**Project Documentation (Knative with Jenkins)**

**Dev Setup**

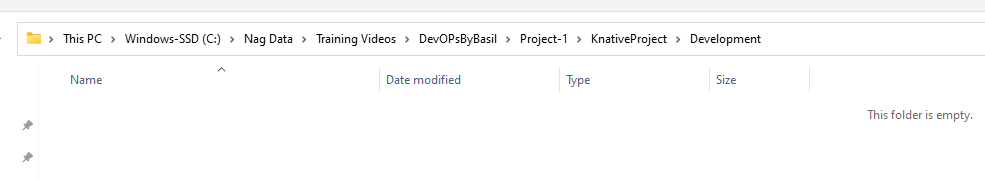
**NOTE:** Ensure you have GitHUb account with proper repository created before you start this task. And also I am preparing my dev environment first with one application (coit-frontend)

**Clone and publish the code from local git to remote repository GitHub:**

**Git Setup: Follow below link to install Git client on your machine.**

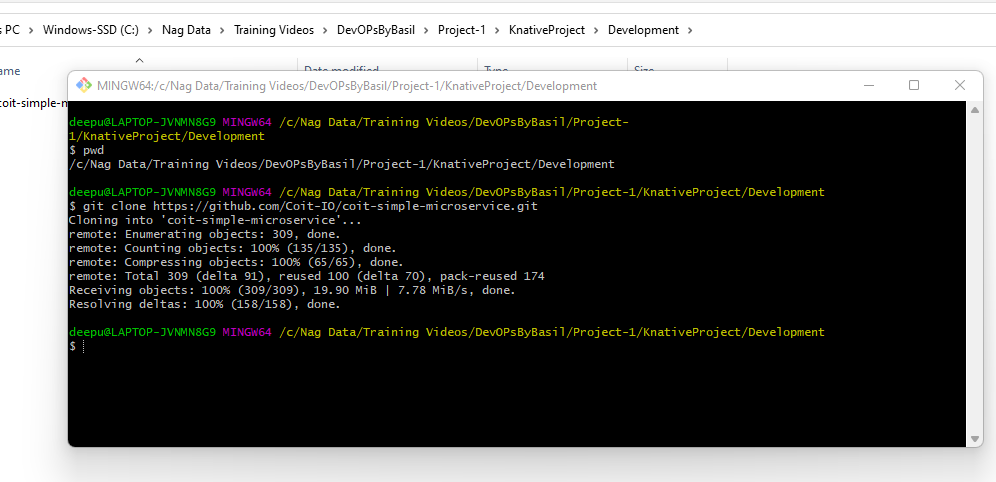
[Git - Downloads (git-scm.com)](https://git-scm.com/downloads)

Create an empty folder with name Development

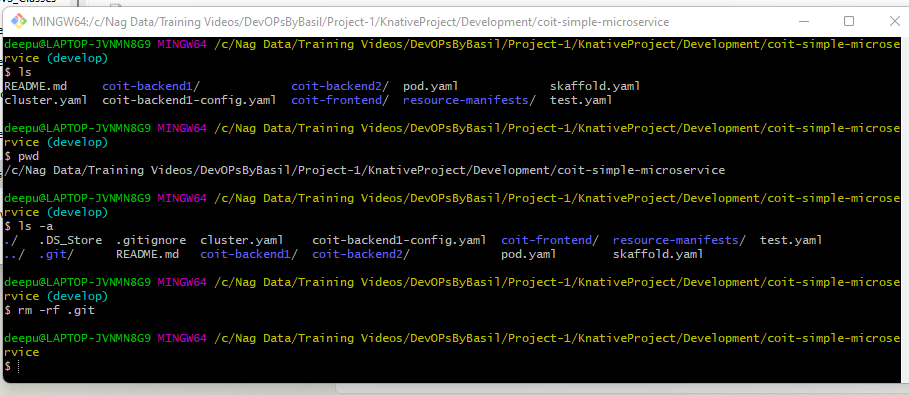
****

Clone coit-simple-microservice to this empty directory by using GitBash terminal that you get once you installed Git client on your machine,

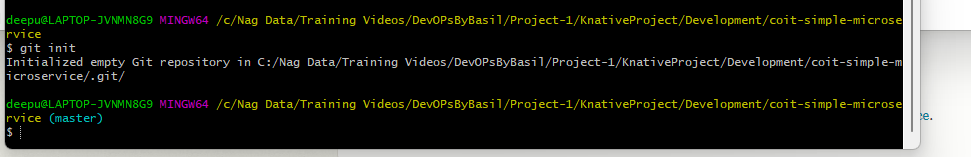
**git clone** [**https://github.com/Coit-IO/coit-simple-microservice.git**](https://github.com/Coit-IO/coit-simple-microservice.git)

****

**Go to “coit-simple-microservice” directory and remove .git folder to create your own local repository and commits.**

****

**Run “git init” command to make current local directory to working directory.**

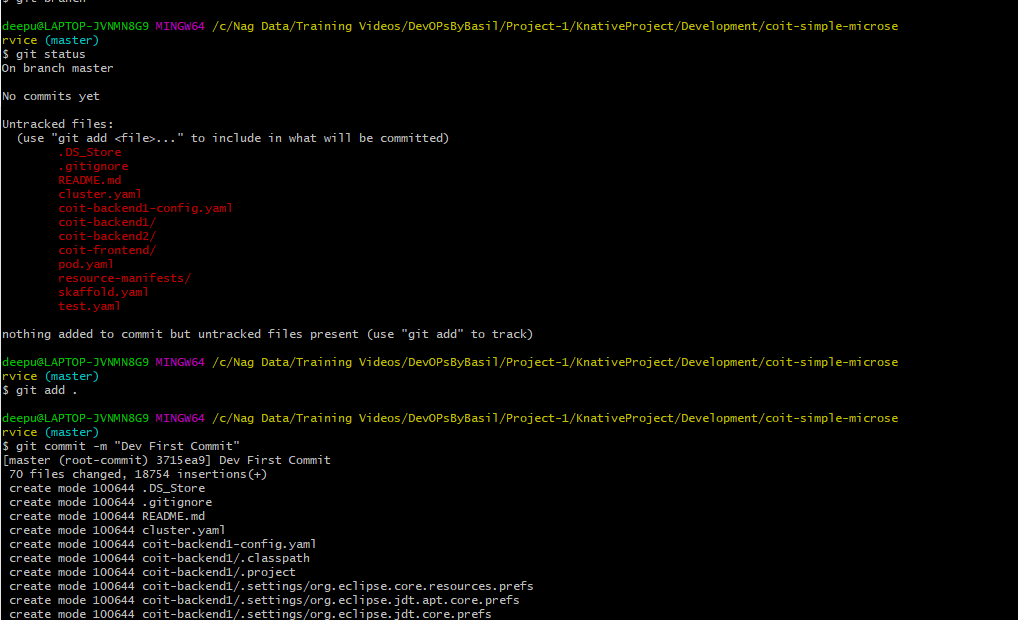
****

**By default it’s a Master branch and we can commit all our code changes to local master branch by running below commands**

**Git status**

**Git add .**

**Git commit -m “Dev First Commit”**

****

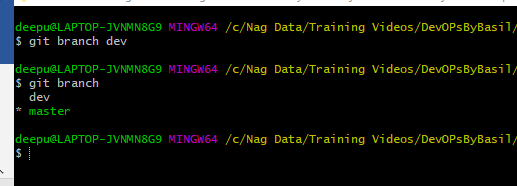
**Since my main module code is committed to local repository, I will create a separate branch for Development env work.**

**Use below command to create a sub branch with name “dev’**

**Git branch “dev”**

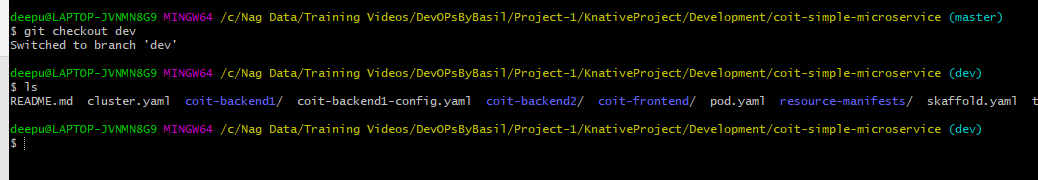
**To see list of branches use below command**

**Git branch**

****

**Now check out to dev branch by using below command**

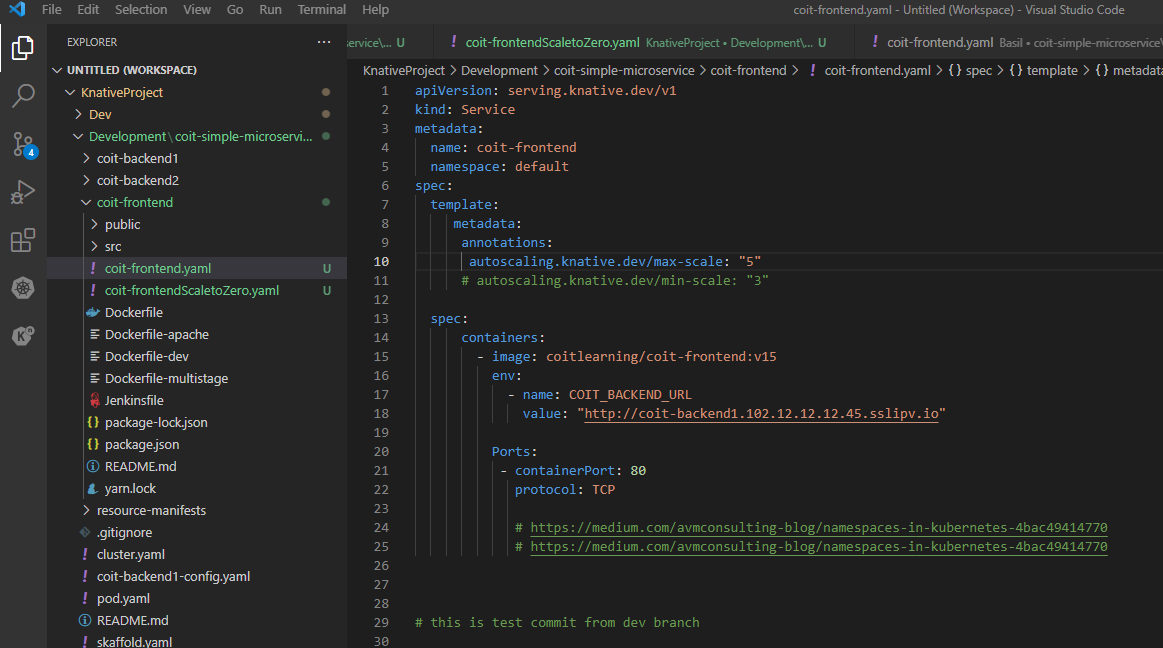
**Git checkout dev**

****

**Here is I want make all modification to my original code as per Development environment requirement like** scale down to zero enabled, and also the maximum number of replicas each application can have will be 5 in production and point github & docker registry images to my own repositories.

Below is the sample code for one of frontend app.

Two yaml files ( one is for setting up max 5 replicas

****

****

**As my required my Kubernetes manifest files are ready with my requirement I can commit these changes to local Dev branch working repository and merge with master branch then push it to my Github remote repository.**

**Below are the command I used to perform above all tasks.**

**Git status**

**Git add .**

**Git commit -m “dev changes -1”**

**Git checkout master**

**Git merge dev**

**git config --global user.name "Nag"**

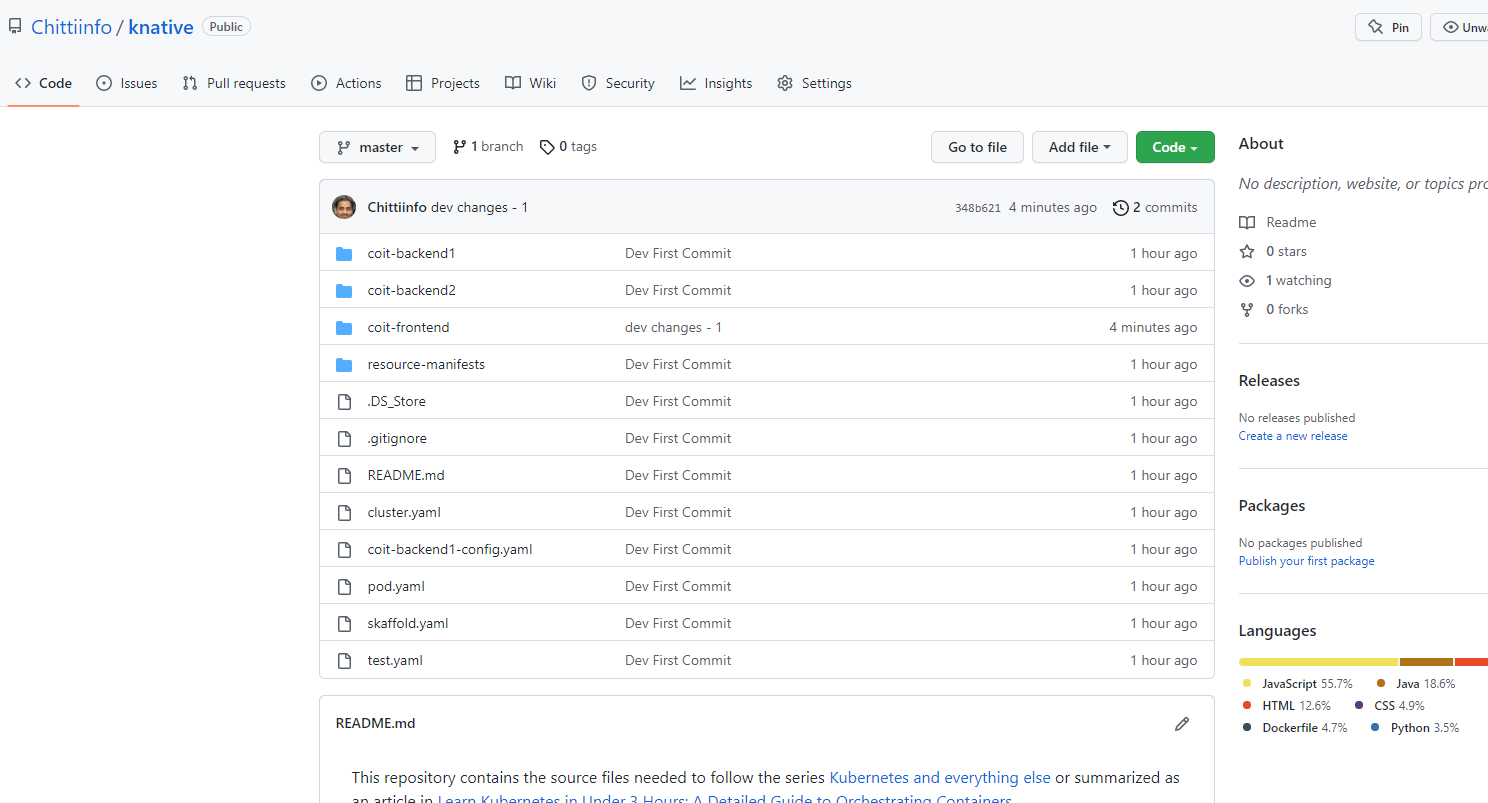
**git config --global user.email "chittiinfo@gmail.com"**

**git remote add origin** [**https://github.com/Chittiinfo/knative.git**](https://github.com/Chittiinfo/knative.git)

**git login**

**git push origin --all**

**you can see my code is now pushed into my remote repository chittiinfo/knative.**

****

**Jenkins work:**

**Create an instance in AWS (Me**

**Install Java and Jenkins**

[**https://www.jenkins.io/doc/book/installing/linux/**](https://www.jenkins.io/doc/book/installing/linux/)

**Install docker**

[**https://docs.docker.com/engine/install/ubuntu/**](https://docs.docker.com/engine/install/ubuntu/)

**Install kubectl**

[**https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/#install-kubectl-binary-with-curl-on-linux**](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/#install-kubectl-binary-with-curl-on-linux)

**Install Kn**

[**https://knative.dev/docs/client/install-kn/#install-the-knative-cli**](https://knative.dev/docs/client/install-kn/#install-the-knative-cli)

**Install Gcloud**

[**https://cloud.google.com/sdk/docs/install**](https://cloud.google.com/sdk/docs/install)

**Build Jenkins job:**

cd coit-frontend

docker build -t chittiinfo/coit-frontend:${APP\_VERSION} . -f Dockerfile-multistage

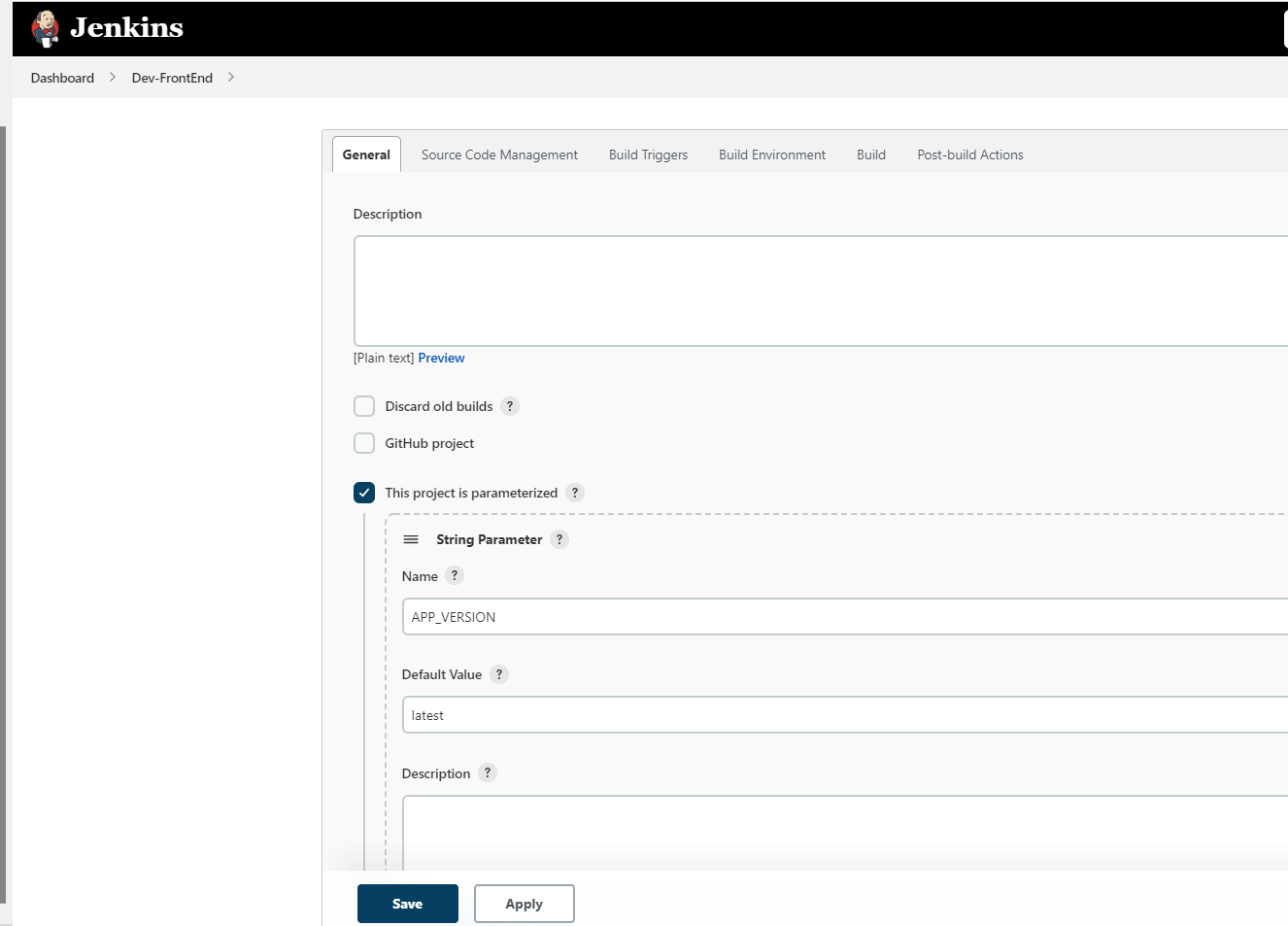
**docker push chittiinfo/coit-frontend:${APP\_VERSION}**

**kubectl apply -f coit-frontendScaletoZero.yaml**

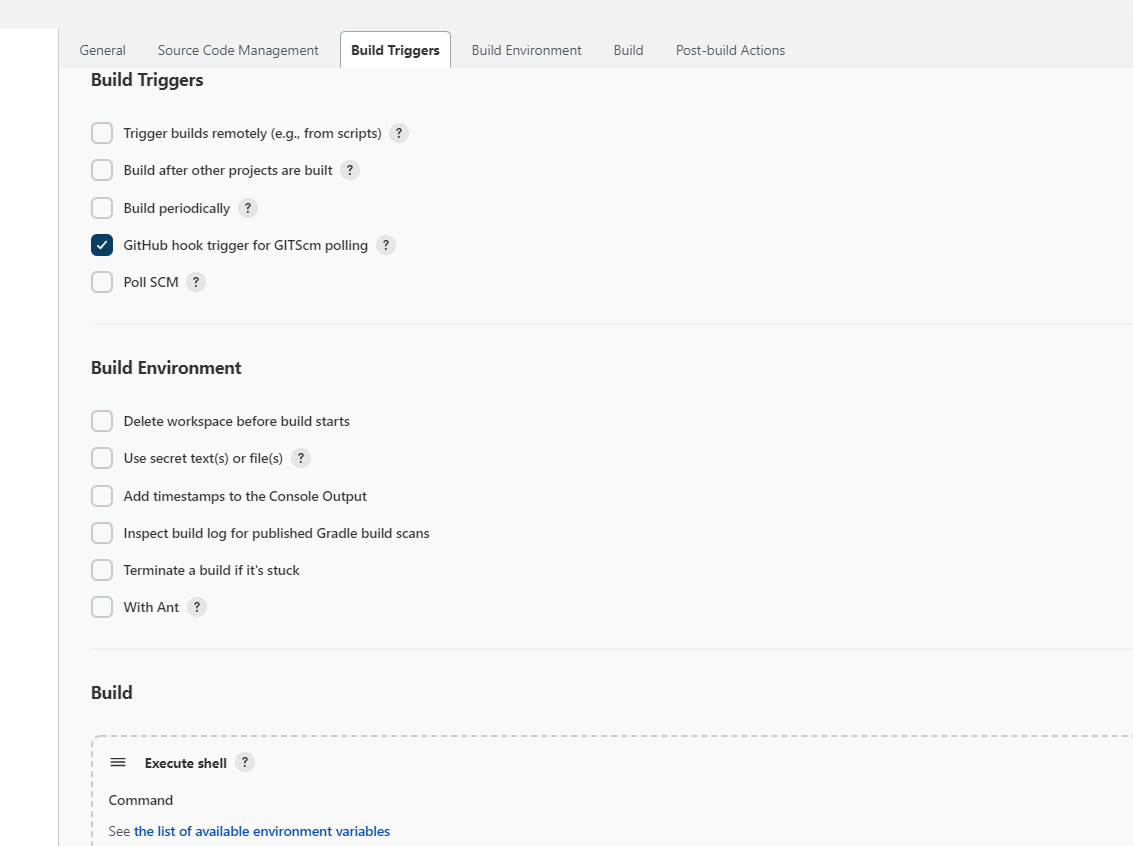
**kn service create coit-frontend --image chittiinfo/coit-frontend: ${APP\_VERSION} --port 80 --scale-max 5 -n developement**

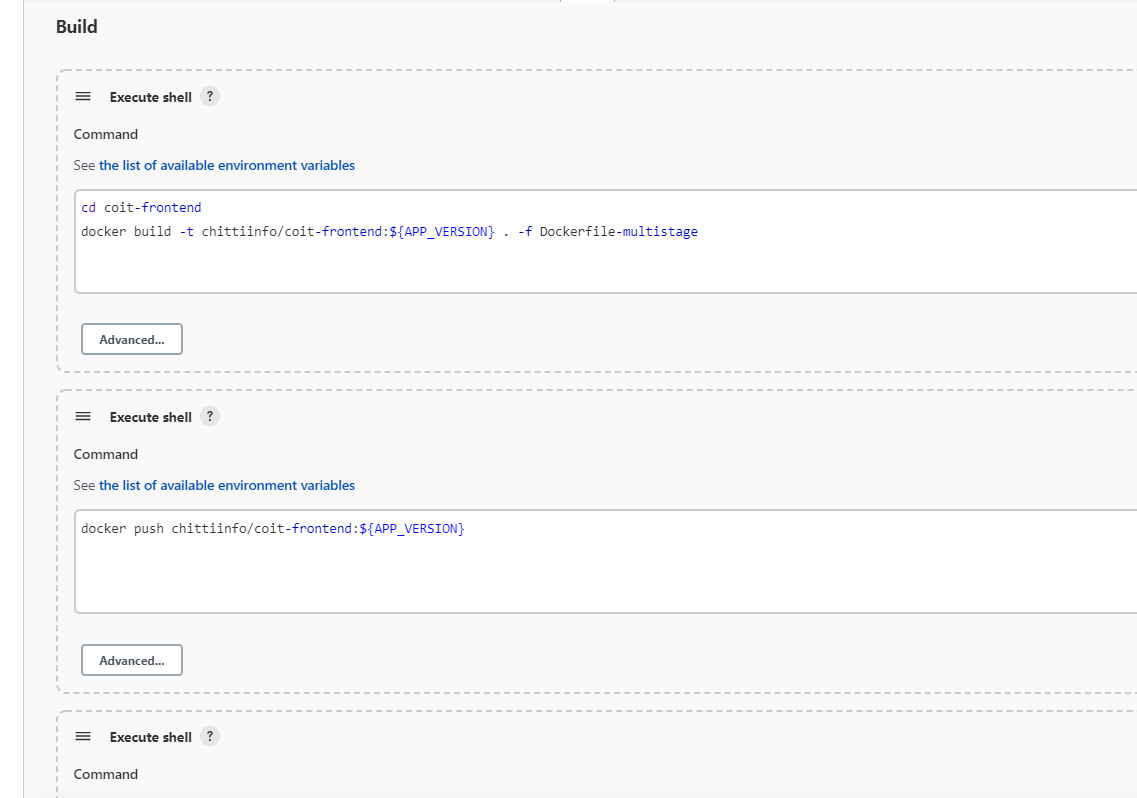
**Email setup:**

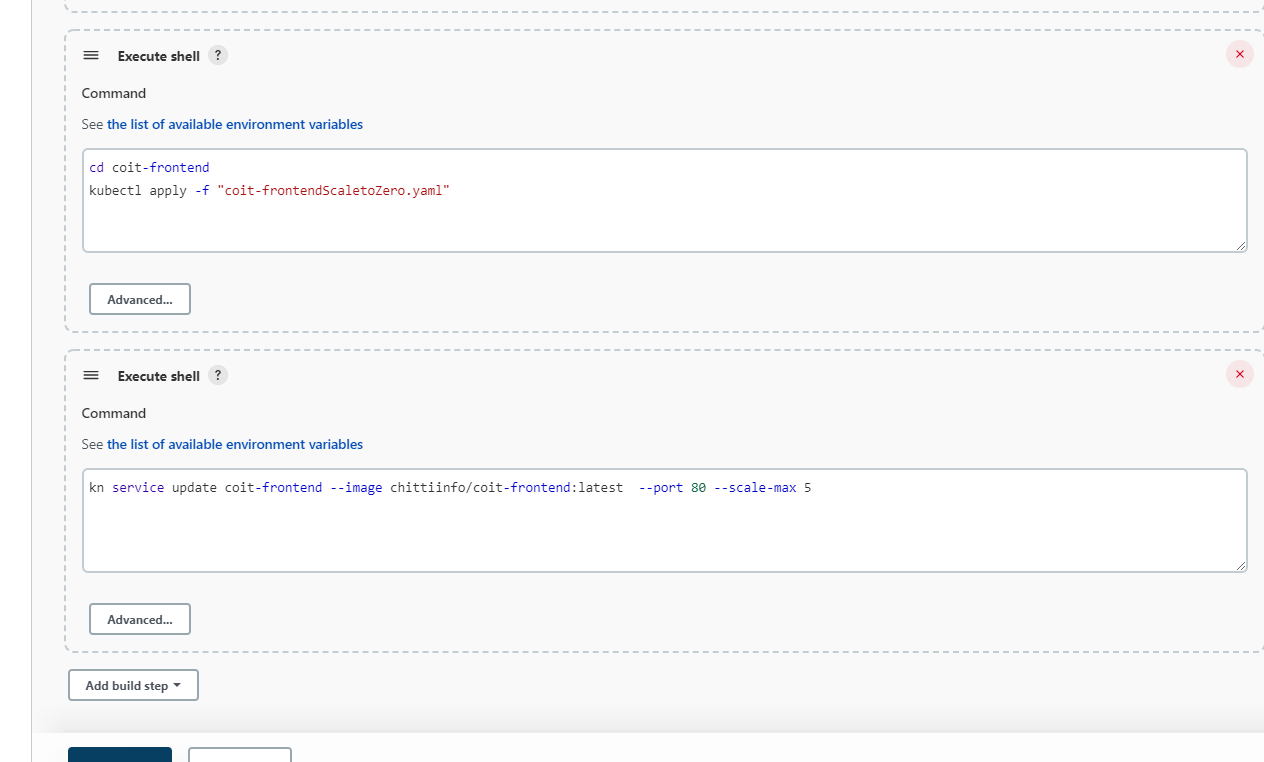
[**https://www.youtube.com/watch?v=MFgbp00hbVI**](https://www.youtube.com/watch?v=MFgbp00hbVI)

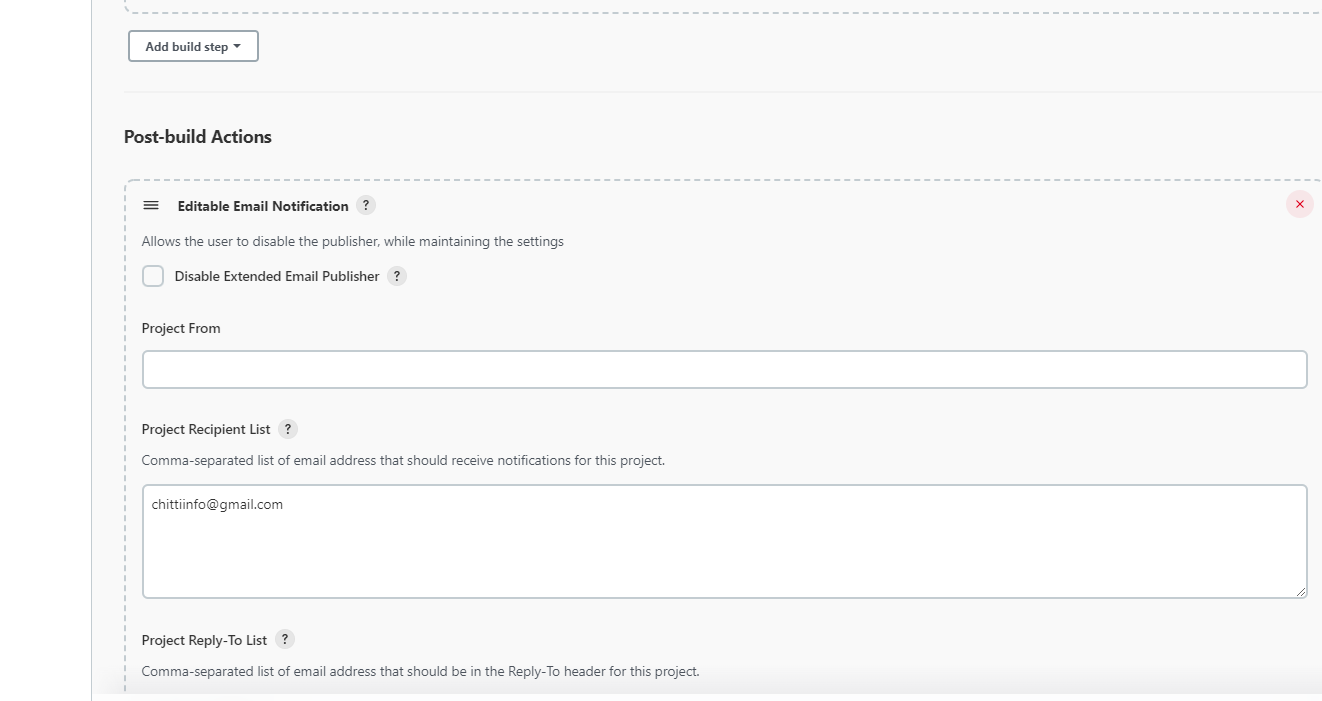
****

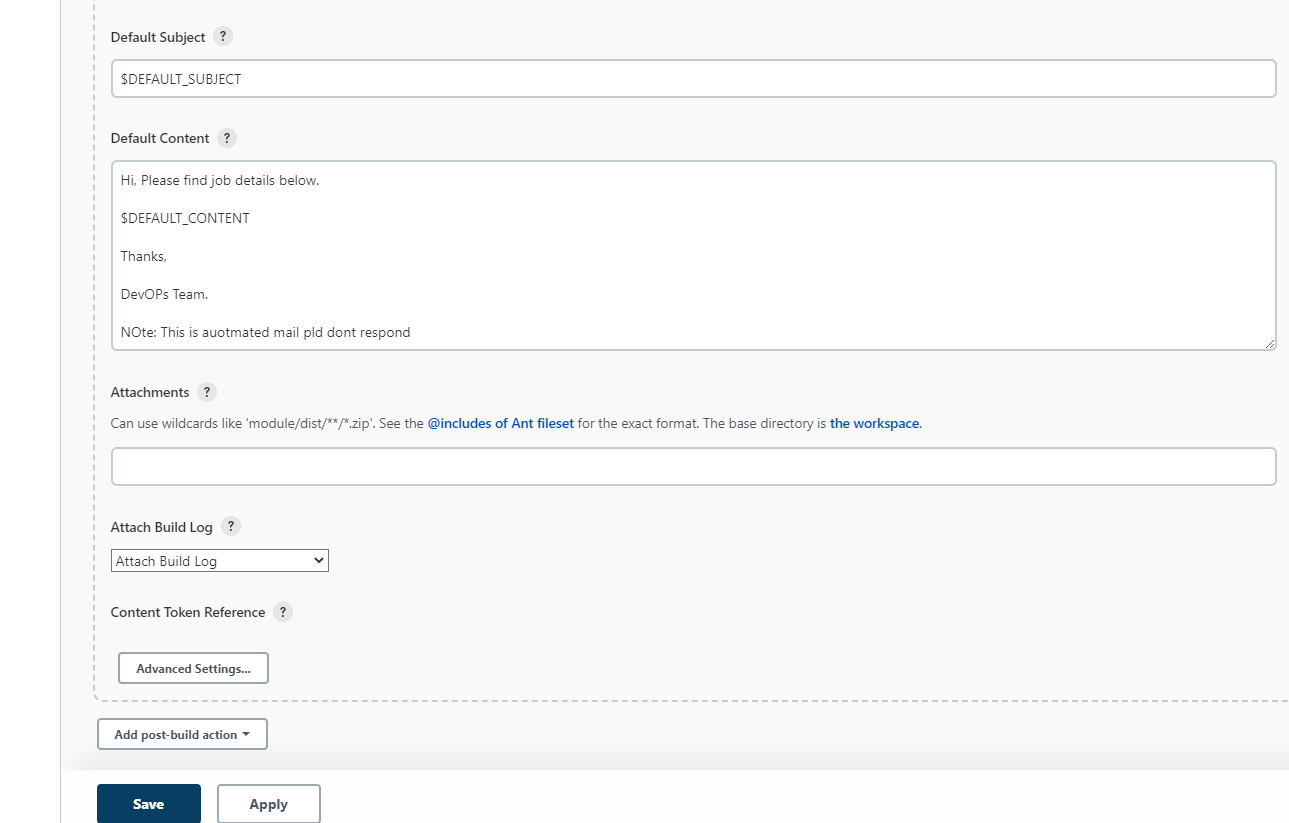
****

****

****

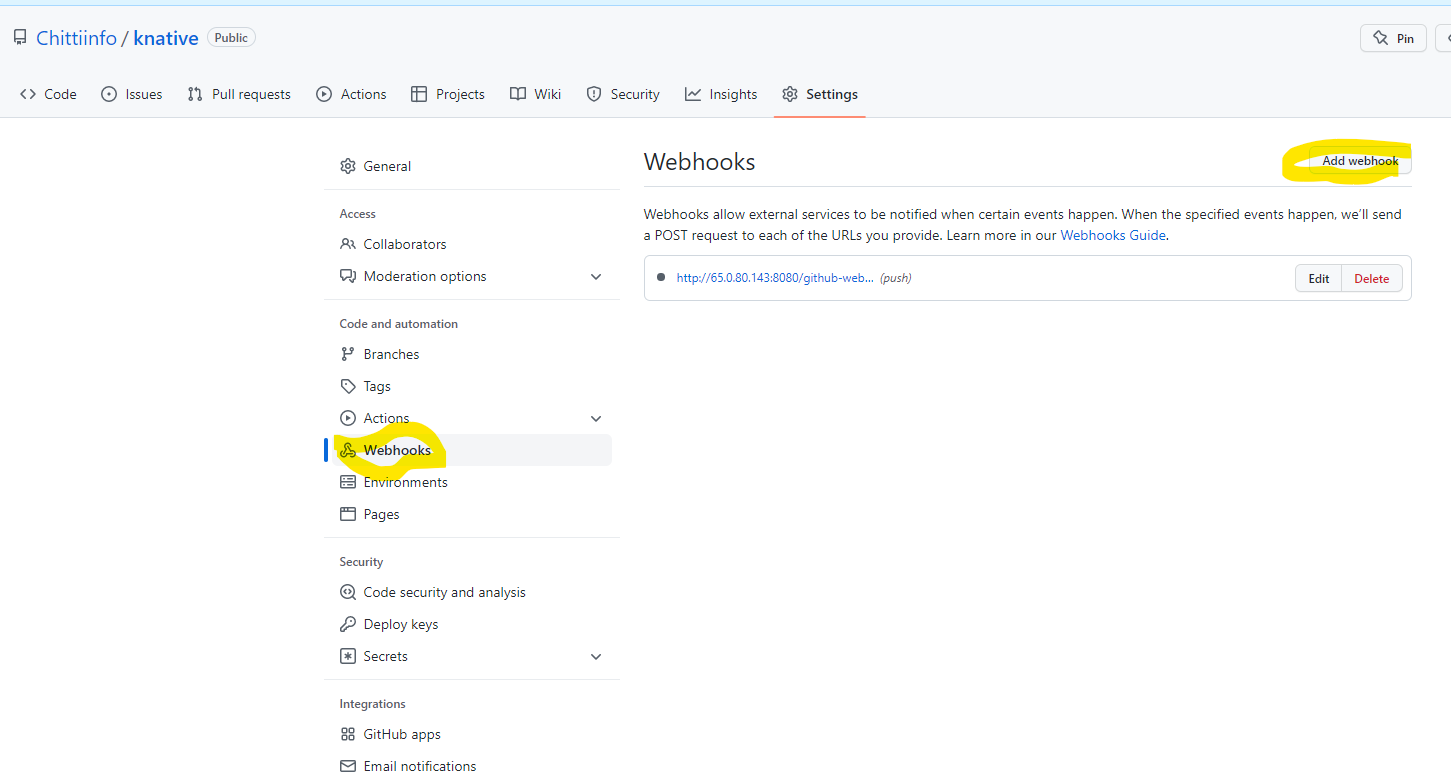
****

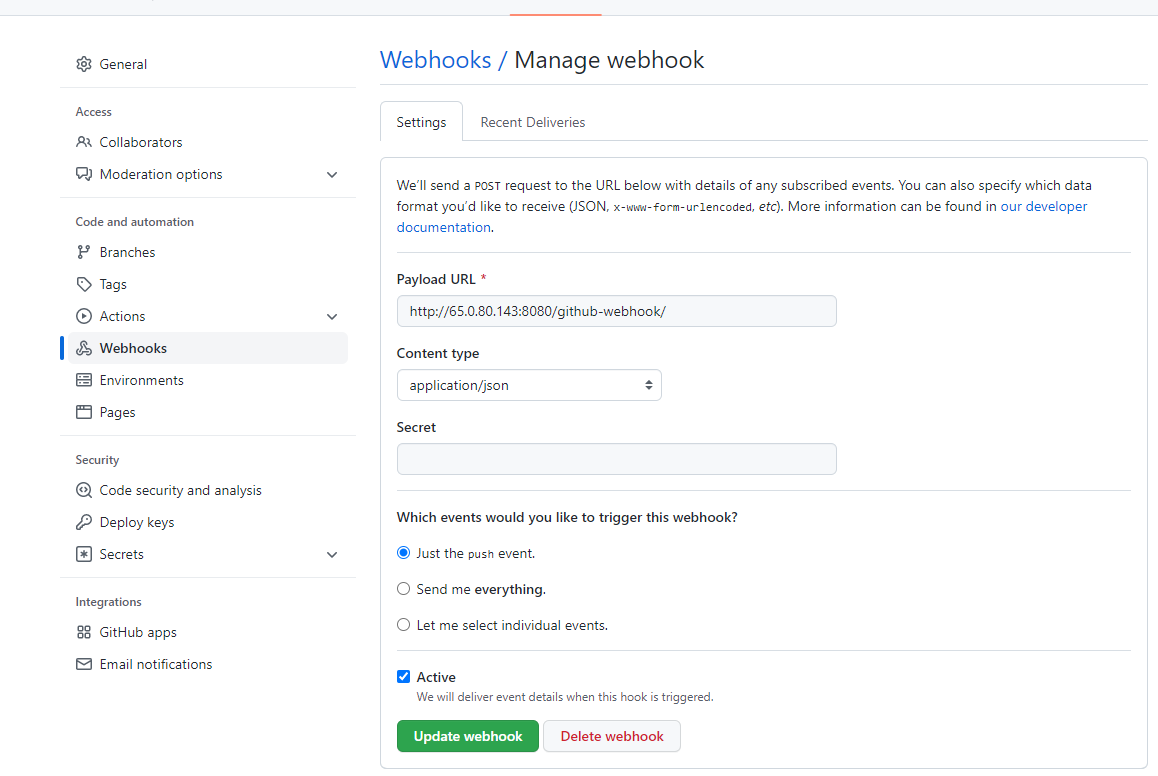
****

****

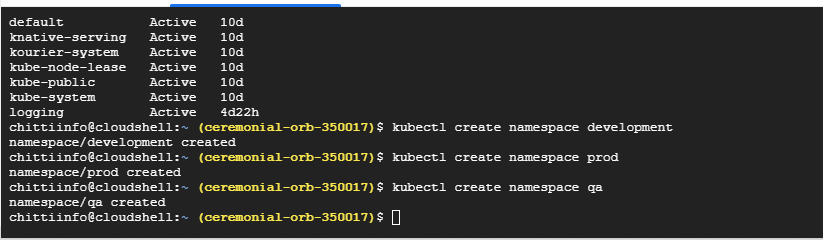
**Automate build with Github code changes:**

Login to you github repository and configure webhooks with Jenkins URL

****

****

**Kubernetes:**

****

**kubectl get configmaps -n knative-serving**

**URLs:**

[**https://medium.com/avmconsulting-blog/namespaces-in-kubernetes-4bac49414770**](https://medium.com/avmconsulting-blog/namespaces-in-kubernetes-4bac49414770)

<https://knative.dev/docs/serving/autoscaling/scale-bounds/#upper-bound>

<https://knative.dev/docs/serving/autoscaling/scale-to-zero/#enable-scale-to-zero>

<https://docs.openshift.com/container-platform/4.7/serverless/cli_tools/kn-serving-ref.html>

**Prod Setup:**

**Steps to perform on local git:**

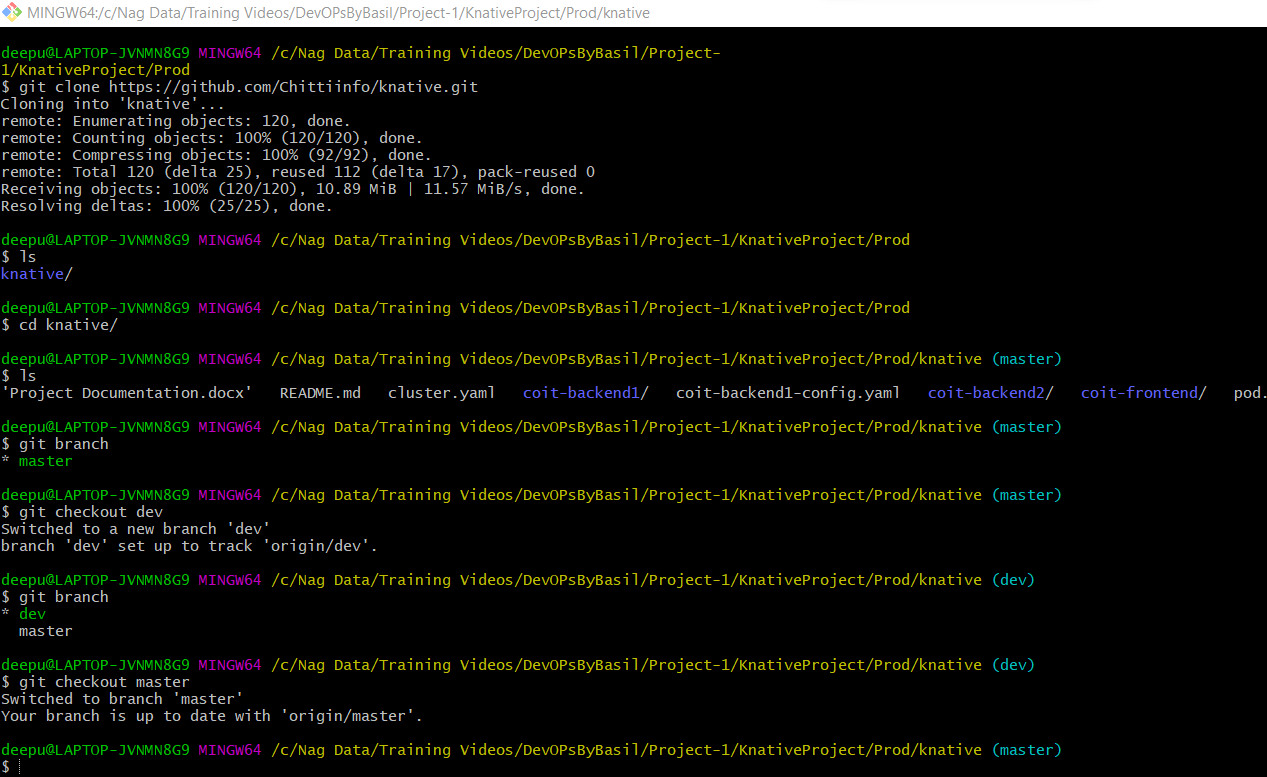
Create a empty folder with name **Prod in your local PC**

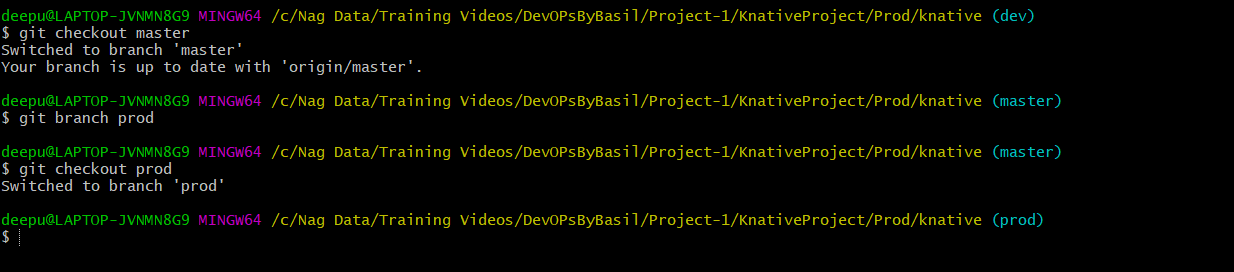
Clone Dev repository.

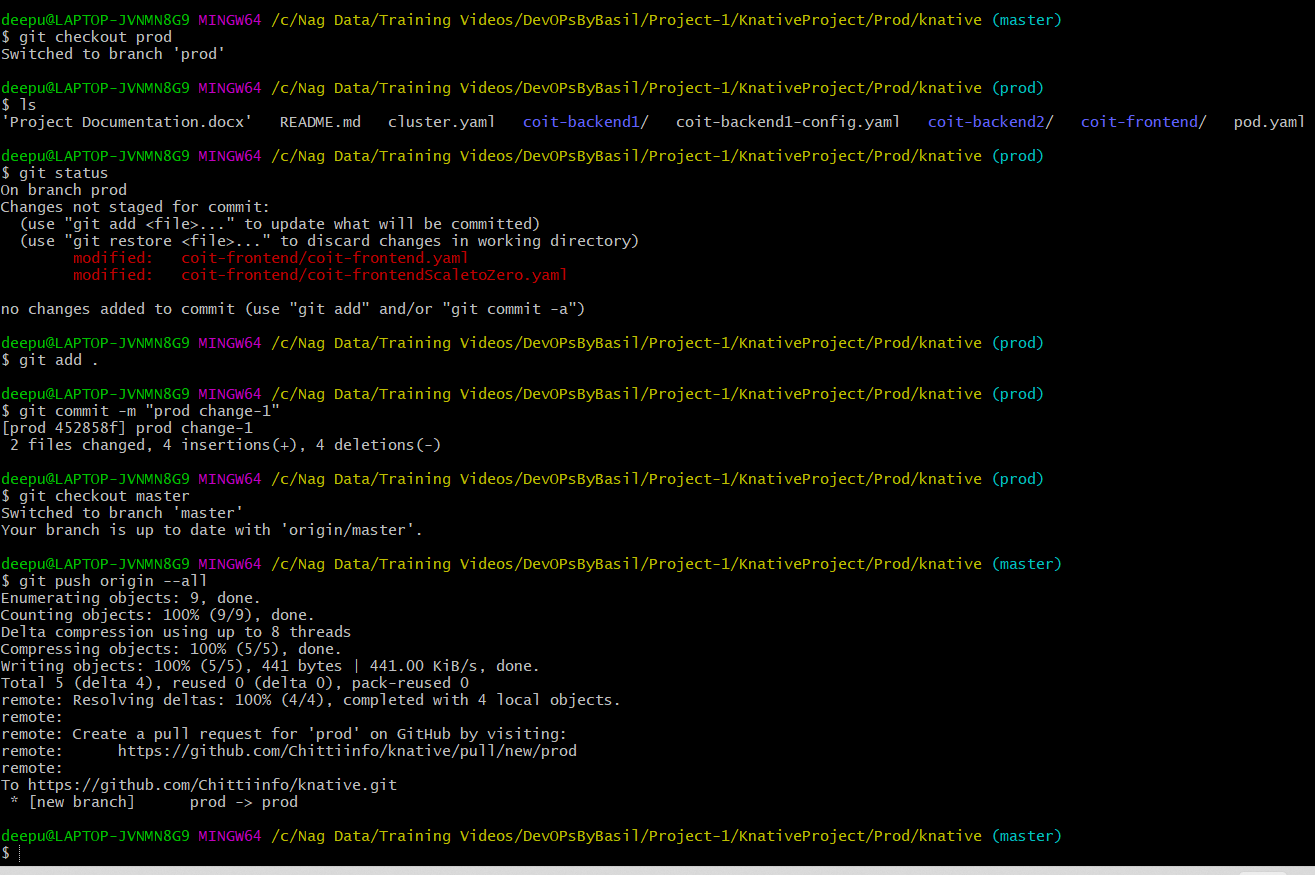
Create a new branch for Prod

Make necessary changes in Prod branch code

Commit and push the changes to Github

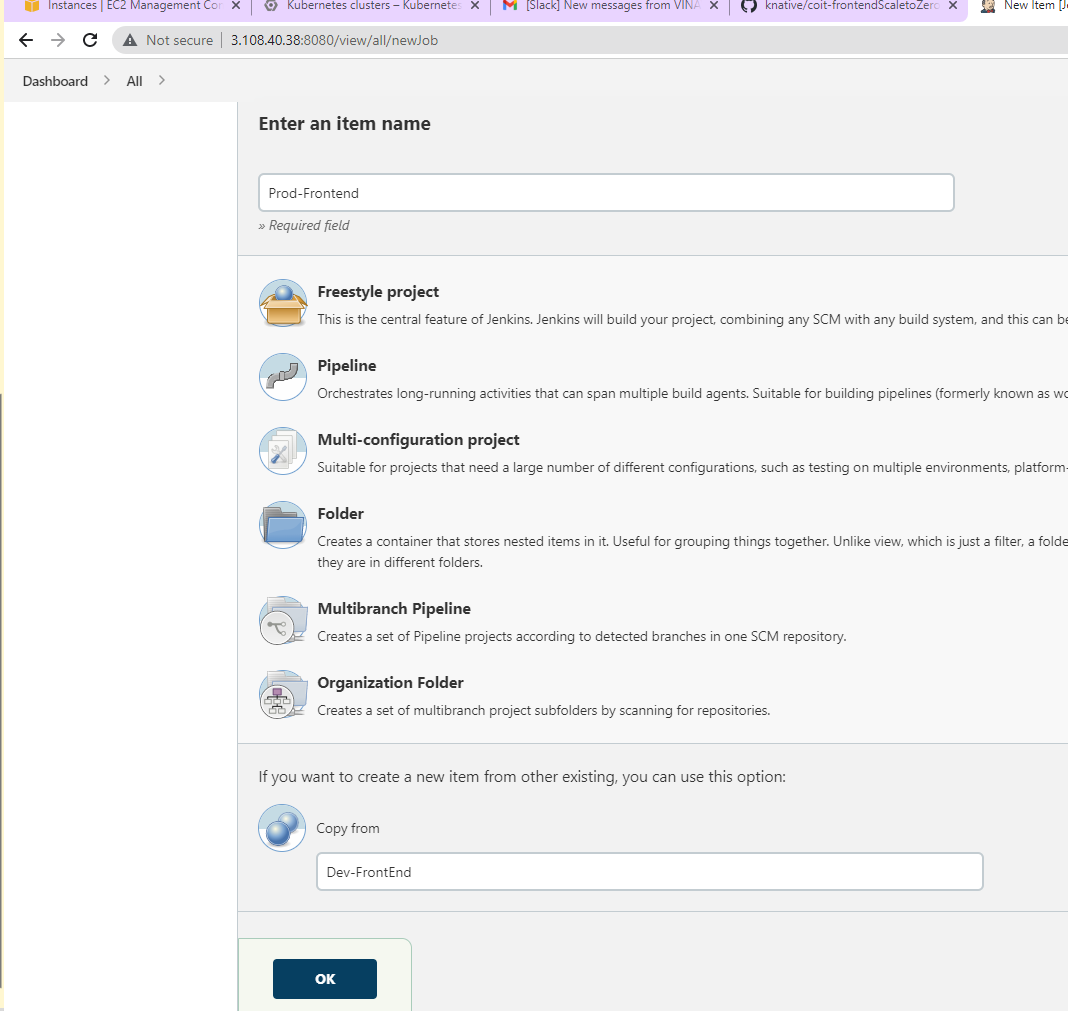






**Jenkins Steps:**

Create a new job and select Copy From existing Dev job.



**Code used in job:**

cd coit-frontend

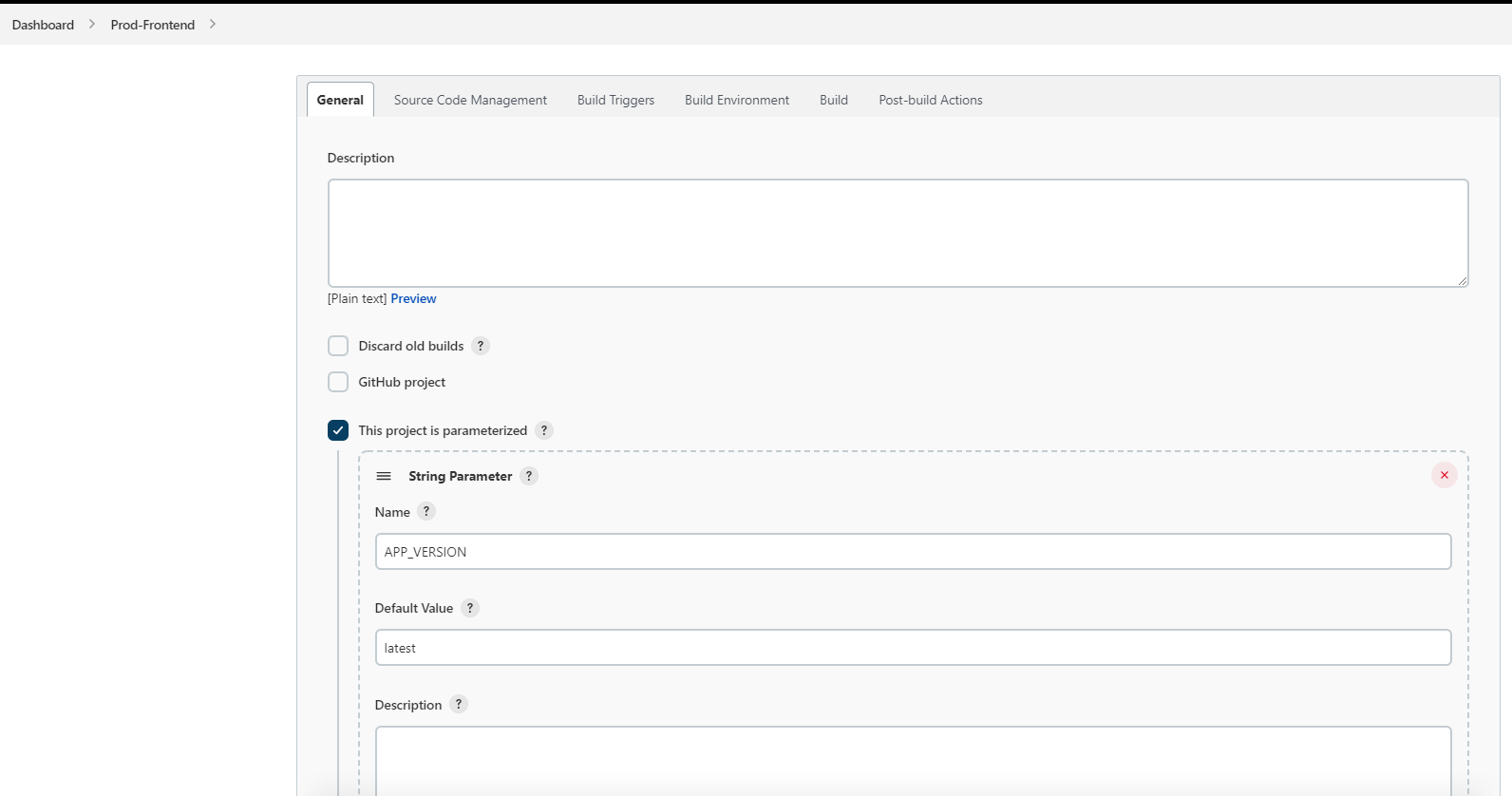
docker build -t chittiinfo/coit-frontend:${APP\_VERSION} . -f Dockerfile-multistage

docker push chittiinfo/coit-frontend:${APP\_VERSION}

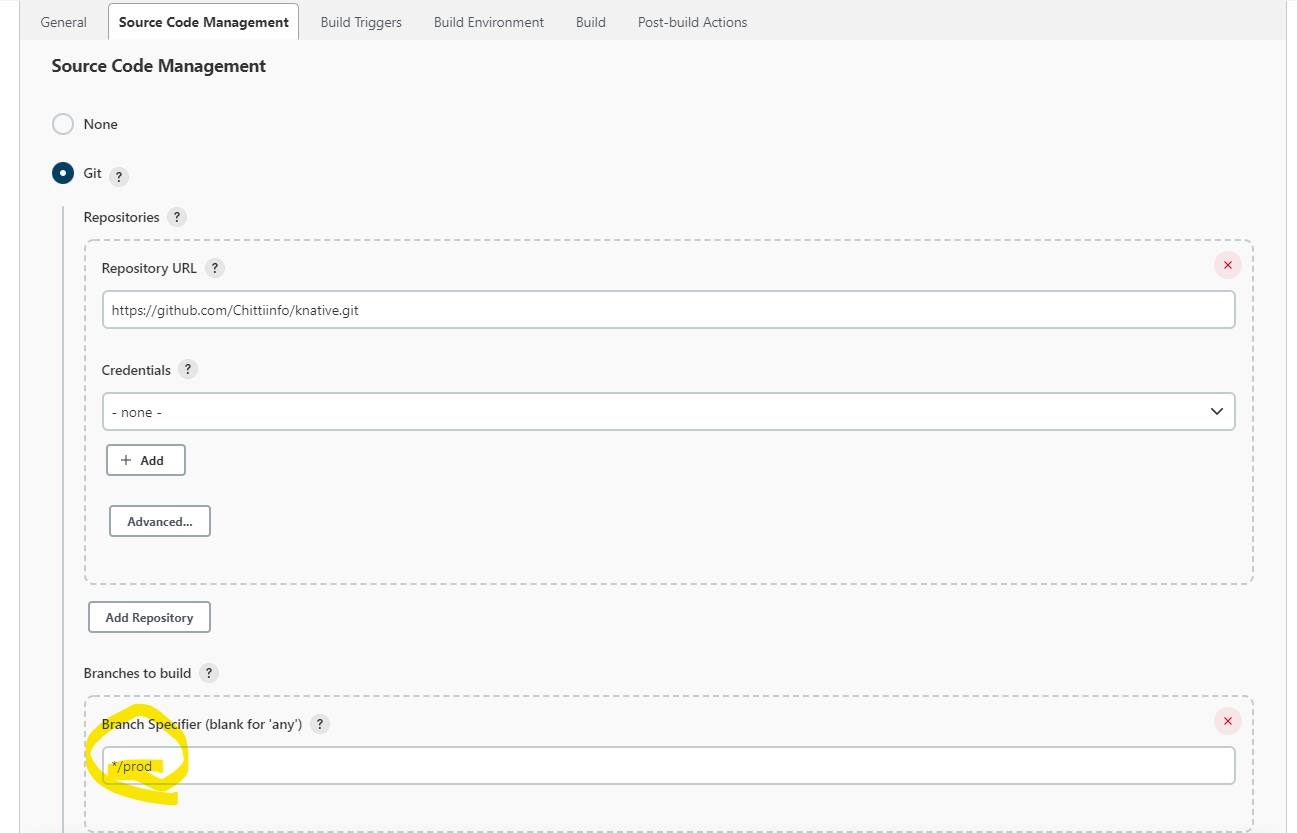
cd coit-frontend

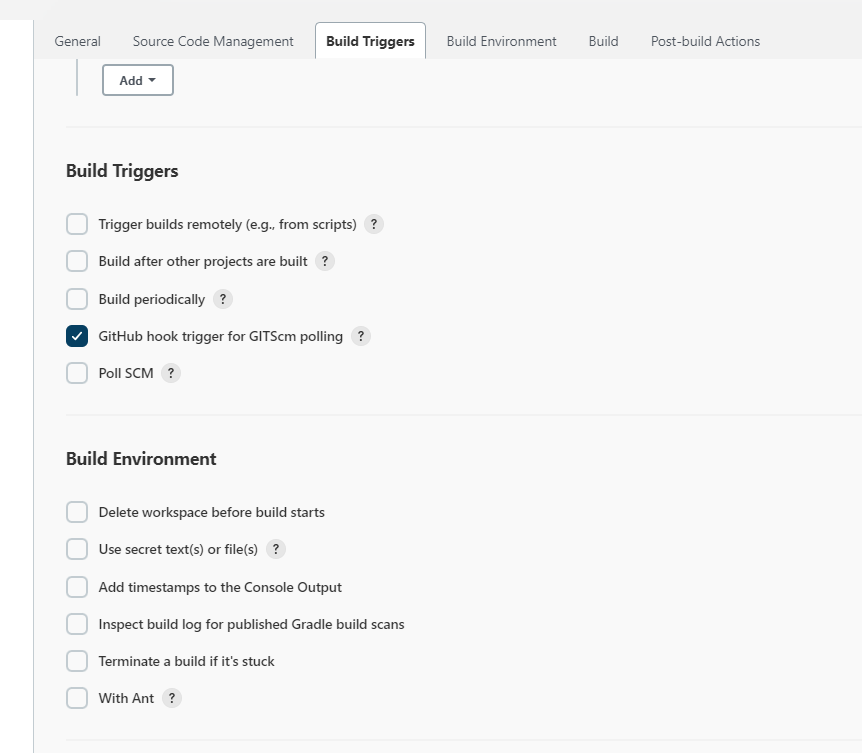
kubectl apply -f "coit-frontendScaletoZero.yaml"

kn service create coit-frontend --image chittiinfo/coit-frontend:${APP\_VERSION} --port 80 --scale-min 5 --scale-max 100 -n prod

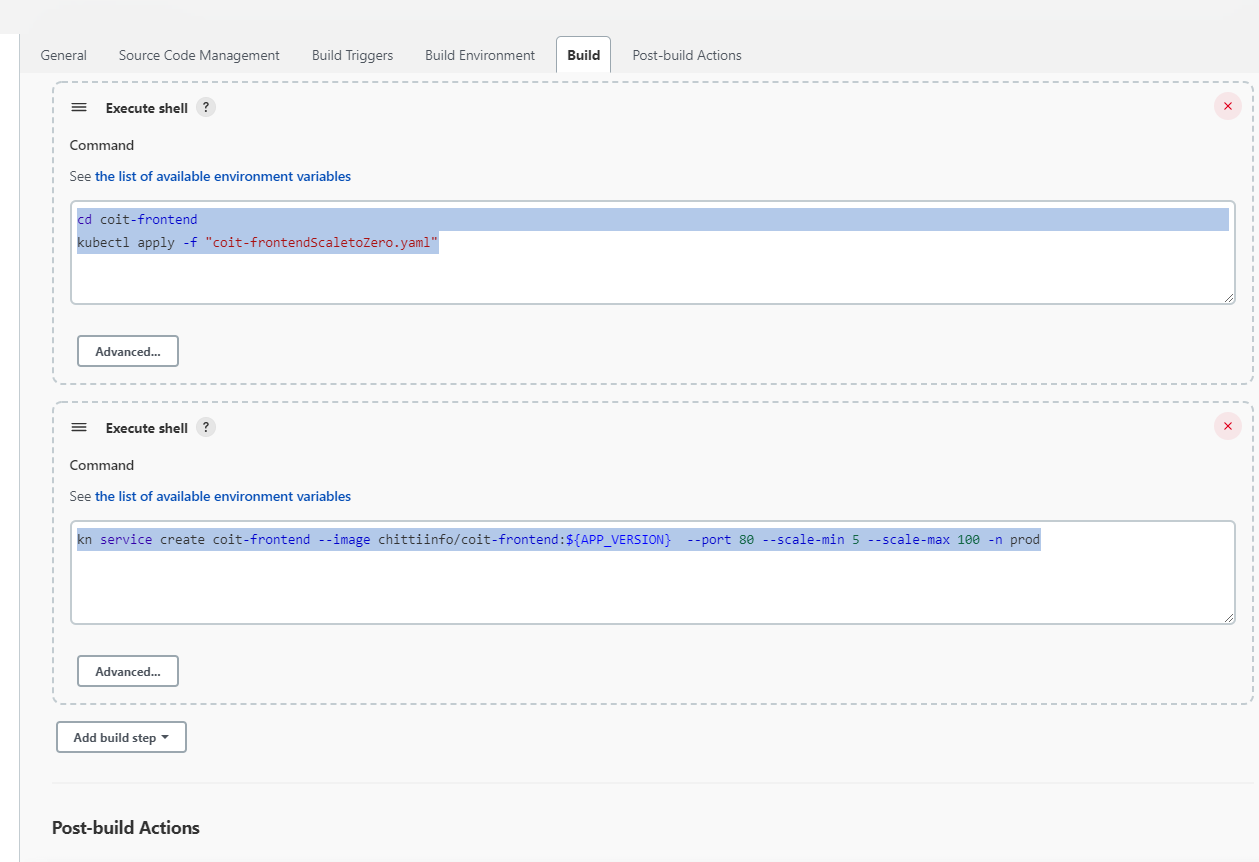


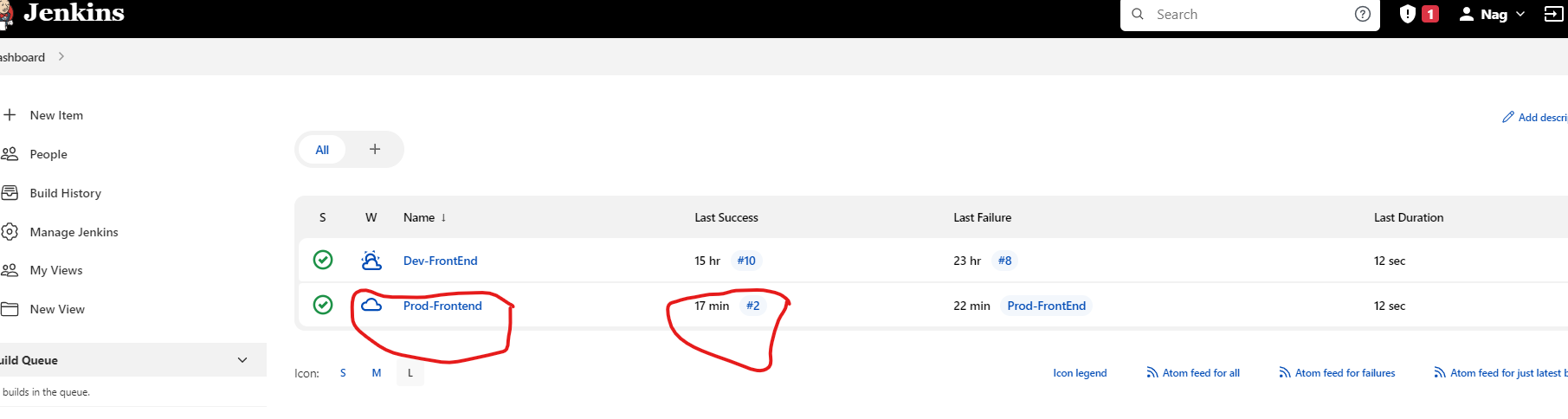
Make sure you define Branch as \*/prod as that’s where we committed our code.





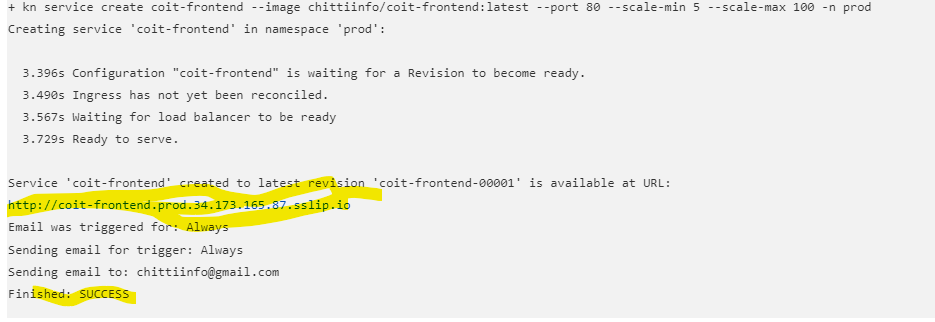






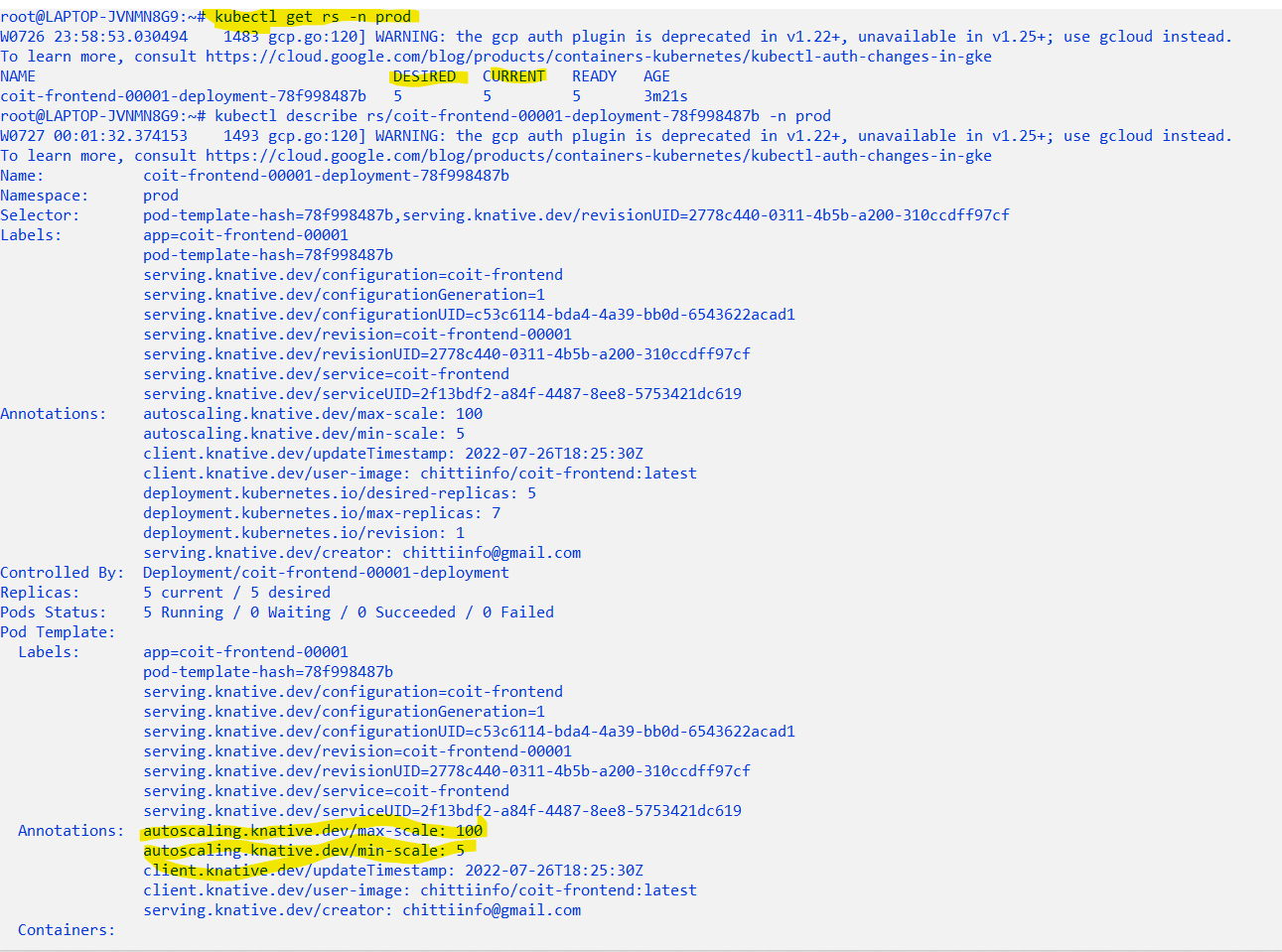
Build History:





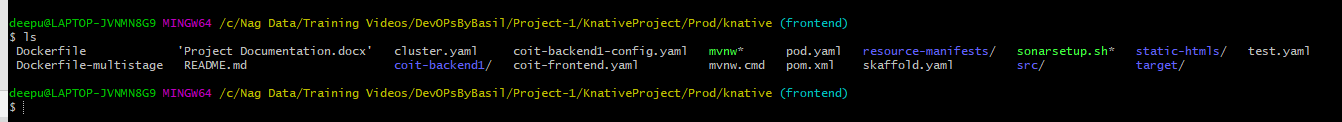
**Validation at Kubernetes:**

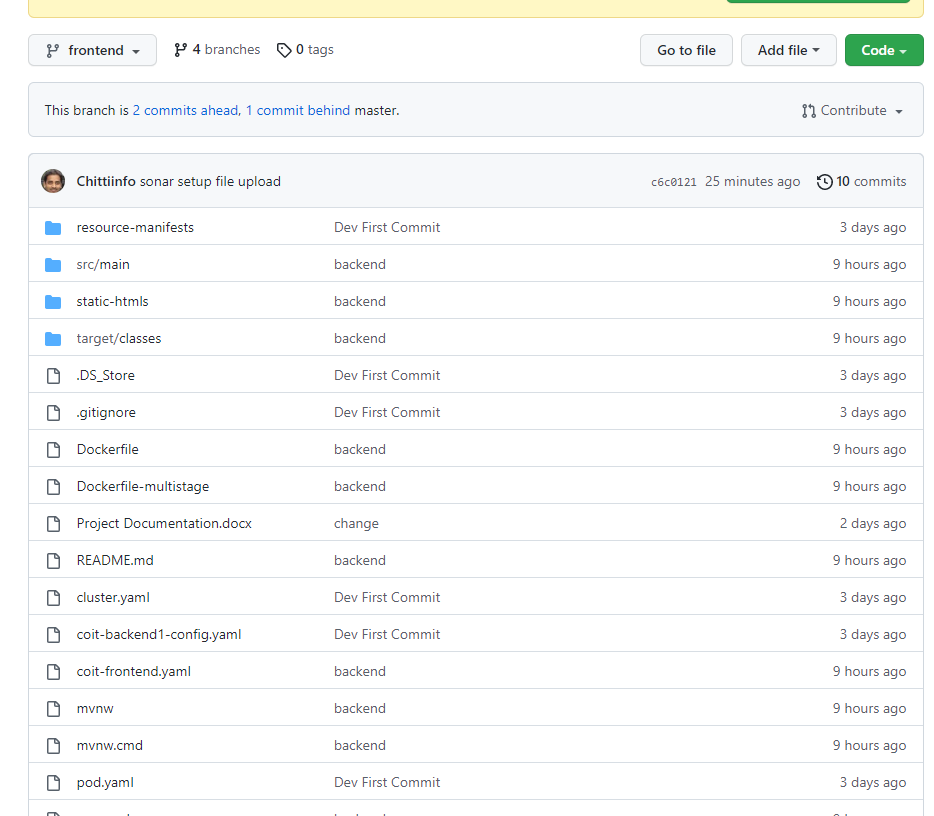




**Testing**

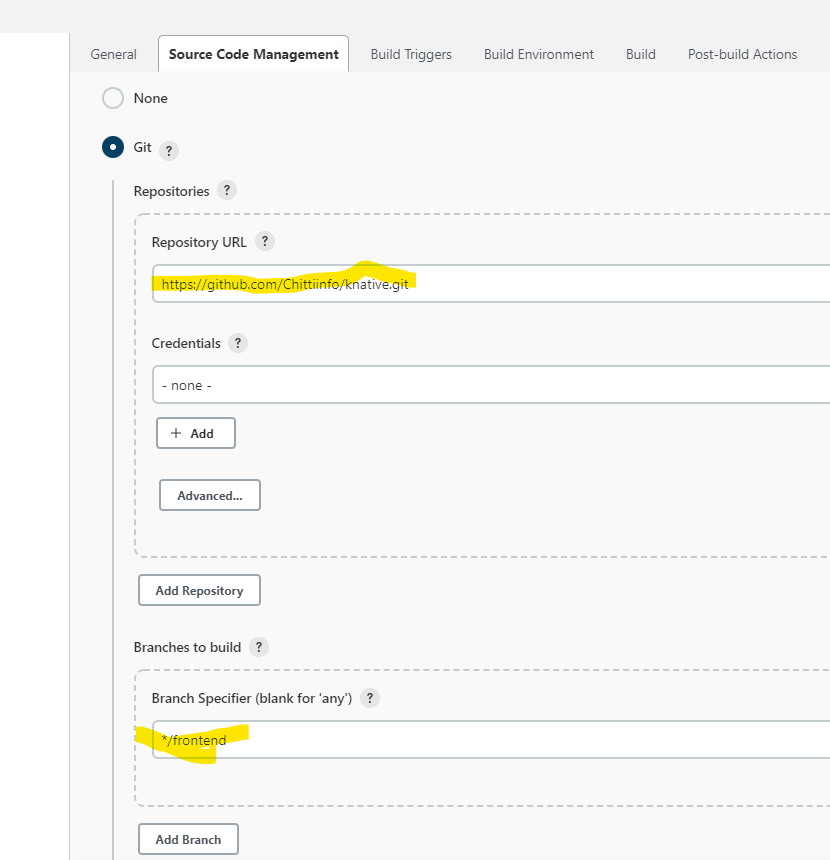
In my case, I am targeting Unit Testing, Integration Testing and Static Code analysis only for Backend1 java code. Ensure you have a branch with only java code to implement below steps.

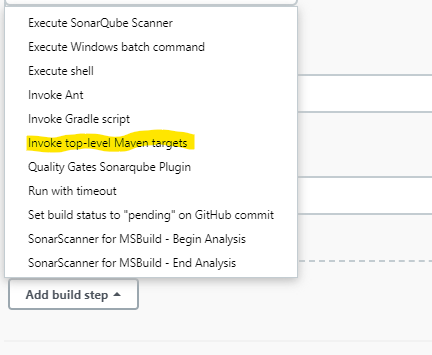


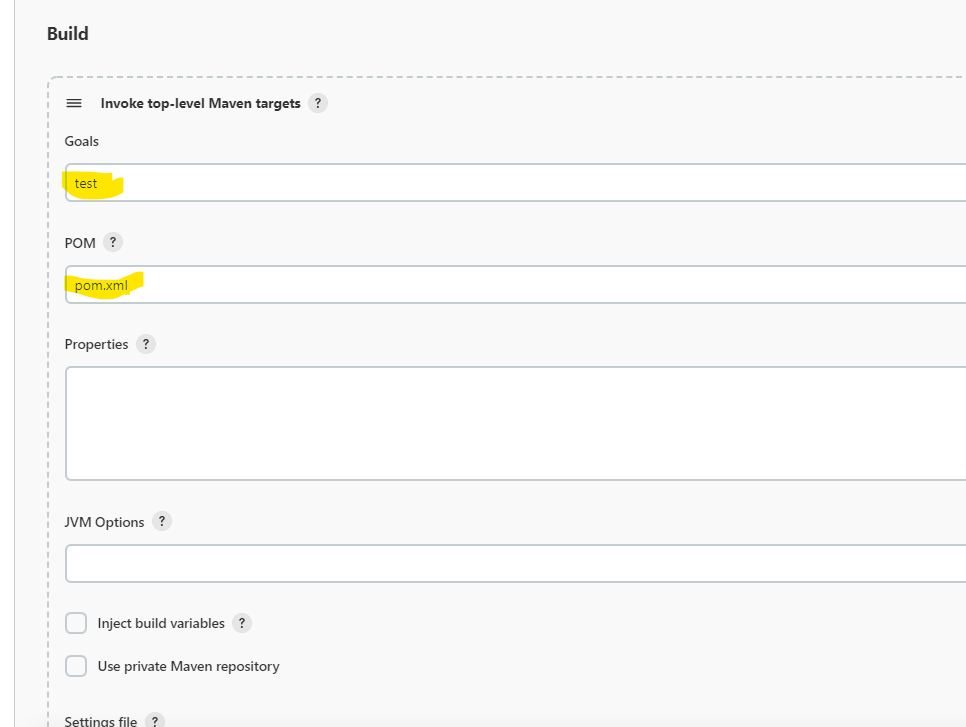


**UNIT TEST:**

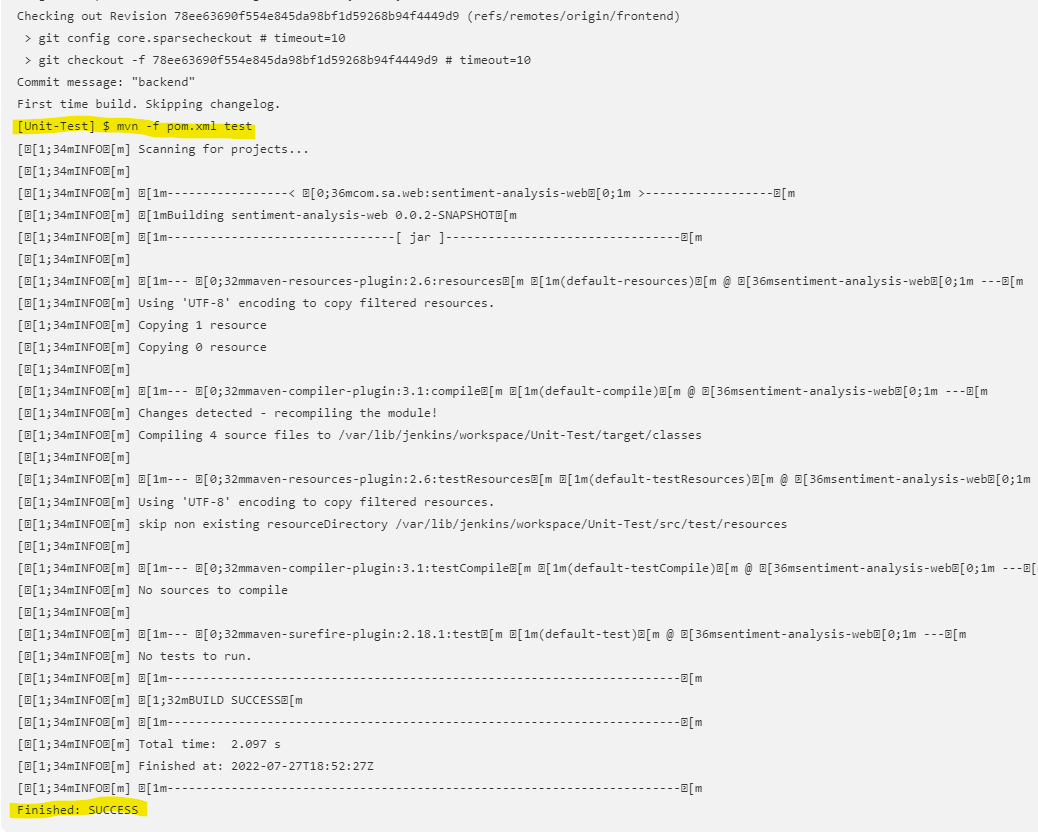
****

****

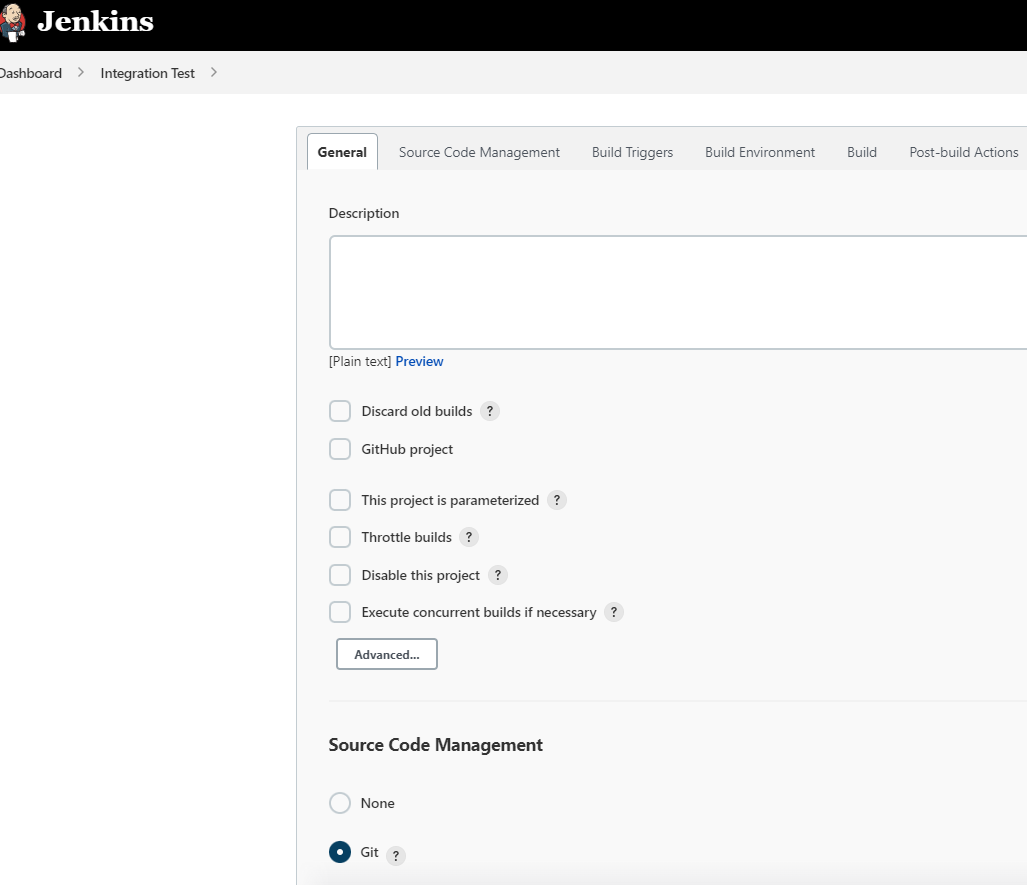
****

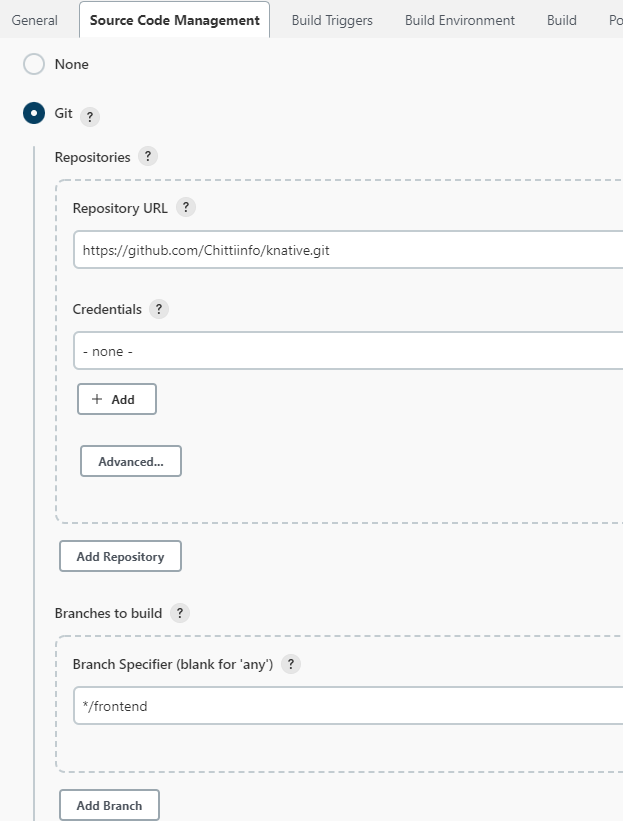
****

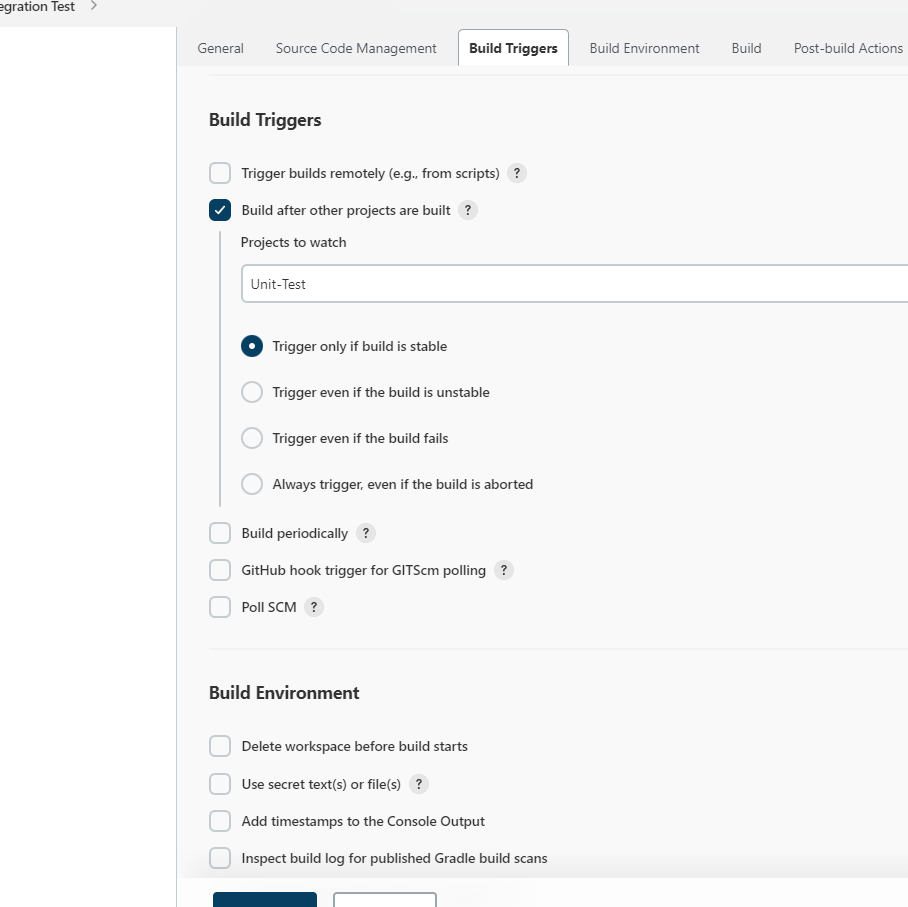
**Unit test build log:**

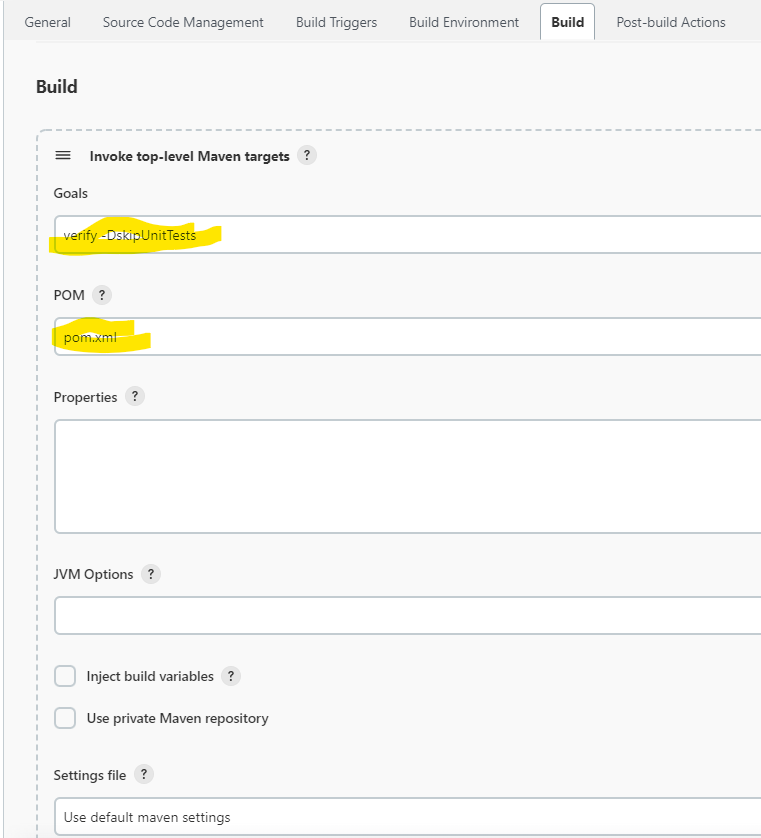
****

**Integration Testing:**

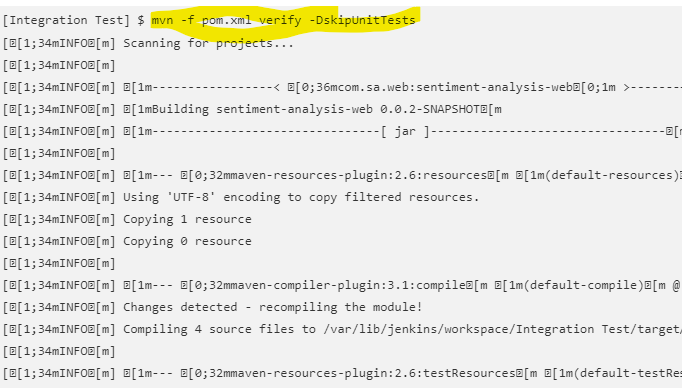
****

****

****

****

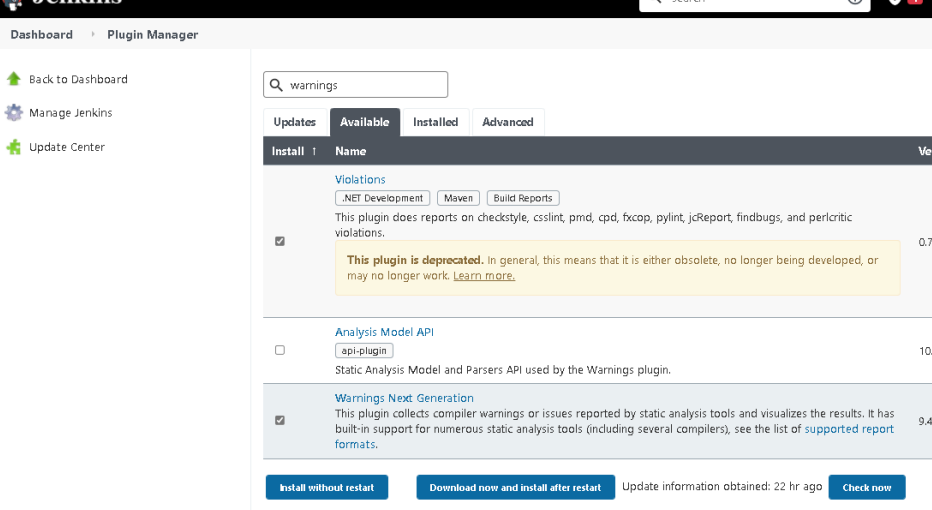
**Build log:**

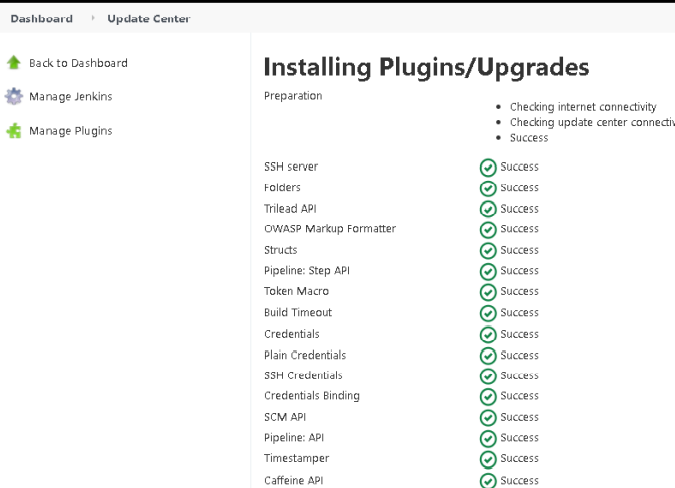
****

****

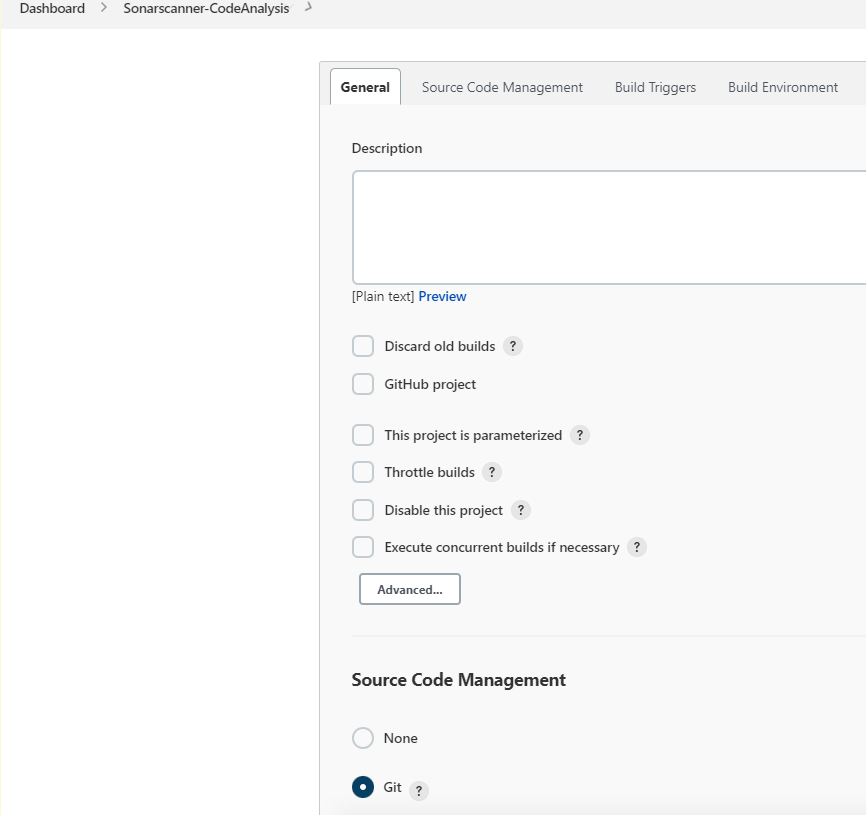
**Static Code Analysis:**

Select copying settings from previous pipeline and Click on ok Install jenkins plugins : → Warnings next generation plugin

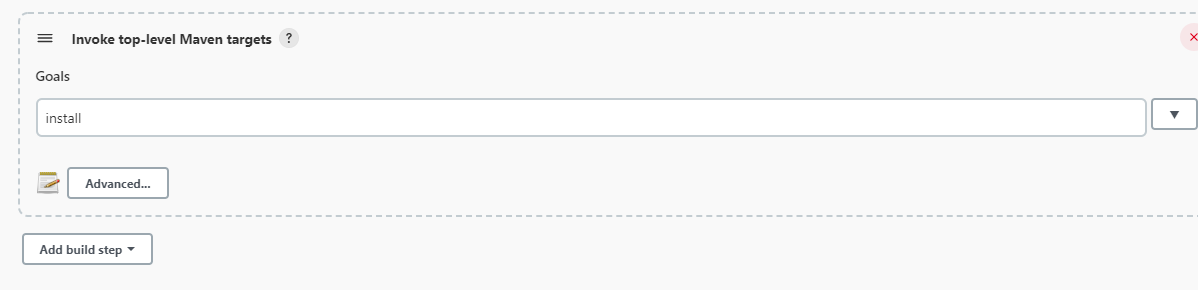
****

****

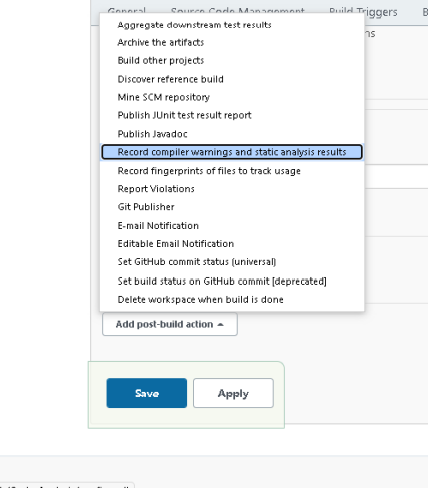
**Job setup:**

****

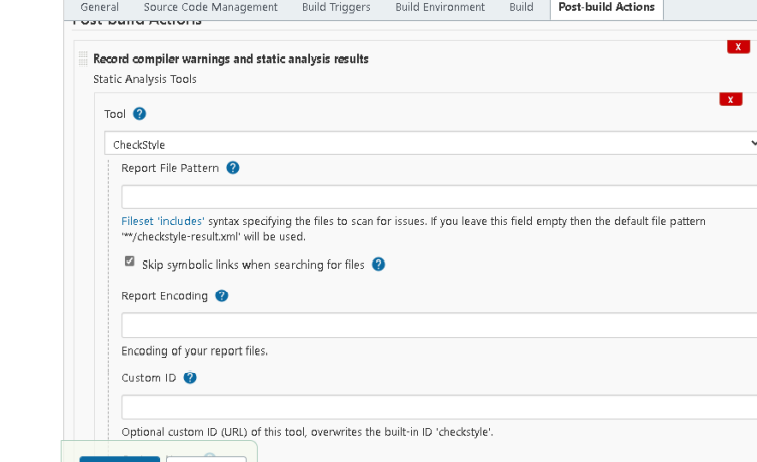
****

****

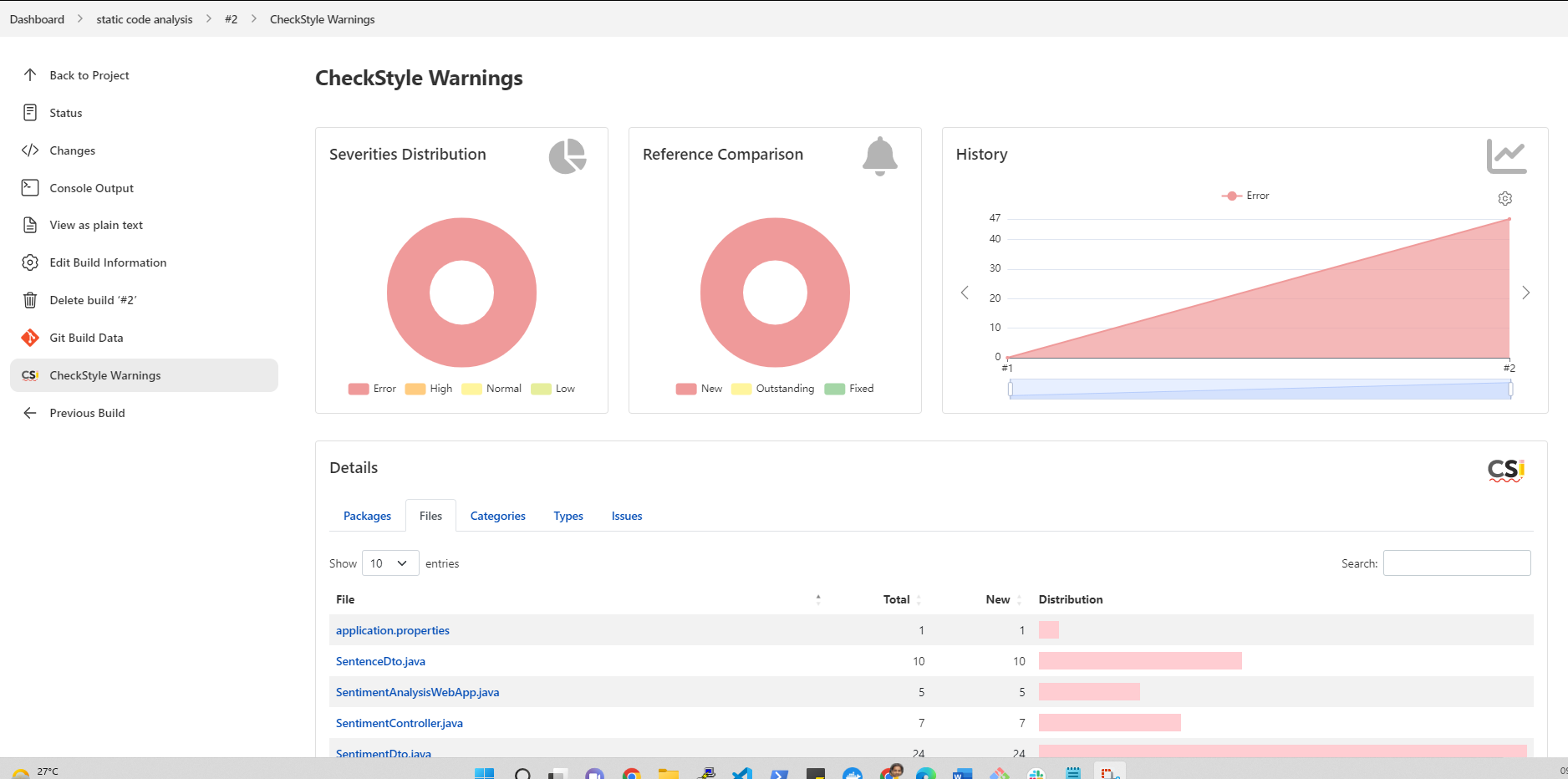
Add a post build action to record compiler warnings and static analysis results

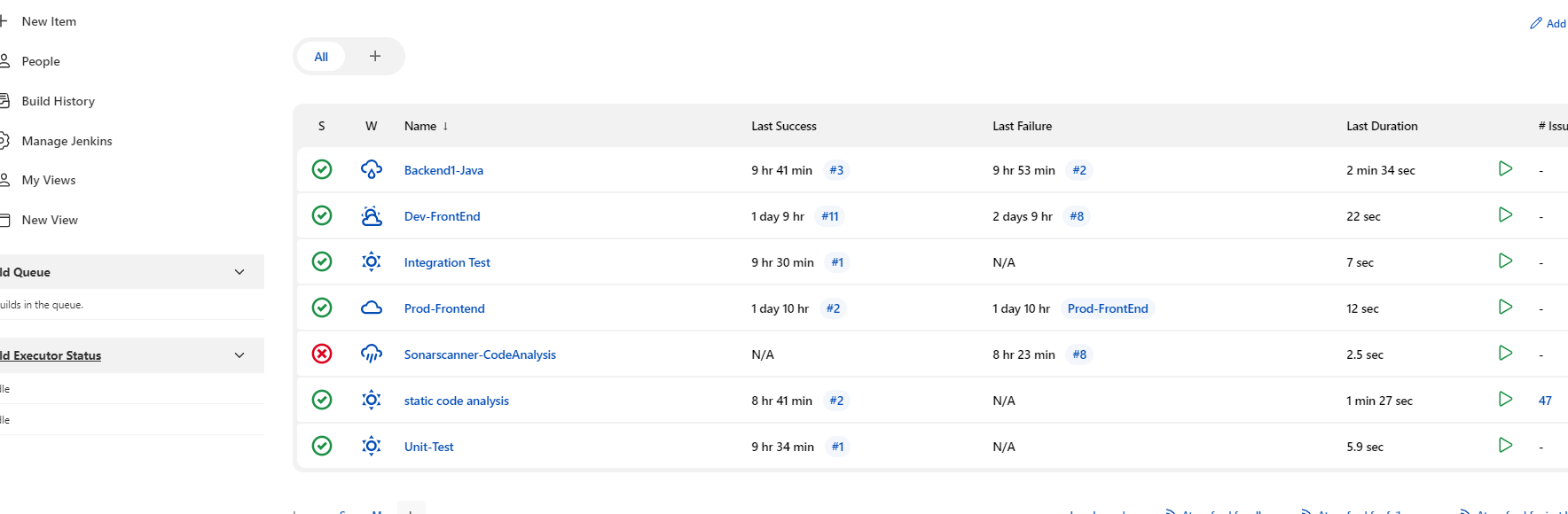
****

In the tools option, select Checkstyle.

****

Run the job with other settings same. After success observe the checkstyle results as shown below.

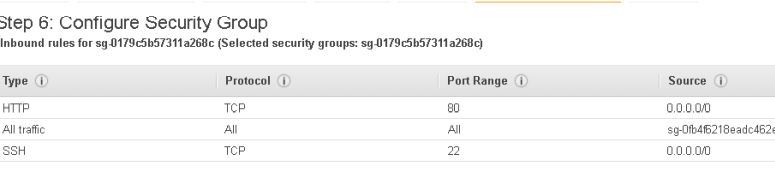


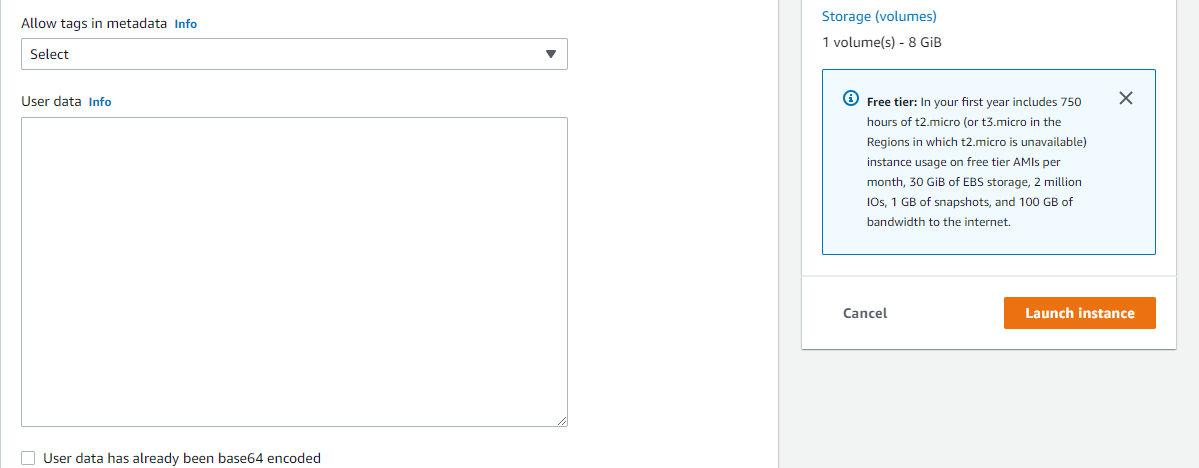


**SonarQube setup**

Create an aws instance with below user data and make sure all pors are opened b/w Jenkins and sonar qube server.

<https://github.com/Chittiinfo/knative/blob/c6c0121d96665c6c360fae6ce08f753a3c296689/sonarsetup.sh>





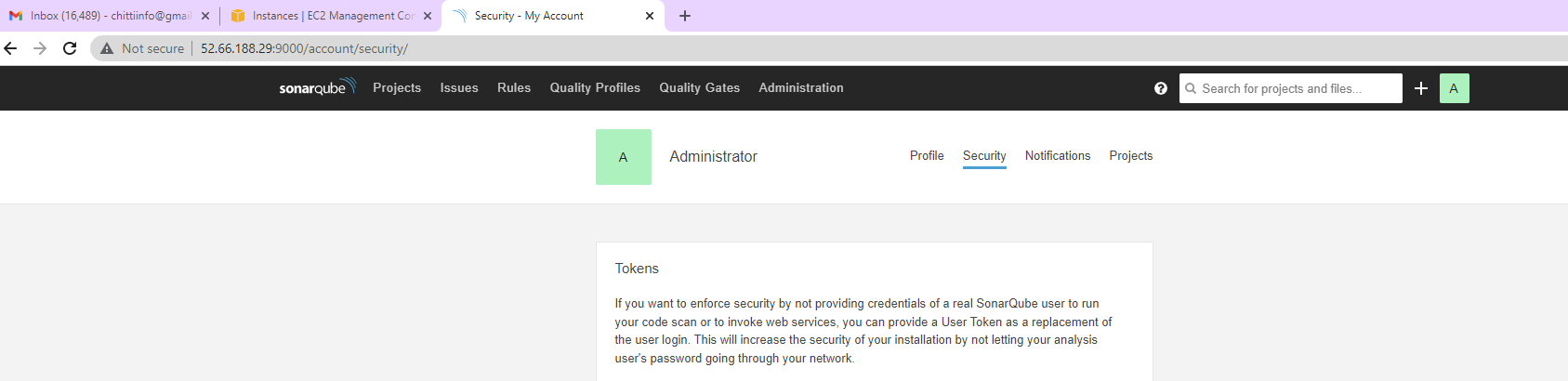


Login to sonarQube with public IP.Usually Sonar Qube works on port 9000, however as we are installing it along with Nginx, we need not provide any port number after public ip in the browser to access sonar

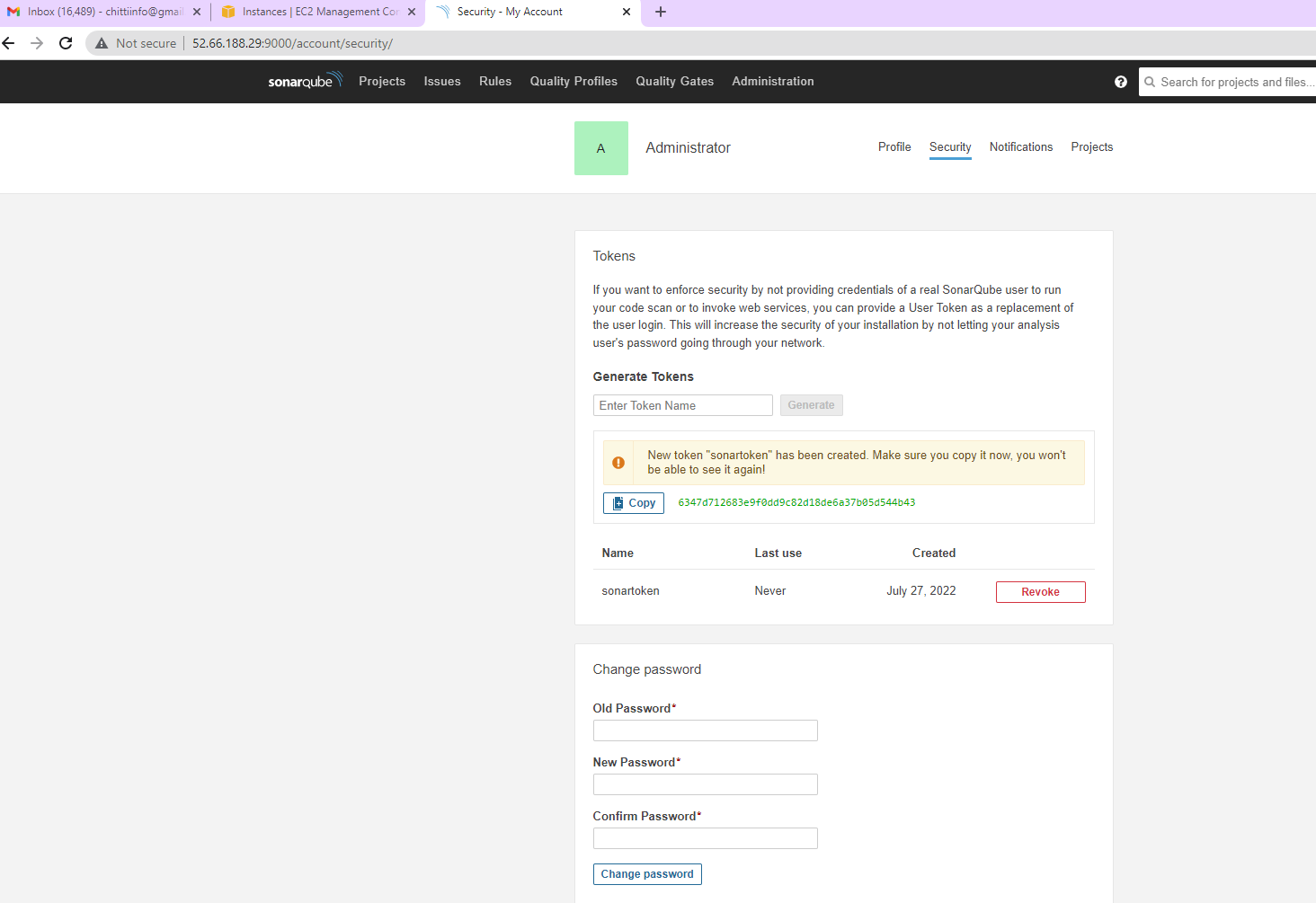
By default password is admin.



Change the default password afterwards.

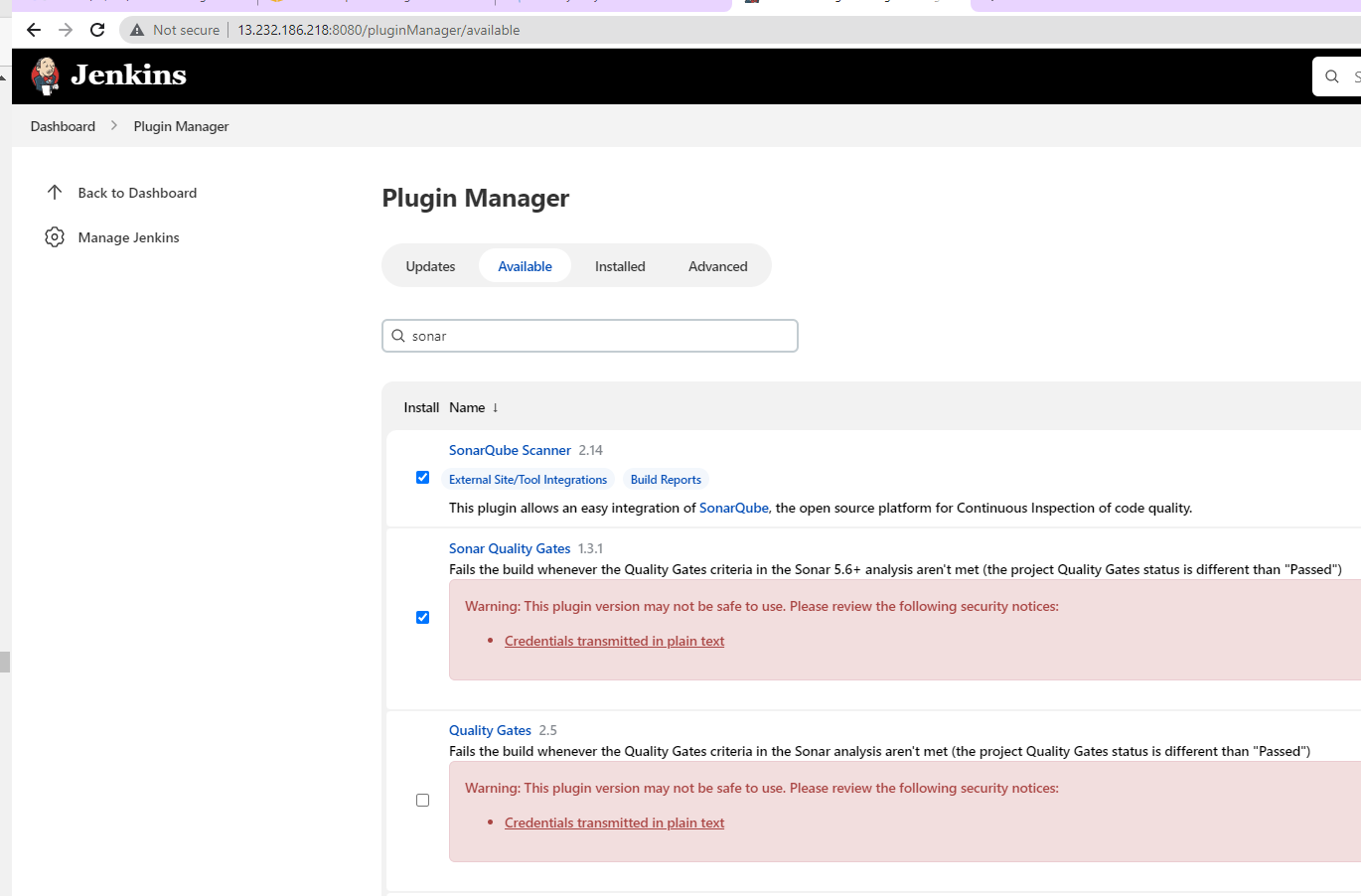
****

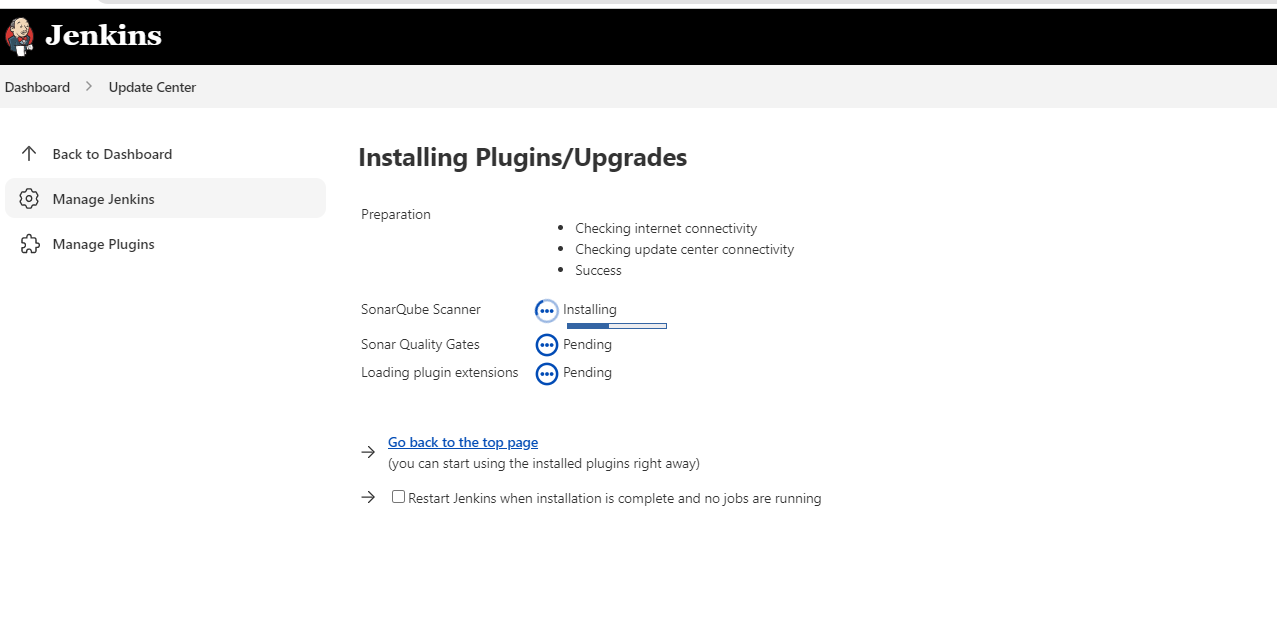
Go to Administrator → my account → security→ Generate tokens

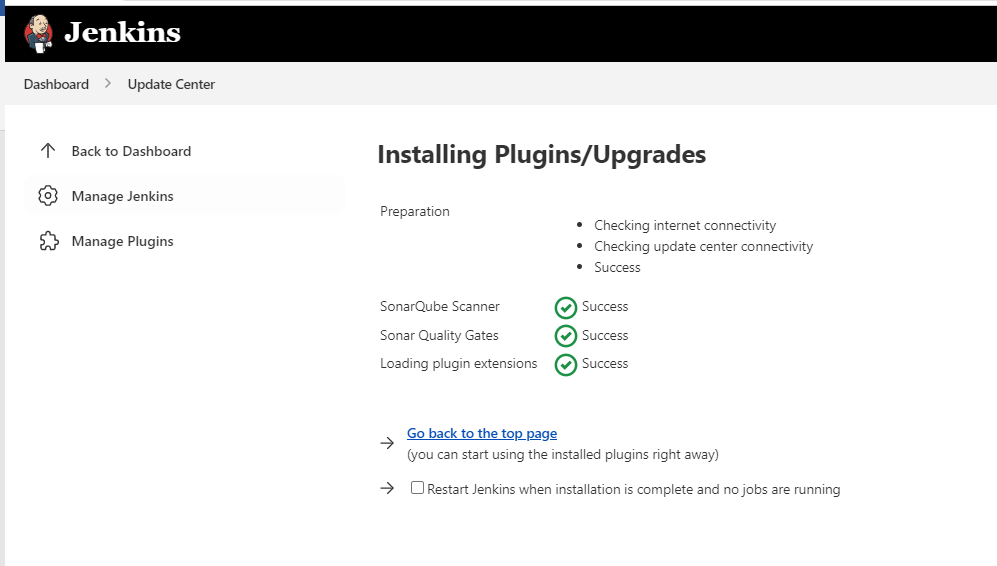
****

Create a token by the name sonartoken. Copy the token and use it in jenkins.

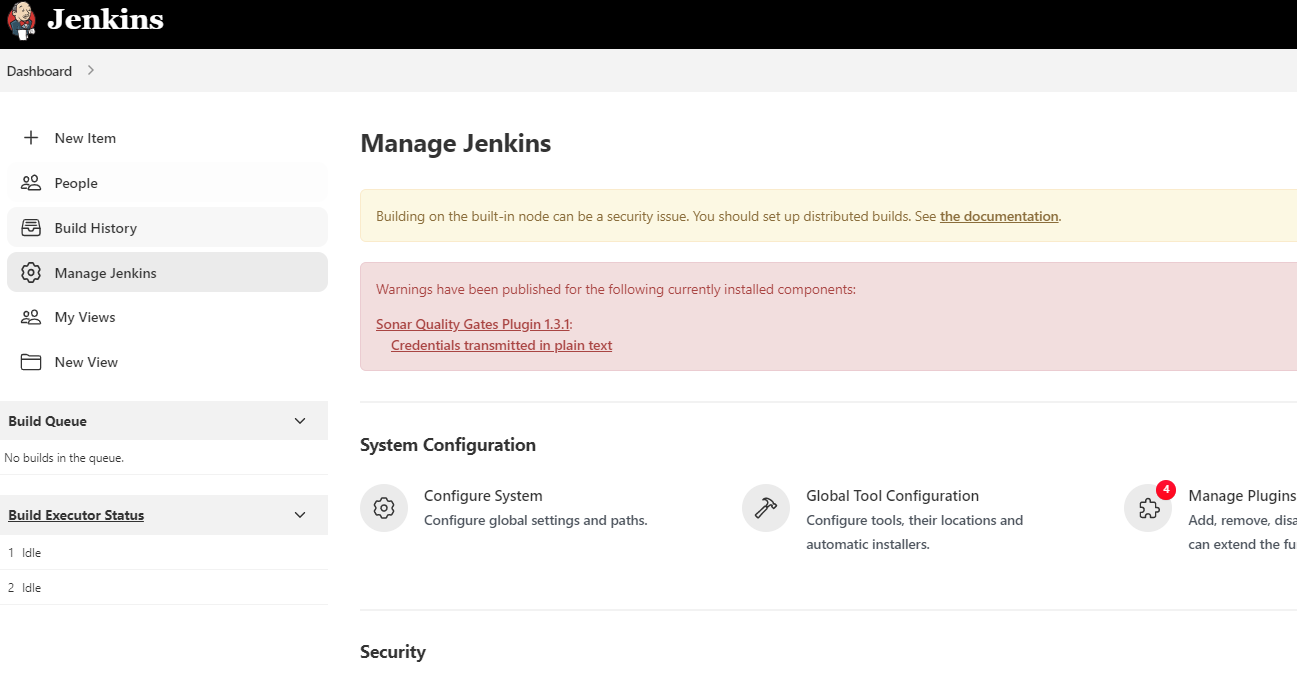
Get sonar plugin in Jenkins: Jenkins → manage jenkins → manage plugins → Available

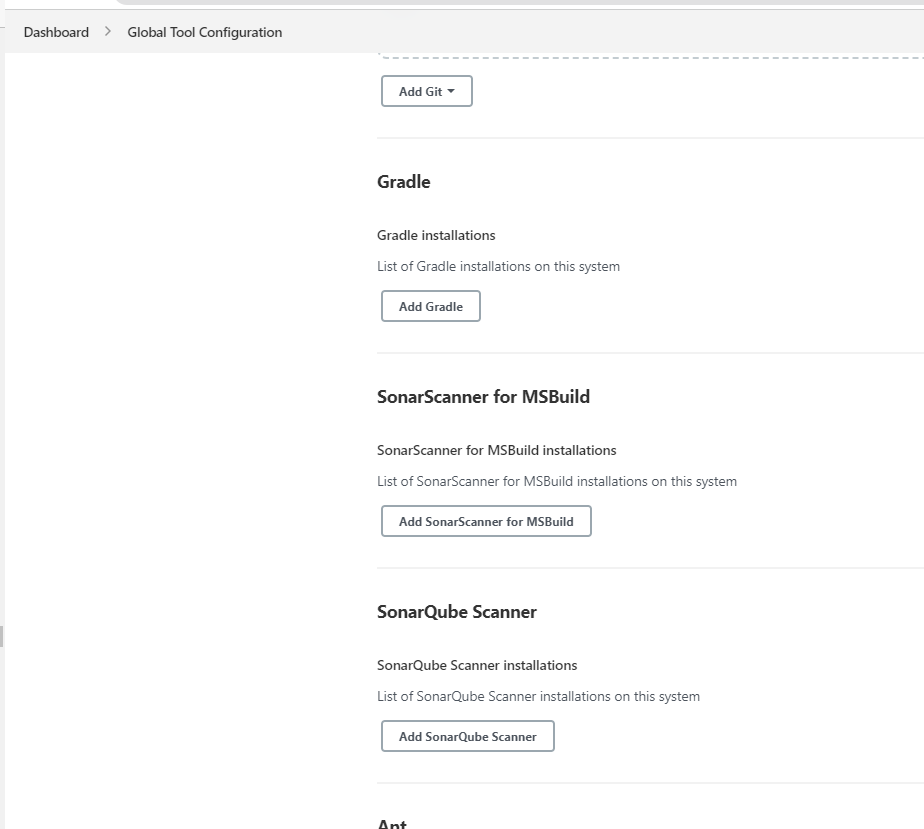
****

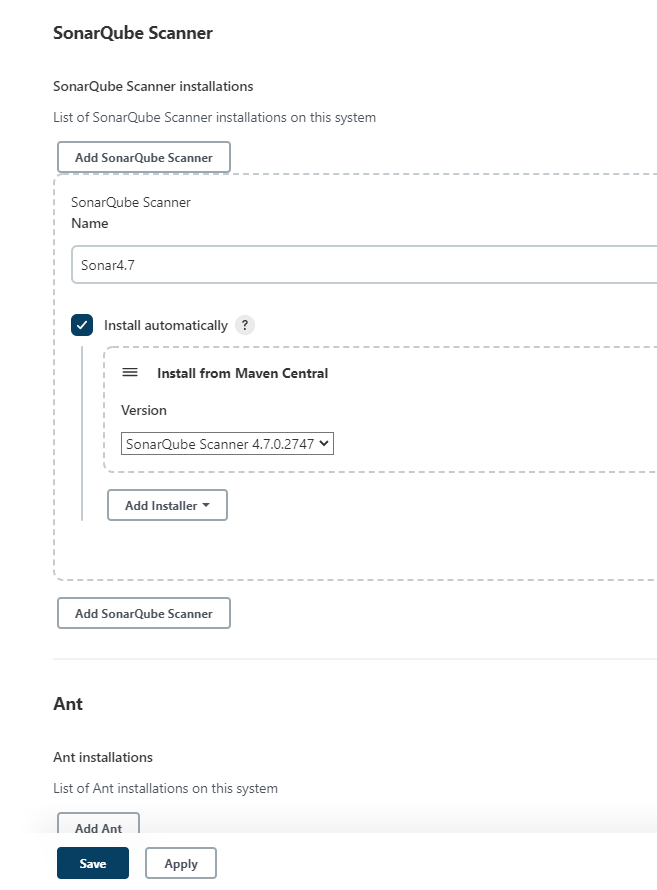
****

****

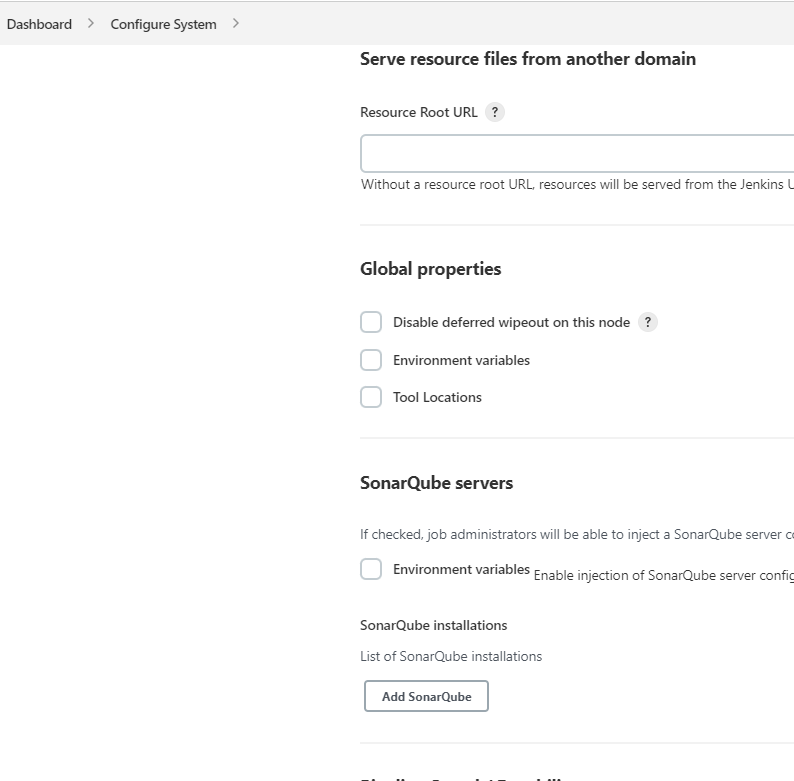
Configure sonar in jenkins: Global tool configuration→

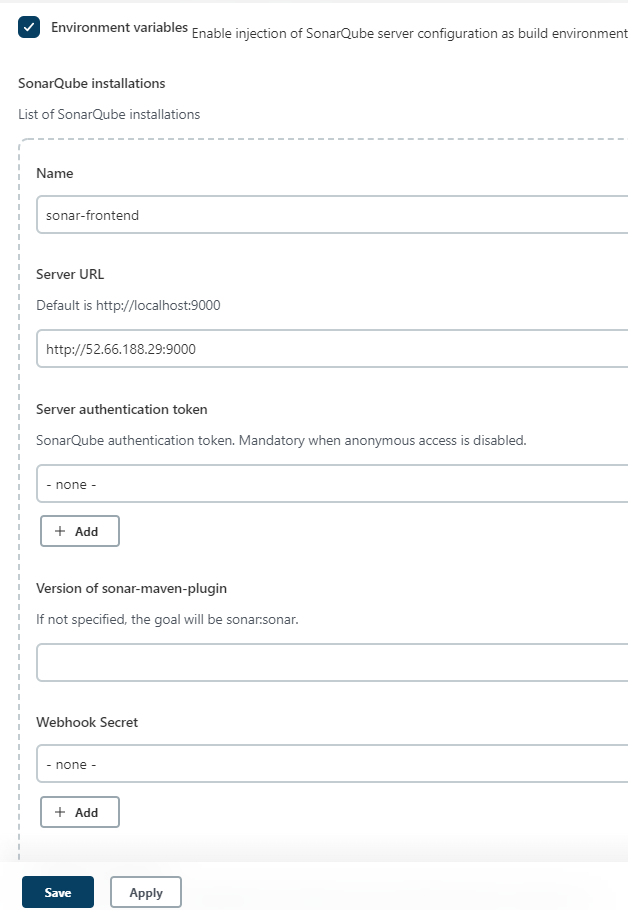
****

****

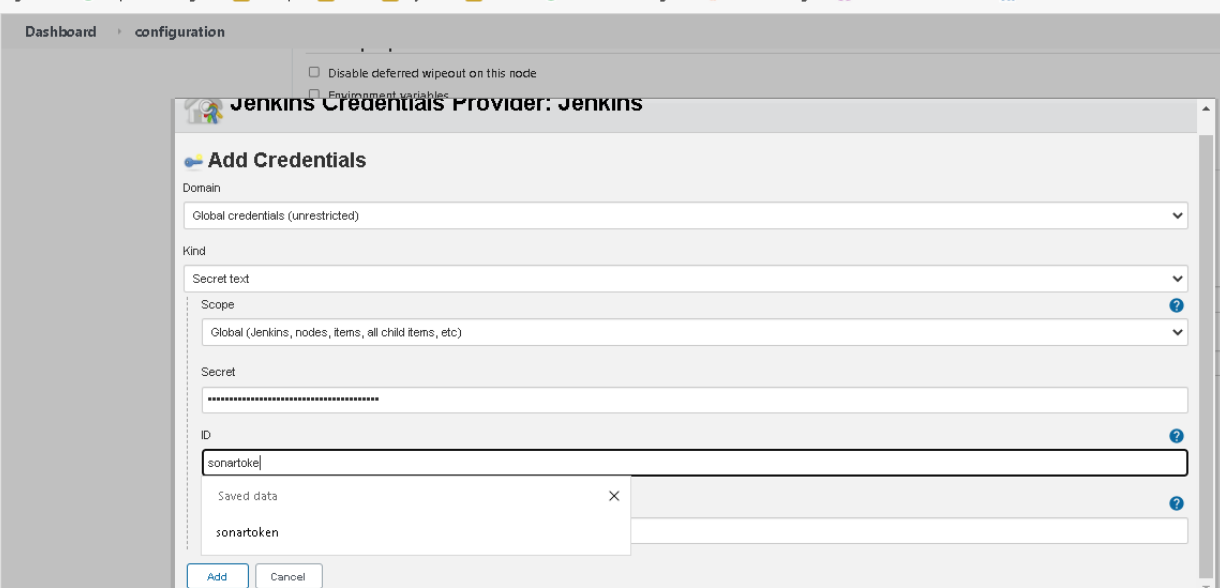
****

Goto jenkins→ configure system→ add sonar and enable environmental variables



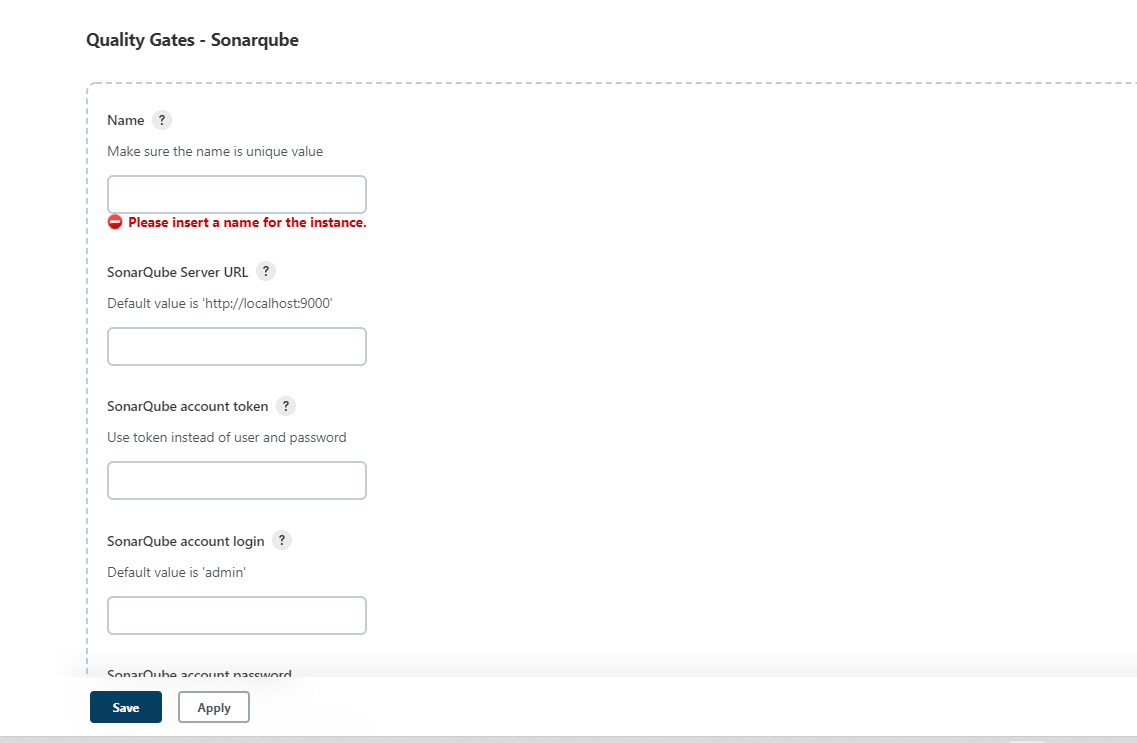


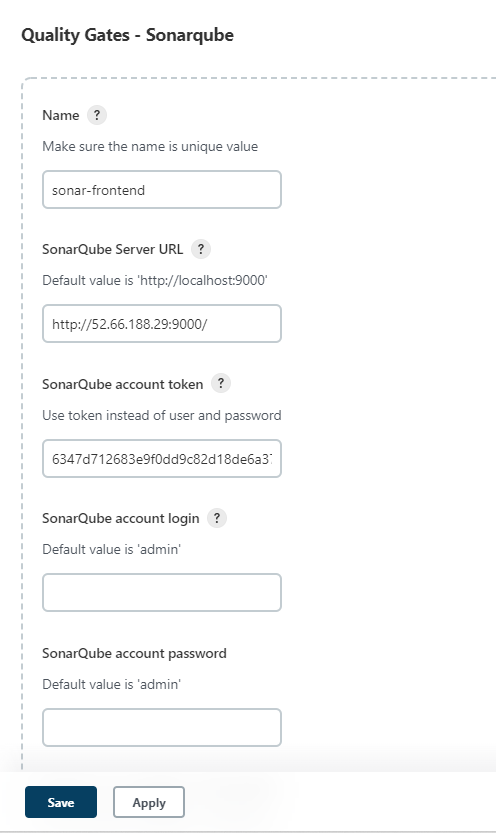
Select add credentials to add sonar token. If it doesn’t open, save without adding a token and later reopen the configuration page and check again.



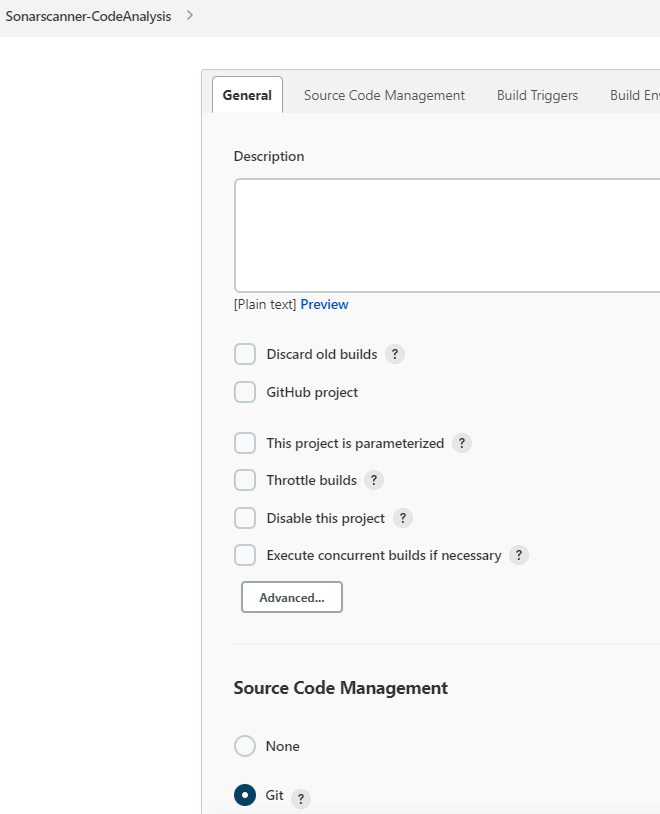
save the configuration after adding the sonar token at the sonar authentication token.

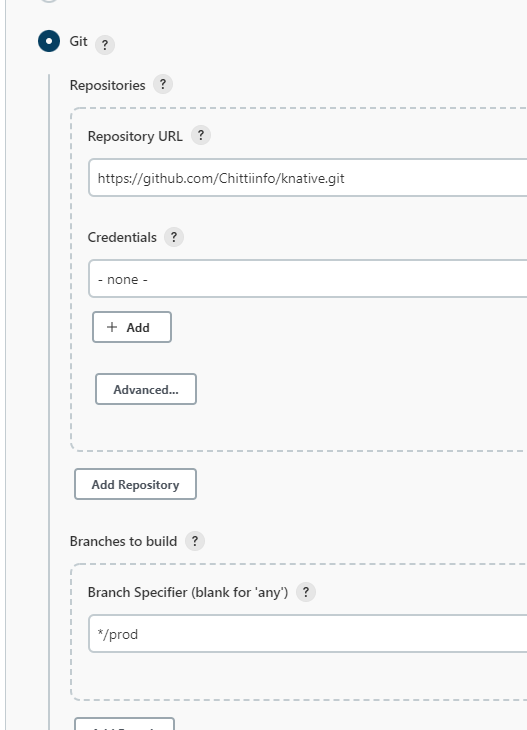
Quality gates give us the option to pass the build to next stages if only certain criteria are fulfilled. Sample Quality Gate configuration is given below.

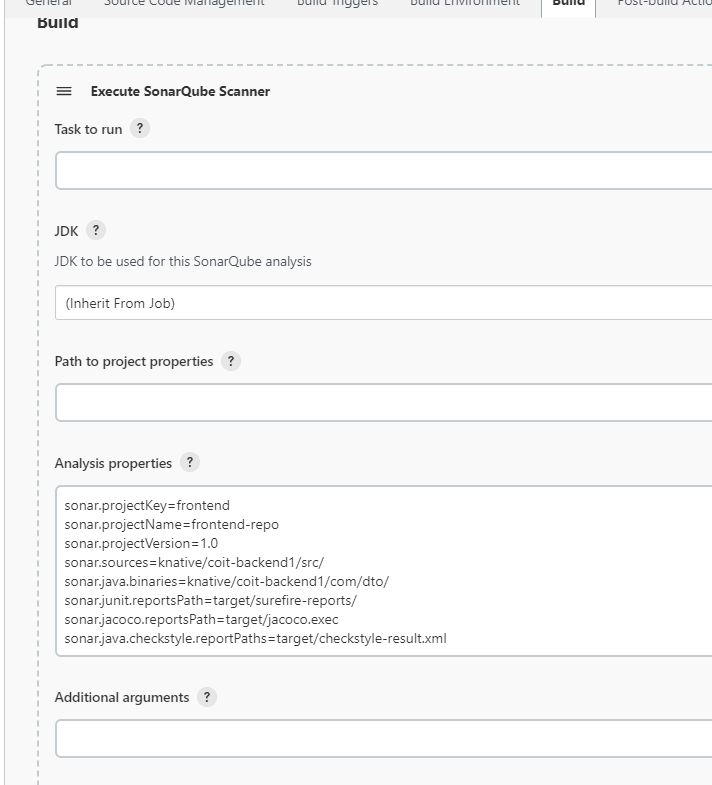




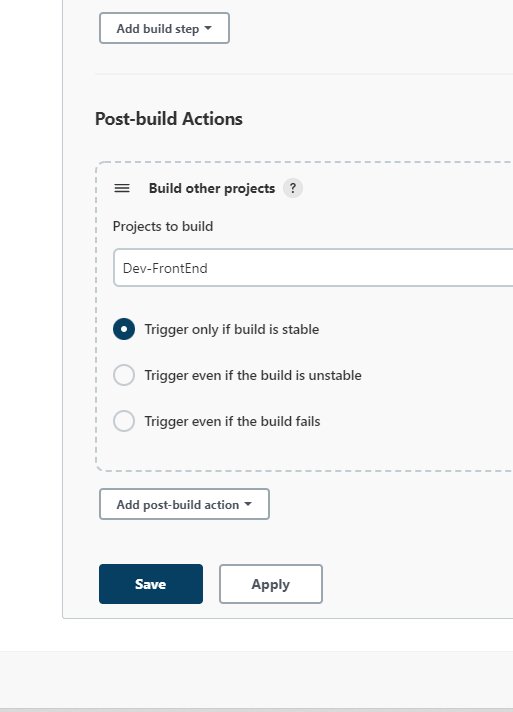
Jenkins job setup for Sonarqube static code analysis:

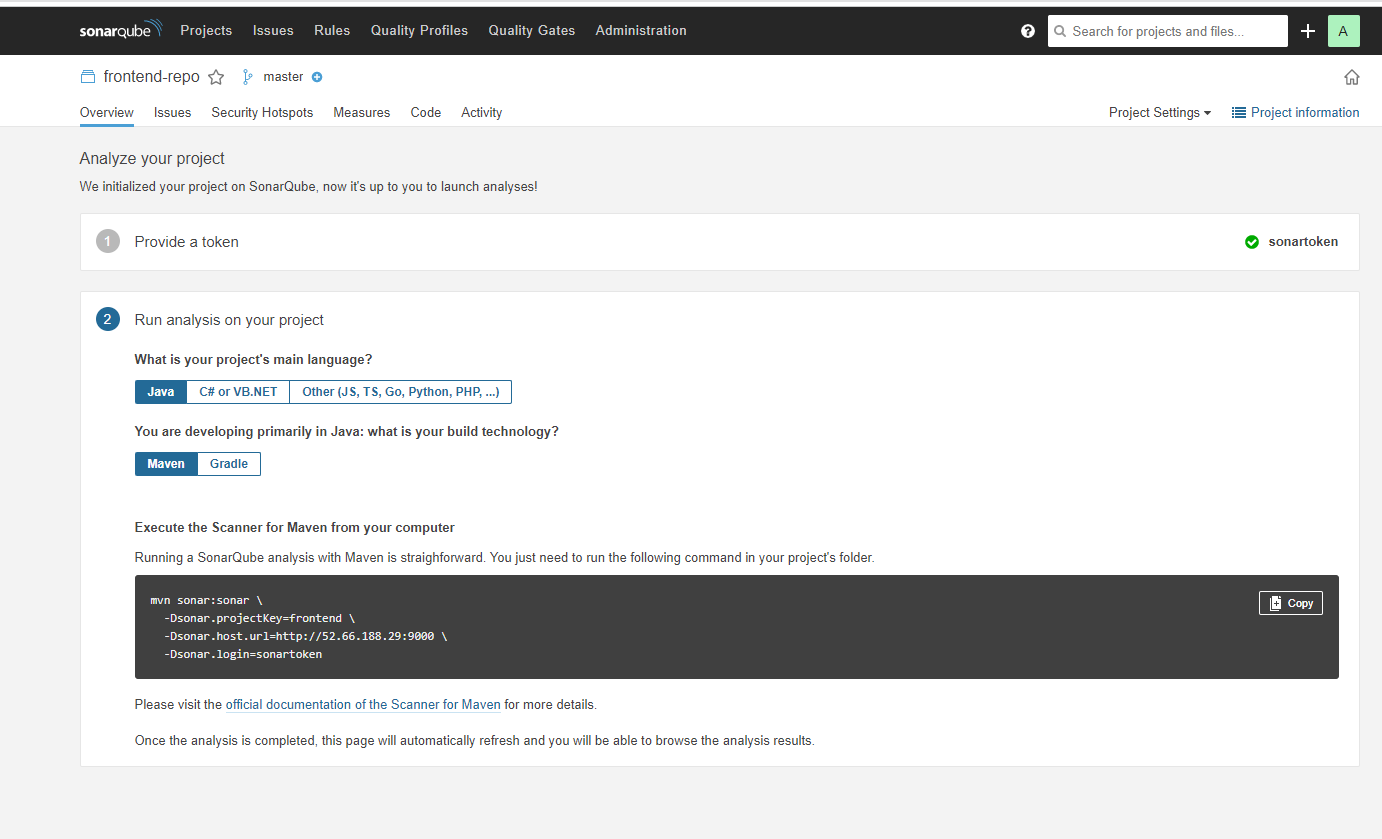




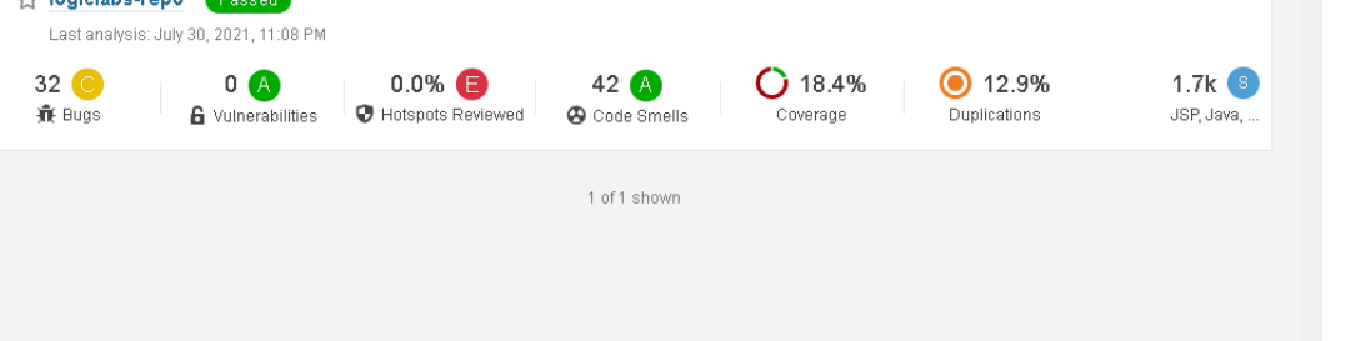




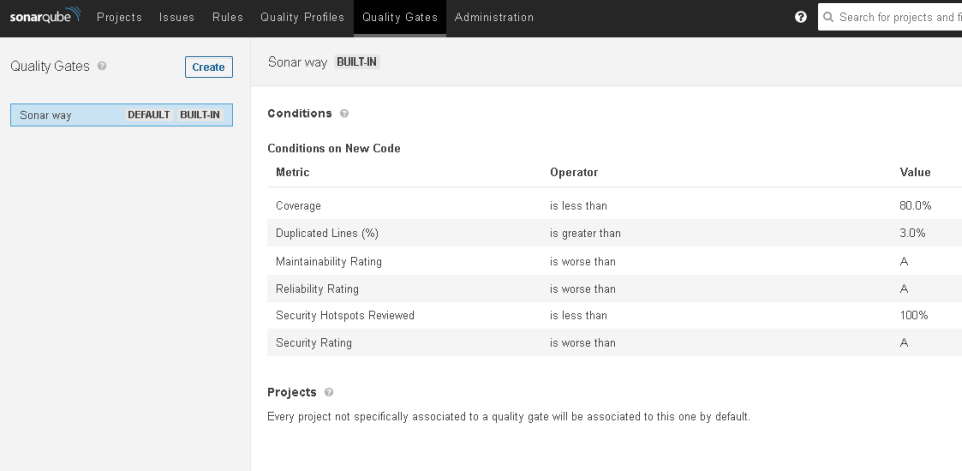


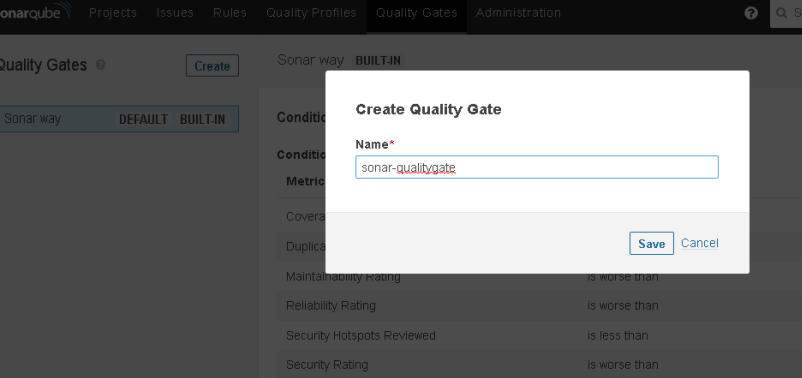


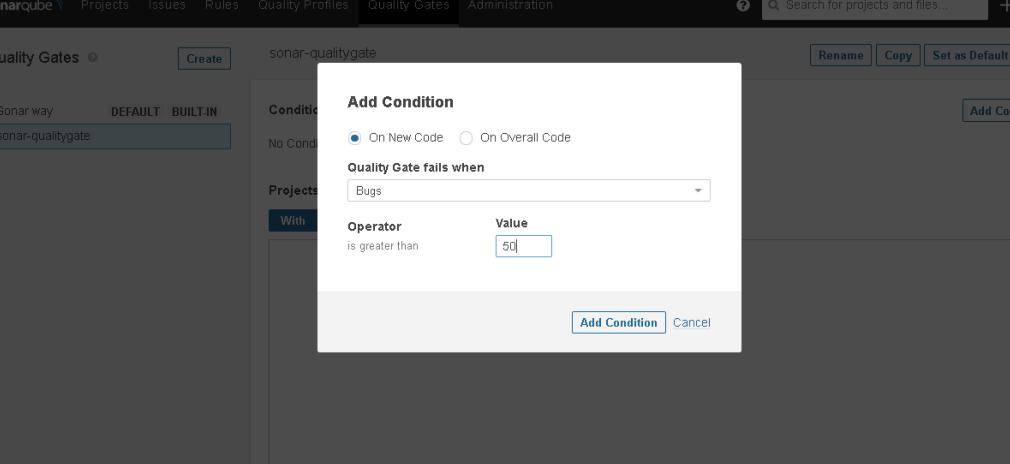
Verification of Static code Analysis from sonar server: After Jenkins job is complete, login to sonar server to verify and obtain the static code analysis results.

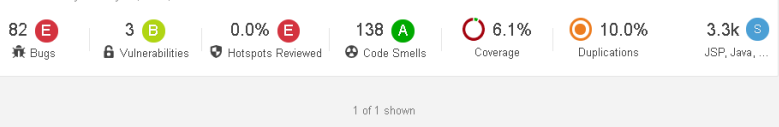


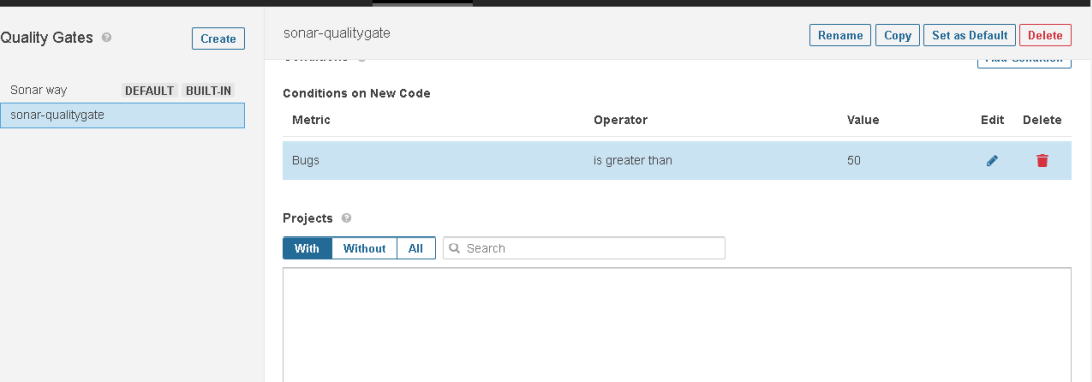
Creating Quality Gates: The below screenshot shows how to configure Quality gates

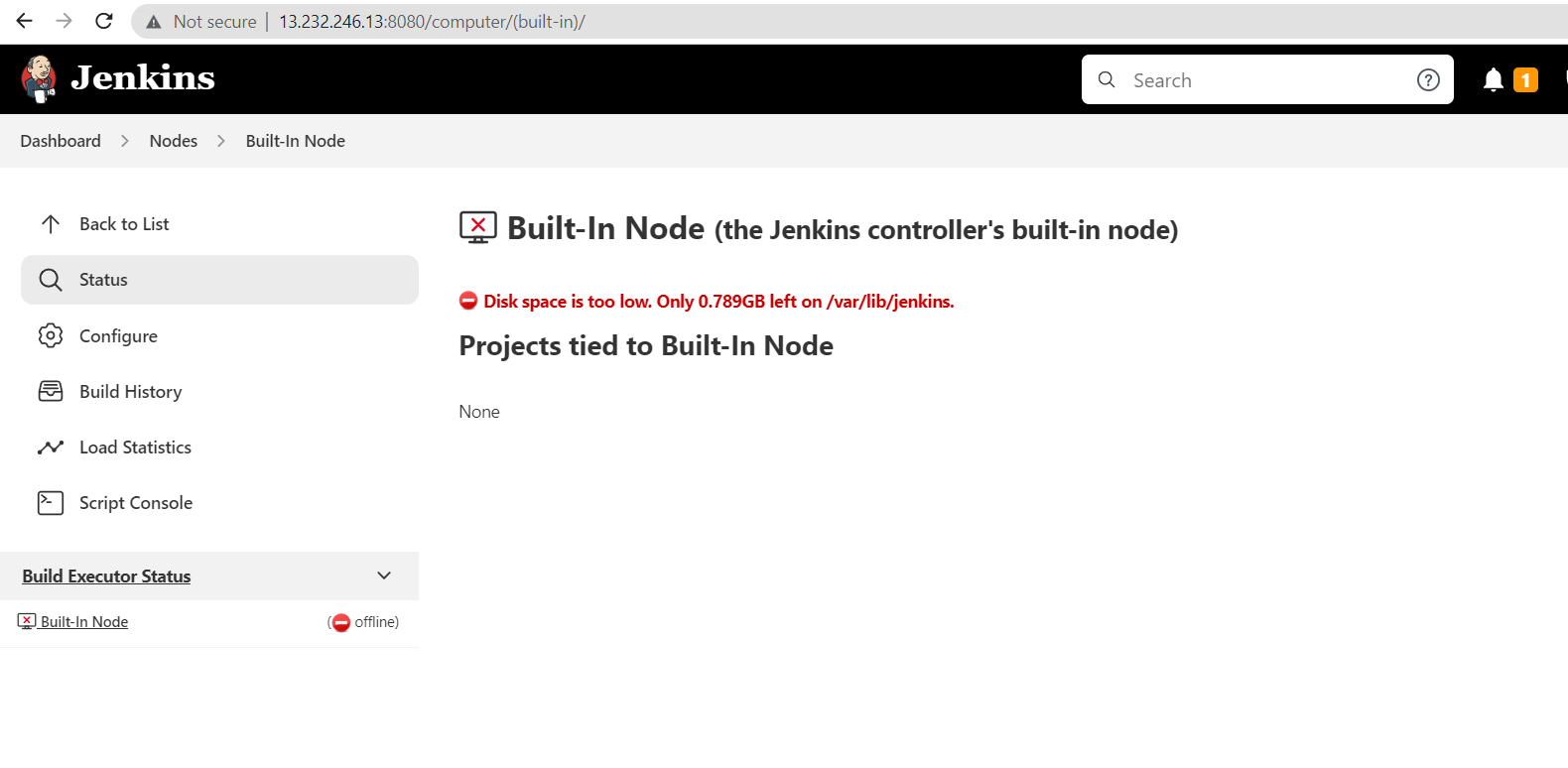












**Pipeline Jobs setup:**

**5 stages:**

**continous download**

**c build**

**c Depl**

**c Test**

**c Delivery**

**Scripted pipeline syntax:**

**------------------------------------**

**node ( 'master/slave')**

**{**

**stage(' Stage in CI-CD')**

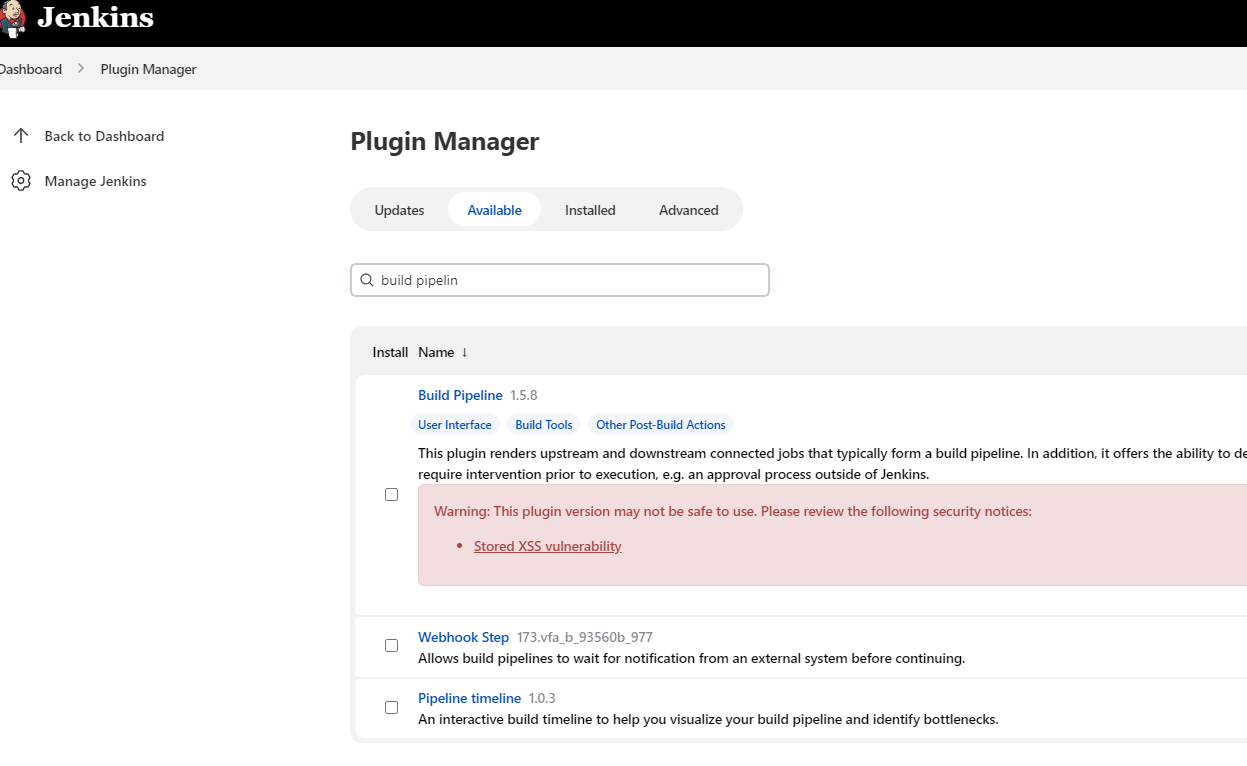
**{**

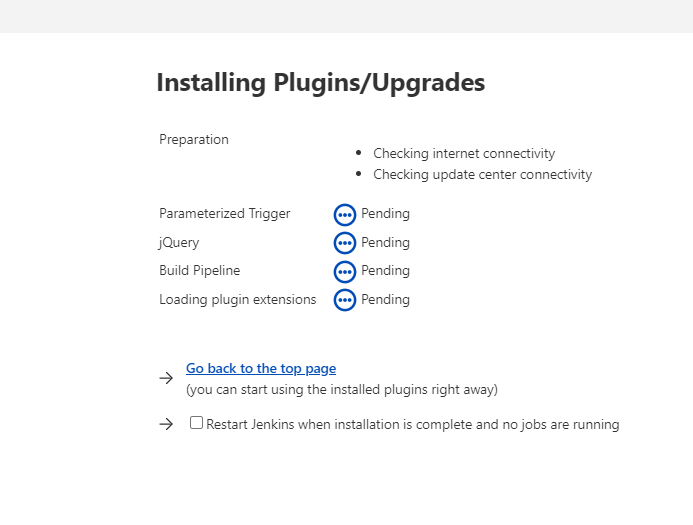
**Groovy code for implementing the stage**

**}**

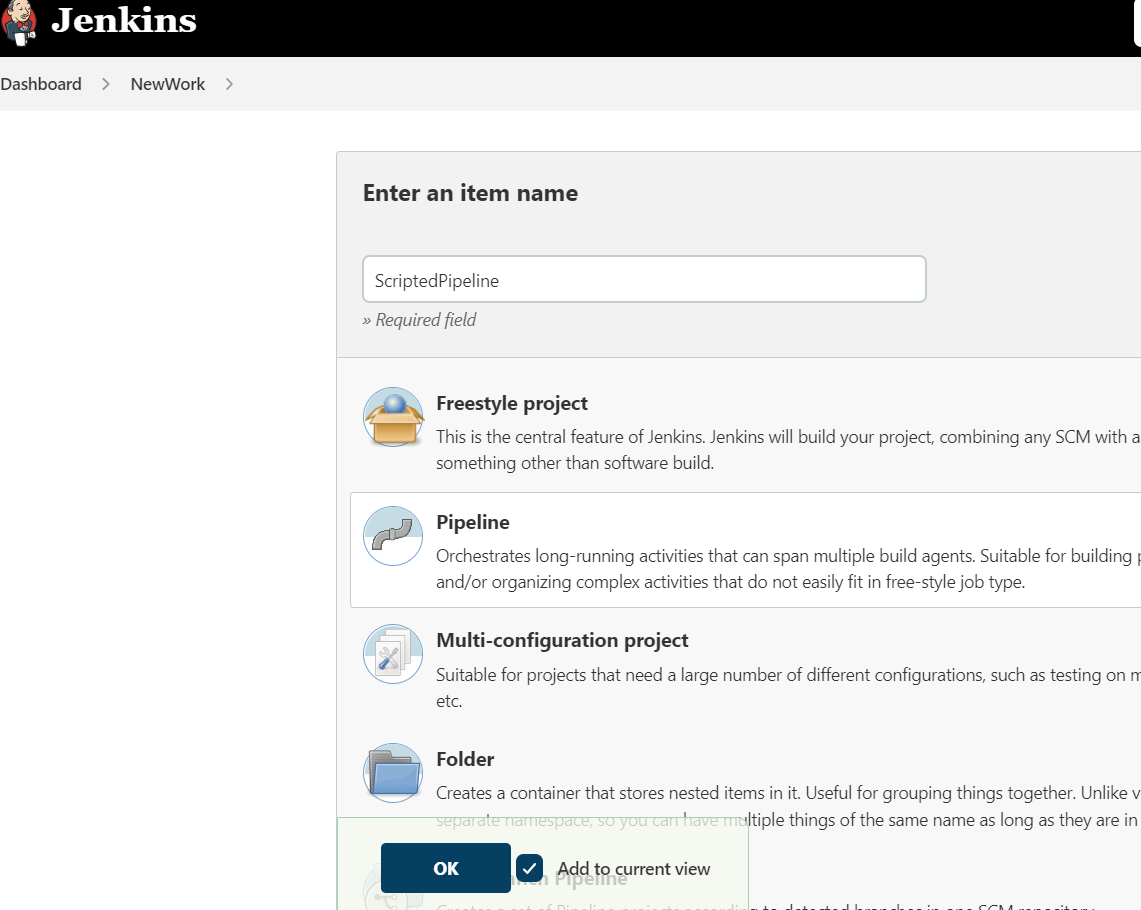
**}**

**Install Build pipeline plugin**

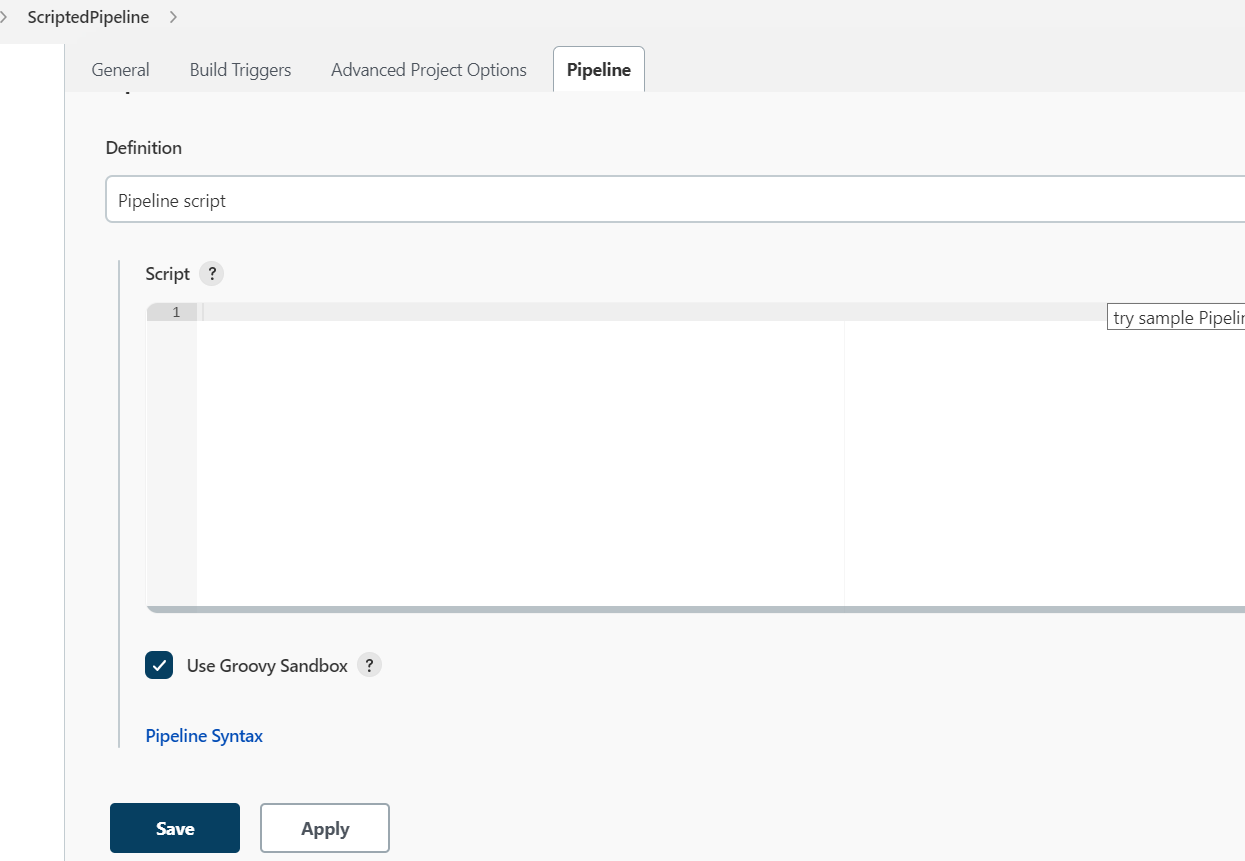
****

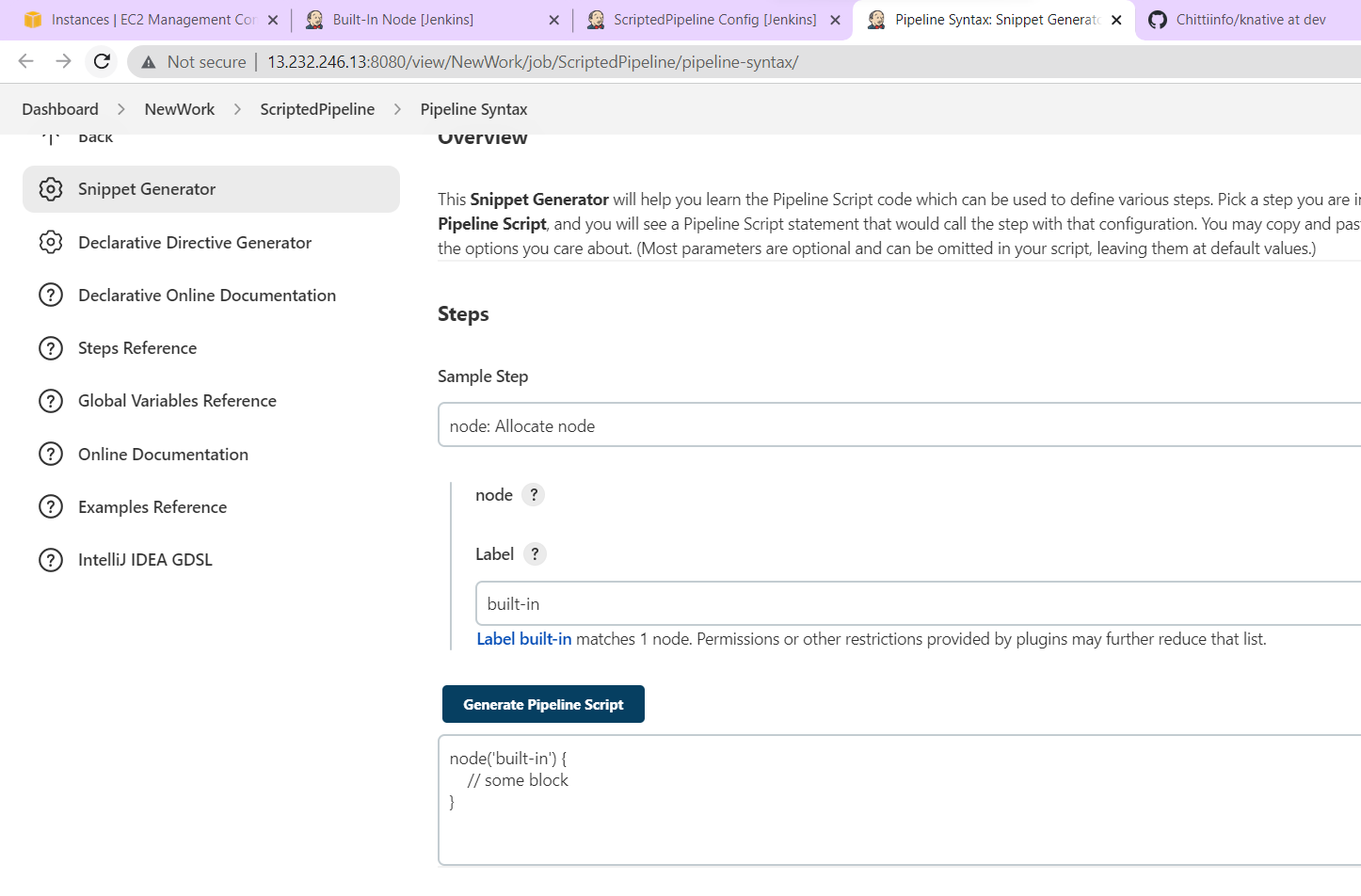
****

**Create new job as “Pipeline” job**

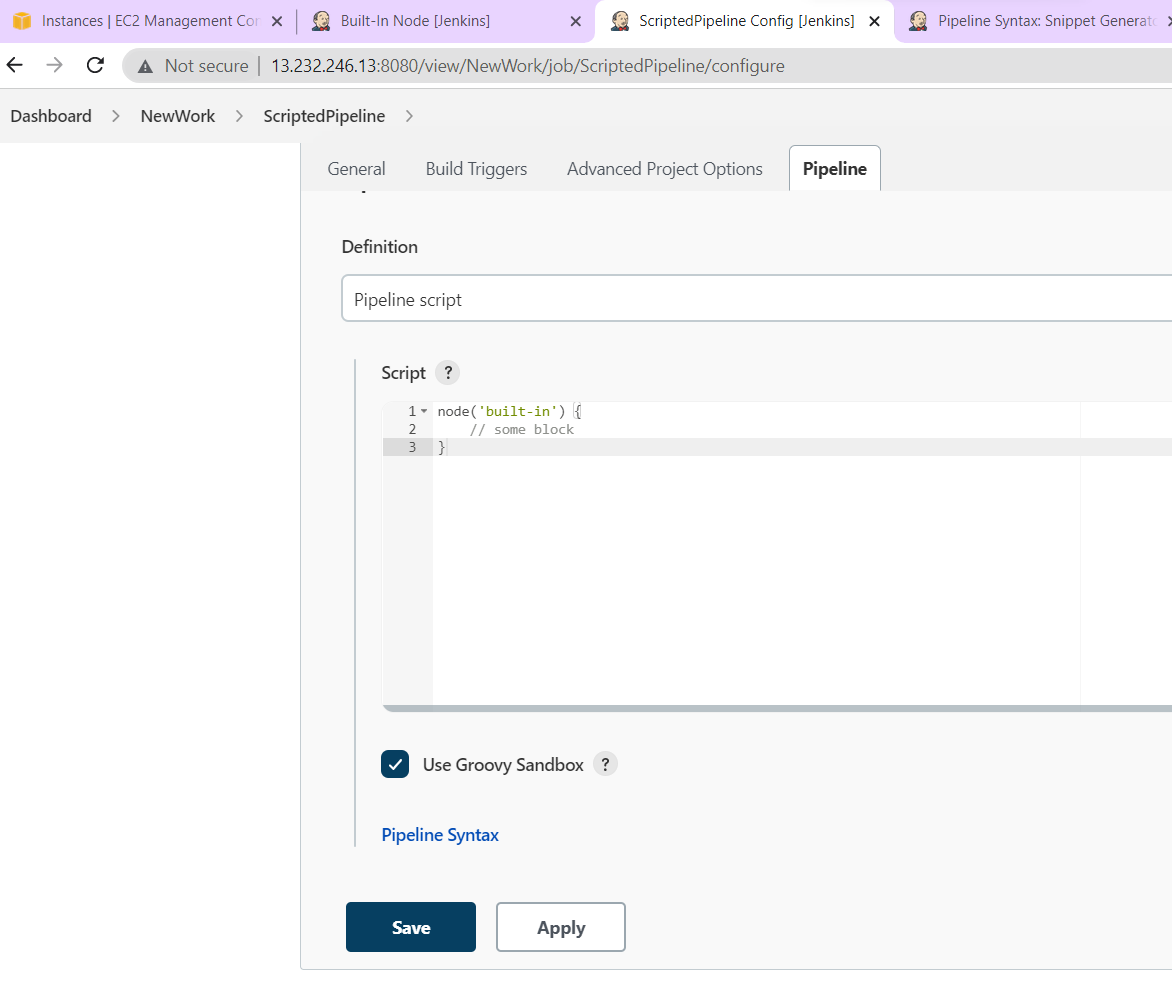
****

**Click on Pipeline syntax to generate groovy scritp**

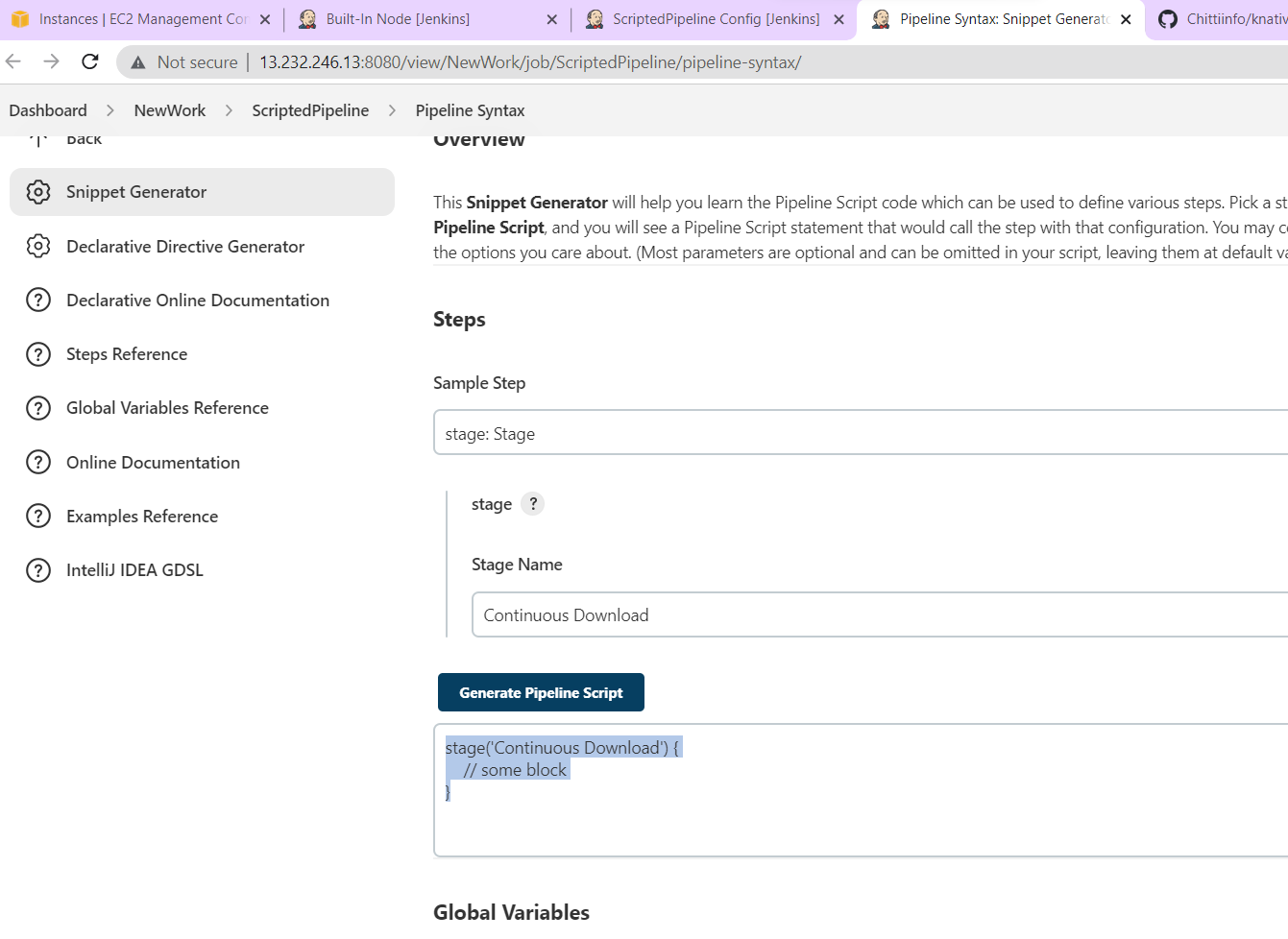
****

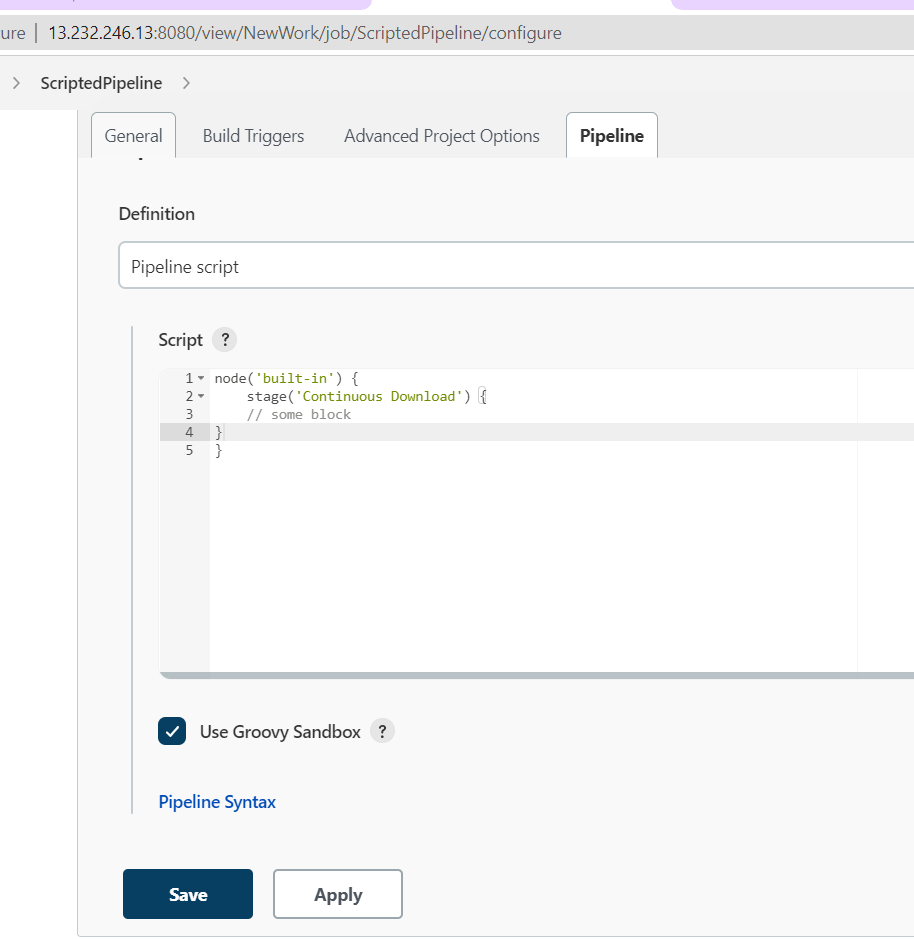
****

**Copy the generated code back on orginal job.**

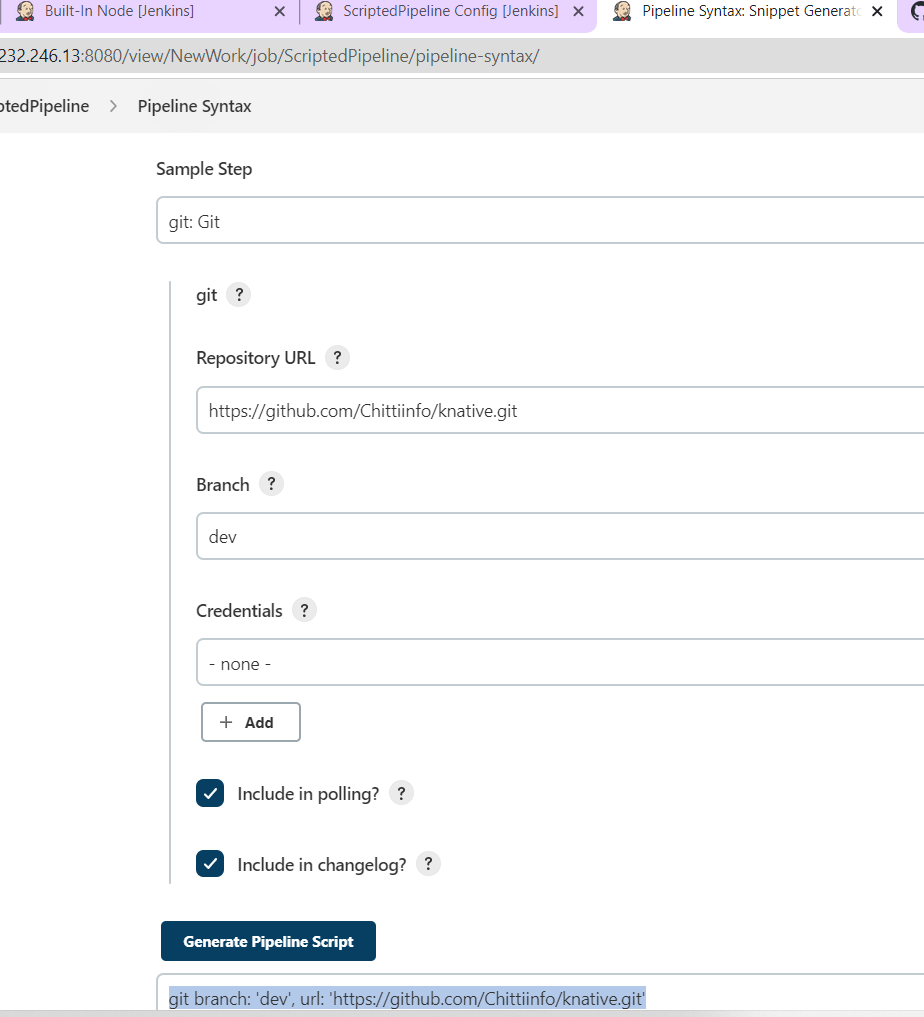
****

**Now generate code for particular stage**

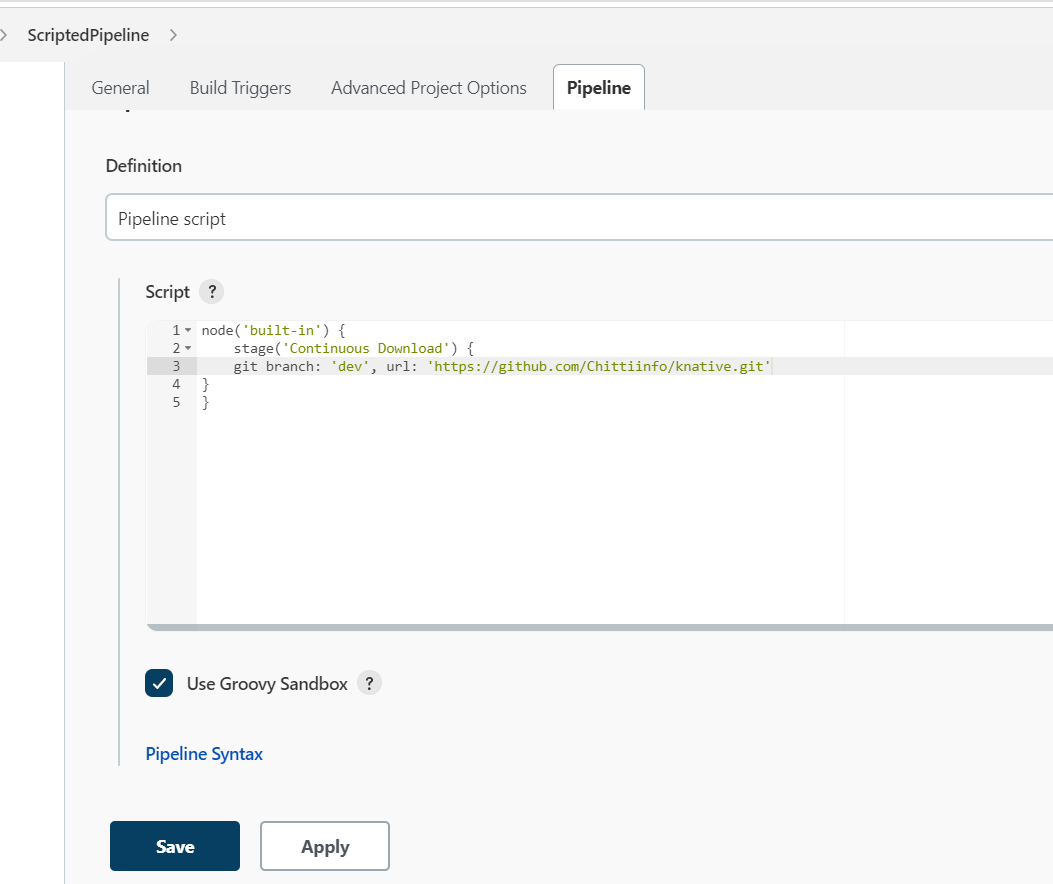
****

****

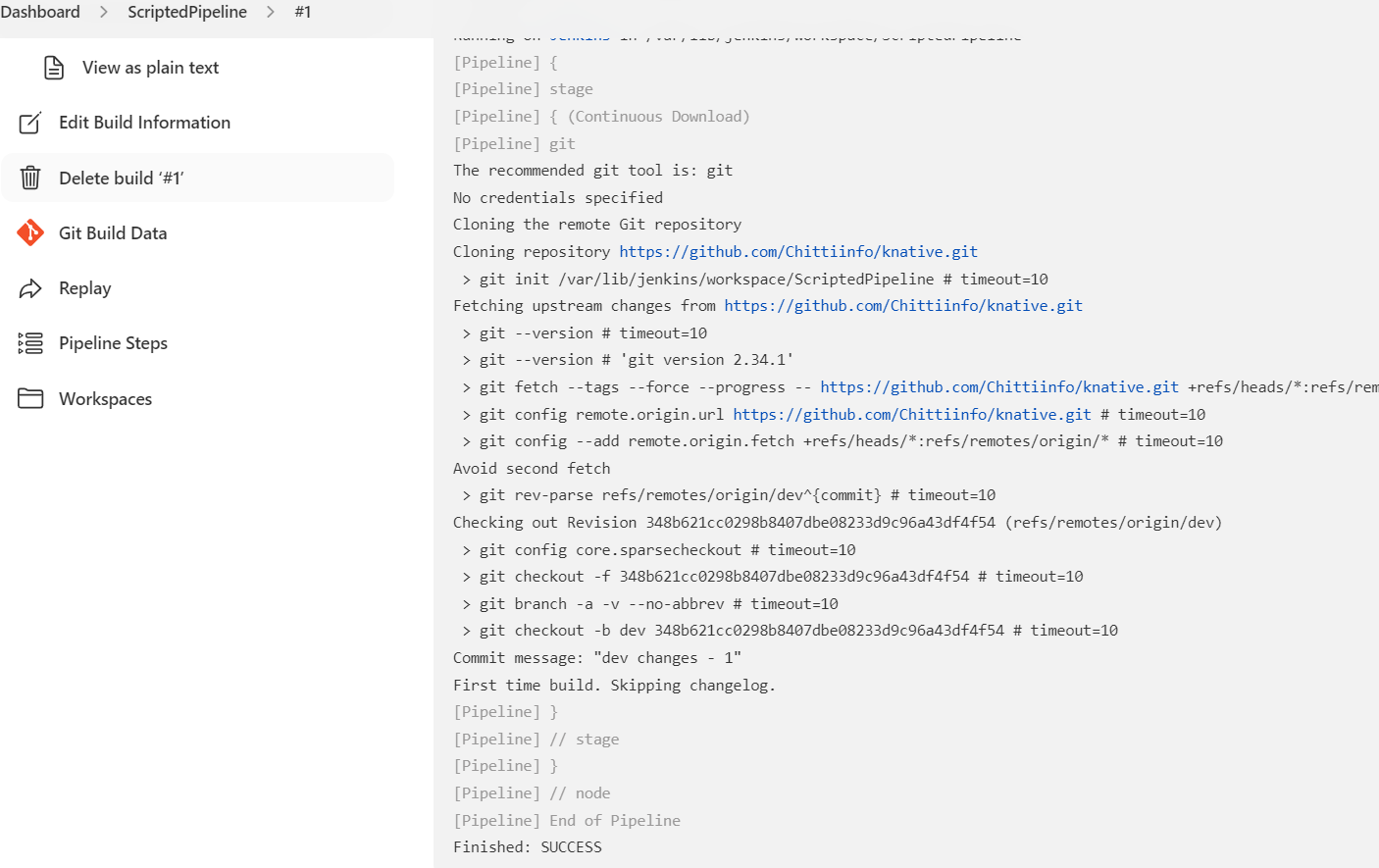
**Now generate code for downloading code from github**

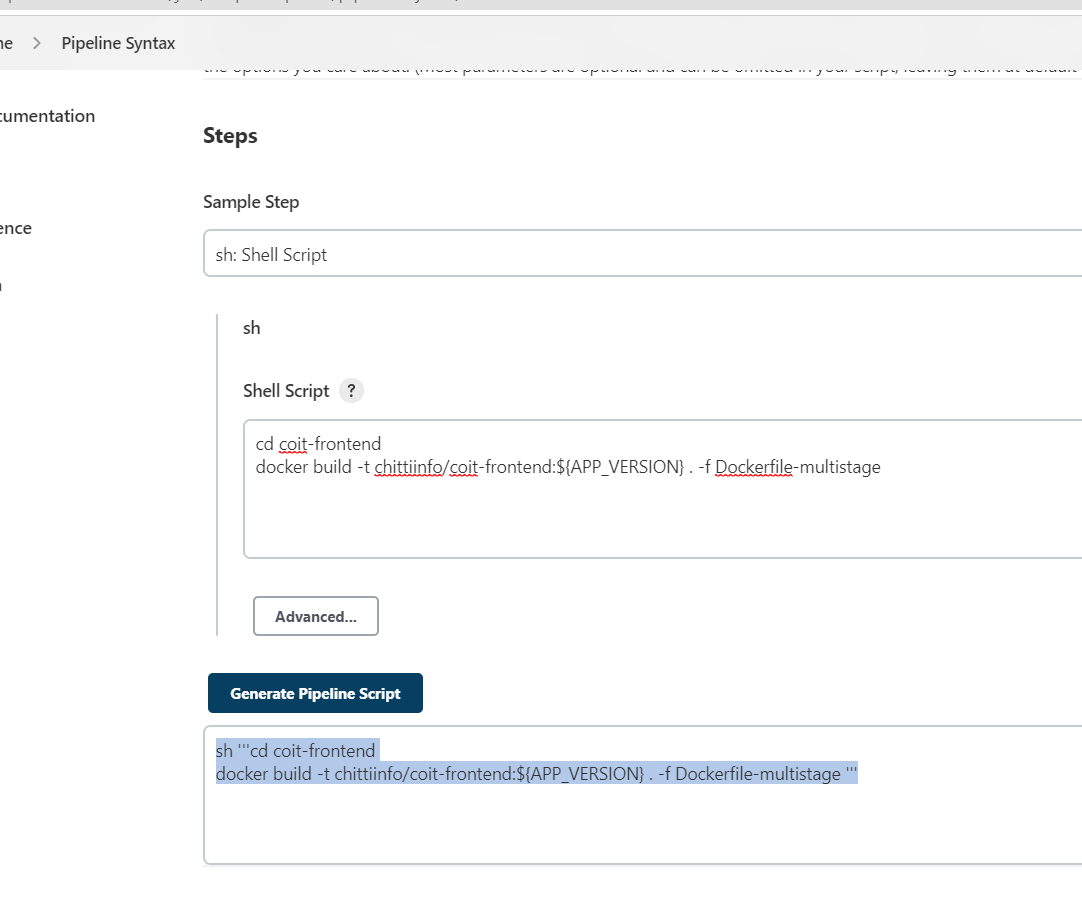
****

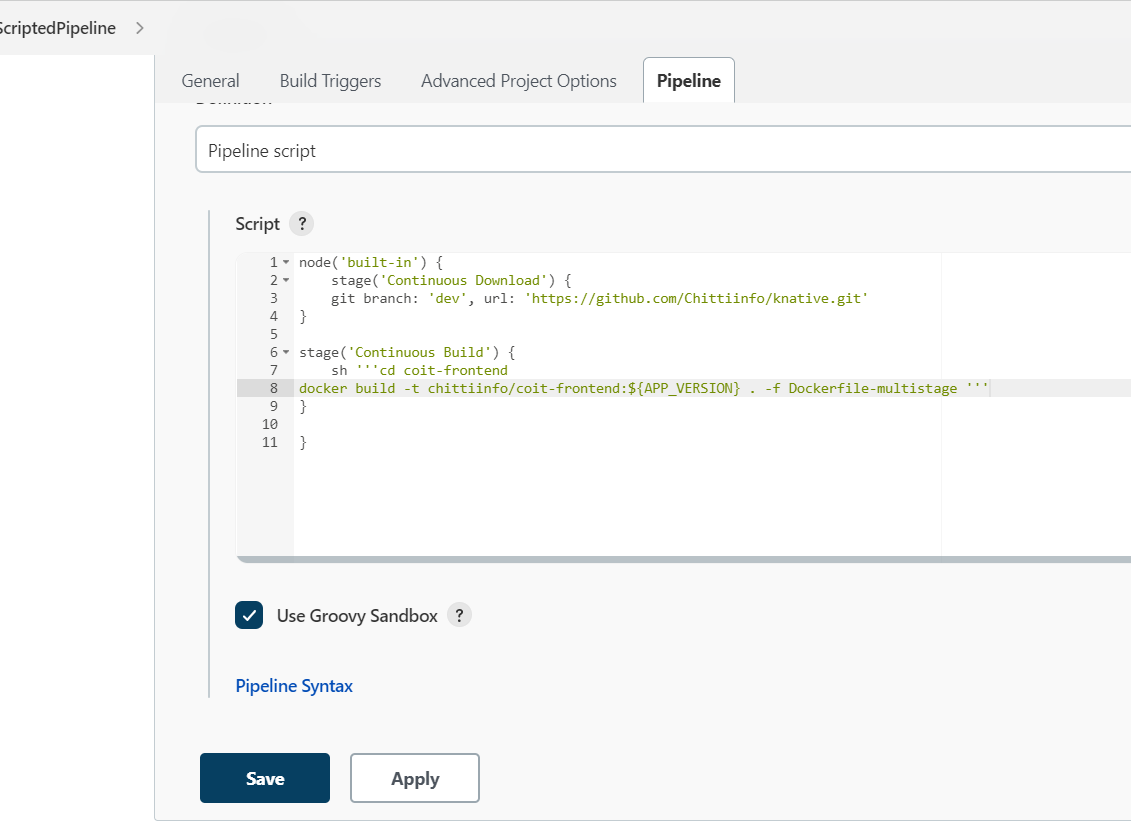
**Copy this code back on pipeline job.**

****

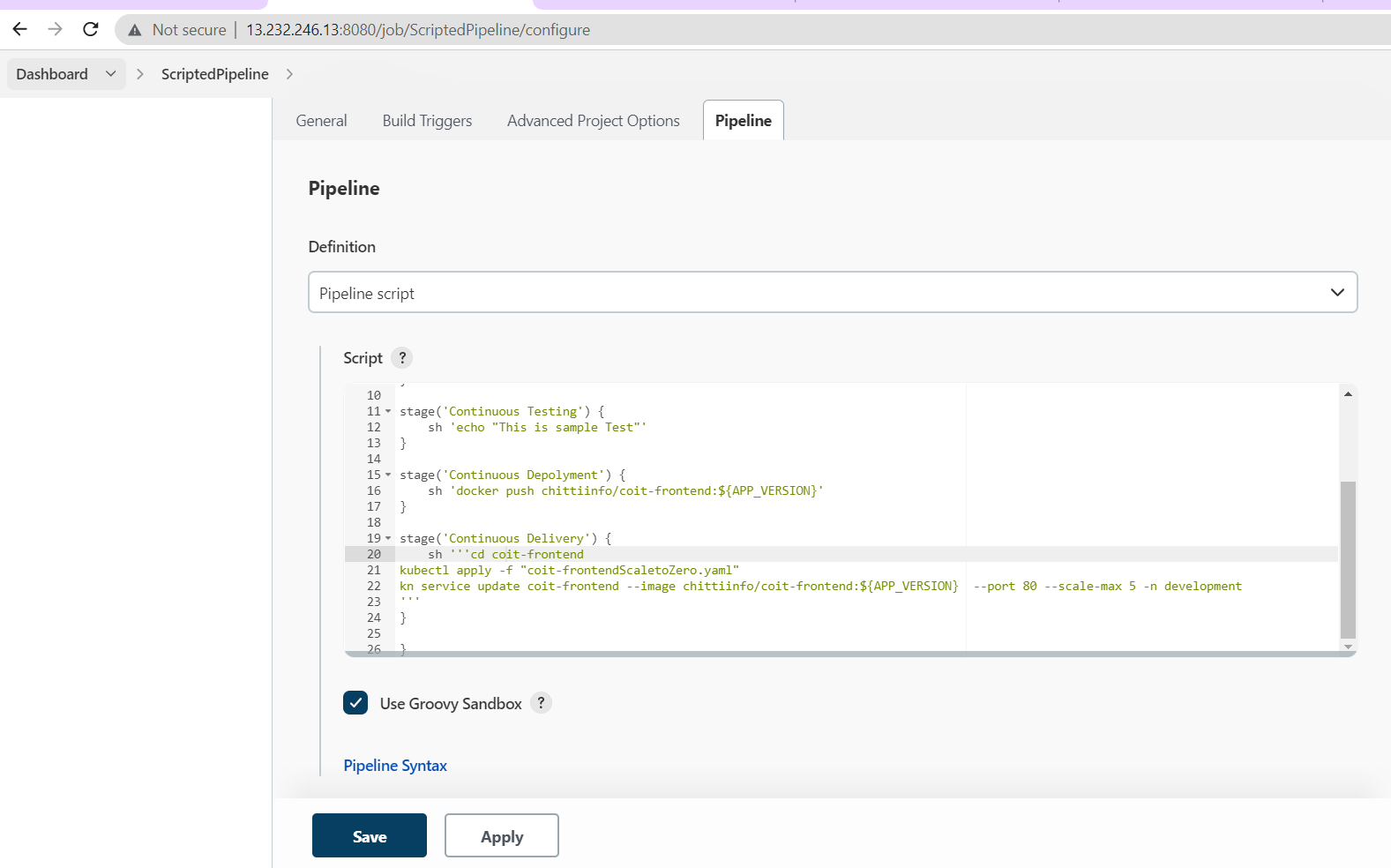
**Now run the build.**

****

****

****

****

****

**Complete pipeline groovy code for Dev:**

**node('built-in') {**

**stage('Continuous Download') {**

**git branch: 'dev', url: 'https://github.com/Chittiinfo/knative.git'**

**}**

**stage('Continuous Build') {**

**sh '''cd coit-frontend**

**docker build -t chittiinfo/coit-frontend:${APP\_VERSION} . -f Dockerfile-multistage '''**

**}**

**stage('Continuous Testing') {**

**sh 'echo "This is sample Test"'**

**}**

**stage('Continuous Depolyment') {**

**sh 'docker push chittiinfo/coit-frontend:${APP\_VERSION}'**

**}**

**stage('Continuous Delivery') {**

**sh '''cd coit-frontend**

**kubectl apply -f "coit-frontendScaletoZero.yaml"**

**kn service update coit-frontend --image chittiinfo/coit-frontend:${APP\_VERSION} --port 80 --scale-max 5 -n development**

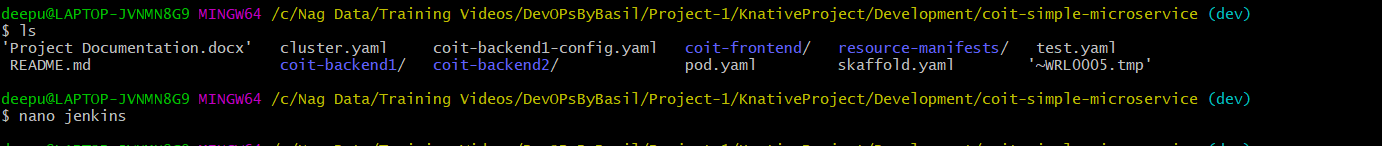
**'''**

**}**

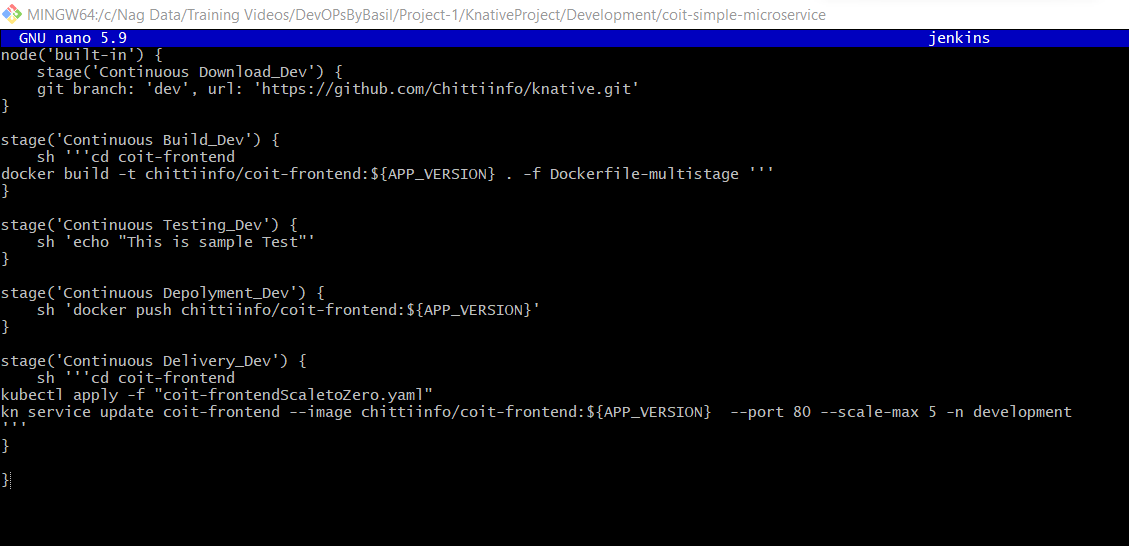
**}**

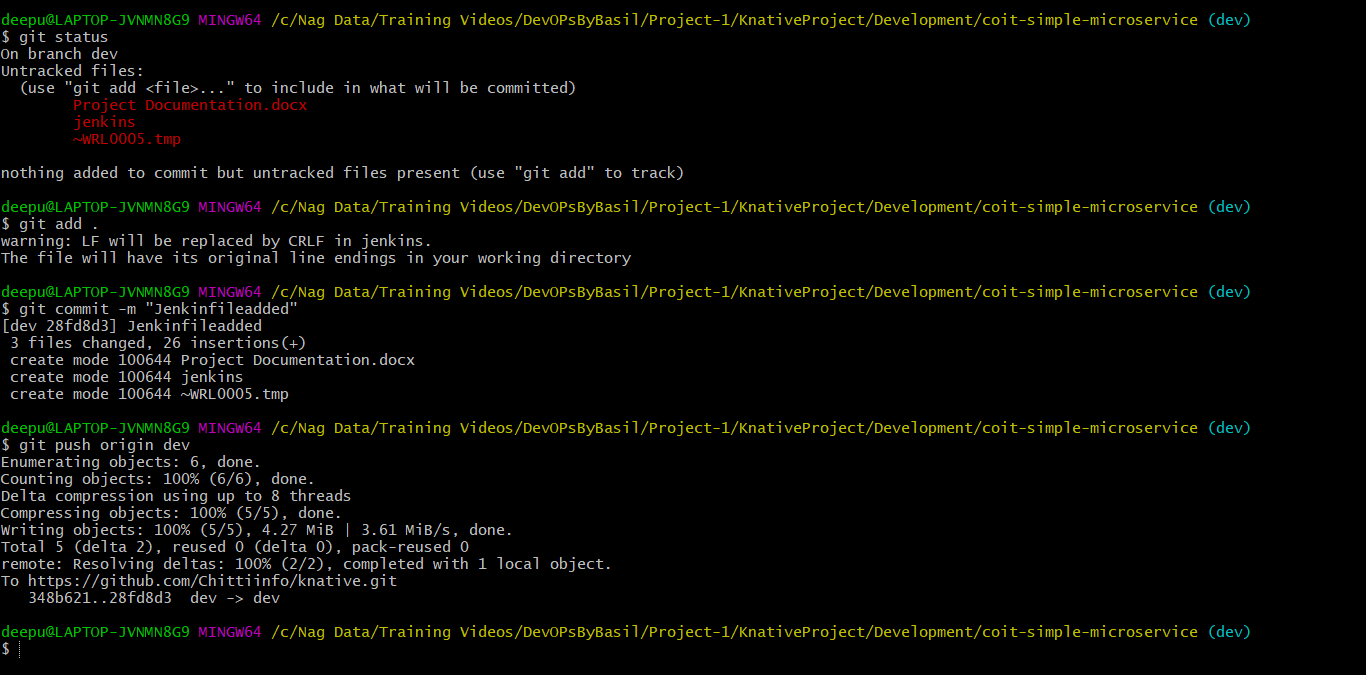
**Multibranch Pipeline:**

****

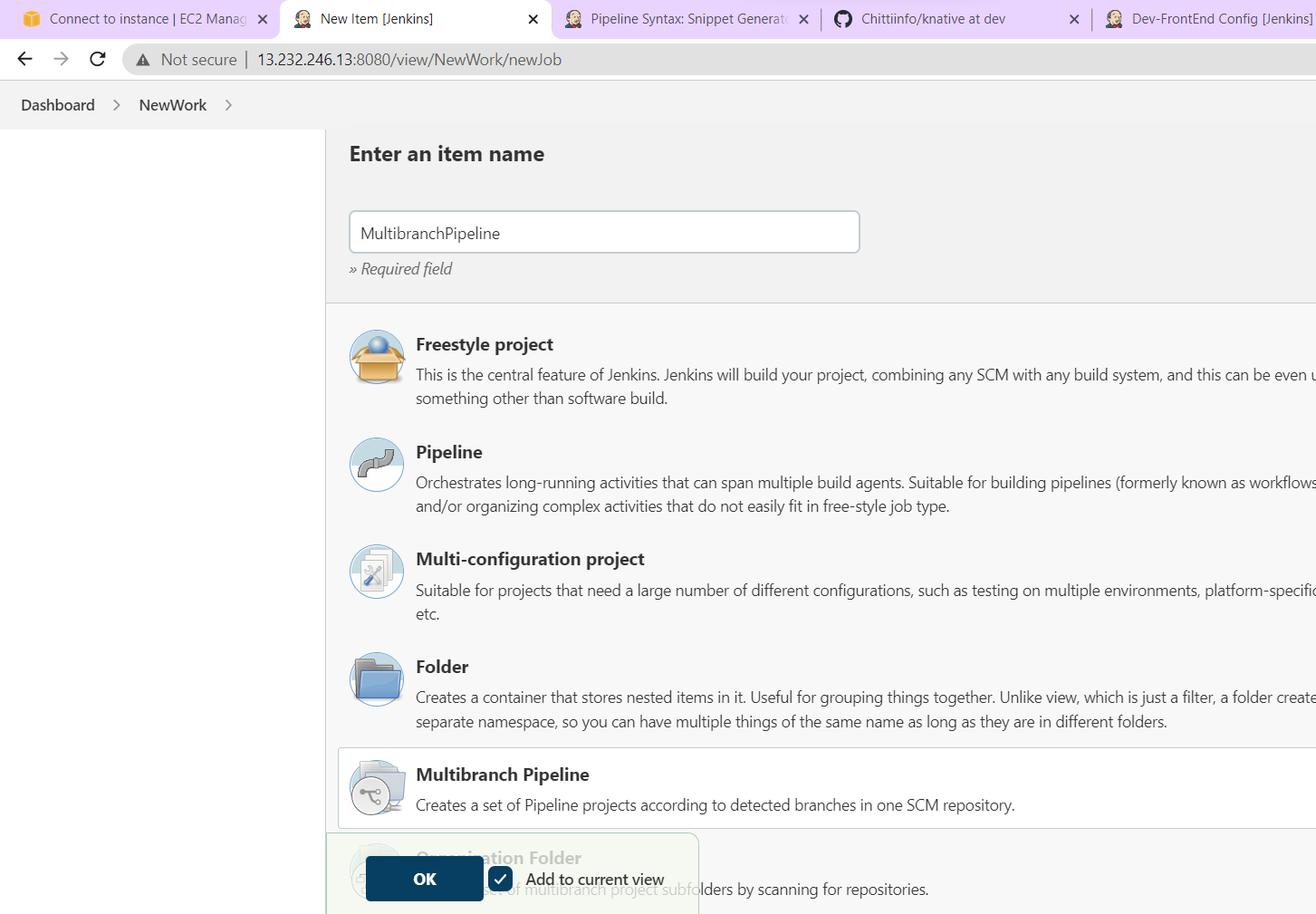
****

**Ensure below groovy code is copied in Jenkins file then commit changes to local then remote repository.**

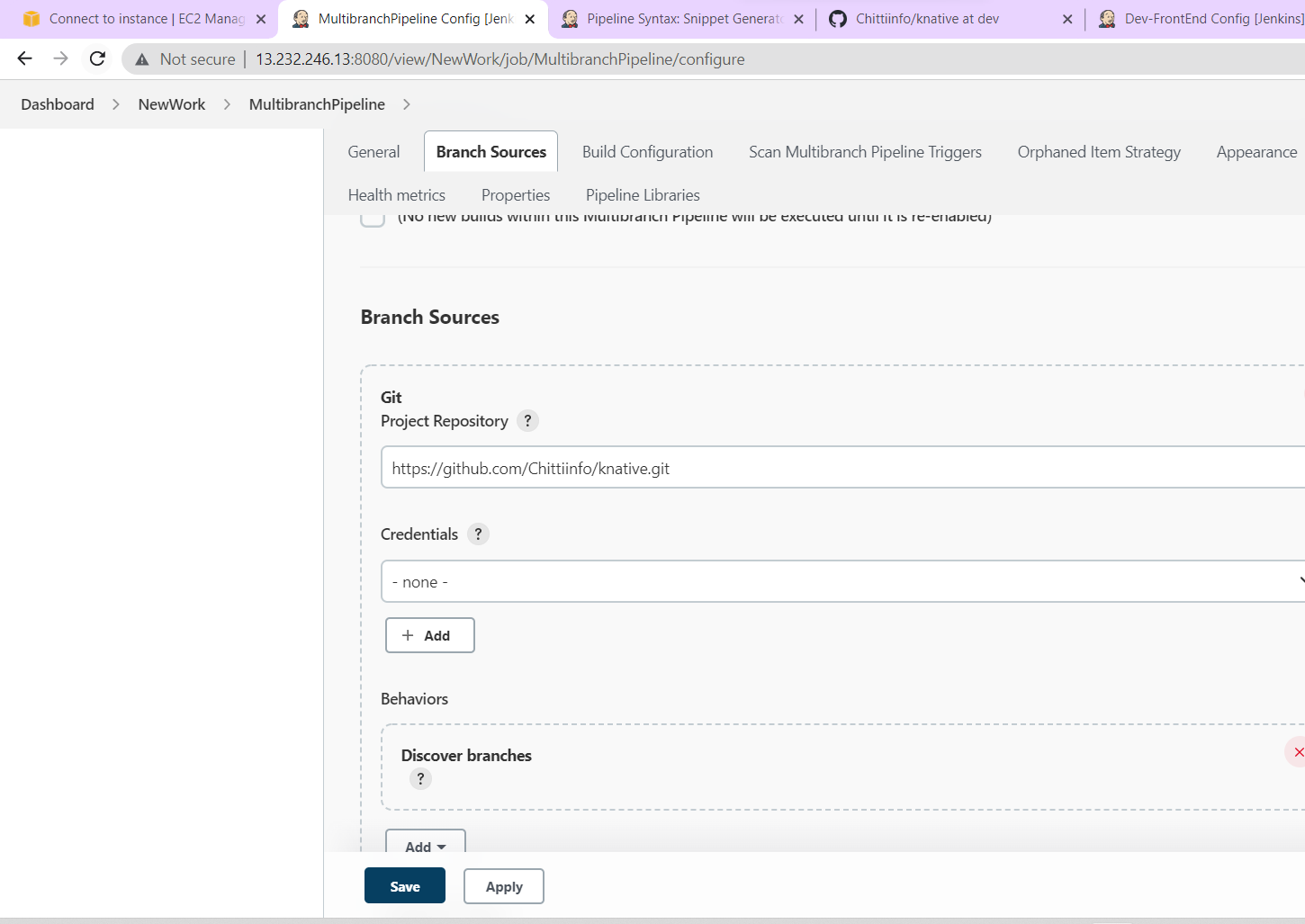
****

****

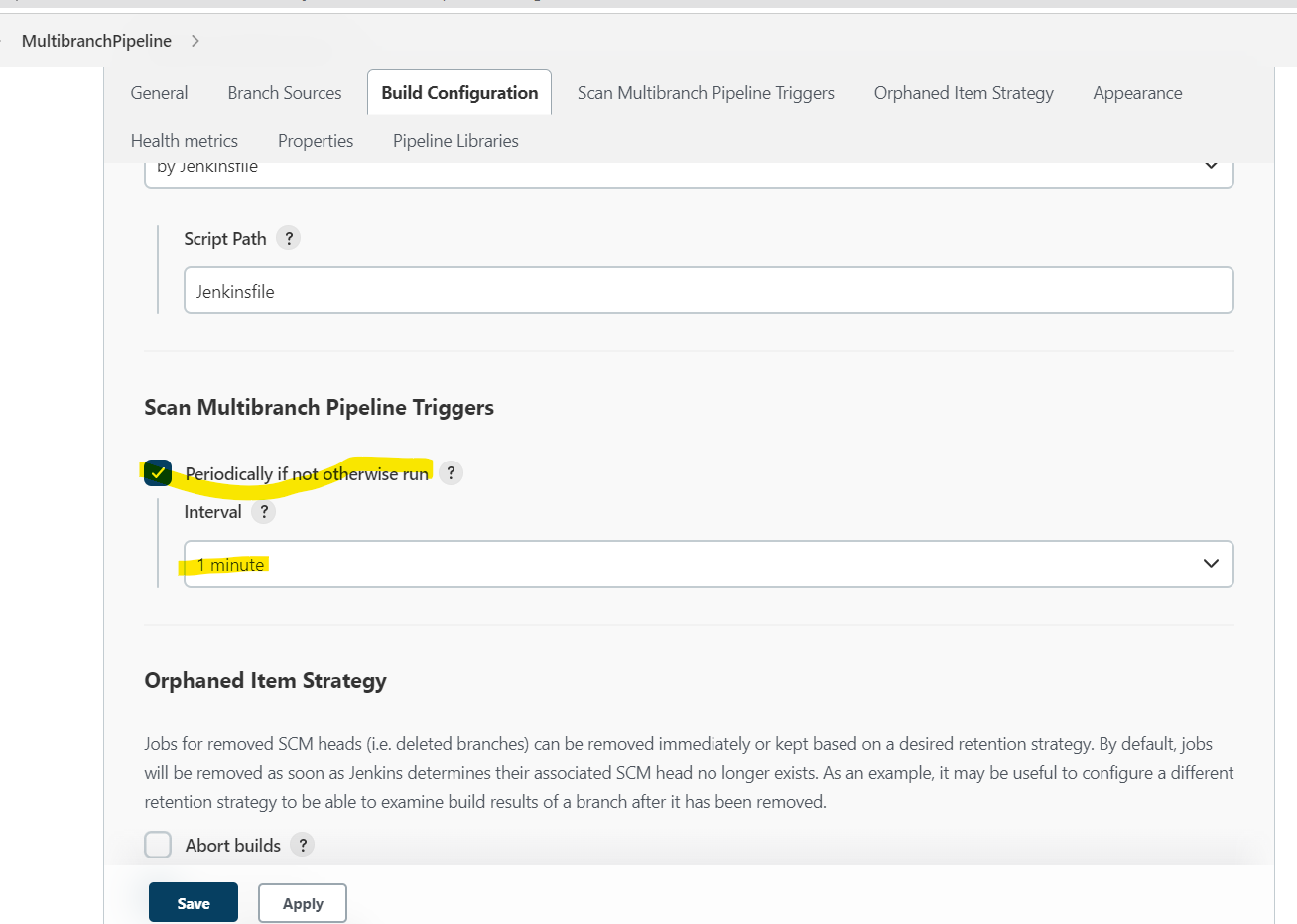
**Once Jenkins file is uploaded into repository, create new Jenkins job with multi branch pipeline option.**

****

**Add github url.**

****

**Select below highlighted option to scan code changes and trigger job automatically.**

****