Tej Mandaliya

Complete Infrastructure Monitoring Using Prometheus and Grafana

Tej Mandaliya

Introduction:

Infrastructure monitoring is a critical aspect of DevOps, enabling proactive detection and resolution of system issues. This project focuses on real-world infrastructure monitoring using **Prometheus, Grafana, and Loki** to dynamically monitor Docker, Kubernetes, and AWS CloudWatch.

The monitoring stack provides insights into the health and performance of containers, Kubernetes clusters, and cloud-based resources, ensuring efficient system operation and resource utilization.

Objective:

The Primary objective of this project is to:

- To set up Prometheus and Grafana for real-time monitoring.
- To collect metrics from Docker containers using cAdvisor.
- To monitor Kubernetes clusters (AWS EKS) with Prometheus and Grafana.
- To implement Loki for centralized log monitoring.
- To create dynamic dashboards in Grafana for visualization.

Technologies used:

- **AWS (Amazon Web Services):** Cloud computing platform used to deploy and manage infrastructure resources.
- **Prometheus:** An open-source monitoring system for collecting metrics from configured targets at given intervals.
- **Grafana:** A visualization tool used to create dashboards and display real-time monitoring data from Prometheus.
- **Loki:** A log aggregation system for collecting and storing logs efficiently.
- **cAdvisor:** A container monitoring tool used to track resource usage and performance statistics of running containers.

Tej Mandaliya

How I Set Up

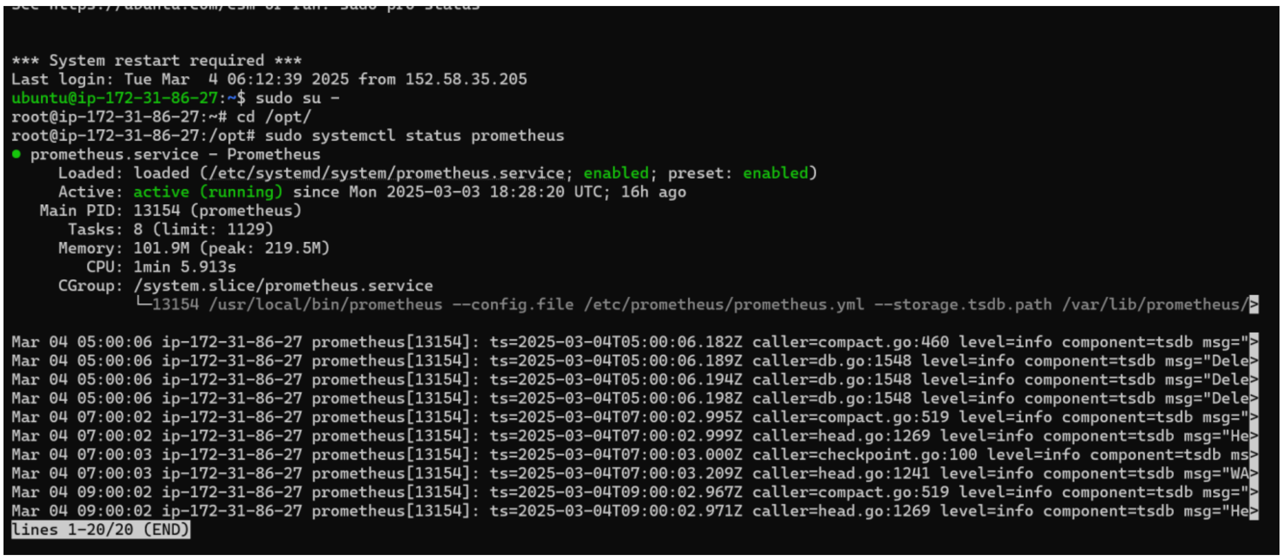
Step 1: Prometheus and Grafana Setup on AWS

Installation and Configuration

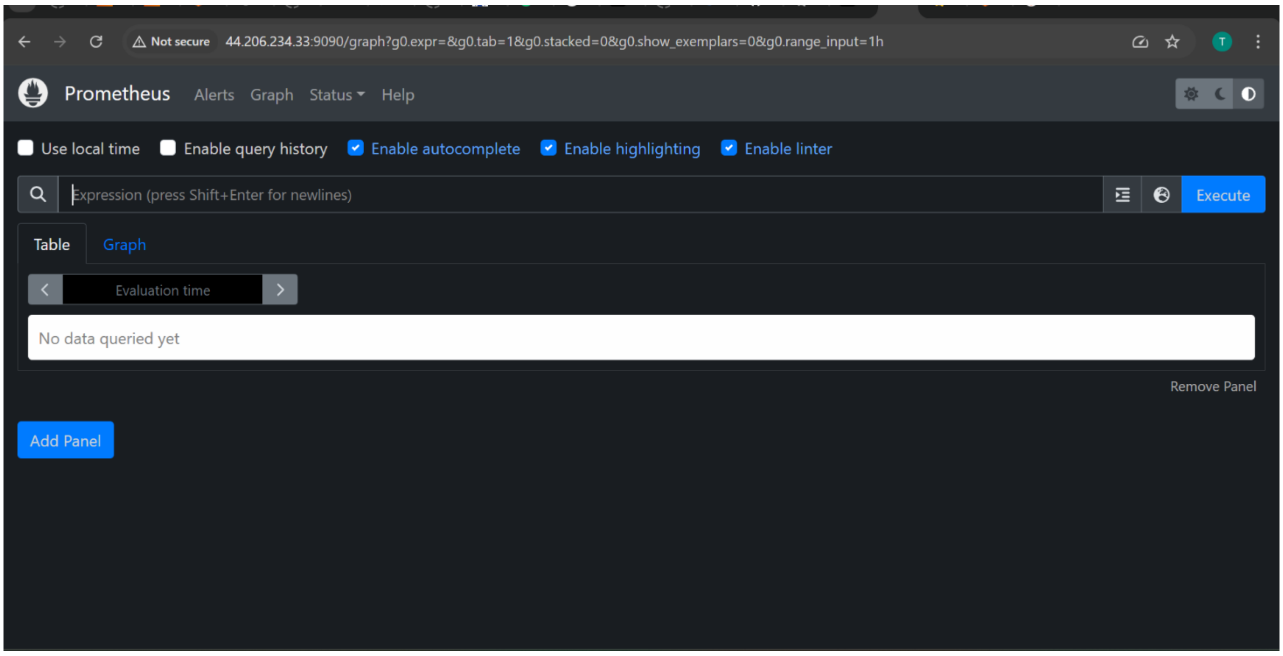
1. Deploy Prometheus and Grafana on an AWS EC2 instance.
2. Configure Prometheus to scrape metrics from various targets (Docker, Kubernetes, AWS services).
3. Create a Grafana dashboard to visualize system and application metrics.
4. Installation Guides:

- [Promethus Installation&Setup]
- [Grafana Setup]

Check :



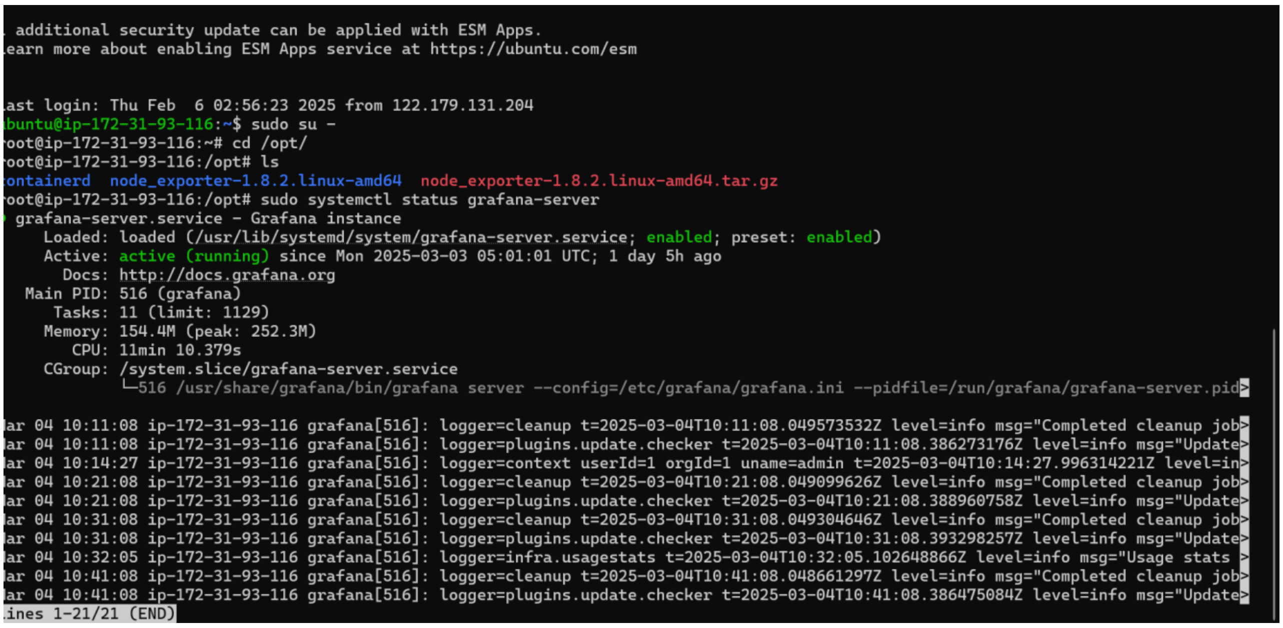
```
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***
Last login: Tue Mar  4 06:12:39 2025 from 152.58.35.205
ubuntu@ip-172-31-86-27:~$ sudo su -
root@ip-172-31-86-27:~# cd /opt/
root@ip-172-31-86-27:/opt# sudo systemctl status prometheus
● prometheus.service - Prometheus
     Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-03 18:28:20 UTC; 16h ago
   Main PID: 13154 (prometheus)
      Tasks: 8 (limit: 1129)
     Memory: 101.9M (peak: 219.5M)
        CPU: 1min 5.913s
     CGroup: /system.slice/prometheus.service
             └─13154 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --storage.tsdb.path /var/lib/prometheus/>

Mar 04 05:00:06 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T05:00:06.182Z caller=compact.go:460 level=info component=tsdb msg=">
Mar 04 05:00:06 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T05:00:06.189Z caller=db.go:1548 level=info component=tsdb msg="Dele>
Mar 04 05:00:06 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T05:00:06.194Z caller=db.go:1548 level=info component=tsdb msg="Dele>
Mar 04 05:00:06 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T05:00:06.198Z caller=db.go:1548 level=info component=tsdb msg="Dele>
Mar 04 07:00:02 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T07:00:02.995Z caller=compact.go:519 level=info component=tsdb msg=">
Mar 04 07:00:02 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T07:00:02.999Z caller=head.go:1269 level=info component=tsdb msg="He>
Mar 04 07:00:03 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T07:00:03.000Z caller=checkpoint.go:100 level=info component=tsdb ms>
Mar 04 07:00:03 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T07:00:03.209Z caller=head.go:1241 level=info component=tsdb msg="WA>
Mar 04 09:00:02 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T09:00:02.967Z caller=compact.go:519 level=info component=tsdb msg=">
Mar 04 09:00:02 ip-172-31-86-27 prometheus[13154]: ts=2025-03-04T09:00:02.971Z caller=head.go:1269 level=info component=tsdb msg="He>
lines 1-20/20 (END)
```



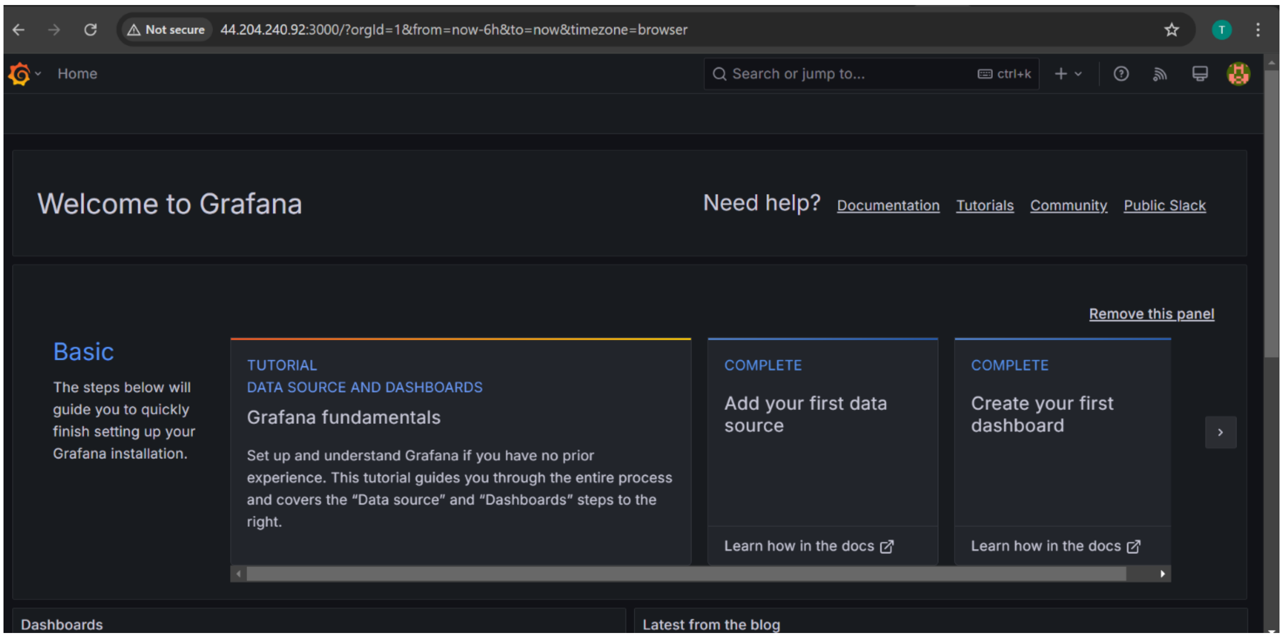
```
Not secure   44.206.234.33:9090/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h
```

Use local time [ ] Enable query history [ ] Enable autocomplete [✓] Enable highlighting [✓] Enable linter [✓]

No data queried yet



```
. additional security update can be applied with ESM Apps.
earn more about enabling ESM Apps service at https://ubuntu.com/esm


ast login: Thu Feb  6 02:56:23 2025 from 122.179.131.204
ibuntu@ip-172-31-93-116:~$ sudo su -
'oot@ip-172-31-93-116:~# cd /opt/
'oot@ip-172-31-93-116:/opt# ls
:ontainerd  node_exporter-1.8.2.linux-amd64  node_exporter-1.8.2.linux-amd64.tar.gz
'oot@ip-172-31-93-116:/opt# sudo systemctl status grafana-server
 grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-03 05:01:01 UTC; 1 day 5h ago
       Docs: http://docs.grafana.org
   Main PID: 516 (grafana)
      Tasks: 11 (limit: 1129)
     Memory: 154.4M (peak: 252.3M)
        CPU: 11min 10.379s
     CGroup: /system.slice/grafana-server.service
             └─516 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid>

lar 04 10:11:08 ip-172-31-93-116 grafana[516]: logger=cleanup t=2025-03-04T10:11:08.049573532Z level=info msg="Completed cleanup job>
lar 04 10:11:08 ip-172-31-93-116 grafana[516]: logger=plugins.update.checker t=2025-03-04T10:11:08.386273176Z level=info msg="Update>
lar 04 10:14:27 ip-172-31-93-116 grafana[516]: logger=context userId=1 orgId=1 uname=admin t=2025-03-04T10:14:27.996314221Z level=in>
lar 04 10:21:08 ip-172-31-93-116 grafana[516]: logger=cleanup t=2025-03-04T10:21:08.049099626Z level=info msg="Completed cleanup job>
lar 04 10:21:08 ip-172-31-93-116 grafana[516]: logger=plugins.update.checker t=2025-03-04T10:21:08.388960758Z level=info msg="Update>
lar 04 10:31:08 ip-172-31-93-116 grafana[516]: logger=cleanup t=2025-03-04T10:31:08.049304646Z level=info msg="Completed cleanup job>
lar 04 10:31:08 ip-172-31-93-116 grafana[516]: logger=plugins.update.checker t=2025-03-04T10:31:08.393298257Z level=info msg="Update>
lar 04 10:32:05 ip-172-31-93-116 grafana[516]: logger=infra.usagestats t=2025-03-04T10:32:05.102648866Z level=info msg="Usage stats >
lar 04 10:41:08 ip-172-31-93-116 grafana[516]: logger=cleanup t=2025-03-04T10:41:08.048661297Z level=info msg="Completed cleanup job>
lar 04 10:41:08 ip-172-31-93-116 grafana[516]: logger=plugins.update.checker t=2025-03-04T10:41:08.386475084Z level=info msg="Update>
.ines 1-21/21 (END)
```


Not secure 44.204.240.92:3000/?orgId=1&from=now-6h&to=now&timezone=browser

Home

Search or jump to... ctrl+k

Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

COMPLETE
Add your first data source
Learn how in the docs

COMPLETE
Create your first dashboard
Learn how in the docs

Dashboards

Latest from the blog

Step 2: Docker Monitoring with cAdvisor

1. Web Server for Testing

- Created a **web server** to generate HTTP requests and test monitoring.
- Verified metrics collection and visualization in Prometheus & Grafana.

2. Docker Server with cAdvisor for Container Monitoring

- **Deployed a Docker server** with multiple running containers.
- Installed **cAdvisor** to collect container metrics.
- Integrated **cAdvisor with Prometheus** for real-time container monitoring.

Installation: cAdvisor Setup

1. **Install Docker** on your server:

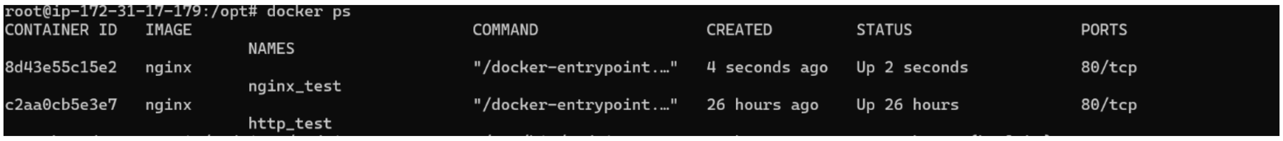
   ```
   sudo apt update && sudo apt install -y docker.io
   ```

2. **Start and enable the Docker service:**
3. sudo systemctl start docker
   sudo systemctl enable docker

4. **Create two sample containers:**
5. sudo docker run -d --name container1 nginx
   sudo docker run -d --name container2 httpd

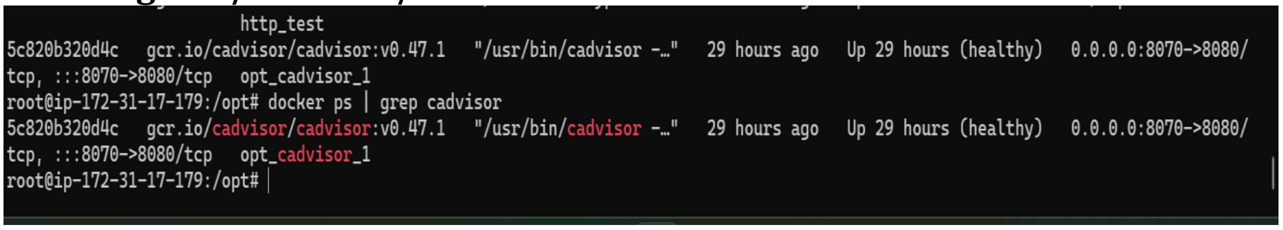
6. **Verify running containers:**

   ```
   sudo docker ps
   ```



```
root@ip-172-31-17-179:/opt# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS           PORTS
                         NAMES
8d43e55c15e2   nginx                                "/docker-entrypoint.…"   4 seconds ago    Up 2 seconds     80/tcp
                         nginx_test
c2aa0cb5e3e7   nginx                                "/docker-entrypoint.…"   26 hours ago     Up 26 hours      80/tcp
                         http_test
```

- **Install cAdvisor:**
- **sudo docker run -d --name=cadvisor \\
  --volume=/:/rootfs:ro \\
  --volume=/var/run:/var/run:rw \\
  --volume=/sys:/sys:ro \\
  --volume=/var/lib/docker/:/var/lib/docker:ro \\
  --publish=8080:8080 \\
  gcr.io/cadvisor/cadvisor**



```
                         http_test
5c820b320d4c   gcr.io/cadvisor/cadvisor:v0.47.1   "/usr/bin/cadvisor -…"   29 hours ago   Up 29 hours (healthy)   0.0.0.0:8070->8080/
tcp, :::8070->8080/tcp   opt_cadvisor_1
root@ip-172-31-17-179:/opt# docker ps | grep cadvisor
5c820b320d4c   gcr.io/cadvisor/cadvisor:v0.47.1   "/usr/bin/cadvisor -…"   29 hours ago   Up 29 hours (healthy)   0.0.0.0:8070->8080/
tcp, :::8070->8080/tcp   opt_cadvisor_1
root@ip-172-31-17-179:/opt#
```

7. **Access cAdvisor UI** at: http://<server-ip>:8080

Configuring Prometheus to Monitor cAdvisor

Edit the **Prometheus configuration file (prometheus.yml)** to add cAdvisor as a data source:

```
scrape_configs:
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['<server-ip>:8080']
```

Restart Prometheus for changes to take effect:

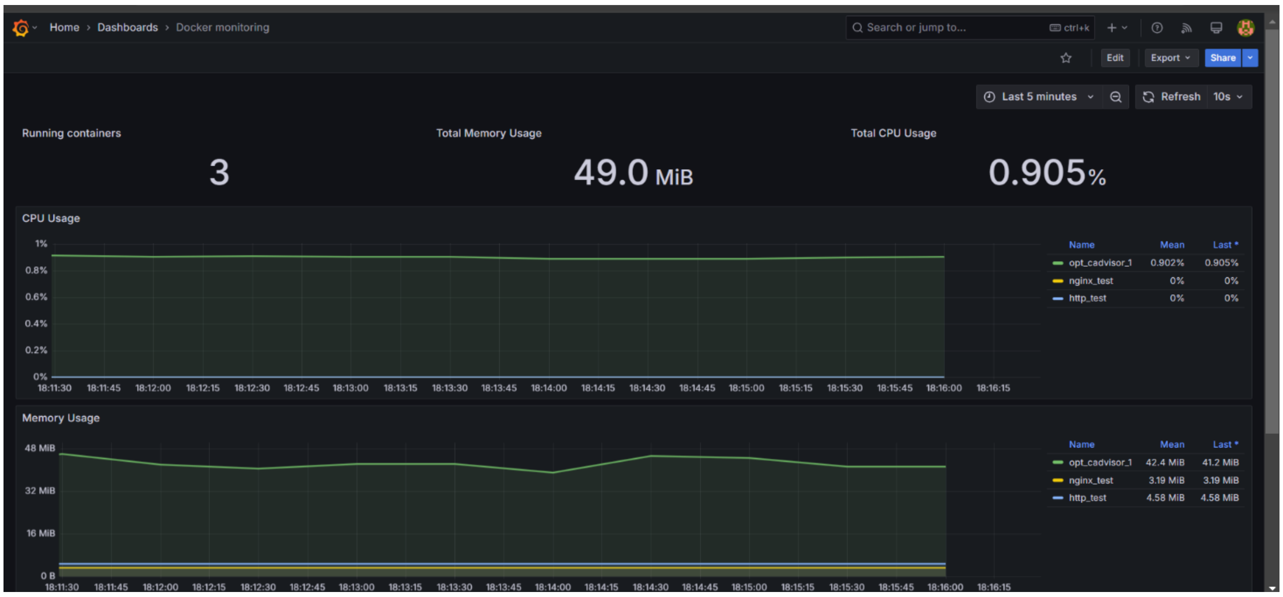
```
sudo systemctl restart prometheus
```

Import Grafana Dashboard for Monitoring

1. **Login to Grafana** (http://<server-ip>:3000)
2. Navigate to **Dashboards → Import**
3. Enter **Dashboard ID: 193**
4. Select **Prometheus as the data source**
5. Click **Import**

Monitoring Containers in Grafana

- **Visualize CPU, memory, network, and disk usage** of running containers.
- **Monitor individual container performance** in real-time.
- **Identify resource consumption trends** and optimize deployments accordingly.



Running containers 3

Total Memory Usage 49.0 MiB

Total CPU Usage 0.905%

CPU Usage

Name	Mean	Last *
opt_cadvisor_1	0.902%	0.905%
nginx_test	0%	0%
http_test	0%	0%

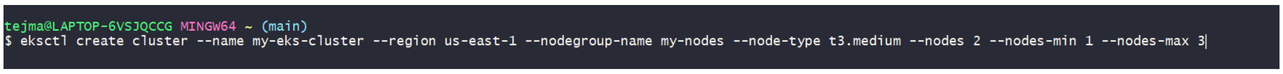
Memory Usage

Name	Mean	Last *
opt_cadvisor_1	42.4 MiB	41.2 MiB
nginx_test	3.19 MiB	3.19 MiB
http_test	4.58 MiB	4.58 MiB

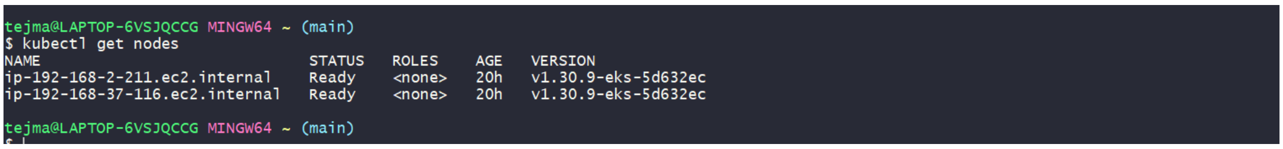
Step 3: Kubernetes (AWS EKS) Setup and Monitoring

1. AWS EKS Setup

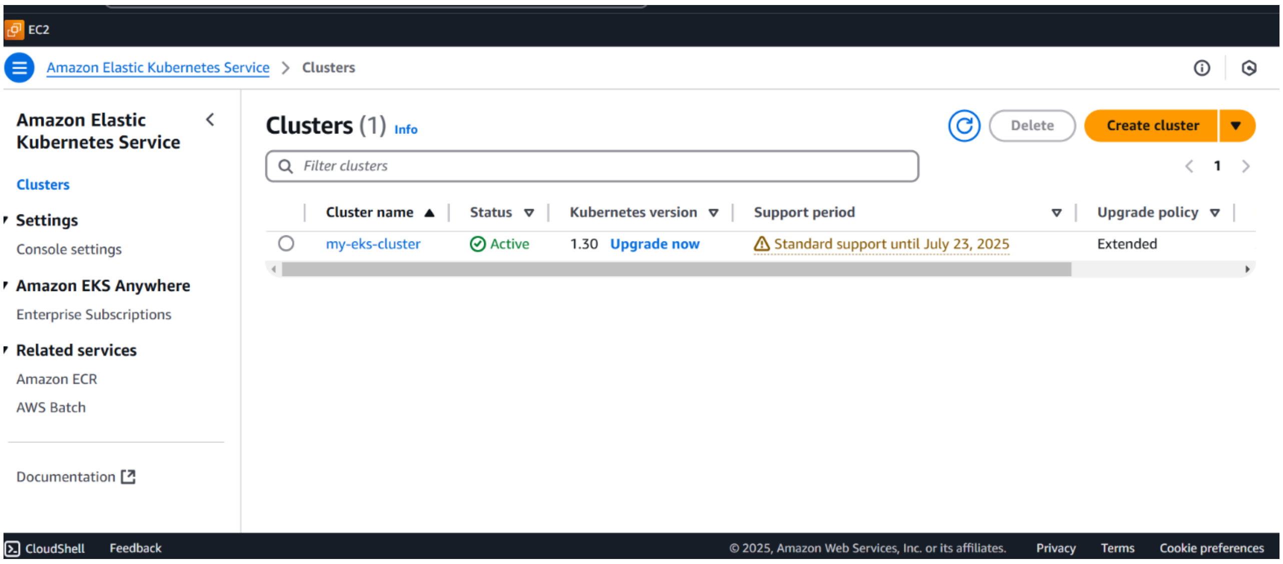
- Set up Amazon EKS for containerized application deployment.
- Installed Prometheus on EKS to monitor cluster health.
- Configured Grafana dashboards for Kubernetes metrics visualization.



```
tejma@LAPTOP-6VSJQCCG MINGW64 ~ (main)
$ eksctl create cluster --name my-eks-cluster --region us-east-1 --nodegroup-name my-nodes --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 3
```

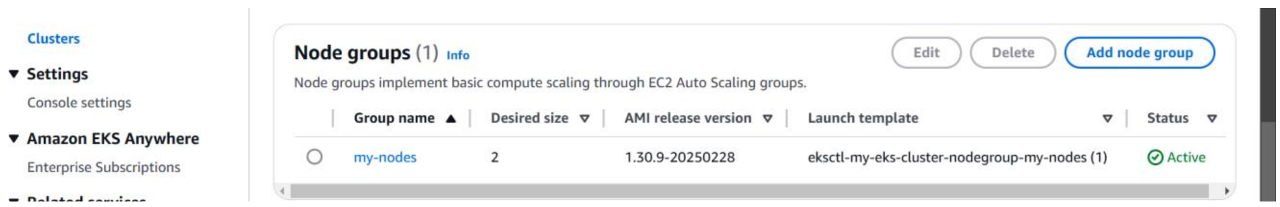


```
tejma@LAPTOP-6VSJQCCG MINGW64 ~ (main)
$ kubectl get nodes
NAME                           STATUS   ROLES    AGE   VERSION
ip-192-168-2-211.ec2.internal  Ready    <none>   20h   v1.30.9-eks-5d632ec
ip-192-168-37-116.ec2.internal Ready    <none>   20h   v1.30.9-eks-5d632ec

tejma@LAPTOP-6VSJQCCG MINGW64 ~ (main)
$
```



Clusters (1) Info

Cluster name	Status	Kubernetes version	Support period	Upgrade policy
my-eks-cluster	Active	1.30 Upgrade now	Standard support until July 23, 2025	Extended



Node groups (1)

Node groups implement basic compute scaling through EC2 Auto Scaling groups.

Group name	Desired size	AMI release version	Launch template	Status
my-nodes	2	1.30.9-20250228	eksctl-my-eks-cluster-nodegroup-my-nodes (1)	Active

2. Steps to Set Up Cluster in AWS CLI

Step 1: Create Namespace for Monitoring

```
kubectl create namespace monitoring
```

Step 2: Add Prometheus Helm Repository

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

Step 3: Install Prometheus on EKS

```
helm install prometheus prometheus-community/prometheus \
  --set alertmanager.persistentVolume.enabled=false \
  --set server.persistentVolume.enabled=false \
  --set server.service.type=NodePort \
  --set server.service.nodePort=30090 -n monitoring
```

Step 4: Verify Prometheus Installation

```
kubectl get pods -n monitoring
```

Step 5: Grafana Dashboard Setup

- Imported Kubernetes Cluster Monitoring Dashboards using Grafana Dashboard IDs: 15760 & 15759.

Conclusion

This project demonstrates a complete infrastructure monitoring setup using Prometheus and Grafana, covering Docker container monitoring with cAdvisor and Kubernetes (AWS EKS) cluster monitoring. The integration of Loki for log monitoring ensures centralized observability.

Through dynamic dashboards in Grafana, this monitoring solution enables real-time system performance tracking, proactive issue detection, and resource optimization for cloud-native applications.