(I would recommend abbreviating your git commands like “branch” to br and “checkout” to co to make typing easier. I don’t remember how but it isn’t hard to find online)

Making a new branch:

git co master (or whatever you want to branch off of, but usually you want to branch off of master)
git br [branchname]
git co [branchname]
git push -u origin [branchname] (this will tell github that your branch exists and will allow you to easily push your branch up with just the command git push)

Merging your branch onto master:

Step 1 Make your branch reflect changes made to master. There have probably been changes made to master since you first made your branch

git co master
git pull (just to make sure your master branch is up to date. Even if it says “Already up-to-date” don’t trust it because it’s often wrong, just pull anyway)
git co [branchname]
git rebase -i master
What this step does is it puts all the commits you made in your branch that are not in master on top of the commits in master.
This will send you to vi, with a text file of your commits and the word “pick” next to each of them. For each of your commits, there are some options of what you want to do with them:

- Pick/p: I want to keep this commit
- Rename/r: Keep the commit but change the commit message (it will send you to a screen after this where you can change the message)
- Squash/s: Move the changes made in this commit to the commit listed before this one, basically combining two commits into one
- Fixup/f: Same as squash but also rename the commit
- Delete/d: get rid of this commit

Mark each commit with what you want to do to it, save the text file and exit (in vi I think the is :wq but I’m not positive. There’s a way to change the default text editor to emacs but I don’t remember, you can find it online)
At this point there may be merge conflicts, which are denoted in files as <<<<<<<. Fix those conflicts to reflect what it should look like, and then run git rebase --continue. If scary things are happening and you think you might have broken something, just run git rebase --abort to stop all of this.
git push -f (this is a dangerous command, it will push all of your changes to github without regard for any kinds of conflicts. In this case it’s necessary because you probably changed the structure of commits in your branch, and regular git push only lets you add more commits, not

change existing ones. But as long as you're only editing your branch, this isn't an issue here, since changing the structure of your commits is exactly what we're trying to do

Step 2 put the changes made in your branch into master

git co master
git pull (just to make sure nobody edited master between when you started and now. If there are changes then restart this process)
git merge [branchname]
git push

Other helpful commands:

git log (this will show you the commits made in your branch)
git br (shows all the branches on your computer and highlights which you are currently on)
git status (this tells you which files have been edited in your branch and if they are staged)
git diff [branch 1]..[branch 2] (this tells you the differences between two branches. It also works with commit hashes displayed in git log instead of branch names)