Session 09: Deploy to GitHub via Git

Pre-Lab:

1. Why do developers use GitHub to store code?

Developers use GitHub to:

- Store source code securely
- Track code changes
- Collaborate with teams
- Manage versions and deployments

2. Which command is used to initialize a Git repository in your project?

git init

3. What file is used to ignore unnecessary files when pushing to GitHub?

.gitignore

In-Lab Tasks

1) Use Git commands to set up your GitHub account, install Git, and push all files from a local repository to GitHub

Step 1: Create GitHub Account

1. Open GitHub website
2. Sign up using email ID
3. Verify email and log in

Step 2: Install Git

- Download Git for Windows
- Complete installation using default options

Verify Installation

git –version

Step 3: Configure Git (One-Time Setup)

git config --global user.name "Your Name"

git config --global user.email your@email.com

Step 4: Create Local Project Directory

mkdir cloud-native-app

cd cloud-native-app

Step 5: Initialize Git Repository

git init

This creates a .git directory to track file changes.

Step 6: Create Sample Project File

echo "Cloud Native Application - Session 9" > README.md

Step 7: Add Files to Staging Area

git add .

Step 8: Commit Files

git commit -m "Initial commit for cloud-native app"

Step 9: Create Repository on GitHub

1. Click **New Repository**
2. Repository name: cloud-native-app
3. Select **Public**
4. Click **Create Repository**

Step 10: Link Local Repo to GitHub

git remote add origin https://github.com/username/cloud-native-app.git

Step 11: Push Files to GitHub

git branch -M main

git push -u origin main

Output (Screenshot to be added)

- All local files pushed successfully
- Repository visible on GitHub

2. Create a sample Git workflow (branching, committing, opening PRs) for cloud-native apps.

Step 12: Create a Feature Branch

git checkout -b feature-docker-support

A separate branch is created for new features without affecting the main branch.

Step 13: Modify Project File

echo "Added Docker support for cloud-native app" >> README.md

Step 14: Stage and Commit Changes

git add README.md

git commit -m "Added Docker support documentation"

Step 15: Push Feature Branch to GitHub

git push origin feature-docker-support

Step 16: Open Pull Request (PR)

1. Open GitHub repository
2. Click **Compare & Pull Request**
3. Add title and description
4. Click **Create Pull Request**

Step 17: Merge Pull Request

- Review changes
- Click **Merge Pull Request**
- Delete feature branch (optional)

Output (Screenshot to be added)

- Feature branch created
- Pull request opened and merged
- Main branch updated safely

Post-Lab

1.Illustrate the process of opening pull requests (PRs) in the context of developing and deploying cloud-native apps.

Git enables developers to work on features independently using branches. GitHub provides **Pull Requests (PRs)** as a mechanism to review, discuss, and safely merge code into the main branch.

In **cloud-native applications**, PRs are critical because:

- Multiple developers work in parallel
- Code must be reviewed before deployment
- CI/CD pipelines are usually triggered by PRs

Step-1: Create Feature Branch

git checkout -b feature-containerization

A separate branch is created to develop a new cloud-native feature (e.g., Docker support).

Step-2: Make Code Changes

Example:

echo "Added Dockerfile and Kubernetes manifests" >> README.md

Step-3: Stage and Commit Changes

git add .

git commit -m "Added containerization support"

Step-4: Push Feature Branch to GitHub

git push origin feature-containerization

The branch is now available on GitHub for collaboration.

Step-5: Open Pull Request on GitHub

1. Open the GitHub repository
2. Click **Compare & Pull Request**
3. Select:
    - **Base branch:** main
    - **Compare branch:** feature-containerization

4. Add PR title and description
5. Click **Create Pull Request**

Step-6: Review Pull Request

- Team members review code changes
- Suggestions or comments may be added
- CI pipeline (build/test) may run automatically

Step-7: Merge Pull Request

1. After approval, click **Merge Pull Request**
2. Confirm merge
3. Delete feature branch (optional but recommended)

Output (Screenshot to be added)

- Feature branch successfully merged into main
- Cloud-native app updated safely
- Deployment pipeline can be triggered