 

**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

**Set up Git Branching**

Name: Monika K Department: ADS



**Introduction and Overview**

Git branching is a crucial technique in version control that enables developers to work on different features, bug fixes, or experiments **without affecting the main codebase**. By creating and managing branches, developers can maintain a **clean and organized workflow**, test new functionalities, and merge changes when they are ready.

This **Proof of Concept (PoC)** will guide you through:

* Creating a new Git branch
* Switching between branches
* Merging changes
* Deleting branches

By following this process, you will gain hands-on experience with **Git branching**, which is essential for collaboration in software development.

**Objectives**

The goal of this project is to:  
✅ Understand the purpose and benefits of Git branching  
✅ Create and manage branches effectively  
✅ Work on new features without affecting the main branch  
✅ Merge changes and resolve conflicts  
✅ Delete unnecessary branches to maintain a clean repository

**Importance of Git Branching**

📌 **Isolates Work** – Developers can work on multiple features simultaneously without interfering with each other.  
📌 **Enhances Collaboration** – Different team members can contribute without affecting the main project.  
📌 **Safe Experimentation** – Try new ideas and revert if necessary without modifying stable code.  
📌 **Facilitates Code Reviews** – Merging only tested and reviewed code ensures better quality.

**Step-by-Step Overview**

**Step 1: Initialize a Git Repository**

1. Open **Command Prompt** or **Terminal**.
2. Navigate to your project folder:

CopyEdit

**cd /path/to/your/project**

1. Initialize a Git repository (if not already done):

CopyEdit

**git init**

**Step 2: Create a New Branch**

1. To check the current branch:

CopyEdit

**git branch**

You will see main (or master) as the default branch.

1. To create a new branch:

CopyEdit

**git branch feature-branch**

Replace **feature-branch** with your desired branch name.

**Step 3: Switch to the New Branch**

1. To move to the newly created branch:

CopyEdit

**git checkout feature-branch**

OR

CopyEdit

**git switch feature-branch**

1. Confirm that you are on the correct branch:

CopyEdit

**git branch**

The active branch will be highlighted with \*.

**Step 4: Make Changes and Commit**

1. Modify or add files in your project folder.
2. Add the changes to the staging area:

CopyEdit

**git add** .

1. Commit the changes with a message:

CopyEdit

**git commit -m "Added new feature"**

**Step 5: Merge the Branch with the Main Branch**

1. Switch back to the main branch:

CopyEdit

git checkout main

2.Merge the feature branch:

CopyEdit

git merge feature-branch

1. If there are conflicts, Git will notify you. Open the conflicting files, resolve the differences, and use:

CopyEdit

**git add** .

git commit -m "Resolved merge conflicts"

**Step 6: Delete a Branch (Optional)**

1. To delete the branch after merging:

CopyEdit

**git branch -d feature-branch**

1. If the branch hasn’t been merged yet and you still want to delete it:

CopyEdit

**git branch -D feature-branch**

**Step 7: Push Changes to Remote Repository (GitHub/GitLab/Bitbucket)**

1. Add the remote repository (if not already added):

CopyEdit

**git remote add origin https://github.com/yourusername/repository.git**

1. Push the branch to the remote repository:

CopyEdit

**git push origin feature-branch**

1. After merging locally, push the updated main branch:

CopyEdit

**git push origin main**