong.

You'll likely see the following message:

```
Wrong user name or password [28000-176]
```

This means that Bitbucket decrypted the password successfully, but the password itself is incorrect. You can verify that by completing these steps:

1. Open `thebitbucket.properties` file, and copy the encrypted password.
2. [Decrypt the password](#).
3. Check if the decrypted password is the same as the one in your backup `thebitbucket.properties` file.

To disable database password encryption, remove the `jdbc.password.decrypter.classname` property from the `thebitbucket.properties` file, and change the encrypted password to the plain text one.

Custom encryption

In addition to the basic and advanced encryption methods that you can use in Bitbucket Data Center and Server, you can also choose to create your own Cipher. This might be especially useful if:

- you're required to use a specific vault to store the password
- you want to use encryption algorithms beyond those we ship with Bitbucket

Step 1. Create a Maven project and get API dependencies

1. Get `password-cipher-api` and `password-cipher-base` dependencies.
 - a. Go to `<Bitbucket_installation_directory>/atlassian-bitbucket/WEB-INF/lib`.
 - b. Copy the following jar files:
 - **password-cipher-api-<version>.jar**: this file contains the API
 - (optional) **password-cipher-base-<version>.jar**: this file contains some sample implementations
2. Create a Maven project.
3. Go to `resources` and create a new folder, named `libs`.
4. Copy the jar files to the `libs` folder.
5. Next, use the following pom:

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId><your_group_ID></groupId>
<artifactId><your_artifact_ID></artifactId>
<version><your_version></version>

<properties>
<maven.compiler.source>1.8</maven.compiler.source>
<maven.compiler.target>1.8</maven.compiler.target>
</properties>

<repositories>
<repository>
<id>local-maven-repo</id>
<url>file:///${project.basedir}/libs</url>
</repository>
</repositories>

<build>
<resources>
<resource>
<directory>src/main/resources/libs</directory>
<excludes>
<exclude>*</exclude>
</excludes>
<filtering>>false</filtering>
</resource>
</resources>
</build>

<dependencies>
<dependency>
<groupId>com.atlassian.db.config</groupId>
<artifactId>password-cipher-api</artifactId>
<version><api_version></version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>com.atlassian.db.config</groupId>
<artifactId>password-cipher-base</artifactId>
<version><base_version></version>
<scope>provided</scope>
</dependency>
</dependencies>
</project>

```

Step 2. Implement the Cipher interface

The Cipher interface contains two methods that you need to implement according to your requirements; `encrypt` and `decrypt`. `decrypt` is called during Bitbucket startup, which means that long-running tasks can affect the startup time. `encrypt` is not called by Bitbucket, as it's only used in the encryption tool.

You can use `Base64Cipher` and `AlgorithmCipher` as examples.

Step 3. Test your implementation

The encryption tool described in [Basic encryption](#) and [Advanced encryption](#), uses the same code as Bitbucket to decrypt the password. You can use it to test your implementation.

Assuming that the CLI and your jar is in the same folder:

```
java -cp ".*" com.atlassian.db.config.password.tools.CipherTool -c your.package.here.ClassName
```

Step 4. Make your library available

Bitbucket must be able to access your library. Your class will be instantiated using reflection. Put the library in the following directory:

```
<Bitbucket_home_directory>/lib
```

Advanced repository management

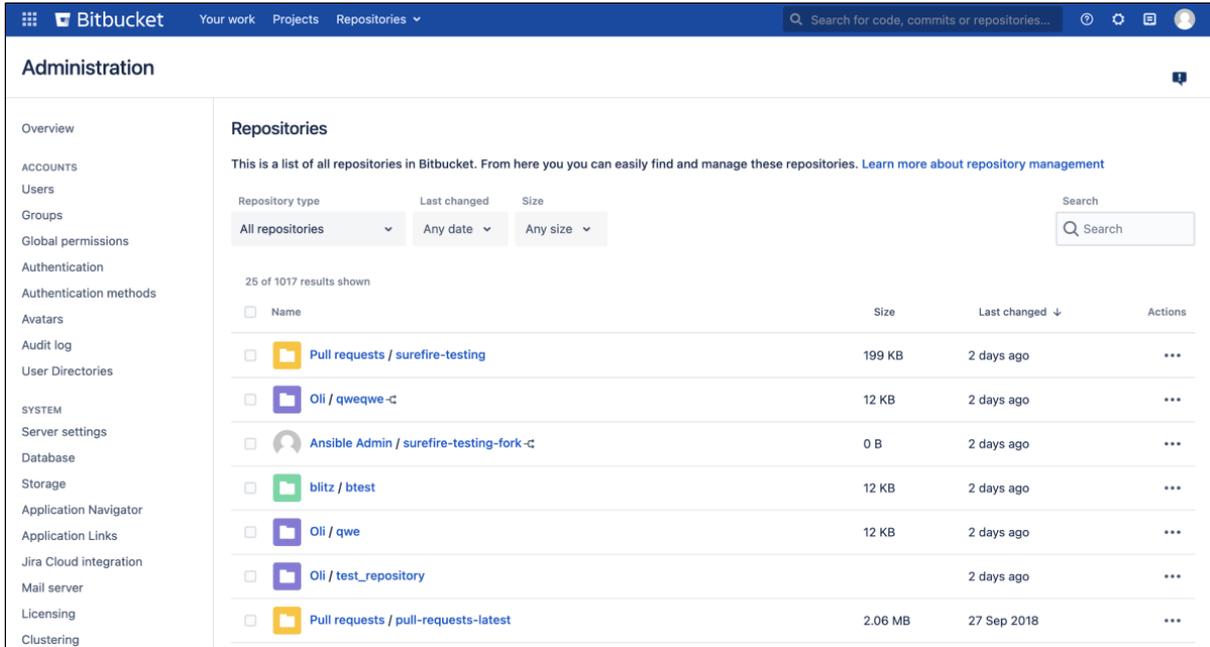
 This feature is available with a Bitbucket Data Center license.

In Bitbucket Data Center, you can manage all of the repositories in your instance from the **Repositories** page in the Administration area.

On this page

- [Repository size](#)
- [Related repositories](#)
- [Delete a repository](#)
- [Move a repository](#)

Related pages: [Forks](#)



The screenshot shows the Bitbucket Administration interface. The left sidebar contains navigation links for Overview, ACCOUNTS (Users, Groups, Global permissions, Authentication, Authentication methods, Avatars, Audit log, User Directories), SYSTEM (Server settings, Database, Storage, Application Navigator, Application Links, Jira Cloud integration, Mail server, Licensing, Clustering), and Repositories. The main content area is titled 'Repositories' and includes a search bar and filter options for Repository type (All repositories), Last changed (Any date), and Size (Any size). Below the filters, it shows '25 of 1017 results shown' and a table of repository entries.

| <input type="checkbox"/> | Name | Size | Last changed ↓ | Actions |
|--------------------------|---|---------|----------------|---------|
| <input type="checkbox"/> |  Pull requests / surefire-testing | 199 KB | 2 days ago | ... |
| <input type="checkbox"/> |  Oli / qweqwe-c | 12 KB | 2 days ago | ... |
| <input type="checkbox"/> |  Ansible Admin / surefire-testing-fork-c | 0 B | 2 days ago | ... |
| <input type="checkbox"/> |  blitz / btest | 12 KB | 2 days ago | ... |
| <input type="checkbox"/> |  Oli / qwe | 12 KB | 2 days ago | ... |
| <input type="checkbox"/> |  Oli / test_repository | | 2 days ago | ... |
| <input type="checkbox"/> |  Pull requests / pull-requests-latest | 2.06 MB | 27 Sep 2018 | ... |

Filtering options include:

- **Repository type** - helps you filter on repositories by their type, such as those that are:
 - in projects
 - public
 - personal
- **Last changed** - lets you specify a date range to find repositories based on the latest changes made to them such as:
 - create
 - push
 - ref-change
 - move
 - changes to name or description
 - changes to settings for default branch, forking, and public access
- **Size** - lets you find repositories based on their size
- **Search** - find repositories by name, description, project, or user.

To view all repositories in Bitbucket, go to  **Repositories** (under Git). Select all or multiple repositories from the list and you'll then have options to perform bulk delete or move actions on them.

Repository size

The repositories in your instance may also display an approximate size. For repositories without forks, the size is calculated by the size of their objects. For forks, the size is calculated by the size of their objects unique to the fork that is not shared with the upstream repository.

If a forked repository has not had anything added to it, it may display as 0 value. If there is no value shown, the size of the repository couldn't be calculated.

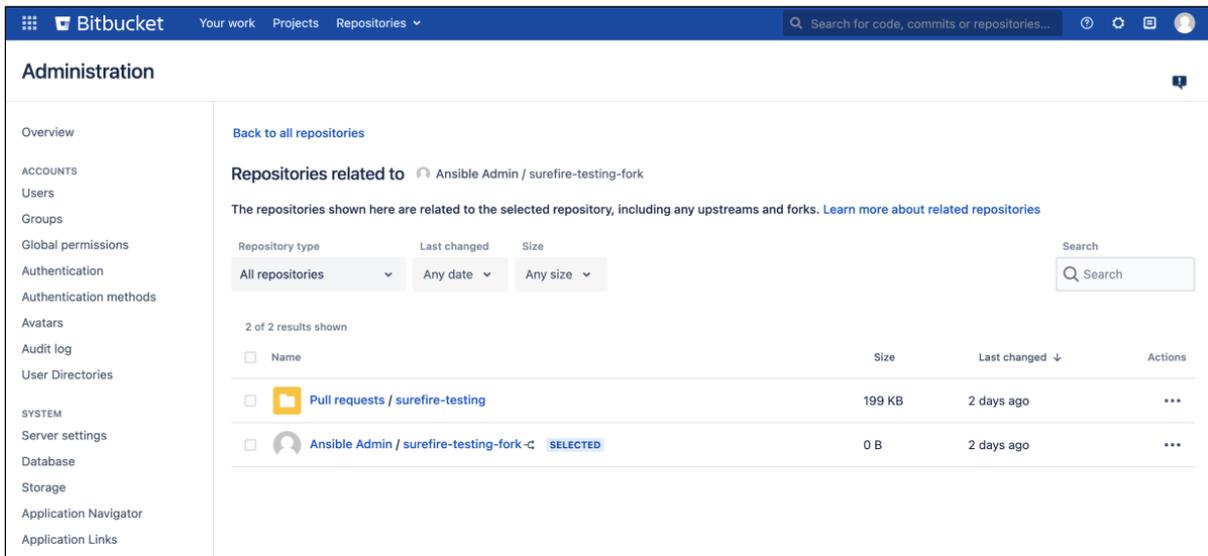
Related repositories

Advanced repository management provides the ability to review all repositories that are related to one selected from the Repositories page.

To view all related repositories:

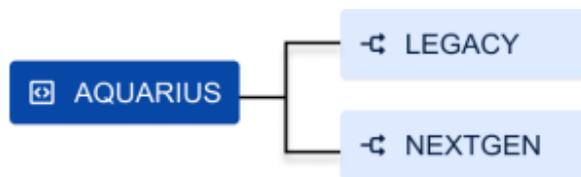
1. On the Repositories page, select > **Related repositories** next to a single repository.
2. A new page then opens so that you can review all related forks and upstream repositories.

The same filter options are available here as on the **Repositories** page.

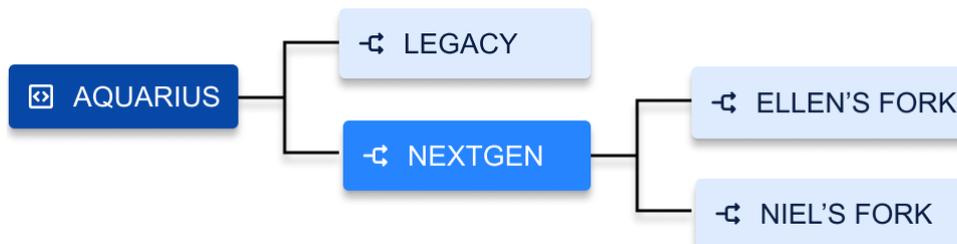


When you [fork a repository](#), you create a *hierarchy* of *related repositories*. The original repository is the *root*, and any parent repository of a fork is considered an *upstream* repository.

For example, the below hierarchy shows us that AQUARIUS is the *root* repository as well as the *upstream* to forks LEGACY and NEXTGEN.



NEXTGEN is forked to create personal repositories. So then, NEXTGEN becomes an *upstream* repository with two *direct forks*, ELLENS FORK and NIELS FORK.



Delete a repository

You can choose to delete one or multiple repositories at once. Deleting a forked repository does not delete the upstream repository, nor will deleting an upstream delete any of the forked repositories.

 Deleting a repository that has related repositories will not remove its contents from the disk until all of its related repositories are deleted.

To delete a single repository:

1. Select a repository from the list.
2. Select > **Delete**.
3. To confirm, select **Delete**.

To delete more than one repository:

1. Select all or individual checkboxes next to each repository.
2. Select **Delete** .
3. To confirm, select **Delete**.

Move a repository

You can choose to move one or multiple repositories at once.

To move a single repository to another project:

1. Select a repository from the list.
2. Select > **Move**.
3. Select the project from the **Project** list where you want to move the repository.
4. If required, change the name of the repository.
5. To confirm, select **Move**.

The repository will inherit the access rights of the project selected, and the clone URL is modified.

You'll have to [update the remote using the command](#), `git remote set-URL`.

To move more than one repository:

1. Select all or individual checkboxes next to each repository.
2. Select **Move** .
3. Select the project from the **Project** list where you want to move the repositories.
4. To confirm, select **Move**.

Xcode for Bitbucket Server

Xcode is the IDE (Integrated Development Environment) for Apple platforms. This app adds a 'Clone in Xcode' button for relevant repositories in Bitbucket Server and Bitbucket Data Center. It works with Xcode 10, the latest version of Xcode, announced at WWDC on 4th June 2018.

When you browse to a repository which has Xcode workspace files, a new "Clone in Xcode" button appears in the clone dialog. This will open Xcode and set it up to clone that repository using whatever URL is active in the drop-down at the time. That means you can clone via HTTP or SSH, from a mirror or from the host, depending on what you select. After the clone completes, Xcode will open the project/workspace.

"Clone in Xcode" is only shown for users browsing from a Mac, and only if it's been enabled. There are two ways to enable the feature:

- Access Bitbucket Server using Xcode 10, which offers the ability to integrate with Bitbucket Cloud and Server. This will automatically enable Xcode features for the user account used.
- Visit your account settings and toggle "Enable 'Clone in Xcode'" (Click your avatar in the upper right and "Manage account")

Once the feature is enabled, any repository which contains an Xcode project or workspace--identified by the presence of `.xcodeproj` and/or `.xcworkspace` directories/files--will display "Clone in Xcode" in the clone dialog.

Data pipeline

 This feature is available with a Bitbucket Data Center license.

Data pipeline provides an easy way to export data from Jira, Confluence, or Bitbucket, and feed it into your existing data platform (like [Tableau](#) or [PowerBI](#)). This allows you to:

- generate richer reports and visualizations of site activity
- better understand how your teams are using your application
- make better decisions on optimizing the use of Jira or Confluence in your organization

You can trigger a data export of the current state data through the REST API, and view the status of your exports in your applications admin console. Data will be exported in CSV format. You can only perform one data export at a time.

For a detailed reference of the exported data's schema, see [Data pipeline export schema](#).

Data pipeline is available in Data Center editions of:

- Jira 8.14 and later
- Confluence 7.12 and later
- Bitbucket 7.13 and later

Requirements

To trigger data exports through the REST API, you'll need:

- A valid Bitbucket Data Center license
- Bitbucket system admin [global permissions](#)

Considerations

There are a number of security and performance impacts you'll need to consider before getting started.

Security

The export will include all data, including PII (Personally Identifiable Information) and restricted content. This is to provide you with as much data as possible, so you can filter and transform to generate the insights you're after.

If you need to filter out data based on security and confidentiality, this must be done after the data is exported.

Exported files are saved in your shared home directory, so you'll also want to check this is secured appropriately.

Performance impact

Exporting data is a resource-intensive process impacting application nodes, your database, and indexes. In our internal testing, we observed performance degradation in all product functions on the node actively performing an export.

To minimize the risk of performance problems, we strongly recommend that you:

- Perform the data export during hours of low activity, or on a node with no activity.
- Limit the amount of data exported through the `fromDate` parameter, as a date further in the past will export more data, resulting in a longer data export.

On this page:

- [Requirements](#)
- [Considerations](#)
- [Performing the data export](#)
- [Automatic data export cancellations](#)
- [Configuring the data export](#)
- [Check the status of an export](#)
- [Output files](#)
- [Location of exported files](#)
- [Troubleshooting failed exports](#)

Our test results showed the following approximate durations for the export.

| | Number | Approximate export duration |
|----------------------|------------|-----------------------------|
| Users | 25,000 | less than a minute |
| Repositories | 6,500 | less than a minute |
| Commits | 18 million | 4 hour 30 minutes |
| Pull Requests | 250,000 | 30 minutes |

The total export time was around 5 hours.

i Test performance VS production

The performance data presented here is based on our own internal testing. The actual duration and impact of dataexport on your own environment will likely differ depending on your infrastructure, configuration, and load.

Our tests were conducted on a two node Data Center instance in AWS:

- EC2 instance type: `m5d.4xlarge`
- RDS instance type: `db.m4.4xlarge`

Performing the dataexport

Use the data pipeline REST API to export data.

i If your application is configured to use a context path, such as `/jira`, remember to include this in the `<base-url>` in the examples below.

To export the current state data, make a POST request to `<base-url>/rest/datapipeline/latest/export`.

Use the `fromDate` parameter to limit the data exported to just entities created or updated after the `fromDate` value.

This parameter only accepts date values set in ISO 8601 format (yyyy-MM-ddTHH:mmTZD). For example:

- `2020-12-30T23:01Z`
- `2020-12-30T22:01+01:00`
(you'll need to use URL encoding in your request, for example `2020-12-30T22%3A03%2B01%3A00`)

If you trigger an export without the `fromDate` parameter, all data from the last 365 days will be exported.

Here is an example request, using cURL and a personal access token for authentication:

```
curl -H "Authorization:Bearer ABCD1234" -H "X-Atlassian-Token: no-check"
-X POST https://myexamplesite.com/rest/datapipeline/latest/
export?fromDate=2020-10-22T01:30:11Z
```

The `"X-Atlassian-Token: no-check"` header is only required for Confluence. You can omit this for Jira.

The POST request returns the following responses.

| Code | Description |
|------|-------------|
|------|-------------|

| | |
|-----|--|
| 202 | <p>Dataexport started. For example:</p> <pre data-bbox="316 197 1430 501">{ "startTime": "2021-03-03T12:08:24.045+11:00", "nodeId": "node1", "jobId": 124, "status": "STARTED", "config": { "exportFrom": "2020-03-03T12:08:24.036+11:00", "forcedExport": false } }</pre> |
| 409 | <p>Another dataexport is already running:</p> <pre data-bbox="316 600 1430 927">{ "startTime": "2021-03-03T12:08:24.045+11:00", "nodeId": "node1", "jobId": 124, "status": "STARTED", "config": { "exportFrom": "2020-03-03T12:08:24.036+11:00", "forcedExport": false } }</pre> |

422 Data export failed due to an inconsistent index:

```
{
  "startTime": "2021-01-13T09:01:01.917+11:00",
  "completedTime": "2021-01-13T09:01:01.986+11:00",
  "nodeId": "node2",
  "jobId": 56,
  "status": "FAILED",
  "config": {
    "exportFrom": "2020-07-17T08:00:00+10:00",
    "forcedExport": false
  },
  "errors": [
    {
      "key": "export.pre.validation.failed",
      "message": "Inconsistent index used for export job."
    }
  ]
}
```

If this occurs, you may need to reindex and then retry the data export.

Alternatively, you can force a data export using the `forceExport=true` query parameter. However, forcing an export on an inconsistent index could result in incomplete data.

The following response is returned when you force an export on an inconsistent index to warn you that the data might be incomplete.

```
{
  "startTime": "2021-01-13T09:01:42.696+11:00",
  "nodeId": "node2",
  "jobId": 57,
  "status": "STARTED",
  "config": {
    "exportFrom": "2020-07-17T08:01:00+10:00",
    "forcedExport": true
  },
  "warnings": [
    {
      "key": "export.pre.validation.failed",
      "message": "Inconsistent index used for export job."
    }
  ]
}
```

Automatic data export cancellations

If a node running a data export is gracefully shut down, the export will be automatically marked as `CANCELLED`.

However, if the JVM is not notified after a crash or hardware-level failure occurs, the export process may get locked. This means you'll need to manually mark the export as `CANCELLED` by making a `DELETE` request. This releases the process lock, allowing you to perform another data export.

Configuring the dataexport

You can configure the format of the export data through the following [configuration properties](#).

| Default value | Description |
|--|-------------|
| <code>plugin.data.pipeline.embedded.line.break.preserve</code> | |

| | |
|---|---|
| <code>false</code> | Specifies whether embedded line breaks should be preserved in the output files. Line breaks can be problematic for some tools such as Hadoop.

This property is set to <code>False</code> by default, which means that line breaks are escaped. |
| <code>plugin.data.pipeline.embedded.line.break.escape.char</code> | |
| <code>\\n</code> | Escaping character for embedded line breaks. By default, we'll print <code>\\n</code> for every embedded line break. |

The following additional properties only apply to Bitbucket.

| Default value | Description |
|---|--|
| <code>plugin.data.pipeline.bitbucket.export.personal.forked.repository.commits</code> | |
| <code>false</code> | Specifies whether commits from forked repositories in personal projects should be exported. Set this property to <code>True</code> to include commits from forked repositories in personal projects. |
| <code>plugin.data.pipeline.bitbucket.export.build.statuses</code> | |
| <code>false</code> | Specifies whether build statuses should be included in the export. Exporting build statuses can take a significant amount of time if you have a lot of builds.

Set this property to <code>true</code> to export build statuses. |
| <code>plugin.data.pipeline.bitbucket.commit.queue.polling.timeout.seconds</code> | |
| <code>20</code> | Time, in seconds, it takes to receive the first commit from git process.

You should only need to change this if you see a <code>CommitStreamingException</code> (this error is usually caused by another underlying problem). |
| <code>plugin.data.pipeline.bitbucket.commit.git.execution.timeout.seconds</code> | |
| <code>3600</code> | Sets the idle and execution timeout for the git ref-list command. You should only need to change this if you see "an error occurred while executing an external process: process timed out" error. |

Check the status of an export

You can check the status of an export and view when your last export ran from within your applications admin console. To view data export status:

1. Go to  **System**.
2. Select **Data pipeline**

There are a number of export statuses:

- **Not started** - no export is currently running
- **Started** - the export is currently running
- **Completed** - the export has completed
- **Cancellation requested** - a cancellation request has been sent
- **Cancelled** - the export was cancelled
- **Failed** - the export failed.

For help resolving failed or cancelled exports, see [Data pipeline troubleshooting](#).

Output files

Each time you perform a data export, we assign a numerical job ID to the task (starting with 1 for your first ever data export). This job ID is used in the file name, and location of the files containing your exported data.

Location of exported files

Exported data is saved as separate CSV files. The files are saved to the following directory:

- `<shared-home>/data-pipeline/export/<job-id>` if you run Bitbucket in a cluster
- `<local-home>/shared/data-pipeline/export/<job-id>` you are using non-clustered Bitbucket.

Within the `<job-id>` directory you will see the following files:

- `build_statuses_<job_id>_<timestamp>.csv`
- `commits_<job_id>_<timestamp>.csv`
- `pull_requests_<job_id>_<timestamp>.csv`
- `repositories_<job_id>_<timestamp>.csv`
- `users_<job_id>_<timestamp>.csv`

To load and transform the data in these files, you'll need to understand the schema. See [Data pipeline export schema](#).

Set a custom export path

By default, the data pipeline exports the files to the home directory, but you can use the REST API to set a custom export path.

To change the root export path, make a PUT request to `<base-url>/rest/datapipeline/1.0/config/export-path`.

In the body of the request pass the absolute path to your preferred directory, for example:

```
{
  "path": "/tmp/new/path"
}
```

The PUT request returns the following response:

| Code | Sample response |
|------|---|
| 200 | <p>If the path is writable and accepted:</p> <pre>{ "exportPath": "/tmp/new/path/data-pipeline/export", "customPathSet": true }</pre> |

To check the export path, make a GET request to `<base-url>/rest/datapipeline/1.0/config/export-path`.

The GET request returns the following responses.

| Code | Sample response |
|------|-----------------|
|------|-----------------|

| | |
|-----|---|
| 200 | <p>When custom path set:</p> <pre data-bbox="316 197 1174 342"> { "exportPath": "/tmp/example/pipeline", "customPathSet": true } </pre> |
| 200 | <p>When custom path not set, the default shared home path will be returned</p> <pre data-bbox="316 454 1174 600"> { "exportPath": "/shared/home/export/path", "customPathSet": false } </pre> |

Revert to the default export path

To revert to the default path, make a DELETE request to `<base-url>/rest/datapipeline/1.0/config/export-path`.

Sample Spark and Hadoop import configurations

If you have an existing Spark or Hadoop instance, use the following references to configure how to import your data for further transformation:

```

%python
# File location
file_location = "/FileStore/**/export_2020_09_24T03_32_18Z.csv"

# Automatically set data type for columns
infer_schema = "true"
# Skip first row as it's a header
first_row_is_header = "true"
# Ignore multiline within double quotes
multiline_support = "true"

# The applied options are for CSV files. For other file types, these will be ignored. Note escape &
# quote options for RFC-4801 compliant files
df = spark.read.format("csv") \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("multiLine", multiline_support) \
  .option("quote", "\"") \
  .option("escape", "\\") \
  .option("encoding", "UTF-8").load(file_location)

display(df)

```

```
CREATE EXTERNAL TABLE IF NOT EXISTS some_db.datapipeline_export (  
  `repository_id` string,  
  `instance_url` string,  
  `url` string,  
  `repository_name` string,  
  `description` string,  
  `hierarchy_id` string,  
  `origin` string,  
  `project_id` string,  
  `project_key` string,  
  `project_name` string,  
  `project_type` string,  
  `forkable` string,  
  `fork` string,  
  `public` string  
)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  
WITH SERDEPROPERTIES (  
  "escapeChar" = "\\\" ,  
  'quoteChar' = '\"',  
  'separatorChar' = ','  
) LOCATION 's3://my-data-pipeline-bucket/test-exports/'  
TBLPROPERTIES ('has_encrypted_data'='false');
```

Troubleshooting failed exports

Exports can fail for a number of reasons, for example if your search index isn't up to date. For guidance on common failures, and how to resolve them, see [Data pipeline troubleshooting](#) in our knowledge base.

Data pipeline export schema

This page describes the structure and data schema of the Bitbucket data export files.

To learn more about how the set up and configure your data pipeline, see [Data pipeline](#).

Output file format and structure

The output files are written in CSV format and are RFC4180 compliant. They have the following characteristics:

- Each file has a header. This includes files from exports that resulted in no data.
- New lines are separated by CRLF characters `\r\n`.
- Fields containing line breaks (CRLF), double quotes, and commas are enclosed in double quote.
- If double-quotes are present inside fields, then a double-quote appearing inside a field are escaped by preceding it with another double quote. For example: "aaa", "b" "bb", "ccc".
- Fields with no data (null values) are represented in the CSV export by two consecutive delimiters (as in, , ,).
- Embedded break lines are escaped by default and printed as n.

On this page:

- [Output file format and structure](#)
- [Repositories file](#)
- [Commits file](#)
- [Pull requests file](#)
- [Build status file \(optional\)](#)

Users file

| Field | Description |
|---------------|---|
| user_id | Type: String
Description: ID of the user
Example: 2853030 |
| instance_url | Type: URL
Description: Base URL of the current instance.
Example: https://yoursitename.com |
| user_name | Type: String
Description: User name of the user.
Example: jsmith |
| user_fullname | Type: String
Description: Full name of the user.
Example: John Smith |
| user_email | Type: Email
Description: Email address of the user.
Example: jsmith@example.com |

Repositories file

| Field | Description |
|-----------------|---|
| repository_id | <p>Type: Number</p> <p>Description: Unique identifier for the repository.</p> <p>Example: 23</p> |
| instance_url | <p>Type: URL</p> <p>Description: Base URL of the current instance.</p> <p>Example: <code>https://yoursitename.com</code></p> |
| url | <p>Type: String</p> <p>Description: URL of the repository.</p> <p>Example: <code>http://yoursitename.com/projects/SAM/repos/sample-repo</code></p> |
| repository_name | <p>Type: String</p> <p>Description: Name of the repository.</p> <p>Example: <code>Sample repo</code></p> |
| description | <p>Type: String</p> <p>Description: Description of the repository.</p> |
| hierarchy_id | <p>Type: Number</p> <p>Description: The unique identifier for the hierarchy of forks to which this repository belongs. All repositories have a hierarchy ID, even if they have no forks.</p> <p>Example: <code>d536510043f3b4bb4f77</code></p> |
| origin | <p>Type: String</p> <p>Description: Repository from which this repository was forked, if any.</p> <p>Example: <code>SAMPLE/my-fork</code></p> |
| project_id | <p>Type: Number</p> <p>Description: Unique identifier for the project.</p> <p>Example: 19</p> |
| project_key | <p>Type: String</p> <p>Description: Key of the project this repo is contained in.</p> <p>Example: <code>SAM</code></p> |
| project_name | <p>Type: String</p> <p>Description: Name of the project this repo is contained in.</p> <p>Example: <code>Sample Project</code></p> |

| | |
|--------------|---|
| project_type | <p>Type: String</p> <p>Description: Whether the project the repository is contained in is normal or personal.</p> <p>Example: Personal</p> |
| forkable | <p>Type: Boolean</p> <p>Description: Allow forks is selected in the repository settings.</p> <p>Example: True</p> |
| fork | <p>Type: Boolean</p> <p>Description: Repository is forked from another repository.</p> <p>Example: False</p> |
| public | <p>Type: Boolean</p> <p>Description: Whether Public access is enabled in the repository settings.</p> <p>Example: False</p> |

Commits file

| Field | Description |
|----------------|---|
| commit_hash | <p>Type: String</p> <p>Description: Unique identifier of the commit.</p> <p>Example: fc57795ddd877f00e51ab9f51a929220b1a5337b</p> |
| instance_url | <p>Type: URL</p> <p>Description: Base URL of the current instance.</p> <p>Example: https://yoursitename.com</p> |
| url | <p>Type: URL</p> <p>Description: URL of the commit.</p> <p>Example: https://yoursitename.com/projects/SAM/repos/sample-repo/commits/fc57795ddd877f00e51ab9f51a929220b1a5337b</p> |
| commit_message | <p>Type: String</p> <p>Description: Commit message included with the commit.</p> |
| repository_id | <p>Type: Number</p> <p>Description: Unique identifier for the repository.</p> <p>Example: 23</p> |

| | |
|-----------------|---|
| author_email | <p>Type: String</p> <p>Description: Email address of the author of the committed code. If the author is not a Bitbucket user, this email will not map to a user in the users table.</p> <p>Example: jsmith@example.com</p> |
| authored_date | <p>Type: Date</p> <p>Description: Date and time the commit was made.</p> <p>Example: 2021-02-04 00:43</p> |
| committer_email | <p>Type: String</p> <p>Description: Email address of the user who committed the code. If the committer is not a Bitbucket user, this email will not map to a user in the users table.</p> <p>Example: jsmith@example.com</p> |
| committed_date | <p>Type: Date</p> <p>Description: Date and time the commit was last modified.</p> <p>Example: 2021-02-04 00:43</p> |

Pull requests file

| Field | Description |
|-----------------|---|
| pull_request_id | <p>Type: String</p> <p>Description: Unique identifier of the pull request. It is a combination to_repository_id & scoped_id.</p> <p>Example: 23-16</p> |
| scoped_id | <p>Type: Number</p> <p>Description: Identifier for the pull request. This ID is only unique within each repository.</p> <p>Example: 16</p> |
| instance_url | <p>Type: URL</p> <p>Description: Base URL of the current instance.</p> <p>Example: https://yoursitename.com</p> |
| url | <p>Type: String</p> <p>Description: URL of the pull request.</p> <p>Example: http://yoursitename.com/projects/SAM/repos/sample-repo/pull-requests/132/</p> |

| | |
|--------------------|---|
| title | <p>Type: String</p> <p>Description: Title of the pull request.</p> <p>Example: TEST-123 fix broken tests</p> |
| description | <p>Type: String</p> <p>Description: Description included with the pull request, limited to XXX characters.</p> |
| author_id | <p>Type: Number</p> <p>Description: Unique identifier of the person who created the pull request.</p> <p>Example: 2853030</p> |
| created_date | <p>Type: Date</p> <p>Description: Date the pull request was created.</p> <p>Example: 2021-01-03 20:57</p> |
| updated_date | <p>Type: Date</p> <p>Description: Date the pull request was updated.</p> <p>Example: 2021-01-03 23:06</p> |
| closed_date | <p>Type: Date</p> <p>Description: Date the pull request was closed.</p> <p>Example: 2021-01-03 23:06</p> |
| from_branch_name | <p>Type: String</p> <p>Description: The name of the source branch.</p> <p>Example: Improve-tests</p> |
| from_commit_hash | <p>Type: String</p> <p>Description: The commit hash for the earliest commit in this pull request.</p> <p>Example: c149b9a3a37fd211613aaf9390866d1778172a07</p> |
| from_repository_id | <p>Type: Number</p> <p>Description: Unique identifier for the source repository.</p> <p>Example: 16</p> |
| to_branch_name | <p>Type: String</p> <p>Description: The name of the destination branch.</p> <p>Example: Master</p> |
| to_commit_hash | <p>Type: String</p> <p>Description: The commit hash for the latest commit in this pull request.</p> <p>Example: d149b9a3a37fd211613aaf9390866d1778172a56</p> |

| | |
|----------------------|---|
| to_repository_id | <p>Type: Number</p> <p>Description: Unique identifier for the destination repository.</p> <p>Example: 23</p> |
| version | <p>Type: Number</p> <p>Description: Version number represents the number of times the pull request has been updated.</p> <p>Example: 3</p> |
| state | <p>Type: String</p> <p>Description: Current status of the pull request. Can be MERGED, DECLINED, or OPEN.</p> <p>Example: MERGED</p> |
| participant_count | <p>Type: Number</p> <p>Description: Number of participants on the pull request.</p> |
| reviewer_count | <p>Type: Number</p> <p>Description: Number of reviewers on the pull request.</p> |
| approvals_count | <p>Type: Number</p> <p>Description: Number of approvals on the pull request.</p> |
| comments_count | <p>Type: Number</p> <p>Description: Number of comments on the pull request.</p> |
| tasks_count | <p>Type: Number</p> <p>Description: Total number of tasks on the pull request.</p> |
| resolved_tasks_count | <p>Type: Number</p> <p>Description: Number of tasks resolved on the pull request.</p> |
| time_to_merge | <p>Type: Number</p> <p>Description: Time between pull request creation and merge, in milliseconds.</p> <p>Example: 24800</p> |

Build status file (optional)

Exporting build statuses can take a significant amount of time depending on the number of build statuses that might be associated with a commit. For this reason, exporting this data is disabled by default. Set the `plugin.data.pipeline.bitbucket.export.build.statuses` system property to `true` to enable it.

| Field | Description |
|-------|-------------|
|-------|-------------|

| | |
|--------------------|---|
| commit_hash | <p>Type: String</p> <p>Description: Commit hash for the commit.</p> <p>Example: fc57795ddd877f00e51ab9f51a929220b1a5337b</p> |
| repository_id | <p>Type: Number</p> <p>Description: Unique identifier of the repository.</p> <p>Example: 23</p> |
| build_key | <p>Type: String</p> <p>Description: Key set in your CI tool. Can be used as a unique identifier when combined with repository ID and commit ID.</p> <p>Example: SAMPLE-PLAN123</p> |
| build_url | <p>Type: String</p> <p>Description: URL of the build in your CI tool.</p> <p>Example: https://bamboo.example.com/browse/SAMPLE-PLAN123-8</p> |
| build_name | <p>Type: String</p> <p>Description: Name of the build.</p> <p>Example: Pre-release build</p> |
| build_number | <p>Type: Number</p> <p>Description: The identifier for the specific run that resulted in this build status. May be empty.</p> <p>Example: 8</p> |
| build_description | <p>Type: String</p> <p>Description: Description of the specific run that resulted in this build status.</p> <p>Example: #8 successful\nin 7 minutes</p> |
| build_duration | <p>Type: Number</p> <p>Description: Duration of the build, in milliseconds.</p> <p>Example: 230400</p> |
| build_created_date | <p>Type: Date</p> <p>Description: Date the build was created.</p> <p>Example: 2021-01-27 22:55</p> |
| build_updated_date | <p>Type: Date</p> <p>Description: Date the build was updated.</p> <p>Example: 2021-01-27 22:55</p> |

| | |
|------------------------|---|
| build_state | Type: String
Description: State of the build. Can include SUCCESSFUL/FAILED/INPROGRESS
Example: SUCCESSFUL |
| build_parent | Type: String
Description: Key of the parent build for this plan. Can be empty.
Example: PLAN |
| build_ref | Type: String
Description: The branch or tag that this build is associated with. Can be empty. |
| build_successful_tests | Type: Number
Description: The number of successful tests in this build. |
| build_failed_tests | Type: Number
Description: The number of failed tests in this build. |
| build_skipped_tests | Type: Number
Description: The number of skipped tests in this build. |

Integrated CI/CD

With Integrated CI/CD you can create a seamless idea to production workflow. It enables you to link Bitbucket Data Center and Server with Bamboo or Jenkins, and to combine this with a [link to Jira Software](#) to get a centralized view of your CI/CD pipeline.

Configuring, maintaining, and monitoring your pipeline can then be done from Bitbucket, allowing you to see build statuses and test results all within the context of your code.

On this page:

- [Introduction](#)
- [Get started](#)
- [Step-by-step guides](#)



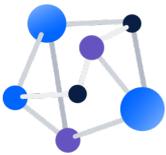
Introduction

- [Integrate your CI/CD pipeline](#)
- [Supported platforms](#)



Get started

- [Link your CI server](#)
- [Configure your CI server](#)
- [View builds information in Bitbucket](#)
- [Perform build actions](#)
- [View build logs](#)
- [Download build artifacts](#)



Step-by-step guides

- [Link Bitbucket with Bamboo](#)
- [Link Bitbucket with Jenkins](#)

Integrate your CI/CD pipeline

[Integrated CI/CD](#) is a comprehensive CI/CD solution. With it, you can integrate Bitbucket Data Center and Server with Bamboo or Jenkins, unlocking a range of benefits.

On this page:

- [Quick and easy set up](#)
- [Simplified pipeline creation and maintenance](#)
- [Centralized contextual feedback](#)
- [Get started](#)

Related pages:

- [Integrated CI/CD](#)
- [Supported platforms](#)

What is CI/CD?

CI/CD is a modern approach to software development that focuses on continuous integration (CI), and continuous development or delivery (CD). [Learn more about CI/CD](#) on the Atlassian CI/CD website.

Quick and easy set up

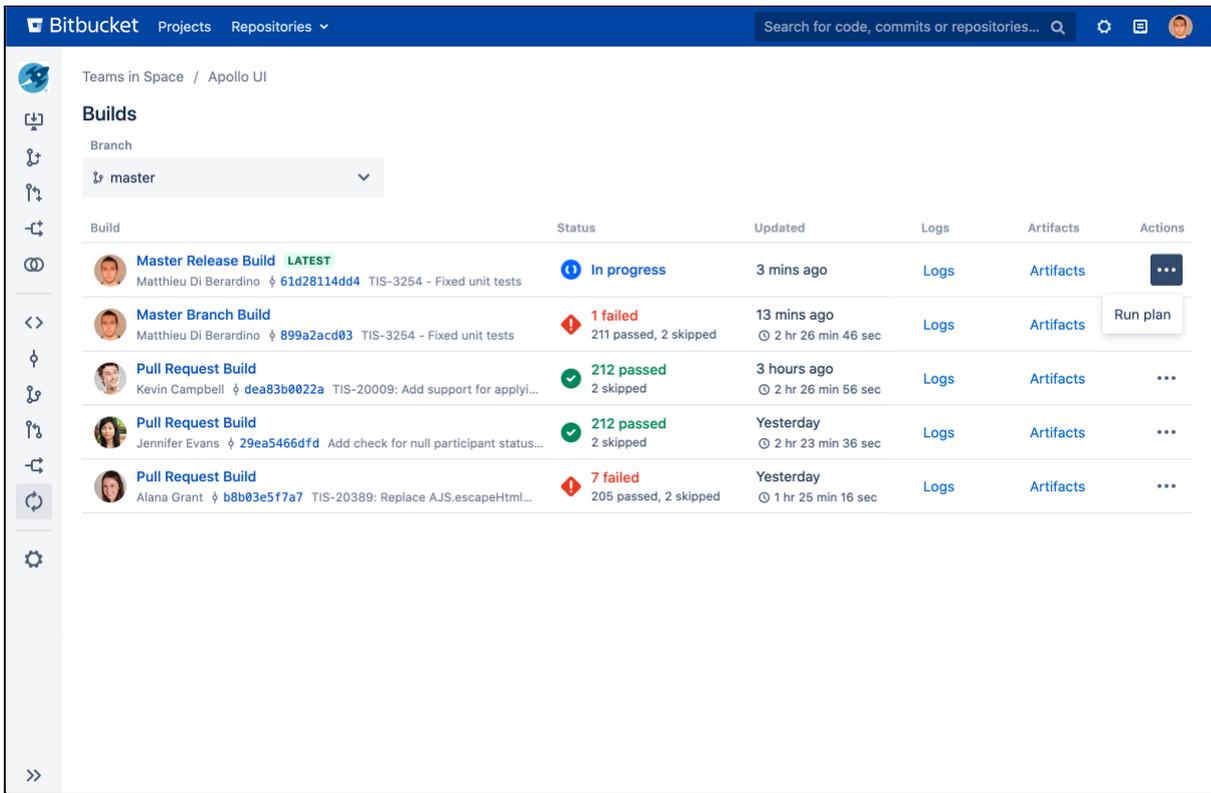
Integrated CI/CD removes the need to use complex third-party or custom integrations to link the applications in your CI/CD pipeline. It comes with the ability to create a seamless link to Bamboo or Jenkins, making it quick and easy to set up. Once this link has been created at a global level, the whole team can then create and maintain their own CI/CD pipelines.

Simplified pipeline creation and maintenance

With its support for [configuration as code](#), Integrated CI/CD makes pipeline creation and maintenance simple. To create a pipeline, you simply create a configuration file, check it into the target repository, and configure Bamboo or Jenkins to detect it. It will then be a versioned artifact, making it easy to keep up-to-date, reducing unreliable builds and incidents.

Centralized contextual feedback

Once you've integrated the applications in your CI/CD pipeline, you'll be able to see feedback where it matters most. [Build statuses and test results](#) will all be pulled into Bitbucket, where you can see them in context. You'll also be able to [act on builds](#) from Bitbucket, reducing the need to switch tools.



Get started

Check out our [Supported Platforms](#) page to see what you need to get started. Then when you're ready, see how to [Link your CI server](#).

Link your CI server

Bitbucket Data Center and Serveris designed to be integrated with Bamboo and Jenkins, and weve streamlined the integration process for both applications.

Integrating can only be done by an administrator of both instances. Once its completed, your team will be able to configure and maintain their own CI/CD pipelines.

On this page:

- [Get maximum benefit from Integrated CI/CD](#)
 - [Integrate using Application Links](#)
 - [Upgrade to a minimum recommended version](#)
- [Integration guides](#)

Related pages:

- [Integrated CI/CD](#)
- [Link Bitbucket with Bamboo](#)
- [Link Bitbucket with Jenkins](#)

Get maximum benefit from Integrated CI/CD

To access the full set of Integrated CI/CD features, you need to:

- Use Application Links when integrating Bitbucket with Bamboo or Jenkins.
- Be on a recommended version [[anchor link to Integration guides table](#)] of all applications and plugins.

In Bitbucket youll then be able to:

- See detailed builds information, like test summaries and durations.
- See links to logs and artifacts for each build.
- Perform actions, such as running builds.

Integrate using Application Links

An important part of integrating Bitbucket with your CI server is connecting them using Application Links, a bundled Atlassian app. Integrating without Application Links is possible, but its not recommended as youll only have access to a limited number of Integrated CI/CD features.

How to set up Application Links is explained in the integration guides linked to below.

Upgrade to a minimum recommended version

Before integrating Bitbucket with Bamboo or Jenkins, we recommend upgrading to a minimum recommended version of these applications, or even better, the latest version. Along with Application Links, this will give you have access to all the features of Integrated CI/CD, and ensure that the environment is as secure and stable as possible.

For more on upgrading, see the [Bamboo upgrade guide](#) or the [Jenkins upgrade guide](#) on Jenkins.io.

Integration guides

| CI server | Minimum recommended version | Integration guides |
|-----------|--|---|
| Bamboo | Bamboo 7.1+ | Bamboo integration |
| Jenkins | Bitbucket Server integration plugin for Jenkins 2.0.0+ | Bitbucket Server integration plugin on Jenkins.io |

If you use a different CI server, you can integrate with Bitbucket and send it detailed builds information using our REST API. For more on how to do this, see our [REST resources](#).

Once youve finishing integrating, see how to [Configure your CI server](#).

Configure your CI server

[Integrated CI/CD](#) allows you to use configuration as code to set up and manage builds and deployments. This approach of storing your configuration as source code has many advantages it enables easier automation, change tracking, validation, and more. For details, check out [the benefits of configuration as code](#).

Before you continue, make sure youve [linked with your CI server](#). Then complete these steps, which well outline in more detail below:

1. [Create a configuration file](#)
2. [Check the configuration file into the target repository](#)
3. [Configure Bamboo or Jenkins](#)

On this page:

- [1. Create a configuration file](#)
- [2. Check the configuration file into the target repository](#)
- [3. Configure Bamboo or Jenkins](#)

Related pages:

- [Bamboo Specs](#)
- [Using a Jenkinsfile](#)

1. Create a configuration file

Integrated CI/CD supports Bamboo and Jenkins, which use their own types of configuration as code files.

| CI server | Configuration file type | Documentation |
|-----------|-------------------------|--|
| Bamboo | Bamboo Specs file | Bamboo Specs documentation
Bamboo Specs reference |
| Jenkins | Jenkinsfile | Jenkinsfile documentation on jenkins.io |

2. Check the configuration file into the target repository

Once youve created your configuration file, check it into theBitbucket repository you want your CI Server to build from. Then you need to configure Bamboo or Jenkins to detect it.

3. Configure Bamboo or Jenkins

If you use Bamboo, you need to [configure Bamboo to scan Bitbucket for Bamboo Specs](#).

If you use Jenkins, you need to [configure a Jenkins pipeline](#) and specify theBitbucket repository to build from.

Then when your build starts, it will appear on the Builds page. Here you can:

- [view builds information](#)
- access [logs](#) and [artifacts](#)
- [perform build actions](#)

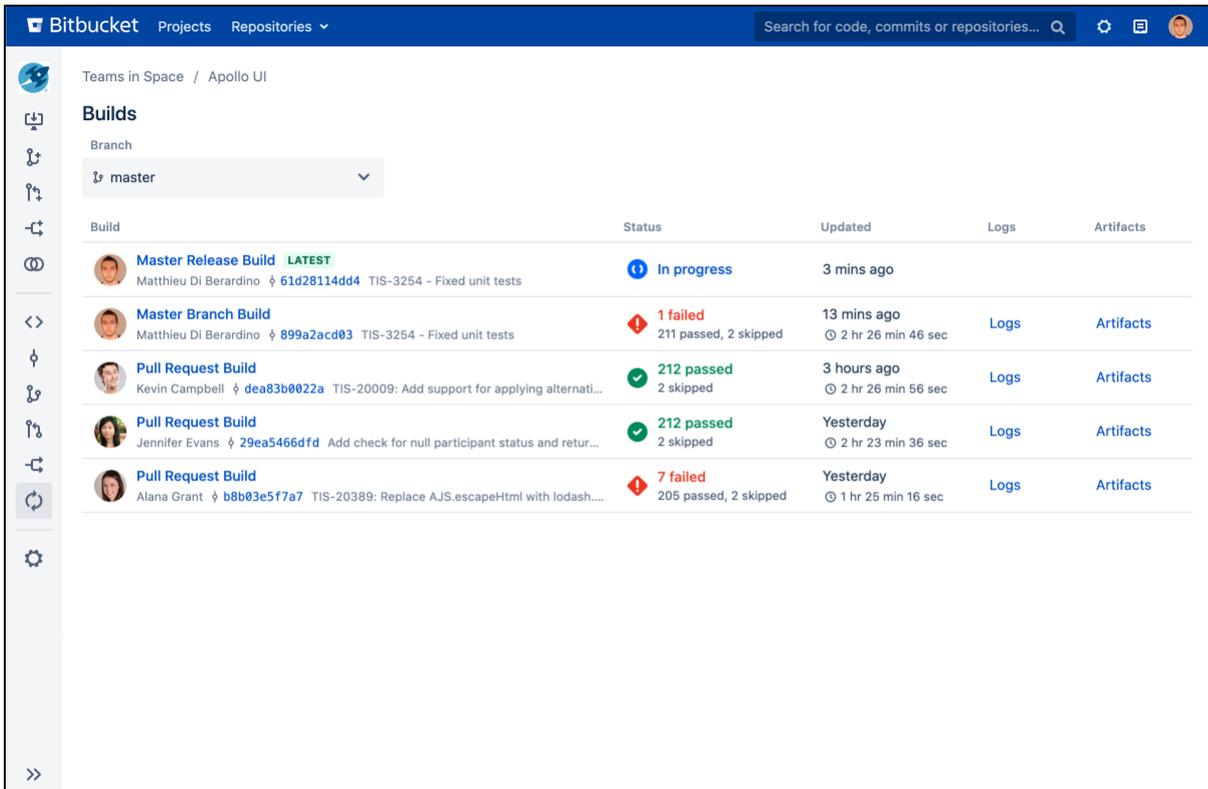
View builds information in Bitbucket

To view builds information in Bitbucket Data Center and Server, you first need to [link it with your CI \(continuous integration\) server](#). Once they're linked, you'll then be able to see build results and additional related information right next to your code where it's most valuable.

On this page:

Related pages:

- [What is Continuous Integration?](#)
- [Bamboo integration](#)
- [Bitbucket Server integration plugin on Jenkins.io](#)



The screenshot shows the Bitbucket interface for viewing build information. The top navigation bar includes 'Bitbucket', 'Projects', and 'Repositories'. A search bar is present for code, commits, or repositories. The main content area is titled 'Builds' and shows a list of build jobs. The 'Branch' is set to 'master'. The build list includes columns for 'Build', 'Status', 'Updated', 'Logs', and 'Artifacts'.

| Build | Status | Updated | Logs | Artifacts |
|--|-----------------------------------|-------------------------------------|----------------------|---------------------------|
| Master Release Build LATEST
Matthieu Di Berardino 61d28114dd4 TIS-3254 - Fixed unit tests | In progress | 3 mins ago | | |
| Master Branch Build
Matthieu Di Berardino 899a2acd03 TIS-3254 - Fixed unit tests | 1 failed
211 passed, 2 skipped | 13 mins ago
⌚ 2 hr 26 min 46 sec | Logs | Artifacts |
| Pull Request Build
Kevin Campbell dea83b0022a TIS-20009: Add support for applying alternati... | 212 passed
2 skipped | 3 hours ago
⌚ 2 hr 26 min 56 sec | Logs | Artifacts |
| Pull Request Build
Jennifer Evans 29ea5466dfd Add check for null participant status and retur... | 212 passed
2 skipped | Yesterday
⌚ 2 hr 23 min 36 sec | Logs | Artifacts |
| Pull Request Build
Alana Grant b8b03e5f7a7 TIS-20389: Replace AJS.escapeHtml with lodash... | 7 failed
205 passed, 2 skipped | Yesterday
⌚ 1 hr 25 min 16 sec | Logs | Artifacts |

Builds information is shown in a number of contextually relevant places in Bitbucket:

- The **Builds** page gives you an overview of all builds across your instance. To find a specific build here you can filter by branch or commit. Once you've found it, you can then see its test results, access direct links to its [logs](#) and [artifacts](#), and [perform build actions](#).
- On the **Pull request** page you can see builds information in two locations: you can see the latest build status in the **Overview** tab, and you can see builds information for each commit in the **Builds** tab. Just like on the Builds page, here you can access log and artifact links, and act on builds.
- On the **Commits** and **Branches** pages you can see the latest build status for each item. For more information, you can also select any build status icon. This will take you to the builds page where the filter will be automatically set to that commit or branch.

Perform build actions

With [Integrated CI/CD](#) you can perform actions on Bamboo or Jenkins builds from Bitbucket Data Center and Server, removing the need to switch tools.

You can act on builds from the Builds page and the pull request Builds tab. To do this, you first need to authorizeBitbucket to perform actions.

On this page:

- [Authorize Bitbucket](#)
- [Perform an action](#)

Related pages:

- [Link your CI server](#)

Authorize Bitbucket

To act on Bamboo or Jenkins builds fromBitbucket you first need to authorize it. Each user only needs to do this once.

To authorize Bitbucket:

1. From the Actions column, selectBitbucket for a build.
2. Select **Authorize actions**.
3. Follow the instructions.

Perform an action

Once youve authorized Bitbucket, you can act on a build by selecting an action from the actions menu. The actions available to you will depend on how your CI server is configured and your level of access.

View build logs

[Integrated CI/CD](#) enables Bitbucket Data Center and Server to display contextual information about a build, such as a link to the builds logs on your CI server. You'll see this link on the Builds page and the pull request builds tab, making it easy to investigate problems when you spot them.

To enable build log links, you need to [Link your CI server](#).

On this page:

Related pages:

- [Integrated CI/CD](#)
- [Link your CI server](#)

Download build artifacts

[Integrated CI/CD](#) enables Bitbucket Data Center and Server to display contextual information about a build, such as a link to the build artifacts on your CI server. You'll see this link on the Builds page and the pull request Builds tab.

To enable build artifacts links, you need to [Link your CI server](#).

On this page:

Related pages:

- [Integrated CI/CD](#)
- [Link your CI server](#)

Link Bitbucket with Bamboo

[Integrated CI/CD](#) enables you to create a link between Bitbucket Data Center and Server and Bamboo, unlocking a range of benefits. Bitbucket can receive build statuses, test results, and other feedback from Bamboo, and display it in context where it matters most. In Bamboo, you can pick a Bitbucket repository and checkout its sources without specifying additional credentials.

Once they're linked, users can then complete the authorization process to [perform Bamboo actions](#) in Bitbucket.

For more on the benefits of linking with Bamboo, see [Bamboo integration](#).

Create an Application Link

Linking with Bamboo is done using Application Links, a bundled Atlassian app.

To create an application link:

1. Go to Bitbucket > **Application Links**.
2. Enter the Bamboo instance URL, then select **Create new link**.
3. Review the **Link applications** dialog and configure these options:
 - If you select **The servers have the same set of users and usernames**, the link will be configured using OAuth (with impersonation) authentication.
 - If you are not an admin on both servers you won't be able to set up a 2-way (reciprocal) application link. If you want to go ahead and create a 1-way link, clear the **I am an administrator on both instances** checkbox.
4. Select **Continue** to go to Bamboo.
5. In Bamboo, select **Continue** to return to Bitbucket. Your Bamboo instance will now be in the list of linked applications, and it will be able to submit [build statuses and test results](#) back to Bitbucket. In addition, Bitbucket will be able to display links to where you can see [logs](#) and download [artifacts](#) on Bamboo for each build.

Having trouble integrating your Atlassian products with Application Links?

We've developed a [guide to troubleshooting application links](#), to help you out. Take a look at it if you need a hand getting around any errors or roadblocks with setting up application links.

Next steps

See how to [configure your CI server](#).

On this page:

- [Create an Application Link](#)
- [Next steps](#)

Related pages:

- [Integrated CI/CD](#)
- [Link your CI server](#)
- [Configure your CI server](#)

Link Bitbucket with Jenkins

[Integrated CI/CD](#) enables you to create a link between Bitbucket Data Center and Server and Jenkins, unlocking a range of benefits. Bitbucket can receive build statuses, test results, and other feedback from Jenkins, and display it in context where it matters most. In Jenkins, you can pick a Bitbucket repository and checkout its sources without specifying additional credentials.

Once they're linked, users can then complete the authorization process to [perform Jenkins actions](#) in Bitbucket.

Create the link

For instructions on linking with Jenkins, see the documentation for the [Bitbucket Server integration plugin for Jenkins](#) on [plugins.jenkins.io](#).

Next steps

See how to [configure your CI server](#).

On this page:

- [Create the link](#)
- [Next steps](#)

Related pages:

- [Integrated CI/CD](#)
- [Link your CI server](#)

Atlassian Bitbucket Integration plugin triggers builds for multiple branches

Issue symptoms:

A push to Bitbucket triggers builds for multiple branches, not only the branch that has received the most recent push.

Explanation:

[Official Atlassian Jenkins integration plugin](#) checks Bitbucket Server for **any** changes in a repository with a configured webhook to Jenkins since the last build was triggered. If a change is found - it builds all the branches that have changed.

Lets consider a situation:

Situation 1:

1. A user pushes a commit to branch **FEAT-1.1.1**.
2. An Auto-merging feature is enabled propagating the changes to **FEAT-1.1** and **Release** branches.

Result:

Jenkins builds branches **FEAT-1.1.1**, **FEAT-1.1**, and **Release**.

Situation 2:

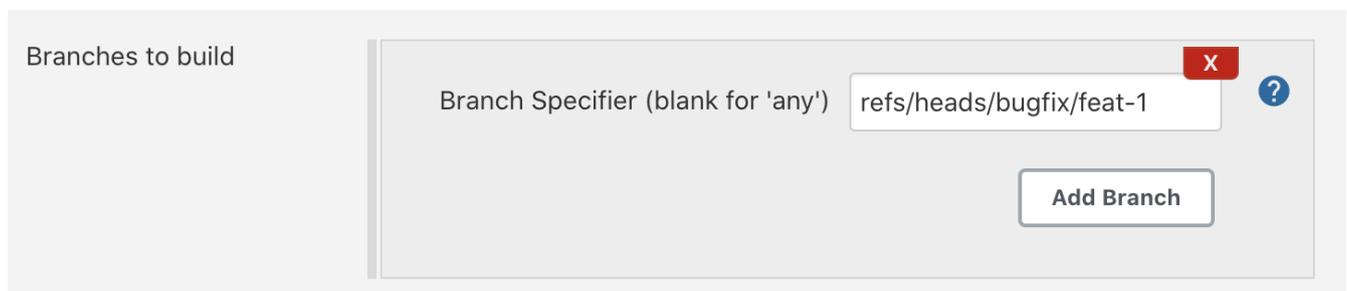
1. Jenkins instance is brought down for maintenance.
2. A user pushes changes to branches **FEAT-1.1**, **FEAT-2.0**, and **BUGFIX-456**.
3. Jenkins instance is brought back up.
4. A user pushes changes to branch **FEAT-2.0**.

Result:

Branches **FEAT-1.1**, **BUGFIX-456**, and **FEAT-2.0** (the most recent commit) will be built in Jenkins.

Once the push to **FEAT-2.0** happens and communication between Bitbucket Server and Jenkins is up and running again, the Jenkins plugin will scan the whole repository in Bitbucket Server for changes since the last scan. It should discover updates to not only the most recently updated branch but also all the others not built yet.

If your intention is to build a specific branch, feel free to use **Branch Specifier** in '**Branches to build**' in Jenkins build configuration:



The screenshot shows the 'Branches to build' configuration in Jenkins. On the left, the text 'Branches to build' is visible. The main configuration area contains a text input field labeled 'Branch Specifier (blank for 'any')' with the value 'refs/heads/bugfix/feat-1'. To the right of the input field is a red 'X' icon and a blue question mark icon. Below the input field is a button labeled 'Add Branch'.

Adding an additional logger for troubleshooting the Atlassian Bitbucket Integration plugin

If you encounter an issue with our official [Atlassian Bitbucket Integration plugin](#) and can not find any errors in the Bitbucket logs, you might need to add an additional logger in Jenkins, to be able to fully troubleshoot the problem.

To add it, please:

1. Go to Manage Jenkins -> System Log -> Log Recorders.
2. Click Add new log recorder
3. Configure it for **com.atlassian** and save:

Configure log recorder ?

Enter the name of the logger you want to enable.

Loggers that record messages at FINE, FINER, or even FINEST can cause performance issues when many messages are being logged. Additionally, more verbose log levels increase the likelihood of credentials being logged. It is recommended that you delete such loggers, or update them to a less verbose level, when you no longer need them.

Name ?

Loggers **Logger** **Log level** ?

List of loggers and the log levels to record

4.

⚠ WARNING: Setting the log level to **ALL** might produce very verbose output causing Jenkins performance issues. Please set up the level accordingly. **⚠**

5. Push a new commit to a repo in Bitbucket that we'd expect to trigger a build in Jenkins.
6. Go to Log records in your new logger, and check for any ERROR messages.

A sample output for a successfully configured plugin:

```
Jan 06, 2021 7:33:30 PM FINE com.atlassian.bitbucket.jenkins.internal.http.HttpRequestExecutorImpl
Bitbucket - call successful

Jan 06, 2021 7:33:30 PM INFO com.atlassian.bitbucket.jenkins.internal.trigger.register.
BitbucketWebhookHandler lambda$findSame$5
Found an existing webhook - {"id":6,"name":"ab88e816a6f116eb52532ef921bfc7deedb48e8b","url":"http://\macos:
8080/bitbucket-server-webhook/trigger","events":["repo:refs_changed"],"active":true}

Jan 06, 2021 7:33:30 PM INFO com.atlassian.bitbucket.jenkins.internal.trigger.
BitbucketWebhookTriggerImpl$BitbucketWebhookTriggerDescriptor registerWebhook
Webhook returned -{"id":6,"name":"ab88e816a6f116eb52532ef921bfc7deedb48e8b","url":"http://\macos:8080\
/bitbucket-server-webhook/trigger","events":["repo:refs_changed"],"active":true}

Jan 06, 2021 7:33:30 PM FINE com.atlassian.bitbucket.jenkins.internal.http.HttpRequestExecutorImpl
Bitbucket - call successful
```

If you're still not able to spot any issue or resolve it, please contact [support](#).

Install or upgrade Bitbucket

About the installation and upgrade guides

The install and upgrade guides have instructions for installing, setting up, or upgrading Bitbucket Data Center and Server.

Information on the features and changes in specific Bitbucket releases can be found in the [Release Notes](#).

For information about using and administering Bitbucket refer to the [Bitbucket documentation](#).

- [Supported platforms](#)
 - [End of support announcements](#)
- [Installing and upgrading Git](#)
- [Bitbucket installation guide](#)
 - [Install a Bitbucket trial](#)
 - [Install Bitbucket Server on Windows](#)
 - [Install Bitbucket Server on Windows from a zip file](#)
 - [Install Bitbucket Server on Linux](#)
 - [Install Bitbucket Server on Linux from an archive file](#)
 - [Running Bitbucket Server with a dedicated user](#)
 - [Running Bitbucket Server as a Windows service](#)
 - [Run Bitbucket as a Linux service](#)
 - [Automated setup for Bitbucket](#)
 - [Start and stop Bitbucket](#)
 - [Install Bitbucket Server from an archive file](#)
 - [Run the Bitbucket installer](#)
- [Bitbucket Server upgrade guide](#)
 - [How to update your add-on](#)
 - [Upgrade Bitbucket from an archive file](#)
 - [Migrate server.xml customizations to bitbucket.properties](#)
 - [Bitbucket Data Center upgrade guide](#)
- [Use Bitbucket in the enterprise](#)
- [Upgrade Bitbucket without downtime](#)
 - [Upgrade a Bitbucket cluster manually without downtime](#)
 - [Upgrade a Bitbucket cluster on AWS without downtime](#)
 - [Upgrade a Bitbucket cluster through the API without downtime](#)

Downloads



[Download Bitbucket documentation in PDF format.](#)

Other resources

[Bitbucket release notes](#)

[Bitbucket Server knowledge base](#)

[Atlassian Answers \(topic: bitbucket-server\)](#)

Supported platforms

This page lists the supported platforms for Bitbucket Data Center and Server 7.14.x.

See [End of support announcements](#) for upcoming changes to platforms supported by Bitbucket.

Please read the supplied information carefully and check if it applies to your instance.

Definitions

✔ Supported - you can use **Bitbucket Data Center and Server 7.14.x** with this platform.

ⓘ Limited - you can evaluate Bitbucket on this platform, but you can't use it to run a production site.

⚠ Deprecated - support for this platform will end in an upcoming release.

On this page:

- [Hardware](#)
- [Environments](#)
- [Cloud Platforms](#)
- [Java](#)
- [Oracle Java & OpenJDK](#)
- [Databases](#)
- [Integrations](#)
- [DVCS](#)
- [Internet protocols](#)
- [Additional tools](#)

Hardware

CPU

Evaluation: 1 core
Production: 2+ cores

Memory:

3GB+

Good to know

- You'll need at least 3GB available memory. We recommend 1GB for Bitbucket and an additional 2GB to support Git operations.
- Your specific hardware requirements will depend on the number and frequency of Git operations and the number of active users. See [Scaling Bitbucket Server](#) for more information.

Environments

✔ Linux

✔ Microsoft Windows (up to 500 users)

ⓘ Apple macOS

Known issues

- **Microsoft Windows is not supported for Bitbucket Data Center, or for 500+ user tiers.**
- You should not use Windows Resilient File System (ReFS) as NTFS is required.
- Apple macOS is evaluation only. MacOS cannot be used for production deployment.

Good to know

- In production environments Bitbucket should be [run from a dedicated user account](#).

Cloud Platforms

✔ Amazon Web Services (AWS)

✔ Microsoft Azure

Known issues

You can't use [Amazon Elastic File System \(EFS\)](#) for Bitbucket's shared home directory.

Good to know

See [Recommendations for running Bitbucket in AWS](#) for more information.

Java

Oracle Java & OpenJDK

Good to know

- ✔ Java 11.0.8+
 - ⚠ Java 11 through 11.0.7
 - ✘ Java 10
 - ✘ Java 9
 - ✔ Java 8 (supported from 1.8 u65+)
- Bitbucket 8.0 will raise the minimum supported Java version to 11.0.8. Java versions 11.0.0 - 11.0.7 are not recommended due to Java bug: [JDK-8241054](#)
 - Bitbucket will install a supported version of AdoptOpenJDK Java JRE that is only available to Bitbucket if necessary. See [Running the Bitbucket Server installer](#).
 - If you choose to pre-install a JRE, we recommend using AdoptOpenJDK's Java 8 JRE, which you can download from the [AdoptOpenJDK website](#).
 - For OpenJDK, download and install instructions for Linux flavors at <http://openjdk.java.net/install/>.
 - Pre-installed Java on some AWS EC2 Linux instances might be installed with a subset of features. See [SSH server fails to start on AWS EC2 instance for more information](#).
 - We support standard OpenJDK builds, including Oracle's official build, alternative builds such as AdoptOpenJDK, and builds shipped with Linux distributions such as Red Hat Enterprise.

Databases

PostgreSQL

✔ 9.6 - 13

⚠ 9.5

⚠ 9.4

MySQL (Bitbucket Server only)

✔ 8.0.11+

✔ 5.7.9+

⚠ 5.6.16+

⚠ 5.5.8+

✔ MariaDB 10.4.6+

✔ MariaDB 10.3.7+

✔ MariaDB 10.2.6+

⚠ MariaDB 10.1.8+

✘ MariaDB 10.0.10

⚠ MariaDB 5.5.23+

Microsoft SQL Server / Microsoft SQL Server Express

✔ 2019

✔ 2017

✔ 2012 - 2016

Known Issues

[Connect Bitbucket to PostgreSQL](#)

Known Issues

MySQL, while supported by Bitbucket, is currently **not** recommended, especially for larger instances, due to inherent performance and deadlock issues that occur in this database engine under heavy load.

See [Connecting Bitbucket Server to MySQL](#) for more details and <http://dev.mysql.com/doc/refman/5.0/en/innodb-deadlocks.html> for some general information on deadlock issues.

You should not use Bitbucket with the following:

- MySQL 5.6.15 and earlier: due to bugs in its query optimizer ([#68424](#), [#69005](#)). See [Connecting Bitbucket Server to MySQL](#) for more information.
- MySQL 5.7.0 - 5.7.8

Do not use MySQL in Bitbucket Data Center.

Known Issues

- [Connect Bitbucket to SQL Server](#)
- **Named Instances:** If you have a named instance on your server, it is not possible to migrate from the internal database to a named instance of SQL Server using the UI procedure. You will need to manually edit the `bitbucket.properties` file as described on the [Connecting to named instances in SQL Server from Bitbucket Server Knowledge Base](#) article.

Good to know

Using [Windows Authentication between a Linux Bitbucket installation and SQL Server](#) is not supported.

Oracle

✓ 19c

✓ 18c

✓ 12c

⚠ 11g

Amazon Aurora(Data Center only)

✓ PostgreSQL 9.6 - 11

Good to know

- The only supported Amazon Aurora config is a PostgreSQL-compatible clustered database with one writer replicating to zero or more readers.[Learn more](#)

H2 (bundled)

ⓘ Bitbucket Server, evaluation only

✓ Bitbucket Mirror

Good to know

- H2 is bundled with Bitbucket for evaluation use only.
- H2 can be used with Bitbucket Data Center mirrors in production.

HSQLDB (bundled)

Bitbucket Server only, evaluation only

⚠ Deprecated

Good to know

- Please see [Connect Bitbucket to an external database](#).
- HSQLDB is not supported in Bitbucket Data Center.
- HSQLDB support was deprecated as of Bitbucket 4.0+. New Bitbucket installs will bundle and use H2 as the default database for evaluation purposes.

Integrations

See [Integrate with Atlassian applications](#) for supported version combinations.

CI server

We recommend upgrading to the latest version of your CI server to ensure that the environment is as secure and stable as possible. See [Integrated CI/CD](#) for more information about linking Bitbucket and your CI application.

Bamboo

✓ 5.6 6.10

✓ 7.0+ (For [Integrated CI/CD](#) features)**Good to know**

- Bamboo 7.0+ is required to enable [Integrated CI/CD](#) features. To access all Integrated CI/CD features you need to [set up an Application Link](#).
- Using Bamboo 5.6 to 6.10 is possible, but its integration with Bitbucket will be minimal.
- Before upgrading, see the [Bamboo upgrade guide](#).

Jenkins

✓ 2.162+

Good to know

- Integrating requires the [Bitbucket Server Integration plugin](#) for Jenkins.

- To access the full set of [Integrated CI/CD](#) features you need to [set up an Application Link](#).
- Using an earlier version of Jenkins is possible, but its integration with Bitbucket will be minimal.
- Before upgrading see the [Jenkins upgrade guide](#) on Jenkins.io.

Browsers

Desktop browsers

- ✓ Chrome (latest stable version)
- ✓ Firefox (latest stable version)
- ✓ MS Edge(latest stable version)
- ✓ Safari (latest stable version)

Good to know

- Bitbucket has a minimum supported browser resolution of 1100px.
- Bitbucket supports the latest stable version of [Chromium-based Microsoft Edge](#) and does not support legacy Microsoft Edge (versions 18 and lower).
- Mobile browsers are not supported.

DVCS

Git server

- ✓ 2.32.x
- ✓ 2.31.x
- ✓ 2.30.x
- ✓ 2.29.x
- ✓ 2.28.x
- ✓ 2.27.x
- ✓ 2.26.x
- ✓ 2.25.x
- ✓ 2.24.x
- ✓ 2.23.x
- ✓ 2.22.x
- ✓ 2.21.x
- ✓ 2.20.x
- ✓ 2.19.x
- ✓ 2.18.x
- ✓ 2.17.x
- ✓ 2.16.x
- ✓ 2.15.x

Git server

We recommend using the most recent supported version of Git on both the Bitbucket instance and clients where possible, subject to the following notes and exceptions:

- The version of Git installed on machines that interact with Bitbucket must be compatible with the version of Git installed for use by the Bitbucket instance.
- When using smart mirroring, you must have a minimum version of Git 2.11.1 installed.
- Do not use Cygwin Git on Windows servers, regardless of version.
- Git for Windows 2.12.2 [has a regression in git status](#) which causes editing files to fail. The regression has been fixed, and the fix was included in Git for Windows 2.12.3.
- [BSERV-9673](#)- File editing fails using Git for Windows 2.12.2

✔ 2.14.x

✔ 2.13.x

✔ 2.12.x (except 2.12.2 on Windows)*

✔ 2.11.x

Git client

✔ 1.8.4.3+

✔ 1.6.6+

Or where Git LFS is used
1.8.2

Git LFS - client

✔ 1.1.0+

Internet protocols

✔ IPv4

✔ IPv6

Additional tools

Perl

✔ 5.8.8+

Elasticsearch

✔ 7.9.3

✔ 7.5.2

✔ 6.8.6

✔ 6.6.1

✔ 6.5.3

⚠ 5.5.1 - 5.5.3

Git client

A bug was fixed in Git version 1.8.4.3 that prevented http push proxying from working.



[Security vulnerability CVE-2016-2324 & CVE-2016-2315] affects multiple Git versions. Both *server* and *client* Git installations should be updated to a patched maintenance version: 2.4.11, 2.5.5, 2.6.6 or 2.7.4 or newer. For instructions see [Installing and upgrading Git](#).

Good to know:

When using Bitbucket in IPv6 environments, **we recommend that hostnames rather than IP addresses are used**.

Good to know

- Perl is usually provided automatically with Git.
- Elasticsearch 6.5.3, 6.6.1, 6.8.6, 7.5.2, and 7.9.3 are supported with the Buckler plugin. Learn [how to secure Elasticsearch with Buckler](#).
- Elasticsearch 6.4 is supported with AWS.

End of support announcements

End of support matrix

The table below summarizes the end of support announcements for recent Bitbucket Data Center and Server (previously known as Stash) releases:

| Platform/functionality | Announcement date | Bitbucket end of support |
|---|-------------------|--------------------------|
| Deprecation of Postgres 9.5 | 2 February 2021 | From Bitbucket 8.0 |
| Deprecation of MySQL 5.6 | 2 February 2021 | From Bitbucket 8.0 |
| Deprecation of Oracle 11g | 2 February 2021 | From Bitbucket 8.0 |
| Deprecation of MySQL 5.5 | 5 March 2020 | From Bitbucket 8.0 |
| Deprecation of Elasticsearch 5.5x | 5 March 2020 | From Bitbucket 8.0 |
| Deprecation of MariaDB 5.5 | 5 March 2020 | From Bitbucket 8.0 |
| Deprecation of MariaDB 10.0 | 5 March 2020 | From Bitbucket 8.0 |
| Deprecation of Internet Explorer 11 | 24 September 2019 | From Bitbucket 7.0 |
| Deprecation of Git 2.10 | 7 August 2018 | From Bitbucket 6.0 |
| Deprecation of Elasticsearch 2.3 | 11 January 2018 | From Bitbucket 6.0 |
| Deprecation of PostgreSQL 9.2.x | 11 January 2018 | From Bitbucket 5.7 |
| Deprecation of database support | 4 October 2016 | From Bitbucket 5.0 |
| Deprecation of Internet Explorer 10 | 5 May 2015 | From Bitbucket 4.0 |
| Deprecation of Internet Explorer 9 | 13 January 2015 | From Stash 3.10 |
| Deprecation of Java 7 | 25 November 2014 | From Bitbucket 4.0 |
| Deprecation of Git versions earlier than v1.8 | 25 November 2014 | From Bitbucket 4.0 |
| Deprecation of Tomcat 7 | 10 September 2014 | From Bitbucket 4.0 |

Why is Atlassian ending support for these platforms?

Atlassian is committed to delivering improvements and bug fixes as fast as possible. We are also committed to providing world class support for all the platforms our customers run our software on. However, as new versions of databases, web browsers etc. are released, the cost of supporting multiple platforms grows exponentially, making it harder to provide the level of support our customers have come to expect from us. Therefore, we no longer support platform versions marked as end of life by the vendor, or very old versions that are no longer widely used.

End of support announcements on this page (most recent announcements first):

- [Deprecated browsers for Bitbucket \(announced 24 September 2019\)](#)
- [Deprecation of Git 2.10 and earlier \(announced 7 August 2018\)](#)
- [Deprecation of Elasticsearch 2.3 in Bitbucket 6.0 for Data Center customers \(announced 11 January 2018\)](#)
- [Deprecation of PostgreSQL 9.2.x and earlier \(announced 11 January 2018\)](#)
- [Deprecation of database support \(announced 4 October 2016\)](#)
- [Deprecation of Internet Explorer 10 \(announced 5 May 2015\)](#)
- [Deprecation of Internet Explorer 9 \(announced 13 January 2015\)](#)
- [Deprecation of Java 7 \(announced 25 November 2014\)](#)

- [Deprecation of Git versions earlier than v1.8 \(announced 25 November 2014\)](#)
- [Deprecation of Tomcat 7 \(announced 10 September 2014\)](#)
- [Deprecation of Internet Explorer 8 \(announced 22 July 2013\)](#)
- [Deprecation of Java 6 \(announced 9 May 2013\)](#)

Deprecated browsers for Bitbucket (announced 24 September 2019)

In 2015 Microsoft released Edge as the browser to supersede Internet Explorer, and in recent times [Microsoft has discouraged the use of Internet Explorer as a default browser](#). To allow us to continue to take advantage of modern web standards that allow us to deliver improved functionality and the best possible user experience across all of our products, we have decided to end support for Internet Explorer 11.

End of support means we will not fix bugs specific to Internet Explorer 11, and will begin to introduce features that aren't compatible with this browser.

When is this happening?

- The last Bitbucket version to support Internet Explorer will be confirmed soon.
- Subsequent versions will not support Internet Explorer 11.

What this means for you

We recommend switching to one of our [supported browsers](#), such as Microsoft Edge, Google Chrome, or Mozilla Firefox.

If you have questions or concerns regarding this announcement, please email eol-announcement@atlassian.com.

Deprecation of Git 2.10 and earlier (announced 7 August 2018)

In Bitbucket 6.0, we will end support for all versions before Git 2.11:

- Versions before Bitbucket 6.0 support Git 2.2.0 and higher, **excluding Git 2.12.2 on Windows** (see [supported platforms](#) for details).
- After 6.0 we will require Git 2.11.0 or newer, **excluding Git 2.12.2 on Windows**.

Deprecation of Elasticsearch 2.3 in Bitbucket 6.0 for Data Center customers (announced 11 January 2018)

Support for Elasticsearch 2.3 for Data Center customers will be ended with version 6.0, with full details noted on the [Bitbucket 5.7 release notes](#). See [Supported platforms](#) for a current list of supported databases.

Deprecation of PostgreSQL 9.2.x and earlier (announced 11 January 2018)

PostgreSQL 9.2.x has been deprecated from version 5.7 of Bitbucket. Support for this is due to end with 6.0. See [Supported platforms](#) for a current list of supported databases.

Deprecation of database support (announced 4 October 2016)

In version 5.0, Bitbucket will no longer support the following databases.

- MariaDB 5.0.x, 5.1.x
- MySQL 5.0.x, 5.1.x
- Oracle 11
- PostgreSQL 9.1.x and earlier
- SQL Server 2008 and 2008 R2

Bitbucket 5.0 is expected to be released around early-2017. See [Supported platforms](#) for a current list of supported databases.

Deprecation of Internet Explorer 10 (announced 5 May 2015)

During Q4 2015, Bitbucket will no longer support Internet Explorer 10, and will only support Internet Explorer 11 and above. See [Supported platforms](#).

Deprecation of Internet Explorer 9 (announced 13 January 2015)

In version 3.10, Stash will no longer support Internet Explorer 9, and will only support Internet Explorer 10 and above. Stash 3.10 is expected to be released around mid-2015. See [Supported platforms](#).

Deprecation of Java 7 (announced 25 November 2014)

In version 4.0, Bitbucket will no longer support Java 7, and will only support Java 8 and above. Bitbucket 4.0 is expected to be released around mid-2015. See [Supported platforms](#).

Deprecation of Git versions earlier than v1.8 (announced 25 November 2014)

In version 4.0, Bitbucket will only support Git 1.8, and later versions (with the exceptions noted in [Supported platforms](#)), on the server. Bitbucket 4.0 is expected to be released around mid-2015.

Deprecation of Tomcat 7 (announced 10 September 2014)

In version 4.0, Bitbucket will no longer support Tomcat 7, and will only support Tomcat 8 and above. Bitbucket 4.0 is expected to be released around mid-2015. See [Supported platforms](#).

Deprecation of Internet Explorer 8 (announced 22 July 2013)

In version 3.0, Stash will no longer support Internet Explorer 8, and will only support Internet Explorer 9 and above. Stash 3.0 is expected to be released mid-2014. See [Supported platforms](#).

Deprecation of Java 6 (announced 9 May 2013)

In version 3.0, Stash will no longer support Java 6.0, and will only support Java 7.0 and above. Stash 3.0 is expected to be released mid-2013. See [Supported platforms](#).

Installing and upgrading Git



[Security vulnerability CVE-2016-2324 & CVE-2016-2315] If you are running a Git version older than 2.4.11, 2.5.5, 2.6.6 or 2.7.4 (all released 17 Mar 2016), you should upgrade Git as soon as possible.

This page describes how to:

- [Check your version of Git](#)
- [Install or upgrade Git on Linux](#)
- [Install or upgrade Git on macOS](#)
- [Install or upgrade Git on Windows](#)

The information on this page applies to installing or upgrading Git on either your local machine, or on the Bitbucket Data Center and Server instance.

Check your version of Git

The versions of Git supported by Bitbucket are listed on [Supported platforms](#).

You can check your current version of Git by running the `git --version` command in a terminal (Linux, macOS) or command prompt (Windows).

For example:

```
git --version
git version 2.7.4
```

If you don't see a supported version of Git, you'll need to either upgrade Git or perform a fresh install, as described below.

Install or upgrade Git on Linux

Use your package manager to install Git. For example, on Ubuntu 13.10, use:

```
sudo apt-get install git
```

Alternative download options are:

- Download the latest stable Git release from the [Git website](#).
- If you are using a different Linux distribution, you may need to use a different package repository to get the latest stable version of Git.
- If you need the most recent version of Git, you might need [to install it from source](#).

Now [check the Git version](#) you should see the new version of Git.

If you still can't see the expected Git version, you may need to add the Git install location to your path. Open your `~/.profile` file in a text editor and add this line, where `<path/to/git>` is the install location for Git:

```
export PATH=$PATH:<path/to/git>
```

You can use the `which git` command to find the install location for Git.

Install or upgrade Git on macOS

Download the latest stable Git release from the [Git website](#).

Click on the downloaded .dmg file, then double-click the .pkg icon to run the installer. This will install the new version of Git over the existing version:



Alternatively, you can:

- Use the native Git bundled with macOS.
- Use [Homebrew](#) to download and install Git.

Now [check the Git version](#) you should see the new version of Git.

If you still can't see the Git version, you may need to add the Git install location to your path. Open your `~/.profile` file in a text editor and add this line, where `<path/to/git>` is the install location for Git:

```
export PATH=$PATH:<path/to/git>
```

You can use the `which git` command to find the install location for Git.

Install or upgrade Git on Windows

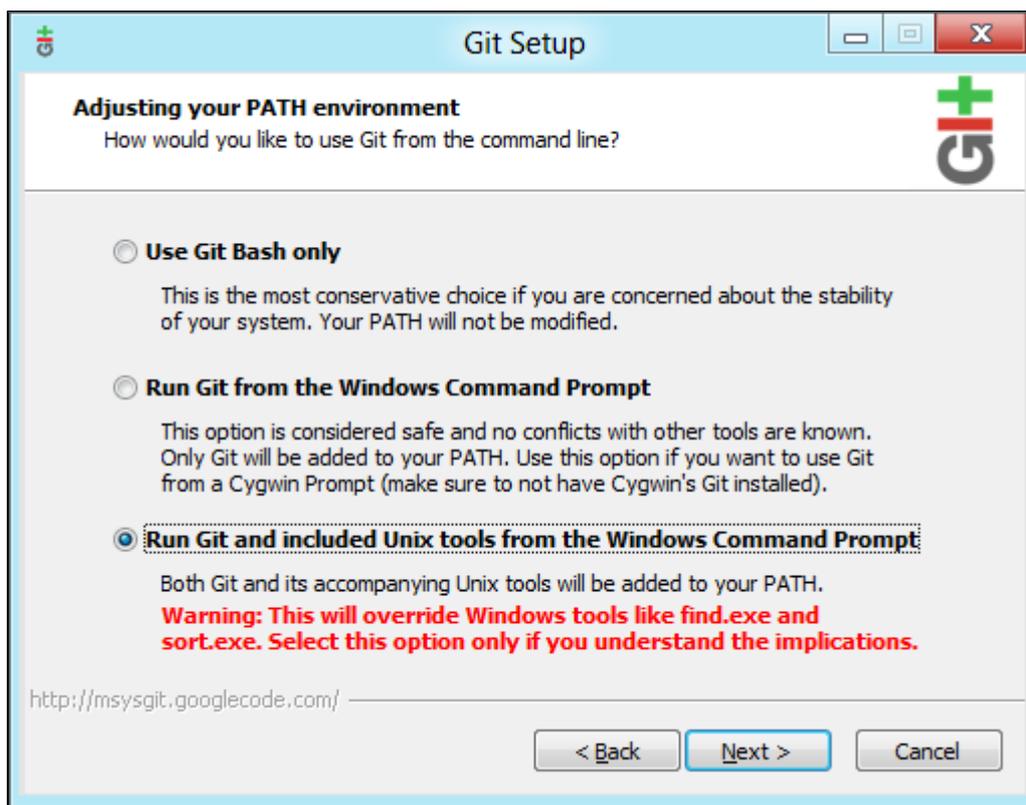
Download the latest stable Git release from the [Git website](#).

Run the Git installer, ensuring that you install into the same location as any existing Git installation. You can use `where git` to locate existing installations. Installing Git for Windows (msysGit) also installs a supported version of Perl.

To ensure that `git.exe` is available in the path, choose either:

- **Run Git from the Windows Command Prompt, or**
- **Run Git and included Unix tools from the Windows Command Prompt.**

Do **not** select **Use Git Bash only** when installing or upgrading Git for the Bitbucket instance-- *this will not work*.



Now, [check the Git version](#) you should see the new version of Git.

⚠ msysGit is the *only supported distribution* when running Bitbucket on Windows. Cygwin Git is *not supported* and has known issues.

If you have successfully installed msysGit but you receive the error "Unable to find git!" when installing Bitbucket, you should abort the installation, restart the Windows server, then restart the Bitbucket installation.

Restart Bitbucket if necessary

If you've been installing or upgrading Git for the Bitbucket instance, rather than for your local machine, you'll need to stop and restart Bitbucket so that it will pick up the upgraded version of Git. See [Start and stop Bitbucket](#) for details.

Bitbucket installation guide

Before you start

Before installing Bitbucket Data Center and Server, check that you meet the minimum system requirements by reading the page [Supported platforms](#).

Choose your installation method

There are a number of ways to install Bitbucket. Choose the method that is best for your environment.

| Install method | Is this right for you? |
|---|---|
| Install a Bitbucket trial <ul style="list-style-type: none">• Windows, Linux or macOS | This is the fastest way to get Bitbucket up and running. If you're evaluating Bitbucket, use this option. You don't need an external database to install a Bitbucket trial. |
| Install Bitbucket Server using an installer <ul style="list-style-type: none">• Windows• Linux | This option uses an installer and is the most straightforward way to get your production site up and running on a Windows or Linux server. |
| Install Bitbucket Server from a zip or archive file <ul style="list-style-type: none">• Windows• Linux | This option requires you to manually install files and configure some system properties. It gives you the most control over the install process. Use this option if there isn't an installer for your operating system. |
| Run Bitbucket in a Docker container <ul style="list-style-type: none">• Bitbucket and Docker | <p>This option gets Bitbucket up and running in no time using a preconfigured Docker image.</p> <p>Atlassian supports running Bitbucket in a Docker container, but we cannot offer support for problems that are related to the environment itself.</p> |
| Run Bitbucket in AWS <ul style="list-style-type: none">• Bitbucket and AWS | Running Bitbucket on Amazon Web Services (AWS) gives you scalable computing capacity without the need to invest in hardware up front while retaining control over where and how your code is hosted within your organization. |
| Install Bitbucket Server on a single node <p>It's the same as a Server installation:</p> <ul style="list-style-type: none">• Windows• Linux | <p>You don't need high availability or disaster recovery but could use features that are exclusive to Data Center. Deploying Data Center on a single node is the same as installing Bitbucket Server, just with a different license, which unlocks new features.</p> <p>Learn more at Running Bitbucket Data Center on a single node.</p> |
| Install Bitbucket in a cluster <ul style="list-style-type: none">• Install Bitbucket Data Center | <p>Jira Data Center with a cluster of nodes is designed for enterprises with large or mission-critical deployments that require continuous uptime, instant scalability, and performance under high load. It can be hosted on your own infrastructure or deployed to AWS or Azure.</p> <p>Learn more at Clustering with Bitbucket</p> |

Note:we do not support installing Bitbucket as a production system on macOS. A macOS download is available for the purposes of evaluating Bitbucket only. There are no limitations to using Bitbucket on a Mac with any one of the [supported browsers](#).

Install a Bitbucket trial

Want to get up and running with Bitbucket quickly? This page will guide you through the steps to install and set up a trial instance of Bitbucket Data Center.



On this page

- Before you begin
1. Download the installer
 2. Install Bitbucket
 3. Set up Bitbucket

If you're ready to set up a production instance of Bitbucket, or you want more control, check out our [other installation guides](#).

Before you begin

Our installers come with all the bits and pieces you need to run the application, but there's a few things you'll need to get up and running:

- A computer or laptop with a supported operating system - you'll be installing Bitbucket so you'll need admin rights.
You can install Bitbucket on a Windows, Linux, or Mac operating system. However Mac isn't supported for production instances.
- A supported web browser - you'll need this to access Bitbucket, we support the latest versions of Chrome and Mozilla Firefox, Internet Explorer 11, and Microsoft Edge.
- A valid email address - you'll need this to generate your evaluation license and create an account. Register with your business email domain to access live chat and phone calls with Bitbucket product specialists (for a limited time).

Ready to get going? Let's start with grabbing the installer.

1. Download the installer

Head to www.atlassian.com/software/bitbucket/download and download the installer for your operating system.

2. Install Bitbucket

The installer allows you to choose the installation and home directories. For this guide we recommend using the default options.

1. Run the installer - we recommend running with an administrator account.
2. Choose **Install a standard instance**, then click **Next**.
3. When prompted, choose to start and launch Bitbucket in a browser.
4. Bitbucket will open in your default browser, and you're ready to start the setup wizard. The application will be available at `http://localhost:7990`.

1. Change to the directory where you downloaded Bitbucket then execute this command to make it executable:

```
$ chmod a+x atlassian-Bitbucket-X.X.X-x64.bin
```

Where `x.x.x` is the Bitbucket version you downloaded.

2. Run the installer - we recommend using `sudo` to run the installer as this will create a dedicated account to run Bitbucket and allow you to run Bitbucket as a service.

```
$ sudo ./atlassian-Bitbucket-X.X.X-x64.bin
```

3. When prompted, choose to start and launch Bitbucket in a browser.
4. Once installation is complete head to `http://localhost:7990` in your browser to begin the setup process.

3. Set up Bitbucket

The Setup Wizard runs automatically when you visit Bitbucket in your browser the first time it's started.

Add your license key

Follow the prompts and head to my.atlassian.com where you can generate a trial Data Center license. Be sure to register with your business email domain to access live chat and phone calls with Bitbucket product specialists.

Create your administrator account

1. Enter details for the administrator account.
2. Click either **Go to Bitbucket** - to go straight to the Bitbucket interface, or **Integrate with Jira** - to create your connection with an existing Jira application.

You can also choose to use a Jira application as your user database during this step. See the page [Configuring Jira integration in the Setup Wizard](#) for details. You can also elect to do this later.

Start using Bitbucket

That's it! Your Bitbucket site is accessible from a URL like this: `http://<computer_name_or_IP_address>:<port>`



What's next?

When setting up Bitbucket in a production environment, we recommend that you configure these aspects next:

- [Connect Bitbucket to a user directory](#) - manage users/groups stored in an external directory.
- [Run Bitbucket as a dedicated user](#) - run Bitbucket from a user account with restricted privileges.
- [Secure the Bitbucket home directory](#) - secure the home directory against unauthorized access.
- [Proxy and secure Bitbucket](#) - run Bitbucket behind a reverse proxy and enable HTTPS access.
- [Establish a data recovery plan](#) - backup the home directory and database of your instance.

Read more about setting up Bitbucket for an enterprise here: [Using Bitbucket in the enterprise](#).

Install Bitbucket Server on Windows

This page describes how to install Bitbucket Server in a production environment, with an external database, using the installer.

This is the most straightforward way to get your production site up and running on a Windows server.



On this page:

Before you begin

Install Bitbucket Server

1. Download Bitbucket Server
2. Run the installer

Set up Bitbucket

3. Connect to your database
4. Add your license key
5. Create your administrator account
6. Start using Bitbucket Server

Other ways to install Bitbucket Server:

- [Evaluation](#)-get your free trial up and running in no time.
- [Zip](#) install Bitbucket Server manually from a zip file.
- [Linux](#) - install Bitbucket Server on a Linux operating system.

Before you begin

Before you install Bitbucket Server, there are a few questions you need to answer.

| | |
|--|---|
| <p>Are you using a supported operating system?</p> | <p>Check the Supported Platforms page for the version of Bitbucket Server you are installing. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none">• The Bitbucket Server installer includes Java (JRE) and Tomcat, so you don't need to install these separately. |
|--|---|

| | |
|--|---|
| <p>Do you want to run Bitbucket Server as a service?</p> | <p>Running Bitbucket Server as a service means that Bitbucket Server will automatically startup when Windows is started.</p> <p>If you choose to run Bitbucket Server as a service:</p> <ul style="list-style-type: none"> • The installer will create a dedicated user account, <code>atlbitbucket</code>, that will run the service. <p>If you choose not to run Bitbucket Server as a service:</p> <ul style="list-style-type: none"> • You will start and stop Bitbucket Server by running the <code>start-bitbucket.bat</code> file in your Bitbucket Server installation directory. • Bitbucket Server will be run as the user account that was used to install Bitbucket Server, or you can choose to run as a dedicated user. • Bitbucket Server will need to be restarted manually if your server is restarted. <p>For more information about these options, see the pages:</p> <ul style="list-style-type: none"> • Running Bitbucket Server as a Windows service • Running Bitbucket Server with a dedicated user |
| <p>Bitbucket uses ports 7990, 7992, and 7993 by default, are they available?</p> | <p>Port 7990: Bitbucket runs on port 7990 by default. If this port is already in use, see Change the port Bitbucket listens on for more guidance on changing this.</p> <p>Ports 7992 and 7993: Bitbucket Server's bundled Elasticsearch instance, which is required for search functionality, requires ports 7992 and 7993 be available. This is not configurable, so ensure these ports are available.</p> |
| <p>Is your database set up and ready to use?</p> | <p>To run Bitbucket in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, and SQL Server. • MySQL is supported for Bitbucket Server instances, but not recommended. See Connecting Bitbucket Server to MySQL for more information. |
| <p>Do you have a Bitbucket Server license?</p> | <p>You'll need a valid license to use Bitbucket Server.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you don't have a license you can create an evaluation license during setup, and be sure to use your business email address. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from Bitbucket Cloud (bitbucket.org), you'll need a new license. |

| | |
|---|--|
| <p>Do you have Git and Perl installed and on the right version?</p> | <p>Bitbucket Server requires Git on the machine that will run Bitbucket Server. If you need to check, install, or upgrade Git on the Bitbucket Server instance machine, see Installing and upgrading Git.</p> <p>Do not upgrade Git to the latest version until you check the Supported Platforms page for which version of Git is currently supported.</p> <p>Check that you have all the other system requirements, including Perl, to avoid any trouble.</p> |
|---|--|

Install Bitbucket Server

1. Download Bitbucket Server

Download the installer - www.atlassian.com/software/bitbucket/download.

2. Run the installer

1. Run the installer. We recommend using a Windows administrator account.
2. Follow the prompts to install Bitbucket. You'll be asked for the following info:
 - a. **Type of Bitbucket instance** - the type of installation, for these instructions select **Standard**.

 If you're installing Data Center on a single node, also choose **Standard**, because this type comes with a bundled Elasticsearch instance and is suitable for a non-clustered Data Center deployment. In the **Data Center** type, you would need to [install it separately](#).

- b. **Installation directory** - where Bitbucket will be installed.
 - c. **Home directory** - where Bitbucket application data will be stored.
 - d. **TCP ports** - the HTTP connector port and control port Bitbucket will run on.
3. Once the installer completes launch Bitbucket in a browser.

Set up Bitbucket

The Setup Wizard runs automatically when you visit Bitbucket Server in your browser the first time.

3. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.

Select **External** as your database, then choose a **Database Type** from the dropdown menu and enter the details of your database.

The JDBC drivers for MySQL / MariaDB are not bundled with Bitbucket Server (due to licensing restrictions), so you need download and install the driver.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).
2. Expand the downloaded `zip/tar.gz` file.
3. Copy the `.jar` file from the extracted directory to your `<Bitbucket home directory>/lib` directory.
4. Restart Bitbucket server.

Also see [Connecting Bitbucket Server to MySQL](#).

4. Add your license key

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

You can also set the base URL at this step, (you can elect to do this later).

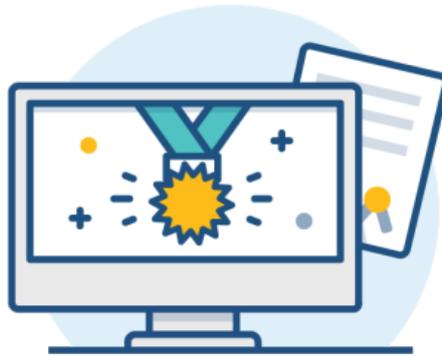
5. Create your administrator account

1. Enter details for the administrator account.
2. Click either **Go to Bitbucket**- to go straight to the Bitbucket interface, or **Integrate with Jira** -to create your connection with an existing Jira application.

You can also chose to use a Jira application as your user database during this step. See the page [Configuring Jira integration in the Setup Wizard](#)for details. You can also elect to do this later.

6. Start using Bitbucket Server

That's it!Your Bitbucket site is accessible from a URL like this: `http://<computer_name_or_IP_address>:<port>`



What's next?

When setting up Bitbucket in a production environment, we recommend that you configure these aspects next:

- [Connect Bitbucket to a user directory](#)- manage users/groups stored in an external directory.
- [Run Bitbucket as a dedicated user](#)- run Bitbucket from a user account with restricted privileges.
- [Secure the Bitbucket home directory](#)-secure the home directory against unauthorized access.
- [Proxy and secure Bitbucket](#) -run Bitbucket behind a reverse proxy and enable HTTPS access.
- [Establish a data recovery plan](#) - backup the home directory and database of your instance.

Read more about setting up Bitbucket for an enterprise here:[Using Bitbucket in the enterprise](#).

Install Bitbucket Server on Windows from a zip file

This page describes how to install Bitbucket Server in a production environment, with an external database, manually using a zip file.

This method gives you the most control of the installation process.



On this page

Before you begin

Install Bitbucket Server

1. Download Bitbucket Server
2. Create the installation directory
3. Create the home directory
4. Start Bitbucket

Set up Bitbucket

5. Connect to your database
6. Add your license key
7. Create your administrator account
8. Start using Bitbucket Server

Other ways to install Bitbucket Server:

- [Evaluation](#)-get your free trial up and running in no time.
- [Installer](#) install Bitbucket Server on Windows installer.
- [Linux](#) - install Bitbucket Server on a Linux operating system.

Before you begin

Before you install Bitbucket Server, there are a few questions you need to answer.

| | |
|---|--|
| <p>Are you using a supported operating system and Java version?</p> | <p>Check the Supported Platforms page for the version of Bitbucket Server you are installing. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none">• We don't support installing Bitbucket Server on macOS for production sites. |
| <p>Do you want to run Bitbucket Server as a service?</p> | <p>Running Bitbucket Server as a service means that Bitbucket Server will automatically startup when Windows is started.</p> <p>You should use the Windows installer if you want to run Bitbucket Server as a service.</p> <p>If you choose not to run Bitbucket Server as a service:</p> <ul style="list-style-type: none">• You will start and stop Bitbucket Server by running the <code>start-bitbucket.bat</code> file in your Bitbucket Server installation directory.• Bitbucket Server will be run as the user account that was used to install Bitbucket Server, or you can choose to run as a dedicated user.• Bitbucket Server will need to be restarted manually if your server is restarted. <p>For more information about these options, see the pages:</p> <ul style="list-style-type: none">• Running Bitbucket Server as a Windows service• Running Bitbucket Server with a dedicated user |

| | |
|---|---|
| <p>Are ports 7990, 7992, and 7993 available?</p> | <p>Port 7990: Bitbucket runs on port 7990 by default. If this port is already in use, see Change the port Bitbucket listens on for more guidance on changing this.</p> <p>Ports 7992 and 7993: Bitbucket Server's bundled Elasticsearch instance, which is required for search functionality, requires ports 7992 and 7993 be available. This is not configurable, so ensure these ports are available.</p> |
| <p>Is your database set up and ready to use?</p> | <p>To run Bitbucket in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, and SQL Server. • MySQL is supported for Bitbucket Server instances, but not recommended. See Connecting Bitbucket Server to MySQL for more information. |
| <p>Do you have a Bitbucket Server license?</p> | <p>You'll need a valid license to use Bitbucket Server.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you don't have a license you can create an evaluation license during setup, and be sure to use your business email address. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from Bitbucket Cloud (bitbucket.org), you'll need a new license. |
| <p>Do you have Git and Perl installed and on the right version?</p> | <p>Bitbucket Server requires Git on the machine that will run Bitbucket Server. If you need to check, install, or upgrade Git on the Bitbucket Server instance machine, see Installing and upgrading Git.</p> <p>Do not upgrade Git to the latest version until you check the Supported Platforms page for which version of Git is currently supported.</p> <p>Check that you have all the other system requirements, including Perl, to avoid any trouble.</p> |
| <p>Is your JAVA_HOME variable set correctly?</p> | <p>Before you install Bitbucket Server, check that you're running a supported Java version and that the JAVA_HOME environment variable is set correctly.</p> <p>To check the JAVA_HOME variable:</p> <p>Open a command prompt and type <code>echo %JAVA_HOME%</code> and hit Enter.</p> <ul style="list-style-type: none"> • If you see a path to your Java installation directory, the JAVA_HOME environment variable has been set correctly. • If nothing is displayed, or only <code>%JAVA_HOME%</code> is returned, you'll need to set the JAVA_HOME environment variable manually. |

| | |
|---|--|
| <p>Do you need to use a remote instance of Elasticsearch?</p> | <p>Bitbucket Server comes with a bundled Elasticsearch instance, which runs as a separate process from the Bitbucket Server application and doesn't require any extra configuration.</p> <p>You can also install a remote Elasticsearch instance on a remote machine, which can provide some advantages allocating memory resources. You're better off using our bundled Elasticsearch that will be automatically configured with its scripts out of the box.</p> <p>If you still want to use a remote instance of Elasticsearch there are instructions for doing so on the page How to Install and configure a remote Elasticsearch instance.</p> |
|---|--|

Install Bitbucket Server

1. Download Bitbucket Server

Download the zip file - www.atlassian.com/software/bitbucket/download.

2. Create the installation directory

1. Create your installation directory (with full control permission) this is where Bitbucket will be installed. Avoid using spaces or special characters in the path. We'll refer to this directory as your `<installation-directory>`.
2. Extract the zip file to your `<installation-directory>`.

3. Create the home directory

1. Create your home directory (with full control permission) this is where your Bitbucket Server data is stored. This should be separate to your installation directory. We'll refer to this directory as your `<Bitbucket home directory>`.
 - Keep the path length to the Bitbucket home directory as short as possible. See [Bitbucket Server is unable to create the merge diff for pull requests](#) for an explanation.
 - Don't use spaces in the path to the Bitbucket home directory.

2. Now tell Bitbucket where your home directory is located. You can do this in one of two ways:

- Edit `<installation-directory>/bin/set-bitbucket-home.bat` file uncomment the `BITBUCKET_HOME` line and add the absolute path to your home directory.

```
rem
set BITBUCKET_HOME=C:\application-data\bitbucket
```

- Or, you can set an environment variable to tell Bitbucket Server where you created your home directory by setting a `BITBUCKET_HOME` environment variable. To do this
 - a. Go to System Environment Variables in Windows.
 - b. Click **Environment Variables**, and then **New** under 'System variables'.
 - c. Enter "BITBUCKET_HOME" as the **Variable name**, and the absolute path to your Bitbucket home directory as the **Variable value**. Don't use a trailing backslash.

4. Start Bitbucket

1. Change directory to the `<installation-directory>` and run this command:

```
bin\start-bitbucket.bat
```

2. In your browser, go to <http://localhost:7990/> and run through the Setup Wizard.

Set up Bitbucket

The Setup Wizard runs automatically when you visit Bitbucket Server in your browser the first time it's started.

5. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.

Select **External** as your database, then choose a **Database Type** from the dropdown menu and enter the details of your database.

The JDBC drivers for MySQL / MariaDB are not bundled with Bitbucket Server (due to licensing restrictions), so you need to download and install the driver.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).
2. Expand the downloaded zip/tar.gz file.
3. Copy the .jar file from the extracted directory to your <Bitbucket home directory>/lib directory.
4. Restart Bitbucket server.

Also see [Connecting Bitbucket Server to MySQL](#).

6. Add your license key

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

You can also set the base URL at this step, (you can elect to do this later).

7. Create your administrator account

1. Enter details for the administrator account.
2. Click either **Go to Bitbucket** - to go straight to the Bitbucket interface, or **Integrate with Jira** - to create your connection with an existing Jira application.

You can also choose to use a Jira application as your user database during this step. See the page [Configuring Jira integration in the Setup Wizard](#) for details. You can also elect to do this later.

8. Start using Bitbucket Server

That's it! Your Bitbucket site is accessible from a URL like this: http://<computer_name_or_IP_address>:<port>



What's next?

When setting up Bitbucket in a production environment, we recommend that you configure these aspects next:

- [Connect Bitbucket to a user directory](#) - manage users/groups stored in an external directory.
- [Run Bitbucket as a dedicated user](#) - run Bitbucket from a user account with restricted privileges.
- [Secure the Bitbucket home directory](#) - secure the home directory against unauthorized access.
- [Proxy and secure Bitbucket](#) - run Bitbucket behind a reverse proxy and enable HTTPS access.
- [Establish a data recovery plan](#) - backup the home directory and database of your instance.

Read more about setting up Bitbucket for an enterprise here: [Using Bitbucket in the enterprise](#).

Install Bitbucket Server on Linux

This page describes how to install Bitbucket Server in a production environment, with an external database, using the Linux installer.

This is the most straightforward way to get your production site up and running on a Linux server.



On this page

[Before you begin](#)

[Install Bitbucket Server](#)

1. [Download Bitbucket Server](#)
2. [Run the installer](#)

[Set up Bitbucket](#)

3. [Connect to your database](#)
4. [Add your license key](#)
5. [Create your administrator account](#)
6. [Start using Bitbucket Server](#)

Other ways to install Bitbucket Server:

- [Evaluation](#)-get your free trial up and running in no time.
- [TAR.GZ](#) install Bitbucket Server manually from an archive file.
- [Windows](#) - install Bitbucket Server on a Windows server.

Before you begin

Before you install Bitbucket Server, there are a few questions you need to answer.

| | |
|--|---|
| <p>Are you using a supported operating system?</p> | <p>Check the Supported Platforms page for the version of Bitbucket Server you are installing. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none">• We don't support installing Bitbucket Server on macOS for production sites.• The Bitbucket Server installer includes Java (JRE) and Tomcat, so you don't need to install these separately. |
|--|---|

| | |
|--|--|
| <p>Do you want to run Bitbucket Server as a service?</p> | <p>Running Bitbucket Server as a service means that Bitbucket Server will automatically start up when Windows is started.</p> <p>If you choose to run Bitbucket Server as a service:</p> <ul style="list-style-type: none"> • You must use <code>sudo</code> to run the installer to be able to install Bitbucket Server as a service. • The installer will create a dedicated user account, <code>atlassian-bitbucket</code>, that will run the service. <p>If you choose not to run Bitbucket Server as a service:</p> <ul style="list-style-type: none"> • You will start and stop Bitbucket Server by running the <code>start-bitbucket.sh</code> file in your Bitbucket Server installation directory. • Bitbucket Server will be run as the user account that was used to install Bitbucket Server, or you can choose to run as a dedicated user. • Bitbucket Server will need to be restarted manually if your server is restarted. <p>For more information about these options, see the pages:</p> <ul style="list-style-type: none"> • Running Bitbucket Server as a Windows service • Running Bitbucket Server with a dedicated user |
| <p>Bitbucket uses ports 7990, 7992, and 7993 by default, are they available?</p> | <p>Port 7990: Bitbucket runs on port 7990 by default. If this port is already in use, see Change the port Bitbucket listens on for more guidance on changing this.</p> <p>Ports 7992 and 7993: Bitbucket Server's bundled Elasticsearch instance, which is required for search functionality, requires ports 7992 and 7993 be available. This is not configurable, so ensure these ports are available.</p> |
| <p>Is your database set up and ready to use?</p> | <p>To run Bitbucket in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, and SQL Server. • MySQL is supported for Bitbucket Server instances, but not recommended. See Connecting Bitbucket Server to MySQL for more information. |
| <p>Do you have a Bitbucket Server license?</p> | <p>You'll need a valid license to use Bitbucket Server.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you don't have a license you can create an evaluation license during setup, and be sure to use your business email address. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from Bitbucket Cloud (bitbucket.org), you'll need a new license. |

| | |
|--|--|
| Do you have Git and Perl installed and on the right version? | <p>Bitbucket Server requires Git on the machine that will run Bitbucket Server. If you need to check, install, or upgrade Git on the Bitbucket Server instance machine, see Installing and upgrading Git.</p> <p>Do not upgrade Git to the latest version until you check the Supported Platforms page for which version of Git is currently supported.</p> <p>Check that you have all the other system requirements, including Perl, to avoid any trouble.</p> |
|--|--|

Install Bitbucket Server

1. Download Bitbucket Server

Download the installer - www.atlassian.com/software/bitbucket/download.

2. Run the installer

1. Make the installer executable.

Change to the directory where you downloaded Bitbucket Server then execute this command:

```
chmod +x atlassian-bitbucket-x.x.x-x64.bin
```

Where `x.x.x` is the Bitbucket Server version you downloaded.

2. Run the installer we recommend using `sudo` to run the installer as this will create a dedicated account to run Bitbucket Server and allow you to run Bitbucket Server as a service.

To use `sudo` to run the installer execute this command:

```
$ sudo ./atlassian-bitbucket-x.x.x-x64.bin
```

Where `x.x.x` is the Bitbucket Server version you downloaded.

You can also run the installer with root user privileges.

3. Follow the prompts to install Bitbucket. You'll be asked for the following info:
 - a. **Type of Bitbucket instance** - the type of installation, for these instructions select **Standard**.
 - b. **Installation directory** - where Bitbucket will be installed.
 - c. **Home directory** - where Bitbucket application data will be stored.
 - d. **TCP ports** - the HTTP connector port and control port Bitbucket will run on.
4. Once the installer completes, head to `http://localhost:7990` in your browser to begin the setup process.

Set up Bitbucket

The Setup Wizard runs automatically when you visit Bitbucket Server in your browser the first time it's started. For details on how the Bitbucket Server Setup Wizard can be completed automatically, see [Automated setup for Bitbucket](#).

3. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.

Select **External** as your database, then choose a **Database Type** from the dropdown menu and enter the details of your database.

The JDBC drivers for MySQL / MariaDB aren't bundled with Bitbucket Server (due to licensing restrictions), so you need to download and install the driver.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).
2. Expand the downloaded zip/tar.gz file.
3. Copy the .jar file from the extracted directory to your <Bitbucket home directory>/lib directory.
4. Restart Bitbucket server.

Also see [Connecting Bitbucket Server to MySQL](#).

4. Add your license key

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

You can also set the base URL at this step, (you can elect to do this later).

5. Create your administrator account

1. Enter details for the administrator account.
2. Click either **Go to Bitbucket** - to go straight to the Bitbucket interface, or **Integrate with Jira** - to create your connection with an existing Jira application.

You can also choose to use a Jira application as your user database during this step. See the page [Configuring Jira integration in the Setup Wizard](#) for details. You can also elect to do this later.

6. Start using Bitbucket Server

That's it! Your Bitbucket site is accessible from a URL like this: `http://<computer_name_or_IP_address>:<port>`



What's next?

When setting up Bitbucket in a production environment, we recommend that you configure these aspects next:

- [Connect Bitbucket to a user directory](#) - manage users/groups stored in an external directory.
- [Run Bitbucket as a dedicated user](#) - run Bitbucket from a user account with restricted privileges.
- [Secure the Bitbucket home directory](#) - secure the home directory against unauthorized access.
- [Proxy and secure Bitbucket](#) - run Bitbucket behind a reverse proxy and enable HTTPS access.
- [Establish a data recovery plan](#) - backup the home directory and database of your instance.

Read more about setting up Bitbucket for an enterprise here: [Using Bitbucket in the enterprise](#).

Install Bitbucket Server on Linux from an archive file

This page describes how to install Bitbucket Server in a production environment, with an external database, manually using atar.gzfile.

This method gives you the most control of the installation process.



On this page

Before you begin

Install Bitbucket Server

1. Download Bitbucket Server
2. Create the installation directory
3. Create the home directory
4. Start Bitbucket

Set up Bitbucket

5. Connect to your database
6. Add your license key
7. Create your administrator account
9. Start using Bitbucket Server

Other ways to install Bitbucket Server:

- [Trial](#)-get a free trial up and running in no time.
- [Installer](#) install Bitbucket using the Linux installer.
- [Windows](#) - install Bitbucket on a Windows server.

Before you begin

Before you install Bitbucket Server, there are a few questions you need to answer.

Are you using a supported operating system and Java version?

Check the [Supported Platforms](#) page for the version of Bitbucket Server you are installing. This will give you info on supported operating systems, databases and browsers.

Good to know:

- We don't support installing Bitbucket Server on macOS for production sites.
- The Bitbucket Server installer includes Java (JRE) and Tomcat, so you don't need to install these separately.

| | |
|--|---|
| <p>Do you want to run Bitbucket Server as a service?</p> | <p>Running Bitbucket Server as a service means that Bitbucket Server will automatically startup when your Linux server is started.</p> <p>You should use the Linux installer if you want to run Bitbucket Server as a service.</p> <p>If you choose not to run Bitbucket Server as a service:</p> <ul style="list-style-type: none"> • You will start and stop Bitbucket Server by running the <code>start-bitbucket.sh</code> file in your Bitbucket Server installation directory. • Bitbucket Server will be run as the user account that was used to install Bitbucket Server, or you can choose to run as a dedicated user. • Bitbucket Server will need to be restarted manually if your server is restarted. <p>For more information about these options, see the pages:</p> <ul style="list-style-type: none"> • Run Bitbucket as a Linux service • Running Bitbucket Server with a dedicated user |
| <p>Bitbucket uses ports 7990, 7992, and 7993 by default, are they available?</p> | <p>Port 7990: Bitbucket runs on port 7990 by default. If this port is already in use, see Change the port Bitbucket listens on for more guidance on changing this.</p> <p>Ports 7992 and 7993: Bitbucket Server's bundled Elasticsearch instance, which is required for search functionality, requires ports 7992 and 7993 be available. This is not configurable, so ensure these ports are available.</p> |
| <p>Is your database set up and ready to use?</p> | <p>To run Bitbucket in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, and SQL Server. • MySQL is supported for Bitbucket Server instances, but not recommended. See Connecting Bitbucket Server to MySQL for more information. |
| <p>Do you have a Bitbucket Server license?</p> | <p>You'll need a valid license to use Bitbucket Server.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you don't have a license you can create an evaluation license during setup, and be sure to use your business email address. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from Bitbucket Cloud (bitbucket.org), you'll need a new license. |

| | |
|---|--|
| <p>Do you have Git and Perl installed and on the right version?</p> | <p>Bitbucket Server requires Git on the machine that will run Bitbucket Server. If you need to check, install, or upgrade Git on the Bitbucket Server instance machine, see Installing and upgrading Git.</p> <p>Do not upgrade Git to the latest version until you check the Supported Platforms page for which version of Git is currently supported.</p> <p>Check that you have all the others system requirements, including Perl, to avoid any trouble.</p> |
| <p>Is your JAVA_HOME variable set correctly?</p> | <p>Before you install Bitbucket Server, check that you're running a supported Java version and that the JAVA_HOME environment variable is set correctly.</p> <p>To check the JAVA_HOME variable:</p> <pre>\$ java -version</pre> <p>To check your JAVA_HOME variable is set correctly:</p> <pre>\$ echo \$JAVA_HOME</pre> <p>If you see a path to your Java installation directory, the JAVA_Home environment variable has been set correctly. If a path is not returned you'll need to set your JAVA_HOME environment variable manually before installing Bitbucket Server.</p> |
| <p>Do you need to use a remote instance of Elasticsearch?</p> | <p>Bitbucket Server comes with a bundled Elasticsearch instance, which runs as a separate process from the Bitbucket Server application and doesn't require any extra configuration.</p> <p>You can also install a remote Elasticsearch instance on a remote machine, which can provide some advantages allocating memory resources. You're better off using our bundled Elasticsearch that will be automatically configured with its scripts out of the box.</p> <p>If you still want to use a remote instance of Elasticsearch there are instructions for doing so on the page How to Install and configure a remote Elasticsearch instance.</p> |

Install Bitbucket Server

1. Download Bitbucket Server

Download the `tar.gz` file - www.atlassian.com/software/bitbucket/download.

2. Create the installation directory

1. Create your installation directory (with full control permission) this is where Bitbucket will be installed. Avoid using spaces or special characters in the path. We'll refer to this directory as your `<installation-directory>`.
2. Extract the `tar.gz` file to your `<installation-directory>`.

3. Create the home directory

1. Create your home directory (with full control permission) this is where your Bitbucket Server data is stored. This should be separate to your installation directory. We'll refer to this directory as your <home-directory>.
2. Edit <installation-directory>/bin/set-bitbucket-home.sh file uncomment the BITBUCKET_HOME line and add the absolute path to your home directory.

4. Start Bitbucket

1. Change directory to the <installation-directory> and run this command:

```
bin/start-bitbucket.sh
```

2. In your browser, go to <http://localhost:7990/> and run through the Setup Wizard.

Set up Bitbucket

The Setup Wizard runs automatically when you visit Bitbucket Server in your browser the first time it's started.

5. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.

Select **External** as your database, then choose a **Database Type** from the dropdown menu and enter the details of your database.

The JDBC drivers for MySQL / MariaDB are not bundled with Bitbucket Server (due to licensing restrictions), so you need download and install the driver.

1. Download the MySQL Connector/J JDBC driver from the [download site](#).
2. Expand the downloaded zip/tar.gz file.
3. Copy the .jar file from the extracted directory to your <Bitbucket home directory>/lib directory.
4. Restart Bitbucket server.

Also see [Connecting Bitbucket Server to MySQL](#).

6. Add your license key

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

You can also set the base URL at this step, (you can elect to do this later).

7. Create your administrator account

1. Enter details for the administrator account.
2. Click either **Go to Bitbucket** - to go straight to the Bitbucket interface, or **Integrate with Jira** - to create your connection with an existing Jira application.

You can also choose to use a Jira application as your user database during this step. See the page [Configuring Jira integration in the Setup Wizard](#) for details. You can also elect to do this later.

9. Start using Bitbucket Server

That's it! Your Bitbucket site is accessible from a URL like this: `http://<computer_name_or_IP_address>:<port>`



What's next?

When setting up Bitbucket in a production environment, we recommend that you configure these aspects next:

- [Connect Bitbucket to a user directory](#) - manage users/groups stored in an external directory.
- [Run Bitbucket as a dedicated user](#) - run Bitbucket from a user account with restricted privileges.
- [Secure the Bitbucket home directory](#) - secure the home directory against unauthorized access.
- [Proxy and secure Bitbucket](#) - run Bitbucket behind a reverse proxy and enable HTTPS access.
- [Establish a data recovery plan](#) - backup the home directory and database of your instance.

Read more about setting up Bitbucket for an enterprise here: [Using Bitbucket in the enterprise](#).

Running Bitbucket Server with a dedicated user

For production installations, we recommend that you create a new dedicated user that will run Bitbucket Server on your system. This user:

- Should be local.
- Should *not* have admin privileges.
- Should be a non-privileged user with read, write, modify and execute access (called "Full control" permission on Windows) on the Bitbucket Server install directory and [home directory](#).

Note that, on Windows, running Bitbucket Server (whether as a service, or not) as a user that is part of the Administrator group can cause Windows to spend a lot of time running permission checks, with a consequent [performance impairment for Git operations](#).

See also [Running Bitbucket Server as a Windows service](#) and [Run Bitbucket as a Linux service](#).

For **Linux**, here is an example of how to create a dedicated user to run Bitbucket Server:

```
$ sudo /usr/sbin/useradd --create-home --home-dir /opt/atlassian/bitbucket --shell /bin/bash atlbitbucket
```

On **Windows**, the user must be a member of the **Performance Monitor Users** group.

Running Bitbucket Server as a Windows service

This page describes how to run Bitbucket Server as a Windows service.

We recommend that you use the Bitbucket Server installer to install Bitbucket Server as a service on Windows. It creates items in the Windows 'Start' menu for starting and stopping Bitbucket Server. See [Getting started](#) for more details.

See [Install Bitbucket Server from an archive file](#) if you are manually installing or upgrading Bitbucket Server from an archive file.

For long-term use on a Windows server, Bitbucket Server should be configured to run as a Windows service. This has the following advantages:

- Bitbucket Server will be automatically restarted when the operating system restarts.
- Bitbucket Server is less likely to be accidentally shut down, as can happen if the console window Bitbucket Server was manually started in is closed.
- Bitbucket Server logs are properly managed by the Windows service.



System administration tasks are [not supported by Atlassian](#). These instructions are only provided as a guide.

Prerequisites

- On any Windows operating system with User Account Control (UAC) such as Windows 7, simply logging in to Windows with an administrator account will not be sufficient to execute the script in the procedure below. You must either disable UAC or run `cmd.exe` as an administrator (e.g. by right-clicking on `cmd.exe` then choose **Run as administrator**).
- Ensure the `JAVA_HOME` variable is set to the root of your Java platform's installation directory.



Your `JAVA_HOME` cannot contain spaces, so the default Java installation directory of `C:\Program Files\Java` won't work.

- Bitbucket Server should be run from a local [dedicated user account](#) that does not have admin privileges and that has read, write and execute access to the Bitbucket home directory and the `<Bitbucket Server installation directory>`. See [Git push operations extremely slow on Windows](#).
- When you run Bitbucket Server as a Windows service, all settings in `_start-webapp.bat` are ignored. Ensure that you have set `BITBUCKET_HOME` as a `system` environment variable, before running the `service.bat` script.

Set up Bitbucket Server as a Windows service

As of version 4.6, installing Bitbucket Server as a Windows service requires installing two services: the Bitbucket Server service, which provides application functionality, and the bundled Bitbucket Elasticsearch service, which provides content indexing and search functionality.

To install Bitbucket Server as a Windows service

On this page:

- [Prerequisites](#)
- [Set up Bitbucket Server as a Windows service](#)
- [Troubleshooting](#)

Related pages:

- [Run Bitbucket as a Linux service](#)
- [Use Bitbucket in the enterprise](#)

1. [Stop Bitbucket Server](#).
2. Create a system environment variable with `BITBUCKET_HOME` as the **Variable name** and the absolute path to your Bitbucket Server [home directory](#) as the **Variable value**. Don't use a trailing backslash. Note that the Bitbucket home directory *should not* be located inside the `<Bitbucket Server installation directory>`. You must do this step *before* running the `service.bat` script in Step 5 below.
3. Create these system environment variables required for the bundled Bitbucket Elasticsearch service:

| Variable name | Variable value |
|---------------|-------------------------------------|
| LOG_DIR | %BITBUCKET_HOME%\log\search |
| DATA_DIR | %BITBUCKET_HOME%\shared\search\data |
| CONF_DIR | %BITBUCKET_HOME%\shared\search |

4. Open a Command Prompt (as an administrator see the [Prerequisites](#) section above).
5. Change directory to `<Bitbucket Server installation directory>\bin`. If a directory in the path has spaces (e.g. `C:\Program Files\..`), use its eight-character equivalent (e.g. `C:\Progra~1\..`).
6. Create a service with the name "BITBUCKET" and a display name of "Atlassian Bitbucket Server BITBUCKET" by running this command:

```
service.bat install
```

If you would like to customize the name you can instead run:

```
service.bat install MyName
```

This creates a service called "MyName" with a display name of "Atlassian Bitbucket Server MyName".

7. Change directory to `<Bitbucket Server installation directory>\elasticsearch`.
8. If you are installing a new instance of Bitbucket Server, or are upgrading from Bitbucket Server 4.5 or earlier, copy the template Elasticsearch config files over to your Bitbucket home directory by running the following command:

```
robocopy /S config-template %BITBUCKET_HOME%\shared\search
```

9. Change directory to `<Bitbucket Server installation directory>\elasticsearch\bin`.
10. Create a service with the name "BITBUCKET_SEARCH" and a display name of "Atlassian Bitbucket Elasticsearch (BITBUCKET_SEARCH)" by running this command:

```
service.bat install
```

If you would like to customise the name you can instead run:

```
service.bat install MyName
```

This creates the service "MyName" with a display name of "Atlassian Bitbucket Elasticsearch (MyName)".

11. Verify the Bitbucket Server and Bitbucket Elasticsearch services start after restarting the machine.

Here is an example of the output you can expect after successfully completed this procedure

```
C:\Atlassian\atlassian-bitbucket-4.6.3\bin>service.bat install
Installing the service 'BITBUCKET' ...
Using CATALINA_HOME:      "C:\Atlassian\atlassian-bitbucket-4.6.3"
Using CATALINA_BASE:     "C:\Atlassian\atlassian-bitbucket-4.6.3"
Using JAVA_HOME:         "c:\java\jdk1.8.0_92"
Using JRE_HOME:          "c:\java\jdk1.8.0_92\jre"
Using JVM:                "c:\java\jdk1.8.0_92\jre\bin\server\jvm.dll"
The service 'BITBUCKET' has been installed.

C:\Atlassian\atlassian-bitbucket-4.6.3\bin>cd ..\elasticsearch\bin
C:\Atlassian\atlassian-bitbucket-4.6.3\elasticsearch\bin>service.bat install
Installing service       : "BITBUCKET_SEARCH"
Using JAVA_HOME (64-bit): "c:\java\jdk1.8.0_92"
The service 'BITBUCKET_SEARCH' has been installed.

C:\Atlassian\atlassian-bitbucket-4.6.3\elasticsearch\bin>net start BITBUCKET_SEARCH
The Atlassian Bitbucket Elasticsearch 2.3.1 (BITBUCKET_SEARCH) service is starting.
The Atlassian Bitbucket Elasticsearch 2.3.1 (BITBUCKET_SEARCH) service was started successfully.

C:\Atlassian\atlassian-bitbucket-4.6.3\elasticsearch\bin>net start BITBUCKET
The Atlassian Bitbucket BITBUCKET service is starting.
The Atlassian Bitbucket BITBUCKET service was started successfully.
```

Troubleshooting

If your service fails to start with code 4, ensure you ran `service.bat install` in a Command Prompt running as an Administrator.

Run Bitbucket as a Linux service

This page describes how to run Bitbucket Data Center and Server as a Linux service, and only applies if you are manually installing or upgrading Bitbucket from an archive file. See the page [Install Bitbucket Server from an archive file](#) for more details.

Bitbucket assumes that the external database is available when it starts; these approaches do not support service dependencies, and the startup scripts will not wait for the external database to become available.

For production use on a Linux server, Bitbucket should be configured to run as a Linux service, that is, as a daemon process. This has the following advantages:

- Bitbucket can be automatically restarted when the operating system restarts.
- Bitbucket can be automatically restarted if it stops for some reason.
- Bitbucket is less likely to be accidentally shut down, as can happen if the terminal Bitbucket was manually started in is closed.
- Logs from the Bitbucket JVM can be properly managed by the service.



System administration tasks are [not supported by Atlassian](#). These instructions are only provided as a guide and may not be up to date with the latest version of your operating system.

On this page:

- [Using the Java Service Wrapper](#)
- [Using an init.d script](#)
- [Using a systemd unit file](#)

Related pages:

- [Running Bitbucket Server as a Windows service](#)
- [Use Bitbucket in the enterprise](#)

Using the Java Service Wrapper

Bitbucket can be run as a service on Linux using the [Java Service Wrapper](#). The Service Wrapper is [known to work with](#) Debian, Ubuntu, and Red Hat.

The Service Wrapper provides the following benefits:

- Allows Bitbucket, which is a Java application, to be run as a service.
- No need for a user to be logged on to the system at all times, or for a command prompt to be open and running on the desktop to be able to run Bitbucket.
- The ability to run Bitbucket in the background as a service, for improved convenience, system performance and security.
- Bitbucket is launched automatically on system startup and does not require that a user be logged in.
- Users are not able to stop, start, or otherwise tamper with Bitbucket unless they are an administrator.
- Can provide advanced failover, error recovery, and analysis features to make sure that Bitbucket has the maximum possible uptime.

Please see <http://wrapper.tanukisoftware.com/doc/english/launch-nix.html> for wrapper installation and configuration instructions.

The service wrapper supports the standard commands for SysV init scripts, so it should work if you just create a symlink to it from `/etc/init.d`.

Using an init.d script

The usual way on Linux to ensure that a process restarts at system restart is to use an `init.d` script. This approach does not restart Bitbucket if it stops by itself.

1. [Start and stop Bitbucket.](#)
2. Create a user, set the permissions to that user, create a home directory for Bitbucket and create a symlink to make upgrades easier:

```
$> curl -OL https://www.atlassian.com/software/stash/downloads/binary/atlassian-bitbucket-X.Y.Z.tar.gz
$> tar xz -C /opt -f atlassian-bitbucket-X.Y.Z.tar.gz
$> ln -s /opt/atlassian-bitbucket-X.Y.Z /opt/atlassian-bitbucket-latest

# Create a home directory
$> mkdir /opt/bitbucket-home

# ! Update permissions and ownership accordingly
```

(Be sure to replace X.Y.Z in the above commands with the version number of Bitbucket.)

3. Create the startup script in `/etc/init.d/bitbucket` with the following contents (Ensure the script is executable by running `chmod 755 bitbucket`):

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:          bitbucket
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Initscript for Atlassian Bitbucket Server
# Description:       Automatically start Atlassian Bitbucket Server when the system starts up.
#                   Provide commands for manually starting and stopping Bitbucket Server.
### END INIT INFO

# Adapt the following lines to your configuration
# RUNUSER: The user to run Bitbucket Server as.
RUNUSER=vagrant

# BITBUCKET_INSTALLDIR: The path to the Bitbucket Server installation directory
BITBUCKET_INSTALLDIR="/opt/atlassian-bitbucket-X.Y.Z"

# BITBUCKET_HOME: Path to the Bitbucket home directory
BITBUCKET_HOME="/opt/bitbucket-home"

# =====
# =====
# =====

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="Atlassian Bitbucket Server"
NAME=bitbucket
PIDFILE=$BITBUCKET_HOME/log/bitbucket.pid
SCRIPTNAME=/etc/init.d/$NAME

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.0-6) to ensure that this file is present.
. /lib/lsb/init-functions

run_with_home() {
    if [ "$RUNUSER" != "$USER" ]; then
        su - "$RUNUSER" -c "export BITBUCKET_HOME=${BITBUCKET_HOME}; ${BITBUCKET_INSTALLDIR}/bin
/$1"
    else
        export BITBUCKET_HOME=${BITBUCKET_HOME}; ${BITBUCKET_INSTALLDIR}/bin/$1
    fi
}
```

```

#
# Function that starts the daemon/service
#
do_start()
{
    run_with_home start-bitbucket.sh
}

#
# Function that stops the daemon/service
#
do_stop()
{
    if [ -e $PIDFILE ]; then
        run_with_home stop-bitbucket.sh
    else
        log_failure_msg "$NAME is not running."
    fi
}

case "$1" in
start)
    [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
    do_start
    case "$?" in
        0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
        2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
stop)
    [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
    do_stop
    case "$?" in
        0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
        2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
    esac
    ;;
status)
    if [ ! -e $PIDFILE ]; then
        log_failure_msg "$NAME is not running."
        return 1
    fi
    status_of_proc -p $PIDFILE "" $NAME && exit 0 || exit $?
    ;;
restart|force-reload)
    #
    # If the "reload" option is implemented then remove the
    # 'force-reload' alias
    #
    log_daemon_msg "Restarting $DESC" "$NAME"
    do_stop
    case "$?" in
        0|1)
            do_start
            case "$?" in
                0) log_end_msg 0 ;;
                1) log_end_msg 1 ;; # Old process is still running
                *) log_end_msg 1 ;; # Failed to start
            esac
            ;;
        *)
            # Failed to stop
            log_end_msg 1
            ;;
    esac
    ;;
*)
    echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-reload}" >&2
    exit 3
    ;;
esac

```

Running on system boot

1. To start on system boot, add the script to the start up process.
For Ubuntu (and other Debian derivatives) use:

```
update-rc.d bitbucket defaults
```

For RHEL (and derivatives) use:

```
chkconfig --add bitbucket --level 0356
```

Note: You may have to install `theredhat-lsbpackage` on RHEL (or derivatives) to provide the LSB functions used in the script.

2. Verify that the Bitbucket service comes back up after restarting the machine.

Using a systemd unit file

Thanks to [Patrick Nelson](#) for calling out this approach, which he set up for a Fedora system. It also works on other distributions that use systemd as the init system. This approach does not restart Bitbucket if it stops by itself.

1. Create a `bitbucket.service` file in your `/etc/systemd/system/` directory with the following lines:

```
[Unit]
Description=Atlassian Bitbucket Server Service
After=syslog.target network.target

[Service]
Type=forking
User=atlbitbucket
ExecStart=/opt/atlassian-bitbucket-X.Y.Z/bin/start-bitbucket.sh
ExecStop=/opt/atlassian-bitbucket-X.Y.Z/bin/stop-bitbucket.sh

[Install]
WantedBy=multi-user.target
```

The value for `User` should be adjusted to match the user that Bitbucket runs as. `ExecStart` and `ExecStop` should be adjusted to match the path to your `<Bitbucket installation directory>`.

2. Enable the service to start at boot time by running the following in a terminal:

```
systemctl enable bitbucket.service
```

3. [Start and stop Bitbucket](#), then restart the system, to check that Bitbucket starts as expected.
4. Use the following commands to manage the service:

Disable the service:

```
systemctl disable bitbucket.service
```

Check that the service is set to start at boot time:

```
if [ -f /etc/systemd/system/*.wants/bitbucket.service ]; then echo "On"; else echo "Off"; fi
```

Manually start and stop the service:

```
systemctl start bitbucket  
systemctl stop bitbucket
```

Check the status of Bitbucket Server:

```
systemctl status bitbucket
```

Automated setup for Bitbucket

This page describes how the Bitbucket Data Center and Server Setup Wizard can be completed automatically, that is without the need for manual interaction in the browser. See [Getting started](#) for an outline of the tasks that the Setup Wizard assists with when setting up Bitbucket manually.

You might want to configure this when automating the provisioning of Bitbucket Data Center, for example. Of course, you'll need to configure provisioning tools such as Puppet or Vagrant yourself.

Note that you can also automate the 'install and launch' phase of provisioning Bitbucket see [Running the Bitbucket Server installer](#) for information about how to run the installer in the console and unattended modes.

1. Get a license for Bitbucket

You can get a new Bitbucket license by:

- Logging in to your [My.Atlassian.com](#) account.
- Contacting Atlassian.

If you already have a licensed instance of Bitbucket, you can find the license in the Bitbucket admin area.

2. Set the configuration properties

After installing Bitbucket, but before you start Bitbucket for the first time, edit the `bitbucket.properties` file to add the properties in the table below. Use the standard format for Java properties files.

Note that the `bitbucket.properties` file is created automatically in the `shared` folder of your [Bitbucket home directory](#) when you perform a [database migration](#). Create the file yourself if it does not yet exist. See [Configuration properties](#) for information about the properties file.

Add these properties to the `bitbucket.properties` file:

| Property | Description |
|--|---|
| <code>setup.displayName=displayName</code> | The display name for the Bitbucket application. |
| <code>setup.baseUrl= https:// bitbucket.yourcompany.com</code> | The base URL. |
| <code>setup.license=AAAB...</code> | The Bitbucket license.

Use the <code>\</code> character at the end of each line if you wish to break the license string over multiple lines. |
| <code>setup.sysadmin.username=username</code> | Credentials for the system admin account. |
| <code>setup.sysadmin.password=password</code> | |
| <code>setup.sysadmin.displayName=John Doe</code> | The display name for the system admin account.

An empty property is ignored. |
| <code>setup.sysadmin.emailAddress=sysadmin@ourcompany.com</code> | The email address for the system admin account. |
| <code>jdbc.driver=org.postgresql.Driver</code> | JDBC connection parameters.

Use the appropriate values for your own JDBC connection. |

| | |
|--|---|
| <pre>jdbc.url=jdbc: postgresql://localhost: 5432/bitbucket</pre> | <p>For Postgres, If no parameter is manually specified in the <code>jdbc.url</code>, Bitbucket Server will automatically append <code>?targetServerType=master</code> in the JDBC URL.</p> |
| <pre>jdbc.user=bitbucket</pre> | |
| <pre>jdbc.password=bitbucket</pre> | |
| <pre>plugin.mirroring. upstream.url=https://bi tbucket.company.com</pre> | <p>(Smart Mirroring only) On the mirror, specifies the base URL of the primary Bitbucket Data Center instance that the new mirror will be mirroring from. See Set up a mirror for more information.</p> |

You should specify the JDBC properties so that Bitbucket can connect to the external database.

If any of the following required properties are not provided in the properties file, when you start Bitbucket the Setup Wizard will launch at the appropriate screen so that you can enter values for those properties.

3. Start Bitbucket

Start Bitbucket as usual. See [Start and stop Bitbucket](#).

Bitbucket reads the `bitbucket.properties` file and applies the setup properties automatically.

When you now visit Bitbucket in the browser, you see the welcome page.

Troubleshooting

The Setup Wizard launches in the browser

The Setup Wizard will run if there are missing configuration properties, such as the license string, in the `bitbucket.properties` file. Check the properties file and compare it with the table in Step 2 above. Alternatively, the setup can be completed using the web UI.

Write access for the `config.properties` file

Once the automated setup process completes, the relevant properties in the `bitbucket.properties` file are commented out. This requires that the system user has write permission on the properties file.

Bitbucket fails to start with a 'Could not acquire change log lock.' error

If Bitbucket is forced to quit while modifying the `config.properties` file, you may not be able to restart Bitbucket, and `atlassian-bitbucket.log` contains the above error.

See this KB article for information about how to resolve this: [Bitbucket Server Does Not Start - Could not acquire change log lock](#)

Start and stop Bitbucket

This page describes the various ways you can start or stop Bitbucket Data Center and Server, depending on which method suits your needs.

Start Bitbucket during installation

The Bitbucket installer can automatically start Bitbucket. If you're not using the installer, use the instructions below that are appropriate for your needs to start Bitbucket.

On this page

- [Start Bitbucket during installation](#)
- [Start/stop Bitbucket](#)
- [Start Bitbucket Data Center](#)
- [Start/stop Bitbucket when running as a service](#)
- [Start Bitbucket without starting bundled Elasticsearch](#)
- [Performing a graceful shutdown \(Linux only\)](#)

Start/stop Bitbucket

To start/stop Bitbucket

For Linux

Change to your `<Bitbucket Server installation directory>`, then use the command that meets your needs:

```
bin/start-bitbucket.sh  
bin/stop-bitbucket.sh
```

For Windows

For automated starting and stopping of Bitbucket Server on Windows, **it should be installed as a service**.

You can use `start-bitbucket.bat` for short term debugging, using Ctrl+C will safely shutdown the instance when it is no longer needed.

To start/stop Bitbucket Server manually when running as a service on Windows

Start and stop the Bitbucket Server service from the services console, on Windows. Ensure you start both the `AtlassianBitbucket` and `AtlassianBitbucketElasticsearch` services.

For macOS

Use the app icons in the `<Bitbucket Server installation directory>`. These link to the `start-bitbucket.sh` and `stop-bitbucket.sh` scripts in `<Bitbucket Server installation directory>/bin`.



[Start Bitbucket Data Center](#)

For Bitbucket Data Center, the `remoteElasticsearchInstance` must be started separately. It is a requirement of the Bitbucket Data Center setup that you have *only one* remote Elasticsearch instance for your entire cluster, as described on [Installing Bitbucket Data Center](#). To start (or restart) a remote Elasticsearch instance, see Elasticsearch documentation for specific instructions.

To start Bitbucket Data Center (*does not start Bitbucket's bundled Elasticsearch instance*)

1. Change to your `<Bitbucket Server installation directory>`
2. Run this command:

```
start-bitbucket.sh --no-search
```

Start/stop Bitbucket when running as a service

On Windows and Linux systems you can choose to have Bitbucket installed as a service, and it will be started automatically when the system boots.

To start/stop Bitbucket manually when running as service:

For Linux

Manage the Bitbucket service with these commands:

```
# service atlbitbucket status
# service atlbitbucket stop
# service atlbitbucket start
```

If you installed Bitbucket as a `systemd` service (as explained in [Run Bitbucket as a Linux service](#)), use these commands instead:

```
# systemctl bitbucket status
# systemctl bitbucket stop
# systemctl bitbucket start
```

Start Bitbucket without starting bundled Elasticsearch

You're running Bitbucket Data Center: It is a requirement of the Bitbucket Data Center setup that you have *only one* remote Elasticsearch instance for your cluster, as described on [Installing Bitbucket Data Center](#).

You have a remote Elasticsearch instance (for whatever reason): When using a remote Elasticsearch instance, you have to start Bitbucket without starting the bundled Elasticsearch instance.

You can also refer to [How to Install and configure a remote Elasticsearch instance](#).

You have a Docker deployment: We strongly recommend that you run Elasticsearch on a separate container. Otherwise, you won't be able to properly trigger a graceful shutdown.

To start Bitbucket without starting the bundled Elasticsearch instance

For Linux

```
start-bitbucket.sh --no-search
```

For Windows

```
start-bitbucket.bat /no-search
```

For Docker installations, use the `-e ELASTICSEARCH_ENABLED=false` parameter. [Learn more about deploying Bitbucket through Docker](#)

Performing a graceful shutdown (Linux only)

You can send a `SIGTERM` signal to Bitbucket to trigger a graceful shutdown. This will give Bitbucket some time to complete any running tasks, in-flight GIT requests, and in-flight user requests before terminating. The `bin/stop-bitbucket.sh` script, `service atlbitbucket stop` command, and `systemctl bitbucket stop` command all use the `SIGTERM` signal to terminate Bitbucket.

During a graceful shutdown:

- Tomcat stops accepting new HTTP connections without interrupting active connections.
- The SSH server stop accepting new SSH connections without interrupting active connections.
- The job scheduler will not start any new jobs, but will not immediately cancel any running jobs.

The following [config properties](#) control how graceful shutdown works:

| Default value | Description |
|---------------|---|
| | <code>plugin.search.indexing.event.shutdown.timeout</code> |
| 30 | The amount of time (in seconds) that Bitbucket should allocate before terminating all active HTTP requests, SSH requests, and running jobs. |
| | <code>plugin.search.indexing.event.graceful.shutdown</code> |
| graceful | Sets whether Bitbucket should initiate a graceful shutdown first upon termination. If you want Bitbucket to shut down immediately without waiting for any tasks to complete, set this to <code>immediate</code> . |

In addition to these config properties, the `BITBUCKET_SHUTDOWN_TIMEOUT` environment variable controls when Bitbucket terminates altogether. This environmental variable will override `plugin.search.indexing.event.shutdown.timeout`.

By default, `BITBUCKET_SHUTDOWN_TIMEOUT` is set to 40. We recommend you keep this environmental variable at least 10 seconds longer than `plugin.search.indexing.event.shutdown.timeout`.

Gracefully shutting down Docker deployments

If Bitbucket and Elasticsearch are running in the same container, you won't be able to trigger a graceful shutdown properly. We strongly recommend that you run Elasticsearch on its own separate container.

If ElasticSearch is not running on the same container, you can trigger a graceful Bitbucket shutdown through `docker stop`. This command will send a `SIGTERM` signal to Bitbucket.

Install Bitbucket Server from an archive file

This page describes how to manually install Bitbucket Server from an archive file. However, we strongly recommend that you use the [Bitbucket Server installer](#) instead, for a quick and trouble-free install experience.

Related pages

- See [Getting started](#), and consider using the installer
- [Use Bitbucket in the enterprise](#)
- [Docker container image for Bitbucket Server](#)

1. Check supported platforms

Check the [Supported platforms](#) page for details of the application servers, databases, operating systems, web browsers and Java and Git versions that we have tested Bitbucket Server with and recommend.

Atlassian only officially supports Bitbucket Server running on x86 hardware and 64-bit derivatives of x86 hardware.

 **Cygwin Git is *not supported*.** No internal testing is done on that platform, and many aspects of Bitbucket Server's functionality (pull requests and forks among them) have known issues. When running Bitbucket Server on Windows, *always* use msysGit.

2. Check your version of Java

In a terminal or command prompt, run this:

```
java -version
```

The version of Java should be **1.8.x**. You'll need a 64-bit version of Java if you have a 64-bit operating system.

Install Java

Download Java Server JRE from [Oracle's website](#), and install it.

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

Check that the system can find Java

In a terminal, run this:

```
echo $JAVA_HOME
```

You should see a path like `/usr/jdk/jdk1.8.0`.

If you don't see a path, then set JAVA_HOME

Do one of the following:

- If `JAVA_HOME` is not set, log in with 'root' level permissions and run:

```
echo JAVA_HOME="path/to/JAVA_HOME" >> /etc/environment
```

where `path/to/JAVA_HOME` may be like: `/usr/jdk/jdk1.8.0`

- If `JAVA_HOME` needs to be changed, open the `/etc/environment` file in a text editor and modify the value for `JAVA_HOME` to:

```
JAVA_HOME="path/to/JAVA_HOME"
```

It should look like: `/usr/jdk/jdk1.8.0`

Install Java

Download Java Server JRE from [Oracle's website](#), and install it.

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

Check that the system can find Java

In a terminal, run this:

```
echo $JAVA_HOME
```

You should see a path like /System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/.

If you don't see a path, then set JAVA_HOME

Open your ~/.profile file in a text editor and insert:

```
JAVA_HOME="path/to/JAVA_HOME"  
export JAVA_HOME
```

where path/to/JAVA_HOME may be like: /System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Home/

Refresh your ~/.profile in the terminal and confirm that JAVA_HOME is set:

```
source ~/.profile  
$JAVA_HOME/bin/java -version
```

You should see a version of Java that is **1.8.x**, like this:

```
java version "1.8.0_1"
```

Install Java

Download Java Server JRE from [Oracle's website](#), and install it.

Now try running 'java -version' again to check the installation. The version of Java should be **1.8.x**.

Check that the system can find Java

Bitbucket Server uses the JAVA_HOME environment variable to find Java. To check that, in a command prompt, run:

```
echo %JAVA_HOME%
```

You should see a path to the root directory of the Java installation. When running Bitbucket Server on Windows, unlike Linux or Unix, JAVA_HOME paths with spaces are just fine.

If you don't see a path, then set JAVA_HOME

Stage 1. Locate the JRE installation directory

If you already know the installation path for the Java Runtime Environment, go to *Stage 2* below. Otherwise, find the installation path by following these instructions:

1. If you didn't change the installation path for the JavaRuntime Environment during installation, it will be in a directory under `C:\Program Files\Java`. Using Explorer, open the directory `C:\Program Files\Java`.
2. Inside that path will be one or more subdirectories such as `C:\Program Files\Java\jre8`.

Stage 2. Set the JAVA_HOME variable

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "JAVA_HOME" as the **Variable name**, and the absolute path to where you installed Java as the **Variable value**. Don't use a trailing backslash, and don't wrap the value in quotes.

Now, in a *new command prompt*, try running `%JAVA_HOME%\bin\java -version`. You should see the same version of Java as you saw in 2. above.

Stage 1. Locate the JRE installation directory

If you already know the installation path for the Java Runtime Environment, go to *Stage 2* below. Otherwise, find the installation path by following these instructions:

1. If you didn't change the installation path for the JavaRuntime Environment during installation, it will be in a directory under `C:\Program Files\Java`. Using Explorer, open the directory `C:\Program Files\Java`.
2. Inside that path will be one or more subdirectories such as `C:\Program Files\Java\jre8`.

Stage 2. Set the JAVA_HOME variable

Once you have identified the JRE installation path:

1. Right-click the **My Computer** icon on your desktop and select **Properties**.
2. Click the **Advanced** tab.
3. Click the **Environment Variables** button.
4. Under **System Variables**, click **New**.
5. Enter the **variable name** as `JAVA_HOME`.
6. Enter the **variable value** as the installation path for the Java Development Kit. Don't use a trailing backslash, and don't wrap the value in quotes.
 - If your Java installation directory has a space in its path name, you should use the shortened path name (e.g. `C:\Progra~1\Java\jre7`) in the environment variable instead.

Note for Windows users on 64-bit systems

Progra~1 = 'Program Files'
Progra~2 = 'Program Files(x86)'

7. Click **OK**.
8. Click **Apply Changes**.
9. Close any command window which was open before you made these changes, and open a new command window. There is no way to reload environment variables from an active command prompt. If the changes do not take effect even after reopening the command window, restart Windows.

Now, in a *new command prompt*, try running `%JAVA_HOME%\bin\java -version`. You should see the same version of Java as you saw in 2. above.

3. Check your versions of Git and Perl

In a terminal or command prompt, run:

```
git --version
perl --version
```

The version of Git should be **1.8.x** or higher. The version of Perl should be **5.8.8** or higher.

If you don't see supported versions of Git and Perl, either install or upgrade them see [Installing and upgrading Git](#).

4. Now it's time to get Bitbucket Server

[Download latest version](#) →

[Download Bitbucket Server](#) from the Atlassian download site.
Looking for the [Bitbucket Server WAR file](#)?

Extract the downloaded file to an install location (without spaces in the path).

The path to the extracted directory is referred to as the `<Bitbucket Server installation directory>` in these instructions.

 Never unzip the Bitbucket Server archive file over the top of an existing Bitbucket Server installation each version of Bitbucket Server includes versioned jar files, such as `bitbucket-model-4.0.0.jar`. If you copy these, you end up with multiple versions of Bitbucket Server's jar files in the classpath, which leads to runtime corruption.

Note that you should use the same user account to both extract Bitbucket Server and to run Bitbucket Server (in Step 6.) to avoid possible permission issues at startup. For production installations, we recommend that you create a new dedicated user that will run Bitbucket Server on your system. See [Running Bitbucket Server with a dedicated user](#).

5. Tell Bitbucket Server where to store your data

The Bitbucket Server [home directory](#) is where your Bitbucket Server data is stored.

If you are upgrading Bitbucket Server, simply update the value of `BITBUCKET_HOME` in the `<Bitbucket Server installation directory>/bin/set-bitbucket-homefile` so the *new* Bitbucket Server installation points to your *existing* Bitbucket Server [home directory](#) (if you use a `BITBUCKET_HOME` environment variable to specify the home directory location, no change is required).

Otherwise, for a new install, create your Bitbucket home directory (without spaces in the name), and then tell Bitbucket Server where you created it by editing the `<Bitbucket Server installation directory>/bin/set-bitbucket-home.sh` (or `set-bitbucket-home.bat`) file uncomment the `BITBUCKET_HOME` line and add the absolute path to your home directory. Here's an example of what that could look like when you're done:

```
#
if [{"x${BITBUCKET_HOME}" = "x"]; then
    export BITBUCKET_HOME="/home/username/bitbucket_home"
fi
```

 **You should not locate your Bitbucket home directory inside the `<Bitbucket Server installation directory>`** they should be entirely separate locations. If you do put the home directory in the `<Bitbucket Server installation directory>` it may be overwritten, and lost, when Bitbucket Server gets upgraded. And by the way, you'll need separate Bitbucket Server home directories if you want to run multiple instances of Bitbucket Server.

Tell Bitbucket Server where you created it by setting `BITBUCKET_HOME` environment variable, for Windows 7, as follows:

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "`BITBUCKET_HOME`" as the **Variable name**, and the absolute path to your Bitbucket home directory as the **Variable value**. Don't use a trailing backslash.

There are a few things to know about setting up the Bitbucket home directory on Windows that will make life easier:

- Keep the path length to the Bitbucket home directory as short as possible. See [Bitbucket Server is unable to create the merge diff for pull requests](#) for an explanation.
- Don't use spaces in the path to the Bitbucket home directory.

6. Move server.xml to your Bitbucket Server home `shared` directory

If this is a new installation, or you are already running Stash 3.8 or above, you can [skip to the next step](#).

If you are upgrading from Stash 3.7 or earlier and you made any changes to `<Bitbucket Server installation directory>/conf/server.xml` (for instance to [secure your server with SSL](#)):

1. In the `<BITBUCKET_HOME>` directory, make a new directory called `shared`.
2. Then, copy your modified `server.xml` file into `<BITBUCKET_HOME>/shared/`. Ensure the copied file is readable by the [user account that runs Bitbucket Server](#).

7. (Optional) Install and configure a remote Elasticsearch instance

Bitbucket Server 4.5+ comes with an bundled [Elasticsearch](#) instance, which runs as a separate process from the Bitbucket Server application, and does not require any extra configuration.

If you plan to use the bundled Elasticsearch instance, jump to the next step, [Start Bitbucket Server!](#)

However, you can also install Elasticsearch on a remote machine, which can provide some advantages allocating memory resources. Read the instructions for installing and configuring a remote Elasticsearch instance here: [How to Install and configure a remote Elasticsearch instance](#).

Elasticsearch ports

Bitbucket Server's bundled Elasticsearch instance requires ports 7992 and 7993 be available to provide code search functionality. This is not configurable, so ensure these ports are available.

8. Start Bitbucket Server!

There are a couple of ways in which you can start Bitbucket Server see [Start and stop Bitbucket](#).

If you've setup a remote Elasticsearch instance you probably do not want to start the bundled Elasticsearch instance.

To start Bitbucket Server with a remote Elasticsearch instance

When using a remote Elasticsearch instance, instead of the bundled Elasticsearch instance, start Bitbucket Server by running `start-webapp.sh` instead of `start-bitbucket.sh`. This starts Bitbucket Server alone without running the bundled Elasticsearch instance.

Finish configuring Bitbucket Server

Now, in your browser, go to <http://localhost:7990/> and run through the Setup Wizard. In the Setup Wizard:

- If you're evaluating Bitbucket Server, select **Internal** at the 'Database' step. Bitbucket Server will use its internal database, and you can easily migrate to external database later. See [Connect Bitbucket to an external database](#).
- Enter your Bitbucket Server license key.
- [Set the base URL for Bitbucket Server](#).
- Set up an administrator account.
- You can set up Jira Software integration, but you can do this later if you wish. See [Configuring Jira integration in the Setup Wizard](#).

9. Set up your mail server

Configure your email server so users can receive a link from Bitbucket Server that lets them generate their own passwords. See [Setting up your mail server](#).

10. Add users and repositories

Now is the time to set up your users in Bitbucket Server, and to tell Bitbucket Server about any existing repositories you have. Please the following pages for the details:

- [Get started with Git](#)
- [Importing code from an existing project](#)

Additional steps for production environments

For production or enterprise environments we recommend that you configure the additional aspects described on [Use Bitbucket in the enterprise](#). The aspects described there are not necessary when you are installing for evaluation purposes only.

If you wish to install Bitbucket Server as a service on Linux or Windows, see either of:

- [Run Bitbucket as a Linux service](#)
- [Running Bitbucket Server as a Windows service](#)

Stopping Bitbucket Server

See [Start and stop Bitbucket](#).

Uninstalling Bitbucket Server

To uninstall Bitbucket Server, stop Bitbucket Server as described above and then delete the `<Bitbucket Server installation directory>` and [Set the home directory](#).

Run the Bitbucket installer

This page provides information about running Bitbucket Data Center and Server with search installer. For high-level information about installing and using Bitbucket see [Getting started](#).

Installers are available for Linux, macOS and Windows operating systems.

The installer will:

- Install Bitbucket into a fresh directory, even if you have an earlier version installed.
- Install a supported version of the Java JRE, which is only available to Bitbucket, if necessary.
- Install a bundled, local instance of Elasticsearch.
- Launch Bitbucket when it finishes.

Additional services provided by the installer, and described on this page, are:

- [Installing Bitbucket Server as a service](#)
- [Running the installer in console and unattended modes](#)

You can also automate the Bitbucket Setup Wizard so that a Bitbucket instance can be completely provisioned automatically see [Automated setup for Bitbucket](#).

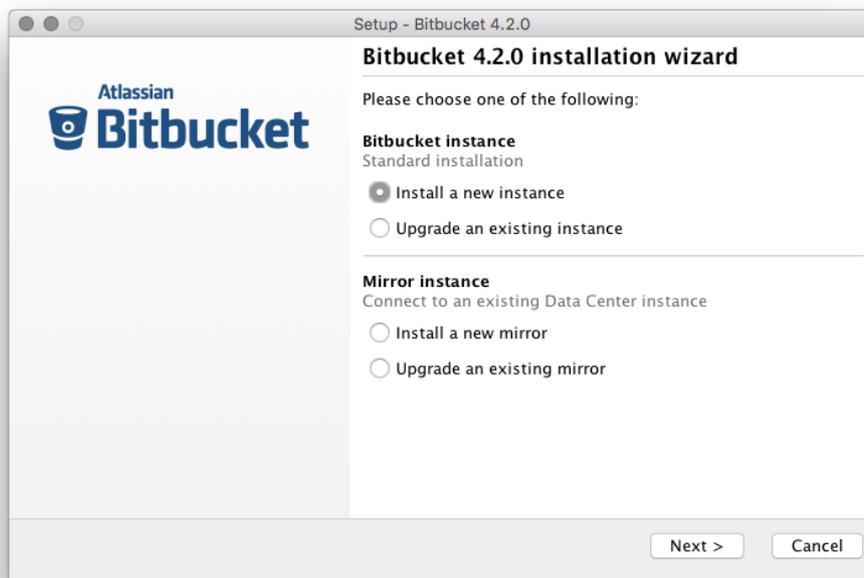
Running the installer

[Download the Bitbucket Server installer](#) from the Atlassian download site.

On Linux, you need to set the executable flag on the installer file before running it:

```
chmod +x atlassian-bitbucket-x.x.x-x64.bin
```

Run the installer, and follow the installation wizard.



Install Bitbucket as a service

On Linux and Windows systems, the installer can install Bitbucket as a service (although not when upgrading an existing instance of Bitbucket).

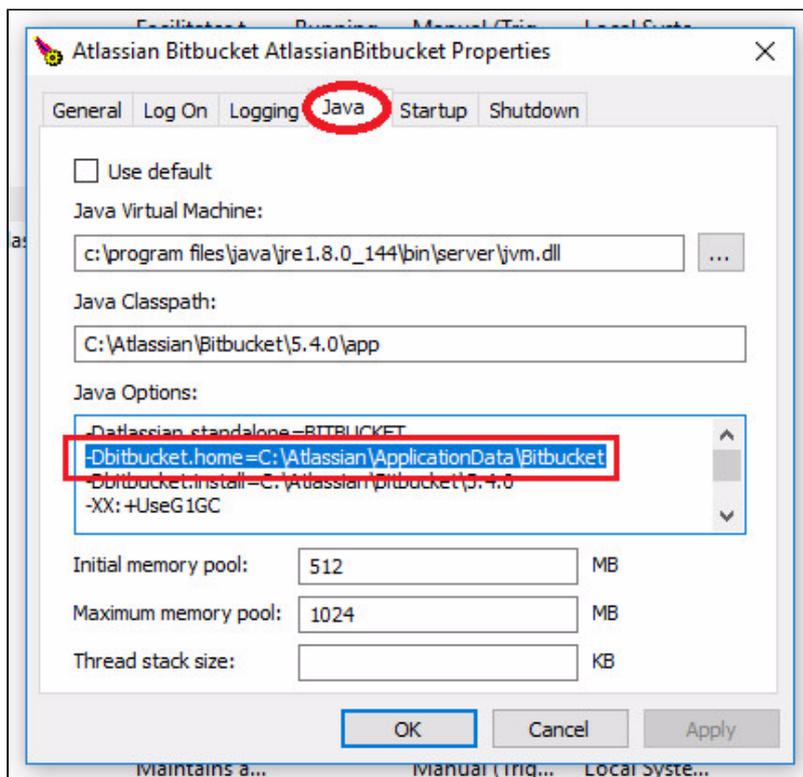
A service account named 'atlassian-bitbucket' will be created.

On Linux

- The 'atlassian-bitbucket' account will be a locked account (it cannot be used to log in to the system).
- The `init.d` script will be linked to run levels 2, 3, 4 and 5. If you wish to change this, you will need to configure it [manually](#).

On Windows (Bitbucket Server)

- The installer generates a password for the service account. As a Windows administrator, you can update the account password if you wish to own the account. You'll also need to update the log on credentials for the service.
- The 'atlassian-bitbucket' account will be configured with `SeServiceLogonRight` so that it can be used by the service. It will also be configured with `SeDenyBatchLogonRight`, `SeDenyInteractiveLogonRight`, `SeDenyNetworkLogonRight`, and `SeDenyRemoteInteractiveLogonRight` so that it cannot be used to log into the machine.
- For Windows services created using the Bitbucket Server installer, the [Bitbucket Server homedirectory](#) location (defined by the `BITBUCKET_HOME` variable) is configured as a Bitbucket Server Service JVM option.
- To change this, go to Service Properties. In the service properties, go to the **Java** tab, and change `bitbucket.home`, as in the below image.



Console and unattended mode

The Bitbucket installer has three modes:

- GUI mode: the default mode for the installer is to display a GUI installer.
- Console mode: if the installer is invoked with the `-c` argument, the interaction with the user is performed in the terminal from which the installer was invoked.
- Unattended mode: if the installer is invoked with the `-q` argument, there is no interaction with the user and the installation is performed automatically with the default values.

Unattended mode also allows you to supply a response file with a `-varfile` option, to supply answers for all questions that are used instead of the defaults. An example response file is:

Example response file

```
// Should Bitbucket Server be installed as a Service? Must be ADMIN (default: true if the process is
running with administrator rights, false otherwise). If false, the home and installation directories must
be specified to point to directories owned by the user
app.install.service$Boolean=true

// The ports Bitbucket Server should bind to (defaults: portChoice=default, httpPort=7990, serverPort=8006)
portChoice=custom
httpPort=7990
serverPort=8006

// Path to the Bitbucket Server HOME directory (default: /var/atlassian/application-data/bitbucket if the
process is running with administrator rights, ~/atlassian/application-data/bitbucket otherwise)
app.bitbucketHome=/var/atlassian/application-data/bitbucket

// The target installation directory (default: /opt/atlassian/bitbucket/<VERSION> if the process is running
with administrator rights, ~/atlassian/bitbucket/<VERSION> otherwise)
app.defaultInstallDir=/opt/atlassian/bitbucket/<VERSION>
```

On Windows, you must tell CMD/PowerShell to wait for the `install4j` process to use `console/unattended` mode:

```
start /wait installer.exe -c
```

On macOS, mount the disk image, then run the Java stub in the installer using this command:

```
/Volumes/Bitbucket Server/Bitbucket Server\ X.X.X\Installer.app/Contents/MacOS/JavaApplicationStub -options
```

where `X.X.X` is the version of Bitbucket, and `-options` can include `-cor`, `-q`, and `-varfile` followed by the path to the response file.

For more information see the [install4j documentation](#).

Further reading

[Use Bitbucket in the enterprise](#)

Bitbucket Server upgrade guide

This guide describes how to upgrade your Bitbucket Server installation to the latest version on Windows or Linux using the installer.

Upgrading to any later version is free if you have current software maintenance. See our [Licensing FAQ](#) to find out more.

Other ways to upgrade Bitbucket:

- [Manually](#) upgrade using an archive file.
- [Data Center](#) upgrade your Data Center cluster
- [Rolling upgrade](#) - upgrade your Data Center cluster to the next bug fix update, without downtime
- [Docker container image for Bitbucket Server](#).

On this page:

Before you begin

Plan your upgrade

1. [Determine your upgrade path](#)
2. [Complete pre-upgrade checks](#)
3. [Upgrade Bitbucket in a test environment](#)

Upgrade Bitbucket

4. [Download Bitbucket Server](#)
5. [Back up your data](#)
6. [Migrate customizations](#)
7. [Stop the application](#)
8. [Run the installer](#)
9. [Start the application](#)

Version-specific update notes

[Check for known issues and troubleshooting](#)

Before you begin

Before you upgrade Bitbucket there are a few questions you need to answer.

| | |
|--|---|
| Is using the installer the right upgrade method for you? | <p>You can choose to upgrade using the installer, or, if the installer is not suitable for your situation you can use an archive file to update your Bitbucket Server instance. Only use an archive file to update Bitbucket Server if the installer is not suitable for your situation, as the installer is the recommended approach to upgrade for most use cases.</p> <p>Read about how to upgrade Bitbucket Server from an archive file.</p> |
| Are you eligible to upgrade? | <p>To check if software maintenance is current for your license, go to  > Licensing and make sure the license support period has not expired.</p> <p>If your support period has expired, follow the prompts to renew your license and reapply it before upgrading.</p> |
| Have our support platforms changed? | <p>Check the Supported platforms page for the version of Bitbucket you are upgrading to. This will give you info on supported operating systems, databases and browsers.</p> |
| Have you modified your <code>server.xml</code> file? | <p>In Bitbucket 5 we changed where you configure properties such as port numbers, context paths, and access protocol (among other things) to centralize custom configuration to a single directory and file, and to make upgrading easier for future releases.</p> <p>Upgrading from any version earlier than or including Bitbucket Server 4.14 to Bitbucket Server 5.0 or later requires that you manually migrate any changes to the <code>server.xml</code> file to the <code>bitbucket.properties</code> file.</p> <p>See the page Migrate customizations from <code>server.xml</code> to <code>bitbucket.properties</code> for instructions on locating your <code>server.xml</code> file and how to migrate your customizations.</p> |

Plan your upgrade

1. Determine your upgrade path

Check the [Supported platforms](#) page to determine if your environment meets the minimum requirements to run the latest version of Bitbucket. Also read the [End of support announcements](#).

You can update from any previous version of Bitbucket Server (or Stash) to the latest version, as there is no required upgrade path.

Bitbucket 7 is a major upgrade

Be sure to read the [Bitbucket Server 7.0 release notes](#), take a full backup, and test your upgrade in a non-production environment before upgrading your production site.

Upgrading from a version older than Bitbucket 7 disables all user-installed apps on startup

Be sure to update your own apps and check the [Atlassian Marketplace](#) to ensure 3rd-party apps are compatible with Bitbucket Server 7 before upgrading.

Upgrading to the next bug fix update

If you are upgrading a multi-node Bitbucket cluster to the next bug fix update (for example, from 7.9.0 to 7.9.4), you can [upgrade with no downtime](#).

2. Complete pre-upgrade checks

1. Check the [Version specific upgrade notes](#) for the version you plan to upgrade to (and any in between).
2. Go to  **Support Tools**, then review the Log analyzer for any issues that may need to be resolved.
3. Check the compatibility of your apps with the version you plan to upgrade to.
 - a. Go to  **Manage apps** > **Bitbucket update check**.
 - b. Choose the version you plan to upgrade to then hit **Check**.

If your users rely on particular apps, you may want to wait until they are compatible before upgrading Bitbucket. App vendors generally update their apps very soon after a major release.

Good to know:

- Upgrading from a version older than Bitbucket Server 6 will automatically disable all user-installed apps. You will need to enable your apps after successfully upgrading, regardless of compatibility with Bitbucket Server 6.
- You can disable an app temporarily if it is not yet compatible.

3. Upgrade Bitbucket in a test environment

1. Create a staging copy of your current production environment.
2. Follow the steps below to upgrade your test environment.
3. Test any unsupported apps, customizations and proxy configuration (if possible) before upgrading your production environment.

Upgrade Bitbucket

This upgrade process does not perform an in-place upgrade, but instead installs the new version of Bitbucket Server into a new installation directory. The new instance uses your existing Bitbucket Server home directory and external database.

If necessary, rolling back an upgrade can only be performed by restoring a backup of *both* the Bitbucket Server [home directory](#) and the Bitbucket Server database. Rolling back requires a consistent home directory and database. You can then reinstall the previous version of the application to the installation directory. Read the [Rollback section](#) for more details.

4. Download Bitbucket Server

Download the installer (the recommended approach) for your operating system - www.atlassian.com/software/bitbucket/download.

5. Back up your data

1. Determine which backup strategy to use.

For **Bitbucket Server** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), the [Backup Client](#) (version 2.7 or later), or [DIY Backup](#) (version 2.12 or later) while Bitbucket is running, or just stop Bitbucket and zip up / snapshot the home directory and database and keep them somewhere safe.

For **Bitbucket Data Center** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), [DIY Backup](#), or take snapshots of the shared home directory (on NFS) and database while all nodes are stopped.

For **Bitbucket mirrors**, the home directory doesn't store any persistent state that can't be reconstructed from the primary Bitbucket instance, but you should still make sure you have a backup of at least the important configuration files such as SSL certificate, `server.xml`, `config/ssh-server-keys.pem`, `bitbucket.properties` file, and so on in a safe place. See [How do I back up my mirrors?](#) for more information.

See the article [Data recovery and backups](#) for detailed information and guidance on creating an effective backup strategy.

2. Back up all the data in your [Bitbucket home directory](#) and [external database](#).

6. Migrate customizations

Since Bitbucket 5, custom server configurations have been kept in a different location. Previously, Tomcat container customizations were made in the `server.xml` file, located in either the `<Bitbucket home directory>/shared` directory (from 3.8 to 4.14), or in `<Stash installation directory>/conf` (for 3.7 and earlier).

If you are upgrading from Bitbucket Server 4.14 or earlier, then you must do the the following:

1. First, determine if you've made customizations to your `server.xml` file. You would have made these changes if you modified the [port](#), [context path](#), or the [access protocol](#), or if you are running Bitbucket Server behind a proxy and modified the Connector element.

| Release | Directory |
|--------------------------------|--|
| Stash 3.7 and earlier | <code><Stash installation directory>/conf/server.xml</code> |
| Stash 3.8 and later | <code><Stash home directory>/shared/server.xml</code> |
| Bitbucket Server 4.0 - 4.14 | <code><Bitbucket home directory>/shared/server.xml</code> |
| Bitbucket Server 5.0 and later | N/A, replaced by <code><Bitbucket home directory>/shared/bitbucket.properties</code> |

2. **If you made any customizations to the `server.xml` file**, you must migrate those changes in the `bitbucket.properties` file in your existing Bitbucket home directory. The `bitbucket.properties` file uses a slightly different syntax, so it is not merely a copy and paste procedure.

Read the page [Migrate customizations from `server.xml` to `bitbucket.properties`](#) and follow the instructions to migrate your customizations. Once your changes are migrated, proceed to the [next step](#).

If you did not make any customizations to the `server.xml` file, proceed to the [next step](#).

Bitbucket Data Center instances on version 3.8 or later, the single `server.xml` file in the `<Bitbucket home directory>/shareddirectory` replaces all the copies of `server.xml` located in the `<Bitbucket installation directory>/conf` directories of the cluster nodes using previous versions of Bitbucket Server (or Stash).

7. Stop the application

To stop Bitbucket Server from the interface, use the Bitbucket Server items in the Windows Start menu.

To stop Bitbucket Server from a command prompt, change directories to the `<Bitbucket Server installation directory>` and run this command:

```
bin\stop-bitbucket.bat
```

To stop Bitbucket Server when running as a service, stop the Bitbucket Server service from the Windows services console.

To stop Bitbucket Server from a terminal, change to the `<Bitbucket Server installation directory>` and run:

```
bin/stop-bitbucket.sh
```

Be sure to *stop all nodes* of Bitbucket Data Center, perform the steps to upgrade for a single node first, then repeat the process on each cluster node.

See the page [Start and stop Bitbucket](#) for more detailed instructions.

8. Run the installer

You can run the installer in GUI, [console](#) or [unattended modes](#).

1. Run the installer.

Run the `.exe` file. We recommend using a Windows administrator account.

Change to the directory where you downloaded Bitbucket Server then execute this command to make the installer executable:

```
chmod +x atlassian-bitbucket-x.x.x-x64.bin
```

We recommend using `sudo` to run the installer as this will create a dedicated account to run Bitbucket Server and allow you to run Bitbucket Server as a service.

To use `sudo` to run the installer, execute this command:

```
$ sudo ./atlassian-bitbucket-x.x.x-x64.bin
```

You can also run the installer with root user privileges.

2. Follow the prompts to upgrade Bitbucket Server.
 - a. At the 'Welcome' step, choose the type of instance to upgrade

For a single node (most users), choose the **Upgrade an existing Bitbucket instance** option.

For Bitbucket Data Center instances, choose the **Upgrade an existing Bitbucket instance** option, and repeat the upgrade process for each application node.

For a Bitbucket mirror, choose the **Upgrade an existing mirror** option.

- b. At the 'Select Bitbucket Server Home' step, select your existing [home directory](#).
- c. Proceed with the remaining installer steps.

9. Start the application

When using the installer, there will be an option to start the application (and launch it in a browser) at the end of the installation wizard.

See the page [Start and stop Bitbucket](#) for more detailed instructions, including how to manually start Bitbucket.

Rollback

If necessary, rolling back an update can only be performed by restoring a backup of *both* the Bitbucket Server [home directory](#) and the Bitbucket Server database. Rolling back requires a consistent home directory and database. You can then reinstall the previous version of the application to the installation directory. Never start an older binary against an upgraded home directory.

Version-specific update notes

This section provides specific update notes for each version of Bitbucket Data Center and Server. These notes supplement the primary upgrade instructions above. You should read the relevant sections for each version between your current version of the application and the version you are upgrading to.

Bitbucket Data Center and Server 7.14 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.14 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.13 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.13 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.12 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.12 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.11 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.11 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Server 7.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server and Data Center 7.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Server 7.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server and Data Center 7.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Server 7.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.7 release notes](#).

- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.6 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.6 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.5 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.5 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.4 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.4 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

New Data Center apps check as part of your license upgrade

Upgrading to a Bitbucket Data Center license now comes with a new apps check. Before we apply your new Bitbucket license, we'll show you if any of your installed apps will need to be upgraded to the [Data Center approved version](#).

We've also introduced new status messages in the **Manage apps** section of your admin console, so you can clearly see if any apps will require a new license.

| New status | Host product license | What it means |
|---|----------------------|---|
|  | Evaluation license | The app vendor offers a Data Center version of this app. If you switch to a paid Data Center product license, you'll need to upgrade this app license to Data Center. |
|  | Full license | The app vendor offers a Data Center version of this app. Your Server app license is no longer compatible with the product. This means the app has either stopped working, or functionality has been lost or compromised. You need to add a Data Center license. |

Learn more about [upgrading your Server apps after you move to Data Center](#)

i Build status data migration

When upgrading to Bitbucket Server 7.4, you may notice an increased load on your database and application server. This is normal. Its due to an asynchronous upgrade task that needs to run to migrate your applications build status data.

While the upgrade task is running, your application will be available and able to serve requests, so your team can continue working as usual.

The upgrade task will run automatically around two minutes after your application starts up. In a multi-node cluster, it will run on a node picked by the system. How long it takes will depend on the size of the data that needs to be migrated. For example, for small instances it can take 10 minutes, and for large instances it can take over an hour.



Before upgrading to Bitbucket Server 7.4, you should test HTTP hosting in a staging environment that mirrors your production setup. This especially includes any installed apps, as app-provided servlet filters may be broken by async HTTP.

Bitbucket Server 7.3 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.3 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.2 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.1 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.0 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

 If you are considering upgrading to bitbucket 7.0 or higher and are using version 2.8 or earlier of the `Bitbucket branch source` Jenkins plugin, there is a known issue that can cause builds to fail. This is because merges are performed lazily in order to manage scaling while changes to pull requests happen continuously.

See

 [BSERV-12284](#) - Eagerly update refs/pull-requests/*/from when a pull request's source branch changes CLOSED

Bitbucket Server 6.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.7 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.6 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.6 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.5 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.5 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.4 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.4 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.3 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.3 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.2 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.1 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.0 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

 **To upgrade to Bitbucket Server 5+, you must migrate customizations to bitbucket.properties**

Be sure to carefully read the [Bitbucket Server upgrade guide](#) to determine what impact this change may have on your instance. The page [Migrate server.xml customizations to bitbucket.properties](#) has specifics on which properties need to be migrated, how to migrate them, and includes some examples that demonstrate how to migrate some common customizations.

 **Start script changes**

Starting from Bitbucket Server 5.0, some startup scripts had their name prefixed with an underscore. These scripts have been deprecated, and you should no longer call these scripts directly. This is to avoid confusion, because previously there were multiple start and stop scripts, making it easy to confuse which one to call. Now, there is only one script, that begins with "start" and one that begins with "stop".

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Bitbucket Server 5.16 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.16 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.15 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.15 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.14 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.14 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.13 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.13 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.12 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.12 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.11 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.11 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.7 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.6 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.6 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.5 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.5 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.4 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.4 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.3 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.3 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.2 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.1 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.0 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

⚠ To upgrade to Bitbucket Server 5+, you must migrate customizations to bitbucket.properties

Be sure to carefully read the [Bitbucket Server upgrade guide](#) to determine what impact this change may have on your instance. The page [Migrate server.xml customizations to bitbucket.properties](#) has specifics on which properties need to be migrated, how to migrate them, and includes some examples that demonstrate how to migrate some common customizations.

⚠ Start script changes

In Bitbucket Server 5.0, some startup scripts have had their name prefixed with an underscore. These scripts have been deprecated, and you should no longer call these scripts directly. This is to avoid confusion, because previously there were multiple start and stop scripts, making it easy to confuse which one to call. Now, there is only one script, that begins with "start" and one that begins with "stop".

❗ BitbucketServer does not support reverse proxies using AJP connections

Starting from version 5.0, Bitbucket Server does not support reverse proxies using AJP connections. While never officially supported, previous versions of Bitbucket Server could use AJP connections. However, with changes to how Bitbucket Server uses Tomcat, using AJP connections with Bitbucket Server (or Data Center) 5.0+ will cause the application to fail on startup.

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.14 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.14 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Known issues

P Key Summary Status

No issues found

Bitbucket Server 4.13 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.13 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Bitbucket Server 4.12 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.12 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

 **Do NOT upgrade to Git 2.11+**

Any version after Git 2.11.0 (including future 2.12+ releases) cannot be used with Bitbucket Server. Bitbucket Server 4.12 will fail on startup if Git 2.11+ is detected. Only upgrade to versions of Git which are explicitly marked supported on our [Supported Platforms](#) page.

 **BSERV-9388** - Pre-receive hooks and branch permissions reject valid pushes on git 2.11

CLOSED

Known issues

| P | Key | Summary | Status |
|---|------------|---|-------------------|
| ▼ | BSERV-9592 | Spotted Typo on Analytics page on Bitbucket Server Settings | GATHERING IMPACT |
| ▼ | BSERV-8201 | Repository deletion times out | GATHERING IMPACT |
| ▼ | BSERV-9675 | Duplicated branches and tags - Multiple entries for the same ref id | LONG TERM BACKLOG |

3 issues

Bitbucket Server 4.11 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.11 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Bitbucket Server 4.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Bitbucket Server 4.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Bitbucket Server 4.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.7 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.6 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.6 release notes](#).
- the Stash [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.5 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.5 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.4 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.4 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Bitbucket Server 4.3 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.3 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Bitbucket Server 4.2 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Bitbucket Server 4.1 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Bitbucket Server 4.0 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#). Note that Bitbucket Server 4.0 does not support Internet Explorer 9 or 10, Java 7, nor versions of Git earlier than 1.8 on the server.

Do you rely on user-installed add-ons?



Your plugins may not be compatible with Bitbucket Server 4.0 (yet)

Upgrading from Stash to Bitbucket Server will disable all user-installed add-ons, whether they are 3rd-party add-ons from Atlassian Marketplace, or your own custom add-ons. User-installed add-ons will need to be checked, upgraded and enabled again once you've finished the upgrade process. You should check the [marketplace listing](#) for 3rd-party add-ons to determine if there is a release that is compatible with Bitbucket Server 4.0. Unless they have been updated to work with Bitbucket Server 4.0, existing add-ons (or plugins) that use the API interfaces that have been removed in Bitbucket Server 4.0 *will not work*. Similarly, custom add-ons need to be upgraded for Bitbucket Server compatibility. See the documentation on [how to update your add-on](#) for details.

If you are upgrading from Stash 3.x to Bitbucket Server 4.x, you should be aware that most user-installed add-ons will be incompatible with Bitbucket Server 4.0. After upgrading, go to **Admin > Manage add-ons**, look for messages of this form, and follow the advice to update:



A newer version of the Universal Plugin Manager is available. [Update Now](#)

[Skip this version](#) [Remind me later](#)

If no newer version is available the add-on will remain disabled.

New dedicated user account 'atlbitbucket'

A new dedicated user account will be created to run Bitbucket Server, called `atlbitbucket`. During the upgrade process, you will have an option to delete the dedicated user account for running Stash, named `atlstash`. For most users, deleting this dedicated user account won't have any negative consequences, however if you rely on this dedicated user accounts, for example in custom backup scripts, you will need to update the user account in those scripts.

Updates to logger names

If you have customized your logging configuration by manually editing the `logback.xml` file (using steps found in the [Configure Stash Logging KB article](#)), you should be aware of changes to several logger names that may require you to update some of your configuration files.

| Stash logger name | Bitbucket Server logger name |
|--------------------------------|------------------------------------|
| <code>stash.application</code> | <code>bitbucket.application</code> |
| <code>stash.profiler</code> | <code>bitbucket.profiler</code> |
| <code>stash.access-log</code> | <code>bitbucket.access-log</code> |
| <code>stash.audit-log</code> | <code>bitbucket.audit-log</code> |
| <code>stash.mail-log</code> | <code>bitbucket.mail-log</code> |

Important security improvement which may require configuration changes

Previous to Bitbucket Server 4.0, a security constraint for redirecting from HTTP to HTTPS was not enforced, meaning users could type "http://<bitbucketserver-url>" into their browser and still be shown a functioning version of Bitbucket Server (or Stash). Included with the release of Bitbucket Server 4.0 was a fix to enforce that security constraint. Using the previous security configuration with Bitbucket Server 4.0 means trying to access the application over an insecure connection, when users type "http" when trying to get to the application, they could encounter erroneous behavior.

As part of upgrading from Stash to Bitbucket Server, it is recommended you update your security configuration in order to avoid encountering this problem. See this Knowledge Base article for more information and specific instructions on addressing this: [Redirect HTTP Requests to HTTPS](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Stash 3.11 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.11 release notes](#).
- the [Stash Supported platforms](#) page. Note that Stash 3.11 [does not support Git 2.0.2 or 2.0.3](#).
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9 or 10, Java 7, nor versions of Git earlier than 1.8 on the server.
- If you have installed the **stash-stream-guard-plugin** (for example, if advised to do so by Atlassian Support), then you should uninstall it when upgrading to 3.11.2 as its functionality is now bundled in Stash.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Stash 3.10 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.10 release notes](#).
- the [Stash Supported platforms](#) page. Note that Stash 3.10 [does not support Git 2.0.2 or 2.0.3](#).
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9 or 10, Java 7, nor versions of Git earlier than 1.8 on the server.

Are you using Git 2.2x - 2.4.0?

Git 2.2.x - 2.4.0 have reported performance issues when interacting with NFS. Hence, these versions are recurrently not supported for Stash Data Center or for Stash Server installations that use NFS mounts for the home directory.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 3.9 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.9 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.9 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Are you using JIRA as a Crowd server?

Stash 3.9 cannot use JIRA 6.4 as a Crowd server due to a bug in JIRA 6.4. Please upgrade to JIRA 6.4.1 before upgrading Stash. (Note that Stash 3.7.2 and 3.8.0 contained a workaround for the bug but the workaround was removed in Stash 3.9)

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 3.8 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.8 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.8 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Have you made customizations to Tomcat's `server.xml` file?

For Stash 3.8 (and future versions) the `server.xml` file is now located in the `<stash home directory>/shared` directory. The benefit of this move is that your customizations to `server.xml` will not have to be redone for future upgrades.

 You still need to update your custom configurations in `shared/server.xml` for the upgrade to Stash 3.8.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 3.7 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.7 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.7 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Do you use JIRA 6.4?



JIRA 6.4 has a known issue with Stash versions 3.4.3 to 3.7.1, which prevents the user synchronization from working. This will only affect you if you use JIRA to manage your users in Stash, and is fixed in Stash 3.7.2.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Stash 3.6 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.6 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.6 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Secure email notifications

Stash 3.6 now:

- Allows you to require STARTTLS support on the mail server when using SMTP. If it is not supported, mail is not sent.
- Supports SMTPS (where the whole protocol conversation uses SSL/TLS).

See [Setting up your mail server](#) for more information.

Note that if you use either SMTP with STARTTLS or SMTPS and connect to a self-signed mail server you may need to import the server's certificate and set up a custom cacerts file for Stash (just as you do for any outbound SSL/TLS connection to a self-signed server). See this [Stash knowledge base article](#) for information about how to do that.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Stash 3.5 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.5 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.5 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Java 7, nor versions of Git earlier than 1.8 on the server.

Stash home directory location

Stash 3.5 and later versions no longer allow the [Set the home directory](#) to be the same directory as, or a subdirectory of, the Stash installation directory. The Stash home directory, as defined by the `STASH_HOME` environment variable, must be in a separate location Stash will fail on startup otherwise.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine in Stash has been changed from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash *is now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

| priority | key | summary | status |
|----------|-----|---------|--------|
|----------|-----|---------|--------|

 Jira project doesn't exist or you don't have permission to view it.

[View these issues in Jira](#)

Stash 3.4 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.4 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.4 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#).

Changed group membership aggregation with multiple user directories

Stash 3.4 uses new schemes to determine the effective group memberships when Stash is connected to multiple user directories, and duplicate user names and group names are used across those directories. The new schemes are:

- 'aggregating membership'
- 'non-aggregating membership'.

These group membership schemes are only used to determine authorization. Authentication is determined on the basis of the group mappings in each directory.

See [Effective memberships with multiple directories](#) in the Crowd documentation for more information.

When you update to Stash 3.4, an update task will apply one of those schemes as follows:

Aggregating membership will be applied to your instance:

- If there is only one active directory.
- If there are multiple active directories but only a single directory applies group memberships to any particular user.
- For example, if directory-1 contains user-a in group-x, and directory-2 contains user-b in group-y, then Stash 3.4 will apply aggregating membership, and permissions will not be affected.

Non-aggregating membership will be applied to your instance:

- If there are multiple active directories and more than one directory applies group memberships to at least one user.
- For example, if directory-1 contains user-a in group-x, and directory-2 contains user-a in group-y, then Stash 3.4 will apply non-aggregating membership. If group-y has admin privileges then user-a would have their privileges escalated in this case if aggregating membership was applied instead when upgrading to Stash 3.4.

Any changes made by the update task will be logged.

A Stash admin can change the membership scheme used by Stash using the following commands:

- To change to *aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d '{"membershipAggregationEnabled":true}' -u <username>:<password> <base-url>/rest/crowd/latest/application
```

- To change to *non-aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d '{"membershipAggregationEnabled":false}' -u <username>:<password> <base-url>/rest/crowd/latest/application
```

Note that changing the aggregation scheme can affect the authorization permissions for your Stash users, and how update operations are performed. Read more about using Stash with [multiple use directories](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Stash 3.3 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.3 release notes](#).
- the [Stash Supported platforms](#) page. Note that Stash 3.3 does not support Git 2.0.2 or 2.0.3.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Stash 3.2 update notes

 Upgrading to Stash 3.2 or later from Stash 3.1 or earlier is irreversible please see the [Home directory migration](#) section below.

Also see:

- the [general update steps](#) section above.
- the [Stash 3.2 release notes](#).
- the [Stash Supported platforms](#) page.
- the [End of support announcements](#).

Note that:

- Stash 3.2 [does not support Git 2.0.2 or 2.0.3](#).

Home directory migration

When you update to Stash 3.2 or later, an update task migrates directories in your Stash home directory to new locations. This is irreversible once you update to Stash 3.2 or later you can not revert to Stash 3.1 or earlier, because of changes to the Stash home directory format.

For most installations, Stash 3.2 is able to perform these moves automatically and transparently. However, in the rare instance where Stash 3.2 can't perform the update automatically, please refer to [Upgrading your Stash home directory for Stash 3.2 manually](#).

Undocumented home directory overrides are no longer supported

Before Stash 3.2 it was possible to override the location of the following subfolders in the Stash home directory, using undocumented environment variables or system properties:

- export
- bin
- caches
- config

- data
- lib
- lib/native
- log

- plugins
- tmp

In Stash 3.2 only the `tmp` subfolder can be overridden in this way. Attempting to override the others will fail on startup. For more information, please see [Stash fails to start - UnsupportedDirectoryOverrideException](#).

Stash analytics

Stash 3.2, and later, collects user event data, unless this is disabled by an administrator. You will see outgoing network traffic to [amazonaws.com](#). See [Change data collection settings](#).

Known issues

P Key Summary Status

No issues found

Stash 3.1 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.1 release notes](#).
- the Stash [Supported platforms](#) page.

Note that:

- Stash [does not support Git 2.0.2 or 2.0.3](#).

Known issues

P Key Summary Status

No issues found

Stash 3.0 update notes



Stash no longer supports Java 6

Stash 3.0 requires at least Java 7, and supports Java 8.

Please see the [End of support announcements](#).



Your plugins may not be compatible with Stash 3.0

The interfaces in the Stash API for plugin developers that *were deprecated* in Stash 2.11 and earlier *have been removed* in Stash 3.0. This means that, unless they have been updated to work with Stash 3.0, existing Stash add-ons (or plugins) that use these interfaces *will not work with Stash 3.0*.

Please see the section below about Stash add-on incompatibilities for more details.

Also see:

- the general [update steps](#) section above.
- the [Stash 3.0 release notes](#).
- the Stash [Supported platforms](#) page.

Note that:

- Stash now has installers for the Linux, macOS and Windows platforms.
- Stash no longer supports Internet Explorer 8, as [previously announced](#).
- Stash does not support the Apache HTTP Server `mod_auth_basic` module.
- Stash [does not support Git 2.0.2 or 2.0.3](#).

Stash add-on incompatibilities

Unless they have been updated to work with Stash 3.0, existing Stash add-ons (or plugins) that use the API interfaces that have been removed in Stash 3.0 *will not work*.

Fresh installs of Stash 3.0 shouldn't encounter any problems. The Stash 'Manage add-ons' page (in the admin area) should only display add-ons from the Marketplace that have been marked as compatible with Stash 3.0. Incompatible add-ons won't be available in the list.

However, if you are upgrading from Stash 2.x to Stash 3.0, you should be aware that some existing installed add-ons may be incompatible with Stash 3.0. After upgrading, you should go to **Admin>Manage add-ons**, look for messages of this form, and follow the advice to update:

 A newer version of the Universal Plugin Manager is available. [Update Now](#) [Skip this version](#) [Remind me later](#)

If no newer version is available, the add-on must be disabled.

Custom add-ons

Please note that your custom locally-developed plugins may be affected by the API removals in Stash 3.0. You will need to update your custom plugins if you want those to work with Stash 3.0. See the [Stash API changelog](#) for details of the deprecated APIs.

Third-party add-ons

You'll need to check on Atlassian Marketplace for the compatibility status of any 3rd-party add-ons that you use.

Third-party add-on developers have been given an Early Access Program (EAP) build of Stash 3.0 in advance of release, and many have already updated their add-ons to be compatible. Add-ons must be explicitly marked by the publisher as compatible with Stash 3.0 for them to appear in 'Manage add-ons' page in Stash. ***This is NOT automatic as was the case with previous minor releases such as 2.10 and 2.11.*** Atlassian can not support issues involving third party Add-ons that are incompatible with Stash 3.0; such support cases must be directed to the third-party publisher of the add-on. See [Managing apps](#).

Atlassian add-ons

All of the Atlassian add-ons for Stash that are available from the Atlassian Marketplace have been updated to be compatible with Stash 3.0. If you use any of these in your Stash installation you'll need to update them to the Stash 3.0 compatible version.

| Add-on | JAR file name | Stash 3.0 compatible version |
|------------------------------------|-------------------------------------|------------------------------|
| Custom Navigation Plugin | custom-navigation-plugin | 2.0.3 |
| Stash Archive Plugin | stash-archive | 1.3.0 |
| Repository git operations plugin | stash-git-ops-plugin | 1.2.1 |
| Stash Auto Unapprove Plugin | stash-auto-unapprove-plugin | 1.1 |
| Stash Protect Unmerged Branch Hook | stash-protect-unmerged-branch-hook | 1.1 |
| Stash Reviewer Suggester | stash-suggest-reviewers | 1.2 |
| Stash Web Post Hooks Plugin | stash-web-post-receive-hooks-plugin | 1.1.0 |
| Realtime Editor for Stash | stash-editor-plugin | 1.0.6 |

Known issues

P Key Summary Status

[No issues found](#)

Stash 2.12 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#). In particular, note that support for Java 6 is deprecated. Stash 3.0+ will require Java 7.
- the [Stash 2.12 release notes](#).

Note that:

- Stash does not yet support Java 8.
- Stash 2.12 does not support Git 1.8.4.3
- Stash does not support the Apache HTTP Server `mod_auth_basic` module.

See [Supported platforms](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.11 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.11 release notes](#).

Note that Stash does not support Git 1.8.4.3, nor does Stash support Java 8 yet. See [Supported platforms](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.10 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.10 release notes](#).

Note that Stash does not support Git 1.8.4.3. See [Supported platforms](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.9 update notes

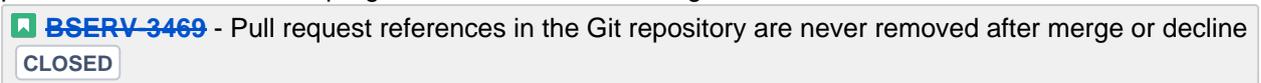
Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.9 release notes](#).

Note that Stash does not support Git 1.8.4.3. See [Supported platforms](#).

Pull Request Ref Optimization

When you first start Stash after upgrading to Stash 2.9 a repository update task runs that optimizes the pull request refs for all repositories managed by Stash. It's important that you do not interrupt this update process. You can track the progress of this in the Stash logs. See



Backup Client Update Required

Version 1.0.3 of the [Stash Backup Client](#) is required to back up Stash 2.9.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.8 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.8 release notes](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.7 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Repository System Information Plugin is now deprecated

The functionality of the [repository system information plugin](#) has now been moved into core Stash. The plugin will still work for Stash 2.x versions but is redundant as of Stash 2.7.

MySQL default isolation level

Stash 2.7.x uses `READ_COMMITTED` instead of the MySQL default isolation level (`REPEATABLE_READ`). This can result in exceptions when installing or upgrading to 2.7.x, if [binary logging](#) is enabled in your MySQL server. More details and a fix can be found in [this KB article](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.6 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.5 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Limited support for JIRA 4.4.x and earlier

JIRA 4.3+ allows for showing commits associated with issues in JIRA. However, viewing issues within Stash is not supported for JIRA 4.4.x and earlier.

Stash 2.4 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.3 update notes

Please also see the [update steps](#) section above.

When upgrading to Stash 2.3 you also need to update the SCMCache plugin, due to recent Stash API changes.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.2 update notes

Please see the [update steps](#) section above.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Stash 2.1 update notes

Please also see the [update steps](#) section above.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

[No issues found](#)

Install location for third-party libraries

As of Stash 2.1 you can install third-party libraries and jar files, such as the MySQL JDBC driver, into `<stash home directory>/lib`. This has the advantage that files in this location are not overwritten, and lost, when you update Stash.

Microsoft SQL Server JDBC driver

Stash 2.1 now uses the Microsoft SQL Server JDBC driver to access Microsoft SQL Server, instead of using the jTDS driver. In most cases, Stash will automatically swap to using the Microsoft driver on update and **no configuration is required**.

If Stash was configured to use Microsoft SQL Server by manually entering a JDBC URL, please refer to [this guide](#).

Stash 2.0 update notes

This section provides specific notes for upgrading to Stash 2.0. See also the [update steps](#) section above.

Tomcat

For Stash 2.0, Tomcat has been updated from version 6 to 7. As part of that update, the `server.xml` file has changed. If you have customized `server.xml` (for example, for `port`, `path` or `hostname`), you can not simply copy your own version across to the updated Stash; you must reapply your customizations to the `server.xml` file for the new version of Stash.

If you were running Stash as a Windows service and are upgrading from 1.x to 2.x you will need to reinstall the Stash service to make it use Tomcat 7.

To uninstall the Stash service you need to execute following commands from `<STASH DISTRIBUTION DIR>\bin`:

```
> net stop <service name>
> service.bat uninstall <service name>
```

You can call this command without the service name if you installed the Stash service with a default name.

After the service is uninstalled you can proceed with the [update steps](#) and [Running Bitbucket Server as a Windows service](#) instructions to configure Stash 2.x running as a service.

Perl

Stash 2.0 requires Perl for its branch permission functionality. If Perl is unavailable, Stash 2.0 will not start.

On Windows machines, Perl will only have been installed by the Git installer if the correct install option was chosen. See [installing and upgrading Git](#).

Existing Git hooks

In order to support Branch Permissions, Stash 2.0 moves existing hooks in the `pre-receive` and `post-receive` folders under `<Stash home directory>/data/repositories/NNNN/hooks` (where `NNN` is the internal repository id) to `.../hooks/pre-receive.d/10_custom` or `.../hooks/post-receive.d/10_custom`. Consequently, custom hooks that use relative path names (e.g. `./foo.sh` or `./dir/foo.sh`) will be broken by the update to Stash 2.0.

Deprecation of Internet Explorer 8

Support for Internet Explorer 8 is deprecated from the release of Stash 2.0. The official end-of-support date is yet to be determined. See [Supported platforms](#) for details.

Known issues

| P | Key | Summary | Status |
|---|-----|---------|--------|
|---|-----|---------|--------|

No issues found

Check for known issues and troubleshooting

If something is not working correctly after you have completed the steps above to update your Stash installation, please check for known Stash issues and try troubleshooting your update as described below:

- **Check for known issues.** Known issues can be seen in the [BSERV project on our issue tracker](#).
- **Bitbucket Server Knowledge Base.** Sometimes we find out about a problem with the latest version of Stash after we have released the software. In such cases we publish information in the [Bitbucket Server Knowledge Base](#).
- If you encounter a problem during the update and cannot solve it, please create a [support ticket](#) and one of our support engineers will help you.

How to update your add-on

This document and referenced resources are here to help reduce the time and effort needed on your part to make your app compatible with Bitbucket 4.0.

 Version numbering for Stash (3.x and earlier) has been left intact; for example, if you're running Stash 3.10.0, you can use app versions listed as compatible with Bitbucket Server 3.10.0.

On this page

- [About the changes](#)
- [How to update your app](#)
- [Other helpful resources](#)
- [Outline of API changes](#)
- [Outline of changes made to the REST API](#)
- [XSRF Protection enabled by default](#)
- [Javascript Events](#)
- [App update strategies](#)
- [Rename checklist](#)

About the changes

Bitbucket Server 4.0 is the biggest API release Stash has ever seen. The decision to break so many apps with this release was not taken lightly. As developers, we understand the friction such changes can cause. However, we strongly believe this short term pain will be for long term gain with a clearer, cleaner, more consistent and more robust API in 4.0 and beyond. We are making quite a few large changes in this release in the interest of consistency, and with the strong hope that we will not need to make such a drastic change again anytime soon.

Backend changes

One of the exciting changes with this release is the new **JDK minimum requirement of 1.8**, that allows you to use a wide new array of language features in your app code. *While Stash 3.x supports running on Java 8; Bitbucket Server supports compiling to Java 8.*

The largest change was **renamed our package namespace** from `com.atlassian.stash` to `com.atlassian.bitbucket`, but this should also be a simple change when you update your plugins. The Bitbucket Server team updated over 100 apps internally and the process was quite straightforward using refactoring support in modern IDEs.

In addition to the repackage, we've also **removed all @deprecated APIs from the codebase**. Most of these had existing replacement methods in place, but some were removed without replacement. You can consult the [latest Stash 3.x documentation](#) for details on what replacement method to use if the new method is not obvious.

Finally, several of our bundled plugins were exporting API (our `com.atlassian.stash:stash-build-integration` plugin, for example), which meant plugin developers added dependencies on that jar. However, only a small portion of the code in that jar was exported. This was a frequent source of plugin issues because plugin developers attempted to use our internal classes. In 4.0, the **exported APIs from all of our plugins have been extracted into separate modules** (like with the `stash-build-integration` example, the build API is now in `com.atlassian.bitbucket.server:bitbucket-build-api`). These new API modules contain *all* of the code that is published for plugin developers to use.

Front-end changes

Our Javascript and Soy API modules have moved to the Bitbucket namespace. AMD Modules, previously found under `stash/api/*`, are now `bitbucket/*`. Non-API modules will be under `bitbucket/internal`. For example, `stash/api/util/navbuilder` is now `bitbucket/util/navbuilder`. For API Soy templates, these are now also under the bitbucket namespace - `Stash.template.branchSelector` is now `bitbucket.component.branchSelector`.

Another front-end change is that most keys including Form Fragments, Web Panel & Section locations, and Web Resource apps have been moved to Bitbucket namespaces. There is more detail on these changes below.

Any methods or modules that were deprecated for removal in 4.0 have been removed.

How to update your app

The appkey of your plugin must not change

The value of your appkey, as defined within the `atlassian-plugin.xml` file, must not change. This is often defined in terms of the maven `groupId` and `artifactId` of your plugin. This value is used by Atlassian Marketplace and other parts of Bitbucket Server as a unique identifier for your plugin. Changing this value could lead to data loss, and will cause you to lose review and download history on the Atlassian Marketplace.

For example, you would replace the variables below with the values for `groupId` and `artifactId` in your `pom.xml`.

```
<atlassian-plugin key="${project.groupId}.${project.artifactId}" ...
```

1. Update your `pom.xml` file to reference the latest version of the Bitbucket Server 4.0. You will need to update version properties for both Bitbucket Server and AMPS, which currently requires a pre-release version to build Bitbucket Server plugins, as well as dependencies on any API artifacts.

```
<dependency>
  <groupId>com.atlassian.bitbucket.server</groupId>
  <artifactId>bitbucket-api</artifactId>
  <version>${bitbucket.version}</version>
  <scope>provided</scope>
</dependency>
...
<properties>
  <bitbucket.version>4.0.0</bitbucket.version>
  <bitbucket.data.version>${bitbucket.version}</bitbucket.data.version>
  <amps.version>6.1.0</amps.version>
  ...
</properties>
```

2. In your `pom.xml` you will also need to change or add configuration for the `bitbucket-maven-plugin` (depending on whether you are supporting both Stash and Bitbucket Server, or just Bitbucket Server):

```
<build>
  <plugins>
    <plugin>
      <groupId>com.atlassian.maven.plugins</groupId>
      <artifactId>bitbucket-maven-plugin</artifactId>
      <version>${amps.version}</version>
      <extensions>true</extensions>
      <configuration>
        <products>
          <product>
            <id>bitbucket</id>
            <instanceId>bitbucket</instanceId>
            <version>${bitbucket.version}</version>
            <dataVersion>${bitbucket.data.version}</dataVersion>
          </product>
        </products>
      </configuration>
    </plugin>
  </plugins>
</build>
```

3. Update your `appName` and `description` in `pom.xml` to reference "Bitbucket" instead of "Stash". For example:

```
<name>Bitbucket Server - Realtime Editor</name>
<description>Provides support for real-time collaborative editing.</description>
```

4. Optional: If your plugin will only support Bitbucket Server, remove any Stash dependencies

```
<groupId>com.atlassian.stash</groupId>
<artifactId>stash-api</artifactId>
```

5. For a class with compilation errors, first remove any `com.atlassian.stash` import statements that are red.
6. Use the suggested imports your IDE provides, and/or consult the API Changelog and table below.
7. Open the `atlassian-plugin.xml` inside your IDE
 - a. Rename any `com.atlassian.stash` imported components to `com.atlassian.bitbucket` (or equivalent as mentioned in the API changelog)
 - b. If you are using any web-resources with a dependency on `com.atlassian.stash.stash-web-api`, change them to `com.atlassian.bitbucket.server.bitbucket-web-api`
 - c. Check for any other changes in your resources required due to renamed frontend API
8. If your app has JavaScript which uses the Stash JavaScript API, change your AMD module imports from `stash/api/*` to `bitbucket/*`
9. Test the app starts in Bitbucket Server using:

```
mvn clean bitbucket:debug
```

Other helpful resources

- [Bitbucket Server developer documentation](#).
- Ask a question on [Atlassian Answers](#), using the "bitbucket-server" topic.
- Review the [End of support announcements](#) to ensure that you're aware of updates to this policy.

Outline of API changes

Java packages

Stash 3.x

`com.atlassian.stash`

Bitbucket 4.x

`com.atlassian.bitbucket`

App Key (in `atlassian-plugin.xml`)

Stash 3.x

```
<atlassian-plugin key="{project.groupId}.{project.artifactId}" ...>
```

or

```
<atlassian-plugin key="com.myorg.stash.awesome-plugin" ...>
```

Bitbucket 4.x

```
<atlassian-plugin key="com.myorg.stash.awesome-plugin" ...>
```

(must not change in either case)

```
<atlassian-plugin key="com.myorg.stash.awesome-plugin" ...>
```

Maven plugin

Stash 3.x

```
<artifactId>maven-stash-plugin</artifactId>
```

e.g.

```
<plugin>
<groupId>com.atlassian.maven.plugins</groupId>
<artifactId>maven-stash-plugin</artifactId>
<version>${amps.version}</version>
<extensions>>true</extensions>
<configuration>
  <products>
    <product>
      <id>stash</id>
      <instanceId>stash</instanceId>
      <version>${stash.version}</version>
      <dataVersion>${stash.data.version}</dataVersion>
    </product>
  </products>
</configuration>
</plugin>
```

Bitbucket 4.x

```
<artifactId>bitbucket-maven-plugin</artifactId>
```

e.g.

```
<plugin>
<groupId>com.atlassian.maven.plugins</groupId>
<artifactId>bitbucket-maven-plugin</artifactId>
<version>6.1.0</version>
<extensions>>true</extensions>
<configuration>
  <products>
    <product>
      <id>bitbucket</id>
      <instanceId>bitbucket</instanceId>
      <version>${bitbucket.version}</version>
      <dataVersion>${bitbucket.version}</dataVersion>
    </product>
  </products>
</configuration>
</plugin>
```

Exceptions

Stash 3.x

```
com.atlassian.stash.exception.ServiceException
```

Bitbucket4.x

```
com.atlassian.bitbucket.ServiceException
```

The monolithic `com.atlassian.stash.exception` package has been removed. The exceptions it previously contained have been moved into the module they belong to. For example, `NoSuchRepositoryException` is now in the `com.atlassian.bitbucket.repository` package.

Java User model

Stash 3.x

```
com.atlassian.stash.user.StashUser
```

Bitbucket 4.x

`com.atlassian.bitbucket.user.ApplicationUser`

Java Authentication Context

Stash 3.x

`com.atlassian.stash.user.StashAuthenticationContext`

Bitbucket 4.x

`com.atlassian.bitbucket.auth.AuthenticationContext`

Java Event model

Stash 3.x

`com.atlassian.stash.event.StashEvent`

Bitbucket 4.x

`com.atlassian.bitbucket.event.ApplicationEvent`

The monolithic `com.atlassian.stash.event` package has been broken down and the events it formerly contained have been moved into subpackages. For example, `RepositoryCreatedEvent` is now in `com.atlassian.bitbucket.event.repository`

Java model

Stash 3.x

`Changeset`, `DetailedChangeset`

Bitbucket 4.x

`Commit`, `Changeset`

We've standardized our naming:

- A "commit" is an event where the contents of a repository are changed
- A "changeset" is the set of changes that exist *between two commits*
- The codebase no longer uses the words "changeset" and "commit" interchangeably; each word refers to a specific concept

All classes and interfaces with `Changeset` in the name have replaced with an equivalent class or interface with `Commit` in the name instead.

What was formerly `aDetailedChangeset`, because the object that should have been called `aCommit` had already stolen the `Changeset` name, is now called `aChangeset`

Java Pull Request Participant model

Stash 3.x

`PullRequestParticipantSearchRequest`

`PullRequestParticipantSearchCriteria`

Bitbucket 4.x

`PullRequestParticipantRequest`

`PullRequestParticipantCriteria`

Soy templates

Stash 3.x

changeset

Bitbucket 4.x

commit

Application constants**Stash 3.x**

com.atlassian.stash.Product

- #NAME="Stash"
- #DATA_CENTER_NAME="Stash Data Center"
- #FULL_NAME="Atlassian Stash"

Bitbucket 4.x

com.atlassian.bitbucket.Product

- #NAME="Bitbucket "
- #DATA_CENTER_NAME="Bitbucket Data Center"
- #FULL_NAME="Atlassian Bitbucket"

License changes**Stash 3.x**

com.atlassian.extras.api.stash.StashLicense

Bitbucket 4.x

com.atlassian.extras.api.bitbucket.BitbucketServerLicense

Add this dependency to your POM:

```
<dependency>
  <groupId>com.atlassian.extras</groupId>
  <artifactId>atlassian-extras-api</artifactId>
  <scope>provided</scope>
</dependency>
```

Javascript API modules**Stash 3.x**

stash/api/* (eg. stash/api/util/server)

Bitbucket 4.x

bitbucket/* (eg. bitbucket/util/server)

Soy API namespaces**Stash 3.x**

stash.template.branchSelector

Bitbucket 4.x

bitbucket.component.branchSelector

Web API plugin module

Stash 3.x

```
com.atlassian.stash.stash-web-api
```

Bitbucket 4.x

```
com.atlassian.bitbucket.server.bitbucket-web-api
```

Note there have been a number of new API resources added, which allow you to better express your web resources dependencies. Please see the [updated Web UI API documentation](#) for more detail about these resources.

Core web plugin module

Stash 3.x

```
com.atlassian.stash.stash-web-plugin
```

Bitbucket 4.x

This core plugin contains internal modules only and should not be referenced by other plugins.

Web Panel & Section Locations

Stash 3.x

```
stash.*
```

```
e.g. stash.branch.list.actions.dropdown
```

Bitbucket 4.x

```
bitbucket.*
```

```
e.g. bitbucket.branch.list.actions.dropdown
```

Web Resource Contexts

Stash 3.x

```
stash.*
```

```
e.g. stash.layout.pullRequest
```

Bitbucket 4.x

```
bitbucket.*
```

```
bitbucket.layout.pullRequest
```

Plugin decorators

Stash 3.x

```
stash.*
```

```
e.g. stash.repository.settings
```

Bitbucket 4.x

```
bitbucket.*
```

```
bitbucket.repository.settings
```

See [plugin decorators documentation](#)

Web Resource Modules

Stash 3.x

```
<stash-resource/>
```

Bitbucket 4.x

```
<client-resource/>
```

Note that `client-resource` will expand `<directory/>` elements in-place, where `stash-resource` expanded them at the end.

Web Item icons (in `atlassian-plugin.xml`)

Stash 3.x

```
<param name="stashIconClass">...</param>
```

Bitbucket 4.x

```
<param name="iconClass">...</param>
```

Web I18n

Stash 3.x

```
stash_i18n
```

Bitbucket 4.x

```
getText,getTextAsHtml
```

`stash_i18n` should not be used; `getText` is a cross-product replacement that doesn't accept a default translation parameter. See [Writing Soy Templates](#) for usage details.

Form Fragments

Stash 3.x

```
stash.*
```

Bitbucket 4.x

```
bitbucket.*
```

Javascript Events

Stash 3.x

```
stash.*
```

Bitbucket 4.x

```
bitbucket.internal.*
```

See below for additional information about JavaScript events.

I18n keys

Stash 3.x

```
stash.*
```

Bitbucket 4.x

bitbucket.*

Java API - Guava**Stash 3.x**

com.google.common.base.Function/Predicate

Guava 11 was available to plugins

Bitbucket 4.x

Java 8 Lambdas

Guava types like Function or Predicate are no longer used in API type signatures, use the corresponding types from java.util.function instead

Guava 18 is available to plugins

Servlet Dependency**Stash 3.x**

```
<groupId>javax.servlet</groupId>  
  <artifactId>servlet-api</artifactId>  
  <scope>provided</scope>  
</dependency>
```

Bitbucket 4.x

```
<groupId>javax.servlet</groupId>  
  <artifactId>javax.servlet-api</artifactId>  
  <scope>provided</scope>  
</dependency>
```

The artifactId has changed to javax.servlet-api from servlet-api.

Branch permissions**Stash 3.x**

canDelete, canWrite

search

Bitbucket 4.x

hasPermission

Checking for access to a Ref is now done by calling the RefRestrictionService's hasPermission method.

The search method for AccessGrants has been removed. AccessGrants can now be accessed by calling getAccessGrants on a RefRestriction.

Exported 3rd party libraries**Stash 3.x**

Apache HTTP Client 3.x is no longer exported by the host app.

Bitbucket 4.x

Use HTTP Client 4.x instead.

```
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <scope>provided</scope>
</dependency>
```

SAL Scheduler deprecated

Stash 3.x

SAL 2.0

```
com.atlassian.sal.api.scheduling.PluginScheduler
```

(Since Bitbucket 4.0.0 will throw an `IllegalArgumentException` if any job data passed to `scheduleJob` does not implement `java.io.Serializable`)

Bitbucket 4.x

SAL 3.0 (see [SAL 3.0 upgrade guide](#) for further details)

Use `com.atlassian.scheduler.SchedulerService` provided by:

```
<groupId>com.atlassian.scheduler</groupId>
<artifactId>atlassian-scheduler</artifactId>
```

instead.

Outline of changes made to the REST API

The *payloads* for some of the REST resources have changed. All URLs are the same except that the default context path is now `/bitbucket` instead of `/stash`.

Self Links

Some REST model classes, like `RestStashUser`, had their "self" links defined two ways:

```
{
  "link": {
    "url": "/users/admin",
    "rel": "self"
  },
  "links": {
    "self": [
      {
        "href": "http://localhost:7990/stash/users/admin"
      }
    ]
  }
}
```

The "link" attribute is from 1.0 and was deprecated in 2.11. From 4.0, the "link" attribute has been removed. The "self" entry in the "links" map remains.

Changeset to Commit

Ref output, such as branches and tags, had a "latestChangeset" attribute. The following output is from the `/projects/KEY/repos/slug/branches` resource:

```

{
  "size": 1,
  "limit": 1,
  "isLastPage": false,
  "values": [
    {
      "id": "refs/heads/search/STASHDEV-8813-search",
      "displayId": "search/STASHDEV-8813-search",
      "latestChangeset": "de307ea7b6abfalaad8de6771d79da0a9a7fd3cb",
      "latestCommit": "de307ea7b6abfalaad8de6771d79da0a9a7fd3cb",
      "isDefault": false
    }
  ],
  "start": 0,
  "nextPageStart": 1
}

```

A "latestCommit" attribute was added in 3.7 and "latestChangeset" was deprecated. It has been removed in 4.0. *This also applies to pull request refs.*

AttributeMap toPropertyMap

Some REST objects, most notably `RestPullRequest`, had an "attributes" attribute which allowed plugin developers to attach arbitrary data. However, the model for this was very restrictive, being defined in `AttributeMap` as `Map<String, Set<String>>`. In 3.2, `PropertyMap`, defined as `Map<String, Object>`, was added to replace the attribute support. However, for performance and API consistency reasons, most existing attributes were not converted over. In 4.0, the changeover is now complete.

The following JSON snippets show the old and new layouts for pull request properties.

```

"attributes": {
  "resolvedTaskCount": [
    "0"
  ],
  "openTaskCount": [
    "1"
  ],
  "commentCount": [
    "2"
  ]
}

```

```

"properties": {
  "commentCount": 2,
  "openTaskCount": 1,
  "resolvedTaskCount": 0
}

```

As you can see, the new "properties" map allows its numeric entries to be numeric, resulting in much more readable, useful output.

PullRequestOrderdefault order

The `getDefaultOrderForState(PullRequestState state)` method in `PullRequestOrder` has been replaced with `getDefaultOrder()`, which always returns `PullRequestOrder.NEWEST`.

XSRF Protection enabled by default

To prevent a malicious hacker submitting a form from a foreign site to a stash instance, XSRF protection has now been enabled by default on all REST resources which do **not** accept the "application/json" Content-Type.

Any client POSTing to these resources will need to add a special header to disable this check.

Header name: X-Atlassian-Token

Header value:no-check

For example, to set a Project Avatar via curl, the following command can be used:

```
$ curl -X POST -u username:password -H "X-Atlassian-Token: no-check" http://localhost:8080/rest/api/1.0/projects/TEST/avatar.png -F avatar=@avatar.png
```

The `-H "X-Atlassian-Token: no-check"` parameter was not needed in 3.x, however is now needed since 4.0

For your own REST resources which do not accept "application/json" (e.g. multipart/form-data, plain/text etc), you can opt-out of XSRF protection by using the `@XsrfProtectionExcluded` annotation.

If you use the excluded annotation, it should be because your endpoint has some other type of protection against XSRF/CSRF attacks.

Javascript Events

Since Stash 3.0 we have provided a Javascript API module for creating and consuming events (`stash/api/util/events`, now `bitbucket/util/events`), however we haven't documented what events Bitbucket (Stash) emits as part of our Developer Docs. Which events should be used and which should be considered internal implementation details were subject to change.

We are actively looking at which events we should consider part of the API. We will document usage and guarantee the stability for events that are part of the API for a major version. [We would like to hear any feedback](#) on which events you make use of in your apps, and why, to aid in our consideration of what to consider for the JS Events API.

App update strategies

There are two primary strategies we are suggesting to update your app, and here we explain how to implement each and how to provide support for your app going forward.

Hard break

The simplest way forward is to branch your app and only release Bitbucket Server 4.0 compatible versions in the future. **Replace** the old `com.atlassian.stash` dependency with the new `com.atlassian.bitbucket` one, fix the resulting compilation errors, and create a new listing on Marketplace.

Backwards/forwards compatibility

The best way to achieve this is to maintain a branch of the "Stash" version of your plugin, and merge changes onto a master branch which contains the "Bitbucket server" version.

Unfortunately, unless you have written a pure cross-product app, building two versions from the same branch of code is in general not worth the overhead.

Rename checklist

1. Beware of changing any Strings which are used as keys for accessing data your app may store. e.g. namespaces used with `PluginSettingsFactory.createSettingsForKey` or prefixes used with `ApplicationPropertyService.getPluginProperty`
2. We strongly recommend that you **do not change your appkey**; if you do, customers won't be able to see the updated version of your app in their Universal Plugin Manager. The default `atlassian-plugin.xml` generated by AMPS uses `key="${project.groupId}.${project.artifactId}"`. Changing your `Maven groupId` or `artifactId` this will change your appkey.

3. If you are using ActiveObjects, you are *strongly* encouraged to set the `namespace` attribute to ensure the unique hash in your table names does not change. Otherwise anyone who has installed your plugin will "lose" all of their data when your plugin starts using new tables! For example, here's how we defined the `<ao/>` module in our ref sync plugin:

```
<ao key="ao" namespace="com.atlassian.stash.stash-repository-ref-sync">
```

Upgrade Bitbucket from an archive file

This page describes how to upgrade Bitbucket Data Center and Server using a zip or tar.gz file.

Upgrading to any later version is free if you have current software maintenance. See our [Licensing FAQ](#) to find out more.

Other ways to upgrade Bitbucket:

- [Installer](#) - the simplest way to upgrade Bitbucket.
- [Docker container image for Bitbucket Server](#).

On this page:

Before you begin

Plan your upgrade

1. [Determine your upgrade path](#)
2. [Complete pre-upgrade checks](#)
3. [Upgrade Bitbucket in a test environment](#)

Upgrade Bitbucket

4. [Back up your data](#)
5. [Stop the application](#)
6. [Download Bitbucket](#)
7. [Extract the file and upgrade Bitbucket](#)
8. [Start Bitbucket](#)

Before you begin

Before you upgrade Bitbucket you should answer these questions about your instance.

Is manual the right upgrade method for you?	<p>You can choose to upgrade using the installer, or, if the installer is not suitable for your situation you can use an archive file to update your Bitbucket instance. Only use an archive file to update Bitbucket if the installer is not suitable for your situation, as the installer is the recommended approach to upgrade for most use cases.</p> <p>Read about upgrading Bitbucket Server using the installer.</p>
Are you eligible to upgrade?	<p>To check if software maintenance is current for your license, go to  > Licensing and make sure the license support period has not expired.</p> <p>If your support period has expired, follow the prompts to renew your license and reapply it before upgrading.</p>
Have our support platforms changed?	<p>Check the Supported platforms page for the version of Bitbucket you are upgrading to. This will give you info on supported operating systems, databases and browsers.</p>
Have you modified your <code>server.xml</code> file?	<p>In Bitbucket 5 we changed where you configure properties such as port numbers, context paths, and access protocol (among other things) to centralize custom configuration to a single directory and file, and to make upgrading easier for future releases.</p> <p>Upgrading from any version earlier than or including Bitbucket Server 4.14 to Bitbucket Server 5.0 or later requires that you manually migrate any changes to the <code>server.xml</code> file to the <code>bitbucket.properties</code> file.</p> <p>See the page Migrate customizations from <code>server.xml</code> to <code>bitbucket.properties</code> for instructions on locating your <code>server.xml</code> file and how to migrate your customizations.</p>

Plan your upgrade

1. Determine your upgrade path

Check the [Supported platforms](#) page to determine if your environment meets the minimum requirements to run the latest version of Bitbucket. Also read the [End of support announcements](#).

You can update from any previous version of Bitbucket Server (or Stash) to the latest version, as there is no required upgrade path.

Bitbucket 7 is a major upgrade

Be sure to read the [Bitbucket Server 7.0 release notes](#), take a full backup, and test your upgrade in a non-production environment before upgrading your production site.

Upgrading from a version older than Bitbucket 7 disables all user-installed apps on startup

Be sure to update your own apps and check the [Atlassian Marketplace](#) to ensure 3rd-party apps are compatible with Bitbucket Server 7 before upgrading.

Upgrading to the next bug fix update

If you are upgrading a multi-node Bitbucket cluster to the next bug fix update (for example, from 7.9.0 to 7.9.4), you can [upgrade with no downtime](#).

2. Complete pre-upgrade checks

1. Check the [Version specific upgrade notes](#) for the version you plan to upgrade to (and any in between).
2. Go to  **Support Tools**, then review the Log analyzer for any issues that may need to be resolved.
3. Check the compatibility of your apps with the version you plan to upgrade to.
 - a. Go to  **Manage apps** > **Bitbucket update check**.
 - b. Choose the version you plan to upgrade to then hit **Check**.

If your users rely on particular apps, you may want to wait until they are compatible before upgrading Bitbucket. App vendors generally update their apps very soon after a major release.

Good to know:

- Upgrading from a version older than Bitbucket Server 6 will automatically disable all user-installed apps. You will need to enable your apps after successfully upgrading, regardless of compatibility with Bitbucket Server 6.
- You can disable an app temporarily if it is not yet compatible.

3. Upgrade Bitbucket in a test environment

1. Create a staging copy of your current production environment.
2. Follow the steps below to upgrade your test environment.
3. Test any unsupported apps, customizations and proxy configuration (if possible) before upgrading your production environment.

Upgrade Bitbucket

This upgrade process does not perform an in-place upgrade, but instead installs the new version of Bitbucket into a new installation directory. The new instance uses your existing Bitbucket home directory and external database.

If necessary, rolling back an upgrade can only be performed by restoring a backup of *both* the Bitbucket home directory and the Bitbucket database. Rolling back requires a consistent home directory and database. You can then reinstall the previous version of the application to the installation directory. Read the [Rollback section](#) for more details.

4. Back up your data

1. Determine which backup strategy to use.

For **Bitbucket Server** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), the [Backup Client](#) (version 2.7 or later), or [DIY Backup](#) (version 2.12 or later) while Bitbucket is running, or just stop Bitbucket and zip up / snapshot the home directory and database and keep them somewhere safe.

For **Bitbucket Data Center** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), [DIY Backup](#), or take snapshots of the shared home directory (on NFS) and database while all nodes are stopped.

For **Bitbucket mirrors**, the home directory doesn't store any persistent state that can't be reconstructed from the primary Bitbucket instance, but you should still make sure you have a backup of at least the important configuration files such as SSL certificate, `server.xml`, `config/ssh-server-keys.pem`, `bitbucket.properties` file, and so on in a safe place. See [How do I back up my mirrors?](#) for more information.

See the article [Data recovery and backups](#) for detailed information and guidance on creating an effective backup strategy.

2. Back up all the data in your [Bitbucket home directory](#) and [external database](#).

5. Stop the application

To stop Bitbucket Server from the interface, use the Bitbucket Server items in the Windows Start menu.

To stop Bitbucket Server from a command prompt, change directories to the `<Bitbucket Server installation directory>` and run this command:

```
bin\stop-bitbucket.bat
```

To stop Bitbucket Server when running as a service, stop the Bitbucket Server service from the Windows services console.

Stop Bitbucket Server from a terminal by changing to the `<Bitbucket Server installation directory>` and running:

```
bin/stop-bitbucket.sh
```

Be sure to *stop all nodes* of Bitbucket Data Center, perform the steps to upgrade for a single node first, then repeat the process on each cluster node.

See the page [Start and stop Bitbucket](#) for more detailed instructions.

6. Download Bitbucket

Download the appropriate file for your operating system -<https://www.atlassian.com/software/bitbucket/download-archives>.

7. Extract the file and upgrade Bitbucket

1. Extract (unzip) the files to a directory (this is your new installation directory, and **must** be different to your existing installation directory).

Use the same user account to extract and to run Bitbucket to avoid permission issues at startup. For production installations, we recommend you use new dedicated user that will run Bitbucket on your system.

See the page [Running Bitbucket Server with a dedicated user](#) for more details.

2. Update the value of `BITBUCKET_HOME` in the `<Bitbucket Server installation directory>/bin/set-bitbucket-home.sh` (or `set-bitbucket-home.bat` on Windows) file so the new Bitbucket installation points to your existing Bitbucket home directory (if you use a `BITBUCKET_HOME` environment variable to specify the home directory location, no change is required).

8. Start Bitbucket

To start Bitbucket Server from a command prompt, change directories to the `<Bitbucket Server installation directory>` and run:

```
bin\start-bitbucket.bat
```

To start Bitbucket Server to run as a service, start the application service from the services console.

To start Bitbucket Server from a terminal, change to the `<Bitbucket Server installation directory>` and run:

```
bin/start-bitbucket.sh
```

For Bitbucket Data Center, Bitbucket Server and the *remote* Elasticsearch instance must be started separately. It is a requirement of the Bitbucket Data Center setup that you have *only one* remote Elasticsearch instance for your entire cluster, as described on [Installing Bitbucket Data Center](#). The `start-bitbucket.sh` command starts both Bitbucket and Elasticsearch, and should not be used to start your Bitbucket Data Center application nodes.

When using a remote Elasticsearch instance, instead of the bundled Elasticsearch instance, start Bitbucket Server by running `start-webapp.sh` instead of `start-bitbucket.sh`.

To start Bitbucket Data Center

1. **Start each of your application nodes.** From the `<Bitbucket Server installation directory>`, run this command for each node:

For Windows

```
bin\start-webapp.bat
```

For Linux

```
bin/start-webapp.sh
```

2. **Start your Elasticsearch node.** From the <Bitbucket Server installation directory>, run this command:

For Windows

```
bin\start-search.bat
```

For Linux

```
bin/start-search.sh
```

Read the page [Start and stop Bitbucket](#) for more details.

Rollback

If necessary, rolling back an update can only be performed by restoring a backup of *both* the Bitbucket Server [home directory](#) and the Bitbucket Server database. Rolling back requires a consistent home directory and database. You can then reinstall the previous version of the application to the installation directory. Never start an older binary against an upgraded home directory.

Version-specific update notes

This section provides specific update notes for each version of Bitbucket Data Center and Server. These notes supplement the primary upgrade instructions above. You should read the relevant sections for each version between your current version of the application and the version you are upgrading to.

Bitbucket Data Center and Server 7.14 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.14 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.13 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.13 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.12 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.12 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.11 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.11 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Data Center and Server 7.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Data Center and Server 7.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Server 7.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server and Data Center 7.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Server 7.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server and Data Center 7.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#)

Bitbucket Server 7.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.7 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.6 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.6 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.5 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.5 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.4 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.4 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

i New Data Center apps check as part of your license upgrade

Upgrading to a Bitbucket Data Center license now comes with a new apps check. Before we apply your new Bitbucket license, we'll show you if any of your installed apps will need to be upgraded to the [Data Center approved version](#).

We've also introduced new status messages in the **Manage apps** section of your admin console, so you can clearly see if any apps will require a new license.

New status	Host product license	What it means
	Evaluation license	The app vendor offers a Data Center version of this app. If you switch to a paid Data Center product license, you'll need to upgrade this app license to Data Center.
	Full license	The app vendor offers a Data Center version of this app. Your Server app license is no longer compatible with the product. This means the app has either stopped working, or functionality has been lost or compromised. You need to add a Data Center license.

Learn more about [upgrading your Server apps after you move to Data Center](#)

i Build status data migration

When upgrading to Bitbucket Server 7.4, you may notice an increased load on your database and application server. This is normal. It's due to an asynchronous upgrade task that needs to run to migrate your applications build status data.

While the upgrade task is running, your application will be available and able to serve requests, so your team can continue working as usual.

The upgrade task will run automatically around two minutes after your application starts up. In a multi-node cluster, it will run on a node picked by the system. How long it takes will depend on the size of the data that needs to be migrated. For example, for small instances it can take 10 minutes, and for large instances it can take over an hour.

 Before upgrading to Bitbucket Server 7.4, you should test HTTP hosting in a staging environment that mirrors your production setup. This especially includes any installed apps, as app-provided servlet filters may be broken by async HTTP.

Bitbucket Server 7.3 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.3 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.2 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.1 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 7.0 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 7.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

 If you are considering upgrading to bitbucket 7.0 or higher and are using version 2.8 or earlier of the `Bitbucket branch source` Jenkins plugin, there is a known issue that can cause builds to fail. This is because merges are performed lazily in order to manage scaling while changes to pull requests happen continuously.
See

 [BSERV-12284](#) - Eagerly update refs/pull-requests/* from when a pull request's source branch changes CLOSED

Bitbucket Server 6.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.7 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.6 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.6 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.5 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.5 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.4 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.4 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.

- the [End of support announcements](#).

Bitbucket Server 6.3 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.3 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.2 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.1 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Bitbucket Server 6.0 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 6.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).



To upgrade to Bitbucket Server 5+, you must migrate customizations to `bitbucket.properties`

Be sure to carefully read the [Bitbucket Server upgrade guide](#) to determine what impact this change may have on your instance. The page [Migrate `server.xml` customizations to `bitbucket.properties`](#) has specifics on which properties need to be migrated, how to migrate them, and includes some examples that demonstrate how to migrate some common customizations.



Start script changes

Starting from Bitbucket Server 5.0, some startup scripts had their name prefixed with an underscore. These scripts have been deprecated, and you should no longer call these scripts directly. This is to avoid confusion, because previously there were multiple start and stop scripts, making it easy to confuse which one to call. Now, there is only one script, that begins with "start" and one that begins with "stop".

Known issues

P Key Summary Status

No issues found

Bitbucket Server 5.16 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.16 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.15 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.15 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.14 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.14 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.13 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.13 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.12 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.12 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.11 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.11 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.7 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.6 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.6 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.5 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.5 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.

- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.4 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.4 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.3 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.3 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.2 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.1 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Bitbucket Server 5.0 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 5.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).



To upgrade to Bitbucket Server 5+, you must migrate customizations to bitbucket.properties

Be sure to carefully read the [Bitbucket Server upgrade guide](#) to determine what impact this change may have on your instance. The page [Migrate server.xml customizations to bitbucket.et.properties](#) has specifics on which properties need to be migrated, how to migrate them, and includes some examples that demonstrate how to migrate some common customizations.

⚠ Start script changes

In Bitbucket Server 5.0, some startup scripts have had their name prefixed with an underscore. These scripts have been deprecated, and you should no longer call these scripts directly. This is to avoid confusion, because previously there were multiple start and stop scripts, making it easy to confuse which one to call. Now, there is only one script, that begins with "start" and one that begins with "stop".

❗ BitbucketServer does not support reverse proxies using AJP connections

Starting from version 5.0, Bitbucket Server does not support reverse proxies using AJP connections. While never officially supported, previous versions of Bitbucket Server could use AJP connections. However, with changes to how Bitbucket Server uses Tomcat, using AJP connections with Bitbucket Server (or Data Center) 5.0+ will cause the application to fail on startup.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Bitbucket Server 4.14 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.14 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements for Bitbucket Server](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Bitbucket Server 4.13 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.13 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Bitbucket Server 4.12 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.12 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

 **Do NOT upgrade to Git 2.11+**

Any version after Git 2.11.0 (including future 2.12+ releases) cannot be used with Bitbucket Server. Bitbucket Server 4.12 will fail on startup if Git 2.11+ is detected. Only upgrade to versions of Git which are explicitly marked supported on our [Supported Platforms](#) page.

 **BSERV-9388** - Pre-receive hooks and branch permissions reject valid pushes on git 2.11

CLOSED

Known issues

P	Key	Summary	Status
▼	BSERV-9592	Spotted Typo on Analytics page on Bitbucket Server Settings	GATHERING IMPACT
▼	BSERV-8201	Repository deletion times out	GATHERING IMPACT
▼	BSERV-9675	Duplicated branches and tags - Multiple entries for the same ref id	LONG TERM BACKLOG

[3 issues](#)

Bitbucket Server 4.11 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.11 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.10 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.10 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.9 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.9 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.8 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.8 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.7 update notes

You should also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.7 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.6 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.6 release notes](#).
- the [Stash Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.5 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.5 release notes](#).
- the [Bitbucket Server Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.4 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.4 release notes](#).
- the [Bitbucket Server Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Bitbucket Server 4.3 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.3 release notes](#).

- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.2 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.2 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.1 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.1 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#).

Known issues

P Key Summary Status

[No issues found](#)

Bitbucket Server 4.0 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Bitbucket Server 4.0 release notes](#).
- the Bitbucket Server [Supported platforms](#) page.
- the [End of support announcements](#). Note that Bitbucket Server 4.0 does not support Internet Explorer 9 or 10, Java 7, nor versions of Git earlier than 1.8 on the server.

Do you rely on user-installed add-ons?

⚠ Your plugins may not be compatible with Bitbucket Server 4.0 (yet)

Upgrading from Stash to Bitbucket Server will disable all user-installed add-ons, whether they are 3rd-party add-ons from Atlassian Marketplace, or your own custom add-ons. User-installed add-ons will need to be checked, upgraded and enabled again once you've finished the upgrade process. You should check [the marketplace listing](#) for 3rd-party add-ons to determine if there is a release that is compatible with Bitbucket Server 4.0. Unless they have been updated to work with Bitbucket Server 4.0, existing add-ons (or plugins) that use the API interfaces that have been removed in Bitbucket Server 4.0 *will not work*. Similarly, custom add-ons need to be upgraded for Bitbucket Server compatibility. See the documentation on [how to update your add-on](#) for details.

If you are upgrading from Stash 3.x to Bitbucket Server 4.x, you should be aware that most user-installed add-ons will be incompatible with Bitbucket Server 4.0. After upgrading, go to **Admin > Manage add-ons**, look for messages of this form, and follow the advice to update:

A newer version of the Universal Plugin Manager is available. [Update Now](#)
[Skip this version](#) [Remind me later](#)

If no newer version is available the add-on will remain disabled.

New dedicated user account 'atlbitbucket'

A new dedicated user account will be created to run Bitbucket Server, called `atlbitbucket`. During the upgrade process, you will have an option to delete the dedicated user account for running Stash, named `atlstash`. For most users, deleting this dedicated user account won't have any negative consequences, however if you rely on this dedicated user accounts, for example in custom backup scripts, you will need to update the user account in those scripts.

Updates to logger names

If you have customized your logging configuration by manually editing the `logback.xml` file (using steps found in the [Configure Stash Logging KB article](#)), you should be aware of changes to several logger names that may require you to update some of your configuration files.

Stash logger name	Bitbucket Server logger name
<code>stash.application</code>	<code>bitbucket.application</code>
<code>stash.profiler</code>	<code>bitbucket.profiler</code>
<code>stash.access-log</code>	<code>bitbucket.access-log</code>
<code>stash.audit-log</code>	<code>bitbucket.audit-log</code>
<code>stash.mail-log</code>	<code>bitbucket.mail-log</code>

Important security improvement which may require configuration changes

Previous to Bitbucket Server 4.0, a security constraint for redirecting from HTTP to HTTPS was not enforced, meaning users could type "http://<bitbucketserver-url>" into their browser and still be shown a functioning version of Bitbucket Server (or Stash). Included with the release of Bitbucket Server 4.0 was a fix to enforce that security constraint. Using the previous security configuration with Bitbucket Server 4.0 means trying to access the application over an insecure connection, when users type "http" when trying to get to the application, they could encounter erroneous behavior.

As part of upgrading from Stash to Bitbucket Server, it is recommended you update your security configuration in order to avoid encountering this problem. See this Knowledge Base article for more information and specific instructions on addressing this: [Redirect HTTP Requests to HTTPS](#).

Known issues

P Key Summary Status

No issues found

Stash 3.11 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.11 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.11 [does not support Git 2.0.2 or 2.0.3](#).
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9 or 10, Java 7, nor versions of Git earlier than 1.8 on the server.
- If you have installed the **stash-stream-guard-plugin** (for example, if advised to do so by Atlassian Support), then you should uninstall it when upgrading to 3.11.2 as its functionality is now bundled in Stash.

Known issues

P Key Summary Status

No issues found

Stash 3.10 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.10 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.10 [does not support Git 2.0.2 or 2.0.3](#).
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9 or 10, Java 7, nor versions of Git earlier than 1.8 on the server.

Are you using Git 2.2x - 2.4.0?

Git 2.2.x - 2.4.0 have reported performance issues when interacting with NFS. Hence, these versions are recurrently not supported for Stash Data Center or for Stash Server installations that use NFS mounts for the home directory.

Known issues

P Key Summary Status

No issues found

Stash 3.9 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.9 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.9 [does not support Git 2.0.2 or 2.0.3](#).
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Are you using JIRA as a Crowd server?

Stash 3.9 cannot use JIRA 6.4 as a Crowd server due to a bug in JIRA 6.4. Please upgrade to JIRA 6.4.1 before upgrading Stash. (Note that Stash 3.7.2 and 3.8.0 contained a workaround for the bug but the workaround was removed in Stash 3.9)

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 3.8 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.8 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.8 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Have you made customizations to Tomcat's `server.xml` file?

For Stash 3.8 (and future versions) the `server.xml` file is now located in the `<stash home directory>/shared` directory. The benefit of this move is that your customizations to `server.xml` will not have to be redone for future upgrades.

⚠ You still need to update your custom configurations in `shared/server.xml` for the upgrade to Stash 3.8.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 3.7 update notes

Also see:

- the general [Upgrade steps](#) section above.
- the [Stash 3.7 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.7 **does not support Git 2.0.2 or 2.0.3**.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Do you use JIRA 6.4?



JIRA 6.4 has a known issue with Stash versions 3.4.3 to 3.7.1, which prevents the user synchronization from working. This will only affect you if you use JIRA to manage your users in Stash, and is fixed in Stash 3.7.2.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash *is now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 3.6 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.6 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.6 *does not support Git 2.0.2 or 2.0.3*.
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Internet Explorer 9, Java 7, nor versions of Git earlier than 1.8 on the server.

Secure email notifications

Stash 3.6 now:

- Allows you to require STARTTLS support on the mail server when using SMTP if it is not supported, mail is not sent.
- Supports SMTPS (where the whole protocol conversation uses SSL/TLS).

See [Setting up your mail server](#) for more information.

Note that if you use either SMTP with STARTTLS or SMTPS and connect to a self-signed mail server you may need to import the server's certificate and set up a custom cacerts file for Stash (just as you do for any outbound SSL/TLS connection to a self-signed server). See this [Stash knowledge base article](#) for information about how to do that.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine was changed in Stash 3.5 from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash *is now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 3.5 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.5 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.5 [does not support Git 2.0.2 or 2.0.3](#).
- the [End of support announcements](#). Note that, in particular, Stash 4.0 will not support Java 7, nor versions of Git earlier than 1.8 on the server.

Stash home directory location

Stash 3.5 and later versions no longer allow the [Set the home directory](#) to be the same directory as, or a subdirectory of, the Stash installation directory. The Stash home directory, as defined by the `STASH_HOME` environment variable, must be in a separate location Stash will fail on startup otherwise.

Deprecation of HighlightJS for syntax highlighting

The highlighting engine in Stash has been changed from HighlightJS to CodeMirror. The use of HighlightJS for syntax highlighting in Stash is *now deprecated*, and will be removed in Stash 4.0. See [Syntax highlight changes](#) for information about how to migrate any custom mappings for HighlightJS that you may have made.

Known issues

priority	key	summary	status
		 Jira project doesn't exist or you don't have permission to view it.	
		View these issues in Jira	

Stash 3.4 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.4 release notes](#).
- the Stash [Supported platforms](#) page. Note that Stash 3.4 [does not support Git 2.0.2 or 2.0.3](#).
- the [End of support announcements](#).

Changed group membership aggregation with multiple user directories

Stash 3.4 uses new schemes to determine the effective group memberships when Stash is connected to multiple user directories, and duplicate user names and group names are used across those directories. The new schemes are:

- 'aggregating membership'
- 'non-aggregating membership'.

These group membership schemes are only used to determine authorization. Authentication is determined on the basis of the group mappings in each directory.

See [Effective memberships with multiple directories](#) in the Crowd documentation for more information.

When you update to Stash 3.4, an update task will apply one of those schemes as follows:

Aggregating membership will be applied to your instance:

- If there is only one active directory.

- If there are multiple active directories but only a single directory applies group memberships to any particular user.
- For example, if directory-1 contains user-a in group-x, and directory-2 contains user-b in group-y, then Stash 3.4 will apply aggregating membership, and permissions will not be affected.

Non-aggregating membership will be applied to your instance:

- If there are multiple active directories and more than one directory applies group memberships to at least one user.
- For example, if directory-1 contains user-a in group-x, and directory-2 contains user-a in group-y, then Stash 3.4 will apply non-aggregating membership. If group-y has admin privileges then user-a would have their privileges escalated in this case if aggregating membership was applied instead when upgrading to Stash 3.4.

Any changes made by the update task will be logged.

A Stash admin can change the membership scheme used by Stash using the following commands:

- To change to *aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d '{"membershipAggregationEnabled":true}' -u <username>:<password> <base-url>/rest/crowd/latest/application
```

- To change to *non-aggregating membership*, substitute your own values for `<username>`, `<password>` and `<base-url>` in this command:

```
curl -H 'Content-type: application/json' -X PUT -d '{"membershipAggregationEnabled":false}' -u <username>:<password> <base-url>/rest/crowd/latest/application
```

Note that changing the aggregation scheme can affect the authorization permissions for your Stash users, and how update operations are performed. Read more about using Stash with [multiple use directories](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 3.3 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.3 release notes](#).
- the [Stash Supported platforms](#) page. Note that Stash 3.3 does not support Git 2.0.2 or 2.0.3.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 3.2 update notes

 Upgrading to Stash 3.2 or later from Stash 3.1 or earlier is irreversible please see the [Home directory migration](#) section below.

Also see:

- the [general update steps](#) section above.
- the [Stash 3.2 release notes](#).
- the [Stash Supported platforms](#) page.
- the [End of support announcements](#).

Note that:

- Stash 3.2 [does not support Git 2.0.2 or 2.0.3](#).

Home directory migration

When you update to Stash 3.2 or later, an update task migrates directories in your Stash home directory to new locations. This is irreversible once you update to Stash 3.2 or later you can not revert to Stash 3.1 or earlier, because of changes to the Stash home directory format.

For most installations, Stash 3.2 is able to perform these moves automatically and transparently. However, in the rare instance where Stash 3.2 can't perform the update automatically, please refer to [Upgrading your Stash home directory for Stash 3.2 manually](#).

Undocumented home directory overrides are no longer supported

Before Stash 3.2 it was possible to override the location of the following subfolders in the Stash home directory, using undocumented environment variables or system properties:

- `export`
- `bin`
- `caches`
- `config`

- `data`
- `lib`
- `lib/native`
- `log`

- `plugins`
- `tmp`

In Stash 3.2 only the `tmp` subfolder can be overridden in this way. Attempting to override the others will fail on startup. For more information, please see [Stash fails to start - UnsupportedDirectoryOverrideException](#).

Stash analytics

Stash 3.2, and later, collects user event data, unless this is disabled by an administrator. You will see outgoing network traffic to amazonaws.com. See [Change data collection settings](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Stash 3.1 update notes

Also see:

- the general [update steps](#) section above.
- the [Stash 3.1 release notes](#).
- the [Stash Supported platforms](#) page.

Note that:

- Stash [does not support Git 2.0.2 or 2.0.3](#).

Known issues

P Key Summary Status

No issues found

Stash 3.0 update notes

Stash no longer supports Java 6

Stash 3.0 requires at least Java 7, and supports Java 8.

Please see the [End of support announcements](#).

Your plugins may not be compatible with Stash 3.0

The interfaces in the Stash API for plugin developers that *were deprecated* in Stash 2.11 and earlier *have been removed* in Stash 3.0. This means that, unless they have been updated to work with Stash 3.0, existing Stash add-ons (or plugins) that use these interfaces *will not work with Stash 3.0*.

Please see the section below about Stash add-on incompatibilities for more details.

Also see:

- the general [update steps](#) section above.
- the [Stash 3.0 release notes](#).
- the [Stash Supported platforms](#) page.

Note that:

- Stash now has installers for the Linux, macOS and Windows platforms.
- Stash no longer supports Internet Explorer 8, as [previously announced](#).
- Stash does not support the Apache HTTP Server `mod_auth_basic` module.
- Stash [does not support Git 2.0.2 or 2.0.3](#).

Stash add-on incompatibilities

Unless they have been updated to work with Stash 3.0, existing Stash add-ons (or plugins) that use the API interfaces that have been removed in Stash 3.0 *will not work*.

Fresh installs of Stash 3.0 shouldn't encounter any problems. The Stash 'Manage add-ons' page (in the admin area) should only display add-ons from the Marketplace that have been marked as compatible with Stash 3.0. Incompatible add-ons won't be available in the list.

However, if you are upgrading from Stash 2.x to Stash 3.0, you should be aware that some existing installed add-ons may be incompatible with Stash 3.0. After upgrading, you should go to **Admin > Manage add-ons**, look for messages of this form, and follow the advice to update:

 A newer version of the Universal Plugin Manager is available. [Update Now](#)

[Skip this version](#) [Remind me later](#)

If no newer version is available, the add-on must be disabled.

Custom add-ons

Please note that your custom locally-developed plugins may be affected by the API removals in Stash 3.0. You will need to update your custom plugins if you want those to work with Stash 3.0. See the [Stash API changelog](#) for details of the deprecated APIs.

Third-party add-ons

You'll need to check on Atlassian Marketplace for the compatibility status of any 3rd-party add-ons that you use.

Third-party add-on developers have been given an Early Access Program (EAP) build of Stash 3.0 in advance of release, and many have already updated their add-ons to be compatible. Add-ons must be explicitly marked by the publisher as compatible with Stash 3.0 for them to appear in 'Manage add-ons' page in Stash. **This is NOT automatic as was the case with previous minor releases such as 2.10 and 2.11.** Atlassian can not support issues involving third party Add-ons that are incompatible with Stash 3.0; such support cases must be directed to the third-party publisher of the add-on. See [Managing apps](#).

Atlassian add-ons

All of the Atlassian add-ons for Stash that are available from the Atlassian Marketplace have been updated to be compatible with Stash 3.0. If you use any of these in your Stash installation you'll need to update them to the Stash 3.0 compatible version.

Add-on	JAR file name	Stash 3.0 compatible version
Custom Navigation Plugin	custom-navigation-plugin	2.0.3
Stash Archive Plugin	stash-archive	1.3.0
Repository git operations plugin	stash-git-ops-plugin	1.2.1
Stash Auto Unapprove Plugin	stash-auto-unapprove-plugin	1.1
Stash Protect Unmerged Branch Hook	stash-protect-unmerged-branch-hook	1.1
Stash Reviewer Suggester	stash-suggest-reviewers	1.2
Stash Web Post Hooks Plugin	stash-web-post-receive-hooks-plugin	1.1.0
Realtime Editor for Stash	stash-editor-plugin	1.0.6

Known issues

P Key Summary Status

[No issues found](#)

Stash 2.12 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#). In particular, note that support for Java 6 is deprecated. Stash 3.0+ will require Java 7.
- the [Stash 2.12 release notes](#).

Note that:

- Stash does not yet support Java 8.
- Stash 2.12 does not support Git 1.8.4.3
- Stash does not support the Apache HTTP Server `mod_auth_basic` module.

See [Supported platforms](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Stash 2.11 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.11 release notes](#).

Note that Stash does not support Git 1.8.4.3, nor does Stash support Java 8 yet. See [Supported platforms](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Stash 2.10 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.10 release notes](#).

Note that Stash does not support Git 1.8.4.3. See [Supported platforms](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Stash 2.9 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.9 release notes](#).

Note that Stash does not support Git 1.8.4.3. See [Supported platforms](#).

Pull Request Ref Optimization

When you first start Stash after upgrading to Stash 2.9 a repository update task runs that optimizes the pull request refs for all repositories managed by Stash. It's important that you do not interrupt this update process. You can track the progress of this in the Stash logs. See

 [BSERV-3469](#) - Pull request references in the Git repository are never removed after merge or decline

CLOSED

Backup Client Update Required

Version 1.0.3 of the [Stash Backup Client](#) is required to back up Stash 2.9.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 2.8 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).
- the [Stash 2.8 release notes](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 2.7 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Repository System Information Plugin is now deprecated

The functionality of the [repository system information plugin](#) has now been moved into core Stash. The plugin will still work for Stash 2.x versions but is redundant as of Stash 2.7.

MySQL default isolation level

Stash 2.7.x uses `READ_COMMITTED` instead of the MySQL default isolation level (`REPEATABLE_READ`). This can result in exceptions when installing or upgrading to 2.7.x, if [binary logging](#) is enabled in your MySQL server. More details and a fix can be found in [this KB article](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Stash 2.6 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Stash 2.5 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Limited support for JIRA 4.4.x and earlier

JIRA 4.3+ allows for showing commits associated with issues in JIRA. However, viewing issues within Stash is not supported for JIRA 4.4.x and earlier.

Stash 2.4 update notes

Also see:

- the [update steps](#) section above.
- the [End of support announcements](#).

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Stash 2.3 update notes

Please also see the [update steps](#) section above.

When upgrading to Stash 2.3 you also need to update the SCMCache plugin, due to recent Stash API changes.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 2.2 update notes

Please see the [update steps](#) section above.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Stash 2.1 update notes

Please also see the [update steps](#) section above.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

No issues found

Install location for third-party libraries

As of Stash 2.1 you can install third-party libraries and jar files, such as the MySQL JDBC driver, into `<stash home directory>/lib`. This has the advantage that files in this location are not overwritten, and lost, when you update Stash.

Microsoft SQL Server JDBC driver

Stash 2.1 now uses the Microsoft SQL Server JDBC driver to access Microsoft SQL Server, instead of using the jTDS driver. In most cases, Stash will automatically swap to using the Microsoft driver on update and **no configuration is required**.

If Stash was configured to use Microsoft SQL Server by manually entering a JDBC URL, please refer to [this guide](#).

Stash 2.0 update notes

This section provides specific notes for upgrading to Stash 2.0. See also the [update steps](#) section above.

Tomcat

For Stash 2.0, Tomcat has been updated from version 6 to 7. As part of that update, the `server.xml` file has changed. If you have customized `server.xml` (for example, for `port`, `path` or `hostname`), you can not simply copy your own version across to the updated Stash; you must reapply your customizations to the `server.xml` file for the new version of Stash.

If you were running Stash as a Windows service and are upgrading from 1.x to 2.x you will need to reinstall the Stash service to make it use Tomcat 7.

To uninstall the Stash service you need to execute following commands from <STASH DISTRIBUTION DIR>\bin:

```
> net stop <service name>
> service.bat uninstall <service name>
```

You can call this command without the service name if you installed the Stash service with a default name.

After the service is uninstalled you can proceed with the [update steps](#) and [Running Bitbucket Server as a Windows service](#) instructions to configure Stash 2.x running as a service.

Perl

Stash 2.0 requires Perl for its branch permission functionality. If Perl is unavailable, Stash 2.0 will not start.

On Windows machines, Perl will only have been installed by the Git installer if the correct install option was chosen. See [Installing and upgrading Git](#).

Existing Git hooks

In order to support Branch Permissions, Stash 2.0 moves existing hooks in the `pre-receive` and `post-receive` folders under <Stash home directory>/data/repositories/NNNN/hooks (where NNN is the internal repository id) to `.../hooks/pre-receive.d/10_custom` or `.../hooks/post-receive.d/10_custom`. Consequently, custom hooks that use relative path names (e.g. `./foo.sh` or `../dir/foo.sh`) will be broken by the update to Stash 2.0.

Deprecation of Internet Explorer 8

Support for Internet Explorer 8 is deprecated from the release of Stash 2.0. The official end-of-support date is yet to be determined. See [Supported platforms](#) for details.

Known issues

P	Key	Summary	Status
---	-----	---------	--------

[No issues found](#)

Migrate server.xml customizations to bitbucket.properties

Bitbucket Server 5.0 introduces some changes to how connector configuration customisation is performed. In Bitbucket Server 4.x and earlier Bitbucket Server startup involved starting Apache Tomcat and deploying the web application into Tomcat. Customising connector configuration (for example to secure connections with SSL) involved updating the Tomcat `server.xml` file.

In Bitbucket Server 5.0 and later, an embedded container (still Apache Tomcat) is started by the application. This allows you to make customizations to connector configuration directly in the `bitbucket.properties` file (the same file that hosts the vast majority of other settings).

Upgrading from any version earlier than Bitbucket Server 4.14 or earlier to Bitbucket Server 5.x or later requires that you manually migrate any changes to the `server.xml` file to the `bitbucket.properties` file.

This document explains in what cases this may be necessary, describes how to perform this migration, and provides some migration examples for common use cases.

How do I know if there were customizations to my `server.xml` file?

The `server.xml` file was where you would make customizations to do a number of things, but most users used it to:

- Secure Bitbucket Server using SSL.
- Run Bitbucket Server behind a reverse proxy.
- Changed the port Bitbucket Server runs on.
- Set up a custom keystore.

How to migrate your customizations to `bitbucket.properties`

To migrate customizations made in the `server.xml` file to the `bitbucket.properties` file

1. Locate your `server.xml` file.

Release	Directory
Stash 3.7 and earlier	<Stash installation directory>/conf/server.xml
Stash 3.8 and later	<Stash home directory>/shared/server.xml
Bitbucket Server 4.0 - 4.14	<Bitbucket home directory>/shared/server.xml
Bitbucket Server 5.0 and later	N/A, replaced by <Bitbucket home directory>/shared/bitbucket.properties

2. Identify and note any customizations made. Ideally you will know which customizations were made and can locate them in your `server.xml` file.

This is not a comprehensive list. Review your `server.xml` file closely to ensure all your customizations are noted.

- a. Securing Bitbucket using SSL.
- b. Running Bitbucket behind a reverse a proxy.
- c. Bypassing the Bitbucket reverse proxy to solve Application Links issues.
- d. Bypassing the Bitbucket proxy to connect directly to a Data Center node (for troubleshooting, or to generate support zips).
- e. Configuring a custom keystore path and password.

3. Locate your `bitbucket.properties` file in the `<Bitbucket home directory>/shared` directory.
4. Look up the customization equivalent in the `bitbucket.properties` file by referring to one of the [migration examples](#) and/or by using the [migration table below](#).
5. Add the equivalent values to the `bitbucket.properties` file. Save and close the file.

The value for your load balancer to support session affinity ("sticky sessions") was previously set to `JSESSIONID`. The default value for Bitbucket 5.0+ has changed to `BITBUCKETSESSIONID`.

You need to *either* change your load balancer cookie configuration to `BITBUCKETSESSIONID`, or, change the value in the `bitbucket.properties` file to `JSESSIONID`.

To change the value in the `bitbucket.properties` file, add this line

```
server.session.cookie.name=JSESSIONID
```

To change this value in your load balancer configuration, locate your load balancer proxy configuration file, change the instances of `JSESSIONID` to `BITBUCKETSESSIONID`.

6. Remove or rename the `server.xml` file (don't delete this file until you confirm the customizations were successfully migrated to your upgraded instance).

Migration reference table

4.x and earlier (<code>server.xml</code>)	5.x and later (<code>bitbucket.properties</code>)
<code>compression="on"</code>	<code>server.compression.enabled=true</code>
<code>compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,application/javascript,application/x-javascript"</code>	<code>server.compression.mime-types=text/css,text/html,text/javascript,text/json,text/plain,text/xml,text/x-javascript,\application/javascript,application/json,application/x-javascript,application/vnd.git-lfs+json</code>
<code>connectionTimeout="20000"</code>	<code>server.connection-timeout=20000</code>
<code>path="/"</code>	<code>server.context-path=/</code>
N/A	<code>server.displayName=Atlassian Bitbucket</code>
<code>port="7990"</code>	<code>server.port=7990</code>
<code>useHttpOnly="true"</code>	<code>server.session.cookie.http-only=true</code>
<code>secure="true"</code>	<code>server.secure=true</code>
<code>SSLEnabled="true"</code>	<code>server.ssl.enabled=true</code>

keyAlias="YourAlias" certificateKeyAlias="YourAlias"	server.ssl.key-alias=tomcat
keystoreFile="/path/to/keystore/bitbucket.jks" certificateKeystoreFile="/path/to/keystore/bitbucket.jks"	server.ssl.key-store=\${bitbucket.shared.home}/config/ssl-keystore
keystorePass="<password value>" certificateKeystorePassword="<password value>"	server.ssl.key-store-password=changeit
sslProtocol="TLSv1.2"	server.ssl.protocol=TLSv1.2
keystoreType="JKS" certificateKeystoreType="JKS"	server.ssl.key-store-type=\${keystore.type:jks}
clientAuth="false" or clientAuth="true"	server.ssl.client-auth=want or server.ssl.client-auth=need
redirectPort="443"	server.redirect-port=443
proxyPort="443"	server.proxy-port=443
proxyName="mycompany.com"	server.proxy-name=mycompany.com
address="192.168.10.10"	server.address=192.168.10.10

For Bitbucket Data Center

These properties are applicable to Bitbucket Data Center instances. In previous versions these were set within load balancer configuration files. Depending on which load balancer you're using, there are many configuration options, so it is not possible to precisely describe the properties or values for every configuration.

Setting these values within the `bitbucket.properties` in file in 5.0+ overrides what's in the load balancer configuration file.

- `server.session.cookie.name=BITBUCKETSESSIONID`
- `server.session.timeout=1800`
- `server.session.tracking-modes=cookie`

Migration examples

Below are some examples that demonstrate some common use cases for customizing the `server.xml` file, and how you would migrate those values to the `bitbucket.properties` file.

For these examples, if there are less properties in the `bitbucket.properties` syntax than in the initial `server.xml` syntax, that indicates the default value would be acceptable and you don't need to add that property to achieve the same result.

Read the [Bitbucket Server config properties - Server section](#) for more details about all of the various properties that can be configured.

For Windows users

When adding values with file paths in them, for instance `server.context-path`, backslashes are escaped by default. Be sure to include two backslashes in any values with file paths in them.

Additional connectors

Bitbucket Server is preconfigured with a single connector and, as previously described, the default configuration for that connector can be configured using properties with a `server.` prefix. For example to change the port from the default 7990 to 7991:

```
server.port=7991
```

Bitbucket Server also supports up to five additional connectors and these are configured using properties with prefix `server.additional-connector.#`, where # is a number between 1 and 5 inclusive. For example, if in addition to the default connector that is configured to listen on port 7990, you wanted to add an SSL secured connector listening on port 8443, you would add these lines to the `bitbucket.properties` file:

```
server.additional-connector.1.port=8443
server.additional-connector.1.ssl.enabled=true
server.additional-connector.1.ssl.key-store=/path/to/keystore/bitbucket.jks
server.additional-connector.1.ssl.key-store-password=<password value>
```

Running Bitbucket behind a reverse proxy secured with SSL

If Bitbucket Server is configured to run behind a reverse proxy that is secured with SSL, the existing `server.xml` file might contain the following connector configuration:

server.xml

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="443"
  compression="on"
  compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,application
/javascript,application/x-javascript"
  secure="true"
  scheme="https"
  proxyName="mycompany.com"
  proxyPort="443" />
```

To achieve the same configuration in Bitbucket Server 5.0 and later, add these entries to the `bitbucket.properties` file:

bitbucket.properties

```
server.secure=true
server.scheme=https
server.proxy-port=443
server.proxy-name=mycompany.com
```

For these examples, if there are less properties in the `bitbucket.properties` syntax than in the `initialserver.xml` syntax, that indicates the default value would be acceptable and you don't need to add that property to achieve the same result. For example, the port for the default connector is 7990, the default protocol is `HTTP/1.1`, and so on.

Bitbucket Server secured with SSL-terminating connector

If Bitbucket Server is secured with SSL, where the SSL connection is terminated at the application rather than a proxy, the existing `server.xml` file might contain the following connector configuration:

server.xml

```
server.port=8443
server.secure=true
server.scheme=https
server.ssl.enabled=true
server.ssl.key-store=/path/to/keystore/bitbucket.jks
server.ssl.key-store-password=<password value>
server.ssl.key-password=<password value>
```

To achieve the same configuration in Bitbucket Server 5.0 and later, add these entries to the `bitbucket.properties` file:

bitbucket.properties

```
server.port=8443
server.secure=true
server.scheme=https
server.ssl.enabled=true
server.ssl.client-auth=want
server.ssl.protocol=TLSv1.2
server.ssl.key-store=/path/to/keystore/bitbucket.jks
server.ssl.key-store-password=<password value>
server.ssl.key-password=<password value>
```

i Both `ssl.key-store-password` and `ssl.key-password` require explicit configuration. Even if you did not configure both values in the `server.xml` file, values for both entries need to be explicitly set in `bitbucket.properties` to secure Bitbucket with SSL.

Bitbucket Server secured with SSL and additional connector to redirect HTTP requests

For some with an SSL-terminating connector configured, you might also have an additional connector that redirects HTTP requests to the HTTPS connector. This would have required you to also specify an additional attribute in the `<Bitbucket installation directory>/atlassian-bitbucket/WEB-INF/web.xml` file.

Here's what that configuration might have looked like prior to Bitbucket Server 5.0:

server.xml

```
<Connector port="7990"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  useBodyEncodingForURI="true"
  redirectPort="8443"
  compression="on"
  compressableMimeType="text/html,text/xml,text/plain,text/css,application/json,application
/javascript,application/x-javascript"/>
```

web.xml

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Restricted URLs</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

To achieve the same configuration in Bitbucket Server 5.0 and later, add these entries to the `bitbucket.propertiesfile`:

bitbucket.properties

```
server.require-ssl=true
server.additional-connector.1.port=7990
server.additional-connector.1.redirect-port=8443
```

For these examples, if there are less properties in the `bitbucket.properties` syntax than in the `initialserver.xml` syntax, that indicates the default value would be acceptable and you don't need to add that property to achieve the same result.

Bitbucket Data Center upgrade guide

This page contains instructions for **upgrading an existing Bitbucket cluster** to the next major or minor version. Upgrading to either version type will incur downtime.

If you are not running Bitbucket in a cluster, follow the instructions in [Bitbucket Server upgrade guide](#).

Upgrading to any later version is free if you have current software maintenance. See our [Licensing FAQ](#) to find out more.

 If you deployed Bitbucket Data Center on AWS using our [AWS Quick Starts](#), refer to the **Upgrading** section of [Administer Bitbucket Data Center in AWS](#) instead.

On this page:

[Plan your upgrade](#)

- [1. Determine your upgrade path](#)
- [2. Complete pre-upgrade checks](#)
- [3. Upgrade Bitbucket in a test environment](#)

[Upgrade Bitbucket Data Center](#)

- [1. Back up your data](#)
- [2. Stop the cluster](#)
- [3. Download Bitbucket](#)
- [4. Upgrade the first node](#)
- [5. Copy Bitbucket to remaining nodes](#)
- [6. Start Bitbucket and check cluster connectivity](#)

Plan your upgrade

1. Determine your upgrade path

Check the [Supported platforms](#) page to determine if your environment meets the minimum requirements to run the latest version of Bitbucket. Also read the [End of support announcements](#).

You can update from any previous version of Bitbucket Server (or Stash) to the latest version, as there is no required upgrade path.

Bitbucket 7 is a major upgrade

Be sure to read the [Bitbucket Server 7.0 release notes](#), take a full backup, and test your upgrade in a non-production environment before upgrading your production site.

Upgrading from a version older than Bitbucket 7 disables all user-installed apps on startup

Be sure to update your own apps and check the [Atlassian Marketplace](#) to ensure 3rd-party apps are compatible with Bitbucket Server 7 before upgrading.

✔ Upgrading to the next bug fix update

If you are upgrading a multi-node Bitbucket cluster to the next bug fix update (for example, from 7.9.0 to 7.9.4), you can [upgrade with no downtime](#).

2. Complete pre-upgrade checks

1. Check the [Version specific upgrade notes](#) for the version you plan to upgrade to (and any in between).
2. Go to  **Support Tools**, then review the Log analyzer for any issues that may need to be resolved.
3. Check the compatibility of your apps with the version you plan to upgrade to.

- a. Go to  **Manage apps** > **Bitbucket update check**.
- b. Choose the version you plan to upgrade to then hit **Check**.

If your users rely on particular apps, you may want to wait until they are compatible before upgrading Bitbucket. App vendors generally update their apps very soon after a major release.

Good to know:

- Upgrading from a version older than Bitbucket Server 6 will automatically disable all user-installed apps. You will need to enable your apps after successfully upgrading, regardless of compatibility with Bitbucket Server 6.
- You can disable an app temporarily if it is not yet compatible.

3. Upgrade Bitbucket in a test environment

1. Create a staging copy of your current production environment.
2. Follow the steps below to upgrade your test environment.
3. Test any unsupported apps, customizations and proxy configuration (if possible) before upgrading your production environment.

Upgrade Bitbucket Data Center

1. Back up your data

1. Determine which backup strategy to use.

For **Bitbucket Server** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), the [Backup Client](#) (version 2.7 or later), or [DIY Backup](#) (version 2.12 or later) while Bitbucket is running, or just stop Bitbucket and zip up / snapshot the home directory and database and keep them somewhere safe.

For **Bitbucket Data Center** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), [DIY Backup](#), or take snapshots of the shared home directory (on NFS) and database while all nodes are stopped.

For **Bitbucket mirrors**, the home directory doesn't store any persistent state that can't be reconstructed from the primary Bitbucket instance, but you should still make sure you have a backup of at least the important configuration files such as SSL certificate, `server.xml`, `config/ssh-server-keys.pem`, `bitbucket.properties` file, and so on in a safe place. See [How do I back up my mirrors?](#) for more information.

See the article [Data recovery and backups](#) for detailed information and guidance on creating an effective backup strategy.

2. Back up all the data in your [Bitbucket home directory](#) and [external database](#).

2. Stop the cluster

You must stop all the nodes in the cluster before upgrading. For more info, see [Starting and stopping Bitbucket](#).

We recommend configuring your load balancer to redirect traffic away from Bitbucket until the upgrade is complete on all nodes.

3. Download Bitbucket

Download the appropriate file for your operating system - <https://www.atlassian.com/software/bitbucket/download-archives>.

4. Upgrade the first node

To upgrade the first node:

1. Extract (unzip) the files to a directory (this will be your new installation directory, and must be different to your existing installation directory)
2. Update the value of `BITBUCKET_HOME` in the `<Installation-directory>/bin/set-bitbucket-home.sh` file so the new Bitbucket installation points to your existing Bitbucket home directory.

i If you use a `BITBUCKET_HOME` environment variable to specify the home directory location, no change is required

3. Copy any other immediately required customizations from the old version to the new one (for example if you are not running Bitbucket on the default ports or if you manage users externally, you'll need to update / copy the relevant files)

i If you configured Bitbucket to run as a Linux service, don't forget to update its service configuration as well. [Learn more about running Bitbucket as a Linux service](#).

4. Start Bitbucket on the node. [Learn more about starting Bitbucket](#)

5. Copy Bitbucket to remaining nodes

The next step is to replicate your upgraded Bitbucket directories to other nodes in the cluster.

1. Stop Bitbucket on the first node.
2. Copy the installation directory and local home directory from the first node to the next node. If the path to the local home directory is different on this node, edit the `bin/set-bitbucket-home.sh` (or `set-bitbucket-home.bat` on Windows) to point to the correct location.
3. Start Bitbucket, and confirm that everything works as expected.
4. Stop Bitbucket on this node before continuing with the next node.

Repeat this process for each remaining node.

6. Start Bitbucket and check cluster connectivity

Once all nodes have been upgraded you can start Bitbucket Data Center on each node, **one at a time** (starting up multiple nodes simultaneously can lead to serious failures).

The Cluster monitoring console (**Administration > Settings > Clustering**) includes information about the active cluster nodes. When the cluster is running properly, you should be able to see the details of each node.

Use Bitbucket in the enterprise

This page describes best practices for using Bitbucket Data Center and Server in enterprise environments. If you're evaluating Bitbucket, we suggest that you begin with [Install a Bitbucket trial](#), instead of this page.

Bitbucket is the Git code management solution for enterprise teams. It allows everyone in your organization to easily collaborate on your Git repositories, while providing enterprise-grade support for:

- user authentication
- repository security
- integration with existing databases and dev environment.

Atlassian offers two deployment options for Bitbucket.

Bitbucket Server

For most organizations, a single instance of Bitbucket Server provides good performance. Continue reading this page for guidance on best practices in setting up a Bitbucket Server instance in a production environment.

Bitbucket Data Center

For larger enterprises that require high availability and greater performance at scale, [Bitbucket Data Center resources](#) uses a cluster of Bitbucket Server nodes to provide Active/Active failover, and is the deployment option of choice.

Feature comparison

Here's a summary of what you get with each license.

Core features	Server license	Data Center license
Integrated CI/CD Two new ways to get feedback on code - the Builds page and the Builds tab - designed to help you avoid switching tools.		
Collaboration via pull requests Review and discuss your code with your team before merging changes. Learn more about pull request collaboration		
Branch permissions Control what users can do on a single branch, branch type, or branch pattern within a repository or project. Learn more about branch permissions		
Flexible workflows Use centralized, forking, gitflow or forking workflows. Learn more about flexible workflows		
Git LFS (Large File Storage) Store large files without the need for an external object store. Learn more about Git LFS		
CI and issue tracker integrations Connect Bitbucket to Jira, Bamboo, Source Tree for a great developer experience. Learn more about integrations		

User management		
Just-in-time user provisioning Allow users to be created and updated automatically when they log in through SAML, SSO, or OpenID Connect SSO. Learn more about JIT user provisioning		
Single sign-on with OpenID Connect Enable seamless integrations with many 3rd party identity providers. Learn more about OpenID Connect		
External user directories Store users in Active Directory, Crowd, Jira or another LDAP directory. Learn more about external user directories		
SAML single sign-on Use a SAML identity provider for authentication and single-sign on. Learn more about SAML single sign-on		
High availability and performance at scale		
Clustering Run Bitbucket on multiple nodes for high availability. Learn more about clustering		
Smart mirroring Improve Git clone speeds for distributed teams working with large repositories. Learn more about smart mirroring		
Content Delivery Network (CDN) support Improve geo-performance for distributed teams. Learn more about CDN		
Elasticsearch Connect Bitbucket to a remote Elasticsearch instance for improved scalability (required for Data Center sites). Learn more about Elasticsearch		
Infrastructure and Control		
Auditing Brings improvements to auditing for projects and repositories, and expanded it to the global admin settings that displays all events in the system. Having this digital record gives admins the higher level of visibility that they need to ensure security and compliance. Learn more about auditing		
Advanced auditing Gives you the visibility and security-relevant digital records needed to help demonstrate compliance, strengthen security, and improve workflow. Learn more about advanced auditing		
App diagnostics Get an overview of the health of your site, including potential performance issues relating to third-party apps. Learn more about diagnostics		
Integrity tests for zero-downtime backup Find and resolve any inconsistencies between the database and home directory, for example after restoring a backup. Learn more about integrity checks		
Rate limiting Control how many external REST API requests users and automations can make. Learn more about rate limiting		
Repository delete policies Secure your data via a new Global Permissions policy. Learn more about policies		

Deployment options		
Your own hardware Run Bitbucket on your own physical servers, virtualized servers, or in the data center of your choice.		
AWS Quick Start and Cloud Formation Templates Use our Cloud Formation Templates to deploy Bitbucket on AWS. Learn more about AWS quick start		

Platform requirements

Although Bitbucket can be run on Windows, Linux and Mac systems, for enterprise use we only recommend, and support using Linux. This recommendation is based on our own testing and experience with using Bitbucket.

See the [Supported platforms](#) page for details of the supported versions of Java, external databases, web browsers and Git.

See [Installing Bitbucket Data Center](#) for detailed information about Bitbucket Data Center requirements.

Performance considerations

In general, Bitbucket is very stable and has low memory consumption. There are no scalability limits other than for Git hosting operations (clone in particular). We know this is the scalability limit of the product; the limit is proportional to the number of cores on the system.

As an example, data collected from an internal Bitbucket instance indicate that for a team of approximately 50 developers, with associated continuous integration infrastructure, we see a peak concurrency of 30 simultaneous clone operations and a mean of 2 simultaneous clone operations. We conservatively expect that a customer with similar usage patterns would be capable of supporting 1000 users on a machine with 40 cores and a supporting amount of RAM. While we expect a peak concurrency larger than 40, Bitbucket is designed to queue incoming requests so as to avoid overwhelming the server.

Bitbucket Server see [Bitbucket Server production server data](#) for data from the Bitbucket Server production instance we run internally at Atlassian.

Bitbucket Data Center see [Bitbucket Data Center Performance](#) for the results of our performance testing for clusters of different sizes.

High availability

If Bitbucket is a critical part of your development workflow, maximizing Bitbucket availability becomes an important consideration.

Bitbucket see [High availability for Bitbucket](#) for the background information you need to set up Bitbucket in a highly available configuration.

Bitbucket Data Center see [Failover for Bitbucket Data Center](#) for information about how Bitbucket Data Center provides HA and almost instant failover.

Scalability

Bitbucket is built with enterprise scaling and infrastructure flexibility in mind, giving administrators control over how Bitbucket fits into their environment:

- For most organizations, a single instance of Bitbucket provides good performance. Continue reading this page for guidance on best practice in setting up a Bitbucket instance in a production environment.
- For larger enterprises that require HA and greater performance at scale, [Bitbucket Data Center](#) uses a cluster of Bitbucket nodes and is the deployment option of choice.

Your single instance of Bitbucket can be easily upgraded to Bitbucket Data Center when the time comes.

Bitbucket see [Scaling Bitbucket Server](#) for information about how you can tune your Bitbucket instance to grow with your organization's needs. See also [Scaling Bitbucket Server for Continuous Integration performance](#) for information specific to Bitbucket performance when CI tools poll Bitbucket for changes.

Bitbucket Data Center see [Adding and removing Data Center nodes](#) for information about how you can rapidly provision extra capacity without downtime.

Provisioning

Some possible approaches to provisioning Bitbucket include:

- [Run the Bitbucket installer](#) in either console or unattended mode
- [Bitbucket Data Center resources](#) - clustered Bitbucket
- [Docker container image for Bitbucket Server](#)

Setting up a production environment

When setting up Bitbucket for a production or enterprise environment, we highly recommend that you configure the following aspects:

Run Bitbucket as a dedicated user

- For production environments Bitbucket should be run from a dedicated user account with restricted privileges. See [Running Bitbucket Server with a dedicated user](#).

Install Bitbucket as a service

- See [Run the Bitbucket installer](#).

Use an external database

- For production environments Bitbucket should use an external database, rather than the embedded database. Set up your external DBMS (for example MySQL) before starting Bitbucket for the first time. This allows you to connect Bitbucket to that DBMS using the Setup Wizard that launches when you first run Bitbucket. See [Connect Bitbucket to an external database](#).

Connect to your existing user directory

- Connect Bitbucket to your existing user directory (for example Active Directory). See [External user directories](#).

Secure the Bitbucket home directory

- For production environments the Bitbucket home directory should be secured against unauthorized access. See [Set the home directory](#).

Secure Bitbucket with HTTPS

- Access to Bitbucket should be secured using HTTP over SSL, especially if your data is sensitive and Bitbucket is exposed to the internet. See [Securing Bitbucket Server with HTTPS](#).

Enable SSH access to Git repositories

- Enable SSH access for your Bitbucket users to Git repositories in Bitbucket so that they can add their own SSH keys to Bitbucket, and then use those SSH keys to secure Git operations between their computer and the Bitbucket instance. See [Enable SSH access to Git repositories](#).

Change the context path for Bitbucket

- If you are running Bitbucket behind a proxy, or you have another Atlassian application (or any Java web application), available at the same hostname and context path as Bitbucket, then you should set a unique context path for Bitbucket. See [Change Bitbucket's context path](#).

Administering a production environment

Upgrading Bitbucket

- For production environments we recommend that you test the Bitbucket upgrade on a QA server before deploying to production. See the [Bitbucket Server upgrade guide](#).

Backups and recovery

- **We highly recommend** that you establish a data recovery plan that is aligned with your company's policies. See [Data recovery and backups](#) for information about tools and backup strategies for Bitbucket.

Logging

- Bitbucket instance logs can be found in `<Bitbucket home directory>/log`. Logs for the bundled Tomcat webserver can be found in `<Bitbucket Server installation directory>/log`. See [Enable debug logging](#).
- Bitbucket displays recent audit events for each repository and project (only visible to Bitbucket admins and system admins), and also creates full audit log files that can be found in the `<Bitbucket home directory>/audit/logs` directory. Note that Bitbucket has an upper limit to the number of log files it maintains, and deletes the oldest file when a new file is created we recommend an automated backup of log files. See [View and configure the audit log](#).]

Upgrade Bitbucket without downtime

To upgrade Bitbucket with no downtime, you'll need to perform a rolling upgrade. A rolling upgrade allows you to upgrade Bitbucket Data Center to a later bug fix version with no downtime. This involves upgrading nodes individually within a multi-node cluster. When you take a node offline to upgrade it, other active nodes keep your Bitbucket site available to users. This allows you to avoid downtime.



Bug fix upgrades only

Bitbucket only allows rolling upgrades for bug fix upgrades (for example, from Bitbucket 7.9.0 to 7.9.4). For platform upgrades (such as Bitbucket 7 to 8) or feature upgrades (such as 7.4 to 7.8), see [Bitbucket Data Center upgrade guide](#) instead.

[Learn more about the different types of upgrades/versions](#)

On this page

- [Before you begin](#)
- [Prepare for the rolling upgrade](#)
- [Perform the rolling upgrade](#)

Bitbucket Data Center features an upgrade mode that allows Bitbucket nodes with different bug fix versions to form a cluster. Upgrade mode lets you upgrade a node to a later bug fix version and then re-connect it to your cluster, letting you take another node offline to upgrade it.

Before you begin

Before you start planning a rolling upgrade, there are a few questions you need to answer.

Does my Bitbucket deployment support rolling upgrades?	You can only perform a rolling upgrade with no downtime on a multi-node Bitbucket cluster. Clustering is only supported on a Bitbucket Data Center license. In addition, a rolling upgrade involves enabling upgrade mode, which is only available in Bitbucket Data Center. Learn more about multi-node clustering in Bitbucket
Do I have enough nodes to support user requests during the rolling upgrade?	You need to take a node offline to upgrade it. During this time, other active nodes will take over the offline nodes workload. Make sure you have enough active nodes to handle user traffic at any given time. Whenever possible, add a node temporarily to your cluster to compensate for offline nodes.

Prepare for the rolling upgrade

1. Complete pre-upgrade checks

1. Check the [Version specific upgrade notes](#) for the version you plan to upgrade to (and any in between).
2. Go to  **Support Tools**, then review the Log analyzer for any issues that may need to be resolved.
3. Check the compatibility of your apps with the version you plan to upgrade to.
 - a. Go to  **Manage apps** > **Bitbucket update check**.
 - b. Choose the version you plan to upgrade to then hit **Check**.

If your users rely on particular apps, you may want to wait until they are compatible before upgrading Bitbucket. App vendors generally update their apps very soon after a major release.

Good to know:

- Upgrading from a version older than Bitbucket Server 6 will automatically disable all user-installed apps. You will need to enable your apps after successfully upgrading, regardless of compatibility with Bitbucket Server 6.
- You can disable an app temporarily if it is not yet compatible.

2. Prevent the installation or upgrade of apps during the upgrade period

If you manage Bitbucket with a team of admins, schedule the rolling upgrade with them. Notify them to postpone any app installs or upgrades until after the rolling upgrade. Installing or upgrading apps during a rolling upgrade could result in unexpected errors.

3. Back up Bitbucket

1. Determine which backup strategy to use.

For **Bitbucket Server** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), the [Backup Client](#) (version 2.7 or later), or [DIY Backup](#) (version 2.12 or later) while Bitbucket is running, or just stop Bitbucket and zip up / snapshot the home directory and database and keep them somewhere safe.

For **Bitbucket Data Center** (version 4.8 or later) instances, you can use [Zero Downtime Backup](#), [DIY Backup](#), or take snapshots of the shared home directory (on NFS) and database while all nodes are stopped.

For **Bitbucket mirrors**, the home directory doesn't store any persistent state that can't be reconstructed from the primary Bitbucket instance, but you should still make sure you have a backup of at least the important configuration files such as SSL certificate, `server.xml`, `config/ssh-server-keys.pem`, `bitbucket.properties` file, and so on in a safe place. See [How do I back up my mirrors?](#) for more information.

See the article [Data recovery and backups](#) for detailed information and guidance on creating an effective backup strategy.

2. Back up all the data in your [Bitbucket home directory](#) and [external database](#).

If your deployment is hosted on AWS, we recommend that you use the AWS native backup facility, which utilizes snapshots to back up your site. For more information, see [AWS Backup](#).

4. Set up a staging environment to test the rolling upgrade

We strongly recommend that you perform the rolling upgrade on a staging or test environment first.

1. Create a staging copy of your current production environment.
2. Follow the steps below to upgrade your test environment.
3. Test any unsupported apps, customizations and proxy configuration (if possible) before upgrading your production environment.

Perform the rolling upgrade

There are three methods for performing a rolling upgrade, depending on what orchestration tools your deployment uses:

Method	Description	Instructions
Manual upgrade	A manual upgrade is suitable for deployments that feature minimal orchestration, particularly in node upgrades. If your deployment is based on our Azure templates , you'll also need to perform a manual upgrade.	Upgrade a Bitbucket cluster manually without downtime
AWS CloudFormation	If your deployment is defined by an AWS CloudFormation template (like our AWS Quick Start), then you can use the same template to orchestrate your upgrade.	

		Upgrade a Bitbucket cluster on AWS without downtime
API-driven	You can orchestrate the entire rolling upgrade process through API calls.	Upgrade a Bitbucket cluster through the API without downtime

Upgrade a Bitbucket cluster manually without downtime

This document provides step-by-step instructions on how to perform a rolling upgrade on deployments with little or no automation. These instructions are also suitable for deployments based on our [Azure templates](#).



For an overview of rolling upgrades (including planning and preparation information), see [Upgrade Bitbucket without downtime](#).

On this page:

[Step 1: Download upgrade files](#)

[Step 2: Enable upgrade mode](#)

[Step 3: Upgrade the first node](#)

[Step 4: Upgrade all other nodes individually](#)

[Step 5: Finalize the upgrade](#)

[Troubleshooting](#)

[Node errors during rolling upgrade](#)

[Disabling upgrade mode](#)

[Rolling back to the original version](#)

[Traffic is disproportionately distributed during or after upgrade](#)

[Disabling upgrade mode](#)

[Rolling back to the original version](#)

[Traffic is disproportionately distributed during or after upgrade](#)

[Disabling upgrade mode](#)

[Rolling back to the original version](#)

[Traffic is disproportionately distributed during or after upgrade](#)

Step 1: Download upgrade files

Before you start the upgrade, download the right Bitbucket version first. You'll be installing this on each node. Remember, you can only upgrade to a higher bug fix version (for example, from Bitbucket 7.9.0 to 7.9.4). Download the file directly from here:

<https://www.atlassian.com/software/bitbucket/download>

Alternatively, you can also use the [Pre-upgrade planning tool](#) to help you download a compatible bug fix version. Choose  **Administration** > **Plan your upgrade** to open the tool.

Step 2: Enable upgrade mode

1. Go to  **Administration** > **Rolling upgrades**.
2. Select the Upgrade mode toggle (1).

Rolling upgrades

A rolling upgrade lets you avoid downtime during bug fix upgrades. Head to [Plan your upgrade](#) to view or download compatible versions. [Learn more about rolling upgrades](#)

Upgrade mode

Upgrade mode lets your cluster accept nodes with different bug fix versions. Enable upgrade mode first before upgrading each node. To avoid downtime, make sure you have enough active nodes to handle traffic at all times.

Upgrade mode Cluster status **READY TO UPGRADE** You can now start upgrading each node. Refer to our [documentation](#) on how to do this.

Cluster overview

Node ID	Name	IP address	Version	Node status
● dd55d932-cbd3-4c90-bab2-11c4f87236a9	Not configured	bb1	7.9.0	ACTIVE
02d3dd9b-631c-4568-85bf-afef3662e15f	Not configured	bb2	7.9.0	ACTIVE
88ee8ee0-ed27-455a-b259-2d793d3feee0	Not configured	bb3	7.9.0	ACTIVE

Once all nodes are upgraded, finalize the upgrade. This will disable upgrade mode.

Enabling upgrade mode allows your cluster to accept nodes running a later bug fix version. This lets you upgrade one node and let it rejoin the cluster (along with the other non-upgraded nodes). Both upgraded and non-upgraded active nodes work together in keeping Bitbucket available to all users.

You can disable Upgrade mode as long as you haven't upgraded any nodes yet.

Step 3: Upgrade the first node

For most environments, upgrading a node during a rolling upgrade consists of four phases:

Start with the least busy node

We recommend that you start upgrading the node with the least number of running tasks and active users. This will typically be the node with the lowest amount of CPU usage.

When you disconnect a node from the load balancer, user requests will no longer be routed to the node. The following table provides guidance how to do so for popular load balancers:

NGINX	NGINX defines groups of cluster nodes through the <code>upstream</code> directive. To prevent the load balancer from connecting to a node, delete the node's entry from its corresponding <code>upstream</code> group. Learn more about the <code>upstream</code> directive in the <code>ngx_http_upstream_module</code>
HAProxy	With HAProxy, you can disable all traffic to the node by putting it in a <code>maint</code> state: <pre>set server <node IP or hostname> state maint</pre> Learn more about forcing a server's administrative state
Apache	You can disable a node (or "worker") by setting its <code>activationmember</code> attribute to <code>disabled</code> . Learn more about advanced load balancer worker properties in Apache
Azure Application Gateway	We provide a deployment template for Bitbucket Data Center on Azure ; this template uses the Azure Application Gateway as its load balancer. The Azure Application Gateway defines each node as a target within a backend pool. Use the Edit backend pool interface to remove your node's corresponding entry. Learn more about adding (and removing) targets from a backend pool

With upgrade mode enabled, you can now upgrade your first node. Start by shutting down Bitbucket gracefully on the node:

1. Access the node through a command line or SSH.
2. Shut down Bitbucket gracefully on the node. This will provide Bitbucket with some time to finish all of its tasks first before going offline. If you [installed Bitbucket manually](#), run the `bin/stop-bitbucket.sh` script to gracefully shut down Bitbucket. [Learn more about gracefully shutting down Bitbucket](#)
3. Wait for the node to go offline. You can monitor its status on the Node status column of the Rolling upgrade pages Cluster overview section.

Once the nodes status is offline, you can start upgrading the node. Copy the Bitbucket files you downloaded (from [Step 1: Download upgrade files](#) section) to the nodes local file system:

To upgrade the first node:

1. Extract (unzip) the files to a directory (this will be your new installation directory, and must be different to your existing installation directory)
2. Update the value of `BITBUCKET_HOME` in the `<Installation-directory>/bin/set-bitbucket-home.sh` file so the new Bitbucket installation points to your existing Bitbucket [home directory](#).

 If you use a `BITBUCKET_HOME` environment variable to specify the home directory location, no change is required

3. Copy any other immediately required customizations from the old version to the new one (for example if you are not running Bitbucket on the default ports or if you manage users externally, you'll need to update / copy the relevant files)

 If you configured Bitbucket to run as a Linux service, don't forget to update its service configuration as well. [Learn more about running Bitbucket as a Linux service](#).

4. Start Bitbucket on the node. [Learn more about starting Bitbucket](#)

After Bitbucket starts successfully on the node, reconnect it to the load balancer. This will allow the node to rejoin the cluster. Wait for it to show up in the **Cluster overview** with an **Active** status. As soon as the first upgraded node joins the cluster, your cluster status will transition to Mixed. This means that you won't be able to disable Upgrade mode until all nodes are running the same version.

Step 4: Upgrade all other nodes individually

After starting the upgraded node, wait for it to show up on the Cluster overview with an Active status. When it does, you can start upgrading another node using the instructions from the previous step. Do this for each remaining node as always, we recommend that you upgrade the node with the lowest level of CPU activity.

Step 5: Finalize the upgrade

Once all nodes are Active and running the same upgraded version, the Finalize upgrade button will become enabled. Click this to complete the rolling upgrade.

After completing the rolling upgrade, you should:

- [Update your apps accordingly](#)
- Perform UAT and other tests as needed

Troubleshooting

Node errors during rolling upgrade

If a node's status transitions to **Error**, it means something went wrong during the upgrade. You can't finish the rolling upgrade if any node has an **Error** status. However, you can still disable Upgrade mode as long as the cluster status is still **Ready to upgrade**.

There are several ways to address this:

- Shut down Bitbucket gracefully on the node. This should disconnect the node from the cluster, allowing the node to transition to an **Offline** status.
- If you can't shut down Bitbucket gracefully, shut down the node altogether.

Once all active nodes are upgraded with no nodes in Error, you can finalize the rolling upgrade. You can investigate any problems with the problematic node afterwards and re-connect it to the cluster once you address the error.

Disabling upgrade mode

You can disable Upgrade mode as long as all nodes in the cluster:

- haven't been upgraded yet
- aren't in an Error state

The cluster's status will transition to Mixed if an upgraded node joins the cluster or a node enters an error state.



Mixed status with Upgrade mode disabled

If a node is in an Error state with Upgrade mode disabled, you can't enable Upgrade mode. Fix the problem or remove the node from the cluster to enable Upgrade mode.

Rolling back to the original version

To roll back upgraded nodes to their original version:

1. Access the node through a command line or SSH.
2. Shut down Bitbucket gracefully on the node.
3. Wait for the node to go offline. You can monitor its status on the Node status column of the Rolling upgrade pages Cluster overview section.
4. Start Bitbucket on the node from your old installation directory. You should not see the setup wizard.
5. If you configured Bitbucket to run as a Linux service, don't forget to update its service configuration as well. [Learn more about running Bitbucket as a Linux service.](#)

Once all nodes are running the same version, the cluster's status will revert back to Ready to upgrade. This will also allow you to disable Upgrade mode.

Traffic is disproportionately distributed during or after upgrade

Some load balancers might use strategies that send a disproportionate amount of active users to a newly-upgraded node. When this happens, the node might become overloaded, slowing down Bitbucket for all users logged in to the node.

To address this, you can also temporarily disconnect the node from the cluster. This will force the load balancer to re-distribute active users between all other available nodes. Afterwards, you can add the node again to the cluster.

Upgrade a Bitbucket cluster on AWS without downtime

This document provides step-by-step instructions on performing a rolling upgrade on an AWS deployment orchestrated through CloudFormation. In particular, these instructions are suitable for Bitbucket Data Center deployments based on our [AWS Quick Starts](#).

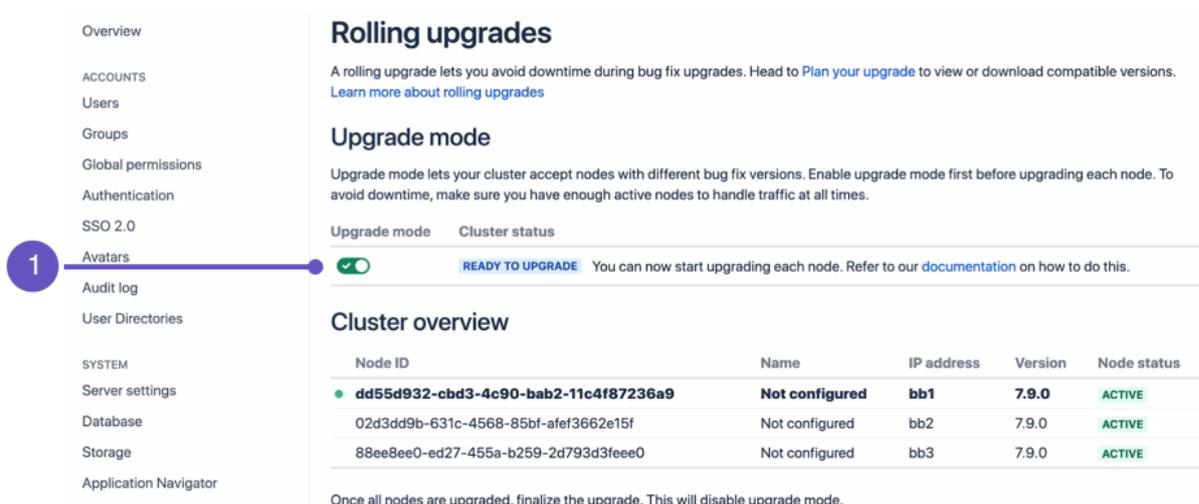
✔ For an overview of rolling upgrades (including planning and preparation information), see [Upgrade Bitbucket without downtime](#).

On this page:

- [Step 1: Enable upgrade mode](#)
- [Step 2: Find all the current application nodes in your stack](#)
- [Step 3: Update your CloudFormation template](#)
- [Step 4: Upgrade another node](#)
- [Step 5: Upgrade all other nodes individually](#)
- [Step 6: Finalize the upgrade](#)
 - [Scale down cluster](#)
 - [Troubleshooting](#)
 - [Disconnect a node from the cluster through the load balancer](#)
 - [Node errors during rolling upgrade](#)
 - [Disabling upgrade mode](#)

Step 1: Enable upgrade mode

1. Go to  **Administration > Rolling upgrades**.
2. Select the Upgrade mode toggle (1).



Rolling upgrades

A rolling upgrade lets you avoid downtime during bug fix upgrades. Head to [Plan your upgrade](#) to view or download compatible versions. [Learn more about rolling upgrades](#)

Upgrade mode

Upgrade mode lets your cluster accept nodes with different bug fix versions. Enable upgrade mode first before upgrading each node. To avoid downtime, make sure you have enough active nodes to handle traffic at all times.

Upgrade mode Cluster status **READY TO UPGRADE** You can now start upgrading each node. Refer to our [documentation](#) on how to do this.

Cluster overview

Node ID	Name	IP address	Version	Node status
● dd55d932-cbd3-4c90-bab2-11c4f87236a9	Not configured	bb1	7.9.0	ACTIVE
02d3dd9b-631c-4568-85bf-afef3662e15f	Not configured	bb2	7.9.0	ACTIVE
88ee8ee0-ed27-455a-b259-2d793d3feee0	Not configured	bb3	7.9.0	ACTIVE

Once all nodes are upgraded, finalize the upgrade. This will disable upgrade mode.

Enabling upgrade mode allows your cluster to accept nodes running a later bug fix version. This lets you upgrade one node and let it rejoin the cluster (along with the other non-upgraded nodes). Both upgraded and non-upgraded active nodes work together in keeping Bitbucket available to all users.

You can disable Upgrade mode as long as you haven't upgraded any nodes yet.

Step 2: Find all the current application nodes in your stack

1. In the AWS console, go to **Services > CloudFormation**. Select your deployments stack to view its Stack Details.
2. Expand the **Resources** drop-down. Look for the **ClusterNodeGroup** and click its Physical ID. This will take you to a page showing the Auto Scaling Group details of your application nodes.
3. In the Auto Scaling Group details, click on the **Instances** tab. Note all of the Instance IDs listed there; you'll be terminating them at a later step.

Step 3: Update your CloudFormation template

Your deployment uses a CloudFormation template that defines each component of your environment. In this case, upgrading Bitbucket means updating the version used in the template. During the upgrade, we highly recommend that you add a node temporarily to your cluster as well.

1. In the AWS console, go to **Services > CloudFormation**. Select your deployments stack to view its Stack Details.
2. In the Stack Details screen, click **Update Stack**.
3. From the **Select Templates** screen, select **Use current template** and click **Next**.
4. Set the **Version** parameter to the version you're updating to. Since this is a rolling upgrade, you can only set this to a later bug fix version (for example, from Bitbucket 7.9.0 to 7.9.4).
5. Add an extra node to your cluster. This will help ensure that your cluster won't have a shortage of nodes for user traffic. To do this, increase the value of the following parameters by 1:
 - **Maximum number of cluster nodes**
 - **Minimum number of cluster nodes**
6. Click **Next**. Click through the next pages, and then to apply the change using the **Update** button.

After updating the stack, you will have one extra node already running the new Bitbucket version. With Upgrade mode enabled, that node will be allowed to join the cluster and start work. Your other nodes won't be upgraded yet.

Mixed status

As soon as the first upgraded node joins the cluster, your cluster status will transition to Mixed. This means that you won't be able to disable Upgrade mode until all nodes are running the same version.

Once the new upgraded node is running and in an Active state, you can start upgrading another node. To do that, shut down and terminate the Bitbucket node AWS will then replace the node with a new one running the updated Bitbucket version.

Step 4: Upgrade another node

Start with the least busy node

We recommend that you start upgrading the node with the least number of running tasks and active users. This is typically the node with the lowest CPU utilization. If you deployed Bitbucket with Amazon CloudWatch, you can view each node's CPU utilization through your CloudWatch dashboard. [Learn more about listing available CloudWatch metrics](#)

In [Step 2](#), you noted the instance ID of each node in your cluster. Choose a node to upgrade first, and terminate it:

1. In the AWS console, go to **Services > EC2**. From there, click **Running Instances**.
2. Check the instance matching your chosen node.

3. From the **Actions** drop-down, select **Instance State > Terminate**.
4. Click through to terminate the instance.

Each time you terminate a Bitbucket node, AWS will automatically replace it. The replacement will be running the new version of Bitbucket. Once the new node's status is Active, you can move on to upgrading another node.

 Only terminate the Bitbucket nodes. Do *not* terminate the NFS Server EC2 instance.

Step 5: Upgrade all other nodes individually

At this point, your cluster should have two nodes running the new version of Bitbucket. You can now upgrade other nodes. To do so, simply repeat the previous step on another node. As always, we recommend that you upgrade the node with the lowest CPU utilization each time.

Step 6: Finalize the upgrade

Once all nodes are Active and running the same upgraded version, the Finalize upgrade button will become enabled. Click this to complete the rolling upgrade.

After completing the rolling upgrade, you should:

- [Update your apps accordingly](#)
- Perform UAT and other tests as needed

Scale down cluster

In [Step 3](#), we added a node temporarily to the cluster as a replacement for each one we terminated. This was to help ensure we'd have enough nodes to handle normal user traffic. After finalizing the upgrade, you can remove that node:

1. In the AWS console, go to **Services > CloudFormation**. Select your deployments stack to view its Stack Details.
2. In the Stack Details screen, click **Update Stack**.
3. From the **Select Templates** screen, select **Use current template** and click **Next**.
4. Decrease the value of the following parameters by 1:
 - **Maximum number of cluster nodes**
 - **Minimum number of cluster nodes**
5. Select **Next**. Click through the next pages, and then to apply the change using the **Update** button.

You can now remove one node from your cluster without AWS replacing it. Find the least busy Bitbucket node and terminate it (refer to [Step 4](#) for detailed instructions).

Troubleshooting

Disconnect a node from the cluster through the load balancer

If an error prevents you from terminating a node, try disconnecting the node from the cluster through the load balancer. In the AWS Classic Load Balancer, each node is registered as a target so to disconnect a node, you'll have to de-register it. See [Register or deregister instances](#) and [Configure connection draining](#) for more information.

Node errors during rolling upgrade

If a node's status transitions to **Error**, it means something went wrong during the upgrade. You can't finish the rolling upgrade if any node has an **Error** status. However, you can still disable Upgrade mode as long as the cluster status is still **Ready to upgrade**.

There are several ways to address this:

- Shut down Bitbucket gracefully on the node. This should disconnect the node from the cluster, allowing the node to transition to an **Offline** status.
- If you can't shut down Bitbucket gracefully, shut down the node altogether.

Once all active nodes are upgraded with no nodes in Error, you can finalize the rolling upgrade. You can investigate any problems with the problematic node afterwards and re-connect it to the cluster once you address the error.

Rolling back to the original version

To roll back the upgraded nodes to the original version:

1. In the AWS console, go to **Services > CloudFormation**. Select your deployments stack to view its Stack Details.
2. In the Stack Details screen, click **Update Stack**.
3. From the **Select Template** screen, select **Use current template** and click **Next**.
4. Set the **Version** parameter to your original version.
5. Click **Next**. Click through the next pages, and then to apply the change using the **Update** button.

Afterwards, terminate each Bitbucket node running the new version of Bitbucket. Each time you do, AWS will replace the node with one running the original version. Wait until the AWS replacement node is Active before terminating another node.

Once all nodes are running the same version, the cluster's status will revert back to Ready to upgrade. This will also allow you to disable Upgrade mode.

Disabling upgrade mode

You can disable Upgrade mode as long as all nodes in the cluster:

- haven't been upgraded yet
- aren't in an Error state

The cluster's status will transition to Mixed if an upgraded node joins the cluster or a node enters an error state.

Mixed status with Upgrade mode disabled

If a node is in an Error state with Upgrade mode disabled, you can't enable Upgrade mode. Fix the problem or remove the node from the cluster to enable Upgrade mode.

Upgrade a Bitbucket cluster through the API without downtime

This document provides guidance on how to initiate and finalize a rolling upgrade through API calls. This upgrade method is suitable for admins with the skills and automation tools to orchestrate maintenance tasks (like upgrades).

✔ For an overview of rolling upgrades (including planning and preparation information), see [Upgrade Bitbucket without downtime](#).

On this page:

- [API reference](#)
- [Downloading upgrade files](#)
- [Initiating a rolling upgrade](#)
- [Upgrading each node individually](#)
- [Finalizing the rolling upgrade](#)
- [Node statuses](#)
- [Cluster statuses](#)
- [Troubleshooting](#)
 - [Node errors during rolling upgrade](#)
 - [Disabling upgrade mode](#)

API reference

The entire rolling upgrade process is governed by the following API:

`https://<host>:<port>/rest/zdu/cluster/zdu/`

This API has the following calls:

/zdu	Get an overview of the cluster's status.
/zdu/start	Enable upgrade mode.
/zdu/state	Get the status of the cluster.
/zdu/nodes/{nodeId}	Get an overview of a node's status.
/zdu/cancel	Disable upgrade mode. You can only use this call if the upgrade progress is not MIXED.
/zdu/approve	Once all nodes are upgraded, finalize the rolling upgrade. This will automatically disable upgrade mode.

For detailed information about each API call, see [Bitbucket REST API Documentation](#).

Authenticating REST API calls

In a secure environment, you'll need to authenticate your REST API calls. See [Personal access tokens](#) and [Bitbucket Server REST API Example - Basic Authentication](#) for more information.

Downloading upgrade files

Before you start the upgrade, download the right Bitbucket version first. You'll be installing this on each node. Remember, you can only upgrade to a higher bug fix version (for example, from Bitbucket 7.9.0 to 7.9.4). Download the file directly from here:

<https://www.atlassian.com/software/bitbucket/download>

Alternatively, you can also use the [Pre-upgrade planning tool](#) to help you download a compatible bug fix version. Choose  **Administration** > **Plan your upgrade** to open the tool.

Initiating a rolling upgrade

To initiate a rolling upgrade, enable rolling upgrade first. To do this, use:

```
https://<host>:<port>/rest/zdu/cluster/zdu/start
```

Enabling upgrade mode allows your cluster to accept nodes running a later bug fix version. This lets you upgrade one node and let it rejoin the cluster (along with the other non-upgraded nodes). Both upgraded and non-upgraded active nodes work together in keeping Bitbucket available to all users.

You can disable upgrade mode as long as you havent upgraded any nodes yet.

Upgrading each node individually

In general, upgrading a node during a rolling upgrade consists of four phases:

 **Start with the least busy node**

We recommend that you start upgrading the node with the least number of running tasks and active users. This will typically be the node with the lowest amount of CPU usage.

When you disconnect a node from the load balancer, user requests will no longer be routed to the node. The following table provides guidance how to do so for popular load balancers:

NGINX	NGINX defines groups of cluster nodes through the <code>upstream</code> directive .To prevent the load balancer from connecting to a node, delete the node's entry from its corresponding <code>upstream</code> group. Learn more about the <code>upstream</code> directive in the <code>ngx_http_upstream_module</code>
HAProxy	With HAProxy, you can disable all traffic to the node by putting it in a <code>maint</code> state: <pre>set server <node IP or hostname> state maint</pre> Learn more about forcing a server's administrative state
Apache	You can disable a node (or "worker") by setting its <code>activationmember</code> attribute to <code>disabled</code> . Learn more about advanced load balancer worker properties in Apache
Azure Application Gateway	We provide a deployment template for Bitbucket Data Center on Azure ; this template uses the Azure Application Gateway as its load balancer. The Azure Application Gateway defines each node as a target within a backend pool. Use the Edit backend pool interface to remove your node's corresponding entry. Learn more about adding (and removing) targets from a backend pool

With upgrade mode enabled, you can now upgrade your first node. Start by shutting down Bitbucket gracefully on the node:

1. Access the node through a command line or SSH.

2. Shut down Bitbucket gracefully on the node. This will provide Bitbucket with some time to finish all of its tasks first before going offline. If you installed Bitbucket manually, run the `bin/stop-bitbucket.sh` script to gracefully shut down Bitbucket. [Learn more about gracefully shutting down Bitbucket](#)
3. Wait for the node to go offline. You can monitor its status on the Node status column of the Rolling upgrade pages Cluster overview section.

Once the nodes status is offline, you can start upgrading the node. Copy the Bitbucket files you downloaded (from [Downloading upgrade files](#) section) to the nodes local file system:

To upgrade the first node:

1. Extract (unzip) the files to a directory (this will be your new installation directory, and must be different to your existing installation directory)
2. Update the value of `BITBUCKET_HOME` in the `<Installation-directory>/bin/set-bitbucket-home.sh` file so the new Bitbucket installation points to your existing Bitbucket home directory.

 If you use a `BITBUCKET_HOME` environment variable to specify the home directory location, no change is required

3. Copy any other immediately required customizations from the old version to the new one (for example if you are not running Bitbucket on the default ports or if you manage users externally, you'll need to update / copy the relevant files)

 If you configured Bitbucket to run as a Linux service, don't forget to update its service configuration as well. [Learn more about running Bitbucket as a Linux service.](#)

4. Start Bitbucket on the node. [Learn more about starting Bitbucket](#)

After Bitbucket starts successfully on the node, reconnect it to the load balancer. This will allow the node to rejoin the cluster. As soon as the first upgraded node joins the cluster, your cluster status will transition to Mixed. This means that you won't be able to disable Upgrade mode until all nodes are running the same version.

Finalizing the rolling upgrade

Once all nodes are upgraded, finalize the rolling upgrade. To do this, use:

```
https://<host>:<port>/rest/zdu/cluster/zdu/approve
```

This call will automatically disable upgrade mode.

After completing the rolling upgrade, you should:

- [Update your apps accordingly](#)
- Perform UAT and other tests as needed

Node statuses

To get the status of a node, use:

```
https://<host>:<port>/rest/zdu/cluster/zdu/nodes/<nodeID>
```

ACTIVE

Bitbucket is connected to the cluster and running with no errors.

STARTING	Bitbucket is still loading, and should transition to Active once finished.
TERMINATING	Bitbucket was gracefully shut down, and should transition to Offline once finished.
OFFLINE	Bitbucket is not responding on the node. This node will be removed from the cluster completely if it is still offline after Upgrade mode is disabled.
ERROR	Something went wrong with Bitbucket on the node.

Cluster statuses

To get the status of the cluster, use:

```
https://<host>:<port>/rest/zdu/cluster/zdu/state
```

STABLE	You can turn on Upgrade mode now.
READY_TO_UPGRADE	Upgrade mode is enabled, but no nodes have been upgraded yet. You can start upgrading your first node now.
MIXED	At least one node is upgraded, but you haven't finished upgrading all nodes yet. Your cluster has nodes running different Bitbucket versions. You need to upgrade all nodes to the same bug fix version to transition to the next status (READY_TO_RUN_UPGRADE_TASKS)
READY_TO_RUN_UPGRADE_TASKS	All nodes have node been upgraded. You can now finalize the rolling upgrade: <pre>https://<host>:<port>/rest/zdu/cluster/zdu/approve</pre>

Troubleshooting

Node errors during rolling upgrade

If a nodes status transitions to **Error**, it means something went wrong during the upgrade. You cant finish the rolling upgrade if any node has an **Error** status. However, you can still disable Upgrade mode as long as the cluster status is still **Ready to upgrade**.

There are several ways to address this:

- Shut down Bitbucket gracefully on the node. This should disconnect the node from the cluster, allowing the node to transition to an **Offline** status.
- If you cant shut down Bitbucket gracefully, shut down the node altogether.

Once all active nodes are upgraded with no nodes in Error, you can finalize the rolling upgrade. You can investigate any problems with the problematic node afterwards and re-connect it to the cluster once you address the error.

Disabling upgrade mode

You can disable Upgrade mode as long as all nodes in the cluster:

- haven't been upgraded yet
- aren't in an Error state

The cluster's status will transition to Mixed if an upgraded node joins the cluster or a node enters an error state.

 Mixed status with Upgrade mode disabled

If a node is in an Error state with Upgrade mode disabled, you can't enable Upgrade mode. Fix the problem or remove the node from the cluster to enable Upgrade mode.

Bitbucket Data Center

Data Center is our self-managed edition of Bitbucket built for enterprises. It provides the deployment flexibility and administrative control you need to manage mission-critical Bitbucket sites. Learn more about the [benefits of Bitbucket Data Center](#) on our website.

Server and Data Center features comparison

Want to see what's included with a Data Center license? Head to the [Bitbucket Data Center and Server feature comparison](#).



You can [purchase a Data Center license](#) or create an evaluation license at my.atlassian.com

Data Center deployment options

You can deploy Bitbucket Data Center in two ways:



Non-clustered (single node)

Run Bitbucket Data Center on a single node, just like a Server installation. This option doesn't require any changes to your infrastructure, but it does allow you to take advantage of Data Center-only features. Quick and easy. [Learn more](#)



Clustered

Run Bitbucket Data Center in a cluster with multiple nodes, and a load balancer to distribute traffic. Clustering is designed for large, or mission-critical, Bitbucket instances, allowing you to provide high availability, and maintain performance as you scale. [Learn more](#)

Get started

Non-clustered

- Learn about [non-clustered architecture and requirements](#)
- [Install Bitbucket Data Center from scratch](#)
- [Migrate from Server to Data Center](#)
- [Move from non-clustered to clustered Data Center](#)

Clustered

- Learn about [clustering architecture and requirements](#)
- [Set up a Data Center cluster](#)
- [Add or remove application nodes](#)
- [Revert to a non-clustered Data Center installation](#)
- [Upgrade Bitbucket without downtime](#)

Running Bitbucket Data Center on a single node

You can run Bitbucket Data Center on a single node, just like a Server installation. This is useful if you don't need cluster-specific benefits such as high availability and performance at scale.

Benefits of running a non-clustered Data Center deployment

There are a range of reasons you may choose a single node Data Center. Some of the benefits include:

- **Keeping your existing infrastructure**
Running on a single node means that you can upgrade from Server to Data Center without adding to your infrastructure. In most cases, moving to Data Center will be as simple as updating your license.
- **Accessing Data Center-only features**
Your Data Center license unlocks a suite of additional security, compliance, and administration features to help you easily manage enterprise-grade Bitbucket instance like SAML single sign-on, smart mirroring, rate limiting, and more. [See the complete list](#)

Architecture

The image below shows a typical configuration:



Bitbucket Data Center deployed on a single node looks just as a Server installation, and consists of:

- Bitbucket Data Center, running on a single node
- A database that Bitbucket read and writes to

Requirements

Non-clustered Data Center deployments follow the same minimum requirements as a Server installation. Check our [Bitbucket Supported platforms](#) for more details.

App compatibility

The process for installing Marketplace apps (also known as add-ons or plugins) in Bitbucket Data Center is the same as for Server. You won't have to stop Bitbucket to install or update an app.

The Atlassian Marketplace indicates apps that are compatible with Bitbucket Jira Data Center. [Learn more about Data Center approved apps](#)

Ready to get started?

Deploying Data Center on a single node will be the same as deploying a Server installation, just with a different license. Head to [Bitbucket installation guide](#) to learn about all the ways in which you can install Bitbucket.

Moving back to Server

If you no longer need Data Center, you can go back to a Server installation. This will be as easy as updating your license, but be aware that you'll immediately lose features that are exclusive to Data Center.

How will the Data Center-only features be affected?

After moving from Data Center to Server, you will lose features exclusive to Data Center. Here's the summary of how this will affect your current Bitbucket instance:

Feature	Description
SAML single sign-on Use a SAML identity provider for authentication and single-sign on.	<ul style="list-style-type: none">• SAML login won't work i.e. the users will be redirected to a login screen always• Users who had a password on their Atlassian account before SAML single sign-on was enabled will use that to log in.• Users who joined after SAML single sign-on was enabled will need to reset their password for their Atlassian account when they next log in.
Content Delivery network (CDN) support Improve geo-performance for distributed teams.	<ul style="list-style-type: none">• No longer available• Any API calls related to the feature will be blocked• If you've already configured your CDN, then switch back to a Server license, the CDN will continue to work until your Confluence instance is restarted.
Rate limiting Control how many external REST API requests users and automations can make.	<ul style="list-style-type: none">• No longer available• Any configured limits that were set up before will become irrelevant and your users will have unlimited external REST API requests the moment the license is downgraded
Smart mirroring Improve Git clone speeds for distributed teams working with large repositories.	<ul style="list-style-type: none">▪ No longer available

Before you begin

- This guide assumes that you're using a standalone Bitbucket Data Center. If you have a cluster of nodes, go to [Moving to a non-clustered Data Center](#) first.
- We recommend completing this process in a staging environment, and running a set of functional tests, integration tests, and performance tests, before making these changes in production.

Moving back to Server

1. Get a license for Bitbucket Server

You can [purchase a Server license](#) or create an evaluation license at my.atlassian.com.

2. Apply the license

1. Go to **Administration > Licensing**.
2. Update the license.

After updating the license, your Bitbucket instance will become a Server one.

Clustering with Bitbucket

 This feature is only for customers with an active [Bitbucket Data Center resources](#) license.

Bitbucket Data Center allows you to run a cluster of multiple Bitbucket nodes, providing high availability, scalable capacity, and performance and scale. We'll tell you about the benefits, and give you an overview of what you'll need to run Bitbucket in a clustered environment.

Ready to get started? See [Set up a Bitbucket Data Center cluster](#).

On this page:

- [Benefits of clustering Architecture](#)
 - [Learn more](#)
- [Infrastructure and requirements](#)
 - [Deploying Bitbucket Data Center on AWS and Azure Server requirements](#)
 - [Cluster nodes requirements](#)
 - [Database Shared home and storage requirements](#)
 - [Load balancer](#)
 - [Network adapters](#)
 - [Elasticsearch node](#)
- [App compatibility](#)
- [Ready to get started?](#)

Benefits of clustering

Clustering is designed for enterprises with large or mission-critical Data Center deployments that require continuous uptime, instant scalability, and performance under high load.

Here are some of the benefits:

- **High availability and failover**

If one node in your cluster goes down, the others take on the load, ensuring your users have uninterrupted access to Bitbucket.

- **Performance at scale**

Each node added to your cluster increases concurrent user capacity, and improves response time as user activity grows.

- **Instant scalability**

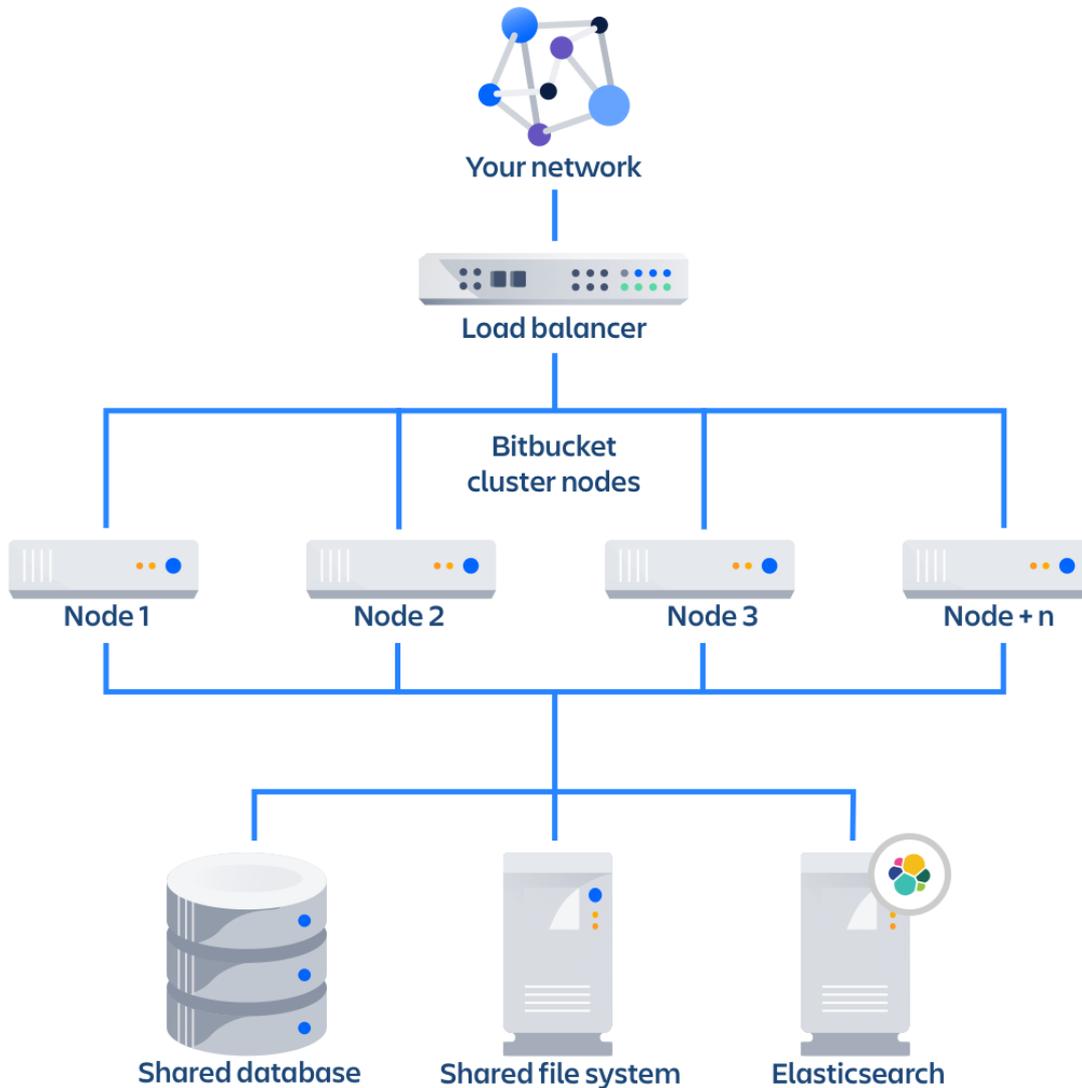
Add new nodes to your cluster without downtime or additional licensing fees. Data and apps are automatically synced.

- **Upgrade with no downtime**

Perform a [rolling upgrade](#) to the latest bug fix update of your feature release, without any downtime. Apply critical bug fixes and security updates to your site while providing users with uninterrupted access to Bitbucket.

Architecture

The image below shows a typical configuration:



A Bitbucket Data Center cluster consists of:

- Multiple identical application nodes running Bitbucket Data Center.
- A load balancer to distribute traffic to all of your application nodes.
- A sharedfilessystem that stores repositories, attachments, and other shared files.
- A database that all nodes read and write to.
- An Elasticsearch instance that enables searching for projects, repositories, and code

All application nodes are active and process requests. A user will access the same Bitbucketnode for all requests until their session times out, they log out, or a node is removed from the cluster.

Learn more

Your Data Center license is based on the number of users in your cluster, rather than the number of nodes. This means you can scale your environment without additional licensing fees for new servers or CPU.

You can monitor the available license seats in the Licensing page in the admin console.

If you wanted to automate this process (for example to send alerts when you are nearing full allocation) you can use the REST API.

Your Bitbucket license determines which features and infrastructure choices are available. Head to [Bitbucket Server and Data Center feature comparison](#) for a full run down of the differences between a Server license and a Data Center license.

To run Bitbucket in a cluster, you'll need an additional home directory, known as the shared home.

Each Bitbucket node has a local home that contains logs, caches, and temporary files. Everything else is stored in the shared home, which is accessible to each Bitbucket node in the cluster.

Here's a summary of what is found in the local home and shared home:

Local home	Shared home
<ul style="list-style-type: none"> logs caches temporary files 	<ul style="list-style-type: none"> configuration files data directory with: <ul style="list-style-type: none"> repositories attachments avatars plugins

When clustered, Bitbucket uses a distributed cache that is managed using Hazelcast. Caches are kept in sync through remote invalidation instead of being replicated or partitioned across all the Bitbucket nodes in a cluster.

Because of this caching solution, to minimize latency, your nodes should be located in the same physical location, or region (for AWS and Azure).

When configuring your cluster nodes you can either supply the IP address of each cluster node, or a multicast address.

If you're using multicast:

Bitbucket will broadcast a join request on the multicast network address. Bitbucket must be able to open a UDP port on this multicast address, or it won't be able to find the other cluster nodes. Once the nodes are discovered, each responds with a unicast (normal) IP address and port where it can be contacted for cache updates. Bitbucket must be able to open a UDP port for regular communication with the other nodes.

A multicast address can be auto-generated from the cluster name, or you can enter your own, during the set-up of the first node.

[Elasticsearch](#) provides search functionality for Bitbucket. It provides a fast, full-text search engine that enables searching for projects, repositories, and code within Bitbucket.

Infrastructure and requirements

The choice of hardware and infrastructure is up to you. Below are some areas to think about when planning your hardware and infrastructure requirements.

Deploying Bitbucket Data Center on AWS and Azure

If you plan to run Bitbucket Data Center on AWS or Azure, you can use our templates to deploy the whole infrastructure. You'll get your Bitbucket Data Center nodes, Elasticsearch, database and storage all configured and ready to use in minutes. For more info, see the following resources:

- [Deploy Bitbucket Data Center in AWS](#)

- [Deploy Bitbucket Data Center in Azure](#)

Server requirements

You should not run additional applications (other than core operating system services) on the same servers as Bitbucket. Running Bitbucket, Jira, and Confluence on a dedicated Atlassian software server works well for small installations but is discouraged when running at scale.

Bitbucket Data Center can be run successfully on virtual machines.

Cluster nodes requirements

Each node does not need to be identical, but for consistent performance we recommend they are as close as possible. All cluster nodes must:

- be a dedicated machine, physical or virtual
- be located in the same data center, or region (for AWS and Azure)
- be connected in a high speed LAN (that is, high bandwidth and low latency)
- have the same OS, Java and application server version. See [Supported platforms](#)
- have the same memory configuration (both the JVM and the physical memory) (recommended)
- be configured with the same time zone (and keep the current time synchronized). Using ntpd or a similar service is a good way to ensure this
- (optional) it's good practice to use a firewall and/or network segregation to make sure that only specific nodes are allowed to connect to a Bitbucket cluster node's [Hazelcast](#) port, which by default is port 5701. This is optional, as password is used to authenticate the nodes

You must ensure the clocks on your nodes don't diverge, as it can result in a range of problems with your cluster.

How many nodes?

Your Data Center license does not restrict the number of nodes in your cluster. The right number of nodes depends on the size and shape of your Bitbucket instance, and the size of your nodes.

See our [Bitbucket Data Center load profiles](#) guide for help sizing your instance. In general, we recommend starting small and growing as you need.

Database

You should ensure your intended database is listed in the current [Supported platforms](#), with one exception: we do **not** support MySQL due to inherent deadlocks that can occur in this database engine at high load. The load on an average cluster solution is higher than on a standalone installation, so it is crucial to use a supported database.

Additional requirements for database high availability

Running Bitbucket Data Center in a cluster removes the application server as a single point of failure. You can also do this for the database through the following supported configurations:

- [Amazon RDS Multi-AZ](#): this database setup features a primary database that replicates to a standby in a different availability zone. If the primary goes down, the standby takes its place.
- [Amazon PostgreSQL-Compatible Aurora](#): this is a cluster featuring a database node replicating to one or more readers (preferably in a different availability zone). If the writer goes down, Aurora will promote one of the writers to take its place.

The **AWS Quick Start deployment option** allows you to deploy Bitbucket Data Center with either one, from scratch. If you want to set up an Amazon Aurora cluster with an existing Bitbucket Data Center instance, refer to [Configuring Bitbucket Data Center to work with Amazon Aurora](#).

Shared home and storage requirements

Bitbucket Data Center requires a high performance shared file system such as a SAN, NAS, RAID server, or high-performance file server optimized for I/O.

- The shared file system must run on a dedicated machine.
- The file system must be available to all cluster nodes via a high-speed LAN (it must be in the same physical data center).
- The shared file system should be accessible via NFS as a single mount point.
- Due to known performance issues, we only support NFSv3 at this time.

Load balancer

You can use the load balancer of your choice. Bitbucket Data Center does **not** bundle a load balancer.

- Your load balancer should run on a dedicated machine.
- Your load balancer must have a high-speed LAN connection to the Bitbucket cluster nodes (that is, high bandwidth and low latency).
- Your load balancer must support **both** HTTP mode (for web traffic) **and** TCP mode (for SSH traffic).
- Terminating SSL (HTTPS) at your load balancer and running plain HTTP from the load balancer to Bitbucket is highly recommended for performance.
- Your load balancer should support "session affinity" (also known as "sticky sessions").
- If you don't have a preference for your load balancer, we provide instructions for haproxy, a popular Open Source software load balancer.

Many load balancers require a URL to constantly check the health of their backends in order to automatically remove them from the pool. It's important to use a stable and fast URL for this, but lightweight enough to not consume unnecessary resources. The following URL returns Jiras status and can be used for this purpose.

URL	Expected content	Expected HTTP status
http://<bitbucketurl>/status	{"state":"RUNNING"}	200 OK

HTTP status code	Response entity	Description
200	{"state":"RUNNING"}	Running normally
500	{"state":"ERROR"}	An error state
503	{"state":"STARTING"}	Application is starting
503	{"state":"STOPPING"}	Application is stopping
200	{"state":"FIRST_RUN"}	Application is running for the first time and has not yet been configured
404		Application failed to start up in an unexpected way (the web application failed to deploy)

Here are some recommendations, when setting up monitoring, that can help a node survive small problems, such as a long GC pause:

- Wait for two consecutive failures before removing a node.
- Allow existing connections to the node to finish, for say 30 seconds, before the node is removed from the pool.

For more info, see [Load balancer configuration options](#) or [Install Bitbucket Data Center](#) (section about configuring the load balancer).

Network adapters

Use separate network adapters for communication between servers. Cluster nodes should have a separate physical network (i.e. separate NICs) for inter-server communication. This is the best way to get the cluster to run fast and reliably. Performance problems are likely to occur if you connect cluster nodes via a network that has lots of other data streaming through it.

Elasticsearch node

Bitbucket Data Center requires a connection to a remote Elasticsearch installation to enable code search. Although code search is not critical for high availability, it is possible to run a cluster of Elasticsearch nodes to achieve high availability for the Bitbucket's code search index. The easiest way to set up and deploy an Elasticsearch cluster for Bitbucket Data Center is to use the Amazon's Elasticsearch service, but you can also set up a remote Elasticsearch instance on your own hardware.

Requirements:

- Bitbucket Data Center currently works with Elasticsearch 5.5.1 - 5.5.3, 6.5.3, and 6.8.6.
- Bitbucket Data Center can have *only one* remote connection to Elasticsearch for your cluster.
- This may be a standalone Elasticsearch installation or a clustered installation behind a load balancer.

For more info, see [Administer code search](#) and [How to install Elasticsearch](#).

App compatibility

The process for installing Marketplace apps (also known as add-ons) in a Bitbucket cluster is the same as for a standalone installation. You will not need to stop the cluster, or bring down any nodes to install or update an app.

The Atlassian Marketplace indicates apps that are compatible with Bitbucket Data Center. [Learn more about Data Center approved apps](#)

Ready to get started?

Head to [Set up a Bitbucket Data Center cluster](#) for a step-by-step guide to enabling and configuring your cluster.

Set up a Bitbucket Data Center cluster

Bitbucket Data Center allows you to run a cluster of multiple Bitbucket nodes, providing high availability, scalable capacity, and performance at scale. This guide walks you through the process of configuring a Data Center cluster on your own infrastructure.

 Not sure if clustering is right for you? Check out [Running Bitbucket Data Center in a cluster](#) for a detailed overview.

Before you begin

Things you should know about when setting up your Data Center:

See our [Supported platforms](#) page for information on the database, Java, and operating systems you'll be able to use. These requirements are the same for Server and Data Center deployments.

You can see a component diagram of a typical Bitbucket Data Center instance, and read about detailed requirements of each component on the page [Bitbucket Data Center requirements](#) and on the [Supported platforms](#) page.

A Bitbucket Data Center instance consists of a cluster of components, each running on a dedicated machine:

- A **cluster of Bitbucket application nodes** all running the same version of Bitbucket Data Center web application. These can be virtual or physical machines, have synchronized clocks (for example, using NTP) and be configured with the identical timezone, are allowed to connect to a Bitbucket cluster node's [Hazelcast](#) port, which by default is port 5701.
- A **load balancer** that supports *both* HTTP mode (for web traffic) and TCP mode (for SSH traffic), and support session affinity ("sticky sessions").
- A **supported external database**, shared and available to all cluster nodes.
- A **shared file system** that is physically located in the same data center, available to all cluster nodes, and accessible by NFS as a single mount point.
- A **remote Elasticsearch instance** with only one remote connection to Bitbucket. The instance may be a standalone Elasticsearch installation or a clustered installation behind a load balancer. For help installing and configuring a remote Elasticsearch instance see our [how to guide](#).

Apps extend what your team can do with Atlassian applications, so it's important to make sure that your team can still use their apps after migrating to Data Center. When you switch to Data Center, you'll be required to switch to the Data Center compatible version of your apps, if one is available.

See [Evaluate apps for Data Center migration](#) for more information.

In this guide we'll use the following terminology:

- **Installation directory:** The directory where you installed Jira.
- **Local home directory:** The home or data directory stored locally on each cluster node (if Jira is not running in a cluster, this is simply known as the home directory).
- **Shared home directory:** The directory you created that is accessible to all nodes in the cluster via the same path.

Set up and configure your cluster

We recommend completing this process in a staging environment, and testing your clustered installation, before moving to production.

Install Bitbucket Data Center on the first application node

First, you'll need to install Bitbucket Data Center on one node:

1. Download the latest installer - www.atlassian.com/software/bitbucket/download.

2. Make the installer executable.

Change to the directory where you downloaded the installer, then execute this command:

```
chmod +x atlassian-bitbucket-x.x.x-x64.bin
```

Where `x.x.x` is the version you downloaded.

3. Run the installer we recommend using `sudo` to run the installer as this will create a dedicated account to run Bitbucket and allow you to run Bitbucket as a service.

To use `sudo` to run the installer execute this command:

```
$ sudo ./atlassian-bitbucket-x.x.x-x64.bin
```

You can also run the installer with root user privileges. Where `x.x.x` is the version you downloaded.

4. Follow the prompts to install Bitbucket. You'll be asked for the following info:
 - a. **Type of Bitbucket instance**- the type of installation, for these instructions select **Data Center**.
 - b. **Installation directory**- where Bitbucket will be installed.
 - c. **Home directory**- where Bitbucket application data will be stored.
 - d. **TCP ports**- the HTTP connector port and control port Bitbucket will run on.

Provision the shared database, filesystem, and Elasticsearch nodes

Once you've installed the first Bitbucket application node, you now need to provision the share database and shared filesystem to use with Bitbucket Data Center.

Step 1. Provision your shared database

Set up your shared database server.

- [Connect Bitbucket to PostgreSQL](#)
- [Connect Bitbucket to SQL Server](#)
- [Connect Bitbucket to Oracle](#)

Ensure your database is configured to allow enough concurrent connections. Bitbucket Server by default uses up to 80 connections **per cluster node**, which can exceed the default connection limit of some databases. For example, in PostgreSQL the default limit is usually 100 connections. If you use PostgreSQL, you may need to edit your `postgresql.conf` file, to increase the value of `max_connections`, and restart Postgres.

See [Connect Bitbucket to an external database](#) for more information, and note that clustered databases are not supported.

We do **not** support MySQL for Bitbucket Data Center at this time due to inherent deadlocks that can occur in this database engine at high load. If you currently use MySQL, you should migrate your data to another supported database (such as PostgreSQL) before upgrading your Bitbucket Server instance to Bitbucket Data Center. You can migrate databases (on a standalone Bitbucket Server instance) using the Migrate database feature in Bitbucket Server's Administration pages.

Step 2. Provision your shared file system

A properly resourced and configured NFS server can perform well even under very heavy load. We've created some recommendations for setting up and configuring your file server for optimal performance.

Bitbucket Data Center requires a high performance shared file system, such as a storage area network (SAN), network-attached storage (NAS), RAID server, or high-performance file server optimized for input/output.

The file system must:

- run on a dedicated machine; avoid hosting other services on your NFS server

- be in the same physical data center
- be available to all Bitbucket nodes via a high-speed LAN (such as 10GB ethernet or Fibre Channel)
- be accessible via NFS as a single mount point
- have the NFS lock service enabled



For reference, our [AWS Quick Start for Bitbucket Data Center](#) deploys NFS and the Bitbucket cluster nodes on the same VPC, availability zone, and subnet.

You need to create a user account named `bitbucket` on the shared file system server. This user account should have read/write permissions to the shared subdirectory of the [Bitbucket Server home directory](#).

To ensure this:

- set `bitbucket` to own all files and folders in the shared subdirectory
- create `bitbucket` with the user `umask 0027`
- assign the same UID to `bitbucket` on all NFS Server and Bitbucket cluster nodes

Where possible, enable the [SCM Cache plugin](#) (already bundled in Bitbucket Data Center) with as much disk or SSD space on your cluster nodes as you can.

An effective SCM Cache can greatly reduce load on your shared file server. Note that the drive or partition used by the SCM Cache is local to each cluster node, not NFS mounted.

See [Scaling Bitbucket Server for Continuous Integration performance](#) for more information.

On your file server, ensure that NFS is configured with enough server processes. For example, some versions of Red Hat Enterprise Linux and CentOS have a default of 8 server processes. If you use either distribution, you may need to edit your `/etc/sysconfig/nfs` file, increase the value of `RPCNFSDCOUNT`, and restart the `nfs` service.

For the file server and cluster nodes, avoid kernel and NFS version combinations that are unstable or have known NFS bugs. We recommend avoiding Linux kernel versions 3.2 to 3.8.

When you provision your [application cluster nodes later](#), we recommend using the following NFS mount options:

```
rw,nfsvers=3,lookupcache=pos,noatime,intr,rsize=32768,wsiz=32768,_netdev
```

Step 4. Migrating from an existing Bitbucket Server instance (optional)

The `shared` subdirectory of the [Bitbucket Server home directory](#) contains all the GIT repositories, configuration data, and other important files. When migrating from an existing Bitbucket Server instance, back up `shared` and restore it on the new Bitbucket Data Centers NFS file system.

The remaining subdirectories (`bin`, `caches`, `export`, `lib`, `log`, `plugins`, and `tmp`) contain only caches and temporary files. You don't need to restore them.

Step 5. Provision your Elasticsearch node

To set up your Elasticsearch server, you will

1. Install Elasticsearch on a remote machine.
2. Configure Elasticsearch's `elasticsearch.yml` file to work with Bitbucket Data Center.
3. Secure Elasticsearch with a username and password that Bitbucket will use to access Elasticsearch, with a minimum of HTTP restricted access.
4. Connect Elasticsearch to Bitbucket.

There are detailed instructions on the page [How to Install and configure a remote Elasticsearch instance](#) to help you provision your remote Elasticsearch installation.

Provision application cluster nodes

Provision cluster node infrastructure. You can automate this using a configuration management tool such as Chef, Puppet, or Vagrant, and/or by spinning up identical virtual machine snapshots.

Step 1. Configure file share mounts

 NFSv3 is the only supported version.

On each cluster node, mount the shared home directory as `${BITBUCKET_HOME}/shared`. Note that **only** the `${BITBUCKET_HOME}/shared` directory should be shared between cluster nodes. All other directories, including `${BITBUCKET_HOME}`, should be node-local (that is, private to each node).

Example

For example, suppose your Bitbucket home directory is `/var/atlassian/application-data/bitbucket`, and your shared home directory is available as an NFS export called `bitbucket-san:/bitbucket-shared`. To configure the mount on each cluster node:

1. Add the following line to `/etc/fstab` on each cluster node.

/etc/fstab

```
bitbucket-san:/bitbucket-shared /var/atlassian/application-data/bitbucket/shared nfs
rw,nfsvers=3,lookupcache=pos,noatime,intr,rsize=32768,wsize=32768,_netdev 0 0
```

2. Mount the share on each node:

```
mkdir -p /var/atlassian/application-data/bitbucket/shared
sudo mount -a
```

Step 2. Synchronize system clocks

Ensure all your cluster nodes have synchronized clocks and identical timezone configuration. Here are some examples for how to do this:

```
sudo yum install ntp
sudo service ntpd start
sudo tzselect
```

```
sudo apt-get install ntp
sudo service ntp start
sudo dpkg-reconfigure tzdata
```

Step 3. Install Bitbucket Data Center on each node

On each cluster node, perform the same steps from [Install Bitbucket Data Center on the first application node](#) section.

Step 4. Start the first cluster node

Edit the file `${BITBUCKET_HOME}/shared/bitbucket.properties` and add the following lines:

```
# Use multicast to discover cluster nodes (recommended).
hazelcast.network.multicast=true

# If your network does not support multicast, you may uncomment the following lines and substitute
# the IP addresses of some or all of your cluster nodes. (Not all of the cluster nodes have to be
# listed here but at least one of them has to be active when a new node joins.)
#hazelcast.network.tcpip=true
#hazelcast.network.tcpip.members=192.168.0.1:5701,192.168.0.2:5701,192.168.0.3:5701

# The following should uniquely identify your cluster on the LAN.
hazelcast.group.name=your-bitbucket-cluster
hazelcast.group.password=your-bitbucket-cluster
```

If you are installing Bitbucket Server in an IPv6 environment, Hazelcast needs an additional system property to work.

This property is only needed if the nodes are configured to use IPv6 to talk to each other.

Edit the file `$(BITBUCKET_INSTALL)/bin/_start-webapp.sh` and edit the `JVM_SUPPORT_RECOMMENDED_ARG` line to look like this:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-Dhazelcast.prefer.ipv4.stack=false"
```

The line is commented out, so be sure to remove the leading `#` to make the line take effect.

Using multicast to discover cluster nodes (`hazelcast.network.multicast=true`) is recommended, but requires all your cluster nodes to be accessible to each other via a multicast-enabled network. If your network does not support multicast then you can set `hazelcast.network.multicast=false`, `hazelcast.network.tcpip=true`, and `hazelcast.network.tcpip.members` to a comma-separated list of cluster nodes instead. Only enable **one** of `hazelcast.network.tcpip` or `hazelcast.network.multicast`, not both.

Choose a name for `hazelcast.group.name` and `hazelcast.group.password` that uniquely identifies the cluster on your LAN. If you have more than one cluster on the same LAN (for example, other Bitbucket Data Center instances or other products based on similar technology such as Confluence Data Center) then you **must** assign each cluster a distinct name, to prevent them from attempting to join together into a "super cluster".

Then start Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

Then go to `http://<bitbucket>:7990/admin/license`, and install your Bitbucket Data Center license. Restart Bitbucket Server for the change to take effect. If you need a Bitbucket Data Center license, you can [purchase one that fits your needs](#), or [get an evaluation license](#).

Install and configure your load balancer

Step 1. Configure protocols and health checks on your load balancer

Your load balancer must proxy three protocols:

Protocol	Typical port on the load balancer	Typical port on the Bitbucket cluster nodes	Notes
HTTP	80	7990	HTTP mode. Session affinity ("sticky sessions") should be enabled using the 52-character <code>BITBUCKETSESSIONID</code> cookie.

HTTPS	443	7990	HTTP mode. Terminating SSL at the load balancer and running plain HTTP to the Bitbucket cluster nodes is highly recommended.
SSH	7999	7999	TCP mode.

❗ Your load balancer must support session affinity ("sticky sessions") using the `BITBUCKETSESSIONID` cookie. Bitbucket Data Center assumes that your load balancer always directs each user's requests to the same cluster node. If it does not, users may be unexpectedly logged out or lose other information that may be stored in their HTTP session.

❗ When choosing a load balancer, it must support the HTTP, HTTPS, and TCP protocols. Note that:

- Apache does **not** support TCP mode load balancing.
- HAProxy versions older than 1.5.0 do **not** support HTTPS.

If your load balancer supports health checks of the cluster nodes, configure it to perform a periodic HTTP GET of `http://<bitbucket>:7990/status`, where `<bitbucket>` is the cluster node's name or IP address. This returns one of two HTTP status codes:

- 200 OK
- 500 Internal Server Error

If a cluster node does not return 200 OK within a reasonable amount of time, the load balancer should not direct any traffic to it.

You should then be able to navigate to `http://<load-balancer>/`, where `<load-balancer>` is your load balancer's name or IP address. This should take you to your Bitbucket Server front page.

Example: HAProxy load balancer

If you don't have a particular preference or policy for load balancers, you can use HAProxy which is a popular Open Source software load balancer.

❗ If you choose HAProxy, you **must** use a minimum version of 1.5.0. Earlier versions of HAProxy do not support HTTPS.

To check which version of HAProxy you use, run the following command:

```
haproxy --version
```

Here is an example `haproxy.cfg` configuration file (typically found in the location `/etc/haproxy/haproxy.cfg`). This assumes:

- Your Bitbucket cluster node is at address 192.168.0.1, listening on the default ports 7990 (HTTP) and 7999 (SSH).
- You have a valid SSL certificate at `/etc/cert.pem`.

haproxy.cfg

```

global
    pidfile      /var/run/haproxy.pid
    maxconn      4000
    user         haproxy
    group        haproxy
    daemon
    tune.ssl.default-dh-param 1024
defaults
    log          global
    option       dontlognull
    option       redispatch
    retries      3
    timeout http-request    10s
    timeout queue          1m
    timeout connect        10s
    timeout client         1m
    timeout server         1m
    timeout http-keep-alive 10s
    timeout check          10s
    maxconn               3000
    errorfile              408 /dev/null          # Workaround for Chrome 35-36 bug.  See http://blog.
haproxy.com/2014/05/26/haproxy-and-http-errors-408-in-chrome/

frontend bitbucket_http_frontend
    bind *:80
    bind *:443 ssl crt /etc/cert.pem ciphers RC4-SHA:AES128-SHA:AES256-SHA
    default_backend bitbucket_http_backend

backend bitbucket_http_backend
    mode http
    option httplog
    option httpchk GET /status
    option forwardfor
    option http-server-close
    appsession BITBUCKETSESSIONID len 52 timeout 1h
    balance roundrobin
    cookie BITBUCKETSESSIONID prefix
    stick-table type string len 52 size 5M expire 30m
    stick store-response set-cookie(BITBUCKETSESSIONID)
    stick on cookie(BITBUCKETSESSIONID)
    server bitbucket01 192.168.0.1:7990 check inter 10000 rise 2 fall 5
    #server bitbucket02 192.168.0.2:7990 check inter 10000 rise 2 fall 5
    # The following "backup" servers are just here to show the startup page when all nodes are
starting up
    server backup01 192.168.0.1:7990 backup
    #server backup02 192.168.0.2:7990 backup

frontend bitbucket_ssh_frontend
    bind *:7999
    default_backend bitbucket_ssh_backend
    timeout client 15m
    maxconn 50

backend bitbucket_ssh_backend
    mode tcp
    balance roundrobin
    server bitbucket01 192.168.0.1:7999 check port 7999
    #server bitbucket02 192.168.0.2:7999 check port 7999
    timeout server 15m

listen admin
    mode http
    bind *:8090
    stats enable
    stats uri /

```



Review the contents of the `haproxy.cfg` file carefully, and customize it for your environment. See <http://www.haproxy.org/> for more information about installing and configuring haproxy.

Once you have configured the `haproxy.cfg` file, start the `haproxy` service.

```
sudo service haproxy start
```

You can also monitor the health of your cluster by navigating to HAProxy's statistics page at `http://<load-balancer>:8090/`. You should see a page similar to this:

HAProxy version 1.5.0, released 2014/06/19

Statistics Report for pid 28972

> General process information

pid = 28972 (process #1, nproc = 1)
 uptime = 0d 20h34m58s
 system limits: memmax = unlimited; ulimit-n = 8019
 maxsock = 8019; maxconn = 4000; maxpipes = 0
 current conns = 3; current pipes = 0/0; conn rate = 2/sec
 Running tasks: 1/17; idle = 100 %

Display option:

- Scope :
- [Hide 'DOWN' servers](#)
- [Refresh now](#)
- [CSV export](#)

External resources:

- [Primary site](#)
- [Updates \(v1.5\)](#)
- [Online manual](#)

stash_http_frontend

Queue	Session rate			Sessions			Bytes		Denied	Errors		Warnings	Server												
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot		Last	In		Out	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn
Frontend	0	12	-	1	22	3 000	60 503			45 043 864	381 514 405	0	0	127			OPEN								

stash_cluster_http_backend

Queue	Session rate			Sessions			Bytes		Denied	Errors		Warnings	Server												
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot		Last	In		Out	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn
stash-app-1	0	0	-	0	25	0 10	49 705	45 103	3s	24 674 486	194 031 639	0	0	0	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	9	0	0s	-
stash-app-2	0	0	-	0	38	0 10	48 129	45 103	3s	20 369 378	187 456 692	0	0	2	0	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	10	0	0s	-
Backend	0	0	0	36	1 22	300	137 605	90 206	3s	45 043 864	381 514 405	0	0	0	2	0	20h34m UP		2	2	0	0	0	0s	

stash_esh_frontend

Queue	Session rate			Sessions			Bytes		Denied	Errors		Warnings	Server												
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot		Last	In		Out	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn
Frontend	1	15	-	0	11	50	145 340			391 467 702	1 493 870 815	0	0	0			OPEN								

stash_cluster_esh_backend

Queue	Session rate			Sessions			Bytes		Denied	Errors		Warnings	Server												
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot		Last	In		Out	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn
stash-app-1	0	0	-	0	7	0 7	54 411	54 411	9s	195 799 631	761 383 500	0	0	5	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
stash-app-2	0	0	-	0	7	0 6	54 410	54 410	13s	195 668 071	732 487 315	0	0	6	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0	0	15	0 11	5	145 340	108 821	9s	391 467 702	1 493 870 815	0	0	0	11	0	20h34m UP		2	2	0	0	0	0s	

Step 2. Configure Bitbucket for HAProxy

Bitbucket needs to be configured to work with HAProxy. For example:

```
<Bitbucket home directory>/shared/bitbucket.properties

server.proxy-name=bitbucket.company.com
server.proxy-port=443
server.secure=true
server.require-ssl=true
```

Read [Securing Bitbucket behind HAProxy using SSL](#) for more details.

Step 3. Add a new Bitbucket application node to the cluster

Go to a new cluster node, and start Bitbucket Server. See [Start and stop Bitbucket](#).

Once Bitbucket Server has started, go to `https://<load-balancer>/admin/clustering`. You should see a page similar to this:

The screenshot shows the Bitbucket Administration page for Clustering. The left sidebar contains navigation links for Overview, ACCOUNTS (Users, Groups, Global permissions, Authentication, SAML Authentication, Avatars, User Directories), SETTINGS (Server settings, Database, Application Navigator, Application Links, Mail server, Licensing, Clustering, Mirrors, Analytics), and ADD-ONS (Find new add-ons). The main content area is titled 'Clustering' and includes a sub-header 'A cluster of multiple Bitbucket nodes provides high availability and performance at scale. Learn more about clustering.' Below this is a table with columns for Node ID, Node name, and Cluster Address. Two nodes are listed: 'test-node-1' with Node ID '8fcc6708-77f1-43dc-9b3b-4452c245335e' and Cluster Address '10.0.0.203:5701', and 'test-node-2' with Node ID 'd31f177d-df5b-4c07-9e4a-8228ee1e9b87' and Cluster Address '10.0.1.148:5701'. Below the table, there is a section for adding new nodes with a plus icon and the text 'New nodes can join the cluster without downtime. Learn how to add a node.'

Node ID	Node name	Cluster Address
8fcc6708-77f1-43dc-9b3b-4452c245335e	test-node-1	10.0.0.203:5701
d31f177d-df5b-4c07-9e4a-8228ee1e9b87	test-node-2	10.0.1.148:5701

Verify that the new node you have started up has successfully joined the cluster. If it does not, please check your network configuration and the `{BITBUCKET_HOME}/log/atlassian-bitbucket.logfile` on all nodes. If you are unable to find a reason for the node failing to join successfully, please contact [Atlassian Support](#).

Step 4. Connect the new Bitbucket cluster node to the load balancer

If you are using your own hardware or software load balancer, consult your vendor's documentation on how to add the new Bitbucket cluster node to the load balancer.

If you are using HAProxy, uncomment these lines

```
server bitbucket02 192.168.0.2:7990 check inter 10000 rise 2 fall 5
```

```
server bitbucket02 192.168.0.2:7999 check port 7999
```

in your `haproxy.cfg` file and restart `haproxy`:

```
sudo service haproxy restart
```

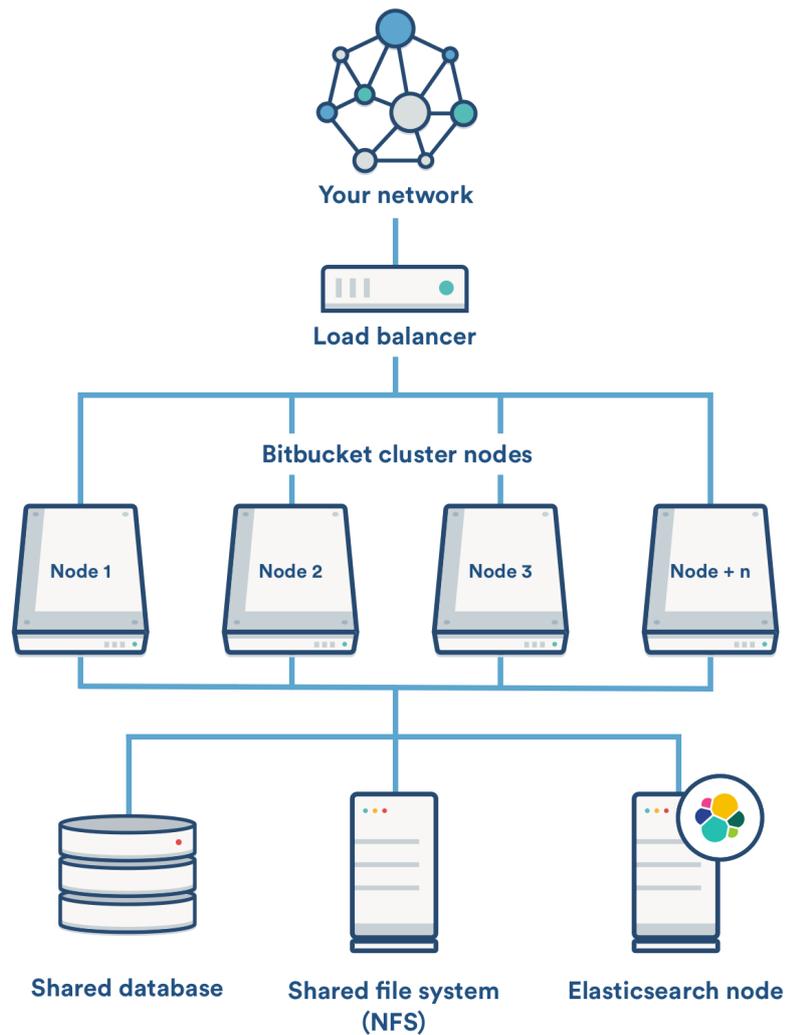
Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure both are receiving traffic and also check that updates done on one node are visible on the other.

Bitbucket Data Center requirements

This page describes the requirements for running a production instance of Bitbucket Data Center. If you're ready to get started installing Bitbucket Data Center, see [Install Bitbucket Data Center](#) for detailed instructions.

Component overview

A Bitbucket Data Center instance consists of a cluster of components, each on a dedicated machine, and connected over a high-speed LAN connection. This diagram depicts a typical Bitbucket Data Center instance with a [load balancer](#), three [application nodes](#), a [shared database](#), a [shared file system](#), and a single remote [Elasticsearch node](#).



Component requirements

Each component has specific requirements, but only the load balancer needs to have a publicly accessible URL. The URL of the Bitbucket Data Center instance will be the URL of the load balancer, so this is the machine that you will need to assign the name of your Bitbucket Server instance in the DNS.

The remaining machines (Bitbucket cluster nodes, shared database, shared file system, and Elasticsearch node) do not need to be publicly accessible to your users.

Bitbucket application nodes

The Bitbucket cluster nodes all run the Bitbucket Data Center web application.

- Each Bitbucket cluster node must be a dedicated machine.
- The machines may be physical or virtual.
- The cluster nodes must be connected in a high speed LAN (that is, high bandwidth and low latency).
- The usual Bitbucket Server [supported platforms](#) requirements, including those for Java and Git, apply to each cluster node.
- The cluster nodes do not all need to be absolutely identical, but for consistent performance we recommend they should be as similar as possible.
- All cluster nodes must run the same version of Bitbucket Data Center.
- All cluster nodes must have synchronized clocks (for example, using NTP) and be configured with the identical timezone.
- Ensure that only permitted cluster nodes are allowed to connect to a Bitbucket cluster node's [Hazelcast](#) port, which by default is port 5701, through the use of a firewall and/or network segregation.

Load balancer

You can use the load balancer of your choice. Bitbucket Data Center does **not** bundle a load balancer.

- Your load balancer should run on a dedicated machine.
- Your load balancer must have a high-speed LAN connection to the Bitbucket cluster nodes (that is, high bandwidth and low latency).
- Your load balancer must support **both** HTTP mode (for web traffic) **and** TCP mode (for SSH traffic).
- Terminating SSL (HTTPS) at your load balancer and running plain HTTP from the load balancer to Bitbucket Server is highly recommended for performance.
- Your load balancer should support "session affinity" (also known as "sticky sessions").
- If you don't have a preference for your load balancer, we provide instructions for [haproxy](#), a popular Open Source software load balancer.

Shared database

You must run Bitbucket Data Center on an external database. You can **not** use Bitbucket Server's internal HSQL or H2 database with Bitbucket Data Center.

- The shared database must run on a dedicated machine.
- The shared database must be available to all cluster nodes via a high-speed LAN (it must be in the same physical data center).
- All the usual database vendors in Bitbucket Server's [supported platforms](#) are supported by Bitbucket Data Center, with one exception: we **do not** recommend MySQL at this time due to inherent deadlocks that can occur in this database engine at high load.

Shared file system

Bitbucket Data Center requires a high performance shared file system such as a SAN, NAS, RAID server, or high-performance file server optimized for I/O.

- The shared file system must run on a dedicated machine.
- The file system must be available to all cluster nodes via a high-speed LAN (it must be in the same physical data center).
- The shared file system should be accessible via NFS as a single mount point.

Stored on the shared file system	Stored locally on each application node
<ul style="list-style-type: none"> • configuration files • data directory, which includes: <ul style="list-style-type: none"> • repositories • attachments • avatars • plugins 	<ul style="list-style-type: none"> • caches • logs • temporary files

For more information on setting up Bitbucket Data Center's shared file server, see [Step 2. Provision your shared file system](#) (in [Install Bitbucket Data Center](#)). This section contains the requirements and recommendations for setting up NFS for Bitbucket Data Center.

Elasticsearch node

Bitbucket Data Center requires a remote Elasticsearch instance, as it is not bundled or installed for Bitbucket Data Center.

- Bitbucket Data Center currently works with Elasticsearch 5.5.1 - 5.5.3, 6.5.3 and 6.6.1.
- Bitbucket Data Center can have *only one* remote connection to Elasticsearch for your cluster.
- This may be a standalone Elasticsearch installation or a clustered installation behind a load balancer.

Get started with Bitbucket Data Center

[Purchase a Bitbucket Data Center license](#), or, [obtain an evaluation license](#).

See [Install Bitbucket Data Center](#) for detailed information about setting up Bitbucket Data Center.

The [Bitbucket Data Center FAQ](#) answers the questions you might have when starting with Bitbucket Data Center.

Install Bitbucket Data Center

These instructions are applicable for installing Bitbucket Data Center on your own hardware.

This guide covers installing for the first time, with no existing data, or migrating your Bitbucket Server to a Data Center instance.



Other ways to install Bitbucket Data Center:

- [AWS](#)-hassle-free deployment in AWS using our Quick Start
- [Azure](#)- reference templates for Microsoft Azure deployment

Install Bitbucket Data Center on a single node

If your organization doesn't need high availability or disaster recovery capabilities right now, you can install Bitbucket Data Center without setting up a cluster.

To install Bitbucket Data Center, without setting up a cluster, follow the instructions for Bitbucket Server:

- [Bitbucket installation guide](#)

The process is almost identical to an ordinary Jira Server installation, just be sure to enter your Data Center license.

Install Bitbucket Data Center in a cluster

If your organization requires continuous uptime, scalability, and performance under heavy load, you'll want to run Bitbucket Data Center in a cluster.

See [Clustering with Bitbucket](#) for a complete overview of hardware and infrastructure considerations.

Before you begin

See our [Supported platforms](#) page for information on the database, Java, and operating systems you'll be able to use. These requirements are the same for Server and Data Center deployments.

You can see a component diagram of a typical Bitbucket Data Center instance, and read about detailed requirements of each component on the page [Bitbucket Data Center requirements](#) and on the [Supported platforms](#) page.

A Bitbucket Data Center instance consists of a cluster of components, each running on a dedicated machine:

- **A cluster of Bitbucket application nodes** all running the same version of Bitbucket Data Center web application. These can be virtual or physical machines, have synchronized clocks (for example, using NTP) and be configured with the identical timezone, and are allowed to connect to a Bitbucket cluster node's [Hazelcast](#) port, which by default is port 5701.
- **A load balancer** that supports *both* HTTP mode (for web traffic) *and* TCP mode (for SSH traffic), and support session affinity ("sticky sessions").
- **A supported external database**, shared and available to all cluster nodes.
- **A shared file system** that is physically located in the same data center, available to all cluster nodes, and accessible by NFS as a single mount point.
- **A remote Elasticsearch instance** with only one remote connection to Bitbucket. The instance may be a standalone Elasticsearch installation or a clustered installation behind a load balancer. For help installing and configuring a remote Elasticsearch instance see our [how to guide](#).

In this guide we'll use the following terminology:

- **Installation directory:** The directory where you installed Bitbucket.
- **Local home directory:** The home or data directory stored locally on each cluster node (if Bitbucket is not running in a cluster, this is simply known as the home directory).
- **Shared home directory:** The directory you created that is accessible to all nodes in the cluster via the same path.

Install Bitbucket Data Center on the first application node

First, you'll need to install Bitbucket Data Center one node:

1. Download the latest installer -www.atlassian.com/software/bitbucket/download.
2. Make the installer executable.

Change to the directory where you downloaded the installer, then execute this command:

```
chmod +x atlassian-bitbucket-x.x.x-x64.bin
```

Where `x.x.x` is the version you downloaded.

3. Run the installer we recommend using `sudo` to run the installer as this will create a dedicated account to run Bitbucket and allow you to run Bitbucket as a service.

To use `sudo` to run the installer execute this command:

```
$ sudo ./atlassian-bitbucket-x.x.x-x64.bin
```

You can also run the installer with root user privileges. Where `x.x.x` is the version you downloaded.

4. Follow the prompts to install Bitbucket. You'll be asked for the following info:
 - a. **Type of Bitbucket instance**- the type of installation, for these instructions select **Data Center**.
 - b. **Installation directory**- where Bitbucket will be installed.
 - c. **Home directory**- where Bitbucket application data will be stored.
 - d. **TCP ports**- the HTTP connector port and control port Bitbucket will run on.

Provision the shared database, filesystem, and Elasticsearch nodes

Once you've installed the first Bitbucket application node, you now need to provision the share database and shared filesystem to use with Bitbucket Data Center.

Step 1. Provision your shared database

Set up your shared database server.

- [Connect Bitbucket to PostgreSQL](#)
- [Connect Bitbucket to SQL Server](#)
- [Connect Bitbucket to Oracle](#)

Ensure your database is configured to allow enough concurrent connections. Bitbucket Server by default uses up to 80 connections **per cluster node**, which can exceed the default connection limit of some databases. For example, in PostgreSQL the default limit is usually 100 connections. If you use PostgreSQL, you may need to edit your `postgresql.conf` file, to increase the value of `max_connections`, and restart Postgres.

See [Connect Bitbucket to an external database](#) for more information, and note that clustered databases are not supported.

We do **not** support MySQL for Bitbucket Data Center at this time due to inherent deadlocks that can occur in this database engine at high load. If you currently use MySQL, you should migrate your data to another supported database (such as PostgreSQL) before upgrading your Bitbucket Server instance to Bitbucket Data Center. You can migrate databases (on a standalone Bitbucket Server instance) using the Migrate database feature in Bitbucket Server's Administration pages.

Step 2. Provision your shared file system

A properly resourced and configured NFS server can perform well even under very heavy load. We've created some recommendations for setting up and configuring your file server for optimal performance.

Bitbucket Data Center requires a high performance shared file system, such as a storage area network (SAN), network-attached storage (NAS), RAID server, or high-performance file server optimized for input/output.

The file system must:

- run on a dedicated machine; avoid hosting other services on your NFS server
- be in the same physical data center
- be available to all Bitbucket nodes via a high-speed LAN (such as 10GB ethernet or Fibre Channel)
- be accessible via NFS as a single mount point
- have the NFS lock service enabled



For reference, our [AWS Quick Start for Bitbucket Data Center](#) deploys NFS and the Bitbucket cluster nodes on the same VPC, availability zone, and subnet.

You need to create a user account named `bitbucket` on the shared file system server. This user account should have read/write permissions to the shared subdirectory of the [Bitbucket Server home directory](#).

To ensure this:

- set `bitbucket` to own all files and folders in the shared subdirectory
- create `bitbucket` with the user `umask 0027`
- assign the same UID to `bitbucket` on all NFS Server and Bitbucket cluster nodes

Where possible, enable the [SCM Cache plugin](#) (already bundled in Bitbucket Data Center) with as much disk or SSD space on your cluster nodes as you can.

An effective SCM Cache can greatly reduce load on your shared file server. Note that the drive or partition used by the SCM Cache is local to each cluster node, not NFS mounted.

See [Scaling Bitbucket Server for Continuous Integration performance](#) for more information.

On your file server, ensure that NFS is configured with enough server processes. For example, some versions of Red Hat Enterprise Linux and CentOS have a default of 8 server processes. If you use either distribution, you may need to edit your `/etc/sysconfig/nfs` file, increase the value of `RPCNFSDCOUNT`, and restart the `nfs` service.

For the file server and cluster nodes, avoid kernel and NFS version combinations that are unstable or have known NFS bugs. We recommend avoiding Linux kernel versions 3.2 to 3.8.

When you provision your [application cluster nodes later](#), we recommend using the following NFS mount options:

```
rw,nfsvers=3,lookupcache=pos,noatime,intr,rsize=32768,wsize=32768,_netdev
```

Step 4. Migrating from an existing Bitbucket Server instance (optional)

The `shared` subdirectory of the [Bitbucket Server home directory](#) contains all the GIT repositories, configuration data, and other important files. When migrating from an existing Bitbucket Server instance, back up `shared` and restore it on the new Bitbucket Data Center's NFS file system.

The remaining subdirectories (`bin`, `caches`, `export`, `lib`, `log`, `plugins`, and `tmp`) contain only caches and temporary files. You don't need to restore them.

Step 5. Provision your Elasticsearch node

To set up your Elasticsearch server, you will

1. Install Elasticsearch on a remote machine.
2. Configure Elasticsearch's `elasticsearch.yml` file to work with Bitbucket Data Center.
3. Secure Elasticsearch with a username and password that Bitbucket will use to access Elasticsearch, with a minimum of HTTP restricted access.
4. Connect Elasticsearch to Bitbucket.

There are detailed instructions on the page [How to Install and configure a remote Elasticsearch instance](#) to help you provision your remote Elasticsearch installation.

Provision application cluster nodes

Provision cluster node infrastructure. You can automate this using a configuration management tool such as Chef, Puppet, or Vagrant, and/or by spinning up identical virtual machine snapshots.

Step 1. Configure file share mounts

 NFSv3 is the only supported version.

On each cluster node, mount the shared home directory as `${BITBUCKET_HOME}/shared`. Note that **only** the `${BITBUCKET_HOME}/shared` directory should be shared between cluster nodes. All other directories, including `${BITBUCKET_HOME}`, should be node-local (that is, private to each node).

Example

For example, suppose your Bitbucket home directory is `/var/atlassian/application-data/bitbucket`, and your shared home directory is available as an NFS export called `bitbucket-san:/bitbucket-shared`. To configure the mount on each cluster node:

1. Add the following line to `/etc/fstab` on each cluster node.

/etc/fstab

```
bitbucket-san:/bitbucket-shared /var/atlassian/application-data/bitbucket/shared nfs
rw,nfsvers=3,lookupcache=pos,noatime,intr,rsize=32768,wsiz=32768,_netdev 0 0
```

2. Mount the share on each node:

```
mkdir -p /var/atlassian/application-data/bitbucket/shared
sudo mount -a
```

Step 2. Synchronize system clocks

Ensure all your cluster nodes have synchronized clocks and identical timezone configuration. Here are some examples for how to do this:

```
sudo yum install ntp
sudo service ntpd start
sudo tzselect
```

```
sudo apt-get install ntp
sudo service ntp start
sudo dpkg-reconfigure tzdata
```

Step 3. Install Bitbucket Data Center on each node

On each cluster node, perform the same steps from [Install Bitbucket Data Center on the first application node](#) section.

Step 4. Start the first cluster node

Edit the file `${BITBUCKET_HOME}/shared/bitbucket.properties` and add the following lines:

```
# Use multicast to discover cluster nodes (recommended).
hazelcast.network.multicast=true

# If your network does not support multicast, you may uncomment the following lines and substitute
# the IP addresses of some or all of your cluster nodes. (Not all of the cluster nodes have to be
# listed here but at least one of them has to be active when a new node joins.)
#hazelcast.network.tcpip=true
#hazelcast.network.tcpip.members=192.168.0.1:5701,192.168.0.2:5701,192.168.0.3:5701

# The following should uniquely identify your cluster on the LAN.
hazelcast.group.name=your-bitbucket-cluster
hazelcast.group.password=your-bitbucket-cluster
```

If you are installing Bitbucket Server in an IPv6 environment, Hazelcast needs an additional system property to work.

This property is only needed if the nodes are configured to use IPv6 to talk to each other.

Edit the file `${BITBUCKET_INSTALL}/bin/_start-webapp.sh` and edit the `JVM_SUPPORT_RECOMMENDED_ARG` line to look like this:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-Dhazelcast.prefer.ipv4.stack=false"
```

The line is commented out, so be sure to remove the leading `#` to make the line take effect.

Using multicast to discover cluster nodes (`hazelcast.network.multicast=true`) is recommended, but requires all your cluster nodes to be accessible to each other via a multicast-enabled network. If your network does not support multicast then you can set `hazelcast.network.multicast=false`, `hazelcast.network.tcpip=true`, and `hazelcast.network.tcpip.members` to a comma-separated list of cluster nodes instead. Only enable **one** of `hazelcast.network.tcpip` or `hazelcast.network.multicast`, not both.

Choose a name for `hazelcast.group.name` and `hazelcast.group.password` that uniquely identifies the cluster on your LAN. If you have more than one cluster on the same LAN (for example, other Bitbucket Data Center instances or other products based on similar technology such as Confluence Data Center) then you **must** assign each cluster a distinct name, to prevent them from attempting to join together into a "super cluster".

Then start Bitbucket Server. See [Starting and stopping Bitbucket Server](#).

Then go to `http://<bitbucket>:7990/admin/license`, and install your Bitbucket Data Center license. Restart Bitbucket Server for the change to take effect. If you need a Bitbucket Data Center license, you can [purchase one that fits your needs](#), or [get an evaluation license](#).

Install and configure your load balancer

Step 1. Configure protocols and health checks on your load balancer

Your load balancer must proxy three protocols:

Protocol	Typical port on the load balancer	Typical port on the Bitbucket cluster nodes	Notes
HTTP	80	7990	HTTP mode. Session affinity ("sticky sessions") should be enabled using the 52-character <code>BITBUCKETSESSIONID</code> cookie.
HTTPS	443	7990	HTTP mode. Terminating SSL at the load balancer and running plain HTTP to the Bitbucket cluster nodes is highly recommended.
SSH	7999	7999	TCP mode.

⚠ Your load balancer must support session affinity ("sticky sessions") using the `BITBUCKETSESSIONID` cookie. Bitbucket Data Center assumes that your load balancer always directs each user's requests to the same cluster node. If it does not, users may be unexpectedly logged out or lose other information that may be stored in their HTTP session.

⚠ When choosing a load balancer, it must support the HTTP, HTTPS, and TCP protocols. Note that:

- Apache does **not** support TCP mode load balancing.
- HAProxy versions older than 1.5.0 do **not** support HTTPS.

If your load balancer supports health checks of the cluster nodes, configure it to perform a periodic HTTP GET of `http://<bitbucket>:7990/status`, where `<bitbucket>` is the cluster node's name or IP address. This returns one of two HTTP status codes:

- 200 OK
- 500 Internal Server Error

If a cluster node does not return 200 OK within a reasonable amount of time, the load balancer should not direct any traffic to it.

You should then be able to navigate to `http://<load-balancer>/`, where `<load-balancer>` is your load balancer's name or IP address. This should take you to your Bitbucket Server front page.

Example: HAProxy load balancer

If you don't have a particular preference or policy for load balancers, you can use HAProxy which is a popular Open Source software load balancer.

⚠ If you choose HAProxy, you **must** use a minimum version of 1.5.0. Earlier versions of HAProxy do not support HTTPS.

To check which version of HAProxy you use, run the following command:

```
haproxy --version
```

Here is an example `haproxy.cfg` configuration file (typically found in the location `/etc/haproxy/haproxy.cfg`). This assumes:

- Your Bitbucket cluster node is at address 192.168.0.1, listening on the default ports 7990 (HTTP) and 7999 (SSH).

- You have a valid SSL certificate at `/etc/cert.pem`.

haproxy.cfg

```
global
    pidfile      /var/run/haproxy.pid
    maxconn      4000
    user         haproxy
    group        haproxy
    daemon
    tune.ssl.default-dh-param 1024
defaults
    log          global
    option       dontlognull
    option       redispatch
    retries      3
    timeout http-request    10s
    timeout queue          1m
    timeout connect        10s
    timeout client         1m
    timeout server         1m
    timeout http-keep-alive 10s
    timeout check          10s
    maxconn               3000
    errorfile              408 /dev/null          # Workaround for Chrome 35-36 bug.  See http://blog.haproxy.com/2014/05/26/haproxy-and-http-errors-408-in-chrome/
frontend bitbucket_http_frontend
    bind *:80
    bind *:443 ssl crt /etc/cert.pem ciphers RC4-SHA:AES128-SHA:AES256-SHA
    default_backend bitbucket_http_backend
backend bitbucket_http_backend
    mode http
    option httplog
    option httpchk GET /status
    option forwardfor
    option http-server-close
    appsession BITBUCKETSESSIONID len 52 timeout 1h
    balance roundrobin
    cookie BITBUCKETSESSIONID prefix
    stick-table type string len 52 size 5M expire 30m
    stick store-response set-cookie(BITBUCKETSESSIONID)
    stick on cookie(BITBUCKETSESSIONID)
    server bitbucket01 192.168.0.1:7990 check inter 10000 rise 2 fall 5
    #server bitbucket02 192.168.0.2:7990 check inter 10000 rise 2 fall 5
    # The following "backup" servers are just here to show the startup page when all nodes are
starting up
    server backup01 192.168.0.1:7990 backup
    #server backup02 192.168.0.2:7990 backup
frontend bitbucket_ssh_frontend
    bind *:7999
    default_backend bitbucket_ssh_backend
    timeout client 15m
    maxconn 50
backend bitbucket_ssh_backend
    mode tcp
    balance roundrobin
    server bitbucket01 192.168.0.1:7999 check port 7999
    #server bitbucket02 192.168.0.2:7999 check port 7999
    timeout server 15m
listen admin
    mode http
    bind *:8090
    stats enable
    stats uri /
```



Review the contents of the `haproxy.cfg` file carefully, and customize it for your environment. See <http://www.haproxy.org/> for more information about installing and configuring haproxy.

Once you have configured the `haproxy.cfg` file, start the `haproxy` service.

```
sudo service haproxy start
```

You can also monitor the health of your cluster by navigating to HAProxy's statistics page at `http://<load-balancer>:8090/`. You should see a page similar to this:

HAProxy version 1.5.0, released 2014/06/19

Statistics Report for pid 28972

> General process information

pid = 28972 (process #1, nbproc = 1)
 uptime = 0d 20h34m58s
 system limits: memmax = unlimited; ulimit-n = 8019
 maxsock = 8019; maxconn = 4000; maxpipes = 0
 current conns = 3; current pipes = 0/0; conn rate = 2/sec
 Running tasks: 1/17; idle = 100 %

active UP
 active UP, going down
 active DOWN, going up
 active or backup DOWN
 active or backup DOWN for maintenance (MAINT)
 active or backup SOFT STOPPED for maintenance

backup UP
 backup UP, going down
 backup DOWN, going up
 not checked

Display option:
 Scope :
 Hide 'DOWN' servers
 Refresh now
 CSV export

External resources:
 Primary site
 Updates (v1.5)
 Online manual

Note: *NOLB*/"DRAIN" = UP with load-balancing disabled.

stash_http_frontend																												
Queue	Session rate			Sessions			Bytes			Denied		Errors		Warnings		Server												
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend	0	12	-	1	22	3 000	60 503			45 043	864	381 514	405	0	0	127			OPEN									

stash_cluster_http_backend																											
Queue	Session rate			Sessions			Bytes			Denied		Errors		Warnings		Server											
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
stash-app-1	0	0	-	0	25	0 10	- 49 705	45 103	3s	24 674	486	194 031	639	0	0	0	0	0	20h34m UP	L7OK in 3ms	1	Y	-	9	0	0s	-
stash-app-2	0	0	-	0	38	0 10	- 48 129	45 103	3s	20 369	378	187 456	692	0	0	0	2	0	20h34m UP	L7OK/200 in 3ms	1	Y	-	10	0	0s	-
Backend	0	0	-	0	36	1 22	300 137 605	90 206	3s	45 043	864	381 514	405	0	0	0	2	0	20h34m UP		2	2	0		0	0s	

stash_esh_frontend																											
Queue	Session rate			Sessions			Bytes			Denied		Errors		Warnings		Server											
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend	1	15	-	0	11	50	145 340			391 467	702	1 493 870	815	0	0	0			OPEN								

stash_cluster_esh_backend																											
Queue	Session rate			Sessions			Bytes			Denied		Errors		Warnings		Server											
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
stash-app-1	0	0	-	0	7	0 7	- 54 411	54 411	9s	195 799	631	761 383	500	0	0	5	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
stash-app-2	0	0	-	0	7	0 6	- 54 410	54 410	13s	195 668	071	732 487	315	0	0	6	0	0	20h34m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0	-	1	15	0 11	5 145 340	108 821	9s	391 467	702	1 493 870	815	0	0	0	11	0	20h34m UP		2	2	0		0	0s	

Step 2. Configure Bitbucket for HAProxy

Bitbucket needs to be configured to work with HAProxy. For example:

```
<Bitbucket home directory>/shared/bitbucket.properties

server.proxy-name=bitbucket.company.com
server.proxy-port=443
server.secure=true
server.require-ssl=true
```

Read [Securing Bitbucket behind HAProxy using SSL](#) for more details.

Step 3. Add a new Bitbucket application node to the cluster

Go to a new cluster node, and start Bitbucket Server. See [Start and stop Bitbucket](#).

Once Bitbucket Server has started, go to `https://<load-balancer>/admin/clustering`. You should see a page similar to this:

Administration

Clustering

A cluster of multiple Bitbucket nodes provides high availability and performance at scale. [Learn more about clustering.](#)

Node ID	Node name	Cluster Address
8fcc6708-77f1-43dc-9b3b-4452c245335e	test-node-1	10.0.0.203:5701
d31f177d-df5b-4c07-9e4a-8228ee1e9b87	test-node-2	10.0.1.148:5701

New nodes can join the cluster without downtime. [Learn how to add a node.](#)

Verify that the new node you have started up has successfully joined the cluster. If it does not, please check your network configuration and the `/${BITBUCKET_HOME}/log/atlassian-bitbucket.logfile` on all nodes. If you are unable to find a reason for the node failing to join successfully, please contact [Atlassian Support](#).

Step 4. Connect the new Bitbucket cluster node to the load balancer

If you are using your own hardware or software load balancer, consult your vendor's documentation on how to add the new Bitbucket cluster node to the load balancer.

If you are using HAProxy, uncomment these lines

```
server bitbucket02 192.168.0.2:7990 check inter 10000 rise 2 fall 5
```

```
server bitbucket02 192.168.0.2:7999 check port 7999
```

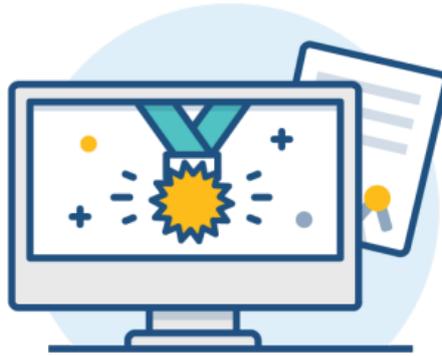
in your `haproxy.cfg` file and restart `haproxy`:

```
sudo service haproxy restart
```

Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure both are receiving traffic and also check that updates done on one node are visible on the other.

Congratulations!

That's it! Bitbucket Data Center is accessible from a URL like this: `http://<load_balancer_IP_address>:<port>`



What's next?

When setting up Bitbucket in a production environment, we recommend that you configure these aspects next:

- [Connect Bitbucket Server to a user directory](#)- manage users/groups stored in an external directory.
- [Run Bitbucket Server as a dedicated user](#)- run Bitbucket Server from a user account with restricted privileges.
- [Secure the Bitbucket home directory](#)-secure the home directory against unauthorized access.
- [Proxy and secure Bitbucket Server](#)-run Bitbucket Server behind a reverse proxy and enable HTTPS access.
- [Establish a data recovery plan](#)- backup the home directory and database of your instance.

Read more about setting up Bitbucket Server for an enterprise here:[Use Bitbucket in the enterprise](#).

Migrate from Server to Data Center

This page outlines the process for migrating an existing Bitbucket Server instance to Bitbucket Data Center.

If you're installing Bitbucket for the first time (you don't have any existing Jira data to migrate), see [Install Bitbucket Data Center](#).

Before you begin

Things you should know about when setting up your Data Center:

It's your Bitbucket license that determines the type of Bitbucket you have: Server or Data Center. Bitbucket will auto-detect the license type when you enter your license key, and automatically unlock any license-specific features.

You can [purchase a Data Center license](#) or create an evaluation license at my.atlassian.com

See our [Supported platforms](#) page for information on the database, Java, and operating systems you'll be able to use. These requirements are the same for Server and Data Center deployments.

Apps extend what your team can do with Atlassian applications, so it's important to make sure that your team can still use their apps after migrating to Data Center. When you switch to Data Center, you'll be required to switch to the Data Center compatible version of your apps, if one is available.

See [Evaluate apps for Data Center migration](#) for more information.

To use Bitbucket Data Center, you must:

- Have a Data Center license (you can [purchase a Data Center license](#) or create an evaluation license at my.atlassian.com)
- Use a [supported](#) external database, operating system and Java version
- Use OAuth authentication if you have [application links](#) to other Atlassian products (such as Confluence)

To run Bitbucket in a cluster, you'll need to meet additional requirements, like setting up a shared home directory or load balancer. You can learn more about it after going to [Set up your cluster](#).

Migrate to non-clustered Data Center

Review and upgrade your apps

If you have any apps installed on your site, you'll need to upgrade to the Data Center app version, if one is available. To avoid any impact to your apps, we recommend you do this before you enter your Bitbucket Data Center license key. Learn more about [Upgrading server apps when you migrate to Data Center](#).

Upgrade your Bitbucket license

1. Go to **Administration > Licensing**.
2. Enter your Bitbucket Data Center license key.

There's no need to restart Bitbucket. Data Center features such as rate limiting and SAML single sign-on will now be available.

Set up your cluster

If your organization requires continuous uptime, scalability, and performance under heavy load, you'll want to run Bitbucket Data Center in a cluster.

To find out more about clustering, including infrastructure requirements, see [Clustering with Bitbucket](#).

If you're ready to set up your cluster now, head to [Set up a Bitbucket Data Center cluster](#).

 Looking to migrate all your Atlassian applications to Data Center? Weve got you covered:

- [Migrate to Bitbucket Data Center](#)
- [Migrate to Crowd Data Center](#)
- [Migrate to Confluence Data Center](#)
- [Migrate to Jira Data Center](#)

Considering moving to cloud? [Plan your cloud migration.](#)

Deploy Bitbucket Data Center in Azure

- [Non-clustered VS clustered environment](#)
- [How it works](#)
 - [Standardized infrastructure](#)
 - [Limitations](#)
- [Deploying Bitbucket Data Center to Azure via Azure marketplace](#)
 - [Bitbucket-specific parameters](#)
 - [Standardized infrastructure parameters](#)
- [Deploying Bitbucket Data Center to Azure using the CLI](#)
 - [Monitoring](#)

If you decide to deploy Bitbucket Data Center in a clustered environment, consider using Microsoft Azure. This platform allows you to scale your deployment elastically by resizing and quickly launching additional nodes, and provides a number of managed services that work out of the box with Bitbucket Data Center. These services make it easier to configure, manage, and maintain your deployment's clustered infrastructure.

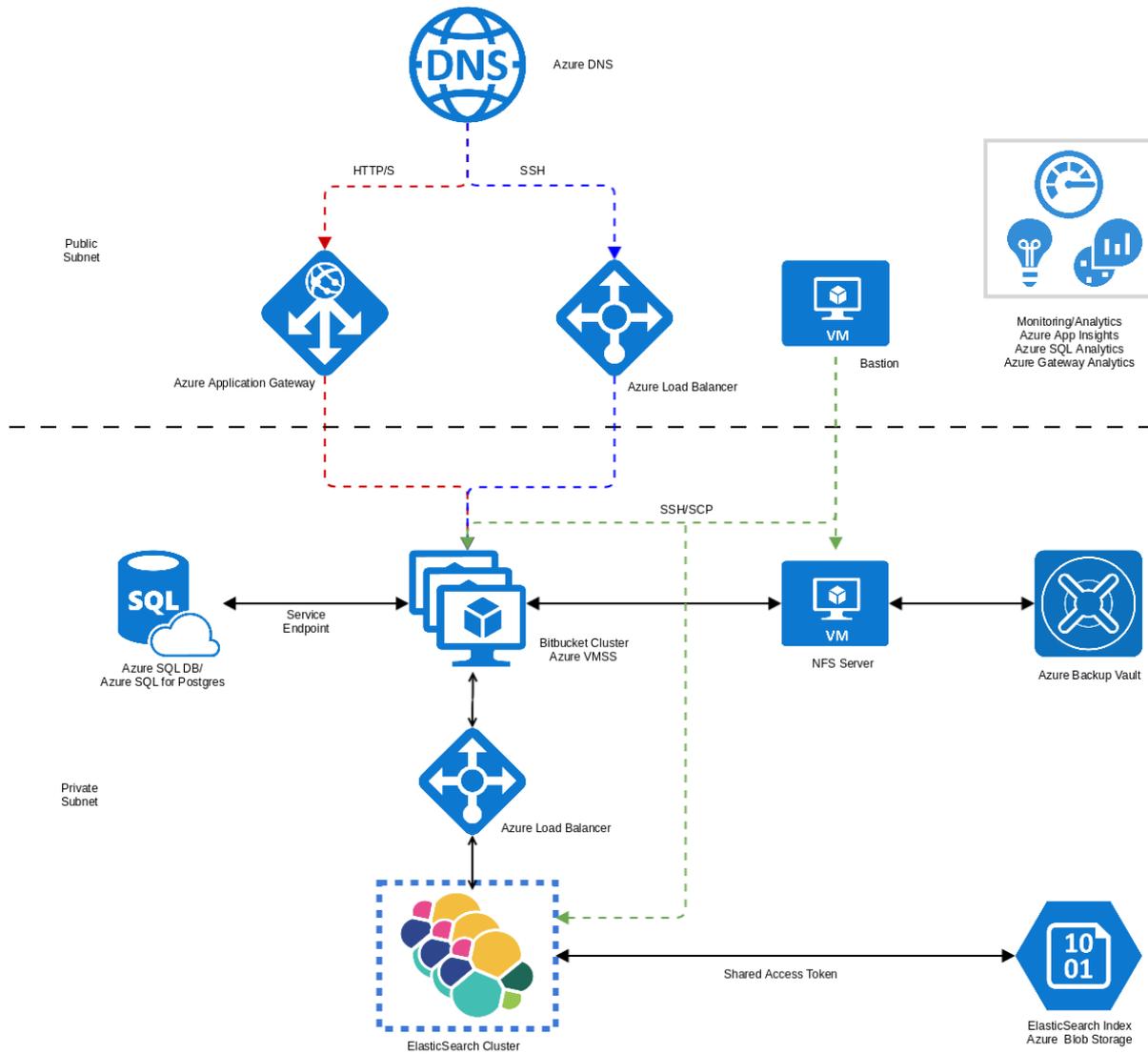
We strongly recommend you set up user management, central logging storage, a backup strategy, and monitoring, just as you would for a Bitbucket Data Center installation running on your own hardware.

Non-clustered VS clustered environment

A single node is adequate for most Small or Medium size deployments, unless you need specific features that require clustering (for example, high availability). If you have an existing Server installation, you can still use its infrastructure when you upgrade to Data Center. Many features exclusive to Data Center (like [SAML single sign-on](#), [self-protection via rate limiting](#), and [CDN support](#)) don't require clustered infrastructure. You can start using these Data Center features by simply upgrading your Server installations license. For more information on whether clustering is right for you, check out [Atlassian Data Center architecture and infrastructure options](#).

How it works

Here's an architectural overview of what you'll get when deploying Bitbucket Data Center using the template:



This deployment contains two different VM scale sets. One scale set runs Bitbucket, while the other runs ElasticSearch. Both consist of one or more Azure standard Linux VM instances as cluster nodes.

The template also provisions an NFS server that will serve as the Bitbucket home directory. It'll be deployed on a Linux VM.

Standardized infrastructure

The [Jira Data Center](#), [Confluence Data Center](#), [Bitbucket Data Center](#), and [Crowd Data Center](#) templates deploy the following infrastructure components identically:

Component	Configuration
Bastion host	This is a lightweight but highly secure Azure Linux VM that controls SSH access to the application cluster nodes.
Application Gateway	By default, this gateway is composed of two instances for high availability. It acts as a HTTP/HTTPS load balancer for your scale set of application cluster nodes.
Monitoring	The ARM templates configure Azure Monitoring to perform basic health and availability monitoring to cluster nodes and database.

Database	You can choose between Azure SQL Database (MS SQL Server-compatible) or Azure PostgreSQL database. Either way, the database will be configured as service endpoints to only allow traffic from the private network that the cluster nodes are in. This restricted traffic setup helps enhance security.
----------	---

Limitations

You can't use the deployment template to upgrade an existing Bitbucket deployment, or to provision new nodes running a different version to the rest of your cluster.

Deploying Bitbucket Data Center to Azure via Azure marketplace

This method uses the Azure Marketplace to deploy Bitbucket Data Center using our deployment templates as a reference.

 You will need **Owner** permissions on the resource group into which the deployment is planned. This means adding the RBAC role **Owner** to your Azure subscription.

For related information, see [What is role-based access control \(RBAC\) for Azure resources?](#) (from the Azure documentation).

To deploy Bitbucket Data Center to Azure using our Marketplace app:

1. Log in to [Azure Portal](#).
2. Choose **Create a resource** to start a new deployment from the **New** menu.
3. In **New**, search for **Atlassian** then select **Bitbucket Data Center** from the list of Marketplace apps.
4. Choose **Create** to start configuring the deployment.
5. Follow the prompts in the wizard to configure your deployment. Refer to the parameters table below for more information.
6. Confirm all the details are correct then click **Create** to purchase the subscription. Deployment will take about 30 minutes.
7. Once deployment is complete, go to the Bitbucket URL listed in the deployment outputs to start using Bitbucket Data Center.

Bitbucket-specific parameters

Parameters	Description
Bitbucket Version	Specify the version of Bitbucket you'd like to install in full (for example, 6.2.0). Head to Release notes for a list of all releases.
Bitbucket admin credentials	Provide a name, email, and password for the initial Bitbucket administrator in your instance.
Bitbucket cluster	Specify the initial number of Bitbucket application nodes, and the size of each node. This can be reconfigured at a later date.
File server	Specify the size of the NFS file server and its disk size.
Elasticsearch details	Specify the initial number of Elasticsearch nodes, along with the instance size and disk size of each one.

Standardized infrastructure parameters

The [Jira Data Center](#), [Confluence Data Center](#), [Bitbucket Data Center](#), and [Crowd Data Center](#) templates all share the same parameters:

Parameter	Description
Subscription	Your Microsoft Azure subscription type.

Resource group	If you have an existing resource group, you can use it, or create a new one.
Location	This is the region where Azure will house your deployment.
SSH Access	Provide an SSH public key to be used to SSH into the instance that will act as bastion host, and a username and password for SSH access to the Bitbucket nodes. See Create and use an SSH public-private key pair for Linux VMs in Azure in the Microsoft Azure documentation.
Database configuration	Choose between an Azure SQL Database, or Azure Database for PostgreSQL. Provide a username and password for the database admin user. <div style="border: 1px solid #ffc107; padding: 10px; margin: 10px 0;"> <p> Existing databases</p> <p>If you want to integrate with an existing database, you'll have to deploy to Azure using the CLI.</p> </div>
CNAME	This is the Canonical Name record (CNAME) for your organization. If you don't provide one, Azure will generate a random sub domain for your instance.
HTTP/SSL	Provide the certificate and password to be used for SSL termination on the Azure Application Gateway.
Monitoring	Choose the monitoring and analytics services that you would like to enable. Subject to availability in your location. See Monitoring for related information.

Deploying Bitbucket Data Center to Azure using the CLI

This method uses the Azure command line interface to deploy Bitbucket Data Center using our deployment templates as a reference. You'll need to [install the Azure CLI](#) to do this.

Using the deployment templates directly allows for greater configuration granularity. All hardware choices such as the number of cluster nodes, size, disk size, and OS type are configurable as parameters.

Head to <https://bitbucket.org/atlassian/atlassian-azure-deployment/src/master/> and check out the README to find out how to to deploy using the CLI.

Monitoring

As a number of the resources we provision are managed by Azure, a number of options are available for monitoring. For example:

- The application gateway will automatically monitor its backend pool (the Bitbucket application nodes), sending the alerts to the Bitbucket admin email address specified in the deployment. See [Application Gateway health monitoring overview](#) in the Microsoft Azure documentation.
- You can use Azure Monitor to check the overall health of your deployment, and dig into particular areas of interest. See [Azure Monitor overview](#) in the Azure documentation.
- Azure SQL Analytics is available for more granular monitoring of your SQL Server database. [Monitor Azure SQL Database using Azure SQL Analytics](#) in the Microsoft Azure documentation.

Note that some of these resources are still in Preview, so may not be available in your region yet.

Administer Bitbucket Data Center in Azure

Once you've deployed Bitbucket Data Center through the [Azure Marketplace template](#), administering the application is similar to managing an application on your own hardware. The exception being that you'll need to go via the jumpbox to access your nodes, and shared home directory.

To access your jumpbox and nodes you'll need:

- The SSH credentials you provided during [setup](#), and
- The URL of your jumpbox. This is listed as the `BASTIONURL` in your deployment's Outputs tab.

Connecting to your Azure jumpbox and nodes over SSH

You can access the jumpbox via a terminal, or using the command line using:

```
$ eval $(ssh-agent)
$ ssh-add ~/.ssh/id_rsa
$ ssh -A JUMPBOX_USERNAME@BASTIONURL
```



The `SSHURL` field in your deployment's Outputs tab shows the SSH command you can use to access your deployment.

Once you've accessed the jumpbox, you can jump to any of the nodes in the cluster, using:

```
$ ssh NODE_IP_ADDRESS
```

Accessing your configuration files

For your Azure deployment, you may need to make changes to your `bitbucket.properties` file, just as you would for a deployment on your own hardware.

Your shared `bitbucket.properties` lives in `/var/atlassian/application-data/bitbucket/shared/bitbucket.properties`

The *shared home* is mounted on each node under `/var/atlassian/application-data/bitbucket/shared`.

So from an existing node (when you're logged in through SSH), you can go to `/var/atlassian/application-data/bitbucket/shared`

Upgrading

Before upgrading to a later version of Bitbucket Data Center:

1. [Check if your apps are compatible](#) with that version. [Update your apps](#) if needed. For more information about managing apps, see [Using the Universal Plugin Manager](#).
2. [Enable integrity checks](#) (if you haven't already).

We strongly recommend that you perform the upgrade first in a *staging* environment before upgrading your production instance. [How to establish staging server environments for Bitbucket Server](#) provides helpful tips on doing so.

To upgrade to a later version of Bitbucket in Azure, you must first [connect to your instance using SSH](#), then follow the steps in the [Bitbucket Server upgrade guide](#).

Backing up

We recommend you use the Azure native backup facility, which utilizes snapshots to back up your Bitbucket Data Center. The following backup advice assumes that you deployed your Bitbucket instance through the [Azure Marketplace template](#).

Database

We use Azure-managed database instances with high availability. Azure provides several excellent options for backing up your database, so you should take some time to work out which will be the best, and most cost effective option for your needs. See the following Azure documentation for your chosen database:

- [SQL Database: Automated backups](#)
- [SQL Database: Backup retention](#)
- [PostgreSQL: Backup concepts](#)

NFS Server

This node handles the shared NFS file system of the Bitbucket application nodes. The [Azure Marketplace template](#) creates a general-purpose Azure storage account, configured with [local redundant storage \(LRS\)](#). Using LRS means there are multiple copies of the data at any one time, providing a basic redundancy strategy for your content.

The template also configures your deployment to back up the NFS server disks into the [Azure Recovery Service vault](#) daily. If you need to take point-in-time backups, use [snapshots](#).

Elasticsearch

The Elasticsearch cluster takes snapshots of its index every hour. Those snapshots are backed up to [Azure Blob Storage](#).

Application nodes

The application nodes are VMs in an [Azure Virtual Machine Scale Set](#). Each application node has a Bitbucket installation directory and a local home directory containing things like logs and search indexes.

Like the NFS Server, application nodes are configured with [local redundant storage](#). This means there are multiple copies of the data at any one time.

If you've manually customised any configuration files in the installation directory (for example velocity templates), you may also want to manually back these up as a reference.

Bastion host

As this VM acts as a jumpbox, and doesn't store any data it doesn't need to be backed up. If the VM becomes unresponsive it can be restarted from the Azure Portal.

Application gateway

The application gateway is highly available. As with the bastion host, it doesn't need to be backed up.

Migrating your existing instance into Azure

If you want to migrate data from an existing instance into Bitbucket Data Center in Azure, we recommend that you use the [Export and import projects and repositories](#) tool. This tool is only available to users with Bitbucket Data Center licenses.

Disaster recovery

See [Disaster recovery guide for Bitbucket Data Center](#) for guidance on developing a disaster recovery strategy. See also information in the Azure documentation about recovering from a region-wide failure [Azure resiliency technical guidance: recovery from a region-wide service disruption](#).

Disaster recovery guide for Bitbucket Data Center

This feature is only for customers with an active [Bitbucket Data Center resources](#) license.

This page demonstrates how Bitbucket Data Center can be configured to implement and manage a disaster recovery strategy. It doesn't, however, cover the broader business practices, like setting the key objectives (RTO, RPO & RCO¹), and standard operating procedures.

- [Overview](#)
- [Setting up a standby system](#)
- [Disaster recovery testing](#)
- [Handling a failover](#)
- [Returning to the primary instance](#)
- [Other resources](#)

A disaster recovery strategy is a key part of a business continuity plan. It defines the processes to follow in the event of a disaster to ensure your business can recover and keep operating from a standby system. This means your Bitbucket instance, and more importantly your Bitbucket managed data (source code), is available in the event that your primary system becomes unavailable.

Overview

Bitbucket Data Center 4.8 or later is required to implement the strategy described in this guide.

Before you begin, read these initial subsections to get familiar with the various components and terminology used in this guide, review the requirements that must be met to follow this guide, and introduce the example scripts used to perform actions described in this guide.

Terminology

Cold standby- This guide describes what is generally referred to as a "cold standby" strategy. That means that the standby Bitbucket instance is not continuously running and that some administrative steps need to be taken to start the standby instance in the event of a disaster scenario.

Recovery Point Objective (RPO)-How up-to-date you require your Bitbucket instance to be after a failure.

Recovery Time Objective (RTO)-How quickly you require your standby system to be available after a failure.

Recovery Cost Objective (RCO)-How much you are willing to spend on your disaster recovery solution.

What is the difference between high availability and disaster recovery?

The terms "high availability", "disaster recovery" and "failover" are often confused. To eliminate confusion we define these terms as:

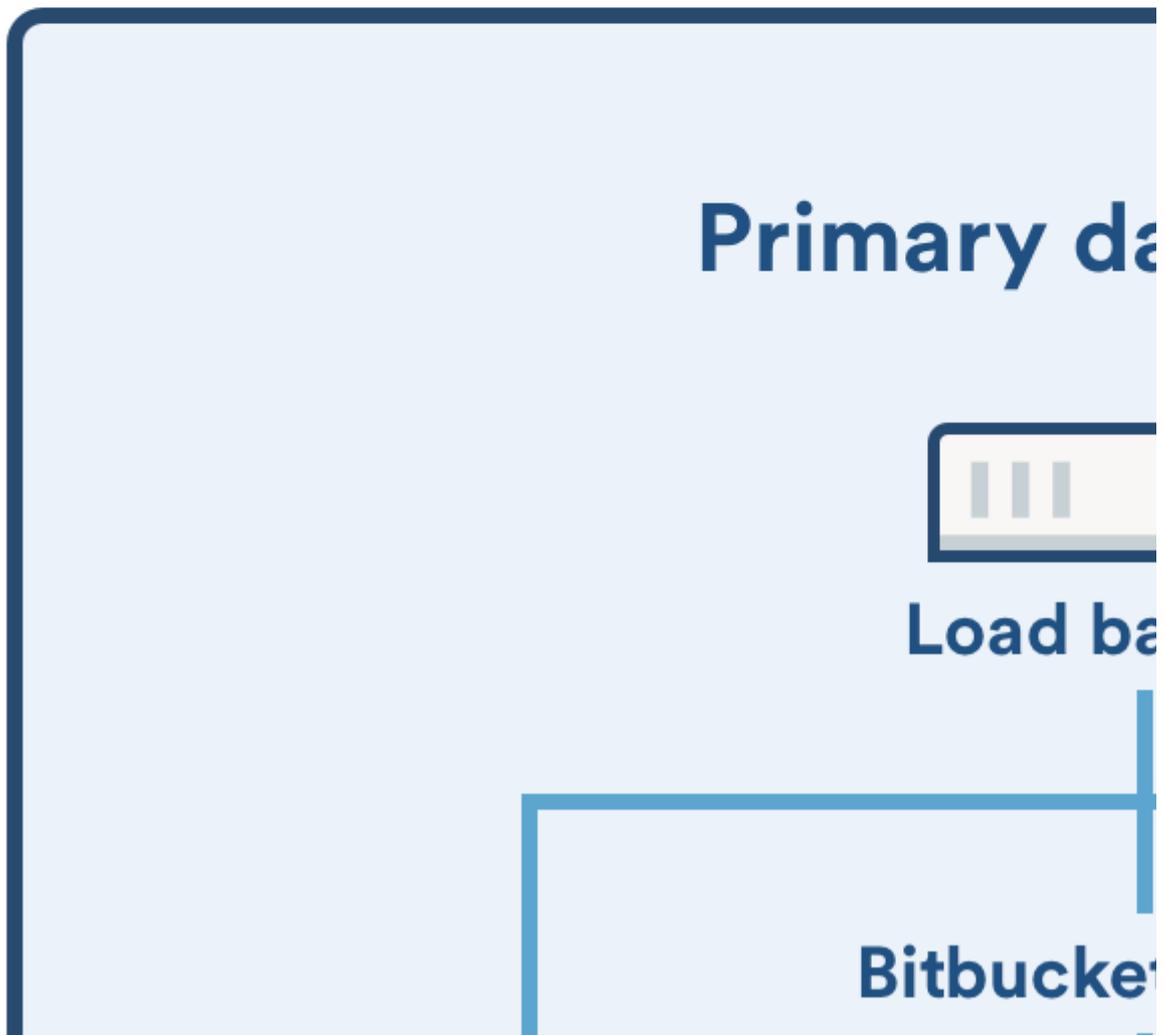
High availability A strategy to provide a specific level of availability. In Bitbucket's case, access to the application and an acceptable response time. Automated correction and failover (within the same location) are usually part of high-availability planning.

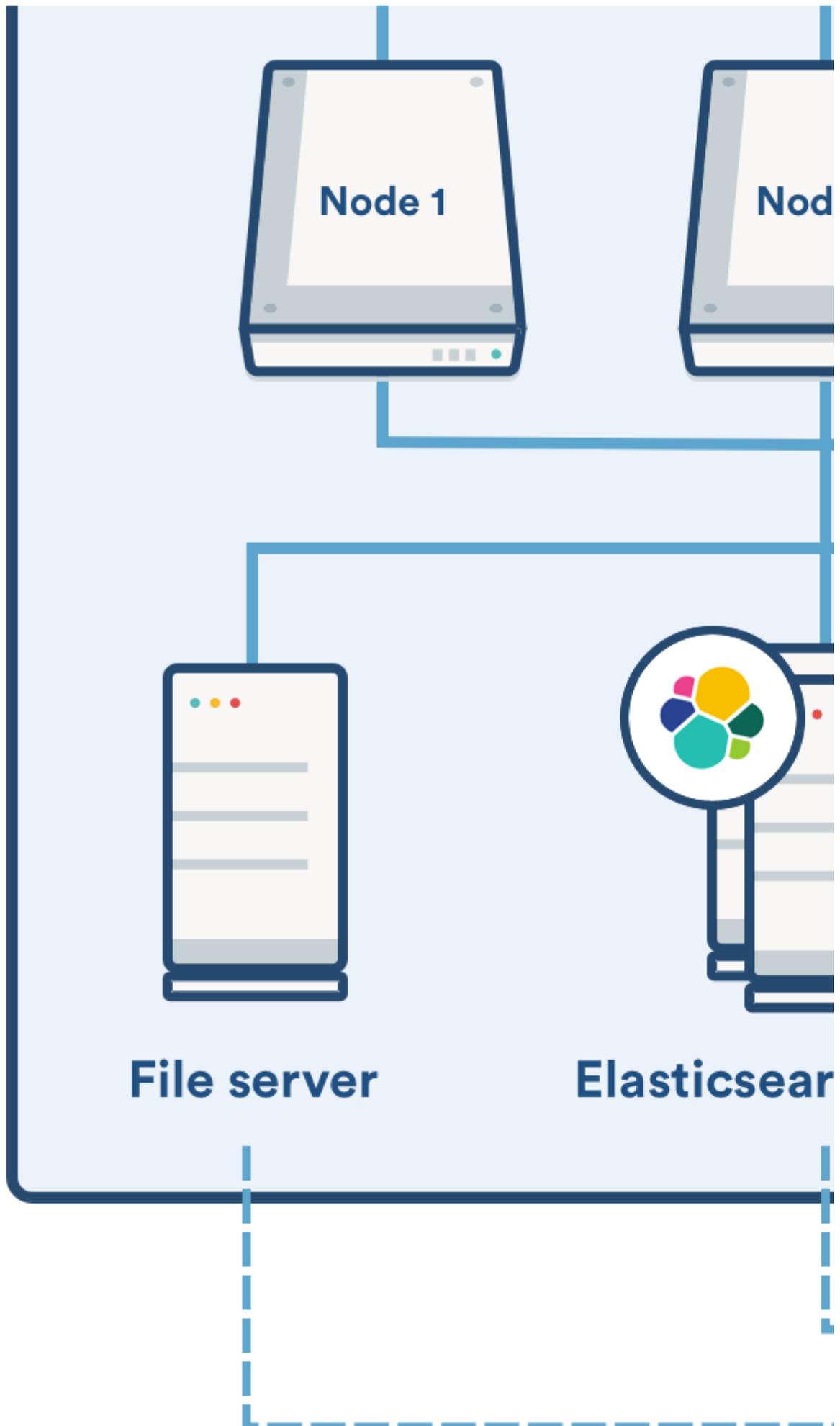
Disaster recovery A strategy to resume operations in an alternate data center (usually in another geographic location), in the event of a disaster whereby the main data center becomes unavailable. Failover (to another location) is a fundamental part of disaster recovery.

Failover is when one machine takes over from another machine, when the aforementioned machine fails. This could be within the same data center or from one data center to another. Failover is usually part of both high availability and disaster recovery planning.

Components

A typical deployment of Bitbucket with a standby for Disaster Recovery is depicted in the following diagram. The standby Bitbucket instances are "cold" (i.e., not running) and the shared file server, database and (optionally) Elasticsearch instances are "warm" (i.e., running) so that replication can occur.





Example component diagram of a typical DR deployment.

Replication of data sources

Replication of your primary Bitbucket system to a standby system requires:

1. your shared home directory to be on a file system which supports atomic snapshot-level replication to a remote standby, and
2. your database to be capable of replication to a remote standby.

File system

The shared home directory contains your repository data, log files, user-installed plugins, and so on (see [Set the home directory](#) for more detail). You need to replicate your shared home directory onto your standby file server. This process needs to be quick, reliable, and incremental to ensure the standby is as up to date as possible.

⚠️ You must choose a file system replication technology that will maintain the data integrity of the Git repositories on disk. ⚠️

For example, a replication technology such as RSync would not meet this requirement, as a repository might change during the transfer, causing corruption.

The best way to achieve the desired level of consistency is to use a file system which supports atomic block level snapshots. All files in the snapshot are effectively frozen at the same point in time regardless of how long the replication to the standby takes. The [examplescripts](#) implement [ZFS](#) as the file system for the Bitbucket shared home, as it allows light weight snapshots which can be transferred and restored on a remote system.

Database

The database contains data about pull requests, comments, users, groups, permissions, and so on. You need to replicate your database and continuously keep it up to date to satisfy your RPO.

See the [Supported platforms - Databases](#) section for more details about choosing a supported database technology.

- Oracle: <http://www.oracle.com/technetwork/database/features/data-integration/index.html>
- PostgreSQL: https://wiki.postgresql.org/wiki/Binary_Replication_Tutorial
- Amazon Relational Database Service (RDS): <https://aws.amazon.com/rds/>

Elasticsearch

This is not technically a key data store as its data derives from a combination of database and repository content. Bitbucket search detects an out of date index and incrementally reconstructs it. However, rebuilding the search index, especially for large installations, can be time consuming, during which search functionality will be impacted.

When Elasticsearch is unavailable the core Bitbucket functionality continues to operate normally. Only the search functionality is impacted if Elasticsearch is offline or out of date. Bitbucket will serve search queries while its index is being rebuilt, but the search results will not be entirely complete or accurate until the index is completely rebuilt. Elasticsearch can be replicated at a lower frequency than the database and file system as Bitbucket only incrementally updates its search index. For example if the search index is a day out of sync, only the changes for the last day need to be indexed. Here are some examples of strategies for Elasticsearch replication:

- No replication Bitbucket will rebuild the search index when the standby instance comes online
- Backup and restore (low frequency) Make regular backups of Elasticsearch and restore them on the standby instance. Bitbucket will bring the search index up to date when the standby comes online
- Backup and restore (high frequency) Snapshot the primary and restore snapshot on the standby at a higher frequency to minimize time to bring the index up to date when the standby comes online

The [examplescripts](#) implement Elasticsearch snapshots using either s3 or the shared filesystem as the snapshot repository. These scripts will configure the snapshot repository and take manual snapshots.

These data sources comprise the entire state of your Bitbucket instance. Files added through other means, such as files added by plugins, will require another means of replication. In these cases, contact the plugin vendor for recommendations.

Worked example scripts

Atlassian provides some worked example scripts in the [atlassian-bitbucket-diy-backup](#) repository that can optionally be used to automate the Disaster Recovery replication and failover process. Worked examples are currently provided only for the following technologies:

File system	Database
<ul style="list-style-type: none"> • ZFS filesystem 	<ul style="list-style-type: none"> • PostgreSQL Log-Shipping Standby Servers • Amazon RDS Read Replicas

These scripts are meant to serve as a starting point, rather than a fully worked DR solution, and should only be used after first configuring and customizing them for your specific environment.

To get the scripts, clone or pull the latest version of [atlassian-bitbucket-diy-backup](#).

```
git clone git@bitbucket.org:atlassianlabs/atlassian-bitbucket-diy-backup.git
```

The scripts must be configured before use by copying the appropriate `bitbucket.diy-backup.vars.sh.example` to `bitbucket.diy-backup.vars.sh`, and editing to match your environment. Valid configurations for Disaster Recovery are:

```
STANDBY_HOME_TYPE=zfs
STANDBY_DATABASE_TYPE=amazon-rds or postgresql
```

You must configure additional variables for these home and database types, and install the same scripts on both the primary and standby systems to use them for Disaster Recovery. Refer to `bitbucket.diy-backup.vars.sh.example` in the [atlassian-bitbucket-diy-backup](#) repository for more information on configuring the scripts for Disaster Recovery.

Update to the latest version of the backup scripts

When you create an EBS snapshot, the scripts now tag it with the name of the EBS volume. This 'Device' tag allows the snapshot to be automatically cleaned up, and without it you cannot restore your snapshot.

If you took your snapshot with scripts created before 18 September 2018, it won't have a 'Device' tag and you won't be able to restore it. To fix this, use the AWS console to add a tag to the EBS snapshot with with "Device" as the key and "<device_name>" as the value. <device_name> should be the device name of the EBS volume holding the shared home directory (e.g. "Device" : "/dev/xvdf").

Setting up a standby system

Step 1. Install Bitbucket Data Center 4.8 or later

Install Bitbucket Data Center on the standby instance the same way you would set up a primary instance. The standby instance is effectively an exact replica of the primary instance, and therefore requires all the components deployed in the primary to be deployed in the standby. This includes a single node or clustered data center instance, database, Elasticsearch and a file server to serve the shared home folder. See [Installing Bitbucket Data Center](#) for specific, detailed installation procedures.

DO NOT start Bitbucket on the standby

If you start the Bitbucket service before the failover process, the database and file server replication may fail, preventing the standby from serving as a replica. Starting your standby instance to test your setup is discussed later in the document in the section [Disaster recovery testing](#).

Step 2. Set up replication to the standby instance

This section contains instructions for setting up replication to your standby instance, so that in the event of a disaster you can failover with minimal data loss.

1. **Set up file server replication**- Set up your standby file server as a replica of your primary. Only the volume containing Bitbucket's shared home directory and any [added data stores](#) need to be replicated. Refer to your file server vendor's documentation for more information.

If using the [atlassian-bitbucket-diy-backup](#) worked example scripts, then you can run this command on the *primary* file server:

```
./setup-disk-replication.sh
```

2. **Set up database replication** -Set up your standby database as a read replica of your primary database. Refer to your database vendor's documentation for more information.

If using the [atlassian-bitbucket-diy-backup](#) worked example scripts, then you can run this command on the *standby* database:

```
./setup-db-replication.sh
```

3. **Set up Elasticsearch replication (optional)**-The developers of Elasticsearch, provide detailed instructions on setting up snapshot repositories and how the snapshots contained in these repositories can be transferred to the standby cluster. See their article titled [Snapshot And Restore](#) for more information. The [examplescripts](#) will configure the snapshot repository and take manual snapshots using either an s3 or shared filesystem repository.

Step 3. Initiate replication

Once you setup replication, ensure the primary instance is replicating to the standby instance. The method of replication varies depending on the chosen technology. Once set up correctly, most database replication technologies are automatic and do not require ongoing manual steps. File system replication technologies may require the ongoing transfer of snapshots from the primary system to the standby to be automated (for example, via `cron`).

If using the [atlassian-bitbucket-diy-backup](#) worked example scripts, then you can manually test file system replication by running this command on the *primary* file server.

```
./replicate-disk.sh
```

Once file system replication is working, you may add a `crontab` entry for this command on a regular interval (e.g., every minute) as determined by your organization's RPO.

Disaster recovery testing

Practice makes perfect, and this applies to verifying your DR failover process, but there is a serious caveat. As Bitbucket's configuration lives in the shared home folder which is replicated from the primary, the settings in the `bitbucket.properties` file points at production instances, which must be changed before starting the standby instance for testing.

You should exercise extreme care when testing any disaster recovery plan. Simple mistakes could cause your live instance to be corrupted, for example, if testing updates are inserted into a production database. You may detrimentally impact your ability to recover from a real disaster while testing your disaster recovery plan.

It is important to note that your standby instance will be configured as if it was the primary, which can cause issues when testing your disaster recovery plan. When running Bitbucket in disaster recovery mode, the integrity checker may attempt to correct errors it finds when the file server is out of sync with the database. This includes merging, re-opening, and declining pull requests which can cause Jira tickets to update via applinks, and could also send out email notifications.

Before testing your DR deployment

Before testing, isolate your standby instance!

Keep the standby data center as isolated as possible from production systems during testing. You will need to prevent replication from the primary whilst testing your standby instance.

To ensure your standby instance is isolated from your primary instance

1. **Isolate your database** -Temporarily pause all replication to the standby database, then promote the standby database.

If using the [atlassian-bitbucket-diy-backup](#) worked example scripts, you can run this command on the *standby* database:

```
./promote-db.sh
```

2. **Isolate your file system** -Temporarily pause replication to the standby file system, then promote the standby file system.

If using the [atlassian-bitbucket-diy-backup](#) worked example scripts, you can run this command on the *standby* file server:

```
./promote-home.sh
```

3. **Edit the `bitbucket.properties` file on the standby**-Update the `bitbucket.properties` file located in the Bitbucket shared home on the standby file server. Change these properties:

- a. The JDBC connection properties.
- b. The Elasticsearch connection properties.
- c. Set `disaster.recovery=true`.
- d. Update any other settings that point at production instances to use their standby counterpart.

Performing the disaster recovery testing

Once you have isolated your standby instance, you can now test your disaster recovery plan.

To test your DR deployment

1. Ensure that the standby database has been promoted and is writable.
2. Ensure that the new shared home directory is ready.
3. Ensure your `bitbucket.properties` file is correctly configured.
4. Start Bitbucket.
5. Monitor the Bitbucket log file and check for consistency issues, as described in [Running integrity checks in Bitbucket Data Center](#).

After testing, reset your DR deployment

Resetting your DR deployment involves restoring the standby components into a state whereby replication can occur, and this state differs based on the chosen file server and database replication technology. In many cases it may be easier to setup your standby infrastructure again.

To reset your DR deployment

1. Once validation of the standby instance is complete, stop Bitbucket.
2. Restore the file server to a state in which replication can begin.
3. Restore the database to a state in which replication can begin.

Handling a failover

In the event your primary instance is unavailable, you need to failover to your standby system. This section describes how to do this, including instructions on how to check the data in your standby system.

To failover to your standby instance

1. **Promote your standby database** - Ensure the standby database is ready to accept connections and can no longer receive any further updates from the primary. Refer to your database vendor's documentation for more information.

If using the [atlassian-bitbucket-diy-backup](#) worked example scripts, then you can run this command on the *standby* database:

```
./promote-db.sh
```

2. **Promote your standby file server** - Ensure the standby file server can not receive any further updates from the primary. Refer to your file server vendor's documentation for more information.

If using the [atlassian-bitbucket-diy-backup](#) worked example scripts, then you can run this command on the *standby* file server:

```
./promote-home.sh
```

3. **Edit the `bitbucket.properties` file on the standby** - Update the `bitbucket.properties` file located in the Bitbucket shared home on the standby file server. Set `disaster.recovery=true`, and ensure all the properties are appropriate for the standby environment. At a minimum, check that these properties are defined:

```
disaster.recovery=true  
  
jdbc.url=<the URL of your standby database>  
  
plugin.search.elasticsearch.baseurl=<the URL of your standby Elasticsearch instance>  
  
hazelcast.network.tcpip.members=<the IP addresses of your standby cluster nodes>
```

 Note that during normal replication, `bitbucket.properties` on the standby is identical to the primary system. During failover, it is important to ensure the configuration of your `bitbucket.properties` file matches the *standby* system's configuration before you start Bitbucket on the standby, or the standby may not start or attempt to connect to services in the primary's environment.

4. Start Bitbucket on one node in the standby instance.
5. Monitor the Bitbucket log file and check for consistency issues, as described in [Running integrity checks in Bitbucket Data Center](#).
6. If required, start the other Bitbucket nodes.
7. Update your DNS, HTTP Proxy, or other front end devices to route traffic to your standby server.
8. Update the mail server configuration if the mail server differs from the production mail server.

Returning to the primary instance

In most cases, you'll want to return to using your primary instance after you've resolved the problems that caused the disaster. There are basically two approaches for this:

1. Schedule a reasonably-sized outage window. Take a full backup of the standby system's home directory and database (for example, with the tools described in [Data recovery and backups](#)), and restore it on the primary system.
2. Run the replication and failover steps in reverse, where the standby system now takes the role of the primary, and the original primary system acts as a standby until you cut over.

Other resources

Troubleshooting

If you encounter problems after failing over to your standby instance, check these FAQs for guidance:

If your database doesn't have the data available that it should, then you'll need to restore the database from a backup.

Application links are stored in the database. If the database replica is up to date, then the application links will be preserved.

You do, however, also need to consider how each end of the link knows the address of the other:

- If you use host names to address the partners in the link and the backup Bitbucket server has the same hostname, via updates to the DNS or similar, then the links should remain intact and working.
- If the application links were built using IP addresses and these aren't the same, then the application links will need to be re-established.
- If you use IP addresses that are valid on the internal company network but your backup system is remote and outside the original firewall, you'll need to re-establish your application links.

Adding and removing Data Center nodes

You can rapidly scale the capacity of Bitbucket Data Center, with nodowntime, by provisioning extra cluster nodes. This page describes how to add another cluster node to an *existing* instance of Bitbucket Data Center.

If you are *moving to Bitbucket Data Center from a single instance of Bitbucket Server*, go straight to [Installing Bitbucket Data Center](#), instead.

If you are *new to Bitbucket Data Center*, we suggest you take a look at [Clustering with Bitbucket](#).

Things to consider before provisioning a cluster node

We highly recommend provisioning cluster nodes using an automated configuration management tool such as Chef, Puppet, or Vagrant, or by spinning up identical virtual machine snapshots.

Minimum requirements for cluster nodes

Each Bitbucket cluster node runs the Bitbucket Data Center web application, and each should meet these minimum requirements:

- Each Bitbucket cluster node must be a dedicated machine, although machines may be physical or virtual.
- Cluster nodes must be connected in a high speed LAN (that is, they must be physically in the same data center).
- Bitbucket Server [supported platforms](#) requirements, including those for Java and Git, apply to each cluster node.
- For consistent performance, each cluster node should be nearly identical, or as similar as possible.
- All cluster nodes must run the same version of Bitbucket Data Center.
- All cluster nodes must have synchronized clocks (for example, using NTP) and be configured with the identical timezone.

Adding a node

1. Mount the shared home directory on the node

Bitbucket Data Center makes use of a shared file system that lives on a dedicated machine and is accessible using NFS. See [Installing Bitbucket Data Center](#) for more information.

Mount the shared home directory as `${BITBUCKET_HOME}/shared`. For example, suppose your Bitbucket home directory is `/var/atlassian/application-data/bitbucket`, and your shared home directory is available as an NFS export called `bitbucket-san:/bitbucket-shared`. Add the following line to `/etc/fstab` on the cluster node:

`/etc/fstab`

```
bitbucket-san:/bitbucket-shared /var/atlassian/application-data/bitbucket/shared nfs nfsvers=3,
lookupcache=pos,noatime,intr,rsize=32768,wsize=32768 0 0
```

Then mount it:

```
mkdir -p /var/atlassian/application-data/bitbucket/shared
sudo mount -a
```

2. Install Bitbucket Data Center on the node

Download the latest Bitbucket Data Center distribution from <https://www.atlassian.com/software/bitbucket/download>, and install Bitbucket Server on the cluster node by either [using the installer](#) or by [installing the application manually from an archive file](#).

3. Add the node to the cluster

Start Bitbucket Server on the new node. See [Start and stop Bitbucket](#). You can optionally give the node a persistent, human readable name by setting a `node.name` system property under `JVM_SUPPORT_RECOMMENDED_ARGS` in `_start-webapp.sh`. For example:

```
-Dcluster.node.name=bitbucket-1
```

Once Bitbucket Server starts, go to `http://<load-balancer>/admin/clustering`. You should see the new node listed, which would look similar to this:

	Node ID	Node name	Cluster Address
	1f781654-e246-4737-84ea-6b11f59255d2	bitbucket-app-2	172.25.1.139:5701
	ca9456d1-6815-410b-b37c-3d5763a6bb88	bitbucket-app-1	172.25.1.12:5701

New nodes can join the cluster without downtime. [Learn how to add a node.](#)

Verify that the new node you have started up has successfully joined the cluster. If it does not, check your network configuration and the `/${BITBUCKET_HOME}/log/atlassian-bitbucket.logfiles` on all nodes. If you are unable to find a reason for the node failing to join successfully, contact [Atlassian Support](#).

4. Connect the node to the load balancer

Bitbucket Data Center makes use of a load balancer to distribute requests from your users to the cluster nodes. If a cluster node goes down, the load balancer immediately detects the failure and automatically directs requests to the other nodes within seconds. See [Installing Bitbucket Data Center](#) for more information.

If you are using a hardware or software load balancer other than HAProxy, consult your vendor's documentation on how to add the new Bitbucket cluster node to the load balancer.

If you are using HAProxy, simply add the following lines to your `haproxy.cfg` file:

```
# In the backend bitbucket_http_backend section, add:
server bitbucket<xx> 192.168.0.<x>:7990 check inter 10000 rise 2 fall 5
```

```
# In the backend bitbucket_ssh_backend section, add:
server bitbucket<xx> 192.168.0.<x>:7999 check port 7999
```

where the values for `<x>` and `<xx>` don't conflict with an existing node.

Now restart HAProxy :

```
sudo service haproxy restart
```

Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure that all are receiving traffic. Also check that updates done on one node are visible on the other nodes.

You can monitor the health of your cluster by navigating to HAProxy's statistics page at `http://<load-balancer>:8090/`. You should see a page similar to this:

HAProxy version 1.5.0, released 2014/06/19

Statistics Report for pid 28972

> General process information

pid = 28972 (process #1, nproc = 1)
 uptime = 0d 20h34m58s
 system limits: memmax = unlimited; ulimit-n = 8019
 maxsock = 8019; maxconn = 4000; maxpipes = 0
 current conns = 3; current pipes = 0/0; conn rate = 2/sec
 Running tasks: 1/17; idle = 100 %

Display option:
 • Scope :
 • [Hide 'DOWN' servers](#)
 • [Refresh now](#)
 • [CSV export](#)

External resources:
 • [Primary site](#)
 • [Updates \(v1.5\)](#)
 • [Online manual](#)

stash_http_frontend

	Queue			Session rate			Sessions				Bytes		Denied		Errors			Warnings			Server												
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle			
Frontend	0	12	-	1	22	3000	60	503					45	043	864	381	514	405	0	0	127				OPEN								

stash_cluster_http_backend

	Queue			Session rate			Sessions				Bytes		Denied		Errors			Warnings			Server														
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle					
stash-app-1	0	0	-	0	25	0	10	-	49	705	45	103	3s	24	674	486	194	031	639	0	0	0	0	20h34m	UP	L7OK/200	in	3ms	1	Y	-	9	0	0s	-
stash-app-2	0	0	-	0	38	0	10	-	48	129	45	103	3s	20	369	378	187	456	692	0	2	0	0	20h34m	UP	L7OK/200	in	3ms	1	Y	-	10	0	0s	-
Backend	0	0	-	0	36	1	22	300	137	605	90	206	3s	45	043	864	381	514	405	0	0	2	0	20h34m	UP				2	2	0	0	0s	-	

stash_esh_frontend

	Queue			Session rate			Sessions				Bytes		Denied		Errors			Warnings			Server											
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle		
Frontend	1	15	-	0	11	50	145	340					391	467	702	1	493	870	815	0	0	0		OPEN								

stash_cluster_esh_backend

	Queue			Session rate			Sessions				Bytes		Denied		Errors			Warnings			Server														
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle					
stash-app-1	0	0	-	0	7	0	7	-	54	411	54	411	9s	195	799	631	761	383	500	0	0	5	0	20h34m	UP	L4OK	in	0ms	1	Y	-	0	0	0s	-
stash-app-2	0	0	-	0	7	0	6	-	54	410	54	410	13s	195	668	071	732	487	315	0	0	6	0	20h34m	UP	L4OK	in	0ms	1	Y	-	0	0	0s	-
Backend	0	0	-	1	15	0	11	5	145	340	108	821	9s	391	467	702	1	493	870	815	0	0	11	0	20h34m	UP				2	2	0	0	0s	-

Removing a node

To remove a node, stop Bitbucket on that node. You can then remove the installation and local home directory as required.

To see the number of nodes remaining, go to **Administration > Clustering**.

Scaling up and down in AWS and Azure

If you deployed Bitbucket Data Center on AWS or Azure, your Bitbucket nodes will be in scaling groups. You will add and remove nodes either by changing the minimum and maximum size of each group or using a scaling plan. See the following resources for more info:

- [Getting started with Bitbucket Data Center on AWS](#)
- [Administering Bitbucket Data Center in Azure](#)

Moving to a non-clustered Data Center

If you no longer need clustering, but still want access to Data Center features, you can go back to a non-clustered (single node) Data Center installation. In these instructions we'll assume that you'll use one of your existing cluster nodes as your new, standalone installation. You'll also need to make some infrastructure changes as part of the switch.

What about the features?

After moving to a single node Data Center, you will lose capabilities that are based on clustering, such as instant scalability or performance and scale. For the complete list of features, see [Server and Data Center feature comparison](#).

Created in 2020 by Atlassian. Licensed under a [Creative Commons Attribution 2.5 Australia License](#).

Before you begin

We recommend completing this process in a staging environment, and running a set of functional tests, integration tests, and performance tests, before making these changes in production.

1. Shut down Bitbucket

Stop all cluster nodes before you proceed.

2. Configure your load balancer

Configure your load balancer to redirect traffic away from all Bitbucket nodes, except the node you plan to use for your standalone installation.

If you no longer need your load balancer, you can remove it at this step.

3. Move items in the cluster shared home back to local home

1. Unmount the `<shared home>` directory and copy its contents to the `<local home>/shared` directory.

 Make sure that the path to the shared subdirectory remains the same, as Git repositories are using absolute paths to other repositories and will break if the path changes.

4. Start Bitbucket

Restart Bitbucket.

 To confirm you're now running a standalone installation, go to **Administration > Clustering**, which will show you the number of nodes in your cluster.

Deploy Bitbucket Data Center in AWS

- [Non-clustered VS clustered environment](#)
- [Deploying Bitbucket Data Center in a cluster using the AWS Quick Start](#)
 - [Advanced customizations](#)
 - [Launching the Quick Start from your own S3 bucket \(recommended\)](#)
 - [Amazon Aurora database for high availability](#)
 - [Amazon CloudWatch for basic monitoring and centralized logging](#)
 - [Auto Scaling groups](#)
 - [Customizing the AWS Quick Start's CloudFormation templates](#)
- [Administering Bitbucket Data Center in AWS](#)
- [Securing Bitbucket within AWS](#)
- [Performance guidelines](#)
- [Mirroring](#)

This page provides an overview of the options available for running self-managed Bitbucket Data Center and Bitbucket Server instances on Amazon Web Services.

Running Bitbucket on [Amazon Web Services](#) (AWS) gives you scalable computing capacity without the need to invest in hardware up front, while retaining control over where and how your code is hosted within your organisation.

To this end, Atlassian provides:

- an AWS Quick Start which deploys a clustered Bitbucket Data Center in a matter of minutes, using AWS best practices for security and availability
- an Amazon CloudFormation template that can be customised for different deployment needs while keeping the process automated
- an Amazon Machine Image (AMI) that can be used for running Bitbucket Server on EC2 as an application server building block in more heavily customised deployments
- tools and guidelines for manually deploying, backing up, restoring, sizing, and administering Bitbucket Server and Bitbucket Data Center instances on AWS

Non-clustered VS clustered environment

A single node is adequate for most Small or Medium size deployments, unless you need specific features that require clustering (for example, high availability). If you have an existing Server installation, you can still use its infrastructure when you upgrade to Data Center. Many features exclusive to Data Center (like [SAML single sign-on](#), [self-protection via rate limiting](#), and [CDN support](#)) don't require clustered infrastructure. You can start using these Data Center features by simply upgrading your Server installations license. For more information on whether clustering is right for you, check out [Atlassian Data Center architecture and infrastructure options](#).

Deploying Bitbucket Data Center in a cluster using the AWS Quick Start

[Get Started!](#)

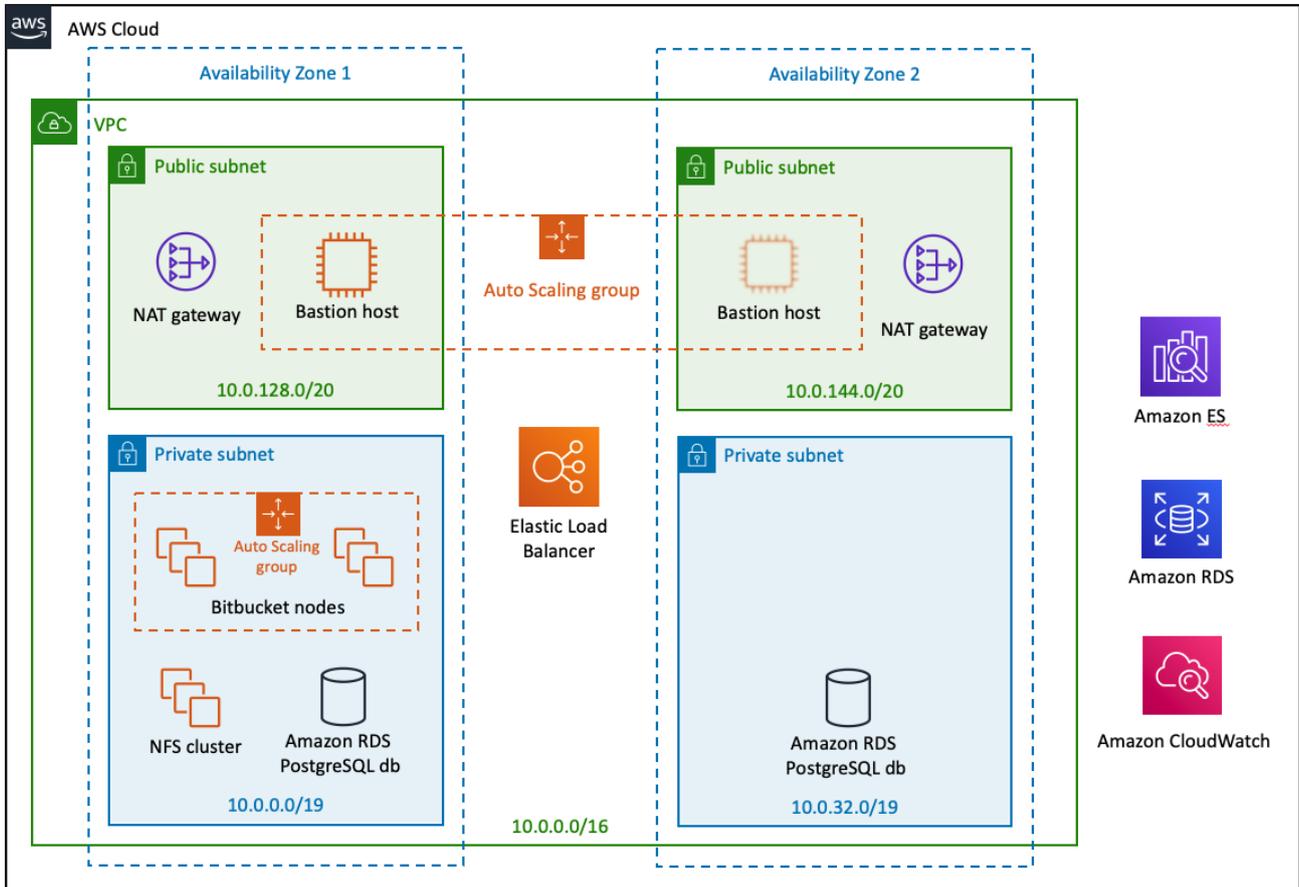
The simplest way to deploy a clustered Bitbucket Data Center on AWS is through the [AWS Quick Start](#). The Quick Start launches, configures, and runs the AWS compute, network, storage, and other resources required to deploy a specific workload on AWS, using AWS best practices for security and availability.

The Quick Start provides two deployment options, each with its own template. The first option deploys the Atlassian Standard Infrastructure (ASI) and then provisions Bitbucket Data Center into this ASI. The second option only provisions Bitbucket Data Center on an existing ASI.

Atlassian Standard Infrastructure (ASI)

The ASI is a virtual private cloud (VPC) that contains the components required by all Atlassian Data Center applications. For more information, see [Atlassian Standard Infrastructure \(ASI\) on AWS](#).

Here's an overview of the architecture deployed by the AWS Quick Start for Bitbucket Data Center:



The deployment consists of the following components:

- **Instances/nodes:** One or more Amazon Elastic Cloud (EC2) instances as cluster nodes, running Bitbucket.
- **Load balancer:** An Amazon Elastic Load Balancer (ELB), which works both as load balancer and SSL-terminating reverse proxy.
- **Database:** Your choice of shared database instance Amazon RDS or Amazon Aurora.
- **Storage:** A shared NFS server to store repositories in a common location accessible to all Bitbucket nodes.
- **Amazon CloudWatch:** Basic monitoring and centralized logging through Amazon's native CloudWatch service.
- An Amazon Elasticsearch Service domain for code and repository search.

For more information on the architecture, components and deployment process, see our [Quick Start Guide](#).

Advanced customizations

To get you up and running as quickly as possible, the Quick Start doesn't allow the same level of customization as a manual installation. You can, however, further customize your deployment through the variables in the [Ansible playbooks](#) we use.

All of our AWS Quick Starts use Ansible playbooks to configure specific components of your deployment. These playbooks are available publicly on this repository:

<https://bitbucket.org/atlassian/dc-deployments-automation>

You can override these configurations by using [Ansible variables](#). Refer to the repository's [README file](#) for more information.

Launching the Quick Start from your own S3 bucket (recommended)

The fastest way to launch the Quick Start is directly from its AWS S3 bucket. However, when you do, any updates we make to the Quick Start templates will propagate directly to your deployment. These updates sometimes involve adding or removing parameters from the templates. This could introduce unexpected (and possibly breaking) changes to your deployment.

For production environments, we recommend that you copy the Quick Start templates into your own S3 bucket. Then, launch them directly from there. Doing this gives you control over when to propagate Quick Start updates to your deployment.

1. Clone the [Quick Start templates](#) (including all of its submodules) to your local machine. From the command line, run:

```
git clone --recurse-submodules https://github.com/aws-quickstart/quickstart-atlassian-bitbucket
```

2. **(Optional)** The [Quick Start templates](#) repository uses the directory structure required by the Quick Start interface. If needed (for example, to minimize storage costs), you can remove all other files except the following:

```
quickstart-atlassian-bitbucket
submodules
  quickstart-atlassian-services
  templates
  quickstart-vpc-for-atlassian-services.yaml
  templates
quickstart-bitbucket-dc-with-vpc.template.yaml
quickstart-bitbucket-dc.template.yaml
```

3. Install and set up the [AWS Command Line Interface](#). This tool will allow you to create an S3 bucket and upload content to it.
4. Create an S3 bucket in your region:

```
aws s3 mb s3://<bucket-name> --region <AWS_REGION>
```

At this point, you can now upload the Quick Start templates to your own S3 bucket. Before you do, you'll have to choose which Quick Start template you'll be using:

- `quickstart-bitbucket-dc-with-vpc.template.yaml`: use this for deploying into a new ASI (end-to-end deployment).
- `quickstart-bitbucket-dc.template.yaml`: use this for deploying into an existing ASI.

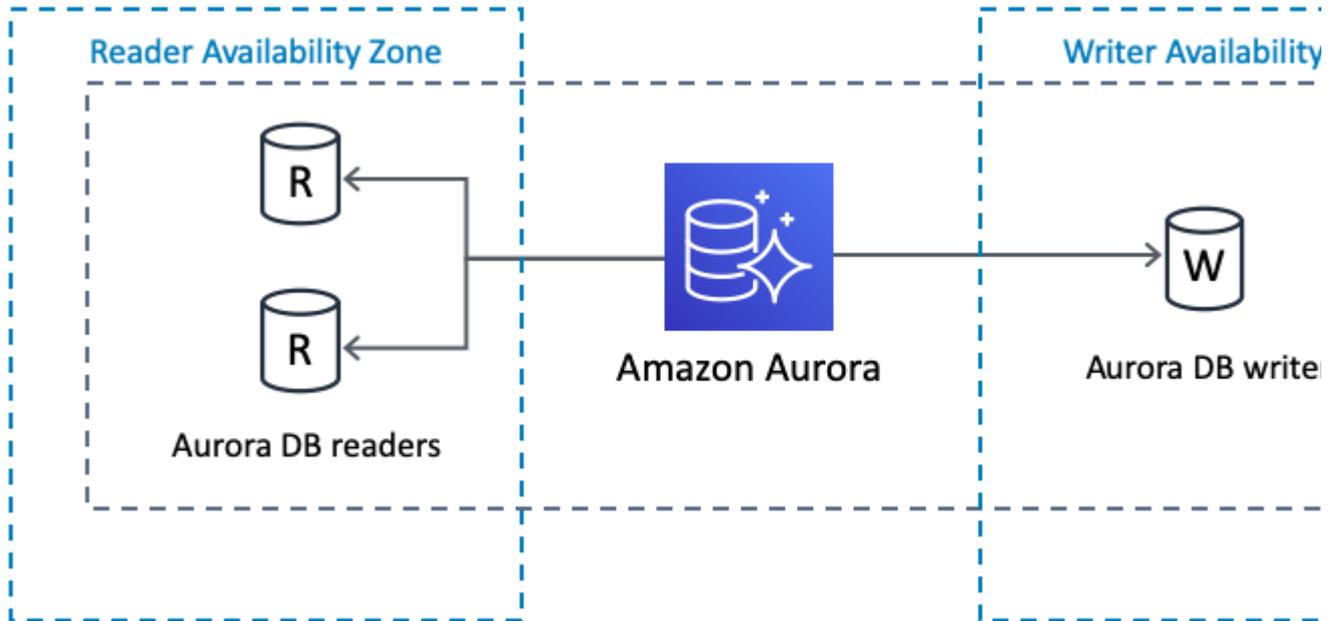
1. In the template you've chosen, the `QSS3BucketName` default value is set to `aws-quickstart`. Replace this default with the name of your S3 bucket.
2. Go into the parent directory of your local clone of the Quick Start templates. From there, upload all the files in local clone to your S3 bucket:

```
aws s3 cp quickstart-atlassian-bitbucket s3://<bucket-name> --recursive --acl public-read
```

3. Once you've uploaded everything, you're ready to deploy your production stack from your S3 bucket. Go to **CloudFormation Create Stack**. When specifying a template, paste in the Object URL of the Quick Start template you'll be using.

Amazon Aurora database for high availability

The Quick Start also allows you to deploy Jira Data Center with an Amazon Aurora clustered database (instead of RDS). This cluster will be PostgreSQL-compatible, featuring a primary database writer that replicates to two database readers. You can also set up the writers and readers in separate availability zones for better resiliency.



If the writer fails, Aurora automatically promotes one of the readers to take its place. For more information, see [Amazon Aurora Features: PostgreSQL-Compatible Edition](#).

i For instructions on manually setting up a new Amazon Aurora clustered database and connecting it to Bitbucket Data Center, see [Configuring Bitbucket Data Center to work with Amazon Aurora](#). Amazon Aurora is supported on [Bitbucket Data Center 6.7](#) and up.

Amazon CloudWatch for basic monitoring and centralized logging

The Quick Start can also provide you with node monitoring through [Amazon CloudWatch](#). This will allow you to track each node's CPU, disk, and network activity, all from a pre-configured [CloudWatch dashboard](#). The dashboard will be configured to display the latest log output, and you can customize the dashboard later on with additional monitoring and metrics.

By default, Amazon CloudWatch will also collect and store logs from each node into a single, central source. This centralized logging allows you to search and analyze your deployment's log data more easily and effectively. See [Analyzing Log Data with CloudWatch Logs Insights](#) and [Search Log Data Using Filter Patterns](#) for more information.

i Amazon CloudWatch provides basic logging and monitoring, but also costs extra. To help reduce the cost of your deployment, you can disable logging or turn off Amazon CloudWatch integration during deployment.

✓ To download your log data (for example, to archive it or analyze it outside of AWS), you'll have to export it first to S3. From there, you can download it. See [Exporting Log Data to Amazon S3](#) for details.

Auto Scaling groups

This Quick Start uses [Auto Scaling groups](#), but only to statically control the number of its cluster nodes. We don't recommend that you use Auto Scaling to dynamically scale the size of your cluster. Adding an application node to the cluster usually takes more than 20 minutes, which isn't fast enough to address sudden load spikes.

If you can identify any periods of high and low load, you can schedule the application node cluster to scale accordingly. See [Scheduled Scaling for Amazon EC2 Auto Scaling](#) for more information.

To study trends in your organization's load, you'll need to monitor the performance of your deployment. Refer to [Bitbucket Data Center sample deployment and monitoring strategy](#) for tips on how to do so.

Customizing the AWS Quick Start's CloudFormation templates

To get you up and running as quickly as possible, the Quick Start doesn't allow the same level of customization as a manual installation. Alternatively, you can customize the CloudFormation templates used by the Quick Start to fit your needs. These templates are available from the following repository:

<https://github.com/aws-quickstart/quickstart-atlassian-bitbucket>

Administering Bitbucket Data Center in AWS

See [Administering Bitbucket Data Center in AWS](#) for information about performing administration tasks on a Bitbucket instance within AWS, including:

- configuring variables when launching Bitbucket in AWS
- maintaining, resizing, upgrading, migrating, and customizing your Bitbucket deployment in AWS
- additional details about the components within the Bitbucket Server AMI

Securing Bitbucket within AWS

AWS is accessed over the public Internet, so it is important to apply appropriate security measures when running Bitbucket Server in AWS. See [Best practices for securing Bitbucket in AWS](#) for security guidance on a range of security topics, including Amazon Virtual Private Cloud (VPC), Security Groups, and SSL.

Performance guidelines

To get the best performance out of your Bitbucket deployment in AWS, it's important not to under-provision your instance's CPU, memory, or I/O resources. Whether you choose to deploy Bitbucket Data Center, which offers performance gains via horizontal scaling, or a single node Bitbucket Server instance, we have specific recommendations on choosing AWS EC2 and EBS settings for best performance per node.

If you are using the CloudFormation template, these settings are already included. Otherwise, see [Infrastructure recommendations for enterprise Bitbucket instances on AWS](#).

Mirroring

Smart Mirroring can drastically improve Git clone speeds for distributed teams working with large repositories. For an overview of the benefits to mirroring, see [Smart Mirroring](#). The [Bitbucket Data Center FAQ](#) also answers many common questions about smart mirroring (and mirroring in general).

For detailed instructions, see [Set up a mirror](#).

Administer Bitbucket Data Center in AWS

- [Connecting to your instance using SSH](#)
- [Configure SSL to enable HTTPS](#)
- [Backing up your instance](#)
- [Upgrading](#)
 - [Step 1: Terminate all running Bitbucket Data Center application nodes](#)
 - [Step 2: Update the version used by your Bitbucket Data Center stack](#)
 - [Step 3: Scale up the number of application nodes](#)
- [Stopping and starting your EC2 instance](#)
 - [If your EC2 instance becomes unavailable after stopping and restarting](#)
- [Migrating your existing Bitbucket Server or Bitbucket Data Center instance into AWS](#)
- [Resizing the data volume in your Bitbucket Server instance](#)
- [Moving your Bitbucket Server data volume between instances](#)
- [Scaling up and down](#)
 - [Vertical VS horizontal scaling](#)

While working with Bitbucket on AWS, you can expand your environment by adding additional nodes, upgrade the existing nodes, or connect to them over SSH.

Connecting to your instance using SSH

You can perform node-level configuration or maintenance tasks on your deployment through the [AWS Systems Manager Sessions Manager](#). This browser-based terminal lets you access your nodes without any SSH Keys or a Bastion host. For more information, see [Getting started with Session Manager](#).

Access via Bastion host

You can also access your nodes via a Bastion host (if you deployed one). To do this, you'll need your SSH private key file (the PEM file you specified for the **Key Name** parameter). Remember, this key can access all nodes in your deployment, so keep this key in a safe place.

The Bastion host acts as your "jump box" to any instance in your deployment's internal subnets. That is, access the Bastion host first, and from there access any instance in your deployment.

The Bastion host's public IP is the `BastionPubIp` output of your deployment's `ATL-BastionStacks` tack. This stack is nested in your deployment's [Atlassian Standard Infrastructure \(ASI\)](#). To access the Bastion host, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem ec2-user@<BastionPubIp>
```

The `ec2-user` has `sudo` access. SSH access is by `root` is not allowed.

Configure SSL to enable HTTPS

To enhance Bitbucket's security, you should use a proper SSL certificate obtained from a reputable Certificate Authority (CA). See [Securing Bitbucket in AWS](#) for instructions on how to do this.

Backing up your instance

The Atlassian Bitbucket Server AMI includes a complete set of Bitbucket Server DIY Backup scripts which has been built specifically for AWS. For instructions on how to backup and restore your instance please refer to [Using Bitbucket Server DIY Backup in AWS](#).

Upgrading

Before upgrading to a later version of Bitbucket Data Center:

1. [Check if your apps are compatible](#) with that version. [Update your apps](#) if needed. For more information about managing apps, see [Using the Universal Plugin Manager](#).
2. [Enable integrity checks](#) (if you haven't already).

We strongly recommend that you perform the upgrade first in a *staging* environment before upgrading your production instance. [How to establish staging server environments for Bitbucket Server](#) provides helpful tips on doing so.

Upgrading a Bitbucket Data Center on AWS involves three steps:

Step 1: Terminate all running Bitbucket Data Center application nodes

Set the number of application nodes used by the Bitbucket Data Center stack to 0. Then, update the stack.

1. In the AWS console, go to **Services > CloudFormation**. Select your deployments stack to view its Stack Details.
2. In the Stack Details screen, click **Update Stack**.
3. From the **Select Templates** screen, select **Use current template** and click **Next**.
4. You'll need to terminate all running nodes. To do that, set the following parameters to 0:
 - a. **Maximum number of cluster nodes**
 - b. **Minimum number of cluster nodes**
5. Click **Next**. Click through the next pages, and then to apply the change using the **Update** button.
6. Once the update is complete, check that all application nodes have been terminated.

Step 2: Update the version used by your Bitbucket Data Center stack

Set the number of application nodes used by Bitbucket Data Center to 1. Configure it to use the version you want. Then, update the stack again.

1. From your deployments Stack Details screen, click **Update Stack** again.
2. From the **Select Templates** screen, select **Use current template** and click **Next**.
3. Set the **Version** parameter to the version you're updating to.
4. Configure your stack to use one node. To do that, set the following parameters to 1:
 - a. **Maximum number of cluster nodes**
 - b. **Minimum number of cluster nodes**
5. Click **Next**. Click through the next pages, and then to apply the change using the **Update** button.

Step 3: Scale up the number of application nodes

You can now scale up your deployment to your original number of application nodes. For detailed instructions on how to do this, see [Scaling up and down](#).

Stopping and starting your EC2 instance

An EC2 instance launched from the Atlassian Bitbucket Server AMI can be stopped and started just as any machine can be powered off and on again.

When stopping your EC2 instance, it is important to first

1. Stop the `atlbitbucket`, `atlbitbucket_search`, and `postgresql93` services.
2. Unmount the `/media/atl` filesystem.

If your EC2 instance becomes unavailable after stopping and restarting

When starting your EC2 instance back up again, if you rely on Amazon's automatically assigned [public IP address](#) (rather than a fixed private IP address or Elastic IP address) to access your instance, your IP address may have changed. When this happens, your instance can become inaccessible and display a "The host name for your Atlassian instance has changed" page. To fix this you need to update the hostname for your Bitbucket Server instance.

To update the hostname for your Bitbucket Server instance

1. Restart the Bitbucket service on all application nodes by running this command, which will update the hostname

```
sudo service atlbitbucket restart
```

2. Wait for Bitbucket Server to restart.
3. If you have also set up Bitbucket Server's Base URL to be the public DNS name or IP address be sure to also [update Bitbucket Server's base URL in the administration screen](#) to reflect the change.

Migrating your existing Bitbucket Server or Bitbucket Data Center instance into AWS

Migrating an existing instance to AWS involves moving consistent backups of your `${BITBUCKET_HOME}` and your database to the AWS instance.

To migrate your existing instance into AWS

1. Check for any known migration issues in the [Bitbucket Server Knowledge Base](#).
2. Alert users to the forthcoming service outage.
3. [Create a user](#) in the Bitbucket Server Internal User Directory with `SYSADMIN` permissions to the instance so you don't get locked out if the new server is unable to connect to your User Directory.
4. Take a backup of your instance with either the [Bitbucket Server Backup Client](#) (Bitbucket Server only) or the [Bitbucket Server DIY Backup](#) (Bitbucket Server or Data Center).
5. Launch Bitbucket Server in AWS using the [Quick Start instructions](#), which uses a CloudFormation template.
6. Connect to your AWS EC2 instance with SSH and upload the backup file.
7. Restore the backup with the same tool used to generate it.
8. If necessary, update the JDBC configuration in the `${BITBUCKET_HOME}/shared/bitbucket.properties` file.

Resizing the data volume in your Bitbucket Server instance

By default, the application data volume in an instance launched from the Atlassian Bitbucket Server AMI is a standard Linux ext4 filesystem, and can be resized using the standard Linux command line tools.

To resize the data volume in your Bitbucket Server instance

1. Stop the `atlbitbucket`, `atlbitbucket_search`, and `postgresql93` services.
2. Unmount the `/media/atl` filesystem.
3. Create a snapshot of the volume to resize.
4. Create a new volume from the snapshot with the desired size, in the same availability zone as your EC2 instance.
5. Detach the old volume and attach the newly resized volume as `/dev/sdf`.
6. Resize `/dev/sdf` using `resize2fs`, verify that its size has changed, and remount it on `/media/atl`.
7. Start the `atlbitbucket`, `atlbitbucket_search`, and `postgresql93` services.

For more information, see [Expanding the Storage Space of an EBS Volume on Linux](#), [Expanding a Linux Partition](#), and the Linux manual pages for `resize2fs` and related commands.

Moving your Bitbucket Server data volume between instances

Occasionally, you may need to move your Bitbucket Server data volume to another instance for example, when setting up staging or production instances, or when moving to an instance to a different availability zone.

There are two approaches to move your Bitbucket Server data volume to another instance

1. Take a backup of your data volume with Bitbucket Server DIY Backup, and restore it on your new instance. See [Using Bitbucket Server DIY Backup in AWS](#) for this option.
2. Launch a new instance from the Atlassian Bitbucket Server AMI with a snapshot of your existing data volume.

i A Bitbucket Server data volume may only be moved to a Bitbucket Server instance of the same or higher version than the original.

To launch a new instance from the Bitbucket Server AMI using a snapshot of your existing Bitbucket Server data volume

1. Stop the `atlbitbucket`, `atlbitbucket_search`, and `postgresql93services` on your existing Bitbucket Server instance.
2. Unmount the `/media/atl` filesystem.
3. Create a snapshot of the Bitbucket Server data volume (the one attached to the instance as `/dev/sdf`).
4. Once the snapshot generation has completed, launch a new instance from the Atlassian Bitbucket Server AMI as described in [Launch Bitbucket in AWS manually](#). When adding storage, change the EBS volume device to `/dev/sdf` as seen below and enter the id of the created snapshot.

Step 4: Add Storage
 Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Type	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root	snap-1b4bed30	10	General Purpose (SSD)	30 / 3000	<input type="checkbox"/>	Not Encrypted
EBS	snap-735d3027	100	General Purpose (SSD)	300 / 3000	<input type="checkbox"/>	Not Encrypted
Instance Store 0	N/A	N/A	N/A	N/A	N/A	Not Encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

5. If the host name (private or public) that users use to reach your Bitbucket Server instance has changed as a result of moving availability zones (or as a result of stopping an instance and starting a new one) you will need to SSH in and run `sudo /opt/atlassian/bin/atl-update-host-name.sh <newhostname>` where `<newhostname>` is the new host name.
6. Once Bitbucket Server has restarted your new instance should be fully available.
7. If the host name has changed you should also update the JDBC URL configuration in the `bitbucket.properties` file (typically located in `/var/atlassian/application-data/bitbucket/shared/`), as well as Bitbucket Server's base URL in the [administration screen](#) to reflect this.

Scaling up and down

To increase or decrease the number of application nodes:

1. Sign in to the AWS Management Console, use the region selector in the navigation bar to choose the AWS Region for your deployment, and open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation/>.

2. Click the **Stack name** of your deployment. This will display your deployment's **Stack info**. From there, click **Update**.
3. On the **Select Template** page, leave **Use current template** selected, and then choose **Next**.
4. On the **Specify Details** page, go to the **Cluster nodes** section of **Parameters**. From there, set your desired number of application nodes in the following parameters:
 - a. **Minimum number of cluster nodes**
 - b. **Maximum number of cluster nodes**
5. Click through to update the stack.



Disabled Auto Scaling

Since your cluster has the same minimum and maximum number of nodes, [Auto Scaling](#) is effectively disabled.

Setting different values for the minimum and maximum number of cluster nodes enables Auto Scaling. This dynamically scale the size of your cluster based on system load.

However, we recommend that you keep Auto Scaling disabled. At present, Auto Scaling can't effectively address sudden spikes in your deployment's system load. This means that you'll have to manually re-scale your cluster depending on the load.

Vertical VS horizontal scaling

Adding new cluster nodes, especially automatically in response to load spikes, is a great way to increase capacity of a cluster temporarily. Beyond a certain point, adding very large numbers of cluster nodes will bring diminishing returns. In general, increasing the size of each node (i.e., "vertical" scaling) will be able to handle a greater sustained capacity than increasing the number of nodes (i.e., "horizontal" scaling), especially if the nodes themselves are small. See [Recommendations for running Bitbucket in AWS](#) for more information.

Recommendations for running Bitbucket Data Center in AWS

Knowing your load profile is useful for planning your instance's growth, looking for inflated metrics, or simply keeping it at a reasonable size. In [Bitbucket Data Center load profiles](#), we showed you some simple guidelines for finding out if your instance was *Small*, *Medium*, *Large*, or *XLarge*. We based these size profiles on Server and Data Center case studies, covering varying infrastructure sizes and configurations.

A single node can be adequate for most Small or Medium size deployments, especially if you don't require high availability. If you have an existing Server installation, you can still use its infrastructure when you upgrade to Data Center. Many features exclusive to Data Center (like [SAML single sign-on](#), [self-protection via rate limiting](#), and [CDN support](#)) don't require clustered infrastructure. You can start using these Data Center features by simply upgrading your Server installations license.

✔ For more information on whether clustering is right for you, check out [Atlassian Data Center architecture and infrastructure options](#).

As your load grows closer to Large or XLarge, you should routinely evaluate your infrastructure. Once your environment starts to experience performance or stability issues, consider migrating to a clustered (or cluster-ready) infrastructure. When you do, keep in mind that it may not be always clear how to do that effectively for example, adding more application nodes to a growing Medium-sized instance doesn't always improve performance (in fact, the opposite might happen).

To help you plan your infrastructure set-up or growth, we ran a series of performance tests on typical Medium, Large, and XLarge instances. We designed these tests to get useful, data-driven recommendations for your clustered deployment's application and database nodes. These recommendations can help you plan a suitable clustered environment, one that is adequate for the size of your projected content and traffic.

ⓘ Note that large repositories might influence performance.

We advise that you [monitor performance](#) on a regular basis.

Approach

We ran all tests in AWS. This allowed us to easily define and automate multiple tests, giving us a large (and fairly reliable) sample.

Each part of our test infrastructure was provisioned from a standard AWS component available to all AWS users. This allows for easy deployment of recommended configurations. You can also use [AWS Quick Starts for deploying Bitbucket Server Data Center](#).

It also means you can look up specifications in [AWS documentation](#). This helps you find equivalent components and configurations if your organization prefers a different cloud platform or bespoke clustered solution.

Some things to consider

To effectively benchmark Bitbucket on a wide range of configurations, we designed tests that could be easily set up and replicated. Accordingly, when referencing our benchmarks for your production environment, consider:

- We didn't install apps on our test instances, as we focused on finding the right configurations for the *core* product. When designing your infrastructure, you need to account for the impact of apps you want to install.
- We used RDS with default settings across all tests. This allowed us to get consistent results with minimal setup and tuning.
- Our test environment used dedicated AWS infrastructure hosted on the same subnet. This helped minimize network latency.
- We used an internal testing tool called Tikit to simulate the influx of git packets. This gave us the ability to measure git request speeds without having to measure client-side git performance. It also meant our tests didn't unpack git refs, as the tool only receives and decrypts git data.

- The performance (response times) of git operations will be affected largely by repository size. Our test repositories averaged 14.2MB in size. We presume that bigger repositories might require stronger hardware.
- [Due to limitations in AWS, we initialized EBS volumes](#) (storage blocks) on the NFS servers before starting the test. Without disk initializations, there is a significant increase in disk latency, and test infrastructure slows for several hours.

We enabled **analytics** on each test instance to collect usage data. For more information, see [Collecting analytics for Bitbucket Server](#).

Methodology

Each test involved applying the same amount of traffic to a Bitbucket data set, but on a different AWS environment. We ran three series of tests, each designed to find optimal configurations for the following components:

- Bitbucket application node
- Database node
- NFS node

To help ensure benchmark reliability, we initialized the EBS volumes and tested each configuration for three hours. We observed stable response times throughout each test. Large instance tests used **Bitbucket Data Center 5.16** while XLarge used **Bitbucket Data Center 6.4**. We used a custom library (Trikit) running v1 protocol to simulate Git traffic.

Data sets

We created a Large-sized Bitbucket Data Center instance with the following dimensions:

Metric	Value (approximate)
Repositories	52,000
Active users	25,000
Pull requests	850,000
Traffic (git operations per hour)	40,000

Content and traffic profiles are based on [Bitbucket Data Center load profiles](#), which put the instances overall load profile at the highest level of *Large* profile. We believe these metrics represent a majority of real-life, Large-sized Bitbucket Data Center instances.

Metric	Value (approximate)
Users	25,000
Groups	50,000
Projects (including personal)	16,700
Comments on pull requests	3,500,000

Metric	Total	Component	Value (approximate)
Total repositories	52,000	Regular repositories	26,000
		Public forks	9,000
		Private repositories	17,000
Total pull requests	859,000	Pull requests open	8,500
		Pull requests merged	850,000

Traffic	40,000	Clones	16,000
(git operations per hour)		Fetches	14,000
		Pushes	10,000

We created an XLarge-sized Bitbucket Data Center instance with the following dimensions:

Metric	Value (approximate)
Repositories	110,000
Active users	50,000
Pull requests	1,790,000
Traffic (git operations per hour)	65,000

Content and traffic profiles are based on [Bitbucket Data Center load profiles](#), which put the instances overall load profile at the XLarge profile. We believe these metrics represent a majority of real-life, XLarge-sized Bitbucket Data Center instances.

Metric	Value (approximate)
Users	25,000
Groups	3,000
Projects (including personal)	52,000
Comments on pull requests	8,700,000

Metric	Total	Component	Value (approximate)
Total repositories	105,000	Regular repositories	52,000
		Public forks	17,000
		Private repositories	35,000
Total pull requests	1,790,000	Pull requests open	130,000
		Pull requests merged	1,660,000
Traffic (git operations per hour)	70,000	Clones	18,700
		Fetches	25,300
		Pushes	26,000

Benchmark

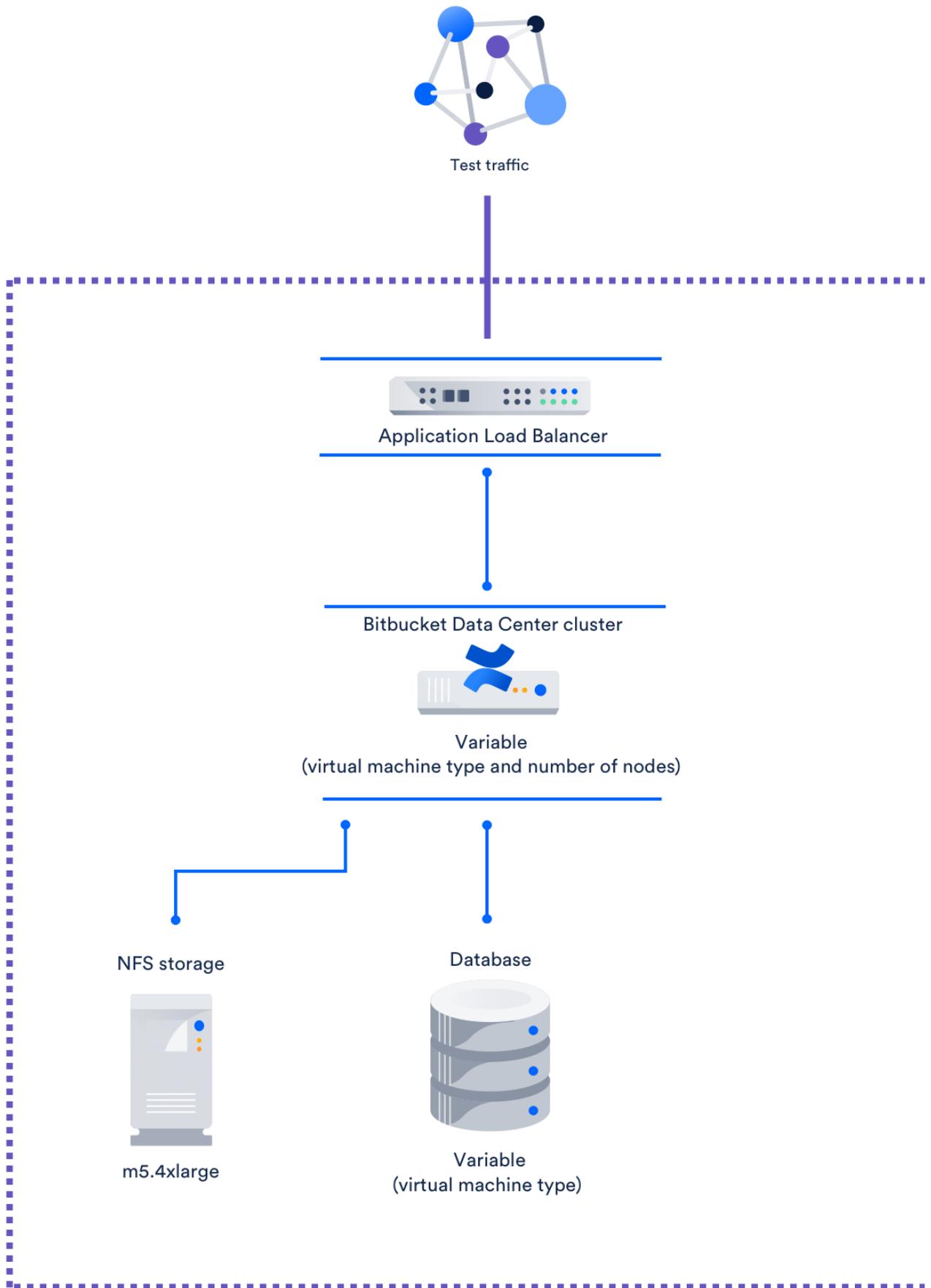
We used the following benchmark metrics for our tests.

Benchmark metric	Threshold	Reason
------------------	-----------	--------

Git throughput , or the number of git hosting operations (fetch/clone/push) per hour	32,700 (Minimum) for Large and 65,400 (Minimum) for XLarge , the higher the better	These thresholds are the upper limits of traffic defined in Bitbucket Data Center load profiles . We chose them due to the spiky nature of git traffic.
Average CPU utilization (for application nodes)	75% (Maximum) , the lower the better	When the application nodes reach an average of CPU usage of 75% and above, Bitbucket's adaptive throttling starts queuing Git hosting operations to ensure the responsiveness of the application for interactive users. This slows down Git operations.
Stability	No nodes go offline	When the infrastructure is inadequate in handling the load it may lead to node crashes.

 The test traffic had fixed sleep times to modulate the volume of git hosting operations. This means the benchmarked git throughput doesn't represent the maximum each configuration can handle.

Architecture



We tested each configuration on a freshly-deployed Bitbucket Server Data Center instance on AWS. Every configuration followed the same structure:

Function	Number of nodes	Virtual machine type	Notes
Application node	Variable	m5.xlarge m5.2xlarge m5.4xlarge m5.12xlarge m5.24xlarge	<p>When testing m5.xlarge (16GB of RAM), we used 8GB for JVM heap. For all others, we used 12GB for JVM heap. Minimum heap (Xms) was set to 1G for all the tests.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p> Weve observed that using a smaller JVM heap (2-3GB) is enough for most instances.</p> </div> <p>Also note that Git operations are expensive in terms of memory consumption and are executed outside of the Java virtual machine. See more on Scaling Bitbucket Server.</p> <p>Each Bitbucket application used 30GB General Purpose SSD (gp2) for local storage. This disk had an attached EBS volume with a baseline of 100 IOPS, burstable to 3,000 IOPS.</p>
Database	1	m5.xlarge m5.2xlarge m5.4xlarge	We used Amazon RDS Postgresql version 9.4.15, with default settings. Each test only featured one node.
NFS storage	1	m5.4xlarge m5.2xlarge m5.xlarge	<p>Our NFS server used a 900GB General Purpose SSD (gp2) for storage. This disk had an attached EBS volume with a baseline of 2700 IOPS, burstable to 3,000 IOPS. As mentioned, we initialized this volume at the start of each test.</p> <p>For more information on setting up Bitbucket Data Center's shared file server, see Step 2. Provision your shared file system (in Install Bitbucket Data Center). This section contains the requirements and recommendations for setting up NFS for Bitbucket Data Center.</p>
Load balancer	1	AWS Application Load Balancer (ELB)	We used AWS Elastic Load Balancer. Application Load Balancer at the time of performance testing doesn't handle SSH traffic.

We ran several case studies of real-life Large and XLarge Bitbucket Data Center instances to find optimal configurations for each component. In particular, we found many used m5 series virtual machine types ([General Purpose Instances](#)). As such, for the application node, we focused on benchmarking different series' configurations.

 Refer to the [AWS documentation on Instance Types](#) (specifically, [General Purpose Instances](#)) for details on each virtual machine type used in our tests.

Recommendations for Large-sized instances

We analyzed our benchmarks and came up with the following optimal configuration:

Best-performing and most cost-effective configuration

Component	Recommendation
-----------	----------------

Application nodes	m5.4xlarge nodes x 4
Database node	m5.2xlarge
NFS node	m5.2xlarge

Performance of this configuration

- Git throughput: 45,844 per hour
- Cost per hour ¹: \$4.168
- Average CPU utilization: 45%

 ¹ In our recommendations for Large-sized profiles, we quoted a *cost per hour* for each configuration. We provide this information to help inform you about the comparative price of each configuration. This cost only calculates the price of the nodes used for the Bitbucket application, database, and NFS nodes. It does not include the cost of using other components of the application like shared home and application load balancer.

These figures are in USD, and were correct as of July 2019.

We measured performance stability in terms of how far the instances average CPU utilization is from the 75% threshold. As mentioned, once we hit this threshold, git operations start to slow down. The further below the instance is from 75%, the less prone it is to slow due to sudden traffic spikes.

However, there are no disadvantages in using larger-size hardware (m5.12xlarge, for example), which will provide better performance.

Low-cost configuration

We also found a low-cost configuration with acceptable performance at **\$2.84** per hour:

Component	Recommendation
Application nodes	m5.4xlarge x 3
Database node	m5.xlarge
NFS node	m5.xlarge

This low-cost configuration offered a lower Git throughput of 43,099 git hosting calls per hour than the optimal configuration. However, this is still above our minimum threshold of 32,700 git hosting calls per hour. The trade-off for the price is fault tolerance. If the instance loses one application node, CPU usage spikes to 85%, which is above our maximum threshold. The instance will survive, but performance will suffer.

The following table shows all test configurations that passed our threshold, that is, above 32,500 git hosting operations per hour and below 75% CPU utilization, with no node crashes. We sorted each configuration by descending throughput.

Application nodes	Database node	NFS node	Git throughput	Cost per hour
m5.4xlarge x 6	m5.4xlarge	m5.4xlarge	46,833	6.800
m5.12xlarge x 2	m5.4xlarge	m5.4xlarge	45,848	6.792
m5.4xlarge x 4	m5.4xlarge	m5.4xlarge	45,844	5.264
m5.2xlarge x 8	m5.4xlarge	m5.4xlarge	45,626	5.264
m5.4xlarge x 3	m5.4xlarge	m5.4xlarge	44,378	4.496

m5.4xlarge x 3	m5.2xlarge	m5.4xlarge	43,936	3.784
m5.2xlarge x 6	m5.4xlarge	m5.4xlarge	43,401	4.496
m5.4xlarge x 3	m5.xlarge	m5.xlarge	43,099	2.840
m5.4xlarge x 3	m5.xlarge	m5.4xlarge	43,085	3.428

As you can see, the configuration m5.4xlarge x 4 nodes for the application doesn't provide the highest git throughput. However, configurations with higher throughput cost more and provide only marginal performance gains.

Recommendations for XLarge instances

We analyzed our benchmarks and came up with the following optimal configuration:

Best-performing configuration

Component	Recommendation
Application nodes	m5.12xlarge x 4
Database node	m5.2xlarge
NFS node	m5.2xlarge

Performance of this configuration

- Git throughput: 75,860 per hour
- Cost per hour¹: \$10.312
- Average CPU utilization: 65%

We measured performance stability in terms of how far the instances average CPU utilization is from the 75% threshold. As mentioned, once we hit this threshold, git operations start to slow down. The further below the instance is from 75%, the less prone it is to slow due to sudden traffic spikes.

 ¹In our recommendations for Extra Large-sized profiles, we quoted a *cost per hour* for each configuration. We provide this information to help inform you about the comparative price of each configuration. This cost only calculates the price of the nodes used for the Bitbucket application, database, and NFS nodes. It does not include the cost of using other components of the application like shared home and application load balancer.

These figures are in USD, and were correct as of July 2019.

Low-cost configuration

We also found a low-cost configuration with good performance at **\$7.02** per hour:

Component	Recommendation
Application nodes	m5.8xlarge x 4
Database node	m5.2xlarge
NFS node	m5.xlarge

This low-cost configuration offered a lower Git throughput of 74,275 git hosting calls per hour than the optimal configuration. However, this is still well above the defined threshold of 65,400 git hosting calls per hour. The trade-off for the price is fault tolerance. There were timeouts and errors observed on the m5.8xlarge x 3 nodes, so performance degradation may be encountered if the an application node goes down.

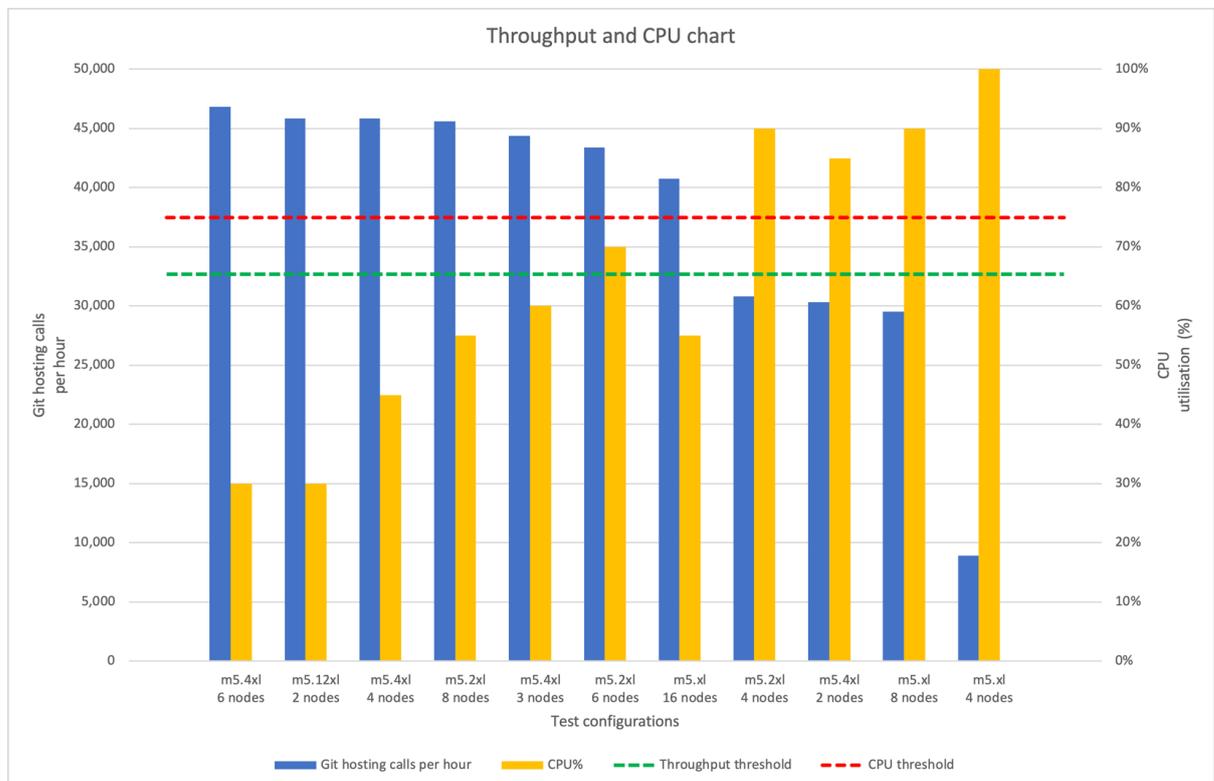
The following table shows all test configurations that passed our threshold, that is, above 32,500 git hosting operations per hour and below 75% CPU utilization, with no node crashes. We sorted each configuration by descending throughput.

Application nodes	Database node	NFS node	Git throughput	Cost per hour
m5.12xlarge x 4	m5.2xlarge	m5.2xlarge	75,860	\$ 10.31
m5.4xlarge x 8	m5.2xlarge	m5.2xlarge	73,374	\$ 7.24
m5.8xlarge x 4	m5.2xlarge	m5.xlarge	74,275	\$ 7.02
m5.4xlarge x 6	m5.2xlarge	m5.2xlarge	71,872	\$ 5.70
m5.12xlarge x 3	m5.2xlarge	m5.2xlarge	66,660	\$ 8.01

Application node test results

Our first test series focused on finding out which AWS virtual machine types to use (and how many) for the application node. For these tests, we used a single **m4.4xlarge** node for the database and single **m4.4xlarge** node for the NFS server.

Benchmarks show the best git throughput came from using m5.4xlarge (16 CPUs) and m5.12xlarge nodes (46 CPUs). You will need at **least three nodes for m5.4xlarge and two nodes for m5.12xlarge**.



CPU is underutilized at 30% for the following application node configurations:

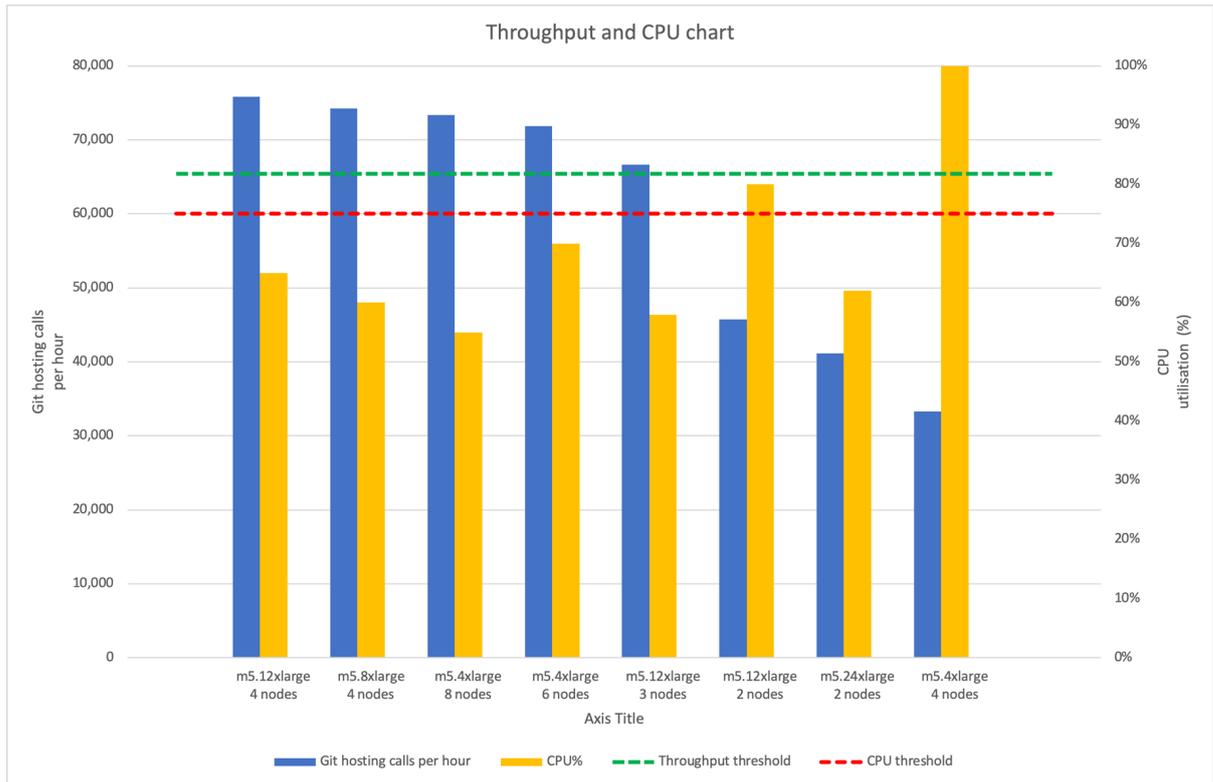
- m5.4xlarge x 6
- m5.12xlarge x 2

This demonstrates both configurations are overprovisioned. It would be more cost-effective to use **three or four m5.4xlarge nodes** for the application.

However, on the three-node m5.4xlarge set-up, the CPU usage would be at ~85% if one of the nodes failed. For this reason, we recommend the **four-node m5.4xlarge** set-up for better fault tolerance.

Our first test series focused on finding out which AWS virtual machine types to use (and how many) for the application node. For these tests, we used a single **m4.2xlarge** node for the database and single **m4.2xlarge** node for the NFS server.

Benchmarks show the best git throughput came from using **m5.12xlarge (48 CPUs)** and **m5.8xlarge nodes (32 CPUs)**. You will need four nodes for both instance types.



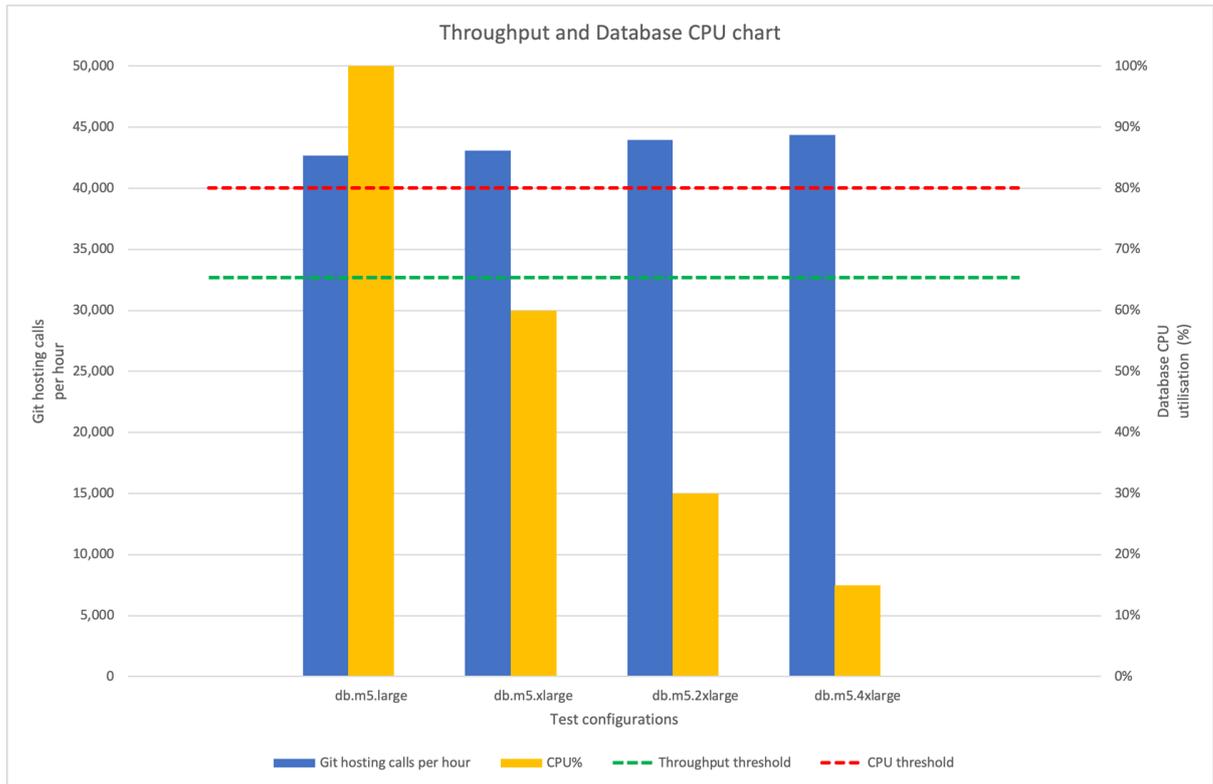
We have also carried out performance testing on 2 nodes (96 CPUs), but this resulted in poor performance, not meeting the threshold. Test results showed that 2 node deploys are not suitable for xlarge load. During the 2 node tests, the time spent on kernel was very high, which was not evident on 4+ nodes.

Database node test results

From the application node test series, we found using **three m5.4xlarge** nodes for the application yielded optimal performance (even if it wasn't the most fault tolerant). For our second test series, we tested this configuration against the following virtual machine types for the database:

- m4.large
- m4.xlarge
- m4.2xlarge
- m4.4xlarge

As expected, the more powerful virtual machine used, the better the performance. We saw the biggest gains in CPU utilization. Git throughput also improved, but only marginally.

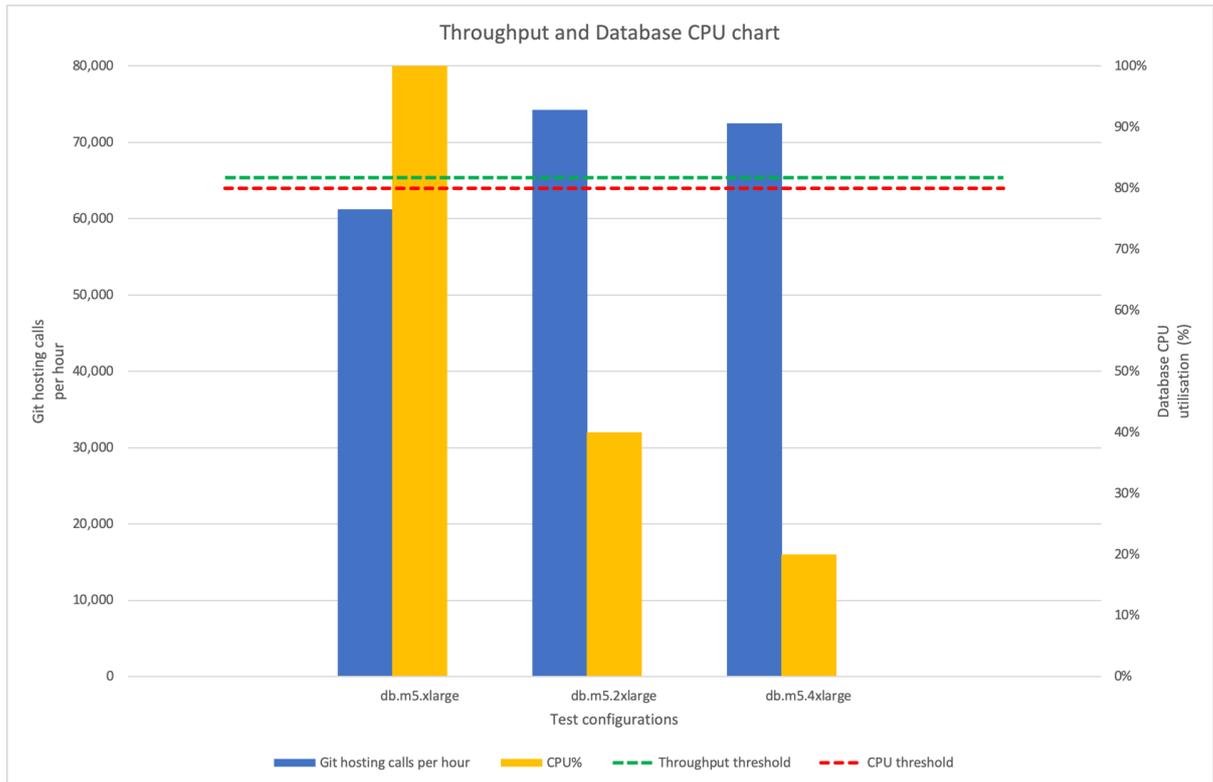


Only **m5.large** failed the CPU utilization threshold. All other tested virtual machine types are acceptable, although, **m5.xlarge** is pretty close to our CPU utilization threshold at 60%.

From the application node test series, we found using **four m5.12xlarge** nodes for the application yielded optimal performance. For our second test series, we tested this configuration against the following virtual machine types for the database:

- m4.xlarge
- m4.2xlarge
- m4.4xlarge

The m4.xlarge was saturated on CPU at 100%, and db.m4.4xlarge did not result in improvements in performance. For this reason, m4.2xlarge remains the recommended instance type for the extra-large load. The CPU utilisation was at ~ 40% on m4.2xlarge.



NFS node test results

In previous tests (where we benchmarked different application and database node configurations), we used **m5.4xlarge** for the NFS node (NFS protocol v3). During each of those tests, NFS node CPU remained highly underutilized at under 18%. We ran further tests to see if we could downgrade the NFS server (and, by extension, find more cost-effective recommendations). Results showed identical git throughput, using the downsized m5.xlarge NFS node. This led to our low-cost recommendation.

Component	Recommendation
Application nodes	m5.4xlarge x 3
Database node	m5.xlarge
NFS node	m5.xlarge

As mentioned, this recommendation costs \$3.044 per hour but offers lower fault tolerance.

Based on other test results, we recommend that, for the NFS node, use at least **m5.xlarge** with **IOPs higher than 1500**.

Benchmarks for the extra-large tests all used **m5.2xlarge** for the NFS instance. During each of those tests, the NFS node CPU remained highly underutilized at 25%. We ran further tests to see if we could downgrade the NFS server (and, by extension, find more cost-effective recommendations). Results showed identical git throughput, using the downsized **m5.xlarge** NFS node with CPU utilisation at 60%.

This led to our low-cost recommendation.

Component	Recommendation
Application nodes	m5.8xlarge x 4

Database node	m5.2xlarge
NFS node	m5.xlarge

Disk I/O

Disk I/O performance is often a limiting factor, so we also paid attention to disk utilization. Our tests revealed the disk specifications we used for the NFS node were appropriate to our traffic:

- 900GB [General Purpose SSD \(gp2\)](#) for storage
- IOPS:
 - Baseline of 2700 IOPS
 - Burstable to 3,000 IOPS.

As mentioned, we [initialized](#) this volume at the start of each test.

 Please be aware this information is only a guideline, as IOP requirements will depend on usage patterns.

The table below shows the I/O impact of our tests on the NFS nodes disk:

Metric	Value
Total throughput (Read + Write throughput)	1,250 IOPS
Read throughput	700 IOPS
Write throughput	550 IOPS
Read bandwidth	100 MB/s
Write bandwidth	10 MB/s
Average queue length	1.3
Average read latency	1.5 ms/op
Average write latency	0.6 ms/op
Disk utilization	45%

Disk I/O performance is often a limiting factor, so we also paid attention to disk utilization. Our tests revealed the disk specifications we used for the NFS node were appropriate to our traffic:

- 1800GB [General Purpose SSD \(gp2\)](#) for storage
- IOPS: baseline of 4500 IOPS

As mentioned, we [initialized](#) this volume at the start of each test.

 Please be aware this information is only a guideline, as IOP requirements will depend on usage patterns.

Metric	Value
Total throughput (Read + Write throughput)	9,00 IOPS
Read throughput	2,700 IOPS
Write throughput	IOPS

Read bandwidth	113 MB/s
Write bandwidth	15 MB/s
Average queue length	3.5
Average read latency	1.0 ms/op
Average write latency	0.70 ms/op
Disk utilization	80 %

Although the average disk utilisation is high at 80%, the read and write latency was low at < 1ms/op. It is recommended that the NFS server disk to have 4500 IOPs or more to ensure that it does not become the bottleneck.

Bitbucket Data Center and Server feature comparison

If you manage your own Bitbucket site (it's not hosted by Atlassian), you'll have either a **Bitbucket Server** or **Bitbucket Data Center** license.

Your Bitbucket license determines which features and infrastructure choices are available.

We want all teams to get the most out of Bitbucket, so all the core features are available for everyone - including projects, repositories, pull requests and workflows.

Feature comparison

Here's a summary of what you get with each license.

Code and collaboration	Server license	Data Center license
Collaboration via pull requests Review and discuss your code with your team before merging changes. Learn more about collaboration via pull requests Draft multiple comments on files and code during a review process. Learn more about the new code review workflow	✓	✓
Branch permissions Control what users can do on a single branch, branch type, or branch pattern within a repository or project. Learn more about branch permissions	✓	✓
Flexible workflows Use centralized, forking, gitflow or forking workflows. Learn more about flexible workflows	✓	✓
Jira Software integration Connect Jira and Bitbucket to automatically link issues and track progress simultaneously across both platforms. Learn more about integrating with Jira Software	✓	✓
APIs and 3rd party integrations Use the Bitbucket API and 3rd party integrations to automate simple tasks, embed data into your own site, and customize your workflow. Learn more about APIs and 3rd party integrations	✓	✓
CI/CD integrations Seamlessly integrate Bitbucket with the worlds leading CI/CD vendors to streamline the path to production. Learn more about our CI/CD integrations	✓	✓ 7.4+
Code insights Get reports, annotations, and metrics to help you and your team improve code quality in pull requests throughout the code review process. Learn more about code insights	✓	✓
Git LFS (Large File Storage) Store large files without the need for an external object store. Learn more about Git LFS	✓	✓

High availability and performance at scale		
Clustering Run Bitbucket on multiple nodes for high availability. Learn more about clustering		
Smart mirroring Improve Git clone speeds for distributed teams working with large repositories. Learn more about Smart Mirroring		
Content Delivery Network (CDN) support Improve geo-performance for distributed teams. Learn more about CDN		 6.8+
Disaster recovery Keep your teams online and source code data available in the event that your primary system becomes unavailable. Learn more about disaster recovery		 6.8+
Elasticsearch Connect Bitbucket to a remote Elasticsearch instance for improved scalability (required for Data Center sites). Learn more about Elasticsearch		
Security and compliance		
(Enforced) merge checks Stop pull requests from being merged until they meet the requirements that you've set. Learn more about merge checks		
Advanced auditing Get more insight into what is happening in your Bitbucket instance. Advanced auditing gives you the ability to track and log actions in your instance developing a security-relevant chronological record that can be exported and stored in third-party monitoring tools. Learn more about advanced auditing		 7.0+
User management		
External user directories Store users in Active Directory, Crowd, Jira or another LDAP directory. Learn more about external user directories		
Just-in-time provisioning Just-in-time user provisioning (JIT provisioning) allows users to be created and updated automatically when they log in through SAML SSO or OpenID Connect (OIDC) SSO to Atlassian Data Center applications such as Jira, Confluence, or Bitbucket. Learn more about just-in-time provisioning		 7.5+
SAML single sign-on Use a SAML identity provider for authentication and single sign-on. Learn more about SAML single sign-on		
OpenID Connect single sign-on Use OpenID Connect for authentication and single sign-on. Learn more about OpenID Connect single sign-on		
Infrastructure and Control		

<p>App diagnostics</p> <p>Get an overview of the health of your site, including potential performance issues relating to third-party apps. Learn more about app diagnostics</p>	✓	✓
<p>Integrity tests for zero-downtime backup</p> <p>Find and resolve any inconsistencies between the database and home directory, for example after restoring a backup. Learn more about Integrity tests for zero-downtime backup</p>	✗	✓
<p>Rate limiting</p> <p>Control how many external REST API requests users and automations can make. Learn more about rate limiting</p>	✗	✓ 6.6
<p>Rolling upgrades</p> <p>Upgrade to the latest bug fix update of the same feature release with no downtime. Learn more</p>	✗	✓ 7.9
Deployment options		
<p>Your own hardware</p> <p>Run Bitbucket on your own physical servers, virtualized servers, or in the data center of your choice.</p>	✓	✓
<p>AWS Quick Start and Cloud Formation Templates</p> <p>Use our Cloud Formation Templates to deploy Bitbucket on AWS. Learn more about AWS Quick Start and Cloud Formation Templates</p>	✗	✓
<p>Azure Quick Start Templates</p> <p>Use our template to deploy Bitbucket on Azure. Learn more about Azure Quick Start Templates</p>	✗	✓

FAQ

Related pages

- [Bitbucket Data Center FAQ](#)
- [Bitbucket Server Knowledge Base Home](#)
- [Support policies](#)

Child pages

- [Bitbucket rebrand FAQ](#)
- [How do I change the external database password](#)
- [Set the home directory](#)
- [Raising a request with Atlassian Support](#)
- [Support policies](#)
- [Contribute to our documentation](#)
- [Change data collection settings](#)

On this page

- [The Bitbucket Server product](#)
 - [Q: Why did you create a new product for Git repository management? Couldn't you build this into Fisheye?](#)
 - [Q: What about Git repository management in Fisheye and Crucible?](#)
 - [Q: Does Fisheye require Bitbucket Server? Does Bitbucket Server require Fisheye? Can they be used together?](#)
 - [Q: Will Bitbucket Server be available for Atlassian Cloud?](#)
 - [Q: What is the difference between Bitbucket Server and Bitbucket Data Center?](#)
- [Repositories](#)
 - [Q: Does Bitbucket Server support Mercurial \(Hg\)? What about other version control systems?](#)
- [Integration](#)
 - [Q: Does Bitbucket Server work with Jira Software? If so, what version of Jira Software do I need to run Bitbucket Server?](#)
 - [Q: Will Bitbucket Server integrate with any other Atlassian Tools? Crowd? Bitbucket? Sourcetree?](#)
 - [Licensing](#)
 - [Q: Does my Bitbucket Server license have to match the number of users in my external directory \(LDAP, Active Directory, Crowd or Jira Software\)?](#)
 - [Q: The number of users in my instance has exceeded my license count. Will Bitbucket Server still work properly?](#)
- [Data recovery and backups](#)
 - [Q: Can I use the Zero Downtime Backup method for versions prior to 4.8?](#)
 - [Q: I forgot the user/password for my old database schema. How will I perform the restore using the backup client? How does it work?](#)
 - [Q: What is the difference between the parameter pairs "bitbucket.user & bitbucket.password" and "jdbc.user & jdbc.password"?](#)
 - [Q: I backed up Bitbucket Server of a particular version using the backup client. Can I restore Bitbucket Server to a newer release version?](#)
- [Troubleshooting](#)
 - [Q: I'm getting a "broken pipe" error when pushing my commits.](#)

The Bitbucket Server product

Q: Why did you create a new product for Git repository management? Couldn't you build this into Fisheye?

A: In Fisheye 2.7 we added basic capabilities to host and manage Git repositories within Fisheye. However, as we were planning future releases, we realized that the architecture of Fisheye, built to index, browse and search across various SCMs, was not adequate for a DVCS repository management tool.

Therefore we have made the decision to build a new product, with a clear focus: hosting and managing Git repositories. Instead of a "Jack of all trades", we will have two products that are focused on 2 very different tasks:

1. Bitbucket Server Host, manage and collaborate on Git repositories, and
2. Fisheye Track, search and browse Subversion, Perforce, Git, Mercurial and CVS repositories in one place.

Q: What about Git repository management in Fisheye and Crucible?

A: *Internally managed* Git repositories were deprecated in Fisheye and Crucible 2.8, and support for these was removed for the Fisheye and Crucible 3.2 releases. We encourage those interested in Git repository management to check out Bitbucket Server.

Fisheye and Crucible will continue to deliver new features and enhancements to help users browse, search, review and visualize across different Version Control Systems *including Git*, Subversion, Mercurial, Perforce and CVS.

Q: Does Fisheye require Bitbucket Server? Does Bitbucket Server require Fisheye? Can they be used together?

A: Fisheye and Bitbucket Server are two separate standalone products that do not require each other.

If you are using multiple source code management systems (SCM) at your organization it makes sense to use both Fisheye and Bitbucket Server. While you are managing your Git repositories with Bitbucket Server, you can use Fisheye to browse, search and reference code from other SCMs including Subversion.

Also, if you are using Git, Bitbucket Server will provide your Git repository management, and Fisheye will be a central place to keep track of changes and search for code across your repositories.

Q: Will Bitbucket Server be available for Atlassian Cloud?

A: Bitbucket Server will not be available in Atlassian Cloud. If you are looking for a distributed version control solution to use with Atlassian Cloud, we recommend using [Bitbucket](#), our cloud-based Git and Mercurial source code hosting solution.

Q: What is the difference between Bitbucket Server and Bitbucket Data Center?

Bitbucket Server is a single instance of Bitbucket Server running on a single machine. It can only handle as much load as a single machine is capable of handling before performance degrades, and if the machine goes down for any reason (for example, hardware failure, network fault, or planned maintenance), then Bitbucket Server is unavailable to users for the duration of the downtime.

Bitbucket Data Center, on the other hand, looks like a single instance of Bitbucket Server to users, but under the hood consists of a cluster of multiple machines ("cluster nodes") each running the Bitbucket Server web application, behind a load balancer. This provides important benefits over Bitbucket Server:

- **Performance at scale:** A cluster of many machines running Bitbucket Server can handle more load than a single machine.
- **High availability:** If one cluster node goes down, then the remaining cluster node(s) can continue servicing requests so users should see little or no loss of availability.
- **Instant scalability:** You can rapidly provision extra capacity without downtime.

For more information see [Bitbucket Data Center](#) and the [Bitbucket Data Center FAQ](#).

Repositories

Q: Does Bitbucket Server support Mercurial (Hg)? What about other version control systems?

A: Currently Bitbucket Server does not support Mercurial. We will be gauging demand for Mercurial support as we move forward - [BSERV-2469](#) - Include Mercurial (Hg) support CLOSED

Integration

Q: Does Bitbucket Server work with Jira Software? If so, what version of Jira Software do I need to run Bitbucket Server?

A: Bitbucket Server works with Jira 4.3+. However, you will require the latest version of the Jira/Fisheye plugin to view commits in Jira Software. See our documentation on [Jira Software integration](#).

Q: Will Bitbucket Server integrate with any other Atlassian Tools? Crowd? Bitbucket? Sourcetree?

A: Bitbucket Server currently integrates with Jira Software, Sourcetree DVCS Mac client and Crowd user management solution. You can also connect to Bitbucket Server via Bamboo to run your builds and deployments and we are planning even tighter integrations in the future.

Licensing

Q: Does my Bitbucket Server license have to match the number of users in my external directory (LDAP, Active Directory, Crowd or Jira Software)?

A: No. You can control which users in your external directory have access to Bitbucket Server, so that the license limit is not exceeded. A user is by [definition](#) any account that has permission to log into the Bitbucket Server application. If you synchronize Bitbucket Server with an external user directory, you can grant access to Bitbucket Server to a subset of users, so as to stay below your license limit. The [Global permissions](#) page explains in detail how to manage login rights for users and groups in Bitbucket Server.

Q: The number of users in my instance has exceeded my license count. Will Bitbucket Server still work properly?

A: As stated in the [Global permissions](#) document, any user assigned "Bitbucket Server User" permission or higher, granted to the individual or via a group, will count towards the license limit. Bitbucket Server will not allow you to grant the "Bitbucket Server User" permission if this will exceed the license limit while manually adding users using Bitbucket Server UI. If you happen to exceed the license limit by connecting your Bitbucket Server instance to a User Directory that contains more users than your license allows, Bitbucket Server gives you a 24 hour grace period to solve the problem. A banner displaying the remaining grace period will be shown to all users to raise awareness of the issue. If the license is still exceeded after the grace period has expired, Bitbucket Server will display a banner with the content below:

 You have more users than your license allows.

Users will not be able to push commits to repositories until you restrict the number of active users to match your license or you upgrade your current license.

You don't have to remove users from the database in order to reduce your license count, instead, you can revoke access to the "Bitbucket Server User" permission through individual or group assignments.

Data recovery and backups

Q: Can I use the Zero Downtime Backup method for versions prior to 4.8?

A: No. [Zero Downtime Backup](#) can only be used with Bitbucket 4.8 or later. See [Data recovery and backups](#) to read more about the various backup options available.

Q: Can I restore the .tar file created by the backup client into a database that is different from my original one (i.e. Oracle -> MySQL, etc.)?

A: Yes you can, as long as you specify all the `jdbc` parameters (`jdbc.override`, `jdbc.driver`, etc.) when running the restore. Please read [Restoring Bitbucket Server into a newly created DB](#) for more details.

Q: I forgot the user/password for my old database schema. How will I perform the restore using the backup client? How does it work?

A: As described in [Restoring Bitbucket Server into a newly created DB](#), the restore client will only restore into an empty home directory and an **empty database**. The new database should be configured following the instructions in [Connect Bitbucket to an external database](#) and its sub-page that corresponds to your database type. If you want to use a different type of database or a different user/password, you just need to specify all the `jDBC` parameters (`jDBC.override`, `jDBC.driver`, etc) when running the restore.

Q: What is the difference between the parameter pairs "bitbucket.user & bitbucket.password" and "jDBC.user & jDBC.password"?

A: `bitbucket.user` and `bitbucket.password` hold the credentials for a Bitbucket Server sys admin user. They are only used during the **backup** procedure so that the backup client can lock Bitbucket Server and instruct Bitbucket Server to perform the database-agnostic backup. The backup client does not need database credentials because the Bitbucket Server system performs the database backup.

`jDBC.user` and `jDBC.password` are only used during the **restore** procedure when `jDBC.override` is set to `true`. They are used to connect to the newly-installed database.

Q: I backed up Bitbucket Server of a particular version using the backup client. Can I restore Bitbucket Server to a newer release version?

A: No. You need to use the same Bitbucket Server binary as the one originally used to back up your instance. Note that using an older Bitbucket Server binary will result in an error downgrades are not possible. See [Bitbucket Server Backup Client](#) for details on the backup/restore procedure.

Once you have restored Bitbucket Server, you can upgrade to a newer version of Bitbucket Server following the instructions in the [Bitbucket Server upgrade guide](#).

Troubleshooting

Q: I'm getting a "broken pipe" error when pushing my commits.

A: This error occurs when the amount of data you're trying to push in one go exceeds Git's http post buffer. Just run the following command to increase it to 500MB.

```
git config http.postBuffer 524288000
```

See [Git push fails with 'fatal: The remote end hung up unexpectedly'](#).

Bitbucket rebrand FAQ

We have unified our Git products under the Bitbucket name. With Bitbucket, now you have a range of options that can be adopted by teams of all sizes and requirements: [Bitbucket Cloud](#) (previously known as Bitbucket), [Bitbucket Server](#) (previously known as Stash) and [Bitbucket Data Center](#) (previously known as Stash Data Center). We hope the following answers some of the more common questions this might raise.

- [Why did we rebrand Stash to Bitbucket Server?](#)
- [What is the impact on my plug-ins? Do I need to reinstall them?](#)
- [How do I raise support issues?](#)
- [Are there any feature differences between deployment options?](#)
- [Why aren't the Server and Cloud features the same?](#)
- [Can I export my Bitbucket repositories between services?](#)
- [Will Bitbucket Server support Mercurial?](#)
- [Where do I go if I have any more questions?](#)
- [Can I develop Bitbucket Connect add-ons for Bitbucket Server?](#)
- [Can I develop P2 add-ons for Bitbucket Cloud?](#)
- [Why is there no free pricing tier for Bitbucket Server?](#)
- [Where can I find more information about Mirroring and Large File Support on Bitbucket Server?](#)
- [Where can I find more information about Projects and Build Status on Bitbucket Cloud?](#)

Why did we rebrand Stash to Bitbucket Server?

To make it easier for you to find a collaborative code management solution that best meets your needs, we have unified our Git products under the Bitbucket name.

- **Bitbucket Server** (previously known as Stash) fits into any enterprise environment because it can be deployed within existing infrastructure or on hosted infrastructure providers such as Amazon Web Services.
- **Bitbucket Cloud** (previously known as Bitbucket) is a fully managed, multi-tenant service used by organizations that do not want to deal with the overhead of managing their own infrastructure.
- **Bitbucket Data Center** (previously known at Stash Data Center) is the deployment option that provides high availability and massive scale for source code management systems. It is especially beneficial for large organizations that have thousands of developers, hundreds of continuous integration servers, and /or require sustained uptime regardless of load.

What is the impact on my plug-ins? Do I need to reinstall them?

Starting September 22nd, 2015, the new brand will be rolled out with the release of Bitbucket Server 4.0 including major API changes from Stash. The renaming of Stash to Bitbucket requires any custom add-ons to be changed, so it is important that you update your own custom add-ons and check for updated marketplace listed add-ons *before* upgrading. The new brand will only affect Bitbucket Server add-ons and will not have an impact on Bitbucket Cloud add-ons.

Immediate action to take:

- Read the [Bitbucket Server 4.0 release notes](#) to understand the add-ons impact
- Assess what you need to do to upgrade
- [Upgrade](#) to Bitbucket Server 4.0

How do I raise support issues?

Go to <https://support.atlassian.com>, select **Dev Tools**, and submit a support request. We'll get you on track.

Are there any feature differences between deployment options?

The high level benefits of Bitbucket Server and Bitbucket Cloud are the same. They both offer fine-grained permissions, pull request workflow, Git repository sharing and management, and a robust set of extension and integration APIs. However, there are some features that are not the same due to the history of the two deployment options. In the future, Bitbucket's deployment options will grow closer together so you have a familiar code management platform with closer feature, function, and UI parity no matter how you access Bitbucket.

There are a few key exceptions:

- **Mercurial support** - We will continue to support and build for Mercurial repository hosting in Bitbucket Cloud. We do not expect to add support for Mercurial in Bitbucket Server at this time.
- **Issue tracking and wiki** - Bitbucket Cloud offers basic wiki support and issue tracking. With the availability of Confluence for documentation, and JiraSoftware for integrated software development planning and tracking, it is not our intention to build the same into Bitbucket Server, however we will continue to improve upon the integration with these products.
- **Snippets** - There are no immediate plans to add support for snippets to Bitbucket Server. A [3rd-party add-on](#) is available for those wishing to manage code snippets in Bitbucket Server.

Why aren't the Server and Cloud features the same?

Bitbucket Server (previously known as Stash) was first released in May 2012 as an enterprise-grade, high performance, self-managed Git repository hosting and collaboration tool. It was built from the ground-up for self-managed deployment. Bitbucket Cloud (previously known as Bitbucket) was acquired by Atlassian in 2010 and was purpose-built for the multi-tenant public cloud. As a result, Bitbucket Cloud and Bitbucket Server had different technical architectures with different feature roadmaps. Over the past 2 years, much progress had been made in bringing conceptual and functional parity to the products, and work will continue to improve upon this.

Can I export my Bitbucket repositories between services?

Yes. Because Git repositories are distributed you can simply upload your repositories into either service. Currently there is no way to migrate other data such as comments and pull requests, but this is something we plan to support at some point in the future.

Will Bitbucket Server support Mercurial?

Mercurial support has not been scheduled and is unlikely to become available in the near future for Bitbucket Server but will continue to be supported in Bitbucket Cloud.

Where do I go if I have any more questions?

Go to [Atlassian Answers](#) if you have more questions.

Can I develop Bitbucket Connect add-ons for Bitbucket Server?

At this time we're unlikely to incorporate support for Atlassian Connect integration framework in Bitbucket Server. We will continue to support the mature and complete Plugins 2 framework - it continues to ship with all of Atlassian's Server products. For more information, check out [Atlassian Connect for JiraSoftware Server and Confluence Server](#).

Can I develop P2 add-ons for Bitbucket Cloud?

There are no plans to support P2 plugins for Bitbucket Cloud. We've built the Atlassian Connect platform for developing add-ons for our Cloud products.

Why is there no free pricing tier for Bitbucket Server?

In absence of a free plan, Atlassian provides a starter license for Bitbucket Server which is \$10 for 10 users. This is a perpetual license and comes with 12 months of maintenance and support. All proceeds go to the charity, [Room to Read](#), where our Starter Licenses have raised millions of dollars.

Where can I find more information about Mirroring and Large File Support on Bitbucket Server?

All current details are shared in this [blog post](#). Follow our Bitbucket blog for more details in the near future.

Where can I find more information about Projects and Build Status on Bitbucket Cloud?

We don't have much to share right now other than the information we have already shared in the [blog post](#). Follow our Bitbucket blog for more details in the near future.

How do I change the external database password

You can change the password that Bitbucket Data Center and Server uses to connect to an external database, however you don't do this from the Bitbucket Administration area you must follow the procedure described below.

Related pages:

- [Connect Bitbucket to an external database](#)

To change the password that Bitbucket uses when connecting to an external database:

1. Stop Bitbucket. See [Start and stop Bitbucket](#).
2. Get your database administrator to change the password on your database.
3. Go to your [Set the home directory](#).

Edit the `bitbucket.properties` file to change the line that looks like:

```
jdbc.password=MY_PASSWORD
```

replacing `MY_PASSWORD` with your new database password.

4. Restart Bitbucket. See [Start and stop Bitbucket](#).

Set the home directory

The home directory is created automatically by the Bitbucket Data Center and Server installer see the [in stallation guide](#) to install and start using it.

The information on this page only applies if you are [manually installing](#) or upgrading Bitbucket.

On this page:

- [What is the home directory?](#)
- [What does the home directory contain?](#)
- [Setting the home directory](#)
- [Securing the home directory](#)
- [About the repositories](#)

What is the home directory?

The home directory is where your Bitbucket data is stored. The home directory location is defined either by the `BITBUCKET_HOME` environment variable, or in the `BITBUCKET_HOME` line of:

- Windows: `<Bitbucket installation directory>\bin\set-bitbucket-home.bat`
- Linux and Mac: `<Bitbucket installation directory>/bin/set-bitbucket-home.sh`

⚠ Bitbucket 4.0 and later versions don't allow the home directory to be the same directory as, or a subdirectory of, the `<Bitbucket installation directory>`. The home directory, as defined by the `BITBUCKET_HOME` variable, must be in a separate location otherwise Bitbucket will fail on startup. And by the way, you'll need separate home directories if you want to run multiple instances of Bitbucket (when these are not nodes for a Bitbucket Data Center).

⚠ Where possible, you should choose a location for your home directory that will *never* need to be moved. Some home contents are location-sensitive, so moving the home directory may corrupt them. Bitbucket attempts to update contents when it detects that the home directory has moved, but the safest approach is to avoid the issue altogether by leaving the home directory in the same location.

⚠ Where possible, do not configure Bitbucket instances to use a UNC path. This can cause issues forking and merging pull requests. Instead, [map a network drive](#) or use the `mklink` command to create a symbolic link to the networked location and update `BITBUCKET_HOME` to reference the new drive letter/path.

What does the home directory contain?

Your home directory contains the following directories and files:

Path	Description
<code>caches</code>	Contains cache and index files. It should be safe for these files to be deleted between application restarts; however, these files must not be modified or deleted while Bitbucket is running.
<code>shared /config</code>	Contains application configuration.
<code>shared /data</code>	Contains the Git repositories, project avatars, and the embedded HSQL database if an external database is not configured.
<code>export</code>	Contains dump files produced during database migrations.
<code>lib</code>	Can contain third-party jars such as the MySQL JDBC driver.
<code>lib /native</code>	Can contain <i>native libraries</i> , such as Tomcat's APR-based native library.
<code>log</code>	Contains log files for Bitbucket.
<code>shared /plugins</code>	Contains plugin related data (such as externally uploaded plugins) for Bitbucket.

tmp	Contains temporary files created by the system. Its contents can safely be deleted when Bitbucket Data Center and Server is <i>not running</i> .
shared /bitbucket. properties	Allows configuring various aspects of how Bitbucket behaves, such as its database connection pool size and the location of the Git binary to use. This file will be created automatically during a database migration. It can be created manually otherwise. See Configuration properties for more information.

Setting the home directory

Note that the home directory is created automatically by the Bitbucket installer .

Only when you are [installing from an archive file](#) will you need to set the value of `BITBUCKET_HOME` yourself, as described in this section.

Create your home directory (without spaces in the name), and then tell Bitbucket where you created it by editing the `<Bitbucket installation directory>/bin/set-bitbucket-home.sh` file. Uncomment the `BITBUCKET_HOME` line and add the absolute path to your home directory. Here's an example of what that could look like when you're done:

```
#
if [ "${STASH_HOME}" = "x" ]; then
    export STASH_HOME="/home/username/stash-home"
fi
```

Create your home directory, and then tell Bitbucket where you created it by setting a `BITBUCKET_HOME` environment variable, as follows.

For Windows 7:

1. Go to **Start**, search for "sys env" and choose **Edit the system environment variables**.
2. Click **Environment Variables**, and then **New** under 'System variables'.
3. Enter "BITBUCKET_HOME" as the **Variable name**, and the absolute path to your home directory as the **Variable value**. Don't use a trailing backslash.

There are a few things to know about setting up the home directory on Windows that will make life easier:

- You *should not* locate your home directory inside the `<Bitbucket installation directory>` - they should be entirely separate locations. If you do put the home directory in the `<Bitbucket installation directory>` it will be overwritten, and lost, when Bitbucket gets upgraded. And, by the way, you can't use the same home directory for multiple instances of Bitbucket.
- Keep the path length to the home directory as short as possible. See [Bitbucket Server is unable to create the merge diff for pull requests](#) for an explanation.
- Don't use spaces in the path to the home directory.

Securing the home directory

The internal database files, the migration dump files and `bitbucket.properties` all contain information that may be considered secret (server settings, salted and hashed user passwords, database passwords, etc).

For production use, we strongly recommend that you secure this directory against unauthorized access.

We recommend the following precautions:

- Assign a separate restricted user account on the machine for running Bitbucket (not a root/administrator user)
 - If you wish to run Bitbucket on port 80, use a separate http front end as described in [Integrate Bitbucket with Apache HTTP Server](#) (do not run as root/Administrator if security of the home directory is important to you)
- Ensure that only the user running Bitbucket can access the home directory, and that this user has read, write and execute permissions, by setting file system permissions appropriately for your operating system.

About the repositories

As noted above, `data` contains the Git repositories being managed by Bitbucket, where "managed by Bitbucket" are the operative words. The repositories are for Bitbucket to interact with, and they are configured and managed accordingly. They are *not* a mechanism for configuring Bitbucket behavior. We *strongly* recommend that customers never modify them, nor interact with them directly. They are *intentionally* structured in a way which does not lend itself well to direct interaction.

Being Git repositories, there are certainly standard aspects to how the repositories on disk are stored and how they function. However, the exact way they are configured *can and does* change between Bitbucket releases. Bitbucket makes *no effort* to preserve unexpected configuration changes which have been applied by customers, and such changes may cause failures at runtime or during upgrades. If there is an aspect of Bitbucket's behavior you wish to configure, please open a feature request on jira.atlassian.com rather than trying to modify the repositories directly.

 Repositories are *location sensitive*. Moving your home directory will result in the system being locked (briefly) on startup while Bitbucket updates the repositories on disk. Assuming the updates are applied successfully, the system will then unlock itself for normal usage.

Where possible, please choose a home location which will not need to be changed later.

Raising a request with Atlassian Support

If you encounter any problems when setting up or using Bitbucket Data Center and Server, please let us know we're here to help!

If you're on a starter license, we suggest you check out the following resources:

- the [Atlassian Community](#) site.
- the [Bitbucket Knowledge Base](#).

If you're interesting in upgrading to get private technical support, have a read [here](#) to get an idea of what needs to be done.

If you've found a bug, or want to request a feature or improvement, raise a ticket in the Bitbucket project of our [public issue tracker](#). Try searching for similar issues - voting for an existing issue is quicker, and avoids duplicates.

If you still need assistance, please raise a support request, either from within Bitbucket or on the Atlassian Support site, as described in the following sections.

Providing as much information as possible about your installation with your initial request will help our Support Engineers to give you a faster and more complete response.

On this page:

- [Raising a Support Request in Bitbucket](#)
- [Raising a support request yourself at Atlassian Support](#)
- [Information you should provide](#)

Raising a Support Request in Bitbucket

To use this method a [mail server](#) that supports large zip file attachments must be configured.

To raise a support request in Bitbucket:

1. Log in as a System Administrator.
2. Go to  > **Troubleshooting and support tools** > **Get help**.
3. Select **Contact Technical Support or Report a Bug**.
4. Create your support request. You'll need to include:
 - **Contact email** This defaults to the email address of the logged-in user. It'll be used to find your support account or, if no account is found, to create a new account. We'll send all notifications and updates about your problem to this address.
 - **Summary** Enter a short summary of the problem.
 - **Description** Include as much information as possible about the problem, such as steps to replicate the problem; messages on the console or in the [log](#); and [profiling logs](#) if you're having a performance issue. See also [Information you should provide](#) below.
 - **Severity** Choose an option to indicate how many users are affected.
 - **Attach support zip** When this is selected, a zip file of your logs and configuration files will be attached to your support request. This can speed up troubleshooting the problem and it should be included if possible. See [Create a Support Zip](#) for info on what's included.
5. Select **Send**.

After sending your support request you'll receive an email with the information you entered. You'll also receive email updates about progress on the problem.

To see the status of your request visit the [Atlassian Support System](#).

Raising a support request yourself at Atlassian Support

To raise a support request at Atlassian Support:

1. Log in as a System Administrator.

2. Go to  > **Troubleshooting and support tools** > **Create support zip**
3. If necessary, select **Customize zip** to choose what is included. For help, see [Create a support zip](#).
4. Select **Create zip**.
5. Select **Download zip** to save it to the [home directory](#) of your Bitbucket instance. For example, <Bitbucket home directory>\shared\export\Bitbucket_XXX.zip.
6. Go to [Atlassian Support](#) and create a support request with the downloaded support zip.

When creating your support request, provide as much information as possible. Include steps to replicate the problem; messages on the console or in the [log](#); and [profiling logs](#) if you're having a performance issue. See also [Information you should provide](#) below.

Information you should provide

In addition to the logs and configuration information that you can include in the Support Request zip file, the following information can help to give you a faster response:

Environment details

- Bitbucket version
- Java version (for example OpenJDK 1.7.0 JRE)
- Git and Perl versions
- Operating system (for example, Windows 7, macOS 10.6.8)
- Database type (for example, MySQL) and version
- Browsers and versions
- Network topology - is Bitbucket running behind a reverse proxy? Is that secured using HTTPS (SSL)?

Configuration

- Java settings, including JVM_MINIMUM_MEMORY, JVM_MAXIMUM_MEMORY

Logs

You may need to adjust the logging level, or enable profiling in Bitbucket, in order to get more detailed logs. See [Enable debug logging](#).

- Debug logs can be found in <Bitbucket home directory>/log.
- Profiling logs can help with analyzing performance issues and can be found in <Bitbucket home directory>/log.

Performance factors

- Number of concurrent Git clones
- Number of users
- The size of the .git directory
- CPU spec, number of cores, whether hyperthreading is enabled
- RAM and cache sizes

Integrations

- Other Atlassian applications (and their versions)
- Which build servers are integrated with Bitbucket, if any?
- Are Application Links configured?

Support policies

Welcome to the support policies index page. Here, you'll find information about how Atlassian Support can help you and how to get in touch with our helpful support engineers. Please choose the relevant page below to find out more.

- [Bug fixing policy](#)
- [New features policy](#)
- [Security Bugfix Policy](#)
- [Find your Support Entitlement Number \(SEN\)](#)

To request support from Atlassian, please raise a support issue in our online support system. To do this, visit support.atlassian.com, log in (creating an account if need be) and create an issue. Our friendly support engineers will get right back to you with an answer.

Bug fixing policy

Summary

- Our Support team will help with workarounds and bug reporting
- We'll generally fix critical bugs in the next maintenance release
- We schedule non-critical bugs according to a variety of considerations

[Report a bug](#)

Developing an app (add-on) for an Atlassian product or using one of our APIs? Report any related bugs in our [Ecosystem Jira](#).

On this page:

- [Summary](#)
- [Bug fixes for server products](#)
- [Bug reports](#)
- [Search existing bug reports](#)
- [How we approach bug fixing](#)
 - [Severity 1 - Critical](#)
 - [Severity 2 - Major](#)
 - [Severity 3 - Minor](#)
- [About our bug fix workflow](#)
- [How to get access to bug fixes](#)
- [Release terminology for Data Center and server products](#)
 - [Long Term Support releases](#)

Important changes to our server and Data Center products

We've ended sales for new server licenses, and will end support for server on February 2, 2024. We're continuing our investment in Data Center with several key improvements. [Learn what this means for you](#)

Bug fixes for server products

We'll continue to provide bug fixes for server products until February 2, 2022 PT. After this, we'll only provide security bug fixes for critical vulnerabilities until the end of support date on February 2, 2024 PT. [Learn more about these changes](#)

Bug reports

Atlassian Support is eager and happy to help verify bugs we take pride in it! Create an issue in our [support system](#), providing as much information as you can about how to replicate the problem you're experiencing. We'll replicate the bug to verify, then lodge the report for you. We'll also try to construct workarounds if possible.

Search existing bug reports

Use our [public issue tracker](#) to search for existing bugs, add your report, and watch the ones that are important to you. When you watch an issue, we'll send you an e-mail notification when the issue's updated.

How we approach bug fixing

Bug fix releases are more frequent than feature releases, and target the most critical bugs affecting customers. The notation for a bug fix release is the final number in the version (the 1 in 6.0.1, for example).

We assess each bug based on the symptom severity (that is, when this bug causes symptoms, how severe are those symptoms). There are three levels of symptom severity.

Severity 1 - Critical

Your application is unavailable. Users aren't able to perform their job function, and no workarounds are available.

- login failure affecting all users
- all or most pages don't display
- out of memory errors cause application failure
- significant data loss
- node communication failures
- administration tools fail.

Severity 2 - Major

A feature is unavailable, application performance is significantly degraded, or users job functions are impaired.

- the application performs slowly and fails intermittently
- application is functional, but frequently used gadgets or macros don't work
- application links fail
- specific editing features fail
- or a Severity 1 (critical) issue where there is a viable workaround.

Severity 3 - Minor

The application or specific feature isn't working as expected, but there is a workaround available. Users experience is impacted, but their job function is not impaired.

- some searches fail
- sections of pages load slowly
- administrative features fail intermittently, but a workaround is available
- visual defects, that don't affect function
- minor translation or localization problems
- keyboard shortcuts not functioning as expected.

Assessing bugs using symptom severity makes sure that we prioritise the most impactful fixes. We give high priority to [security issues](#).

About our bug fix workflow

If you watch or mark a bug as affecting your team, its useful to understand how we review, prioritize, and resolve them in our public issue tracker jira.atlassian.com.

We prioritize issues using a metric called User Impact Score (UIS), which is individually calculated for every issue. It takes into account the number of affected users, the severity of the issue, recent interest, and the percentage of users affected per instance. The higher the UIS score, the more pervasive and severe the issue is.

We have also standardised our workflow statuses across Data Center and server products to make it easy for you to see where an issue is at. Heres the current workflow, and a description of each status.



Workflow status	Definition	Phase
Needs triage	This issue is waiting to be reviewed by a member of the Atlassian product team. Typically, only recently created issues are in this status. Our product teams review these issues regularly.	Review
Gathering impact	This issue has been reviewed, but needs more supporting information to gauge how pervasive the problem is.	Prioritization
Long term backlog	A fix for this issue is required, but planned for farther in the future. This is because its not as severe or pervasive as other issues.	
Short term backlog	A fix for this issue is required, and will be prioritised in the near future. This is because its more severe or pervasive than other issues.	
In progress	The development team is currently working on this issue.	Implementation
In review	A fix for this issue has been proposed and is being reviewed and quality-tested by the development team.	
Waiting for release	A fix for this issue has been implemented and is waiting to be shipped in a release.	
Closed	Work on this issue is complete. If its fixed, the resolution will be Fixed and the Fix Version field will indicate the product version that contains the fix. If no code changes were required, the resolution will be Duplicate', 'Won't fix', 'Handled by support', 'Timed out', or similar.	Closure

How to get access to bug fixes

To get access to bug fixes you will need to upgrade to a release that contains the fix.

Release terminology for Data Center and server products

- **Platform release**(example: 4.0) contains significant or breaking changes. For example changes or removal of existing APIs, significant changes to the user experience, or removal or a major feature.
- **Feature release**(example: 4.6) can contain new features, changes to existing features, changes to supported platforms (such as databases, operating systems, Git versions), or removal of features. These were previously referred to as 'major' releases by most products.

- **Bugfix release**(example: 4.6.2) can contain bug fixes and stability and performance improvements. Depending on the nature of the fixes they may introduce minor changes to existing features, but do not include new features or high-risk changes, so can be adopted quickly. We recommend regularly upgrading to the latest bugfix release for your current version. These were previously referred to as 'maintenance' releases by most products.

In addition to the three main release types, a feature release can also be designated a **Long Term Support release**(formerly known as an Enterprise release), which means it will receive bug fixes for a longer period of time than a standard feature release.

Long Term Support releases

Long Term Support releases (formerly known as Enterprise releases) are for Server and Data Center customers who prefer to allow more time to prepare for upgrades to new feature versions, but still need to receive critical bug fixes. If you only upgrade to a new feature version about once a year, a Long Term Support release may be a good fit for your organisation. For Jira Software and Confluence we will:

- Designate a feature release as a Long Term Support release,at least every 12 months.
- Backport critical security fixes, as outlined in our current security bug fix policy, and fixes relating to stability, data integrity or critical performance issues.
- Make bug fix releases available for the designated version until it reaches end of life.
- Provide a change log of all changes between one Long Term Support release and the next to make upgrading easier.

Not all bug fixes will be backported. We'll target the bugs and regressions that we deem most critical, focusing on stability, data integrity, or performance issues. There may also be some fixes that we choose not to backport due to risk, complexity or because the fix requires changes to an API, code used by third party apps (also known as add-ons), or infrastructure that we would usually reserve for a platform release.

For Jira Software Data Center customers, we'll endeavour to allow zero downtime upgrades between one Long Term Support release and the next Long Term Support release, but can't guarantee that down time will not be required, depending on the nature of the changes. The change log will indicate if zero downtime upgrade will be available.

In the example below, version 4.2 has been designated a Long Term Support release. The number of bug fix releases and timing illustrated below is just an example, your product's release cadence may differ.



i Long Term Support changes for server customers

If you have a server license, you'll only be eligible to upgrade to versions released prior to February 2, 2024 PT, when we officially [end support for our server product line](#).

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

New features policy

 This policy does not apply to bugs. See our [Server Bug Fix Policy](#) or [Cloud Bug Fix Policy](#) to learn about our approach to bug fixing.

How we choose what to implement

There are many factors that influence our product roadmaps and determine the features we implement. When making decisions about what to prioritize and work on, we combine your feedback and suggestions with insights from our support teams, product analytics, research findings, and more. This information, combined with our medium- and long-term product and platform vision, determines what we implement and its priority order.

How to track when features are implemented

Cloud products

We're continuously improving and updating our Cloud products. To see the latest changes, take a look at the [Atlassian Cloud release notes blog](#).

Data Center products

When a new feature or improvement is scheduled, we'll update the fix version on the relevant Jira issue to indicate the earliest product version that will include the change. This update often happens close to the product release date.

For a summary of changes, see the release notes for your product:

- [Jira Software](#)|[Jira Service Desk](#)|[Jira platform](#)|[Advanced Roadmaps for Jira](#)
- [Confluence](#)|[Questions for Confluence](#)|[Team Calendars for Confluence](#)
- [Bitbucket](#)|[Bamboo](#)|[Fisheye](#)|[Crucible](#)

Server products

We're simplifying our self-managed offerings and sharpening our focus to our cloud and Data Center products. This means we've discontinued new feature development in our server product line. [Learn more about these changes](#)

We'll still be offering bug fixes for server customers with active maintenance. For details, see our [Atlassian Data Center and Server bug fix policy](#).

Product roadmaps

We publish a [public roadmap](#) for Jira Cloud products, Confluence Cloud, Bitbucket Cloud, and our Cloud Platform. This lets you know what's coming soon and what we're thinking about for future updates.

The [Atlassian Cloud release notes blog](#) and [Bitbucket Cloud blog](#) may also contain information on upcoming changes.

We don't provide specific release dates for upcoming changes.

Feature and improvement suggestions

We encourage you to suggest improvements and new features for our products. You can create feature suggestions, or vote, watch, and comment on existing suggestions, at <https://jira.atlassian.com/>.

We get a large number of suggestions and feature requests. Your comments and votes on suggestions help us understand what you're passionate about and how you want our products to support you and your team. The most helpful information you can provide us when commenting on issues is how a particular suggestion would help you. If you describe your use-case to us, and how the suggested change would benefit you and your team, it lets us gain a much deeper understanding of the need behind the suggestion.

Suggestions often have an impact on what we work on, even if we ultimately choose not to implement a suggestion exactly as its described. Our ultimate goal is to understand what you and all of our customers need and to create products that meet those needs. Occasionally, thatll mean implementing a suggestion as described, but it usually means working to understand the need behind the suggestion and how we can meet that need for as many users as possible.

While we endeavor to update and respond to popular suggestions, the volume we receive means there will often be occasions when we cant provide an update or response. We dont provide any compensation or credit for feature suggestions that we implement.

Join the conversation on Atlassian Community

Our Product Managers regularly post articles about new features and changes to the [Atlassian Community](#). You can comment on these posts, ask questions, and discuss with our PMs and other Atlassian users.

Release terminology for Data Center and server products

- **Platform release**(example: 4.0) contains significant or breaking changes. For example changes or removal of existing APIs, significant changes to the user experience, or removal or a major feature.
- **Feature release**(example: 4.6) can contain new features, changes to existing features, changes to supported platforms (such as databases, operating systems, Git versions), or removal of features. These were previously referred to as 'major' releases by most products.
- **Bugfix release**(example: 4.6.2) can contain bug fixes and stability and performance improvements. Depending on the nature of the fixes they may introduce minor changes to existing features, but do not include new features or high-risk changes, so can be adopted quickly. We recommend regularly upgrading to the latest bugfix release for your current version. These were previously referred to as 'maintenance' releases by most products.

In addition to the three main release types, a feature release can also be designated a **Long Term Support release**(formerly known as an Enterprise release), which means it will receive bug fixes for a longer period of time than a standard feature release.

Long Term Support releases

Long Term Support releases (formerly known as Enterprise releases) are for Server and Data Center customers who prefer to allow more time to prepare for upgrades to new feature versions, but still need to receive critical bug fixes. If you only upgrade to a new feature version about once a year, a Long Term Support release may be a good fit for your organisation. For Jira Software and Confluence we will:

- Designate a feature release as a Long Term Support release,at least every 12 months.
- Backport critical security fixes, as outlined in our current security bug fix policy, and fixes relating to stability, data integrity or critical performance issues.
- Make bug fix releases available for the designated version until it reaches end of life.
- Provide a change log of all changes between one Long Term Support release and the next to make upgrading easier.

Not all bug fixes will be backported. We'll target the bugs and regressions that we deem most critical, focusing on stability, data integrity, or performance issues. There may also be some fixes that we choose not to backport due to risk, complexity or because the fix requires changes to an API, code used by third party apps (also known as add-ons), or infrastructure that we would usually reserve for a platform release.

For Jira Software Data Center customers, we'll endeavour to allow zero downtime upgrades between one Long Term Support release and the next Long Term Support release, but can't guarantee that down time will not be required, depending on the nature of the changes. The change log will indicate if zero downtime upgrade will be available.

In the example below, version 4.2 has been designated a Long Term Support release. The number of bug fix releases and timing illustrated below is just an example, your product's release cadence may differ.



i Long Term Support changes for server customers

If you have a server license, you'll only be eligible to upgrade to versions released prior to February 2, 2024 PT, when we officially [end support for our server product line](#).

Further reading

See [Atlassian Support Offerings](#) for more support-related information.

Security Bugfix Policy

See [Security @ Atlassian](#) for more information on our security bugfix policy.

Find your Support Entitlement Number (SEN)

Your Bitbucket Data Center and Server Support Entitlement Number (SEN) is required when raising a support request in our Support system: <http://support.atlassian.com>.

See [How to find your Support Entitlement Number \(SEN\)](#) in the Support space for more general information about how Atlassian Support uses this number.

The three ways of finding your SEN are described below.

On this page:

- [Method 1 Check the administration area](#)
- [Method 2 Check my.atlassian.com](#)
- [Method 3 Check your Atlassian Invoice](#)

Method 1 Check the administration area

To find your SEN in the administration area:

1. Select the  icon in the header.
2. Select **Licensing** in the left navigation panel (under 'Settings'). The SEN is shown as in the screenshot below:

Server ID 

License

License **Bitbucket (Server): Evaluation**

Licensed to **Atlassian**

Support Entitlement Number (SEN) 

Expires 

User limit **Unlimited**

3. Clustering **Not available with this license**

Method 2 Check my.atlassian.com

To find your SEN via my.atlassian.com:

1. Log into my.atlassian.com as the Account Holder or Technical Contact for your Bitbucket product.
2. The SEN will be shown, as per the screenshot below:

Bitbucket (Server): Evaluation

Atlassian

Server ID	[REDACTED]	License Key	AAABJw TaxVhIV d mCZj+e 9gEVpF DX
SEN ?	SEN-[REDACTED]		

Actions Buy | Dc

Method 3 Check your Atlassian Invoice

Your Support Entitlement Number (SEN) appears on the third page of your Atlassian Invoice.

Contribute to our documentation

Would you like to share your Bitbucket Data Center and Server hints, tips and techniques with us and with other users? We welcome your feedback.

Using the feedback form

On the bottom of every page in our documentation, we've added the ability to submit feedback on pages. If you have any of the following (or something else entirely!):

- Something new to be added, such as a new page, or a screenshot.
- An error or typo, or something that has become out of date.
- A request for an addition.
- An improvement to the existing page.
- A comment about how great something is, or a complaint about something.

Please feel free to use this function to help us to improve the quality of our documentation, and we'll do our best to get to it in a timely fashion.

Helpful suggestions when submitting using the feedback form

We've listed some ideas below that really help us when we read and triage your feedback.

- **We like knowing what version you're using, same as when logging a bugfix ticket:** Not all versions of the documentation are the same, and sometimes it's not clear which version a suggestion refers to.
- **If it's an error, copy and paste it or screenshot exactly the problem:** This isn't always necessary, but sometimes something that seems really obvious in your head, is not immediately clear to another person.
- Nobody is perfect so we love to fix spelling mistakes or other little oversights. No problem is too small to submit using the form!

A few small things to remember

Just a few reminders for when you have found something that is really annoying or important to you:

- **There are real people at the other end helping you:** We understand that when you find an error, it can be really frustrating, especially when it impacts your work or favorite project. Please remember there are actual people reading all of this and triaging the tickets.
- **Sometimes we can't get things done immediately:** We understand some changes are pretty small, and it might seem like things are moving pretty slowly. If that's the case, rest assured we're working as hard as we can! No issue is unimportant, but often resource has to be focused on errors, rather than improvements.
- **We can't incorporate everything:** Sometimes you may have a really cool idea for an improvement that can't be actioned no matter what we do. Here at Atlassian we have a lot of freedom, but there are style guides, theme restrictions and other limitations. For example, if you really want a scenario documented but it's a real doozy of an edge case, we have to weigh up whether it will help the majority of the users of the documentation, or whether we should document something that more people are experiencing.

On this page:

- [Using the feedback form](#)
 - [Helpful suggestions when submitting using the feedback form](#)
 - [A few small things to remember](#)

Change data collection settings

We are continuously working to make Bitbucket Data Center and Server better. Data about how you use Bitbucket helps us do that. We have updated our Privacy Policy so that we may collect usage data automatically unless you disable collection. The data we collect includes information about the systems on which your Bitbucket installation is operating, the features you use, and your use of common IT terminology within the product. For more details, see our [Privacy Policy](#), in particular the 'Analytics Information from Downloadable Products' section.

See also our [End User Agreement](#).

How to change data collection settings?

You can opt in to, or out of, data collection at any time. An admin can change the data collection settings by going to **Analytics** (under 'Settings') in the Bitbucket admin area.

How is data collected?

We use the Atlassian Analytics plugin to collect event data in Bitbucket. Analytics logs are stored locally and then periodically uploaded to a secure location.

Bitbucket Server and Data Center GDPR support guides

We've created this set of support guides to help you with any GDPR-specific obligations that you may have. The guides are designed to identify where personal data may be located within the product, and, where possible, assist with responding to data subject access requests, including in modifying, restricting or deleting data.

Where a workaround involves running SQL scripts directly on your database, we strongly recommend you do this first on a non-production, staging or test environment before running the scripts on your production database. We also strongly recommend you take a backup of your data before making any modifications via SQL scripts.

- [Automated individual decision-making, including profiling in Bitbucket Server and Data Center](#)
- [Communication of personal data breaches in Bitbucket Server and Data Center](#)
- [Data protection by design and by default in Bitbucket Server and Data Center](#)
- [Records of processing activities in Bitbucket Server and Data Center](#)
- [Right of access by the data subject in Bitbucket Server and Data Center](#)
- [Right to data portability in Bitbucket Server and Data Center](#)
- [Right to erasure in Bitbucket Server and Data Center](#)
- [Right to object in Bitbucket Server and Data Center](#)
- [Right to rectification in Bitbucket Server and Data Center](#)
- [Right to restriction of processing in Bitbucket Server and Data Center](#)
- [Security of processing in Bitbucket Server and Data Center](#)
- [Transfers of personal data to third countries or international organisations in Bitbucket Server and Data Center](#)

GDPR and Atlassian Server and Data Center applications

We understand that customers using Atlassian Server or Data Center applications may have GDPR-specific obligations with respect to the data stored in and processed through those applications. For example, customers may need to access, modify or delete personal data upon request by certain individuals. We've prepared a GDPR FAQ, and guides for our Server and Data Center products, and hope they'll be helpful in assisting our customers with their GDPR-specific obligations.

Please note, when you use Atlassian Server or Data Center applications, Atlassian does not access, collect, store or otherwise process personal data you choose to store within those applications, except in limited cases where such data is provided for incidental support services. For this reason, Atlassian is neither a data processor nor data controller under GDPR for the purposes of the personal data you choose to store within Atlassian Server or Data Center applications. Atlassian may access analytics events associated with your technical use of the products, but only where permitted by your administrator. Such analytics information is filtered to exclude any personal data prior to the analytics information leaving your environment. For more information on the analytics information collected through Atlassian downloadable products, please see our [Privacy Policy](#).

Using multiple Atlassian products?

Check out our GDPR support guides for your Atlassian products:

- [Bamboo Server GDPR support guides](#)
- [Bitbucket Server and Data Center GDPR support guides](#)
- [Confluence Server and Data Center GDPR support guides](#)
- [Crowd Server and Data Center GDPR support guides](#)
- [Fisheye and Crucible Server GDPR support guides](#)
- [Hipchat Server GDPR support guides](#)
- [Hipchat Data Center GDPR support guides](#)
- [Jira Core, Jira Software and Jira Service Management \(formerly Jira Service Desk\) Server and Data Center GDPR support guides](#)
- [Advanced Roadmaps \(formerly Portfolio for Jira\) Server and Data Center GDPR support guides](#)

Automated individual decision-making, including profiling in Bitbucket Server and Data Center

Article 22 of the GDPR gives individuals the right to information about the reasoning behind any decision made about the individual based solely on automated processing, which produces legal effects concerning him or her or similarly significantly affects him or her. Automated decision making of this kind is not part of the core functionality of any Atlassian server or data center products. It is theoretically possible to configure our products to produce automated decisions (e.g, by configuring a JIRA workflow), but this is a configuration only the customer would be aware of and therefore solely responsible for explaining to an individual requesting such information.

Additional notes

 **There may be limitations based on your product version.**

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

 **Third-party add-ons may store personal data in their own database tables or on the filesystem.**

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Communication of personal data breaches in Bitbucket Server and Data Center

The GDPR requires data controllers and data processors to notify regulators and, in some cases, individuals in the event of a personal data breach ([as defined in the GDPR](#)). Please note, when you store personal data in Atlassian server or data center products, the personal data stays on systems within your own environment. Atlassian does not access, store, or otherwise process the personal data you choose to store within the products and is neither a data controller or processor for that data. For more information regarding security-specific configurations available to you through our products, please see the article on [Security of Processing](#).

In the event that we identify a bug or other vulnerability in our own product or plugins that could lead to a personal data breach on within your environment, we will notify you as soon as possible (including, but not limited to via email or the release of an update). In all cases, we will also work to patch the bug or vulnerability as soon as possible.

See also [Security of processing in Bitbucket Server and Data Center](#).

Additional notes

There may be limitations based on your product version.

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

Third-party add-ons may store personal data in their own database tables or on the filesystem.

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Data protection by design and by default in Bitbucket Server and Data Center

[Article 25 of the GDPR](#) sets forth the principle of data protection by design and by default. This is a broad principle with varying meaning and application depending on the context and type of personal data being processed. This principle is unique to each organization, and should always be evaluated with the assistance of legal counsel to determine all efforts required to comply. These efforts may include ensuring certain third party applications you use to process personal data are configured to default to the most privacy-friendly settings available whenever personal data is input. Below is a summary of relevant settings and configurations available through certain Atlassian products, and a discussion of any limitations.

Permissions and restrictions for content

Repository access

Bitbucket Server contains a [permission system](#) that limits the access of specified users to repositories. By default, only administrators have access to repositories. Non-admin users can be given access to [all repositories on a system](#), [all repositories within a specific project](#), or [individual repositories](#). In addition, [access to a repository](#) may be given to all anonymous users (those without an account on the system).

All users with access to a repository can see the repository's full Git history. Each change within a Git repository includes the author's display name, email and avatar photo (either the avatar photo uploaded to Bitbucket Server or a photo on the third-party [Gravatar](#) site). Logged-in users with access to the repository can also see pull requests and other user's comments on commits.

Any logged-in user may view another user's profile information (user name, display name, email address and avatar).

Anonymous users

By default, Bitbucket Server will not show any data to anonymous users. If a repository administrator enables [public access](#) to a repository, data contained within that repository's Git history is available to anonymous users. Each change within a Git repository includes the author's display name, email and avatar photo (either the avatar photo uploaded to Bitbucket Server or a photo that the user stored on the third-party [Gravatar](#) site). For public repositories, this data (author display name, email and avatar photo) is displayed to anonymous users.

Anonymous users may not view a user's profile information (though some of the user profile information is available from public repositories).

Public signup

Bitbucket Server can be configured for public signup (similar to [the public signup feature in Jira](#)). This feature allows anybody with access to the instance to create an account.

With this feature enabled, an anonymous user can easily sign up and get access to more user data (user profiles, comments on commits and pull requests, etc).

Limitations

User profiles

The Bitbucket Server permissions system does not control access to user profile information. Any logged-in user can see another user's profile.

Data in Git repositories

Each change within a Git repository includes the author's display name, email and avatar photo (either the avatar photo uploaded to Bitbucket Server or a photo that the user stored on the third-party [Gravatar](#) site). For public repositories, this data (author display name, email and avatar photo) is displayed to anonymous users.

Additional notes

 **There may be limitations based on your product version.**

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

 **Third-party add-ons may store personal data in their own database tables or on the filesystem.**

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Records of processing activities in Bitbucket Server and Data Center

[Article 30 of the GDPR](#) requires that data controllers and data processors (as defined under the regulation) keep detailed records of what personal data elements they process, why they process the data, where the data is stored, transferred, shared and with whom, how the data is secured and any limitations that may apply to an individual's request to have personal data erased. When you use Atlassian server or data center products, our products may be an application within the scope of your records of processing activities. Whether or not you need to document records of processing activities associated with personal data stored within the product is a determination you should always make with the assistance of legal counsel.

Please note, when you store personal data in Atlassian server or data center products, the personal data stays on systems within your own environment. Atlassian does not access, store, or otherwise process the personal data you choose to store within the products and is neither a data controller or processor for that data.

Below is a list of related articles you may find helpful in compiling the information required for your own records of processing activities:

See also:

- [Right to erasure in Bitbucket Server and Data Center](#)
- [Security of processing in Bitbucket Server and Data Center](#)

Additional notes

There may be limitations based on your product version.

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

Third-party add-ons may store personal data in their own database tables or on the filesystem.

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Right of access by the data subject in Bitbucket Server and Data Center

Under [Article 15 of the GDPR](#), individuals have the right to understand what personal data is being processed about them and the lawfulness of the processing. The GDPR requires that you take reasonable steps to provide this information to the individual, where requested. Whether or not you need to provide the individual with access to personal data stored within the product and the lawfulness of the processing will vary on a case-by-case basis, and is a determination you should always make with the assistance of legal counsel. Once you have determined you have an obligation to provide an individual with access to personal data processed through the product, we have provided the following instructions on how to do so within certain Atlassian products.

Description

The following table lists where user account-level personal data may be stored in a default Bitbucket Data Center and Server installation.

What is it?	What does it get used for?	Where is it stored
Your username (such as <i>jsmith</i>)	Your username is stored so you can log into Bitbucket	Login information is stored in the database
	Your username is used as your personal project key (such as <i>~jsmith</i>)	Information about personal projects is stored in the database
	Your username is stored in the search index when you have a personal project	The search index is stored on the file system
	Your username will appear in the audit log when you make administrative changes to Bitbucket	Audit logs is stored in the database and on the file system
	Your username will appear in access logs, as you browse pages and use Git with Bitbucket	Access logs are stored on the file system
	Your username will appear in Bitbucket mentions in pull request and commit comments	Comments are stored in the database
	Your username will be used to keep track of your application preferences	Application preferences are store in the database
	Your username may be used to store access tokens to external systems (Jira, Hipchat, 3LO, etc)	Access tokens are stored in the database
Your Display Name (such as <i>John Smith</i>)	Your display name is stored on your profile, so Bitbucket can display your name instead of your username	Your profile is stored in the database
	Your display name is stored in the search index when you have a personal project	The search index is stored on the file system
	Your display name may be stored in pull request and commit comments	Comments are stored in the database
	Your display name is stored whenever you commit to a Git repository and in cached Git data	Git repositories and caches are stored on the file system

Your Email Address (such as jsmith@example.tld)	Your email address is stored on your profile, so Bitbucket knows where to send you notifications about content	Your profile is stored in the database
	Your email address is stored with any GPG keys you upload to Bitbucket	GPG keys are stored in the database
	Your email address is stored whenever you commit to a Git repository and in cached Git data	Git repositories and caches are stored on the file system
Your Avatar photo	Your avatar photo may be stored on your profile to help identify you to other users of Bitbucket. <i>By default, Bitbucket will look for your avatar in Gravatar, using a one-way hash of your email address.</i>	Avatar photos are stored on the file system

Workaround

Please read [Bitbucket: Right to erasure](#) for steps on how to remove this personal data.

Additional notes

There may be limitations based on your product version.

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

Third-party add-ons may store personal data in their own database tables or on the filesystem.

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Right to data portability in Bitbucket Server and Data Center

Under limited circumstances, [Article 20 of the GDPR](#) allows a data subject to make a request of a data controller to transfer their personal data to another service provider. The right of the data subject to make this request under [Article 20](#) is highly contextual and you should seek the advice of legal counsel in processing any such request. If you do require the ability to provide personal data in a format as required under [Article 20](#), we suggest you use the content export functions available within the product to do so.

See also [Right of access by the data subject in Bitbucket Server and Data Center](#).

Additional notes

 **There may be limitations based on your product version.**

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

 **Third-party add-ons may store personal data in their own database tables or on the filesystem.**

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Right to erasure in Bitbucket Server and Data Center

Under [Article 17 of the GDPR](#), individuals have the right to have personal data erased. This is also known as the right to be forgotten. The right is not absolute and only applies in certain circumstances. Whether or not you are required to honor an individual's request to have personal data deleted will vary on a case-by-case basis, and is a determination you should always make with the assistance of legal counsel. Once you have determined you have an obligation to delete personal data, we have provided the following instructions on how to do so within certain Atlassian products.

Personal data stored within the product can be divided into one of two areas: 1) account-level personal data; and 2) free-form text. Account-level personal data are data fields that exist within the product for the sole purpose of identifying an individual throughout the product. Examples of account-level personal data include the user's display name, profile picture or avatar and email address. These data elements are generally visible from the user's profile and are used throughout the product to point back to the user's profile when the user is @mentioned or tagged on in certain spaces or content. Deleting account-level personal data elements will automatically remove those data elements throughout the product where the relevant account-level data elements appear and in the database (subject to some limitations discussed below).

If you have included personal data in free-form text, either typed into content spaces or as a custom field label, you will need to use the product's global search feature to surface this personal data and delete it on a case-by-case basis.

Description

In **Bitbucket Server and Data Center 5.16 and later**, administrators can anonymize a deleted user account to remove that user's personal data. See [Users and groups](#) for more information.

In **Bitbucket Server and Data Center 5.15 and earlier**, there are workarounds available to help you remove a specific user's personal data from Bitbucket Server. The SQL queries in this guide are written for PostgreSQL, but can be easily adapted to your system's database.

 Where a workaround involves running SQL scripts directly on your database, we strongly recommend you do this first on a non-production, staging or test environment before running the scripts on your production database. We also strongly recommend you take a backup of your data before making any modifications via SQL scripts.

Version compatibility

All below workarounds are compatible with Bitbucket Server and Data Center 4.0 to 5.15.

Workaround

Deleting a user and removing their personal data

This workaround applies to Bitbucket Server and Data Center 4.0 to 5.15.

Step 1 - Delete the user

A deleted user is no longer able to login or make changes, but their data (SSH keys, GPG keys, access tokens, etc) is preserved for 7 days. After 7 days, the data will be deleted automatically.

 This step **must** be completed to ensure the following steps can be performed successfully. If you don't complete this step first, and have attempted to perform the following steps, which subsequently fail, you will need to restore your system from your backup, and start the full process again.

To delete a user, use one of the following methods, depending on whether your Bitbucket Server instance is using an internal or external user directory.

Internal User Directory	Delete the user from within Bitbucket Server User Management Settings .
External User Directory	<ol style="list-style-type: none"> 1. Delete the user or replace the user's email address and display name in the External Directory with a non-identifying data element. 2. Perform a manual resync of the external directory in Bitbucket Server.

Step 2 - Find the user's ID

Find the `user_id` of the user being deleted, and keep it handy for use in subsequent steps.

 `__username__` is the login name of the user whose personal data you wish to remove.

```
SELECT
  user_id
FROM
  sta_normal_user
WHERE
  name='__username__';
```

Step 3 - Remove the user and their personal project

Change the username and personal project to the user's ID. The queries below must be run in the order given.

 `__user_id__` is the userid value obtained in step 2
`__username__` is the login name of the user whose personal data you wish to remove.

Note the tilde (~) in the second query.

```
UPDATE
  sta_normal_user
SET
  name='__user_id__',
  slug='__user_id__'
WHERE
  user_id = __user_id__;

UPDATE
  project
SET
  name='~__user_id__',
  project_key='~__user_id__'
WHERE
  name = '~__username__';
```

Step 4 - Remove user mentions

Replace user mentions with the new, anonymized username.

 `__user_id__` is the userid value obtained in step 2
`__username__` is the login name of the user whose personal data you wish to remove.

For Bitbucket Server 4.0.0 to 5.3.7, replace user mentions in the `sta_comment` table.

```
UPDATE
  sta_comment
SET
  comment_text=REPLACE(comment_text, '@__username__', '@__user_id__')
WHERE
  comment_text like '%@__username__%';
```

For Bitbucket Server 5.0.0 and above, replace user mentions in the `bb_comment` table. Please note, in Bitbucket Server 5.0.0 - 5.3.7, user mentions will be in both tables.

```
UPDATE
  bb_comment
SET
  comment_text=REPLACE(comment_text, '@__username__', '@__user_id__')
WHERE
  comment_text like '%@__username__%';
```

Replace user mentions in pull request descriptions.

```
UPDATE
  sta_pull_request
SET
  description=REPLACE(description, '@__username__', '@__user_id__')
WHERE
  description like '%@__username__%';
```

Step 5 - Delete the user's OAuth tokens

The user's OAuth tokens allow Bitbucket Server to perform actions on behalf of the user in other systems (like Jira or Bamboo).

 `__username__` is the login name of the user whose personal data you wish to remove.

```
DELETE FROM
  plugin_setting
WHERE
  key_name like 'com.atlassian.oauth.serviceprovider.ServiceProviderTokenStore%'
AND key_value like '%__username__%'
```

Step 6 - Remove the user's dismissed dialogs

Update records of the user's dismissed dialogs.

 `__user_id__` is the userid value obtained in step 2
`__username__` is the login name of the user whose personal data you wish to remove.

```
UPDATE
  plugin_setting
SET
  key_name=REPLACE(key_name, '__username__', '__user_id__')
WHERE
  key_name like 'chaperone:%:__username__';
```

Step 7 - Remove the user's avatar images

If the user uploaded any personal avatar photos, they will be at `<Bitbucket home directory>/shared/data/avatars/users/``__user_id__``/` (where `__user_id__` is the userid value obtained in step 2).

Delete any files in that directory.

Step 8 - Restart the server

[Restart the server](#) to flush all caches and force the UI to update.

Limitations

Data in Git repositories

This limitation applies to **all versions** of Bitbucket Server and Data Center.

This guide does not discuss removing a user's personal data from a Git repository. The personal data stored within a Git repository is essential for auditing and providing a chain of license contribution and authorship.

Each Git repository hold the user's display name and email address against every change they made to the repository. Bitbucket Server will display this information and use a hash of the user's email address to look up an avatar photo on the third-party [Gravatarsite](#).

If you are considering anonymizing a user's data in a Git repository, you can achieve this by following the [Git documentation on filter branch](#).

 Anonymizing data in this way could have a significant impact on development teams, their tools and continuous integration systems. As a result, we strongly recommend against purging the user from Git history.

If you do attempt to rewrite history (using `filter branch` or another method), old commits will still be retained in two circumstances:

1. A forked repository will never remove the old commits in case a fork requires them.
2. If a commit was commented on in a pull request or commit view, the commit will not be removed in order to show the comment in its original context.

Audit, access and application logs

Bitbucket Server logs information for auditing and diagnostic purposes. All these logs may contain the user's username, display name and email address. The access and application logs may also contain the user's IP address.

	Description	Purpose	Location
Audit logs	A list of all audited events	Identify authorized and unauthorized changes, or suspicious activity over a period of time (see Audit logging in Bitbucket Server and How to read the Bitbucket Server Log Formats)	<Bitbucket home directory>/log/audit/
Audit table	A truncated list of recent audit events		Database
Access logs	A list of who accessed the system and what they accessed		<Bitbucket home directory>/log/atlassian-bitbucket-access.log <Bitbucket home directory>/log/atlassian-bitbucket-access-YYYY-MM-DD.log

Application logs	A list of errors, warnings and diagnostic information	May be required to diagnose problems with Bitbucket Server (see Bitbucket Server debug logging and Configure Bitbucket Server Logging)	<Bitbucket home directory>/log/atlassian-bitbucket.log <Bitbucket home directory>/log/atlassian-bitbucket-YYYY-MM-DD.log
-------------------------	---	---	---

You can safely delete any audit logs, access logs or application logs that you no longer need for auditing /diagnostic purposes.

If you wish to delete the list of recent audit events generated by a user, use the following queries:

 `__user_id__` below is the userid value obtained in step 2

```
DELETE FROM
  "AO_BD73C3_PROJECT_AUDIT"
WHERE
  "USER" = __user_id__;

DELETE FROM
  "AO_BD73C3_REPOSITORY_AUDIT"
WHERE
  "USER" = __user_id__;
```

Webhook logging

If you're running Bitbucket Server 5.4.0 or above, Bitbucket Server keeps a record of triggered webhooks in the database, please read [Troubleshooting webhooks](#). Webhooks sent as the result of a user action will contain that user's username. If you no longer need these entries for auditing purposes, you can delete them.

 `__username__` below is the login name of the user whose personal data you wish to remove.

```
DELETE FROM
  "AO_371AEF_HIST_INVOCATION"
WHERE
  "REQUEST_BODY" like '%"name": "__username__"%';
```

Search data

Search data is essential for Bitbucket's repository and code search features. The user's display name, username and avatar are stored in the search data in order to allow searching their personal project.

When the system is restarted, it will reindex all projects and remove the deleted user's data. If you want to force this process to happen before a restart, follow the steps in [Resolve Elasticsearch 404 error](#) by rebuilding the index.

Browser cache

The user's avatar image is cached aggressively by the browser and such caching is not completely under the control of Bitbucket Server. The browser may periodically refresh/purge its cache, and so the avatars will eventually disappear, but exact timings depends on the browser and user's local configuration.

Additional notes

 **There may be limitations based on your product version.**

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

 **Third-party add-ons may store personal data in their own database tables or on the filesystem.**

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Right to object in Bitbucket Server and Data Center

Under limited circumstances, [Article 21 of the GDPR](#) allows a data subject to object to the further processing of their personal data. The right of the data subject to object under [Article 21](#) is highly contextual and you should seek the advice of legal counsel in processing any such request. If you require the ability to restrict further processing of specific personal data within an Atlassian product, we suggest you follow the instructions set forth in the [Right of Rectification](#) article, replacing any personal data elements with elements that do not identify the individual in order to restrict further processing until the issue is resolved. When and if the processing may resume, you can replace the non-identifying elements with the original personal data elements.

See also [Right to rectification in Bitbucket Server and Data Center](#).

Additional notes

 **There may be limitations based on your product version.**

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

 **Third-party add-ons may store personal data in their own database tables or on the filesystem.**

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Right to rectification in Bitbucket Server and Data Center

Under [Article 16 of the GDPR](#), you have the right to have inaccurate personal data rectified. The GDPR requires that you take reasonable steps to rectify the individual's personal data where requested. An example of such a request may be an individual requesting their display name be updated to reflect a name change. Whether or not modifying personal data stored within the product is within the scope of reasonable steps required to honor the individual's request will vary on a case-by-case basis, and is a determination you should always make with the assistance of legal counsel. Once you have determined you have an obligation to rectify personal data, we have provided the following instructions on how to do so within certain Atlassian products.

Personal data stored within the product can be divided into one of two areas: 1) account-level personal data; and 2) free-form text. Account-level personal data are data fields that exist within the product for the sole purpose of identifying an individual throughout the product. Examples of account-level personal data include the user's display name, profile picture or avatar and email address. These data elements are generally visible from the user's profile and are used throughout the product to point back to the user's profile when the user is @mentioned or tagged on in certain spaces or content. Changing account-level personal data elements will automatically populate that change throughout the product where the relevant account-level data elements appear.

If you have included personal data in free-form text, either typed into content spaces or as a custom field label, you will need to use the product's global search feature to surface this personal data and rectify it on a case-by-case basis.

Description

These workarounds will help users change their personal data in Bitbucket Server.

Version compatibility

All versions of Bitbucket Server.

Workaround

Changing display name, email address and avatar photo

Bitbucket Server's configuration affects how users change their display name and email address.

Users managed and stored in an Internal Directory

When users are [managed and stored within the Bitbucket Server internal directory](#), they can view and edit their profile data, including changing their display name, email address and avatar photo.

Users managed and stored in an External Directory

When Bitbucket Server is [integrated with a directory server](#), display name and email address are fetched from the directory server and must be changed there. Users may still change their avatar photo from within Bitbucket Server.

Limitations

Username

Once a user has been created, it is not possible to change their username.

Data in Git repositories

When someone makes a change to a Git repository, the display name and email address they provide (which may be different from the display name and email address stored in Bitbucket Server) are stored with the change. Changing this information is only possible through a process called "rewriting history", which requires rewriting the original change and all following changes.

Given Git's value as an audit tool and chain of authorship, most development teams strongly discourage rewriting history.

Audit, access, application and webhook logs

Bitbucket Server logs information for auditing and diagnostic purposes. All these logs may contain the user's username, display name and email address. The access and application logs may also contain the user's IP address. Any personal data stored in these logs is non-modifiable.

Please read [How to read the Bitbucket Server Log Formats](#), [Audit Logging in Bitbucket Server](#) and [Troubleshooting webhooks](#) to see the kind of data stored in the logs.

The following table shows the default retention policy for logs:

Audit logs	Keep up to 100 25MB files
Access logs	Keep up to 10 25MB files
Application logs	Kept for 31 days
Webhook logs	Kept for 30 days

After the above limits have been reached, the logs are deleted. System administrators may change the above defaults.

Additional notes

There may be limitations based on your product version.

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

Third-party add-ons may store personal data in their own database tables or on the filesystem.

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Right to restriction of processing in Bitbucket Server and Data Center

Under limited circumstances, [Article 18 of the GDPR](#) allows a data subject to request that the processing of their personal data be restricted, rather than entirely deleted. The right of the data subject under [Article 18](#) is highly contextual and you should seek the advice of legal counsel in processing any such request. If you require the ability to restrict processing specific personal data within an Atlassian product, we suggest you follow the instructions set forth in the [Right of Rectification](#) article, replacing any personal data elements with elements that do not identify the individual in order to restrict any further processing until the issue is resolved. When and if the processing may resume, you can replace the non-identifying elements with the original personal data elements.

See also [Right to rectification in Bitbucket Server and Data Center](#).

Additional notes

 **There may be limitations based on your product version.**

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

 **Third-party add-ons may store personal data in their own database tables or on the filesystem.**

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Security of processing in Bitbucket Server and Data Center

The GDPR requires that personal data be processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage. What measures you use to secure the personal data depend on the type of personal data processed, the risk to the individual and relevant industry standard practices. Security measures implemented will vary on a case-by-case basis, and you should be chosen with the assistance of legal counsel. Below is a summary of security tools and configurations available to you within certain Atlassian products, along with how to implement.

General advice

Atlassian recommends our customers implement a secure and reliable network that ensures the protection of its users' data, in the infrastructure that is hosting our products.

[Using Bitbucket Server in the enterprise](#) lists some general considerations and advice for running a production Bitbucket Server system.

Securing user connections

Atlassian strongly recommends customers implement SSL, to secure the TCP communication between Bitbucket Server and its users, or any other system interacting with it.

[Proxying and securing Bitbucket Server](#) covers a number of different ways to implement SSL with Bitbucket Server.

Bitbucket Server and Elasticsearch

Bitbucket Server 4.6 and above, include Elasticsearch to support search functionality. For single-node Bitbucket Server systems, the default Elasticsearch configuration will communicate with Bitbucket Server over an authenticated, encrypted connection. Multi-node Bitbucket Data Center systems must configure their own Elasticsearch installation.

[Using an Elasticsearch cluster with Bitbucket Data Center](#) gives advice on how to secure an Elasticsearch installation.

Load balancing a Bitbucket Data Center system

In Bitbucket Data Center, you are required to have a load balancer in place for the load to be spread across the cluster nodes. For increased security, we recommend that you secure the communication between the load balancer and the Bitbucket Server nodes. There are some examples given in the [Bitbucket Data Center installation guide](#).

On-premise cloud

If you are hosting your Bitbucket Server instance on a cloud service, work with your cloud service provider to develop a suitable plan for securing your environment.

See [securing Bitbucket Server in AWS](#) for considerations when hosting in AWS.

Security best practice

Atlassian offers some best practices for securing both your network, as well as your product. Please review the documentation about [Configuring Confluence Security](#).

Atlassian security and Bug Bounty program

Atlassian releases regular security advisories, to inform our customers about vulnerabilities. These can be viewed and tracked on the [Bitbucket Server security advisories](#) page.

Atlassian also offers the community a way to contribute to enhancing the security of our products through the [Vulnerability BugBounty program](#).

Limitations

Please note the following limitations:

1. Bitbucket Server does not provide **adataencryption** feature.
2. In **BitbucketData Center**, communication between nodes happens over tcp sockets, but is unencrypted.

Additional notes

There may be limitations based on your product version.

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

Third-party add-ons may store personal data in their own database tables or on the filesystem.

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).

Transfers of personal data to third countries or international organisations in Bitbucket Server and Data Center

Various articles under GDPR require data controllers and data processors to keep detailed records regarding transfers of personal data. Atlassian has no such records to maintain because no personal data is transferred to or processed by Atlassian through your use of Atlassian server or data center products.

When you store personal data in Atlassian server or data center products, the personal data stays on systems within your own environment. Atlassian does not access, collect, store or otherwise process personal data in connection with providing those downloadable products to Server and Data Center customers, except in limited cases where such data is provided for incidental support services. Atlassian may access analytics events associated with your technical use of the products, but only where permitted by your administrator. Such analytics information is filtered to exclude any personal data prior to the analytics information leaving your environment. For more information on the analytics information collected through Atlassian downloadable products, please see our [Privacy Policy](#).

Additional notes

 **There may be limitations based on your product version.**

Note, the above-related GDPR workaround has been optimized for the latest version of this product. If you are running on a legacy version of the product, the efficacy of the workaround may be limited. Please consider upgrading to the latest product version to optimize the workarounds available under this article.

 **Third-party add-ons may store personal data in their own database tables or on the filesystem.**

The above article in support of your GDPR compliance efforts applies only to personal data stored within the Atlassian server and data center products. To the extent you have installed third-party add-ons within your server or data center environment, you will need to contact that third-party add-on provider to understand what personal data from your server or data center environment they may access, transfer or otherwise process and how they will support your GDPR compliance efforts.

If you are a server or data center customer, Atlassian does not access, store, or otherwise process the personal data you choose to store within the products. For information about personal data Atlassian processes, see our [Privacy Policy](#).