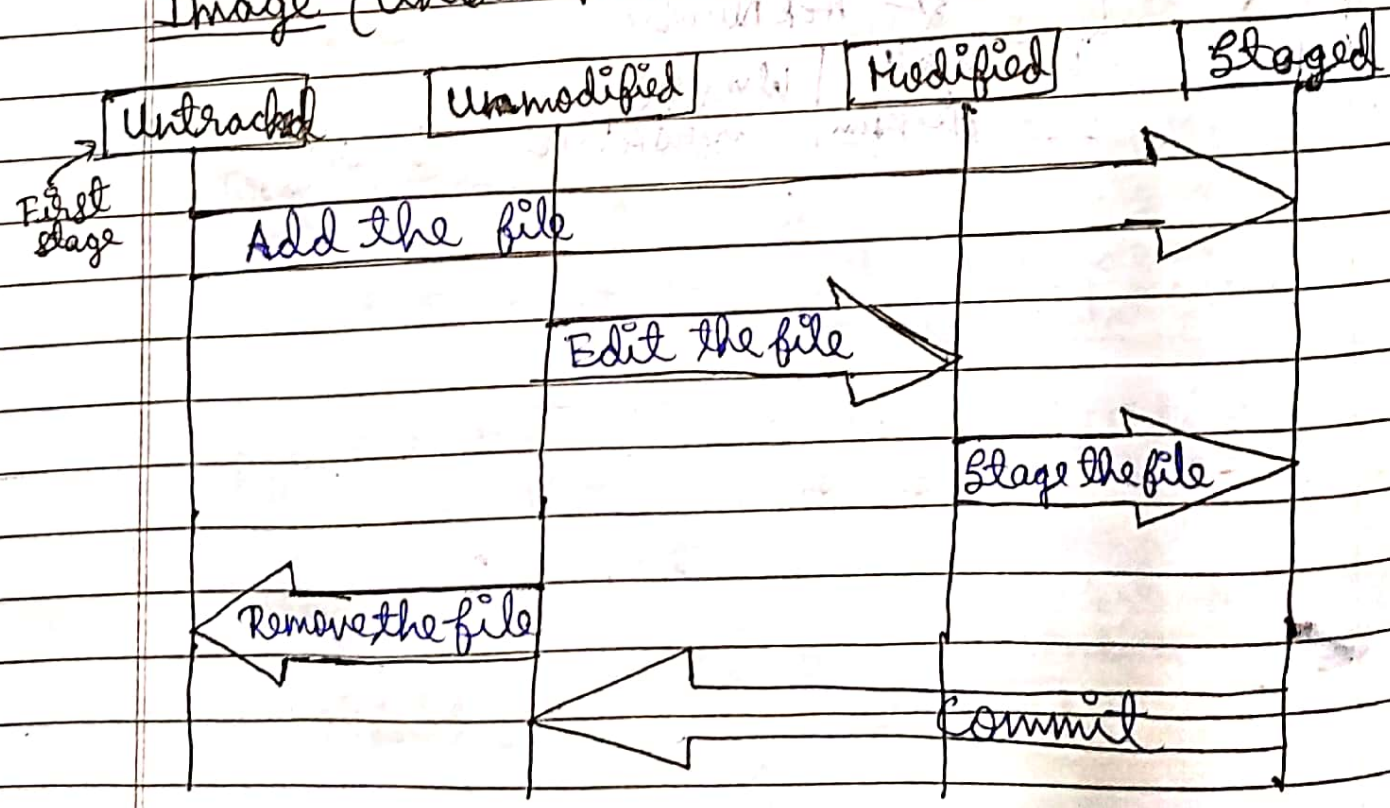
Git and Github

Page No.

→ Git is a modern and widely used open source software distributed version control system in the world.
→ It is a software developed to manage projects with high speed and efficiency.
→ It tracks the history of changes as people and teams collaborate on projects together.
→ As developers make changes to the project, any earlier version of the project can be recovered at any time.

Github:→ Github is a Git repository hosting service.
→ Github also facilities with many of its features, such as access control and collaboration. It provides a web-based graphical interface.

Image (Untracked and Staged File)



Commands

1) Git config :- → First and necessary command used on the git command line.

→ It configures the user name and email address to be used with our commits.

Command :- $ git config --global user.name "Sandy"

$ git config --global user.email "email"

To see :- $ git config --global user.name

$ git config --global user.email

2) git init command :- → It is used to create a local repository.

→ The init command will initialize an empty repository.

Command :- $ git init

→ .git folder is created which is hidden

Note :- We can see hidden folders by command :-

ls -lart :-

3) git status :- → This command is used to display the state of working directory & staging area.

→ It allows us to see which changes have been staged, which haven't and which files aren't being tracked by git.

→ It also lists the files that were changed and those you've still need to add or commit.

Command:- $ git status

4) git log :- → This command is used to check the commit history.
→ 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.

Command:- $ git log
& $ git log -3 or $ git log -p -3 → shows the local of added, removed & updated lines
shows descriptions

5) touch command :- → touch command is used to create empty files.
→ We can create multiple empty files by executing it once.

Syntax:-
touch <filename>
OR
touch <file1> <file2> ...

6) Add Command :- This command is used to add one or more files to staging area.

Syntax :- To add one file
$ git add <filename>

To add more than one file
$ git add .
OR
$ git add -A

7) revert command :- → The git revert command is used to apply revert operation.

Page No.
Date :

→ It is an undo type command

→ It helps in to revert back to previous commit.

(Suppose we have made a change to a file, say, 'files.txt' of our project. And later, we remind that we have made a wrong commit in the wrong file. Now, if we want to undo the changes then we can do so.)

Syntax :- To revert a commit, we need the commit reference id. We can get that by git log command.

i.e. $ git revert <commit-ish>

↑ reference id of commit by git log command

8) Remove Command :- → git rm command is used to remove individual files or a collection of files

→ The key function of git rm is to remove tracked files from the git index (staging area)

Syntax :- If we want to remove the created directory from our repository :-

$ git rm -rf <dirname>

& If we want to remove the file from our repository :-

$ git rm <file Name>

→ it removes the file from both locations i.e. git staging area & from present working directory

(remove only from staging area but present at hard disk)

& $ git rm --cached <file Name>

→ (It will remove only from staging area)

9) git commit :- → It records or snapshots the file permanently in the version history with a message.

(It is the next command after the git add.)

↳ It is used to record the changes in the repository which has done in staging area.

→ Two different commits will never overwrite because each commit has its own commit-id.

Syntax :- $ git commit -m "Message"

10) git checkout :- → git checkout <filename> command is used to revert or discard the untracked and uncommitted changes made to the specified file in the current repository.

→ It simply takes you back to the last commit which we made from modified changes.

Syntax :-

$ git checkout <file name>

To For multiple modified files & we want the files (which are modified) back, then

$ git checkout -f

11) git diff :- → compares working tree with staging area (compare)

→ It is used to show changes b/w working tree and staging area

Syntax :-

$ git diff

(compares staging area with last commit)

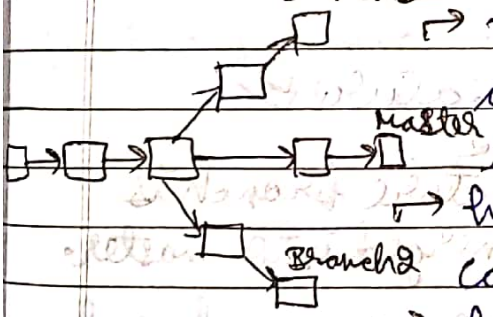
Note :- git diff --staged will only show changes to files in the staged area

12) git Ignore :- When sharing your code with others, there are often files or parts of your project, you don't want to share.

Examples :- → log files (It includes startup messages, system changes, unexpected shutdowns, errors & warnings and other important processes.)
→ temporary files
→ hidden files
→ personal files

- Git can specify which files or parts of your project should be ignored by Git using a .gitignore file.
- Git will not track files and folders specified in .gitignore.
  However, the .gitignore file itself is tracked by Git.

13) git branch :- → A branch is a new/separate version of main repository.



→ Branches allow us to work on different parts of a project without impacting the main branch.
→ When the work is complete, a branch can be merged with main proj branch.
→ We can even switch b/w branches and work on different projects without interfering each other.
→ Branching in git is very lightweight and fast!

Syntax :- git branch <branch Name> → branch is created

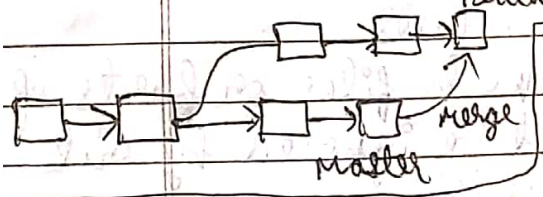
& We can check listing of all branches as :- git branch

Also, To switch to newly created branch
git checkout <branch Name> → this becomes our current branch

Note: To directly add & switch to newly created branch, we use

git checkout -b <branch Name>

14) git merge :→ Git merge is used to combine two branches.



→ It joins two or This command facilitates us to take the data created by git branch and integrate them into a single branch.

→ git merge will associate a series of commits into one unified history.

Steps:

1) git checkout master

2) git merge feature

feature branch is merged into master.

15) git remote :→ Remote repository is a shared repository where all team members use to exchange their changes.

→ A remote repo. is stored on a code hosting service like an internal server, github, subversion, and more.

→ The developers can perform many operations with the remote server. These operations can be a clone, fetch, push, pull and more.

(If we have added different URL by mistake in git remote add origin command Then, we can remove it by command: git remote remove origin)

Syntax:

git remote add <shortname> (origin generally) <remote URL>

Note:- To change Remote's URL (say from HTTPS to SSH → for private)
Use $ git remote set-url <remote name> <new URL>

16) git clone:- → Cloning is the act of making a copy of any target repository.
↳ can be remote or local
→ git clone is a command-line utility which is used to make a local copy of a remote repository.
→ original repository is located on a remote server, often from a git service like Git Hub, Bit Bucket or GitLab.
→ The remote repo. URL is referred to the origin.

Syntax:- git clone <repo URL>

17) git push:- → The push term refers to upload local repository content to a remote repository.
→ git push command is used to push into remote repository.

Syntax:-
$ git push <option> <Remote URL> <branch Name>
Eg:- git push -u origin master

-u means always pushes in master branch & we can directly write git push without writing (origin master).