**Application Deployment**

(Deploy the given React application to a production -ready)

**Step-1**

Create fork of the given repo: sriram-R-krishnan/devops-build

A screenshot of a search box

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Now clone the github repository,

gitclone https://github.com/SureshV1994/devops-build

A black screen with white text

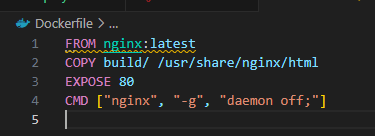
AI-generated content may be incorrect.

cd devops-build

**Step-2:** Create Dockerfile and Dockercompose.yaml file

**Dockerfile**

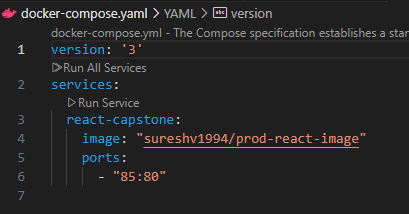
|  |
| --- |
| FROM nginx:latest  COPY build /usr/share/nginx/html  EXPOSE 80  CMD [ "nginx", "-g", "daemon off;" ] |



* The -g, is used to override the default nginx configuration.
* Normally nginx will run in the background (daemon mode) without docker , but since we are using in docker , the container must keep running, so we are keeping the nginx running in the foreground to keep the container alive.

**docker-compose.yaml**

|  |
| --- |
| version: '3'  services:  react-capstone:  image: "sureshv1994/dev-react-image"  ports:  - "85:80" |



**Step-3:** create build.sh (for building the image and pushing it to concerned docker hub ) deploy.sh (for pulling the pushed image from concerned docker hub and deploying it to aws)

build.sh

|  |
| --- |
| #!bin/bash  echo “Current Branch: $BRANCH\_NAME”  if [ “$BRANCH\_NAME” == “dev” ]; then  docker build -t suresh/dev-react-image .  echo "Suresh@1994" | docker login -u "suresh" --password-stdin  docker push suresh/dev-react-image:latest  elif [ "$BRANCH\_NAME" == "main" ]; then  docker build -t suresh/prod-react-image .  echo "Nivetha@1234" | docker login -u "suresh" --password-stdin  docker push suresh/prod-react-image:latest  fi |

chmod +x build.sh

A computer screen shot of text

AI-generated content may be incorrect.

deploy.sh

#!/bin/bash

echo "Deployment Branch: $BRANCH\_NAME"

docker stop my-reactapp-container || true

docker rm my-reactapp-container || true

if [ "$BRANCH\_NAME" == "dev" ]; then

docker pull suresh/dev-react-image:latest

docker run -d -p 80:80 --name my-reactapp-container suresh/dev-react-image:latest

elif [ "$BRANCH\_NAME" == "main" ]; then

docker pull suresh/prod-react-image:latest

docker run -d -p 80:80 --name my-reactapp-container sureshv/prod-react-image:latest

fi

chmod +x deploy.sh

A screen shot of a computer program

AI-generated content may be incorrect.

.gitignore

node\_modules

build

.env

A screenshot of a computer program

AI-generated content may be incorrect.

.dockerignore

node\_modules

Dockerfile

docker-compose.yml

A screenshot of a computer

AI-generated content may be incorrect.

ls -a

A black background with white text

AI-generated content may be incorrect.

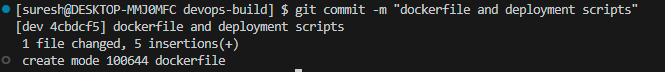
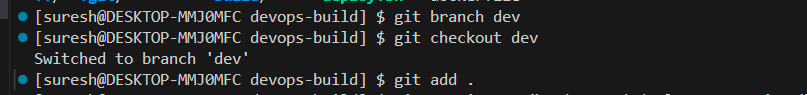
**Step-4:** push the files to github repo dev branch

git branch dev

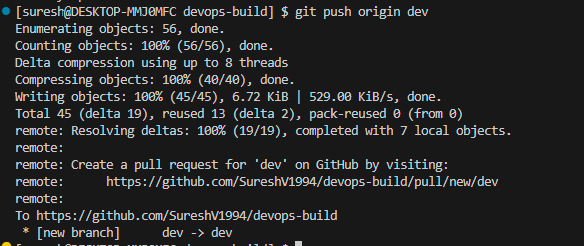
git checkout dev

git add .

git commit -m “docker and deployment scripts”



git push origin dev



A screenshot of a computer

AI-generated content may be incorrect.

**Step-5:** In docker hub, create repo dev-react-image– public ; prod-react-image– private

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

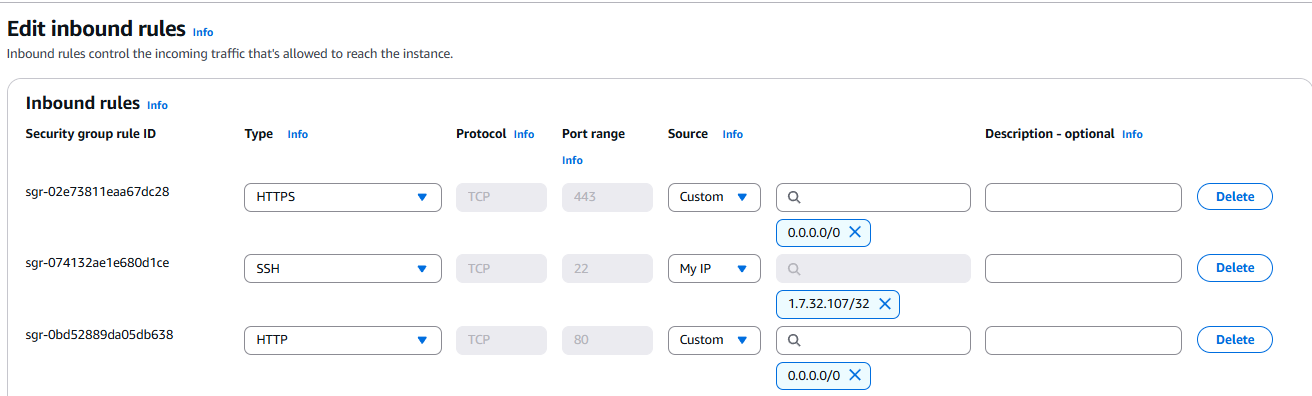
AI-generated content may be incorrect.

**Step-6:** Launch ec2 instance – chosen ubuntu machine and install jenkins

A screenshot of a computer

AI-generated content may be incorrect.

edit inbound rules as below



connecting via ec2 instance connect

sudo apt update

sudo apt install fontconfig openjdk-17-jre

A computer screen shot of a black screen

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

A screenshot of a computer

AI-generated content may be incorrect.

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list **>** /dev/null

sudo apt update

sudo apt-get install Jenkins

A screenshot of a computer program

AI-generated content may be incorrect.

sudo systemctl enable jenkins

sudo systemctl start jenkins

Access Jenkins via http://54.167.152.201:8080



sudo cat /var/lib/jenkins/secrets/initialAdminPassword



copy paste the initial admin pw

A screenshot of a computer

AI-generated content may be incorrect.

give the credentials

A screenshot of a computer

AI-generated content may be incorrect.

Install docker plugin >> manage Jenkins >> plugin >>Available plugins

A screenshot of a chat

AI-generated content may be incorrect.

Check whether git, pipeline is installed or install it.

Install SSH Agent

A screenshot of a computer

AI-generated content may be incorrect.

Install Github integration

A white rectangular object with black lines

AI-generated content may be incorrect.

Multibranch pipeline

A screenshot of a computer

AI-generated content may be incorrect.

**Step 7:** add the webhook in the github so that when push occurs in the git, webhook trigger Jenkins to automate the process.

Goto githubrepo > settings > webhooks>add webhook>Jenkins url > application/json>just the push event

A screenshot of a webhook

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a webbook

AI-generated content may be incorrect.

**Step 8:** Create multibranching pipeline in jenkins

Creating pipeline job in Jenkins, so that when push occurs in dev branch, webhook trigger will trigger Jenkins to run build.sh file. Upon running, my Image will be built and pushed to docker hub dev repo which will be pulled and deployed. The same way when merged with main and push occurs, it will be pushed to prod repo in docker hub which will be pulled and deployed.

For Multibranching pipeline, we should add pipeline script in Jenkinsfile and push it to github repo since we cannot add script like we do in regular pipeline in Jenkins.

Jenkinsfile

pipeline {

agent any

stages {

stage('Build and Push Docker Image') {

steps {

// Grant executable permissions to the build script

sh 'chmod +x deploy.sh'

sh 'chmod +x build.sh'

// Build the Docker image using the build script

sh './build.sh'

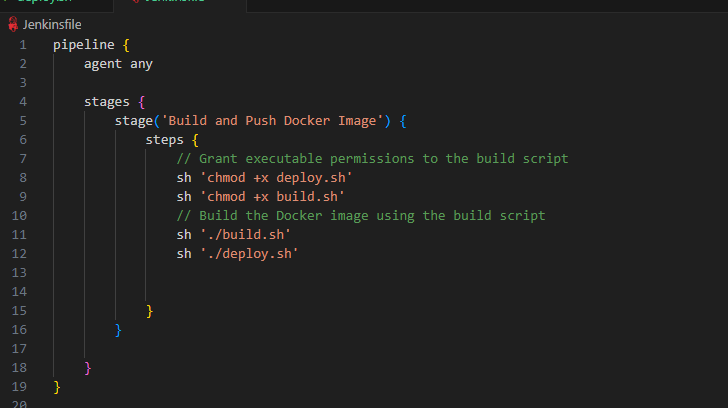
sh './deploy.sh'

}

}

}

}



Push it to all branches in git hub

A screen shot of a computer screen

AI-generated content may be incorrect. A screen shot of a computer

AI-generated content may be incorrect.

Create a my-reactapp1 - pipeline in Jenkins

A screenshot of a computer program

AI-generated content may be incorrect.

Choose git hub under branch source and give the git hub url

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a browser window

AI-generated content may be incorrect.

Enable scan repo triggers

A screenshot of a computer

AI-generated content may be incorrect.

Then save.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 9:** Install docker on ec2

sudo apt update

sudo apt install -y docker.io

A screenshot of a computer

AI-generated content may be incorrect.

sudo systemctl start docker

sudo systemctl enable docker

A black background with white text

AI-generated content may be incorrect.

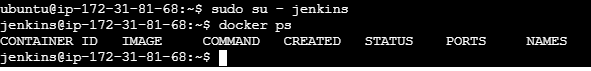
sudo usermod -aG docker Jenkins -- we are creating Jenkins user in docker because Jenkins has to configure docker. docker need sudo command to run but Jenkins usually wont be running as root user so we are creating user.

sudo usermod -aG docker ubuntu

A black and white text

AI-generated content may be incorrect.

go inside Jenkins , ubuntu and check whether it can access docker





To establish ssh connection,

Go inside jenkins

create directory .ssh

A screen shot of a computer

AI-generated content may be incorrect.

cd .ssh

suresh94.pem

copy the content from our pem file and paste it here

-----BEGIN OPENSSH PRIVATE KEY-----

b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtz

c2gtZWQyNTUxOQAAACAl1ZNlF9hE4xvgdrG9sgrsIrncZTSFPqOz6NkbpH31/gAA

AIgmRP8fJkT/HwAAAAtzc2gtZWQyNTUxOQAAACAl1ZNlF9hE4xvgdrG9sgrsIrnc

ZTSFPqOz6NkbpH31/gAAAEAwUQIBATAFBgMrZXAEIgQgfxBs0Ukj19YH8PafeBDC

siXVk2UX2ETjG+B2sb2yCuwiudxlNIU+o7Po2RukffX+AAAAAAECAwQF

-----END OPENSSH PRIVATE KEY-----

chmod 400 suresh94.pem

then try ssh- ssh -i /var/lib/jenkins/.ssh/ suresh94.pem ubuntu@54.165.106.99

A computer screen with white text

AI-generated content may be incorrect.

Now Jenkins can ssh to ubuntu machine.

Now push the code

**Step 10:** Verification:

**Test Dev Deployment**

Pushing the code to dev branch

git commit --allow-empty -m "Test Webhook21"

git push origin dev



webhook is triggering the push

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Jenkins is automatically start build and dev repo image has been built and deployed successfully.

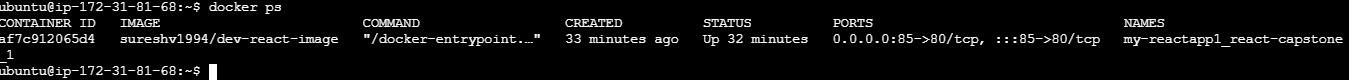
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

docker ps



Docker hub – Image pushed to dev repo

A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

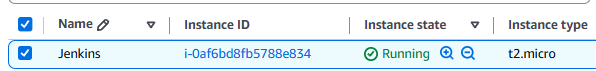
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Deployment in ec2 instance



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect.

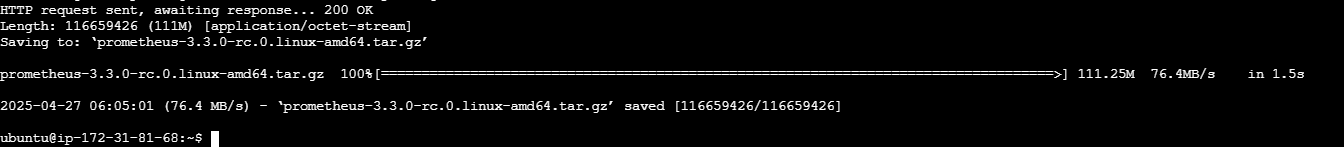
A person holding a tablet

AI-generated content may be incorrect.

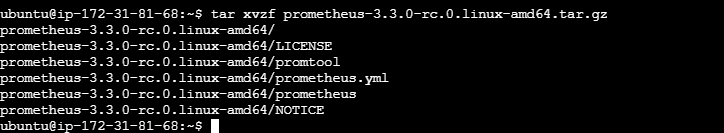
**Step 11:** Monitoring : using Prometheus, node\_exporter and alert manager

**Prometheus**

wget <https://github.com/prometheus/prometheus/releases/download/v3.3.0-rc.0/prometheus-3.3.0-rc.0.linux-amd64.tar.gz>



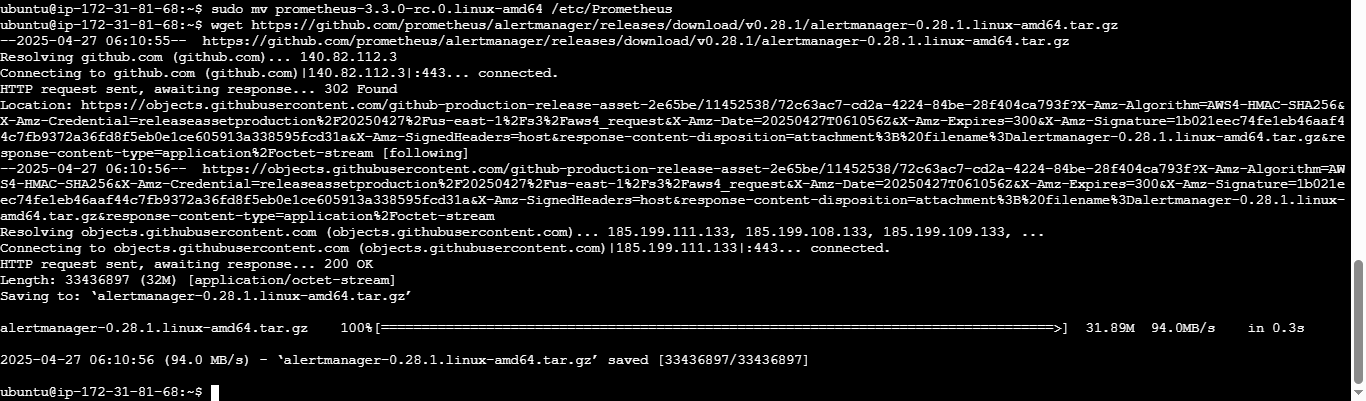
tar xvzf prometheus-3.3.0-rc.0.linux-amd64.tar.gz



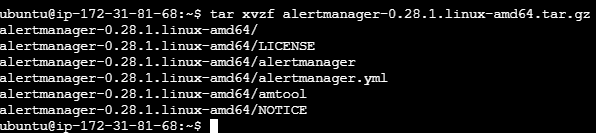
sudo mv prometheus-3.3.0-rc.0.linux-amd64 /etc/Prometheus

**Also download alert manager in same Prometheus doc**

wgt<https://github.com/prometheus/alertmanager/releases/download/v0.28.1/alertmanager-0.28.1.linux-amd64.tar.gz>



tar xvzf alertmanager-0.28.1.linux-amd64.tar.gz



sudo mkdir -p /etc/alertmanager

sudo mv alertmanager-0.28.1.linux-amd64/\* /etc/alertmanager/



**Also download node\_exporter in same Prometheus doc**

* The reason of downloading node\_exporter is, when I run Prometheus without node exporter, it cannot scrape the react app metrics, it was throwing error.
* Prometheus is a **pull-based monitoring system** that collects metrics from targets that expose them in the **Prometheus format** but my ec2 instance was not exposing that metrics to Prometheus. Also our react app is a front end app and it is not generating any metrics , so Prometheus cannot scrape react app metrics in the required format.
* To overcome this installing node\_exporter , this exposes the metrics in Prometheus format so Prometheus scrape node exporter and monitor the metrics like health.

wget<https://github.com/prometheus/node_exporter/releases/download/v1.9.1/node_exporter-1.9.1.linux-amd64.tar.gz>



tar xvzf node\_exporter-1.9.1.linux-amd64.tar.gz



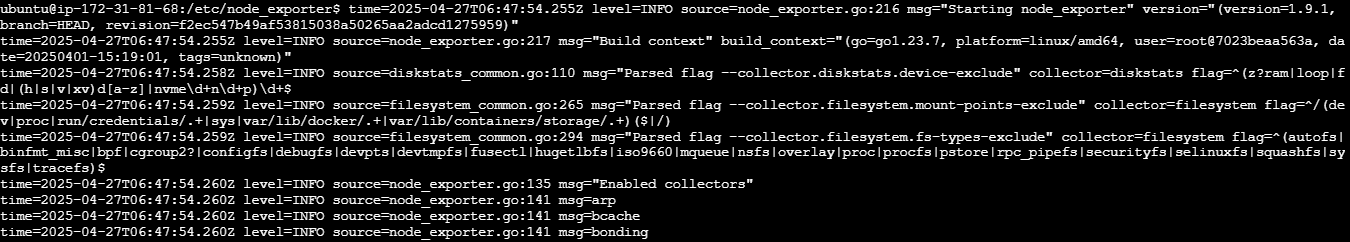
sudo mv node\_exporter-1.9.1.linux-amd64 /etc/node\_exporter



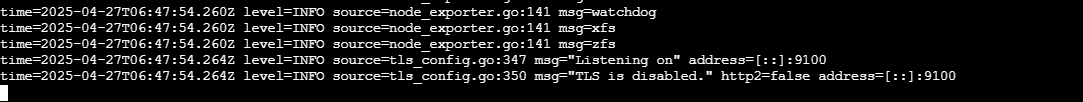
cd /etc

cd node\_exporter

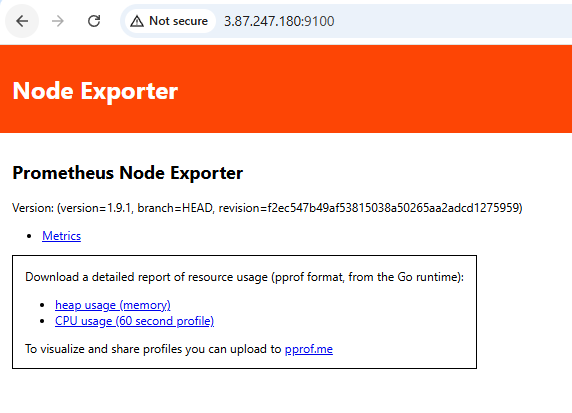
./node\_exporter &



Note the Port number – 9100



Verify it is running or not using http:// <http://3.87.247.180:85:9100>



Now go to Prometheus.yaml file

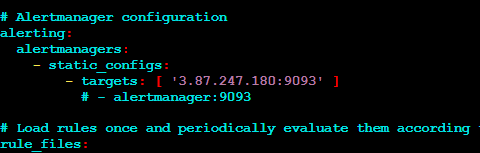
cd /etc

cd Prometheus

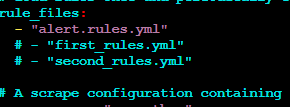
vi prometheus.yml



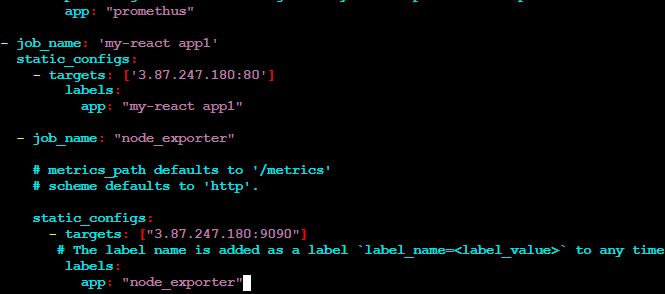
modify – add alertmanager port



add alert.rules.yml



Add the below job



then save and quit

cd /etc/Prometheus

sudo nano alert.rules.yml



groups:

- name: ApplicationAlerts

rules:

- alert: ApplicationDown

expr: up{app="my-react app1"} == 0

for: 1m

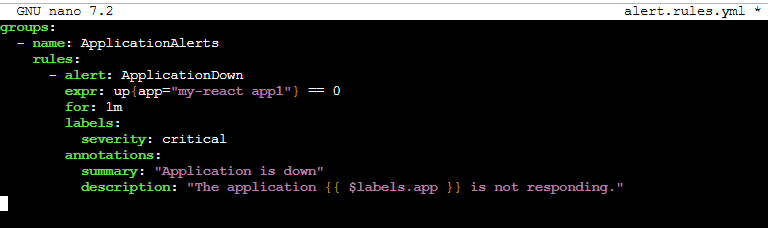
labels:

severity: critical

annotations

summary: "Application is down"

description: "The application {{ $labels.app }} is not responding."

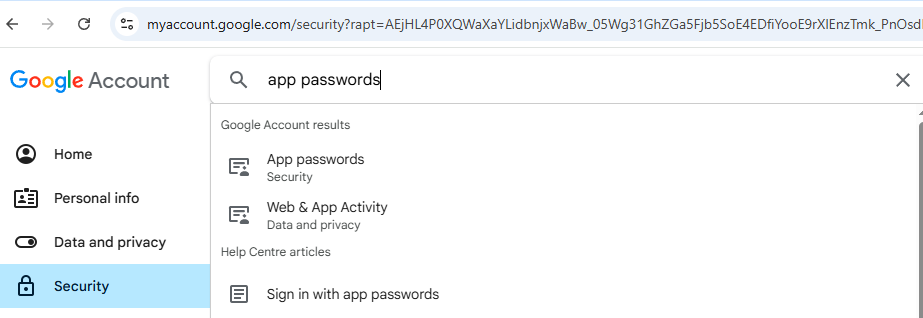


cd alertmanager

A black background with white text

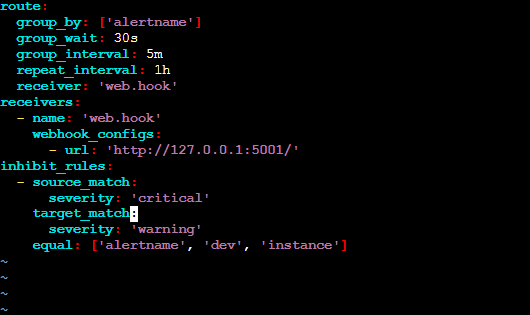
AI-generated content may be incorrect.

Before editing alert manager yaml, go to gmail and generate app password,



My app pw : cwtm tvrw edrk yaan

vi alertmanager.yml



Here the webhook config is given, we can add email config alongside or delete webhook and add email config.

Modify

Add the below content

route:

receiver: "email"

receivers:

- name: "email"

email\_configs:

- to: "vjsuru94@gmail.com"

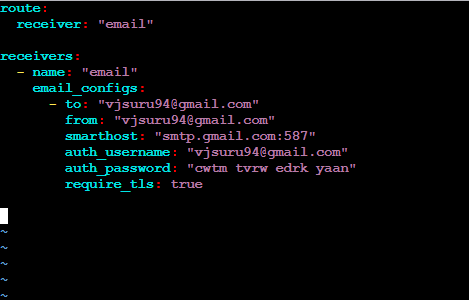
from: " vjsuru94@gmail.com"

smarthost: "smtp.gmail.com:587"

auth\_username: "vjsuru94@gmail.com"

auth\_password: “cwtm tvrw edrk yaan"

require\_tls: true



Save

Here auth pw can be set as env variable as well for security purpose

auth\_password: "{{ env.EMAIL\_PASSWORD }}"

export EMAIL\_PASSWORD="enwv pqvn lxbt zalv"

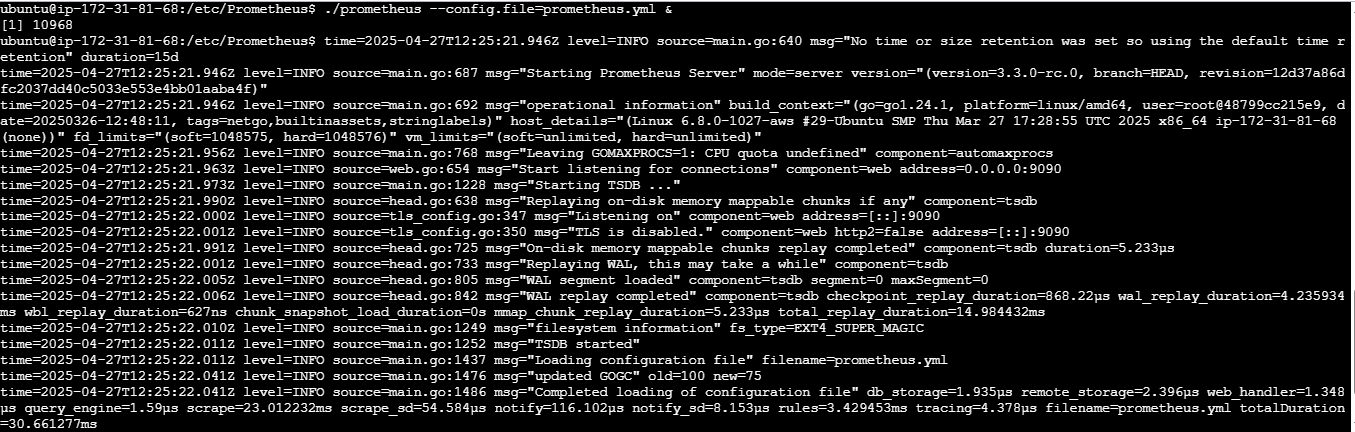
Now **prometheus.yml and alert.rules.yml ➝ prometheus directory.**

**alertmanager.yml ➝ alertmanager directory.**

**Node\_exporter ➝ node\_exporter directory**

Starting Prometheus

./prometheus --config.file=prometheus.yml &



cd etc

cd alertmanager

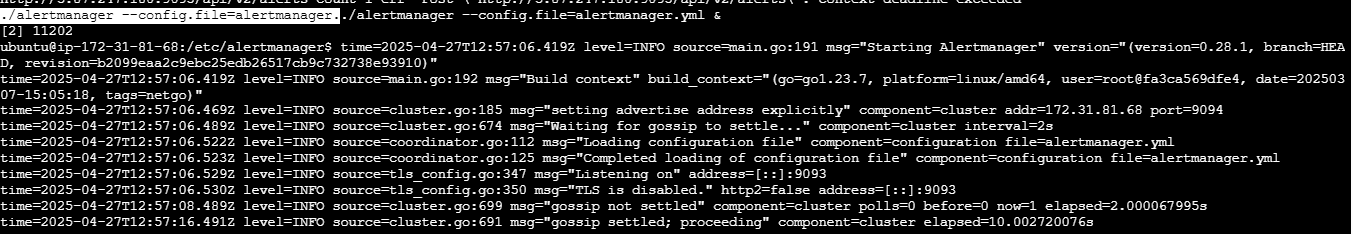
sudo mkdir -p /etc/alertmanager/data

sudo chown -R ubuntu:ubuntu /etc/alertmanager/data

we are giving alertmanager permission to create the data



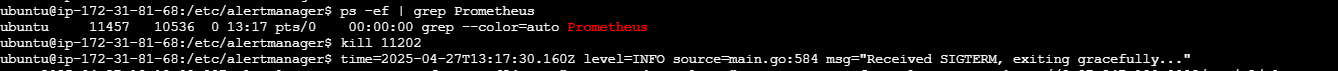
./alertmanager --config.file=alertmanager.yml &



Restarting Prometheus

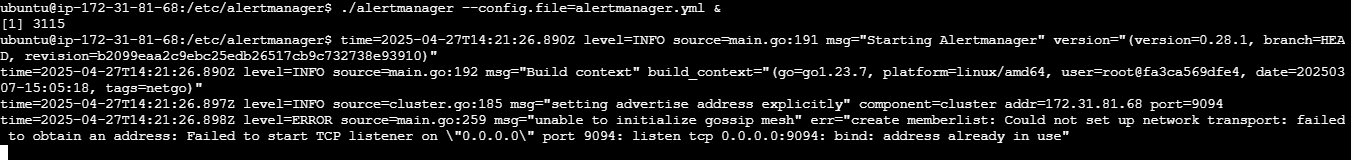
Ps -ef | grep Prometheus

Kill 11202



Cd Prometheus

Running it again



**Prometheus url**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

http://3.87.247.180:9090/

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Choose status >> target health

A screenshot of a computer

AI-generated content may be incorrect.

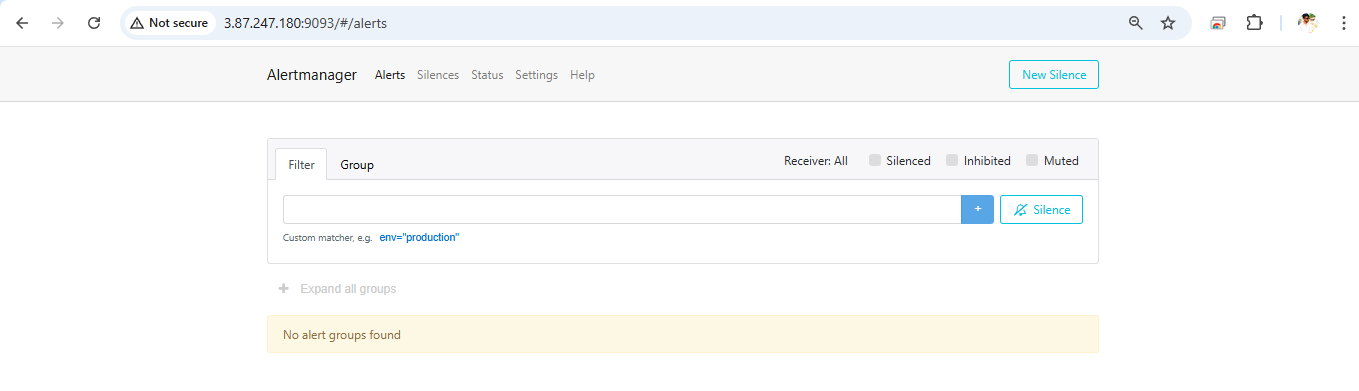
Monitoring the health

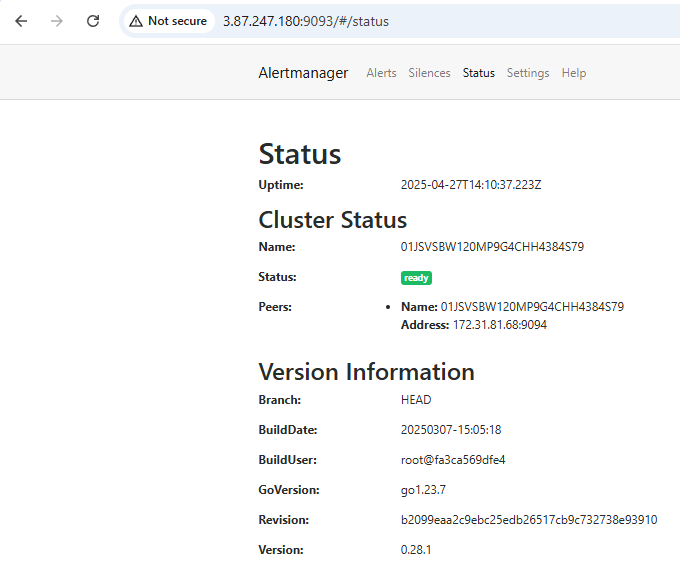
A screenshot of a computer

AI-generated content may be incorrect.

**Alertmanager url**

<http://3.87.247.180//:9093/>





**Ports: Jenkins -8080**

**Prometheus – 9090**

**Node-exporter – 9100**

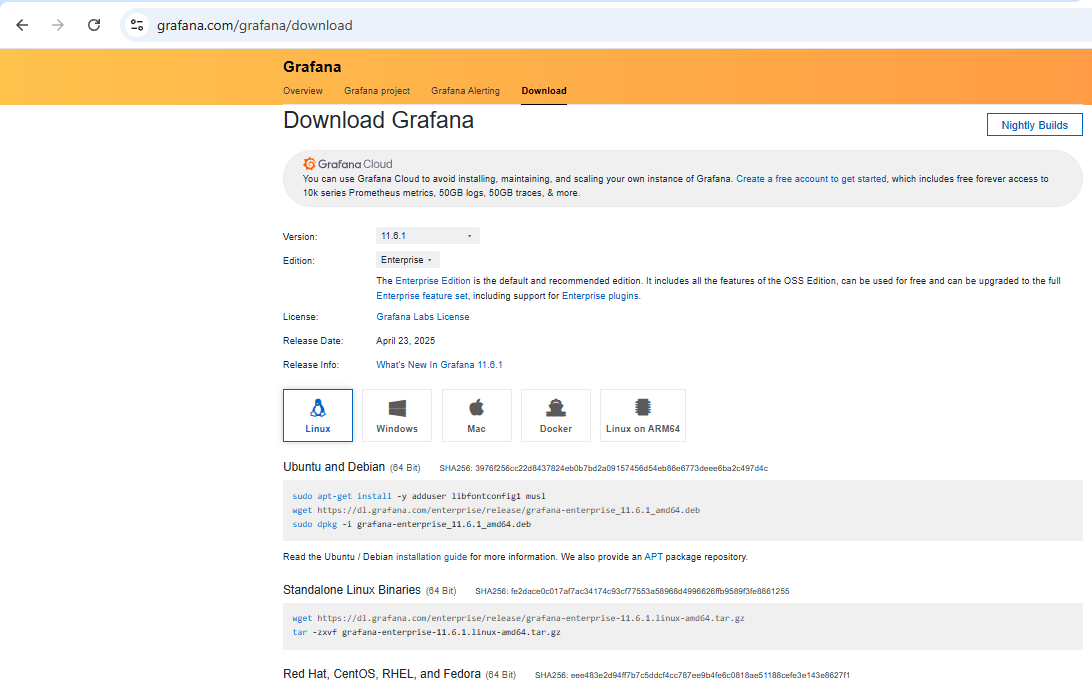
**Alert manager -9093**

**Checklist :**

1. **Jenkins (login page, configuration settings, execute step commands). - done**
2. **AWS (EC2 console, SG configurations) - done**
3. **Docker Hub repo with image tags - done**
4. **Deployed site page -done**
5. **Monitoring health check status – done (along with the mail trigger)**

**Optional**: Have installed Grafana as well for visualization

Visit the website [Download Grafana | Grafana Labs](https://grafana.com/grafana/download) to get the Grafana download link



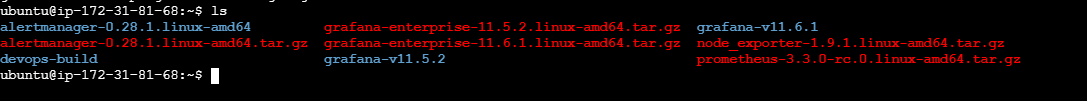
wget <https://dl.grafana.com/enterprise/release/grafana-enterprise-11.6.0.linux-amd64.tar.gz>

A black screen with white text

AI-generated content may be incorrect.

tar -xvzf grafana-enterprise-11.5.2.linux-amd64.tar.gz

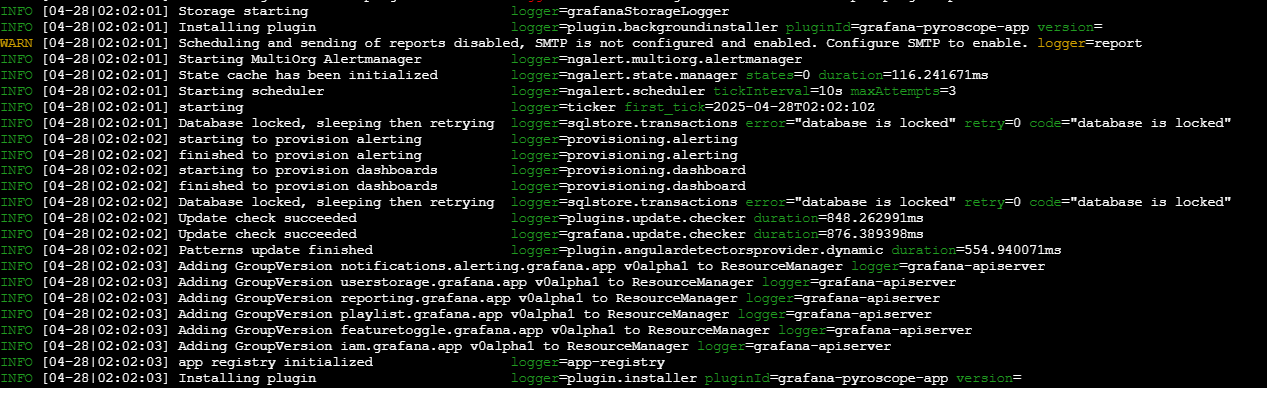
ls

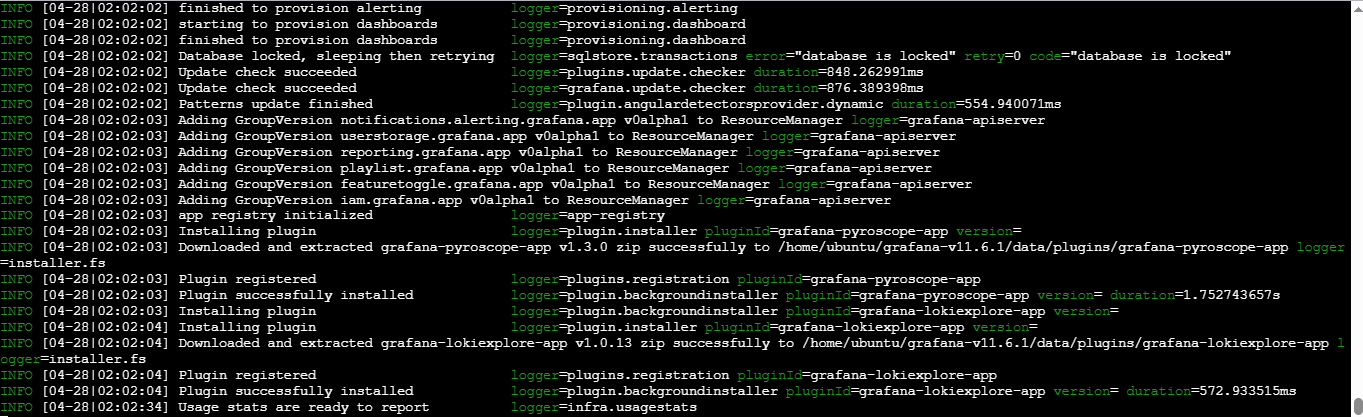


cd grafana-v11.6.0

cd bin

./grafana-server &

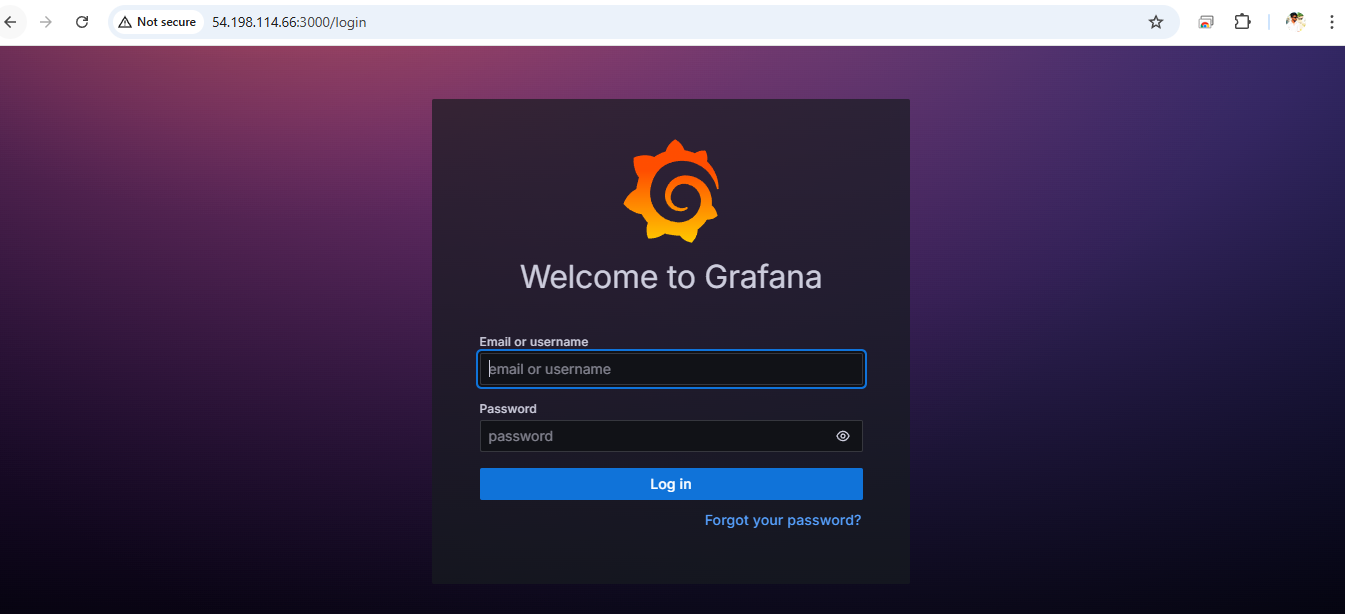




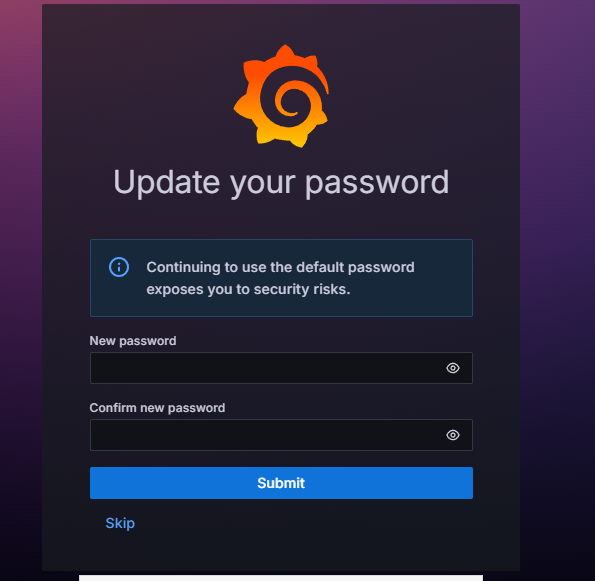
Logged out and logged back, so ip got changed

**Grafana port : 3000**

<http://54.198.114.66:3000>

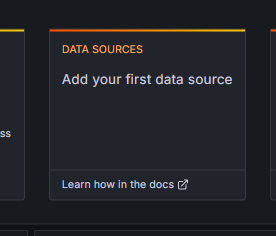


Login using credentials, use admin as the default password

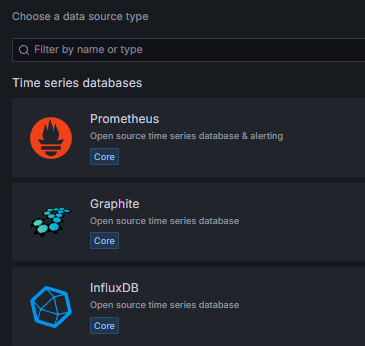




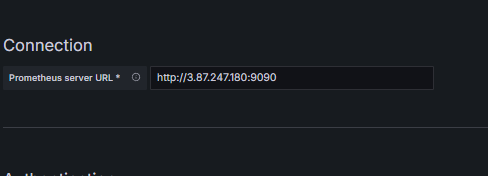
Add the data sources

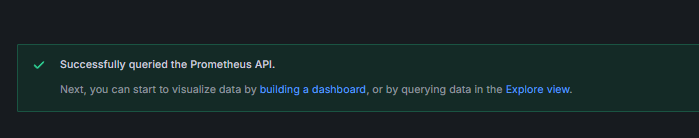


Choose Prometheus

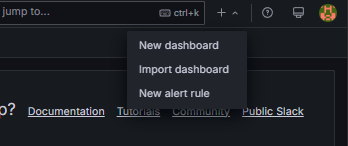


Give the Prometheus url



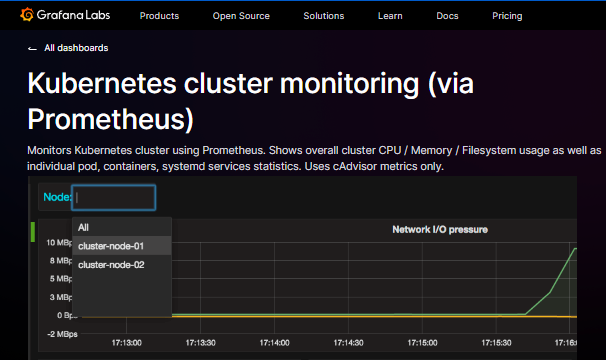


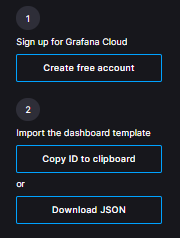
Then go to home> new dashboard or import dashboard



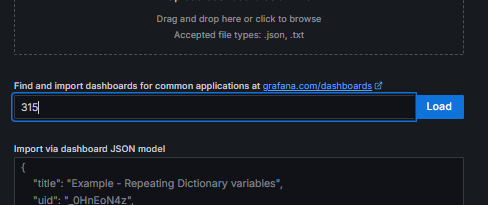
For import dashboard

Visit Grafana labs >> choose the desired dashboard and copy the id

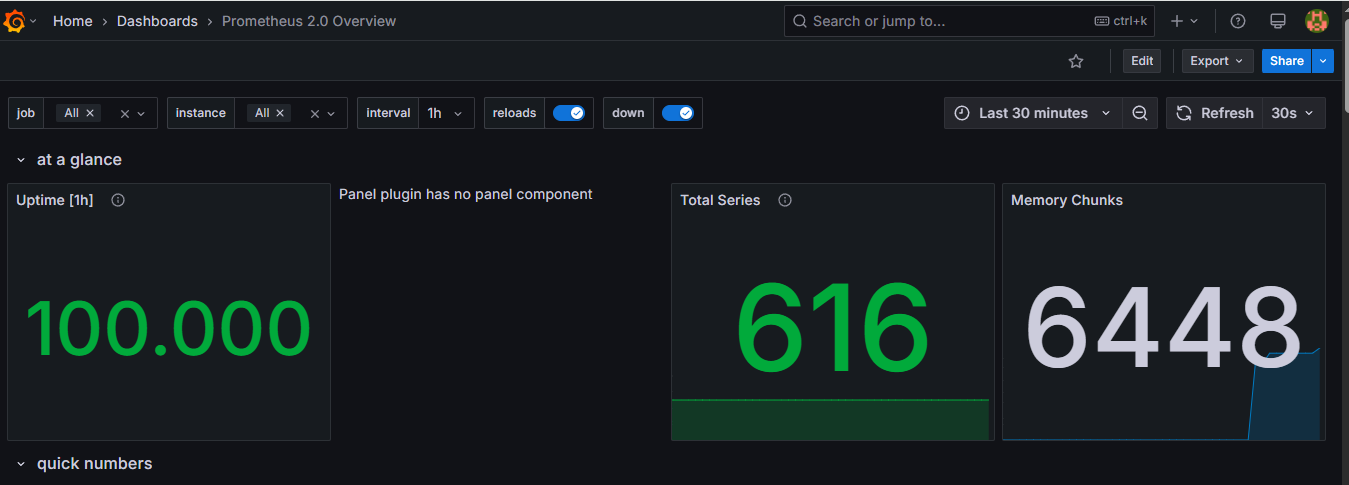




Paste the id and load



Set the query to view the dashboard



Explore different virtualization options