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: ANANDA KRISHNAN S A

DEPT: INFORMATION TECHNOLOGY

INTRODUCTION

Git branching is an essential feature that enables developers to work on new features, bug fixes, or experiments without affecting the main codebase. By creating separate branches, teams can collaborate efficiently, test new functionality, and merge stable changes into the main branch. This workflow ensures better version control and minimizes conflicts in the development process. In this guide, we will walk through the process of setting up a new Git branch, adding a feature, and merging it into the main branch.

OBJECTIVES

- Understand the importance of Git branching in version control.
- Create a new branch for feature development or testing.
- Add and commit changes to the new branch.
- Merge the feature branch back into the main branch.
- Delete unnecessary branches to keep the repository clean.

STEP-BY-STEP OVERVIEW

Step 1: Clone Your Git Repository (If Not Cloned)

- If you haven't cloned your repository yet, run:

```
git clone <repository-url>
cd <repository-name>
```

- If you're already inside the repo, ensure you're on the latest version:

```
git pull origin main
```

Step 2: Create a New Branch

- To create and switch to a new branch:

```
git checkout -b feature-branch
```

- This creates a branch named `feature-branch` and switches to it.

Step 3: Add a New Feature

- Modify or add files as needed. For example, create a new Python file:

```
echo "print('New feature added!')" > new_feature.py
```

- Stage the changes:

```
git add new_feature.py
```

- Commit the changes:

```
git commit -m "Added new feature file"
```

Step 4: Push the Branch to Remote Repository

- Push your new branch to GitHub or another Git server:

```
git push origin feature-branch
```

Step 5: Merge the Feature Branch into Main

- Switch back to the main branch:

```
git checkout main
git pull origin main   # Ensure it's up to date
```

- Merge the feature branch:

```
git merge feature-branch
```

- If there are no conflicts, push the changes:

```
git push origin main
```

Step 6: Delete the Feature Branch (Optional)

- After merging, you can delete the feature branch:

```
git branch -d feature-branch
git push origin --delete feature-branch  # Remove from remote
```

Conclusion:

Git branching makes development more organized by allowing new features or fixes to be worked on separately. Once tested, changes can be merged back into the main branch without affecting the existing code. Proper branch management keeps the repository clean and ensures a smooth workflow.

THANK YOU!