**Git and Github**

Git was created by **Linus Torvalds** in **2005** to develop Linux Kernel. It is also used as an important distributed version-control tool for **the DevOps**.

Version control system : tracks changes in code.

Git is popular, free,open source, fast and scalable

Features of git:

* To track the history of the project

Ex : while building a website, if we think to add extra features but that extra features have some issues, so we want to implement in the later version but the changes made in this version should be preserved and the new change which we left for next version should be build on top of the changes made in this version.

* To collobarate with other people so that it doesnt create any conflicts when two or more people work on the same module.

Github : website where developers store and manage the code.

We upload project in the form of folder. These folders are called as reposiotory.(repos)

Config

configures the user. The Git config command is the first and necessary command used on the Git command line. This command sets the author name and email address to be used with your commits. Git config is also used in other scenarios.

Get and set configuration variables that control all facets of how Git looks and operates.

Set the name:

$ git config --global user.name "User name"

Set the email:

$ git config --global user.email "himanshudubey481@gmail.com"

Set the default editor:

$ git config --global core.editor Vim

Check the setting:

$ git config -list

Clone

make a copy of a repository from an existing URL.

git clone <link>

Status

used to display the state of the working directory and the staging area.

git status

Change or modify file (untracked file) ----> add file ( staged file) ----> commitg file (unchanged)

Add ( stage)

To add new or changed files into working directory in the staging area

Git add <file name>

Git add . ( to add all the files from the folder

Commit

To record the change

Git commit –m ‘message’

Push

To add files from the local computer to the repository

Local->repo

Git push origin <branch name>

Init

To create new git repository

Git init

Git remote add origin <link>

Git remote –v ( to verify origin)

Git branch ( to check branch)

Git branch –m rename\_branch\_name ( to rename branch name)

Git push origin rename\_branch\_name

Branches

Git branch ( to check in what branch we are)

Git branch –m new\_name

Git checkout another\_branch ( to go from one branch to another)

Git checkout –b new\_branch ( to move into new branch)

Git branch –d branch\_name ( to delete branch)

Merge

1)Git diff branch\_name ( to compare commits,branches and everything that is changed in the branch)

2)Git merge main ( add the compared branch to main

Another method is to use pull request

Log

To check the commit history

Git log

By default, if no argument passed, Git log shows the most recent commits first. We can limit the number of log entries displayed by passing a number as an option, such as -3 to show only the last three entries.

Git log -3

.gitignore

Specify intentionally untracked files that Git should ignore.

Create .gitignore:

touch .gitignore

List the ignored files:

$ git ls-files -i --exclude-standard

Pull

Used to fetch and download content from the remote repository and immediately update the local repository to match the content with the remote repositpory

Git pull origin main

Stash

Allows you to change branch without committing the current branch

Switch branches without committing the current branch. Stash current work:

$ git stash

Saving stashes with a message:

$ git stash save ""

Check the stored stashes:

$ git stash list

Re-apply the changes that you just stashed:

$ git stash apply

Track the stashes and their changes:

$ git stash show

Re-apply the previous commits:

$ git stash pop

Delete a most recent stash from the queue:

$ git stash drop

Delete all the available stashes at once:

$ git stash clear

Stash work on a separate branch:

$ git stash branch

Cherry pic

Apply the changes introduced by some existing commit:

$ git cherry-pick

**Git rebase**  
Apply a sequence of commits from distinct branches into a final commit.  
$ git rebase

Git tag

It is like a snapshot of a specific point in the Git history. Tags are often used to mark specific releases of a software project. Unlike branches, tags are typically not moved once they are created.

NOTE:

git fetch is more conservative and only brings changes into your local repository, leaving your working directory and local branches unchanged.

git pull automatically merges the fetched changes into your current branch, potentially modifying your working directory.

In summary, if you want to see what changes are available on the remote but not yet in your local repository, use git fetch. If you want to both fetch and merge the changes into your current branch, use git pull.