

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.

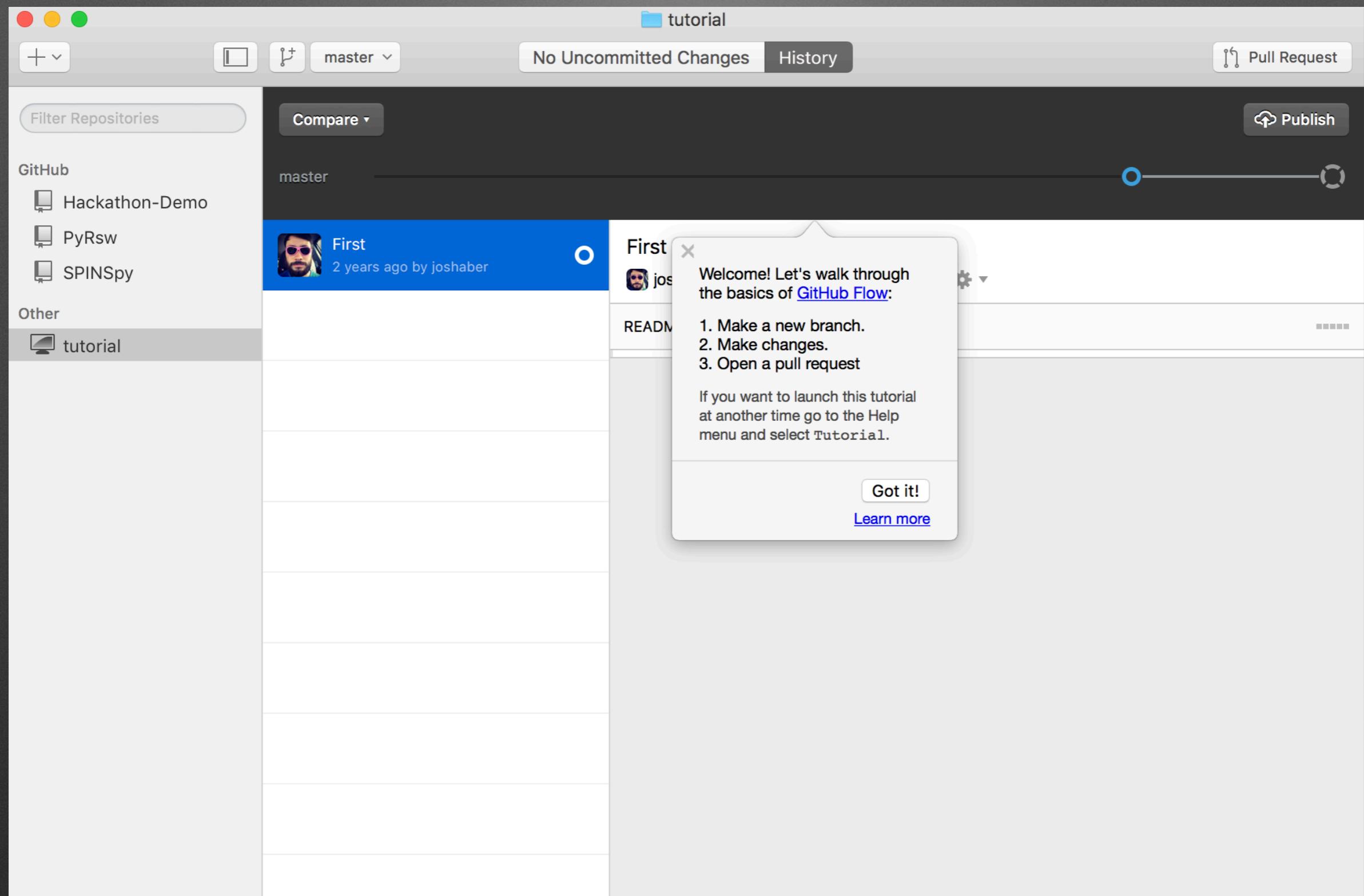
More on Branches

- 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 be merged using a *pull request*.

Let's work through some of the main features.

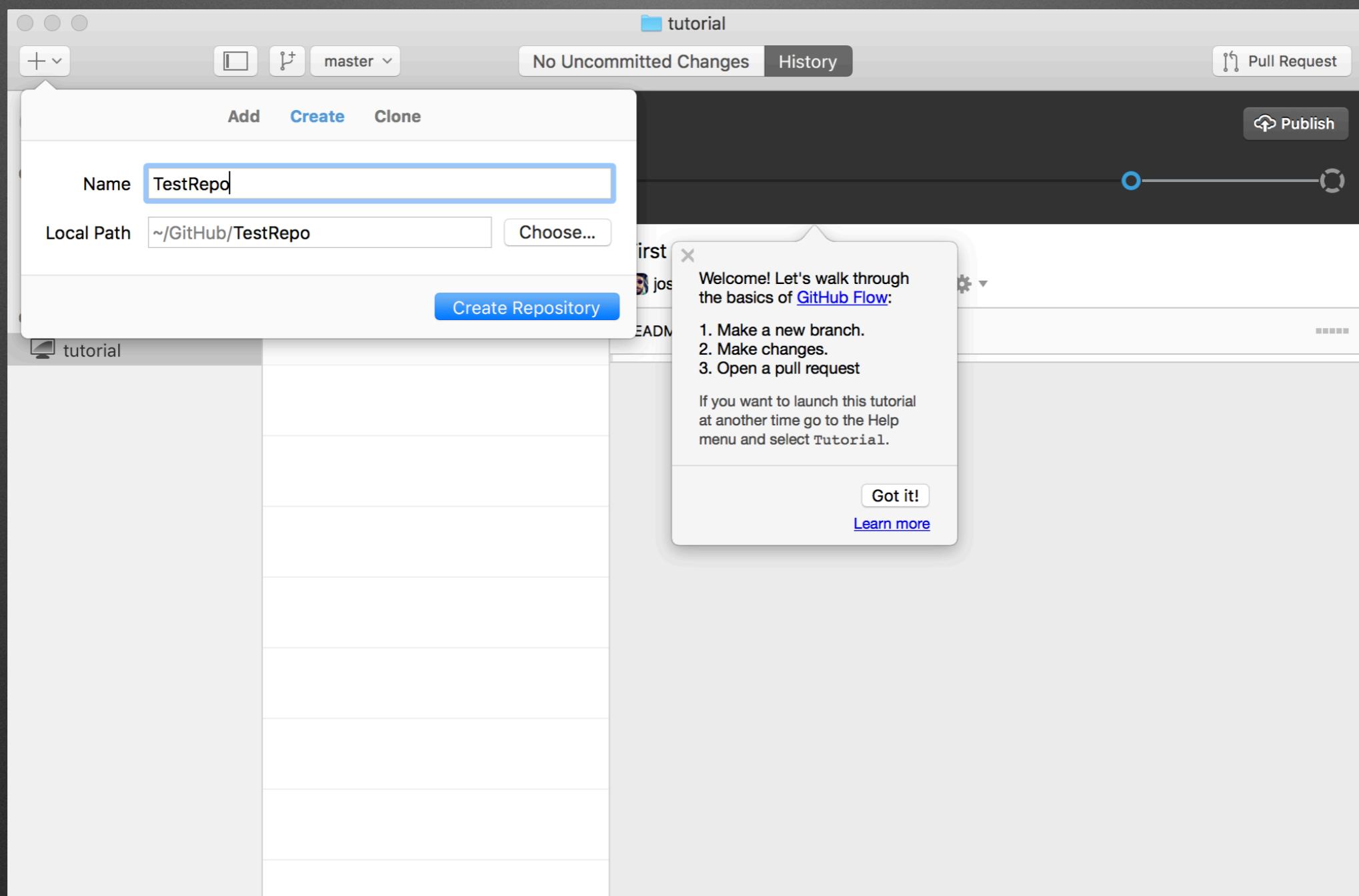
- Will be using the GitHub Desktop App
 - You can certainly use git through the command-line, but this workshop focuses on the GUI method.
- We'll look at:
 - creating/accessing a repo
 - creating commits and branches
 - merging branches back into the master branch
 - resolving code conflicts
 - reviewing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

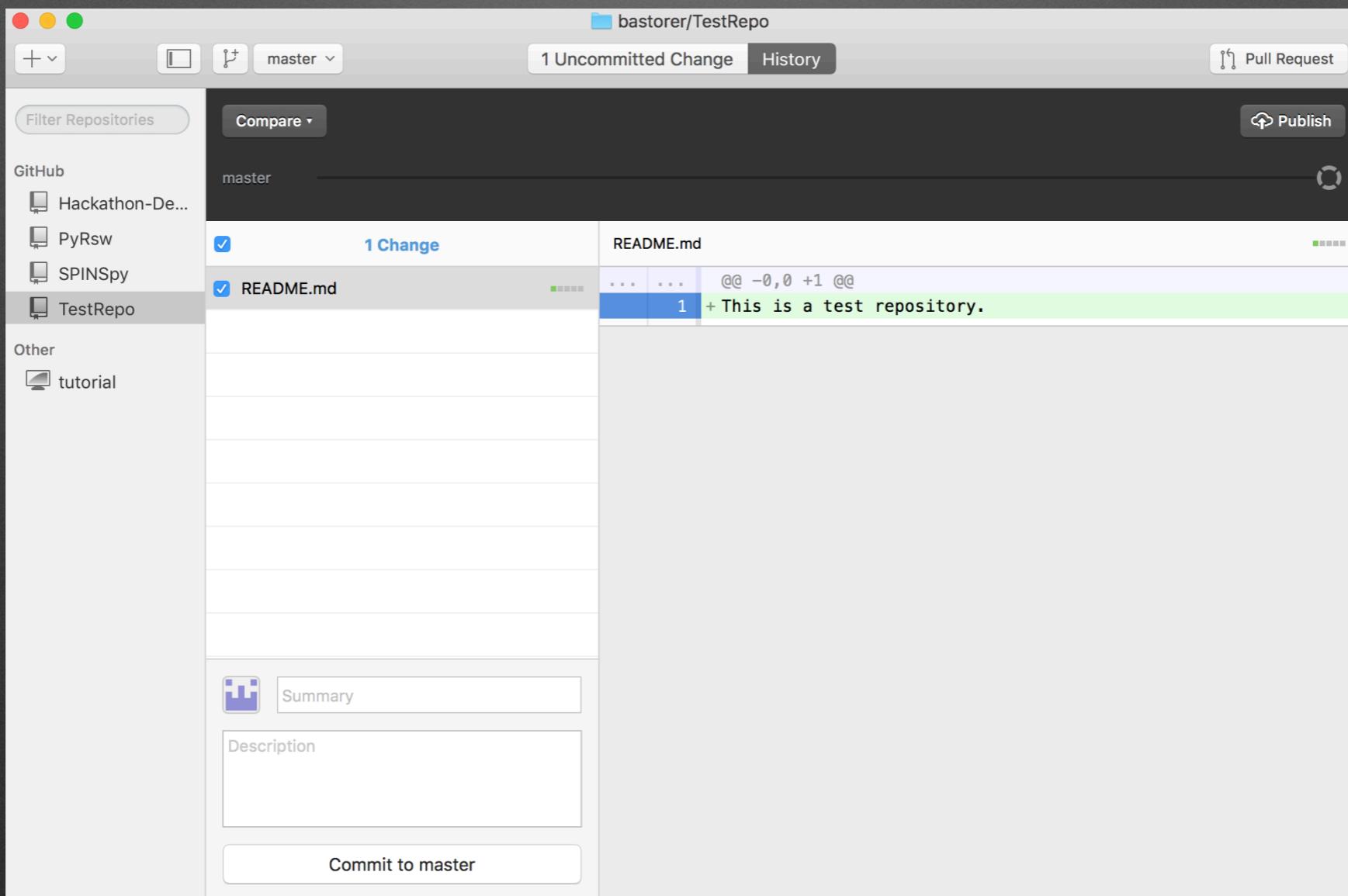
- Search for the desired repo on GitHub, select clone
- Access this presentation at: <https://github.com/bastorer/TestRepo>

The screenshot shows a GitHub repository page for 'ddeepwel / SPINS_deepwell'. The page includes navigation tabs for 'Code', 'Issues 4', 'Pull requests 0', 'Projects 2', 'Wiki', 'Pulse', and 'Graphs'. Key statistics are displayed: 125 commits, 1 branch, 0 releases, and 1 contributor. A dropdown menu for cloning the repository is open, showing options for 'Clone with HTTPS' (selected) and 'Use SSH', along with the URL https://github.com/ddeepwel/SPINS_deepwell. Below the stats, a list of recent commits is shown:

File	Description	Date
ddeepwel Merge branch 'Deepwell'	Merge branch 'Deepwell' of https://github.com/ddeepwel/SPINS_deepwell (indicated by ellipsis)
matlab	Renamed my_nice_contour to spins_contour for namespace consistency	5 years ago
src	Create & implement an easy to use diagnostic framework	3 years ago
systems	Update belize system config for MPI version update	5 months ago
.gitignore	Added boost build to make_deps for antique systems	4 years ago
ERRATA	Added ERRATA file for known bugs/misfeatures	3 years ago
README.md	Update README.md	5 months ago
make_deps.sh	Add -ftz for builds on winisk	4 years ago

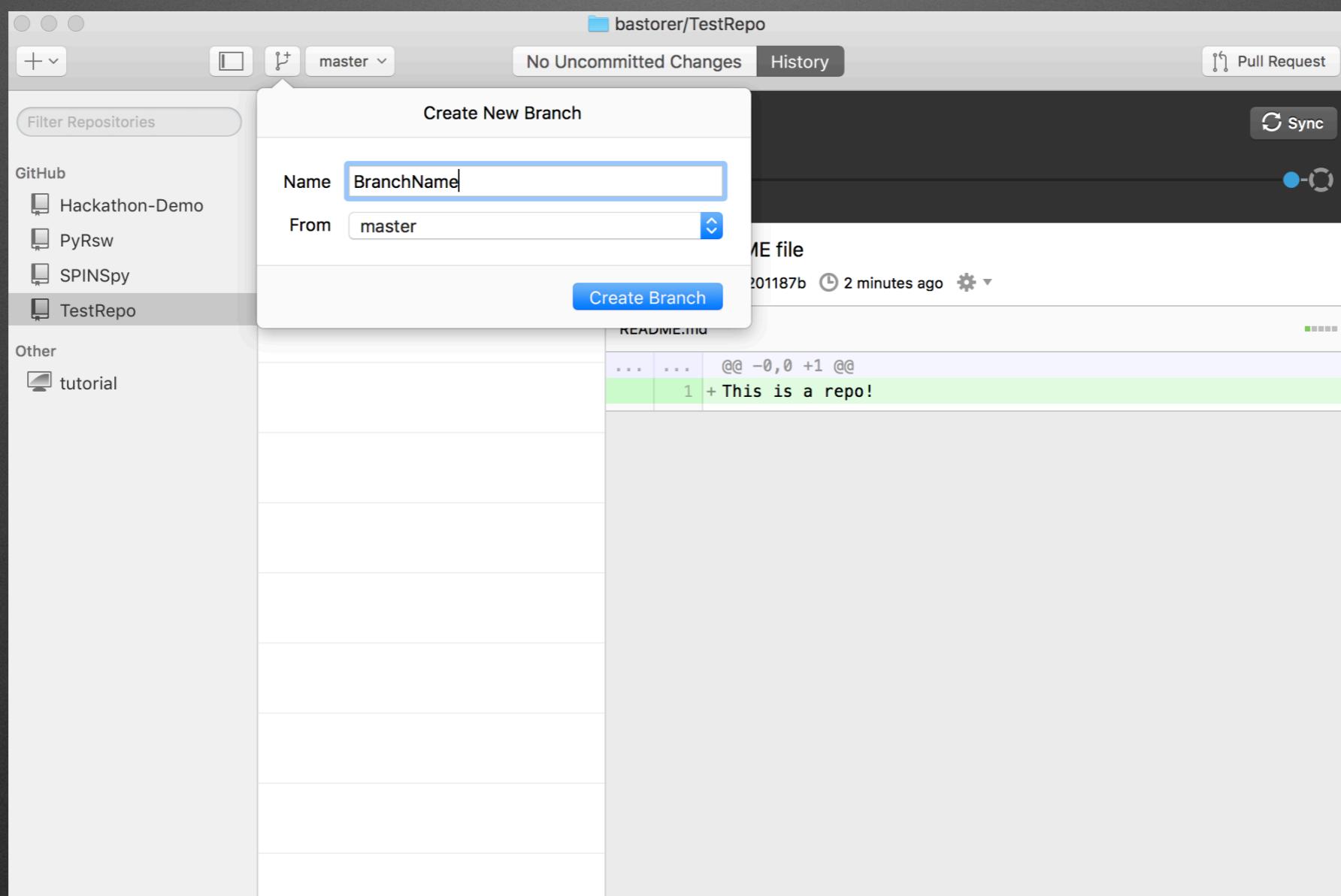
Making a Commit

- 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.



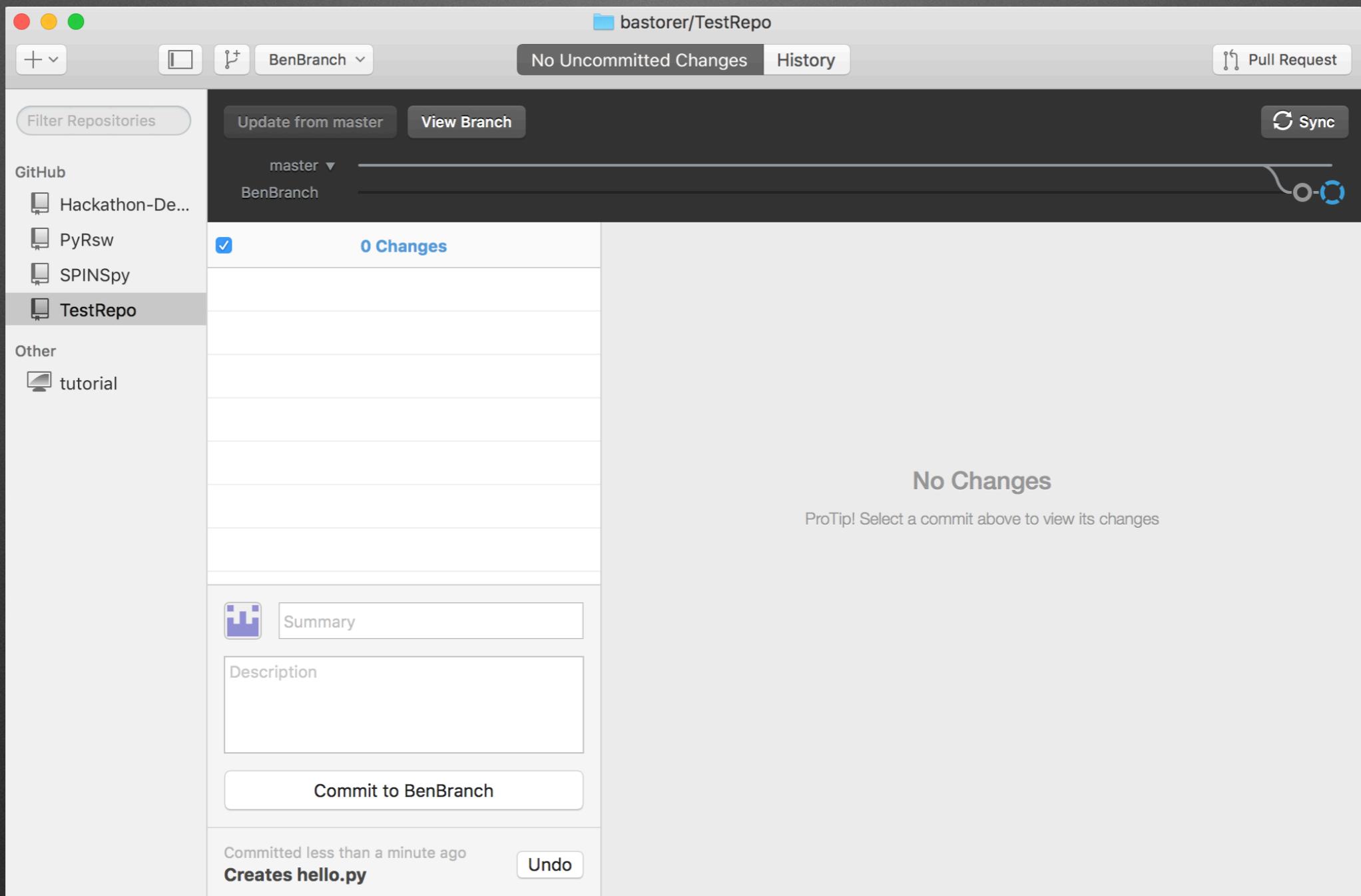
Making a Branch

- 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.

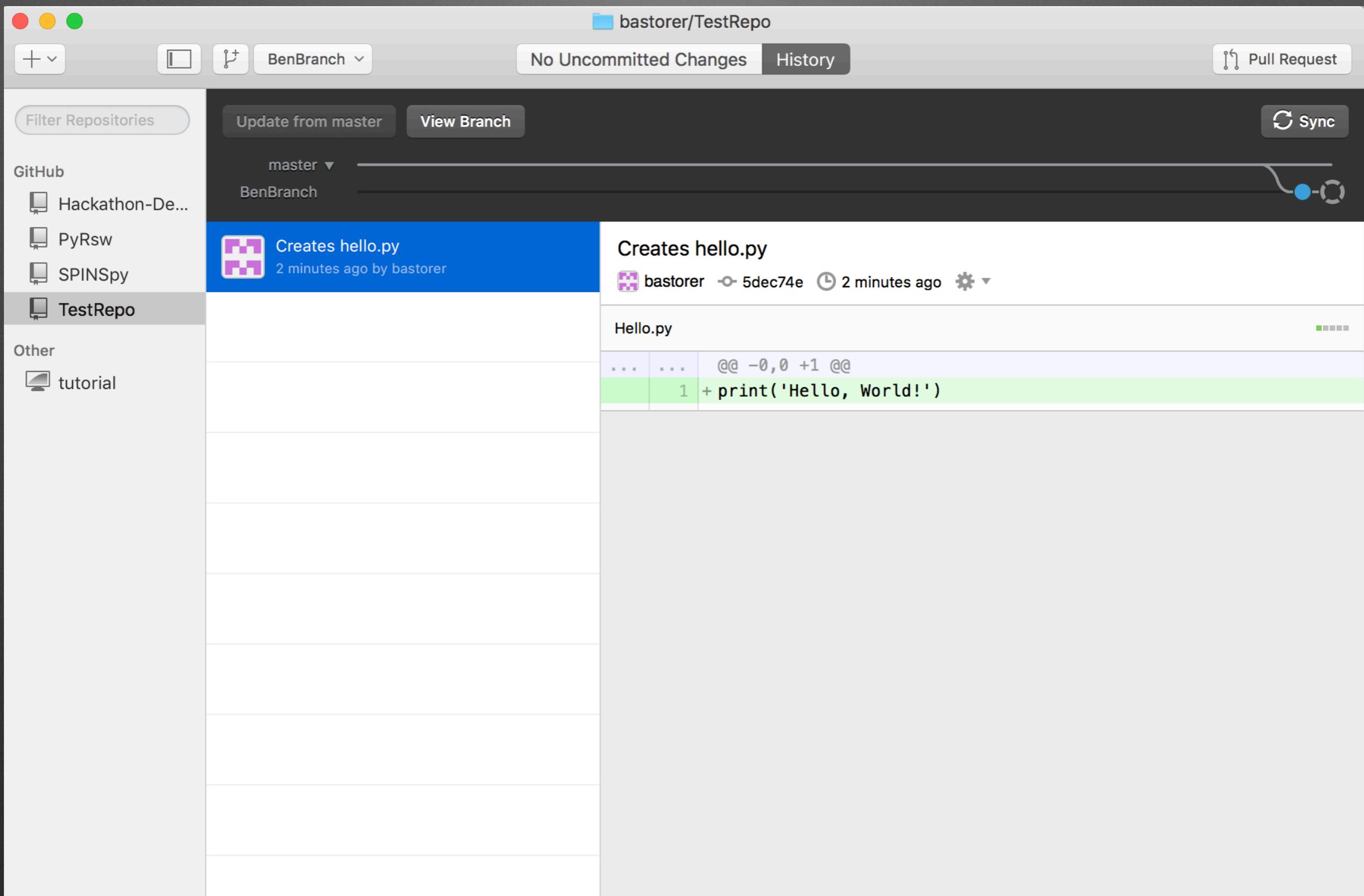


Commit to the Branch

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

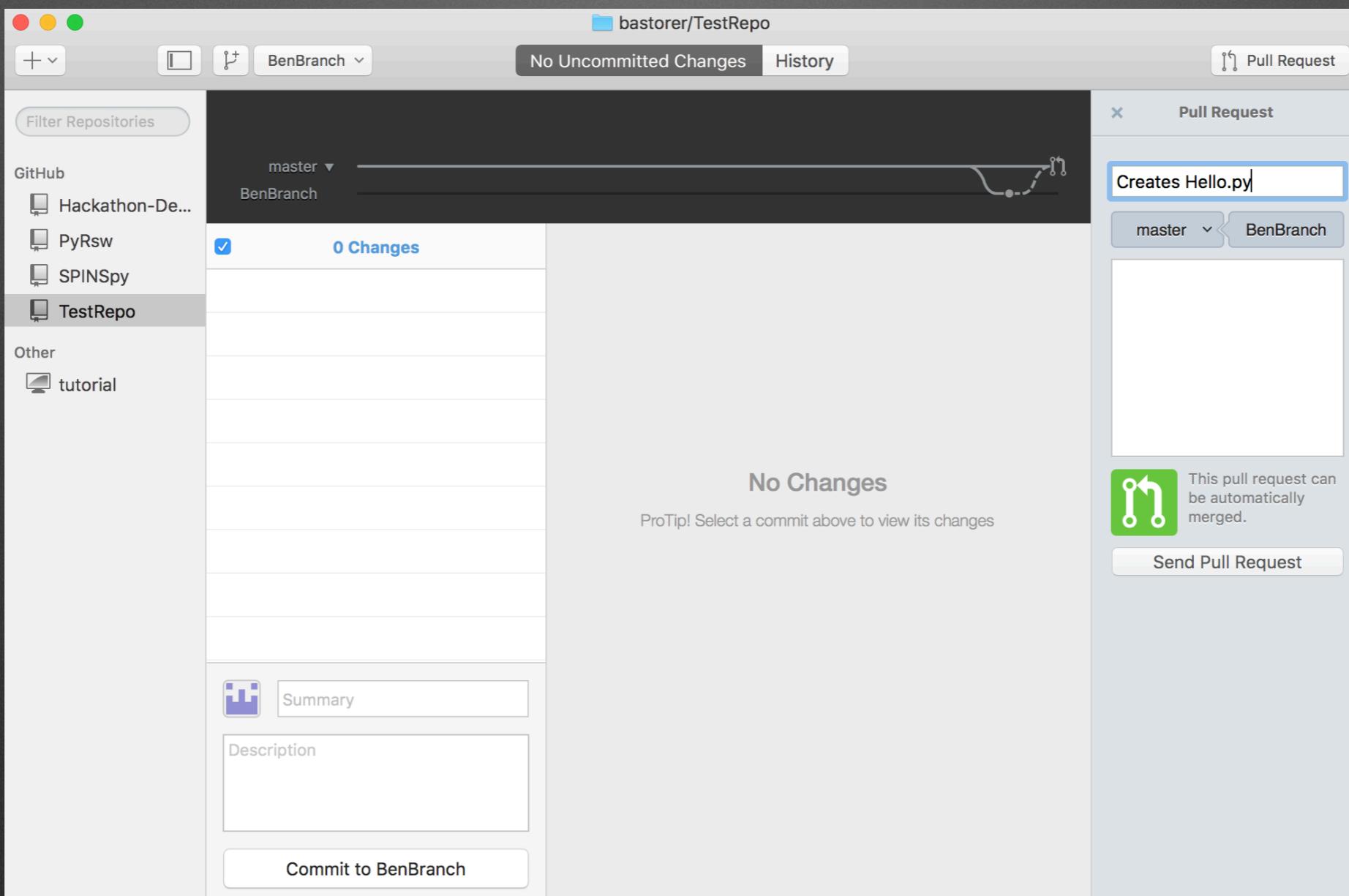


- Like the repo, need to *publish* the branch to push it to GitHub.
- Branch structure shown in Desktop App, *History* tab shows commits



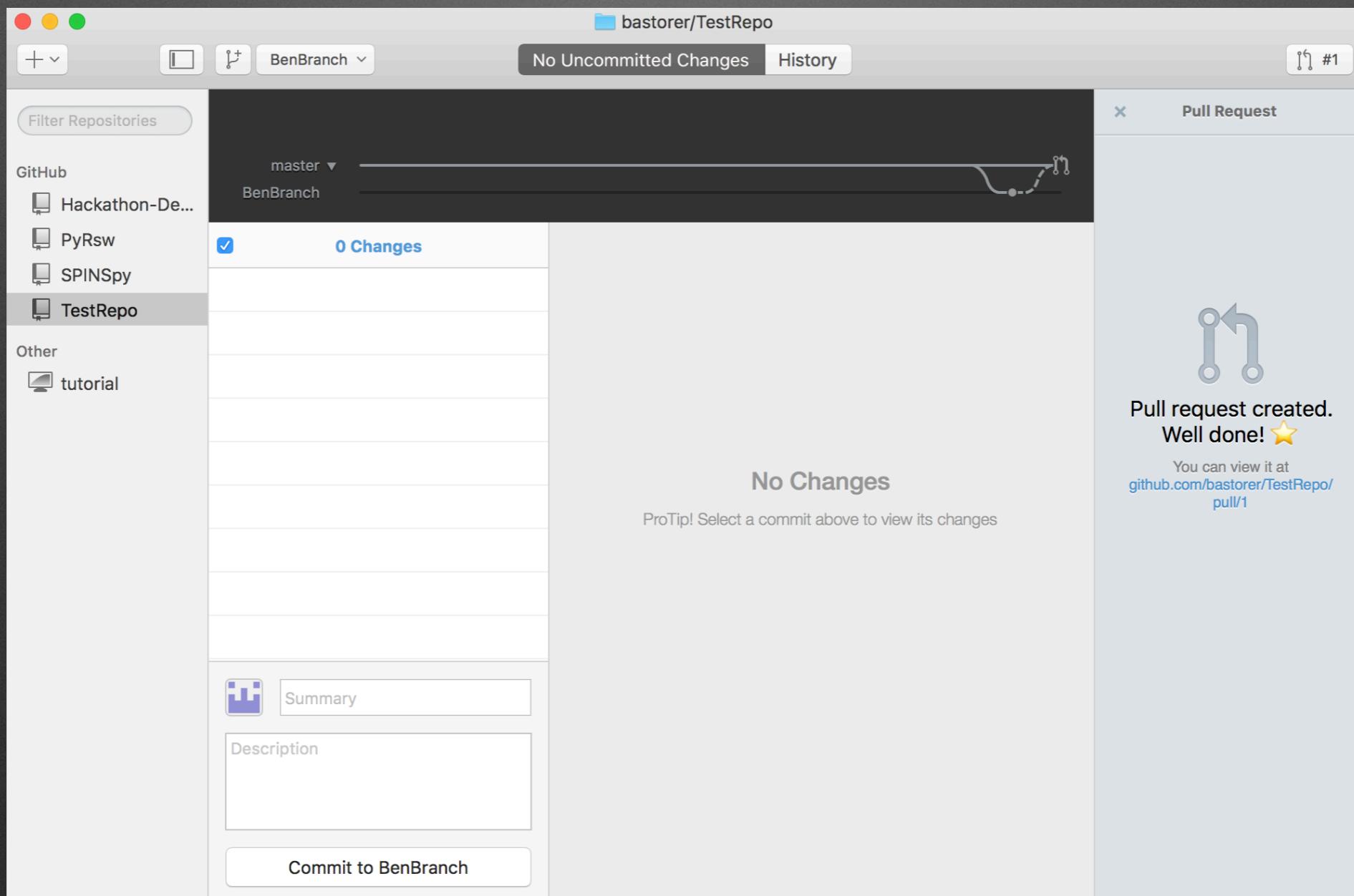
Merging the Branch

- 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.



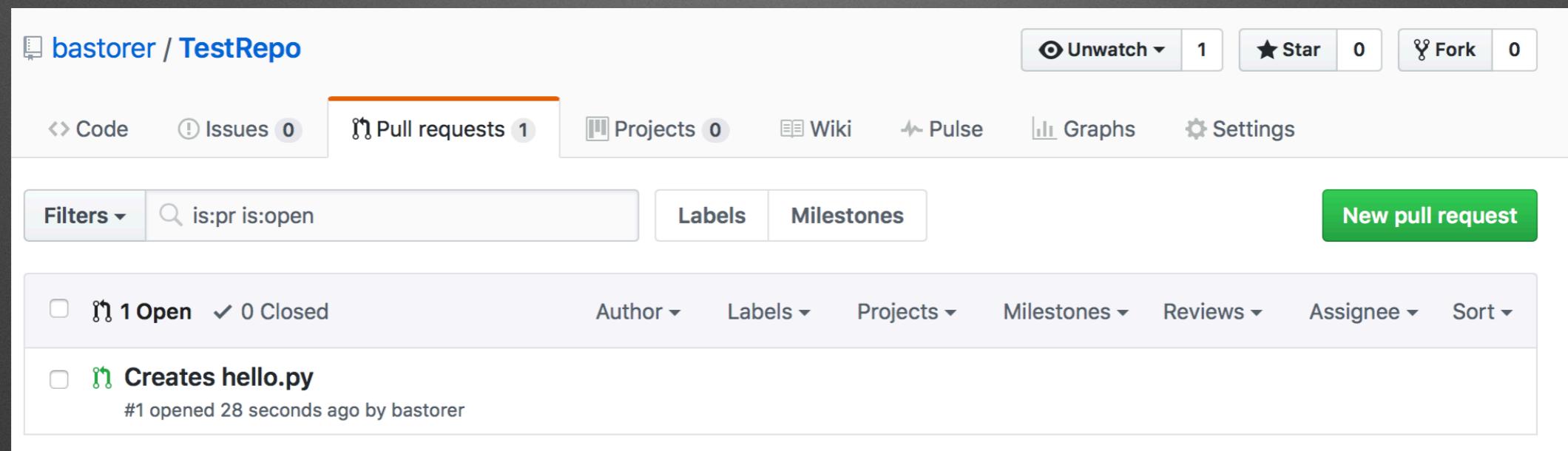
Merging the Branch

- 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

bastorer/TestRepo

+ BenBranch No Uncommitted Changes History Pull Request

Filter Repositories

GitHub

- Hackathon-De...
- PyRsw
- SPINSpy
- TestRepo**

Other

- tutorial

master BenBranch

Modified print statement.
2 minutes ago by bastorer

Creates Hello.py
10 minutes ago by bastorer

Modified print statement.
bastorer f0acd40 2 minutes ago

Hello.py

...	...	@@ -1 +1 @@
1	-	print('Hello, World!')
	1	+ print('Hello, World! I am on BenBranch.')

Pull Request

Modified print statement.

master BenBranch

Description

This pull request can't be automatically

Send Pull Request

The screenshot shows a GitHub desktop interface with a 'Pull Request' open. The repository is 'bastorer/TestRepo'. The 'BenBranch' tab is selected. The 'No Uncommitted Changes' button is active. On the left, there's a sidebar with a 'GitHub' section containing 'Hackathon-De...', 'PyRsw', 'SPINSpy', and the 'TestRepo' repository, which is currently selected. Below that is an 'Other' section with 'tutorial'. In the main area, a timeline shows the 'master' branch merging into the 'BenBranch' branch. Two commits are visible: one from 'bastorer' modifying the print statement and another from 'bastorer' creating the 'Hello.py' file. The 'Hello.py' file is shown with a conflict in the 'print' statement. The 'Pull Request' panel on the right has a title 'Modified print statement.' and a description field. It also includes a note: 'This pull request can't be automatically' and a 'Send Pull Request' button. A red icon with a circular arrow is displayed next to the note.

Handling Conflicts

The screenshot illustrates the GitHub pull request interface for handling conflicts.

Comment Section: A comment from user **bastorer** is shown, stating "Modified print statement." with commit hash **f0acd40**. The interface includes an "Owner" button, a smiley face icon, and a pencil icon.

Conflict Alert: A warning message indicates that "This branch has conflicts that must be resolved". It provides links to the [web editor](#) or the [command line](#) to resolve conflicts. A "Resolve conflicts" button is present. Below this, a section titled "Conflicting files" lists "Hello.py". A "Merge pull request" button with a dropdown arrow is also visible.

Comment Form: A "Leave a comment" input field is provided with rich text editing tools above it. The tools include "Write" and "Preview" tabs, and icons for bold, italic, quote, list, and other common text operations. A note at the bottom says "Attach files by dragging & dropping or [selecting them](#)".

Footer: At the bottom, a note states "Styling with Markdown is supported". Buttons for "Close pull request" and "Comment" are located at the bottom right.

Handling Conflicts

- Before resolving

The screenshot shows a conflict viewer for a file named `Hello.py`. At the top, it displays "1 conflict". Below the conflict, there is a code block with the following content:

```
1 <<<<< BenBranch
2 print('Hello, World! I am on BenBranch.')
3 =====
4 print('Hello, World! I am on another branch.')
5 >>>>> master
6
```

The code is color-coded to indicate the source of each line: red for BenBranch, blue for another branch, and black for master.

- After resolving

The screenshot shows a conflict resolution interface. At the top, it says "#3: Modified print statement. > Conflicts". On the right, there are buttons for "Resolved all conflicts" (green checkmark) and "Commit changes" (green button). Below this, a table lists the conflicting files:

1 conflicting file	Resolved
Hello.py Hello.py	✓ Resolved

The "Hello.py" row has a green checkmark icon next to it. In the code editor area, the file `Hello.py` contains the following single line:

```
1 print('Hello, World! I am on the master branch.')|
```

Handling Conflicts

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

Modify Hello.py #3 Edit

Open bastorer wants to merge 2 commits into `master` from `BenBranch`

Conversation 0 Commits 2 Files changed 1 +1 -1

bastorer commented 6 minutes ago

Add some text to the greeting.

Owner +

Reviewers No reviews—request one

Assignees No one—assign yourself

Labels None yet

Projects None yet

Milestone No milestone

Modify Hello.py ... 1983c1c
Merge branch 'master' into BenBranch 9ae2736

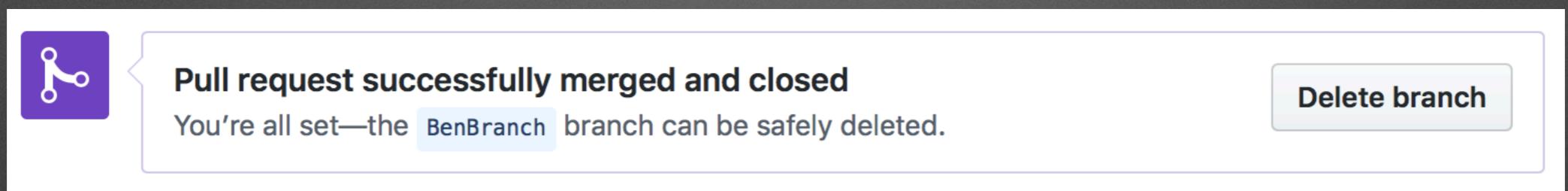
Add more commits by pushing to the **BenBranch** branch on **bastorer/TestRepo**.

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](#).

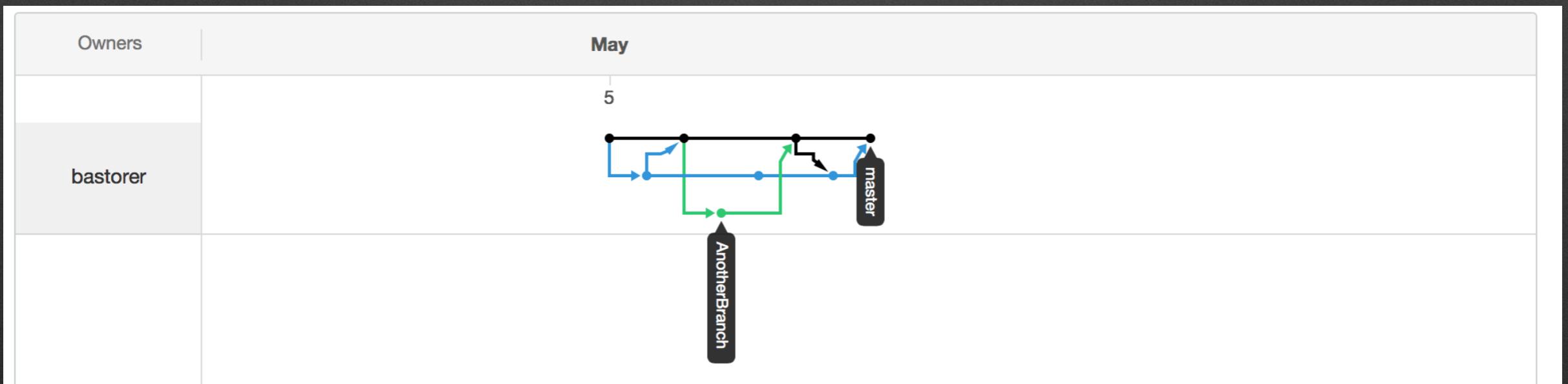
Pruning Branches / Network

- 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.

Add evil laugh. #4

[Open](#) bastorer wants to merge 1 commit into `master` from `BenBranch`

Conversation 0 Commits 1 Files changed 1

Changes from all commits ▾ 1 file ▾ +2 -1

3	>Hello.py
...	@@ -1 +1,2 @@
1	-print('Hello, World, from the master branch!')
1	+print('Hello, World, from the master branch!')
2	+print('Muahahahaha!')

Unified Split Review changes ▾

Submit your review

Review summary

Leave a comment

Comment
Submit general feedback without explicit approval.

Approve
Submit feedback and approve merging these changes.

Request changes
Submit feedback that must be addressed before merging.

Submit review

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.