**Docker Metrics Monitoring with Prometheus and Grafana**

**Prometheus:  
It is an open-source monitoring and alerting system specifically designed for DevOps practices. It functions by collecting metrics from various targets such as servers, containers, and applications using a pull-based model. The collected metrics are then stored in a time-series database, enabling efficient querying and visualization of the data. Prometheus uses port number 9090 for accessing its web interface and supports flexible alerting features based on predefined thresholds.**

**Grafana:  
Grafana is an open-source data visualization and monitoring tool with a user-friendly interface. It allows you to create customizable dashboards, charts, graphs, and alerts to monitor metrics and performance indicators. Grafana supports various data sources, including Prometheus. By default, Grafana runs on port number 3000, and it provides a powerful platform for exploring and analyzing data in real-time.**

**“LET’S DO IT”**

**STEP 1: Launch Instance on AWS & Docker Installation:**

**AMI — ubuntu 20.04LTS   
Instance Type — t2.micro  
EBS Volume — 30GiB**

**DOCKER INSTALLATIONS ON UBUNTU :**

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

**STEP 2: Install Grafana:**

sudo su  
sudo apt-get install -y apt-transport-https  
sudo apt-get install -y software-properties-common wget  
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add –  
echo "deb https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list  
echo "deb https://packages.grafana.com/enterprise/deb beta main" | sudo tee -a /etc/apt/sources.list.d/grafana.list  
sudo apt-get update  
sudo apt-get install grafana-enterprise  
sudo apt-get install -y apt-transport-https  
sudo apt-get install -y software-properties-common wget  
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -  
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list  
echo "deb https://packages.grafana.com/oss/deb beta main" | sudo tee -a /etc/apt/sources.list.d/grafana.list  
sudo apt-get update  
sudo apt-get install grafana

**STEP 3: Create a Docker Container :**

**Create a Node.js web application container**

**Make a Directory for the Application**

mkdir Node.js  
cd Node.js

**Generate a package.json the file that describes your app and its dependencies:**

vi package.json

{  
 "name": "docker\_web\_app",  
 "version": "1.0.0",  
 "description": "Node.js on Docker",  
 "author": "First Last <first.last@example.com>",  
 "main": "server.js",  
 "scripts": {  
 "start": "node server.js"  
 },  
 "dependencies": {  
 "express": "^4.18.2"  
 }  
**}**

**Then ,**

apt install npm  
npm install  
  
#It will generate a package-lock.json file which will be copied for Docker image.

**Create a server.js file that defines a web app using the**[**Express.js**](https://expressjs.com/)**framework:**

vi server.js

'use strict';  
  
const express = require('express');  
  
// Constants  
const PORT = 8080;  
const HOST = '0.0.0.0';  
  
// App  
const app = express();  
app.get('/', (req, res) => {  
 res.send('Hello Raj');  
});  
  
app.listen(PORT, HOST, () => {  
 console.log(`Running on http://${HOST}:${PORT}`);  
});

**Make a Dockerfile:**

vi Dockerfile

**Inside the file paste the below content:**

FROM node:18  
  
# Create app directory  
WORKDIR /usr/src/app  
  
# Install app dependencies  
# A wildcard is used to ensure both package.json AND package-lock.json are copied  
# where available (npm@5+)  
COPY package\*.json ./  
  
RUN npm install  
# If you are building your code for production  
# RUN npm ci --omit=dev  
  
# Bundle app source  
COPY . .  
  
EXPOSE 8080  
CMD [ "node", "server.js" ]

**Make a .dockerignore file in the same directory Dockerfile**

vi .dockerignore

node\_modules  
npm-debug.log  
  
#This will prevent your local modules and debug logs from being copied onto your Docker image and possibly overwriting modules installed within your image.

**Build Dockerfile to create a docker image**

docker build . -t <username>/node-web-app

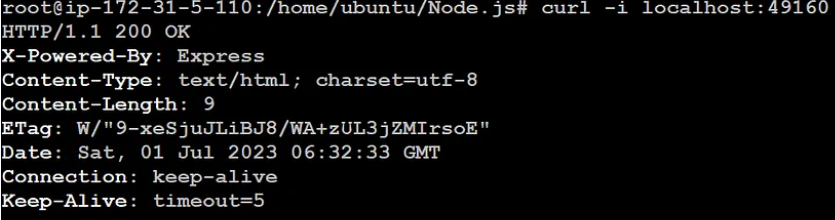
**Run the Image which we created now,**

docker run -p 49160:8080 -d shrini/node-web-app

**Get into the Container using EXEC Command**

docker exec -it <container id> /bin/bash  
exit #to get out from the container

curl -i localhost:49160

****

**STEP 4: Create a httpd Container**

cd ..  
mkdir httpd  
cd httpd  
  
vi Dockerfile

FROM httpd:2.4  
COPY index.html /usr/local/apache2/htdocs/  
EXPOSE 80

**Create a new file“index.html” in the same directory as the Dockerfile.**

vi index.html

# you can add your own HTML Content

**Build the Docker Image**

docker build -t <name> .

**Run the Docker Image**

docker run -d -p 8080:80 <name>

**STEP 5: Build and Run Prometheus Docker Container**

Create a **prometheus.yaml** file

vi prometheus.yaml

global:  
 scrape\_interval: 15s  
  
 external\_labels:  
 monitor: 'codelab-monitor'  
  
scrape\_configs:  
 - job\_name: 'prometheus'  
 scrape\_interval: 5s  
  
 static\_configs:  
 - targets: ['PUBLIC IP:9090']

**Run the prometheus container using the created prometheus.yaml file**

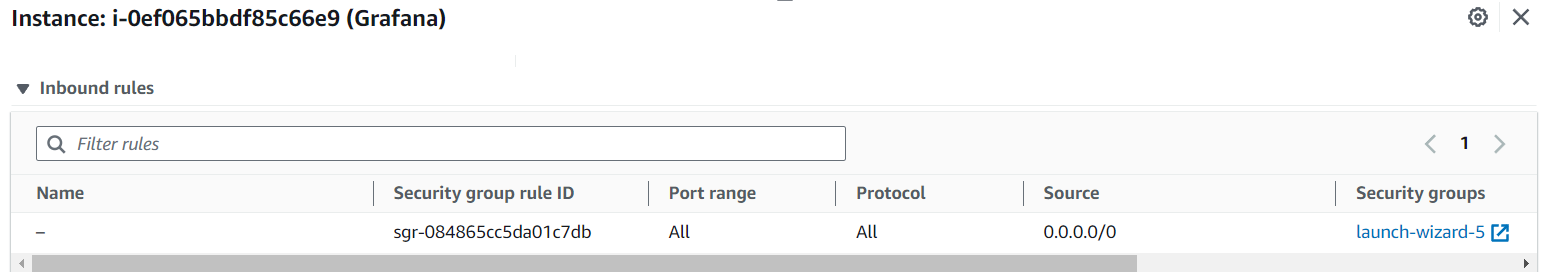
sudo docker run -d --name prometheus -v $(pwd):/home/ubuntu/httpd/ -p 9090:9090 prom/prometheus --config.file=/home/ubuntu/httpd/prometheus.yaml

**So after this we will three containers with the command of docker ps:**

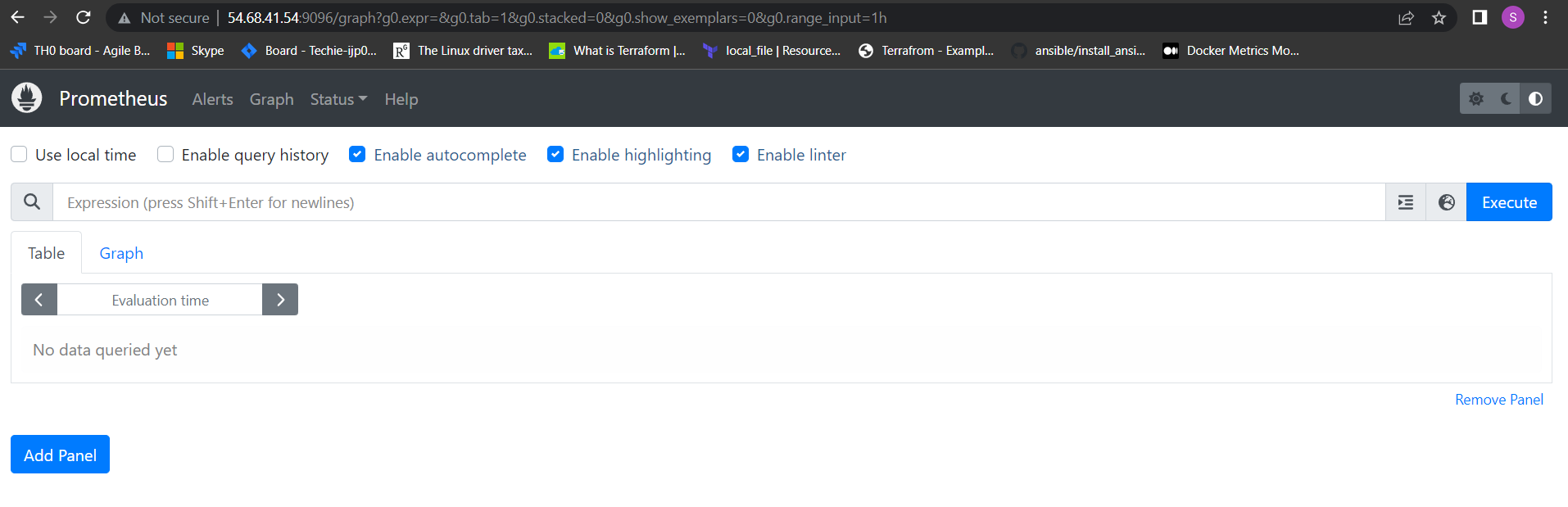
****

* **Port 9090 for prometheus,**
* **Port 49160 for node.js app server,**
* **Port 8080 for apache-server,**
* **Port 3000 for Grafana,**
* **Port 9323 for docker logs.**

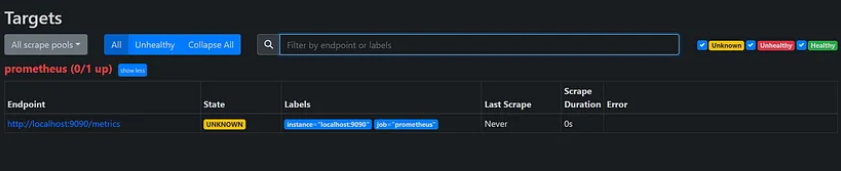
**Make all ports open in the inbound rules of the Instance Security Groups**

****

**<ip\_address:9090> — Prometheus Dashboard**

****

**And on the “status” targets we have to add Prometheus:**

****

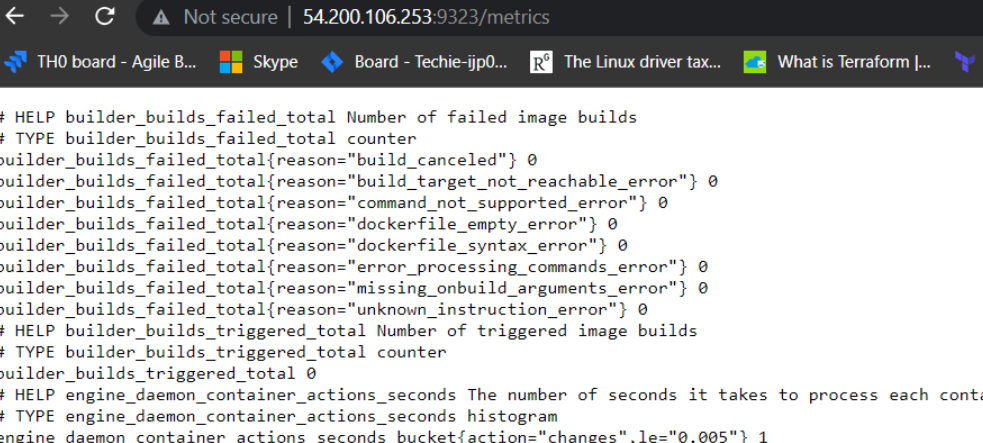
**STEP 6: Integrate the docker containers and share the real time logs with Grafana**

* **Edit the *daemon.json file* in *etc/docker/daemon.json* directory**
* **If you don’t have it , no problem create one.**
* **And paste the below values in your daemon.json**

{  
 "metrics-addr" : "0.0.0.0:9323",  
 "experimental" : true  
}

**sudo systemctl restart docker**

* **check logs on http://<public-ip>:9323, It shows docker logs on this website**

****

* **Add Docker job in *prometheus.yaml* file (file we created for the prometheus docker image ).**

- job\_name: 'docker'  
 scrape\_interval: 5s  
 static\_configs:  
 - targets: ['65.1.148.23:9323']

docker restart <container id>

## STEP 7: Check the logs and create docker dashboard on Grafana UI.

## In Grafana you have to log in with **admin** user and **admin** password on http://<public-ip>:3000

## 

## 

## 

## After adding datasource we can check our container stats in grafana:

## 

## 

## Linking part is completed, In my instance currently "3" container is running and in Grafana also it is showing "3"

**“And that’s it successfully we are using grafana for monitoring docker metrics with the help of Prometheus “**

**----------------------------------------------------SHRINIVAS---------------------------------------------------------**