Merging in Git

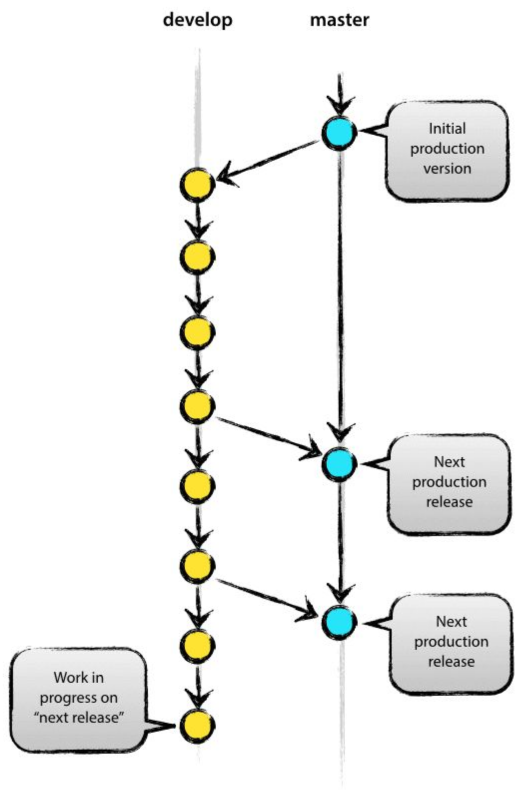
Branching

Recall that branching creates a copy of the master repository, on which you can work without disrupting the main branch.

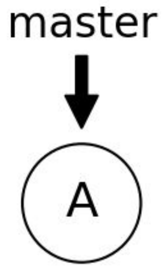
But what happens when you are ready to share your work and combine it into the master branch?

You can merge your work back in.

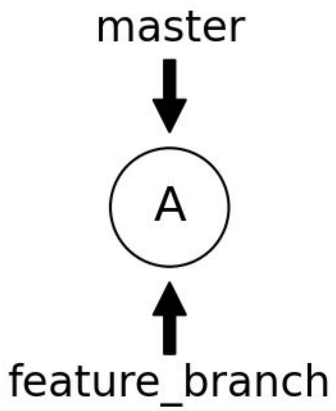
See https://www.atlassian.com/git/tutorials/using-branches/git-merge for a great blog post on this subject.


develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

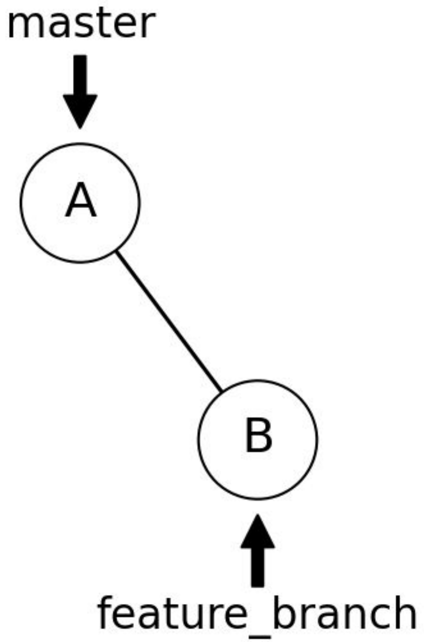
Let’s say your repository has just the master branch.



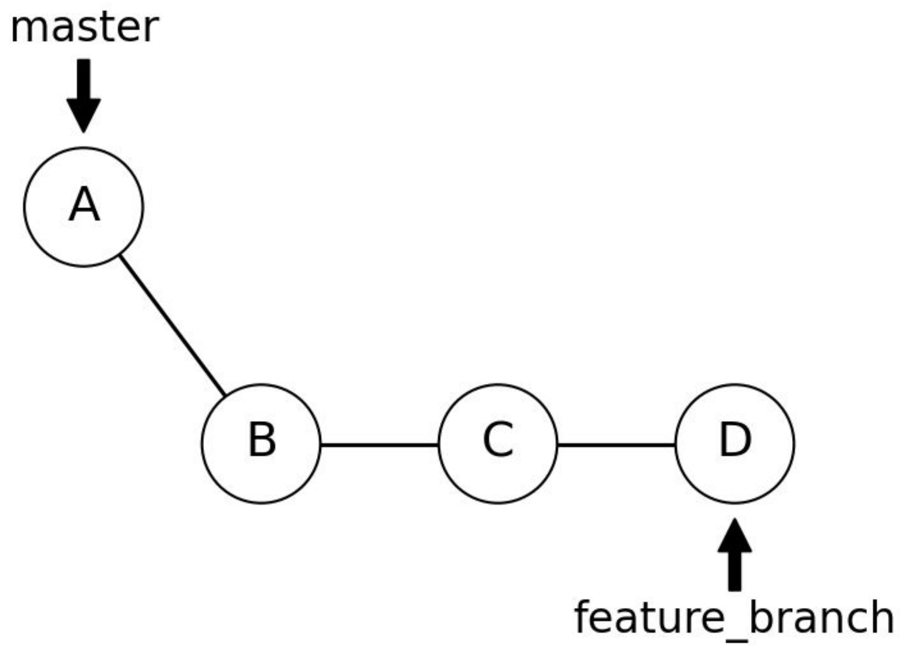
You create a feature_branch to work on an issue.



Then do some work and commit on the feature branch.

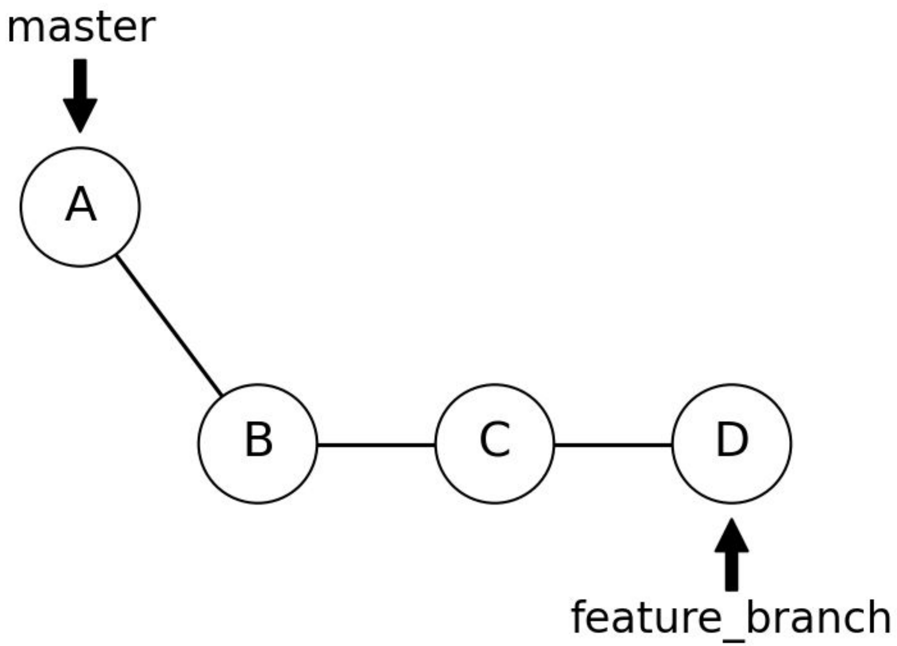


Then more commits on the feature branch.



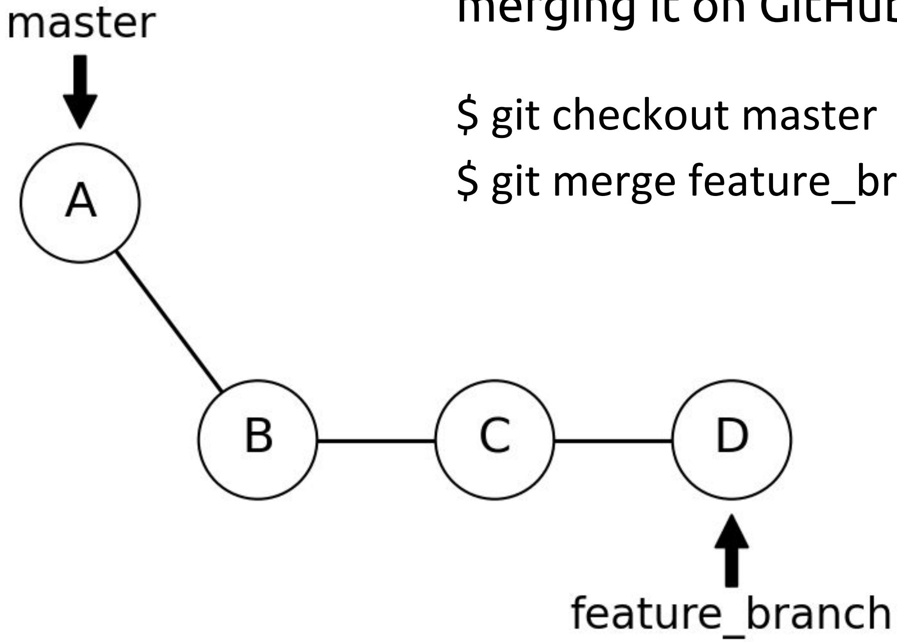
Then more commits on the feature branch.

Now, you want to merge your changes into master.



Then more commits on the feature branch.

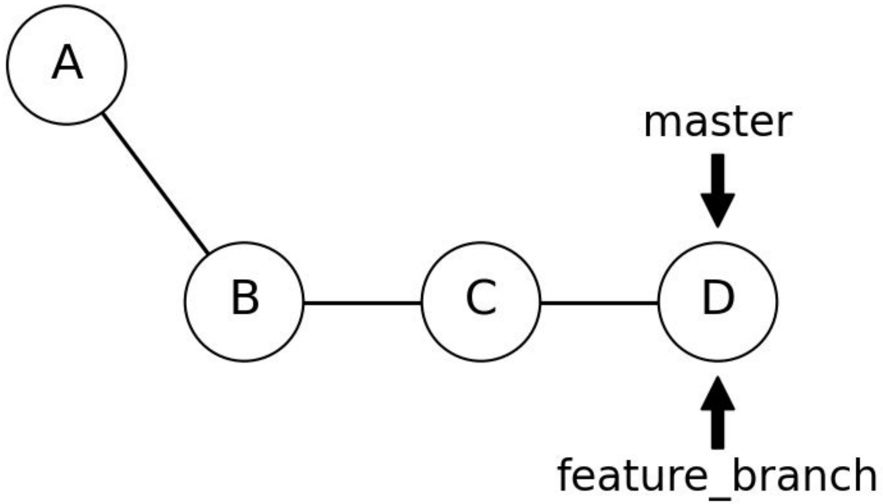
Now, you want to merge your changes into master.



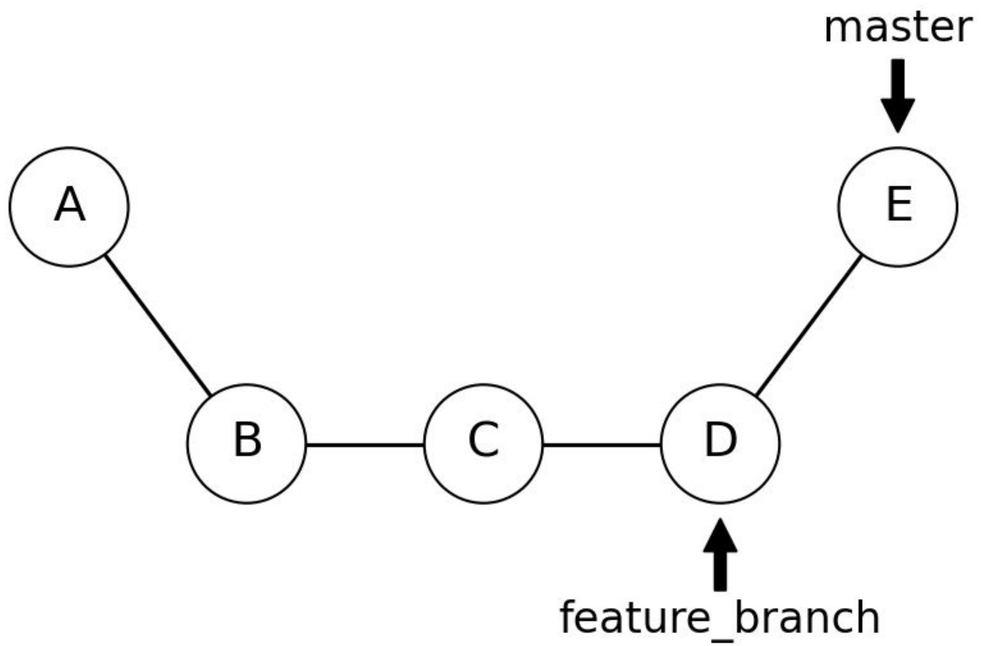
This can be done via a **pull request** and merging it on GitHub or by doing:

```
$ git checkout master
$ git merge feature_branch
```

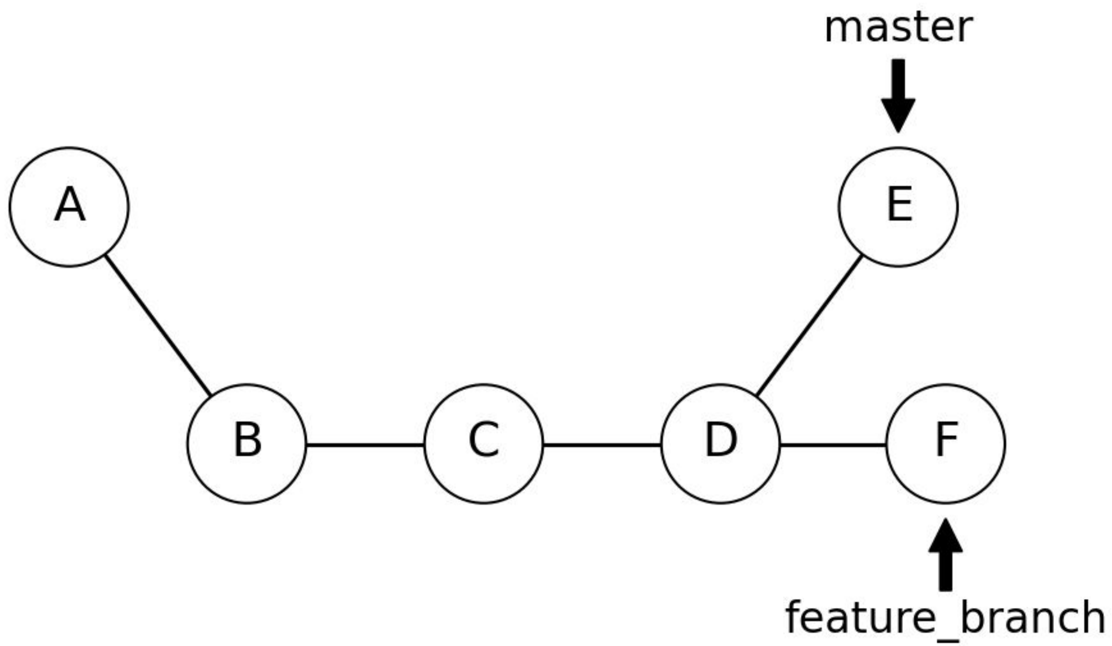
In this case, since there were no commits on master since the feature_branch was created, it will do a **fast-forward merge**.



After that, you can make further commits on master.

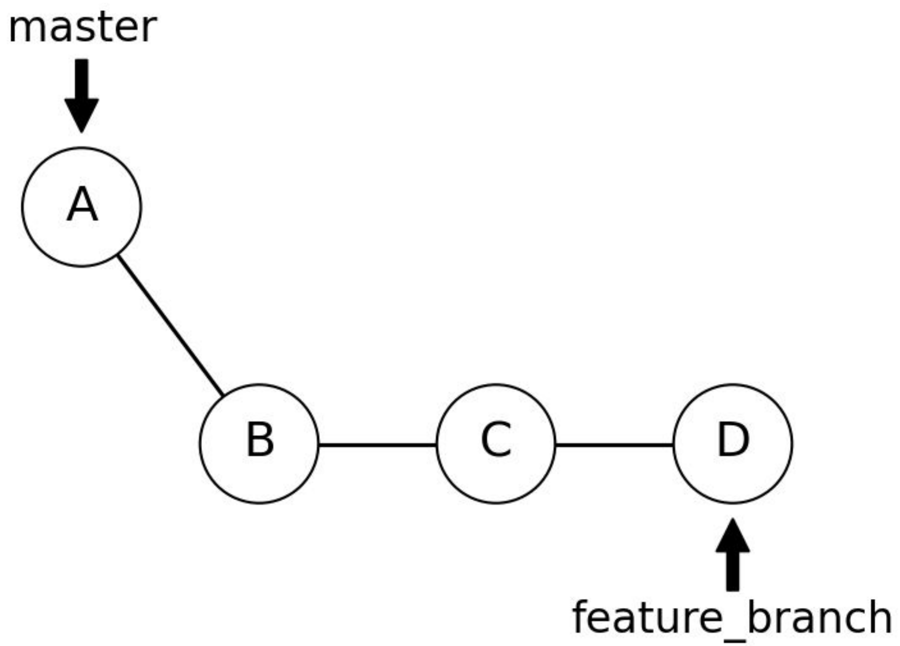


Or you could make commits on feature_branch.

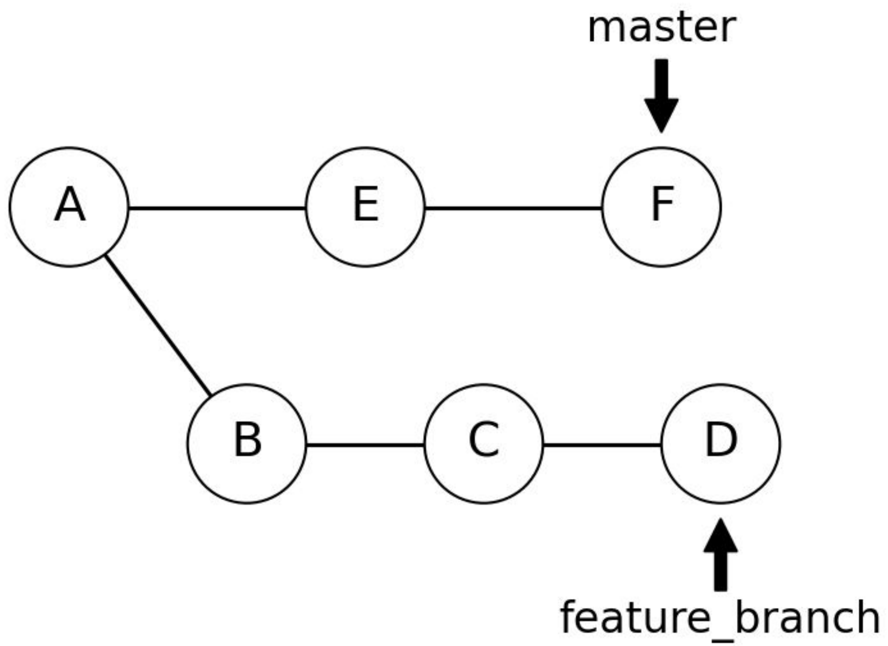


But, what if there had been other commits on master prior to the merge?

We have some new commits on the feature branch.

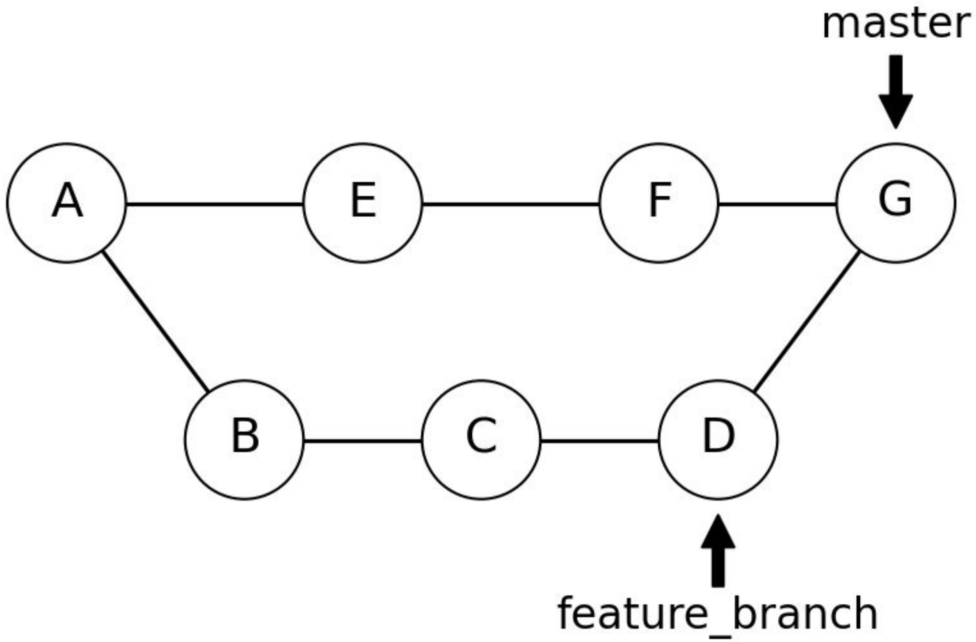


This time, there have been commits both on master and on feature_branch

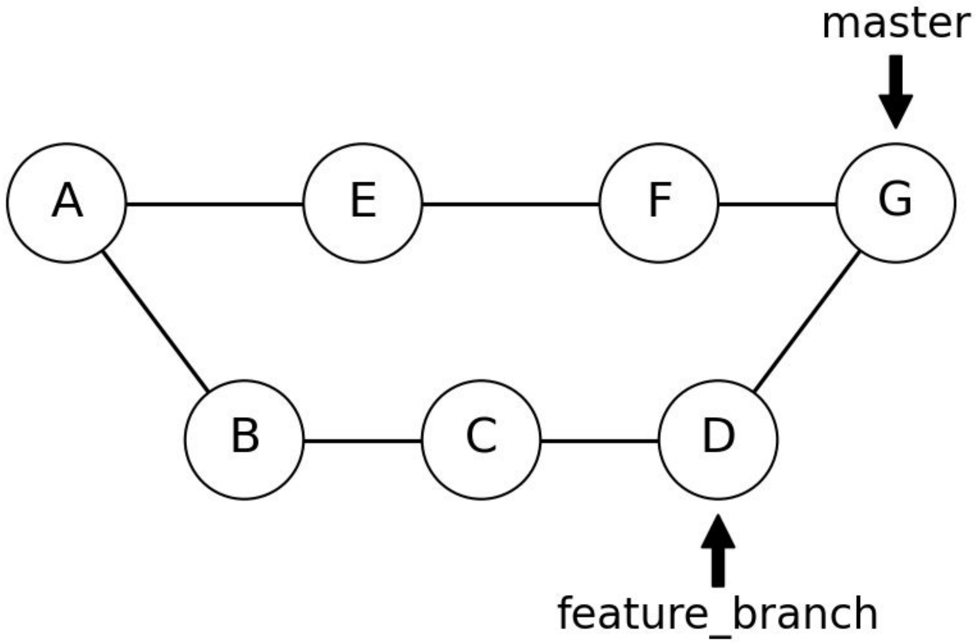


This will perform a “**3-way merge**” where git will combine the commits of the two branches and create a new commit.

After the merge, master will have A, B, C, D, E, F, and G and feature_branch will have A, B, C, and D.



What could go wrong?



Merge Conflicts

Git will do its best to combine the changes from both branches.

However, if there are changes in the same part of a file on both branches, it may not be able to resolve those changes.

You have a **merge conflict** and must resolve it.

Merge Conflicts

If we had a merge conflict when trying to merge feature_branch into master, git will append

```
<<<<<<< HEAD
version on master
=======
version on feature_branch
>>>>>>> feature_branch
```

You need to go in and resolve these changes and then commit them.

Avoiding Merge Conflicts

Before creating a new branch off of master, make sure that you have the latest changes by doing

```
$ git checkout master
$ git pull
```

Avoiding Merge Conflicts

Before creating a new branch off of master, make sure that you have the latest changes by doing

```
$ git checkout master
$ git pull
```

Merge conflicts are always annoying but doubly so for Jupyter Notebooks, so be careful so that you can avoid them.

Git Pull

```
$ git pull
```

grabs any changes from GitHub and merges them with your local branch.

Note that it will fetch from GitHub’s version of your *current branch*, so make sure to check which branch you are on prior to doing it by using

```
$ git branch
```