GIT AND GITHUB

WHAT IS GIT ?

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.
It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

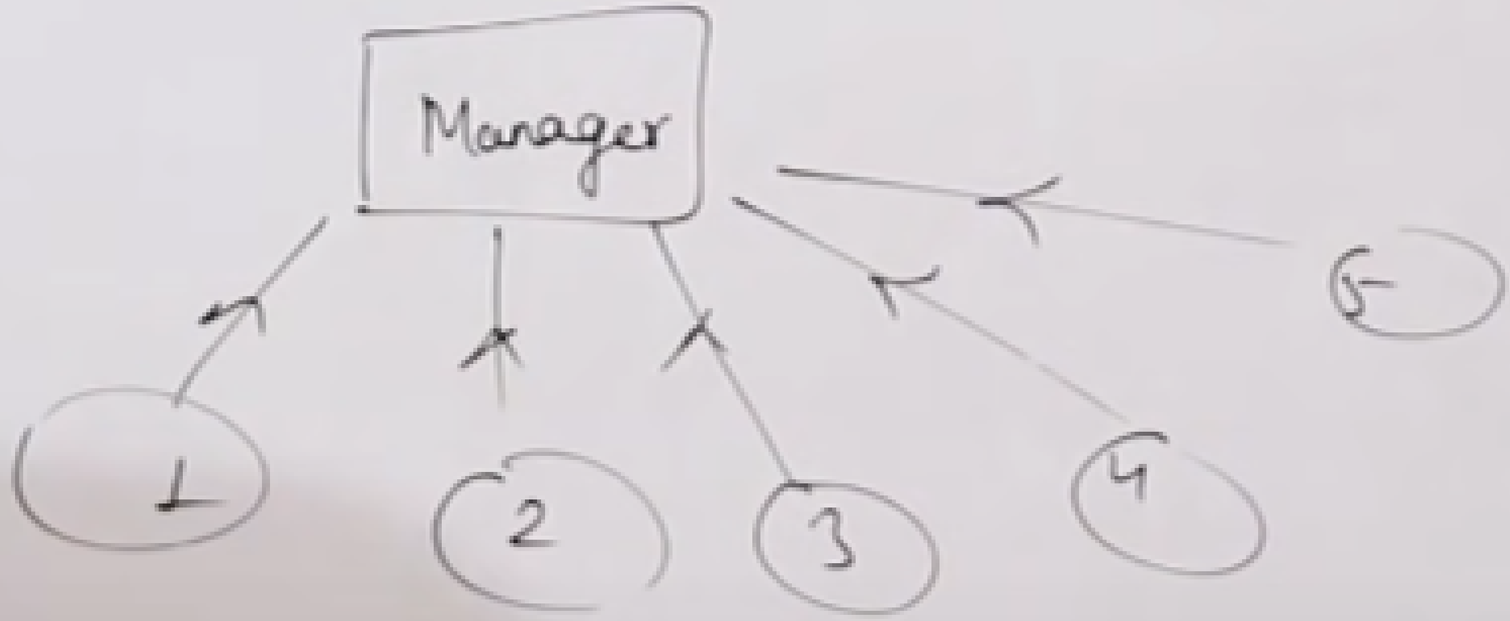
Why Git ?

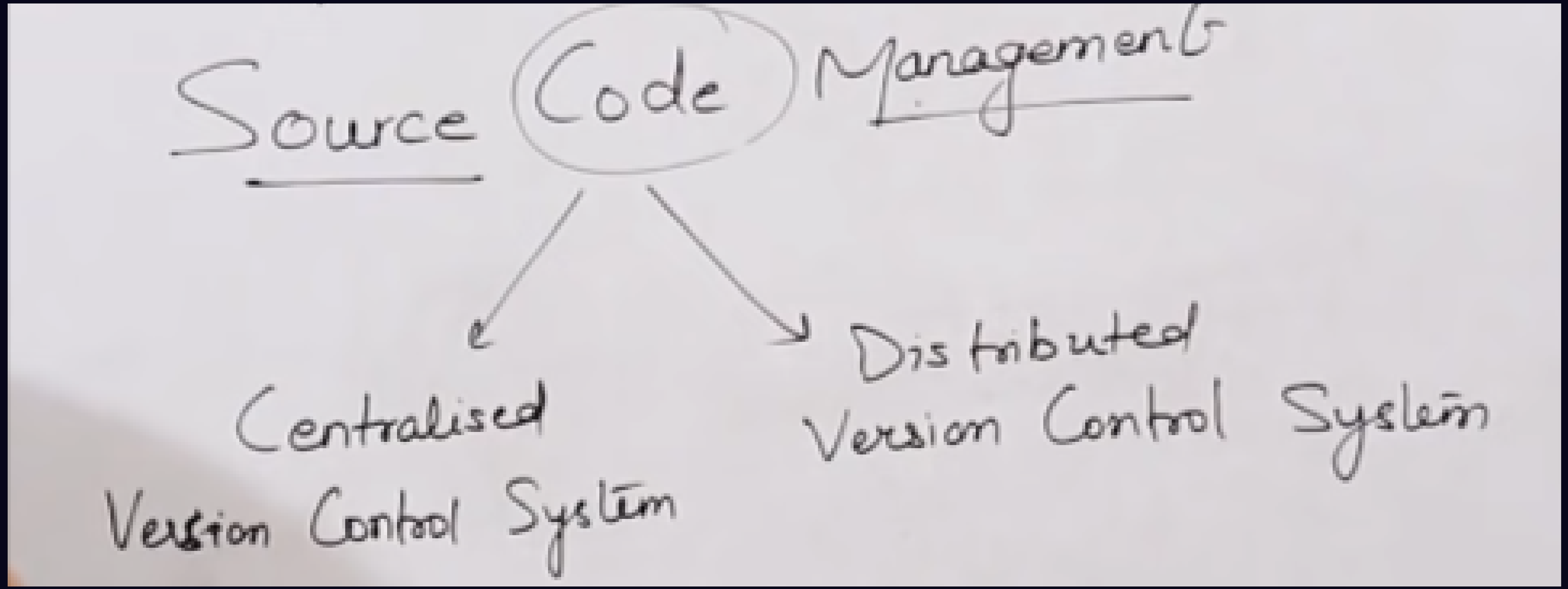
- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

Software Configuration Management

or

Source Code Management



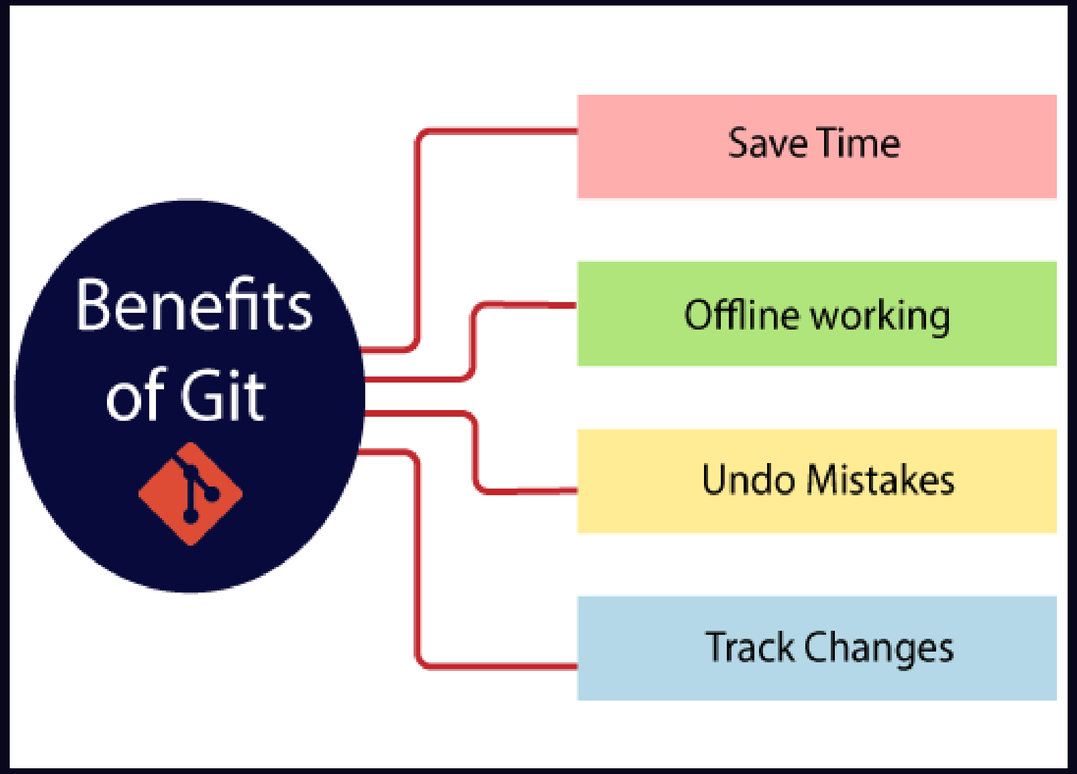

Source Code Management
Centralised Version Control System
Distributed Version Control System

Git is an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

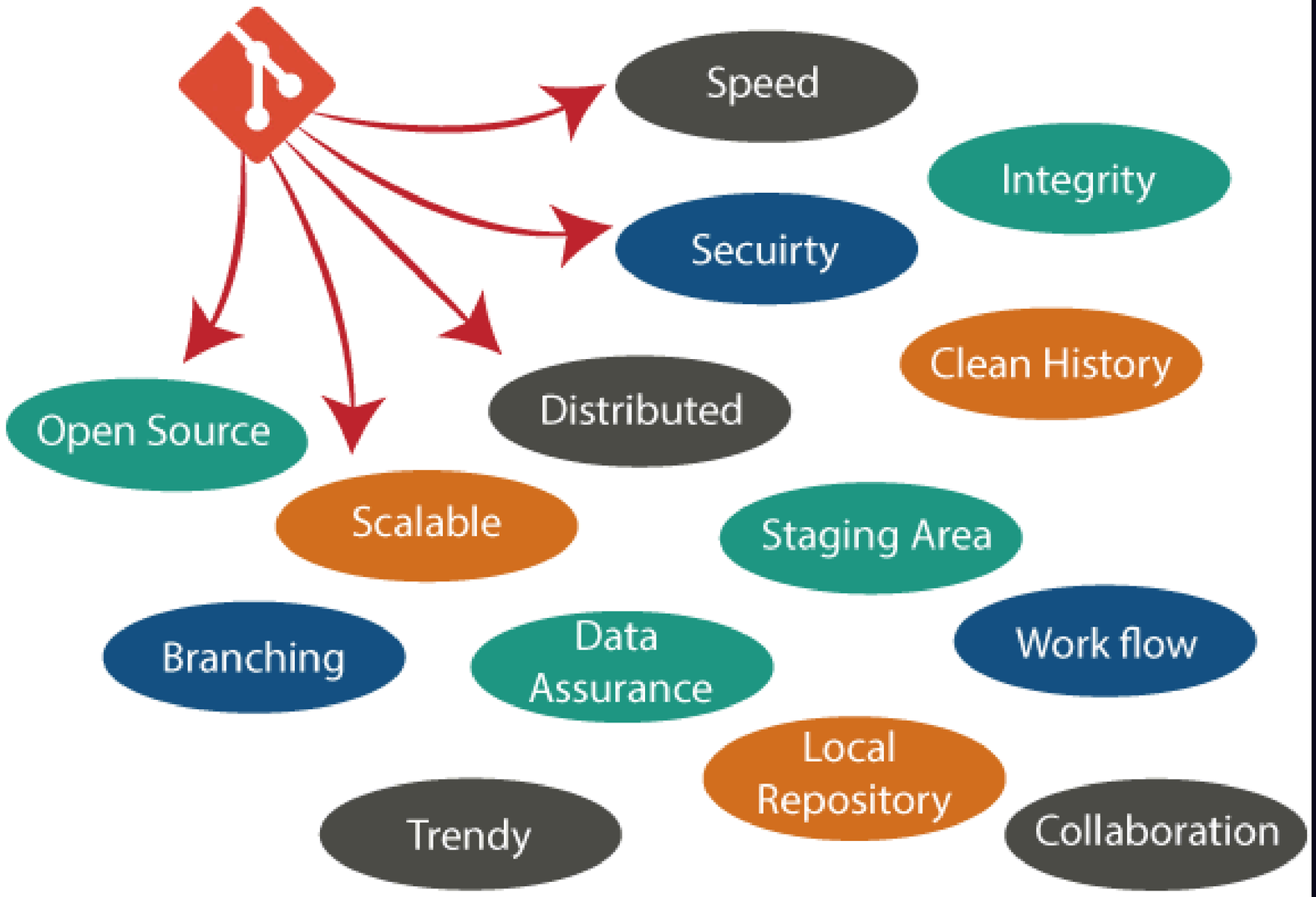
Git is foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly.

Benefits of Git

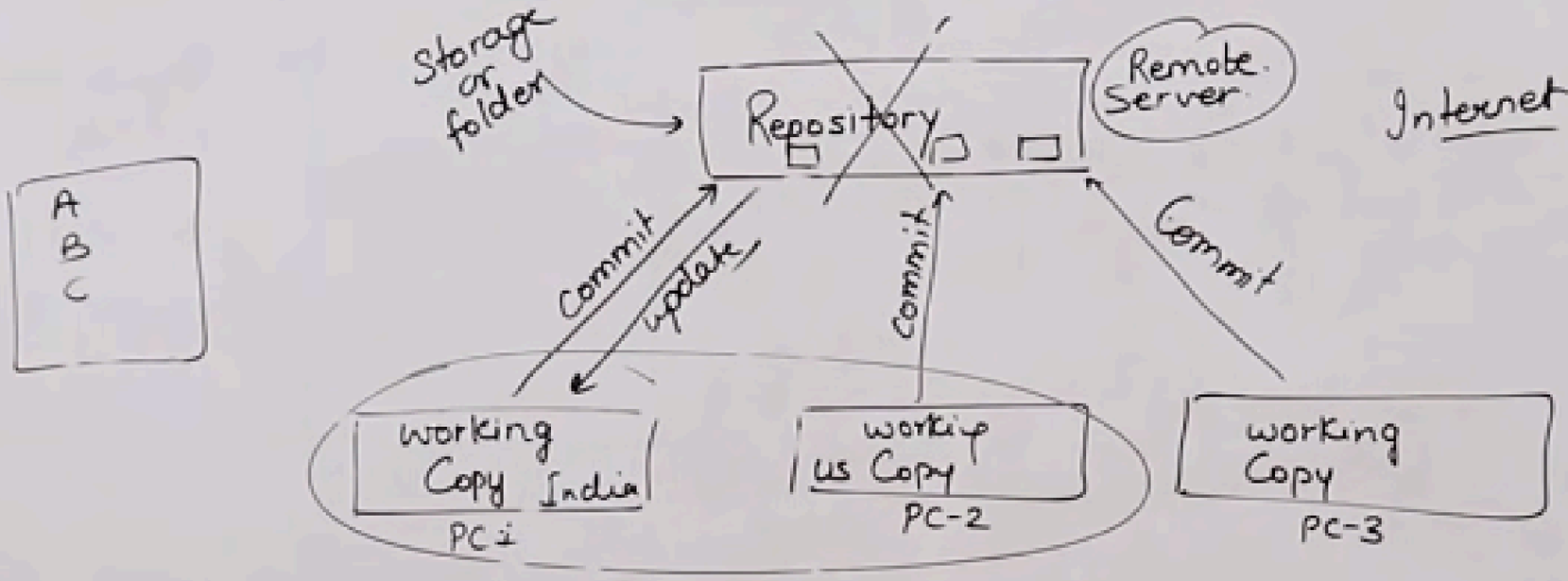
A version control application allows us to keep track of all the changes that we make in the files of our project. Every time we make changes in files of an existing project, we can push those changes to a repository. Other developers are allowed to pull your changes from the repository and continue to work with the updates that you added to the project files.



Why Git?


Speed
Integrity
Secuirty
Clean History
Distributed
Open Source
Scalable
Staging Area
Branching
Data Assurance
Work flow
Local Repository
Trendy
Collaboration

Centralised Version Control System (CVCS)



Drawback

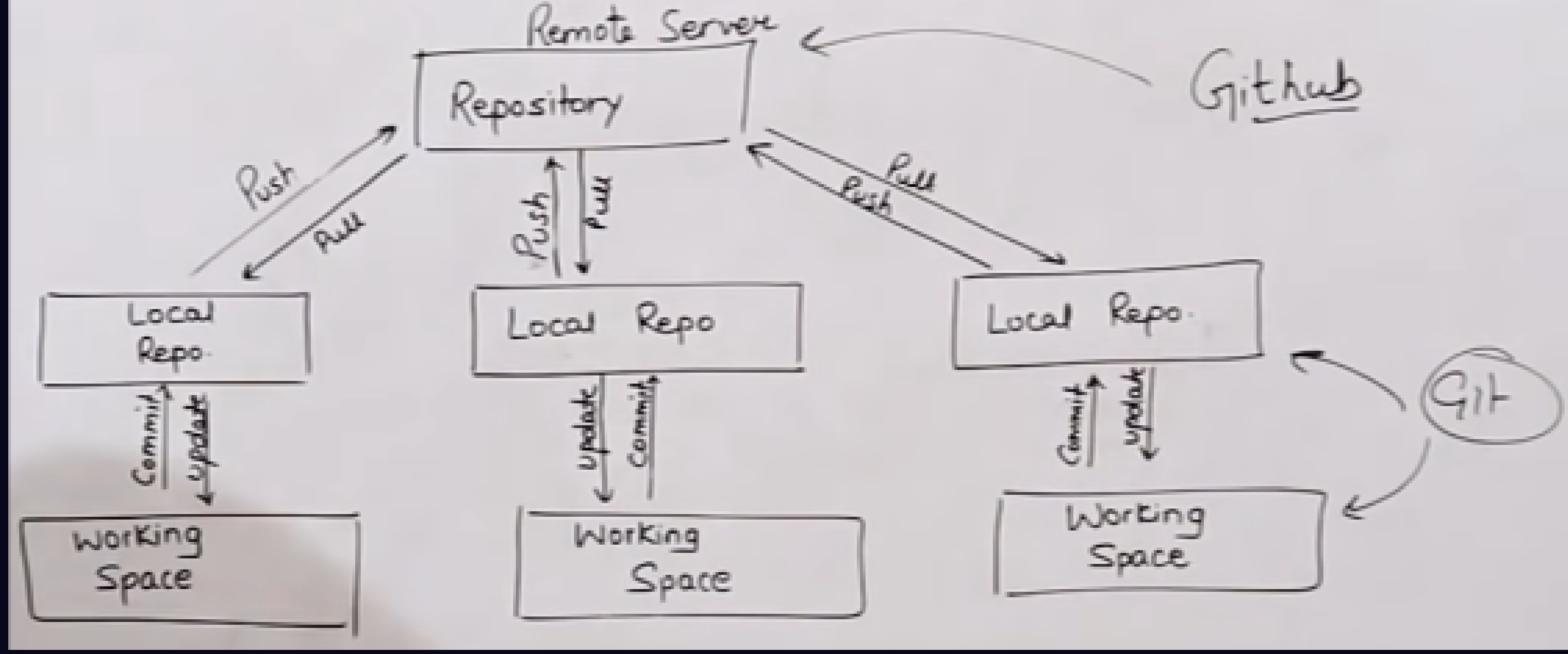
It is not locally available, meaning you always need to be Connected to a network to perform any action.

Since everything is Centralised, if Central Server gets failed, you will lose entire data eg → SVN tool.

Lec-11 Introduction to Git

Distributed Version Control System

Linus Torvald 2005



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.

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, developers Cannot Work.

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.

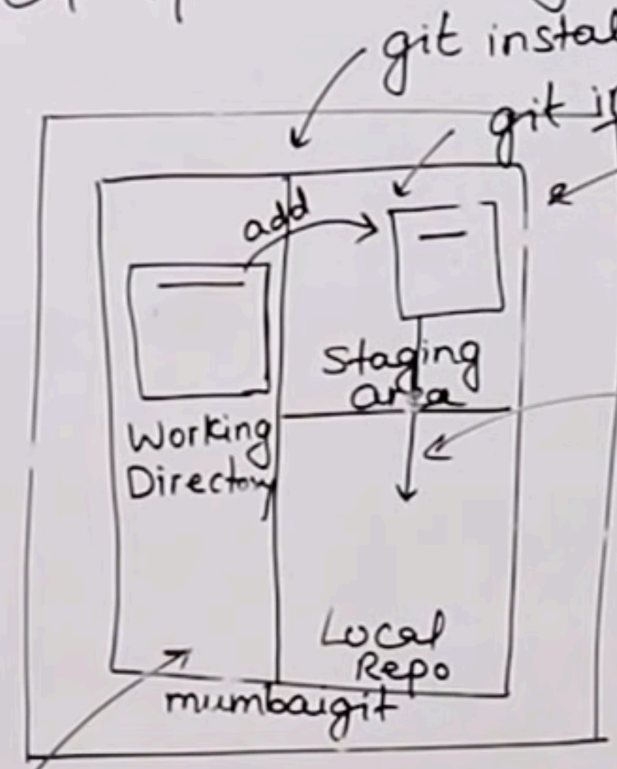
→ DVCS System are working fine on offline mode as a Client Copies the Entire repository on their local machine.

→ 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.

1. Workspace / working directory
2. Staging area
3. Local Repo

Stages of git / Workflow



free

index.html
index.css

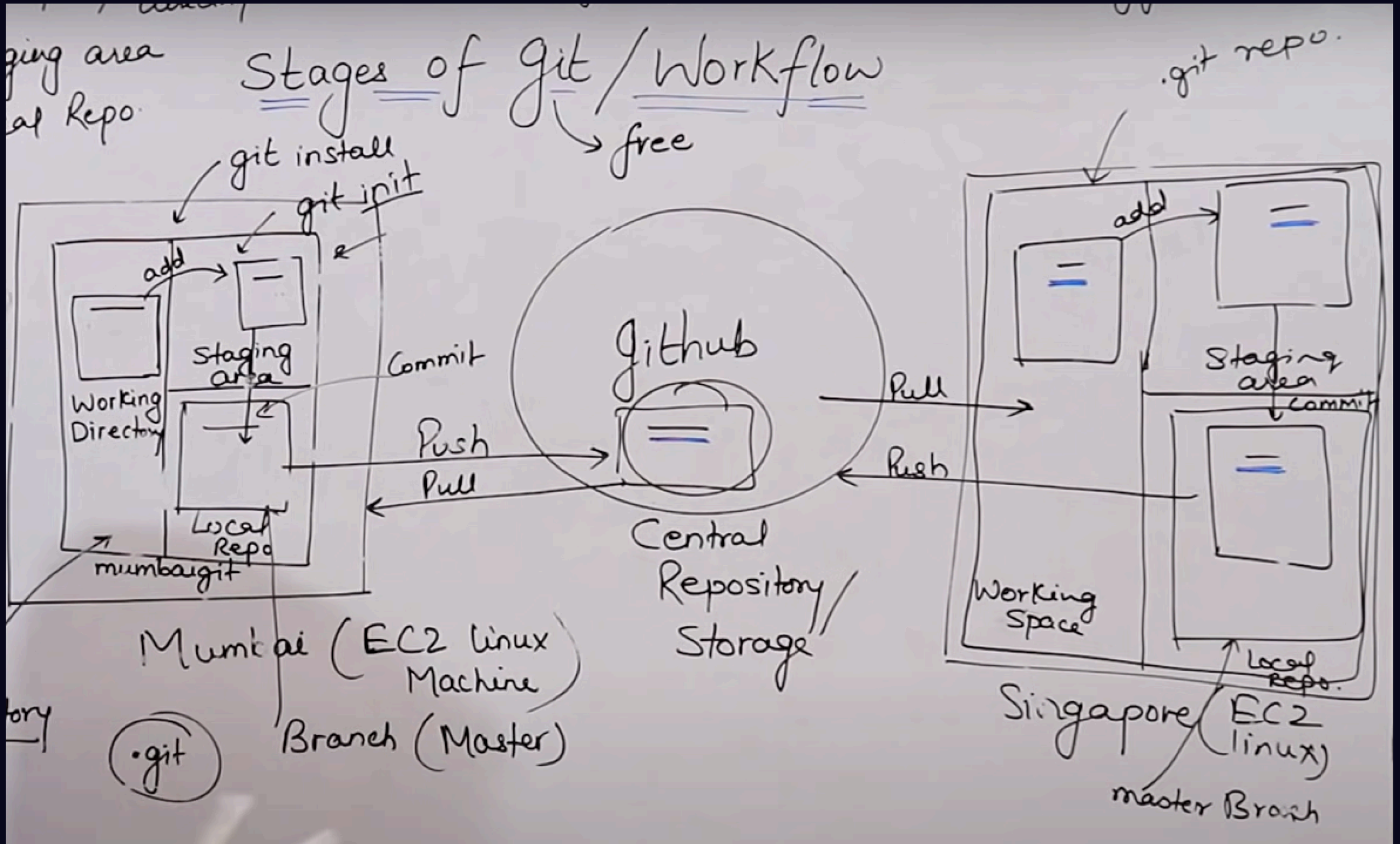
Commit (save)
Snapshot

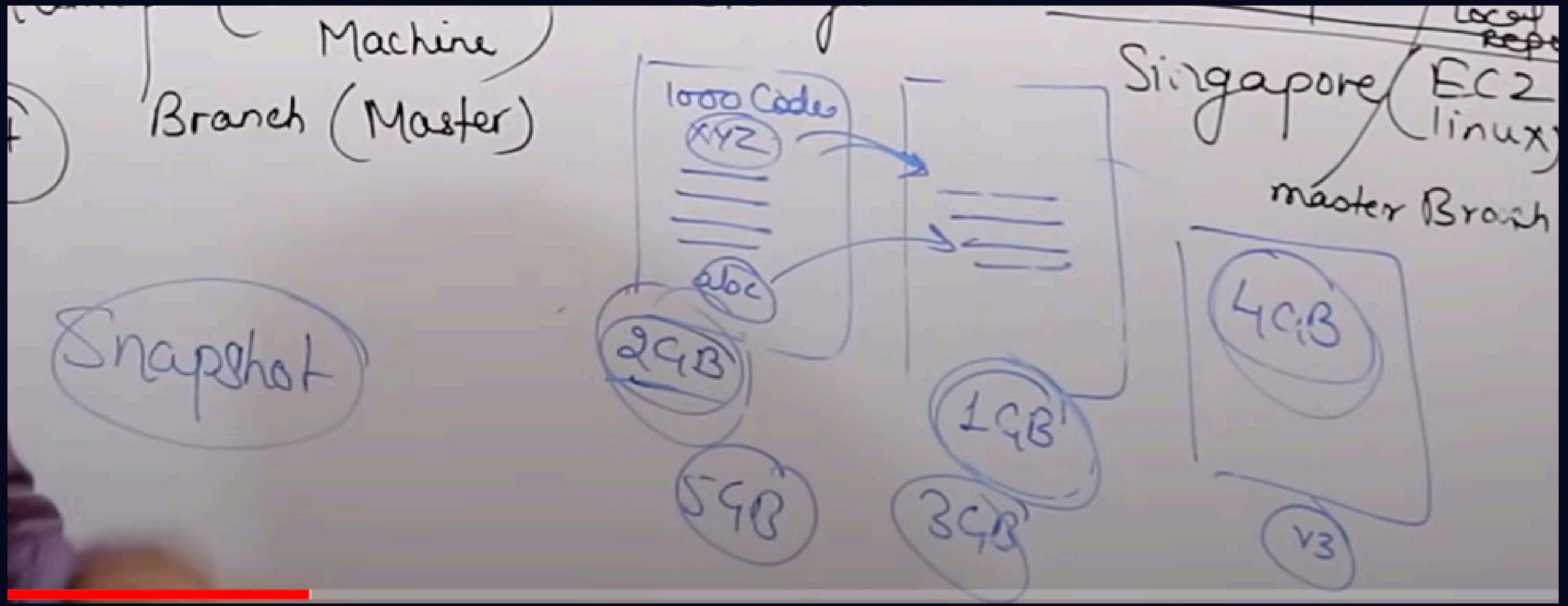
Commit ID
40 alphanumeric
Tag → For login

Mumbai (EC2 linux Machine)

Local Repository

.git


Stages of git / Workflow
free
git install
git init
add
Staging area
Working Directory
Commit
Local Repo
mumbaigit
Push
Pull
github
Central Repository / Storage
Pull
Push
.git repo.
add
Staging area
commit
Working Space
Local Repo.
Mumbai (EC2 linux Machine)
Branch (Master)
.git
Singapore (EC2 linux)
master Branch


Machine
Branch (Master)
1000 Code
XYZ
abc
2GB
5GB
Snapshot
1GB
3GB
Singapore
EC2
linux
master Brach
4GB
V3

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 Contains metadata also.

Working directory

→ Where you see files physically and do modification.

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

Commit

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

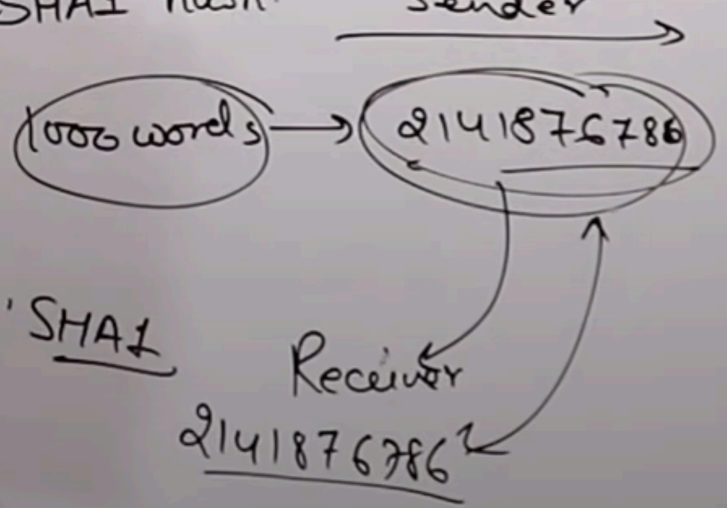
→ It is 40 alpha-numeric Characters.

→ 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.



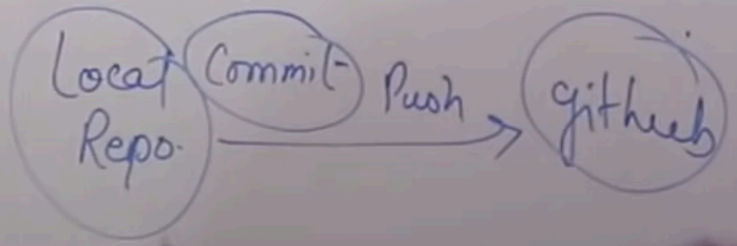
PUSH

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 operation Copies the Changes from a Remote Repository to a local machine.

The pull operation is used for synchro-nisation between two repo.



Branch

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

→ Each task has one Separate Branch.

→ Finally merges (Code) all Branches.

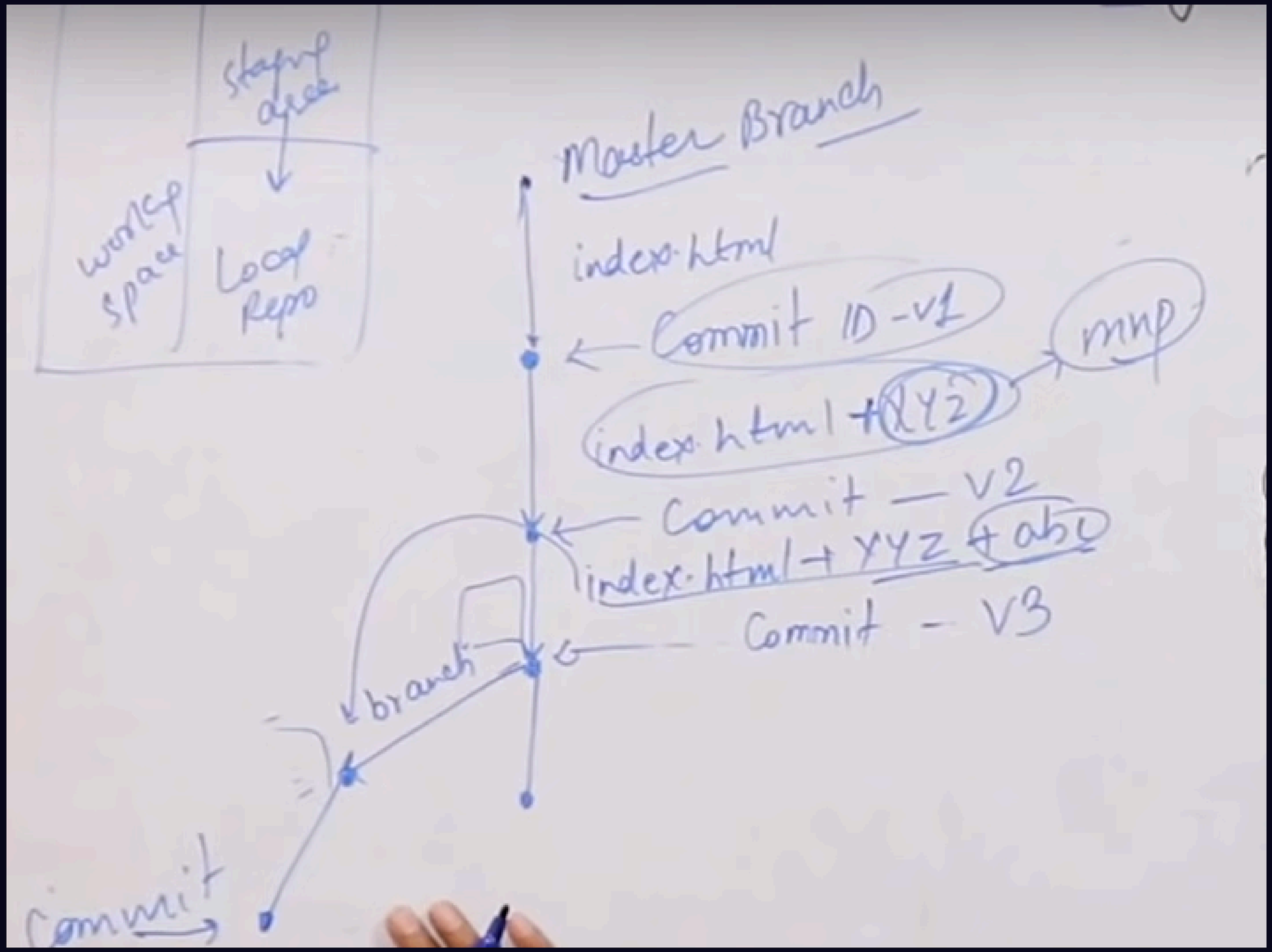
→ useful when you want to work parallely.

→ 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 untill you Commit. Once you Commit, then that file belongs to that particular Branch.


staging area
working space
Local Repo
Master Branch
index.html
Commit ID - V1
mnp
index.html + XYZ
Commit — V2
index.html → XYZ + abc
Commit - V3
branch
commit

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 the Codes to the new branch.

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.