Lecture: https://www.youtube.com/watch?v=5OsPD3LhX8o&list=PLzOJMOiC_x7foeOOFmvNu1K-pFlZi6idu

> What is Git?
> Git is a version-control software that is used in software development to maintain records of changes made to text-based files

Functions of Git

- **Staging** files for a local repository
- **Committing** files to a local repository
- **Cloning** files from a remote repository
- **Pushing** files to a remote repository
- **Pulling** files from a remote repository
- **Branching** - creating different paths
  - For example, master/dev/topic or dev/test/prod
- **Repositories (or "repos")** can be either public or private and users can be given read or write privileges
- **Team collaboration**: tracks authorship, history, and use **annotated tagging**
- Continuous Integration and Continuous Delivery (CI/CD)

> What is a commit?
> Staging **MUST** be done **IMMEDIATELY BEFORE** committing, **EVERY** time
> **To create a commit, Git **requires** information about the committer (name/email) and a message contributed by a the committer
> Each commit receives its own unique SHA-1 Hash, which is derived from:
>
> - Who made the commit
> - Changes made to files
> - Files added
> - Messages attached

PUSHING: Send files prepared inside your local repository to a remote repository
CLONING/PULLING: Clone copies of a remote repository to use locally
BRANCHING/MERGING: You merge 2 branches, creating a new commit which is the child of two commits - one from each branch. However, the branches themselves remain.
**Any differences in a file shared by both parents will result in a merge conflict that you must resolve.

Common Git commands

- `git init` : Init a repo
- `git add filename.ext` : Stage a file
- `git add .` : Stage all files in the current folder
- `git status` : Display current status in git
- `git commit -m "Commit message"` : Commit w/message
- `git tag -a tagLabel -m "tag annotation"` : Create an annotated tag with message
- `git tag tagLabel` : Create a tag without message
- `git checkout commitID` : Checks out a commit. commitID can be either tag or SHA1 id
- `git clone <url>` : Clone a remote repo
- `git push origin master` : Pushes the "master" branch to the "origin" remote (must sign in to write)
- `git remote add remoteHandleName url` : adds the remote repo
- `git pull remoteHandleName branchName` : pulls from the remote repo
- `git branch branchName` : creates a new branch
- `git checkout branchName` : Checks out a branch
- `git merge branchName` : merges branchName into the current branch
- `git log -10` : displays the last 10 commits on the current branch

- `git log --all --oneline --graph --decorate`: View a graph of commits on different branches

Continuous Integration & Continuous Delivery (Deployment)

Continuous integration: A software development practice where members of a team use a version control system and frequently integrate their work to the same location, such as a main branch
Continuous Delivery: A software development methodology where the release process is automated
Git repo pushes can trigger changes in, for example, a running web app.
Certain tools (I.e. AWS CodeBuild or Jenkins) will even allow you to add automated testing stages in between the commit and the deployment. Example: test your that your webpage contains a banner image.
Read about AWS Elastic Beanstalk, CodeBuild, CodeDeploy, and CodePipeline for more details.

Git Demo: https://www.youtube.com/watch?v=EdiElyHWxSM&list=PLzOJMOiC_x7c7oJ7-C2lDAvLWx_090dLL

1 Adding and Committing

```
git init -> Start a local repo
touch first.txt
git add . # Add first.txt to repo
git commit -m "First Message" # Commit

touch second.txt
git add .
git commit -m "Second Message"

git log # Logs
git log --oneline
git log --all
git checkout <hash> # You get a snapshot of the commit you specify
git checkout master # Go to master branch

echo something > second.txt
git add .
git commit -m "Third Commit"
```

2 Branching and Merging

Creating a new branch

```
git branch newBranch
git checkout newBranch
```

Merging a branch

```
git merge newBranch
git log --oneline --graph --all
```

> In my case, when I did `git merge newBranch` I was told to create a commit message in a vim editor. And you know how the hardest part of vim is getting OUT, adding a commit message would probably help

3 Tagging

```
git tag Version6
# Create an annotation for the current commit
# You can using checkout + annotation instead of hash
```

4 Pushing

```
git remote add origin <URL>
git push origin master
```

5 Cloning

```
git clone <URL>
```