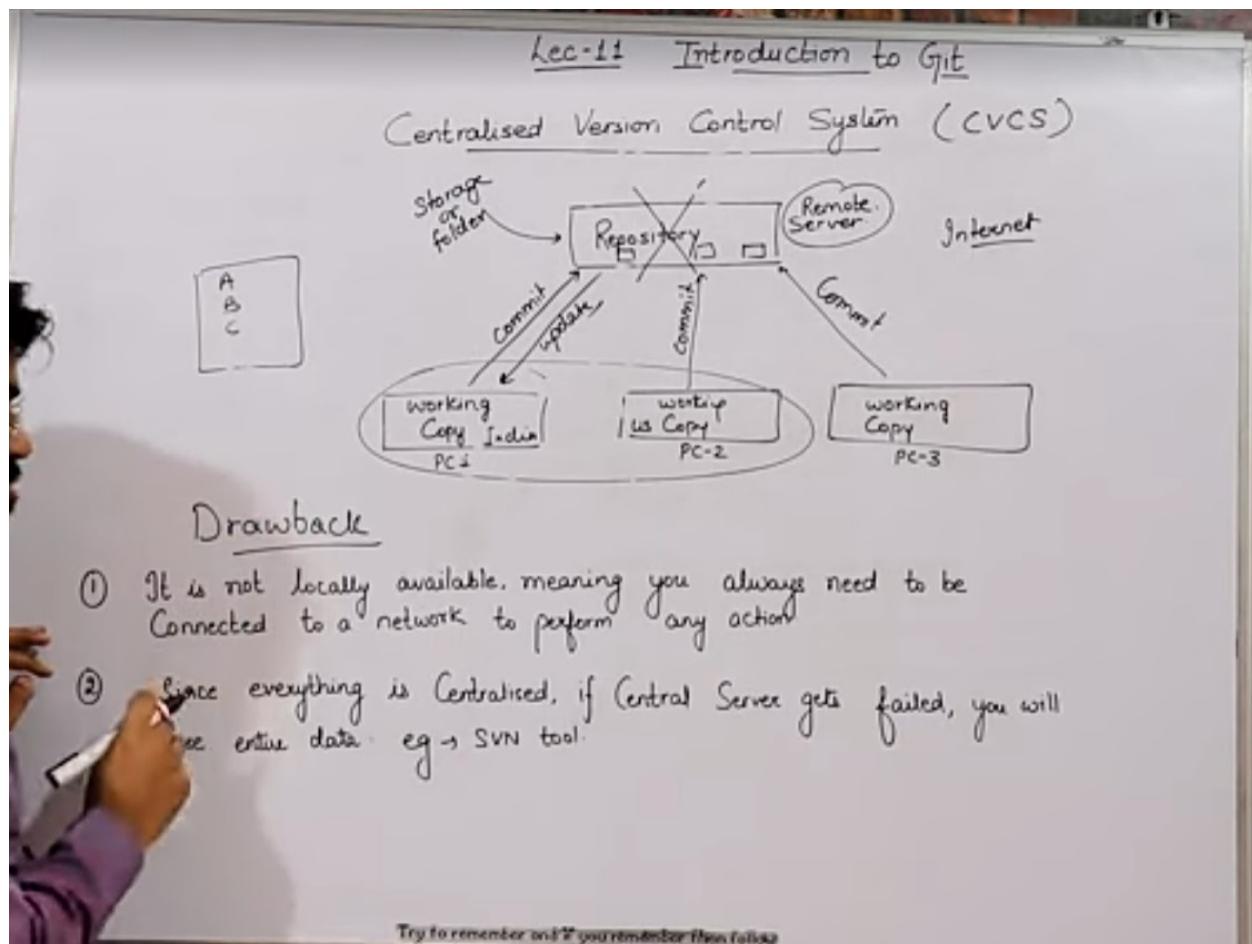


GIT

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

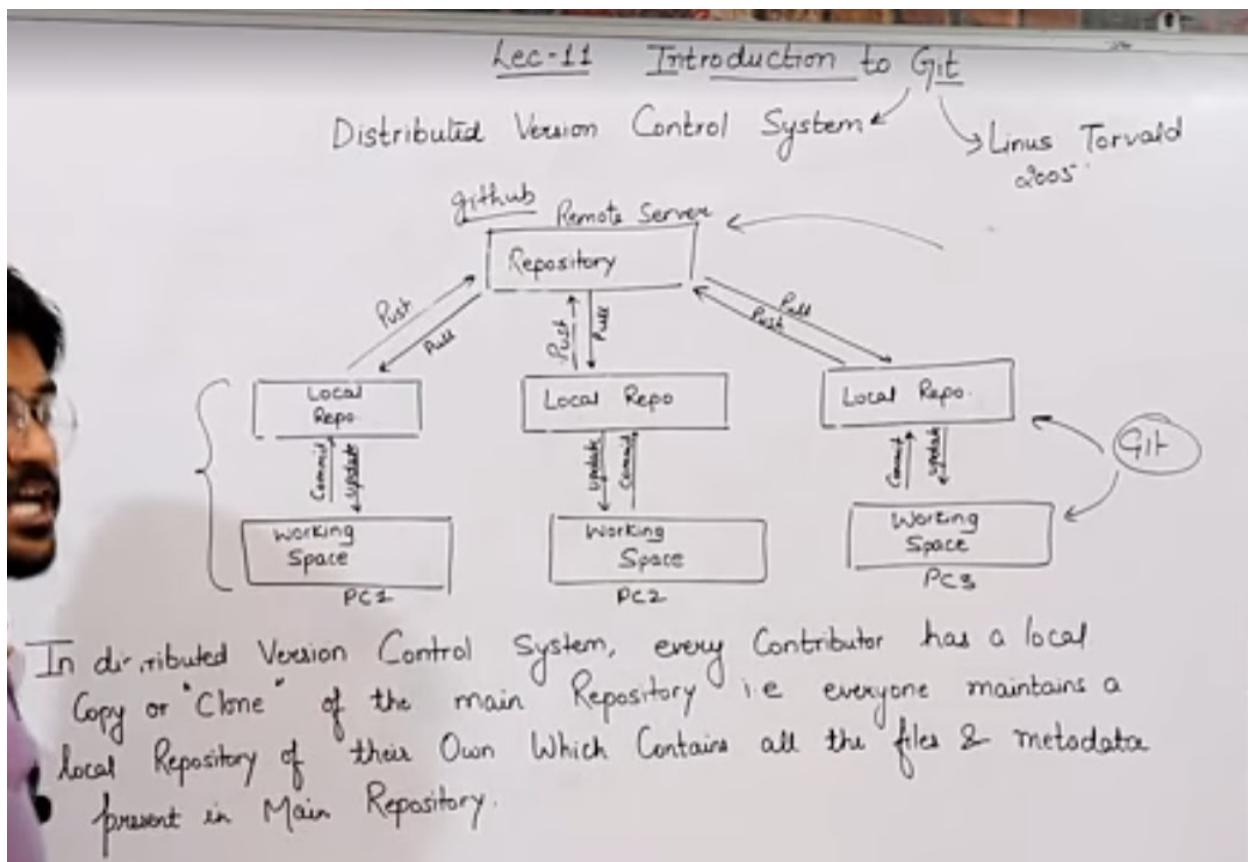
1.Centralised VCS



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 face 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, developer Cannot Work.

2.Distributed VCS

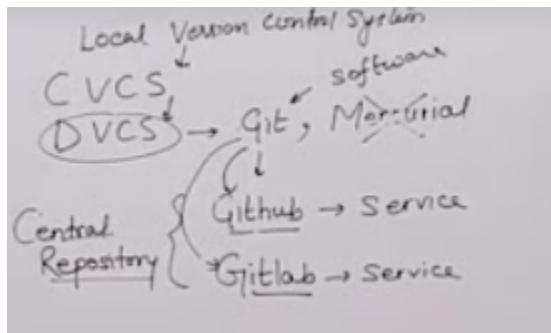


Introduction to Git

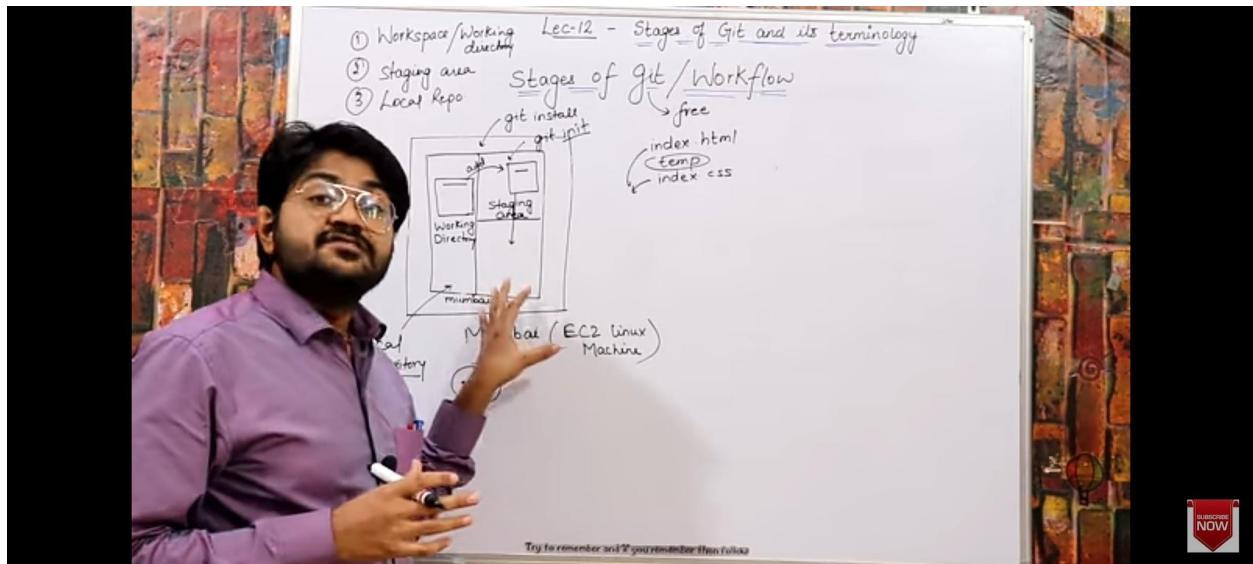
DVCS

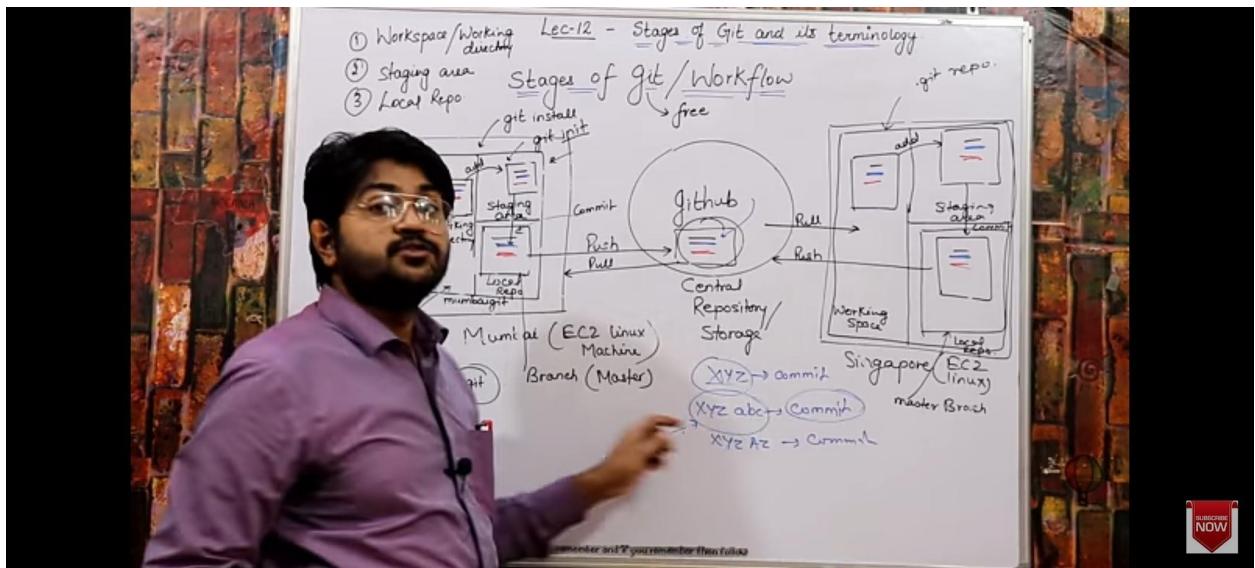
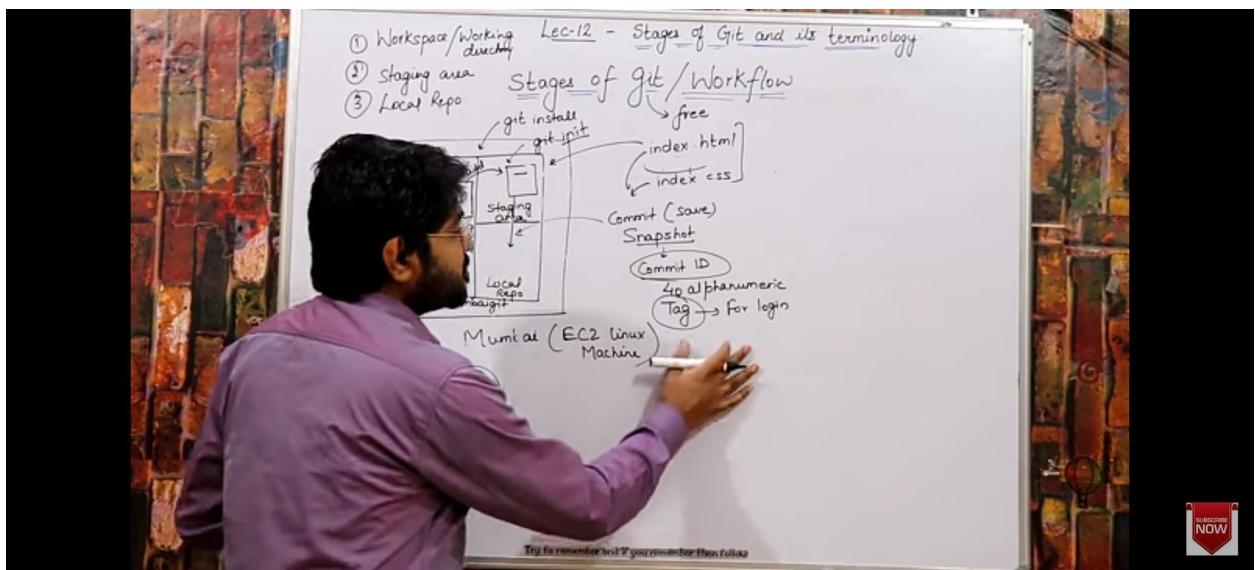
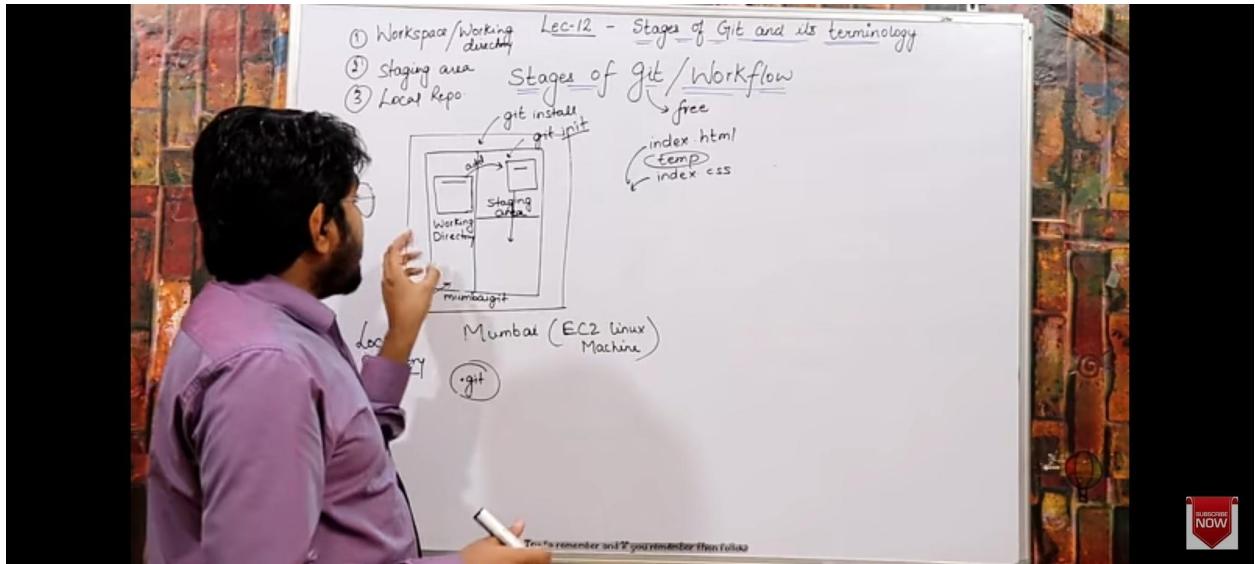
- In DVCS, each Client Can have a local repo. as well and have a Complete history on it. Client need to push the Changes to branch which will then be pushed to server Repository
- DVCS systems are difficult for beginners. Multiple Commands needs to be remembered
- Working on branches is easier in DVCS. 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

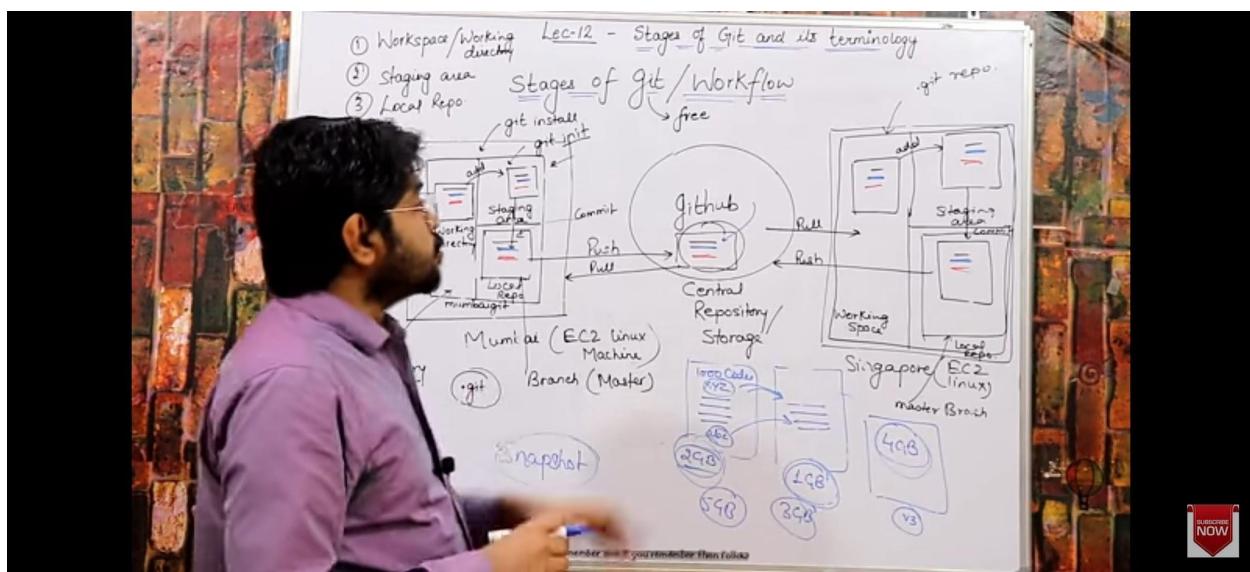
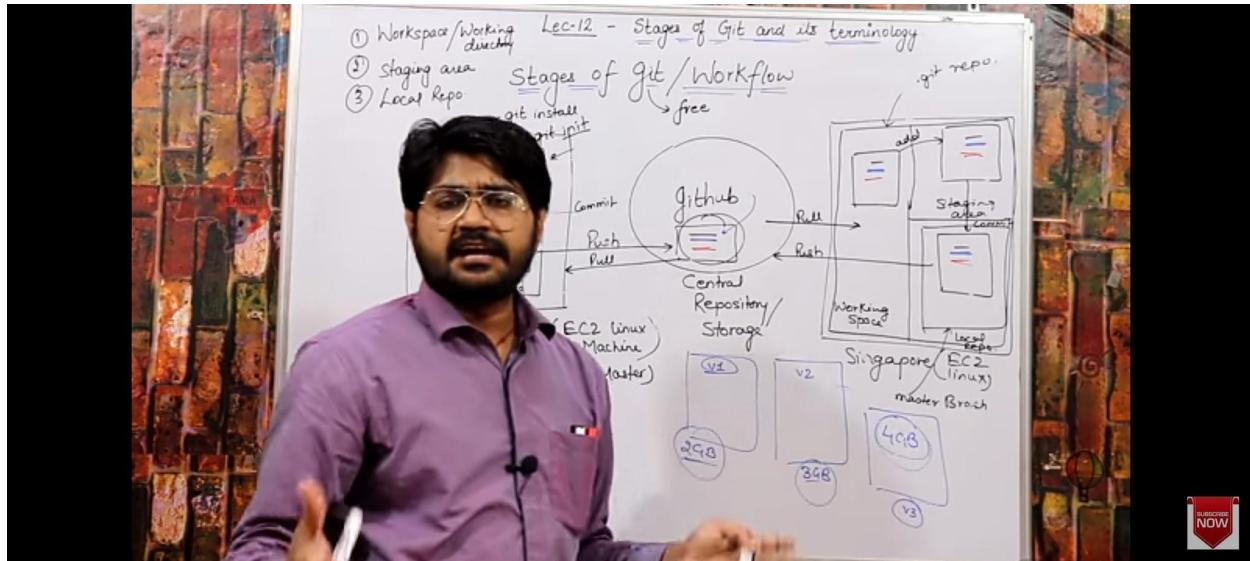
Stages of Git and its Terminology



Stepwise explanation:







TERMINOLOGIES:

Repository

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

Server

Commit Log

- It stores all Repositories.
- It Contains metadata also.

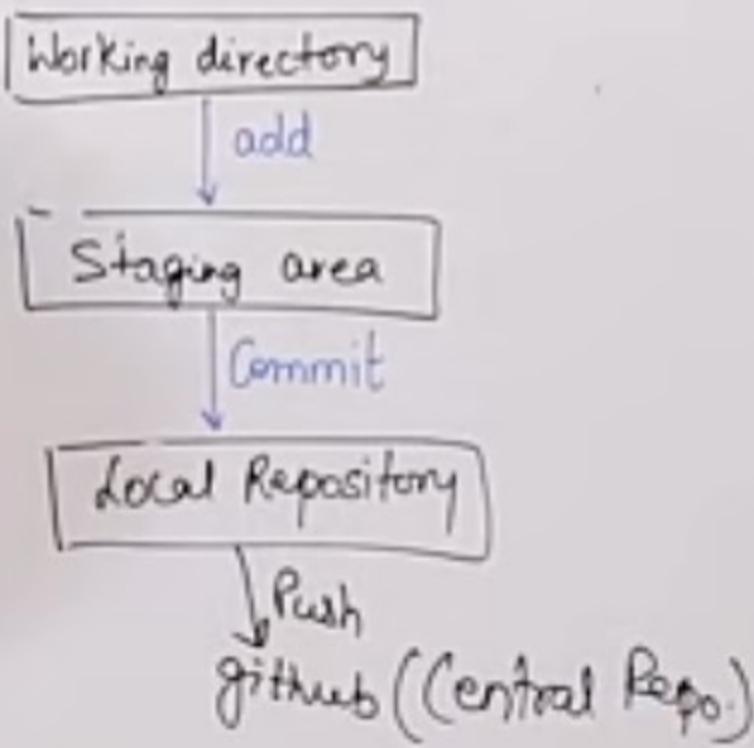
Working Directory

Working directory

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

of Git and its terminology

In other CVCS, developer 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

Commit-ID / Version-ID / Version

- Reference to identify each Change
- To identify who changed the file

Tags

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

Snapshots

- Represents Some data of particular time
- It is always Incremental i.e It stores the changes (appended data) Only Not entire Copy

Commit

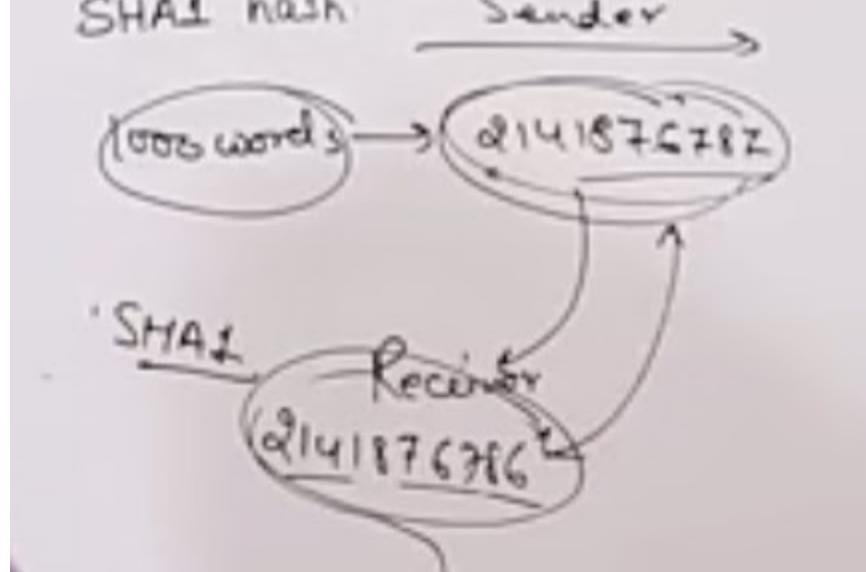
Commit

- 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.

(1000 words) → d14187c

git → Integrity

- Commit is also named as SHA1 hash



PUSH

PUSH

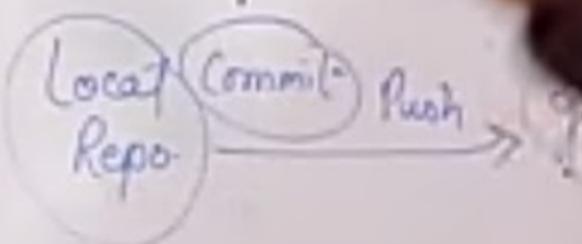
Push Operations Copies changes from a local Repository instances to a Remote or Central Repo. This is used to store the changes permanently into the git Repository.

PULL

Pull

Pull operation Copies the Changes from a Remote Repository to a local repo.

The pull operation is used for synchronization between two repos.

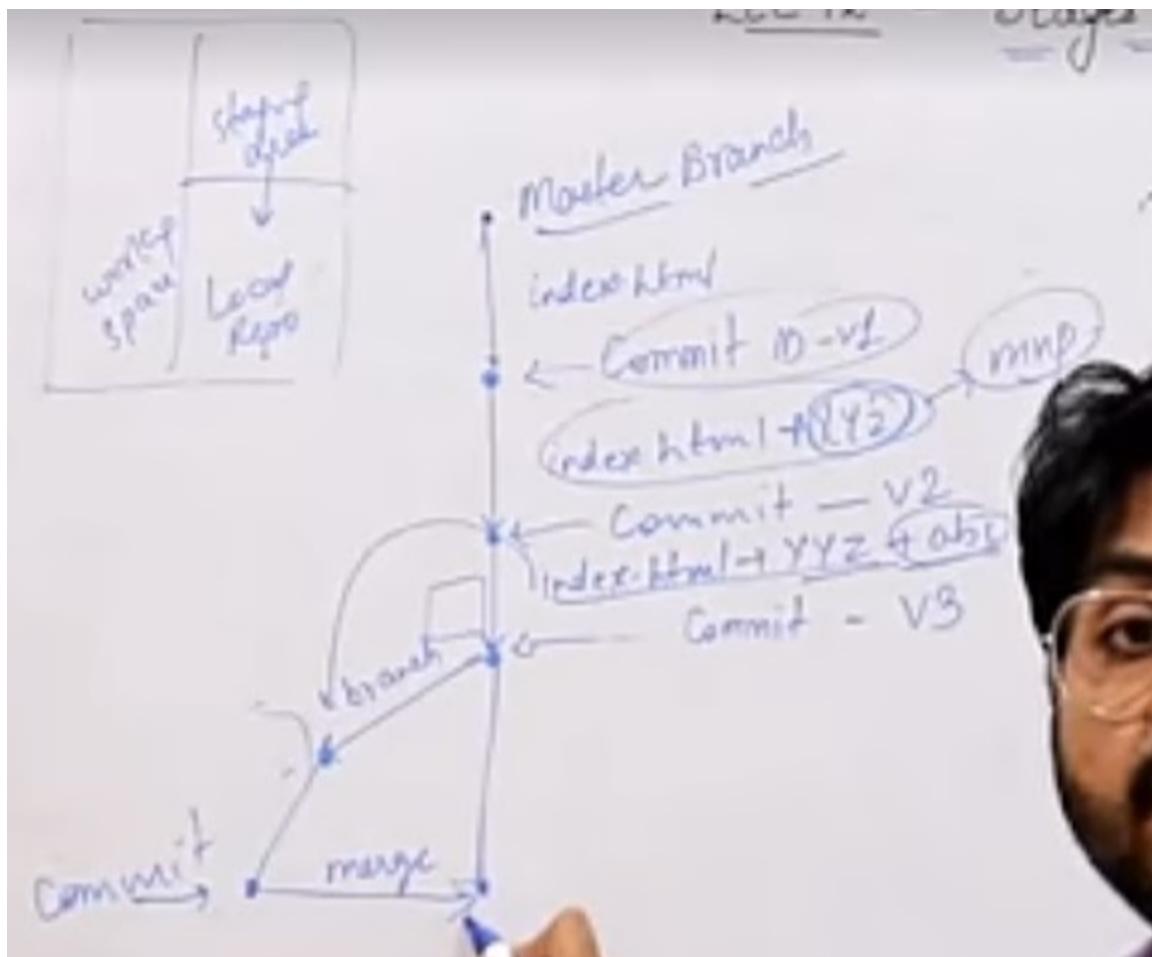


Branch

Branch

Product is same, do one Repository,
but different task.

- Each task has one Separate Branch
- Finally merges (Code) all Branches
- Useful when you want to work parallelly.
- 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

Advantages of git

- Free and Open Source
- Fast and 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 nodes to the new branch.

Types of Repository

Type of Repositories

Bare Repositories (Central Repo)

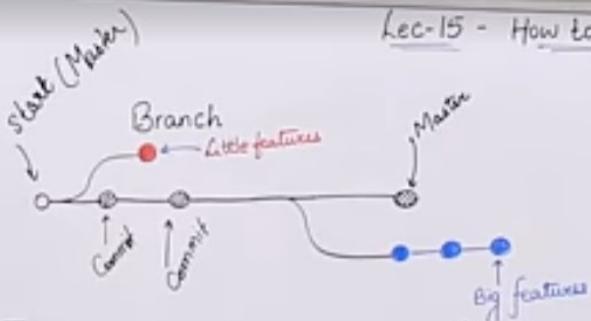
- Store and Share Only
- All Central Repositories are Bare Repo.

Non-Bare Repositories (Local Repo)

- Where you can modify the files
- All local Repositories are non-bare Repositories.

BRANCH

Lec-15 - 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.

- 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 file belongs to that Particular Branch.
- When Created new Branch, data of existing Branch is Copied to new Branch.

Try to remember and if you remember then follow

Git Stashing

Lec-15 - How to C

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 files)