**Version control system / Source code management system :-** It is the system which track your changes, maintain proper log of your changes, which allow multiple developers to work simultaneously.

**Types of Version control system:-**

1. Local version control system
2. Centralized version control system (ex:- SVN)
3. Destributed version control system (ex:- git)

**Architecture of Version control system :-**

A picture containing screenshot

Description automatically generated

**Architecture of git:-**

A screenshot of a cell phone

Description automatically generated

Notes:-

1. Staging area is a virtual layer between working directory and local repo.
2. Working directory and local repo are physically same but logically different

Ways to create local repository :-

1. Get copy from remote repo
2. Manually create local repo

**LABS :-**

1. Install git
2. Go to specified folder location where you want to store the project
3. Initialize the git :- **git init**

The above command will create .git folder inside your directory

1. Create some files (ex:- index1.html)
2. **git status**

Note :- If status shows as untracked, this means files are in working directory but not added to git( i.e in staging area)

1. **git add file\_name** (ex:- git add index1.html)

Note :- the above command will add your file to staging area

1. **git status**

Note :- If status shows as Changes to be committed, this means files are in staging area but not committed to local repository.

1. **git commit -m "some commit message "** (i.e git commit -m "First Commit ")

Note :- this command commit the file to our local repository

1. If it asks for credential, execute below two command otherwise you can ignore

**git config --global user.name "FIRST\_NAME LAST\_NAME"**

**git config --global user.email "MY\_NAME@example.com"**

1. **git status**

Note :- If status shows as nothing to commit, working tree clean, this means files are committed to local repository.

1. To see the files in local repository, execute below command

**git ls-files**

1. To see the history of command already executed use below command

**history**

1. To see the commit history, execute the below command

**git log**

1. To see only commitId and message, execute below command

**git log --oneline**

1. If you want to see any particular commit, execute below command

**git show commitId** (ex:- git show 1def101)

1. Change some file contents which are already committed and execute **git status**

Note :- It show status as Changes not staged for commit. Again execute git add file\_name and **git commit -m "message"** or **git commit -a -m "message"**

1. To see the unstaged changes ( you have modified the file but not added to git, it will compare working directory file with staged file) execute the below command

**git diff file\_name** (ex:- git diff file1.txt)

1. To see the staged changes ( you have modified the file ,added to git, it will compare staged file with local repository) execute the below command

**git diff --staged file\_name** (ex:- git diff --staged file1.txt)

* **git command to compare files with different scenario**
* To compare the working directory vs last staged area changes
  + $ git difftool
* Compare working directory vs local git (last commit in current branch)
  + $ git difftool HEAD
* Compare staged area vs local git repo last commit
  + $ git difftool --staged HEAD
* For limiting file to choose one file out of many file (working area vs staging area)
  + $ git difftool - -filename
* To see the commit log
  + $ git log - -oneline
* To see differences between any of the two commits.
  + $ git difftool commitID1 HEAD (HEAD means last commit in current branch)
  + $ git difftool HEAD HEAD^ (HEAD vs HEAD^ means last commit in current branch vs 2nd last commit in current branch)
  + $ git difftool commitId1 commitId2 (compare commit with commitId1 vs commitId2)
* Compare from central(origin/master) to local git(master)
* $ git difftool master origin/master

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. To remove the file from Working directory as well as from Local repository, execute below command

**git rm file\_name** (ex :- git rm file1.txt)

1. To remove the file from Local repository but not from Working directory, execute below command

**git rm --cached file\_name** ( ex:- git rm --cached index.html)

Note:- Here we have deleted file from local repo but its still available in working directory, so when you will do **git status** , it will show the status as Untracked files.

To solve this create one file **.gitignore** and write the deleted file name or the file name which you want to ignore inside .gitignore

1. To revert the changes, execute below command

**git revert commitId**

1. Git works with pointers, HEAD is the pointer in git which always points the top most commit in the log history

A screenshot of a cell phone

Description automatically generated

Now in above picture I want my HEAD should point 1def101 Second Commit, by doing so we will lose all the log history and all changes above that line. We can achieve this by doing reset at commit level. Execute the below command to do this

**git reset --hard commitId** ( ex:- git reset --hard 1def101 )

Note:- After executing above command by any means we can’t revert the changes. It is called as destructive command. This command will not generate any commitId

1. Normally for different requirement or for different releases we maintain different branches. Default branch is master. To check the list of branch, execute the below command

**git branch**

1. To create a new branch execute the below command

**git checkout -b new\_branch\_name old\_branch\_name**

1. To merge the code from one branch to other branch execute the below command

**git merge source\_branch\_name destination\_branch\_name**

**ex:-** git merge NEW\_BRANCH master

here from NEW\_BRANCH we are merging to master

source\_branch\_name :- from where you want to merge

destination\_branch\_name :- on which branch you want to merge