GIT

It is a version control system/ tool/ software.

- ❖ Version control system- tools to track changes in code.

GIT is used for 2 works:

- Track the history
- Collaborate

Github: It is a website that allows developers to store and manage their code using GIT.

There is a file in Github repository named README.md which stores related information (md-markdown).

In the README.md there is initial commit written which indicates changes done for the first time in the repository.

We can also use basic HTML in README.md

COMMANDS

git --version: to show the version

ls: to list the files

clear: to clear the window

git config --global user.name “My Name”: to set the username

git config --global user.email “xyz@gmail.com”: to set email

git config --list: gives the list of commands used in git

- ❖ credentials.helper: stores all the credentials

~: root directory/ main folder

clone: to clone a repository on our local machine

- ❖ Syntax: git clone <-some link->

cd: to change directory

- ❖ Syntax: cd <-directory name->

status: displays the status of the code

- ❖ Syntax: git status

- These are of 4 types:
    - Untracked: new files that git doesn't yet track (added and not committed)
    - Modified: changed
    - Staged: file is ready to be committed
    - Unmodified: unchanged

add: adds new or changed files in working directories to the git staging area.

- Syntax: git add <-file name-> (to add 1 file)
- git add . (to add more than 1 file)

commit: it is the record of change

- Syntax: git commit -m "some message"

push: upload local repo content to remote repo

- Syntax: git push origin main {main: branch}

init: used to create a new git repo.

- Syntax: git init (to make a git repo)
  git remote add origin <-link-> (to set the origin)
  git remote -v (to verify remote)
  git branch (to check branch)
  git branch -M <-new branch name-> (to rename branch)
  git push -u origin main (-u: if we want to work on the project for some long time then if we put this then we only have to write git push)

WORKFLOW (Local git)

Github repo -> Clone -> Changes -> Add -> Commit -> Push

GIT BRANCHES

If a product has different features and different users work on it then separate features are formed.

Why?

As no developer likes to wait for a feature which he wants so the users make their copies.

git checkout <-branch name->: to navigate to other branches

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

git branch -d <-branch name->: to delete branch

git diff <-branch name->: to compare commits, branches, files etc

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

OR

Create a PR (Pull Request)

PR: It let's tell others about changes you've pushed to a branch in a repository on Github (If in 1 project various developers want to merge with main)

git pull origin main: used to fetch and download content from a remote repo and immediately update the local repo to match that content.

MERGE CONFLICTS

When there are changes in the files of the different branches on the same spot then it arises as it is not able to resolve differences.

UNDO CHANGES

Case 1: staged changes (added but not committed)

- Syntax: git reset <-file name-> (for 1 file)
  git reset (for multiple files)

Case 2: committed changes (for 1 commit)

- Syntax: git reset HEAD~1

Case 3: committed changes (for multiple commits)

- Syntax: git reset <-commit hash-> (changes will be remove from git)
  git reset --hard <-commit hash-> (changes will be removed from local)

git log: to check the history of commits (To quit: q)

FORK

- It is a new repo that shares same code and visibility settings with the other repo.
- It is a rough copy.

CHERRY PICKING

It is the process of picking a particular commit from one branch to another branch.