PROMETHEUS

PROMETHEUS - **Used for Event monitoring and Alerting**.

It records real-time metrics in a TIME SERIES DATABASE built using a HTTP pull model, with flexible queries and real-time alerting. Prometheus collects data and metrics from different services and stores them according to a unique identifier.

Prometheus collects and stores its metrics as time series data, i.e., metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels

TSB - TIME SERIES DATABASE **-** A time series database (TSDB) is a database optimized for time-stamped or time series data. Time series data are simply measurements or events that are tracked, monitored, down sampled, and aggregated over time. This could be server metrics, application performance monitoring, network data, sensor data, events, clicks, trades in a market, and many other types of analytics data.

PROMQL - A query language for Prometheus monitoring system. It is designed for building powerful yet simple queries for graphs, alerts, or derived time series.

EXPORTER – Software that gets existing metrics from a third-party system and export them to the metric format that the Prometheus server can understand.

NODE EXPORTER

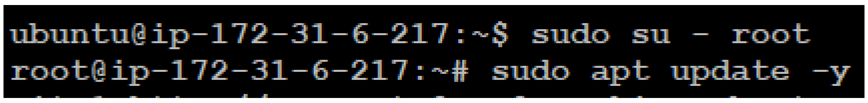
Prometheus exporter for hardware and OS metrics exposed by UNIX kernels, written in Go with pluggable metric collectors. The node exporter listens on HTTP port 9100 by default

MYSQL EXPORTER

Supported versions: MySQL >= 5.6 & MariaDB >= 10.2

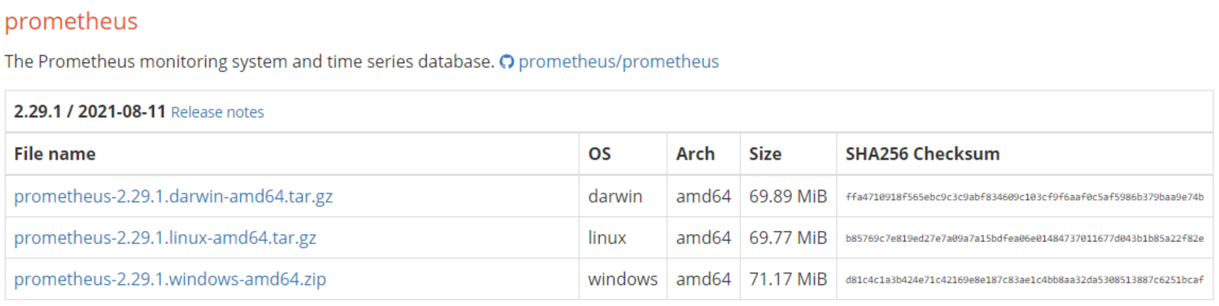
PROMETHEUS INSTALLATION

1. Login to **UBUNTU** Machine in AWS Console as ROOT user
2. Check for all updates using command – "sudo apt update -y"



```
ubuntu@ip-172-31-6-217:~$ sudo su - root
root@ip-172-31-6-217:~# sudo apt update -y
```

3. Download PROMETHEUS package for LINUX OS from https://prometheus.io/download/



prometheus

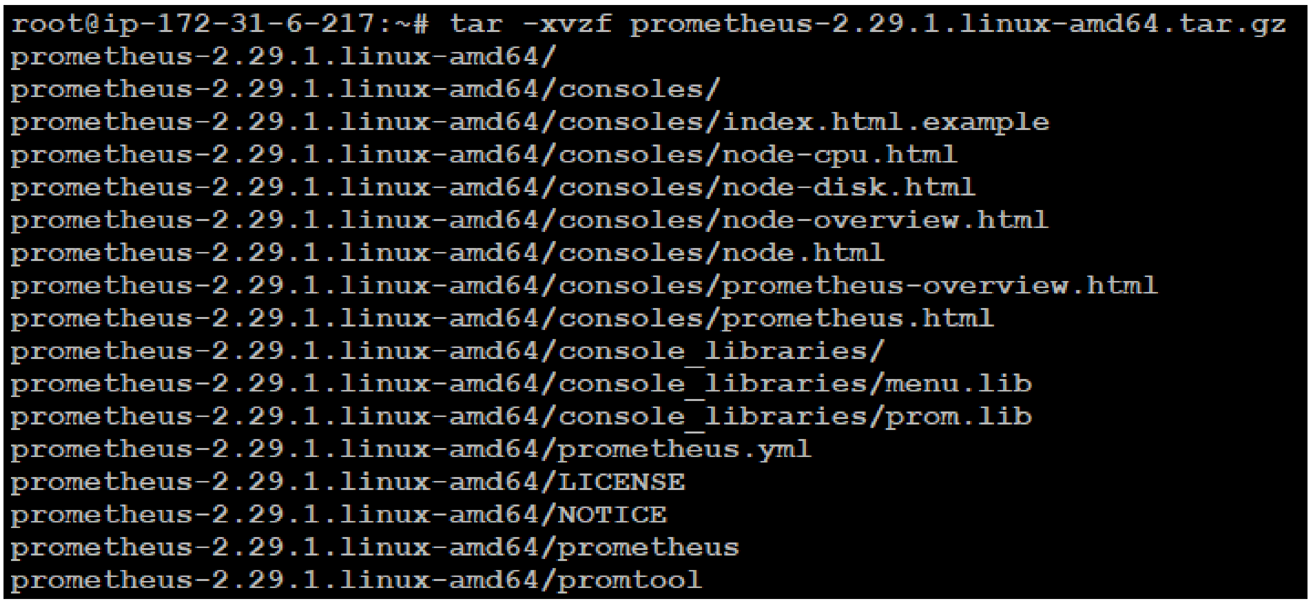
The Prometheus monitoring system and time series database. prometheus/prometheus

2.29.1 / 2021-08-11 Release notes

File name	OS	Arch	Size	SHA256 Checksum
prometheus-2.29.1.darwin-amd64.tar.gz	darwin	amd64	69.89 MiB	ffa4710918f565ebc9c3c9abf834609c103cf9f6aaf0c5af5986b379baa9e74b
prometheus-2.29.1.linux-amd64.tar.gz	linux	amd64	69.77 MiB	b85769c7e819ed27e7a09a7a15bdfea06e01484737011677d043b1b85a22f82e
prometheus-2.29.1.windows-amd64.zip	windows	amd64	71.17 MiB	d81c4c1a3b424e71c42169e8e187c83ae1c4bb8aa32da5308513887c6251bcaf

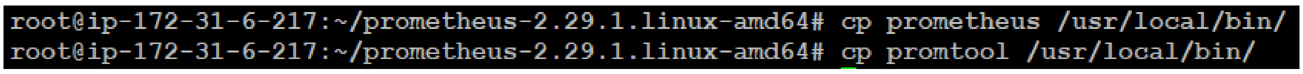
4. Using WGET command, Download to the respective server
   wget https://github.com/prometheus/prometheus/releases/download/v2.29.1/prometheus-2.29.1.linux-amd64.tar.gz

5. Extract **PROMETHEUS DIRECTORY** using TAR Command
   root@ip-172-31-6-217:~# tar -xvzf prometheus-2.29.1.linux-amd64.tar.gz



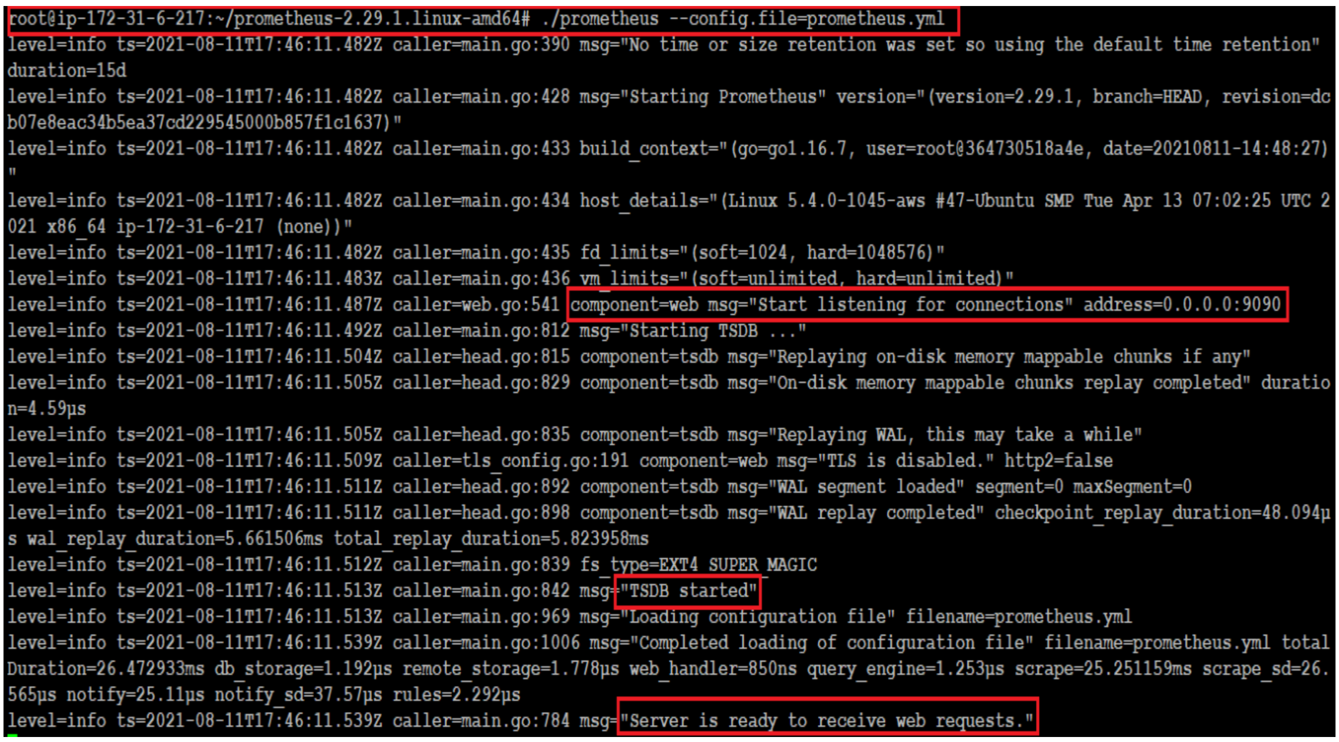
```
root@ip-172-31-6-217:~# tar -xvzf prometheus-2.29.1.linux-amd64.tar.gz
prometheus-2.29.1.linux-amd64/
prometheus-2.29.1.linux-amd64/consoles/
prometheus-2.29.1.linux-amd64/consoles/index.html.example
prometheus-2.29.1.linux-amd64/consoles/node-cpu.html
prometheus-2.29.1.linux-amd64/consoles/node-disk.html
prometheus-2.29.1.linux-amd64/consoles/node-overview.html
prometheus-2.29.1.linux-amd64/consoles/node.html
prometheus-2.29.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.29.1.linux-amd64/consoles/prometheus.html
prometheus-2.29.1.linux-amd64/console_libraries/
prometheus-2.29.1.linux-amd64/console_libraries/menu.lib
prometheus-2.29.1.linux-amd64/console_libraries/prom.lib
prometheus-2.29.1.linux-amd64/prometheus.yml
prometheus-2.29.1.linux-amd64/LICENSE
prometheus-2.29.1.linux-amd64/NOTICE
prometheus-2.29.1.linux-amd64/prometheus
prometheus-2.29.1.linux-amd64/promtool
```

6. Navigate to prometheus-2.29.1.linux-amd64/ and copy "prometheus" & "promtool" to /usr/local/bin



```
root@ip-172-31-6-217:~/prometheus-2.29.1.linux-amd64# cp prometheus /usr/local/bin/
root@ip-172-31-6-217:~/prometheus-2.29.1.linux-amd64# cp promtool /usr/local/bin/
```

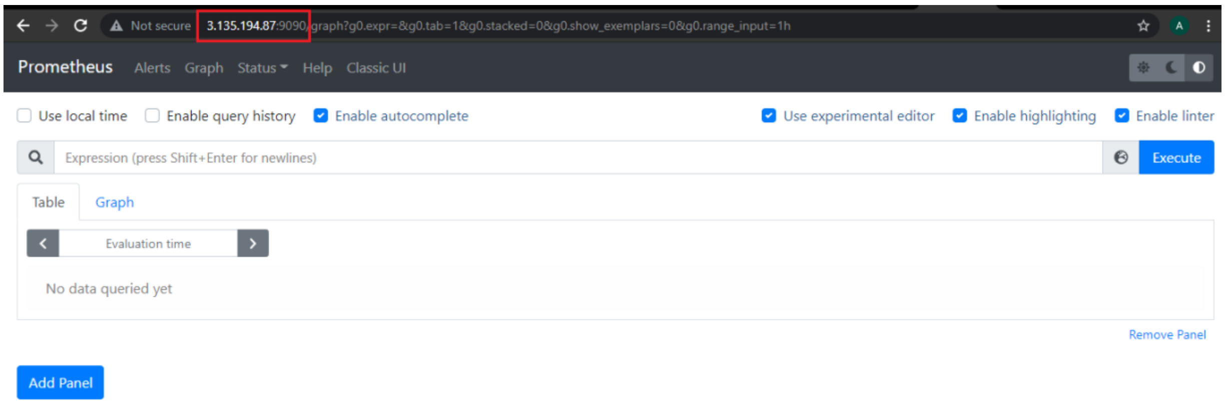
7. Start **PROMETHEUS** process using "./prometheus --config.file=prometheus.yml"



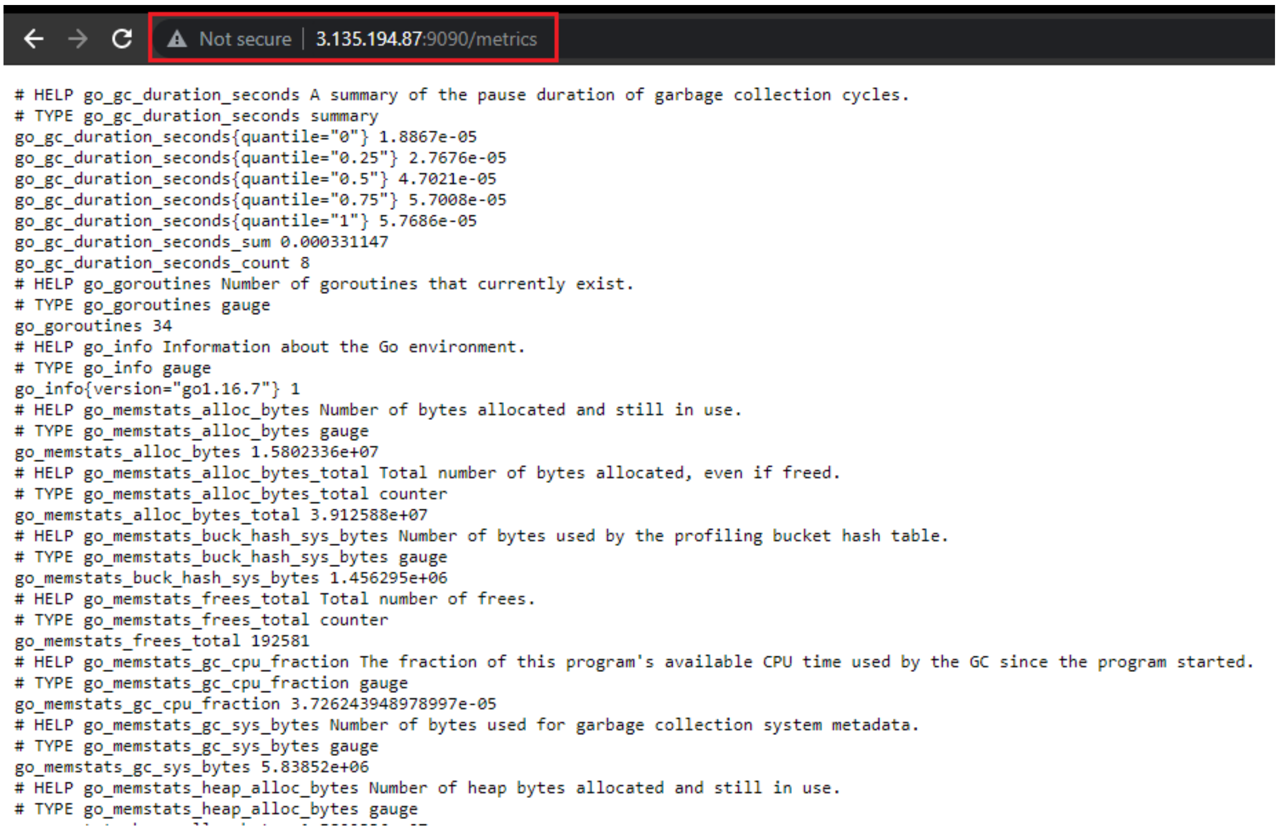
```
root@ip-172-31-6-217:~/prometheus-2.29.1.linux-amd64# ./prometheus --config.file=prometheus.yml
level=info ts=2021-08-11T17:46:11.482Z caller=main.go:390 msg="No time or size retention was set so using the default time retention"
duration=15d
level=info ts=2021-08-11T17:46:11.482Z caller=main.go:428 msg="Starting Prometheus" version="(version=2.29.1, branch=HEAD, revision=dc
b07e8eac34b5ea37cd229545000b857f1c1637)"
level=info ts=2021-08-11T17:46:11.482Z caller=main.go:433 build_context="(go=go1.16.7, user=root@364730518a4e, date=20210811-14:48:27)
"
level=info ts=2021-08-11T17:46:11.482Z caller=main.go:434 host_details="(Linux 5.4.0-1045-aws #47-Ubuntu SMP Tue Apr 13 07:02:25 UTC 2
021 x86_64 ip-172-31-6-217 (none))"
level=info ts=2021-08-11T17:46:11.482Z caller=main.go:435 fd_limits="(soft=1024, hard=1048576)"
level=info ts=2021-08-11T17:46:11.483Z caller=main.go:436 vm_limits="(soft=unlimited, hard=unlimited)"
level=info ts=2021-08-11T17:46:11.487Z caller=web.go:541 component=web msg="Start listening for connections" address=0.0.0.0:9090
level=info ts=2021-08-11T17:46:11.492Z caller=main.go:812 msg="Starting TSDB ..."
level=info ts=2021-08-11T17:46:11.504Z caller=head.go:815 component=tsdb msg="Replaying on-disk memory mappable chunks if any"
level=info ts=2021-08-11T17:46:11.505Z caller=head.go:829 component=tsdb msg="On-disk memory mappable chunks replay completed" duratio
n=4.59µs
level=info ts=2021-08-11T17:46:11.505Z caller=head.go:835 component=tsdb msg="Replaying WAL, this may take a while"
level=info ts=2021-08-11T17:46:11.509Z caller=tls_config.go:191 component=web msg="TLS is disabled." http2=false
level=info ts=2021-08-11T17:46:11.511Z caller=head.go:892 component=tsdb msg="WAL segment loaded" segment=0 maxSegment=0
level=info ts=2021-08-11T17:46:11.511Z caller=head.go:898 component=tsdb msg="WAL replay completed" checkpoint_replay_duration=48.094µ
s wal_replay_duration=5.661506ms total_replay_duration=5.823958ms
level=info ts=2021-08-11T17:46:11.512Z caller=main.go:839 fs_type=EXT4_SUPER_MAGIC
level=info ts=2021-08-11T17:46:11.513Z caller=main.go:842 msg="TSDB started"
level=info ts=2021-08-11T17:46:11.513Z caller=main.go:969 msg="Loading configuration file" filename=prometheus.yml
level=info ts=2021-08-11T17:46:11.539Z caller=main.go:1006 msg="Completed loading of configuration file" filename=prometheus.yml total
Duration=26.472933ms db_storage=1.192µs remote_storage=1.778µs web_handler=850ns query_engine=1.253µs scrape=25.251159ms scrape_sd=26.
565µs notify=25.11µs notify_sd=37.57µs rules=2.292µs
level=info ts=2021-08-11T17:46:11.539Z caller=main.go:784 msg="Server is ready to receive web requests."
```

8. Verify by entering <IP_ADDRESS:9090> in the Browser as below :

   3.135.194.87:9090



9. Change <IP_ADDRESS:9090>/metrics to check the available metrics of the host



```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.8867e-05
go_gc_duration_seconds{quantile="0.25"} 2.7676e-05
go_gc_duration_seconds{quantile="0.5"} 4.7021e-05
go_gc_duration_seconds{quantile="0.75"} 5.7008e-05
go_gc_duration_seconds{quantile="1"} 5.7686e-05
go_gc_duration_seconds_sum 0.000331147
go_gc_duration_seconds_count 8
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 34
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.16.7"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.5802336e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 3.912588e+07
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.456295e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 192581
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 3.726243948978997e-05
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 5.83852e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
```

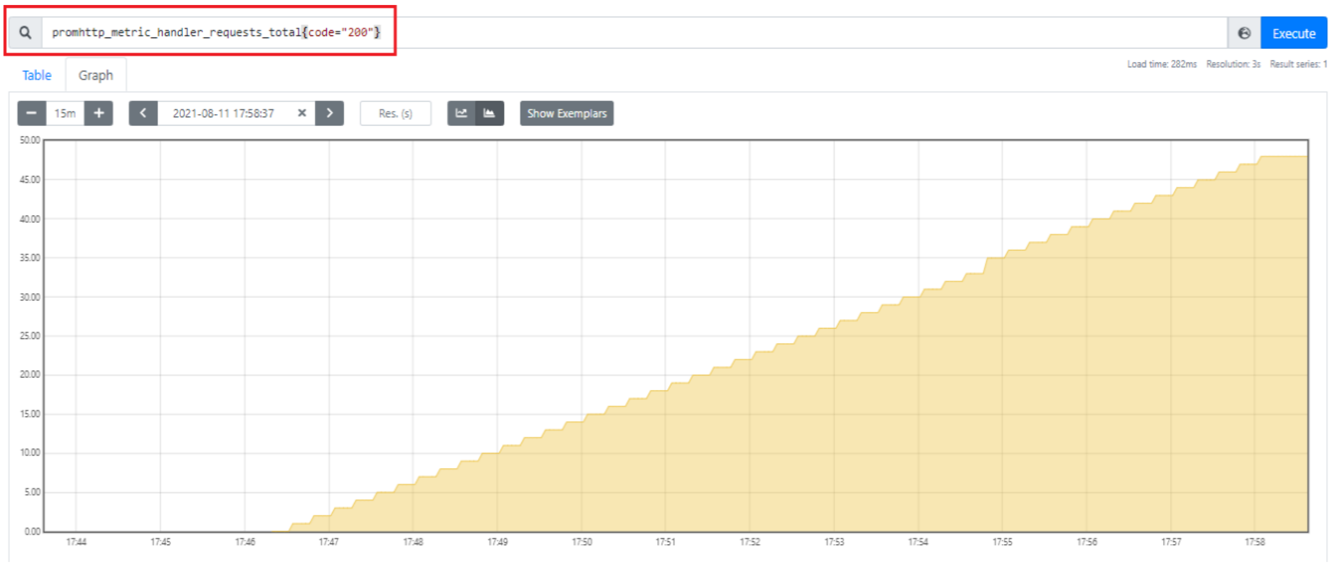
10. Check for some of metrics as below

For CPU –

process_cpu_seconds_total

HTTP status code –

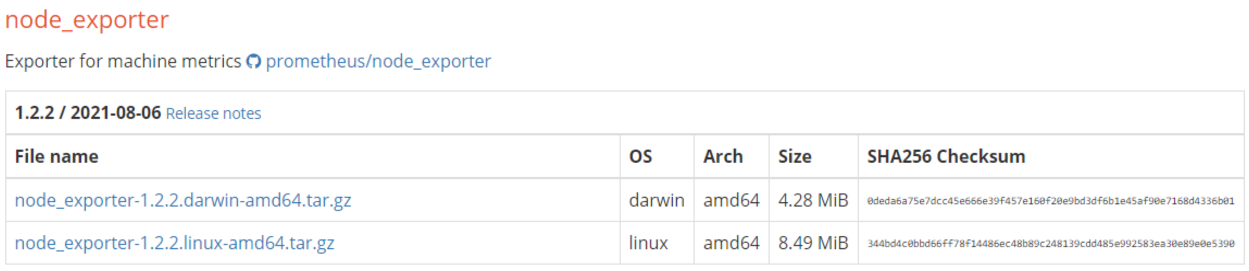
promhttp_metric_handler_requests_total{code="200"}
promhttp_metric_handler_requests_total{code="500"}
promhttp_metric_handler_requests_total{code="503"}



```
promhttp_metric_handler_requests_total{code="200"}
```

INSTALLATION OF NODE EXPORTER

1. Login to UBUNTU machine (Depending upon your requirement create the number of servers) For better understanding, create more than 2 servers for Assessment.
2. Follow same steps – Login as ROOT user and check for updates.
3. Navigate to NODE EXPORTER in https://prometheus.io/download/



node_exporter

Exporter for machine metrics prometheus/node_exporter

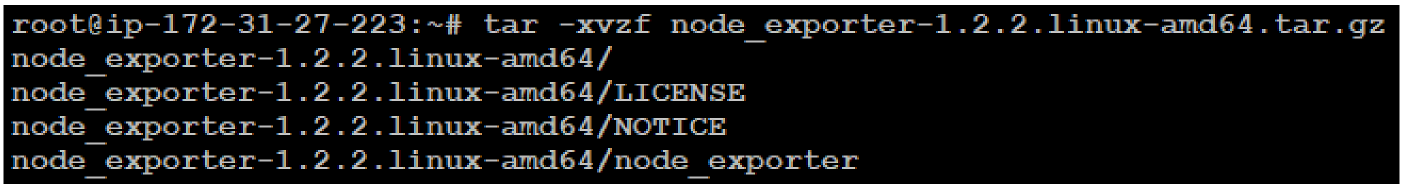
1.2.2 / 2021-08-06 Release notes

File name	OS	Arch	Size	SHA256 Checksum
node_exporter-1.2.2.darwin-amd64.tar.gz	darwin	amd64	4.28 MiB	0deda6a75e7dcc45e666e39f457e160f20e9bd3df6b1e45af90e7168d4336b01
node_exporter-1.2.2.linux-amd64.tar.gz	linux	amd64	8.49 MiB	344bd4c0bbd66ff78f14486ec48b89c248139cdd485e992583ea30e89e0e5390

4. Using **WGET** command – Extract Node Exporter file to the respective servers

   wget https://github.com/prometheus/node_exporter/releases/download/v1.2.2/node_exporter-1.2.2.linux-amd64.tar.gz

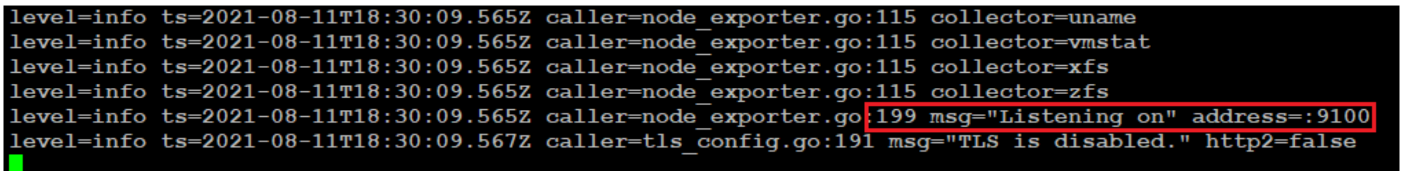
5. Extract using **TAR** command : root@ip-172-31-27-223:~# tar -xvzf node_exporter-1.2.2.linux-amd64.tar.gz



```
root@ip-172-31-27-223:~# tar -xvzf node_exporter-1.2.2.linux-amd64.tar.gz
node_exporter-1.2.2.linux-amd64/
node_exporter-1.2.2.linux-amd64/LICENSE
node_exporter-1.2.2.linux-amd64/NOTICE
node_exporter-1.2.2.linux-amd64/node_exporter
```

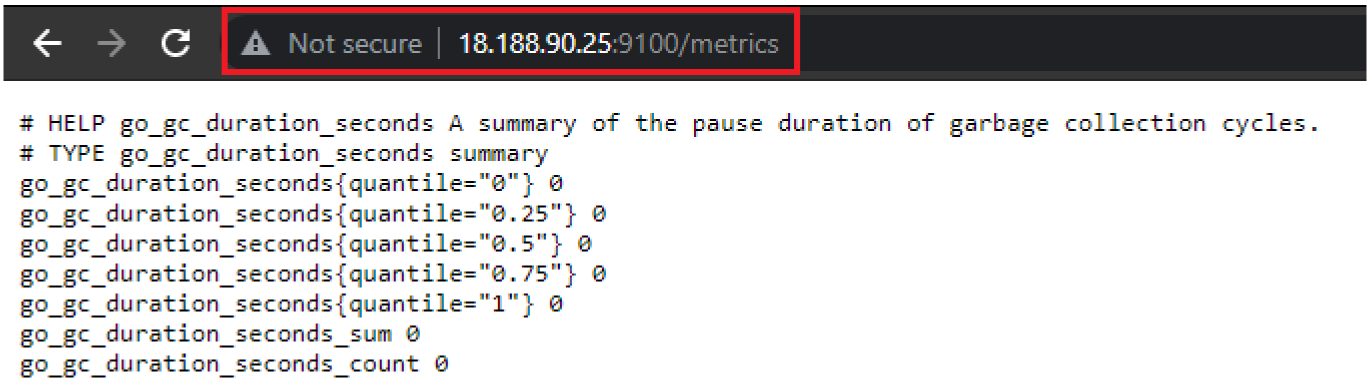
6. Navigate to node_exporter-1.2.2.linux-amd64/ and start NODE EXPORTER PROCESS by below command

root@ip-172-31-27-223:~/node_exporter-1.2.2.linux-amd64# ./node_exporter &



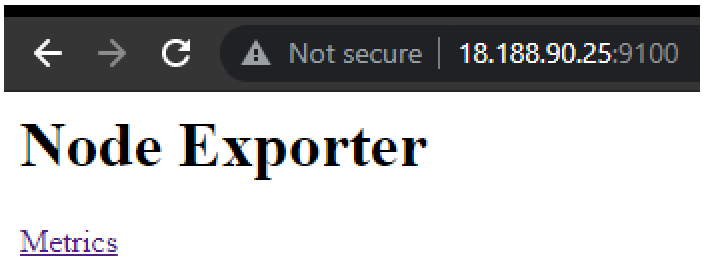
```
level=info ts=2021-08-11T18:30:09.565Z caller=node_exporter.go:115 collector=uname
level=info ts=2021-08-11T18:30:09.565Z caller=node_exporter.go:115 collector=vmstat
level=info ts=2021-08-11T18:30:09.565Z caller=node_exporter.go:115 collector=xfs
level=info ts=2021-08-11T18:30:09.565Z caller=node_exporter.go:115 collector=zfs
level=info ts=2021-08-11T18:30:09.565Z caller=node_exporter.go:199 msg="Listening on" address=:9100
level=info ts=2021-08-11T18:30:09.567Z caller=tls_config.go:191 msg="TLS is disabled." http2=false
```

7. Verify using <IP_ADDRESS:9100> - By default Node Exporter uses **PORT 9100**



Not secure | 18.188.90.25:9100/metrics

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
```



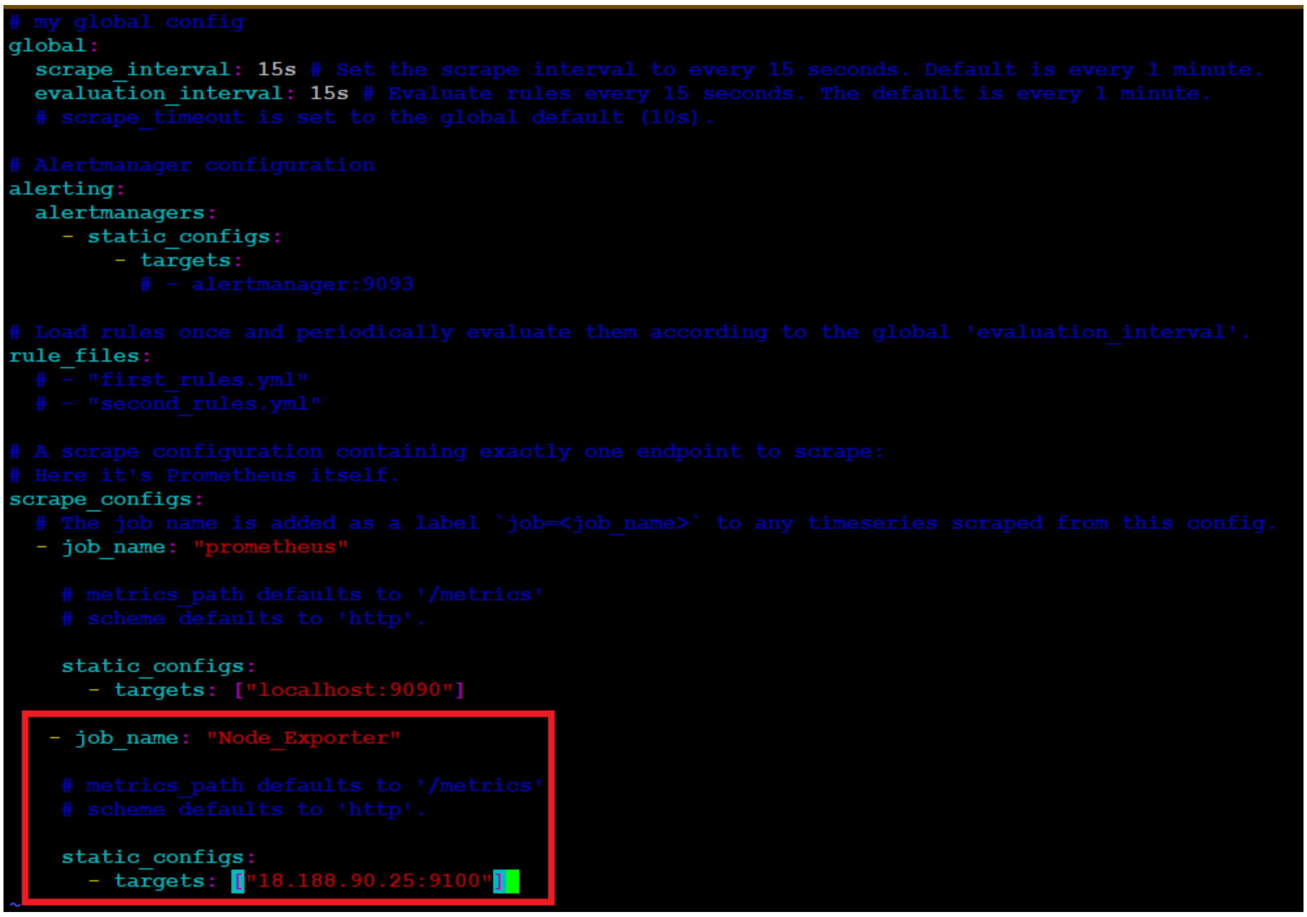
Not secure | 18.188.90.25:9100

Node Exporter

Metrics

Now we need to map **NODE EXPORTER** server to our **PROMETHEUS** server to fetch metrics under **PROMETHEUS** UI – (Refer **STEP 8** in **PROMETHEUS INSTALLATION - 3.135.194.87:9090**)

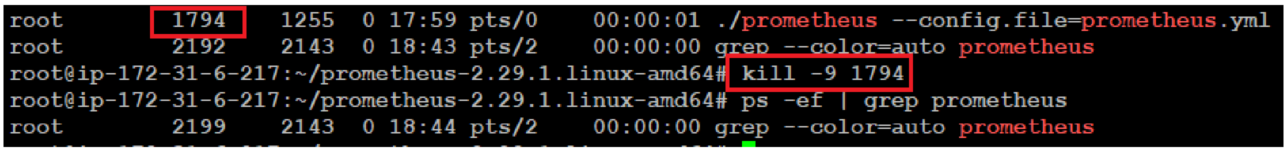
1. Login PROMETHEUS SERVER and navigate to /root/prometheus-2.29.1.linux-amd64
2. Open **prometheus.yml** file and add below details



```
# 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: "Node_Exporter"

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

    static_configs:
      - targets: ["18.188.90.25:9100"]
~
```

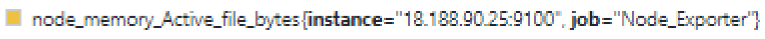
```
  - job_name: "Node_Exporter"

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

    static_configs:
      - targets: ["18.188.90.25:9100"] ----→ IP_ADDRESS:9100
```

3. Stop PROMETHEUS PROCESS by Killing it
   ps -ef | grep prometheus
   kill -9 <PID>



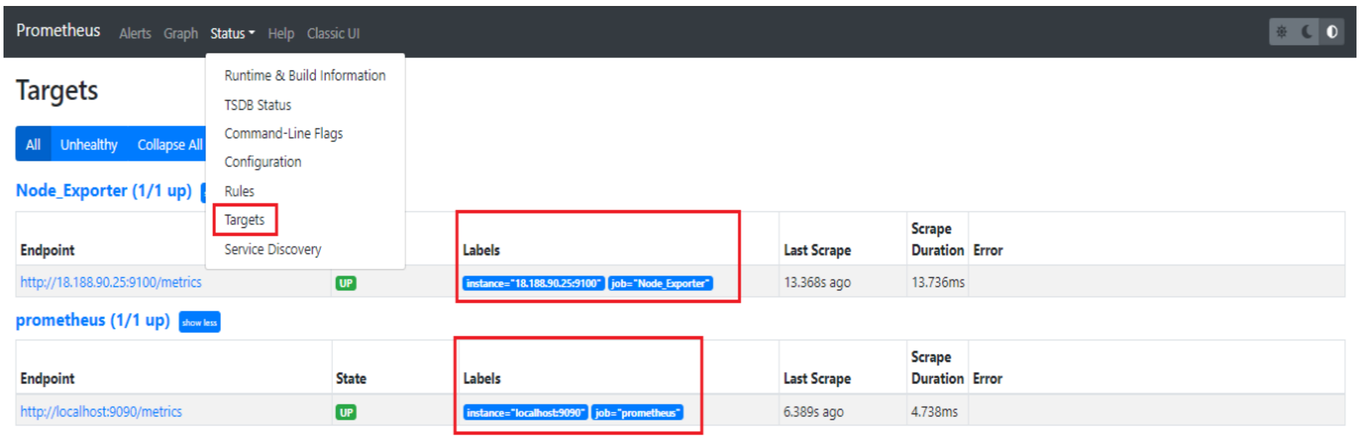
```
root      1794   1255  0 17:59 pts/0    00:00:01 ./prometheus --config.file=prometheus.yml
root      2192   2143  0 18:43 pts/2    00:00:00 grep --color=auto prometheus
root@ip-172-31-6-217:~/prometheus-2.29.1.linux-amd64# kill -9 1794
root@ip-172-31-6-217:~/prometheus-2.29.1.linux-amd64# ps -ef | grep prometheus
root      2199   2143  0 18:44 pts/2    00:00:00 grep --color=auto prometheus
```

4. You will be able to see Metrics from your **NODE EXPORTER SERVER**.
   a. Check the JOB NAME – Here it is as "NODE EXPORTER"



```
node_memory_Active_file_bytes{instance="18.188.90.25:9100", job="Node_Exporter"}
```

FINAL VERIFICATION can be done under **TARGETS** to see existing **JOBS** and **SERVERS** mapped



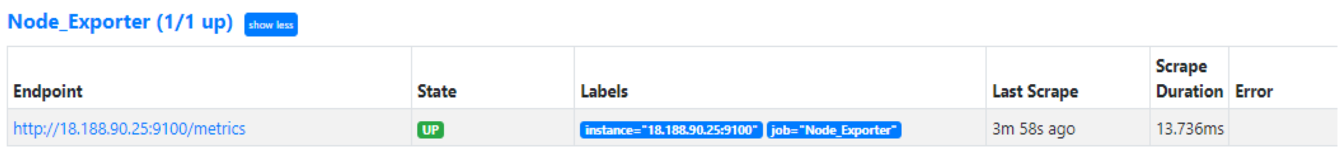
Targets

Node_Exporter (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://18.188.90.25:9100/metrics	UP	instance="18.188.90.25:9100" job="Node_Exporter"	13.368s ago	13.736ms	

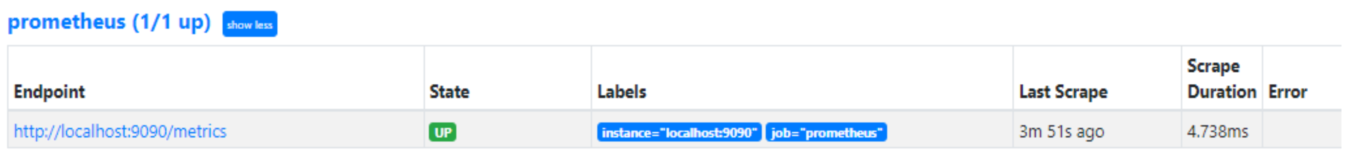
prometheus (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	6.389s ago	4.738ms	



Node_Exporter (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://18.188.90.25:9100/metrics	UP	instance="18.188.90.25:9100" job="Node_Exporter"	3m 58s ago	13.736ms	



prometheus (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	3m 51s ago	4.738ms	

WHY WE NEED GRAFANA?

PROMETHEUS can store data, but it can't be so effective in visualisations like running multiple queries and Dashboards aren't very attractive. Where **GRAFANA** acts as the interface for Analysis and Visualization.

GRAFANA

Grafana is open-source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored. It provides to turn Time-series database (TSDB) data into beautiful graphs and Visualizations.

POINTS TO REMEMBER:

DATA SOURCES - Grafana supports many different storage backends for your time series data (data source). Each data source has a specific Query Editor that is customized for the features and capabilities that the data source exposes.

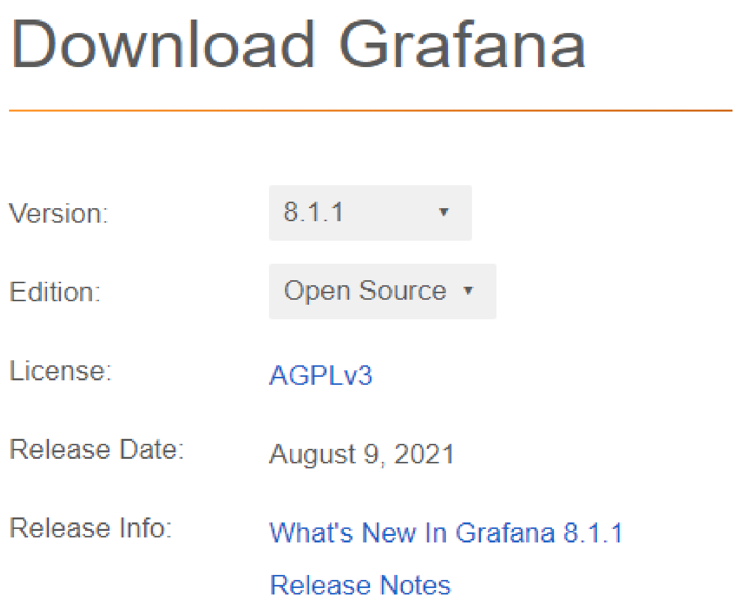
EXAMPLE OF DATA SOURCES – Prometheus, AWS CloudWatch, MYSQL, Azure Monitor

HISTOGRAM - A histogram is a graphical representation of the distribution of numerical data. It groups values into buckets (sometimes also called bins) and then counts how many values fall into each bucket.

HEATMAPS - A heatmap is like a histogram, but over time where each time slice represents its own histogram. Instead of using bar height as a representation of frequency, it uses cells and colours the cell proportional to the number of values in the bucket.

INSTALLATION OF GRAFANA

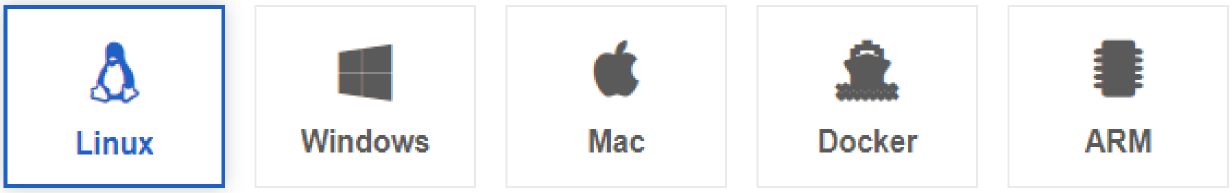
1. Login to PROMETHEUS server (or create a new server to avoid confusion)
2. Download latest version of Grafana from https://grafana.com/grafana/download



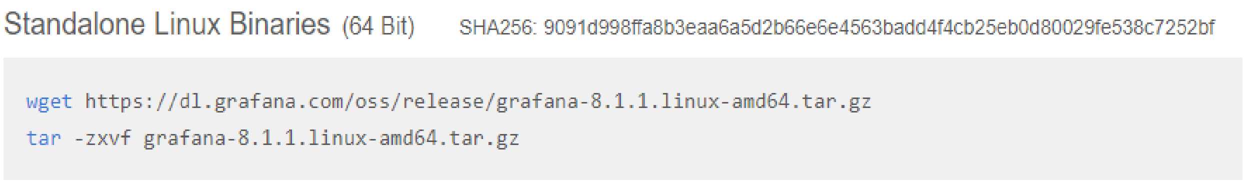
Download Grafana

Version:	8.1.1 ▾
Edition:	Open Source ▾
License:	AGPLv3
Release Date:	August 9, 2021
Release Info:	What's New In Grafana 8.1.1 Release Notes

3. Select the version **(8.1.1)** and Edition as **OPEN SOURCE** and **LINUX** OS



4. Under Standalone Linux Binaries, you can find the commands to directly install in your Machine

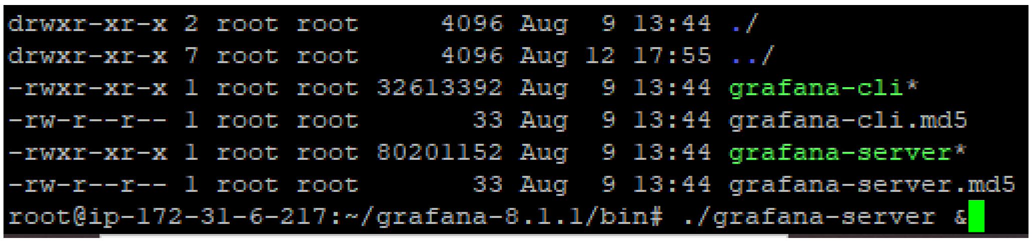


Standalone Linux Binaries (64 Bit) SHA256: 9091d998ffa8b3eaa6a5d2b66e6e4563badd4f4cb25eb0d80029fe538c7252bf

```
wget https://dl.grafana.com/oss/release/grafana-8.1.1.linux-amd64.tar.gz
tar -zxvf grafana-8.1.1.linux-amd64.tar.gz
```

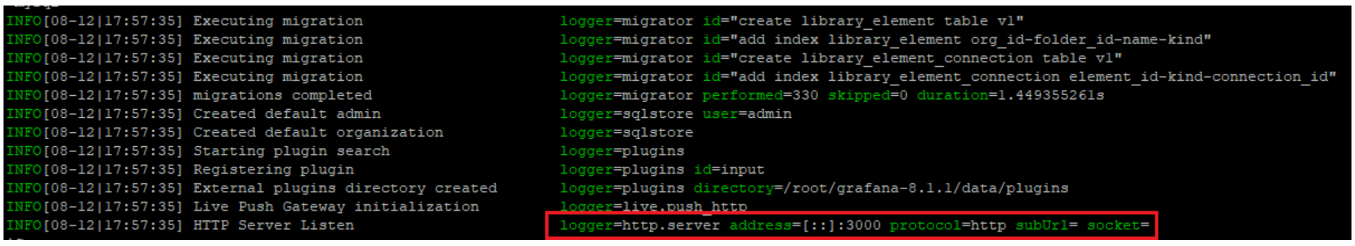
5. Navigate to grafana-8.1.1/ and BIN directory and execute below command:

root@ip-172-31-6-217:~/grafana-8.1.1/bin# ./grafana-server &



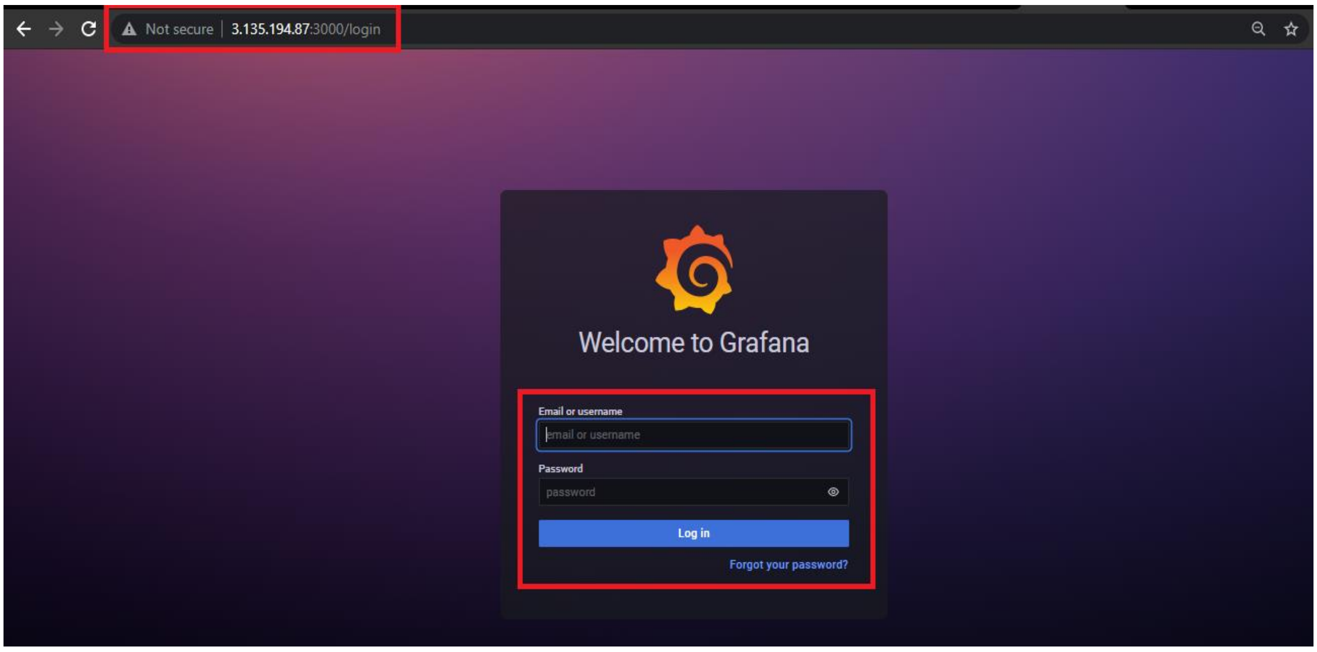
```
drwxr-xr-x 2 root root      4096 Aug  9 13:44 ./
drwxr-xr-x 7 root root      4096 Aug 12 17:55 ../
-rwxr-xr-x 1 root root 32613392 Aug  9 13:44 grafana-cli*
-rw-r--r-- 1 root root       33 Aug  9 13:44 grafana-cli.md5
-rwxr-xr-x 1 root root 80201152 Aug  9 13:44 grafana-server*
-rw-r--r-- 1 root root       33 Aug  9 13:44 grafana-server.md5
root@ip-172-31-6-217:~/grafana-8.1.1/bin# ./grafana-server &
```

6. Check if process is started properly by verifying output as below :



```
INFO[08-12|17:57:35] Executing migration                      logger=migrator id="create library_element table v1"
INFO[08-12|17:57:35] Executing migration                      logger=migrator id="add index library_element org_id-folder_id-name-kind"
INFO[08-12|17:57:35] Executing migration                      logger=migrator id="create library_element_connection table v1"
INFO[08-12|17:57:35] Executing migration                      logger=migrator id="add index library_element_connection element_id-kind-connection_id"
INFO[08-12|17:57:35] migrations completed                     logger=migrator performed=330 skipped=0 duration=1.449355261s
INFO[08-12|17:57:35] Created default admin                    logger=sqlstore user=admin
INFO[08-12|17:57:35] Created default organization             logger=sqlstore
INFO[08-12|17:57:35] Starting plugin search                   logger=plugins
INFO[08-12|17:57:35] Registering plugin                       logger=plugins id=input
INFO[08-12|17:57:35] External plugins directory created       logger=plugins directory=/root/grafana-8.1.1/data/plugins
INFO[08-12|17:57:35] Live Push Gateway initialization         logger=live.push_http
INFO[08-12|17:57:35] HTTP Server Listen                       logger=http.server address=[::]:3000 protocol=http subUrl= socket=
```

7. Cross verify in browser using IP Address of the Machine with **Port# 3000**



Not secure | 3.135.194.87:3000/login

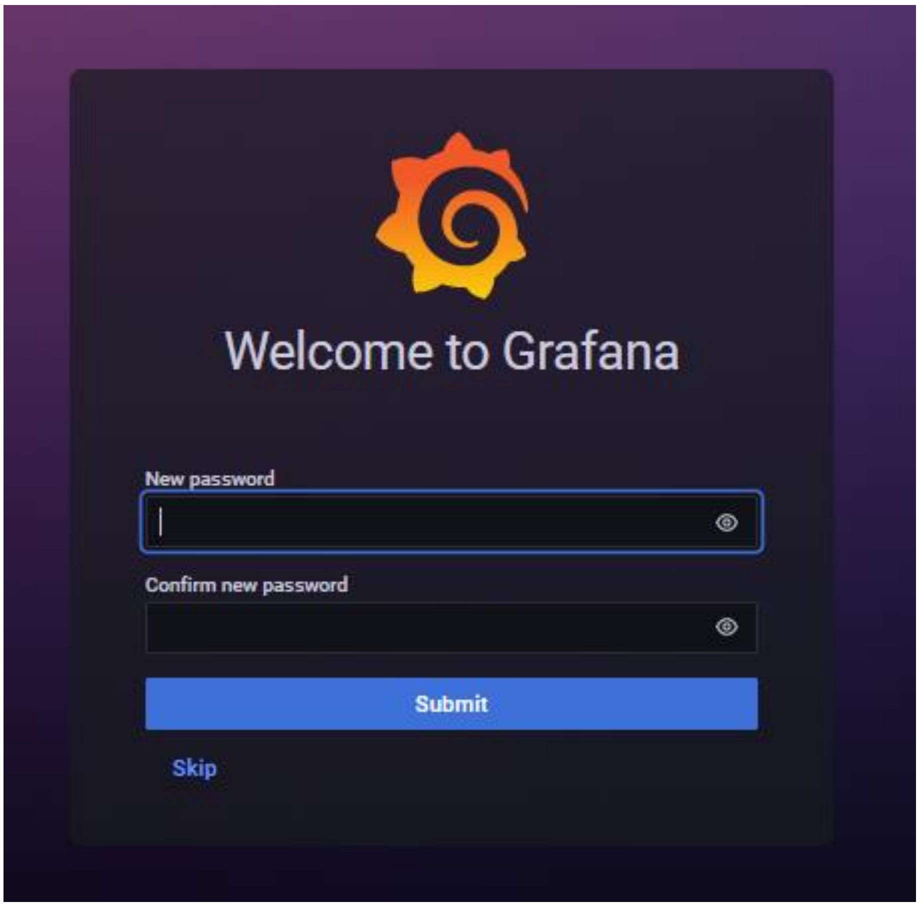
Welcome to Grafana

8. When you're trying to login **GRAFANA UI** for the FIRST time – Enter Username and Password as **ADMIN**

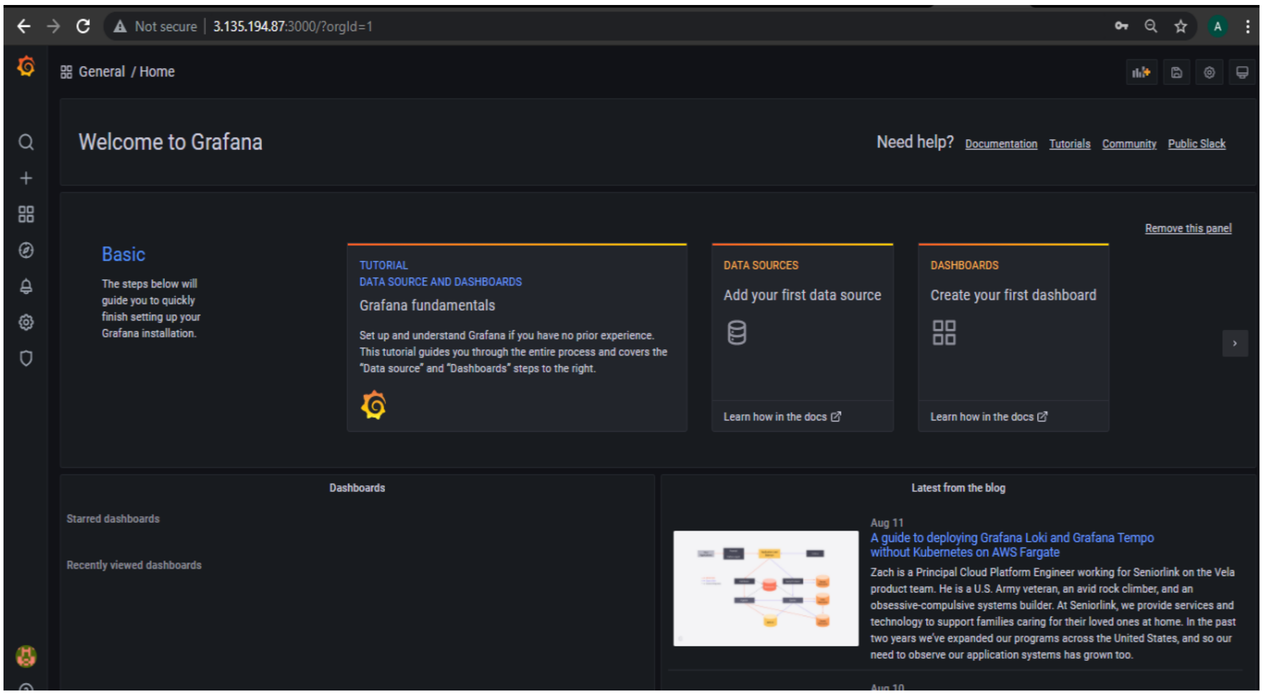
USERNAME: admin

PASSWORD: admin

Once above credentials is entered, it will prompt for a **NEW** Password:



GRAFANA UI is ready!!!



Not secure | 3.135.194.87:3000/?orgId=1

General / 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

Recently viewed dashboards

Latest from the blog

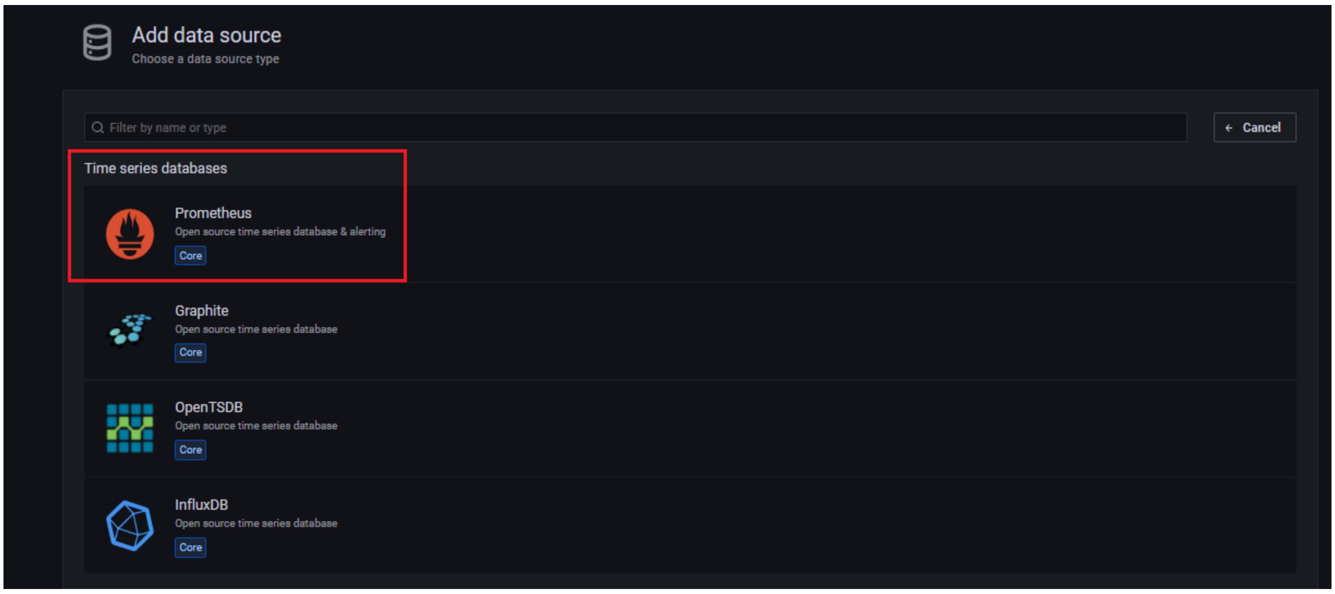
Aug 11
A guide to deploying Grafana Loki and Grafana Tempo without Kubernetes on AWS Fargate

Zach is a Principal Cloud Platform Engineer working for Seniorlink on the Vela product team. He is a U.S. Army veteran, an avid rock climber, and an obsessive-compulsive systems builder. At Seniorlink, we provide services and technology to support families caring for their loved ones at home. In the past two years we've expanded our programs across the United States, and so our need to observe our application systems has grown too.

Aug 10

ADDING PROMETHEUS TO GRAFANA

1. Add **DATASOURCE** as **PROMETHEUS**


Add data source
Choose a data source type

Time series databases

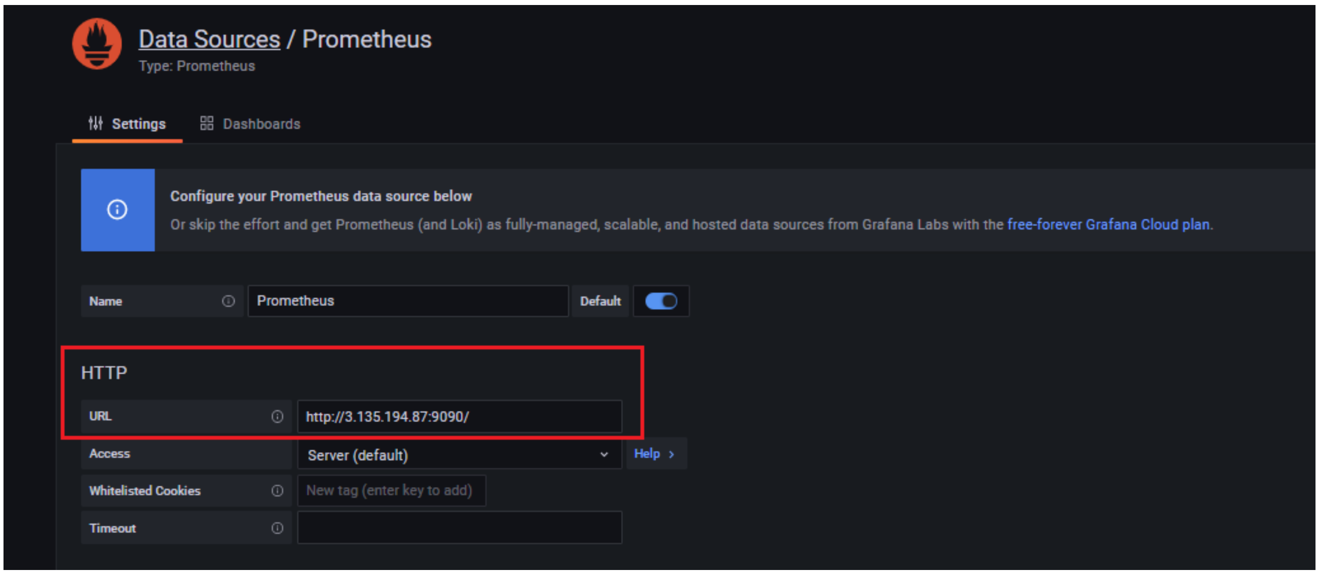
Prometheus
Open source time series database & alerting
Core

Graphite
Open source time series database
Core

OpenTSDB
Open source time series database
Core

InfluxDB
Open source time series database
Core

2. Add **PROMETHEUS** server with **PORT# 9090** as http://3.135.194.87:9090/

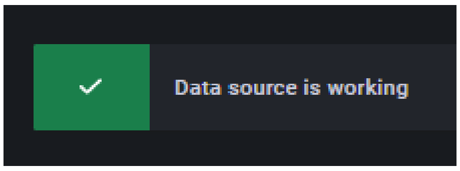

Data Sources / Prometheus
Type: Prometheus

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

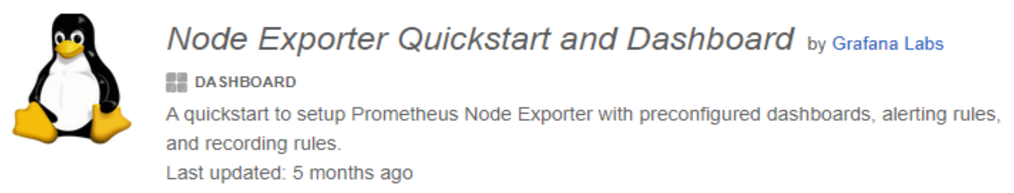
Name: Prometheus
Default: on

HTTP
URL: http://3.135.194.87:9090/
Access: Server (default)
Whitelisted Cookies: New tag (enter key to add)
Timeout:

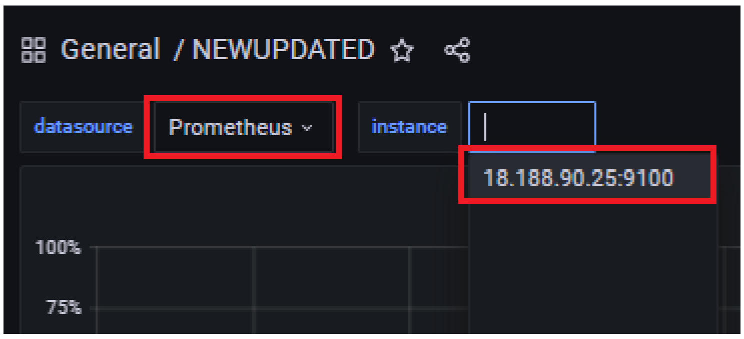
3. Save and TEST the connection – It should display as below


✓ Data source is working

4. Import DASHBOARD to Visualise the METRICS from https://grafana.com/grafana/dashboards


Node Exporter Quickstart and Dashboard by Grafana Labs
DASHBOARD
A quickstart to setup Prometheus Node Exporter with preconfigured dashboards, alerting rules, and recording rules.
Last updated: 5 months ago

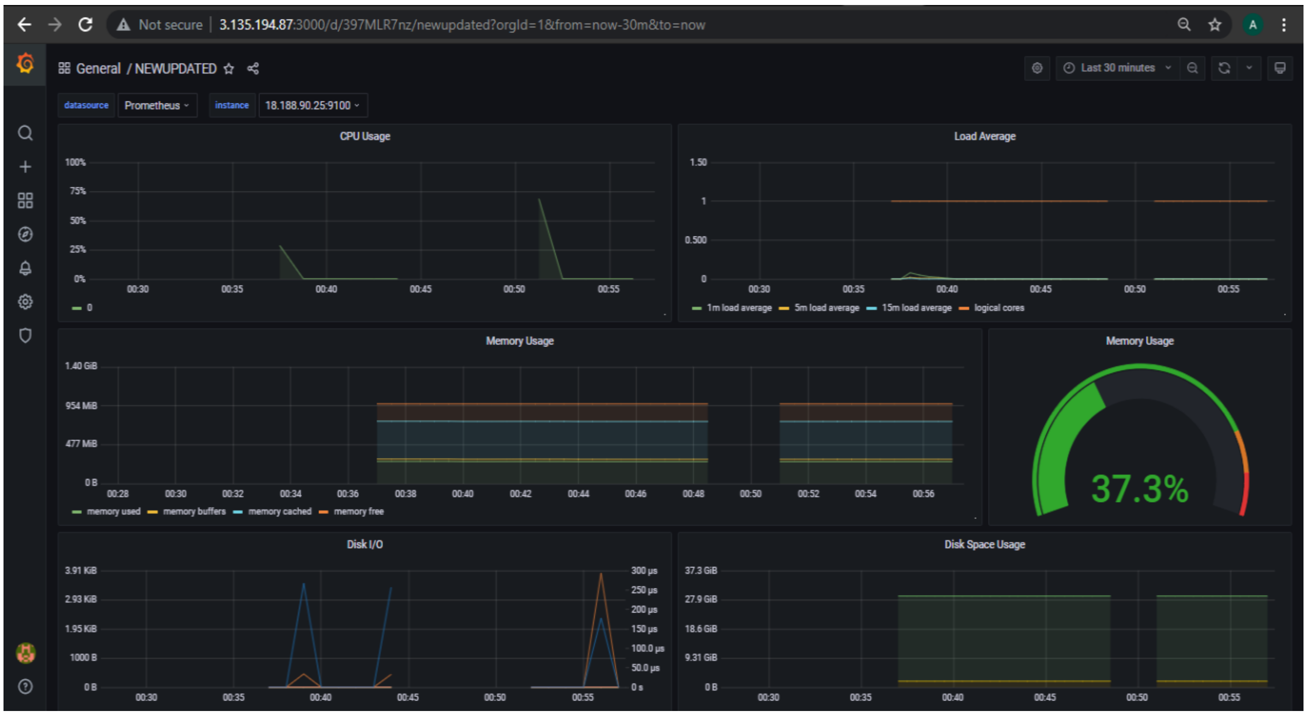
5. Once Dashboard is IMPORTED, check if the exact IP ADDRESS of NODE EXPORTER is popping up in the Dashboard DROP DOWN as below


General / NEWUPDATED

datasource: Prometheus

instance: 18.188.90.25:9100

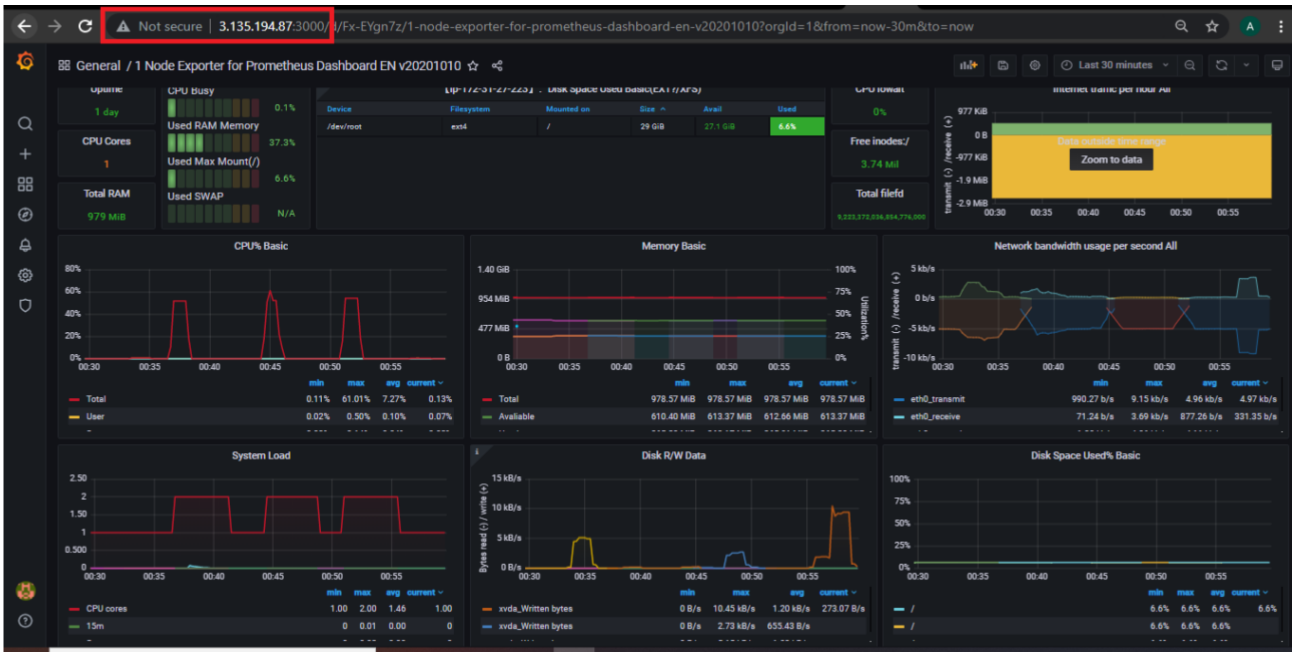
DASHBOARD IS CREATED:


Not secure | 3.135.194.87:3000/d/397MLR7nz/newupdated?orgId=1&from=now-30m&to=now

General / NEWUPDATED

datasource: Prometheus

instance: 18.188.90.25:9100


Not secure | 3.135.194.87:3000/d/Fx-EYgn7z/1-node-exporter-for-prometheus-dashboard-en-v20201010?orgId=1&from=now-30m&to=now

General / 1 Node Exporter for Prometheus Dashboard EN v20201010

Uptime: 1 day
CPU Cores: 1
Total RAM: 979 MiB
CPU Busy: 0.1%
Used RAM Memory: 37.3%
Used Max Mount(/): 6.6%
Used SWAP: N/A
CPU iowait: 0%
Free inodes:/: 3.74 Mil

Device	Filesystem	Mounted on	Size	Avail	Used
/dev/root	ext4	/	29 GiB	27.1 GiB	6.6%

CPU% Basic	min	max	avg	current
Total	0.11%	61.01%	7.27%	0.13%
User	0.02%	0.50%	0.10%	0.07%

Memory Basic	min	max	avg	current
Total	978.57 MiB	978.57 MiB	978.57 MiB	978.57 MiB
Available	610.40 MiB	613.37 MiB	612.66 MiB	613.37 MiB

Network bandwidth usage per second All	min	max	avg	current
eth0_transmit	990.27 b/s	9.15 kb/s	4.96 kb/s	4.97 kb/s
eth0_receive	71.24 b/s	3.69 kb/s	877.26 b/s	331.35 b/s

System Load	min	max	avg	current
CPU cores	1.00	2.00	1.46	1.00
15m	0	0.01	0.00	0

Disk R/W Data	min	max	avg	current
xvda_Written bytes	0 B/s	10.45 kB/s	1.20 kB/s	273.07 B/s
xvda_Written bytes	0 B/s	2.73 kB/s	655.43 B/s	

Disk Space Used% Basic	min	max	avg	current
/	6.6%	6.6%	6.6%	6.6%
/	6.6%	6.6%	6.6%	