



Introduction to git

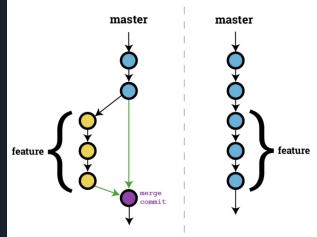
Faraaz Sareshwala

What is Version Control?

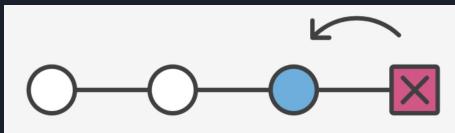
```
--- a/src/core/dom/helpers/normalize-scoped-slots.js
+++ b/src/core/dom/helpers/normalize-scoped-slots.js
@@ -68,8 +68,8 @@ export function normalizeScopedSlots (
}

function normalizeScopedSlot(normalizedSlots, key, fn) {
- const normalized = scope => {
+ let res = fn(scope || ())
+ const normalized = function () {
+ let res = arguments.length ? fn.apply(null, arguments) : fn()
res = res ?? typeof res === 'object' ? Array.isArray(res)
? res : single vnode
: normalizeChildren(res)
- normalizedChildren(res)
diff --git a/test/unit/features/component/component-scoped-slot.spec.js b/test/
index b67360009..30e51b9393 100644
--- a/test/unit/features/component/component-scoped-slot.spec.js
```

Track changes made to source files within the project



Merge changes from multiple developers

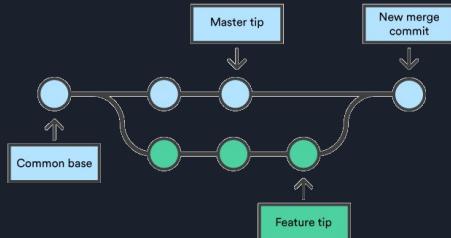


Undo bad changes easily

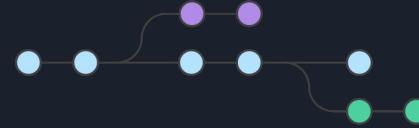
Reasons to Use Version Control Systems



Accountability: who wrote this code?



Merging: easily incorporate others' changes



Branching: support different versions of software, features, etc

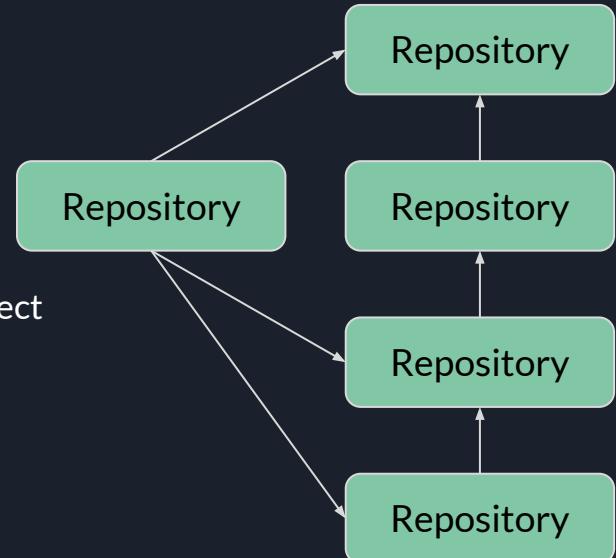
```
* | 8a2f398 (magimonkey/db_index) Added an index to aid performance
|/
* | 40a4ae0 (origin/stubs-are-not-numbers, stubs-are-not-numbers) don't use json_num
|/
* | 981699 (richsage/talk-attend, richsage-talk-attend) Update Frisby tests to check
|   6dd3dc5 Update "starred" retrieval through Talks API.
|   5b5650c Patch script changed to "star" from "attend".
|   a5ad803 Update Frisby tests for talk.attending_count and talk.attendee_count.
|   049003 Add attending and attendee count into talkBy-speaker output.
|   3f77136 Add attending and attendee count into talk output. This is for the talkBy
|   06ab8bd Add talk.attending_uri to output.
* | 1921d4a Add POST and DELETE handling for talk attendance.
* | a136cac Add talk-attendance value retrieval via GET
* | d541378 Add patch for user_talk_attend table.
|/
* | 001df9c Merge branch 'lornajane-event-user-constraint'
|/
* | 8c08e3 (origin/event-user-constraint, lornajane/event-user-constraint, lorna
|   \|
* | 89729e1 (d8gen-dupes) making the user/event attending checks better
|   \|
* | 3f38a2f add duplicate detection for users attending events
```

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:

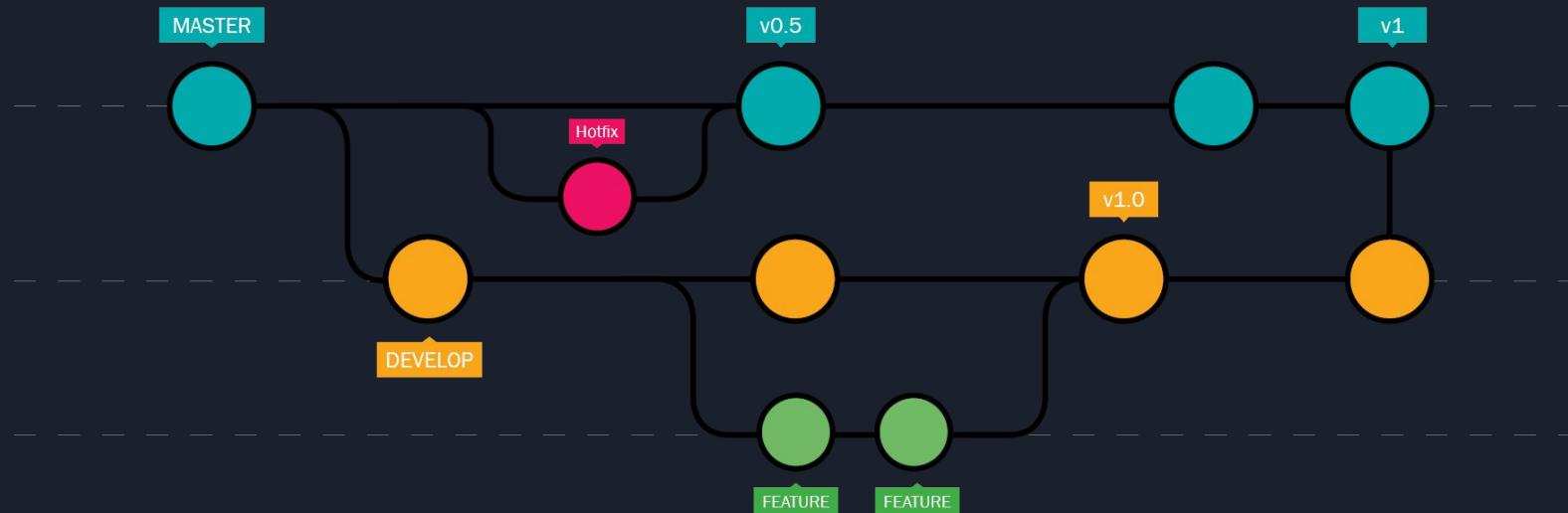
- Centralized
 - Single point of failure (i.e. Server)
 - Need permission from project owner to make changes (e.g. SVN, discourages contribution)
 - Non-atomic commits (e.g. CVS, part of a commit accepted?!)
 - Confusing branching and merging (e.g. SVN, no one ends up using it, development occurs in trunk)
-
- ```
graph TD; Server[Server] --> Repository1[Repository]; Server --> Repository2[Repository]; Server --> Repository3[Repository]
```
- The diagram illustrates a centralized version control system. At the top center is a rounded rectangular box labeled "Server". Three white arrows point downwards from the "Server" box to three separate rounded rectangular boxes, each labeled "Repository". This visual representation emphasizes that all changes and data are stored in a single, central location, which is a key characteristic of previous generation VCS systems like SVN.

# 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

Working Tree

Staging Area

Local Repository

Remote Repository

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)

git add

git commit

git push

git pull

git checkout

# 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



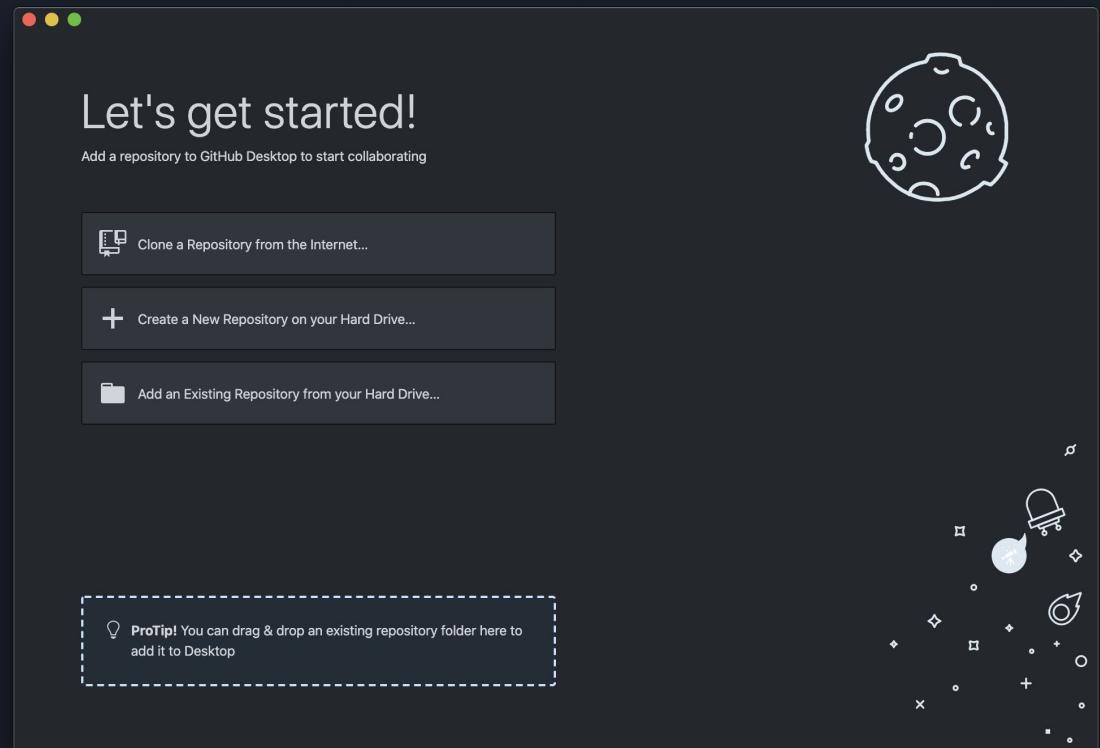
All the features of git, but in an easy to use GUI

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

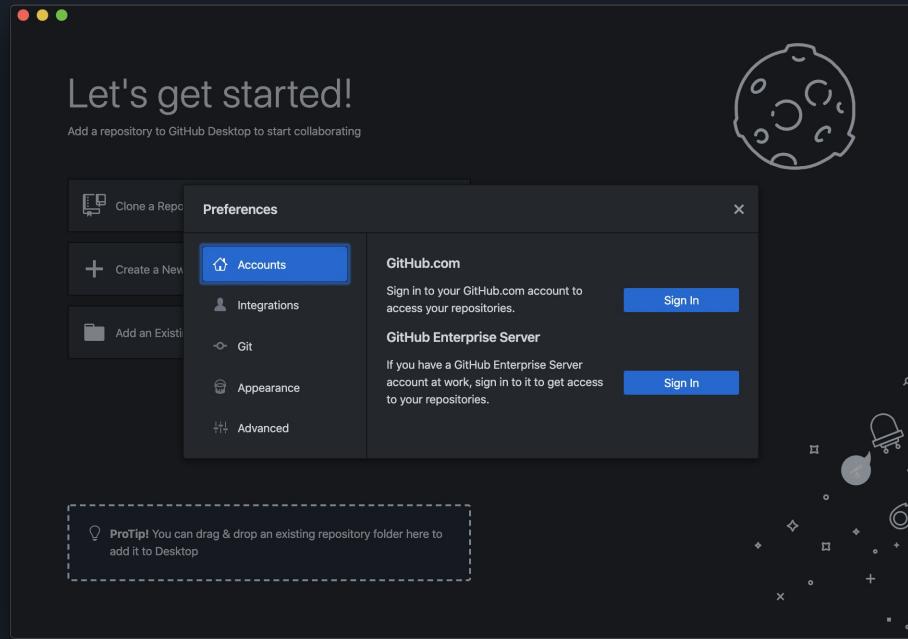
# GitHub Desktop

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

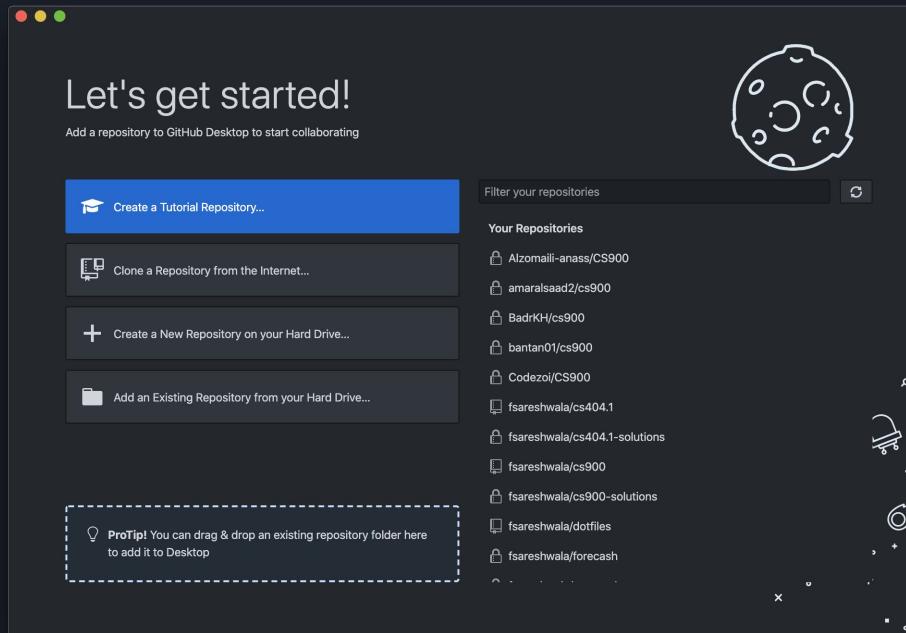
See Homework 1 for the link  
to the student setup  
instructions.



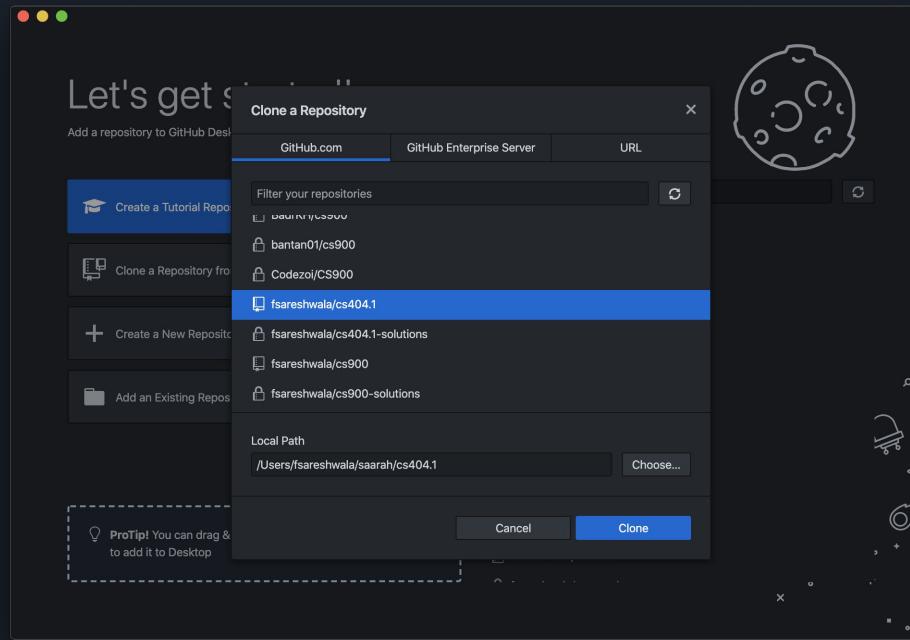
# GitHub Desktop (Sign in)



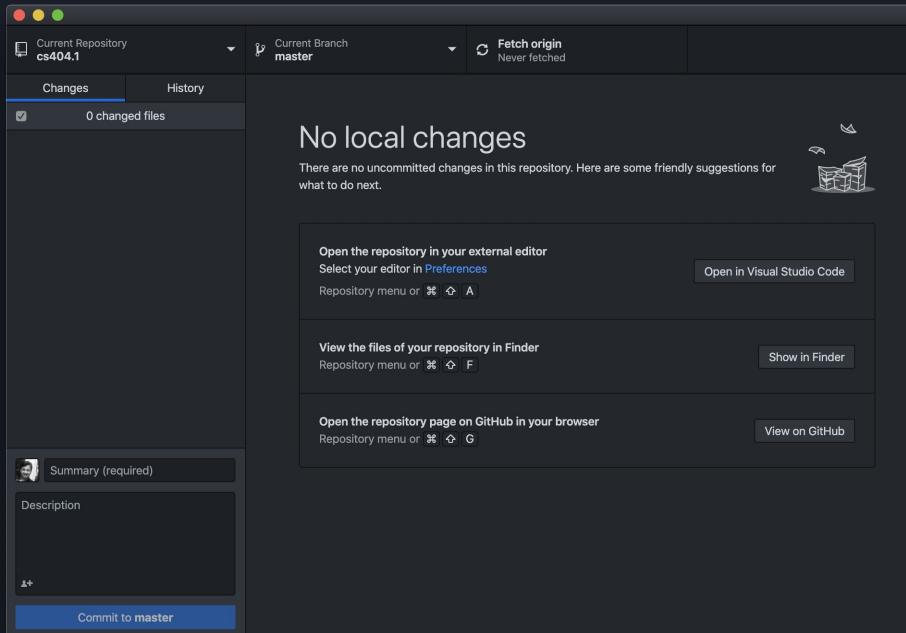
# GitHub Desktop (Main Screen)



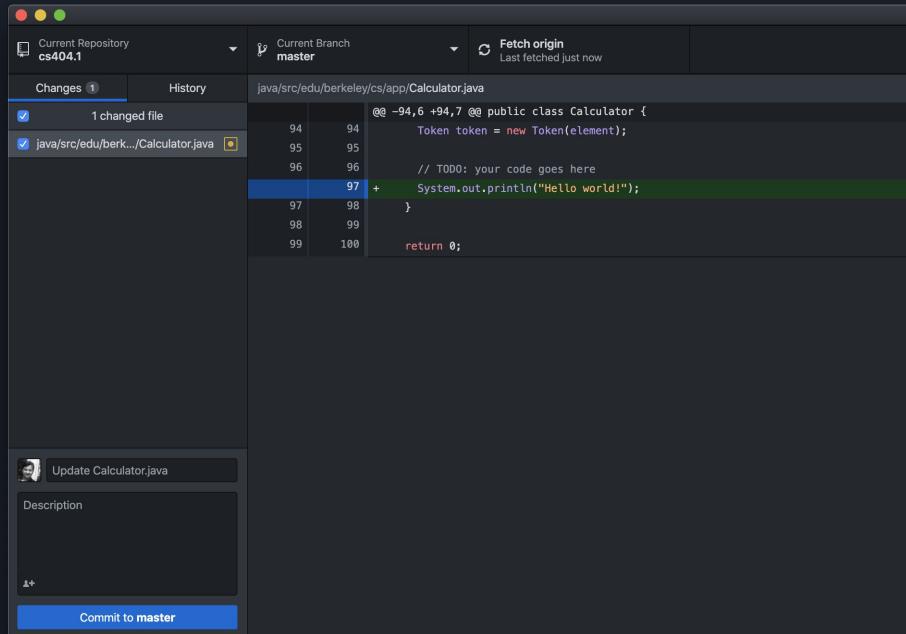
# GitHub Desktop (Clone)



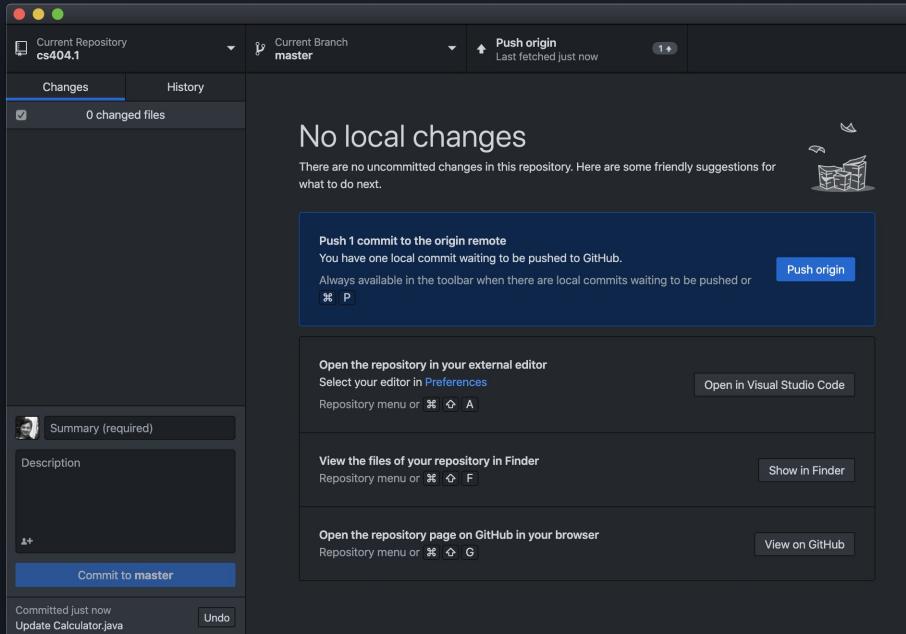
# GitHub Desktop (Cloned)



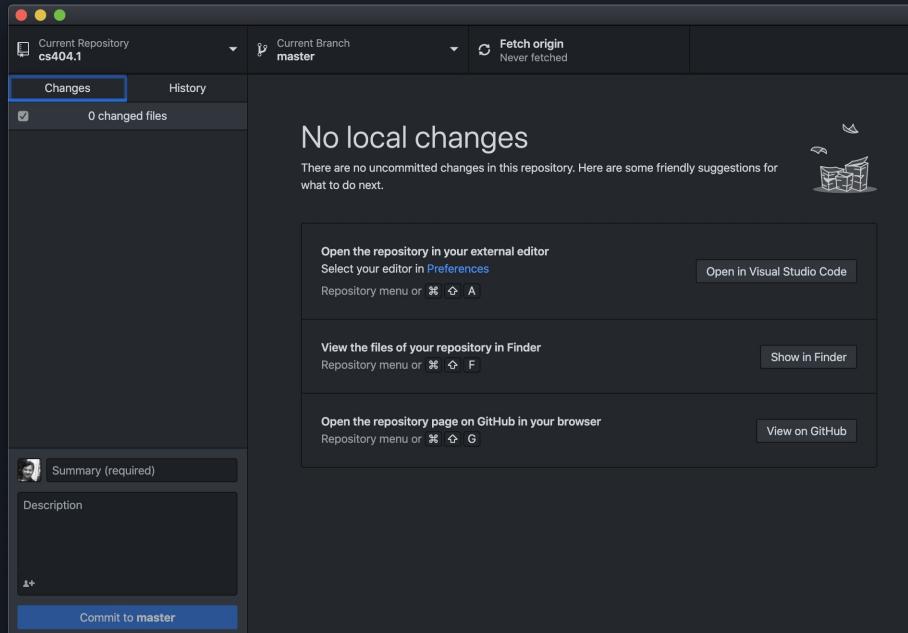
# GitHub Desktop (Changes)



# GitHub Desktop (Local Commits)



# GitHub Desktop (Fetch)



# GitHub Desktop (Remote Commits)

