#Devops Assembly Line Assignment

Creating a Jenkins Pipeline Which Integrates the Development, Testing and Deployment Environment ultimately resulting end to end automation.

#***Assuming that git, docker and Jenkins are installed in the respective systems and Jenkins has been assigned all permissions in sudoers file in Redhat 8.***

 # Also, assuming the dev user knows all basic git commands. If not you can click here to go through all git commands.

Step 1 → *Start Jenkins from your operational environment (#here I am using RHEL version 8 for the same which is running via virtualization (i.e. VMware) on Windows 10 as host OS)*

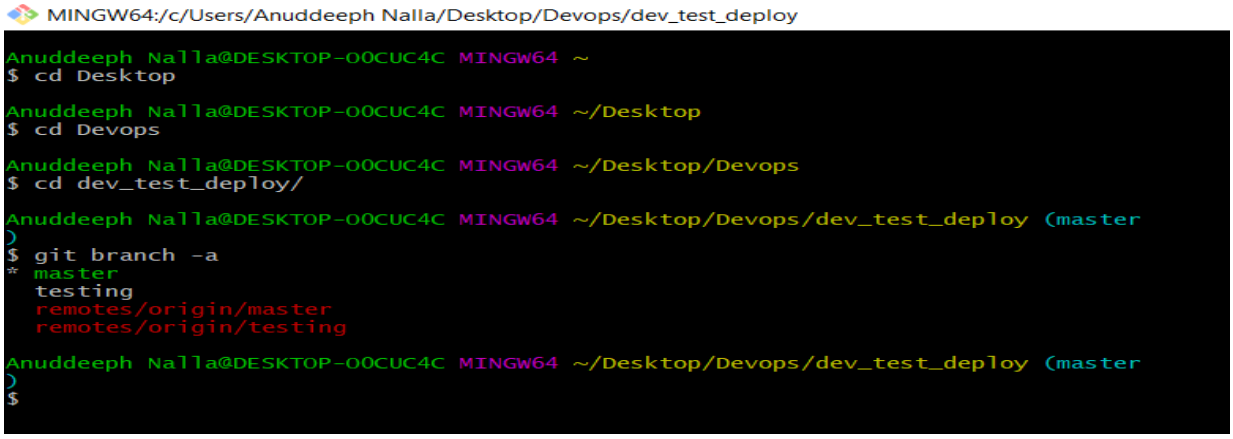
Step 2 → *Ask your developer to launch his environment (#here I am using Windows 10 as development environment)*

Step 3 → *Here I am using my GitHub repository as remote server to upload local repo files using git commit and push. Create a new repository without checking on initializing readme.md and remotely add it to your local Git repo.*

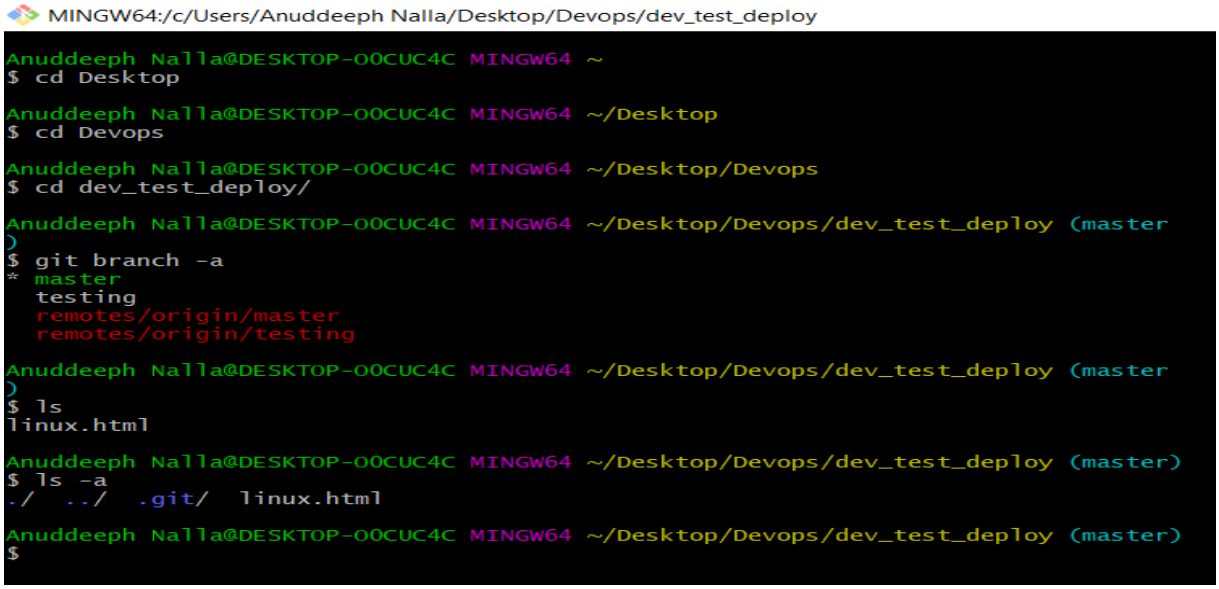
Step 4 → Now in local development environment, I have created a new branch i.e. testing branch along with the master. (master is default)

- ➔ The command used to create branch is (git branch testing) and to switch (git checkout testing)

- ➔ Another way is to create and switch at a time by using (git checkout -b testing)



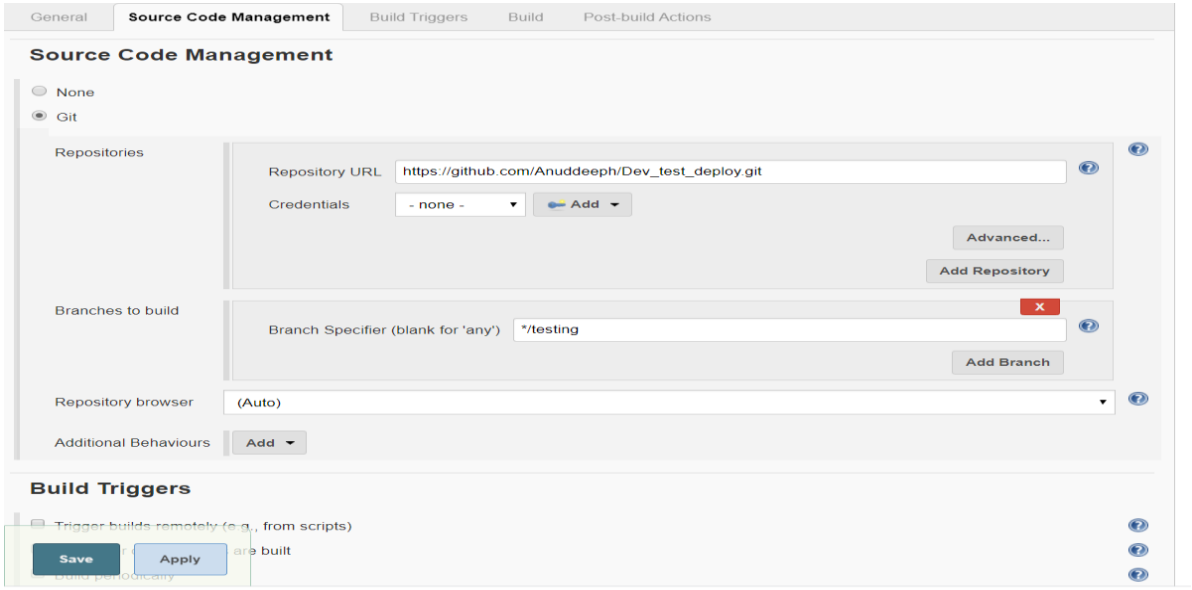
Step 5 → I have a file linux.html in master branch which is also pushed/uploaded to testing branch. (using git push -u origin testing # you have to do this for the first time only after that git commit and push is enough)



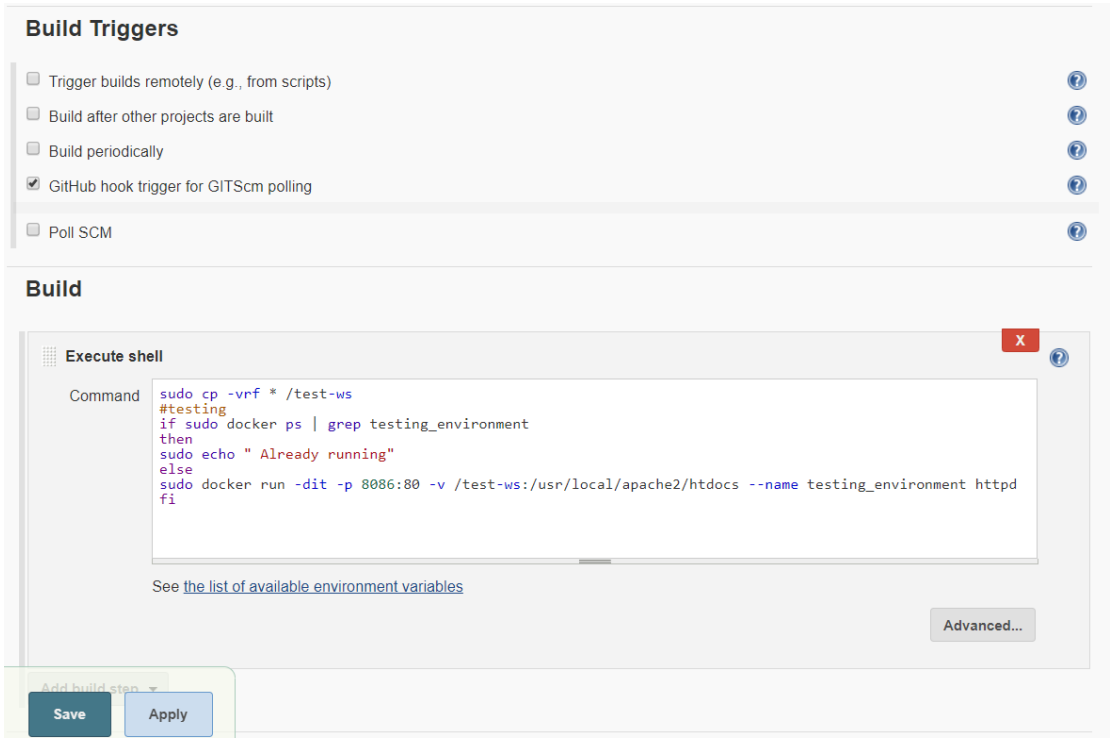
Step 7 → Now after modifying something in my testing branch, I decided to give this to my Quality Assurance Team i.e. operations' team to check the code on their environment for testing and approval. (Using RHEL_8 here)

Step 8 → Open Jenkins dashboard and create a new job named as Testing Job for the Testing_of_code. In that put Follow these steps, Here I am using a docker container with httpd image to generate a testing environment running on port

8086. The container is attached to the directory test-ws, test-ws was a directory created in RHEL-8.



→Don't forget to mention the testing branch!!



➔ Do not forget to tick on GitHub hook trigger for GITScm polling.

The Job firstly copies the downloaded data to test-ws from GitHub, when triggered by webhooks(explained further) and then checks if a container named testing_environment is running, if not then it launches the container of same name with httpd image on port 8086, here test-ws works as mounted volume for the container, and we also exposing the webserver using -p.

The Job is triggered on GitHub webhooks. For this give the path of your GitHub repository and go to your GitHub repository settings.

➔ Open GitHub, go to the settings, select webhooks.

➔ Add Webhook

➔ Switch to RHEL-8, for tunnel, and run ngrok program (i.e. #./ngrok http 8080)

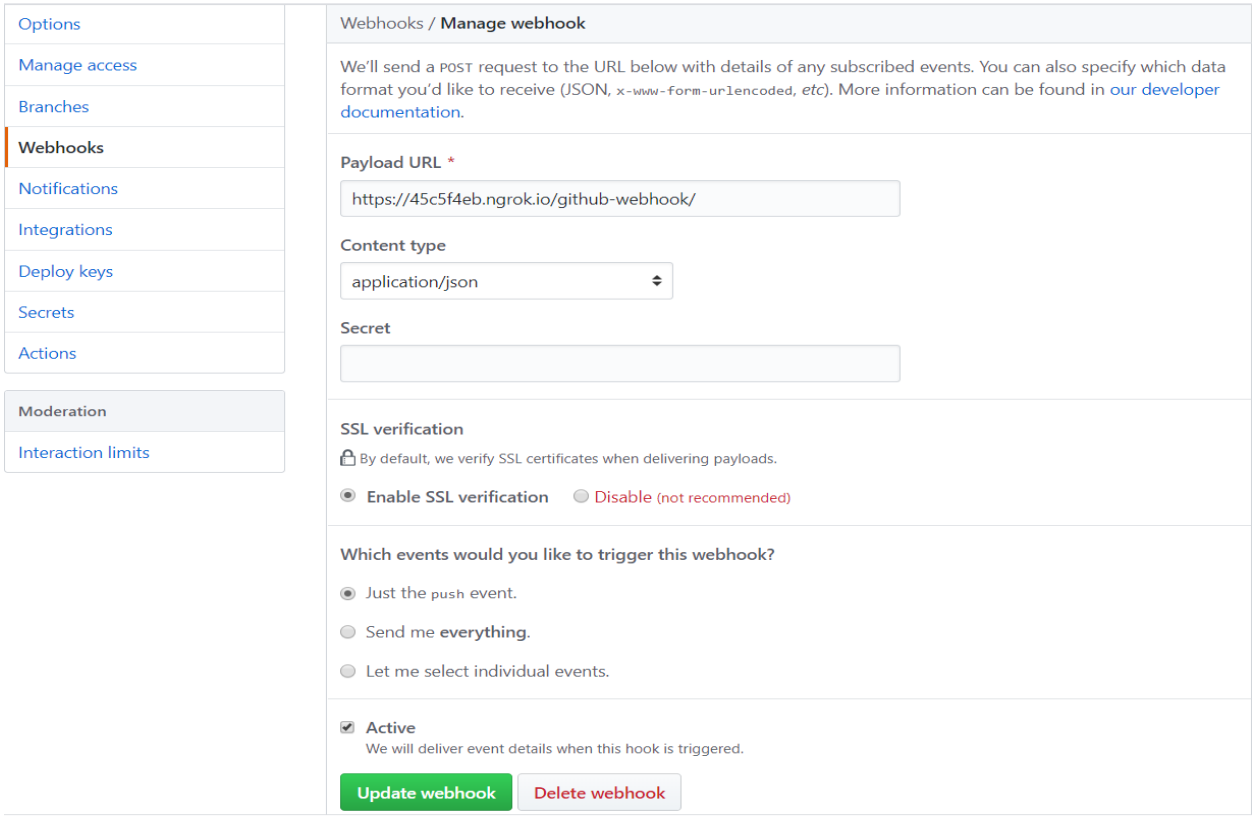
➔ Copy the ngrok link into the webhook and add last /github-webhook/ (for example https://45c5f4eb.ngrok.io/githun/webhook/)

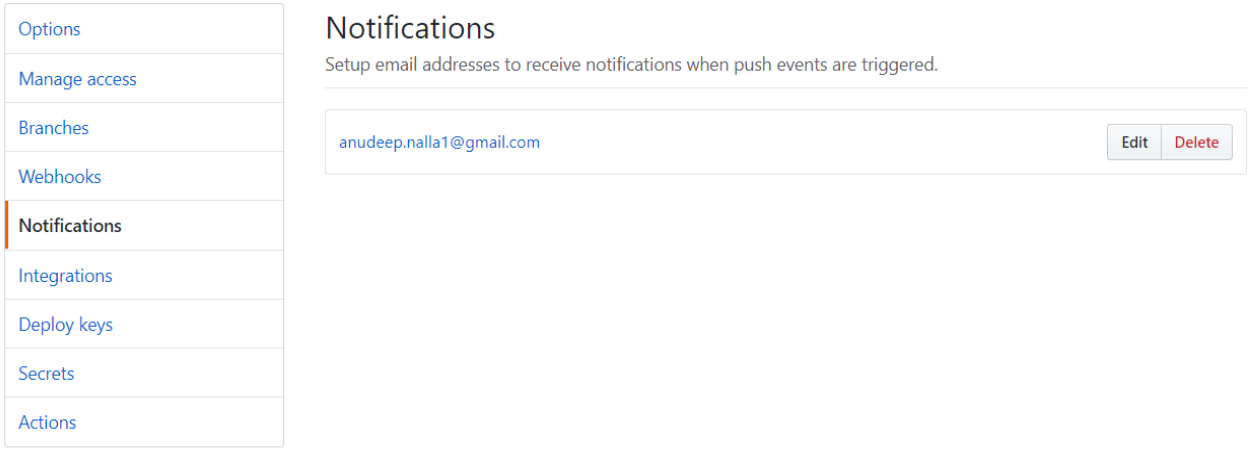
➔ Select application/json in the Content type box.

➔ Select your desired options

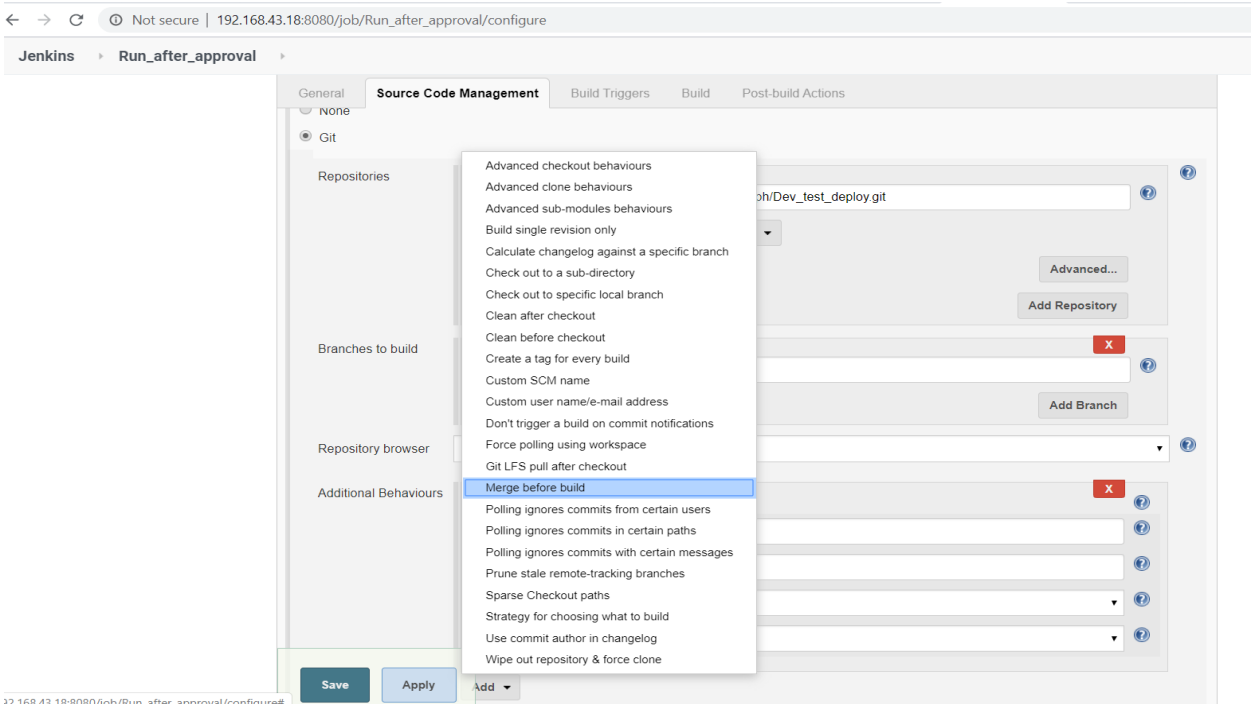
➔ Checkbox active and Update webhook.

➔ Also go to notifications option located under webhooks and place the email id of operation's team/(here for learning yours) to get notified whenever a new update is performed in the repository.

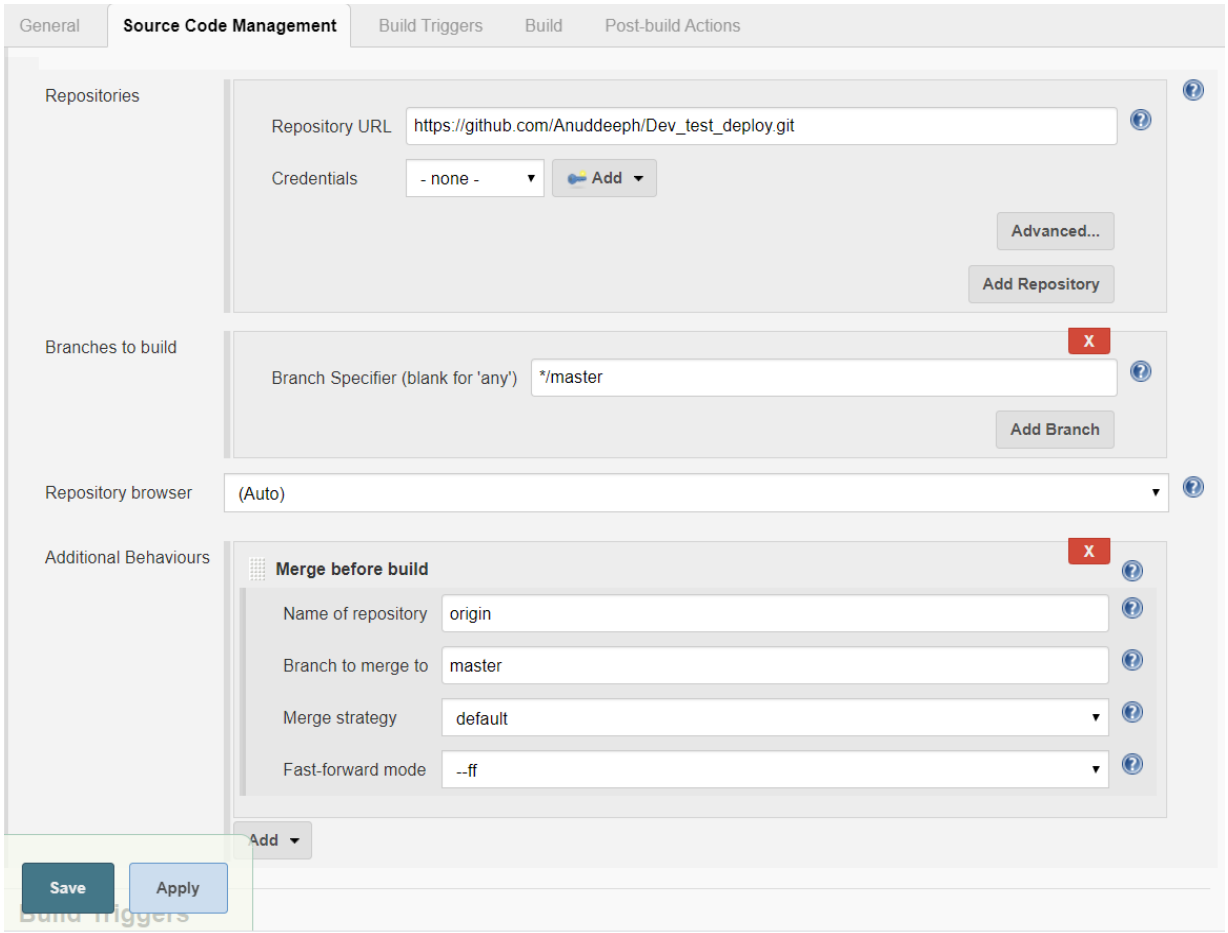




Step 6 → Fall back to your Jenkins dashboard and create another new job named as Run_after_approval as this is the job you will / QAT team will build/run this job when it checks the testing environment and the site/code is ready to be launched for clients.



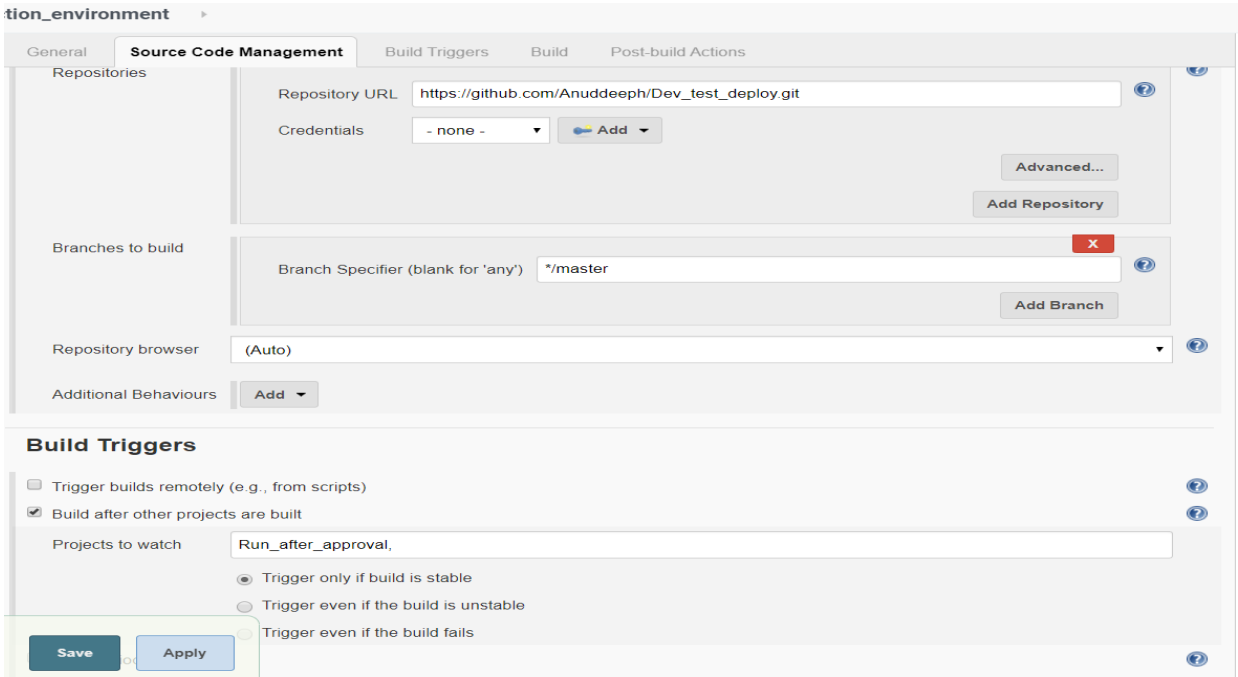
➔ Go to the additional behaviors and select “Merge before build”.



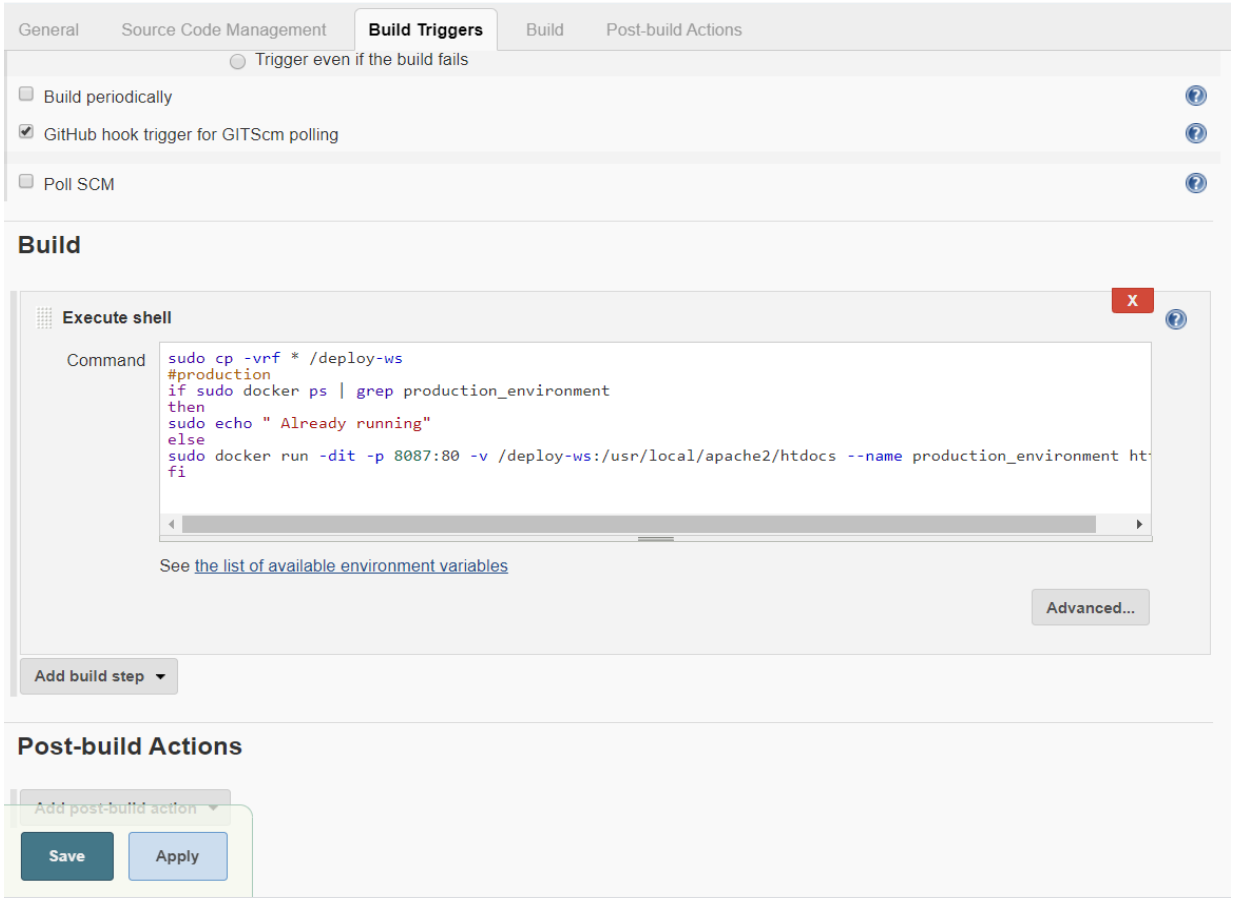
➔ Input all the details and branch names accordingly.

#If merge conflicts occur please go manually and solve the conflict first and then run this job.

Step 7 → Again in Jenkins dashboard, create a Job named as Deployment_to_Production_Environment. This job is to be run just after the QAT team approves and builds "Run_after_approval job." So, to chain this job to the former one goes to Build Triggers and tick Build after other projects are build, then mention the job name you want to chain it (Here Run after approval.)



Step 8 → The following commands theory is same as the testing_environment_job build theory and only difference is that it is pulling the code from the master branch to deploy-ws directory and deploying it to the production_environment running on port 8087.



FINALLY, RUN ALL THE JOBS ACCORDINGLY GIVEN INSTRUCTIONS AND YOUR END TO END JENKINS PIPELINE INTEGRATING DEV, GIT, GITHUB, DOCKER CONTAINERS, OPs are Now fully functional.