Repository:

> Repository is a place where you have all your codes or kind of flder on server.

> Changes are personal to that repository.

Server:

> It store all repository, it contains metadata also.

Working Directory:

> Where you see files physically do modification.

> At a time you can work on a particular branch.

In other CVCS, developers generally makes modifications and commit their changes directly to the repository. But git (DVCS) uses a d/f strategy. Git does not track each and every modified file. Whenever you do commit an operation, git looks for a file

in staging area. Only those files present in the staging area are considered for commit and not all the modified files.

Working Directory > add > Staging area > commit > Working Repository > Push> github

Commit:

>Store changes in repository, you will get one commit-ID.

> It is 40 alpanum character.

> It users SHA-1 checksum concept.

> Even if you change one dot commit-ID will change.

> It actually helps you to track the changes.

> Commit is also names as SHA1 hash.

Commit-ID/Version-ID/Version:

> Reference to identify each change, to identify who changes the file.

Tags:

> Tags assign a meaningful name with a specific version in the repository. Once a tag is created for a particular save, even if you create a new commit, it will be updated.

Snapshot:

> Represents some data of particular time.

> It is always incremental i.e. It stores the changes appended data only.

Push:

> Push operator copies changes from a local repository instance to a remote or central repository. This is used to store the changes permanently into the git repository.

Pull:

> Pull operation copies the changes from the remote repo to a local machine. The pull operation is a used for sync between two repo.

Branch:

> Product is same,

yum install git -y : To install git

which git : To know path of git

git diff : To see the modification happen in any file from it's previous version

git version : To know the version

git config --global user.name "ashish" : To config profile

git config --global user.email "ashishlodhi09@gmail.com"

git config --list : To list all config

git status : To check the status if the code is working area or pushed to staging area

git add . : To add file from working directory to staging area

git commit -m "message" : To add file from staging area to local repository

git log : To know the commit id and log who made those changes

git reset -hard <commit version can be copied from log> : To go to previous version of commit in the file the file will remove the code added in the file from the version selected in the command.

git log -1 : To show the last commit only

git log --oneline : To show all commits in one line

git log --grep "Word" : To find that commit where you have added this "word" in it's comment while commit

git show <commit id> : To show details what commit has been done to code base

git remote add origin <git url> : To add local repository to remote repo

--> git remote add origin https://github.com/ashish0908/centralgit.git

\*\* It will not work github removed password authentication use access token

git branch -M main : To create branch if it is a new repositroy

git branch < firstbranch > : To create a new branch to work on in local repository

\*\* To push local branch to github repository

1. git checkout firstbranch
2. git push origin firstbranch

\*\* Now to merge the new updates of firstbranch to master of git

1. git checkout master
2. git pull origin master: Pull the latest changes (important if others are working on it)
3. git merge firstbranch: Merge firstbranch into master
4. git push origin master: Push the updated master to GitHub

git push -u origin main/master: Whatever origin we will add it will push the code from local repo to that remote repo

git pull origin main/master: To pull code from repo

git remote set-url: For changing your remote URL

git merge origin/main : To merge the fetched changes into your branch

git rebase origin/main : To incorporate the fetched changes by applying your local commits on top of the updated main branch, you can use rebase.

--------------------------------------------------------------------------------------------------------------------------------------------------

To ignore the files while commiting name a file define all the files format you don't want to be get committed.

create a file and define all the format in that file you want to ignore

> vim .gitignore

\*.java

\*.css

~

~

:wq

> git add .gitgnore

> git commit -m "ignore file formats in this file"

> git push

All the files will be pushed to remote repo except the format mentioned in this (.gitignore) file

\* ctrl+L : to clear screen

---------------------------------------------------------------------------------------------------------------------------------------------------

\* Untill we add files from working area to staging area we will be able to see that file from all branches once we add that to staging area of any particular branch we will not see it in any other branch.

git branch : To see list of available branches (\* Represent the current branch you are working on)

git branch <branch name> : To create a new branch

--> Whenever we create a branch all data which is there in main branch will get copied to it's feature branch just do "ls" to check after you checkout to feature branch

git checkout <name of branch you want to switch> : To switch branch

\* Merge : We use pulling mechanism to merge branches (Ex > (why) If we have created a feature branch and worked on it now we want to add that to main branch we will just pull the new files/code from feature branch not the complete files as while creating branch we have already copied everthing from main branch to feature branch)

git merge <branch name> : When we want to merge any feature branch to main branch/another feature branch just go to the branch where where you want the feature branch to get merged then execute this command.

\* If you want any feature branch to get pushed to central repo after doing commit execute (below cmd) then it will show pull request url open that in browser merge it from browser.(\*doubt)

git push --set-upstream origin <featured branch name> : To push featured branch to central repo

\* Git conflict : When same name file have different content in different branches and you do merge resolve conflict then add/commi.

\* Git stashing : When we have a requirement where we have to clean our working area for some other code and we want that we can continue with the current code after sometime. We can put the current work to stashing area. Later when required we can start working on it again.

git stash : To stash an item

git stash list : To see the list of stashed items

git stash apply stash {Number of file} : To get file from stashed area to working area (It will copy/paste)

git stash clear : To clear the stash item from the stashed area after we added and commited that file and it's not required to be saved.

git reset <filename>: If we have added the file to staging area and now we don't want to commit it for a single file. Once executed that file will not appear again in working directory.

git reset. : Same as above but for all files. (Done on private file)

git reset -- hard : To remove the file from both staging area and working area in once command.

\* Git revert : Helps to undo an existing commit.(Done on public file) It does not delete any data in this process instead. Rather git creates a new commit with the included files reverted to their pervious state. So your version control history moves forward while the state of your files moves backward.

git revert <commit id> : To skip that particular commit. -->> Once executed add a message why to skip this commit then save.

git clean -n(dry run) / -f(forcefully) : To remove untracked files it will delete the files which we don't want to commit or not added to staging area.

\* Tag : Tag operation allows giving meaninful names to specific version in the repository

git tag -a <tagname> -m "<message>" <commit-id> : To apply tag

git tag : To show all tags

git show <tag name/commit-id-a > : To see particular commit content by using tag

git tag -d <tag name> : To delete a tag

\* Github clone: The repo which is already there in central repo will get copied to your local repo with same content and name.

git clone <url> : To clone repo to local machine