# 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

This is Octocat, GitHub's mascot

**GitHub**

# 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

frc    frc-robot    robotics    first-steamworks

631 commits          19 branches          39 releases          20 contributors

Branch: master ▾    New pull request                                          Find file    Clone or download ▾

WilsonBerkow Merge branch 'champs-develop'

| | | | Clone with HTTPS ⑦ |
|---|---|---|---|
| 📁 .settings | Add Eclipse .settings for formatting and Clean Up | | Use Git or checkout with SVN using the web URL. |
| 📁 hooks | Add post-commit hook for amending author | | https://github.com/Team694/Rafael.git 📋 |
| 📁 images | Fix bug in LiftVision loop | | |
| 📁 lib | Update stuyvision.jar | | Open in Desktop          Download ZIP |
| 📁 src/com/stuypulse/frc2017 | Make auton chooser use ScoreHPGearRedCommand for the approach-HP chooser | | 11 months ago |
| 📄 .classpath | Fix .classpath to match master | | 9 months ago |
| 📄 .gitignore | Remove .DS_Store files and add to .gitignore | | 10 months ago |
| 📄 .project | Add test for the gear | | 10 months ago |
| 📄 BSD_LICENSE.txt | Implement OrderedSendableChooser | | 9 months ago |
| 📄 README.md | Add installations for NavX and CTRE to README | | 10 months ago |
| 📄 build.properties | Update package in build.properties | | 11 months ago |
| 📄 build.xml | Add initial project structure | | 11 months ago |
| 📄 setup-hooks.sh | Add post-commit hook for amending author | | 9 months ago |

# 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/Team694/newbie-ed-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 <u>fetch</u> and <u>merge</u> ).

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.

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)

Additionally, if you want to add your name to the end of a commit on a computer you aren't logged in on, just do the following

`$ git commit --author="First Last <email@email.com>" -m "Commit message"`

- THE EMAIL MUST BE ASSOCIATED WITH A GITHUB ACCOUNT!

# 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!
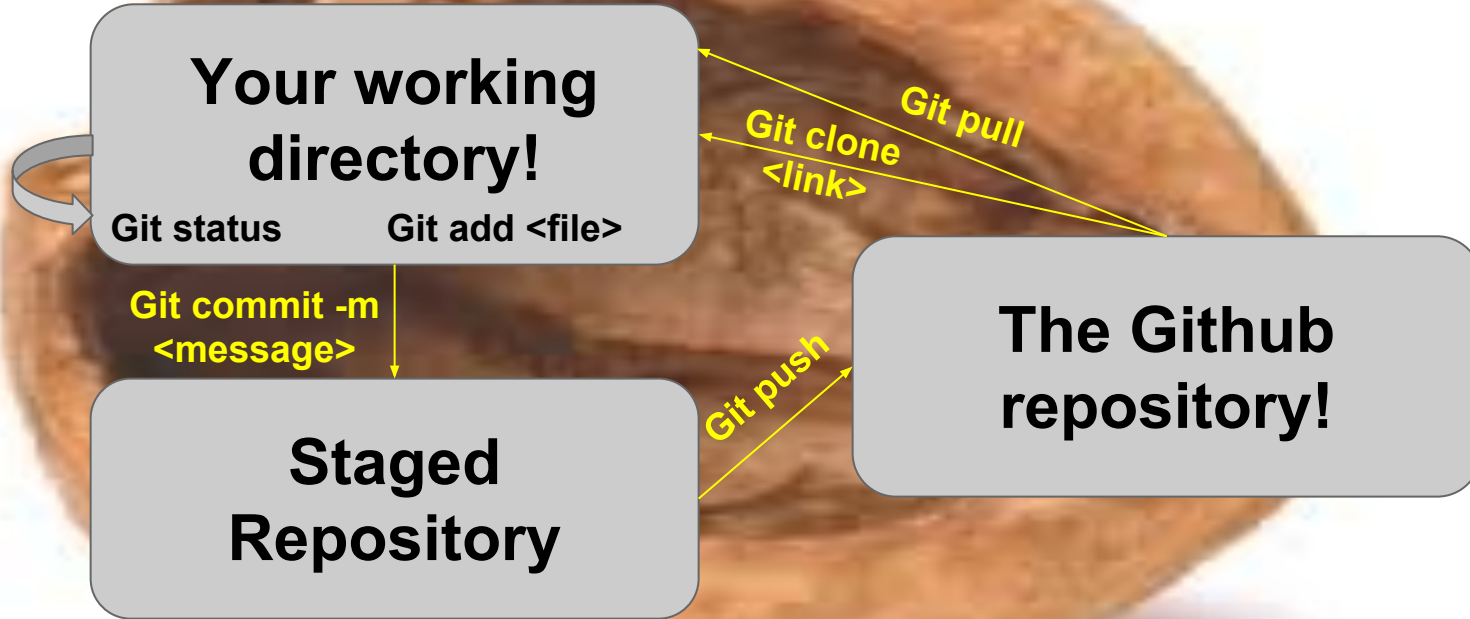
**Always pull before you push.**

# 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

**Your working directory!**

Git status          Git add <file>

Git commit -m <message>

**Staged Repository**

Git clone <link>

Git pull

Git push

**The Github repository!**

# 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!**