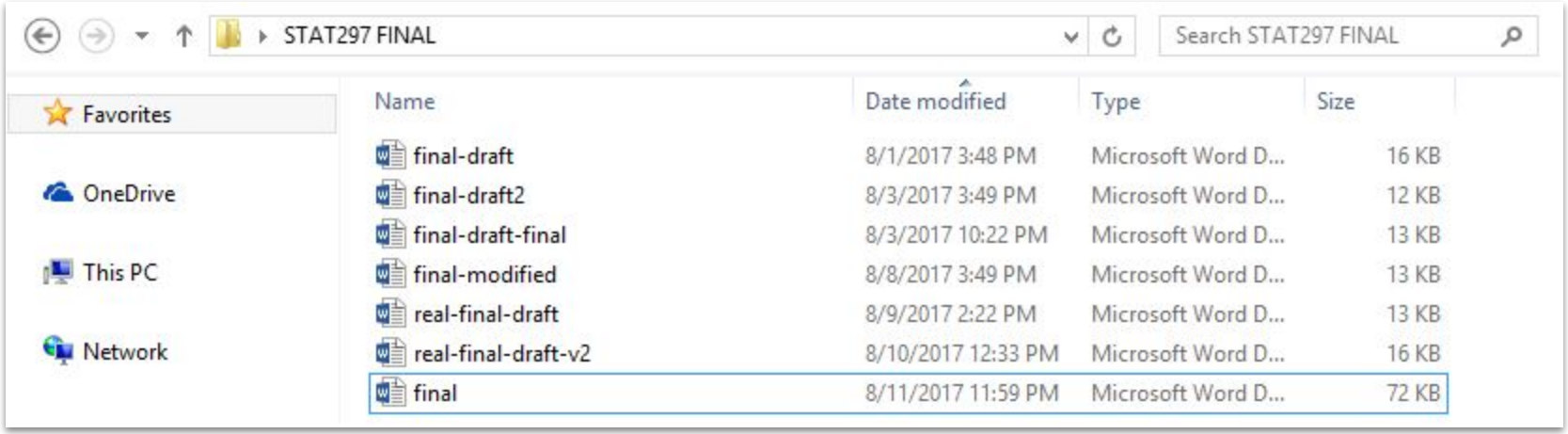
Introducing Git and GitHub

// FLATIRON SCHOOL

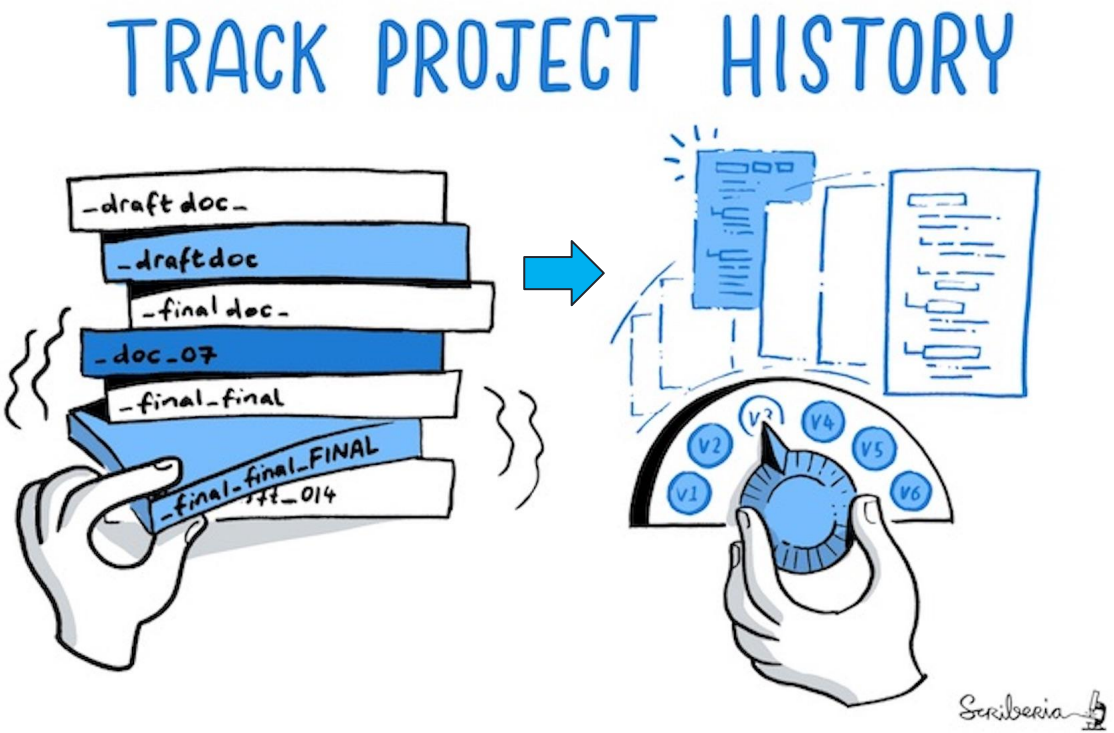
Learning Objectives
(You Will Be Able To)

- **Fork** a repository on GitHub
- **Clone** a repository from GitHub
- **Add** updates to track for a commit
- **Commit** changes (with meaningful messages)
- **Push** local changes to a remote repository
- **Pull** remote changes to your local repository

Look Familiar?



Enter: Version Control



Important Distinction!

git

github
SOCIAL CODING

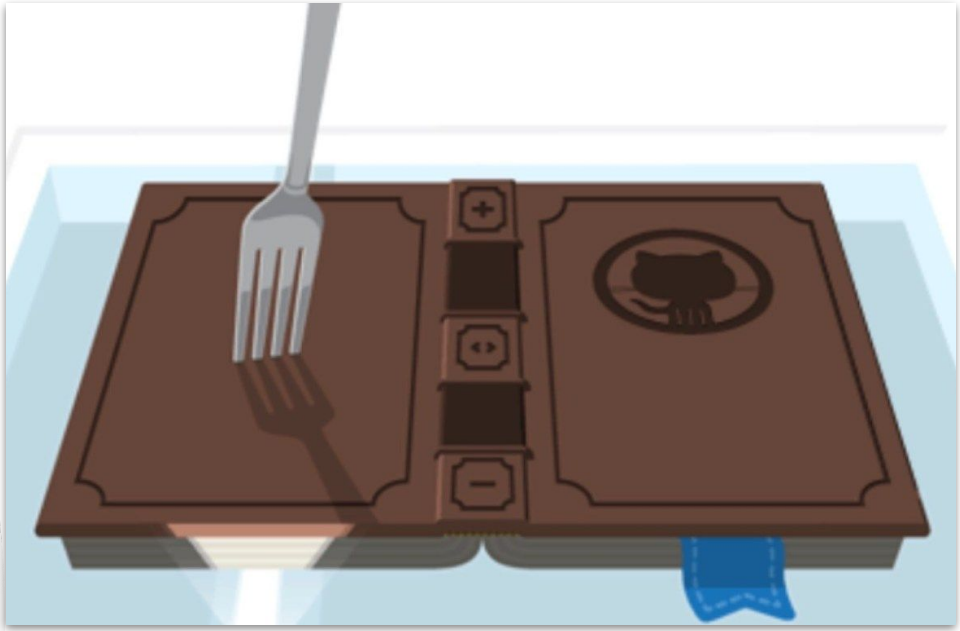
Let's Do This Together

github.com/learn-co-curriculum/dsc-running-jupyter-locally-lab

Fork a Repository

Forking creates your own personal copy over on your own personal GitHub.

- On a fork, you can freely experiment with changes without affecting the original repository you copied from.
- This is the best way to use someone else's repository as a starting point for your own projects!
- If you like, you can later submit those changes to the original repository in order to collaborate.

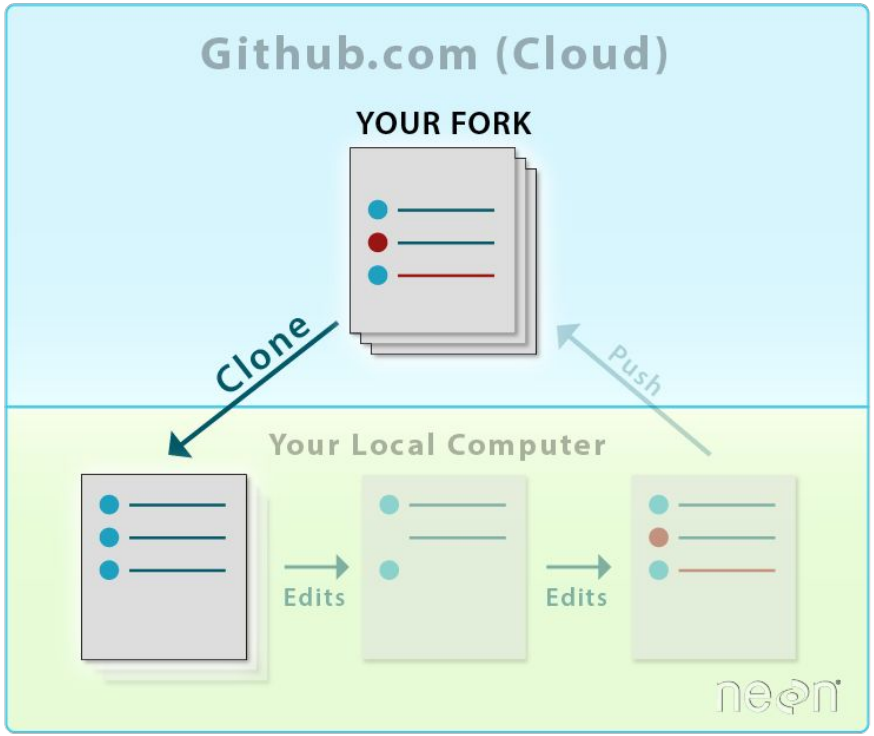


Clone a Repository

Cloning makes a copy of an existing online repository on your local machine.

```
git clone [URL]
```

The difference between forking and cloning is that cloning moves from cloud to local, as opposed to forking which moves code from someone else's remote repository to your own remote repository - all in the cloud.



Make Local Changes

This isn’t a git command - just make local changes to the jupyter notebook, or add a new file to the repository so we can see what it looks like to keep track of changes.

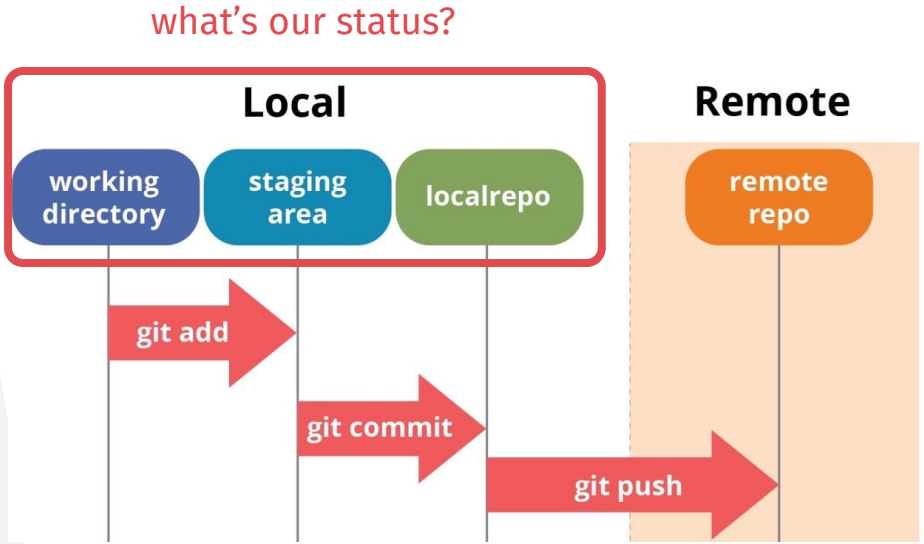
Check Your Status

You made some changes, but what does that look like from Git's point of view? In other words, where are we in the process, on our local drive?

```
git status
```

Git differentiates between staged and unstaged files (as well as tracked and untracked files).

- **Red** represents unstaged files
- **Green** represents staged files

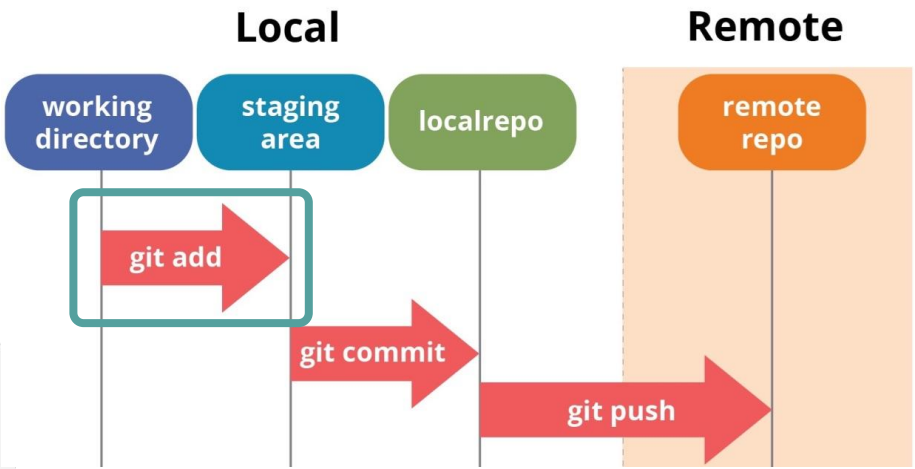


There are more steps in Git than you think you need...

Add Your Changes

Adding files tells Git which changes you'd like to stage, to eventually be committed to your local repository.

```
git add [FILE]
```



Commit Your Changes

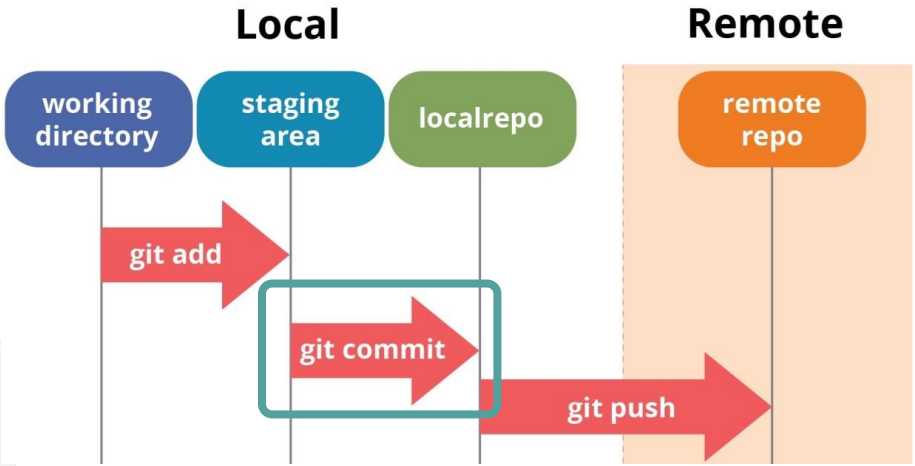
Committing tells Git that these changes shouldn't just be tracked, they're actually ready to be a part of your local repository.

```
git commit -m "[MEANINGFUL MESSAGE]"
```

Commit messages should be written in the present tense, and should finish the sentence:

"If I apply these changes, they should..."

- "Add spell check feature"
- "Fix super awful bug written by David"
- "Update gifs in README.md"



Don’t let this be you! Informative messages only!

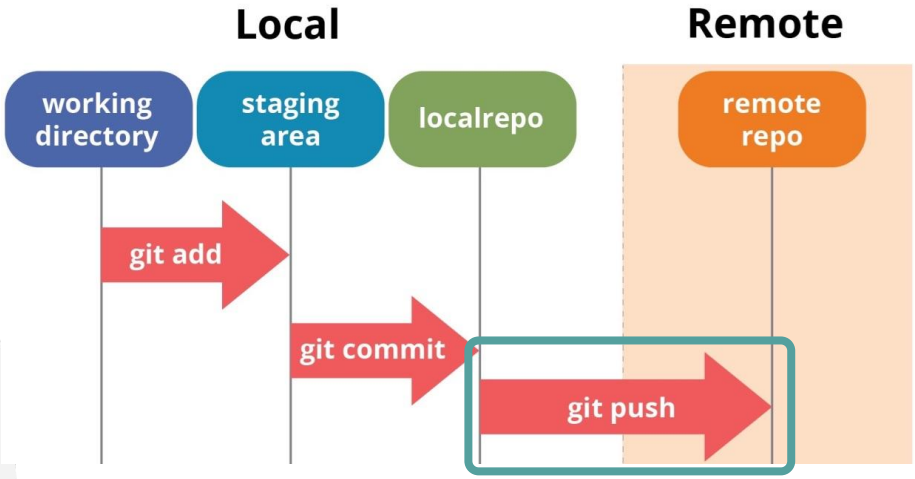
Push Your Changes

Pushing uploads your local repository to a remote repository (say, your GitHub)

`git push [REMOTE LOCATION] [BRANCH]`

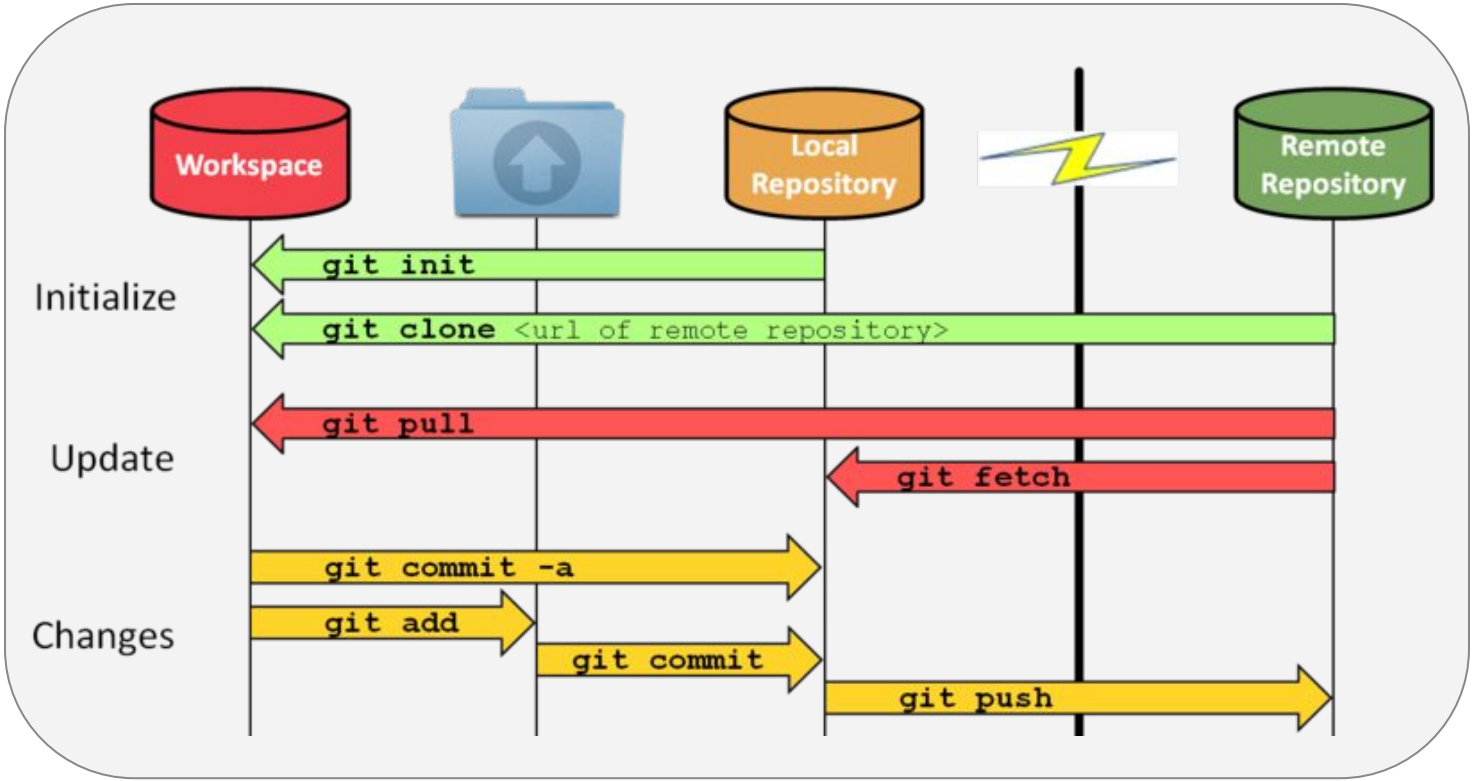
(ex: `git push origin main`)

This is when you connect your local work to your remote repository - it also acts as you backing up your work to the cloud!



Programmer Protocol

A Bigger Picture

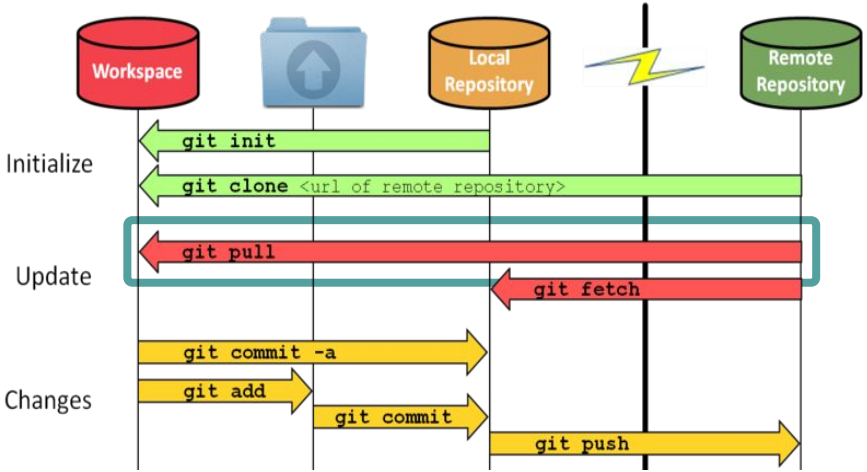


Pull Other Changes

Pulling brings remote changes down to your local workspace.

```
git pull [REMOTE LOCATION] [BRANCH]
```

If there are changes in the remote repository to which you're trying to push changes, you'll need to pull the changes down to your local machine before pushing your work back up.



Progress So Far

What have we covered?

- Starting from an existing repository
- Forking workflow
- Pushing changes

What haven’t we covered?

- Creating your own repo from scratch
- Branching workflow
- Merging changes

Additional Resources

- This blog post is a nicely laid out walkthrough of git
- Git has a whole open-source book on how to use git
- Atlassian has their own version of GitHub, but their tutorial on Git is solid
- Stumbled into a merge conflict? Learn how to resolve it in this blog post
- Git and GitHub For Poets YouTube series

Any Questions?