Git and GitHub

Initializing an empty git repository

git init

.git folder will be created

A .git folder is required to log every commit history and every other information required for your remote repository, version control, commits etc.

# To see what you have changed

git status

The git status command is used to display the state of the repository and staging area. It allows us to see the tracked, untracked files and changes.

## How to add the changes in git staging area

git add .

if you want to add all the files then use . and if you want to add particular file for ex-index.html then use

git add index.html

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.

## Commiting the changes

git commit –m “the message you want to give”

## Removing changes from stage

git restore --staged .

if you want to remove particular file for ex-index.html from stage then use

git restore –staged index.html

## Viewing the overall history of the project

git log

## Removing a commit from the history of the project

git reset commit\_id

## Stashing changes

git stash

git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.

if you want to commit the changes but want to exclude particular file and want save in the local so that you can commit that in next commit

to status particular file for ex- welcome.txt

git stash welcome.txt

Popping Stash

git stash pop

### Now the previously stashed files will be ready to commit

## Clearing Stash

git stash clear

this will clear the stashed files , now you can’t get these files at the later time

## Connecting remote repository to local repository

git remote add origin ‘repository link’

## How to push the changes for particular branch

if you want to push the changes to master branch then

git push origin master

## Git branches

why we use branches

branches are a part of your everyday development process. Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes

for example we are pushing the angular and java code so for both the things we can create separate branch so work will be organised , angular changes will go to angular branch and java changes will go to java branch

How to create the branch

git branch ‘branch\_name’

let’s say you want to create feature branch then

git branch feature

how to switch to particular branch

git checkout feature

now head will point to feature branch, the head is a pointer that says all the new commits that you will made will be added on the head currently head is on the feature branch so all the changes will be go to feature branch

Never push the changes directly to main branch because this is the branch from where actually the code is actually being used by the people

How to see the currently selected branch

git branch