SOFTWARE ENGINEERING

UNIT – 2

TOPIC – 6

INTRODUCTION TO VERSION CONTROL SYSTEM, GIT, GITHUB

Introduction to Version Control Systems (VCS)

In software development, managing and organizing code changes is essential. Projects can involve multiple developers who work on different parts of the code simultaneously. Without a way to track changes, things can quickly get confusing, and mistakes can easily happen. This is where Version Control Systems (VCS) come in.

Key Points:

- VCS tools help in **tracking, managing, and organizing code changes**.
- They **store different versions** of the code, so you can go back to previous versions if something goes wrong.
- VCS enables **multiple developers** to work on the same project, allowing easy **collaboration**.
- Example: Think of VCS like a "track changes" feature in a Word document, but for code, where you can see who made changes, when, and why.

What is a Version Control System (VCS)?

A Version Control System (VCS) is a tool that helps keep track of the changes made to files over time. It stores every version of the code, so you can revert to earlier versions if needed. It allows developers to work together without overwriting each other's changes.

Benefits of VCS:

1. **Keeps a history of changes:** Every modification is recorded, and you can see who made which change.

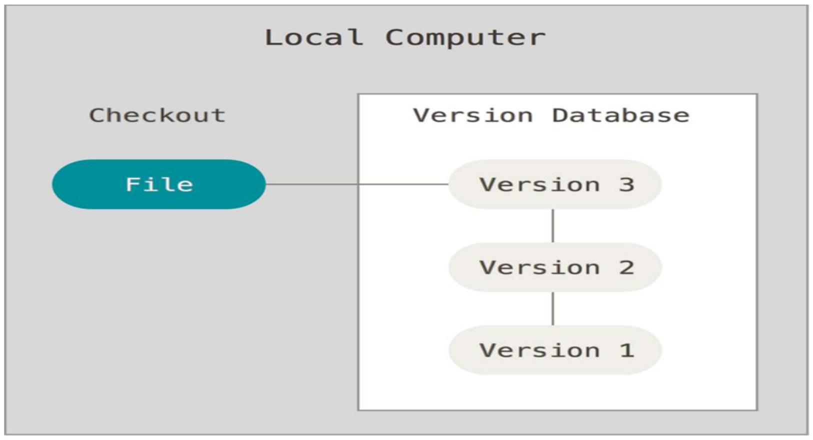
2. **Collaboration:** Multiple developers can work on the same project simultaneously without conflicts.
3. **Easy recovery:** If a mistake is made, you can go back to a previous version without losing any work.

Example: Imagine you're writing a report, and every time you save it, a backup copy is created. You can always go back to an earlier version if the new changes don't work out.

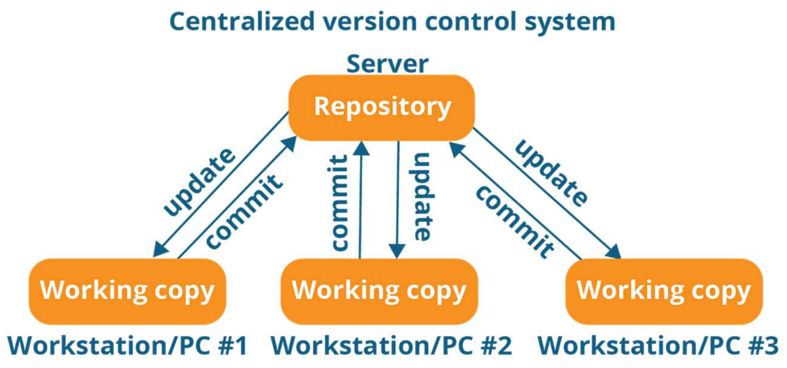
Types of Version Control Systems

There are three main types of VCS, each suited to different situations:

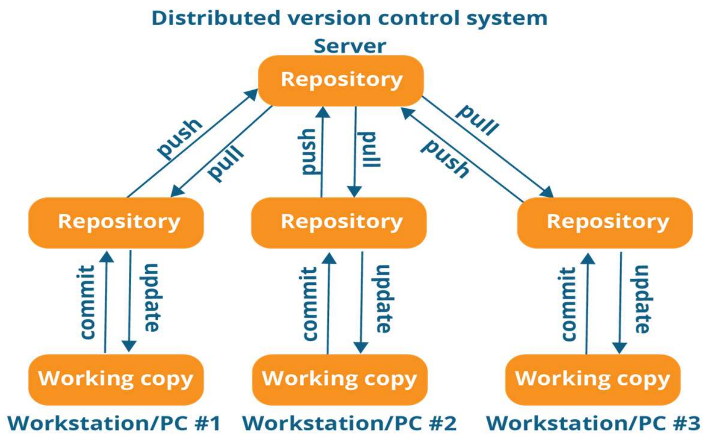
1. **Local Version Control System:**
   - Stores versions on your local computer.
   - **Problem:** If your computer crashes, you lose everything.
   - **Example:** Saving different versions of a file on your desktop.



2. **Centralized Version Control System:**
   - All versions are stored on a central server. Users connect to this server to access the files.
   - **Problem:** If the server crashes, you could lose all your work.
   - **Example:** Like a shared folder on Google Drive that everyone can edit.



3. **Distributed Version Control System (DVCS):**
   - Each user has a complete copy of the project, including its entire history.
   - **Advantage:** Even if the main server fails, copies exist on multiple devices, ensuring data safety.
   - **Example:** Think of it like everyone on the team having their own complete backup of a shared project folder.



Why Distributed VCS is Better:

- Provides **more safety** because data is stored in multiple places.
- Allows **better collaboration** since everyone has a complete copy, making it easy to work offline.

Introduction to Git

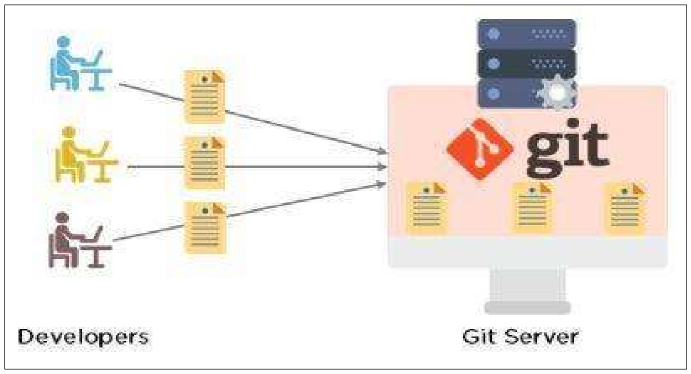
As projects grow larger and involve more people, managing changes can become difficult. This is where **Git**, a Distributed VCS, comes in.

What is Git?

- **Git** is a tool that tracks changes in files, especially code, over time.
- It was created by **Linus Torvalds** in **2005** to manage the development of the Linux kernel.
- Git helps multiple people work on the same project simultaneously without causing conflicts.

Features of Git:

1. **Free and open-source:** Anyone can use Git without cost.
2. **Supports collaboration:** Multiple developers can work on the same project at once.
3. **Non-linear development:** Git allows you to create and manage multiple branches of a project.
4. **Tracks entire history:** Every change is saved, so you can go back to any version.



Why Git is Popular:

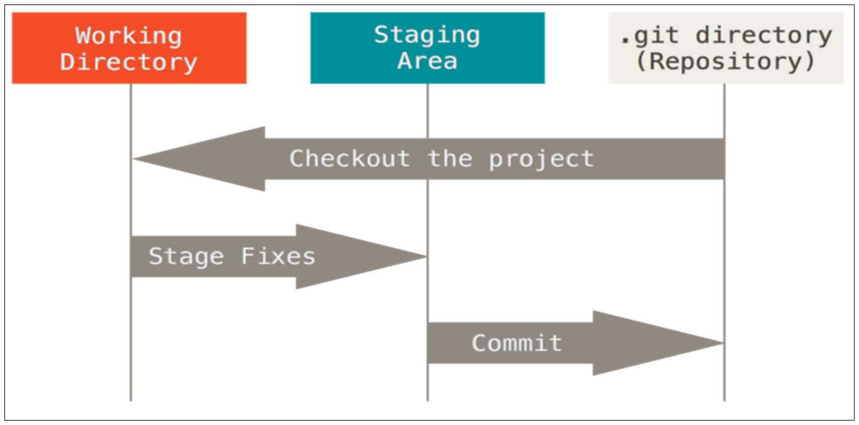
- Efficiently handles large projects with many contributors.
- Allows for **branching**, where developers can create separate copies of the code to work on features without affecting the main codebase.

How Git Works (Git Workflow)

To use Git effectively, it's important to understand its workflow, which involves three main stages:

1. **Working Directory:**
   - This is where you make changes to your files.
   - **Example:** Editing a file on your computer.

2. **Staging Area:**
   - This is where you prepare selected changes to be saved permanently.
   - **Example:** Think of it as placing files in a "to-do list" before adding them to your final project.
3. **Git Repository:**
   - This is where all your changes are stored, including the entire history of your project.
   - **Example:** A folder containing all your backups and old versions of the files.



Steps in Git Workflow:

1. **Edit files** in the Working Directory.
2. **Stage** the files you want to commit (save permanently) in the Staging Area.
3. **Commit** the changes to the Git Repository, where they are saved permanently.

Transition to GitHub: Why Do We Need It?

While Git is great for tracking changes, there's a need for **sharing and collaborating on code** across different locations. This is where **GitHub** comes in.

Problem with Just Git:

- If you're working on a project with people who are not in the same place, it's difficult to share changes and keep everything up-to-date.
- GitHub solves this by providing a **platform** where you can **store your Git projects online**, making it easier for teams to collaborate.

What is GitHub?

GitHub is a **web-based platform** that hosts Git repositories, making it easier for teams to **share, collaborate, and manage** their code projects.

Key Points:

1. **Definition:** GitHub is an online service where developers can store, manage, and share their Git projects.
2. **Purpose:** Allows teams to work on the same project from different locations, making collaboration easy.
3. **Owned by:** Microsoft since 2018.
4. **Example:** Imagine GitHub as a social network for programmers where they can share and discuss their code.

Features of GitHub

GitHub offers many features that make project management and collaboration more efficient:

1. **Project Management:** Helps teams track tasks, assign work, and organize projects.
   - **Example:** Like a to-do list that everyone can see and update.
2. **Secure Code Packages:** Safely store and share pieces of code with your team or publicly.
   - **Example:** Uploading a useful function that others can use.
3. **Team Collaboration:** Tools to communicate, review, and manage code changes.
   - **Example:** Commenting on a teammate's code to suggest improvements.
4. **Pull Requests (Code Review):** Discuss and review code changes before merging them into the main project.
   - **Example:** Like submitting a draft for approval before it's published.
5. **Security Tools:** Detect and fix weaknesses in your code.
   - **Example:** Checking for errors before releasing a product.
6. **Code Hosting:** Millions of repositories hosted, with easy access to project details and history.

Alternatives to GitHub

While GitHub is the most popular platform for hosting Git projects, there are other services that provide similar features:

1. **GitLab:** A complete platform for DevOps and collaboration, offering tools for project management and CI/CD.
2. **Bitbucket:** Often used by smaller teams; integrates well with Atlassian tools like Jira.
3. **AWS Code Commit:** Managed Git hosting provided by Amazon Web Services.
4. **SourceForge:** Known for hosting open-source software projects.

Differences Between Git and GitHub

While Git and GitHub are used together, they serve different purposes. Here's how they compare:

Aspect	Git	GitHub
Type	Software tool for tracking code changes	Online platform for hosting Git projects
Interface	Works through the command line	User-friendly graphical web interface
Location	Installed on your local computer	Accessible through a web browser
Purpose	Manages versions and history of code	Helps in sharing, collaborating, and managing projects
Ownership	Developed by Linus Torvalds	Owned by Microsoft
Focus	Version control and tracking code changes	Collaboration, project management, and code sharing
Example	A tool to save and track changes in your project files	A website where you can share your Git projects with others

Git: An essential tool that tracks changes, manages versions, and helps developers work on code without conflicts.

GitHub: A platform that allows teams to share and collaborate on Git projects from anywhere in the world.