Placement Empowerment Program

Cloud Computing and DevOps Centre

Set Up Git Branching

Create a new branch in your Git repository for testing. Add a new feature and merge it.

Name: Hariharan v

Department: ADS

St. JOSEPH'S COLLEGE OF ENGINEERING | St. JOSEPH'S INSTITUTE OF TECHNOLOGY

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction

Git is an essential tool for collaborative and version-controlled development. One of its most powerful features is branching, which allows developers to work on new features, bug fixes, or experiments independently of the main codebase. Proper use of Git branching ensures smoother collaboration and reduces the risk of conflicts during development. This document provides a step-by-step guide to creating a new branch, adding a feature, and merging it into the main branch.

Objective

The objective of this task is to:

- Understand Git branching and merging concepts.
- Create a new branch in a Git repository for testing or feature development.
- Add a new feature to the branch.
- Merge the changes back into the main branch after testing.

Step 1: Understanding the Requirements

Key Concepts:

- **Branching:** Create a separate branch for new features or testing to avoid disrupting the main codebase.
- **Merging:** Integrate changes from a feature branch back into the main branch after testing.

Tools/Commands:

- git branch - Create a new branch.
- git checkout - Switch to a different branch.
- git merge - Merge one branch into another.

Step 2: Instructions

1. Create a New Branch

Use the following commands to create and switch to a new branch:

git branch feature-branch

git checkout feature-branch

Alternatively, you can combine the two steps:

git checkout -b feature-branch

Replace feature-branch with a name relevant to your new feature.

---

2. Add a New Feature

- Make changes to your code in the feature branch.
- Stage and commit your changes:
- git add .

git commit -m "Added new feature to feature-branch"

---

3. Test the Changes

- Run the necessary tests to ensure the new feature works as expected.

---

4. Merge Changes Into the Main Branch

- Switch back to the main branch:

git checkout main

- Merge the feature branch into the main branch:

git merge feature-branch

- Resolve any merge conflicts if they occur.

---

5. Clean Up

- Optionally, delete the feature branch after merging:

git branch -d feature-branch

Step 3: Best Practices

1. Always test your changes before merging.
2. Use descriptive branch names to easily identify the purpose of each branch.
3. Regularly pull updates from the main branch to your feature branch to avoid conflicts.

Conclusion

By using Git branching effectively, you can streamline your development workflow, minimize conflicts, and work collaboratively with your team. This task demonstrates how to create and merge branches, an essential skill for modern software development.