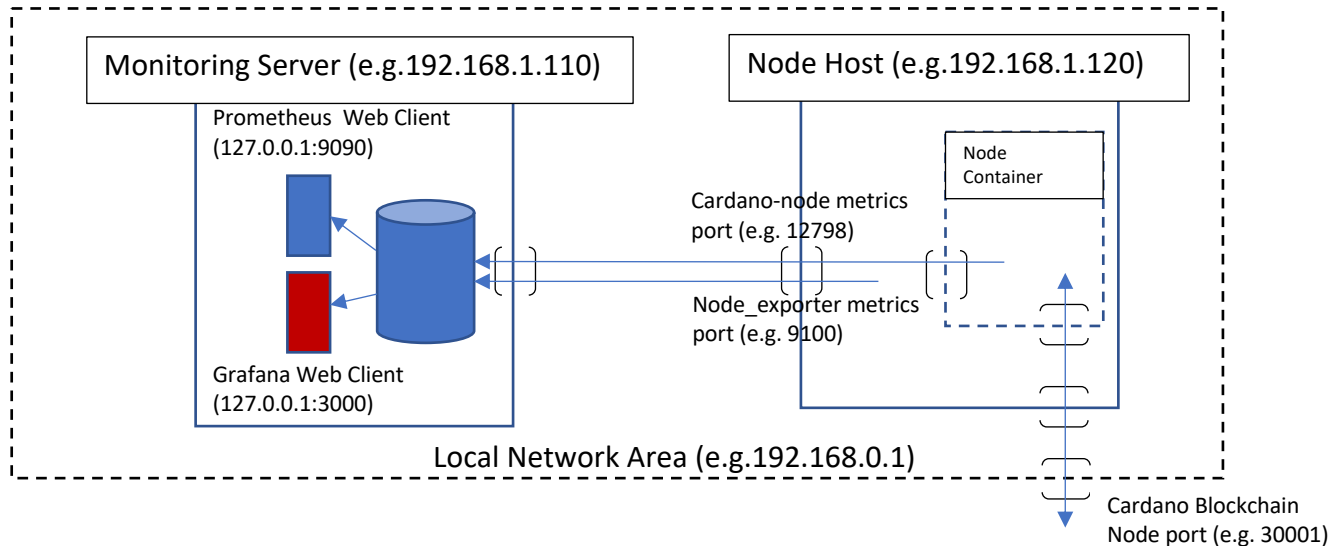


# Configuring Prometheus & Grafana Monitoring System for a Cardano Node Running in a Docker Container

## 1. Goal

Setting up a Prometheus & Grafana monitoring system for a Cardano node, within the same local area network, according to the diagram below:



## 2. Prerequisites

The following instructions assume that you have successfully installed and activated a Cardano node based on the step-by-step instructions by Easy1 stake pool (visit his GitHub repository for more information: <https://github.com/speedwing/cardano-staking-pool-edu>).

## 3. Install & Configure Prometheus

- 1) If the node container is currently running, we'll need to stop it and remove it in order to apply some changes in the **run-node.sh** script and the **mainnet/testnet-config.json** configuration file.

```
docker ps -a # retrieve the container id
docker stop "container id"
docker rm "container id"
```

*Note: As we are now changing settings in the original files provided by Easy1, it is suggested to create a copy of the entire **cardano-staking-pool-edu** folder and name it for instance **cardano-staking-pool-edu-copy**. In this case we still keep the original files provided by Easy1 unchanged on the RaspPi.*

- 2) So let's copy the folder and its files.

```
cd /home/ubuntu
sudo cp -avr cardano-staking-pool-edu cardano-staking-pool-edu-copy
```

- 3) Now that we have made a copy of the folder, we can make the desired changes in it.

```
cd cardano-staking-pool-edu-copy/cardano-node
sudo nano run-node.sh
```

- 4) We now opened the **run-node.sh** script in the nano editor. We have to add and change following parts:

*Note: This example is made for the BP (block producing node). Same concept can be applied to the relay node.*

- Add Prometheus port forwarding from container to host (see yellow highlighted parts below)
- Change path to the **mainnet/testnet-config.json** file (see green highlighted parts below)

```
docker run --name "cardano-node-${NETWORK}" -d -v $DB_FOLDER:/db -v
/home/ubuntu/cardano-staking-pool-edu-copy/cardano-
node/config/testnet /etc/config/cardano-node/config -v
/home/ubuntu/.keys/testnet:/etc/config/cardano-node/keys -p
30001:30001 -p 12798:12798 -e CARDANO_NODE_SOCKET_PATH=/db/node.socket
"${@:3}" "cardano-node:${IMAGE_TAG}" \
"cardano-node run \
--topology /etc/config/cardano-node/config/${NETWORK}-topology.json \
--database-path /db \
--socket-path /db/node.socket \
--host-addr 0.0.0.0 \
--port $CARDANO_NODE_PORT \
--config /etc/config/cardano-node/config/${NETWORK}-config.json \
--shelley-kes-key ${KES_SKEY_PATH} \
--shelley-vrf-key ${VRF_SKEY_PATH} \
--shelley-operational-certificate ${NODE_OP_CERT_PATH}"
```

- 5) Now that we have applied these changes, save the file and close nano with the following keyboard combination:

```
^o
(Press Enter)
^x
```

- 6) We now need to make a change in the **mainnet/testnet-config.json** file that is then deployed from our new **run-node.sh** command we just modified. So let's open it in nano:

```
sudo nano /home/ubuntu/cardano-staking-pool-edu-copy/cardano-
node/config/testnet/testnet-config.json
```

- 7) Scroll down to the line **"hasPrometheus":** [ and change the IP address from **"127.0.0.1"** to **"0.0.0.0"**

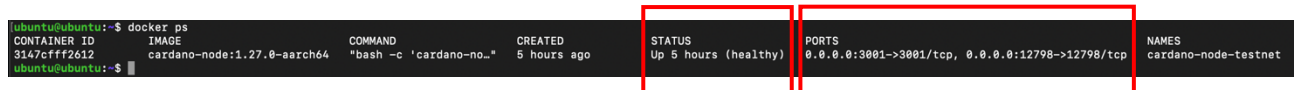
- 8) So now that all the needed changes have been made, let's start the node.

```
cd ~/cardano-staking-pool-edu-copy/cardano-node
NETWORK=testnet ./run-node.sh /home/ubuntu/cardano-node/testnet 30001 --restart
unless-stopped
```

- 9) Check if the node successfully started, has the port forwarded and it gives a healthy state.

```
docker ps
```

- 10) We should see something like this:



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
3147cfff2612	cardano-node:1.27.0-aarch64	"bash -c 'cardano-no..."	5 hours ago	Up 5 hours (healthy)	0.0.0.0:3001->3001/tcp, 0.0.0.0:12798->12798/tcp	cardano-node-testnet

Now we configured the node for transmitting the Prometheus metrics to the host over port 12798. We now need to set up a Prometheus server that scrapes the Prometheus metrics from the node via the 12798 port. We can use another RaspPi or just our own PC/Mac. Whatever device you use, let's run Ubuntu 20.04 on it too (just for simplicity).

- 11) Log in to your device where you intend to set-up the Prometheus server and open the terminal.

- 12) Let's install Prometheus.

```
sudo apt install prometheus
```

- 13) We need to configure the **prometheus.yml** file so that the server is scraping the metrics from the node. Let's open it with nano.

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

- 14) Scroll down where the **scrape\_configs:** section start.

```
Change job_name: 'node' → 'cardano'
Insert the IP and port of the cardano node under
static_configs:
  - targets: ['192.168.1.120:12798']
```

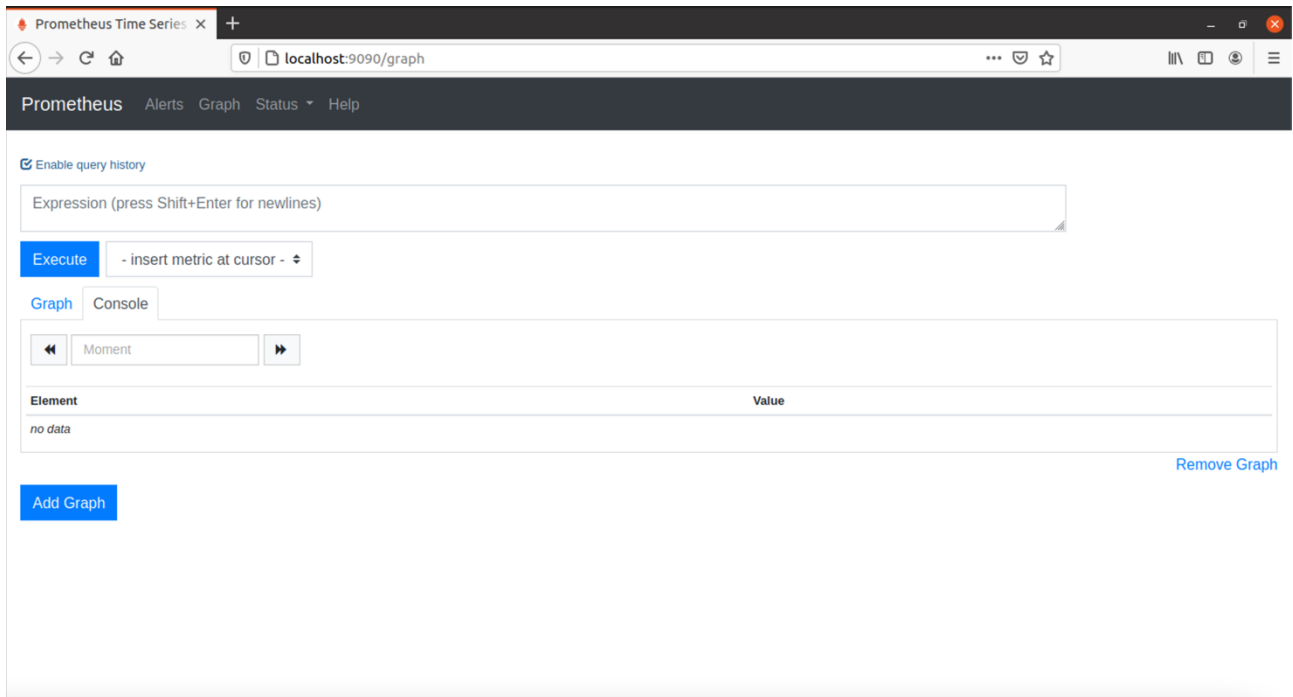
- 15) Now that we have applied these changes, save the file and close nano with the following keyboard combination:

```
^o
(Press Enter)
^x
```

16) Let's restart the Prometheus service with the new configuration

```
sudo systemctl restart prometheus.service
```

17) Open the web browser and enter the address <http://localhost:9090/graph>. You should see the following screen:



18) Go to **Status > Targets** and you should see this screen:

**cardano (1/1 up)** [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<a href="http://192.168.1.112:12798/metrics">http://192.168.1.112:12798/metrics</a>	UP	<a href="#">instance="192.168.1.112:12798"</a> <a href="#">job="cardano"</a>	2.829s ago	8.845ms	

**node (1/1 up)** [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<a href="http://localhost:9100/metrics">http://localhost:9100/metrics</a>	UP	<a href="#">instance="localhost:9100"</a> <a href="#">job="node"</a>	6.705s ago	76.86ms	

Check if the cardano endpoint state is UP (like in the screen above). If so, CONGRATULATIONS, you successfully installed your Prometheus monitoring system.

## 4. Install & Configure Grafana

Prometheus is certainly a great monitoring tool but as soon as we'd like to represent several metrics on the same dashboard for a better overview of the nodes performance, then Prometheus could become inadequate. Grafana provides a much better graphical representation of the metrics provided by Prometheus. So let us install and configure a nice Grafana dashboard.

- 1) First we need to download and extract the Prometheus **node\_exporter** on our Node host. This allows us to scrape additional important metrics about the node that are not provided within the cardano-node metrics. Log into your node and execute the following commands:

```
wget https://github.com/prometheus/node_exporter/releases/download/v1.1.2/node_exporter-1.1.2.linux-arm64.tar.gz
tar xvfz node_exporter-1.1.2.linux-arm64.tar.gz
cd node_exporter-1.1.2.linux-arm64
```

- 2) Since we want to run **node\_exporter** in a continuous mode in the background, we use **tmux**. This allows us to run a session in parallel in the background. Just remember that you have to execute the same commands every time you shutdown or restart the node:

```
tmux
./node_exporter
Press the following key combination ^b d (means control + b together followed by d)
tmux ls
You should now see something like this: 0: 1 windows (created Mon May 24 15:53:15 2021)
```

- 3) The **node\_exporter** is now running (in the background) on the node host machine under port 9100. At this point we need to enable this port in the firewall:

```
sudo ufw allow 9100/tcp
```

- 4) On the Monitoring server machine we now have to change the **prometheus.yml** configuration file in order to scrape metrics from the **node\_exporter** over port 9100.

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

- 5) Scroll down where the **scrape\_configs:** section start.

Within the **scrape\_configs:** section add the following parameter:

```
- job_name: 'node_exporter'
  scrape_interval: 5s
  static_configs:
    - targets: ['192.168.1.120:9100']
```

- 6) Now that we have applied these changes, save the file and close nano with the following keyboard combination:

```
^O
(Press Enter)
^X
```

- 7) Let's restart the Prometheus service with the new configuration:

```
sudo systemctl restart prometheus.service
```

- 8) We have to install the Grafana on our monitoring server now:

```
sudo apt-get install -y adduser libfontconfig1
wget https://dl.grafana.com/oss/release/grafana\_7.5.7\_arm64.deb
sudo dpkg -i grafana_7.5.7_arm64.deb
```

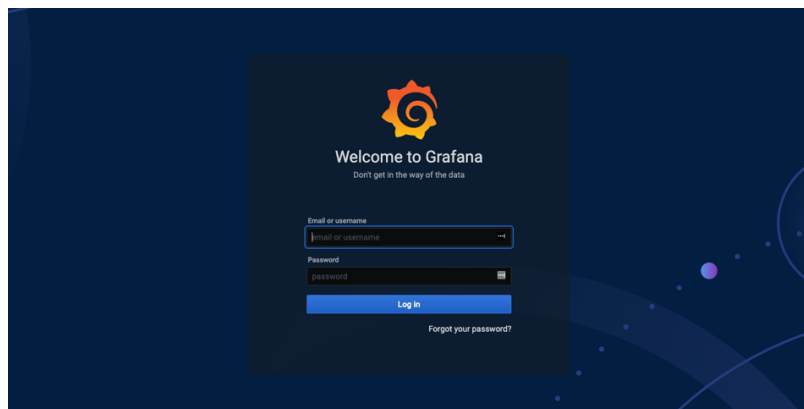
*Note: To download the latest version of Grafana visit this site:*  
<https://grafana.com/grafana/download?platform=arm>

- 9) Start Grafana-server with:

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

- 10) Open the web browser and go to **localhost:3000**

The Grafana client should appear in the browser now:



From this point onwards I refer to the official Cardano-node documentation for the configuration of the Grafana dashboard. Go to <https://docs.cardano.org/projects/cardano-node/en/latest/logging-monitoring/grafana.html> and follow the instructions in the chapter **Configuring your dashboard**.

**Important note:** The IOHK dashboard json configuration file is outdated on the Cardano-node website. MINI POOL provides an updated version for node version 1.27.0. It can be downloaded from the MINI Stake pool GitHub repository:  
[https://raw.githubusercontent.com/jterrier84/minipool/main/grafana\\_dashboard.json](https://raw.githubusercontent.com/jterrier84/minipool/main/grafana_dashboard.json)