⇒ sudo su - ⇒ switch to root user.

root@svn ~:# su user1

⇒ netstat -lntp

⇒ svn log

⇒ svn update

⇒ diff

⇒ svn status

⇒ svn copy -r3 trunk/ tags/tabletag } to create tag

⇒ svn commit -m "table tag"

⇒ svn copy -r4 trunk/ branches/DemoBranch → create Branch

⇒ svn commit -m "new branch"

⇒ user1@svn : ~/testrepo

⇒ . → current dir
.. → parent "
/ → root "
~ → home "
- → prev working directory.

⇒ svn merge ../branches/loginpage/ } code merged

⇒ svn commit -m "code merged"

⇒ history | grep 'merge'

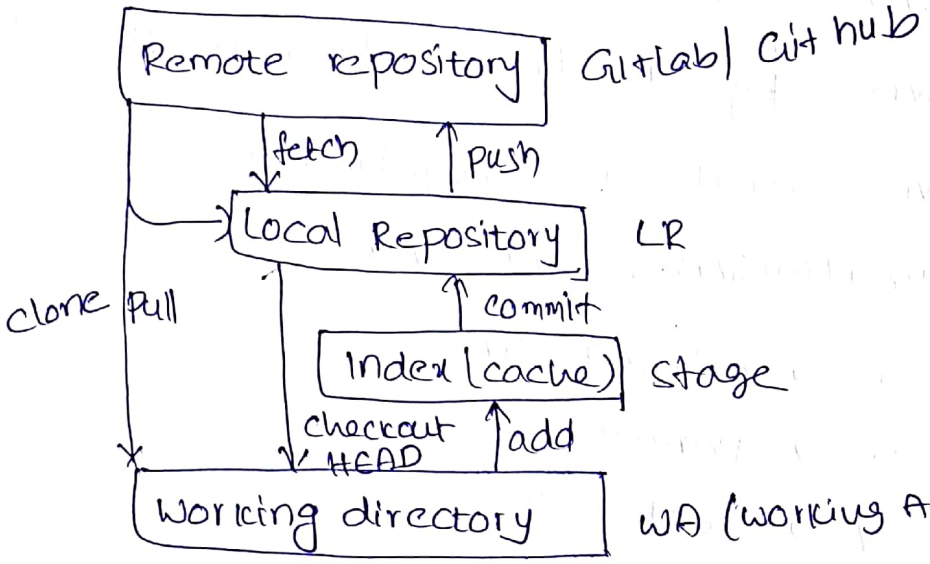
GIT:- GIT is an open source, distributed version control system (VCS) design to handle everything from small to very large projects with speed and efficiency.

→ GIT is a distributed

→ It is commanly used fd source code management (SCM). (or) VCS (version ctrl system) (or) RCS (Revision control system).

Three tree Architecture:-

Git Data flow



GIT workflow:-

GIT VS SVN (CVS)

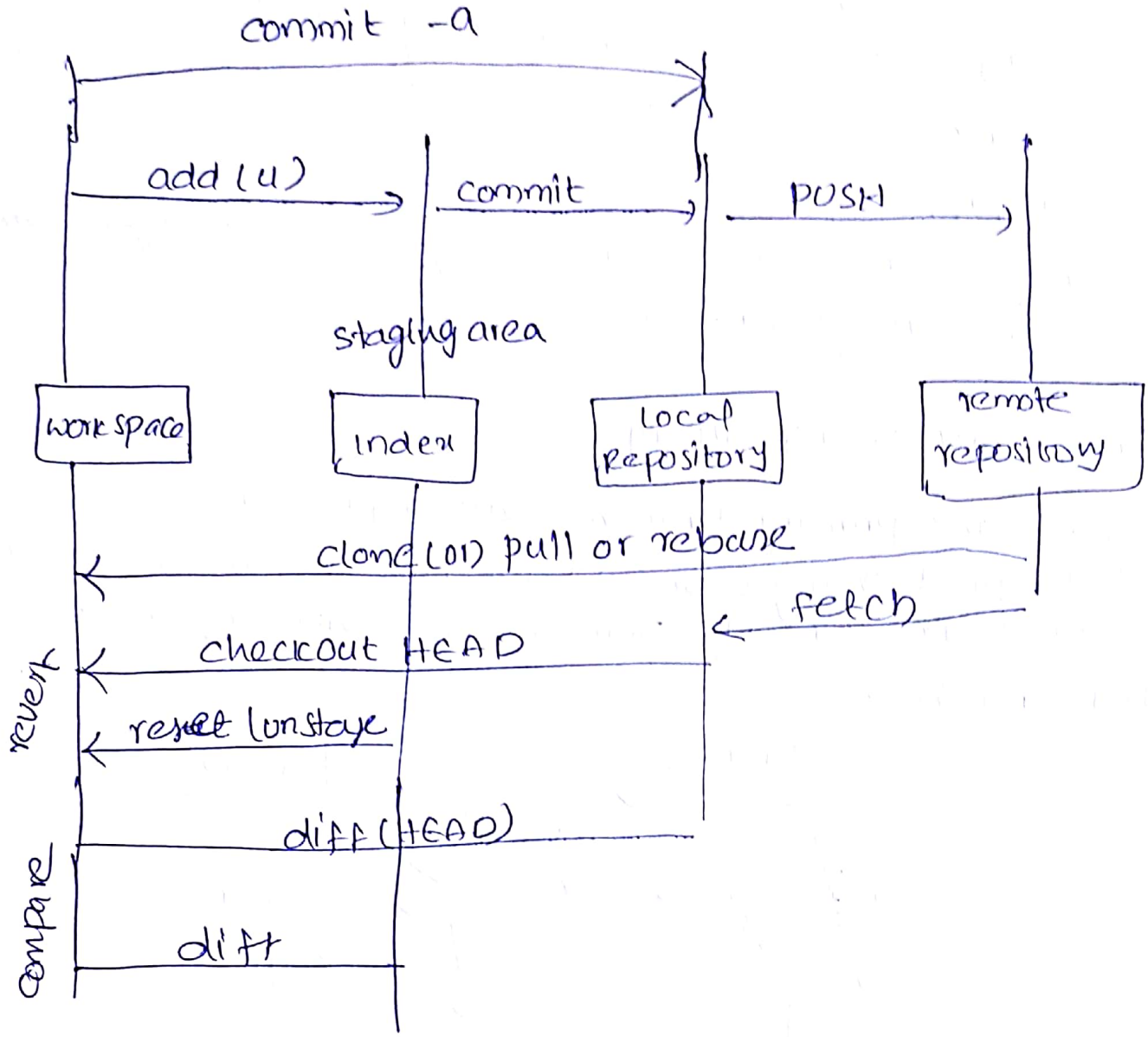
→ Fetch or clone (create a copy of the remote repository) (compare to cvs checkout)

→ Modify the files in local branch

→ stage the files (no cvs comparision)

→ commit the files locally (no cvs comparision)

→ push changes to remote repository (compare to cvs Commit).



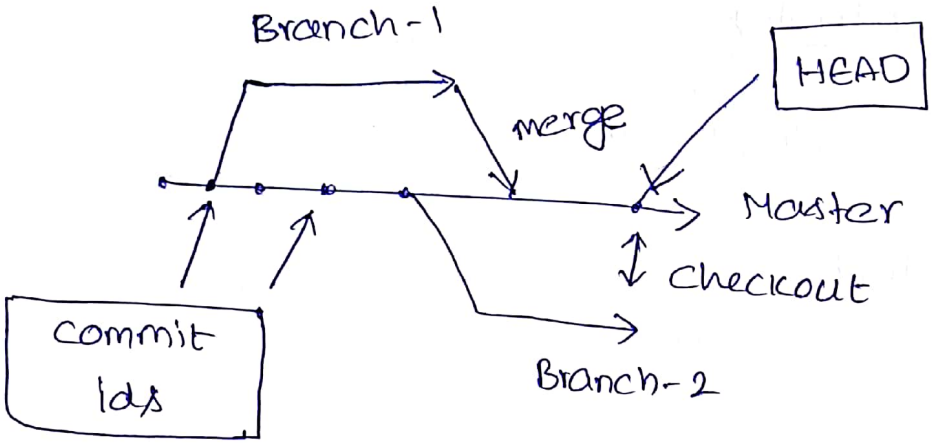
GIT Terminology :-

Repository ("Repo") :- GIT reprository as a directory that stores all the files, folders and content needed for your project.

Branch :- A version of the repository that diverges from the main working project.

Clone :- copy of repository

Master :- The primary branch of all repositories, all committed and accepted changes should be on the master branch. you can work directly from master branch, or create other branches.



checkout :- The git checkout command is used to switch branches in a repository.

Merge :- Taking changes from one branch and adding them into another (traditionally master) branch.

root@git: apt-get update

→ apt-get install git-core -y

→ ~~git~~ git --version

In Redhat:-

yum update
yum install git-core
git --version

} Redhat

→ copy clone url

→ git clone url

→ git init (empty repository .git will create)

→ cd devOAM (repo)

→ root@git: ~/devOAM# → config b/w local & global repo

git config --global user.name satyadevops

" " " user.email " "

~~# git~~ g. ls -a → hidden files will shown.

→ # cat .gitconfig → configuration details will reside here.

→ git status

→ git add readme.md (stage area)

→ git commit -m "meta data" (local repository)

→ git push origin master (to move code to global repo)

→ username:

→ password:

P.T.O

→ avoid passwords use key

→ ls -a

.ssh → contains · Public key (.pub)
· Private key (.pem)

password less authentication.

→ cd .ssh

→ we have create keys.

→ .ssh # ssh-keygen

→ ls

→ cat id-rsa.pub (copy the key)

→ In github settings → deploy keys → add key → allow write access → submit.

→ now add to repository

→ root@git:~/devopsAM #

git remote set-url origin git@github.com:sathyade-vops/devopsam.git.

sathyade-vops → account

devopsam.git → project.

→ vi demo.txt (add some code)

→ git add demo.txt

→ git commit -m "new demo file"

→ git push origin master

→ /devopsam # mkdir myD

→ git mv demo.txt myD/ → move demo.txt to myD directory

→ git commit -am "file moved"

↳ add and commit.

→ remove file

- git rm demo.java

→ ~~Reso~~ git status.

→ git reset HEAD demo.java (recover the deleted file)
→ git checkout demo.java ⇒ revert the code changes.
→ git push origin master
→ git log ↴
commit Id contains 40 charecters.
→ git log --oneline (shows only commitId & msgs)
→ git log -p (code changes - i.e new code)
→ want logs for specific dates.
↳ git log --since=6-aug-2018 --until=7-aug-2018
→ git branch
→ git branch dev
→ git branch
→ git log --oneline
→ git checkout dev → switch to dev branch
→ git branch
→ vi index.html
→ git add index.html
→ git commit -m "new branch code"
→ git push origin dev
→ git checkout master
→ git merge dev ↳ Branch_name.

root@git:~/devops #

→ git branch qa ⇒ create qa branch

→ git checkout qa ⇒ move to qa

→ vi demo.java

→ git add demo.java

→ git commit -m ~~-java~~ "demo.java"

→ git push origin qa

→ git branch devqAM

→ vi sample.txt

→ git add sample.txt

→ git commit -m "sample code"

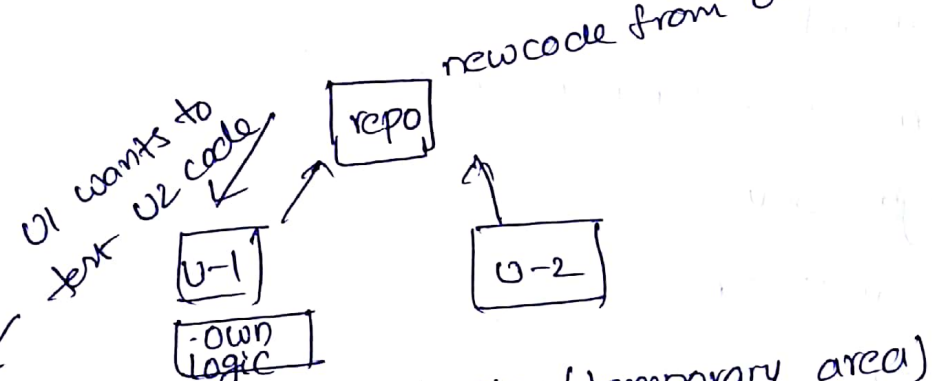
→ git checkout ~~branch~~ qa

→ ~~git branch~~

→ git merge master → conflict will raise. open then file save & commit.

→ git add demo.java

→ git commit -m "master code"



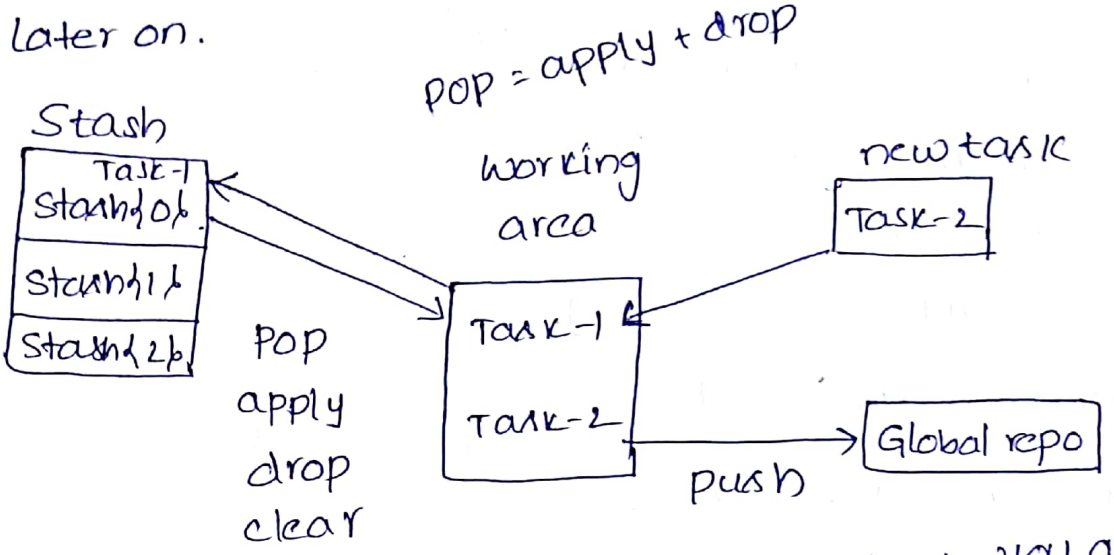
for this we have stash (temporary area)

⇓

Temporary memory area.

GIT stash :-

→ GIT stash temporarily shelves (or stashes) changes you have made to your working copy so you can work on something else, and then comeback and re-apply them later on.



→ stashing is a way to pause what you are currently working on and comeback to it later.

→ history | grep 'stash'

→ git stash save "Arth.oprs" } → to save stash

git ~~list~~ stash list

git stast apply stash@{0} } to apply stash

git stash save "mul_div funs"

git stash list

git stash pop stash@{0} } → to apply & drop stash

git stash list

git stash drop stash@{0} } → to drop a stash

git stash clear

→ delete branch

git branch -d qa
git branch -D dev

→ delete from local branch
-d "after merge"
-D "before merge"

→ delete remote branch

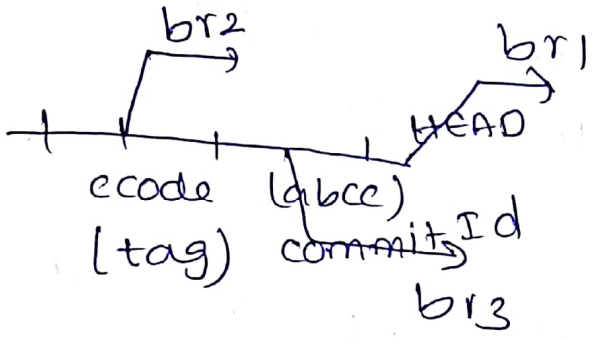
git push origin --delete dev
git push origin --delete qa

→ to delete a remote branch

creating tags

git branch br1
git branch br2 ccode → tag
git branch br3 abcc

HEAD IS MOST RECENT COMMIT ID



Tag is for label snapshot

git tag demotag
↳ created at HEAD

git tag ccode mnopqr → create tag for specific commit Id

ccode tag created for mnopqr commit Id

git push origin demotag → to push tag to central
git tag --list

git tag -d ccode → local

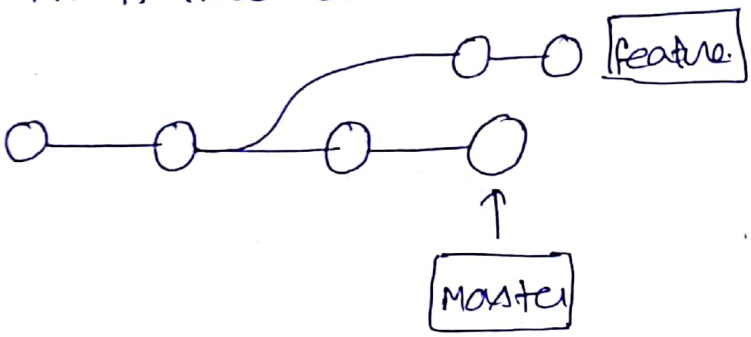
git push origin --delete ccode → central

→ git pull => pull the changes from global repository.

GIT Merge & Rebase :-

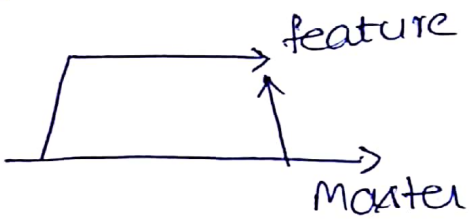
Merge :-

Merge takes all the changes in one branch and merges them into another branch in one commit.



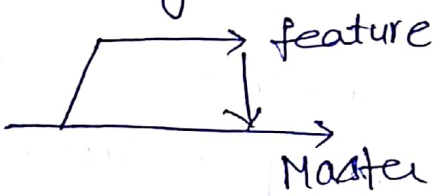
→ # git checkout feature

git merge master



→ # git checkout master

git merge feature

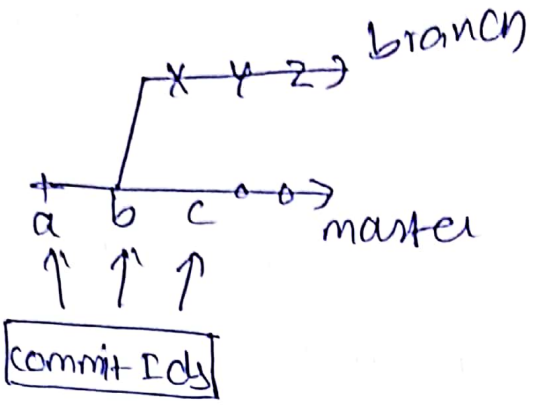


Git Rebase :- As it name suggests, rebase exists to change the "base" of the branch. which means its origin commit. It replays a series of commits on top of a new base.

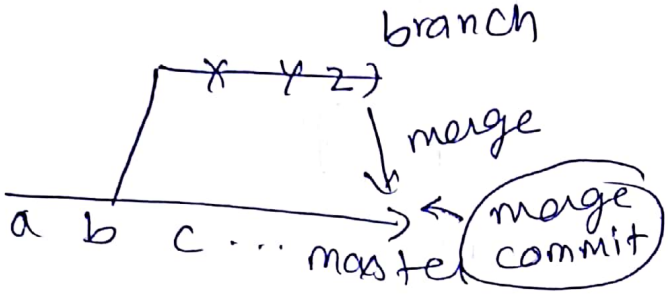
As an alternative to merging, you can rebase the feature branch onto master branch by following commands

git checkout master

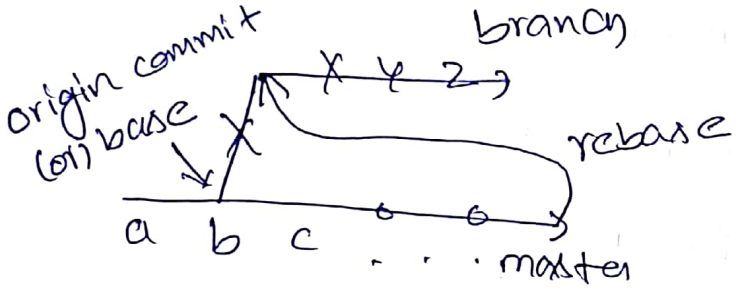
git rebase feature



merge :-



rebase :-



→ This moves the entire feature branch to begin on the top of master branch, effectively incorporating all of the new commits in master. But instead of using a merge ~~command~~ commit rebasing re-writes the project history by creating brand new commits for each commit in the

→ git branch br1
→ git checkout br1
→ vi demo.java
→ add some code here
→ git add demo.java
→ git commit -m "Deposit.class"
→ vi demo.java
→ add some lines
→ git add demo.java
→ git commit -m "again added"
→ git branch
→ git log --oneline
→ recent two adds show in log (branch)
→ git push origin br1
→ git checkout master
→ git log --oneline
→ vi demo.java (two lines not there)
→ git rebase br1
→ git log --oneline
→ vi demo.java (now lines available) ⇒
→ git push origin master (push to central repo).
→ vi sample.c
→ add some code
→ git add sample.c

→ git commit -m "newCode"

→ git log --oneline

→ now want to reset this commit

→ git reset --hard HEAD~ code lines

→ hard will reset commit history and ~~log history.~~

→ vi Sample.c (we reverted)

→ git log --oneline

→ git reset --soft # C17146e

↳ CommitId.

→ Then only commit history will delete & code remains same.

→ vi sample.c

→ add some new code

→ git add sample.c

→ git commit -m "sathya"

→ git log --oneline

→ git reset --soft 640a25

→ git reset --soft HEAD~

GIT commit Revert:-

git reset --hard HEAD^1

git reset --hard HEAD~

HARD RESET:- If you don't want to keep your changes that you made.

git reset --hard HEAD^ → commit Id

Soft Reset. If you want to keep your changes

git reset --soft HEAD^

↳ commit Id

→ git branch sathya
→ git checkout sathya
→ vi demo.java
→ add some code
→ git add demo.java
→ git commit -m "Ansible code"
→ vi demo.java
→ some code add
→ git add demo.java
→ git commit -m "puppet code"
→ git log --oneline
→ git checkout master
→ git log --oneline
→ git branch
→ git branch -d sathya
→ it not fully merged.
→ git branch -D sathya
→ How to revert the branch?
→ git reflog
↳ all commit ids including deleted branch logs.

→ c4cbfe9
2868921
⋮

→ git checkout -b sathya c4bfe9
↳ commit id (recent)
→ git log --oneline
↳ branch name.

To restore deleted branch:-

Step1:- find the old branch commit Ids using

git reflog

Step2:- To restore the branch

git checkout -b <branch> <recent commit Id>

git log --oneline

→ git remote set-url origin git@github.com:sathya-devops/demo.git

↳ project name

→ # git push origin --delete br1

→ # git push origin --delete tag1

→ # git stash apply <stashid>

git branch -d br1

git tag -d tag

git reset --hard HEAD~

git reset --soft HEAD~

git merge br1

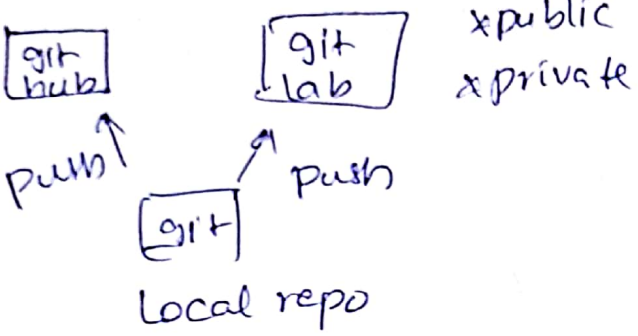
go to .get → It contains all details reg repository.

→ git hooks contains all repository states.

⇒ repository → settings → collaborators, we can add user as add collaborators. now this user also can add/edit my code

→ delete repository from settings.

www.github.com Global repo on-promise services



GIT LAB:-

→ Gitlab is written in Ruby, GIT Lab CE includes a host of features that enable s/w development teams to consolidates source code, track and manage releases.

→ GITLAB is an open source, cloud based Git repository and version control system used by multiple organizations world wide

→ It increases code quality, deploy code changes, and track the evolution of s/w over time.

→ need 3-4 GB of RAM.

→ apt-get update

→ sudo apt-get install -y curl openssh-server ca-certificates.

→ install postfix for email

→

To create new project:-

Git global setup:-

git config --global user.name "Administrator"

create new Repository!. ← IP address

```
# git clone http://<server IP>/root/my demoproject.git
# cd mydemoproject
# touch Readme.md
# git add Readme.md
# git commit -m "add Readme"
# git push -u origin master
```