ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DIPLOMA WALLAH

UNIT 02 :- VERSION CONTROL (GIT)

---

1. Why Do We Need a Version Control System (VCS)?

Definition:
A Version Control System (VCS) is software that helps track and manage changes to files over time. It keeps a history of who made which changes, allows reverting to earlier versions, and supports collaboration among multiple developers. VCS ensures code integrity, prevents accidental data loss, and simplifies teamwork.

Explanation:
In software development, multiple developers often work on the same project. Without VCS, tracking who made changes, resolving conflicts, and rolling back mistakes would be chaotic. VCS maintains a detailed history, enables branching for testing new features without affecting the main project, and simplifies code integration.

Example:
Three developers working on a website—one updates the homepage, another fixes the login form, and a third adds new CSS. VCS records each change, allowing safe merging and rollback if bugs appear.

Hinglish Summary:
VCS code ke purane versions ko track karta hai aur team ke saath safe collaboration allow karta hai.

Applications:

- Team collaboration in software development
- Code version history and tracking
- Bug fixing and rollback
- Feature experimentation
- Managing multiple project versions

Advantages:

- Tracks all file changes
- Enables collaboration
- Prevents accidental loss of work
- Supports branching and merging

- Simplifies deployment

Disadvantages:

- Learning curve for beginners
- Initial setup required
- Merge conflicts can occur
- Repository can become large
- Mismanagement can cause errors

---

2. Fundamentals of Git

Definition:
Git is a **distributed version control system** that tracks changes in source code and enables multiple developers to work collaboratively. Each developer has a local repository that contains the entire history of the project.

Explanation:
Git allows offline commits, local branches, and fast version tracking. It is reliable, efficient, and widely used in modern software development. Git maintains snapshots of project files, which can be restored, compared, or merged anytime. Its branching system allows experimenting with new features safely.

Example:
Developers can clone a repository, make changes in separate branches, and later merge into the main branch without overwriting others' work.

Hinglish Summary:
Git ek distributed system hai jo code ko track aur manage karne mein help karta hai aur team collaboration easy banata hai.

Applications:

- Software development collaboration
- Tracking project history
- Rolling back to stable versions
- Experimenting with new features
- Deployment and version management

Advantages:

- Distributed: works offline
- Fast performance

- Branching and merging support
- Reliable and secure
- Open-source and widely supported

Disadvantages:

- Initial learning curve
- Command-line interface can confuse beginners
- Merge conflicts can occur
- Requires regular commits for effectiveness
- Mismanagement can break workflow

---

3. Git Installation and Setup

Definition:
Installing Git sets up the software environment to track, commit, and manage code changes on your computer.

Explanation:
After installation, configuring Git ensures that commits are associated with your identity. Git works on Windows, macOS, and Linux. Correct installation and setup are the first steps before starting any Git workflow.

Installation Steps:

- **Windows:** Download from git-scm.com and install.
- **macOS:** git --version in Terminal, prompts installation if missing.
- **Linux:** sudo apt-get install git or sudo yum install git

Configuration Commands:

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

git --version

Example:
After setup, every commit shows your name and email, which is important in team projects for accountability.

Hinglish Summary:
Git install aur configure karne ke baad, aap commits track kar sakte hain aur identity set hoti hai.

4. Basic Local Git Operations

a. Creating a Repository

- **Command:** git init
- **Explanation:** Initializes a new repository in the current folder, creating a .git directory to track changes.
- **Example:**

mkdir MyProject

cd MyProject

git init

- **Hinglish Summary:** git init se folder mein Git repository start hoti hai.

b. Cloning a Repository

- **Command:** git clone <repository-url>
- **Explanation:** Copies an existing repository with all its history to your local machine.
- **Example:**

git clone https://github.com/user/MyProject.git

- **Hinglish Summary:** git clone se remote repository ka local copy banta hai.

c. Checking Status

- **Command:** git status
- **Explanation:** Shows which files are modified, staged, or untracked.
- **Hinglish Summary:** git status se file changes ka status pata chalta hai.

d. Staging Changes

- **Command:** git add <file-name>
- **Explanation:** Prepares modified files for commit.
- **Hinglish Summary:** git add se files commit ke liye ready hote hain.

e. Committing Changes

- **Command:** git commit -m "message"
- **Explanation:** Saves staged changes into the repository with a descriptive message.

- **Hinglish Summary:** git commit se changes repository mein save hote hain.

f. Viewing Commit History

- **Command:** git log
- **Explanation:** Shows all commits with ID, author, date, and message.
- **Hinglish Summary:** git log se purane commits dekhe ja sakte hain.

---

5. Git Branching and Merging

a. Creating a Branch

- **Command:** git branch <branch-name>
- **Explanation:** Creates a separate branch to develop new features without affecting the main branch.
- **Example:** git branch feature-login
- **Hinglish Summary:** Branch banakar alag feature develop kar sakte hain.

b. Switching Branches

- **Command:** git checkout <branch-name>
- **Explanation:** Switches the working directory to a different branch.
- **Example:** git checkout feature-login
- **Hinglish Summary:** Kisi bhi branch par switch karna easy hota hai.

c. Merging Branches

- **Command:** git merge <branch-name>
- **Explanation:** Integrates changes from one branch into another.
- **Example:** Merge feature-login into main.
- **Hinglish Summary:** Branch ke changes main branch mein aa jate hain.

d. Deleting a Branch

- **Command:** git branch -d <branch-name>
- **Explanation:** Deletes a branch after merging to keep the repository clean.
- **Example:** git branch -d feature-login
- **Hinglish Summary:** Unwanted branch delete karte hain.

---

6. Remote Repositories

Definition:
A remote repository is a version of your project hosted on the internet or network, enabling collaboration.

Explanation:
Developers push changes to and pull updates from a remote repository. Popular platforms include GitHub, GitLab, and Bitbucket.

Example:

- Push local commits: git push origin main
- Pull latest changes: git pull origin main

Hinglish Summary:
Remote repositories se aap changes share aur team ke updates le sakte hain.

---

7. Practical Workflow Example

1. Clone repository: git clone <url>
2. Create branch: git branch feature1
3. Switch branch: git checkout feature1
4. Make changes and stage: git add .
5. Commit changes: git commit -m "Added feature"
6. Merge branch: git checkout main && git merge feature1
7. Push to remote: git push origin main

Hinglish Summary:
Git workflow mein clone, branch, edit, commit, merge aur push steps follow karte hain.

Made with 💗 by Sagar Sangam