# Git and Github Tutorial

**git clone <link> :** to clone a folder from github (remote) to local (on laptop).
Note : Do not add the angular brackets while adding the link

eg. git clone https://github.com/shreyamali01/DSA-practice.git

Basic terminal command - cd

Auto complete in terminal is done by 'Tab' key

ls - to list all the files in that file

'ls -a' : to list out hidden files, it will list out all files
''

cd .. : to exit the folder using command line

**git status :** tells us about the status of our git folder. Whenever, there are any changes on the vs code file which are not updated on the github folder, we get a prompt that our file is modified and the code is not changed on github

Whenever any change is made in the code, the change on the github folder is made in two stages, add and commit.

There are four types of status that we can observe on using the **git status** command :

1. untracked : new files that git has not tracked yet
2. modified : changed
3. staged : file is ready to be committed
4. unmodified : unchanged

**git add <file name> :** adds new or modified files in the git staged area

**git add . :** to add all the files on github

**git commit -m "some message" :** record of the change made

**git push origin main :** to upload local remote repository to remote repository

What to do if the new folder is created on the local machine itself?

init - used to create a new git repository

**git init**
**mkdir** : to make new directory using command line

**git remote add origin <link> :** adding a new remote repository

**git remote -v :** to verify remote repository

**git branch :** to check branch

**git branch -M <new name> :** to rename the branch name

**git push -u origin main :** This option sets the upstream branch for the current branch. This means that future git push or git pull commands will automatically push to or pull from this branch without needing to specify it again. It's a way of telling Git to remember where to push/pull changes by default.

### GIT BRANCH COMMANDS

**git branch :** to check the branch we are in

**git branch -M <name> :** to rename the new branch

**git checkout -b <new branch name> :** to create a new branch

**git checkout <branch name where you want to go> :** to navigate in branches

**git branch -d <name> :** deleting the branch

## MERGING CODES

There are two ways of merging the code -

*Way 1*

**git diff <branch name> :** to check the differences between both branches

**git merge <branch name> :** to merge 2 branches

*Way 2*

Creating a pull request : Requesting them to merge your branch in the main code

**git pull origin main :** used to fetch and download content from a remote repository to local repository to update the local repository to match the content.

## RESOLVING MERGE CONFLICTS

An event that takes place when git is unable to automatically resolve the differences in the code between two commits

## UNDOING CHANGES

Case 1 : staged changes

**git reset <filename>**
**git reset**

Case 2 : for committed changes (one commit)

**git reset HEAD~1** : only the recent most / head commit

Case 3 : for committed changes (many commits)

**git log :** to check the git commit

**git reset <commit hash>**
**git reset --hard <commit hash> :** to reverse the commits on VScode too.

## FORKING A REPOSITORY

Creating a rough copy of the repository, which also has the other repository's code but it is our own rough copy.