**Devops Application Deployment**

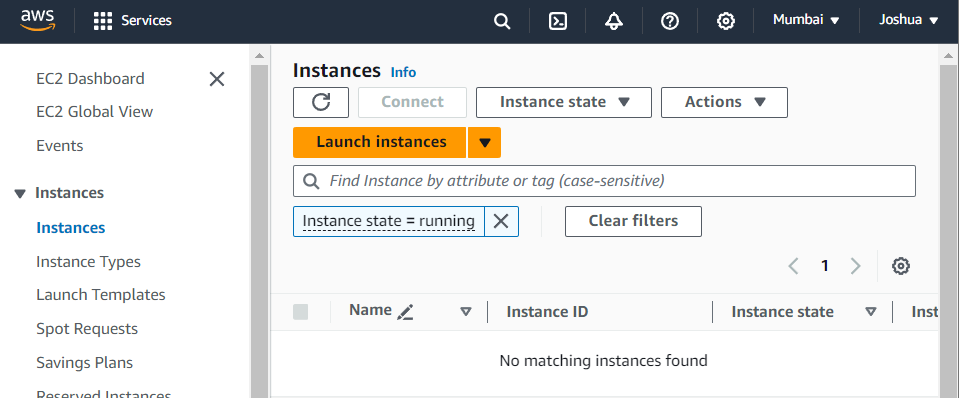
<https://docs.google.com/document/d/11kbFude1yL9C3r--VX-HM_bZYuWNusVI1P_mITJTDc4/edit>

**My github link:**<https://github.com/Joshuaaj/react-joz.git> (see both main and dev)

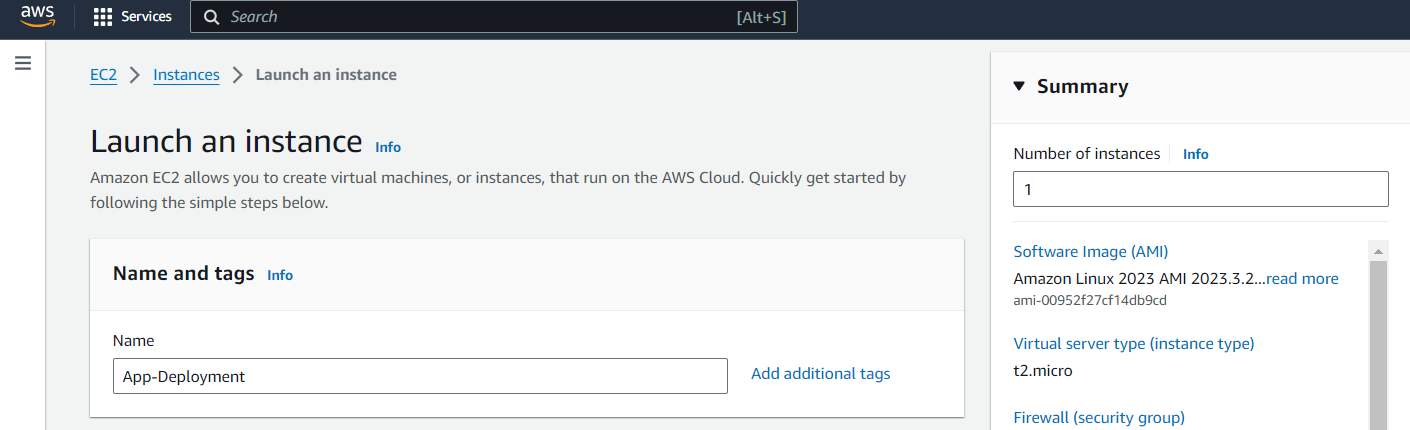
**Create a EC2 instance** ,

First in the AWS services search for EC2 instance → Click on instances→ Click launch instance → Give a name for the instance → select the required instance type and required settings → Launch instance → Connect instance

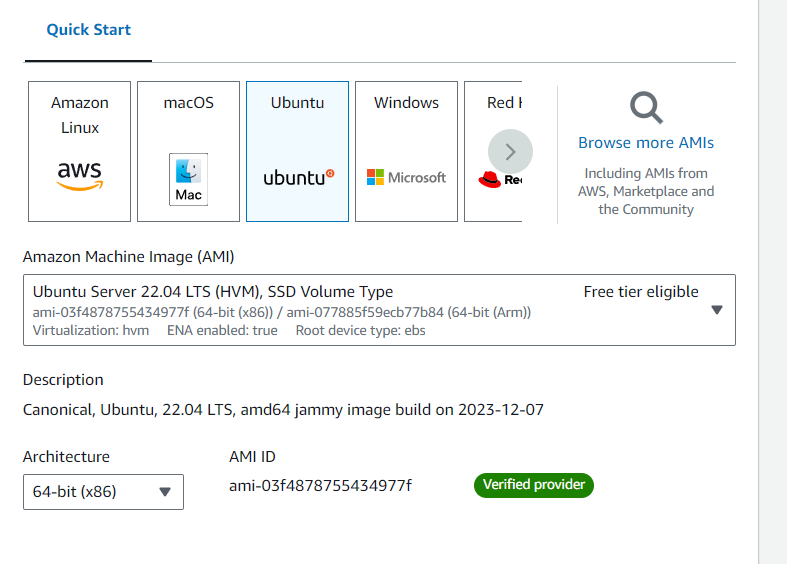
Click launch instance



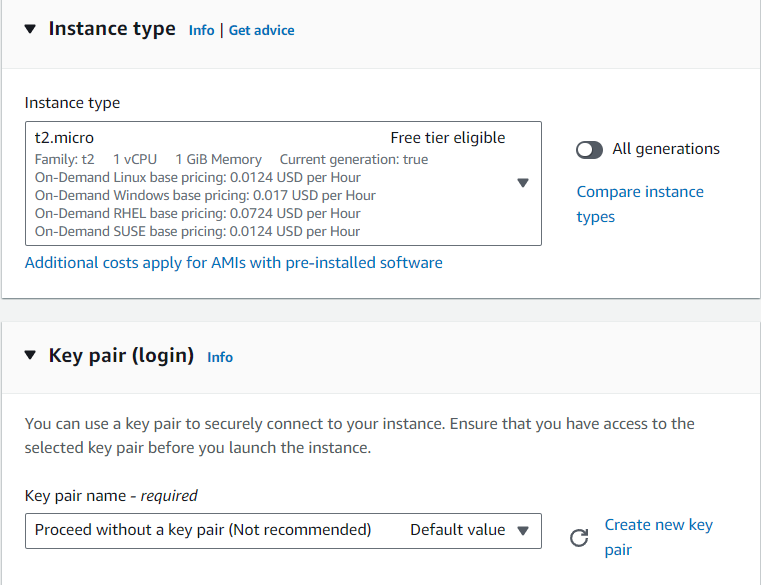
Give a name for the instance



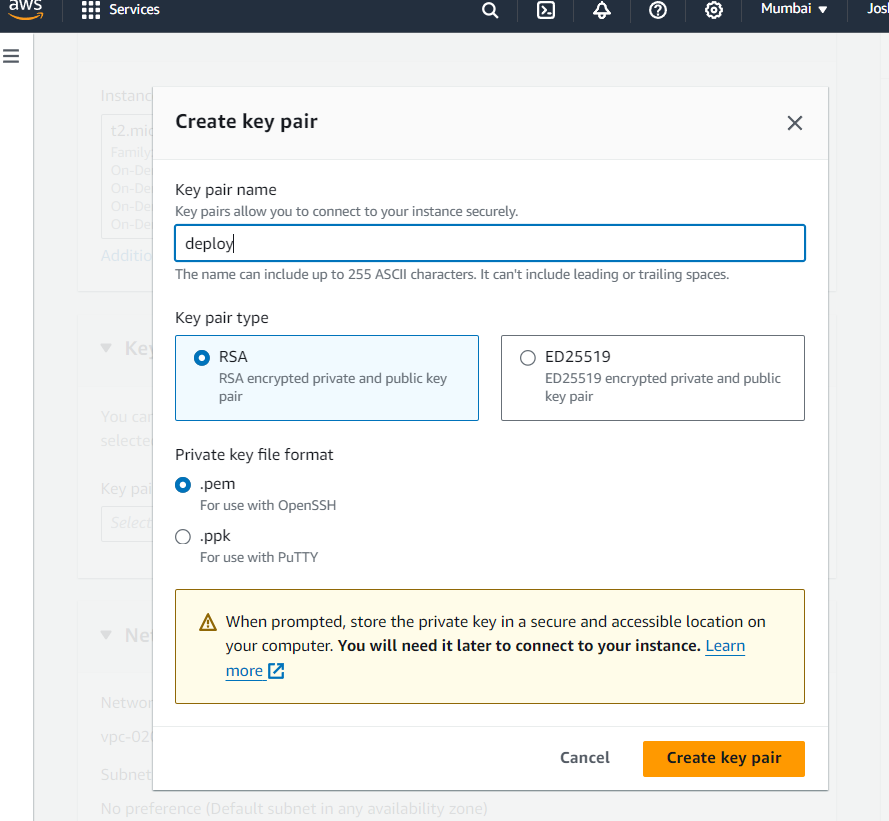
Select ubuntu AMI image

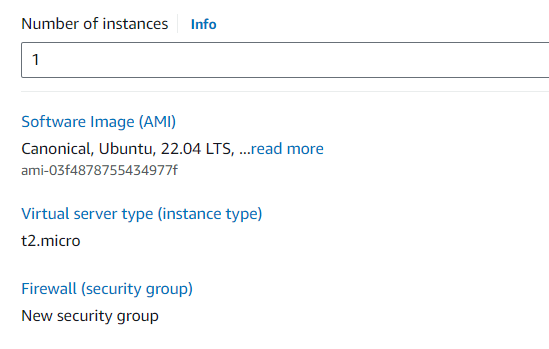
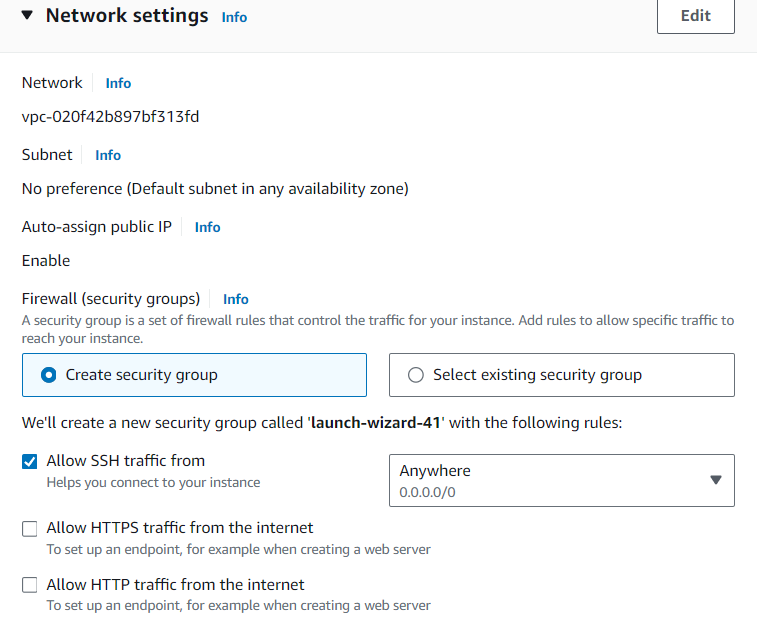


Select instance type and key pair

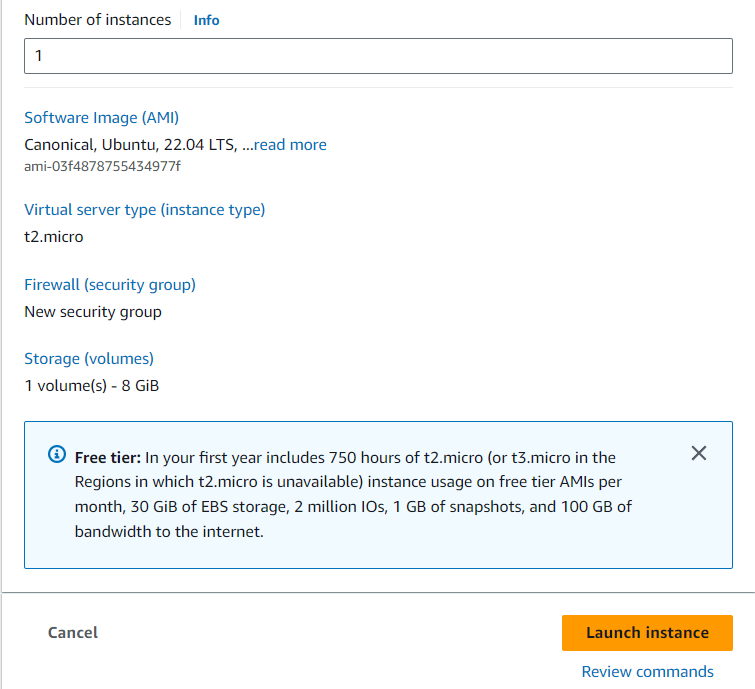


Set key pair

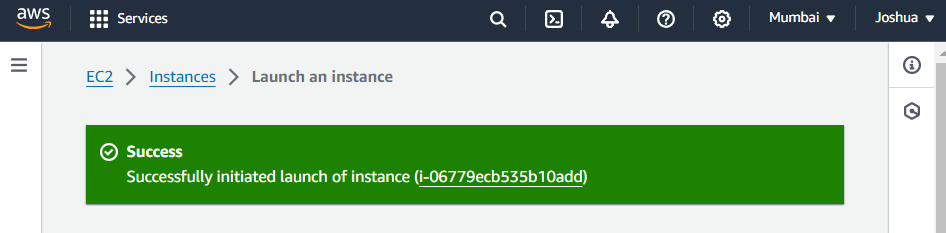




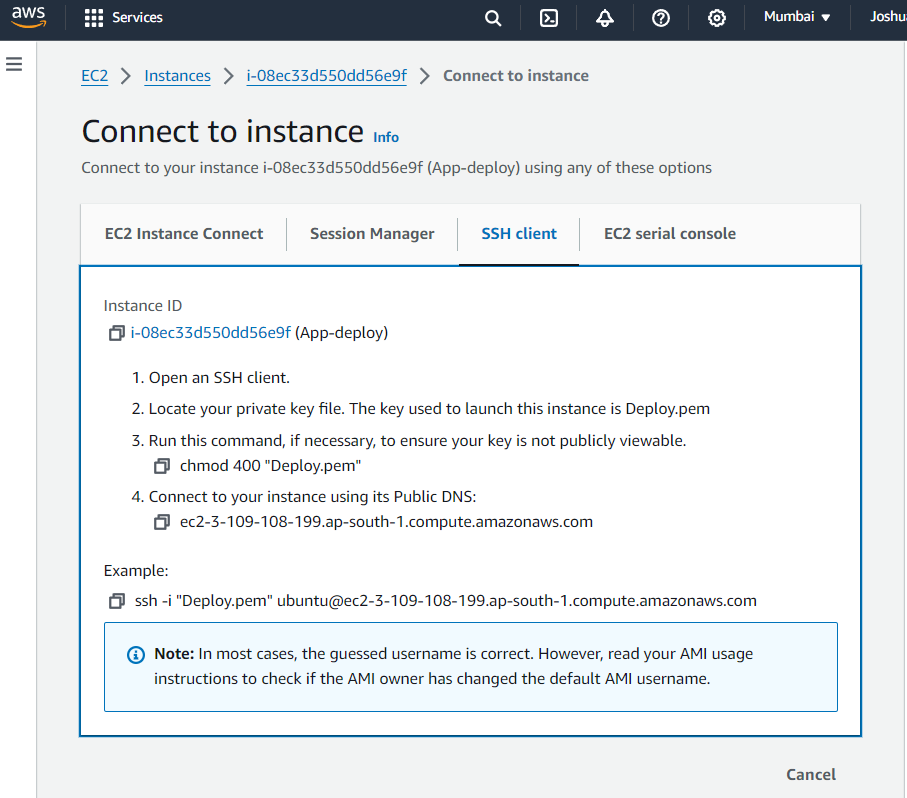
Then give launch instance



The instance will be launched successfully



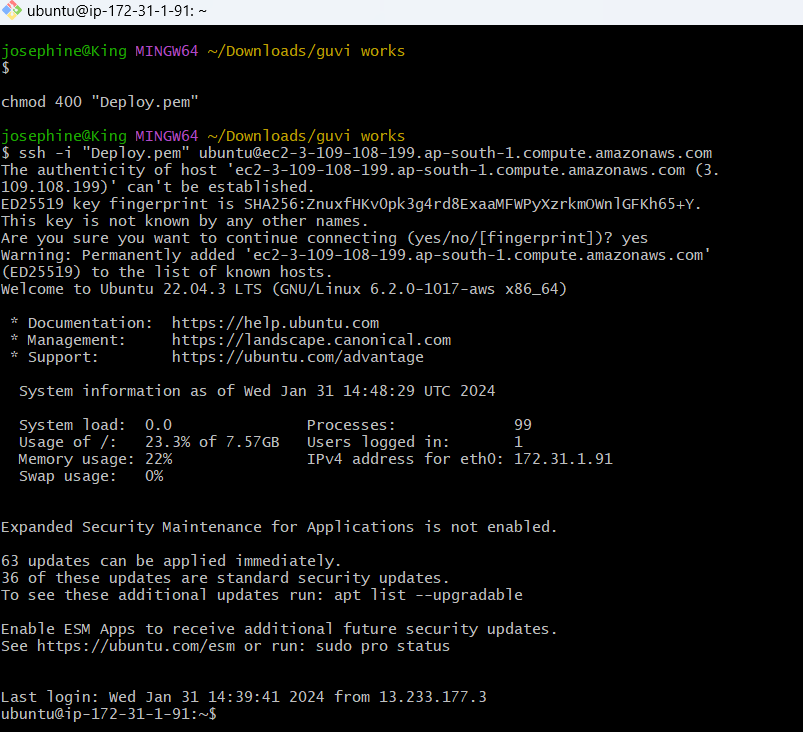
Now click on connect instance → go to ssh client



Now connect the instance into gitbash if you want (or proceed without pem key) by using the created pem key by giving following commands

chmod 400 "Deploy.pem"

ssh -i "Deploy.pem" ubuntu@ec2-3-109-108-199.ap-south-1.compute.amazonaws.com



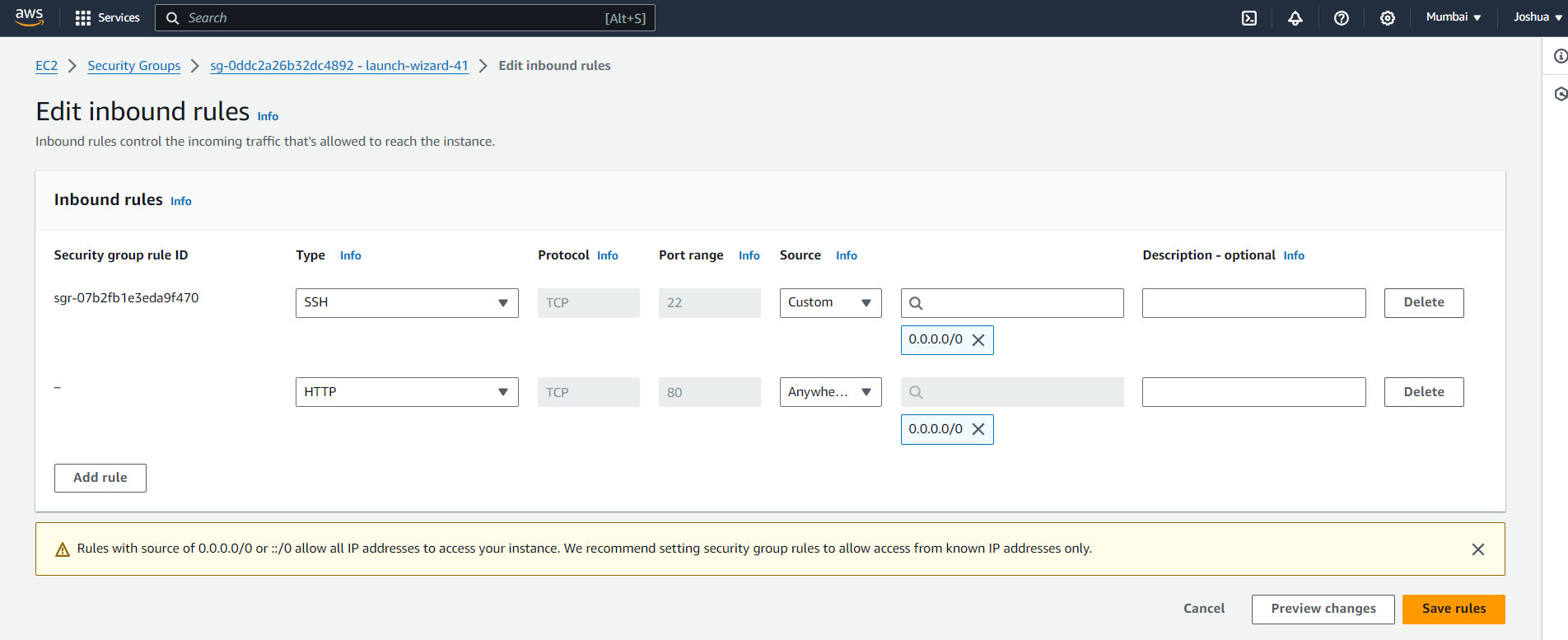
Update the instance by using sudo apt update

sudo apt install git

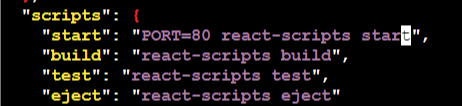
Clone the repo url using git clone <https://github.com/rvsp/reactjs-demo.git>

(i have renamed it into react-joz)

Now in the EC2 instance security settings → open security group →select Edit inbound rules →set port 80 →Save rules

.

Now go to the package.json file modify start script to set port



Now you can run the react app But in most cases it wont work

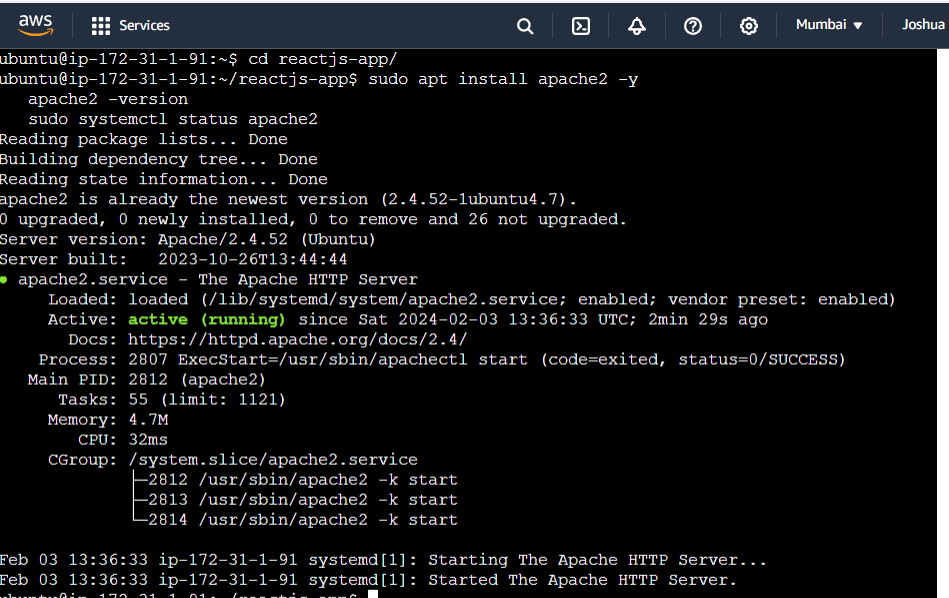
OR

**Install apache web server** using

sudo apt install apache2 -y

apache2 -version

sudo systemctl status apache2



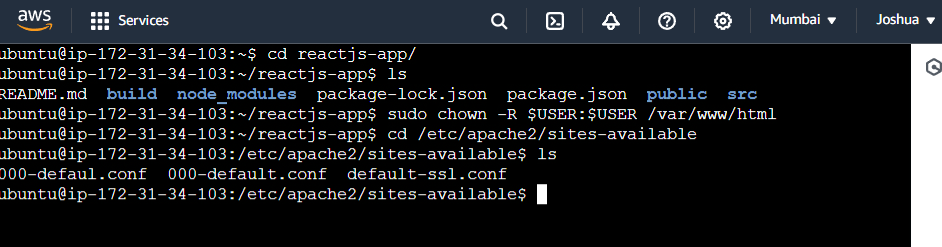
Now open the react-app by giving cd react-app→ Then list it using ls

Now copy the build folder containing the static files of the application into var/www/html directory to override the default configuration of apache. First grant user permission to that folder .Override the permission using ,

sudo chown -R $USER:$USER /var/www/html

To set the default port into required port open the below folder,

cd /etc/apache2/sites-available

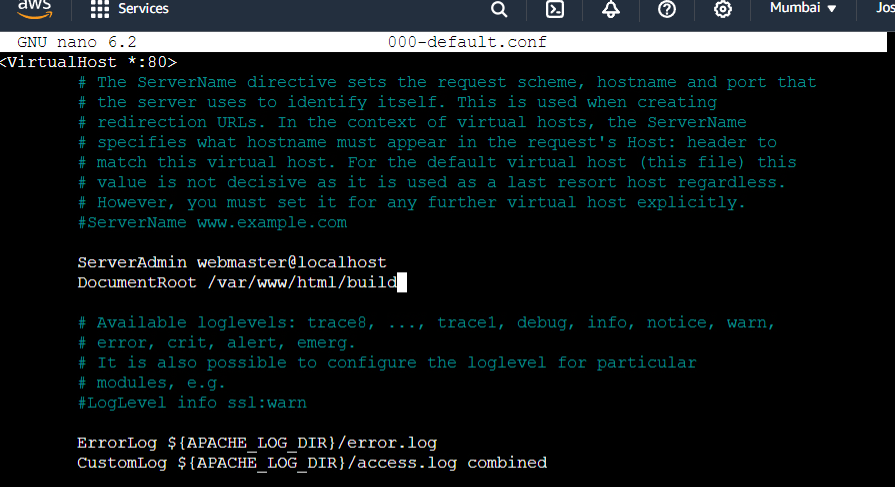


Now view the file inside it using ls command

Then to edit the file use,

sudo nano 000-default.conf

Add the build in the documentRoot as shown below

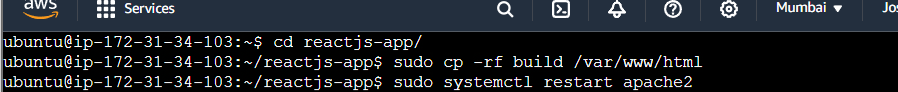


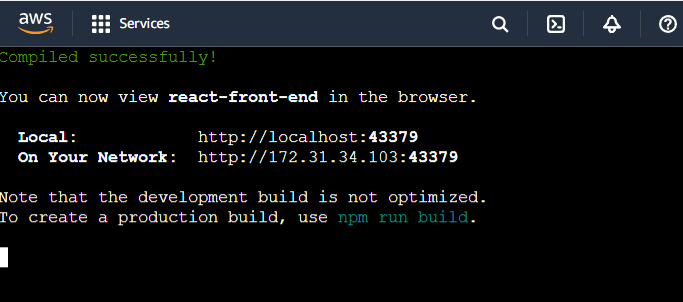
Now go to the react-app folder and copy the build to the apache2 by,

sudo cp -rf build /var/www/html

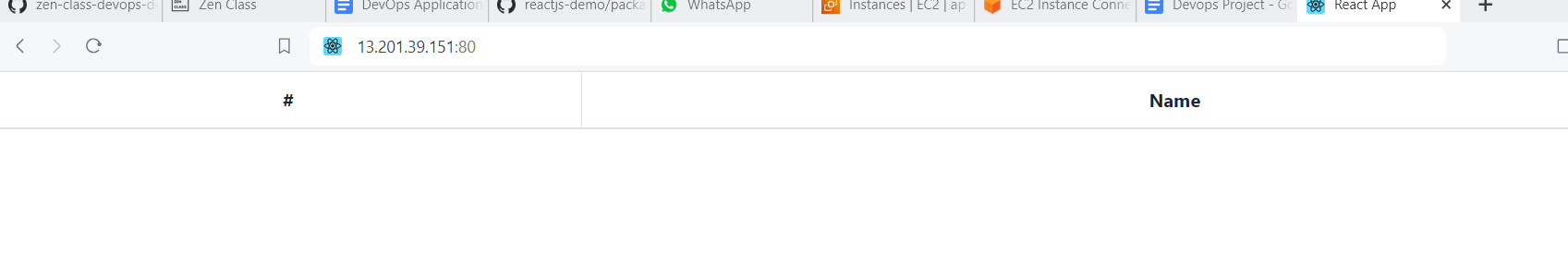
Restart the apache server by running the following command

sudo systemctl restart apache2



Then run the react app using sudo npm start

Now the app will be running in the public ip: 80 port



**Now install Docker and Docker-compose**

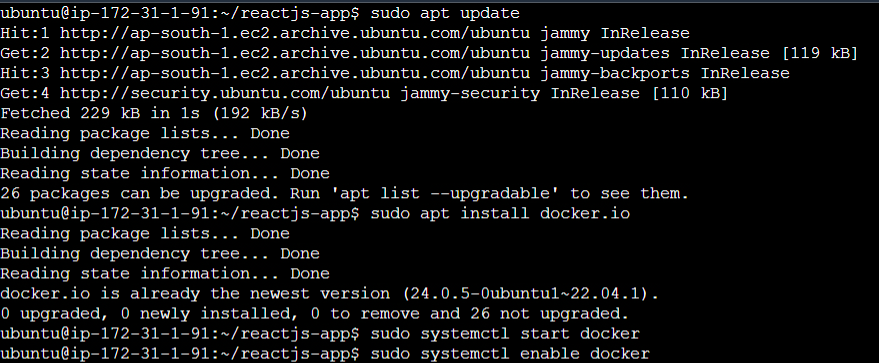
For ubuntu machine use the following command to install docker ,

sudo apt update

sudo apt install docker.io

sudo systemctl start docker

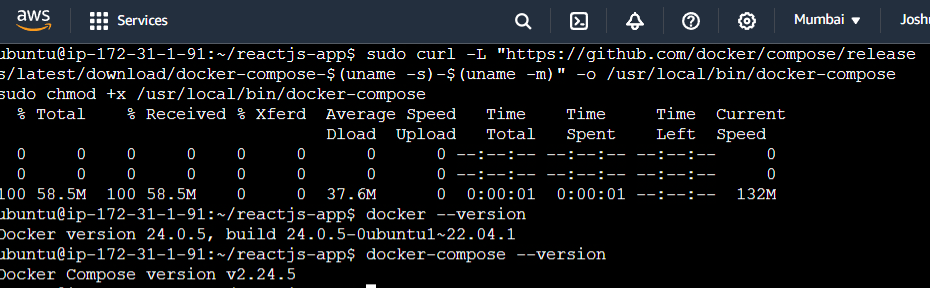
sudo systemctl enable docker



To install docker compose use,

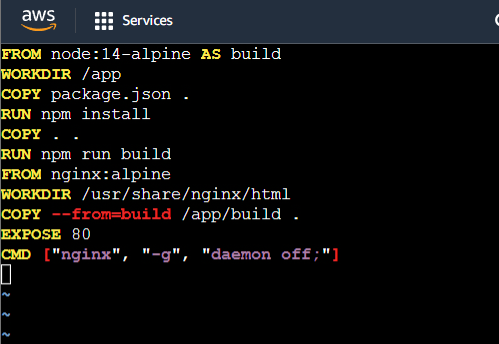
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose



**Now to Dockerize the application Create a Dockerfile by using,**

Vi Dockerfile and write the script



# set the base image and name the stage “builder”

FROM node:14-alpine AS builder

# set the working directory within the container

WORKDIR /app

# Copies packages into the working directory

COPY package\*.json ./

# Install node.js dependencies

RUN npm install

# copies the entire content of the current directory into working directory

COPY . .

# Builds the react application

RUN npm run build

# sets the base image to nginx with alpine linux

FROM nginx:alpine

# set the working directory within the container to nginx default html directory

WORKDIR /usr/share/nginx/html

# Copy the built React app from the "builder" stage into the current working directory.

COPY --from=builder /app/build .

# Expose port 80, saying that the container will listen on port 80

EXPOSE 80

# Specify the command to run when the container starts. In this it starts NGINX and runs it in the foreground

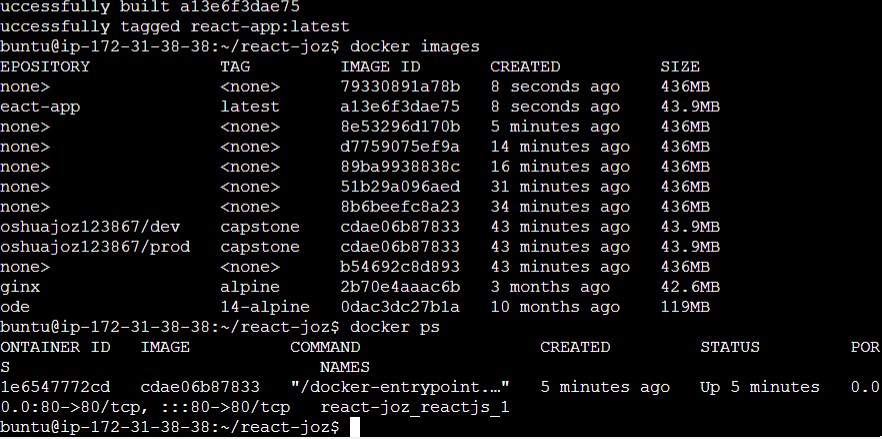
CMD ["nginx", "-g" , "daemon off;"]

Now build and run the Dockerfile by using,

docker build -t react-app .

Then run the container in port 80 by using,

docker run -dp 80:80 react-app



**Create a docker-compose file to use the above image,**

version: '3'

services:

Web:

# I have given github name

Image: Joshuaaj/dev

ports:

- "80:80"

Now in main branch it will be as

version: '3'

services:

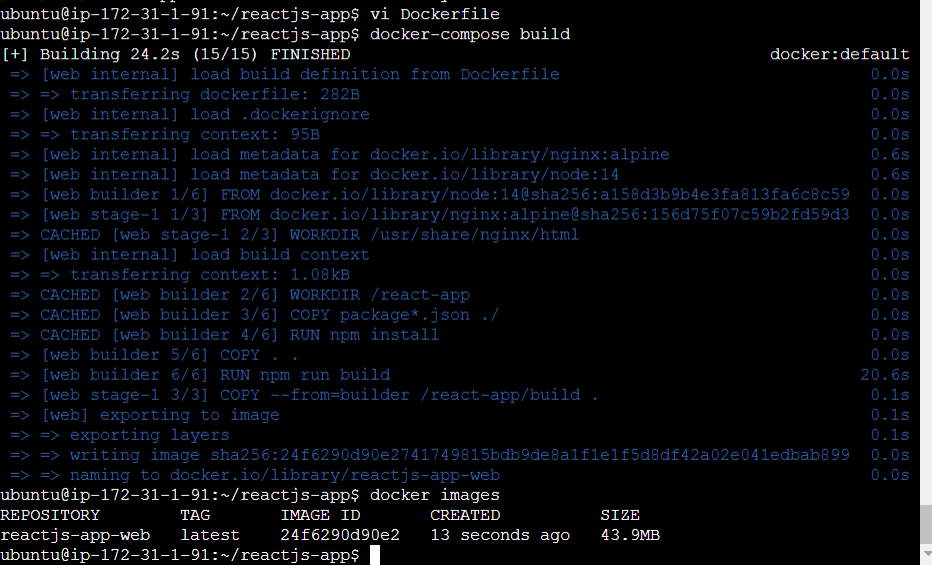
reactjs:

image: react-app

ports:

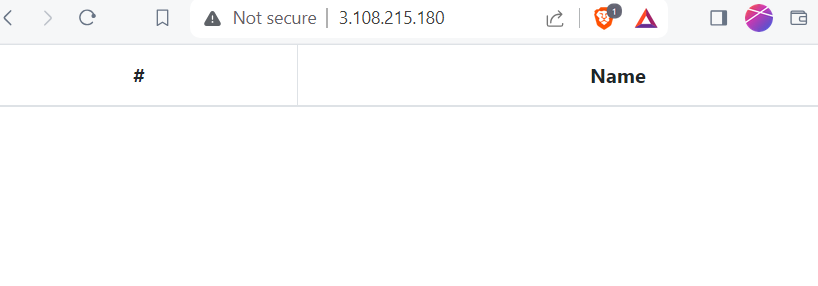
- "80:80"

Then run it using docker-compose up



To stop the container use docker-compose down

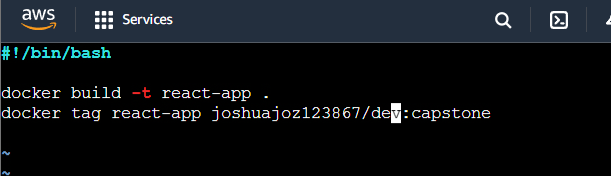
Now if we run the docker compose file the react application will be running in publicips: 3.108.215.180



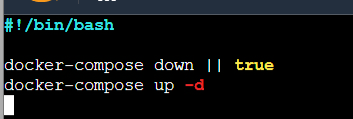
Now i am going to write bash script for building the docker images.so i am writing a bash.sh file as follow,

Create using touch build.sh

Open it using vi build.sh and now write the script as shown,



Then open deploy.sh file and write the script(i am doing it using my ec2 instance so i have given its ssh configs and running it)



Then build and run it using chmod +x build.sh deploy.sh

Then ./build.sh or ./deploy.sh

**Push the code to github to dev branch (use dockerignore & gitignore files)**

First initialize the git repository using

git init

Create a .ignore file

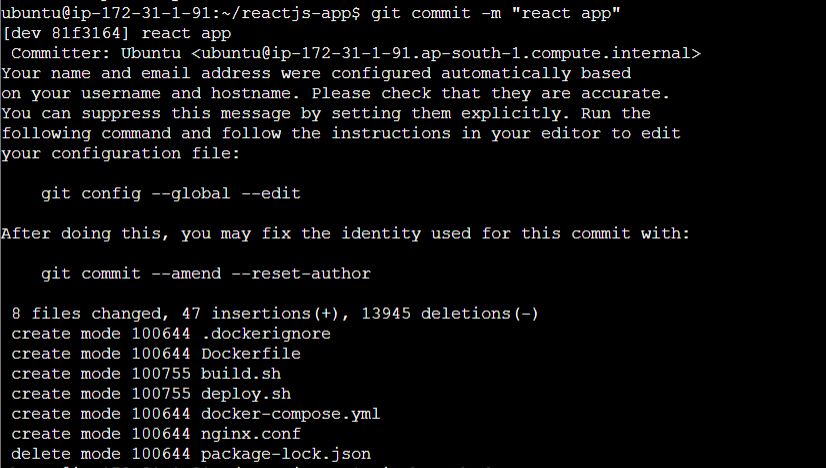
echo -e "node\_modules/\n.env\nbuild/" > .gitignore

To mention the file in the directory that is to be ignored

Add and commit changes using,

git add .

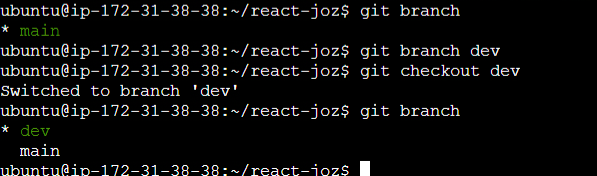
git commit -m “react app”



Now create a dev branch and switch into it using ,

git branch dev

git checkout dev

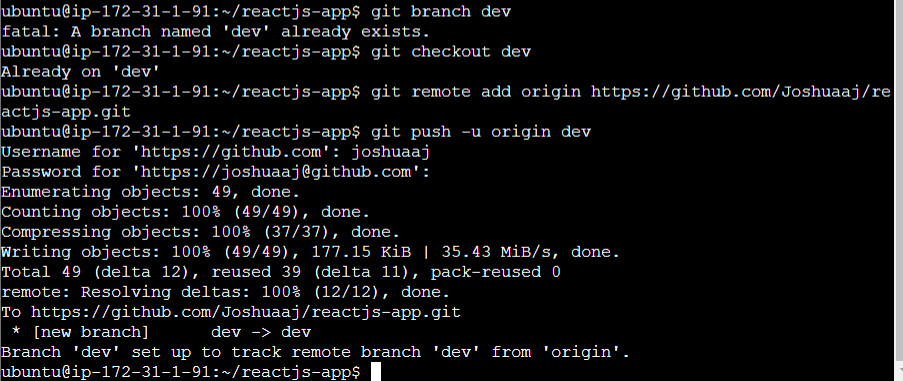


Push it to github using,

git remote add origin <https://github.com/Joshuaaj/reactjs-app.git>

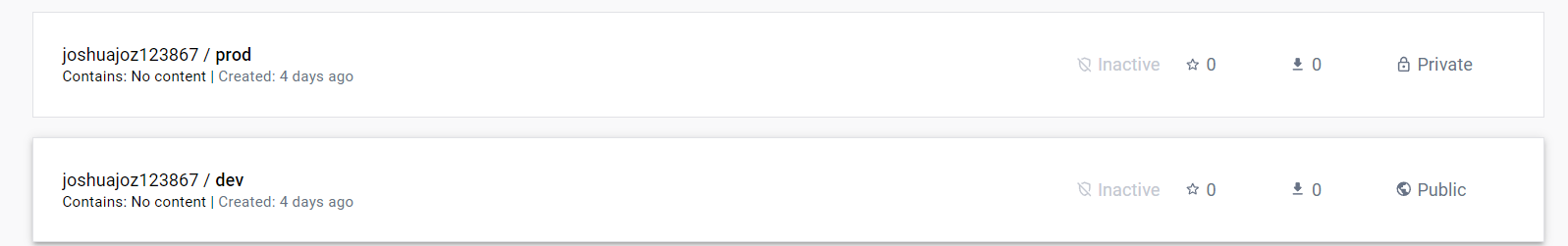
git push -u origin dev

Now the code will be pushed and it will be located in dev branch



**Creating dev and prod repo in docker hub**

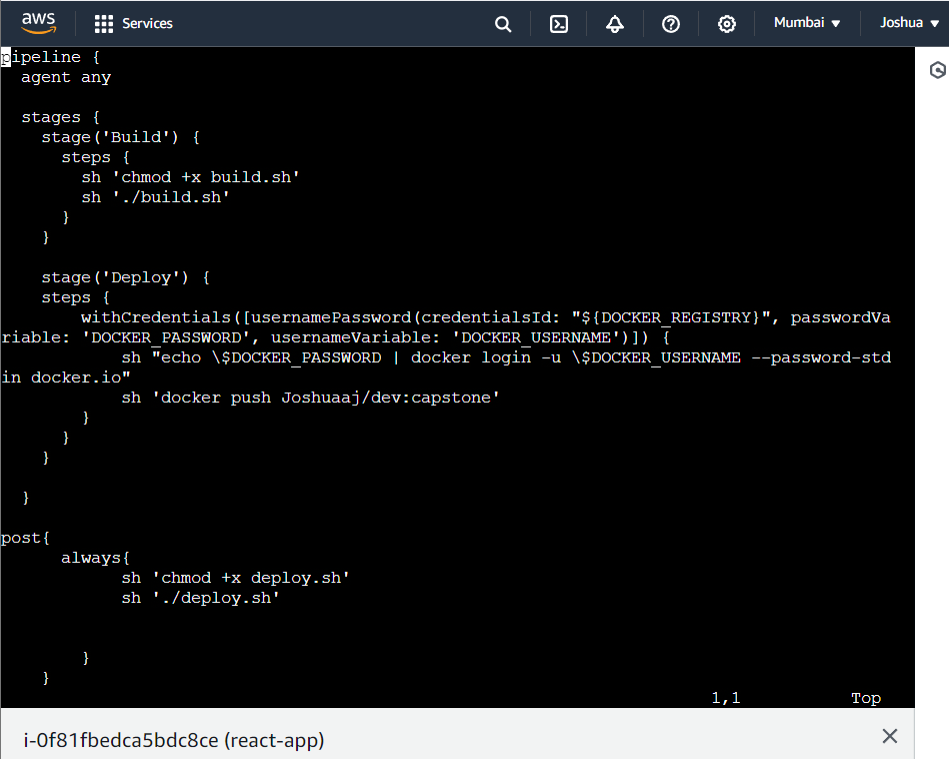
Login into your docker hub ,Click create repository

Give name as dev and set visibility as public and create another using prod name and set it as private.

**Jenkinsfile**

Write the jenkinsfile by ,

Vi Jenkinsfile

****

pipeline {

agent any

stages {

stage('Build') {

steps {

sh 'chmod +x build.sh'

sh './build.sh'

}

}

stage('Deploy') {

steps {

withCredentials([usernamePassword(credentialsId: "${DOCKER\_REGISTRY}", passwordVariable: 'DOCKER\_PASSWORD', usernameVariable: 'DOCKER\_USERNAME')]) {

sh "echo \$DOCKER\_PASSWORD | docker login -u \$DOCKER\_USERNAME --password-stdin docker.io"

sh 'docker push Joshuaaj/dev:capstone'

}

}

}

}

post{

always{

sh 'chmod +x deploy.sh'

sh './deploy.sh'

}

}

}

Push this into git and update the changes in both “prod” and “dev”

**Now install and configure Jenkins**

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

sudo apt install openjdk-11-jdk -y

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt update

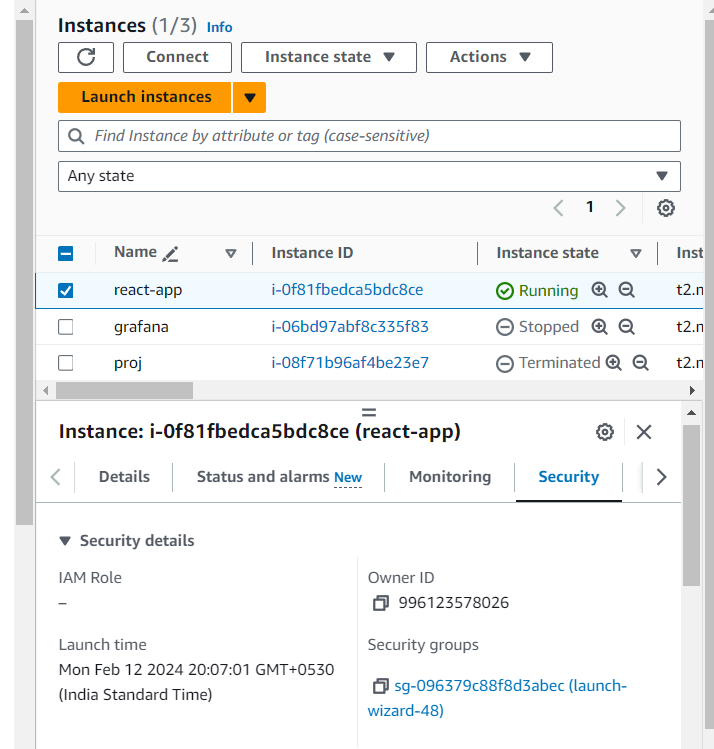
sudo apt install jenkins -y

Then check the version using

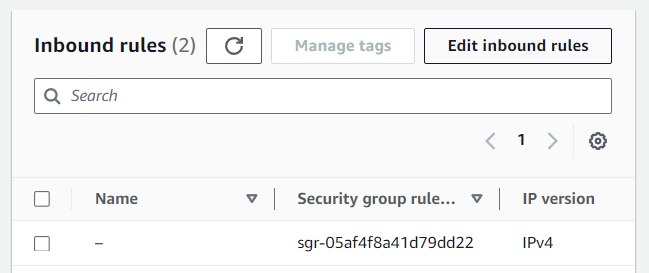
jenkins –version



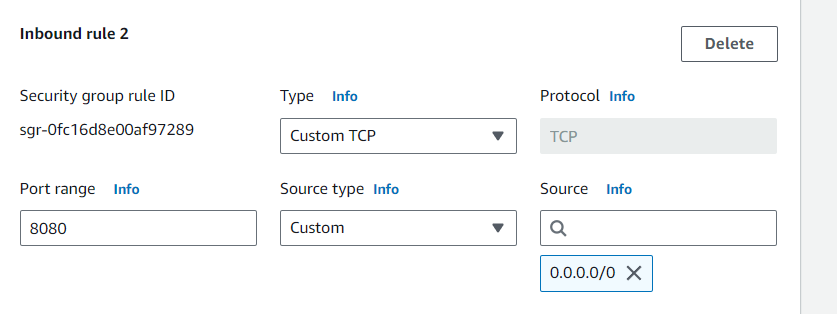
Then go to security group in your ec2 instance and set inbound rules and set 8080 port for jenkins.



Select edit inbound rule,



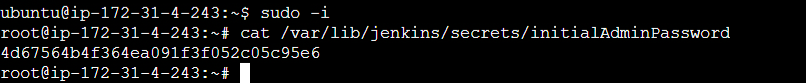
Allow 8080 port and save rules.



Now login to jenkins using public ip of the created ec2 instance and port

<http://3.110.143.145:8080>

Search this in the ec2 instance

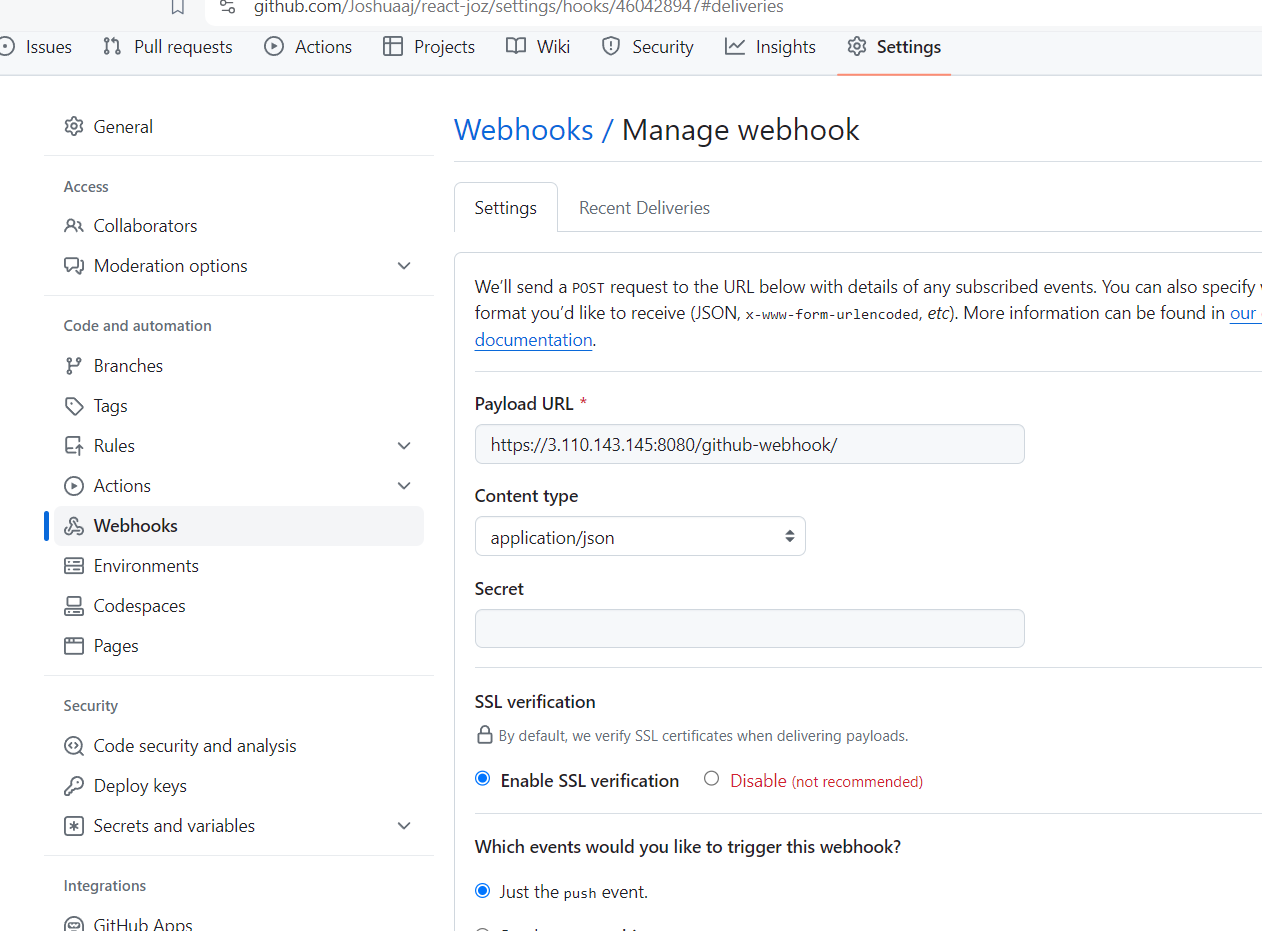


Give continue it will ask for suggested plugins select it.

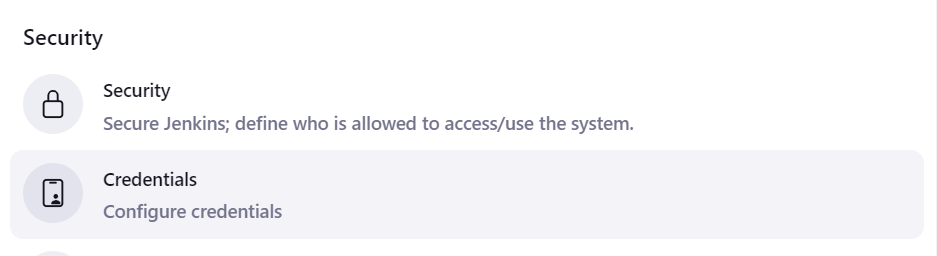
Then give username and password then login into jenkins and download the required plugins like github,docker and docker integration etc.

Then setup required credentials,

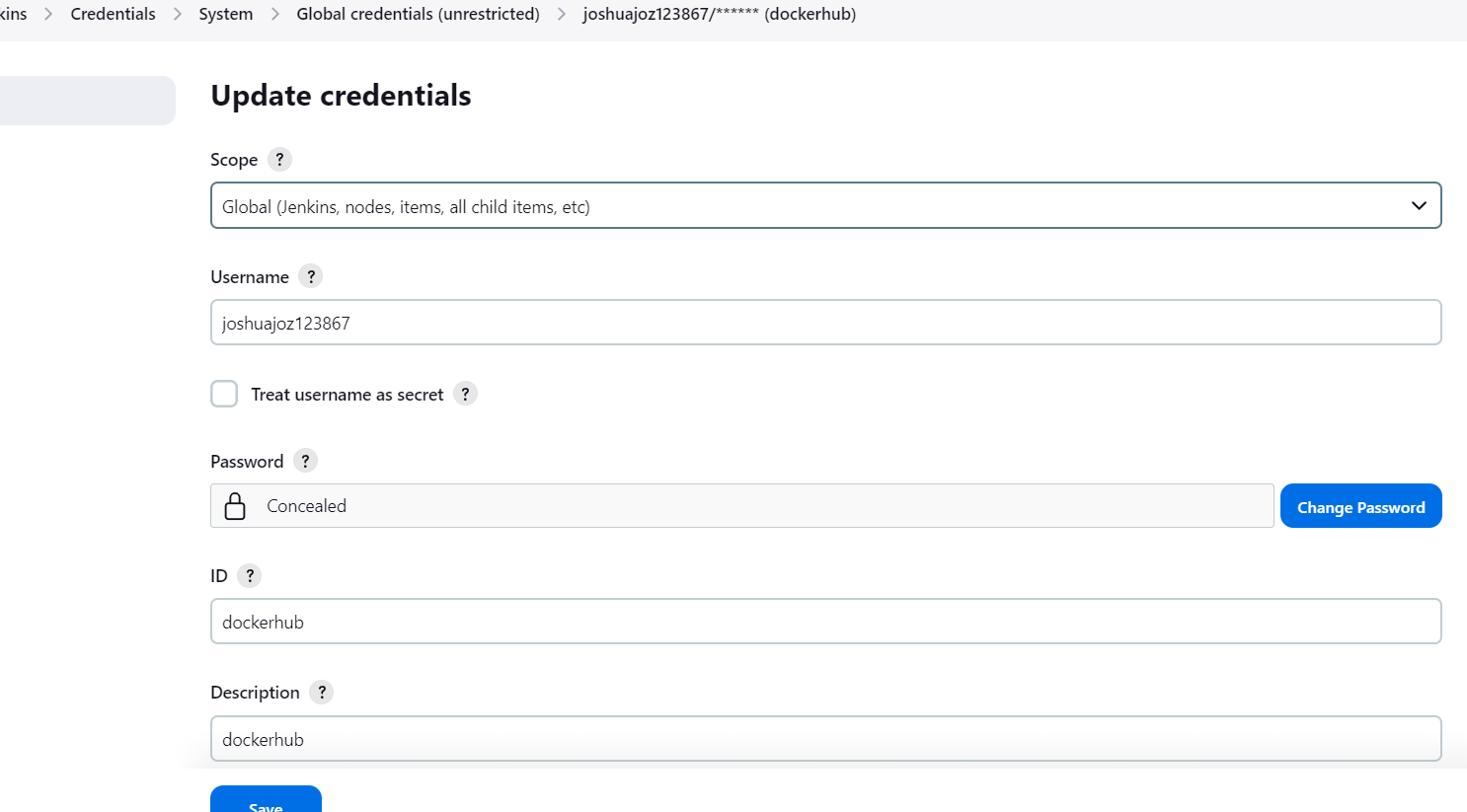
Now in git hub

Open your repository and select settings select webhook and do as shown ,

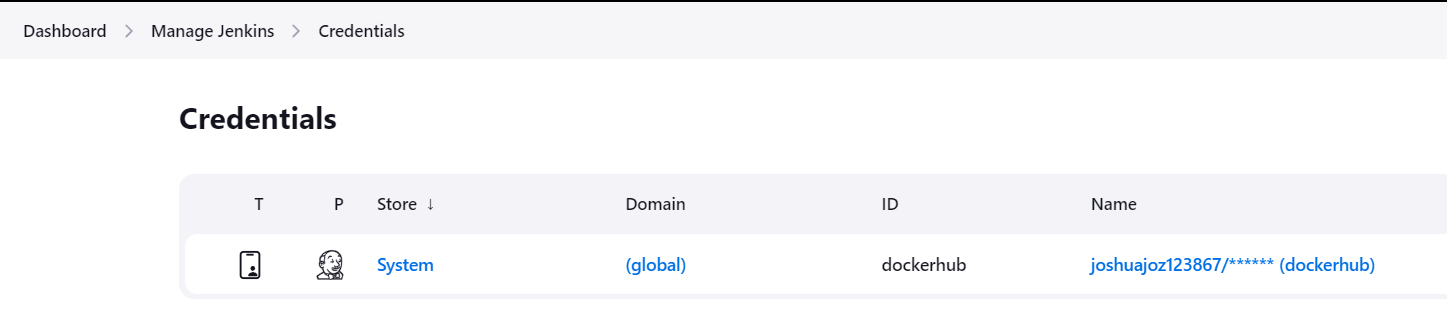
Go to manage jenkins



Update the credentials

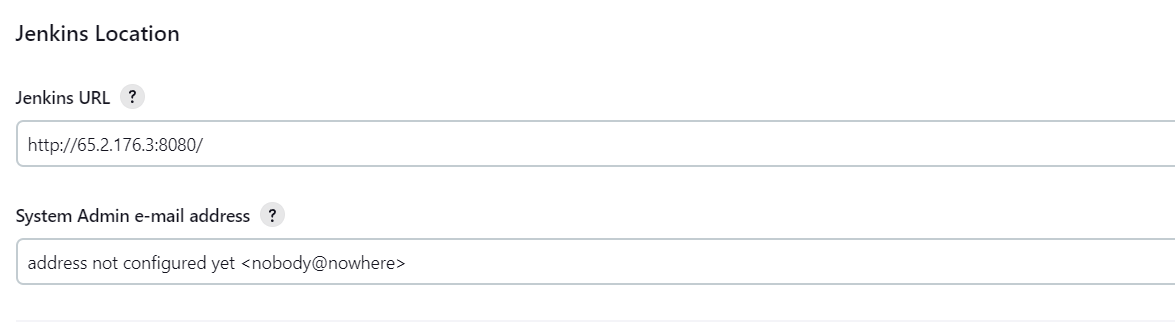


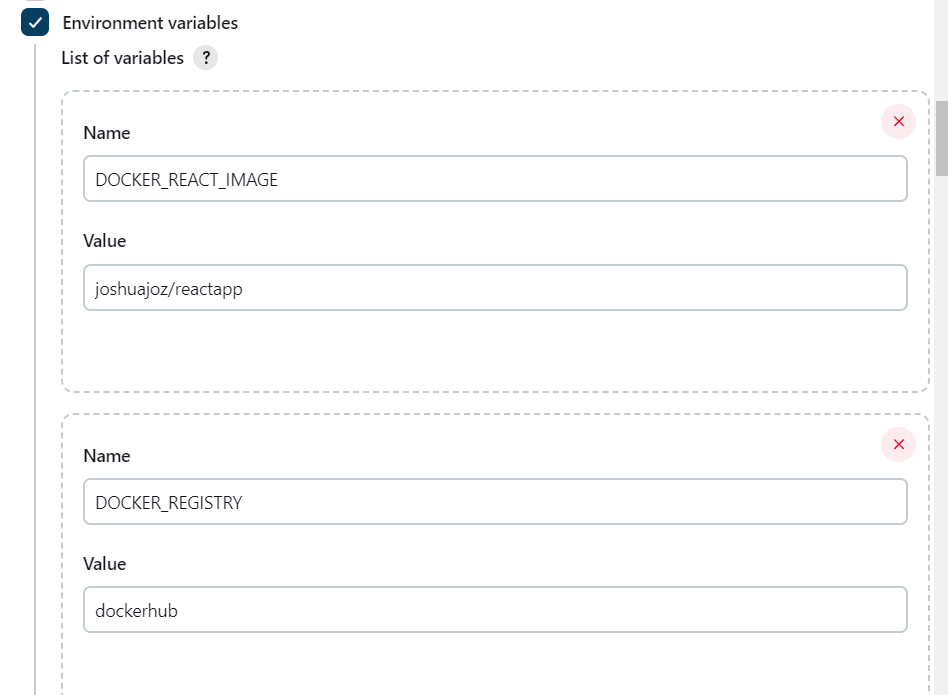
And it will be as shown below



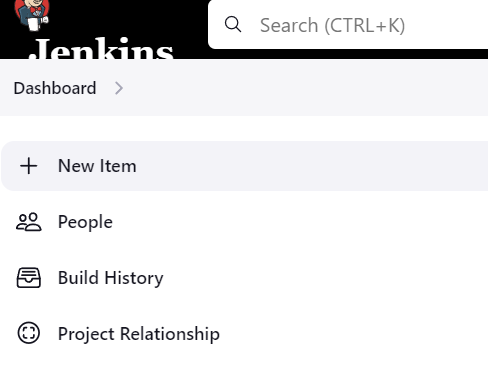
Go to system in manage jenkins.







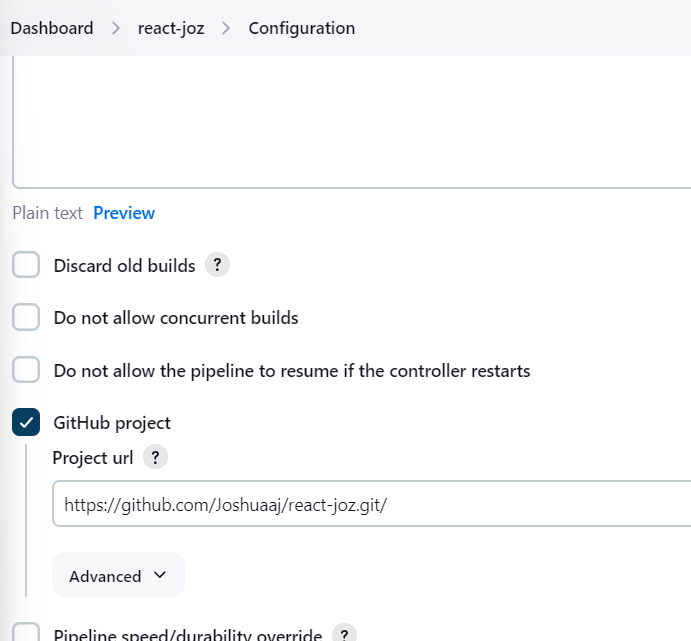
Now go to dashboard and select new item



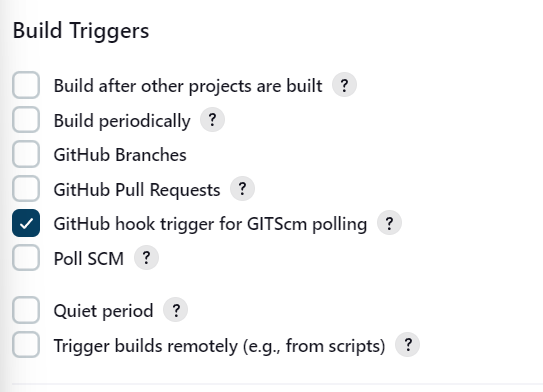
Create a pipeline by giving name and select pipeline.

And do the configuration as required,

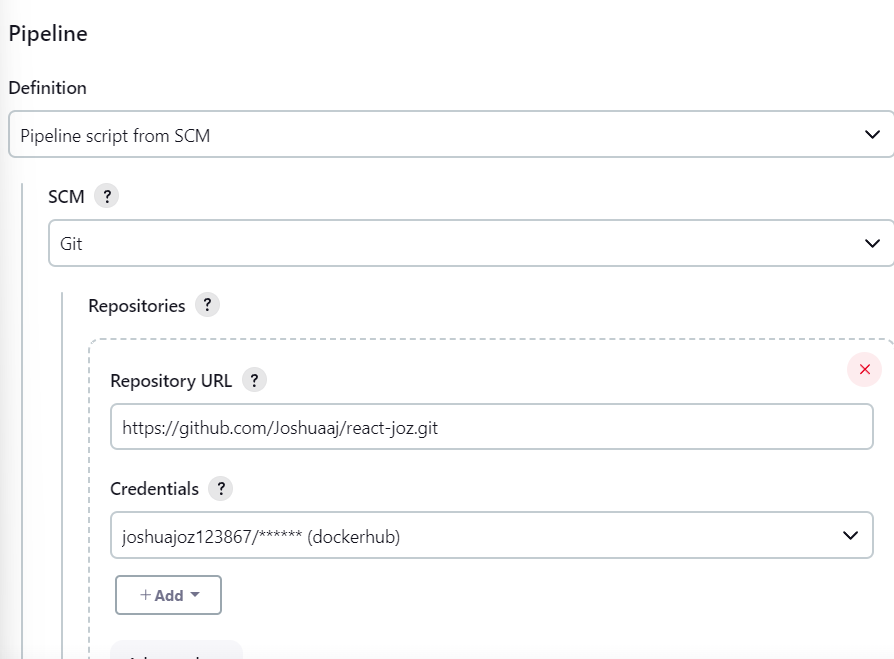
Give the github url



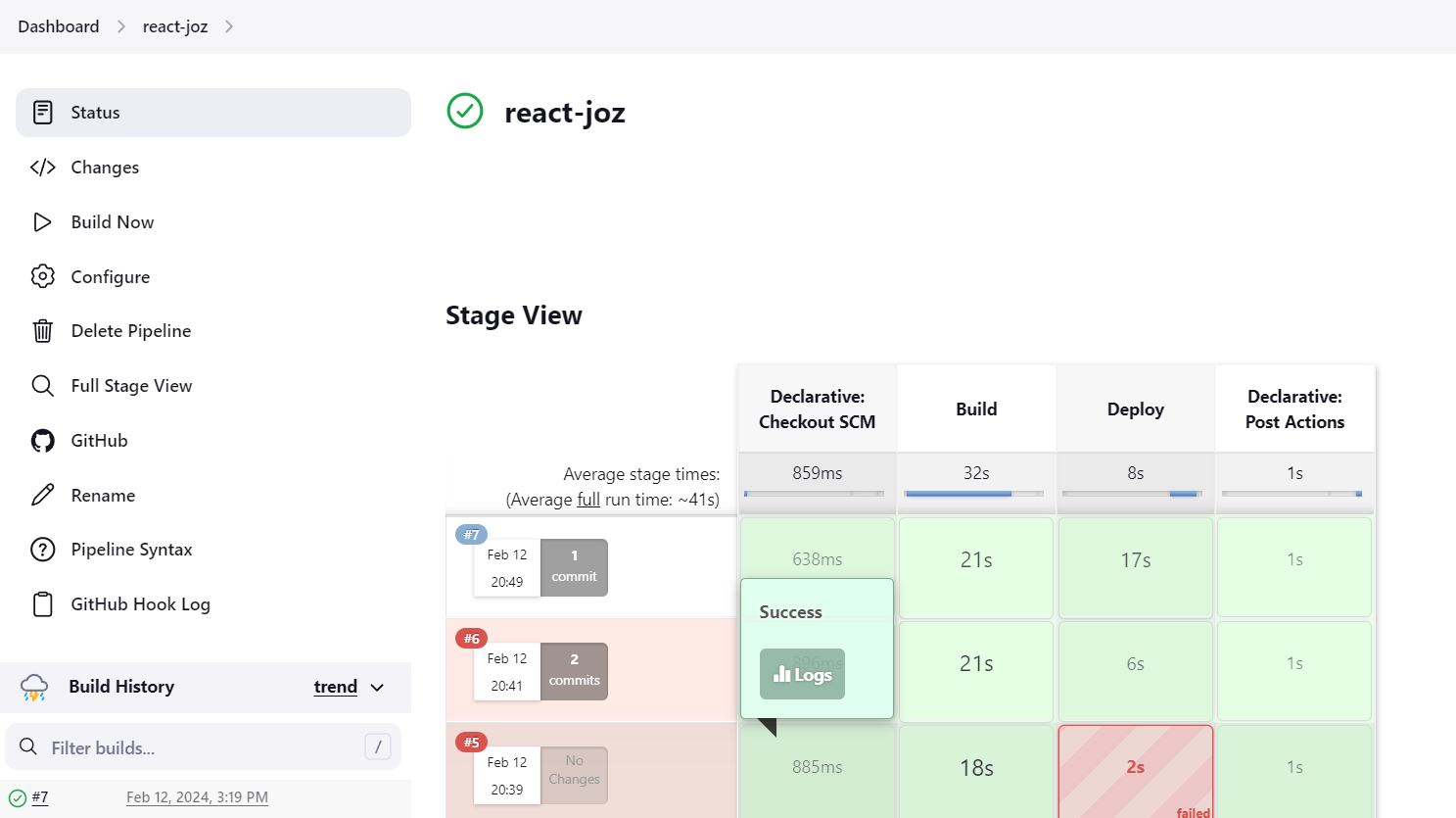
Set build trigger,



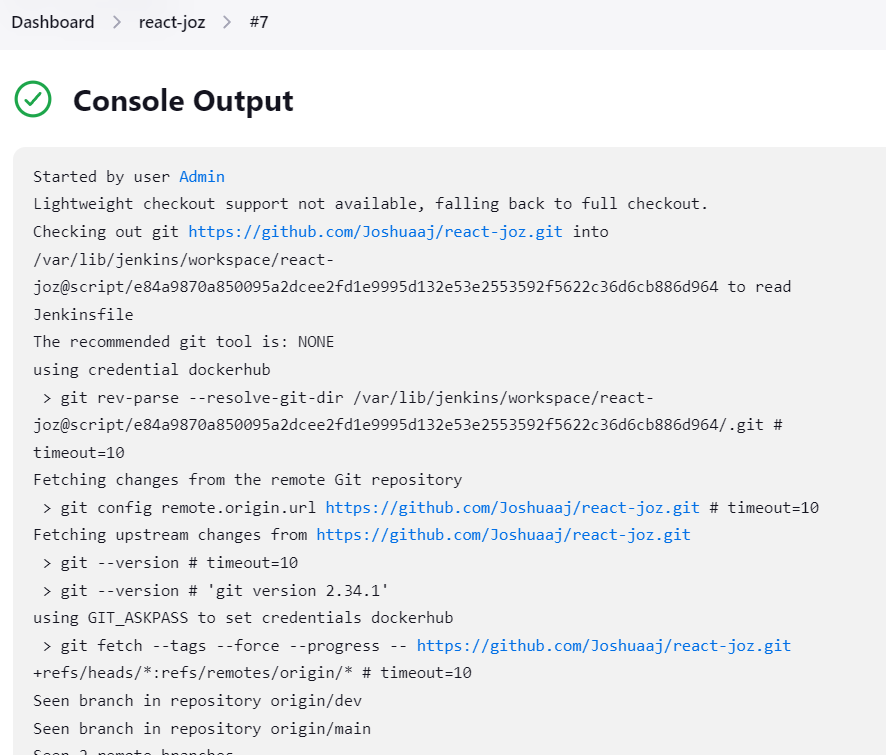
In pipeline in SCM choose git and give github repo url,

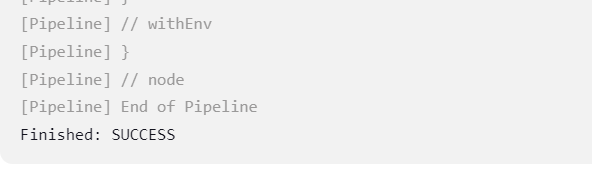


Build and run the application

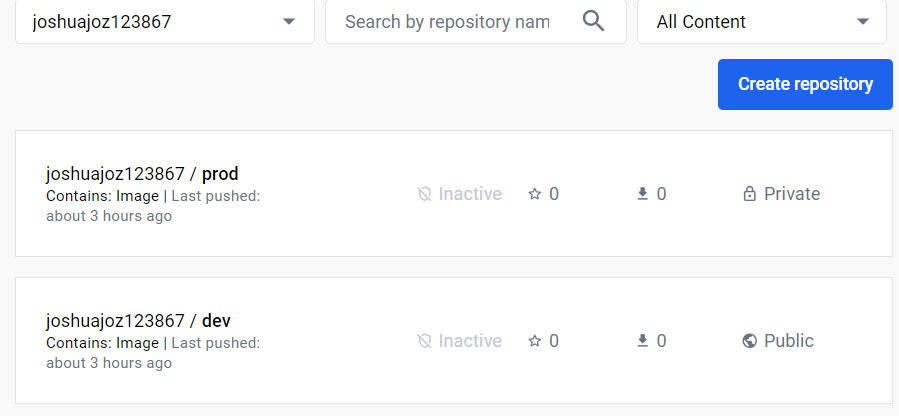


Console output will be as shown





**Now the build images would pushed to docker hub.**



**Now in aws t2 micro instance launch and deploy the react app**

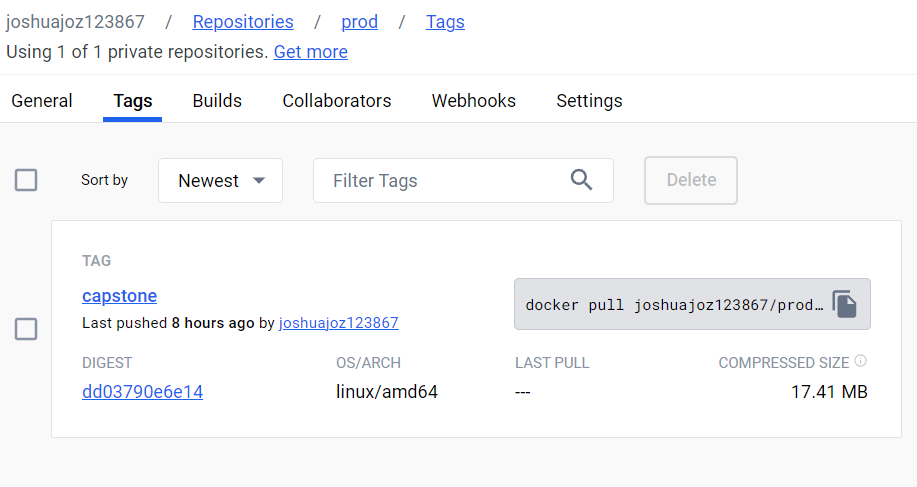
Create an ec2 instance with the required credential

Connect the instance

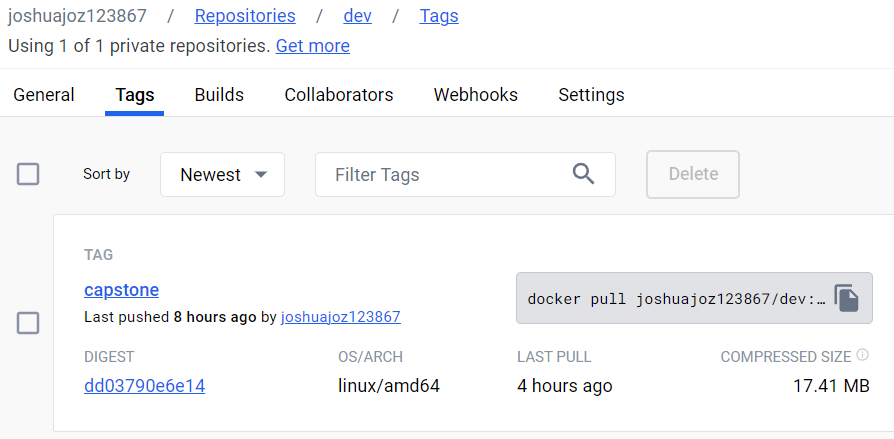
Give sudo apt-update

Install docker and login docker

Pull the image from prod

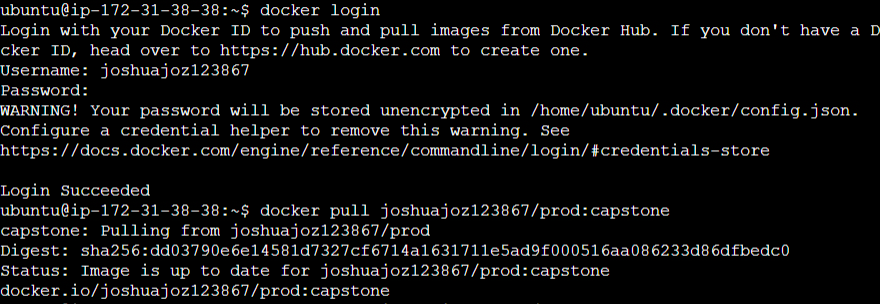
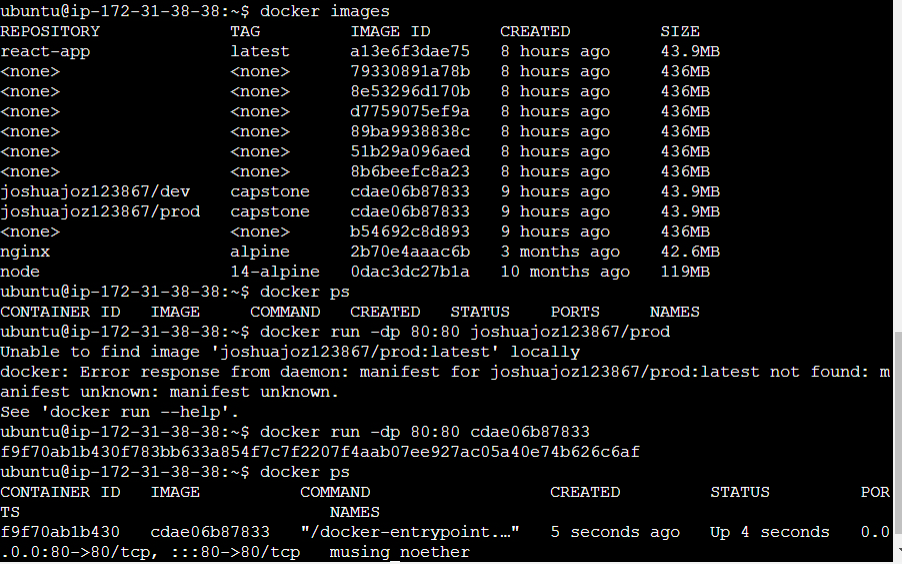


The repo in dev will be as shown below,

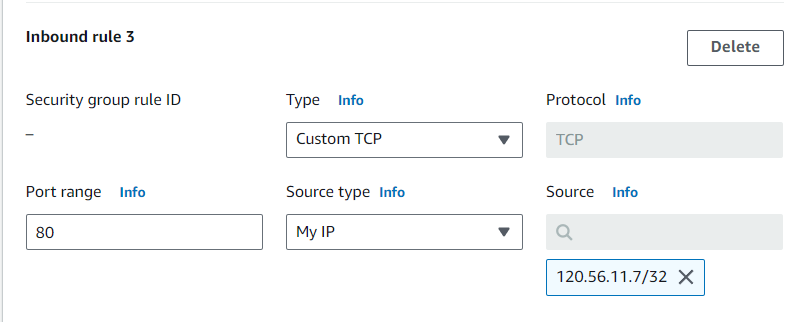


Give docker pull joshuajoz123867/prod:capstone

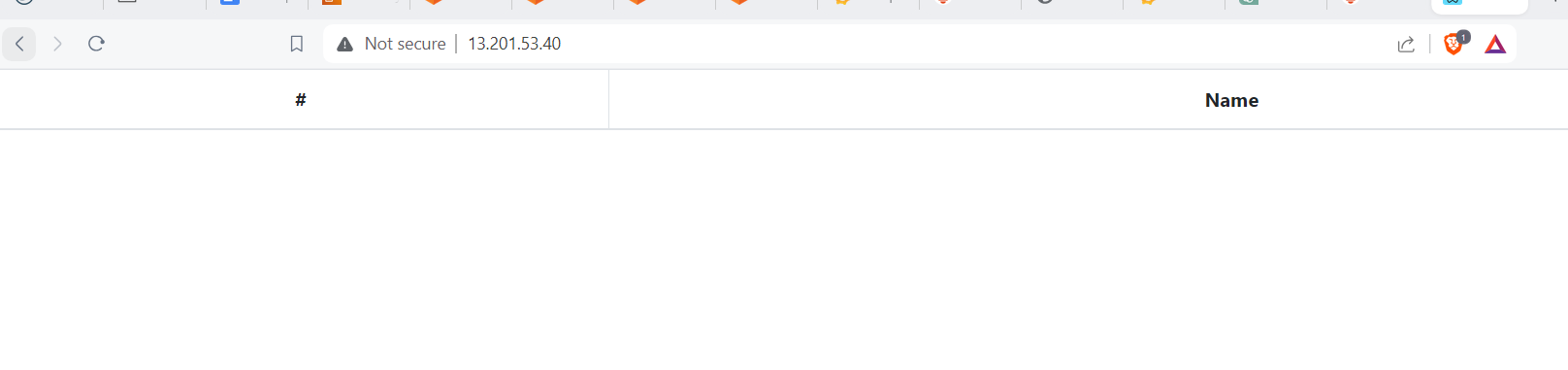
docker run -dp 80:80 containerid



Set port in the instance security →edit inbound rules



Now the react app will be running



Grafana monitoring

Install prometheus using,( <https://devops4solutions.com/monitoring-using-prometheus-and-grafana-on-aws-ec2/> )

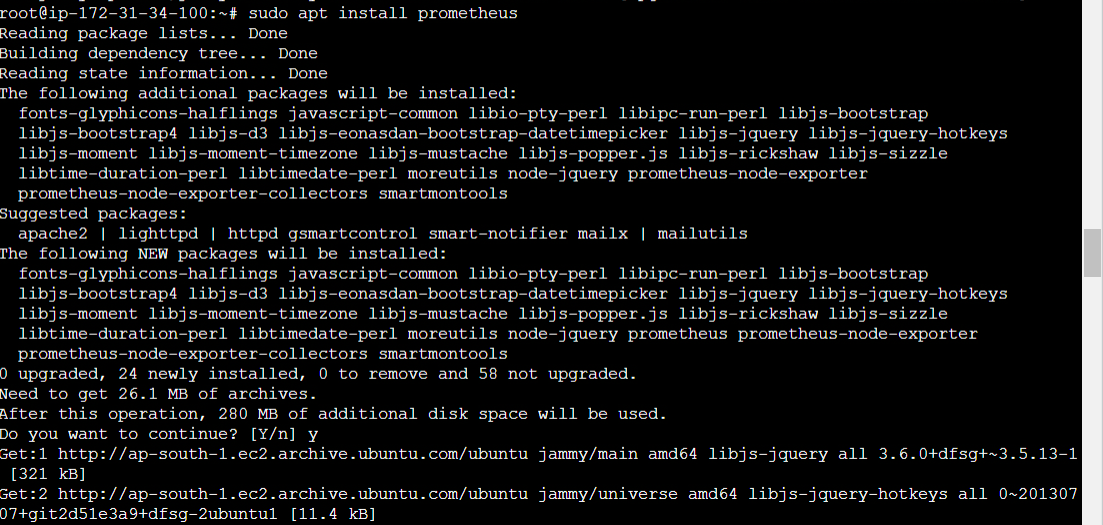
sudo apt-get install -y gnupg2 curl software-properties-common

curl -fsSL https://packages.grafana.com/gpg.key | sudo apt-key add -

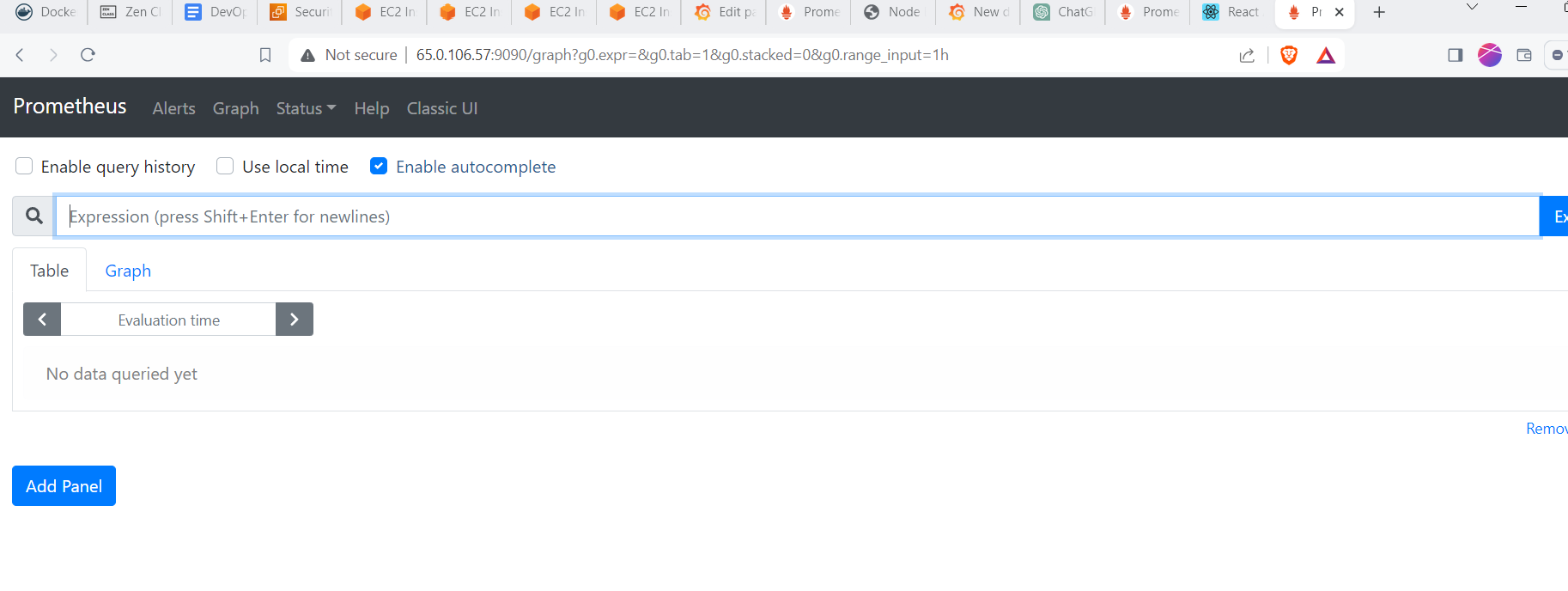
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"

Then ,sudo apt update

Sudo apt install prometheus







Install node exporter using,

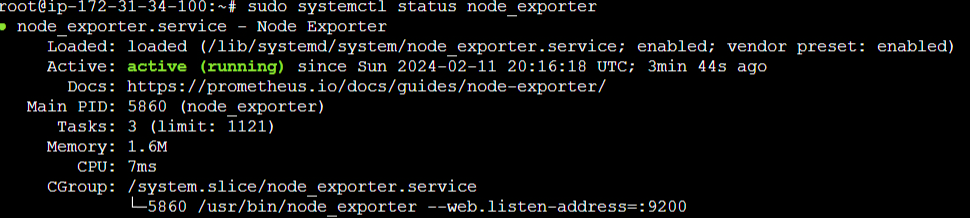
sudo useradd --no-create-home --shell /bin/false node\_exporter

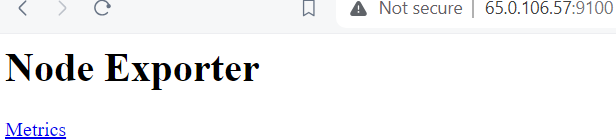
wget https://github.com/prometheus/node\_exporter/releases/download/v2.3.2/node\_exporter-2.3.2.linux-amd64.tar.gz

tar xvfz node\_exporter-2.3.2.linux-amd64.tar.gz

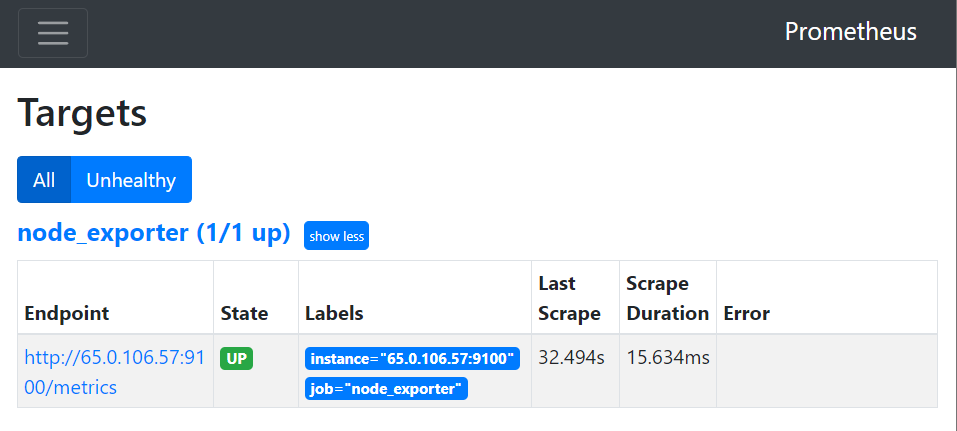
cd node\_exporter-2.3.2.linux-amd64

sudo cp node\_exporter /usr/local/bin





**Target**



Now install grafana using,

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

sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"

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

sudo apt-get update

sudo apt-get install -y grafana

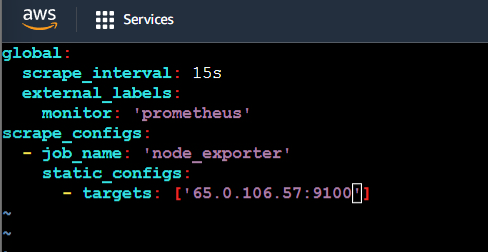
sudo systemctl start grafana-server

sudo systemctl enable grafana-server

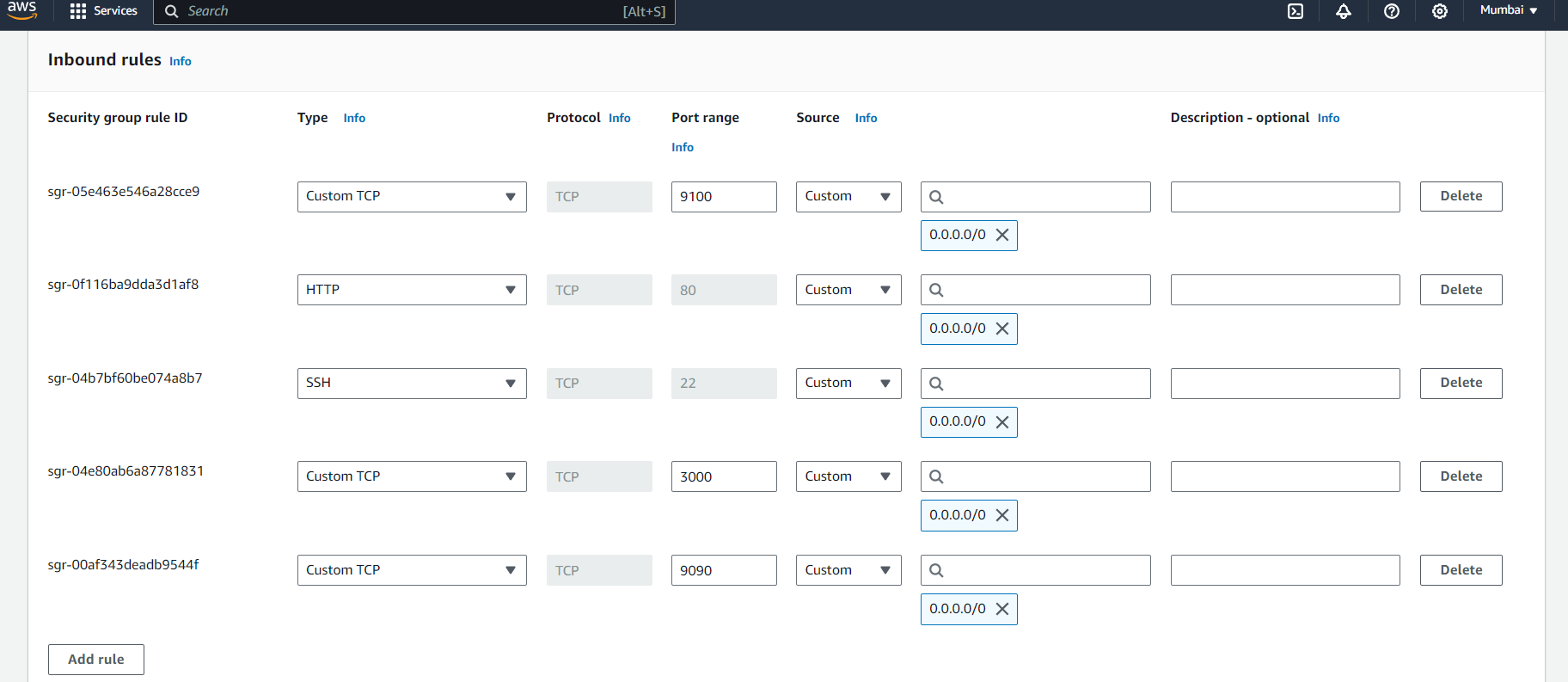


Configure prometheus for node using,

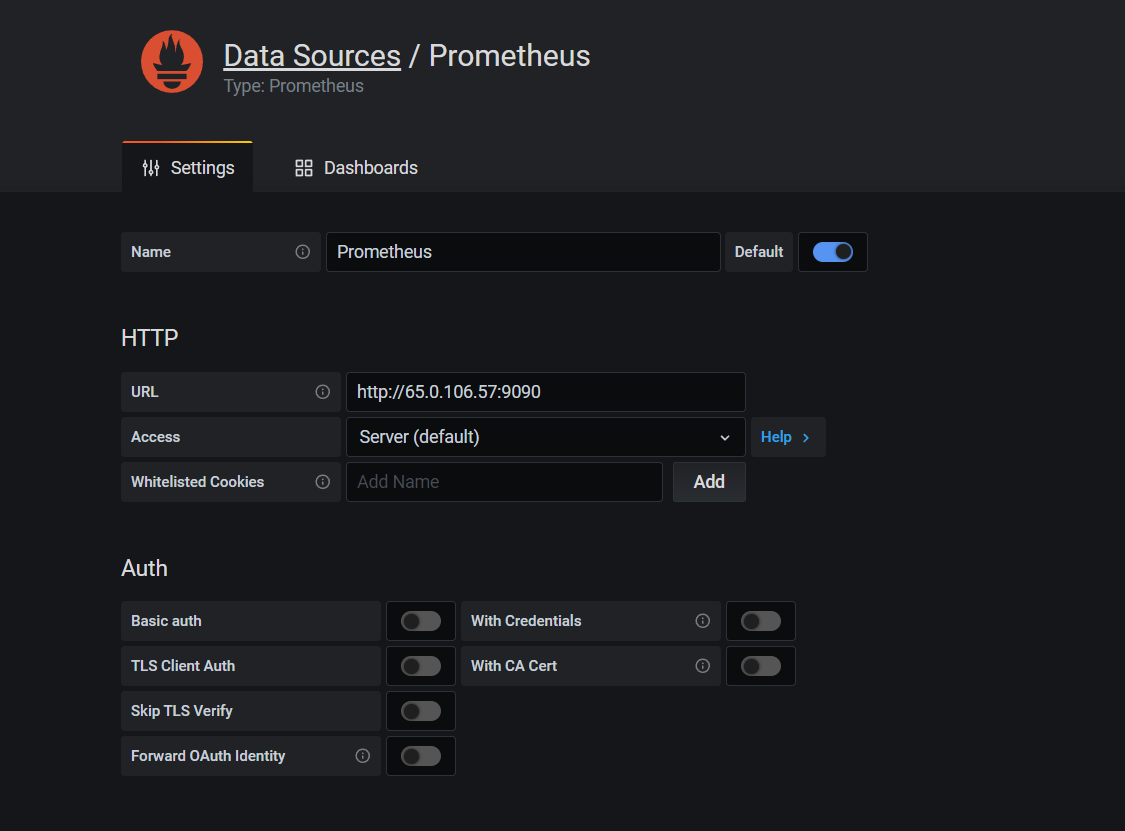
sudo vi /etc/prometheus/prometheus.yml



Set inbound rules in security of your instance



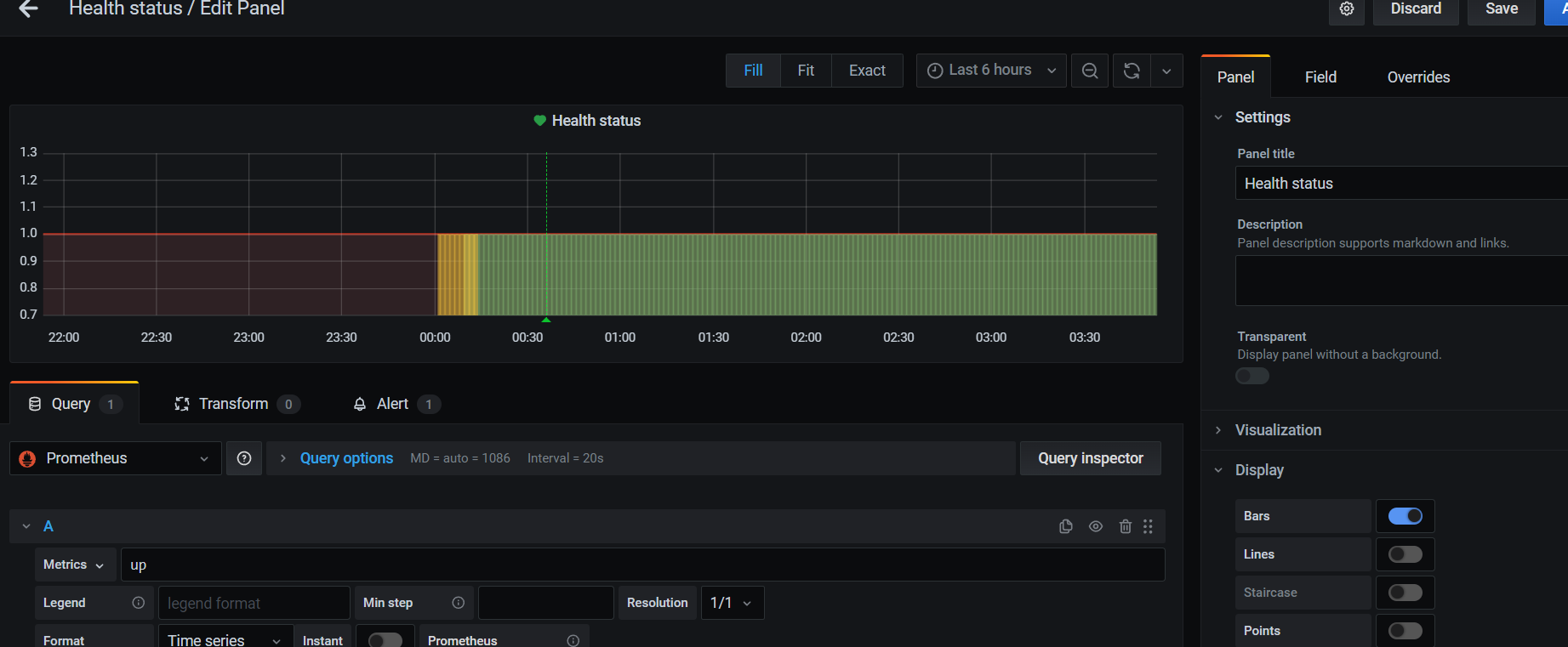
Now open your grafana and go to configuration and add data source



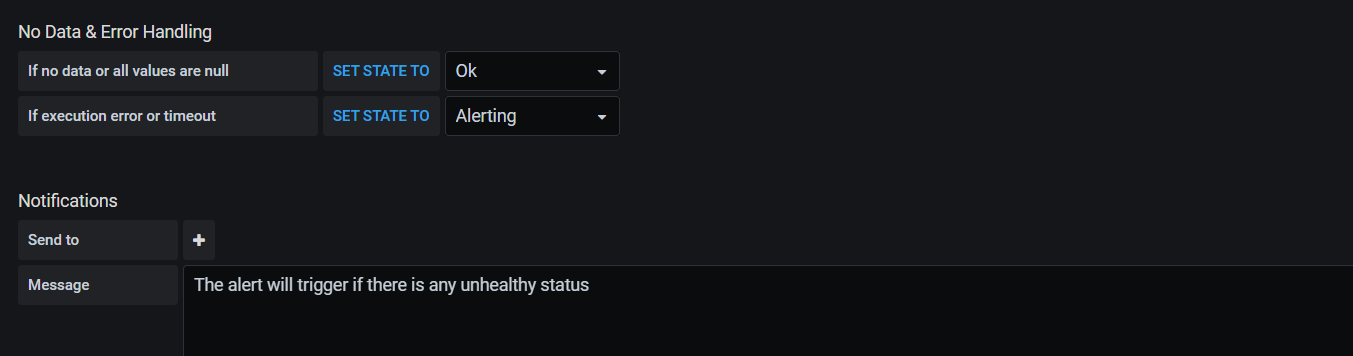
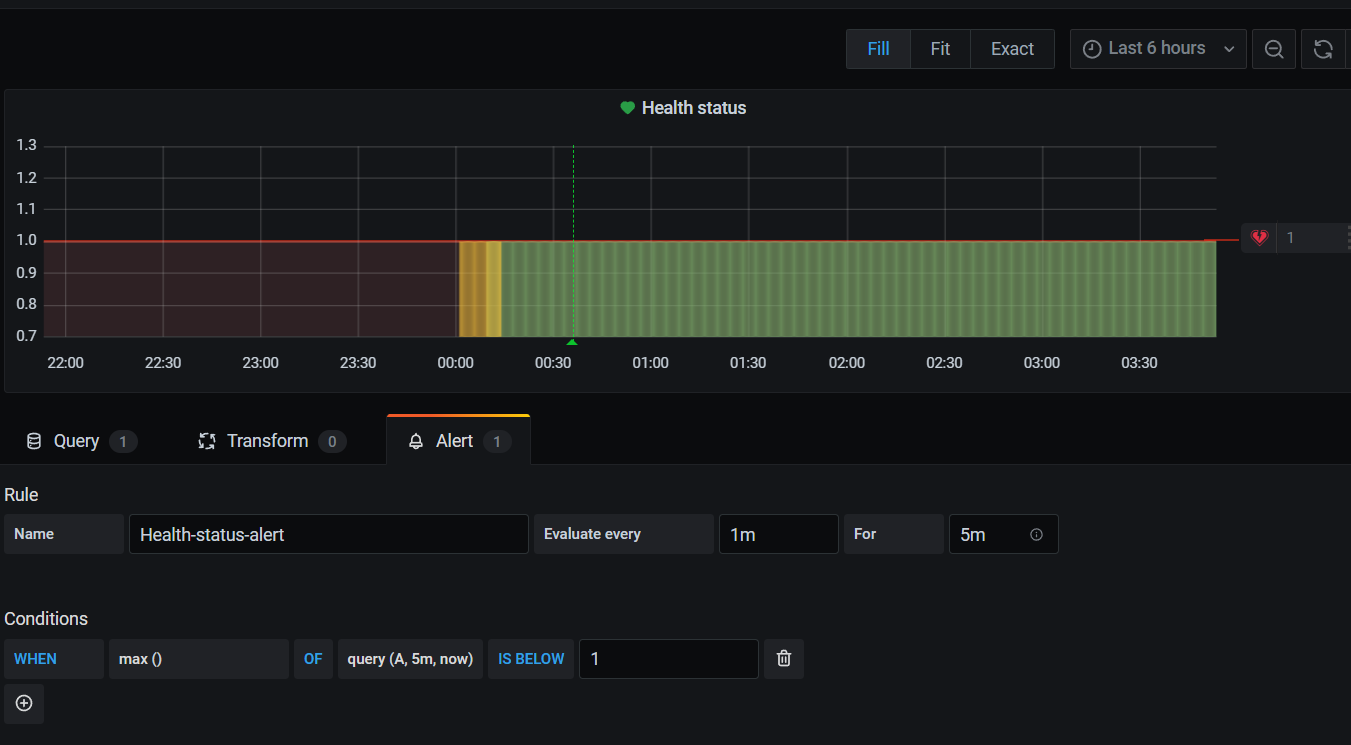


Now go to dashboard and add new dashboard

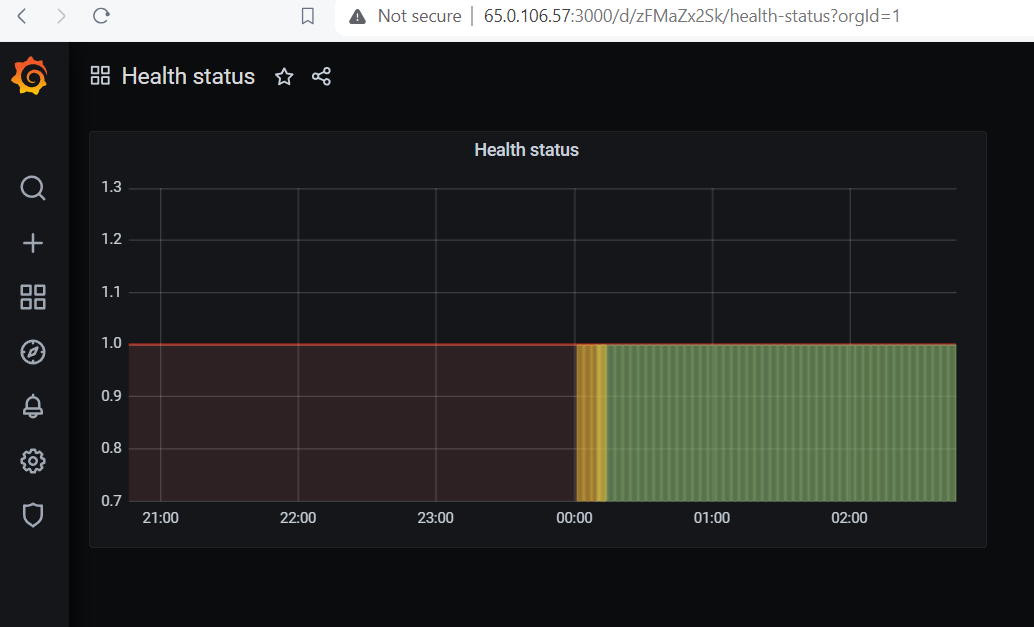
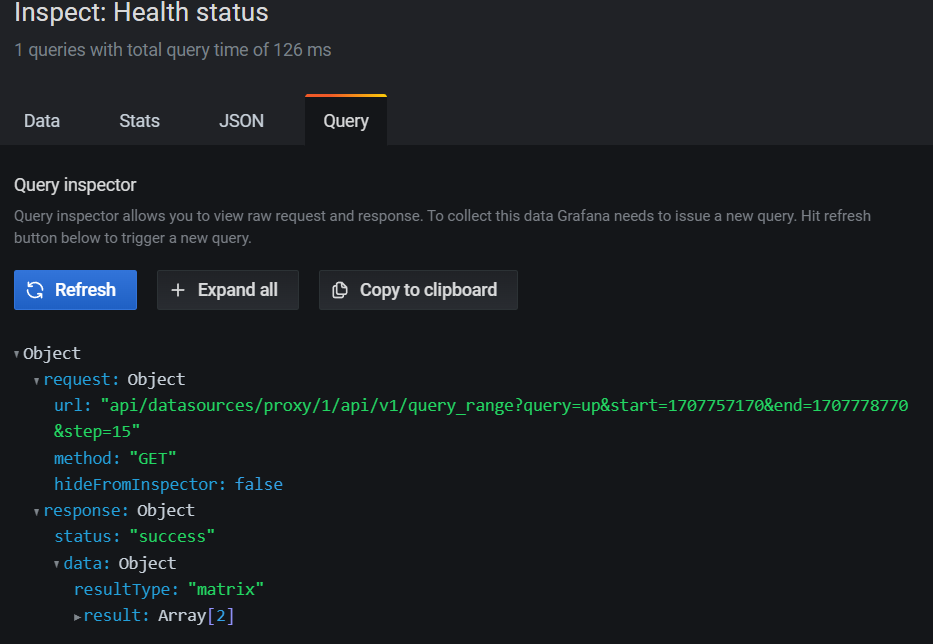
I have given name as health status



Created an alert



Output will be as shown below



Done.