Lesson 04 Demo 05

Setting up Docker Monitoring Using Prometheus and Grafana

Objective: To set up Docker monitoring using Prometheus and Grafana for visualizing system and container metrics through a Grafana dashboard

Tools required: Linux operating system, Docker, and Docker Compose

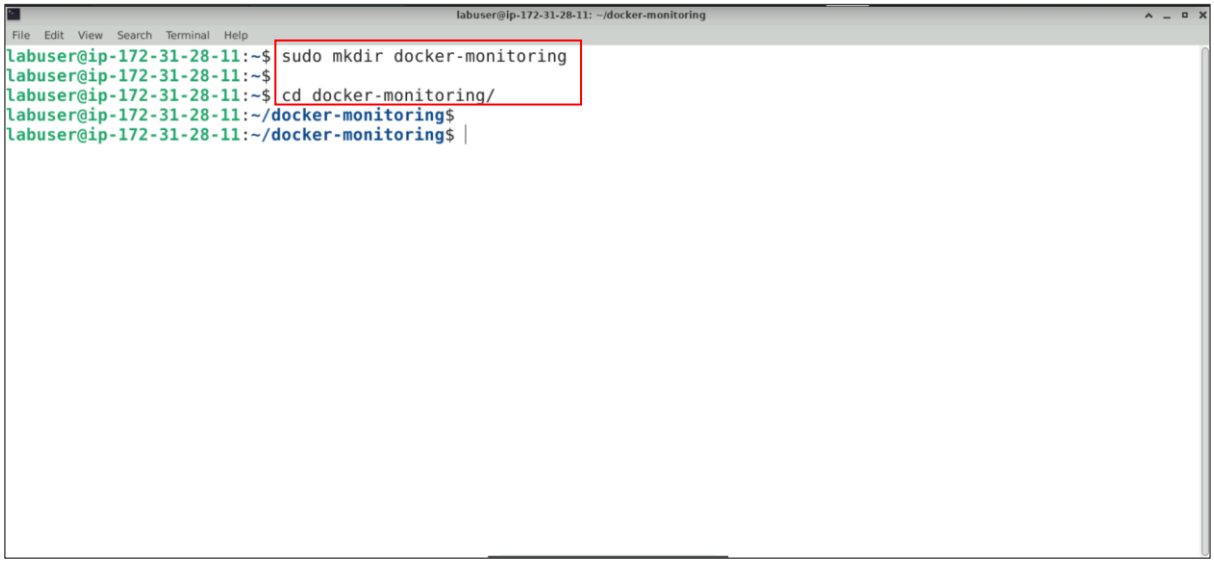
Prerequisites: A basic understanding of Prometheus and Grafana
Refer to Demo 02 from Lesson 04 for creating multi-panels

Steps to be followed:

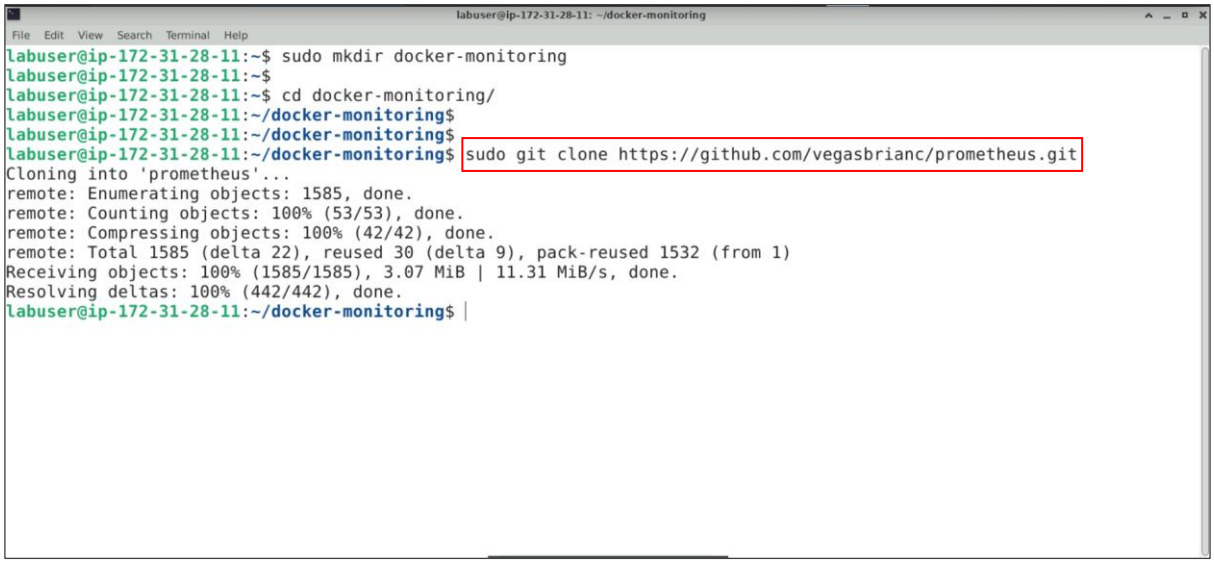
1. Configure the Prometheus monitoring stack with Docker
2. Visualize Docker and host metrics in Grafana
3. Create custom panels for Docker metrics

Step 1: Configure the Prometheus monitoring stack with Docker

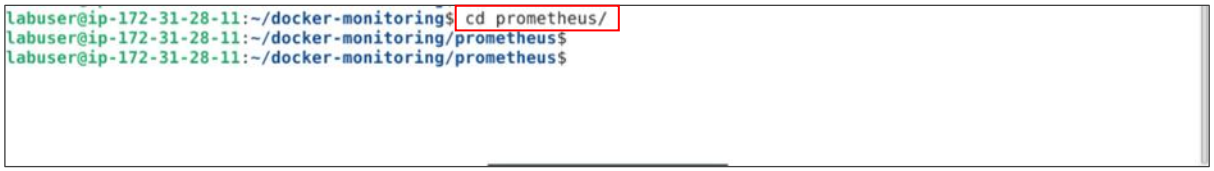
1.1 Open Terminal, create a new directory, and navigate to it using the following commands:
sudo mkdir docker-monitoring
cd docker-monitoring



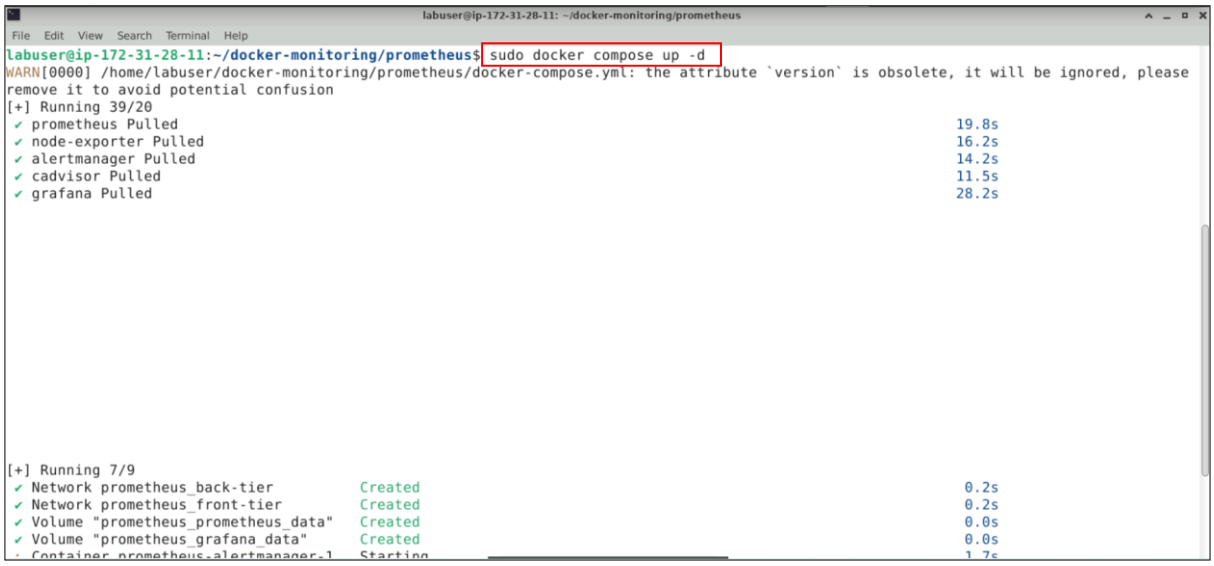
1.2 Clone the Prometheus monitoring setup repository from GitHub using the following command:
sudo git clone https://github.com/vegasbrianc/prometheus.git



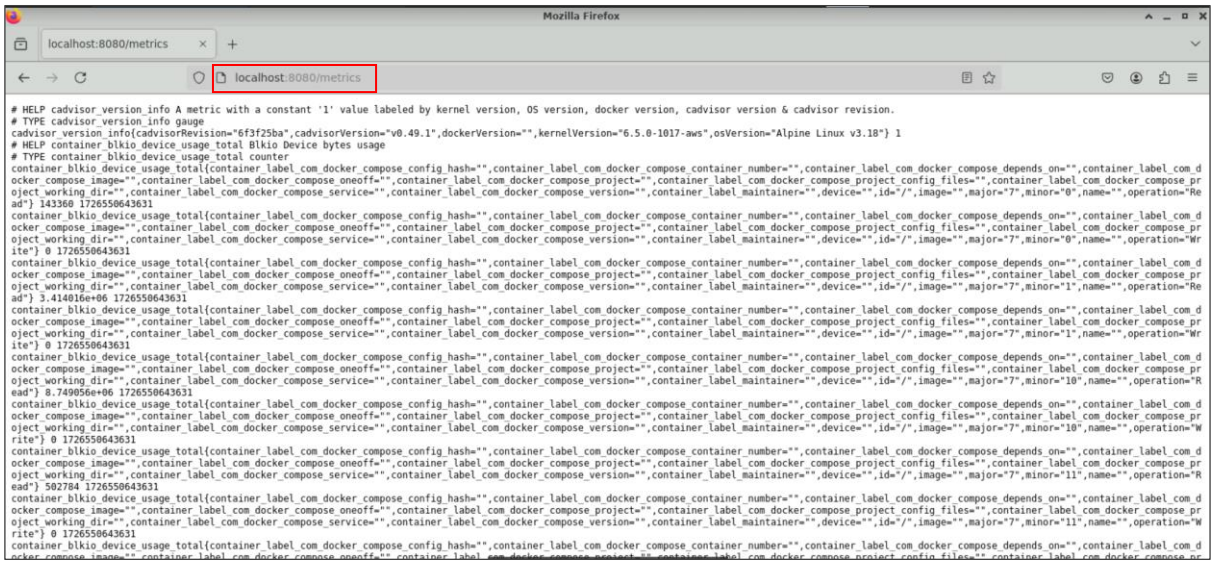
1.3 Change the current directory to the newly cloned **prometheus** directory using the following command:
cd prometheus/



1.4 Start Docker containers using the following command:
sudo docker compose up -d



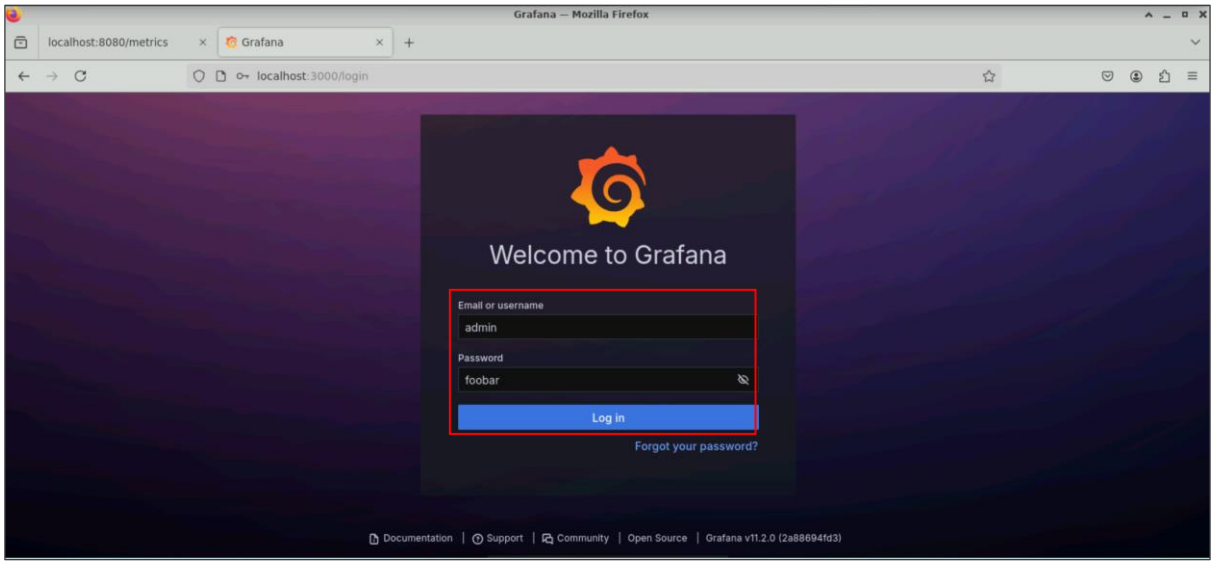
1.5 Open the browser and access the cAdvisor Docker metrics using the following URL: **http://localhost:8080/metrics** as shown below:



Note: If you encounter any port conflict issues, run the command **sudo lsof -i:<port_number>** to check which process is using the port. Then, stop or kill the process using **sudo systemctl stop <process_name>** or **sudo kill -9 <process_PID>**

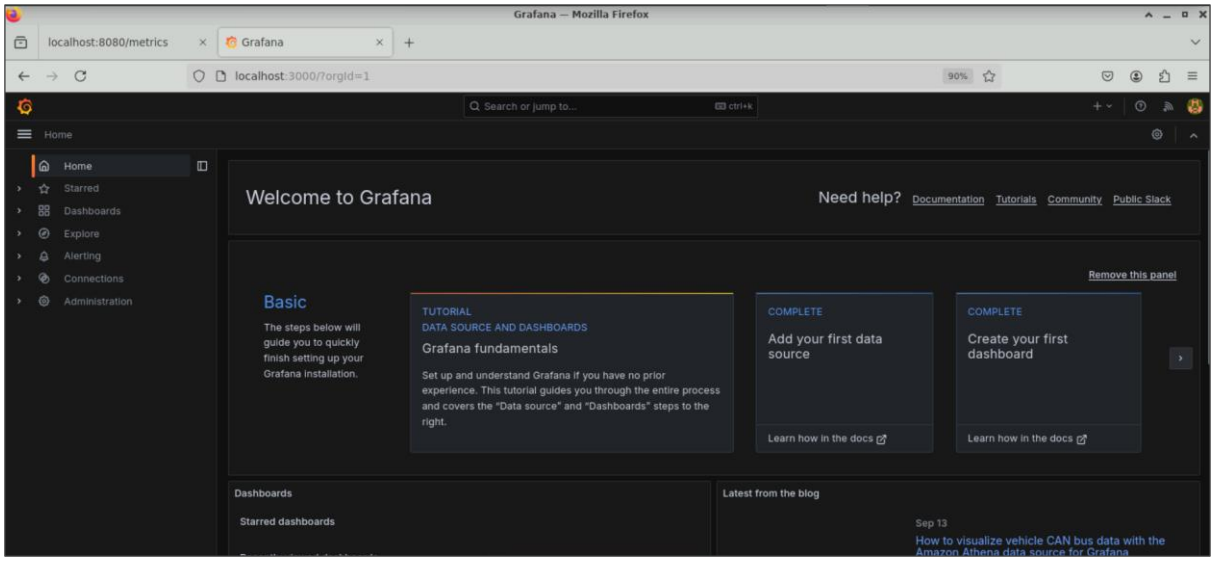
Step 2: Visualize Docker and host metrics in Grafana

2.1 Open the Grafana console using **http://localhost:3000** in the browser. Enter the username and password as **admin** and click on **Log in**.

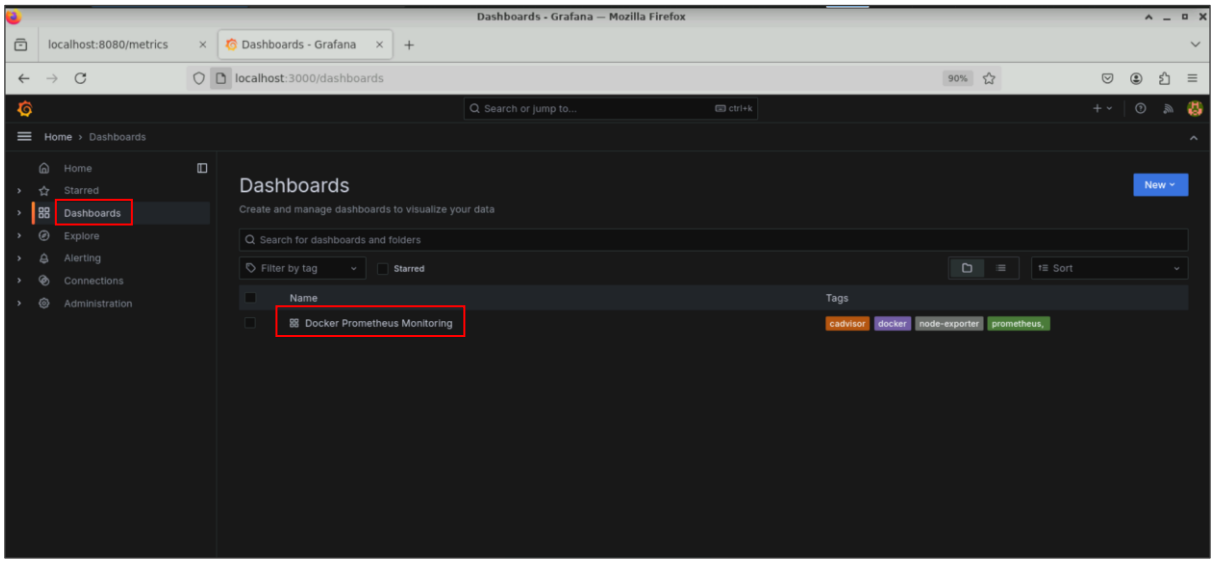


Note: If you want to verify the password, you can either access the URL **https://github.com/vegasbrianc/prometheus** or change the directory to **docker-monitoring/prometheus/grafana** and run the command **cat config.monitoring**

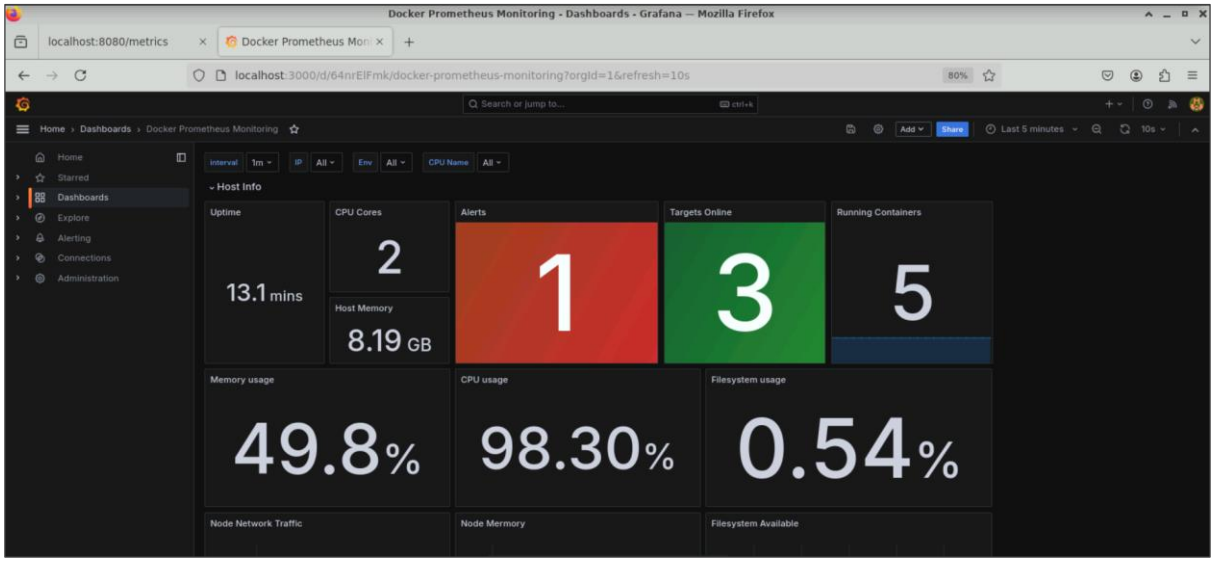
The Grafana dashboard appears as shown below:

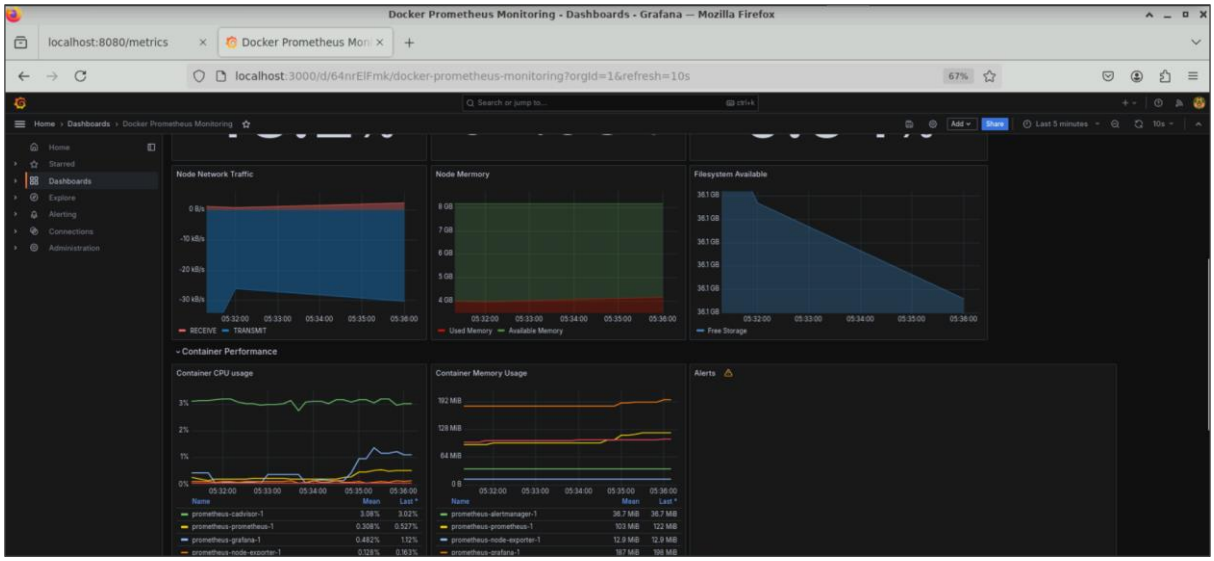


2.2 In the left-side menu, click **Dashboards**, then select the created dashboard **Docker Prometheus Monitoring**



The dashboard displays Docker and system metrics as shown below:

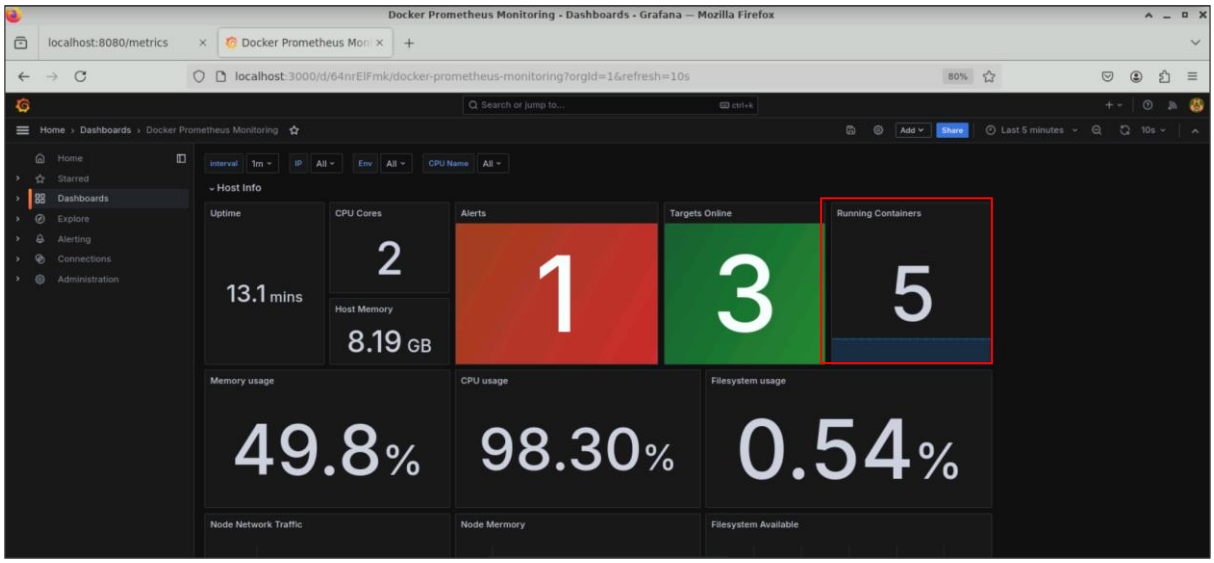




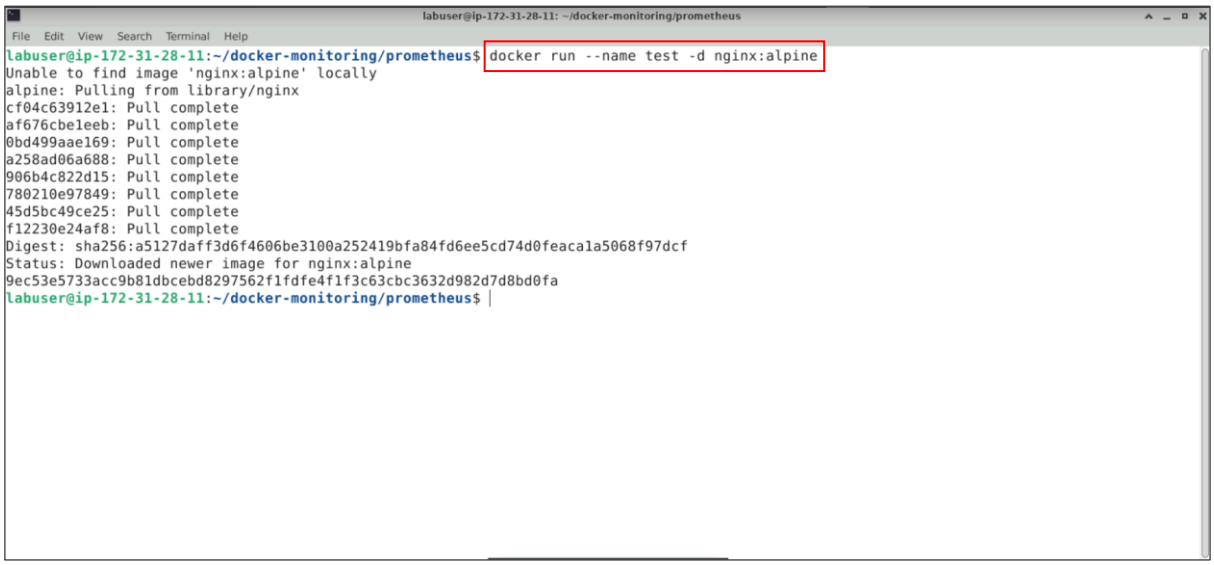
Note: The first section represents system metrics with gauges and text or graphs that are minimalistic and easy to visualize.
System or host metrics: Time up, memory usage/swap, disk usage, load, network, CPU usage, and disk I/O
Alerts are set for disk usage, memory usage, and load usage to warn when the metrics reach critical levels.

The second part of the dashboard focuses on Docker metrics, displayed in more detail, mostly using graphs.
Docker metrics: CPU usage per container, sent network per container, received network per container, and memory usage/swap per container

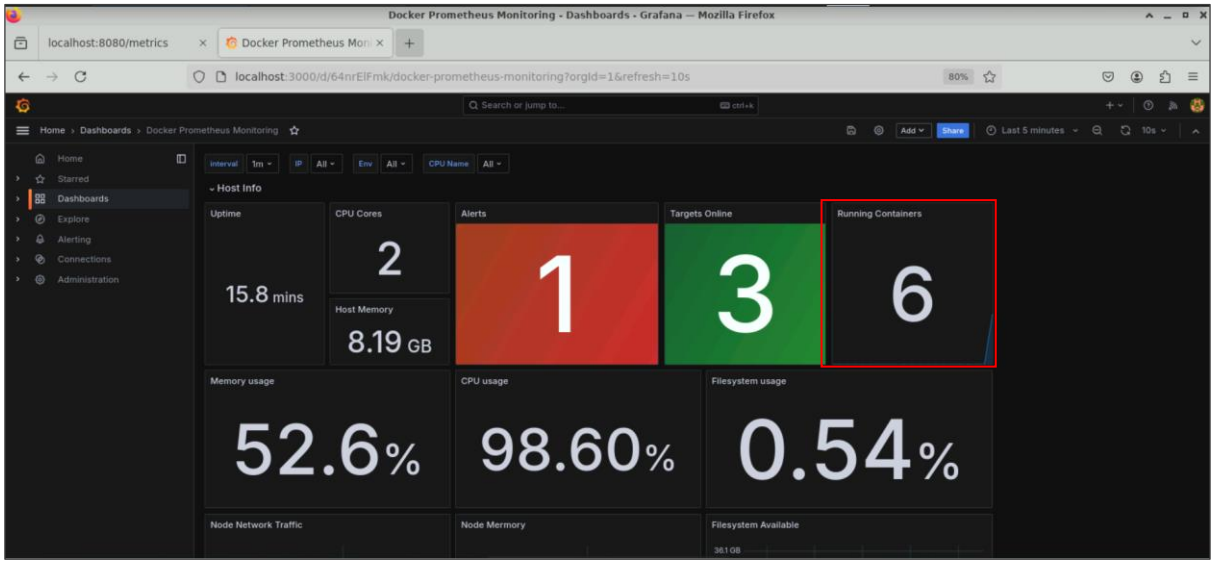
By default, 5 containers are displayed in the **Running Containers** section of the dashboard panel.



2.3 Navigate to the terminal and run the following command to start a new Docker container:
docker run --name test -d nginx:alpine

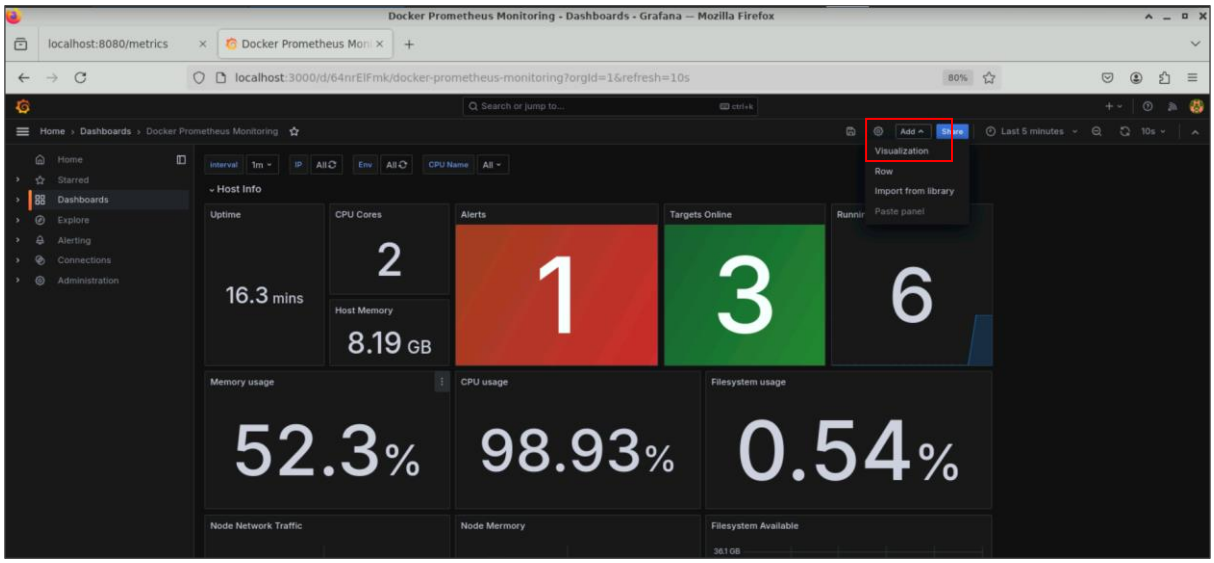


2.4 Refresh the Grafana dashboard and observe the count of **Running Containers**

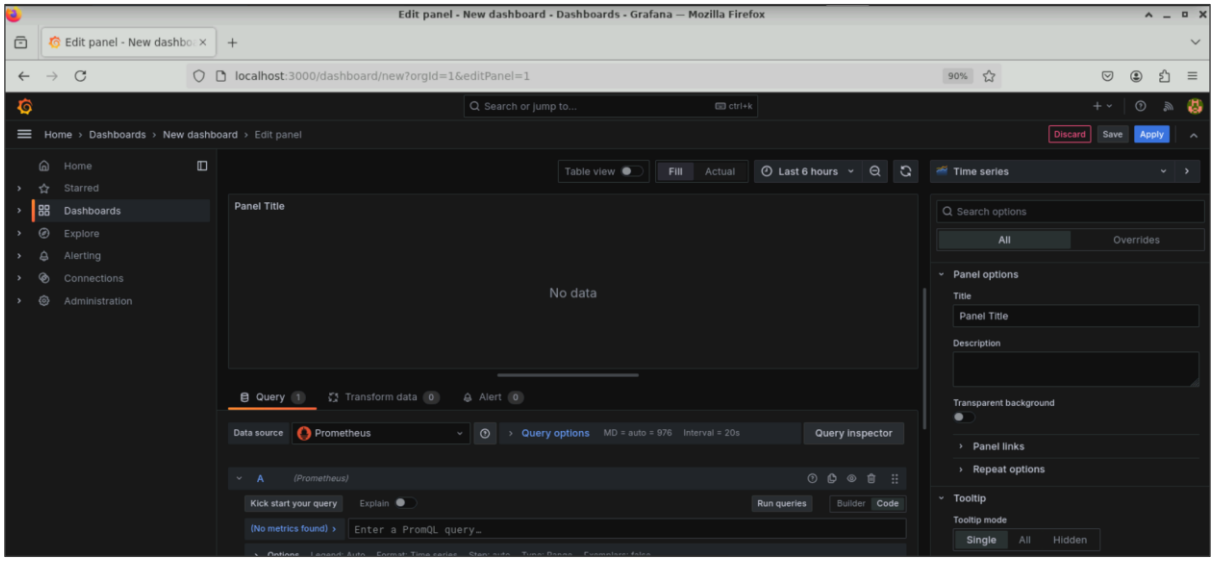


Step 3: Create custom panels for Docker metrics

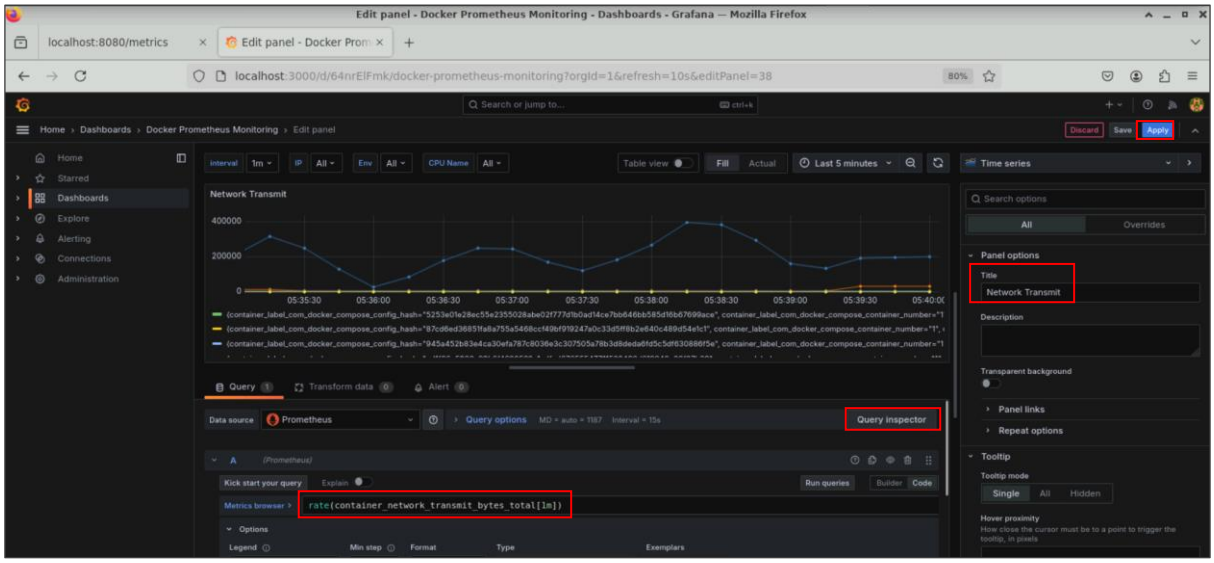
3.1 Click on **Add** and then select **Visualization** to add more panels



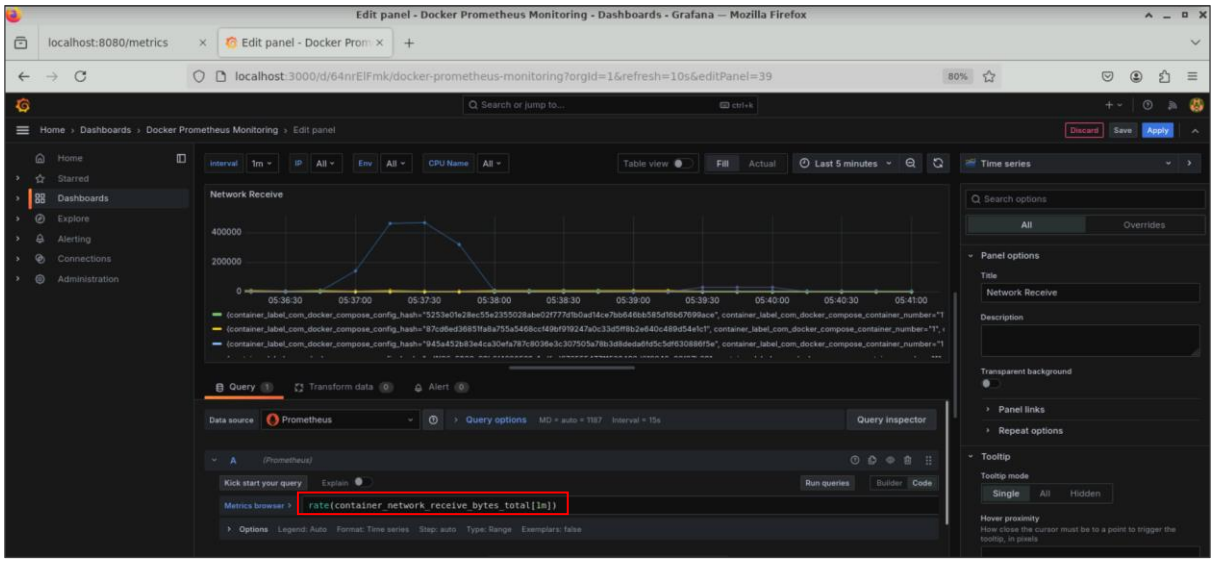
A new panel appears as shown below:



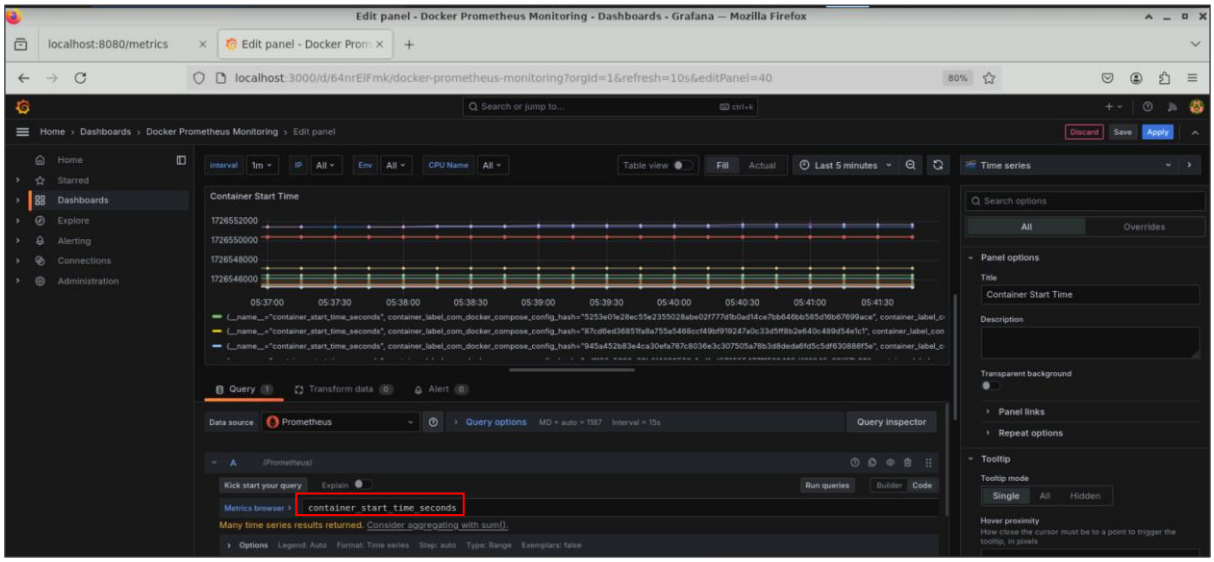
3.2 Change the panel name, run the following query, observe the graph, and click **Apply** to save the changes:
rate(container_network_transmit_bytes_total[1m])



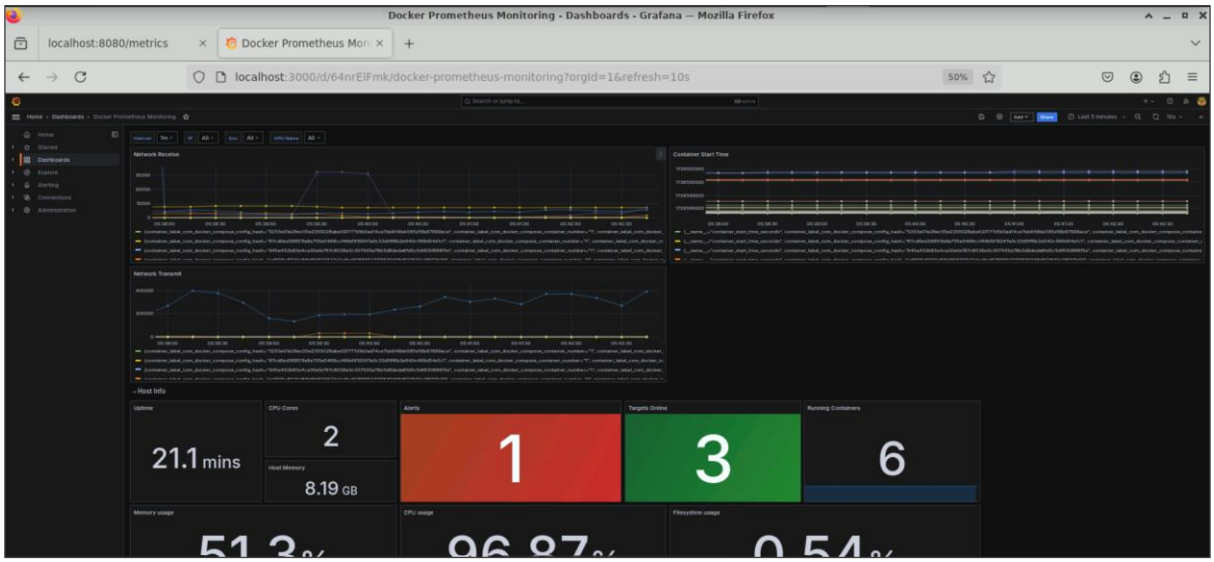
3.3 Add another panel, run the following query, and save it:
rate(container_network_receive_bytes_total[1m])



3.4 Add more panels by executing the following query:
container_start_time_seconds



The newly added panels appear at the top of the Docker container dashboard as shown below:



By following these steps, you have successfully configured Docker containers for Prometheus, Grafana, Node Exporter, and cAdvisor to visualize Docker and system metrics, create new panels, and monitor various metrics.