Introduction to Git and GitHub

Team 694, Software Engineering

What is Git?

- Git is a **version control system.**

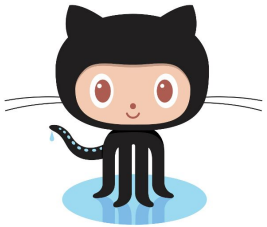
It allows you to **collaborate on big projects** with others by **easily managing different versions of the code** and allowing people to work on different parts of the project at the same time.

Essentially, it’s like Google Drive with many more perks and great integration with coding.

What is GitHub?

- The most widely used **web-based** version-control and collaboration platform for software developers in the US
- a Git repository hosting service

In other words, GitHub is a website for hosting projects that use git



GitHub

This is Octocat,
GitHub's mascot

Git Commands

`$ git status` - **List the changes you have made, and which you have staged**

`$ git log` - **Show past commits**

`$ git add <file>` - **Stage changes made to the file for commit**

`$ git commit` - **Commits the changes you have staged**

`$ git push` - **Upload your local commits to the online repository**

`$ git pull` - **Updates the current branch**

`$ git clone <link>` - **Downloads the repository at the link you entered**

These are the commands you'll use most (add --help for more information)

Why is ~~Gamora~~ Git?

Git is composed of 3 different storage systems

- Working directory
  - Where the files go when you hit ctrl + s
- Staged Repository
  - Local repository stored on computer
- GitHub Repository
  - Online repository that everyone else can access

Cloning

Cloning an existing repository is probably what many of you will be doing, since our repositories are already made.

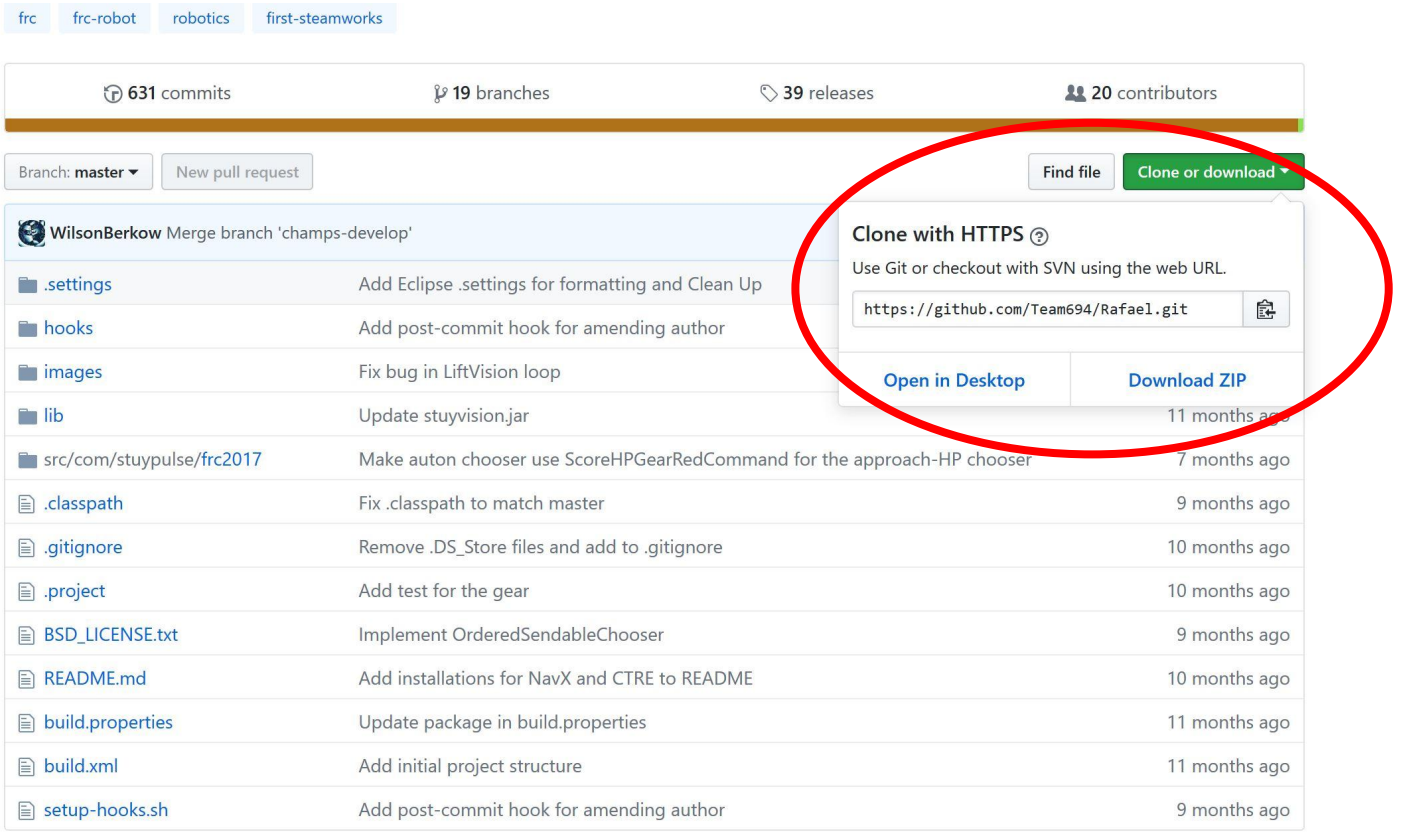
The following command is used:

- `$ git clone [link to repository]`

This will **copy the repository into a folder that has all the files into the current directory you are in**.

Creates the staged repository and the working directory

Cloning a Directory



Cloning Activity

Go to https://github.com/huiminwu/newbie-ed-school-2019

and copy the URL

On Git Bash, type :

```
$ git clone https://github.com/huiminwu/newbie-ed-school-2019
```

Make sure you cloned the repository by typing `$ ls`

Change directory into newbie-ed-2019 using `$ cd newbie-ed-2019`

Pulling

Pulling from the online GitHub repository **allows the puller to obtain the latest changes** from the online repository and apply them to the repository on the puller's computer (this is a combination of the git commands fetch and merge).

Pulling from an online repository:

- `$ git pull`

Applies to working directory. Does not overwrite files that have been edited.

Adding

After you've made your changes, you have to **add** them.

This means that you're preparing to move your changes from the working repository to the staged repository (preparing to **commit**).

This is done with:

- `$ git add <file>`

***** To check what changes you have before adding, do:

- `$ git status`

Adding Activity

Change directory into the Activities folder using **`$ cd Activities`**

Create a file named <your_name>.txt (hint: **`$ notepad <your_name>.txt`**)

In the file, put your name, grade, your GitHub username, and an ASCII art of your choice

Save the file and check on Git Bash using **`$ git status`** (filename should be in red)

Add the file using **`$ git add <your_name>.txt`**

Make sure using **`$ git status`** that the file is now listed in green

Commits

Git works with a system made up of **commits**. Whenever you edit or add a new file to a repository, you must commit your changes.

Commits should be followed by a commit message, which is noted with a flag following the command as shown:

- `$ git commit -m "Add commit message"`

NOTE that commit messages are in **PRESENT TENSE** and **THE FIRST WORD IS A VERB.**

Eg. Moves the changes in the working directory to the staged repository

Commits (Continued)

If you're on one of the SE computers, make sure to type this in first:

```
$ git config --global --unset user.name
```

```
$ git config --global --unset user.email
```

Additionally, to commit, use this if you're using a computer that isn't yours

```
$ git -c user.name='<your username>' -c user.email='<your email>'
commit -m '<commit message>'
```

- THE EMAIL MUST BE ASSOCIATED WITH A GITHUB ACCOUNT!

If you're using your own computer/account: `$ git commit -m "message"`

Pushing

Committing something is not enough to just make the changes appear on an external git server. You must **push those changes to store them in the cloud.**

Pushing is a very simple process that consists of the following command:

- `$ git push`

After this, you log into your Git account (to confirm you are allowed to add to a repository) and the push should commence!

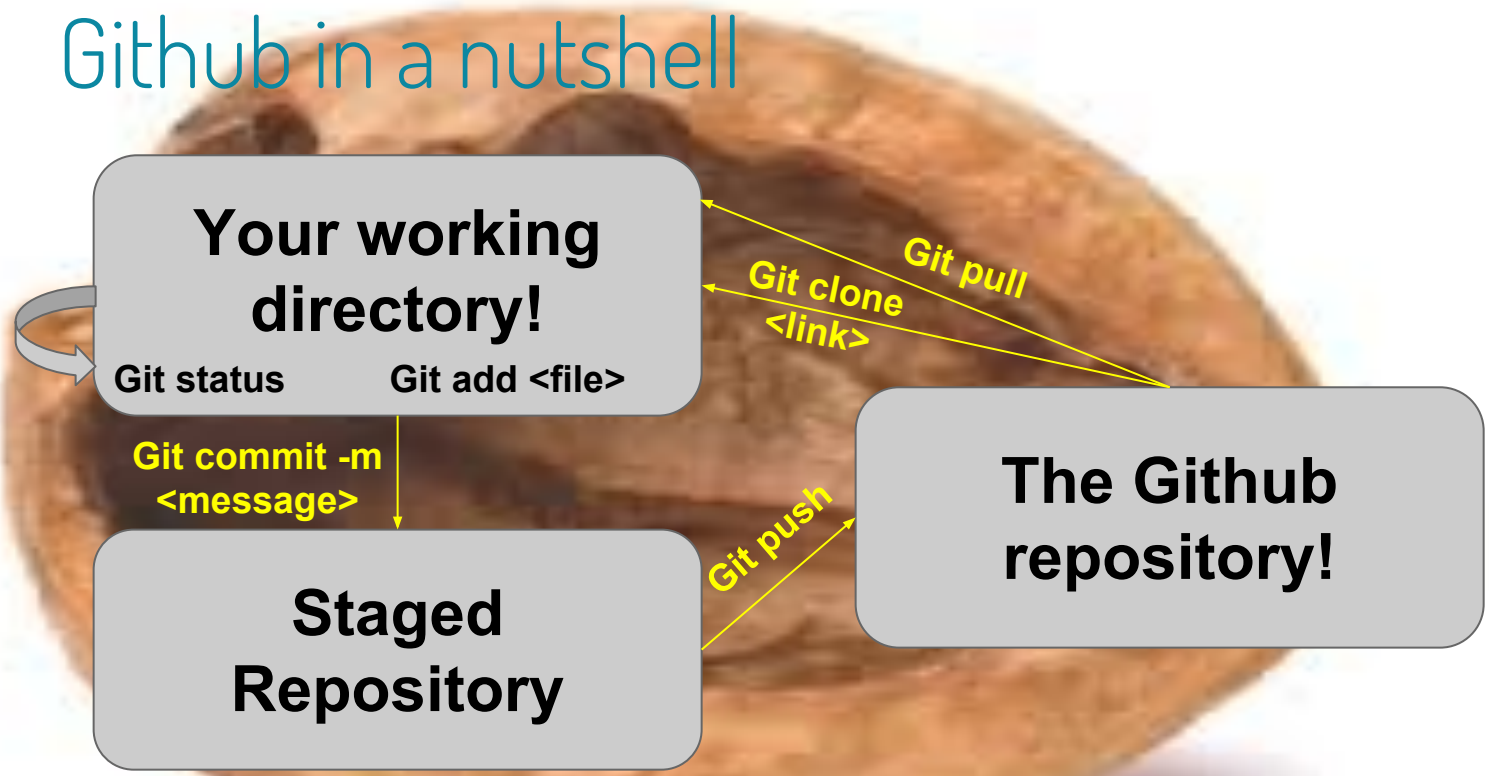
Always pull before you push.

Committing and Pushing Activity

Commit and push the file you added before with your name as the commit message.

(Yes, I’m making the font really big to make this page look less empty.)

Github in a nutshell



Local branches

Local branches are **branches that can only be viewed by your machine**.

`git branch` : when run with no arguments, it simply lists all of your local branches:

- **`$ git branch`**

```
  iss53
* master
  Testing
```

- The * is used to **indicate a branch you are currently on**.

Creating a new branch

$ git branch <branchname> : This **creates a new branch**

So say I want to create the new branch named stuypulse, you would type:

```
$ git branch stuypulse
```

*** THIS DOES NOT CHECK OUT THE NEW BRANCH! ***

So now, when you do "**$ git branch** ", you get:

```
$ git branch
  iss53
* master
  Testing
  stuypulse
```

Pushing branches

```
git push origin [branch name]
```

If the branch is new, you must tell the repository to create it:

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

Git checkout

When wanting to switch branches, there are two options:

- `git checkout <existing-branch>`
  - This makes `<existing-branch>` the current branch
- `git checkout -b <new-branch>`
  - This creates **`<new-branch>`** and then makes it the current branch.
  - Basically, it's **`git branch <new-branch>`** combined with **`git checkout <new-branch>`**.
- `git push -u origin <branch>`

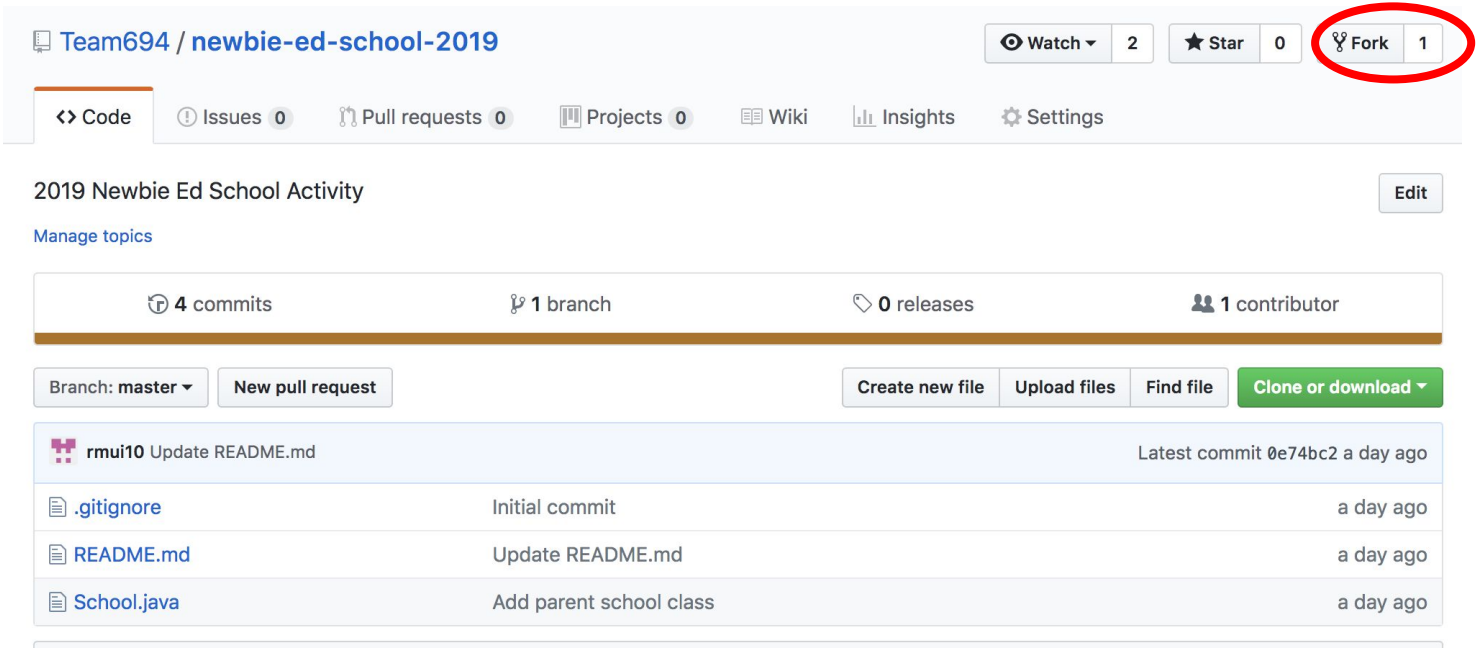
Deleting a branch

```
git branch -d [branch name]
```

Remember, you must be out of no-longer-wanted branch to delete it!

Forking

- Forking copies a repository from another account to your account



Arrays, Method Overloading, and this

Arrays

Arrays are how you store multiple values of the same data type. Eg. you can store all of the numbers of the fibonacci sequence together into one variable. You can use arrays for any kind of data type. To store a data type as an array, add “[]” after it

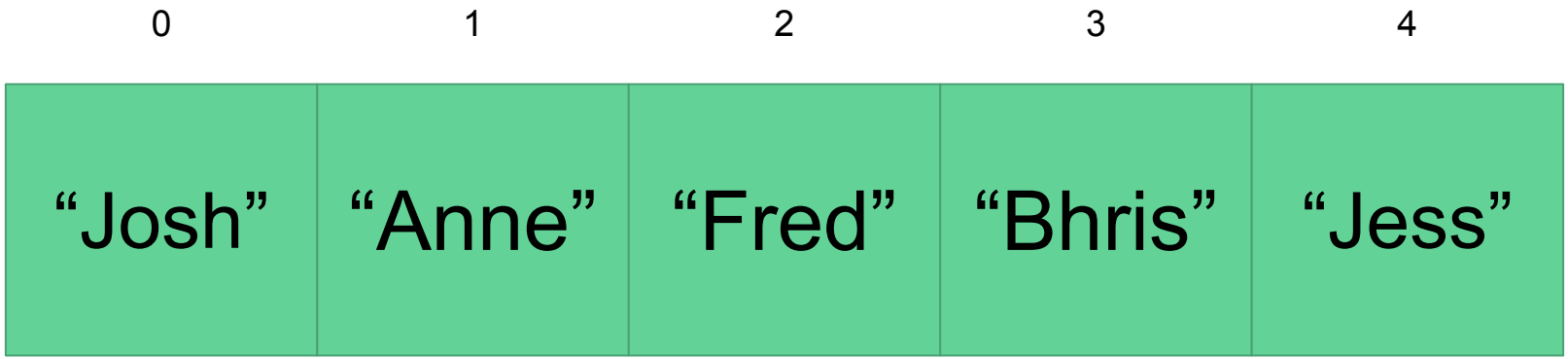
```
int[] anArray; //Declares an array of ints called anArray
```

```
Food[] shoppingList; //Declares an array of instances of foods that make up a shoppingList
```

Intro To Indexes

Every time you add a value to an array, it is stored as an index, the first value you store is assigned an index value of 0, then 1, then 2. Use these indexes to identify the values that you input when you call them later on. Each index corresponds to a different value, so that only one value can be assigned to each index.

```
String[0] seKoolKidz = “Josh”;
System.out.println(seKoolKidz[0]); //prints “Josh”
```



Syntax

```
String[] seKoolKidz = new String[2]; //String now has 2 indexes

seKoolKidz[0] = “Josh”;

seKoolKidz[1] = “Coolmin”; //Array is now full

Presidents[] badOnes = {
    new Presidents(“Donald J. Trump”),
    new Presidents(“Andrew Jackson”),
    new Presidents(“Richard Nixon”)
}; //Note different syntax, and use of a different data type
```

Method Overloading

Method overloading is when you have multiple methods all with the same name, but with different parameters. The difference is that the parameters must have different types, or that there are a different amount of parameters.

```
println(int n) {...}

println(double n) {...}

println(String n) {...}

println(int j) {//error because the data type is the same}

println(int n, int n) {...}
```

This (as in this, not this)

`this` refers to the current class. You cannot use `this` within non-static methods. It can be used to refer to variables, and call the constructor. Methods shown below are both constructors for Manhattan class.

```
int x;
public Manhattan(int x) {
    this.x = x;
}
System.out.println(x);
```

```
int coolVarName;
public Manhattan(int x) {
    coolVarName = x;
}
System.out.println(coolVarName);
```

```
//Both examples shown print out the same value
```

Using this For Constructors

```
public class PokemonGenerator {
    int statTotal; String name; String type;
    public PokemonGenerator(String name,int statTotal,String type) {
        this.name = name; this.type = type;
        this.statTotal = statTotal; //this constructor does stuff
    }
    public PokemonGenerator(String name) {
        this(name, 425, “Normal”); //refer to bottom constructor
    }
    public PokemonGenerator(String name, int statTotal) {
        this(name, statTotal, “Normal”); //refer to next constructor
    }
```

Activity

Create an array that has 3 indexes, and store the PokemonGenerator class within the array. Each time you use the PokemonGenerator class, you should use a varying amount of parameters in the object. For extra credit, name the pokemon in the photo below

Project

- Split up into groups of 4 (this will probably be the people sitting in the same row)
- Fork the newbie-ed-school-2019 repository from https://github.com/Team694/newbie-ed-school-2019
- Work on the project described in the README.md file
- Create a branch for each class
- Send a pull request to the Team694 repo when you’re done