Version Control with Git

Organizing and Structuring Code Development

Set-up

- Installing Git: https://git-scm.com
- Set up an account: GitHub.com (web interface and command-line), git.uwawterloo.ca (command-line)
- Install the GitHub Desktop App (desktop.github.com)
  - Works on both Windows and OS X
- A useful git demo: try.github.io

Why use Git?

- Multiple people modifying a set of files can get messy
    - Someone accidentally uses an old version, two people change the same file and have to figure out how to combine them, someone breaks something and you have to figure out where it happened…
- Git provides a convenient way to keep track of your progress by saving ‘checkpoints’ of your work along the way, letting you look over past changes
- Provides a tool for each member to work on the project without having to worry (too much) about overlapping.
- Makes it easy to share code updates with the other members of your group.

Basic Git Structure

- A Git project is called a repository
- There are two main components to a Git repository
  - Commits: A commit is a ‘unit of work’, or ‘checkpoint’ for your code. They track your progress.
  - Branches: Branches are different concurrent ‘versions’ or ‘workspaces’ of your project.

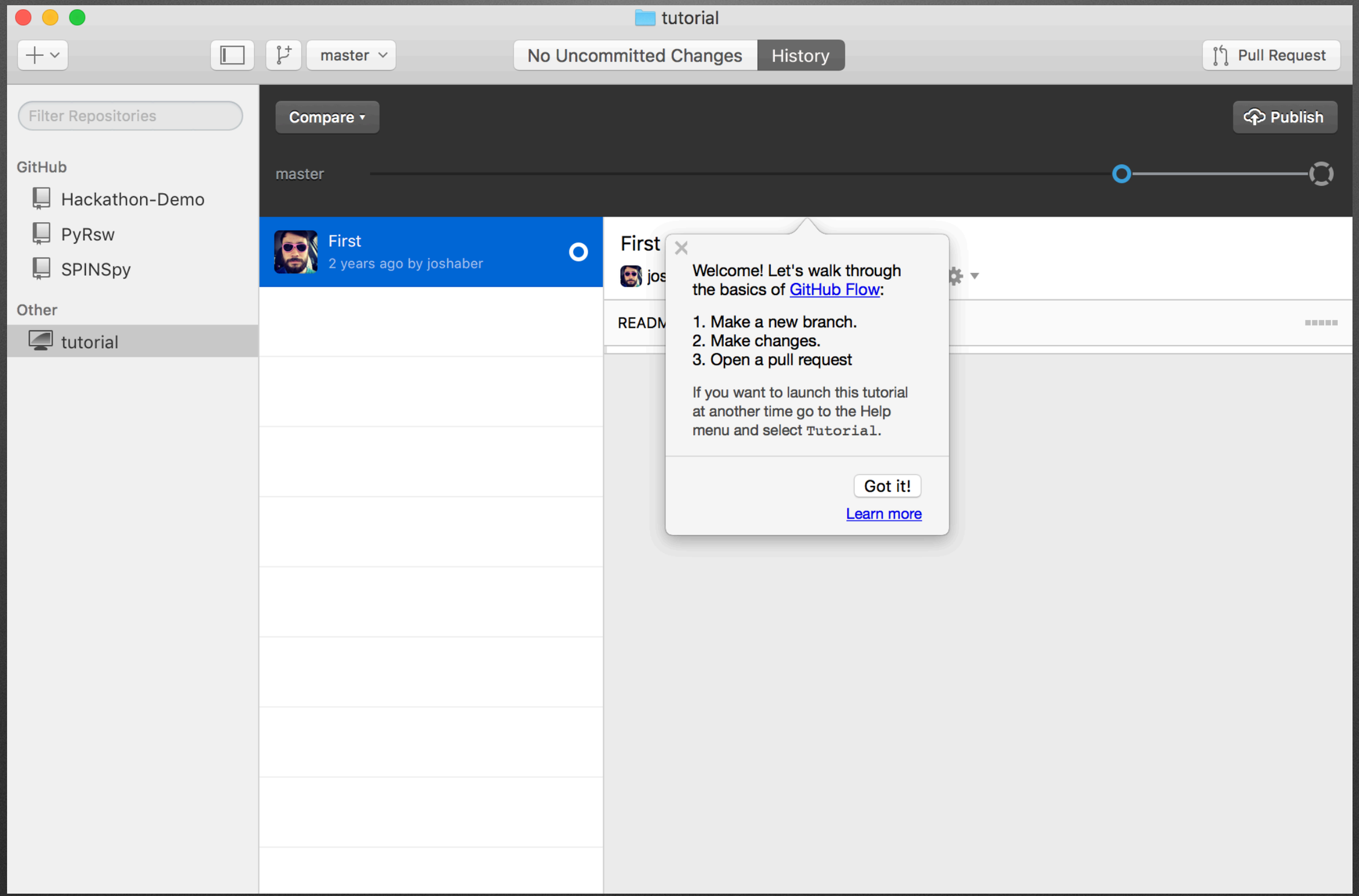
More on Branches

- Every repository has at least one branch, called *master*.
    - The master branch represents the ‘best working version’ of your project.
- When developing new features, you will typically work on another branch. Changes made on one branch **do not** impact other branches.
    - This gives you the opportunity to test and develop new code before releasing it on the *master* branch.
    - The new code can the be merged using a *pull request.*

Let’s work through some of the main features.

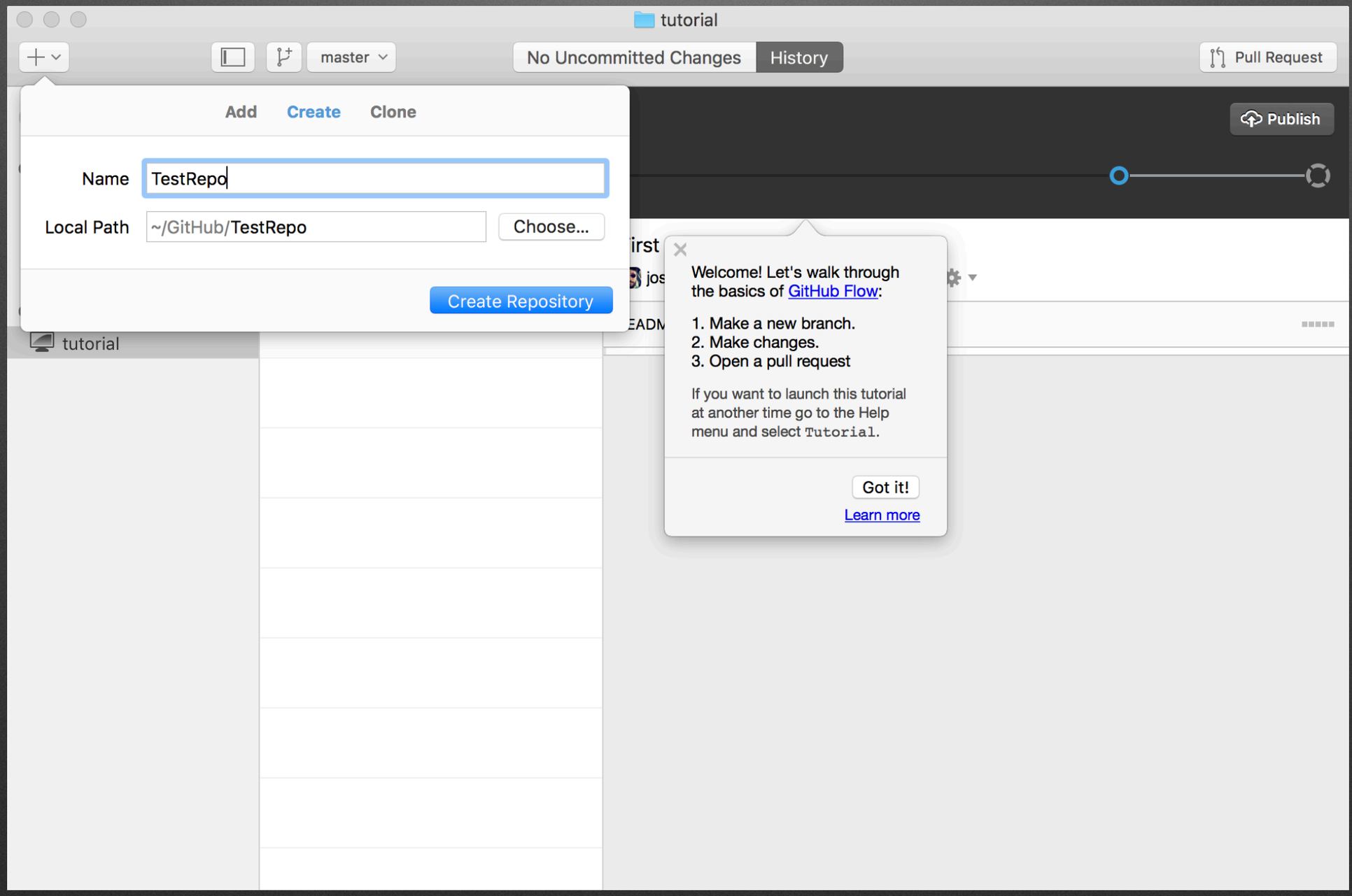
- Will be using the GitHub Desktop App
    - You can certainly use git through the command-line, but this workshop focuses on the GUI method.
- We’ll look at:
    - creating/accessing a repo
    - creating commits and branches
    - merging branches back into the master branch
    - resolving code conflicts
    - reviewing code

Using the GitHub App



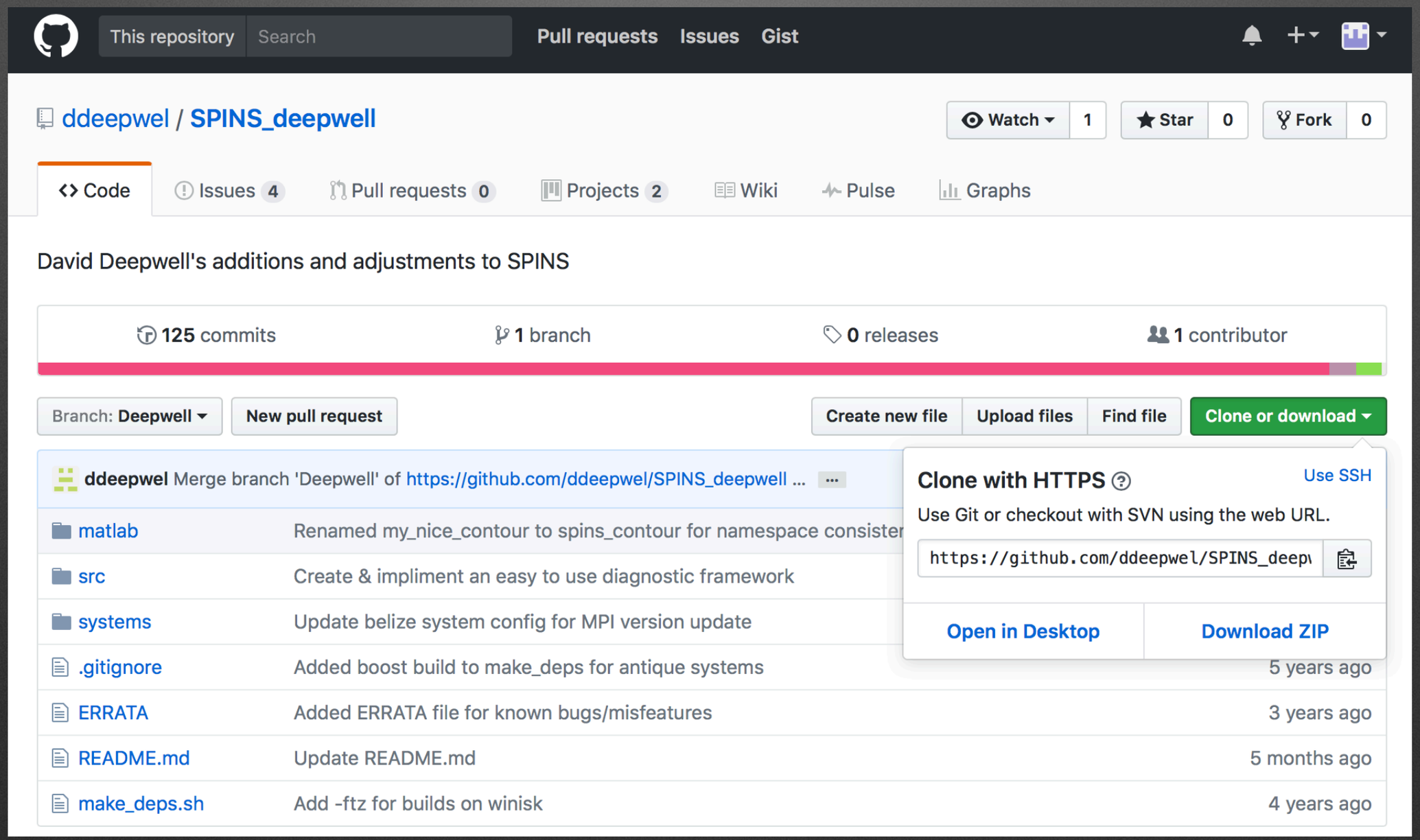
Creating a Repo

- Only **one** person in the group needs to create the repo
- *Publish* the repo to make it available on GitHub



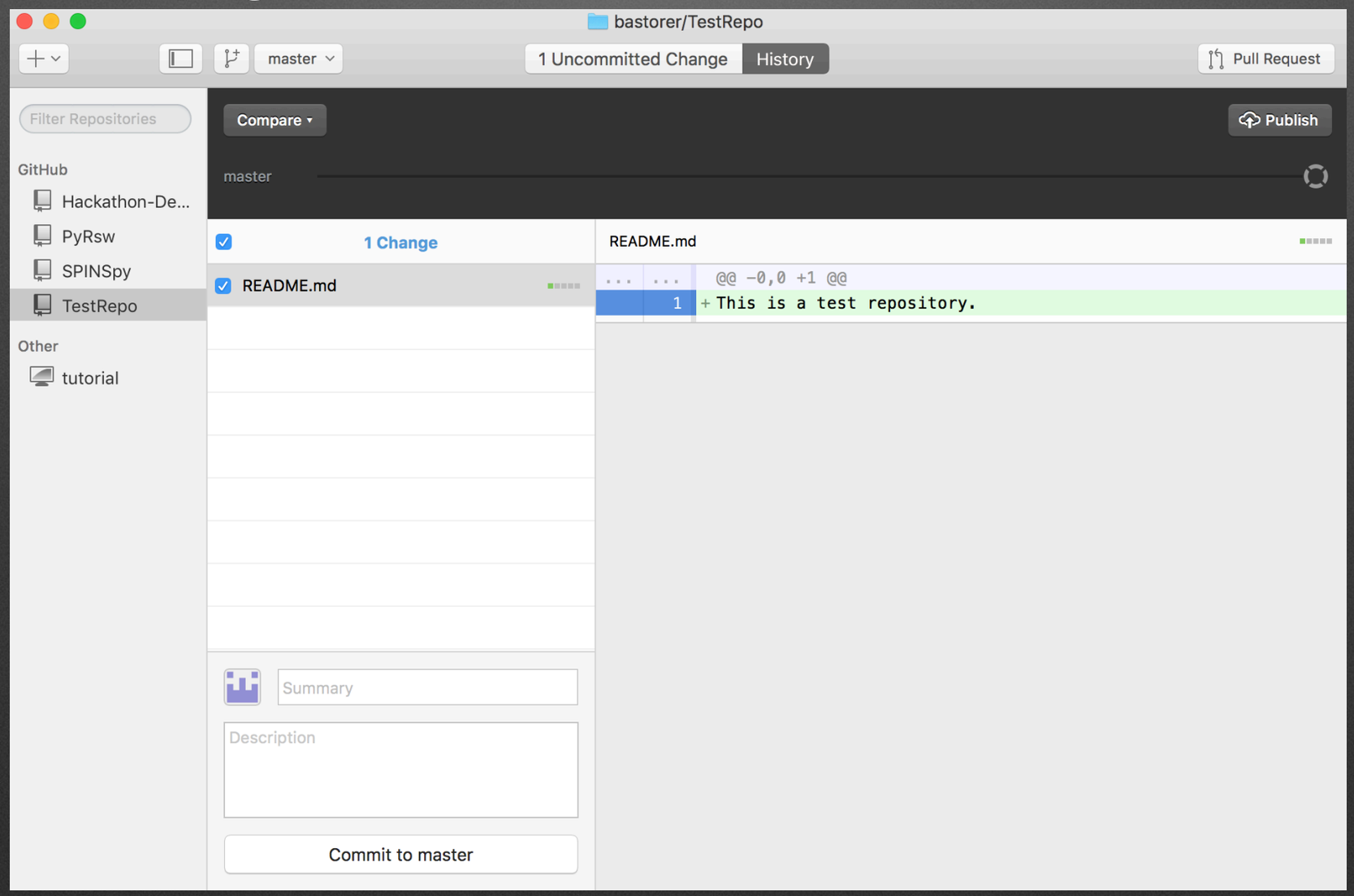
Cloning a Repo

- Search for the desired repo on GitHub, select clone
- Access this presentation at: https://github.com/bastorer/TestRepo



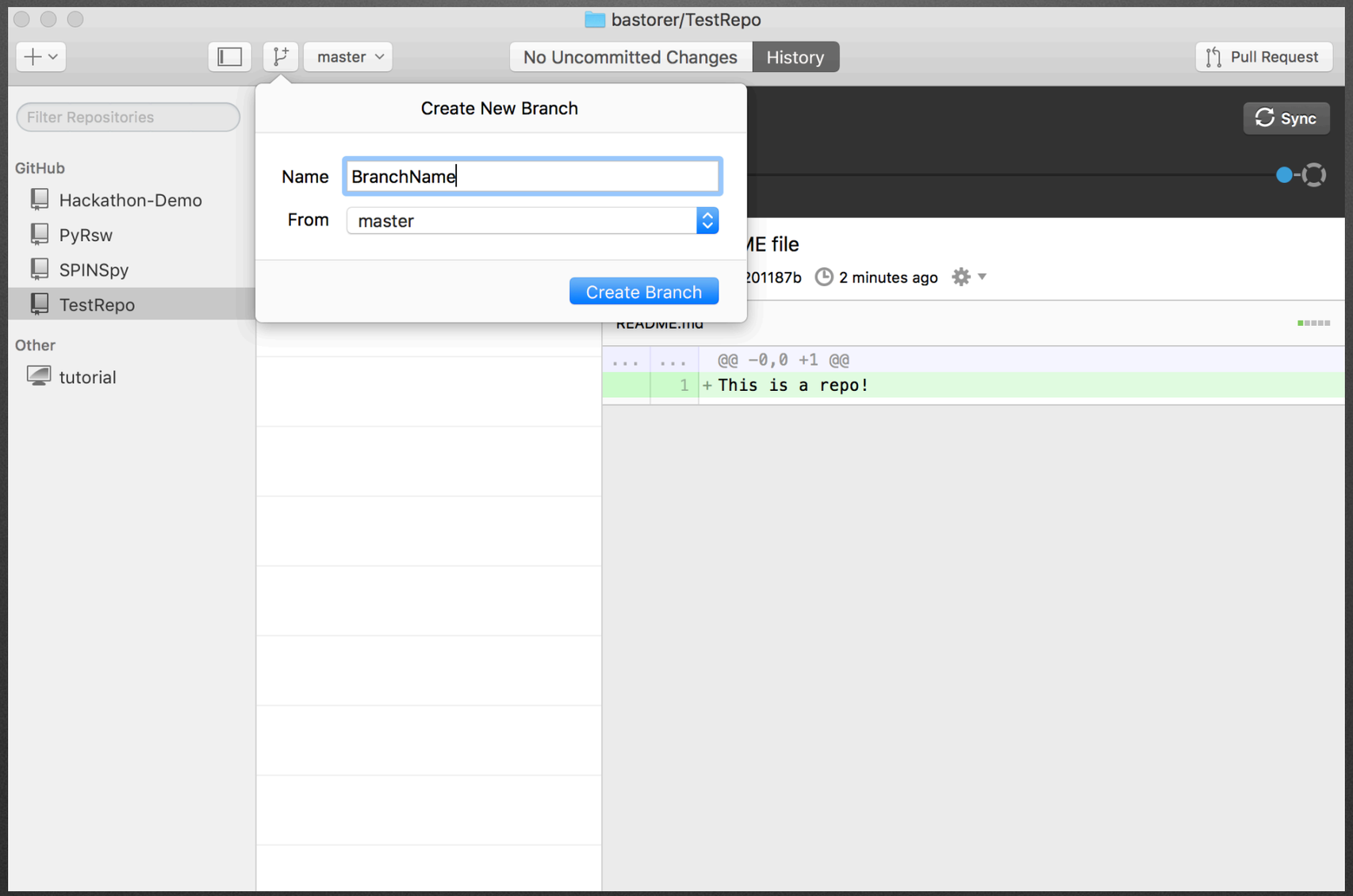
Making a Commit

- Right now we have an empty repository. Let's add our first file, *README.md*.
  - This file provides a description for the project and shows up on the GitHub page.



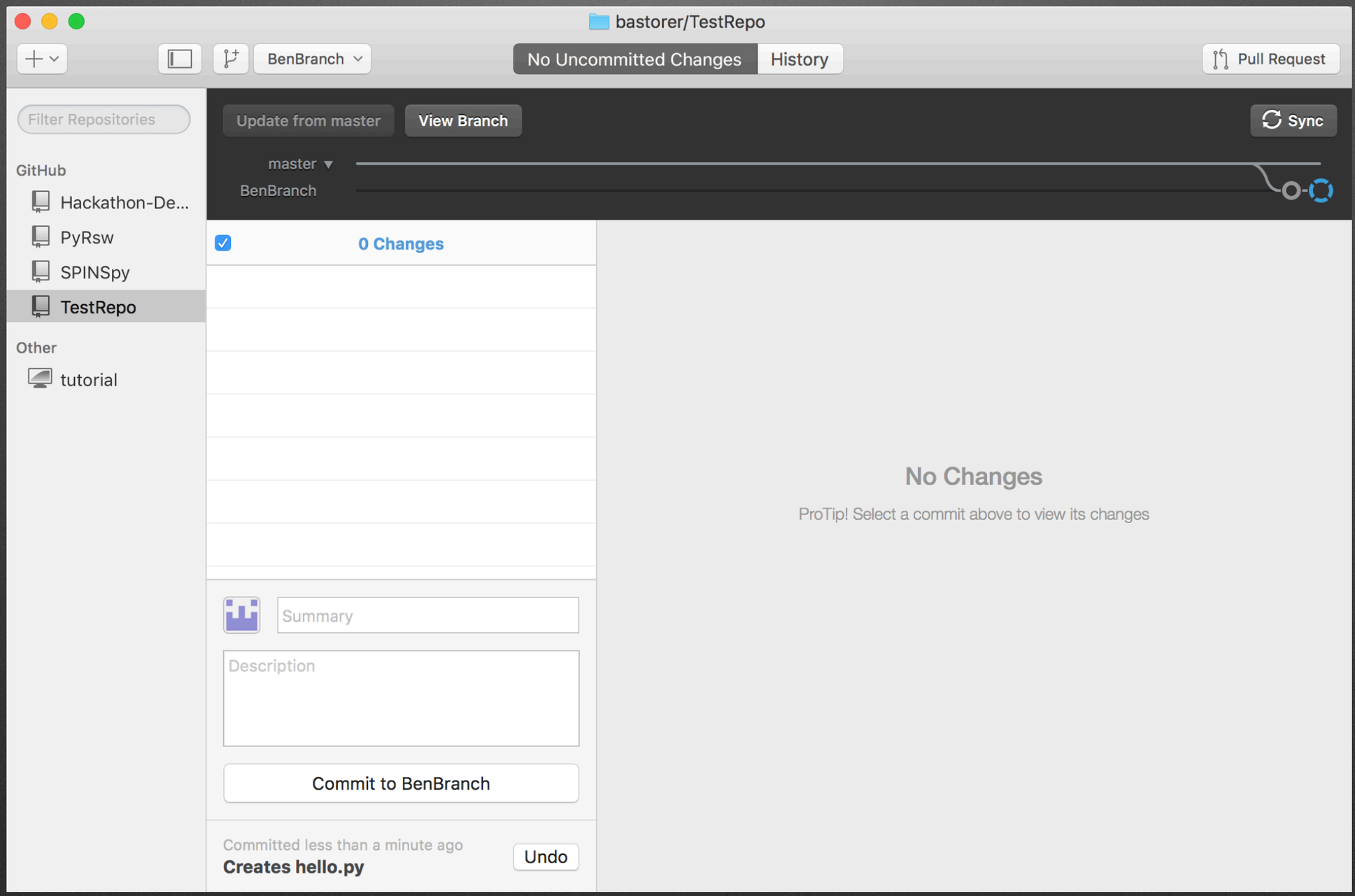
Making a Branch

- Generally, it's best to not change the master branch directly, but instead to work on a development branch.
  - It may be a good idea for each member of your team to have their own branch.

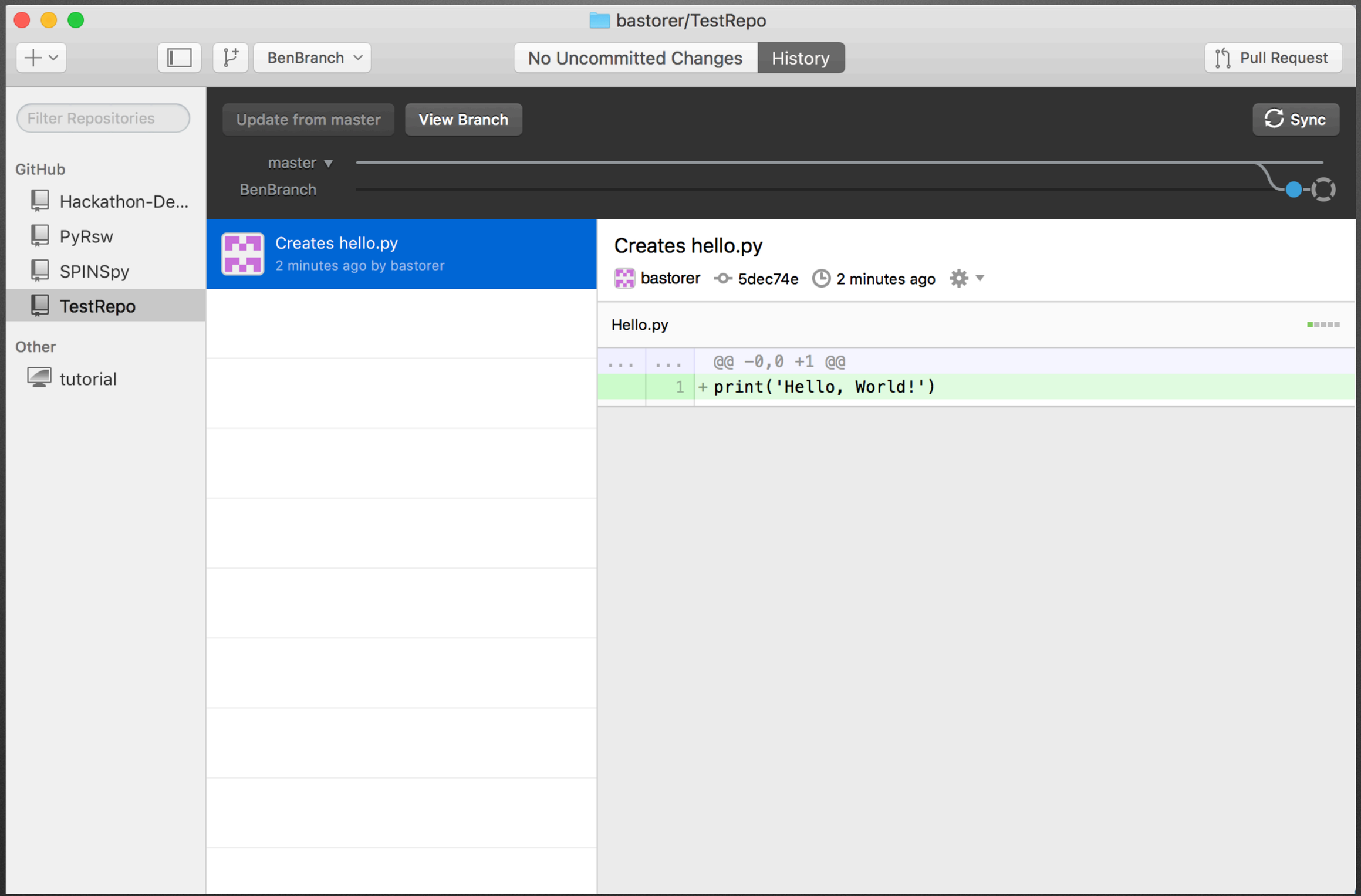


Commit to the Branch

- Let's create a file *Hello.py* on our new branch, and commit the new file.



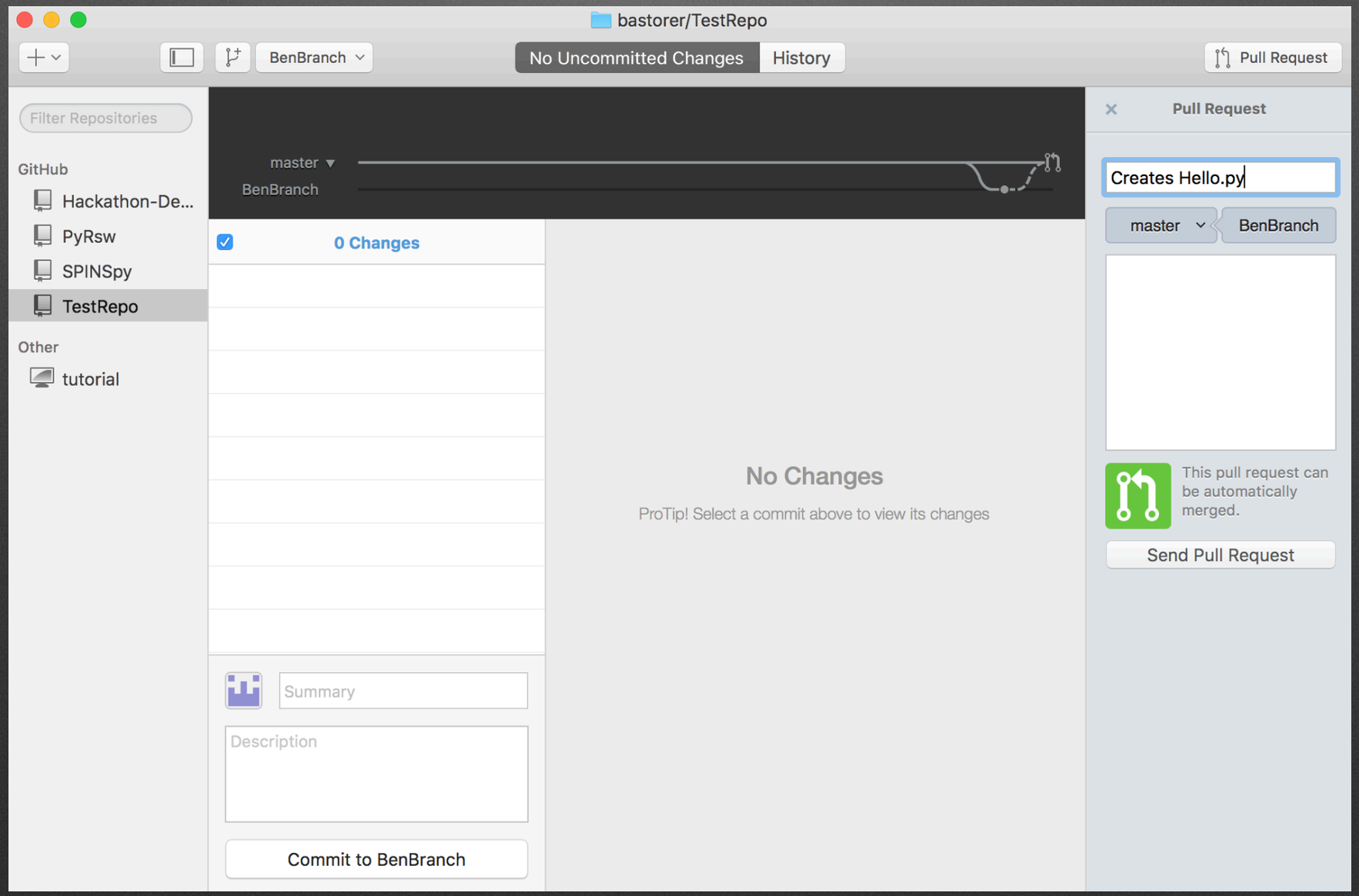
- Like the repo, need to *publish* the branch to push it to GitHub.
- Branch structure shown in Desktop App, *History* tab shows commits



Merging the Branch

- Suppose that we are happy with the work on our branch and want to include it in the master branch.

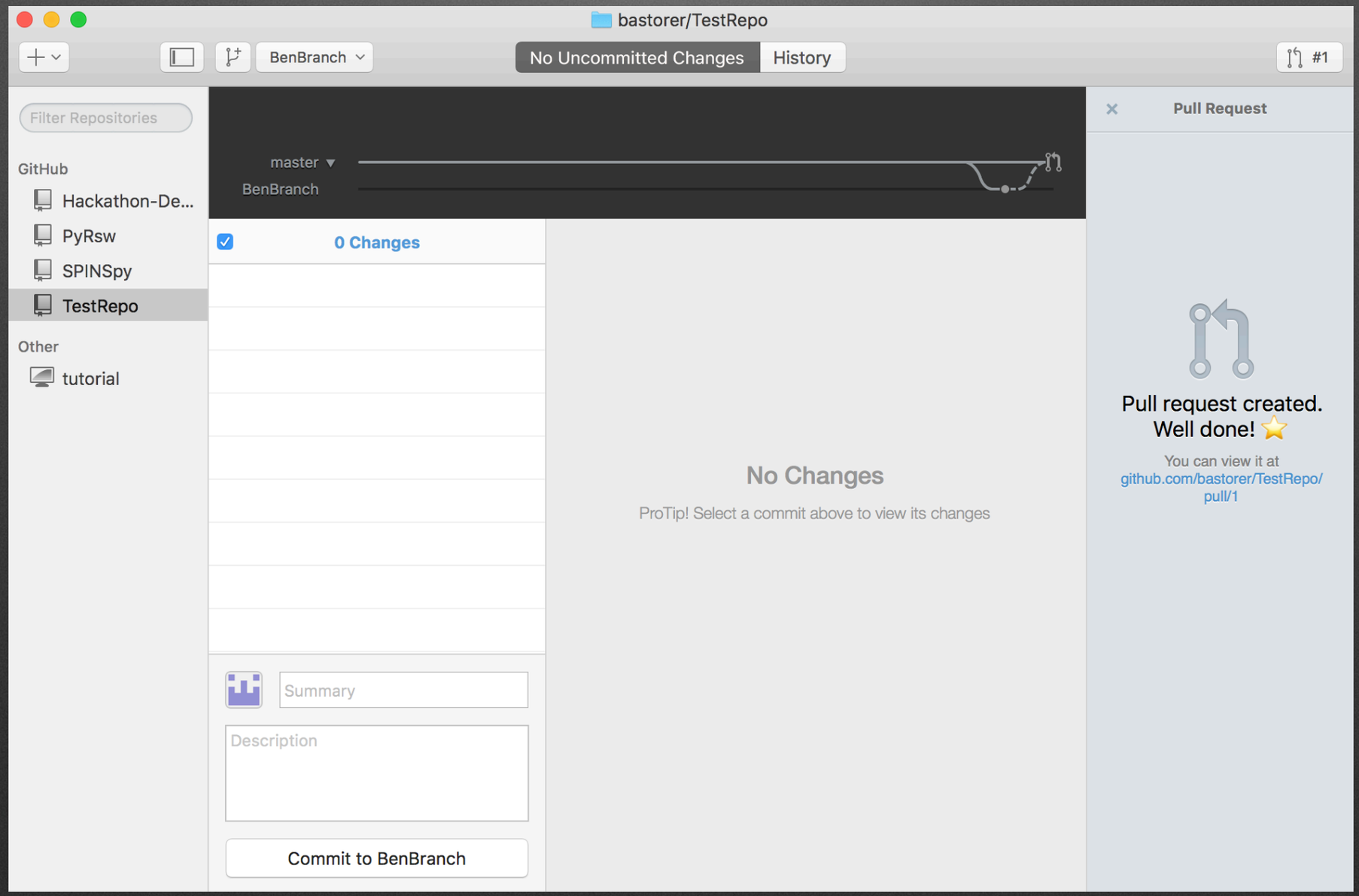
- We can do this by submitting a ‘pull request’, requesting to merge our branch into the master branch.



Merging the Branch

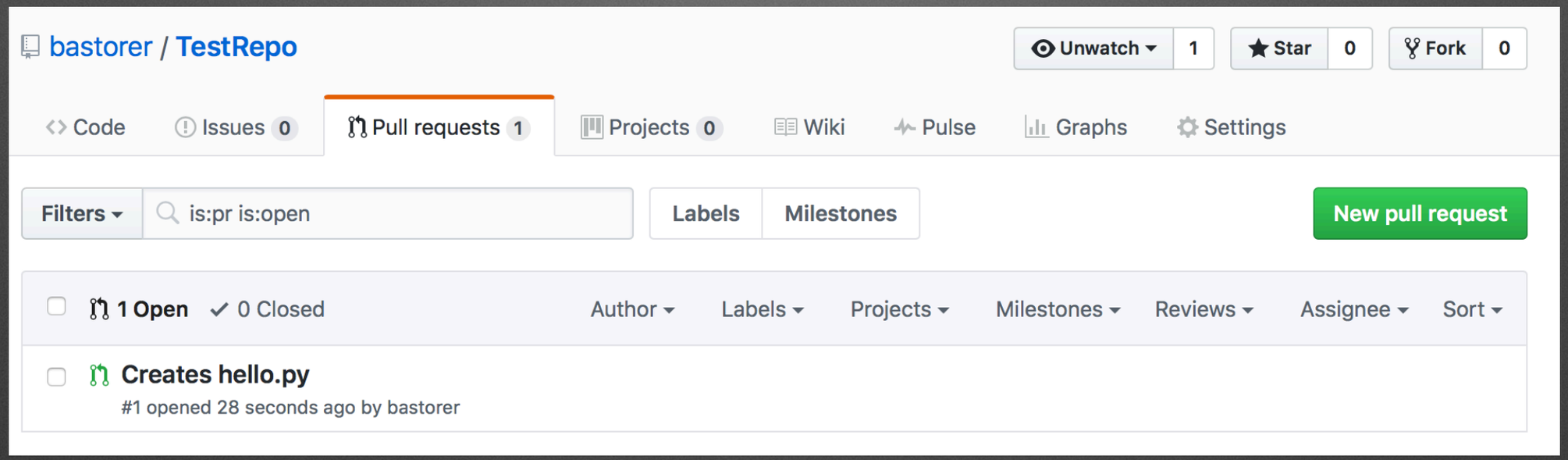
- Suppose that we are happy with the work on our branch and want to include it in the master branch.

- We can do this by submitting a ‘pull request’, requesting to merge our branch into the master branch.



Accepting the Pull Request

- To accept the pull request, or to see other pull requests, go to the GitHub page for the repo.



Accepting the Pull Request

Creates hello.py #1

Edit

Open **bastorer** wants to merge 1 commit into `master` from `BenBranch`

Conversation 0 | Commits 1 | Files changed 1 | +1 −0

bastorer commented 40 seconds ago — Owner

No description provided.

Creates hello.py 5dec74e

Add more commits by pushing to the **BenBranch** branch on **bastorer/TestRepo**.

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request | You can also open this in GitHub Desktop or view command line instructions.

Reviewers

No reviews—request one

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Handling Conflicts

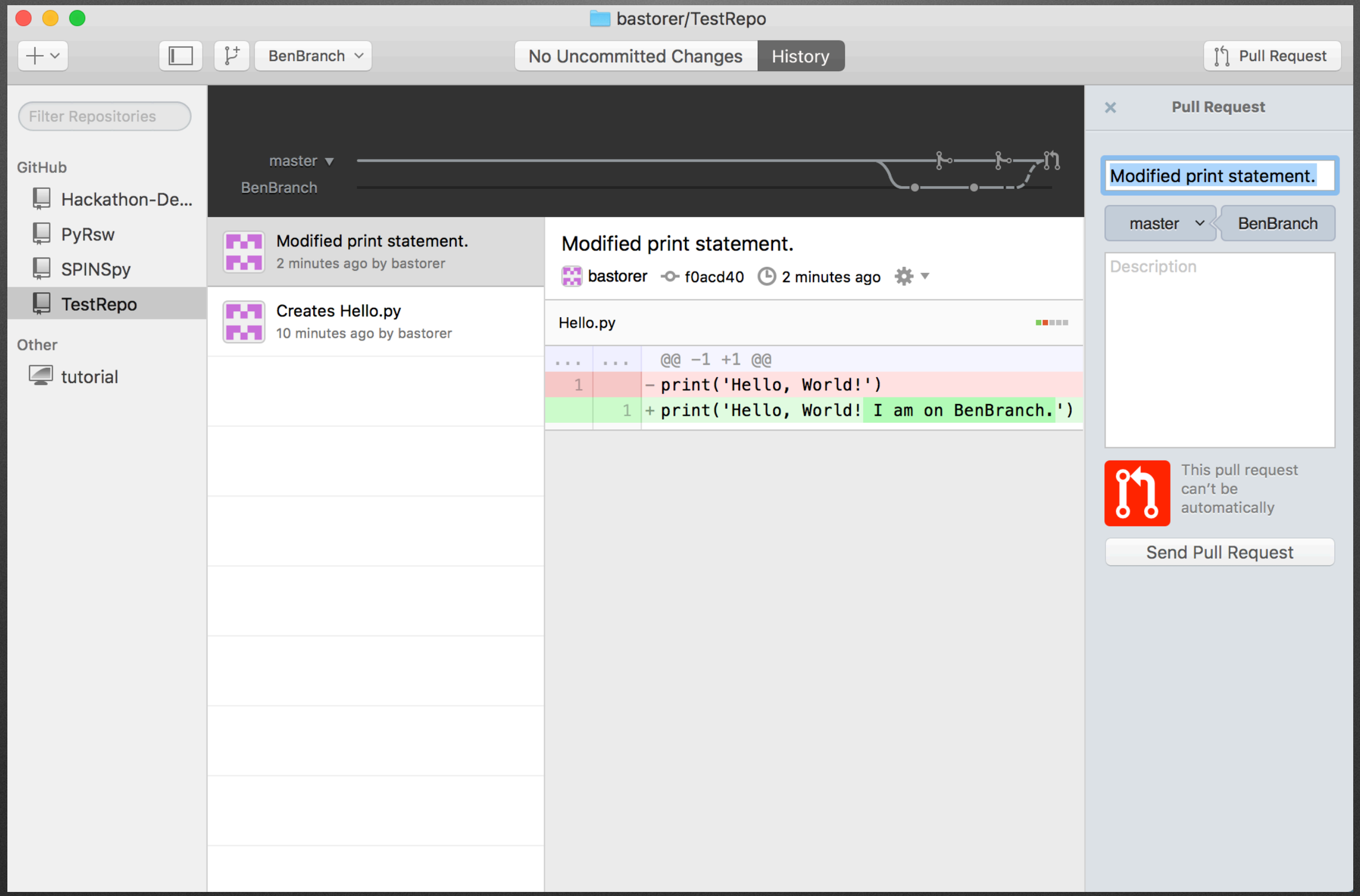
- What happens if two people, working on different branches, modify the same file, and then try to merge those changes with a pull request?

- Since each person will most likely be working on their own branch, this is certainly possible.

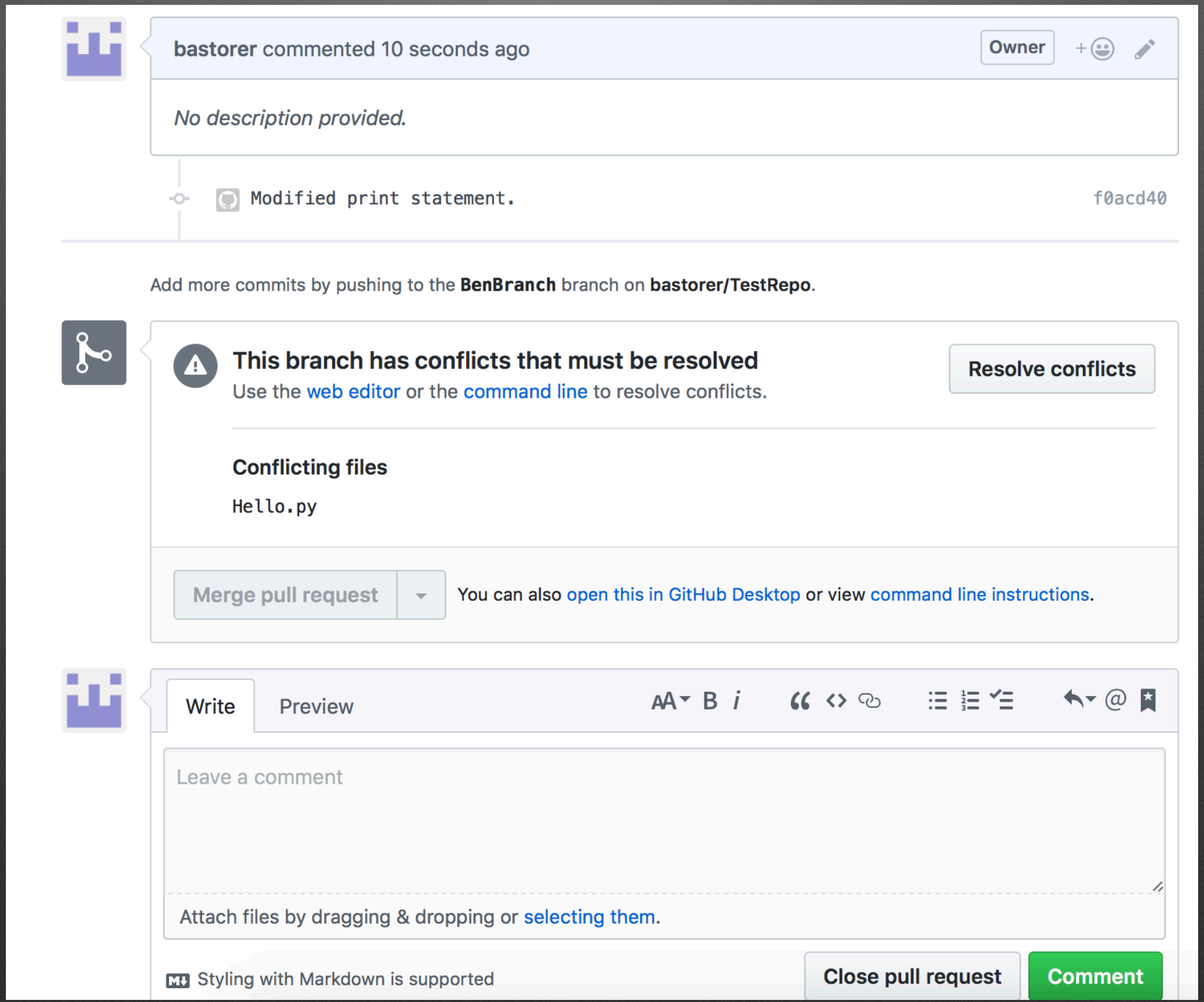
- Fortunately, GitHub makes it very easy to resolve these kinds of conflicts.

- Let’s create a new branch and force a conflict to see how we can resolve it.

Handling Conflicts



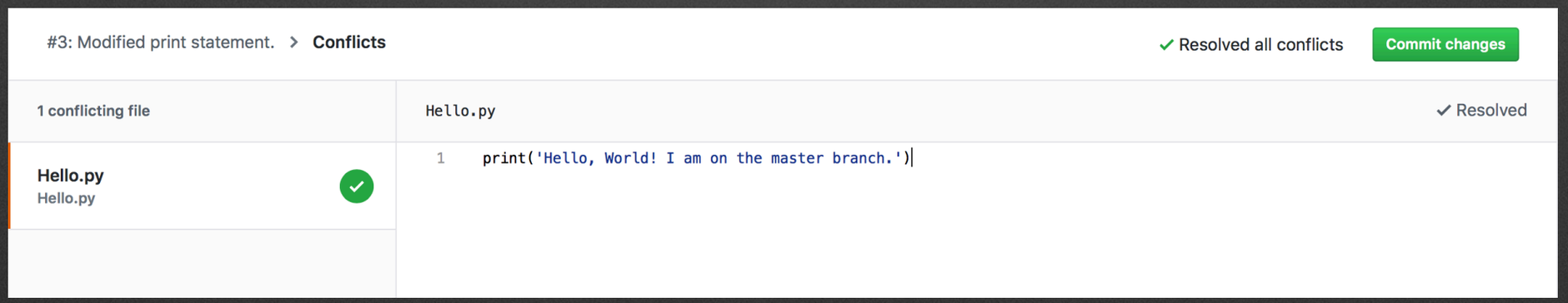
Handling Conflicts



Handling Conflicts

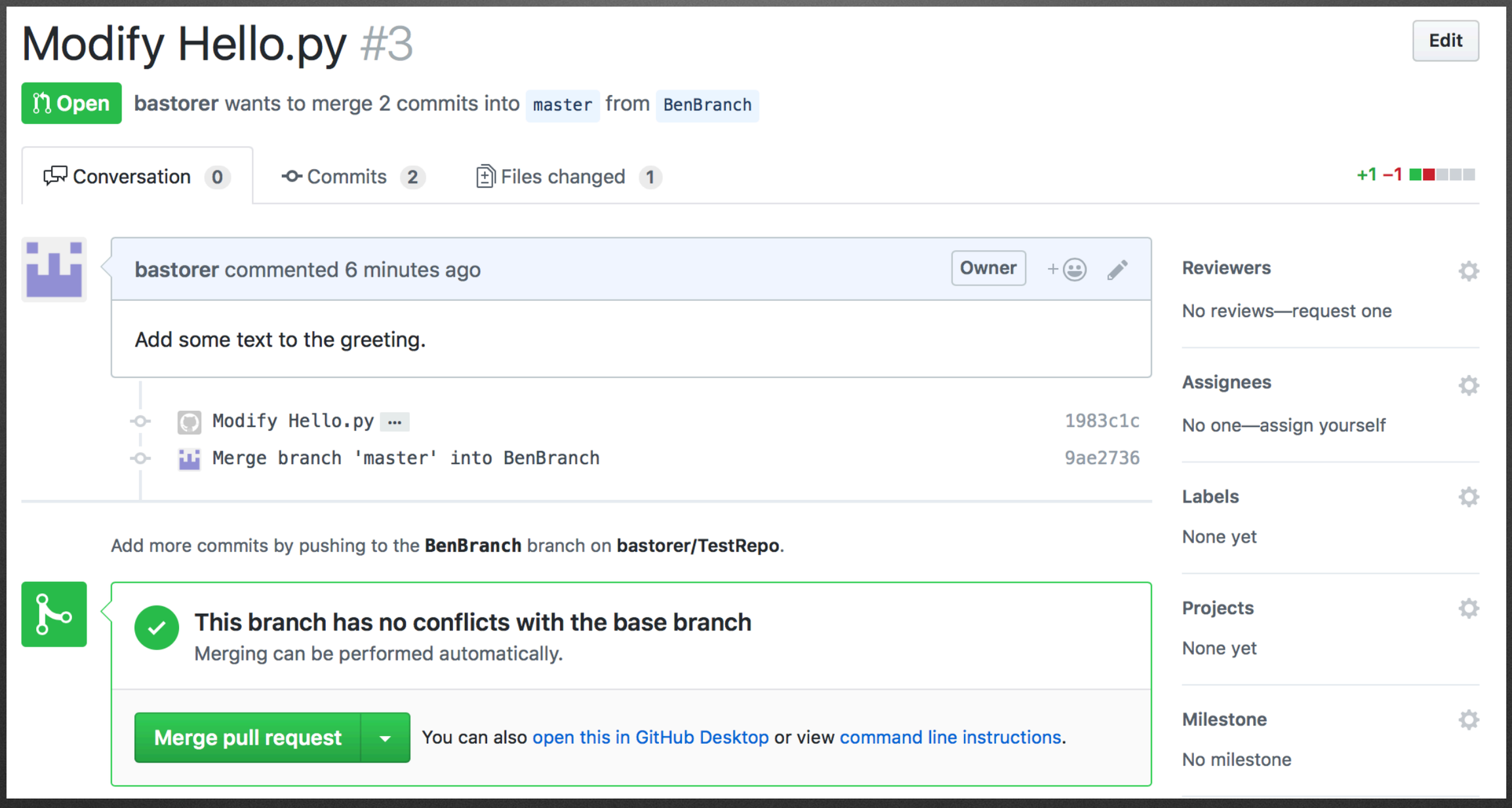
- Before resolving

- After resolving



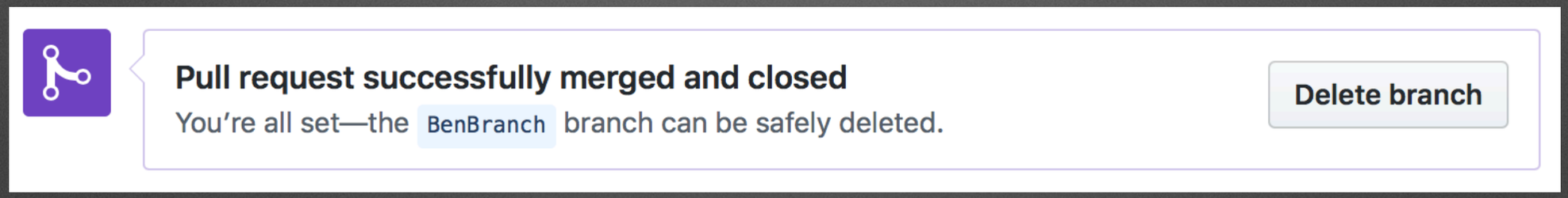
Handling Conflicts

- After resolving the conflicts, we can proceed to accept the pull request normally.



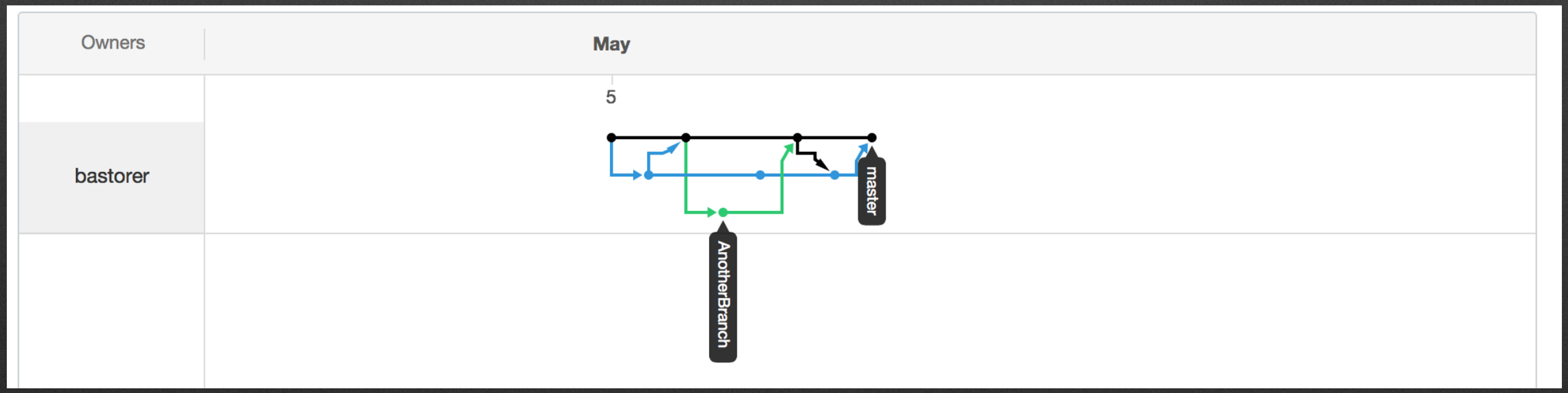
Pruning Branches / Network

- After resolving pull requests, GitHub gives you the option to *prune* the merged branch. If you no longer need the branch, then pruning helps to maintain a clean network for your repository.



Pruning Branches / Network

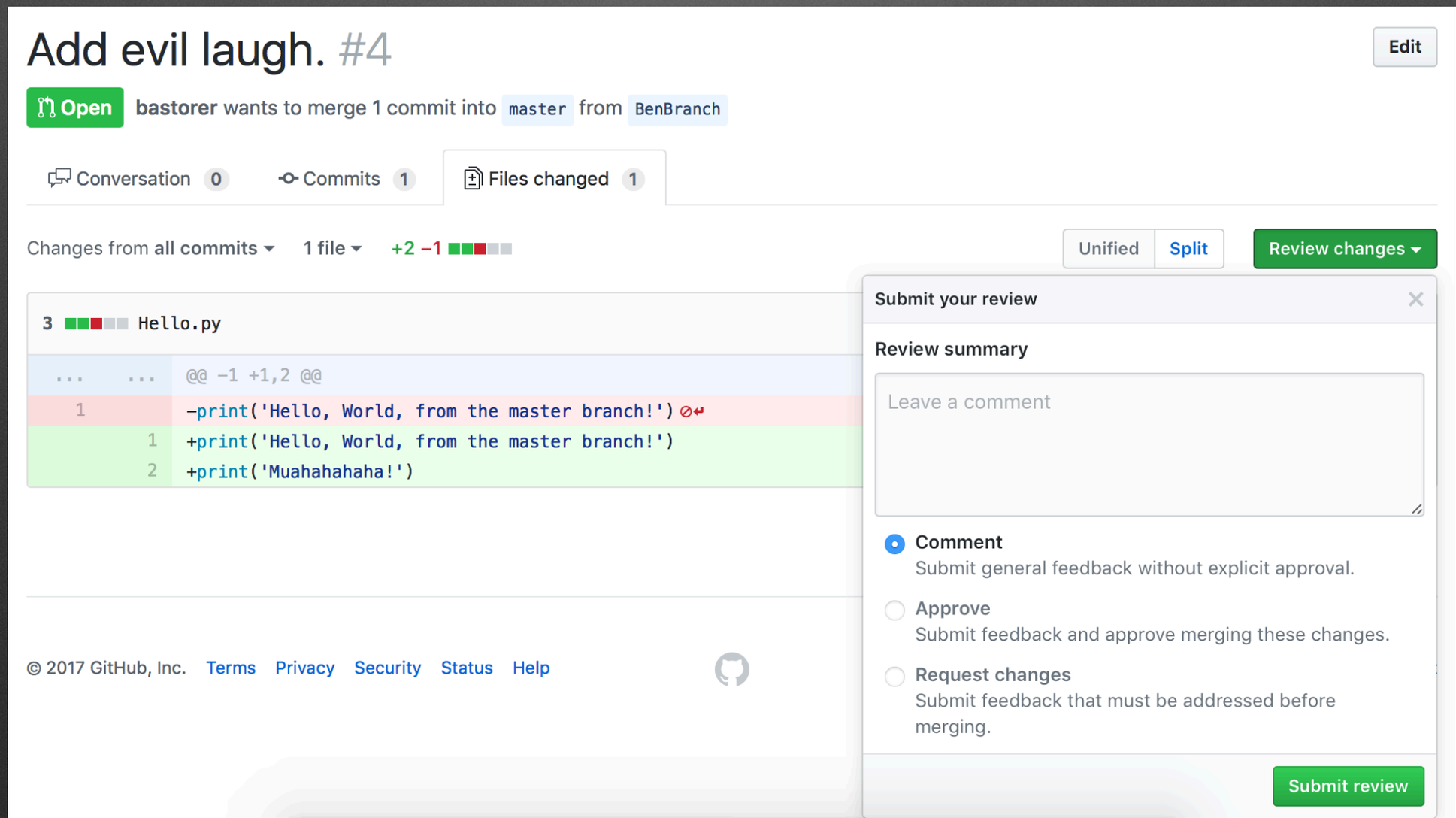
- The network (under *graphs* tab on GitHub) shows the branch/commit structure for your repo.
  - Each circle indicates a commit, each pathway indicates a branch.
- The *master* branch is black.
- We created a branch (blue), and merged it into the master.
- Then we created another branch (green), which we then merged into master.
- The last two paths show the conflict resolution used for the final merge.



Code Review

- Before accepting a pull request, it is a good idea to go over the code to make sure that you agree with the changes.

- Fortunately, you'll be able to talk to each other about the submitted code, but GitHub also provides a code review tool.



Resources

- Installing Git: https://git-scm.com
- Set up an account: GitHub.com (web interface and command-line), git.uwawterloo.ca (command-line)
- GitHub Desktop App (desktop.github.com)
- A useful git demo: try.github.io
- Two useful review sheets (for command-line):
  - https://education.github.com/git-cheat-sheet-education.pdf
  - https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf
- GitHub documentation is very user-friendly
- And, as always, Google! (i.e stack exchange)

More on Commits

- Generally speaking, it's better to commit more often than less often.
- Each commit should only do one main thing. For example, if you created a new page, and changed the colour scheme on another page, that should be two commits: one for the colour change, one for the new page.