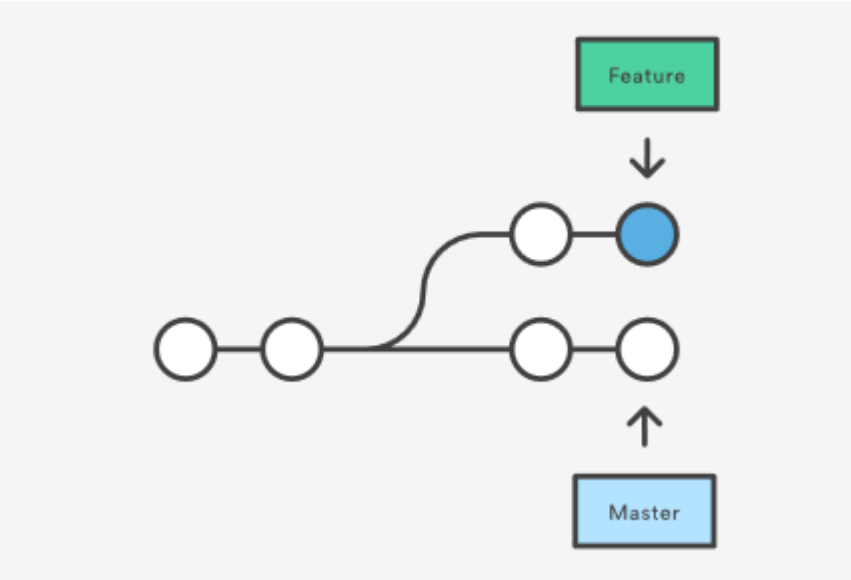
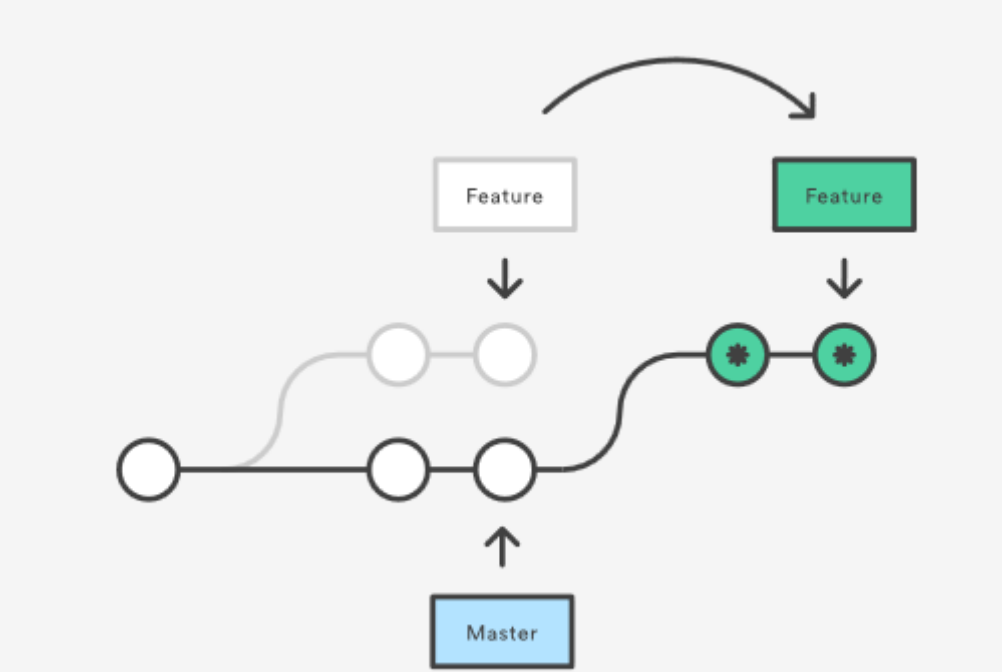




Rebasing: Moving a commit from one branch to another

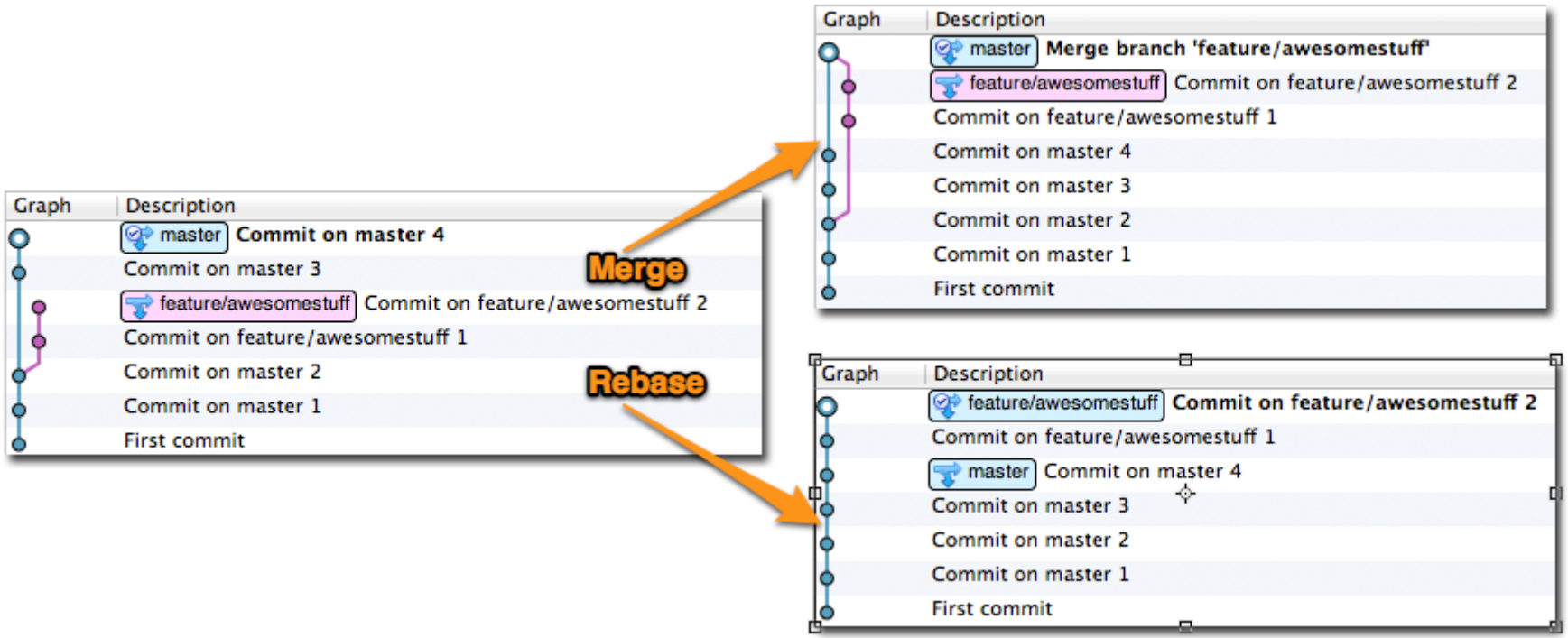
You have two options for integrating your feature into the master branch:

- merging directly or
- rebasing and then merging.

The former option results in a 3-way merge and a merge commit, while the latter results in a fast-forward merge and a perfectly linear history.

See Code flow...
https://github.com/nvie/gitflow

Other links:
https://www.atlassian.com/git/tutorials/rewriting-history/git-rebase


Graph
Description
master
Commit on master 4
Commit on master 3
feature/awesomestuff
Commit on feature/awesomestuff 2
Commit on feature/awesomestuff 1
Commit on master 2
Commit on master 1
First commit
Merge
Rebase
Graph
Description
master
Merge branch 'feature/awesomestuff'
feature/awesomestuff
Commit on feature/awesomestuff 2
Commit on feature/awesomestuff 1
Commit on master 4
Commit on master 3
Commit on master 2
Commit on master 1
First commit
Graph
Description
feature/awesomestuff
Commit on feature/awesomestuff 2
Commit on feature/awesomestuff 1
master
Commit on master 4
Commit on master 3
Commit on master 2
Commit on master 1
First commit

Merging Pros and Cons

- **Merging Pros**
  - Simple to use and understand.
  - Maintains the original context of the source branch.
  - The commits on the source branch remain separate from other branch commits, **provided no fast-forward merge is done**.
    - Can be useful in the case of feature branches
      - merge from feature to another branch later.
  - Existing commits on the source branch are unchanged and remain valid; it doesn’t matter if they’ve been shared with others.

Merging Pros and Cons

- **Merging Cons**
  - If the need to merge arises simply because multiple people are working on the same branch in parallel, **the merges don’t serve any useful historic purpose and create clutter**.
  - Therefore if you are using only a single branch where multiple people are working together, you may want to consider using rebasing instead.

Rebasing Pros and Cons

- **Rebasing Pros**
  - Simplifies your history.
  - **Is the most intuitive and clutter-free way to combine commits from multiple developers in a shared branch**

Rebasing Pros and Cons

- **Rebasing Cons**
  - Slightly more complex, especially under conflict conditions. Each commit is rebased in order, and a conflict will interrupt the process of rebasing multiple commits. With a conflict, you have to resolve the conflict in order to continue the rebase.
  - Rewriting of history has implications if you've previously pushed those commits elsewhere.
    - In Mercurial, you cannot push commits that you later intend to rebase, because anyone pulling from the remote will get them.
    - In Git, you may push commits you may want to rebase later (as a backup) but *only* if it's to a remote branch that *only you use.* If anyone else checks out that branch and you later rebase it, it's going to get very confusing.

When to use what?

- **Shared branches**
  - With shared branches, several people commit to the same branch, and they find that when pushing, they get an error indicating that someone else has pushed first.
    - In this case, ‘Pull with rebase’ approach is recommended.
    - Pulling down other people’s changes and immediately rebasing your commits on top of these latest changes, allowing you to push the combined result back as a linear history.
      - It’s important to note that your commits must not have been shared with others yet.
  - **The only time you shouldn’t rebase and should merge instead is if you’ve shared your outstanding commits already with someone else via another mechanism**, e.g. you’ve pushed them to another public repository, or submitted a patch or pull request somewhere.

Feature branches

- Final Merge
- Keep your feature branch up to date

Feature branches

- Final Merge
  - When building a feature on a separate branch, **you're usually going to want to keep these commits together in order to illustrate that they are part of a cohesive line of development**.
  - Retaining this context allows you to identify the feature development easily, and potentially use it as a unit later, such as merging it again into a different branch, submitting it as a pull request to a different repository, and so on.
  - **Therefore, you're going to want to merge rather than rebase** when you complete your final re-integration, since merging gives you a single defined integration point for that feature branch and allows easy identification of the commits that it comprised.

Feature branches

- Keep your feature branch up to date
  - There are two ways you can bring your feature branch up to date:
    - Periodically merge from the (future) destination branch into your feature branch. This approach used to cause headaches in old systems like Subversion, but actually works fine in Git and Mercurial.
    - Periodically rebase your feature branch onto the current state of the destination branch

Summary

- When multiple developers work on a shared branch, pull & rebase your outgoing commits to keep history cleaner (Git and Mercurial)
- To re-integrate a completed feature branch, use merge (and opt-out of fast-forward commits in Git)
- To bring a feature branch up to date with its base branch:
  - Prefer rebasing your feature branch onto the latest base branch if:
    - You haven’t pushed this branch anywhere yet, or
    - You’re using Git, and you know for sure that other people will not have checked out your feature branch
    - Otherwise, merge the latest base changes into your feature branch

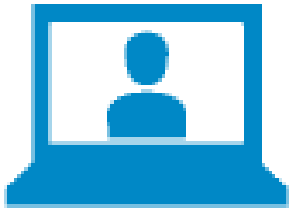
Thank you for being with Edureka!!

We would like to remind you that, your association with edureka does not stop here!

Remember

You have lifetime access to the updated courses and contents you are registered for!

You get life time support for all the issues you might encounter on your registered courses



You get free access to the webinars which are delivered across courses

You can get a discount on every next course you like from edureka! For more details keep checking your LMS

You are liable for referral benefits, if you refer anyone to edureka!

Please post all your reviews here: http://www.quora.com/Reviews-of-Edureka-online-education

Course Url


Thank you!