Git and Github

A developer’s best friend

What is Git?

What is Git?

- Git is a Version Control System (VCS) designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams

What is Git?

- Git is a Version Control System (VCS) designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams
- It allows you to see changes you make to your code and easily revert them.

What is Git?

- Git is a Version Control System (VCS) designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams

- It allows you to see changes you make to your code and easily revert them.

- It is NOT GITHUB!

Ok, then what is Github?

- **Github.com** is a website that hosts git repositories on a remote server

Ok, then what is Github?

- **Github.com** is a website that hosts git repositories on a remote server
- Hosting repositories on Github facilitates the sharing of codebases among teams by providing a GUI to easily fork or clone repos to a local machine

Ok, then what is Github?

- **Github.com** is a website that hosts git repositories on a remote server

- Hosting repositories on Github facilitates the sharing of codebases among teams by providing a GUI to easily fork or clone repos to a local machine

- By pushing your repositories to Github, you will pretty much automatically create your own developer portfolio as well!

Confirm that you have git

- Open your terminal and run ‘git’
- If you see a ‘command not recognized’ error, you probably haven’t installed git yet.
- We can solve your issue when the lectures are over - sit tight for now!

Configuring git

Your commits will have your name and email attached to them. To confirm that this information is correct, run the following commands:

```
$ git config --global user.name
> should be your name, i.e. Jon Rosado
$ git config --global user.email
> should be your email, i.e. jon.rosado42@gmail.com
```

To fix either, just add the desired value in quotes after the command:

```
$ git config --global user.name “Jon Rosado”
$ git config --global user.email “jon.rodado42@gmail.com"
```

Using Git / Github

Connecting git with Github

- In order to prevent having to enter your password each time you push up to Github, you must configure git and Github to recognize Secured Shell (SSH) keys that you generate.

- To check and see if you have any recognized SSH keys active on Github, go to https://github.com/settings/keys

- If you do not see any SSH keys listed, you do not have SSH configured with Github. We can assist with this later.

- If using Mac OS X, you can also configure your keychain to automatically enter your password via HTTPS.

Connecting git with Github

- From your project directory, run `git init` to initialize a git repository.

- Go to Github, and create a new repository with the name of your project.

- Follow the instructions on Github to connect your initialized git repository to the remote server on Github.

- *Please Note: you must have files in your project directory to commit in order to push anything to your remote server.

Basic git / Github workflow

- Stage edits to be committed to your git repository by using `` `git add <file name>` `` to track the files that you want to add directly or `` `git add .` `` to add all files at the current directory level that you have worked on.

- Commit changes using `` `git commit -m <message>` `` and be sure to leave a short but descriptive message detailing what the commit will change when merged to the master branch.

- Push changes to save them by using `` `git push origin <branch name>` ``

git branching

- `git branch` to view current branches in repo
- `git checkout -b <branch name>` to create a new branch with branch name
- `git checkout <branch name>` without ‘`-b`’ flag to switch to existing branches

git merge: pros

- Generally the easiest option to merge your master branch into your current working feature branch.
- You can `git checkout feature` and then `git merge master` or you could just do it with one command: `git merge feature master`

Merge conflicts

- Merge conflicts arise when two members of the same development team work on the same file and try to merge in their respective changes.

- The proper way to avoid merge conflicts would be to ensure that only one branch is fully committed, pushed, and merged to master, allowing the other branch to integrate any changes before attempting to push and merge to master.

- If merge conflicts arise, don’t fret! Many text editors (as well as Github) provide tools to help track down conflicts and resolve them. Often times, it will show incoming changes juxtaposed with the current state of your file, and allow you to choose which to keep (one or both).