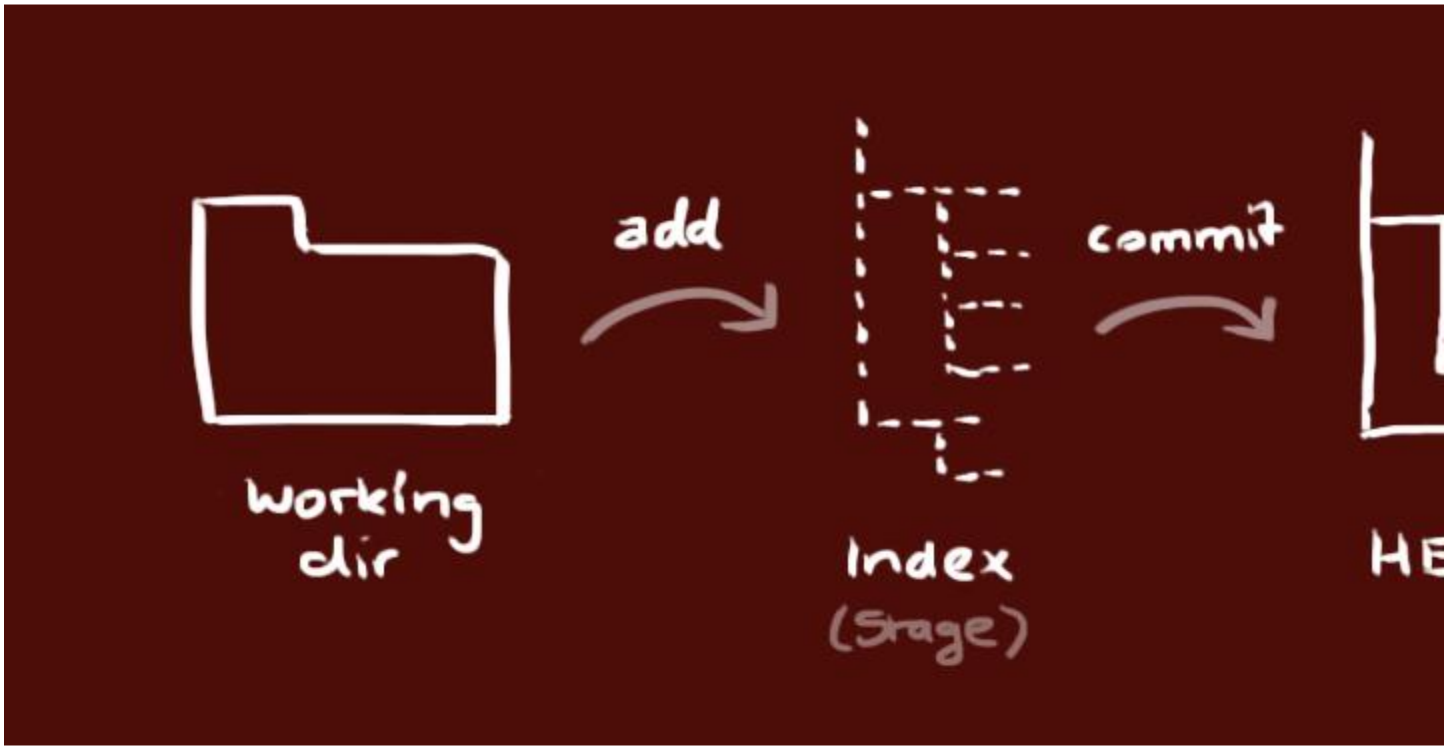
Learning Git

- git init   - initiate git repository

- git status -view the status of files in the repository (tracked/untracked)'

- git add <file> - Now file will be ready to commit. This is called STAGING. Because when we modify file and use 'git status' command we see file as both tracked and untracked.



https://www.quora.com/What-is-the-meaning-of-commit-and-stage-in-git

- `git config --global user.email "dupandit@ncsu.edu"`
  git config --global user.name "Darshit" --to tell git who you are

- git commit - once the file is staged. We are ready to commit that is adding the file to the repository.
pres I and type "Initial commit". Hit esc key to exit insert mode and Then type :wq which will rewrite the current file and exit

- git commit -m <text that is changed> -------- this will commit the updated file in the command line only

- git log ----------------- presents author of commit and the date. Useful while working in a collaboration

- Suppose we modify the original file and add a new file <file2>. Then do git status we will see to unstaged files in it. Now instead of staging one by one we can use this command.
  git add . ---- will stage all the files'
  git add *.html ---------- will stage all html type of files

- There are cases when we dont need to stage all type of files (say .log) files. For that use :
creat .gitignore file by using :
 touch .gitignore --- there add text *.log

- git commit -m 'new updates' -------- will stage new files and also this .gitignore file
- git commit -a -m "Add existing file"
- git commit -m "Adding a new file"

BRANCHING AND MERGING :

- git branch <name> ----- it will create a branch for you where you want to type code

- git checkout <name> --- it will switch you to <name> branch
create new index file and if you change the existing file and do git status then you will see files to ccommit as expected, but

- git checkout master ----- master is always the name of the main branch
When you check the file location in this main branch 'master', new created file and updates will not be there. If you do git status, nothing is there to commit.

So there requires merging.

- git merge <name> --- name of your branch. It will merge to the master and you will see updates to be commited and

READ ABOUT MERGECONFLICT --------pending

###################################
STASH

lets say I am in MyBranch, I created a new file. I didn't commit but I checked out to master. We can see that the new file is also visible in Master which is not desirable. For that we need *STASH*.

So when you create a file and do 'git add .' 'git status', you see files to be commited.
Now when you do
-'git stash'.
Stash does is take dirty state of branch (including tracked modifications and staged changes) and saves it on a stack of unfinished changes that can be reapplied at any time.

To resume working, use

- git stash apply -----

###########################
REMOTE REPOSITORIES

Till now everything is happening locally. You may want to work with remote repositories like "Github".

- git remote ---- list of remote repository.

-git clone "link of remote repo" ---- put the whole repository data and state of stages and all in our local repository

- cd Learning-Git
- git remote -------------------this will show origin. Means it is a repository.

- git fetch origin ------ fetch all the new updates from this repository since last fetched or cloned. But it will not merge into your work. We need to do that manually.

- git pull origin -----------fetch +merge

Now say you made changes in that fetched files from new repository or created a new file in the new repository.
Then commit you work, shortcut command below which includes both staging and commiting.
- git commit -a -m 'new updates' or 'Add existing files'

- git push origin master -----------push updates from our repo to the remote repo that is origin

- git remote add MyRepo http://github.com/ ----- to add new local repository

- git fetch MyRepo

Summary :
A repository is simply a place where the history of your work is stored. It often lives in a `.git`subdirectory of your working copy - a copy of the most recent state of the files you're working on.
To fork a project (take the source from someone's repository at certain point in time, and apply your own diverging changes to it), you would clone the remote repository to create a copy of it, then do your own work in your local repository and commit changes.

Within a repository you have branches, which are effectively forks within your own repository. Your branches will have an ancestor commit in your repository, and will diverge from that commit with your changes. You can later merge your branch changes. Branches let you work on multiple disparate features at once.

You can also track individual branches in remote repositories. This allows you to pull in changes from another individual's branches and to merge them into a branch of your own. This may be useful if you and a friend are working on a new feature together.