**Tutorial 10**

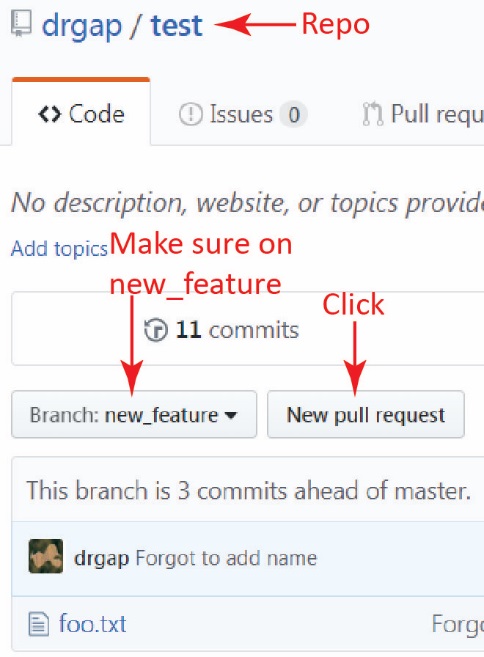
***Pull Requests***

The workflow I suggest is:

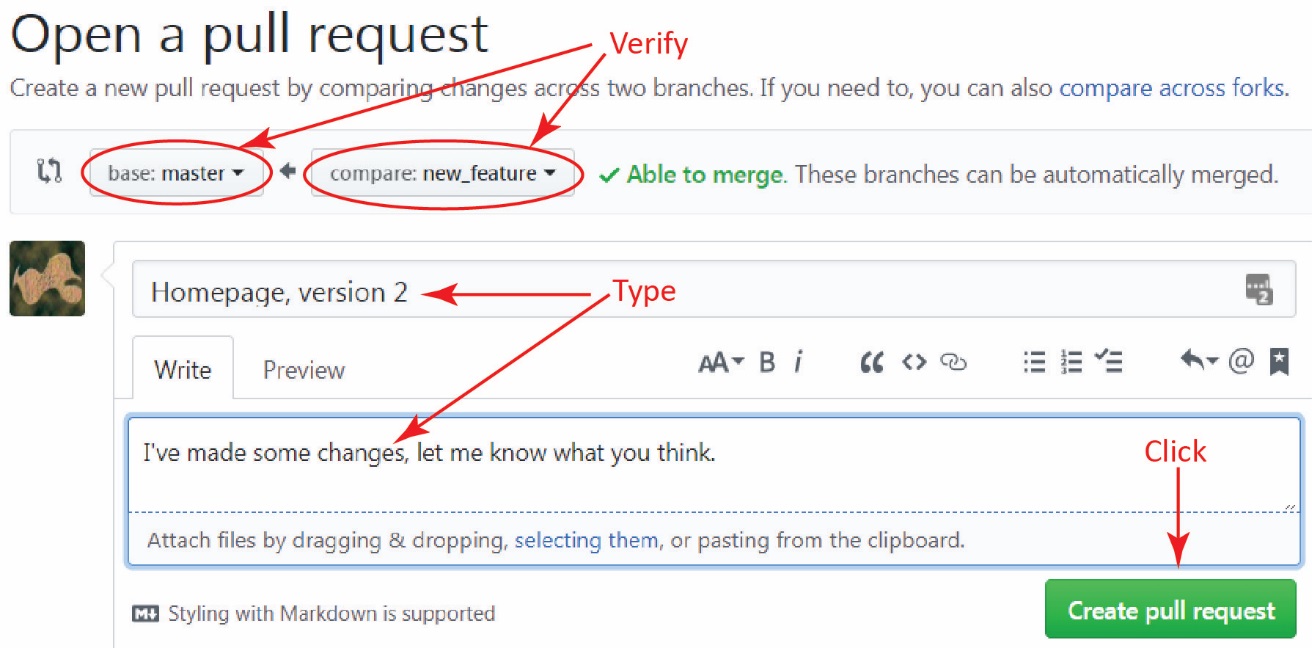
1. Assume we have working code in the *master* branch on GitHub.
2. You create your own branch. Perhaps you should use your name as each team member will have their own branch.
3. Clone or pull your changes locally.
4. Make changes, stage, commit – all locally. Make more changes, stage, commit.
5. Push to GitHub.
6. Open a pull request. A pull request is a mechanism that allows your team members to review your code before you merge it with master.
7. After your pull request has been reviewed, merge it with master.

This tutorial illustrates these steps and continues from Tutorial 9.

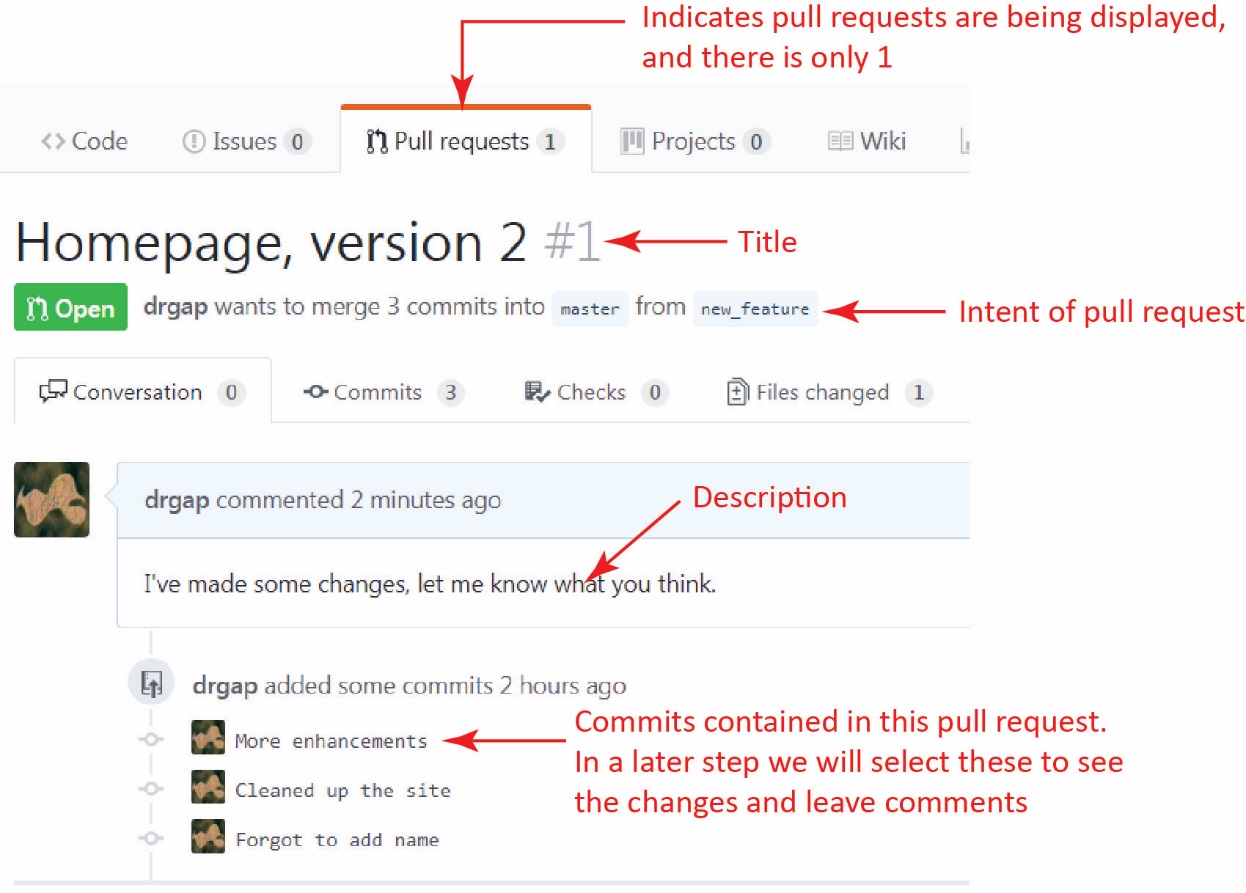
**Steps to Complete**

1. On your local machine, delete everything from *foo.txt* except the first line (which might be: “Change from local clone.”). Stage and commit.
2. On your local machine, add your name to the end of *foo.txt*. Stage and commit. (We are creating a longer commit history to use later)
3. Push your changes to the *new\_feature* branch on GitHub.
4. (Read, no action required) Anytime two branches are different on GitHub, we can open a *pull request*. A pull request is a comparison between two branches, a base branch and a branch that is different. A pull request also contains all commits that make the base branch differ from the other branch. Once a pull request is created, collaborators can review the proposed changes (commits) and leave comments. After all collaborators are satisfied with the changes, you can merge the changes into the base (*master*) branch.
5. Display the homepage for your *test* repository, make sure the active branch is *new\_feature,* and then click *New pull request* as shown on the right.

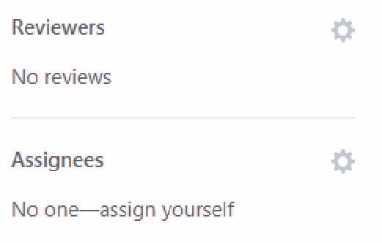
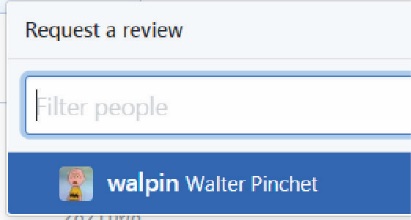
1. On the pull request dialog, do the following (as shown in the figure below):
2. Verify that *master* and *new\_feature* are selected in the drop-downs. Change if necessary. Notice the direction of the arrow. That is indicating that we are examining how *new\_feature* is different from *master*.
3. Type a title and description
4. Click *Create pull request*



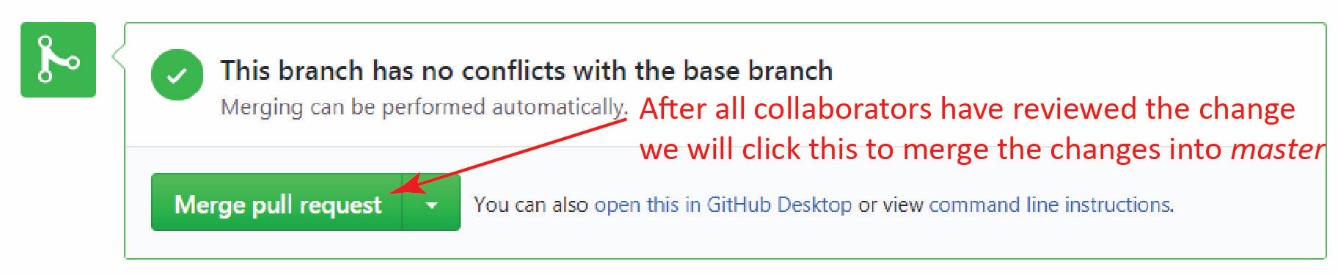
1. (Read, no action required) The resulting display is rather large, so we examine it in pieces.
2. All pull requests are being displayed (there is only 1) as well as the title, description, and associated commits.



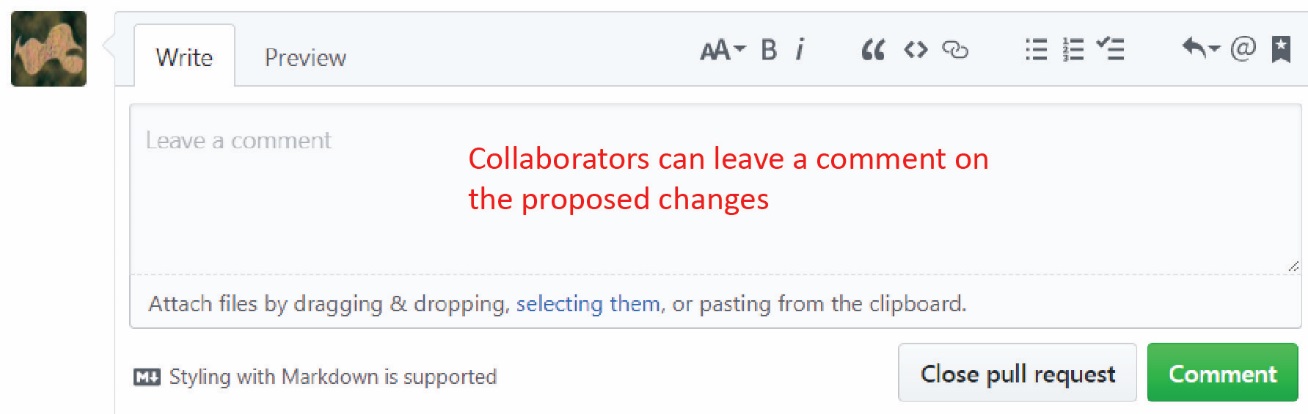
1. In the upper-right of the pull request, you’ll see a place to assign *Reviewers* and *Assignees* (see figure on the left below)*.* Here, you can assign other members of your team to review your pull request. To do this the other use must be a collaborator for the repo. I’ve made *walpin* a collaborator on my *drgap* repository and assigned him as a reviewer. We will see how this comes into play shortly. You will not assign a reviewer for this tutorial, but you will for you project work.

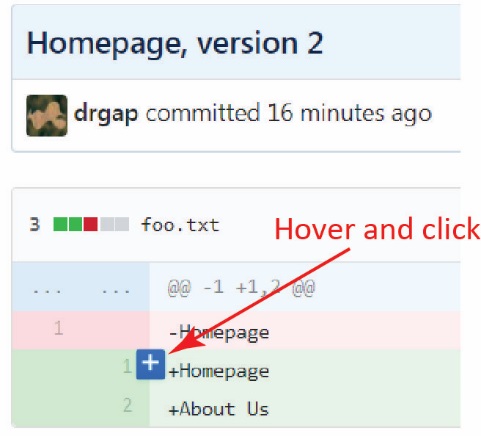
 

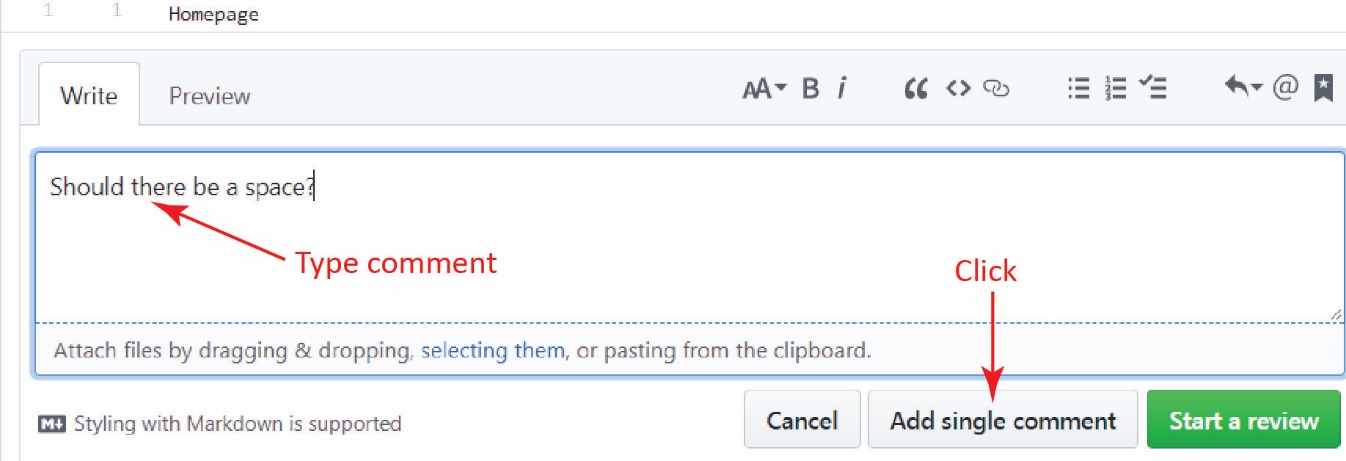
1. Next, we see a section that we will use **later** to merge the *new\_feature* branch into *master*.

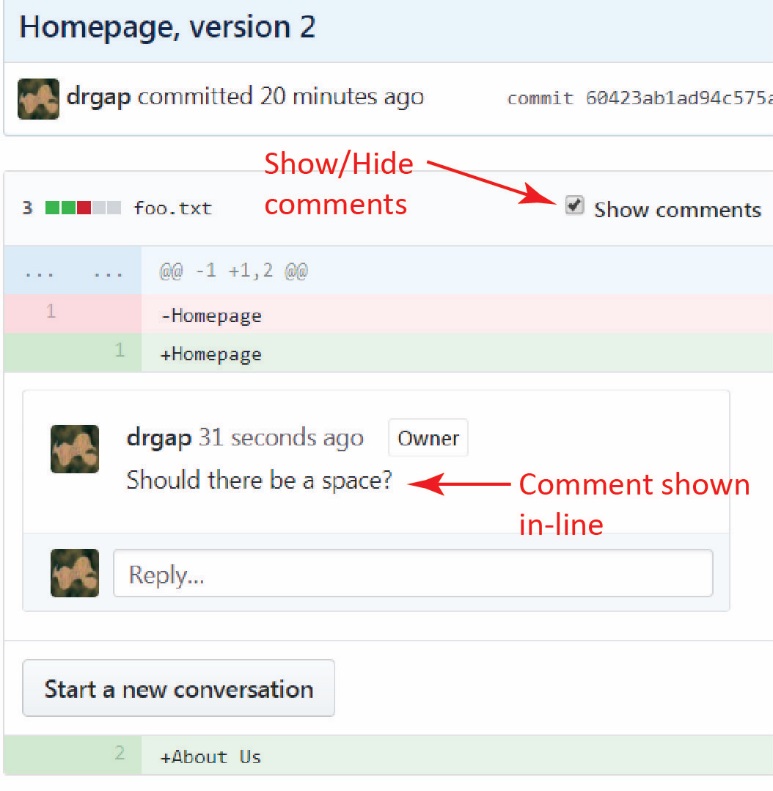
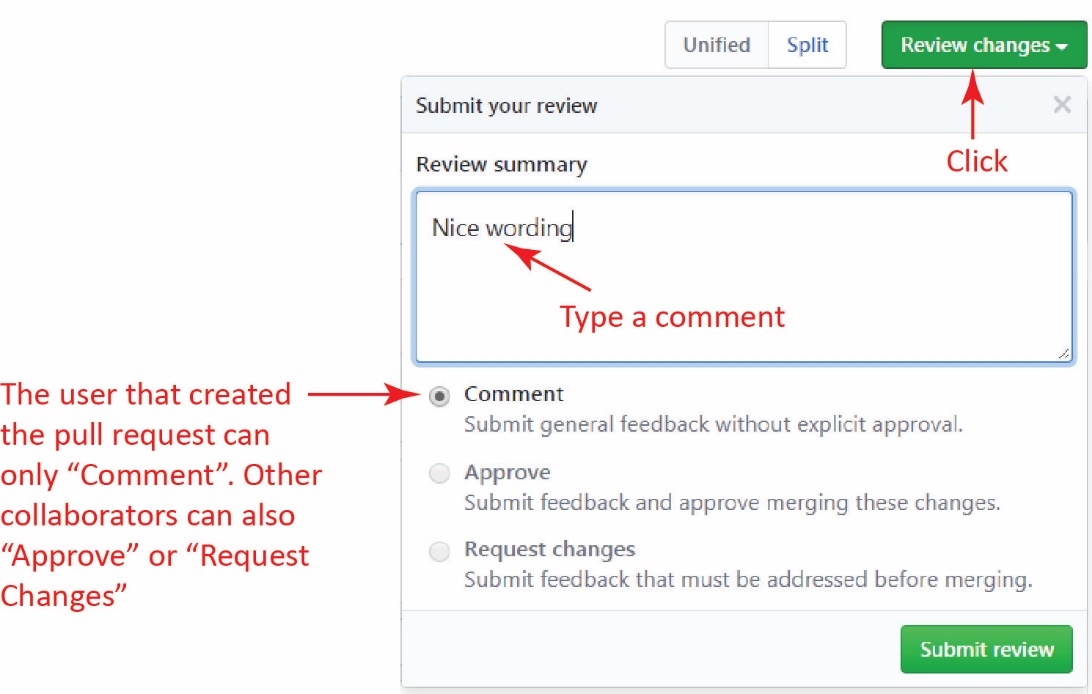


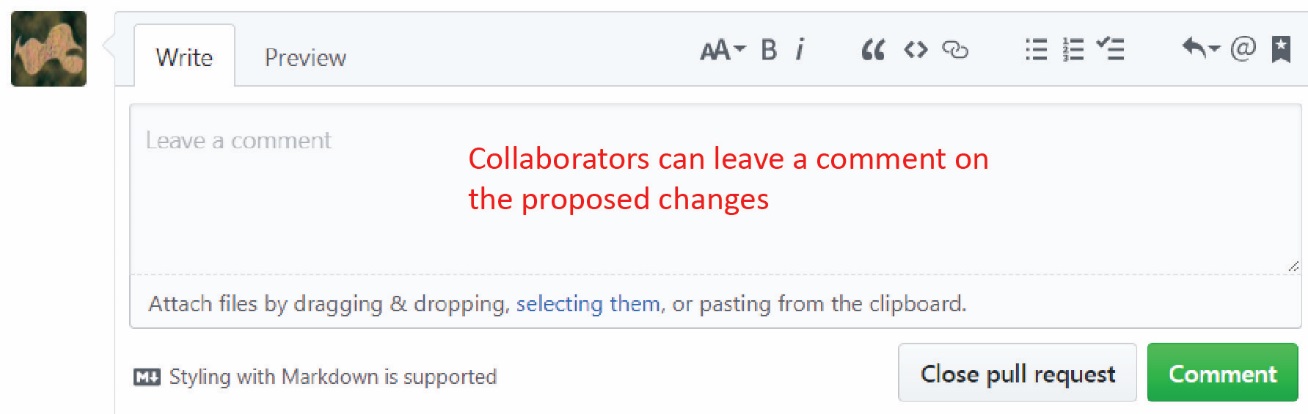
1. Towards the bottom we see a section where we can write a comment on the proposed changes about the entire pull request.



1. The commits listed in the pull request are sorted oldest to newest. You can view and comment on any of them. Do the following:
2. Click one of the commits in this pull request
3. Hover your mouse over one of the lines shown and a “+” will appear. Click it.
4. As shown below, type a comment about a particular line of code. Do this now and and then click *Add single comment*.



1. The result is shown on the right. As you can see the comments are shown in-line and they can be turned off.
2. Click *Review changes* on the right side of the window, as shown on the right, type a comment and then click *Submit review.*
3. Towards the bottom of the pull request display (as we saw earlier), we see a section where we can write a comment on the proposed changes about the entire pull request. Type something there now and press *Comment.*

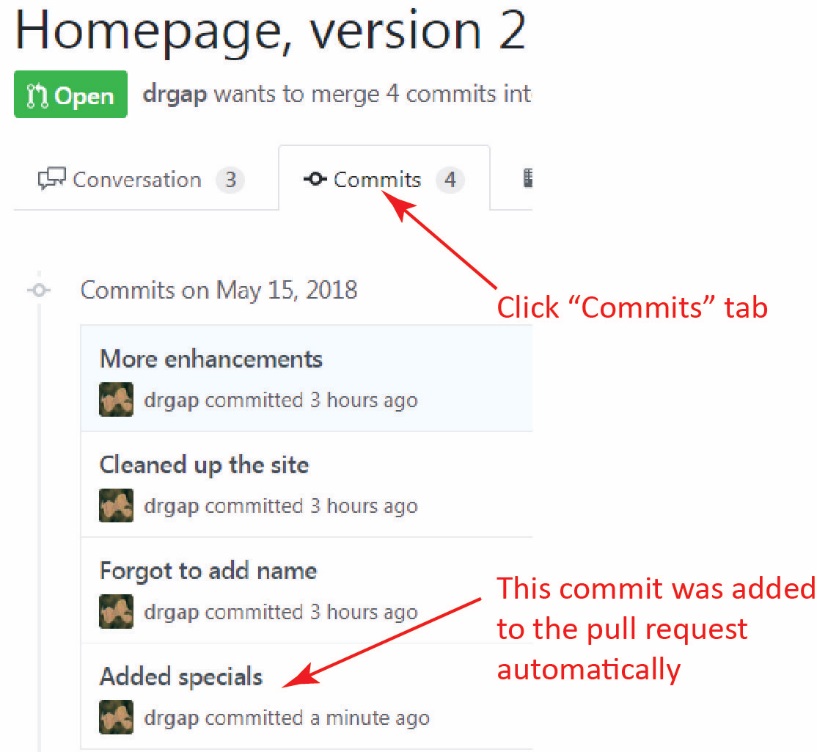


1. (Read, no action required). In summary, you can leave a comment about:
2. A single line of code in a commit.
3. A particular commit
4. The whole pull request
5. Go back to your local repository, add “Specials” to the end of *foo.txt,* stage and commit and then push to the *ver2* remote branch.

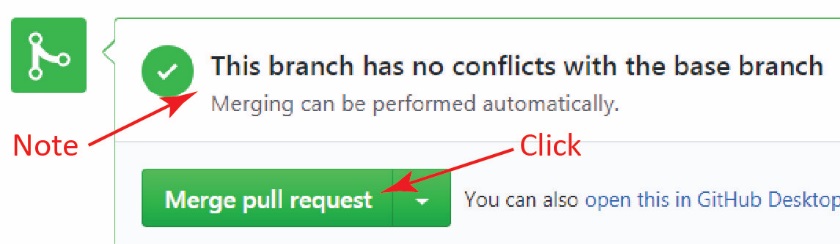
**λ**  notepad foo.txt

**λ**  git commit –a -m "Added specials"

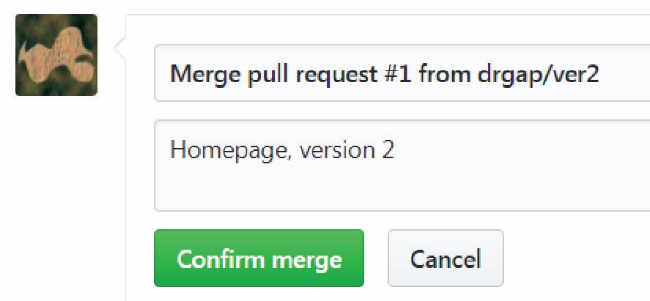
**λ**  git push -u origin new\_feature

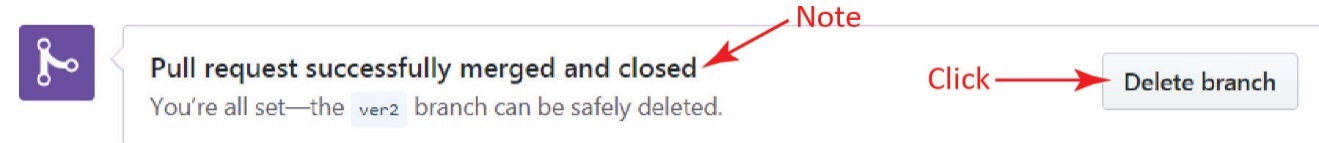
1. On GitHub, display the pull request and click the “Commits” tab. Notice that the push added the most recent commit to this pull request.

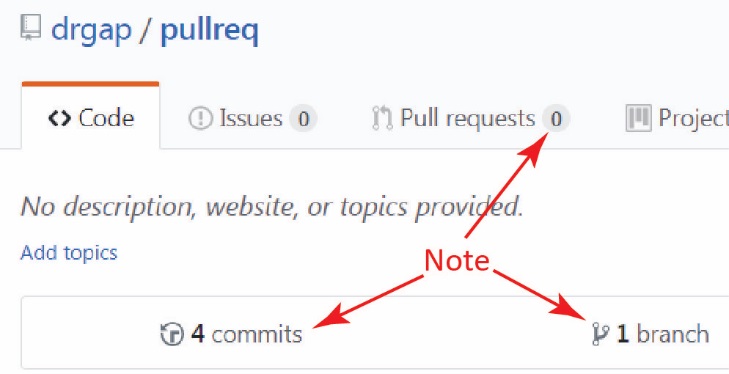
At this point a collaborator (or you) could drill into the commit and comment/review or return to the “Conversation” tab and do similar things there.

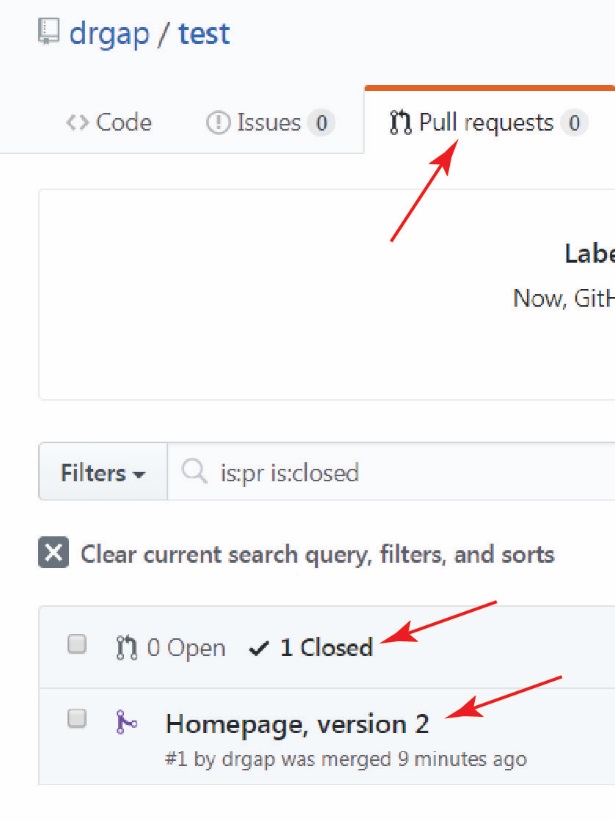
1. Click the “Conversation” tab and scroll down to the Merge section as shown on the right. Click *Merge pull request.*

Note: the dropdown has 3 options for merging. We are using the top (default) one which keeps the commits separate. The second option combines all the commits into one. The third option does a *rebase*. Remember, you should do a “rebase & merge” if there are conflicts so as to leave *master* in working order.

1. The view is updated as shown on the right. Click *Confirm merge*.
2. The view is updated as shown below. At this point *master* and *new\_feature* are the same. Many times we would delete the *new\_feature* branch, although there could be reason to keep it or reuse it. Here, we will delete it, so click *Delete branch.*



1. Return to the default view for the repository by clicking the name of the repository toward the upper-left, *test*. (the figure on the right shows *pullreq* instead of *test*)Note the following:
2. There is only 1 branch now, *master*
3. There have been 13 total commits (for mine, figure on right).
4. There are no open pull requests. However, if we click the *Pull requests* tab and then click *Closed* we can see the pull request we just merged and closed. (see figure below on next page)



1. **Do the following:**
2. Make a screen shot similar to the one in Step 18c above. Make sure it shows your User ID and the closed pull request.
3. Place the image in the *HW VCS* document in the appropriate place.
4. The image should easily readable without zooming in or out.