Git / GitHub

Git is a distributed version control system to manage the files of your project.

*Installing Git*

<https://git-scm.com/>

<https://git-scm.com/downloads>

From this page, you can download the latest version of git. This comes with command line tools, but git is also integrated into many development environments. In addition there are several GUI frontends for Windows, Mac, and Linux. We’ll work our way up to these:

<https://git-scm.com/downloads/guis>

1. Install Git

2. Create a GitHub account

3. Create a local repository

Repositories are where you will store all the files of your project. Open a terminal, and navigate to the folder where you plan to create the repository. The git program is run at the command line, and it has a variety of arguments. Try:

> git

What are some of the different options?

Now try:

> git init

This will create a git repository in the folder/directory that you are in. Try examining the files in the directory now – anything there?

4. Add a file to the project. Open a text editor and create a simple program in the language of your choice. Save that file to the same folder. Now that there has been a change to the folder, when you run another particular git command, git will make the appropriate changes to the repository. First run:

> git status

This will give you all the info git has on the repository. Notice the file you created – it knows it is there, but it will not add it to the repository until you tell it to do so. This requires making a *commit* – these are records of changes since the last time you committed changes to the repo.

There is also the staging environment – this is where are files are added before they can be committed. Try:

> git add YourFileName

And rerun the status check:

> git status

Great, the file is in the staging area! Now commit:

> git commit -m “Who said commitment is hard?”

The -m flag is important – it is a message that describes the changes/additions/etc that have been made. Try to make it as meaningful as possible – your future self will thank you. Also, note the output from the commit.

5. Branch your code. Branches allow you to try something new (for example, an experimental feature) without affecting the main branch of development. If you are happy with what you have created, you can merge the branch with main (or master). Try:

> git checkout -b TestFeature

The -b flag indicates you want to create a new branch. This not only creates a branch, but it places you working on the branch (you are no longer on master branch). To check out the branches, try:

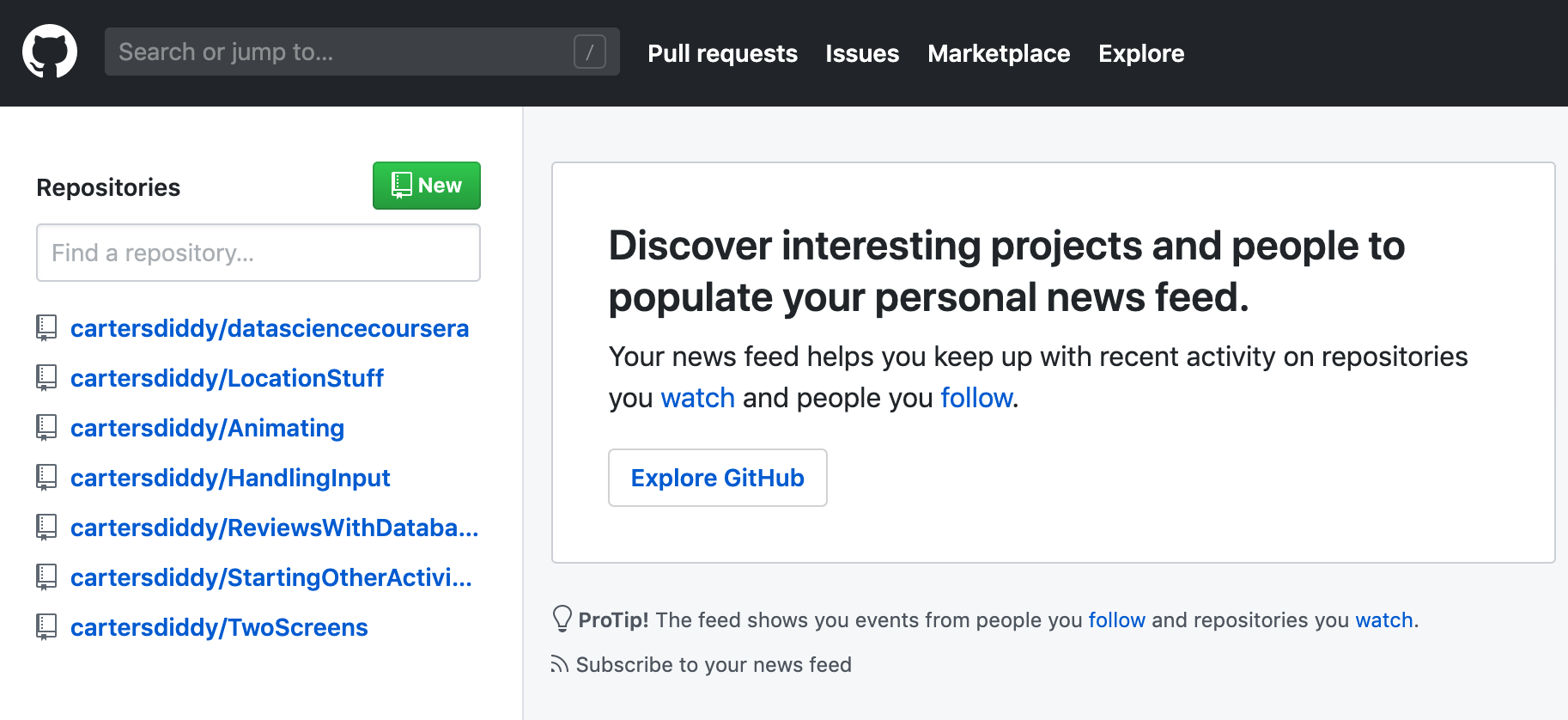
> git branch

To switch back to master:

> git checkout master

Whatever branch you are on is where changes will be made when you make commits. Other branches will not see those changes until you have merged them.

6. Create a repository on GitHub. GitHub allows you to store your code non-locally so it can be shared by multiple people and on different machines. It is easy to create a repository – log in to your account and click the New button:



Because we have created the repository locally already, we will push it to this repository from the command line.

> git remote add origin https://github.com/cartersdiddy/Tester.git

> git push -u origin master

The first line associates your GitHub repository with the local one. The second line pushes the master branch out to this GitGub repo. Check out your project on GitHub – is your file there?

Now make some changes in your other branch. Be sure that you are currently using that branch and either add a new file or make a change to the existing one. Then:

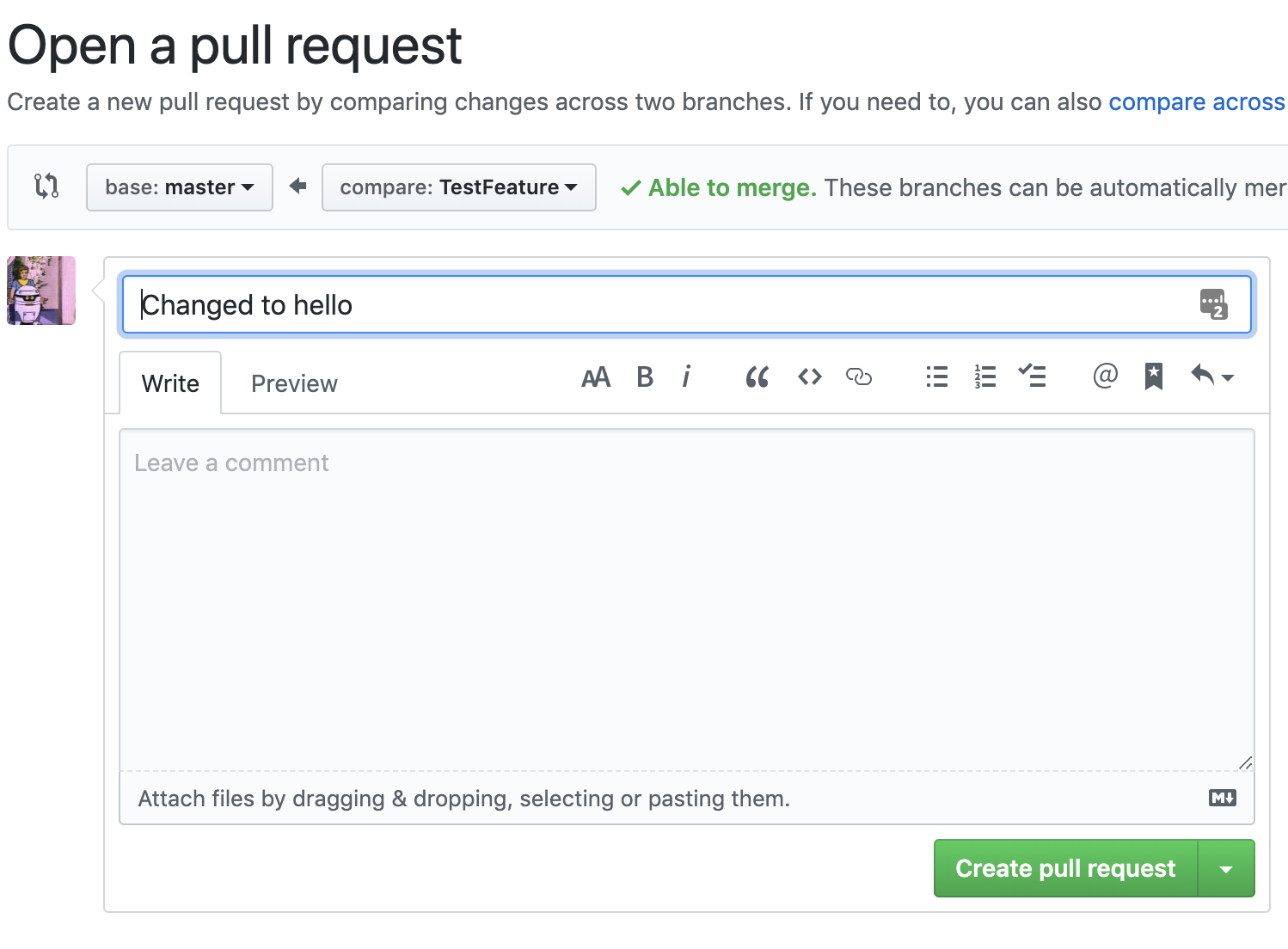
> git add YourFileName

> git commit -m “trying something out”

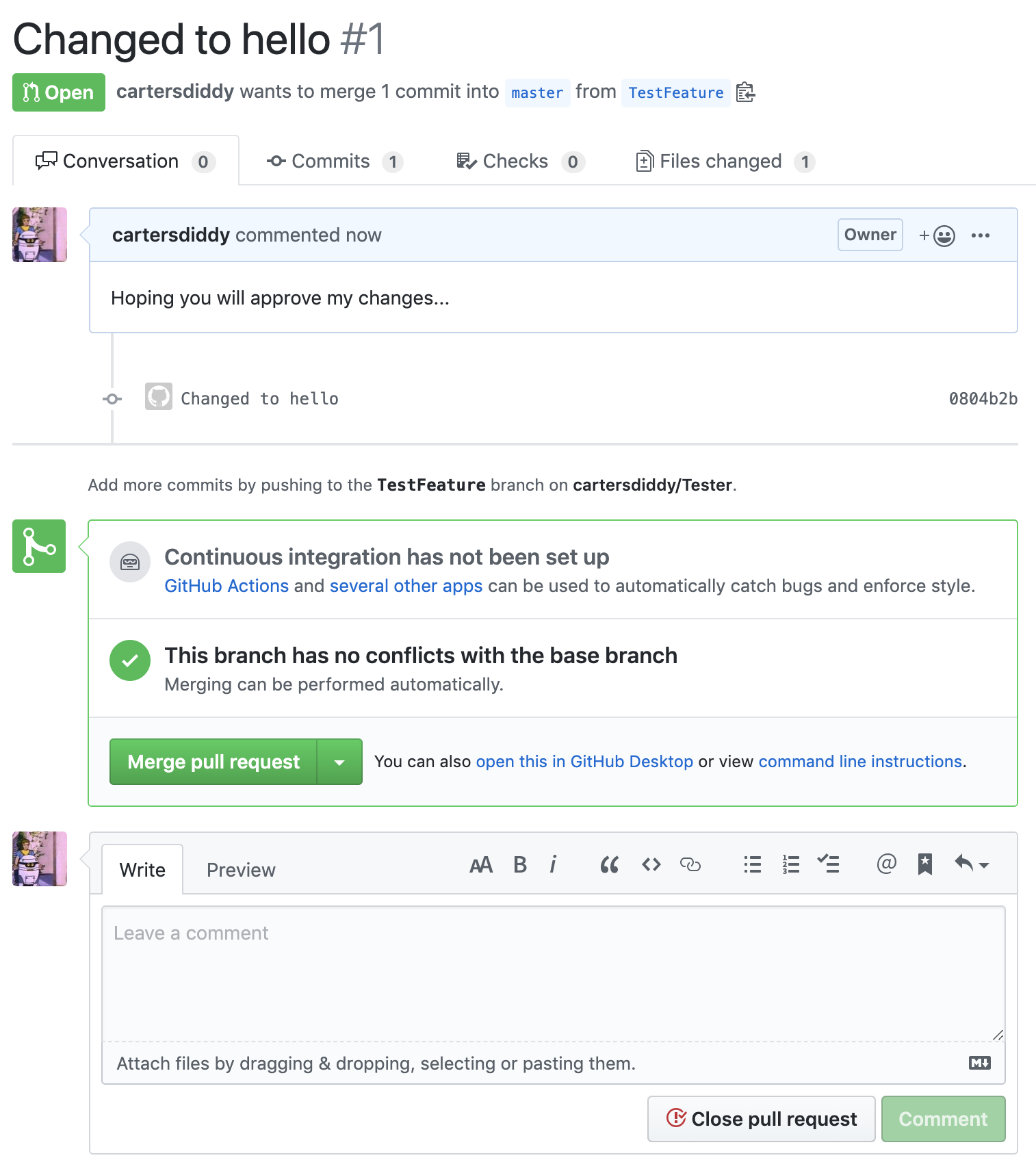
> git push origin ExperimentalBranchName

Now check out your GitHub project again. Notice anything different?

8. Pull from GitHub. So we can push things out to GitHub, but how do get things down from it? Click the Compare & pull request button. You should see something like:



You can add messages, notes, files, etc, here. Click Create pull request, and you are rewarded with this screen:



9. Merge branches. The big green Merge pull request button means this is a “mergeable” action – there are no conflicts. Click the button! (But also note that this can be done in GitHub Desktop as well as via command line). You can delete the branch with the button if you would like (a good idea generally).

Now go check out the master branch. What is in the file now?

10. Bring the changes back to your machine:

> git pull origin master

> git log

Git