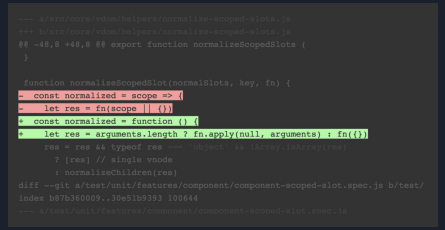
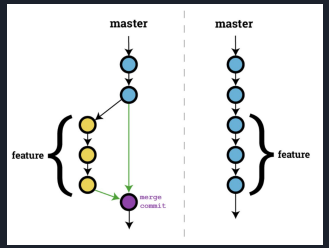
Introduction to git

Faraaz Sareshwala

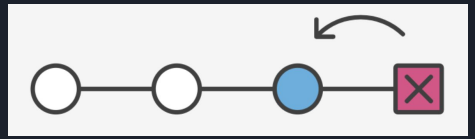
What is Version Control?



Track changes made to source files within the project



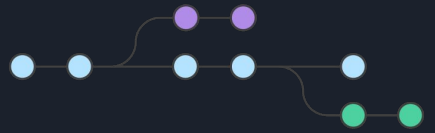
Merge changes from multiple developers



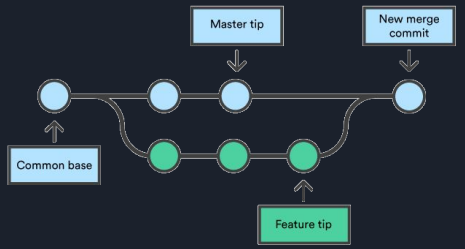
Undo bad changes easily

Reasons to Use Version Control Systems

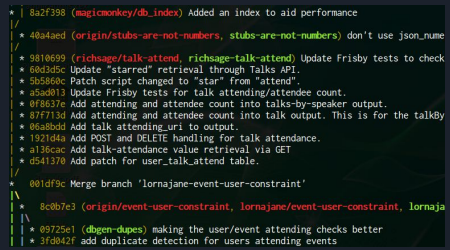
Accountability: who wrote this code?



Branching: support different versions of software, features, etc



Merging: easily incorporate others' changes

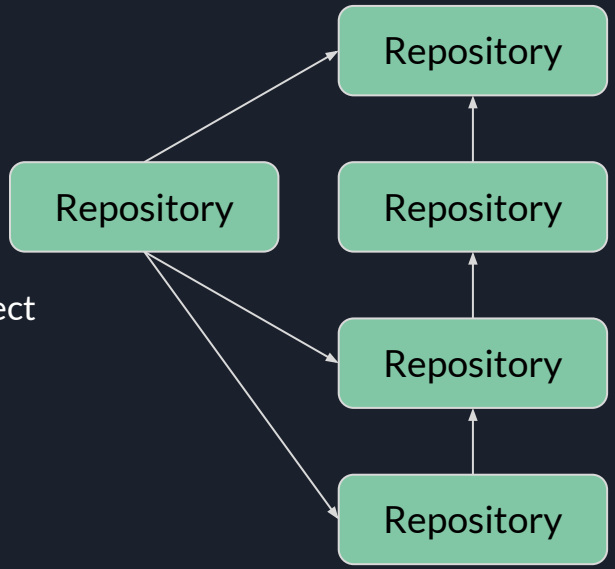


Record Keeping: commit logs show what got committed, when

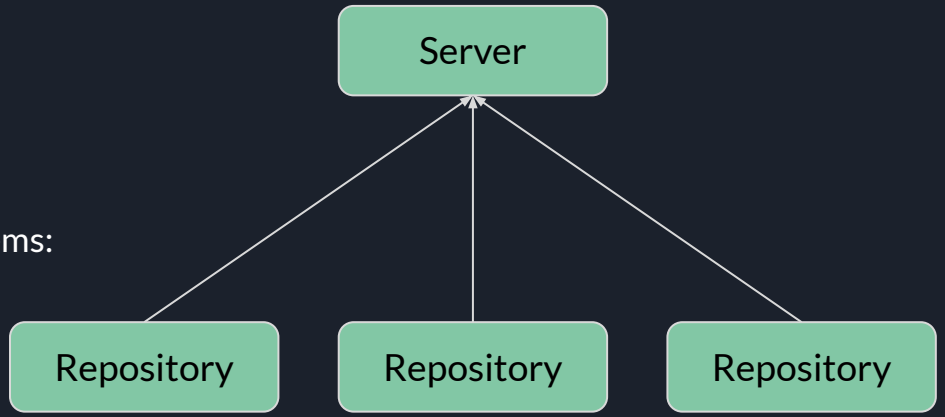
What's So Special About git?

git is a distributed version control system:

- Anyone can clone public repositories
- All changes are local until you push them
- Anyone can make changes without affecting main project
- If project owners accept changes, they can pull them
- No repository is more important than another
- Fast and reliable merges
- Atomic commits



How is git Better?



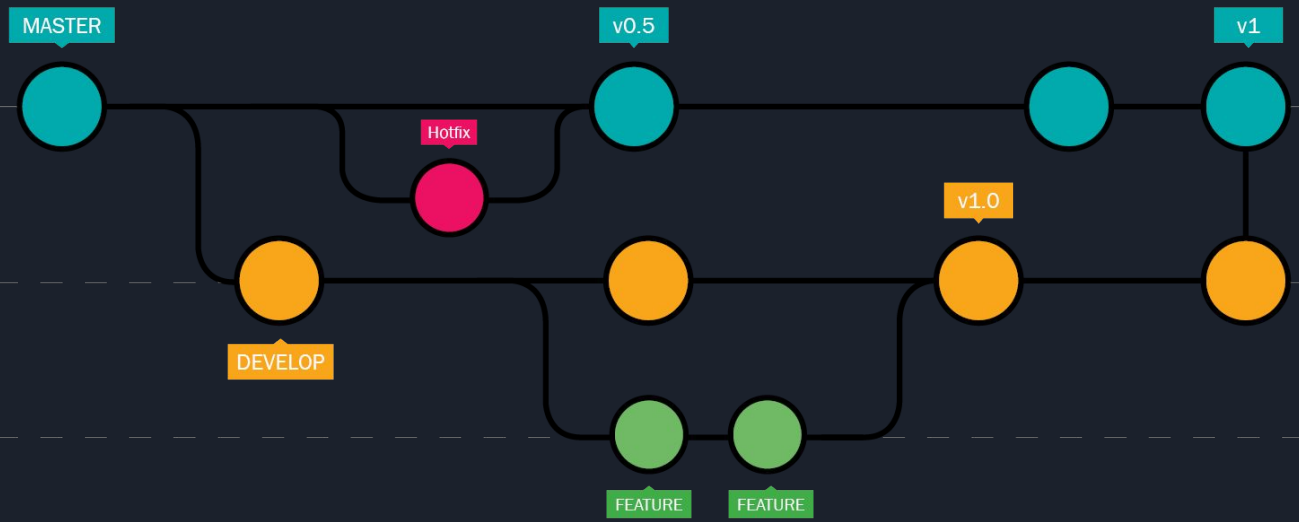
Previous generations of version control systems:

- Centralized
- Single point of failure (i.e. Server)
- Need permission from project owner to make changes (e.g. SVN, discourages contribution)
- Non-atomic commits (e.g. CVS, part of a commit accepted?!)
- Confusing branching and merging (e.g. SVN, no one ends up using it, development occurs in trunk)

Forking

- Each user owns their own repository
- To make changes to a project, you need either:
  - Write permission to the main repository (unlikely)
  - Create an entirely new project (rare)
  - Make a copy for yourself (i.e. clone, fork)

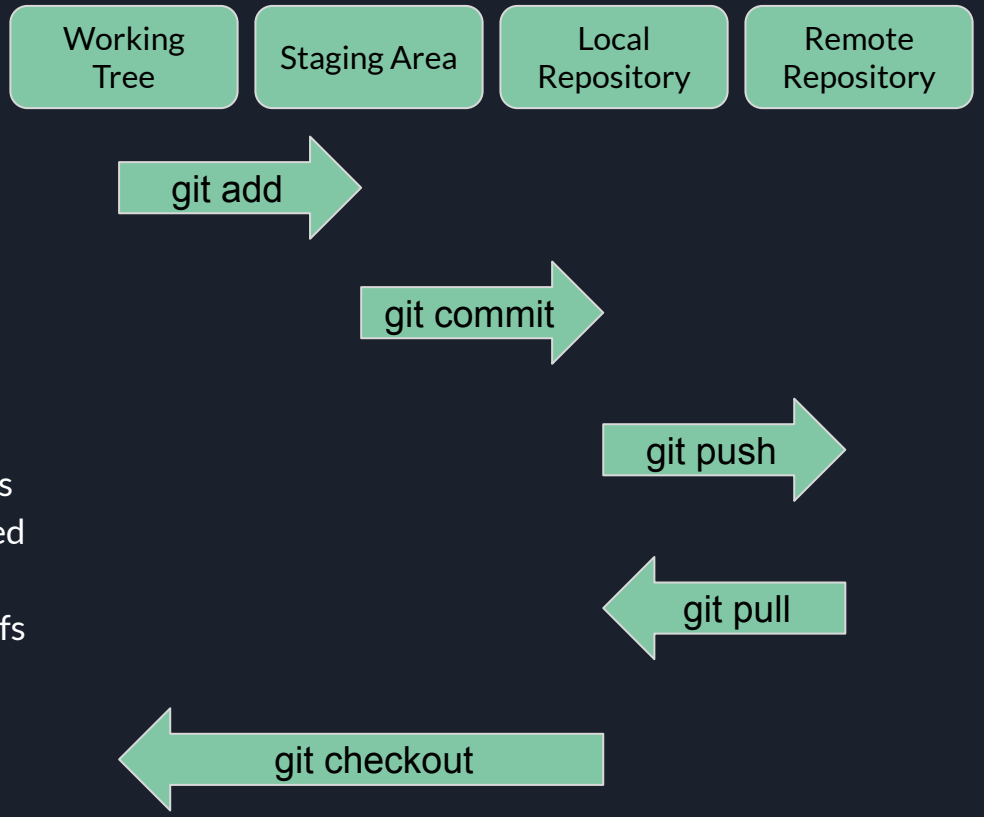
Branching and Merging



Terminology

Git maintains separate file sections:

- **Working Tree:** holds files that are part of the repository
    - This is where you do your work
- **Staging Area:** where modifications are kept before they are committed
- **Local Repository:** collection of diffs (commits)



File States

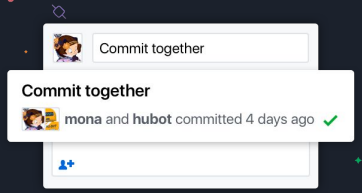
A file in the working tree of git can be in the following states:

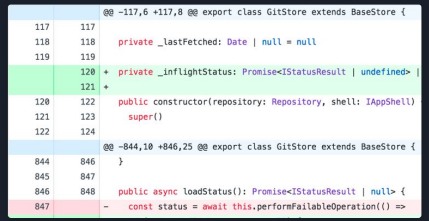
- Untracked
  - The file is not tracked by git
  - It has never been staged or committed
- Modified
  - The file is tracked by git
  - Has some local modifications that have not yet been staged to the index
- Staged
  - The file has has some local modifications and is in the staging area, ready to be committed

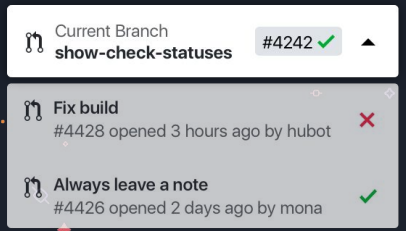
```
On branch master
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)

modified: hello.py

Untracked files:
(use "git add <file>..." to include in what
will be committed)

hello.pyc
```

- Tracked
  - The file is committed, tracked, and there is no local modification to it

GitHub Desktop







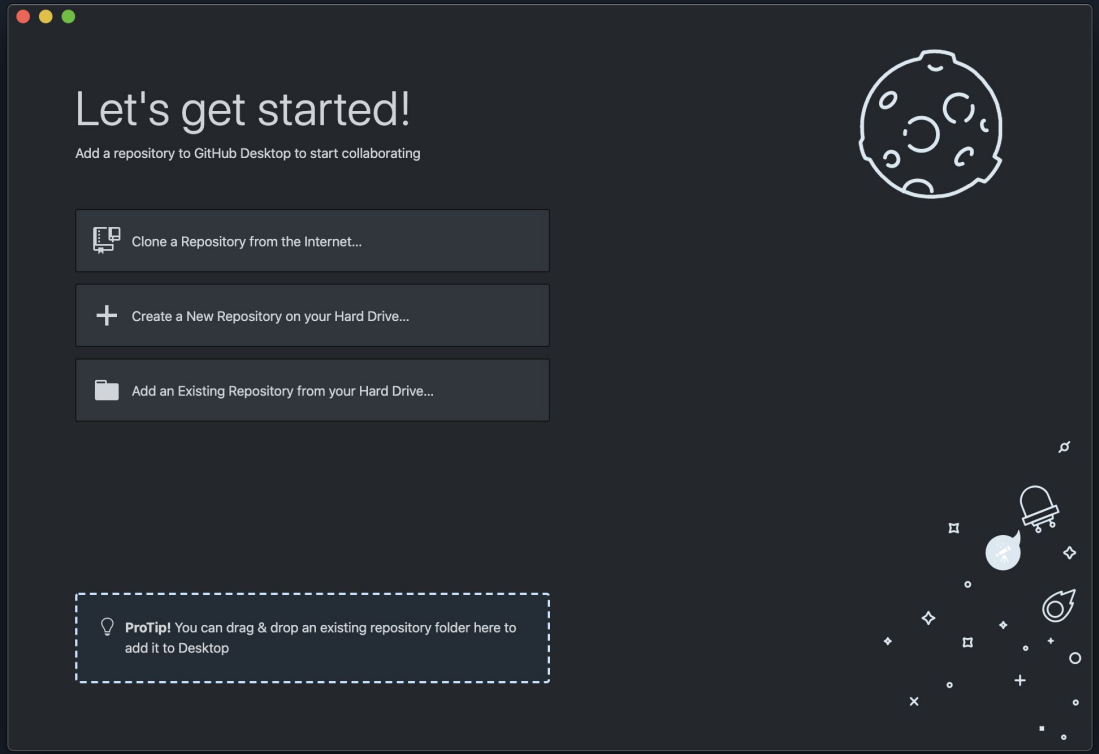
All the features of git, but in an easy to use GUI

Download and install from https://desktop.github.com

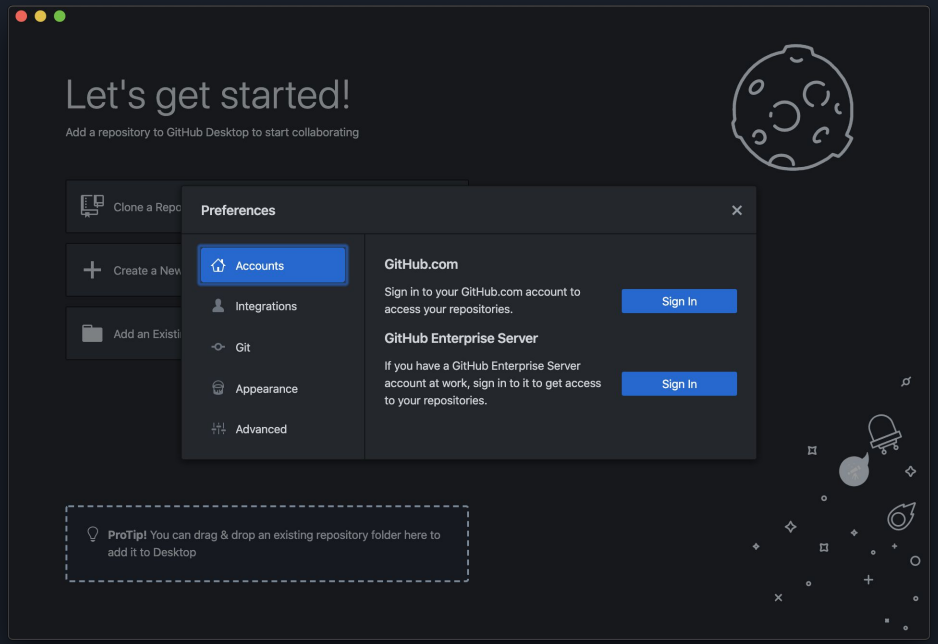
GitHub Desktop

Following instructions assume you have already completed the student setup instructions.

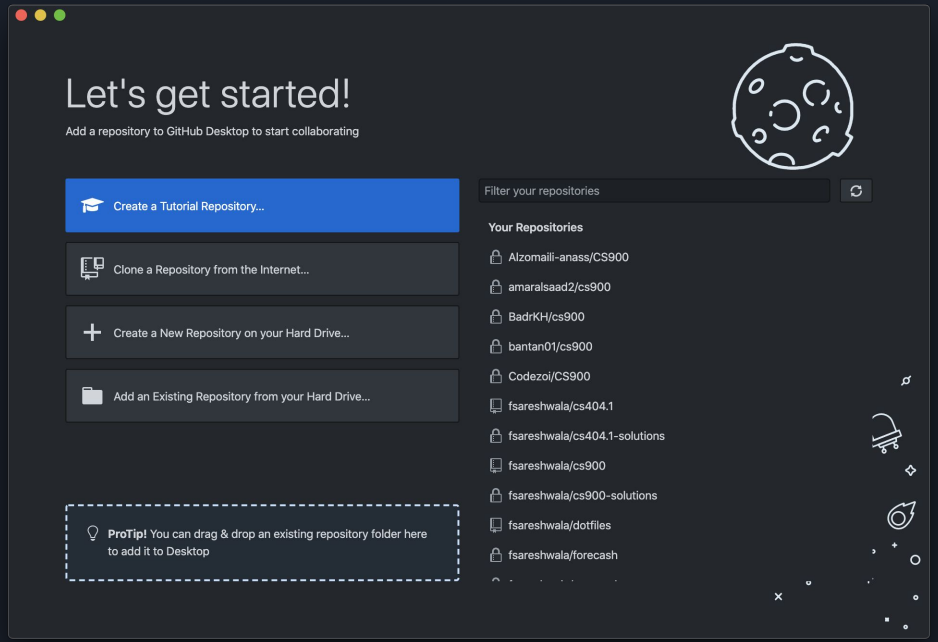
See Homework 1 for the link to the student setup instructions.



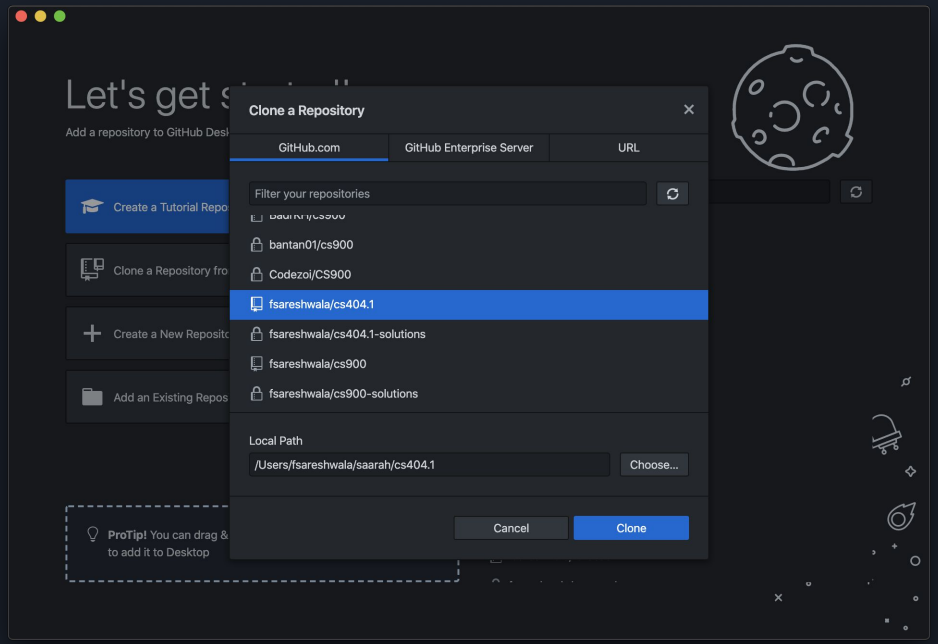
GitHub Desktop (Sign in)



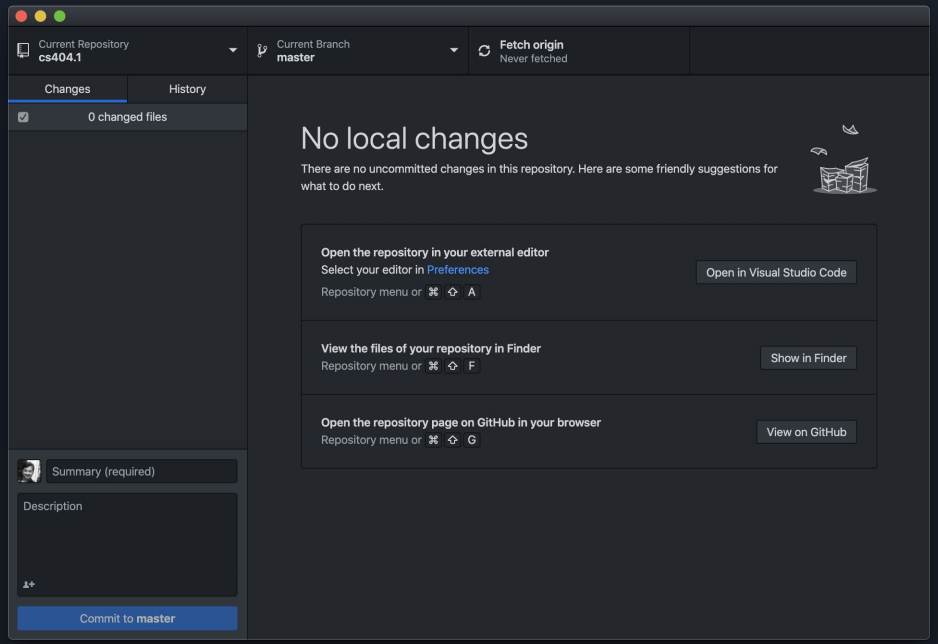
GitHub Desktop (Main Screen)



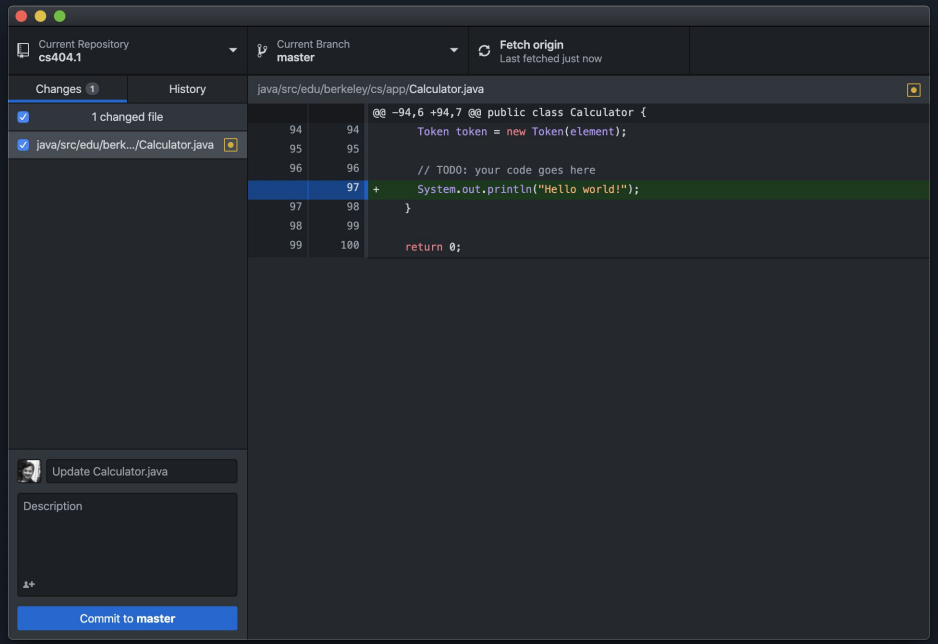
GitHub Desktop (Clone)



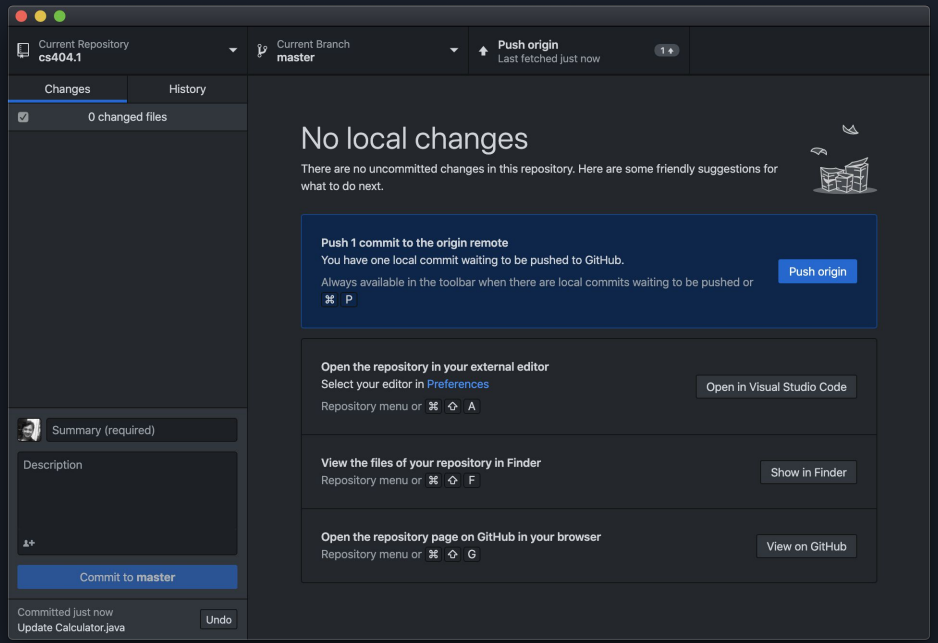
GitHub Desktop (Cloned)



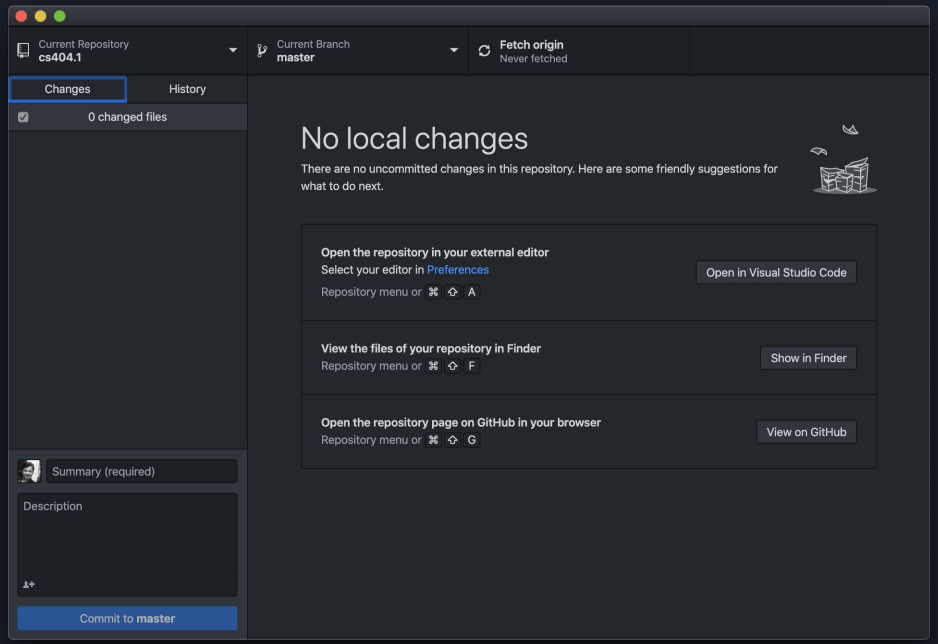
GitHub Desktop (Changes)



GitHub Desktop (Local Commits)



GitHub Desktop (Fetch)



GitHub Desktop (Remote Commits)