 

**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: Rohan.R Department: CSE



**Introduction**

In modern software development, version control systems like Git are essential tools for managing changes in a project and collaborating with others. Git allows developers to work on multiple features or fixes in parallel through the concept of **branching**. Branching enables isolated work on different tasks without affecting the main codebase, making it a powerful method for feature development, bug fixing, or experimentation.

**Objective**

The objective of this exercise is to:

1. Create a new branch in your Git repository for testing.
2. Add a new feature or change to the branch.
3. Test the new feature or changes.
4. Merge the new feature back into the main branch after confirming it works as expected.

**Step-by-Step Procedure**

**1. Create a New Branch**

First, navigate to your Git repository in the terminal. To create a new branch for testing purposes, use the following command:

git branch <branch-name>

**2. Switch to the New Branch**

After creating the new branch, you need to switch to it in order to start working. Use the git checkout command:

git checkout <branch-name>

**3. Add a New Feature or Modify Files**

Now that you are on the new branch, make the necessary changes to the codebase. This could involve adding new files, modifying existing

ones, or implementing a feature. For example, you might add a simple change like modifying a README file or creating a new script.

**4. Stage and Commit Changes**

After making the changes, stage the modified files using:

git add <file-name>

Or to stage all changes, use:

git add .

Then, commit the changes with a descriptive message:

git commit -m "Added new feature for testing"

**5. Test the Changes Locally**

Before merging, it is important to ensure that your changes work as expected. Run any necessary tests or manually check the changes to verify that the new feature is functioning correctly.

**6. Switch Back to the Main Branch**

Once you've confirmed that your changes are ready to be merged, switch back to the main branch (often called main or master):

git checkout main

**7. Merge the New Feature Branch into the Main Branch**

Now, merge the changes from your feature branch into the main branch. Use the git merge command:

git merge <branch-name>

**8. Push the Changes to the Remote Repository**

After the merge, push the changes to your remote repository (e.g., GitHub, GitLab, etc.):

git push origin main

**9. Delete the Feature Branch (Optional)**

If you no longer need the feature branch after merging, you can delete it to keep the repository clean:

git branch -d <branch-name>

To delete the branch remotely, use:

git push origin --delete <branch-name>

**Conclusion**

Following the steps above, you’ve successfully created a new Git branch, worked on a feature, tested the changes, and merged them into the main branch. This process is fundamental to Git’s branching and merging features, which help teams collaborate more effectively while minimizing the risk of introducing errors into the main codebase. Mastering these concepts will improve your workflow and help you manage development tasks more efficiently.