Git:

An Unpleasant person in british english

Repository:

Storage Space where the project resides.
It can be in LocalFolder, GitHub, Online Host.
People working on the project are termed as contributors in github

Git Config:

–global : info about .gitconfig directory. (Mostly used)
–system: info about users on the system
–local : particular project repository info is stored

Commands:

clear : Used to clear the screen

cd **"Your Directory Name"** : Moves to the specified directory

cat **"Your File Name"**: Used to display the contents of the file

rm -rf **"Your Directory Name"** : Removes to the specified directory

git config - - global user.name **"Your Name"** : Used to set the username

git config - - global user.email **"Your Email ID"**: Used to set the user email

git config - - list : Used to see the configurations made

git help **"Your Command"** : Used to see the manual of the command

ls -al : Used to see the hidden files

git clone **<repo-url>** : Used to clone the repository

git status : Used to know the condition of the git

git status - - short / git status -s : Short description of git condition is displayed

- ?? - untracked
- A - staging Area
- M - modified Files

git add **"Your File Name"**: Used to update changes and move the file from working directory to staging area.

git add **.** : Used to update changes and move all the files from working directory to staging area.

git restore **"Your File Name"**: Used to discard changes and move the file from working directory to staging area.

git commit -m **"Your Message"**: Used to commit the changes and move the file from staging area to local Repository.

git commit -v : gives more details about git condition and opens notepad

git push origin master : Used to move the file from the local Repository to Github Repository.

git init : Used to initialize an existing folder to a git repository

git remote : Used to know the server name

git remote add origin **<repo-url>** : Used to setup a server with name origin

git push -u origin master : Used to push the folder into the github repository

notepad .gitignore : Used to create a gitignore file.

git diff : displays the content difference of the file in unmodified state and modified state.

git log : Used to know the previous commit history of the project
git log - - pretty=fuller: previous commit history is displayed with complete info.
git log - - oneline: the previous commit history is displayed in single lines.
git log - - oneline - -all: previous commit history of all branches is displayed.

git log - - oneline - -graph: previous commit history of all branches is graphically displayed.

git branch **“Your Branch Name”**: Used to create a new branch
git checkout **“Your Branch Name”**: Used to move into that branch

git checkout -b **“Your Branch Name”**: Used to create a new branch & move into that branch.

git merge **“Your Branch Name”**: Used to merge the branches. Branch name should be the ones different from the current branch

git branch -d **“Your Branch Name”**: Used to delete the branch

git branch -a: Used to see all the branches

git branch - - merge: Used to check which branches are merged

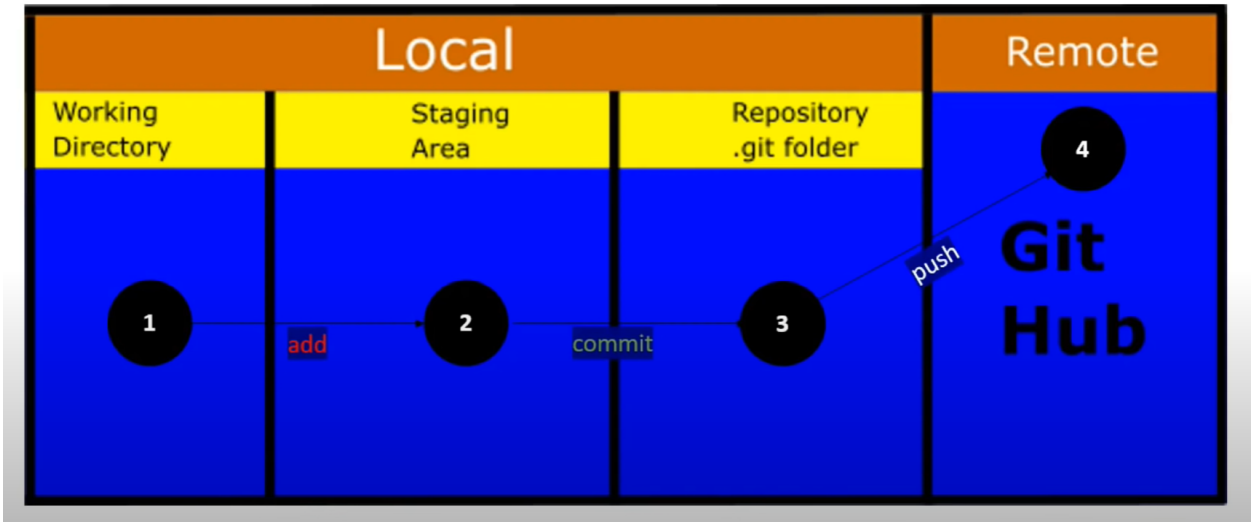
git branch - - no-merge: Used to check which branches are not merged

Git Workflow

Commit : Information and details about the change made in repository

Clone : Exact Copy

Tracking/UnTracking :
.git folder maintains a history of changes made/ being made on a repository on certain files known as Tracking files and on which it doesn't maintain are Untracking Files.



1 -> The place where code is written/changes are made to the code
2 -> PreProcessing
3 -> Local Repository
4 -> GitHub Repository

MASTER & ORIGIN

The default branch is the MASTER branch when we obtain code from a repository
The default server name is ORIGIN.

CLONE Existing Repo:

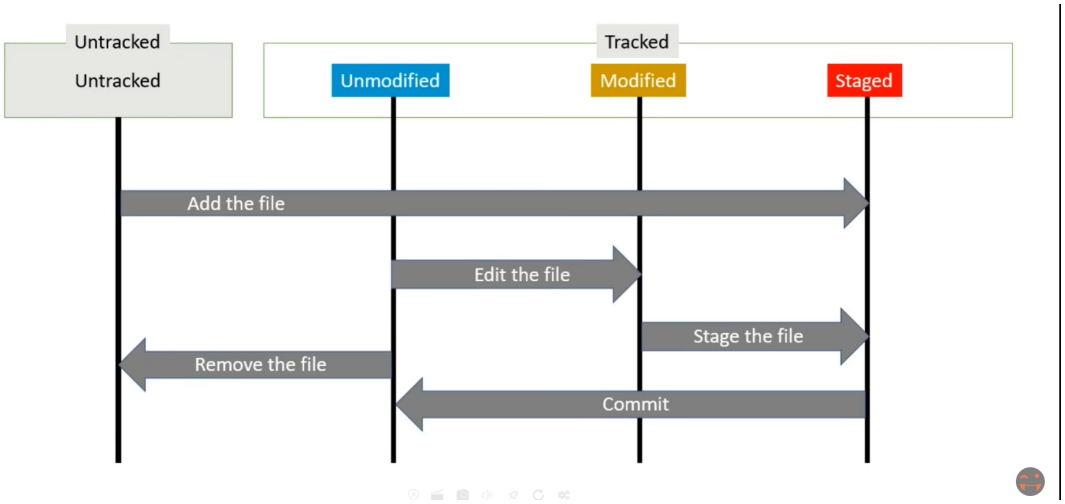
git clone **<repo-url>**
git clone **<repo-url> <preferred-name>**
The repository which we are cloning becomes a folder (same name by default, can be changed) in our local device and the branch of the repository is MASTER (by default).

Initializing GIT REPO in existing Folder:

git init
A hidden folder .git is created once the existing folder is initialized to a git repo

STATUS LIFE CYCLE :

All the files when initialized will be **Untracked** files and when **git add** is used then they move to **Staged** state and then when **git commit** is used they move to **Unmodified** state and are ready to be pushed into Github repository. When changes are made in a file at the unmodified state it moves to the **Modified** state and when **git add** is used it moves to **Staged** state and the cycle continues…



GIT IGNORE:

- Create a .gitignore file(text document) using **notepad .gitignore** command
- Enter accordingly
    - Specific file : File_name
    - File Pattern : .txt, .html
    - But exclude this file : !hello.txt
    - Folders : Folder_name/

GIT DIFF:

It shows the content difference at modified state and unmodified state/Staged state.
git status always shows the files that are modified whereas the git diff shows the content modified.

GIT COMMIT:

git commit -m **"Your Message".** Each commit is considered as a snapshot
The first commit done to the project is known as root commit.
Each commit has a commit Id which is of 40 characters and is generated internally.
Clears the working tree.

GIT LOG:

Used to know the previous commit history of the project.
One can see the commit Id when **git log** command is used
Shows the entire commit History in reverse chronological order.

Author : The one who created the project
Commit : The one who made the commit

BRANCHING:

MASTER

Default Branch incurred automatically before we perform any further task.
It is incurred when we clone a git repo or when we initialize a local repo.

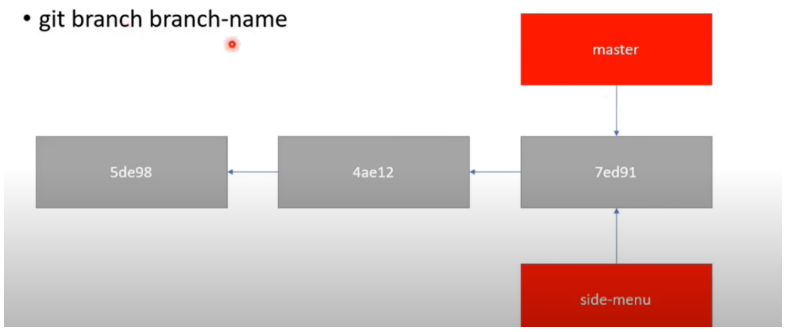
Each file will have a specific blob which is contained by a tree. The information we had until the point of commit gets stored in each commit.

The First commit is the parent of all the commits and the rest of the commits are binded in parent-children relation consequently. Suppose A,B,C,D are 4 commits made and A being the first commit then :

A ← B ← C ← D

Each Commit is Linked with the previous one .

Branches are the ones that point out the recent commit. **HEAD** is a pointer which points out the current branch we are in.



Switching branches can change the files in the working directory.

Generally changes aren't made in the master branch. Changes to a file/module are made in some other branches and then merged into master branch

MERGING:

Fast Forward Merge : when two commits are merged and they are derived from the same ancestor then it's a fast forward merge.

Recursive strategy : when two commits are merged and they are derived from a different ancestor then Recursive strategy is implemented in merging.

MERGE CONFLICTS:

When changes(commits) are made at the same location in two different branches, when they are merged a merge conflict arises.

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
        This is all about blogging, sharing, expressing one's views with Koushik GARA @sklm
=======
        This is all about blogging, sharing, expressing one's views with Koushik @vizag
>>>>>>> side-menu (Incoming Change)
```

Current change : change made in current branch
Incoming change : change made in another branch

Here changes are made in the same file and at the same line hence a merge conflict arises; when the two branches are tried to merge.

Now , to resolve the conflict either of the changes can be considered.

REMOTE BRANCHING:

When we clone a repository we obtain some remote references. These remote references remain static and any changes made will create local commits