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: HARSHINI A **Department:** CSE

Introduction

In this Proof of Concept (POC), Git is utilized for version control to streamline the development process. It enables developers to work on new features in separate branches, keeping them independent from the main branch. Once the features are complete, they can be merged back, promoting structured and collaborative development.

Objectives

1. **Parallel Development** – Allows multiple developers to work on different features or fixes simultaneously without interfering with each other's code.

2. **Code Isolation** – Enables experimentation and testing without affecting the main codebase.

3. **Efficient Collaboration** – Facilitates teamwork by allowing developers to contribute independently and merge their work seamlessly.

4. **Bug Fixing and Hotfixes** – Helps isolate and quickly resolve issues without disrupting ongoing development.

5. **Version Control and History Tracking** – Maintains a clear history of changes, making it easy to track who made what changes and why.

6. **Code Review and Quality Assurance** – Encourages structured code reviews before merging changes into the main branch.

Importance

1. **Enhances Workflow Efficiency** – Developers can work simultaneously on features, bug fixes, or improvements without overwriting each other's changes.

2. **Reduces Conflicts** – By isolating changes, Git branching minimizes conflicts and makes merges more manageable.

3. **Encourages Experimentation** – Allows developers to try new ideas without risking the stability of the main branch.

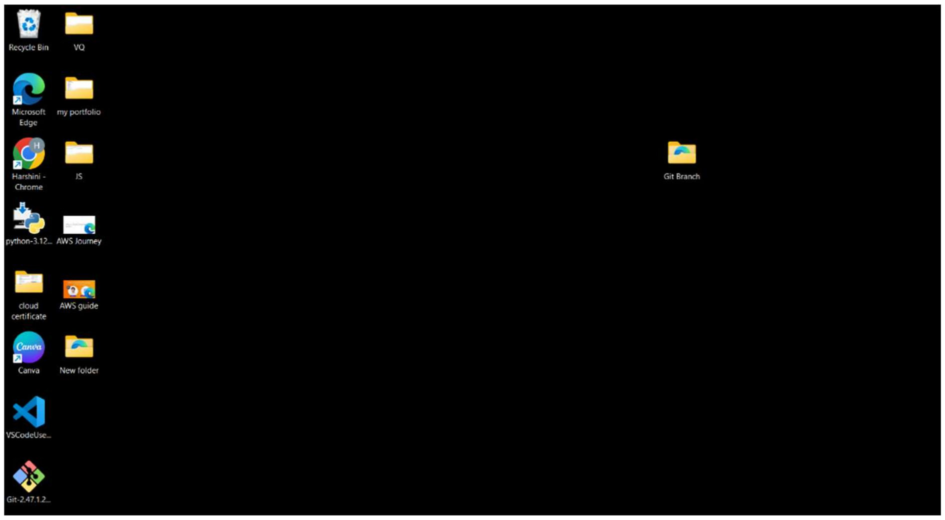
4. **Supports Multiple Versions** – Makes it easy to maintain different versions of software, such as production, development, and experimental branches.

5. **Simplifies Rollbacks** – If a feature branch introduces problems, it can be discarded without affecting the stable codebase.

Step-by-Step Overview

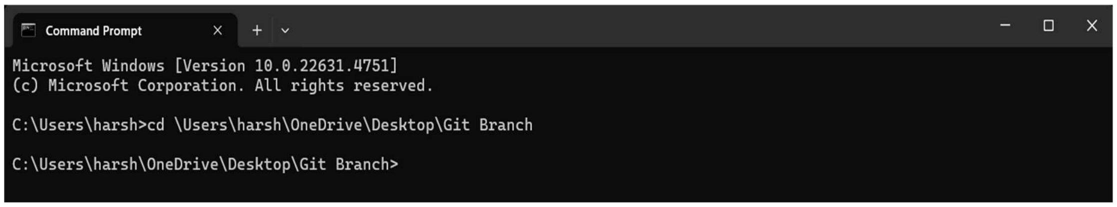
Step 1:

Create a folder in desktop and name the folder



Step 2:

Set the path to the folder created in first step in command prompt



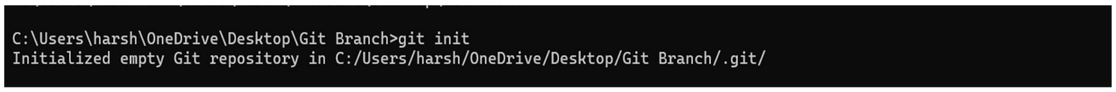
```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\harsh>cd \Users\harsh\OneDrive\Desktop\Git Branch

C:\Users\harsh\OneDrive\Desktop\Git Branch>
```

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.



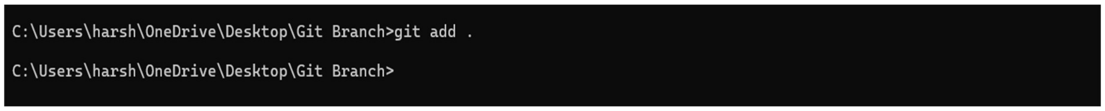
```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git init
Initialized empty Git repository in C:/Users/harsh/OneDrive/Desktop/Git Branch/.git/
```

Step 4:

Create a simple file and add it to the repository

Add the File to Git by using the command

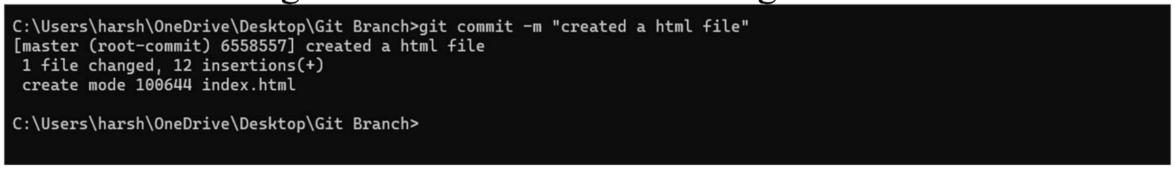
git add .



```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git add .

C:\Users\harsh\OneDrive\Desktop\Git Branch>
```

Step 5:

Save this change in Git with a commit message

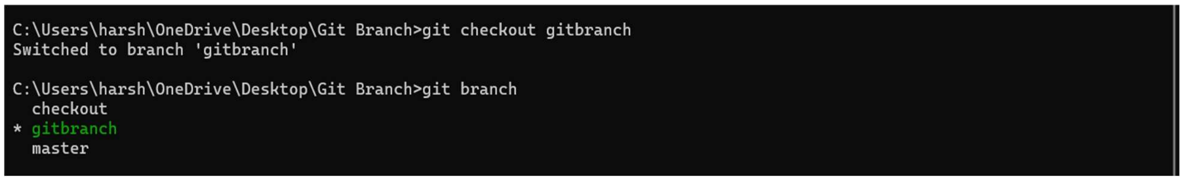


```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git commit -m "created a html file"
[master (root-commit) 6558557] created a html file
 1 file changed, 12 insertions(+)
 create mode 100644 index.html

C:\Users\harsh\OneDrive\Desktop\Git Branch>
```

Step 6:

Create and switch to a new branch called (gitbranch1) using the command

git checkout (new branch name)



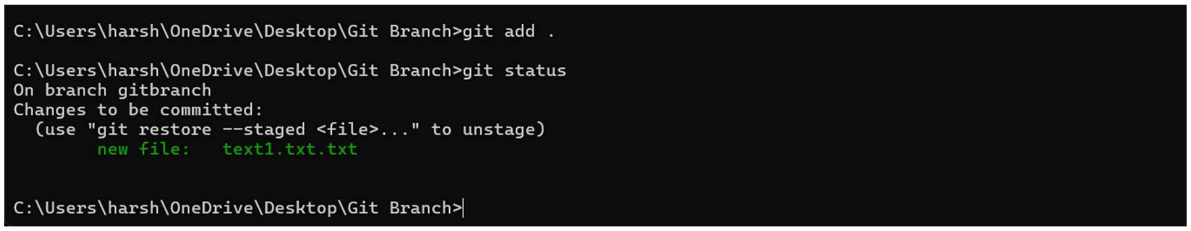
```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git checkout gitbranch
Switched to branch 'gitbranch'

C:\Users\harsh\OneDrive\Desktop\Git Branch>git branch
  checkout
* gitbranch
  master
```

Step 7:

Let’s create a new text file int Git Branch folder and add the file in git using the command

git add .

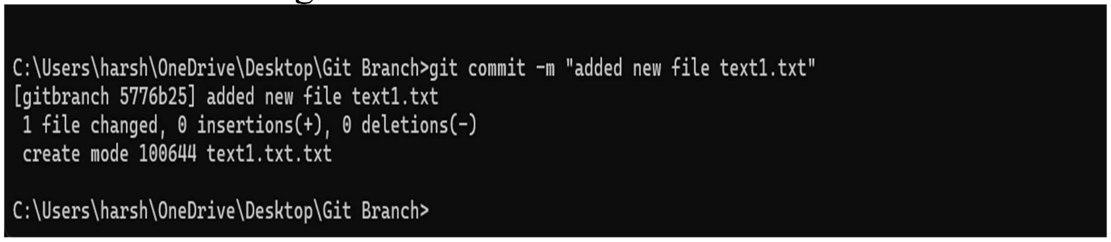
Now, stage the changes and check the status



```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git add .

C:\Users\harsh\OneDrive\Desktop\Git Branch>git status
On branch gitbranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   text1.txt.txt


C:\Users\harsh\OneDrive\Desktop\Git Branch>
```

Step 8:

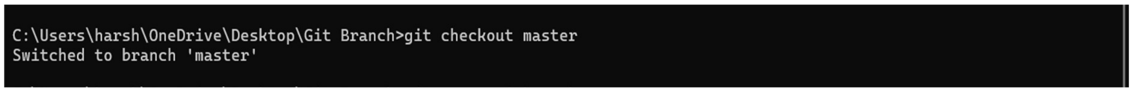
Commit the changes



```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git commit -m "added new file text1.txt"
[gitbranch 5776b25] added new file text1.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 text1.txt.txt

C:\Users\harsh\OneDrive\Desktop\Git Branch>
```

Step 9:

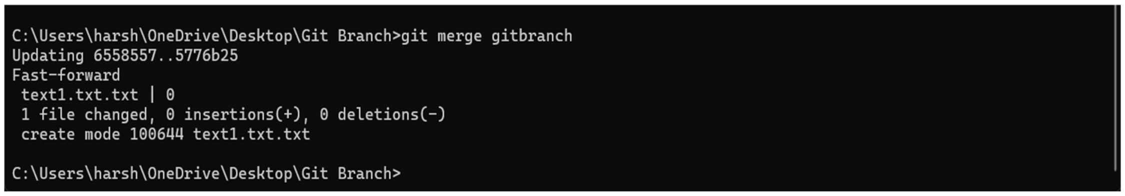
Switch to the master Branch



```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git checkout master
Switched to branch 'master'
```

Step 10:

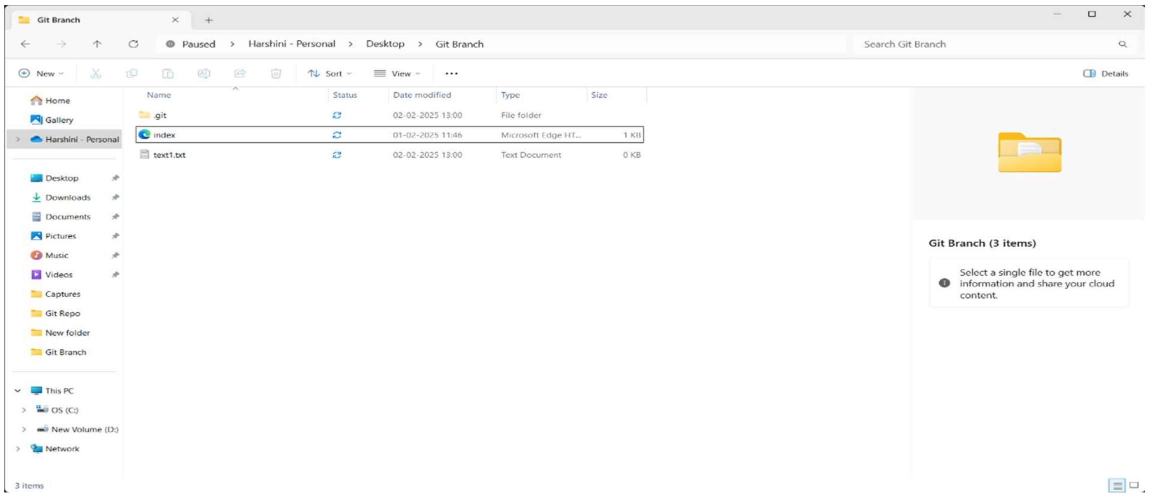
Merge Changes from git branch to master



```
C:\Users\harsh\OneDrive\Desktop\Git Branch>git merge gitbranch
Updating 6558557..5776b25
Fast-forward
 text1.txt.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 text1.txt.txt

C:\Users\harsh\OneDrive\Desktop\Git Branch>
```

Step 11:

Now, check the files in the folder:



Name	Status	Date modified	Type	Size
.git		02-02-2025 13:00	File folder	
index		01-02-2025 11:46	Microsoft Edge HT...	1 KB
text1.txt		02-02-2025 13:00	Text Document	0 KB

Git Branch (3 items)

Select a single file to get more information and share your cloud content.

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.