Placement Empowerment Program

Cloud Computing and DevOps Centre

Visualize Cloud Application Metrics with Grafana: Connect Grafana to Prometheus and create dashboards for monitoring CPU, memory, and HTTP requests.

Name: Saravana Krishnan J Department: IT

Introduction

In cloud-based environments, monitoring system performance is essential to maintain application availability, optimize resource usage, and prevent failures. **Prometheus** is a powerful open-source monitoring and alerting toolkit that collects time-series data, while **Grafana** provides an interactive interface to visualize and analyze this data.

This PoC focuses on setting up **Prometheus and Grafana** on an **AWS EC2 instance** running **Ubuntu** to monitor system metrics such as **CPU usage, memory consumption, and network traffic**

Overview

This PoC demonstrates how to:

1. Install and configure **Prometheus** on an AWS EC2 instance.
2. Set up **Node Exporter** to collect system-level metrics (CPU, RAM, Disk, Network).
3. Install and configure **Grafana** for visualizing Prometheus metrics.
4. Add **Prometheus as a data source** in Grafana.
5. Run queries in Grafana to view real-time monitoring data.

By completing this PoC, you will have a working **monitoring setup** that collects, stores, and visualizes system performance data in **real-time**.

Objective

The primary goals of this PoC are:

1. Set up **Prometheus** to scrape system metrics.

2. Configure **Node Exporter** to collect CPU, memory, and network statistics.

3. Install **Grafana** and integrate it with Prometheus.

4. Learn how to write **PromQL queries** to analyze metrics.

5. Verify system monitoring by querying network traffic data.

At the end of this PoC, we will have a functional **monitoring stack** with **Prometheus for data collection** and **Grafana for visualization**.

Importance

1. **Real-time Monitoring** – Track system health and performance metrics in real-time.

2. **Early Issue Detection** – Detect anomalies before they lead to failures.

3. **Historical Data Analysis** – Store and analyze performance trends over time.

4. **Scalability** – Easily integrates with cloud environments like AWS, Kubernetes, and Docker.

5. **Alerting & Notifications** – Set up alerts for system issues.

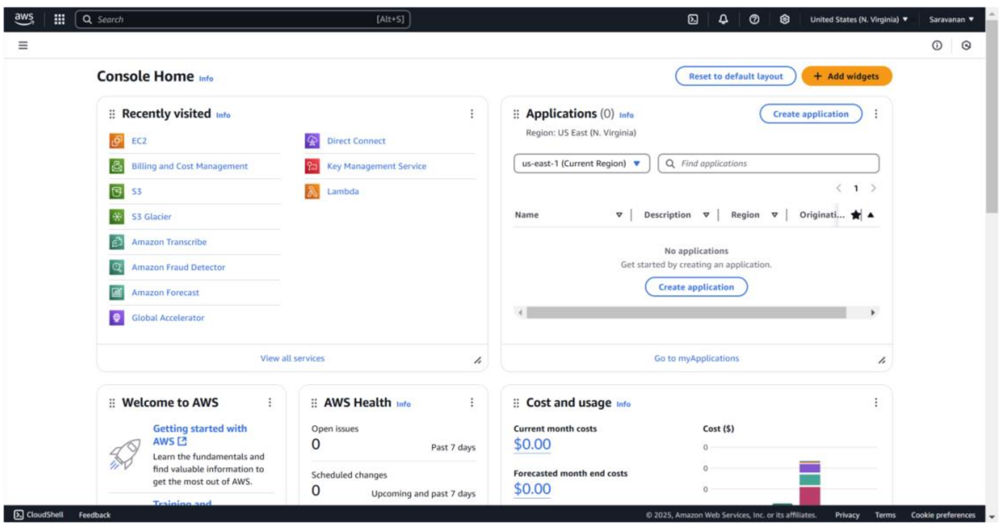
6. **Open-source & Cost-effective** – Eliminates the need for expensive monitoring tools.

Step-by-Step Overview

Step 1:

1. Go to AWS Management Console.

2. Enter your username and password to log in.



Step 2:

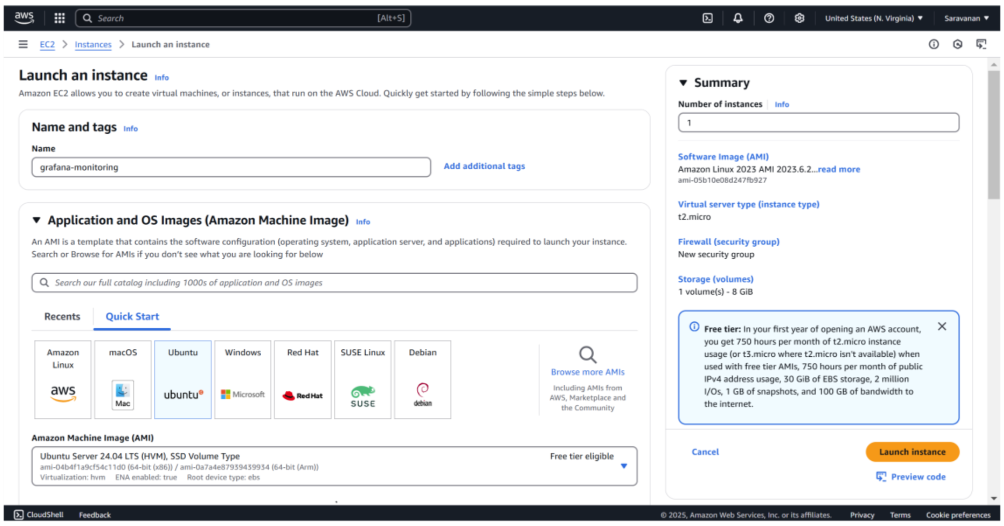
1. Navigate to **EC2** → **Launch Instance**.

2. Choose an **Ubuntu** OS.

3. Configure the security group:

   Allow inbound rules for:

   - SSH (Port **22**) → Your IP
   - Node Exporter **(9100)** → Anywhere
   - Prometheus (Port **9090**) → Anywhere



EC2 > Instances > Launch an instance

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: grafana-monitoring

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux | Debian

Browse more AMIs — Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-04b4f1a9cf54c11d0 (64-bit (x86)) / ami-0a7a4e87939439934 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.6.2...read more
ami-05b10e08d247fb927

Virtual server type (instance type): t2.micro

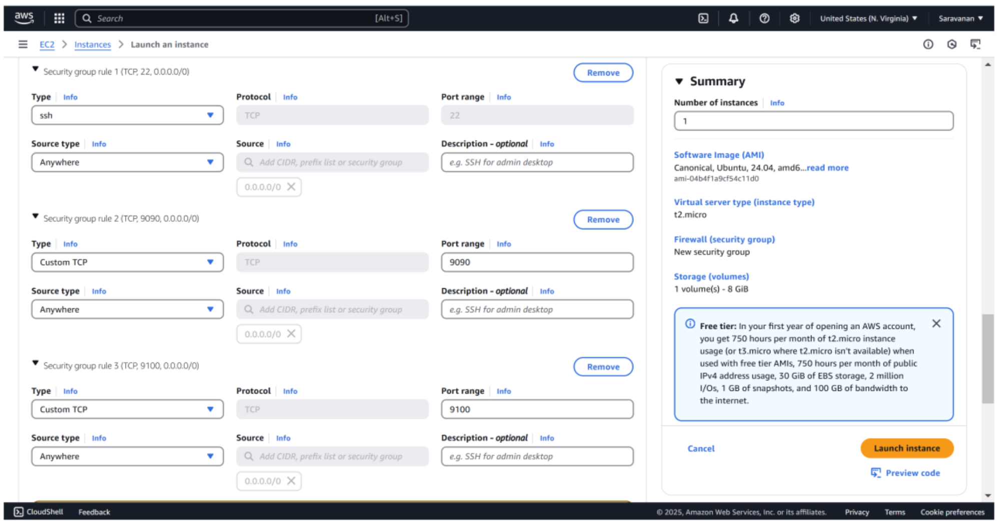
Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel | Launch instance | Preview code

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



EC2 > Instances > Launch an instance

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type	Protocol	Port range	Source type	Source	Description - optional
ssh	TCP	22	Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

Security group rule 2 (TCP, 9090, 0.0.0.0/0)

Type	Protocol	Port range	Source type	Source	Description - optional
Custom TCP	TCP	9090	Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

Security group rule 3 (TCP, 9100, 0.0.0.0/0)

Type	Protocol	Port range	Source type	Source	Description - optional
Custom TCP	TCP	9100	Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-04b4f1a9cf54c11d0

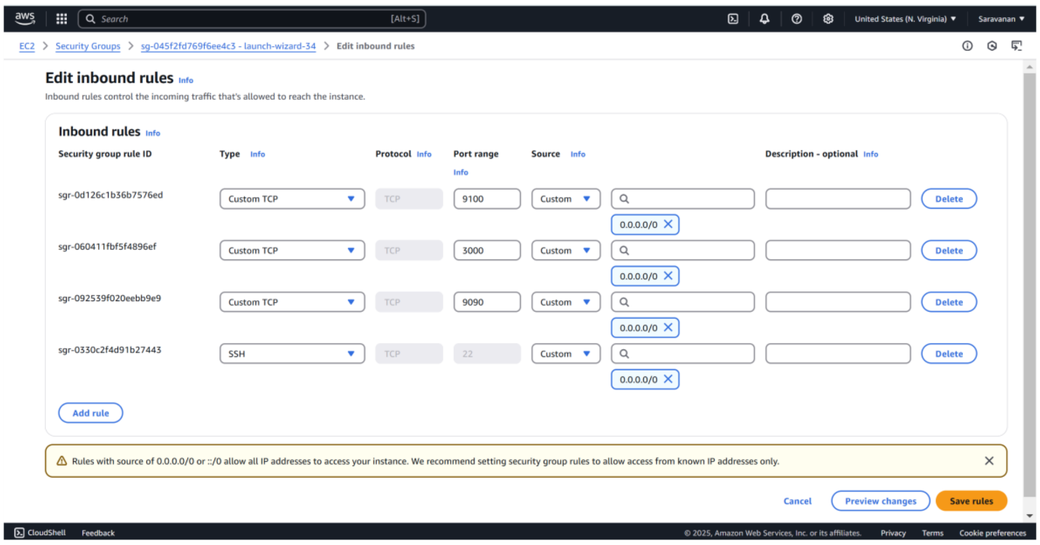
Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel | Launch instance | Preview code



EC2 > Security Groups > sg-045f2fd769f6ee4c3 - launch-wizard-34 > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0d126c1b36b7576ed	Custom TCP	TCP	9100	Custom	0.0.0.0/0	
sgr-060411fbf5f4896ef	Custom TCP	TCP	3000	Custom	0.0.0.0/0	
sgr-092539f020eebb9e9	Custom TCP	TCP	9090	Custom	0.0.0.0/0	
sgr-0330c2f4d91b27443	SSH	TCP	22	Custom	0.0.0.0/0	

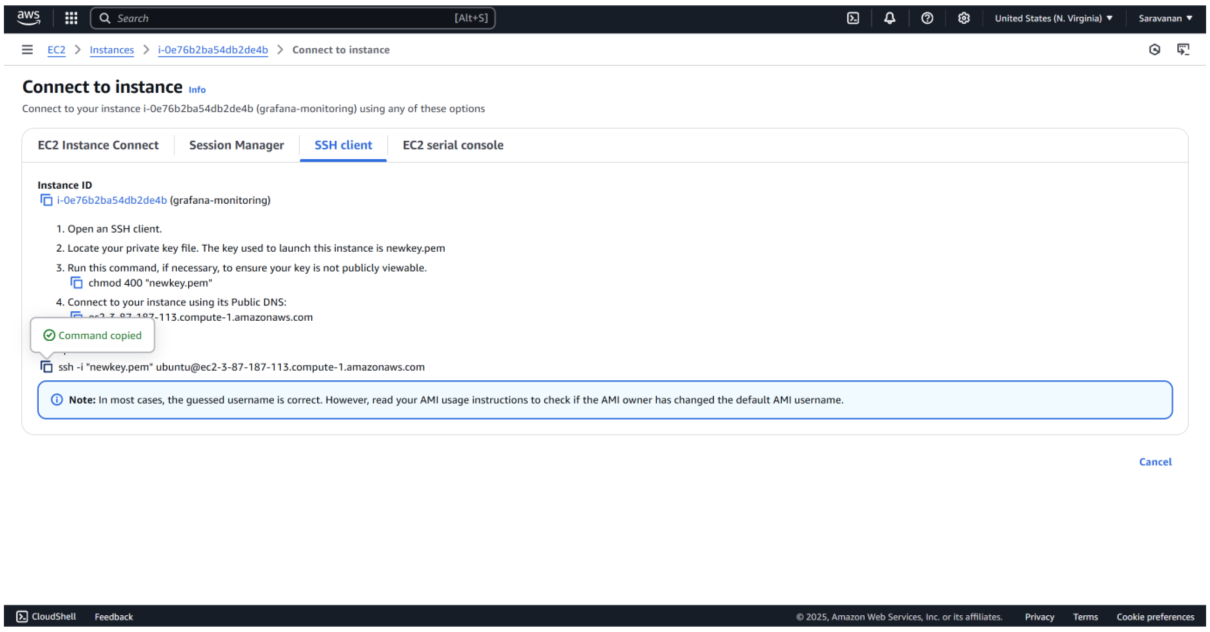
Add rule

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel | Preview changes | Save rules

Step 3:

Connect to the instance via SSH in Command prompt.


aws
Search [Alt+S]
United States (N. Virginia)
Saravanan
EC2 > Instances > i-0e76b2ba54db2de4b > Connect to instance

Connect to instance Info

Connect to your instance i-0e76b2ba54db2de4b (grafana-monitoring) using any of these options

Instance ID

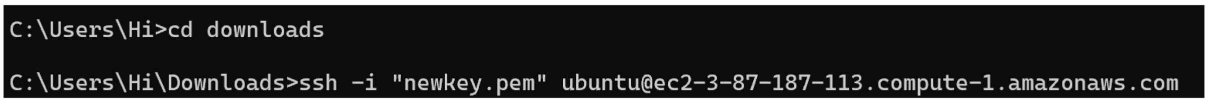
i-0e76b2ba54db2de4b (grafana-monitoring)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is newkey.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "newkey.pem"
4. Connect to your instance using its Public DNS:
   ec2-3-87-187-113.compute-1.amazonaws.com

Command copied

ssh -i "newkey.pem" ubuntu@ec2-3-87-187-113.compute-1.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

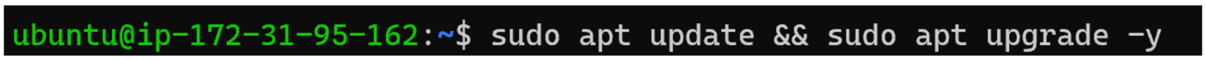


```
C:\Users\Hi>cd downloads

C:\Users\Hi\Downloads>ssh -i "newkey.pem" ubuntu@ec2-3-87-187-113.compute-1.amazonaws.com
```

Step 4:

Run the following command to update the package list and upgrade existing packages:

sudo apt update && sudo apt upgrade -y

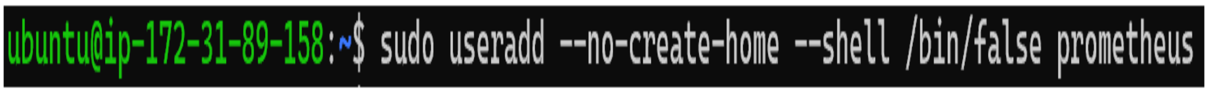


```
ubuntu@ip-172-31-95-162:~$ sudo apt update && sudo apt upgrade -y
```

Step 5:

Create a Prometheus user

sudo useradd --no-create-home --shell /bin/false Prometheus

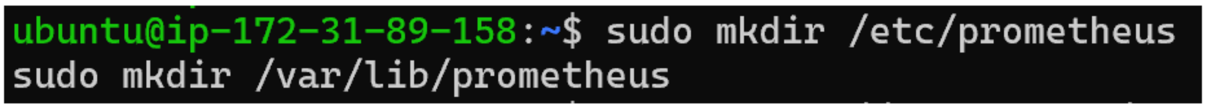


```
ubuntu@ip-172-31-89-158:~$ sudo useradd --no-create-home --shell /bin/false prometheus
```

Step 6:

Create required directories

sudo mkdir /etc/prometheus
sudo mkdir /var/lib/Prometheus

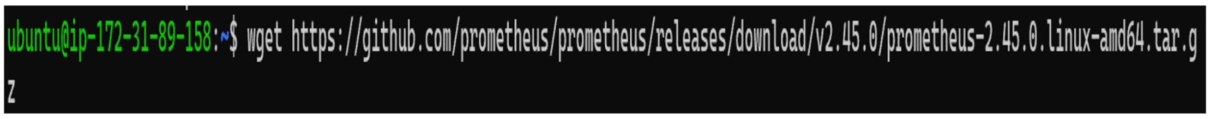


```
ubuntu@ip-172-31-89-158:~$ sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
```

Step 7:

Download Prometheus

https://github.com/prometheus/prometheus/releases/download/v2.45.0/prometheus-2.45.0.linux-amd64.tar.gz

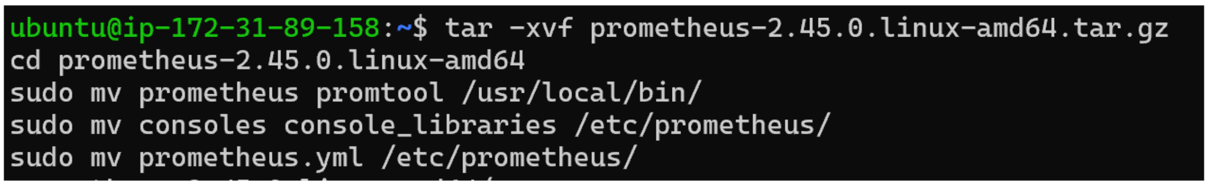


```
ubuntu@ip-172-31-89-158:~$ wget https://github.com/prometheus/prometheus/releases/download/v2.45.0/prometheus-2.45.0.linux-amd64.tar.g
z
```

Step 8:

Extract and move files

tar -xvf prometheus-2.45.0.linux-amd64.tar.gz
cd prometheus-2.45.0.linux-amd64
sudo mv prometheus promtool /usr/local/bin/
sudo mv consoles console_libraries /etc/prometheus/
sudo mv prometheus.yml /etc/prometheus/

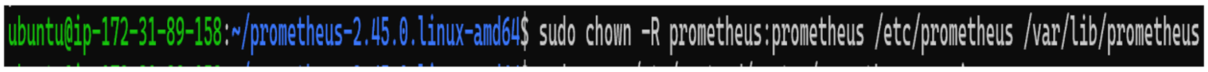


```
ubuntu@ip-172-31-89-158:~$ tar -xvf prometheus-2.45.0.linux-amd64.tar.gz
cd prometheus-2.45.0.linux-amd64
sudo mv prometheus promtool /usr/local/bin/
sudo mv consoles console_libraries /etc/prometheus/
sudo mv prometheus.yml /etc/prometheus/
```

Step 9:

Set permissions

sudo chown -R prometheus:prometheus /etc/prometheus /var/lib/Prometheus

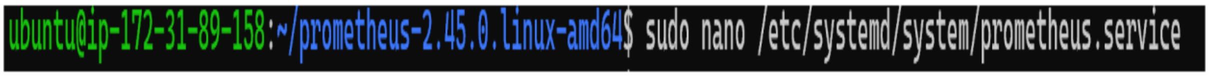


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64$ sudo chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
```

Step 10:

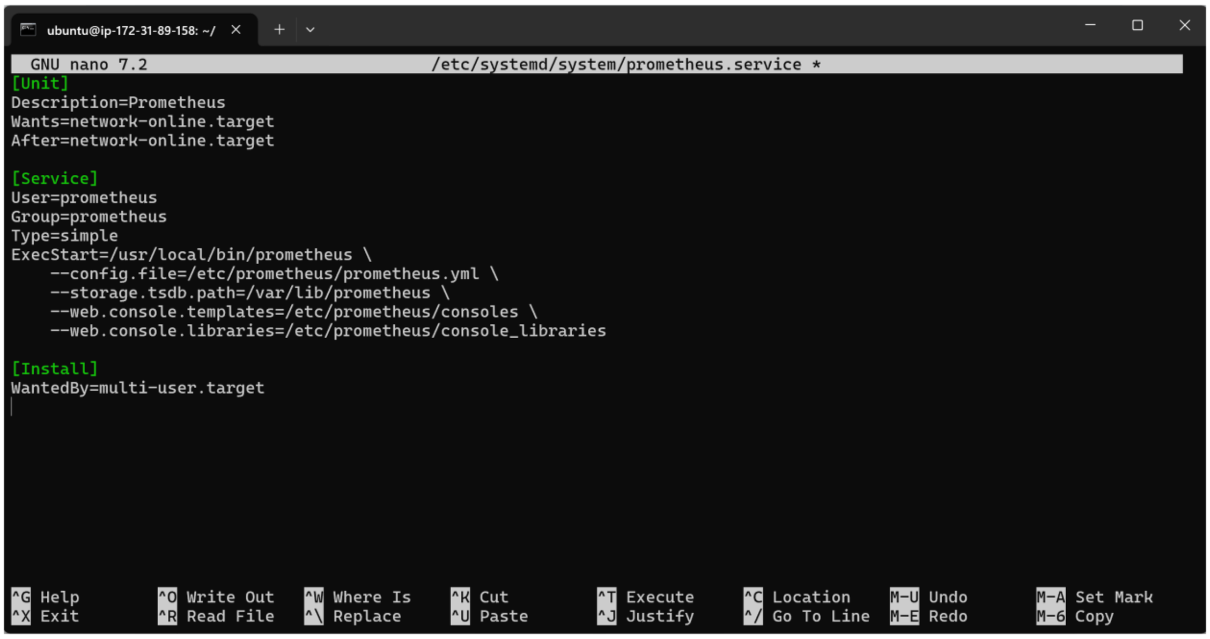
1. Create a systemd service file

sudo nano /etc/systemd/system/prometheus.service



```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64$ sudo nano /etc/systemd/system/prometheus.service
```

2. Paste this content inside the file:



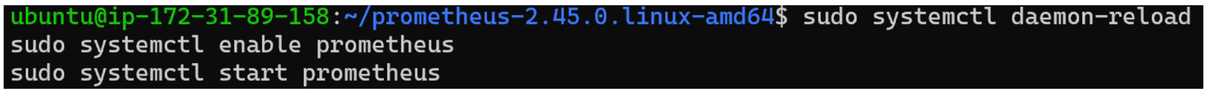
```
ubuntu@ip-172-31-89-158: ~/
  GNU nano 7.2                    /etc/systemd/system/prometheus.service *
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
    --config.file=/etc/prometheus/prometheus.yml \
    --storage.tsdb.path=/var/lib/prometheus \
    --web.console.templates=/etc/prometheus/consoles \
    --web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Press **CTRL + O** and then **Enter**. Then Press **Ctrl+X** .

Step 11:

Reload systemd and start Prometheus

sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl start Prometheus

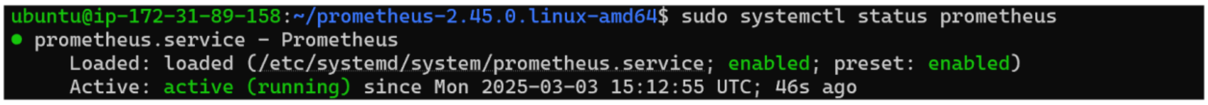


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64$ sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl start prometheus
```

Step 12:

Verify Prometheus is running

sudo systemctl status Prometheus

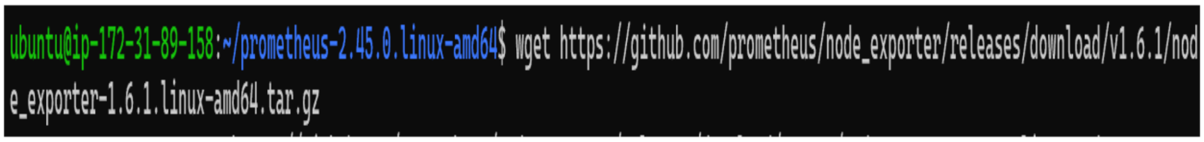


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64$ sudo systemctl status prometheus
● prometheus.service - Prometheus
     Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-03 15:12:55 UTC; 46s ago
```

Step 13:

Download Node Exporter

https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz

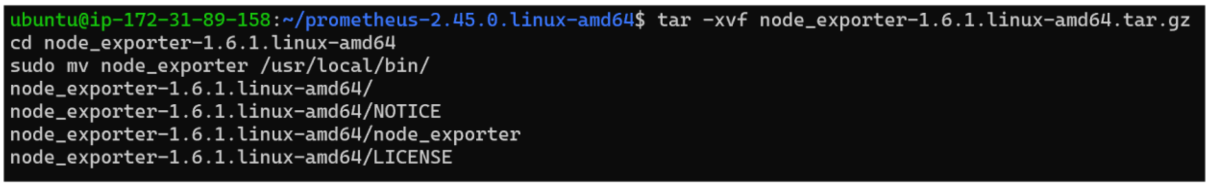


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64$ wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/nod
e_exporter-1.6.1.linux-amd64.tar.gz
```

Step 14:

Extract and move files

tar -xvf node_exporter-1.6.1.linux-amd64.tar.gz
cd node_exporter-1.6.1.linux-amd64
sudo mv node_exporter /usr/local/bin/

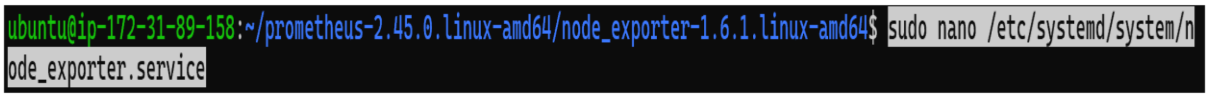


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64$ tar -xvf node_exporter-1.6.1.linux-amd64.tar.gz
cd node_exporter-1.6.1.linux-amd64
sudo mv node_exporter /usr/local/bin/
node_exporter-1.6.1.linux-amd64/
node_exporter-1.6.1.linux-amd64/NOTICE
node_exporter-1.6.1.linux-amd64/node_exporter
node_exporter-1.6.1.linux-amd64/LICENSE
```

Step 15:

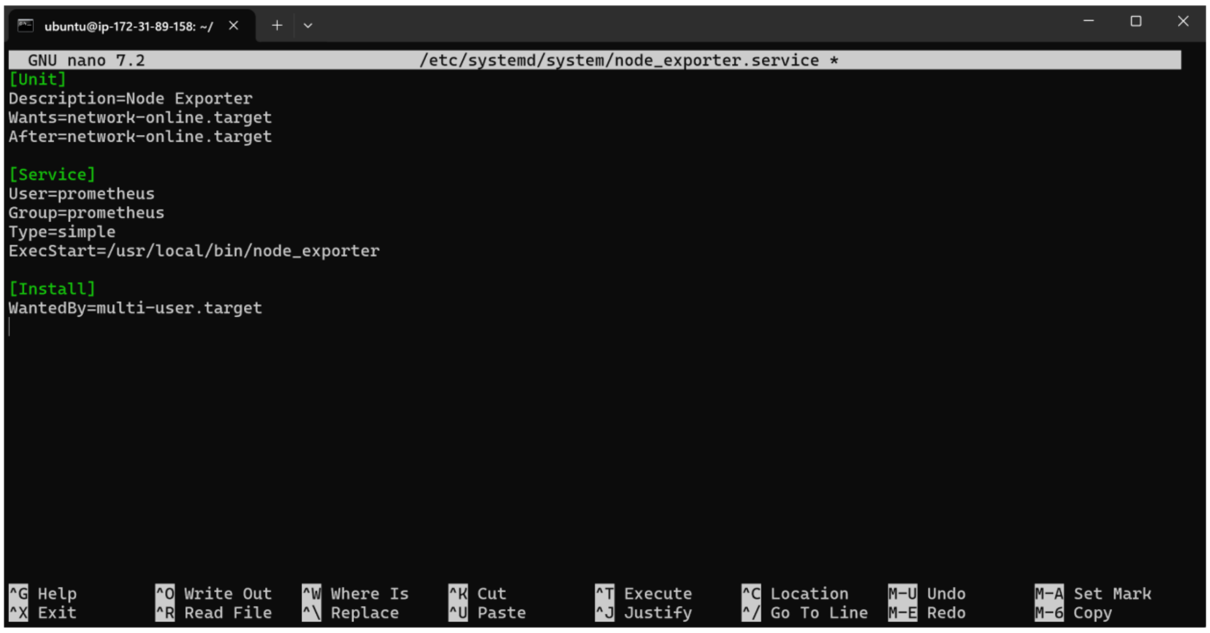
1. Create a systemd service file

sudo nano /etc/systemd/system/node_exporter.service



```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo nano /etc/systemd/system/node_exporter.service
```

2. Paste this content inside the file:



ubuntu@ip-172-31-89-158: ~/
GNU nano 7.2
/etc/systemd/system/node_exporter.service *

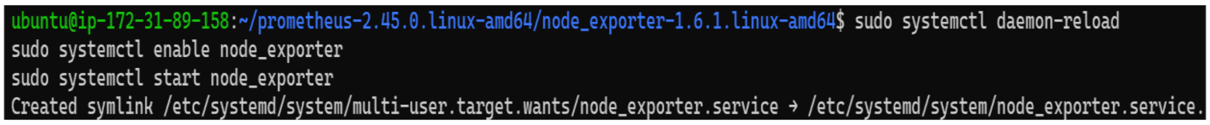
```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

Press **CTRL + O** and then **Enter**. Then Press **Ctrl+X** .

Step 16:

Reload systemd and start Node Exporter

sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter

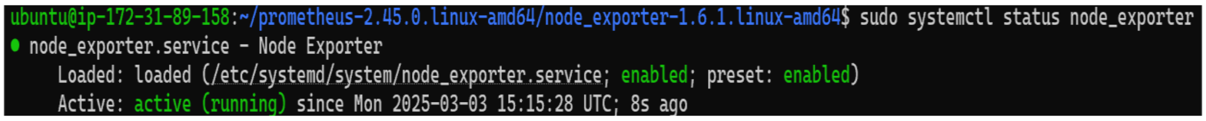


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/node_exporter.service.
```

Step 17:

Verify Node Exporter is running

sudo systemctl status node_exporter

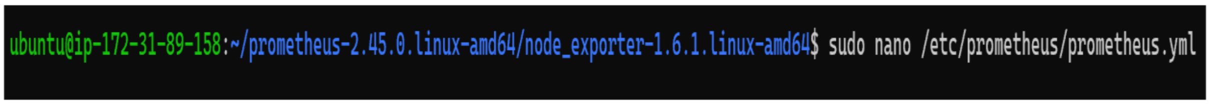


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-03 15:15:28 UTC; 8s ago
```

Step 18:

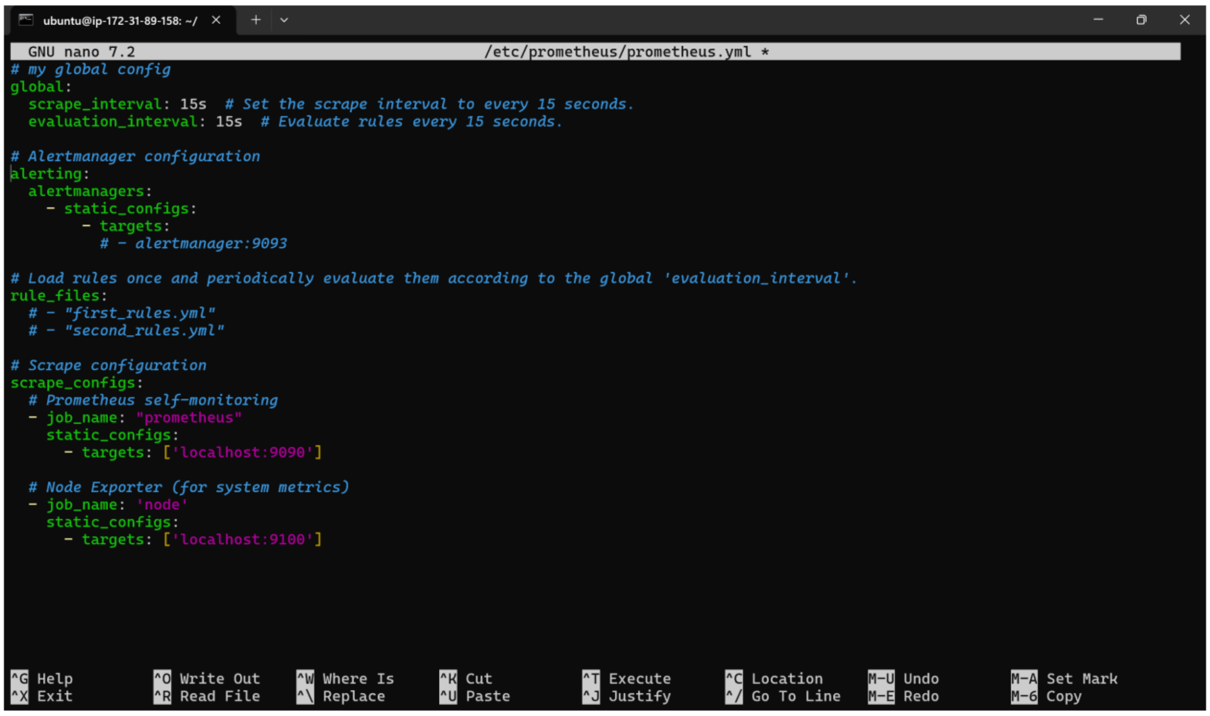
1. Edit Prometheus config

sudo nano /etc/prometheus/prometheus.yml



```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo nano /etc/prometheus/prometheus.yml
```

2. Add these lines at the end of the **scrape_configs** section



```
  ubuntu@ip-172-31-89-158: ~/
  GNU nano 7.2                              /etc/prometheus/prometheus.yml *
# my global config
global:
  scrape_interval: 15s  # Set the scrape interval to every 15 seconds.
  evaluation_interval: 15s  # Evaluate rules every 15 seconds.

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# Scrape configuration
scrape_configs:
  # Prometheus self-monitoring
  - job_name: "prometheus"
    static_configs:
      - targets: ['localhost:9090']

  # Node Exporter (for system metrics)
  - job_name: 'node'
    static_configs:
      - targets: ['localhost:9100']

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Press **CTRL + O** and then **Enter**. Then Press **Ctrl+X** .

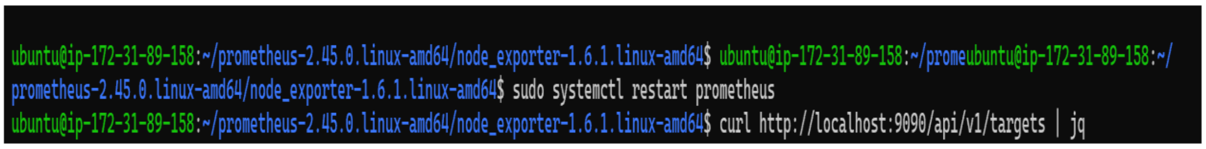
Step 19:

Restart Prometheus

sudo systemctl restart prometheus

Verify Prometheus can see Node Exporter

curl http://localhost:9090/api/v1/targets | jq



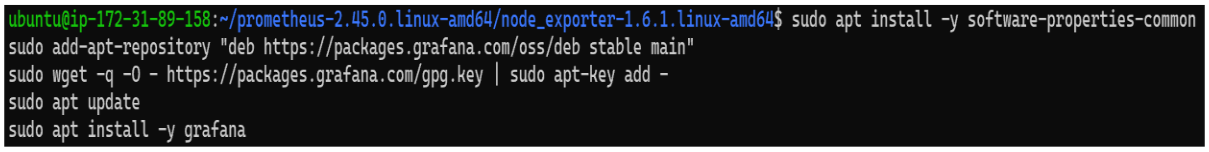
```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ ubuntu@ip-172-31-89-158:~/promeubuntu@ip-172-31-89-158:~/
prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo systemctl restart prometheus
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ curl http://localhost:9090/api/v1/targets | jq
```

Expected Output: health: "up"

Step 20:

Download and Install Grafana

sudo apt install -y software-properties-common
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
sudo wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt update
sudo apt install -y grafana

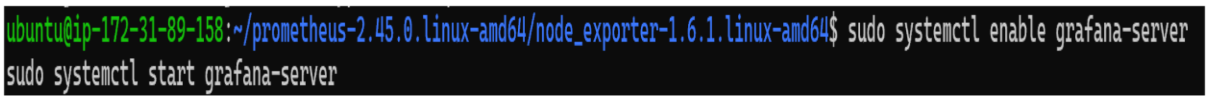


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo apt install -y software-properties-common
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
sudo wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt update
sudo apt install -y grafana
```

Step 21:

Start and enable Grafana

sudo systemctl enable grafana-server
sudo systemctl start grafana-server

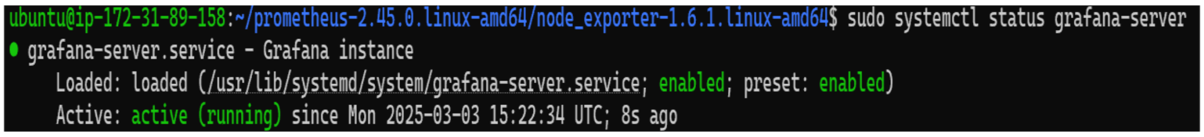


```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Step 22:

Check if Grafana is running

sudo systemctl status grafana-server



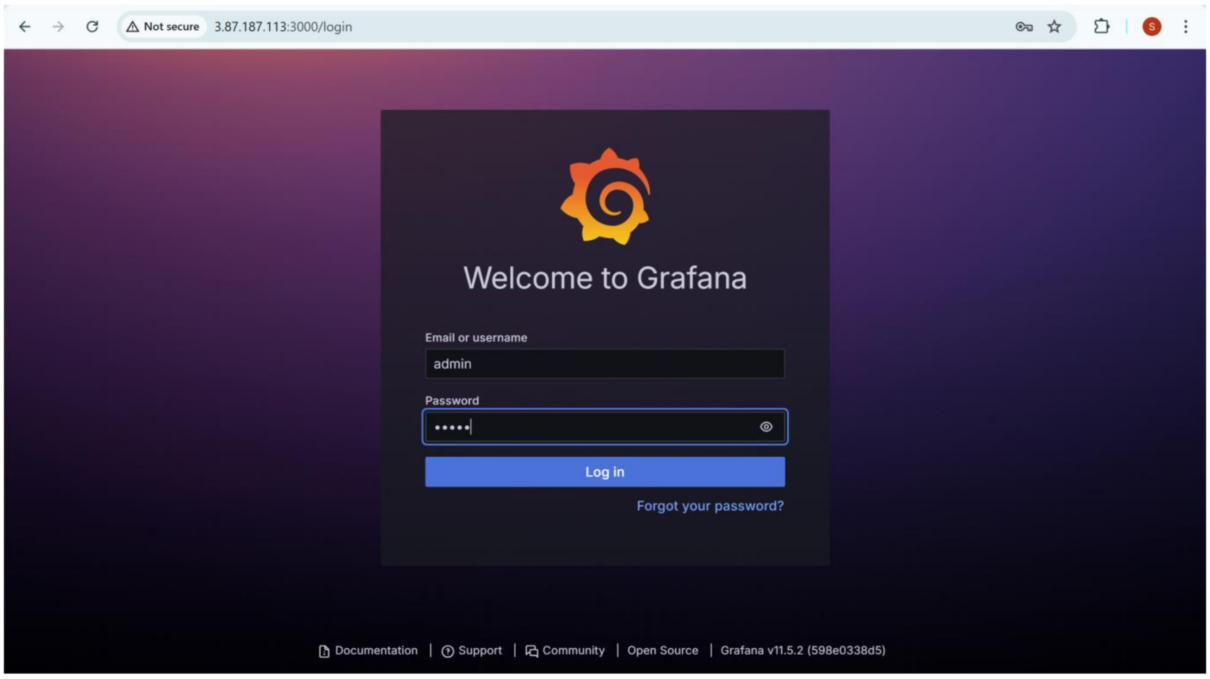
```
ubuntu@ip-172-31-89-158:~/prometheus-2.45.0.linux-amd64/node_exporter-1.6.1.linux-amd64$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-03 15:22:34 UTC; 8s ago
```

Step 23:

Go to:

http://<your-ec2-public-ip>:3000

- Username: admin
- Password: admin (default, change it later)

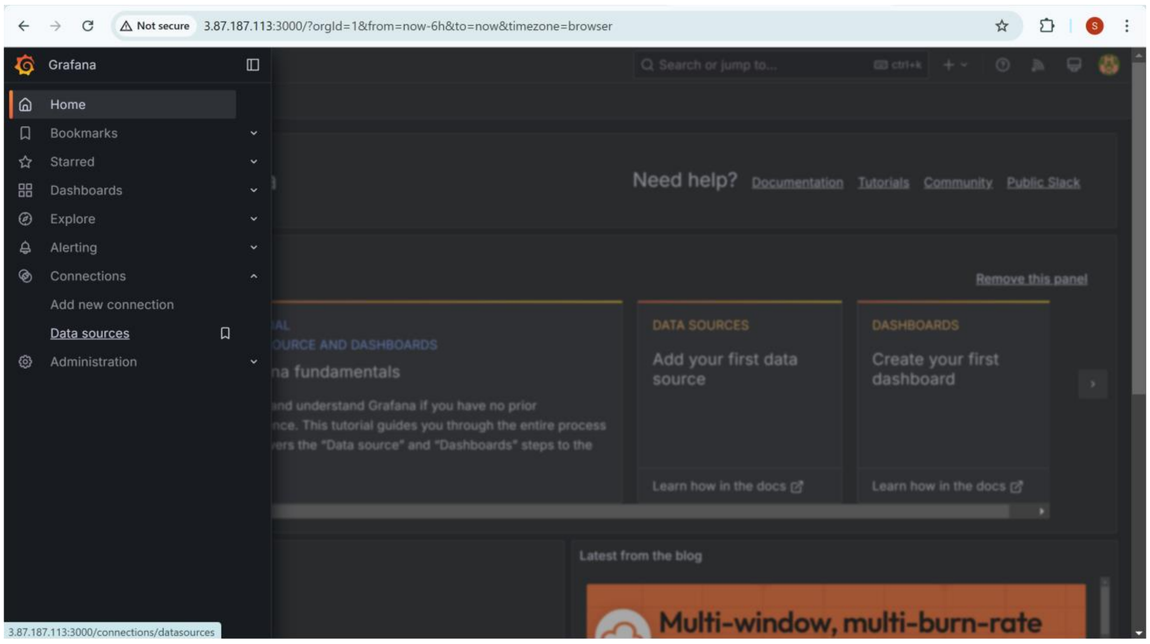


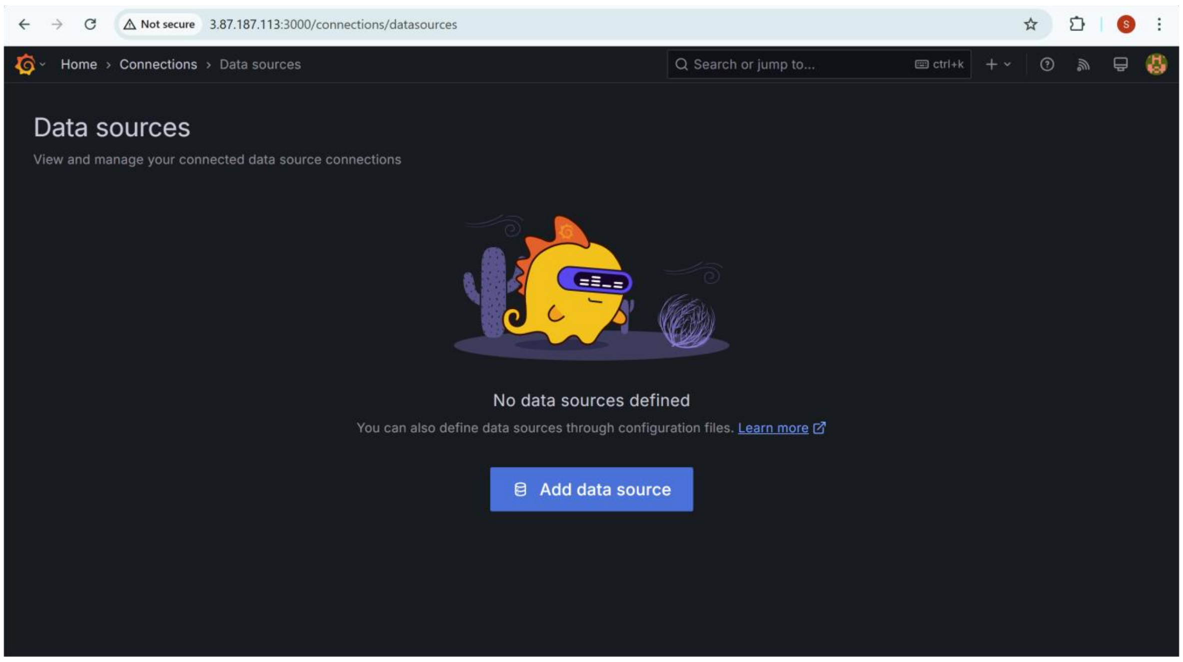
Step 24:

1. Click **Connections → Data Sources → Add new data source**
2. Select **Prometheus**
3. Change URL to:

http://localhost:9090

4. Click **Save & Test**



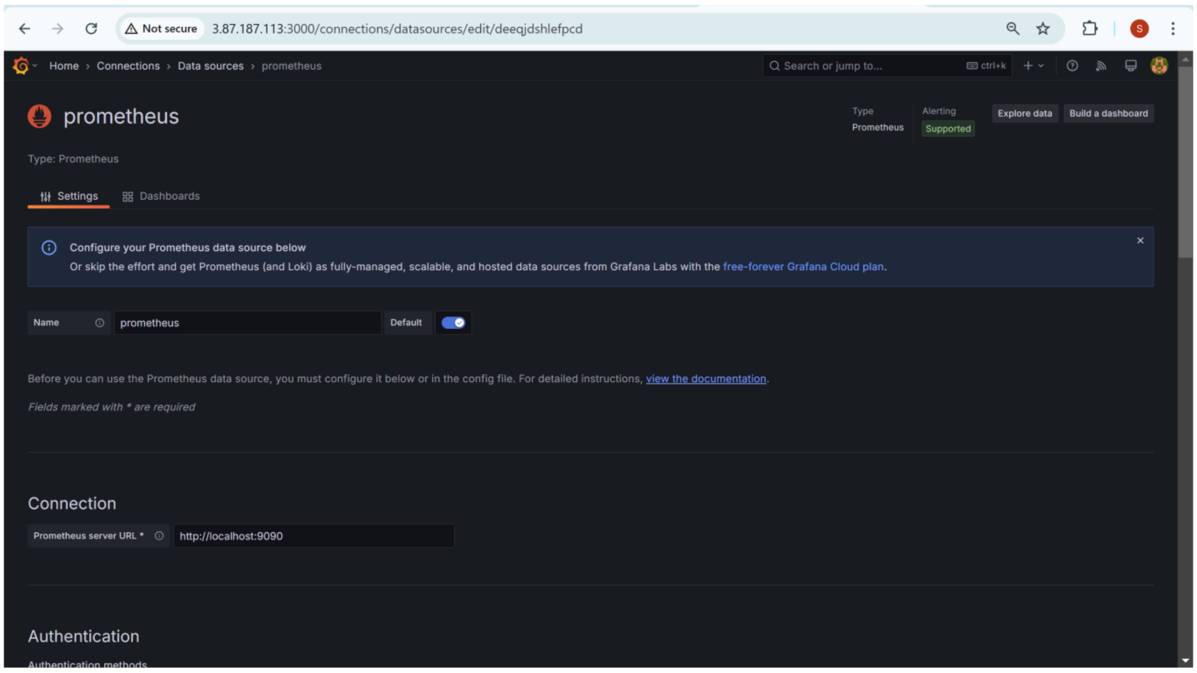

Not secure
3.87.187.113:3000/connections/datasources
Home › Connections › Data sources
Search or jump to...
ctrl+k

Data sources

View and manage your connected data source connections

No data sources defined

You can also define data sources through configuration files. Learn more


Not secure
3.87.187.113:3000/connections/datasources/edit/deeqjdshlefpcd
Home › Connections › Data sources › prometheus
Search or jump to...
ctrl+k

prometheus

Type: Prometheus

Type Prometheus | Alerting Supported | Explore data | Build a dashboard

Settings | Dashboards

Configure your Prometheus data source below
Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the free-forever Grafana Cloud plan.

Name: prometheus | Default: on

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

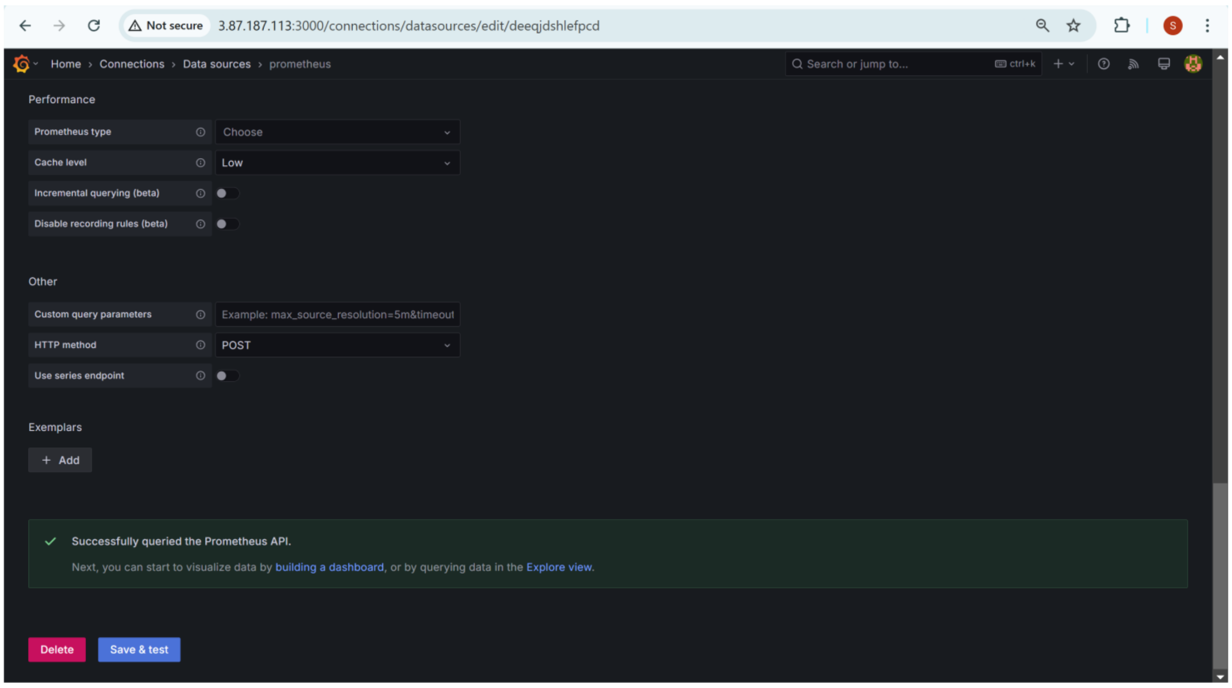
*Fields marked with * are required*

Connection

Prometheus server URL *: http://localhost:9090

Authentication

Authentication methods


Not secure
3.87.187.113:3000/connections/datasources/edit/deeqjdshlefpcd
Home › Connections › Data sources › prometheus
Search or jump to...
ctrl+k

Performance

Prometheus type	Choose
Cache level	Low
Incremental querying (beta)	off
Disable recording rules (beta)	off

Other

Custom query parameters	Example: max_source_resolution=5m&timeout
HTTP method	POST
Use series endpoint	off

Exemplars

+ Add

✓ Successfully queried the Prometheus API.
Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.

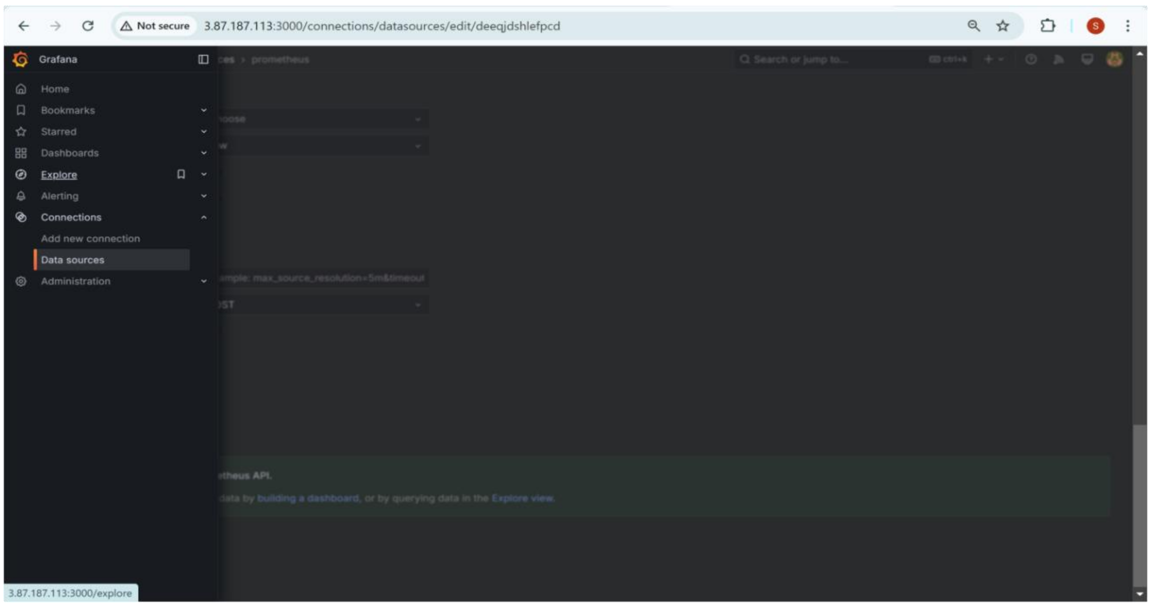
Delete | Save & test

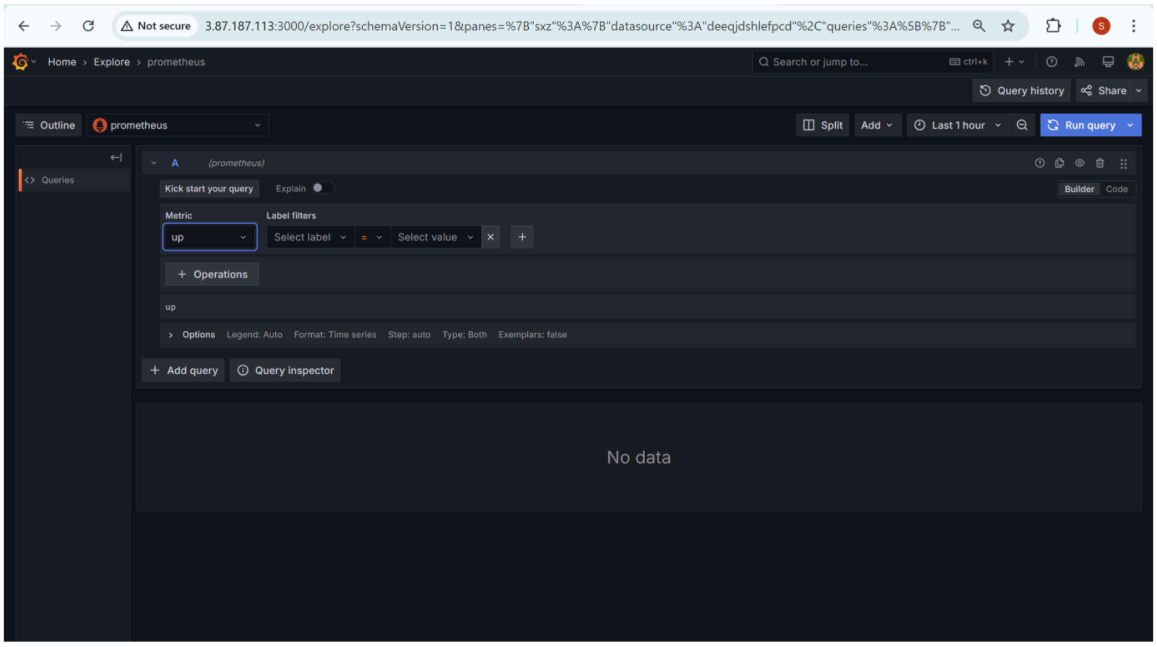
Step 25:

1. Click **Explore (Compass Icon)**.
2. Select **Prometheus** as the data source.
3. In the query box, type:

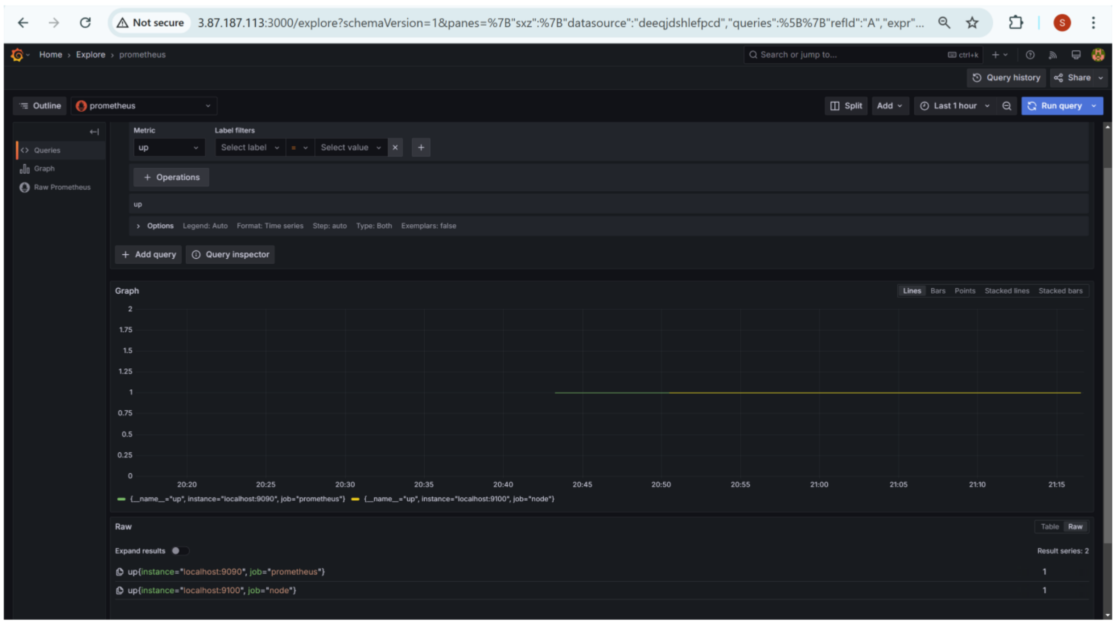
   up

4. Click **Run Query**.





Not secure 3.87.187.113:3000/explore?schemaVersion=1&panes=%7B"sxz"%3A%7B"datasource"%3A"deeqjdshlefpcd"%2C"queries"%3A%5B%7B"...
Home › Explore › prometheus
Search or jump to...
Query history
Share
Outline
prometheus
Split
Add
Last 1 hour
Run query
Queries
A (prometheus)
Kick start your query
Explain
Builder
Code
Metric
up
Label filters
Select label
=
Select value
Operations
up
Options Legend: Auto Format: Time series Step: auto Type: Both Exemplars: false
Add query
Query inspector
No data



Metric: up | Label filters: Select label = Select value

```
up
```

Options   Legend: Auto   Format: Time series   Step: auto   Type: Both   Exemplars: false

{__name__="up", instance="localhost:9090", job="prometheus"}   {__name__="up", instance="localhost:9100", job="node"}

Result series: 2

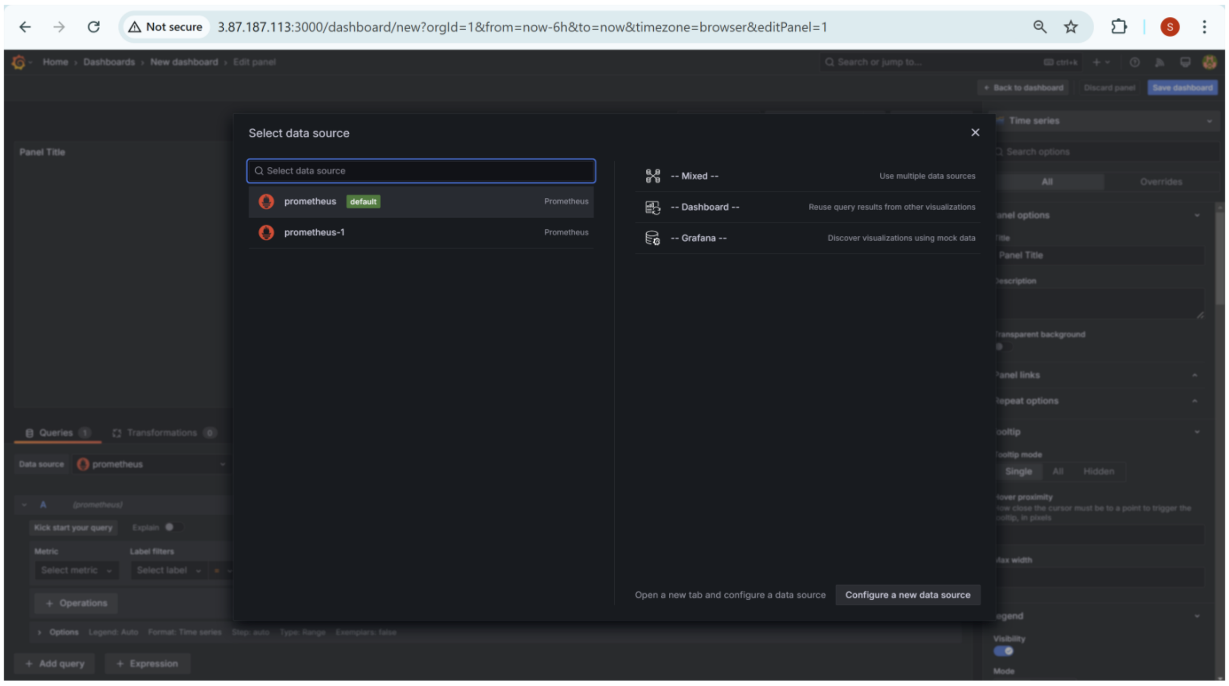
```
up{instance="localhost:9090", job="prometheus"}    1
up{instance="localhost:9100", job="node"}          1
```

Step 26:

1. **Click on Dashboards → New Dashboard**
2. **Click on "Add Visualization"**
3. Select **Prometheus** as the Data Source
4. Enter the **query**:

   node_cpu_seconds_total

5. Click **Run Query**

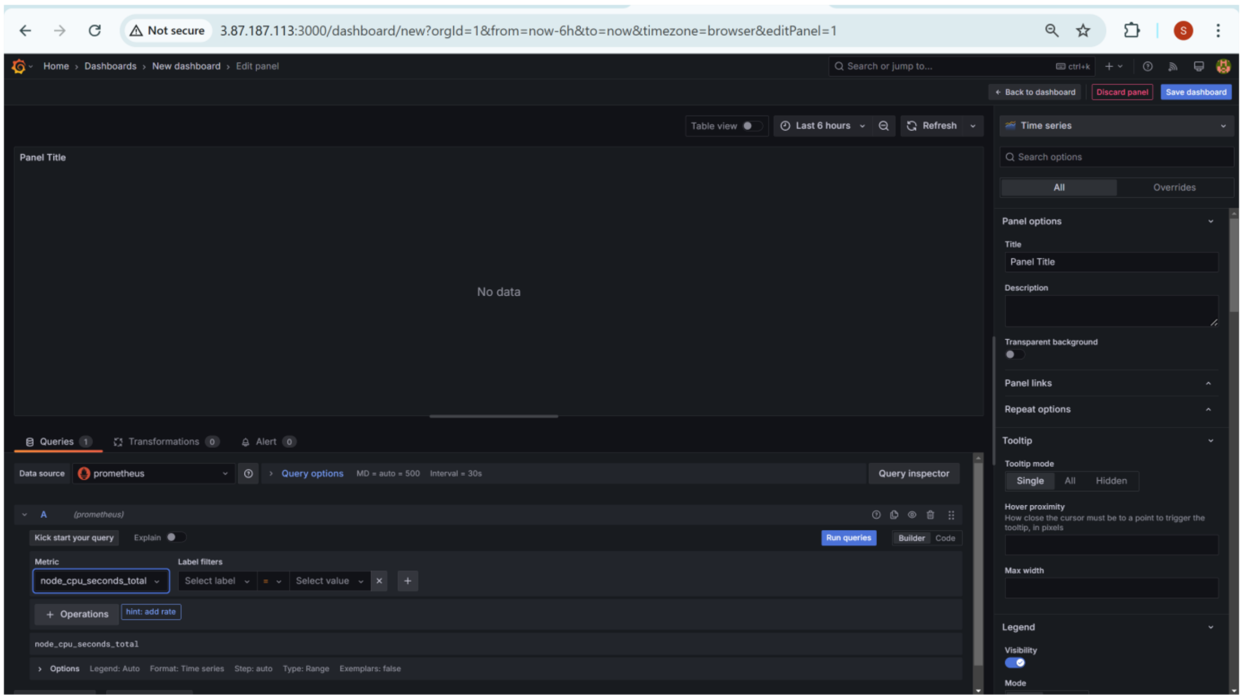

3.87.187.113:3000/dashboard/new?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1

Select data source

prometheus	default	Prometheus
prometheus-1		Prometheus

-- Mixed --	Use multiple data sources
-- Dashboard --	Reuse query results from other visualizations
-- Grafana --	Discover visualizations using mock data

Open a new tab and configure a data source


3.87.187.113:3000/dashboard/new?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1

Data source: prometheus

Query options MD = auto = 500 Interval = 30s

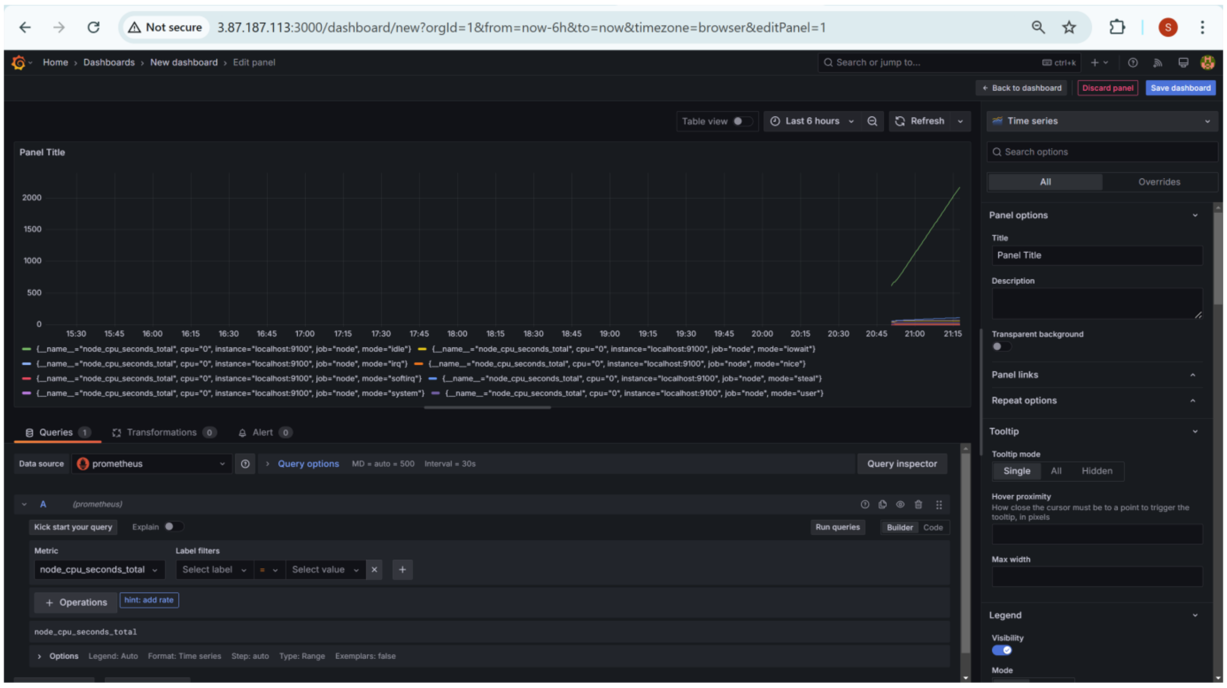
A (prometheus)

Metric: node_cpu_seconds_total

```
node_cpu_seconds_total
```

Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false

Panel options
Title: Panel Title
Tooltip mode: [✓ Single] All Hidden
Hover proximity: How close the cursor must be to a point to trigger the tooltip, in pixels



3.87.187.113:3000/dashboard/new?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1

Home › Dashboards › New dashboard › Edit panel

Panel Title

```
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="idle"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="iowait"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="irq"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="nice"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="softirq"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="steal"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="system"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="user"}
```

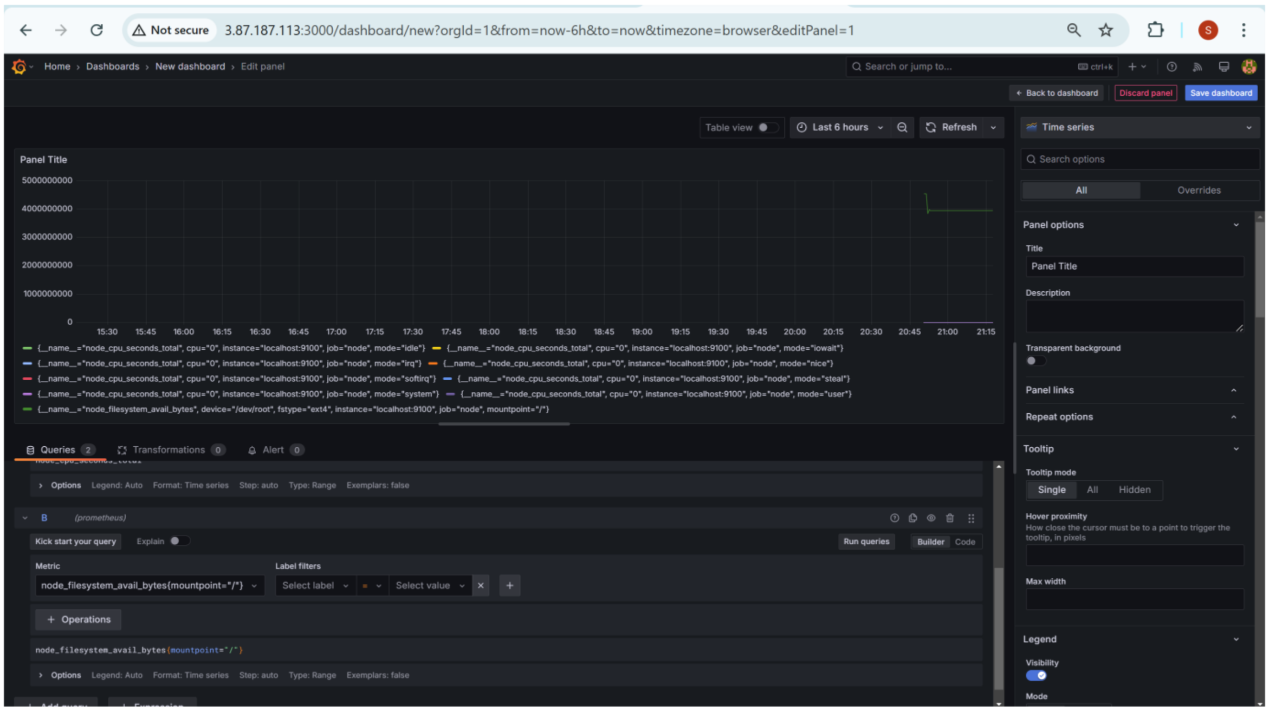
Data source: prometheus — Query options: MD = auto = 500, Interval = 30s

A (prometheus)

Metric: node_cpu_seconds_total

```
node_cpu_seconds_total
```

Options: Legend: Auto, Format: Time series, Step: auto, Type: Range, Exemplars: false



3.87.187.113:3000/dashboard/new?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1

Home › Dashboards › New dashboard › Edit panel

Panel Title

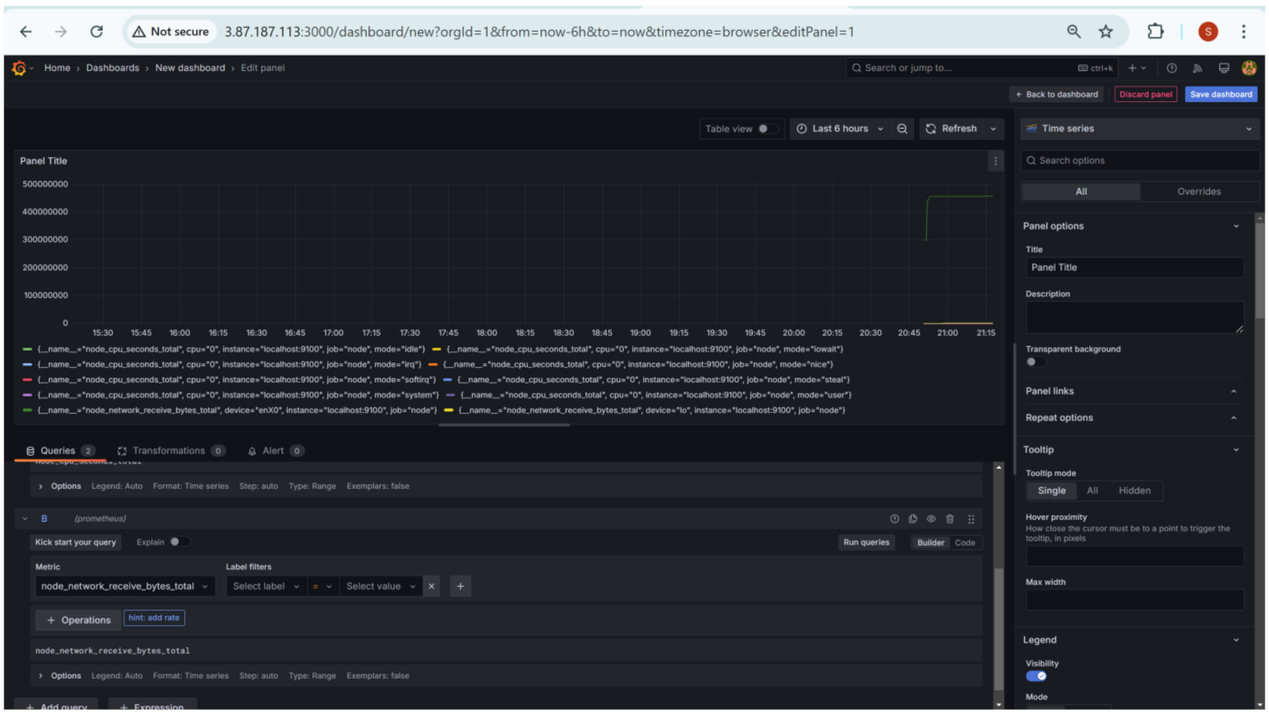
```
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="idle"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="iowait"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="irq"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="nice"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="softirq"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="steal"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="system"}
{__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="user"}
{__name__="node_filesystem_avail_bytes", device="/dev/root", fstype="ext4", instance="localhost:9100", job="node", mountpoint="/"}
```

B (prometheus)

Metric: node_filesystem_avail_bytes{mountpoint="/"}

```
node_filesystem_avail_bytes{mountpoint="/"}
```

Options: Legend: Auto, Format: Time series, Step: auto, Type: Range, Exemplars: false



3.87.187.113:3000/dashboard/new?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1

Panel Title

```
500000000
400000000
300000000
200000000
100000000
0
  15:30 15:45 16:00 16:15 16:30 16:45 17:00 17:15 17:30 17:45 18:00 18:15 18:30 18:45 19:00 19:15 19:30 19:45 20:00 20:15 20:30 20:45 21:00 21:15
```

- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="idle"}
- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="iowait"}
- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="irq"}
- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="nice"}
- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="softirq"}
- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="steal"}
- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="system"}
- {__name__="node_cpu_seconds_total", cpu="0", instance="localhost:9100", job="node", mode="user"}
- {__name__="node_network_receive_bytes_total", device="enX0", instance="localhost:9100", job="node"}
- {__name__="node_network_receive_bytes_total", device="lo", instance="localhost:9100", job="node"}

Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false

B (prometheus)

Kick start your query Explain — Run queries — Builder Code

Metric: node_network_receive_bytes_total
Label filters: Select label = Select value

+ Operations hint: add rate

```
node_network_receive_bytes_total
```

Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false

Panel options
Title: Panel Title
Description:
Transparent background
Panel links
Repeat options

Tooltip
Tooltip mode: [✓ Single] All Hidden
Hover proximity
How close the cursor must be to a point to trigger the tooltip, in pixels
Max width

Legend
Visibility ✓
Mode

Outcomes

By completing this **Prometheus and Grafana for System Monitoring on AWS EC2** PoC, you will:

1. **Understand Cloud-Based Monitoring** – Gain hands-on experience in setting up Prometheus and Grafana for real-time system monitoring on AWS EC2.

2. **Collect and Analyze System Metrics** – Learn how to scrape and visualize CPU usage, memory consumption, disk performance, and network traffic data using Prometheus and Node Exporter.

3. **Configure PromQL Queries for Analysis** – Develop proficiency in writing PromQL queries to analyze system performance, detect anomalies, and optimize resource utilization.

4. **Set Up Secure Monitoring Infrastructure** – Implement a reliable monitoring stack with Prometheus as the data source and Grafana as the visualization tool.

5. **Enable Real-Time Visualization** – Utilize Grafana dashboards to monitor and visualize key system metrics, improving observability and decision-making.

6. **Enhance Troubleshooting and Alerting** – Gain insights into system behavior and set up alerting mechanisms to proactively detect and resolve issues before they impact performance.

7. **Improve Cloud Resource Management** – Optimize cloud resource usage by continuously monitoring system performance, ensuring efficient workload distribution and cost management.

8. **Apply DevOps Best Practices** – Learn how monitoring fits into DevOps workflows, enhancing automation, incident response, and overall system reliability.