

Installation Manual: Monitoring system using Prometheus and Grafana

Station:

EDA Office

Drogon Server

Hardware:

Supermicro: Ubuntu server 22.04 (Jammy Jellyfish)

Table of Contents

Overview	3
Hardware/Software Requirements.....	3
Initial setup and preparation.....	3
Configuration	5
Install Node Exporter	6
Node Exporter service.....	7
Dashboards	9
Conclusion.....	11

Overview

This documentation is used for the successful installation of a monitoring system on Linux based instruments running Ubuntu Server 20.04 or later. The system used in this manual is the Drogon server in the EDA office running Ubuntu Server 22.04

The monitoring system that will be used is Prometheus, which is an open-source monitoring system which records real-time metrics in a time series database.

Hardware/Software Requirements

The hardware required for the setup of the Prometheus system monitoring software is as follows:

- Computer with Ubuntu OS 20.04 or later
- Screen and keyboard/remote login
- Prior setup to local network with a static IP

Initial setup and preparation

The Ubuntu server will need to be updated with all the relevant packages, this can be done with the following command in your terminal.

```
sudo apt update
```

Make sure that the server is connected to the internet or configured to run from a offline repository.

Setup Procedure

In order to make the necessary system changes, a new system account for Prometheus is required. This will make troubleshooting much easier in the future, it provides a bit more security to the services running and simplifies administration.

Create a system account for Prometheus with the following command:

```
Sudo useradd \  
--system \  
--no-create-home \  
--shell /bin/false prometheus
```

This will create a system account without a home directory and prevents logging in as a user.

Check for the latest version of Prometheus here:

<https://prometheus.io/download/#prometheus>

You'll need to download the latest version for the linux_amd64... server, using the command:

```
wget  
https://github.com/prometheus/prometheus/releases/download/v2.37.0/prometheus-  
2.37.0.linux-amd64.tar.gz
```

Extract all the file from the folder above:

```
tar -xvf prometheus-2.32.1.linux-amd64.tar.gz
```

Create a directory for the data to be stored and the Prometheus configuration files:

(An external data directory may be implemented, i.e. sdb1 etc)

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

Change to the new directory and move the following files:

```
cd prometheus-2.37.0.linux-amd64  
sudo mv prometheus promtool /usr/local/bin/  
sudo mv consoles/ console_libraries/ /etc/prometheus/  
sudo mv prometheus.yml /etc/prometheus/prometheus.yml
```

Change the permission for Prometheus system account:

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

Verify that Prometheus is installed and executable:

```
Prometheus --version
```

The output should look similar to the bellow:

```
EDA/Drogon
engineer@~$ prometheus --version
prometheus, version 2.37.0 (branch: HEAD, revision: b41e0750abf5cc18d8233161560731de05199330)
  build user:      root@0ebb6827e27f
  build date:      20220714-15:13:18
  go version:      go1.18.4
  platform:        linux/amd64
engineer@~$
```

Configuration

The server setup will be using systemd for running our Prometheus.service configuration file.

Create a new configuration file with the following command:

```
sudo nano /etc/systemd/system/prometheus.service
```

Enter the following code:

```
EDA/Drogon
GNU nano 6.2 /etc/systemd/system/prometheus.service
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

StartLimitIntervalSec=500
StartLimitBurst=5

[Service]
User=prometheus
Group=prometheus
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/prometheus \
  --config.file=/etc/prometheus/prometheus.yml \
  --storage.tsdb.path=/data \
  --web.console.templates=/etc/prometheus/consoles \
  --web.console.libraries=/etc/prometheus/console_libraries \
  --web.listen-address=10.160.11.56:9090 \
  --web.enable-lifecycle

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

Note the web.listen-address=10.160.11.56:9090, it's default is localhost:9090

When running it within Docker/Nginx or for further Docker containers, explicitly specify the host IP address. Running over all networks would be 0.0.0.0:9090

Exit and save the new file. Now that the configuration file is setup, you'll need to enable the service on systemd with the following command:

```
sudo systemctl enable prometheus
sudo systemctl start prometheus           //Start the service
sudo systemctl status prometheus         //Verify it's running
```

Install Node Exporter

The Node Exporter will be used to collect Linux system metrics. The configuration is as follows.

Create a separate system account for Node Explorer:

```
sudo useradd \
--system \
--no-create-home \
--shell /bin/false node_exporter
```

Check for the latest release here: https://prometheus.io/download/#node_exporter

```
wget
https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_exporter-1.3.1.linux-amd64.tar.gz
```

Extract all the folder contents:

```
tar -xvf node_exporter-1.3.1.linux-amd64.tar.gz
```

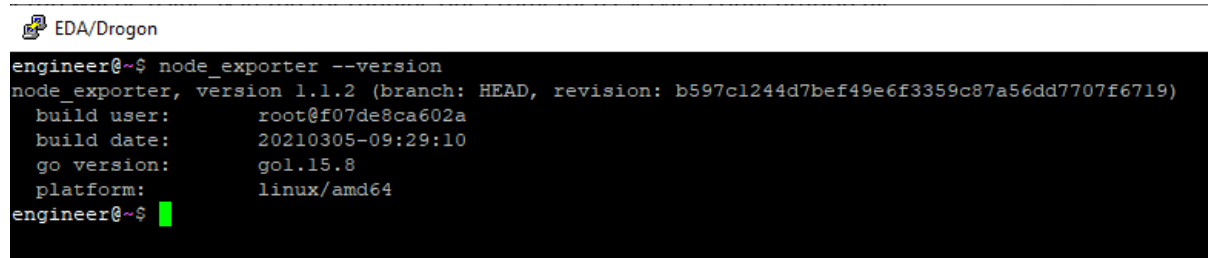
Move the files to the correct directories:

```
sudo mv \
node_exporter-1.3.1.linux-amd64/node_exporter \
/usr/local/bin/
```

Verify that Node Exporter can run:

```
node_exporter --version
```

You'll get a similar output as bellow:



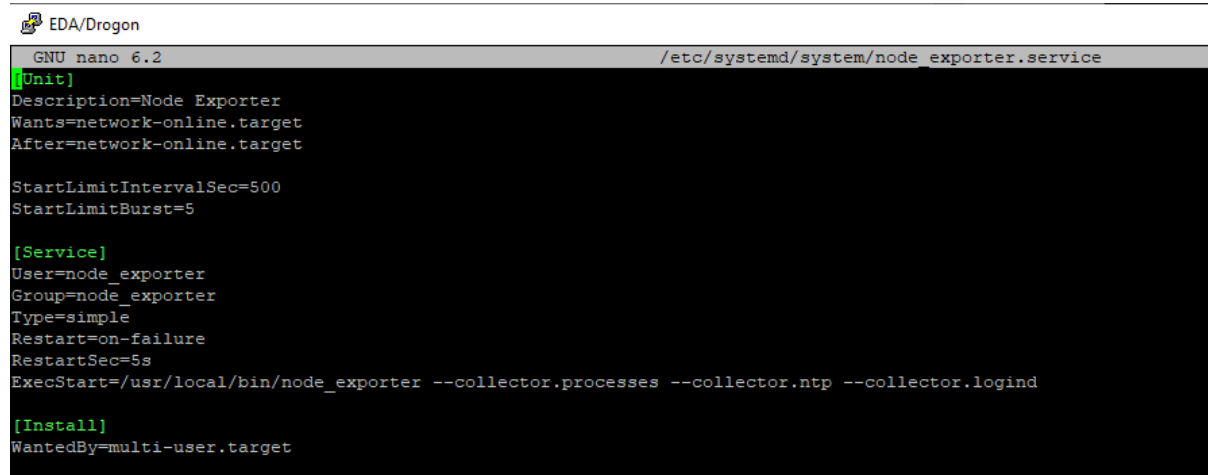
```
EDA/Drogon
engineer@~$ node_exporter --version
node_exporter, version 1.1.2 (branch: HEAD, revision: b597c1244d7bef49e6f3359c87a56dd7707f6719)
  build user:      root@f07de8ca602a
  build date:      20210305-09:29:10
  go version:      go1.15.8
  platform:        linux/amd64
engineer@~$
```

Node Exporter service

Next, a new service is required for the node_exporter.service to run on systemd.

```
sudo nano /etc/systemd/system/node_exporter.service
```

Edit the file as follow:



```
GNU nano 6.2 /etc/systemd/system/node_exporter.service
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

StartLimitIntervalSec=500
StartLimitBurst=5

[Service]
User=node_exporter
Group=node_exporter
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/node_exporter --collector.processes --collector.ntp --collector.logind

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

Exit and save the new configuration.

To automatically start the Node Exporter service after a reboot, run the following command:

```
sudo systemctl enable node_exporter
```

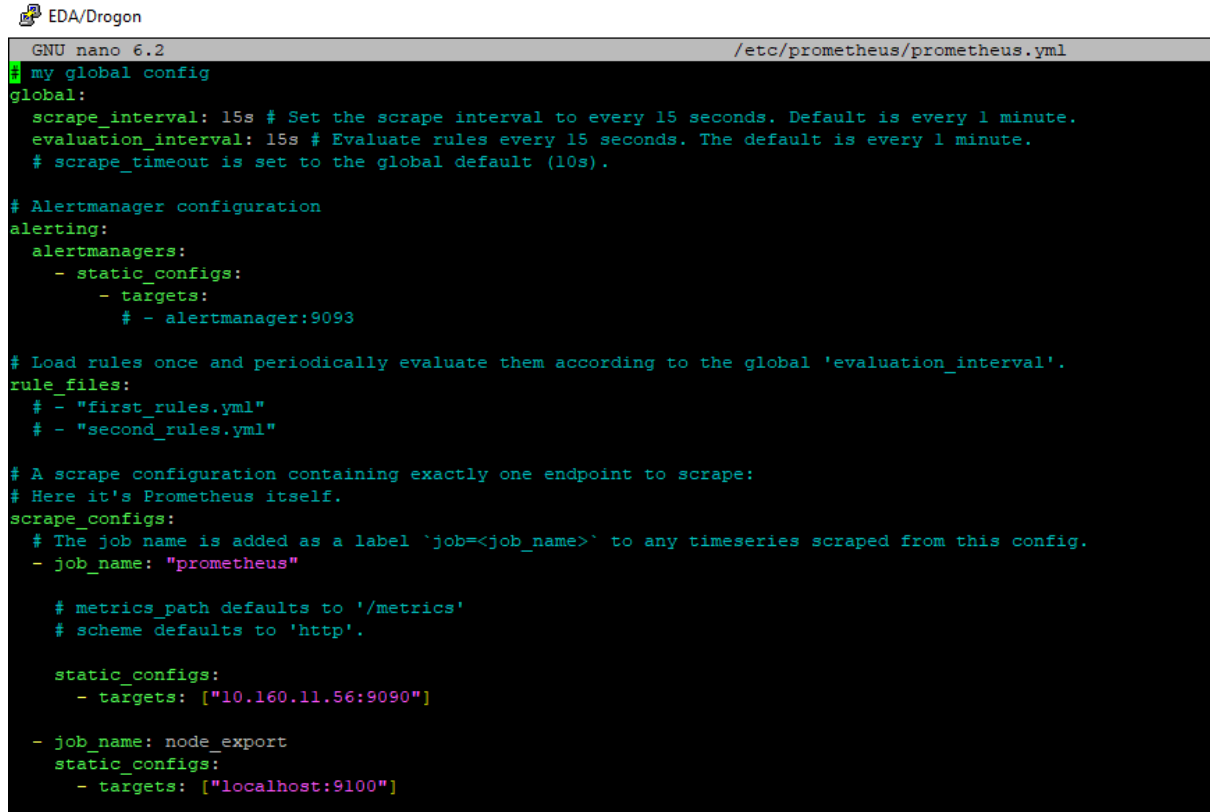
Start the new service:

```
sudo systemctl start node_exporter
```

```
sudo systemctl status node_exporter //Verify it's running
```

Create a static target: Complete as in screen snippet

```
sudo nano /etc/prometheus/prometheus.yml
```



```
GNU nano 6.2 /etc/prometheus/prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

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

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

    static_configs:
      - targets: ["10.160.11.56:9090"]

  - job_name: node_exporter
    static_configs:
      - targets: ["localhost:9100"]
```

Check and Reload the Prometheus configuration without restarting the service, by running the following commands:

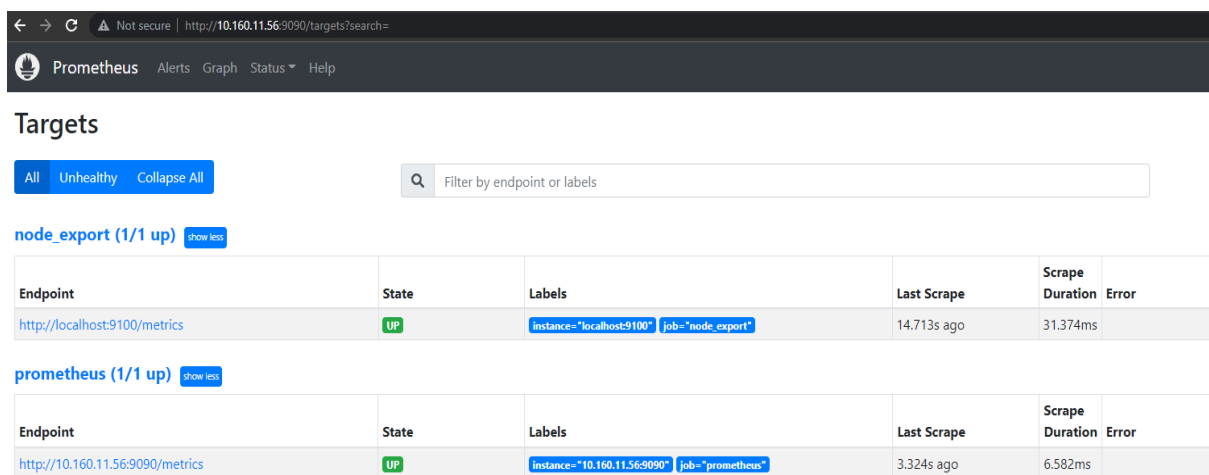
```
promtool check config /etc/prometheus/prometheus.yml
```

```
curl -X POST http://localhost:9090/-/reload //Replace localhost with the host IP
if it does not want to reload...
```


Dashboards

To test the both targets created above, “Prometheus” and “node_export”, open a web browser and open the IP address of the host PC with port 9090. On this interface, you’ll be able to find the Prometheus dashboards and customise them. Seeing that we want all our instruments on one dashboard interface, we’ll post the data from Prometheus to a Grafana dashboard. The “Prometheus” target will be used for a custom dashboard on Grafana, whereas the “node_export” target is used with the Node Exporter Full precompiled dashboard.

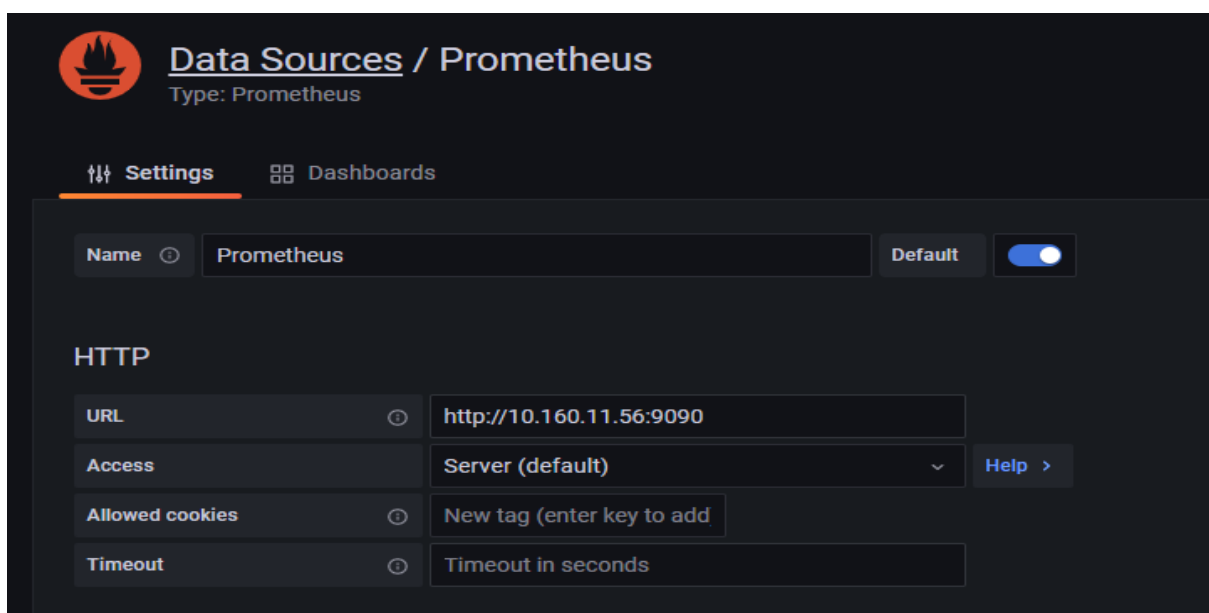
The Prometheus interface under ‘Targets’ should be similar to the image below with both targets being online.



The screenshot shows the Prometheus web interface at `http://10.160.11.56:9090/targets?search=`. The 'Targets' section is active, showing two targets: 'node_export' and 'prometheus'. Both are in an 'UP' state. The 'node_export' target has an endpoint of `http://localhost:9100/metrics` and labels `instance="localhost9100"` and `job="node_export"`. The 'prometheus' target has an endpoint of `http://10.160.11.56:9090/metrics` and labels `instance="10.160.11.56:9090"` and `job="prometheus"`.

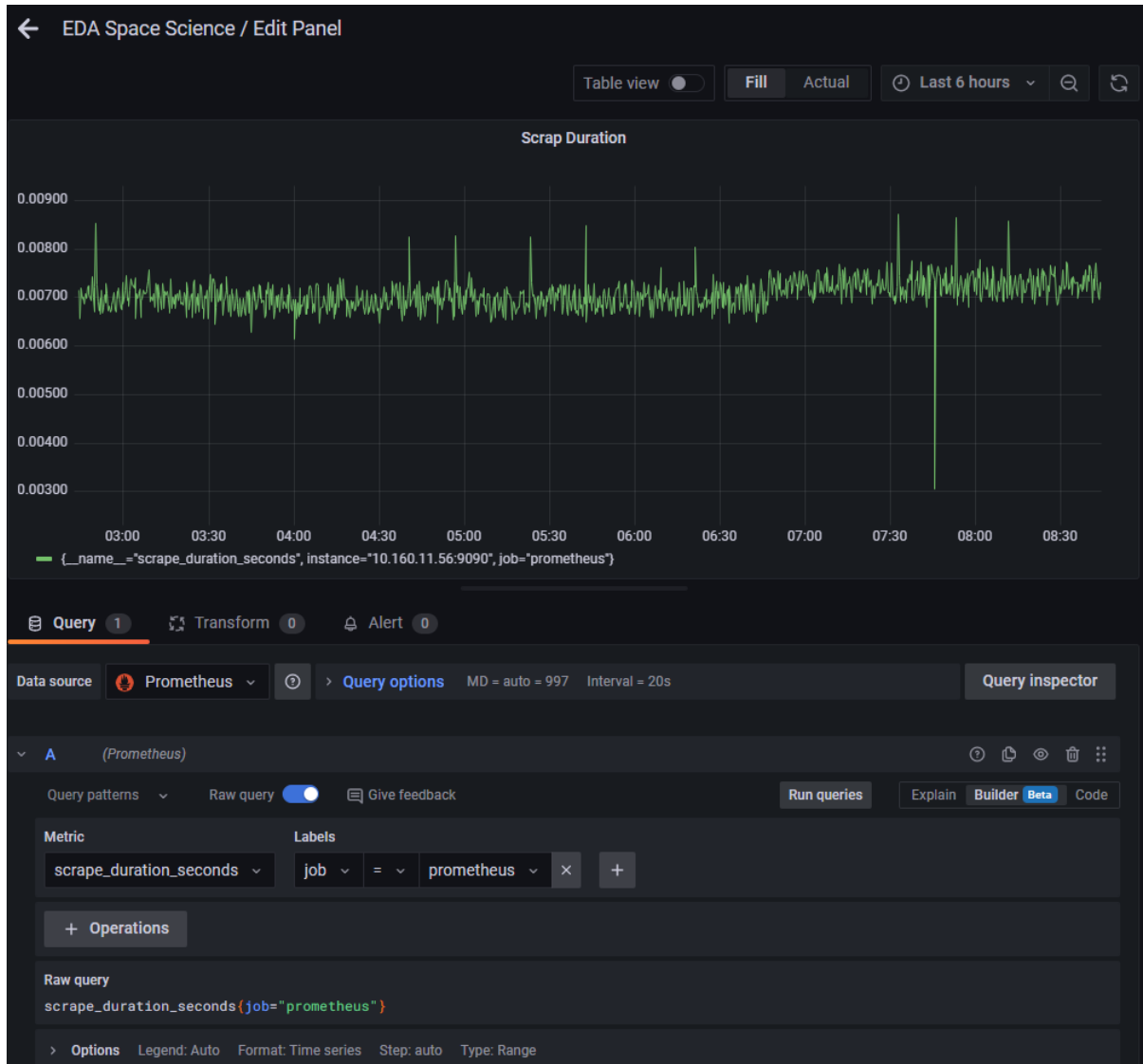
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<code>http://localhost:9100/metrics</code>	UP	<code>instance="localhost9100"</code> <code>job="node_export"</code>	14.713s ago	31.374ms	
<code>http://10.160.11.56:9090/metrics</code>	UP	<code>instance="10.160.11.56:9090"</code> <code>job="prometheus"</code>	3.324s ago	6.582ms	

On Grafana, go to the Data Sources page and select the Prometheus data source. Then type in your Prometheus host pc’s IP address with the port 9090. Click ‘Save & test’



The screenshot shows the Grafana 'Data Sources / Prometheus' configuration page. The 'Name' is 'Prometheus' and it is set as the 'Default' data source. Under the 'HTTP' section, the 'URL' is `http://10.160.11.56:9090`, 'Access' is 'Server (default)', 'Allowed cookies' is 'New tag (enter key to add)', and 'Timeout' is 'Timeout in seconds'.

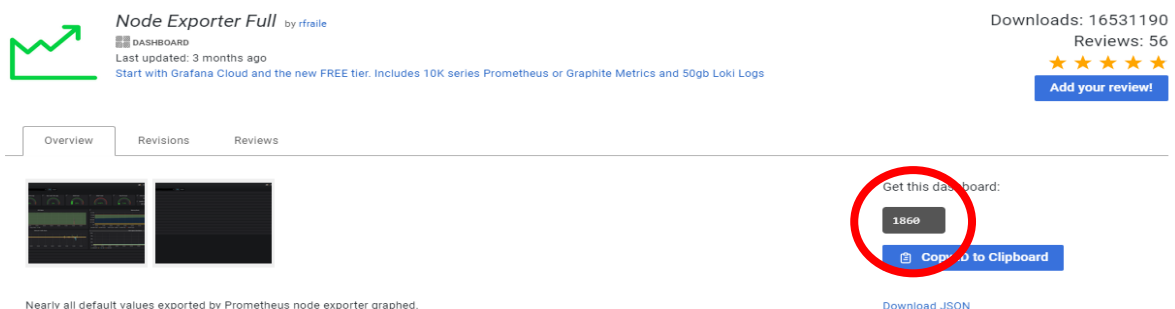
Creating a new dashboard with the Prometheus data source by adding a new panel and choose Prometheus as the data source with the Metric of your choose to display and customise the dash board. Below is an example of the Scrape Duration on the Drogon PC.



The node exporter has it's own dashboard that is available for to import from the Grafana main page.

Go to the Grafana Dash board web page, <https://grafana.com/grafana/dashboards/>

Search for 'Node Exporter Full' , open it to find the import ID: "1860"

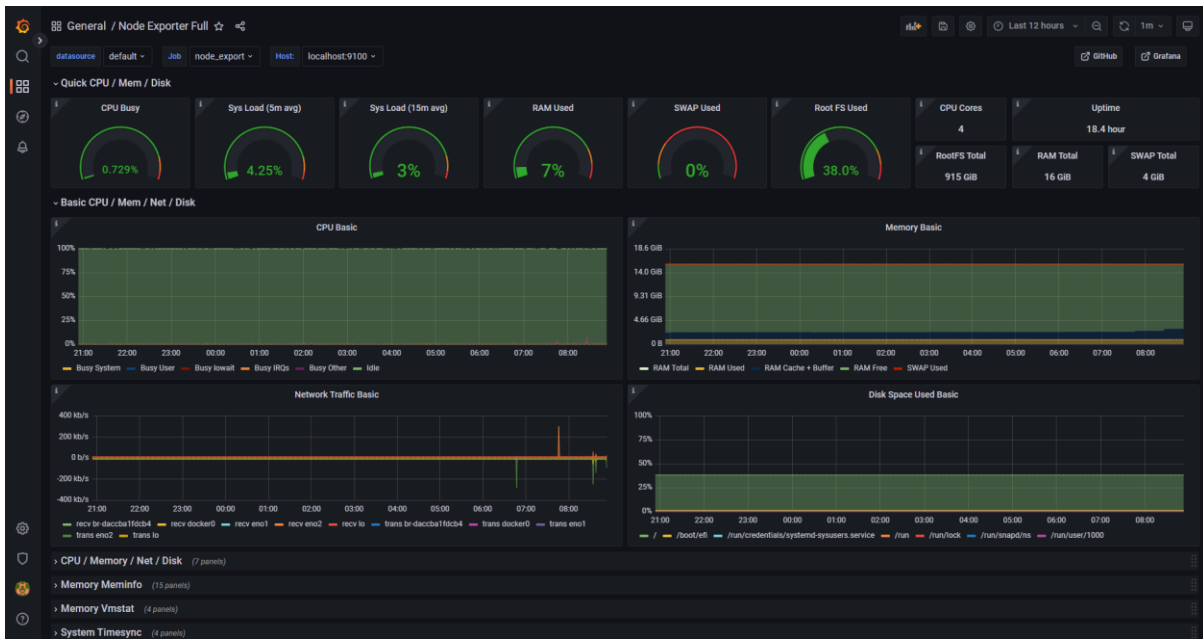


The screenshot shows the Grafana dashboard listing for 'Node Exporter Full' by rfralle. The dashboard has 16531190 downloads and 56 reviews. The 'Overview' tab is selected, showing a preview of the dashboard. The 'Get this dashboard:' section shows the import ID '1860' circled in red, with a 'Copy ID to Clipboard' button. Below the preview, it says 'Nearly all default values exported by Prometheus node exporter graphed.'

Go to the Grafana dashboard and open the import dashboard page. Paste the import ID, 1860, then just click 'Load'

The Node Exporter Dashboard should be displayed now under your dashboards as Node Exporter Full.

It should look similar to the below example.



Conclusion

All instruments will need to be on the same Grafana server with their own dashboards and/or a combined dashboard. All documentation for this setup and others, please refer to the EDA Github repository. <https://github.com/ss-eda>

Please keep in mind coding ethics and practises, by documenting and commenting on all scripts and work done. Upload the documentation with any scripts onto the Github repository.