Github is a website where you store your code in the cloud. And it is also called as **Version Control Sytem.**

**Repository** is a directory where you will have different directories and files.

**Pull Request:-** we will fork(copy) some others github code into our private repo and we will make any

Changes in the code and push to our private repo and send pull request to the members where we copied the code from. If they feel like the code what we changed then they will accept the pull request.

**Issues:-**  We can write any issues what we have with the code and the owners of the repository will check and try to fix that issue.

**Wiki:-** It means we can write documentation about the project.

**Pulse:-** It is like a dashboard of your github account.

**Graph:-** To see contributors making how many pull request or issues, etc... In a graphical way.

Commands:-

**git clone** address of the repo :- To copy that repo in our local system.

**git status:-** To see if we have any changes we need to commit to Github repo.

**Staging Area:-** it is the place where you will add your changed code first and then we will commit local version control.

**git add [filename]:-** To add the file into staging area

**git commit –m ‘message’:-** git will commit all the files in staging area to local version control database.

**git log:-** To see all commit history.

**git push:-** git will push all the committed data from local version control system to remote server.

**git difftool HEAD:-** git will show you the difference b/w old code and modified code.(\*you have to see after adding the new code into staging area)

**Undo Uncommitted Changes:-**

**git checkout -- filename:-** To undo the modified changes in the file before adding into staging area.

**git checkout -- . :-** To undo the modified changes in all files.

**Undo Committed Changes:-**

**git revert (committed log id):-** It will revert the committed change and commit implicitly.

**git revert -n (committed log id) :-**It will revert the commit change but will not commit that changed one. We have to commit explicitly.

**git reset --hard (committed log id):-** Git will delete all the committed code after the mentioned log id.

**Creating, Merging and Deleting a Branch:-**

**git branch:-** It will show the branches.

**git branch branch\_name:- I**t will create branch name.

**git checkout branch\_name:-** git will switch to the given branch name.

\*We have to be in the branch where we want to copy the modified branch and type

**git merge branch\_name:-** it will merge the given branch name modified files into the branch you were in.

**git checkout –b branch\_name:-** To create a branch and checkout to that branch.

**git branch –d branch\_name:-** To delete the branch

**HEAD:** HEAD is a reference to the most recent commit to the present branch.

**git show HEAD** or **git show recent\_commit\_id:- Both** will get the recent commit id details.

**git difftool HEAD~3 HEAD~1:-** To see the difference between two commits data.we have to put older commit first and newest commit next. **HEAD~3** means the present minus 2 commits data and **HEAD~1** means present commit.

**.gitignore file:-** if we want to ignore files to commit, you can add those file names in .gitignore file and commit. So the file names specified in .gitignore file will be ignored.