Software configuration Management
or
Source code Management

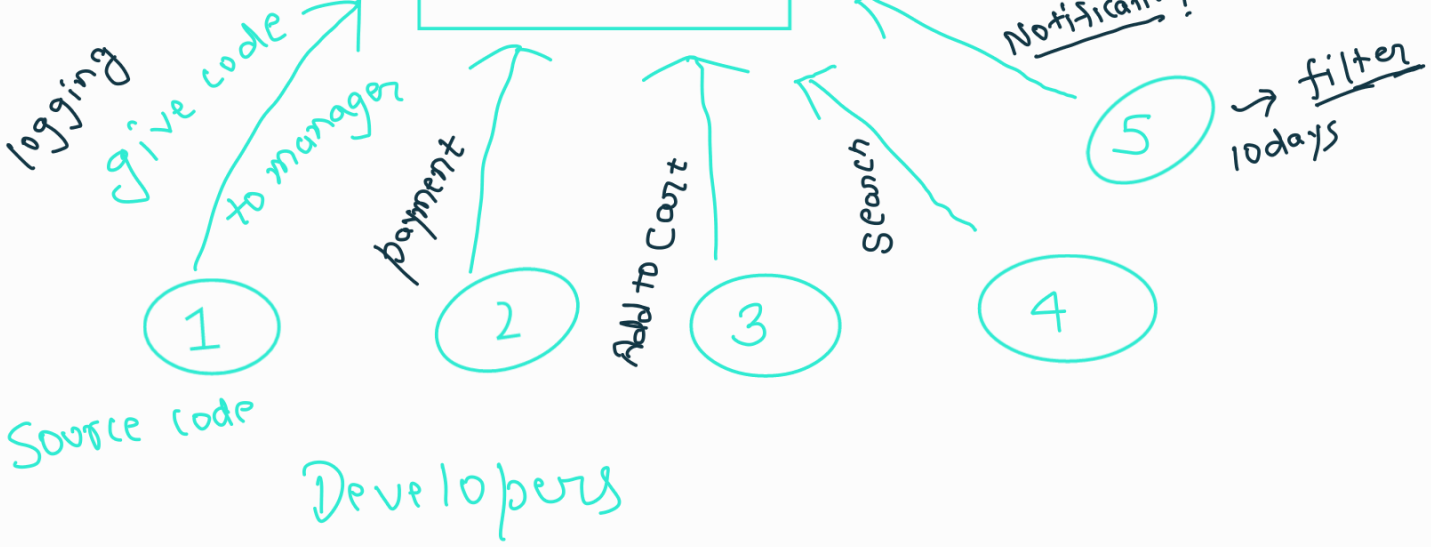
→ For designing a flipkart website, there will be many developers who are working on it.

→ One will work on logging page, one on payment page, one on cart page and so on.

→ Basically every developer will be writing code seperately for the task Assigned to them.

→ Previously for adding code of different-different developer, we have to do it manually.

{ Hectic job for manager.



Source code Management

(Practile to do work)

Centralised Version control System

(previous)

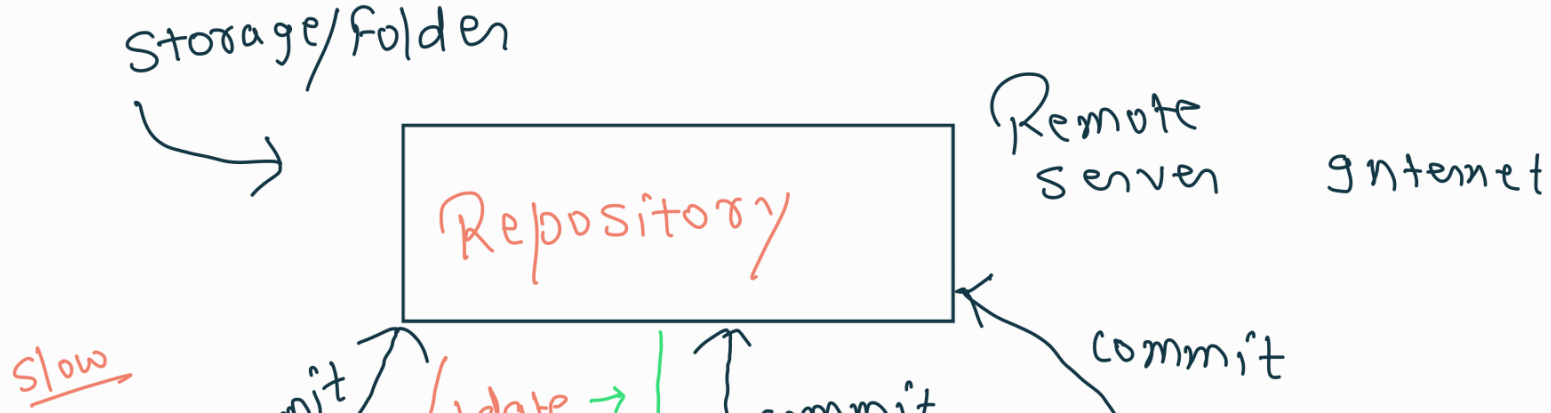
e.g → SVN

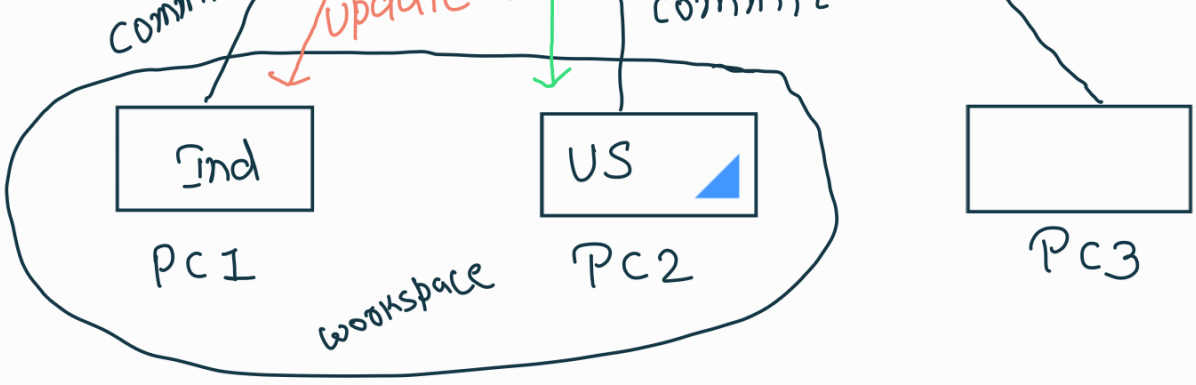
Distributed version control system

e.g → Git

Centralised Version control system (CVCS)

Slower





Drawbacks

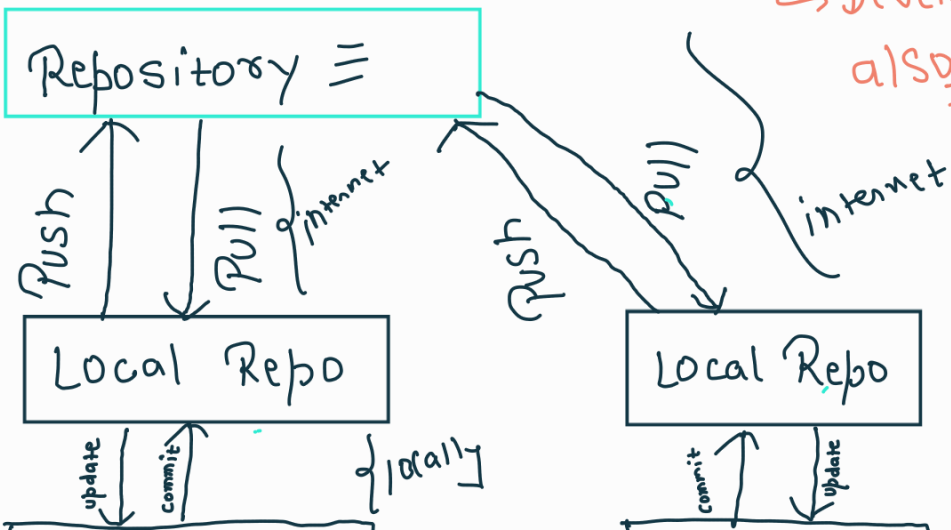
→ It is not locally available, meaning you always need to be connected to a network to perform any action.

→ Since everything is centralised, if central server gets failed you will loose entire data. e.g → SVN tool

Distributed version control system (Git)

microsoft
|
Github
on this level

→ Linus Torvald (2005)
→ Developed linux also



Git works here

↓
works on local machine.

→ git works on linux kernel.

→ git is a software.

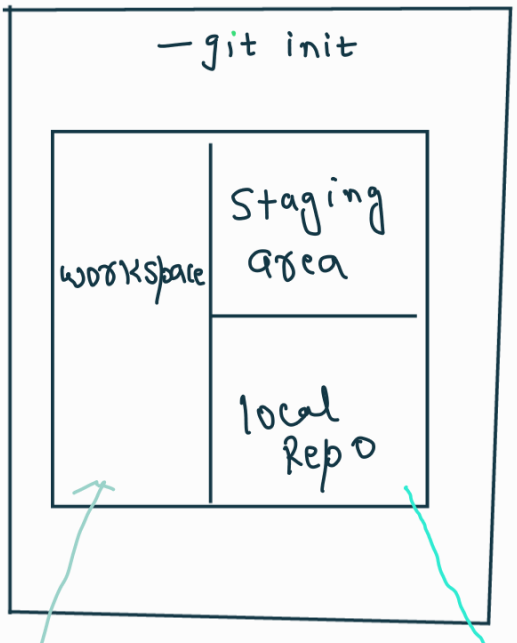
→ In distributed version control system, every contributor has a local copy or "clone" of the main repository i.e everyone maintains a local Repository of their own which contains all the files & metadata present in main Repository.

→ Maintains version of the code and every new push gives new commit id.

CVCS	DVCS
SVN	Git
→ Client get local copy of source from Server.	→ Each client have a local repo.

→ Easy to learn and setup.	→ Difficult for beginners Multiple commands needs to be remembered.
→ Do not provide Offline access.	→ works on offline acces as client have local repo.
→ Slower as every command need to communicate with Server.	→ faster as mostly user deals with local copy without hitting server.

Stages of git / workflow



Stage

(i) workspace / working directory

(ii) Staging Area

(iii) local Repo

Snapshot

↓

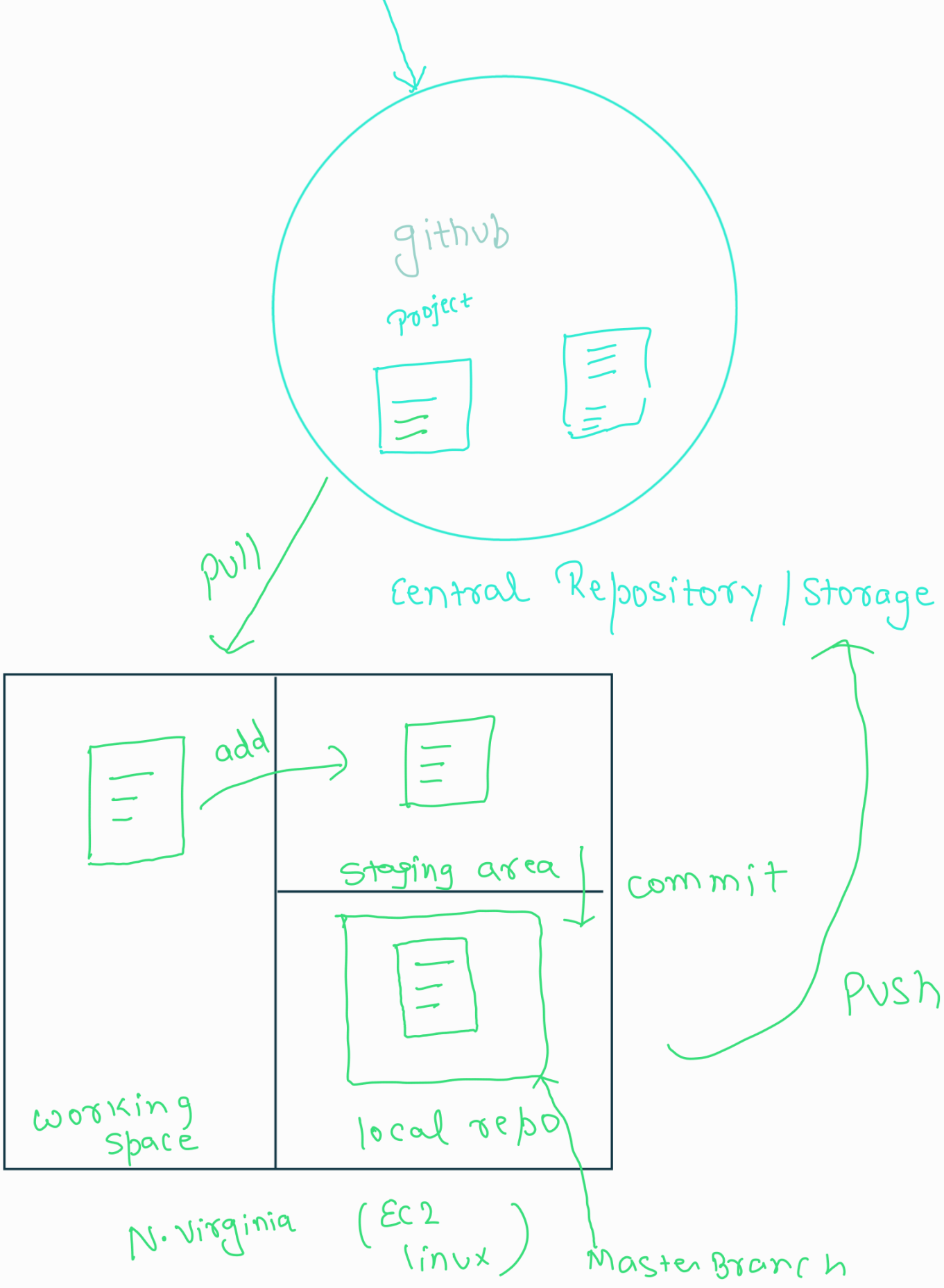
commit ID _ 40 alphanumeric

Mumbai (EC2 linux Machine)

Local Repository

.git

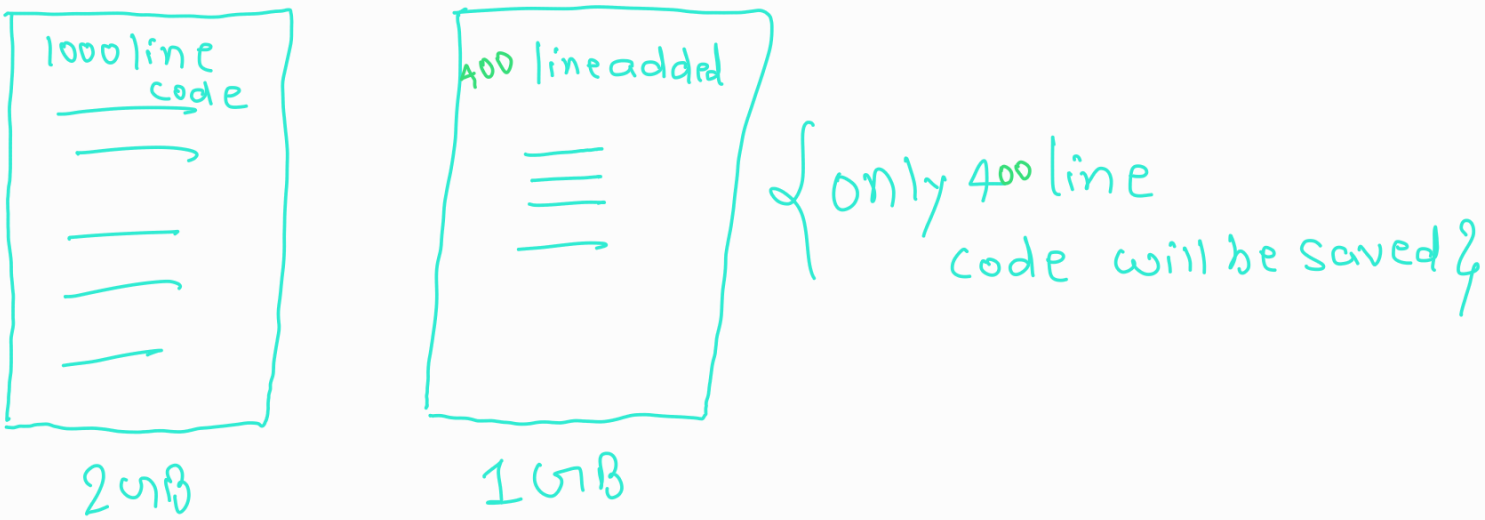
Push



Working Space → where we write code

Staging area → freeze the file which we want to commit.

local repo → where we save the snapshot of code.



3MB (previously) - now we are saving the 2MB

Repository :

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

→ changes are personal to that

particular Repository.

→ Can be local or central.

Server: Github

→ Stores all repositories.

→ It contains metadata also.

Working directory

→ where you see files physically and do modification.

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

Note:-

In other Central Version control System (CVCS), developers generally makes modifications and commit their changes directly to the Repository.
But git uses a diffirent Strategy.
Git does not track each and every

modified file. whenever you do commit an operation git looks for the file present in the 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
Local Repository
↓ Push
github central Repo

Commit

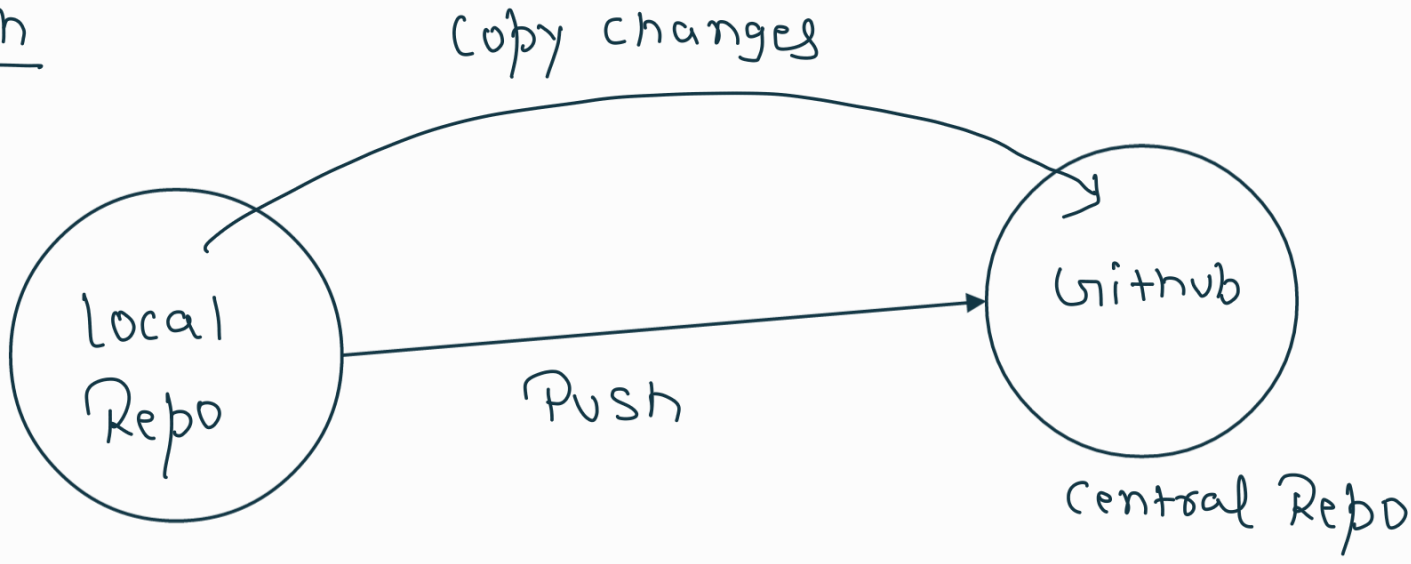
→ Store changes in local Repository. You will get one commit-ID:
40 alphanumeric characte

→ Even if you change one dot,

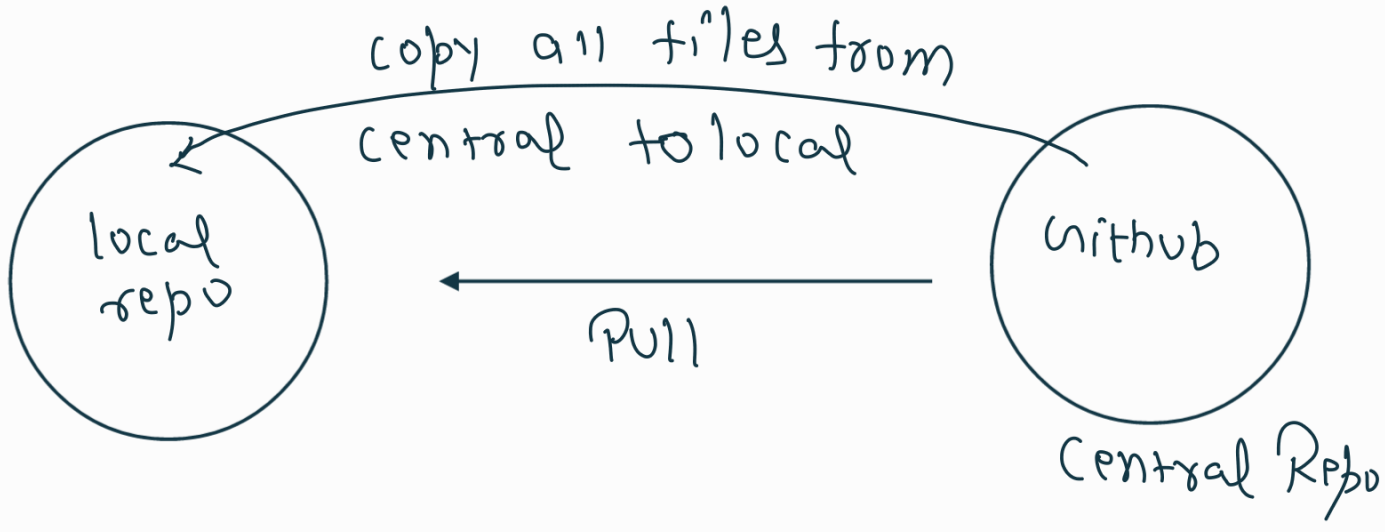
commit-ID will get change.

→ git helps you to track changes.

Push

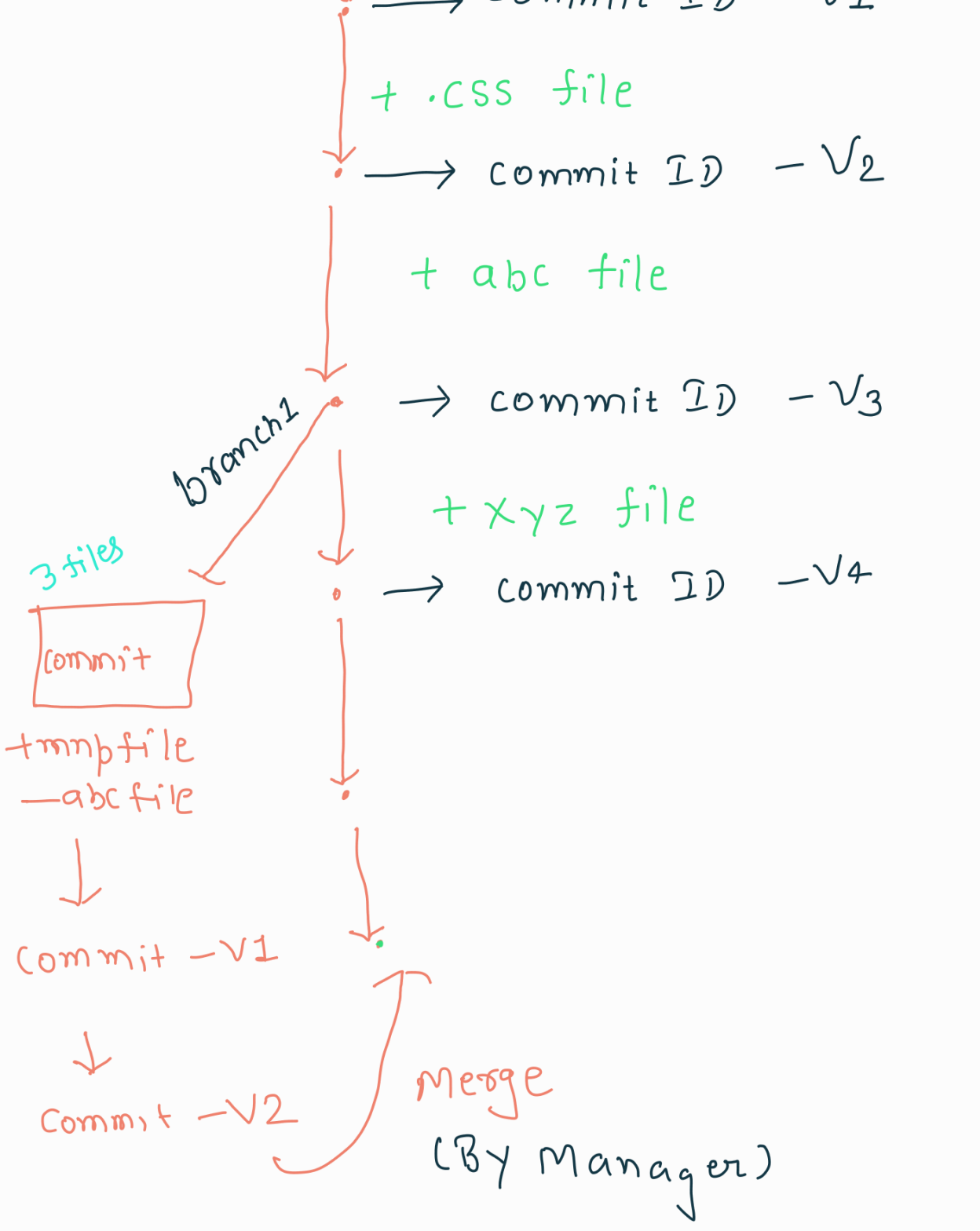


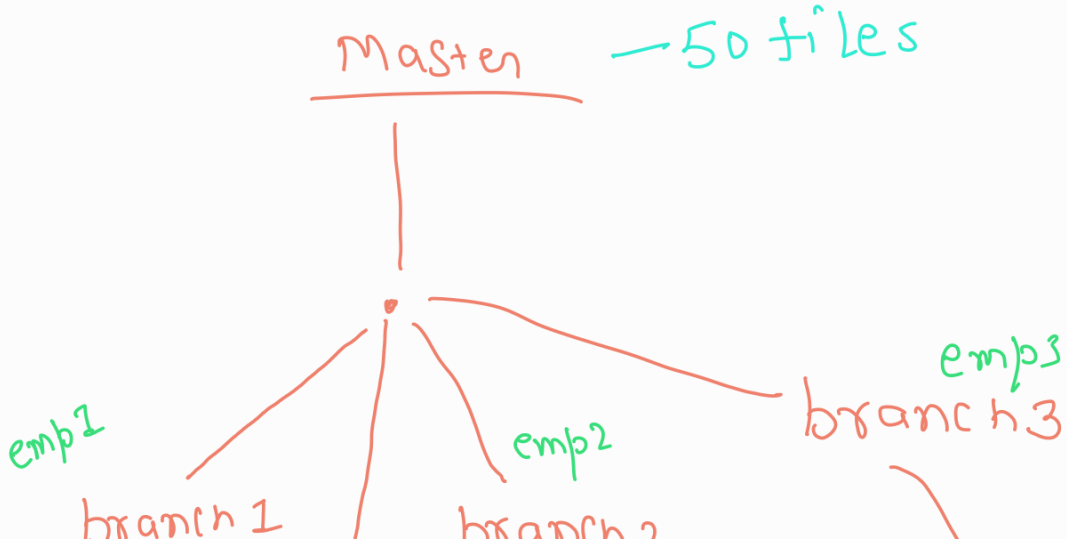
Pull



Branch

Master Branch (Default)
+index.html
→ commit ID = V1

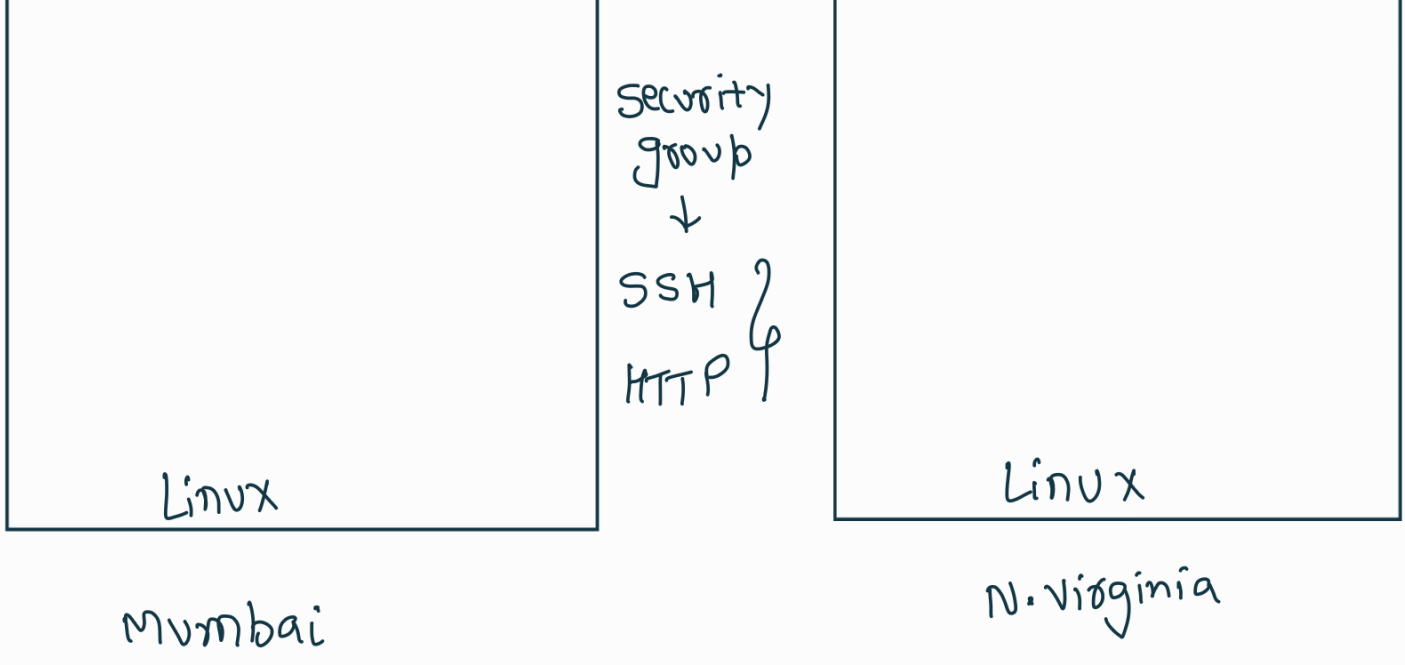

+ .css file
commit ID - V2
+ abc file
commit ID - V3
branch1
+ xyz file
3 files
commit ID - V4
commit
+tmpfile
-abc file
commit -V1
Commit -V2
Merge
(By Manager)


Master
-50 files
emp1
branch1
emp2
emp3
branch3

→ Benficial when you want to work parallely.

→ Can create one branch on the basis of another Branch.

How to use git and create github account



Putty - access or by SSh

To install and configure

→ sudo su root

→ yum update -y

→ yum install git -y

→ git --version

→ git config --global username "kishan"

your name

mumbaiemb

→ git config --global useremail "-@gmail.com"

→ git config --list

central Repository

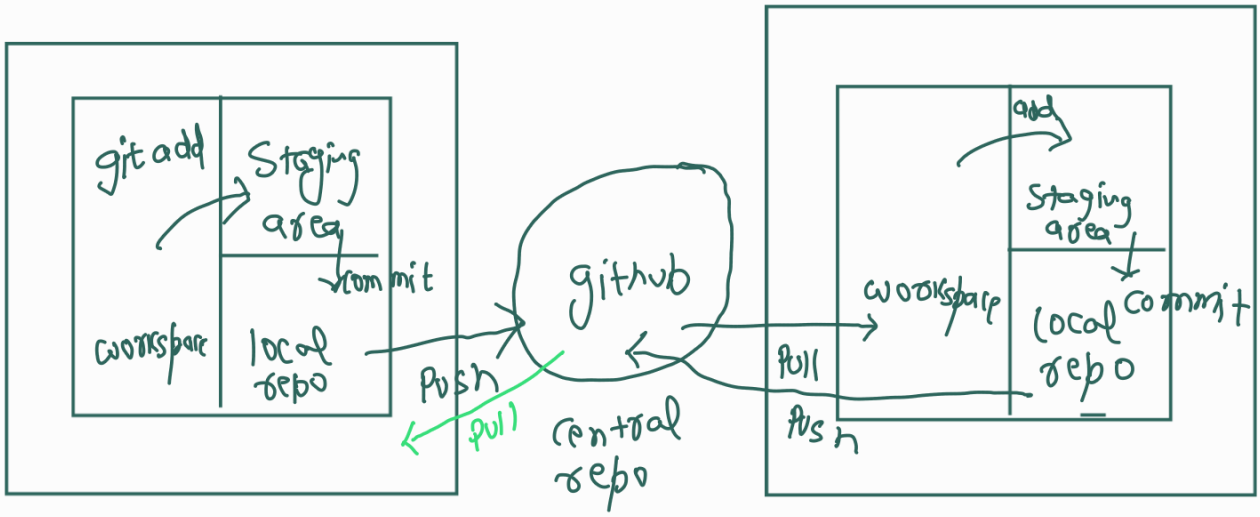
Mumbai

N. virginia.

{ will see this in lab demo }

Also create a github account

How to add, commit, push & pull from github



Mumbai

N. virginia

Steps :-

→ login into Mumbai EC2 instance
→ Switch to root user (sudo su root)

→ Create one directory and go inside it.
→ git init (will convert your normal directory to git repository (local repo).)

→ touch myfile (put some data)

→ git status (it will tell the status of repo, which files are present etc.)

→ git add . ← dot (will add everything from workspace to staging area)

→ git commit -m "1st commit from mumbai"

→ git status (working area is clean now)

→ git log → will show the global username,

email, time at which commit happened, commit id.

→ git show <commit-id>

↳ gt will info about the code which is committed.

→ git remote add origin <centralgiturl>

↓

will get from github website.

→ git push -u origin master

(enter username and password)

(will do the lab now in mumbai EC2)

(Refer lab doc of github)

Steps:-

→ Now login to N.virginia EC2 instance.

→ Sudo su root (To switch to root user)

→ Create one directory and 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 file or add one more file.

→ git status

→ git add .

→ git commit -m "North virginia commit 1"

→ git status

→ git log

→ git push origin master

{ Will do the lab in N. Virginia EC2 instance }

After this go to mumbai ec2 instance and again pull the new code.

→ git pull origin master

.gitignore

→ To create some files while committing create one hidden file .gitignore and enter file format which you want to ignore.

for e.g → vi .gitignore

*.css
*.java

{ name should be exactly same.

{ if you do not want to commit .css, .java files.

{ file1.css, file2.css, abc.java, bcd.java } these files will be ignored.

→ git add .gitignore

→ git commit -m " lates update exclude

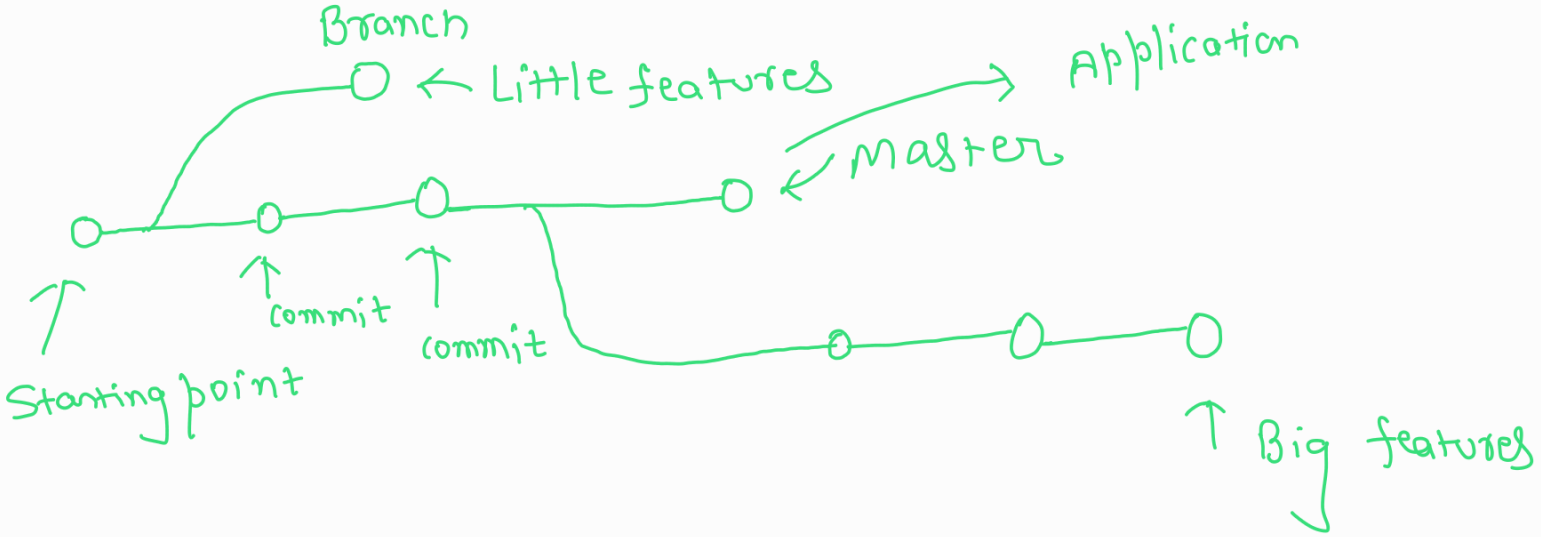
css, java"

→ git status
→ create some text, java & css file and add them by running "git add ."

foreg → touch file1.txt file2.css file3.java

→ ls
→ git status
→ git add .
→ git status
→ git commit -m "new commit excluded css and java file"

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 in Branches, it's not only possible to work on Both of them in parallel but it also keeps the main Master Branch free from error.

Notes :→

→ Each task has one separate Branch.

→ After done with code, Merge other Branches with master.

→ Useful for parallel development.

→ can create any no. of branches.

→ changes are personal to that particular Branch.

→ Defaut 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 belong to that particular branch.

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

→ First start the linux ec2 instance
  → sudo su root (Mumbai)
→ cd <gitdir> (Local Repo)
→ git log --oneline
  → will show log in oneline.

→ git branch
  * Master { will show all the Branch
  * → current working branch }

→ git branch branch1
  ↳ will create branch1

→ git branch
  * Master { cricket two batsman
  striker score-58 *

Branch 1

→ git checkout Branch 1

↳ Now you will land to Branch 1.

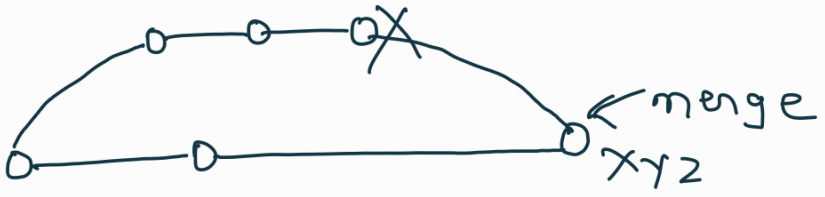
→ git branch

master

* Branch 1

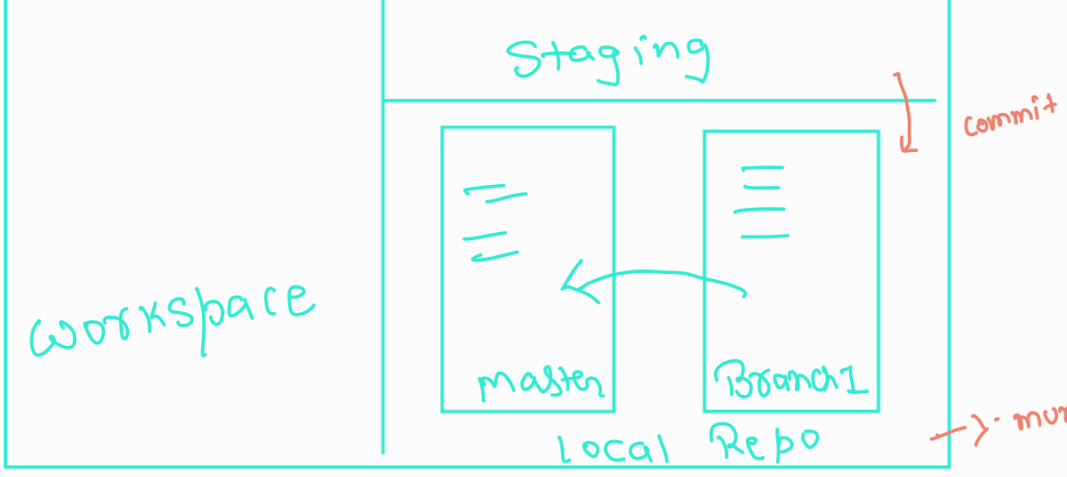
→ git branch -d branch 1

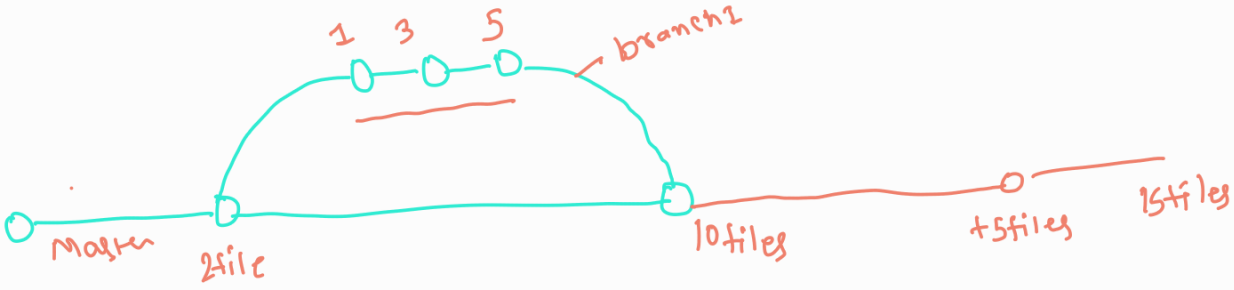
↳ To delete Branch



* If not merged then use -D for forcefully delete.

Merge





→ you can't Merge branches of different Repositories. (Mumbai to N.virginia not possible)

→ we use pulling mechanisms to Merge Branches. (copy paste) not (cut paste)

→ git merge <branch-name>

git conflict

↳ when same file having different content in diffirent branches, if you

do merge, conflict occurs. (Resolve conflict then add and commit).

→ conflict occurs when you merge Branches.

Master — File 1

Branch1 — File 1

-My name is Kishan — common — My name is Kishan.

add only this line

merge

I am Roy

↳ No conflict

-My name is Kishan I am devops

merge conflict → manually

→ If data is completely different then conflict occurs.

→ Git is confused which line to keep first and to which keep second.

first and to work on top second.

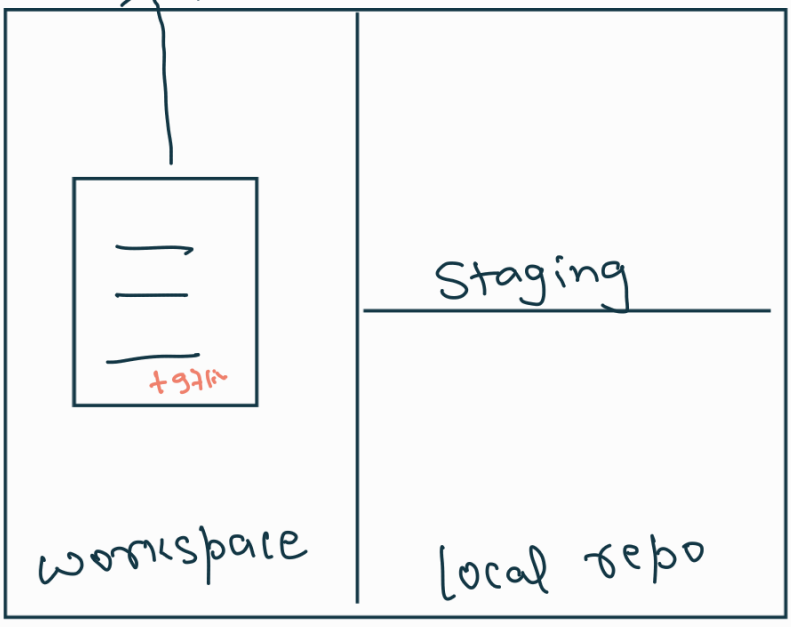
Git Stashing

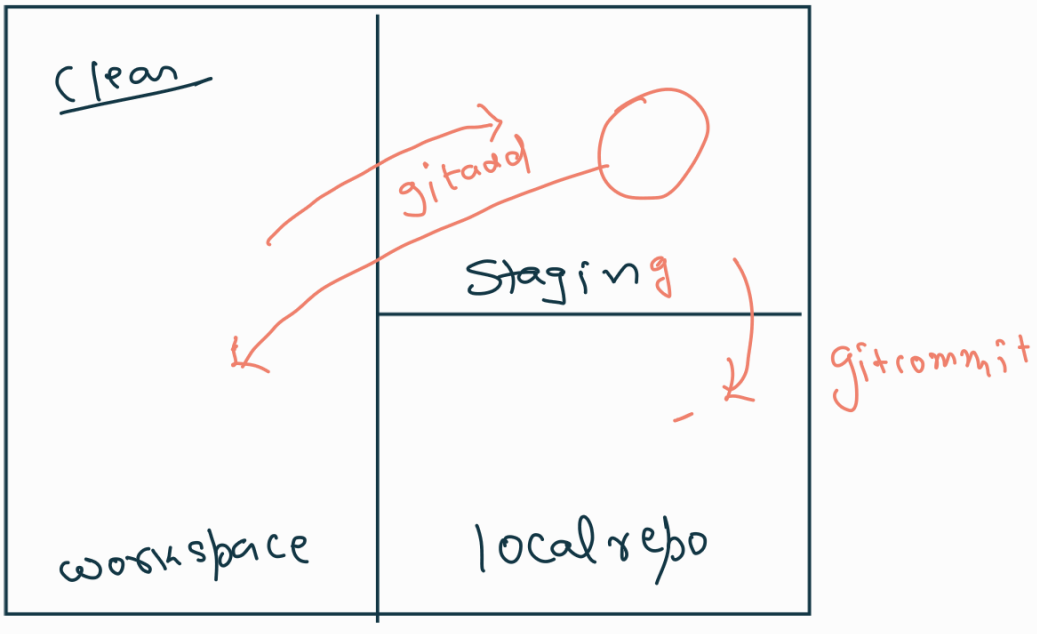
Stash { 100 lines of code → commit ?

→ 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 feature work for few hours.

→ you cannot commit your partial code and also cannot throw away your changes. So you need some temporary storage, where you can store your partial changes and late commit it.

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





→ git stash → To stash an item

→ git stash list → To see stashed item list.

→ git stash apply stash@{0}

↳ To apply stashed items

Then you can add and commit.

→ git stash clear
↳ To clear stash files.

git Reset

* If you added a file in staging area, after that if you want to remove that file from staging area then we use git reset command.

To reset Staging area

```
git reset <filename>
git reset .
```

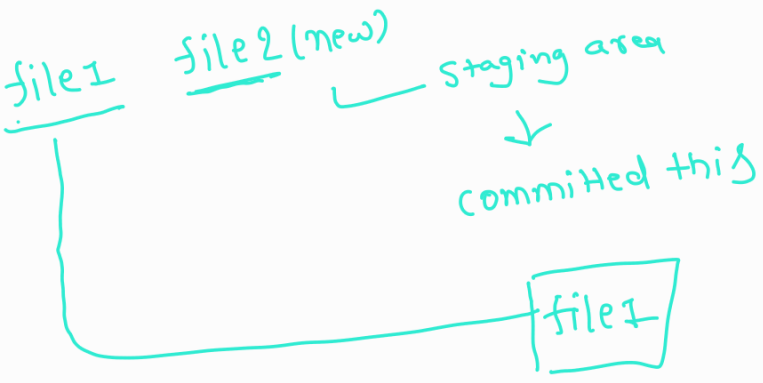
To reset the changes from both

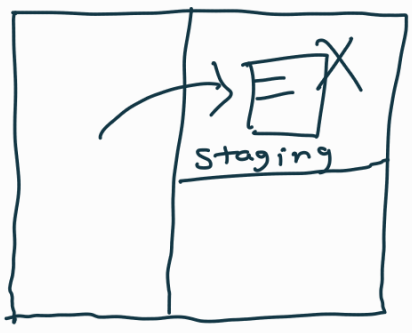
staging area and working directory at a time.

→ git reset --hard.

Git revert

↳ helps you undo an existing commit.





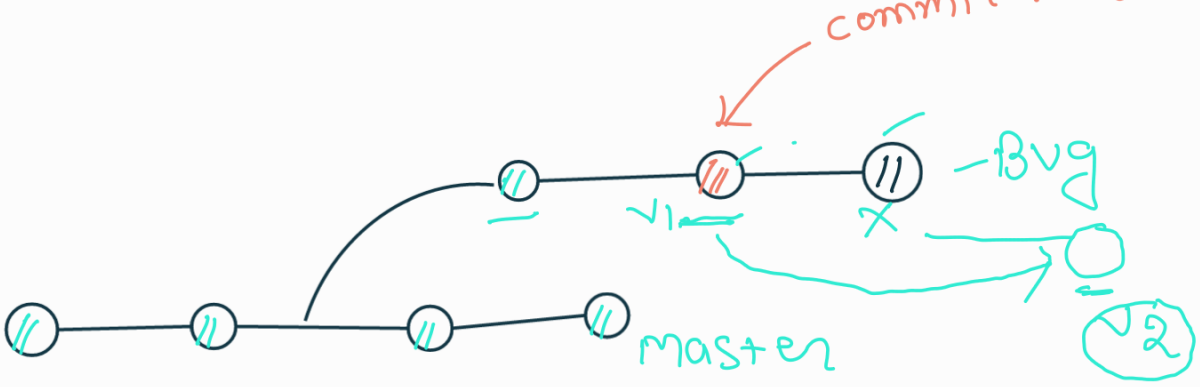
-git reset

git reset – before commit

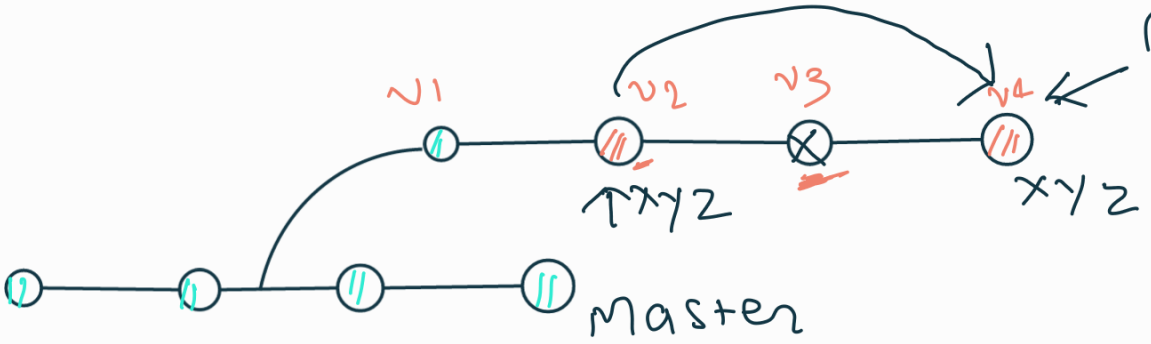
git revert – After commit (bullet fired from gun) cannot be deleted.

↳ git revert 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 file

moves backwards.

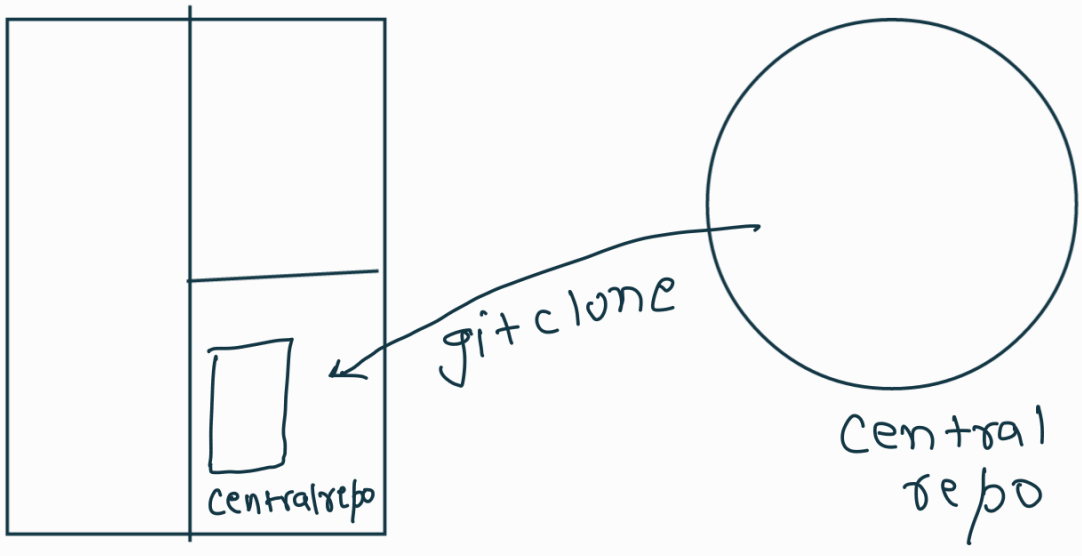


Before Revert



After Reverting

Github clone



→ open github website
→ login and choose existing repo
→ Now in linux machine
git clone <url of github repo>

→ It creates a local repo automatically in linux machine with the same name as in github account.

Git vs Github

CLI

GUI