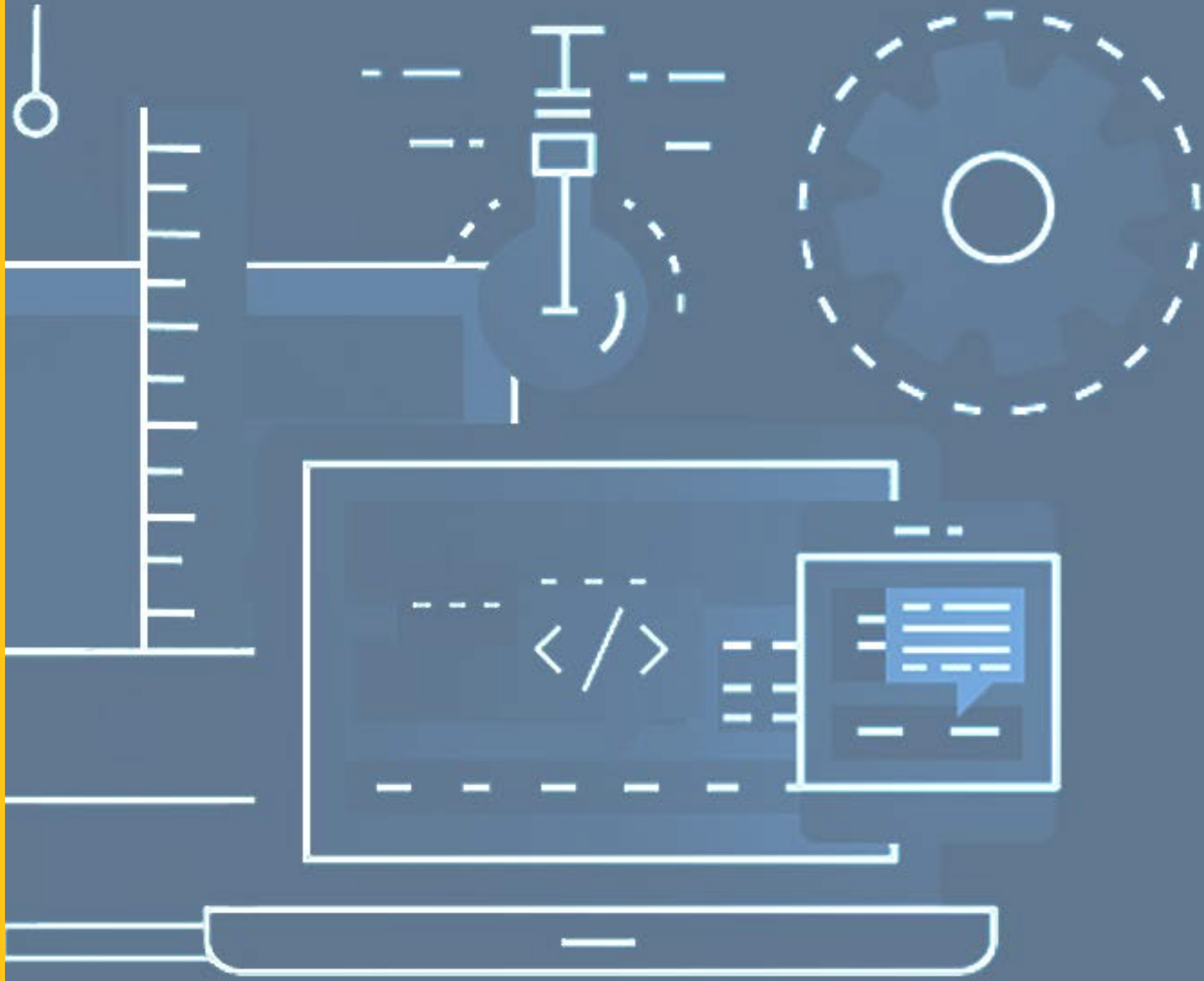
SHELLY GRAHAM, 01/07/2021

WEB DEV 3 WINTER 2021

Week 1: Git



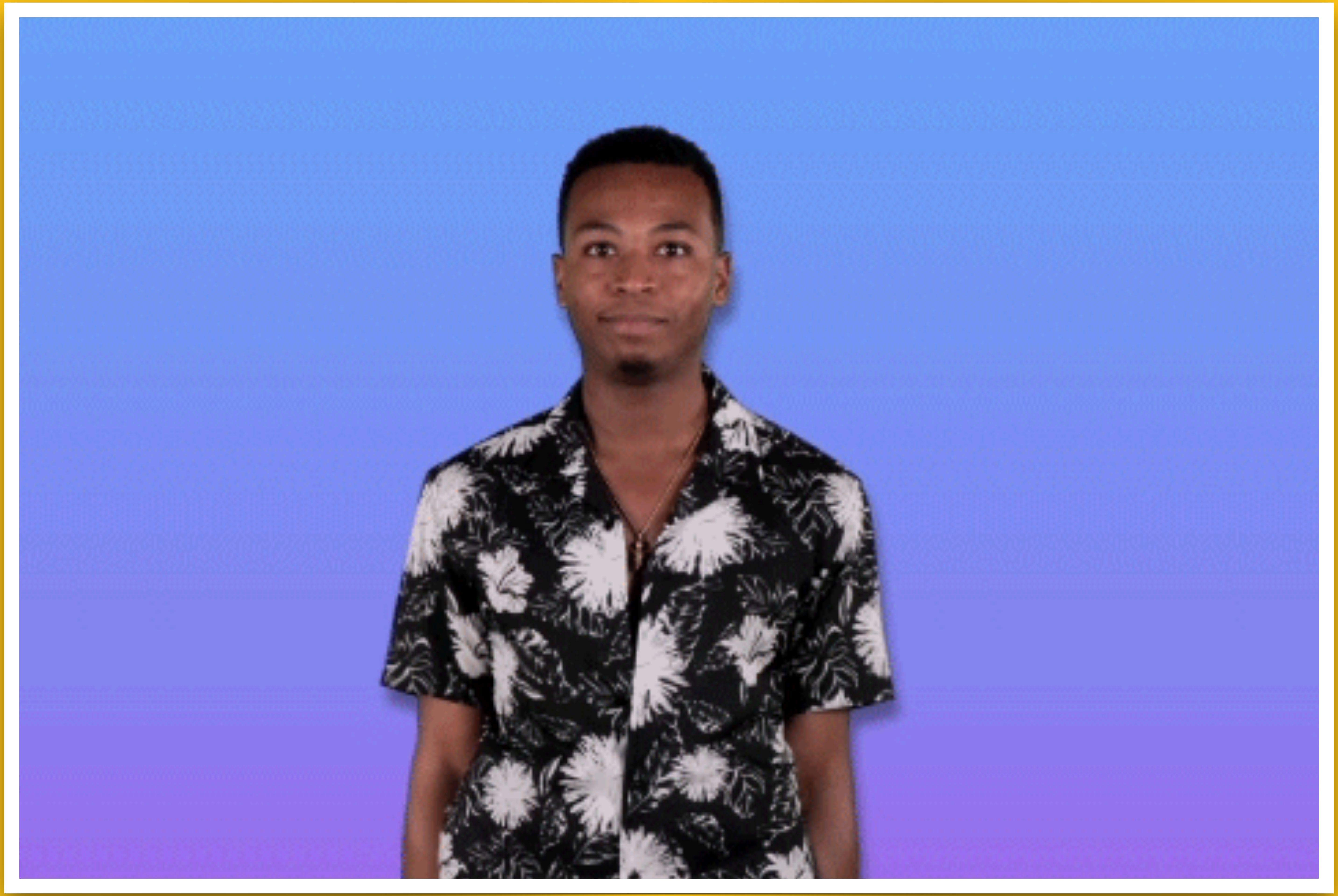
THAT'S ME!

- 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



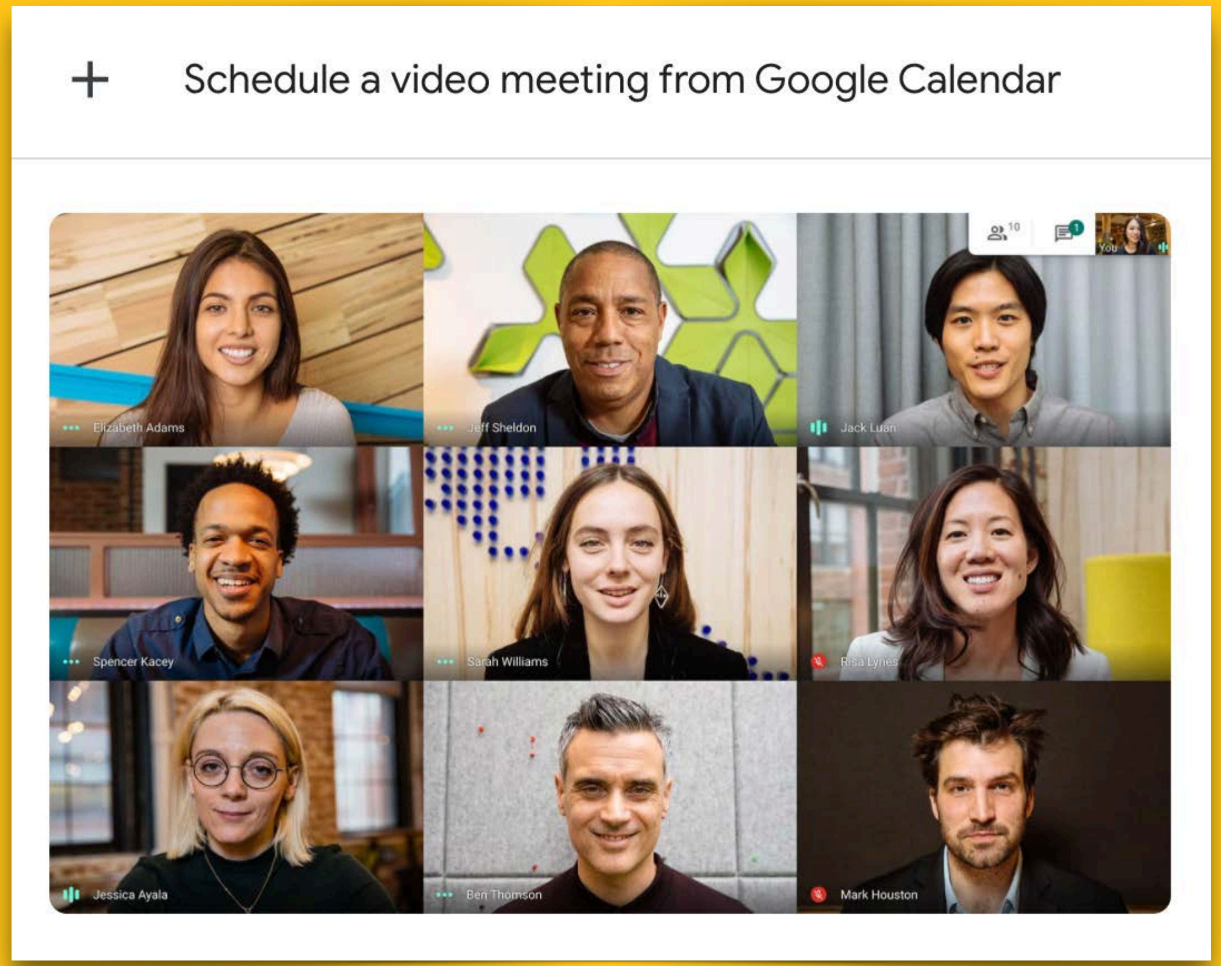
SYLLABUS

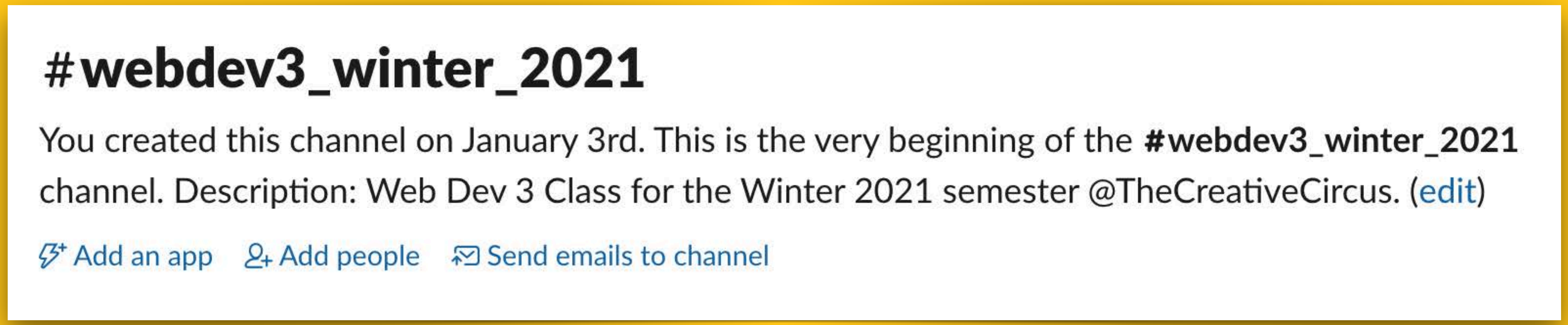
- Week 1: Git
- Week 2: Sass / Language Preprocessing
- Week 3: Big Images, SVG, Image Optimization
- Week 4: Advanced Backgrounds, SVG, Transitions
- Week 5: CSS Flex
- Week 6: CSS Grid
- Week 7: Loading Performance and SEO
- Week 8: Accessibility
- Week 9: Polish Projects
- Week 10: Polish Projects

Syllabus Link

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
- Git Repo: https://github.com/michellejames/WebDev3_Winter2021
- E-Mail: michellegraham90@yahoo.com



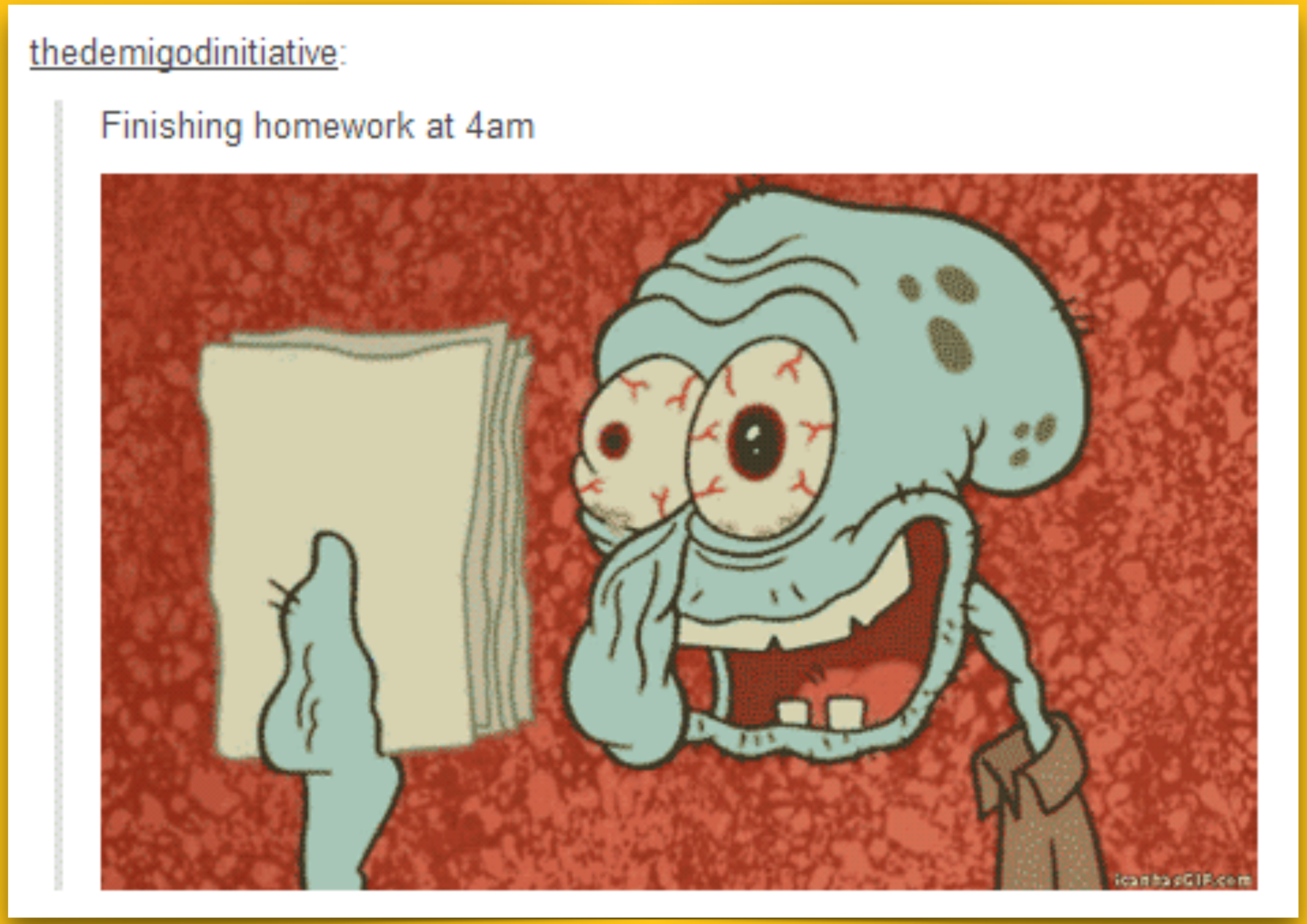


#webdev3_winter_2021

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)

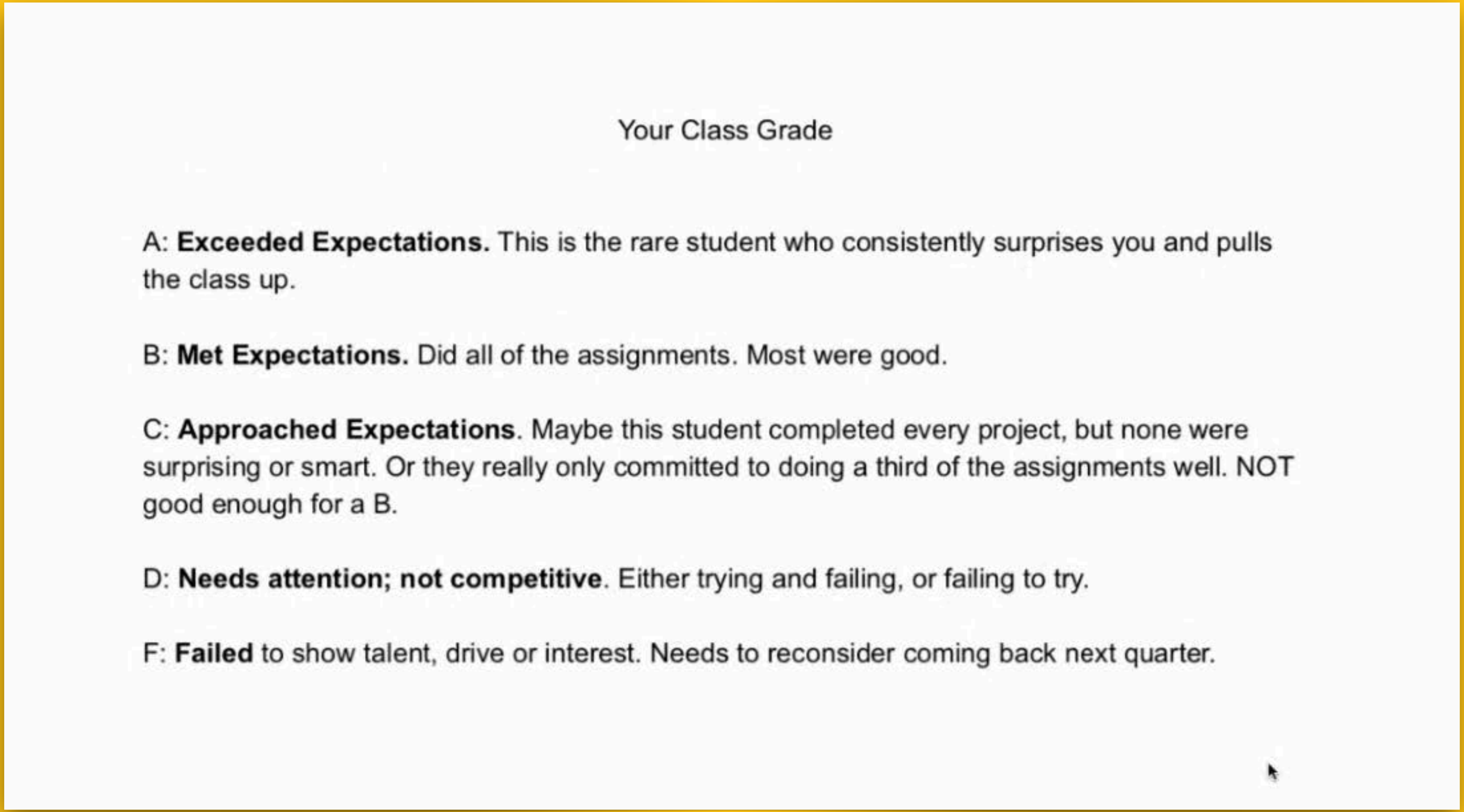
IMPORTANT THINGS - HOMEWORK

- Homework is due one minute before class
- 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


thedemigodinitiative:

Finishing homework at 4am

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.

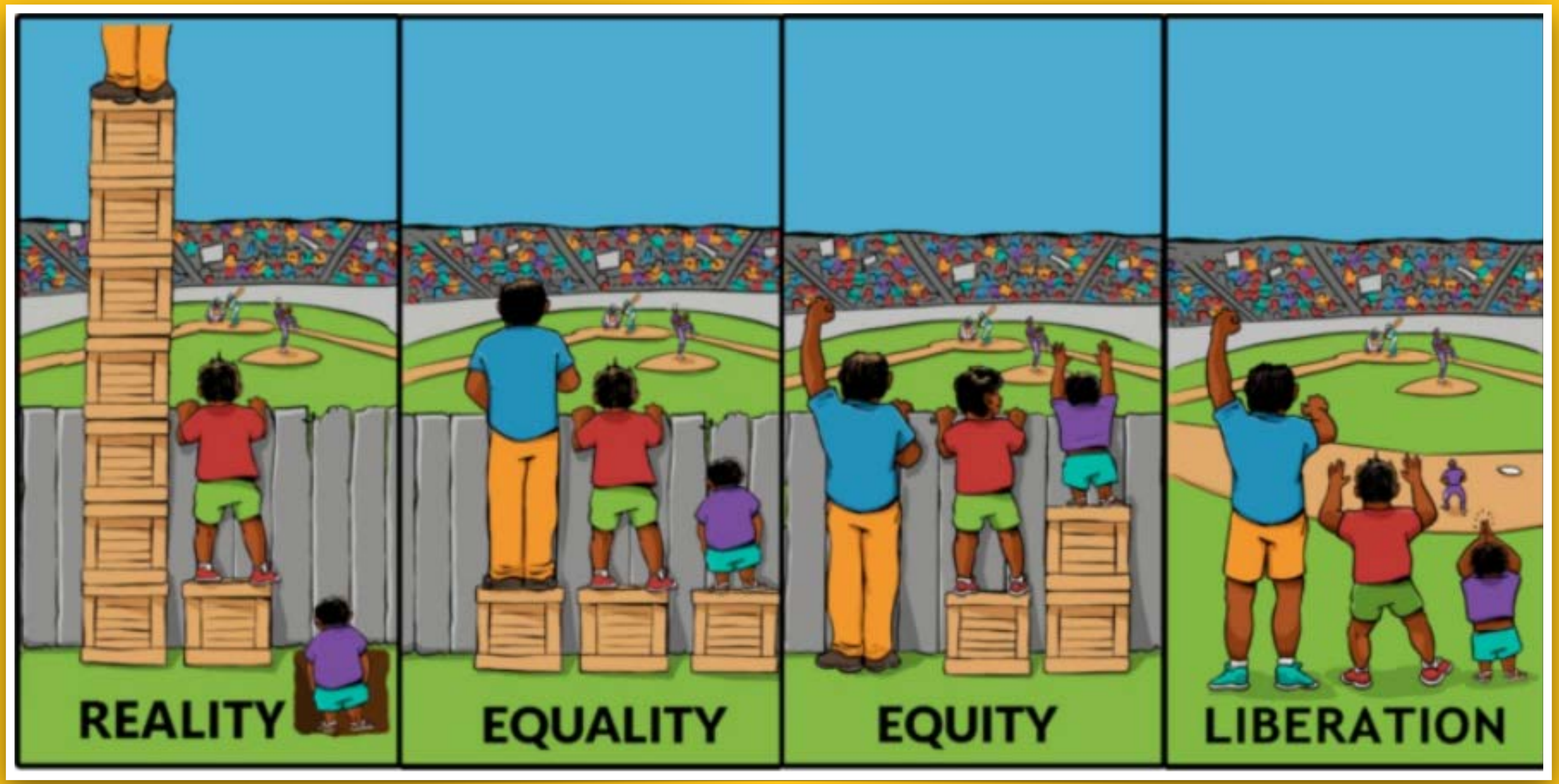
C: **Approached Expectations**. Maybe this student completed every project, but none were surprising or smart. Or they really only committed to doing a third of the assignments well. NOT good enough for a B.

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

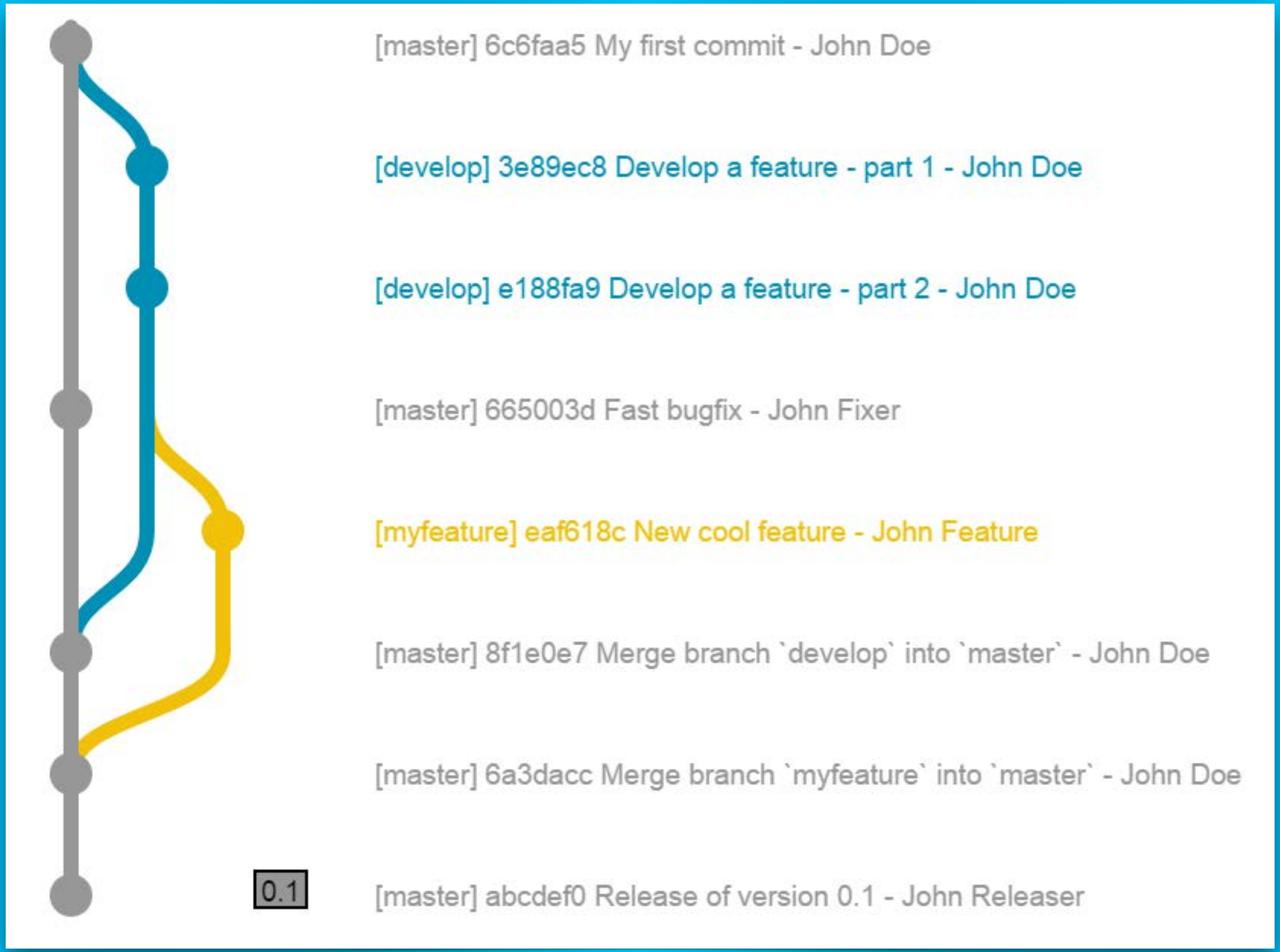
- Inclusivity: Think outside of your own personal realm
- Equality: Help one another figure it out
- Equity: You might need more or less help to thrive - both is OK!



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



[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

[myfeature] eaf618c New cool feature - John Feature

[master] 8f1e0e7 Merge branch `develop` into `master` - John Doe

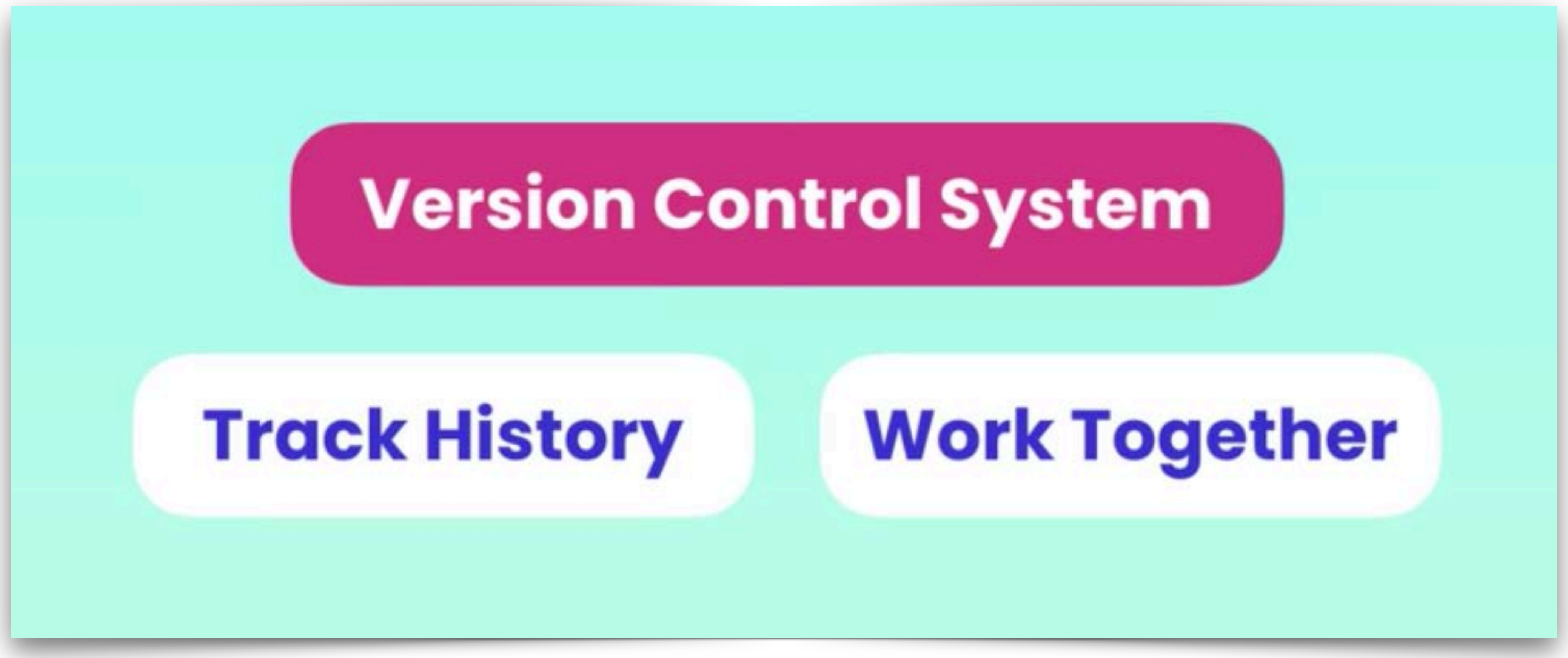
[master] 6a3dacc Merge branch `myfeature` into `master` - John Doe

[master] abcdef0 Release of version 0.1 - John Releaser

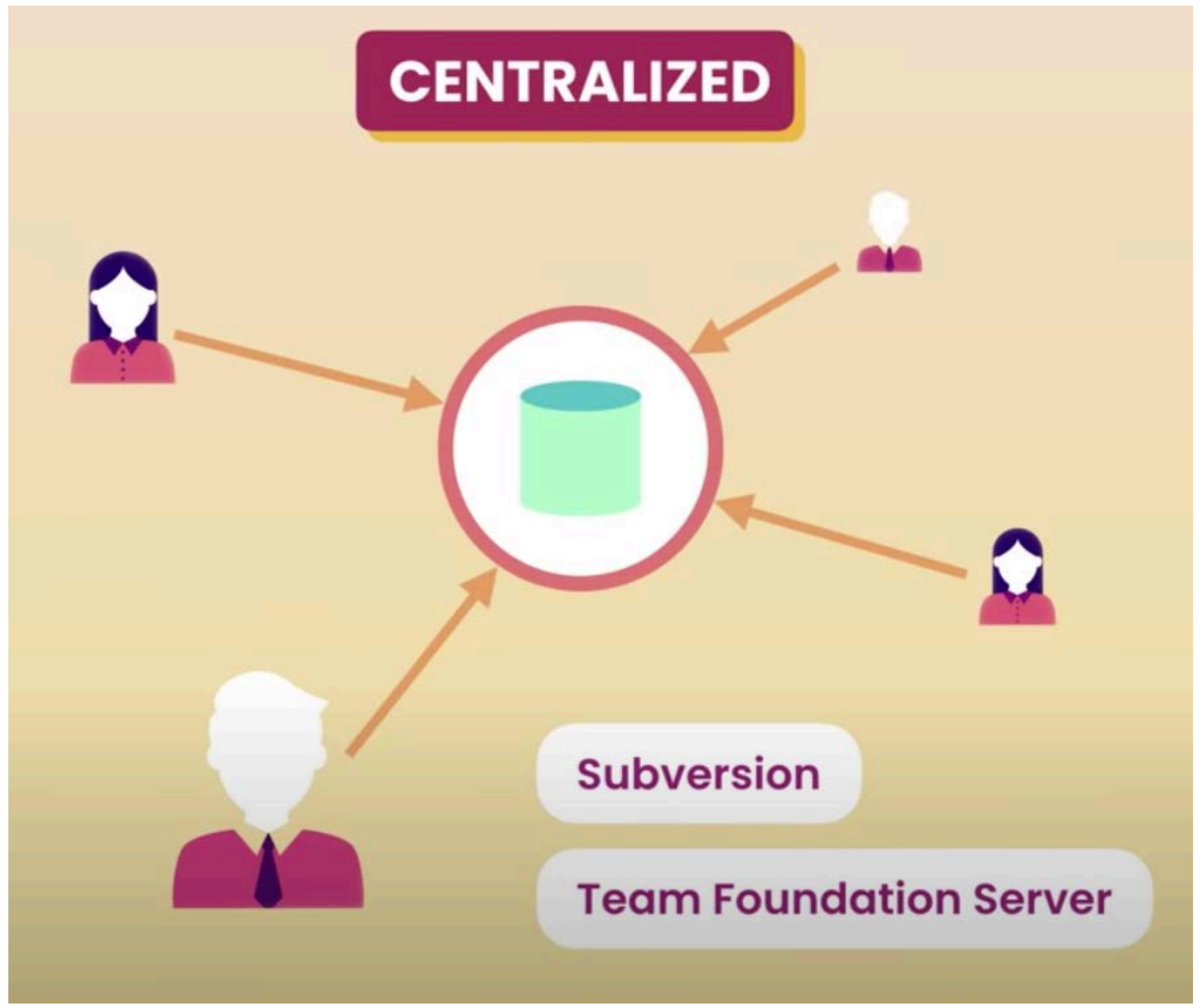
WHAT IS GIT?

- Most popular version control system in the world - as of today
- Free, Open Source, Fast, Scalable

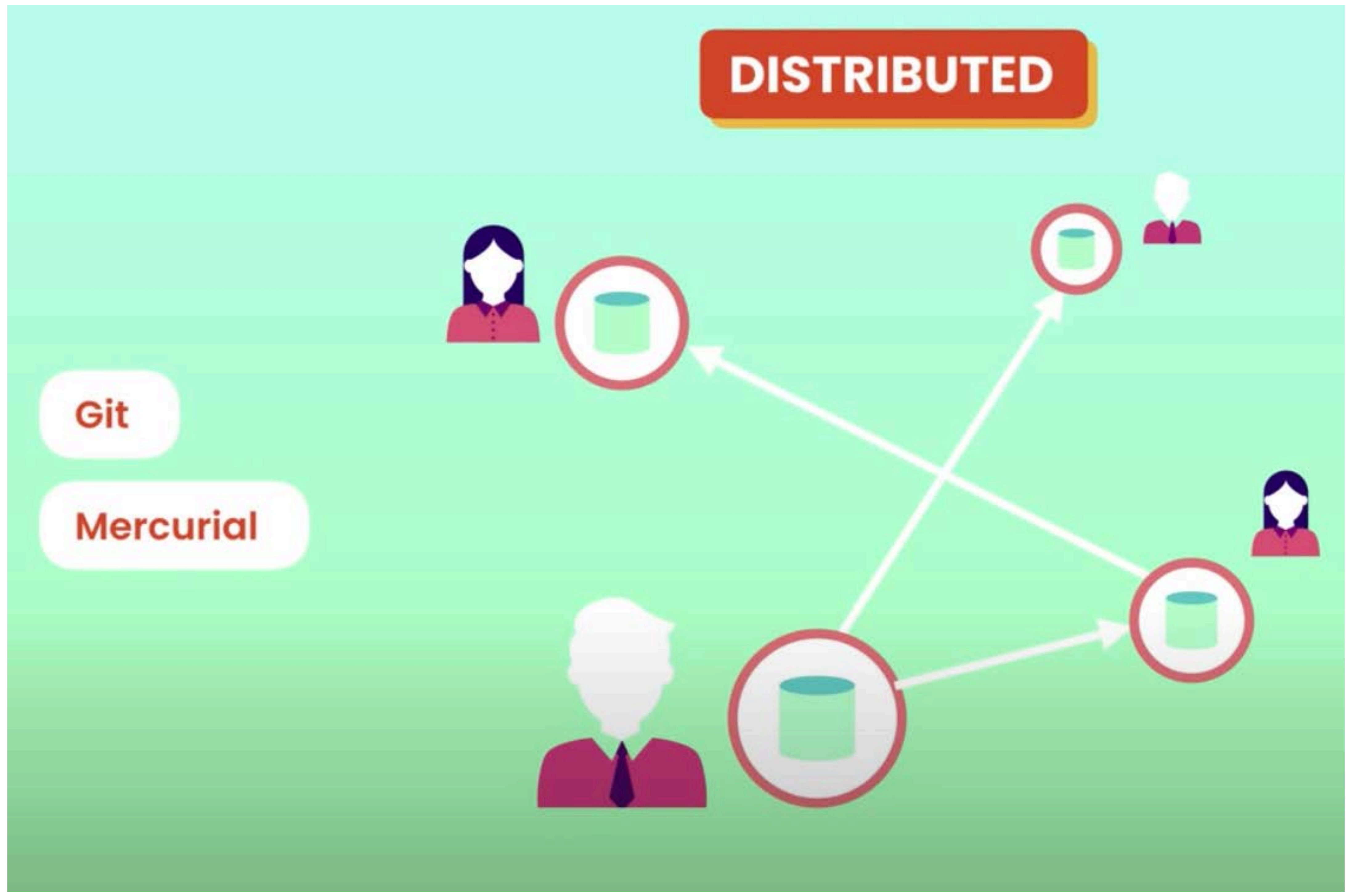
- It records the changes made to code over time in a special databank aka <u>repository</u>
- Makes it easy to collaborate with people on one project
- Works well on a wide range of operating systems and IDEs (Integrated Development Environments)



TWO DIFFERENT VERSION CONTROL SYSTEMS



VS



OTHER VERSION CONTROL SYSTEMS AKA CLIENTS & HOSTS

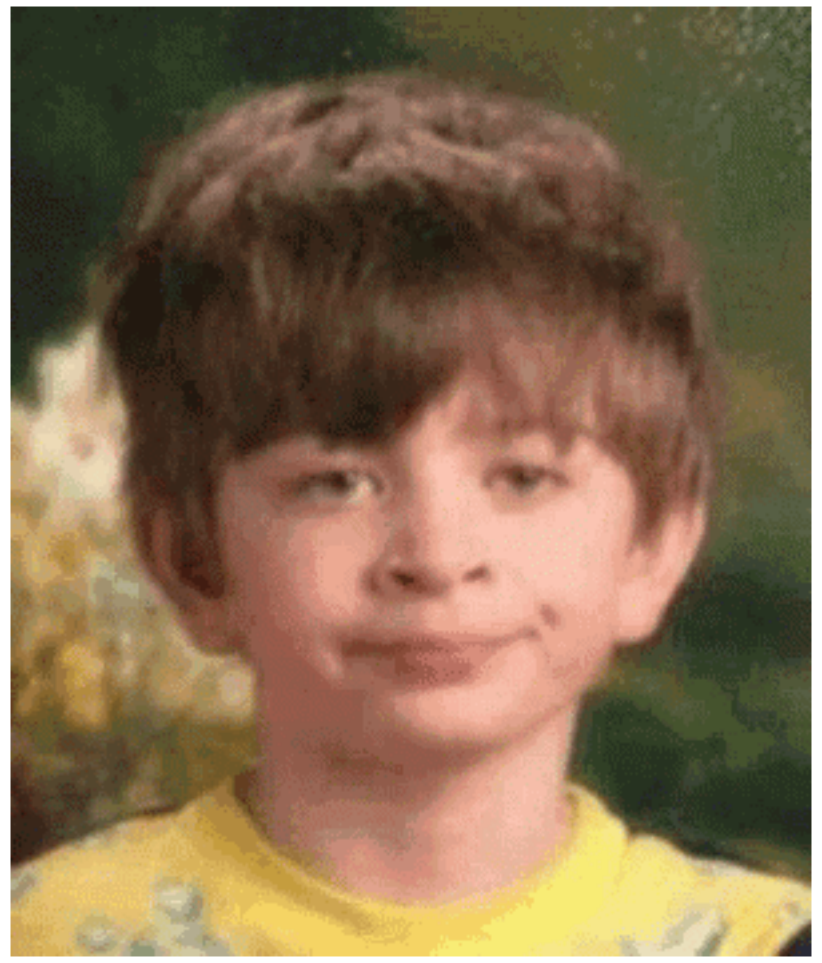
Clients are applications to visualize all those commands. We will use GitHub Desktop or simply the command line.

Hosts are places to store all those commands, i.e. your code. We will use GitHub.com

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

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

Just. So. Many.



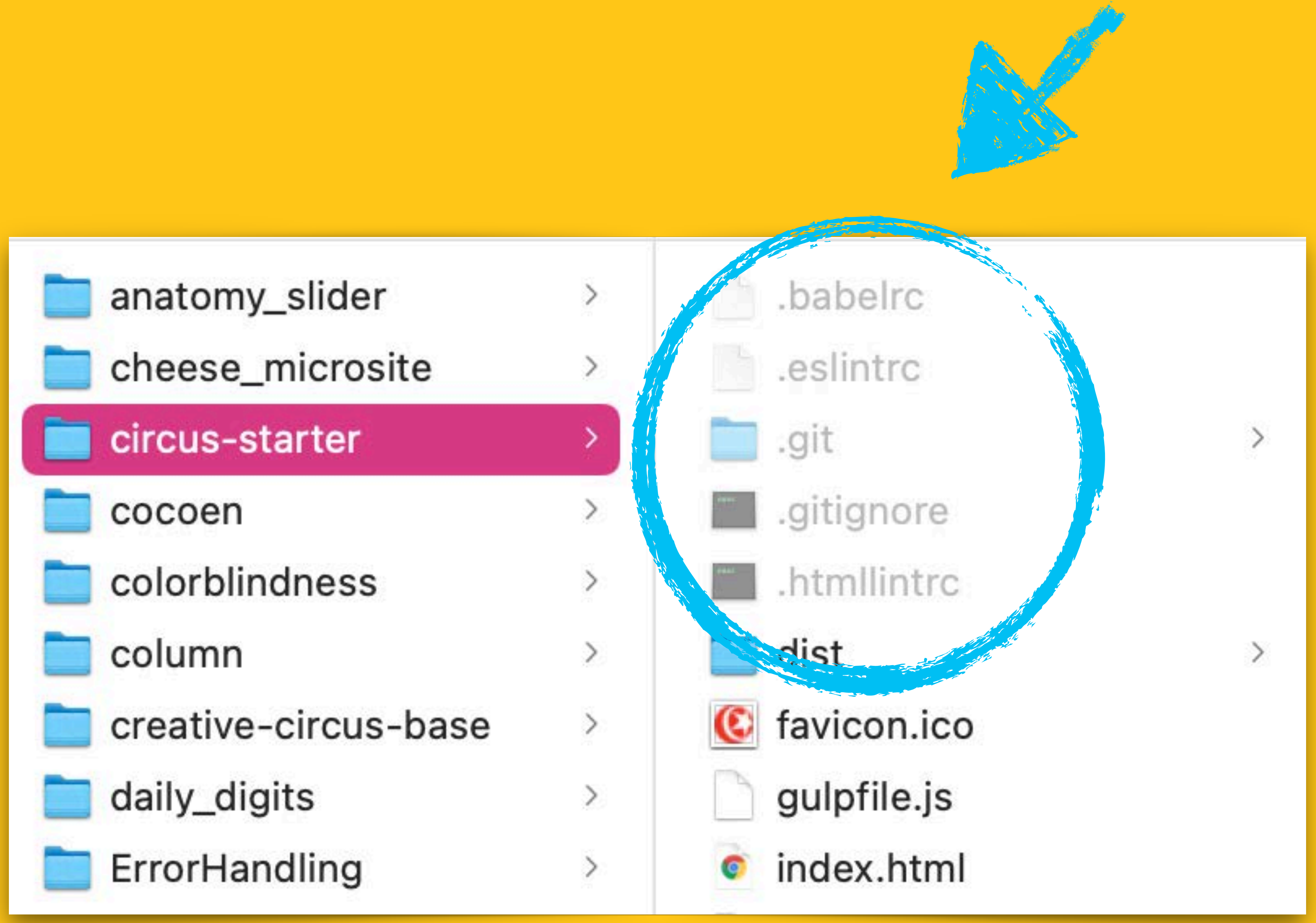
Source

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/FILES

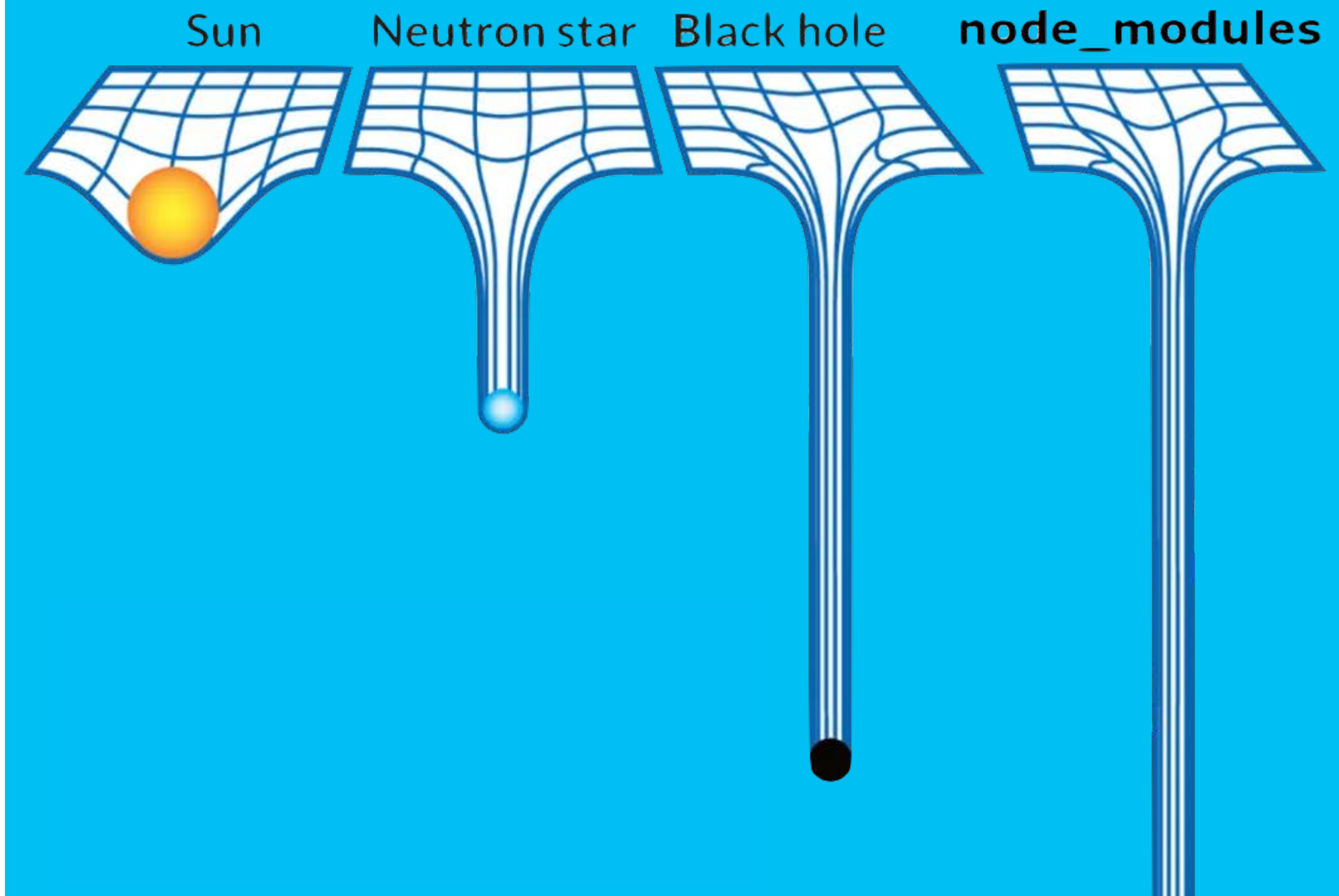
.GIT FOLDER

- 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!!!

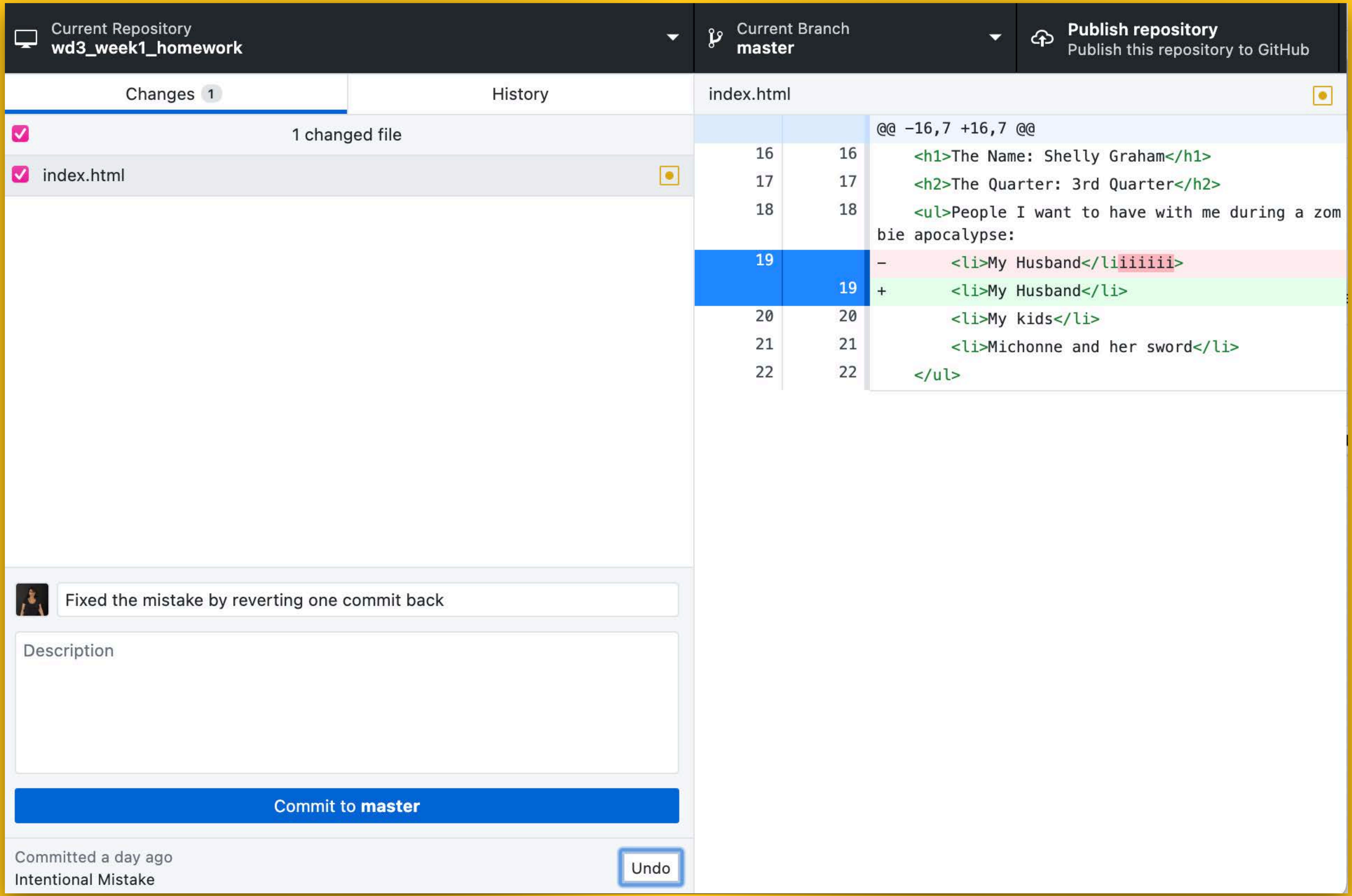
Heaviest Objects In the Universe:



Terminal command structure

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

OR SIMPLY USE GITHUB DESKTOP



```
index.html
@@ -16,7 +16,7 @@
     <h1>The Name: Shelly Graham</h1>
     <h2>The Quarter: 3rd Quarter</h2>
     <ul>People I want to have with me during a zom
bie apocalypse:
-        <li>My Husband</liiiiiii>
+        <li>My Husband</li>
         <li>My kids</li>
         <li>Michonne and her sword</li>
     </ul>
```

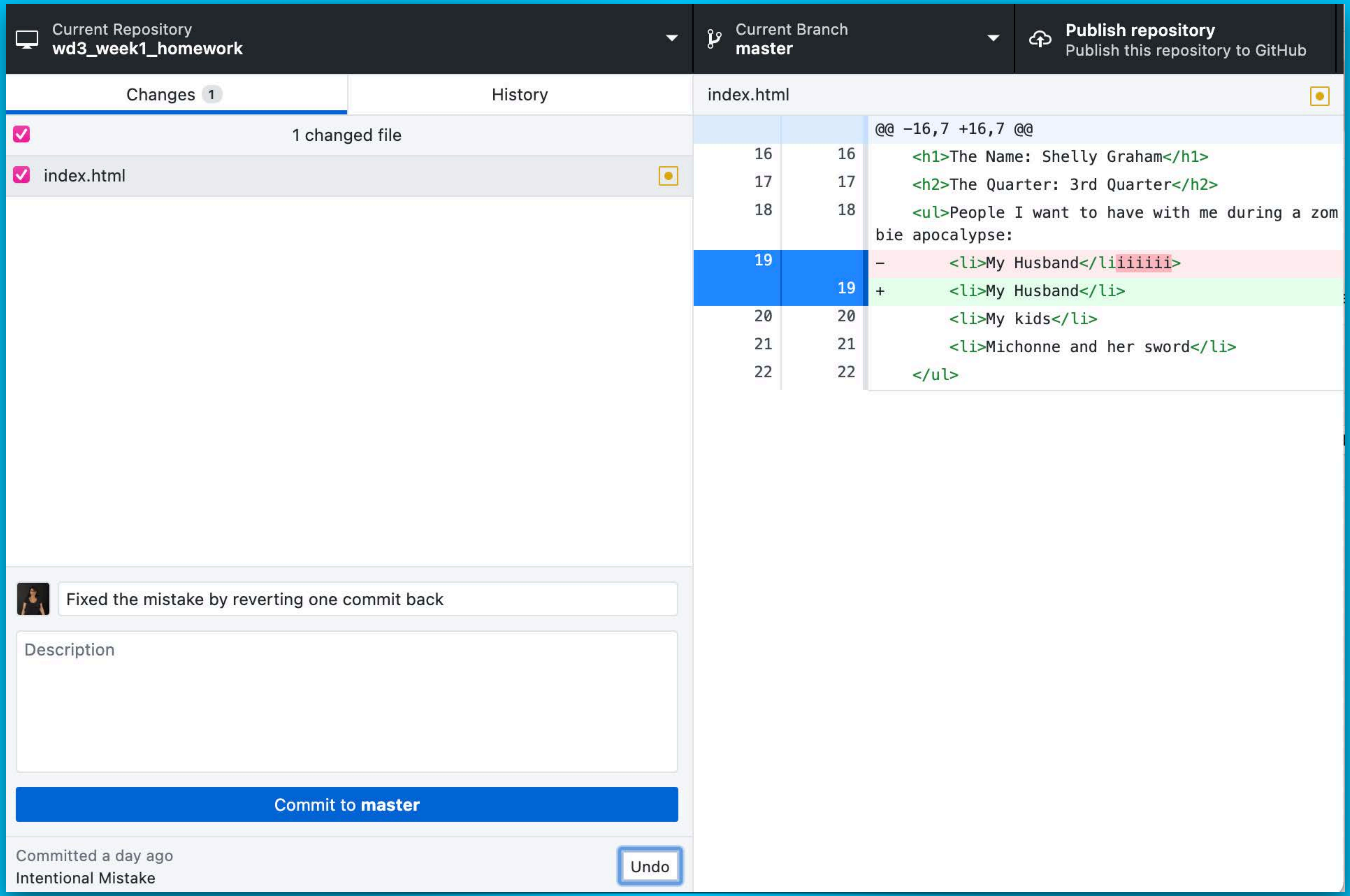
Fixed the mistake by reverting one commit back

Description

Committed a day ago
Intentional Mistake

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!



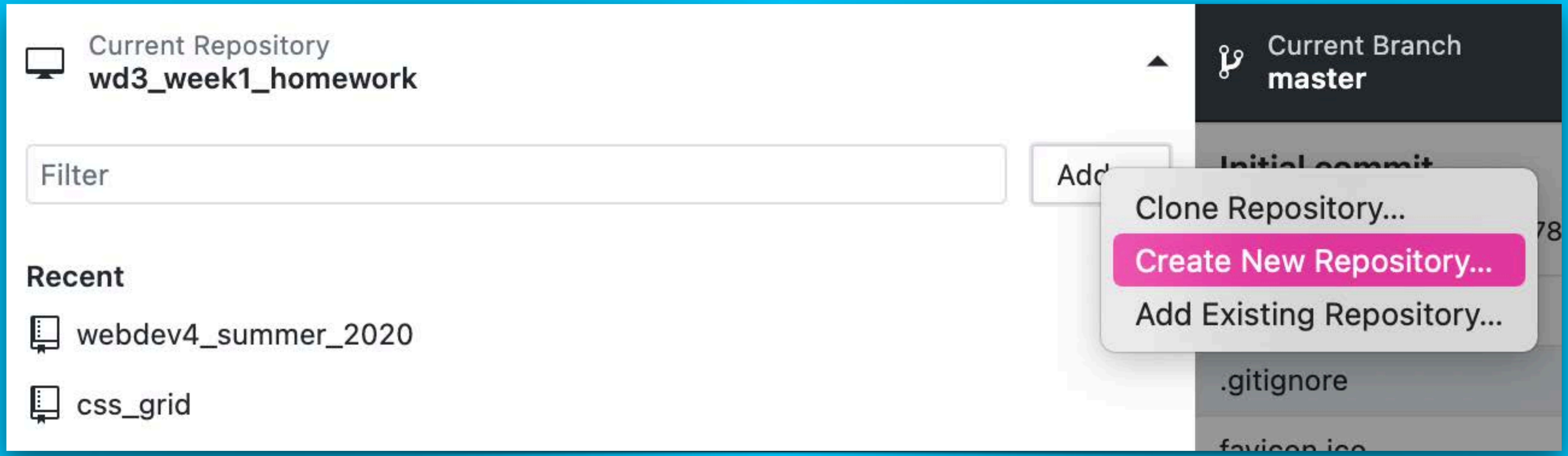
```
@@ -16,7 +16,7 @@
    <h1>The Name: Shelly Graham</h1>
    <h2>The Quarter: 3rd Quarter</h2>
    <ul>People I want to have with me during a zombie apocalypse:
-       <li>My Husband</liiiiiii>
+       <li>My Husband</li>
        <li>My kids</li>
        <li>Michonne and her sword</li>
    </ul>
```

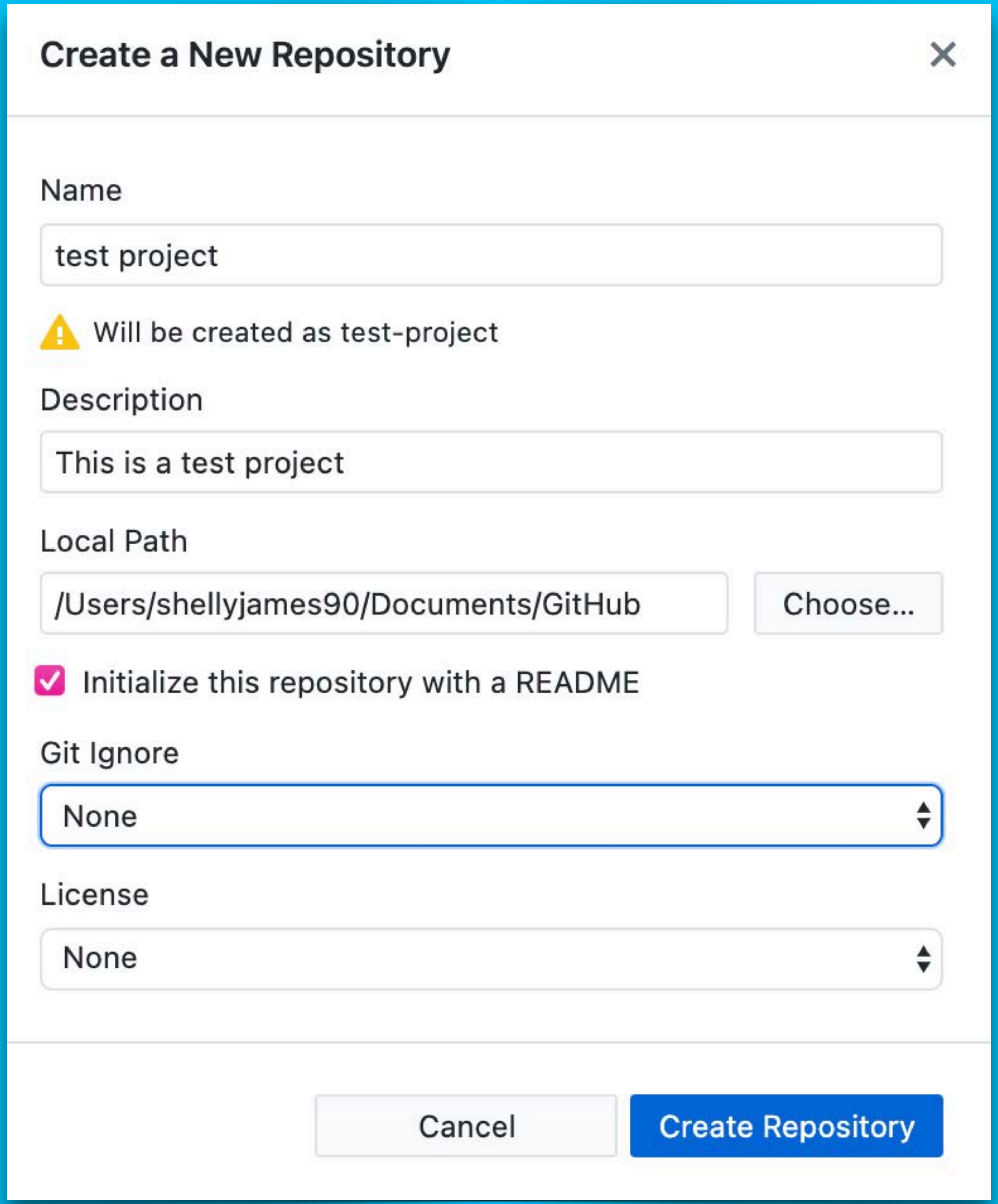
Fixed the mistake by reverting one commit back

Committed a day ago
Intentional Mistake

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

[✓] Initialize this repository with a README

Git Ignore: None

License: None

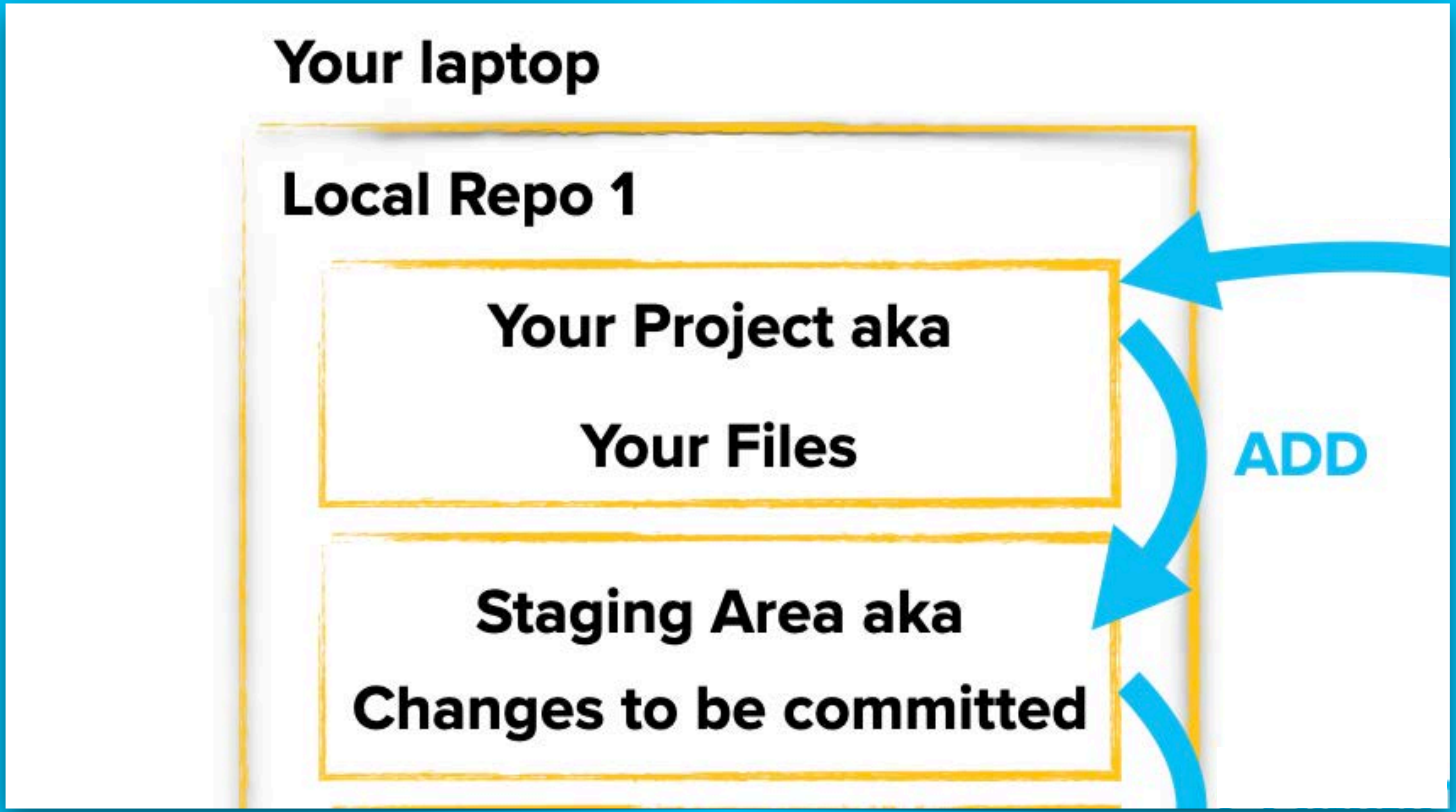
THE BASICS

1. Add
2. Commit
3. Pull
4. Push
5. Clone
6. Branching
7. Merging
8. Fork

Not so basic

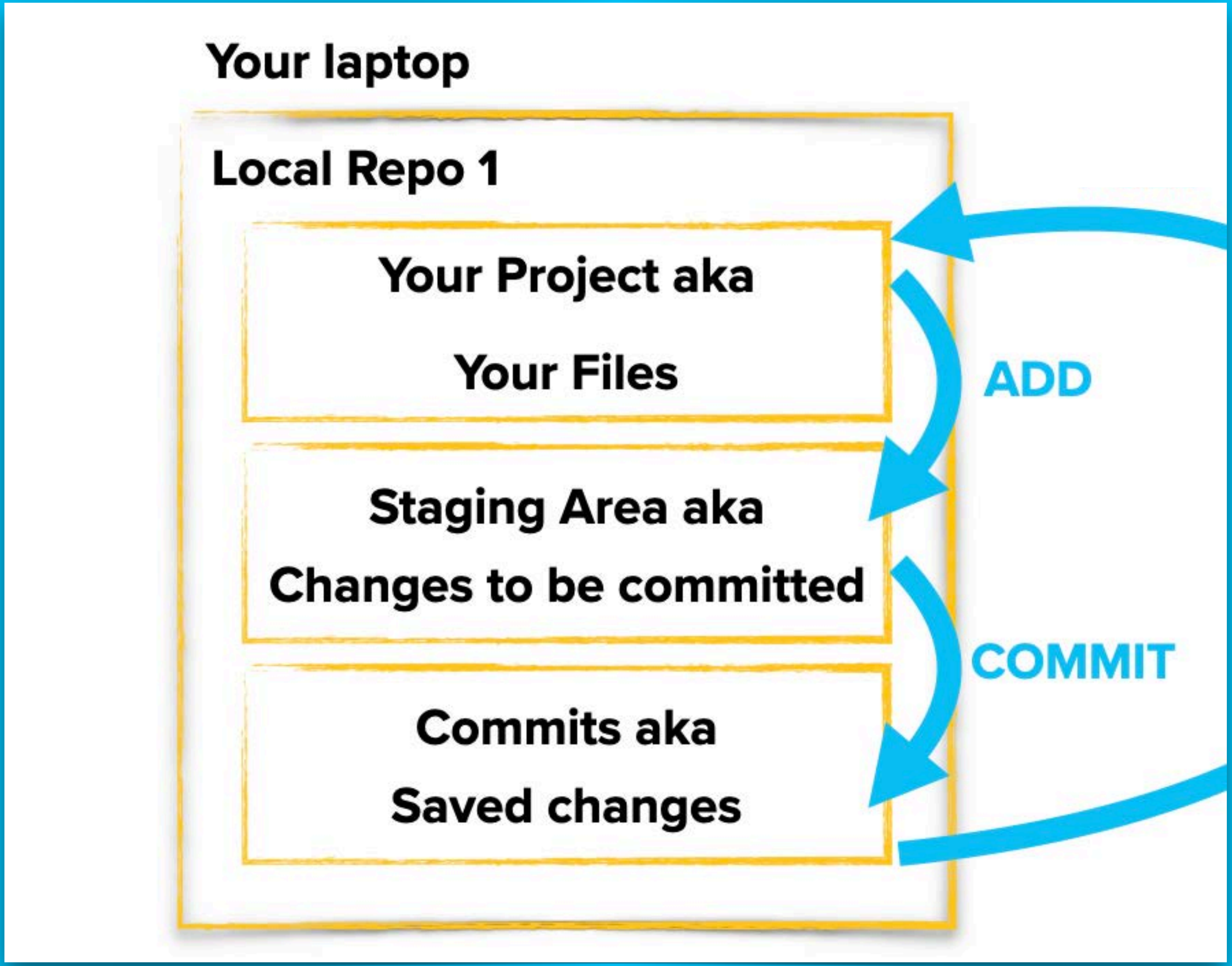
1. ADD

- 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!



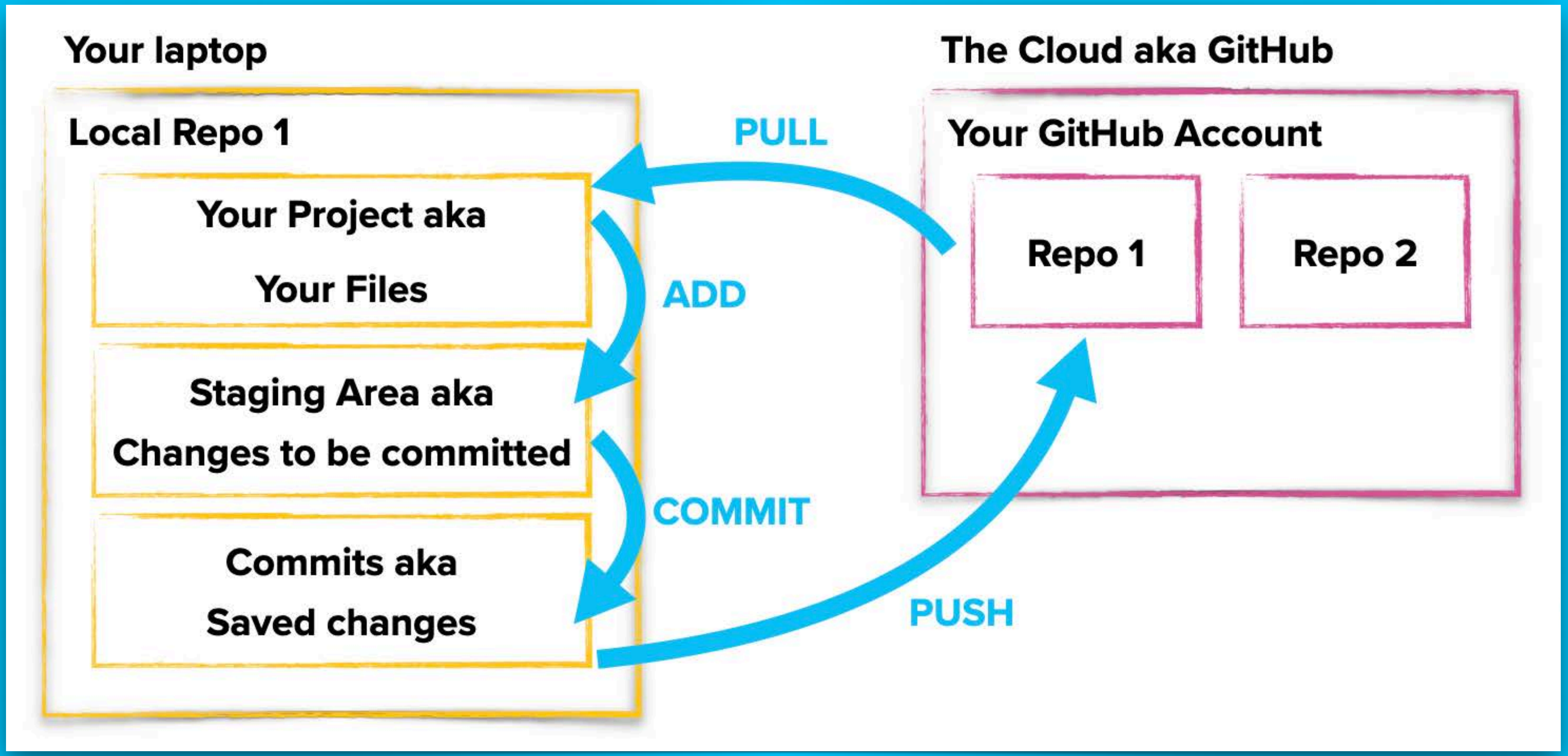
2. COMMIT

- 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”`



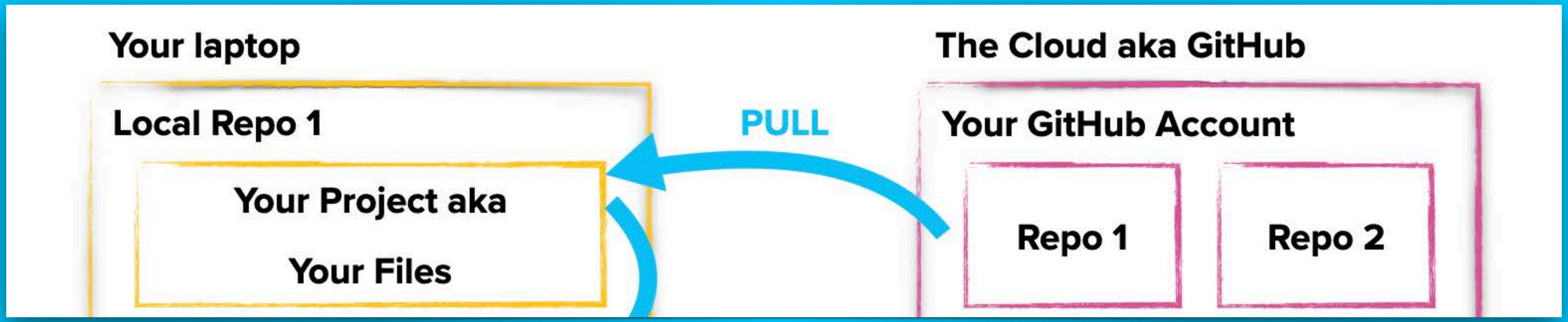
3. PUSH

- 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



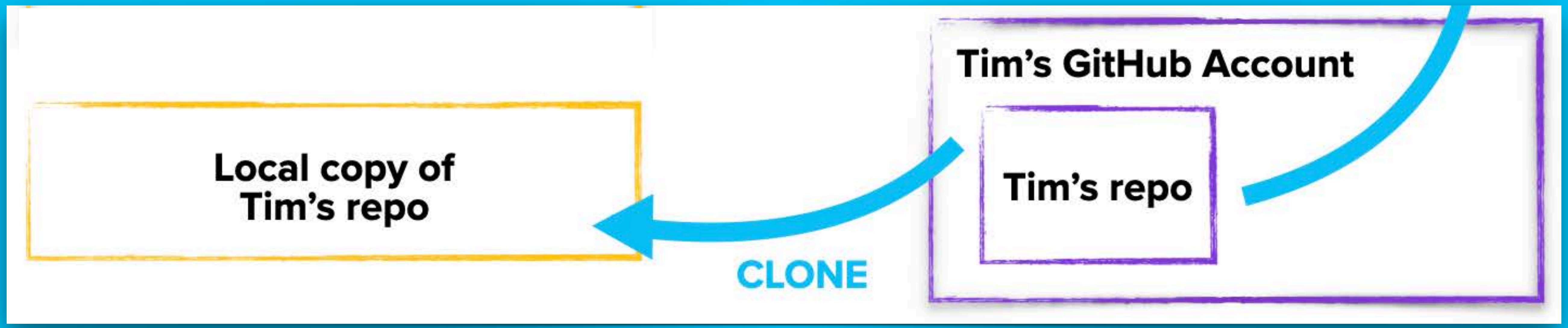
4. PULL

- 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!



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!



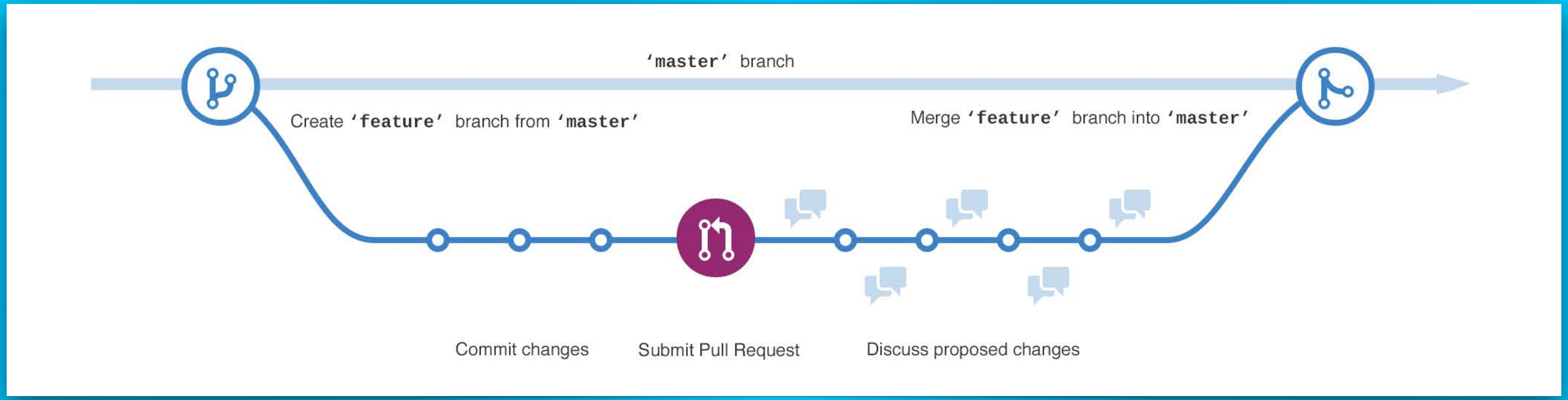
6. BRANCHING

- “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”

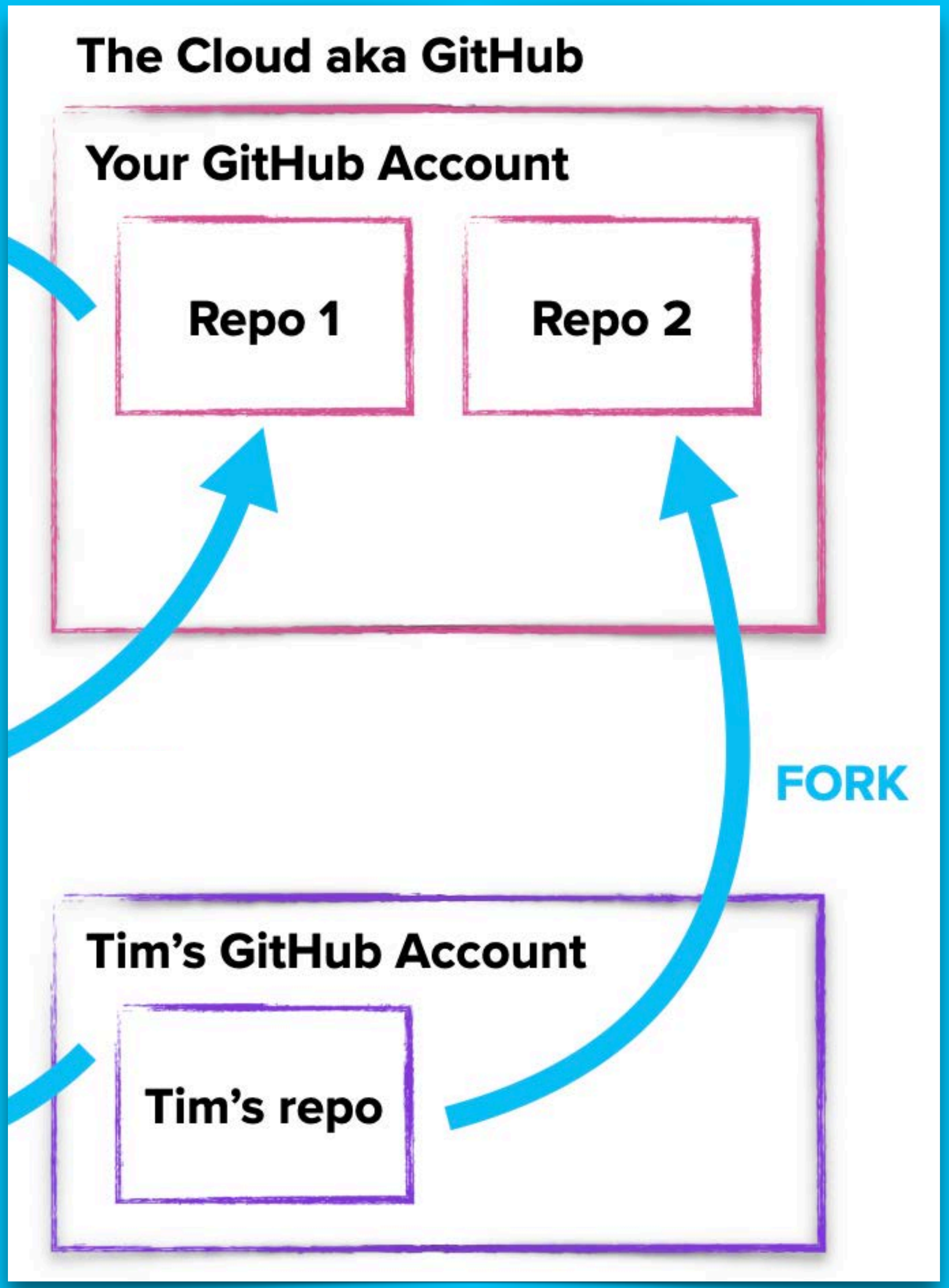
7. MERGING

- “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!

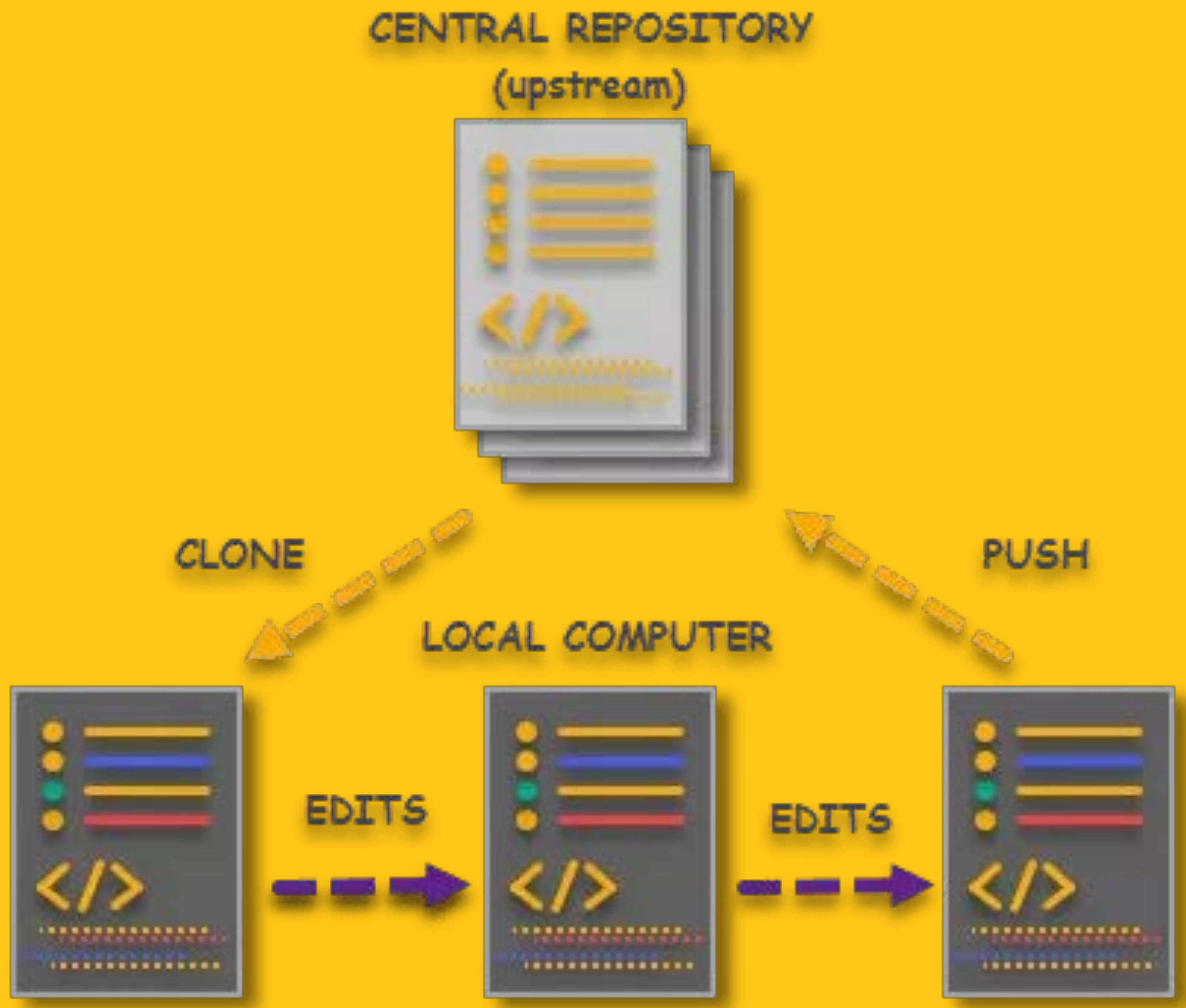

'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

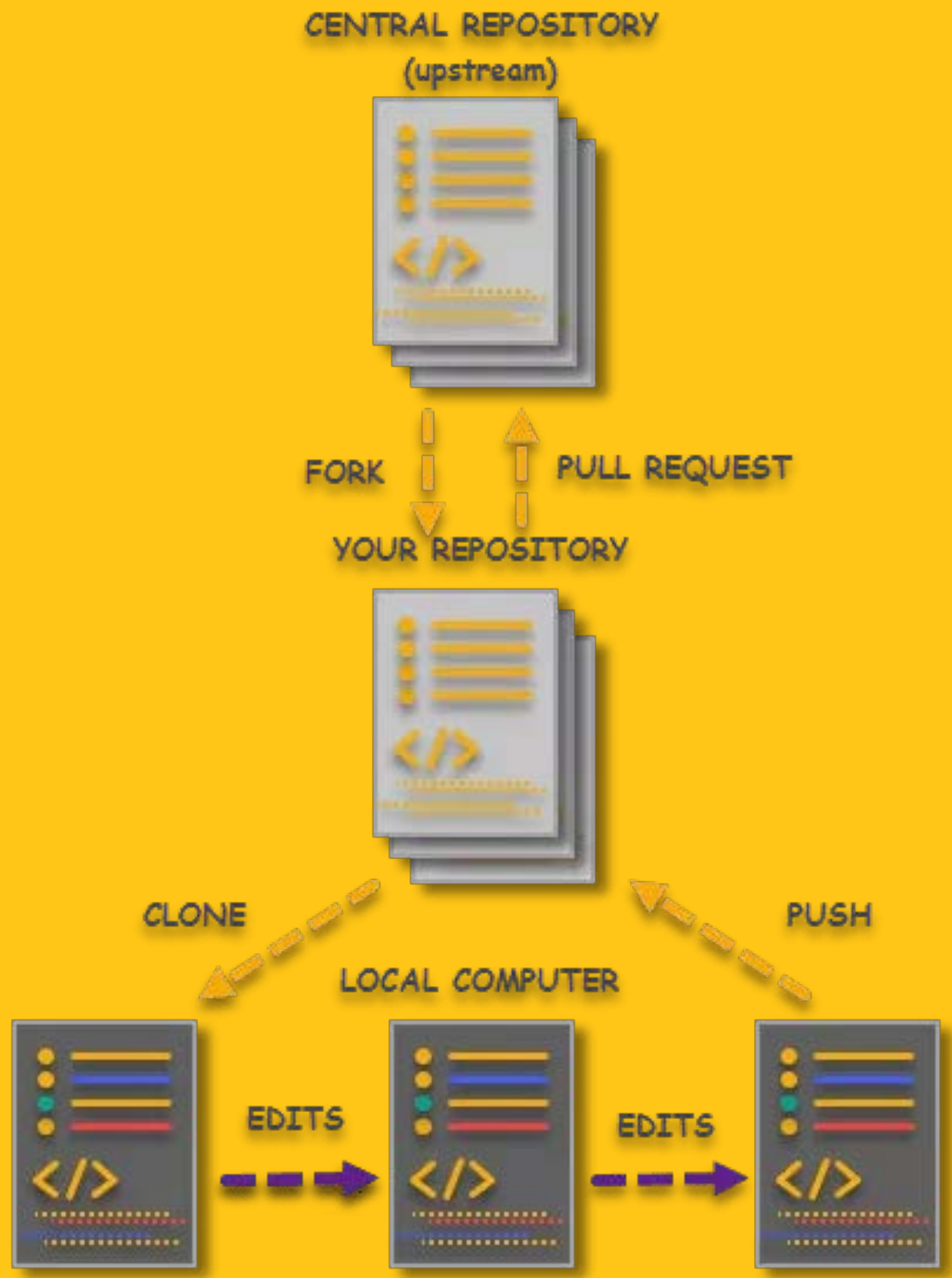
8. FORK

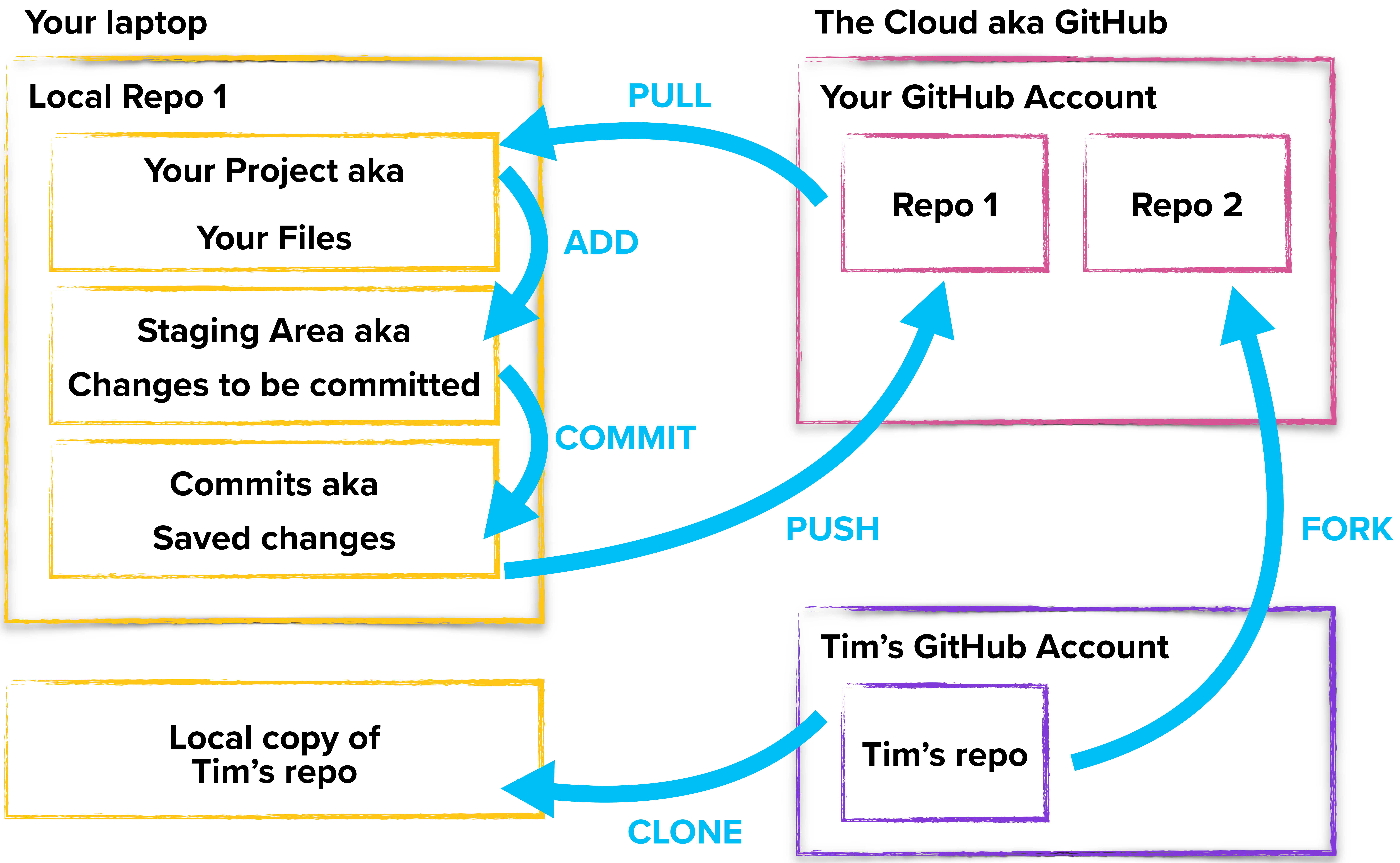
- Duplicating someone else's repo for yourself to edit
- On GitHub.com, use Fork Button!
- Creates a copy of the project that you now own
- You can also pull updates from the original project
- Boom. Now you have a local copy of a remote Git repo!



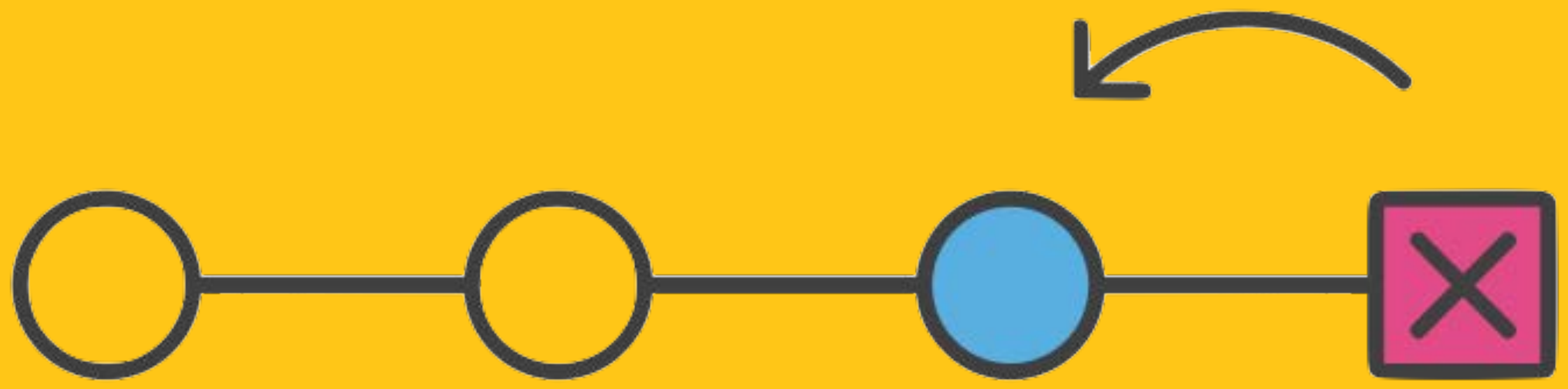
CLONING = COMMAND VS FORKING = CONCEPT






Your laptop
The Cloud aka GitHub
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
PULL
ADD
COMMIT
PUSH
FORK
CLONE
Your GitHub Account
Repo 1
Repo 2
Tim's GitHub Account
Tim's repo

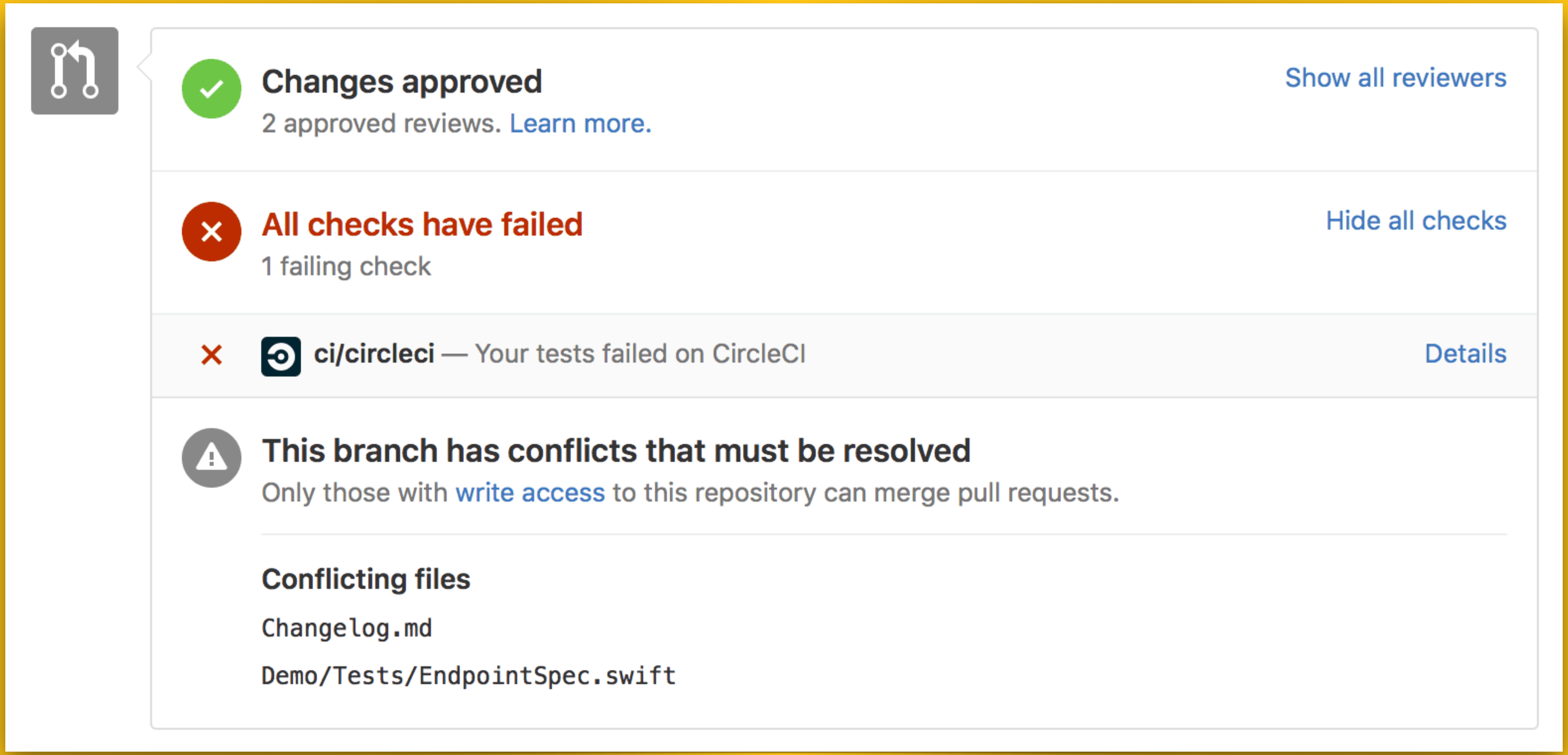
REVERTING



- 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 go back to 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
2 approved reviews. Learn more.
Show all reviewers

All checks have failed
1 failing check
Hide all checks

✗ **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

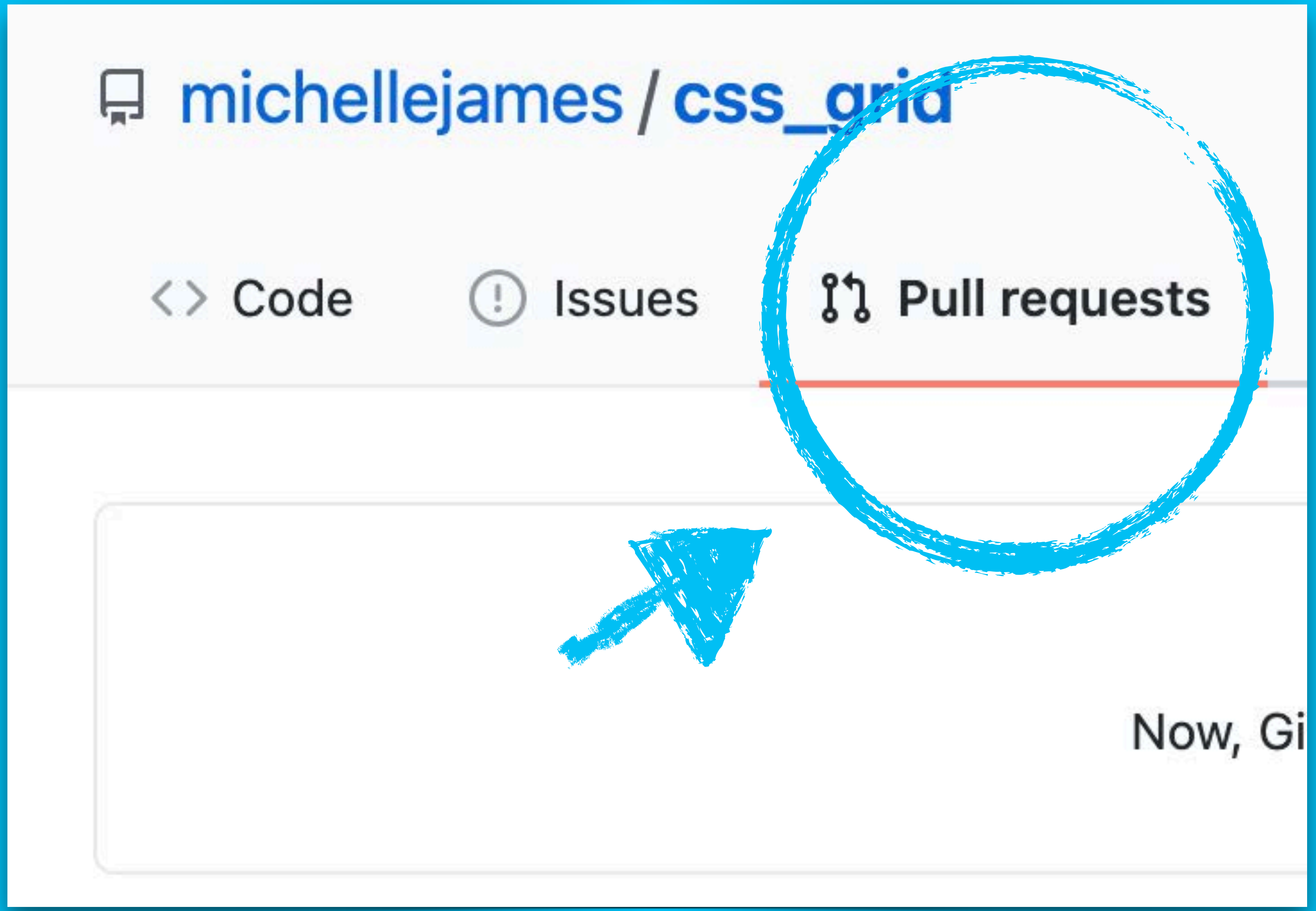
- Type “git mergetool” in command line
- SourceTree
- VSCode
- Sublime Merge
- Meld
- kDiff

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:

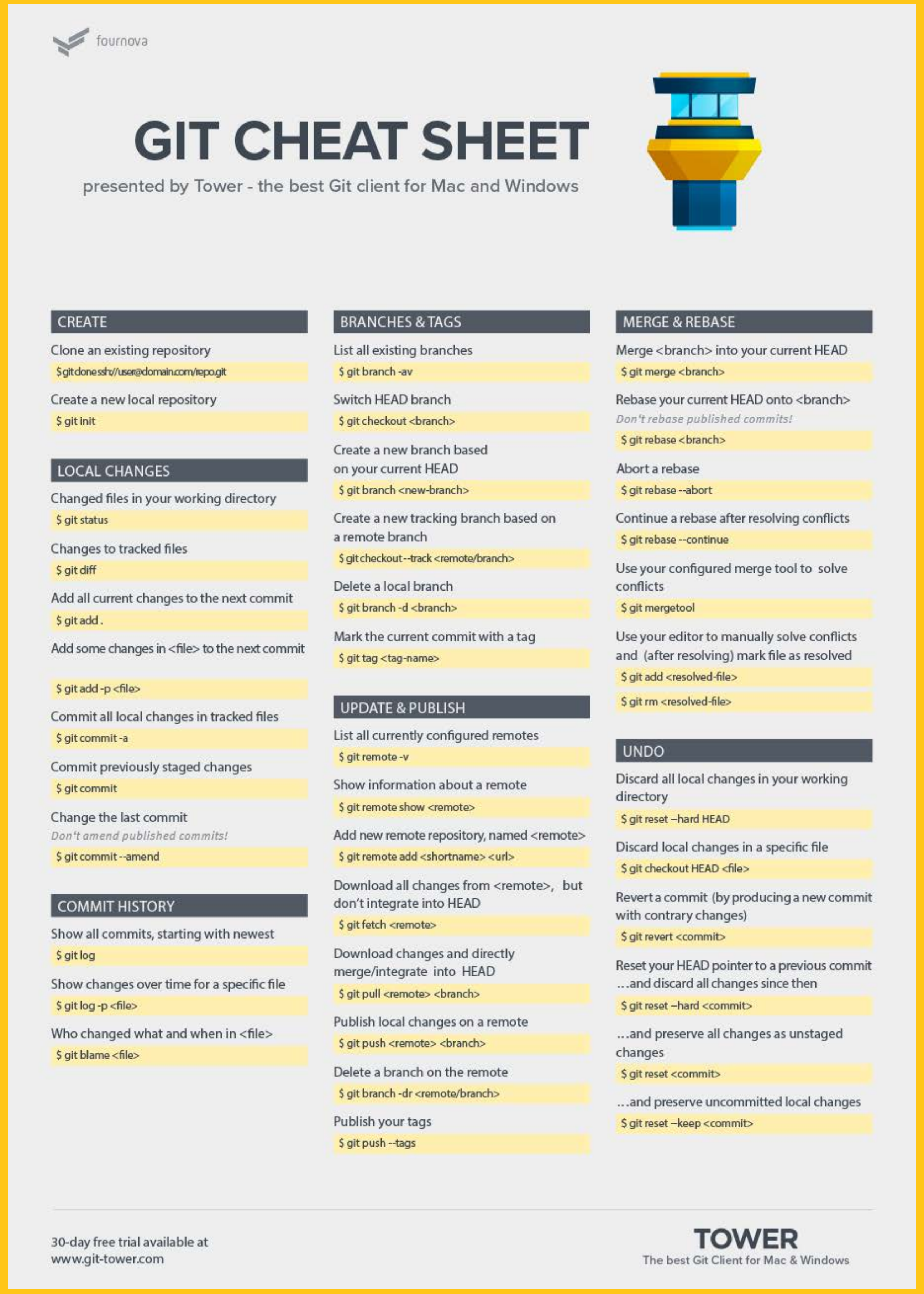
- Make sure you're in the right branch
- Hit the Pull Request button
- Add a description and send it
- Owner can review and accept or decline



QUESTIONS?

GIT AND COMMAND LINE CHEAT SHEETS

- Git Documentation



fournova

GIT CHEAT SHEET

presented by Tower - the best Git client for Mac and Windows

CREATE

Clone an existing repository
`$ git clone ssh://user@domain.com/repo.git`

Create a new local repository
`$ git init`

LOCAL CHANGES

Changed files in your working directory
`$ git status`

Changes to tracked files
`$ git diff`

Add all current changes to the next commit
`$ git add .`

Add some changes in <file> to the next commit
`$ git add -p <file>`

Commit all local changes in tracked files
`$ git commit -a`

Commit previously staged changes
`$ git commit`

Change the last commit
Don't amend published commits!
`$ git commit --amend`

COMMIT HISTORY

Show all commits, starting with newest
`$ git log`

Show changes over time for a specific file
`$ git log -p <file>`

Who changed what and when in <file>
`$ git blame <file>`

BRANCHES & TAGS

List all existing branches
`$ git branch -av`

Switch HEAD branch
`$ git checkout <branch>`

Create a new branch based on your current HEAD
`$ git branch <new-branch>`

Create a new tracking branch based on a remote branch
`$ git checkout --track <remote/branch>`

Delete a local branch
`$ git branch -d <branch>`

Mark the current commit with a tag
`$ git tag <tag-name>`

UPDATE & PUBLISH

List all currently configured remotes
`$ git remote -v`

Show information about a remote
`$ git remote show <remote>`

Add new remote repository, named <remote>
`$ git remote add <shortname> <url>`

Download all changes from <remote>, but don't integrate into HEAD
`$ git fetch <remote>`

Download changes and directly merge/integrate into HEAD
`$ git pull <remote> <branch>`

Publish local changes on a remote
`$ git push <remote> <branch>`

Delete a branch on the remote
`$ git branch -dr <remote/branch>`

Publish your tags
`$ git push --tags`

MERGE & REBASE

Merge <branch> into your current HEAD
`$ git merge <branch>`

Rebase your current HEAD onto <branch>
Don't rebase published commits!
`$ git rebase <branch>`

Abort a rebase
`$ git rebase --abort`

Continue a rebase after resolving conflicts
`$ git rebase --continue`

Use your configured merge tool to solve conflicts
`$ git mergetool`

Use your editor to manually solve conflicts and (after resolving) mark file as resolved
`$ git add <resolved-file>`
`$ git rm <resolved-file>`

UNDO

Discard all local changes in your working directory
`$ git reset –hard HEAD`

Discard local changes in a specific file
`$ git checkout HEAD <file>`

Revert a commit (by producing a new commit with contrary changes)
`$ git revert <commit>`

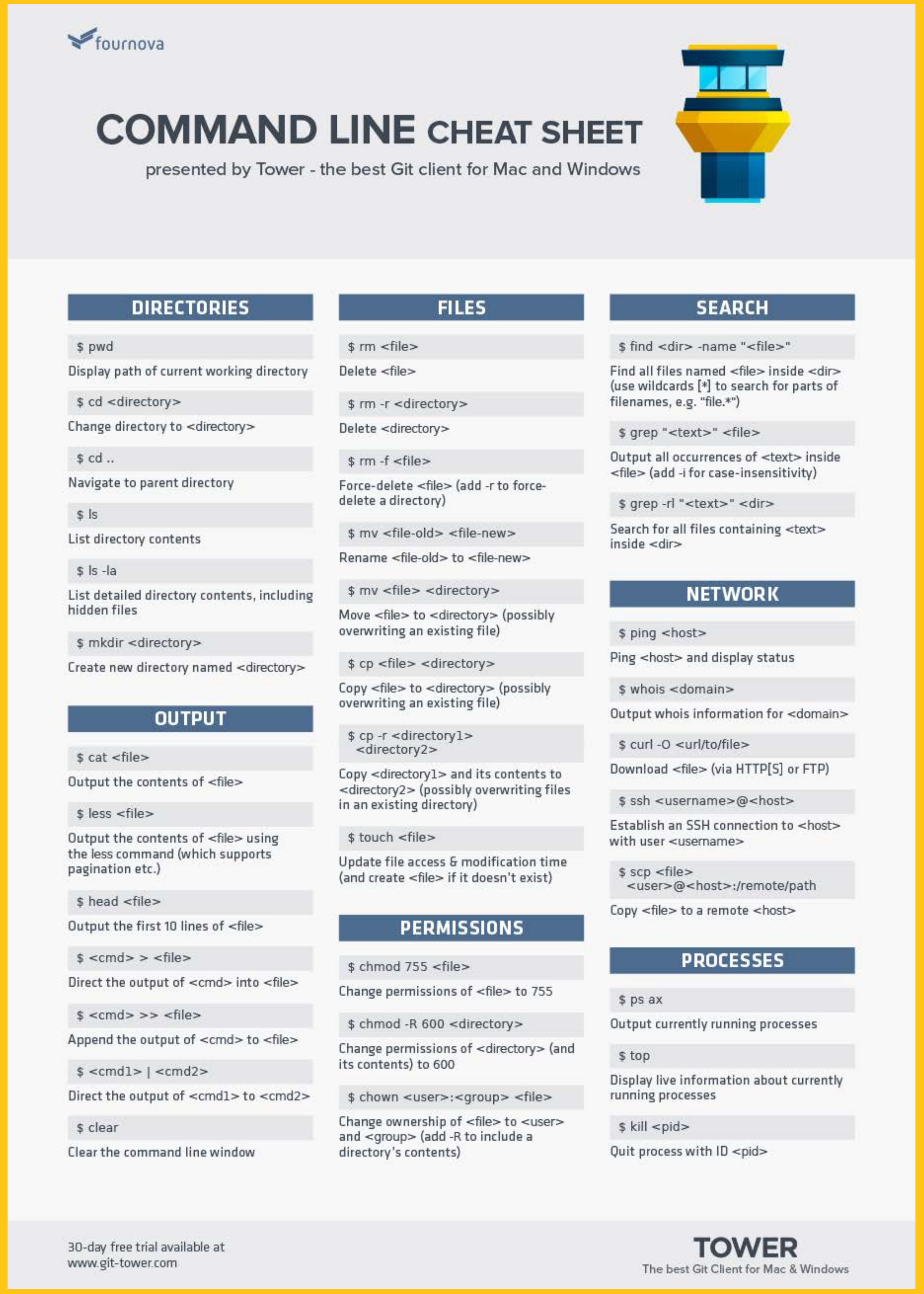
Reset your HEAD pointer to a previous commit
…and discard all changes since then
`$ git reset –hard <commit>`

…and preserve all changes as unstaged changes
`$ git reset <commit>`

…and preserve uncommitted local changes
`$ git reset –keep <commit>`

30-day free trial available at
www.git-tower.com

TOWER
The best Git Client for Mac & Windows



fournova

COMMAND LINE CHEAT SHEET

presented by Tower - the best Git client for Mac and Windows

DIRECTORIES

`$ pwd`
Display path of current working directory

`$ cd <directory>`
Change directory to <directory>

`$ cd ..`
Navigate to parent directory

`$ ls`
List directory contents

`$ ls -la`
List detailed directory contents, including hidden files

`$ mkdir <directory>`
Create new directory named <directory>

OUTPUT

`$ cat <file>`
Output the contents of <file>

`$ less <file>`
Output the contents of <file> using the less command (which supports pagination etc.)

`$ head <file>`
Output the first 10 lines of <file>

`$ <cmd> > <file>`
Direct the output of <cmd> into <file>

`$ <cmd> >> <file>`
Append the output of <cmd> to <file>

`$ <cmd1> | <cmd2>`
Direct the output of <cmd1> to <cmd2>

`$ clear`
Clear the command line window

FILES

`$ rm <file>`
Delete <file>

`$ rm -r <directory>`
Delete <directory>

`$ rm -f <file>`
Force-delete <file> (add -r to force-delete a directory)

`$ mv <file-old> <file-new>`
Rename <file-old> to <file-new>

`$ mv <file> <directory>`
Move <file> to <directory> (possibly overwriting an existing file)

`$ cp <file> <directory>`
Copy <file> to <directory> (possibly overwriting an existing file)

`$ cp -r <directory1> <directory2>`
Copy <directory1> and its contents to <directory2> (possibly overwriting files in an existing directory)

`$ touch <file>`
Update file access & modification time (and create <file> if it doesn't exist)

PERMISSIONS

`$ chmod 755 <file>`
Change permissions of <file> to 755

`$ chmod -R 600 <directory>`
Change permissions of <directory> (and its contents) to 600

`$ chown <user>:<group> <file>`
Change ownership of <file> to <user> and <group> (add -R to include a directory's contents)

SEARCH

`$ find <dir> -name "<file>"`
Find all files named <file> inside <dir> (use wildcards [*] to search for parts of filenames, e.g. "file.*")

`$ grep "<text>" <file>`
Output all occurrences of <text> inside <file> (add -i for case-insensitivity)

`$ grep -rl "<text>" <dir>`
Search for all files containing <text> inside <dir>

NETWORK

`$ ping <host>`
Ping <host> and display status

`$ whois <domain>`
Output whois information for <domain>

`$ curl -O <url/to/file>`
Download <file> (via HTTP[S] or FTP)

`$ ssh <username>@<host>`
Establish an SSH connection to <host> with user <username>

`$ scp <file> <user>@<host>:/remote/path`
Copy <file> to a remote <host>

PROCESSES

`$ ps ax`
Output currently running processes

`$ top`
Display live information about currently running processes

`$ kill <pid>`
Quit process with ID <pid>

30-day free trial available at
www.git-tower.com

TOWER
The best Git Client for Mac & Windows

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
  - 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