Course: CI/CD & Git Best Practices

Introduction to CI/CD

CI/CD stands for **Continuous Integration** and **Continuous Delivery/Deployment**. These are essential practices to automate and secure software development.

- **Continuous Integration (CI)**: regularly merging code changes into a shared branch, with automatic testing.
- **Continuous Delivery/Deployment (CD)**: automating the delivery of validated code to a test environment, then to production.

Why Use CI/CD?

- **Quick error detection** via automated tests.
- **More frequent and reliable delivery** of software.
- **Reduced deployment stress** (fewer manual errors).
- **Improved collaboration** between developers.

Typical CI/CD Pipeline

1. **Push** code to a Git repository
2. **Automatic build** (compilation, packaging)
3. **Automated tests** (unit, integration)
4. **Deployment** to a staging/test environment
5. **Deployment** to production after approval

Git Best Practices

1. Use Dedicated Branches

- `main` or `master`: stable branch deployed to production
- `develop` or `test`: integration branch to test new features
- Feature branches (`feature/xyz`): one branch per new feature or fix, created from `develop`
- Hotfix branches (`hotfix/xyz`): for urgent production bug fixes

2. Work with Pull Requests (or Merge Requests)

- Enable code review before merging
- Encourage discussion and code quality
- Often include automatic validations (tests)

3. Clear and Atomic Commits

- One commit = one logical change
- Explicit, standardized commit messages (e.g., `[FEATURE] Add contact page`)

4. Integrate Tests into the Workflow

- Run tests locally before pushing
- Use CI to automatically validate branches

5. Never Push Directly to `main`

- Always go through a PR from a `feature` branch to `develop`, then from `develop` to `main`

Git + CI/CD Workflow Example

```
main (production)
   ^
   |
develop (integration)
  ^        ^
  |        |
feature/x  feature/y (new features)
```

Common CI/CD Tools

- **GitHub Actions**
- **GitLab CI/CD**
- **Jenkins**
- **CircleCI**
- **Travis CI**

CI/CD Configuration Best Practices

- Automate all possible tests
- Keep pipelines fast (run heavy tests separately)
- Enforce merge rules (e.g., mandatory tests, code review)
- Monitor and analyze build failures to respond quickly

Summary

Concept	Key Role
CI	Integration and automated testing on every push
CD	Fast and automatic delivery to test and production
Git main	Stable production branch
Git develop/test	Branch for integrating new features
Feature branch	Isolated development of new functionality
Pull Request	Code review and validation before merging