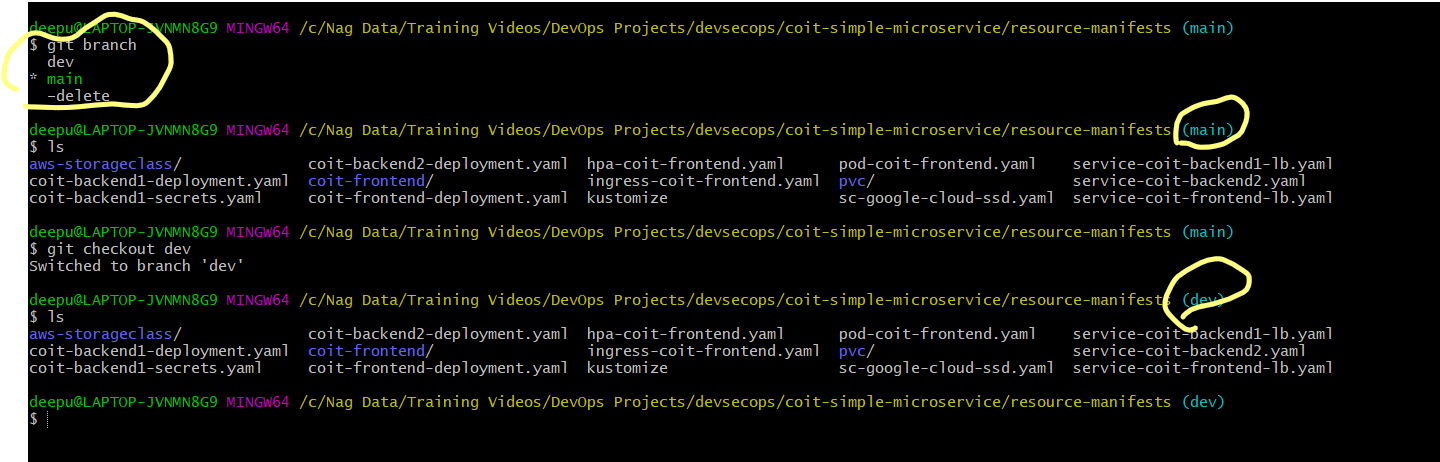
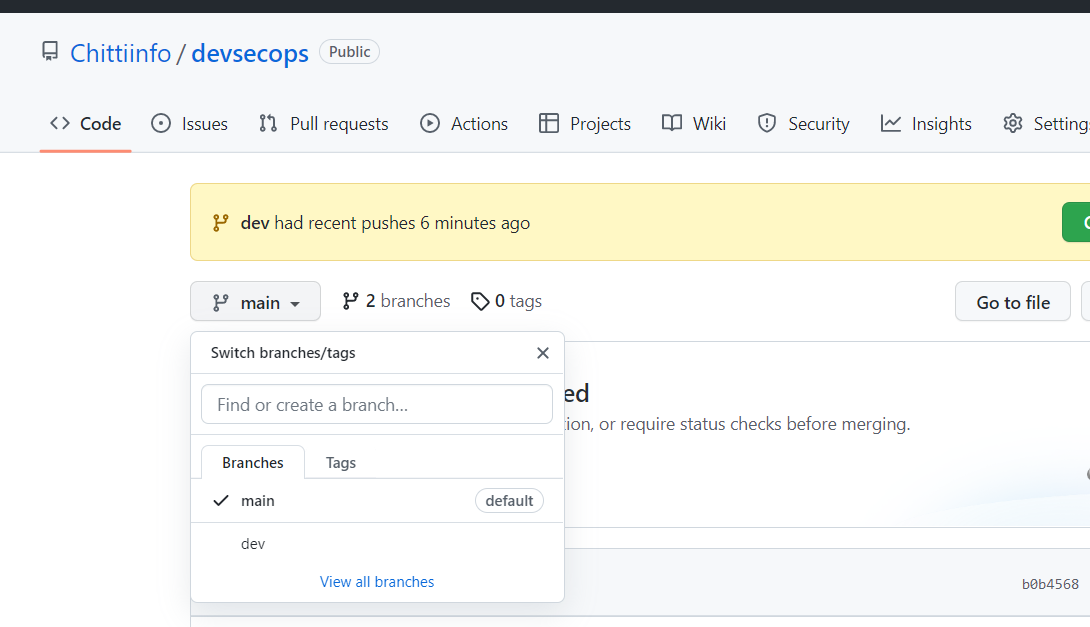
DEVSECOPS PROJECT

GIT Work:

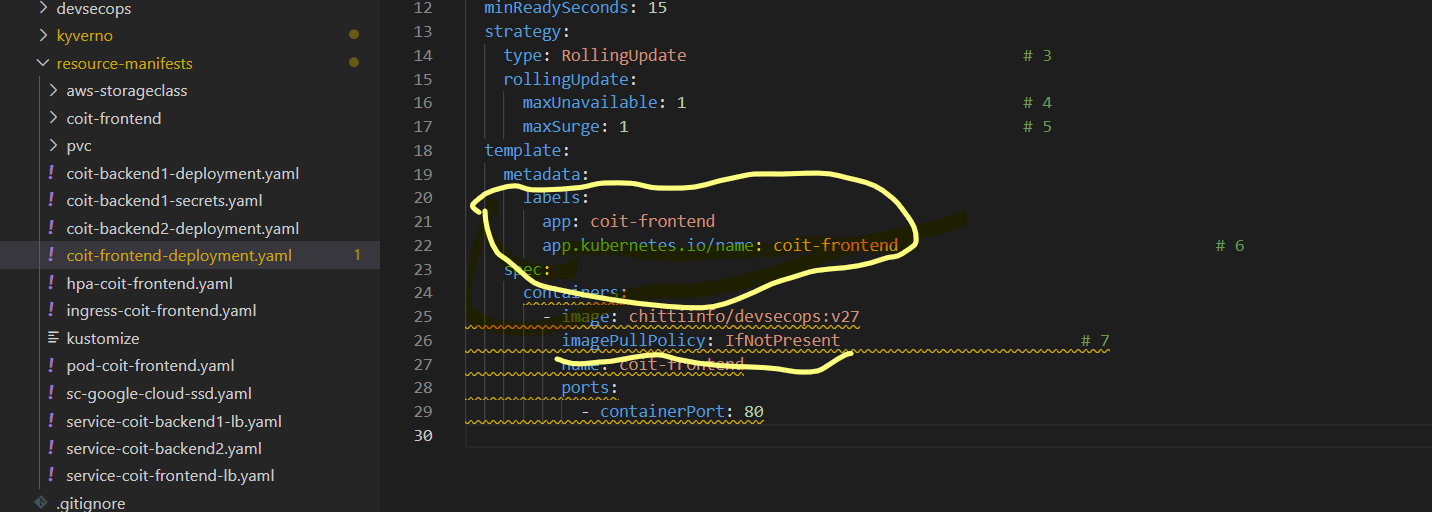
I have two branches (main & dev). I want to make all code level changes in Dev branch and once code is tested and stable, i will merge dev with main branch with pull request.

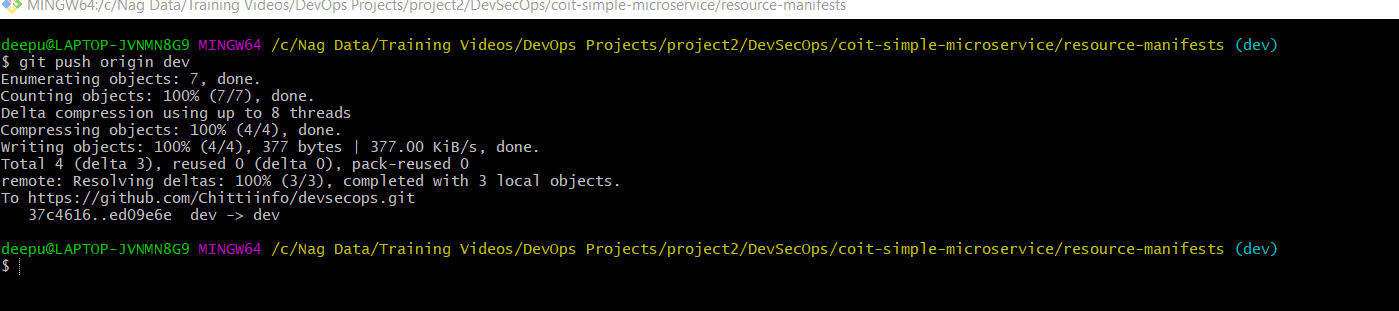
https://github.com/Chittiinfo/devsecops.git



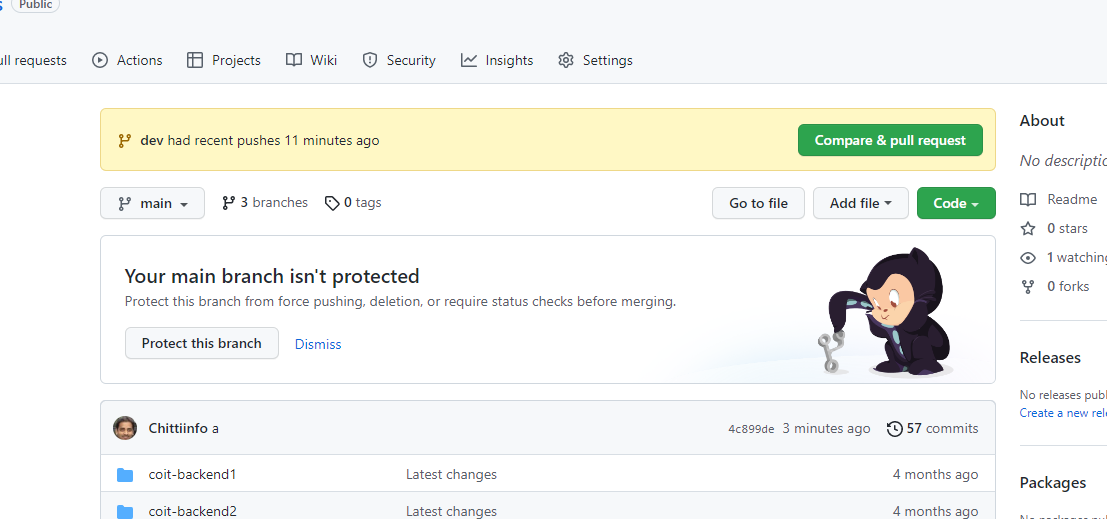


I have just ensured my deb branch deployment yml file have below kyverno label and proper image name then created to pull request to merge this code with main branch code

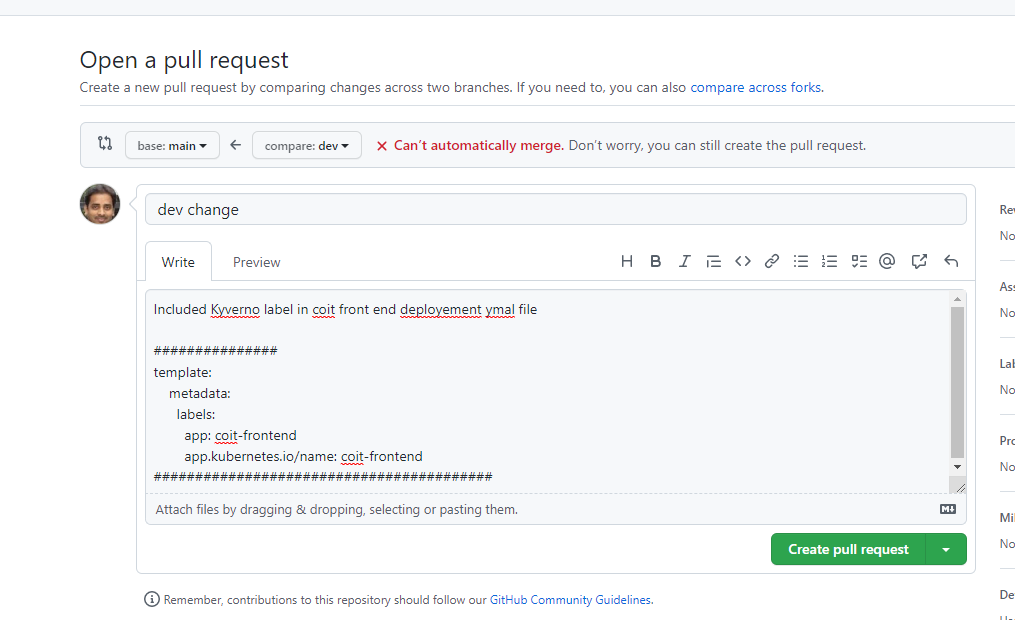




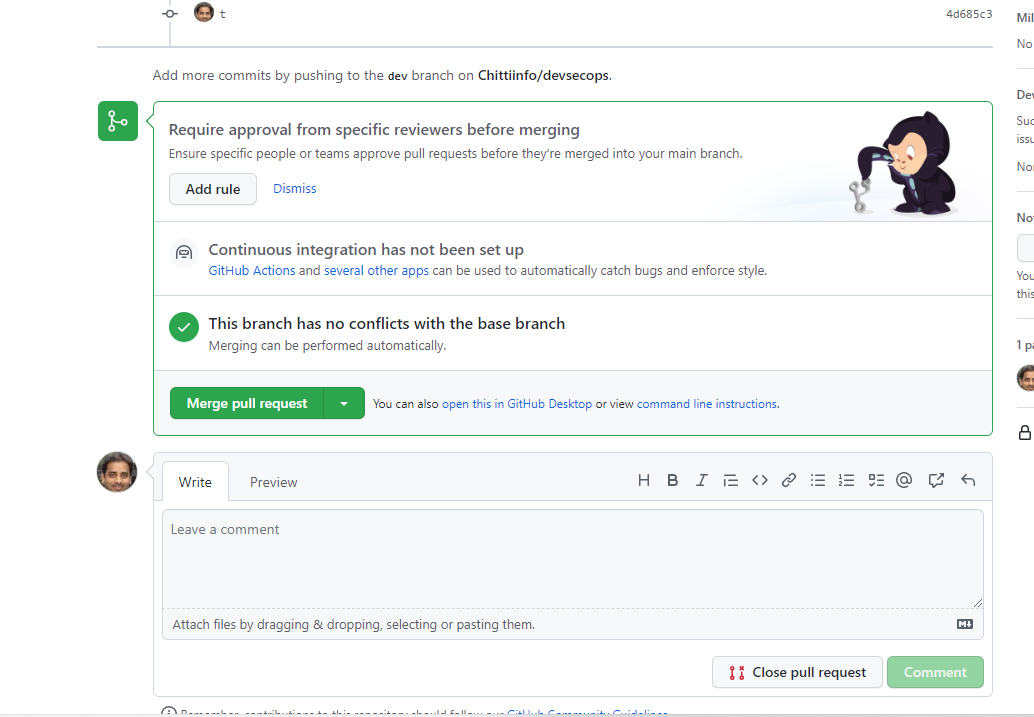
Once you push the changes to your repo, the Compare & pull request button will appear in GitHub.

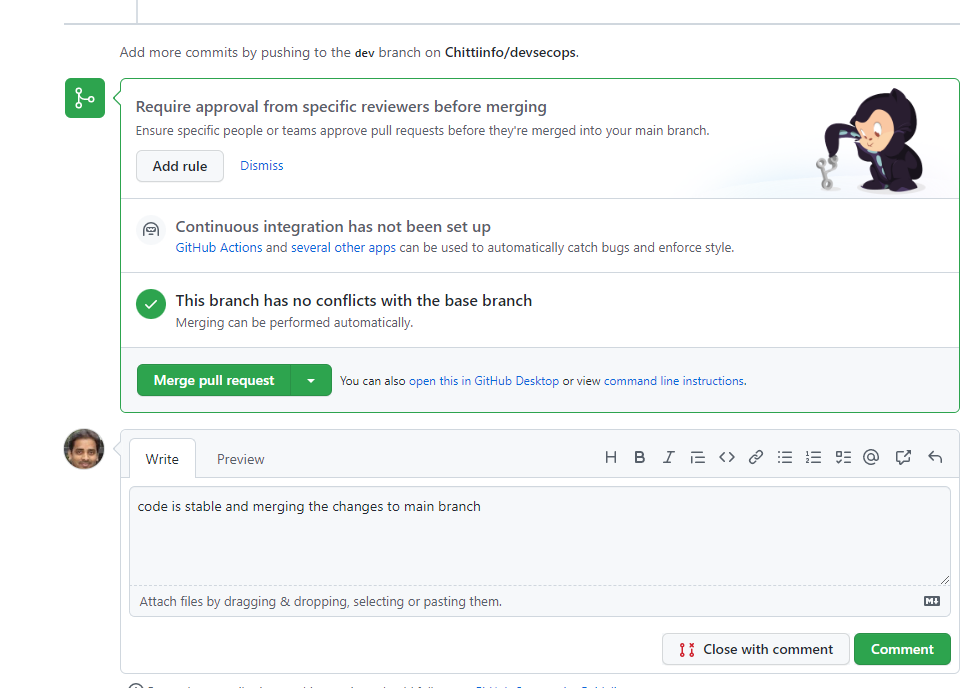


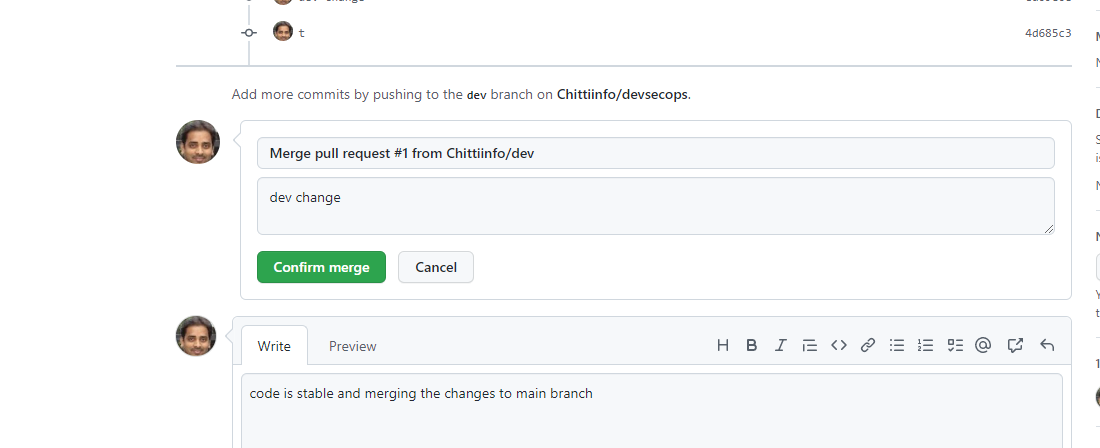
Click it and you'll be taken to this screen:

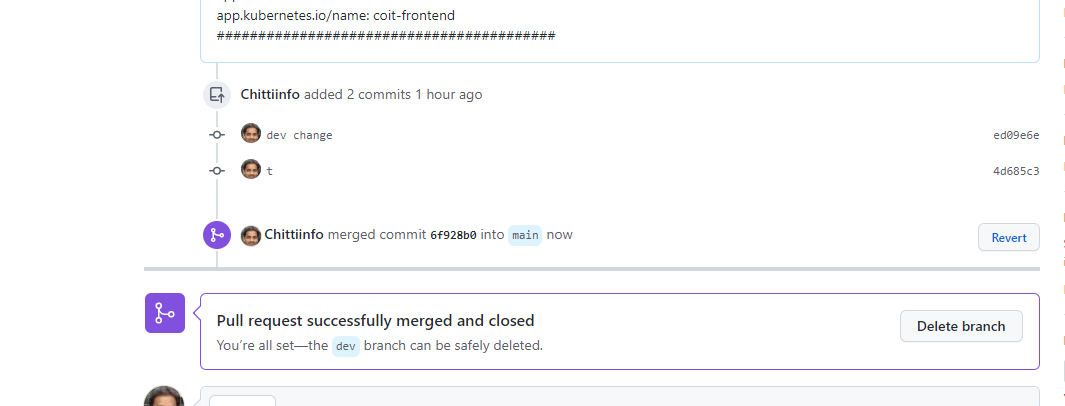


Open a pull request by clicking the Create pull request button. This allows the repo's maintainers to review your contribution. From here, they can merge it if it is good, or they may ask you to make some changes.



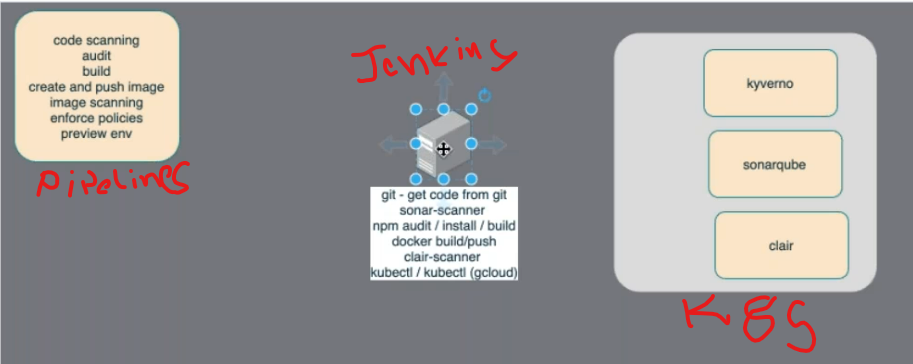






Tools Setup:

In this project, we are going to install Jenkins in dedicated EC2 server and Sonar Qube & Clair tools on GKE or EKS cluster.



**Server Side Tools:** ( Needs to be installed on GKE or EKS)

Kyverno,Sonarqube, Clair

**Client side Tools: (Needs to be installed on Jenkins Server)**

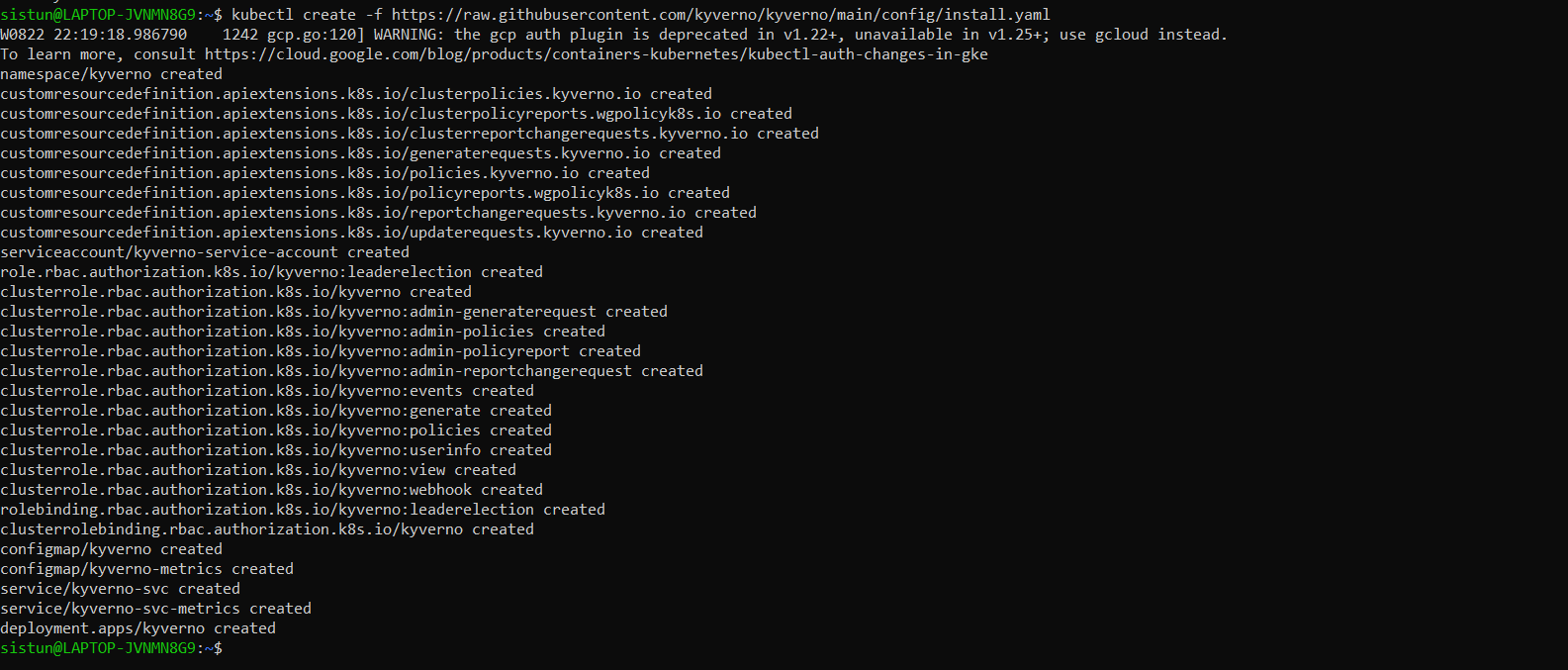
GIT, SonarScanner,Npm, docker,clair-scanner,kubectl,gcloud

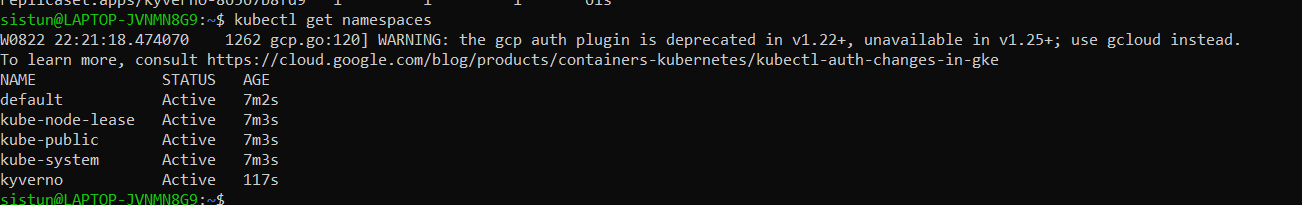
**Install Kyverno tool in GKE:**

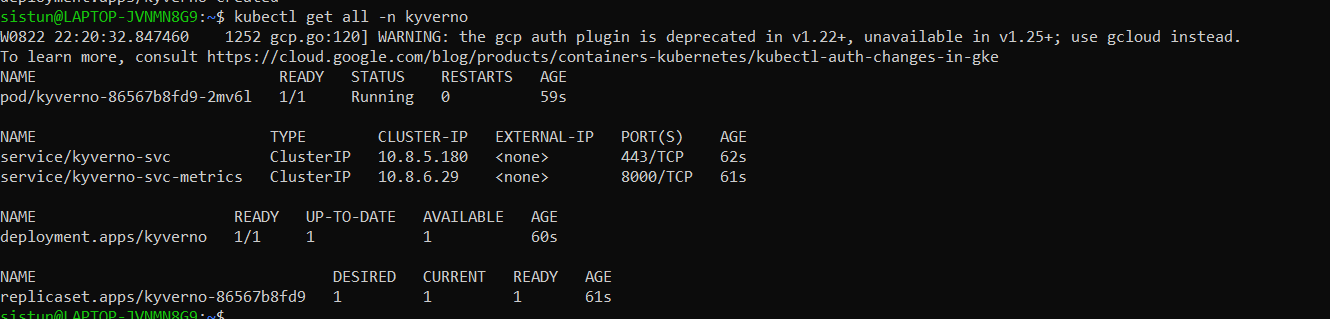
<https://kyverno.io/docs/installation/>

run below command to install kyverno

kubectl create -f https://raw.githubusercontent.com/kyverno/kyverno/main/config/install.yaml





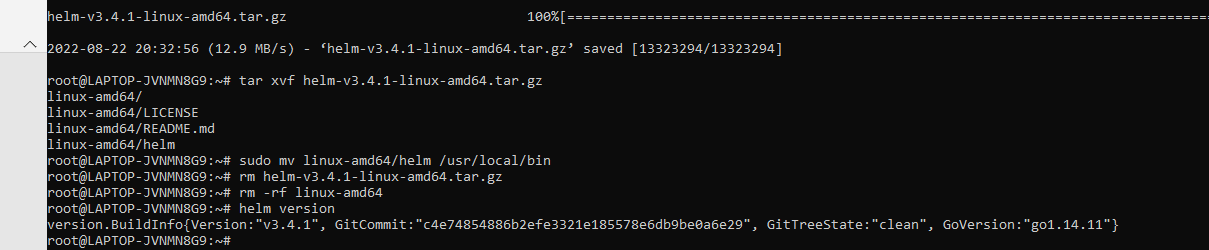


**Install SonarQube on GKE:**

We are using HELM packer manger to setup SonarQube on GKE

Note: Install HELM tool on your local Ubuntu machine before you use helm commands to install sonarqube on GKE

<https://phoenixnap.com/kb/install-helm>



https://docs.sonarqube.org/latest/setup/sonarqube-on-kubernetes/

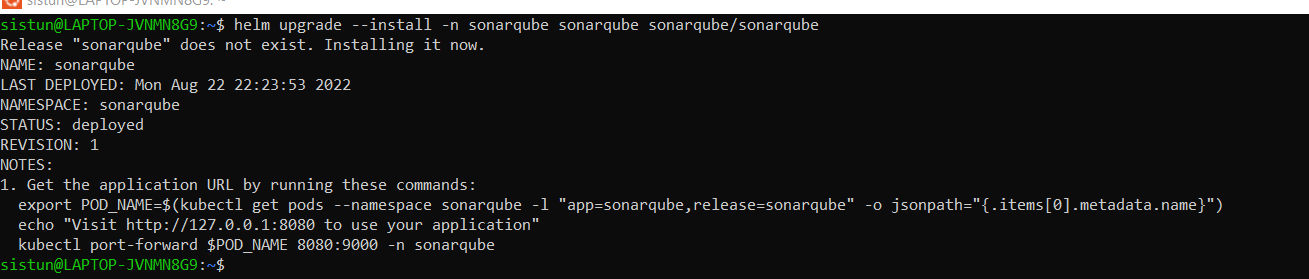
To install the Helm Chart from our Helm Repository, you can use the following commands:

helm repo add sonarqube https://SonarSource.github.io/helm-chart-sonarqube

helm repo update

kubectl create namespace sonarqube

helm upgrade --install -n sonarqube sonarqube sonarqube/sonarqube



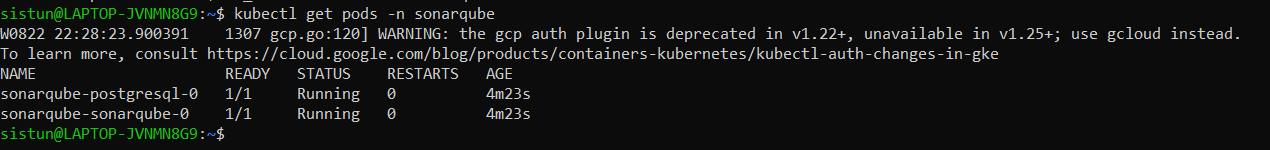
NOTES:

1. Get the application URL by running these commands:

export POD\_NAME=$(kubectl get pods --namespace sonarqube -l "app=sonarqube,release=sonarqube" -o jsonpath="{.items[0].metadata.name}")

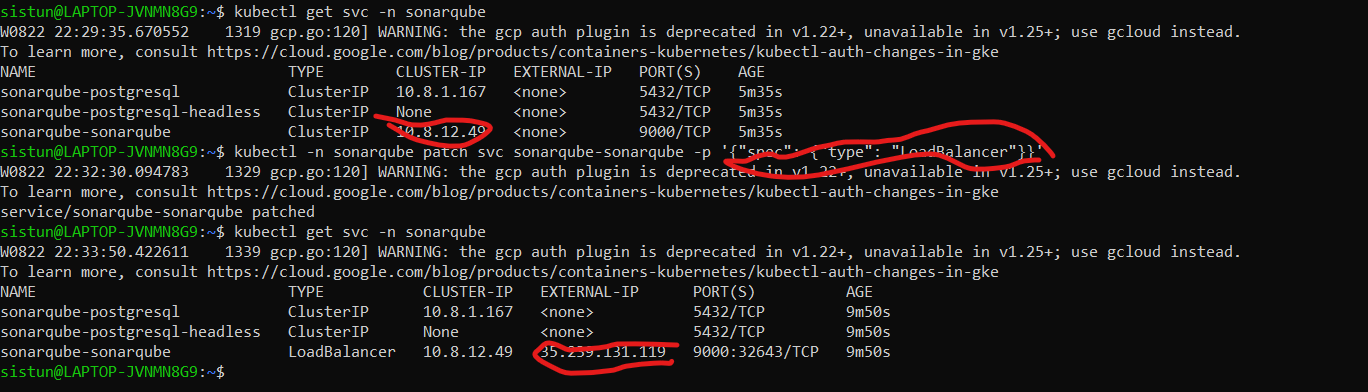
echo "Visit http://127.0.0.1:8080 to use your application"

kubectl port-forward $POD\_NAME 8080:9000 -n sonarqube



By default sonarqube created as a cluster IP and we can change it to load balancer type by using patch command so that we can use sonarqube for Jenkins communication.

kubectl -n sonarqube patch svc sonarqube-sonarqube -p '{"spec": {"type": "LoadBalancer"}}'



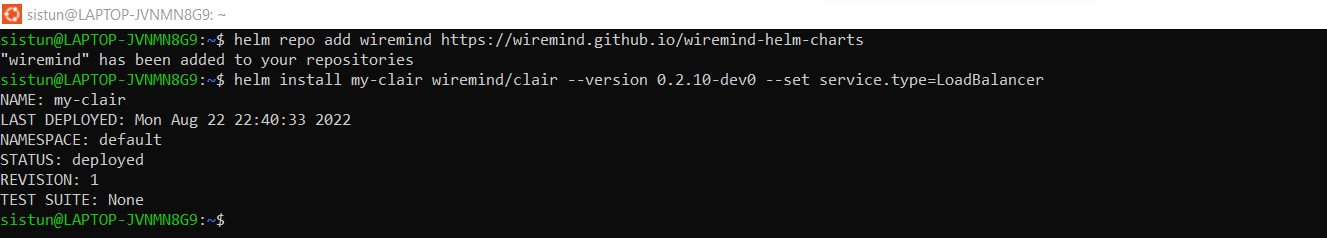
**Install Clair On GKE: Use HELM chart**

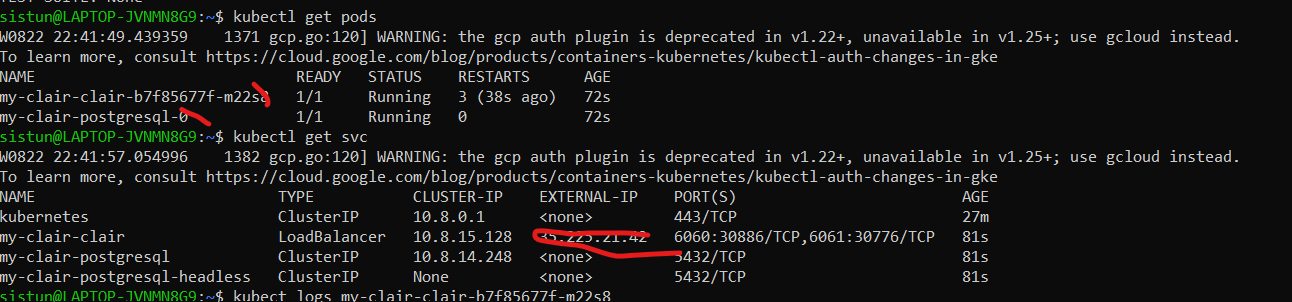
<https://artifacthub.io/packages/helm/wiremind/clair?modal=install>

helm repo add wiremind <https://wiremind.github.io/wiremind-helm-charts>

Make sure service type is changed to Load Balancer

helm install my-clair wiremind/clair --version 0.2.10-dev0 --set service.type=LoadBalancer

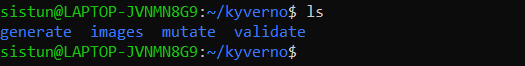


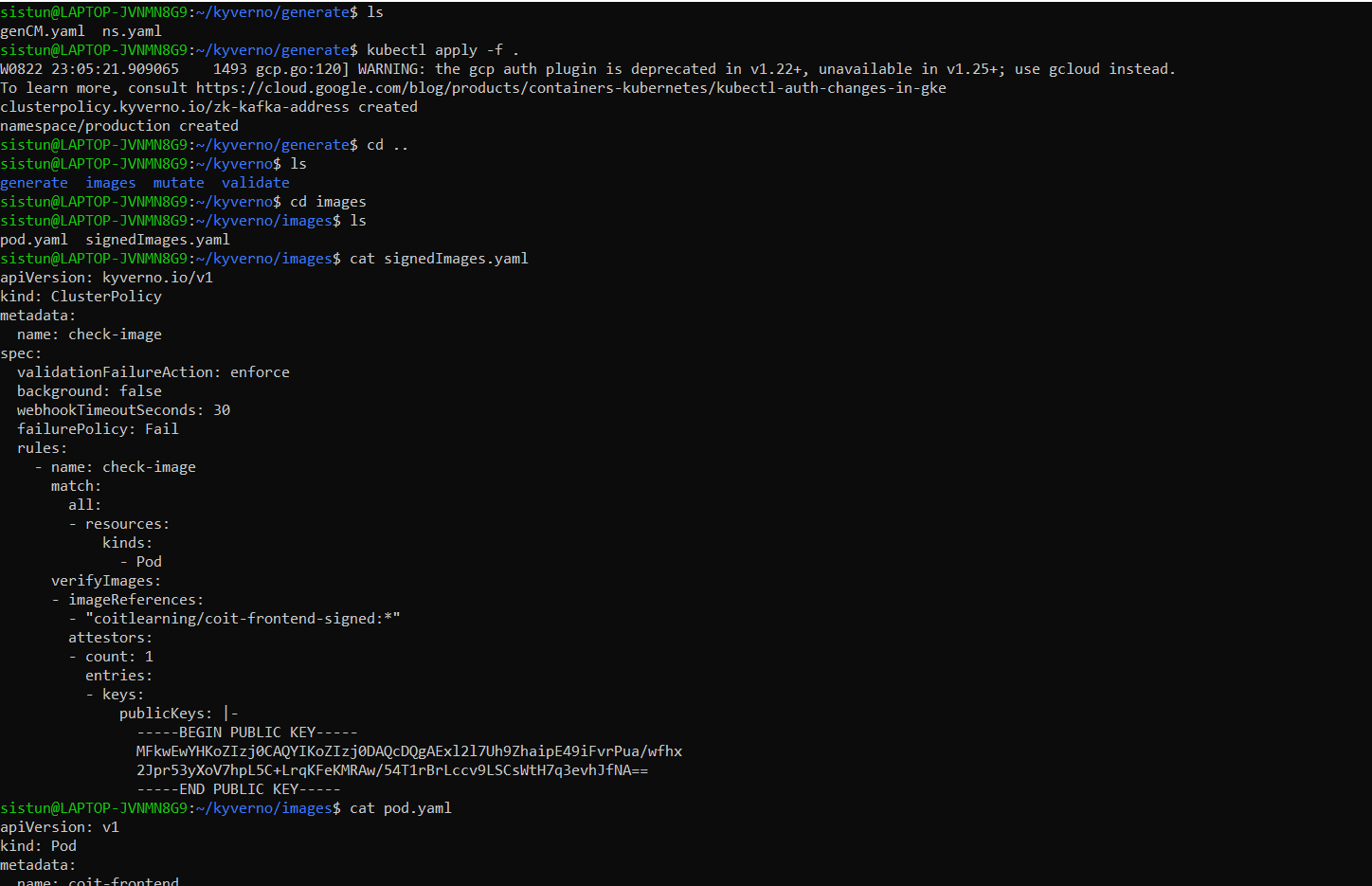


**Setup Kverno admission control Policies:**

**Clone all the polices from here** [**https://github.com/Chittiinfo/kyverno**](https://github.com/Chittiinfo/kyverno) **and apply them on GKE**

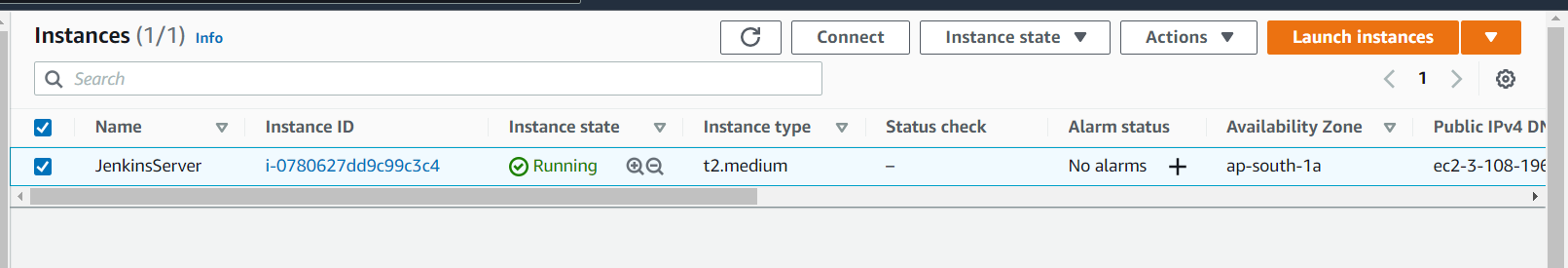
**Applly all below polices just by running kubectl apply -f .**

****

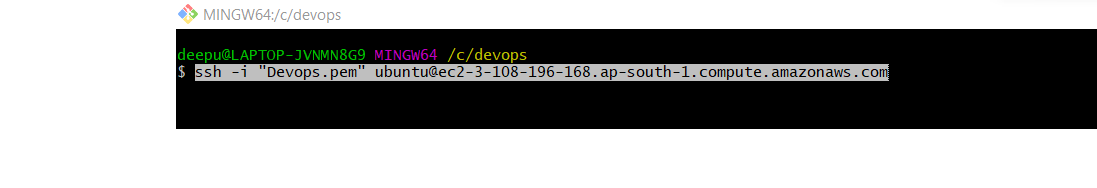
****

Setup Jenkins Server:

Setup Jenkins Server with t2.medium and 40 GB space in AWS



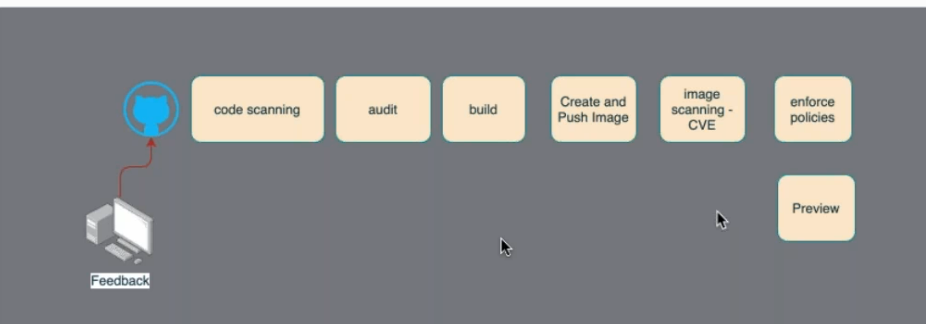
Login into Jenkins server



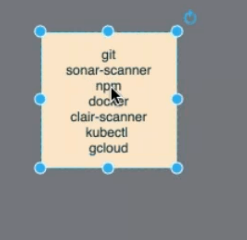
**Install Jenkins on EC2 machine.**

<https://www.jenkins.io/doc/book/installing/linux/#debianubuntu>

High Level jenkins pipelines steps:

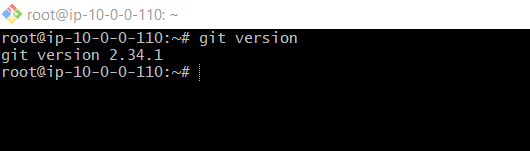


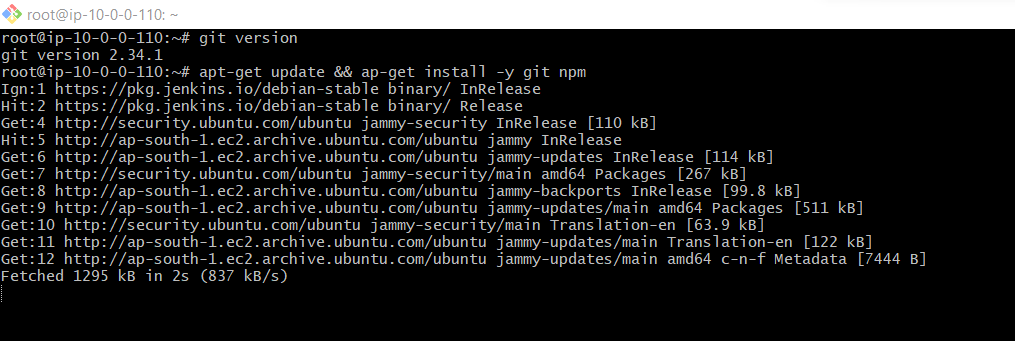
**Install all client side software’s on Jenkins Server:**

****

**Insatll Git and NPM:**

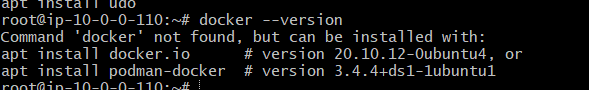
apt-get update && ap-get install -y git npm





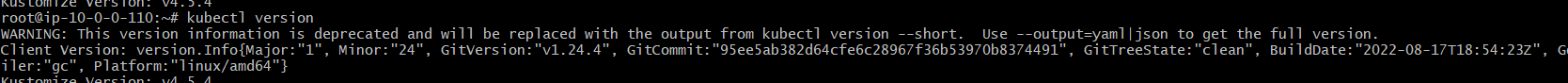
**Install Docker**:

<https://docs.docker.com/engine/install/ubuntu/>



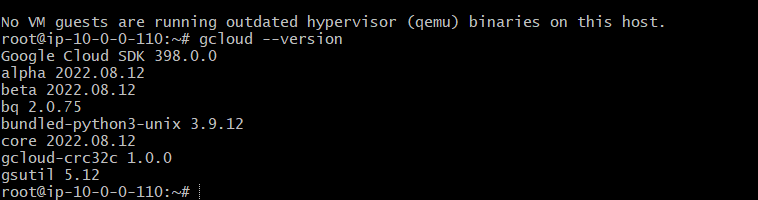
**Install KubeCtl** (Root user on Jenkins)

<https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/#install-kubectl-binary-with-curl-on-linux>



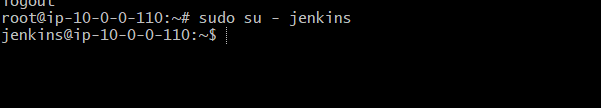
**Install Gcloud** (Root user on Jenkins)

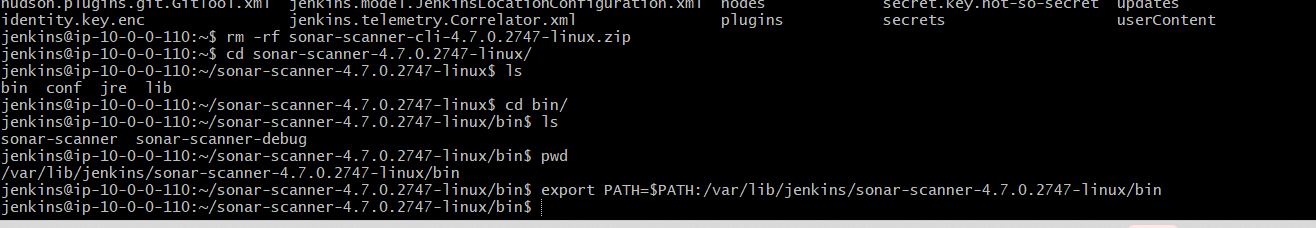
<https://cloud.google.com/sdk/docs/install#deb>



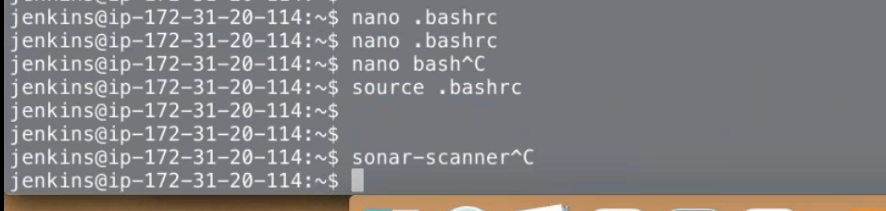
**Sonar Scanner:**

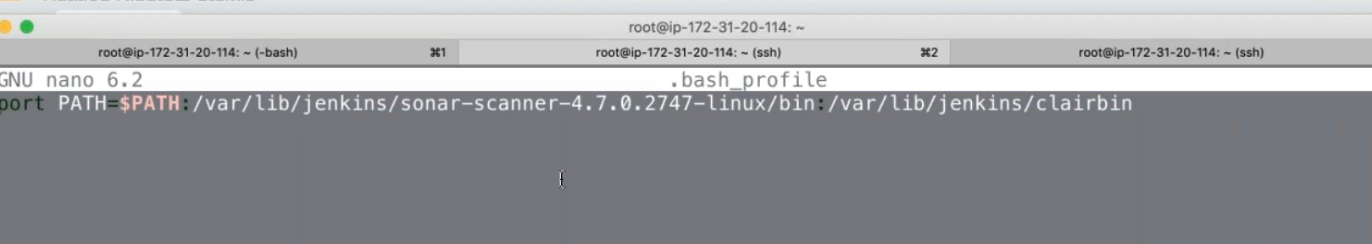
Switch to Jenkins user





export PATH=$PATH:/var/lib/jenkins/sonar-scanner-4.7.0.2747-linux/bin

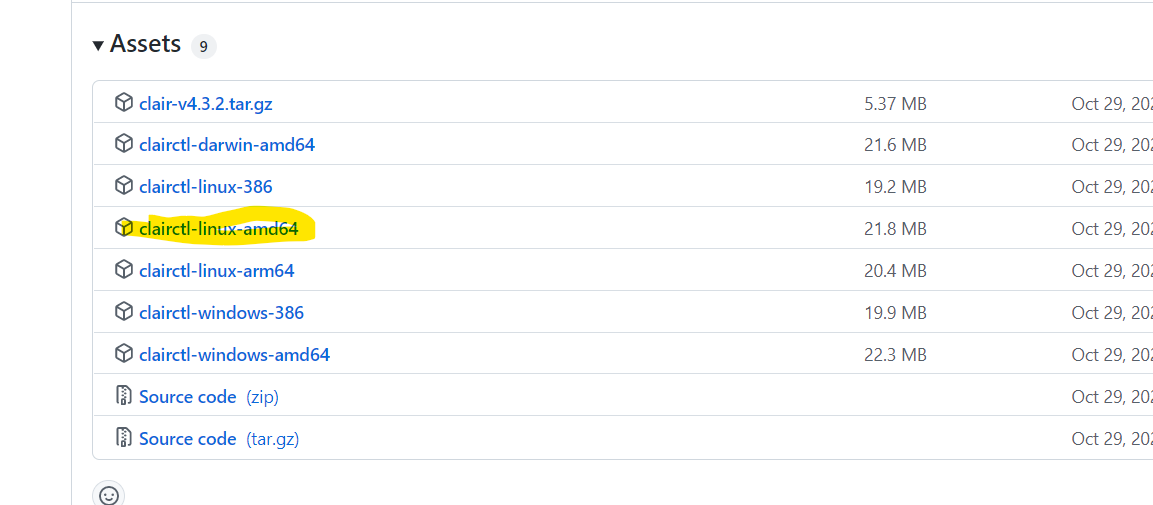




Make sure above path is placed id .bashrc and run source .bashrc command to use sonar-scanner from anywhere

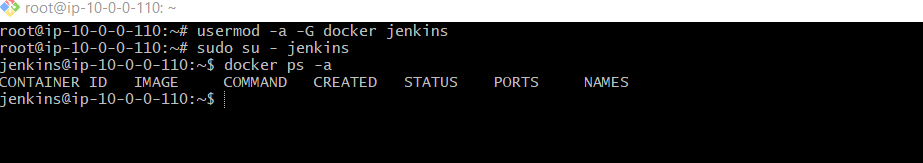
**Insatll Clair Scanner:**

<https://github.com/quay/clair/releases?page=2>

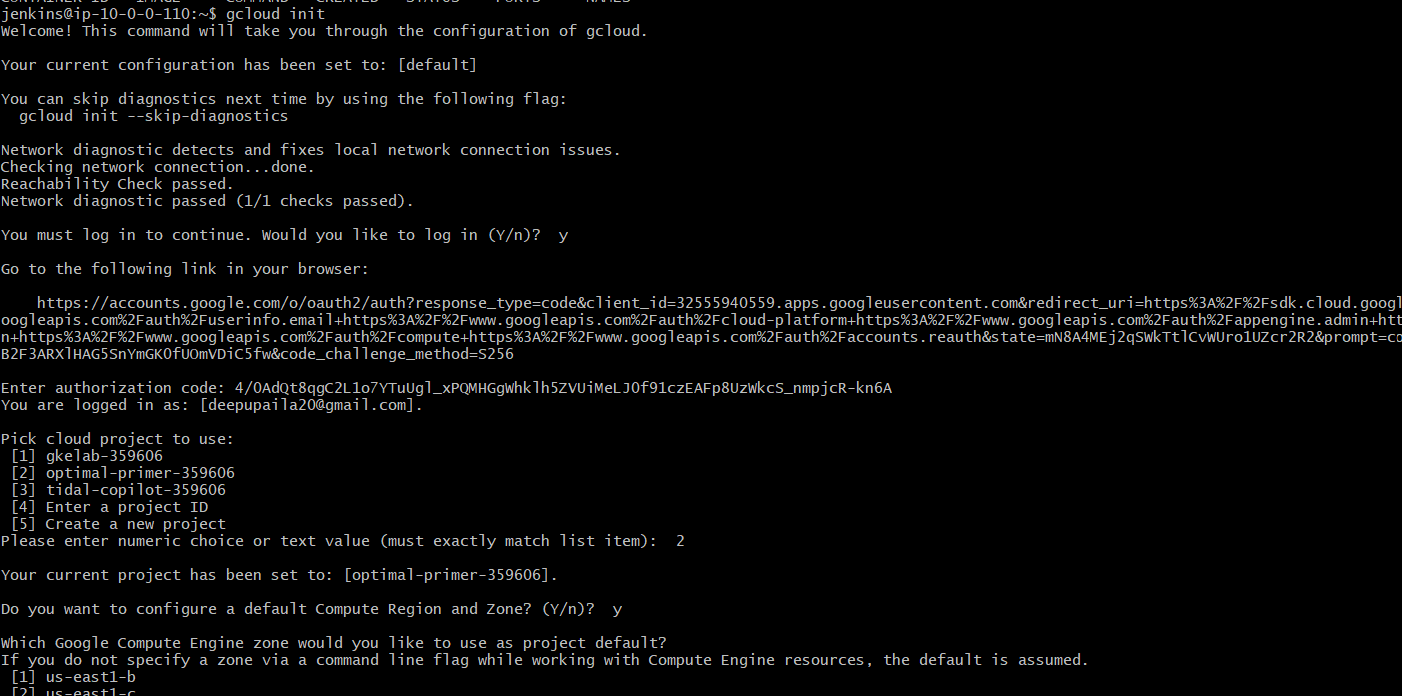


**Initial Configurations on Jenkins:**

Grant docker access to Jenkins user.

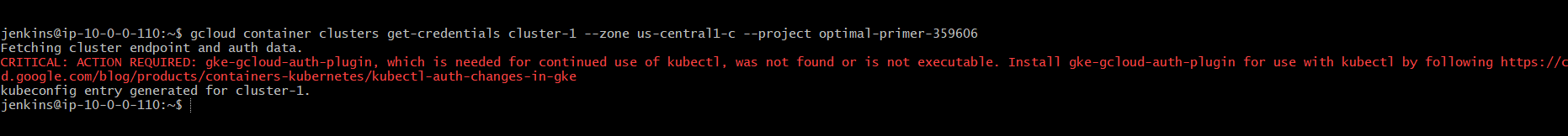


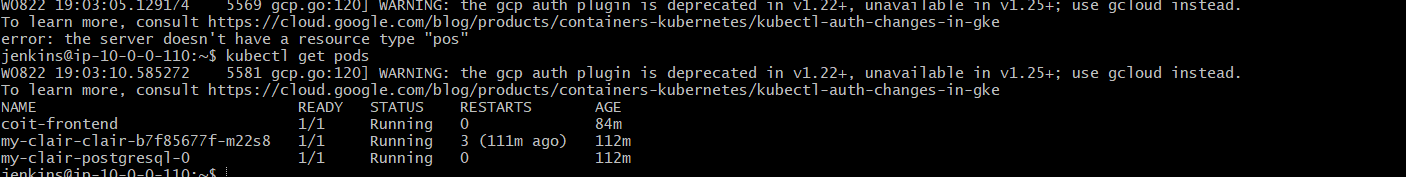
Connection to GCloud:



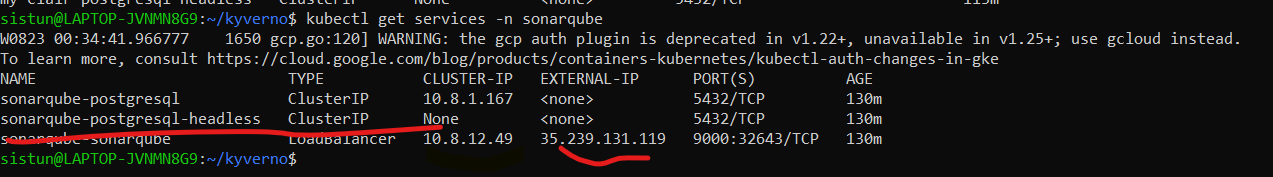
Connect to active GKE cluster with below command.

gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project optimal-primer-359606

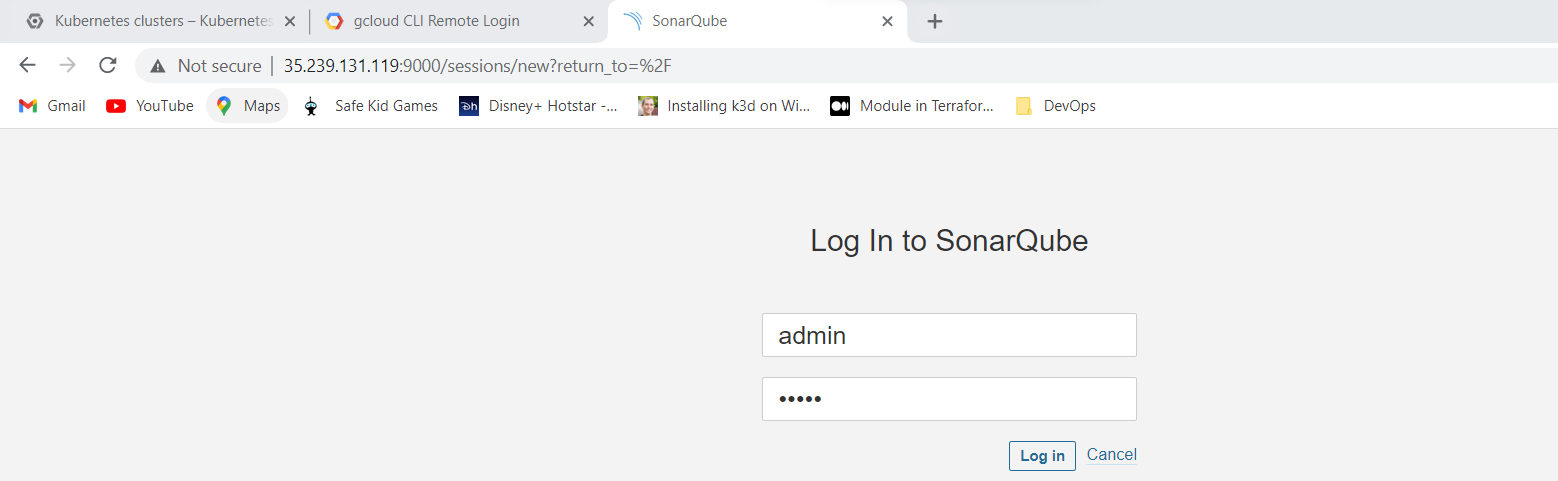




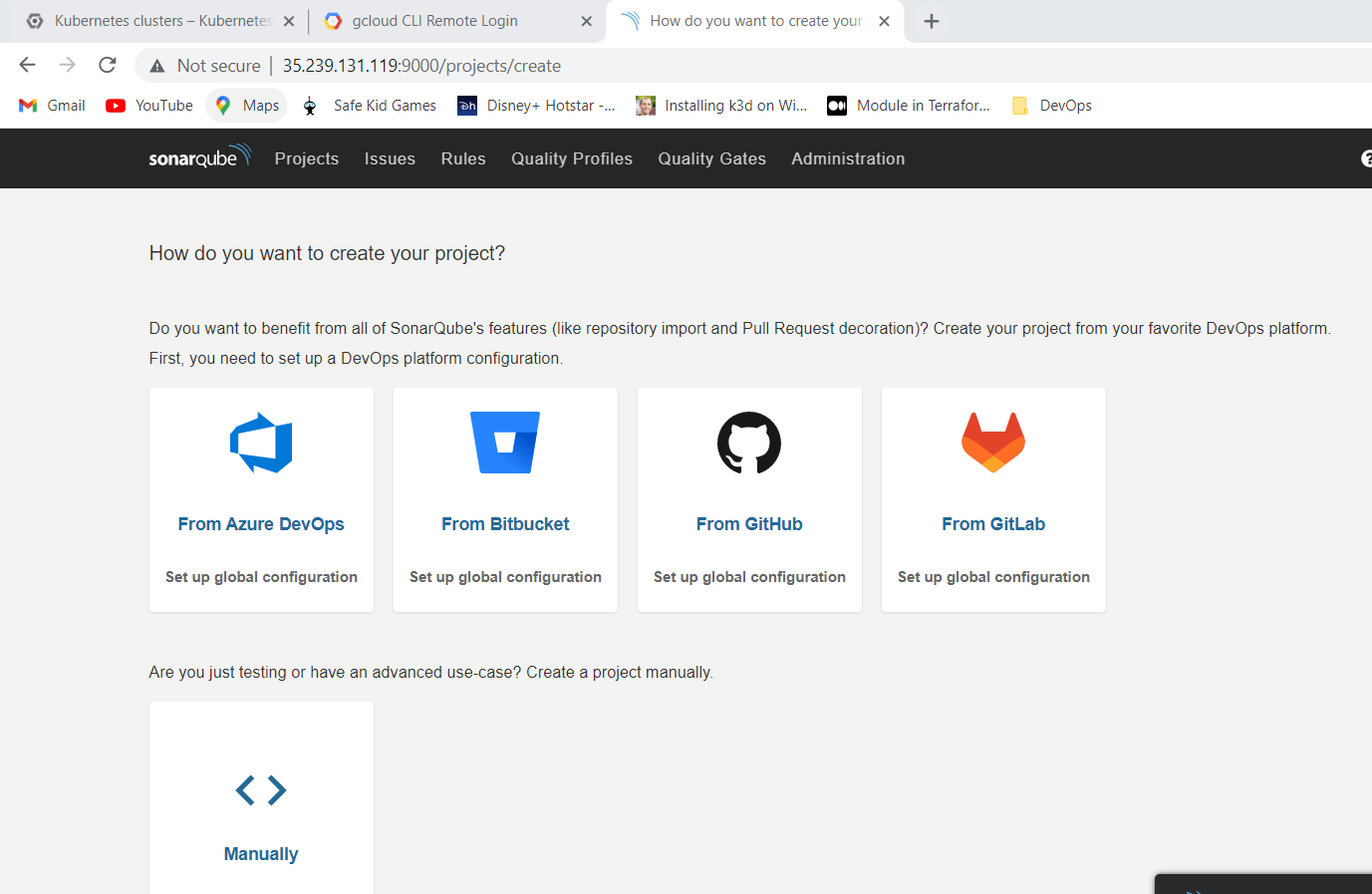
Sonar Qube Connection Setup:



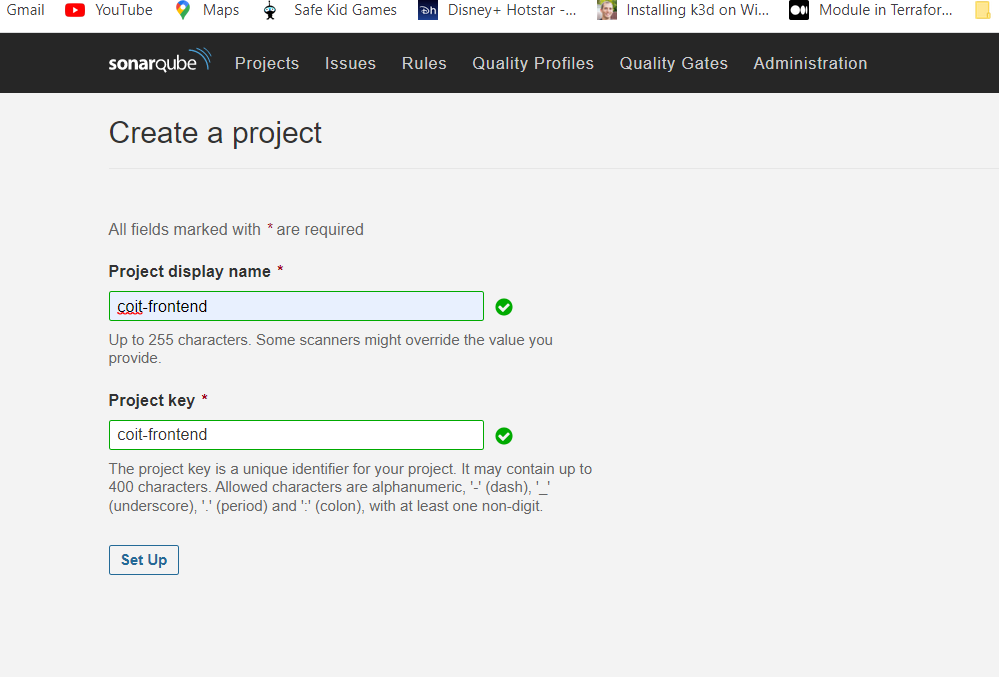
<http://35.239.131.119:9000/sessions/new?return_to=%2F>



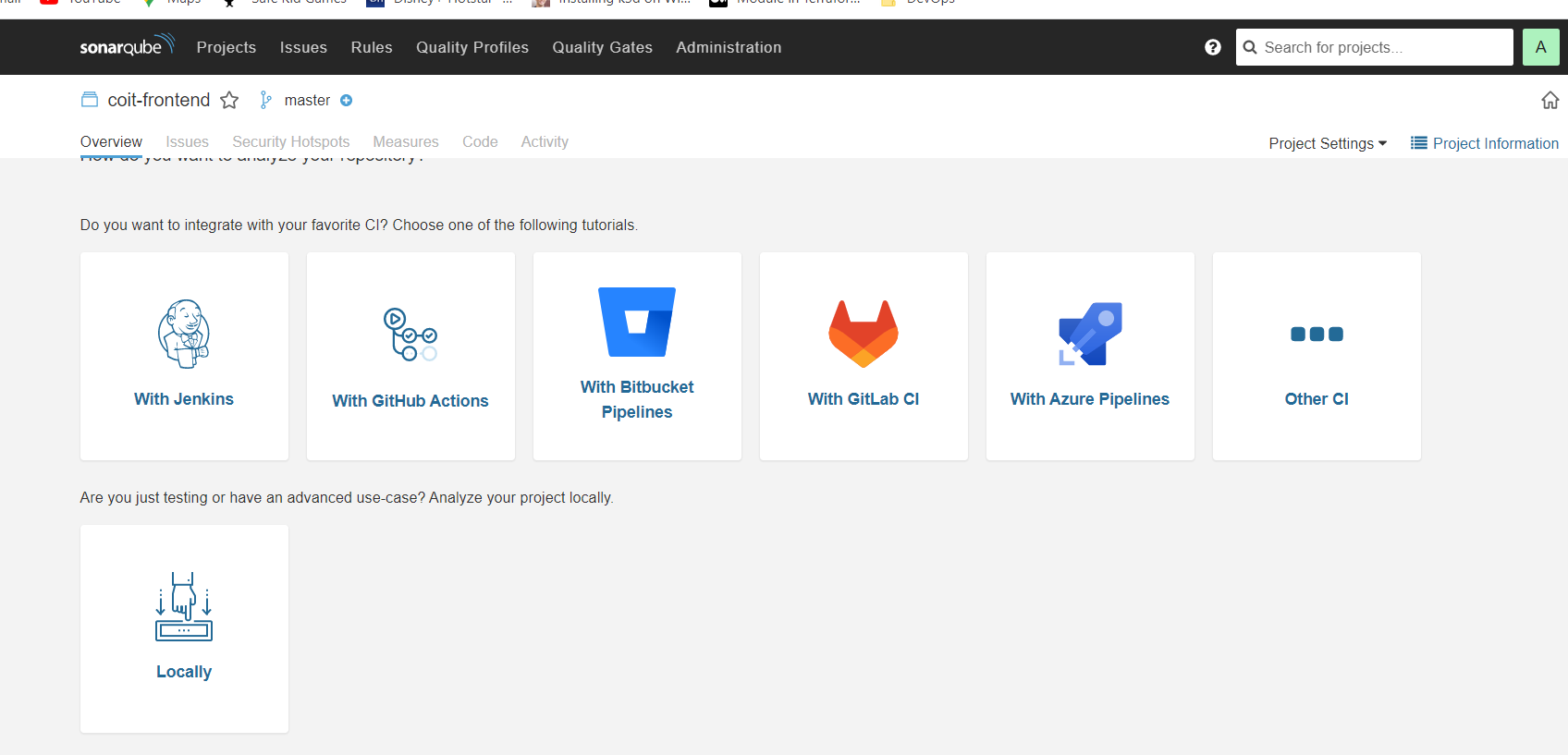
Select Manaully



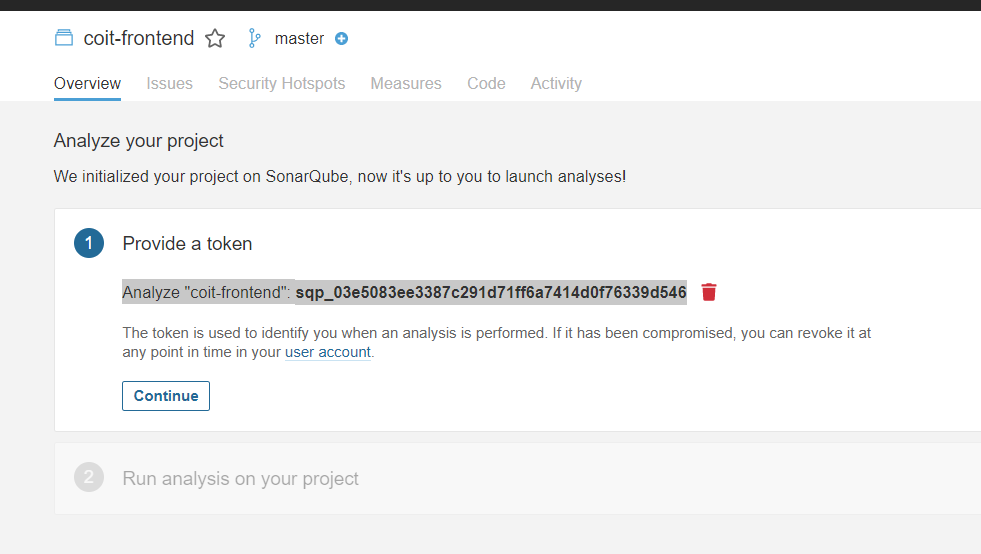
Create a Project:

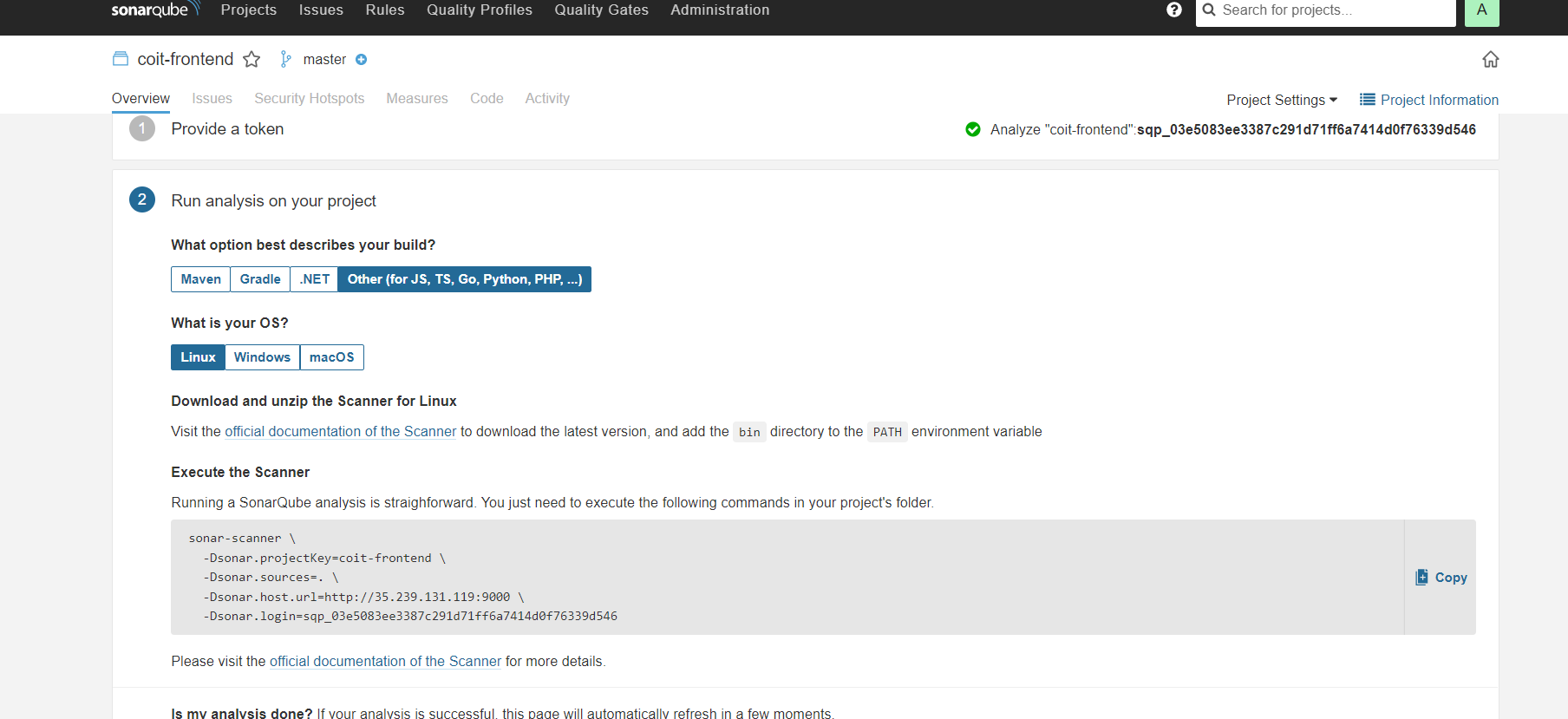


Select Locally:



Analyze "coit-frontend": **sqp\_03e5083ee3387c291d71ff6a7414d0f76339d546**





sonar-scanner \

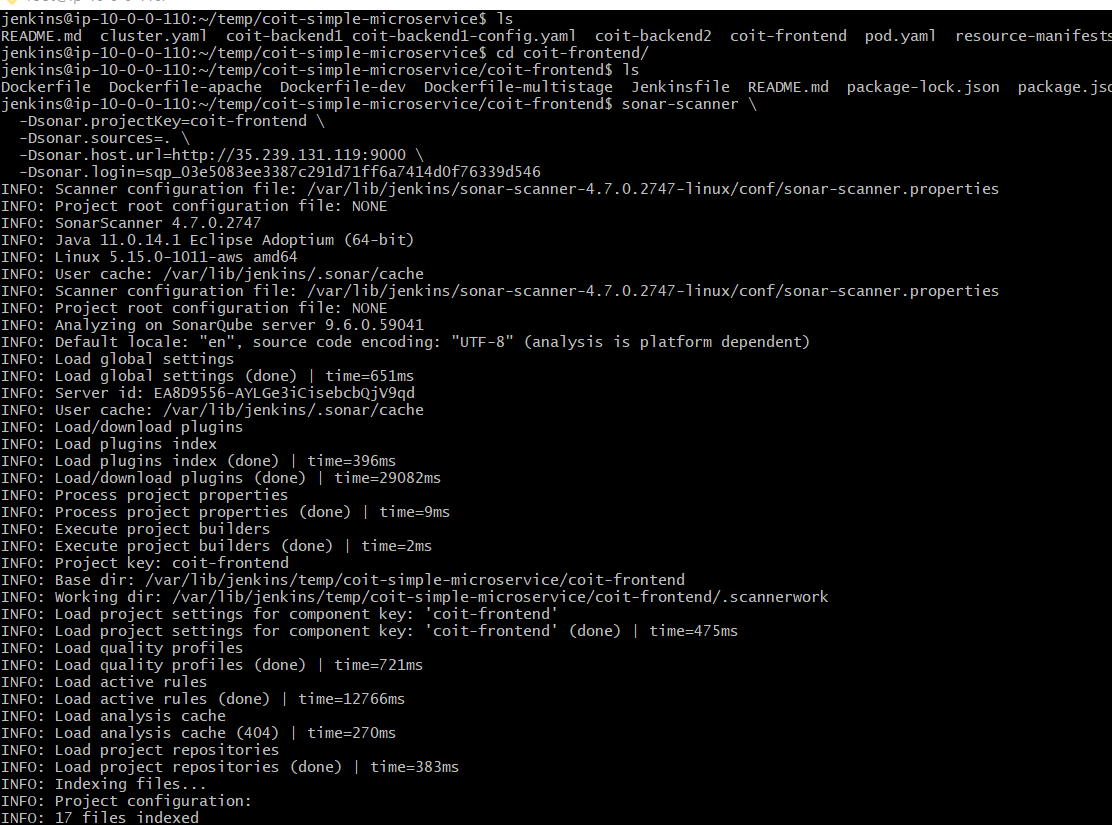
-Dsonar.projectKey=coit-frontend \

-Dsonar.sources=. \

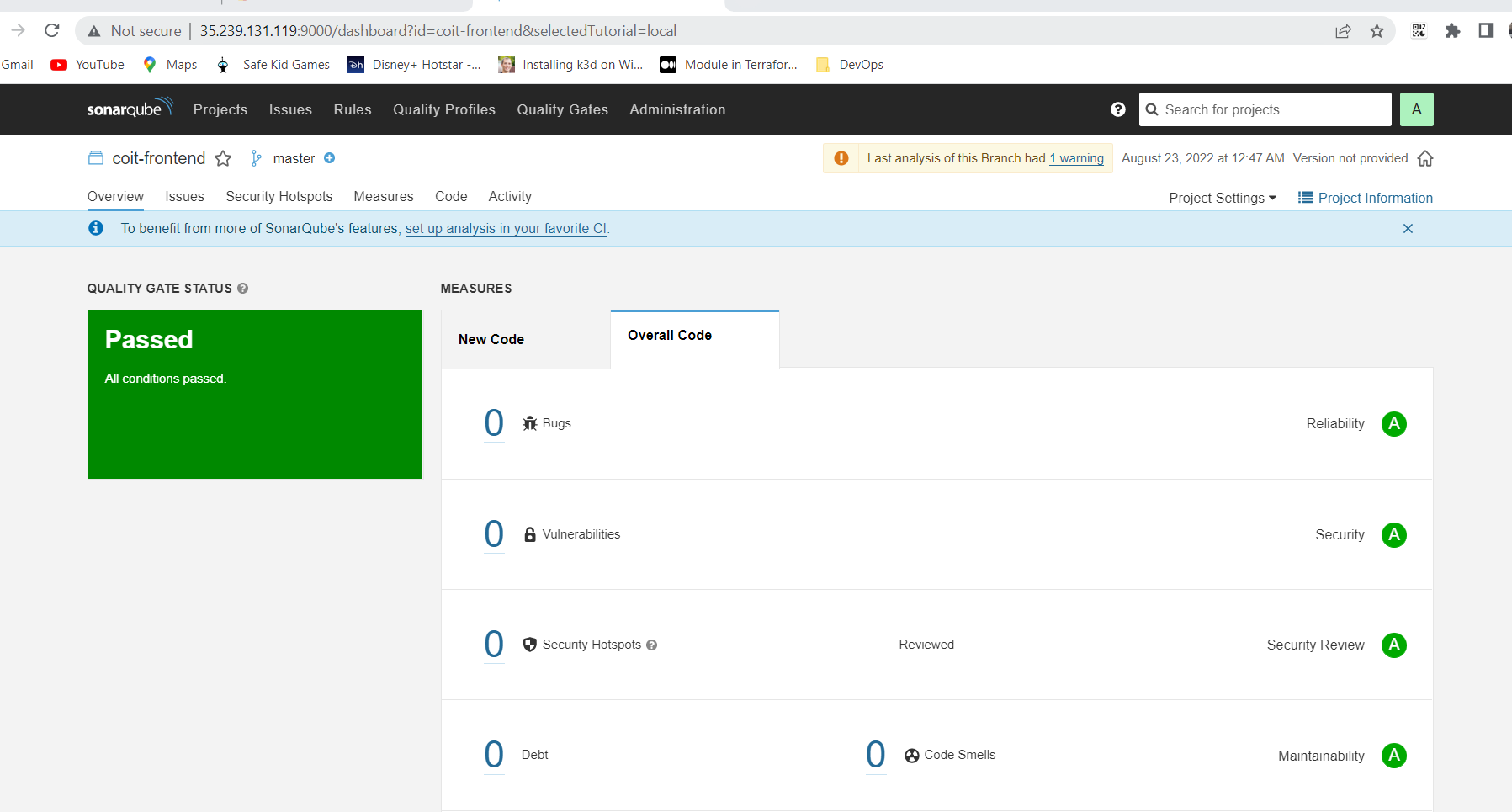
-Dsonar.host.url=http://35.239.131.119:9000 \

-Dsonar.login=sqp\_03e5083ee3387c291d71ff6a7414d0f76339d546

**Run sonar scanner from Jenkins server.**

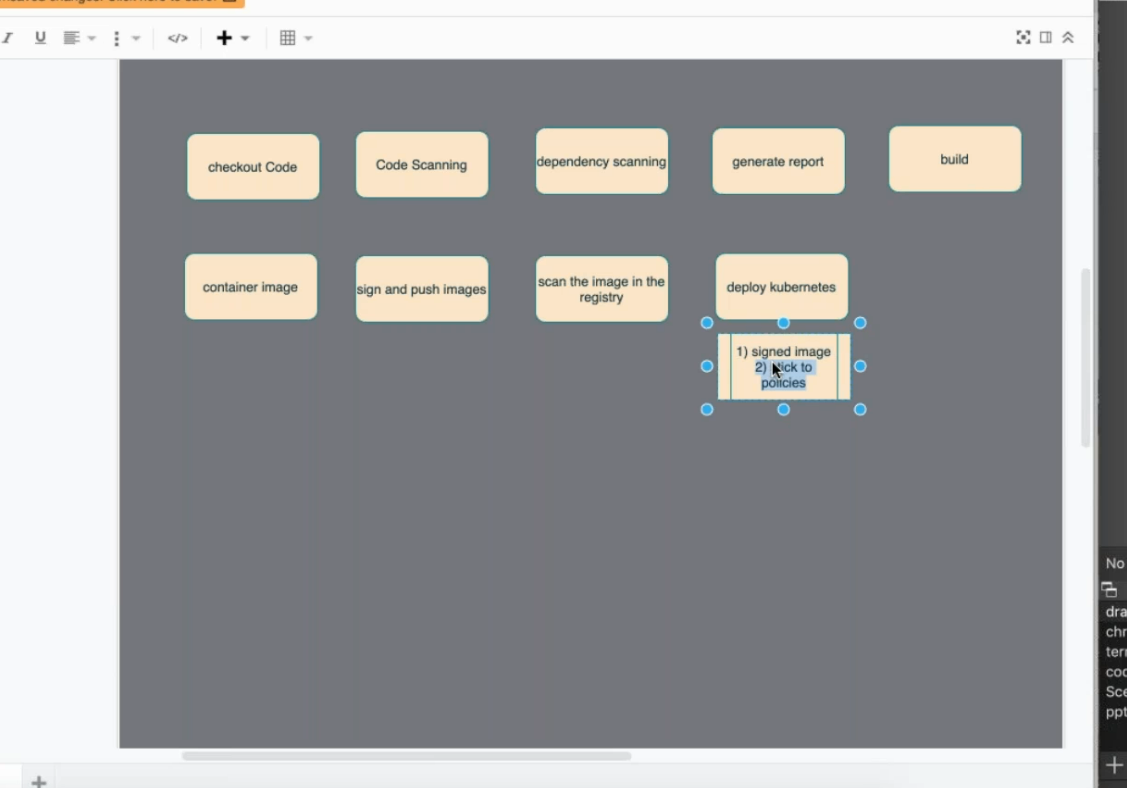


Once scan is successful we can see results in SonarQube console.



**Jenkins Pipeline setup**:

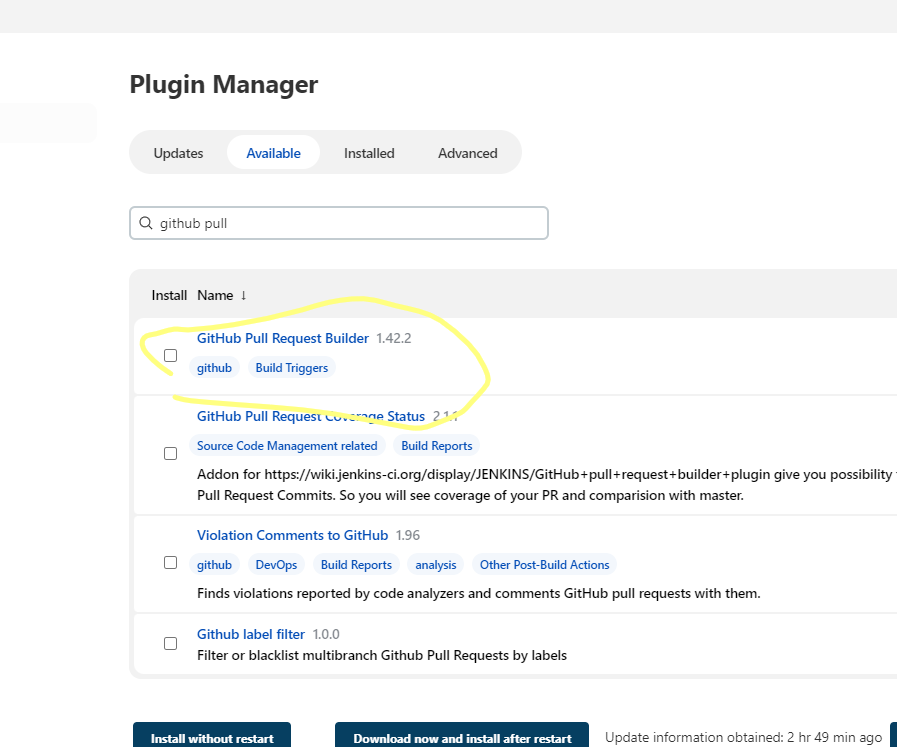
We are going to create below jobs in jenkins

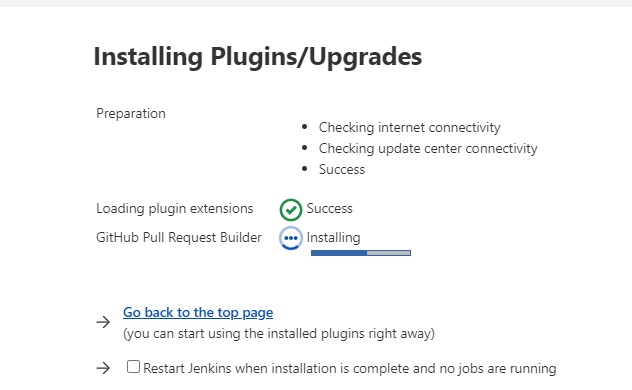


**Preparing Jenkins job to start CI job automatically when there is a pull request in GitHub:**

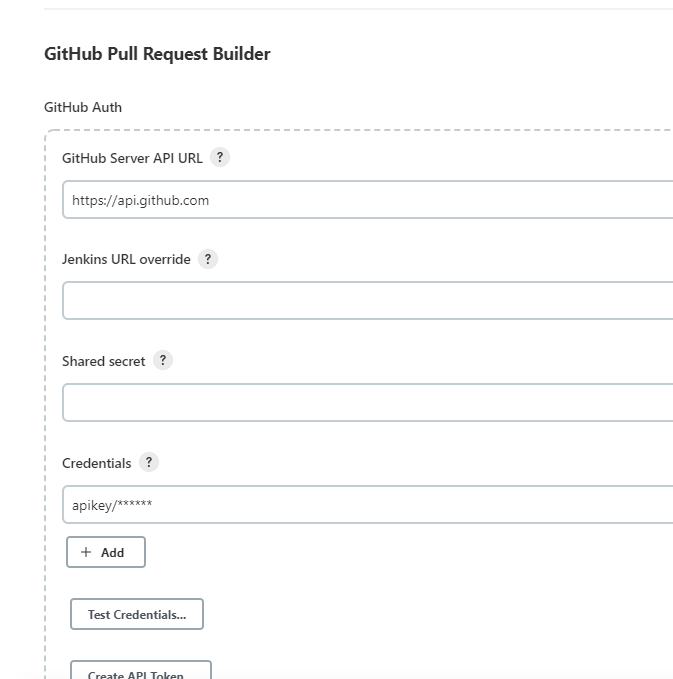
<https://www.youtube.com/watch?v=IaCBp5EwKJE&t=157s>

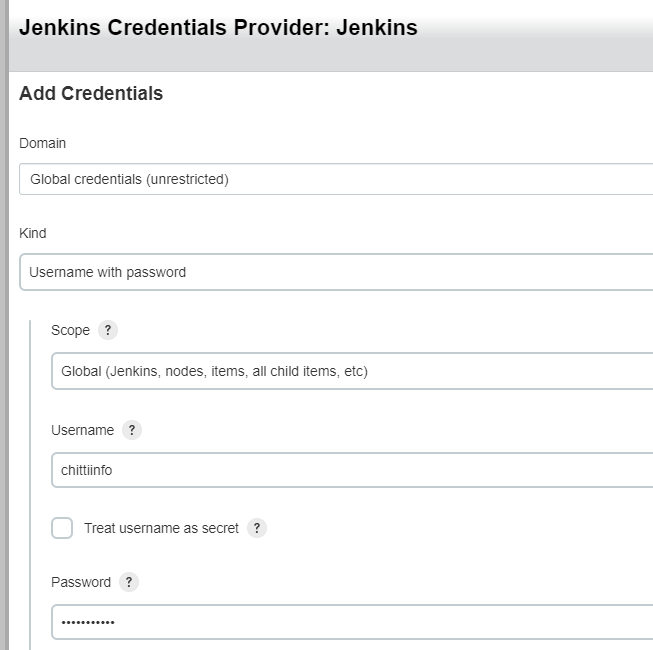
Since our automated CI process should approve this pull request and merge the code with the develop branch if it validates the code change I am using below “github pull request builder” plugin to do that job.

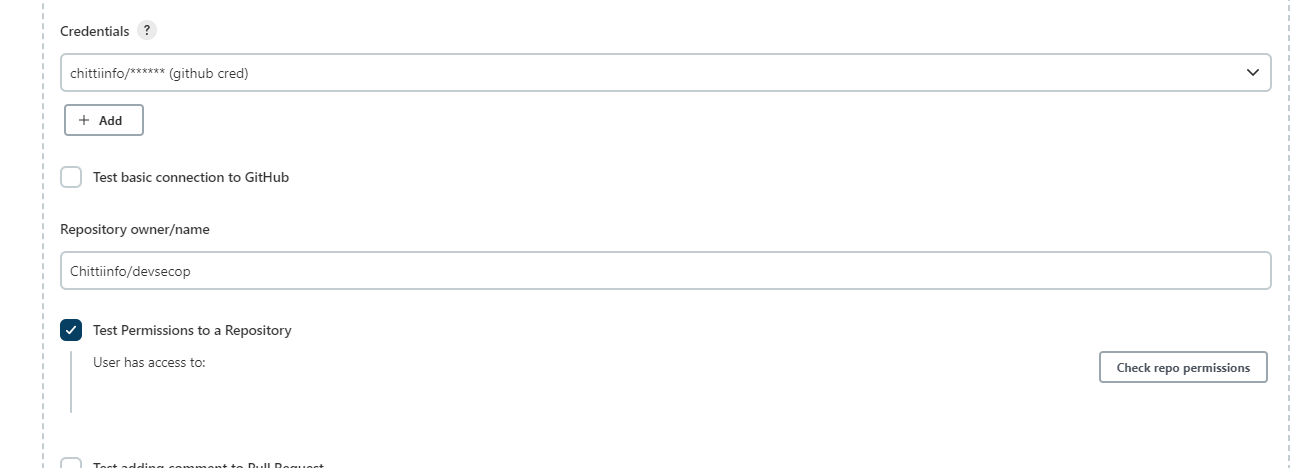




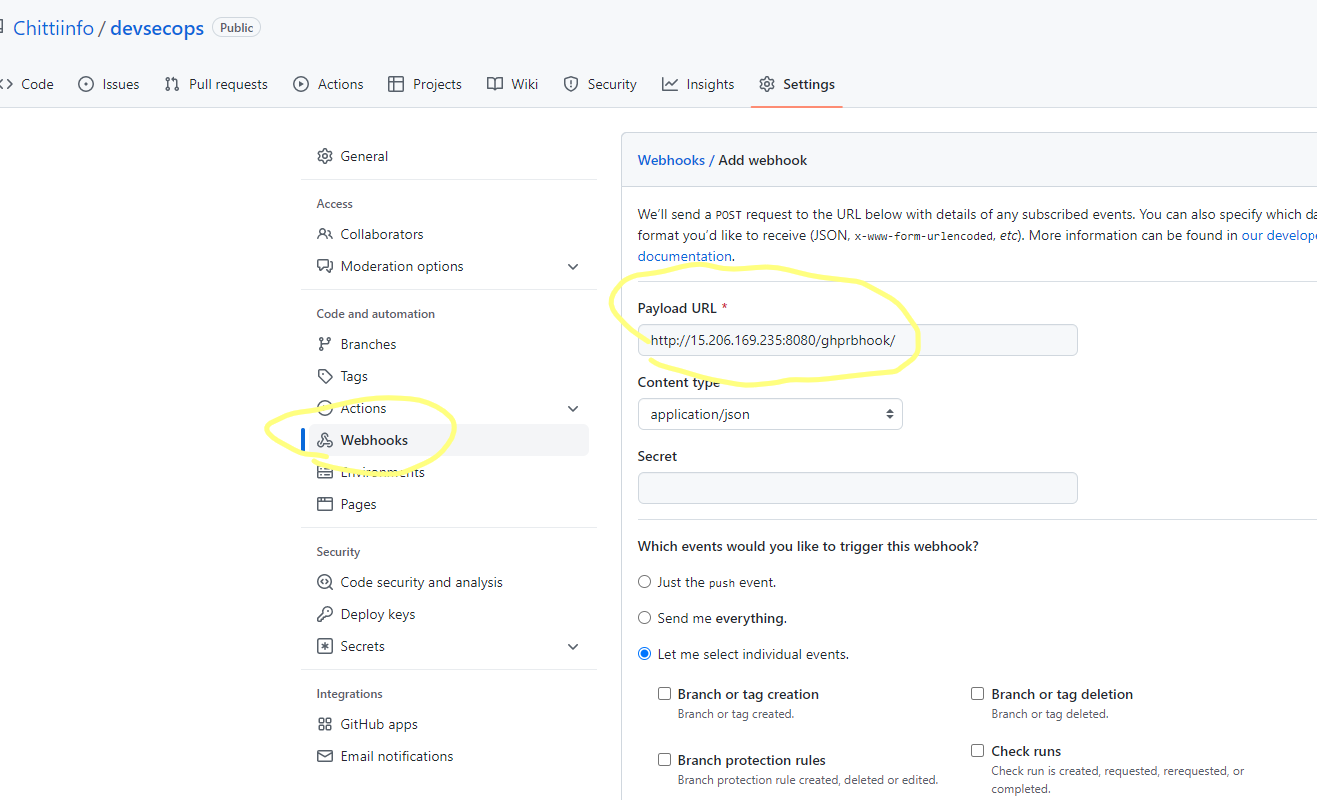
Once plugin is installed configure pull request from Jenkins Configure System

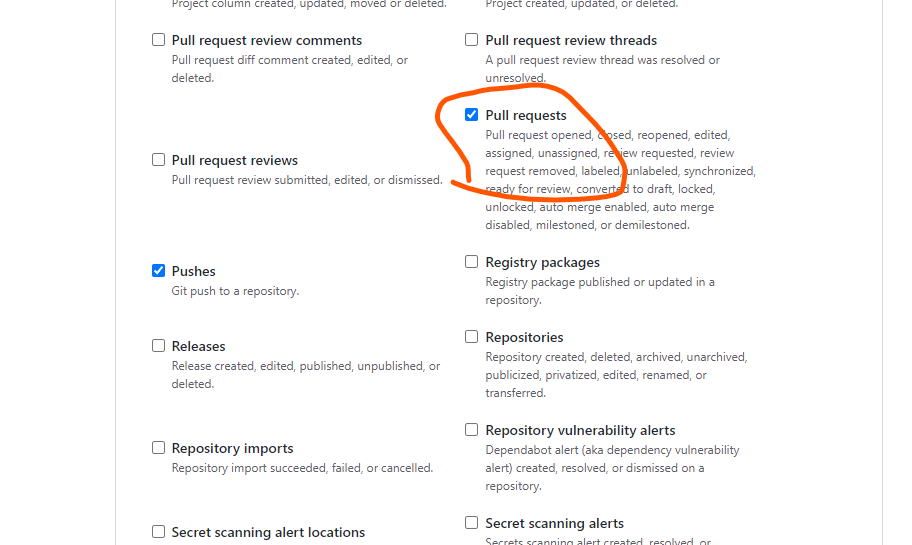


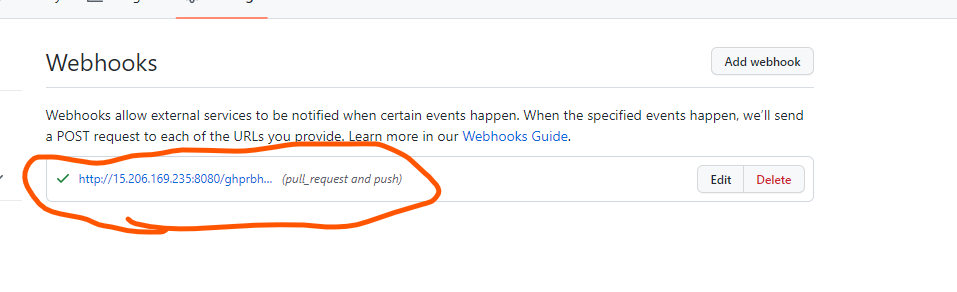




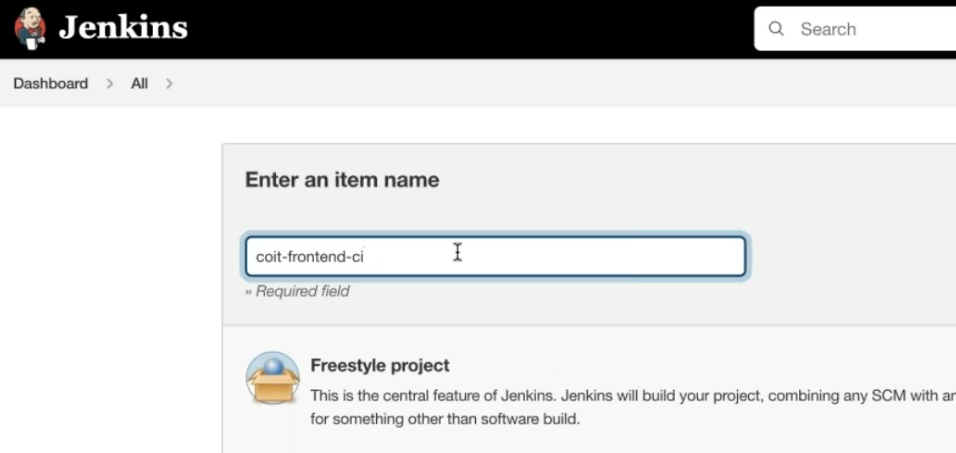
Configure below settings from GitHUb repository.

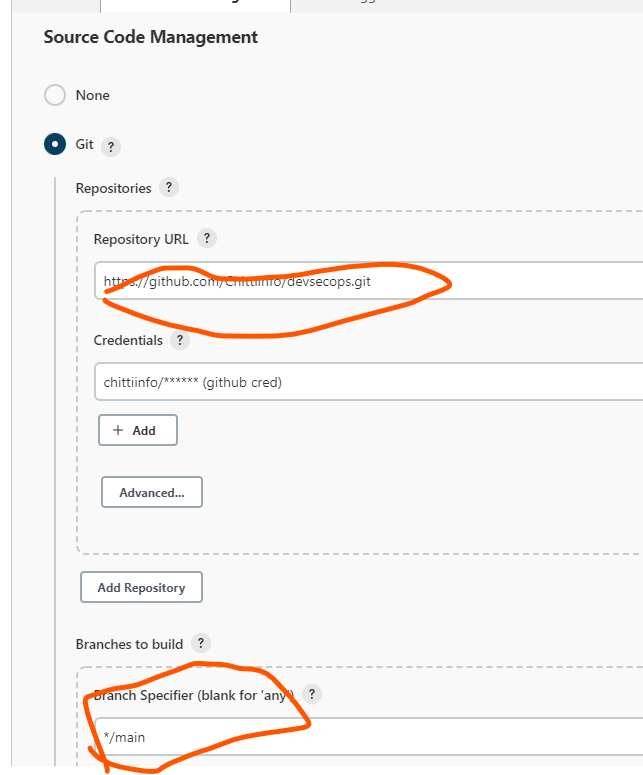


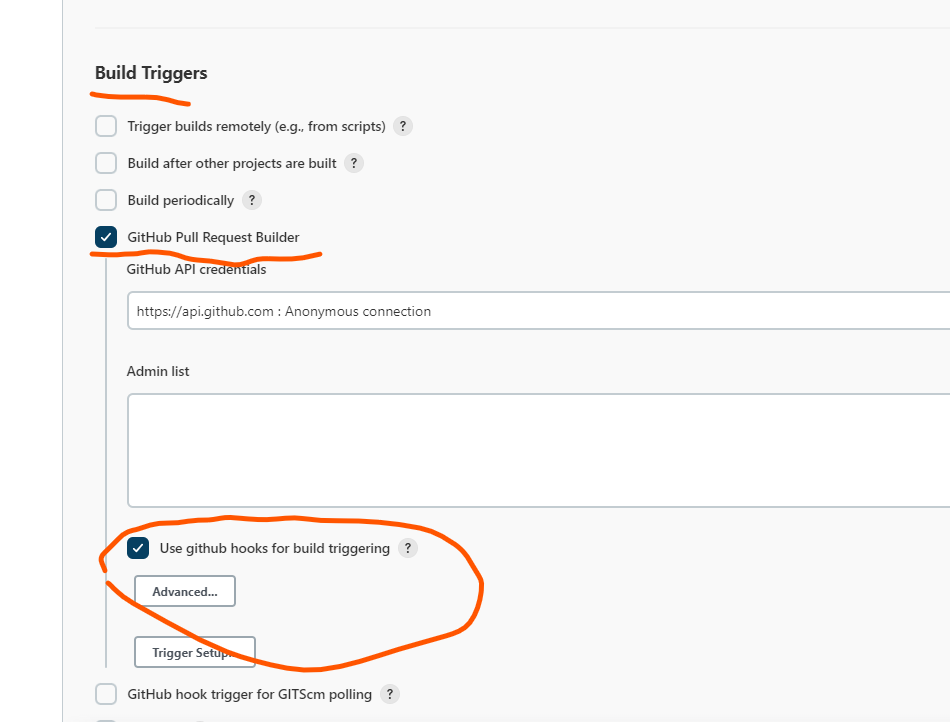




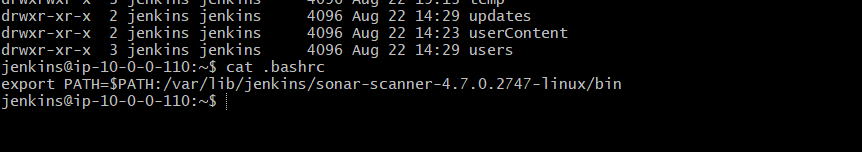
**Create a free style job:**

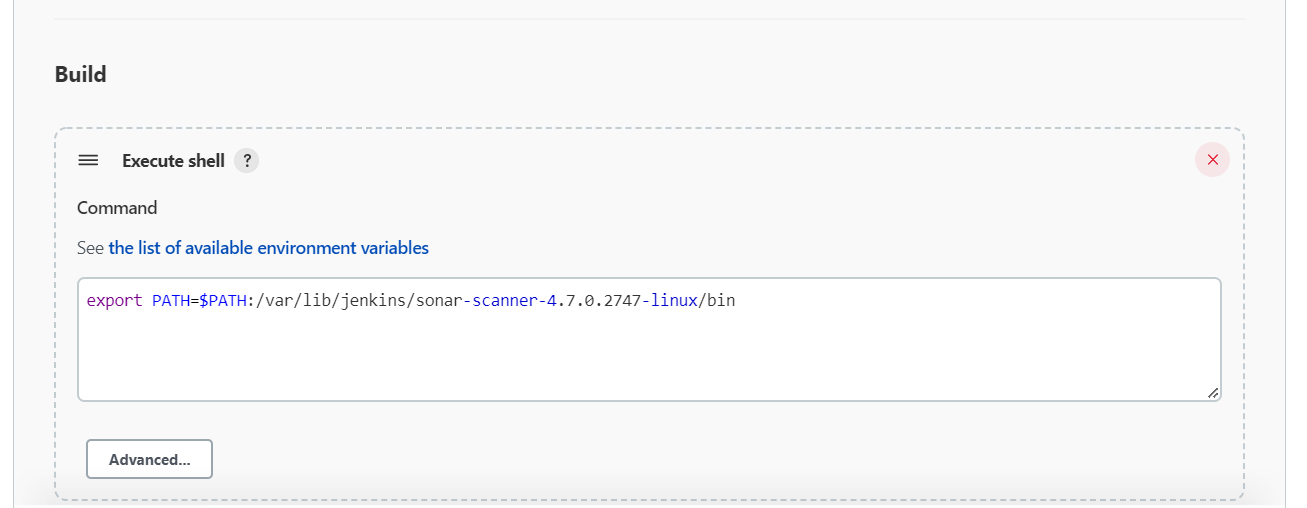






Get enviorment variable from Jenkins server from .bashrc file and include in pipeline





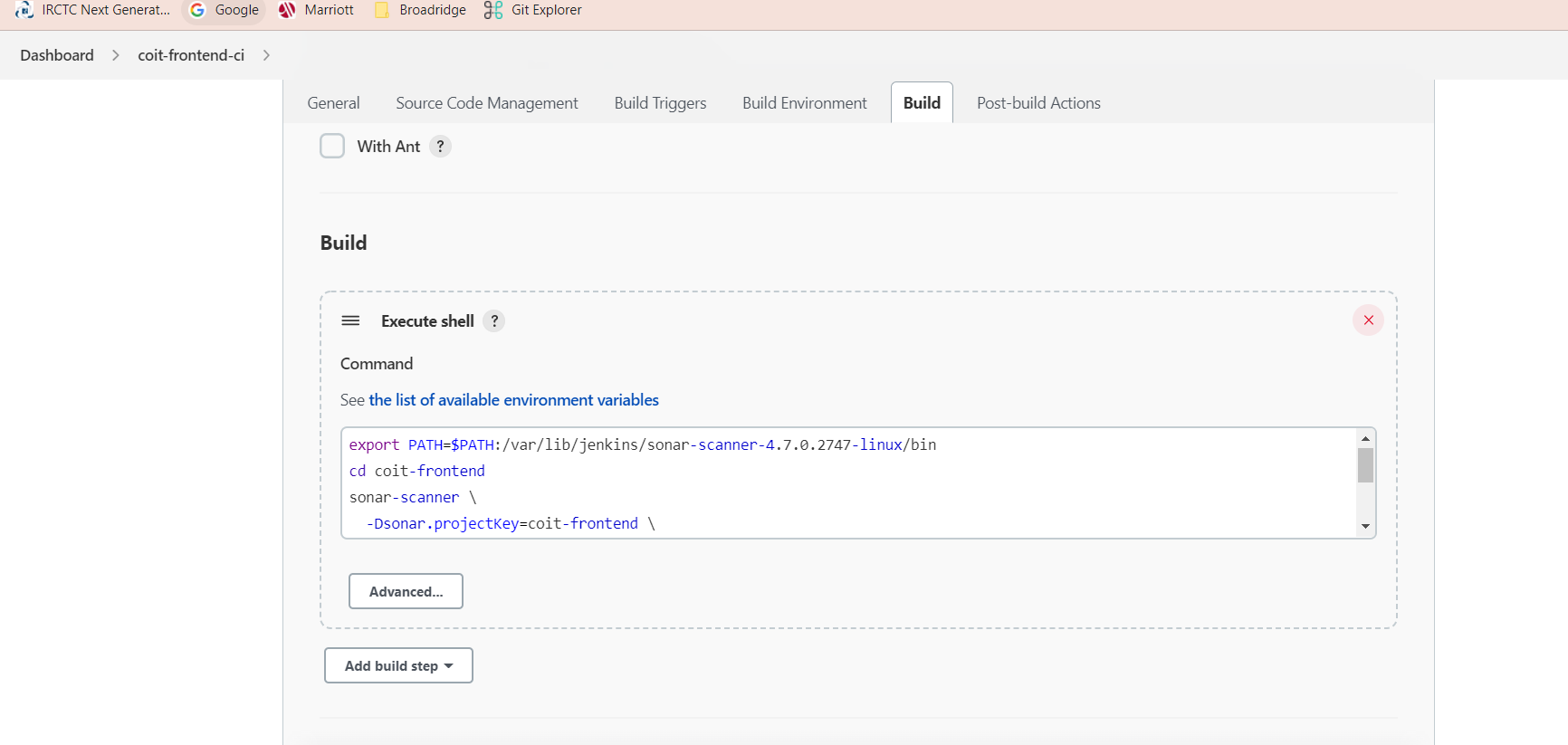
sonar-scanner \

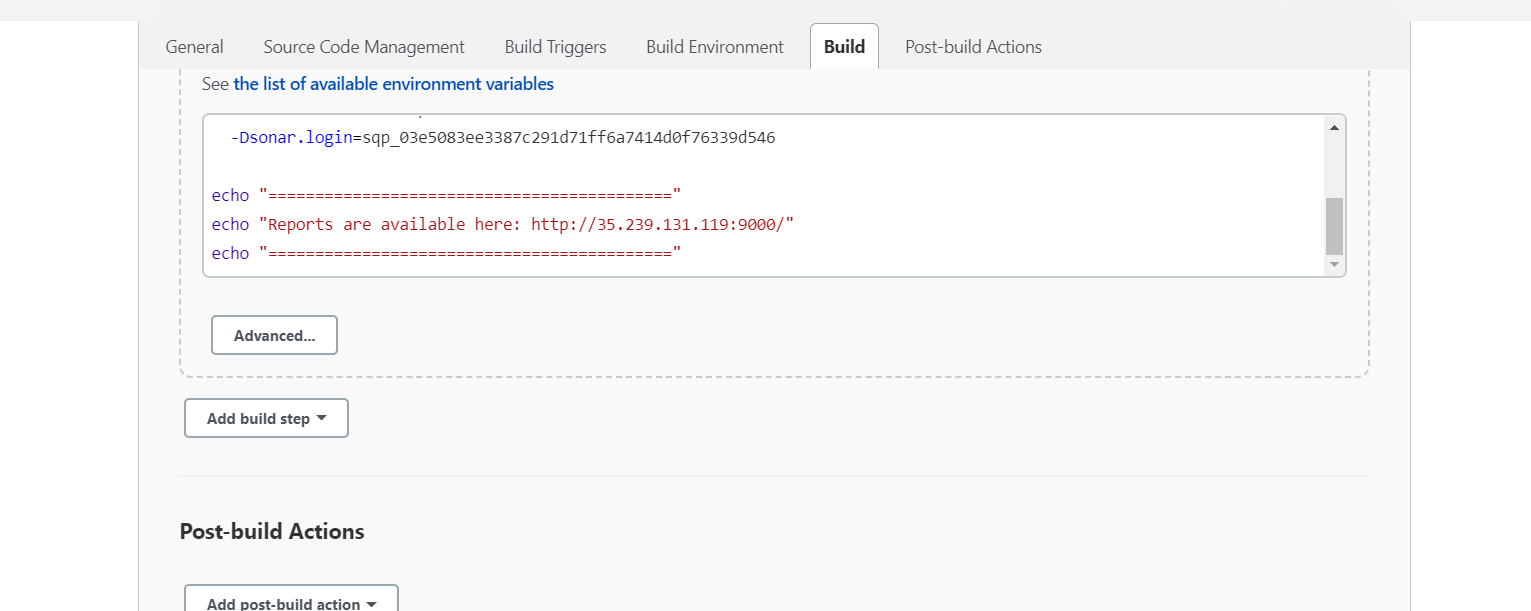
-Dsonar.projectKey=coit-frontend \

-Dsonar.sources=. \

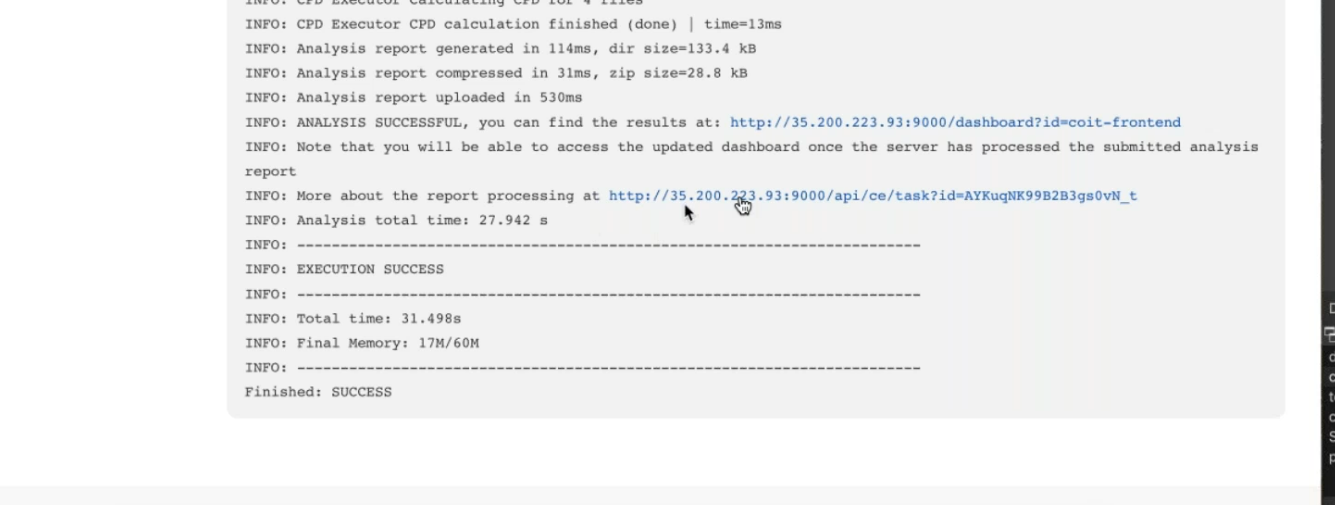
-Dsonar.host.url=http://35.239.131.119:9000 \

-Dsonar.login=sqp\_03e5083ee3387c291d71ff6a7414d0f76339d546





Save and run the job



Commands:

export PATH=$PATH:/var/lib/jenkins/sonar-scanner-4.7.0.2747-linux/bin

cd coit-frontend

sonar-scanner \

-Dsonar.projectKey=coit-frontend \

-Dsonar.sources=. \

-Dsonar.host.url=http://35.239.131.119:9000 \

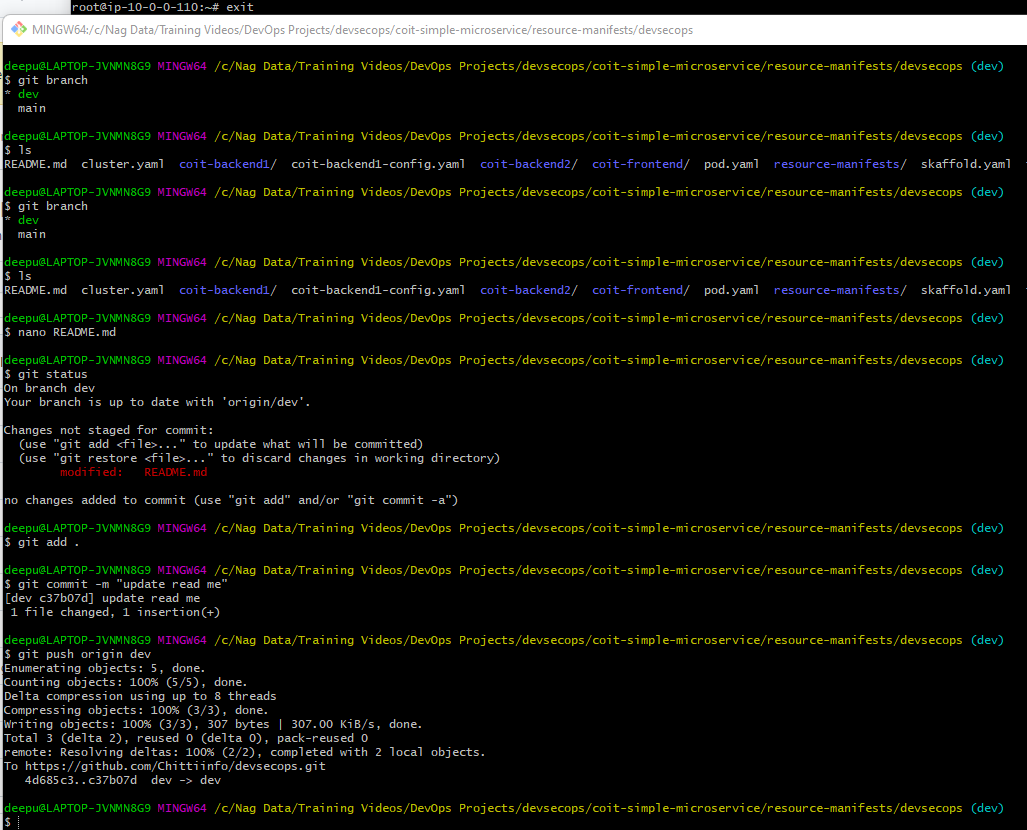
-Dsonar.login=sqp\_03e5083ee3387c291d71ff6a7414d0f76339d546

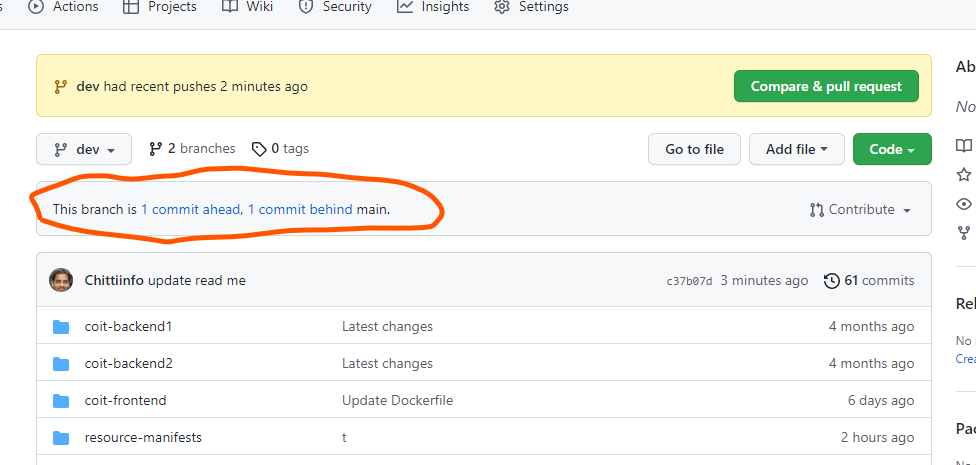
echo "==========================================="

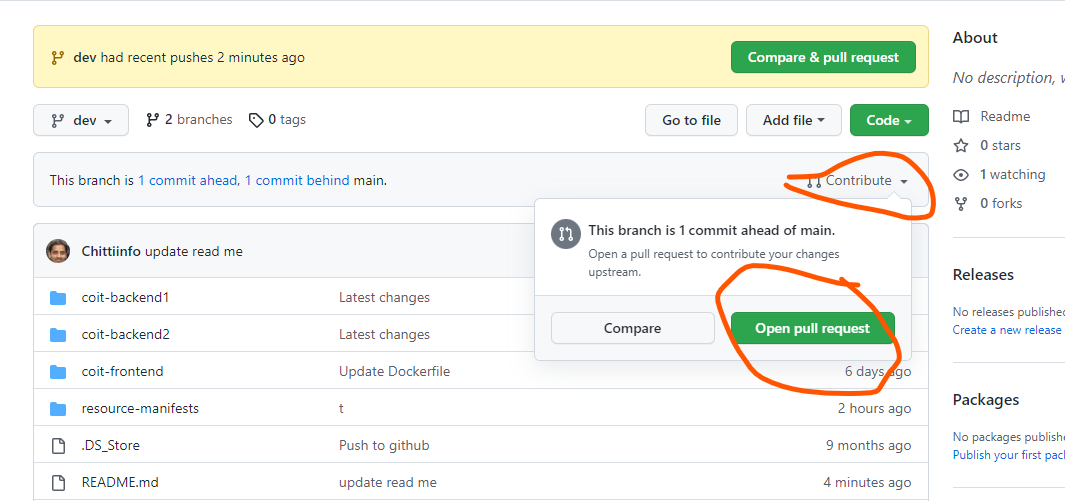
echo "Reports are available here: http://35.239.131.119:9000/"

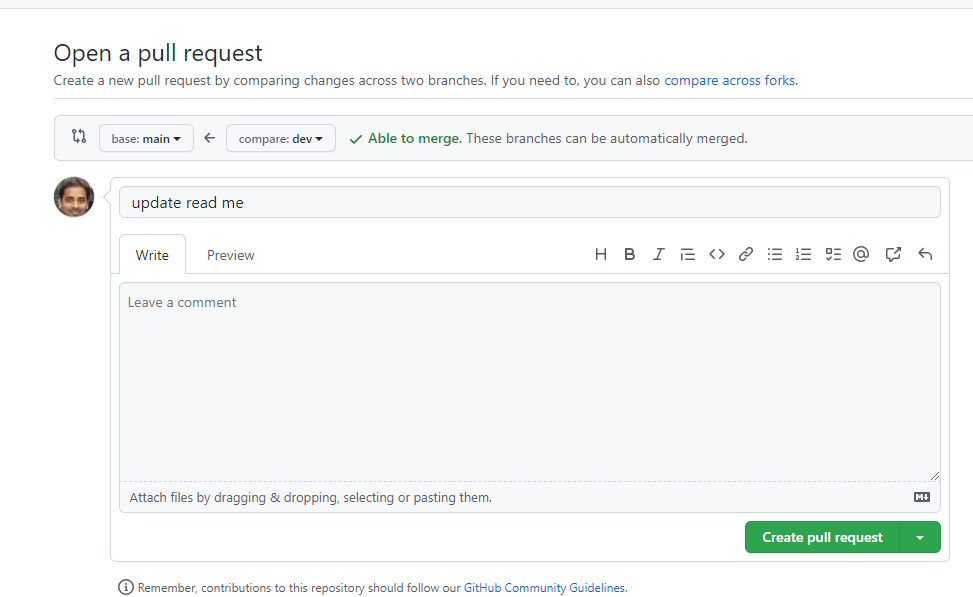
echo "==========================================="

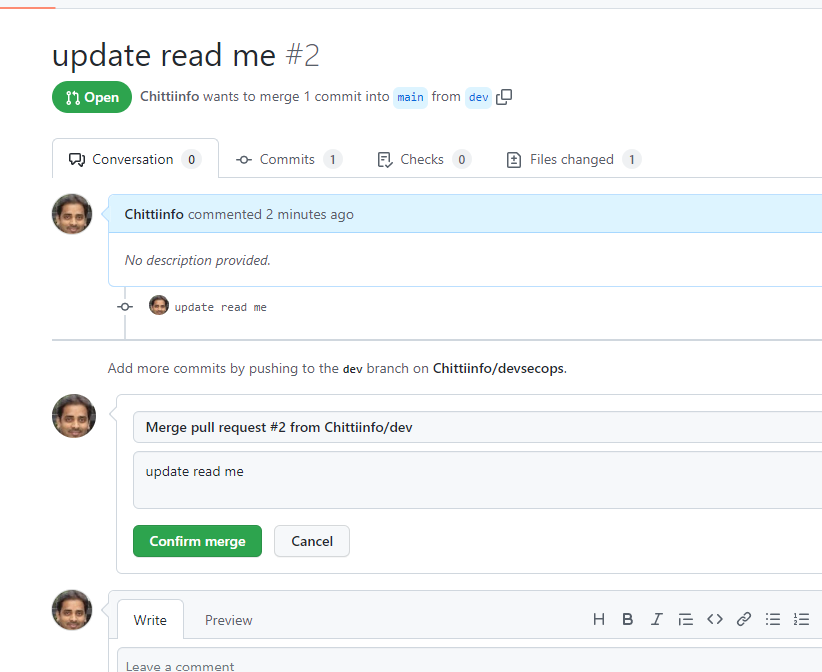
**Testing pull request CI Build:**





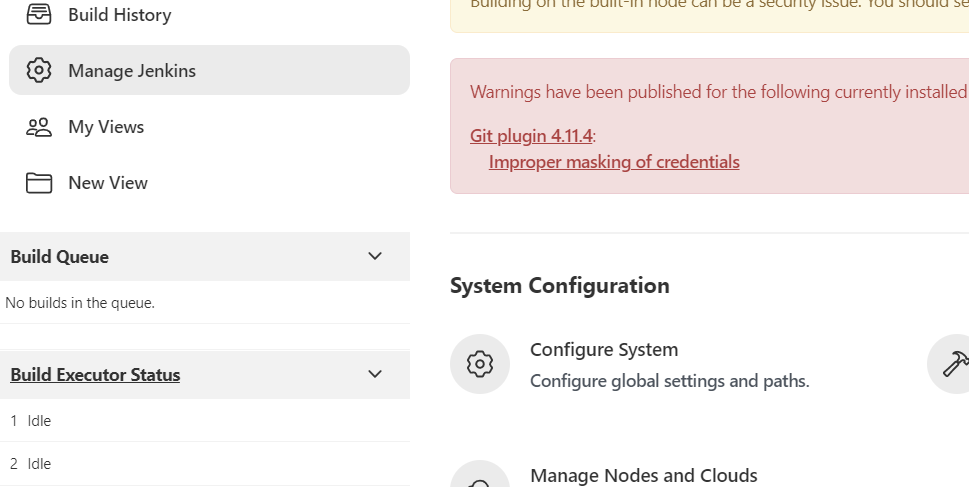


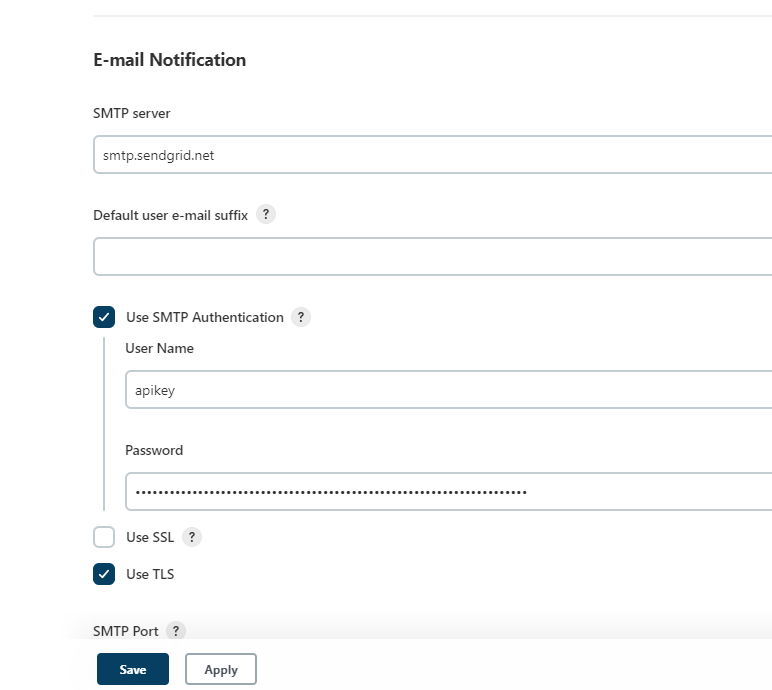




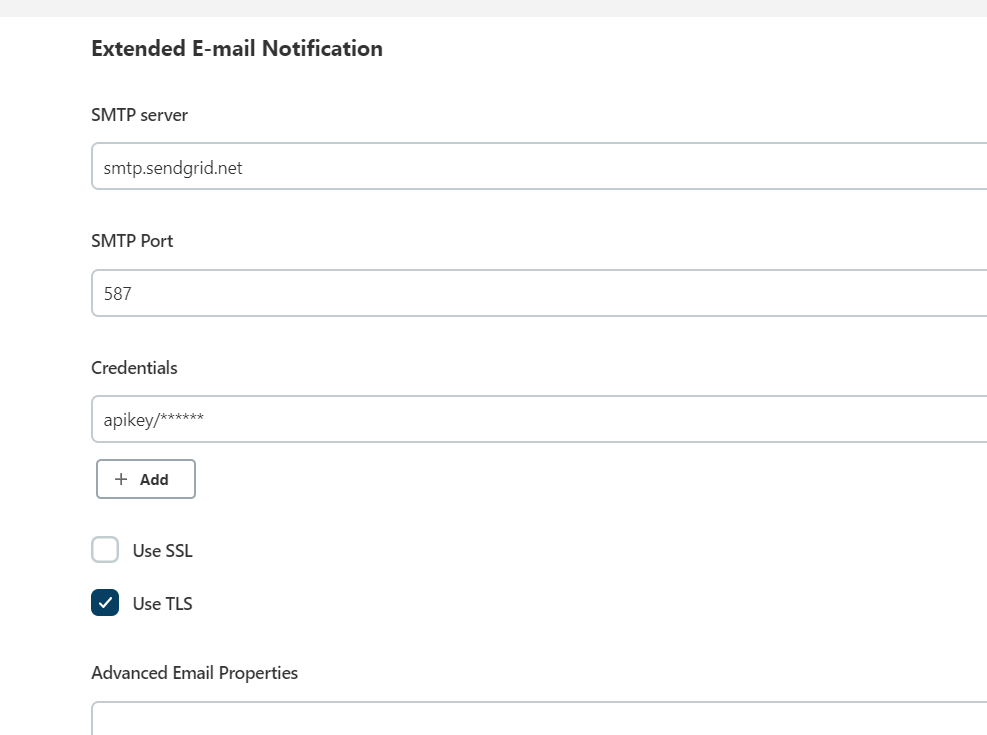
**E-mail Notification Setup:**

**Configure mail notifications from “configure system” option.**

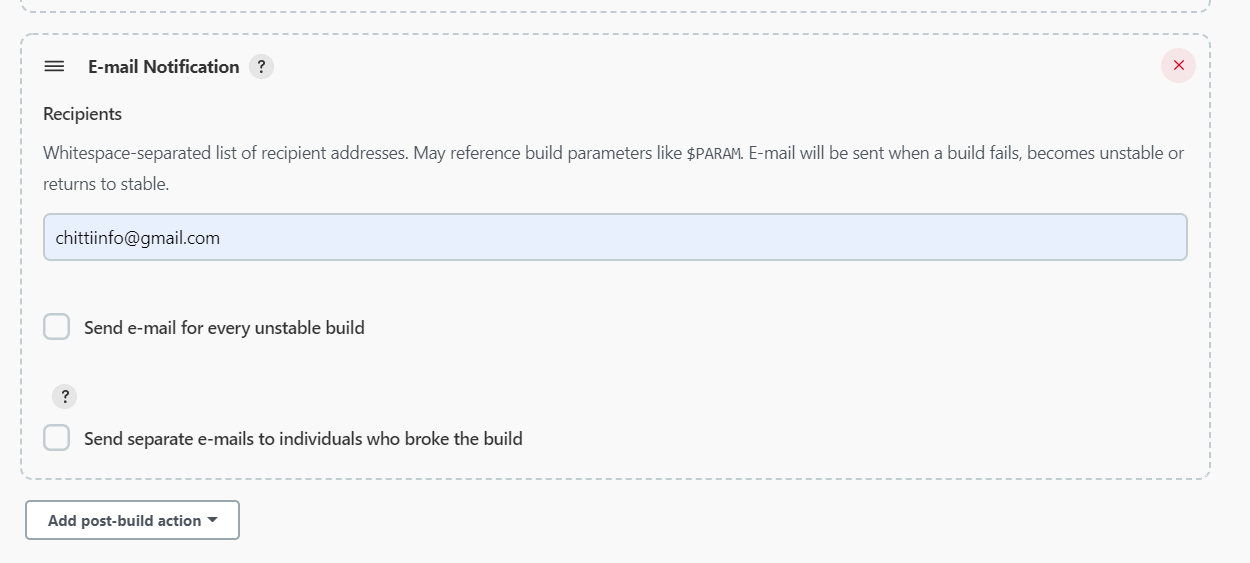
****



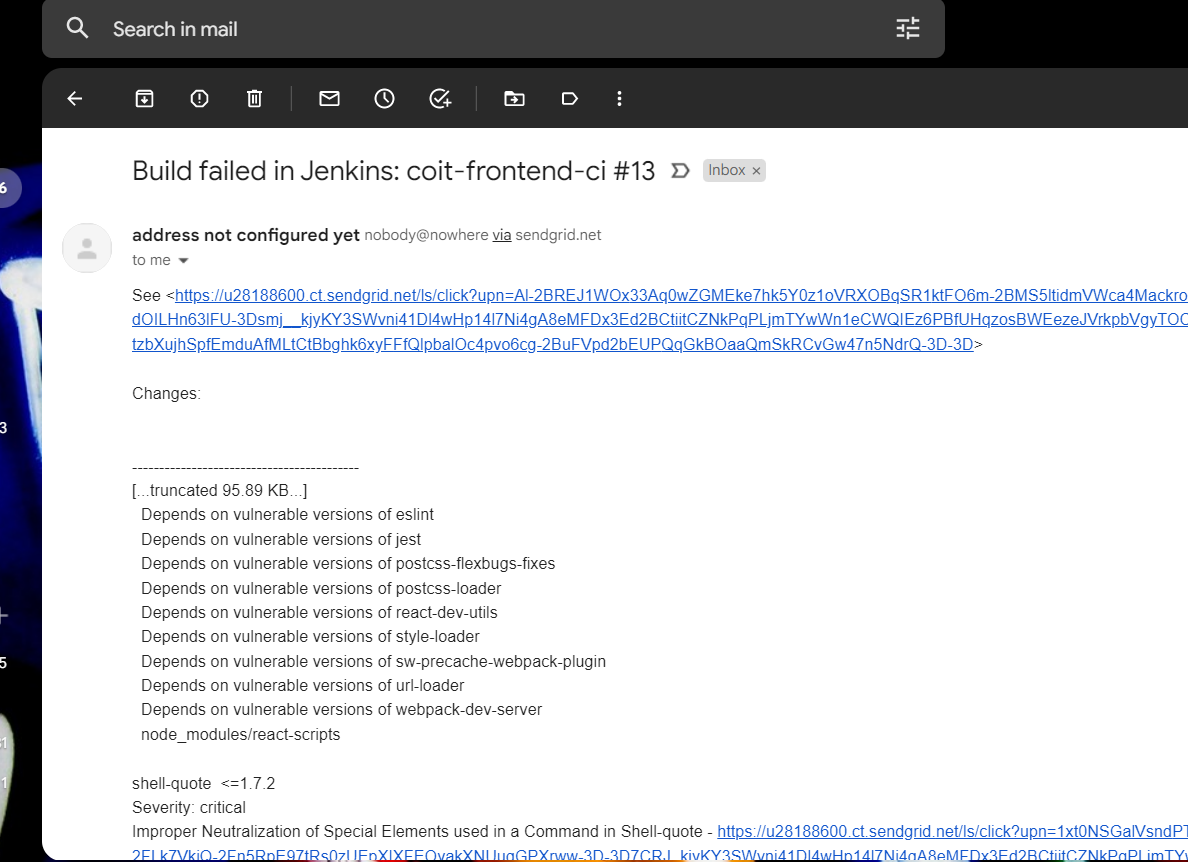




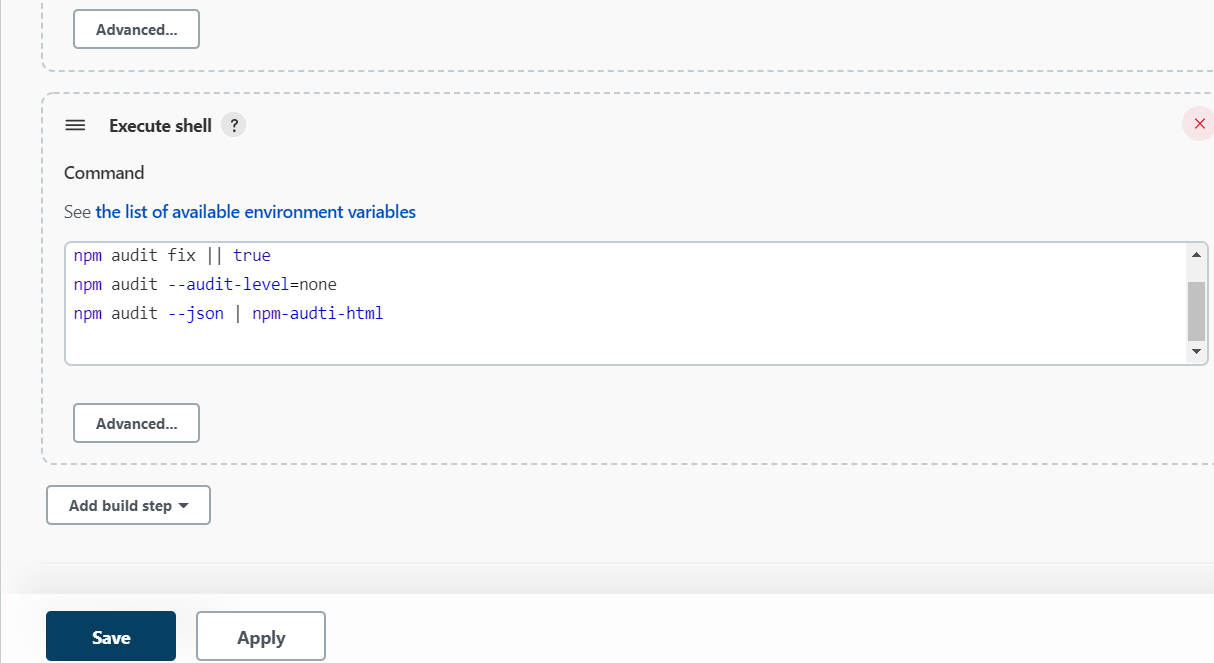
Add below post build action for mail notifications:



Received build fail notification on mail id:



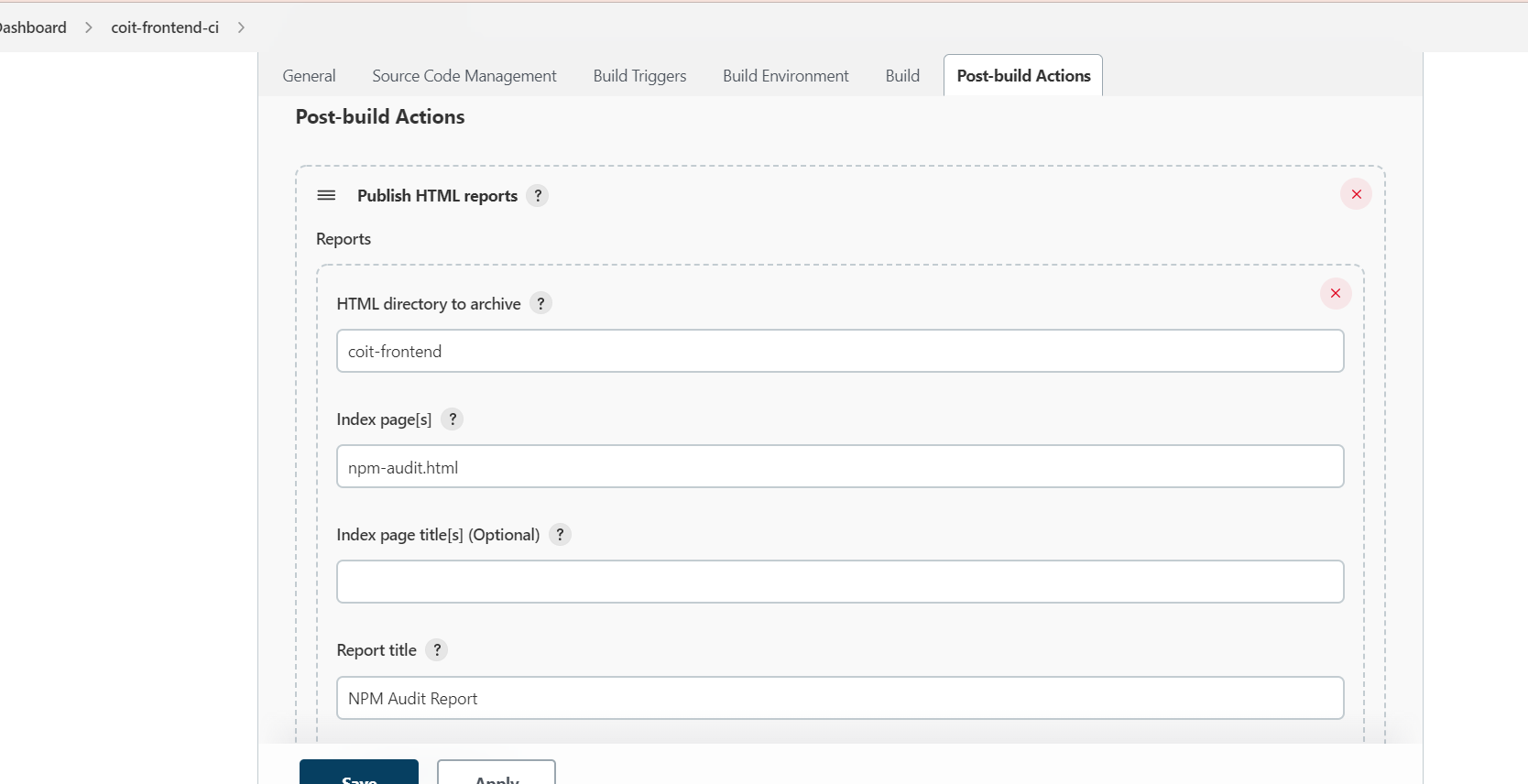
**Dependency scanning with NPM audit and fix:**



Install HTML Publisher plugin to view reports in html format

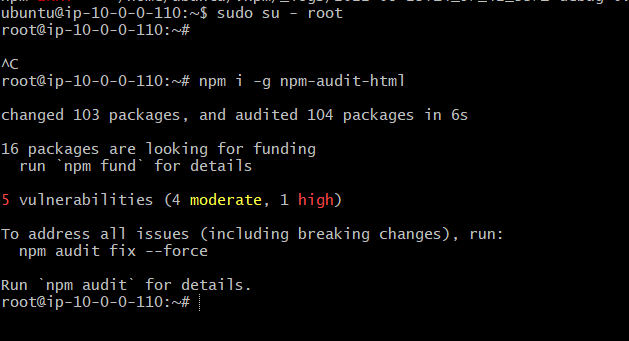


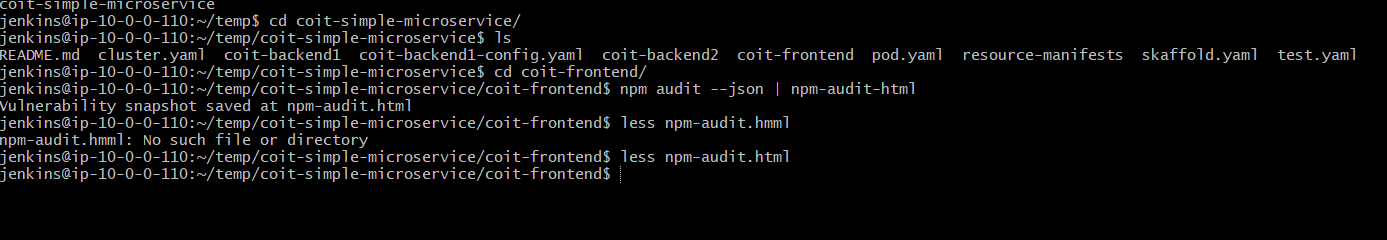
Add below post build actions once plugin is enabled to view reports in html format

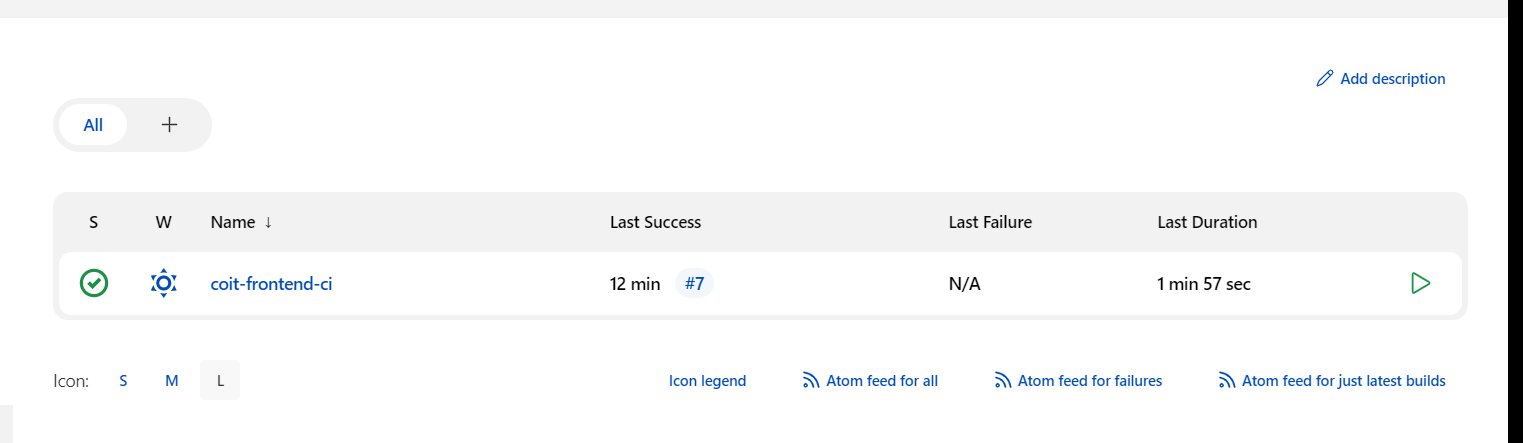


NOTE: Run below command with root user in jenkins

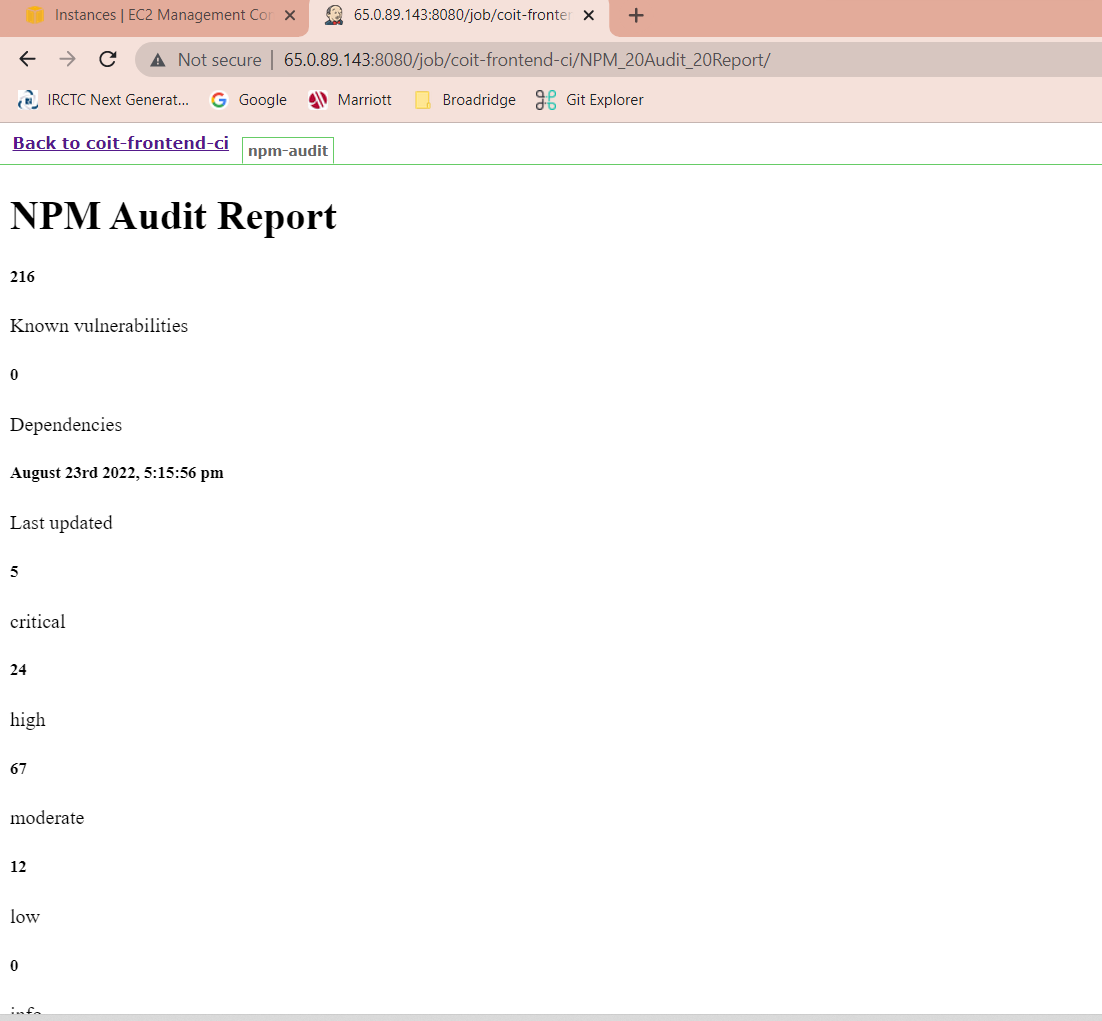
npm i -g npm-audit-html











Commands:

# Dependency vulnaribility check

echo "###################### Dependency vulneribility check #################"

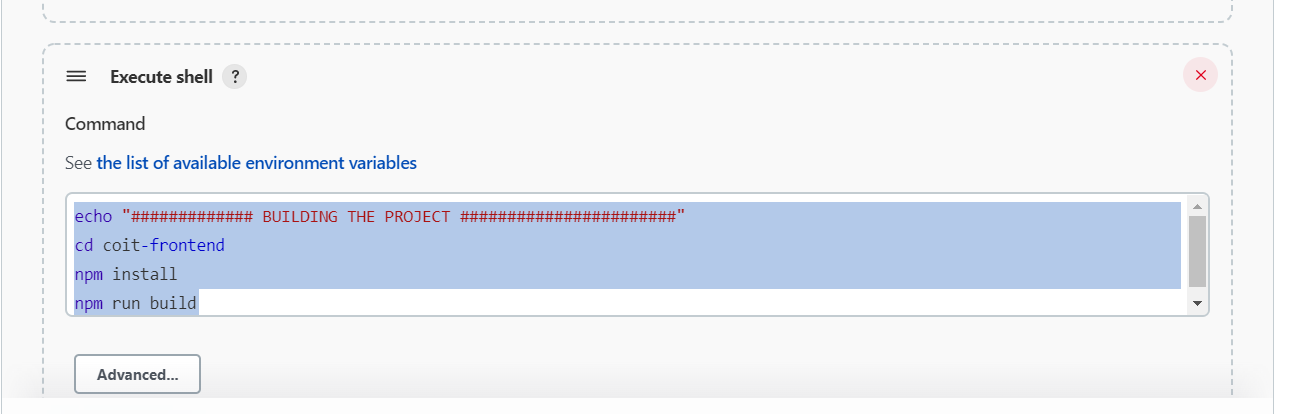
cd coit-frontend

npm audit fix || true

npm audit --audit-level=none

npm audit --json | npm-audit-html

**Build the code:**



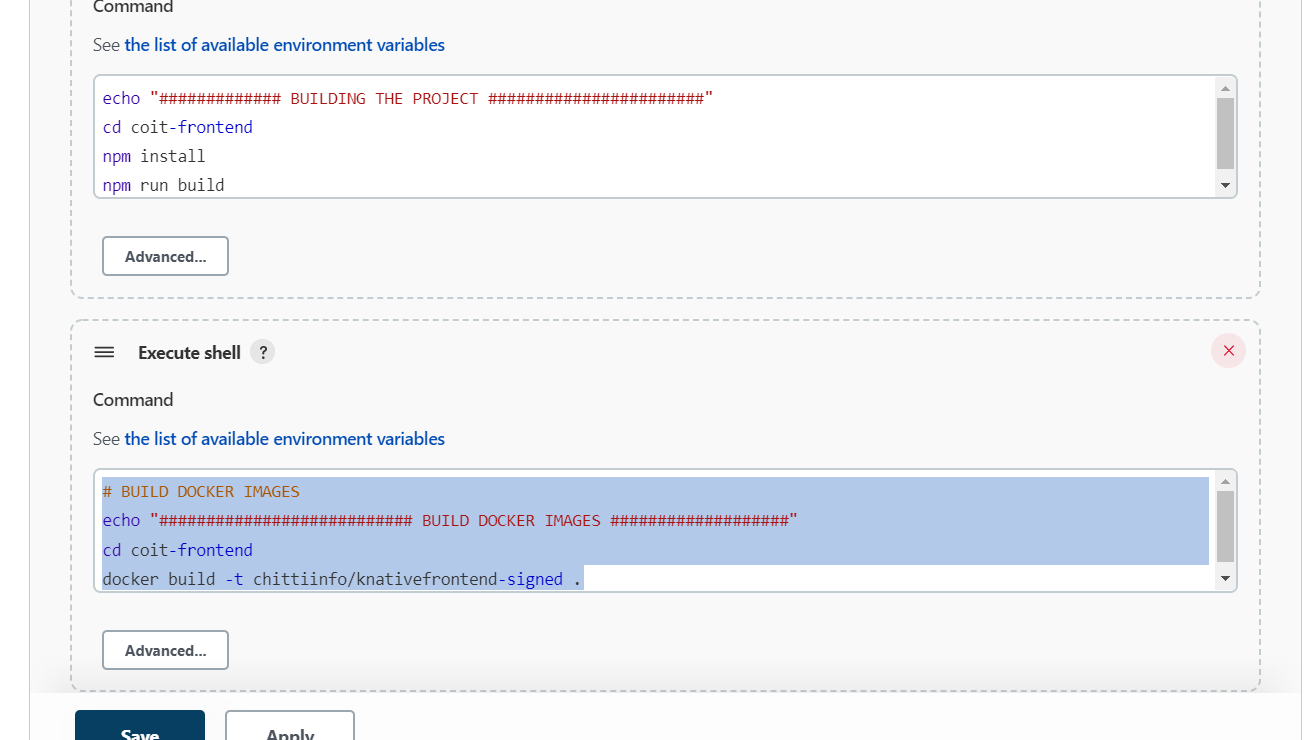
echo "############# BUILDING THE PROJECT #######################"

cd coit-frontend

npm install

npm run build

**Build the docker image:**



# BUILD DOCKER IMAGES

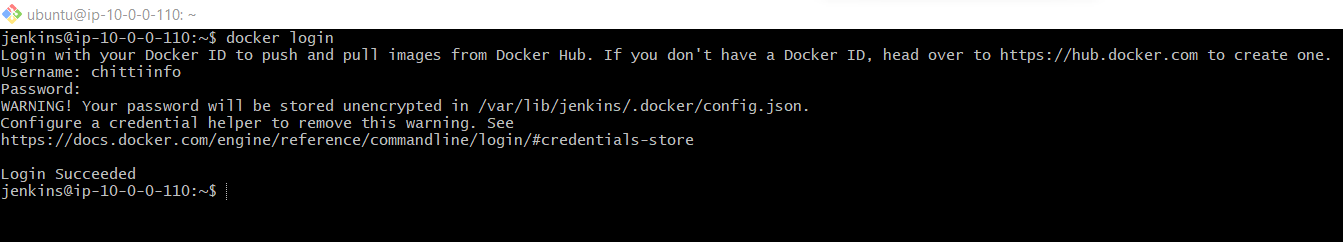
echo "########################### BUILD DOCKER IMAGES ###################"

cd coit-frontend

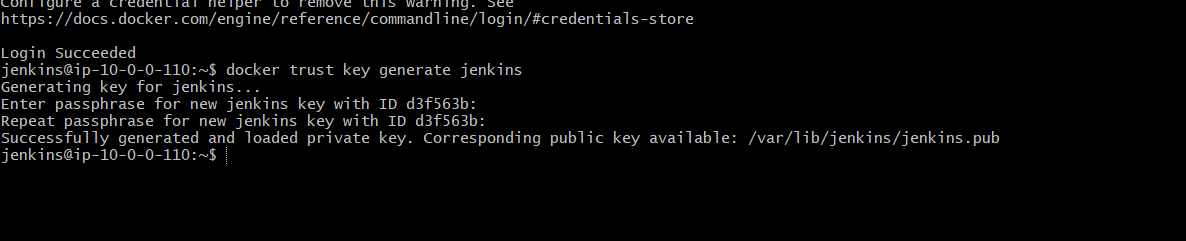
docker build -t chittiinfo/knativefrontend-signed .

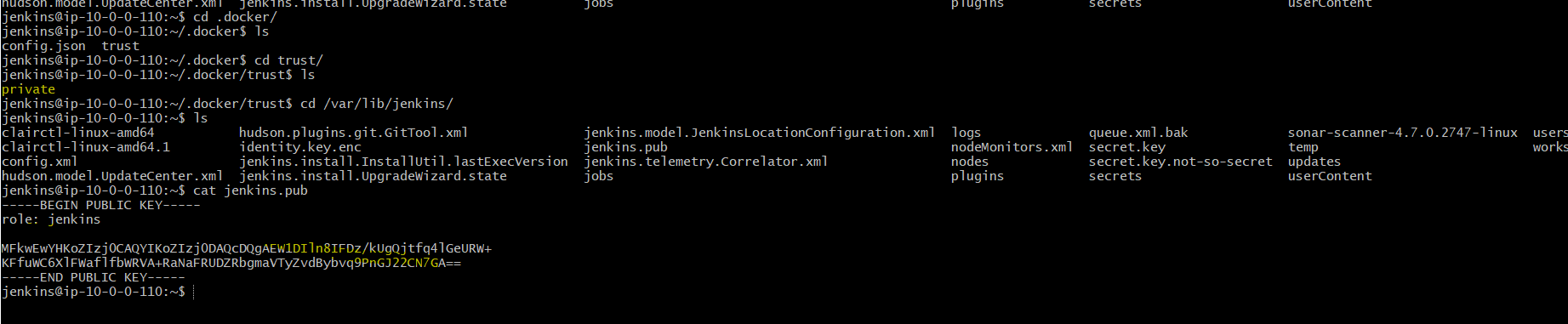
**Sign the images:**

**Login to Jenkins servers and connect to docker**

****

docker trust key generate Jenkins ( To generate trust keys for Jenkins user)





Add user to docker registry:

docker trust signer add jenkins --key jenkins.pub chittiinfo/devsecops

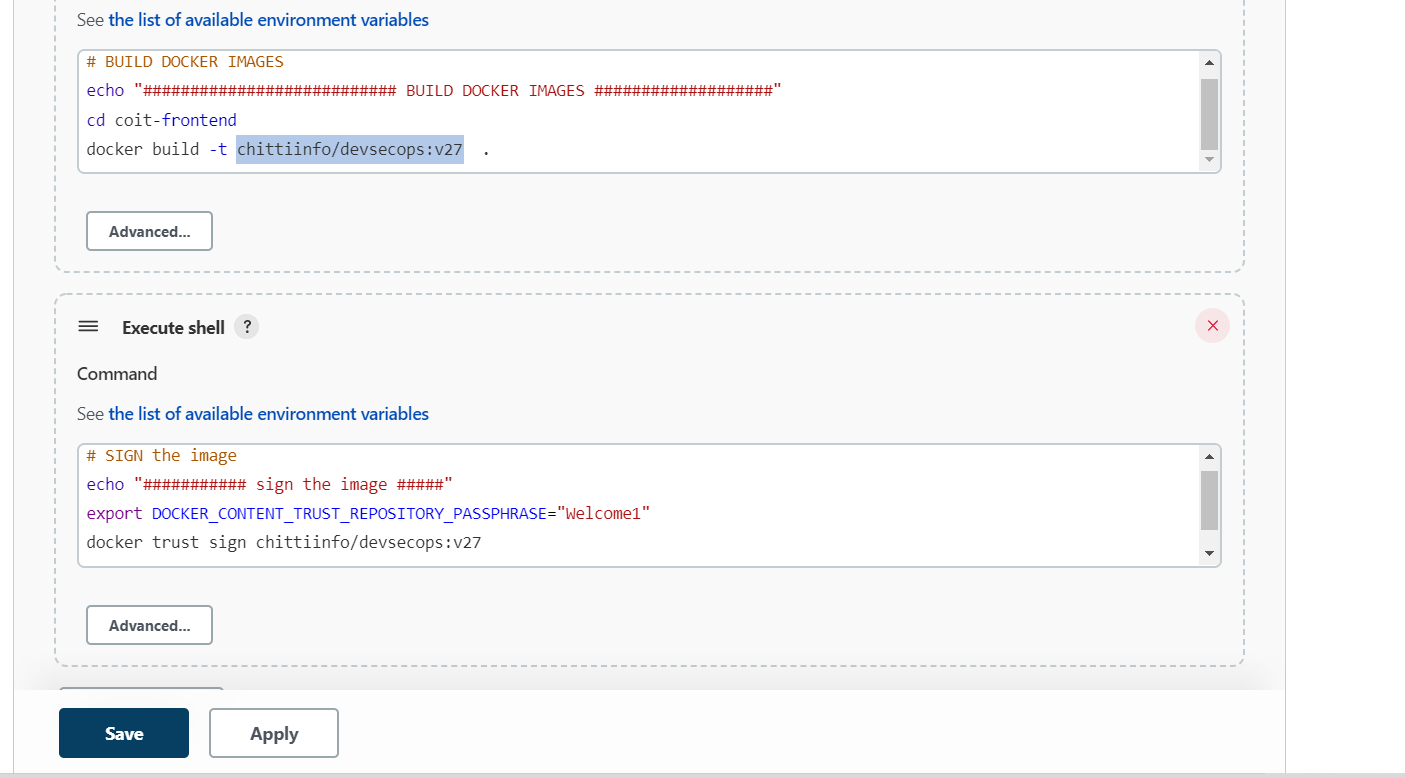


# SIGN the image

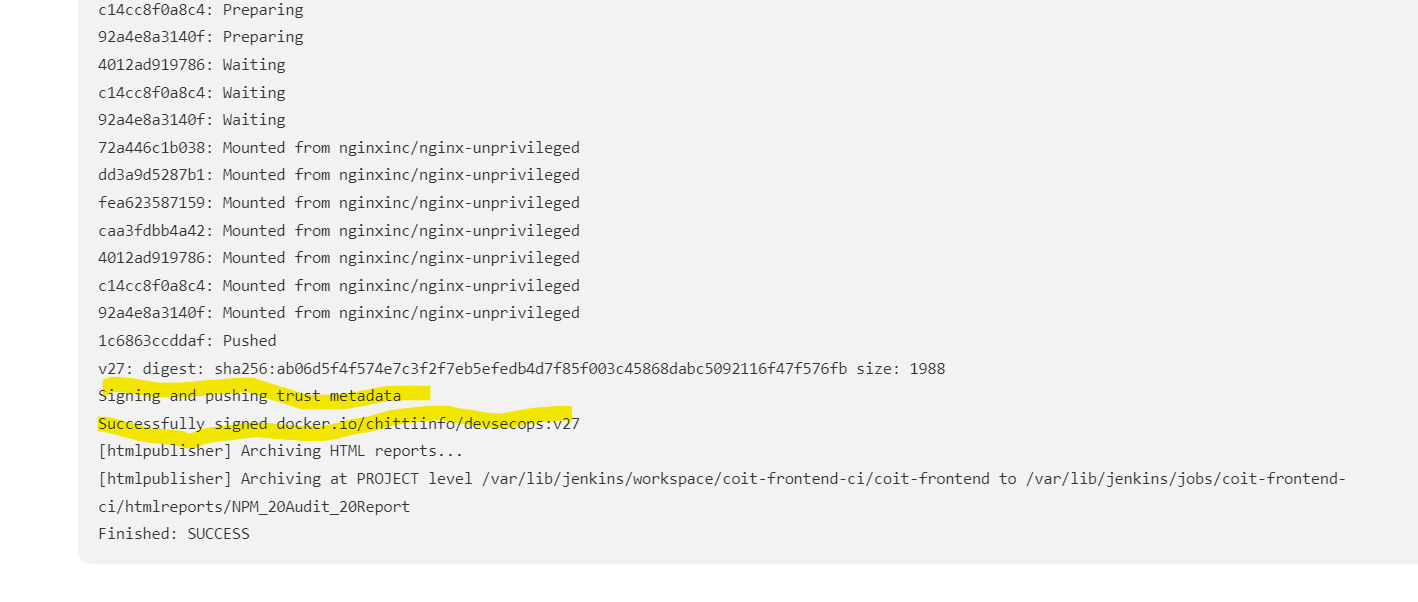
echo "########### sign the image #####"

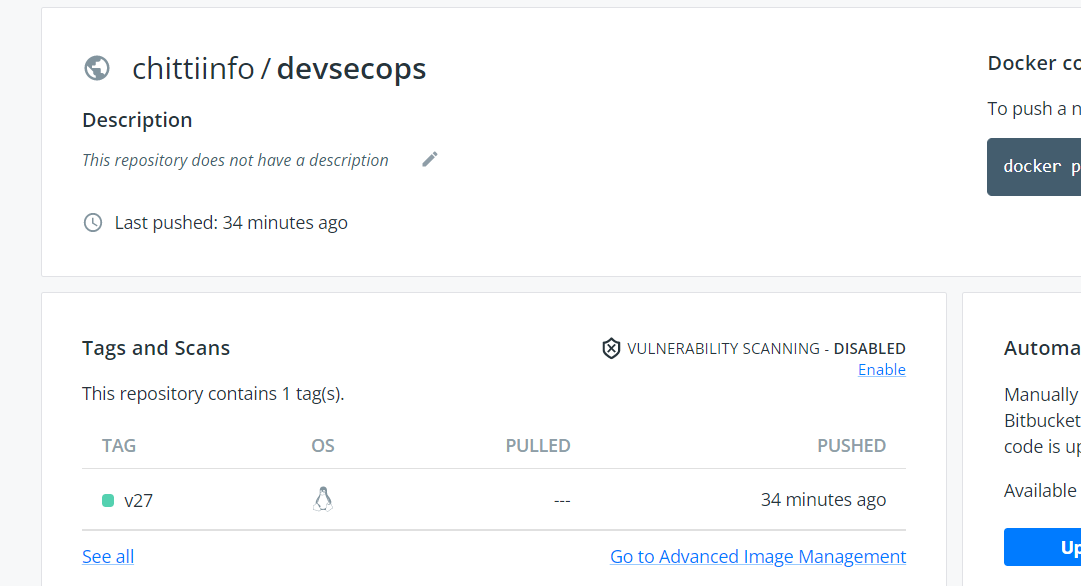
export DOCKER\_CONTENT\_TRUST\_REPOSITORY\_PASSPHRASE="Welcome1"

docker trust sign chittiinfo/devsecops:v27



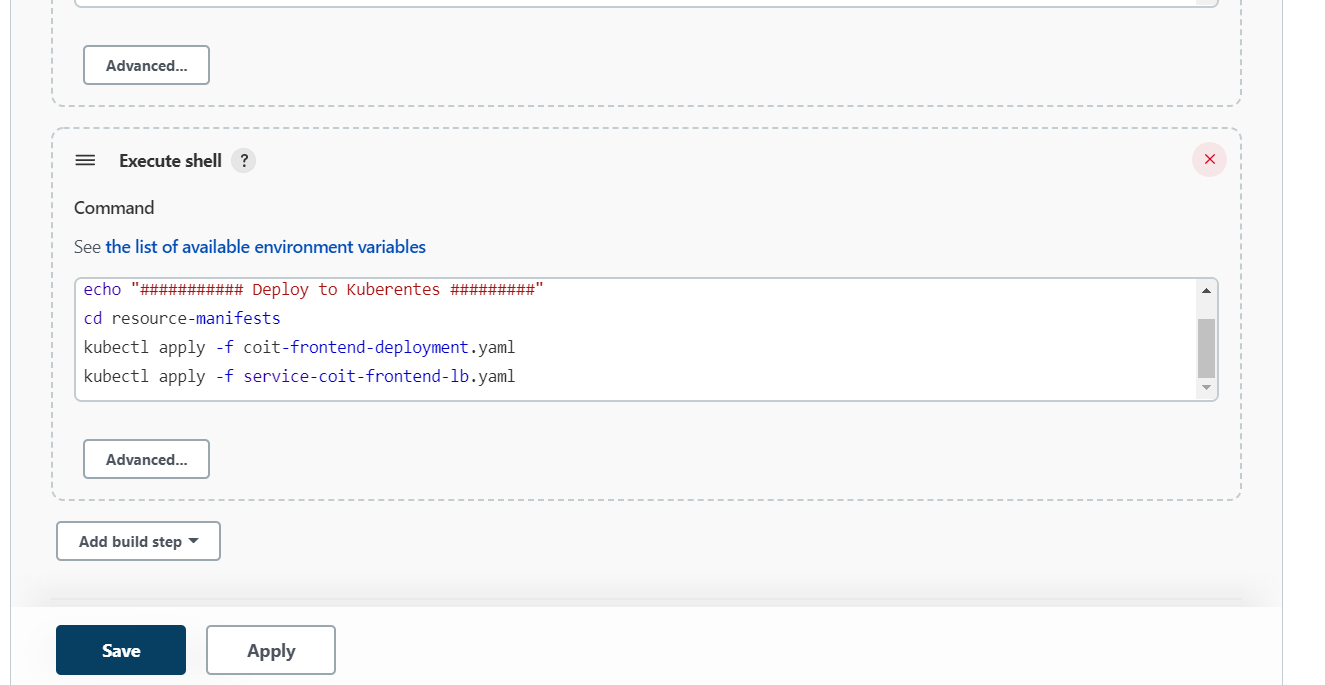






**Scanning images with clair or SNYK ( Skipping this step for now)**

**Deploy application to Kubernetes:**



# Deploy to KUbernetes

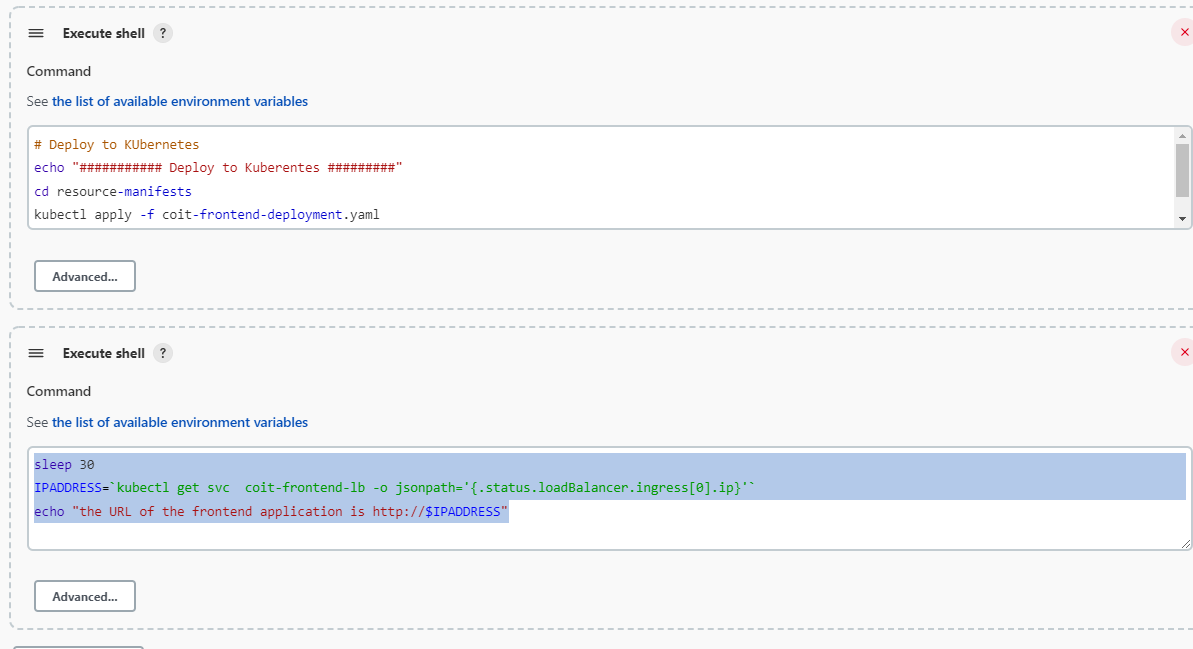
echo "########### Deploy to Kuberentes #########"

cd resource-manifests

kubectl apply -f coit-frontend-deployment.yaml

kubectl apply -f service-coit-frontend-lb.yaml

**show the service IP to developers**



sleep 30

IPADDRESS=`kubectl get svc coit-frontend-lb -o jsonpath='{.status.loadBalancer.ingress[0].ip}'`

echo "the URL of the frontend application is <http://$IPADDRESS>

Complete Pipelines screenshots:

