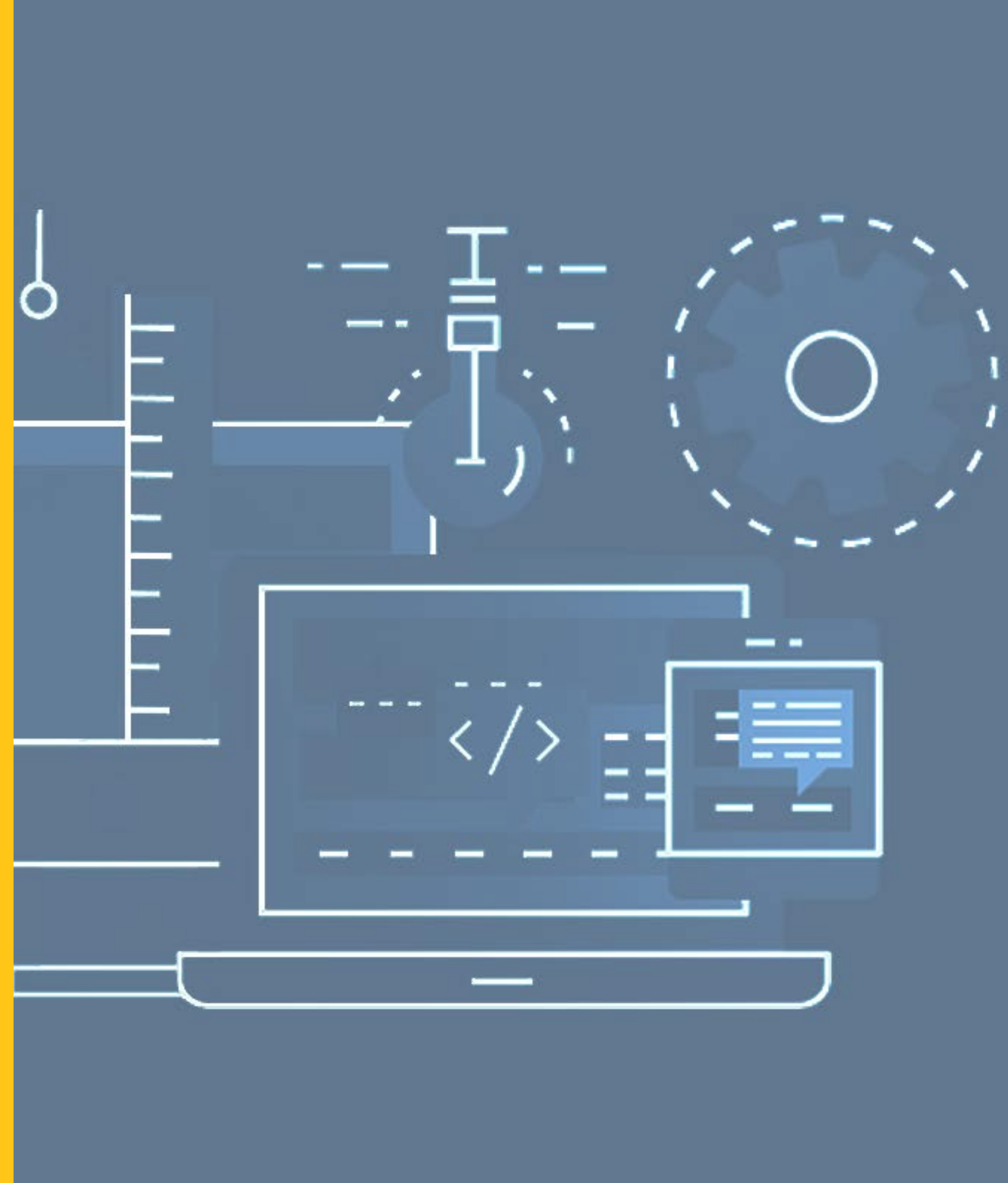


SHELLY GRAHAM, 01/07/2021

# WEB DEV 3

## WINTER 2021

Week 1: Git







THAT'S ME!

- From Germany
- Living in Charlotte
- Freelancer
- Pasta Lover
- Soccer Enthusiast
- Recent Animal Crossing Addict



# TELL ME ABOUT YOU

**NAME**

**AGE**

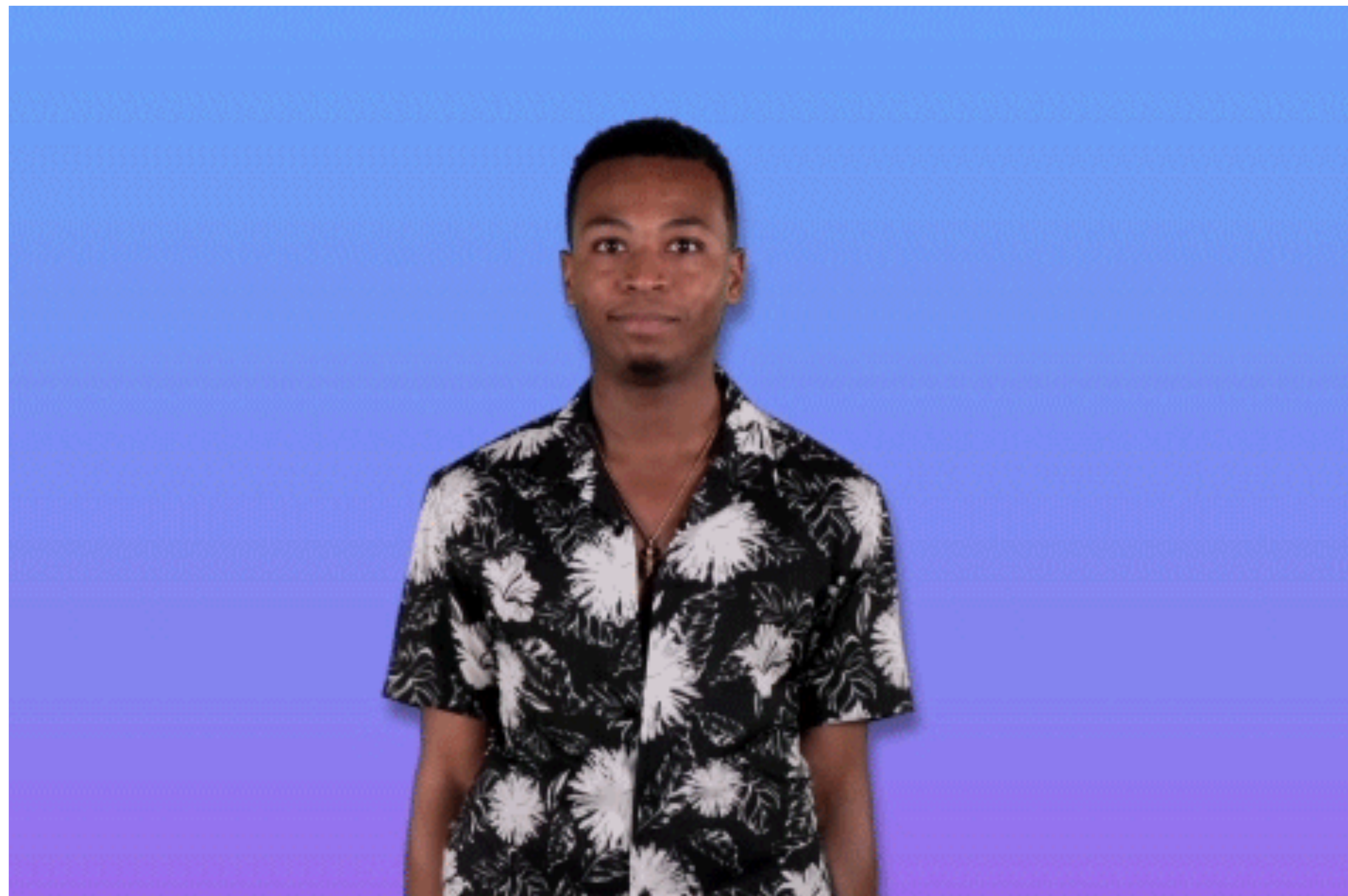
**HOBBIES**

**WHY CODING?**

**FUN FACT ABOUT YOU**



# QUICK REMINDER



# SYLLABUS

- **Week 1: Git**
- **Week 2: Sass / Language Preprocessing**
- **Week 3: Big Images, SVG, Image Optimization**
- **Week 4: Advanced Backgrounds, SVG, Transitions**
- **Week 5: Accessibility**
- **Week 6: Loading Performance and SEO**
- **Week 7: CSS Grid**
- **Week 8: CSS Grid**
- **Week 9: Polish Projects**
- **Week 10: Polish Projects**



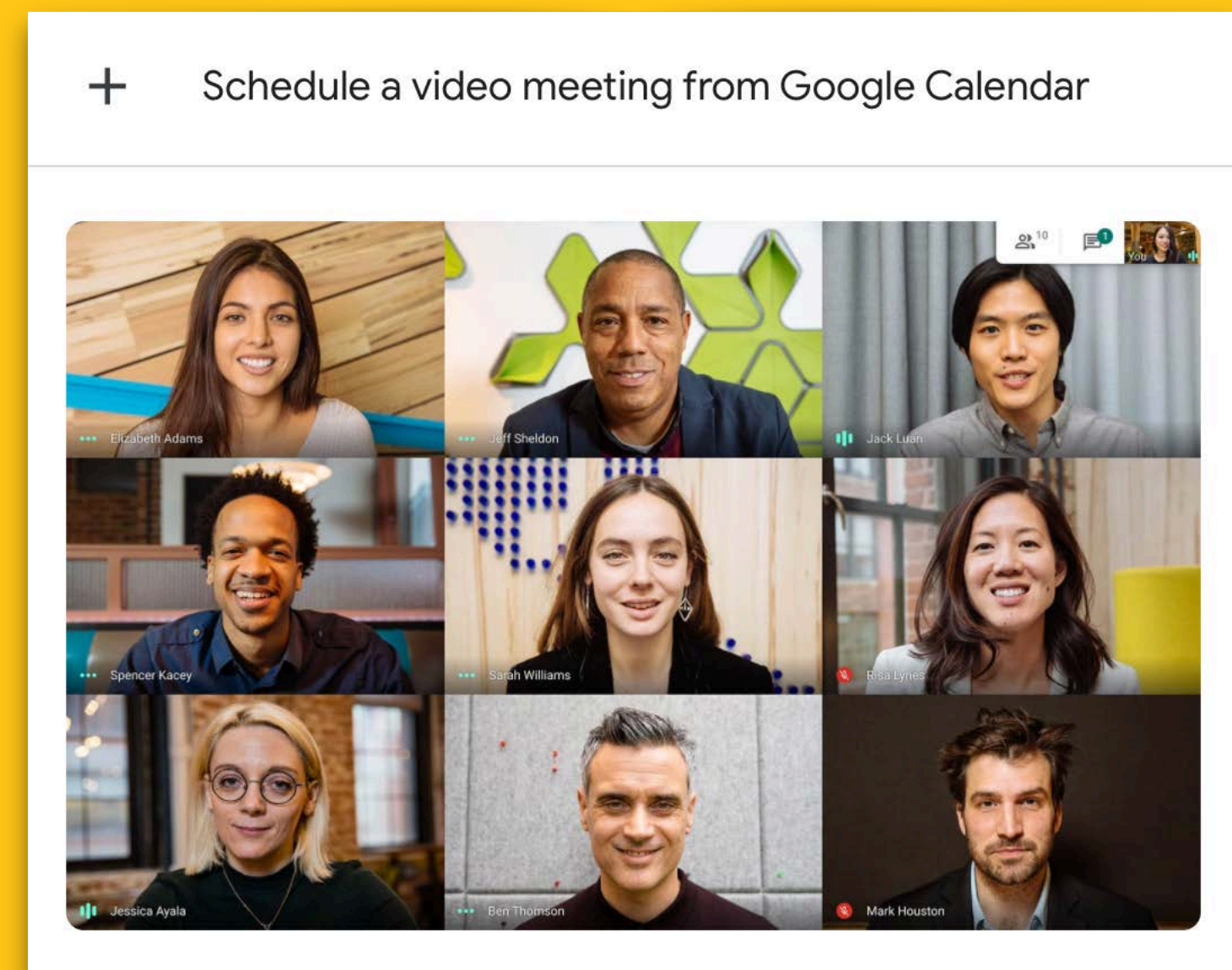
# HOW TO STAY IN TOUCH

- Slack: **#webdev3-winter-2021** & **@Shelly Graham**
- Google Meet: <https://meet.google.com/ikh-ijko-ebt>
- Google Classroom Class Code: **styum5f**
- E-Mail: **michellegraham90@yahoo.com**

## #webdev3\_winter\_2021

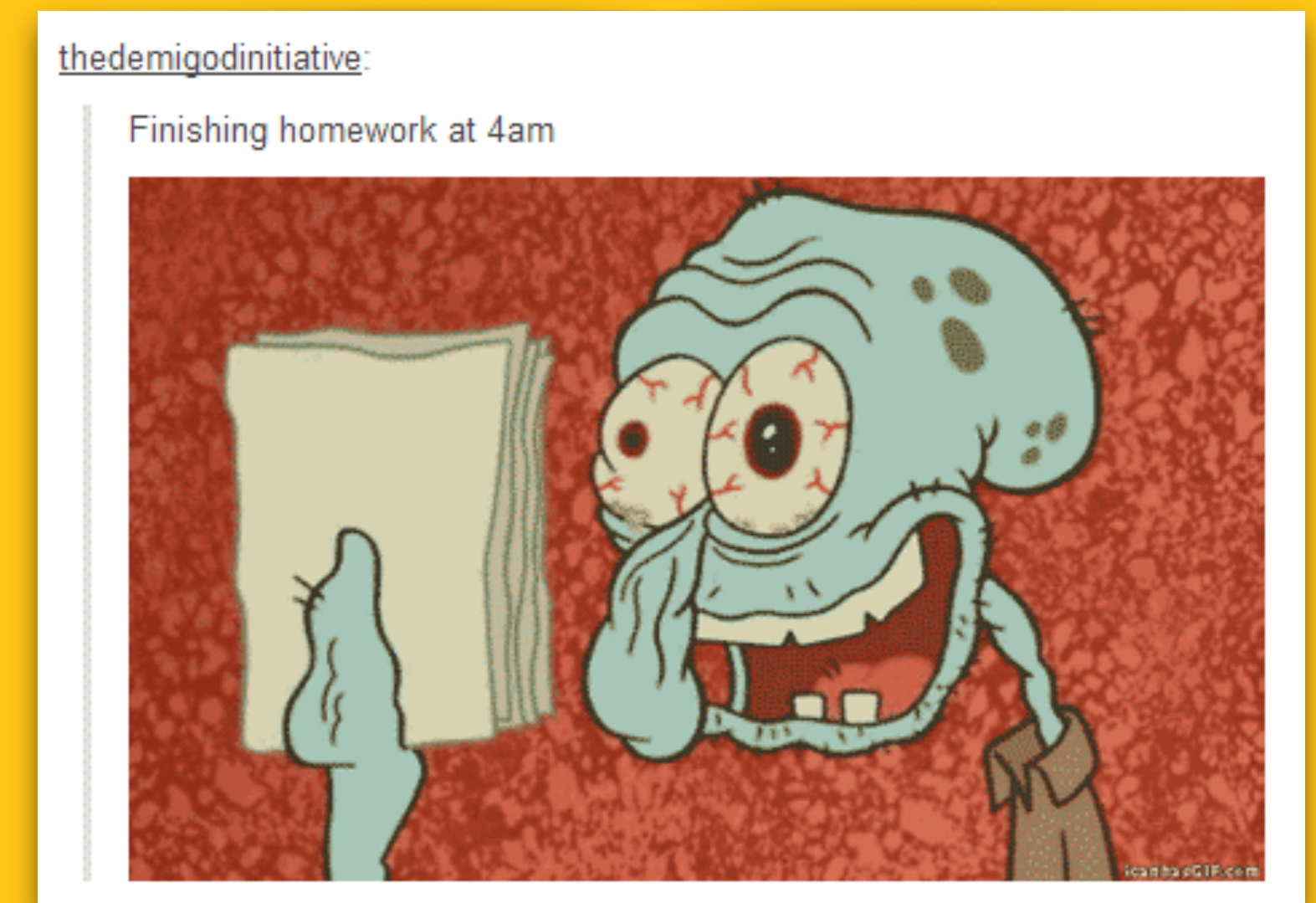
You created this channel on January 3rd. This is the very beginning of the #webdev3\_winter\_2021 channel. Description: Web Dev 3 Class for the Winter 2021 semester @TheCreativeCircus. ([edit](#))

[Add an app](#) [Add people](#) [Send emails to channel](#)



# IMPORTANT THINGS - HOMEWORK

- Homework is due one minute before class starts
- Upload homework to GitHub, post link in Google Classroom Assignment
- You should have code to look at every class
- If I can't see results, I at least want to see progress





# IMPORTANT THINGS - GRADES

## Your Class Grade

A: **Exceeded Expectations.** This is the rare student who consistently surprises you and pulls the class up.

B: **Met Expectations.** Did all of the assignments. Most were good.

C: **Approached Expectations.** Maybe this student completed every project, but none were surprising or smart. Or they really only committed to doing a third of the assignments well. NOT good enough for a B.

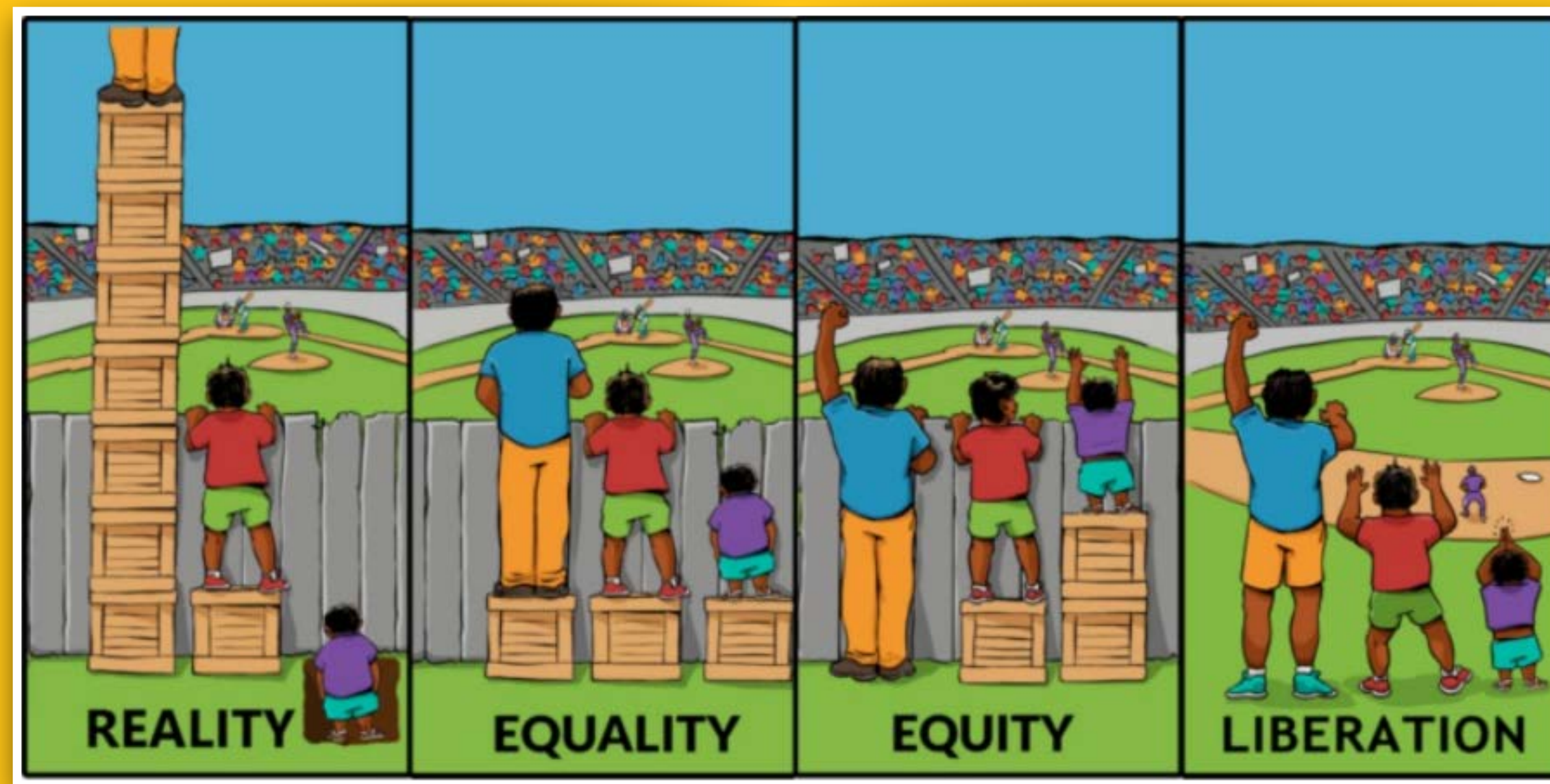
D: **Needs attention; not competitive.** Either trying and failing, or failing to try.

F: **Failed** to show talent, drive or interest. Needs to reconsider coming back next quarter.



# IMPORTANT THINGS - INTERACTION

- **Inclusivity:** Think outside of your own personal realm
- **Equality:** Help one another figure it out
- **Equity:** You might need more or less help to thrive - both is OK!



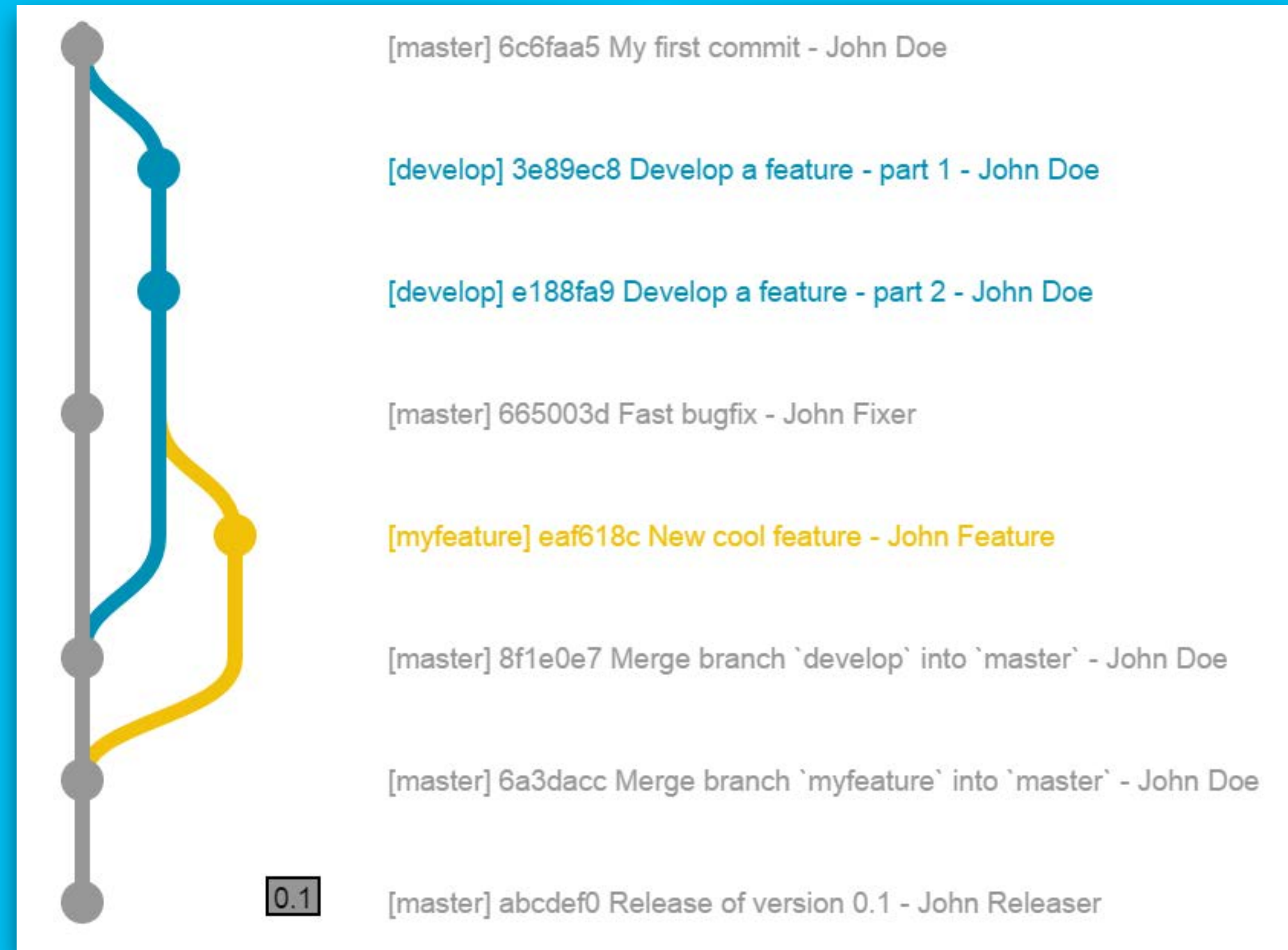
# IMPORTANT THINGS - TIME MANAGEMENT

- If you can't finish a project, it will reflect in your project grade
- This is school but treat it as your job!
- If I can do it, so can you!



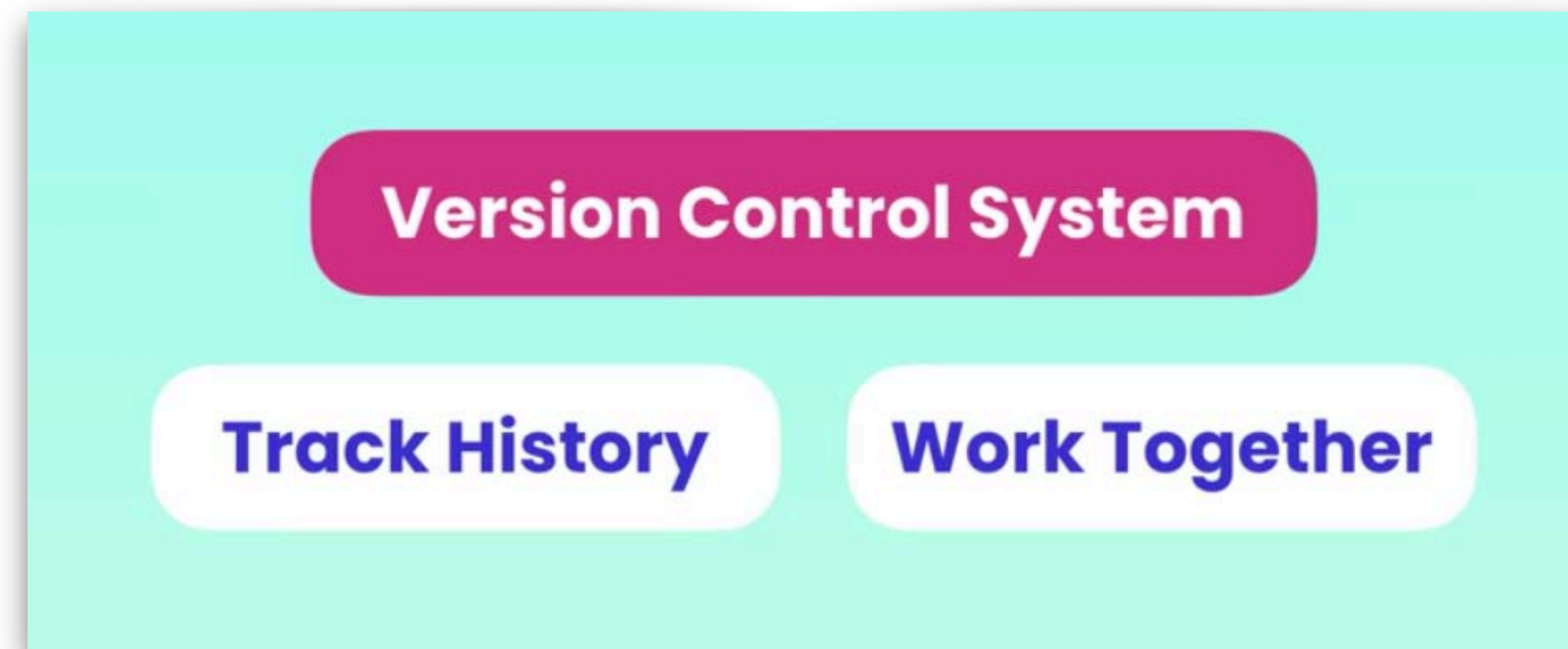


# WEEK 1: GIT



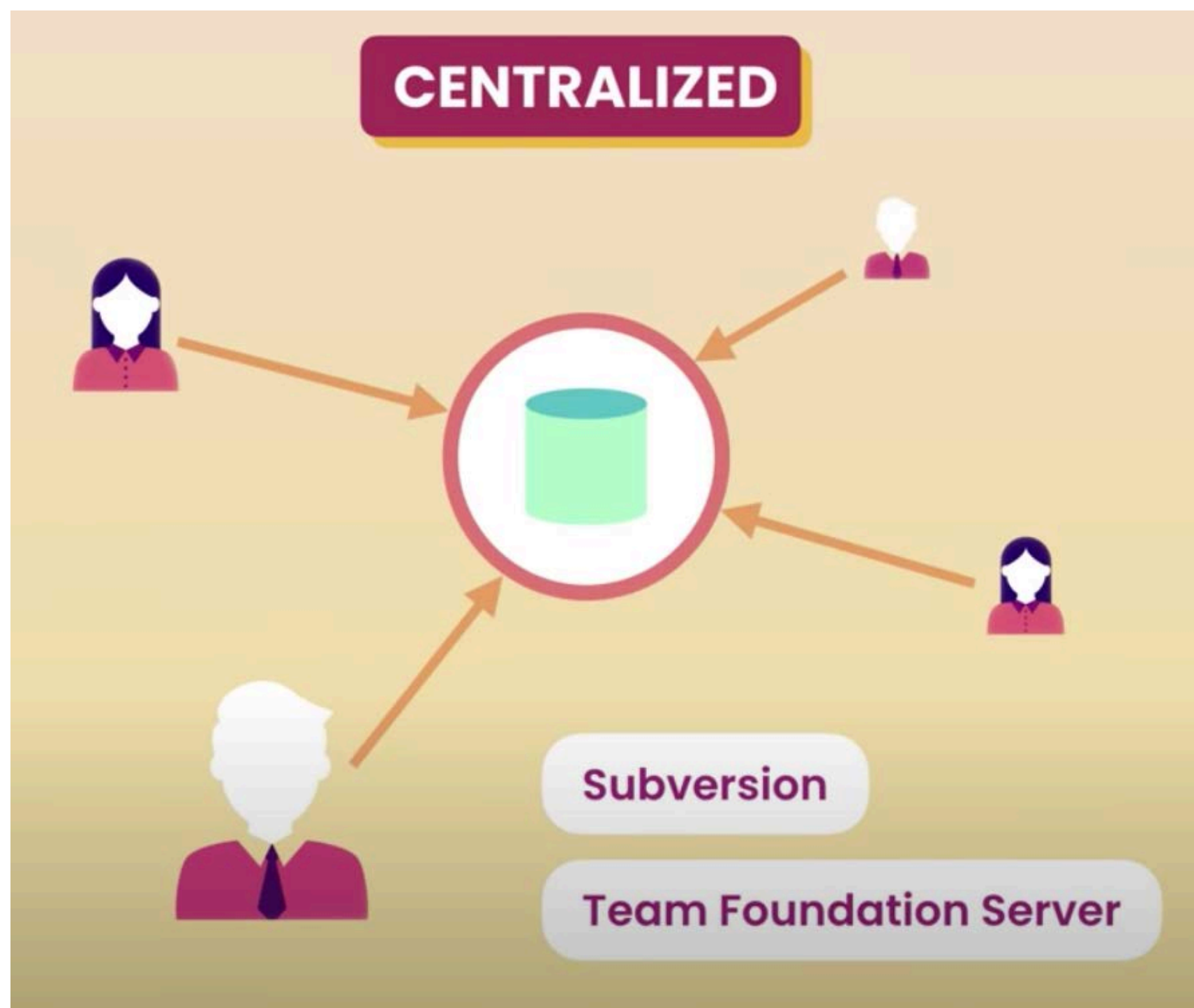
# WHAT IS GIT?

- Most popular version control system in the world - as of today
- Free, Open Source, Fast, Scalable
- It records the changes made to code over time in a special databank aka repository
- Makes it easy to collaborate with people on one project
- Works well on a wide range of operating systems and IDEs (Integrated Development Environments)

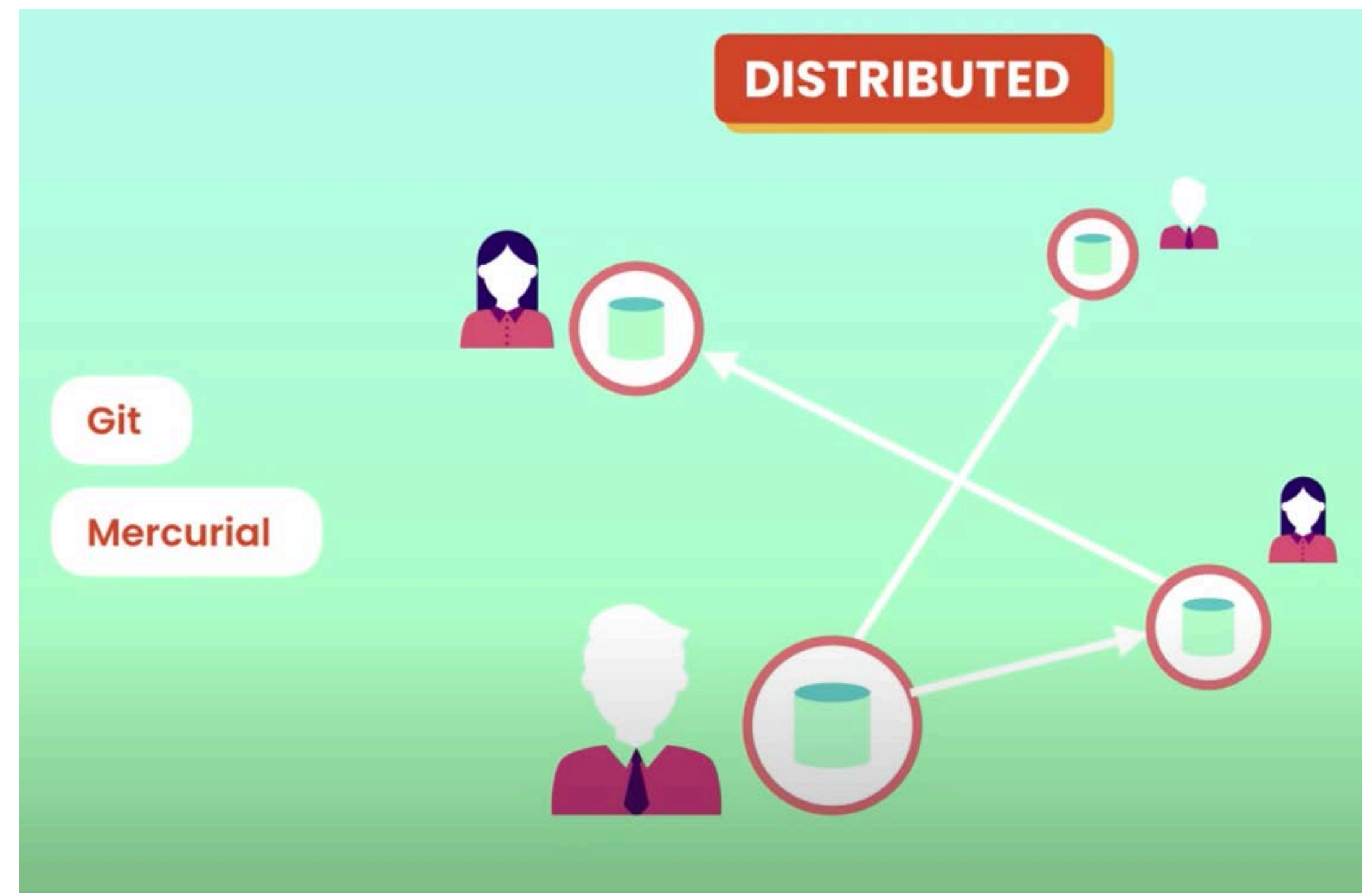




# TWO DIFFERENT VERSION CONTROL SYSTEMS



VS



# OTHER VERSION CONTROL SYSTEMS AKA CLIENTS & HOSTS

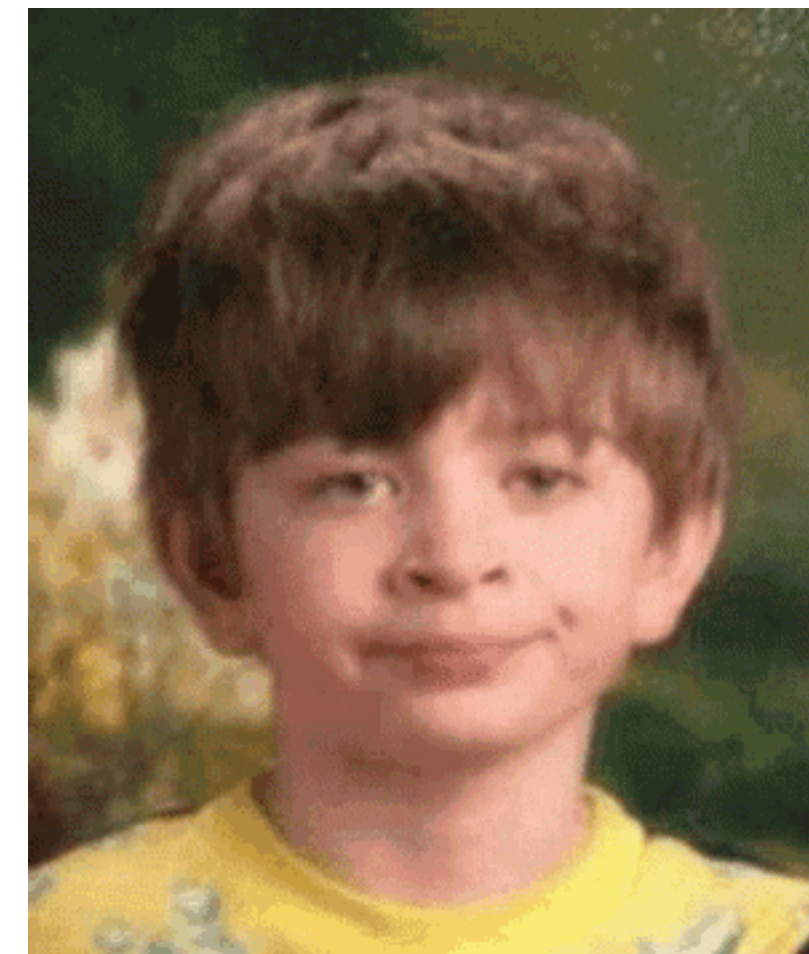
Clients are applications to visualize all those commands. We will use GitHub Desktop or simply the command line.

Hosts are places to store all those commands, i.e. your code. We will use [GitHub.com](https://github.com)

Client and host don't have to be the same, they can differ!

- GitLab
- Gogs
- BitBucket
- GitBucket
- SourceForge
- Beanstalk
- Fabricator...

So. Many.



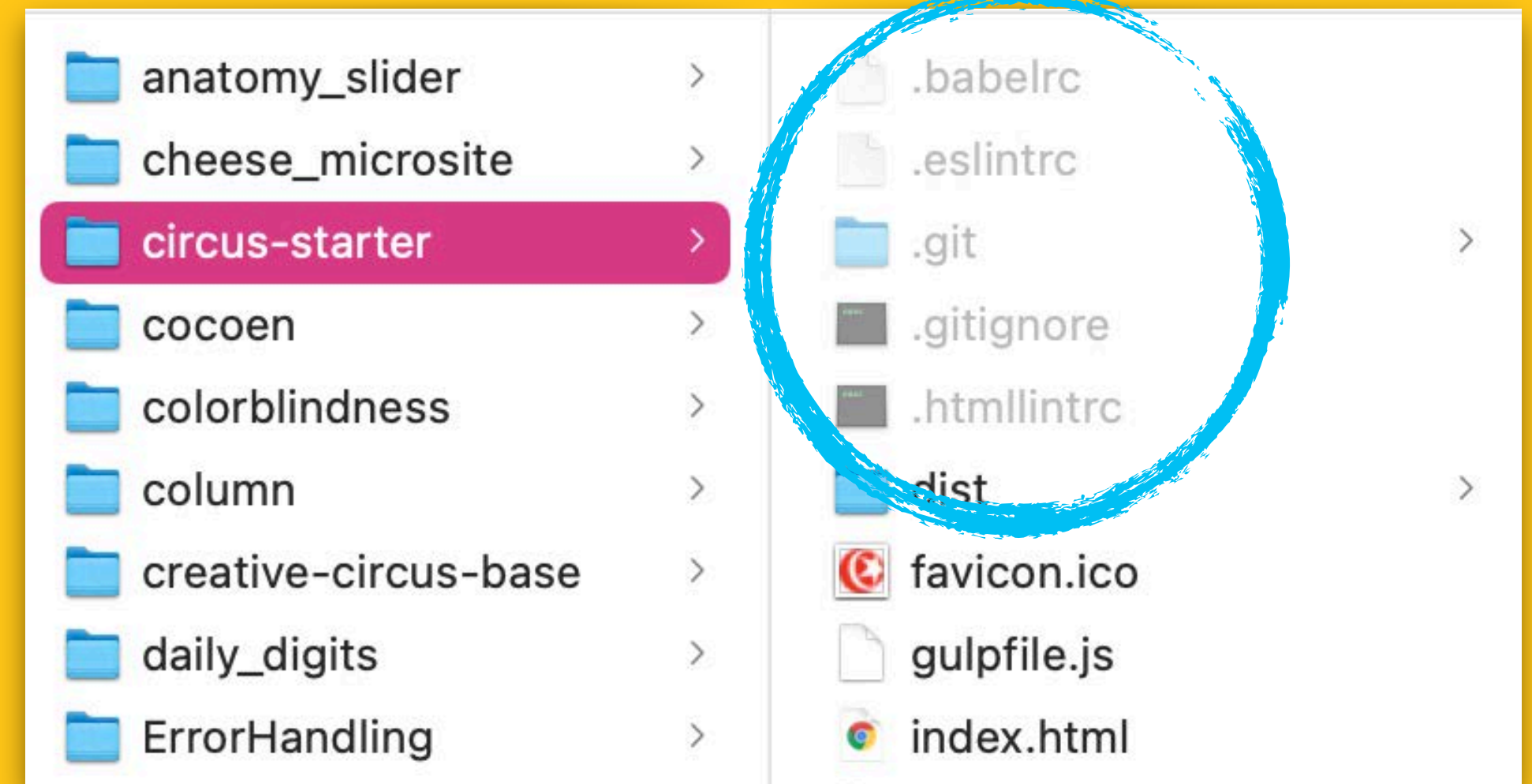
Source



**GIT. FOR. EVERY. PROJECT.**

# DID YOU KNOW ABOUT **HIDDEN FILES**?

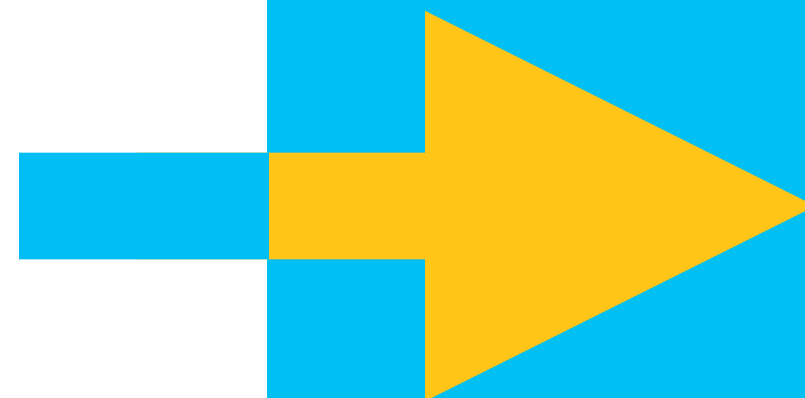
- Files you usually don't see
- Recognize them with a “.” dot in front of files
- Show hidden files on Mac: **CMD + SHIFT + .**
- Show hidden files on Windows: In File Explorer, click the “View” tab and click the “Hidden Items” checkbox





# 1. HOW TO START A LOCAL GIT REPO

1. Open Terminal
2. Type “cd” and drag your project folder into Terminal
3. Type “git init”
4. Boom. Now you have a local Git repo!



From here, you can use Terminal commands like:

- git status
- git add .
- git commit -m “commit message”
- git branch -av (all verbose)
- git branch “branch name”
- git checkout “branch name”
- git push —> publish local changes remote
- git fetch —> download remote changes
- git pull —> download & integrate remote changes

## 2. IMPORTANT FOLDERS

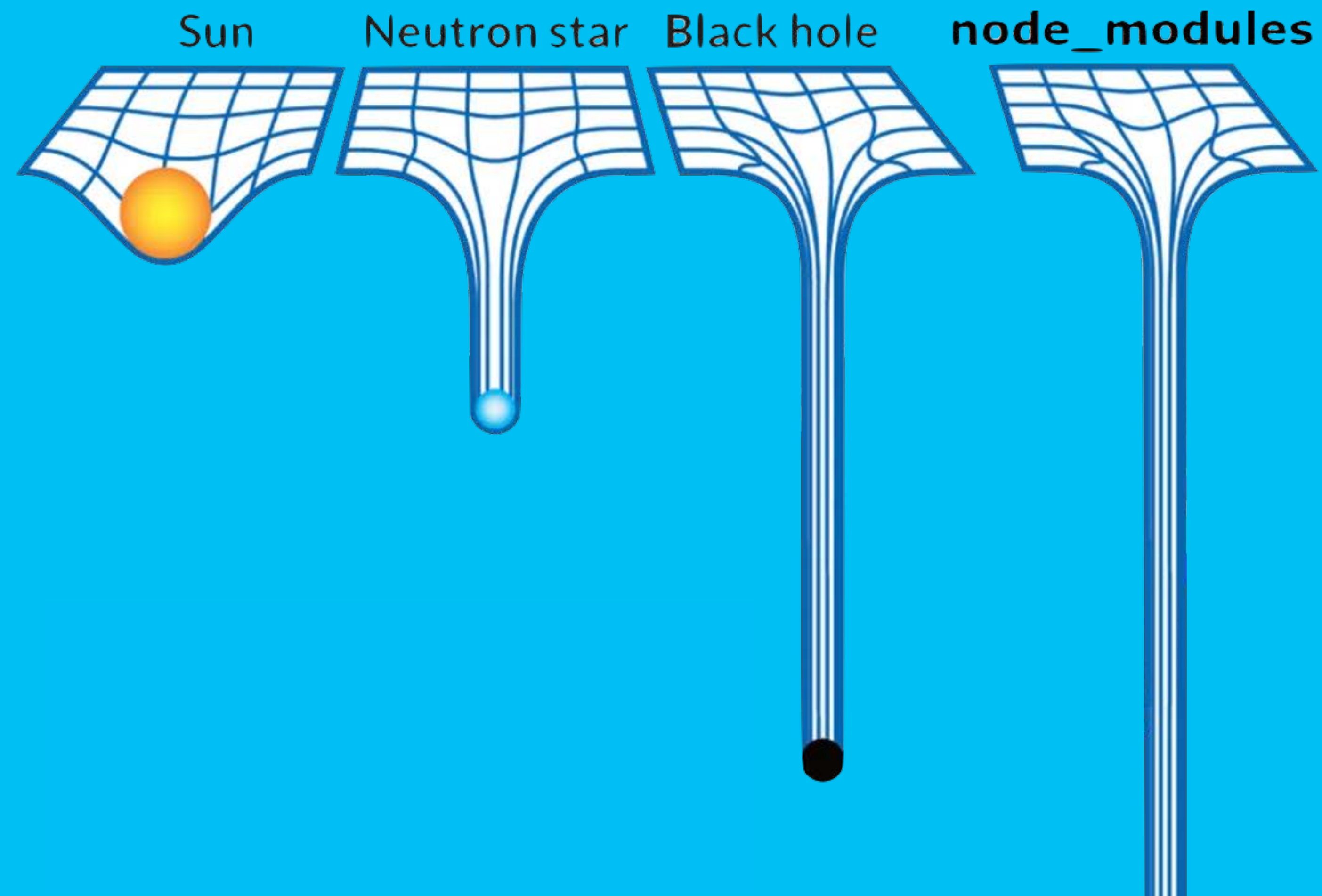
### .GIT FOLDER

- Your actual git repo
- Includes config files and project history

### .GITIGNORE FOLDER

- Prevents certain files and folders from being added to repo (e.g. node modules)
- **NEVER COMMIT NODE MODULES!!!**

## Heaviest Objects In the Universe:

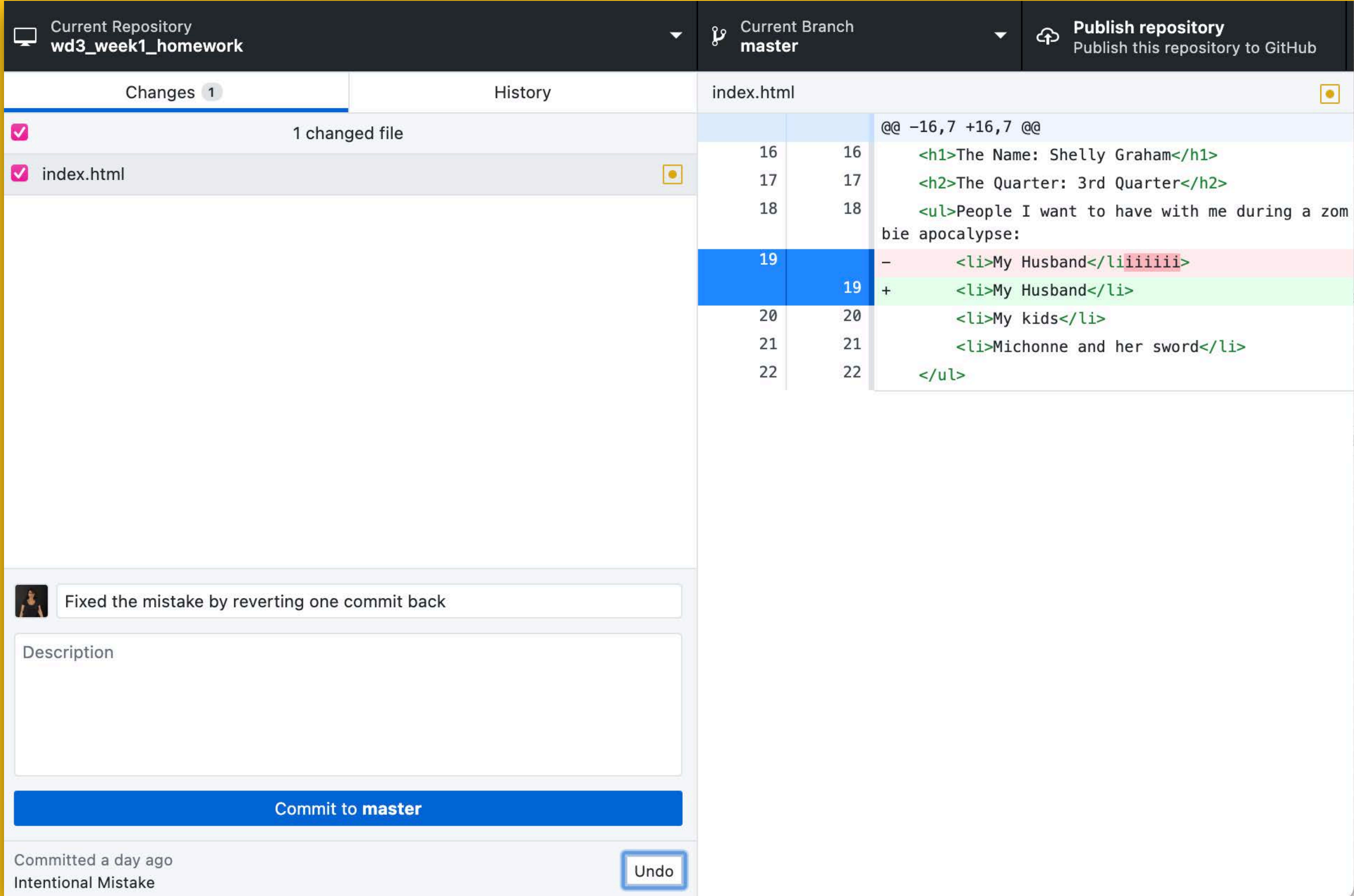




# Terminal command structure

command	option	option	argument
git	commit	-m	"initial commit"

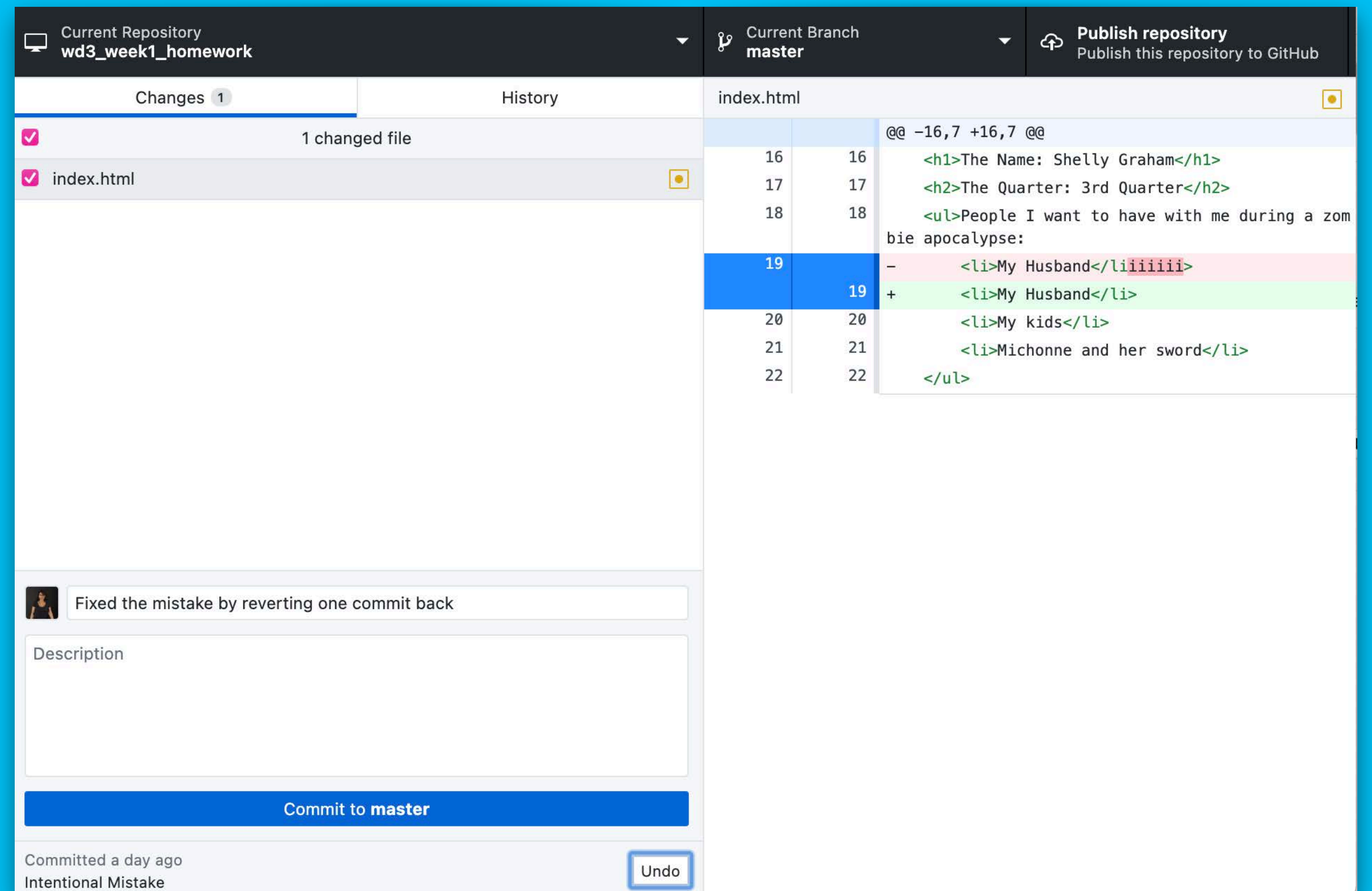
# OR SIMPLY USE GITHUB DESKTOP





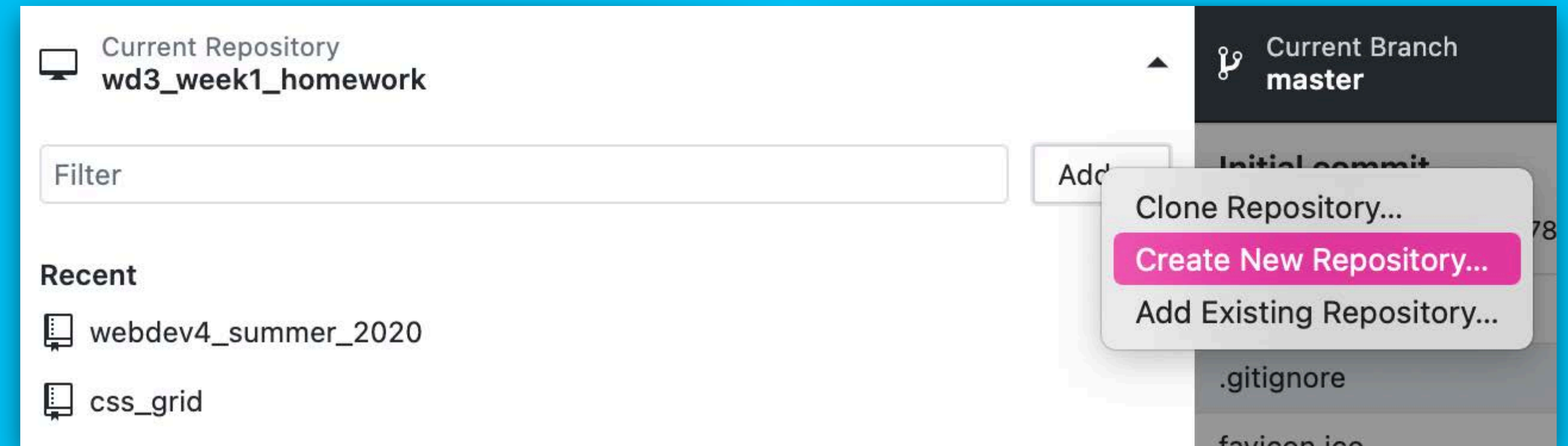
# 3. HOW TO UPLOAD YOUR REPO TO GITHUB DESKTOP

1. Drag project folder into GitHub Desktop
2. Hit “Publish repository”
3. Boom. Now you have a remote Git repo!



# 3.1. HOW TO START A REPO FROM SCRATCH

1. Create a folder in your Finder
2. Drag folder into GitHub Desktop
3. Click "Create repository"
4. Add files, Publish
5. That's it. Start adding project files.



Create a New Repository

Name

test project

⚠

Will be created as test-project

Description

This is a test project

Local Path

/Users/shellyjames90/Documents/GitHub

Choose...

☒ Initialize this repository with a README

Git Ignore

None

License

None

Cancel

Create Repository



# THE BASICS

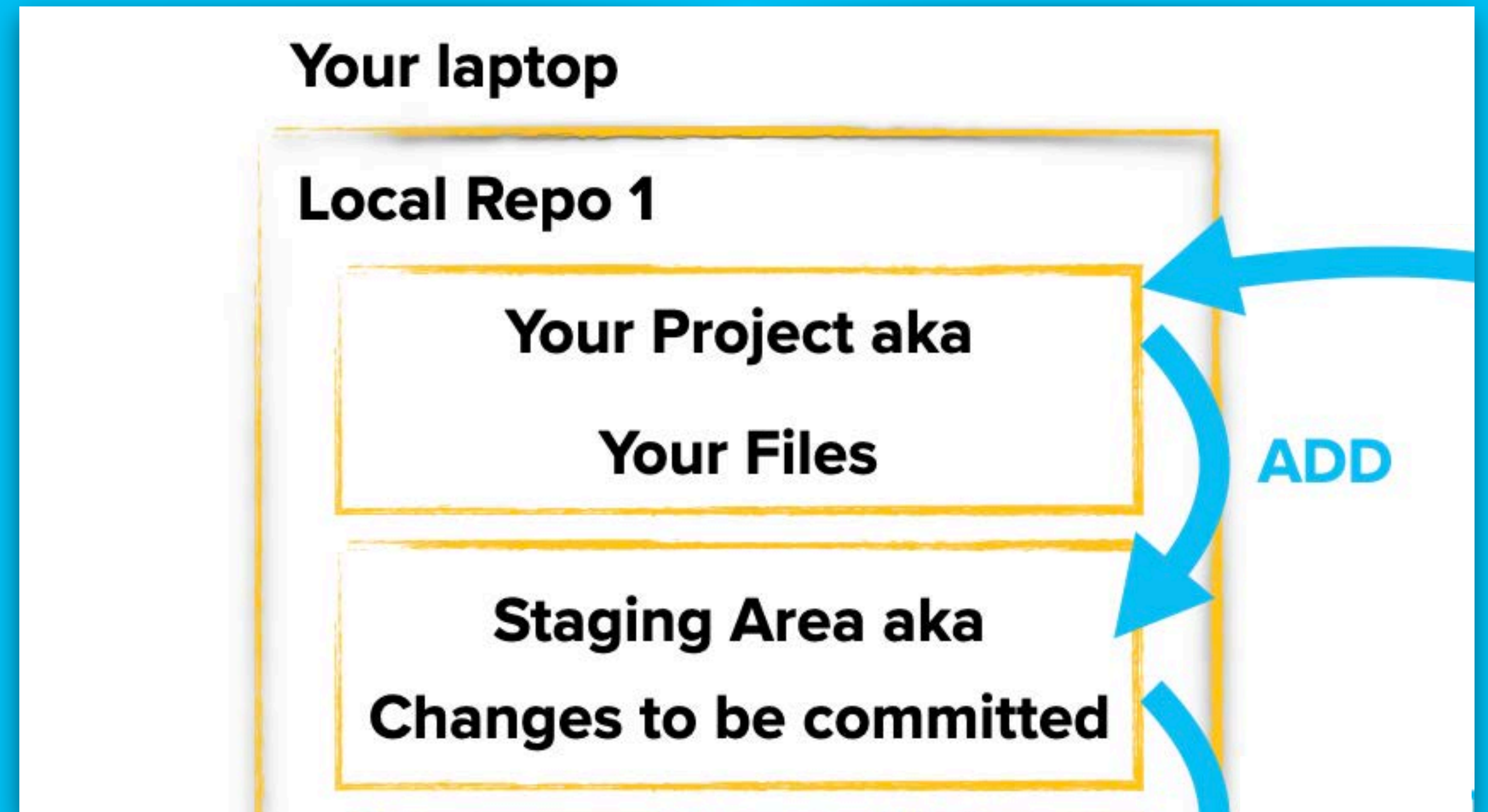
1. Add
2. Commit
3. Pull
4. Push
5. Clone

6. Branching
7. Merging
8. Fork

Not so basic

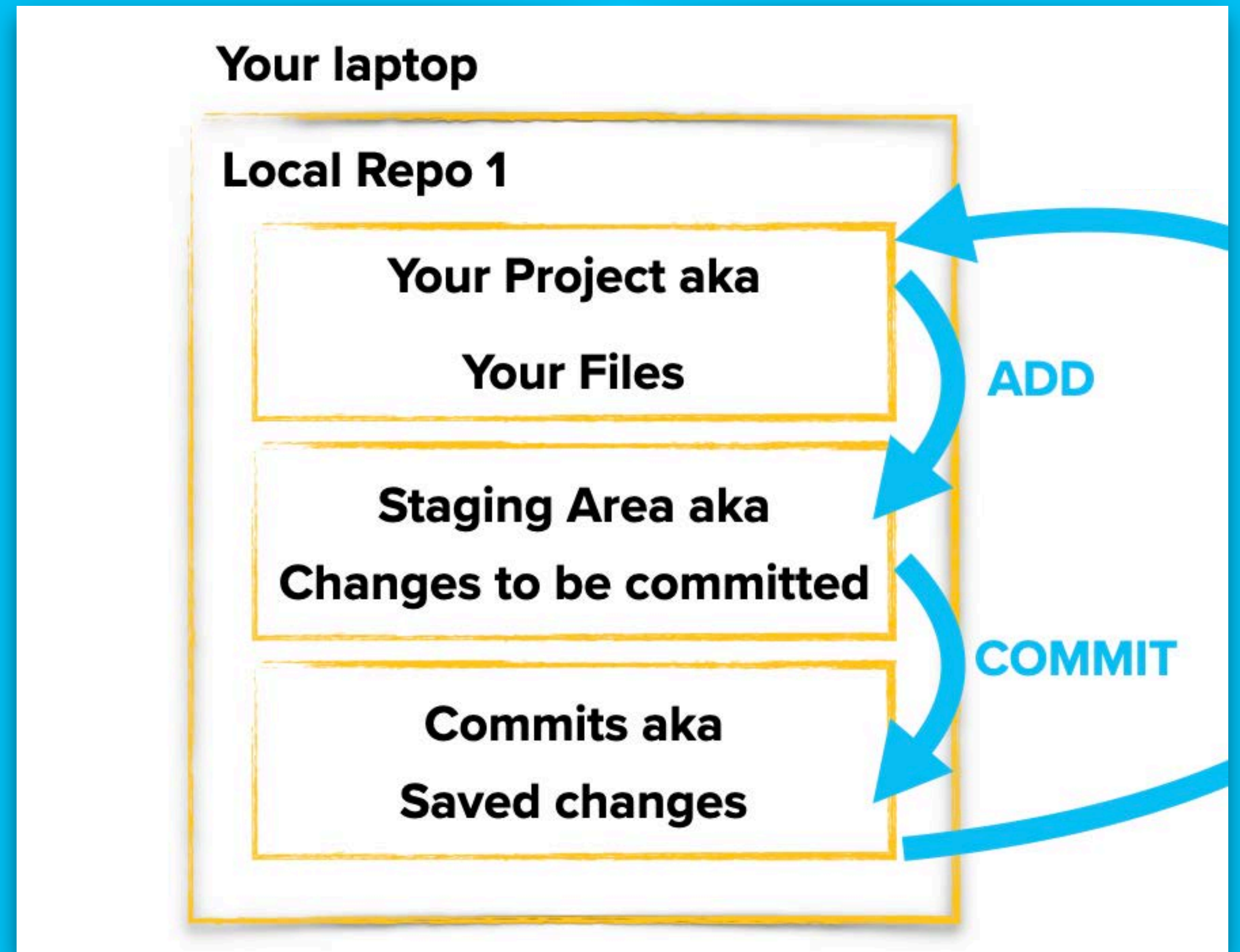
# 1. ADD

- ADD a particular set of files and/or changes to your local repo
- Type:  
`git add [files to add] or  
git add .`
- Boom. You've added files and changes to your repo! This is also called the staging area!



## 2. COMMIT

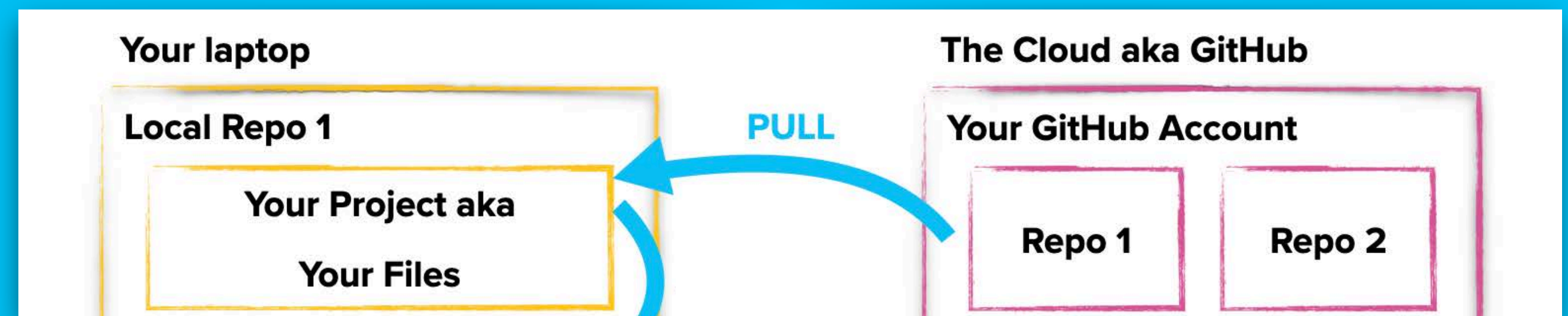
- COMMIT a particular set of changes in your local repo.
- Commit early, commit often. Commit before the end of the day. There is no such thing as too many commits.
- Does not affect your remote repo yet!
- Type:  
`git commit -m "Your commit"`





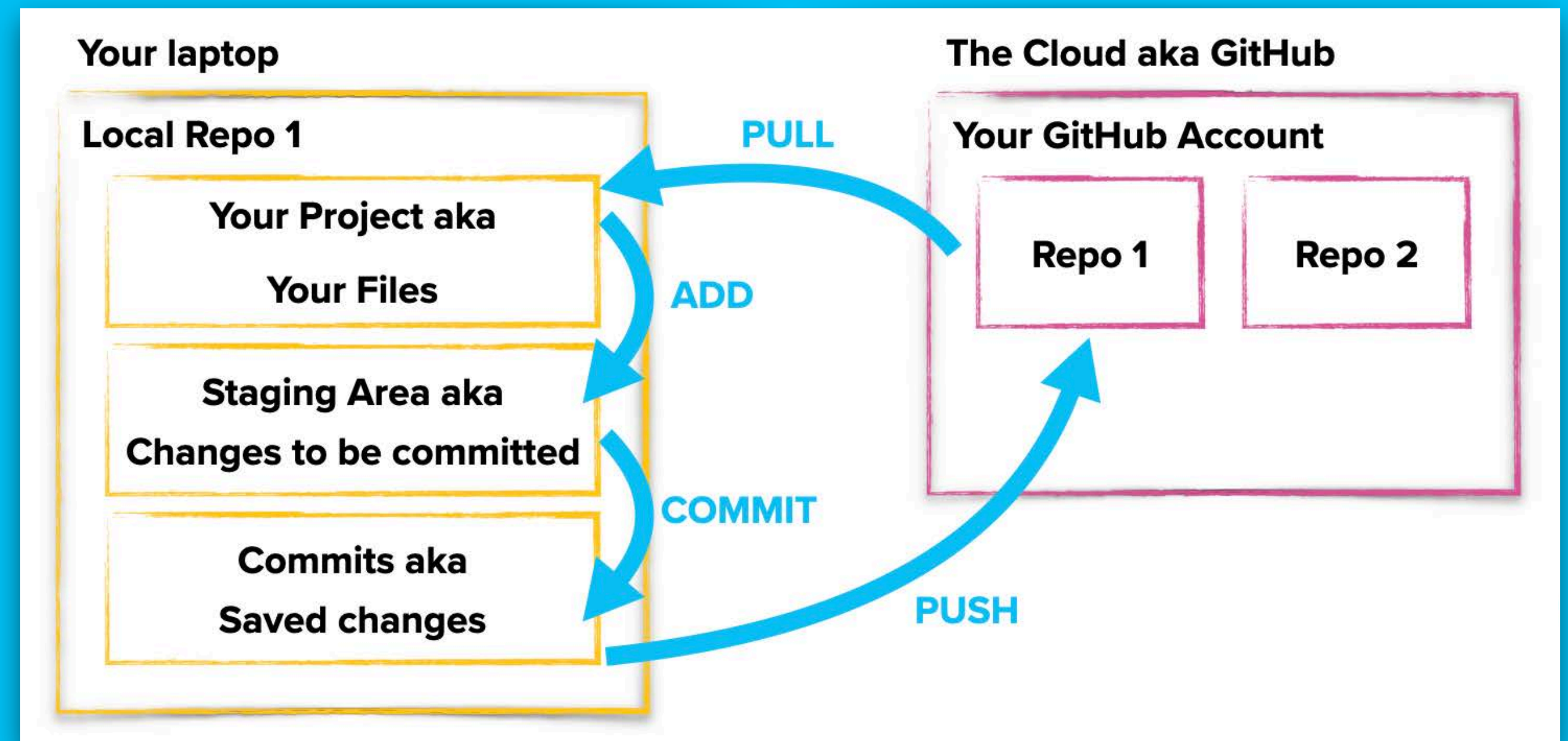
# 3. PULL

- **PULL** any new commits that the remote server may have down to your local repo
- Always pull before pushing your own commits!
- Type:  
**git pull**
- Boom. Now you have the latest remote version of a repo on your local machine!



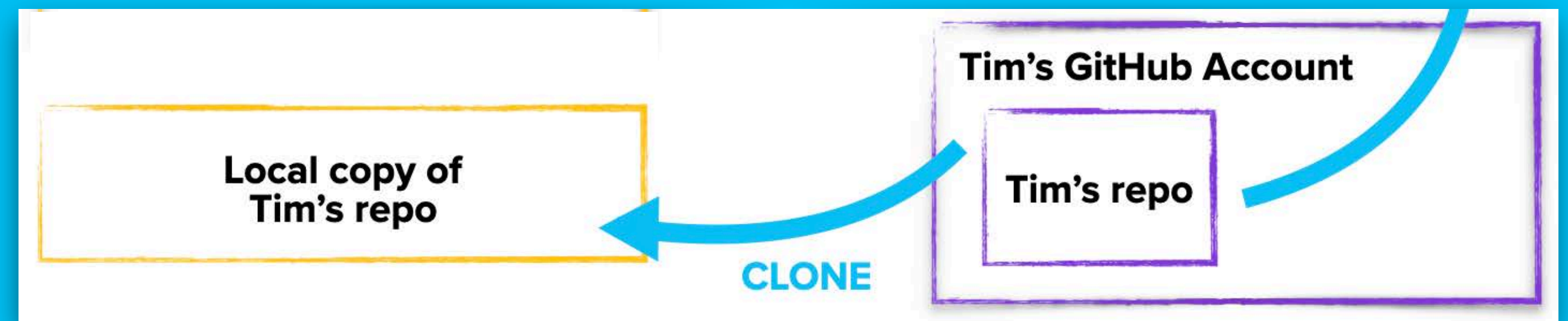
# 4. PUSH

- PUSH the local commits you made to the remote server
- Type:  
`git push origin master`
- Boom. Now you updated the remote repo with your latest local version



# 5. CLONE

- Download a **LOCAL** copy of a git repo from a **REMOTE** server aka another user, including all history.
- In Terminal, cd into path where you want to clone the project
- Type:  
`git clone https://github.com/CyanLetter/WebDev3_Spring2018`
- Boom. Now you have a local copy of a remote Git repo!





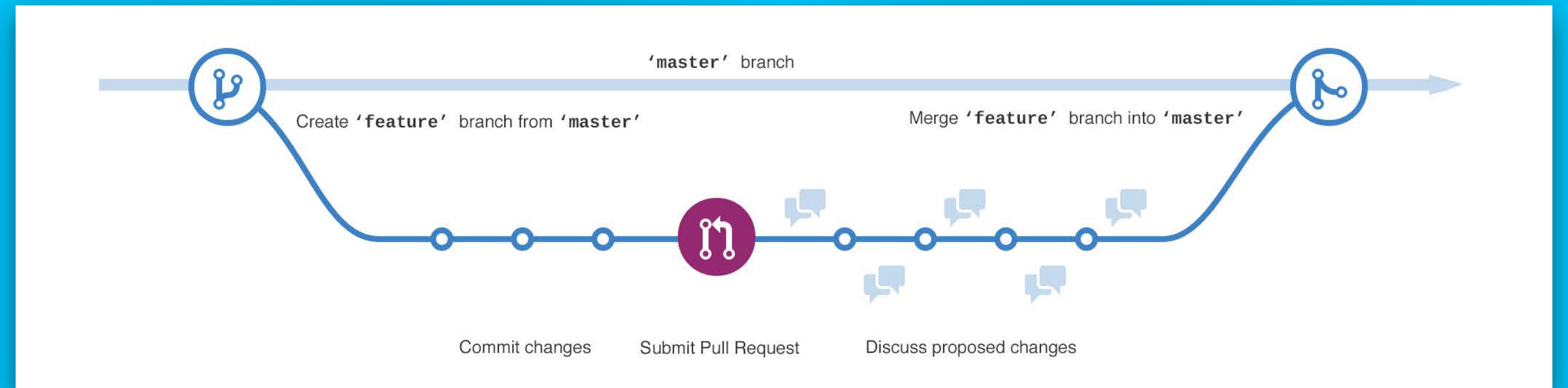
# 6. BRANCHING

- “Making an alternate timeline”
- Good practice for a plugin or new page
- The master branch should never be broken, branches let you work without breaking master
- Great for working in teams, still helpful if it's just you
- Make sure to stay in sync with mbranch via **git pull**

- List all current branches:  
**git branch -av**
- Create a branch:  
**git branch “branch name”**
- Switch to a branch:  
**git checkout “branch name”**
- Delete a branch:  
**git branch -d “branch name”**

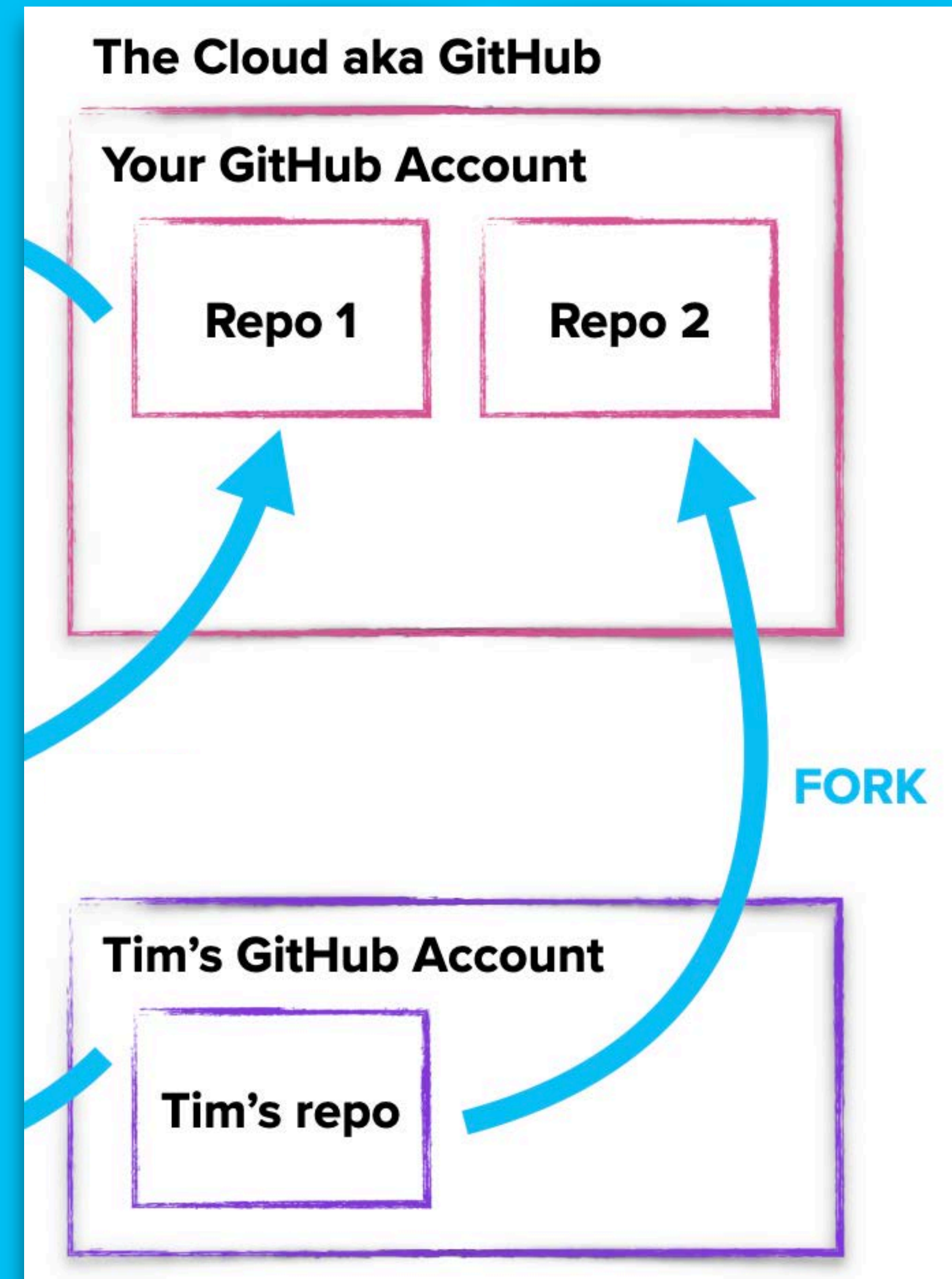
# 7. MERGING

- “Merging the timelines”
- When you're finished with a branch:
  - Switch to the branch you want to merge into (e.g. master) via **git checkout “master”**
  - Pick the branch you want to merge with:  
**git merge “branch name”**
- Resolve conflicts, commit. Boom. You have now merged a branch into your master branch!



# 8. FORK

- Duplicating someone else's repo for yourself to edit
- On GitHub.com, use Fork Button!
- Creates a copy of the project that you now own
- You can also pull updates from the original project
- Boom. Now you have a local copy of a remote Git repo!





## Your laptop

### Local Repo 1

Your Project aka  
Your Files

Staging Area aka  
Changes to be committed

Commits aka  
Saved changes

Local copy of  
Tim's repo

## The Cloud aka GitHub

### Your GitHub Account

Repo 1

Repo 2

### Tim's GitHub Account

Tim's repo

PULL

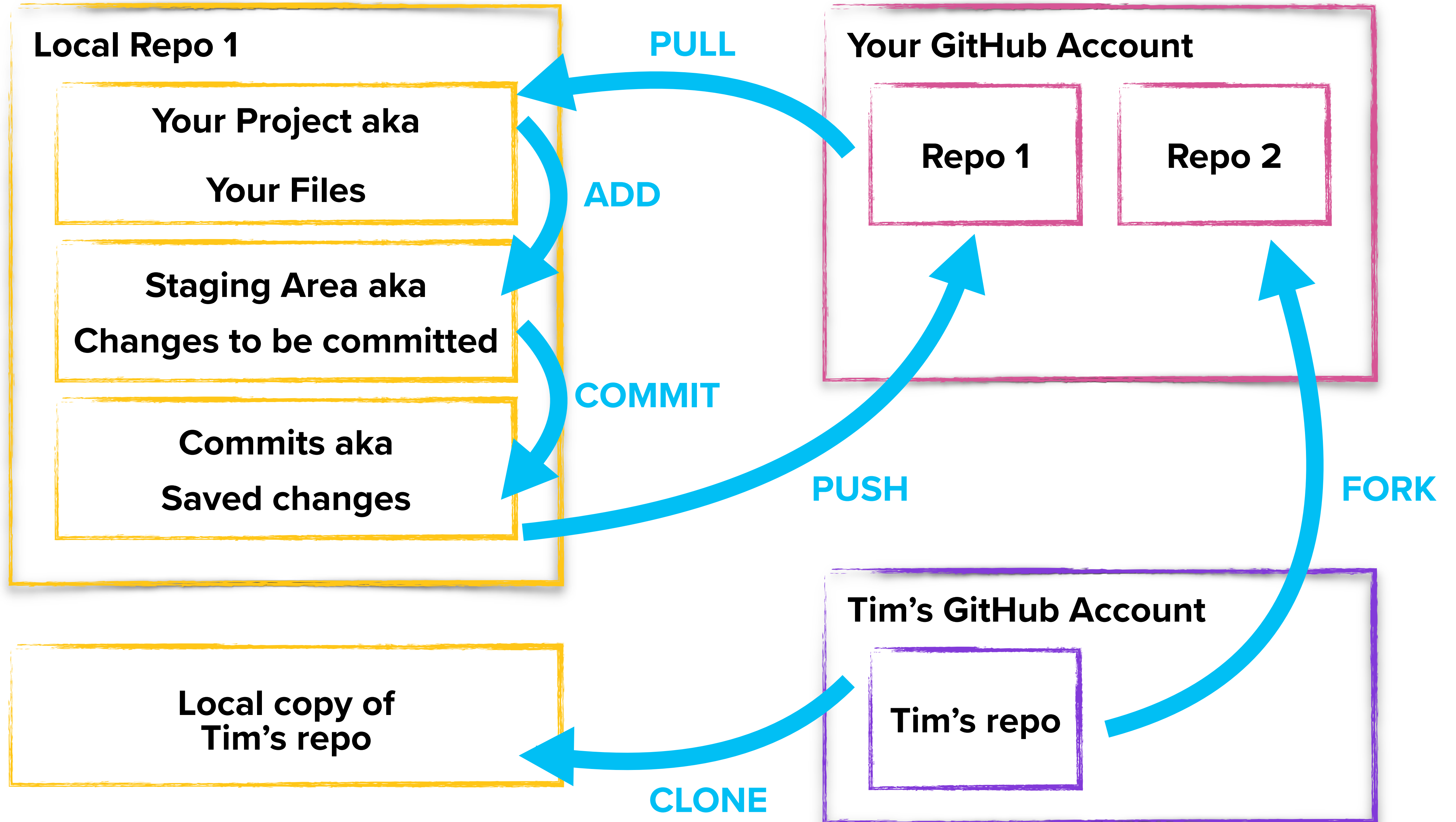
ADD

COMMIT

PUSH

FORK

CLONE




# REVERTING




- Restoring files to a previous version
- Uncommitted changes: In GitHub Desktop, right click on the file you changed and select “Discard Changes”
- Going back to previous commits:
  - In the history tab, right click the commit you want to undo and select 'Copy SHA'
  - In Terminal, type: `git revert --no-commit <SHA Number>..HEAD`
  - Make sure to include the “..HEAD” at the end of the SHA number
  - Lots of dangerous ways to do this that rewrite history. Command line is the easiest and safest


# MERGE CONFLICTS

- Oh no! Two people changed the same thing! How do you fix it?
- You can resolve manually or use merge tools






**Changes approved**[Show all reviewers](#)

2 approved reviews. [Learn more.](#)

**All checks have failed**[Hide all checks](#)

1 failing check

 **ci/circleci** — Your tests failed on CircleCI [Details](#)

**This branch has conflicts that must be resolved**

Only those with [write access](#) to this repository can merge pull requests.

**Conflicting files**  
Changelog.md  
Demo/Tests/EndpointSpec.swift



# RESOLVE BY HAND

- Things on top are yours, things on bottom are theirs.
- Delete what you don't want to keep.

```
<<<<<<< HEAD
Stuff from your laptop (LOCAL)
=====
Stuff from the server (REMOTE)
>>>>>>> origin/master
```

# RESOLVE WITH MERGE TOOLS

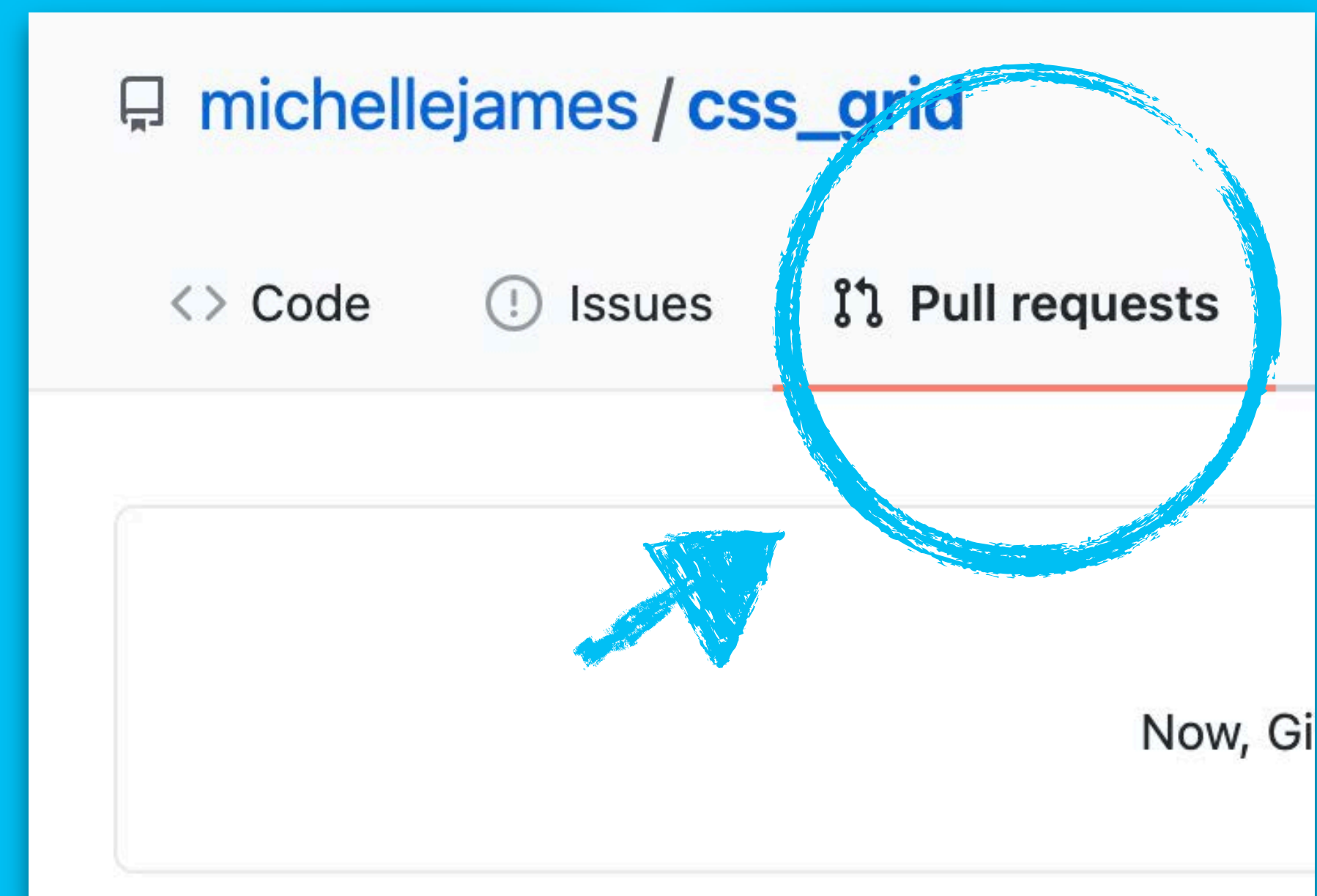
- **SourceTree**
- **VSCode**
- **Sublime Merge**
- **Meld**
- **kDiff**

# 9. PULL REQUEST

- A request to merge your changes back into the original repo
- It's best to do this from a branch
- This is how people fix bugs and add features in open source projects!

## How to do a pull request:

- Make sure you're in the right branch
- Hit the Pull Request button
- Add a description and send it
- Owner can review and accept or decline





**QUESTIONS?**

# HOMEWORK - PART 1

- Make a new folder called “wd3\_week1\_homework”, and add a copy of the circus starter to it. Create a git repository with that project and do the following:
  - Commit the default circus starter project with the message “Initial Commit”
  - Delete the contents of README.md and replace it with “<your name> - WD3 - Git Homework”
  - Use markdown syntax to make your name bold in the readme and the “WD3 - Git Homework” part cursive
  - Delete the body of the HTML file and replace it with basic markup that includes your name, your quarter, and a list of three things that this pandemic has taught you
  - Commit this to the master branch with the message "Second commit"
- Next, demonstrate the use of branches:
  - Make a new branch called “styling”
  - Add some color and font settings to make your HTML page look nicer. Like you would want an “About Me” page on your portfolio to look
  - Commit your changes to the “styling” branch as you make them. Make sure your commit messages describe what you've done. I want to see at least two commits
  - Merge those changes back into the main branch

# HOMework P - PART 2

- Next, demonstrate how to revert undesired changes:
  - Switch back to your master branch
  - Add a horrible mistake to your HTML file
  - Commit that mistake and push it to [GitHub.com](https://github.com)
  - Then, revert to the previous commit, without rewriting history
  - Commit the reverted changes
- Finally, create a pull request in your classmates' repository:
  - Post a link to your repo in the Slack channel
  - Create a fork of your classmates' repositories
  - Add to their HTML file. Say something nice, encouraging or funny
  - Submit a pull request for your change
  - Accept the pull requests that come in for your own project
- Submit a link to your repo via Google Classroom. I should be able to look through your commit history and see these exact steps carried out.



**FIN**