In order to request a code review, we must have work on a different branch, and then request that that branch be pulled into the main branch. At this point, when we make the pull request, we can add a reviewer who can look at and comment the code to make sure it is ok to pull that code into the main branch.

Note that ideally, a version-controlled workflow can be summarized as follows (for more detail, review Simon's slides):

1. You have a task to contribute to a repository
2. You create a branch to work on that task
3. You work on the branch, committing and pushing in an organized way
4. When you are satisfied with having completed the task, you create a pull request to merge your branch into the main branch
    a. You create the pull request, and in the process request a code review
    b. After the code has been reviewed and is approved, complete the merge
    c. Delete your branch to keep things tidy.

In section, I demonstrated the core activities of step 3 which are the heart of version control. Here, I will go over steps 2 and 4, which are the heart of collaboration.

1: Make a new branch and switch to it
First, in your terminal "cd" into the repository folder. Then, to create the new branch type:

`git branch` *`branch-name`*

```
(base) Lucys-MacBook-Air:are212-group3 lhackett$ git branch lucy-test
(base) Lucys-MacBook-Air:are212-group3 lhackett$ git branch
  lucy-test
* main
(base) Lucys-MacBook-Air:are212-group3 lhackett$
```

*The command `git branch` lists all available branches locally; the branch with * is the one you are currently on. Use this to check which branch you are on.*

Now we're going to go to this new branch, so that our changes are pushed to this branch instead of to the main branch. This is done by typing:

`git checkout` *`branch-name`*

```
(base) Lucys-MacBook-Air:are212-group3 lhackett$ git checkout lucy-test
Switched to branch 'lucy-test'
(base) Lucys-MacBook-Air:are212-group3 lhackett$
```

Now all changes you add - commit - push will be to the new branch, **not** to the main branch. This is nice if you are collaborating more closely with others to make changes without affecting them.

Step 2: Work on this branch, and push to it remotely

Now I'm going to work on my code, add - commit to this branch….

```
(base) Lucys-MacBook-Air:are212-group3 lhackett$ git add .
(base) Lucys-MacBook-Air:are212-group3 lhackett$ git commit -m "Some modified fi
gure titles"
[lucy-test a0938ac] Some modified figure titles
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Now to push to my lucy-test branch, I'm going to modify the push command so that git knows what branch to push to:

`git push --set-upstream origin` *`branch-name`*

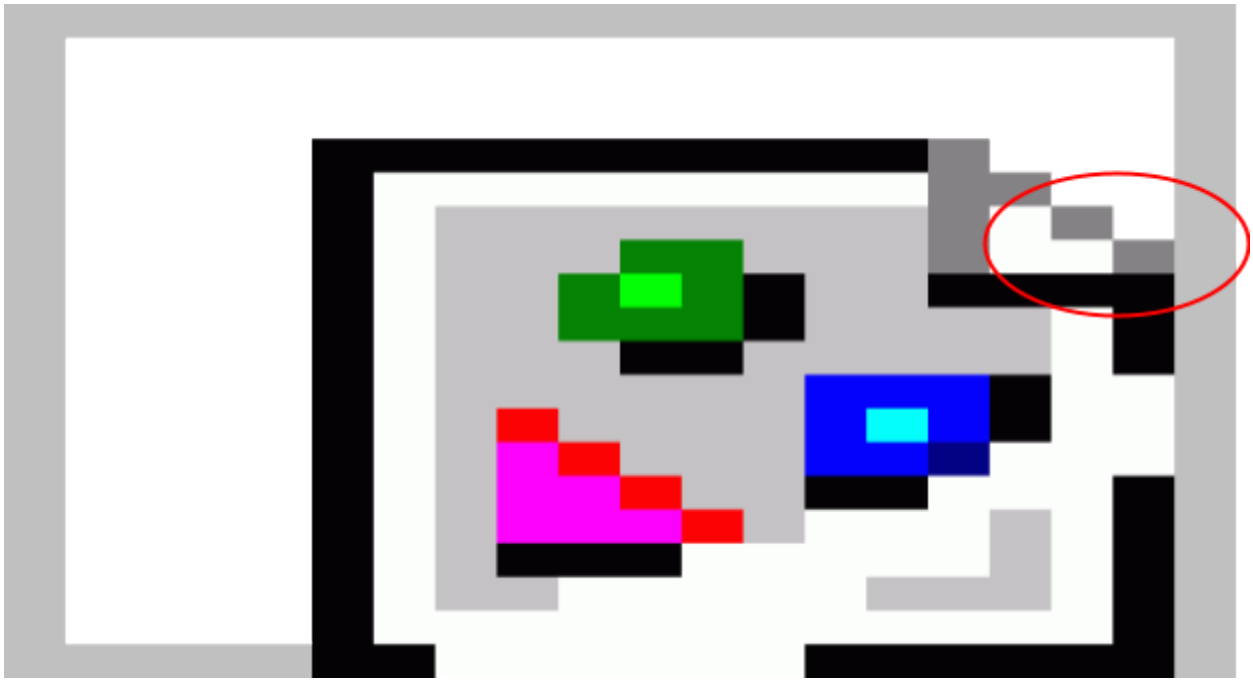
```
(base) Lucys-MacBook-Air:are212-group3 lhackett$ git push --set-upstream origin
lucy-test
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 465 bytes | 232.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'lucy-test' on GitHub by visiting:
remote:      https://github.com/lghackett/are212-group3/pull/new/lucy-test
remote:
To https://github.com/lghackett/are212-group3.git
 * [new branch]      lucy-test -> lucy-test
Branch 'lucy-test' set up to track remote branch 'lucy-test' from 'origin'.
```

Note that this can also be done in two separate steps, where first you set the new branch's remote upstream, then you push. Also, once you have run the option --set-upstream once, you will not have to do this again. Your future push commands can look like:

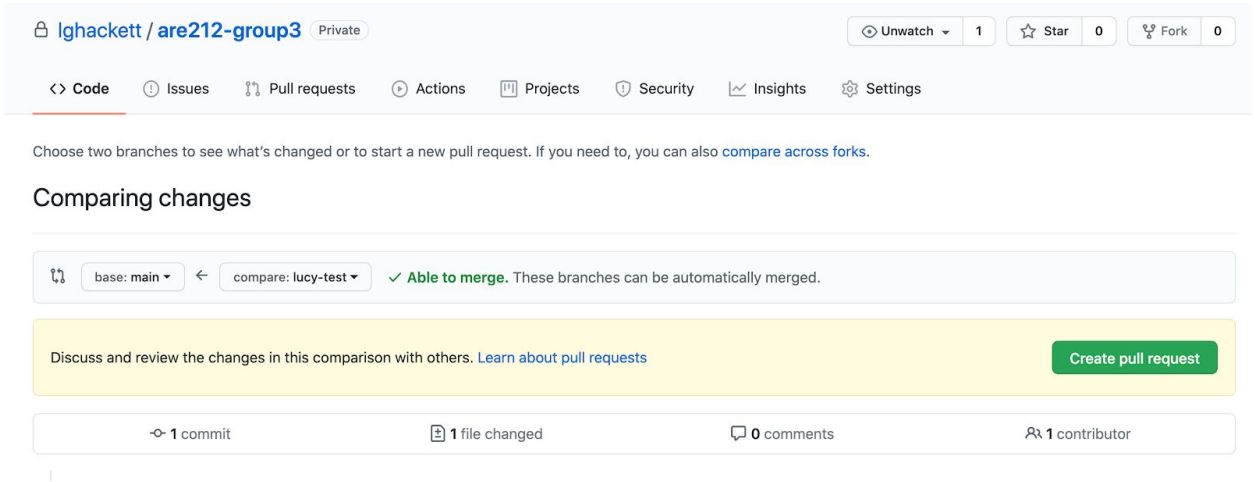
`git push`

Step 3: Create a pull request and add reviewers

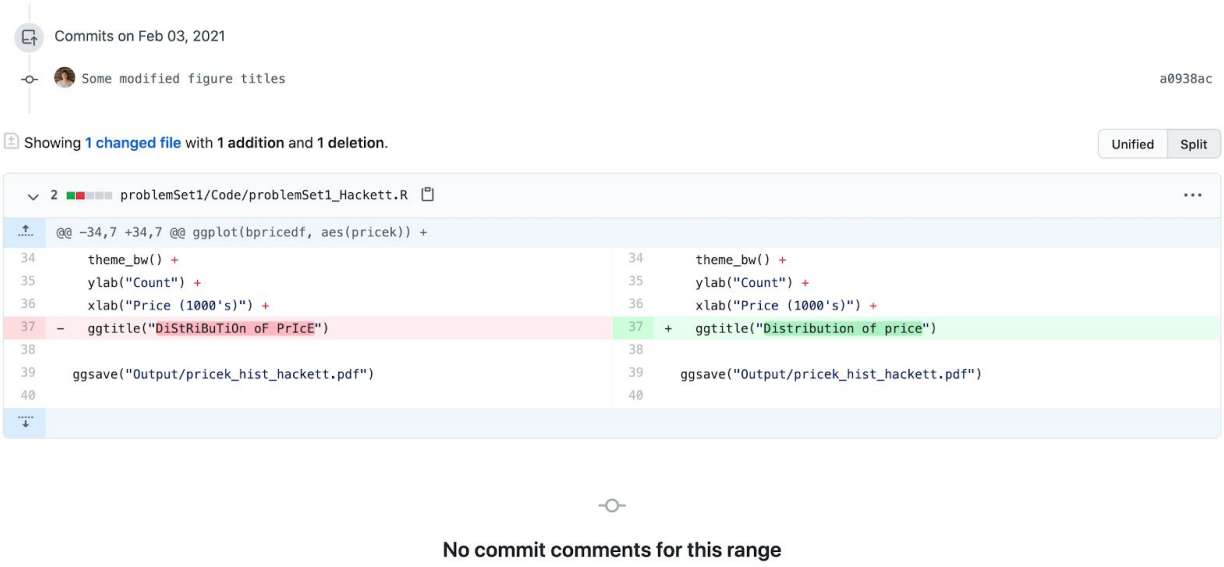
Now, I am going to request that my lucy-test branch be merged into the main branch. This is the stage at which I can add reviewers. So go to the repository on github.com, and click on the **pull request** tab, then click the green **Create new pull request** button.



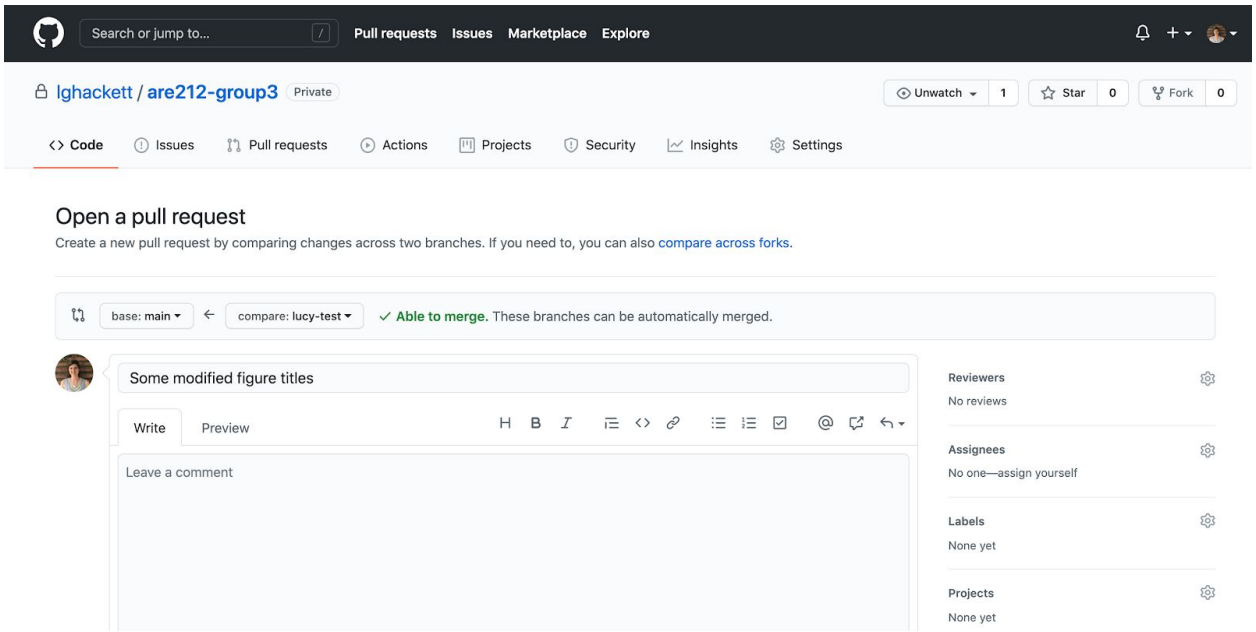
Now, at the top, you will see two selectable boxes with an arrow between them. You will leave the base as "main", because that is the branch we are pulling into. The branch that is being pulled is the "compare" branch, so for this field select the branch you created.



Scrolling down, you'll see git gives you a nice visualization of the differences between branches if the file is shared between branches.



Next, select **Create pull request**. Now you'll be able to give the request a title, and add comments to it if you'd like. Also, on the right-hand side of the page you'll see various tabs. Where it says **Reviewers**, click on the gear sign and select a reviewer.



Once this is done, click **Create pull request.**

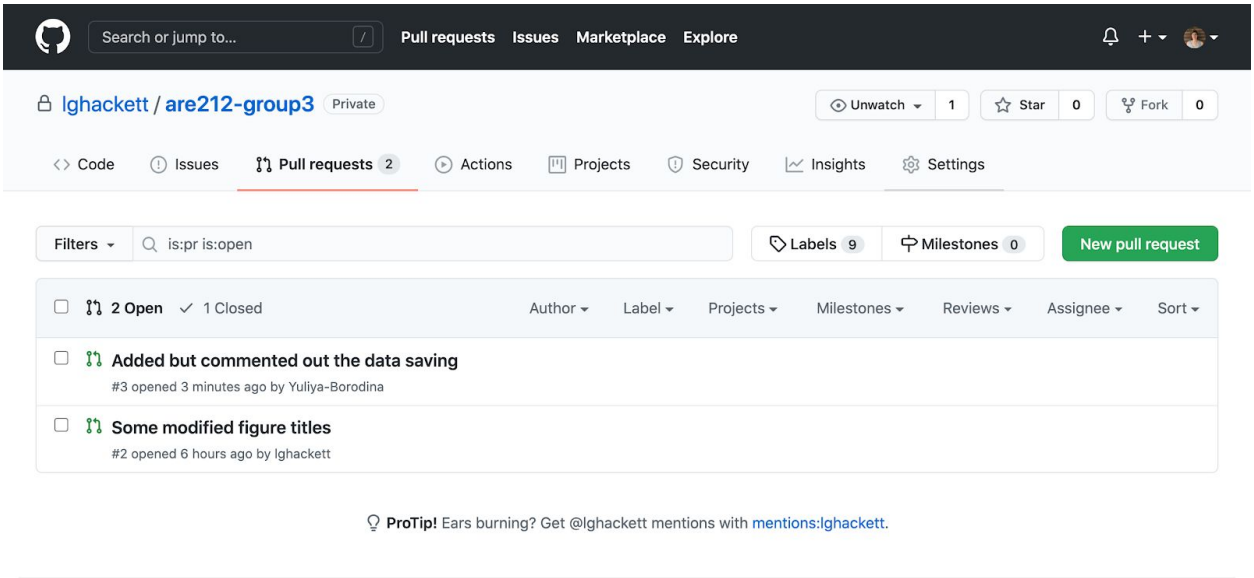
Note that even if you don't use the code review functionality of GitHub, this is the process for working on branches and merging back in. Ideally, anytime you work on something you would take out a branch to do so, and then request that that branch be merged in. So you're now officially a git rockstar.

In the next sections, I will first go through the review from the perspective of the reviewer, and finally discuss the end goal of this whole process: merging your branch into the main branch.

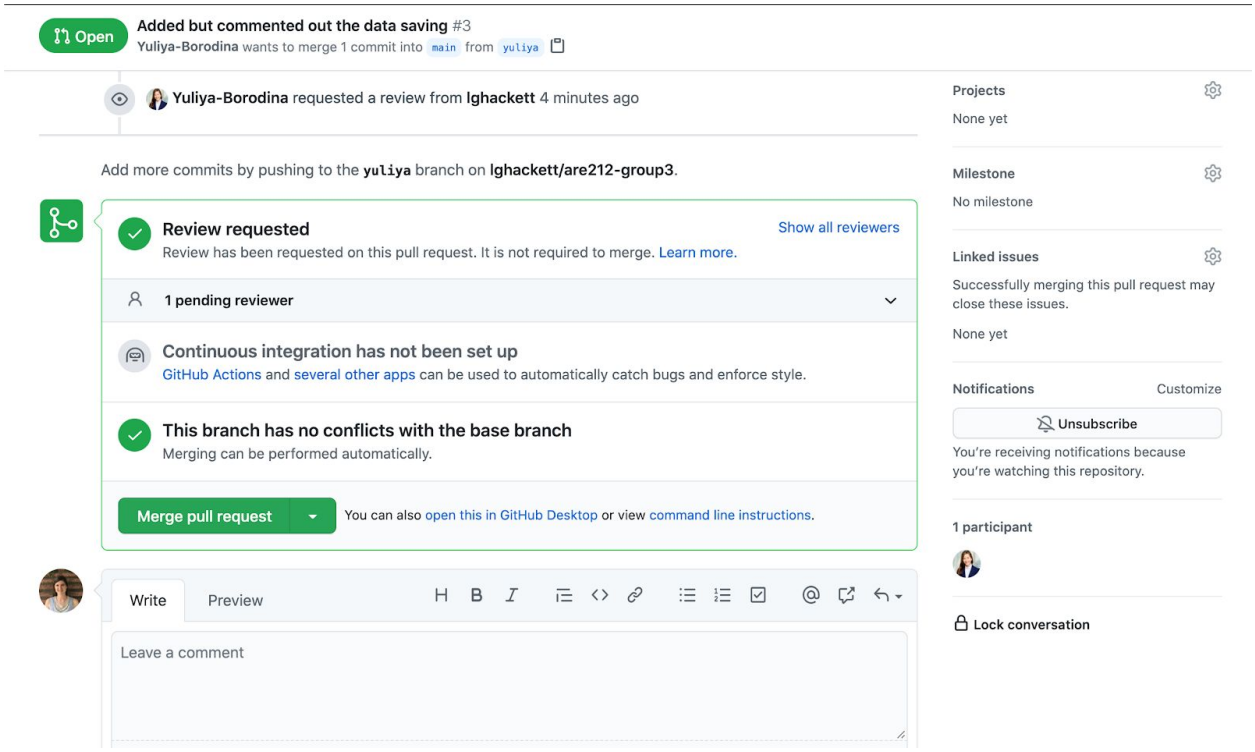
Reviewing code

What happens when someone requests a review from you?

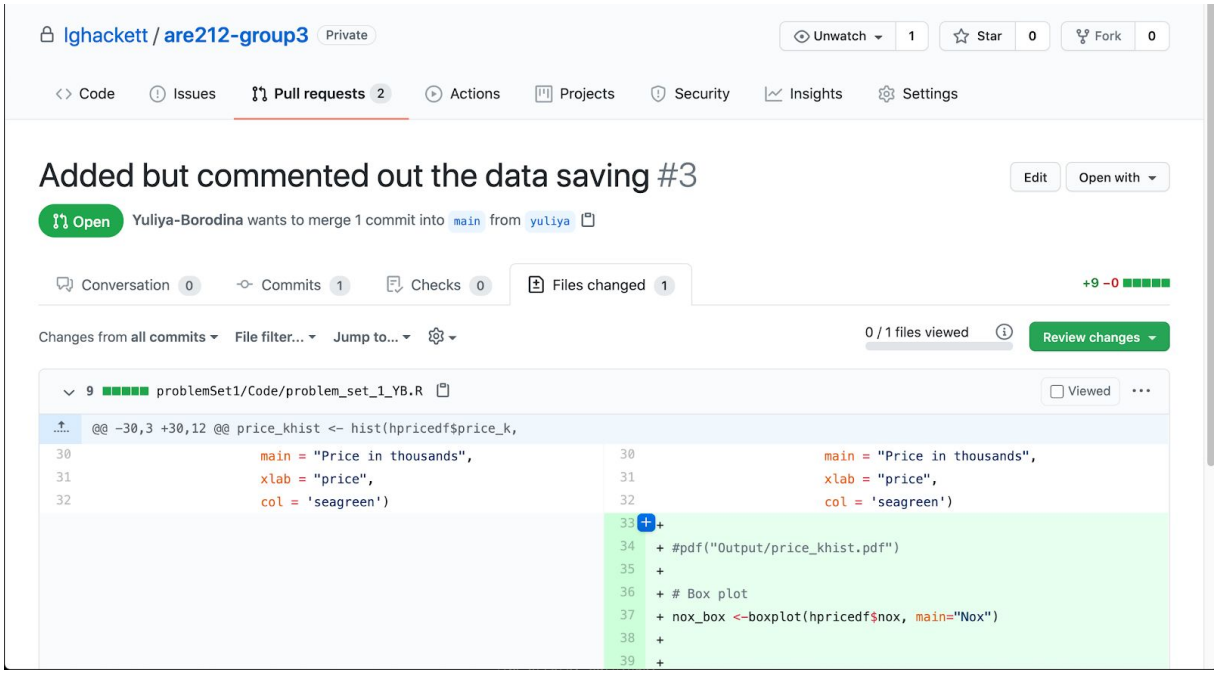
Step 1: Take a look at their pull request and comment
In the repository, go to the Pull requests tab. There you will see all open pull requests; find the one you are meant to review and click on its title.



On the next page, you'll see a kind of summary of the "conversation" about this pull request. Scrolling down, you will see "Review requested"; click on the down arrow on the right side of that tab to see who the review was requested from.



Scrolling back up, we're going to change from the "Conversation" tab to the **Files Changed** tab to get a look at their code. Here you will see all the changes that github can find from the last commit. If it is a totally new file, this will be the entire file.

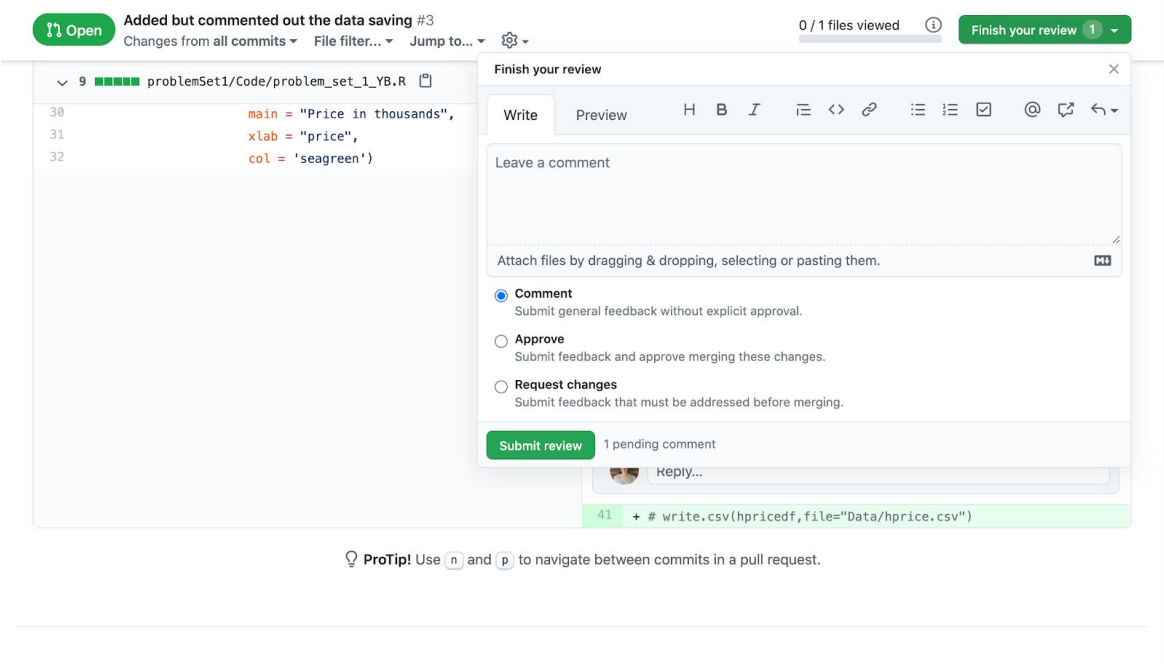


You can comment by hovering your mouse over specific lines of code, and clicking to start a comment. Here you can see that you have the option to "Add a single comment" or "Start a review". If you have just one comment, a single comment is fine. If you start a review, then all the comments you make will be bundled together in a single "Review", similar to how a push (can) bundle together several commits into one organized action.

In this tutorial, I chose to start a review for completeness.

When you are finished commenting, click the green **Finish your review** button at the top right. Then you have the option to add an overall message to the author of the code, and you can choose between a couple of different options.

In our case, "approving" the code isn't such an important concept, but you can choose this if everything looks good to you for immediate merging. Otherwise, I'd suggest the "Comment" option. I'm going to go with comment since in this iteration, just my general comments were what was solicited. Finally, click **Submit review.**



Now you will be directed back to the Conversations tab, where you can see your comments integrated into the conversation.

Final step: Merging

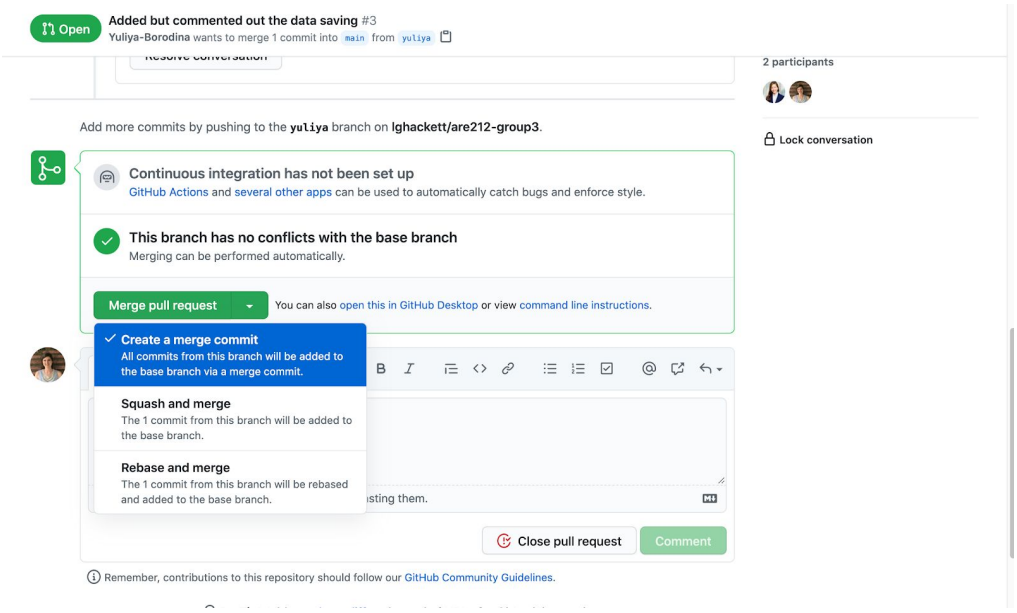
It depends on your team's organization and workflow who will do the actual merging; for our purposes, the author of the code can probably do their own merge. If you contribute to someone else's project online (who you may or may not know) they will have the exclusive right to merge.

For our purposes:

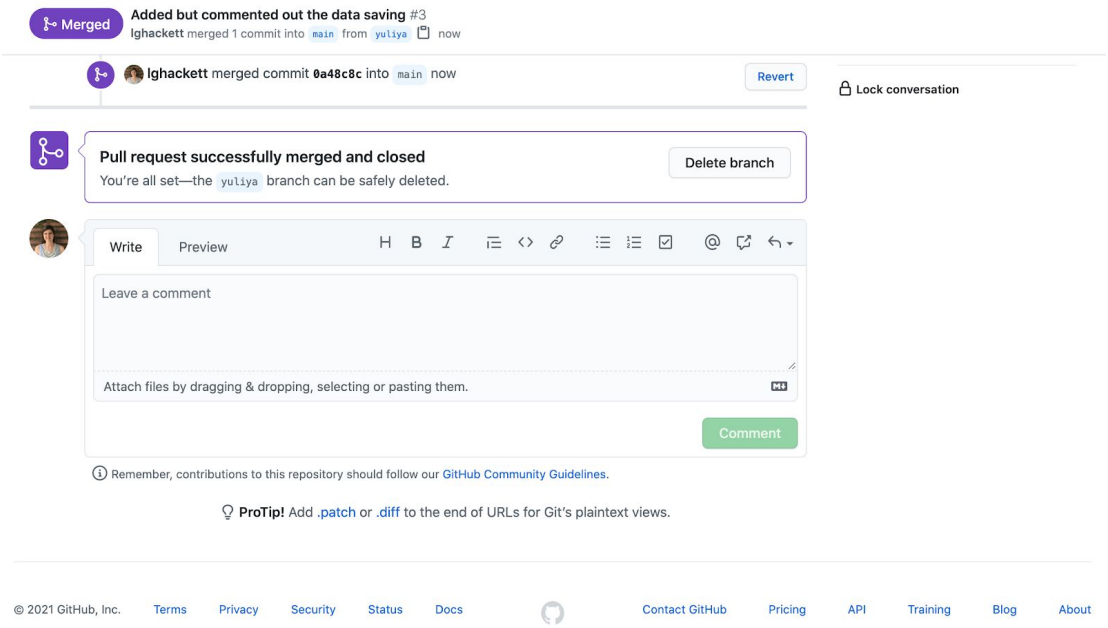
Let the author of the code review your comments, then they can decide to either:

1. Merge the pull request into the main branch if everything looks good and there are no conflicts.[1]
2. Make edits to the branch you wanted to merge and commit these to that branch. Then, when you push to the branch (in this example, *lucy-test*), the commits will appear as part of the pull request; essentially, you will have updated the pull request. At this point if everything looks good, you can merge the branch into main.
3. Cancel the pull request altogether. This is usually not necessary; try and pull request only if you're decently sure your code is ready.

[1] Note that in merge there are several ways to merge; "Squash and merge" will collapse all the commits you made to this branch into one, creating a "cleaner" git history, but that loses information. This is preferable if the branch has been developed for a relatively long time with many commits that may confuse the overall workflow of the project. If this is not the case, feel free to use "merge commit", which will preserve all the previous commit history. In the Google internal version control system, all merges are done through a version of debasing, which I will not discuss here.



Once the branch is merged in, the finished pull request will look like this:



Note the option to **Delete branch;** it is good housekeeping to delete the branch so that there are no "ghost branches" hanging around that no longer serve a purpose. When you make more changes later, you can (and should) always make a new branch for that specific purpose.

If you click "Delete branch", you have only deleted the branch *remotely* in GitHub. As a final step, delete the branch *locally* in your computer first switching to the main branch in the console:

And now, to delete the branch, type:

```
git branch -d branch-name
```