**GIT**

Git is an open source, distributed version control system.

* Git is content tracker, it stores all our code changes.
* We can able to see

. Which changes we made

.Who made the changes

.When we made the changes

Repositories:-

**Local repository:-** is on our own local machine so you have direct access to it.

Working Area: are all active changes. Git doesn’t know what to do with them yet, it just know that these files contains some updates.

Staging Area: contains new changes that will soon be committed

Committed files: all commits

**Remote repository:-** usually a centralized server and is entirely optional.

**Git help <command>** 🡪 to view additional help on each command [for this we must install man pages]

**Sudo apt-get install git-man**

Git init – to initialize an empty git repository after this .git folder will be created

Master branch –default branch

Storing changes in a local git repository is as called **committing**

* A commit record the changes of the repository compared to its previous state

**Git restore <file>**  (unknowingly if we update file) if we updated the file after commit; then we do not want that changes to be updated at that case we will use restore command

Git restore –staged <file> (to un stage a file)

**Git add .** – to stage all the files in working area

**Git rm <option> <file>** 🡪 to remove a file from the staging area

--cached –will retain the file in our directory

-f –will delete the file permanently.

To make a file untracked if It is already tracked

\*if I want to ignore a file from add/commit commands then we have to use .gitignore

**Echo “notes.txt” >> .gitignore**

**git status** 🡪 to see files under working area

**git log** 🡪 to see the information that you need to know about all the commits

**git log –oneline** 🡪 to show in one line

**git log –name-only** 🡪 to see commit history with file names

**git log –decorate** 🡪 to see who made the changes

branch – pointers to certain commit

**git checkout –b** <branch name> 🡪 to create new branch and switch to that branch

**git checkout** <branch name> 🡪 switch branch

**git branch** <branch name> 🡪 creates a new branch

**git branch** 🡪 to see all the branches

**git fetch origin master** 🡪 to update the origin master branch in our local repo

retrieving changes of remote repo to local repo.

**git merge origin/master** 🡪 merging changes of one branch to another

git pull origin master = git fetch + git merge

git fork <url> 🡪 if we are not a part of the project but we are interested to contribute to the project at that case we can able to copy the project using fork command

to add permissions for the collaborator settings --- > collaborators

**git rebase master** 🡪 getting changes from branch to another branch, we are modifying git history(hash)

**git rebase –i HEAD~4** 🡪 for meging all the commit messages to single commit [pick, squash]

**git cherry-pick <commit-hash>** 🡪 used to pick single commit from one branch to another

revert command is useful when you want to undo the changes that you made but keep those changes into your git history. It will create new commit to keep change.

**Git reset –soft** 🡪 commit changes will be saves [unstage]

**Git reset –hard** 🡪 delete permanently

**Git stash** 🡪 quick way to set aside your current changes and go back to clean working directory

**Git stash pop** 🡪 get back changes

**Git stash list** 🡪 to see stash files

**Git stash show stash{0}** 🡪 to see particular stash file

**Git stash pop stash {1}** 🡪 to get back specific one.

Git reflog 🡪 command shows us all the actions that have taken on our repo. This include merge, revert, rebase, reset etc.

Git log only shows commit history, but git reflog shows the actions that are performed.

Git pull –rebase –allow-unrelated-histories 🡪 to allowing unrelated while pushing