Lecture 1: Build your working environment

COSC 526: Introduction to Data Mining

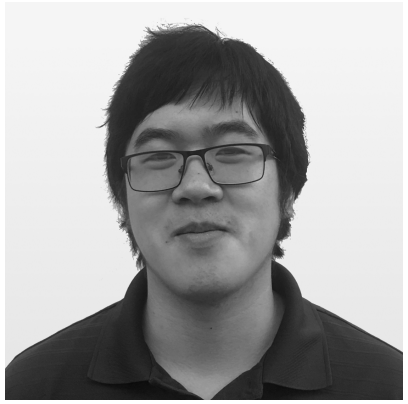
THE UNIVERSITY OF TENNESSEE KNOXVILLE

Instructors:

Michela Taufer

Assistants to the Instructor:

Ian Lumsden



Nigel Tan

Kae Suarez

Paula Olaya

Leo Valera

Build your working environment

- We **use git & GitHub** to distribute & collect assignments as well as other class materials (e.g., slides, code, and datasets)
  - Create a GitHub username (if you have not one)
  - Install git and github desktop on your laptop
- We use **Jupyter notebook** for our assignments and project
  - Follow the steps in the file *Lecutre01_student_notes* to install Jupyter
- We use **XSEDE Jetstream** as our cloud platform for assignments and the semester project
  - Introduce its use in the 2nd lecture
  - Create accounts and provide you with access to the cloud in the 4th lecture

GitHub and Git

- **GitHub:** web-based hosting service for **version control**
  - Use it to distribute and collect assignments, to share class materials (e.g., slides, codes, and datasets)
  - Provide us with your GitHub username
- **Git:** software used by GitHub
  - Install git on your laptop
- **Class GitHub Repository:**
  - Clone the course repository
  - https://github.com/CISC879-BigData/courses-UTK-COSC526-S21

Git and GitHub

Version Control System

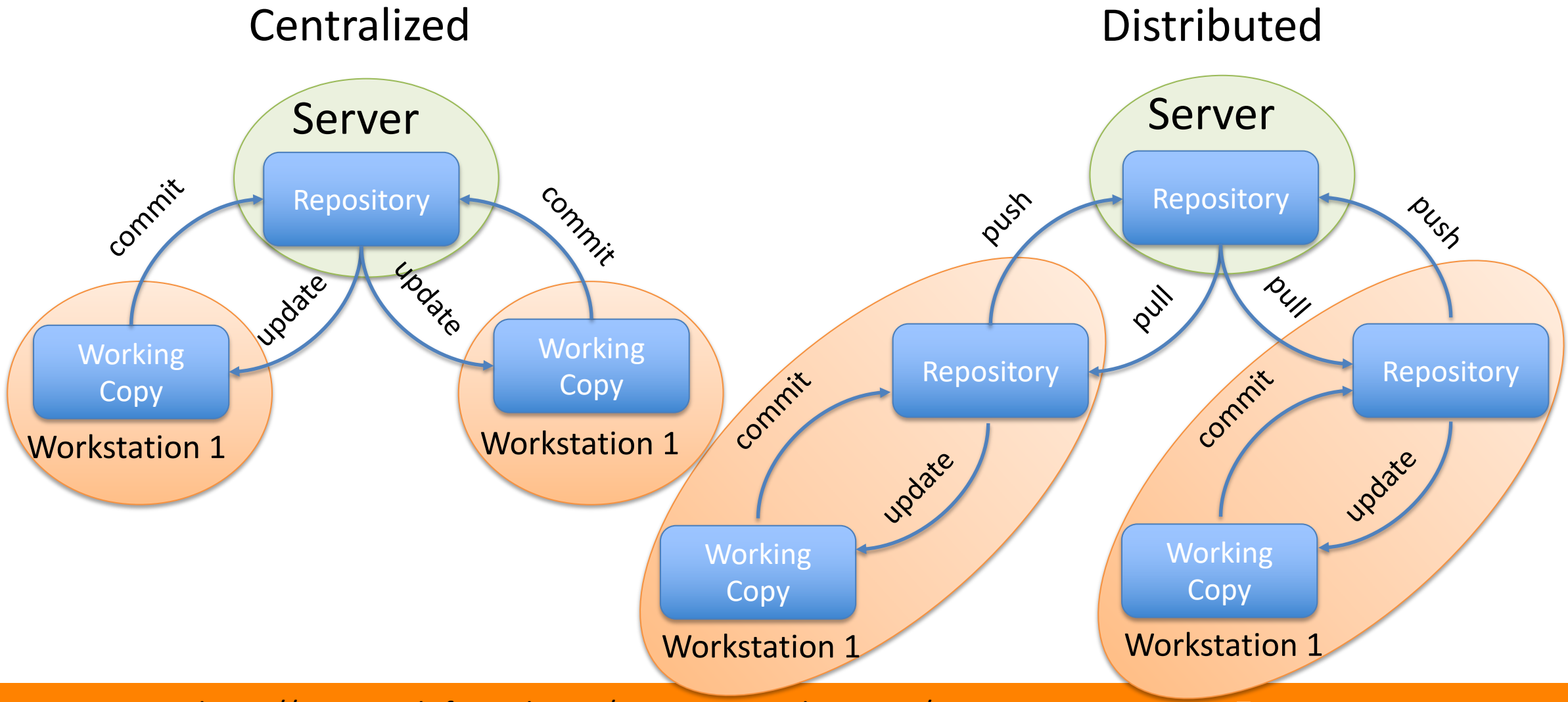
- **Version control system**

  "Version control is a **system** that records changes to a file or set of files over time so that you can recall specific **versions** later."

  Source: About Version Control – Git; git-scm.com › Getting-Started-About-Version-Control

- Version control systems can be:
  - Centralized
  - Distributed

Centralized and Distributed Version Control System



Source: https://www.geeksforgeeks.org/version-control-systems/

Why we use Version Control System

- Multiple people can work simultaneously on a single project
  - Everyone works on and edits their own copy of the files and it is up to them when they wish to share the changes made by them with the rest of the team.
- Work done simultaneously by different members of a team can be integrated
  - When conflicting edits are made by two people to the same line of a file, then human assistance is requested by the version control system in deciding what should be done.
- One person can use multiple computers to work on a project
- Users have access to the historical versions of a project
  - This is insurance against computer crashes or data loss
  - If any mistake is made, one can easily roll back to a previous version. It is also possible to undo specific edits that too without losing the work done in the meanwhile
  - It can be easily known when, why, and by whom any part of a file was edited

Source: https://www.geeksforgeeks.org/version-control-systems/

Git and GitHub

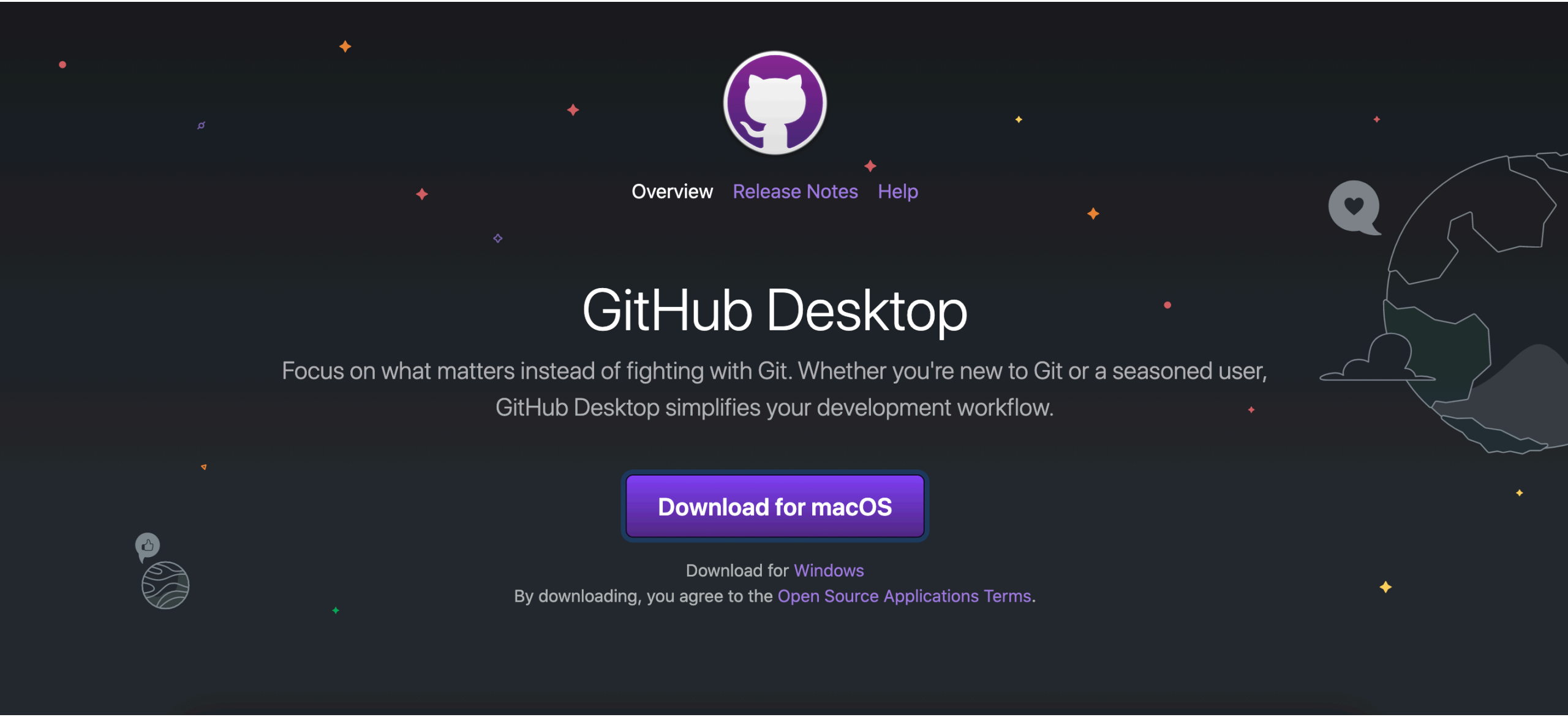
- Git: a free, open source distributed **version control system**
- GitHub: cloud hosted git repositories

Material built from:

http://rogerdudler.github.io/git-guide/
https://marklodato.github.io/visual-git-guide/index-en.html
https://kbroman.org/github_tutorial/pages/init.html
https://kbroman.org/github_tutorial/pages/routine.html

Install git on your laptop (if you have not yet)

- Windows or Mac
  - Download the GitHub Desktop application: https://desktop.github.com
  - Find GitHub's installation help here:
    https://help.github.com/desktop/guides/getting-started-with-github-desktop/
- Linux or Mac
  - Install the command-line interface (CLI) of git
  - Find help here: https://www.atlassian.com/git/tutorials/install-git
- Log into github



Source: https://desktop.github.com

Welcome to GitHub Desktop

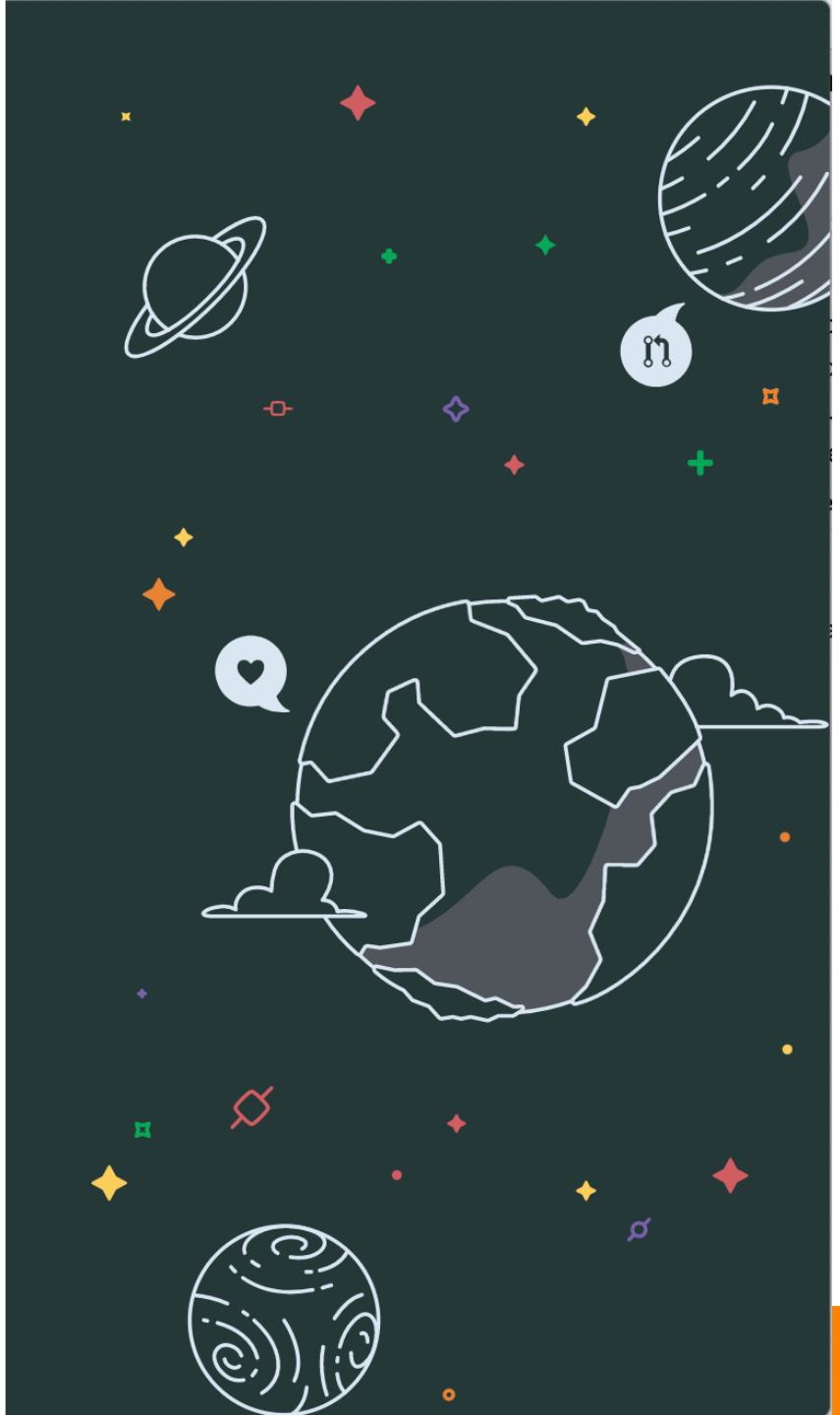
GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise Server. Sign in below to get started with your existing projects.

New to GitHub? **Create your free account.**

Sign in to GitHub.com

Sign in to GitHub Enterprise Server

Skip this step



Connect new repos to github

- Go to github
- Log in to your account

You don’t have a github account?

Connect new repos to github

- Go to github
- Log in to your account

You don’t have a github account?

- Get a github account
  - Go to the GitHub Sign Up page https://github.com/join
  - Create a free account

Connect new repos to github

- Go to github
- Log in to your account

You don't have a github account?

- Get a github account
  - Go to the GitHub Sign Up page https://github.com/join
  - Create a free account
- Share your github account with us:
  - Complete form **https://forms.gle/BBdtpSQeYaeut3ru9**

Use git and GitHub

- The routine use of git involves just a few commands:
  - initiate a repository
  - add and commit
  - push and pull
  - status
  - diff
- You can deal with git and github via:
  - GitHub Desktop (GUI)
  - Command Line (CLI)

GitHub Desktop Application

Welcome to GitHub Desktop

GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise Server. Sign in below to get started with your existing projects.

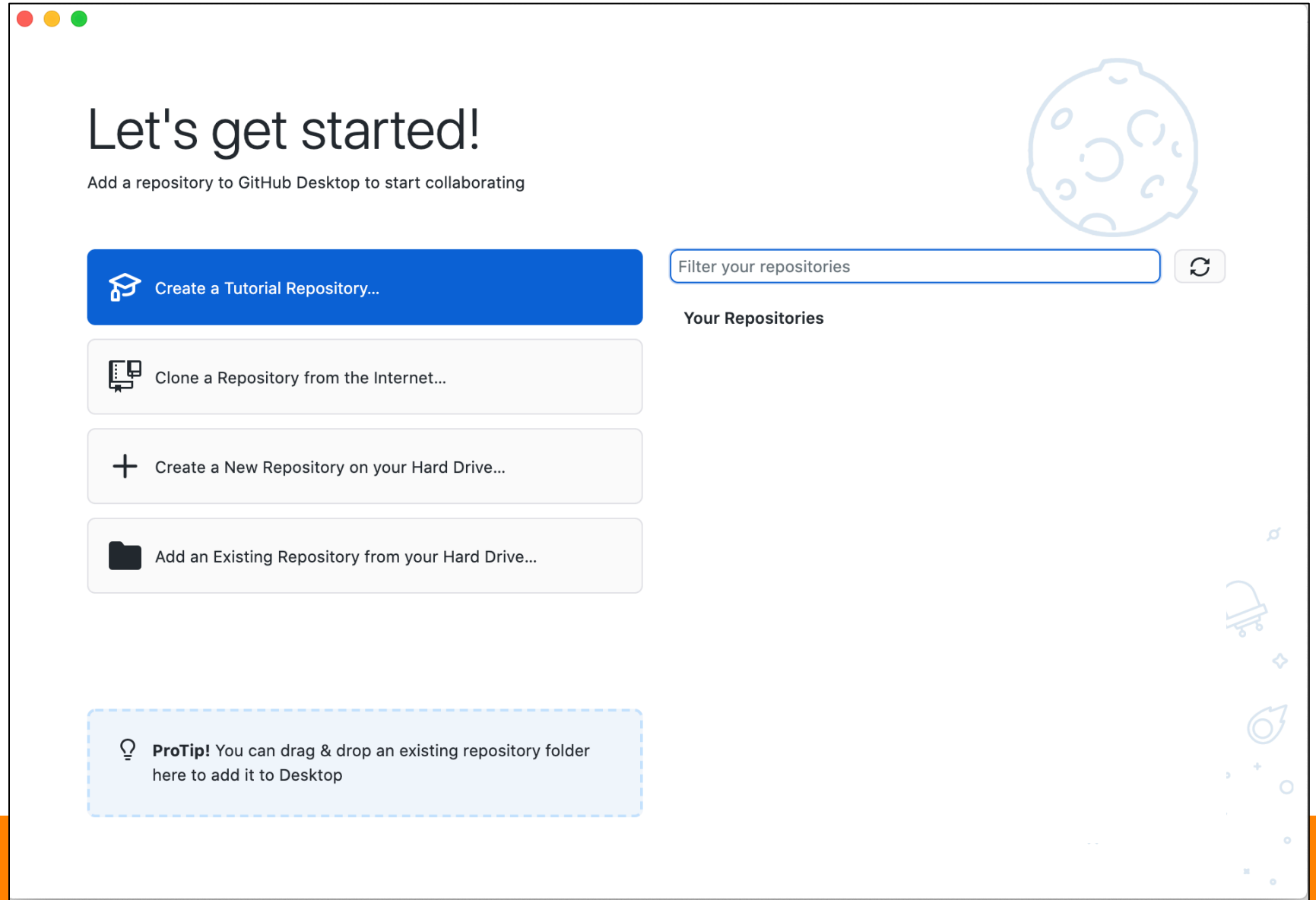
New to GitHub? **Create your free account.**

Sign in to GitHub.com

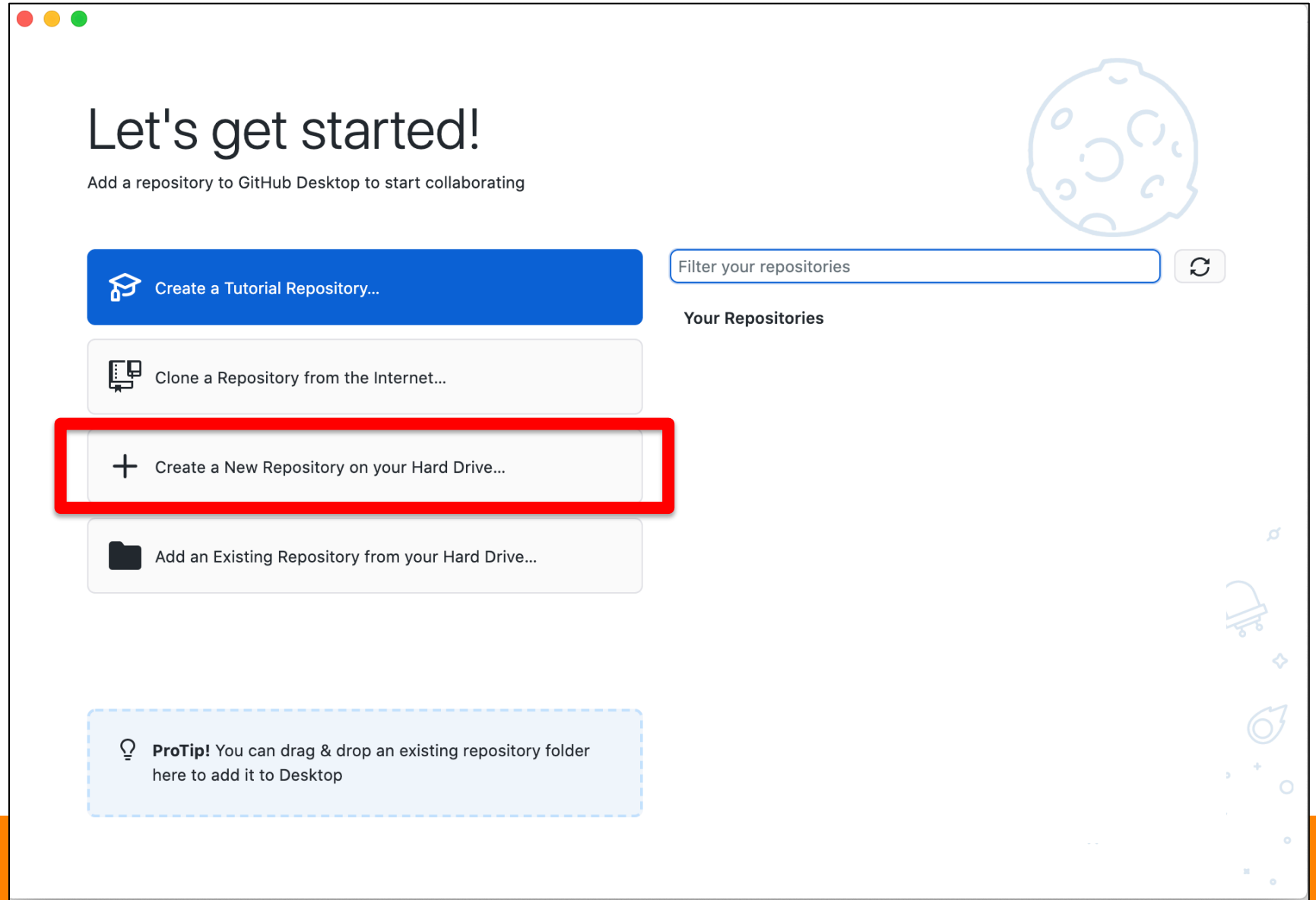
Sign in to GitHub Enterprise Server

Skip this step

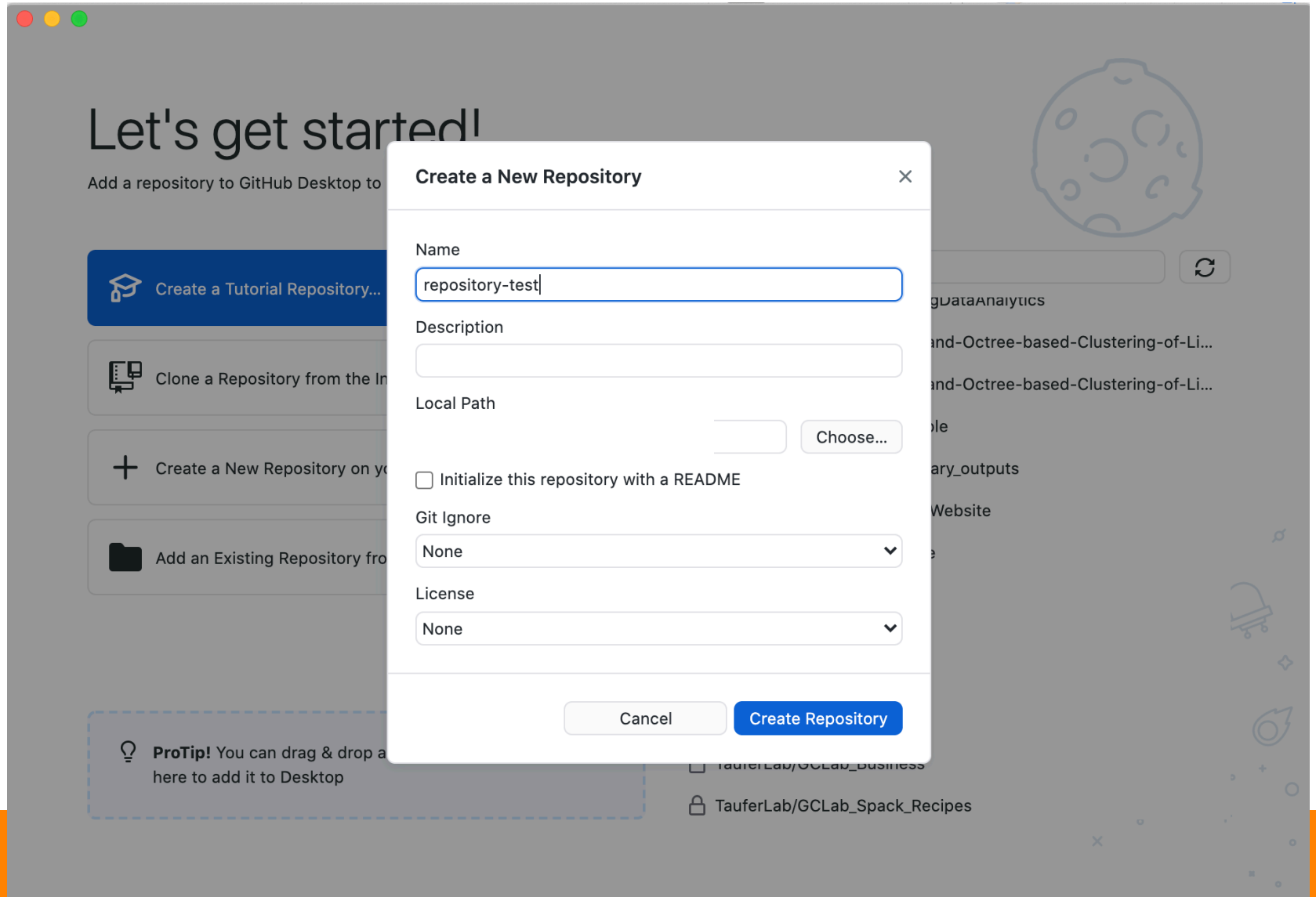
GitHub Desktop Application: Let's start



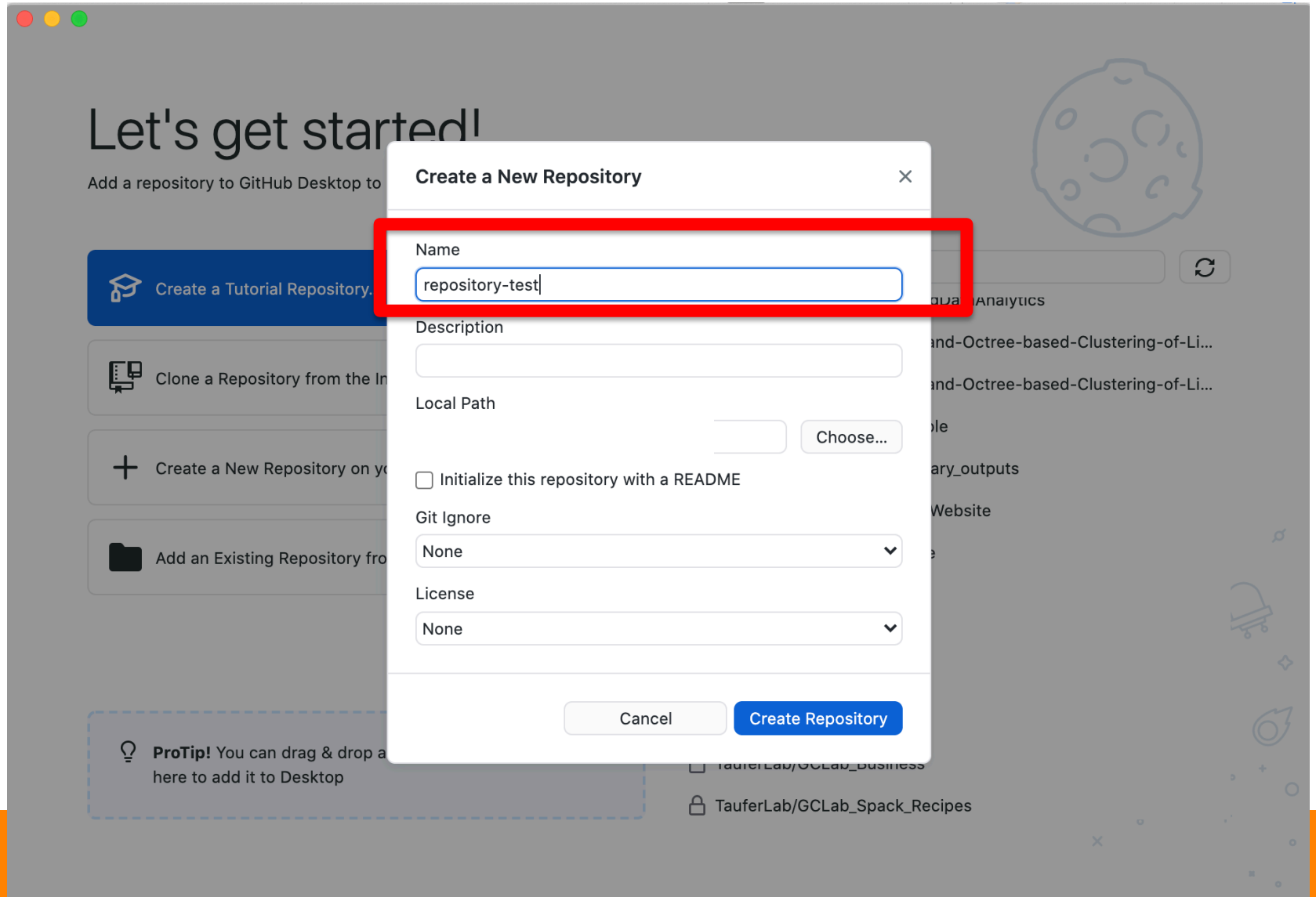
GitHub Desktop Application: Let's start



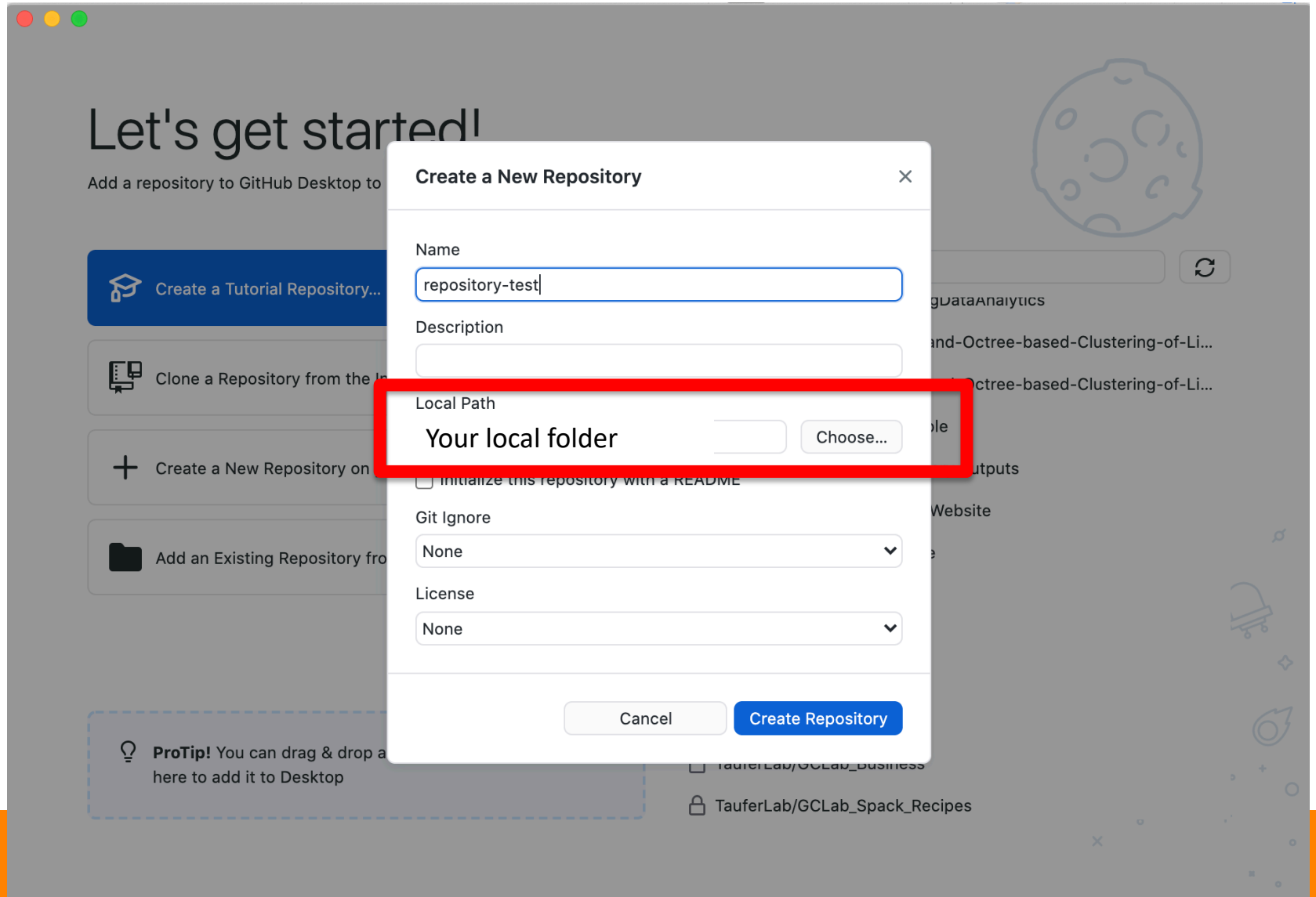
Create a Repository (I)



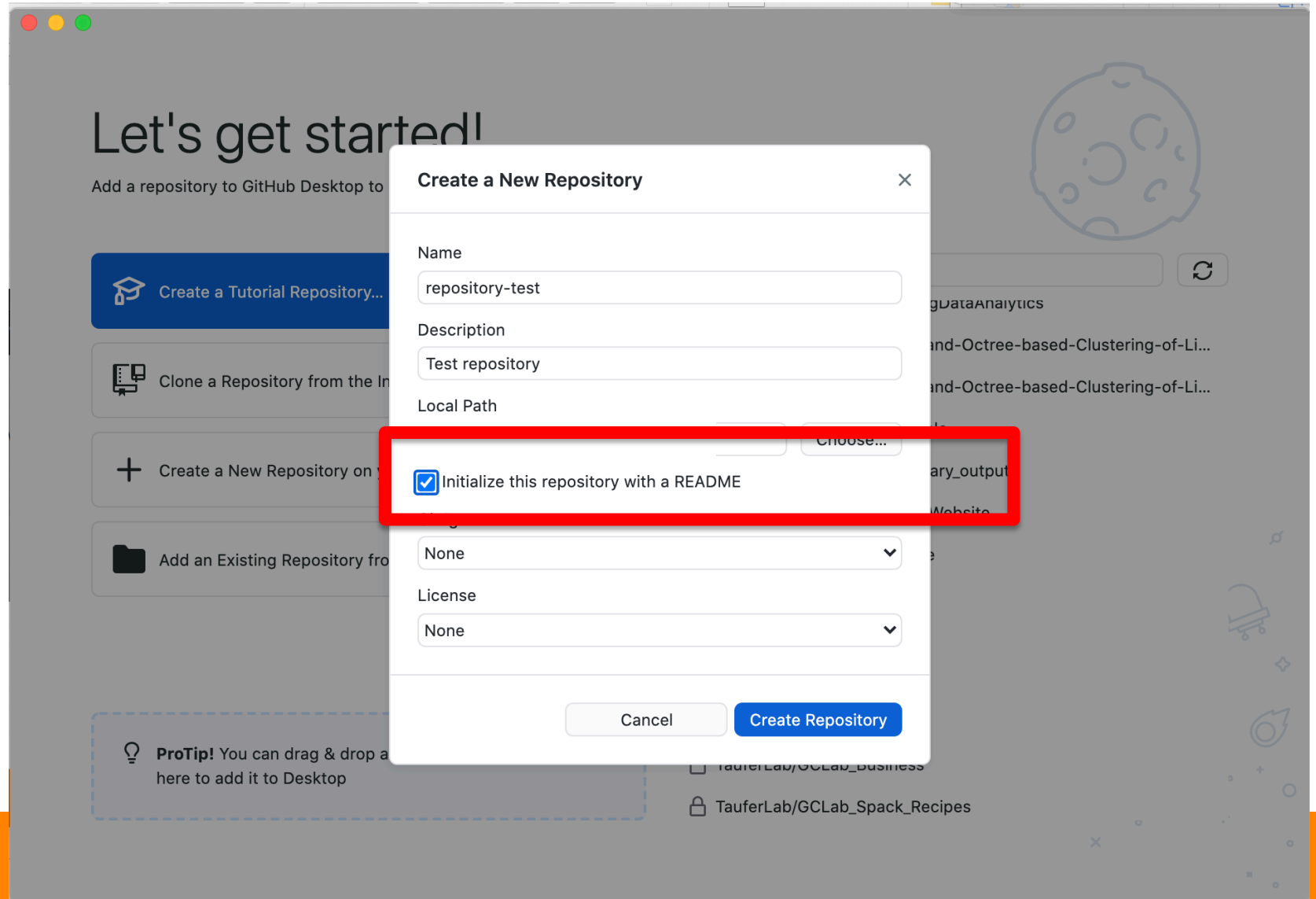
Create a Repository (II)



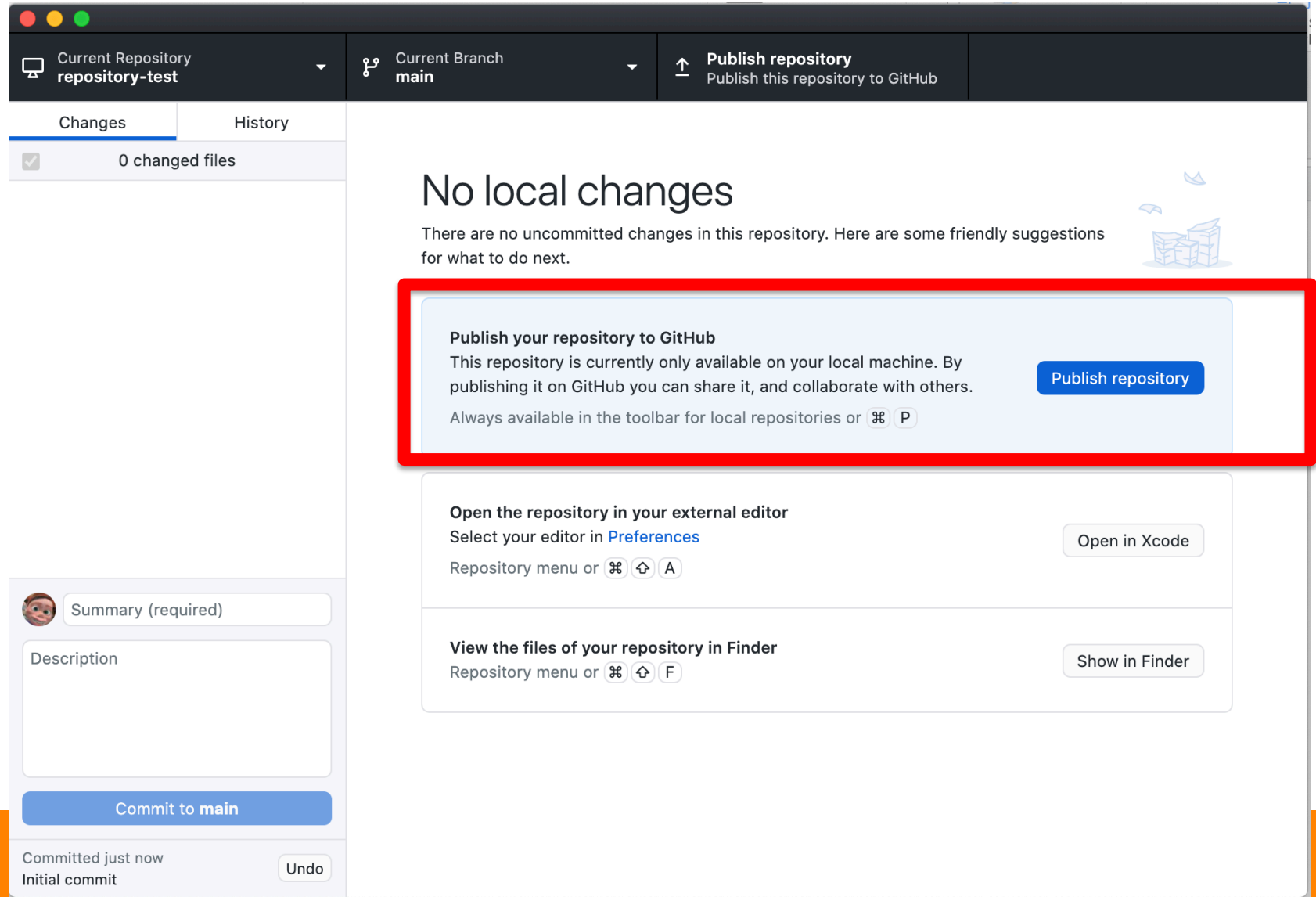
Create a Repository (III)



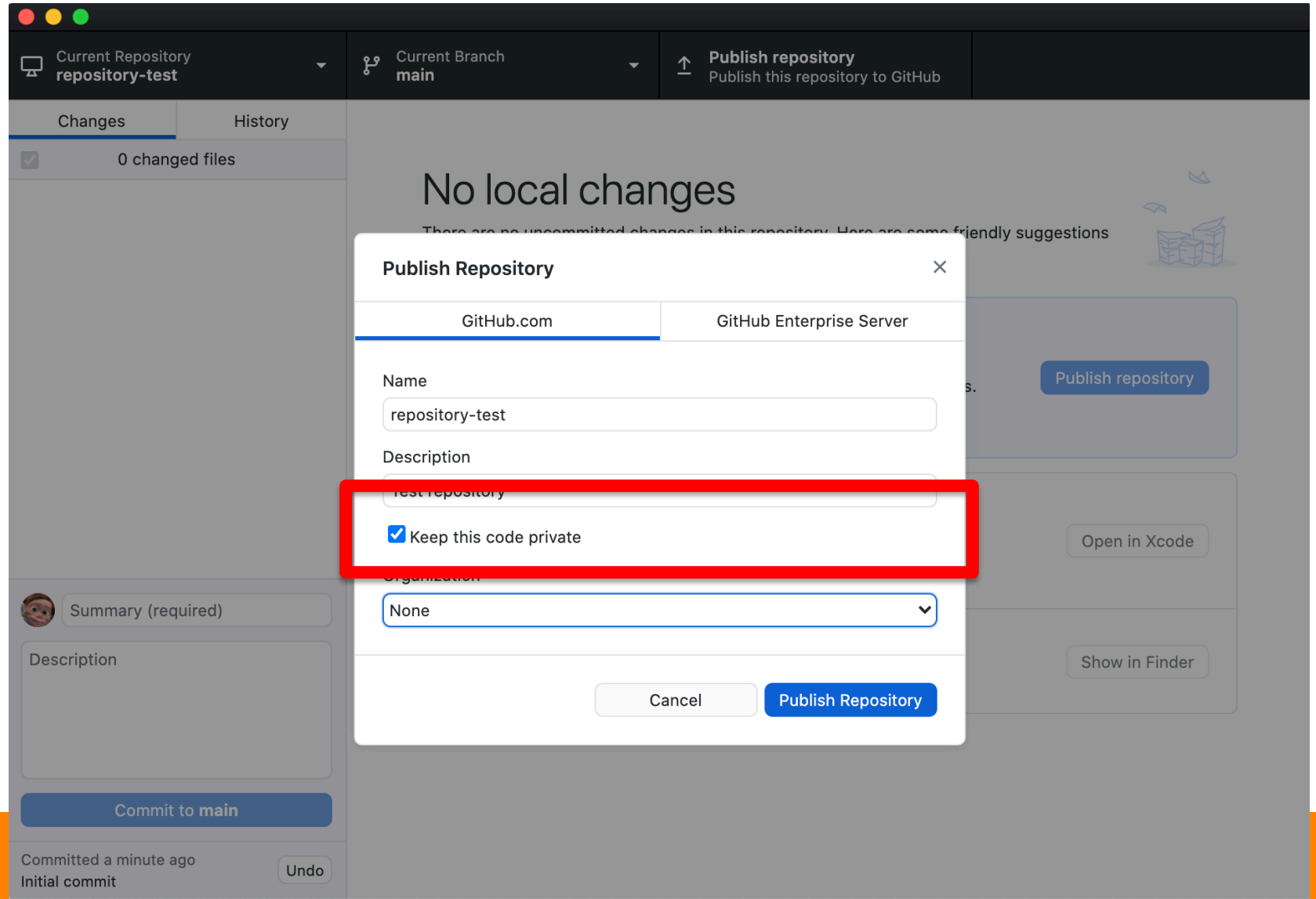
Create a Repository (IV)



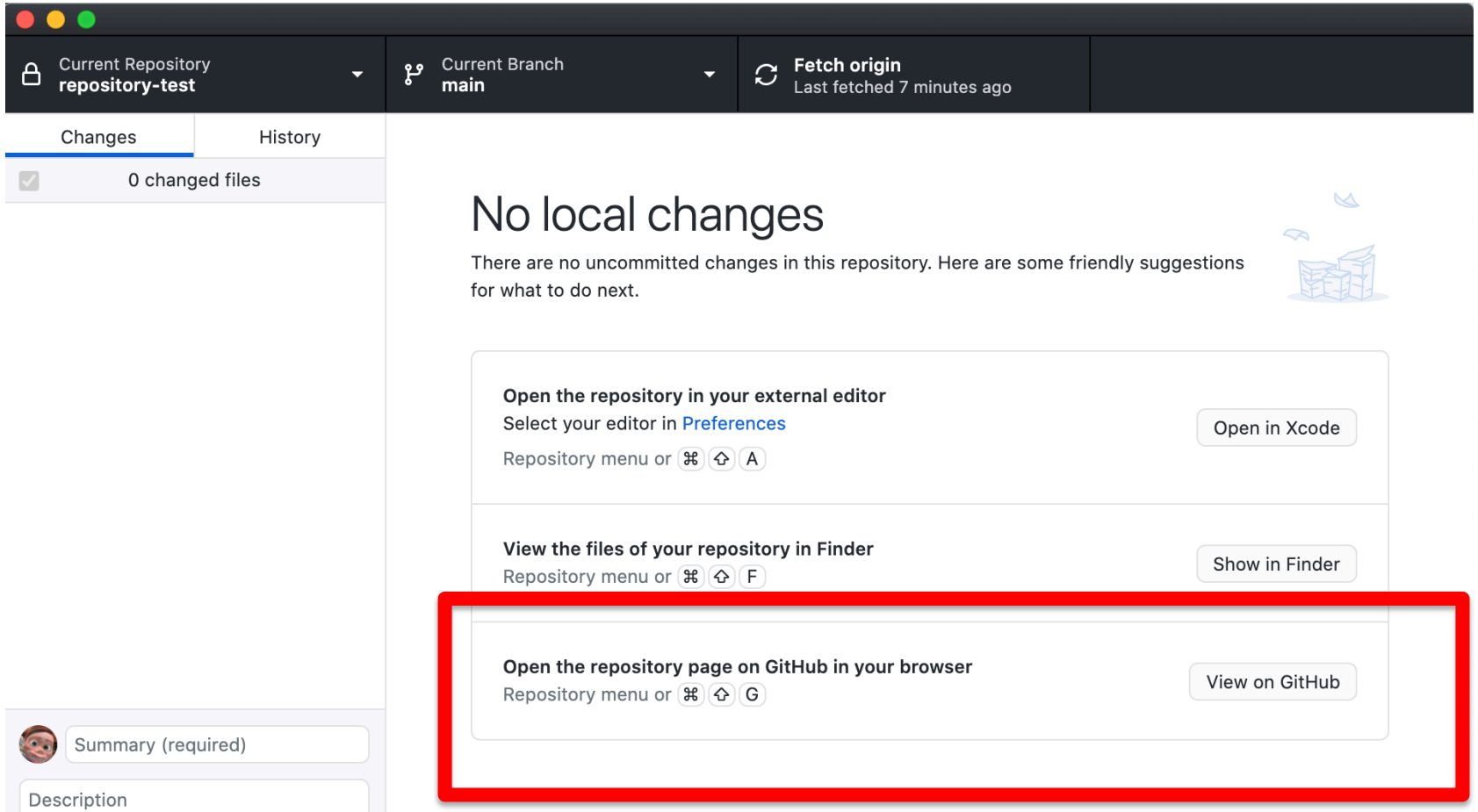
Publish your new Repository (I)



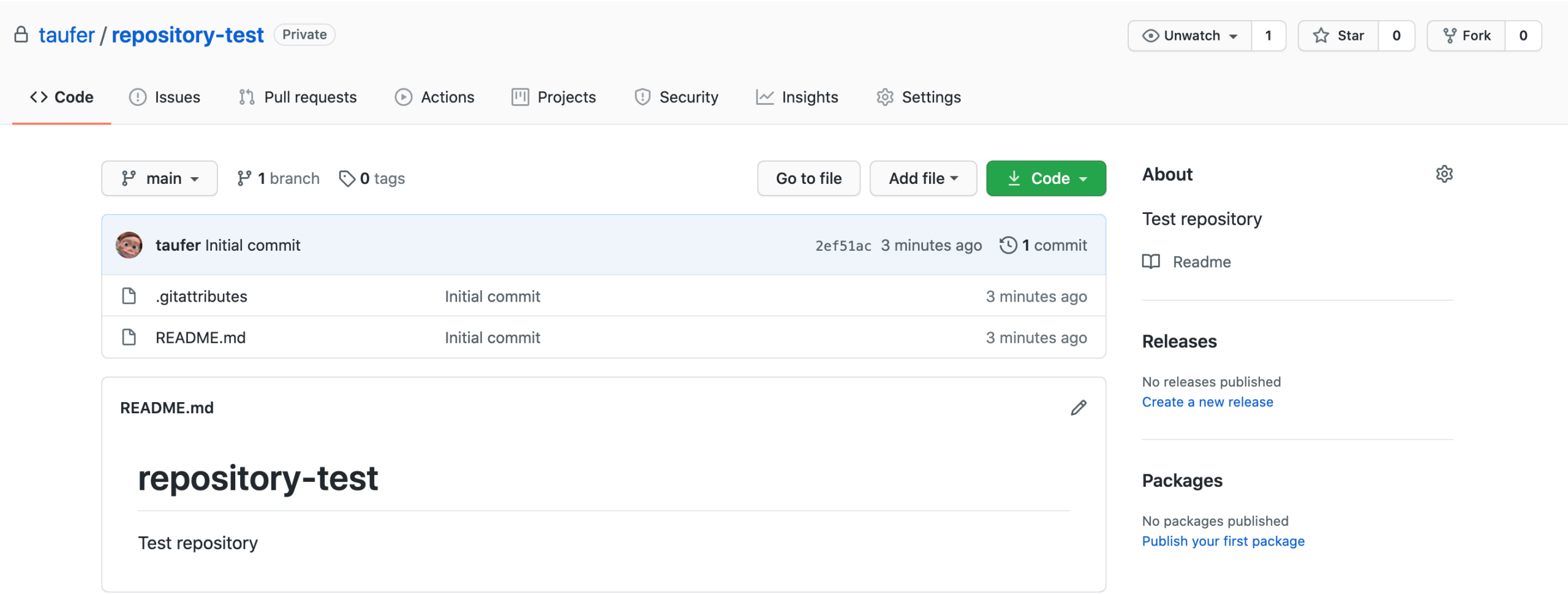
Publish your new Repository (II)



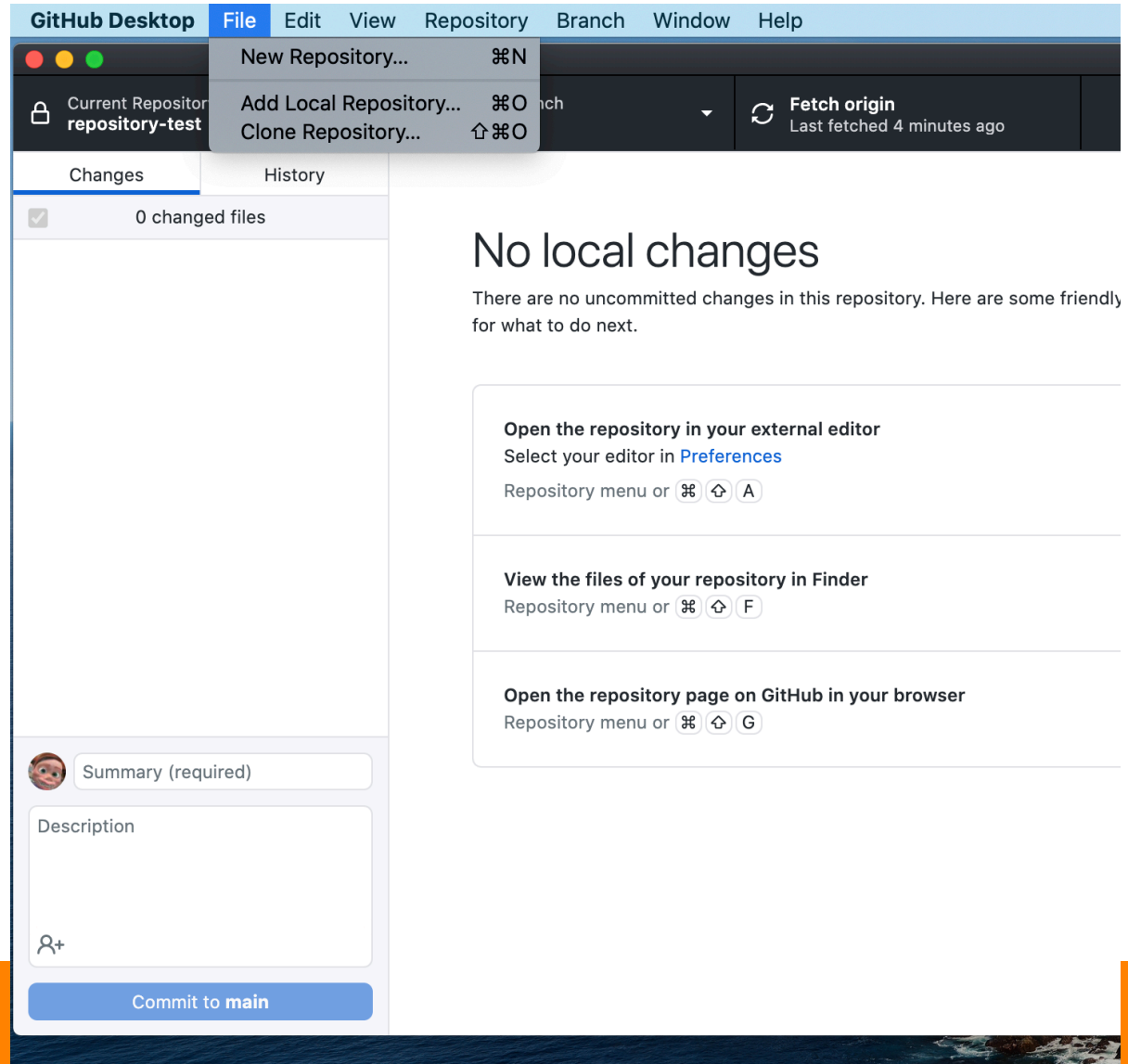
Publish your new Repository (III)



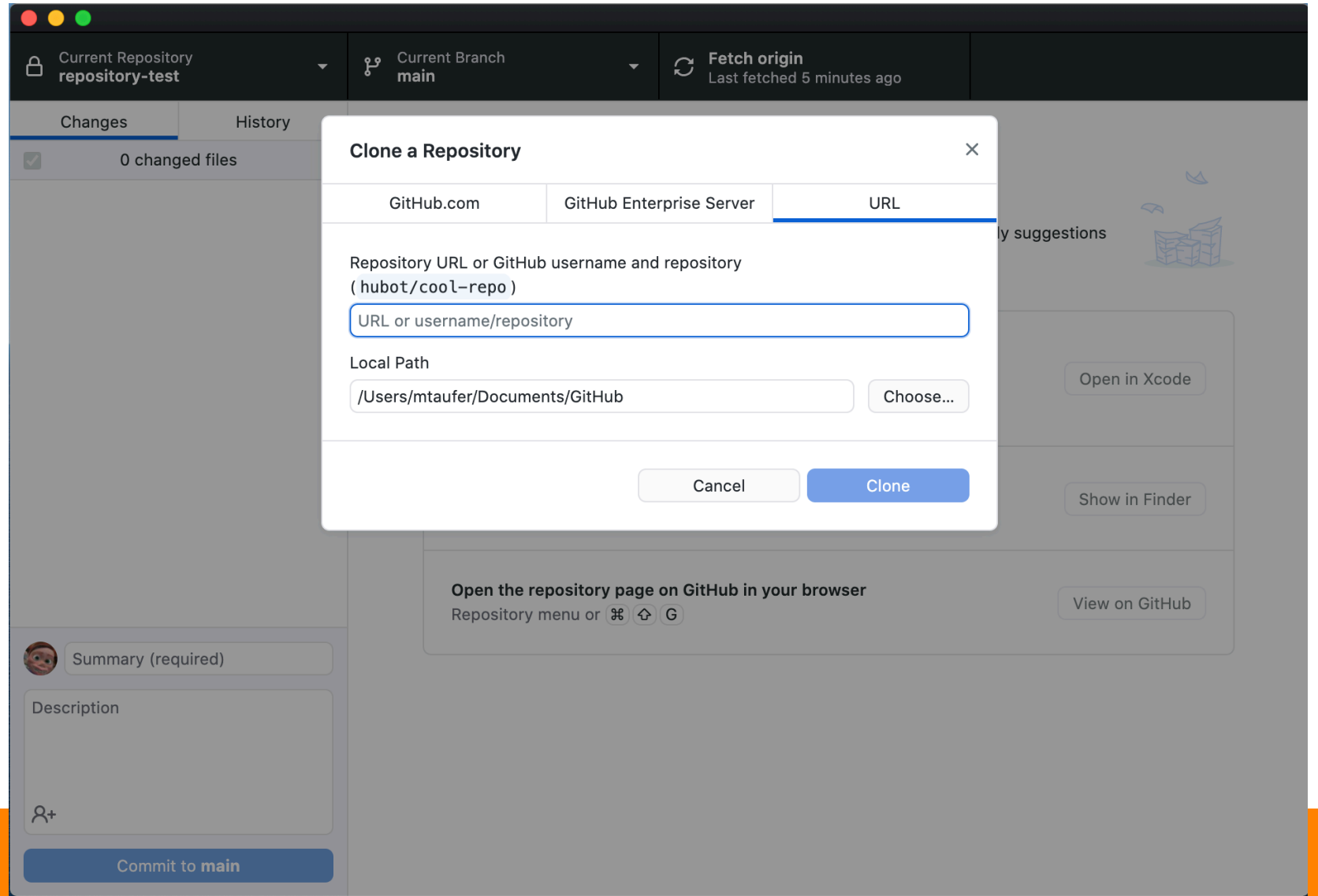
Publish your new Repository (IV)



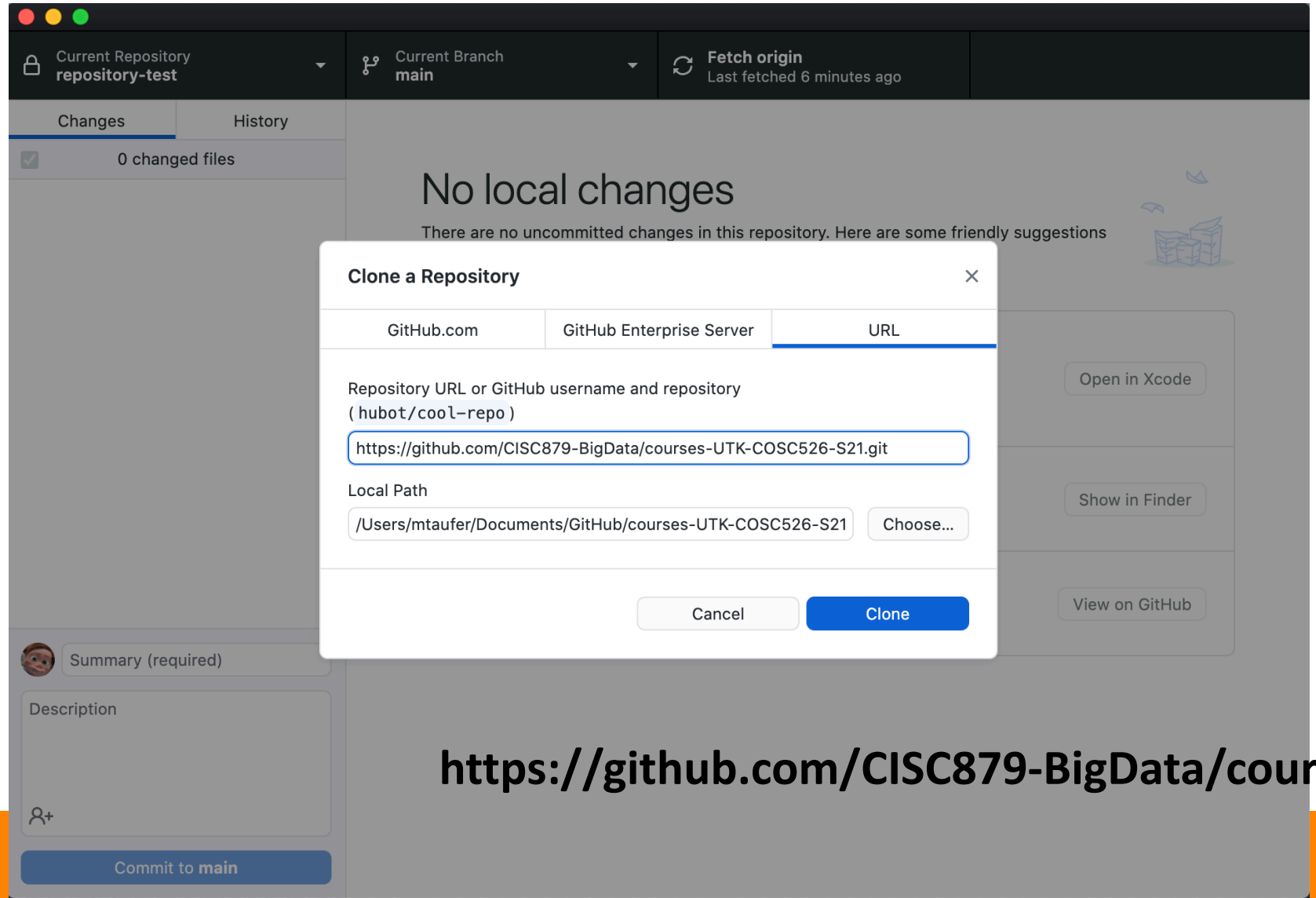
Clone the Course Repository (I)



Clone the Course Repository (II)

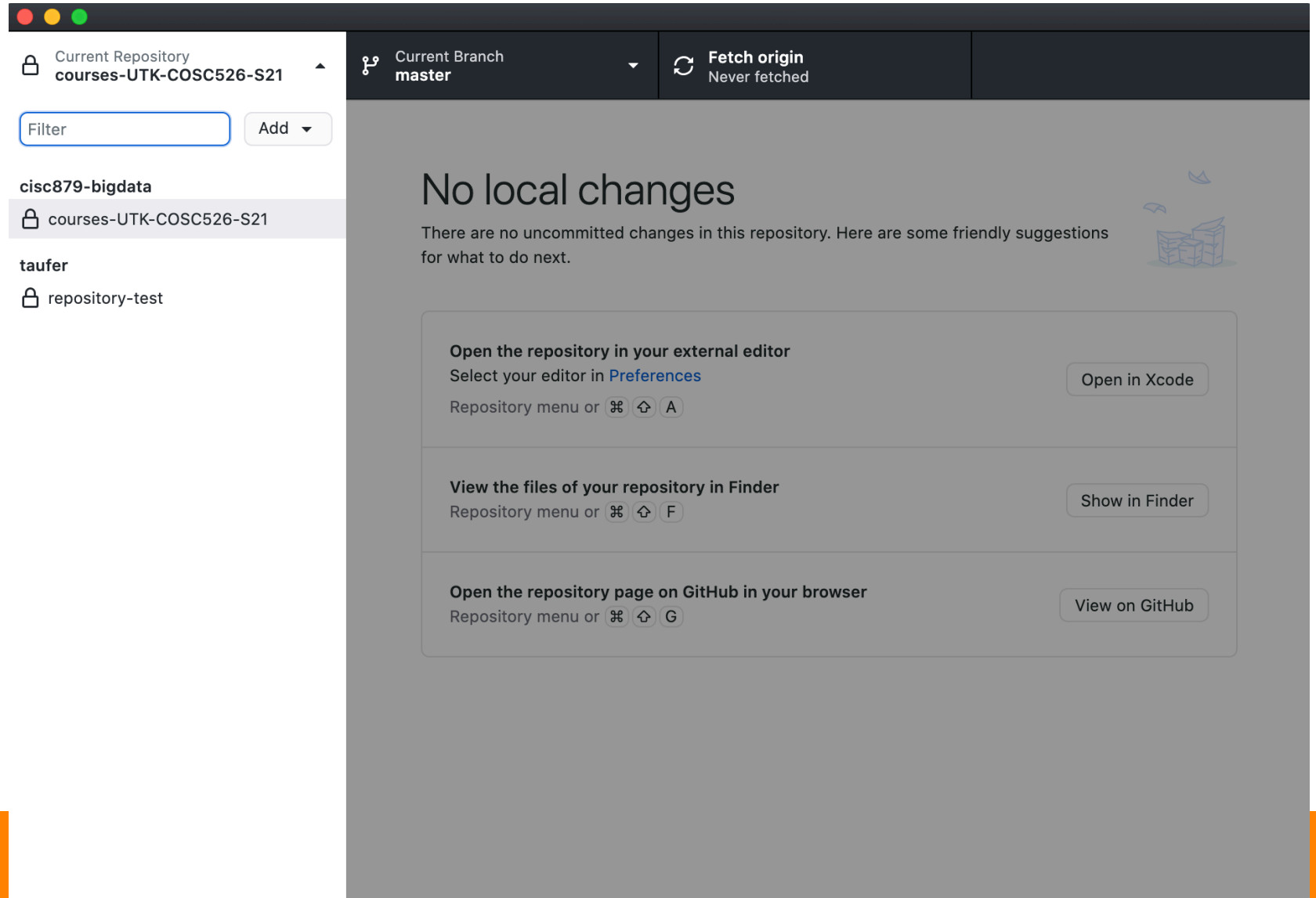


Clone the Course Repository (III)

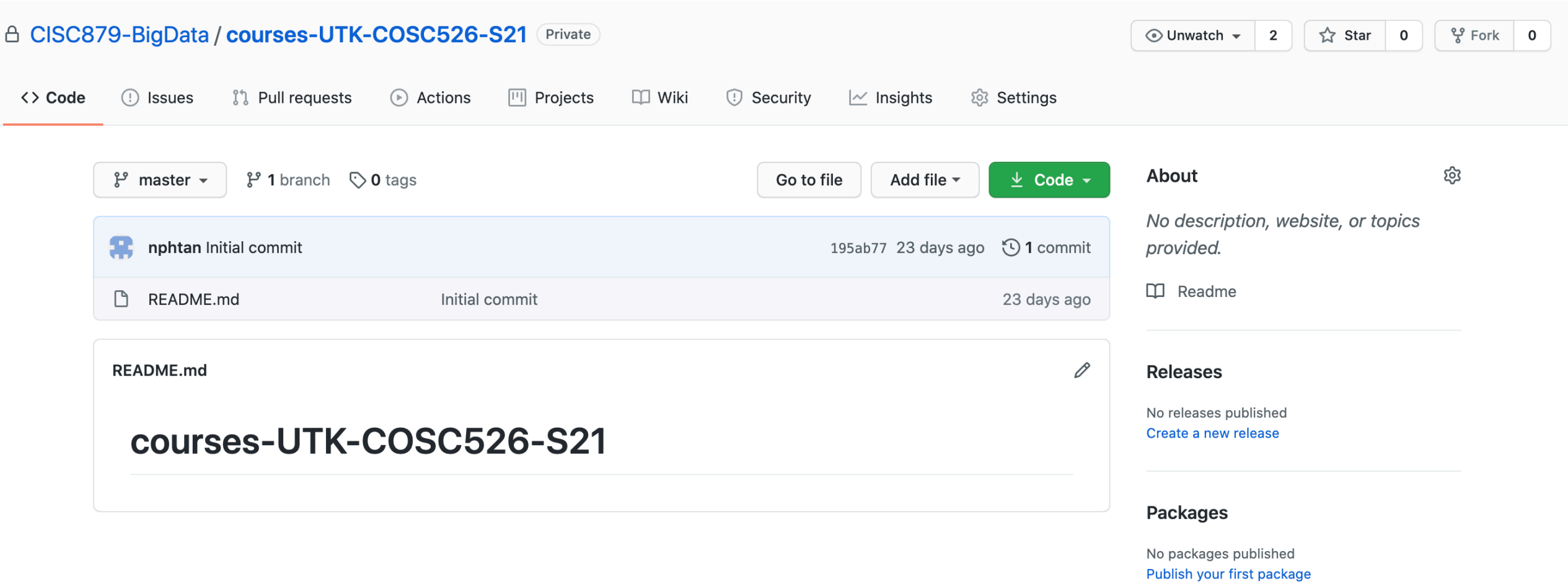


https://github.com/CISC879-BigData/courses-UTK-COSC526-S21.git

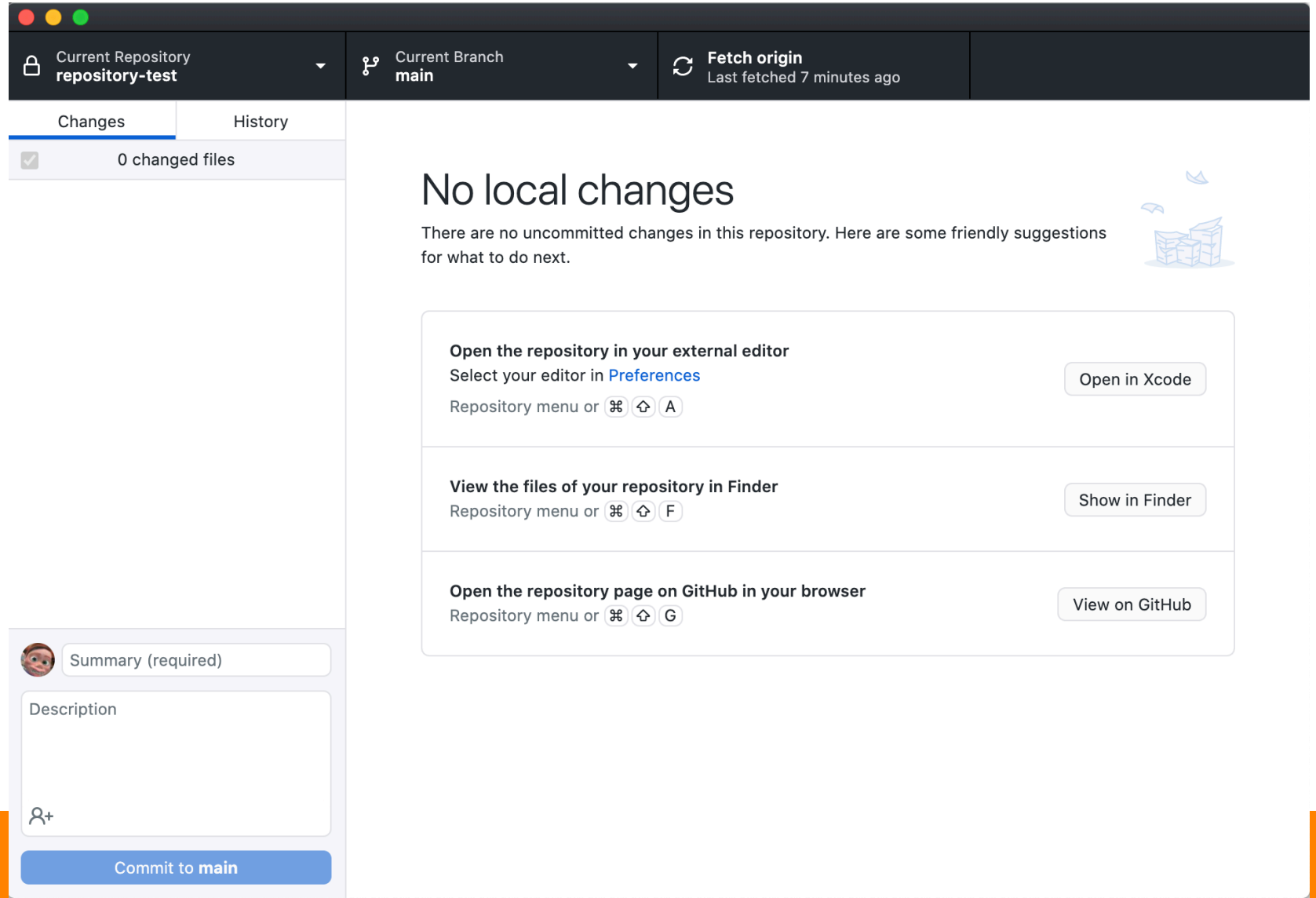
Clone the Course Repository (IV)



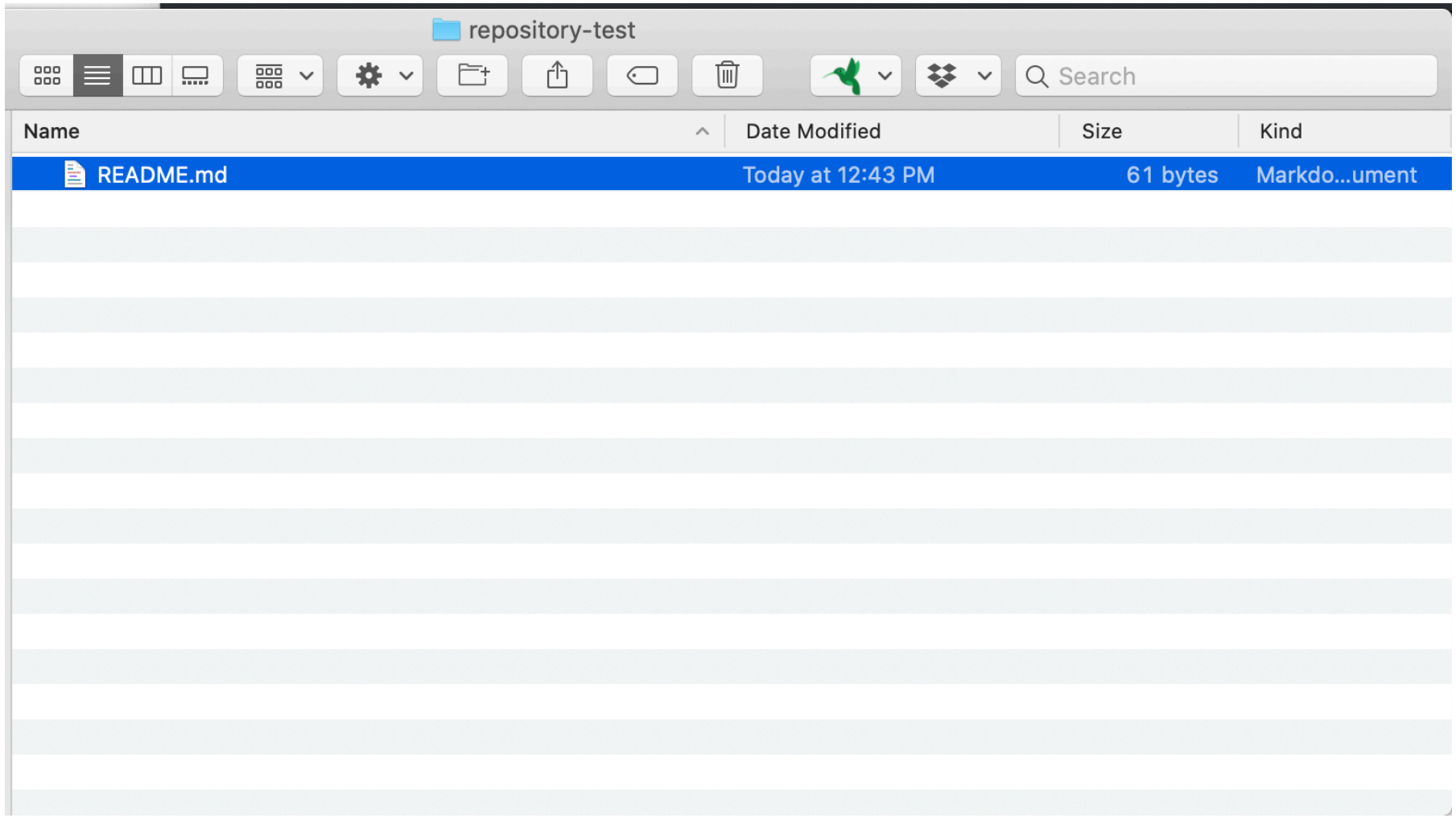
Clone the Course Repository (V)



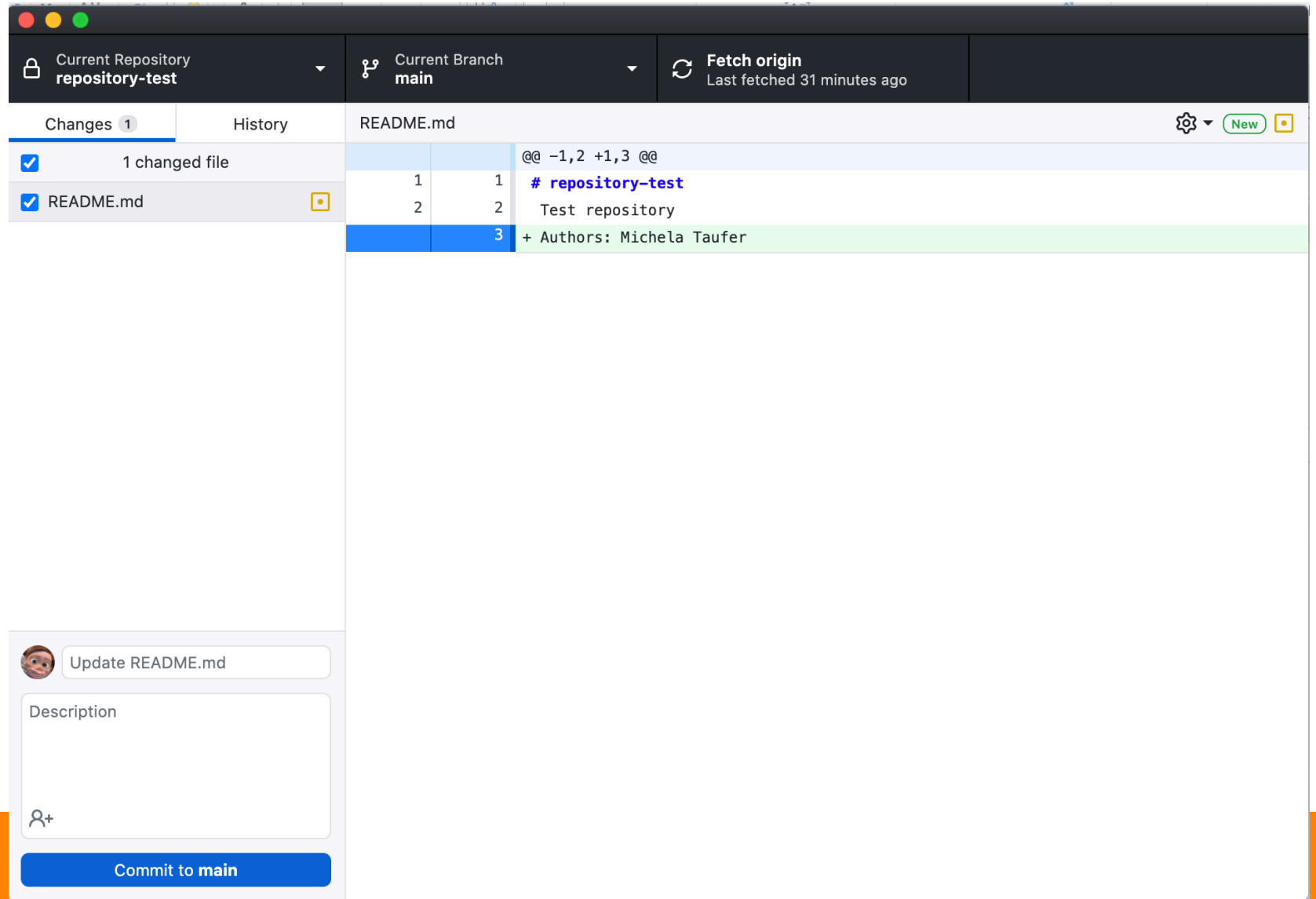
Committing Changes to a Repository (I)



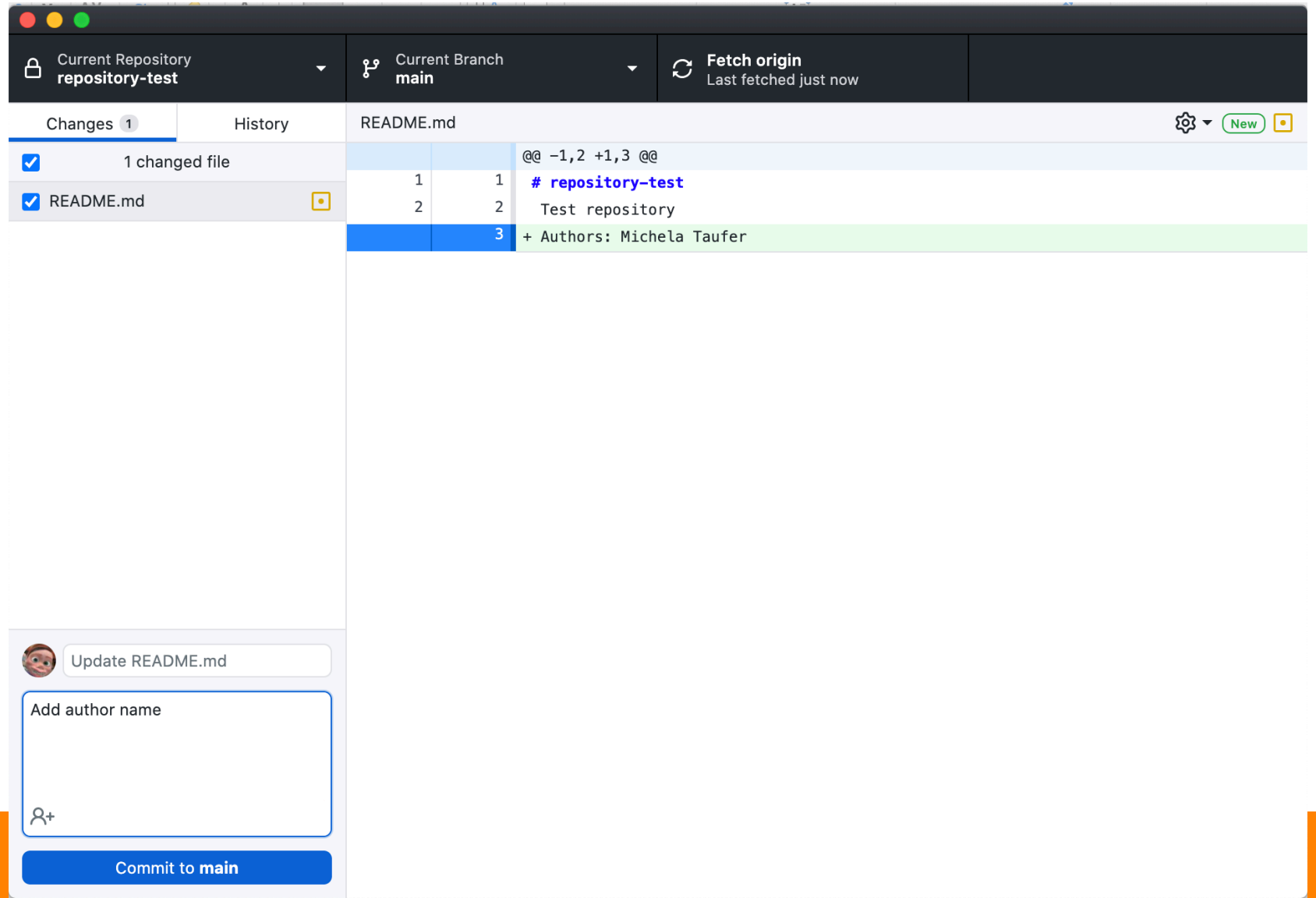
Committing Changes to a Repository (II)



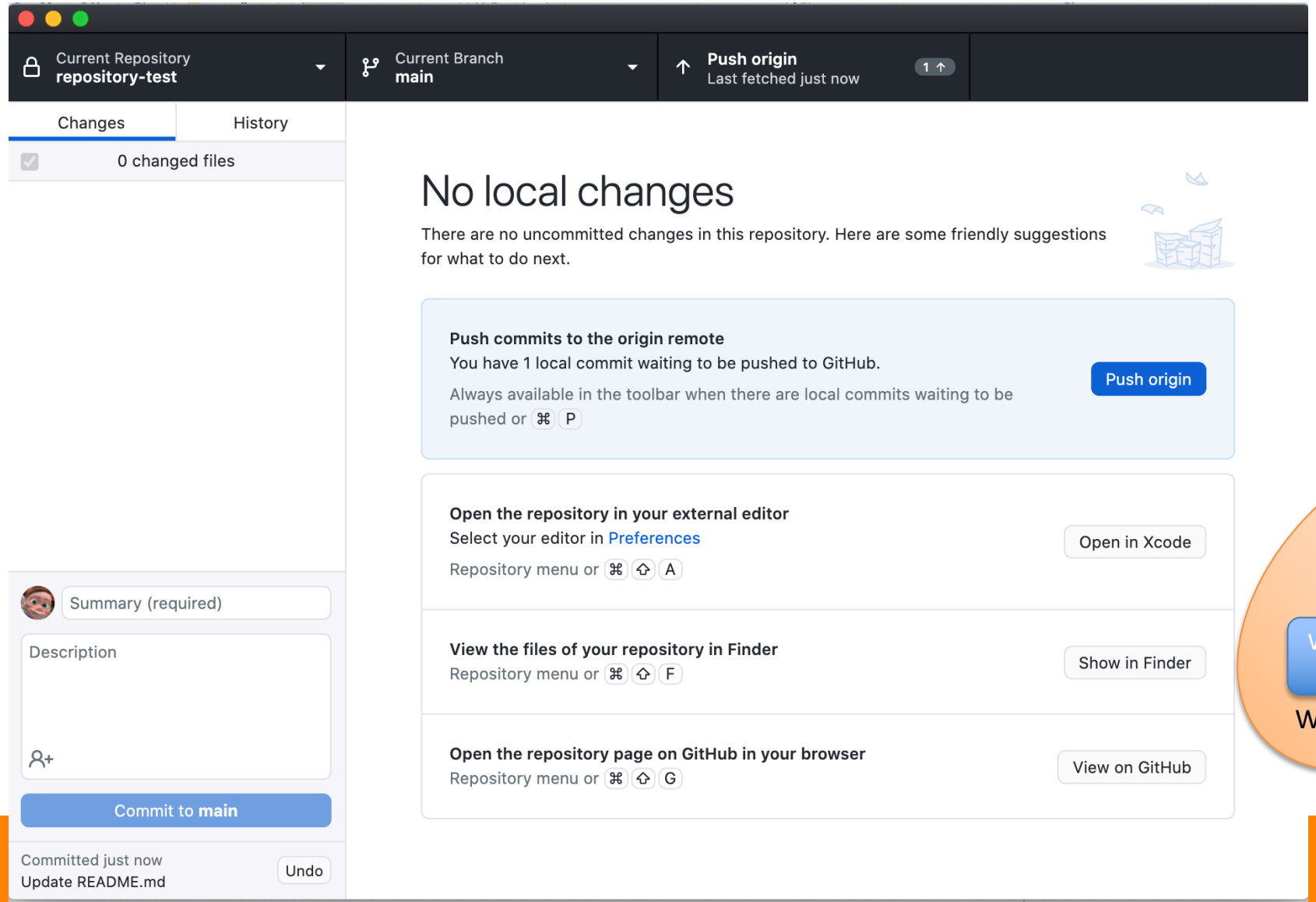
Committing Changes to a Repository (III)

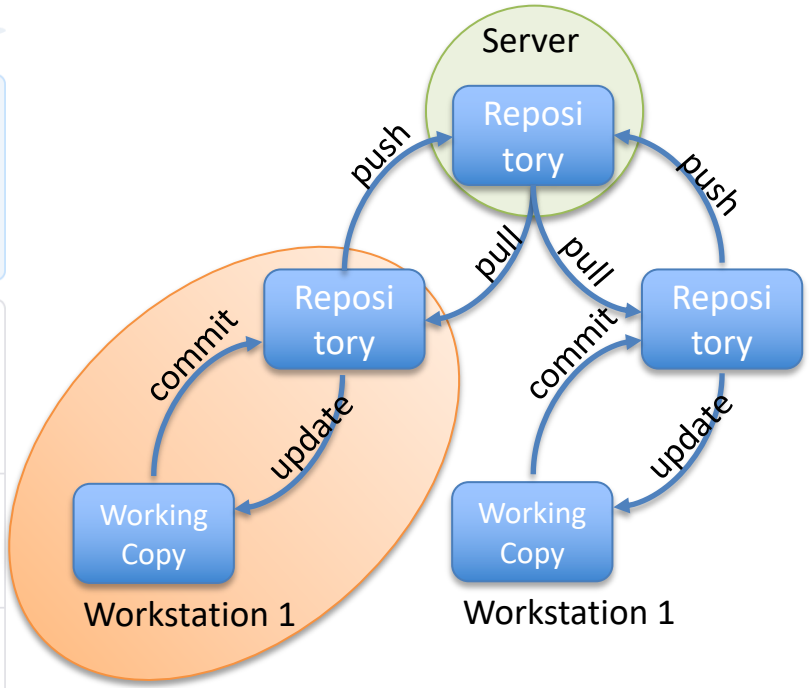


Committing Changes to a Repository (IV)

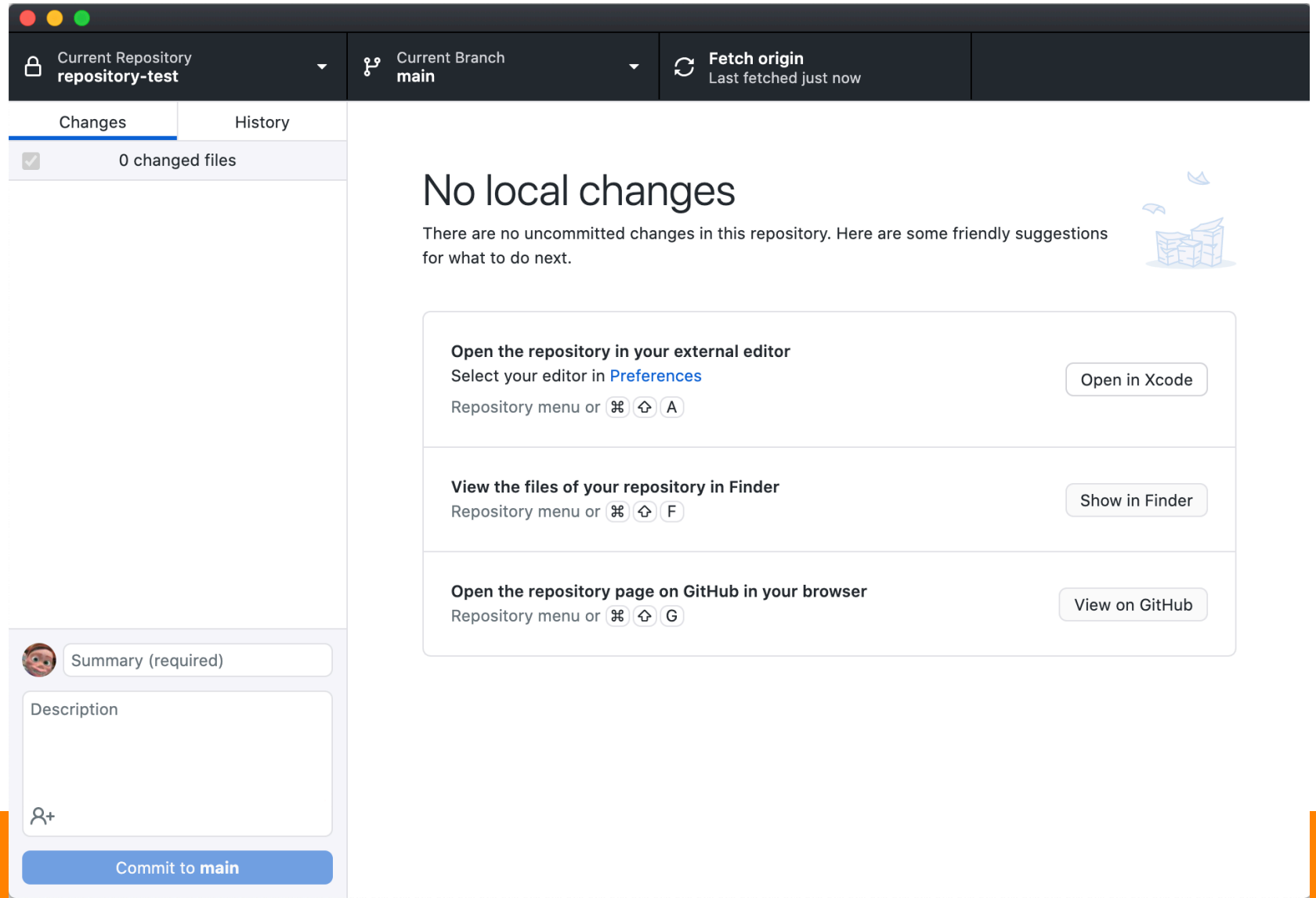


Committing Changes to a Repository (V)

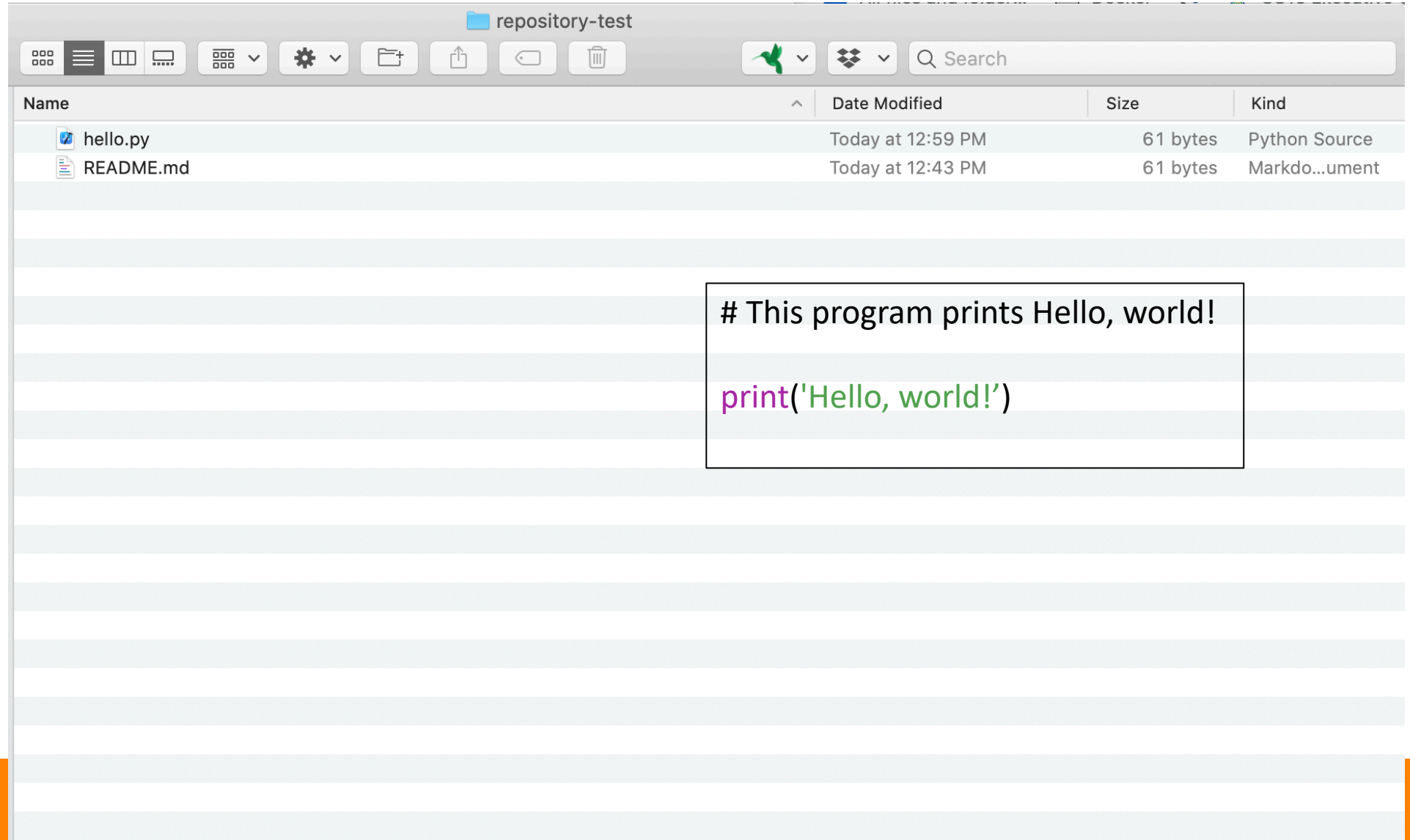




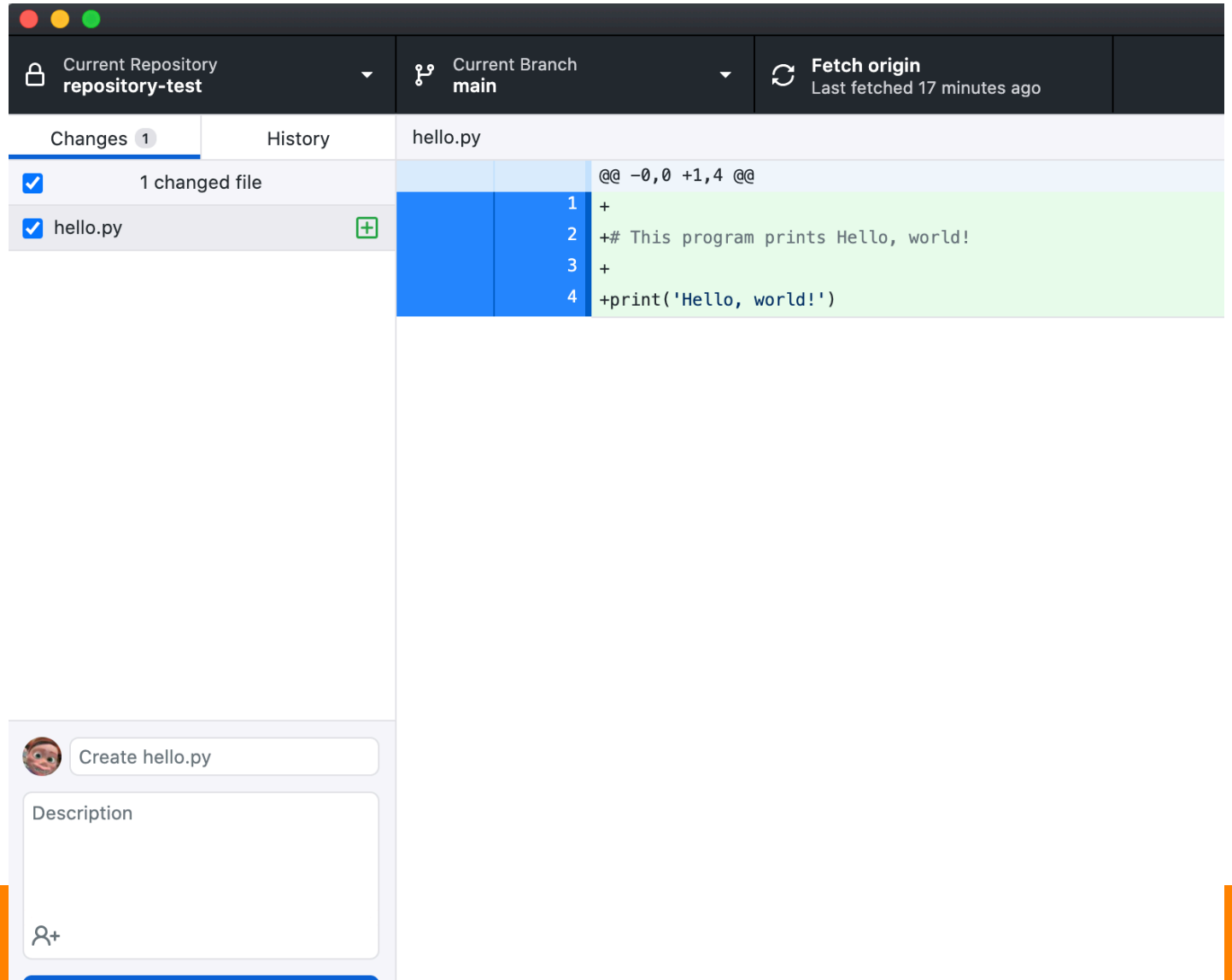
Committing Changes to a Repository (VI)



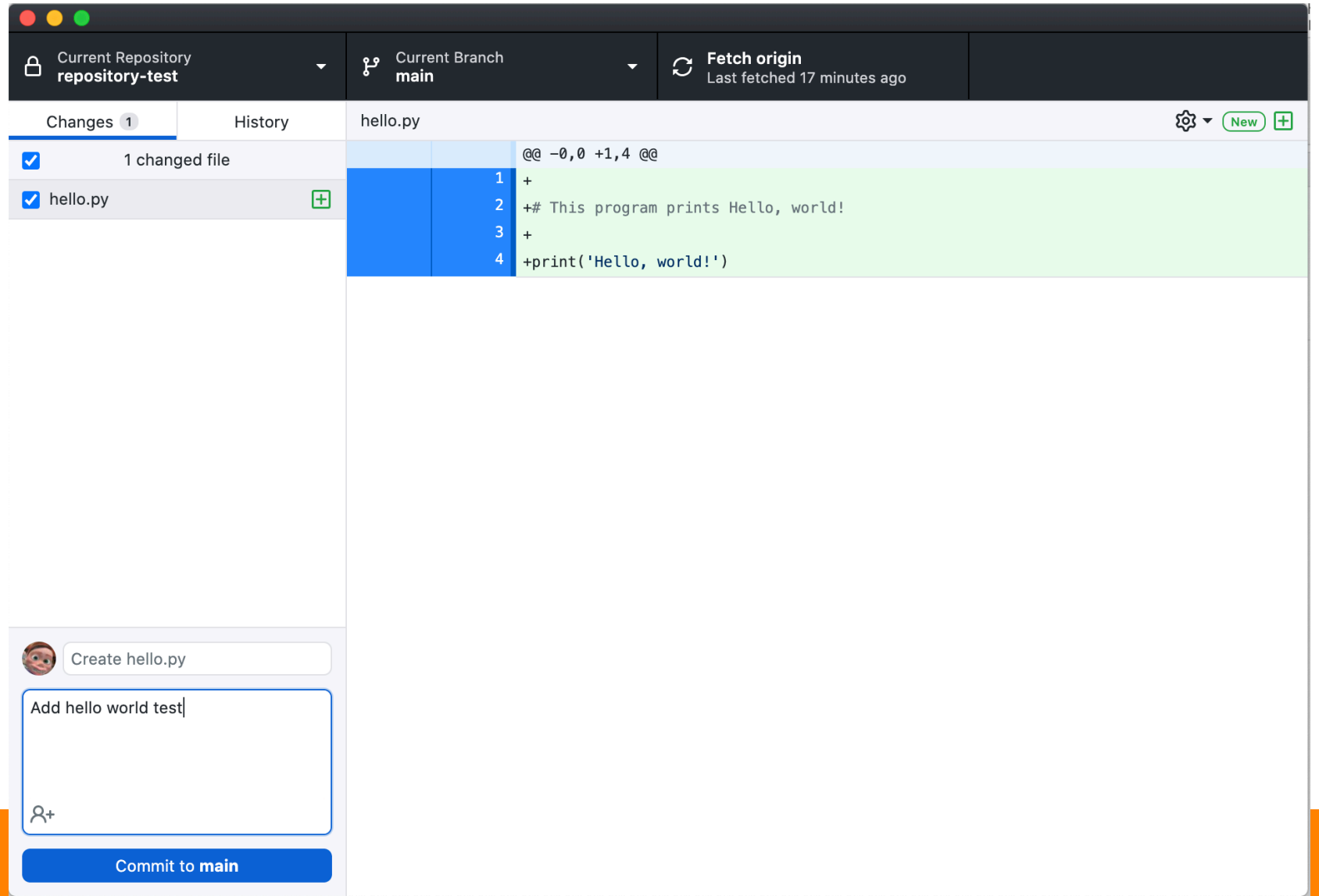
Add and Create new Files in a Repository (I)



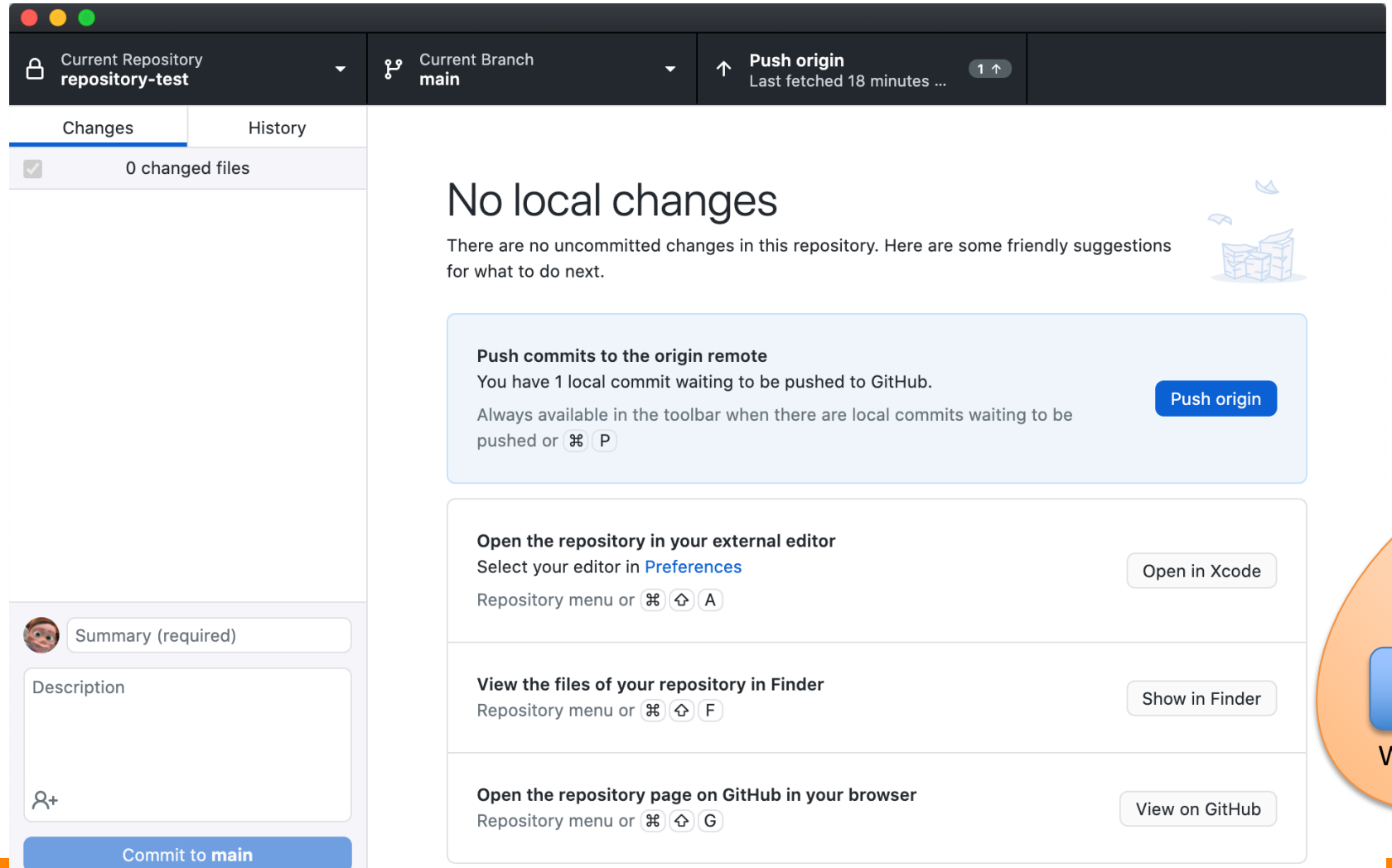
Add and Create new Files in a Repository (II)

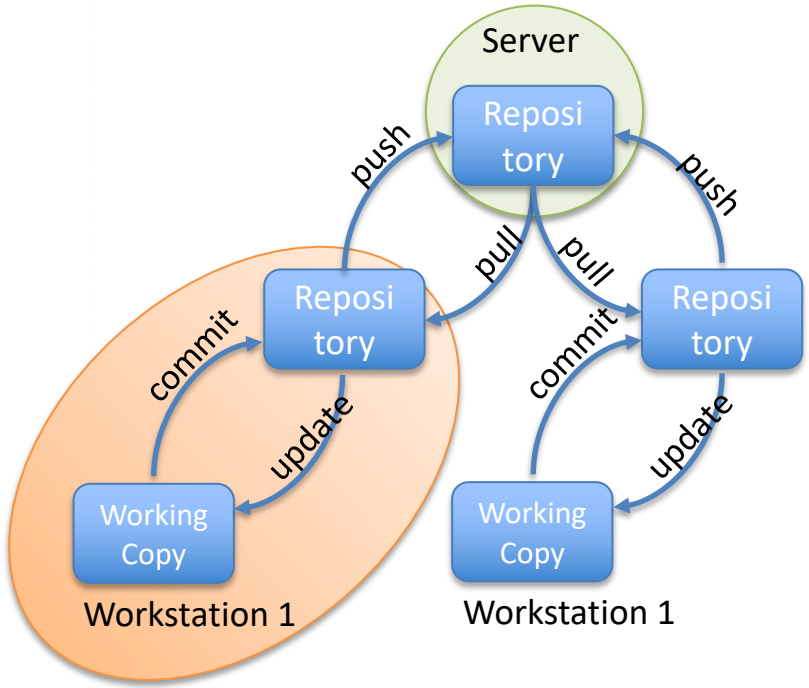


Add and Create new Files in a Repository (I)

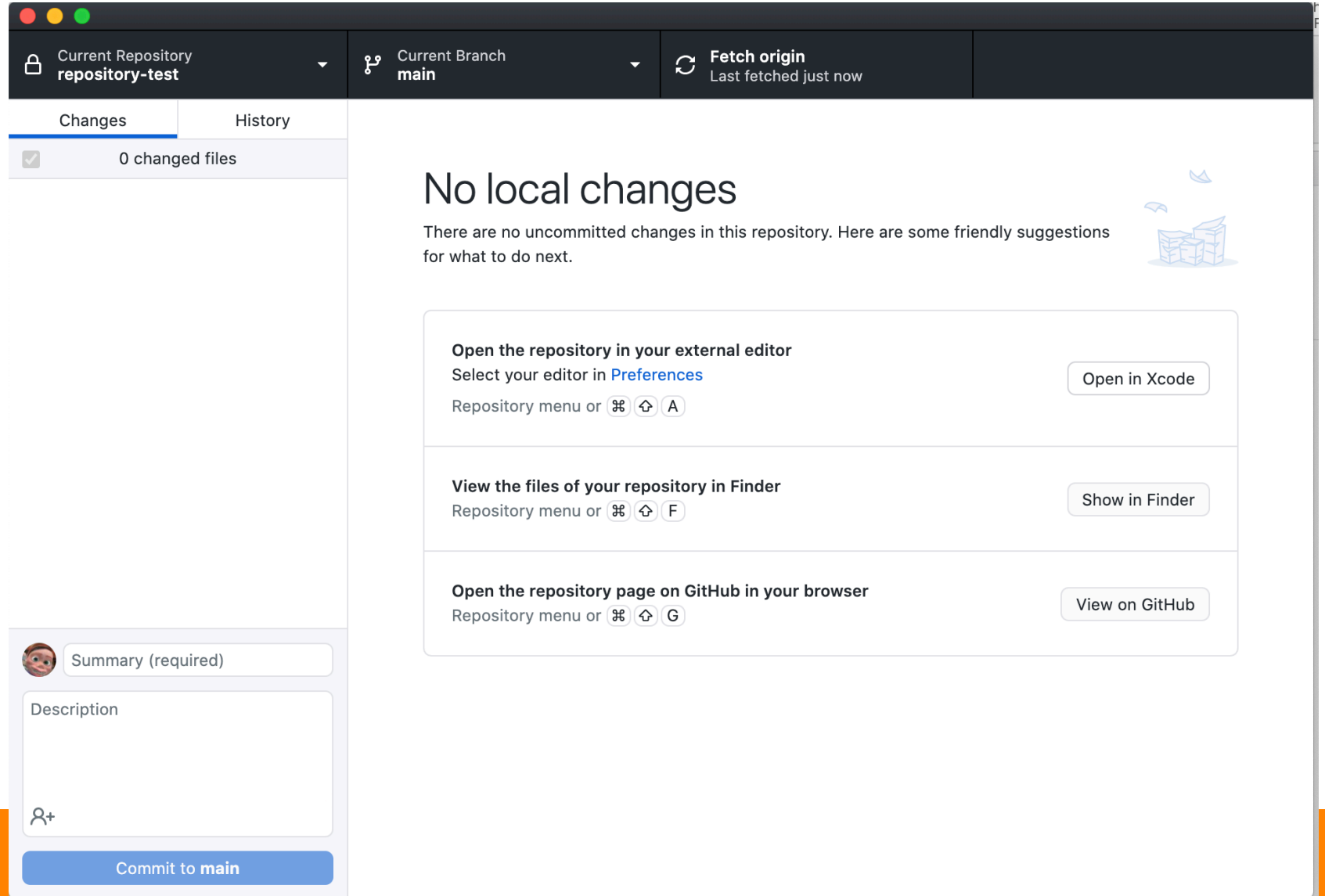


Add and Create new Files in a Repository (I)





Add and Create new Files in a Repository (I)



Pull from Repository (I)

- Not ready yet, bear with us ....

Command-line interface (CLI)

CLI: Your first time with git and GitHub (I)

CLI for git on Windows: https://gitforwindows.org

After installing git and GitHub desktop, if you are using CLI:

- Set up git with your user name and email

```
$ git config --global user.name "Your name here"
$ git config --global user.email "your_email@example.com"
```

- Set up ssh on your computer
  - Look to see if you have files ~/.ssh/id_rsa and ~/.ssh/id_rsa.pub.
  - If not, create such public/private keys:

```
$ ssh-keygen -t rsa -C "your_email@example.com"
```

Source : https://kbroman.org/github_tutorial/pages/first_time.html

CLI: Your first time with git and GitHub (II)

- Copy your public key (the contents of the newly created id_rsa.pub file) into your clipboard – e.g., on Mac

```
$ pbcopy < ~/.ssh/id_rsa.pub
```

- Paste your ssh public key into your github account settings
  - Go to your github Account Settings
  - Click "SSH Keys" on the left.
  - Click "Add SSH Key" on the right.
  - Add a label (like "My laptop") and paste the public key into the big text box.

From: https://kbroman.org/github_tutorial/pages/first_time.html

CLI: Your first time with git and GitHub (III)

- In a terminal/shell, type the following to test it:

```
$ ssh -T git@github.com
```

- If it says something like the following, it worked:

```
Hi username! You've successfully authenticated, but Github does
not provide shell access.
```

From: https://kbroman.org/github_tutorial/pages/first_time.html

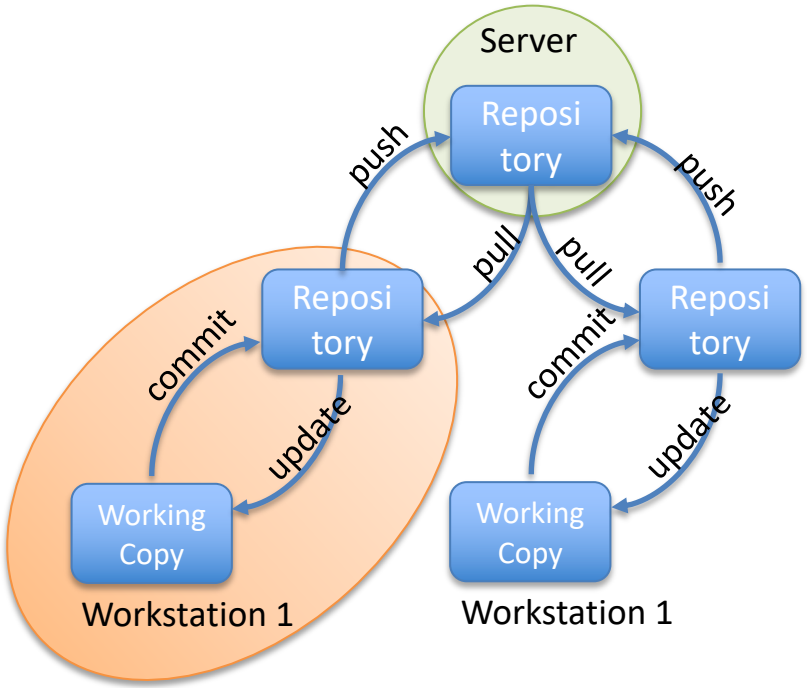
Use git and GitHub

- The routine use of git involves just a few commands:
  - init
  - add and commit
  - push and pull
  - status
  - diff

A new repo from scratch

- Create a directory to contain the project
  $ mkdir YourRepos
- Go into the new directory
- Type *git init*
  $ git init
- Create a new file – ReadMe.md is a good start
  $ echo "# YourRepos " >> ReadMe.md
- Type *git add* to add the file
  $ git add README.md
- Type *git commit* to commit the file
  $ git commit -m "first commit”

At this point, your repos is ONLY local



Keep the repo clean

- Create a **.gitignore** file to indicate all of the files you don't want to track in a (sub)directory

  $ git add .gitignore

- Create a **.gitignore_global** file to indicate all of the files you don't want to track for the entire directory. You have to tell git about the global .gitignore

  $ git add .gitignore_global

  $ git config --global core.excludesfile ~/.gitignore_global

.gitignore_global and .gitignore

*~

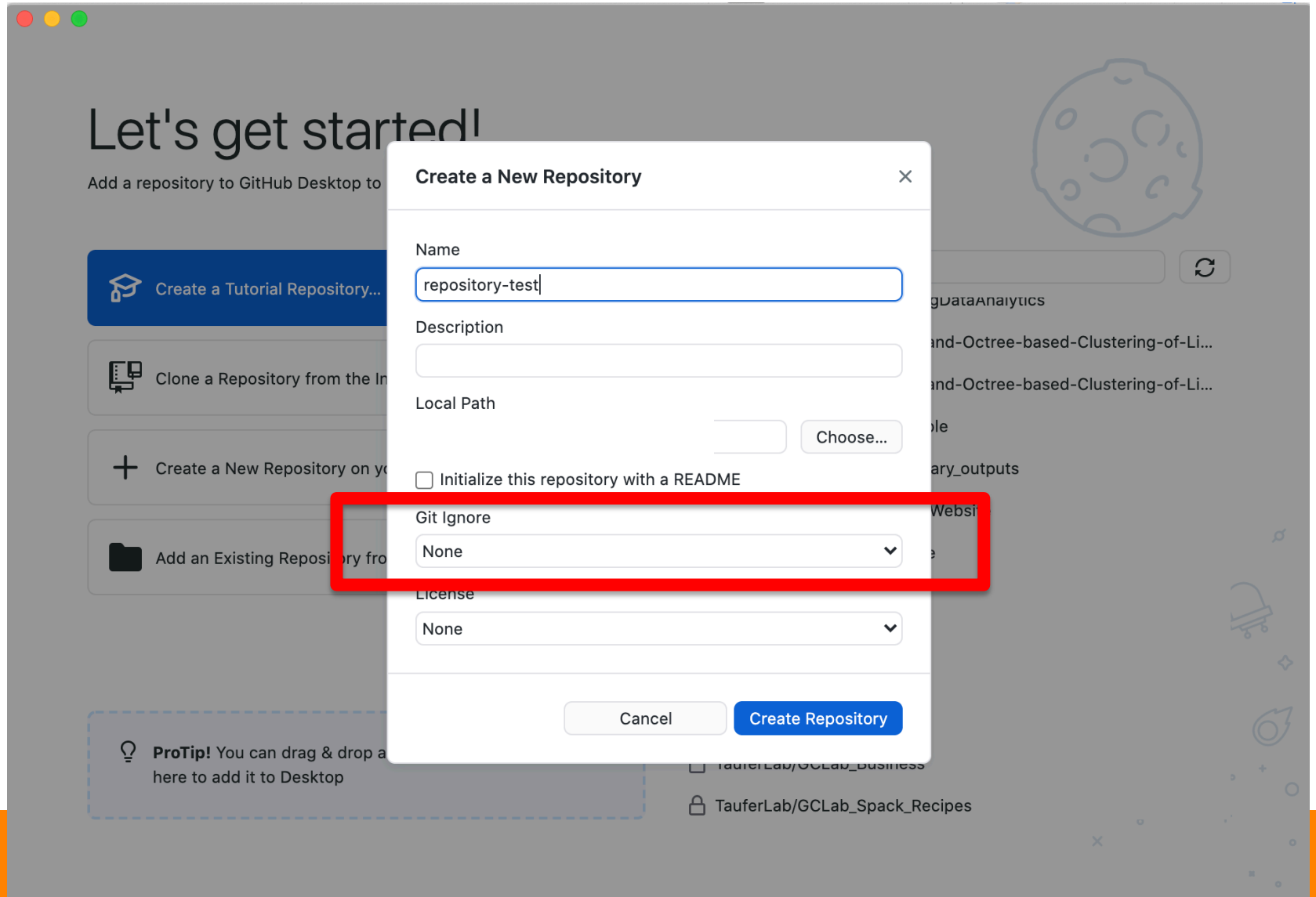
.*~

.DS_Store

.Rhistory

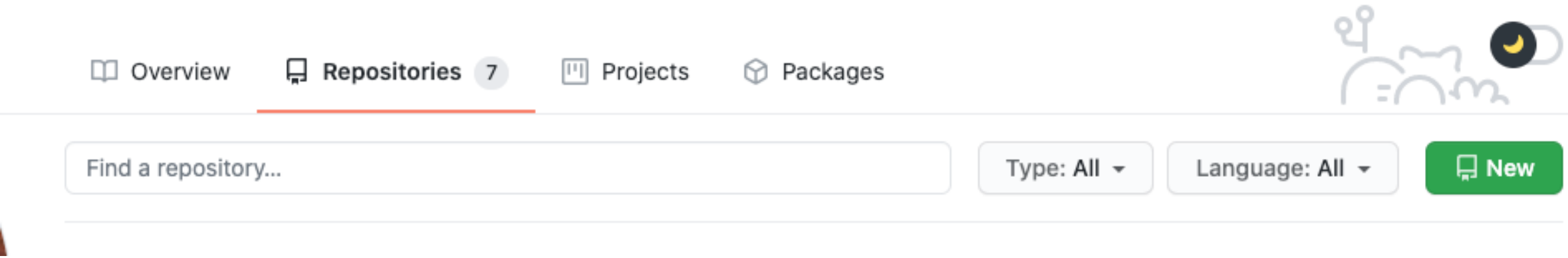
.RData

Create a Repository (I) repeat ….



Connect new repo to GitHub (I)

- Go to github
- Log in to your account
- Click the ”New” repository button in the top-right
- Click the “Create repository” button



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * **Repository name** *

taufer /

Great repository names are short and memorable. Need inspiration? How about **scaling-invention**?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

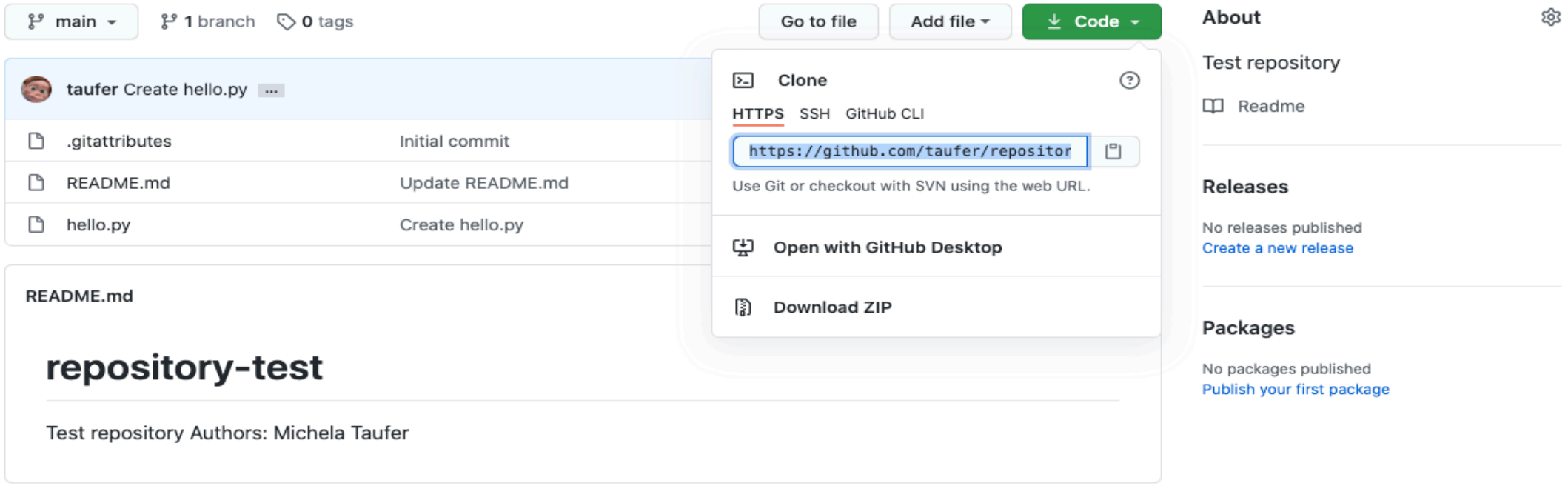
Choose a license
A license tells others what they can and can't do with your code. Learn more.

Create repository

Connect new repo to GitHub (II)

- Clone a new or old repository

  git clone https://github.com/YourAccount/YourRepos.git



Add and commit (I)

- Add completely new files as well as to “add” modifications to files that already exist in the repository (e.g., README.md)

```
$ mkdir Lecture01
$ touch Lecture01/README.md
$ emacs README.md
$ git add Lecture01 Lecture01/README.md README.md
```

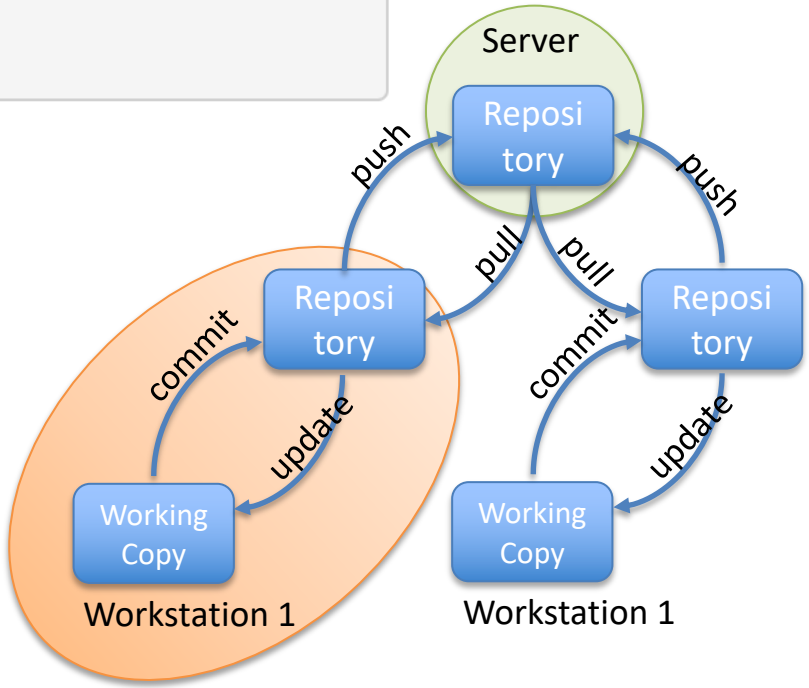
From: https://kbroman.org/github_tutorial/pages/routine.html

Add and commit (II)

- Use git commit to add the modifications to the repo

  $ git commit OR $ git commit -m “Message: 40 / 60 characters”

```
$ git commit -m "Fix such and such"
```



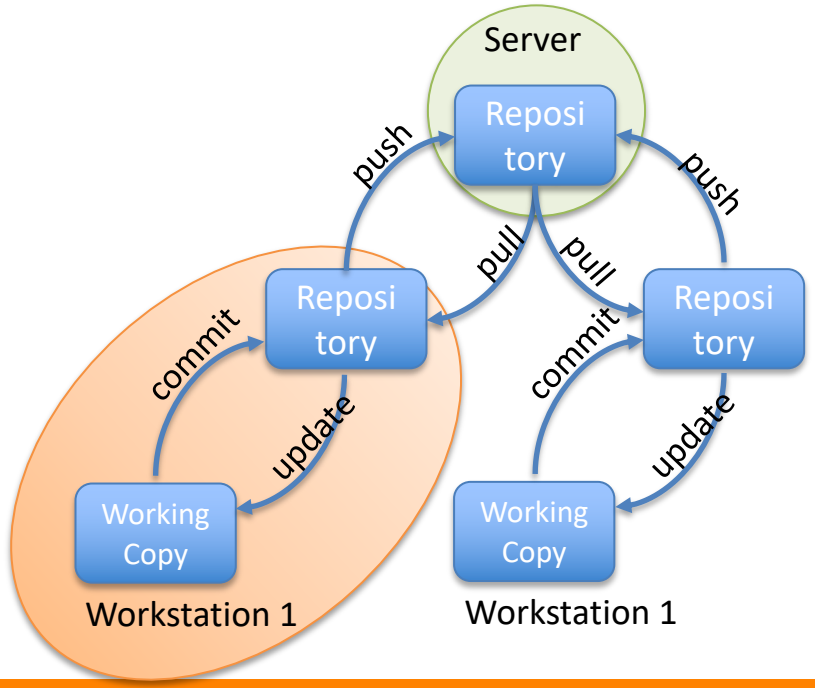
From: https://kbroman.org/github_tutorial/pages/routine.html

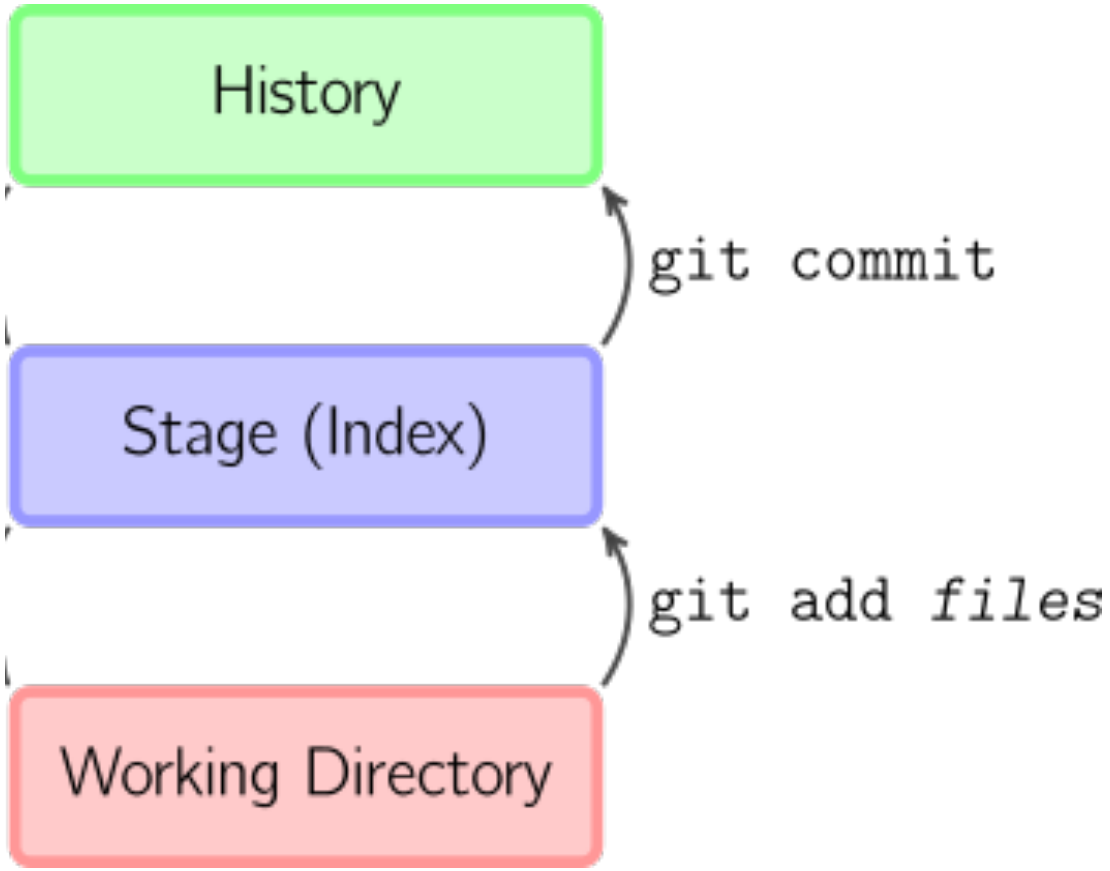
Add and commit (II)

$ git add <filename>

$ git add

$ git commit -m "Commit message”



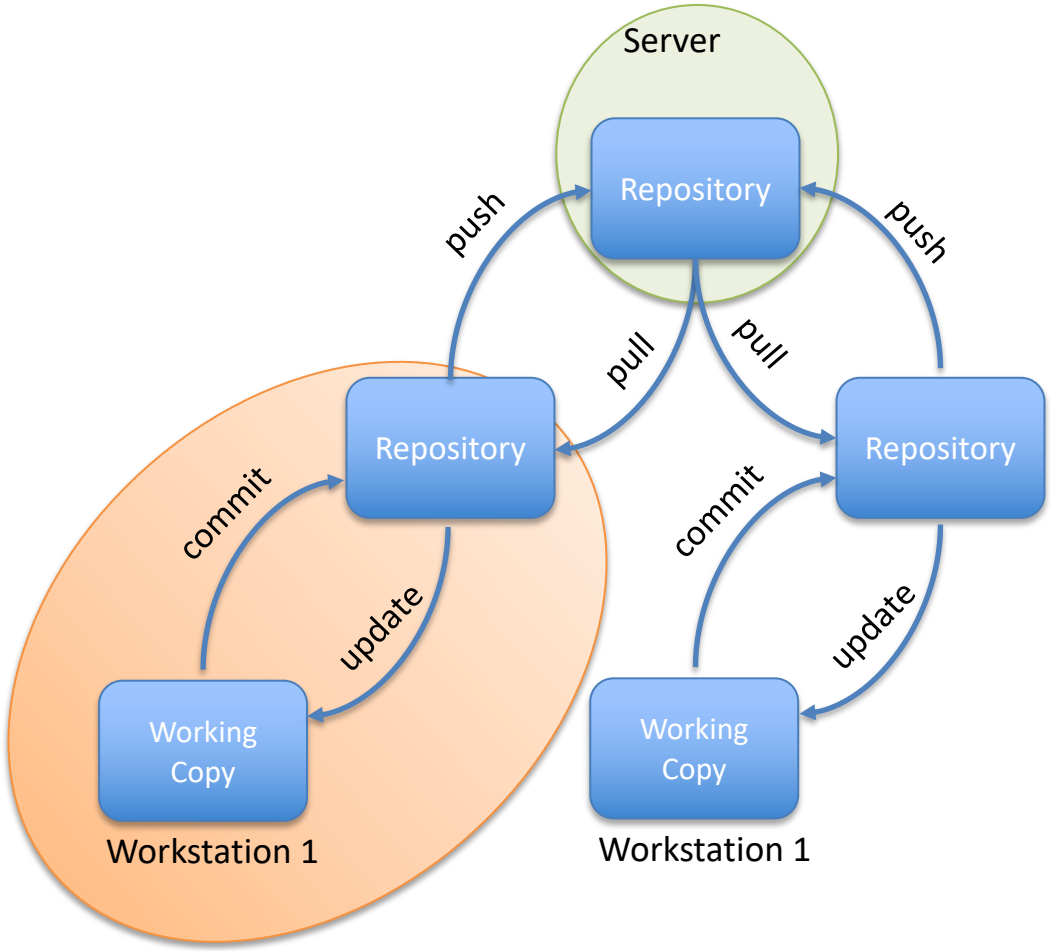


Push to and pull from github

- After any *git add* and *git commit* commands, changes are in the **HEAD** of your local working copy
- To push committed changes to your remote repository - github, type:

  $ git push
- To pull committed changes from your remote repository - github, type:

  $ git pull



From: https://kbroman.org/github_tutorial/pages/routine.html

Differences github

- To extract differences and changes, type:

  $ git diff

- To extract changes in one specific file, type

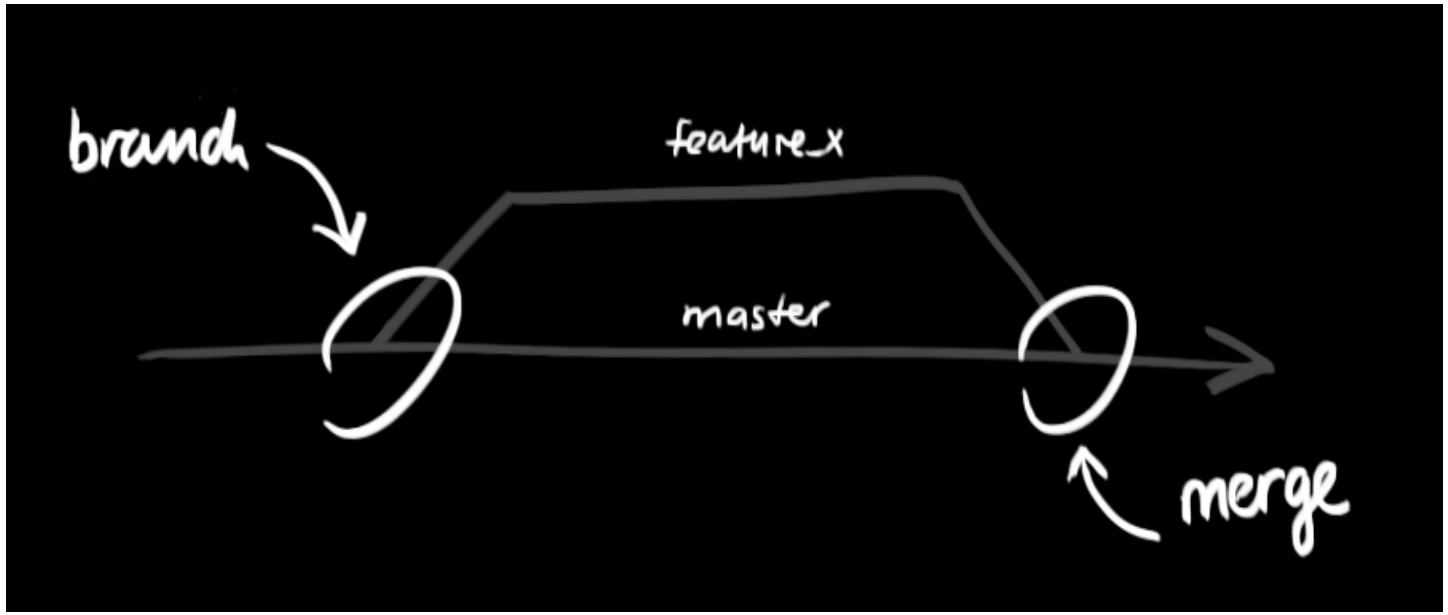
  $ git diff README.md

From: https://kbroman.org/github_tutorial/pages/routine.html

More about GitHub (Optional)

Branching and branches

- Branches are used to develop features isolated from each other
- The *master* branch is the "default" branch when you create a repository
- Use other branches for development and merge them back to the master branch upon completion



Branching and branches

- Create a new branch named "feature_x" and switch to it:

  $ git checkout -b feature_x
- Switching back to master:

  $ git checkout master
- Delete the feature branch:

  $ git branch -d feature_x
- Pushing a branch for others to use:

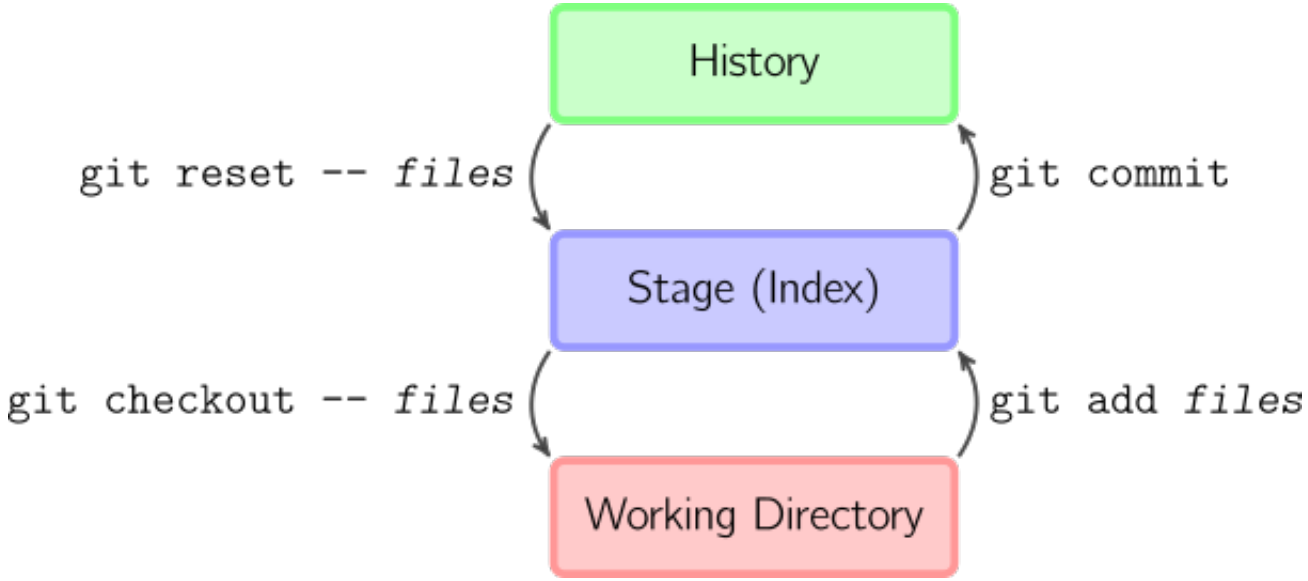
  $ git push origin <branch>

Syncing & Merging

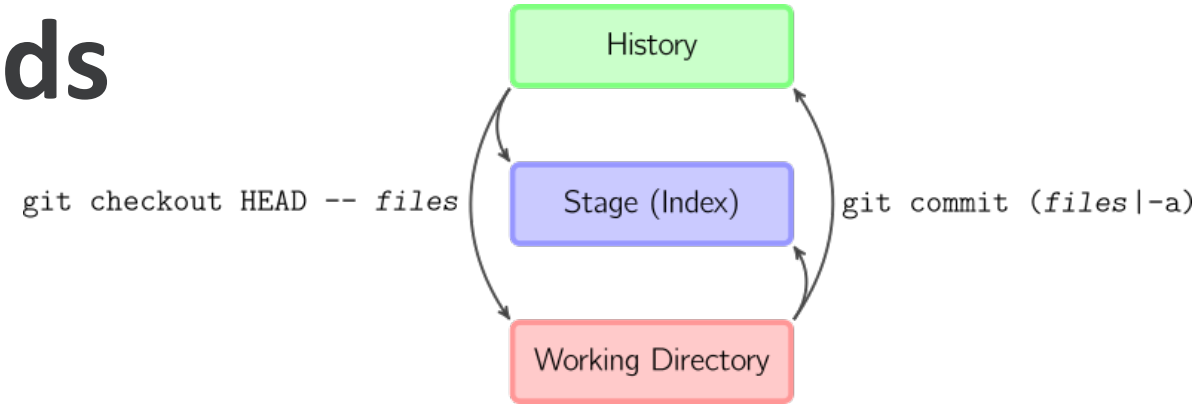
- Updating your local repository to the newest commit:
  - git pull
- Merge another branch into your active branch (e.g. master):
  - git merge <branch>
- After resolving merge conflicts, you need to mark them as merged:
  - git add <filename>
- You can also preview them by using:
  - git diff <source_branch> <target_branch>

Undoing

- Unstage changes:
  - git reset HEAD <file>...
- Discard changes in working directory:
  - git checkout -- <file> …



Combining Commands



- Add + Commit:
  - git commit –a
  - equivalent to running git add on all filenames that existed in the latest commit, and then running git commit
- Reset + Checkout:
  - git checkout HEAD -- <file> …
  - copies *files* from the latest commit to both the stage and the working directory

Python, Anaconda, and Jupyter Notebook

Python and Anaconda

- **Python:** It is Python 3.7!
- **Anaconda:** Python distribution
  - Include many popular packages by default, including Jupyter notebook
  - Make installing additional packages easy
- Your task:
  - Install Anaconda and launch the Jupyter notebook

Install Anaconda

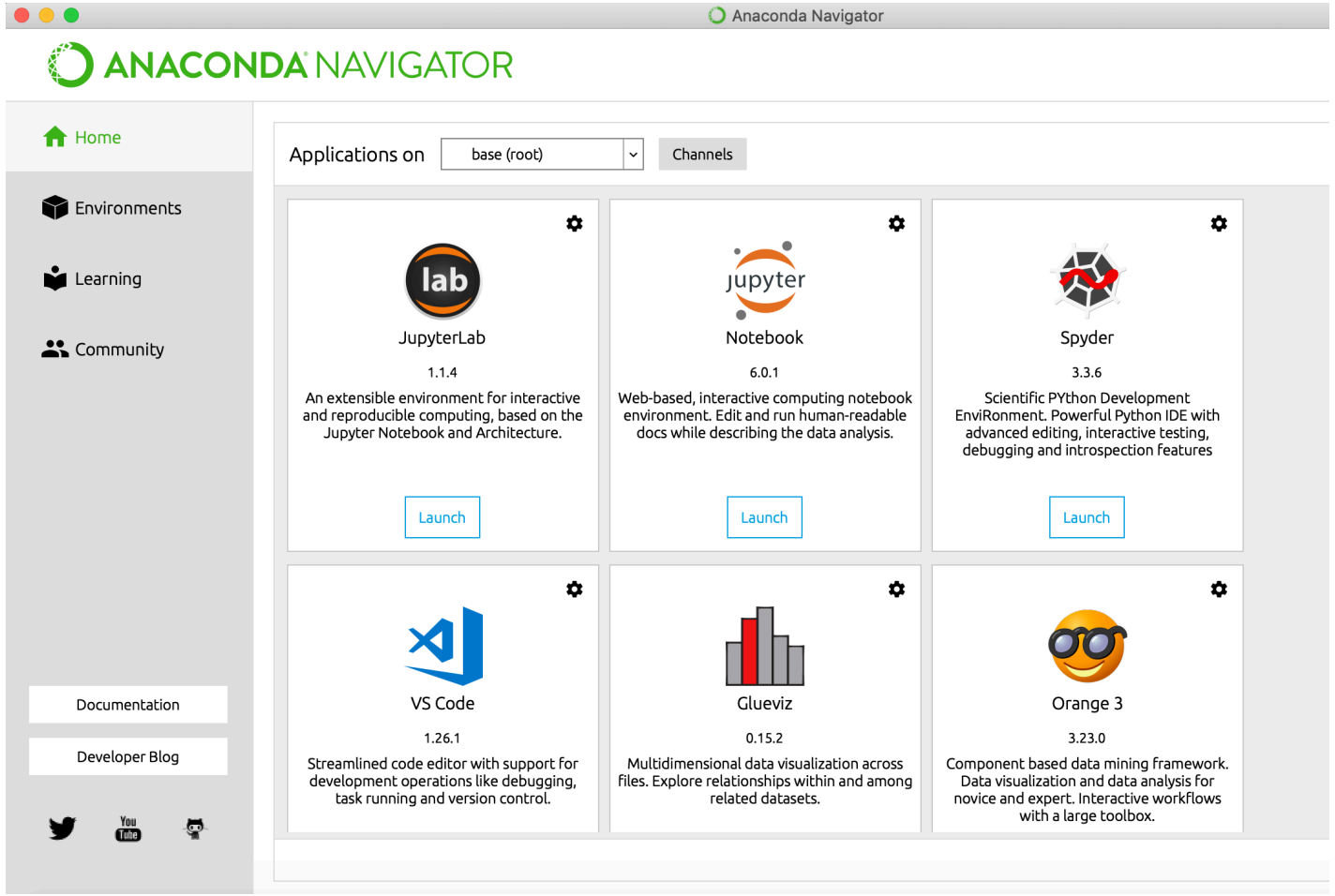
- To install Anaconda (Distribution, not Enterprise or Cloud), follow the installation instructions for your operating system:
- Windows

  https://docs.anaconda.com/anaconda/install/windows/
- Mac

  https://docs.anaconda.com/anaconda/install/mac-os/
- Linux

  https://docs.anaconda.com/anaconda/install/linux/

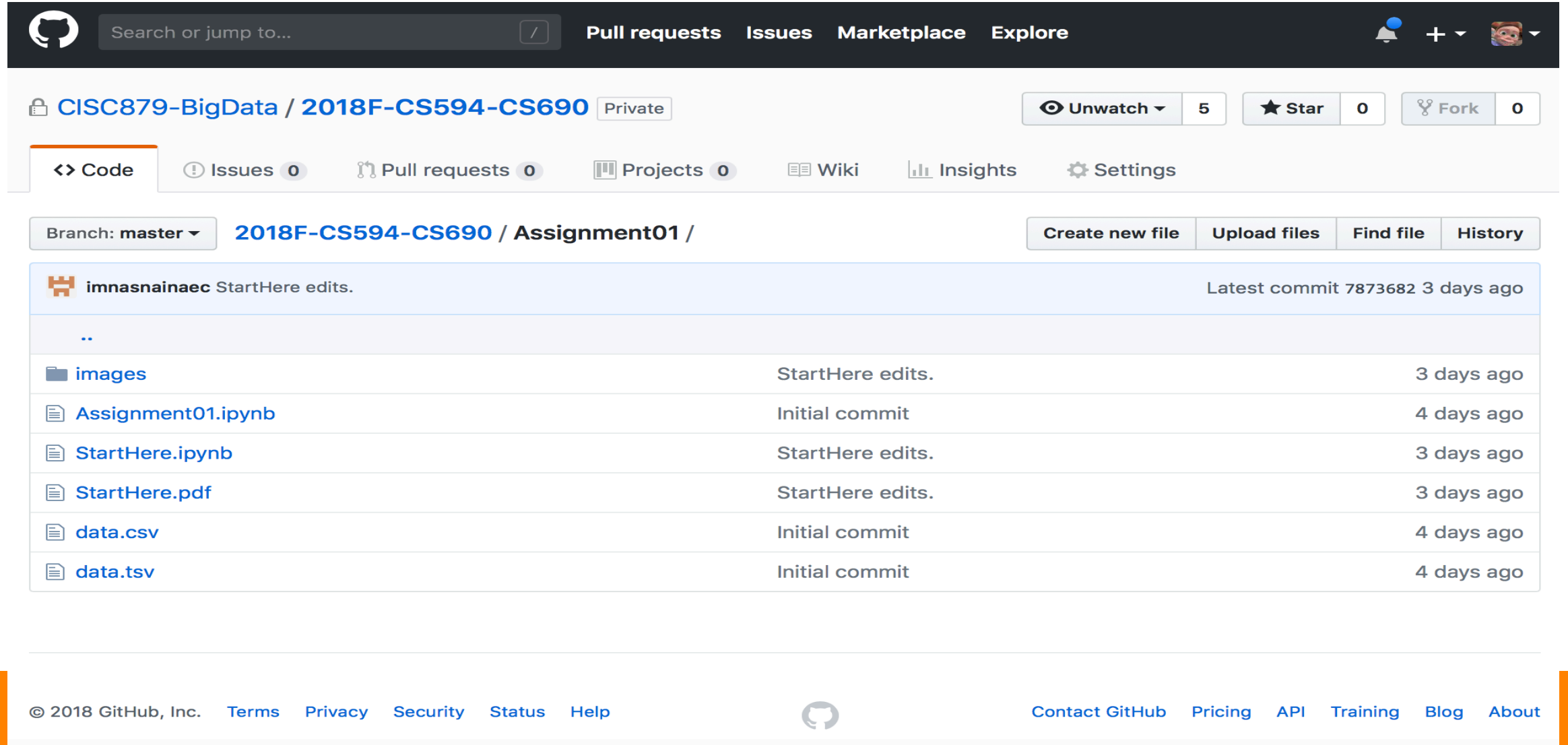
Anaconda Navigator



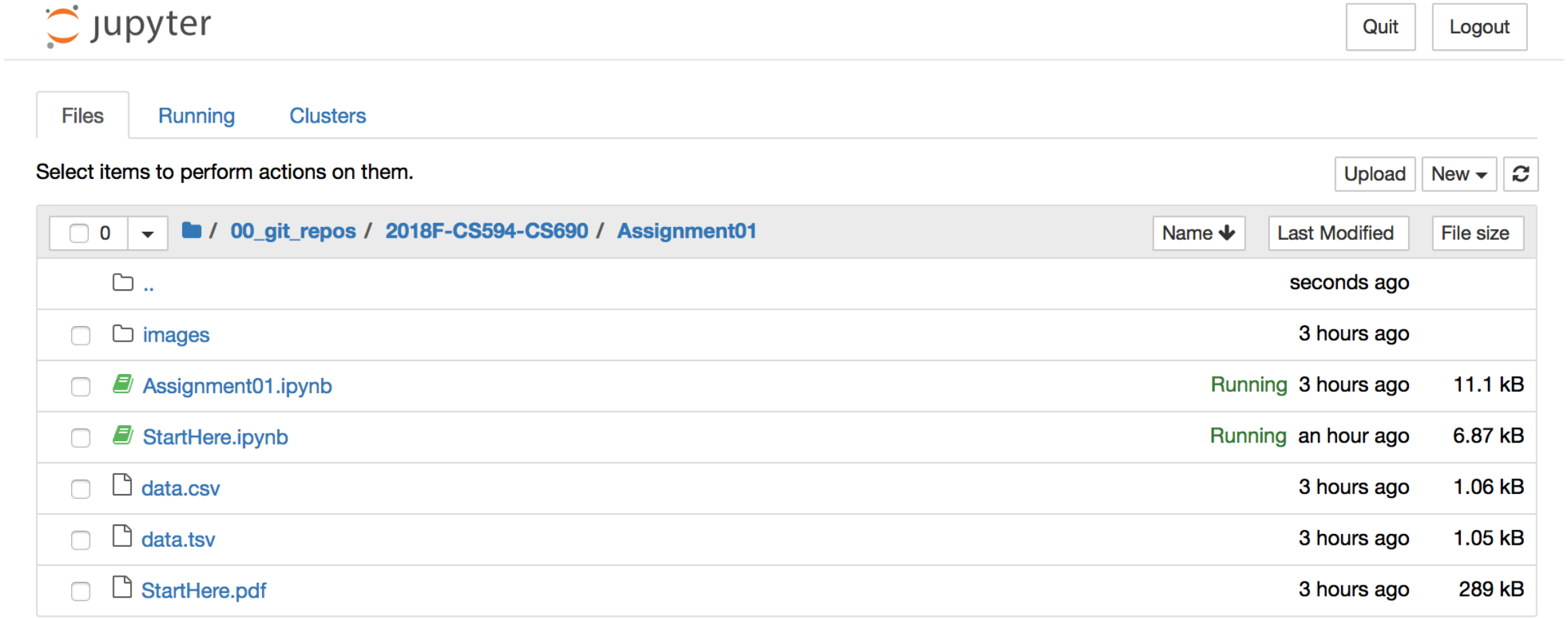
Jupyter Notebook

- **Jupyter notebook:** Our notebook for data analytics
- Programming in a browser
  - Create code in a cell – code in edit mode
  - Run code in a cell – code in command mode
  - Write text before and after code cells – markdown
- Your task:
  - Launch Jupyter from Anaconda GUI (or command line)

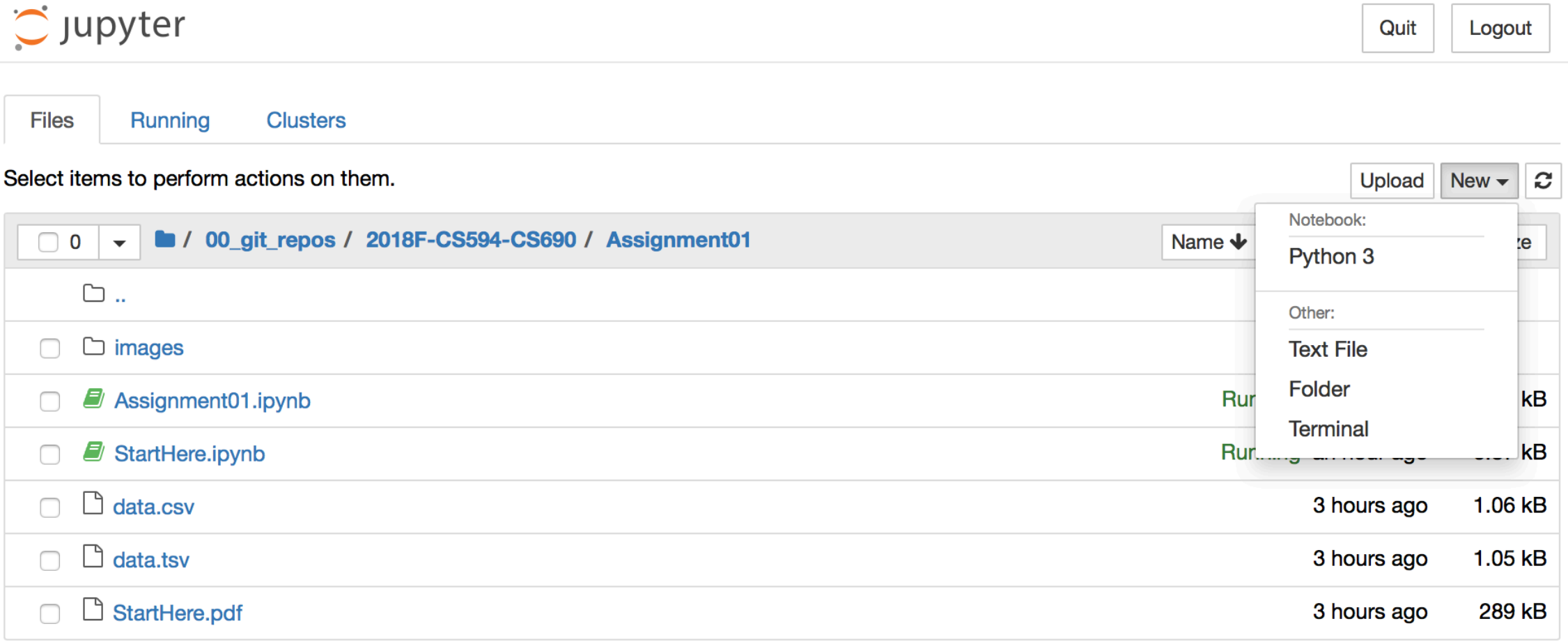
Open your assignment directory



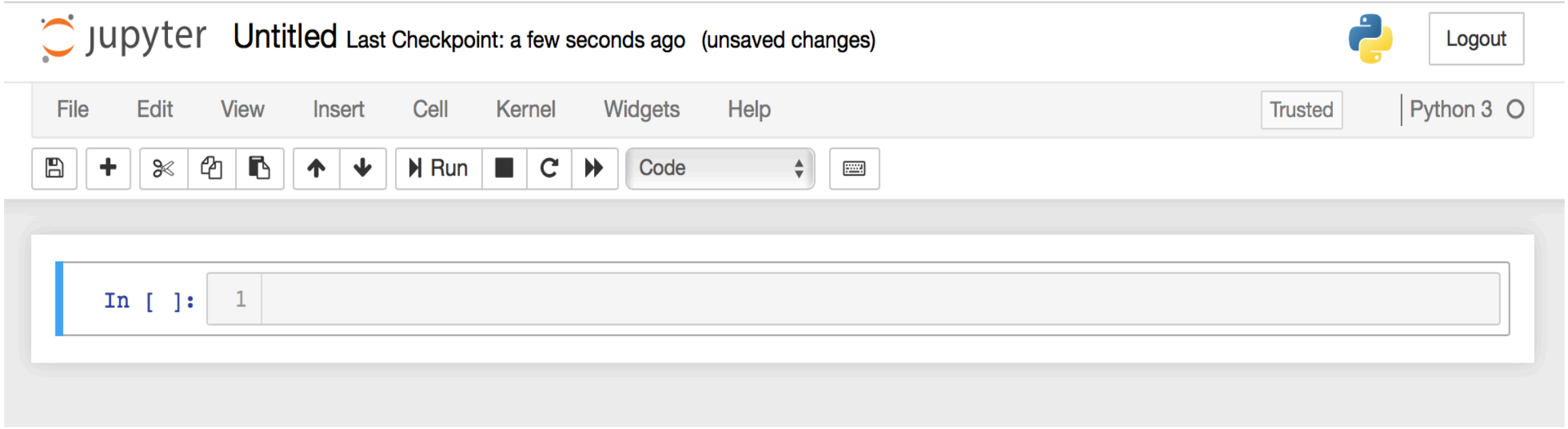
Open your assignment directory



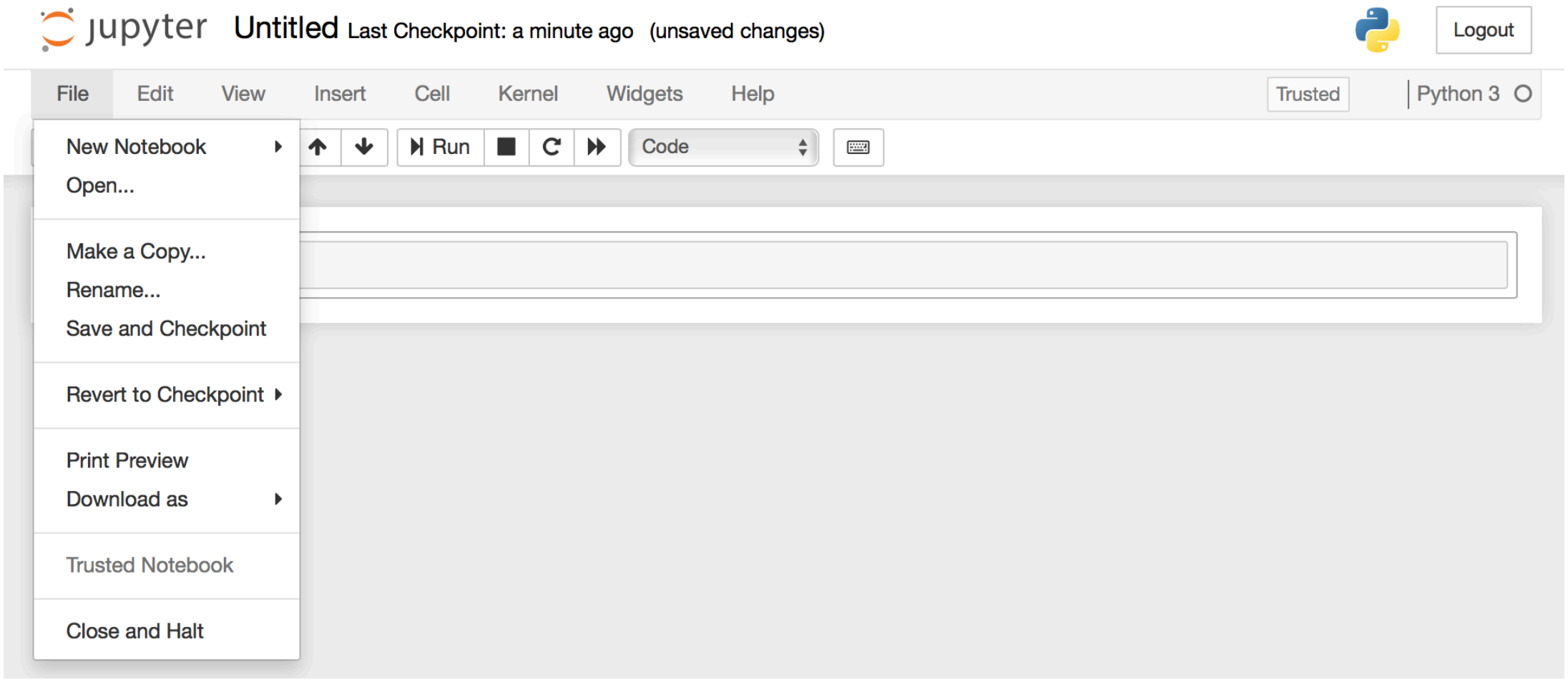
Create your own notebook



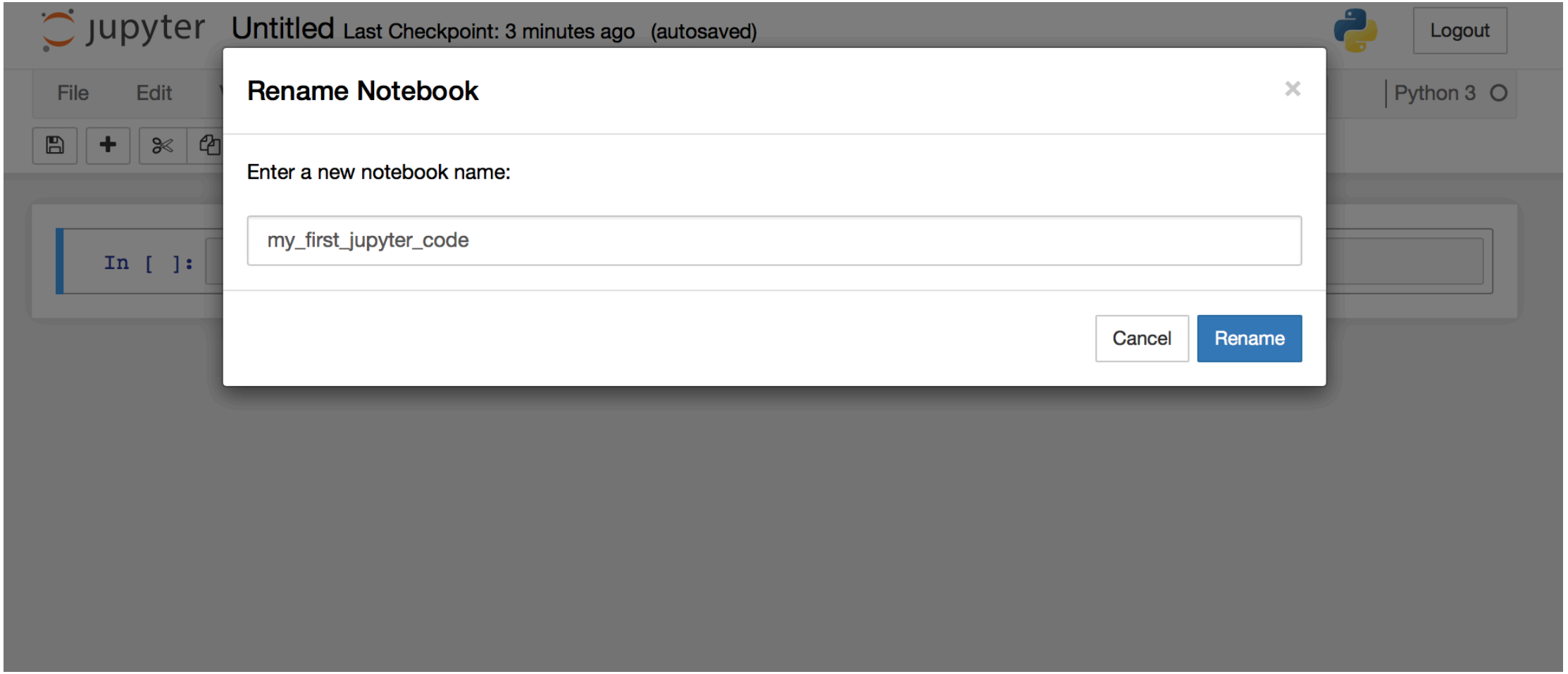
Create your own notebook



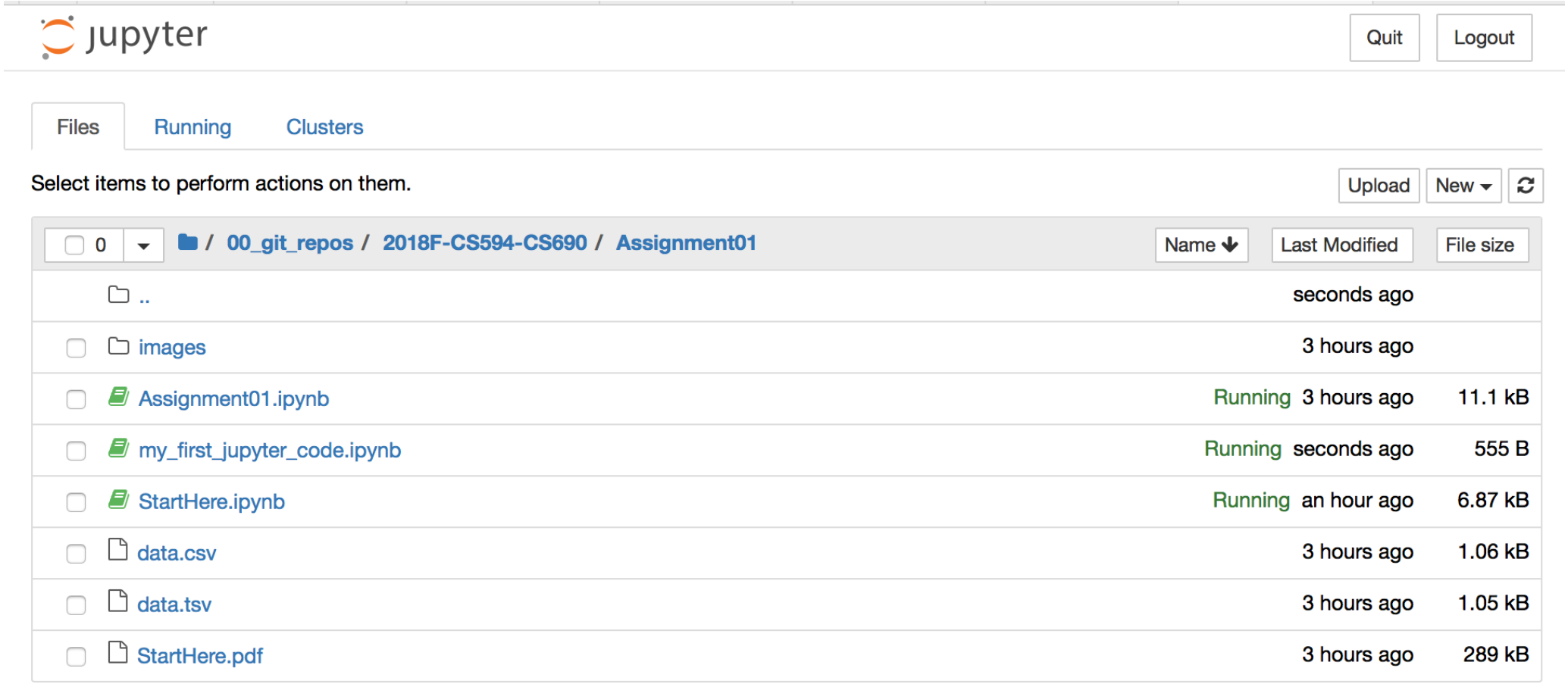
Create your own notebook



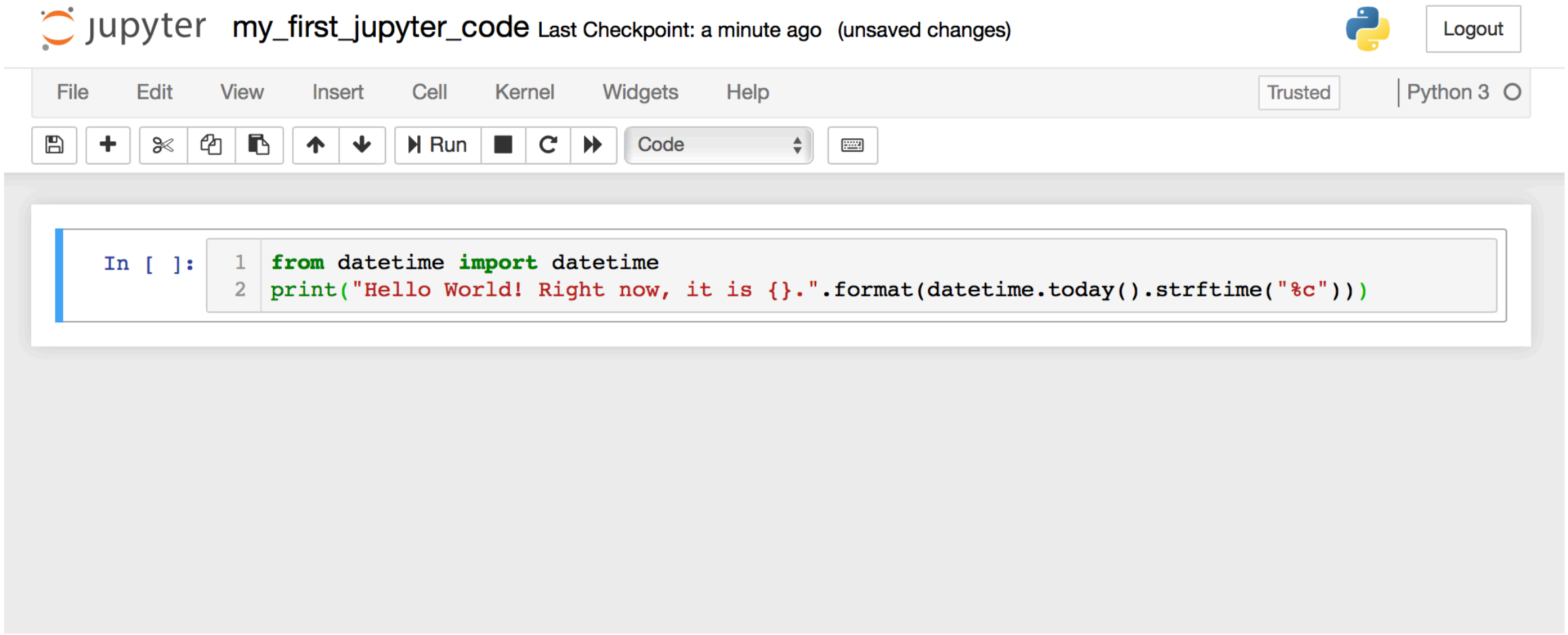
Rename the file



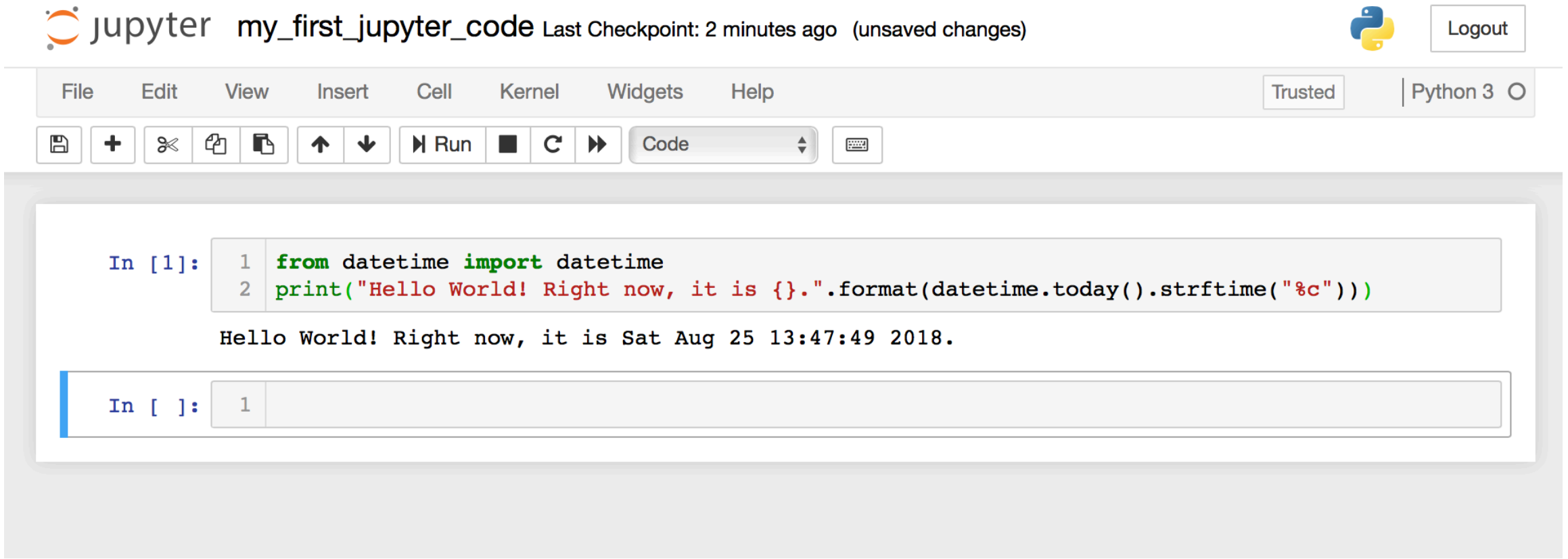
Create your own notebook



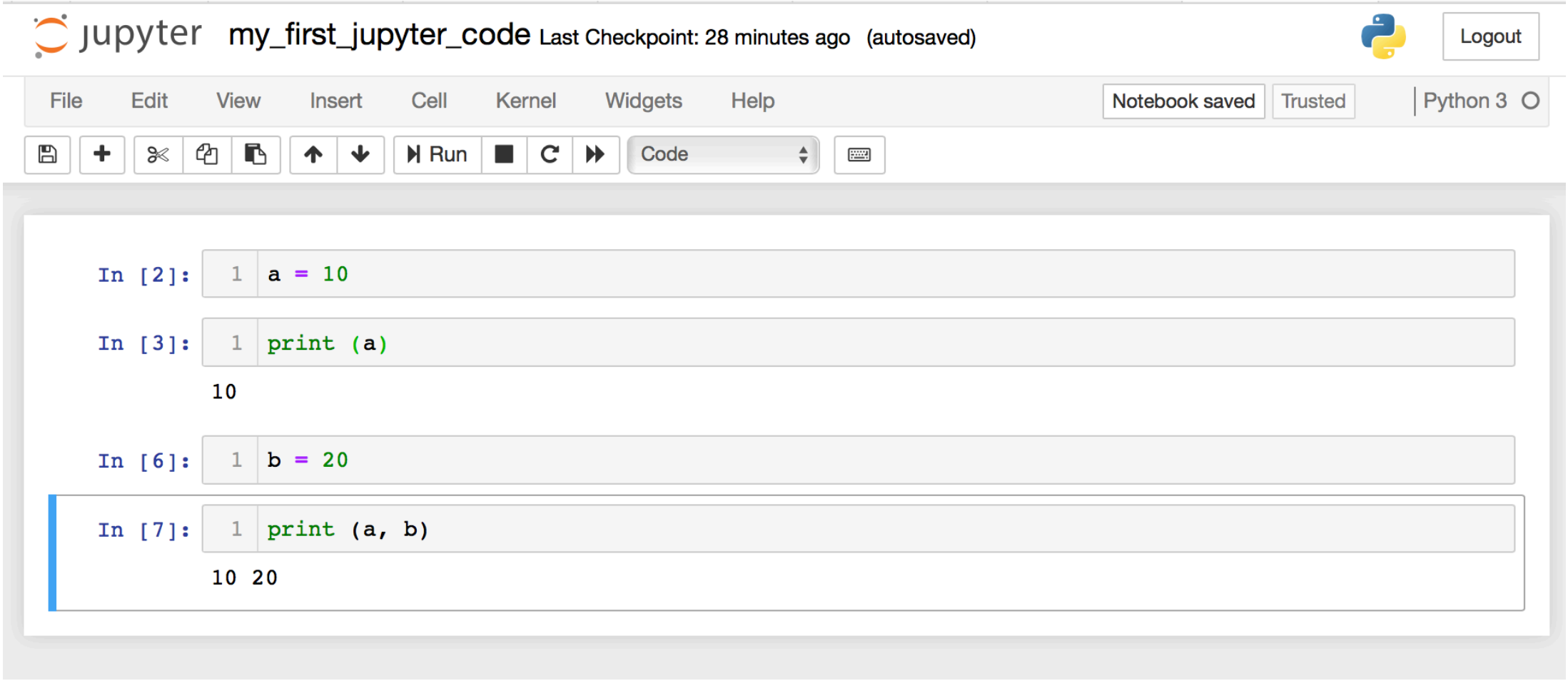
Create a code cell



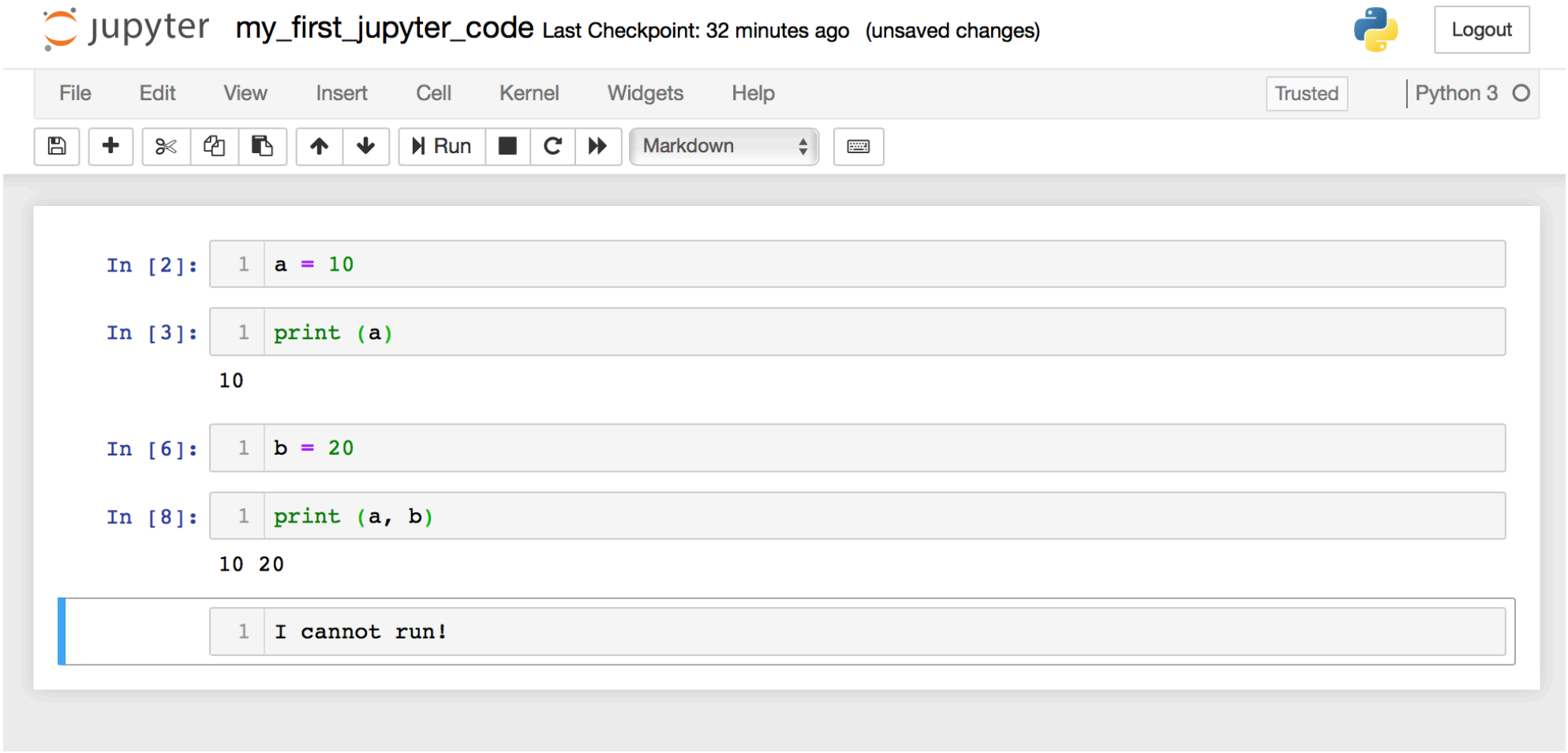
Run your code code



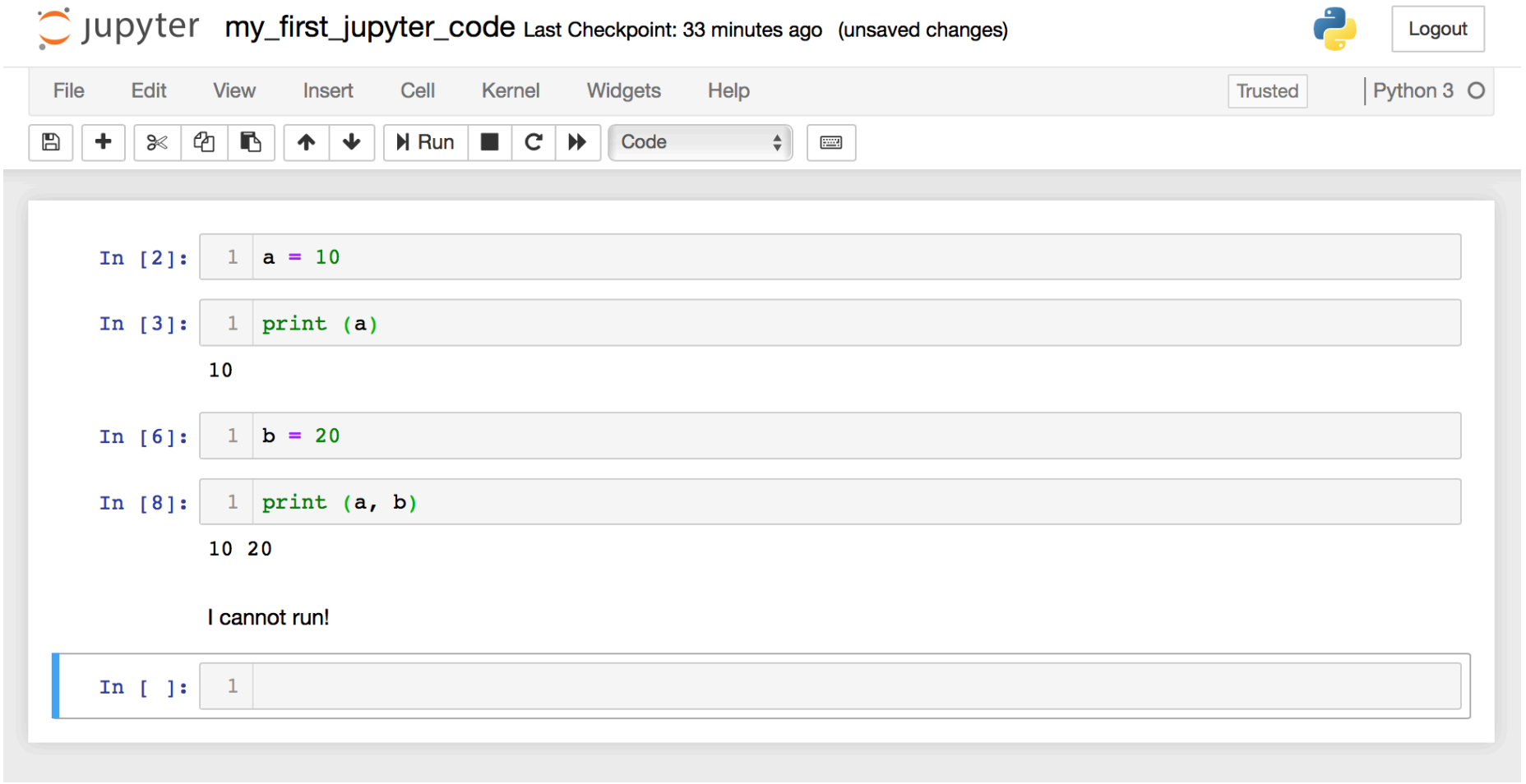
Propagations: from cell to cell



Add text to your notebook



Add text to your notebook



Course Repository

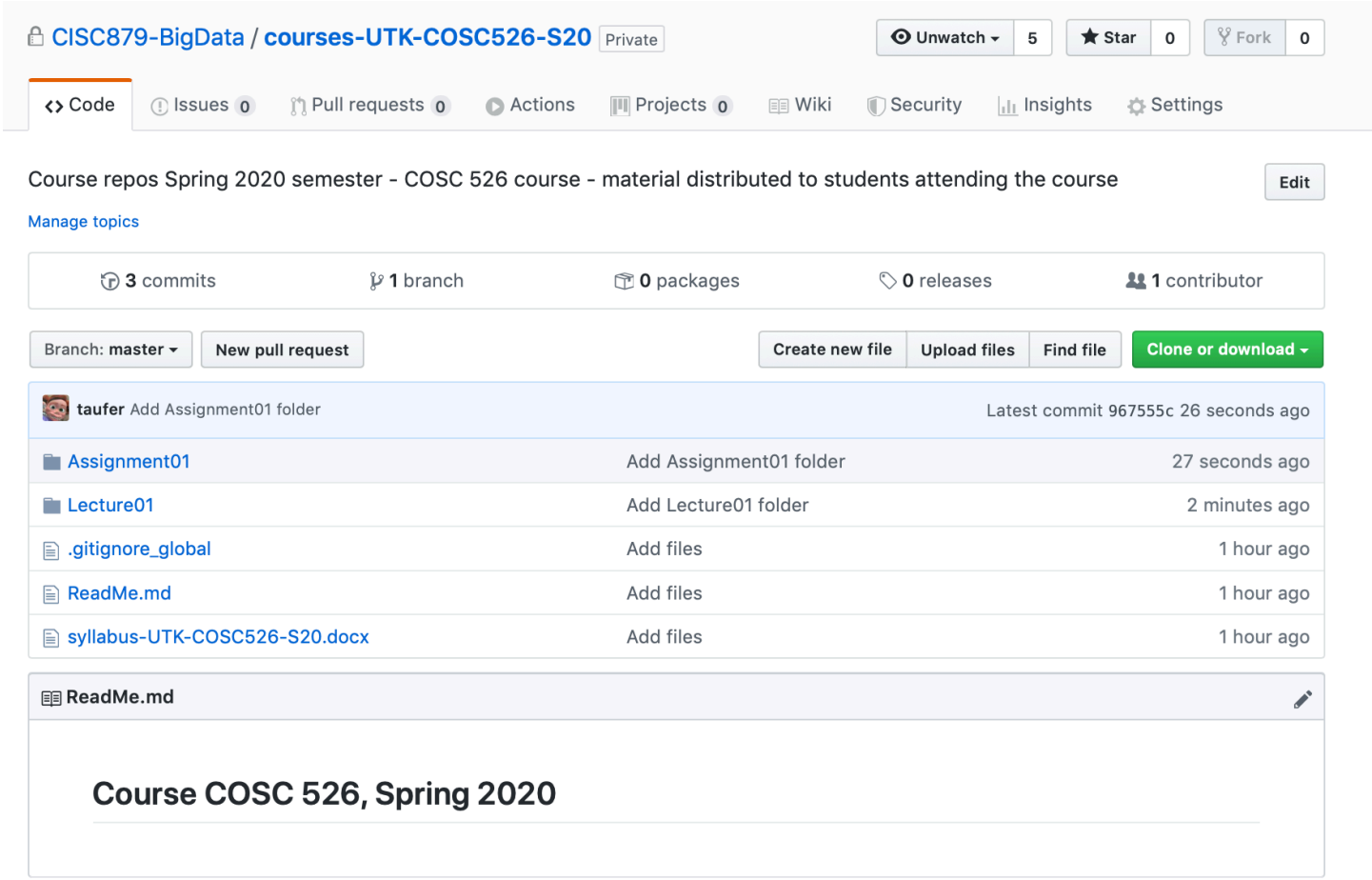
Clone our Class Repository

- Using either the GitHub Desktop app or the git CLI, clone the class repo:

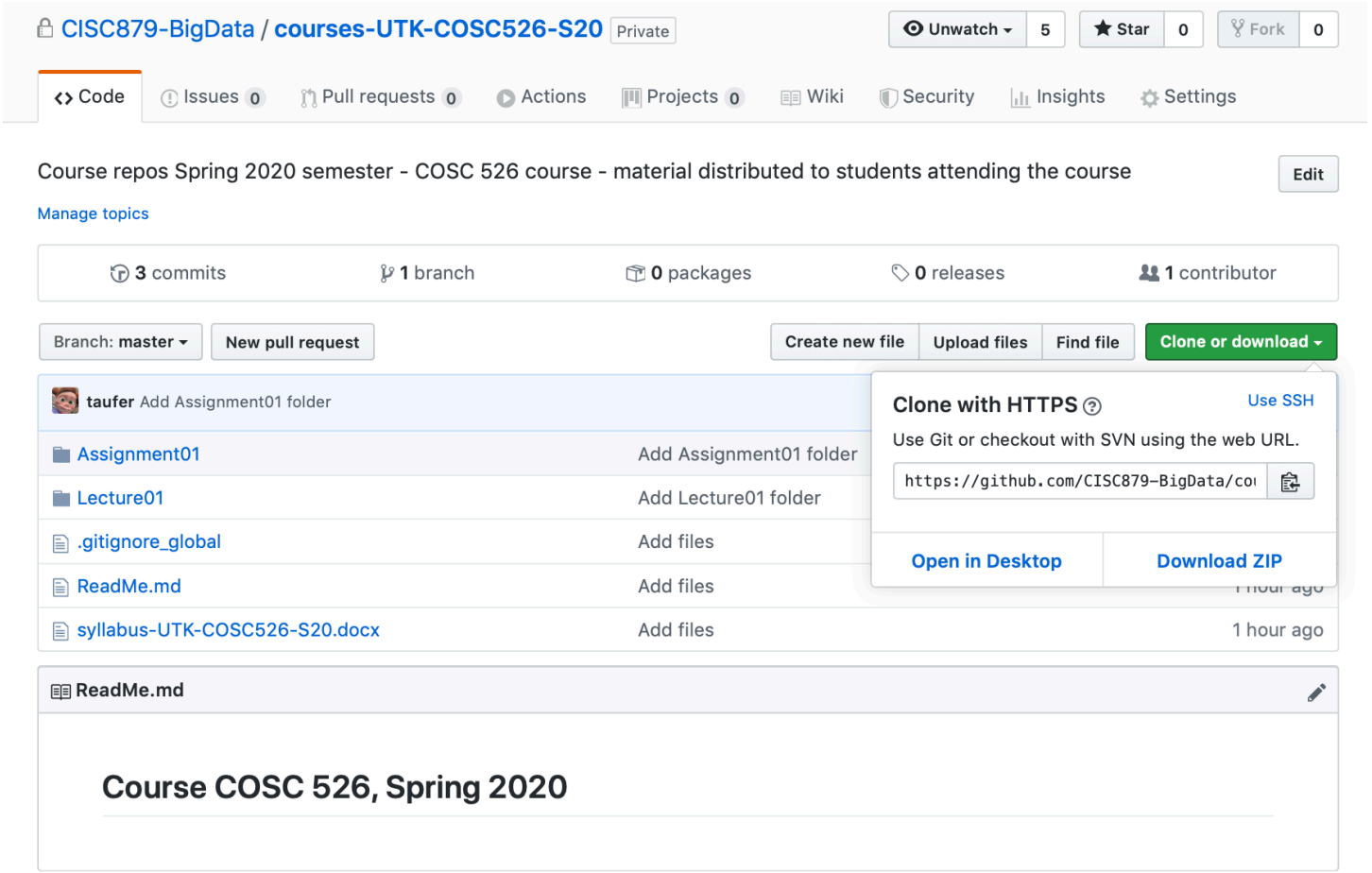
  https://github.com/CISC879-BigData/2021S-COSC526

Clone the course repository

https://github.com/CISC879-BigData/courses-UTK-COSC526-S21



Clone the course repository



Open from CLI

$ git clone https://github.com/CISC879-BigData/courses-UTK-COSC526-S21.git

Open from GitHub Desktop