THE KNOWLEDGE HOUSE

MODULE 3

GIT FUNDAMENTALS

OBJECTIVE

To read and understand the basics of Git Fundamentals.

EXERCISE:

- We are going to make two files 1 is a README File which will add text and titles to our GitHub repo.
- We will create a module2.txt file in the Module 2 folder in it you will write in your own words the answer to the following 3 questions:
    a. What is Git and how is it different than GitHub.
    b. Why do we use the terminal ?
    c. Explain 3 benefits version control in your own words.
- Create a new branch Module2, check into it and push the new files.
- Merge Module2 into Master.

MODULE 3
GIT FUNDAMENTALS

WHAT IS GIT?

Git is the most commonly used version control system for developers period, and GitHub, a public place to host Git Repos is basically the biggest single online asset a budding developer needs to get noticed by employers.

WHAT IS A VERSION CONTROL SYSTEM?

A version control system is basically a tool for managing changes to your codebase (the folders and files your code is written in) in a way that allows for changes to your code to be tracked over time and makes code being worked on by more than one person available in a central place.
Basically it's google docs for code: it allows multiple people to work on the same files, and for changes to be tracked, enabling developers to compare each other's changes and roll projects back to a previous version if the current version is broken.

Git also uses a branching system. Branches are basically ways to maintain the codebase in different states simultaneously.

So let's say we have a codebase for our app creatively named APP. Usually, a project would have at least 3 working branches.

The default branch is called MASTER and is the first branch created when a new git repo is created. Usually Master is where your finished and tested changes would go, be deployed to the web and ultimately your users.

Next, we have staging, staging is where code changes that on their own have been tested are integrated and tested in a full system.

Lastly, we have feature branches, where new features in development are saved and tested before being ready to integrate into the larger project.

MODULE 3
GIT FUNDAMENTALS

WHAT IS A VERSION CONTROL SYSTEM? CONT.

Example:

A team on an e-commerce site is looking to build out a new shopping cart experience for customers.

The team developing this new feature will most likely be working off a Git branch called **cart** here is where the team will build and test code running only on their individual computers until the cart is ready to be tested on a live staging site. This transfer would happen by **MERGING** branches basically combining the cart branch into the staging branch, with the team working through any conflicts that arise.

Once the new cart system is working on the developer's local machines (the ones they physically have in front of them), the merge would allow them to take this new feature and test if it works on the live site, however often when pushing from a local machine to one hosted on the web things break.

This is why we use a staging branch which is connected to a staging server. A staging server is a web host that puts up a private (sometimes password protected and sometimes not) clone of the live site, which while on the web is really only accessed internally. Here we can test new things in a live environment before putting them out for the world to see.

Once things have been tested and are shown to work on staging we would wait till the app has some downtime (say 4:30 am on a Sunday) and merge our staging and master branches updating master and pushing the changes to the actual live site.

This explanation shows the why, but let's talk a bit about the how.

FIRST THINGS FIRST, A NOTE

GIT is not GitHub

MODULE 3
GIT FUNDAMENTALS

WHAT IS A VERSION CONTROL SYSTEM? CONT.

This is a rookie mistake but please note these are not the same thing. Git is the actual version control tool to create repos or repositories which are basically just folders that use git to track changes to the files and folders within.

If we were to activate git in your TKH_Modules directory for example git would track any changes made to any files in that folder or the folders contained in that folder.

GitHub is just a platform used to save and store your code, and track its versions in a more visual way. Also, Github is used to collaborate with others because if everyone is working on a codebase a way to host that codebase in a place that is accessible by all team members is important. **Think working on a google doc, vs working on a word doc.**

Basically, changes made on any team member's computer can be pushed up to a central location hosted on Github.

It is important to note that Github is not the only place to host Git Repos, alternatives like BitBucket are also viable, however, Github is the most popular Git Hosting/Sharing Platform.

INSTALLING GIT

Installing on MacOS: Just type git into the terminal, if you have it it will give print out all the git command options, if not it will just prompt you to download.

MODULE 3
GIT FUNDAMENTALS

INSTALLING GIT CONT.

Installing on Windows:

Just go to **https://git-scm.com/download/win** and the download will start automatically.

Ex 3 Download Git if you do not have it.

Ex 4 Signing up for GitHub: **https://github.com**

Now here are some of the basic commands for initializing a git repo (taking a folder and allowing git to track it:

```
$git init /directory_name
#for specific directory
$git init
#initializes hit in your current working directory
```

MODULE 3
GIT FUNDAMENTALS

INSTALLING GIT CONT.

Now there are three ways to connect a repo to GitHub for online tracking:

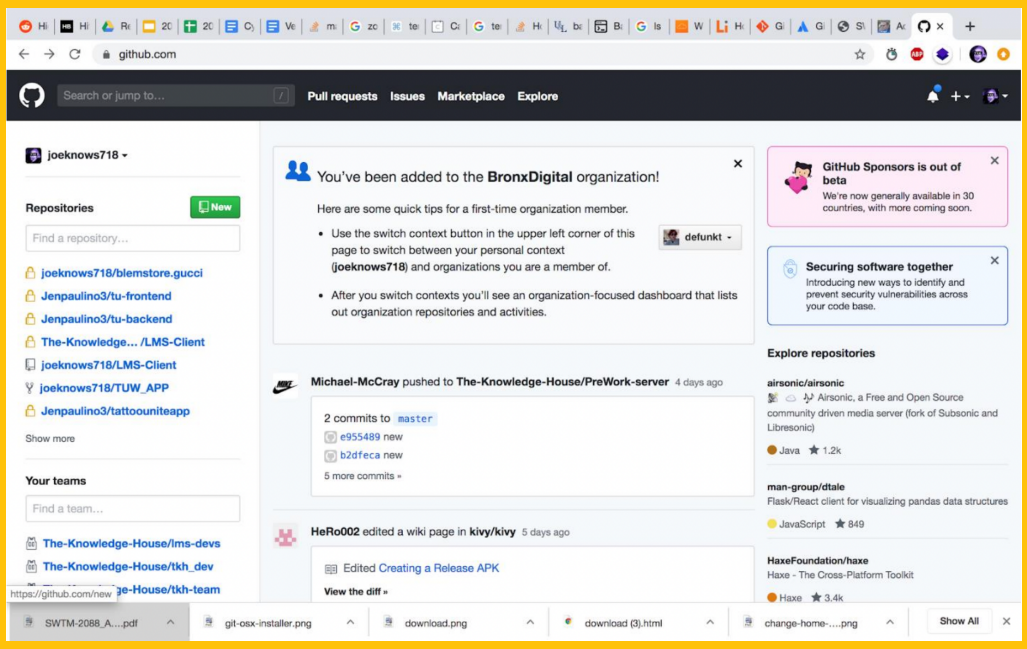
- You create an empty repo on GitHub, and then connect it to an existing git repo on your computer.
- You create an empty repo on GitHub, and clone that repo into your computer

You create an empty repo on github, and then connect it to an existing git repo on your computer:

First step is you go into your terminal and navigate into the home directory you want to initialize your hit repo in:

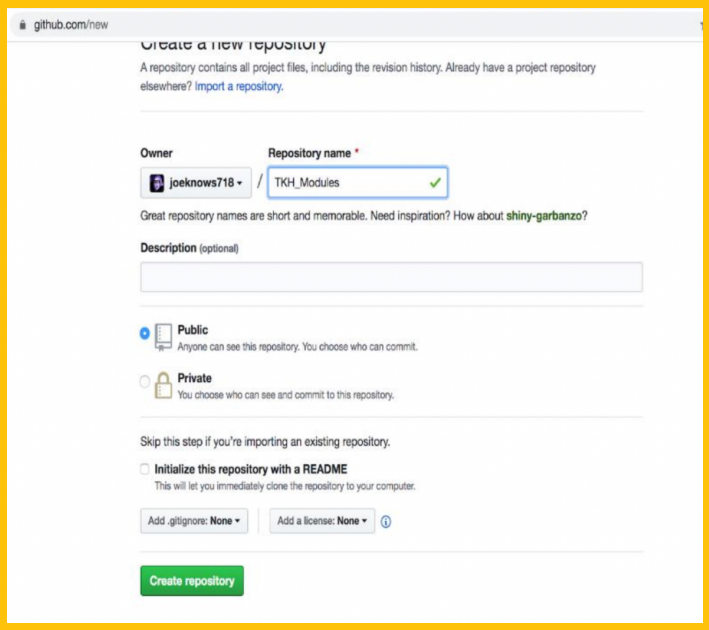
```
$ git init
```

Next you need to create a repo in Github:

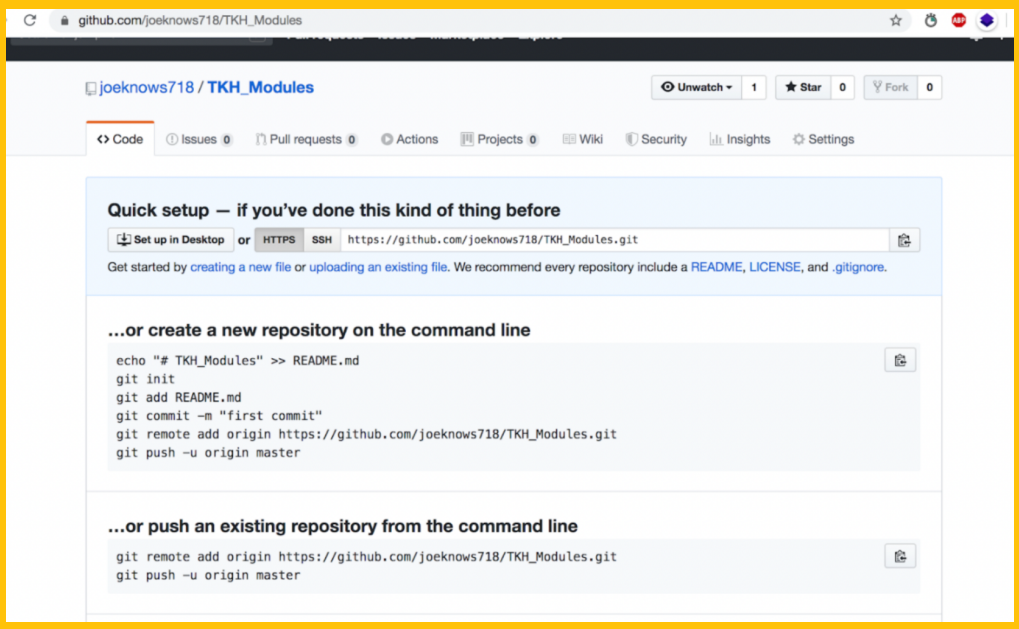


MODULE 3
GIT FUNDAMENTALS

INSTALLING GIT CONT.



Once it is created it will give you the remote location link for your repo



MODULE 3
GIT FUNDAMENTALS

INSTALLING GIT CONT.

Now we have already created our file structure so we want to connect our TKH_Modules folder to our new TKH_Modules Repo:

```
$cd /THK_Modules
$git init
#we initialize our git repo
$git add .
#the add command takes everything we have in our directory and adds them to be #tracked by git
$git commit -m "first commit"
#git only pushes changes as 'commits' along with a message describing what the #changes are. You can not push to git without a message
$git remote add origin <YOUR GITHUB REPO URL>
#this command connects your local git repo to the empty github repo you created earlier
$git remote -v
#this commands will print out the remote connection for your repo, if the #previous step worked the address of your github repo should show up as origin #master
$ git push origin master
#pushes all files and folders to github
```

Now we have successfully pushed to our repo. Congrats your using Git!

GIT BRANCHING

As mentioned earlier, not only do we save things to our git repo's but we also create branches because nothing should ever be sent to the main branch unless it always works. What a branch does is essentially create two separate repos for the same code base. Think of it as having two save games, one save where you test new stuff, and one that's your official profile.
In our module submissions, we want a different branch for each homework submission so we can track your submissions from branch to branch. We also want your Master branch to be updated.

Your repo should have the following branches at the end:

- Master
- Module 1: Empty set of folders with the txt responses from module 1 in the module 1 folder
- Module 2: Folder structure with your module two submission
- Module 3: Folder structure with your module three submission
- So on to module 7

MODULE 3
GIT FUNDAMENTALS

GIT BRANCHING CONT.

Your goal will be at the end of each module to create, checkout, and submit your work on a specific branch and merge that branch back into master, updating your master.

First, let's create a new branch for module 1 submissions, push to that branch and then merge that branch into master.

```
$git checkout -b Module1
#this creates and checks us into a new branch called Module 1, we now have two branches
# Master and Module1, and we are currently checked into Module1 meaning changes we make #will be saved in module 1 unless explicitly told not to be.
$git branch
#this will list out all our active branches
$git add .
$git commit -m "pushing to module 1 branch”
$git push origin Module1
#pushes your files in their current state to the Module 1 Branch
```

Now we want to merge Module1 into our Master branch

```
$git checkout Master
#first we checkout our Master branch so that we move from working in our Module1 branch back into master
$git merge Module1
#because we checkout Master, this command takes Module1 and merges it into the branch we are currently checked into, in this case master.
```

The last thing we should touch on for now is README files, these are basically files located in the home directory of a repo (the one you used init in) that allows Github to show some title and description info for your project. Below are some GitHub README best practices and how to’s:
https://guides.github.com/features/wikis/

Now that we have a strong idea about git fundamentals here is a resource for more git commands:
https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet

Video Tutorials

- **https://youtu.be/USjZcfj8yxE**
- **https://youtu.be/8JJ101D3knE**
- **https://youtu.be/RGOj5yH7evk**