PROMETHEUS GRAFANA CONFIGURATION

Followed Below document

NOTE: - https://devops4solutions.com/monitoring-using-prometheus-and-grafana-on-aws-ec2/

Section: A System logs Monitoring

Requirements Monit0r the servers in main server: -

- Prometheus
- Grafana
- Node_exporter

Requirements for target servers: -

System metrics monitoring -Node_exporter
Docker metrics monitoring -Node_exporter

- Jenkins metrics monitoring - Node_exporter and Install Prometheus plugin in Jenkins dashboard

Introduction: -

Prometheus:

Prometheus is an open-source technology designed to provide monitoring and alerting functionality for cloud-native environments, including Kubernetes. It can collect and store metrics as time-series data, recording information with a timestamp.

Prometheus query language (PromQL) to filter, aggregate, ingest, and query millions of unique time series metrics from your self-managed Kubernetes clusters. Automatically scale as your ingestion and query needs grow and maintain consistent response times for large container deployments.

Step1: - Installation of the Prometheus

A) Create a new user and add new directories for Prometheus

```
  $sudo useradd --no-create-home prometheus
$sudo mkdir /etc/prometheus
$sudo mkdir /var/lib/prometheus
```

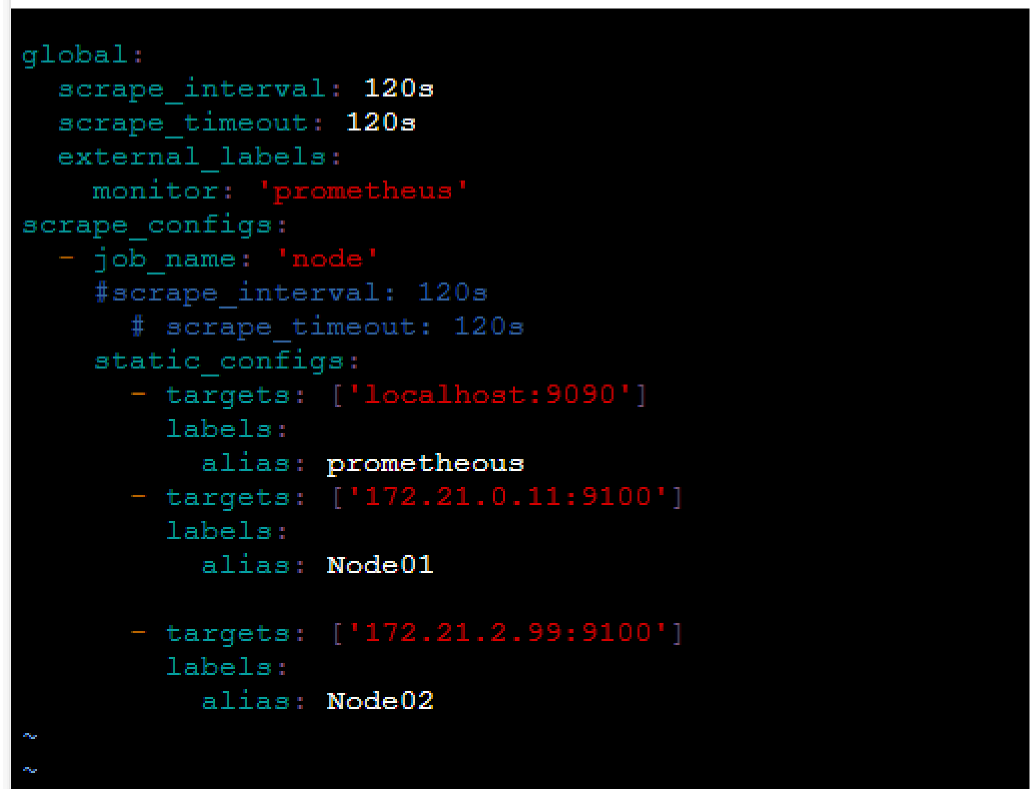
B) Download the Prometheus, extract it and put it in `/usr/local/bin` folder

`Wget`**https://github.com/prometheus/prometheus/releases/download/v2.23.0/prometheus-2.23.0.linux-amd64.tar.gz** **(or)** Head to https://prometheus.io/download/ and download the latest binary for the Prometheus

```
 tar -xvf prometheus-2.23.0.linux-amd64.tar.gz
sudo cp prometheus-2.23.0.linux-amd64/prometheus /usr/local/bin
sudo cp prometheus-2.23.0.linux-amd64/promtool /usr/local/bin
sudo cp -r prometheus-2.23.0.linux-amd64/consoles /etc/prometheus/

sudo cp -r prometheus-2.23.0.linux-amd64/console_libraries /etc/prometheus

sudo cp prometheus-2.23.0.linux-amd64/promtool /usr/local/bin/
```

C) configure Prometheus to monitor itself using yaml file

Create a **`prometheus.yml`** file at **`/etc/prometheus/prometheus.yml`** with the below content



```
global:
  scrape_interval: 120s
  scrape_timeout: 120s
  external_labels:
    monitor: 'prometheus'
scrape_configs:
  - job_name: 'node'
    #scrape_interval: 120s
      # scrape_timeout: 120s
    static_configs:
      - targets: ['localhost:9090']
        labels:
          alias: prometheous
      - targets: ['172.21.0.11:9100']
        labels:
          alias: Node01

      - targets: ['172.21.2.99:9100']
        labels:
          alias: Node02
~
~
```

Step2:- Security Groups Configuration

Ensure ports are enabled:

Port **`9090`**—Prometheus Server

Port **`9100`**—Prometheus Node Exporter

Port **`3002`**—Grafana

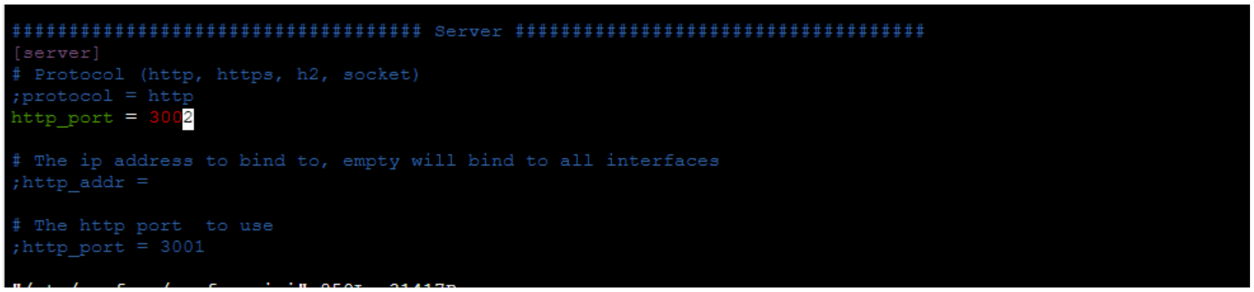
Port **9323**=--docker

Port **8080--**Jenkins(poc account)

To change the default Grafana GUI port number, you need to modify the configuration file. Here's how you can do it:

1. Locate the Grafana configuration file. The default location is **`/etc/grafana/grafana.ini`** on Linux

sudo vi /etc/grafana/grafana.ini



```
#################################### Server ####################################
[server]
# Protocol (http, https, h2, socket)
;protocol = http
http_port = 3002

# The ip address to bind to, empty will bind to all interfaces
;http_addr =

# The http port  to use
;http_port = 3001
```

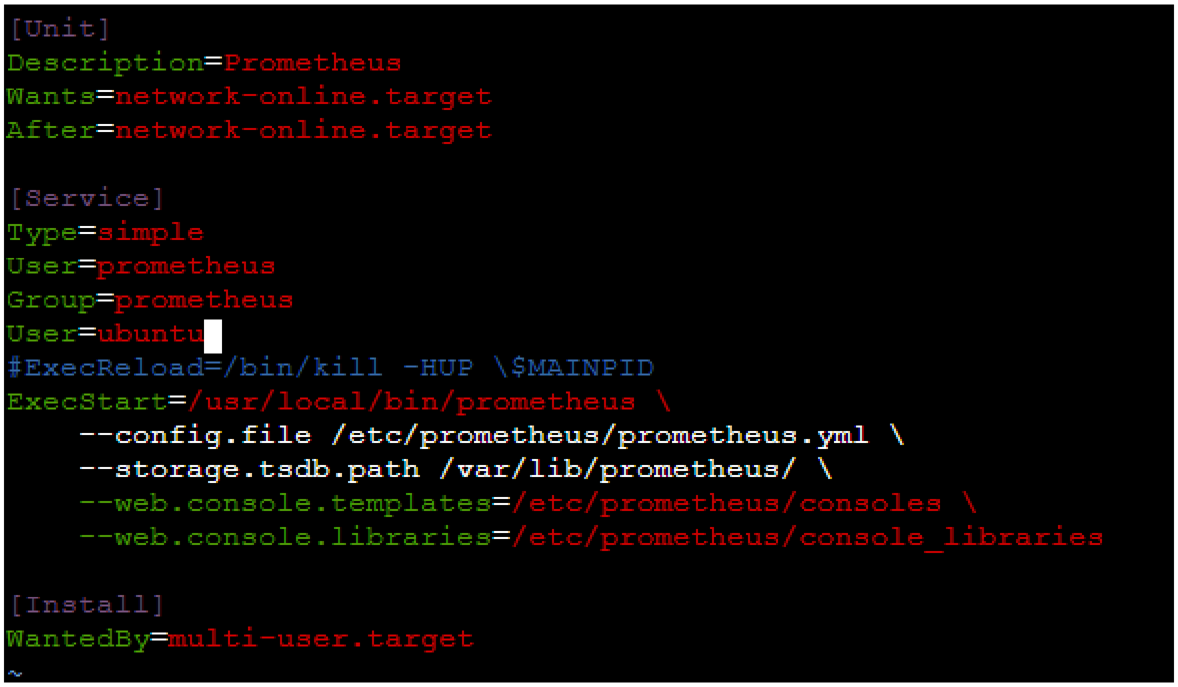
2. Restart the service after changing the ini file

sudo systemctl restart grafana-server

Step3:- **Prometheus as a Service file to server restart service automatically**

Now we want to run the Prometheus as a Service so that in case of server restart service will come automatically**.**

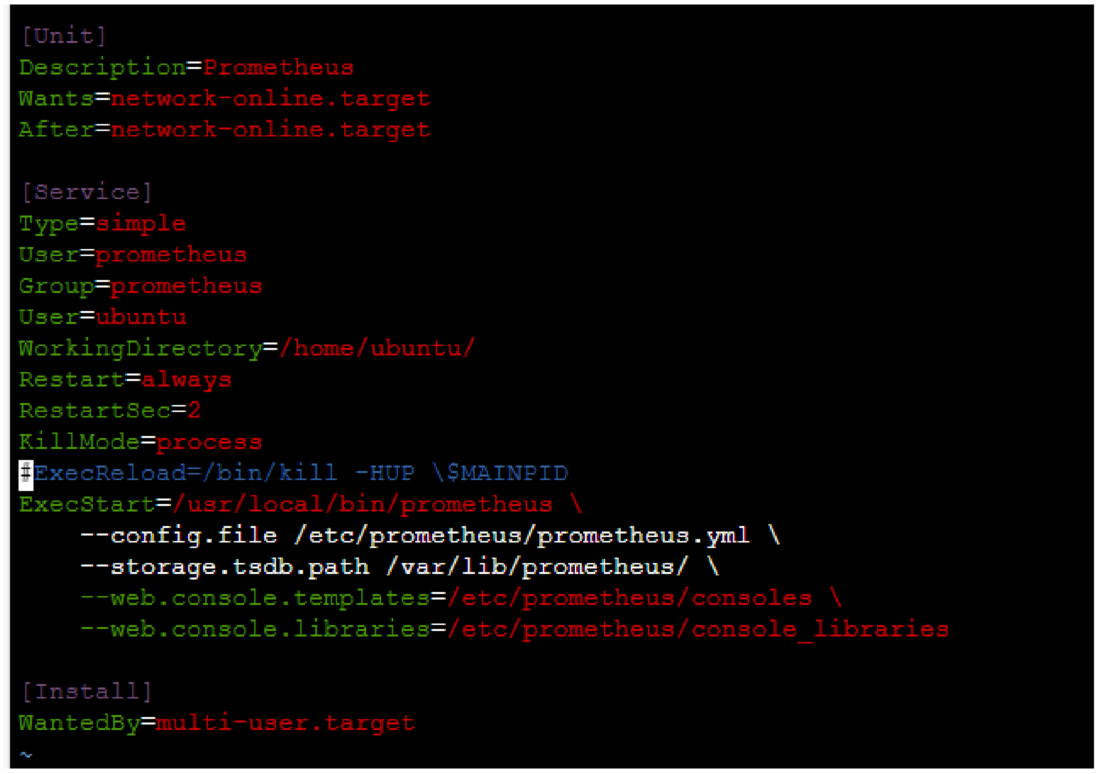
Let's create a file **`/etc/systemd/system/prometheus.service`** with the below content:



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

[Service]
Type=simple
User=prometheus
Group=prometheus
User=ubuntu
#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

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

NOTE:- Getting error like failed the prometheous server when stop and start or screen once locked

So, we added below script



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

[Service]
Type=simple
User=prometheus
Group=prometheus
User=ubuntu
WorkingDirectory=/home/ubuntu/
Restart=always
RestartSec=2
KillMode=process
#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

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

Step3: - Change the ownerships

Change the ownership of all folders and files which we have created to the user which we have created in the first step

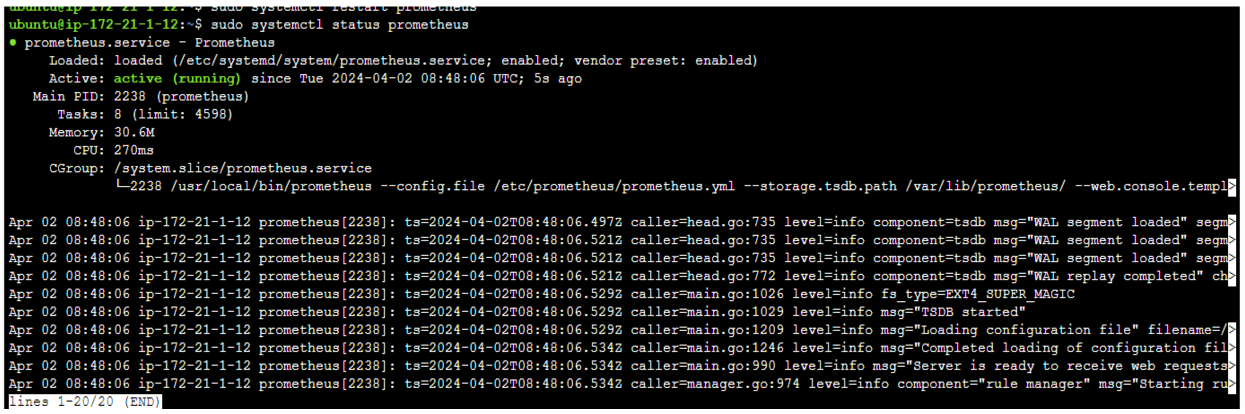
```
sudo chown prometheus:prometheus /etc/prometheus
sudo chown prometheus:prometheus /usr/local/bin/prometheus
sudo chown prometheus:prometheus /usr/local/bin/promtool

sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
sudo chown -R prometheus:prometheus /var/lib/prometheus
```

Step4:- configure the service and start and run continiously

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

Step5:- Checking the service and status of prometheous

```
sudo systemctl status prometheus
```

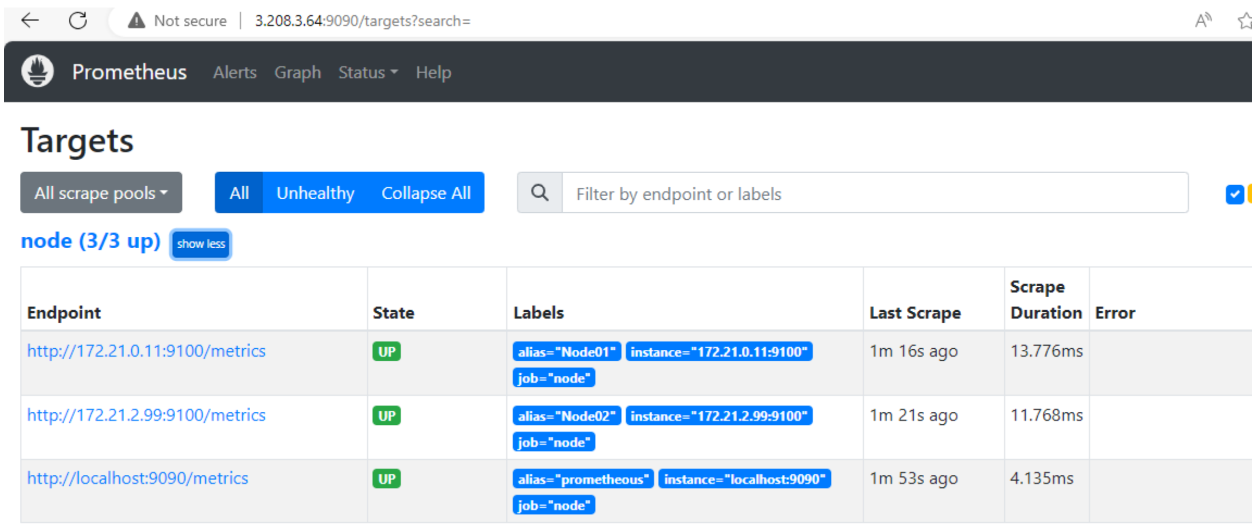


```
ubuntu@ip-172-21-1-12:~$ sudo systemctl restart prometheus
ubuntu@ip-172-21-1-12:~$ sudo systemctl status prometheus
● prometheus.service - Prometheus
     Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-04-02 08:48:06 UTC; 5s ago
   Main PID: 2238 (prometheus)
      Tasks: 8 (limit: 4598)
     Memory: 30.6M
        CPU: 270ms
     CGroup: /system.slice/prometheus.service
             └─2238 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --storage.tsdb.path /var/lib/prometheus/ --web.console.templ>

Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.497Z caller=head.go:735 level=info component=tsdb msg="WAL segment loaded" segm>
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.521Z caller=head.go:735 level=info component=tsdb msg="WAL segment loaded" segm>
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.521Z caller=head.go:735 level=info component=tsdb msg="WAL segment loaded" segm>
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.521Z caller=head.go:772 level=info component=tsdb msg="WAL replay completed" ch>
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.529Z caller=main.go:1026 level=info fs_type=EXT4_SUPER_MAGIC
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.529Z caller=main.go:1029 level=info msg="TSDB started"
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.529Z caller=main.go:1209 level=info msg="Loading configuration file" filename=/>
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.534Z caller=main.go:1246 level=info msg="Completed loading of configuration fil>
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.534Z caller=main.go:990 level=info msg="Server is ready to receive web requests>
Apr 02 08:48:06 ip-172-21-1-12 prometheus[2238]: ts=2024-04-02T08:48:06.534Z caller=manager.go:974 level=info component="rule manager" msg="Starting ru>
lines 1-20/20 (END)
```

Now open it on the browser using below url:

Type in browser **<ipaddress>:9090** to get the prometheous dashboard

Its show the prometheous configured successfully if **state=up**



Targets

node (3/3 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.21.0.11:9100/metrics	UP	alias="Node01" instance="172.21.0.11:9100" job="node"	1m 16s ago	13.776ms	
http://172.21.2.99:9100/metrics	UP	alias="Node02" instance="172.21.2.99:9100" job="node"	1m 21s ago	11.768ms	
http://localhost:9090/metrics	UP	alias="prometheous" instance="localhost:9090" job="node"	1m 53s ago	4.135ms	

Node Exporter: -

The node exporter is an open-source technology which enables you to measure various machine resources such as memory, disk and CPU utilization.

The Node Exporter is an agent that gathers system metrics and exposes them in a format which can be ingested by Prometheus. The Node Exporter is a project that is maintained through the Prometheus project.

"To monitor your servers, you need to install the node exporter on all your target machines, which is like a monitoring agent on all the servers."

Step1: - Install Node Exporter

A) Create a node exporter user

```
sudo useradd -rs /bin/false node_exporter
```

B) Download the Node Exporter, extract it and put it in `/usr/local/bin` folder

Head to https://prometheus.io/download/ and download the latest binary for the node exporter

(or)

```
wget
https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node_exporter-0.18.1.linux-amd64.tar.gz
```

```
tar xvzf node_exporter-0.18.1.linux-amd64.tar.gz

sudo useradd -rs /bin/false node_exporter

Sudo cp node_exporter-0.18.1.linux-amd64/node_exporter /usr/local/bin
```

Step2:- set the correct permissions to binary file

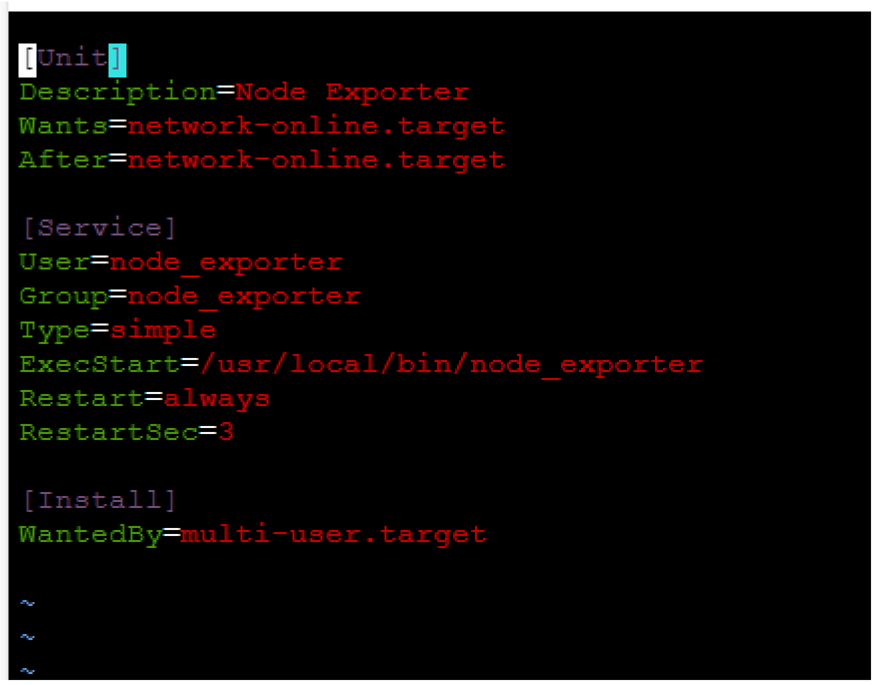
```
Sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

Step3:- create a Node_exporter new service file

Navigate to /etc/systemd/system

```
cd /etc/systemd/system
```

```
sudo vi node_exporter.service
```



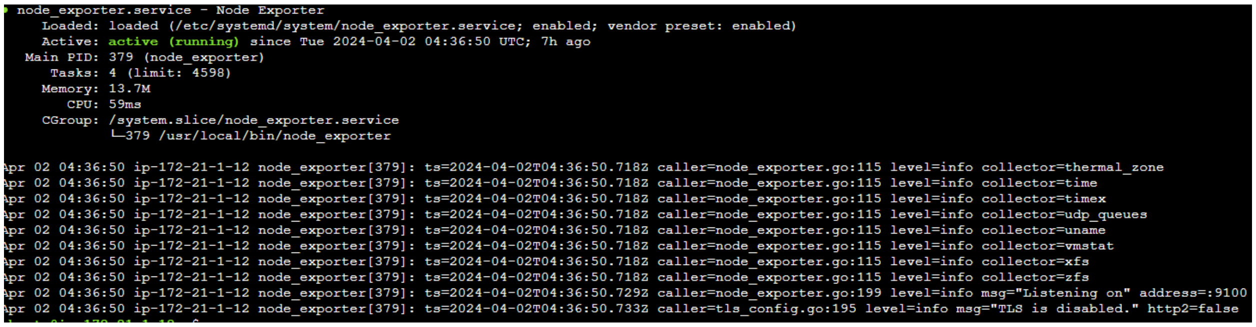
```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target

~
~
~
```

Step4:- configure the service and start and run continiously

```
sudo systemctl daemon-reload

sudo systemctl start node_exporter

sudo systemctl enable node_exporter
```

Step5:- Checking status of Node_exporter and verifying

```
sudo systemctl status node_exporter.service
```



```
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-04-02 04:36:50 UTC; 7h ago
   Main PID: 379 (node_exporter)
      Tasks: 4 (limit: 4598)
     Memory: 13.7M
        CPU: 59ms
     CGroup: /system.slice/node_exporter.service
             └─379 /usr/local/bin/node_exporter

Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=thermal_zone
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=time
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=timex
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=udp_queues
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=uname
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=vmstat
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=xfs
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.718Z caller=node_exporter.go:115 level=info collector=zfs
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.729Z caller=node_exporter.go:199 level=info msg="Listening on" address=:9100
Apr 02 04:36:50 ip-172-21-1-12 node_exporter[379]: ts=2024-04-02T04:36:50.733Z caller=tls_config.go:195 level=info msg="TLS is disabled." http2=false
```

Verify that your node exporter is correctly up and running with a simple curl command

curl **http://localhost:9100/metrics**

Grafana: -

Grafana is an open-source analytics and interactive visualization web application.

Grafana is a powerful tool for DevOps teams, helping to monitor, visualize, and understand the vast amount of data generated by their systems and applications. Here's what you need to know: Central Monitoring: Grafana consolidates data from various sources like Prometheus, Loki, and more into customizable dashboards.

Grafana is a tool used to analyze and visualize data. However, this data would have to be stored somewhere in order for Grafana to access and display it. These databases are what we refer to as data sources, and a Grafana datasource is simply any database from which it can pull data.

Step1:- Install Grafana

A)Create a Grafana user

```
sudo apt-get install -y adduser libfontconfig1
```

B)Download the Grafana, extract

To install Grafana, head over to https://grafana.com/grafana/download and download the latest binaries available for you

```
wget https://dl.grafana.com/oss/release/grafana_7.3.4_amd64.deb
```

```
        sudo dpkg -i grafana_7.3.4_amd64.deb
```

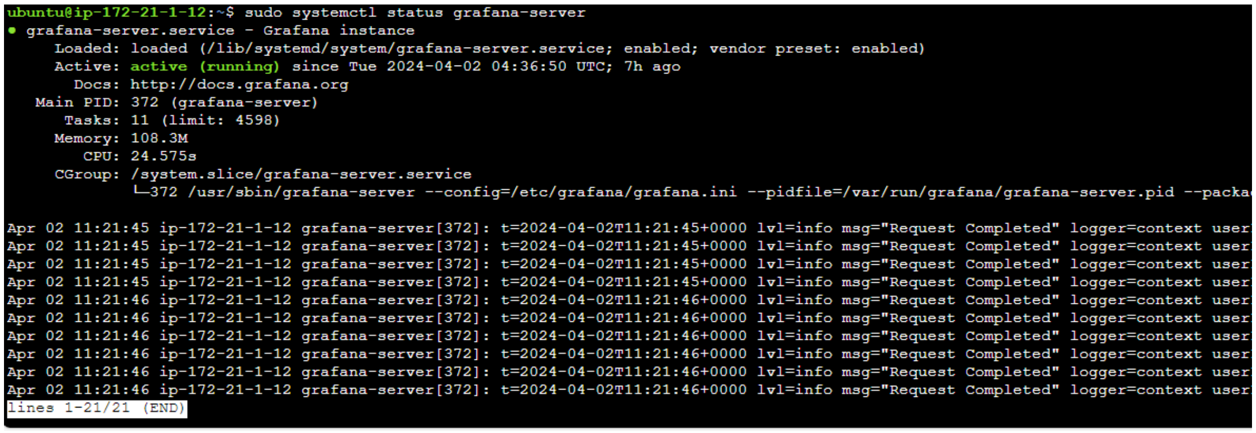
Step2:- configure the service and start and run continiously

```
      sudo systemctl daemon-reload
      sudo systemctl start grafana-server
      sudo systemctl enable grafana-server.service
```

Step3:- Checking status of `Grafana-server`

```
     sudo systemctl status grafana-server
```

NOTE: while checking the status we get error like failed means please refer the below link
which was not installed properly packages
https://community.grafana.com/t/unable-to-install-grafana-from-apt-repository-on-debian-bookworm/119040



```
ubuntu@ip-172-21-1-12:~$ 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 Tue 2024-04-02 04:36:50 UTC; 7h ago
       Docs: http://docs.grafana.org
   Main PID: 372 (grafana-server)
      Tasks: 11 (limit: 4598)
     Memory: 108.3M
        CPU: 24.575s
     CGroup: /system.slice/grafana-server.service
             └─372 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --pidfile=/var/run/grafana/grafana-server.pid --packa

Apr 02 11:21:45 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:45+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:45 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:45+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:45 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:45+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:45 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:45+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:46 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:46+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:46 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:46+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:46 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:46+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:46 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:46+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:46 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:46+0000 lvl=info msg="Request Completed" logger=context user
Apr 02 11:21:46 ip-172-21-1-12 grafana-server[372]: t=2024-04-02T11:21:46+0000 lvl=info msg="Request Completed" logger=context user
lines 1-21/21 (END)
```

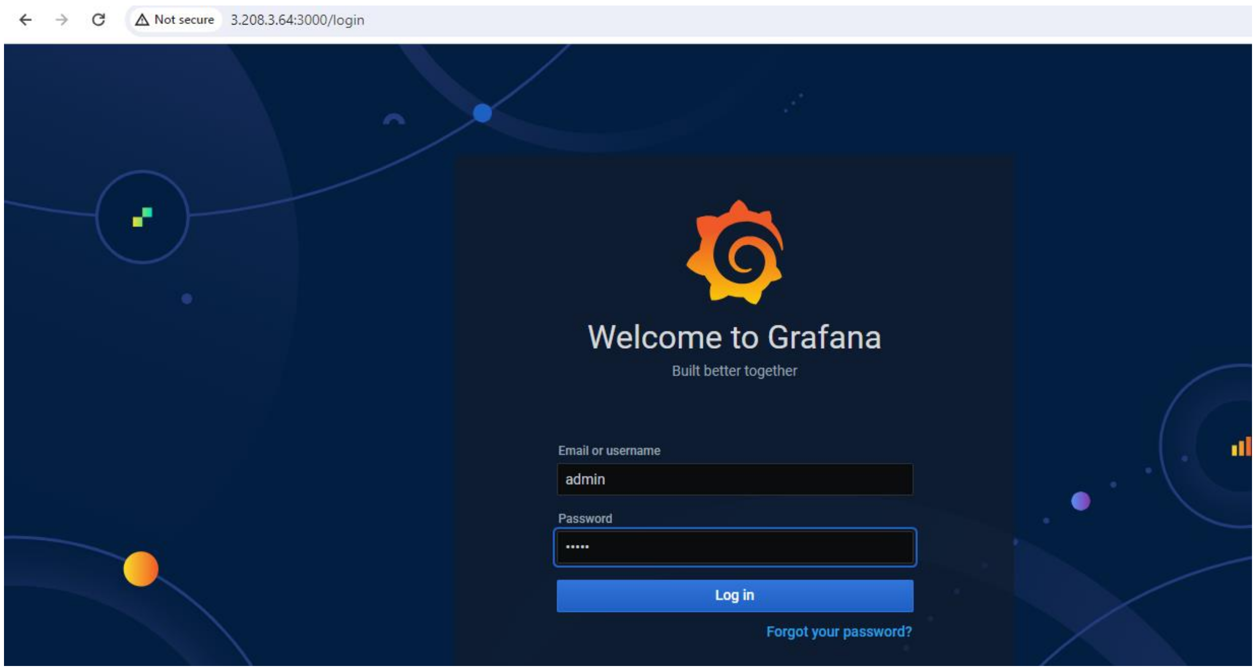
step6: - Grafana Dashboard Login

Now open it on the browser using below url:

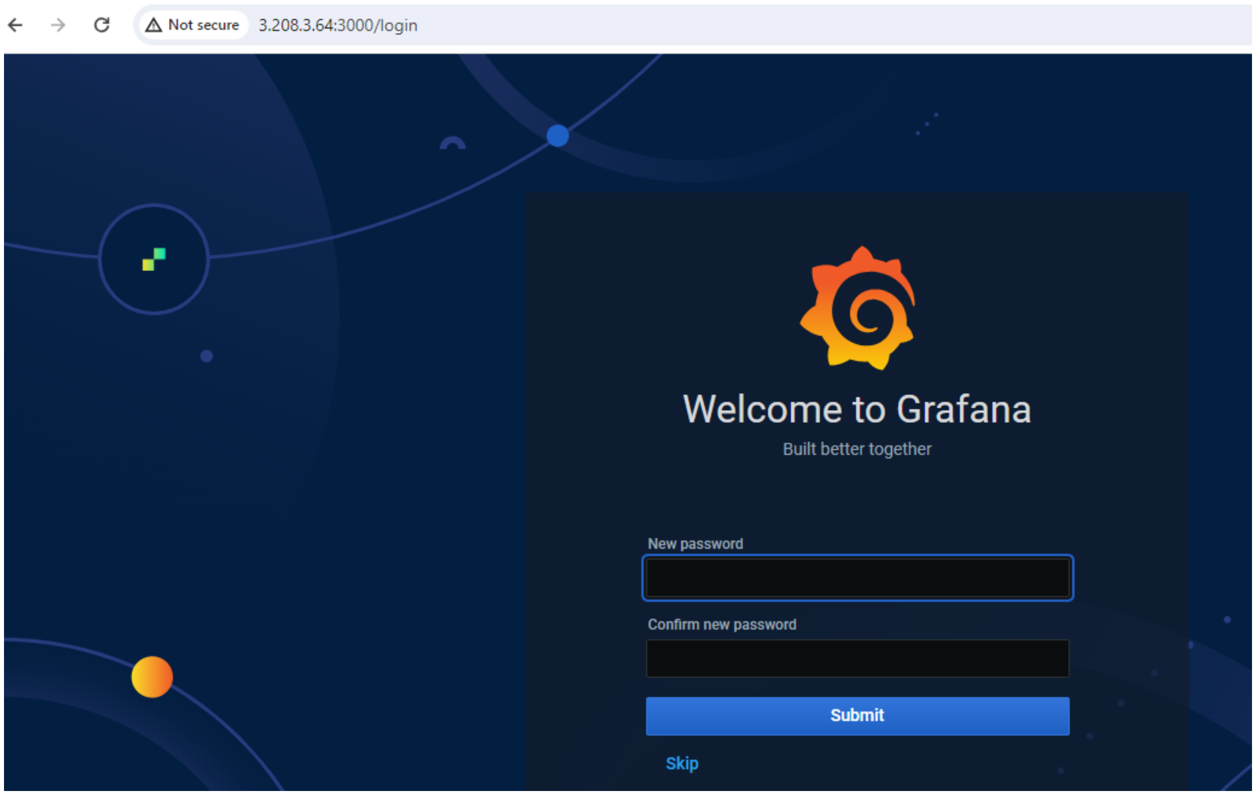
http://publicip:3000

Login with default credentials username: **`admin`** and password **`admin`**

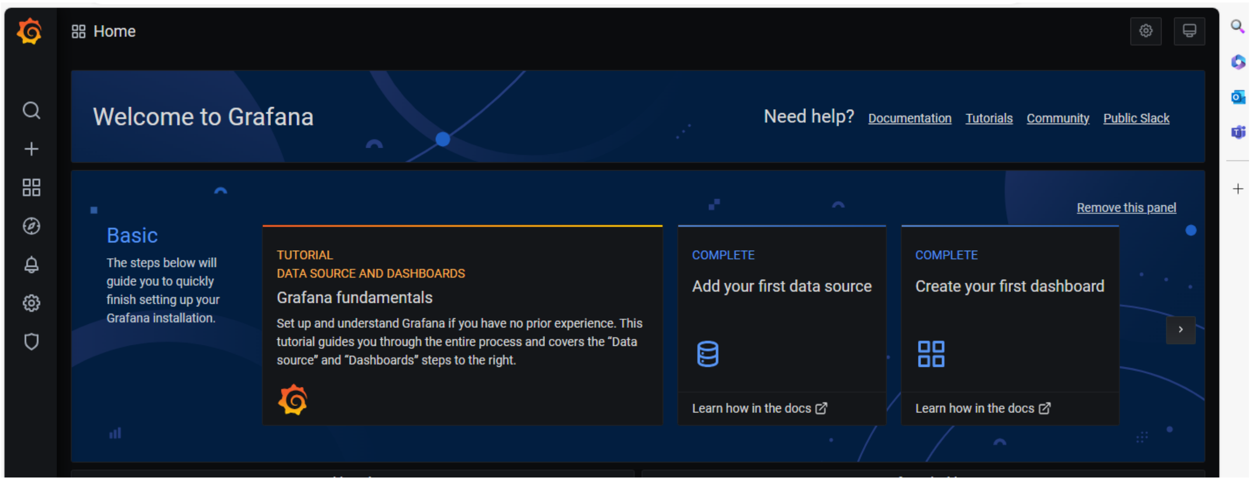
NOTE: - Make sure that port **`3000`** is open for this instance.



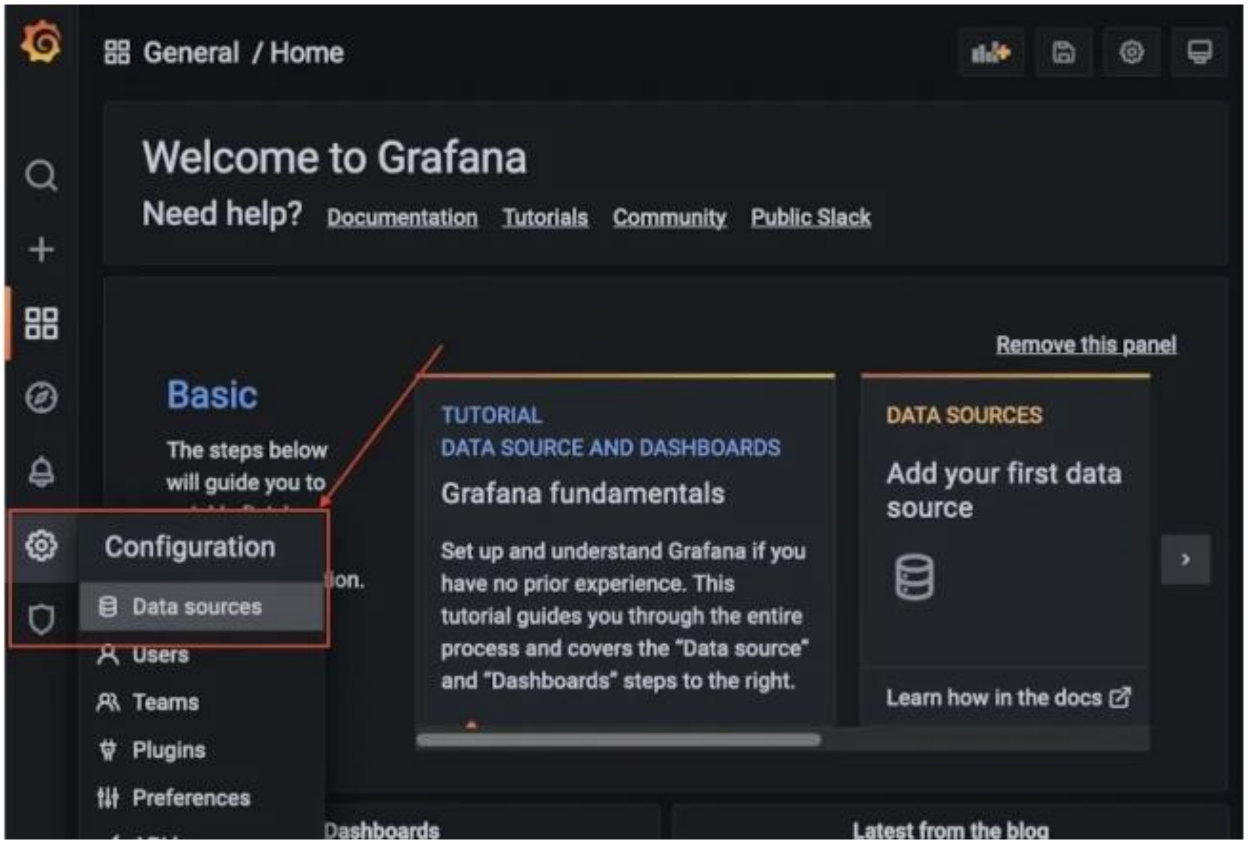
After Login it will ask for Password changes



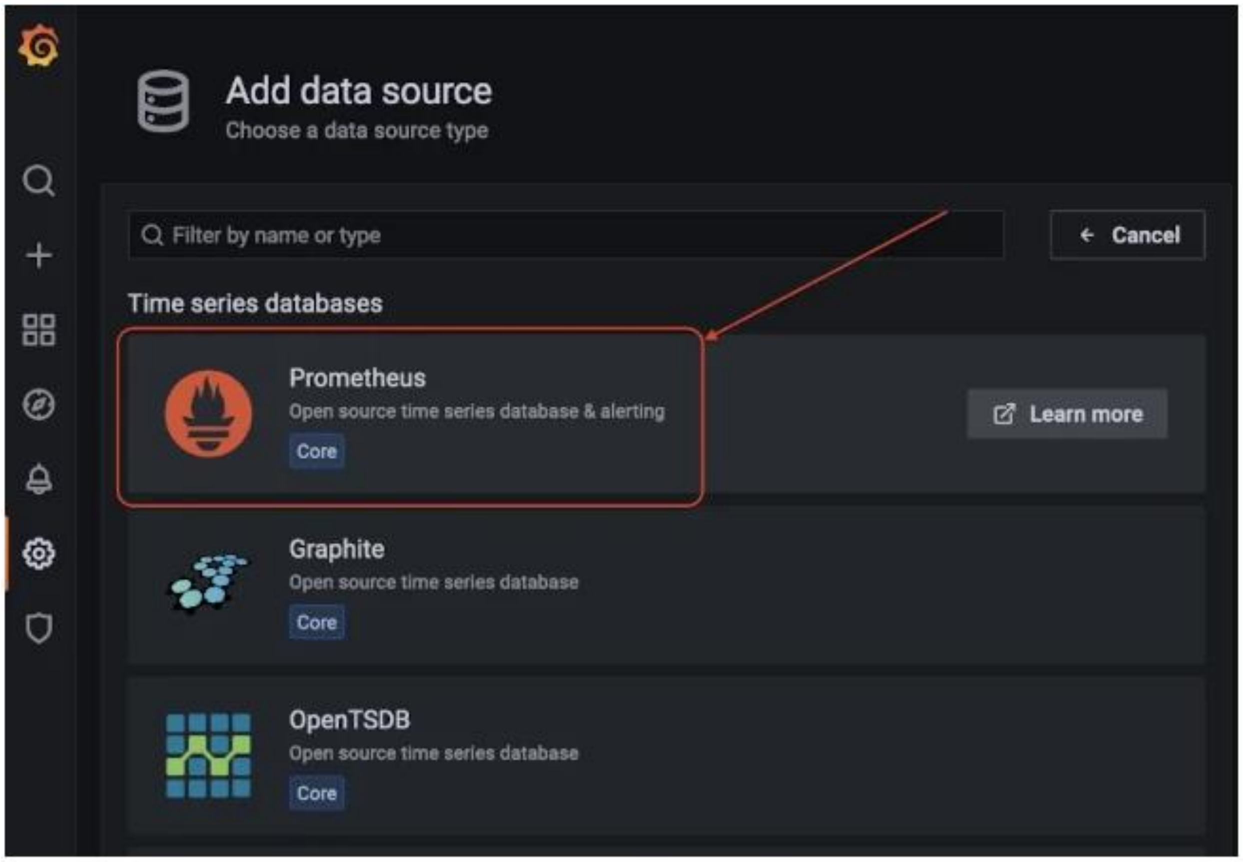
A) Add Prometheus DataSource


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.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your first dashboard
Learn how in the docs

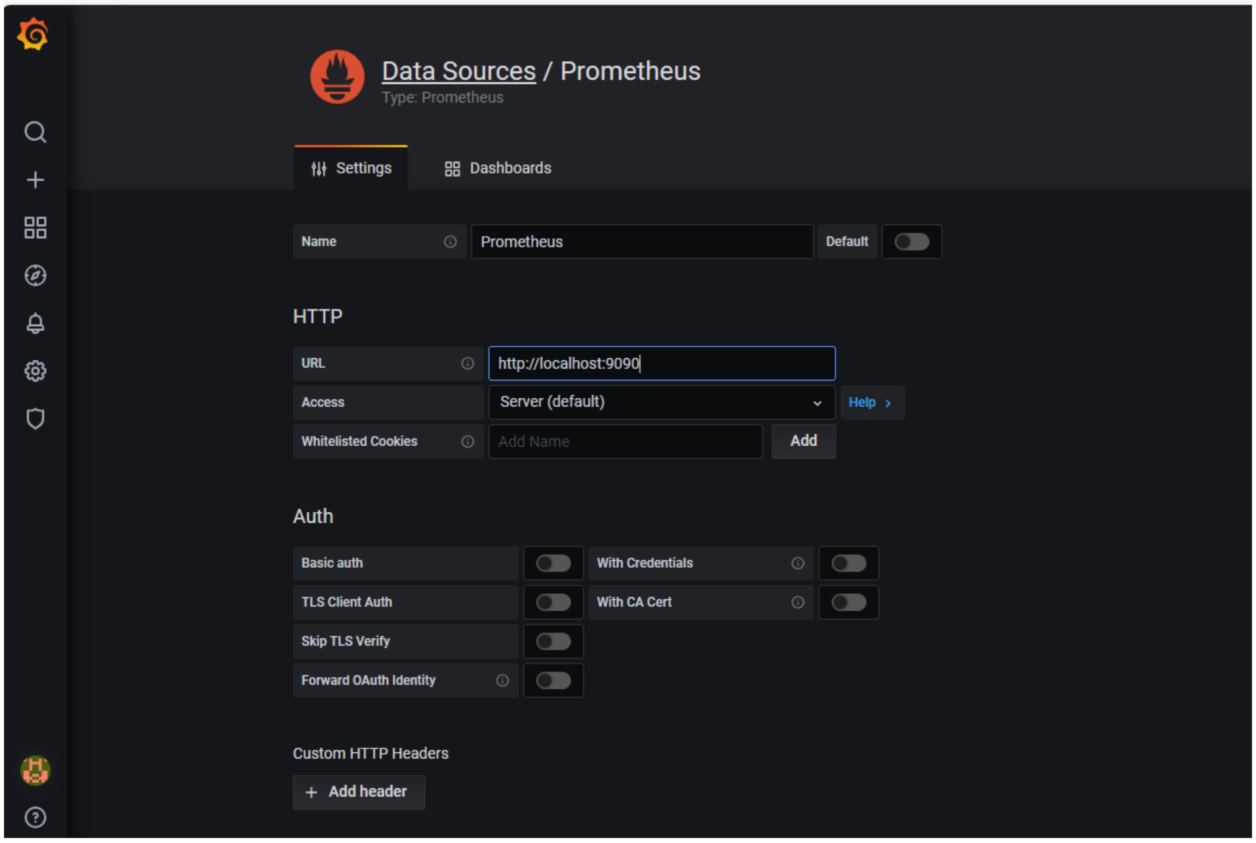
Click on Setting ->datasources


General / Home
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Remove this panel
Basic
The steps below will guide you to
Configuration
Data sources
Users
Teams
Plugins
Preferences
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
Latest from the blog

Select the *Prometheus* as preferred data source -


Add data source
Choose a data source type
Filter by name or type
Cancel
Time series databases
Prometheus
Open source time series database & alerting
Core
Learn more
Graphite
Open source time series database
Core
OpenTSDB
Open source time series database
Core

Enter the `hostname or IP address` of the prometheus server



Data Sources / Prometheus
Type: Prometheus

Name: Prometheus — Default

HTTP

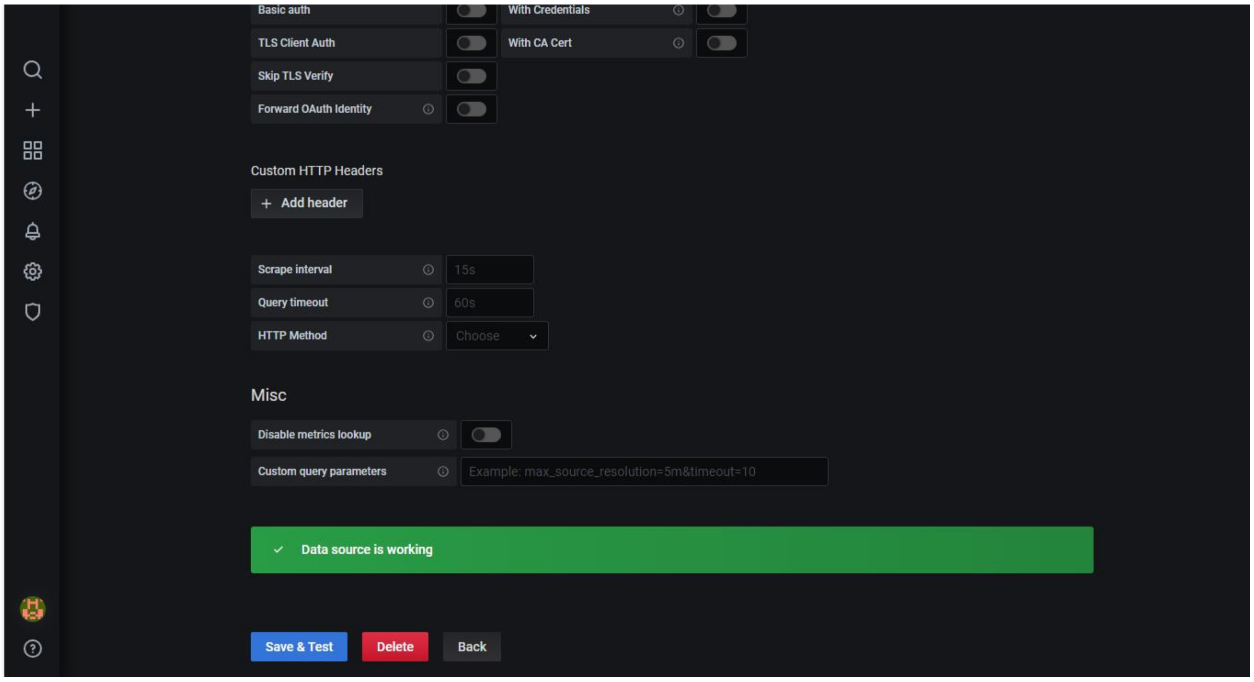
URL: http://localhost:9090
Access: Server (default) — Help
Whitelisted Cookies: Add Name — Add

Auth

Basic auth — With Credentials
TLS Client Auth — With CA Cert
Skip TLS Verify
Forward OAuth Identity

Custom HTTP Headers

Click on **SAVE & TEST**



Basic auth — With Credentials
TLS Client Auth — With CA Cert
Skip TLS Verify
Forward OAuth Identity

Custom HTTP Headers

Scrape interval: 15s
Query timeout: 60s
HTTP Method: Choose

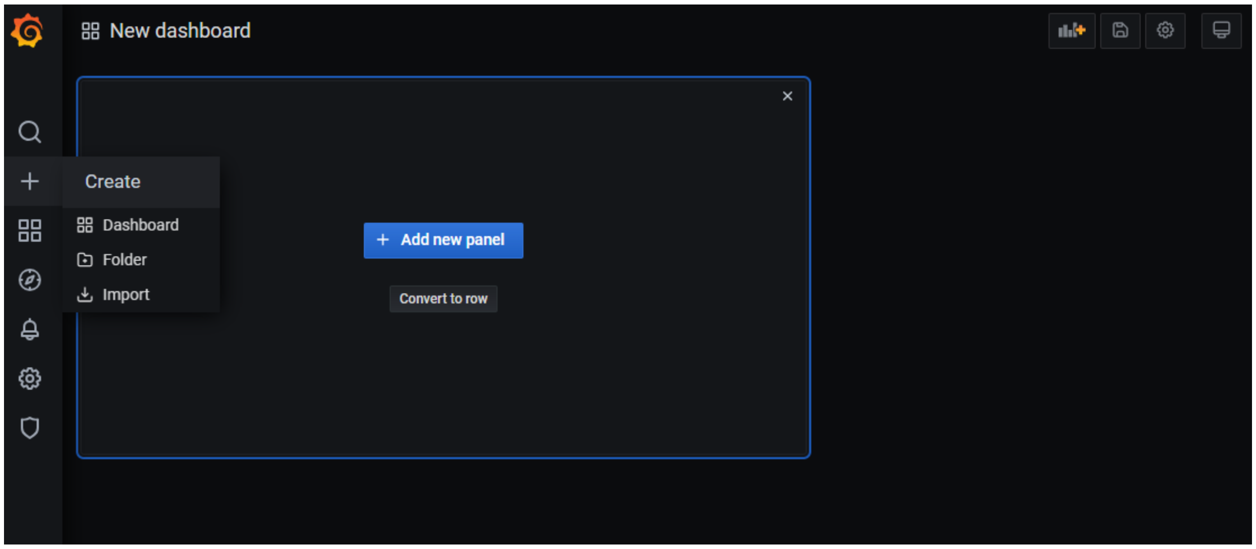
Misc

Disable metrics lookup
Custom query parameters: Example: max_source_resolution=5m&timeout=10

✓ Data source is working

B) **Importing the dashboard**

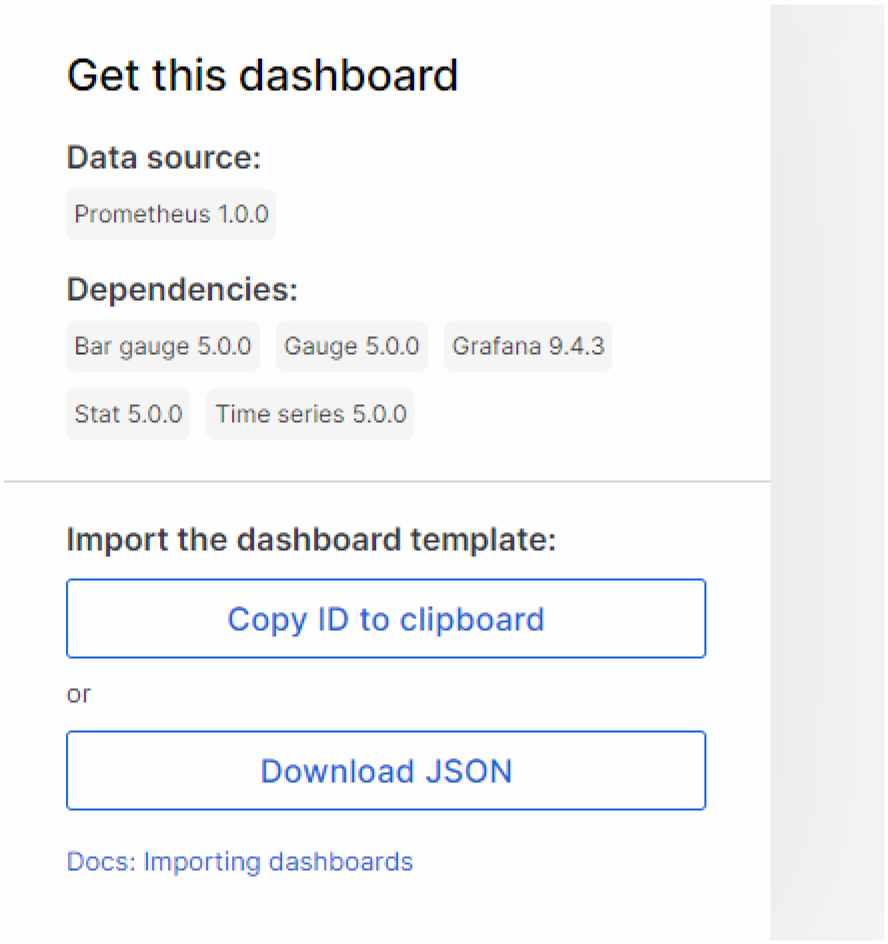
- **CLICK on add(+) symbol to import or create dashboards**



- **Import Grafana Dashboard from Grafana Labs**

Now after settings the data source we can import pre-existing opensource dashboard from Grafana Labs using the Dashboard ID.

Goto Grafana Dashboard search some sample dashboard and download the json file



Get this dashboard

Data source:

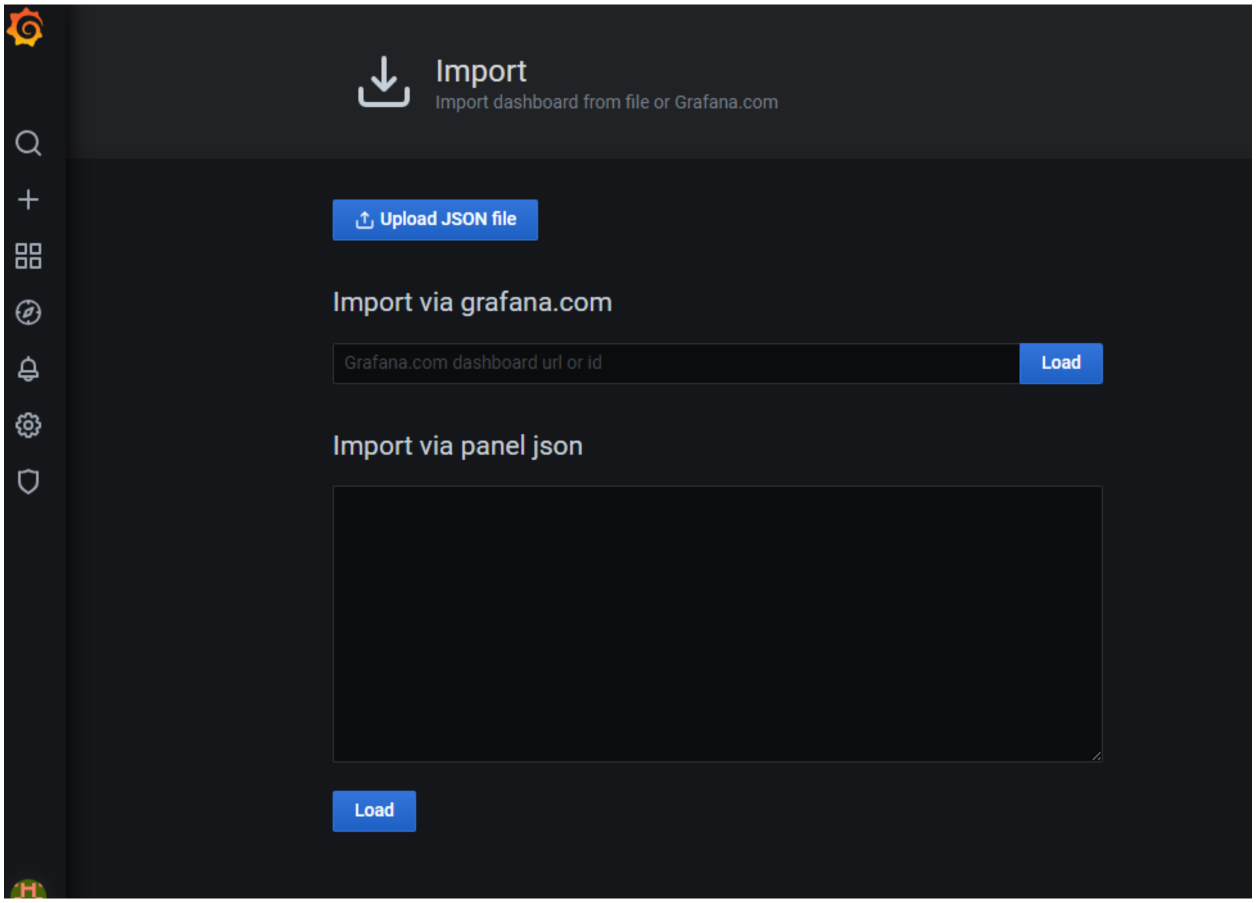
Prometheus 1.0.0

Dependencies:

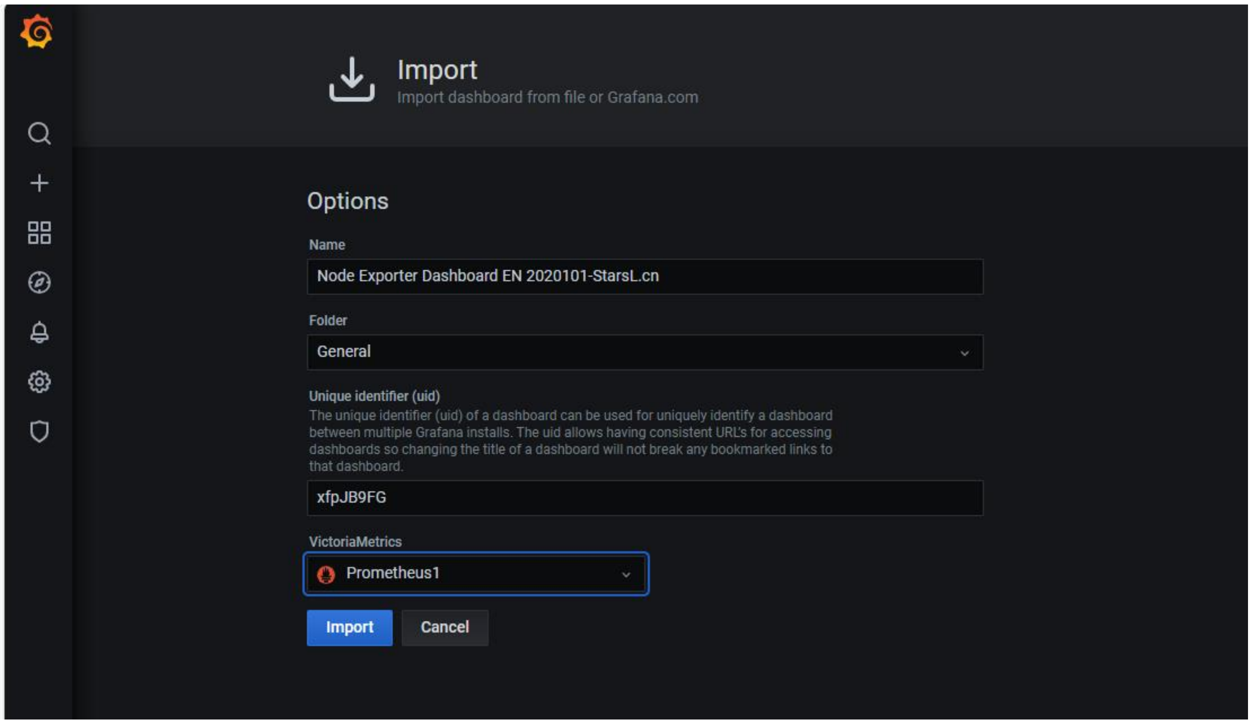
Bar gauge 5.0.0 Gauge 5.0.0 Grafana 9.4.3 Stat 5.0.0 Time series 5.0.0

Import the dashboard template:

Import json file click on **upload json file**


Import
Import dashboard from file or Grafana.com

Select the downloaded **Json** file and **DataSource**


Import
Import dashboard from file or Grafana.com

Options

Name: Node Exporter Dashboard EN 2020101-StarsL.cn

Folder: General

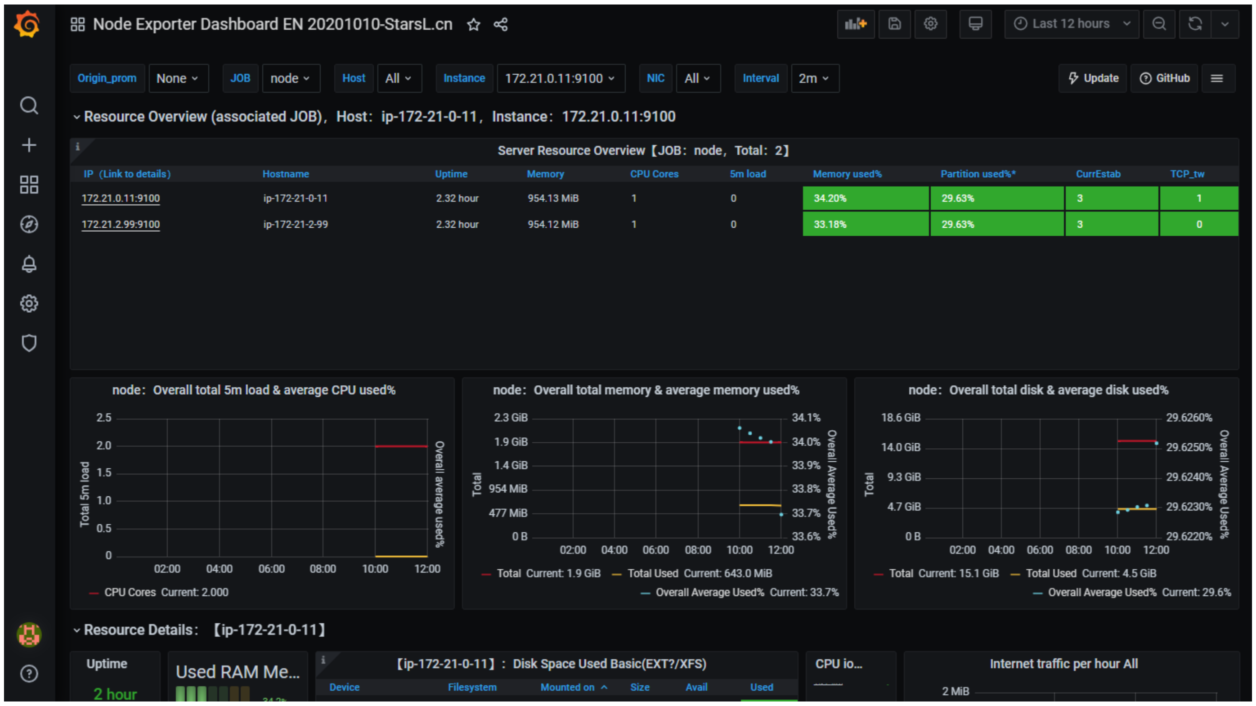
Unique identifier (uid)
The unique identifier (uid) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The uid allows having consistent URL's for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

xfpJB9FG

VictoriaMetrics: Prometheus1

Click on import

After import the dashboard it will automatically show cases the graphs


Node Exporter Dashboard EN 20201010-StarsL.cn
Last 12 hours
Update
GitHub
node: Overall total 5m load & average CPU used%
CPU Cores Current: 2.000
node: Overall total memory & average memory used%
Total Current: 1.9 GiB
Total Used Current: 643.0 MiB
Overall Average Used% Current: 33.7%
node: Overall total disk & average disk used%
Total Current: 15.1 GiB
Total Used Current: 4.5 GiB
Overall Average Used% Current: 29.6%
Uptime
Used RAM Me...
CPU io...
Internet traffic per hour All

Origin_prom None JOB node Host All Instance 172.21.0.11:9100 NIC All Interval 2m

Resource Overview (associated JOB), Host: ip-172-21-0-11, Instance: 172.21.0.11:9100

Server Resource Overview【JOB: node, Total: 2】

IP (Link to details)	Hostname	Uptime	Memory	CPU Cores	5m load	Memory used%	Partition used%*	CurrEstab	TCP_tw
172.21.0.11:9100	ip-172-21-0-11	2.32 hour	954.13 MiB	1	0	34.20%	29.63%	3	1
172.21.2.99:9100	ip-172-21-2-99	2.32 hour	954.12 MiB	1	0	33.18%	29.63%	3	0

Resource Details: 【ip-172-21-0-11】

【ip-172-21-0-11】: Disk Space Used Basic(EXT?/XFS)

Device	Filesystem	Mounted on	Size	Avail	Used

Section :B Docker Metrics Monitoring: -

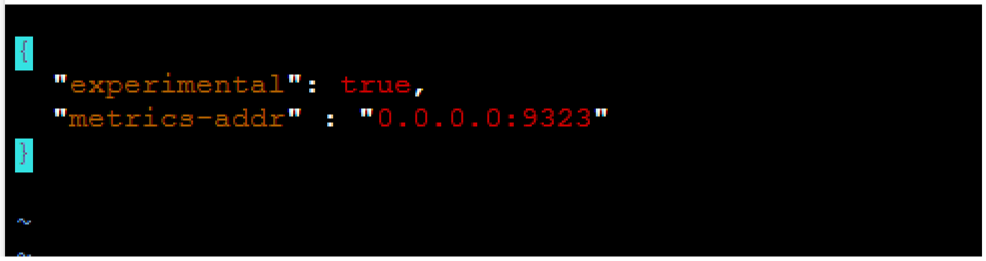
Integrating Docker with Prometheus and Grafana

Step1:-Install Node_exporter

To Install the Node_exporter follow the Above steps

Step2:- enable daemon metrics

- Specify the metrics-address in the daemon.json configuration file. This daemon expects the file to be located at one of the following locations by default. If the file doesn’t exist, create it. `/etc/docker/daemon.json`
- Add the following configuration

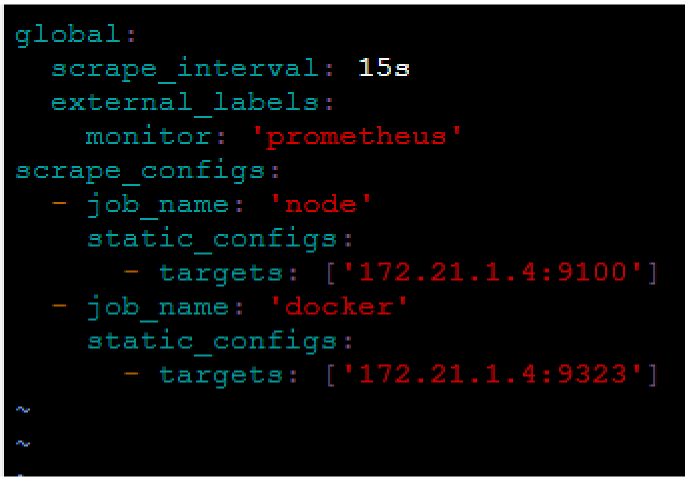


```
{
  "experimental": true,
  "metrics-addr" : "0.0.0.0:9323"
}
~
```

Docker now exposes Prometheus-compatible metrics on port 9323.

Step3: **configure Prometheus to monitor itself using yaml file. Create a `prometheus.yml` file at `/etc/prometheus/prometheus.yml` with the below content**

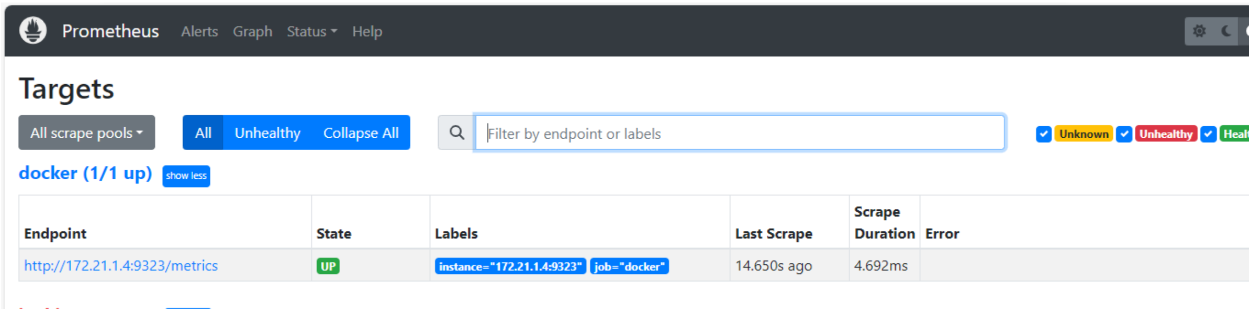
NOTE: -We can create a special job to handle any kind of metrics.



```
global:
  scrape_interval: 15s
  external_labels:
    monitor: 'prometheus'
scrape_configs:
  - job_name: 'node'
    static_configs:
      - targets: ['172.21.1.4:9100']
  - job_name: 'docker'
    static_configs:
      - targets: ['172.21.1.4:9323']
~
~
```

NOTE: After updating the Prometheus YAML configuration file, it's necessary to restart the Prometheus service for the changes to take effect.

```
sudo systemctl restart prometheus
```



Targets

docker (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.21.1.4:9323/metrics	UP	instance="172.21.1.4:9323" job="docker"	14.650s ago	4.692ms	

Step 4: Grafana Dashboard for Docker Metric

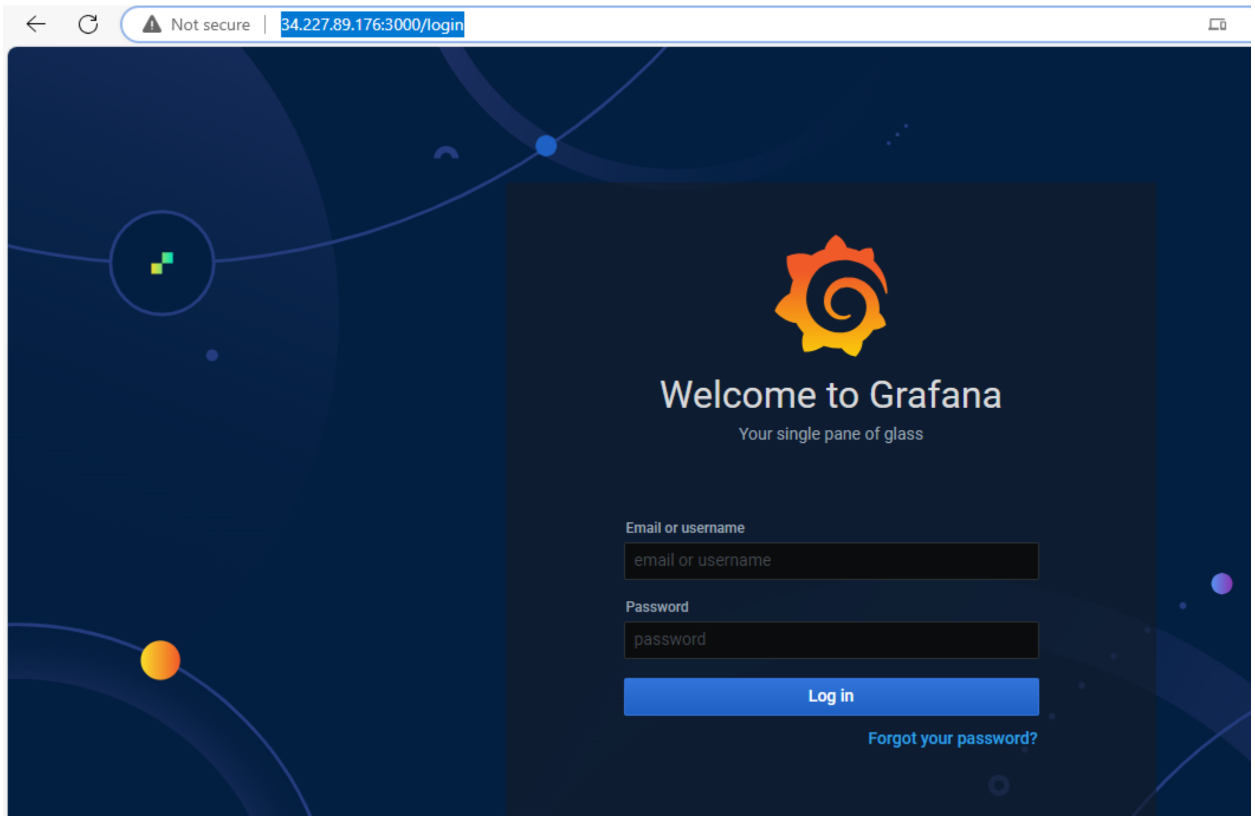
Add Prometheus as a Data Source

We need to add Prometheus as the data source in Grafana. Go to **Connections** > **Data Sources** and click the button "**Add new data source**". Then we need to enter the IP address of the Prometheus server and it port (default 9090) in the URL

- **Login to Grafana Dashboard**

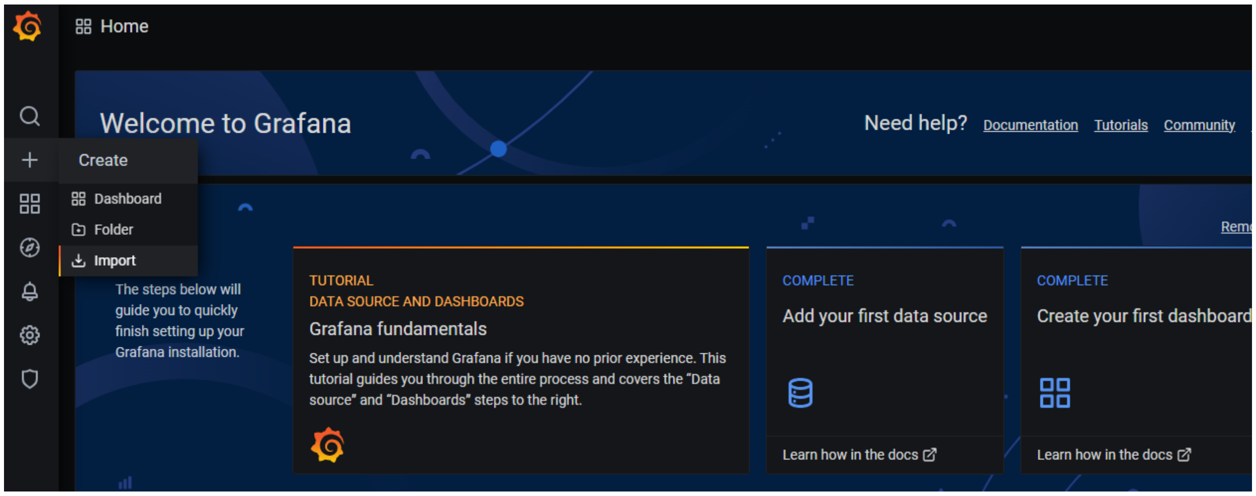
  Username:-admin

  Password:-admin



- **Import Dashboards**

  Docker Dashboard: Imported ID 1229

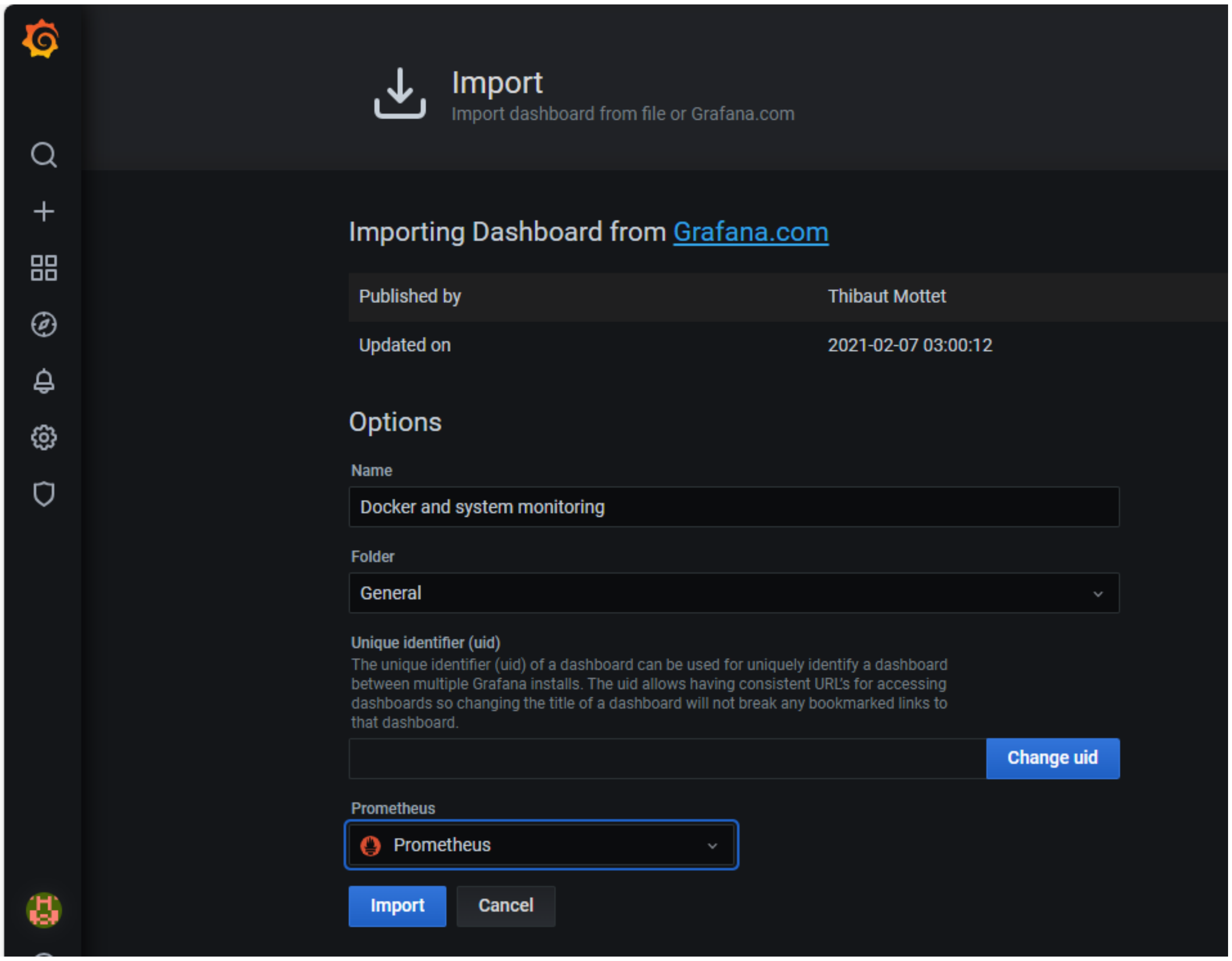


The steps below will guide you to quickly finish setting up your Grafana installation.

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

COMPLETE
Create your first dashboard
Learn how in the docs



Importing Dashboard from Grafana.com

Published by	Thibaut Mottet
Updated on	2021-02-07 03:00:12

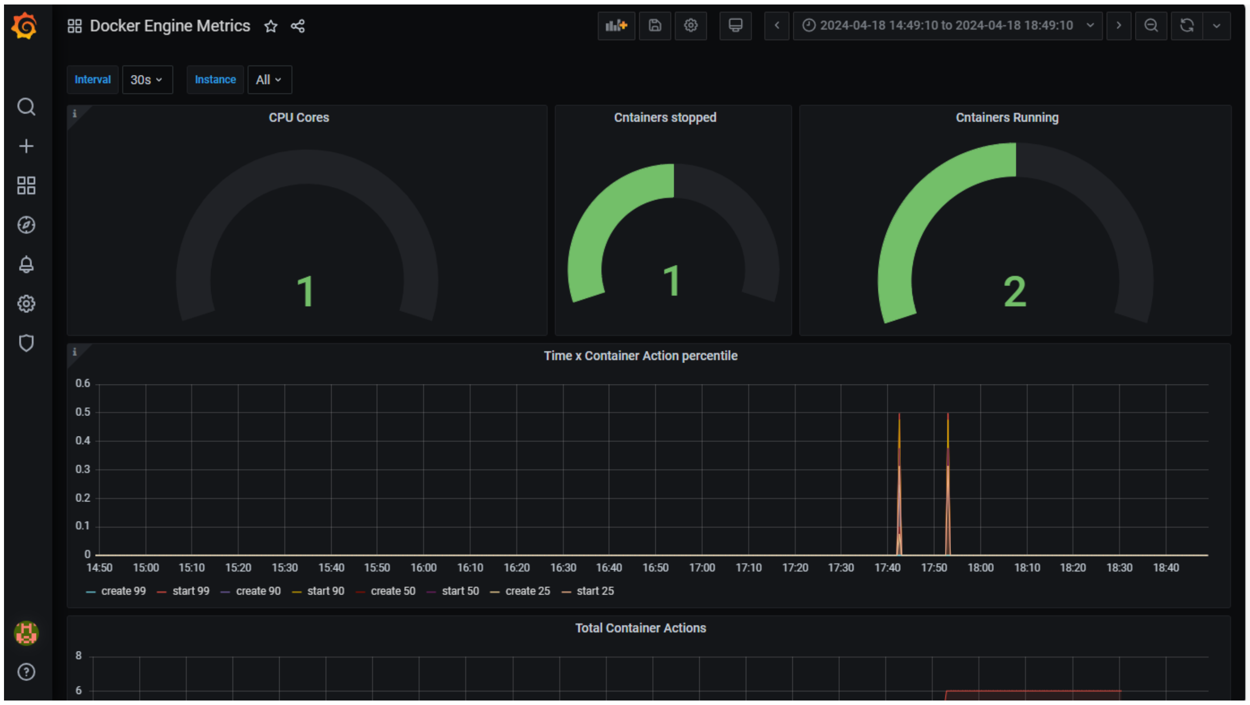
Options

Name: Docker and system monitoring

Folder: General

Unique identifier (uid)
The unique identifier (uid) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The uid allows having consistent URL's for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

Prometheus: Prometheus



Section :c Jenkins Metrics Monitoring:

Integrating Jenkins with Prometheus and Grafana

Step 1: Install Node Exporter

Follow the steps outlined previously to install Node Exporter. Ensure that Node Exporter is running on the Jenkins server to collect system metrics.

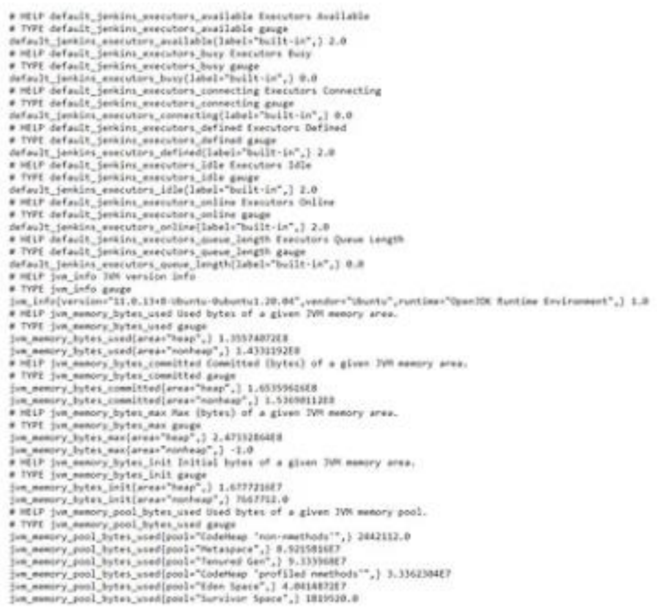
Step 2:-Install Prometheus plugin in Jenkins

- We need to install the plugin of **Prometheus** in Jenkins so that Prometheus can gather all the metrics of Jenkins
- In Jenkins click on manage plugin and search for **Prometheus metrics plugin** and click on install

Prometheus metrics plugin
Expose Jenkins metrics in prometheus format
2.0.10
Uninstall

Prometheus Metrics Plugin

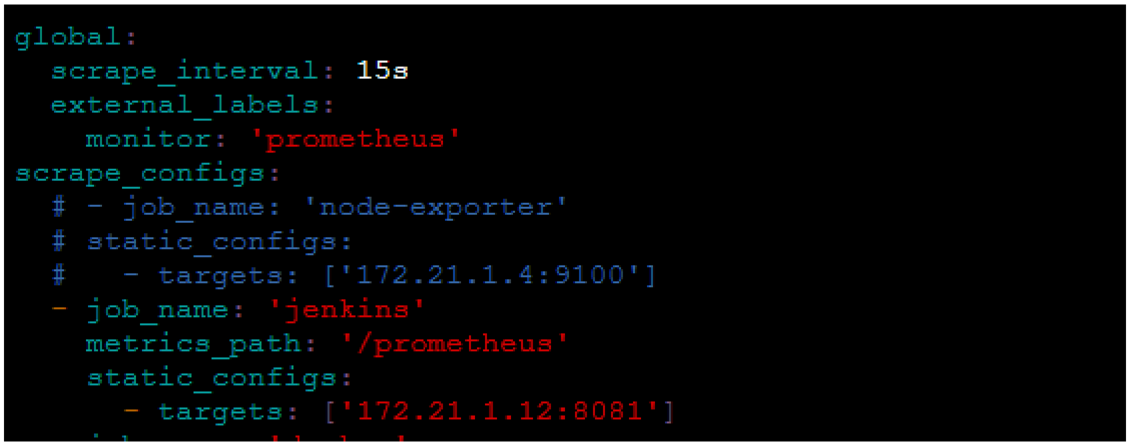
- The default path for Jenkins metrics is **<Public-IP:8080/prometheus>**



```
# HELP default_jenkins_executors_available Executors Available
# TYPE default_jenkins_executors_available gauge
default_jenkins_executors_available{label="built-in",} 2.0
# HELP default_jenkins_executors_busy Executors Busy
# TYPE default_jenkins_executors_busy gauge
default_jenkins_executors_busy{label="built-in",} 0.0
# HELP default_jenkins_executors_connecting Executors Connecting
# TYPE default_jenkins_executors_connecting gauge
default_jenkins_executors_connecting{label="built-in",} 0.0
# HELP default_jenkins_executors_defined Executors Defined
# TYPE default_jenkins_executors_defined gauge
default_jenkins_executors_defined{label="built-in",} 2.0
# HELP default_jenkins_executors_idle Executors Idle
# TYPE default_jenkins_executors_idle gauge
default_jenkins_executors_idle{label="built-in",} 2.0
# HELP default_jenkins_executors_online Executors Online
# TYPE default_jenkins_executors_online gauge
default_jenkins_executors_online{label="built-in",} 2.0
# HELP default_jenkins_executors_queue_length Executors Queue Length
# TYPE default_jenkins_executors_queue_length gauge
default_jenkins_executors_queue_length{label="built-in",} 0.0
# HELP jvm_info JVM version info
# TYPE jvm_info gauge
jvm_info{version="11.0.13+8-Ubuntu-0ubuntu1.20.04",vendor="Ubuntu",runtime="OpenJDK Runtime Environment",} 1.0
# HELP jvm_memory_bytes_used Used bytes of a given JVM memory area.
# TYPE jvm_memory_bytes_used gauge
jvm_memory_bytes_used{area="heap",} 1.35574072E8
jvm_memory_bytes_used{area="nonheap",} 1.4331192E8
# HELP jvm_memory_bytes_committed Committed (bytes) of a given JVM memory area.
# TYPE jvm_memory_bytes_committed gauge
jvm_memory_bytes_committed{area="heap",} 1.65359616E8
jvm_memory_bytes_committed{area="nonheap",} 1.53690112E8
# HELP jvm_memory_bytes_max Max (bytes) of a given JVM memory area.
# TYPE jvm_memory_bytes_max gauge
jvm_memory_bytes_max{area="heap",} 2.47132864E8
jvm_memory_bytes_max{area="nonheap",} -1.0
# HELP jvm_memory_bytes_init Initial bytes of a given JVM memory area.
# TYPE jvm_memory_bytes_init gauge
jvm_memory_bytes_init{area="heap",} 1.6777216E7
jvm_memory_bytes_init{area="nonheap",} 7667712.0
# HELP jvm_memory_pool_bytes_used Used bytes of a given JVM memory pool.
# TYPE jvm_memory_pool_bytes_used gauge
jvm_memory_pool_bytes_used{pool="CodeHeap 'non-nmethods'",} 2842112.0
jvm_memory_pool_bytes_used{pool="Metaspace",} 8.9215816E7
jvm_memory_pool_bytes_used{pool="Tenured Gen",} 9.333968E7
jvm_memory_pool_bytes_used{pool="CodeHeap 'profiled nmethods'",} 3.3362304E7
jvm_memory_pool_bytes_used{pool="Eden Space",} 4.0414872E7
jvm_memory_pool_bytes_used{pool="Survivor Space",} 1819520.0
```

Jenkins Metrics

Step 2:- In order for Prometheus to gather the metrics we need to define below code in **prometheus.yml** under the **scrape_configs**



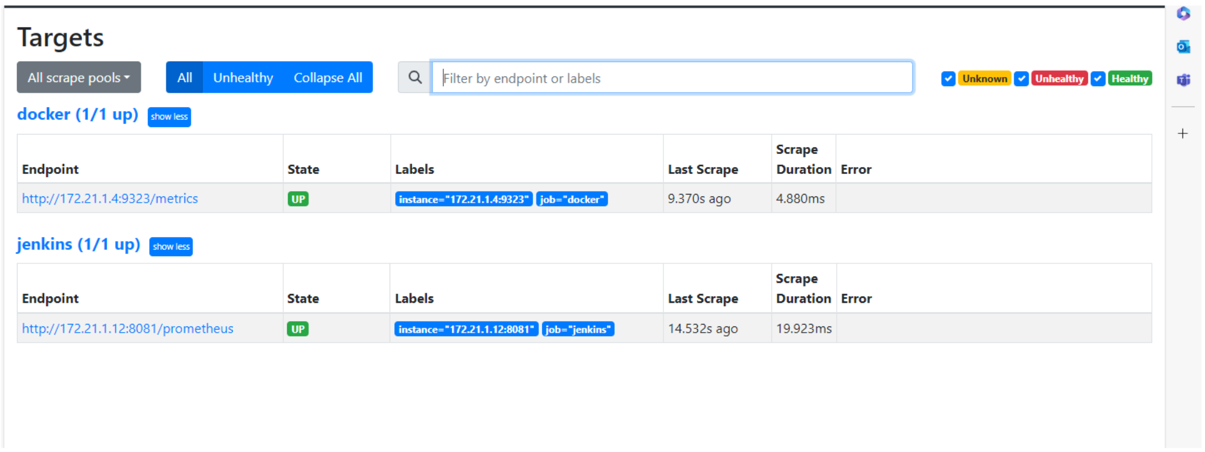
```
global:
  scrape_interval: 15s
  external_labels:
    monitor: 'prometheus'
scrape_configs:
  # - job_name: 'node-exporter'
  # static_configs:
  #    - targets: ['172.21.1.4:9100']
  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['172.21.1.12:8081']
```

- ONCE YML FILE CHANGES MADE RUN RESTART COMMAND

  Sudo systemctl restart prometheous

Step 2:- Verify Jenkins Port Availability

To ensure that the Jenkins port is up and running, you can check the Prometheus dashboard. Look for metrics related to Jenkins or the specific port Jenkins is running on to confirm its availability and status.


Targets

All scrape pools | All | Unhealthy | Collapse All | Filter by endpoint or labels | ✓ Unknown ✓ Unhealthy ✓ Healthy

docker (1/1 up) show less

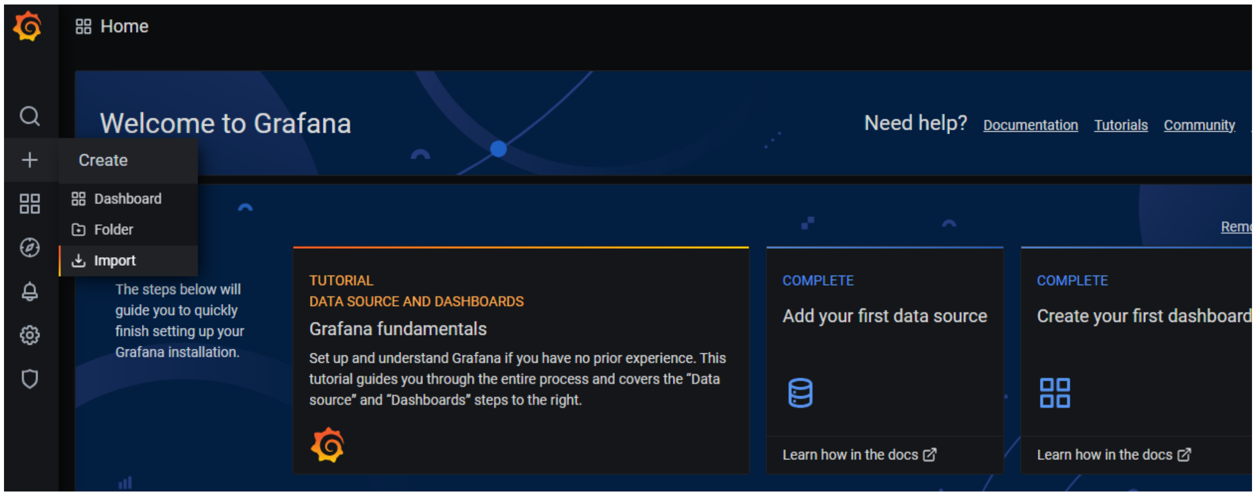
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.21.1.4:9323/metrics	UP	instance="172.21.1.4:9323" job="docker"	9.370s ago	4.880ms	

jenkins (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.21.1.12:8081/prometheus	UP	instance="172.21.1.12:8081" job="jenkins"	14.532s ago	19.923ms	

Step 3:-Create a Jenkins Dashboard

- A) **Import Dashboard**

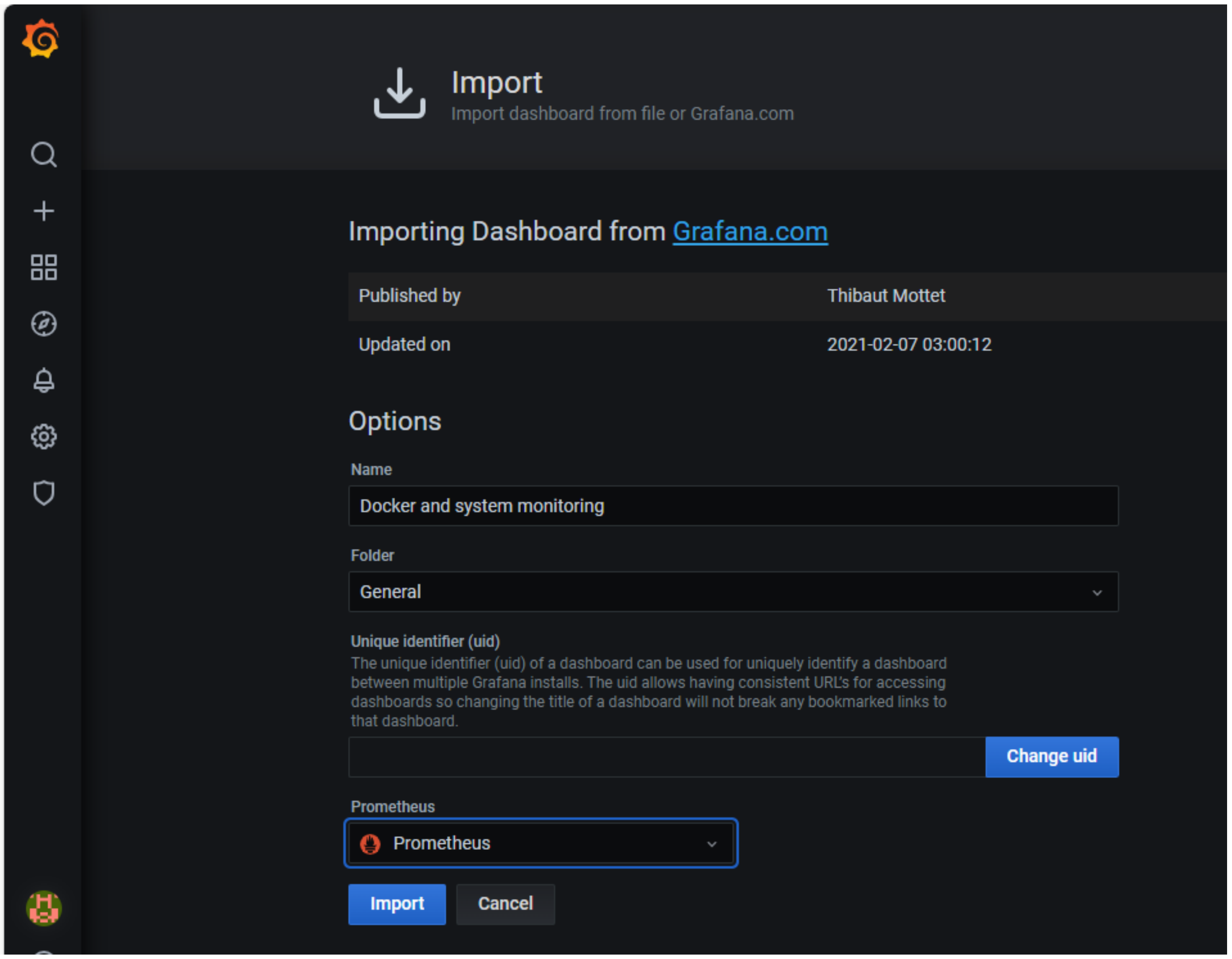


The steps below will guide you to quickly finish setting up your Grafana installation.

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

COMPLETE
Create your first dashboard
Learn how in the docs


Import
Import dashboard from file or Grafana.com

Importing Dashboard from Grafana.com

Published by	Thibaut Mottet
Updated on	2021-02-07 03:00:12

Options

Name
Docker and system monitoring

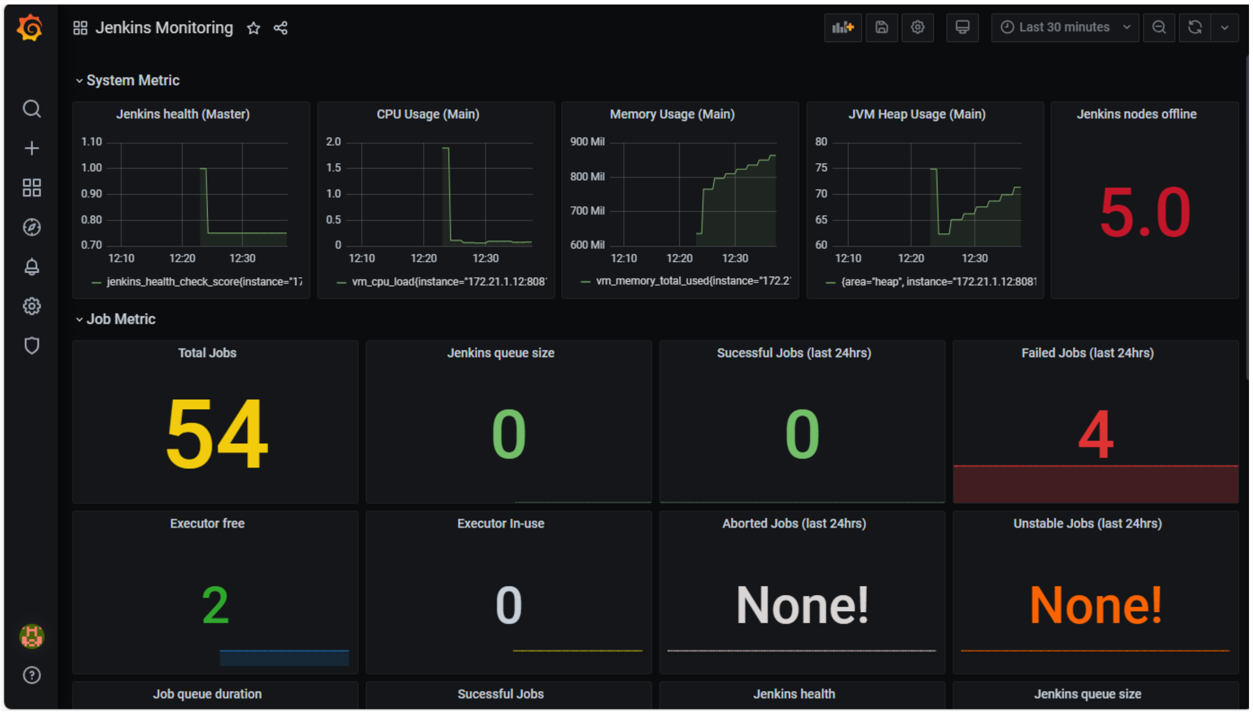
Folder
General

Unique identifier (uid)
The unique identifier (uid) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The uid allows having consistent URL's for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

Prometheus
Prometheus

Select Data Source

- Ensure that you select Prometheus as the data source while importing the Jenkins dashboard, consistent with the previously configured Prometheus setup.
- Dashboard id 14550


Jenkins Monitoring
Last 30 minutes
System Metric
Jenkins health (Master)
1.10
1.00
0.90
0.80
0.70
12:10
12:20
12:30
jenkins_health_check_score{instance="17
CPU Usage (Main)
2.0
1.5
1.0
0.5
0
12:10
12:20
12:30
vm_cpu_load{instance="172.21.1.12:808
Memory Usage (Main)
900 Mil
800 Mil
700 Mil
600 Mil
12:10
12:20
12:30
vm_memory_total_used{instance="172.2
JVM Heap Usage (Main)
80
75
70
65
60
12:10
12:20
12:30
{area="heap", instance="172.21.1.12:8081
Jenkins nodes offline
5.0
Job Metric
Total Jobs
54
Jenkins queue size
0
Sucessful Jobs (last 24hrs)
0
Failed Jobs (last 24hrs)
4
Executor free
2
Executor In-use
0
Aborted Jobs (last 24hrs)
None!
Unstable Jobs (last 24hrs)
None!
Job queue duration
Sucessful Jobs
Jenkins health
Jenkins queue size