**28/02/2025**

**GIT:**

GIT stands for global information tracker

Git is used for maintaining the histories of versions

Git consists of all the features of previous versions

Git is also known as version control system

**Source code:** finalized code is called as source code

Git is used for security purpose

It is maintaining all the versions along with modifications done on specific application

Git can record all the modifications done by timestamp

Git is a distributed version control system

**Git is mainly used for 2 reasons**

* Version control system
* It is used for collaboration

**Why do we use git:**

Maintain multiple versions of the same file

Functionalities of VCS

Allow multiple developers to write a code simultaneously

It doesn’t allow overwriting

We can not copy code

Git is fast when comparing with other version control tools.

Multiple developers collaborate and work on the same module

This will work as backup for project source code

Type of VCS:

1. Centralized Version Control System:

* CVCS is a sub version control system
* CVCS consist of current version data only
* Developers should connect to the server after then only they should start doing their code
* Everyone can access the server

**Disadvantages:**

No version history maintained

No security

If server destroyed then everything will be loss

1. Distributed Version control System:

* It is a server
* DVCS is also called as Decentralized version control system
* Each and every developer need not want to connect to server every time while they want to write code

**Difference between DVCS and CVCS**

**Centralized Version Control System:**

Centralized version control is a version control system using client server architecture and server contains all the history of source code.

CVCS systems are easy to learn and set up

Working on branches is difficult in CVCS. Developer often faces merge conflicts

CVCS systems do not provide offline access

CVCS is slower as every command need to communicate with server

If CVCS server is down, developers can not work

**Distributed Version Control System:**

Distributed version control is a version control where each client can have same copy of source code as a server has and both server and client maintain history of source code

DVCS systems are difficult for beginners. Multiple commands needs to be remembered

Working on branches in easier in distribution version control systems. Developer faces lesser conflicts

Distributed version control systems are workable offline as a client copies the entire repository on their local machine.

DVCS is faster as mostly user deals with local copy without hitting server every time.

If DVCS server is down, developer can work using their local copies.

Version control system:

A **version control system** is a tool used to manage changes to files over time. It allows users to track modifications, revert to previous versions, and manage conflicts when multiple people work on the same files.

**Types of VCS:**

1. **Centralized VCS (CVCS)** – A single central repository (e.g., SVN).
2. **Distributed VCS (DVCS)** – Multiple copies of the repository (e.g., Git, Mercurial).

* Tracks changes with timestamps and author details.
* Enables rollback to previous versions.
* Supports branching and merging.
* Prevents accidental overwrites.
* Provides a history of project evolution.

**Collaboration**

**Collaboration** refers to the process of multiple people working together on a project. This can involve communication, task management, and shared access to documents or code.

**Types of Collaboration:**

1. **Asynchronous Collaboration** – Team members contribute at different times (e.g., using Git, Google Docs).
2. **Synchronous Collaboration** – Team members work simultaneously (e.g., pair programming, live editing in Google Docs).

* Shared access to files and code.
* Real-time or asynchronous contributions.
* Communication tools (Slack, Zoom, GitHub Discussions).
* Conflict resolution mechanisms (pull requests, comments).
* Task tracking (Jira).

**GIT:**

Git is a client server architecture

Here git bash is a client

Git is the server

**Repository:**

Group of project files are stored in single area

It can be thought of as a database of changes. It contains all the edits and historical versions (snapshots) of the project.

Each project consists of one project

There is no limit to create repositories in github

**Local Repository:**

It is stored in your local computer

**Remote repository:**

It is stored on a remote server

It is also called as Global repositories

**Cloning:** Extracting the files/code from remote repository to the local repository

Command: **git clone URL**

**Push:** we will send the files to the remote repository

**Pull:** we will bring or extract the files from remote to local Repository

03/03/2025:

Git has two types of protocols

1. HTTPs: Hypertext transfer protocol security

It is mainly used in banking sectors

1. SSH: Secure shell

It is mainly used in IT sectors

Real time environments:

1. Development
2. QA:
3. UAT:
4. Production

GIT Commands:

1. **git –version:** it is used to check the version of git.
2. **git config --global --list:** To display the identities if git already have the details of the user

If don’t have identities then execute the below to commands

1. **git config --global user.name ‘name’**
2. **git config --global user.email ‘email’**

Again run second command once done the above two commands

**Creating git folder**

**Mkdir dirname:** to create new directory

**Cd dirname:** to change the directory

1. **git init :** this command is used to initialize the empty repository

**Adding files to git:**

1. **git add <file\_name>:** is used to add file to the repository
2. **git add <file\_name1> <file\_name2>….<file\_namen> :** it is used to add multiple files to git repository
3. **git add -all:** It is used to add all files in the folder
4. **git commit -m ‘message’:** adding commits keep track of our progress and changes as we work

By adding clear messages to each commit, it is easy for yourself (and others) to see what has changed and when.

To know others about your changes or modifications

1. **git push -u origin master:**

* By default we are in **master branch**

1. **git rm -cached <file\_name>** : to remove the unwanted files
2. **git init -b main:** switch to the other branch
3. **git remote add origin <URL>:** specifies that you are adding a remote repository, with the specified url, as an origin to your local git repo
4. **git push -u origin master:** to publish local commits

**Creating SSH key:**

Step1: Open git bash

Run the below commands for generating SSH key

Step2: ssh-keygen -t ed25519 -c [vijayabhavani2807@gmail.com](mailto:vijayabhavani2807@gmail.com)

Step3: run the above command -🡪 press enter -🡪press enter -🡪 press y

Step4: run the below command

cat ~/.ssh/id\_ed25519.pub

step5: press enter

****

then it will generate one key 🡪 copy the key 🡪 open git hub 🡪 open settings 🡪 click on SSH and GRS keys 🡪 click on add SSH Key 🡪 enter title 🡪 select Authentication key 🡪 paste the key in key text box🡪 click SSH key



