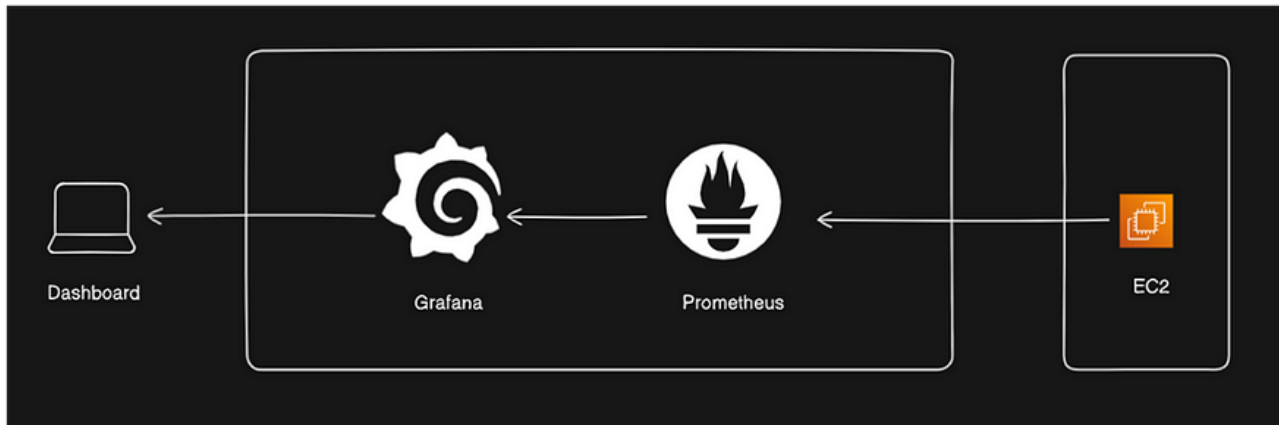# GRAFANA INSTALL ON UBUNTU EC2

Introduction

In today's dynamic IT landscape, effective infrastructure monitoring is crucial for ensuring optimal performance and reliability. This project aims to explore the power of Grafana and Prometheus in monitoring CPU and memory utilization on EC2 instances. By leveraging these tools, we can gain valuable insights into our infrastructure's health and performance, enabling us to make data-driven decisions and proactively address any issues.

Aim

The aim of this project is to set up a robust monitoring system using Grafana and Prometheus to track CPU and memory utilization on EC2 instances. By monitoring these key metrics in real-time, we can identify performance bottlenecks, optimize resource allocation, and ensure the smooth operation of our infrastructure.

Objective

- Configure Prometheus to scrape metrics from EC2 instances

- Set up Grafana to visualize CPU and memory utilization metrics

- Create dashboards in Grafana to monitor and analyze infrastructure performance

- Establish alerting mechanisms to notify stakeholders of any anomalies or issues

- Optimize resource allocation based on monitoring data to improve overall infrastructure efficiency

Process

1. Setting up EC2 instance
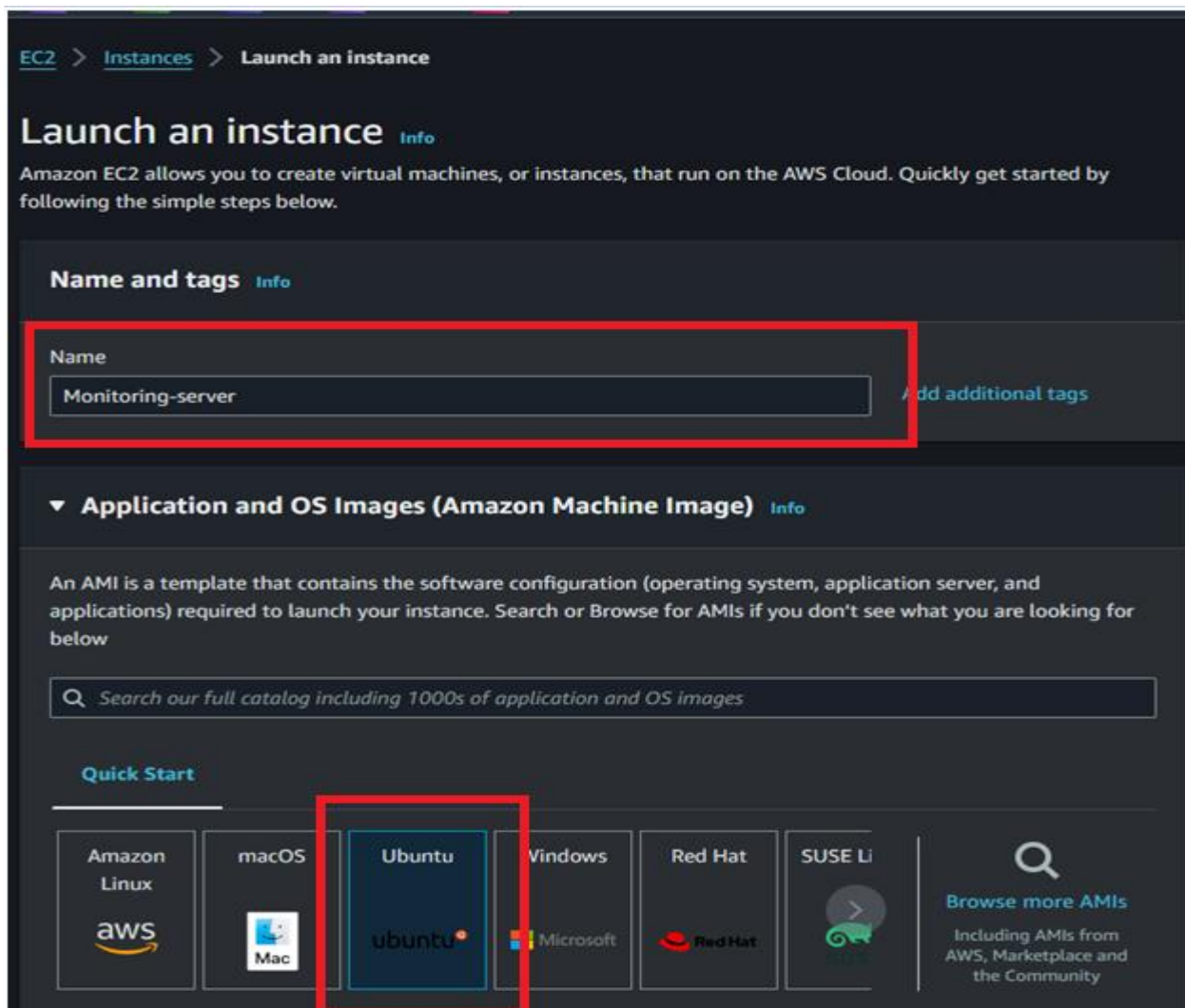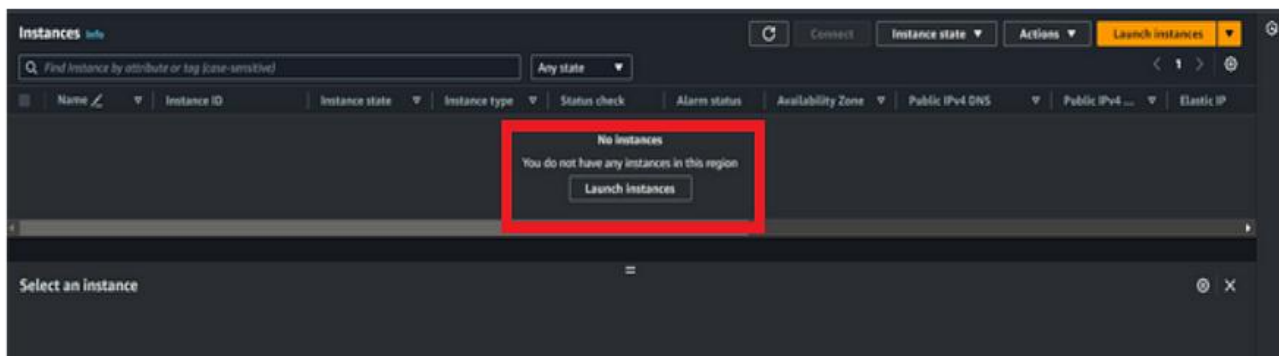
Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
ami-03f4878755434977f (64-bit (x86)) / ami-077885f59ecb77b84 (64-bit (Arm))
Virtualization: hvm   ENA enabled: true   Root device type: ebs

Free tier eligible

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-12-07

Architecture

64-bit (x86)

AMI ID

ami-03f4878755434977f

Verified provider

▼ **Instance type** Info | Get advice

Instance type

t2.micro
Family: t2   1 vCPU   1 GiB Memory   Current generation: true
On-Demand Linux base pricing: 0.0124 USD per Hour
On-Demand Windows base pricing: 0.017 USD per Hour
On-Demand RHEL base pricing: 0.0724 USD per Hour
On-Demand SUSE base pricing: 0.0124 USD per Hour

Free tier eligible

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ **Key pair (login)** Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Mumbai_new_AD

↻ Create new key pair

▼ **Network settings** Info

Edit

Network | Info

vpc-00c3ebfca0fa4dfc5

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

Enable

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

◉ Create security group   ○ Select existing security group

We'll create a new security group called **launch-wizard-4** with the following rules:

☑ **Allow SSH traffic from**
Helps you connect to your instance

Anywhere
0.0.0.0/0

☐ **Allow HTTPS traffic from the internet**
To set up an endpoint, for example when creating a web server

☐ **Allow HTTP traffic from the internet**
To set up an endpoint, for example when creating a web server
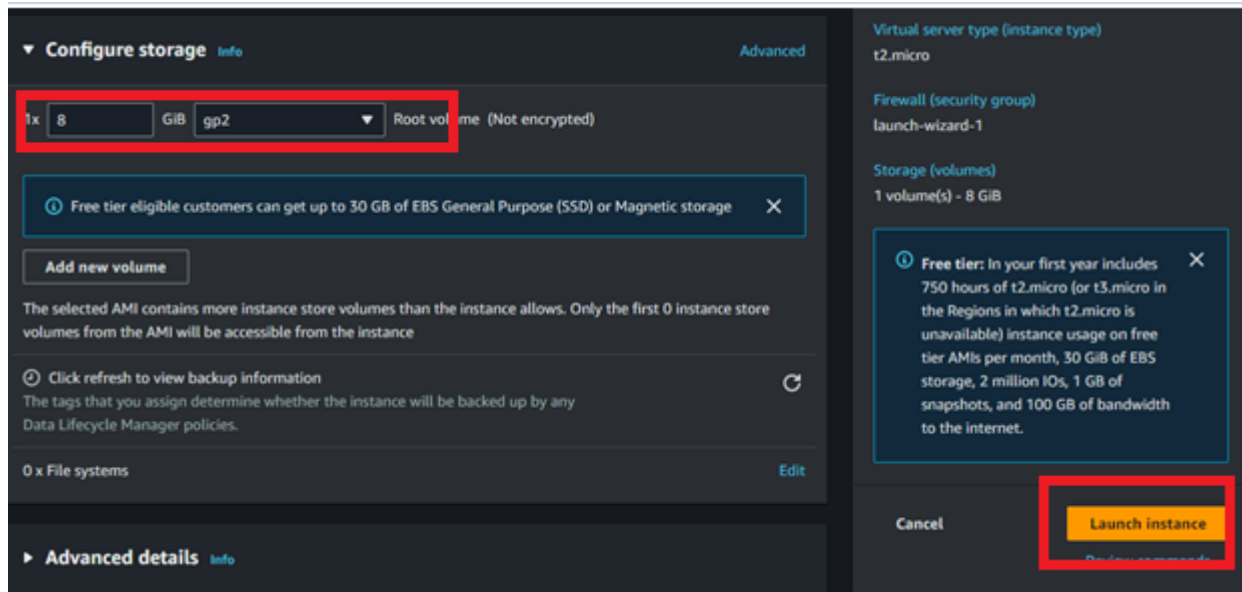
2. Configuring Grafana

- Install and configure Grafana on the same server as Prometheus

- Connect Grafana to Prometheus as a data source

About Grafana

Grafana is an open-source analytics and visualization platform that allows users to query, visualize, alert on, and understand their metrics no matter where they are stored. It enables users to create, explore, and share dashboards with team members to monitor the metrics and understand trends in real-time. Grafana supports a wide range of data sources, including Prometheus, InfluxDB, Elasticsearch, Graphite, and more.

About Prometheus

Prometheus, on the other hand, is an open-source monitoring and alerting toolkit designed for reliability, scalability, and automation of metrics collection, storage, and querying. It collects metrics from monitored targets by scraping HTTP endpoints, and stores the data in a time-series database where it can be queried, analyzed, and visualized. Prometheus also includes a powerful query language (PromQL) for slicing and dicing the collected data to gain insights into system performance and behavior. It is often used in conjunction with Grafana for creating dashboards and visualizing the collected metrics.

Install Grafana

sudo apt-get update

sudo apt-get install -y software-properties-common curl

Add the GPG key securely

curl -fsSL https://packages.grafana.com/gpg.key | sudo tee /usr/share/keyrings/grafana.gpg > /dev/null

Add Grafana APT repo with signed-by option

echo "deb [signed-by=/usr/share/keyrings/grafana.gpg] https://packages.grafana.com/oss/deb stable main" | \

sudo tee /etc/apt/sources.list.d/grafana.list > /dev/null

Update and install Grafana

sudo apt-get update

sudo apt-get install -y grafana

```
root@ip-172-31-39-112:~/Grafana# chmod +x Install-Grafana.sh
root@ip-172-31-39-112:~/Grafana# ls
Install-Grafana.sh
root@ip-172-31-39-112:~/Grafana# |
```

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

```
root@ip-172-31-39-112:~/Grafana# systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-01-30 08:31:14 UTC; 31s ago
       Docs: http://docs.grafana.org
   Main PID: 3343 (grafana)
      Tasks: 10 (limit: 1121)
     Memory: 76.7M
        CPU: 2.245s
     CGroup: /system.slice/grafana-server.service
             └─3343 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths.logs=/var/log/grafana cfg:default

Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=ticker t=2024-01-30T08:31:20.812587943Z level=info msg=starting first_tick=2024-01-30T08:31:30Z
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=local.finder t=2024-01-30T08:31:20.813800066Z level=warn msg="Skipping finding plugins as directory does not exist" path=/var/lib/grafana/plugins
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=grafanaStorageLogger t=2024-01-30T08:31:20.818389533Z level=info msg="Storage starting"
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=grafana-apiserver t=2024-01-30T08:31:20.833920671Z level=info msg="Authentication is disabled"
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=ngalert.multiorg.alertmanager t=2024-01-30T08:31:20.839456387Z level=info msg="Starting MultiOrg Alertmanager"
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=sqlstore.transactions t=2024-01-30T08:31:20.85042566Z level=info msg="Database locked, sleeping then retrying" error="database is locked" retry=0
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=grafana-apiserver t=2024-01-30T08:31:20.858911258Z level=info msg="Adding GroupVersion playlist.grafana.app v0alpha1 to ResourceManager"
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=http.server t=2024-01-30T08:31:20.878820547Z level=info msg="HTTP Server Listen" address=[::]:3000 protocol=http subUrl= socket=
Jan 30 08:31:20 ip-172-31-39-112 grafana[3343]: logger=grafana.update.checker t=2024-01-30T08:31:20.956913532Z level=info msg="Update check succeeded" duration=141.872372ms
Jan 30 08:31:21 ip-172-31-39-112 grafana[3343]: logger=plugins.update.checker t=2024-01-30T08:31:21.219350759Z level=info msg="Update check succeeded" duration=401.875191ms
lines 1-21/21 (END)
```

Install Prometheus

```
root@ip-172-31-39-112:~/Grafana# sudo apt-get install prometheus
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fonts-glyphicons-halflings javascript-common libio-pty-perl libipc-run-perl libjs-bootstrap
```

sudo apt-get install Prometheus

sudo systemctl start Prometheus
sudo systemctl enable prometheus

sudo apt update

sudo apt install -y stress

Multi cloud with devops by veera nareshit

```
root@ip-172-31-39-112:~/Grafana# sudo systemctl status prometheus
● prometheus.service - Monitoring system and time series database
     Loaded: loaded (/lib/systemd/system/prometheus.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-01-30 08:32:52 UTC; 22s ago
       Docs: https://prometheus.io/docs/introduction/overview/
             man:prometheus(1)
   Main PID: 4484 (prometheus)
      Tasks: 7 (limit: 1121)
     Memory: 18.5M
        CPU: 156ms
     CGroup: /system.slice/prometheus.service
             └─4484 /usr/bin/prometheus
```

Accessing Grafana:

We can access Grafana via browser by using our instance IP **:3000**

Username is admin

Password is admin

After that change the password
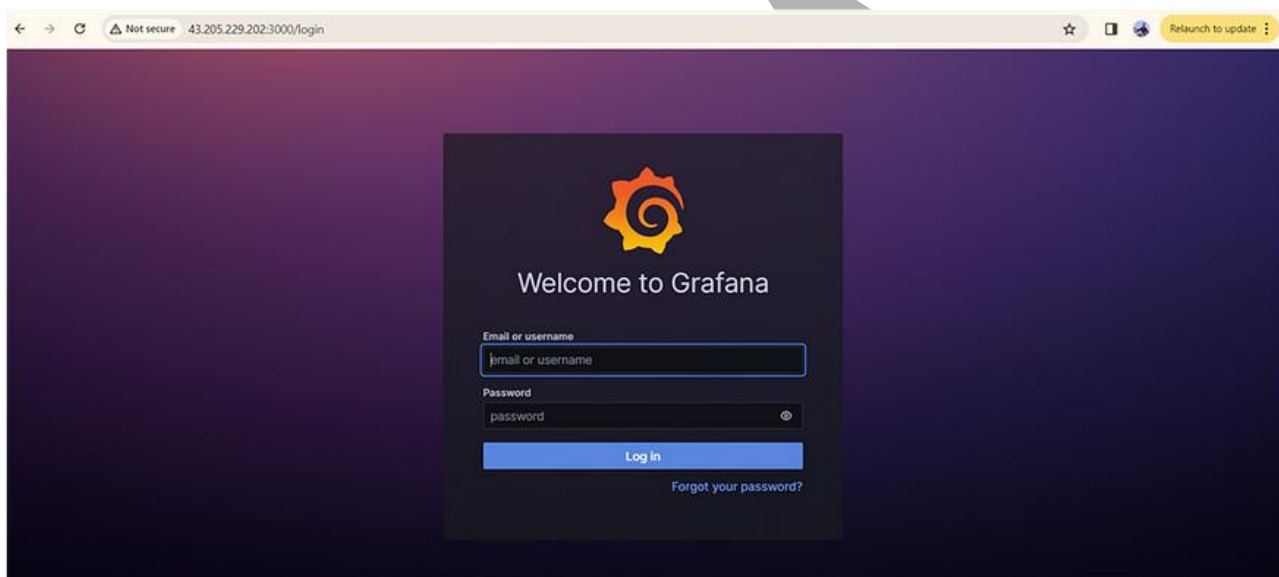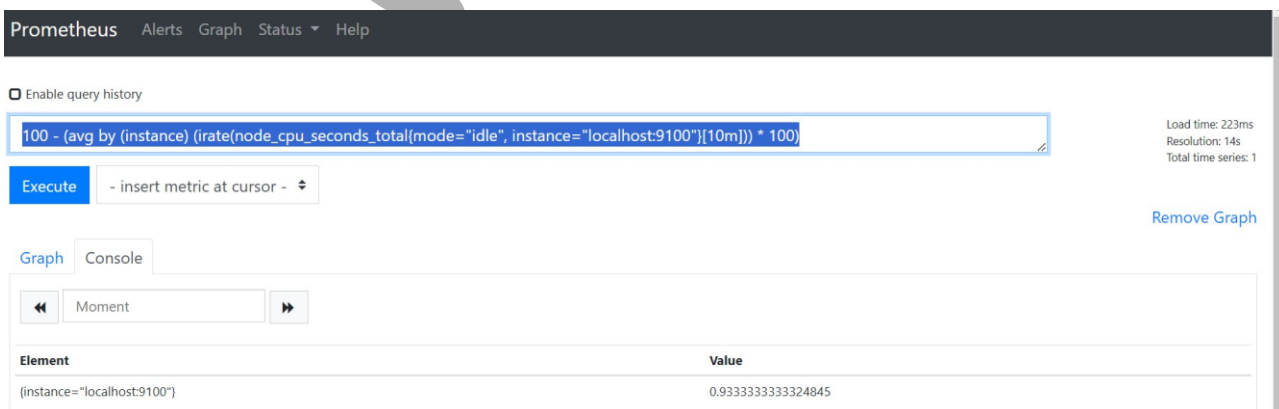

Accessing Prometheus:

We can access Grafana via browser by using our instance IP **:9090**

Run below query to check the load

100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle", instance="localhost:9100"}[10m])) * 100)

Explanation:

Multi cloud with devops by veera nareshit

Multi cloud with devops by veera nareshit

node_cpu_seconds_total{mode="idle"}:

This metric tracks the total number of seconds the CPU has spent in an idle state, which is available for all servers (instances) that are being monitored by Prometheus. The {mode="idle"} part filters for idle time only.

irate(node_cpu_seconds_total{mode="idle"}[5m]):

The irate() function calculates the rate of change (per second) for the node_cpu_seconds_total metric over the last 5 minutes ([5m]). It returns how much idle time occurred during the last 5-minute window for each instance.

avg by (instance):

This function averages the rate of idle CPU time over all CPU cores for each instance. So, it aggregates the CPU idle rate per instance, providing a single value per instance.

* 100 and 100 - (...):

The multiplication by 100 converts the idle rate into a percentage.

Subtracting this from 100 calculates the CPU usage percentage. This shows how much time the CPU has been actively used, rather than idle, for each instance.

3. Creating Dashboards

- Design custom dashboards in Grafana to visualize CPU and memory utilization metrics
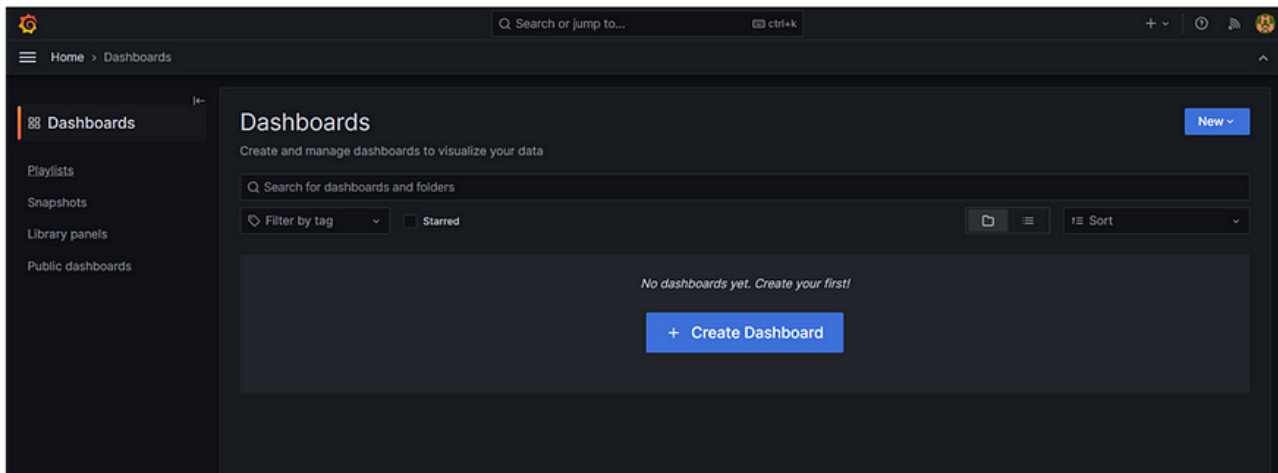- Customize graphs and panels to display relevant information in real-time
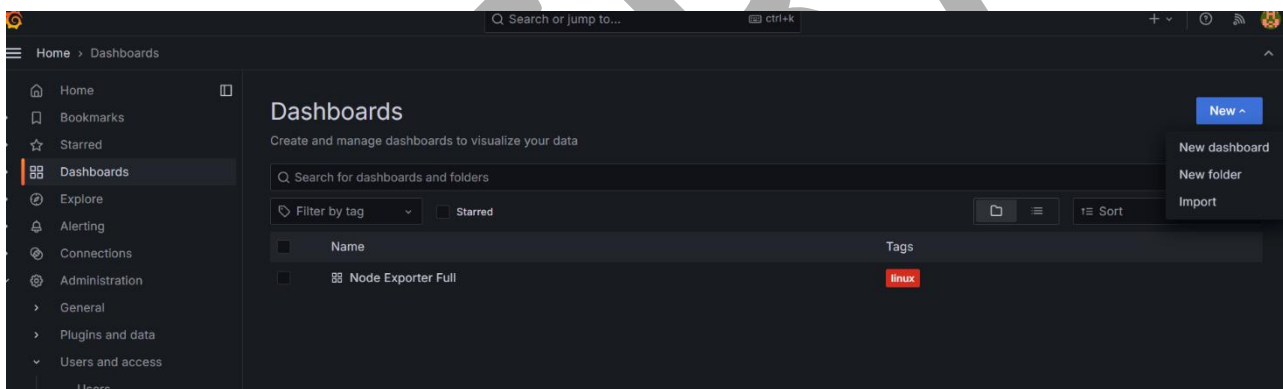

Add Prometheus

Multi cloud with devops by veera nareshit
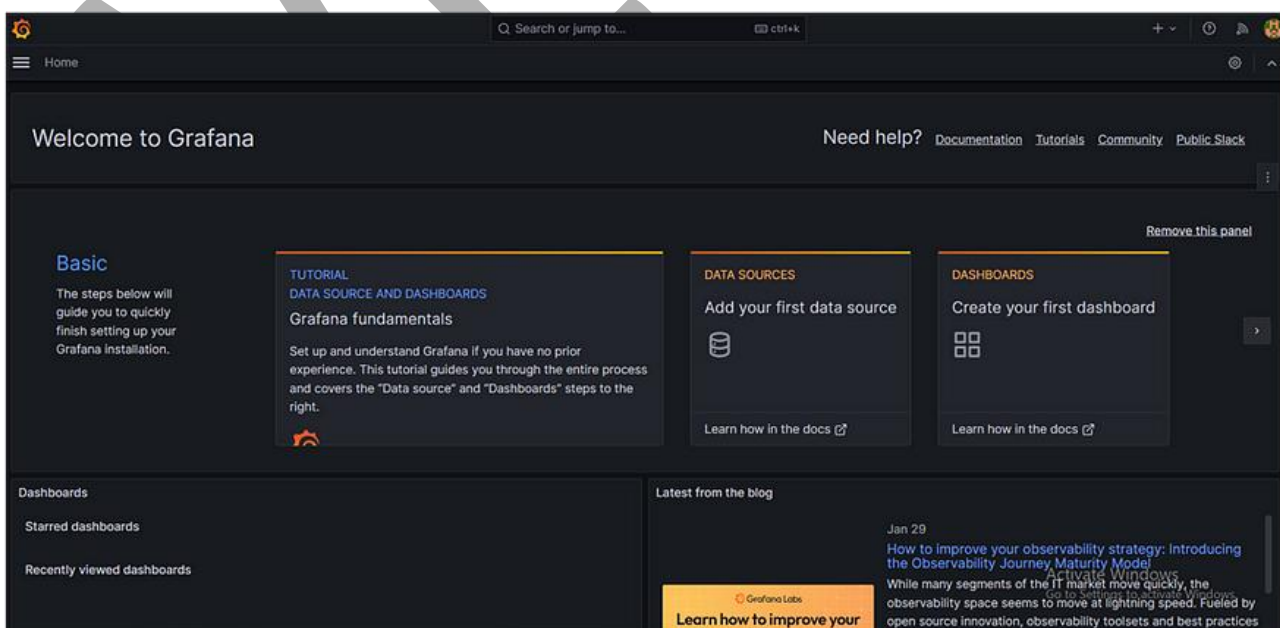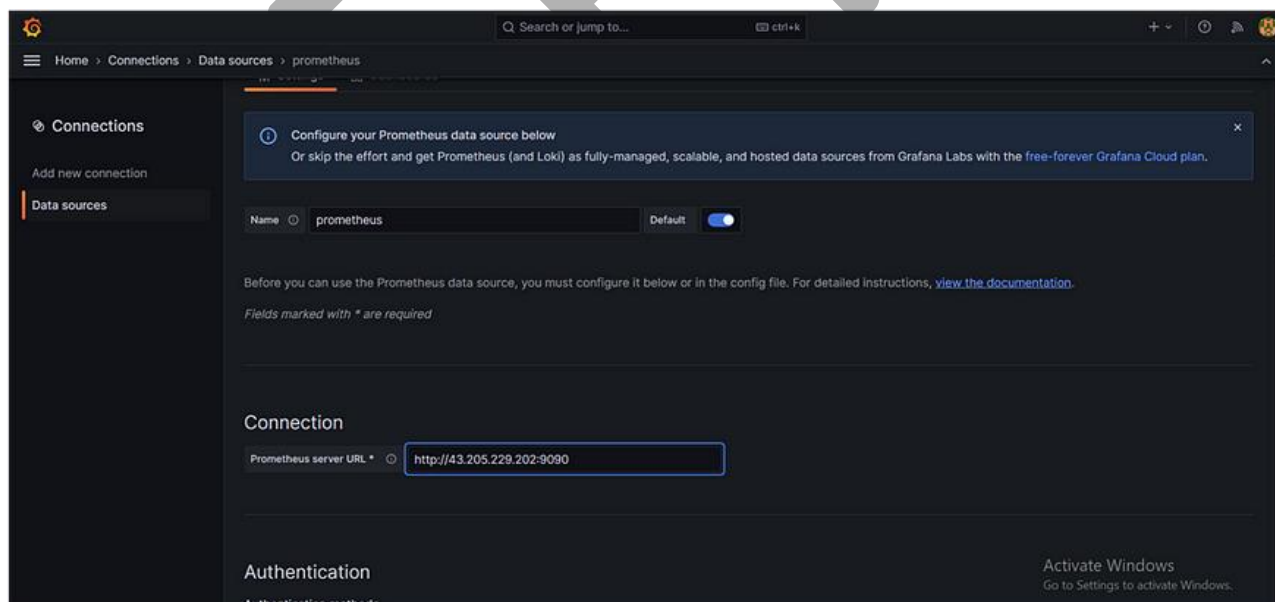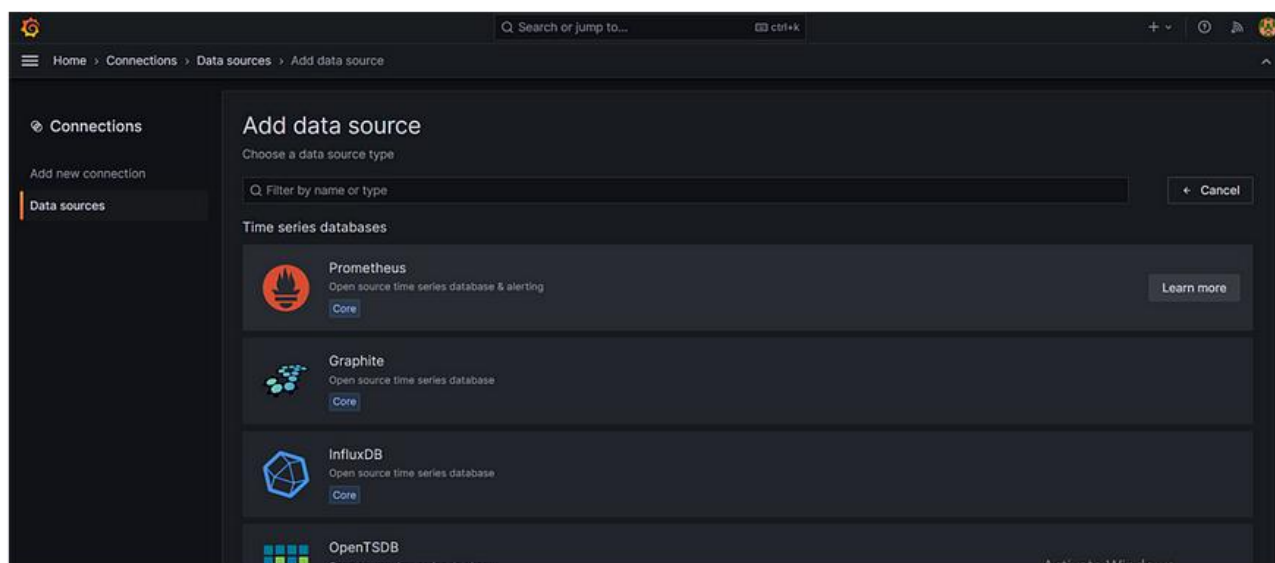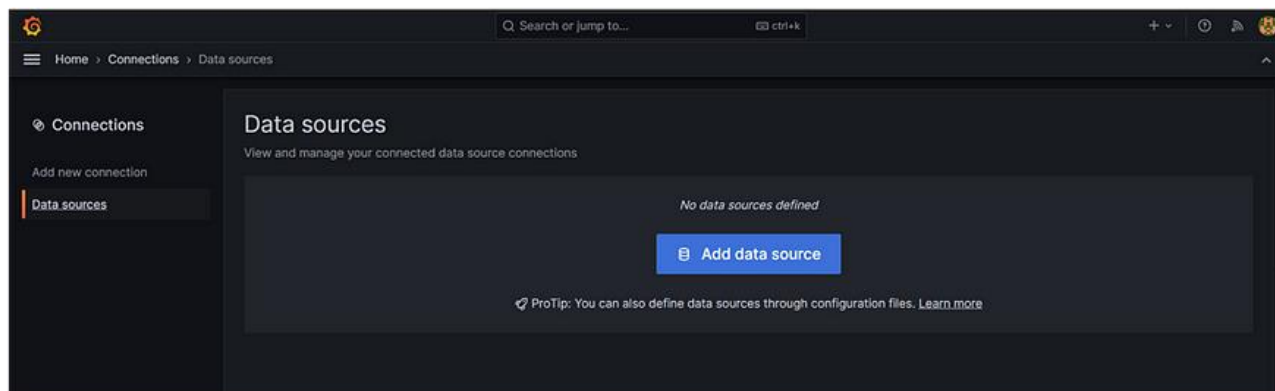
Create Dashboard

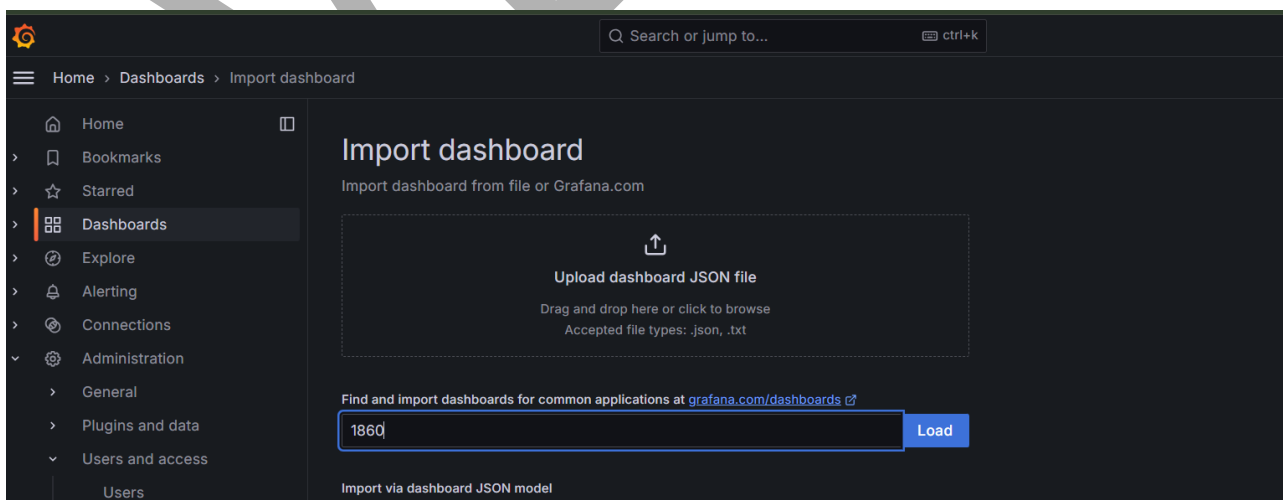Click on new

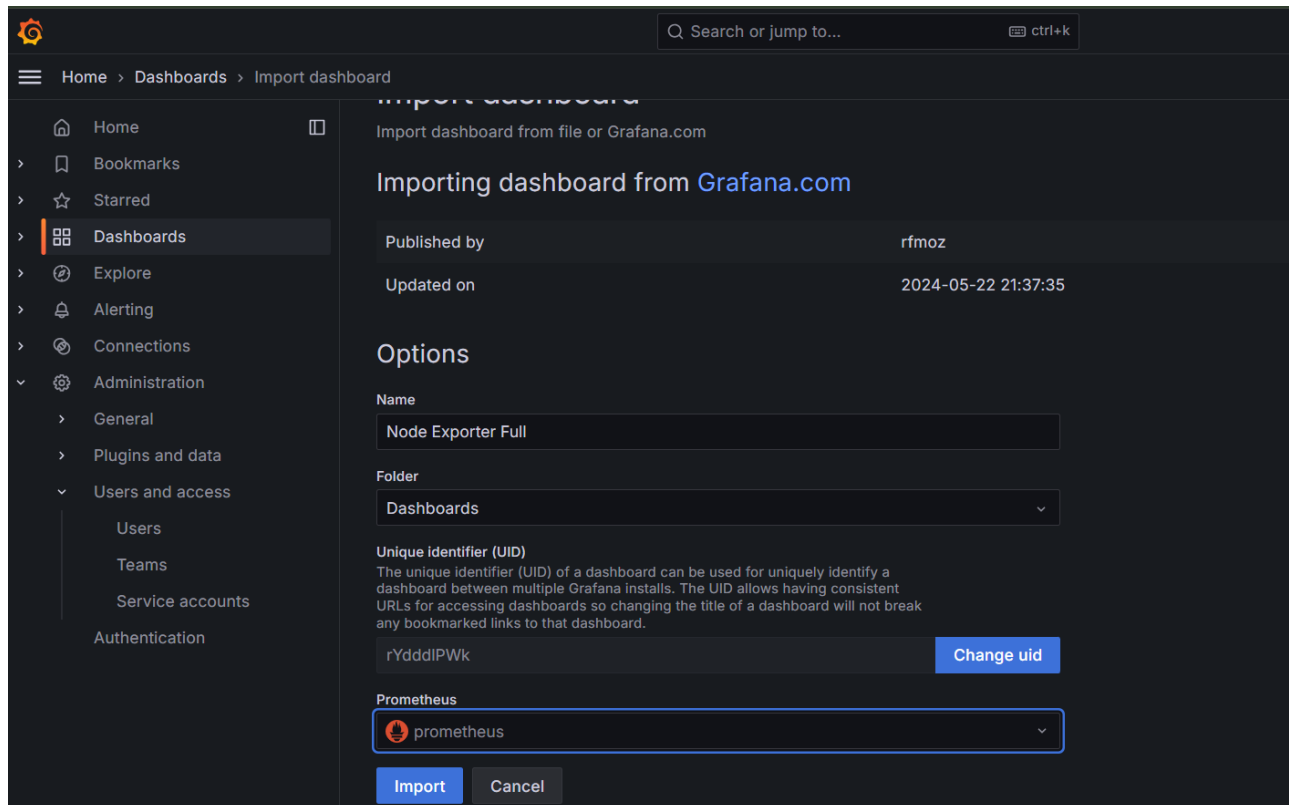Multi cloud with devops by veera nareshit

Click on import

Enter the dasbord number ==1860 then click on load

Select the source is prometheus and clicki on import

Multi cloud with devops by veera nareshit

Multi cloud with devops by veera nareshit

Install stress

sudo apt update

sudo apt install -y stress

stress --vm 1 --vm-bytes 512mb --vm-keep --timeout 60s

stress --cpu 1 --timeout 60

Multi cloud with devops by veera nareshit

Multi cloud with devops by veera nareshit


```
root@ip-172-31-39-112:~# sudo apt-get install stress
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  stress
0 upgraded, 1 newly installed, 0 to remove and 55 not upgraded.
Need to get 18.4 kB of archives.
After this operation, 52.2 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 stress amd64 1.0.5-1 [18.4 kB]
Fetched 18.4 kB in 0s (45.3 kB/s)
Selecting previously unselected package stress.
(Reading database ... 76102 files and directories currently installed.)
```

Final Dashboard

Conclusion

Multi cloud with devops by veera nareshit