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: Akshaya 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).

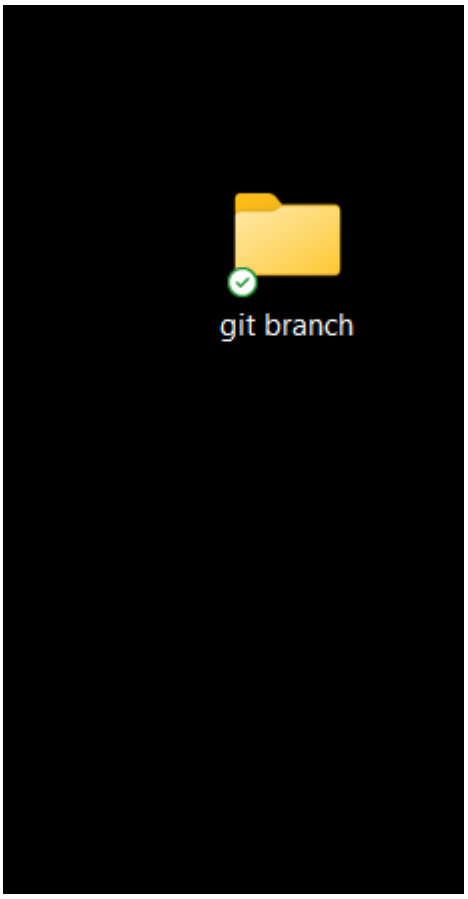
4. Efficiency: Merging branches allows rapid integration of new features without disrupting ongoing work.

5. Clean Workflow: Deleting feature branches after merging keeps the repository clean and manageable.

Step-by-Step Overview Step

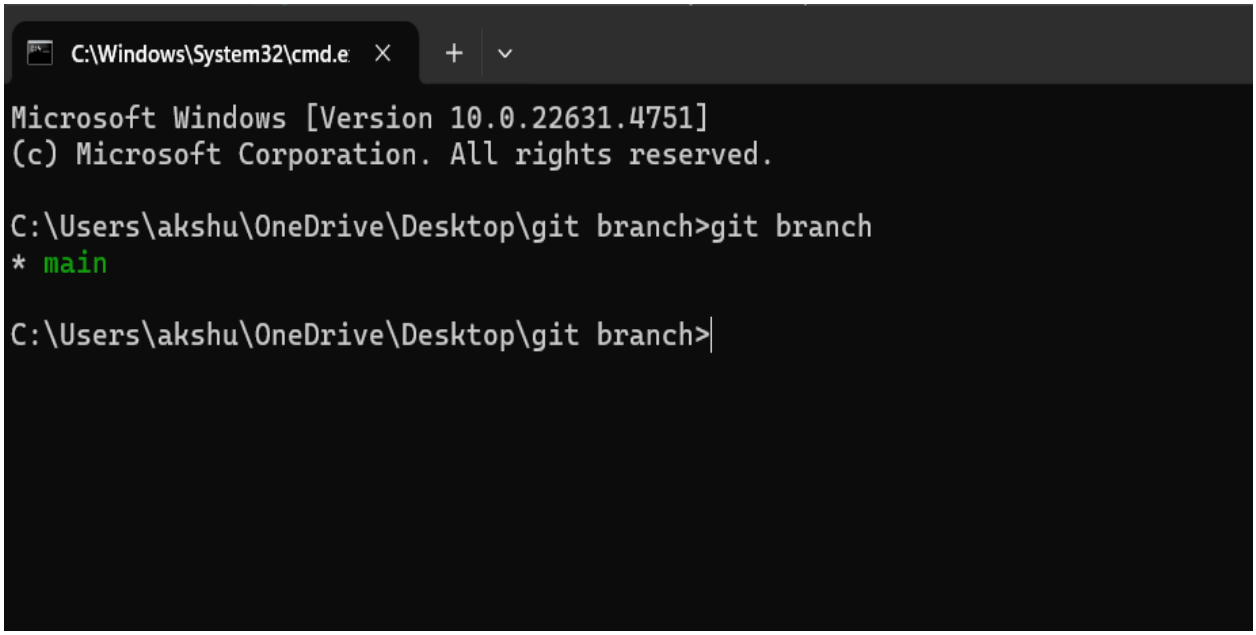
1:

Create a folder and name it (Git_Branching).



Step 2:

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

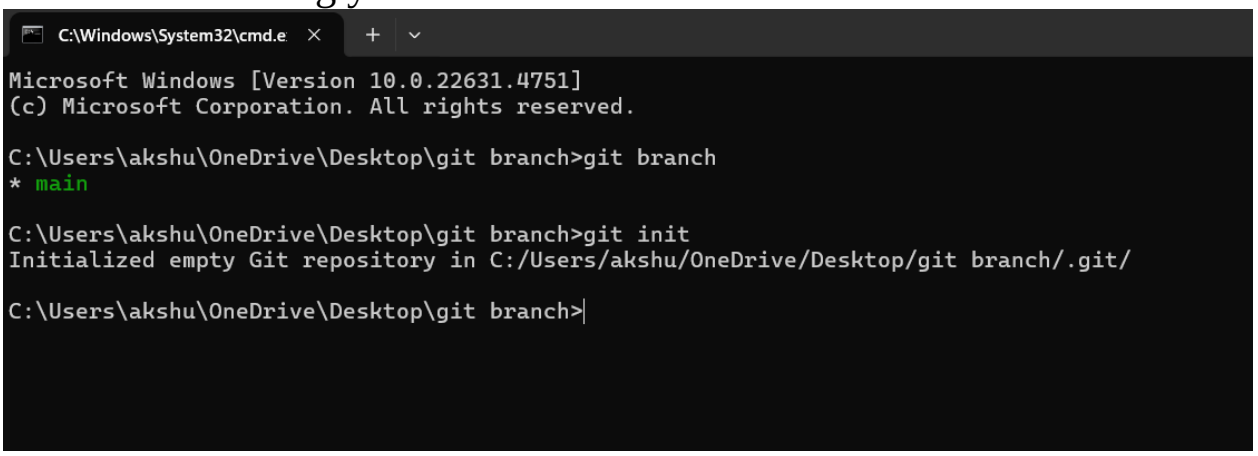


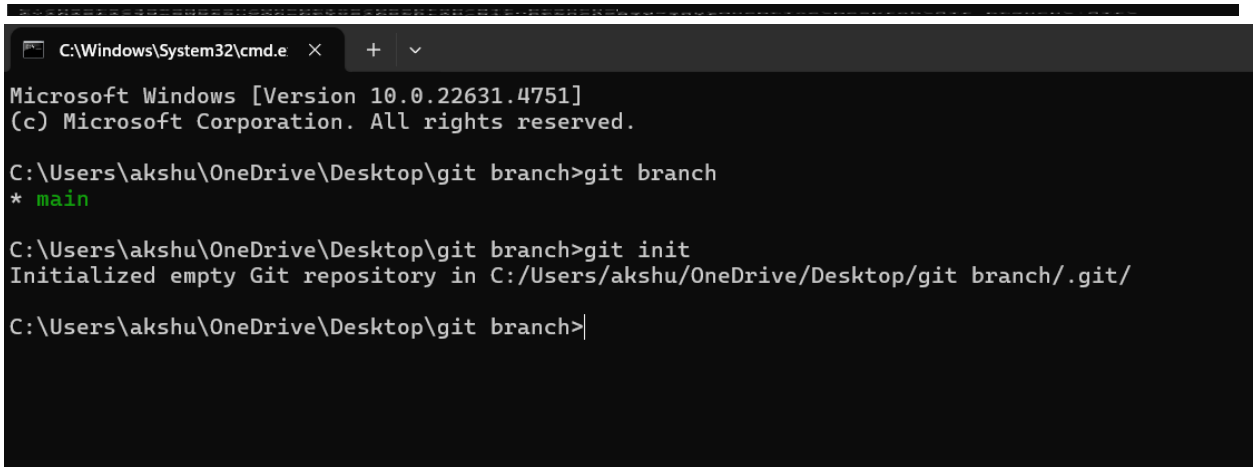
Step 3:

Initialize Git by typing this command:

git init

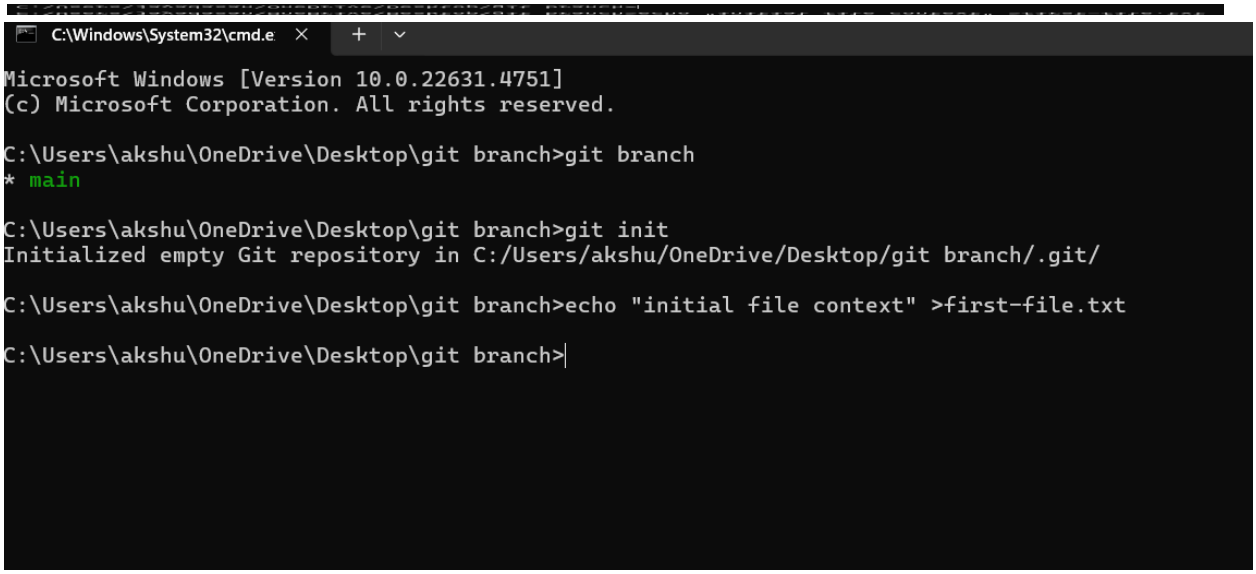
This command will create a .git folder inside your folder, which tells Git to start tracking your files.





Step 4:

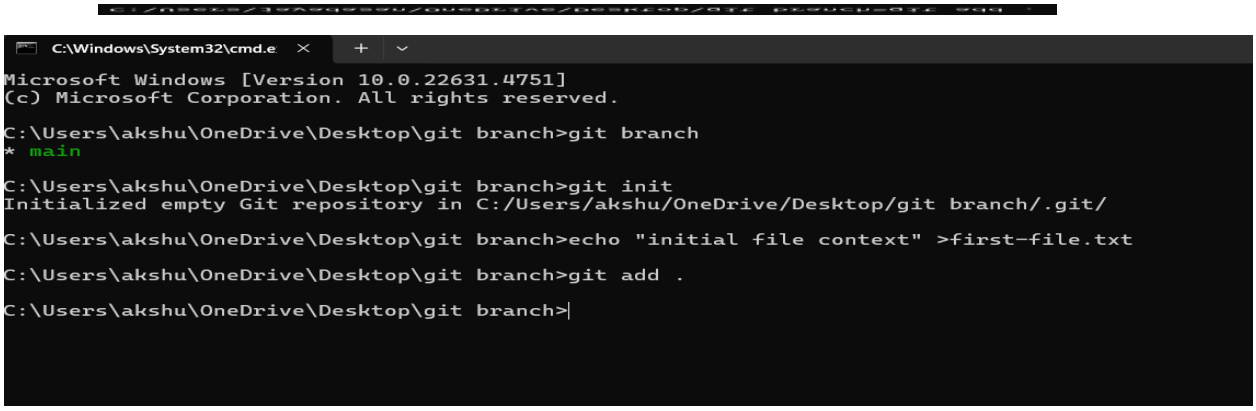
Create a simple file to start the repository:



Step 5:

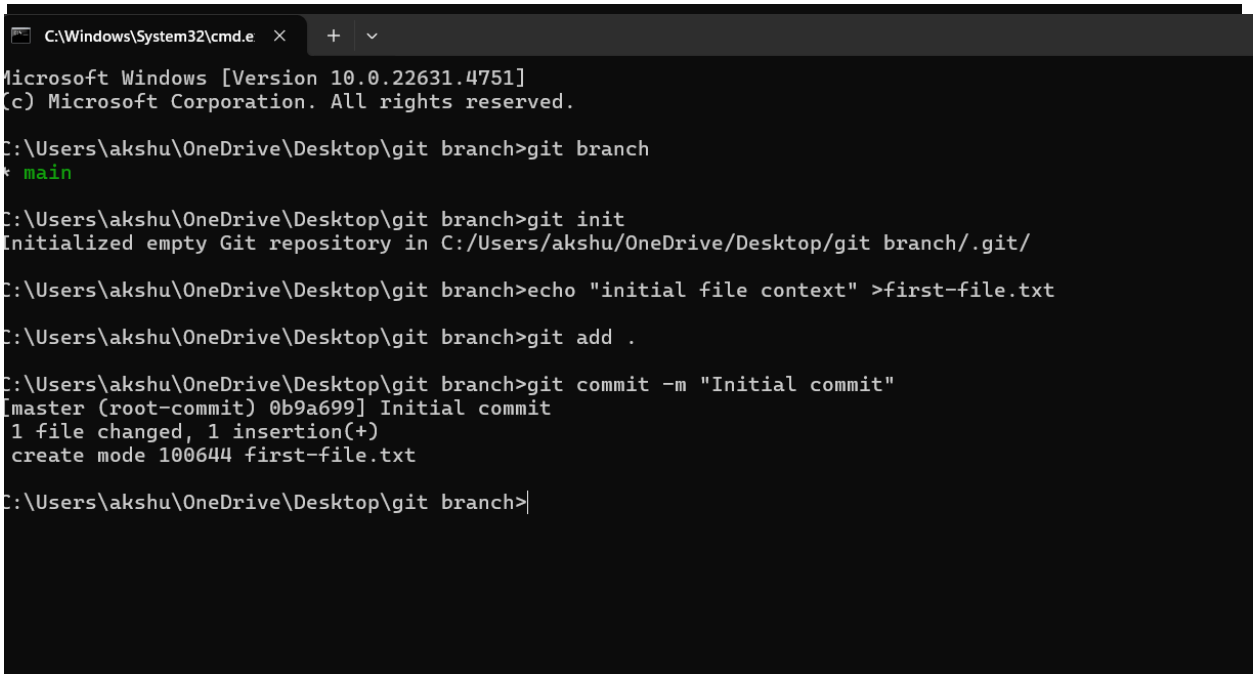
Add the File to Git

Tell Git to track this file:



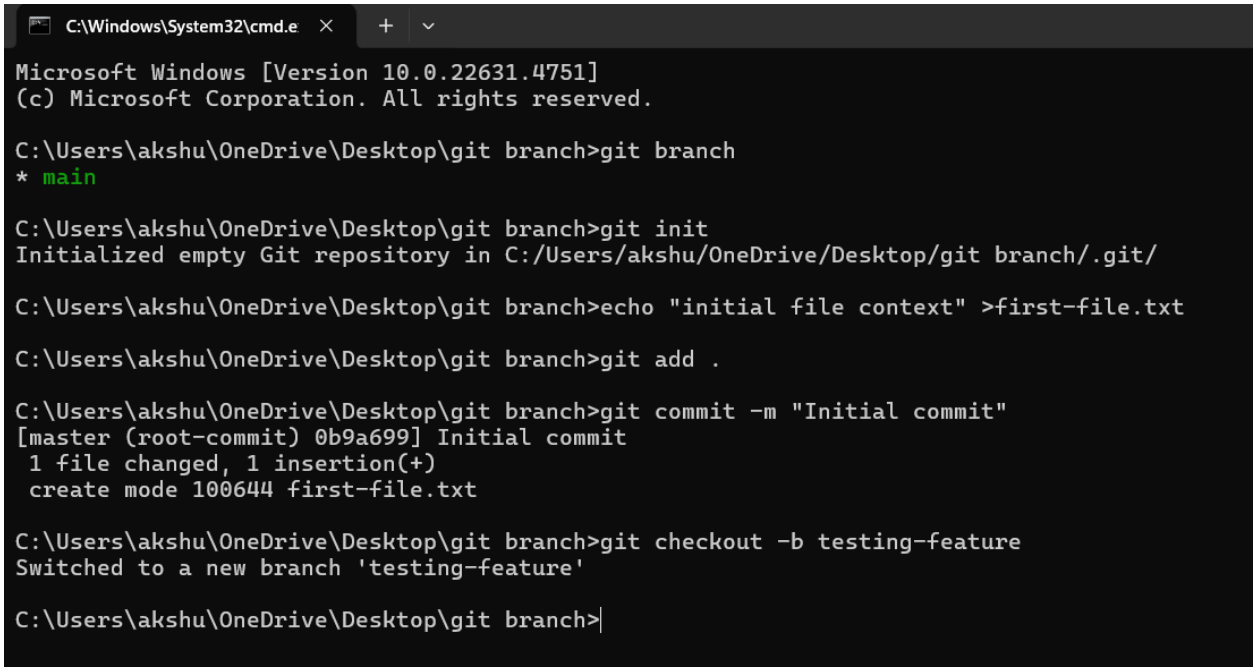
Step 6:

Save this change in Git with a commit message.



Step 7:

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



Step 8:

Let’s add a new file for our feature:

Step 9:

Now, stage the changes:

Step 10:

Commit the changes:

```
C:\Users\akshu\OneDrive\Desktop\git branch>git commit -m "Add new feature fi
le"
```

Step 11:

Switch to the master Branch

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

C:\Users\akshu\OneDrive\Desktop\git branch>
```

Step 12:

Merge Changes from testing-feature to master

```
Updating d680de9..d5ec286
Fast-forward
 first-file.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Step 13:

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

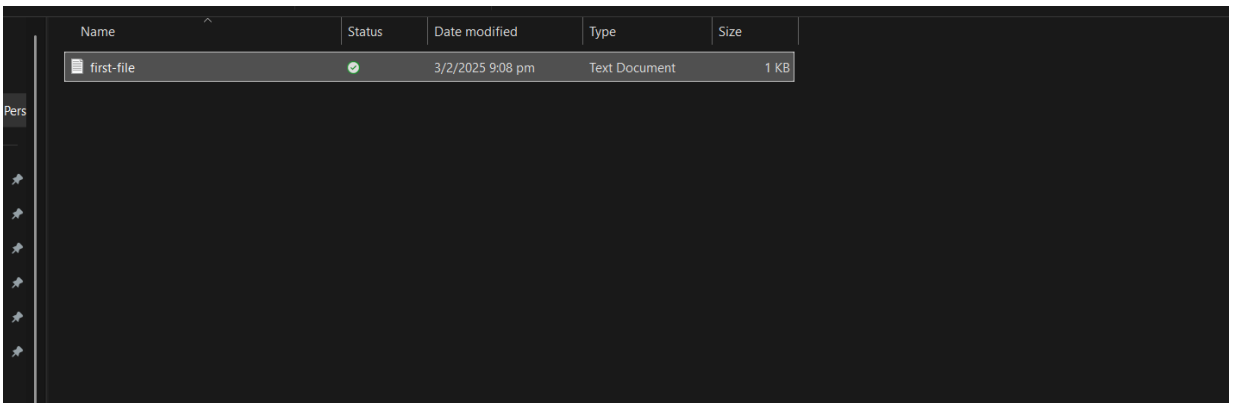

```
Updating d680de9..d5ec286
Fast-forward
 first-file.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```


Step 14:

Now, check the files in the folder:

```
C:\Users\akshu\OneDrive\Desktop\git branch>git branch -d testing-feature
Deleted branch testing-feature (was 0b9a699).

C:\Users\akshu\OneDrive\Desktop\git branch>
```



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.