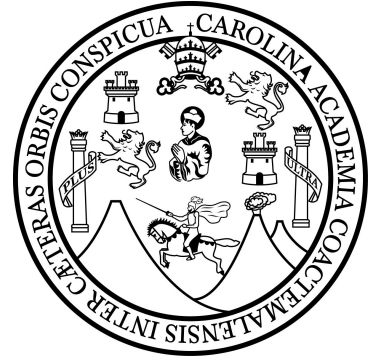


UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
LABORATORIO DE SISTEMAS OPERATIVOS 2



INSTALACIÓN Y CONFIGURACIÓN DE PROMETHEUS Y GRAFANA

Henry Adolfo Galvez - 201612499
Jose Ramiro Mateo Pu - 201603189
Herlindo René Corona Arenales - 201612219
Aux. Andrea Vicente

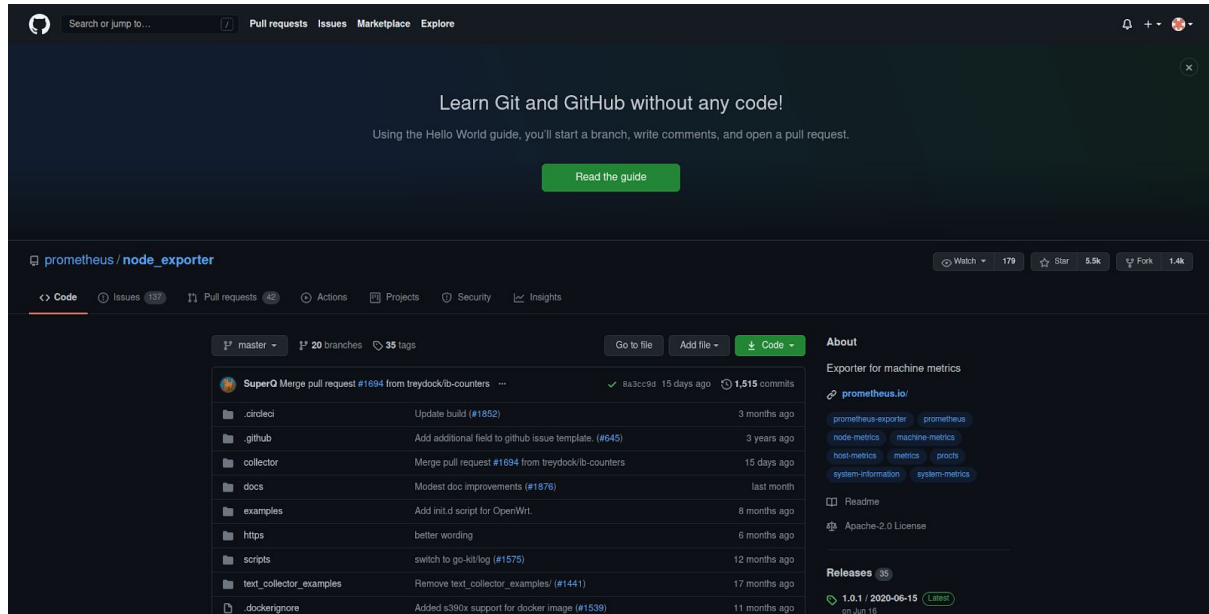
Índice

Índice	2
Descarga de Imagenes	3
Creación de contenedores	7
Prometheus node_exporter	7
Prometheus	7
Grafana	8
Creación de paneles con Prometheus	9
Uso de CPU	9
Uso de memoria	9
Uso de disco duro	10
Tráfico de entrada	10
Tráfico de salida	11
Creación de paneles con Grafana	11
Creación de data source	11
Panel de uso de CPU	13
Panel de uso de memoria	13
Panel de uso de disco duro	14
Panel de tráfico de entrada	14
Panel tráfico de salida	15
Panel Final	15

Descarga de Imagenes

Para realizar la instalación de Prometheus y Grafana haremos uso de imágenes de docker para facilitar el proceso de instalación y dedicarnos más al proceso de creación de métricas a medir dentro de la interfaz de grafana. Para poder realizar mediciones del estado actual de la máquina haremos uso de `prometheus node_exporter` el cual recolecta información del estado actual de la máquina.

Para ello realizaremos los siguientes pasos:



Ingresamos al repositorio oficial de `prometheus node_exporter`

Docker

The `node_exporter` is designed to monitor the host system. It's not recommended to deploy it as a Docker container because it requires access to the host system.

For situations where Docker deployment is needed, some extra flags must be used to allow the `node_exporter` access to the host namespaces.

Be aware that any non-root mount points you want to monitor will need to be bind-mounted into the container.

If you start container for host monitoring, specify `path.rootfs` argument. This argument must match path in bind-mount of host root. The `node_exporter` will use `path.rootfs` as prefix to access host filesystem.

```
docker run -d \
  --net="host" \
  --pid="host" \
  -v "/:/host:ro,rslave" \
  quay.io/prometheus/node-exporter:latest \
  --path.rootfs=/host
```

Buscamos la instrucciones para identificar la imagen de `node_exporter` que se desea bajar para hacer uso de docker

```
hr@hr: ~$ sudo docker pull quay.io/prometheus/node-exporter:latest
[sudo] password for hr:
latest: Pulling from prometheus/node-exporter
86fa074c6765: Pull complete
ed1cd1c6cd7a: Pull complete
ff1bb132ce7b: Pull complete
Digest: sha256:cf66a6bbd573fd819ea09c72e21b528e9252d58d01ae13564a29749de1e48e0f
Status: Downloaded newer image for quay.io/prometheus/node-exporter:latest
quay.io/prometheus/node-exporter:latest
hr@hr:~$
```

Bajamos la imagen de node_exporter

The screenshot shows the official Prometheus website. The navigation bar includes links for Docs, Download, Community, Support & Training, and Blog. The main content area is titled 'INSTALLATION' and is divided into two columns. The left column contains a sidebar with links to various sections: Introduction, Concepts, Prometheus (selected), Visualization, Operating, Instrumenting, Alerting, and Best Practices. The right column contains the 'INSTALLATION' section, which includes a 'Using pre-compiled binaries' section, a 'From source' section, and a 'Using Docker' section. The 'Using Docker' section provides instructions on how to run Prometheus using Docker, including a code block for the command: `docker run -p 9090:9090 prom/prometheus`. It also mentions that the Prometheus image uses a volume to store metrics and provides a link to the 'Volumes & bind-mount' section.

Nos dirigimos a la página oficial de Prometheus y buscamos la instrucciones de instalación

Using Docker

◦ SaltStack

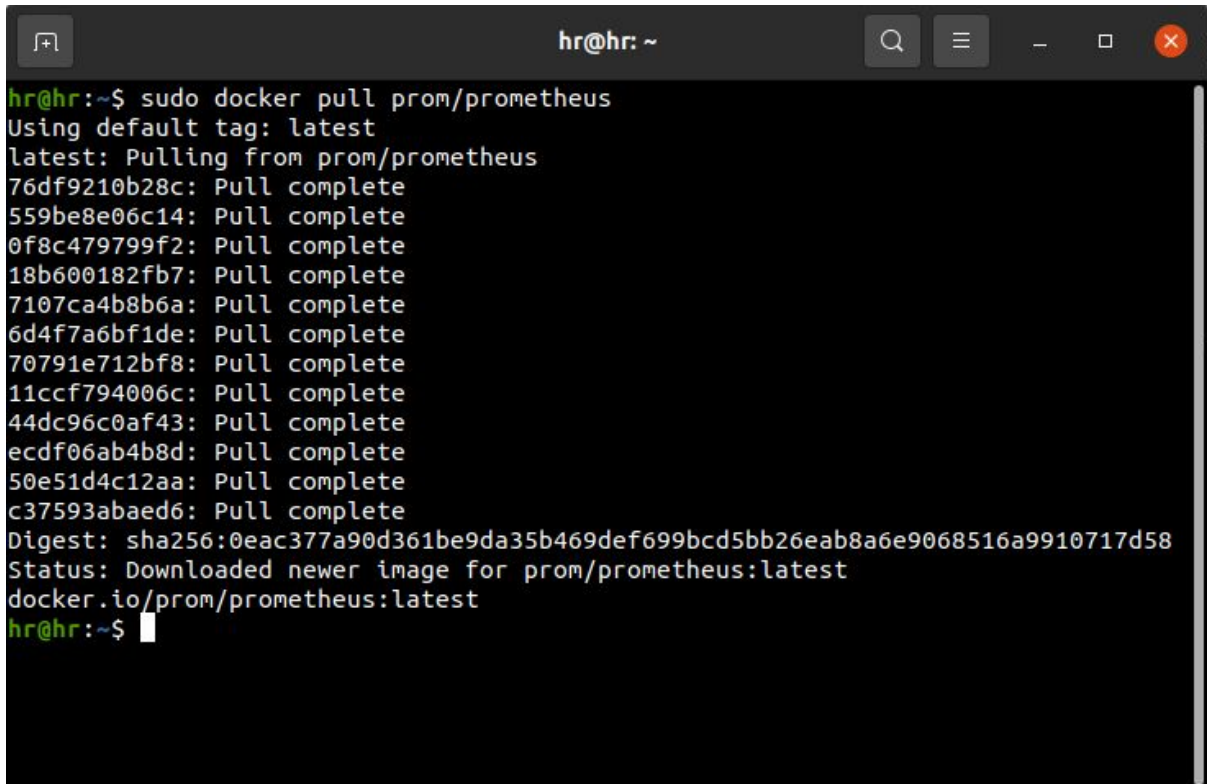
All Prometheus services are available as Docker images on [Quay.io](#) or [Docker Hub](#).

Running Prometheus on Docker is as simple as `docker run -p 9090:9090 prom/prometheus`. This starts Prometheus with a sample configuration and exposes it on port 9090.

The Prometheus image uses a volume to store the actual metrics. For production deployments it is highly recommended to use a [named volume](#) to ease managing the data on Prometheus upgrades.

To provide your own configuration, there are several options. Here are two examples.

Buscamos el nombre de la imagen que debemos descargar para hacer uso de prometheus

A terminal window with a dark background and light-colored text. The window title is 'hr@hr: ~'. The command 'sudo docker pull prom/prometheus' has been executed. The output shows the image being pulled from Docker Hub, listing various layers and their pull status as 'complete'. It also displays the image's digest and status, indicating it was a newer version. The prompt 'hr@hr:~\$' is visible at the bottom.

```
hr@hr:~$ sudo docker pull prom/prometheus
Using default tag: latest
latest: Pulling from prom/prometheus
76df9210b28c: Pull complete
559be8e06c14: Pull complete
0f8c479799f2: Pull complete
18b600182fb7: Pull complete
7107ca4b8b6a: Pull complete
6d4f7a6bf1de: Pull complete
70791e712bf8: Pull complete
11ccf794006c: Pull complete
44dc96c0af43: Pull complete
ecdf06ab4b8d: Pull complete
50e51d4c12aa: Pull complete
c37593abaed6: Pull complete
Digest: sha256:0eac377a90d361be9da35b469def699bcd5bb26eab8a6e9068516a9910717d58
Status: Downloaded newer image for prom/prometheus:latest
docker.io/prom/prometheus:latest
hr@hr:~$
```

Descargamos la imagen de Prometheus

[Home](#) > [Installation](#) > Run Grafana Docker image

Run Grafana Docker image

You can install and run Grafana using the official Docker image. It comes in two variants: Alpine and Ubuntu.

This page also contains important information about [migrating from earlier Docker image versions](#).

Alpine image (recommended)

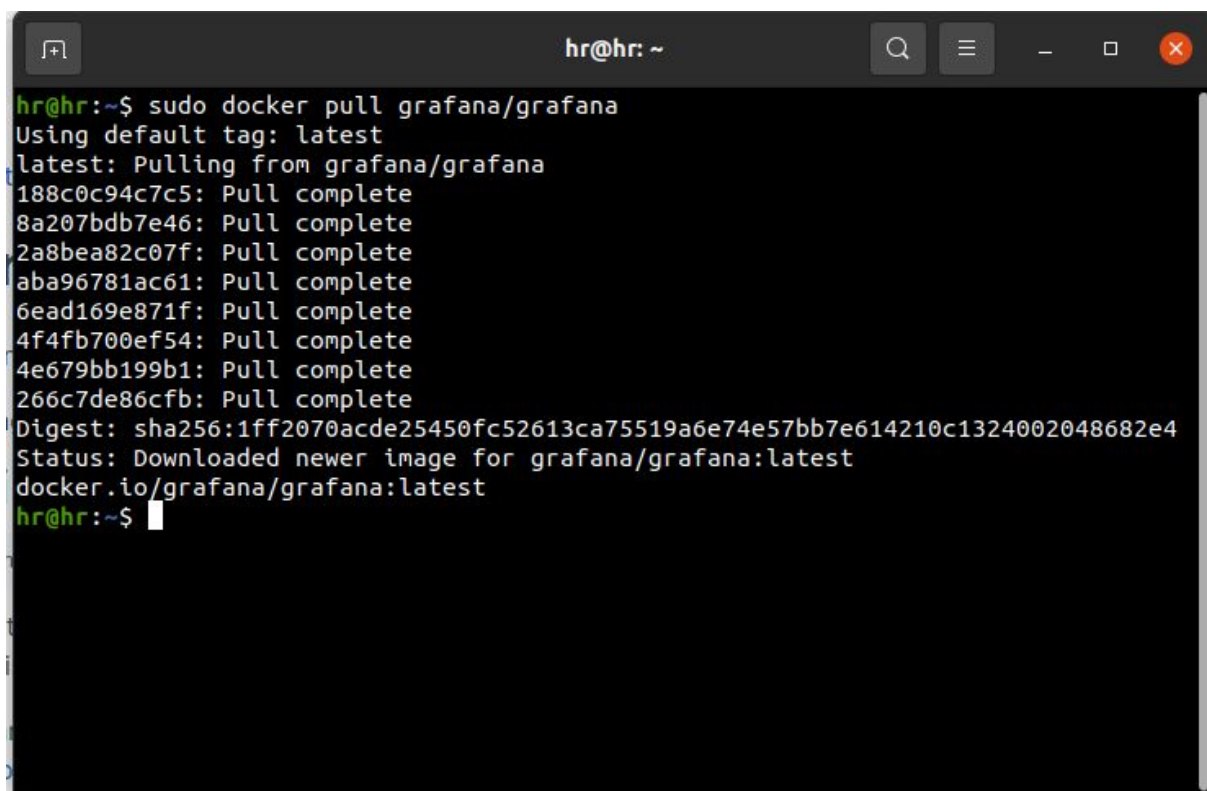
```
grafana/grafana:<version>
```

This is the default image. It's based on the popular [Alpine Linux project](#), available in [the Alpine official image](#). Alpine Linux is much smaller than most distribution base images, and thus leads to slimmer and more secure images.

This variant is highly recommended when security and final image size being as small as possible is desired. The main caveat to note is that it uses [musl libc](#) instead of [glibc and friends](#), so certain software might run into issues depending on the depth of their libc requirements. However, most software don't have an issue with this, so this variant is usually a very safe choice.

Note: The `grafana/grafana:<version>` image was based on [Ubuntu](#) before version 6.4.0.

Nos dirigimos a la página oficial de grafana para identificar el nombre de la imagen de grafana a descarga en docker

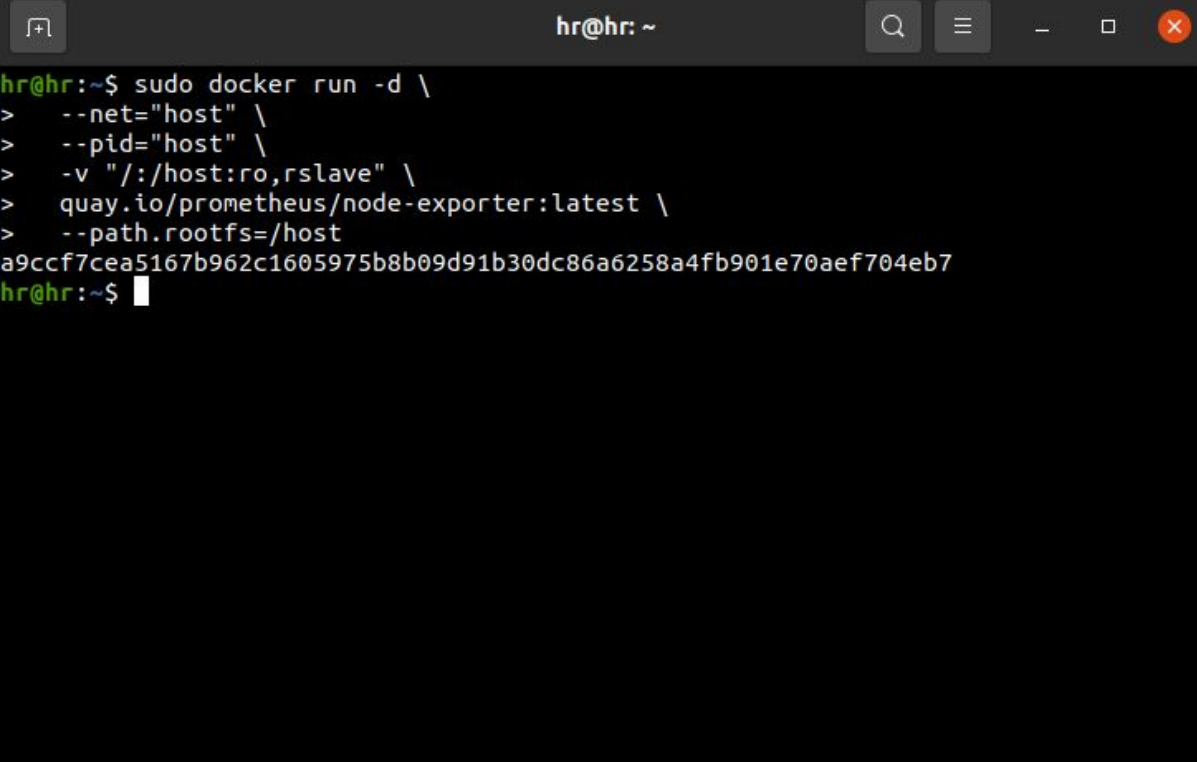


```
hr@hr: ~  
hr@hr:~$ sudo docker pull grafana/grafana  
Using default tag: latest  
latest: Pulling from grafana/grafana  
188c0c94c7c5: Pull complete  
8a207bdb7e46: Pull complete  
2a8bea82c07f: Pull complete  
aba96781ac61: Pull complete  
6ead169e871f: Pull complete  
4f4fb700ef54: Pull complete  
4e679bb199b1: Pull complete  
266c7de86cfb: Pull complete  
Digest: sha256:1ff2070acde25450fc52613ca75519a6e74e57bb7e614210c1324002048682e4  
Status: Downloaded newer image for grafana/grafana:latest  
docker.io/grafana/grafana:latest  
hr@hr:~$
```

Descargamos la imagen de grafana

Creación de contenedores

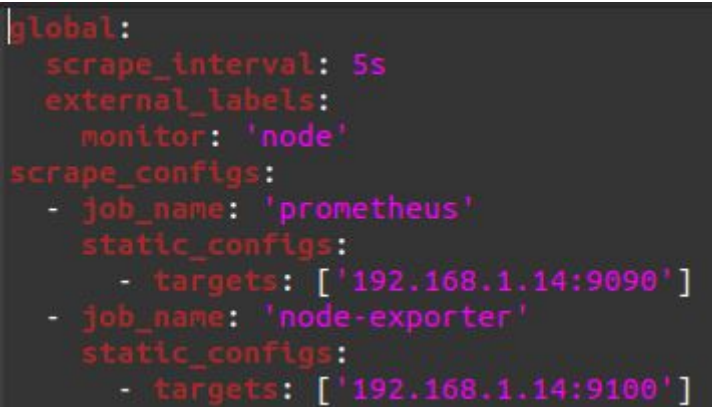
Prometheus node_exporter

A terminal window with a dark background and light green text. The window title is 'hr@hr: ~'. The command being executed is 'sudo docker run -d \ --net="host" \ --pid="host" \ -v "/:/host:ro,rslave" \ quay.io/prometheus/node-exporter:latest \ --path.rootfs=/host'. The output shows a long container ID 'a9ccf7cea5167b962c1605975b8b09d91b30dc86a6258a4fb901e70aef704eb7' followed by a new prompt 'hr@hr:~\$'.

```
hr@hr:~$ sudo docker run -d \  
> --net="host" \  
> --pid="host" \  
> -v "/:/host:ro,rslave" \  
> quay.io/prometheus/node-exporter:latest \  
> --path.rootfs=/host  
a9ccf7cea5167b962c1605975b8b09d91b30dc86a6258a4fb901e70aef704eb7  
hr@hr:~$
```

Creamos el contenedor haciendo uso de la herramienta de docker indicando que la red corresponderá al host

Prometheus

A code block showing a YAML configuration for Prometheus. The configuration includes a 'global' section with 'scrape_interval' set to '5s' and 'external_labels' with 'monitor' set to 'node'. The 'scrape_configs' section contains two jobs: 'prometheus' with a target at '192.168.1.14:9090' and 'node-exporter' with a target at '192.168.1.14:9100'.

```
global:  
  scrape_interval: 5s  
  external_labels:  
    monitor: 'node'  
scrape_configs:  
  - job_name: 'prometheus'  
    static_configs:  
      - targets: ['192.168.1.14:9090']  
  - job_name: 'node-exporter'  
    static_configs:  
      - targets: ['192.168.1.14:9100']
```

Previo a iniciar con la creación del contenedor creamos un archivo de tipo yaml con las siguientes configuraciones indicando que se hará uso de node-exporter y de la ip de la máquina host


```
hr@hr: ~/Desktop/prometheus
hr@hr:~/Desktop/prometheus$ sudo docker run -d -p 9090:9090 -v $PWD/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus
80d081e9cfd5a6fee17cfa11256abdb8802931b8694644c0ea186d52855952f1
hr@hr:~/Desktop/prometheus$
```

Creamos el contenedor exponiendo el puerto 9090 y enlazando el archivo prometheus.yml con el del contenedor

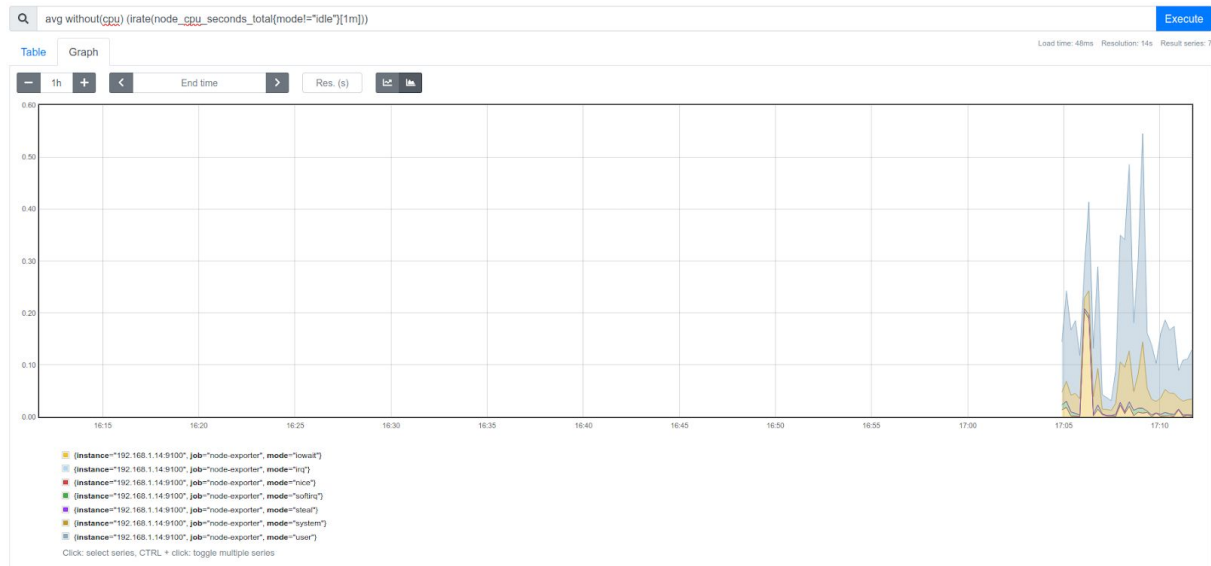
Grafana

```
hr@hr: ~/Desktop/prometheus
hr@hr:~/Desktop/prometheus$ sudo docker run -d -p 3000:3000 grafana/grafana
d5bf03706904b1dfc62f5c5b6e976cbbc5212897eb573b7ba1fd683cc7c0a4de
hr@hr:~/Desktop/prometheus$
```

Creamos el contenedor de grafana exponiendo el puerto 3000

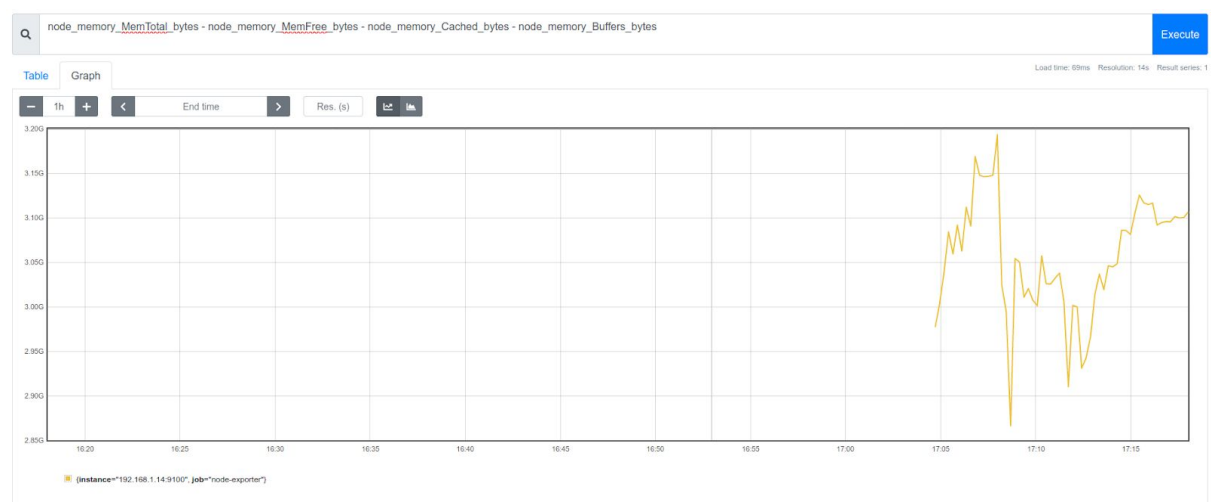
Creación de paneles con Prometheus

Uso de CPU



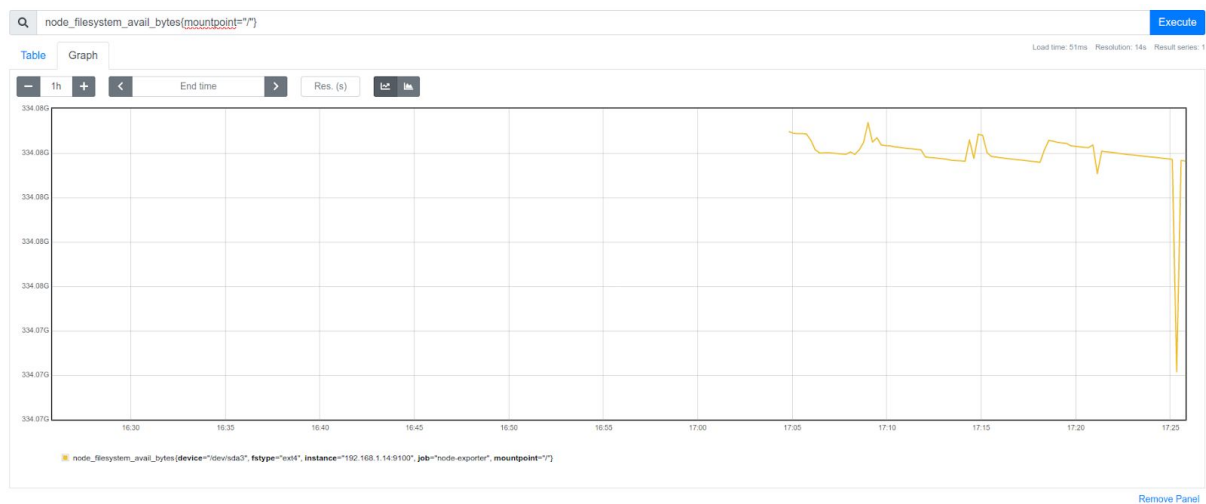
Se utiliza el siguiente query para filtrar la información creada desde node exporter de esta manera podemos ver el uso promedio de todos los cpu de la computadora separados por tipo ya sea uso de sistema, usuario, iowait, etc.

Uso de memoria



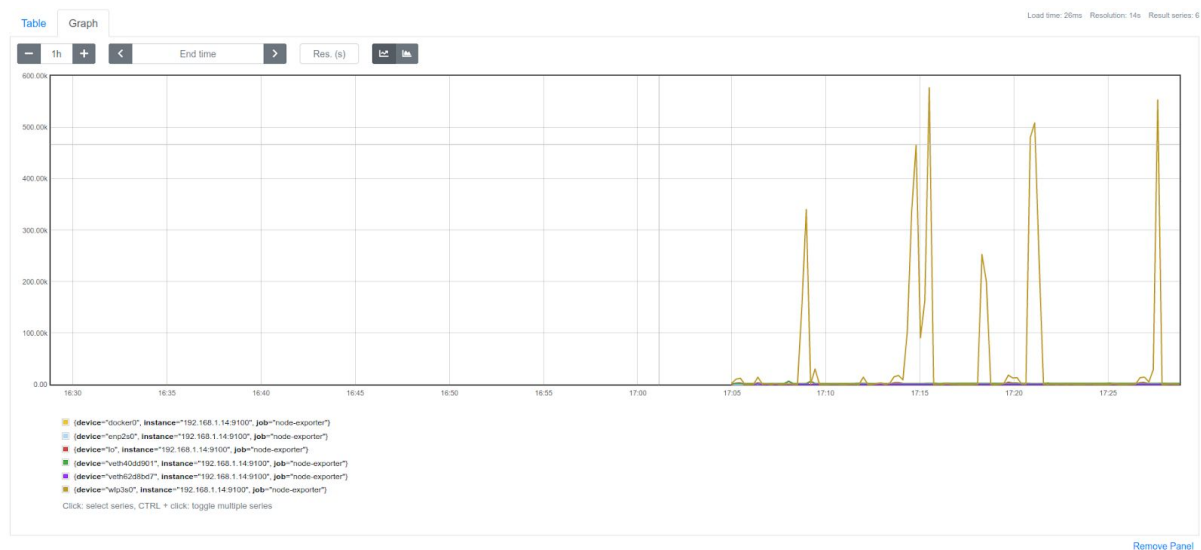
Al igual que con el uso del cpu se utiliza un query para filtrar la información, acá lo que se hace es restar el espacio de memoria libre y la memoria de uso específico a la memoria total utilizada

Uso de disco duro



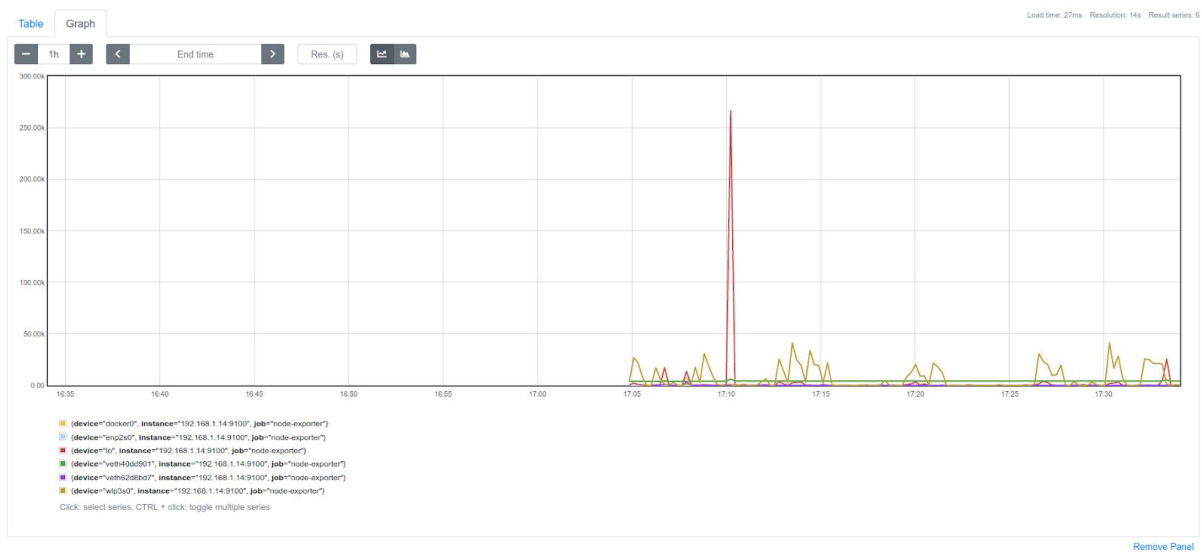
Para realizar un uso de la memoria lo que se realiza es un query con el espacio libre realizando una modificación para que solo muestra información acerca del punto de montaje del sistema operativo

Tráfico de entrada



Para el tráfico de entrada se hace uso de node_exporter y se ve todo el tráfico de entrada de todos los adaptadores en el sistema

Tráfico de salida

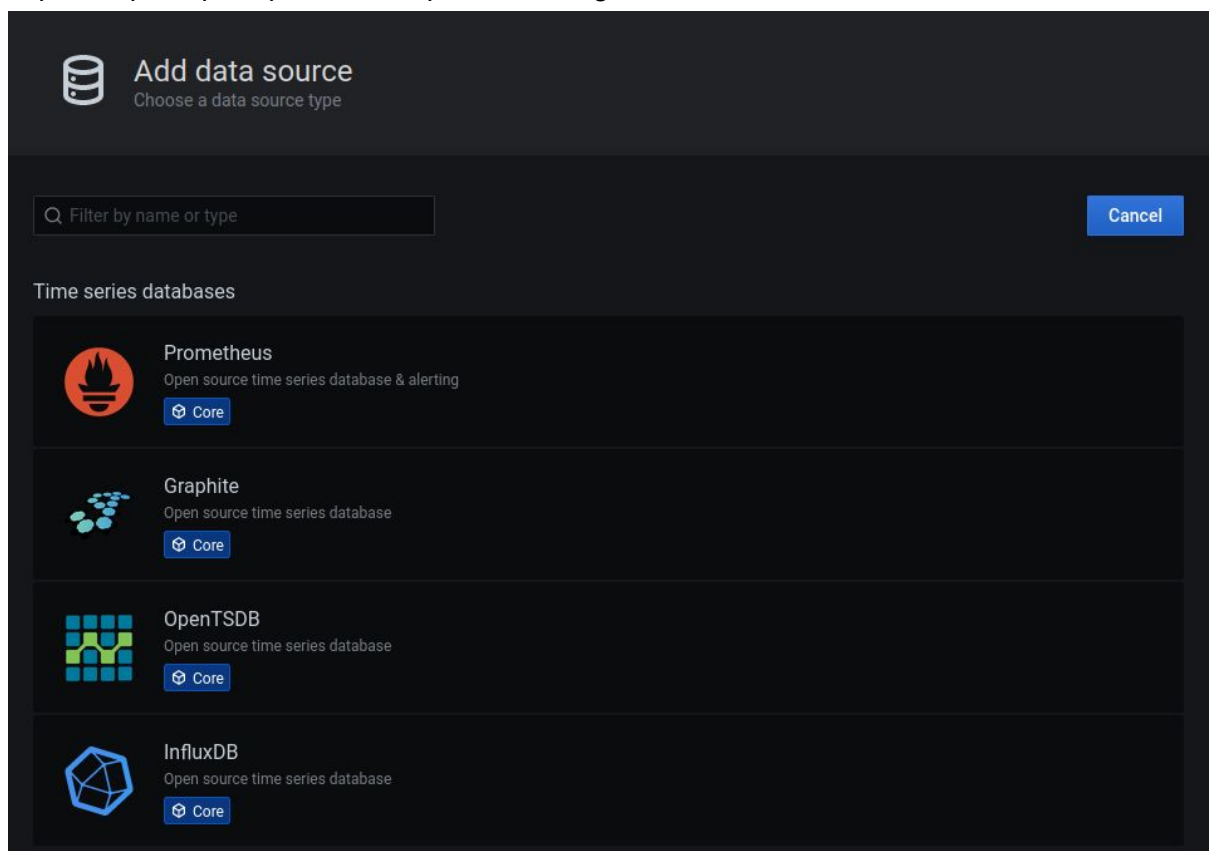



Para el tráfico de salida se hace uso de node_exporter y se ve el tráfico de salida de todos los adaptadores del sistema

Creación de paneles con Grafana

Creación de data source

El primer paso para poder crear paneles con grafana consiste en crear un nuevo datasource





Data Sources / Prometheus

Type: Prometheus

Settings

Dashboards

Name ⓘ

Prometheus

Default

☒

HTTP

URL ⓘ

http://localhost:9090

Access

Browser ▾

[Help >](#)

Auth

Basic auth

☐

With Credentials ⓘ

☐

Scrape interval ⓘ

15s

Query timeout ⓘ

60s

HTTP Method ⓘ

Choose ▾

Misc

Disable metrics lookup ⓘ

☐

Custom query parameters ⓘ

Example: max_source_resolution=5m&timeout=10

Save & Test

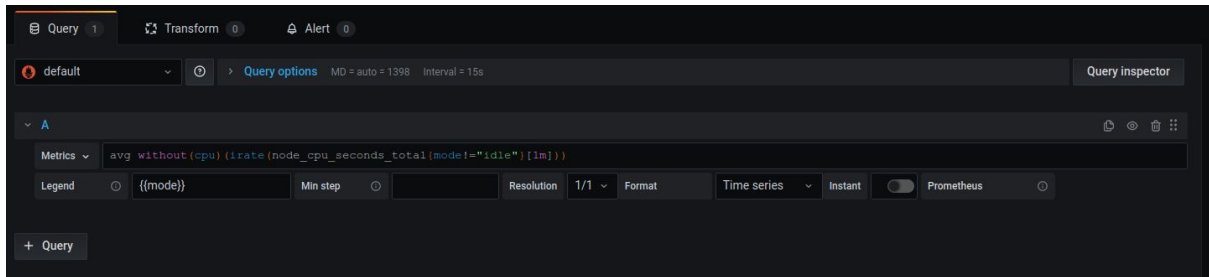
Delete

Back

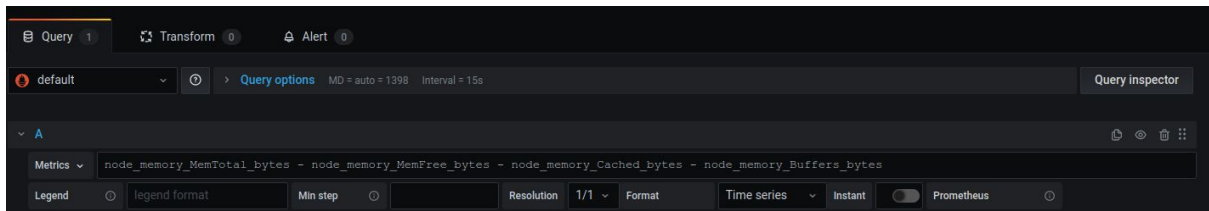
En Grafana se configura indicando el puerto en el que se encuentra funcionando Prometheus, e indicando que el acceso es haciendo uso del navegador o proxy

Panel de uso de CPU

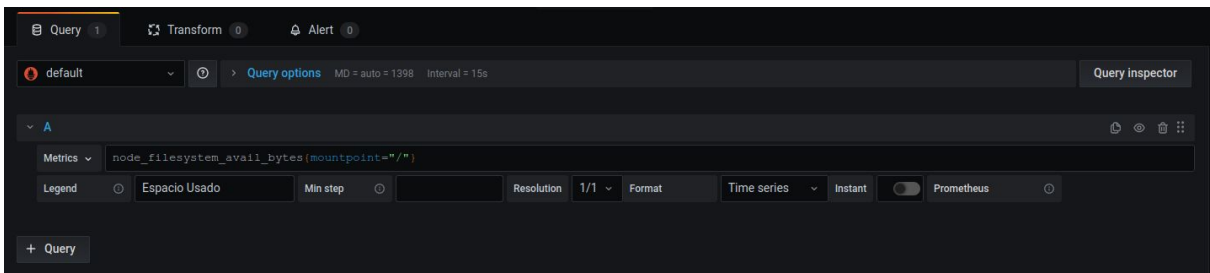
La ventaja del uso de Grafana es que el query es exactamente igual a el query de Prometheus.



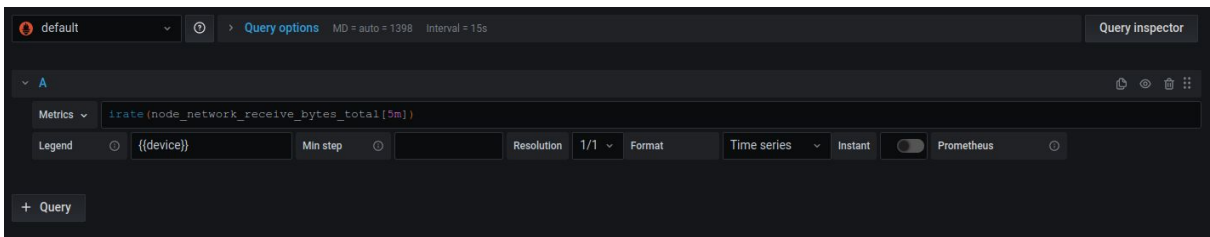
Panel de uso de memoria



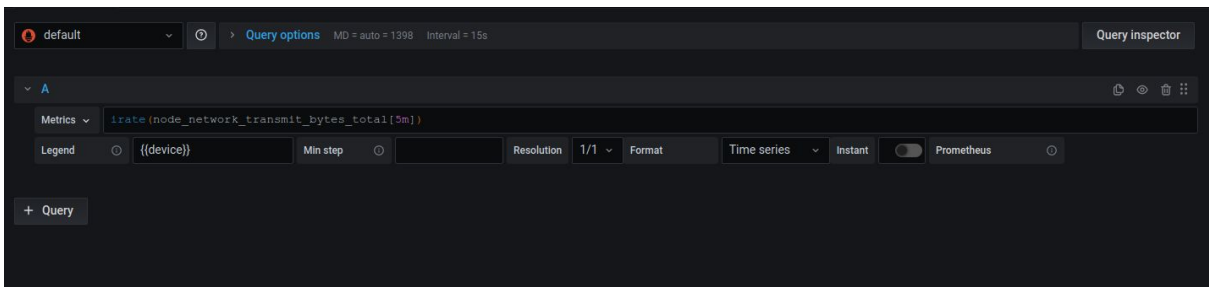
Panel de uso de disco duro



Panel de tráfico de entrada



Panel tráfico de salida



Panel Final

