📘 Complete Git Curriculum & Lab Manual

Day 1 – Introduction & Basics

1. What is a Version Control System (VCS)?

- A VCS tracks file changes over time.
- Benefits: collaboration, rollback, history, and backup.

Types:

- **Centralized (CVCS):** One central server (SVN, CVS).
- **Distributed (DVCS):** Every developer has a full copy (Git, Mercurial).

---

2. What is Git?

- Created by **Linus Torvalds in 2005** for Linux kernel.
- Distributed, fast, secure, widely used.

Alternatives: SVN, Mercurial, Perforce.

3. Git Installation & Setup

Linux:

- sudo apt update
- sudo apt install git -y

Windows/Mac: Download from git-scm.com.

Configure:

- git config --global user.name "Your Name"
- git config --global user.email "your@email.com"
- git config --list

4. Key Terminology

- **Repository (Repo):** Folder tracked by Git.
- **Commit:** Snapshot of changes.
- **Branch:** Independent line of development.
- **Merge:** Combine branches.
- **Remote:** Central repo (GitHub).
- **Staging Area:** Place before committing.

5. Basic Git Workflow

Working Directory → Staging Area → Local Repo → Remote Repo

Lab 1: Create a Repository

- mkdir my-repo
- cd my-repo
- git init

✅ Output: `Initialized empty Git repository...`

Lab 2: Staging & Committing

- echo "Hello Git" > hello.txt
- git status
- git add hello.txt
- git commit -m "First commit"
- git log --oneline

Lab 3: GitHub Setup & Push

1. Create new repo on GitHub.

2. Connect remote:

- git remote add origin https://github.com/user/my-repo.git
- git branch -M main
- git push -u origin main

Lab 4: Clone a Repository

- git clone https://github.com/user/my-repo.git

Day 2 – Intermediate Git

6. Staging vs. Unstaging

- git add file.txt
- git reset file.txt

---

7. Git Stash

- echo "Temporary work" >> hello.txt
- git stash
- git stash list
- git stash apply

---

8. Branching

- **Why?** → Parallel development.

- git branch feature-1
- git switch feature-1
- echo "Feature work" > feature.txt
- git add feature.txt
- git commit -m "Added feature"
- git switch main
- git merge feature-1

Lab 5: Collaboration Workflow

1. Student A pushes changes.

2. Student B pulls changes:

- git pull origin main

Day 3 – Advanced Git

9. Merge Conflicts

- # In main
- echo "Main edit" > conflict.txt
- git add conflict.txt
- git commit -m "Main change"
-
- # In branch
- git switch -c conflict-branch
- echo "Branch edit" > conflict.txt
- git add conflict.txt
- git commit -m "Branch change"
-
- # Merge
- git switch main
- git merge conflict-branch

✅ Resolve manually → commit.

---

10. Rebase

- git switch feature-branch
- git rebase main

---

11. Tags

- git tag v1.0
- git push origin v1.0
- git tag -a v1.1 -m "Release v1.1"
- git push origin --tags

---

12. Git Ignore

`.gitignore` file:

- *.log
- node_modules/
- secrets.txt

- git add .gitignore
- git commit -m "Added gitignore"

---

13. Fork & Pull Requests

1. Fork repo on GitHub.

2. Clone fork:

- git clone https://github.com/student/forked-repo.git

3. Create branch → commit → push → Open **Pull Request**.

---

14. Reset & Revert

- git reset --soft HEAD~1 # undo commit, keep changes
- git reset --hard HEAD~1 # undo commit, discard changes
- git revert <commit-hash> # safely revert

---

15. Log & History

- git log --oneline --graph --all
- git blame hello.txt

---

16. Upstream Remote (Syncing Forks)

- git remote add upstream https://github.com/original/repo.git
- git fetch upstream
- git merge upstream/main

---

17. Cherry-pick

- git checkout main
- git cherry-pick <commit-hash>

---

18. Real-world Branching Strategy

GitFlow model:

- `main` → stable code.
- `develop` → integration.
- `feature/*` → new features.
- `release/*` → pre-release fixes.
- `hotfix/*` → urgent bug fixes.

✅ Outcome

By the end of this curriculum, students will be able to:

- Understand **VCS concepts**.
- Use **Git commands confidently**.
- Collaborate with **branches, PRs, merges, rebases**.
- Apply **real-world Git workflows** used in DevOps projects.

-