Режим fast-forward в Git

APRIL 17, 2015

Reading time ~3 minutes

Перевод статьи, посвященной вопросу - что такое fast-forward в системе Git

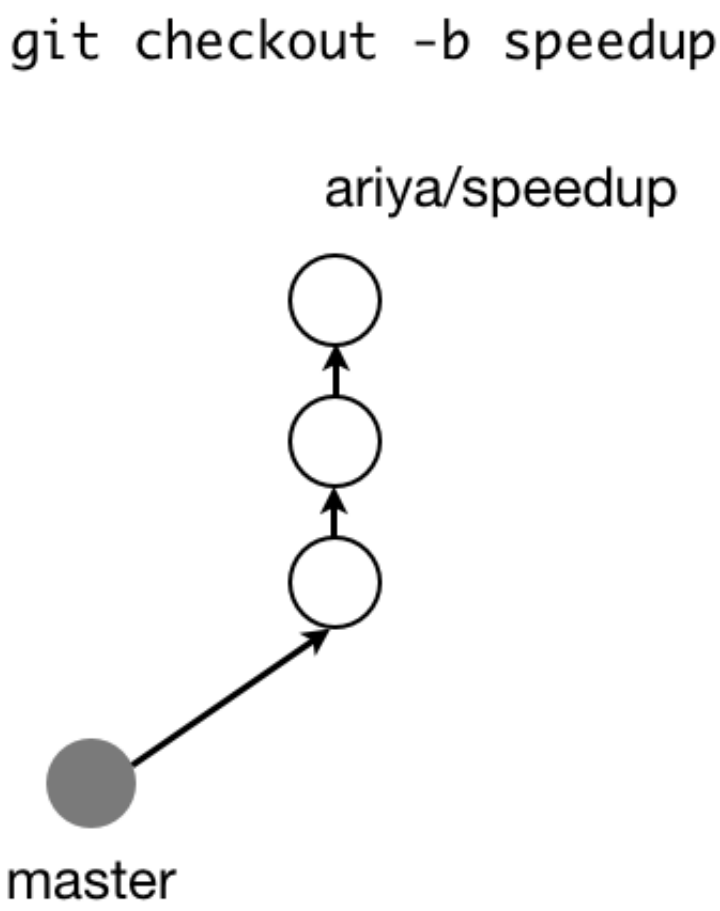
Оригинал данного вольного перевода размещен здесь - Fast-Forward Git Merge. Статья довольно свежая - от 20.09.2013. Почему перевод статьи? Потому что на русском ничего не нашел (окромя книги официальной).

Слияние (`merge`) ветвей является весьма распространенной и обыденной операцией, выполняемой в системе Git. Однако, в некоторых случаях Git выполняет слияние ветвей в режиме **fast-forward**. Но что такое режим **fast-forward** и чем он отличается от *обычного* режима слияния ветвей.

Давайте разберем этот вопрос на конкретном примере. Допустим, при работе в репозитории мною была создана ветка `speedup` из текущей ветки `master` :

```
$ git checkout -b speedup
```

Поработав некоторое время в этой ветке, я создал несколько `commit` ‘ов (три коммита, три белых кружка на рисунке):

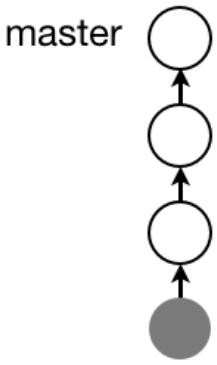


После этого я решил, что работа в ветке `speedup` мною выполнена и я сделал `push` этой ветки на свой удаленный репозиторий.

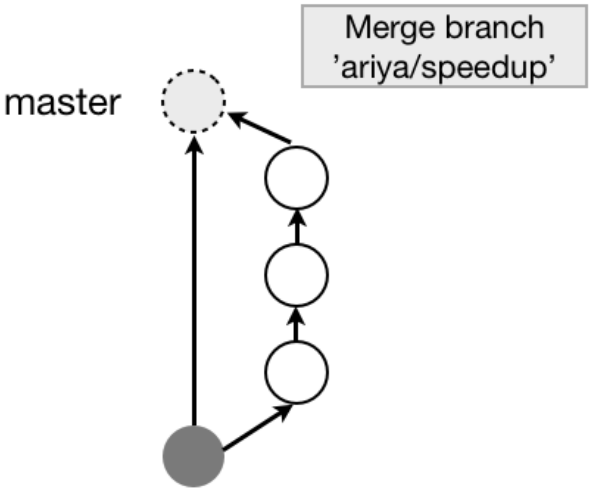
Между тем, обратите внимание на тот важный факт, что в ветке `master` **не было внесено никаких изменений** с того самого момента, когда я сделал ответвление от нее - ветку `speedup`.

Когда менеджер проекта уведомится о том факте, что разрабатываемая мною ветка готова для интеграции в проект, то им будет выполнены стандартные шаги по публикации моей работы - команды `git fetch` и `git merge`. Так как в ветке `master` не было внесено изменений с того момента (серый кружок), как была создана ветка `speedup`, то при слиянии система Git применит метод **fast-forward**. В этом случае вся последовательность коммитов будет иметь *линейный* вид (**левая часть** рисунка):

```
git fetch ariya
git merge ariya/speedup
```




```
git fetch ariya
git merge -no-ff ariya/speedup
```



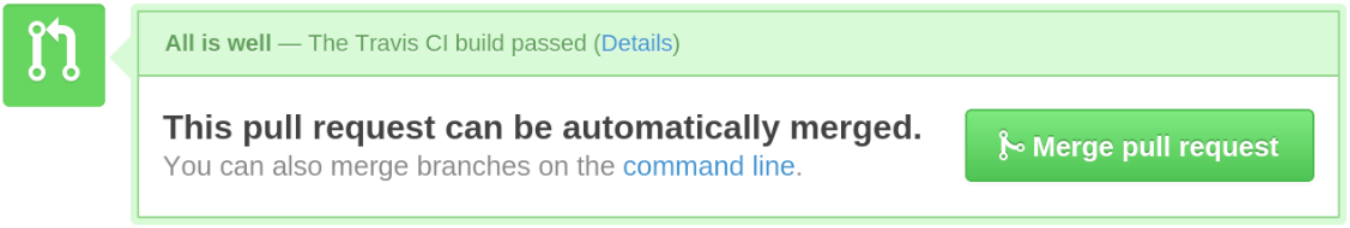
Примечание переводчика: другими словами, Git просто произведет перемещение указателя HEAD с серого кружка (последний коммит ветки `master`) на последний белый кружок (последний коммит ветки `speedup`). И будет считать, что все это - ветка `master`. Получится своеобразный *проброс указателя* HEAD вдоль последовательной линии коммитов. Именно поэтому автор упоминает термин *линейного* вида.

Другим вариантом слияния веток в данном случае является использование флага `-no-ff` в команде `git merge`, что представляет из себя сокращение от - **no fast-forward**. В этом случае ситуация будет несколько иной - ее схематичное изображение на **правой** части рисунка вверху.

В этом случае при слиянии веток `master` и `speedup` системой Git создается коммит (кружок со штрихованной границей), задача которого - **информировать о слиянии веток**. Другими словами - *цель* и *задача* этого коммита только в информировании о слиянии веток `master` и `speedup`.

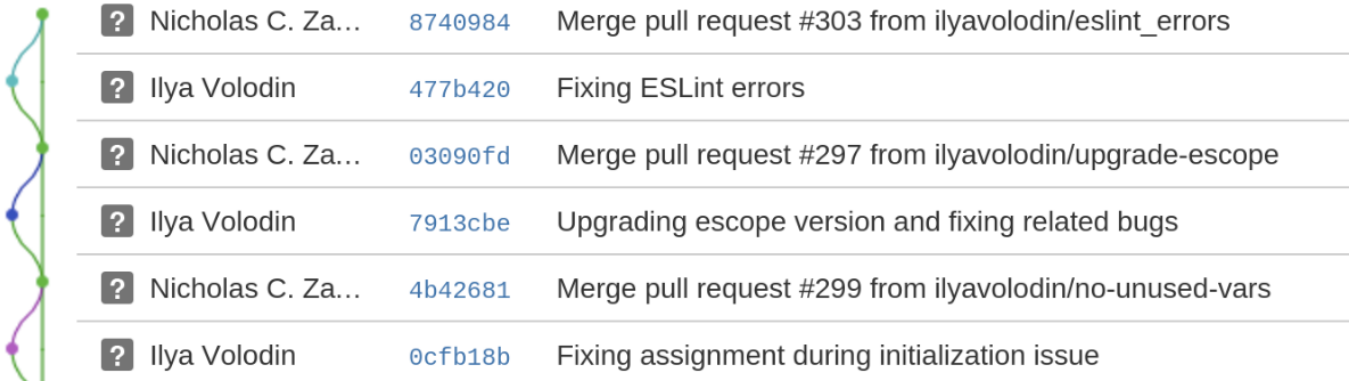
В ситуациях подобного рода система Git всегда старается применить режим **fast-forward**, если это возможно. Однако, такое поведение Git можно легко изменить на режим **no fast-forward** и сделать его поведением системы по умолчанию.

Пожалуй, самой главной неприятной неожиданностью при использовании `merge` в режиме **no fast-forward** - это когда вы нажимаете ту самую зеленую кнопку “Merge” в web-интерфейсе сервиса GitHub при выполнении pull request:



К сожалению, на сегодняшний день web-интерфейс GitHub выполняет операцию слияния (`merge`) так, как если бы для этой команды был установлен флаг `-no-ff` . Другими словами, даже если при слиянии веток ситуация позволяет использовать режим **fast-forward**, GitHub не будет его использовать.

Одним из разумных объяснений такого поведения GitHub является тот факт, что каждая операция `pull request` должна быть однозначно **идентифицирована**. К примеру, несколько последних коммитов проекта (в качестве примера мною был взят проект ESLint - ничего личного) могут выглядеть таким образом:



Внимательно посмотрите на приведенный выше рисунок. На нем хорошо видно, что некоторые слияния (`merge`) веток могут быть выполнены в режиме **fast-forward**. Но своеобразный подход GitHub к слиянию веток привел к тому, что линейная история коммитов была превращена в нечто похожее на рисунок железнодорожного пути.

Если в нескольких словвах подвести общий итог вышесказанного, то он окажется следующим.

Режим **no fast-forward** хранит всю информацию о слияниях веток. Такой подход может оказаться запутанным и сложным, если необходимо прочитать историю коммитов.

С другой стороны, режим **fast-forward** хранит не всю информацию о слияниях веток. Такой подход более простой для чтения, но в этом случае становится неочевидным история веток проекта.

На этом все.
