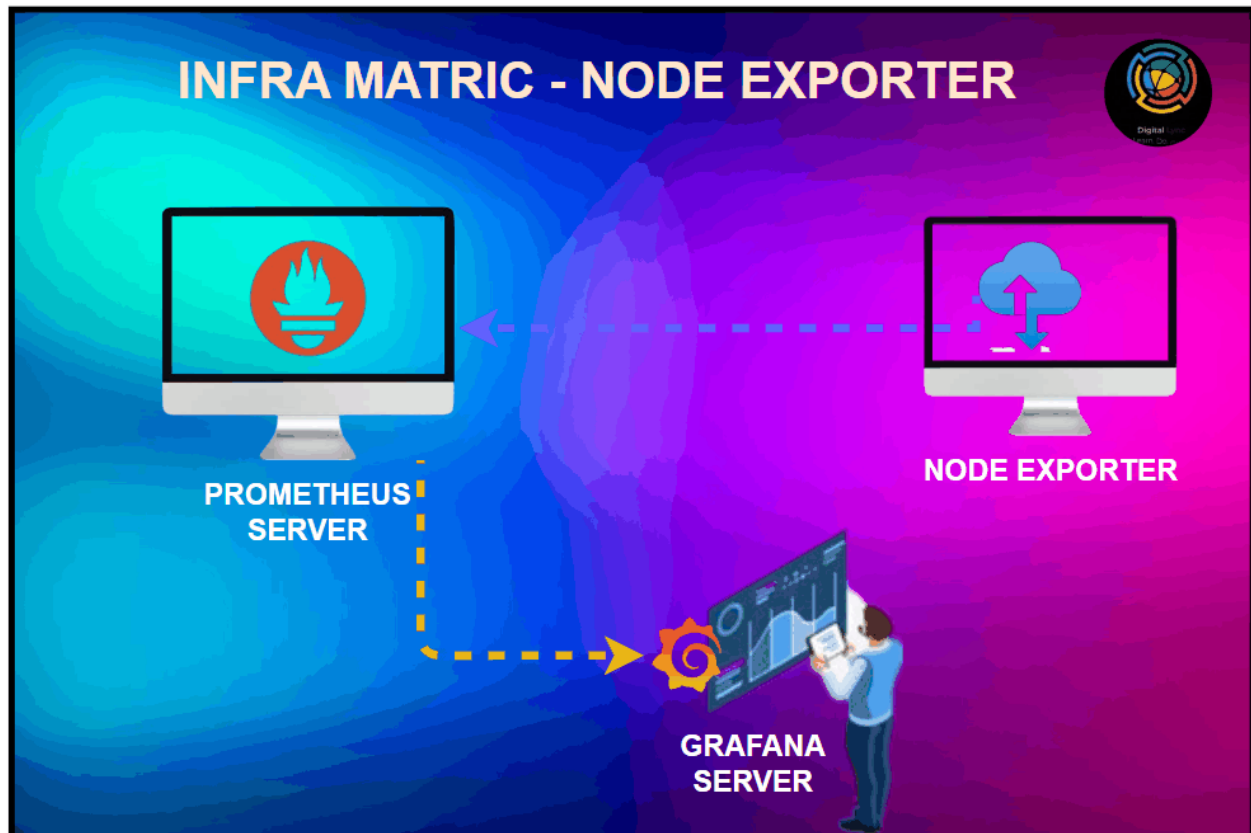


# Stories 3: Infra metrics - Node Exporter - Prometheus - Grafana



- ☐ INTRODUCTION TO PROMETHEUS AND GRAFANA
- ☐ ARHCITECTURE OF PROMETHEUS
- ☐ PROMETHEUS INSTALLATION ON LINUX
- ☐ NODE EXPORTER ON APPLICATION
- ☐ CONFIGURATIONS OF PROMETHEUS WITH NODE-EXPORTER
- ☐ SCRAPING THE METRICS FROM THE APPLICATION SERVER
- ☐ GRAFANA INSTALLATION
- ☐ CONFIGURING PROMETHEUS WITH GRAFANA

## ❑ GRAFANA DASHBOARDS FOR APPLICATION SERVER

### Prometheus and Grafana

- Prometheus and Grafana two powerful tools that are used for monitoring and visualization in the world of cloud technology.
- Prometheus is an open-source monitoring and alerting system, while Grafana is a feature-rich data visualization platform.
- Together, they form a robust combination that helps organizations gain insights into their systems' performance and health.

### Introduction to Prometheus

- *Prometheus as a monitoring and alerting toolkit*
- *Developed at SoundCloud*
- *Built with a focus on reliability, scalability, and simplicity*
- *Core components: Prometheus Server, exporters, and client*



*Prometheus provides several key features that make it a powerful monitoring tool:*

## **Data Collection:**

*Prometheus collects metrics from various sources, including HTTP endpoints, exporters, and client libraries. It supports dynamic service discovery, allowing it to automatically discover and monitor new instances as they come online.*

## **Data Storage:**

*Prometheus has its own built-in time series database (TSDB) that efficiently stores and indexes collected metrics. The TSDB allows for efficient querying and analysis of historical data.*

## **Query Language:**

*Prometheus uses PromQL (Prometheus Query Language) to query and aggregate collected metrics. PromQL allows users to perform powerful and flexible queries to extract specific information from the collected data.*

## **Alerting:**

*Prometheus has a built-in alerting mechanism that allows users to define alerting rules based on metric thresholds or more complex conditions. When an alert rule is triggered, Prometheus can send notifications to various integrations, such as email, PagerDuty, or slack .*

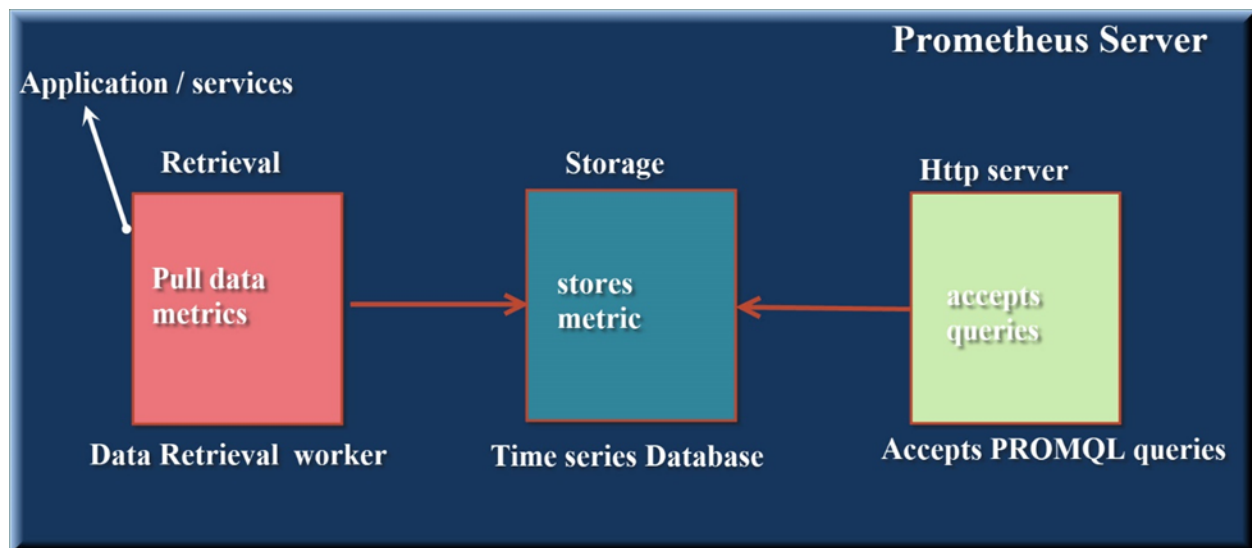
## **Visualization:**

*While Prometheus itself provides a basic web-based interface for querying and exploring metrics, it is often used in conjunction with visualization tools like Grafana. Grafana integrates with Prometheus to provide a feature-rich, customizable dashboarding and visualization experience.*

## **Time Series Database:**

*Prometheus uses its own time-series database for storing collected metrics. The data is stored in a compressed and efficient format, enabling quick retrieval and analysis.*

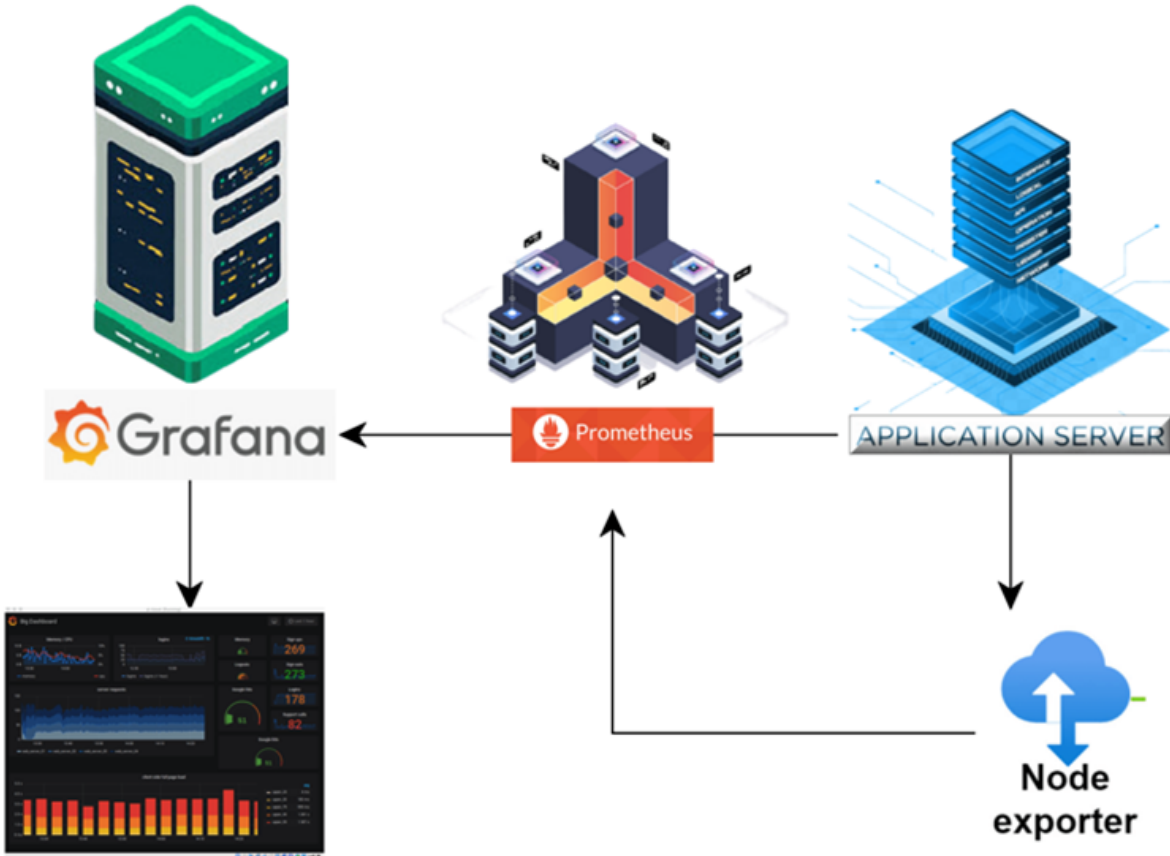
# ARCHITECTURE OF PROMETHEUS



## Lab :

Launch Three Servers

# Prometheus and Grafana Infra Metrics



Install Prometheus in Prometheus server, Application server and grafana server

Server 1	1. Prometheus Installation on Linux	
Server 2	2. Application Server	
server 3	3. Grafana Server	

Instances (12) <a href="#">Info</a>						
<input type="text" value="Find instance by attribute or tag (case-sensitive)"/>						
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	
<input type="checkbox"/>	Prometheus	<a href="#">i-0072d8864a90f7776</a>	Running	t2.medium	2/2 checks passed	
<input type="checkbox"/>	Grafana	<a href="#">i-0a522ea4fbcfb9335</a>	Running	t2.medium	2/2 checks passed	
<input type="checkbox"/>	Node Exporter / App server	<a href="#">i-041bb4e3664722405</a>	Running	t2.medium	2/2 checks passed	

Server 1 (Prometheus server)

**Launch the Ubuntu version 20 Virtual Machine:**

- Install the ubuntu version 20 from the AWS Cloud
- Allow the Inbound rule 9090 which is the port number of Prometheus
- To install the Prometheus visit the official website site <https://prometheus.io>
- For the Installation of Prometheus we have created a below script file
- Name of the script file is sh Prometheus repo  
**<https://github.com/mubeen507/Prometheus.git>**

<https://github.com/prometheus/prometheus/releases/download/v2.50.0-rc.1/prometheus-2.50.0-rc.1.linux-amd64.tar.gz>

```
#!/bin/bash
sudo apt update

sudo wget https://github.com/prometheus/prometheus/releases/dow

sudo groupadd --system prometheus

sudo useradd -s /sbin/nologin --system -g prometheus prometheus

sudo mkdir /var/lib/prometheus

sudo mkdir -p /etc/prometheus/rules

sudo mkdir -p /etc/prometheus/rules.s

sudo mkdir -p /etc/prometheus/files_sd
```

```

sudo tar xvf prometheus-2.45.0-rc.0.linux-amd64.tar.gz

cd prometheus-2.45.0-rc.0.linux-amd64

sudo mv prometheus promtool /usr/local/bin/

sudo mv prometheus.yml /etc/prometheus/prometheus.yml

sudo tee /etc/systemd/system/prometheus.service<<EOF
[Unit]
Description=Prometheus
Documentation=https://prometheus.io/docs/introduction/overview/
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=prometheus
Group=prometheus
ExecReload=/bin/kill -HUP $MAINPID
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 \
  --web.listen-address=0.0.0.0:9090 \
  --web.external-url=

SyslogIdentifier=prometheus
Restart=always

[Install]
WantedBy=multi-user.target
EOF

```

Starting Service , enabling service and assinging permissions

```
sudo chown -R prometheus:prometheus /etc/prometheus

sudo chown -R prometheus:prometheus /etc/prometheus/*

sudo chmod -R 775 /etc/prometheus

sudo chmod -R 755 /etc/prometheus/*

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

sudo chown -R prometheus:prometheus /var/lib/prometheus/*

sudo systemctl daemon-reload

sudo systemctl start prometheus

sudo systemctl enable prometheus
```

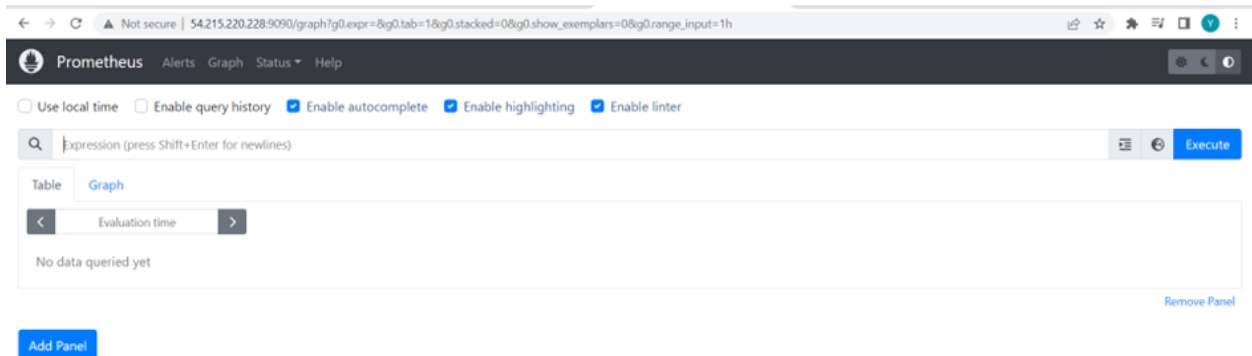
Explanation:

The line "WantedBy=multi-user.target" appears to be a configuration directive in a systemd service unit file. In systemd, service unit files are used to define and manage services and daemons on a Linux system. The "WantedBy" directive specifies the target units that should start the service when those targets are activated.

In this case, "WantedBy=multi-user.target" means that the service is configured to start when the "multi-user.target" is activated. The "multi-user.target" is typically a target unit that represents the system's multi-user runlevel, which is a state where the system is fully operational and available for multiple users to log in.



- Take the public ip of the instance and check in browser for the Prometheus application .



- **Successfully we launched the Prometheus application**

Install node exporter in application server

## 2. NODE EXPORTER ON APPLICATION SERVER

Server 2 (Node exporter with any appserver)

### Launch the Ubuntu version 20 Virtual Machine:

- Install the ubuntu version 20 from the AWS Cloud
- Allow the Inbound rule 9100 which is the port number of node-exporter
- To install the node-exporter visit the official website site <https://prometheus.io>
- For the Installation of node-exporter we have created a below script file
- Name of the script file is node-exporter.sh

```
#!/bin/bash
sudo apt update

sudo wget https://github.com/prometheus/node_exporter/releases/
```

```

sudo groupadd --system prometheus

sudo useradd -s /sbin/nologin --system -g prometheus prometheus

sudo mkdir /var/lib/node

sudo tar xvf node_exporter-1.6.0.linux-amd64.tar.gz

cd node_exporter-1.6.0.linux-amd64

sudo mv node_exporter /var/lib/node

sudo tee /etc/systemd/system/node.service<<EOF
[Unit]
Description=Prometheus Node Exporter
Documentation=https://prometheus.io/docs/introduction/overview/
Wants=network-online.target
After=network-online.target

[Service]
Type=simple
User=prometheus
Group=prometheus
ExecReload=/bin/kill -HUP $MAINPID
ExecStart=/var/lib/node/node_exporter

SyslogIdentifier=prometheus_node_exporter
Restart=always

[Install]
WantedBy=multi-user.target
EOF

```

```
sudo chown -R prometheus:prometheus /var/lib/node

sudo chown -R prometheus:prometheus /var/lib/node/*

sudo chmod -R 775 /var/lib/node

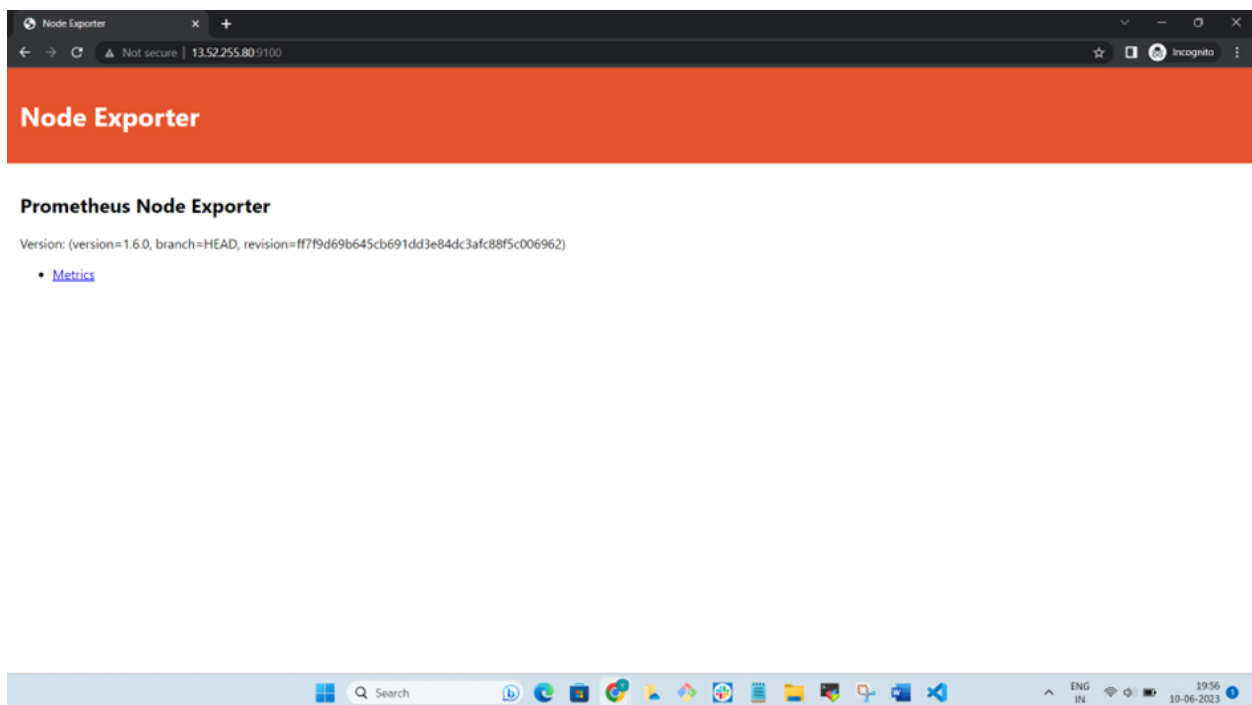
sudo chmod -R 755 /var/lib/node/*

sudo systemctl daemon-reload

sudo systemctl start node

sudo systemctl enable node
```

- Take the public ip of the instance and check in browser for the node-exporter



Configure Prometheus with node exporter in  
Prometheus server by editing Prometheus.yml file

### 3.CONFIGURATIONS OF PROMETHEUS WITH NODE-EXPORTER

- Inorder to configure the Prometheus with the node-exporter application server we need to edit the Prometheus.yml file which is located in /etc/prometheus/prometheus.yml
- We need to configure the private ip address of the node-exporter server along with port number.
- Here the example will be 31.15.104:9100.

```
ubuntu@ip-172-31-6-65:~$ cd /etc/prometheus/
ubuntu@ip-172-31-6-65:/etc/prometheus$ cat prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# 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"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: my-application

    static_configs:
      - targets: ["172.31.15.104:9100"]
```

- If you check the status of the Prometheus in Targets section you will find the node-exporter status up and running restart the Prometheus

sudo systemctl restart prometheus

The screenshot shows the Prometheus web interface. The 'Targets' section lists two scrape targets:

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.31.15.104:9100/metrics	UP	instance="172.31.15.104:9100" job="my-application"	6.817s ago	11.723ms	
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	8.557s ago	4.240ms	

#### 4. SCRAPING THE METRICS FROM THE APPLICATION SERVER

##### FREE RAM METRIC:

- In order to get the free RAM on an application server we need to provide PROMQL Query on the Prometheus server.

`node_memory_MemFree_bytes /1024 /1024 /1024`

The screenshot shows the Prometheus query editor with the query `node_memory_MemFree_bytes /1024 /1024 /1024` entered. The 'Table' view is selected, showing the following result:

Labels	Value
{instance="172.31.15.104:9100", job="my-application"}	3.2449188232421875

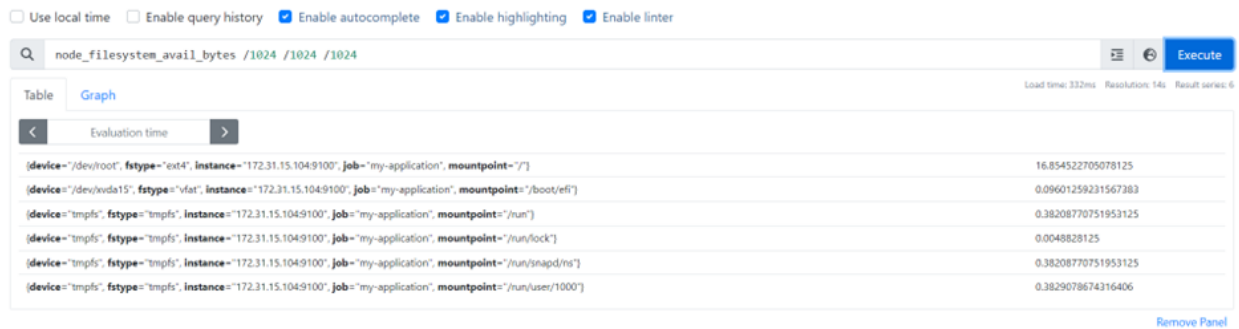
- Checking the free RAM on the Application Server by the command `free -h`

```
ubuntu@ip-172-31-15-104:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:           3.8Gi        198Mi        3.2Gi         0.0Ki         400Mi        3.4Gi
Swap:              0B              0B              0B
```

## HARD DISK SPACE:

- In order to scrape the Hard Disk Space from the Application Server we need to Query the Prometheus Server

```
node_filesystem_avail_bytes /1024 /1024 /1024
```



- Checking the Hard Disk Space on the Application Server by the command `df -h`.

```
ubuntu@ip-172-31-15-104:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        20G   2.4G   17G   13% /
devtmpfs         2.0G     0   2.0G    0% /dev
tmpfs            2.0G     0   2.0G    0% /dev/shm
tmpfs            393M   864K   392M    1% /run
tmpfs            5.0M     0   5.0M    0% /run/lock
tmpfs            2.0G     0   2.0G    0% /sys/fs/cgroup
/dev/loop0       25M    25M     0 100% /snap/amazon-ssm-agent/6563
/dev/loop1       25M    25M     0 100% /snap/amazon-ssm-agent/6312
/dev/loop2       56M    56M     0 100% /snap/core18/2751
/dev/loop3       56M    56M     0 100% /snap/core18/2785
/dev/loop5       92M    92M     0 100% /snap/lxd/24061
/dev/loop6       54M    54M     0 100% /snap/snapd/19122
/dev/loop4       64M    64M     0 100% /snap/core20/1891
/dev/loop7       54M    54M     0 100% /snap/snapd/19361
/dev/xvda15     105M   6.1M    99M    6% /boot/efi
tmpfs            393M     0   393M    0% /run/user/1000
ubuntu@ip-172-31-15-104:~$
```

## TOTAL HARD DISK SIZE:

- In order to get the Total Hard Disk Size on an application server we need to provide PROMQL Query on the Prometheus server.

```
node_filesystem_size_bytes /1024 /1024 /1024
```

☐ Use local time
 ☐ Enable query history
 ☒ Enable autocomplete
 ☒ Enable highlighting
 ☒ Enable linter

Q `node_filesystem_size_bytes /1024 /1024 /1024` Execute

Table [Graph](#) Load time: 487ms Resolution: 14s Result series: 6

Evaluation time	
{device="/dev/root", fstype="ext4", instance="172.31.15.104:9100", job="my-application", mountpoint="/"}	19.20184326171875
{device="/dev/xvda15", fstype="vfat", instance="172.31.15.104:9100", job="my-application", mountpoint="/boot/efi"}	0.10190773010253906
{device="tmpfs", fstype="tmpfs", instance="172.31.15.104:9100", job="my-application", mountpoint="/run"}	0.38291168212890625
{device="tmpfs", fstype="tmpfs", instance="172.31.15.104:9100", job="my-application", mountpoint="/run/lock"}	0.0048828125
{device="tmpfs", fstype="tmpfs", instance="172.31.15.104:9100", job="my-application", mountpoint="/run/snapd/ns"}	0.38291168212890625
{device="tmpfs", fstype="tmpfs", instance="172.31.15.104:9100", job="my-application", mountpoint="/run/user/1000"}	0.3829078674316406

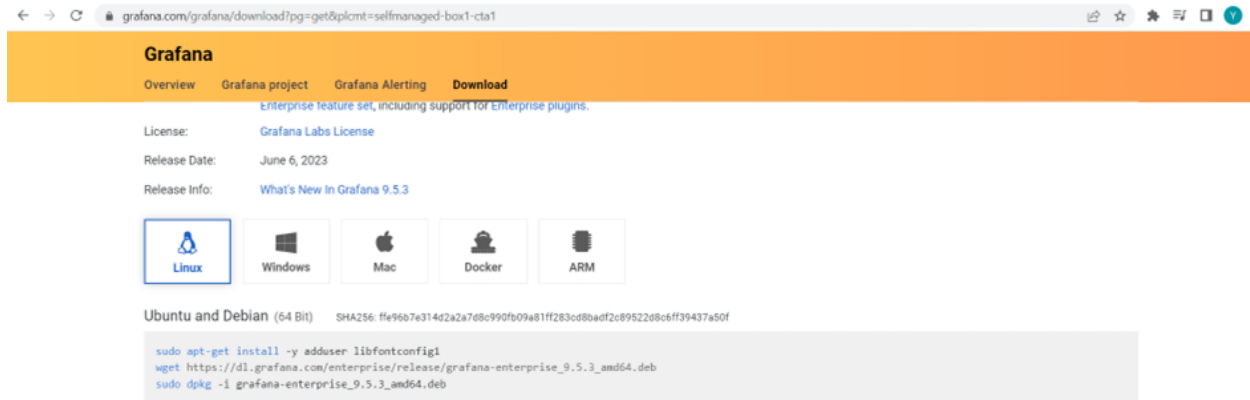
[Remove Panel](#)

## 5. GRAFANA

## Launch the Ubuntu version 20 Virtual Machine:

- Install the ubuntu version 20 from the AWS Cloud

- Allow the Inbound rule 3000 which is the port number of Grafana
- To install the Grafana visit the official website site <https://grafana.com>
- For the Installation of Grafana we have created a below script file
- Name of the script file is sh



```

#!/bin/bash
sudo apt-get install -y adduser libfontconfig1
wget https://dl.grafana.com/enterprise/release/grafana-enterprise_9.5.3_amd64.deb
sudo dpkg -i grafana-enterprise_9.5.3_amd64.deb

sudo systemctl start grafana-server
sudo systemctl enable grafana-server
sudo systemctl status grafana-server
  
```

- Take the public ip of the instance and check in browser for the Grafana application .
-



```

sudo 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 Sat 2023-06-10 06:38:27 UTC; 7h ago
     Docs: http://docs.grafana.org
   Main PID: 1124 (grafana)
    Tasks: 11 (limit: 4686)
   Memory: 160.6M
   CGroup: /system.slice/grafana-server.service
           └─1124 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default

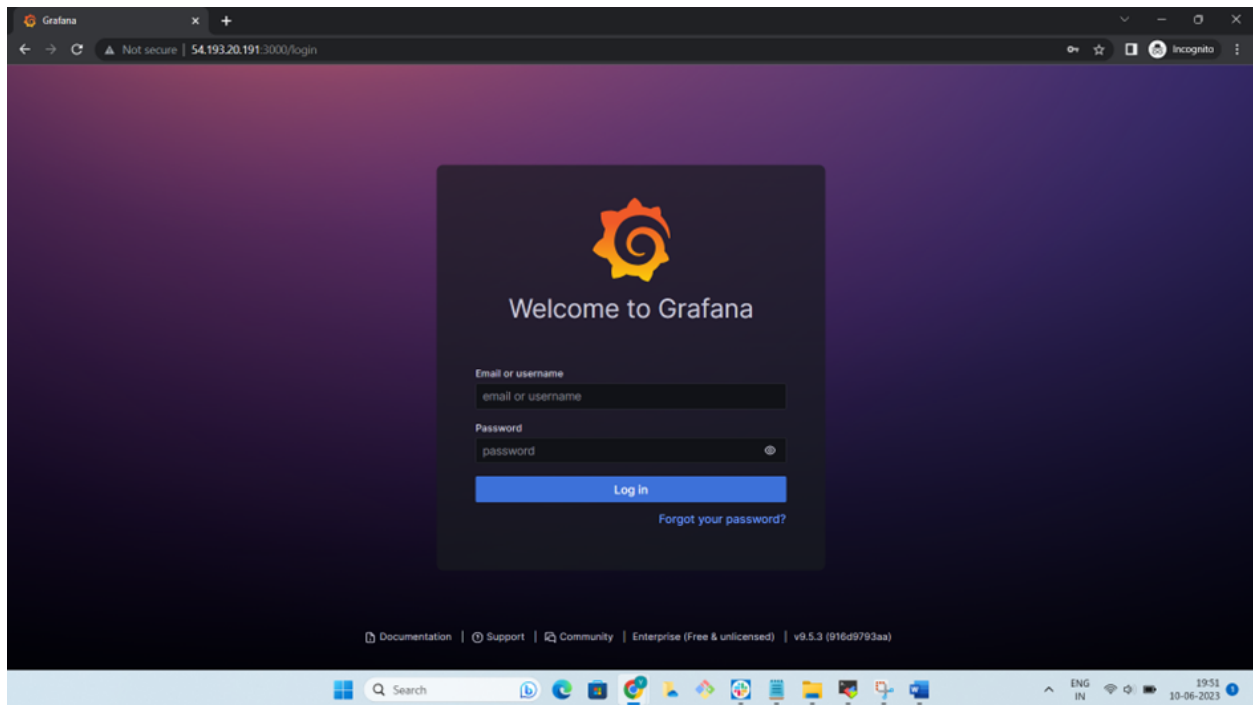
Jun 10 13:58:30 ip-172-31-11-114 grafana[1124]: logger=grafana.update.checker t=2023-06-10T13:58:30.45682964Z level=info msg="Update check succeeded" duration=
Jun 10 13:58:30 ip-172-31-11-114 grafana[1124]: logger=plugins.update.checker t=2023-06-10T13:58:30.590998551Z level=info msg="Update check succeeded" duration=
Jun 10 14:02:56 ip-172-31-11-114 grafana[1124]: logger=context userId=1 orgId=1 uname=admin t=2023-06-10T14:02:56.224331992Z level=info msg="Request Completed"
Jun 10 14:03:00 ip-172-31-11-114 grafana[1124]: logger=ngalert.state.manager rule uid=b777de44-32ca-4c7a-b328-7d53f44dccc3c org_id=1 t=2023-06-10T14:03:00.00652
Jun 10 14:08:30 ip-172-31-11-114 grafana[1124]: logger=cleanup t=2023-06-10T14:08:30.388274069Z level=info msg="Completed cleanup jobs" duration=1.429677ms
Jun 10 14:08:30 ip-172-31-11-114 grafana[1124]: logger=grafana.update.checker t=2023-06-10T14:08:30.460786653Z level=info msg="Update check succeeded" duration=
Jun 10 14:08:30 ip-172-31-11-114 grafana[1124]: logger=plugins.update.checker t=2023-06-10T14:08:30.592490618Z level=info msg="Update check succeeded" duration=
Jun 10 14:18:30 ip-172-31-11-114 grafana[1124]: logger=cleanup t=2023-06-10T14:18:30.38585197Z level=info msg="Completed cleanup jobs" duration=1.342657ms
Jun 10 14:18:30 ip-172-31-11-114 grafana[1124]: logger=grafana.update.checker t=2023-06-10T14:18:30.457722014Z level=info msg="Update check succeeded" duration=
Jun 10 14:18:30 ip-172-31-11-114 grafana[1124]: logger=plugins.update.checker t=2023-06-10T14:18:30.590189437Z level=info msg="Update check succeeded" duration=

```

## • Successfully we launched the Grafana application

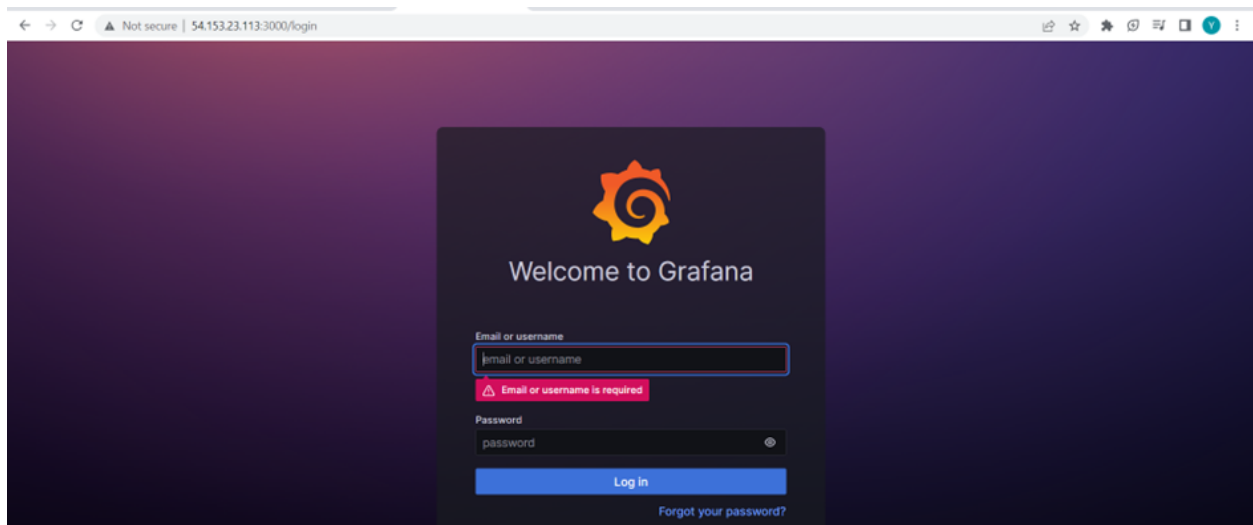
### 6. CONFIGURING PROMETHEUS WITH GRAFANA

- Login into the Grafana Server with the username and password
- After Login into the Grafana goto the Connections option and click on your connections
- Click on Add data source and add Prometheus as shown in the below pics
- After giving the Prometheus URL click on SAVE&TEST option in order to test the connection between Prometheus and Grafana.
- Successfully Prometheus and Grafana configured
-



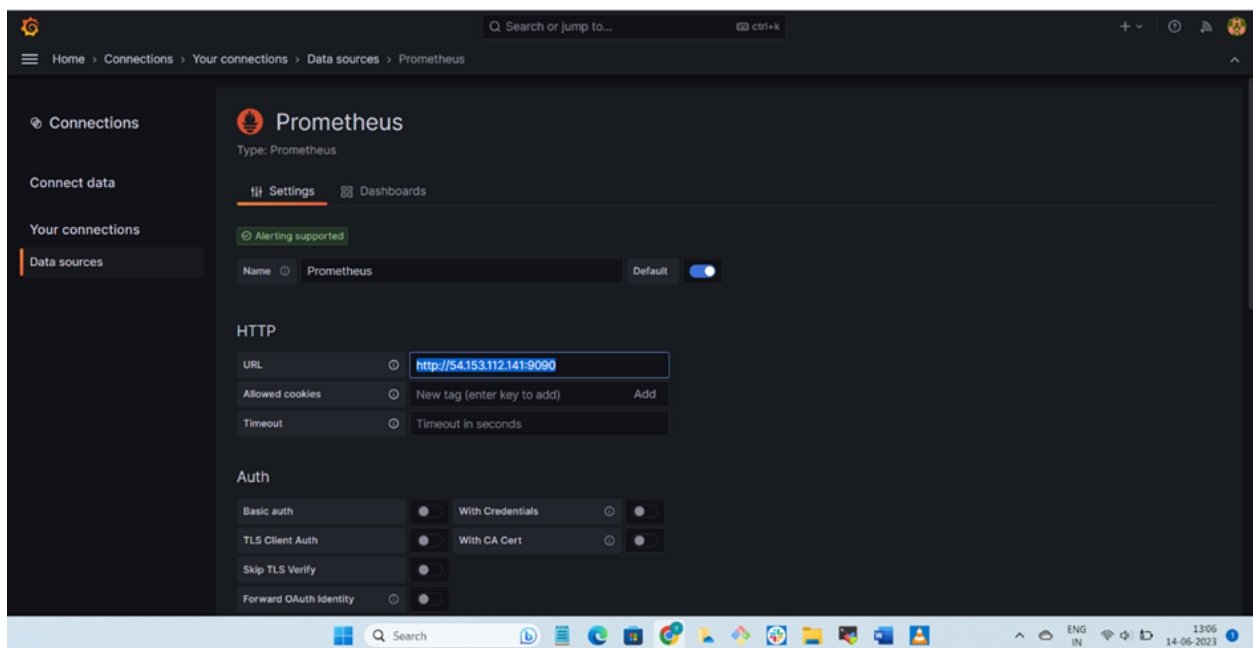
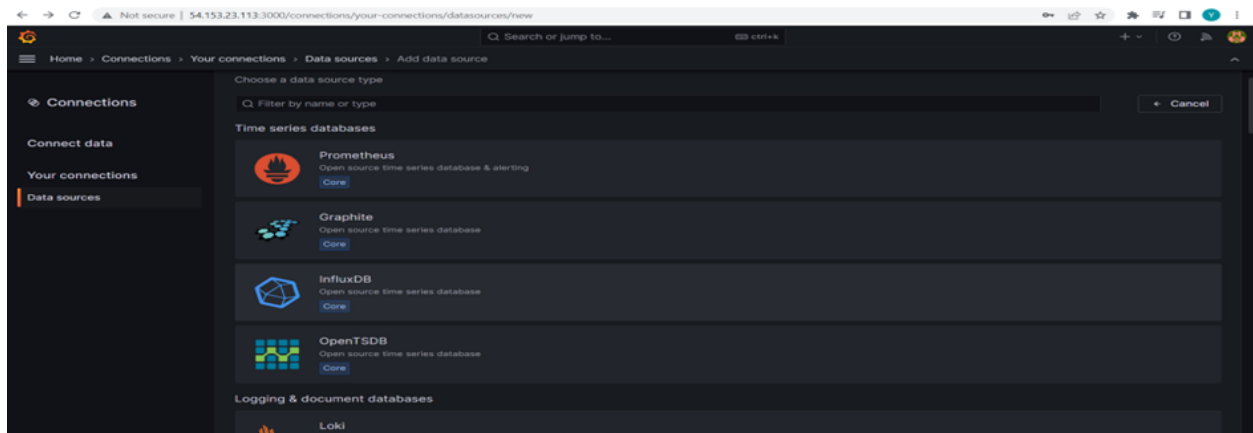
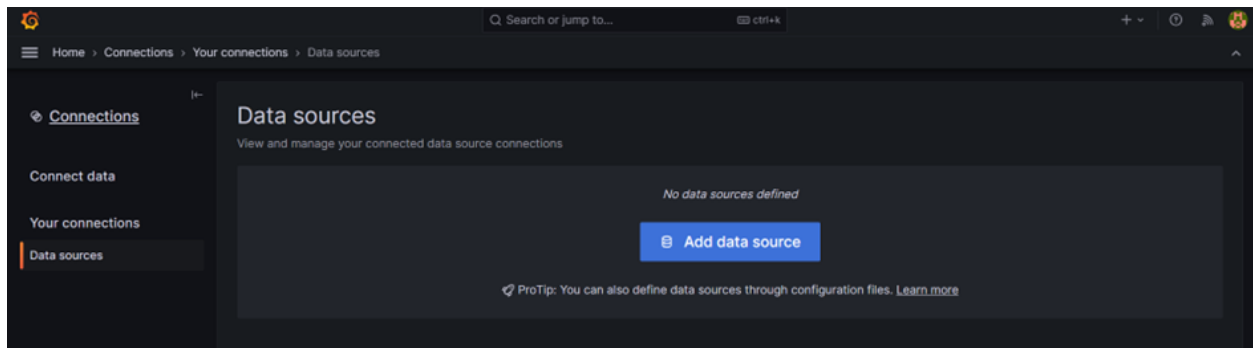
## 6. CONFIGURING PROMETHEUS WITH GRAFANA

- Login into the Grafana Server with the username and password
- 

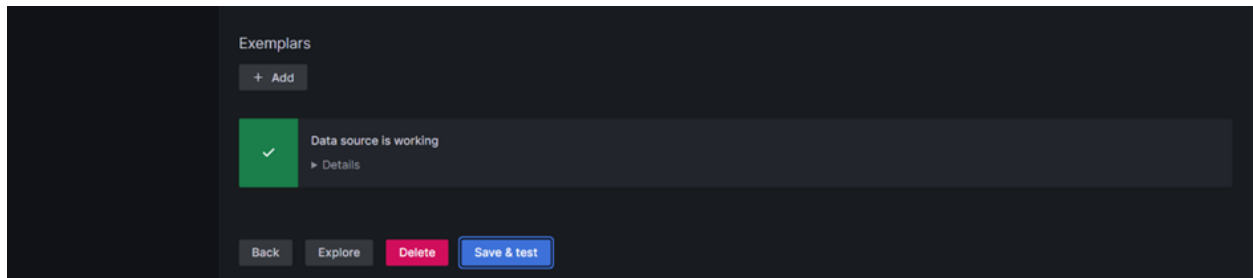


- After Login into the Grafana goto the Connections option and click on your connections

- Click on Add data source and add Prometheus as shown in the below pics
- 



- After giving the Prometheus URL click on SAVE&TEST option in order to test the connection between Prometheus and Grafana.
- 



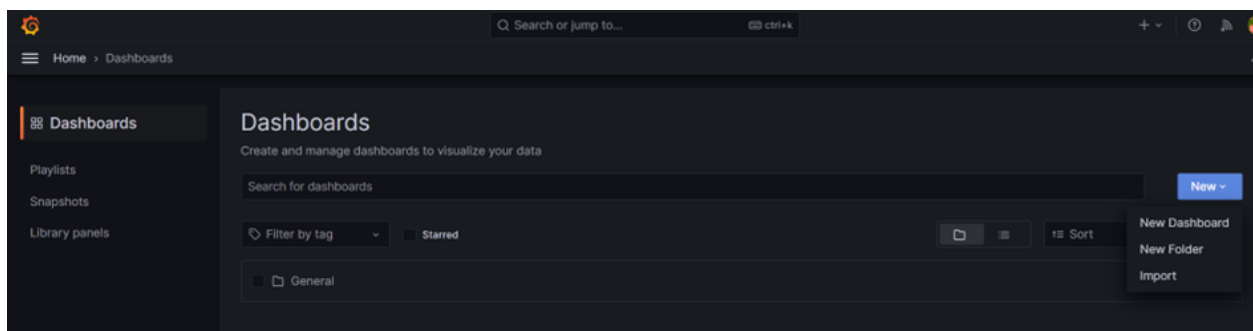
- Successfully Prometheus and Grafana configured

## 7. GRAFANA DASHBOARDS FOR APPLICATION SERVER

- Creating the Grafana Dashboards of the Application server metrics for FREE RAM , HARD DISK SPACE , TOTAL HARD DISK SIZE.

### 1.FREE RAM METRIC:

- Creating the Dashboard for the FREE RAM follow the below steps
- Create a new folder with the name Digital-lync
- 



## Create a new folder

Folders provide a way to group dashboards and alert rules.

Folder name

Create Cancel

### Digital-lync

Manage folder dashboards and permissions

[Dashboards](#) [Panels](#) [Alert rules](#) [Permissions](#) [Settings](#)

Search for dashboards

*This folder doesn't have any dashboards yet*

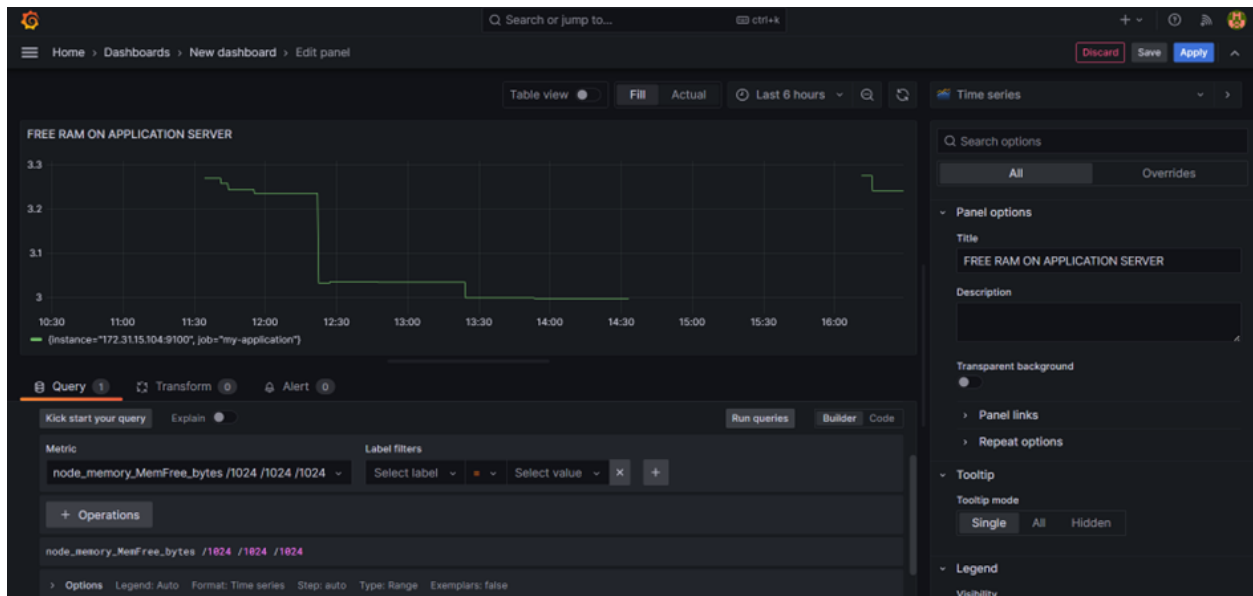
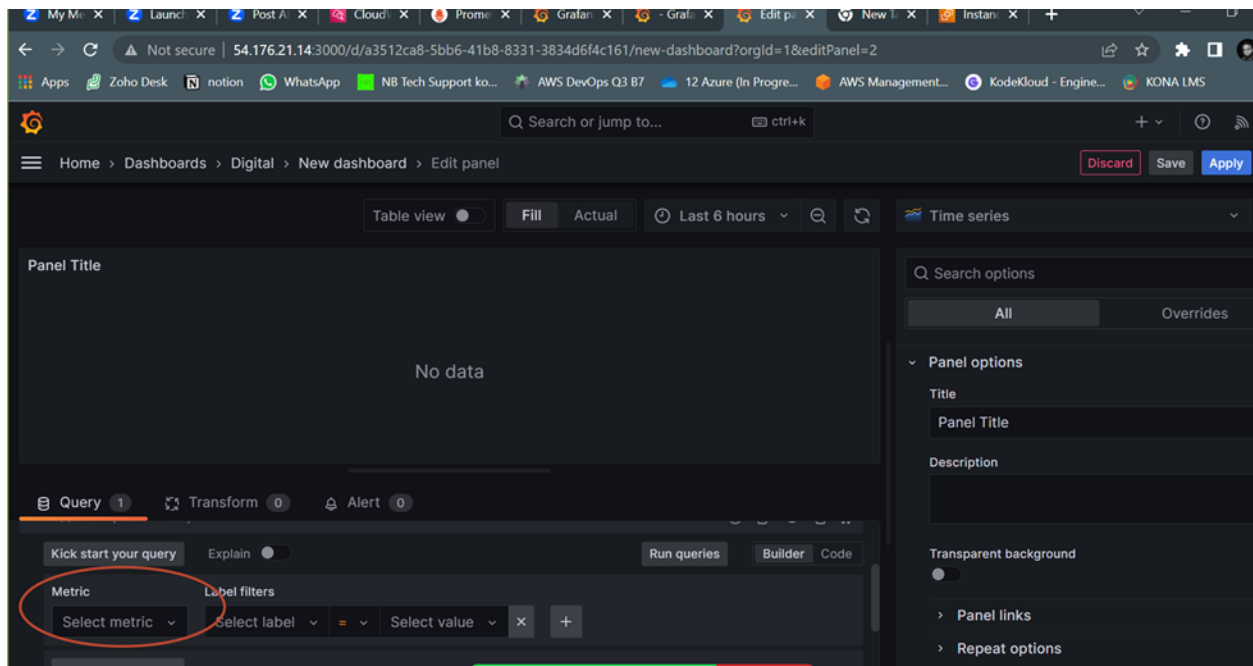
[+ Create Dashboard](#)

ProTip: Add/move dashboards to your folder at -> [Manage dashboards](#)

- Now create the Dashboard inside the folder Digital-lync
- We need to give the Prometheus query on the Grafana Dashboard as shown below then it will fetch the data from the Application server with the Prometheus Tool.

```
node_memory_MemFree_bytes /1024 /1024 /1024
```

Paste the query here



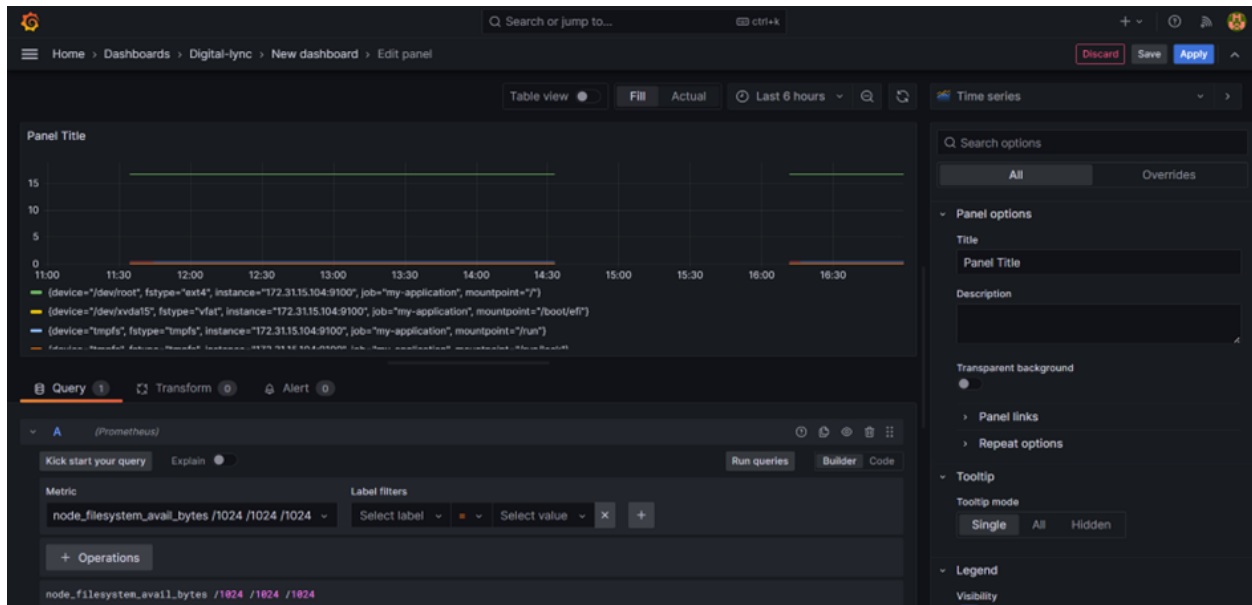
- The Grafana will fetch the live Free RAM data from the Application server with the provided time interval

## 2.HARD DISK SPACE METRIC:

- In the same dashboard create a another new visualization for the Hard Disk Space Metric.

- We need to give the Hard Disk Space Prometheus query on the Grafana Dashboard as shown below then it will fetch the data from the Application server with the Prometheus Tool.

```
node_filesystem_avail_bytes /1024 /1024 /1024
```

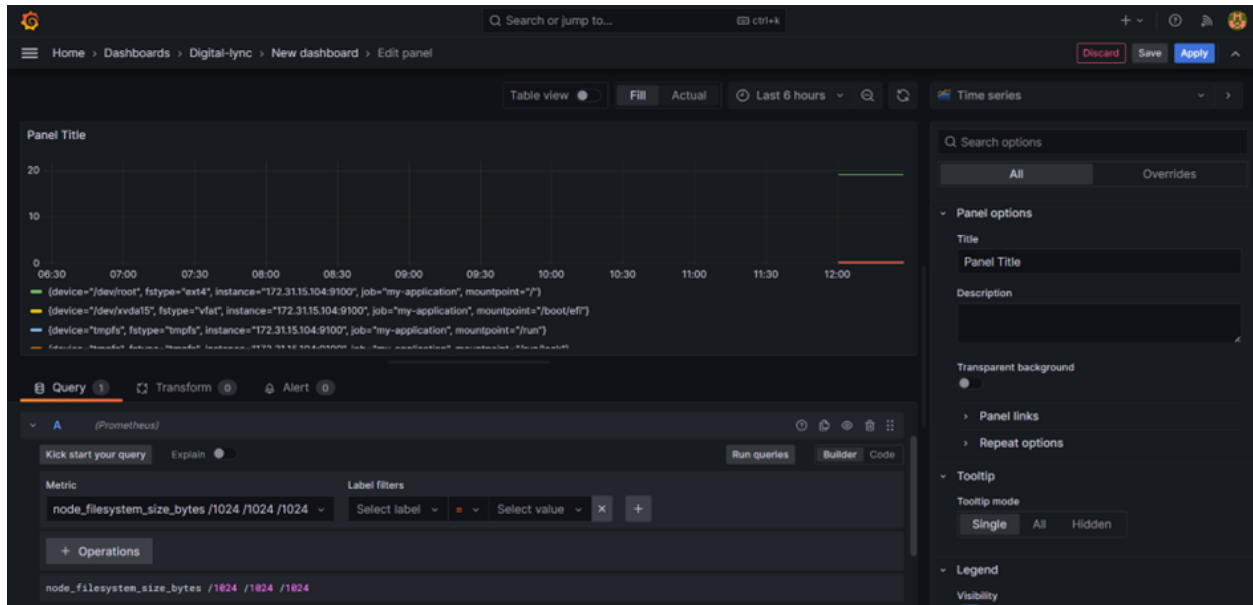


- The Grafana will fetch the live Hard Disk size Sapce data from the Application server.

## 1. TOTAL HARD DISK SIZE:

- In the same dashboard create a another new visualization for the Total Hard Disk Space Metric.
- We need to give the Total Hard Disk Space Prometheus query on the Grafana Dashboard as shown below then it will fetch the data from the Application server with the Prometheus Tool.

```
node_filesystem_size_bytes /1024 /1024 /1024
```



- The Grafana will fetch the live Total Hard Disk size Sapce data from the Application server .