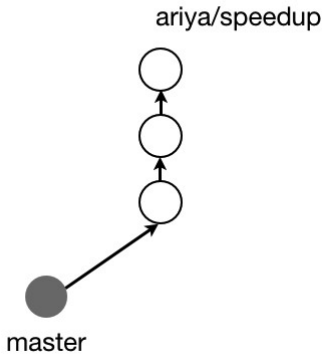
Fast-Forward Git Merge

Source : http://ariya.ofilabs.com/2013/09/fast-forward-git-merge.html

Merging a branch is a pretty common operation when using Git. In some circumstances, Git by default will try to merge a branch in a **fast-forward** mode. How is this different with a merge *without* fast-forwarding?

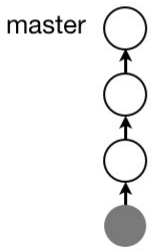
Let us assume that I created a topic branch named `speedup` from the current `master`. After working on this branch for a while (three commits, those white circles), I finally decided that I am done and then I pushed it to my own remote. Meanwhile, nothing else happened in the master branch, it remained in the same state right before I branched off. The situation is depicted in the following diagram.

```
git checkout -b speedup
```

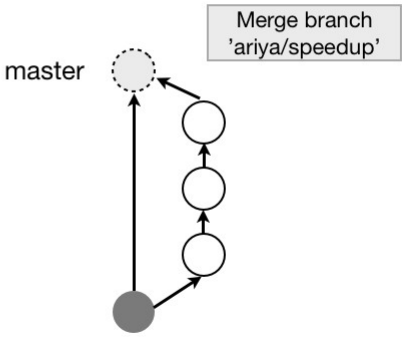


Once the project maintainer got notified that my branch is ready to be integrated, she might use the usual steps of `git fetch` followed by `git merge` and thus, my work is landed in the source tree. Because master has not been changed since the commit (gray circle) which serves as the base for the said topic branch, Git will perform the merge using **fast-forward**. The whole series of the commits will be *linear*. The history will look like the diagram below (**left** side).

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




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

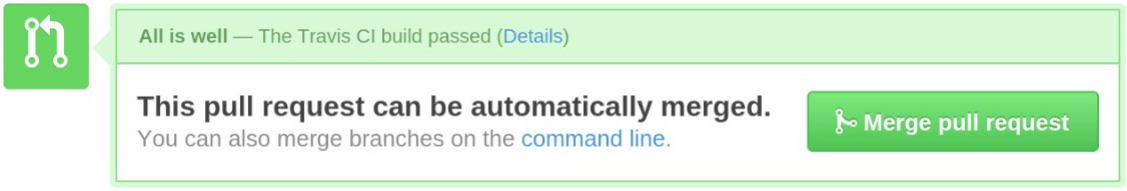


Another variant of the merge is to use `-no-ff` option (it stands for **no fast-forward**). In this case, the history looks slightly different (**right** side), there is an additional commit (dotted circle) emphasizing the merge. This commit even has the right message informing us about the merged branch.

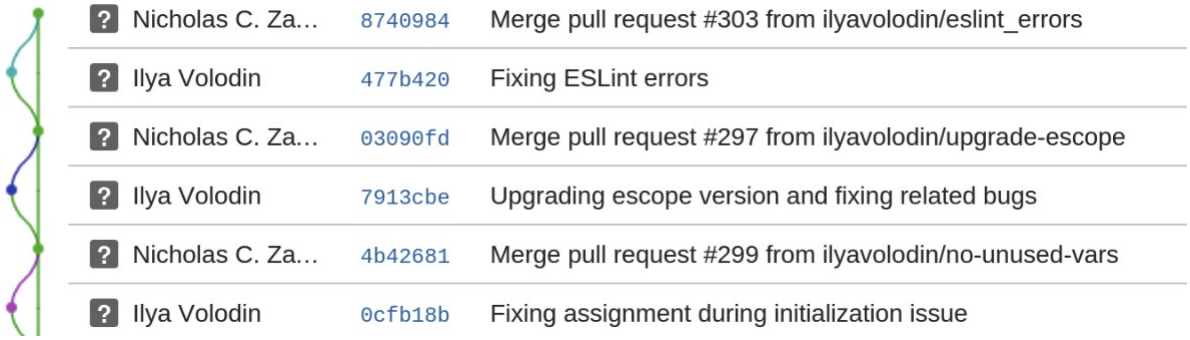
The default behavior of Git is to use fast-forwarding whenever possible. This can be

changed, the no fast-forward mode can be easily set as the default merge using the right proper configuration.

Perhaps the typical encounter of non fast-forward merge is via the use of the green *Merge* button on GitHub, as part of its pull request workflow. When someone creates a pull request, there is a choice to merge the change (whenever GitHub thinks it is possible to do it) just pressing this button on the project page.



Unfortunately, at least as of now, GitHub's web interface will perform the merge as if you would specify `--no-ff`. In other words, even if there is a possibility of fast-forwarding, GitHub will *not* do that. One possible explanation is so that the pull request could be **identified**. For example, the few recent commits of a project (I picked ESLint as an example, nothing particular about the project) can look like this:



Looking at the graph, it is clear that those few patches can be merged using fast-forward mode. Alas, GitHub style of merge turned the commit history from a linear progression to something that resembling a railroad diagram.

In short, non fast-forward merge keeps the notion of **explicit branches** . It may complicate the commit history with its non-linear outcome at the price of preserving the *source* of the branches (pull requests, when using GitHub). On the other hand, fast-forward merge keeps the changesets in a **linear history**, making it easier to use other tools (log, blame, bisect). The source of each branch will not be obvious, although this is not a big deal if the project mandates the strict cross-reference between the commit message and its issue tracker.