



# Using Git & GitHub

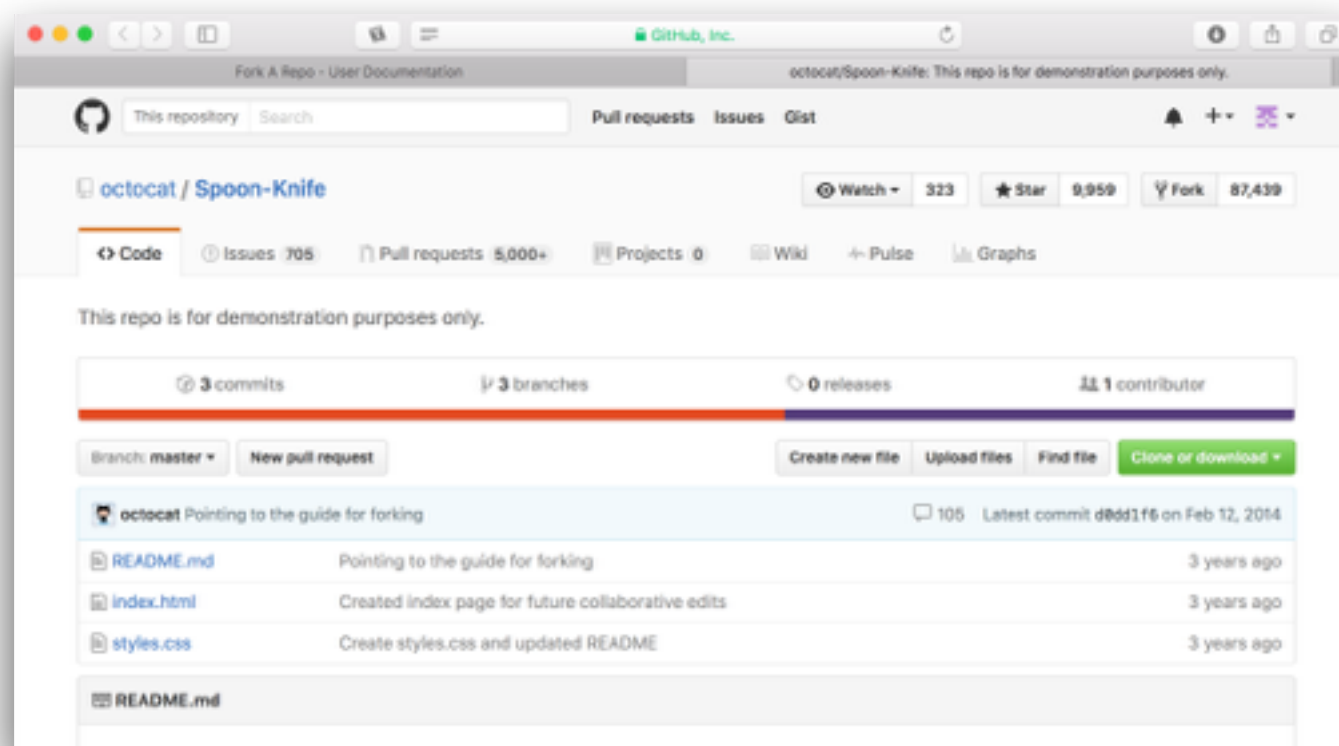
`/* an introduction */`

# What Are They?

- **Git** is a software development tool that tracks development of code and uses repositories
- **GitHub** is an online repository platform that allows you to store your code on their servers
- Provides a good framework for...
  - sharing your code
  - collaborating with others on code
  - updating or editing code

# About Repositories

- **Repositories** (“repos”) are where you store your code and files related to it.
- This can include programs, documentation, directories, and even pictures!

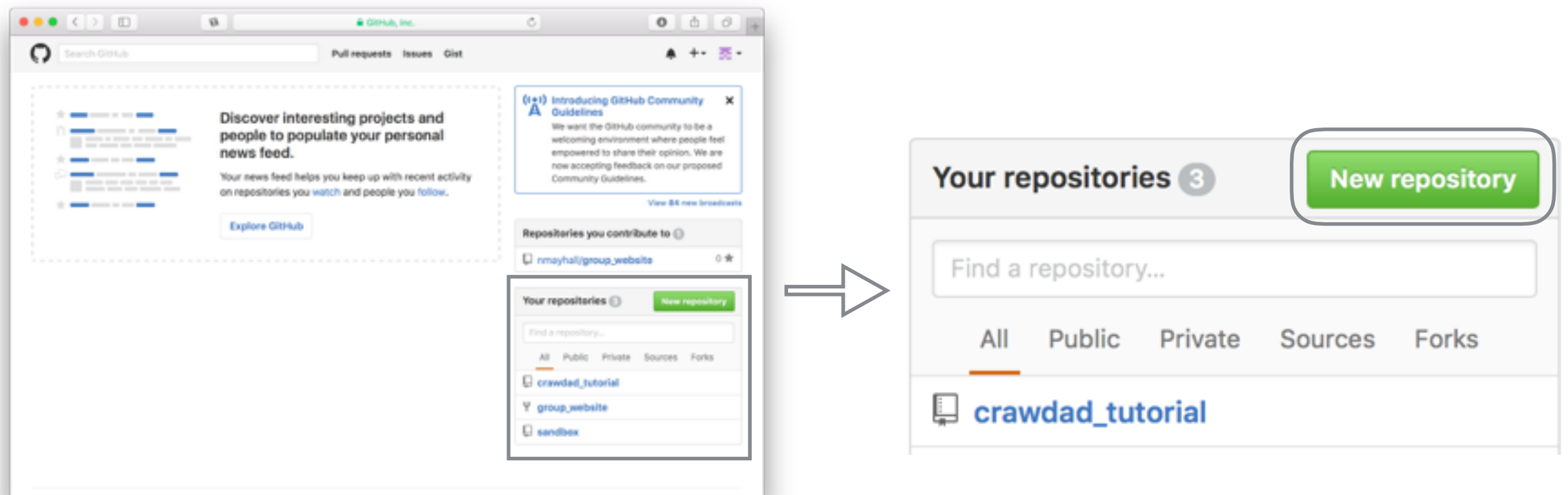


# Creating Repositories

- Using Git (command line):

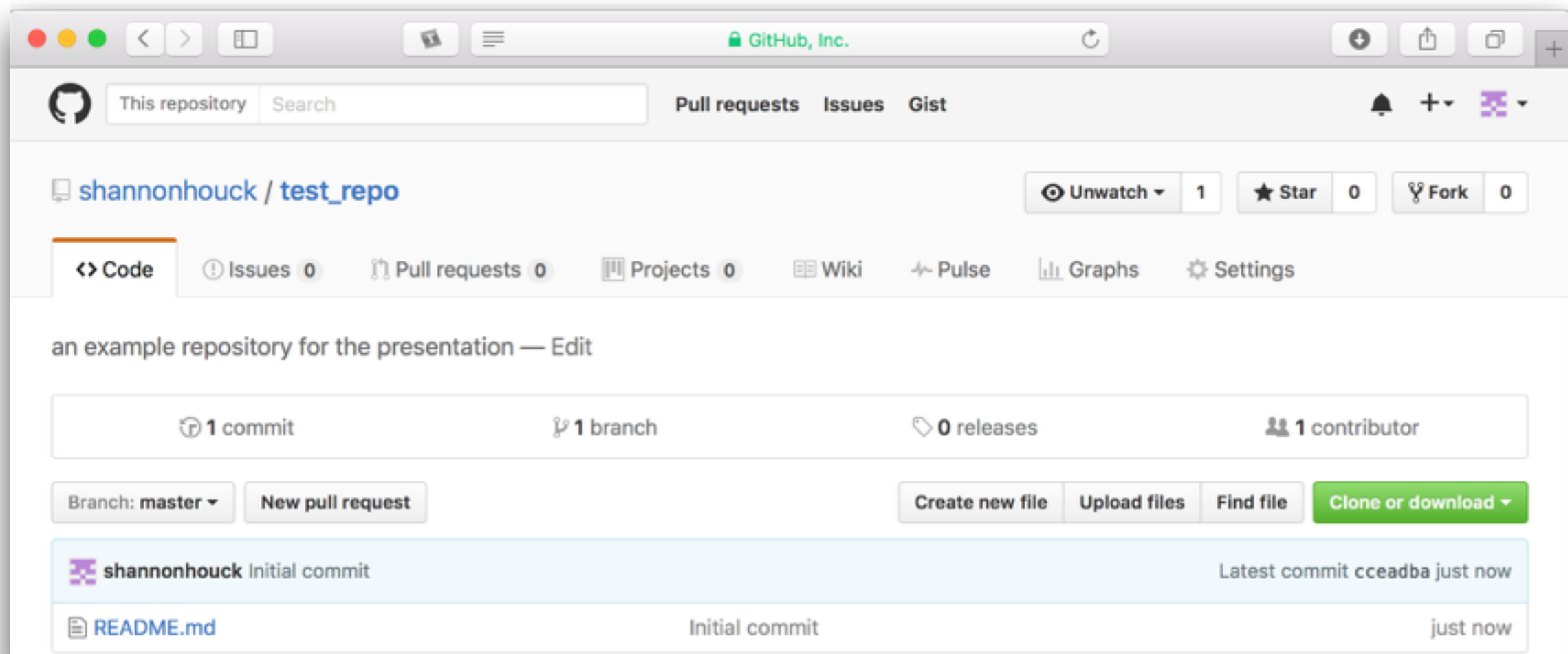
```
$ git init [directory path]
```

- Using the GitHub website:



# Repository (Results)

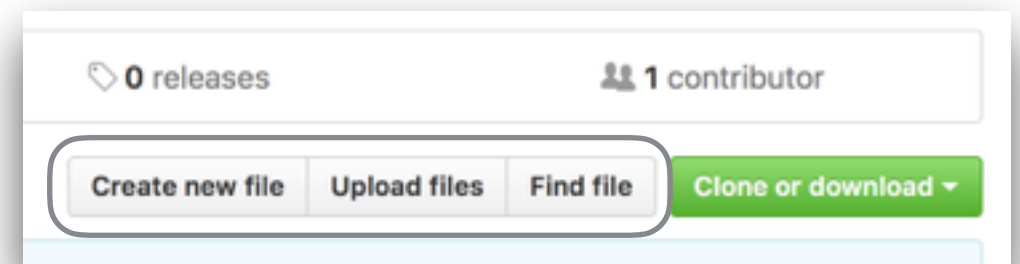
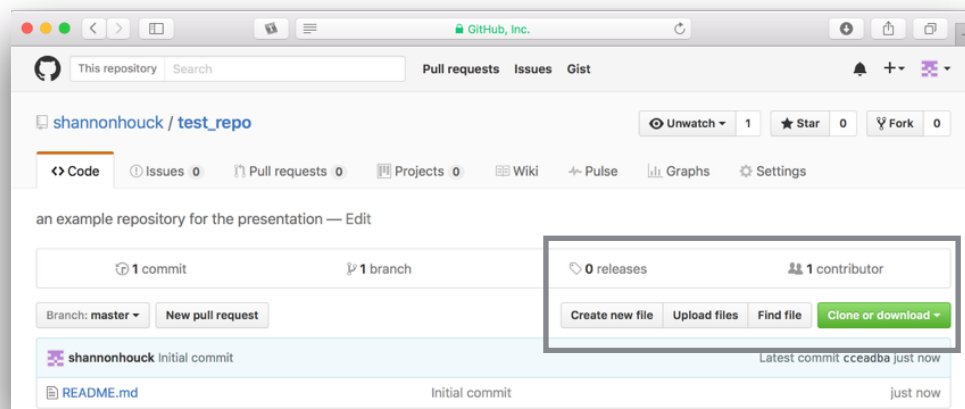
```
shannonhouck — -bash — 80x46
Shannons-MacBook-Pro:~ shannonhouck$ git init test_repo
Initialized empty Git repository in /Users/shannonhouck/test_repo/.git/
Shannons-MacBook-Pro:~ shannonhouck$
```



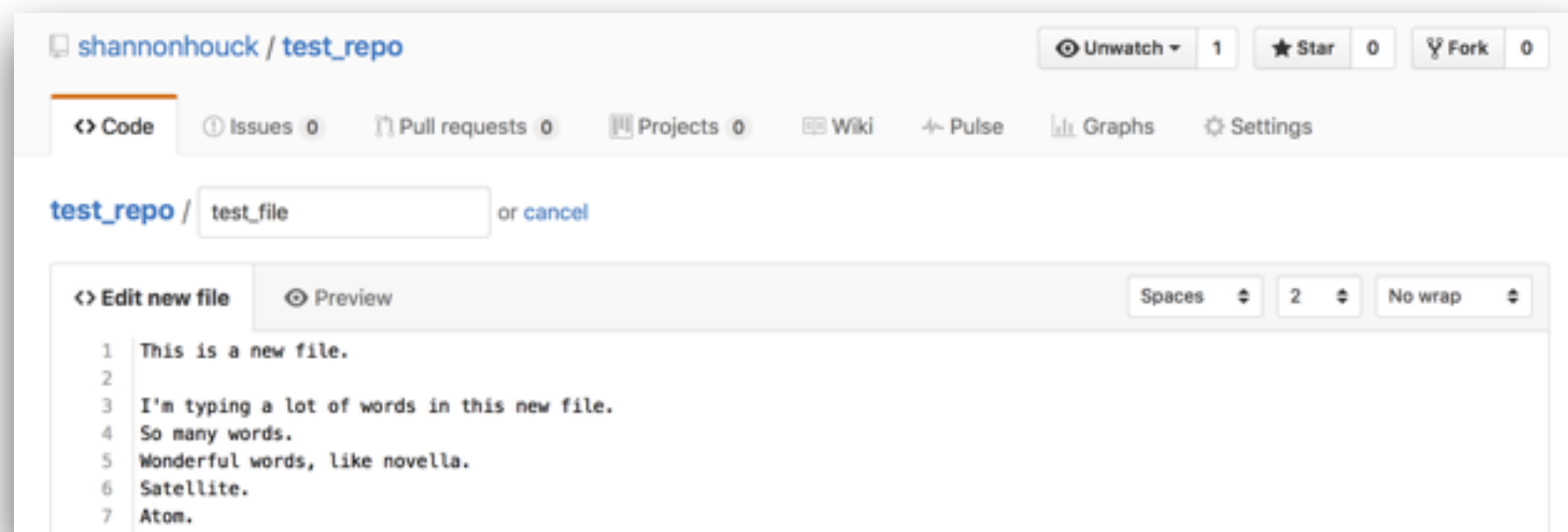
# Adding

- On GitHub, it's fairly easy. Make a file...

(1)

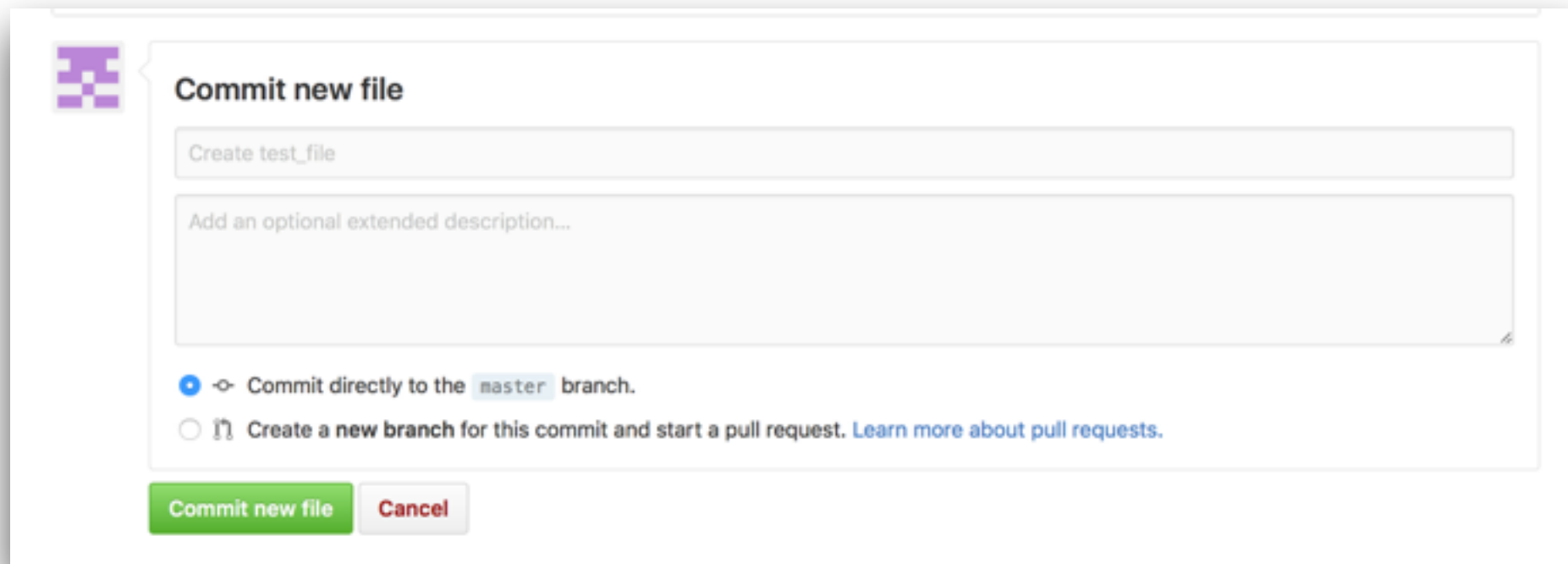


(2)



# Committing

- After editing, commit the changes!
- A **commit** finalizes the changes that you've made to your document; before the commit, the previous document (if there is one) is still the “official” version.



The screenshot shows the 'Commit new file' dialog in GitHub. It features a purple GitHub logo in the top left corner. The main title is 'Commit new file'. Below the title, there is a text input field containing 'Create test\_file'. Underneath that is a larger text area with the placeholder text 'Add an optional extended description...'. At the bottom, there are two radio button options: the first is selected and labeled 'Commit directly to the master branch.', and the second is labeled 'Create a new branch for this commit and start a pull request. Learn more about pull requests.' At the very bottom, there are two buttons: a green 'Commit new file' button and a grey 'Cancel' button.

# Add With Git

- In Git, you can create a new file in the repository the same way you would in any other directory:

```
$ touch [filename]
```

- Problem: New files aren't part of the repository until you tell Git to pay attention to them.
- Solution: 

```
$ git add [filename]
```
- This only needs to be done once!



# Commit With Git

- You can edit a tracked file with vim, as usual:

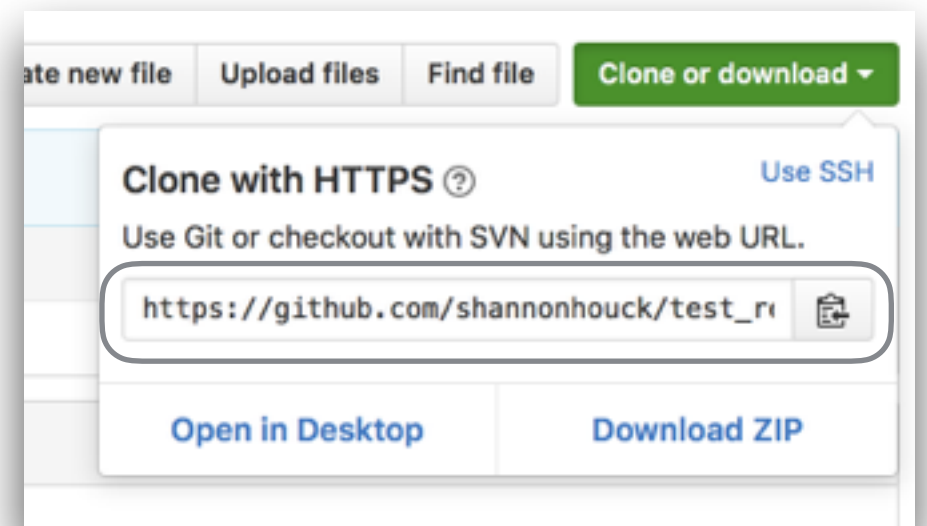
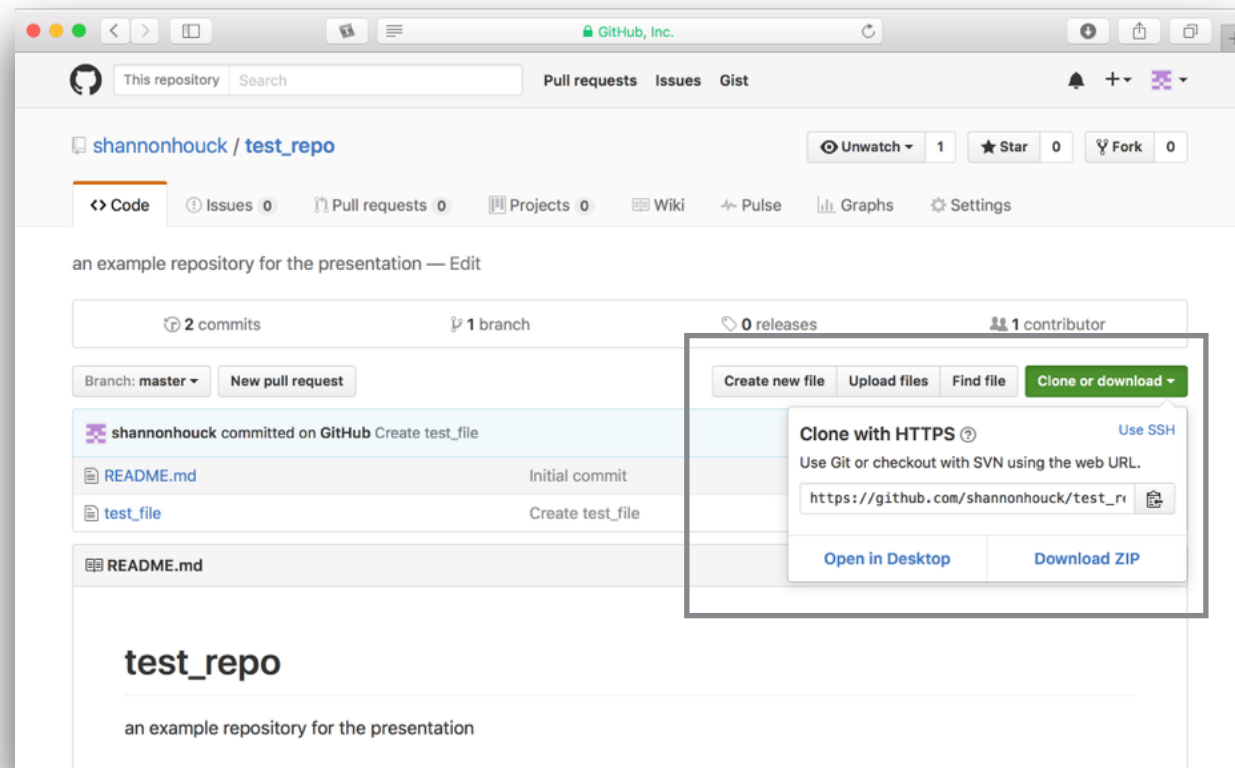
```
$ vim filename
```

- Problem: Changes aren't committed by default!
- Solution: \$ git commit *filename*
- You can add the -a flag to commit all tracked files at once, or use a wildcard to commit a subset.

# Cloning

- **Cloning** a repository allows you to download a copy to your local machine

```
$ git clone [GitHub URL]
```



# Cloning

- You can edit the clone on your own machine just like you would a normal repository- add files, commit changes, and so on.
- When you commit changes, though, they only change the repo on your local machine...
- Solution? Pushing!

# Pushing

- **Pushing** lets you “push” all of your committed local changes to the remote repository

```
$ git push [repo name] [branch]
```

- If the two fields are left blank, it defaults to “origin” and the branch you cloned from. (It won’t push if you’re not authorized to change a repository!)

# Pushing (Results)

```
test_repo — -bash — 76x12

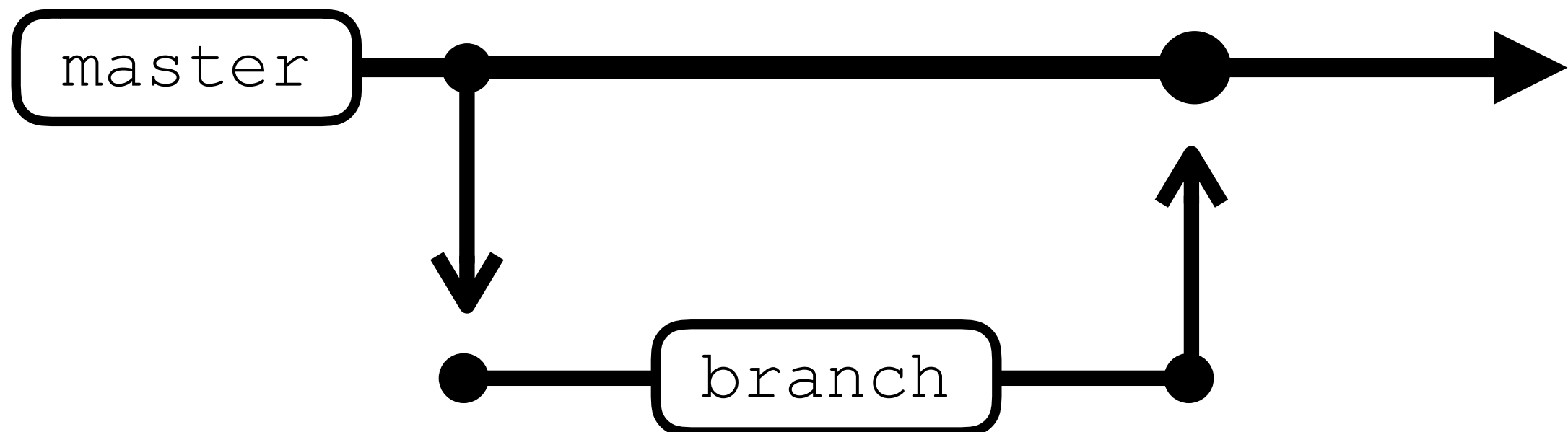
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
Shannons-MacBook-Pro:intro shannonhouck$ git push
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/shannonhouck/crawdad_tutorial.git
 dfdfbdc..d17865e  master -> master
Shannons-MacBook-Pro:intro shannonhouck$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

The screenshot shows the GitHub web interface for the repository 'shannonhouck / crawdad\_tutorial'. The 'Code' tab is selected, displaying a list of files in the 'intro' directory. The files include 'helloworld.cpp', 'helloworld.o', 'input.txt', 'main.cpp', 'main.o', 'molecule.cpp', 'molecule.h', 'molecule.o', and 'water'. Each file entry shows a commit message snippet and the time '2 days ago'. The top of the page shows repository statistics: 1 star, 0 forks, and 0 pull requests. The branch 'master' is currently selected.

File	Commit Message	Time
helloworld.cpp	Adding like all of the files ever	2 days ago
helloworld.o	Adding like all of the files ever	2 days ago
input.txt	Modified constructor to read a (properly formatted) file... No error ...	2 days ago
main.cpp	Modified constructor to read a (properly formatted) file... No error ...	2 days ago
main.o	Modified constructor to read a (properly formatted) file... No error ...	2 days ago
molecule.cpp	Modified constructor to read a (properly formatted) file... No error ...	2 days ago
molecule.h	Modified constructor to read a (properly formatted) file... No error ...	2 days ago
molecule.o	Modified constructor to read a (properly formatted) file... No error ...	2 days ago
water	Modified constructor to read a (properly formatted) file... No error ...	2 days ago

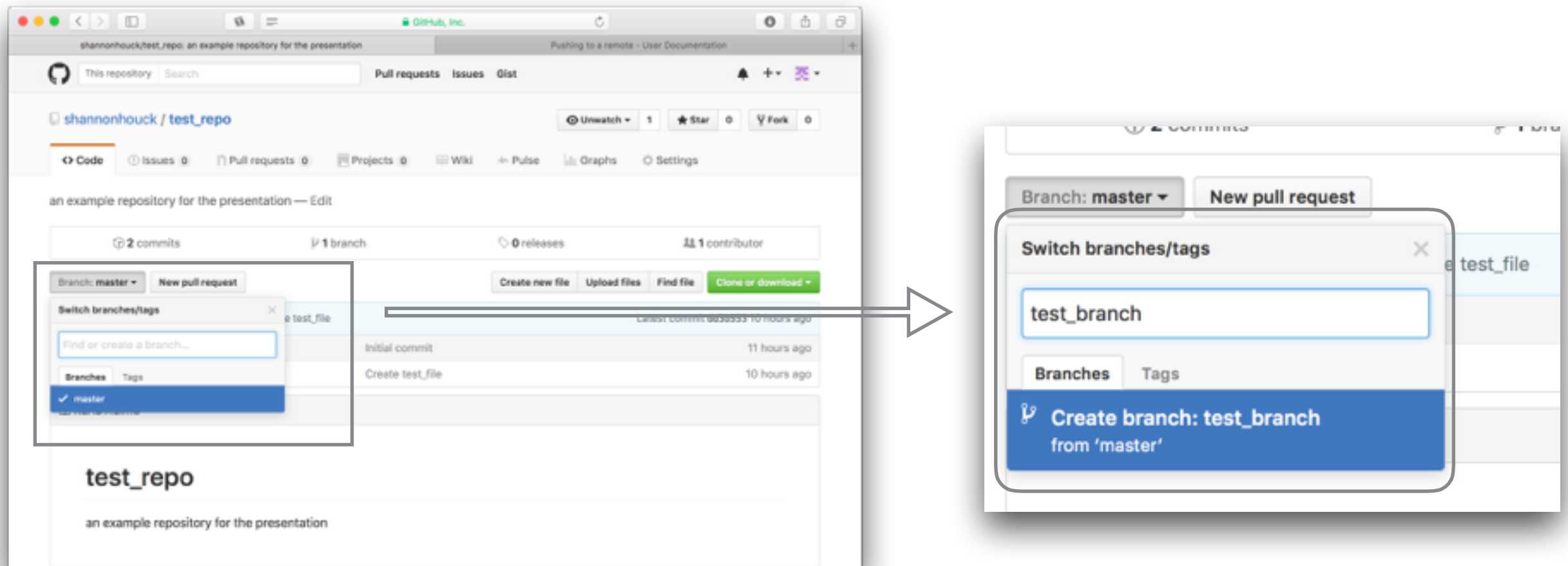
# Branching

- **Branching** lets you edit code (and make commits) without messing up the original code. You make a branch, edit it, and **merge** it back into the master branch at the end!



# Branching (GitHub)

- As usual, the GUI makes things easy to find...



# Branching (Git)

- Creating a new branch from the command line

```
$ git branch [branch name]
```

- To change from your current branch to a different branch, use the checkout command

```
$ git checkout [branch name]
```



# Merging

- After you're satisfied with your changes, you can **merge** a branch back into the master branch:

```
$ git merge [branch to merge]
```

- Now, all of your commits have been merged with the master branch!
- After merging, you can delete the branch:

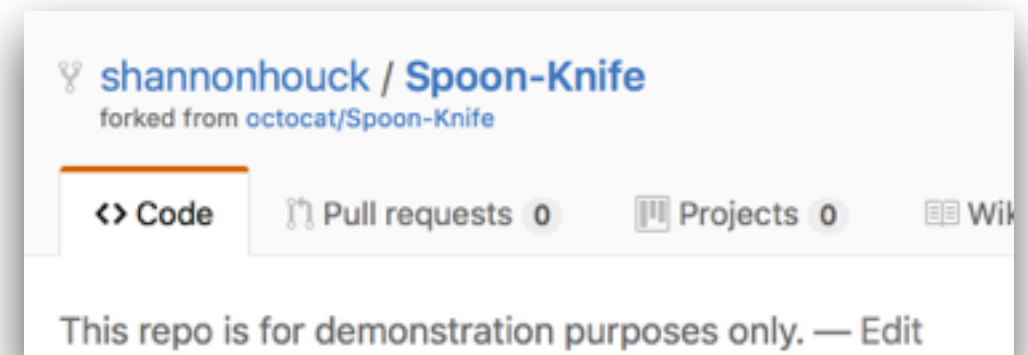
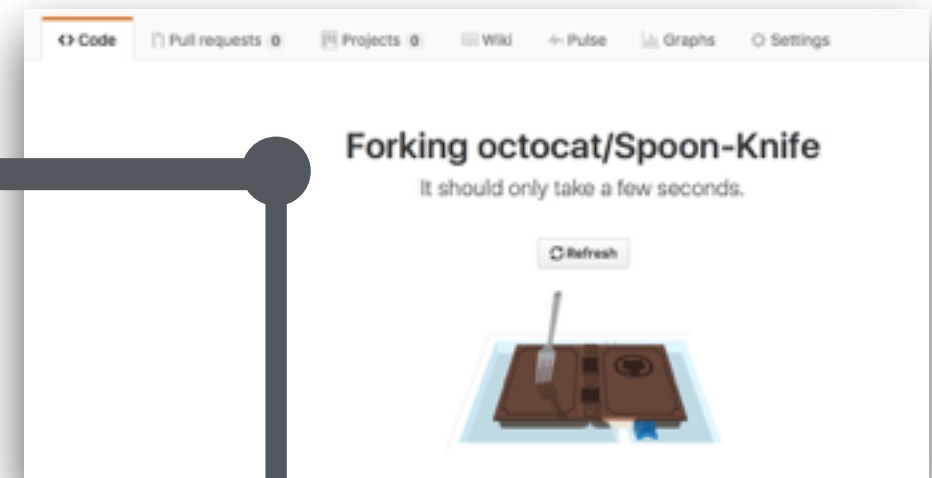
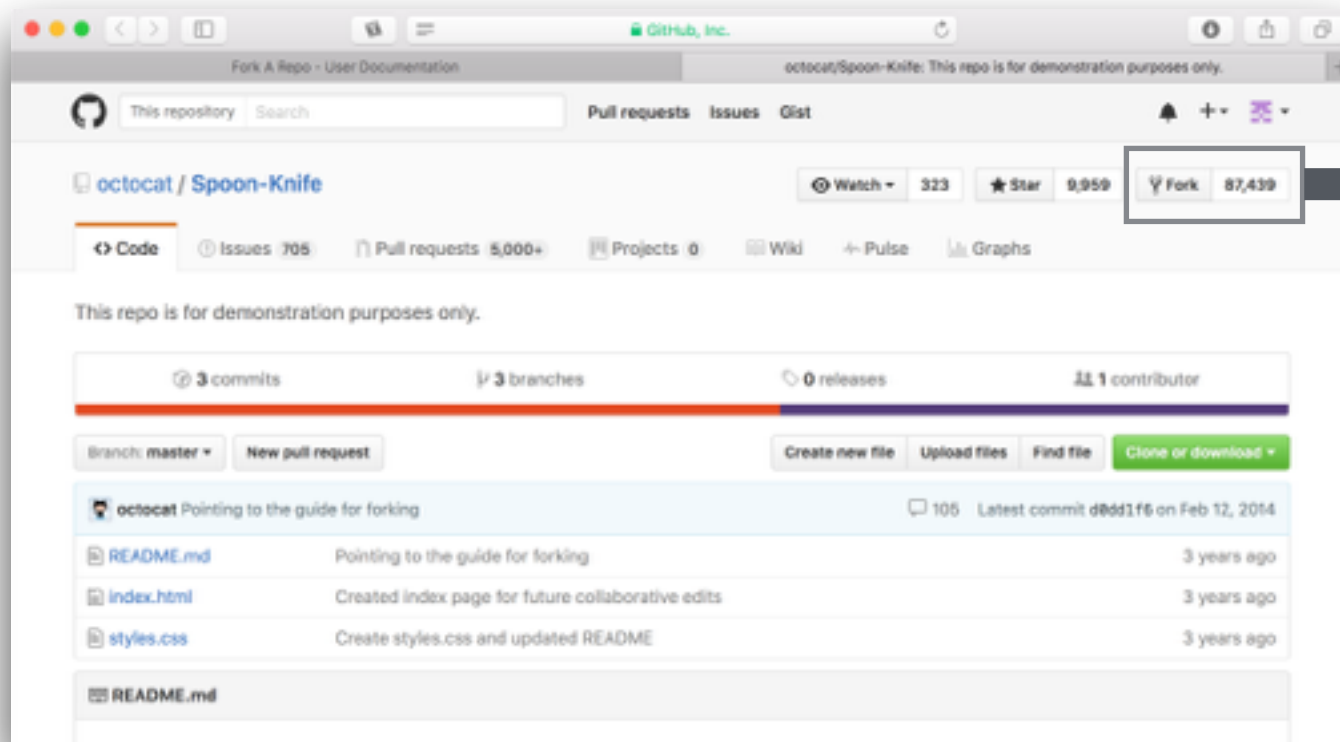
```
$ git -d branch [branch name]
```

# Forking

- **Forking** is similar to branching in that you create a copy based on another project, but in this case the original repo belongs to someone besides you
- Like branching, it lets you mess with code without worrying about ruining the original project

# Forking

- To create the original fork, use GitHub...



# Forking

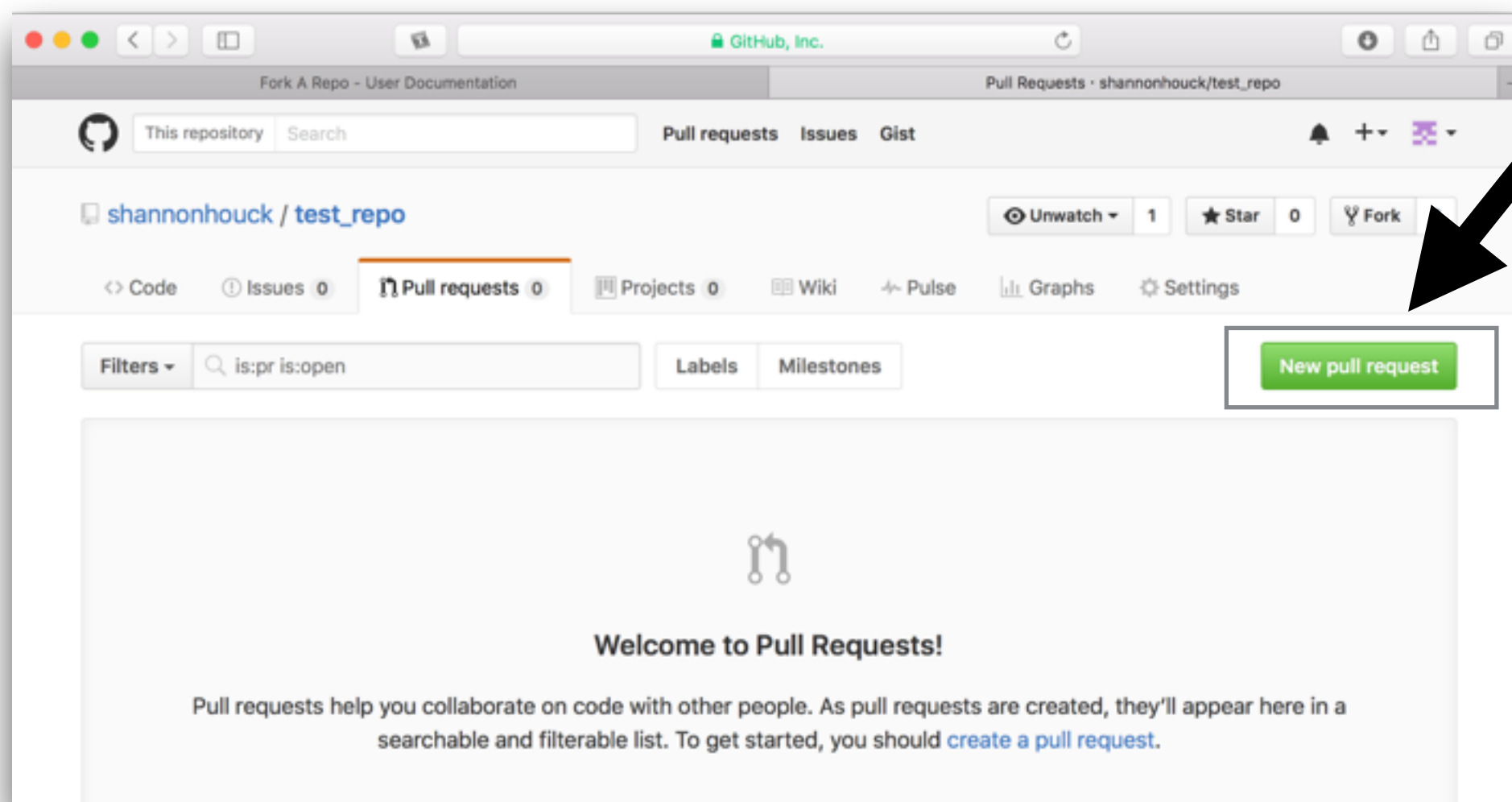
- Now, you have two options:
  1. Clone forked repo to your workstation
  2. Edit forked repo directly on GitHub
- You can use the same techniques introduced earlier to make commits, push changes, branch, and so on, just like with a normal repository
- How to merge the fork back into the original repository...?

# Pull Requests

- A **pull request** asks that the master version of the repo be updated with the commits you've made to a different version (branch, fork, etc.)
- You need a pull request to merge changes from a forked branch back into the original repo
- You also need it to merge branches on GitHub
- You do *not* need to make a pull request if you're just working on the command line!

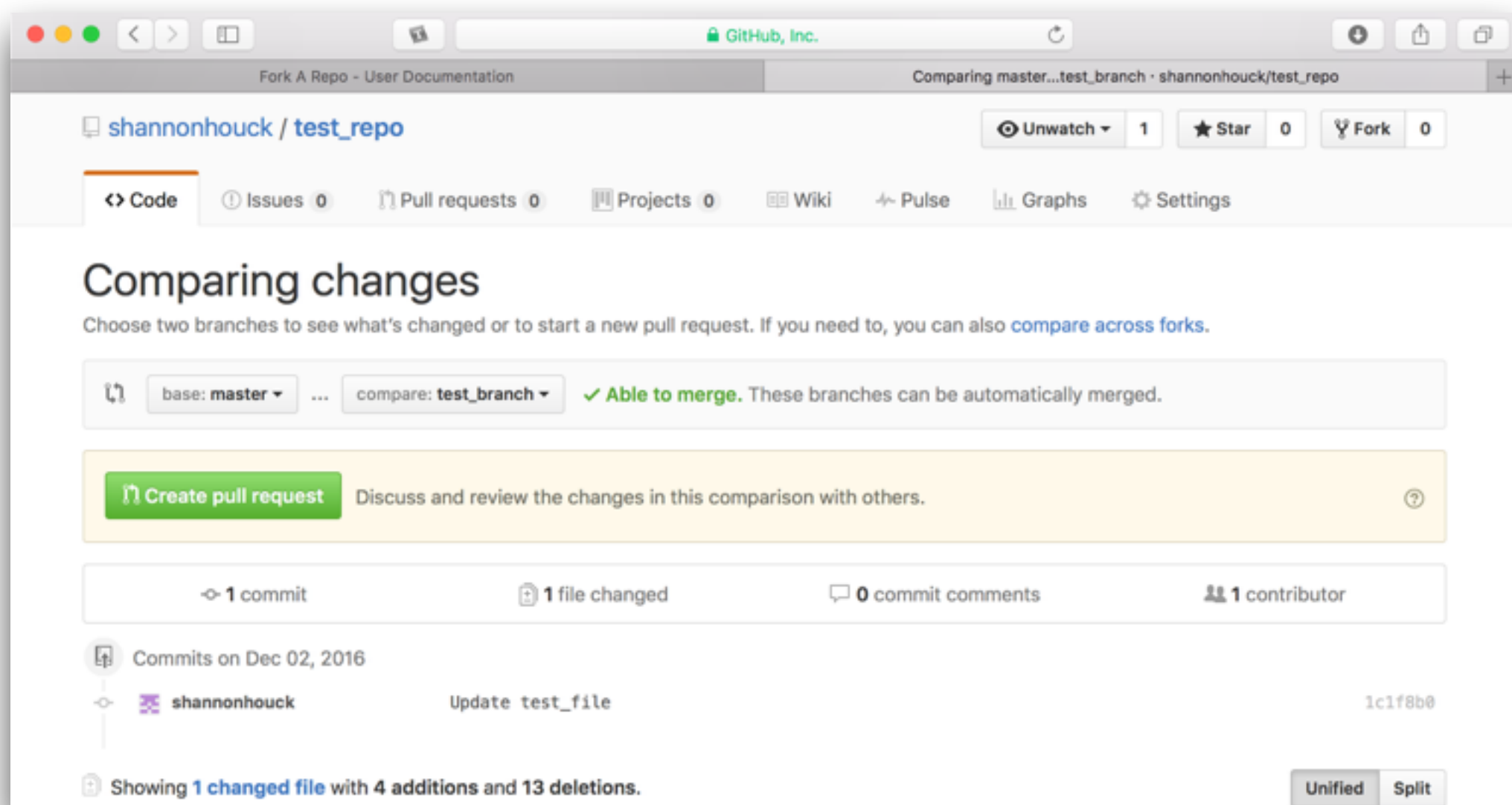
# Pull Requests

- From your repository, go to Pull Requests and click the “New pull request” button



# Pull Requests

- This opens a new pull request. Select branches.



# Pull Requests

- Examine the changes before submitting it



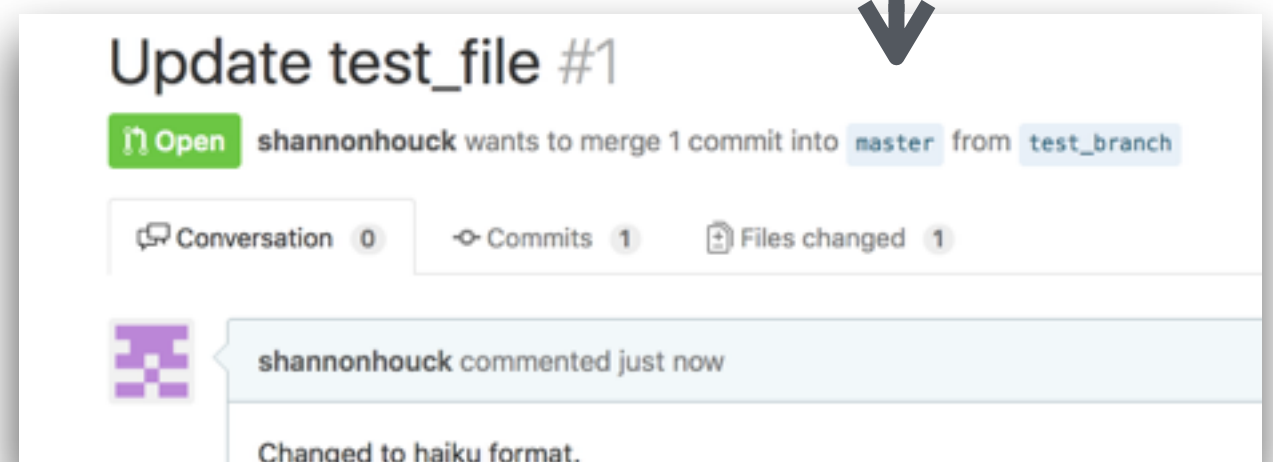
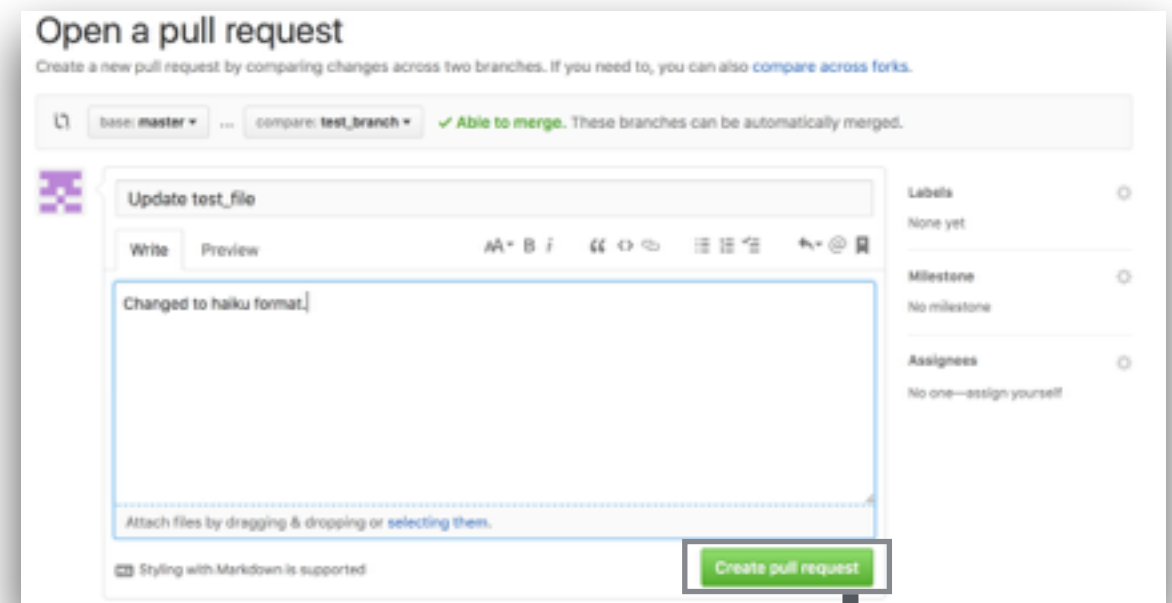
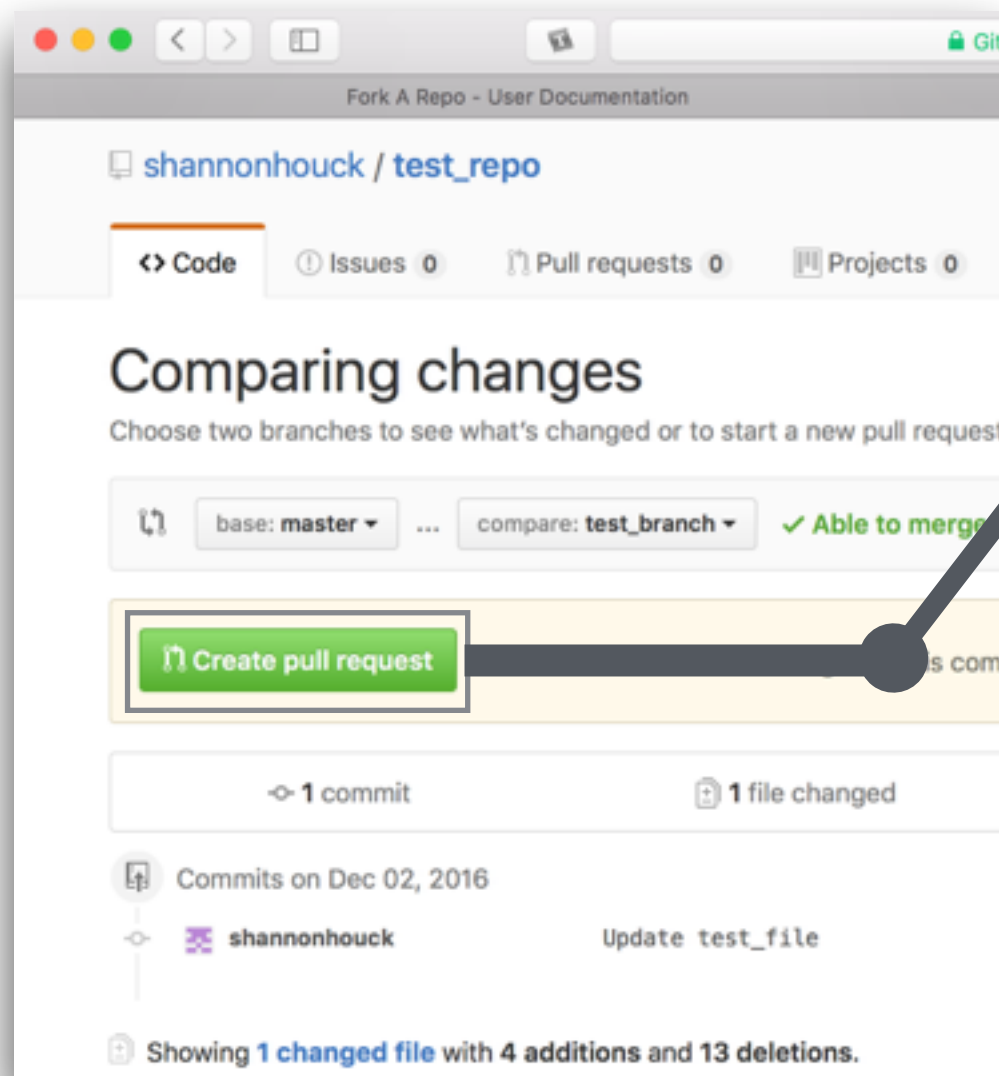
The screenshot shows a diff viewer interface. At the top, it says "Showing 1 changed file with 4 additions and 13 deletions." There are buttons for "Unified" and "Split" views. Below this, the file name "test\_file" is shown with a line number "17". The diff content is as follows:

...	...	@@ -1,14 +1,5 @@
1		-This is a new file.
	1	+This file is changed.
2	2	
3		-I'm typing a lot of words in this new file.
4		-So many words.
5		-Wonderful words, like novella.
6		-Satellite.
7		-Atom.
8		-North.
9		-Indigo.
10		-Pumpkin seeds.
11		-Discrete mathematics.
12		-Pencils.
13		-Teacups.
14		-...
	3	+I am removing the words.
	4	+
	5	+Minimalism.



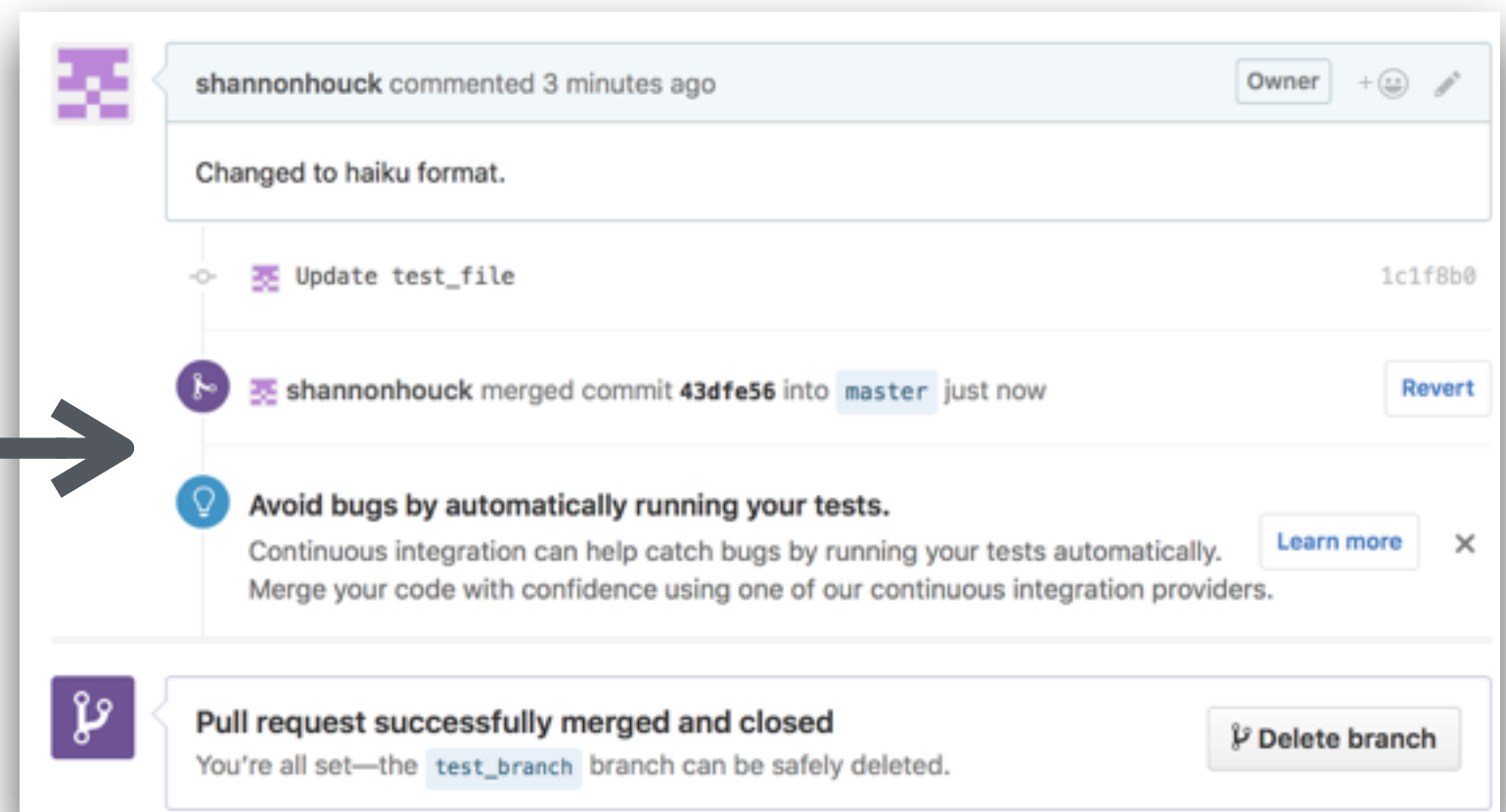
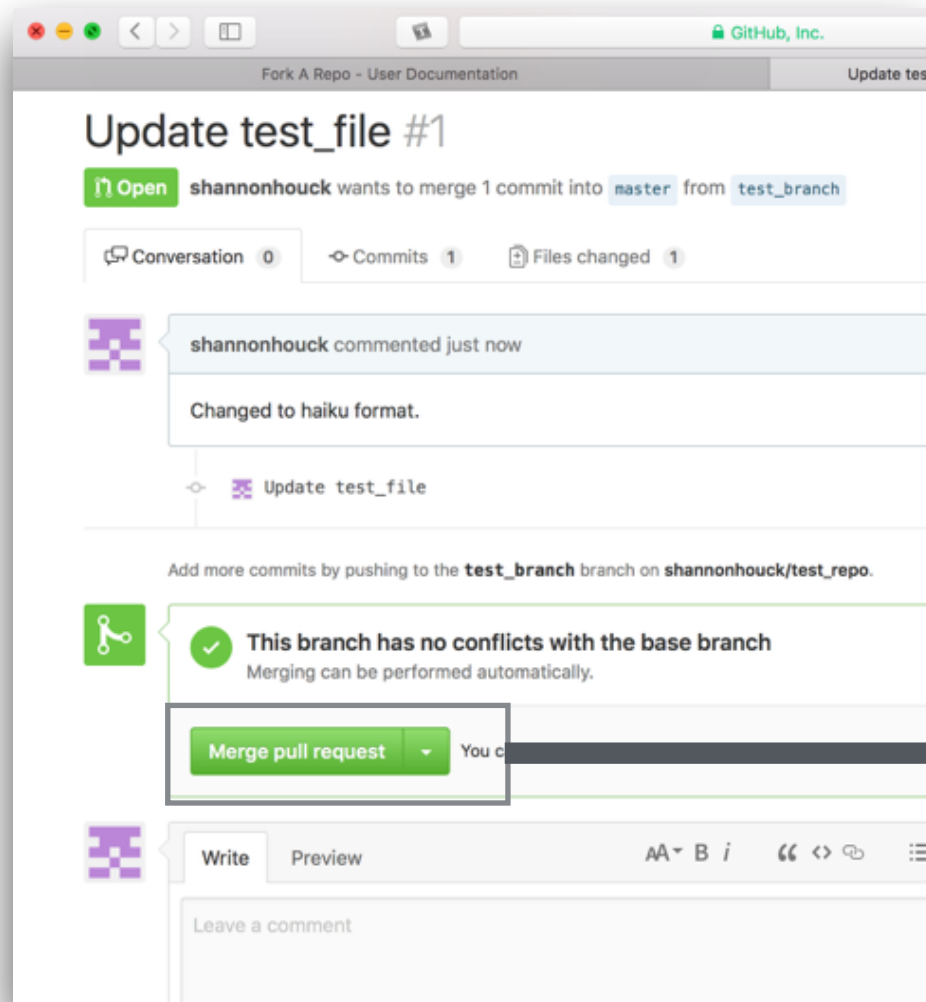
# Pull Requests

- Now, submit the request for review!



# Pull Requests

- Now that you've made a pull request, merge it!



# Pull Requests

- Now, looking at the original repository, the file has been updated to match test\_branch.

The screenshot shows the GitHub interface for the repository 'shannonhouck / test\_repo'. At the top, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below this is a navigation bar with tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The 'Code' tab is selected. Below the navigation bar, there is a section for the file 'test\_repo / test\_file'. It shows a pull request by 'shannonhouck' titled 'Update test\_file' with commit hash '1c1f8b0' and a timestamp of '24 minutes ago'. Below the pull request information, it says '1 contributor'. The file content is displayed below, showing 6 lines (3 sloc) and 61 Bytes. The content is: 'This file is changed.', 'I am removing the words.', and 'Minimalism.'.

shannonhouck / test\_repo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Branch: master test\_repo / test\_file Find file Copy path

shannonhouck Update test\_file 1c1f8b0 24 minutes ago

1 contributor

6 lines (3 sloc) | 61 Bytes Raw Blame History

```
1 This file is changed.
2
3 I am removing the words.
4
5 Minimalism.
```

# Other Useful Commands

- `$ git status`  
Prints the current branch and how many commits you have on the current branch that haven't been pushed yet
- `$ git reset HEAD --hard`  
Restores last commit (i.e. changes all tracked files back to where they were at the last commit)

Any Questions?

```
/* thank you */
```