



UNIVERSIDAD DE GRANADA

Centro de Procesamiento de Datos

Práctica 10 – Monitorización de Recursos con Grafana

Arturo Alonso Carbonero

Creación de las máquinas virtuales

Para el desarrollo de esta práctica, se han empleado las mismas máquinas con Ubuntu que para las prácticas anteriores. En la imagen siguiente se puede ver cómo se han creado las mismas empleando Vagrant. Concretamente, se han creado dos máquinas, **Nodo1P10** y **Nodo2P10**, de las cuales la primera actuará como máquina principal.

Dentro del *Vagrantfile* hay algunas opciones innecesarias, pero se ha optado por mantener la estructura exacta de los ficheros anteriores, únicamente modificando el nombre las máquinas así como sus direcciones IP.

```
Vagrantfile
1  vagrant.configure("2") do |config|
2    config.vm.box = "ubuntu/focal64"
3    config.vm.provision "shell", inline: <<-SHELL
4    config.vm.box_download_insecure = true
5      sed -i 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd_config
6      systemctl restart sshd.service
7      echo "192.168.56.19 nodo1P10" >> /etc/hosts
8    SHELL
9
10   config.vm.define :nodo1P10 do |ub_config|
11     ub_config.vm.hostname = "nodo1P10.vm"
12     ub_config.vm.network "private_network" , ip:"192.168.56.19"
13     ub_config.vm.provider :virtualbox do |vb|
14       vb.name = "nodo1P10"
15       vb.customize ["modifyvm", :id, "--memory", "1024"]
16       vb.customize ["modifyvm", :id, "--cpus", "1"]
17     end
18   end
19
20   config.vm.define :nodo2P10 do |ub_config|
21     ub_config.vm.hostname = "nodo2P10.vm"
22     ub_config.vm.network "private_network" , ip:"192.168.56.20"
23     ub_config.vm.provider :virtualbox do |vb|
24       vb.name = "nodo2P10"
25       vb.customize ["modifyvm", :id, "--memory", "1024"]
26       vb.customize ["modifyvm", :id, "--cpus", "1"]
27     end
28   end
29 end
```

Vagrantfile para las máquinas

Instalación de los componentes

Para crear los contenedores con Grafana e InfluxDB, se ha optado por crear un fichero *docker-compose.yml* para lanzar directamente ambos contenedores con las imágenes pertinentes. La creación de las máquinas ha sido lo más sencilla posible, indicando únicamente las imágenes, los puertos necesarios y la red, tal y como se ve en la siguiente imagen.

```

C:\ vagrant@nodo1P10: ~
GNU nano 4.8                                docker
version: "3"

networks:
  monitoring:

services:
  influxdb:
    image: influxdb:latest
    ports:
      - 8086:8086
    networks:
      - monitoring

  grafana:
    image: grafana/grafana:9.0.4
    ports:
      - 3000:3000
    networks:
      - monitoring

```

Fichero docker-compose.yml

Para lanzar los contenedores, como hemos visto en prácticas anteriores, ejecutamos el comando **docker-compose up -d**. La opción **-d** es para que los contenedores se ejecuten en segundo plano.

```

vagrant@nodo1P10:~/practica10$ sudo docker-compose up -d
[+] Running 21/21
  grafana 9 layers [00000000]    0B/0B    Pulled                269.7s
    ab6db1bc80d0 Pull complete    94.9s
    ae97cf61a993 Pull complete    95.7s
    3cdf1610ea9a Pull complete   100.9s
    4c449d4b5982 Pull complete   102.1s
    5afbacc101c13 Pull complete   105.5s
    8c10b181c7ac Pull complete   265.2s
    6e819c7a6bad Pull complete   265.8s
    1b7e4f37d00f Pull complete   266.0s
    069d748ea1ea Pull complete   266.3s
  influxdb 10 layers [00000000]    0B/0B    Pulled                257.4s
    1f7ce2fa46ab Pull complete   112.1s
    a1e16ccbb43e Pull complete   119.2s
    f63a7bc1934b Pull complete   121.3s
    0b0cc3c0c6d9 Pull complete   122.0s
    6061761101c0 Pull complete   123.0s
    68bd7f35f519 Pull complete   235.8s
    42be173071fe Pull complete   245.4s
    883524767a4d Pull complete   246.8s
    5fbf9c48eb4f Pull complete   247.7s
    2fee5c5d2531 Pull complete   248.3s

```

Obtención de las imágenes

```

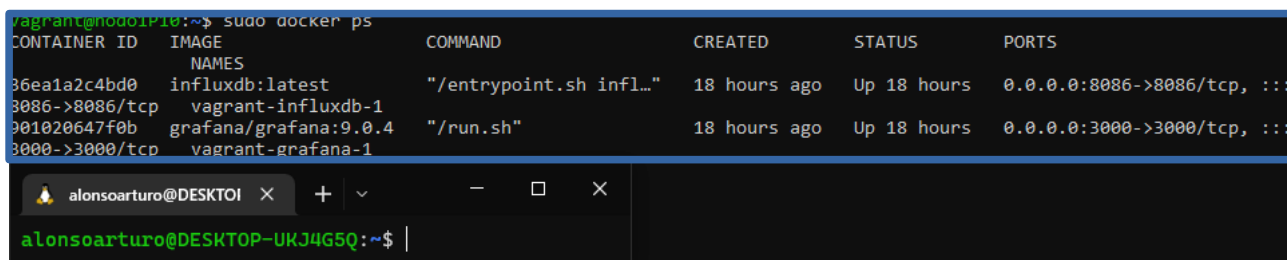
[+] Running 2/2
  Container vagrant-grafana-1   Started                8.2s
  Container vagrant-influxdb-1 Started                8.2s

```

Contenedores creados

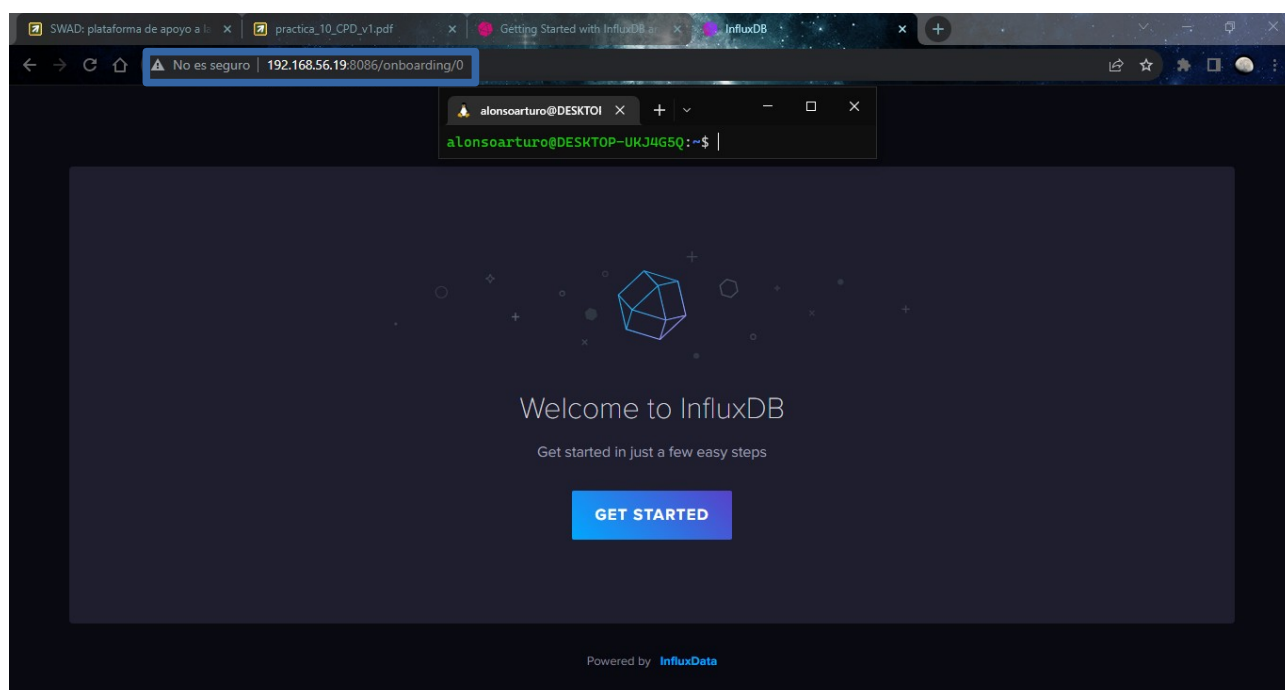
Podemos hacer uso del comando **docker ps** para comprobar que los contenedores se han creado correctamente y que se encuentran en ejecución.

```
alonsoarturo@node01p10:~$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
36ea1a2c4bd0   influxdb:latest                     "/entrypoint.sh infl..." 18 hours ago   Up 18 hours   0.0.0.0:8086->8086/tcp, :::
3086->8086/tcp   vagrant-influxdb-1
901020647f0b   grafana/grafana:9.0.4              "/run.sh"               18 hours ago   Up 18 hours   0.0.0.0:3000->3000/tcp, :::
3000->3000/tcp   vagrant-grafana-1
```

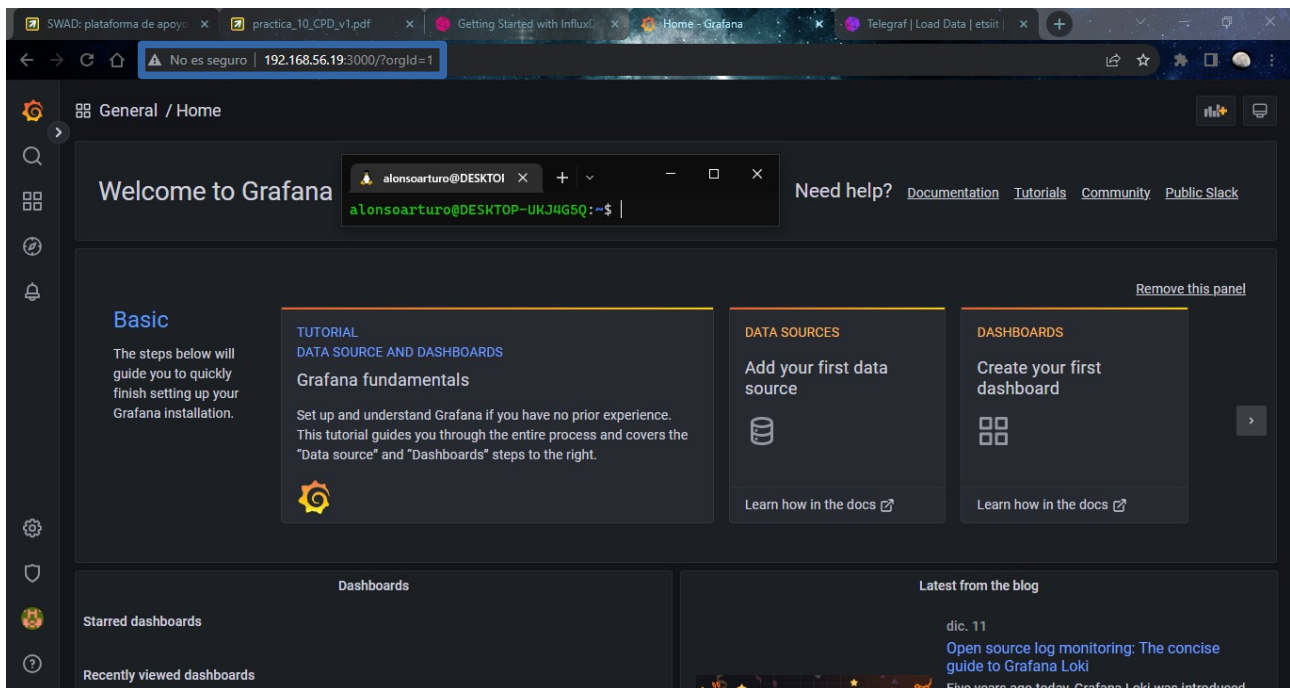


Contenedores en ejecución

Si el procedimiento ha sido el correcto y existe conectividad entre el anfitrión y la máquina, podemos acceder, desde un navegador, a las interfaces de Grafana e InfluxDB a través de los puertos 3000 y 8086 respectivamente, empleando la dirección IP del nodo principal.



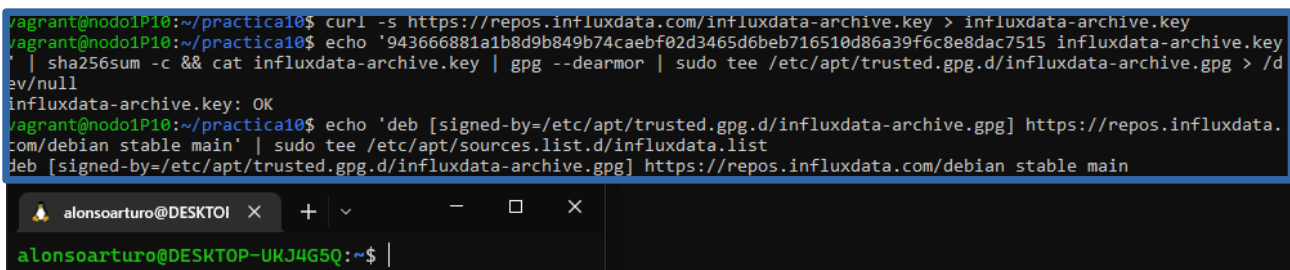
Acceso a InfluxDB



Acceso a Grafana

Para instalar Telegraf, es necesario hacerlo desde el repositorio de InfluxDB. En la imagen siguiente se muestra como hacerlo.

En primer lugar, obtenemos, mediante **curl**, la clave de firma de InfluxDB. Comprobamos, mediante **echo** y **sha256sum**, la integridad de la clave con SHA-256. Después, hacemos uso de **cat** y **tee** para desempaquetar la clave GPG y agregarla a las claves de confianza. Mediante **echo** y **tee**, añadimos el repositorio de InfluxDB a la lista de recursos de APT. Finalmente, mediante **apt**, actualizamos los paquetes e instalamos Telegraf directamente en el nodo (sin hacer uso de contenedores). Repetimos el proceso en la segunda máquina siguiendo los mismos pasos.



Instalación de Telegraf

En este punto, ya disponemos de las herramientas necesarias para realizar la práctica. Sin embargo, hay que realizar ciertas configuraciones para poder comenzar a manejar los datos que Telegraf recoja. En primer lugar, ejecutamos el comando **influx setup** en el contenedor, mediante **docker exec -it**, y creamos un usuario inicial, el cual dispondrá de todos los permisos, un token para todos los accesos, una organización y un *bucket*.

```
vagrant@nodo1P10: ~  
vagrant@nodo1P10:~$ sudo docker exec -it 36ea1a2c4bd0 influx setup --name influxP10 --host http://192.168.56.19:8086 -u admin -p adminP10 -o etsiit -b influx10 -t tokenP10 -r 0 -f  
User      Organization  Bucket  
admin     etsiit        influx10
```

Configuración para InfluxDB

Mediante las credenciales recientemente creadas, accedemos a la interfaz de InfluxDB y accedemos al apartado de Telegraf. En él, podemos generar de forma automática diferentes configuraciones para el mismo sin tener que modificar de forma manual el fichero de configuración. En este caso, el *input* será el proporcionado por el plugin **CPU**, que ofrece información sobre la CPU del sistema, y el *output* será el contenedor de InfluxDB de la arquitectura. De esta forma, generamos un token de API para poder aplicar la configuración en la máquina principal mediante el comando **telegraf --config**.

Create a Telegraf Configuration

Your API token is required for pushing data into InfluxDB. You can copy the following command to your terminal window to set an environment variable with your API token.

```
export INFLUX_TOKEN=Krw7rYh5NDnJH3mBis0veLz5cgB_Gob1LZ95g9-w_rD4rrdIo7JEtAxcpp7qxp6tZqCQA1qUCqsdeIxtjkgQ==
```

COPY TO CLIPBOARD

GENERATE NEW API TOKEN

CLI

3. Start Telegraf

Finally, you can run the following command to start the Telegraf agent running on your machine.

```
telegraf --config http://192.168.56.19:8086/api/v2/telegrafs/0c473cc03a903000
```

COPY TO CLIPBOARD

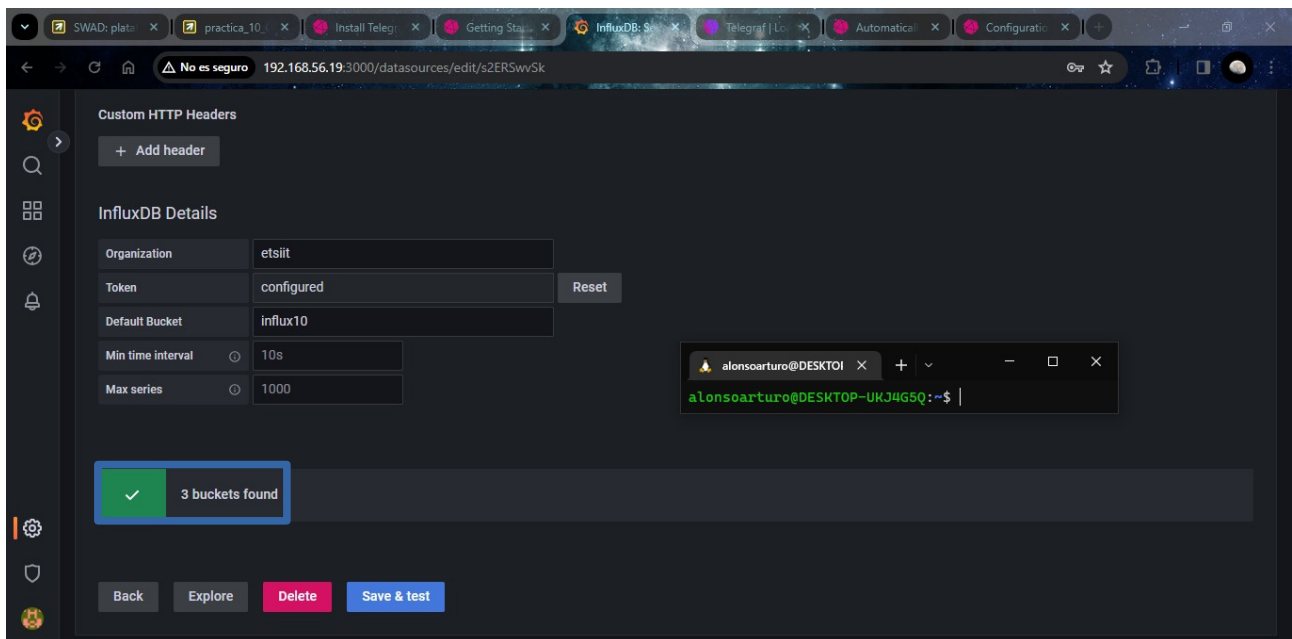
CLI

FINISH

alonsoarturo@DESKTOI x + - □ x
alonsoarturo@DESKTOP-UKJ4G5Q:~\$

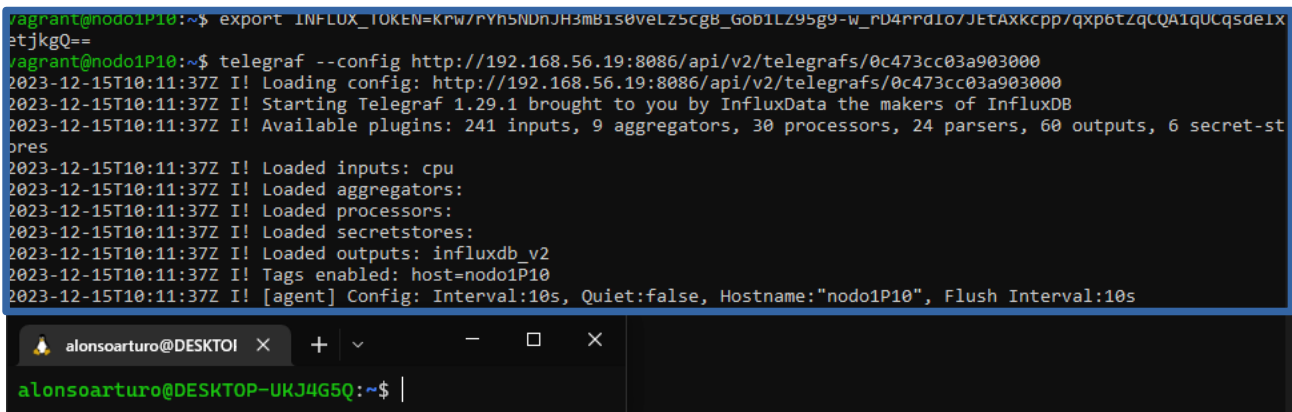
Token de la configuración

En la interfaz de Grafana, accedemos al apartado de configuración para agregar una fuente de datos. Seleccionamos el *bucket* e indicamos las credenciales, la organización y el token creados anteriormente. Si Grafana es capaz de ver la fuente, tras aplicar la selección, indicará que se han encontrado tres *buckets*.



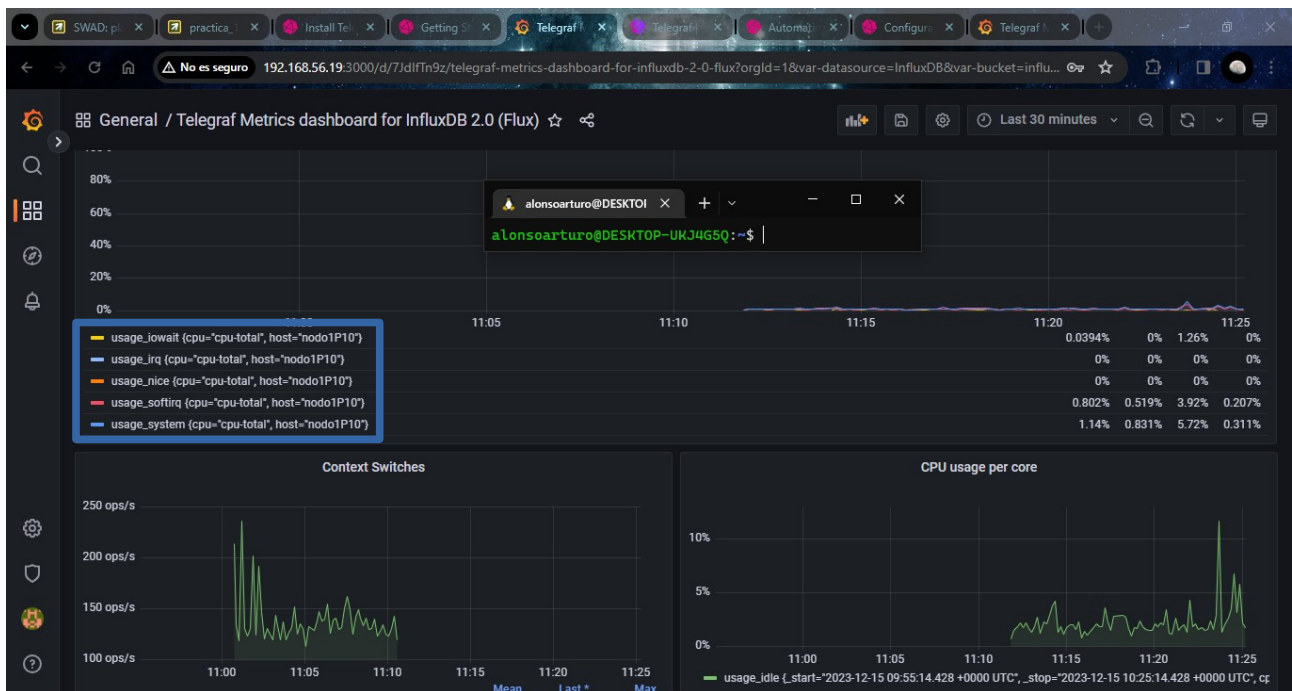
Fuente de datos

Ahora, si ejecutamos el comando para aplicar la configuración, en ambas máquinas, Telegraf comenzará a recolectar información y la enviará a InfluxDB.

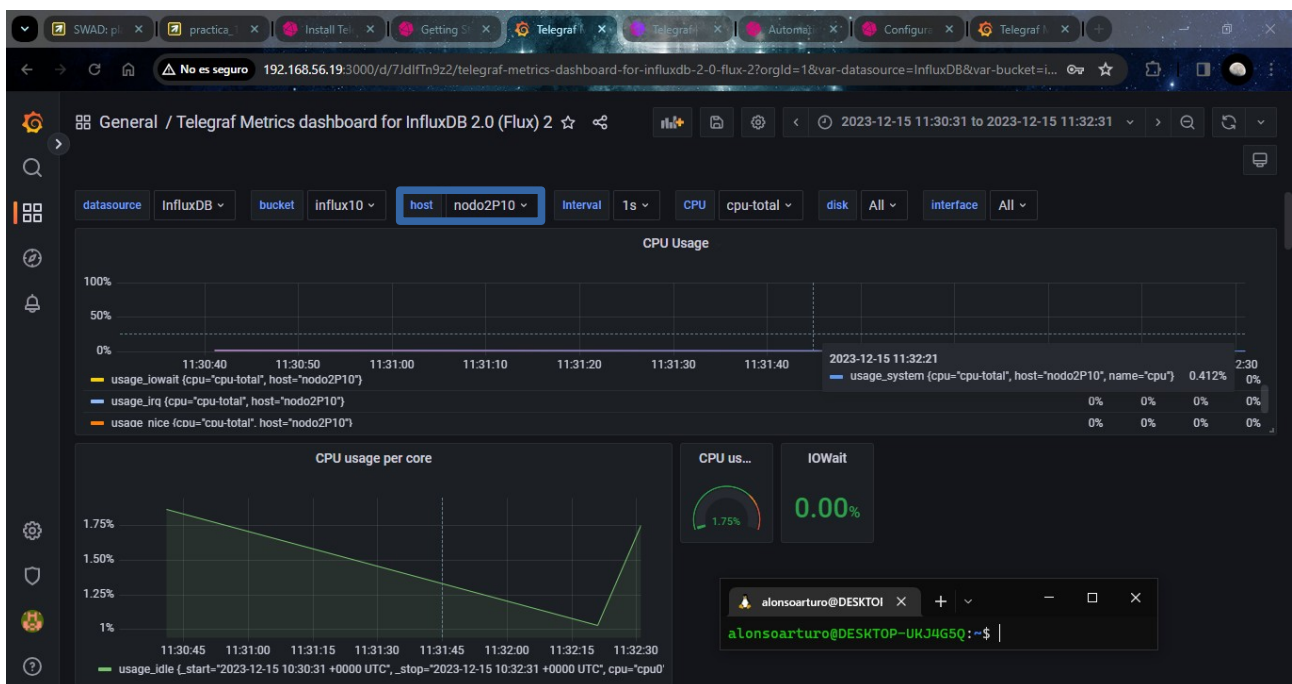


Aplicación de la configuración

Finalmente, en la interfaz de Grafana, podemos tanto crear como importar *dashboards*, en las cuales podemos añadir paneles para mostrar la información que nos interese. En este caso, se muestra información básica de la CPU de ambos nodos.



Información del nodo principal



Información del nodo secundario

Referencias

- <https://lovethepenguin.com/docker-how-to-work-with-grafana-and-influxdb-188a78d61b7>
- <https://docs.influxdata.com/influxdb/v1/administration/ports/>
- <https://www.influxdata.com/blog/getting-started-influxdb-grafana/>
- <https://docs.influxdata.com/telegraf/v1/configuration/>
- <https://docs.influxdata.com/telegraf/v1/install/>
- <https://docs.influxdata.com/influxdb/v2/write-data/no-code/use-telegraf/manual-config/>
- <https://docs.ockam.io/guides/examples/telegraf-+-influxdb>
- <https://www.influxdata.com/blog/how-to-setup-influxdb-telegraf-and-grafana-on-docker-part-1/>
- <https://www.influxdata.com/blog/how-to-setup-influxdb-telegraf-and-grafana-on-docker-part-2/>
- <https://www.alvarovf.com/sistemas/openstack/2021/02/09/metricas-telegraf-influxdb-grafana.html>
- <https://jorgedelacruz.uk/2021/04/14/looking-for-the-perfect-dashboard-influxdb-telegraf-and-grafana-part-i-installing-influxdb-telegraf-and-grafana-on-ubuntu-20-04-lts/>
- <https://sbcode.net/grafana/install-telegraf-agent/>
- <https://docs.influxdata.com/influxdb/v2/write-data/no-code/use-telegraf/auto-config/>
- https://hub.docker.com/_/influxdb
- <https://grafana.com/grafana/dashboards/15650-telegraf-influxdb-2-0-flux/>