Chapter 1 - Git and GitLab – Mastering Version Control

1. Presentation of Git and GitLab

What is Git?

Git is a **distributed version control system (VCS)** that helps developers track changes in source code during software development. It allows multiple developers to collaborate efficiently while maintaining a history of changes.

Features of Git:

- Distributed architecture
- Version tracking
- Branching and merging
- Staging area
- Lightweight and fast

What is GitLab?

GitLab is a **web-based DevOps lifecycle tool** that provides Git repository management, CI/CD pipelines, issue tracking, and more. It is an alternative to GitHub and Bitbucket, offering both self-hosted and cloud-hosted solutions.

Features of GitLab:

- Repository hosting
- Integrated Continuous Integration/Continuous Deployment (CI/CD)
- Issue tracking and project management
- Code review and collaboration tools
- Security and compliance features

---

2. Basic Functioning of Git

Git Workflow Overview

Git operates in three main states:

1. **Working Directory** – Where modifications to files occur.
2. **Staging Area (Index)** – Where changes are prepared for commit.
3. **Repository (Local/Remote)** – Where committed changes are stored.

Common Git Commands

- `git init` – Initialize a new repository
- `git clone <repo_url>` – Clone an existing repository
- `git add <file>` – Stage a file for commit
- `git commit -m "message"` – Commit staged changes
- `git status` – Check the status of files
- `git log` – View commit history
- `git diff` – Compare changes between versions
- `git reset` – Undo changes

---

3. Repositories

What is a Repository?

A **Git repository** is a collection of files and their version history, used to track project changes.

Types of Repositories:

1. **Local Repository** – Stored on a developer's machine.
2. **Remote Repository** – Hosted on platforms like GitLab, GitHub, or Bitbucket.

Creating and Managing Repositories

- Create a new repository:

```
git init
```

- Clone an existing repository:

```
git clone <repo_url>
```

- View repository details:

```
git remote -v
```

---

4. Branches

What are Branches?

A **branch** is an independent line of development that allows developers to work on features without affecting the main project.

Common Branching Strategies:

- **Main (Master) Branch** – The default production-ready branch.
- **Feature Branch** – Created for new features.
- **Bugfix Branch** – Created for fixing issues.
- **Release Branch** – Prepares code for release.

Managing Branches

- Create a new branch:

```
git branch <branch_name>
```

- Switch to a branch:

```
git checkout <branch_name>
```

- List all branches:

```
git branch
```

- Delete a branch:

```
git branch -d <branch_name>
```

---

5. Merging and Rebasing

Merging

Merging is the process of integrating changes from one branch into another.

- Merge a branch into the current branch:

```
git merge <branch_name>
```

- View merge conflicts:

```
git status
```

Rebasing

Rebasing is an alternative to merging that applies changes from one branch onto another in a linear history.

- Start a rebase:

```
git rebase <branch_name>
```

- Resolve conflicts and continue:

```
git rebase --continue
```

---

6. GitHub

What is GitHub?

GitHub is a cloud-based Git repository hosting service that provides version control, collaboration tools, and integrations.

GitHub vs GitLab

Feature	GitHub	GitLab
Hosting	Cloud-based	Cloud & Self-hosted
CI/CD	External tools	Built-in CI/CD
Free Private Repos	Yes	Yes
Project Management	Basic	Advanced
Security Features	Limited	Advanced

Using Git with GitHub

- Push changes to GitHub:

```
git push origin <branch_name>
```

- Pull changes from GitHub:

```
git pull origin <branch_name>
```

- Create a Pull Request (PR) for review
- Collaborate with teams through Issues and Discussions