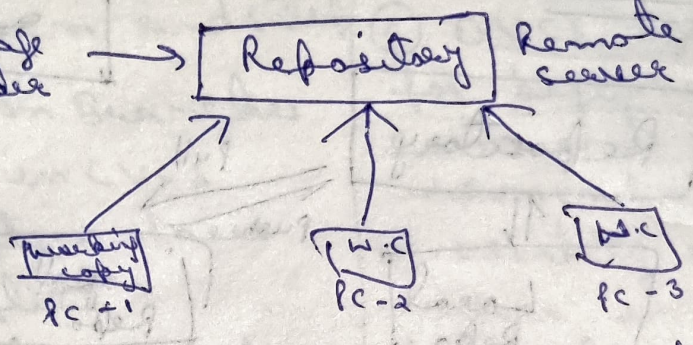
GIT is a distributive version control system

Software Configuration management
or
Source code management

→ In IT sector

Centralised version control system (CVCS)



- In this system anyone see the work of any person / all the workers.
- In this all the persons see stores our data code in Repository. (Repository is storage)

DRAWBACK

1. It is not locally available meaning you always need to be connected to a network to perform any action.
2. since everything is centralised if central server gets failed, you all loose entire data eg → SVN Tool

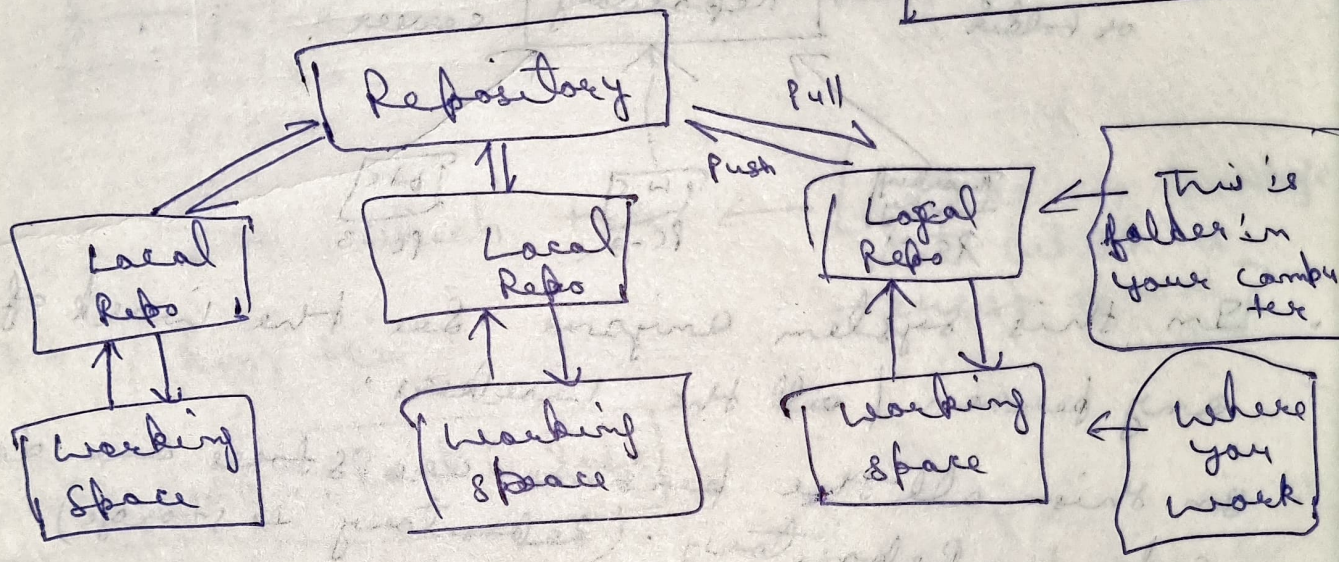
☆ GIT is developed by Linus Torvald in 2005

☆ Distributive version control system

☆ Git is a software which can be easily installed

☆ DVCS

~~Github~~
Github is used as storage element



☆ In distributive version control system every contributor has a local copy or "clone" of the main Repository i.e everyone maintain a Repository of their own which contains all the files & metadata present in main Repository.

- not use of Internet at Git level
- Internet is used only to store data in Repository (Server Repository)
- Backup of data is available in local Repo.
- It is fast; Reliable
- used to store diff version (changes in code) so we get easily any version anytime.

CVCS

1. In CVCS, a client need to get local copy of source from server, do the changes and commit those changes to central source on server
2. CVCS system are easy to learn and setup
3. working on Branches is difficult in CVCS developer often faces merge conflict.
4. CVCS system do not provide offline access.
5. CVCS is slower as every command need to communicate with server
6. If CVCS server is down developer cannot work

DVCS

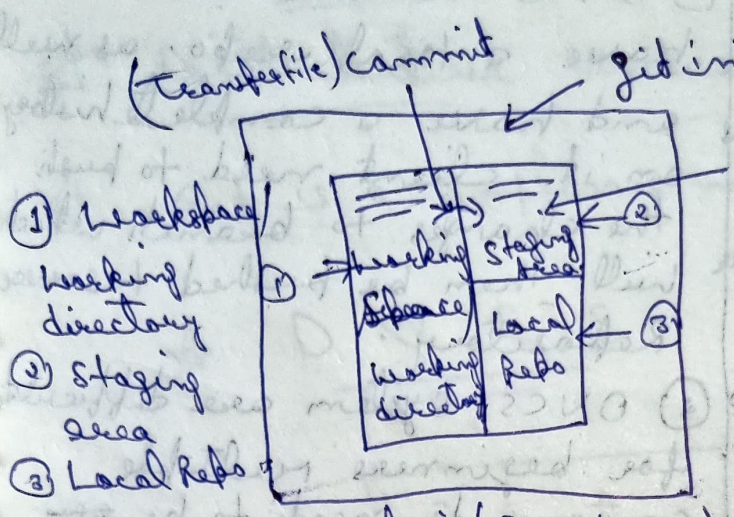
1. In DVCS each client can have a local repo. as well and have a complete history on it client need to push the changes to branch which will then be pushed to server Repository.
2. DVCS system are difficult for beginners multiple commands needs to be remembered.
3. working on branches is easier in DVCS developer faces less conflict.
4. DVCS system are working fine on offline mode as a client copies the entire repository on their local machine.
5. DVCS is faster as mostly user deals with local copy without hitting server everytime.
6. If DVCS server is down developer can work using their local copies.

Git → software, Github → service.

DVCS → Git, Mercurial

Bitkeeper is used by Linus Torvald but this is software is developed by other company & the tool is going on.

Stages of git | work flow



cloud computing

cloud computing :- NIST (national Institute of standard and technology)

- It is a model for enabling ubiquitous convenient on demand network access to a shared pool of configuration computing resources (for ex → network, servers, storage, application & service) that can be rabidly provisioned and released with minimal management effort or service provider interaction.
- Everything are access easily but Internet is connected. without Internet can't access.

FOR Git Install

① Two Instances are created in which SSH and HTTP are free available.

② ~~Install~~ → yum update -y [for update]

③ yum install git -y [EC2 user, Sudo su]

④ which git for checking where it is present.

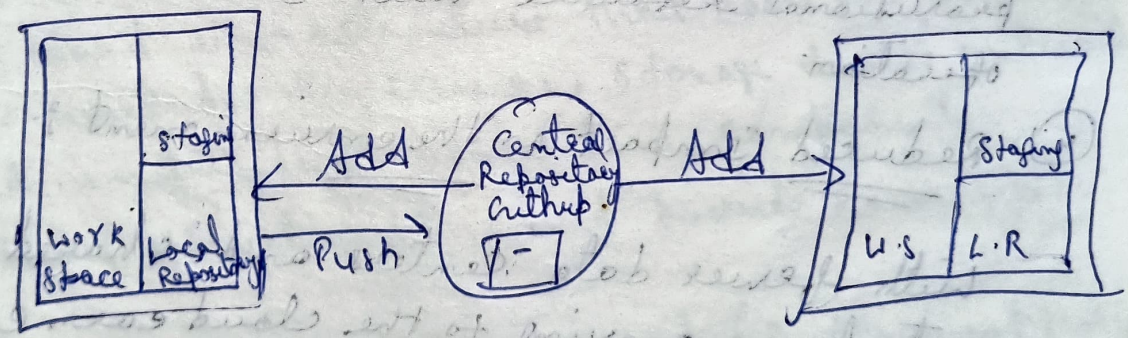
⑤ git --version [For checking version] 2.37.1

⑥ git config --global user.name "rohit"

⑦ git config --global user.email "rohit@"

⑧ git config --list [To check name & email]

Stages of git / Work flow



↑ steps for connect with github.

→ Login in mumbai EC2 instance
→ create one directory and go inside it
→ git init (for initialize)
→ touch myfile (put some data)
→

→ git status [check which are present inside working directory]

→ git add . [to add with github] [working area to staging area]

→ git commit -m "1st commit from Mumbai"
↑
for commit

→ git status [No work inside working area bcz you commit work into local Repo.]

→ git log [tell who commit this work name, email]

→ git show <commit-id> [to see the work of that Id]

→ git remote add origin <central git url> [to add with github]

→ git Push -u origin master [Post the work in github]

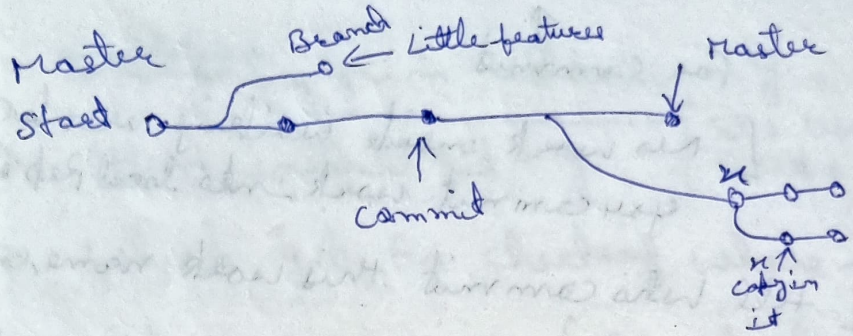
→ Enter, username & password

☆ For Pull the Data ☆

git init
git remote add origin <github repo url>
[git clone <URL Github>]
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 updated"
git status

git log
git push origin master

How to create Branch



The diagram above visualizes a Repository with two isolated lines of development, one for a little features by developing them in Branches, its not only possible to work on both of them in parallel, but it also keeps the main master branch free from errors.

→ Each task has one separate Branch.
→ After done with code, merge other branches with master.
→ This 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.

Sudo su
< mumbaigit >
git log --oneline ← To see all the (content) commit.
git Branch
* master
# git Branch branch1
# git Branch
* master
Branch1
git checkout branch1
git Branch
master
*Branch1

git branch -d branch1
↑
to delete a branch
git branch -D Branch1
↑
forcefully delete

In this the content of master is copy first time

To change Branch go to another branch

Branch1

git checkout master
ls
file x
git checkout Branch1
git checkout master
ls
— No file here bcz this is commited.
git checkout Branch1

git checkout
cat > file x
git add .
git commit
ls
file x
git add .
git commit -m "x"
ls
file x
git log --oneline
x ← latest commit

So In Branch first time the master Branch content is copy in that Branch which are make them by you. then after commit it can't show Before commit it SHOW.

Repository :

- Repository is a place where you have all your codes or kind of folder on server
- It is a kind of folder related to one product
- Changes are personal to that particular Repository

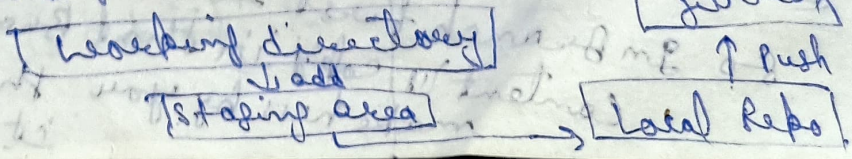
Server :

- It stores all Repositories
- It contain metadata also.

Working directory :

- Where you see files physically and do modification
- At a time, you can work on particular Branch

In other CVCS, developers generally makes modification and commit their changes directly to the Repository; But git uses a different strategy Git does not track each and every modified file whenever you do the files present in the staging area only those files present in the staging area are considered for commit & not all the modified files



Important command for central Repo to Git

1. Git init
2. Git status
3. Git add .
4. Git log (git log --online)
5. Git show
6. Git remote add origin (url of Github)
   (Git clone < URL Github)
7. Git commit -m "comment"
8. Git push -u origin master
   (Git init)
9. Git pull -u origin master

SSH-Key gen (setting github SSH & Copy)

SSH -T git@github

Git Fetch

Git Clone < Github < URL >