Introduction to Git & Workflow

✅ What is Git?

- Distributed Version Control System
- Tracks changes over time
- Enables collaboration

✅ What is a Git Workflow?

- A defined strategy for how developers use branches
- Ensures smooth teamwork and releases

✅ Why Workflows Matter?

- Prevent conflicts
- Organize features
- Control releases
- Enable CI/CD

✅ 2. Core Git Concepts (Foundation)

- Repository (local vs remote)
- Commit, Branch, Merge, Rebase
- Working directory, Staging area
- HEAD, origin, upstream

✅ 3. Basic Workflow (Solo Developer)

```
git init
git add .
git
```

✅ MODULE 1: Setup & Story

🎯 Goal:

Simulate a real software team building a feature using Git.

Scenario:

- Project: “Online Shopping App”
- Team Members:
    - Dev A – works on `login feature`
    - Dev B – works on `register feature`
    - Dev C – fixes bug
    - Reviewer
    - Release Manager

✅ MODULE 2: Workflow Overview (with diagram)

1. Clone repo
2. Create feature branch
3. Make changes
4. Commit
5. Push
6. Create Pull Request (PR)
7. Code review
8. Merge to `develop`
9. Test / Fix conflicts
10. Release to `main` branch
11. Tag version

✅ MODULE 3: Branch Strategy Explained

✅ main / master → Production

✅ develop → Integration / QA

✅ feature/* → New features

✅ bugfix/* → Small fixes

✅ hotfix/* → Urgent prod fix

✅ release/* → Pre-release testing

✅ MODULE 4: Simulation – Step by Step

🔹 Step 1: Project Setup (Trainer)

```
git init
git checkout -b main
git checkout -b develop
git push origin main
git push origin develop
```

🔹 Step 2: Developer Creates Feature Branch

Dev A:

```
git checkout develop
git checkout -b feature/login
```

Dev B:

```
git checkout develop
git checkout -b feature/register
```

🔹 Step 3: Coding & Committing

Dev A edits `login.js`

```
git add login.js
git commit -m "Add login page UI"
```

Dev B edits `register.js`

```
git add register.js
git commit -m "Add registration form"
```

🔹 Step 4: Push & Create Pull Request

```
git push origin feature/login
```

Go to GitHub → Open Pull Request → Assign Reviewer

🔹 Step 5: Code Review Simulation

Reviewer checks:
✅ Code style
✅ Naming
✅ Security checks
✅ Suggestions comments

PR comments example:

Please validate email format
Move logic to service layer

Dev fixes and pushes again:

```
git commit -am "Fix email validation"
git push
```

Reviewer → Approves → Merge

🔹 Step 6: Merge to Develop (By Maintainer)

```
git checkout develop
git merge feature/login --no-ff
```

(Delete branch after merge)

🔹 Step 7: Simulate Merge Conflict

Dev A and Dev B both edited same `routes.js`

Trainer creates conflict manually:

- Dev A merges first
- Dev B tries to merge → conflict

```
git merge feature/register
// CONFLICT!
```

Learners resolve:

```
git add routes.js
git commit -m "Resolve conflict"
```

🔹 Step 8: Release Preparation

Release Manager:

```
git checkout develop
git checkout -b release/v1.0
```

- Test features
- Bug fixes
- Documentation

Merge into main

```
git checkout main
git merge release/v1.0
```

Tag version

```
git tag -a v1.0 -m "Release 1.0"
git push origin v1.0
```

□ Step 9: Merge back to develop

```
git checkout develop
git merge main
```

✅ MODULE 5: Include Hotfix Simulation

Production bug found

```
git checkout main
git checkout -b hotfix/payment-error
```

Fix → Commit → Test → Merge → Tag → Merge back to develop

✅ MODULE 6: Advanced Concepts

✅ Rebase vs Merge (show difference)
✅ Squash commits
✅ Git stash workflow
✅ Git revert vs reset
✅ Cherry-pick hotfix to old release

✅ MODULE 7: Best Practices Discussion

✅ Small commits
✅ Meaningful messages
✅ Avoid long-lived branches
✅ Always pull before push
✅ Protect main branch
✅ Enable CI/CD on PR merges
✅ Use CODEOWNERS & Reviews

✅ MODULE 8: Real-time Group Exercise

Each participant:

- Fork repo
- Create own feature
- Raise PR
- Review someone else’s PR
- Resolve conflicts
- Merge to develop

✅ MODULE 9: Common Mistakes Simulation

- Forgot to pull
- Pushed to wrong branch
- Merged without review
- Messed up history with reset
- Overwritten others' code

Trainer shows **how to fix each**.

✅ MODULE 10: Recap & Quiz

✅ What is the purpose of develop branch?
✅ When do we use hotfix?

✅ What happens if two people push to same file?
✅ Merge vs Rebase difference?
✅ Why use tags?

✅ MODULE 11: Final Project (End-to-End Simulation)

Build new feature + bugfix + release with team roles.

Learners must:
✅ Plan branch strategy
✅ Create branches
✅ Submit PR
✅ Review code
✅ Solve conflict
✅ Release tagged version
✅ Show activity log

✅ MODULE 12: Bonus: Real Company Workflows

- GitHub Flow (used in startups)
- GitLab Flow
- Trunk-Based Development (used at Google)
- Feature toggles for CI/CD
