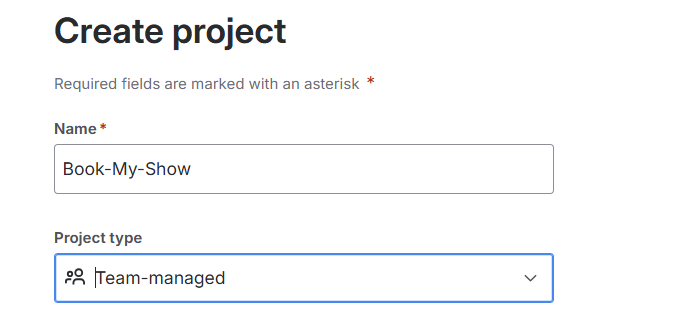
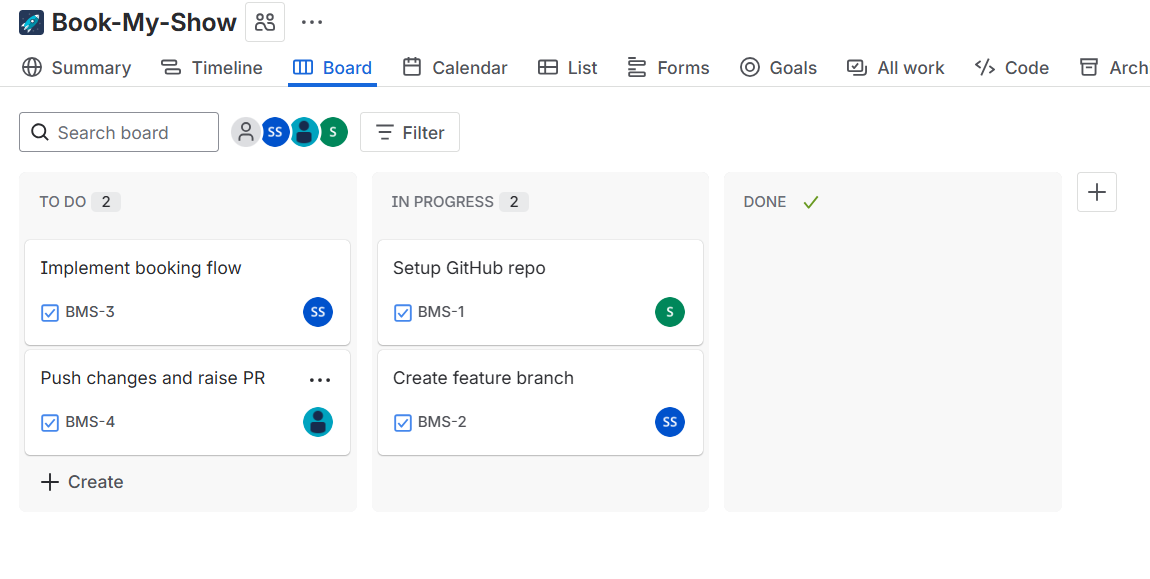
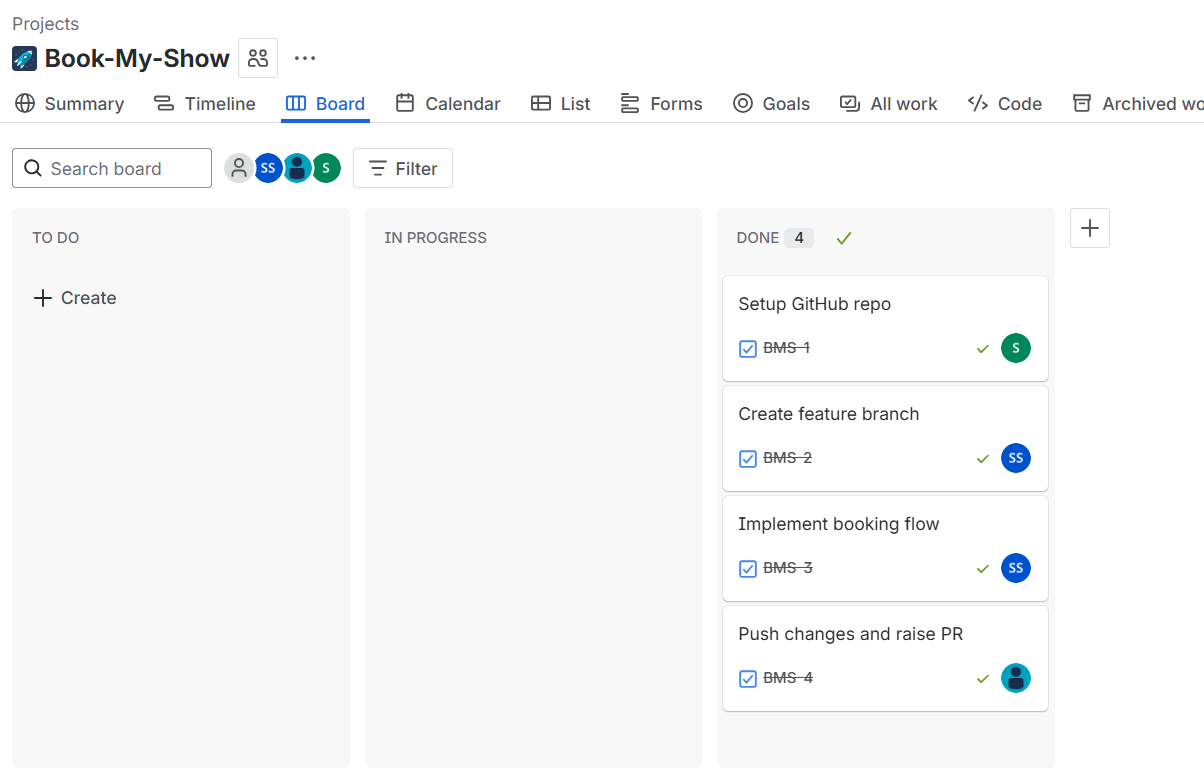
Go to jira create a team managed kanban project:







GITHUB PR:

Go to the repo: [Book-My-Show](https://github.com/akshu20791/Book-My-Show).

Click **Fork** (top right) → this will create a copy in your GitHub account (syaddays/Book-My-Show).

Clone your fork instead:  
  
 git clone https://github.com/syaddays/Book-My-Show.git

cd Book-My-Show

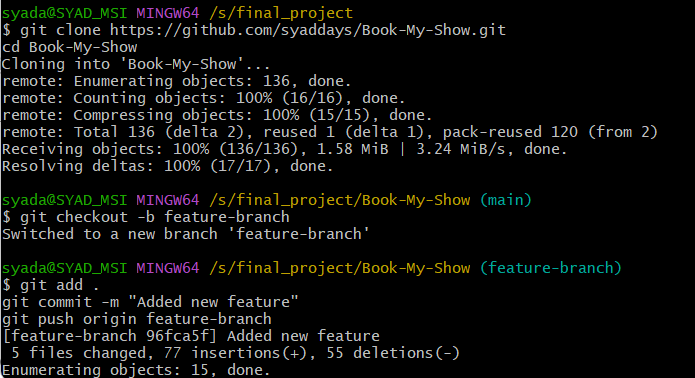
Create your feature branch, make changes, commit.  
  
 git checkout -b feature-branch

git add .

git commit -m "Added new feature"

git push origin feature-branch

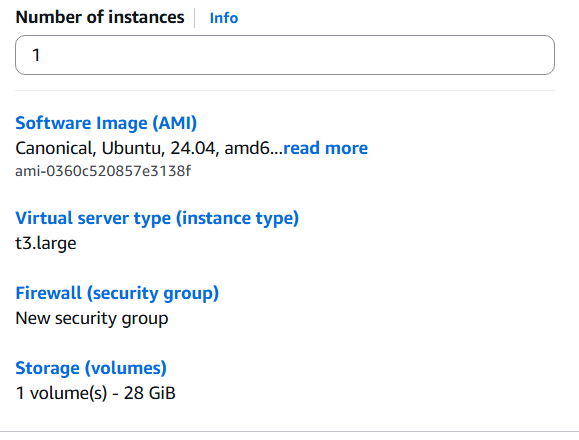
Open your fork on GitHub → click **“Compare & pull request”** to submit a PR **to the original repo (akshu20791/Book-My-Show)**.





[Added new feature by syaddays · Pull Request #76 · akshu20791/Book-My-Show](https://github.com/akshu20791/Book-My-Show/pull/76)

Create an ec2 machine t3 large 28 gb ,ubuntu



Connect to the machine

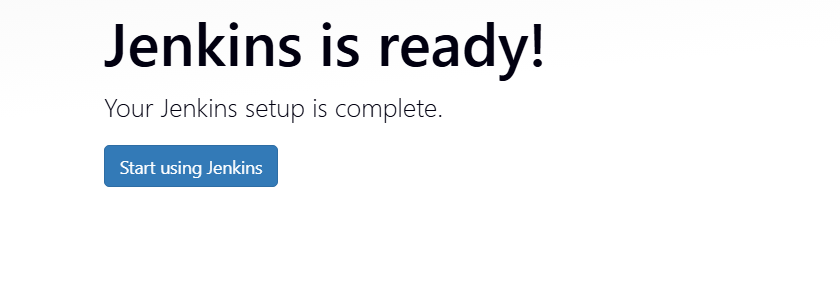
Instal jenkins:

wget https://raw.githubusercontent.com/akshu20791/Deployment-script/main/jenkins.sh

ls

chmod +x [jenkins.sh](http://jenkins.sh)

./jenkins.sh



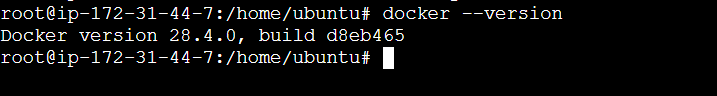
Install docker:

1.sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installDocker.sh -P /tmp

2.sudo chmod 755 /tmp/installDocker.sh

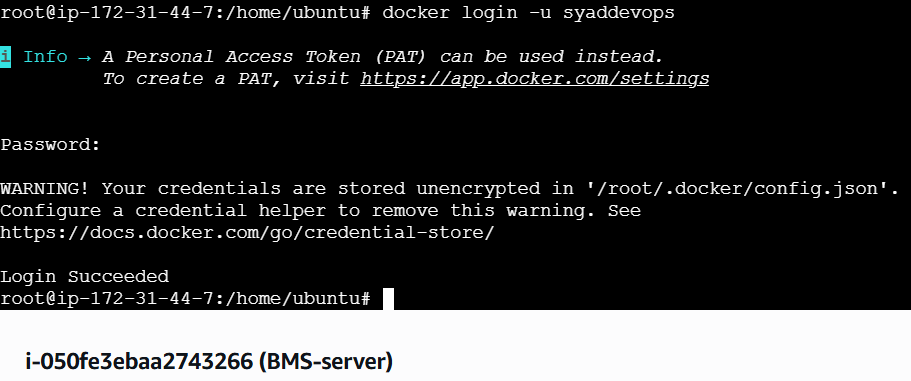
3.sudo bash /tmp/installDocker.sh

4.sudo systemctl restart docker.service



**LOGIN TO DOCKERHUB ID:**

docker login -u <DockerHubUserName>



**INSTALL TRIVY:**

1.vi [trivy.sh](http://trivy.sh)

#!/bin/bash

sudo apt-get install wget apt-transport-https gnupg

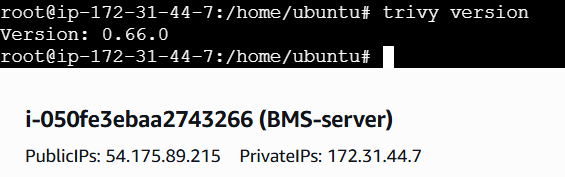
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/nullecho "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb generic main" | sudo tee -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy

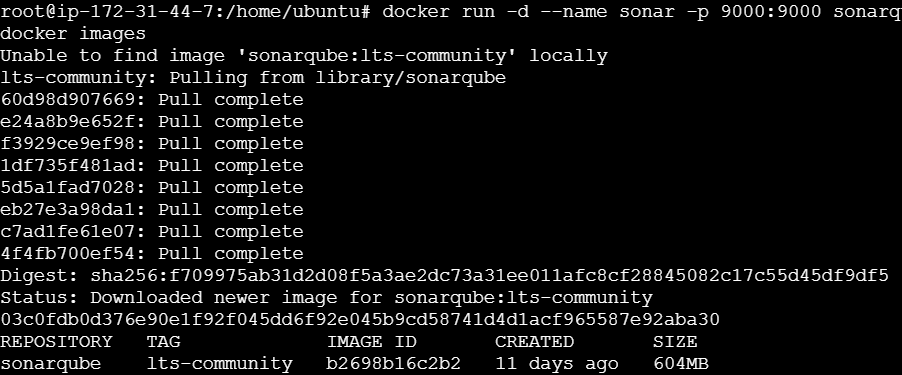
2. chmod +x [trivy.sh](http://trivy.sh)

3. ./[trivy.sh](http://trivy.sh)

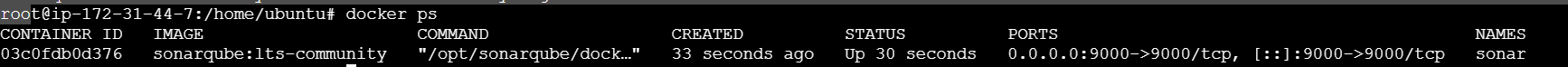


SETUP SONARQUBE:

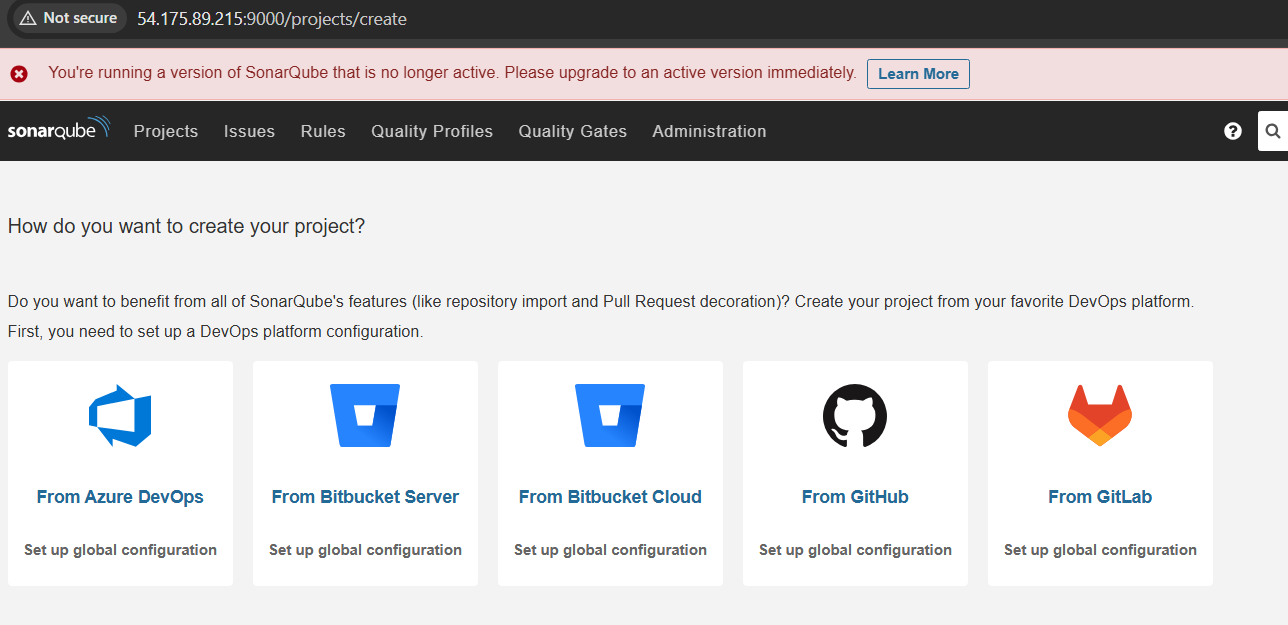
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community  
docker images



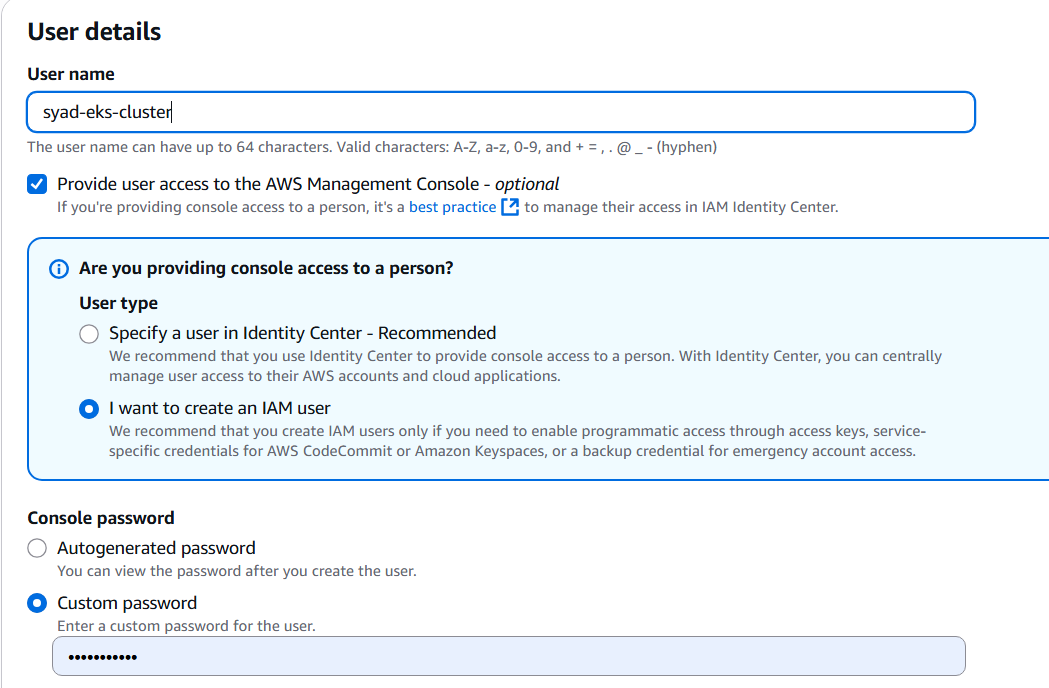
docker ps



Copy the ip address port :9000  
default user,password is “admin”

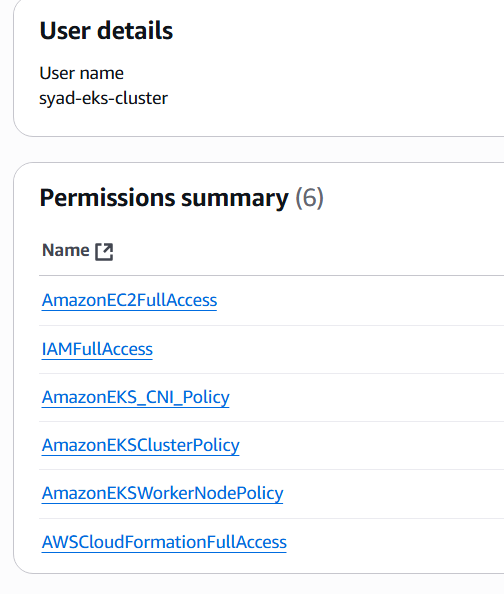


go to IAM console create a user   
username- syad-eks-cluster (Give your name)  
check access console   
check I want to create an IAM user  
  
custom password : give your own password   
  
uncheck users must create a new password at next sign-in



click on next , donot select any policies as of now click on next and create user

Click on users under permissions select add permissions   
add permissions (in dropdown)  
  
Attach policies directly



now we have to create one more policy which is inline policy   
again click on add permissions under dropdown select “create inline policy”  
  
Select JSON delete the script and copy paste below line  
{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

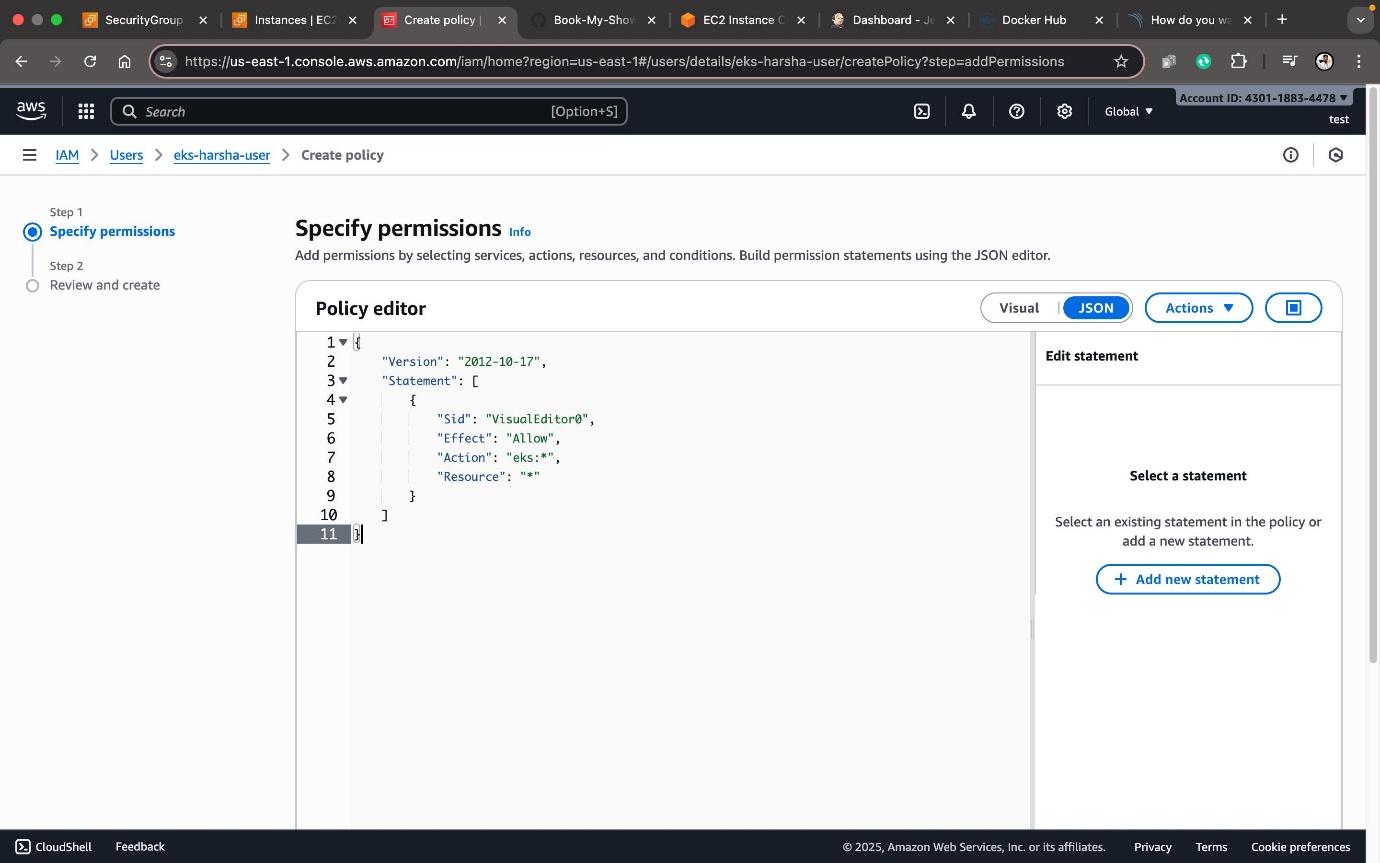
"Effect": "Allow",

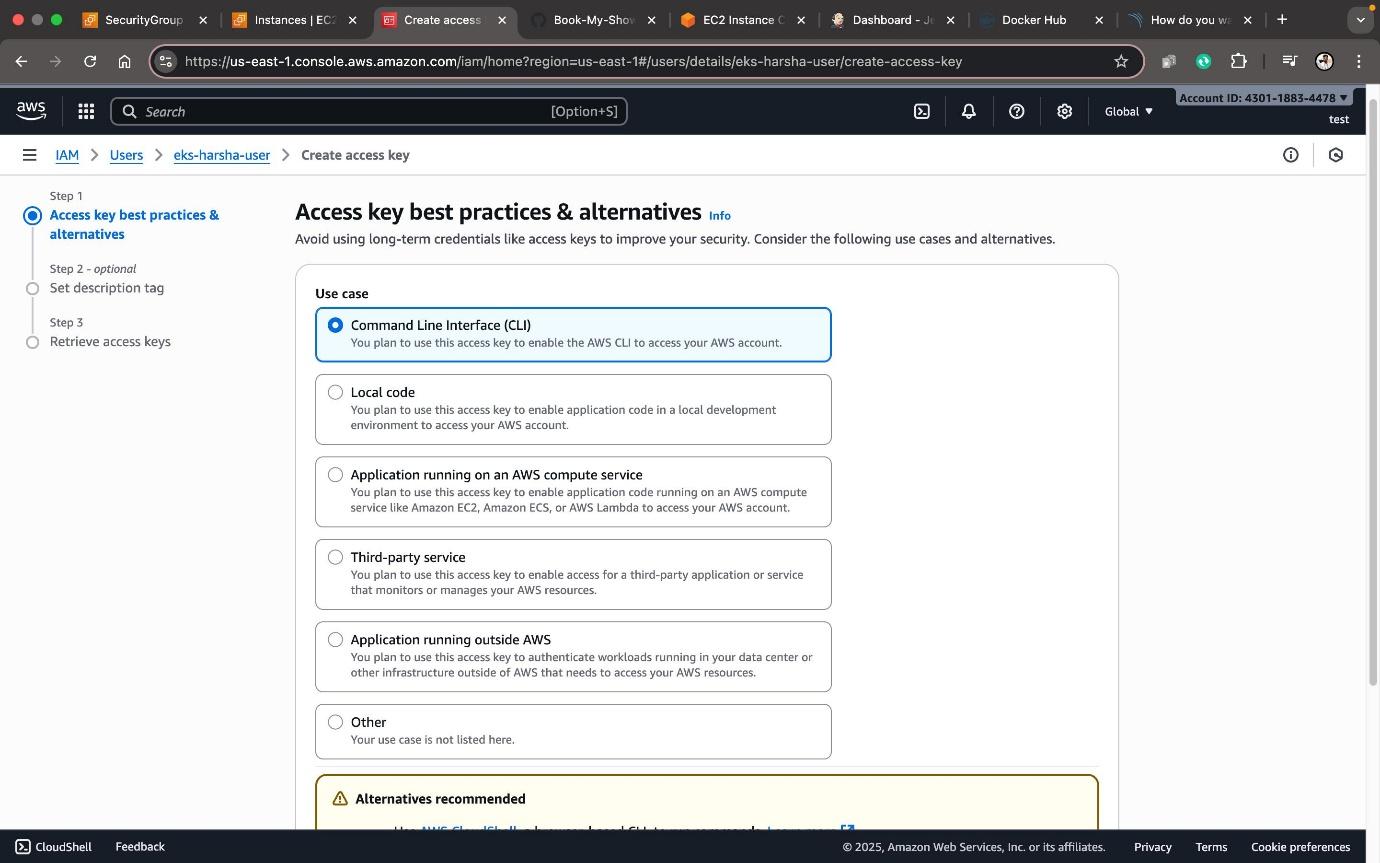
"Action": "eks:\*",

"Resource": "\*"

}

] }



We need to create access key   
Go to Security credentials in eks-syad-user  
  
scroll down and create access key   
select Command Line Interface(CLI)  


Download .csv file.

INSTALL AWS CLI:

1. vi[**aws.sh**](http://aws.sh)

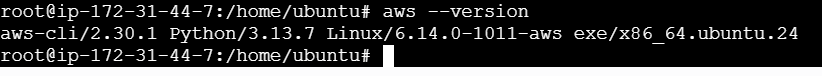
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

sudo apt install unzip

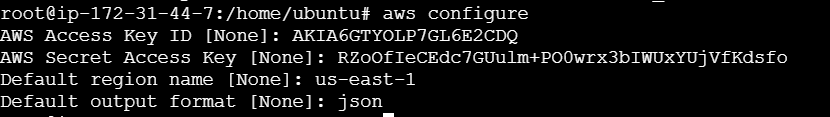
unzip awscliv2.zip

sudo ./aws/install

1. sudo chmod +x [aws.sh](http://aws.sh)
2. ./[aws.sh](http://aws.sh)
3. aws –version



1. Aws configure: enter your access and secret key, region and format as json.



INSTALL K8S:

1. vi [kubectl.sh](http://kubectl.sh)

curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin

kubectl version --short –client

1. sudo chmod +x [kubectl.sh](http://kubectl.sh)
2. ./kubectl.sh

INSTALL EKSCTL:

1. vi eksctl.sh

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

1. sudo chmod +x [eksctl.sh](http://eksctl.sh)
2. ./eksctl.sh



CREATE CLUSTER:

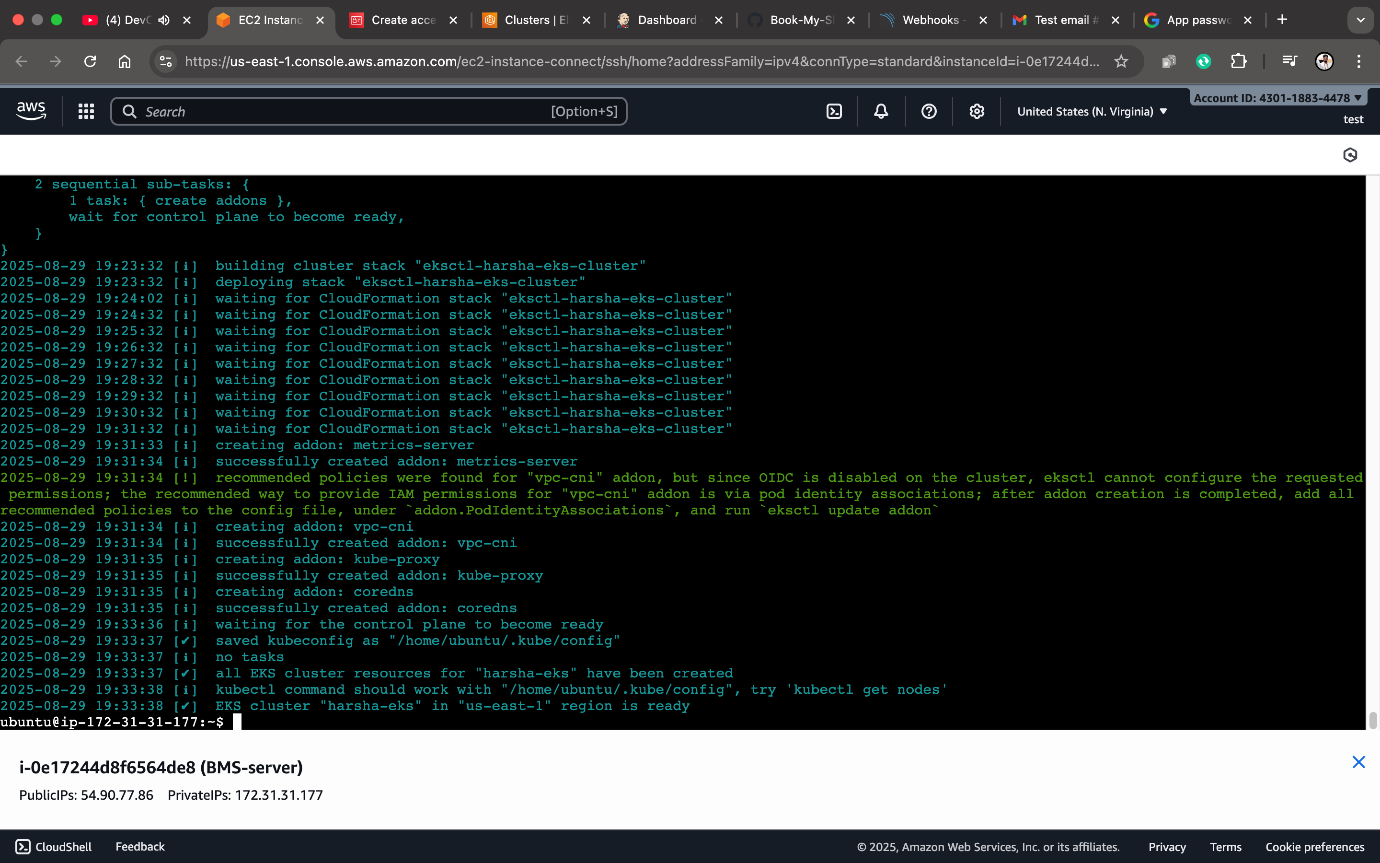
eksctl create cluster --name=Syad-eks \

--region=us-east-1 \

--zones=us-east-1a,us-east-1b \

--version=1.30 \

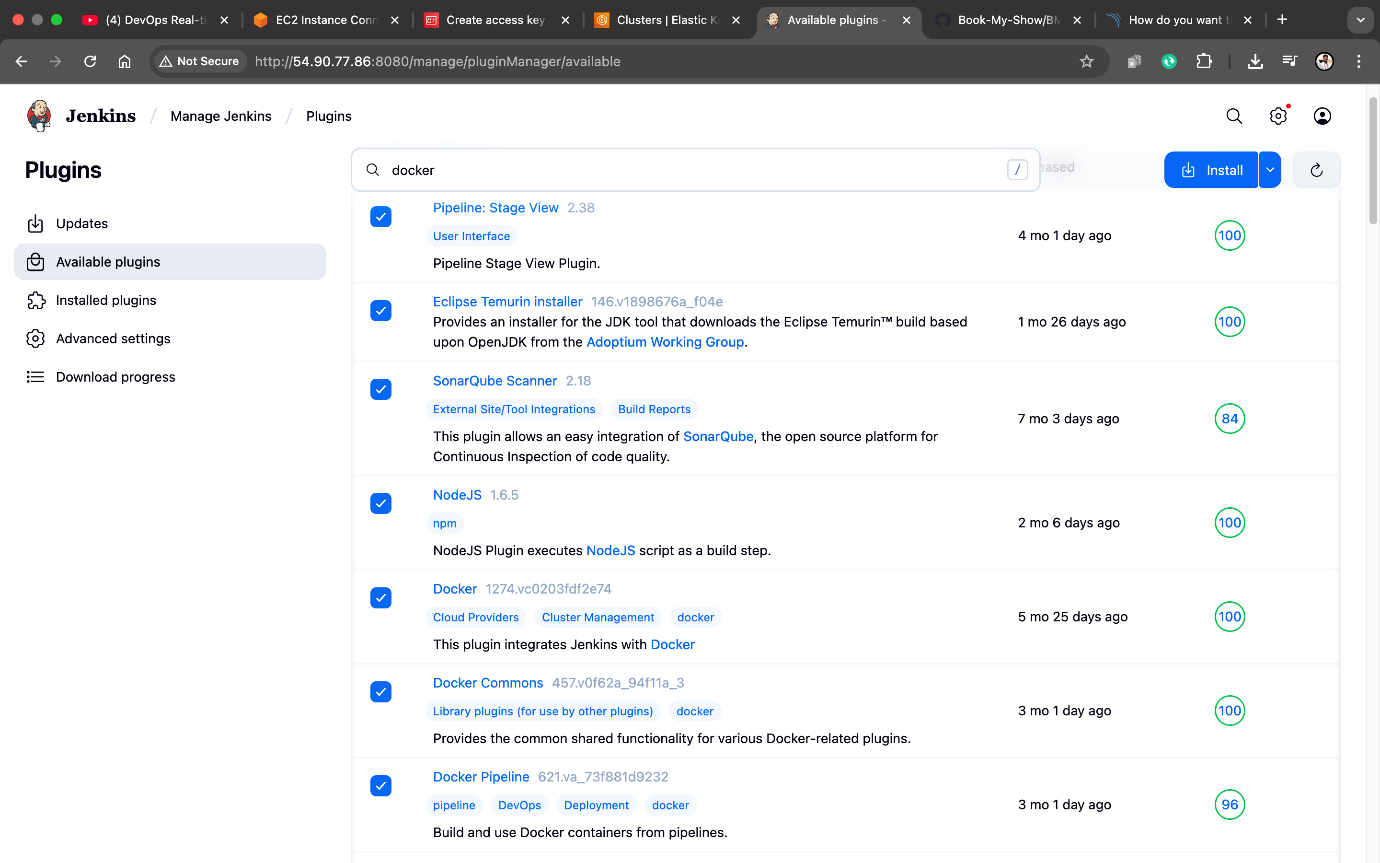
--without-nodegroup



Now open Jenkins and install plugins   
manage Jenkins-plugins-available plugins

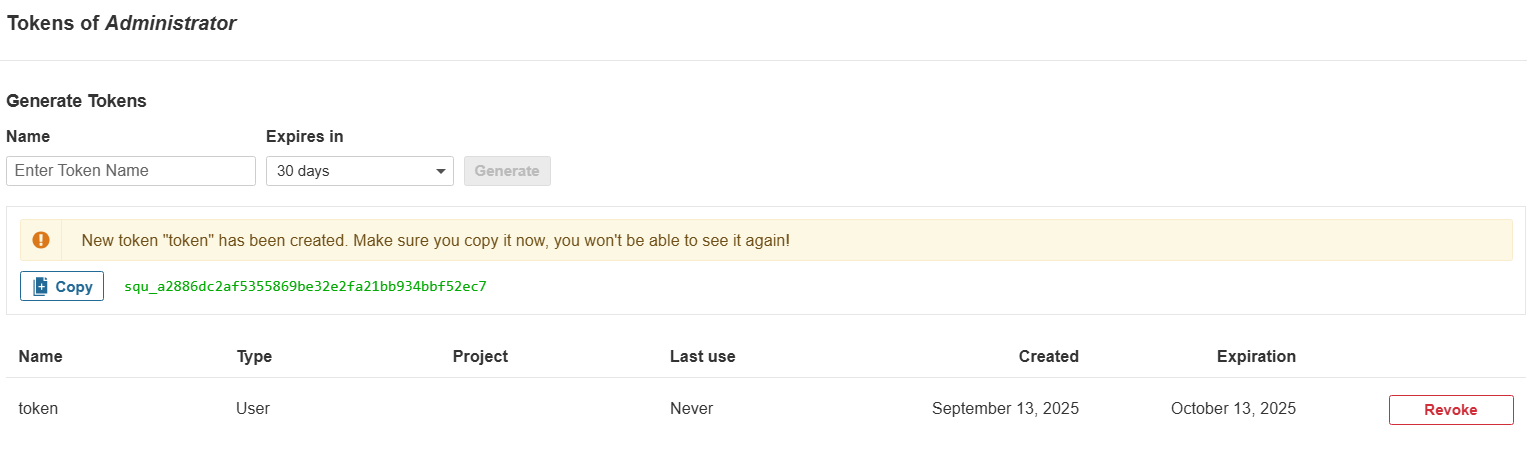
Install below plugins;

Eclipse Temurin Installer,  
SonarQube scanner,   
NodeJS,  
Docker,  
Docker Commons,  
Docker Pipeline,  
Docker API,  
docker-build-step,  
OWASP dependency check,  
Pipeline stage view,  
Email Extension Template,  
Kubernetes,  
Kubernetes CLI,  
Kubernetes Client API,  
Kubernetes Credentials,  
Config File Provider,  
Prometheus metrics.



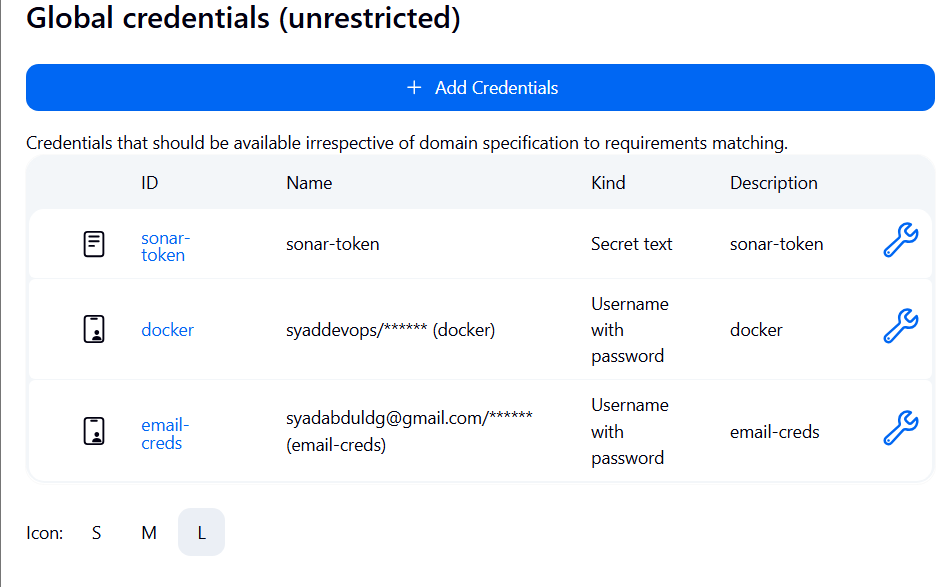


Prometheus metrics requires restart check the restart Jenkins box and re-login into Jenkins   
  
Goto sonarqube under administration click goto Security dropdown select users click on 4 dashes and create a token.

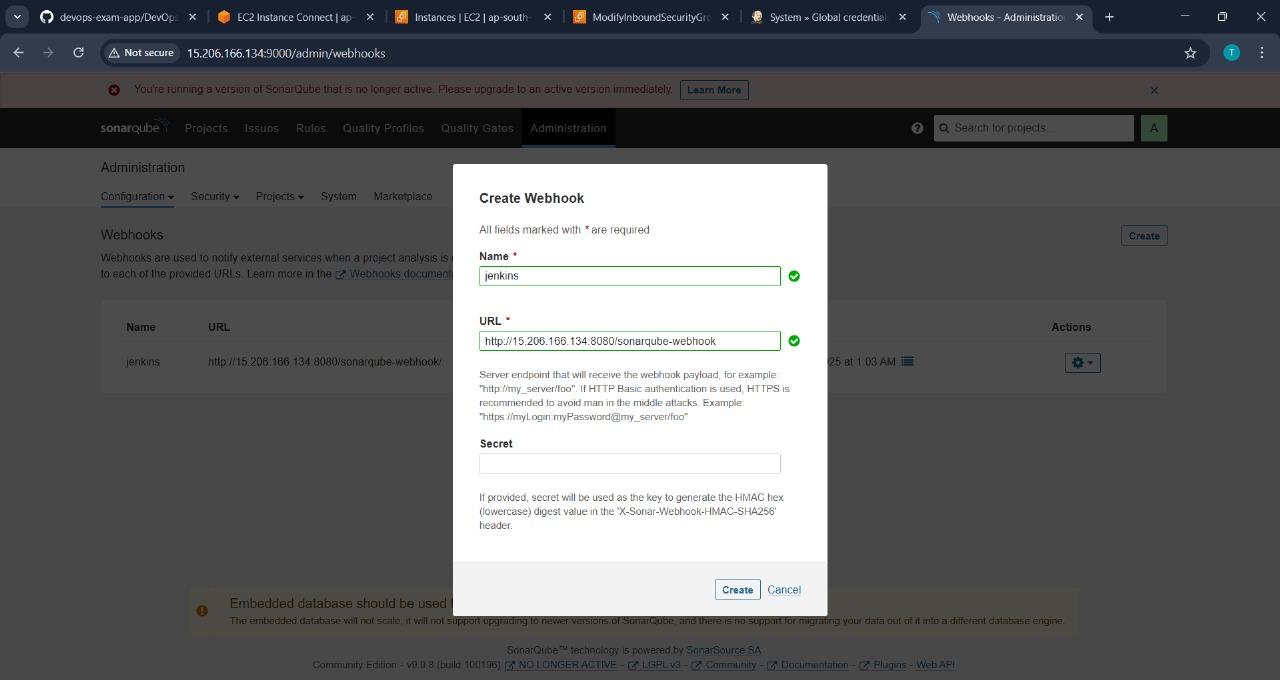
Go to sonarqube and create token: administrator->security->user  


Go to Jenkins   
manage Jenkins > credentials > System > Global credentials

Create sonarqube credentials as secret text id: sonar-token  
Add docker creds as well   
Add credentials select kind as username with password   
username: yourdocker username  
password : yourdocker password



We need to create sonarqube Webhook as well  
under Administration goto configuration > select Webhooks

Name:Jenkins  
URL: <http://publicip:8080/sonarqube-webhook>/ (give / at end)  
no secret click on create  


Go to manage Jenkins and tools under JDK installations

----------------------------------------------------------  
click on Add jdk  
name: jdk17  
select install automatically

under installer dropdown  
select “install from adoptium.net”   
version? : select jdk-17.0.8.1+1 version

-----------------------------------------------------------

Add SonarQube scanner installations   
Name: sonar-scanner  
select install automatically  
version?: SonarQube scanner 7.0.1.4817  
------------------------------------------------------------  
Node JS installation   
Name : node23  
select install automatically

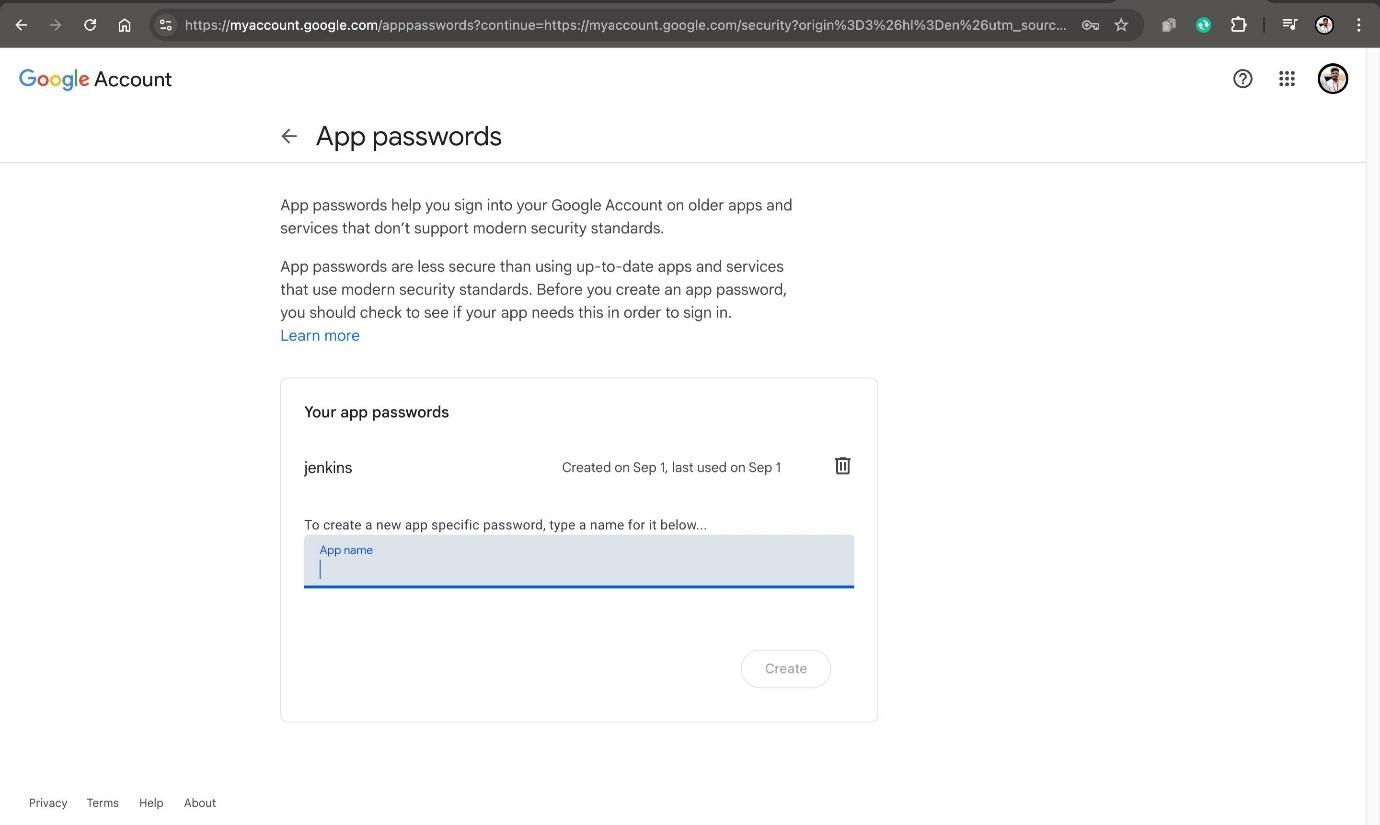
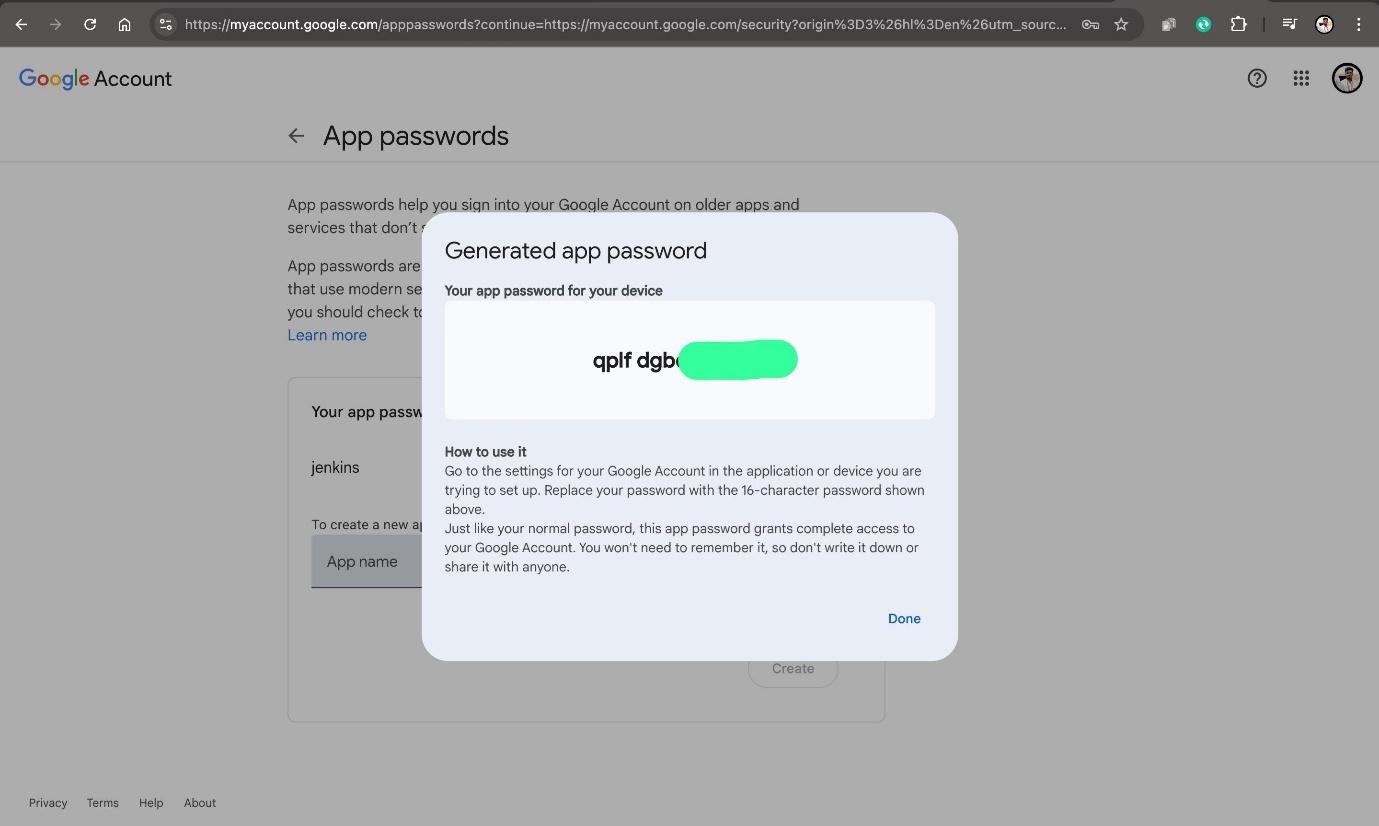
Version?: NodeJS 23.7.0  
--------------------------------------------------------------  
Dependency-check installations  
Add Dependency-check

Name: DP-Check  
select install automatically  
add installer   
install from github.com  
  
version?: dependency-check 12.0.2  
------------------------------------------------------------------  
add docker   
name: docker

select install automatically  
add installer  
Download from docker.com  
  
click on apply and save  
  
we need to tell where sonarqube is running to Jenkins as well.  
  
go to system configuration under system   
search for sonarQube-servers  
  
click on Add SonarQube  
  
Name: sonar-server  
URL: http://public ip:9000 -------------(donot keep / at the end )  
server authentication token dropdown   
select Sonar-token   
  
click on apply and save  
  
we need to integrate Email

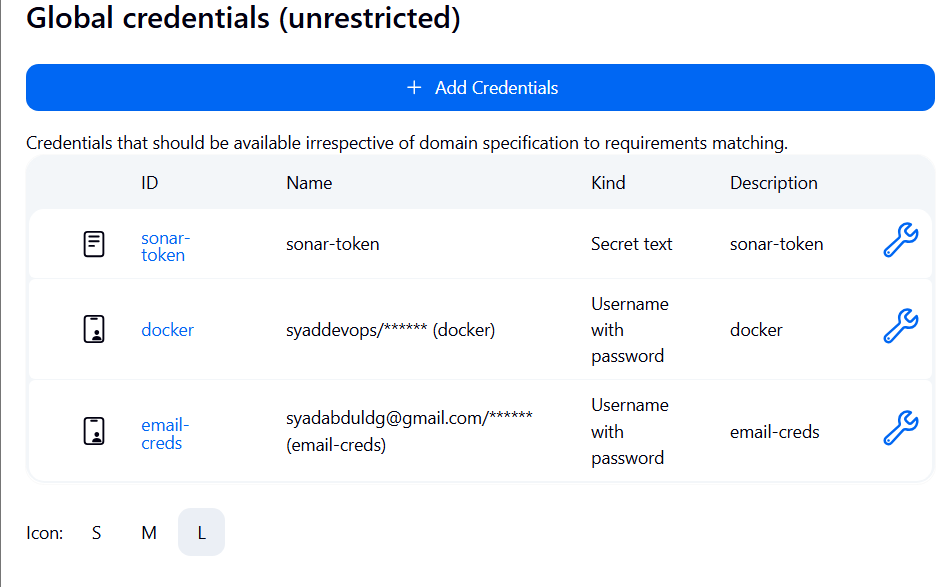
we need password to access gmail  
  
to get secret password got to your gmail click on profile icon click on Manage google account   
in left pane click on security make sure TWO FACTOR AUTHENTICATION is done

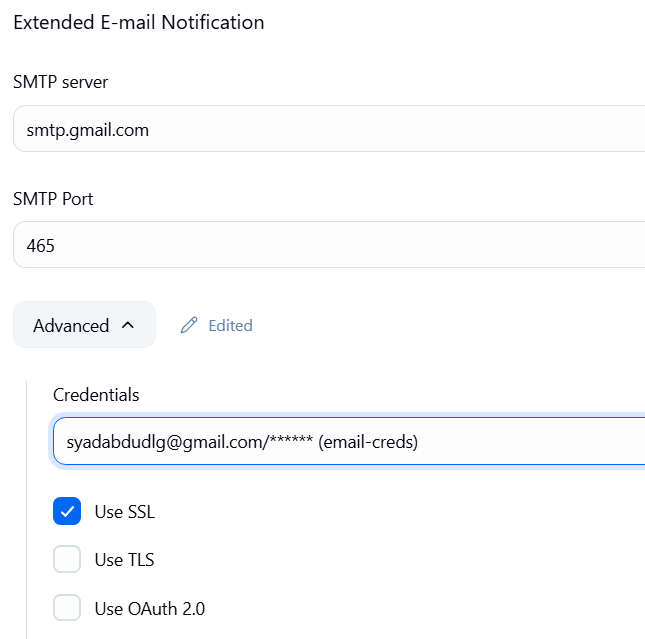


Search for app password  
create app password with   
name jenkins   
  


manage Jenkins > security click on global > add credentials

kind: username with password  
scope: global  
Username: your email id  
password: paste the generated password without spaces

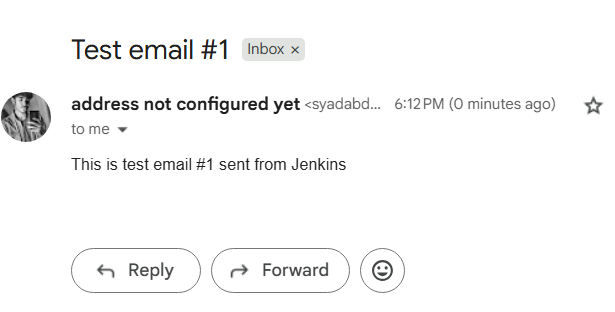
ID: email-creds  
description: email-creds  


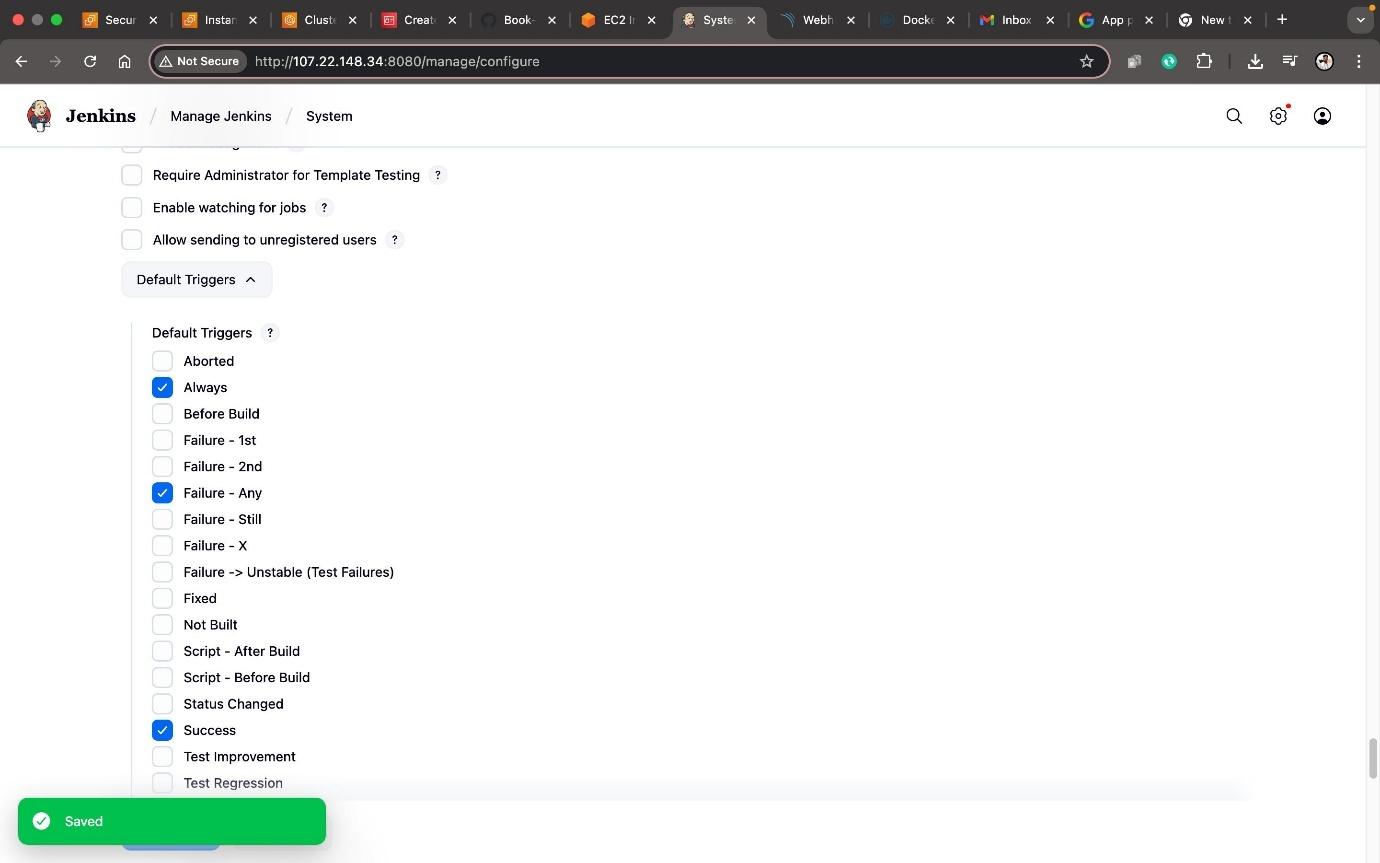
Got to Manage Jenkins and system   
extended E-mail notifications: SMTP server   
smtp.gmail.com  
  


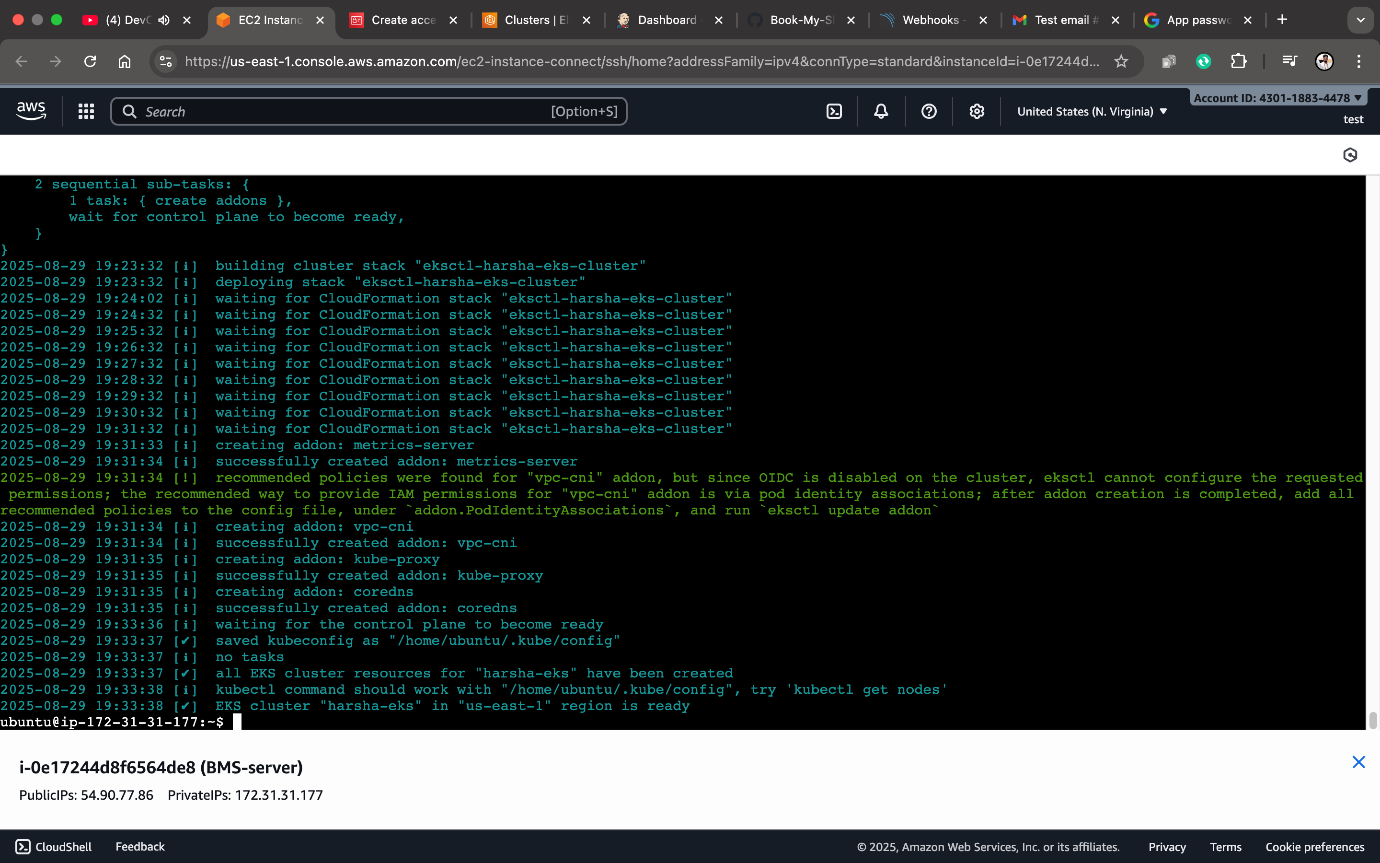
Choose only use SSL.

Default content type : HTML  
  
**E-mail Notification**  
  
SMTP server   
smtp.gmail.com  
  
   


Password : paste the generated password without spaces  
  
check Use SSL  
SMTP port: 465  
Reply to address: your email address  
  
check Test configuration by sending test e-mail  
provide your email id  
click on test config.

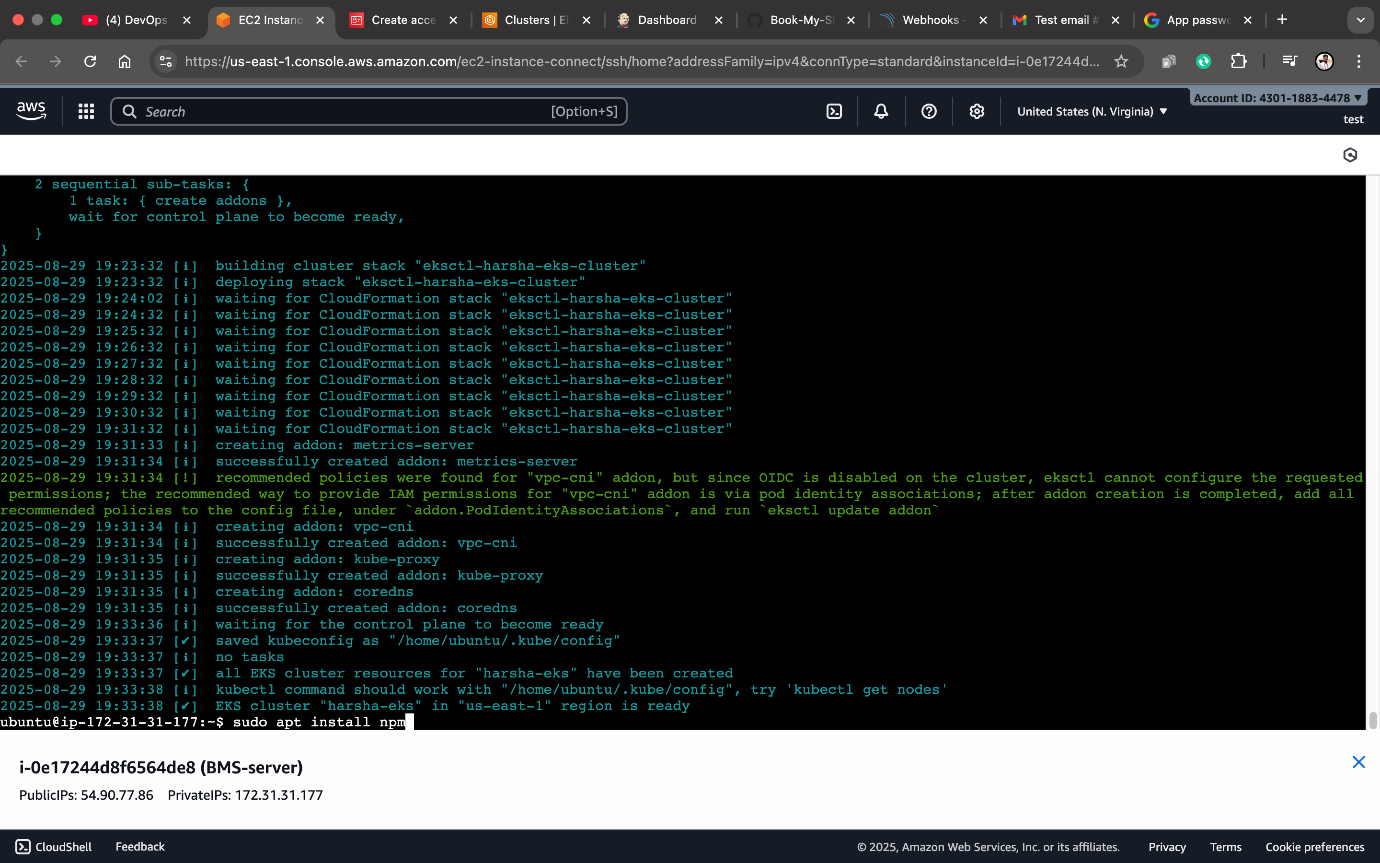


Check your inbox for “Test email”  
   
Search for default Triggers   
under dropdown   
select   


Now go back to machine to install NPM node port manager  
  
as we can see EKS clusters are ready   


Go to eks console and click on clusters you can see your cluster with the name Syad-eks  
  
But worker nodes are not available   
 to do that

sudo apt install npm

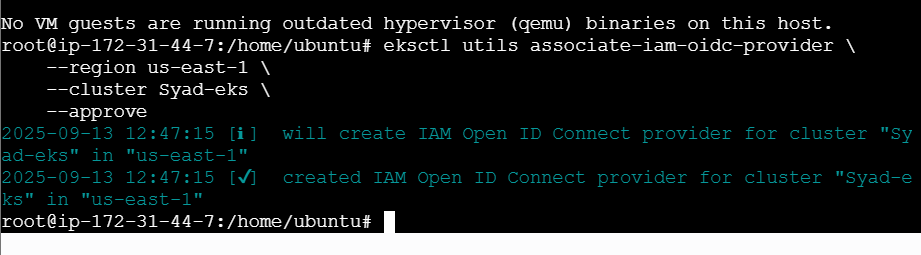
****

eksctl utils associate-iam-oidc-provider \

--region us-east-1 \

--cluster Syad-eks \

--approve



**Before executing the below command, in the 'ssh-public-key' keep the '<PEM FILE NAME>' (dont give .pem. Just give the pem file name) which was used to create Jenkins Server**

eksctl create nodegroup --cluster=Syad-eks \ # <- change your cluster name

--region=us-east-1 \

--name=node2 \

--node-type=t3.medium \

--nodes=3 \

--nodes-min=2 \

--nodes-max=4 \

--node-volume-size=20 \

--ssh-access \

--ssh-public-key=kuberadmin \ #<- change you pem file name

--managed \

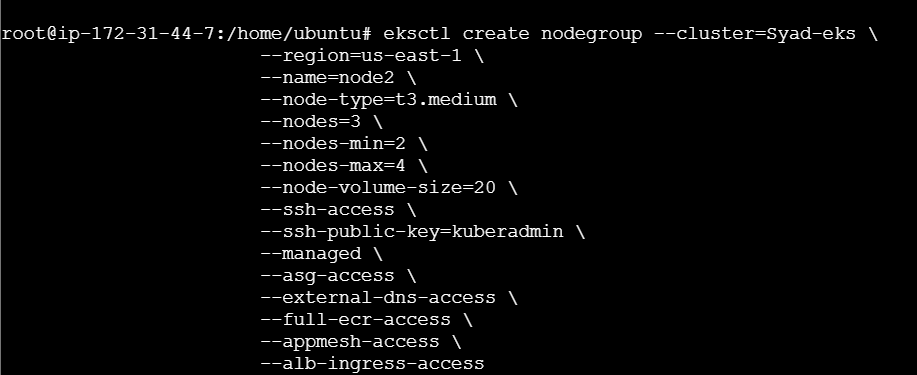
--asg-access \

--external-dns-access \

--full-ecr-access \

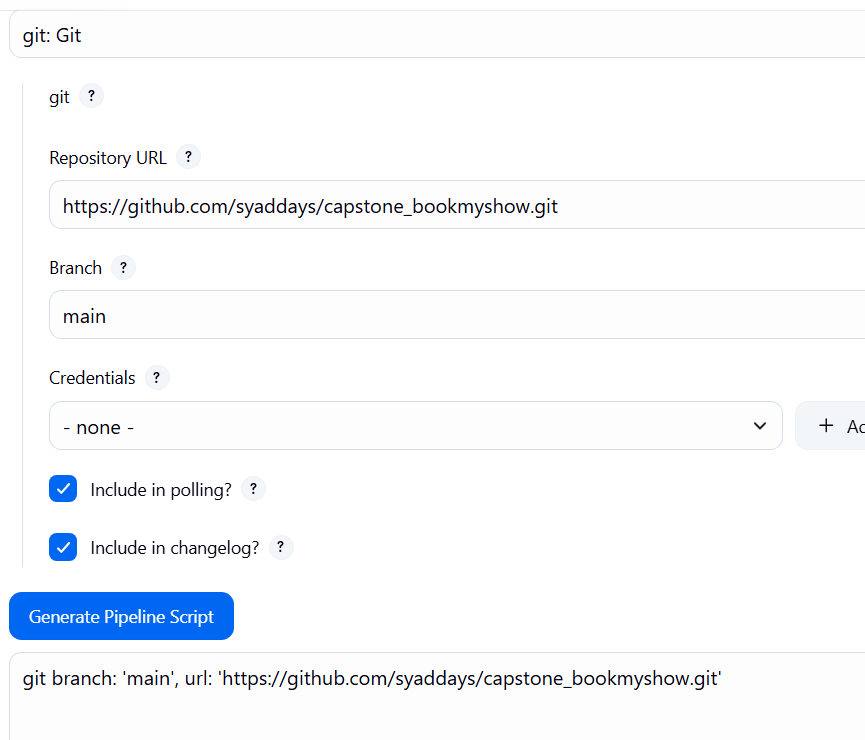
--appmesh-access \

--alb-ingress-access



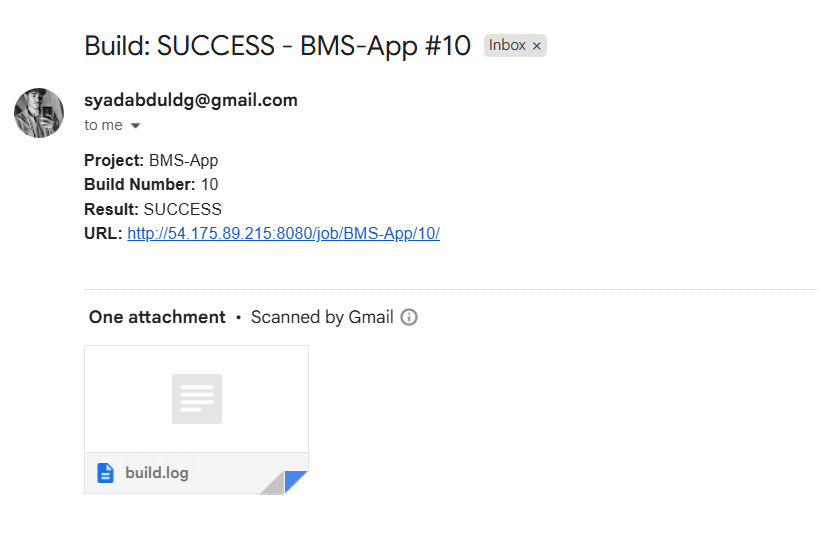
create new item (project) in Jenkins   
name give it as BMS-App  
select pipeline click on ok

Select pipeline syntax in newtab paste the forked repository and copy the url  
sample step : git:Git



**Paste the Jenkinsfile1 in the script after generating the pipeline syntax  
make changes to   
1.) docker username  
2.) git url (your git url)  
3.) at last block of code change E-mail (your email address)  
  
click on apply and click on save  
click on build now OWASP takes around 45min for execution.**

****



Jenkinsfile1:

pipeline {

agent any

tools {

jdk 'jdk17'

nodejs 'node23'

}

environment {

SCANNER\_HOME = tool 'sonar-scanner'

}

stages {

stage('Clean Workspace') {

steps {

cleanWs()

}

}

stage('Checkout from Git') {

steps {

git branch: 'main', url: 'https://github.com/syaddays/capstone\_bookmyshow.git'

sh 'ls -la' // Verify files after checkout

}

}

stage('SonarQube Analysis') {

steps {

withSonarQubeEnv('sonar-server') {

sh '''

$SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=BMS \

-Dsonar.projectKey=BMS

'''

}

}

}

stage('Quality Gate') {

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

}

}

stage('Install Dependencies') {

steps {

sh '''

cd bookmyshow-app

ls -la # Verify package.json exists

if [ -f package.json ]; then

rm -rf node\_modules package-lock.json # Remove old dependencies

npm install # Install fresh dependencies

else

echo "Error: package.json not found in bookmyshow-app!"

exit 1

fi

'''

}

}

stage('OWASP FS Scan') {

steps {

dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'

dependencyCheckPublisher pattern: '\*\*/dependency-check-report.xml'

}

}

stage('Trivy FS Scan') {

steps {

sh 'trivy fs . > trivyfs.txt'

}

}

stage('Docker Build & Push') {

steps {

script {

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh '''

echo "Building Docker image..."

docker build --no-cache -t syaddevops/bms:latest -f bookmyshow-app/Dockerfile bookmyshow-app

echo "Pushing Docker image to registry..."

docker push syaddevops/bms:latest

'''

}

}

}

}

stage('Deploy to Container') {

steps {

sh '''

echo "Stopping and removing old container..."

docker stop bms || true

docker rm bms || true

echo "Running new container on port 3000..."

docker run -d --restart=always --name bms -p 3000:3000 syaddevops/bms:latest

echo "Checking running containers..."

docker ps -a

echo "Fetching logs..."

sleep 5 # Give time for the app to start

docker logs bms

'''

}

}

}

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

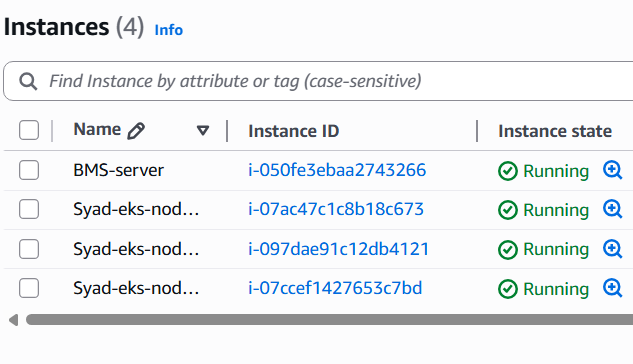
to: 'syadabduldg@gmail.com',

attachmentsPattern: 'trivyfs.txt,trivyimage.txt'

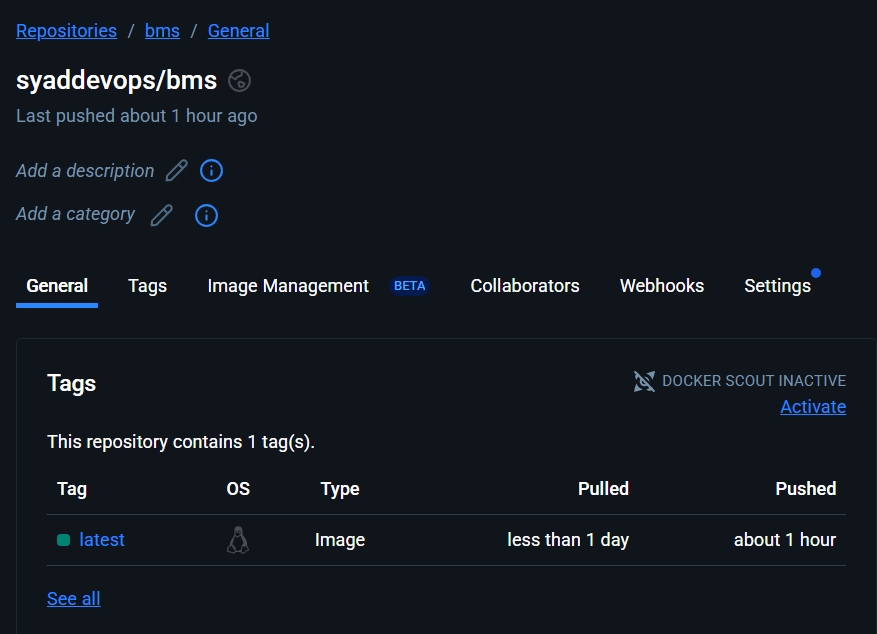
}

}

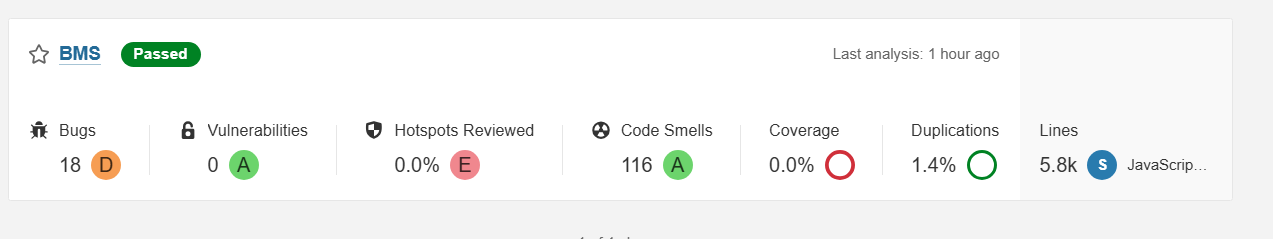
}



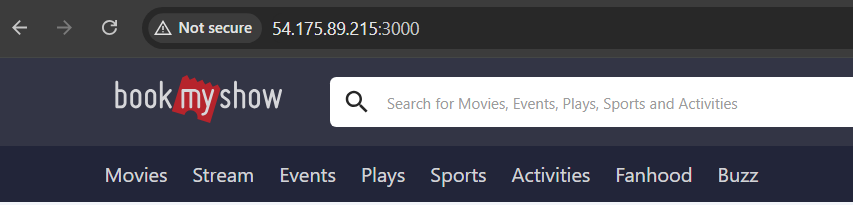
Dockerhub repo : https://hub.docker.com/repository/docker/syaddevops/bms/general



SONARQUBE ANALYSIS (Quality Gate)



DOCKER DEPLOYMENT ON PORT 3000:



ps aux | grep jenkins

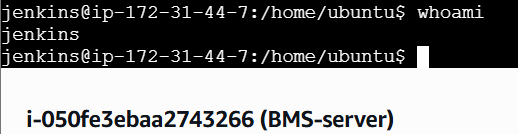
sudo -su jenkins

whoam i

aws configure

aws sts get-caller-identity

you should be able to see UserId ,Account and Arn number as below



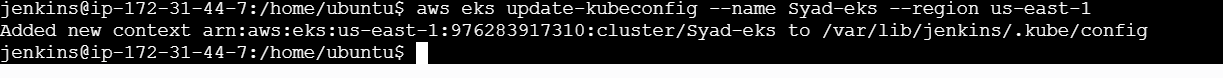
exit

whoami

sudo systemctl restart jenkins

sudo -su jenkins

aws eks update-kubeconfig --name Syad-eks --region us-east-1



**re login into your jenkins for Jenkinsfile2  
  
click on configure and copy the entire script and paste here   
change the following**

**1.)git url**

**2.)email**

**3.)docker username**

**4.)AWS\_region us-east-1**

pipeline {

agent any

tools {

jdk 'jdk17'

nodejs 'node23'

}

environment {

SCANNER\_HOME = tool 'sonar-scanner'

}

stages {

stage('Clean Workspace') {

steps {

cleanWs()

}

}

stage('Checkout from Git') {

steps {

git branch: 'main', url: 'https://github.com/syaddays/capstone\_bookmyshow.git'

sh 'ls -la' // Verify files after checkout

}

}

stage('SonarQube Analysis') {

steps {

withSonarQubeEnv('sonar-server') {

sh '''

$SCANNER\_HOME/bin/sonar-scanner -Dsonar.projectName=BMS \

-Dsonar.projectKey=BMS

'''

}

}

}

stage('Quality Gate') {

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

}

}

}

stage('Install Dependencies') {

steps {

sh '''

cd bookmyshow-app

ls -la # Verify package.json exists

if [ -f package.json ]; then

rm -rf node\_modules package-lock.json # Remove old dependencies

npm install # Install fresh dependencies

else

echo "Error: package.json not found in bookmyshow-app!"

exit 1

fi

'''

}

}

stage('Docker Build & Push') {

steps {

script {

withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {

sh '''

echo "Building Docker image..."

docker build --no-cache -t syaddevops/bms:latest -f bookmyshow-app/Dockerfile bookmyshow-app

echo "Pushing Docker image to registry..."

docker push syaddevops/bms:latest

'''

}

}

}

}

stage('Deploy to Container') {

steps {

sh '''

echo "Stopping and removing old container..."

docker stop bms || true

docker rm bms || true

echo "Running new container on port 3000..."

docker run -d --restart=always --name bms -p 3000:3000 syaddevops/bms:latest

echo "Checking running containers..."

docker ps -a

echo "Fetching logs..."

sleep 5 # Give time for the app to start

docker logs bms

'''

}

}

}

post {

always {

emailext attachLog: true,

subject: "'${currentBuild.result}'",

body: "Project: ${env.JOB\_NAME}<br/>" +

"Build Number: ${env.BUILD\_NUMBER}<br/>" +

"URL: ${env.BUILD\_URL}<br/>",

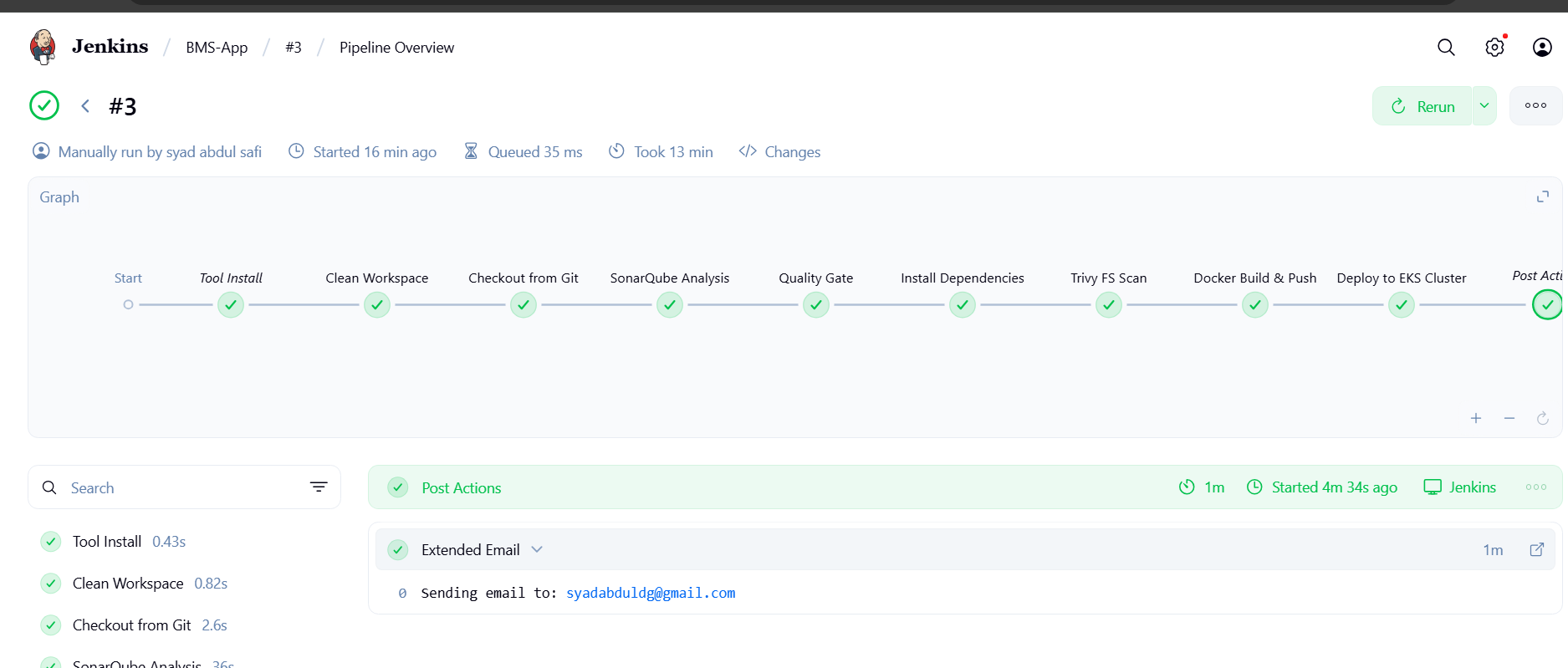
to: 'syadabduldg@gmail.com',

attachmentsPattern: 'trivyfs.txt,trivyimage.txt'

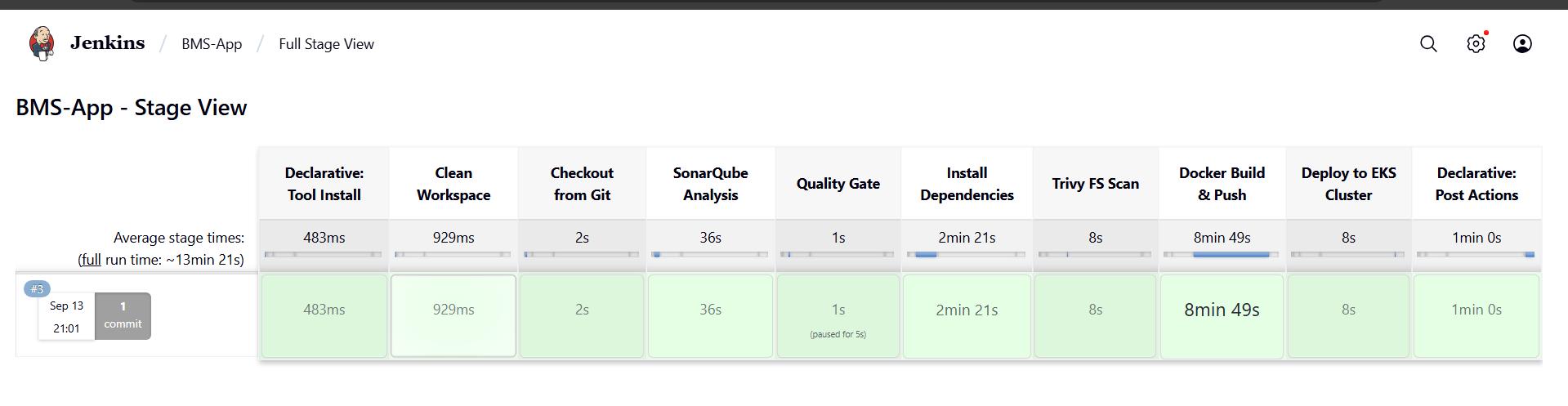
}

}

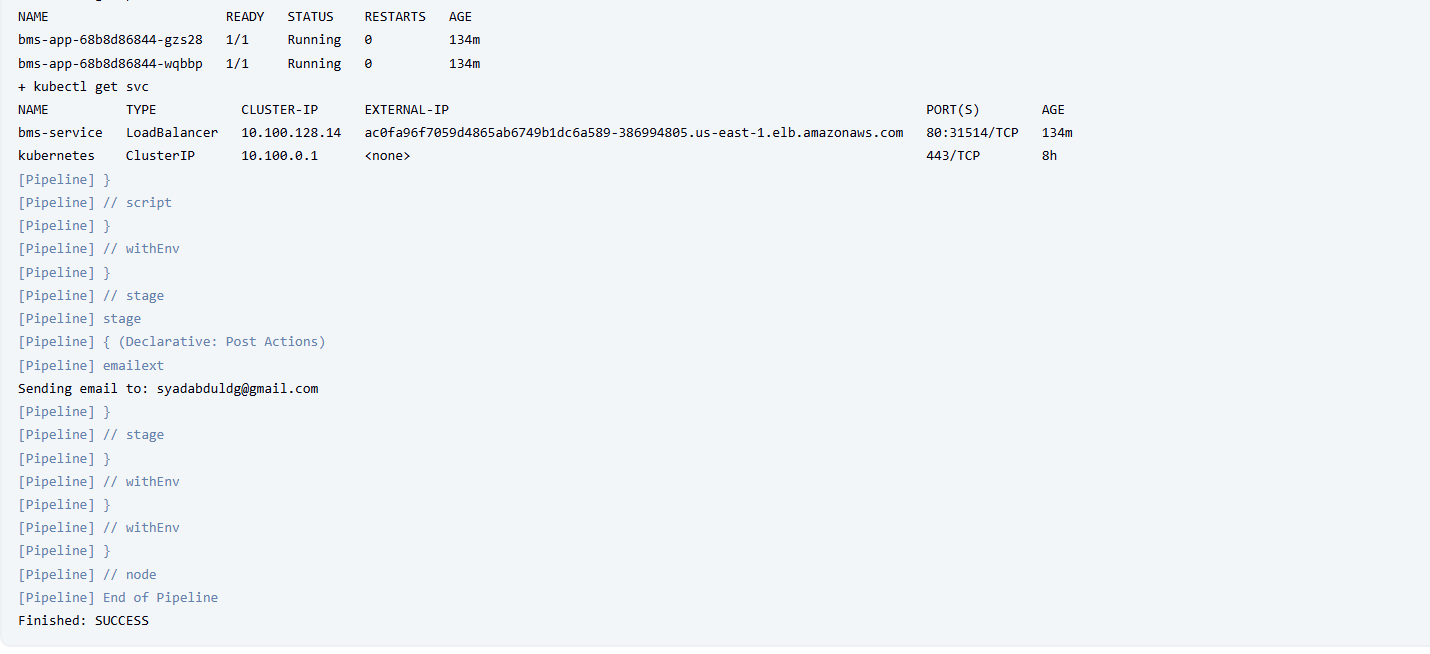
}



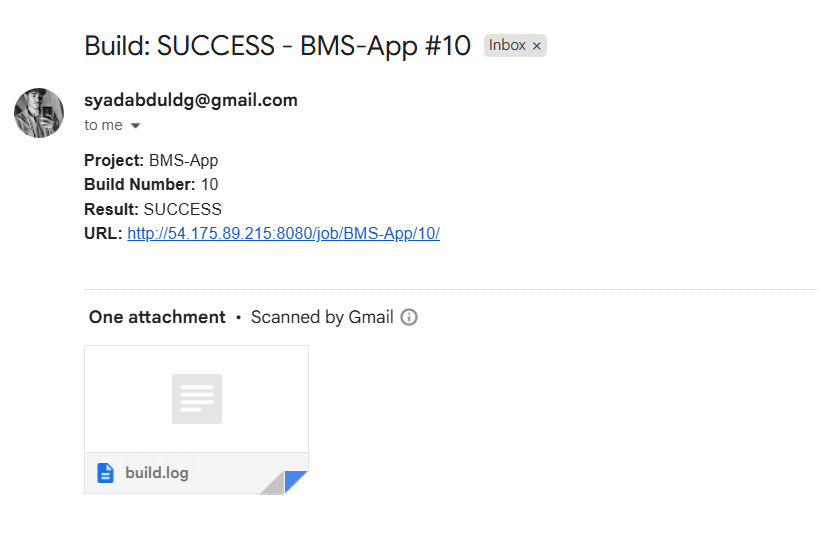
Pipeline stage view



Pipeline console log:



EMAIL NOTIFICATION:



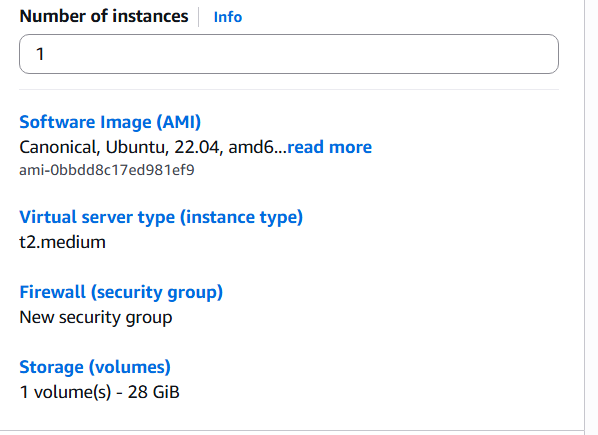
MONITORING:

**monitoring the applications using Prometheus and Grafana**

For this we need to launch another instance name it as monitoring-server

Select ubuntu but change the version to 22.04 ver and t2.medium   
  
a pop up message will appear click on Confirm changes   
  
select the same key pair   
config storage as 1x28 click on launch instance

Create new instance for monitoring:



sudo apt update

sudo useradd \

--system \

--no-create-home \

--shell /bin/false prometheus



sudo wget <https://github.com/prometheus/prometheus/releases/download/v2.47.1/prometheus-2.47.1.linux-amd64.tar.gz>

tar -xvf prometheus-2.47.1.linux-amd64.tar.gz

sudo mkdir -p /data /etc/prometheus

cd prometheus-2.47.1.linux-amd64/

sudo mv prometheus promtool /usr/local/bin/

sudo mv consoles/ console\_libraries/ /etc/prometheus/

sudo mv prometheus.yml /etc/prometheus/prometheus.yml

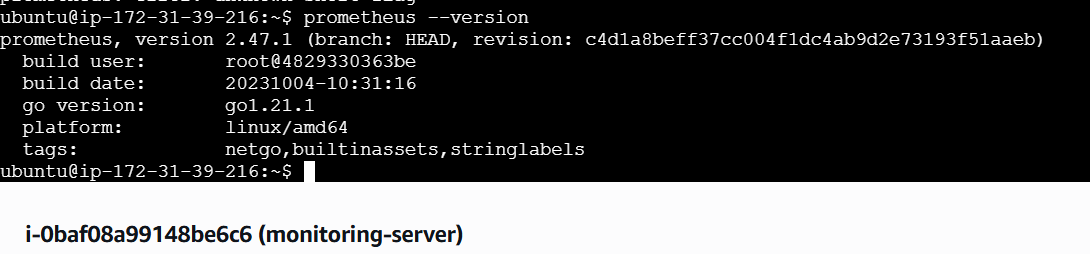
sudo chown -R prometheus:prometheus /etc/prometheus/ /data/

cd

ls

rm -rf prometheus-2.47.1.linux-amd64.tar.gz

prometheus –versio**n**



sudo vi /etc/systemd/system/prometheus.service

[Unit]

Description=Prometheus

Wants=network-online.target

After=network-online.target

StartLimitIntervalSec=500

StartLimitBurst=5

[Service]

User=prometheus

Group=prometheus

Type=simple

Restart=on-failure

RestartSec=5s

ExecStart=/usr/local/bin/prometheus \

--config.file=/etc/prometheus/prometheus.yml \

--storage.tsdb.path=/data \

--web.console.templates=/etc/prometheus/consoles \

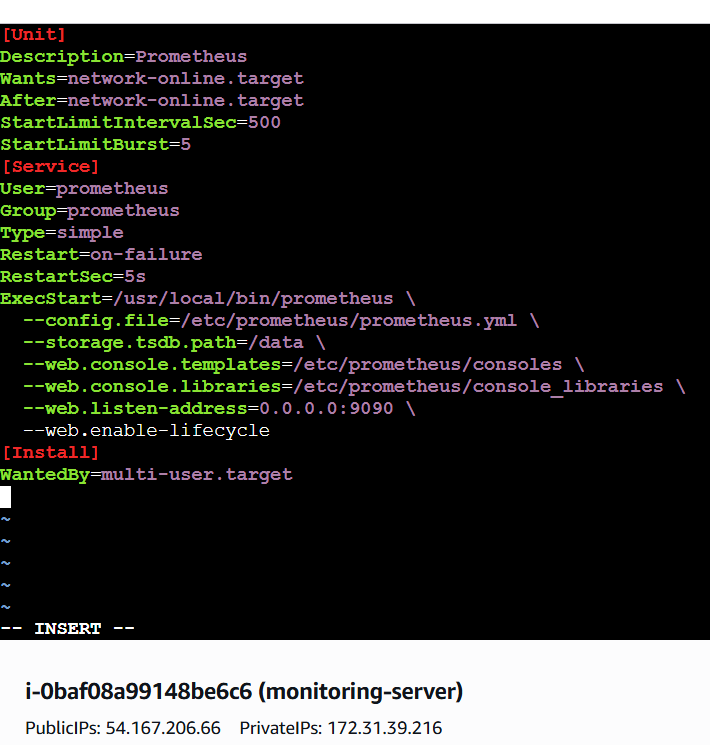
--web.console.libraries=/etc/prometheus/console\_libraries \

--web.listen-address=0.0.0.0:9090 \

--web.enable-lifecycle

[Install]

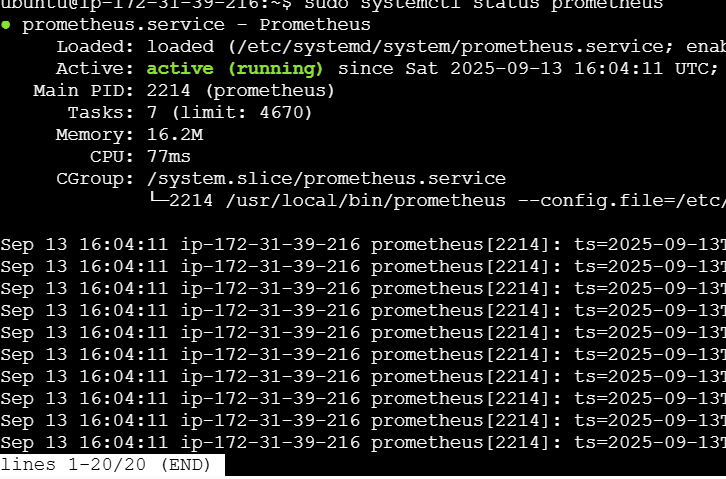
WantedBy=multi-user.target



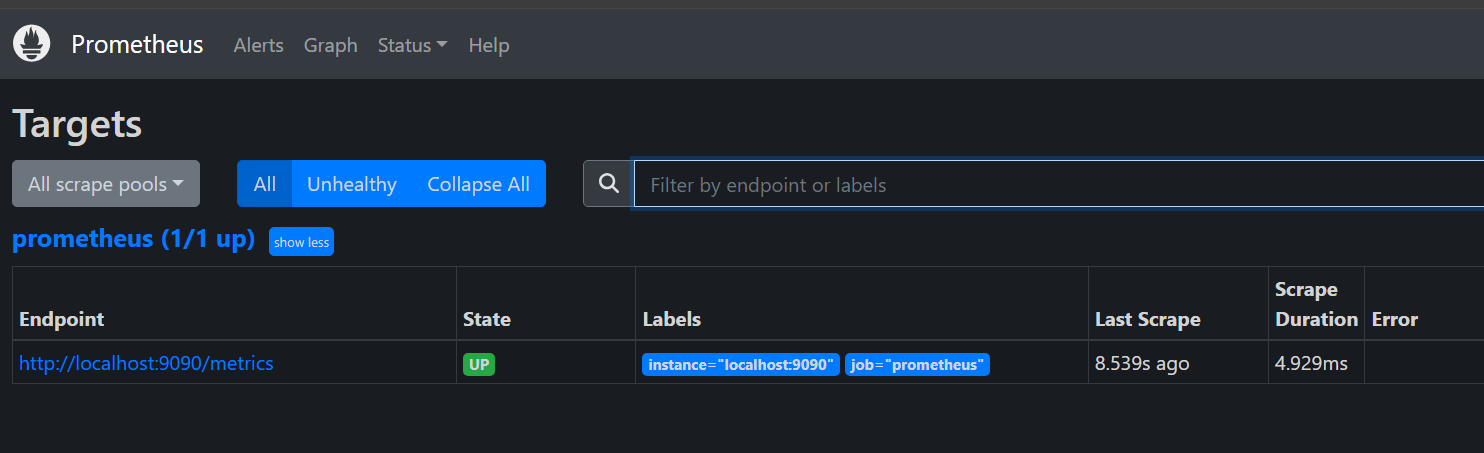
sudo systemctl enable prometheus

sudo systemctl start prometheus

sudo systemctl status prometheus



Open Port No. 9090 for Monitoring Server VM and Access Prometheus



Create a system user for Node Exporter and download Node Exporter:

sudo useradd --system --no-create-home --shell /bin/false node\_exporter

wget https://github.com/prometheus/node\_exporter/releases/download/v1.6.1/node\_exporter-1.6.1.linux-amd64.tar.gz

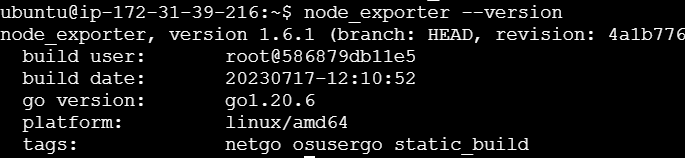
Extract Node Exporter files, move the binary, and clean up:

tar -xvf node\_exporter-1.6.1.linux-amd64.tar.gz

sudo mv node\_exporter-1.6.1.linux-amd64/node\_exporter /usr/local/bin/

rm -rf node\_exporter\*

node\_exporter --version



Create a systemd unit configuration file for Node Exporter:

sudo vi /etc/systemd/system/node\_exporter.service

[Unit]

Description=Node Exporter

Wants=network-online.target

After=network-online.target

StartLimitIntervalSec=500

StartLimitBurst=5

[Service]

User=node\_exporter

Group=node\_exporter

Type=simple

Restart=on-failure

RestartSec=5s

ExecStart=/usr/local/bin/node\_exporter --collector.logind

[Install]

WantedBy=multi-user.target

**Enable and start Node Exporter:**

sudo systemctl enable node\_exporter

sudo systemctl start node\_exporter

Verify the Node Exporter's status:

sudo systemctl status node\_exporter

You can see "active (running)" in green colour

Press control+c to come out of the file



**Prometheus Configuration:**

cd /etc/prometheus/

sudo vi prometheus.yml

- job\_name: 'node\_exporter'

static\_configs:

- targets: ['54.167.206.66:9100']

- job\_name: 'jenkins'

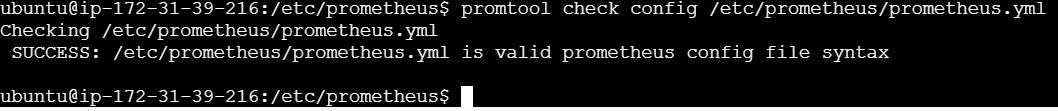
metrics\_path: '/prometheus'

static\_configs:

- targets: ['54.167.206.66:8080']

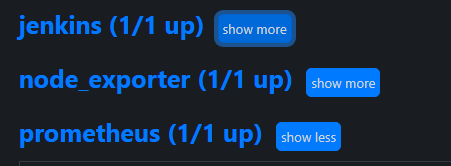
promtool check config /etc/prometheus/prometheus.yml

You will see success return.



curl -X POST <http://localhost:9090/-/reload>

this will restart the Prometheus

reload the page you will see this   


**Install Grafana (Execute in Monitoring Server VM)**

sudo apt-get update

sudo apt-get install -y apt-transport-https software-properties-common

cd

pwd

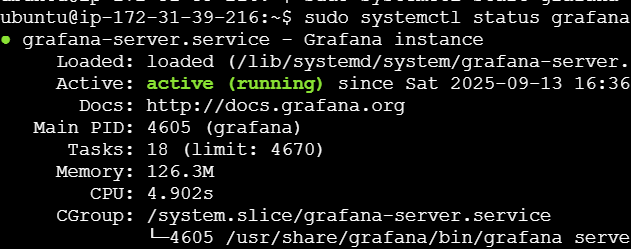
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

sudo apt-get update && \

sudo apt-get install -y grafana && \

sudo systemctl enable grafana-server && \

sudo systemctl start grafana-server

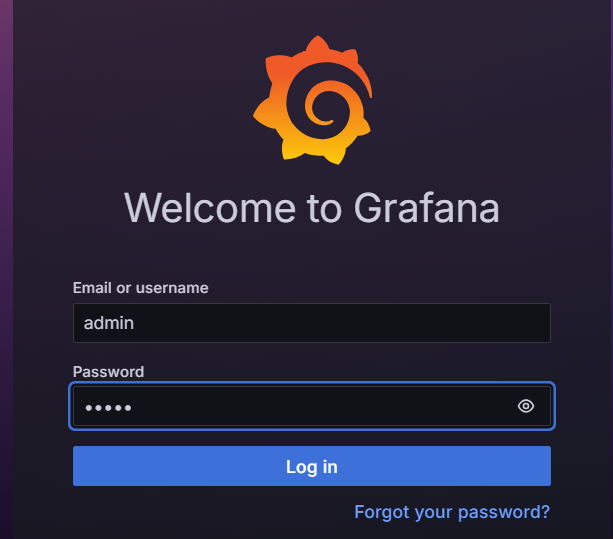


**Press Ctrl + c or q**

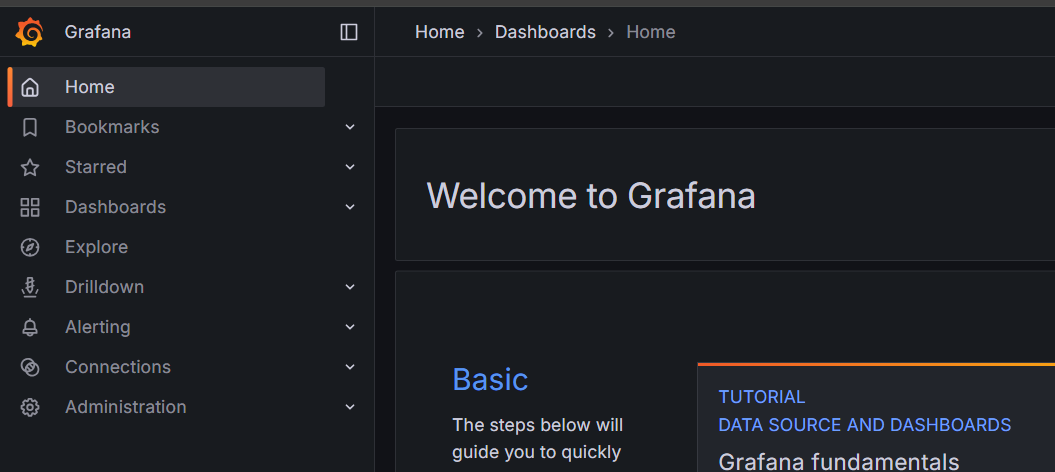
**The default port for Grafana is 3000**

**http://<monitoring-server-ip>:3000**

**User name and pwd are admin and admin default**



Change pswrd.



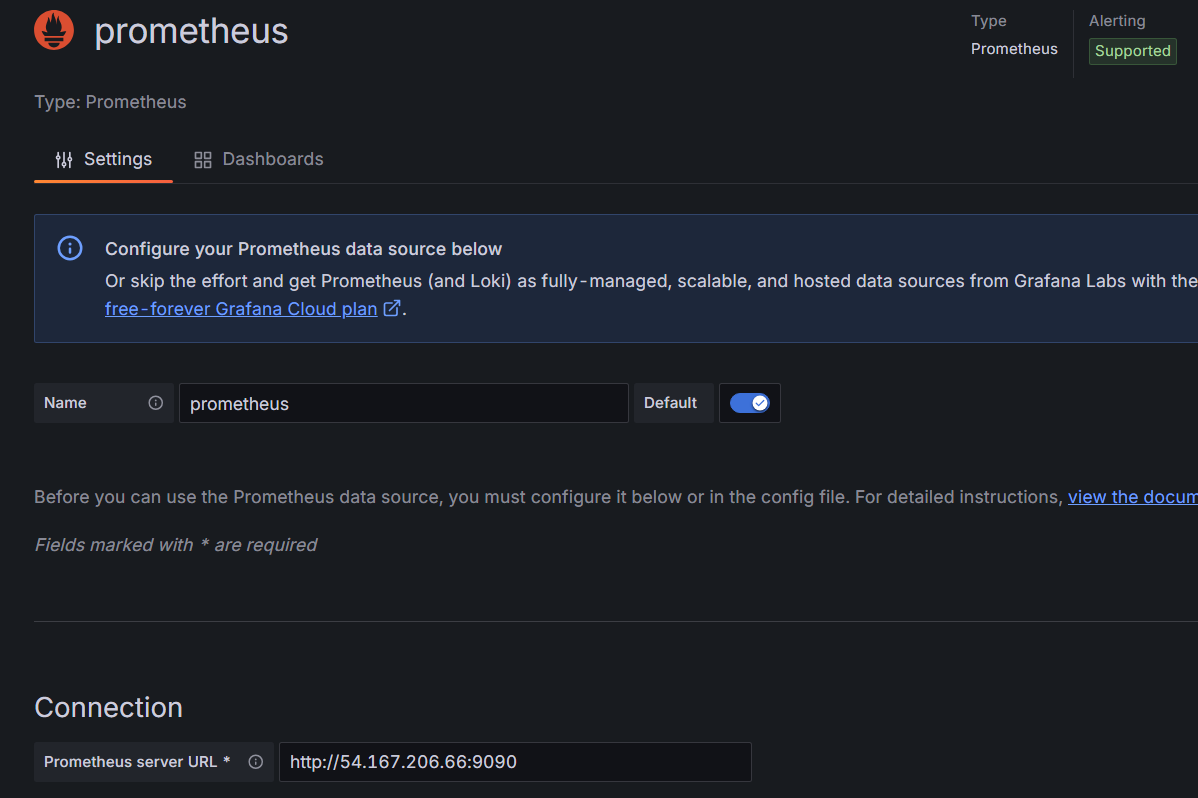
Click on DATA SOURCES

Click on + right top corner

Select import Dashboard in the dropdown

Select Prometheus

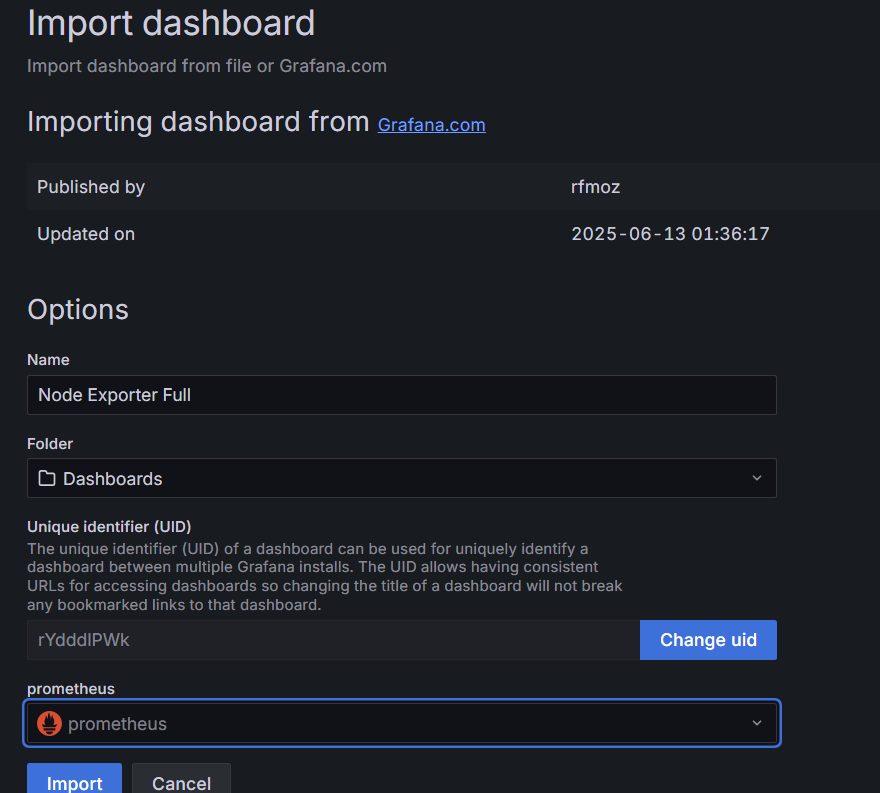
Give Prometheus url

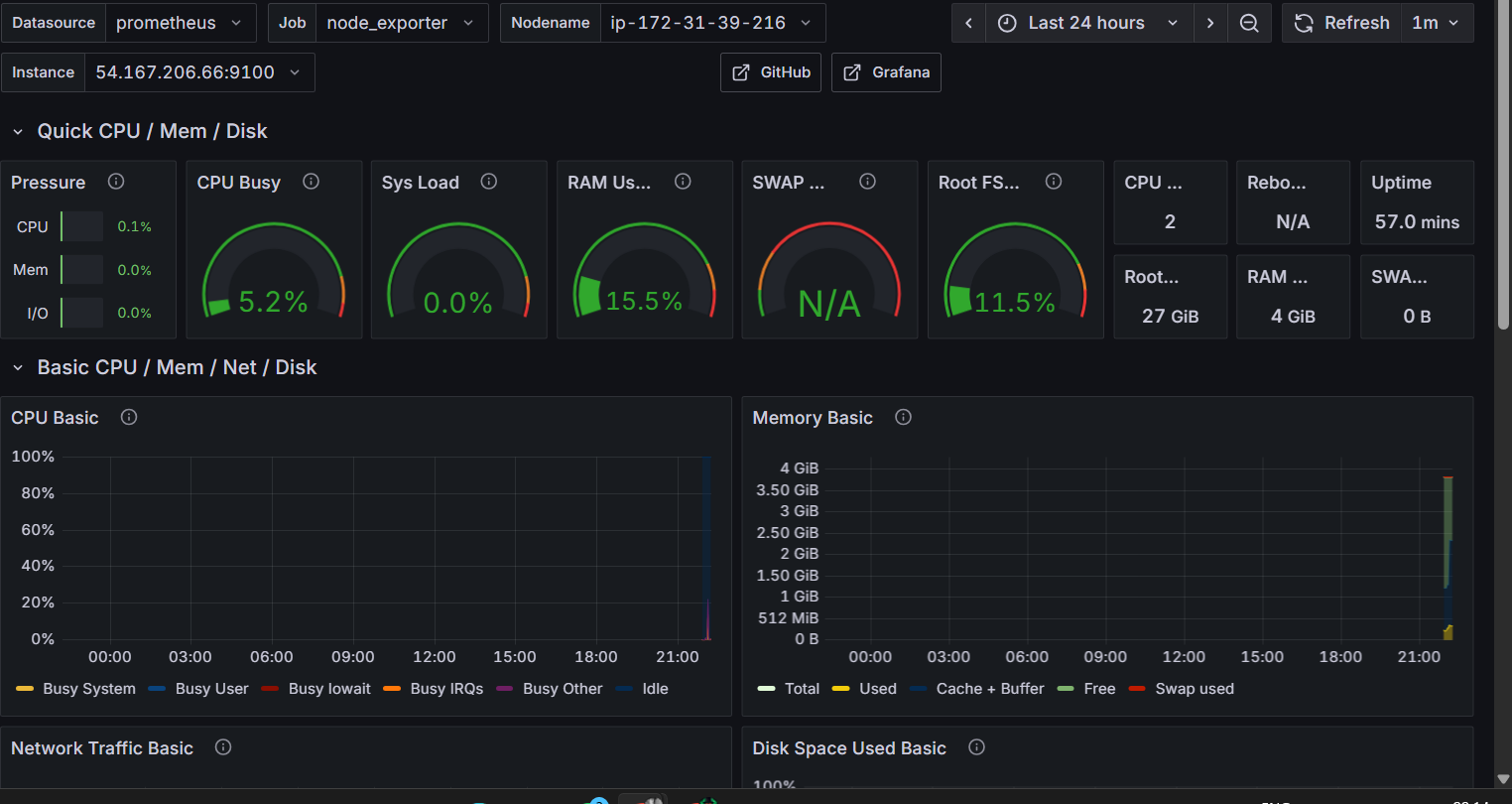


Click on save and test

Again import dashboard

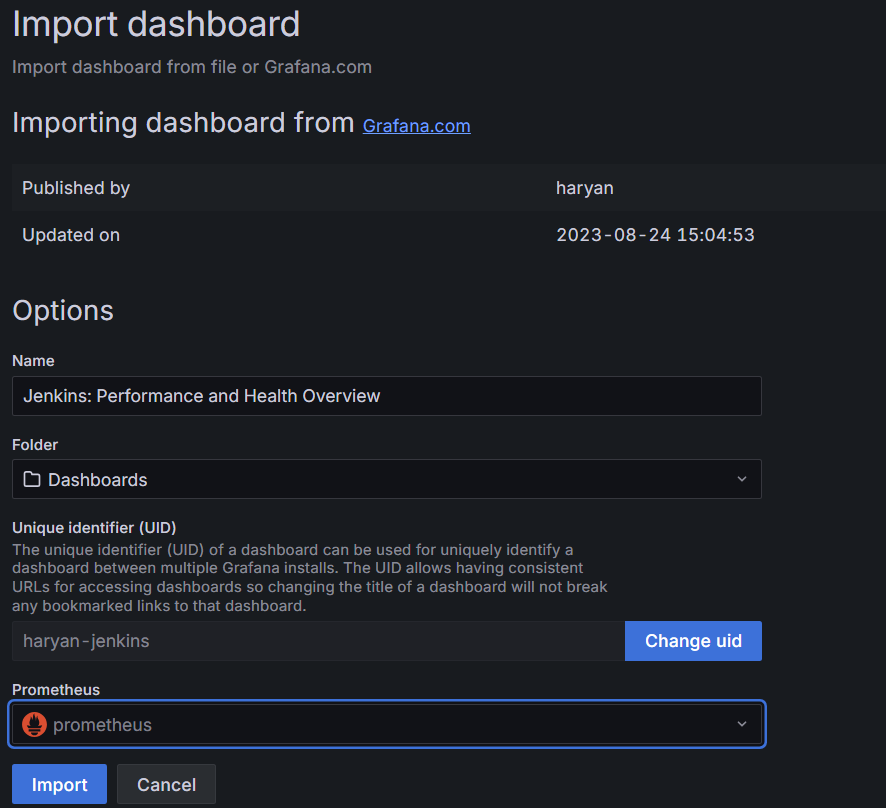
(URL: <https://grafana.com/grafana/dashboards/1860-node-exporter-full/>)

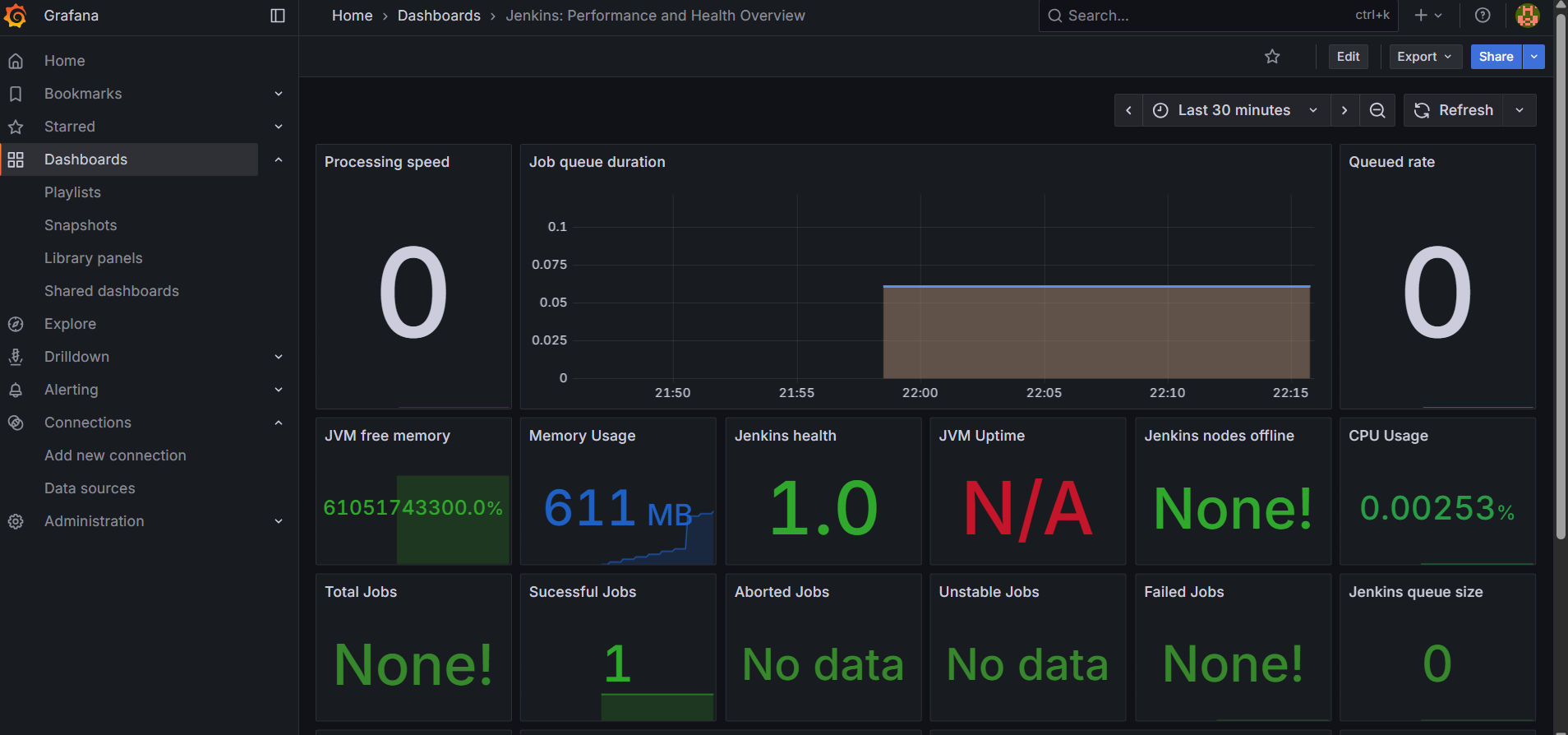




For jenkins as well   
click on + dropdown import Dashboard  
(URL: <https://grafana.com/grafana/dashboards/9964-jenkins-performance-and-health-overview/>)

select source as prometheues



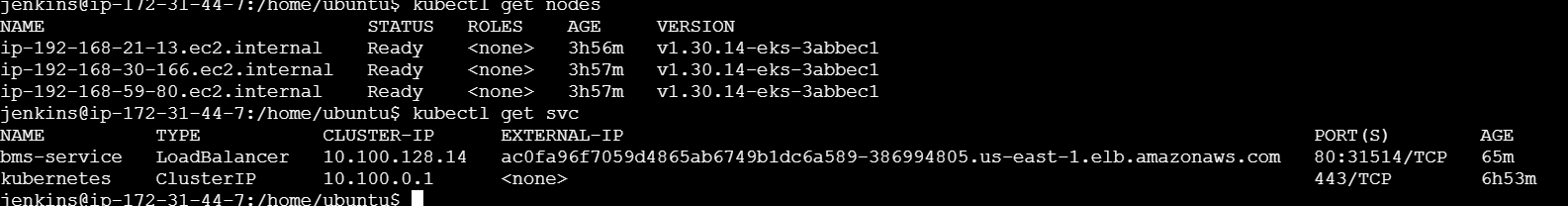


deploy the website using load balancer:

kubectl get nodes

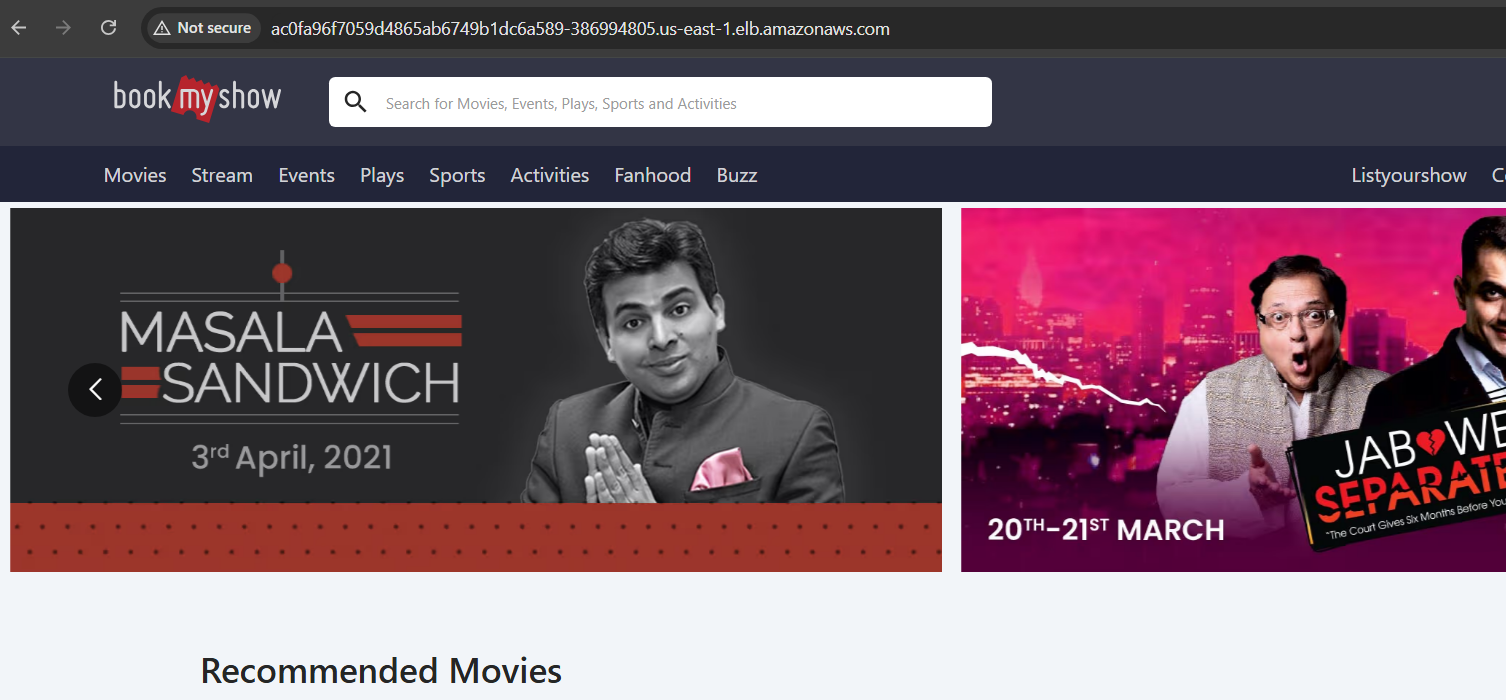
kubectl get svc

available nodes will appear



Go to external ip of bms-service to view website:

http://ac0fa96f7059d4865ab6749b1dc6a589-386994805.us-east-1.elb.amazonaws.com/



Clean up:

eksctl delete cluster --name Syad-eks --region us-east-1

