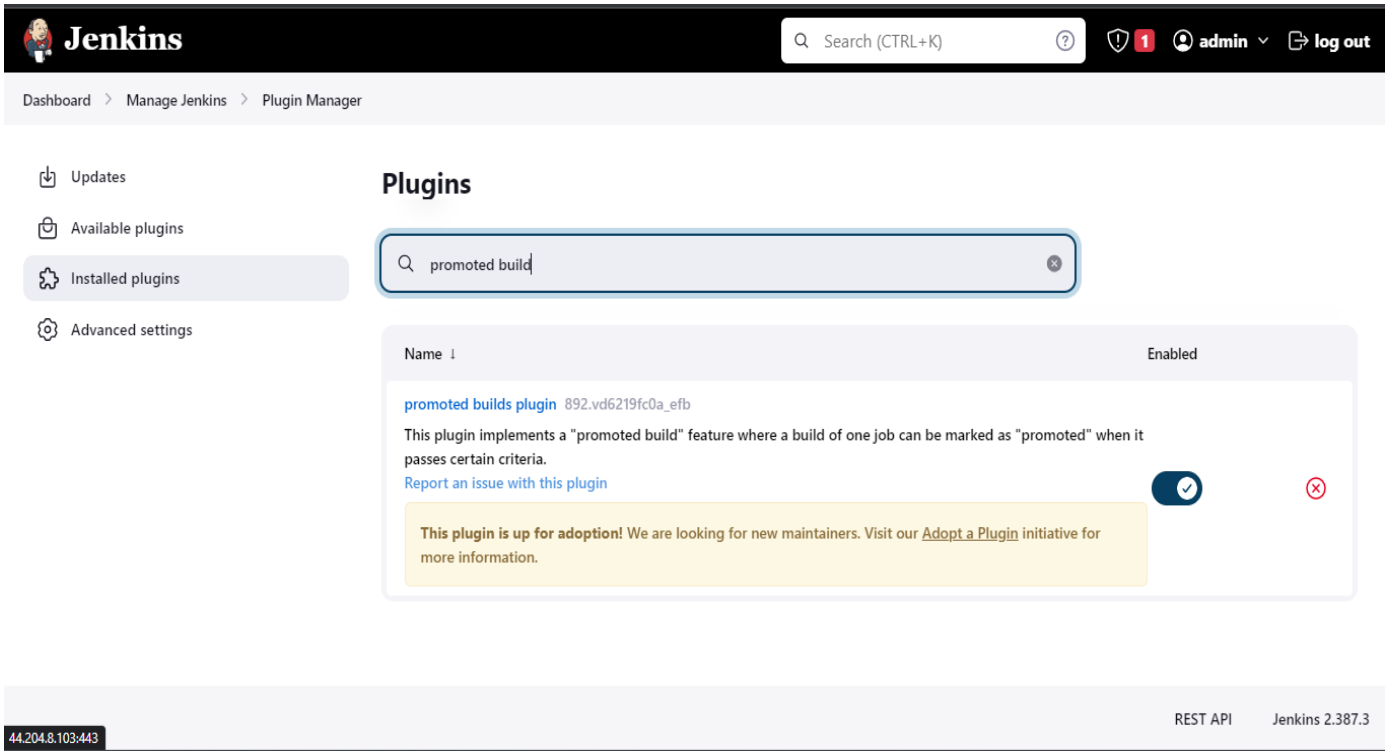
1. Promotion build plugin - scenario

The "Promoted Builds" plugin in Jenkins is used to add manual or automatic promotion processes to your build jobs. It allows you to define criteria for promoting a build, such as passing specific tests, meeting certain quality metrics, or going through a manual approval process.

Here's how you can use the Promoted Builds plugin in Jenkins:

Step1:

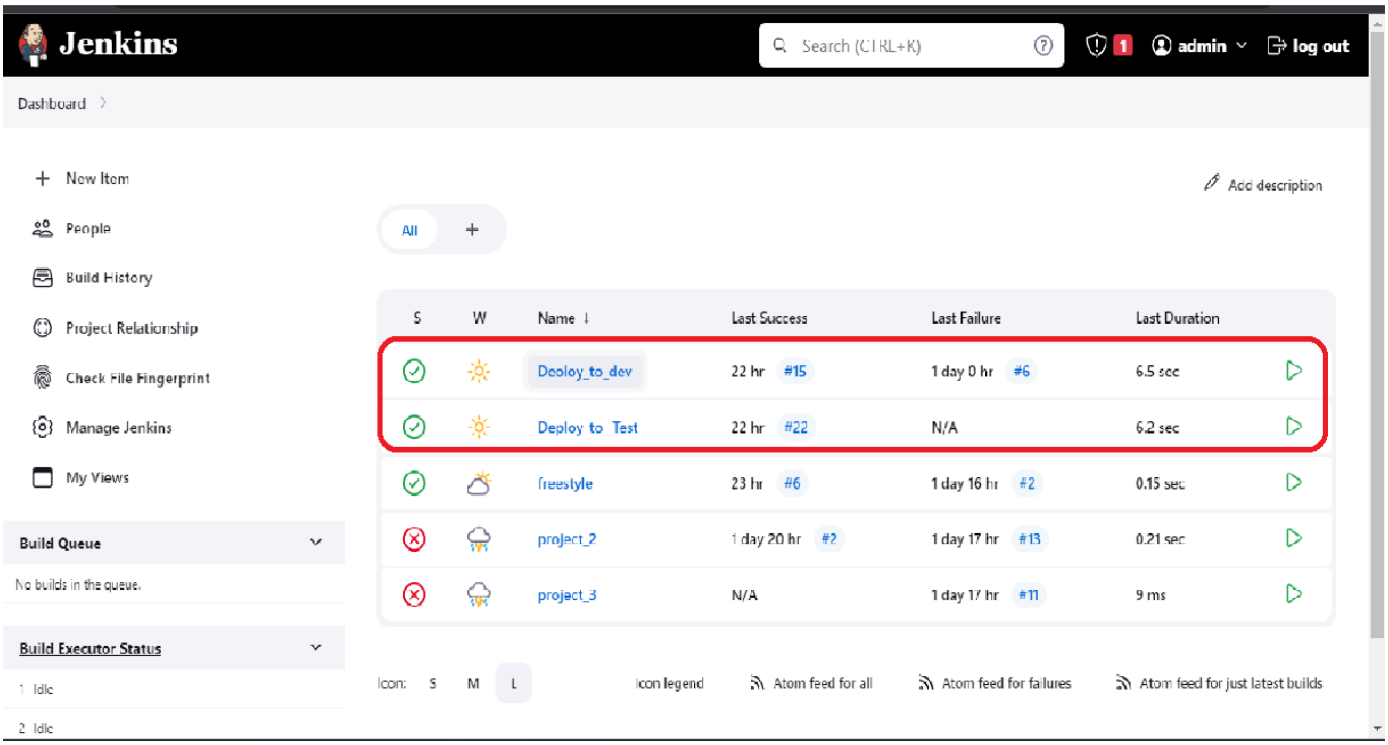
Install the Promoted Builds plugin: Go to Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available tab -> Search for "Promoted Builds" -> Select the checkbox next to it -> Click on "Install without restart" or "Download now and install after restart".



Step2:

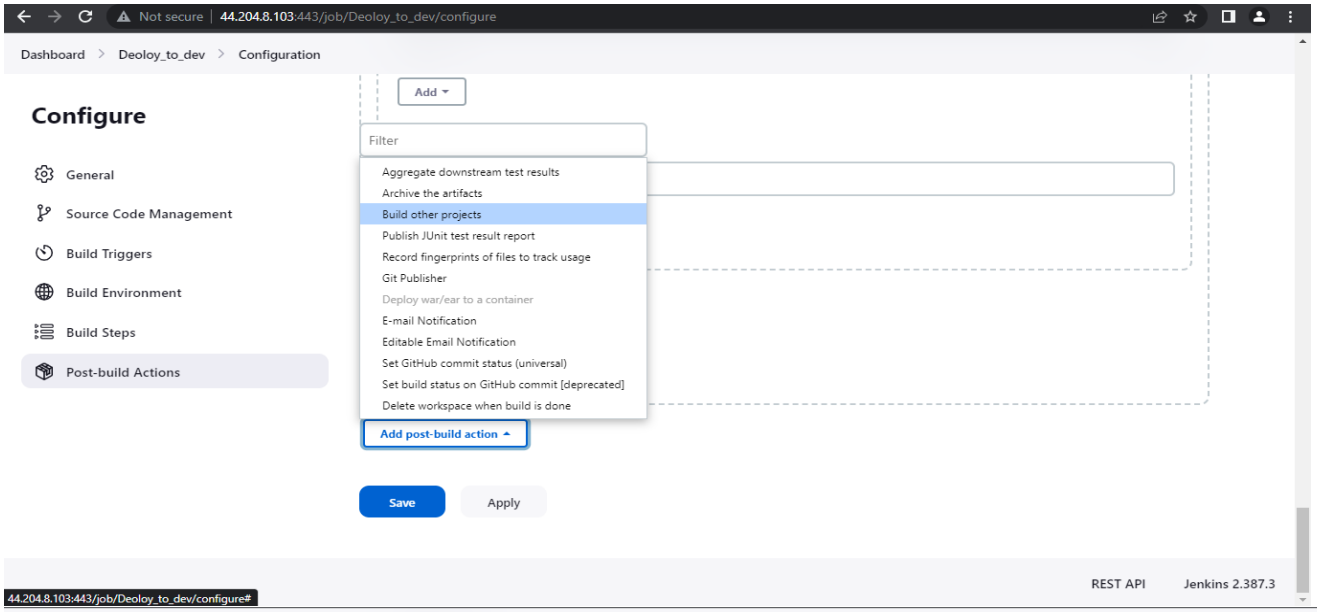
Configure a job with promotion criteria: Open the Jenkins job you want to configure promotions for.

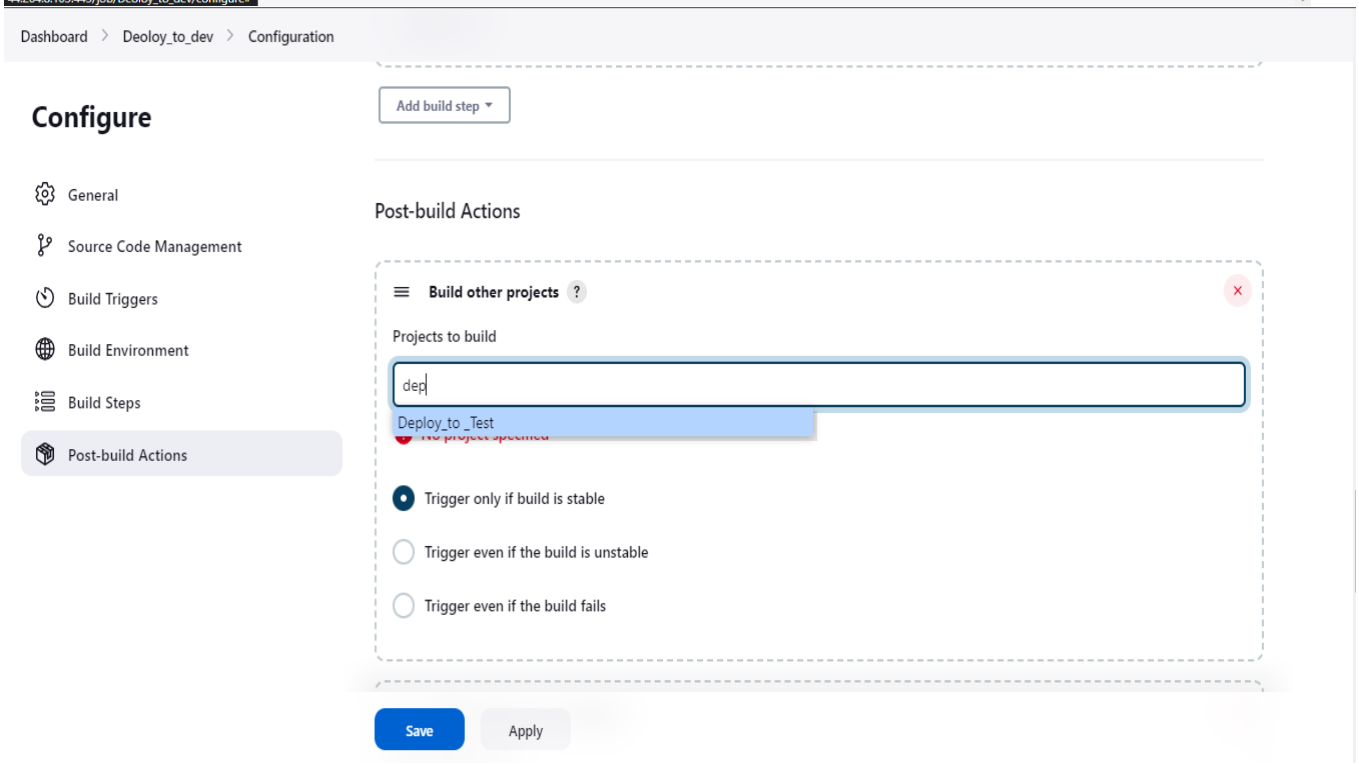
So, I have already created 2 free style jobs “Deploy_to _dev and Deploy_to _Test”.



Step3:

Add the promotion process: Scroll down to the "Post-build Actions" section and click on "Add post-build action". Select "Promote builds when..." from the dropdown menu.

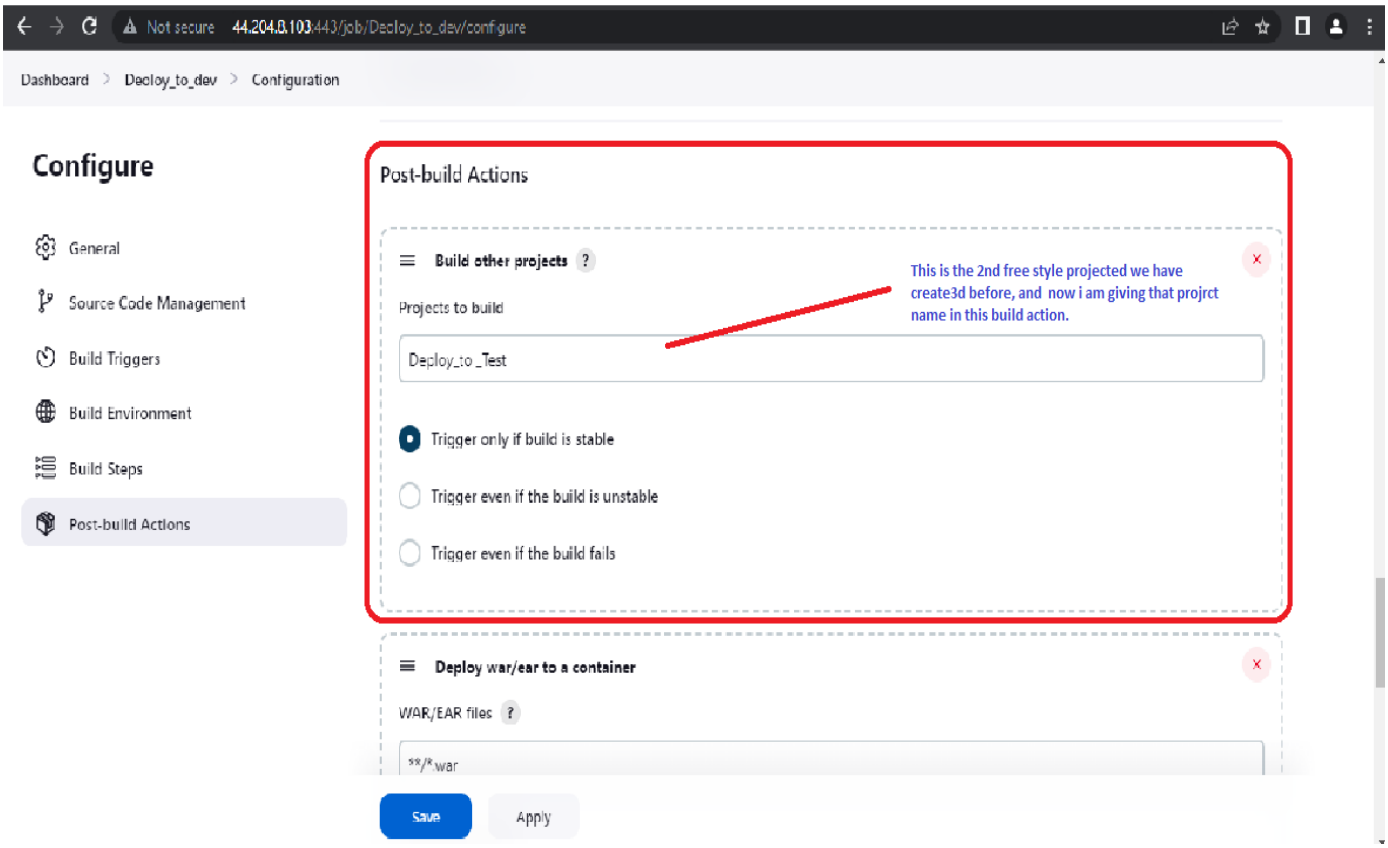




Step4:

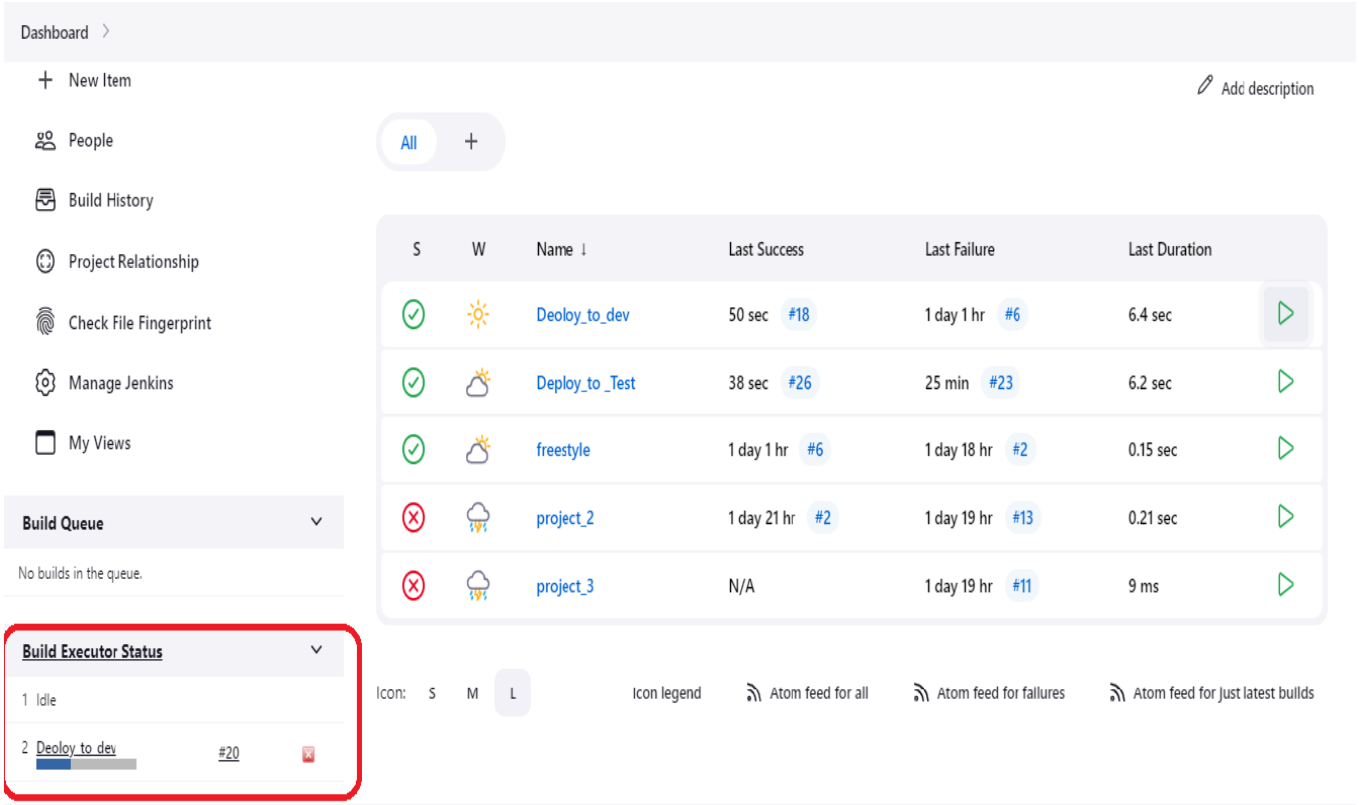
Define promotion criteria: In the promotion configuration, you can specify the criteria that need to be met for the build to be promoted. For example, you can add conditions like "Build stability" to check if the build is stable or "Conditional steps (single)" to add custom conditions using Groovy script.

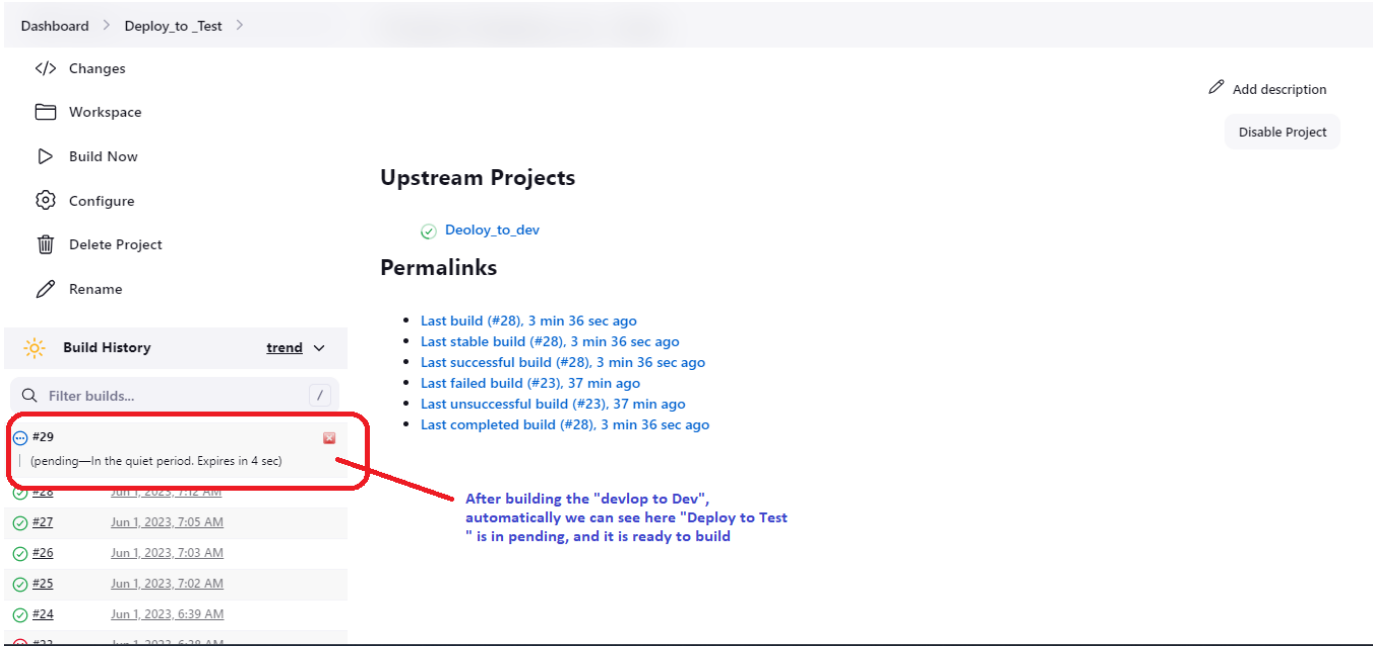
Save the configuration: Click on "Save" to apply the promotion configuration to the job.



Now, when the build completes, Jenkins will evaluate the defined criteria, and if they are met, it will trigger the configured promotion actions. The promotion process can be automated or require manual approval, depending on how you configure it.

You can also view the promotion history and manually trigger promotions for specific builds from the Jenkins job page.





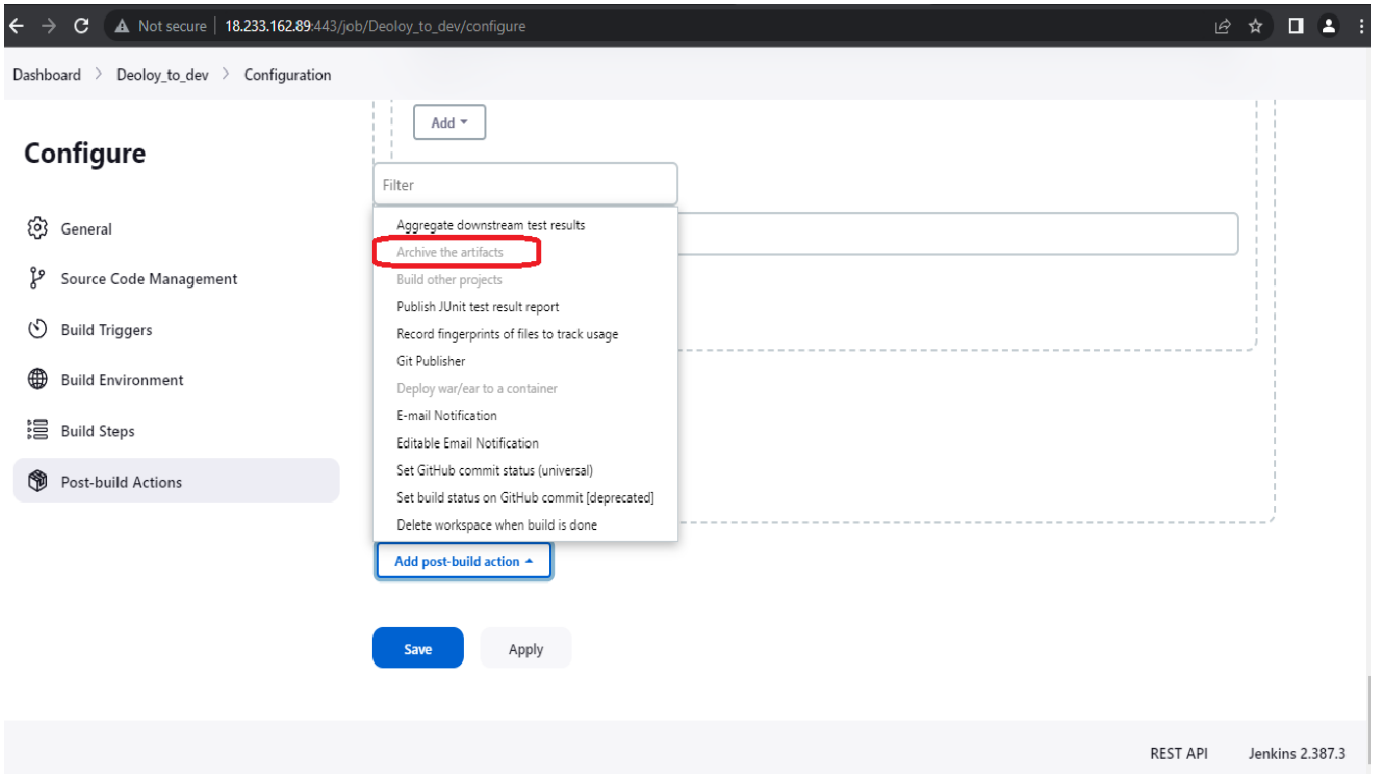
This is how **Promoted Build Plugin** Works...

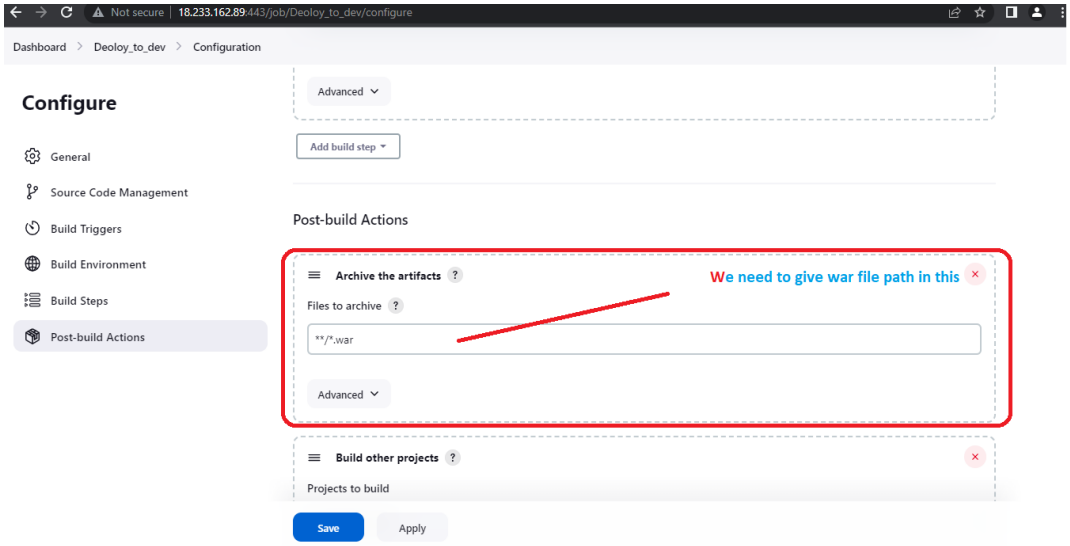
2. Using Jenkins deploying the code and executed the code through Dev to Test and Test to Production environment.

As we have done till now.....In the before task, same thing to be followed and from now we need to add some post build action, then we ll get the required output.

Step1:

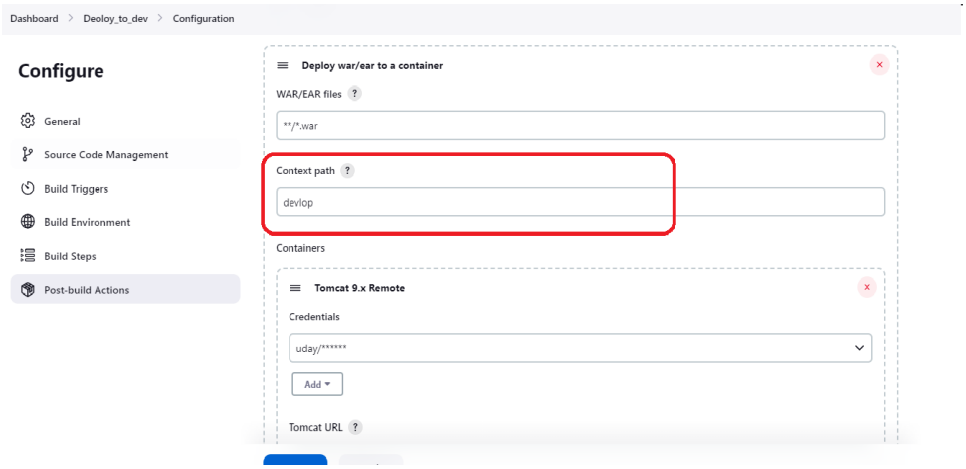
Go to “deploy_to_dev” and configure options, then at the bottom we find Add Post-Build Action, we need to add the Archive the Anticraft..





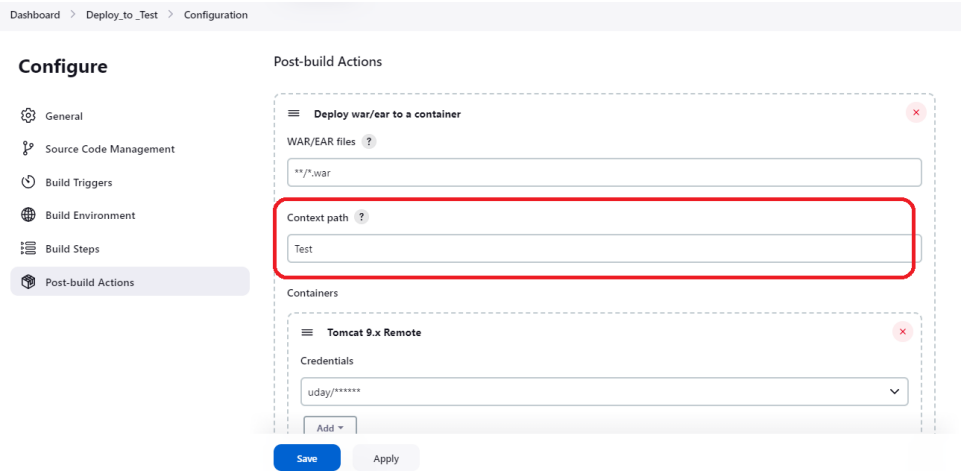
We need to give war file path in this

Need to give the Content Path..



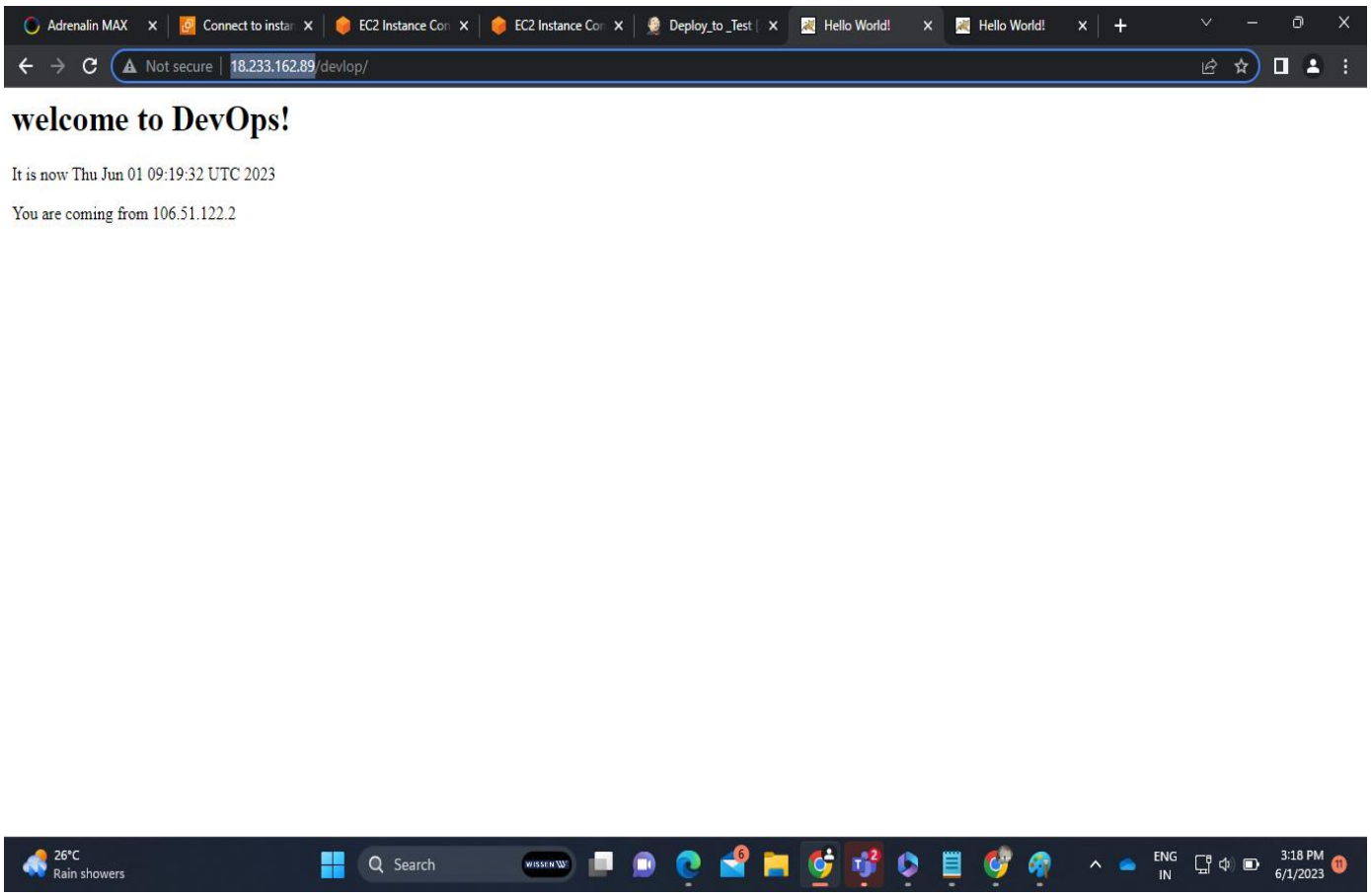
Step2:

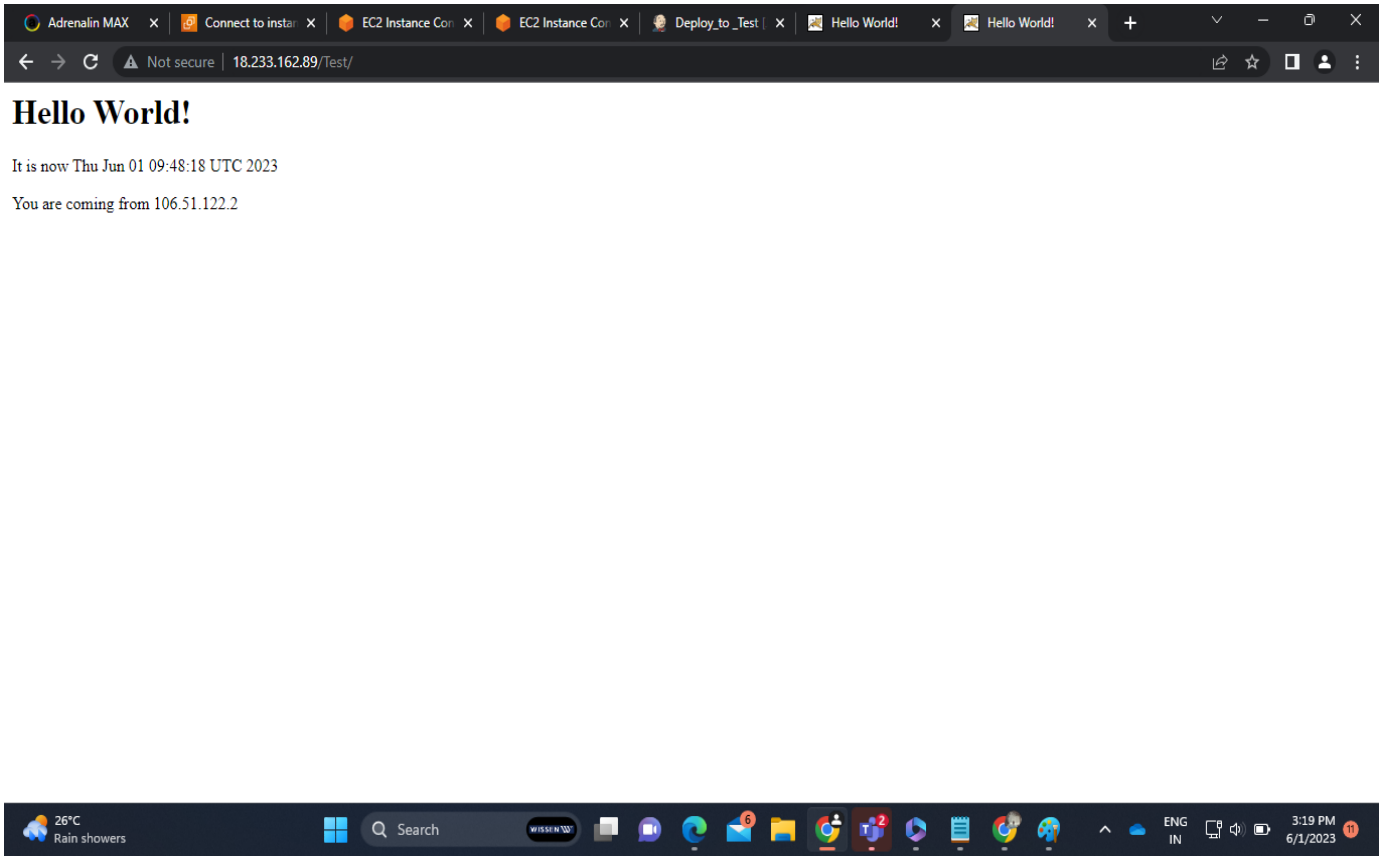
We need to do the same thing in the other project “Deploy_to_Test”. Giving war file Path . And need to give the content path “Test”..



Step3:

Now build both the projects..

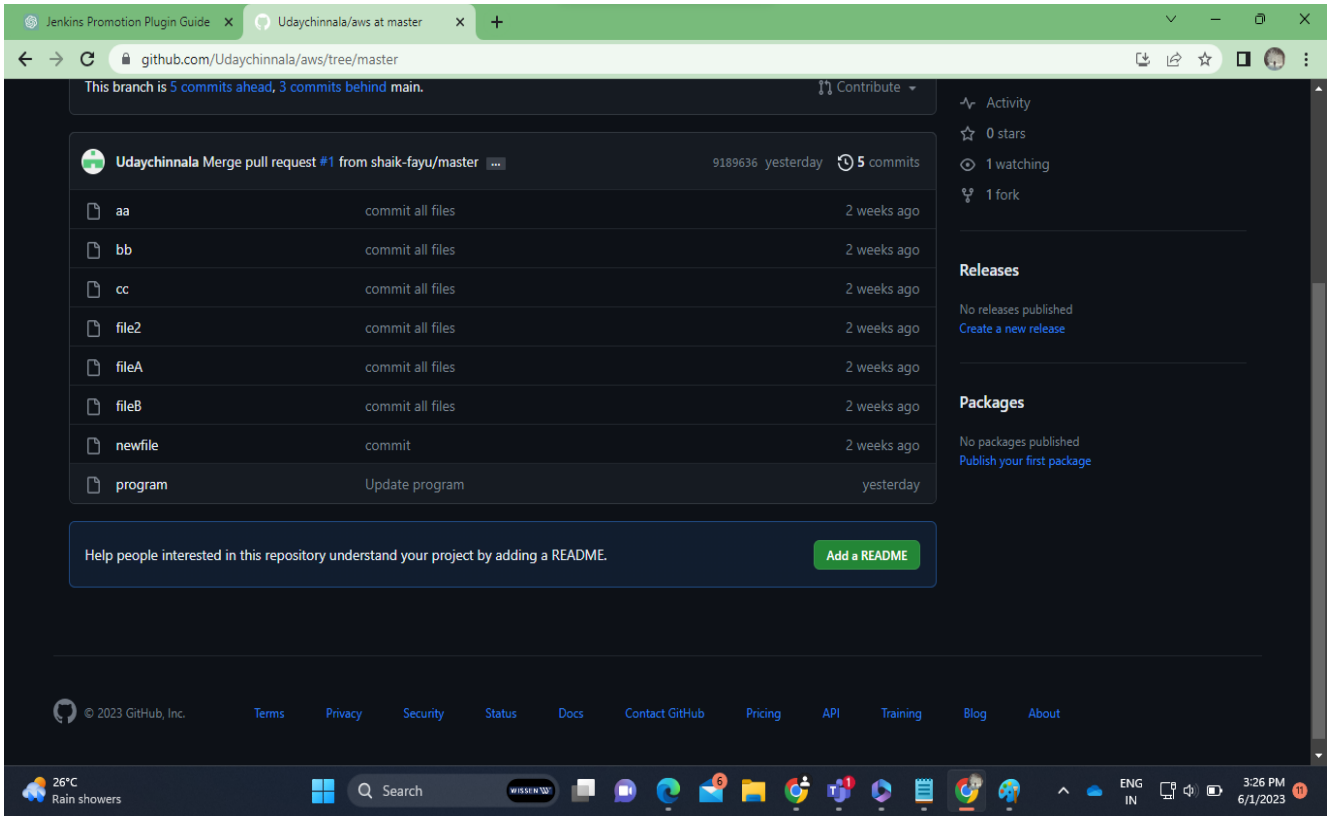




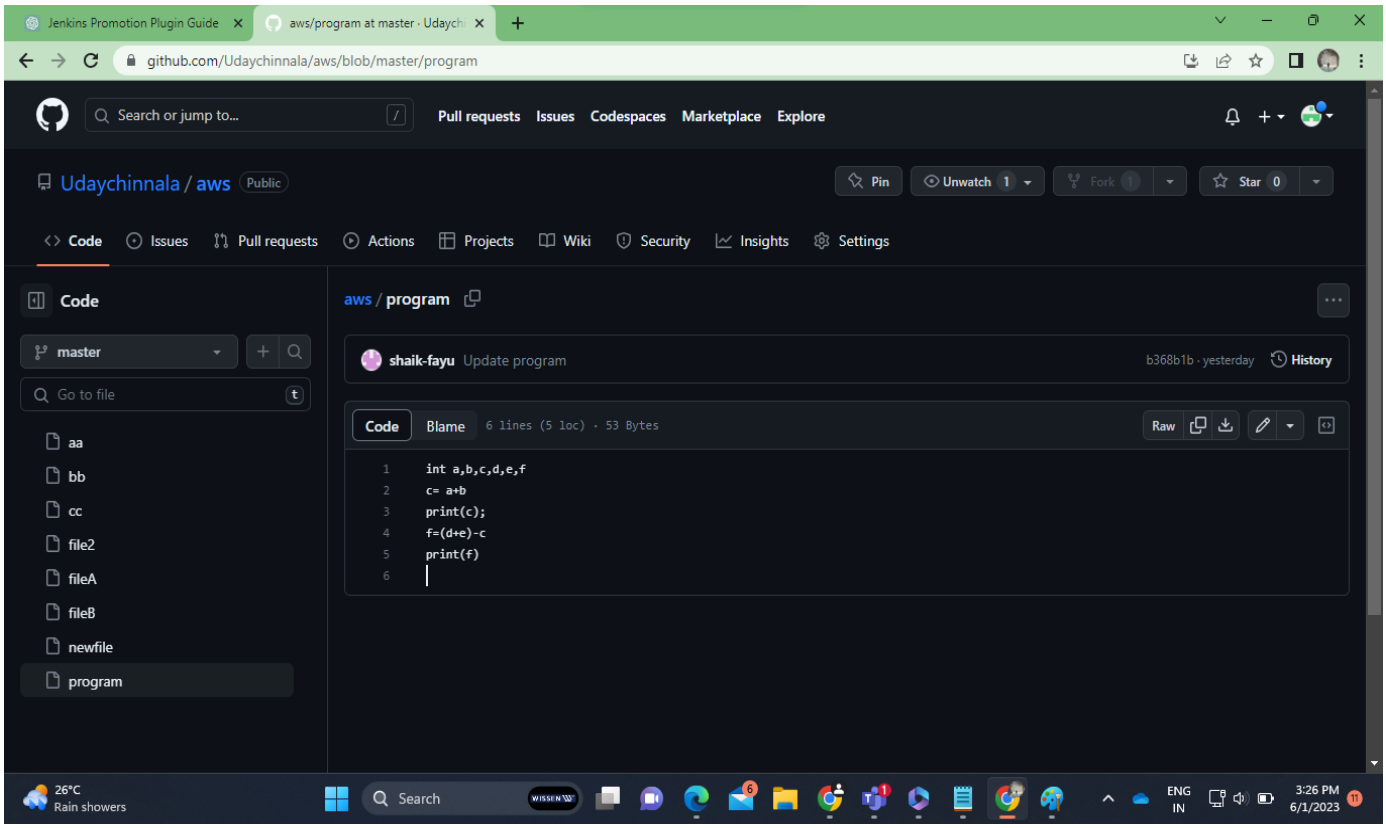
3.PULL Request

Firstly , lets create a text file, and write a code..(named as Program)..

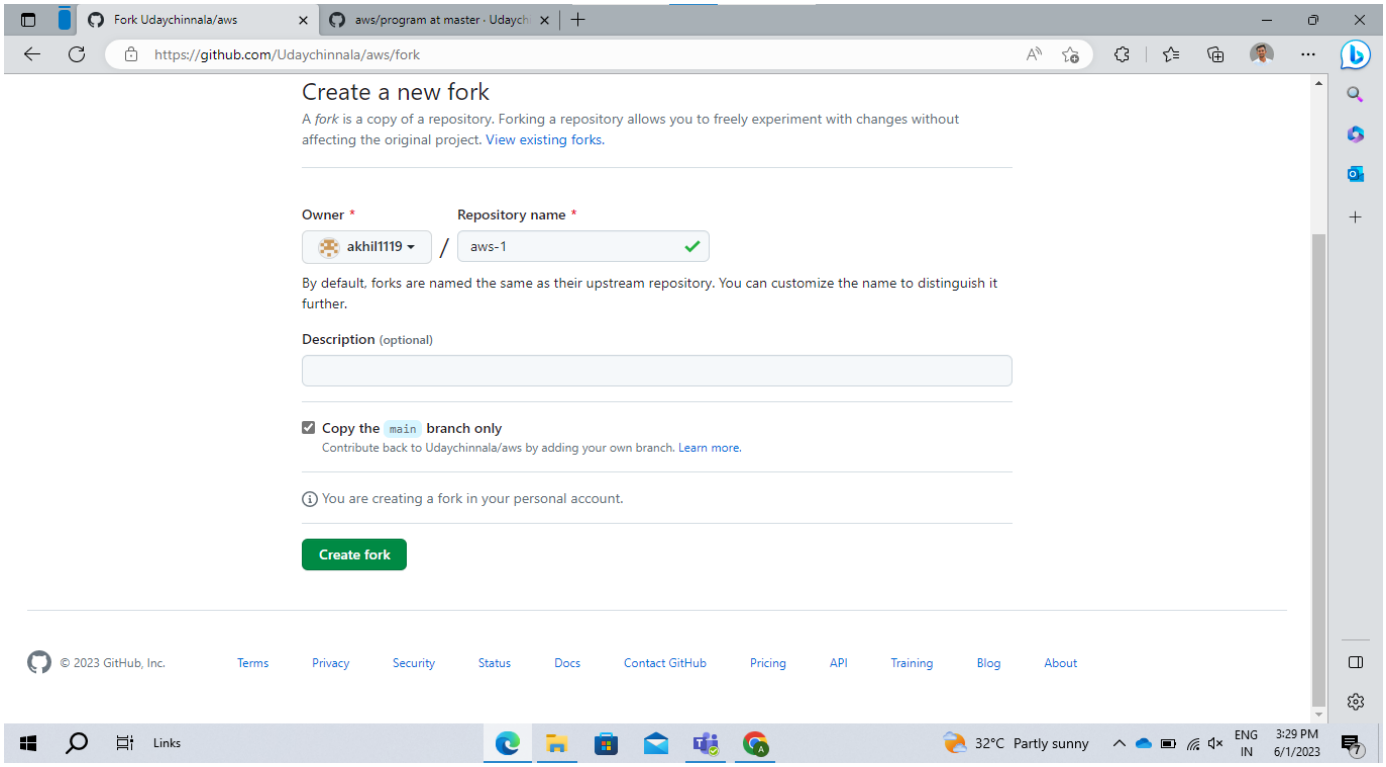
Emp Id:6736 Uday Kumar



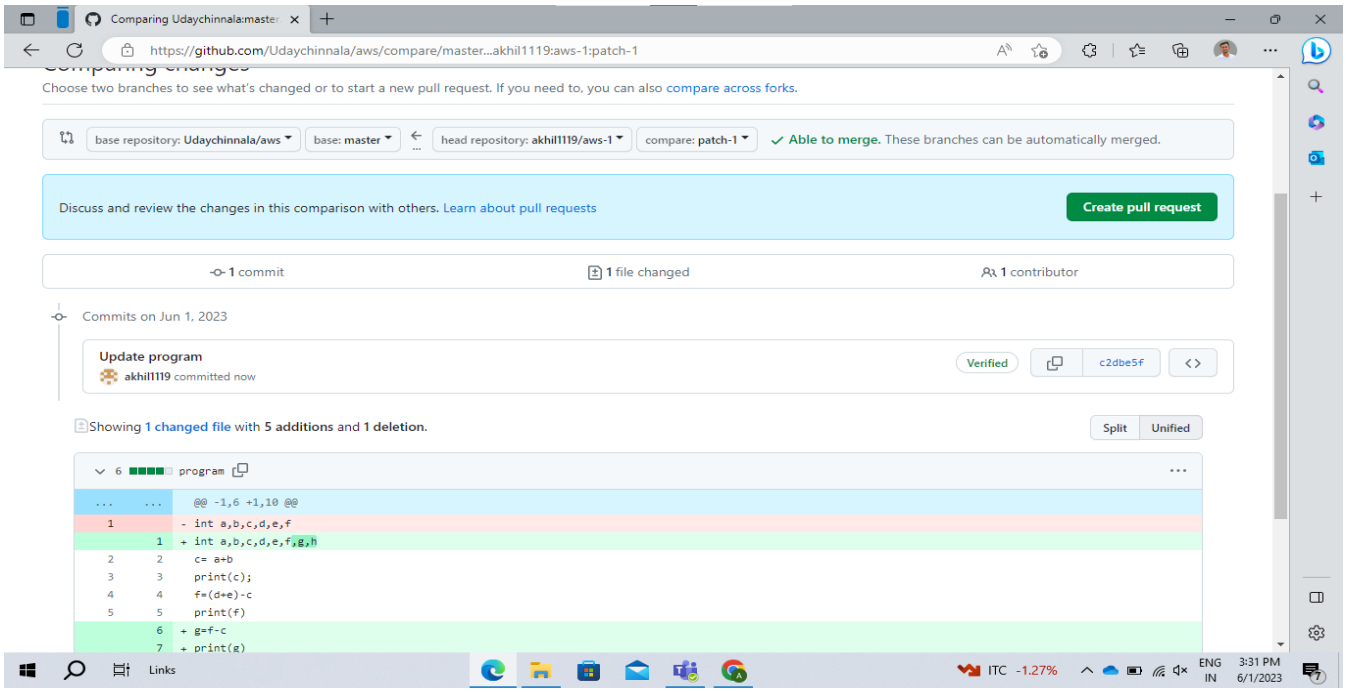
This is the code in the Program file..



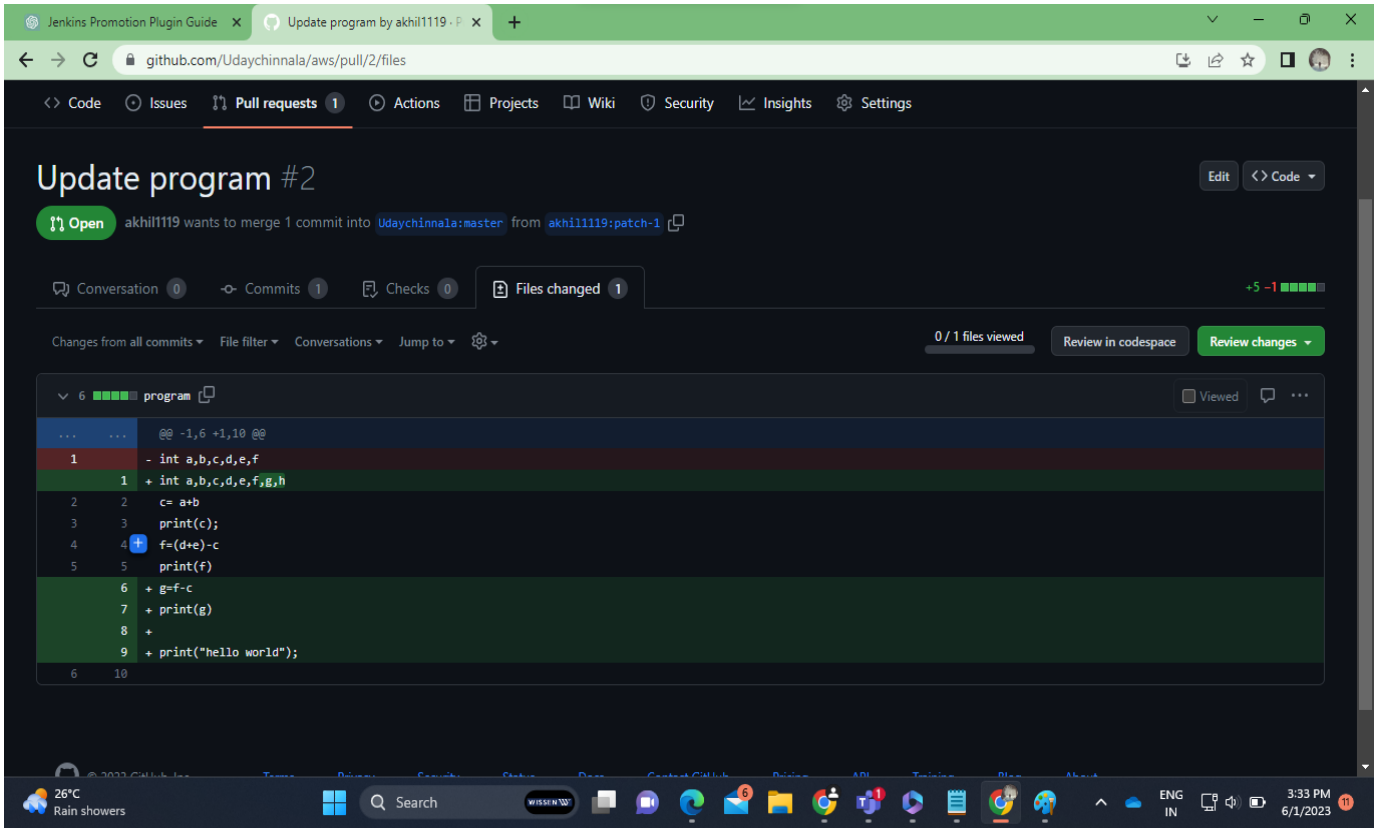
Copy the link and open with another Git-Hub Accout..and fork the code into the git-hub Account .



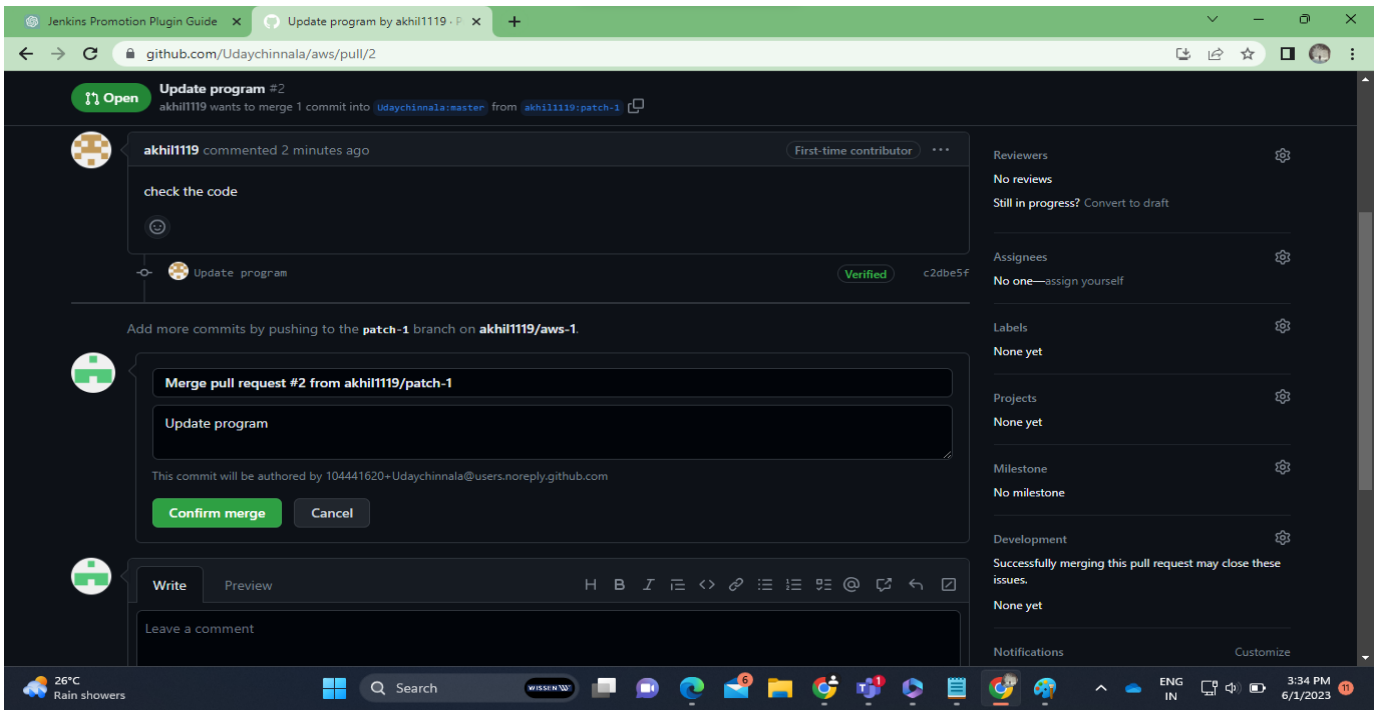
Do some changes in the file and commit the changes...then we ll get a pull request.



Now we can see the changes made on another account, and green colour indicates the new code commits, and red indicates what we have done..

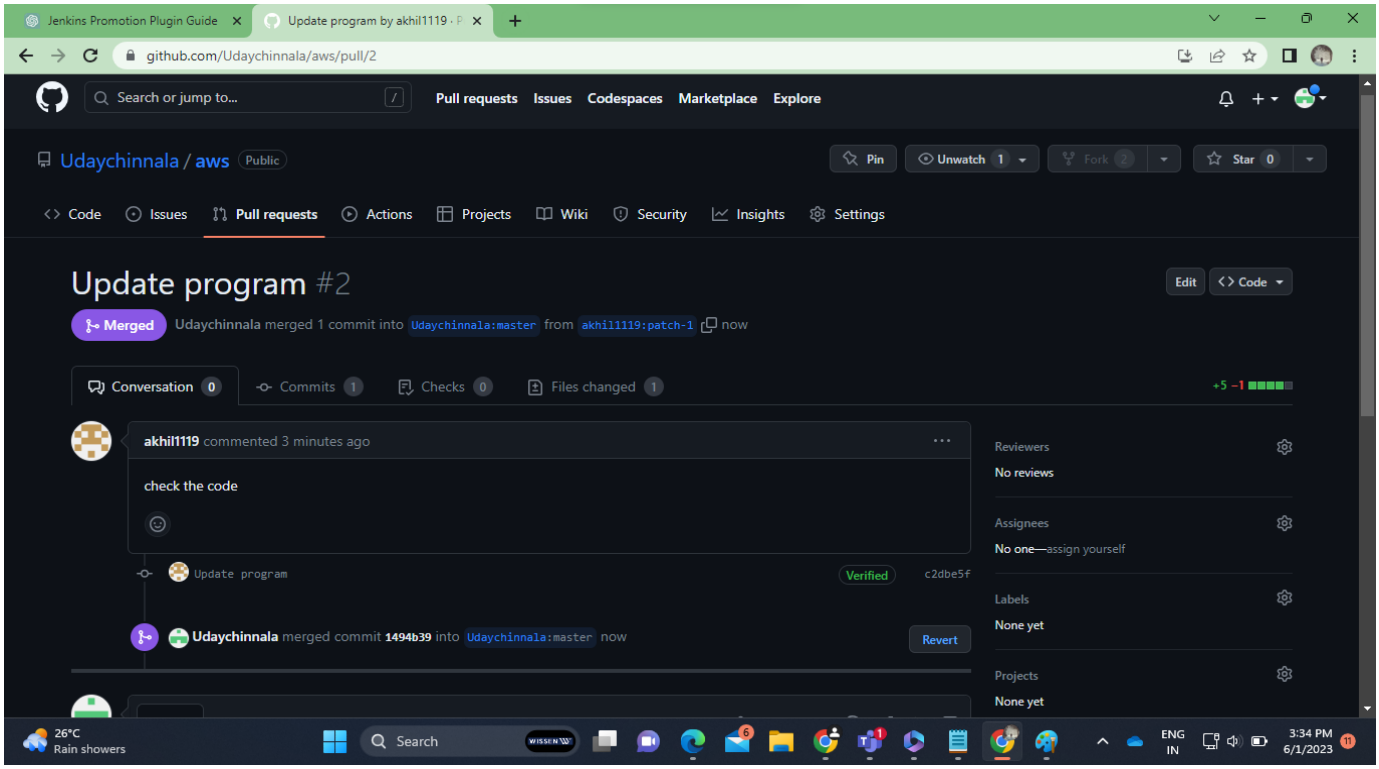


If everything is proper, then we can merge the changes ..so for that we need to go with conform merge.



Now it has successfully merged into our code.

Emp Id:6736 Uday Kumar



Here we can see the latest code that we have Merged