Version Control & Git Basics

1.1 Version Control: Definition and Purpose

Definition

Version Control is a system that tracks changes to files over time, enabling collaboration, history tracking, and easy rollback to previous versions.

Purpose of Version Control

✅ Maintains a history of changes.
✅ Enables collaboration among multiple developers.
✅ Helps in recovering previous versions in case of errors.
✅ Facilitates parallel development through branching.

Task: Explain the importance of version control in a software project.

✅ **Answer:**

- Version control helps developers track and manage changes in code.
- It allows teams to collaborate, work on different features simultaneously, and revert to previous versions if necessary.

1.2 Types of Version Control Systems

1. Centralized Version Control (CVCS)

✔ Uses a single central repository.

✔ Developers must be connected to the server to access the history.

✔ Examples: **Subversion (SVN), Perforce**.

2. Distributed Version Control (DVCS)

✔ Every developer has a complete copy of the repository.

✔ Enables offline work and fast operations.

✔ Examples: **Git, Mercurial**.

Comparison: CVCS vs. DVCS

Feature	Centralized (CVCS)	Distributed (DVCS)
Repository	Single server	Each user has a full copy
Offline Work	❌ No	✅ Yes
Performance	Slower	Faster
Example	SVN	Git

Task: Identify whether Git is a centralized or distributed system.

✅ **Answer:** Git is a **Distributed Version Control System (DVCS)** because each developer has a full copy of the repository, allowing offline work and independent version tracking.

1.3 Benefits of Version Control

✅ Maintains a history of all file changes.
✅ Allows multiple developers to collaborate.
✅ Provides a backup in case of data loss.
✅ Enables working on multiple features using branches.
✅ Prevents code conflicts with automated merging.

1.4 Introduction to Git and GitHub

1. What is Git?

Git is a distributed version control system that tracks changes in source code efficiently and enables collaboration.

2. What is GitHub?

GitHub is a remote repository hosting service that enables developers to store, share, and collaborate on Git projects.

Task: Explain the difference between Git and GitHub.

✅ **Answer:**

✔ **Git** is a version control system used to track changes in code.

✔ **GitHub** is an online platform for storing and sharing Git repositories.

1.5 Setting Up Git

Install Git

✔ **Windows**: Download from git-scm.com.

✔ **Mac**: Install via Homebrew:

bash

```bash
brew install git
```

✔ **Linux**:

bash

```bash
sudo apt install git
```

Configure Git

bash

```bash
git config --global user.name "Your Name"
git config --global user.email "your-email@example.com"
```

Task: Set up Git with your name and email.

✅ **Solution:**

bash

```
git config --global user.name "John Doe"
git config --global user.email "johndoe@example.com"
```

Basic Git Commands

2.1 Initializing and Cloning a Repository

✔ Initialize a Git Repository

bash

```
git init
```

📌 **Explanation:** Creates a new local repository.

✔ Clone an Existing Repository

bash

```
git clone https://github.com/user/repo.git
```

📌 **Explanation:** Copies a remote repository to the local machine.

Task: Initialize a Git repository and check its status.

✅ **Solution:**

bash

```bash
git init
git status
```

2.2 Tracking and Committing Changes

✔ Check Status of Files

bash

```bash
git status
```

📌 **Explanation:** Shows the status of modified, staged, and untracked files.

✔ Add Files to Staging Area

bash

```bash
git add file.txt
```

📌 **Explanation:** Moves `file.txt` to the staging area.

✔ Commit Changes

bash

```bash
git commit -m "Initial commit"
```

📌 **Explanation:** Saves the staged changes with a message.

Task: Stage and commit a file.

✅ **Solution:**

bash

```
git add index.html
git commit -m "Added index.html"
```

2.3 Viewing Commit History

✔ View Commit Log

bash

```
git log
```

📌 **Explanation:** Displays the commit history.

✔ View Changes in a File

bash

```
git diff
```

📌 **Explanation:** Shows differences between versions.

Task: Check the last 5 commits.

✅ **Solution:**

bash

```
git log --oneline -5
```

Branching, Merging & Collaboration

3.1 Branching in Git

✔ Create a New Branch

bash

```
git branch feature-branch
```

📌 **Explanation:** Creates a new branch called `feature-branch`.

✔ List All Branches

bash

```
git branch
```

📌 **Explanation:** Displays all branches.

✔ Switch Between Branches

bash

```bash
git checkout feature-branch
```

📌 **Explanation:** Switches to `feature-branch`.

Task: Create a branch named `dev` and switch to it.

✅ **Solution:**

bash

```bash
git branch dev
git checkout dev
```

3.2 Merging Branches

✔ Merge a Branch into Master

bash

```bash
git checkout master
git merge feature-branch
```

📌 **Explanation:** Merges `feature-branch` into `master`.

Task: Merge `dev` branch into `master`.

✅ **Solution:**

bash

```bash
git checkout master
git merge dev
```

3.3 Pushing and Pulling Changes

✔ Push Local Changes to Remote

bash

```bash
git push origin master
```

📌 **Explanation:** Uploads local commits to the remote repository.

✔ Pull Updates from Remote

bash

```bash
git pull origin master
```

📌 **Explanation:** Fetches and merges updates from the remote repository.

Task: Push changes to GitHub.

✅ **Solution:**

bash

```bash
git push origin master
```

3.4 Resolving Merge Conflicts

✔ **Identify Conflicts**

bash

```bash
git status
```

✔ **Open Conflict File**

- The file will show conflict markers (`<<<<<<<`, `=======`, `>>>>>>>`).
- Manually edit the file to resolve conflicts.

✔ **Mark Conflict as Resolved**

bash

```bash
git add conflicted-file.txt
git commit -m "Resolved merge conflict"
```

Task: Resolve a merge conflict.

✅ **Solution:**

1. Open the conflict file.
2. Edit and keep the correct version.
3. Stage and commit the file.

Learning Outcomes

✔ Understand **Version Control** and its importance.

✔ Learn **Git Basics**: Initializing, Cloning, Staging, and Committing.

✔ Use **Git Branching & Merging** for parallel development.

✔ Collaborate with **GitHub**, using push and pull commands.

✔ Handle **merge conflicts** effectively.

✅ **You have successfully learned the fundamentals of Git and Version Control!** 🎉