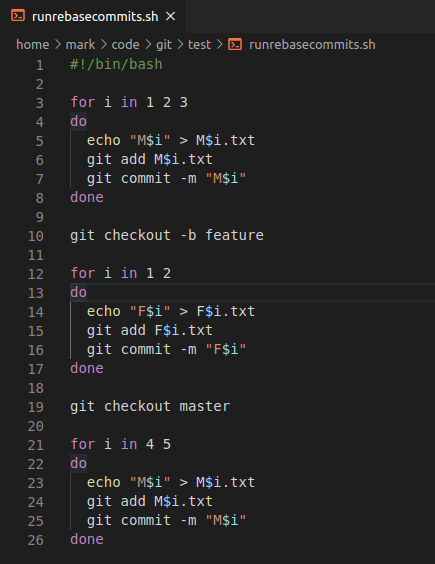
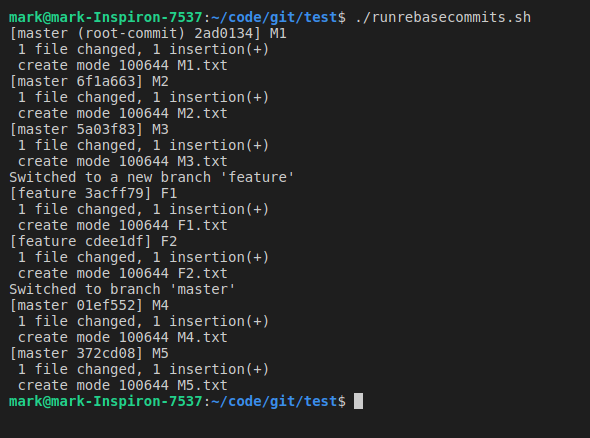
### **The Basic Rebase**

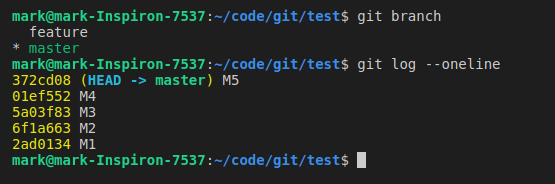
I have a bash shell script that creates three commits in the master ranch. Then, the script creates and checks out a new branch called feature. In the feature branch two commits are created. Finally, two more commits are created in the master branch.



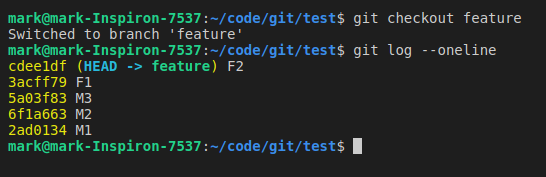
Here we run the script:



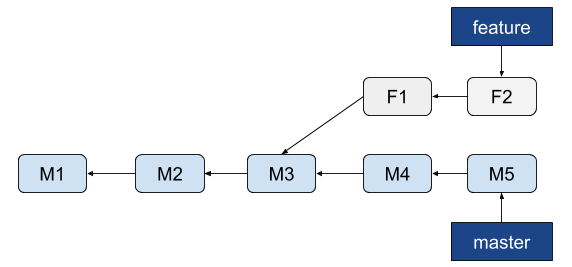
We will take a look at the master branch:



Now, a look at the feature branch:

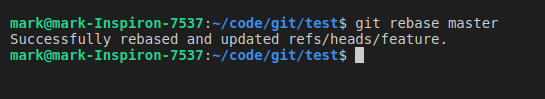


Note that the feature branch history includes the initial commits from the master branch. In addition, you should note that we diverged our work when making commits on two different branches.



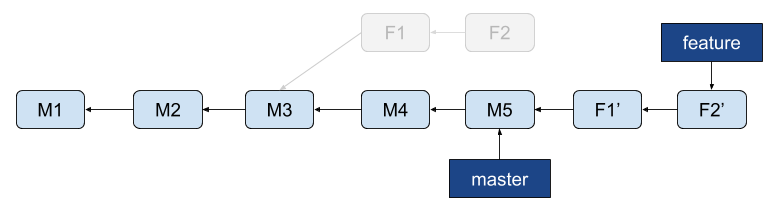
Let’s now take the changes that was introduced in F1 and F2 and reapply it on top of M5. In Git, this is called rebasing. With the rebase command, you can take all the changes that were committed on one branch and replay them on a different branch.

For this example, we are on the feature branch. From here we rebase it onto the master branch as follows:

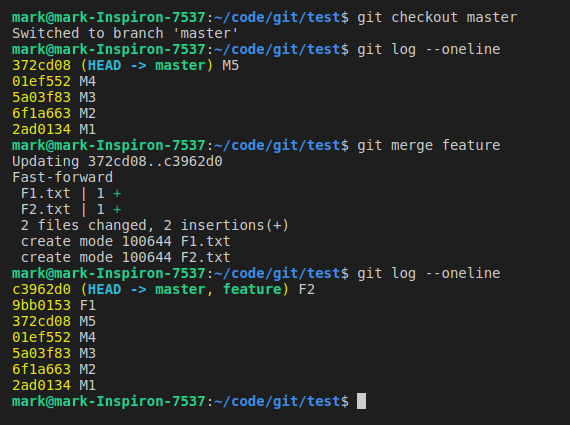


As per <https://git-scm.com/book/en/v2/Git-Branching-Rebasing>, “This operation works by going to the common ancestor of the two branches (the one you’re on and the one you’re rebasing onto), getting the diff introduced by each commit of the branch you’re on, saving those diffs to temporary files, resetting the current branch to the same commit as the branch you are rebasing onto, and finally applying each change in turn.”

Now that we have rebased the feature branch commits onto the master branch, here is the following commit history:



Next, you can go back to the master branch, view its history, do a fast-forward merge, and then view its history again to see the .



Here is the result:

