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: Shana R S

Department : CSE

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

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction:

In this task, you will learn how to set up and work with Git branching. By creating a new branch for testing, making changes, and merging them into the main branch, you will gain hands-on experience in managing different features of your project while keeping your codebase organized. This task will help you understand how to use Git for version control and collaborate effectively in a software development workflow.

Why Is This Task Important?

- **Work on new features safely** without affecting the main project.
- **Collaborate with others** by keeping changes separate until they're ready.
- **Track changes and revert to previous versions** if needed.
- **Follow industry best practices** for managing and maintaining code.

Steps Followed

- **Initialized Git repository**: `git init`
- **Created a new file**: `echo "Hello, World!" > testfile.txt`
- **Staged the file**: `git add testfile.txt`
- **Committed the file**: `git commit -m "Initial commit with testfile.txt"`
- **Created a new branch**: `git branch testing-branch`
- **Switched to the new branch**: `git checkout testing-branch`
- **Added a new feature**: `echo "New feature" >> testfile.txt`
- **Staged and committed the changes**: `git add testfile.txt` and `git commit -m "Added a new feature"`
- **Switched back to the master branch**: `git checkout master`
- **Merged the changes from testing-branch**: `git merge testing-branch`
- **Verified the changes**: `git log --oneline`

Where Can This Be Applied?

- **Software Development**: For working on features or bug fixes without affecting the main project.
- **Team Collaboration**: Helps multiple people work on different parts of a project simultaneously.
- **Feature Development**: Test new features before merging them into the main code.
- **Bug Fixes**: Isolate bug fixes and test them independently.
- **Open-Source Projects**: Work on contributions safely without changing the main project.

Step-by-Step Overview

Step1: Initialize a Git Repository

☑ **Instruction:**

- You initialized a new Git repository:

```
git init
```

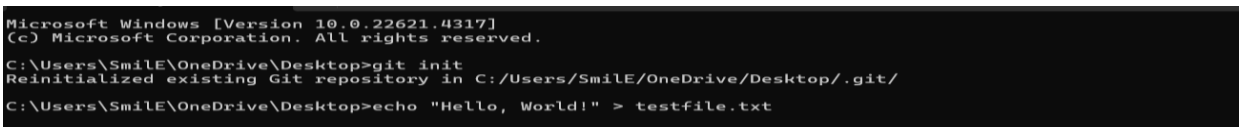
This creates a hidden .git folder, making the directory a Git repository for version control.

Step 2 : Create a New File

☑ **Instruction:**

- You created a new file named testfile.txt with some content:

```
echo "Hello, World!" > testfile.txt
```



This sets up an initial file for tracking changes in Git.

Step 3 : Add the File to Staging Area

☑ Instruction:

- You staged the file for commit:
  git add testfile.txt

```
C:\Users\SmilE\OneDrive\Desktop>git add testfile.txt
```

This tells Git to track testfile.txt and include it in the next commit.

Step 4 : Commit the File

☑ Instruction:

- You committed the file with a message:
  git commit -m "Initial commit with testfile.txt"

```
C:\Users\SmilE\OneDrive\Desktop>git commit -m "Initial commit with testfile.txt"
[master bae4c11] Initial commit with testfile.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
```

This saves a snapshot of your project with a descriptive message for tracking changes.

Step 5 : Create and Switch to a New Branch

☑ Instruction:

- You created a new branch named testing-branch:
  git branch testing-branch
- You switched to that branch:
  git checkout testing-branch

```
C:\Users\SmilE\OneDrive\Desktop\MyGittest>git branch testing-branch

C:\Users\SmilE\OneDrive\Desktop\MyGittest>git checkout testing-branch
Switched to branch 'testing-branch'
```

Creating a separate branch allows you to work on new features without affecting `master`.

Step 6 : Modify the File on testing-branch

☑ **Instruction:**

- You updated testfile.txt by adding a new feature:
  echo "This is a new feature added on the testing branch." >> testfile.txt

```
C:\Users\SmilE\OneDrive\Desktop\MyGittest>echo "This is a new feature added on the testing branch." >> testfile.txt
```

This simulates adding a new feature while working in a separate branch.

Step 7 : Stage and Commit the Changes

☑ **Instruction:**

- You staged the modified file:

  git add testfile.txt
- You committed the changes with a message:

  git commit -m "Added a new feature in testing-branch"

```
C:\Users\SmilE\OneDrive\Desktop\MyGittest>echo "This is a new feature added on the testing branch." >> testfile.txt
```

Committing records the changes, allowing you to track updates in the branch.

Step 8 : Switch Back to master Branch

☑ **Instruction:**

- You switched back to `master`:
  git checkout master

```
C:\Users\SmilE\OneDrive\Desktop\MyGittest>git checkout master
Switched to branch 'master'
```

Before merging, you need to be on the branch where you want to integrate the changes.

Step 9 : Merge Changes from testing-branch into master

☑ **Instruction:**

- You merged the changes into **`master`**:
  git merge testing-branch

```
C:\Users\SmilE\OneDrive\Desktop\MyGittest>git merge testing-branch
Updating c0b29c4..a8da8d3
Fast-forward
 testfile.txt | 1 +
 1 file changed, 1 insertion(+)
```

This integrates the new feature from testing-branch into the main codebase.

Step10: Verify the Merge

☑ **Instruction:**

- You checked the commit history:
  git log --oneline

```
C:\Users\SmilE\OneDrive\Desktop\MyGittest>git merge testing-branch
Updating c0b29c4..a8da8d3
Fast-forward
 testfile.txt | 1 +
 1 file changed, 1 insertion(+)

C:\Users\SmilE\OneDrive\Desktop\MyGittest>type testfile.txt
"Hello, World!"
"This is a new feature added on the testing branch."

C:\Users\SmilE\OneDrive\Desktop\MyGittest>git log --oneline
a8da8d3 (HEAD -> master, testing-branch) Added a new feature in testing-branch
c0b29c4 Initial commit with testfile.txt
```

This confirms that the merge was successful and shows the commit history

Outcome

By completing this POC, you will:

- Have version-controlled your project by using Git.
- Have created a new branch (testing-branch) for testing features without affecting the main project.
- Have added, committed, and merged a new feature into the master branch.
- Have learned key Git commands (git branch, git checkout, git add, git commit, git merge) for managing and tracking changes in your project.