File System Hierarchy

/home → home directory for other users.
/root → It is home directory for root user.
/boot → It contains bootable files for Linux. eg → initrd
/etc → It contains all configuration files.
/usr → by default software are installed in this directory
/bin → It contains commands used by all users.
/sbin → It contains command used by only root user. Including root user.
/opt → Optional application software packages.
/dev → Essential device files. This include terminal devices, usb or any device attached to the system.

How to create a file

- cat
- touch
- vi/vim
- nano

vi/vim, nano → editor

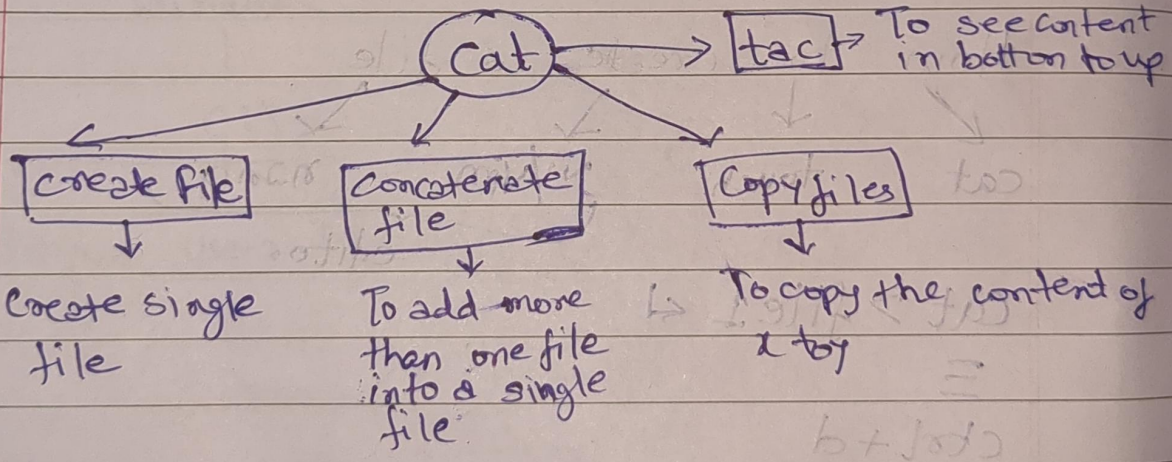
cat > file1 ↵
=
ctrl+d

touch → to create empty file

Launch Amazon Linux
Acces through putty
Login as : ec2-user
[ec2-user @ip]$ sudo su ↵
↑ Superuser do switch user
user

[root @ip]#
↑
root

Cat Command
The cat command is one of the most universal tools, yet all it does is copy standard input to standard output.



[root@ip]#cat > file1↵
What is this
How are you
[Ctrl+d]
[root@ip]# ls ↵
o/p⇒ file1

[root@ip]# cat > file2↵
Hello → ctrl+d
[root@ip]# ls↵
o/p⇒file1 file2
[root@ip]# cat >> file1
Thankyou ctrl+d
Add data in existing file

[root@ip]# cat file1
What is this
How are you
Thankyou
[root@ip]# cat file1 file2 > all
Combine both data
[root@ip]# cat all
What is this
How are you
Thankyou
Hello

Touch Command → update only access time, modify time of file.

- Create an empty file
- Create multiple empty file
- Change all timestamp of file

File 1 stat file 1

Access Time
Modify Time
Change Time

Time stamp → Chang Time (Last time when file metadata was changed)

- Access Time (Last time when a file was accessed) — touch -a
- Modify Time (when a file was modified Lastime) — touch -m

```
[root@ip]# touch file1↵
[root@ip]# ls
o/p file1
[root@ip]# touch file1 file2 file3 file4↵
[root@ip]# ls
o/p file1 file2 file3 file4
[root@ip]# touch file2↵
```

↑ to change all time stamp

```
[root@ip] stat file2↵
```

[root@ip]# touch -a file3
↑ change only access time
[root@ip]# touch -m file3
↑ change only modify time.

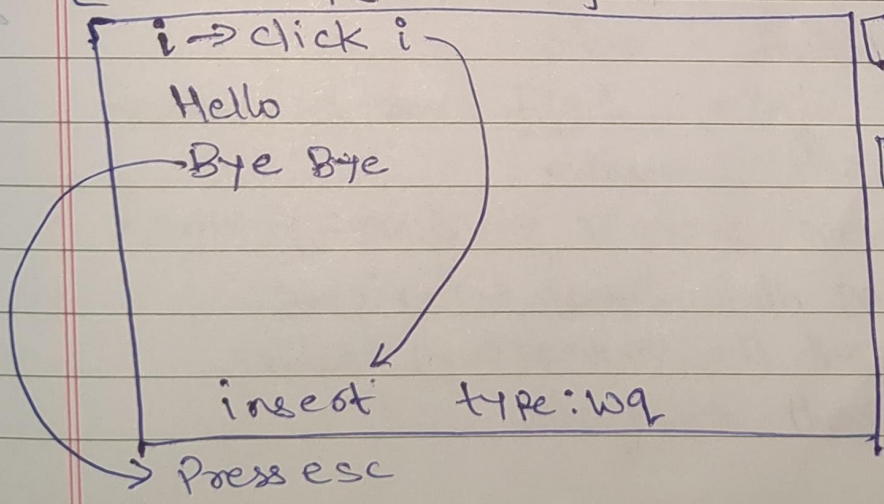
Vi editor

→ A programmer text editor
⇒ It can be used to edit all kinds of plain text, it is specially useful for editing programs mainly used for unix programs.

Note: :w → to save
:wq or :x to save & quit
:q → quit
:q! → force quit, no save

'Vi' is a standard whereas 'nano' has to be available depending on the linux you use.

[root@ip]# vi file a ↵



[root@ip]# ls
o/p file a
[root@ip]# cat file a
Hello
Bye Bye

Nano Command

[root @ ip] # nano file b ↵

Ctr + X → Y → ↵ (enter)

[root@ ip]# ls

file b

[root@ ip]# cat file b

Ctrl + O

↓ for overwriting

[root @ ip]# ls -l or ll

↓ gives details of all file

[root@ip]# ls - a → shows hidden files

[root @ ip] # history

↓ shows all commands we ran that day

How to create a directory
[root@ip]# mkdir dir1 ↵
[root@ip]# ls ↵
[root@ip]# mkdir -p dir1/dir2/dir3
[root@ip]# pwd → print working directory

How to copy a file
How to cut & paste file
How to rename file or directory
How to create hidden file or directory
How to remove file or directory
⇒ Try some command like less, more, Head & Tail.

Hiden file → [root@ip]# touch .file1
[root@ip]# ls -a → shows hidden files

Copy a file → [root@ip]# cp file1 file2
file1 ↓ source
file2 ↓ destination

Cut & Paste → mv file1 dir1
file1 ↓ source
dir1 ↓ destination

Rename → mv file1 myfile
file1 ↑ oldname
myfile ↑ newname

Remove → rmdir → remove specified directory
rmdir -p → remove both parent & child directory
rmdir -pv → removes all the parent & subdirectory along with the verbose.

classmate
Date
Page

rm -rf → Removes even non-empty file & directory
rm -rp → Removes non-empty directories including parent & subdirectory
rm -r → Removes empty directory

[root @ ip] # echo "Hello"
[root@ip] # echo "Welcome" > filez
[root@ip] # cat filez
Welcome
[root@ip] # echo "Namaste" >> filez
[root@ip] # cat filez
o/p Welcome
Namaste
[root@ip]#echo > filez , empty file
[root@ip]# cat filez
o/p → nothing

[root @ ip]# grep → find specific word present

[root@ip]# grep root /etc/passwd
↓ find where root is present

[root@ip]# sort
Rishabh
Welcome
India
Software
o/p India
Rishabh
Software
Welcome

Commands

useradd → To create user

groupadd → To create group

gpasswd -a/-M → To add user into group, to add multiple user

ln → hardlink → for backup

ln -s → softlink → shortcut

tar → Tar is an achiver used to combine multiple files into one

tape archive

gzip → gzip is a compression tool used to reduce the size of a file.

wget → wget is the non-interactive network downloader.

passwd aws → To add password

[root@ip]#useradd rishabh → to add user

[root@ip]# cat /etc/passwd → to see user

[root@ip]# groupadd rishabhgp → to add group

[root@ip]# cat /etc/group → To see group

[root@ip]# gpasswd -a rishabh rishaphgp

↓ add user into group ↑

[root@ip]# gpasswd -M rishabh, aman, sameer rishabhgp

↑ add multiple user to group ↑

[root@ip]# ln -s file1 softfile1

file1 ↓ Original file

softfile1 ↑ shortcut link

[root@ip]# ls -l

[root@ip]# ln file2 backupfile2

[root@ip]# ls

[root@ip]# tar -cvf dirx.tar dirx

forcefully

c → create

v → verbose

dirx/diry/dirz

[root@ip]# gzip dirx.tar

[root@ip]# ls

→ dirx.tar.gz

[root@ip]# gunzip dirx.tar.gz

[root@ip]# ls

→ dirx.tar

[root@ip]# tar -xvf dirx.tar → extract tar file

x → extract

[root@ip]# wget <url> ↵

wget → to download any file

[root@ip]# useradd rishabh
[root@ip]# gpasswd -M ajay,vikas rishabhgp
[root@ip]# cat /etc/group

Access Modes / Permissions

Access Mode		File	Directory
r	4	To display the content	To list the content
w	2	To Modify	To create or remove
x	1	To execute file	To enter into directory

Commands

chmod → Used to change the access mode of a file

chown → change the owner of the file or directory

chgrp → change the group of file

su - ansible → switch to ansible user.

userdel → Delete user

Userdel -r jenkins → delete user with home directory

groupdel → Delete group.

d | rwx | r-x | r-- 1 root root 16 July 01 04:40 dirx

4+2+1 | 4+0+1 | 4+0+0

7 | 5 | 4

[root@ip]# touch file1

[root@ip]# mkdir dirx

[root@ip]# ls -l → symbolic link

-rwx r-xr-- 1 root root 0 July 01 04:00 File

File or directory

owner or rootuser

group

other

owner

group

File size in bytes

r w x

↓ ↓ ↓

read write execute

d rwx rwx rwx

4+2+1 4+2+1 4+2+1

= 7 = 7 = 7

[root@ip]# chmod 777 dirx

- r-x -wx r--

4+0+1 0+2+1 4+0+0

5 3 4

[root@ip]# chmod 534 dirx

[root@ip]# chown rishabh file1

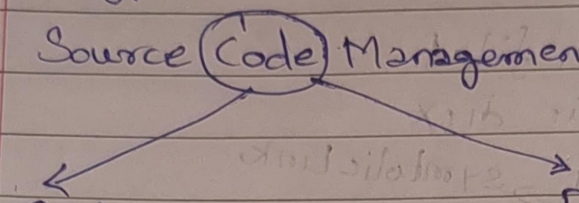
[root@ip]# chgrp devops file1

u=user/owner g=group o=other

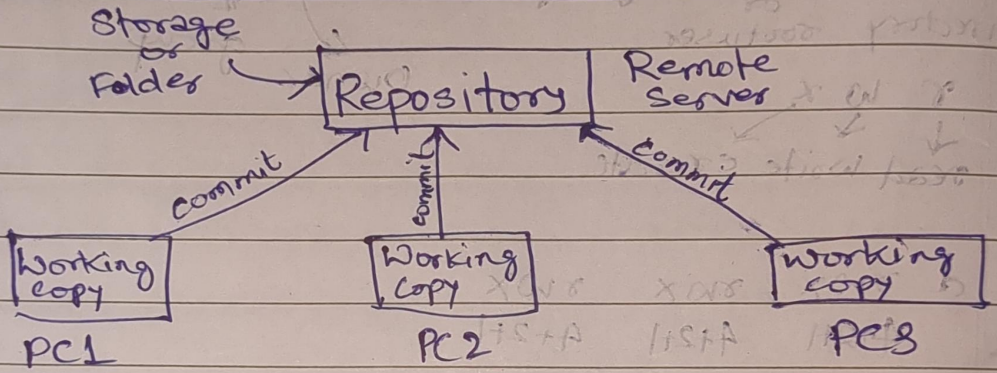
chmod u-wx, g+w, o=w file1

Introduction to Git

Software Configuration Management
or
Source Code Management



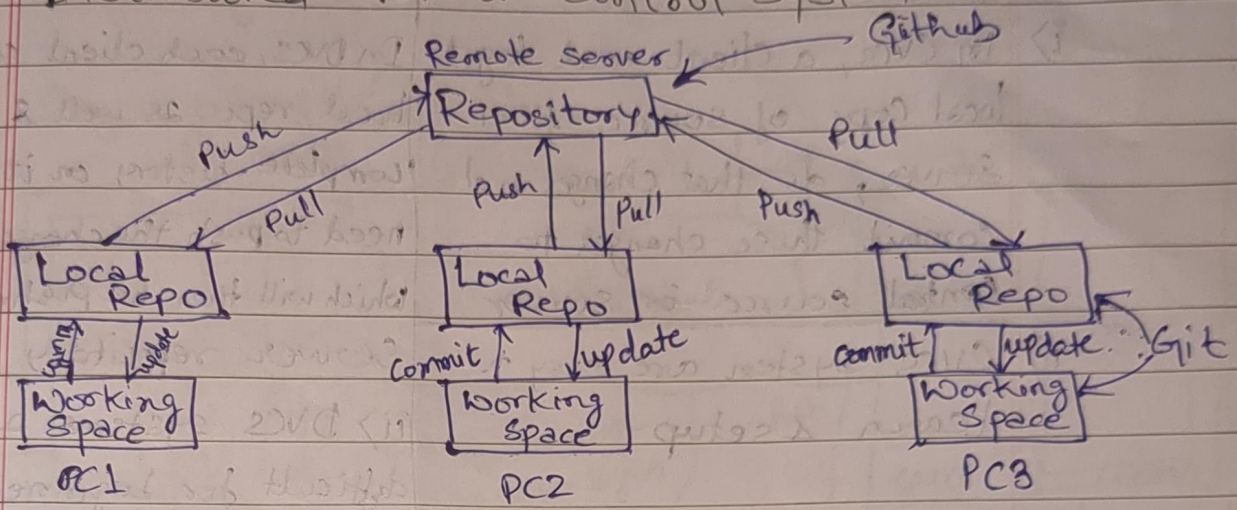
Centralized Version Control System



Drawbacks

1. It is not locally available, meaning you always need to be connected to a network to perform any action.
2. Since everything is centralized, if central server gets failed you will loose entire data. eg. SVN tool

Git → Distributed Version Control System

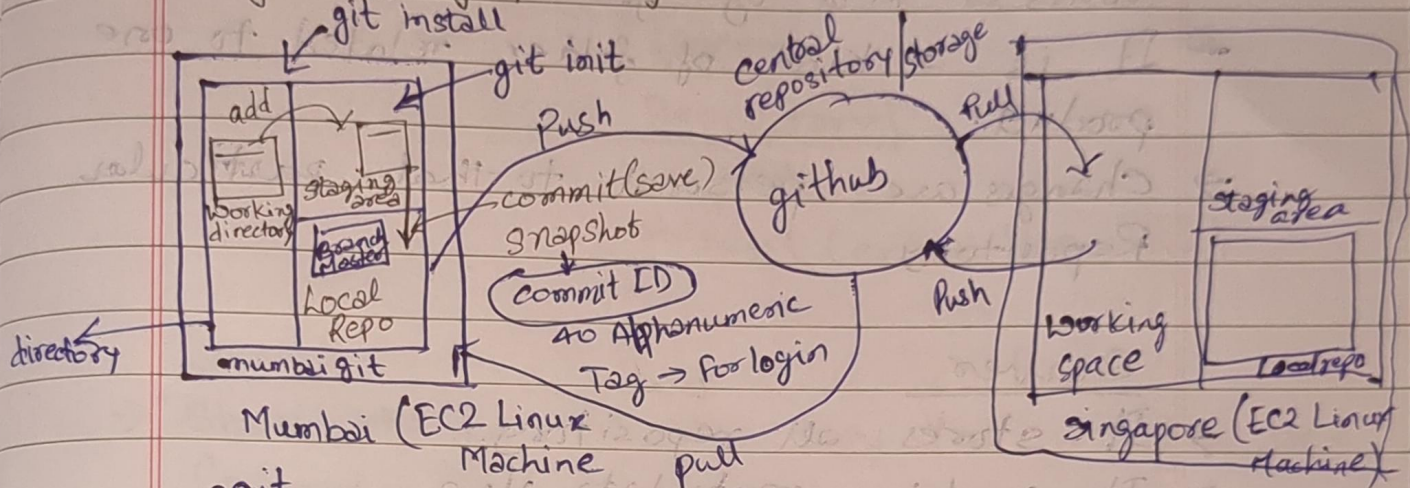


In distributed version control system, every contributor has a local copy or "clone" of the main Repository i.e every one maintains a local Repository of their own which contains all the files & metadata present in a Main Repository.

	CVCS	DVCS
i)	In CVCS, a client need to get local copy of source from server, do that changes and commit those changes to central source on server	In DVCS, each client can have a local repo as well & have a complete history on it. Client need to push the changes to branch which will then be pushed to server repository
ii)	CVCS system are easy to learn & setup	ii) DVCS systems are difficult for beginners multiple commands need to be remembered.
iii)	Working on branches is difficult developer often faces merge conflict	Working on branches is easier in DVCS developers faces less conflict.
iv)	CVCS system does not provide offline access	DVCS system are working fine on offline mode as a client copies the entire repo on the local machine
v)	CVCS is slower as every command need to communicate with server	DVCS is faster as mostly user deals with local copy without hitting server everytime.
vi)	If CVCS server is down developers cannot work.	If DVCS server is down developer can work using their local copies.

Stages of Git & its terminology

Stages of git/workflow



•git

After doing git init directory is converted into git repository or local repository

Stages

1) WorkSpace/Working Directory
2) Staging area
3) Local Repo

Repository

→ Repository is a place where you have all your codes or kind of folder on servers.

→ It is a kind of folder related to one product.

→ Changes are personal to that particular Repository.

Server

→ It stores all repository. local

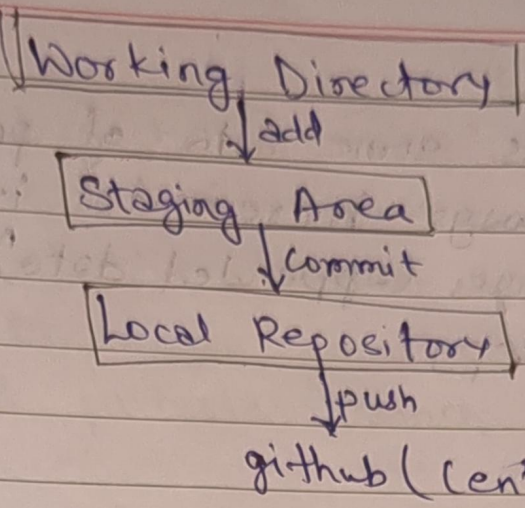
→ It contains metadata also.

Working Directory | WorkSpace

→ where you see files physically and modification.

→ At a time, you can work on particular branch.

In other cvcs, developers generally makes modification & commit their changes directly to the Repository. But git uses a different strategy. Git does not track each & every modified file. Whenever you do commit an operation git looks for the files present in the Stagin area only those files present in the staging area are considered for commit & not all the modified files.



Commit

→ Store changes in repository. You will get one Commit-ID.

→ It is 40 alpha-numeric characters.

→ It uses SHA-1 checksum concept.

→ Even if you change one dot commit-ID will also get 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 changed 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 not be updated.

Snapshots

→ Represents some data of particular time.
→ It is always incremental i.e It stores the changes (appended data) only not entire copy.

PUSH

Push operations copies changes from a local Repository instances to a Remote or Central Repo. This is used to store the changes permanently into the git repository.

Pull

Pull operations copies the changes from a Remote Repository to a local Machine. The pull operation is used for synchronization between two repo.

Branch

→ Product is same, so one Repository but different task.
→ Each task has one separate branch.
→ Finally merges (code) all branches.
→ Useful when you want to work pairallely.
→ Can create one branch on the basis of another branch.

→ Changes are personal to that particular branch.
→ Default branch is 'Master'.
→ File created in workspace will be visible in any of the branch workspace until you commit. Once you commit, then that file belongs to that particular branch.

Advantages of git

→ Free & Open Source.
→ Fast & small → as most of the operations are performed locally, therefore it is fast.
→ Security → git uses a common cryptographic hash function called Secure Hash function (SHA1) to name and identify objects within its database.
→ No need of powerful hardware.
→ Easier Branching → If we create a new branch, it will copy all the code to the new branch.

Types of Repositories

Bare Repositories (central Repo)

⇒ Store & Share only

⇒ All central Repositories are bare

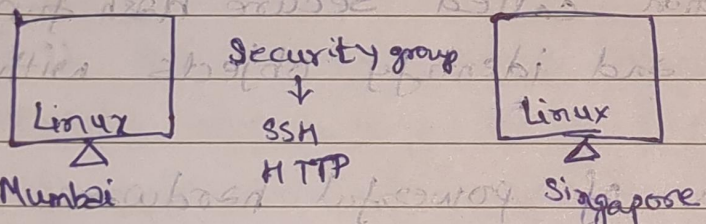
Non-Bare Repositories (Local Repo)

⇒ where you can modify the files.

⇒ All local repositories are non-bare repositories

How to use git and create github account

Create two EC2 instance (Linux)



⇒ sudo su → which git

⇒ yum update -y

⇒ yum install git -y

⇒ git --version

⇒ git config --global username "Rishabh/rish"

⇒ git config --global user.email "Rishabh@gmail.com"

⇒ git config --list

How to commit, push & pull from github

→ Login into mumbai EC2 instance
→ Create one directory and go inside it.
→ git init
→ touch myfile (put some data)
→ git status
→ git add .
→ git commit -m "1st commit from mumbai"
→ git status.
→ git log
→ git show <commit-id>
→ git remote add origin <Centralgit url>
→ git push -u origin master
(enter username & password)

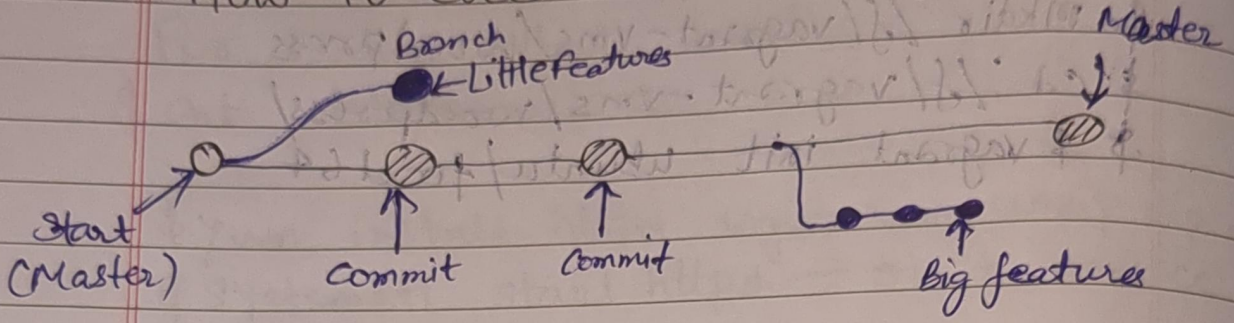
→ Now go to singapore EC2 instance.
→ Create one directory & go inside it.
→ git init
→ git remote add origin <github repo url)
→ git pull -u origin master
→ git log
→ git show <commit-id>
Now add some code in the file
→ git status → git add.
→ git commit -m "singapore update 1"
→ git status →git log →git push origin master.

To ignore some files while committing
Create one hidden file .gitignore and enter format which you want to ignore.
for eg → vi .gitignore
enter → *.css
*.java

→ git add .gitignore
→ git commit -m "latest update exclude css"
→ git status
→ Create some test, java & css files and add them by ignoring "git add"
for eg:- touch file1.txt file2.txt file3.java file4.css
→ ls
→ git status
→ git add
→ git status
→ git commit -m "my test files only"

git log -1 → last commit
git log -2 → last 2 commit.
git log --oneline → see all commit in one line.
git log --grep "ignore"
↓
see specific commit

How to create Branch, Merge & Stash



The diagram above visualizes a Repository with two isolated lines of development, one for a little features and one for a longer-running features. By developing them is Branches, its not only possible to work on both of them in parallel, but it also keeps the main Master Branch free from error.

⇒ Each task has one separate branch.
⇒ After done with code, Merge other branches with master.
⇒ The concept is useful for parallel development.
⇒ You can create any no. of branches.
⇒ Changes are personal to that particular branch.
⇒ Default branch is 'Master'

→ Files created in Workspace will be visible in any of the branch workspace until you commit. Once you commit then that files belongs to that particular branch.

→ When created new Branch, data of existing branch is copied to new branch.

To see list of available Branches
git branch

Create a New Branch
git branch <Branchname>

To switch Branch <Name of branch u want to go>
git log --oneline

Login to EC2 instance

$ sudo su

$ <mumbaigit> git log --oneline

$ git branch

* Master

current

$ git branch <branchname> → create new branch

branch 1

$ git branch

* Master

Branch 1

$ git checkout Branch 1

* git branch

Master

* Branch 1

* git branch -d Branch1 → To delete branch

-D

↓

forcefully delete

Merge

You can't merge branches of different Repositories.

We use Pulling mechanism to Merge Branches.

git merge <branchname>

To verify the merge

git log

To Push to Origin Central Repo like Github

git push origin master

Git conflict

When same file having different content in different branches, if you do merge, conflict occurs (Resolve confict then add commit)

→ Conflict occurs when you merge Branches

Open the conflict file in master branch

vi filename

Git Stashing

Suppose you are implementing a new feature for your product. Your code is in progress and suddenly a customer escalation comes. Because of this, you have to keep aside your new features work for few hours. You cannot commit your partial code and also cannot throw away your changes. So you need some temporary stage, where you can store your partial changes and later on commit it.

→ To stash an item (only applies to modified files. not new files).

→ To stash an item
$ git stash

→ To see stashed items list
$ git stash list

→ To apply stashed items
$ git stash apply stash@{0} → number
Then you can add & commit.

To clear the stash items
$ git stash clear

Git Reset

git Reset is a powerful command that is used to undo local changes to the state of a git rep

To reset staging area
```
git reset <filename>
git reset .
```
To reset the changes from both staging area & working directory at a time
```
git reset --hard
```

Git Revert

The revert command helps you undo an existing commit.

→ It does not delete any data in this process instead. Rather git creates a new commit with the included files reverted to their previous state. So, your version control history moves forward while the state of your files moves backward.

Reset → before commit
Revert → After commit.

→ sudo su
→ cd mgit
→ ls
→ git status
→ cat >newfile {create one new file}
→ Hi, Final code for app
→ git add .
→ git commit -m "code"
→ git log --oneline
→ git revert <commit-id>

How to remove untracked files
git clean -n (dry run)
git clean -f (forcefully)

Tags
Tag Operation allows giving meaningful names to a specific version in the repository
To apply tag
→ git tag -a <tagname> -m <message> <commit-ID>

To see the list of tags
→ git tag

To see particular commit content by using tag

→ git show <tagname>

To delete a tag

→ git tag -d <tagname>

Github clone

→ Open github website

→ Login & choose existing repository

→ Now, go to your linux Machine, & run command

→ git clone <url of github repo>

It creates a local Repo automatically in Linux m/c with the same name as in github account.