Introduction to GIT

Why do we use Git?

Imagine that you are working on a large team of data scientist trying to develop a data science application. Some people will work on one part of the application, while others are working on other parts.

If team member A writes a bunch of new code, how do you make sure everyone has the newest version of the code?

Now imagine team member B is also working on changes and they want to give everyone their update. This team member was so focus on their part of the code, they haven't gotten any of the changes from team member A. How do they give everyone their changes in a way they don't lose A's changes?

What is Git?

Git is a type of **version control system** (VCS) that makes it easier to track changes to files. For example, when you edit a file, git can help you determine exactly *what* changed, *who* changed it, and *why*.

Git keeps track of all the changes made to files and then works to reconcile those changes to keep one integrated file and a record of all the changes.

How will we use Git?

1. Deliver the materials like lecture notes and coding challenges not on Learn.co
2. Turn in some of the coding challenges not Learn.co
3. Keep track of the code you write for your projects.
4. Showcase the projects you work on so potential employers can see them.

How will we use Git?

We will not use many of the features of Git.

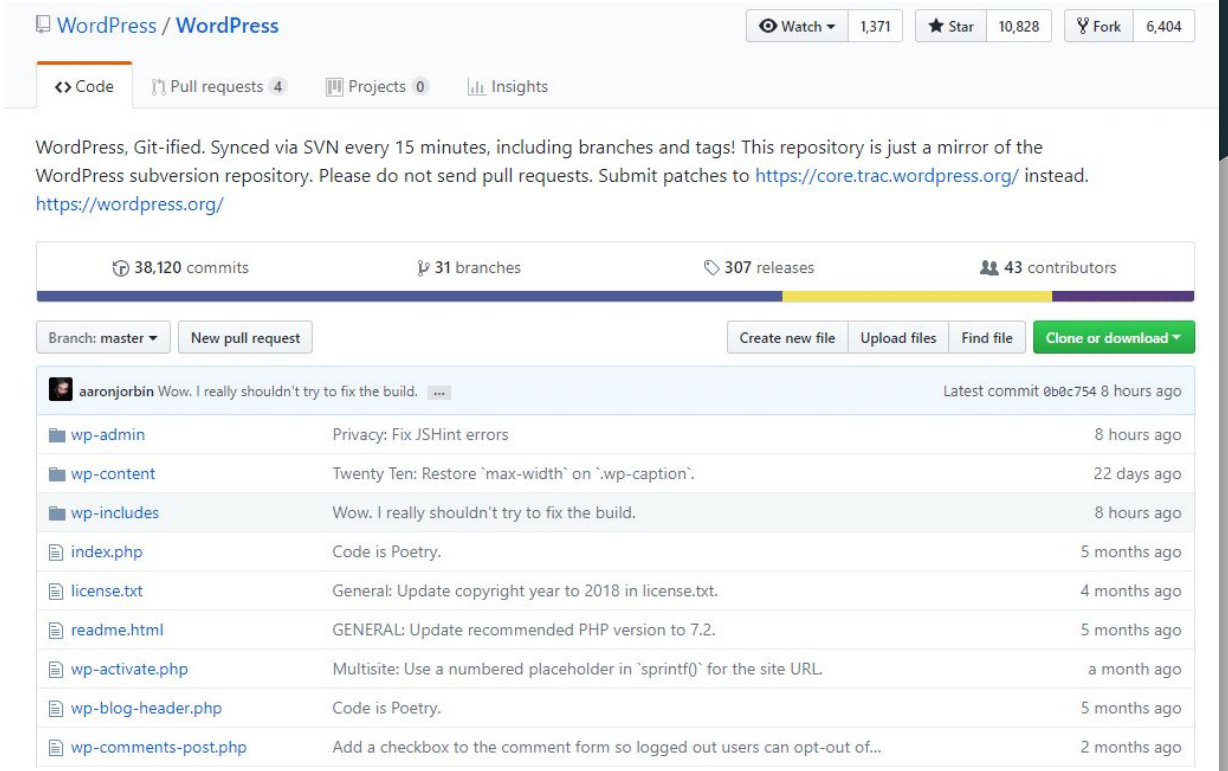
1. You will mainly work with Jupyter Notebooks, which git has a difficult time tracking the changes made.
2. We will not be working on large team projects where we need to manage multiple people contributing to the same code base.

What is GitHub

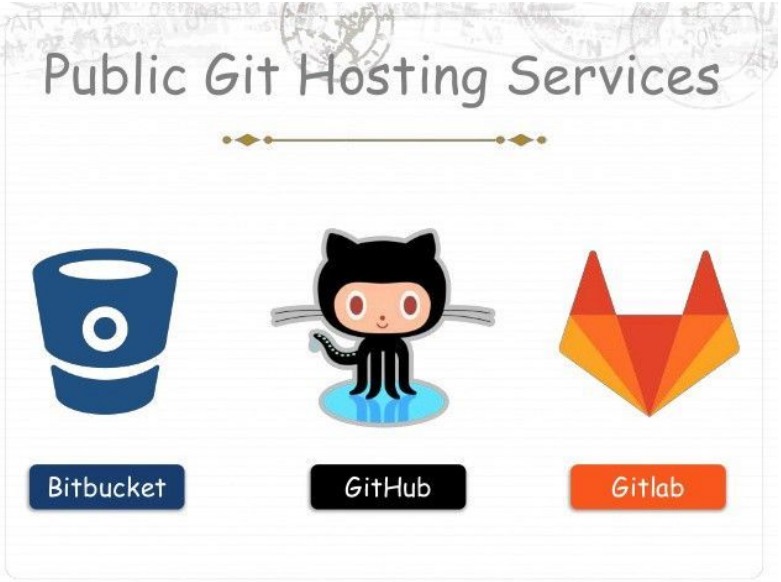
GitHub is a for-profit company that offers a cloud-based Git repository hosting service.

Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration.

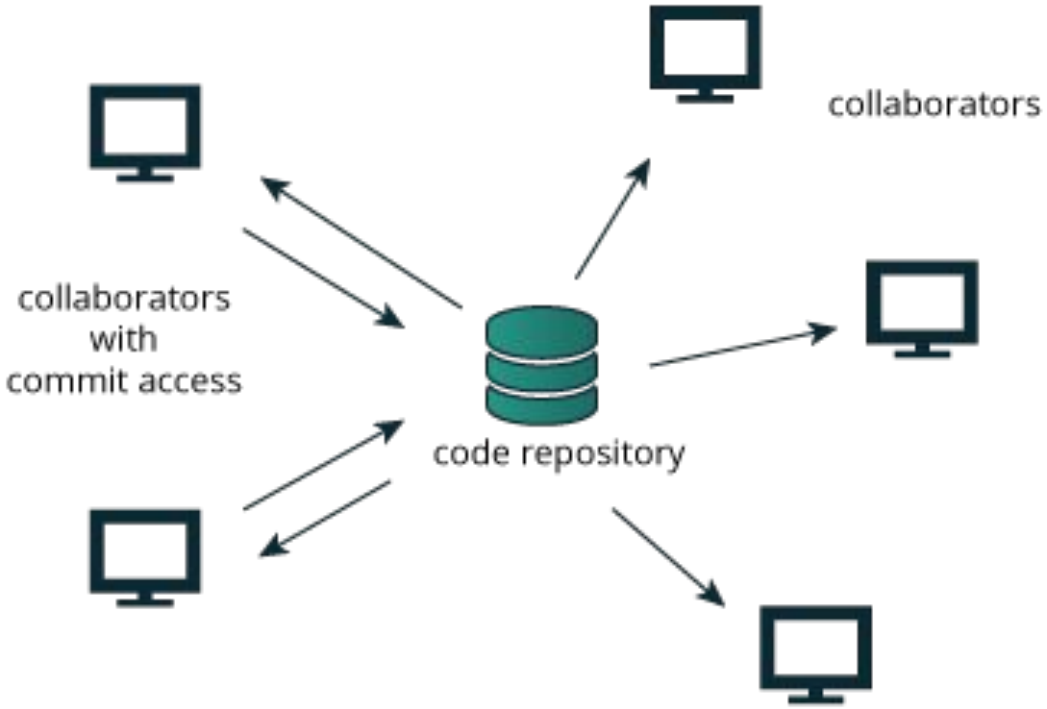
It allows you to put a git repo in the cloud and have everyone pull from and update from there.



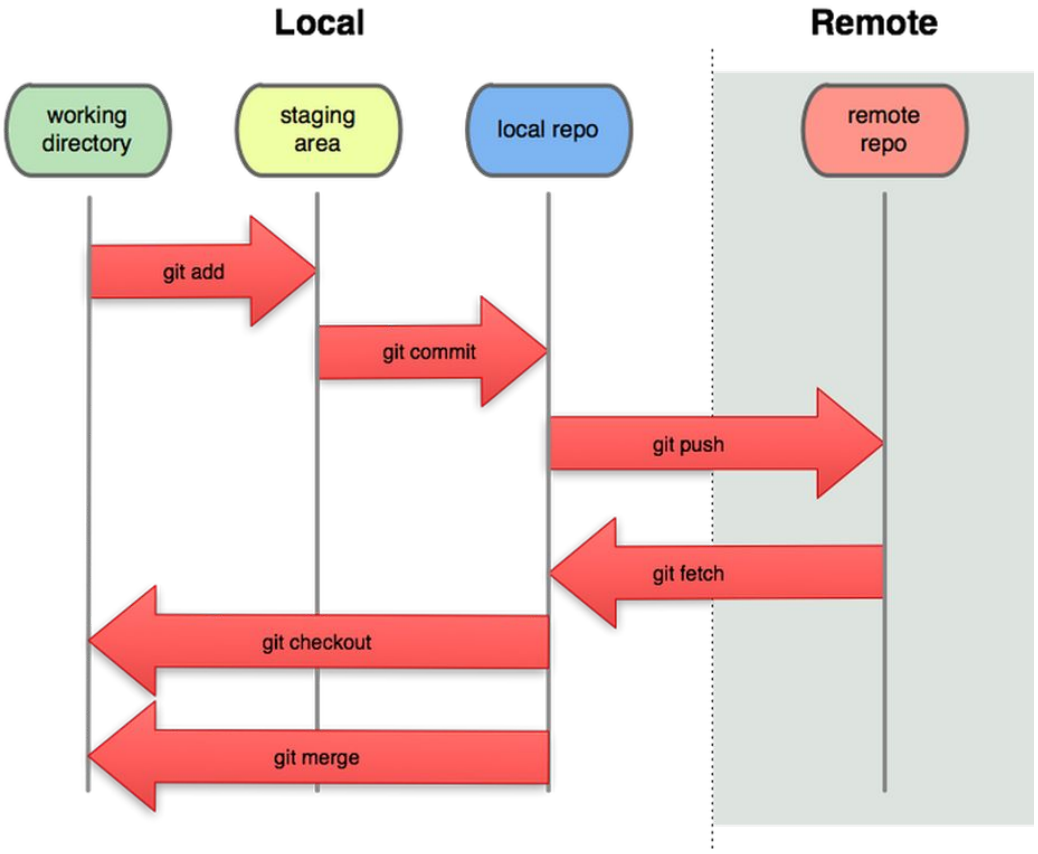
GitHub is not the only git hosting service.



GitHub


collaborators
collaborators
with
commit access
code repository

How Does Git Work?



Starting a Repository

INIT

If you have an existing local directory that you want to *initialize* for version control, use the `init` command to instruct Git to begin tracking the directory. This creates a `.git` directory that contains the Git configuration files.

Once the directory has been initialized, you can add a remote repository and send changes to GitLab.com. You will also need to create a new project in GitHub for your Git repository.

Starting a Repository

Clone a repository

This means downloading a copy of the repository files to your local computer.

For example, if you want to clone out class repository, first copy the clone link. Then, open your terminal, and cd to the location on your local computer where you want to put these files. Type git clone then paste the link as shown below if you want to clone the master branch.

```
git clone https://github.com/JamesOkunlade/old-apple.git
```

Git add and commit

Think of them as capture and save. You can’t save a thing if you don’t capture it first. Hence, the add command should always precede the commit command. You can use the add command to point at the particular file you want to capture in its current state, you use the commit to save a copy of what you captured.

```
git add index.html
```

To capture all the files you will use `git add .` and to capture the current state of a particular file, say index.html, you will have to type

After taking the snapshots, you will then have to commit and save your snapshots to your local repository using the following:

```
git commit -m ‘commit message’
```

Git Commit Workflow

For example:

```
git add index.html
```

```
git commit -m ‘the form feature button created’
```

You can do the two together with the && operator as shown below;

```
git add index.html && git commit -m ‘footer html structure created’
```

GIT Push

Just like saving your snapshots to a Google Photos album for whomever you share the album with, think of git push as sending your local repository to the remote repository for others to access.

```
git push -u origin branch-name
```

Fetch

Fetch the latest info about a repo:

```
git fetch
```

This will download the latest info about the repo from origin (such as all the different branches stored on GitHub).

It doesn’t change any of your local files — just updates the tracking data stored in the .git folder.

Merge

```
git merge <other-branch-name>
```

This will take all commits that exist on the other-branch-name branch and integrate them into your own current branch.

⚠️ This uses whatever branch data is stored locally, so make sure you've run git fetch first to download the latest info.

For example, if someone else adds a few commits to the master branch of origin, you can do the following to download their changes and update your own local master branch:

```
git checkout master      # Make sure you're on the right branch.

git fetch                # Download any new info from origin.

git merge origin/master  # Merge the 'origin/master' branch into your current branch.
```

✨ The name origin/master here literally means the origin/master checkpoint on your computer. Git uses this notation to differentiate branches of the same name (e.g. master) located in different places (e.g. your own branches vs. origin's branches).

GIT Pull

As a shortcut, you can use the **pull** command to both *fetch* and *merge* all in one step. This is more common than merging manually like above:

```
git pull origin master
```

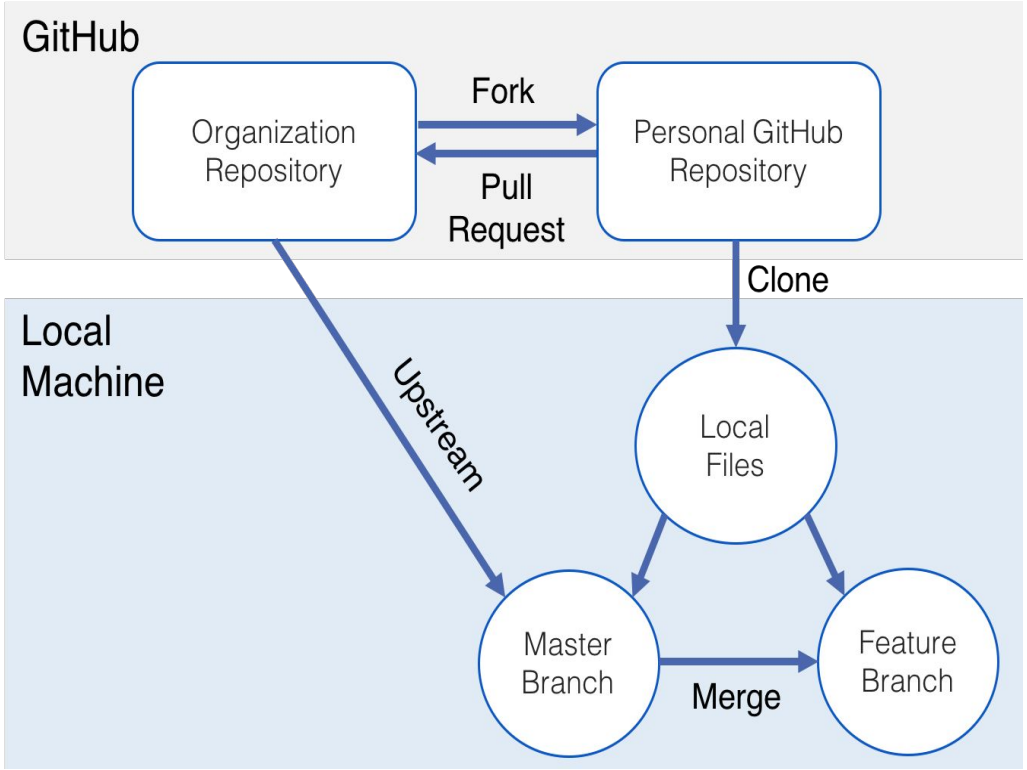
Here we separate the words origin and master (without a slash like above). We don't want to use the origin/master checkpoint on our own computer, because that's stored offline and is probably out of date. We instead want to fetch directly from the master branch of the remote endpoint called origin.

Real Life Example

```
git clone https://github.com/cooperka/emoji-commit-messages.git
cd emoji-commit-messages
git status
git checkout -b my-new-feature
echo “This is a cool new file” > my-file.txt
git status
git add --all
git status
git diff HEAD
git commit -m “Add my-file.txt”
git status
git log
git push origin HEAD
git checkout master
git pull
```

Working with the Lecture Repo

1) Fork the lecture repo to your personal account.
2) Clone down the forked repo.
3) Add the learn-co lecture notes repo as the remote, this step only happens once
4) Check the remote is set and your lecture notes repo is correct. You should see your forked repo after origin, and the learn-co-students repo after upstream



Resources

https://agripongit.vincenttunru.com/

https://juristr.com/blog/2013/04/git-explained/

https://hackernoon.com/understanding-git-fcffd87c15a3

https://www.freecodecamp.org/news/the-essentials-of-git-explained-in-five-minutes-d554019eded9/

https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html