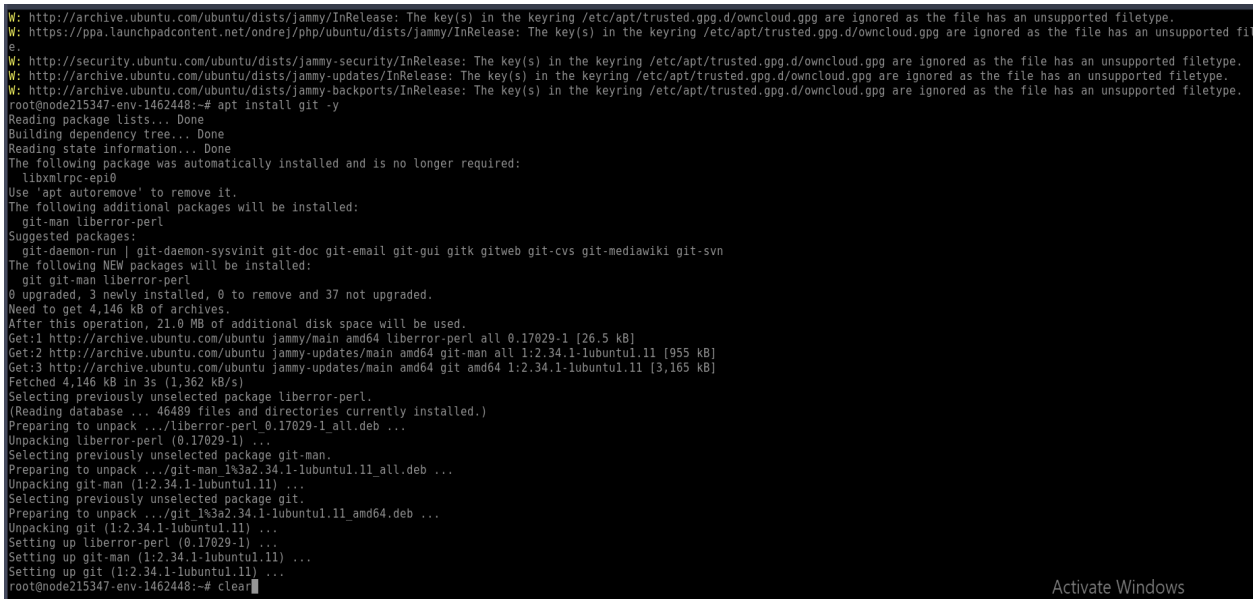
GitHub Workflow Documentation

This documentation outlines the step-by-step process for setting up a GitHub repository, creating branches, and simulating a complete development workflow.

1. Create a Repository on GitHub

Steps:

1. Log in to your GitHub account.
2. Click **New** to create a new repository.
3. Enter a repository name and description, and choose whether it's public or private.
4. Click **Create repository**.
5. Next we have to install **Git** on our local.



2. Clone the Repository

Command:

-> git clone https://github.com/your-username/repository-name.git

-> cd repository-name

Explanation:

- git clone: Clones the repository to your local machine.
- cd: Navigates into the cloned repository directory.

3. Create the Main Branch

-> git checkout -b main

-> git push -u origin main

Probably we used this above commands to create main branch. But I already create main branch as default branch on github manually.

Explanation:

- git checkout -b main: Creates a new branch named main and switches to it.
- git push -u origin main: Pushes the main branch to the remote repository and sets it as the upstream branch.

4. Create Development and Feature Branches

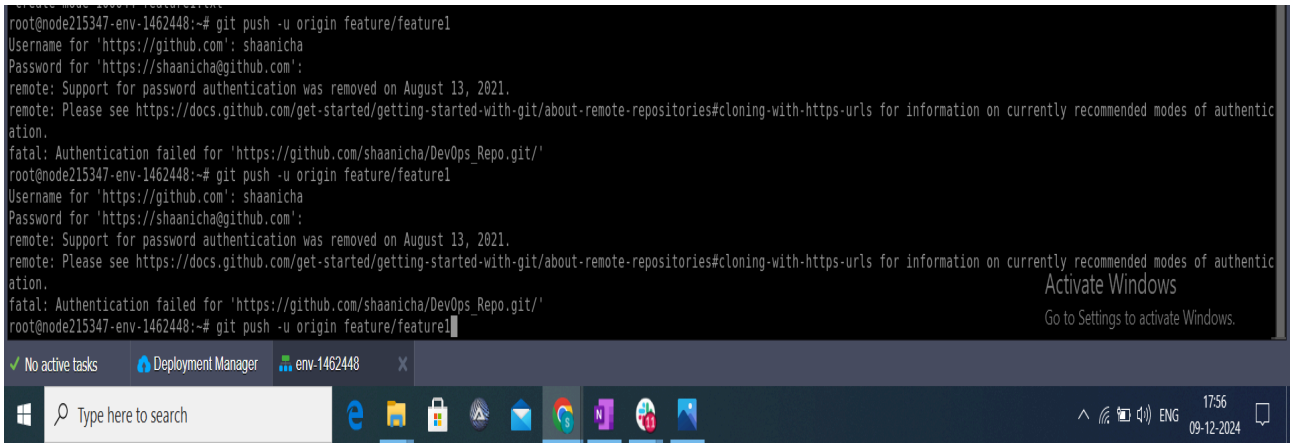
Commands:

-> git checkout -b feature/feature1

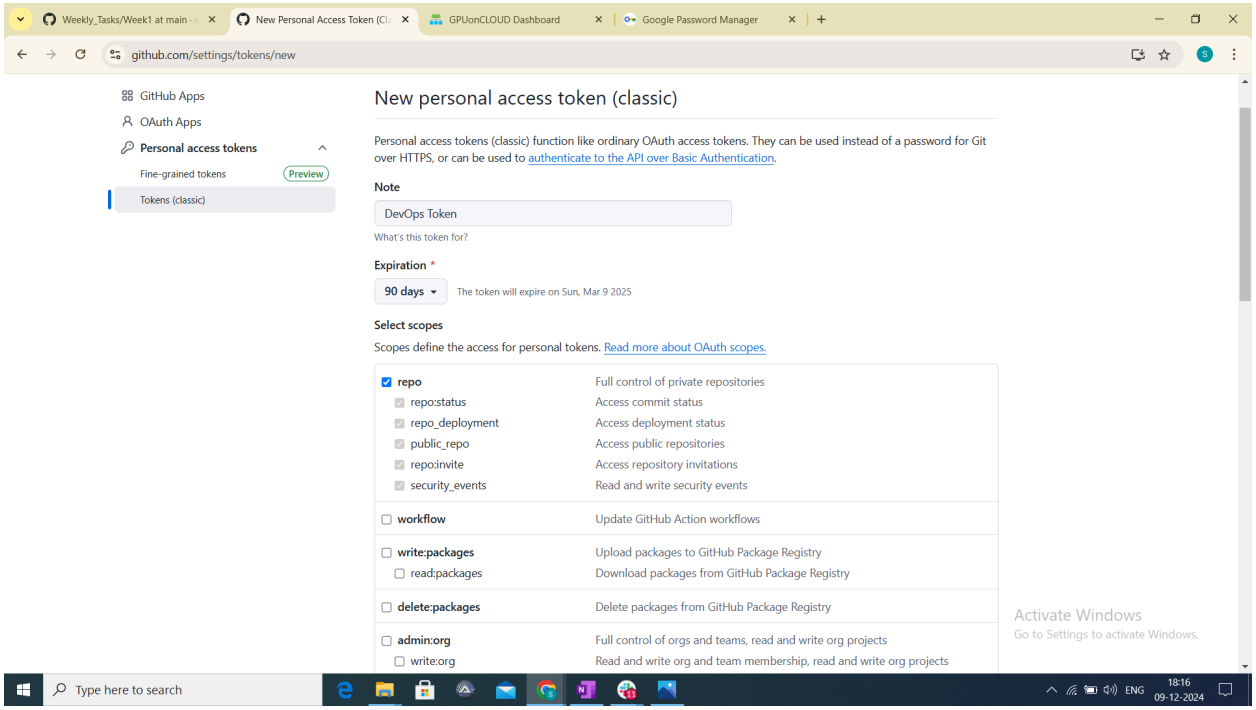
```
root@node215347-env-1462448:~#
root@node215347-env-1462448:~# git checkout -b feature/feature1
Switched to a new branch 'feature/feature1'
root@node215347-env-1462448:~#
root@node215347-env-1462448:~# git status
On branch feature/feature1
```

-> git push origin feature/feature1

After this command aksed for password and username. If we enter our password and github username it show error.



So, We need to create a personal access token.and once it is created it gives one key password. While push operation we have to use this key as username.



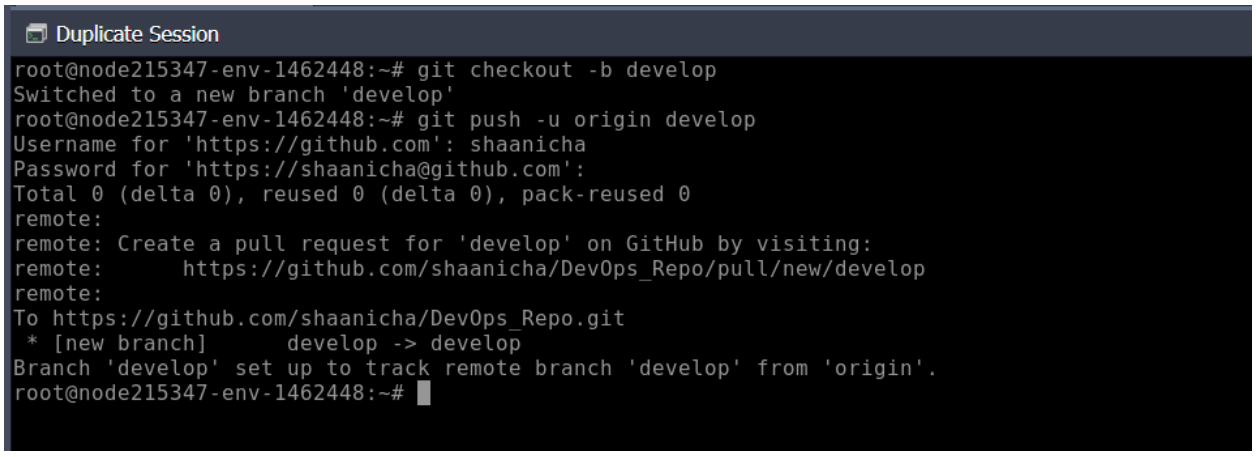
Commands:

-> git checkout -b develop

-> git push -u origin develop

Explanation:

- git checkout -b develop: Creates and switches to the develop branch.



5. Work on a Feature Branch

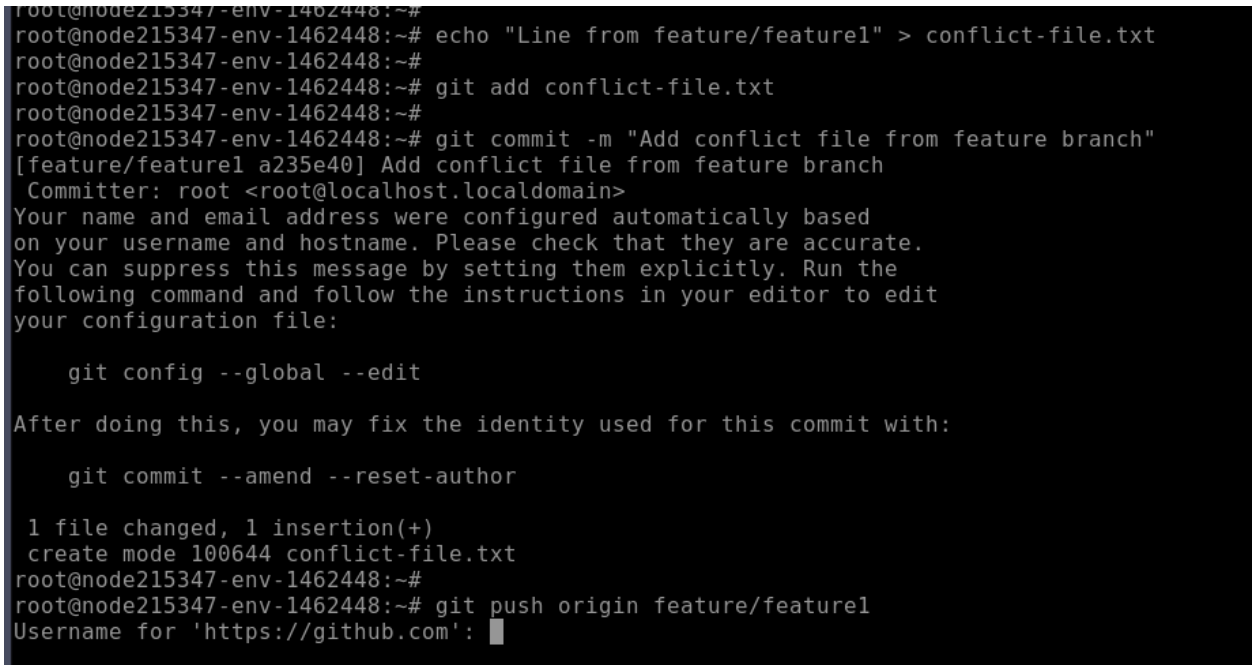
Commands:

-> echo "Line from feature/feature1" > conflict-file.txt

-> git add conflict-file.txt

-> git commit -m "Add conflict file from feature branch"

-> git push origin feature/feature1



Explanation:

- echo: Adds content to a new file named feature1.txt.
- git add: Stages the file for commit.
- git commit: Commits the changes with a descriptive message.
- git push: Pushes the changes to the remote feature branch.

5. Work on a Develop Branch:

Commands:

-> echo "Line from develop" > conflict-file.txt

-> git add conflict-file.txt

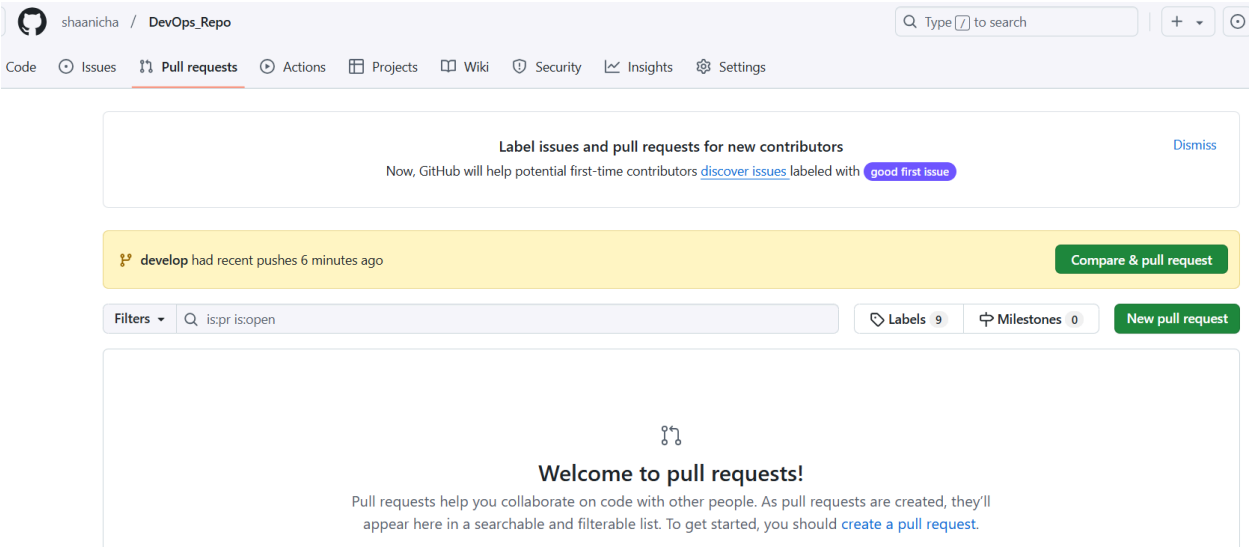
-> git commit -m "Add conflict file from develop"

-> git push origin develop

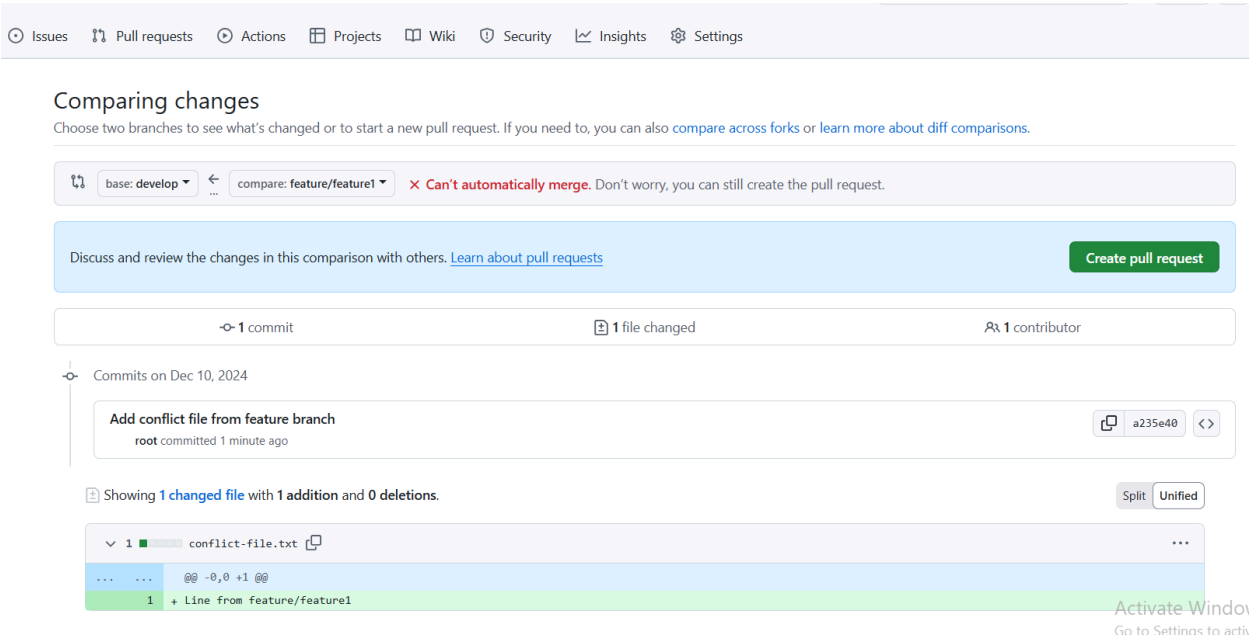
6. Create a Pull Request (PR)

Steps:

1. Navigate to the repository on GitHub.
2. Go to the **Pull Requests** tab and click **New Pull Request**.
3. Select feature/feature1 as the source branch and develop as the target branch.
4. Add a title and description, then click **Create Pull Request**.



1. **Create a Pull Request from feature/feature1 to develop:**
   - On GitHub, create a pull request.
   - This will now show a **merge conflict** because the same file (conflict-file.txt) has conflicting changes in both branches.



7. Resolve Merge Conflicts

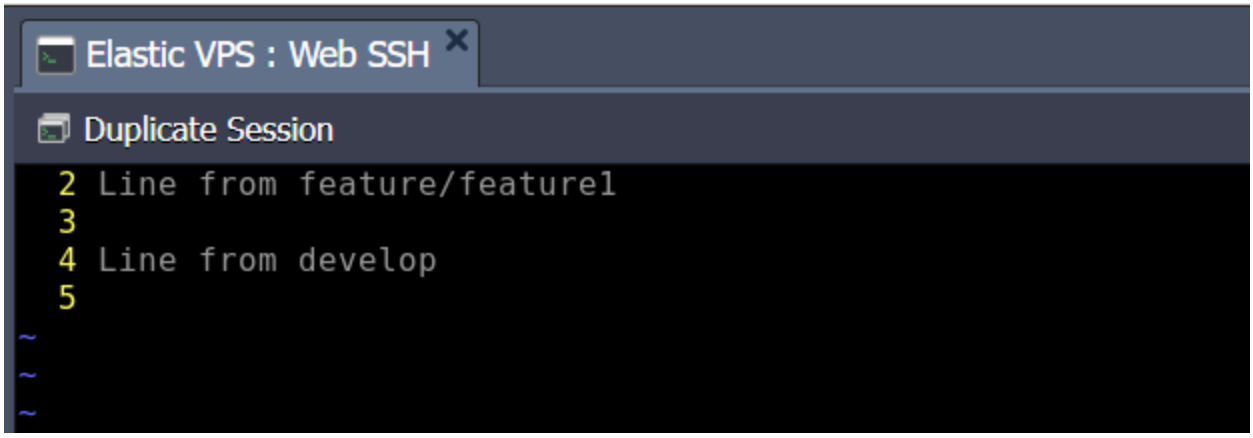
1. Identify the Conflicted File: From the output:

2. Open the Conflicted File: Open conflict-file.txt in a text editor

-> vim conflict-file.txt

Resolve the Conflict: Decide which content to keep, combine, or modify.

For example:



3. Save and Exit the File:

4. Mark the Conflict as Resolved: Add the resolved file to the staging area:

-> git add conflict-file.txt

5. Commit the Changes: Once the conflict is resolved, commit the changes:

-> git commit -m "Resolve merge conflict in conflict-file.txt"

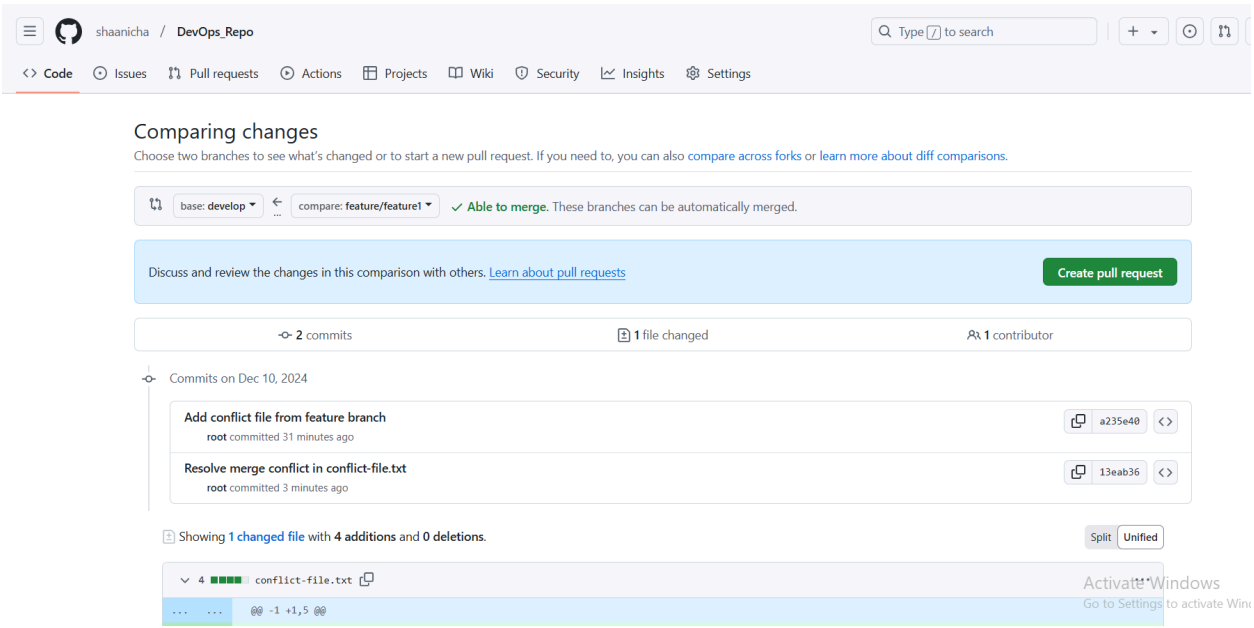
6. Push the Updated Branch: Push the resolved branch back to the remote repository:

-> git push origin feature/feature1

```
Elastic VPS : Web SSH
Duplicate Session
root@node215347-env-1462448:~# git add conflict-file.txt
root@node215347-env-1462448:~# git commit -m "Resolve merge conflict in conflict-file.txt"
[feature/feature1 13eab36] Resolve merge conflict in conflict-file.txt
 Committer: root <root@localhost.localdomain>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

root@node215347-env-1462448:~# git push origin feature/feature1
Username for 'https://github.com': shaanicha
Password for 'https://shaanicha@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 365 bytes | 365.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/shaanicha/DevOps_Repo.git
   a235e40..13eab36  feature/feature1 -> feature/feature1
root@node215347-env-1462448:~#
```

7. Check the Pull Request on GitHub

- Go back to your GitHub repository and view the pull request.
- It should now indicate that the conflict is resolved and allow you to merge the branches.



7. Merge Develop into main

Commands:

-> git checkout main

-> git merge develop

->git push origin main

Explanation:

git merge: Integrates changes from develop into main.

git push: Updates the remote main branch with the merged changes.

Created By

Shraddha Chaudhari