# Git and Github

## Introduce the problem to be solved

## What is GIT?

## Install GIT

## Initialize GIT

## What is a Repository?

## Using GIT

## What is a commit?

## Modified, Staged and Committed files

## Introduce the problem to be solved

1. Developers have programming projects in a directory structure:

**/RobotErnie**

/src

file1.java file2.java

/bin

1. 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

## 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: <https://www.youtube.com/watch?v=3RjQznt-8kE&index=1&list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR>

## Initalize Git

git –version

// set user.name and user.email

git config –global user.name tomharron

git config –global user.email [tom.harron@school.com](mailto: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.



## 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

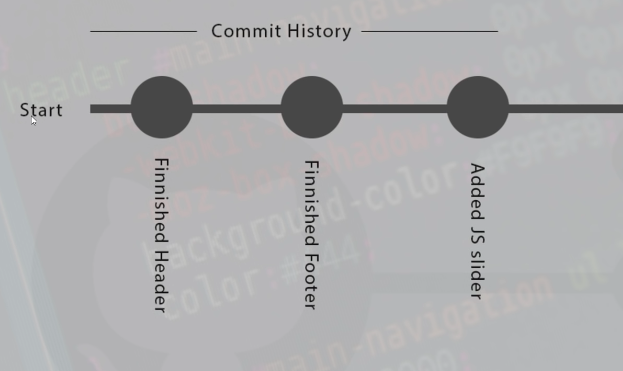
git add GoodRobot.java

// 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.

## 

## 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”)

git commit -m “added welcome message”

// look at the history of changes

git log

git log --oneline