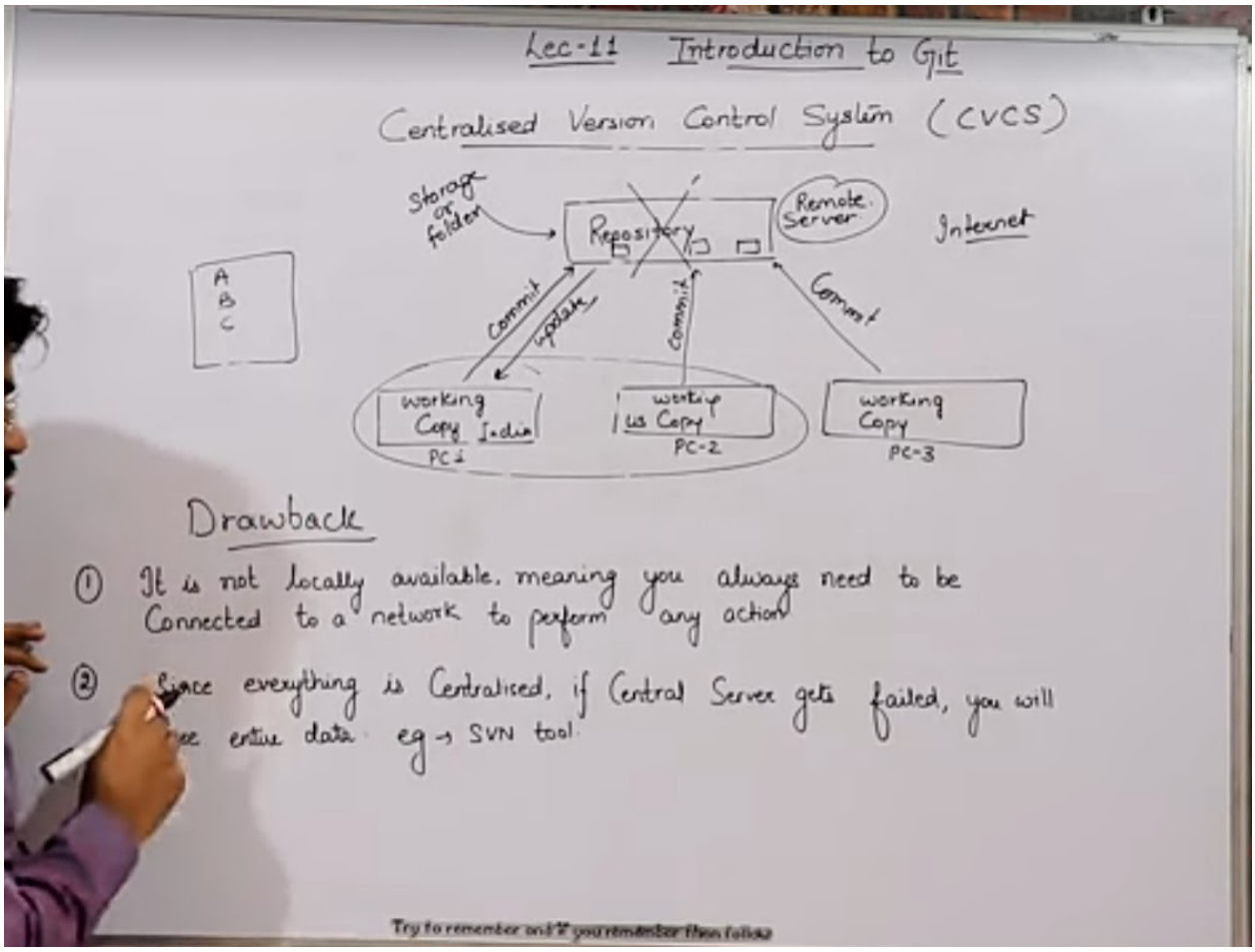
GIT

Software Configuration Management or Source Code Management or Version Control System(VCS)

1.Centralised VCS



Lec-11

CVCS

→ 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.

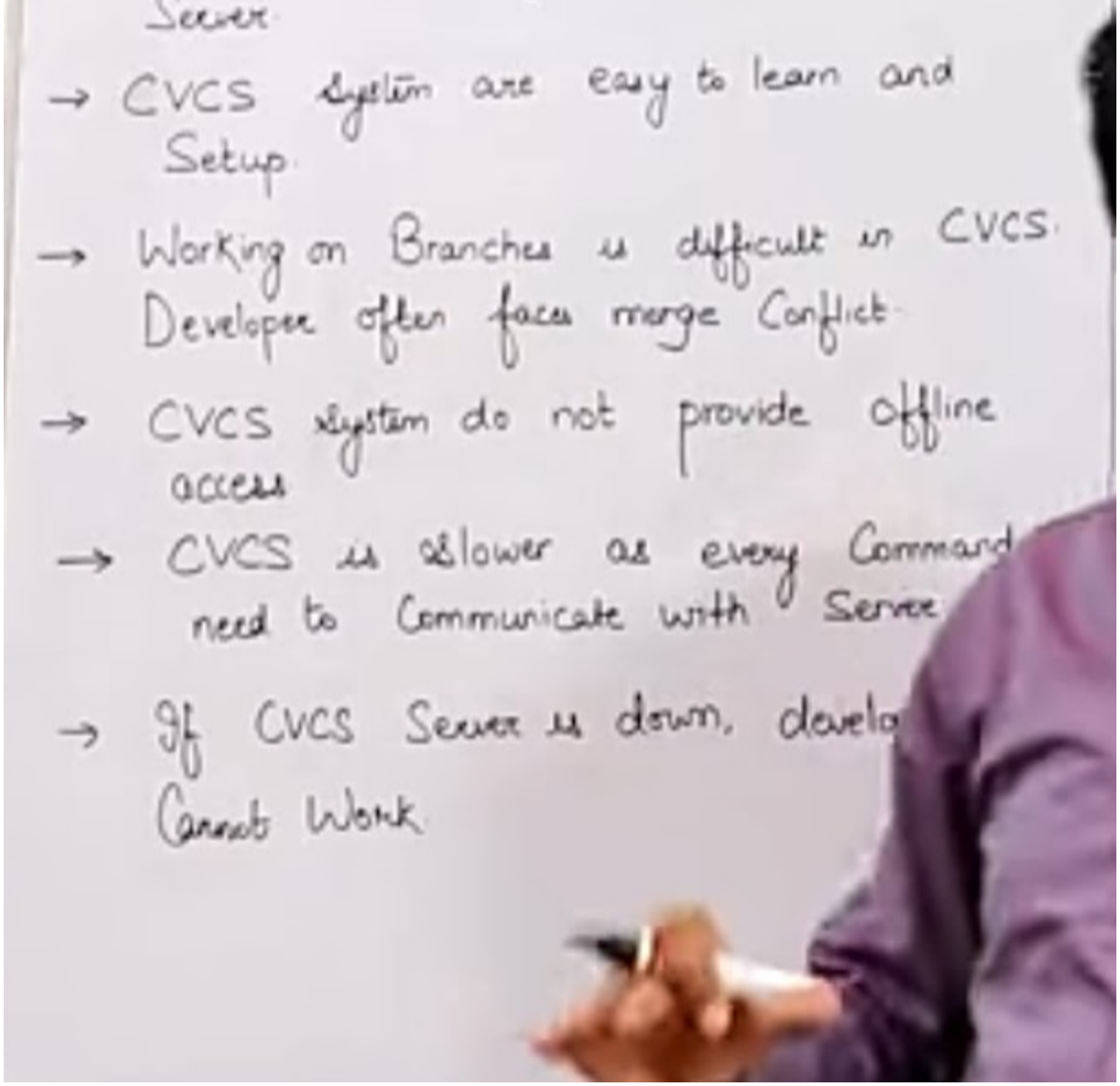
→ CVCS system are easy to learn and Setup.

→ Working on Branches is difficult in CVCS. Developer often faces merge Conflict.

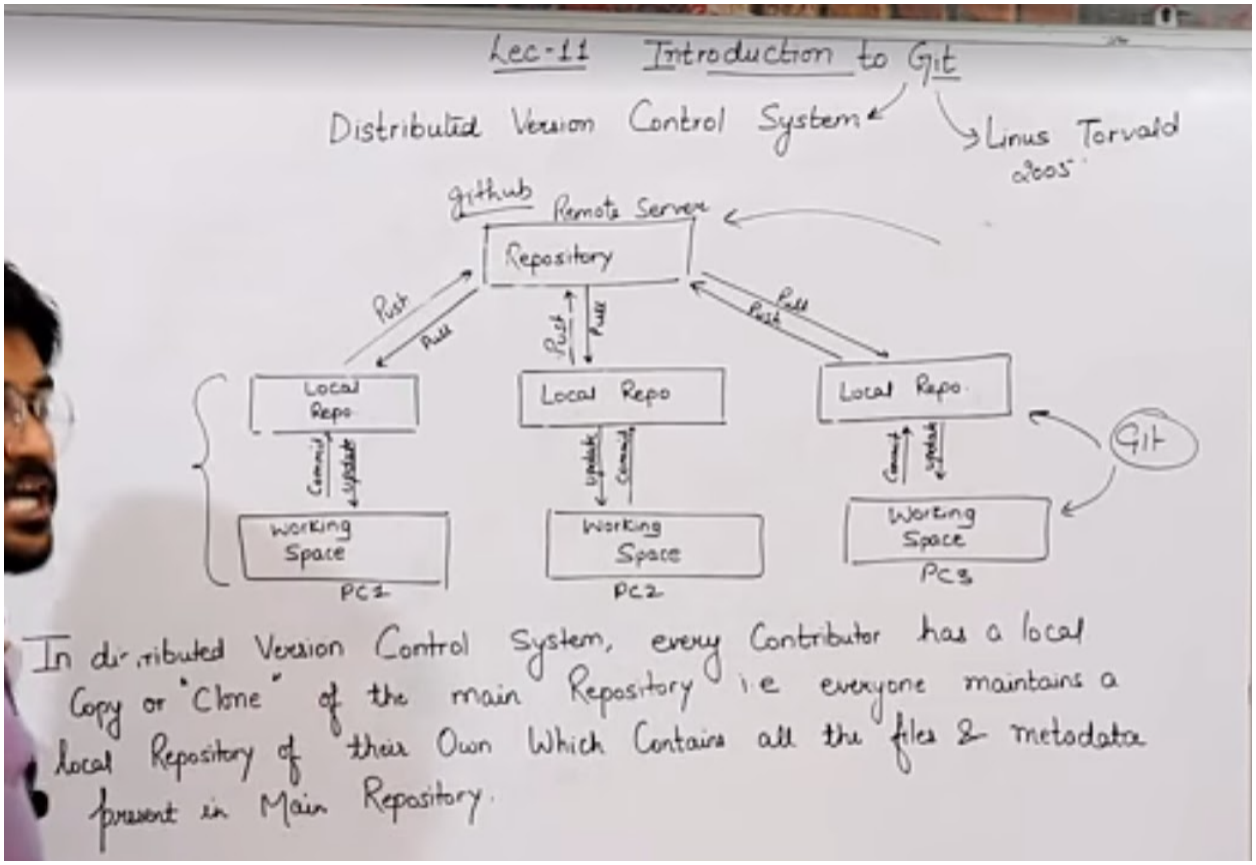
→ CVCS system do not provide offline access.

→ CVCS is slower as every Command need to Communicate with Server.

→ If CVCS Server is down, develo... Cannot Work.



2.Distributed VCS



Introduction to Git

DVCS

→ 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

→ DVCS systems are difficult for beginners Multiple Commands needs to be remembered

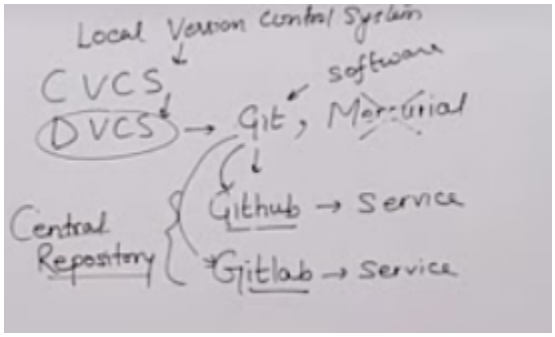
→ Working on branches is easier in DVCS Developers faces less Conflict

→ DVCS system are working fine on offline mode as a Client Copies the Entire repository on their local machine

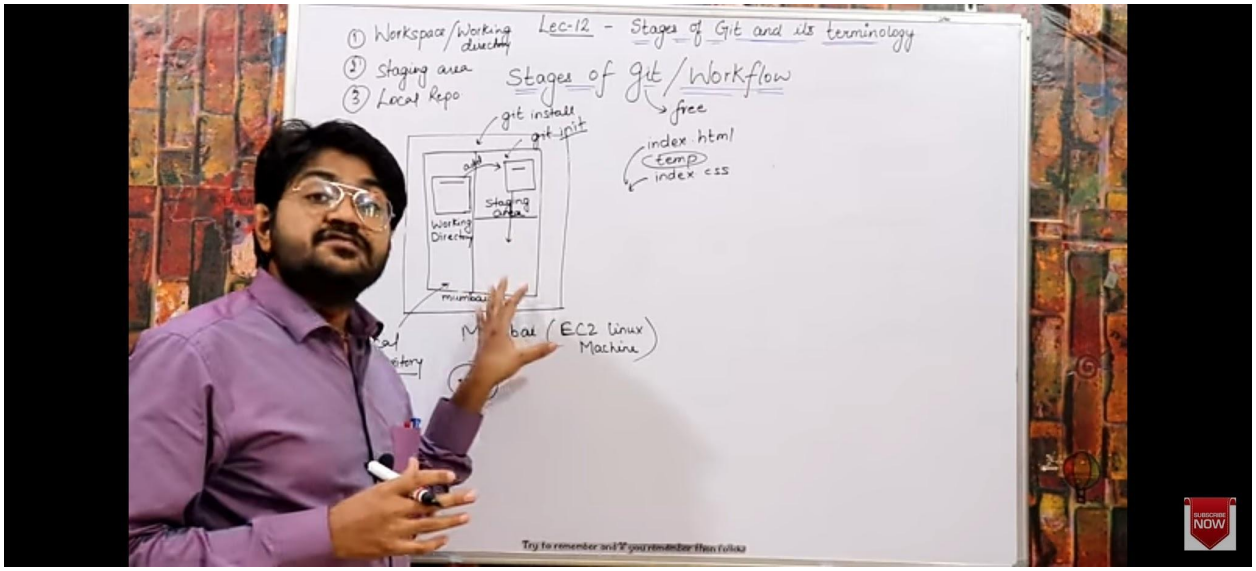
→ DVCS is faster as mostly user deals with local Copy without hitting Server everytime

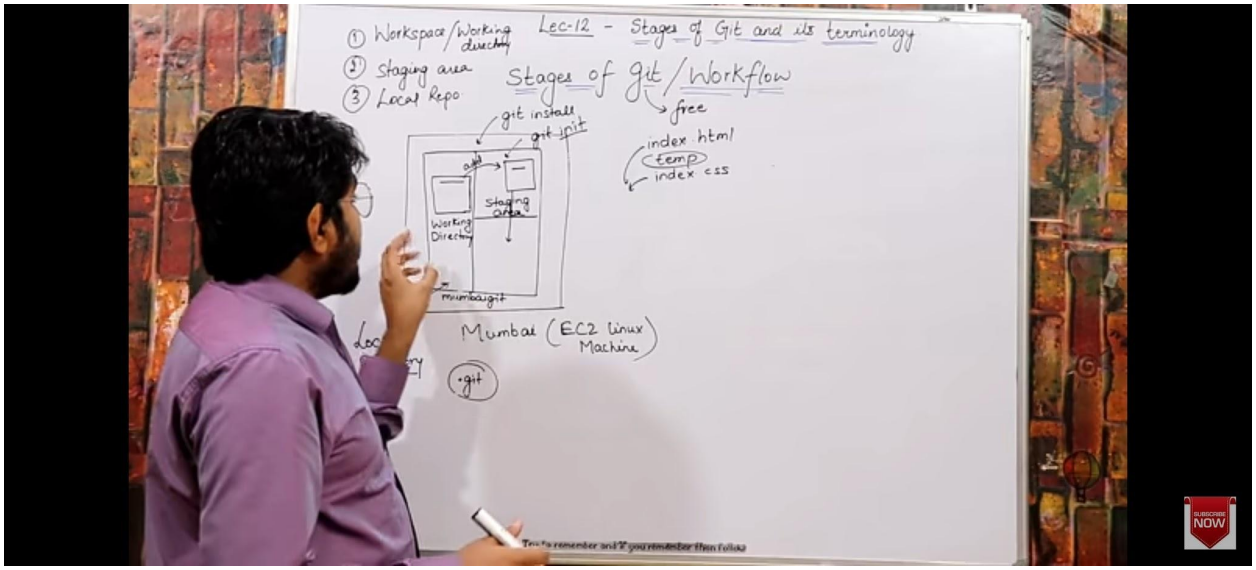
→ If DVCS Server is down, developer Can work using their local Copies

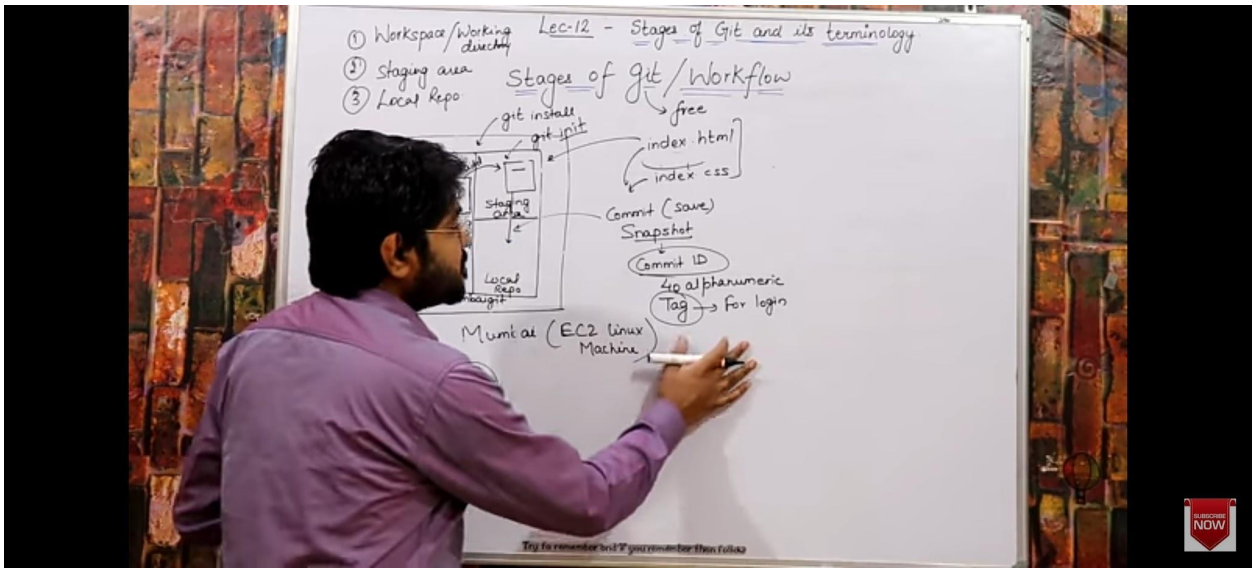
Stages of Git and its Terminology

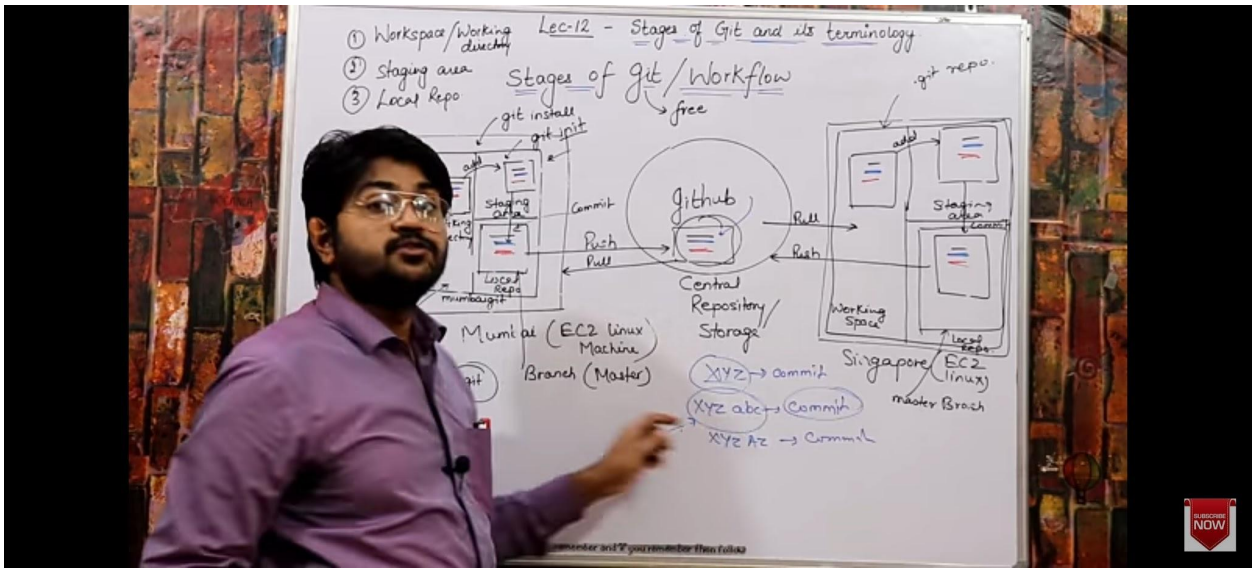


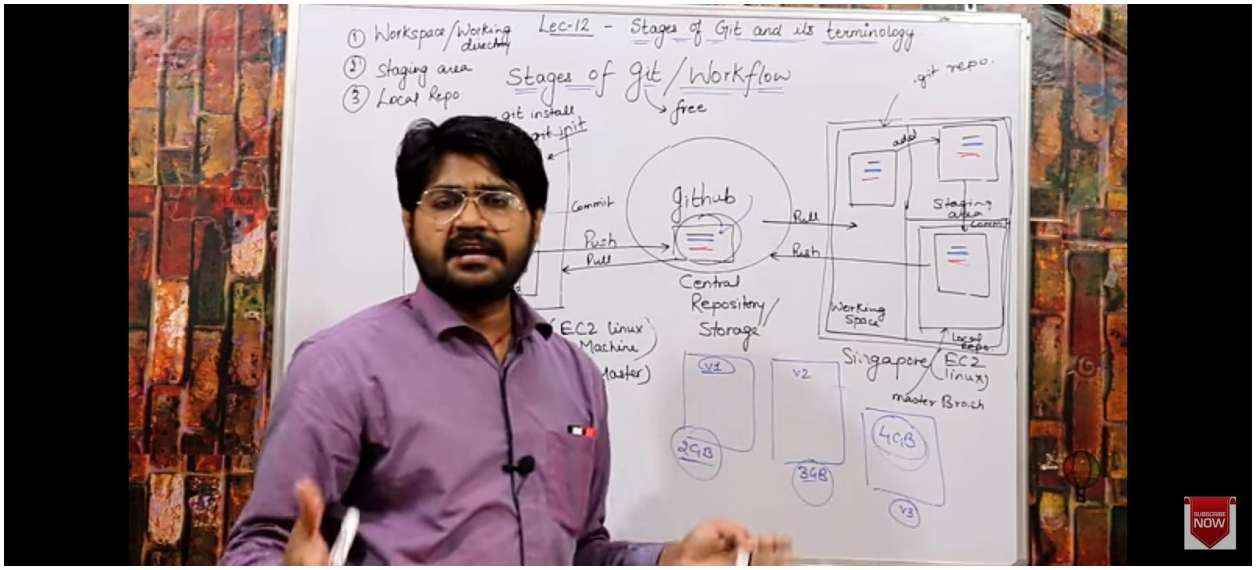
Stepwise explanation:

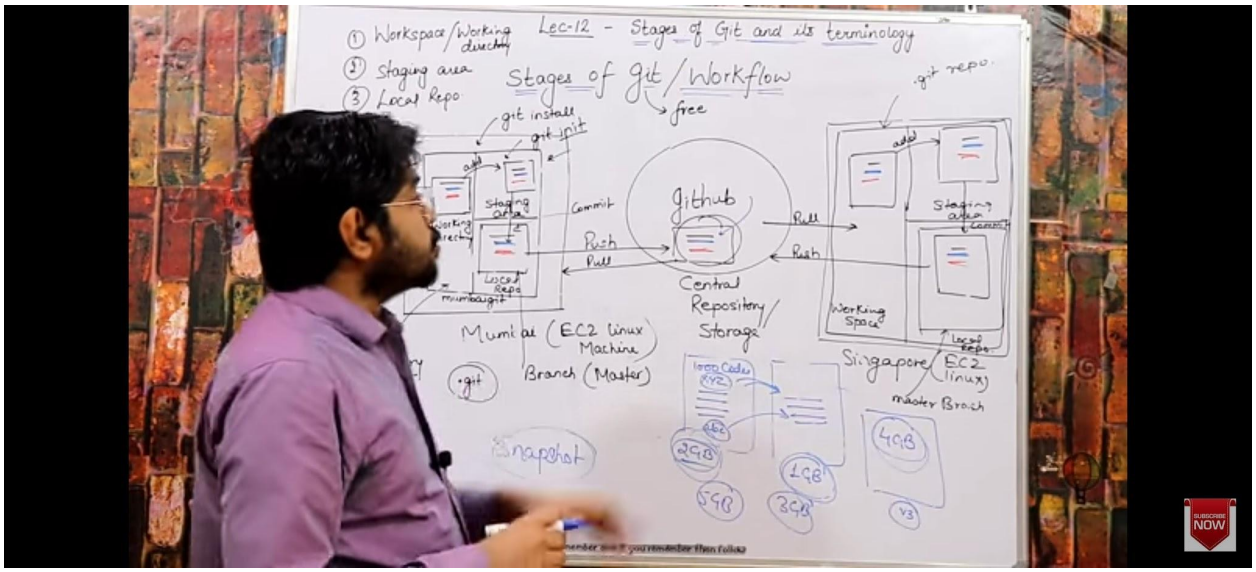



Lec-12 - Stages of Git and its terminology
1 Workspace/Working directory
2 Staging area
3 Local Repo
Stages of git/Workflow
free
git install
git init
index.html
temp
index.css
Working Directory
Staging area
Mumbai (EC2 linux Machine)
.git


Lec-12 - Stages of Git and its terminology
1 Workspace/Working directory
2 Staging area
3 Local Repo
Stages of git/Workflow
free
git install
git init
index.html
index.css
Staging area
Commit (save)
Snapshot
Commit ID
40 alphanumeric
Tag → For login
Local Repo
Mumbai (EC2 linux Machine)

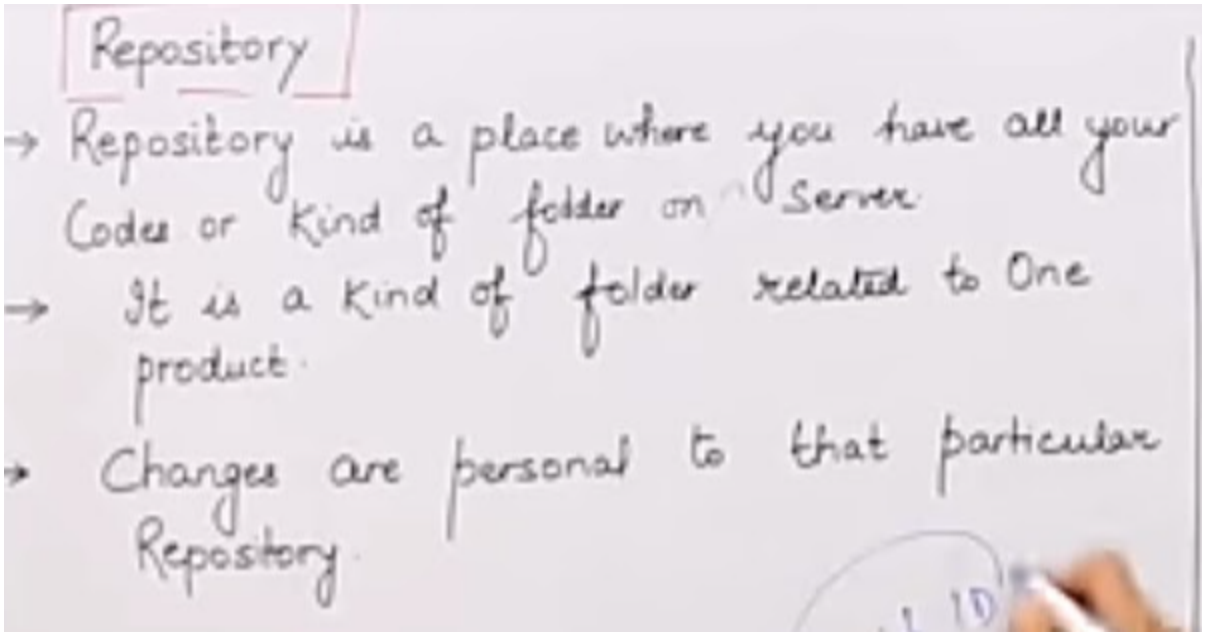

Lec-12 - Stages of Git and its terminology
1 Workspace/Working directory
2 Staging area
3 Local Repo
Stages of git/Workflow
free
.git repo
git install
git init
Staging area
Commit
Push
Pull
github
Central Repository/ Storage
Working Space
Singapore (EC2 linux)
master Branch
Mumbai (EC2 linux Machine)
Branch (Master)
XYZ → commit
XYZ abc → Commit
XYZ Az → Commit



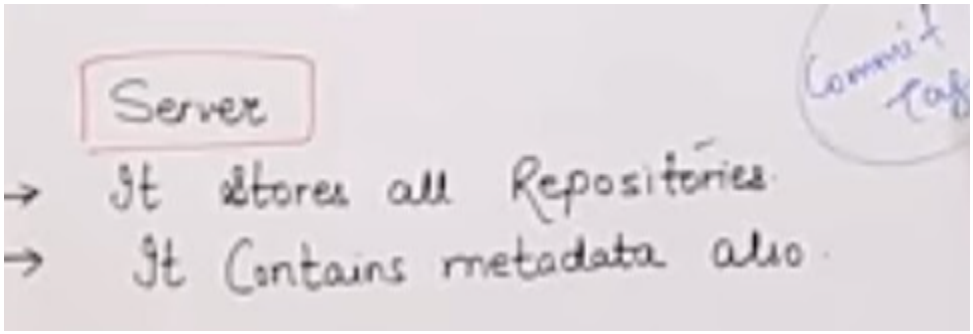


TERMINOLOGIES:

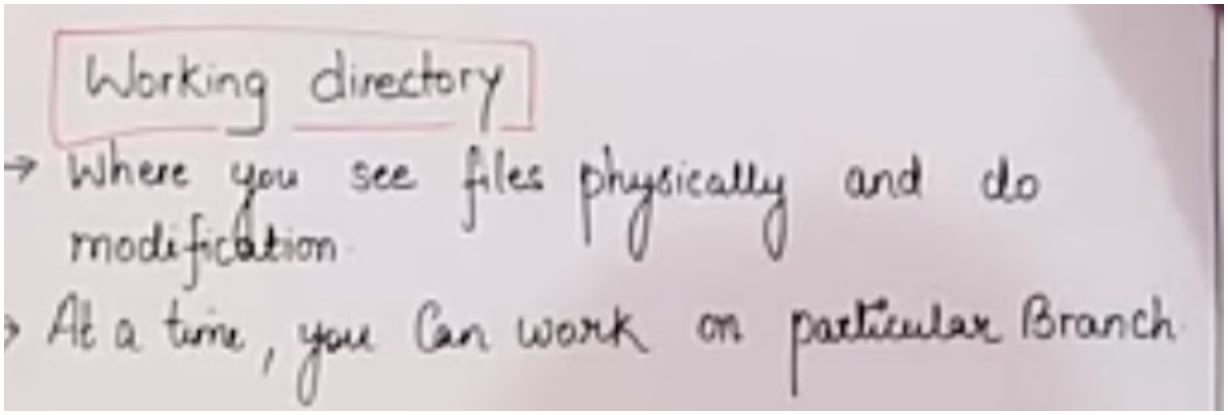
Repository



Server

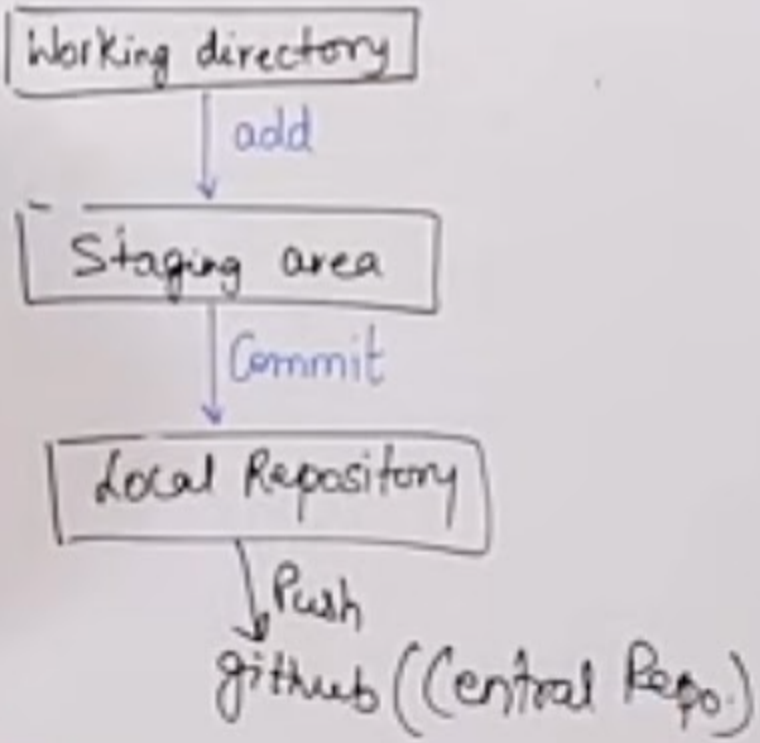


Working Directory

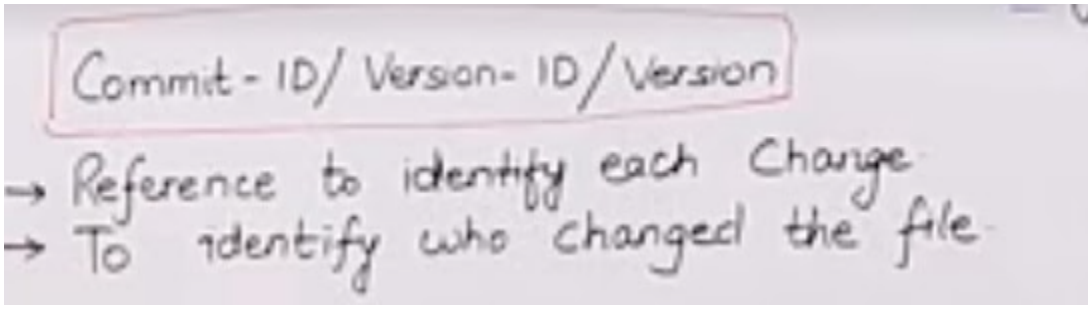


of Git and its terminology

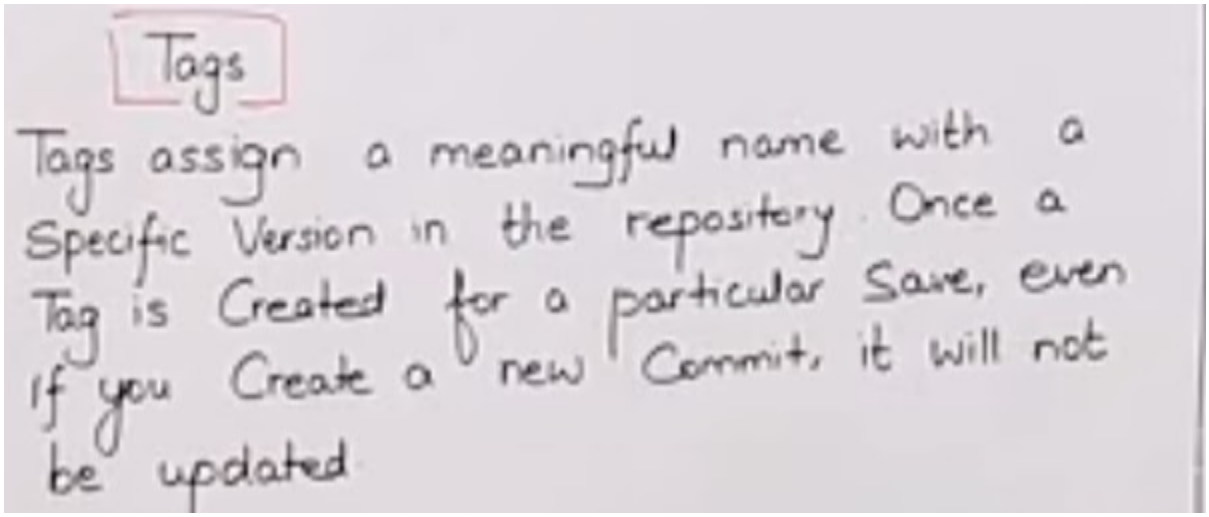
In other CVCS, developers generally makes modifications and Commit their Changes directly to the Repository. But git uses a different Strategy. Git does not track each and every modified file. Whenever you do commit an Operation, git looks for the files present in the staging area. Only those files present in the staging area are considered for Commit and not all the modified files.



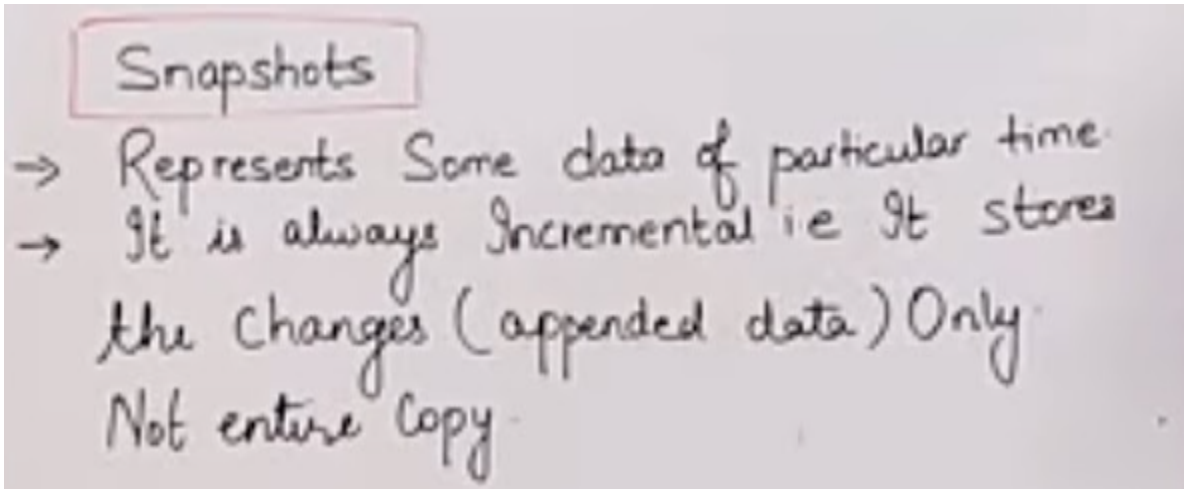
Commid-ID/Version ID/Version



Tags



Snapshots



Commit

Commit

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

→ It is 40 alpha-numeric Characters.

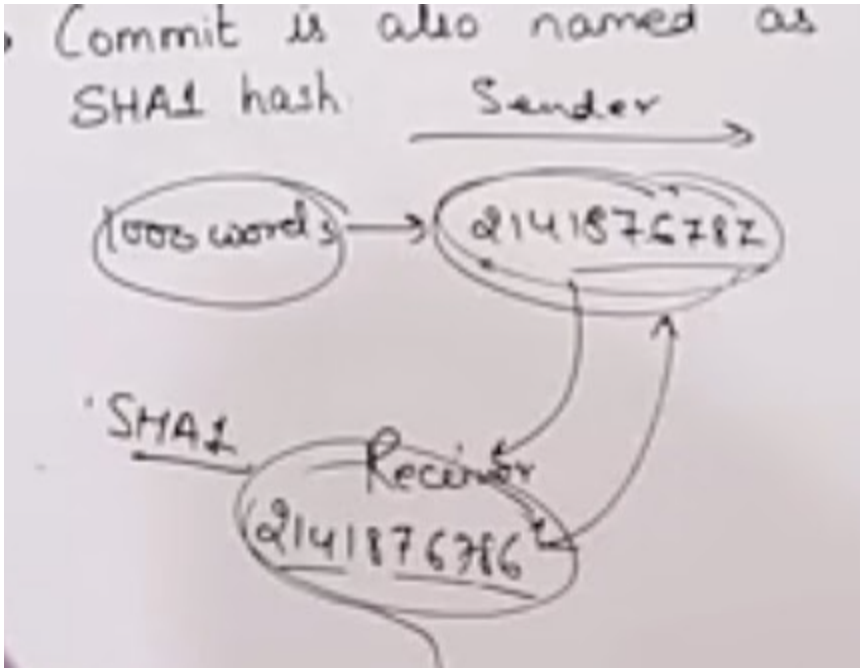
→ It uses SHA-1 Checksum Concept.

→ Even if you change one dot, Commit-ID will get change

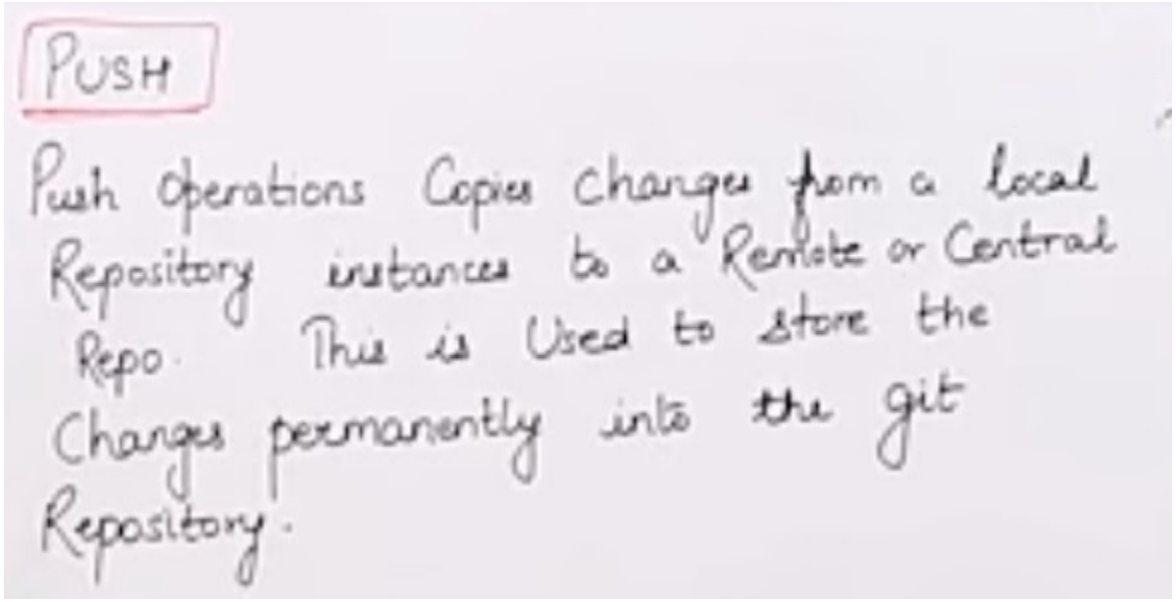
→ It actually helps you to track the Changes.

→ Commit is also named as SHA1 hash.

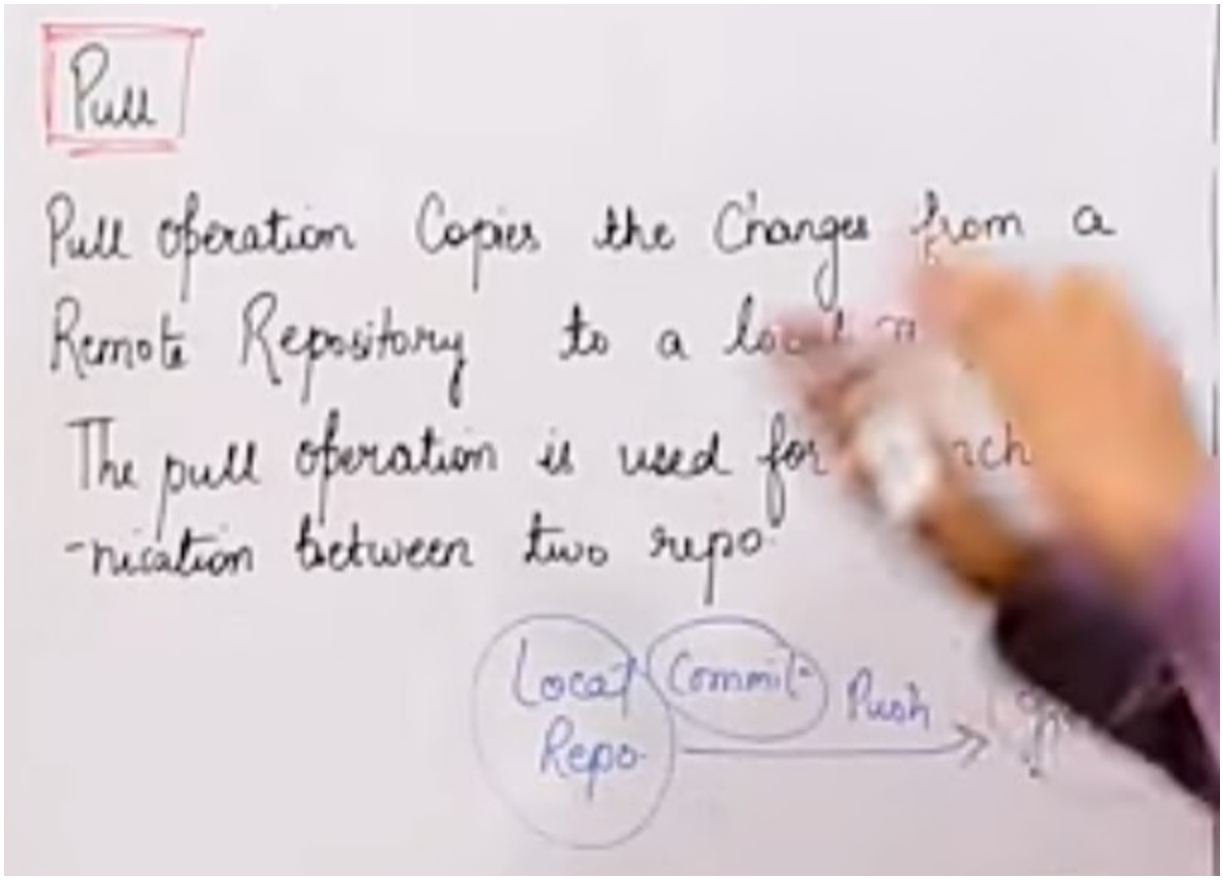
git → Integrity


Commit is also named as
SHA1 hash
Sender
1000 words
2141876782
SHA1
Receiver
2141876782

PUSH


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
Pull operation Copies the Changes from a
Remote Repository to a
The pull operation is used for
-nication between two repo
Local
Repo
Commit
Push

Branch

Branch

Product is same, so one Repository. but different task

→ Each task has one separate Branch

→ finally merges (Code) all Branches

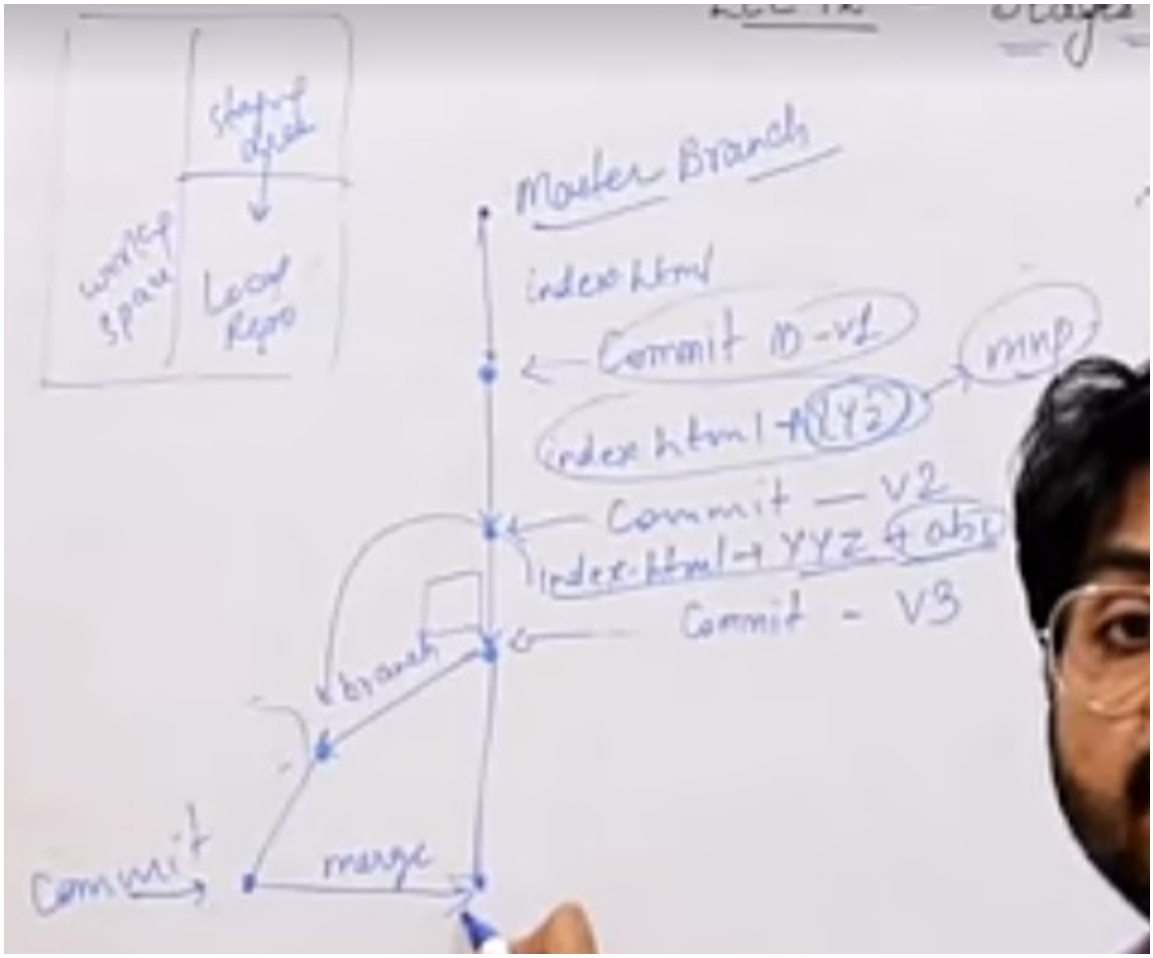
→ useful when you want to work parallely

→ Can Create One branch On the basis of another Branch

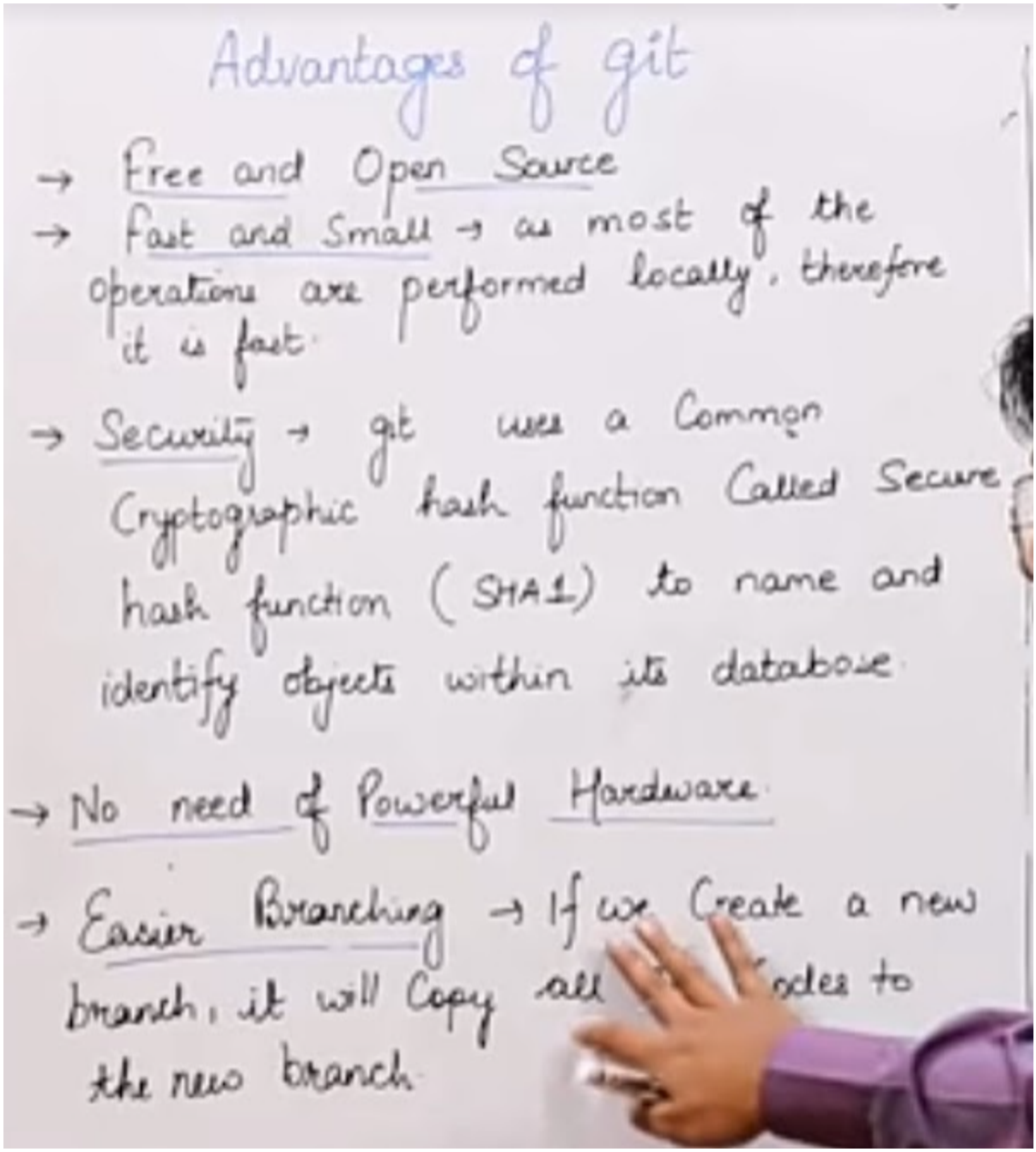
→ Changes are personal to that particular Branch

→ Default Branch is 'Master'

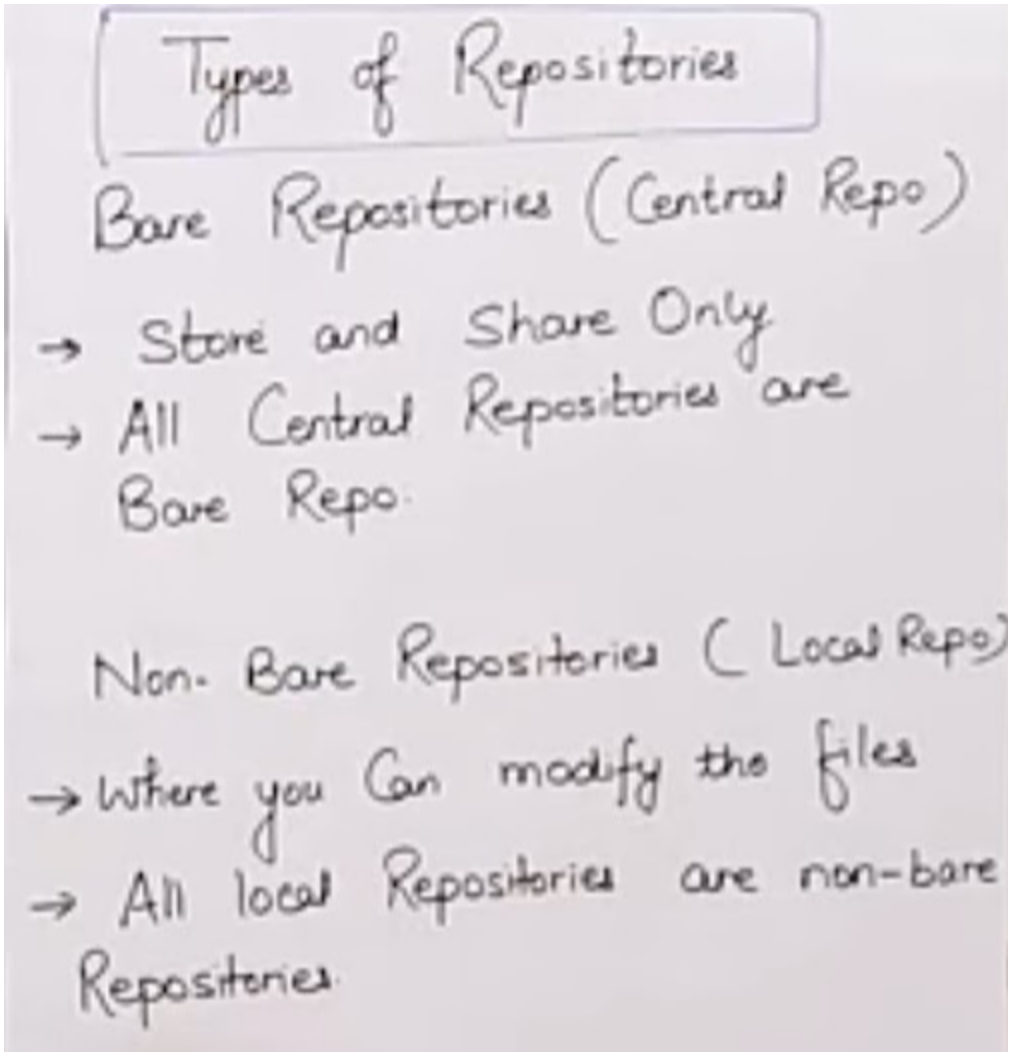
→ File Created in workspace will be visible in any of the branch Workspace until you Commit Once you Commit, then that file belongs to that particular Branch



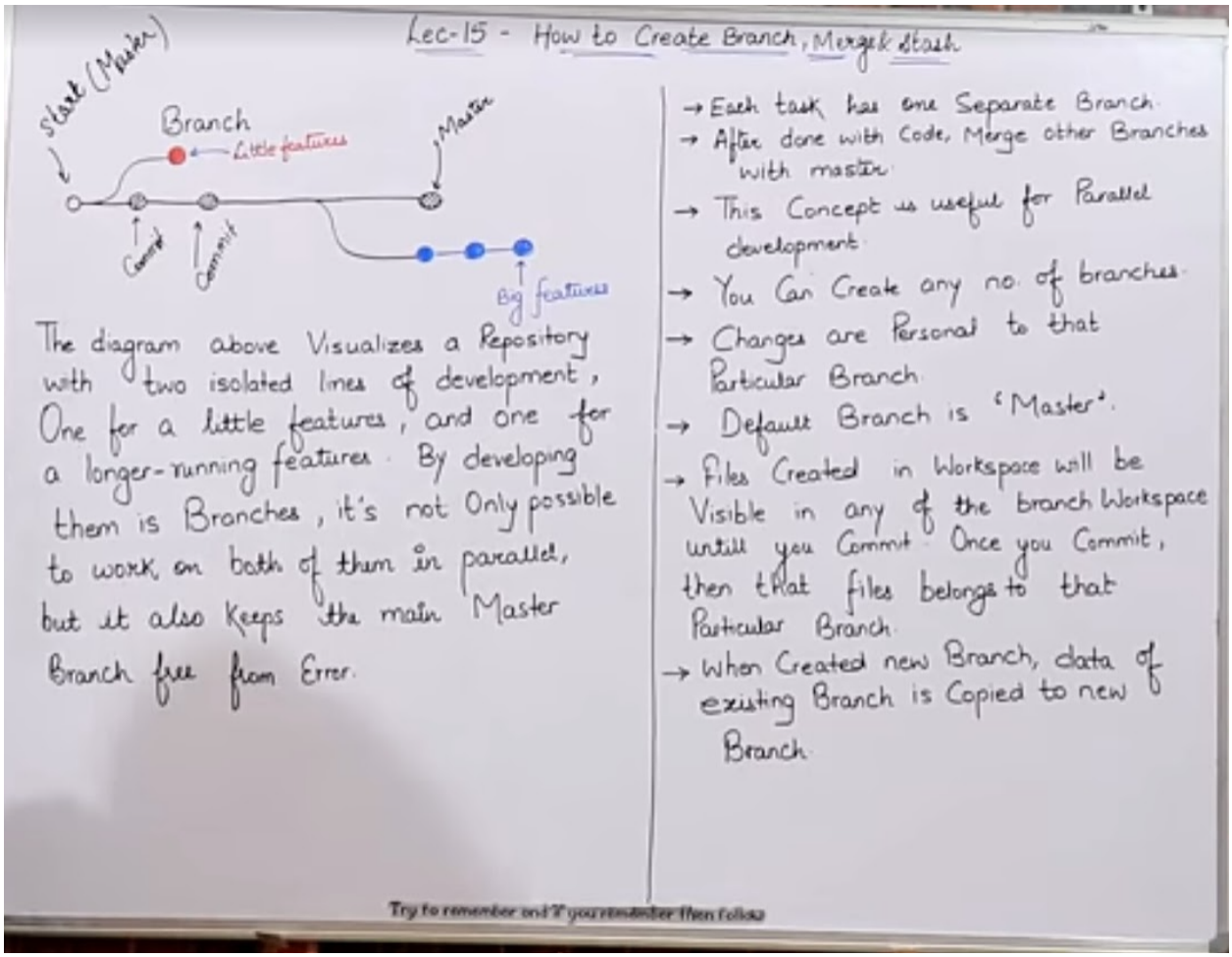
ADVANTAGES OF GIT



Types of Repository


Types of Repositories
Bare Repositories (Central Repo)
→ Store and Share Only
→ All Central Repositories are Bare Repo.
Non-Bare Repositories (Local Repo)
→ Where you Can modify the files
→ All local Repositories are non-bare Repositories.

BRANCH



Git Stashing

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 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 later on Commit it.

→ To stash an item (Only applies to modified files Not new file)