Git and GitHub

Git

Version control system is a tools that helps to track changes in code.

Git is a version control system.

Git is mainly used for two reasons-

* Track the history of code.
* Collaborate with team or other person.

GitHub

Website that allows developers to store and manage their code using Git. We store code or anything in GitHub like a folder (Repository).

Setting Up Git

We can set our git in various platform, such as-

* Visual Studio Code
* Git Bash (Windows)
* Terminal (Mac)

1. For verifying git is set in our computer or not, we write in terminal, git --version

Configuring Git

There is two types of configuration in git. One is global and another one is local. Global means when we make a git change on system, that change will be applied to a specific email ID.

Local means we will change different account within a repository or whith in a project.

* git config --global user.name “My Name”
* git config --global user.email [@email.com](https://@email.com)
* git config –list

Clone and Status

Clone – Cloning a repository on our local machine. Clone means create a duplicate something. There is two types of clone.

Remote- we create a copy of a file in our laptop or computer that is on GitHub.

Local- we want to make a copy of a file that have in our computer or laptop.

* git clone <-some link->

The another command is cd. Cd means change directory (folder). When we keep many folder in a folder and want to work on a specific file, then we use cd command.

* cd <-file\_name->
* cd .. ( use for work on main folder )
* mkdik (use for create a new folder )
* ls (use for show file )
* ls –a (use for show all file with hidden file also)

Status- display the state of the code

* git status

**Some short name when we work git->**

U – untracked -> New files that git doesn’t yet track

M – modified -> changed

A – add file -> file is ready to be committed

Add and Commit

Add- adds new or changed file in our working directory to the Git staging area.

* git add <-file\_name->
* git add . (used for add all file in git)

Commit- It is the record of changes

* git commit -m “some message”

Push Command

Push- upload local repo content to remote repo.

* git push origin main

1. Process of push file in GitHub from Git->

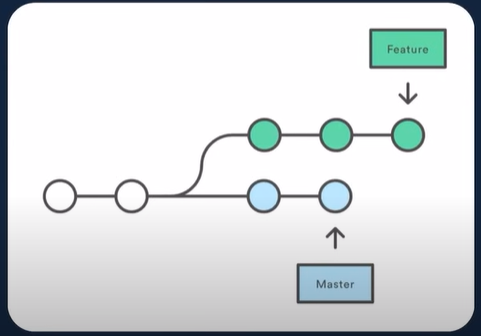
* git init
* git add . (add all file) / git add file.txt (add only one file that i want) / git add \*.txt (add all txt file)
* git commit -m “name of commit”
* git branch -M main (to rename branch)
* git branch (to check branch)
* git remote add origin <- link ->
* git remote –v (to verify remote)
* git push –u origin main

Workflow

To create a repo on GitHub --> clone --> changes --> add --> commit --> push.

Git Branches

When many people work on a repo then we create Branche.



When multiple developers work on a project, it is necessary to create branches. By creating branches, the frontend developer can work on the frontend while the backend developer can work on the backend. Later, the two branches can also be marged.

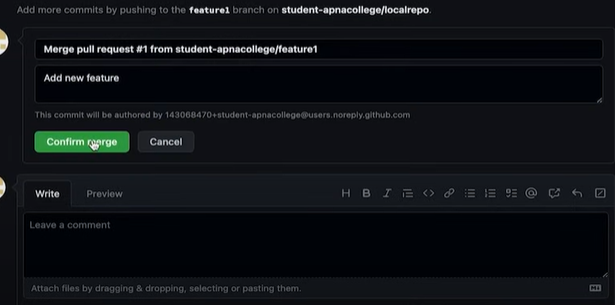
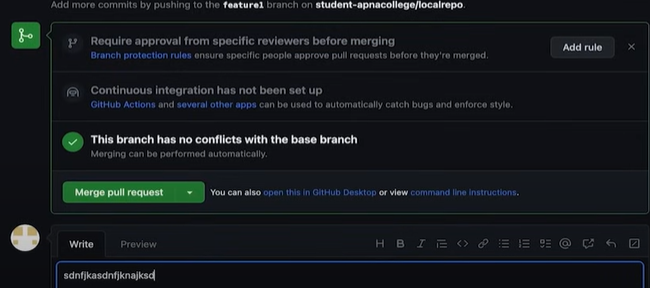
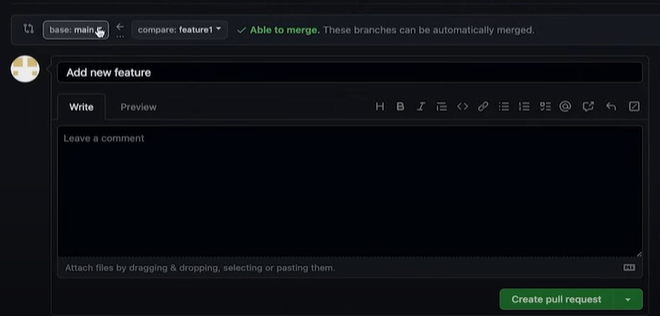
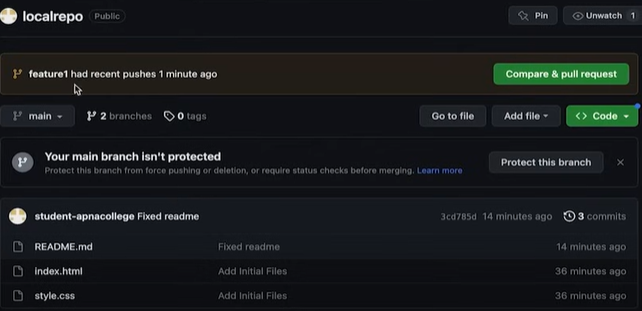
Branch Commands

* git branch (to check branch )
* git branch –M “name” (to rename branch)
* git checkout <- branch name -> (to navigate one branch to another branch )
* git checkout -b <- new branch name -> (to create new branch )
* git branch –d <- branch name -> (to delete branch ) we cannot delete the branch we are on.
* git diff <- branch name -> (to compare commits, branches, files & more)
* git merge <- branch name -> (to merge 2 branches)

**The another way to merge branches.**

Create a pull request.

Pull request -> It lets you tell others about changes you have pushed to a branch in a repository on GitHub. When we apply for pull request the senior developer check our changes in project if the changes is fine then the senior developer accept the pull and he/she merge the branches. The way of create pull request.



The changes we can see on our GitHub. If we want to see the changes in our local repo the we have to write some command.

* git pull origin main

Used to fetch and download content from remote repo and immediately update the local to match that content.

Resolving Merge Conflicts

When working on multiple branches and adding features or new things to the same line, and merging those branches results in merge conflicts. Since multiple features are added on the same line, Git gets confused about which features should be kept. That’s when we face conflicts. And we have to resolve this manually.

Undoing Changes

Undoing is when we remove things that we don’t need to add to our project, such as making changes or adding features. There are three types of changes:

**Type-1:**

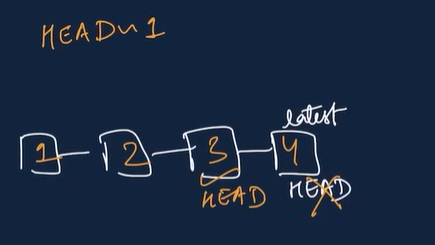
Staged Changes- ( which has been added but not committed. )

* git reset <- file name ->
* git reset

**Type-2:**

Committed changes- (for one commit)

* git reset HEAD~1

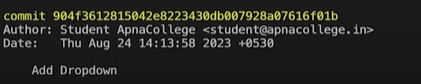


**Type-3:**

Committed Changes- (for many commit)

* git reset <-commit hash->
* git reset --hard <- commit hash ->

This is use for when we work on new stage and we want to back the previous stage without any modify or anything.



* git log (use for see all previous commit)

Fork

A fork is a new repository that shares code and visibility settings with the original “upstream” repository.

Fork is a rough copy.

**Thank You**