# Git and Github

## Introduce the problem to be solved

1) Developers have programming projects in a directory structure:

```
/RobotErnie
     /src
          file1.java          file2.java
     /bin
```

2) Many people manage multiple versions by creating multiple copies of the directory structure

```
/RobotErnieOriginal
     /src
          file1.java          file2.java
     /bin

/RobotErnieBack
     /src
          file1.java          file2.java
     /bin

/RobotErnieCompetition
     /src
          file1.java          file2.java     file3.java
     /bin
```

- The problem is it is hard to remember what changes were made to each version (folder)
- If you make a change to one folder structure, it is hard to know if the change made it into other folders/versions.

## What is GIT?

**Git** is a distributed version control system

- Records changes to files over time – **in one folder**
- Ability to retrieve files at certain point in time (backup)
- Work on new features (experimental) without messing up the "master" code. (i.e. branches)
- Easily collaborate with other developers

**GIT and GITHUB are very popular/common within software development. Learn it!

## Install Git

Install Git (all platforms): https://git-scm.com/downloads

Another option for Windows… http://cmder.net/

"cmder" is a console emulator for Windows, and it includes GIT

Git and GitHub tutorial (YouTube), 12 videos (5-10 min each): https://www.youtube.com/watch?v=3RjQznt-8kE&index=1&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR

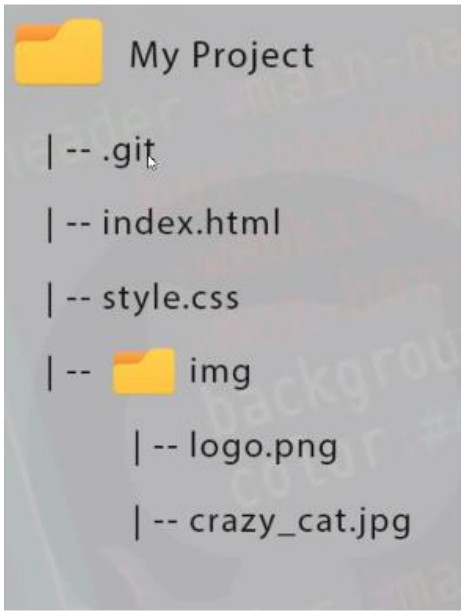
## Initialize Git

```
git –version

// set user.name and user.email
git config –global user.name tomharron
git config –global user.email tom.harron@school.com

// display my user name
git config user.name

// initialize git for one of your projects.
// this makes create a git repository
cd /your/project/folder
git init


// show all file (even hidden files)
ls -a
```

## What is a Repository?

This creates a **.git** folder in your project root folder (wherever you ran the "git init" command).

This **.git** folder is your REPOSITORY (or "Repo)". A Repository is a **project folder** where all the changes you make to any of the files in your folder or sub-folder are stored and tracked.


```
My Project
|-- .git
|-- index.html
|-- style.css
|--  img
      |-- logo.png
      |-- crazy_cat.jpg
```

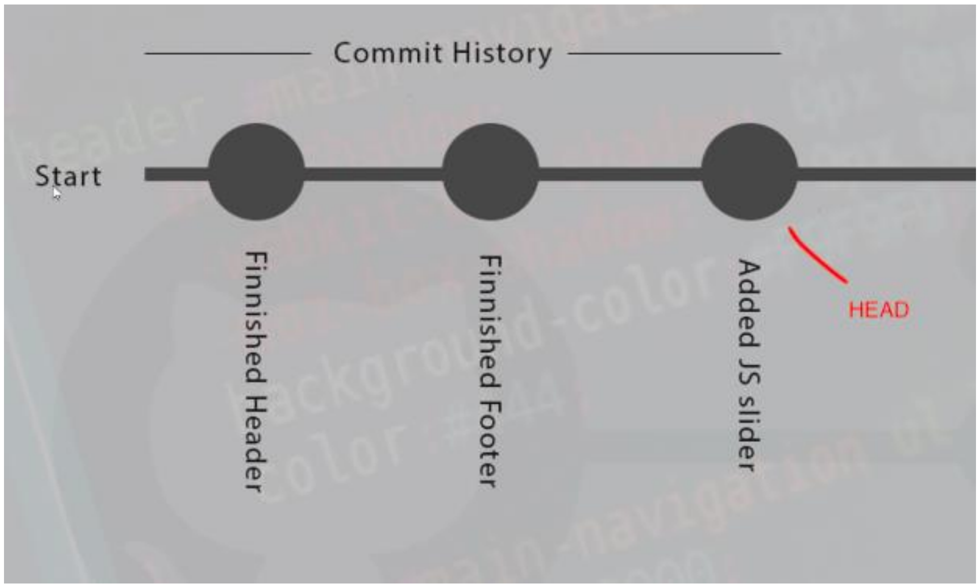
## Using GIT

### Basic commands

```
    // show status of your files
    git status

    // TODO:  Make a code change to GoodRobot.java

    // display the changes
    git status

    // TODO:  Undo code changes, show with git status
    // TODO:  Redo the code changes

    // add this file to staging area (. means "all")
    git add .

    // commit the change to the repository
    git commit -m "added welcome message"


    // TODO:  Add another welcome message

    // add this file to staging area
    git add GoodRobot.java

    // commit the change to the repository
    git commit -m "added 2nd welcome message"
```

### What is a commit?

A "commit" is essentially a save point. It is a point in time where you want to save your changes. GIT will track the changes for each commit and you could rewind to a previous commit. This would revert ALL files associated with your project to that point in time!



## Modified, Staged and Committed files

**Modified:** When you edit a file, git notices that and considers it "Modified." To see this, you can issue the command:

```
    git status
```

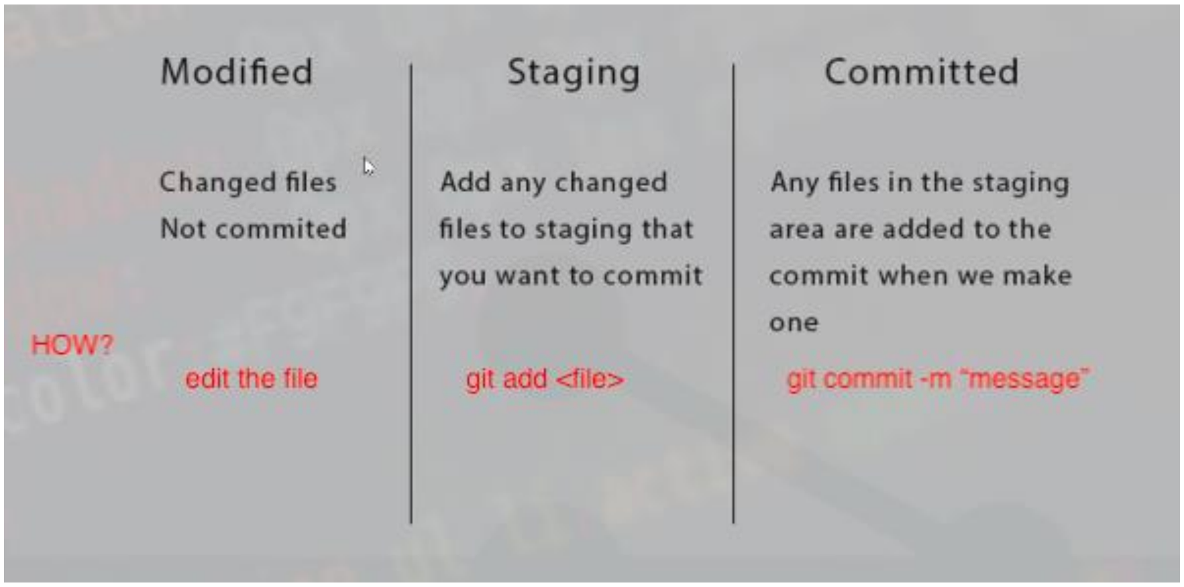
**Staged**: Once a file (or files) are modified, you can add them to the staging area. To do this, you issue one of these commnds:

```
    git add <filename>       // add a specific file to the staging area
    git add .           // add all modified files to the staging area
    git status               // see files in the staging area
```

**Commit**: To commit your change(s) to the repository issue a commit command:

```
    git commit -m "message"
```

- A commit creates another "save point" that you could go back to.


Modified
Changed files
Not commited
HOW?
edit the file
Staging
Add any changed files to staging that you want to commit
git add <file>
Committed
Any files in the staging area are added to the commit when we make one
git commit -m "message"

## Git add, commit and log

// no changes
```
git status
```

// **TODO** – change a file, see that it is modified
```
git status
```

// add the staging area
```
git add GoodRobot.java
```

// remove the file from the staging area
```
git reset HEAD .
```

// add the staging area. (all files)
```
git add .
```

// make a commit.  (that is, a "save point") – creates a commit id (SHA – secure hash algorithm)
```
git commit -m "added welcome message"
```

// look at the history of changes. (long identified used to reference a commit)
```
git log
git log --oneline
```

// **TODO** – make another commit

## Why have a staging area?

1) Level of safety to make conscious decision of what you will commit
2) If you have 10 changed files, you could decide to only STAGE one or two files and commit them.
    a. This gives you the ability to choose a subset of files (like for one feature)

## Why have commits?

1) A commit should be one (or more files) related to a specific change in the code. For example:
    a. Added a new class
    b. Added a new method (feature)
    c. Fixed a bug
    d. Etc.
2) A commit should be well defined (not tons of code changes) – keep is focused! So the commit message clearly describes the change.
3) This will make the git log command very helpful – to review the history of your project!

## Rewind changes

Git gives you the ability to "back in time" and restore your project to its state at a point in time in the past! And then you could move back up to the "latest-greatest" code.

// go back to a specific commit. (reference the short version of unique id)
```
git checkout ######
```

// go back to the top, latest-greatest. (HEAD)
```
git checkout master
```

## Git has much more!

I did not cover several items – watch the videos!

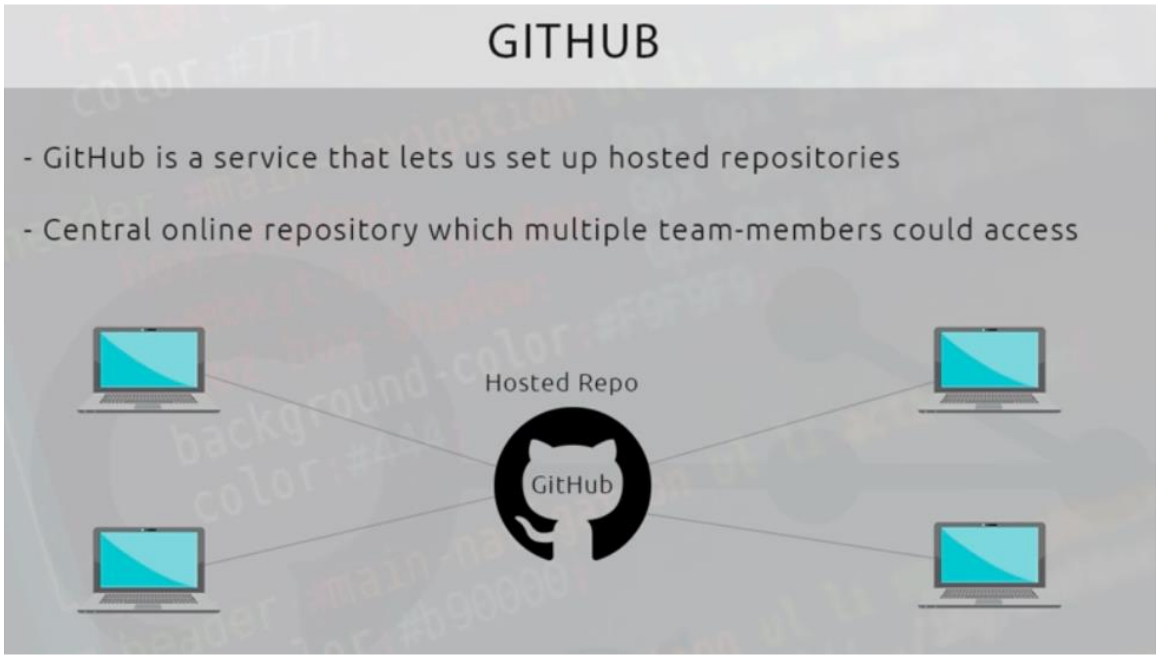
Git and GitHub tutorial (YouTube), 12 videos (5-10 min each): https://www.youtube.com/watch?v=3RjQznt-8kE&index=1&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR

I also recommend Ry's Git Tutorial (eBook): https://www.amazon.com/Rys-Git-Tutorial-Ryan-Hodson-ebook/dp/B00QFIA5OC

Many, many other videos and tutorials available!!

## Introduction to GitHub

1) Online service to host project
2) Share code with other developers
3) Developers download project and work on them
4) Upload your edit and merge them into main codebase



## GITHUB

- GitHub is a service that lets us set up hosted repositories
- Central online repository which multiple team-members could access

The “remote repository” is hosted by GitHub.

The four individuals can “clone” the repository (all folders, all files) – copying the repository to a local folder.

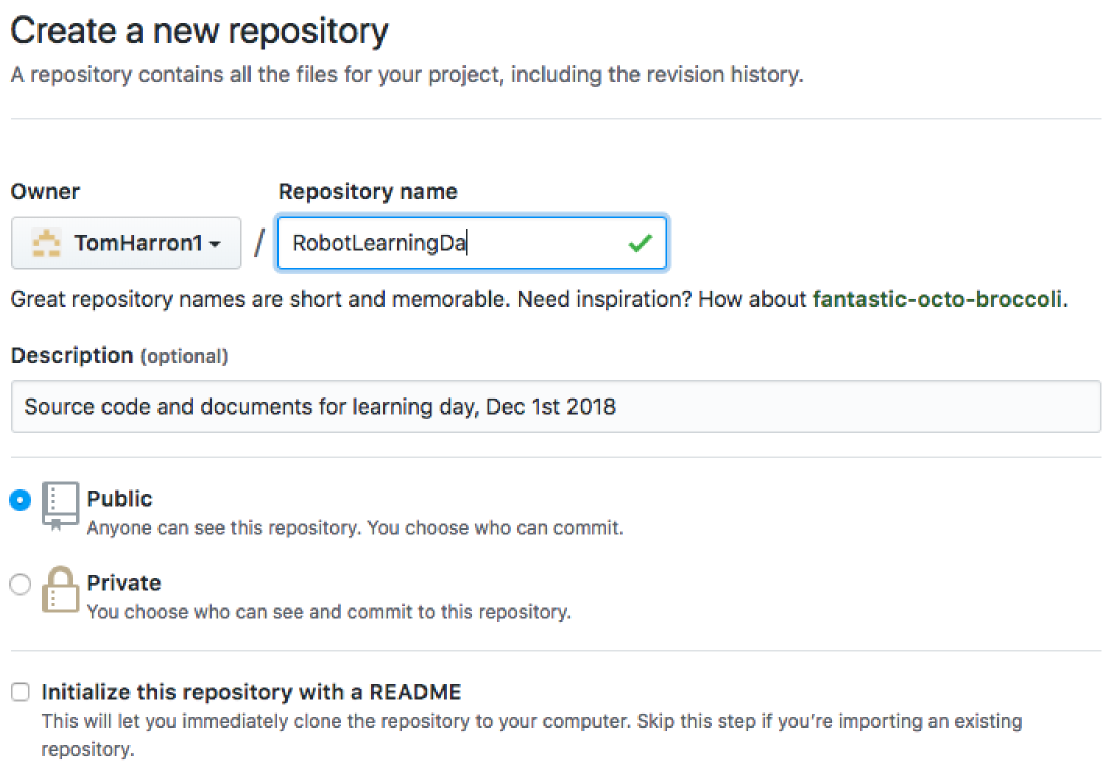
Any individual can then make code changes (in their local repository) and upload them to the central, or hosted repository.

## Create a GitHub account

Go to https://github.com/
Enter Username, email and Password.   That’s it!

**Note:** Free accounts are PUBLIC – anyone could view/copy your repository. You need to pay a small fee ($7/month) to make them private.

Robotic’s teams get a private organization account.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: TomHarron1 / Repository name: RobotLearningDa

Great repository names are short and memorable. Need inspiration? How about fantastic-octo-broccoli.

Description (optional): Source code and documents for learning day, Dec 1st 2018

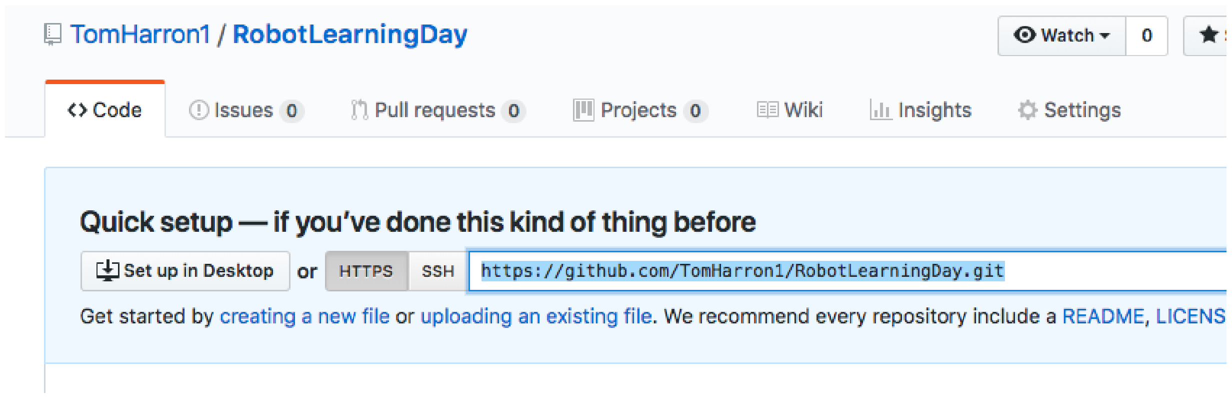
[✓ Public] Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

The created (remote repository is accessible via the generated URL:

https://github.com/TomHarron1/RobotLearningDay.git



TomHarron1 / RobotLearningDay

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/TomHarron1/RobotLearningDay.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENS

## Push our code to the remote repository

```
git push https://github.com/TomHarron1/RobotLearningDay.git master
```

GitHub repository displays folders, files and commit history!

## Make a local change and push to the remote repository

// **TODO** – change a file, see that it is modified

```
git status
```

// add the staging area

```
git add GoodRobot.java
```

// make a commit
```
git commit -m "welcome to GitHub message"
```

// push our changes to remote repository
```
git push https://github.com/TomHarron1/RobotLearningDay.git master
```

## Clone the repository

// change directory to a new folder on your machine
```
cd /RobotCopy
```

// clone the entire repository to your folder
```
git clone https://github.com/TomHarron1/RobotLearningDay.git
```

You can now:

- make changes to the files on your local repository. (modify, git add to stage, commit)
- push the changes (i.e. the commit) to the remote repository. (git push command)

Another cool things (after you install git):

- you can clone any of the 1000's of project in github!