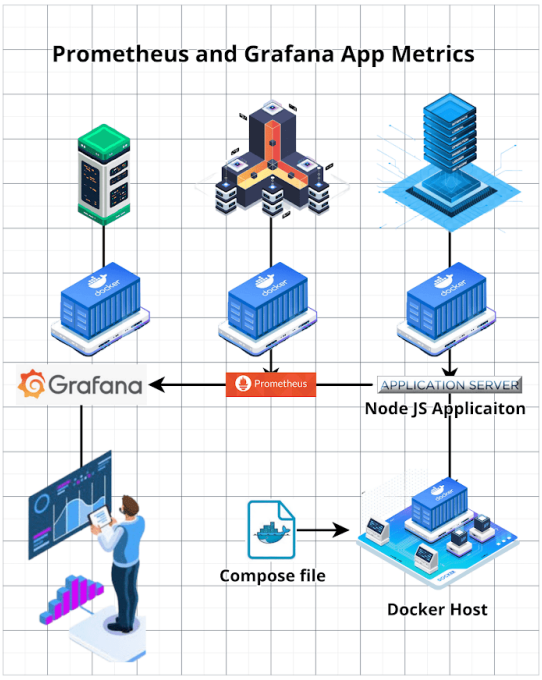
Storie 4 : Deploying Prometheus , grafana and nodejs on docker - App Metrics

PROMETHEUS AND GRAFANA WITH NODEJS APP METRIC



Launch the Ubuntu version 20 Virtual Machine:

- **Launch A ubuntu version 20 Instance from AWS Cloud**
- **Allow the Inbound rules 83,9091,3000 which are the port numbers of Nodejs server, Prometheus, Grafana.**

Install Docker :

- **To Install Docker on Ubuntu instance visit the below site**

https://get.docker.com

```
# ----
# ============================================================================
#
# To install the latest stable versions of Docker CLI, Docker Engine, and their
# dependencies:
#
# 1. download the script
#
#   $ curl -fsSL https://get.docker.com -o install-docker.sh
#
# 2. verify the script's content
#
#   $ cat install-docker.sh
#
# 3. run the script with --dry-run to verify the steps it executes
#
#   $ sh install-docker.sh --dry-run
#
# 4. run the script either as root, or using sudo to perform the installation.
#
#   $ sudo sh install-docker.sh
#
# Command-line options
# ============================================================================
```

- **Below commands are the Docker Installation commands**

```
curl -fsSL https://get.docker.com -o install-docker.sh
```

```
sudo sh install-docker.sh
```

```
sudo apt  install docker-compose -y
```

```
sudo usermod -aG docker $USER
```

```
newgrp docker
```

- **Clone the repository from the Github**

sudo git clone https://github.com/mubeen507/prometheus-grafana-app-monitoring.git

```
ubuntu@ip-172-31-11-114:~$ sudo git clone https://github.com/yellaiahgithub/prometheus-grafana-app-monitoring.git
Cloning into 'prometheus-grafana-app-monitoring'...
remote: Enumerating objects: 333, done.
remote: Counting objects: 100% (333/333), done.
remote: Compressing objects: 100% (181/181), done.
remote: Total 333 (delta 153), reused 321 (delta 147), pack-reused 0
Receiving objects: 100% (333/333), 1.09 MiB | 7.39 MiB/s, done.
Resolving deltas: 100% (153/153), done.
```

- **List of directories inside repo as shown in the pic**

```
ubuntu@ip-172-31-11-114:~/prometheus-grafana-app-monitoring$ ll
total 32
drwxr-xr-x 5 root   root   4096 Jun 20 07:58 ./
drwxr-xr-x 6 ubuntu ubuntu 4096 Jun 20 07:58 ../
drwxr-xr-x 8 root   root   4096 Jun 20 07:58 .git/
-rw-r--r-- 1 root   root   1151 Jun 20 07:58 docker-compose.yaml
drwxr-xr-x 2 root   root   4096 Jun 20 07:58 grafana-data/
drwxr-xr-x 3 root   root   4096 Jun 20 07:58 nodejs-application/
-rw-r--r-- 1 root   root    257 Jun 20 07:58 prometheus.yaml
-rw-r--r-- 1 root   root   1143 Jun 20 07:58 readme.md
```

- **Cat the docker-compose.yaml file**

```
ubuntu@ip-172-31-11-114:~/prometheus-grafana-app-monitoring$ cat docker-compose.yaml
version: "3"
services:
  nodejs-application:
    build:
      context: ./nodejs-application
    container_name: nodejs-application
    image: nodejs-application
    ports:
      - "83:5000"
  prometheus:
    container_name: prometheus-svc
    image: prom/prometheus
    ports:
      - "9091:9090"
    command: --config.file=/etc/prometheus/prometheus.yaml
    volumes:
      - ./prometheus.yaml:/etc/prometheus/prometheus.yaml
  grafana:
    image: grafana/grafana:5.0.4
    ports:
      - "3000:3000"
    environment:
      - GF_AUTH_BASIC_ENABLED=false
      - GF_AUTH_ANONYMOUS_ENABLED=true
      - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin
  grafana-dashboards:
    image: alpine:3.10
    depends_on:
      - grafana
    volumes:
      - ./grafana-data:/grafana
    command: >
      /bin/sh -c "
        apk add --no-cache curl
        echo 'waiting for grafana'
        sleep 5s
        cd /grafana
        curl --request POST http://grafana:3000/api/datasources --header 'Content-Type: application/json' -d @datasources.json
        curl --request POST http://grafana:3000/api/dashboards/db --header 'Content-Type: application/json' -d @dashboard.json"
```

Compose file Explanation

Here's an explanation of what this Docker Compose snippet is doing:

1. **Prometheus Service**:
    - **container_name**: Sets the name of the Prometheus container to "prometheus-svc."
    - **image**: Specifies the Docker image to use for Prometheus, in this case, "prom/prometheus."
    - **ports**: Maps port 9091 on the host to port 9090 in the Prometheus container.
    - **command**: Sets the command that will be executed when the container starts. It tells Prometheus to use the configuration file located at "/etc/prometheus/prometheus.yaml."
    - **volumes**: Mounts the local "prometheus.yaml" file into the container at the path "/etc/prometheus/prometheus.yaml." This allows you to provide a custom Prometheus configuration.
2. **Grafana Service**:
    - **image**: Specifies the Docker image to use for Grafana, "grafana/grafana:5.0.4."
    - **ports**: Maps port 3000 on the host to port 3000 in the Grafana container.
    - **environment**: Sets environment variables for Grafana. In this case, it disables basic authentication, enables anonymous access, and assigns the "Admin" role to anonymous users.
3. **Grafana Dashboards Service**:
    - **image**: Uses the Alpine Linux base image "alpine:3.10" for this service.
    - **depends_on**: Specifies that this service depends on the "grafana" service, meaning it will only start after the "grafana" service has started.
    - **volumes**: Mounts a local directory called "grafana-data" into the container at "/grafana."
    - **command**: Defines a multi-line shell command to be executed when the container starts. This command does the following:
        - Installs the "curl" utility in the container.

- Waits for Grafana to start (sleeps for 5 seconds).
- Changes the working directory to "/grafana."
- Sends HTTP POST requests to the Grafana API to configure datasources and import a dashboard. The data for these operations is read from external files "datasources.json" and "dashboard.json," which should be present in the local directory.

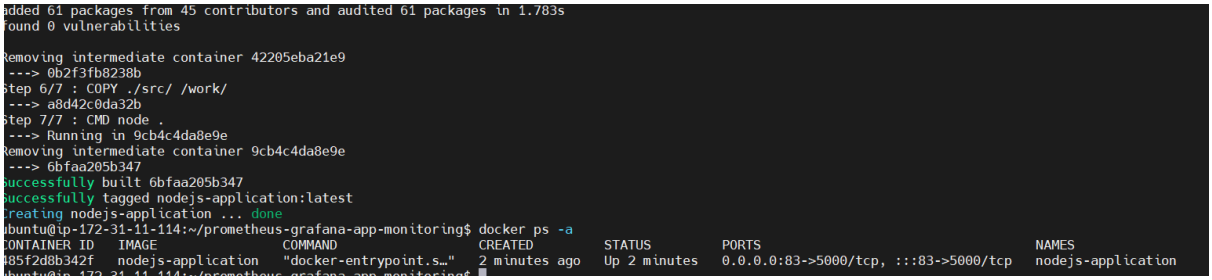
he **curl** command you provided is making an HTTP POST request to a URL, specifically to the Grafana API, and sending a JSON payload from a file named "datasources.json." Here's a breakdown of the command:

- **curl**: This is a command-line tool used for making HTTP requests.
- **-request POST**: Specifies that you want to make an HTTP POST request to the specified URL.
- **http://grafana:3000/api/datasources**: This is the URL to which the POST request is being sent. It's the endpoint for creating or configuring datasources in Grafana. In this case, "grafana" is used as the hostname, and it's assumed that this resolves to the correct IP address of the Grafana server within your Docker network. Port 3000 is the default port for Grafana's web interface and API.
- **-header 'Content-Type: application/json'**: This specifies an HTTP header with the name "Content-Type" and the value "application/json." It indicates that the data being sent in the request body is in JSON format.
- **d @datasources.json**: This option specifies the data to be sent in the HTTP POST request body. The "@" symbol followed by a file path indicates that the data should be read from the file named "datasources.json" and included in the request body. The content of this file should be a JSON-formatted configuration for a Grafana datasource.

Nodejs Application:

- **Run the Nodejs Application from the docker-compose.yaml file**
- **command to run the application using docker compose is**

```
docker-compose up -d --build nodejs-application
```



- **Check the nodejs application running or not with ip address followed by port number 83 which is defined in docker compose file.**

- **Nodejs Application Successfully running**

Prometheus server:

- **From the docker-compose.yml file run the Prometheus container by using the below command**

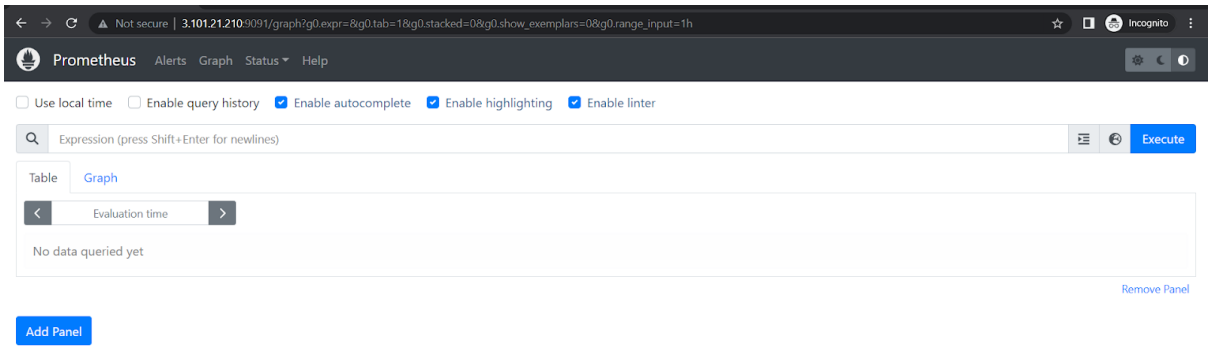
```
docker-compose up -d prometheus
```

```
ubuntu@ip-172-31-11-114:~/prometheus-grafana-app-monitoring$ docker-compose up -d prometheus
Pulling prometheus (prom/prometheus:)...
latest: Pulling from prom/prometheus
9b231b23b5cd: Pull complete
fab2335a7565: Pull complete
ff30808c587f: Pull complete
1d98eea8984b: Pull complete
4ed5d4519de0: Pull complete
bbd547bdb0c3: Pull complete
b1efab437686: Pull complete
aec8b68b2c06: Pull complete
b1eaa2b6d160: Pull complete
f12fd594aab9: Pull complete
3045f5e49715: Pull complete
69b94e1ae55f: Pull complete
Digest: sha256:0f0b7feb6f02620df7d493ad7437b6ee95b6d16d8d18799f3607124e501444b1
Status: Downloaded newer image for prom/prometheus:latest
Creating prometheus-svc ... done
```

- **Check the Running containers**

```
ubuntu@ip-172-31-11-114:~/prometheus-grafana-app-monitoring$ docker ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                                       NAMES
80b44a0a0dd2   prom/prometheus    "/bin/prometheus --c…"   8 seconds ago    Up 5 seconds    0.0.0.0:9091->9090/tcp, :::9091->9090/tcp   prometheus-svc
485f2d8b342f   nodejs-application "docker-entrypoint.s…"   16 minutes ago   Up 16 minutes   0.0.0.0:83->5000/tcp, :::83->5000/tcp       nodejs-application
```

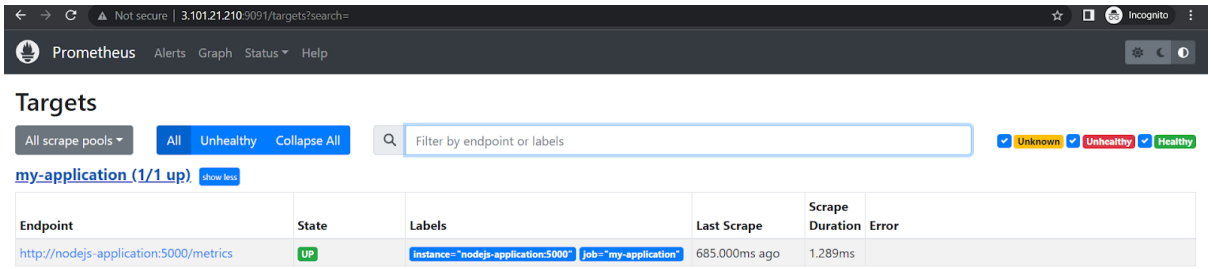
- **Check the Prometheus app from the Browser**



- **Inorder to scrape the metrics from the nodejs server , the Prometheus need to configure the Prometheus.yml file as shown below**

```
ubuntu@ip-172-31-11-114:~/prometheus-grafana-app-monitoring$ cat prometheus.yaml
global:
  scrape_interval:     5s
  evaluation_interval: 30s
scrape_configs:
- job_name: my-application
  honor_labels: true
  static_configs:
  - targets: ['nodejs-application:5000']
```

- Now check the status of the nodejs application inside the Prometheus

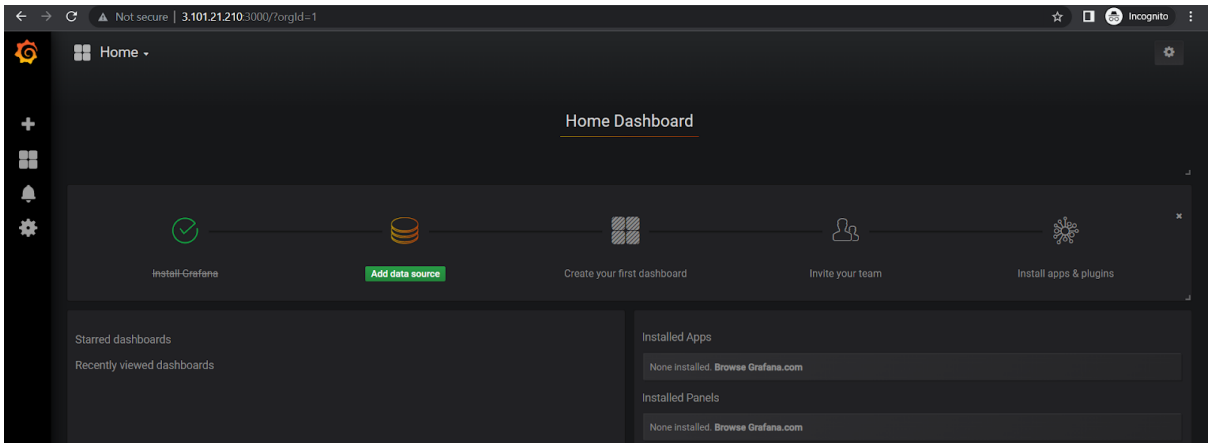


- The above pic represents the connectivity between nodejs server and Prometheus server.

Grafana server:

- **From the docker-compose.yml file run the Grafana container by the below command**

```
docker-compose up -d grafana
```



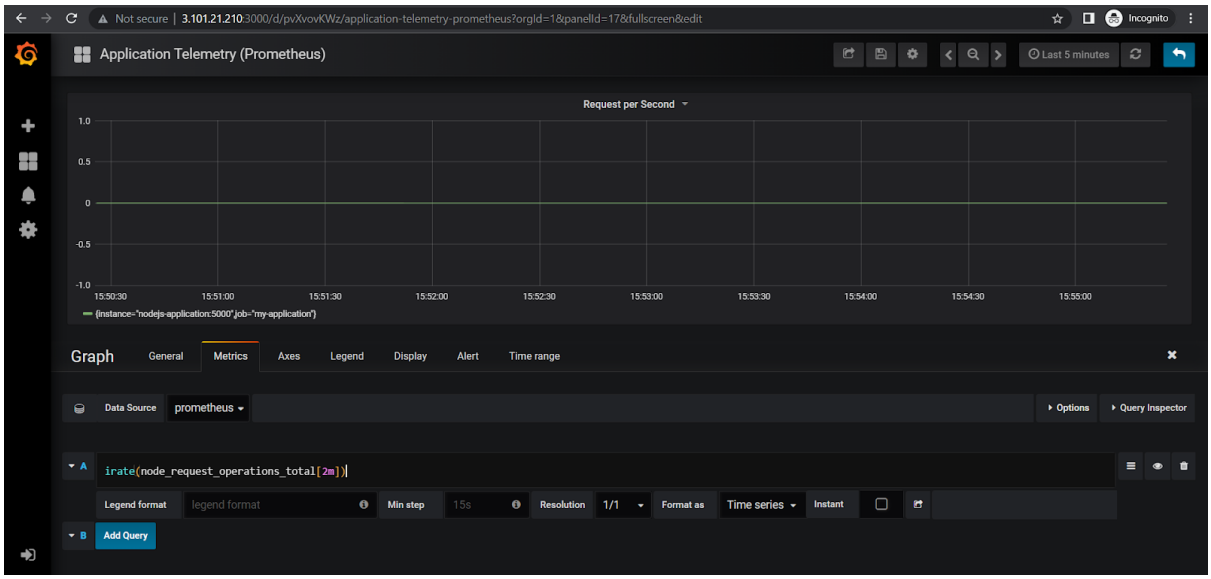
Grafana Dashboard:

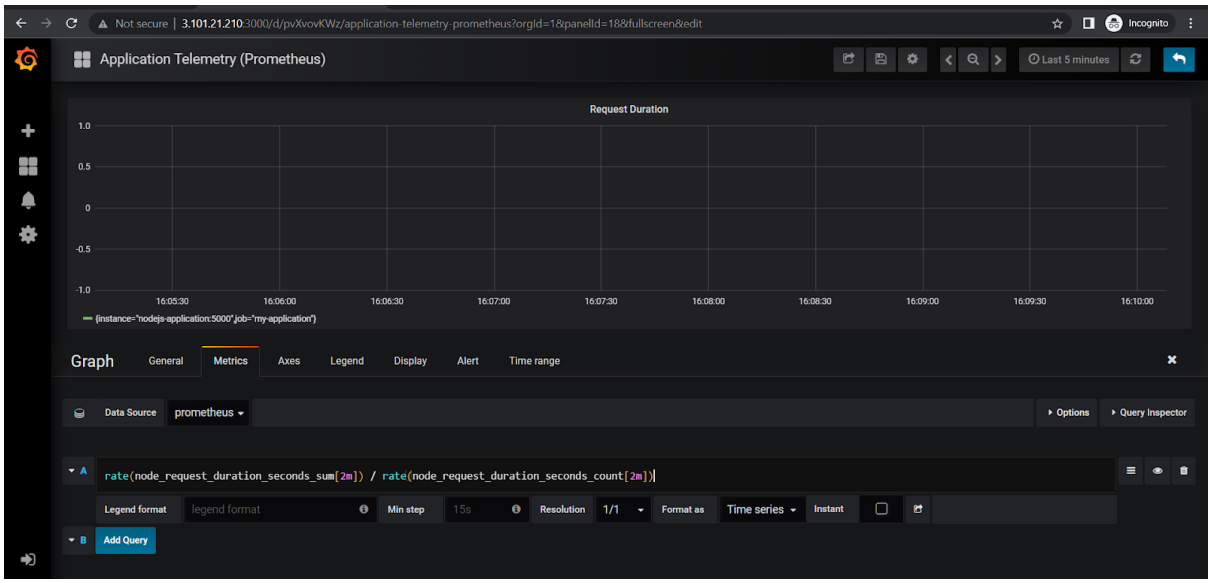
- **From the docker-compose.yml file run the Grafana dashboard container by the below command**

docker-compose up -d grafana-dashboards

```
ubuntu@ip-172-31-11-114:~/prometheus-grafana-app-monitoring$ docker-compose up -d grafana-dashboards
Pulling grafana-dashboards (alpine:3.10)...
3.10: Pulling from library/alpine
396c31837116: Pull complete
Digest: sha256:451eee8bedcb2f029756dc3e9d73bab0e7943c1ac55cff3a4861c52a0fdd3e98
Status: Downloaded newer image for alpine:3.10
prometheus-grafana-app-monitoring_grafana_1 is up-to-date
Creating prometheus-grafana-app-monitoring_grafana-dashboards_1 ... done
ubuntu@ip-172-31-11-114:~/prometheus-grafana-app-monitoring$
```

- **Check for the Grafana Dashboards**





- **Above dashboard represents the request per second which is hitting by incoming traffic to our Nodejs Application server**

Now check the Application metrics on Grafana Dashboard:

- **Click on the Nodejs Application reload button number of times(for hitting application multiple times).**
- **Now goto Grafana Dashboard will find the spikes(Gradually increase in spikes in dashboard)**

Before requests:

After Requests: