



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: Harshad S

Department: CSE

Introduction:

In this Proof of Concept (POC), Git is used for version control to manage the development workflow. Git allows developers to create separate branches for new features, isolate them from the main branch, and merge them back after completion. This ensures organized and collaborative development.

Overview:

This POC demonstrates how to:

1. Initialize a Git repository.
2. Create and switch between branches.
3. Commit changes in different branches.
4. Merge feature branches into the main branch.
5. Delete branches after completing the work.

Objectives:

1. To initialize and set up a Git repository.
2. To create and manage feature branches (e.g., testing-feature).
3. To demonstrate adding, committing, and merging code.
4. To showcase how to delete branches after their purpose is served.

5. To learn how to resolve merge conflicts if any arise during the process.

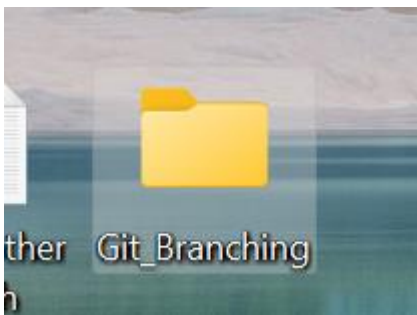
Importance:

- 1. Version Control:** Helps track changes, revert to previous versions, and avoid conflicts in the codebase.
- 2. Collaboration:** Different team members can work on separate features simultaneously without interfering with each other's work.
- 3. Branching:** Isolates new features or bug fixes, ensuring stability in the main branch (master or main).
- 4. Efficiency:** Merging branches allows rapid integration of new features without disrupting ongoing work.
- 5. Clean Workflow:** Deleting feature branches after merging keeps the repository clean and manageable.

Step-by-Step Overview Step

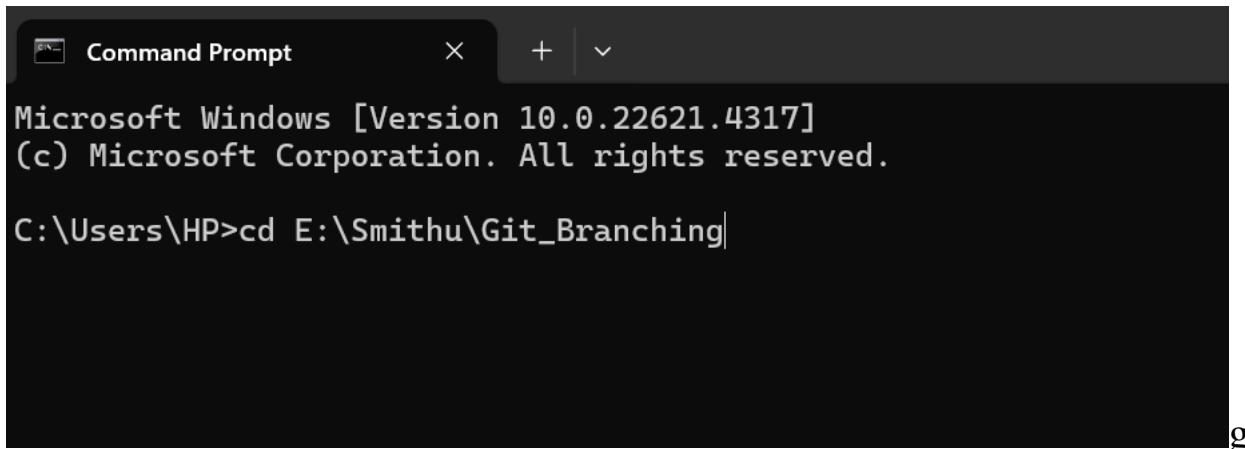
1:

Create a folder and name it (Git_Branching).



Step 2:

Set the path to the folder created in first step (Git_Branching).

A screenshot of a Windows Command Prompt window. The title bar says "Command Prompt". The text inside shows the Windows version and copyright information, followed by the command to change the directory to E:\Smithu\Git_Branching.

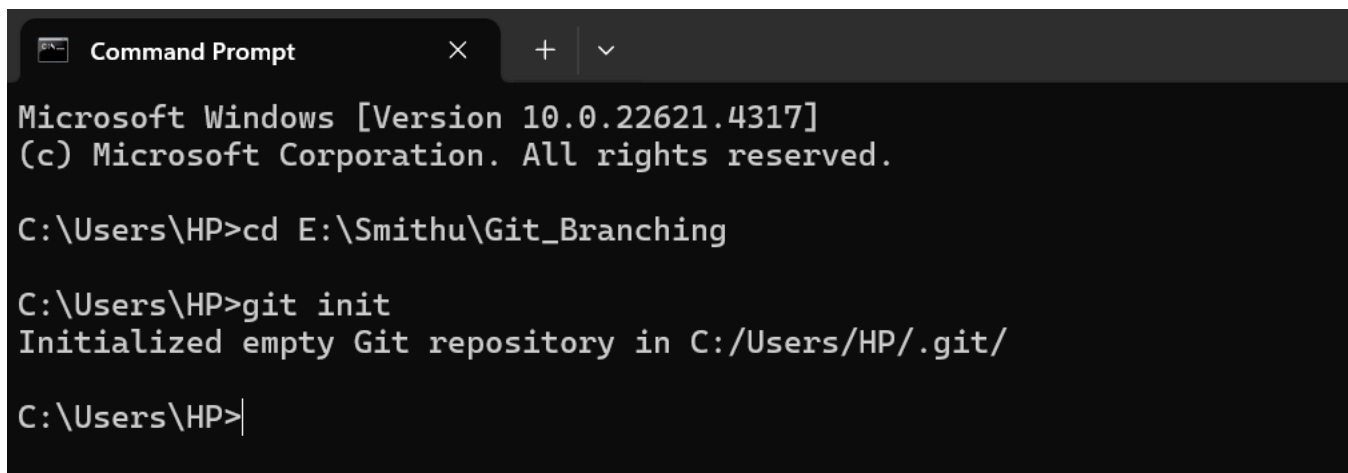
```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd E:\Smithu\Git_Branching|
```

Step 3:

Initialize Git by typing this command:

git init

A screenshot of a Windows Command Prompt window showing the execution of the 'git init' command. The output indicates that an empty Git repository has been initialized in the current directory.

```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>cd E:\Smithu\Git_Branching

C:\Users\HP>git init
Initialized empty Git repository in C:/Users/HP/.git/

C:\Users\HP>|
```

This command will create a .git folder inside your folder, which tells Git to start tracking your files.

Step 4:

```
Command Prompt
C:\Users\HP>cd E:\Smithu\Git_Branching

C:\Users\HP>git init
Initialized empty Git repository in C:/Users/HP/.git/

C:\Users\HP>echo "Initial file"
"Initial file"

C:\Users\HP>|
```

Create a

simple file to start the repository:

Step 5:

Add the File to Git

Tell Git to track this file:

```
Command Prompt
Initialized empty Git repository in C:/Users/HP/.git/

C:\Users\HP>echo "Initial file"
"Initial file"

C:\Users\HP>E:\Smithu\Git_Branching\.git
'E:\Smithu\Git_Branching\.git' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\HP>git add .|
```

Step 6:

Save this change in Git with a commit message.

```
[master (root-commit) 9595e67] Initial commit
1 file changed, 1 insertion(+)
create mode 100644 first-file.txt
```

Step 7:

```
C:\Users\HP> cd E:\Smithu\Git_Branching
C:\Users\HP>git checkout|
```

Create and switch to a new branch called testing-feature.

Step 8:

```
C:\Users\HP>cd E:\Smithu\Git_Branching
C:\Users\HP>echo "This is a new feature" > new-feature.txt
C:\Users\HP>|
```

Let's add a new file for our feature:

Step 9:

```
fatal: adding files failed
C:\Users\HP>git add.|
```

Now, stage the changes:

Step 10:

Commit the changes:

```
Updating 1150f3c..f4bd75d
Fast-forward
 new-feature.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 new-feature.txt
```

Step 11:

```
Switched to branch 'master'
```

Switch to the master Branch

Step 12:

```
Updating 1150f3c..f4bd75d
Fast-forward
 new-feature.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 new-feature.txt
```

Merge Changes from testing-feature to master

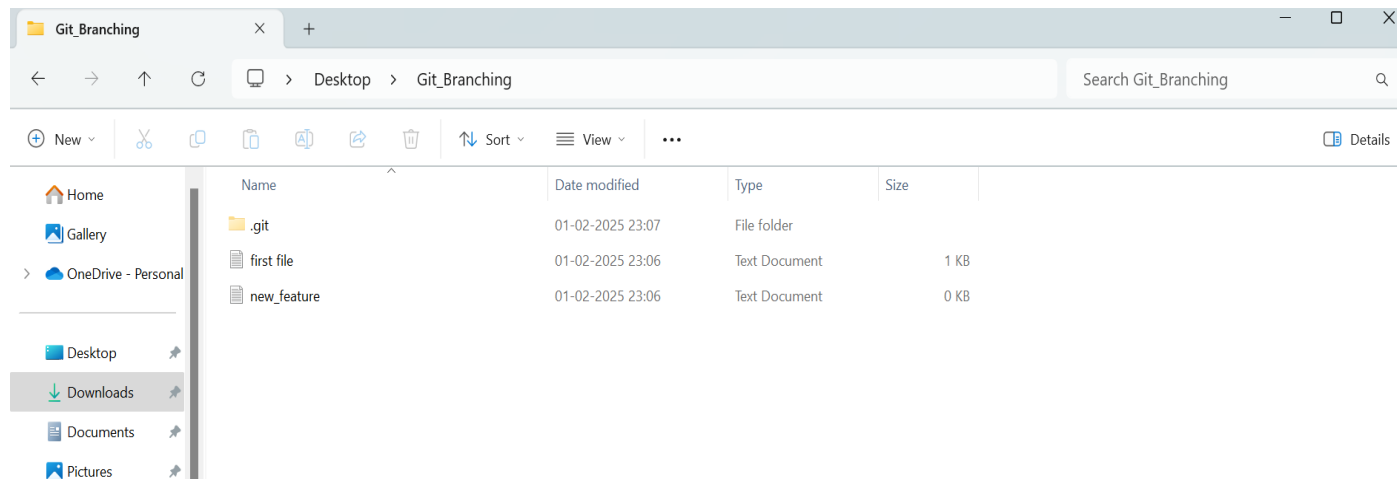
Step 13:

Once the merge is done, you can delete the testing-feature branch.

Step 14:

Now, check the files in the folder:

```
29-01-2025 23:30 <DIR> .
29-01-2025 23:05 <DIR> ..
29-01-2025 23:26 25 first-file.txt
29-01-2025 23:30 25 new-feature.txt
                2 File(s)          50 bytes
                2 Dir(s) 103,125,147,648 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`.