

Práctica 10: Monitorización de recursos con Grafana

Autor: Manuel Díaz-Meco Terrés

Fecha: 19 diciembre de 2024

Introducción

En esta práctica se ha implementado un sistema de monitorización de recursos utilizando Telegraf, InfluxDB y Grafana. El objetivo es recopilar métricas de dos máquinas virtuales, almacenarlas en una base de datos de series temporales y visualizarlas mediante gráficos interactivos en Grafana.

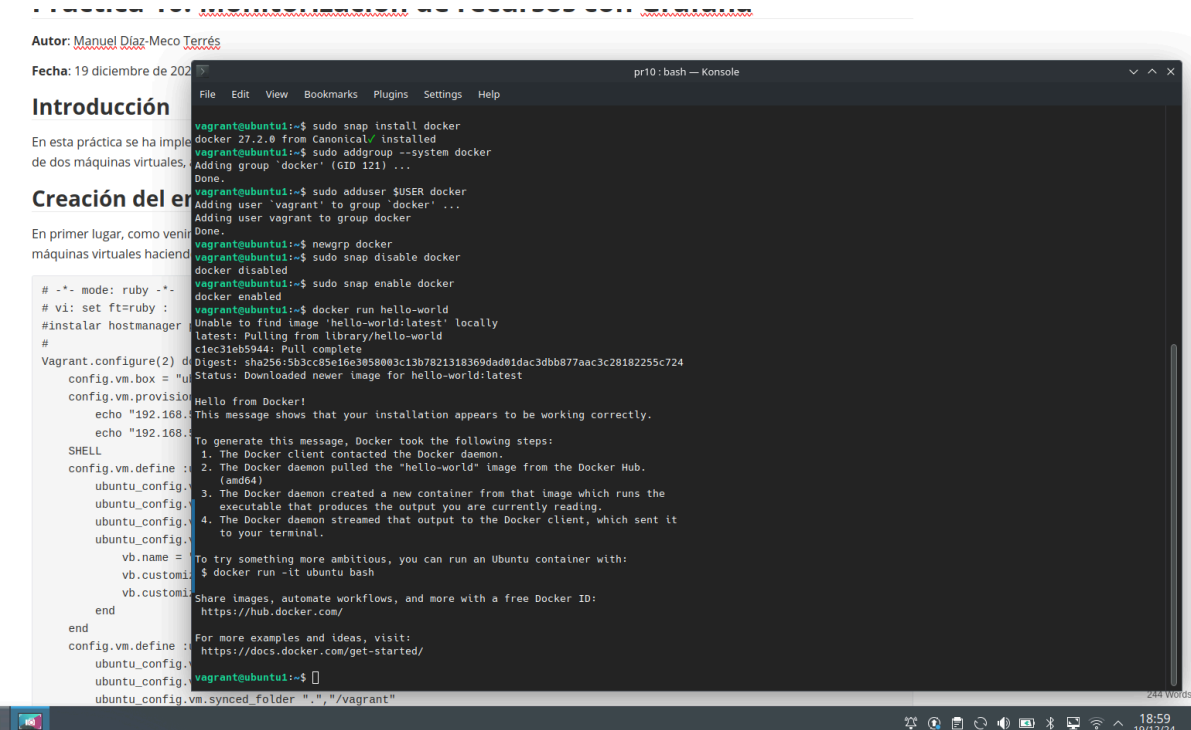
Creación y configuración del entorno de máquinas virtuales

En primer lugar, como venimos haciendo en prácticamente todas las prácticas anteriores (valga la redundancia), creamos y levantamos nuestras máquinas virtuales haciendo uso de *vagrant* y de un *Vagrantfile*.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
#instalar hostmanager plugin
#
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/jammy64"
  config.vm.provision "shell", inline: <<-SHELL
    echo "192.168.56.11 ubuntu1" >> /etc/hosts
    echo "192.168.56.12 ubuntu2" >> /etc/hosts
  SHELL
  config.vm.define :ubuntu1 do |ubuntu_config|
    ubuntu_config.vm.hostname = "ubuntu1.vm"
    ubuntu_config.vm.network "private_network" , ip:"192.168.56.11"
    ubuntu_config.vm.synced_folder ".", "/vagrant"
    ubuntu_config.vm.provider :virtualbox do |vb|
      vb.name = "ubuntu1"
      vb.customize ["modifyvm", :id, "--memory", "2048"]
      vb.customize ["modifyvm", :id, "--cpus", "2"]
    end
  end
  config.vm.define :ubuntu2 do |ubuntu_config|
    ubuntu_config.vm.hostname = "ubuntu2.vm"
    ubuntu_config.vm.network "private_network" , ip:"192.168.56.12"
    ubuntu_config.vm.synced_folder ".", "/vagrant"
    ubuntu_config.vm.provider :virtualbox do |vb|
      vb.name = "ubuntu2"
      vb.customize ["modifyvm", :id, "--memory", "2048"]
      vb.customize ["modifyvm", :id, "--cpus", "2"]
    end
  end
end
```

Y levantamos haciendo uso del comando `vagrant up`

Una vez ya creadas instalamos, en la máquina 1, *docker* y creamos el archivo `docker-compose.yml` como se indica en el gui3n.



Una vez hemos probado que funciona docker y que está configurado para usarlo sin privilegios de superusuario. El `docker-compose.yml` es el siguiente:

```
version: '3.6'

services:
  influxdb:
    image: influxdb:2.5.1-alpine
    container_name: influxdb
    restart: unless-stopped
    ports:
      - '8086:8086'
    volumes:
      - influxdb_data:/var/lib/influxdb2
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=supersecretpassword
      - DOCKER_INFLUXDB_INIT_ORG=my-org
      - DOCKER_INFLUXDB_INIT_BUCKET=my-bucket
      - DOCKER_INFLUXDB_INIT_RETENTION=1w

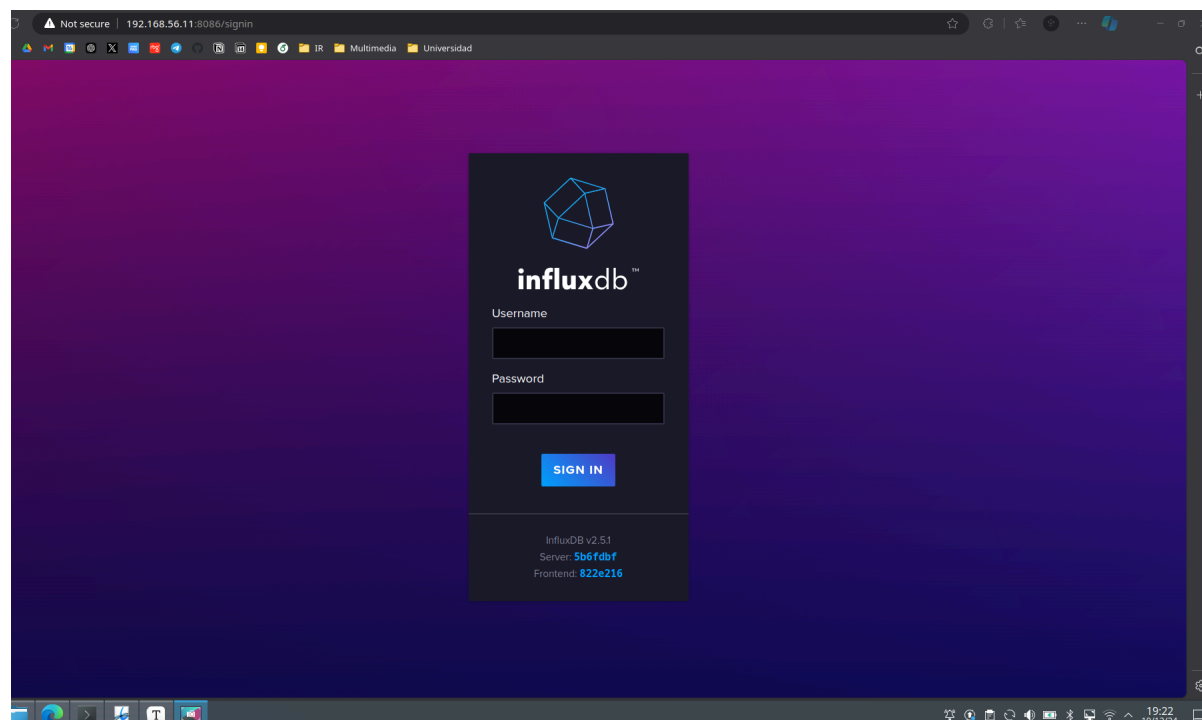
  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    restart: unless-stopped
    depends_on:
      - influxdb
    ports:
      - '3000:3000'
```

```
volumes:
  - grafana_data:/var/lib/grafana
environment:
  - GF_SECURITY_ADMIN_USER=admin
  - GF_SECURITY_ADMIN_PASSWORD=admin1234
  - GF_INSTALL_PLUGINS=
```

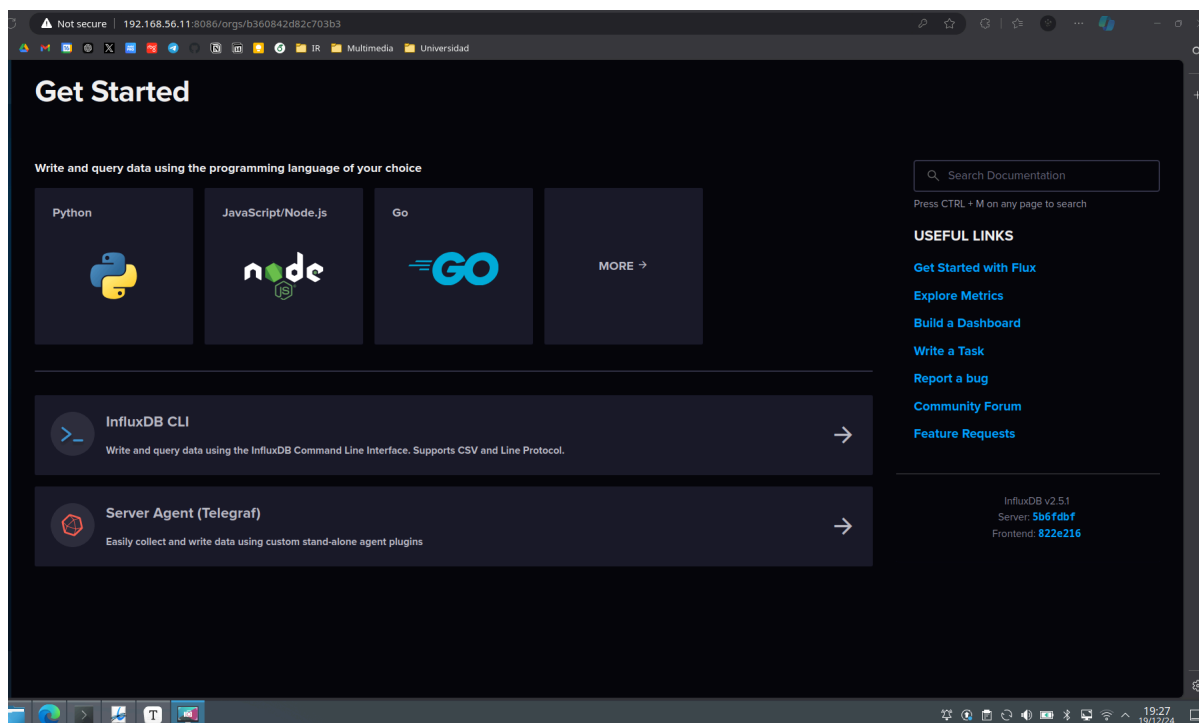
```
volumes:
  influxdb_data:
  grafana_data:
```

```
vagrant@ubuntu1:~$ mkdir pr10
vagrant@ubuntu1:~$ cd pr10/
vagrant@ubuntu1:~/pr10$ nano docker-compose.yml
vagrant@ubuntu1:~/pr10$ docker compose up -d
[+] Running 21/21
  ✓ influxdb 9 layers [#####] 0B/0B Pulled 28.5s
    ✓ f56be85fc22e Pull complete 7.4s
    ✓ 347b672f7645 Pull complete 8.8s
    ✓ c1afb36336fe Pull complete 16.8s
    ✓ 03b2ee057b40 Pull complete 17.4s
    ✓ 119fb4b43c8 Pull complete 21.8s
    ✓ 9ea7d2556fbf Pull complete 22.5s
    ✓ 70cadf8cf66c Pull complete 18.2s
    ✓ fab0fd66e726 Pull complete 19.8s
    ✓ 3e7be7ac61a2 Pull complete 20.7s
  ✓ grafana 10 layers [#####] 0B/0B Pulled 34.1s
    ✓ da9db072f522 Pull complete 1.0s
    ✓ ebc9c122b722 Pull complete 0.5s
    ✓ b2ddcf85da57 Pull complete 1.8s
    ✓ 00a90dd48f56 Pull complete 1.9s
    ✓ d7013a437817 Pull complete 1.9s
    ✓ 1739eae9cb10c Pull complete 2.3s
    ✓ 8964b0551c55 Pull complete 15.2s
    ✓ e0862465767a Pull complete 17.2s
    ✓ 989e41ffb5ca Pull complete 3.2s
    ✓ 9b4925e32b92 Pull complete 3.9s
[+] Running 5/5
  ✓ Network "pr10_default" Created 0.4s
  ✓ Volume "pr10_grafana_data" Created 0.0s
  ✓ Volume "pr10_influxdb_data" Created 0.0s
  ✓ Container influxdb Started 0.4s
  ✓ Container grafana Started 0.1s
vagrant@ubuntu1:~/pr10$
```

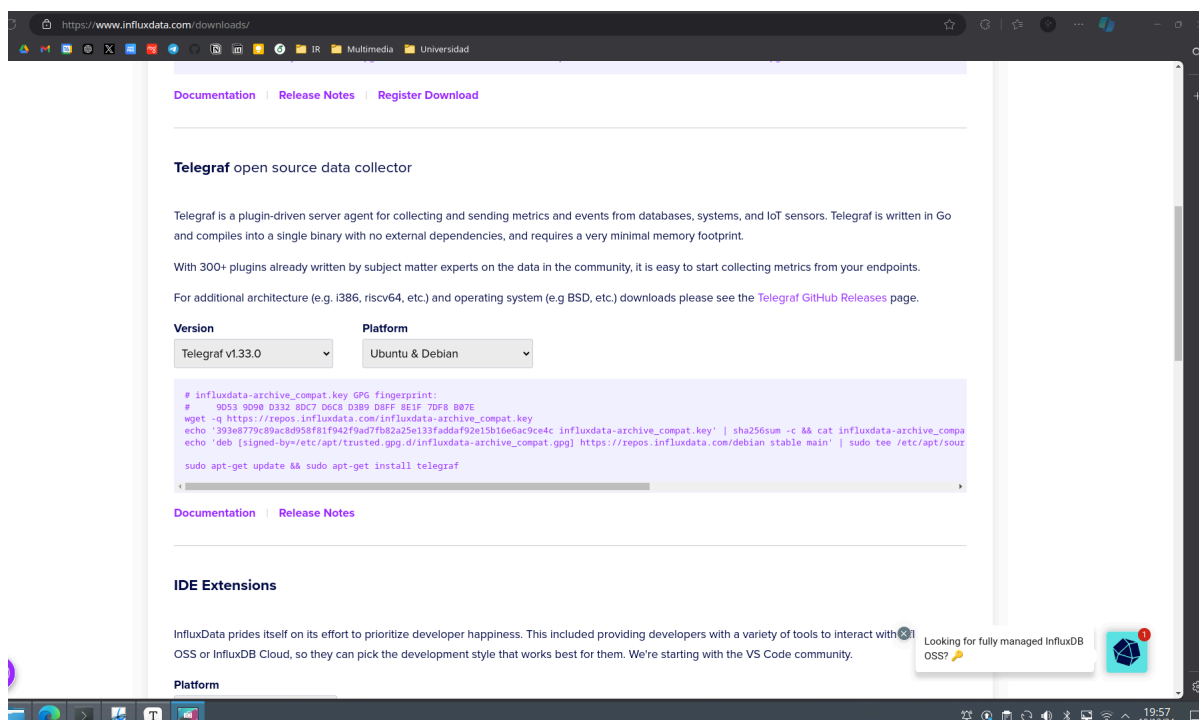
Accediendo a <http://192.168.56.11:8086/> encontramos la página de registro de *InfluxDB*:



Accedemos con lo que hemos puesto en el dockerfile:



Ahora, en la sección de *Telegraf* creamos un token con todos los permisos que nos permitirá conectar ambos servicios. Ahora instalamos *Telegraf* en la máquina virtual siguiendo los comandos que se indican en su página web:



```
pr10: bash — Konsole
File Edit View Bookmarks Plugins Settings Help
vagrant@ubuntu1:~$ wget -q https://repos.influxdata.com/influxdata-archive_compat.key
echo '395e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b10e6ac9ce4c influxdata-archive_compat.key' | sha256sum -c && cat influxdata-archive_compat.key |
gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg > /dev/null
echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.com/debian stable main' | sudo tee /etc/apt/sources.list
./influxdata.list
sudo apt-get update && sudo apt-get install telegraf
influxdata-archive_compat.key: OK
deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.com/debian stable main
Get:1 https://repos.influxdata.com/debian stable InRelease [6907 B]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:6 https://repos.influxdata.com/debian stable/main amd64 Packages [14.1 kB]
Fetched 21.0 kB in 1s (14.5 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  telegraf
0 upgraded, 1 newly installed, 0 to remove and 37 not upgraded.
Need to get 71.4 MB of archives.
After this operation, 269 MB of additional disk space will be used.
Get:1 https://repos.influxdata.com/debian stable/main amd64 telegraf amd64 1.33.0-1 [71.4 MB]
Fetched 71.4 MB in 7s (11.5 MB/s)
Selecting previously unselected package telegraf.
(Reading database ... 64883 files and directories currently installed.)
Preparing to unpack .../telegraf_1.33.0-1_amd64.deb ...
Unpacking telegraf (1.33.0-1) ...
Setting up telegraf (1.33.0-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/telegraf.service → /lib/systemd/system/telegraf.service.
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

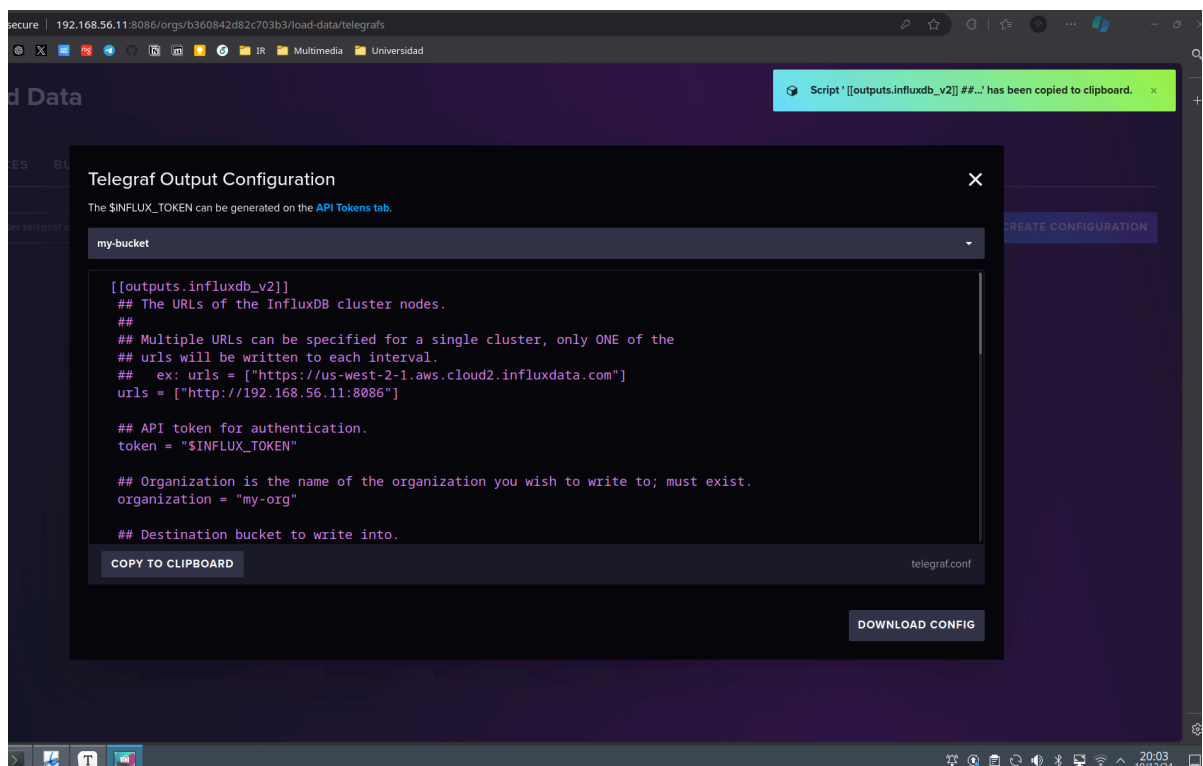
No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.
```

430 Words
20:00 19/12/24

Para enlazar *Telegraf* y *InfluxDB* tenemos que volver a la página de *InfluxDB*, acceder a la parte de **InfluxDB Output Plugin**



Ahora, copiamos lo que se nos da en el archivo `/etc/telegraf/telegraf.conf`:

```
#####
# CPD PRACTICA 10 #
#####

[[outputs.influxdb_v2]]
  ## The URLs of the InfluxDB cluster nodes.
  ##
  ## Multiple URLs can be specified for a single cluster, only ONE of the
  ## urls will be written to each interval.
  ## ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
  urls = ["http://192.168.56.11:8086"]

  ## API token for authentication.
  token = "$INFLUX_TOKEN"

  ## Organization is the name of the organization you wish to write to; must exist.
  organization = "my-org"

  ## Destination bucket to write into.
```

```

## urls will be written to each interval.
##   ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
urls = ["http://192.168.56.11:8086"]

## API token for authentication.
token = "*****QCTmnh4kw-3jUORiCTr4FCChe863vX1jW2EL40VtD1GBzrRzL1YIF-
AODXQ*****"

## Organization is the name of the organization you wish to write to; must
exist.
organization = "my-org"

## Destination bucket to write into.
bucket = "my-bucket"

## The value of this tag will be used to determine the bucket.  If this
## tag is not set the 'bucket' option is used as the default.
# bucket_tag = ""

## If true, the bucket tag will not be added to the metric.
# exclude_bucket_tag = false

## Timeout for HTTP messages.
# timeout = "5s"

## Additional HTTP headers
# http_headers = {"X-Special-Header" = "Special-Value"}

## HTTP Proxy override, if unset values the standard proxy environment
## variables are consulted to determine which proxy, if any, should be used.
# http_proxy = "http://corporate.proxy:3128"

## HTTP User-Agent
# user_agent = "telegraf"

## Content-Encoding for write request body, can be set to "gzip" to
## compress body or "identity" to apply no encoding.
# content_encoding = "gzip"

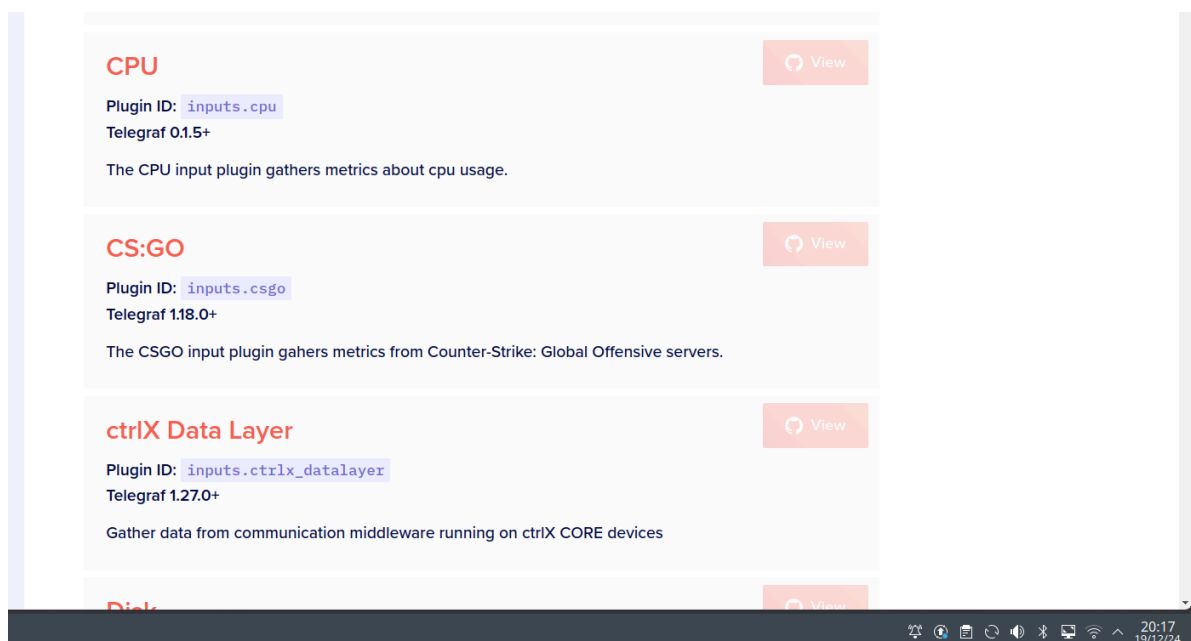
## Enable or disable uint support for writing uints influxdb 2.0.
# influx_uint_support = false

## Optional TLS Config for use on HTTP connections.
# tls_ca = "/etc/telegraf/ca.pem"
# tls_cert = "/etc/telegraf/cert.pem"
# tls_key = "/etc/telegraf/key.pem"
## Use TLS but skip chain & host verification
# insecure_skip_verify = false

#####

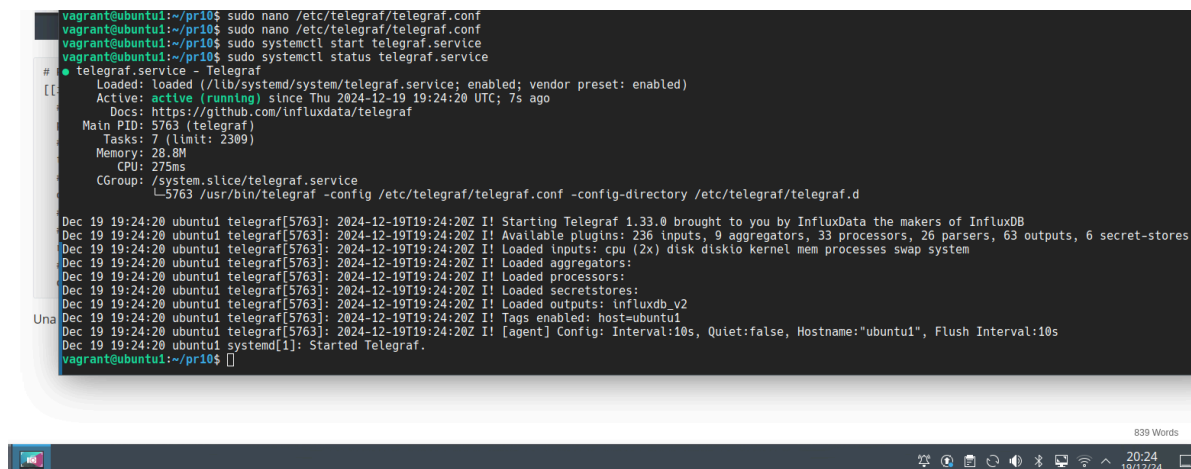
```

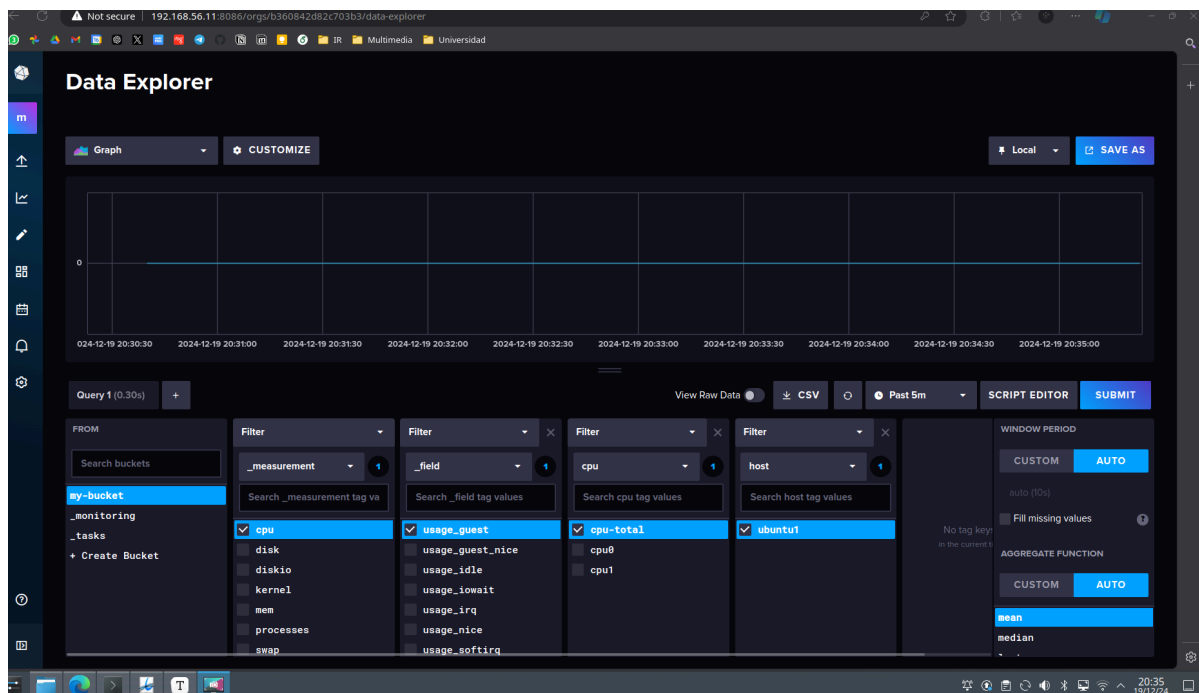
Notar que he quitado algunos de los caracteres del token, lo más seguro sería utilizar una variable. Ahora, tenemos que añadir la configuración necesaria para poder hacer una monitorización de los datos que se quieran. En nuestro caso, vamos a utilizar los datos sobre nuestra CPU.



```
# Read metrics about cpu usage
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states
  ## NOTE: The resulting 'time_active' field INCLUDES 'iowait'!
  report_active = false
  ## If true and the info is available then add core_id and physical_id tags
  core_tags = false
```

Una vez añadido esto último al archivo de configuración de *Telegraf* debemos iniciar el servicio para poder observar los resultados en *InfluxDB*



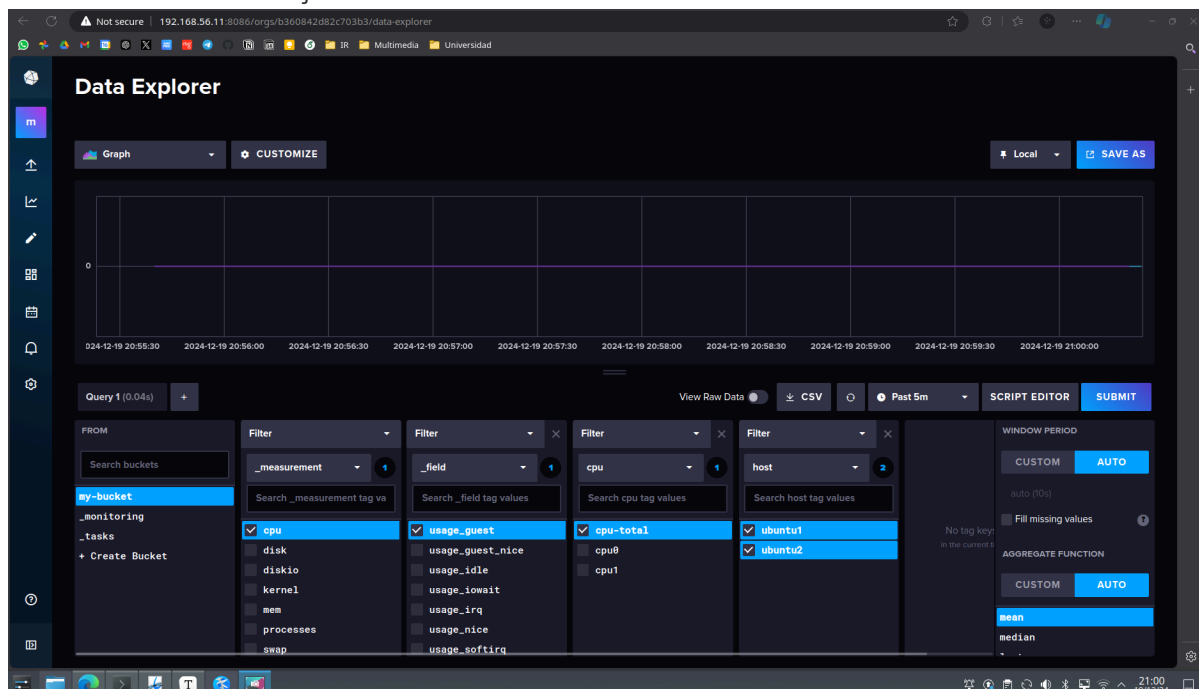


Ahora, debemos repetir el proceso de la instalación y configuración de *Telegraf* en la segunda máquina virtual. Obtemos entonces:

```
vagrant@ubuntu2:~$ sudo systemctl start telegraf.service
vagrant@ubuntu2:~$ sudo systemctl status telegraf.service
● telegraf.service - Telegraf
   Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2024-12-19 19:52:18 UTC; 5s ago
     Docs: https://github.com/influxdata/telegraf
    Main PID: 3887 (telegraf)
      Tasks: 0 (limit: 2309)
   Memory: 27.4M
      CPU: 51ms
   CGroup: /system.slice/telegraf.service
           └─3887 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.d

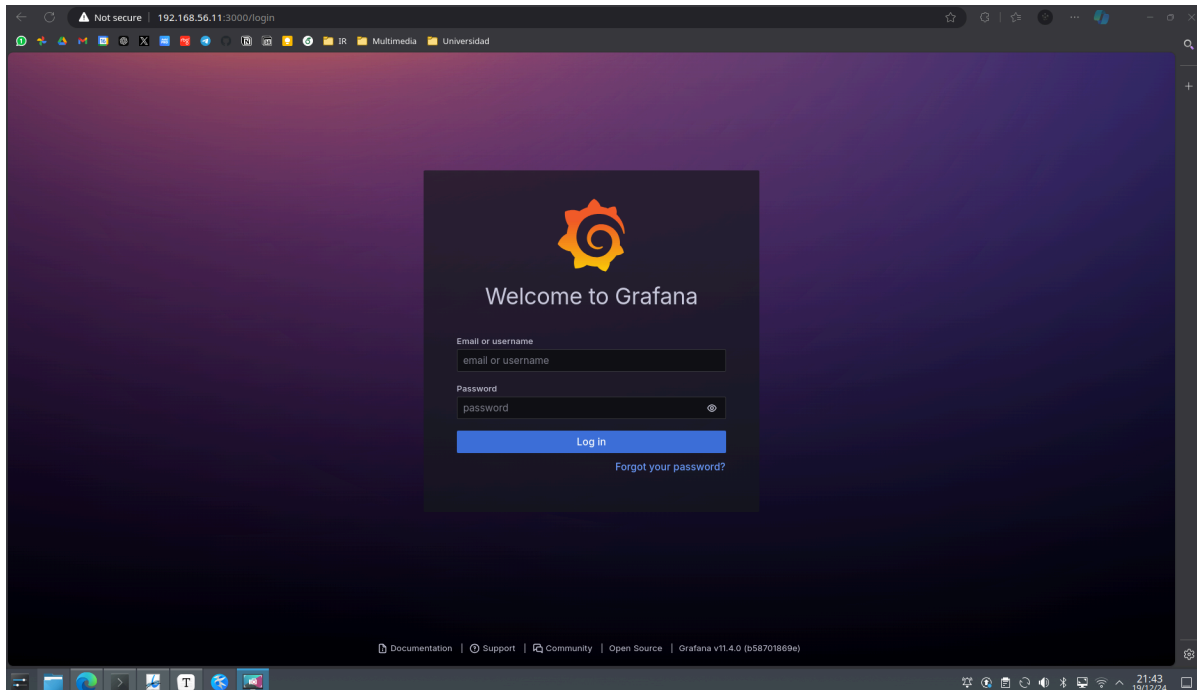
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Starting Telegraf 1.33.0 brought to you by InfluxData the makers of InfluxDB
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Available plugins: 236 inputs, 9 aggregators, 33 processors, 26 parsers, 63 outputs, 6 secret-stores
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Loaded inputs: cpu (2x) disk diskio kernel mem processes swap system
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Loaded aggregators:
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Loaded processors:
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Loaded secretstores:
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Loaded outputs: influxdb_v2
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! Tags enabled: host=ubuntu2
Dec 19 19:52:18 ubuntu2 systemd[1]: Started Telegraf.
Dec 19 19:52:18 ubuntu2 telegraf[3887]: 2024-12-19T19:52:18Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"ubuntu2", Flush Interval:10s
vagrant@ubuntu2:~$
```

También esto se ve reflejado en nuestra base de datos:

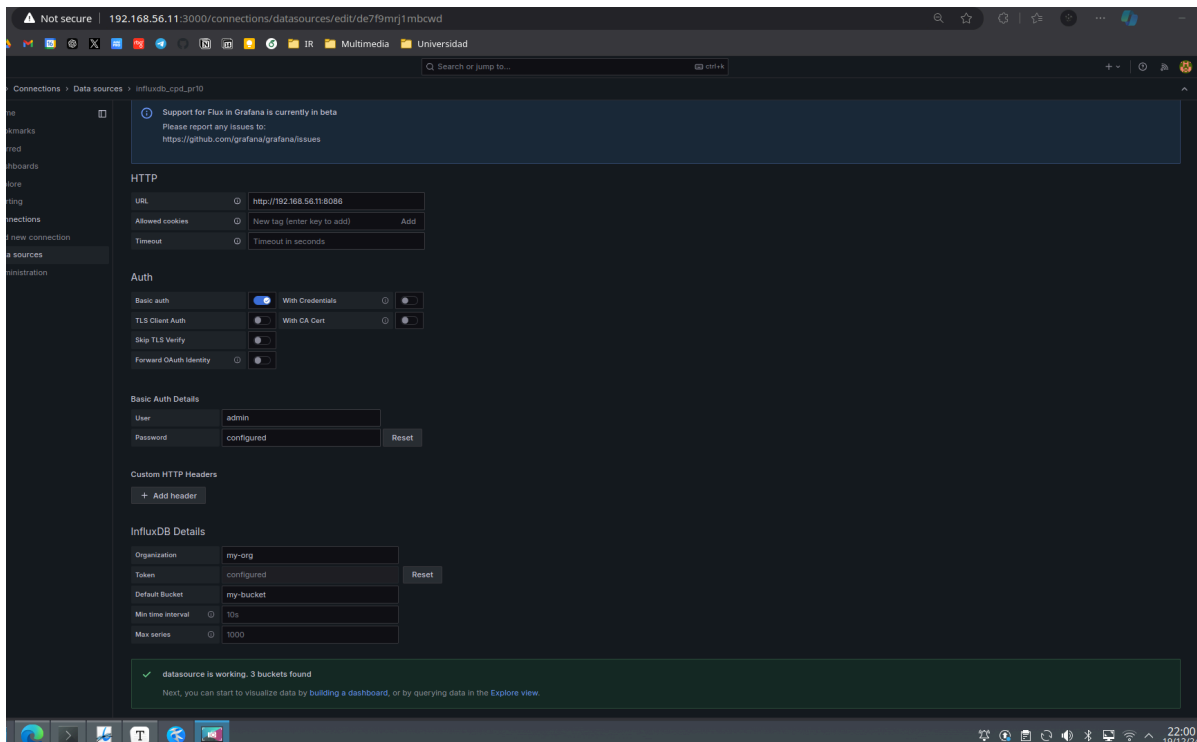


Grafana

Pasamos ahora con la última parte, la de configurar *Grafana* para que muestre los datos recibidos de ambas máquinas. Para acceder a *Grafana* vamos a la dirección dada en el `docker-compose.yml`: <http://192.168.56.11:3000>



Accedemos, al igual que antes, con los parámetros del dockerfile. Una vez que ya estamos dentro hemos de acceder a *Connections* --> *Add New Conenction*, seleccionamos *InfluxDB* y lo configuramos como se muestra a continuación:



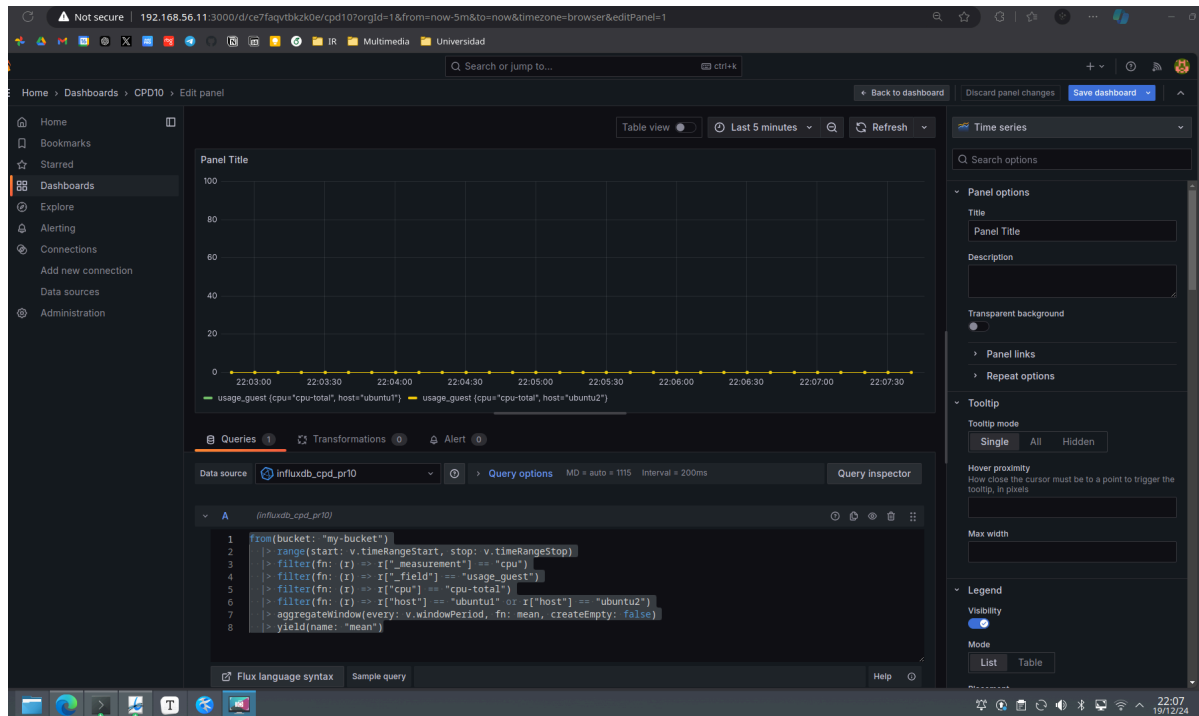
Ahora pinchamos donde aparece '*building a dashboard*' para poder crear la visualización de los datos. De primeras, no tendremos datos, es el momento de volver a la página de *InfluxDB* y copiar la *query* que se nos da para mostrar los datos de sendas máquinas virtuales ya configuradas, en nuestro caso:

```

from(bucket: "my-bucket")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "cpu")
  |> filter(fn: (r) => r["_field"] == "usage_guest")
  |> filter(fn: (r) => r["cpu"] == "cpu-total")
  |> filter(fn: (r) => r["host"] == "ubuntu1" or r["host"] == "ubuntu2")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")

```

Copiamos esto mismo en el apartado de *query* de nuestro dashboard de *Grafana* y guardamos los cambios:



Con esto conseguimos un panel básico para poder visualizar los datos a través del uso de Grafana, Telegraf e InfluxDB. Podríamos añadir más puglins de visualización, ya sabemos que Grafana es una potente herramienta en este aspecto.