MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CIT 6314 SYSTEM ADMINISTRATION

Trimester 1, 2024/2025

Project

Lecturer: Dr. Kairulanuar bin Ab Kadir

STUDENT NAME	STUDENT ID
Kristina Geetha Menon a/p Christopher Menon	1201201474

Table of Contents

1. **About the Project** 3
2. **Installation and Configuration of InfluxDB** 3
3. **Installation and Configuration of Telegraf** 6
4. **Installation and Configuration of Grafana** 11

1. About the Project

Component	Purpose
InfluxDB	A time-series database where Telegraf's system metrics are kept.
Telegraf	An agent that gathers system metrics and transmits them to InfluxDB, such as CPU usage, network utilization, etc.
Grafana	A visualization tool for building dashboards on real time that shows the metrics kept in InfluxDB.

2. Installation and Configuration of InfluxDB

a) Step 1: Add the InfluxDB repository

Command: sudo tee /etc/yum.repos.d/influxdb.repo <<EOF

[influxdb] – repository named "influxdb"

name=InfluxDB Repository – repository label

baseurl=https://repos.influxdata.com/centos/9/x86_64/stable – URL of the repository package located

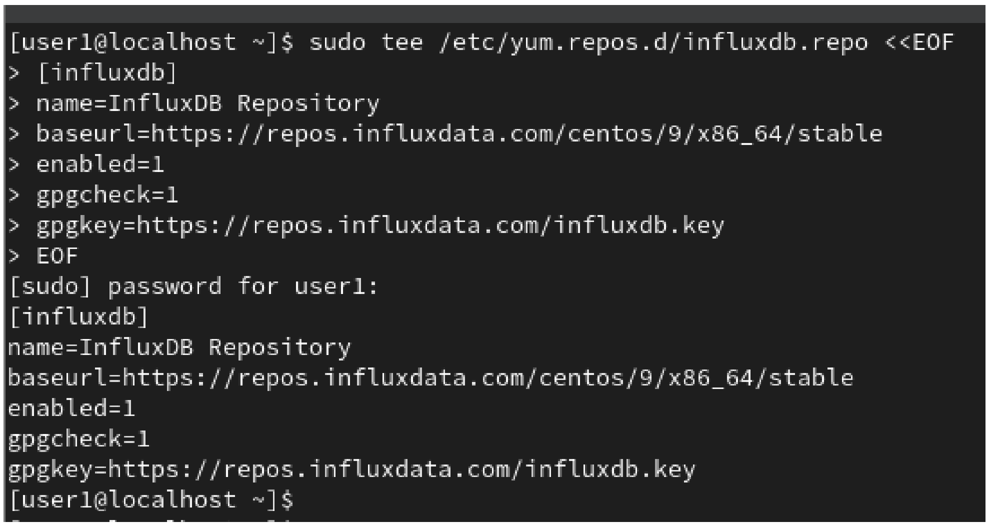
enabled=1 – makes this repository usable with yum

gpgcheck=1 – verify the package signature prior to installation

gpgkey=https://repos.influxdata.com/influxdb.key – URL to retrieve the GPG key for signature verification

EOF

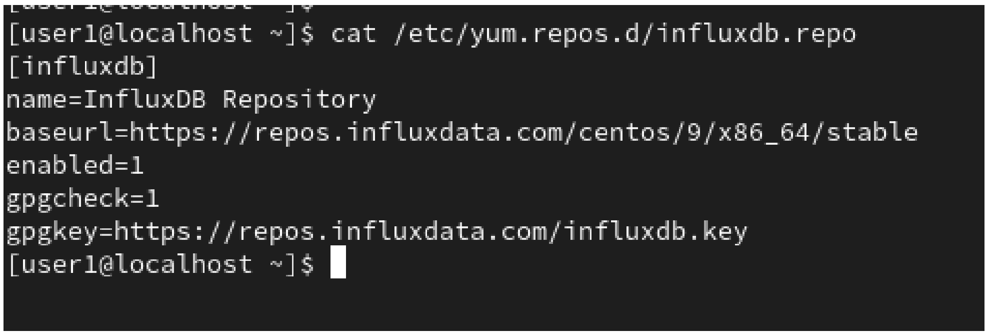
Function: Creates a CentOS repository configuration file which enables the yum package manager to be used to install and update InfluxDB.



```
[user1@localhost ~]$ sudo tee /etc/yum.repos.d/influxdb.repo <<EOF
> [influxdb]
> name=InfluxDB Repository
> baseurl=https://repos.influxdata.com/centos/9/x86_64/stable
> enabled=1
> gpgcheck=1
> gpgkey=https://repos.influxdata.com/influxdb.key
> EOF
[sudo] password for user1:
[influxdb]
name=InfluxDB Repository
baseurl=https://repos.influxdata.com/centos/9/x86_64/stable
enabled=1
gpgcheck=1
gpgkey=https://repos.influxdata.com/influxdb.key
[user1@localhost ~]$
```

b) Step 2: Reconfirm the creation of CentOS repository configuration file for InfluxDB.

Command: cat /etc/yum.repos.d/influxdb.repo

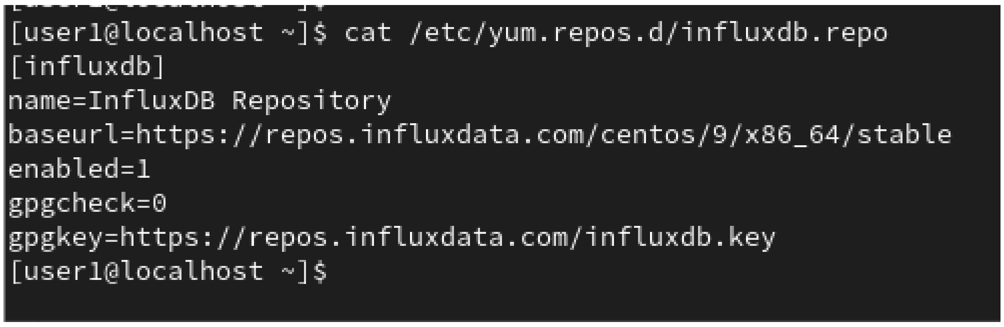


```
[user1@localhost ~]$ cat /etc/yum.repos.d/influxdb.repo
[influxdb]
name=InfluxDB Repository
baseurl=https://repos.influxdata.com/centos/9/x86_64/stable
enabled=1
gpgcheck=1
gpgkey=https://repos.influxdata.com/influxdb.key
[user1@localhost ~]$
```

Problem: The installation of InfluxDB (implemented in (c)) could not be done as the issue was found within the “gpgcheck = 1”. This line ensures the verification of package signatures prior to installation. Since there was an issue verifying it hence, the below will be the amendment of the configuration file:

“gpgcheck = 1” to “gpgcheck = 0”.

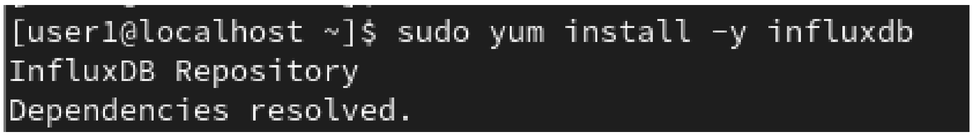
By setting the “gpgcheck = 0” this will allow the installation of InfluxDB to be carried out successfully.



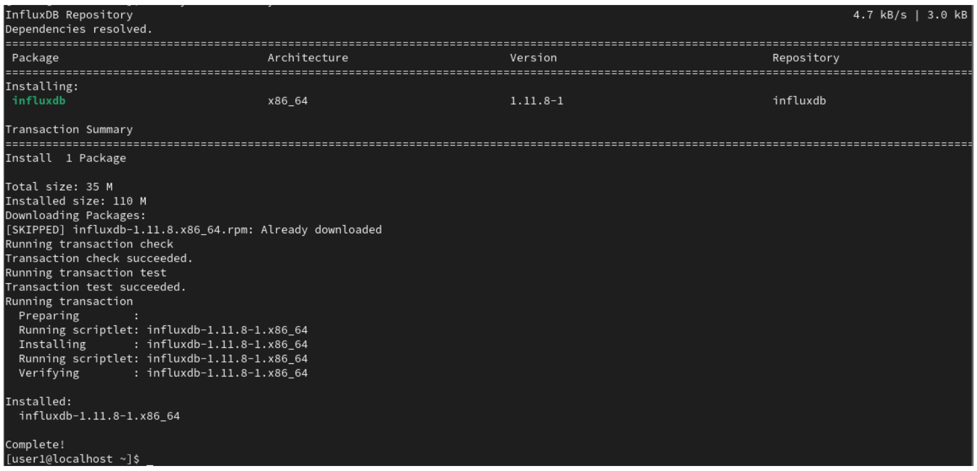
```
[user1@localhost ~]$ cat /etc/yum.repos.d/influxdb.repo
[influxdb]
name=InfluxDB Repository
baseurl=https://repos.influxdata.com/centos/9/x86_64/stable
enabled=1
gpgcheck=0
gpgkey=https://repos.influxdata.com/influxdb.key
[user1@localhost ~]$
```

c) Step 3: Install InfluxDB.

Command: sudo yum install -y influxdb



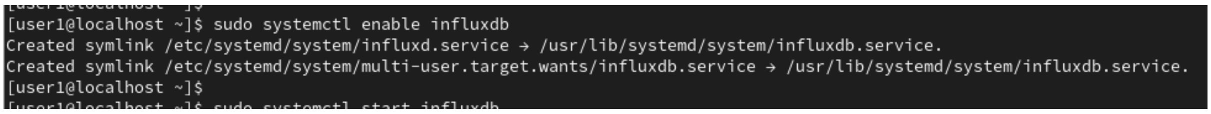
```
[user1@localhost ~]$ sudo yum install -y influxdb
InfluxDB Repository
Dependencies resolved.
```



```
InfluxDB Repository                                                                                          4.7 kB/s | 3.0 kB
Dependencies resolved.
=================================================================================================================================
 Package                              Architecture                     Version                           Repository
=================================================================================================================================
Installing:
 influxdb                             x86_64                           1.11.8-1                          influxdb

Transaction Summary
=================================================================================================================================
Install  1 Package

Total size: 35 M
Installed size: 110 M
Downloading Packages:
[SKIPPED] influxdb-1.11.8.x86_64.rpm: Already downloaded
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Running scriptlet: influxdb-1.11.8-1.x86_64
  Installing       : influxdb-1.11.8-1.x86_64
  Running scriptlet: influxdb-1.11.8-1.x86_64
  Verifying        : influxdb-1.11.8-1.x86_64

Installed:
  influxdb-1.11.8-1.x86_64

Complete!
[user1@localhost ~]$ _
```

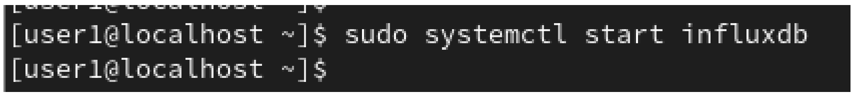
d) Step 4: Once the installation of InfluxDB has been carried out successfully, it is required to be enabled and start the process.

Command: sudo systemctl enable influxdb



```
[user1@localhost ~]$ sudo systemctl enable influxdb
Created symlink /etc/systemd/system/influxd.service → /usr/lib/systemd/system/influxdb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/influxdb.service → /usr/lib/systemd/system/influxdb.service.
[user1@localhost ~]$
```

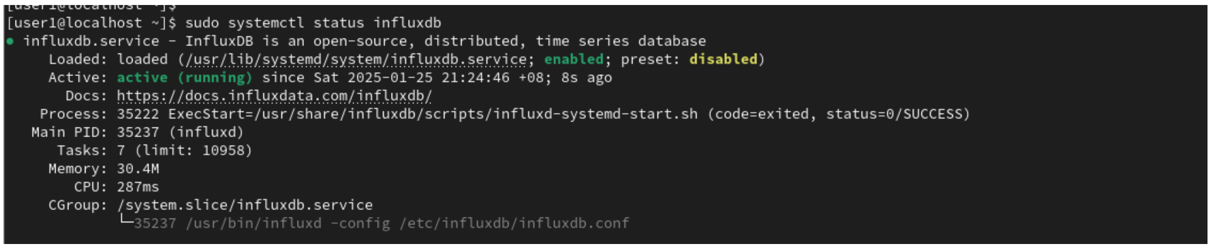
Command: sudo systemctl start influxdb



```
[user1@localhost ~]$ sudo systemctl start influxdb
[user1@localhost ~]$
```

Command: sudo systemctl status influxdb

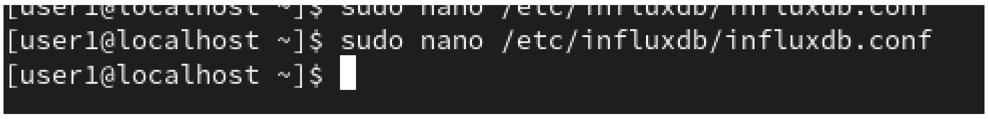
Output: InfluxDB is on an active state.



```
[user1@localhost ~]$ sudo systemctl status influxdb
● influxdb.service - InfluxDB is an open-source, distributed, time series database
     Loaded: loaded (/usr/lib/systemd/system/influxdb.service; enabled; preset: disabled)
     Active: active (running) since Sat 2025-01-25 21:24:46 +08; 8s ago
       Docs: https://docs.influxdata.com/influxdb/
    Process: 35222 ExecStart=/usr/share/influxdb/scripts/influxd-systemd-start.sh (code=exited, status=0/SUCCESS)
   Main PID: 35237 (influxd)
      Tasks: 7 (limit: 10958)
     Memory: 30.4M
        CPU: 287ms
     CGroup: /system.slice/influxdb.service
             └─35237 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```

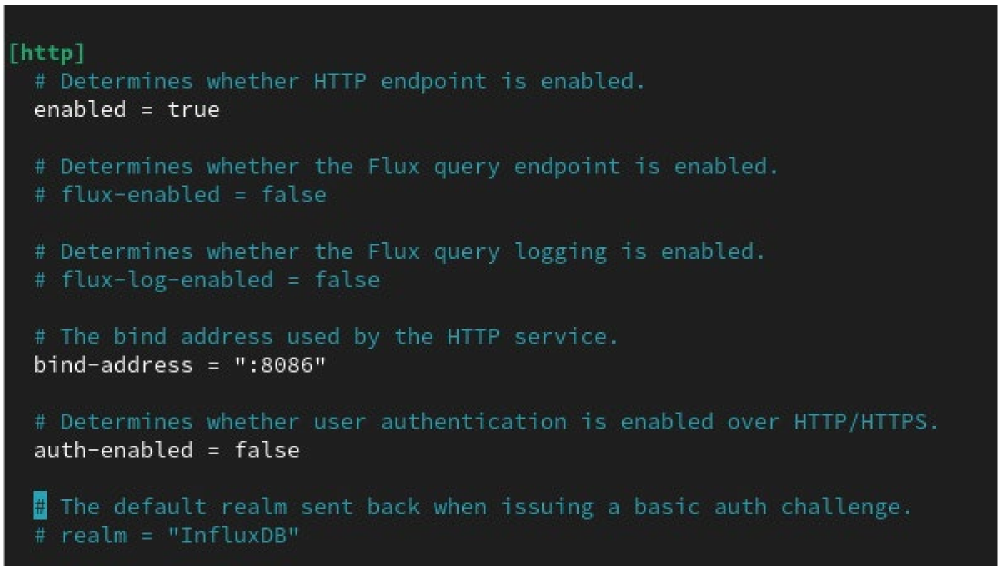
e) Step 5: Set up InfluxDB to receive data.

Command: sudo nano /etc/influxdb/influxdb.conf – to open the configuration file.



```
[user1@localhost ~]$ sudo nano /etc/influxdb/influxdb.conf
[user1@localhost ~]$ █
```

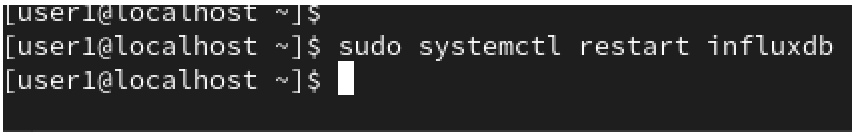
At the influxdb.conf (influxdb configuration file), ensure that http is set to enabled which will allows to accept incoming data. The default InflxDB port address is ":8086" which will be set under the bind-address and lastly disable the “auth-enabled” (disables authentication) to simplify the testing of this project.



```
[http]
  # Determines whether HTTP endpoint is enabled.
  enabled = true

  # Determines whether the Flux query endpoint is enabled.
  # flux-enabled = false

  # Determines whether the Flux query logging is enabled.
  # flux-log-enabled = false

  # The bind address used by the HTTP service.
  bind-address = ":8086"

  # Determines whether user authentication is enabled over HTTP/HTTPS.
  auth-enabled = false

  # The default realm sent back when issuing a basic auth challenge.
  # realm = "InfluxDB"
```

f) Step 6: Once the InfluxDB configuration file has been set, restart the influxdb.

Command: sudo systemctl restart influxdb



```
[user1@localhost ~]$
[user1@localhost ~]$ sudo systemctl restart influxdb
[user1@localhost ~]$ 
```

3. Installation and Configuration of Telegraf

a) Step 1: Add the telegraf repository.

Command:

sudo tee /etc/yum.repos.d/telegraf.repo <<EOF

[telegraf] - repository named “telegraf”

name=Telegraf Repository – repository label

baseurl=https://repos.influxdata.com/centos/9/x86_64/stable - URL of the repository package located

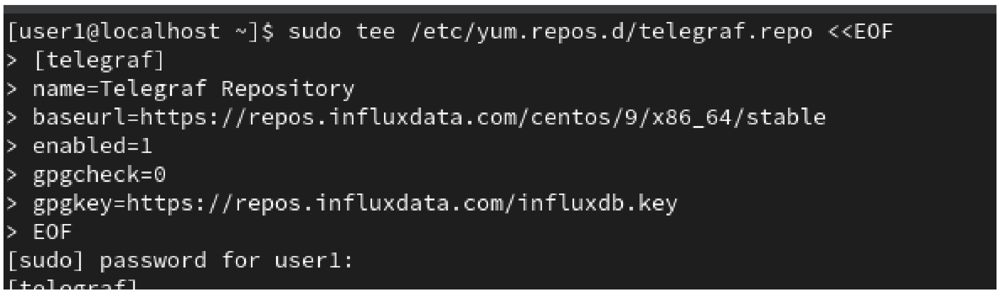
enabled=1 - makes this repository usable with yum

gpgcheck=0 – disable the verification of the package signature prior to installation

gpgkey=https://repos.influxdata.com/influxdb.key - URL to retrieve the GPG key for signature verification

EOF

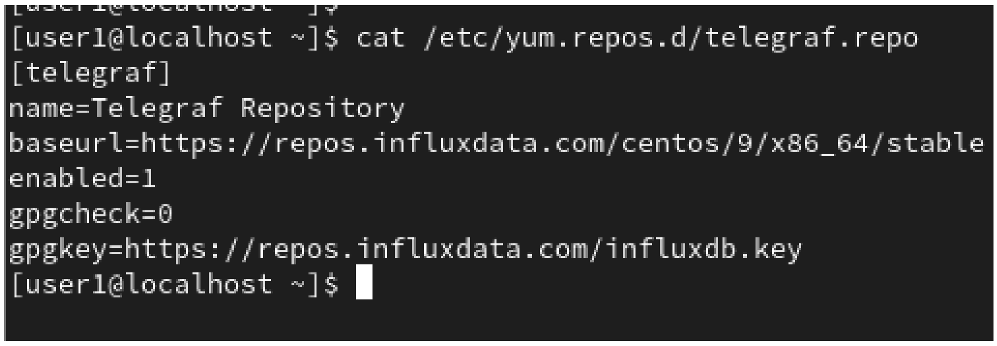
Function: Enables Telegraf to be installed and managed using yum.



```
[user1@localhost ~]$ sudo tee /etc/yum.repos.d/telegraf.repo <<EOF
> [telegraf]
> name=Telegraf Repository
> baseurl=https://repos.influxdata.com/centos/9/x86_64/stable
> enabled=1
> gpgcheck=0
> gpgkey=https://repos.influxdata.com/influxdb.key
> EOF
[sudo] password for user1:
```

b) Step 2: Reconfirm the creation of CentOS repository configuration file for Telegraf.

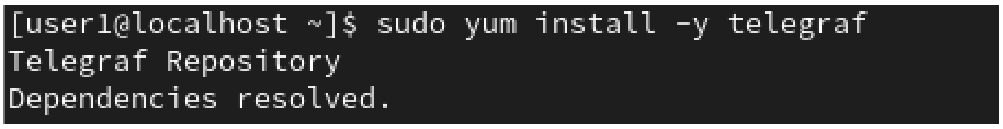
Command: cat /etc/yum.repos.d/telegraf.repo



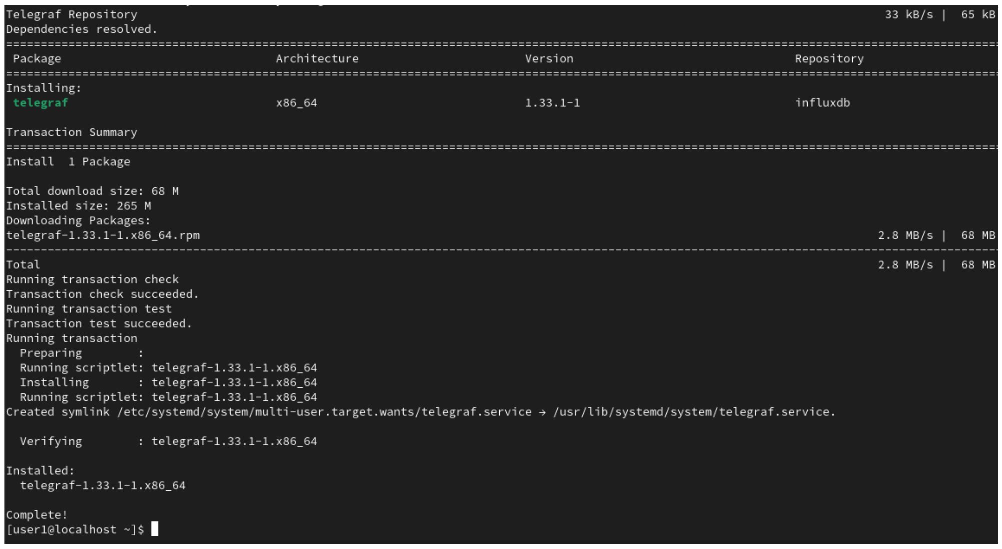
```
[user1@localhost ~]$ cat /etc/yum.repos.d/telegraf.repo
[telegraf]
name=Telegraf Repository
baseurl=https://repos.influxdata.com/centos/9/x86_64/stable
enabled=1
gpgcheck=0
gpgkey=https://repos.influxdata.com/influxdb.key
[user1@localhost ~]$
```

c) Step 3: Install Telegraf

Command: sudo yum install -y telegraf



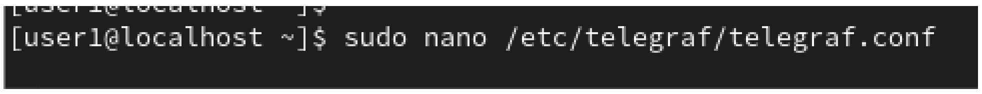
```
[user1@localhost ~]$ sudo yum install -y telegraf
Telegraf Repository
Dependencies resolved.
```



```
Telegraf Repository                                                                                          33 kB/s |  65 kB
Dependencies resolved.
=============================================================================================================================
 Package                             Architecture                     Version                          Repository
=============================================================================================================================
Installing:
 telegraf                            x86_64                           1.33.1-1                         influxdb

Transaction Summary
=============================================================================================================================
Install  1 Package

Total download size: 68 M
Installed size: 265 M
Downloading Packages:
telegraf-1.33.1-1.x86_64.rpm                                                                                 2.8 MB/s |  68 MB
-----------------------------------------------------------------------------------------------------------------------------
Total                                                                                                        2.8 MB/s |  68 MB
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Running scriptlet: telegraf-1.33.1-1.x86_64
  Installing       : telegraf-1.33.1-1.x86_64
  Running scriptlet: telegraf-1.33.1-1.x86_64
Created symlink /etc/systemd/system/multi-user.target.wants/telegraf.service → /usr/lib/systemd/system/telegraf.service.

  Verifying        : telegraf-1.33.1-1.x86_64

Installed:
  telegraf-1.33.1-1.x86_64

Complete!
[user1@localhost ~]$
```

d) Step 4: Set up the Telegraf configuration file with the appropriate output and input plugins to collect metrics.

Command: sudo nano /etc/telegraf/telegraf.conf



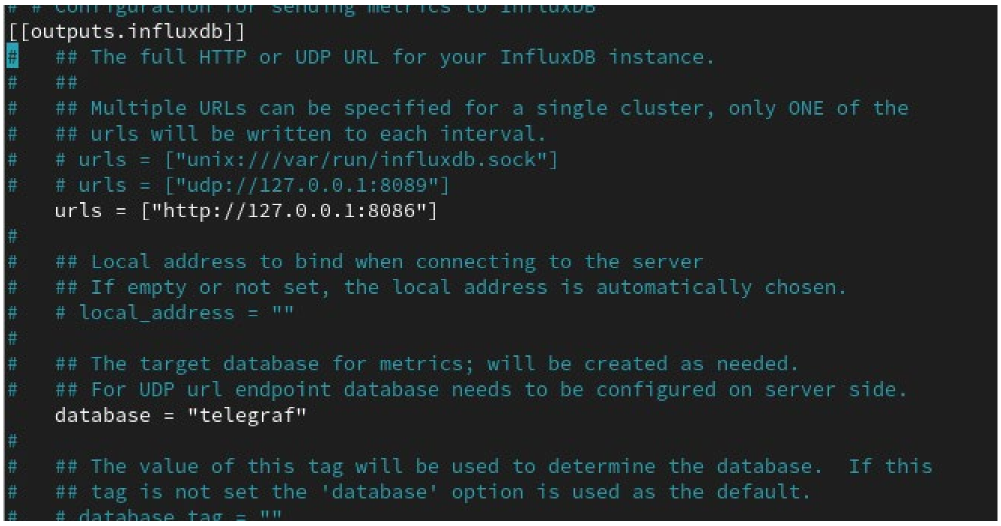
```
[user1@localhost ~]$ sudo nano /etc/telegraf/telegraf.conf
```

Set up the output plugins for Telegraf to send the collected metrics as per the details below:

[[outputs.influxdb]] – output of the Telegraf

urls = ["http://127.0.0.1:8086"] – output url of the InfluxDB running

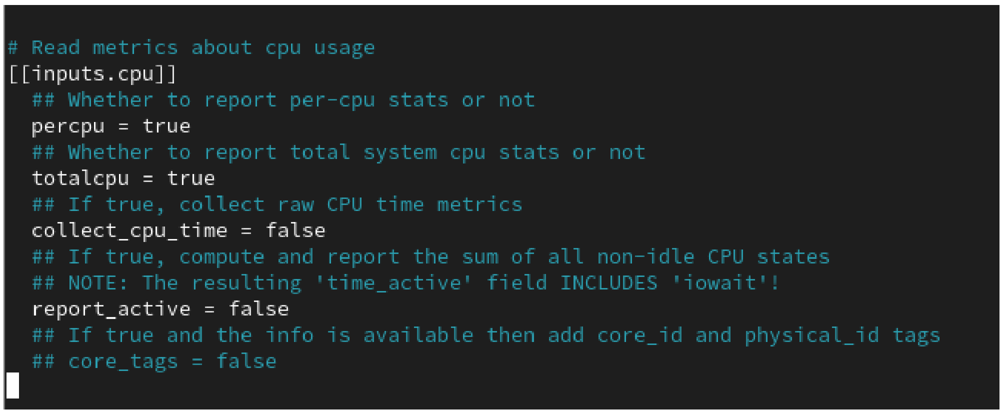
database = "telegraf"



```
# # Configuration for sending metrics to InfluxDB
[[outputs.influxdb]]
#   ## The full HTTP or UDP URL for your InfluxDB instance.
#   ##
#   ## Multiple URLs can be specified for a single cluster, only ONE of the
#   ## urls will be written to each interval.
#   # urls = ["unix:///var/run/influxdb.sock"]
#   # urls = ["udp://127.0.0.1:8089"]
    urls = ["http://127.0.0.1:8086"]
#
#   ## Local address to bind when connecting to the server
#   ## If empty or not set, the local address is automatically chosen.
#   # local_address = ""
#
#   ## The target database for metrics; will be created as needed.
#   ## For UDP url endpoint database needs to be configured on server side.
    database = "telegraf"
#
#   ## The value of this tag will be used to determine the database.  If this
#   ## tag is not set the 'database' option is used as the default.
#   # database_tag = ""
```

Input plugin 1: Configuration of CPU usage metrics to be collected – [[inputs.cpu]]

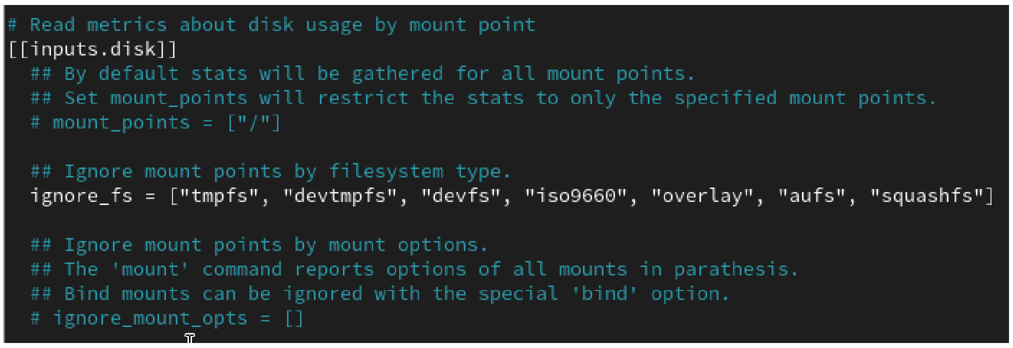
<u>Function</u>: It will collect metrics for each individual CPU core, overall CPU usage for all cores, disable the collection of CPU time ("true" if needed finer details for diagnosing and troubleshooting performance), and report_active to false.



```
# Read metrics about cpu usage
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states
  ## NOTE: The resulting 'time_active' field INCLUDES 'iowait'!
  report_active = false
  ## If true and the info is available then add core_id and physical_id tags
  ## core_tags = false
```

Input plugin 2: Configuration of Disk usage metrics to be collected – [[inputs.disk]]

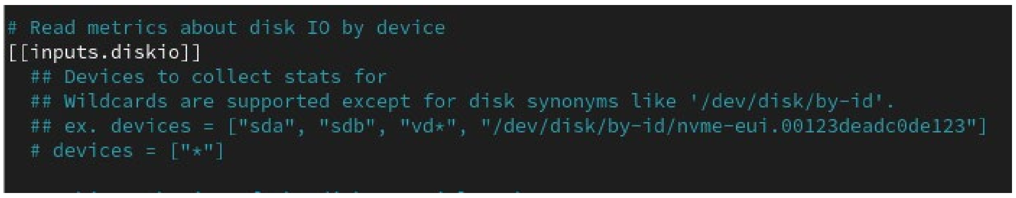
<u>Function</u>: This metrics is responsible for gathering data on disk utilization from all mounted file system which is essential for making sure systems have sufficient storage and aid in quick identifying of disk-related problems.



```
# Read metrics about disk usage by mount point
[[inputs.disk]]
  ## By default stats will be gathered for all mount points.
  ## Set mount_points will restrict the stats to only the specified mount points.
  # mount_points = ["/"]

  ## Ignore mount points by filesystem type.
  ignore_fs = ["tmpfs", "devtmpfs", "devfs", "iso9660", "overlay", "aufs", "squashfs"]

  ## Ignore mount points by mount options.
  ## The 'mount' command reports options of all mounts in parathesis.
  ## Bind mounts can be ignored with the special 'bind' option.
  # ignore_mount_opts = []
```

Input plugin 3: Configuration of Disk IO metrics to be collected – [[inputs.diskio]]

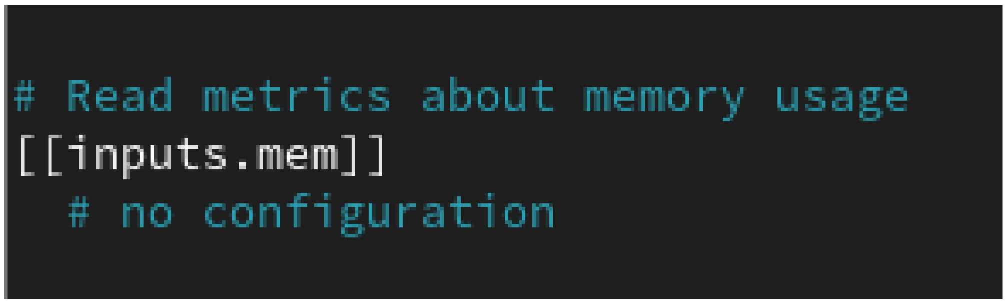
Function: The Disk I/O configuration is responsible for gathering disk input/output (I/O) statistics. This will aid in disk performance, locating issues, and improving system storage. At Grafana, it will display the graph at the speed the data read and write on the disk.



```
# Read metrics about disk IO by device
[[inputs.diskio]]
  ## Devices to collect stats for
  ## Wildcards are supported except for disk synonyms like '/dev/disk/by-id'.
  ## ex. devices = ["sda", "sdb", "vd*", "/dev/disk/by-id/nvme-eui.00123deadc0de123"]
  # devices = ["*"]
```

Input plugin 4: Configuration of Memory usage metrics to be collected – [[inputs.mem]].

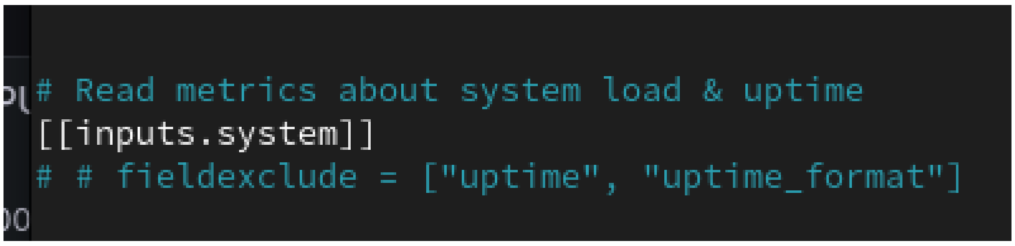
Function: To collect metrics of the memory usage and display in Grafana as "Total Memory", "Available Memory", and "Total Memory Used".



```
# Read metrics about memory usage
[[inputs.mem]]
  # no configuration
```

Input plugin 5: Configuration of Load Average metrics to be collected – [[inputs.system]].

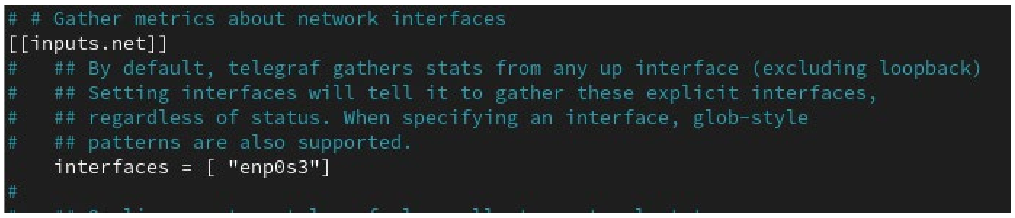
Function: To read the metrics on system load and uptime. Grafana will display the system load time on the average number of processes waiting to be executed in the last 1 minute, 5 minutes, and 15 minutes. This will assist in tracking CPU load and identifying bottlenecks.



```
# Read metrics about system load & uptime
[[inputs.system]]
# # fieldexclude = ["uptime", "uptime_format"]
```

Input plugin 6: Configuration of Network Utilization metrics to be collected – [[inputs.net]].

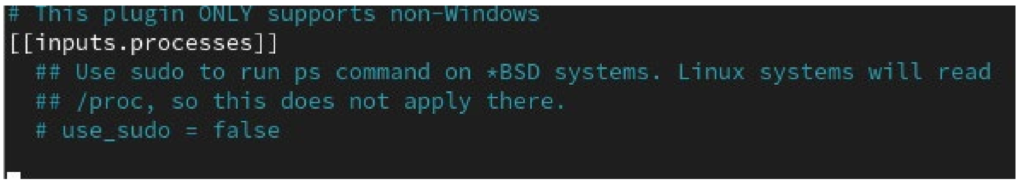
Function: This metrics will be collecting network-related metrics from the system's network interfaces, which is "enp0s3", for the purpose of network health and traffic monitoring. Grafana will display the total amount of data sent and received as well as the total number of packets sent and received.



```
# # Gather metrics about network interfaces
[[inputs.net]]
#   ## By default, telegraf gathers stats from any up interface (excluding loopback)
#   ## Setting interfaces will tell it to gather these explicit interfaces,
#   ## regardless of status. When specifying an interface, glob-style
#   ## patterns are also supported.
    interfaces = [ "enp0s3"]
#
```

Input plugin 7: Configuration of Process Monitoring metrics to be collected – [[inputs.processes]].

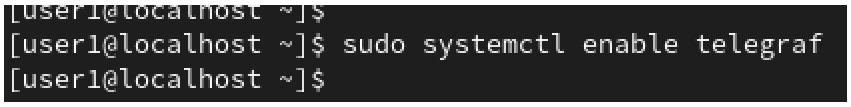
Function: The metrics will monitor process-related tasks on a system as in collecting number of processes running, sleeping, blocked, and zombies which is useful for monitoring system health.



```
# This plugin ONLY supports non-Windows
[[inputs.processes]]
  ## Use sudo to run ps command on *BSD systems. Linux systems will read
  ## /proc, so this does not apply there.
  # use_sudo = false
```

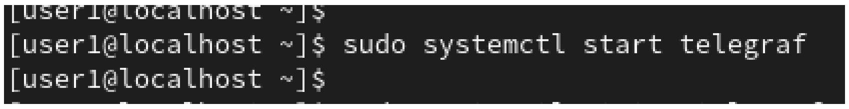
e) Step 5: Once the installation and configuration of Telegraf has been carried out successfully, it is required to be enabled and start the process.

Command: sudo systemctl enable telegraf



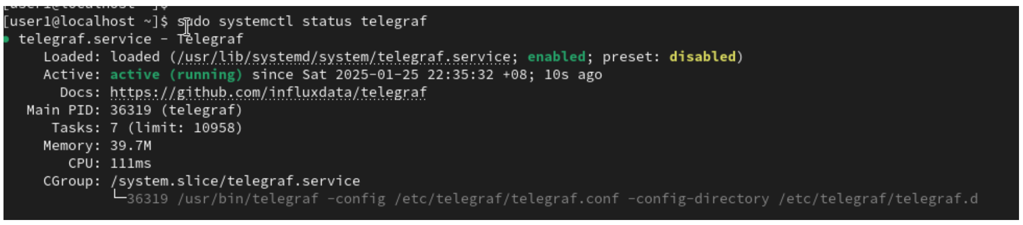
```
[user1@localhost ~]$
[user1@localhost ~]$ sudo systemctl enable telegraf
[user1@localhost ~]$
```

Command: sudo systemctl start telegraf



```
[user1@localhost ~]$
[user1@localhost ~]$ sudo systemctl start telegraf
[user1@localhost ~]$
```

Command: sudo systemctl status telegraf



```
[user1@localhost ~]$ sudo systemctl status telegraf
● telegraf.service - Telegraf
     Loaded: loaded (/usr/lib/systemd/system/telegraf.service; enabled; preset: disabled)
     Active: active (running) since Sat 2025-01-25 22:35:32 +08; 10s ago
       Docs: https://github.com/influxdata/telegraf
   Main PID: 36319 (telegraf)
      Tasks: 7 (limit: 10958)
     Memory: 39.7M
        CPU: 111ms
     CGroup: /system.slice/telegraf.service
             └─36319 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.d
```

4. Installation and Configuration of Grafana

a) Step 1: Add the grafana repository.

Command: sudo tee /etc/yum.repos.d/grafana.repo <<EOF

[grafana] – repository named “grafana”

name=Grafana Repository – repository label

baseurl=https://rpm.grafana.com/oss/rpm – URL of the repository package located

repo_gpgcheck=0 - disable the verification of the repository metadata

enabled=1 – make this repository usable with yum

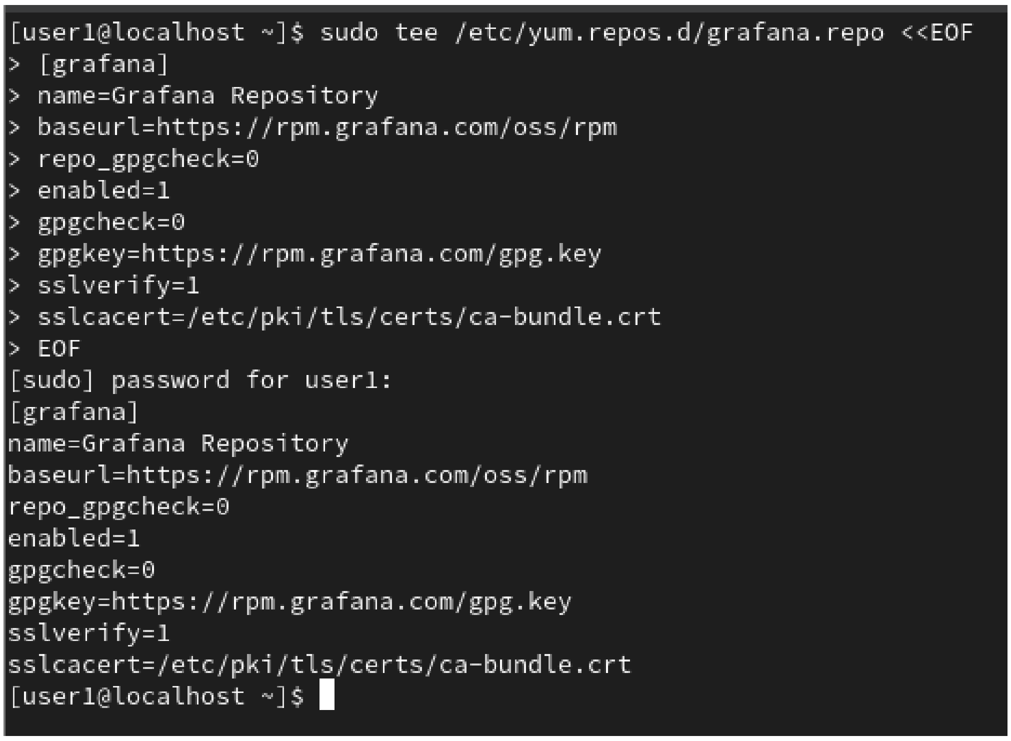
gpgcheck=0 – disable the verification of the package signature prior to installation

gpgkey=https://rpm.grafana.com/gpg.key – URL to retrieve the GPG key for signature verification

sslverify=1 – enable verification of SSL/TLS when downloading packages

sslcacert=/etc/pki/tls/certs/ca-bundle.crt – location of the SSL certificate

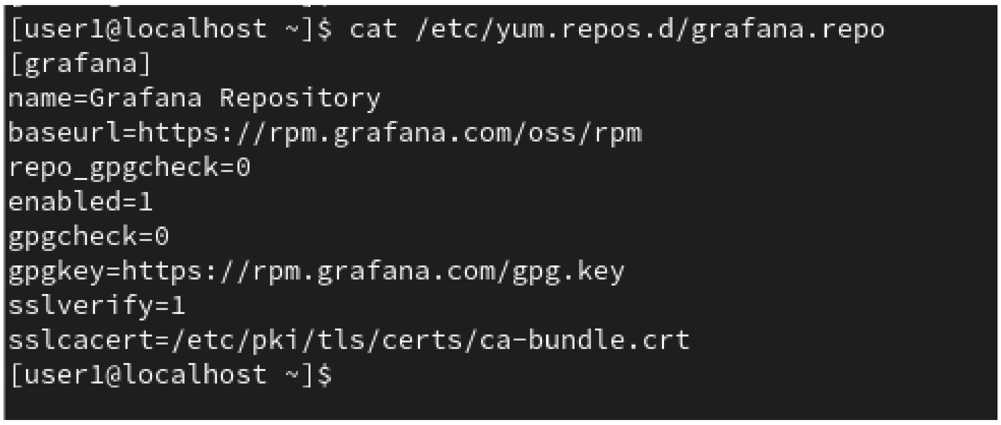
EOF



```
[user1@localhost ~]$ sudo tee /etc/yum.repos.d/grafana.repo <<EOF
> [grafana]
> name=Grafana Repository
> baseurl=https://rpm.grafana.com/oss/rpm
> repo_gpgcheck=0
> enabled=1
> gpgcheck=0
> gpgkey=https://rpm.grafana.com/gpg.key
> sslverify=1
> sslcacert=/etc/pki/tls/certs/ca-bundle.crt
> EOF
[sudo] password for user1:
[grafana]
name=Grafana Repository
baseurl=https://rpm.grafana.com/oss/rpm
repo_gpgcheck=0
enabled=1
gpgcheck=0
gpgkey=https://rpm.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
[user1@localhost ~]$ 
```

b) Step 2: Reconfirm the creation of CentOS repository configuration file for Grafana.

Command: cat /etc/yum.repos.d/grafana.repo


```
[user1@localhost ~]$ cat /etc/yum.repos.d/grafana.repo
[grafana]
name=Grafana Repository
baseurl=https://rpm.grafana.com/oss/rpm
repo_gpgcheck=0
enabled=1
gpgcheck=0
gpgkey=https://rpm.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
[user1@localhost ~]$
```

c) Step 3: Solution to the error of Grafana installation.

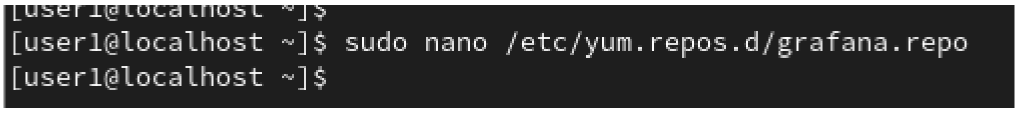
Previous base URL: https://rpm.grafana.com/oss/rpm

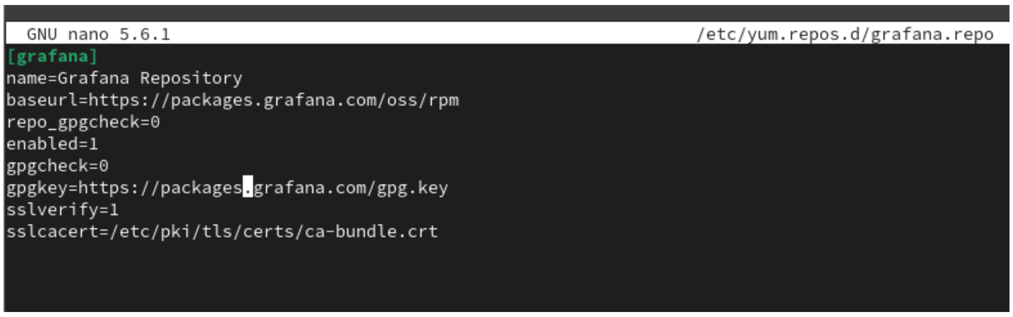
New base URL: https://packages.grafana.com/oss/rpm

Previous GPG Key URL: https://rpm.grafana.com/gpg.key

New GPG Key URL: https://packages.grafana.com/gpg.key

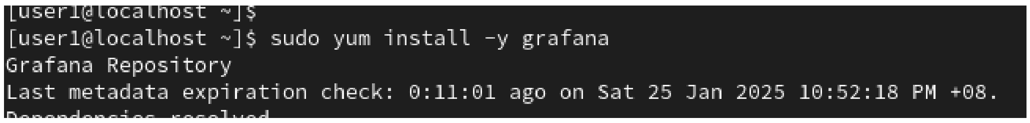
Reason: The usage of rpm in the URL has been deprecated, hence, updating it ensures that the system points to the correct repository and able to perform the Grafana installation.

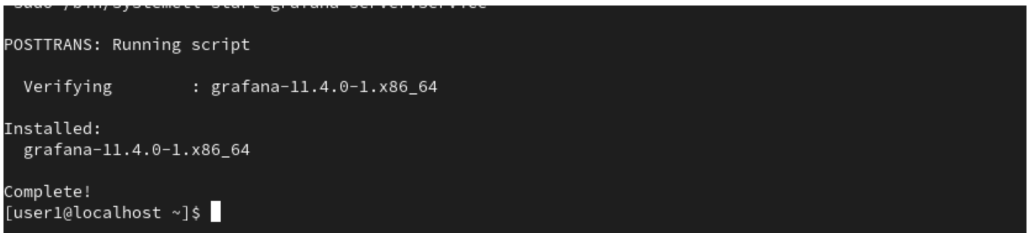

```
[user1@localhost ~]$
[user1@localhost ~]$ sudo nano /etc/yum.repos.d/grafana.repo
[user1@localhost ~]$
```


```
 GNU nano 5.6.1                                         /etc/yum.repos.d/grafana.repo
[grafana]
name=Grafana Repository
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=0
enabled=1
gpgcheck=0
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

d) Step 4: Installation of Grafana.

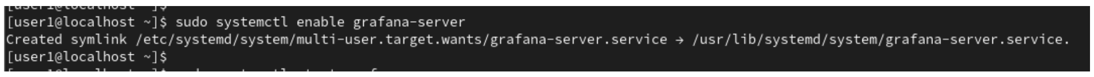
Command: sudo yum install -y grafana


```
[user1@localhost ~]$
[user1@localhost ~]$ sudo yum install -y grafana
Grafana Repository
Last metadata expiration check: 0:11:01 ago on Sat 25 Jan 2025 10:52:18 PM +08.
Dependencies resolved.
```


```
sudo /bin/systemctl start grafana-server.service

POSTTRANS: Running script

  Verifying        : grafana-11.4.0-1.x86_64

Installed:
  grafana-11.4.0-1.x86_64

Complete!
[user1@localhost ~]$
```

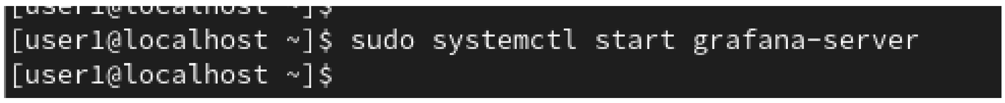
e) Step 5: Once the installation and configuration of Grafana has been carried out successfully, it is required to be enabled and start the process.

Command: sudo systemctl enable grafana-server



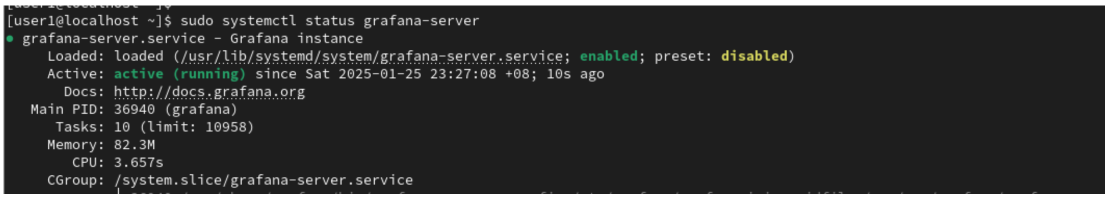
```
[user1@localhost ~]$ sudo systemctl enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /usr/lib/systemd/system/grafana-server.service.
[user1@localhost ~]$
```

Command: sudo systemctl start grafana-server



```
[user1@localhost ~]$ sudo systemctl start grafana-server
[user1@localhost ~]$
```

Command: sudo systemctl status grafana-server

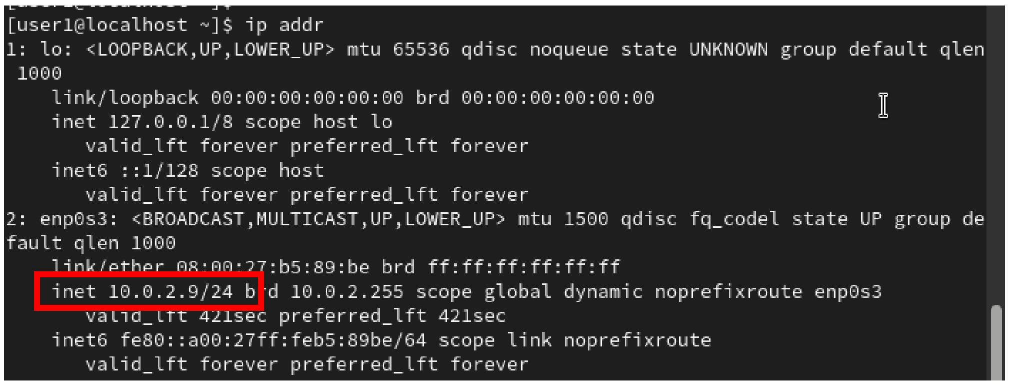


```
[user1@localhost ~]$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: disabled)
     Active: active (running) since Sat 2025-01-25 23:27:08 +08; 10s ago
       Docs: http://docs.grafana.org
   Main PID: 36940 (grafana)
      Tasks: 10 (limit: 10958)
     Memory: 82.3M
        CPU: 3.657s
     CGroup: /system.slice/grafana-server.service
```

f) Step 6: Access Grafana via a web browser and navigate to:

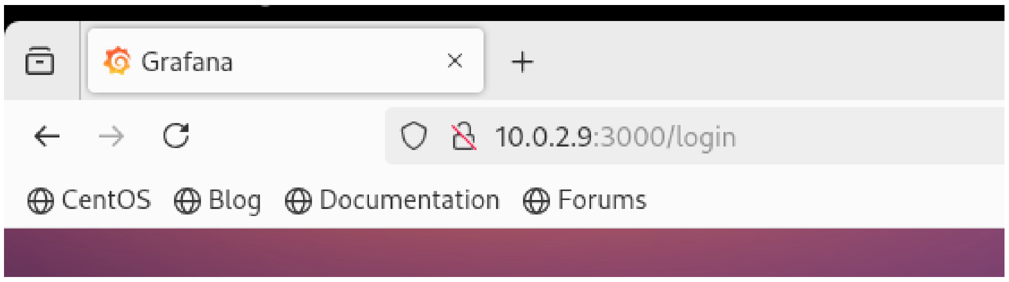
http://<ip_server>:3000

IP Server: 10.0.2.9



```
[user1@localhost ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group de
fault qlen 1000
    link/ether 08:00:27:b5:89:be brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.9/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 421sec preferred_lft 421sec
    inet6 fe80::a00:27ff:feb5:89be/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

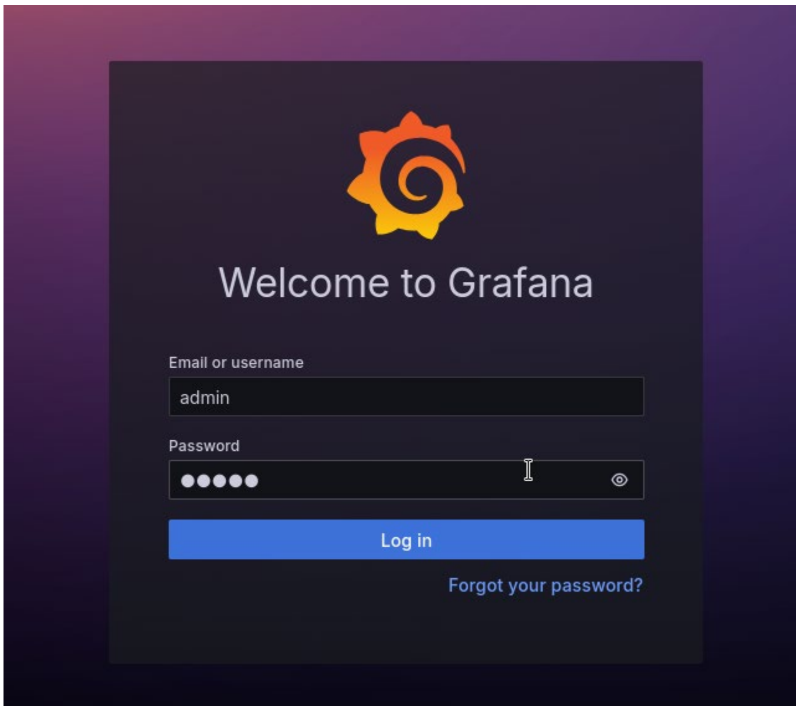
Access http://10.0.2.9:3000



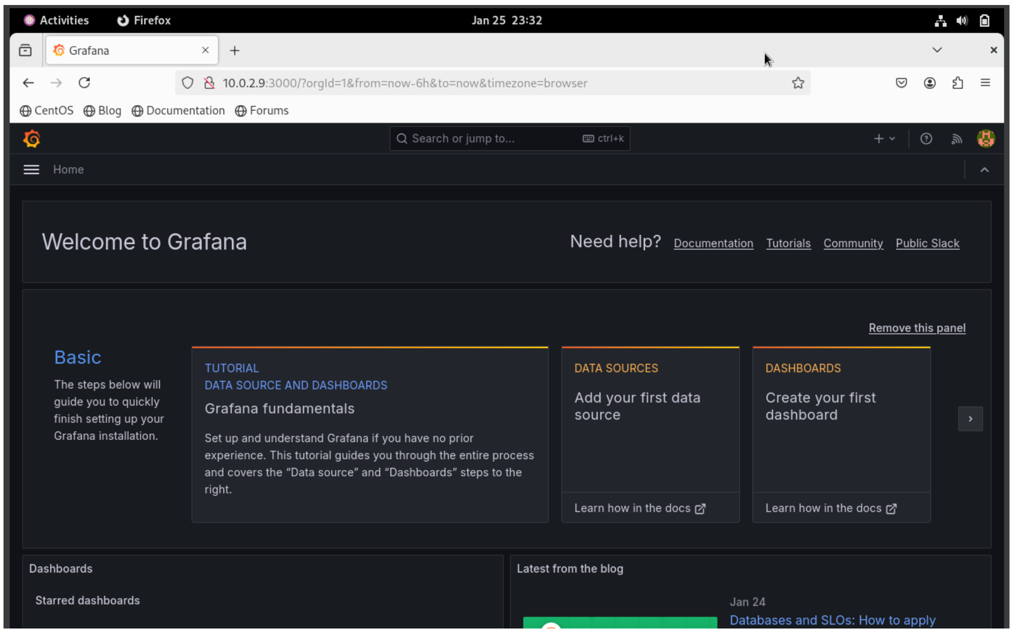
Login Grafana using the default login below:

Username: admin

Password: admin

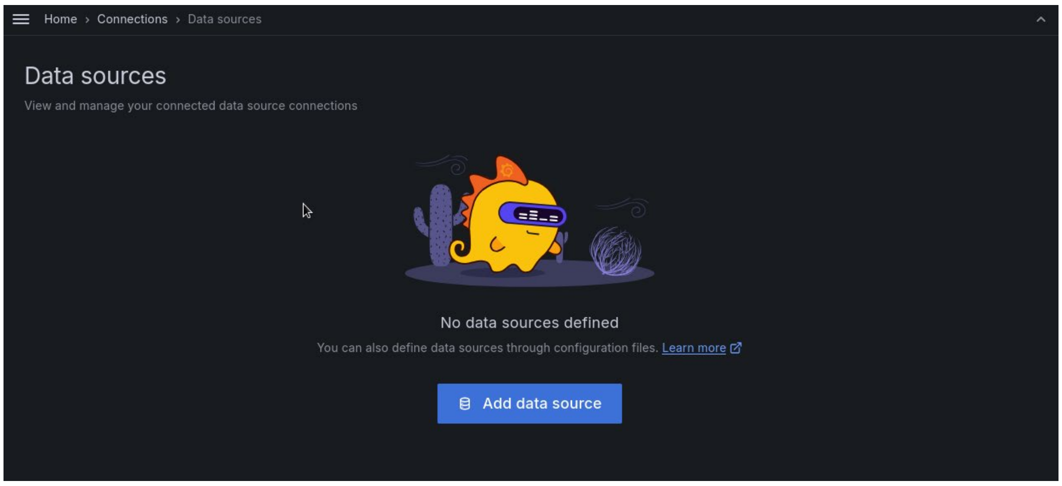


Landing of Grafana Home Screen.


Activities
Firefox
Jan 25 23:32
Grafana
10.0.2.9:3000/?orgId=1&from=now-6h&to=now&timezone=browser
CentOS
Blog
Documentation
Forums
Search or jump to...
ctrl+k
Home
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.
DATA SOURCES
Add your first data source
Learn how in the docs
DASHBOARDS
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Latest from the blog
Jan 24
Databases and SLOs: How to apply

g) Step 7: Adding and configuring data source – influxdb

Search for "Data Source" at the Grafana home screen and configure as below:



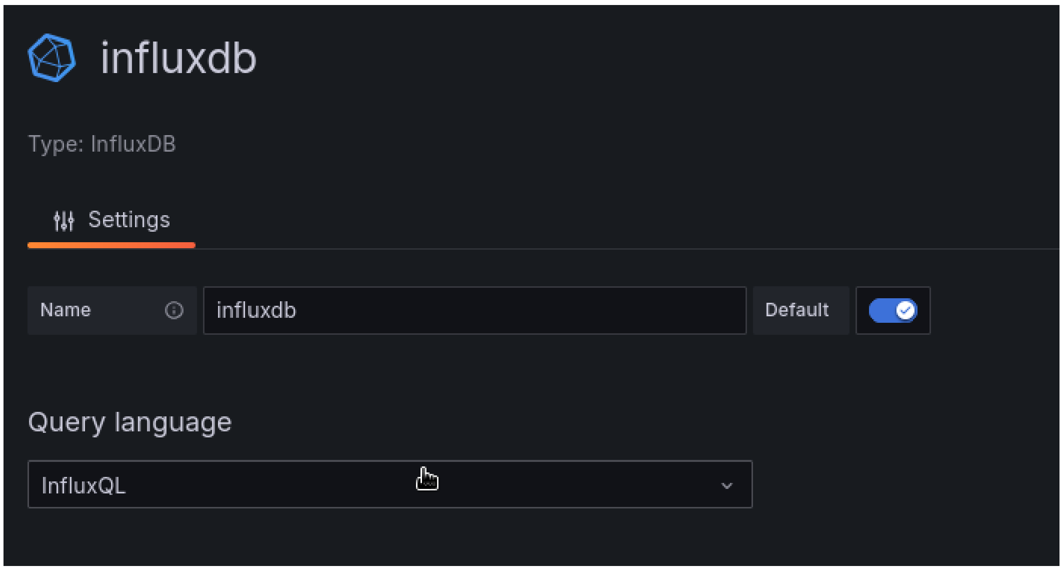
Data sources

View and manage your connected data source connections

No data sources defined

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

Select influxDB to be the data source in order to display the metrics at Grafana.



influxdb

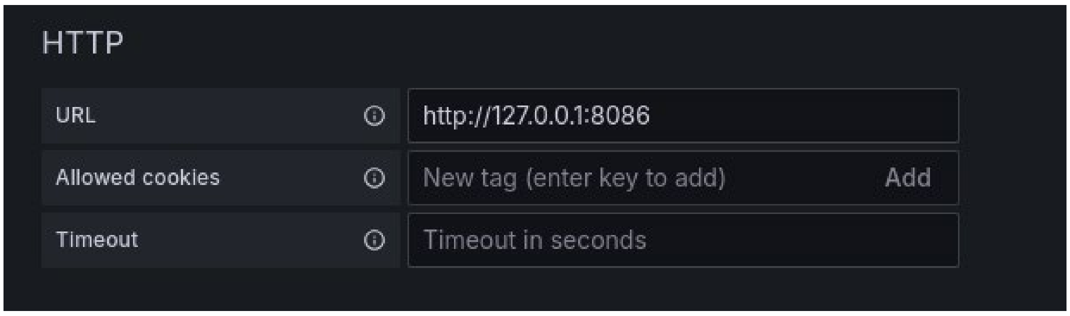
Type: InfluxDB

Name: influxdb   Default: on

Query language: InfluxQL

Enter the details of the configuration of data source as below:

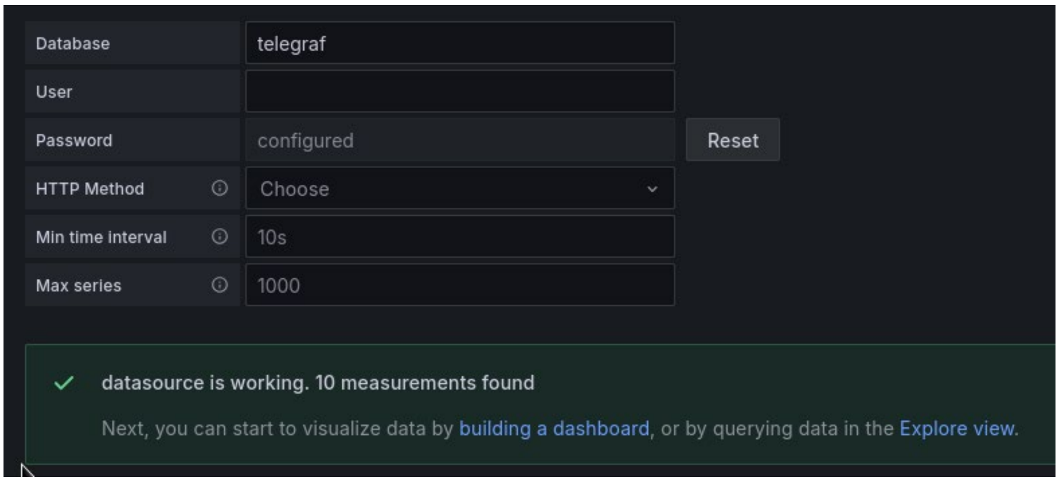
- URL of the influxDB – http://127.0.0.1:8086



HTTP

URL	http://127.0.0.1:8086
Allowed cookies	New tag (enter key to add) Add
Timeout	Timeout in seconds

- Database: telegraf
- Test the configuration of the data source.
- Output: The data source is working as in collecting and receiving metrics.

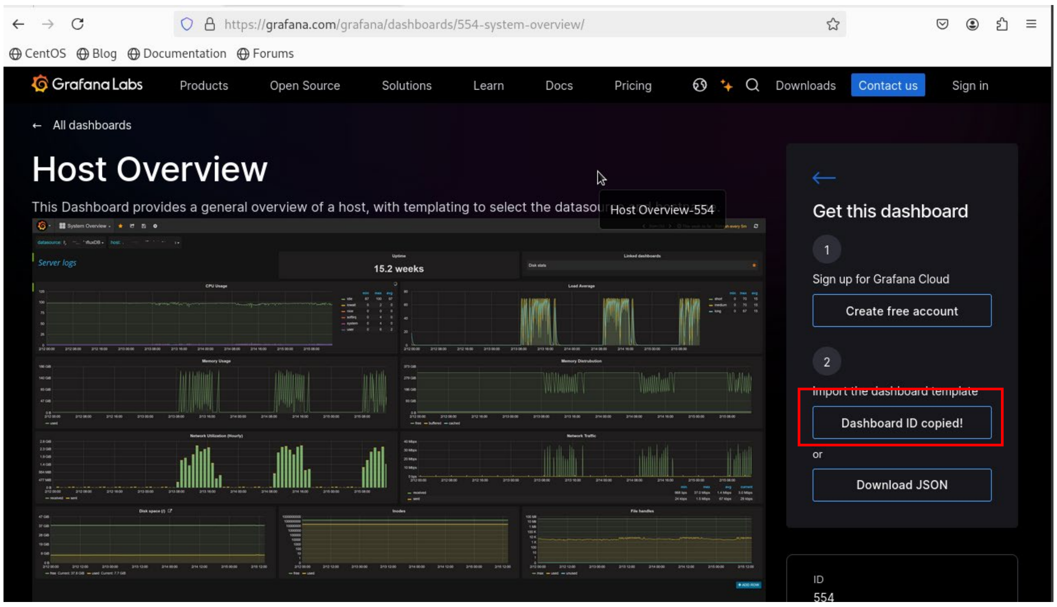


Database	telegraf
User	
Password	configured
HTTP Method	Choose
Min time interval	10s
Max series	1000

datasource is working. 10 measurements found

Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.

Access the website, https://grafana.com/grafana/dashboards/554, to utilize the template as our Grafana Dashboard and copy the "Dashboard ID".

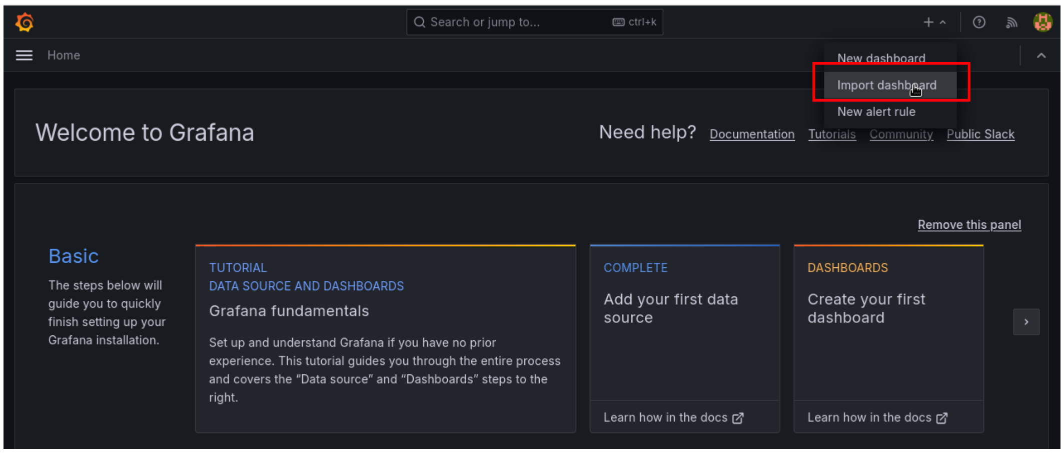


https://grafana.com/grafana/dashboards/554-system-overview/

Host Overview

This Dashboard provides a general overview of a host, with templating to select the datasou

At the Grafana Home Screen, click on the '+' sign at the top right corner and click on "Import dashboard".



Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Basic

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

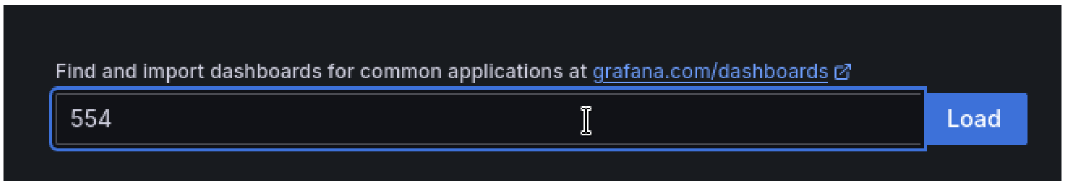
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals

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.

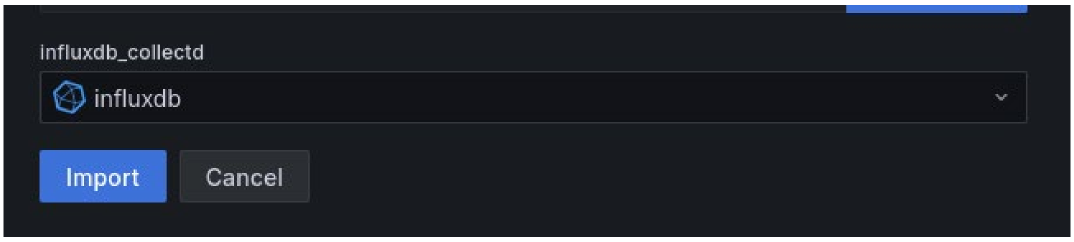
COMPLETE
Add your first data source
Learn how in the docs

DASHBOARDS
Create your first dashboard
Learn how in the docs

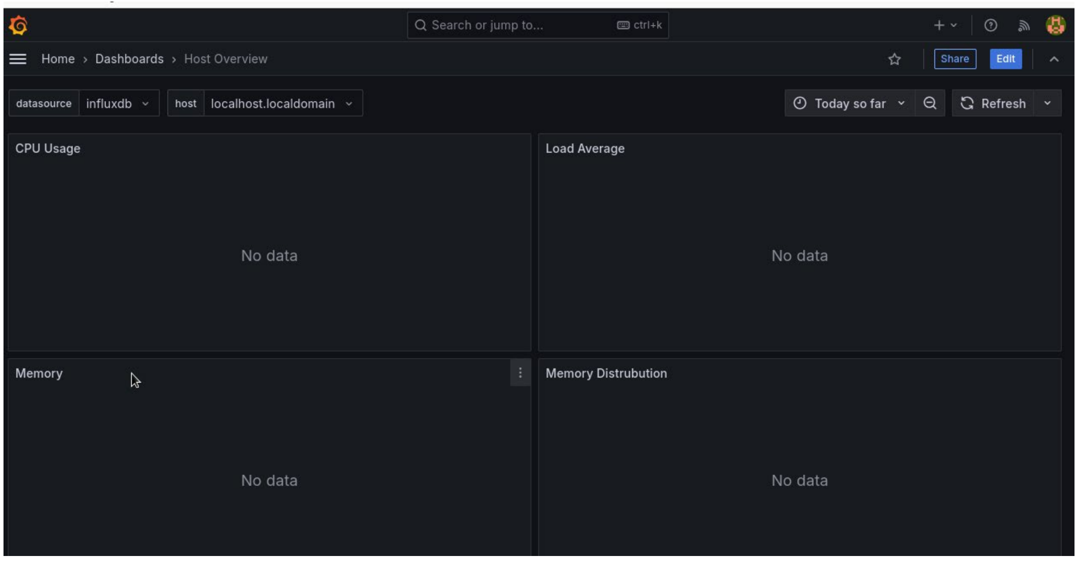
Then, paste the “Dashboard ID” that was copied from the Grafana template (https://grafana.com/grafana/dashboards/554)


Find and import dashboards for common applications at grafana.com/dashboards
554
Load

Lastly, click on “Import” to utilize the Grafana Dashboard.



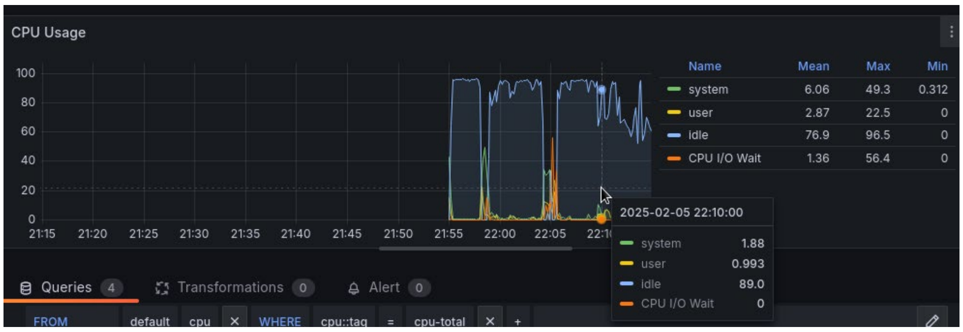
Grafana Dashboard template has been successfully added and displayed.



h) Step 8: Configure the dashboard panel as in adding queries to get the metrics from InfluxDB and display it here.

- **Metrics 1: CPU Usage**
- The image below gives a thorough overview of the Grafana panel for CPU utilization monitoring along with the InfluxDB query setups. In order to evaluate overall CPU performance and identify possible obstacles, the panel displays important CPU metrics such as system usage, user processes, idle CPU time, and I/O wait time.
- The X-axis measures time, and the Y-axis shows CPU utilization as a percentage.

- The importance of monitoring CPU usage are to prevent overloading of the system by user processes or I/O waits, and assist in resource allocation by balancing between user and system usage.

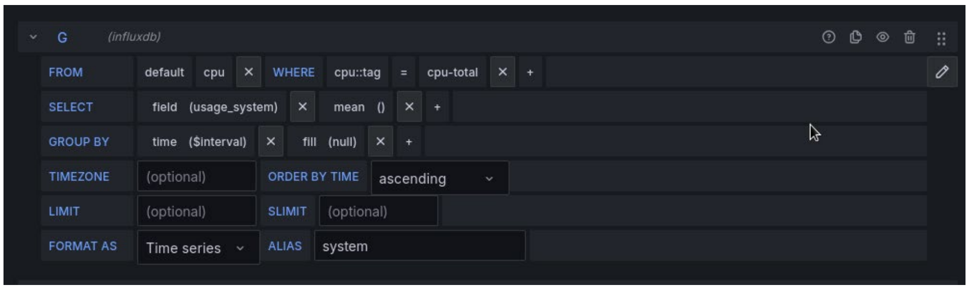


Name	Mean	Max	Min
system	6.06	49.3	0.312
user	2.87	22.5	0
idle	76.9	96.5	0
CPU I/O Wait	1.36	56.4	0

```
2025-02-05 22:10:00
system        1.88
user          0.993
idle          89.0
CPU I/O Wait  0
```

- Query 1: System Usage

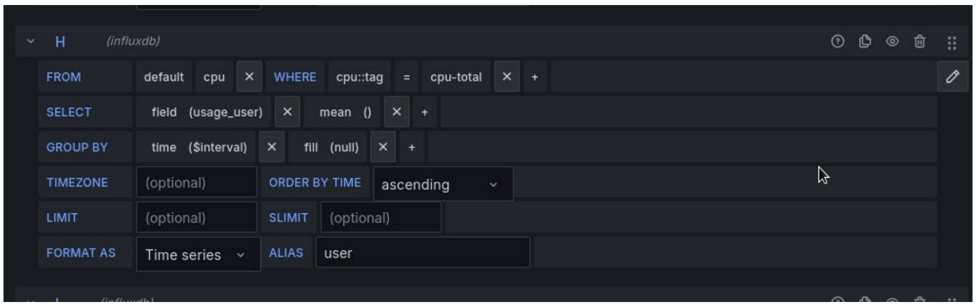
  Keeps track of the proportion of CPU utilization that system-level processes are responsible for.



```
G (influxdb)
FROM       default  cpu  WHERE cpu::tag = cpu-total
SELECT     field (usage_system)  mean ()
GROUP BY   time ($interval)  fill (null)
TIMEZONE   (optional)  ORDER BY TIME  ascending
LIMIT      (optional)  SLIMIT  (optional)
FORMAT AS  Time series  ALIAS  system
```

- Query 2: User Processes

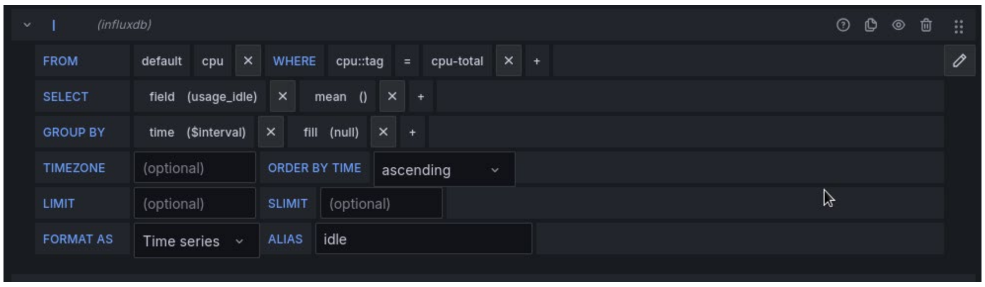
  Monitors the proportion of CPU that user-level processes such as application use.



```
H (influxdb)
FROM       default  cpu  WHERE cpu::tag = cpu-total
SELECT     field (usage_user)  mean ()
GROUP BY   time ($interval)  fill (null)
TIMEZONE   (optional)  ORDER BY TIME  ascending
LIMIT      (optional)  SLIMIT  (optional)
FORMAT AS  Time series  ALIAS  user
```

- Query 3: Idle CPU Time

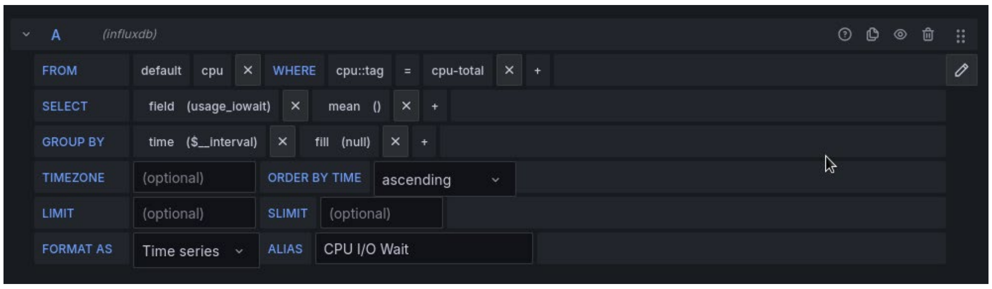
  Tracks the percentage of time the CPU is idle.



```
I (influxdb)
FROM       default  cpu  WHERE cpu::tag = cpu-total
SELECT     field (usage_idle)  mean ()
GROUP BY   time ($interval)  fill (null)
TIMEZONE   (optional)  ORDER BY TIME  ascending
LIMIT      (optional)  SLIMIT  (optional)
FORMAT AS  Time series  ALIAS  idle
```

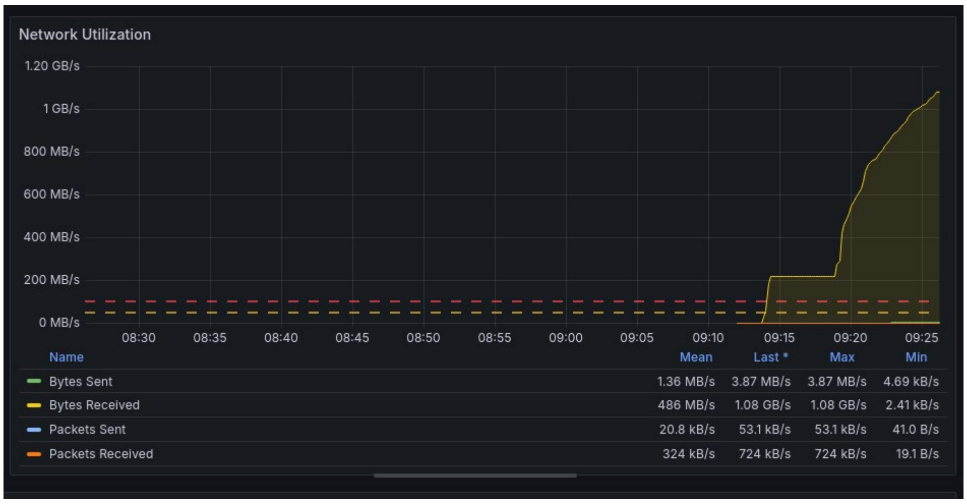
- Query 4: CPU I/O Wait

  Shows the proportion of time spent waiting for I/O activities such as disk or network access by the CPU.



```
A   (influxdb)
FROM        default  cpu  WHERE  cpu::tag  =  cpu-total
SELECT      field (usage_iowait)   mean ()
GROUP BY    time ($__interval)   fill (null)
TIMEZONE    (optional)     ORDER BY TIME  ascending
LIMIT       (optional)     SLIMIT  (optional)
FORMAT AS   Time series    ALIAS  CPU I/O Wait
```

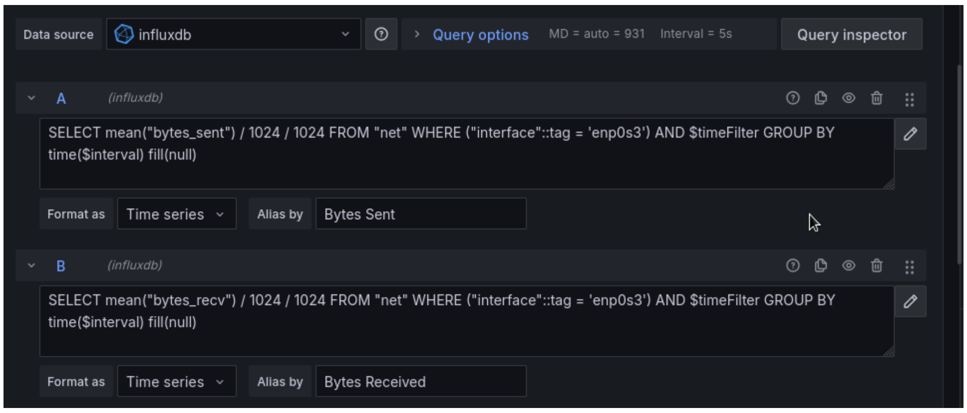
- **Metrics 2: Network Utilization**
- Key metrics including bytes transmitted, bytes received, packets sent, and packets received over a certain time are displayed in the graph. The X-axis shows the time, while the Y-axis shows the data rate in megabytes per second (MB/s), increasing to gigabytes (GB/s) when needed.
- A summary of network performance is provided with the legend beneath the graph, which lists the mean, last, max, and min values for each statistic.
- Based on the graph below, a spike in incoming network traffic is shown by the growing trend in bytes received, which may imply heavy network activity or massive file transfers during that time.
- Network Utilization visual is important in identifying any bottlenecks, detect bandwidth usage, and maintain network stability.



Name	Mean	Last *	Max	Min
🟩 Bytes Sent	1.36 MB/s	3.87 MB/s	3.87 MB/s	4.69 kB/s
🟨 Bytes Received	486 MB/s	1.08 GB/s	1.08 GB/s	2.41 kB/s
🟦 Packets Sent	20.8 kB/s	53.1 kB/s	53.1 kB/s	41.0 B/s
🟧 Packets Received	324 kB/s	724 kB/s	724 kB/s	19.1 B/s

- Query 1: Bytes Sent
- To monitor the total volume of data sent over the network.
- Query 2: Bytes Received

- To monitor the total volume of data received over the network.



Data source: influxdb — Query options MD = auto = 931 Interval = 5s — Query inspector

A (influxdb)

```
SELECT mean("bytes_sent") / 1024 / 1024 FROM "net" WHERE ("interface"::tag = 'enp0s3') AND $timeFilter GROUP BY time($interval) fill(null)
```

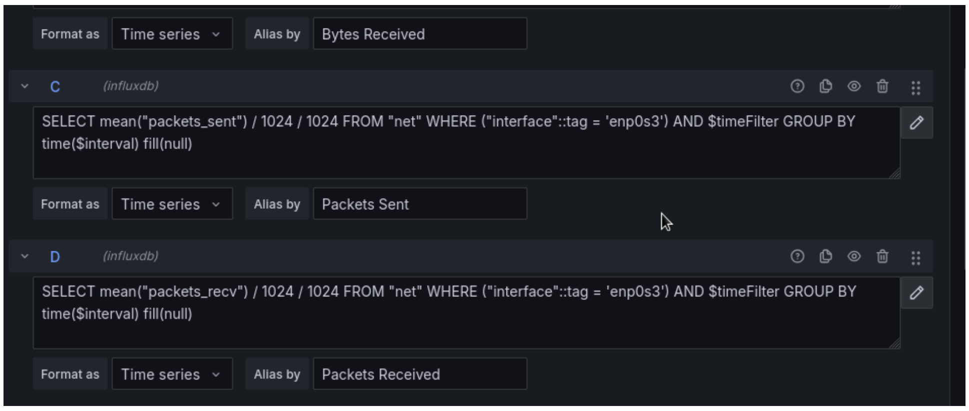
Format as: Time series — Alias by: Bytes Sent

B (influxdb)

```
SELECT mean("bytes_recv") / 1024 / 1024 FROM "net" WHERE ("interface"::tag = 'enp0s3') AND $timeFilter GROUP BY time($interval) fill(null)
```

Format as: Time series — Alias by: Bytes Received

- Query 3: Packets Sent
- Query 4: Packets Received
- Monitor the number of packets handled.



Format as: Time series — Alias by: Bytes Received

C (influxdb)

```
SELECT mean("packets_sent") / 1024 / 1024 FROM "net" WHERE ("interface"::tag = 'enp0s3') AND $timeFilter GROUP BY time($interval) fill(null)
```

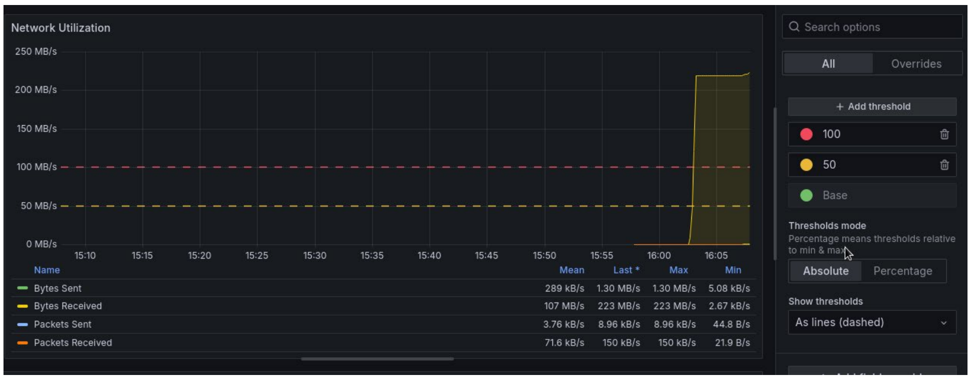
Format as: Time series — Alias by: Packets Sent

D (influxdb)

```
SELECT mean("packets_recv") / 1024 / 1024 FROM "net" WHERE ("interface"::tag = 'enp0s3') AND $timeFilter GROUP BY time($interval) fill(null)
```

Format as: Time series — Alias by: Packets Received

- Threshold Implementation
- These thresholds are set to absolute mode and dynamically change according to the system's network utilization. They are displayed on the graph as dashed horizontal lines.
- Red threshold (100 MB/s): Critical level, network usage is dangerously high.
- Yellow threshold (50 MB/s): Warning level, moderate network usage that needs monitoring.



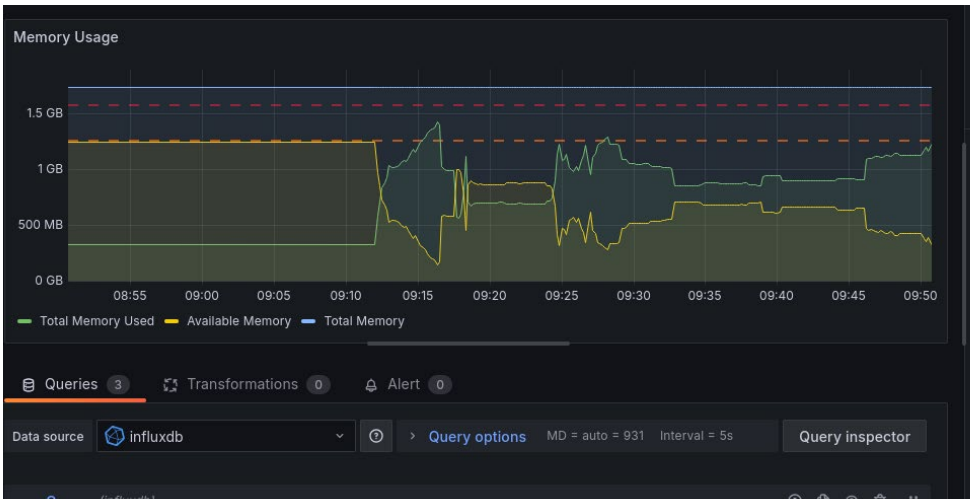
Name	Mean	Last *	Max	Min
Bytes Sent	289 kB/s	1.30 MB/s	1.30 MB/s	5.08 kB/s
Bytes Received	107 MB/s	223 MB/s	223 MB/s	2.67 kB/s
Packets Sent	3.76 kB/s	8.96 kB/s	8.96 kB/s	44.8 B/s
Packets Received	71.6 kB/s	150 kB/s	150 kB/s	21.9 B/s

Thresholds: ● 100, ● 50, ● Base

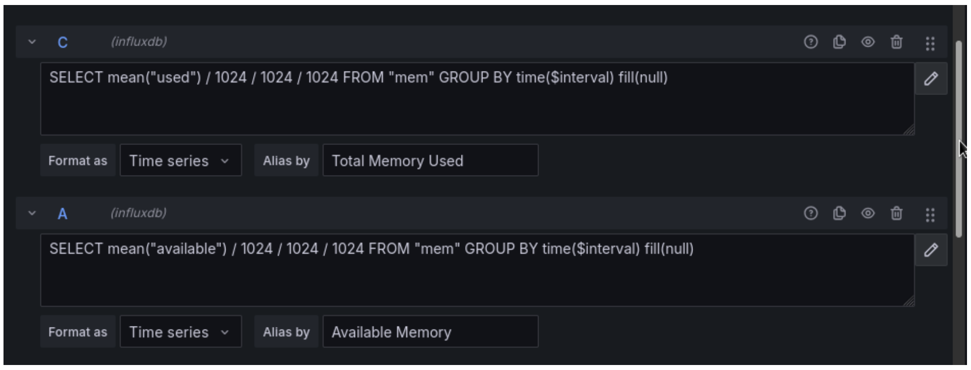
Thresholds mode: Percentage means thresholds relative to min & max — [✓ Absolute] Percentage

Show thresholds: As lines (dashed)

- **Metrics 3: Memory usage**
- The relationship between total memory, memory used, and available memory over time can be seen visually in the Grafana panel. While the X-axis shows time, the Y-axis shows memory in gigabytes (GB).
- Green line: Indicates total memory that is currently being used.
  Yellow line: Indicates available memory.
  Blue line: Represents the total system memory.
- Memory usage metrics can help in monitoring system performance, preventing it from degradation and allocating resources as efficiently as possible to improve system stability.
- Based on the image below, it is possible to see memory usage spikes and variations, which show that the system has different resource requirements.



- Query 1: Total memory used.
- Query 2: Available memory.



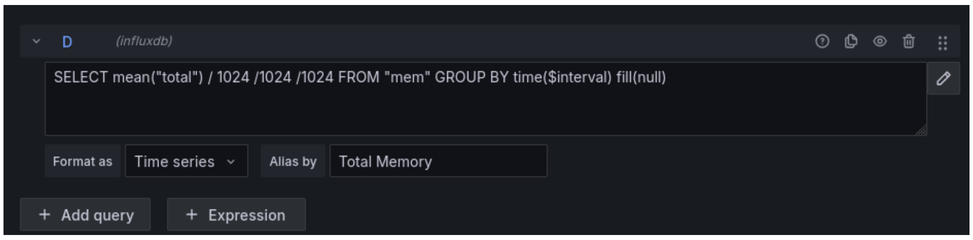
```
SELECT mean("used") / 1024 / 1024 / 1024 FROM "mem" GROUP BY time($interval) fill(null)
```

Format as Time series Alias by Total Memory Used

```
SELECT mean("available") / 1024 / 1024 / 1024 FROM "mem" GROUP BY time($interval) fill(null)
```

Format as Time series Alias by Available Memory

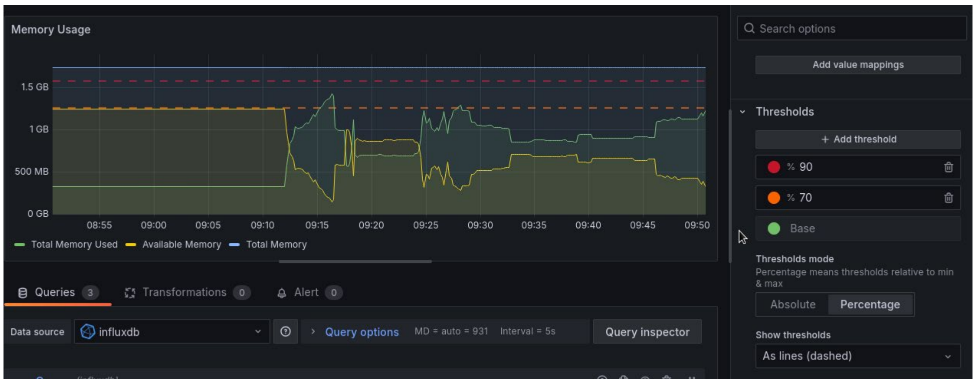
- Query 3: Total system memory.



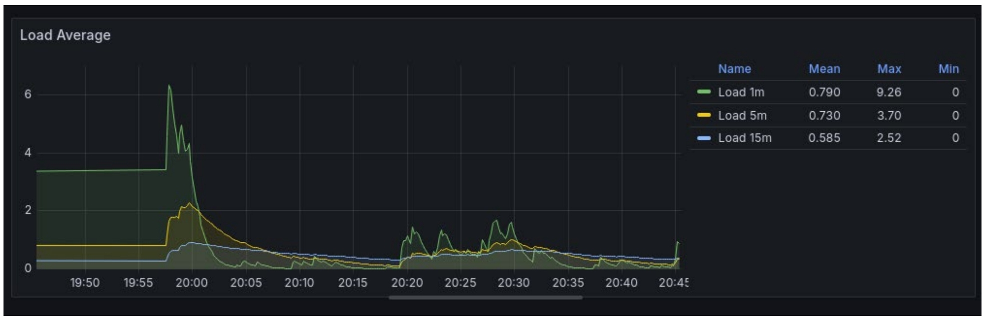
```
SELECT mean("total") / 1024 /1024 /1024 FROM "mem" GROUP BY time($interval) fill(null)
```

Format as: Time series
Alias by: Total Memory

- Threshold Implementation
- These thresholds are set to % mode and dynamically change according to the system's total RAM. They are displayed on the graph as dashed horizontal lines.
- Red threshold (90%): Critical level, memory usage is dangerously high.
- Orange threshold (70%): Warning level, moderate memory usage that needs monitoring.



- **Metrics 4: Load Average**
- The Grafana panel for system loads average monitoring is demonstrated in this section. It uses queries set up in InfluxDB to provide important load metrics over time.
- Time is shown by the X-axis, while the load value is measured by the Y-axis. The graph makes it easier to see whether the system is mild or heavy load.
- The legend includes:
  Mean: Represents the average system load over the selected time range.
  Max: Represents the highest recorded load value.
  Min: Represents the minimum load during the period.
- An estimate of the number of processes awaiting execution is provided by load averages. Potential overloading is indicated by a load figure that is near or more than the number of CPU cores. Low load values indicate underutilization of the system.

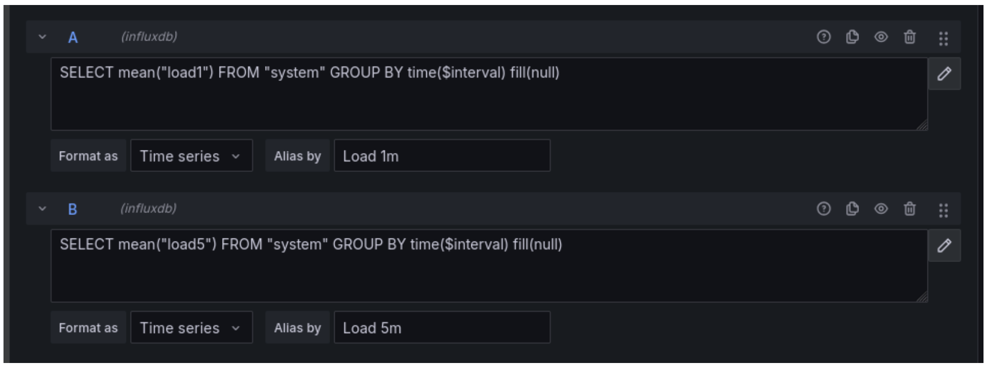


Name	Mean	Max	Min
🟩 Load 1m	0.790	9.26	0
🟨 Load 5m	0.730	3.70	0
🟦 Load 15m	0.585	2.52	0

- Query 1: Configured to fetch the 1-minute load averages.

  Monitors the system load average over the last 1 minute. This is helpful for recording short-term increases in the demand for resources.
- Query 2: Configured to fetch the 5-minutes load averages.

  Provides a medium-term view of workload by tracking the average system load over the last 5 minutes.



```
SELECT mean("load1") FROM "system" GROUP BY time($interval) fill(null)
```

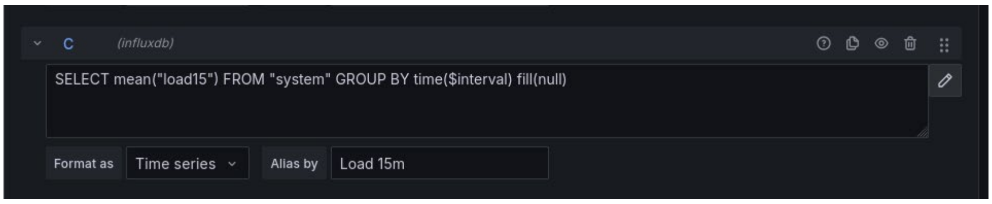
Format as: Time series — Alias by: Load 1m

```
SELECT mean("load5") FROM "system" GROUP BY time($interval) fill(null)
```

Format as: Time series — Alias by: Load 5m

- Query 3: Configured to fetch the 15-minutes load averages.

  Provides a more comprehensive view of total system usage by displaying the average system load over the last 15 minutes.

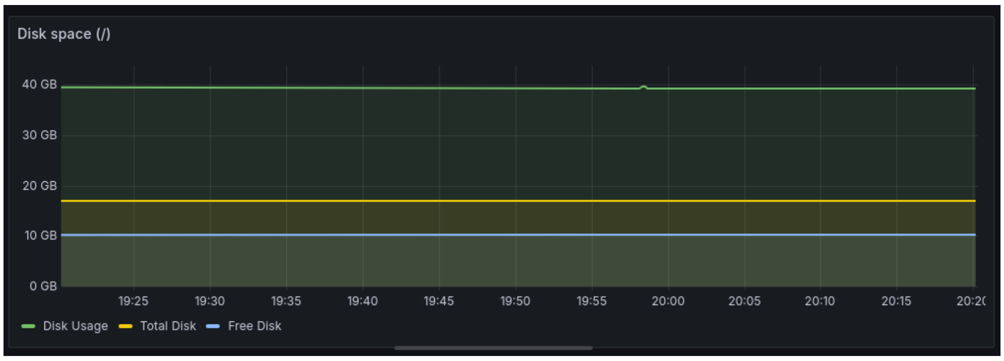


```
SELECT mean("load15") FROM "system" GROUP BY time($interval) fill(null)
```

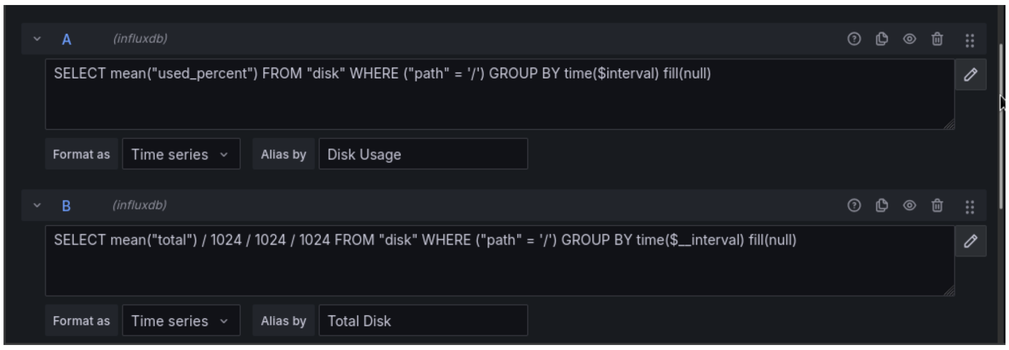
Format as: Time series — Alias by: Load 15m

- **Metrics 5: Disk Space (/)**
- The image below displays the Grafana panel for tracking disk space use on the root directory (/). This configuration helps identify storage bottlenecks or possible disk capacity problems by giving real-time insights into disk utilization.
- The X-axis records the timeline, allowing for both historical and real-time disk utilization analysis, while the Y-axis measures the disk space in gigabytes (GB).

- By enabling proactive maintenance and guaranteeing system stability, the disk monitoring configuration offers early notice of possible storage problems. It is especially helpful in settings where system functions depend on storage space or where it is limited.



- Query 1: Disk Usage
  Tracks the proportion of disk space that is presently occupied by the root directory (/).
- Query 2: Total Disk
  Shows the total amount of disk space in gigabytes (GB) on the root directory (/).



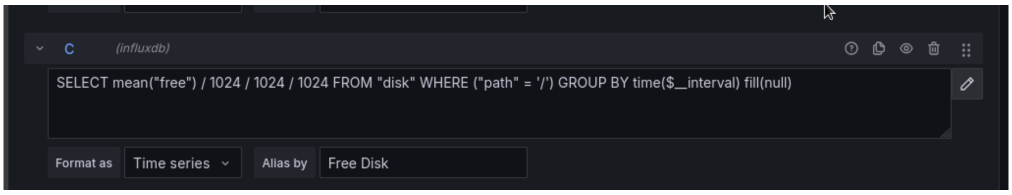
```
SELECT mean("used_percent") FROM "disk" WHERE ("path" = '/') GROUP BY time($interval) fill(null)
```

Format as: Time series
Alias by: Disk Usage

```
SELECT mean("total") / 1024 / 1024 / 1024 FROM "disk" WHERE ("path" = '/') GROUP BY time($__interval) fill(null)
```

Format as: Time series
Alias by: Total Disk

- Query 3: Free Disk
  Displays the gigabytes (GB) of free disk space that is available in the root directory (/).

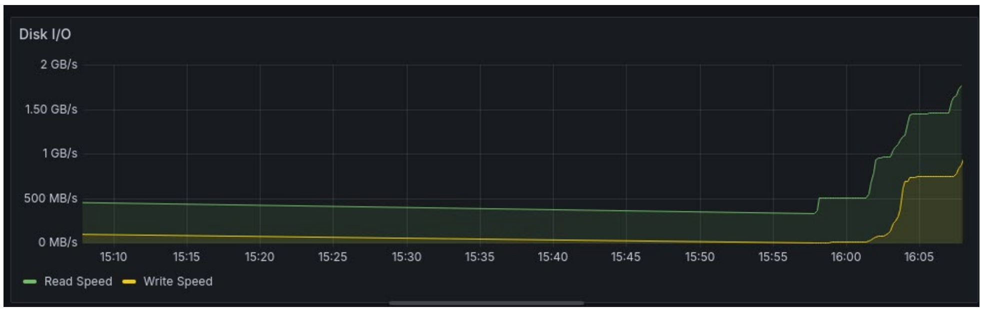


```
SELECT mean("free") / 1024 / 1024 / 1024 FROM "disk" WHERE ("path" = '/') GROUP BY time($__interval) fill(null)
```

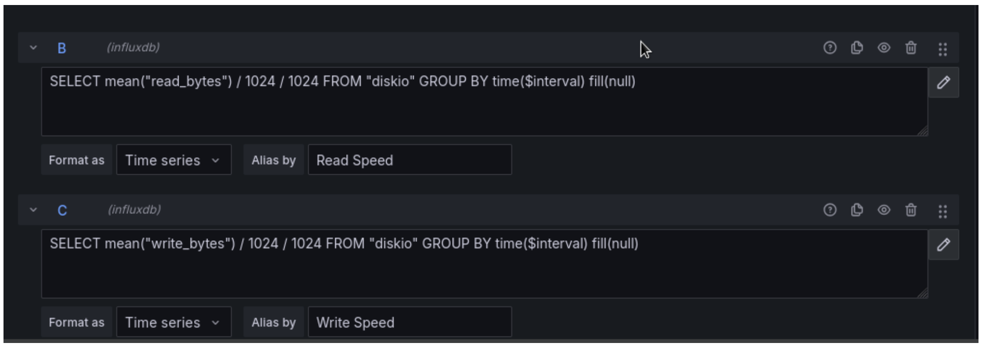
Format as: Time series
Alias by: Free Disk

- **Metrics 6: Disk I/O**
- This section demonstrates the configuration of read and write speed using InfluxDB and displays the Grafana disk I/O monitoring panel.

- Monitoring Disk I/O is significantly important to prevent system performance degradation, identify bottlenecks (such as heavy disk activity that could lead to application slowness), and plan on additional disk resources when needed.
- Green line (Read speed): The speed at which data is being read from the disk. Yellow line (Write speed); The speed at which data is being written to the disk.



- Query 1: Read Speed
  Monitors the number of bytes read over time to get the average read speed in megabytes per second (MB/s).
- Query 2: Write Speed
  Tracks the average write speed in megabytes per second (MB/s) by counting the number of bytes that are written to the disk over time.



B (influxdb)

```
SELECT mean("read_bytes") / 1024 / 1024 FROM "diskio" GROUP BY time($interval) fill(null)
```

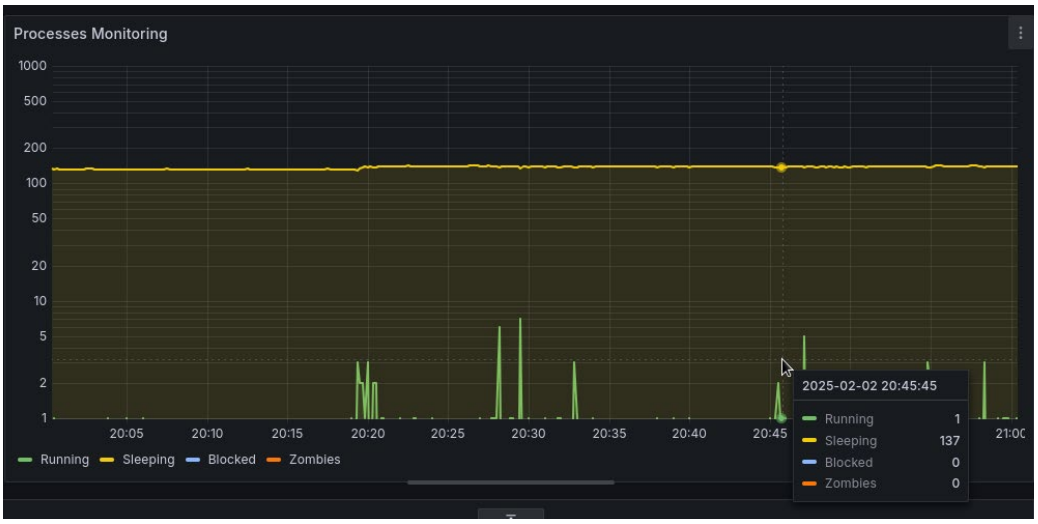
Format as: Time series
Alias by: Read Speed

C (influxdb)

```
SELECT mean("write_bytes") / 1024 / 1024 FROM "diskio" GROUP BY time($interval) fill(null)
```

Format as: Time series
Alias by: Write Speed

- **Metrics 7: Process Monitoring**
- The image below shows the Grafana panel for system process monitoring, which tracks the status of processes that are blocked, sleeping, running, and zombie-stated using key queries from InfluxDB.
- Time is represented by the X-axis, while the number of processes is shown on the Y-axis.
- Process state monitoring is crucial for diagnosing resource conflicts or application faults as well as for analyzing system performance.

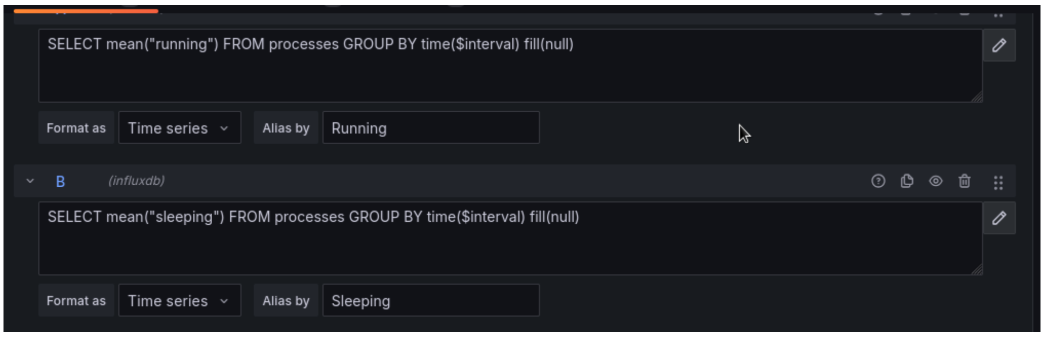


- Query 1: Running

  Keep track of how many processes are running on the system at any given time.

- Query 2: Sleeping

  Keep track of processes that are inactive (awaiting resources).



```
SELECT mean("running") FROM processes GROUP BY time($interval) fill(null)
```

Format as Time series Alias by Running

B (influxdb)

```
SELECT mean("sleeping") FROM processes GROUP BY time($interval) fill(null)
```

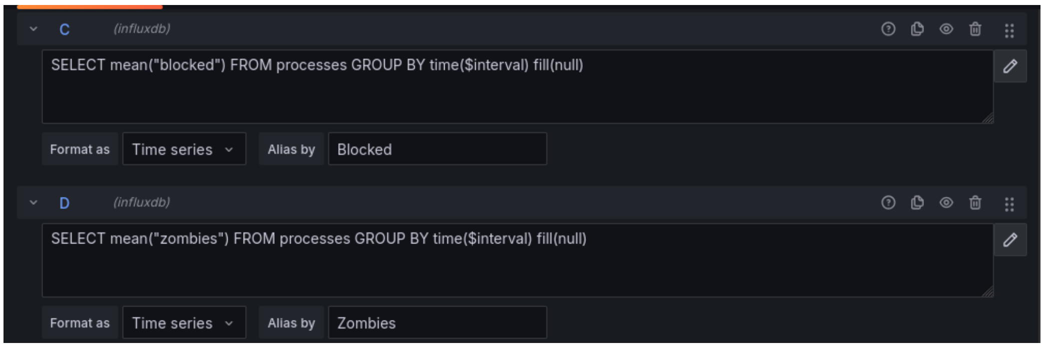
Format as Time series Alias by Sleeping

- Query 3: Blocked

  Keep track of how many processes are blocked and awaiting input, output, or other resources.

- Query 4: Zombies

  Identifies processes that have completed their execution but still have entries in the process table, which suggests a problem with resource cleanup.



C (influxdb)

```
SELECT mean("blocked") FROM processes GROUP BY time($interval) fill(null)
```

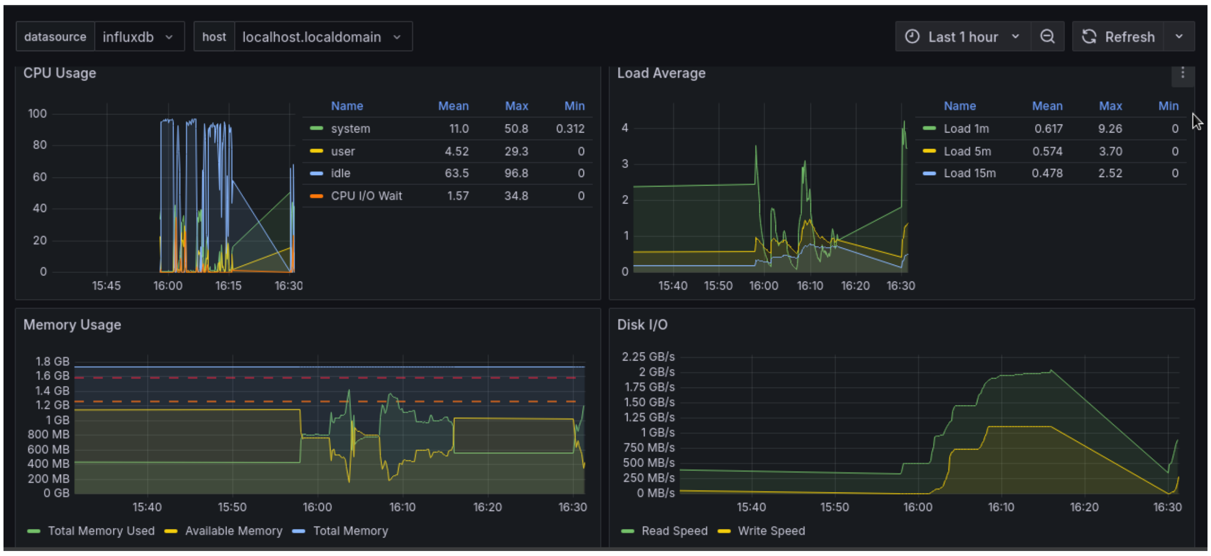
Format as Time series Alias by Blocked

D (influxdb)

```
SELECT mean("zombies") FROM processes GROUP BY time($interval) fill(null)
```

Format as Time series Alias by Zombies

Overview of the Grafana Dashboard

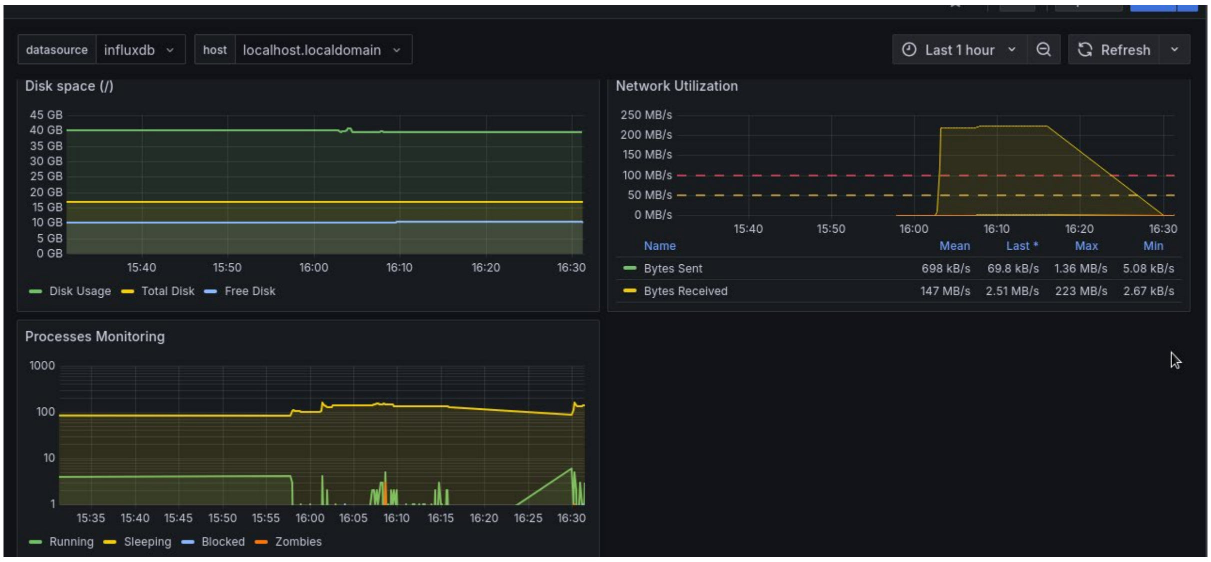


CPU Usage

Name	Mean	Max	Min
system	11.0	50.8	0.312
user	4.52	29.3	0
idle	63.5	96.8	0
CPU I/O Wait	1.57	34.8	0

Load Average

Name	Mean	Max	Min
Load 1m	0.617	9.26	0
Load 5m	0.574	3.70	0
Load 15m	0.478	2.52	0



Network Utilization

Name	Mean	Last *	Max	Min
Bytes Sent	698 kB/s	69.8 kB/s	1.36 MB/s	5.08 kB/s
Bytes Received	147 MB/s	2.51 MB/s	223 MB/s	2.67 kB/s