Placement Empowerment Program

Cloud Computing and DevOps Centre

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

Name: CHANDRU S

Department: INFORMATION TECHNOLOGY

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

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction

This Proof of Concept (POC) demonstrates how Git facilitates version control and streamlines the development workflow. By creating separate branches for new features, developers can work independently without affecting the main branch. Once the feature is complete, it can be merged back seamlessly, ensuring a structured and collaborative development process.

Overview

This Proof of Concept (POC) covers the fundamental steps of using Git for version control, including:

1. Initializing a Git repository.
2. Creating and switching between branches.
3. Committing changes in different branches.
4. Merging feature branches into the main branch.
5. Deleting branches after completing development.

By following these steps, developers can efficiently manage their workflow and maintain a well-structured codebase.

Objectives

1. Initialize and configure a Git repository.
2. Create, switch, and manage feature branches (e.g., testing-feature).
3. Demonstrate the process of adding, committing, and merging changes.
4. Learn how to delete branches after their purpose is fulfilled.
5. Understand how to identify and resolve merge conflicts effectively.

Importance

- **Version Control**: Tracks changes, enables rollback to previous versions, and minimizes code conflicts.
- **Collaboration:** Allows multiple developers to work on different features simultaneously without interference.
- **Branching:** Isolates new features and bug fixes, ensuring the stability of the main branch.
- **Efficiency:** Merging branches facilitates seamless integration of new features without disrupting development.

- **Clean Workflow:** Removing feature branches after merging keeps the repository organized and manageable.

Step-by-Step Overview

Step 1:
Create a new folder and name it **Git_Branching**.

Step 2:
Set the path to the folder created in first step (**Git_Branching**).

```
C:\Users\chandru>cd C:\Users\chandru\OneDrive\Desktop\Git_Branching
```

Step 3:
- Open a terminal or command prompt and run:
  git init
- This initializes a Git repository by creating a .git folder inside **Git_Branching**.

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git init
Initialized empty Git repository in C:/Users/chandru/OneDrive/Desktop/Git_Branching/.git/
```

Step 4:
Add a simple file to start tracking changes.

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>echo "Initial file content" > first-file.txt
```

Step 5:
Add the file by running:
git add .

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git add .
```

Step 6:
Save the changes with a commit message:

git commit -m "Initial commit"

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git commit -m "Initial commit"
[master (root-commit) b6ca1ec] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 first-file.txt
```

Step 7:
Create and switch to a new branch named testing-feature:

git checkout -b testing-feature

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git checkout -b testing-feature
Switched to a new branch 'testing-feature'
```

Step 8:
Create another file (e.g., feature.txt) to work on the new branch.

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>echo "Initial file content" > feature.txt
```

Step 9:
Add the new file to the staging area:

git add .

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git add .
```

Step 10:

Save the changes in the testing-feature branch:

git commit -m "Add new feature file"

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git commit -m "Add new feature file"
[testing-feature 9daf7e6] Add new feature file
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
```

Step 11:

Move back to the main branch:

git checkout master

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git checkout master
Switched to branch 'master'
```

Step 12:

Merge the feature branch into the master branch:

git merge testing-feature

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git merge testing-feature
Updating b6ca1ec..9daf7e6
Fast-forward
 feature.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
```

Step 13:

Once merged, delete the testing-feature branch:

git branch -d testing-feature

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>git branch -d testing-feature
Deleted branch testing-feature (was 9daf7e6).
```

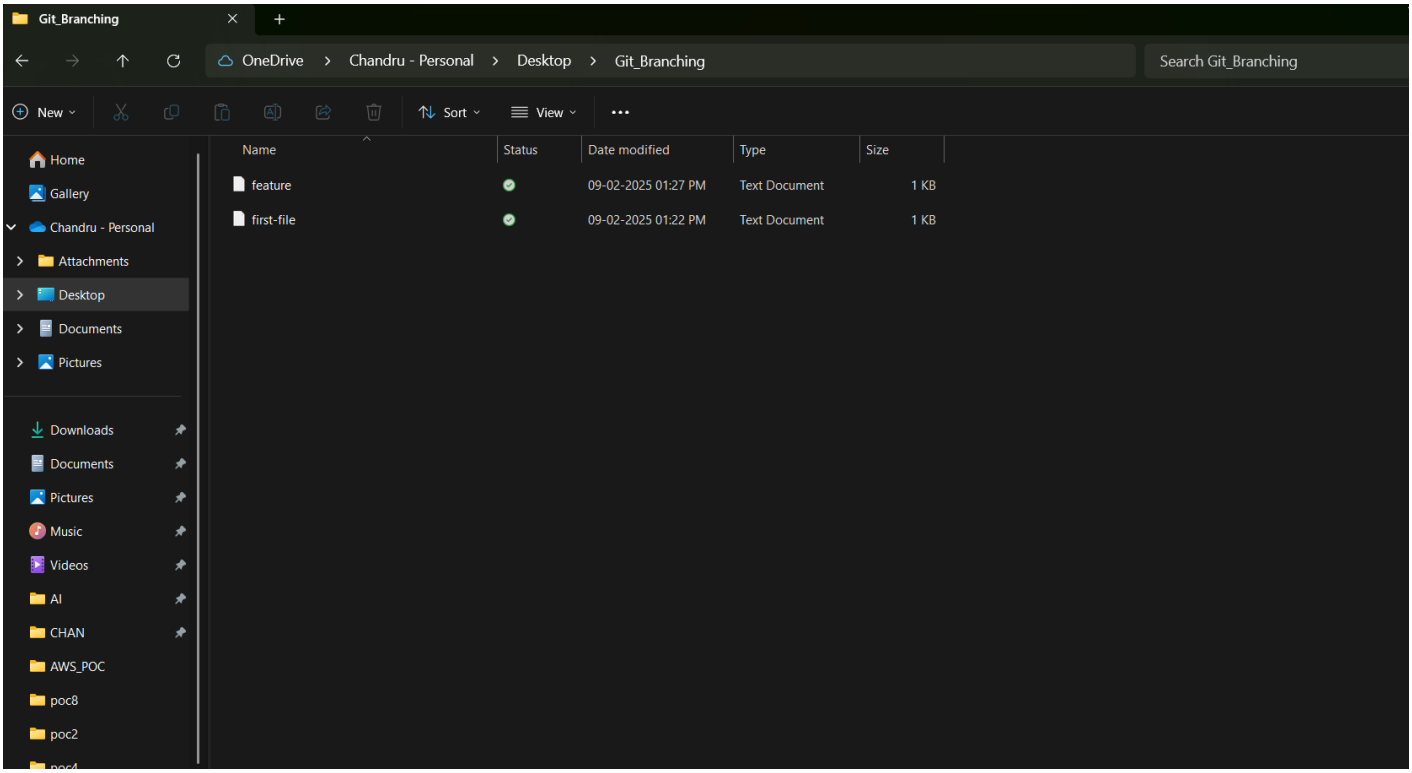
Step 14:

Now, check the files in the folder:

```
C:\Users\chandru\OneDrive\Desktop\Git_Branching>dir
 Volume in drive C is Windows-SSD
 Volume Serial Number is 7C35-5834

 Directory of C:\Users\chandru\OneDrive\Desktop\Git_Branching

09-02-2025  01.27 PM    <DIR>          .
09-02-2025  01.14 PM    <DIR>          ..
09-02-2025  01.27 PM                25 feature.txt
09-02-2025  01.22 PM                25 first-file.txt
               2 File(s)             50 bytes
               2 Dir(s)  320,754,892,800 bytes free
```



Outcome

By completing this PoC of managing branches in Git for a local repository, you will:

1. Successfully initialize a Git repository in your local project folder.
2. Create and manage multiple branches for feature development and experimentation.
3. Track and commit changes made to files in different branches.
4. Merge feature branches back into the main branch while maintaining project integrity.
5. Gain hands-on experience with key Git commands such as git init, git add, git commit, git checkout, and git merge.