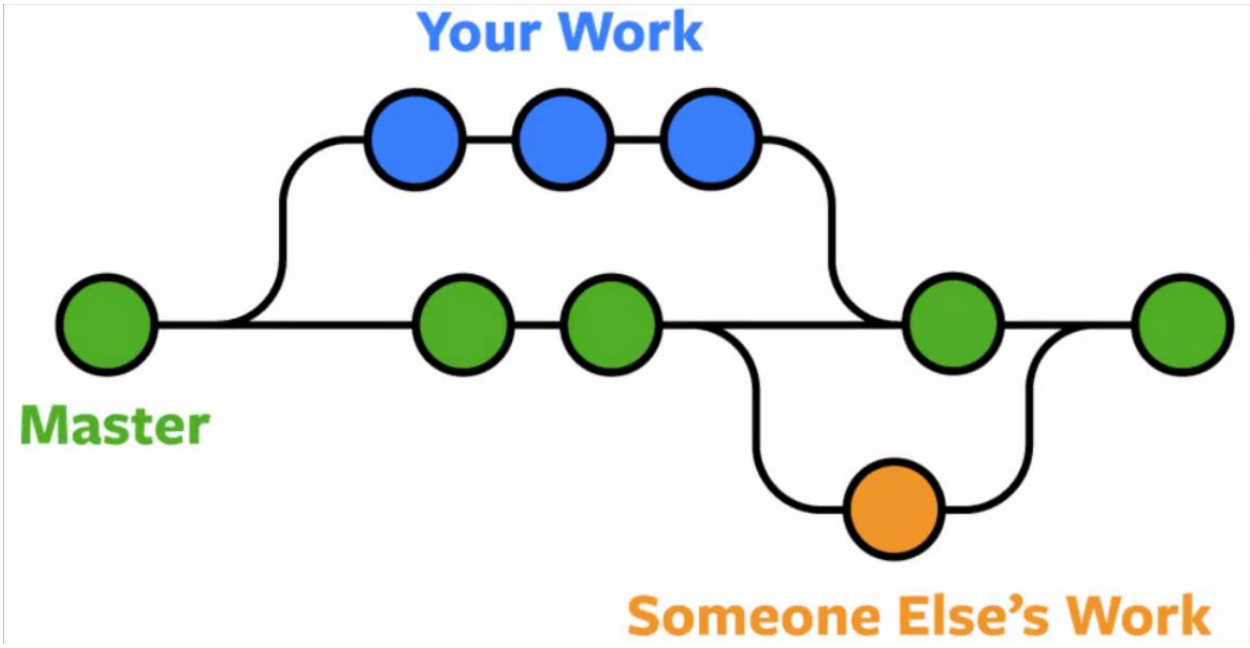
Git Branching

We normally work on the default branch (now called "main", previously called "master"). But think about a big group of people working on the same codebase – you wouldn't work on the same branch because this would step on each other's code.

Git branching makes it easier to collaborate, safer to push up commits without overriding the "Production" code, means we can test it before committing to main branch.

Best not to work on the main branch – most companies will have policies for how they handle GitHub.

We can branch off and work on what's sometimes called a "feature" branch or "dev" branch.

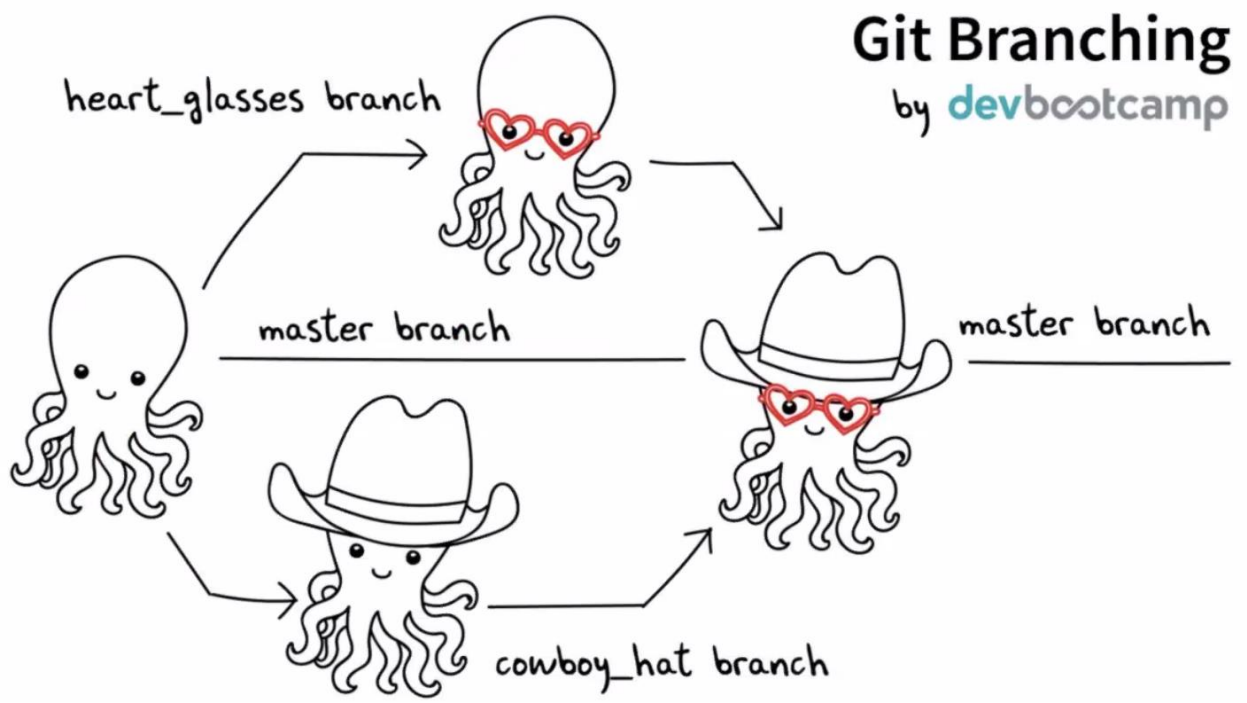


(Master is now Main)

We branch off to make our own changes then make a "pull request" to merge our changes in to the main branch. This may then start a code review so you can discuss how to merge (e.g. what to keep). Git will ask you what to keep.

Different to forking, as forking makes a clone (snapshot) of the repo on your GitHub account without affecting the main repo.

A more visual representation of how this works:



Then we looked at the GitHub docs in pairs (https://docs.github.com/en/get-started/quickstart/github-flow)

Liz Demo of Git

Clone a repo

Used git checkout -b readme-content *-b makes a new branch then we give it a name(checkout alone checks out to an existing branch). Use a descriptive name for what you are working on*

Used git branch to see all branches with an indicator of which branch you are on

```
lizkaufman branching-demo git branch
  main
* readme-content
```

Change between branches using git checkout *branch_name*

```
lizkaufman branching-demo git checkout main
M       README.md
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
lizkaufman branching-demo git checkout readme-content
M       README.md
Switched to branch 'readme-content'
```

If you make a change, then add, commit, push - you will get a fatal error:

```
lizkaufman branching-demo git add .
lizkaufman branching-demo git commit -m 'added line about handiness of branching'
[readme-content 3e8decb] added line about handiness of branching
 1 file changed, 3 insertions(+), 1 deletion(-)
lizkaufman branching-demo git push
fatal: The current branch readme-content has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin readme-content
```

Upstream refers to the remove repo on GitHub. We have made our branch locally, so remote repo doesn't know about our branch!
Git tells you what to do (bottom line of code in above screenshot) and everything will be good:

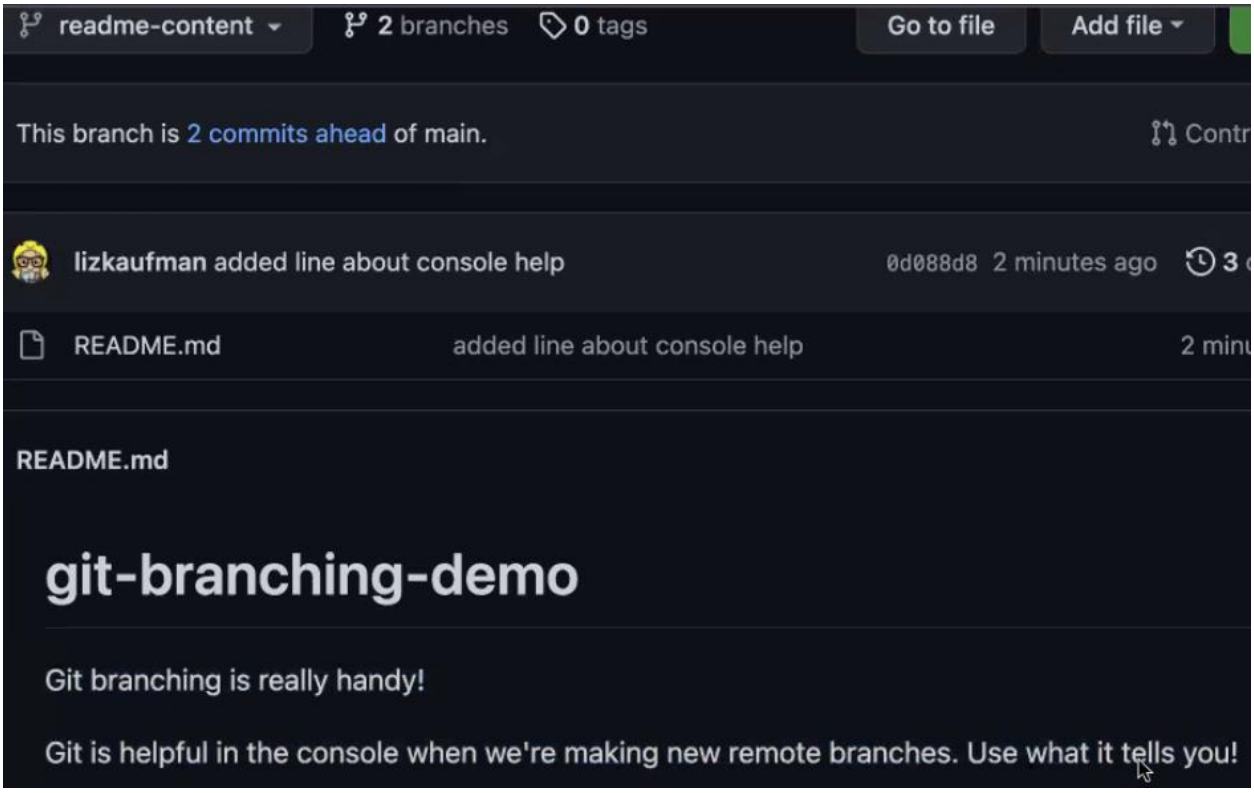
```
 * [new branch]      readme-content -> readme-content
Branch 'readme-content' set up to track remote branch 'readme-content' from 'origin'.
```

So, Git is helpful in the console, use what it tells you!

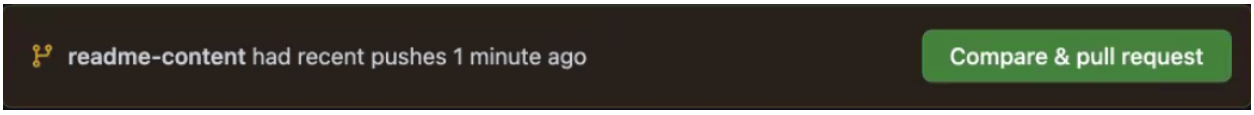
These 2 lines are only on the branch:

They disappear if you checkout to main! So, Git will update your files to reflect the current branch.

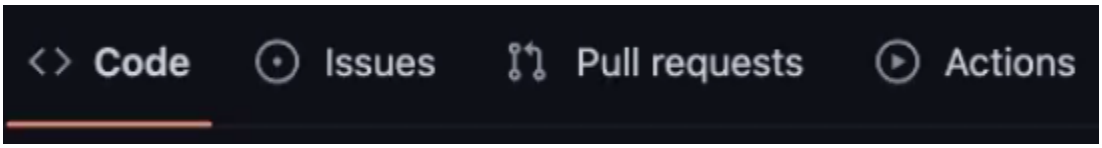
You can see branches on GitHub:



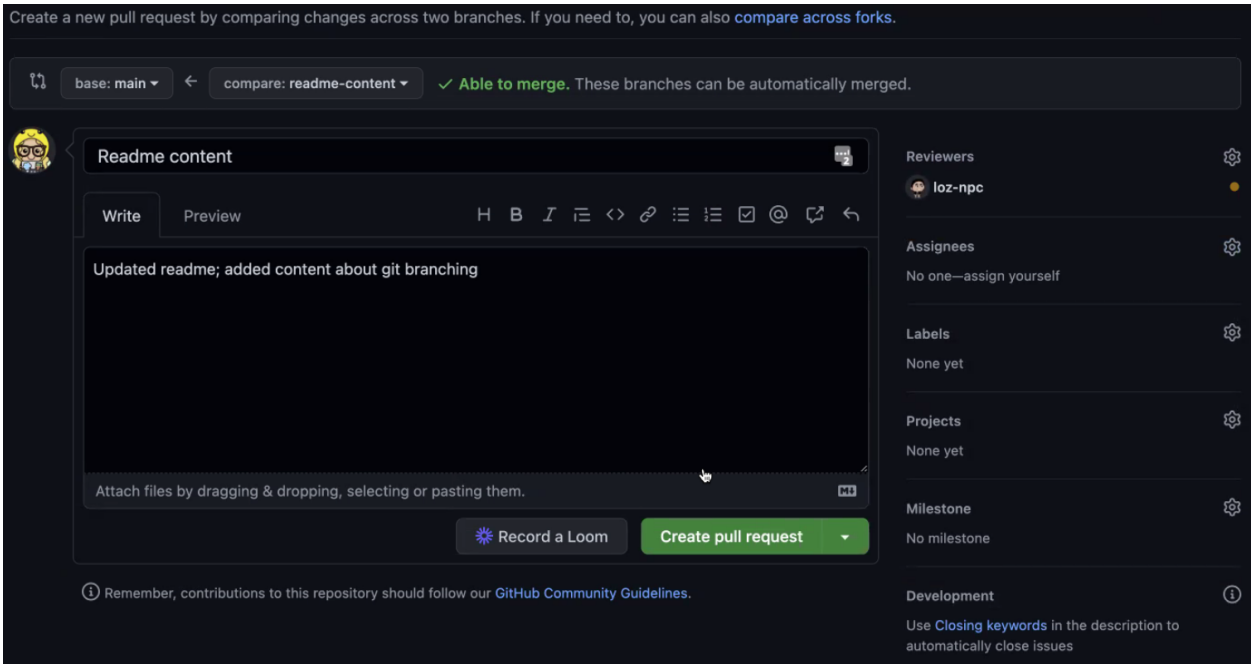
It gives you a nudge to do a pull request:



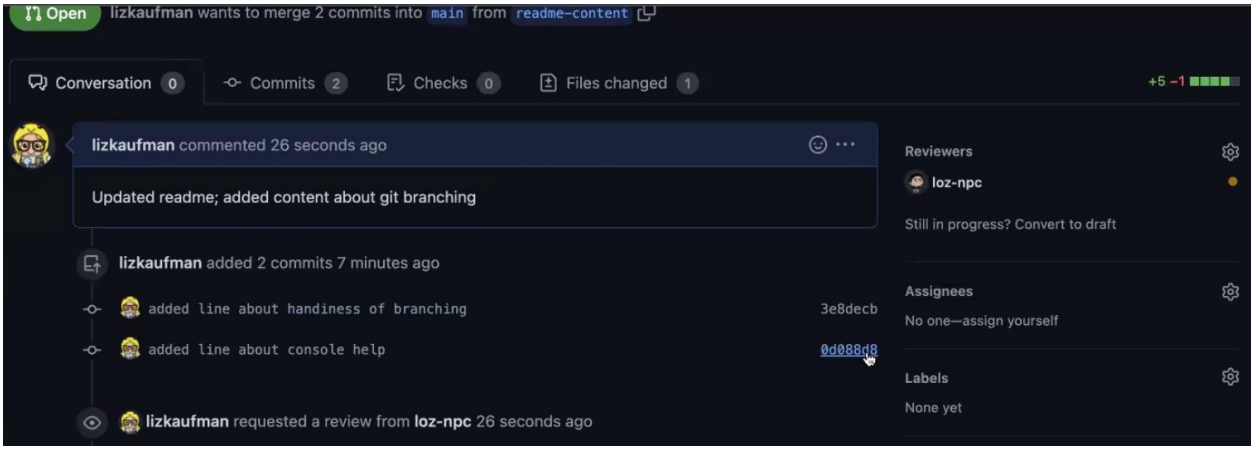
Or use pull requests tab at top of repo:



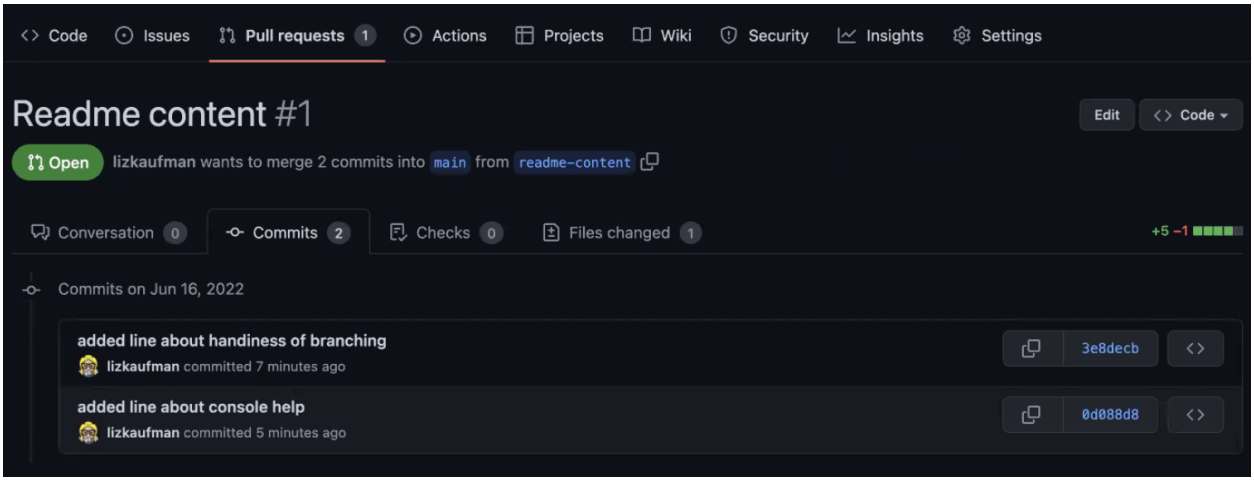
Give a description to go along with the commit messages. You can use the options on the right hand to specify people to review:



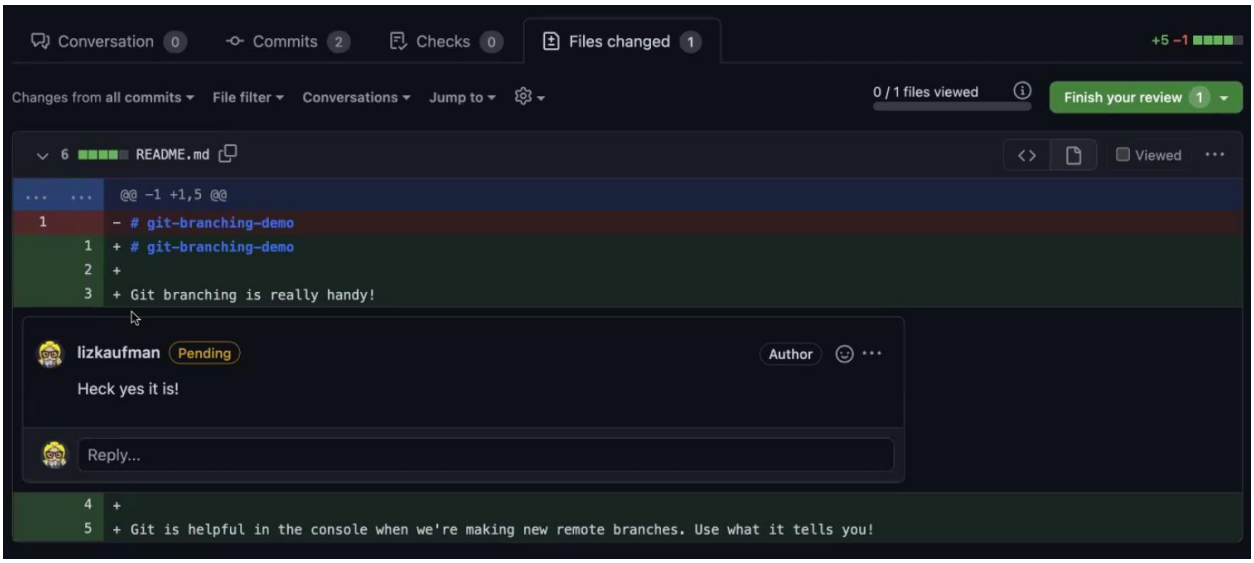
Once you create the pull request, it will show you the summary of it with a timeline so you can see everything that's happened:



You can view the commits and click on them to see the changes for each one :

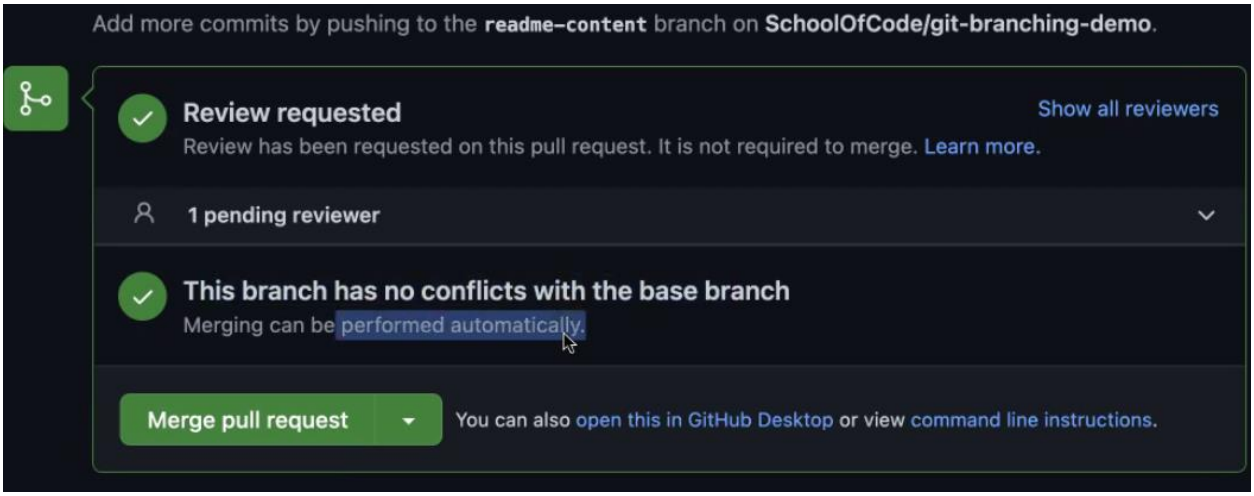


You can add comments to individual lines of code:

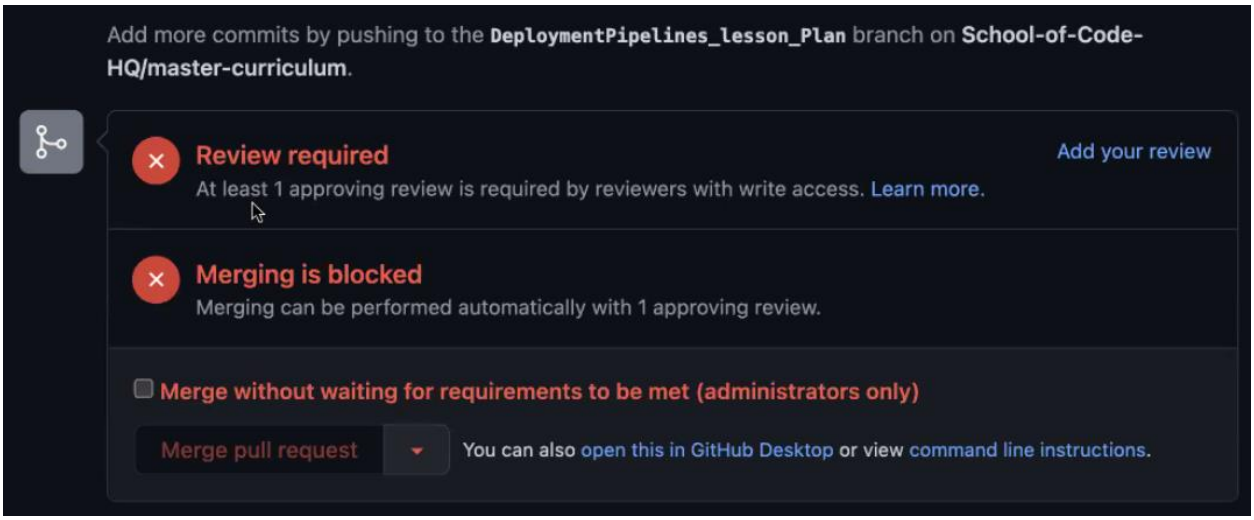


These are the most basic things being demoed

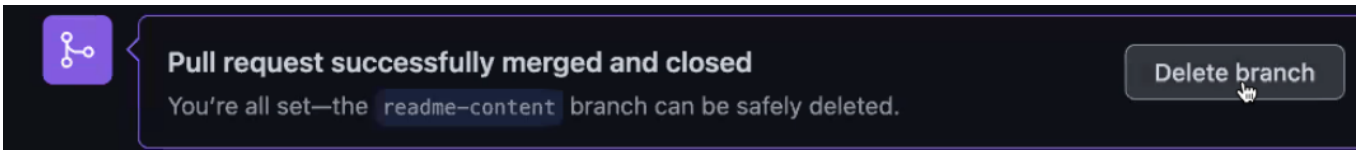
Once review complete (remember your reviewer may have to sign off on this before the option shows), you can merge the pull request:



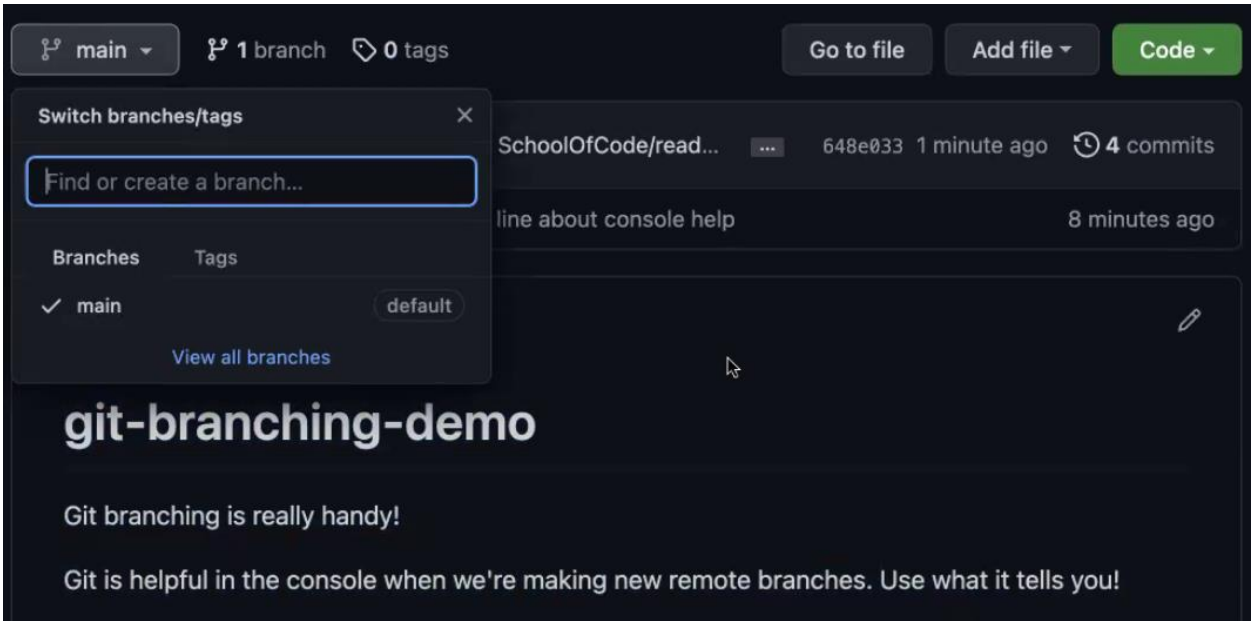
You may also need to deal with merge conflicts, but there aren't any in the above screenshot. Liz later showed an example of needing review:



Once it is merged, you can then delete the branch (this is reversible):



Merged in to main with no branches:



N.B. Remember to checkout back on to the main branch and pull the changes down!!

We haven't had time to cover **merge conflicts**, so will post resources and videos in Slack.

Afternoon workshop was working in groups to research aspects of agile – each group had to record a video and the list will be sent out. Link to list is here: https://docs.google.com/spreadsheets/d/1fBjfR0MgjMNRDa12VJEKfpEnqdNr6aFJ_tqpGtvs7FM/edit?usp=sharing