**Git**

**Global information Tracker:**

* It is also called as version control system.
* It is central repository is using which we can manage our project source code.
* It maintains all modifications happening to a specified file.
* Because of versions troubleshooting & fixing bugs is easy
* If something goes wrong in current version we can roolback to previous.
* It is mainly used for collaborative work.
* Easy way to the fixing Bugs .
* Easy way to communicating.

**Version:**

The Git version refers to the specific release of the Git software you are using. Each version includes a unique set of features, bug fixes, and performance improvements. Git versions follow a semantic numbering system, typically in the format:

Format

MAJOR.MINOR.PATCH

* MAJOR: Significant updates that may introduce breaking changes.
* MINOR: New features and improvements, backward compatible.
* PATCH: Bug fixes and security updates.

Example

* 2.42.0:
  + 2: Major version.
  + 42: Minor version, introduces enhancements and new features.
  + 0: Patch version, focuses on bug fixes.
* It records the modifications when modified (Time stamp) and why it is modified.
* Git is distributed version control system.

**Used for two reasons:**

Git is a distributed version control system widely used for managing and tracking changes in source code or other files in a collaborative environment. It is a powerful tool for developers, designers, and project teams. Here's why Git is used**:**

**1. Version Control**

**Git helps track changes in files over time.**

* You can see a complete history of edits, additions, and deletions.
* Easily revert to previous versions if something goes wrong.

**2. Collaboration**

**Git enables multiple people to work on the same project simultaneously.**

* Teams can work on different branches of the same codebase.
* Merges changes seamlessly using advanced conflict resolution tools.

**3. Distributed System**

**Unlike centralized systems, Git doesn't require constant internet access.**

* Every team member has a complete copy of the project and its history.
* Changes can be made offline and synchronized later.

**4. Branching and Merging**

**Git's lightweight branching model is one of its strongest features.**

* Create independent branches to work on features or fixes.
* Merge branches back into the main project with ease.

**5. Backup and Security**

**Git acts as a backup of your project because each clone contains the entire project history.**

* You can recover from data loss quickly.
* Encryption and access control options ensure secure collaboration**.**

**6. Automation and Integration**

**Git integrates seamlessly with tools and platforms like GitHub, GitLab, and Bitbucket.**

* Automate deployments, testing, and builds.
* Use hooks to trigger scripts when certain actions occur (e.g., commits).

**7. Widely Adopted**

**Git is industry-standard and widely supported.**

* Tools, tutorials, and community support are abundant.
* Open-source projects often use Git for version control, making it essential for contributors**.**

**Functionalities of version control system:**

* Allows multiple developers to develop the code simultaneously.
* Does not allow over writing each other changes.
* Maintains a history of every version.

(v1)Reason1----v2(Reason1+Reason2)----V3(Reason1+Reason2+Reason3)----v100

* Git is fast moving version control system when compare to other.
* Multiple developers can easily collaborate & work on same project.
* It also works as backing up our project code.

**Types of VCS:**

**2 Types:**

1. **CVCS:** centralized version control system

----SVN-bub version control system.

**CVCS** stands for **Centralized Version Control System**. It is a type of version control system where the version history of files is stored on a single central server, and clients access this server to check out and commit changes.

**Centralized Repository**:

* All version control data is stored on a central server.
* Developers work on local copies and sync with the central server.
* **Examples**:
  + Subversion (SVN)
  + CVS (Concurrent Versions System)
  + Perforce
* **Advantages**:
  + Simplified administration since everything is centralized.
  + Easy to understand and manage for small teams.
  + Centralized backup ensures version history is preserved.
* **Disadvantages**:
  + Single point of failure: If the central server goes down, work halts for the entire team.
  + Limited offline capabilities: Users cannot commit or view full history without access to the server.

In contrast, **DVCS** (Distributed Version Control Systems) like Git and Mercurial store the full repository history on each client machine, offering more flexibility and resilience.

1. **DVCS:** Distributed VCS /Decentralized

VCS-------Git

**DVCS**, especially Git, is widely adopted in modern software development due to its flexibility and efficiency in collaborative environments.

**DVCS** stands for **Distributed Version Control System**. Unlike **CVCS**, where all version control data is stored on a central server, DVCS allows each developer to have a complete copy of the entire repository, including the full history of changes.

**Key Features of DVCS:**

1. **Distributed Repository**:
   * Every developer has a full copy of the repository, including all history.
   * Developers can work independently and commit changes locally without needing a connection to a central server.
2. **Examples**:
   * Git
   * Mercurial
   * Bazaar
3. **Advantages**:
   * **Offline Work**: Developers can commit changes, view history, and work without internet access.
   * **Resilience**: Since every developer has a complete copy, the system is not dependent on a single server.
   * **Speed**: Local operations (commits, diffs, branching) are faster because they don't rely on a central server.
   * **Branching and Merging**: These operations are easier and more powerful in DVCS compared to CVCS.
4. **Disadvantages**:
   * **Complexity**: DVCS can be harder to learn and manage for beginners due to advanced features like rebasing.
   * **Storage Overhead**: Since each user has a full copy of the repository, it can consume more disk space compared to CVCS.

**Comparison with CVCS:**

| **Feature** | **CVCS** | **DVCS** |
| --- | --- | --- |
| **Repository Location** | Central server | Fully distributed |
| **Offline Work** | Limited | Fully supported |
| **Performance** | Slower for some operations | Faster for local operations |
| **Single Point of Failure** | Yes, central server failure | No, each user has a full copy |

**DVCS**, especially Git, is widely adopted in modern software development due to its flexibility and efficiency in collaborative environments.

-----------Older version of git is **SVN((Subversion version control)**

It consists of current version data.V1--🡪V2 Each & Every developer needs to connect to this server& then needs to develop their codes.

**Disadvantages:**

1.Servers can be accessed by anyone.

2. If server is destroyed every thing will be lost.

**GIT hub**: It is a server

Git: version control system.

Decentralized /Distributed VCS: Github-🡪server each & every developer need not to connect to the github (server), Create a repository in github---remote repository.

In github we have an option of opting the code----cloning;

**Cloning**:

**using cloning, we can bring our codes from remote repository (server---Github) to local repository (own laptops/Systems).**

Cloning generally refers to creating an exact copy or duplicate of something. Its meaning can vary depending on the context:

**1. In Computing and Technology**

* Repository Cloning: In version control systems like Git or SVN, cloning means copying a remote repository to your local machine, including its entire history and files.
  + Example: git clone <repository-url> downloads all the project files and history from a remote Git repository.
* Disk Cloning: Duplicating the entire content of a hard drive or storage device to another.

**Command for cloning:**

Git clone URL

**Repository:**

Group pf project files to store in one single area. Each project has one repository. Github has many number of repositorys.

A **repository** (often abbreviated as "repo") is a central location where data, files, or code is stored and managed. Its specific meaning can vary based on context, but in most cases, it refers to a system used in version control to track changes to files over time.

**In Software Development (Version Control Systems):**

A repository is used to manage and store a project's source code, documentation, and other files, along with their history and revisions.

 **Local Repository**: Stored on a developer's computer. It is a private copy of the code and its history.

Getting the remote repository to or local repository

(our own laptops/systems)

 **Remote Repository**: Stored on a server or hosting service (e.g., GitHub, GitLab, Bitbucket). It is shared among team members for collaboration.

**Open the power shell:**

**Commands:**

* + 1. Git –version
    2. Git config
    3. Git config --global --list
    4. Git config --global user

**Git push commands:**

1. **git init**

**Purpose: Initializes a new Git repository.**

1. **2. git add <file\_name>**

**Purpose: Stages a file to be included in the next commit.**

1. **3. git commit -m "message"**

**Purpose: Commits staged changes to the local repository with a descriptive message.**

1. **4. git remote add origin <URL>**

**Purpose: Adds a remote repository to your local Git repository.**

1. **5. git push -u origin main**

**Purpose: Pushes your local commits to a remote repository and sets the upstream branch.**