



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: Keshika D Department: ADS

Introduction:

In this task, we explore Git branching, which allows developers to work on new features independently from the main codebase. We create a new branch called testing to implement a feature, commit the changes, and then merge them back into the main branch. The process involves checking the current branch, making changes, and resolving conflicts if necessary. After merging, we verify the changes through commit history and file existence. This helps maintain clean and efficient workflow in collaborative development.

Overview

Here's what we will cover in this setup:

Start by navigating to your Git repository using the terminal.

- 1. Check the current branch using `git branch` to confirm you're on the correct branch (usually main).**
- 2. Create and switch to a new branch called testing using `git checkout -b testing`.**
- 3. Make changes by adding or modifying files (e.g., creating a new feature).**
- 4. Stage the changes with `git add` and commit them with `git commit`.**
- 5. Switch back to the main branch and merge the testing branch into it using `git merge`.**
- 6. Verify the merge by checking commit history with `git log` and confirm changes are present in the main branch.**

Objectives:

The objective of this task is to learn and implement Git branching to manage feature development separately from the main codebase. It aims to demonstrate how to create a new branch, add and commit changes, and merge those changes back into the main branch. The task also focuses on resolving potential merge conflicts and verifying successful changes. Additionally, it provides practice in navigating Git history to confirm the integration of new features. Ultimately, it improves version control skills in a collaborative development environment.

Step-by-Step Overview

Step 1:

1. Navigate to Your Git Repository:

Open the terminal and navigate to the folder where your Git repository is located. You can use the `cd` command to change the directory:

```
C:\WINDOWS\system32\cmd.exe
```

```
Microsoft Windows [Version 10.0.19045.5371]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\TEMP.DESKTOP-S0M6S0K.007>E:
```

```
E:\>cd E:\Keshika
```

```
E:\Keshika>cd E:\Keshika\PEP TASKS
```

Step 2

2. Check the Current Branch:

Before creating a new branch, it's important to know which branch you're currently on. Usually, it's main use this command : git branch

```
C:\WINDOWS\system32\cmd.exe
```

```
Microsoft Windows [Version 10.0.19045.5371]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\TEMP.DESKTOP-S0M6S0K.007>E:
```

```
E:\>cd E:\Keshika
```

```
E:\Keshika>cd E:\Keshika\PEP TASKS
```

```
E:\Keshika\PEP TASKS>git branch
```

```
E:\Keshika\PEP TASKS> git branch
```

Step 3

3. Create a New Branch:

To create a new branch, use the following command:

git checkout -b testing

```
E:\Keshika\PEP TASKS> git checkout -b testing
Switched to a new branch 'testing'
```

Step 4: Make Changes:

you're on the testing branch, you can add a new feature. For example, you might add a new file or edit existing one. For this example, let's say you add a file called feature.t

command :echo "This is my new feature!" > feature.txt

```
E:\Keshika\PEP TASKS> echo "This is my new feature!" > feature.txt
```

Step 5

Stage and Commit Changes:

After making changes to the files, you need to stage and commit those change command : git

add feature.txt

commit changes:

command: git commit -m "Added new feature: feature.txt"

```
E:\Keshika\PEP TASKS> git branch
* testing

E:\Keshika\PEP TASKS> git branch -r

E:\Keshika\PEP TASKS> git checkout -b main
Switched to a new branch 'main'
```

```
E:\Keshika\PEP TASKS> echo "This is my new feature!">feature.txt

E:\Keshika\PEP TASKS> git add feature.txt

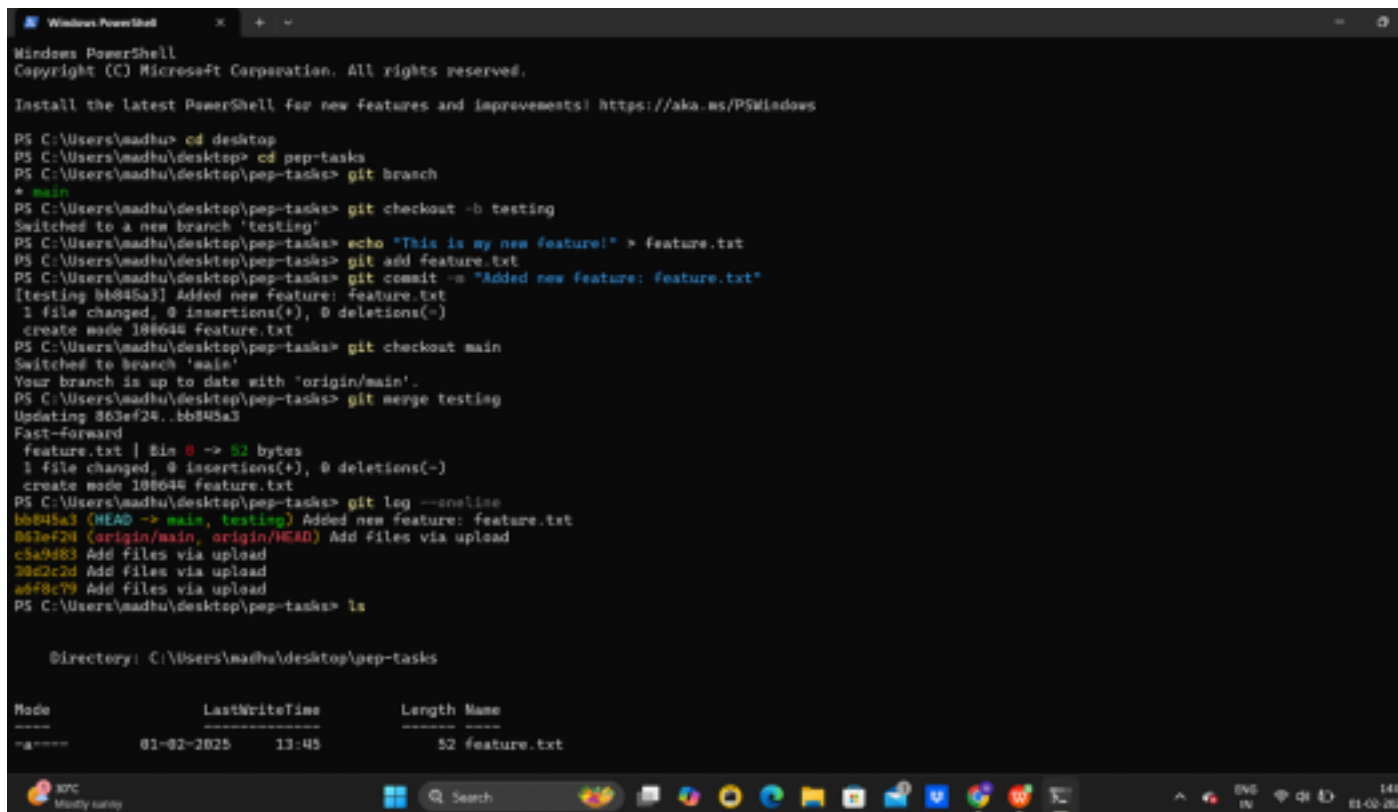
E:\Keshika\PEP TASKS> git commit -m "Added new feature: feature.txt"
[main 77ad092] Added new feature: feature.txt
1 file changed, 1 insertion(+), 1 deletion(-)
```

Step 6

Switch Back to the Main Branch:

After committing the changes in the testing branch, switch back to the main branch :

using this command -> checkout main



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\madhu> cd desktop
PS C:\Users\madhu\desktop> cd pep-tasks
PS C:\Users\madhu\desktop\pep-tasks> git branch
* main
PS C:\Users\madhu\desktop\pep-tasks> git checkout -b testing
Switched to a new branch 'testing'
PS C:\Users\madhu\desktop\pep-tasks> echo "This is my new feature!" > feature.txt
PS C:\Users\madhu\desktop\pep-tasks> git add feature.txt
PS C:\Users\madhu\desktop\pep-tasks> git commit -m "Added new feature: feature.txt"
[testing bb045a3] Added new feature: feature.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 feature.txt
PS C:\Users\madhu\desktop\pep-tasks> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\madhu\desktop\pep-tasks> git merge testing
Updating 863ef24..bb045a3
Fast-forward
 feature.txt | Bin 0 -> 52 bytes
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 feature.txt
PS C:\Users\madhu\desktop\pep-tasks> git log --oneline
bb045a3 (HEAD -> main, testing) Added new feature: feature.txt
863ef24 (origin/main, origin/HEAD) Add files via upload
c5e9d83 Add files via upload
30d2c2d Add files via upload
a6f8c79 Add files via upload
PS C:\Users\madhu\desktop\pep-tasks> ls

Directory: C:\Users\madhu\desktop\pep-tasks

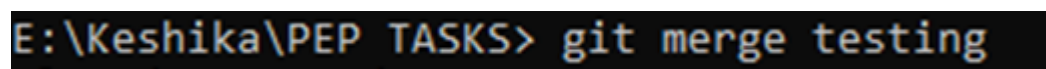
Mode                LastWriteTime         Length Name
----                -
-a-----         01-02-2025   13:45             52 feature.txt
```

Step 7:

Merge the Testing Branch into Main:

Now that you're on the main branch, merge the changes from the testing branch: using

command : git merge testing



```
E:\Keshika\PEP TASKS> git merge testing
```

Step 8:

you can verify your merging using this command : `ls`

it lists the files present in the current directory make sure your file is present.

OUTCOMES:

The outcome of this task is the successful creation and management of Git branches for feature development. You will be able to independently work on a new feature in a separate branch and merge it into the main branch without affecting the main codebase. The task will enhance your ability to handle commit history and track changes effectively. You will gain experience in resolving merge conflicts ensuring a smooth integration of new features. Additionally, the task improves your understanding of Git workflows in collaborative projects. By the end, you will be proficient in basic version control techniques with Git.

