**Merge conflicts in GitHub**

A **merge conflict** in Git is a situation that arises when two or more developers make changes to the same part of a file in different branches, and Git is unable to automatically determine which changes should take precedence. This results in a "conflict" that must be manually resolved by a developer to combine the changes in a meaningful way.

This usually happens when:

1. Two people edit the **same line of code** in different branches.
2. A file is **deleted in one branch** but modified in another.
3. Changes are made **to the same section of a file** in both branches.

* **Collaboration Context:**
  + Merge conflicts typically occur in collaborative environments where multiple developers are working on the same codebase.
* **Same Part of a File:**
  + The conflict arises specifically when changes overlap in the same lines or sections of a file.
* **Automatic Resolution Failure:**
  + Git's automatic merge mechanism fails because it cannot decide which version of the code to keep.
* **Manual Intervention Required:**
  + A developer must step in to review the conflicting changes, decide how to combine them, and manually resolve the conflict.

**Example Scenario:**

* **Developer A** modifies line 10 of script.tf in the main branch.
* **Developer B** modifies the same line 10 of script.tf in the feature branch.
* When Developer B tries to merge feature into main, Git detects a conflict because both developers changed the same line.

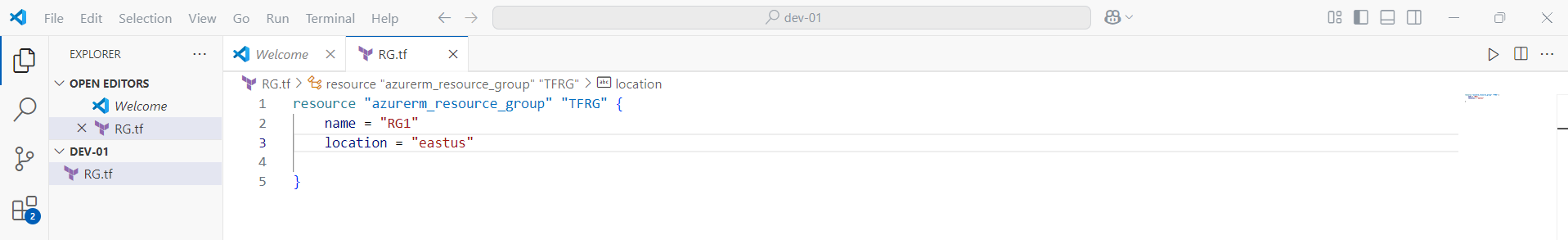
A merge conflict is Git's way of saying, "I found overlapping changes in the same part of a file, and I need your help to figure out how to combine them." It’s a natural part of collaborative development and requires manual resolution to ensure the codebase remains consistent and functional.

**Let’s Perform the Merge conflict practically.**

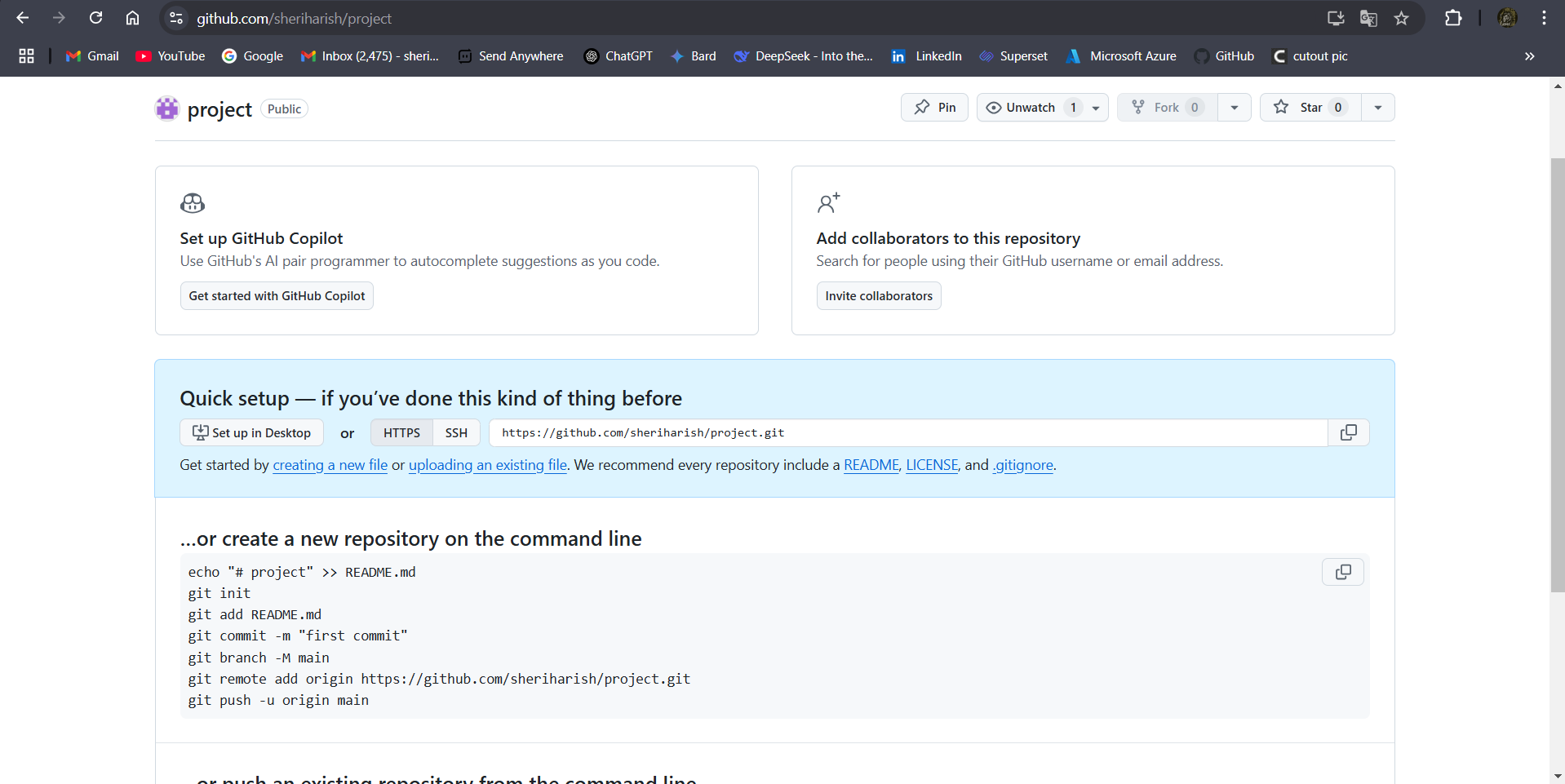
Consider two developers (Dev-01 & Dev-02) who were working on the same source code of a project

Both are trying to edit the same line in the code.

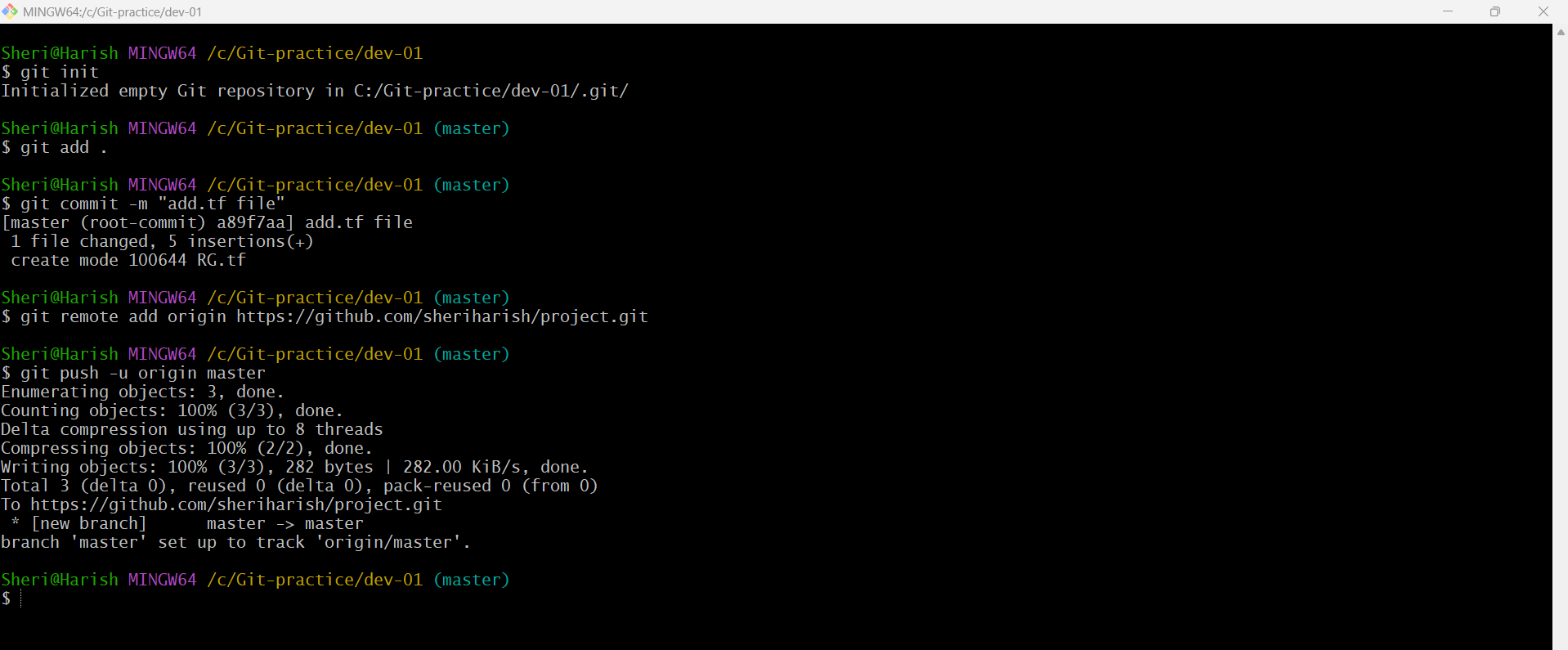
**Step1:** create a terraform configuration file in any folder (Dev-01).

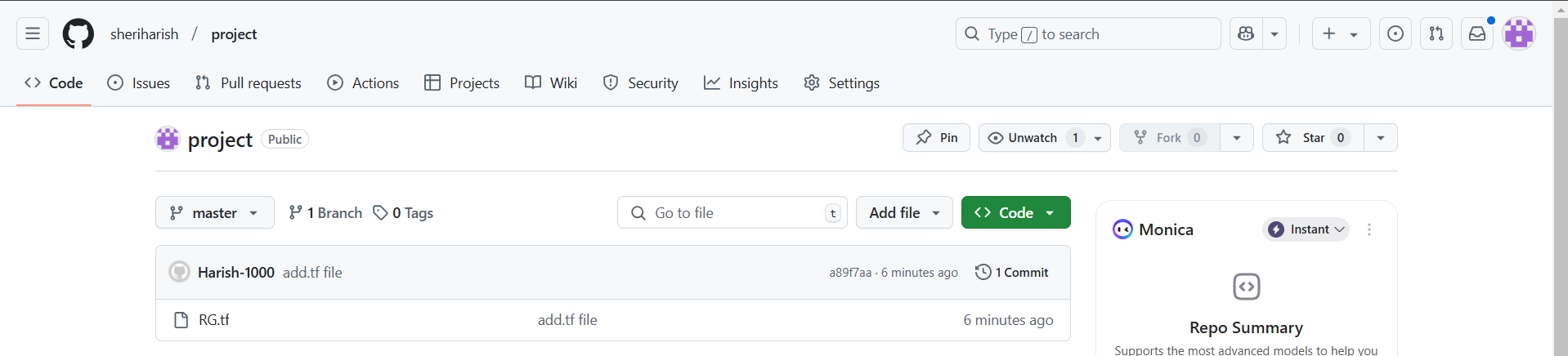


**Step2**: Create the repository in GitHub.

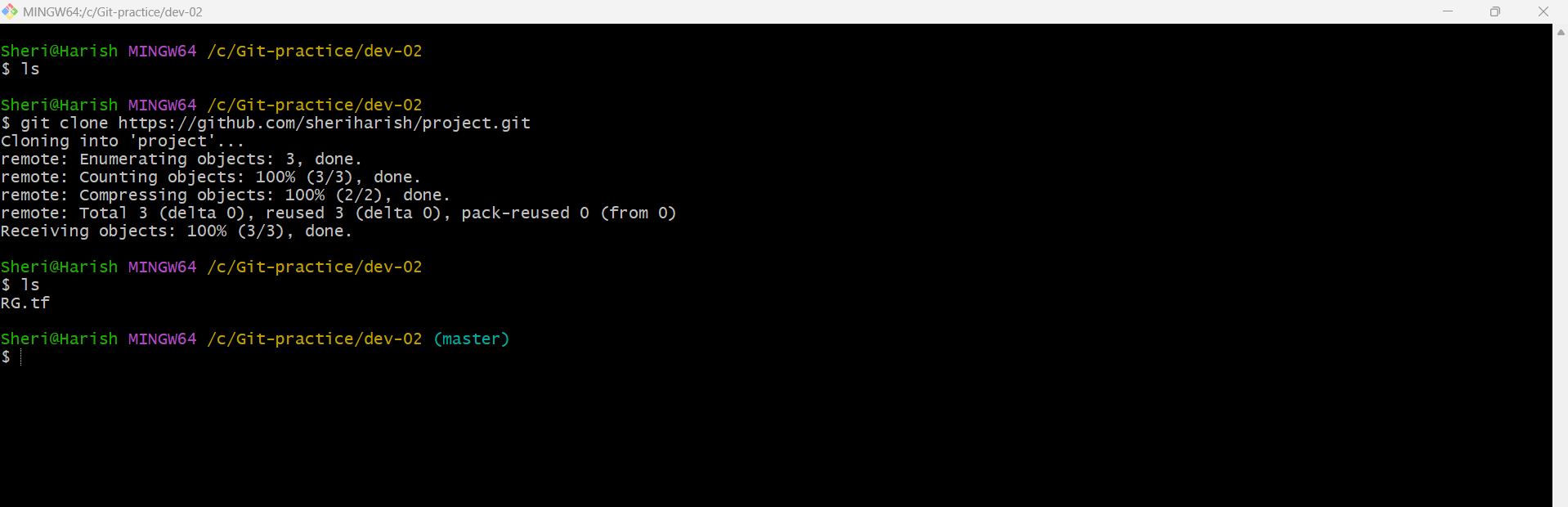


**Step3:** Push the terraform configuration file into the repository (project).

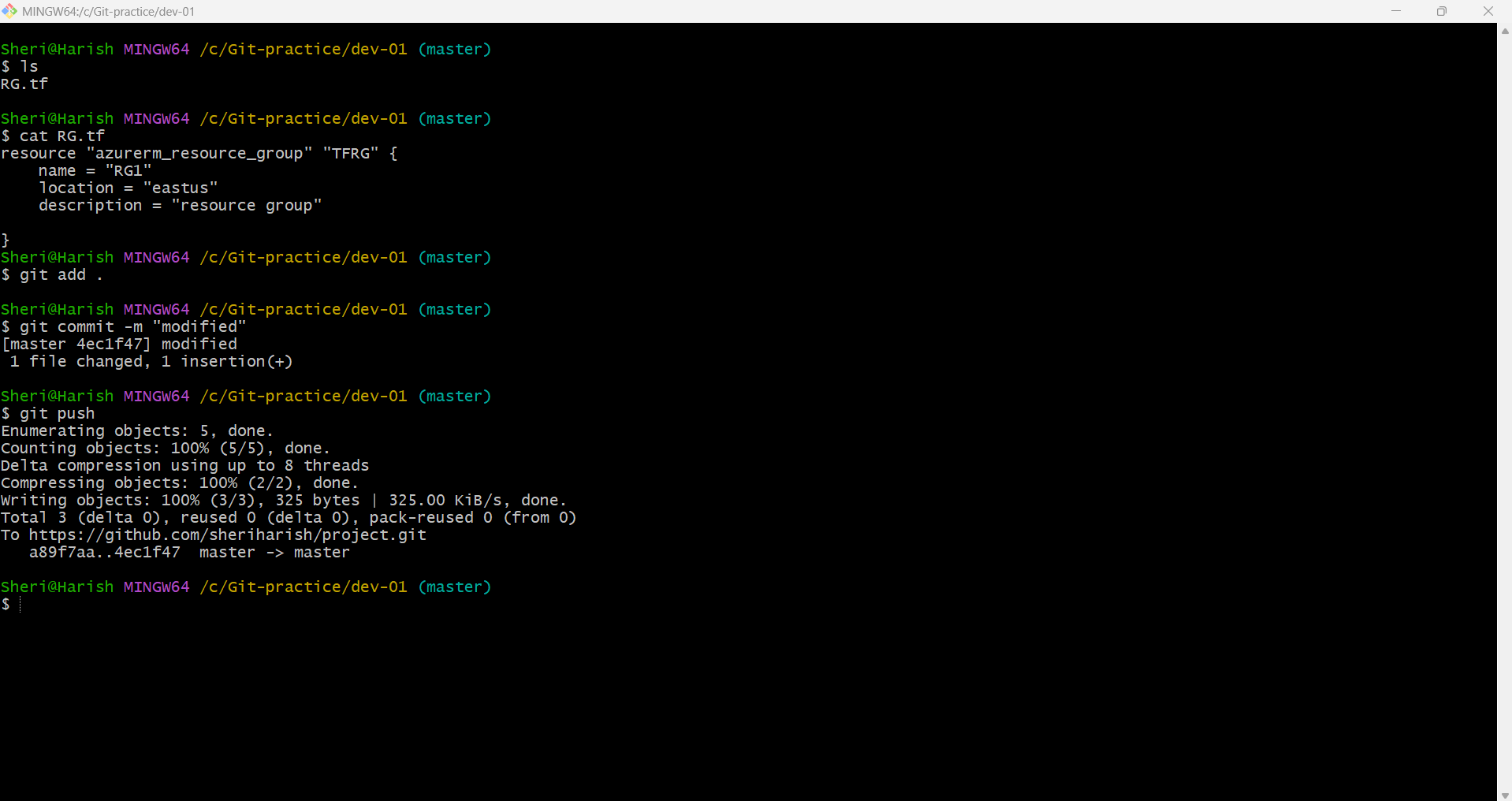




**Step4:** Create another directory or folder (Dev-02) and pull code from the same repository (project) into the Dev-02 folder.



**Step3:** Now add the content on the line 4 of terraform configuration file and then push it into the repository (project) from Dev-01 folder.



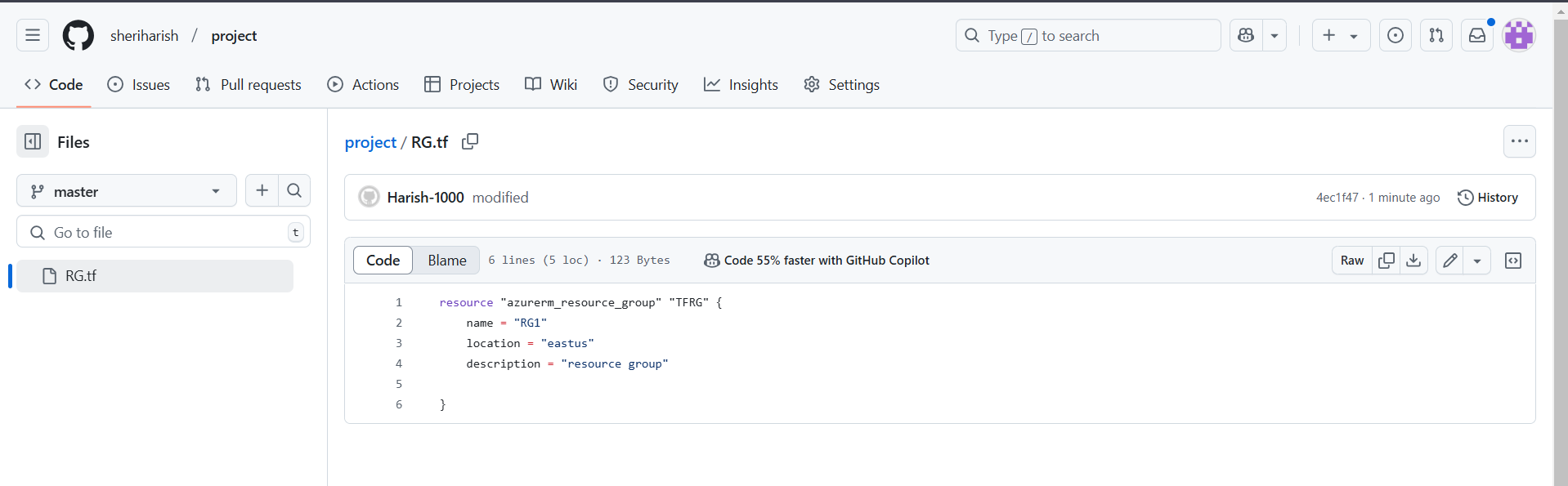
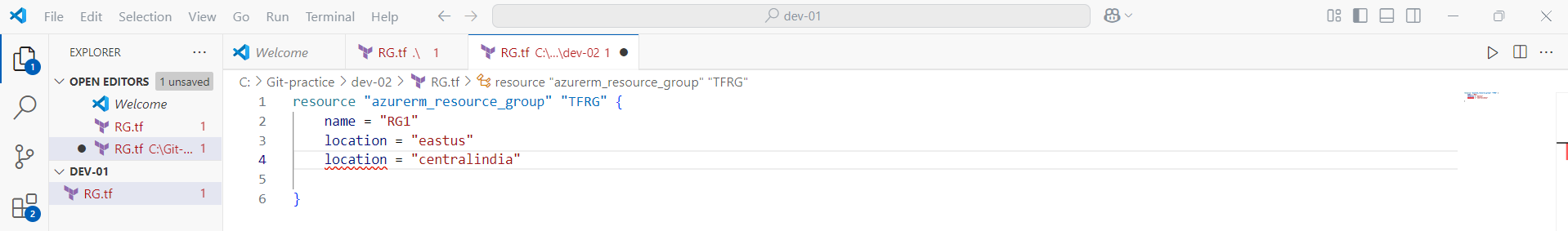


Fig: New line is added at line number 4

Step4: At the same time the developer 2 (Dev-02) is modified the same line 4 with different content.



When Developer 2 (Dev-02) trying to pull the existing content from the repository (project) it will show some error message as shown below figure.

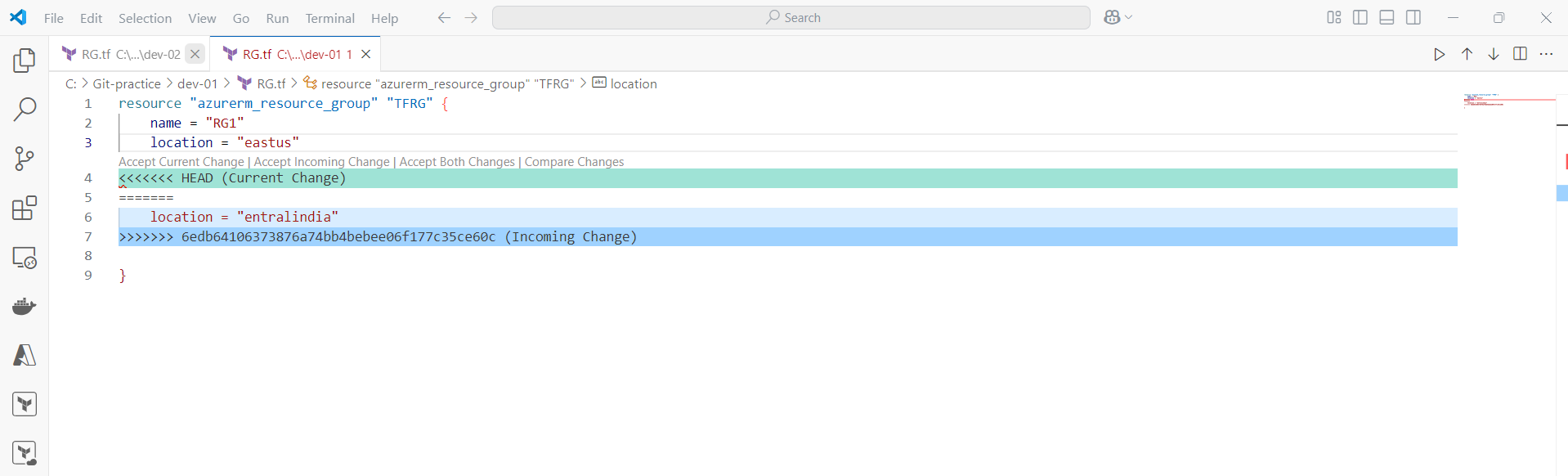
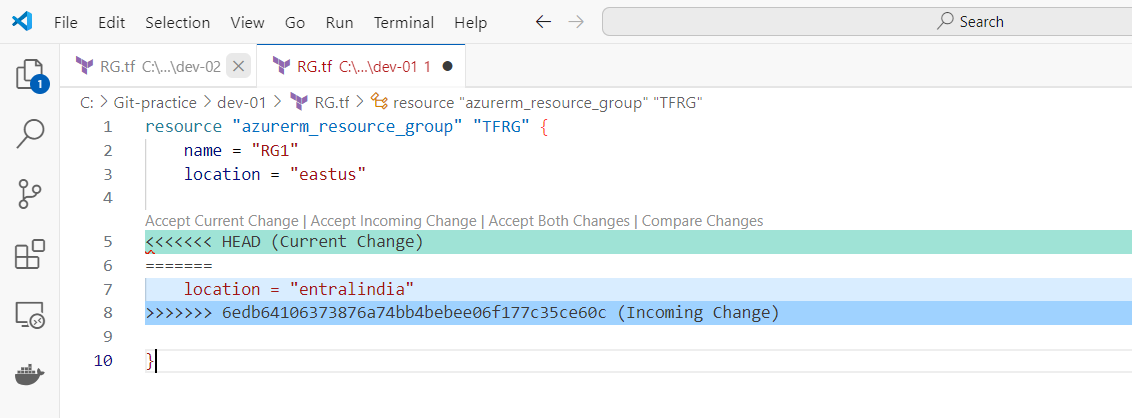


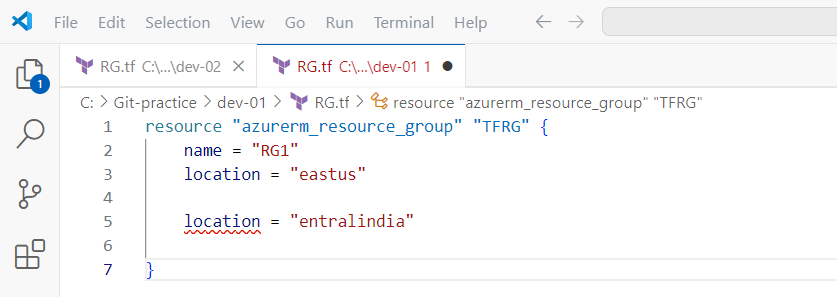
Fig: Merge conflict.

The above merge conflict is happened when both the developers (Dev-01 & Dev-02) are trying to modify or edit the same line of the terraform code.

The Git will ask for which change is to be done either “**incoming change”** or **“current change”** or “**accept both changes”** as shown in above figure.

If Accept both changes then both modifications are pushed into the repository.

****



Then do the “**git rebase –continue”** command, it is used to **resume an ongoing rebase process** after resolving conflicts. It tells Git that you have manually fixed the conflicts and want to continue rebasing the remaining commits.

Next do **git commit** command and the **git push command** so that the files are updated in the remote repository successfully.

**git rebase –continue 🡪 git commit command 🡪 git push command.**

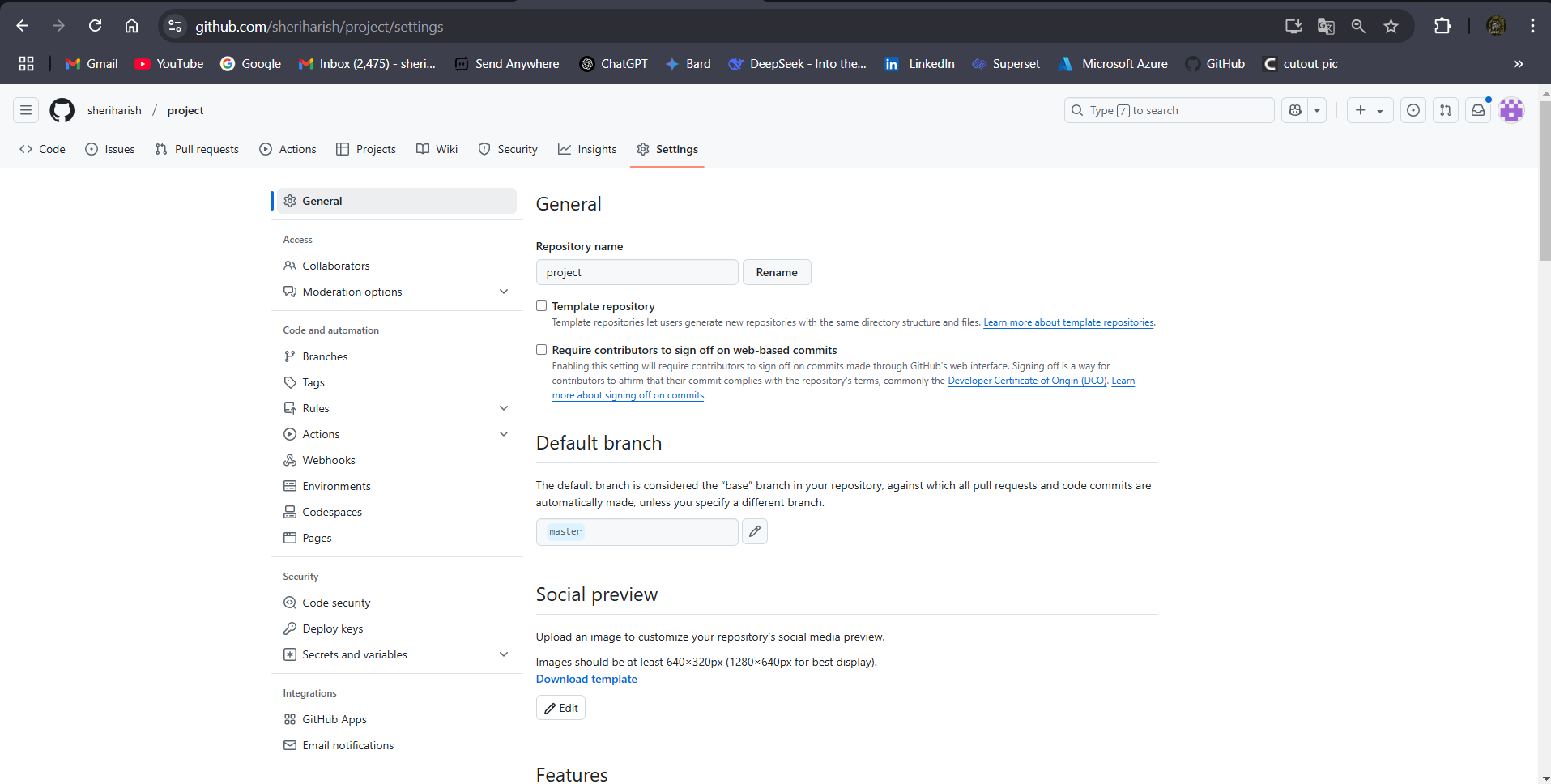
**Using of GitHub in Ubuntu VM**

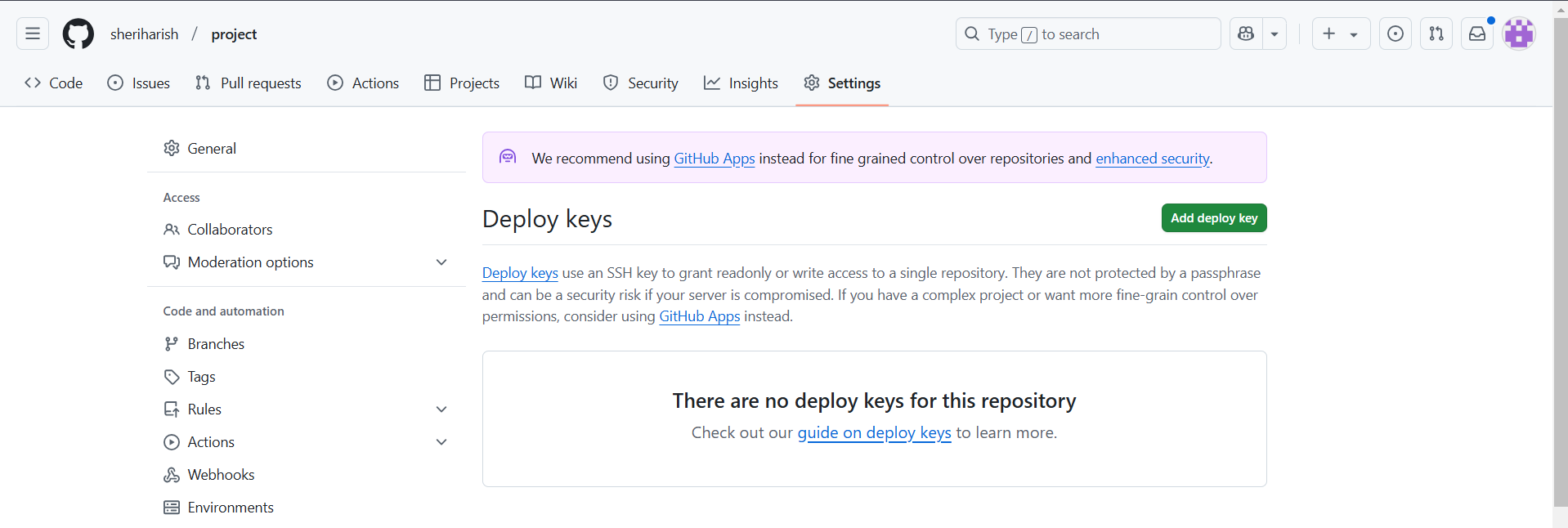
Generally in windows machine we authenticate the GitHub using password or personal access token (PAT). Where as coming to Ubuntu machine we use the access keys in order to authenticate with GitHub.

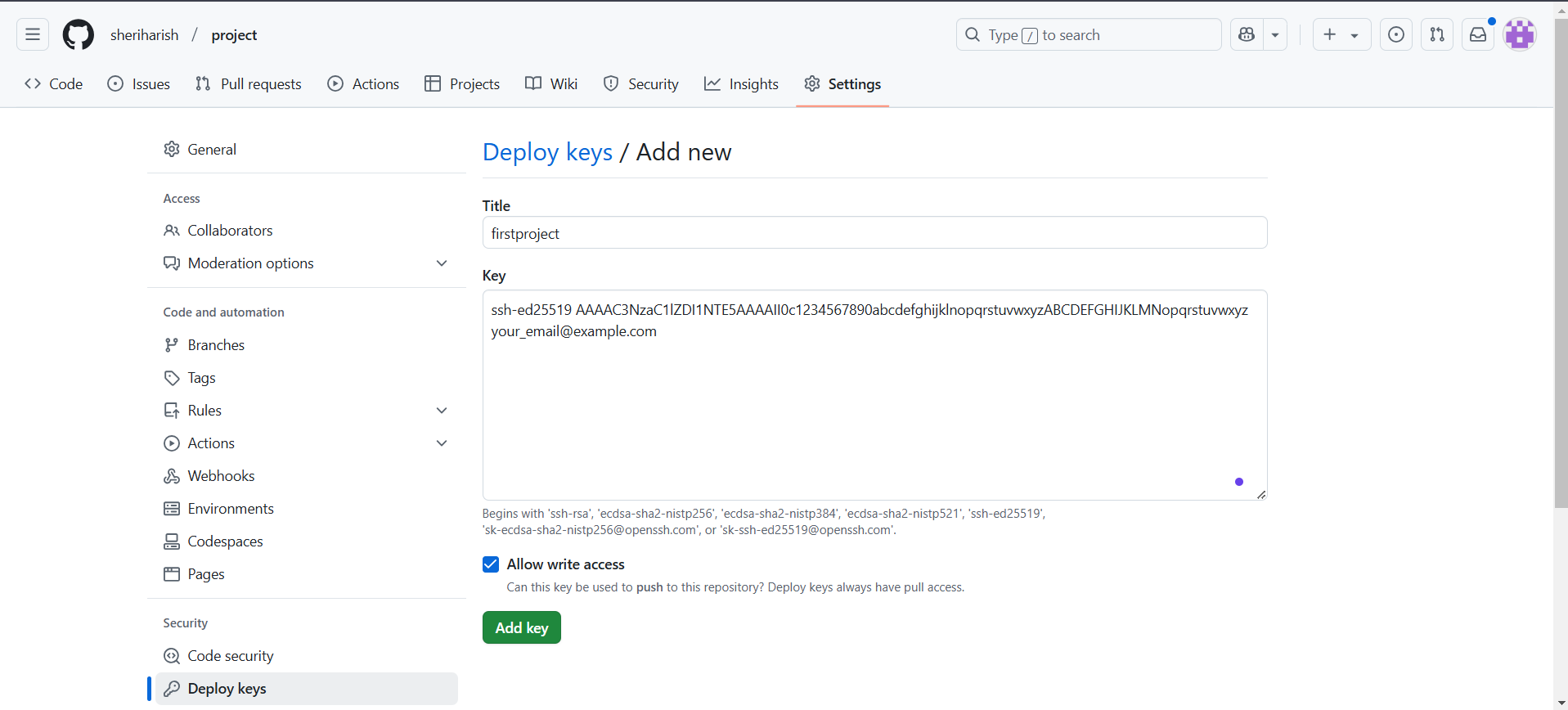
**Note:** In Ubuntu or Linux machine Git is installed by default. We no need to install in again.

The process of authentication in Ubuntu VM:

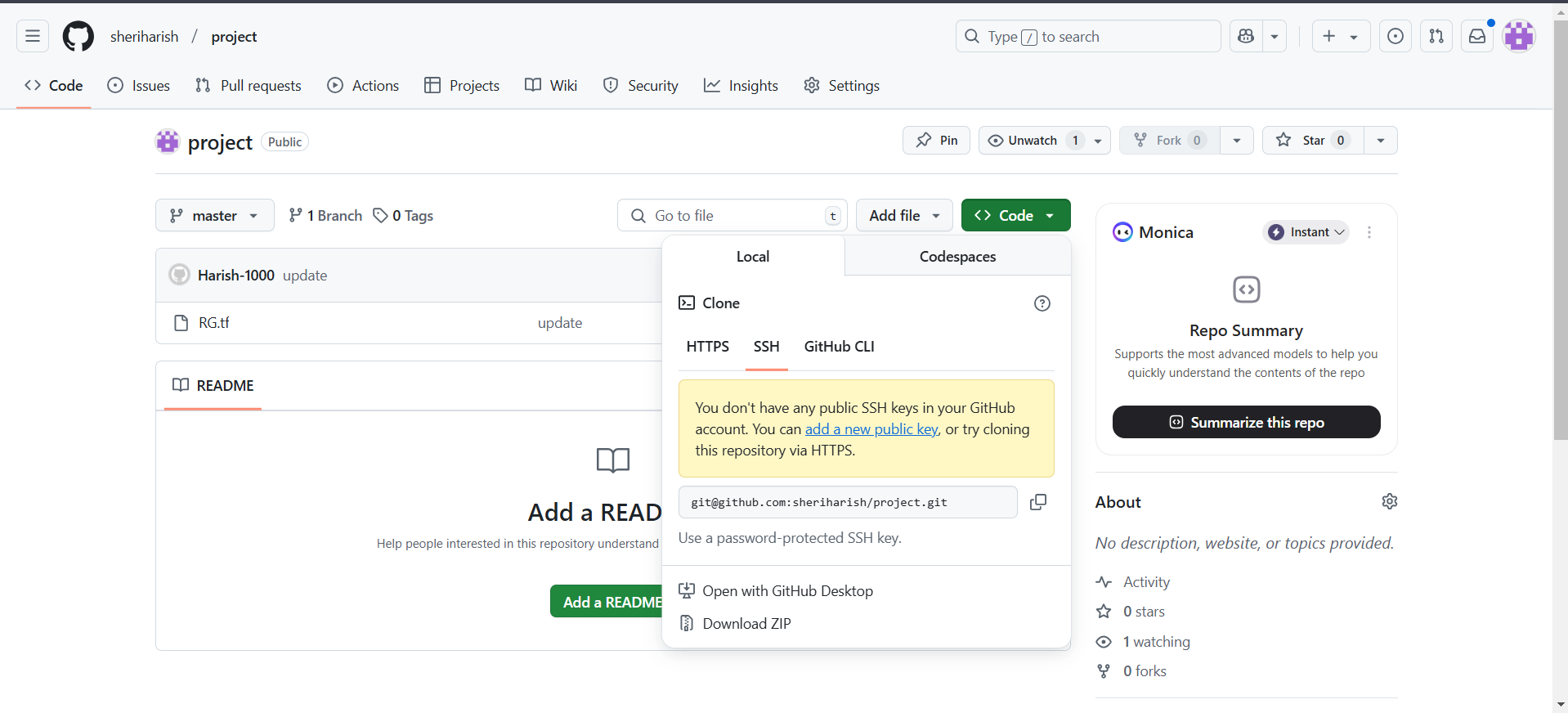
1. In Ubuntu terminal use the command **“ssh-keygen”** it will generate the key in the path of **“/root/.ssh/id\_ed25519.pub”.**
2. And to view the key use the command **“cat /root/.ssh/id\_ed25519.pub”**
3. Copy this key and **deploy key** in the current working repository of GitHub.







**Note:** While cloning the remote repository from git to Ubuntu machine instead of using HTTP link use SSH link.



After authentication very think is same whatever we performed in the windows machine are can be performed in Ubuntu machine also.