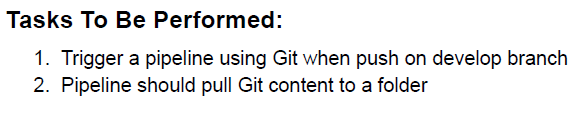
**Jenkins Assignment – 1**

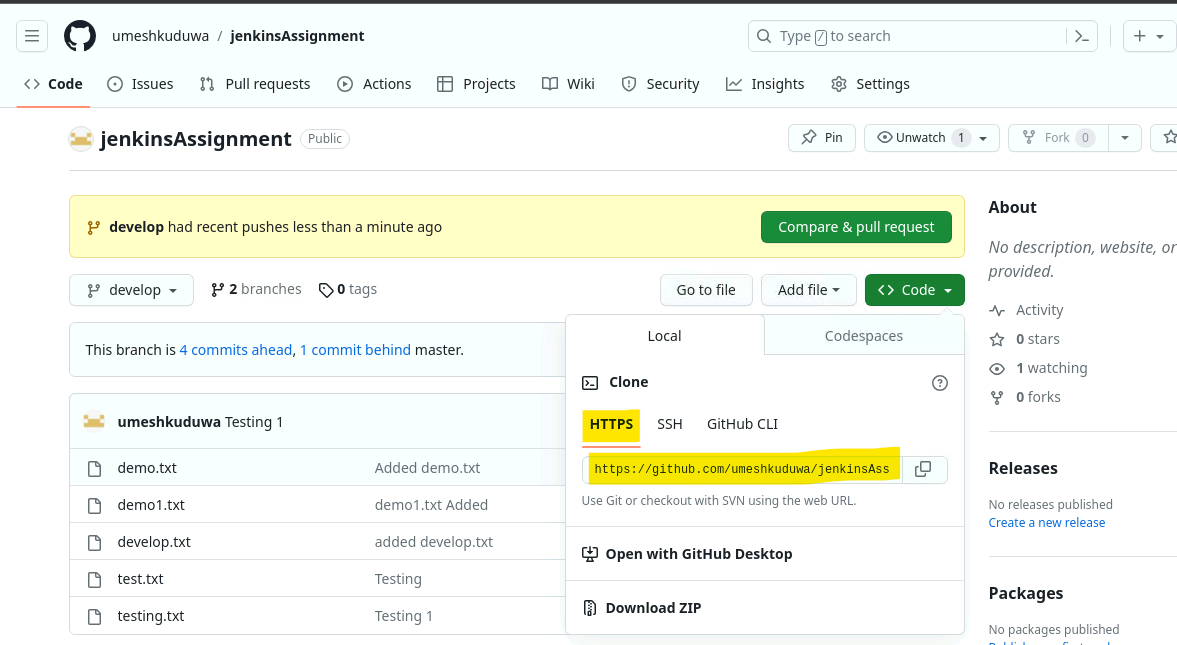


Tasks To Be Performed:

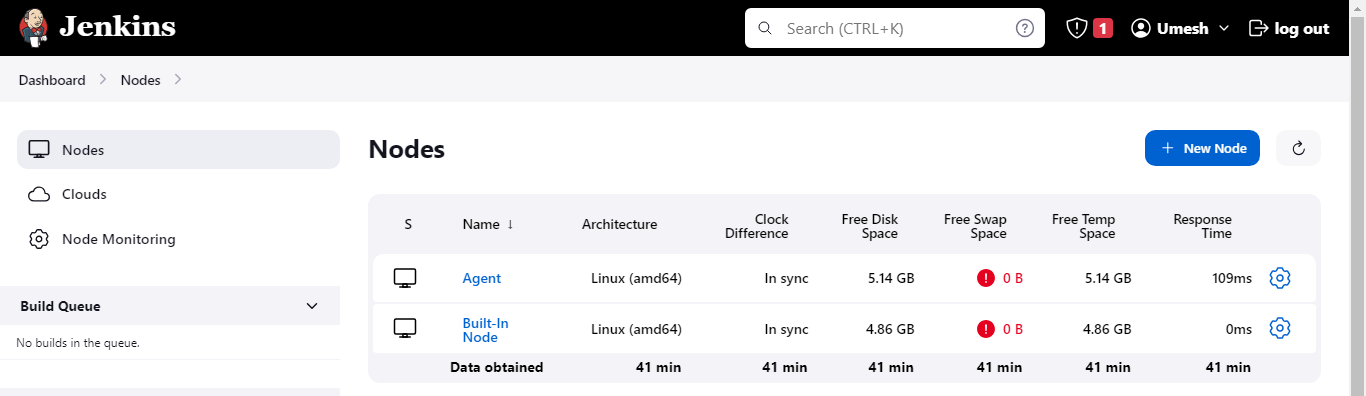
1. Trigger a pipeline using Git when push on develop branch
2. Pipeline should pull Git content to a folder

Solution:

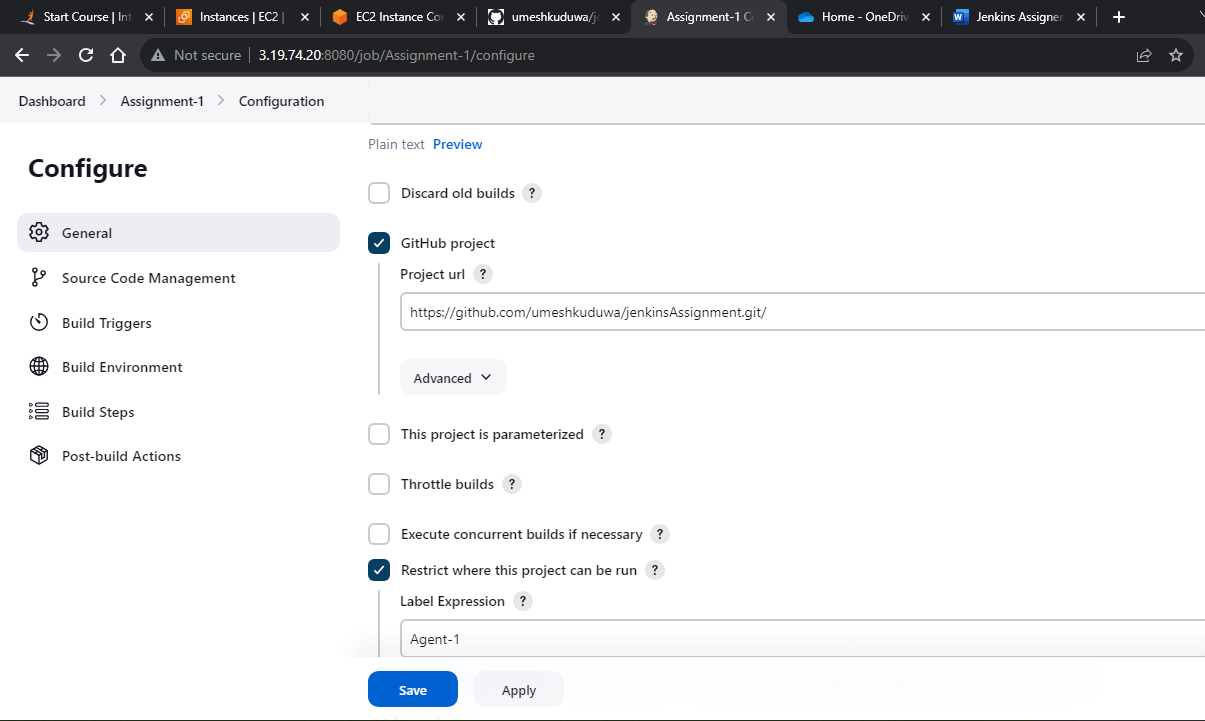
1. Create a Github Repo and Copy the URL to the Repo.



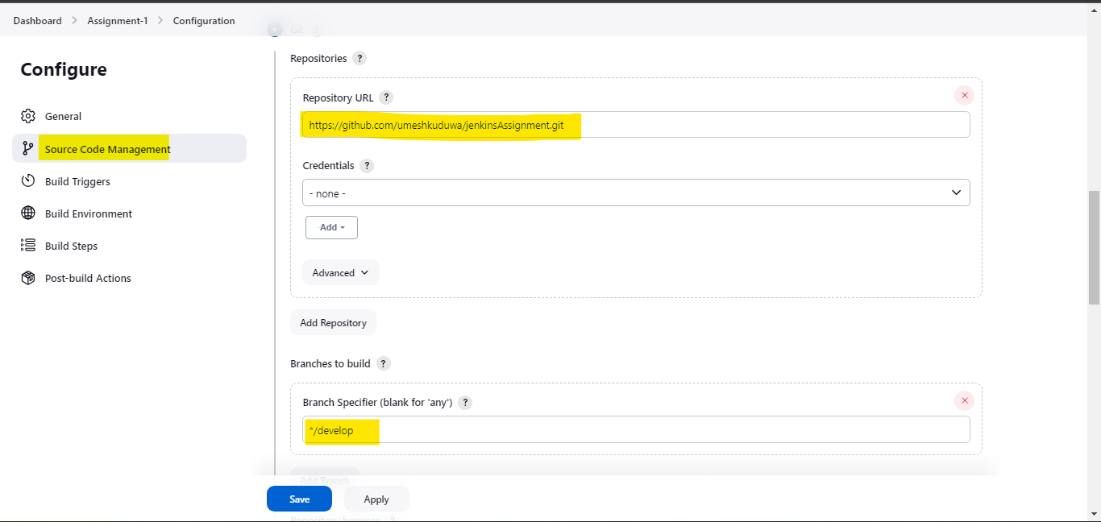
1. Setup the Jenkins Master and Node. Install Jenkins and Java on Master node, Java on Slave node. After setting up, add the node into jenkins. Jenkins would be accessible via <http://public-ip-of-master-node:8080>



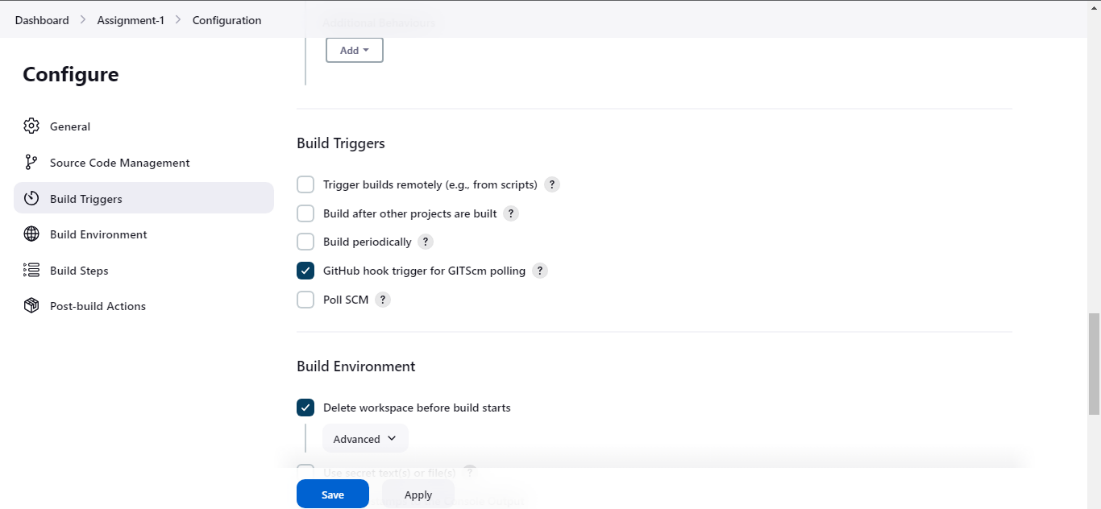
1. Create a New Job. Add the above copied repo URL into Jenkins Job. Check the Restrict where the project can be Run checkbox, and search for the slave node label. So that everytime the Job executes it will run on Slave Node.



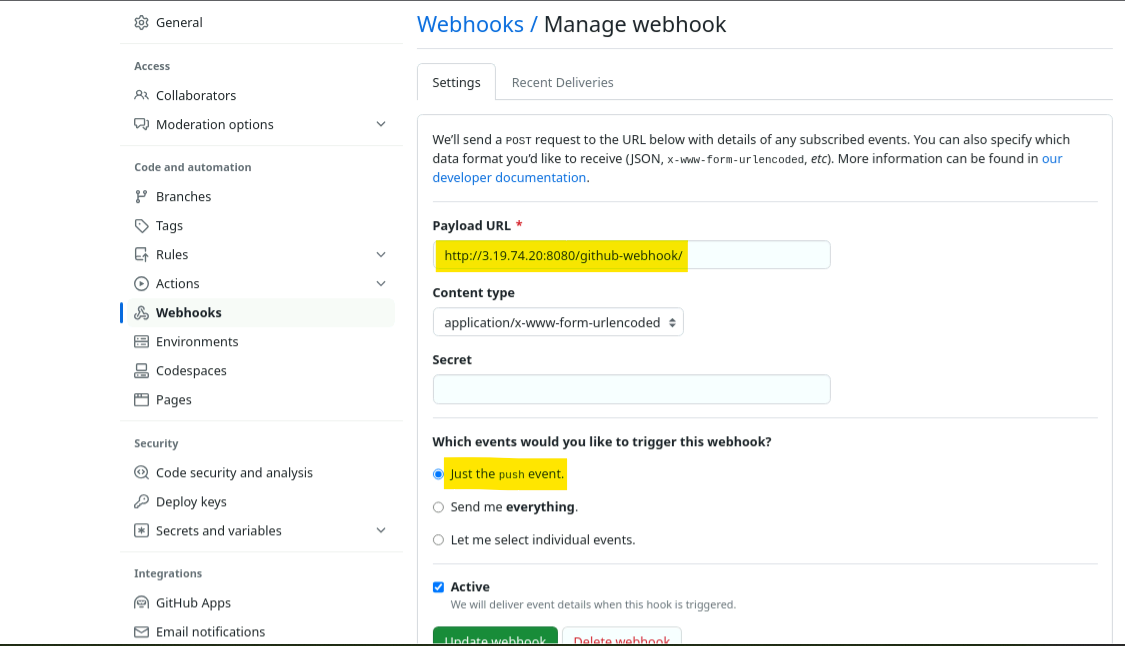
1. In the Source Code Management Tab, add the Github Repo again and type the branch as “develop”.



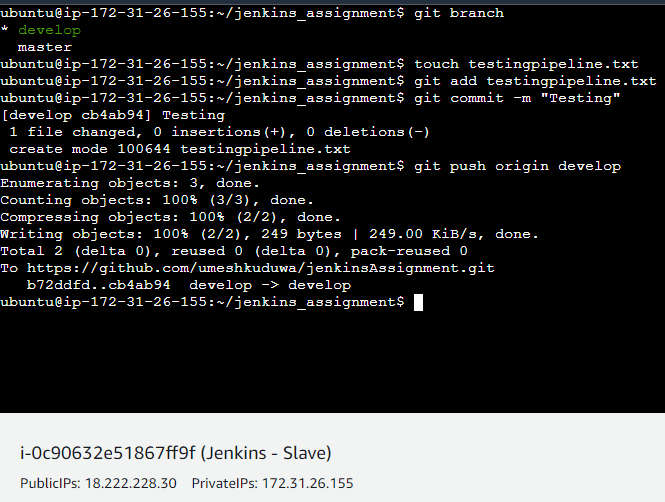
1. Check the GitHub hook trigger for GITScm polling checkbox to Enable webhook.



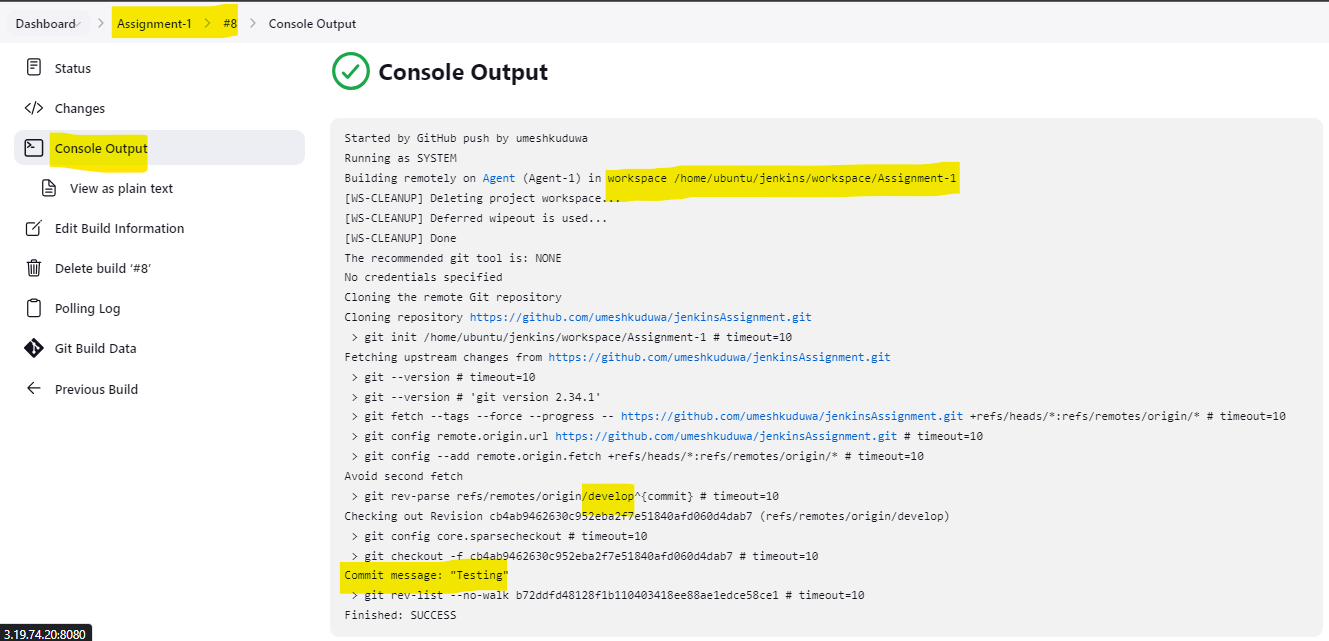
1. Setup the WebHook in GitHub. Add the jenkins Link with extension /github-webook/



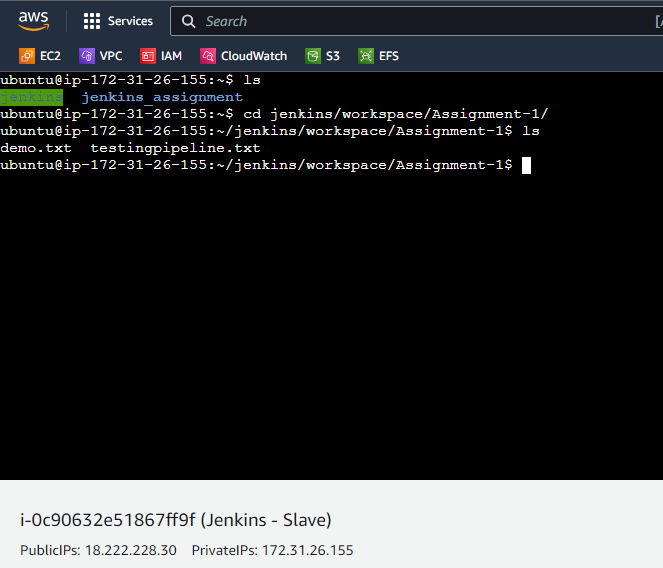
1. Now, push any files into develop branch and the Jenkins will automatically copy the files pushed into the workspace.
   1. This below code will create develop branch, create a text file, commit and push it to remote branch : develop.



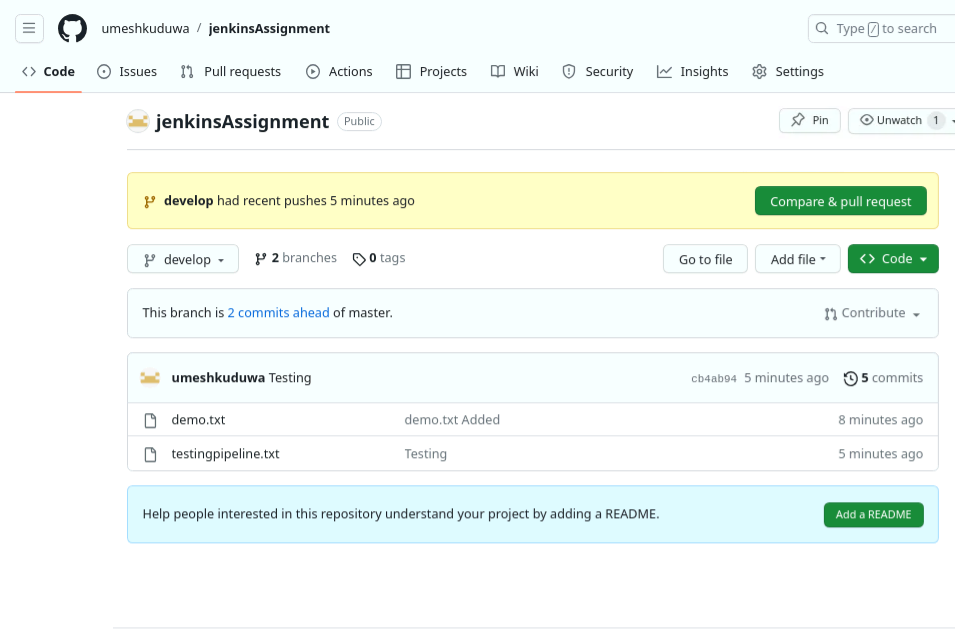
* 1. Below you can see, Jenkins had automatically triggered a Job where the file which was pushed to remote is now copied into Slave Node’s Workspace.



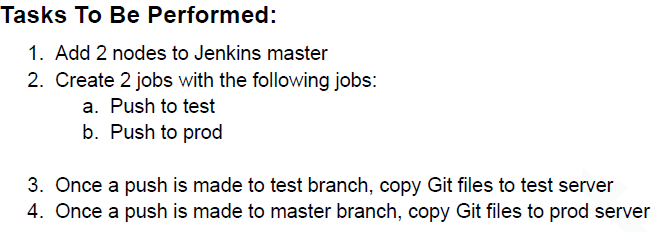
* 1. Now go into the Jenkins Root URL I.e /home/ubuntu/jenkins and you will find that a new folder “workspace” created. Under that, a folder with the task name would have been created and inside that you will find the files of develop branch.



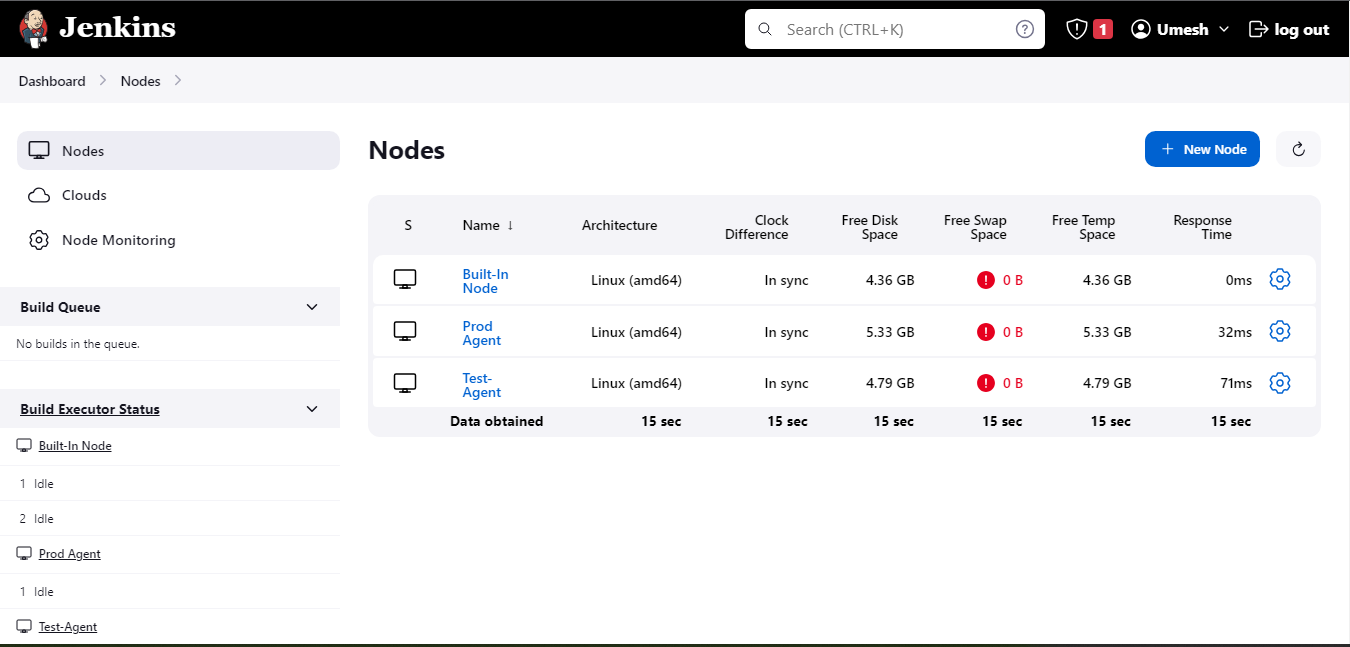
GitHub Remote Develop Branch :



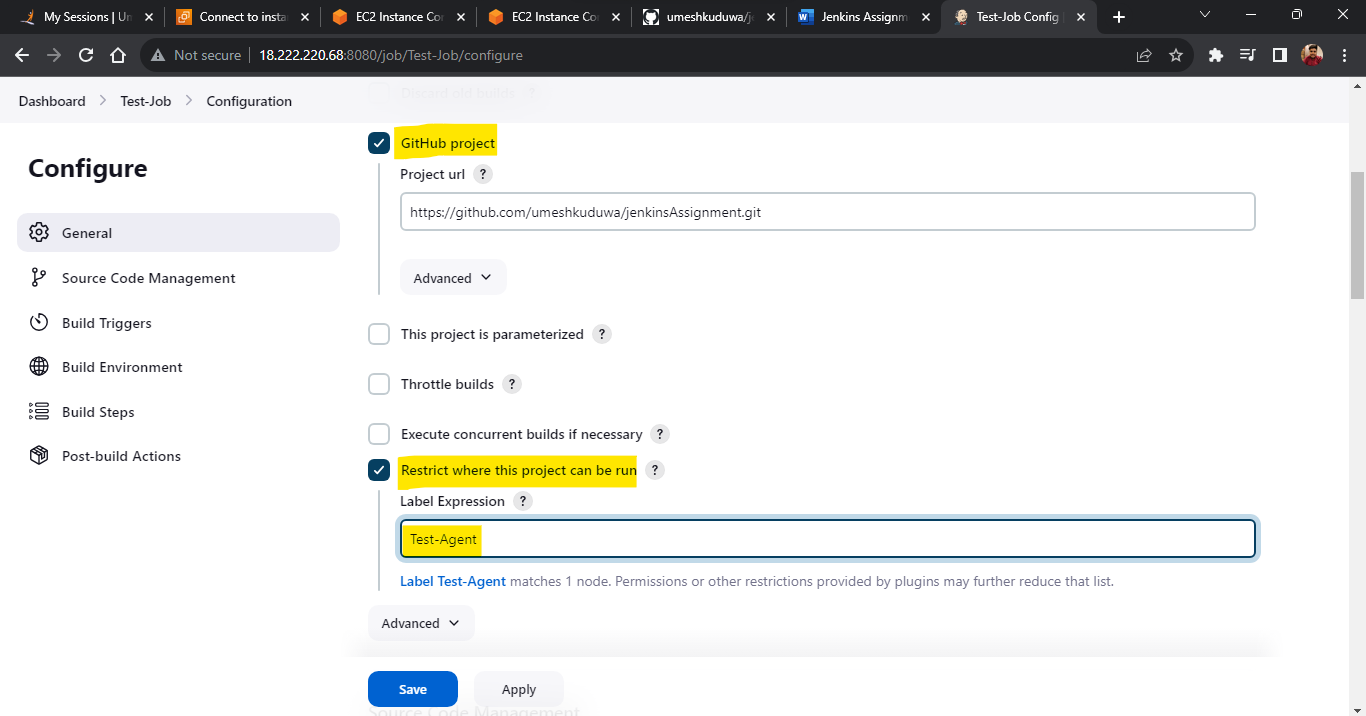
**Jenkins Assignment – 2**



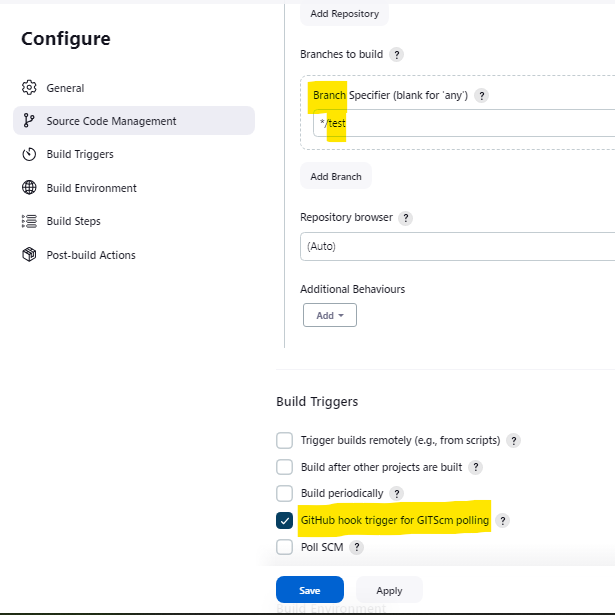
1. Setup 2 Slaves and tie it up to Jenkins Master. Name the 2 Nodes as Test-Agent and Prod-Agent.



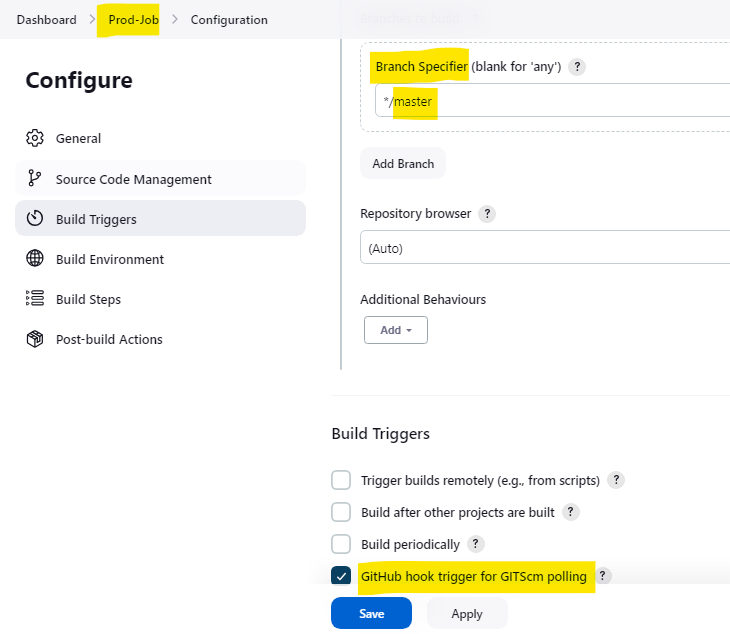
1. Setup 2 Jobs, 1 Job should be test (i.e when a push is made to test branch, it should copy the git files to Test-Agent Slave) and 1 Job should be prod (i.e when a push is made to master branch, it should copy the git files to Prod-Agent Slave)
   1. Test Job should be configured to run only on Test-Agent Node.



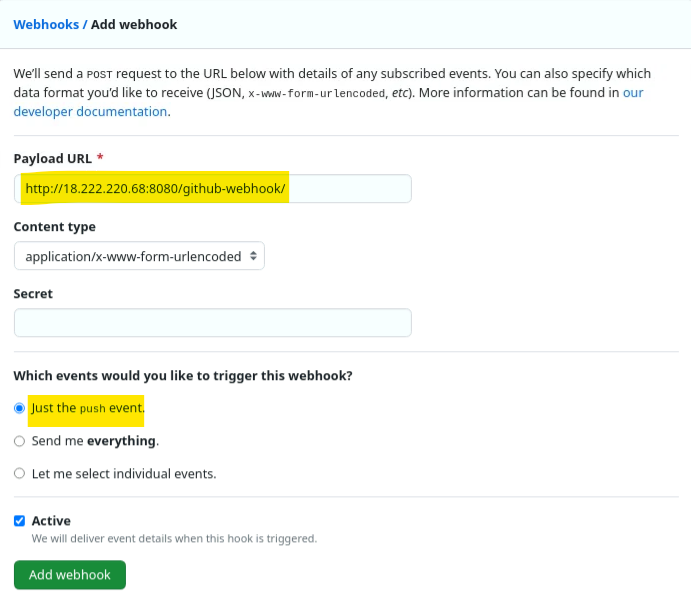
* 1. Mention the Branch as Test and Tick the GitHub polling.



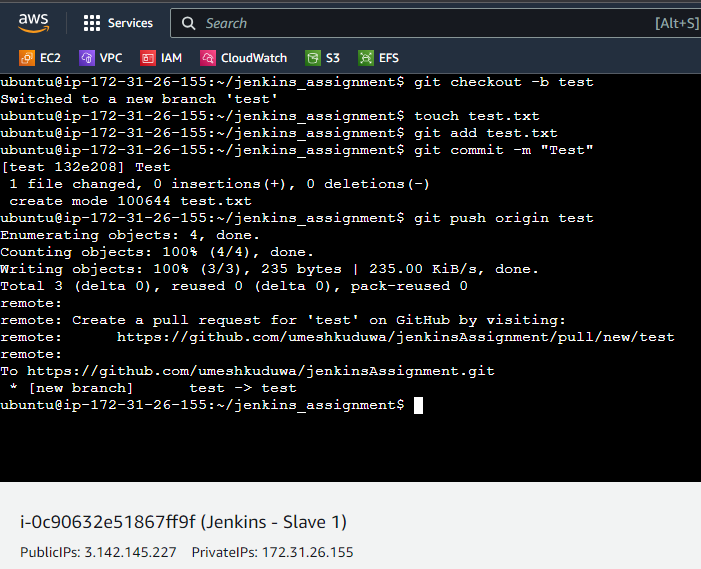
* 1. Same setting would be for the Prod-Job aswell.



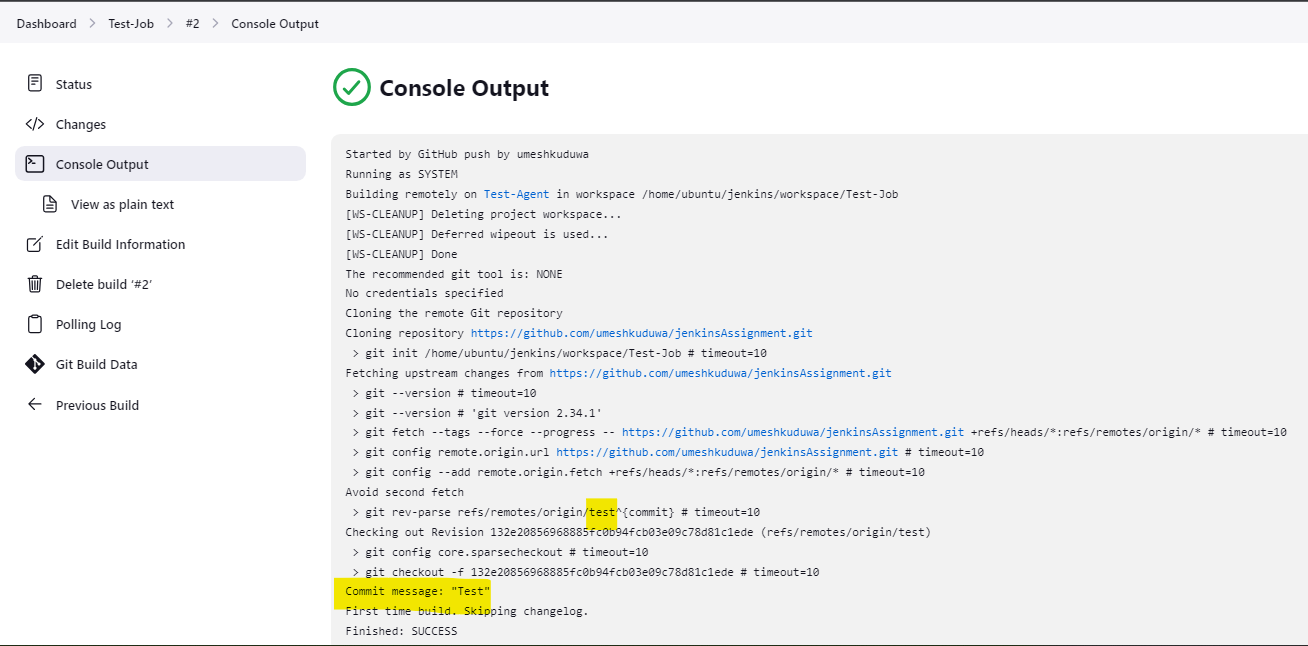
1. Create a webhook for the same.



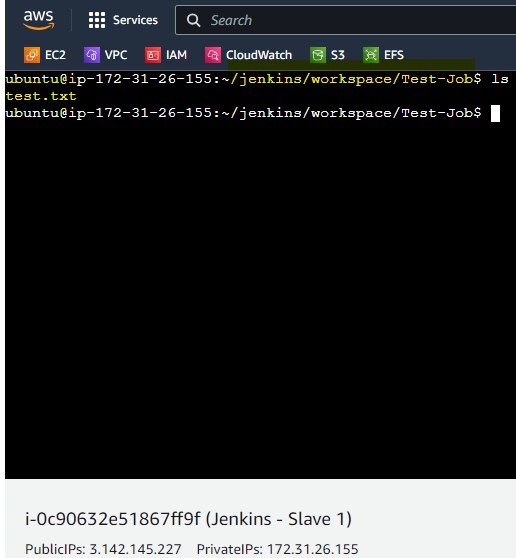
1. Checked if the Test-Agent works by pushing files to test branch on remote repo.
   1. Git Commands:



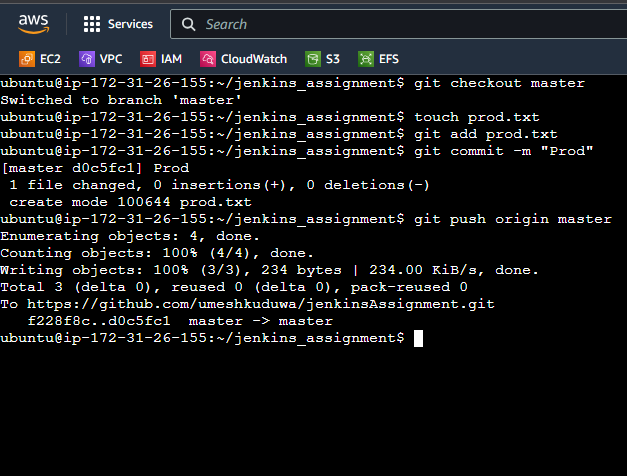
* 1. As observed, the Test-Job ran successfully.



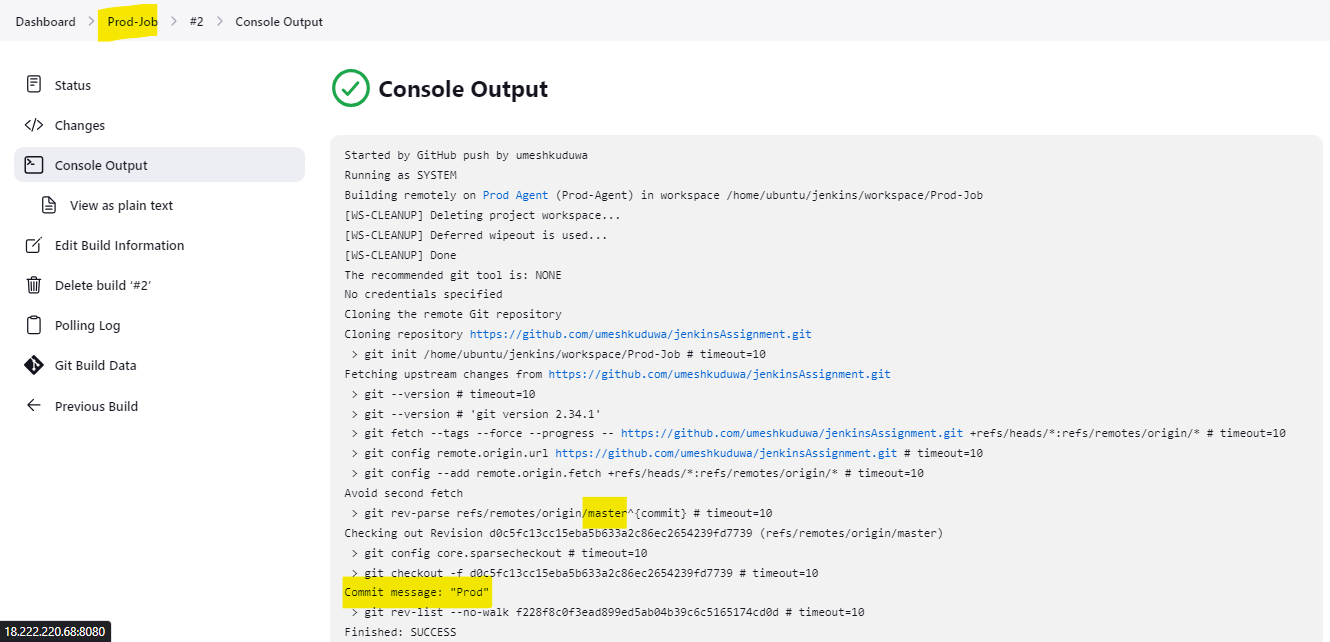
* 1. Checking on the Test-Slave machine if the code is copied.



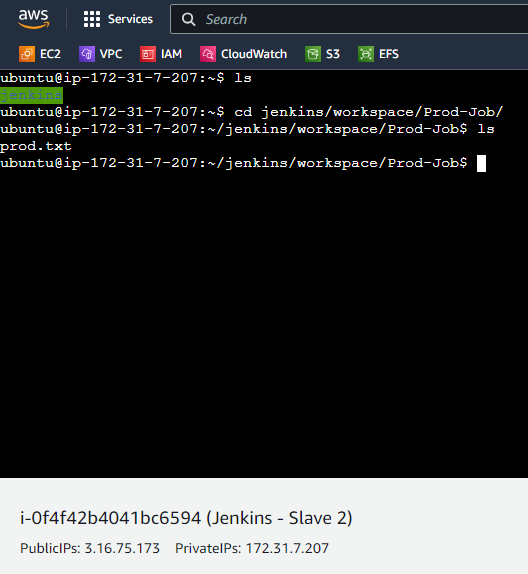
1. Similarly checking for Prod-Job if it triggers automatically.
   1. Git Commands:



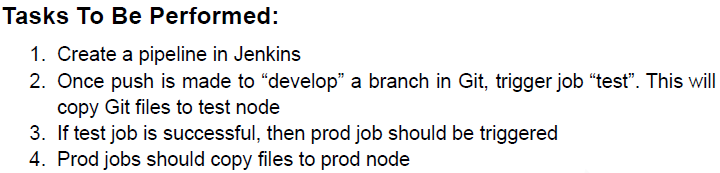
* 1. As observed, the Prod-Job ran successfully.



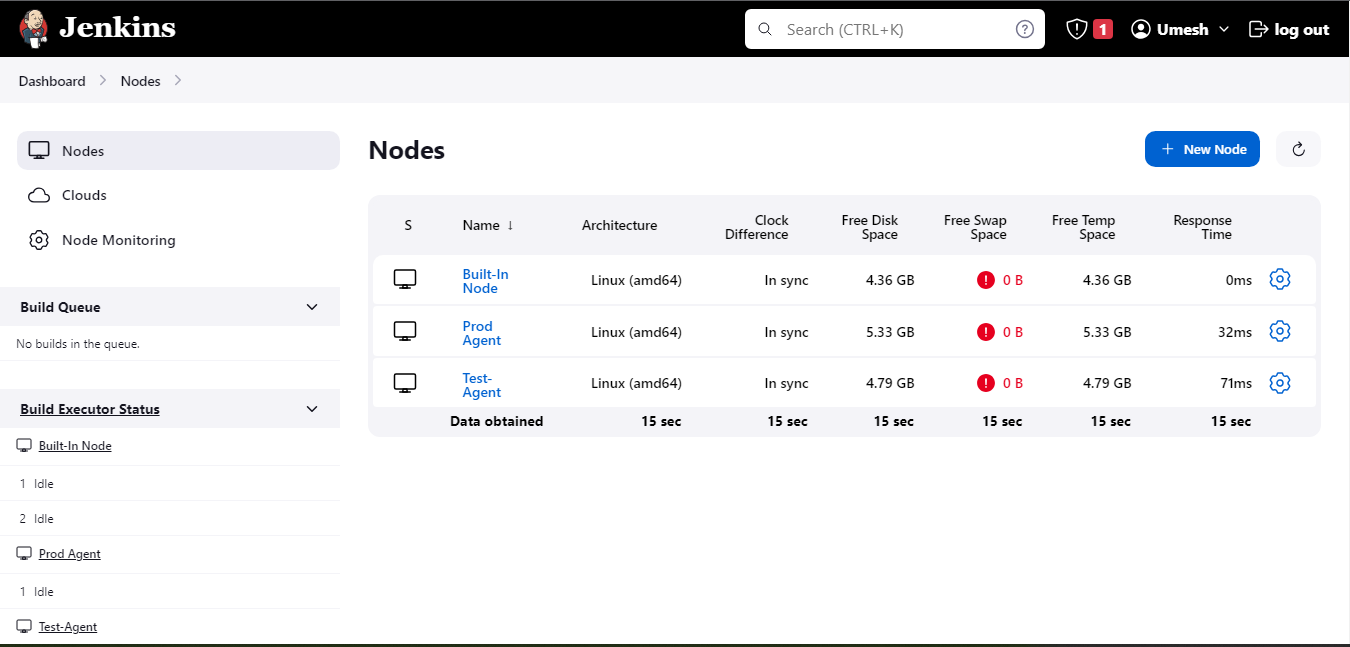
* 1. Checking on the Prod-Slave machine if the code is copied.



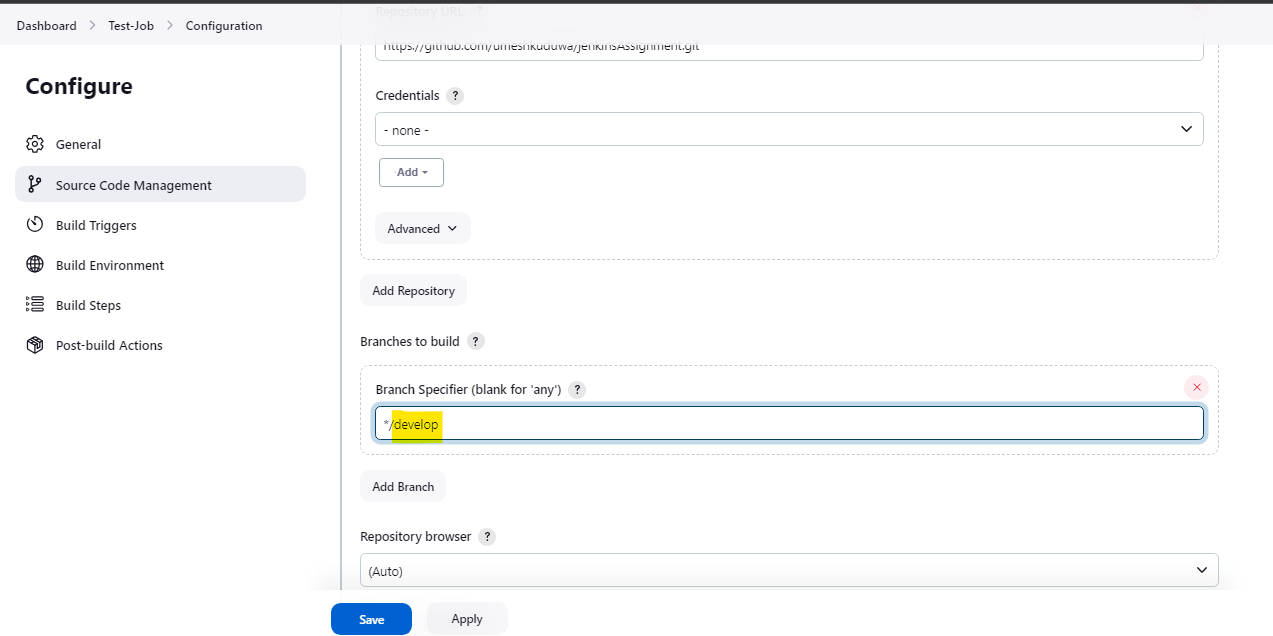
**Jenkins Assignment – 3**



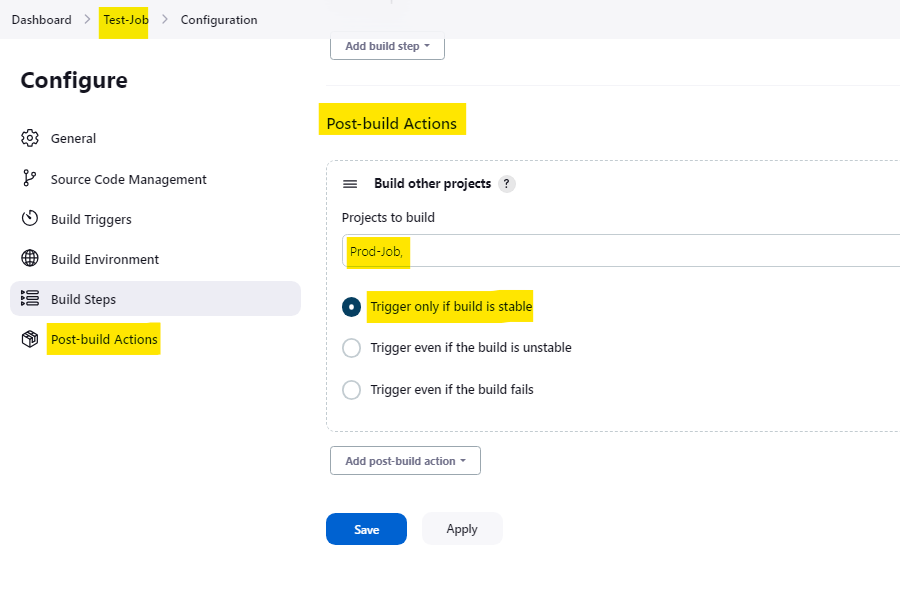
1. Setup 2 Slaves and tie it up to Jenkins Master. Name the 2 Nodes as Test-Agent and Prod-Agent.



1. Setup 2 Jobs, 1 Job should be test (i.e when a push is made to develop branch, it should copy the git files to Test-Agent Slave) and 1 Job should be prod (i.e whenever the test-job is successful, the same files should be copied in Prod-Agent Slave aswell)
   1. Test Job should be configured to run only on Test-Agent Node.

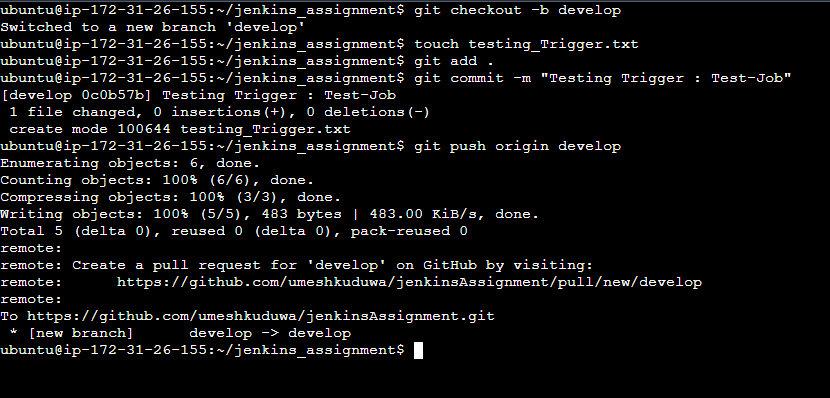


* 1. Configure the Post-Build action to Trigger Prod-Job whenever the Test-Job is Stable.

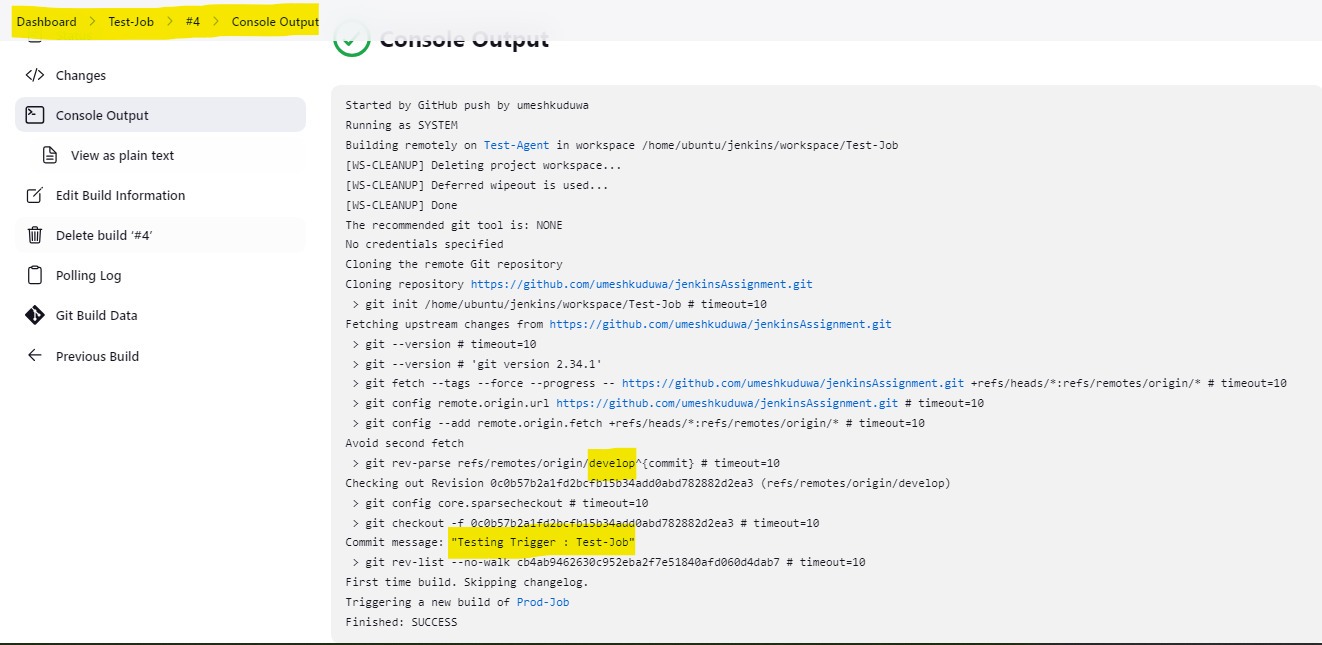


* 1. Configure the Prod-Job aswell, mention Branch as develop and untick the WebHook Polling Option.

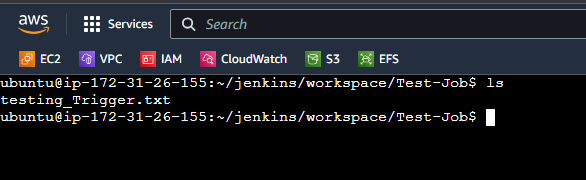
1. Try pushing a file to develop branch and see if both the job triggers :
   1. Git Commands:



* 1. Test-Job in Jenkins:



* 1. Checking if the Test-Slave has the file:



* 1. Since the build is Successful, checking if Prod has the file:

