



St. JOSEPH'S
GROUP OF INSTITUTIONS
OMR, CHENNAI - 119



Placement Empowerment Program

Cloud Computing and DevOps Centre

Git branching and merging

Name: Esly Abro

Department: IT



Introduction:

In this we will explore **Git branching and merging**, key concepts in managing collaborative development. Git branching allows us to work on new features independently from the main codebase, while merging brings those changes back into the main branch once they are ready. We'll create a new branch, implement changes, and merge those changes back into the main branch using Git commands in a Windows environment.

Overview:

This covers:

1. Creating a new branch for testing a feature.
2. Making and committing changes on the new branch.
3. Merging the changes back into the main branch.
4. Verifying the success of the merge.

Objectives:

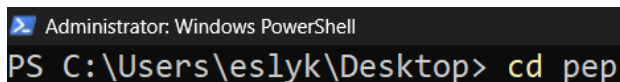
- Learn how to create and switch branches using Git.
- Understand how to commit changes and manage versions using Git.
- Gain practical experience with the `git merge` command to incorporate new features into the main codebase.
- Practice using branching for collaborative development and conflict resolution.

Step-by-Step Overview:

Step 1: Navigate to the Git repository

- Open Command Prompt (cmd) or PowerShell.
- Navigate to your Git project using the `cd` command.

```
cd path\to\your\repository
```

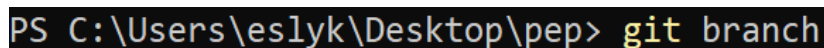


```
Administrator: Windows PowerShell
PS C:\Users\eslyk\Desktop> cd pep
```

Step 2: Check the Current Branch

- Before creating a new branch, confirm you're on the main branch.

```
git branch
```



```
PS C:\Users\eslyk\Desktop\pep> git branch
```

If not on the main branch, switch to it using:

```
git checkout main
```

```
PS C:\Users\eslyk\Desktop\pep> git checkout main
```

Step 3: Create a New Branch

- Create and switch to a new branch called testing where the new feature will be developed.

```
git checkout -b testing
```

```
PS C:\Users\eslyk\Desktop\pep> git checkout -b testing  
Switched to a new branch 'testing'
```

Step 4: Make Changes

- Add a new feature by creating or modifying a file. For example, create a file called feature.txt:

```
echo "This is my new feature!" > feature.txt
```

Step 5: Stage and Commit Changes

- Add the changes to the staging area:

```
git add feature.txt
```

- Commit the changes with a descriptive message:

```
git commit -m "Added new feature in feature.txt"
```

```
PS C:\Users\eslyk\Desktop\pep> echo "New Feature" > feature.txt  
PS C:\Users\eslyk\Desktop\pep> git add feature.txt  
PS C:\Users\eslyk\Desktop\pep> git commit -m "Added new feature: feature.txt"  
[testing (root-commit) 9b5cda5] Added new feature: feature.txt  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 Desktop/pep/feature.txt
```

Step C: Switch Back to the Main Branch

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

```
git checkout main
```

Step 7: Merge the Testing Branch into Main

- Merge the changes from the testing branch into the main branch:

```
git merge testing
```

```
PS C:\Users\eslyk\Desktop\pep> git merge testing
Updating 863ef24..bb845a3
Fast-forward
 feature.txt | Bin 0 -> 52 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature.txt
```

Step 8: Verify the Merge

- Use the following commands to verify that the file was successfully merged:

```
git log --oneline
ls
```

```
PS C:\Users\eslyk\Desktop\pep> ls

Directory: C:\Users\eslyk\Desktop\pep

Mode                LastWriteTime         Length Name
----                -
-a-----          2/4/2025   9:12 PM             74 feature.txt

PS C:\Users\eslyk\Desktop\pep>
```

This will list the files in the directory and show the commit history, confirming that `feature.txt` was added.

Outcomes:

- You will have successfully created and managed Git branches for feature development.
- You will understand how to create a new branch, commit changes, and merge those changes into the main branch.
- This task improves your understanding of how to handle multiple features and maintain a clean and efficient Git workflow in collaborative development environments.

By the end, you will have gained practical experience in version control with Git, which is essential for working in teams and managing multiple features effectively.

