Web App Monitoring Using

Prometheus and Grafana

This guide explains how to monitor a Flask web application deployed through docker compose running on AWS using Prometheus (metrics collection) and Grafana (visualization). We'll cover everything from setup to dashboard creation.

Introduction

- **Prometheus**: An open-source monitoring system that collects metrics and stores them in a time-series database.

- **Grafana**: A visualization tool that connects to Prometheus and displays metrics in dashboards.

- **Goal**: Monitor:

- **Host-level metrics** (CPU, memory, disk) using **Node Exporter**.
- **Container-level metrics** using **cAdvisor**.
- Application metrics via Prometheus queries.

Architecture Overview

- Web Server Instance:

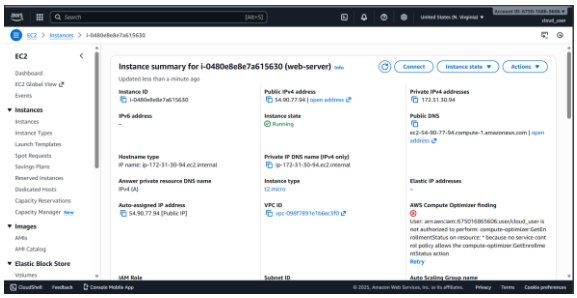
- Runs Flask app (port 5000).
- Exposes:
    - Node Exporter → port 9100.
    - cAdvisor → port 8080.

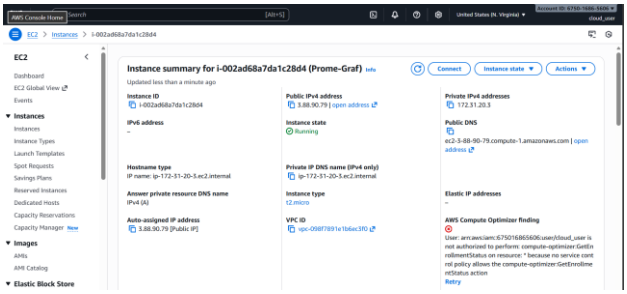
- Monitoring Instance:

- Runs Prometheus (port 9090) and Grafana (port 3000).
- Collects metrics from Web Server instance.

LET'S MONITOR OUR APP FROM SCRATCH STEP BY STEP

Step 1: Launch EC2 Instances (Amazon-linux t2.micro)





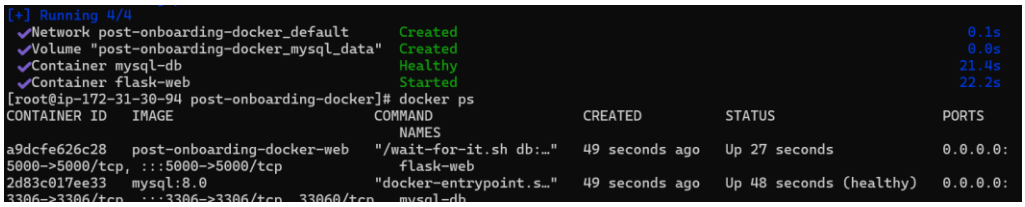
- **Web Server EC2**:

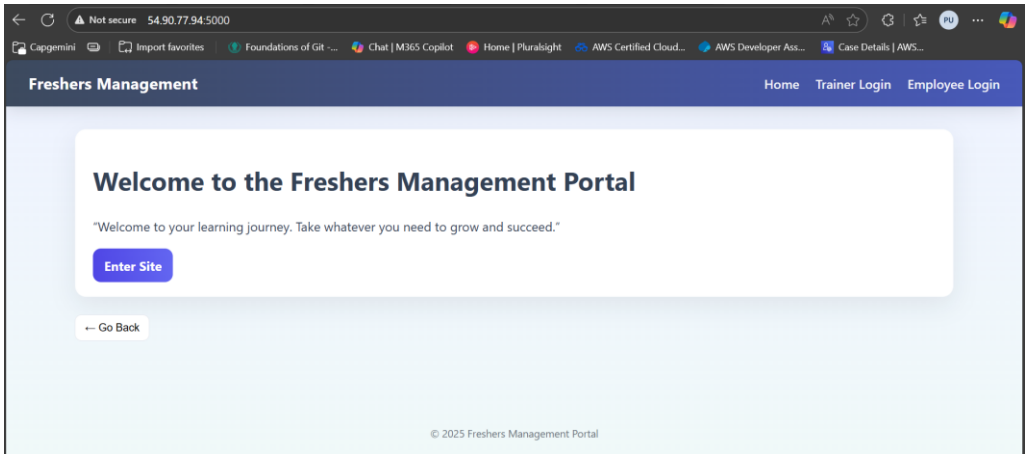
  - Security Group:
    - 5000 → Flask app.
    - 9100 → Node Exporter.
    - 8080 → cAdvisor.

- **Monitoring EC2**:

  - Security Group:
    - 9090 → Prometheus.
    - 3000 → Grafana.

>> SSH INTO THEM (Make sure your app is running, here we have deployed it using docker compose. (On the Web server Instance)





Step 2: Deploy App and Exporters

On **Web Server Instance**:

- Ensure Flask app is running (via Docker Compose).
- Add **Node Exporter** and **cAdvisor** containers in docker-compose.yml:

```
services:
  node-exporter:
    image: prom/node-exporter
    ports:
      - "9100:9100"

  cadvisor:
    image: gcr.io/cadvisor/cadvisor
    ports:
      - "8080:8080"
```

- Or pull the Node Exporter and Cadvisor images into the instance

```
[ec2-user@ip-172-31-30-94 ~]$
docker run -d \
  --name=cadvisor \
  --volume=/:/rootfs:ro \
  --volume=/var/run:/var/run:ro \
  --volume=/sys:/sys:ro \
  --volume=/var/lib/docker/:/var/lib/docker:ro \
  --publish=8080:8080 \
  gcr.io/cadvisor/cadvisor:latest

docker run -d \
  --name=node-exporter \
  --publish=9100:9100 \
  --volume=/proc:/host/proc:ro \
  --volume=/sys:/host/sys:ro \
  --volume=/:/rootfs:ro \
  prom/node-exporter:latest \
  --path.procfs=/host/proc \
  --path.sysfs=/host/sys \
  --path.rootfs=/rootfs
```

Run >> docker-compose up -d

Step 3: Set Up Prometheus and Grafana

On **Monitoring Instance**:

- Setup Instance

```
sudo yum update -y
sudo yum install -y docker
sudo service docker start
sudo systemctl enable docker
sudo usermod -aG docker ec2-user

# Install Compose plugin system-wide
sudo mkdir -p /usr/local/lib/docker/cli-plugins/
sudo curl -SL https://github.com/docker/compose/releases/download/v2.29.7/docker-compose-
linux-x86_64 \
  -o /usr/local/lib/docker/cli-plugins/docker-compose
sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose
docker compose version
```

- Create prometheus.yml:

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node'
    static_configs:
      - targets: ['<WEB_SERVER_PUBLIC_IP>:9100']

  - job_name: 'cadvisor'
    static_configs:
      - targets: ['<WEB_SERVER_PUBLIC_IP>:8080']
```

- Create docker-compose.yml for Prometheus + Grafana:

```
services:
  prometheus:
    image: prom/prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml

  grafana:
    image: grafana/grafana
    ports:
      - "3000:3000"
```

Run >> docker-compose up -d

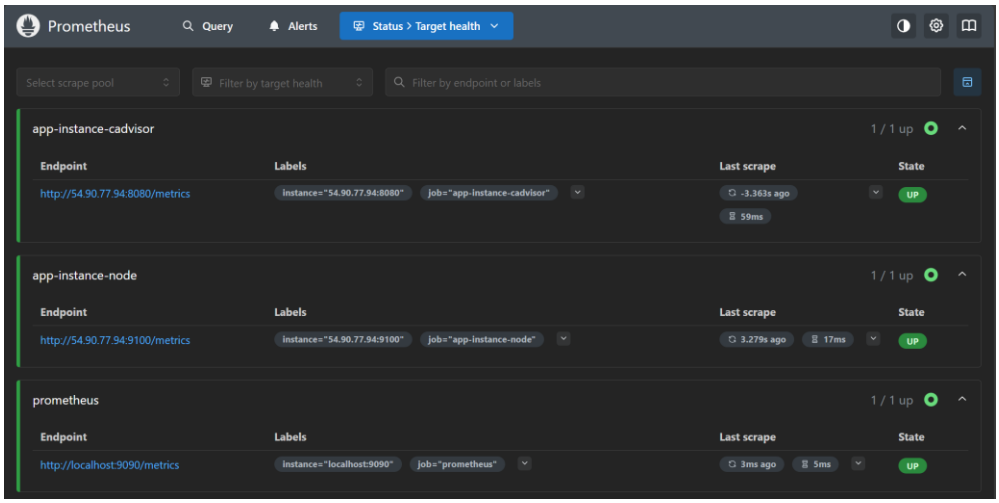
Step 4: Access Dashboards

- **Prometheus**: http://<Monitoring_IP>:9090

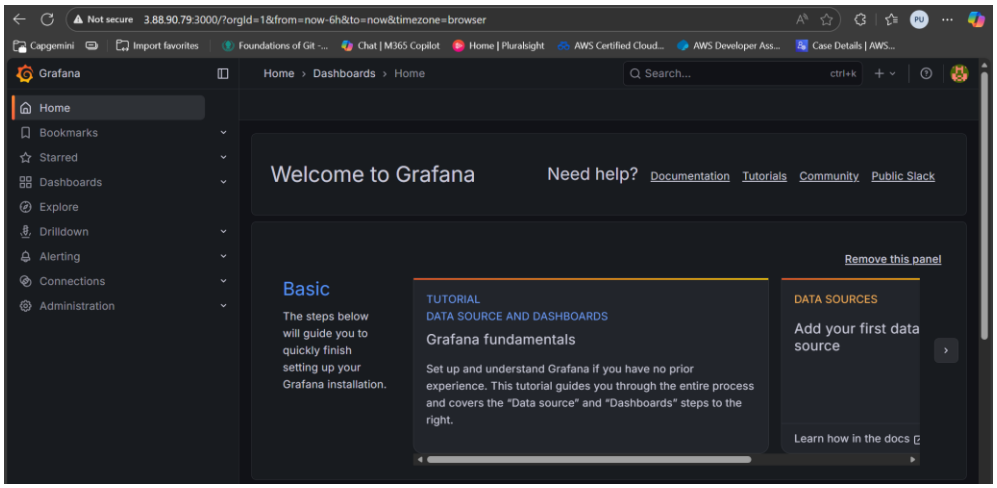
- Check **Targets** → should show Node Exporter and cAdvisor as **UP**.

- Grafana: http://<Monitoring_IP>:3000

- Default login: admin / admin.



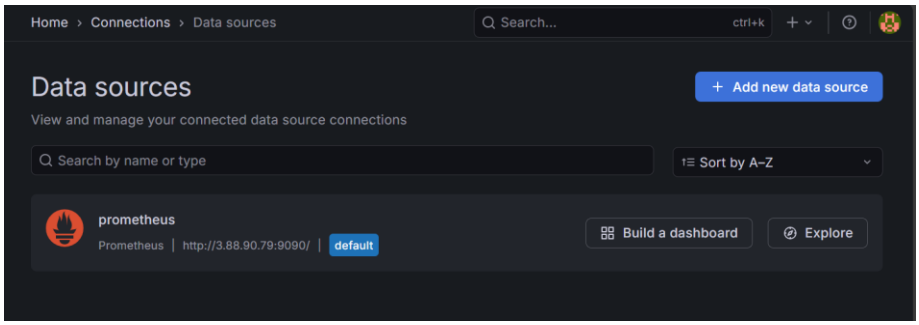
Check Targets on Prometheus Dashboard



Step 5: Connect Prometheus to Grafana

In Grafana:

- **Add Data Source** → Prometheus.
- URL: http://<Monitoring_IP>:9090.



Step 6: Create Dashboards

- Import **Node Exporter dashboard** (ID: 1860).

- Add custom panels for:

  - CPU usage.
  - Memory usage.
  - Container stats.

- Use **PromQL** queries in Prometheus for advanced metrics.

Prometheus Queries: PromQL+Intro.xlsx
Grafana Queries: GrafanaPanelQueries.xlsx

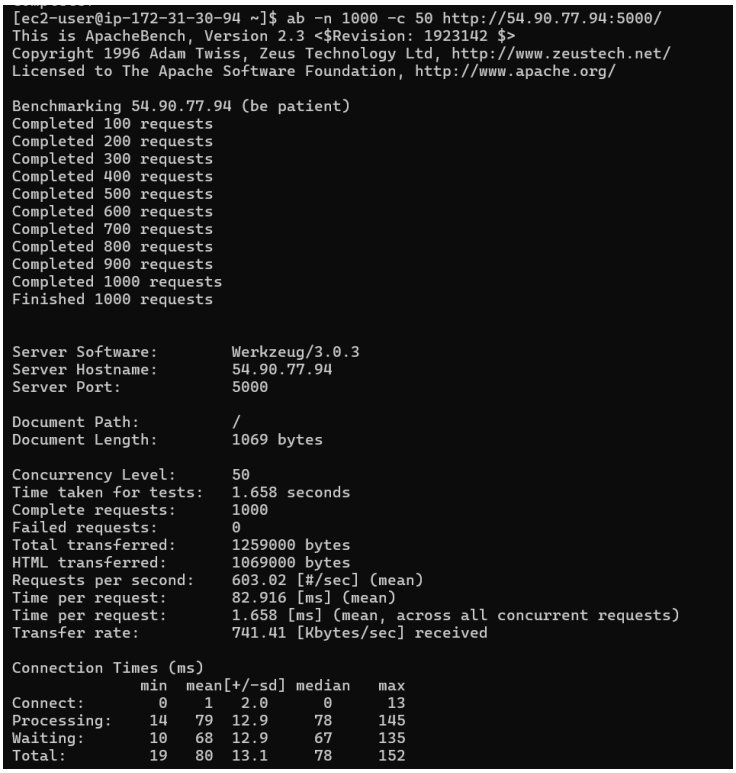
Step 7: Generate Traffic

To see meaningful metrics:

- Send requests to Flask app via its public IP or ALB DNS.
- Example:

  >> for i in {1..100}; do curl http://<WEB_SERVER_PUBLIC_IP>:5000; done

  OR Tools like APACHE BENCHMARK

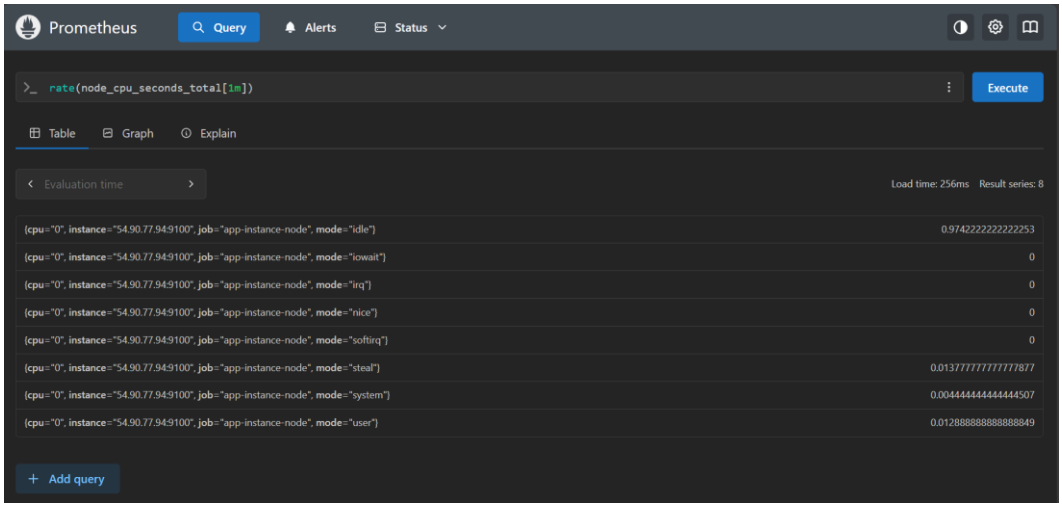


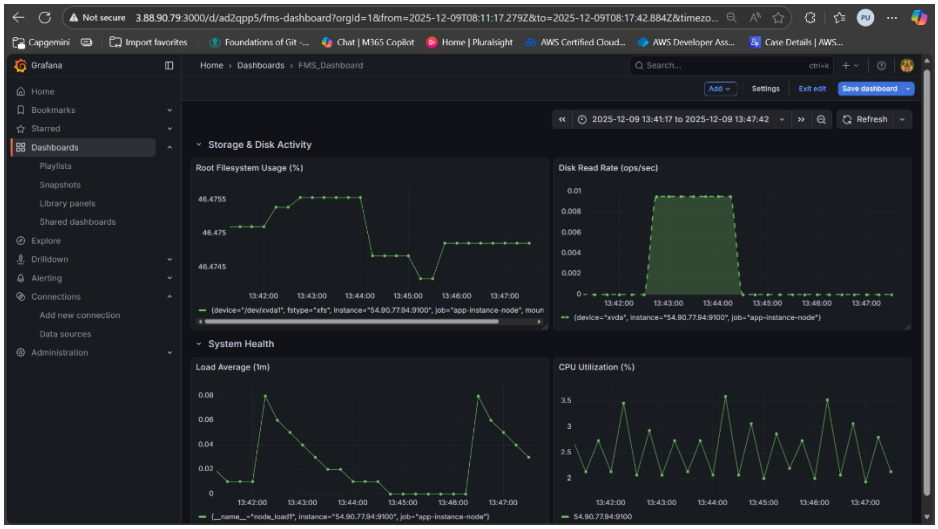
```
[ec2-user@ip-172-31-30-94 ~]$ ab -n 1000 -c 50 http://54.90.77.94:5000/
This is ApacheBench, Version 2.3 <$Revision: 1923142 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 54.90.77.94 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Werkzeug/3.0.3
Server Hostname:        54.90.77.94
Server Port:            5000

Document Path:          /
Document Length:        1069 bytes

Concurrency Level:      50
Time taken for tests:   1.658 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      1259000 bytes
HTML transferred:       1069000 bytes
Requests per second:    603.02 [#/sec] (mean)
Time per request:       82.916 [ms] (mean)
Time per request:       1.658 [ms] (mean, across all concurrent requests)
Transfer rate:          741.41 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   2.0      0      13
Processing:    14   79  12.9     78     145
Waiting:       10   68  12.9     67     135
Total:         19   80  13.1     78     152
```

Step 8: Verify

- Prometheus → Check metrics and targets.

- Grafana → Dashboards show real-time data.





Outcome:

- **Prometheus** scrapes metrics from Node Exporter and cAdvisor.
- **Grafana** visualizes metrics in dashboards.
- You can monitor:
    > Host health.
    > Container performance.
    > Application responsiveness.

Next Steps (Further Integrations)

- Add **Alertmanager** for alerts.

- Secure Grafana with HTTPS.

- Scale monitoring for multiple nodes.