



Manual de Influxdb

Pablo Cisterna

16 de enero 2025

Índice

1	Resumen.	3
1.1	¿Qué es influxdb?	3
1.2	¿Qué son las librerías clientes?	3
1.3	¿Qué es Telegraf?	3
1.4	Introducción a proyecto.	3
2	Instalaciones.	4
2.1	Influxdb.	4
2.2	Telegraf.	4
2.3	Cliente de python.	5
3	Configuración y uso.	6
3.1	Telegraf.	6
3.2	Cliente de python.	12
3.2.1	¿Cómo inició el cliente?	12
3.2.2	¿Cómo se sube la información?	12
3.2.3	¿Cómo se saca la información desde InfluxDB?	13
3.3	Influxdb.	15
3.3.1	Subida de mediciones.	15
3.3.2	Explorador de datos.	16
3.3.3	Notebooks.	19
3.3.4	Dashboard.	20
3.3.5	Precauciones.	21
4	Proyecto en influxdb.	22
4.1	Generación de datos.	22
4.2	Procesamiento de datos.	23
5	Referencias.	26

1. Resumen.

1.1. ¿Qué es influxdb?

Influxdb es una base de datos de series temporales open source creada por Errplane (luego Influx-Data), lanzado originalmente en 2013, esta base de datos es usada por grandes compañías como Tesla, NVidia y IBM.

Que la base de datos sea de serie temporal significa que, al guardar los datos (o medidas), estas también se les asignan una fecha (o timestamp), que en caso de influxdb tiene una precisión en los nanosegundos.

Influxdb se usa principalmente en monitoreo de operaciones, IOT (Internet of things), métricas de aplicaciones y analítica en tiempo real.

1.2. ¿Qué son las librerías clientes?

Las librerías cliente son una serie de funciones que permite la comunicación entre un programa Python, GO, Java, etc. y la base de datos influxdb.

Para poder usarla es necesario ya haber hecho la configuración inicial de influxdb ya que es necesario tener una organización, bucket y token API.

En el caso de este manual solo usaremos el cliente de Python.

1.3. ¿Qué es Telegraf?

Telegraf es un agente de servicio open source que permite la colección de métricas, logs, etc., las cuales son subidas a bases de datos como Influxdb, Kafka, MongoDB, etc.

1.4. Introducción a proyecto.

Estas tecnologías nos dan la capacidad de guardar, descargar y procesar medidas realizadas, ya sea en la vida real o simuladas.

Para entender el uso de influxdb y su cliente de python, realice un proyecto en el cual se simulaba un mall y las temperaturas fuera de este, con el cual ver si la temperatura afectaba a la estadía de las personas.

Al final de este manual se mostrará más de este proyecto.

2. Instalaciones.

2.1. Influxdb.

Para instalar influxdb en linux, copia los siguientes comandos en la terminal:

```
# influxdata-archive_compat.key GPG fingerprint:
# 9D53 9D90 D332 8DC7 D6C8 D3B9 D8FF 8E1F 7DF8 B07E
wget -q https://repos.influxdata.com/influxdata-archive_compat.key
echo '393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c
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.d/influxdata.list

sudo apt-get update && sudo apt-get install influxdb2
```

Con esto InfluxDB se habrá instalado en el sistema y arrancará siempre que se inicie la máquina. Para poder acceder a la base de datos ingresa lo siguiente a tu navegador:

<http://localhost:8086>

Al ingresar, se te pedirá crear un usuario que manejara el uso de la base de datos, una organización que guardará tus buckets y un primer bucket para guardar las mediciones. Teniendo esto listo ya podrás usar tu base de datos.

Para más información visita:

<https://docs.influxdata.com/influxdb/v2/install/?t=Linux#download-and-install-influxdb-v2>

2.2. Telegraf.

Para instalar Telegraf en linux, tienes que copiar los siguientes comandos en la terminal:

```
# influxdata-archive_compat.key GPG fingerprint:
# 9D53 9D90 D332 8DC7 D6C8 D3B9 D8FF 8E1F 7DF8 B07E
wget -q https://repos.influxdata.com/influxdata-archive_compat.key
echo '393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c
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.d/influxdata.list

sudo apt-get update && sudo apt-get install telegraf
```

Una vez teniendo el paquete instalado se necesita una configuración para poder ejecutarlo, esto se puede hacer usando InfluxDB.

Para más información visita:

<https://docs.influxdata.com/telegraf/v1/install/>

2.3. Cliente de python.

InfluxDB también se puede usar en Python, para lograr esto, primero hay que crear un entorno virtual de python, para ello primero instalaremos venv para crear el entorno:

```
sudo apt-get install python3-pip  
sudo apt install python3-venv
```

Una vez instalado tenemos que crear el entorno:

```
python3 -m venv <Path donde se quiere guardar el entorno>
```

Para iniciar el entorno tenemos que ejecutar el siguiente comando en la terminal:

```
source path/to/env/bin/activate
```

Una vez iniciado el entorno tenemos que instalar el cliente python:

```
pip install influxdb-client
```

Con esto ya tendrías el cliente de python instalado.

En este manual solo se verá el cliente de python, pero también existen otros clientes para varios lenguajes de programación como Arduino, GO, Java, C#, etc. Estas y otras descargas de InfluxData se pueden encontrar en el siguiente link:

<https://www.influxdata.com/influxdata-downloads/>

3. Configuración y uso.

3.1. Telegraf.

Para poder utilizar telegraf es necesario darle una configuración, la cual consiste de los plugins que se quieran utilizar y los periodos de subida de medidas, esto se puede realizar fácilmente usando InfluxDB.

Para ello tenemos que ingresar a la base de datos ya instalada, ingresa el siguiente link a tu navegador:

<http://localhost:8086>

Una vez dentro se te pedirá el usuario y contraseña para poder ingresar a la base de datos, luego de iniciar la sesión te saldrá una pestaña que se vera así:

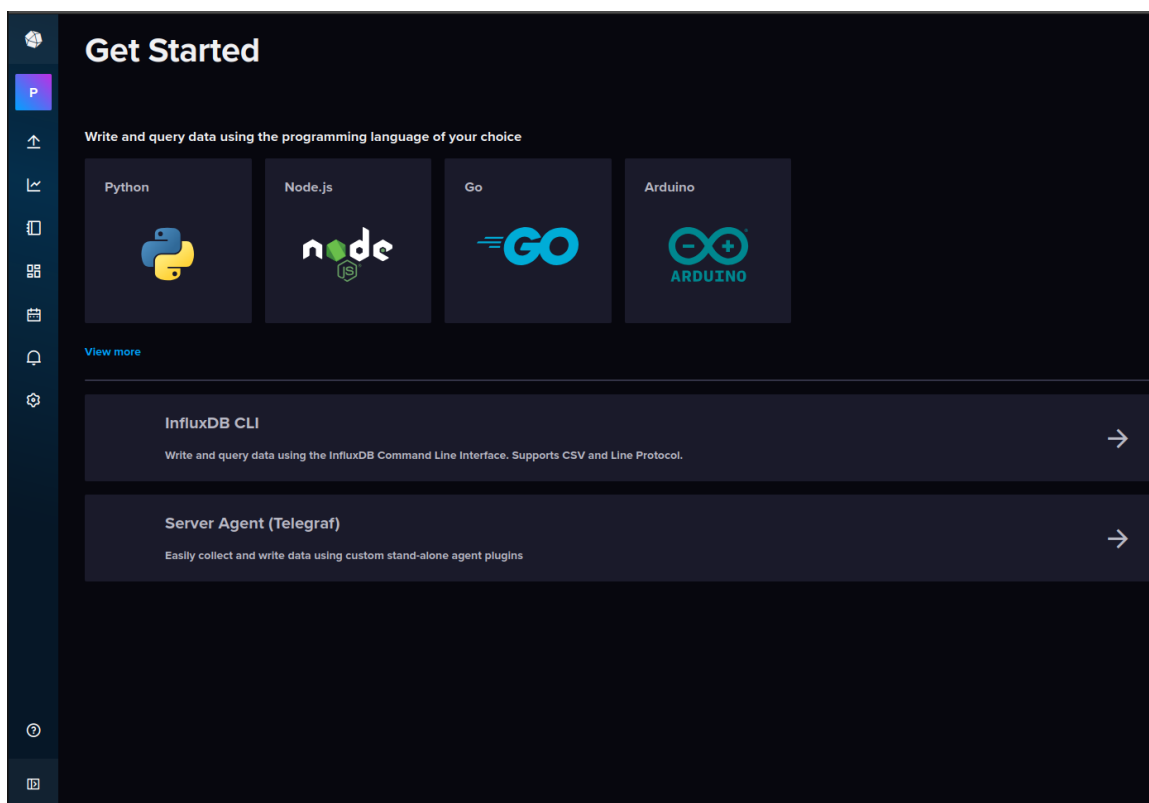


Figura 1: Menú al abrir Influxdb.

A la izquierda aparecerán unas pestañas:

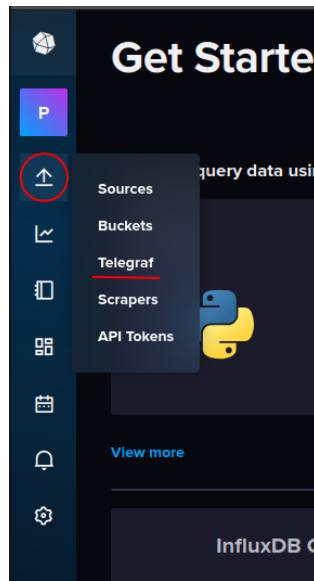


Figura 2: Menú al seleccionar la pestaña de subidas.

Al ingresar a la pestaña Marcada en la figura anterior saldrá un boton llamado **CREATE CONFIGURATION**, hazle click y se abrirá esta ventana:

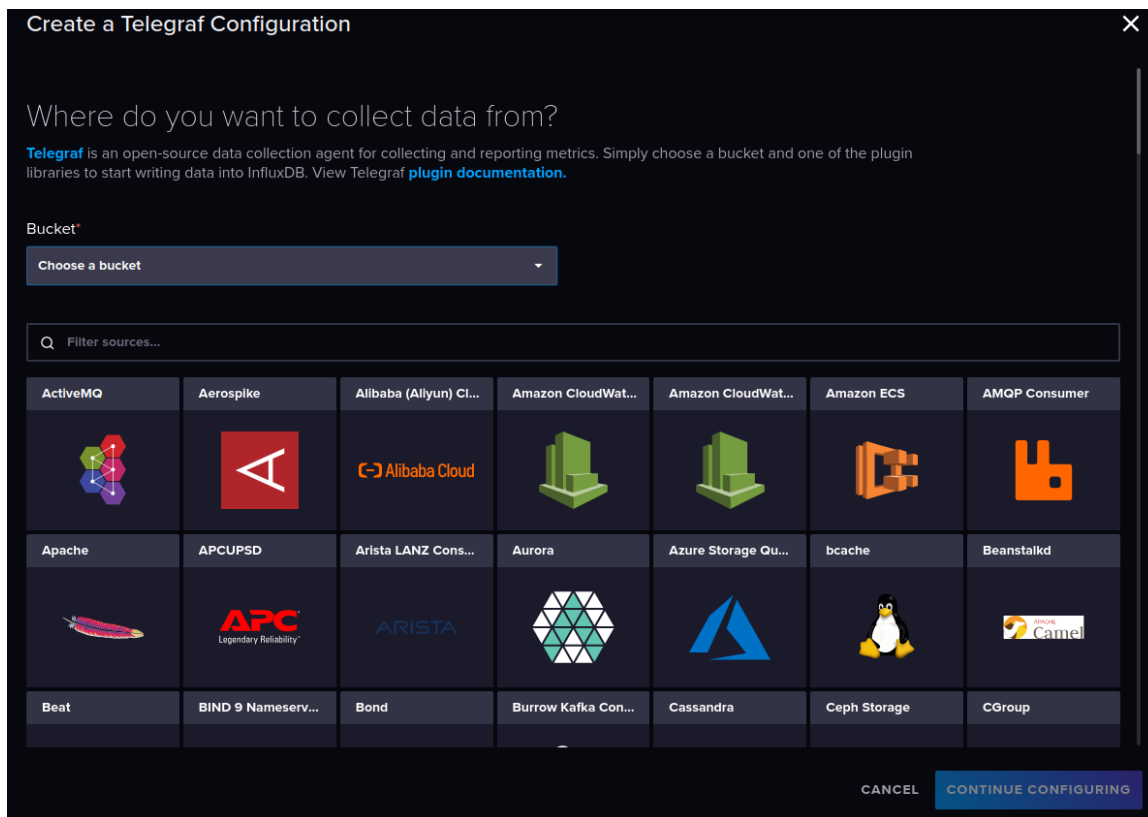
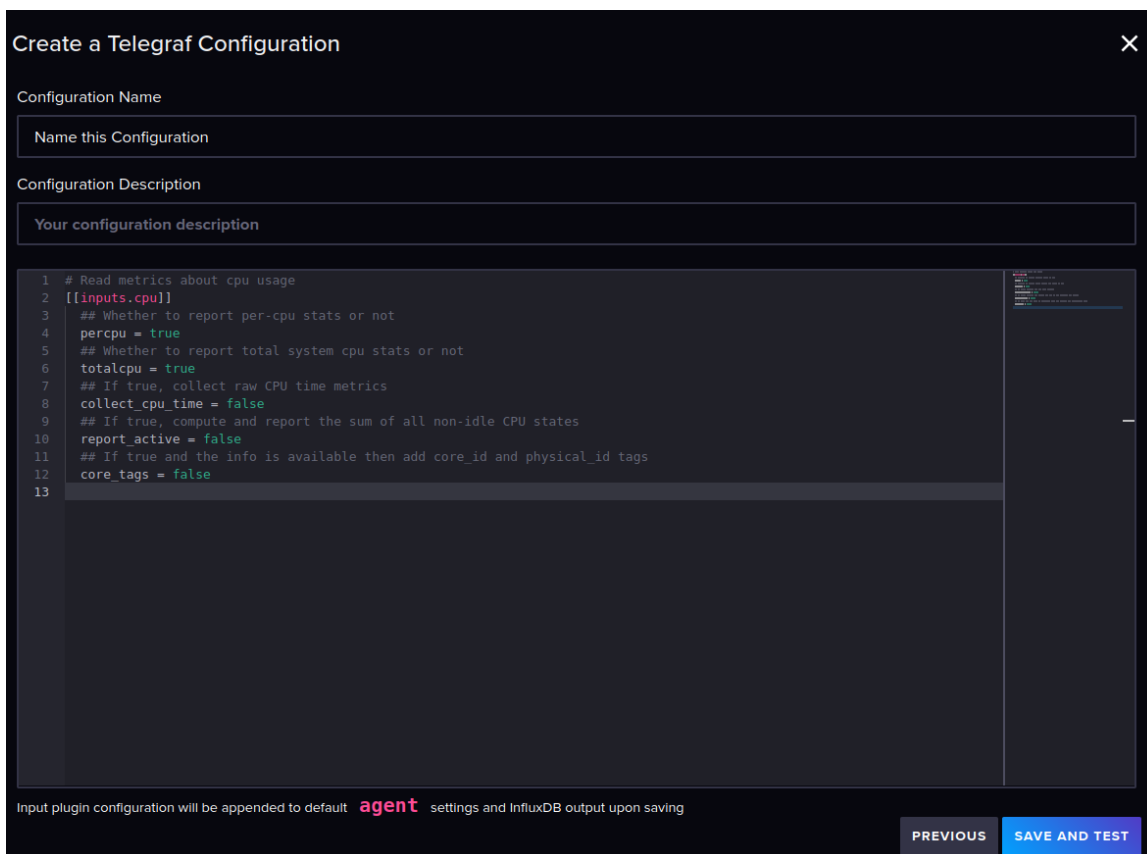


Figura 3: Menú de selección de bucket y plugin.

Aquí podrás seleccionar el bucket donde quieras guardar tus medidas y un primer plugin, en este caso usaremos **CPU**, que nos subirá a nuestra base de datos información sobre el procesador de la máquina, cuando esté listo, selecciona el botón **CONTINUE CONFIGURATION**, lo cual abrirá la siguiente ventana:



Create a Telegraf Configuration [X]

Configuration Name

Name this Configuration

Configuration Description

Your configuration description

```

1 # Read metrics about cpu usage
2 [[inputs.cpu]]
3 ## Whether to report per-cpu stats or not
4 percpu = true
5 ## Whether to report total system cpu stats or not
6 totalcpu = true
7 ## If true, collect raw CPU time metrics
8 collect_cpu_time = false
9 ## If true, compute and report the sum of all non-idle CPU states
10 report_active = false
11 ## If true and the info is available then add core_id and physical_id tags
12 core_tags = false
13

```

Input plugin configuration will be appended to default **agent** settings and InfluxDB output upon saving

PREVIOUS SAVE AND TEST

Figura 4: Nombre, descripción y configuración de instancia de telegraf.

Aquí podrás darle un nombre y descripción a la configuración, además de poder agregar más plugins. Para agregar más plugins, tienes que ingresar a la siguiente página:

<https://docs.influxdata.com/telegraf/v1/plugins/>

Una vez dentro se verá de la siguiente forma:

Plugin directory

Telegraf is a plugin-driven agent that collects, processes, aggregates, and writes metrics. It supports four categories of plugins: input, output, aggregator, and processor. In addition to the included plugins, you can run *external plugins* that integrate with the Telegraf Execd processor plugin.

Plugin type	Plugin category	
<input type="checkbox"/> Input(253)	<input type="checkbox"/> Applications(32)	<input type="checkbox"/> Logging(15)
<input type="checkbox"/> Output(62)	<input type="checkbox"/> Build & Deploy(10)	<input type="checkbox"/> Messaging(26)
<input type="checkbox"/> Aggregator(9)	<input type="checkbox"/> Cloud(32)	<input type="checkbox"/> Networking(50)
<input type="checkbox"/> Processor(33)	<input type="checkbox"/> Containers(11)	<input type="checkbox"/> Servers(28)
<input type="checkbox"/> External(20)	<input type="checkbox"/> Data Stores(40)	<input type="checkbox"/> Systems(65)
	<input type="checkbox"/> IoT(15)	<input type="checkbox"/> Web(29)

Operating system	Status
<input type="checkbox"/> Linux(328)	<input type="checkbox"/> New(0)
<input type="checkbox"/> macOS(311)	<input type="checkbox"/> Deprecated(3)
<input type="checkbox"/> Windows(313)	

Figura 5: Selección de características del plugin.

Para buscar plugins, le daremos click a los casilleros **Input** y **Linux**, lo cual filtra la búsqueda.

Para este ejemplo usaremos el plugin **Mem**, que nos dará los valores de uso de la memoria primaria, como el Espacio usado y el espacio libre:


The mdstat plugin gathers statistics about any Linux MD RAID arrays configured on the host by reading /proc/mdstat.

Mem

Plugin ID: `inputs.mem`

Telegraf 0.1.5+

The Mem input plugin collects system memory metrics. For a more complete explanation of the difference between used and actual_used RAM, see [Linux ate my ram](#).

 View

Memcached

Plugin ID: `inputs.memcached`


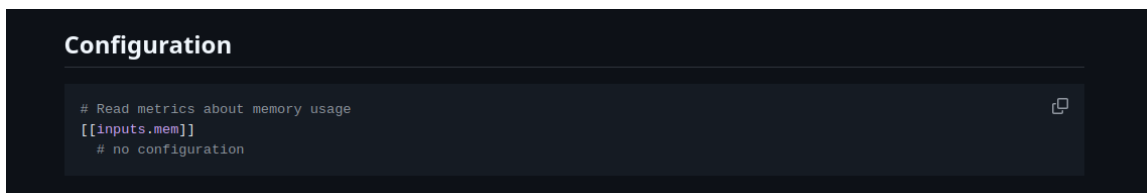
 View

Figura 6: Búsqueda de plugins.

Le daremos click al botón **View**, lo cual nos abrirá un repositorio de github, vamos al apartado configuración y copiamos el bloque de código:

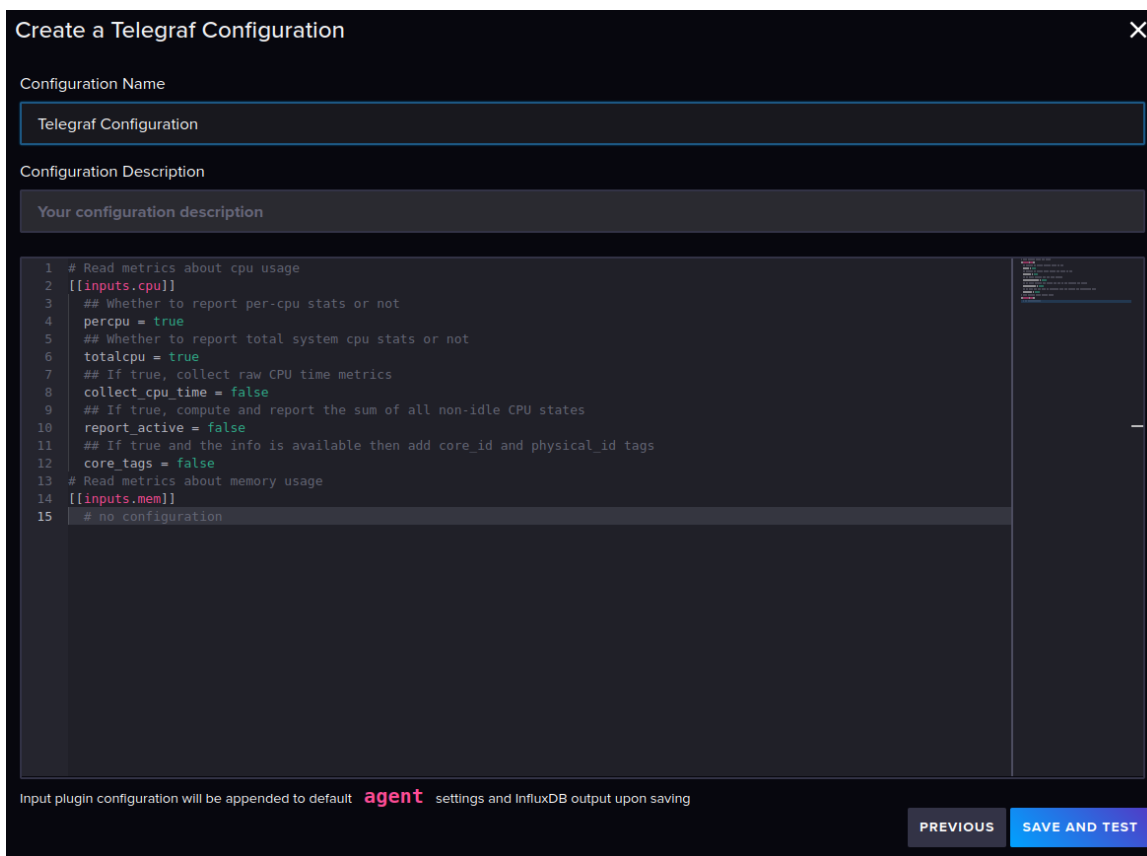


Configuration

```
# Read metrics about memory usage
[[inputs.mem]]
# no configuration
```

Figura 7: Bloque de código en github.

Y lo pegamos al bloque de configuración que teníamos en la página de la base de datos:



Create a Telegraf Configuration

Configuration Name
Telegraf Configuration

Configuration Description
Your configuration description

```
1 # Read metrics about cpu usage
2 [[inputs.cpu]]
3 ## Whether to report per-cpu stats or not
4 percpu = true
5 ## Whether to report total system cpu stats or not
6 totalcpu = true
7 ## If true, collect raw CPU time metrics
8 collect_cpu_time = false
9 ## If true, compute and report the sum of all non-idle CPU states
10 report_active = false
11 ## If true and the info is available then add core_id and physical_id tags
12 core_tags = false
13 # Read metrics about memory usage
14 [[inputs.mem]]
15 # no configuration
```

Input plugin configuration will be appended to default **agent** settings and InfluxDB output upon saving

PREVIOUS **SAVE AND TEST**

Figura 8: Nombre, descripción y configuración de instancia de telegraf con plugin ingresado.

Le damos click al botón **SAVE AND TEST** y nos saldrá una nueva pestaña:

Telegraf Setup Instructions ✕

1. Install the Latest Telegraf

You can install the latest Telegraf by visiting the [InfluxData Downloads](#) page. If you already have Telegraf installed on your system, make sure it's up to date. You will need version 1.9.2 or higher.

2. Configure your API Token

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=<INFLUX_TOKEN>
```

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://localhost:8086/api/v2/telegrafs/0e2a3775eda18000
```

COPY TO CLIPBOARD CLI

Figura 9: Instrucciones de set up en la UI de influxdb.

Copiamos los dos comandos que salen en la ventana y los pegamos en la terminal.

Con esto ya tendríamos configurado y corriendo Telegraf, para cambiar la configuración, regresa a la pestaña telegraf en InfluxDB y haz click en el nombre de la configuración, lo cual abrirá un bloque de código donde podrás cambiar cosas como el intervalo de subida o agregar más plugins.

Para más información visita:

<https://docs.influxdata.com/telegraf/v1/>

3.2. Cliente de python.

3.2.1. ¿Cómo inició el cliente?

Una vez ingresado al entorno virtual de python es necesario importar las librerías al programa de python, lo cual se hace de la siguiente forma:

```
import influxdb_client as idb
from influxdb_client.client.write_api import SYNCHRONOUS
```

Una vez hecho esto, deberás ingresar los datos donde guardarás y/o sacarás la información:

```
token = "token"
org = "organizacion"
url = "url"
bucket = "bucket"
```

Donde:

- token**: Token de InfluxDB.
- org**: Organización donde se guardan los datos.
- url**: URL de InfluxDB (En este caso <http://localhost:8086>).
- bucket**: "Bucket donde se guarda la información dentro de la organización.

Luego hay que crear un objeto cliente:

```
client_idb = idb.InfluxDBClient(
    url = url,
    token = token,
    org = org
)
```

Con esto ya estaría hecho el cliente de InfluxDB.

3.2.2. ¿Cómo se sube la información?

Para poder enviar información a la base de datos necesitas haber creado el objeto cliente para poder crear una instancia de escritura, lo cual se hace con la siguiente línea de código:

```
write_api = client_idb.write_api(write_options=SYNCHRONOUS)
```

Y luego necesitas crear un "punto", esto es lo que guardará la información que se guardará en la base de datos con los siguientes valores:

- Point("Nombre de la Categoría")**: Guarda en una categoría.
- tag("Nombre de la columna", "Valor de la columna")**: Guarda valores de metadata de la medida realizada, acepta valores numéricos y strings.
- field("Nombre del valor", "Valor")**: Guarda los valores de la medida realizada, field solo acepta valores numéricos.
- time(datetime)**: Sirve para cambiar la timestamp con el cual se subirá el punto a influxdb, tiene que tener el siguiente formato: AAAA-MM-DDTHH:MM:SSZ. Ejemplo: 2025-01-01T12:00:00Z

Un ejemplo de uso sería: Medición de 15°C de temperatura para la ciudad de Valdivia:

```
p = idb.Point("Medida").tag("Ciudad", "Valdivia").field("Temperatura", 15)
```

con esto ya estaría hecho el punto, ahora hay que subirlo a la base de datos con la siguiente línea de código:

```
write_api.write(bucket=bucket, org=org, record=p)
```

Donde:

-bucket: Es el bucket donde se quiere guardar el punto.

-org: Es la organización donde se quiere guardar el punto.

-record: Es donde va el punto que se quiere subir, esto puede ser un único punto o una lista de punto, lo cual provocará que todos los puntos tengan la misma timestamp.

Al momento de subir los puntos se le asignará una timestamp.

Con esto ya tendrías subida tu información a la base de datos en InfluxDB.

3.2.3. ¿Cómo se saca la información desde InfluxDB?

Para poder sacar o “consultar” la información de la base de datos en InfluxDB hay que crear una instancia de consulta, para ello necesitas haber hecho un objeto cliente y luego usar la siguiente línea de código:

```
query_api = client_idb.query_api()
```

Una vez hecho esto tienes que crear una consulta en Flux, con la cual existen varios comandos como:

-from(bucket: “Nombre del bucket”): Aquí se declara el nombre del bucket donde se quiere sacar la información.

range(start: “Desde donde se quiere empezar”, stop: “Hasta donde se quiere leer”): Aquí se declara desde dónde hasta cuando se quiere sacar la información:

-start: Puede tomar valores relativos como -1h (desde una hora antes de realizar la consulta), puede empezar desde cierta ID de puntos o desde una fecha en específico.

-stop: puede tomar los mismos tipos de valor que start mientras sean valores que vengan después cronológicamente, stop también se puede sacar del comando range, lo cual significa que mostrará todos los valores desde start hasta el momento de realizar la consulta.

filter(fn: (r) => r.”Tipo de columna” == “Valor en columna”): Filtra los valores que se recibirán en la consulta. Ejemplo:

filter(fn: (r) => r._measurement == “Cliente”): solo recibe valores que estén en la categoría “Clientes”

group(columns: [“Nombre de columnas”]): agrupa los valores según sus columnas. Ejemplo:
group(columns: [Ciudad]) agrupa los valores según los valores en la columna “Ciudad”

Un ejemplo de Flux en python sería:

```
query = 'from(bucket: "mybucket")\n      |> range(start: -15m)\n      |> filter(fn: (r) => r._measurement == "Temperatura")\n      |> group(columns: ["Ciudad"])
```

Nota: El comando from siempre va en la primera línea, al terminar debe ir ‘\’, al comenzar una línea siguiente tiene que comenzar con ‘|>’, para más información sobre consultas Flux visita: <https://docs.influxdata.com/flux/v0/>

Una vez hecha la consulta hay que enviarla a InfluxDB con el siguiente comando:

```
result = query_api.query(org=org, query=query)
```

Una vez recibido los valores se tendrán como una tabla, la cual es necesario recorrer de la siguiente forma:

```
for table in result:
    for record in table.records:
```

Existen varias funciones que extraen información de las medidas, como:

get_time(): Muestra el timestamp de de la medida que se encuentra.

get_value(): Muestra el valor guardado.

get_field(): Muestra el nombre del valor guardado.

values.get("Nombre"): devuelve el valor de la medida en la columna "Nombre".

get_measurement(): Devuelve el nombre de la categoría de la medida.

Un ejemplo de esto sería:

```
ciudad = [record.values.get("Ciudad"),record.get_field(),record.get_value()]
print(ciudad)
```

Lo cual nos daría:

["Valdivia",Temperatura,15]

Con esto ya tendríamos el uso básico del Cliente de python de Influxdb, para más Información busca: <https://docs.influxdata.com/influxdb/cloud/api-guide/client-libraries/python/>

3.3. Influxdb.

3.3.1. Subida de mediciones.

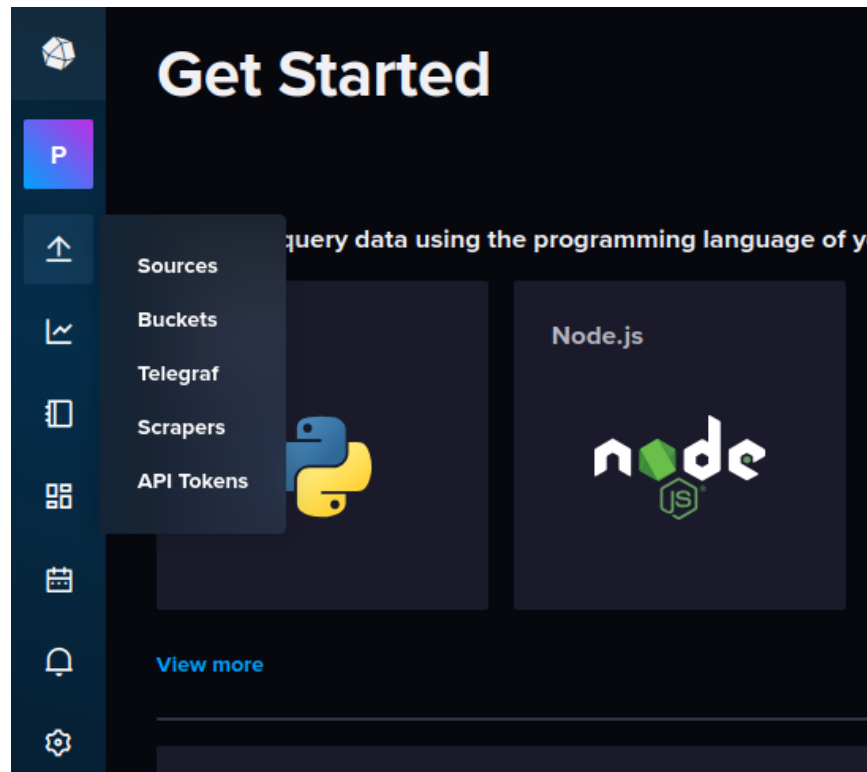


Figura 10: Pestaña de subida de mediciones.

En la esta pestaña tenemos las siguientes opciones:

Sources: Donde se va a sacar la información, incluye:

- Subir un archivo CSV.
- Usar alguna de los clientes disponible como Python, GO , C#, Java, etc.
- Usar alguno de los plugins de Telegraf.

Buckets: Es donde se puede guardar, crear y configurar los buckets donde se guardaran las medidas subidas a influxdb.

Telegraf: Donde se crean, guardan y modifican las configuraciones de Telegraf.

Scrapers: Es donde se crean y guardan scrapers, los cuales sacan información desde las páginas web asignadas.

API Tokens: Es donde se guardan los API tokens creados, también se les puede modificar los permisos asignados, pero solo mostrará la key al momento de crear el token.

3.3.2. Explorador de datos.

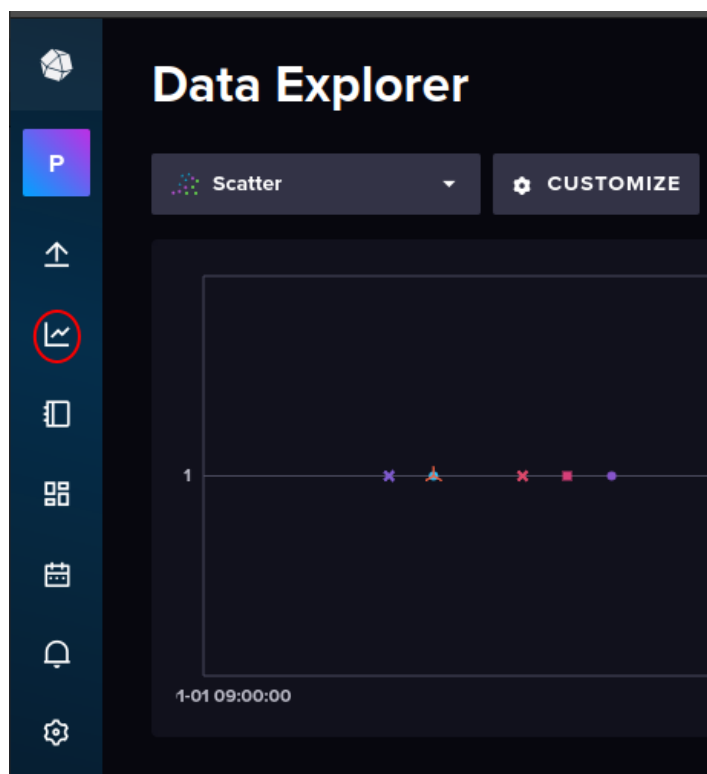


Figura 11: Pestaña de explorador de datos.

Esta pestaña abrirá el Explorador de datos, aquí se puede visualizar las medidas subidas a los buckets, también se puede crear una consulta Flux.

El explorador de datos consiste de 3 partes:



Figura 12: Gráfico.

Nos permite visualizar las mediciones subidas a influxdb, también se puede modificar el tipo de gráfico y la leyenda de las medidas del gráfico.

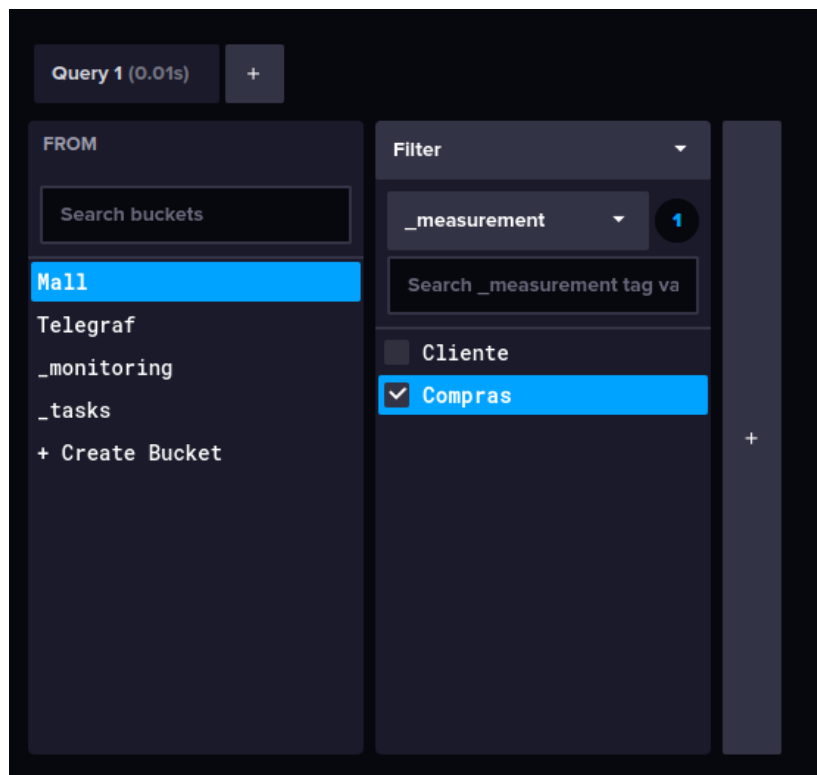


Figura 13: Constructor de consultas.

Nos permite agrupar las medidas dependiendo de las características necesitadas, en el caso de la figura anterior, permite hacerlo seleccionando casillas de las medidas que se quieran agrupar.

Para mayor control también se nos permite hacer una realizar la consulta como una consulta FLUX, para acceder a esta opción tienes que clickear el botón **SCRIPT EDITOR**, el cual abrirá un bloque de código donde podrás crear la consulta.



Figura 14: Consulta Flux.

Nota: para regresar al constructor de consultas es necesario guardar la consulta flux en la máquina ya que influxdb no tiene la capacidad de guardarlas, esto se debe a que el constructor es una versión simplificada del editor.

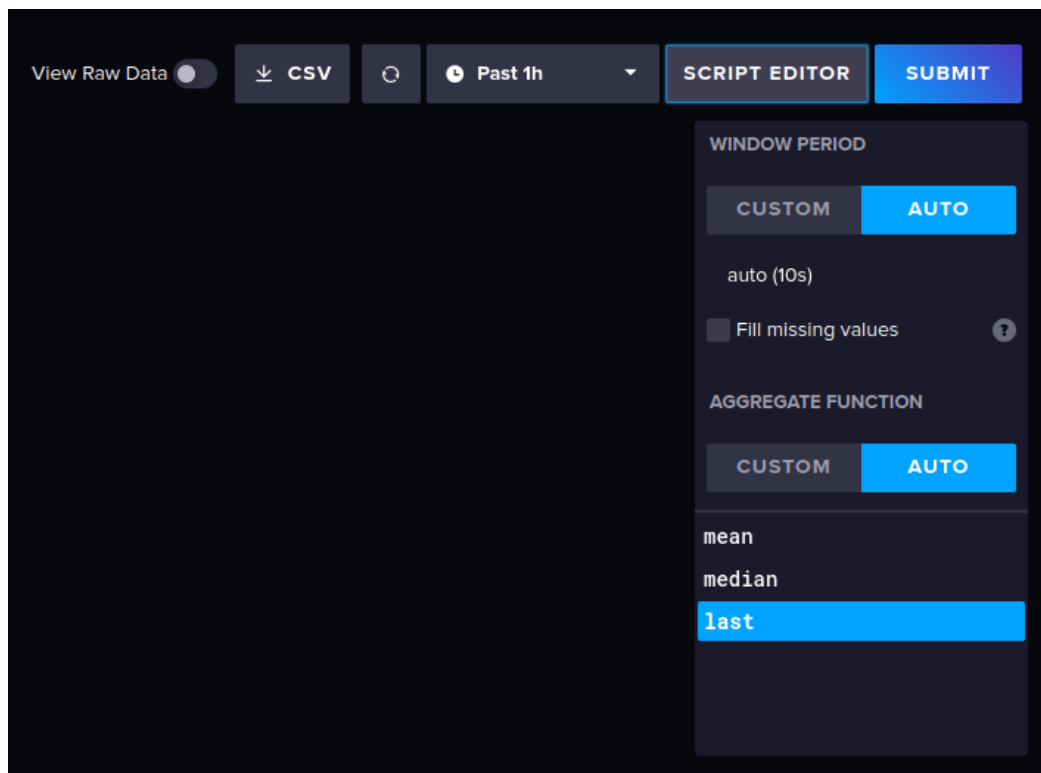


Figura 15: Configuración de visualización.

Nos permite cambiar el periodo de tiempo en el que se sacan las mediciones. Por defecto permite periodos desde 1 minuto hasta 30 días, pero también permite periodos personalizados que dependen de qué tan grande sea la información de la consulta (250mb descomprimidos).

Window period: es el tiempo mínimo entre cada timestamp de la medida, se pueden usar medidas predeterminadas o usar medidas personalizadas con precisión en los nanosegundos.

Aggregate function: Dado lo anterior, existen ocasiones donde habrán varias medidas dentro del periodo, aggregate function permite solucionar este problema, por defecto hace un promedio (mean) entre las medidas, pero también puede mostrar la primera(first) o la última(last) medida dentro del periodo. Existen otras formas de solucionar este problema, pero para ello es necesario solucionarlo en la consulta de FLUX.

3.3.3. Notebooks.

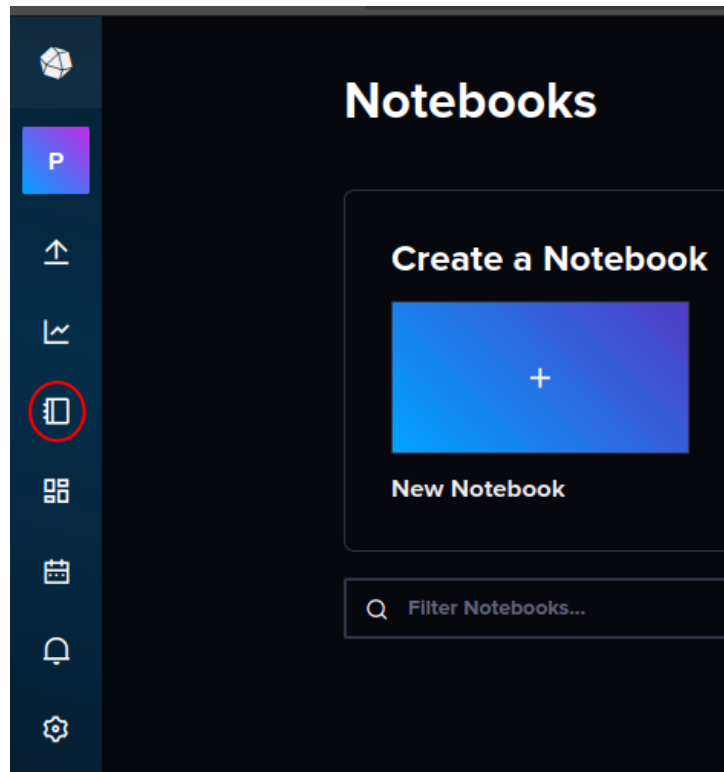


Figura 16: Notebooks.

Es una herramienta que permite visualizar, procesar y explorar los datos, Para mas informacion visita : <https://docs.influxdata.com/influxdb/v2/tools/notebooks/create-notebook/>

3.3.4. Dashboard.

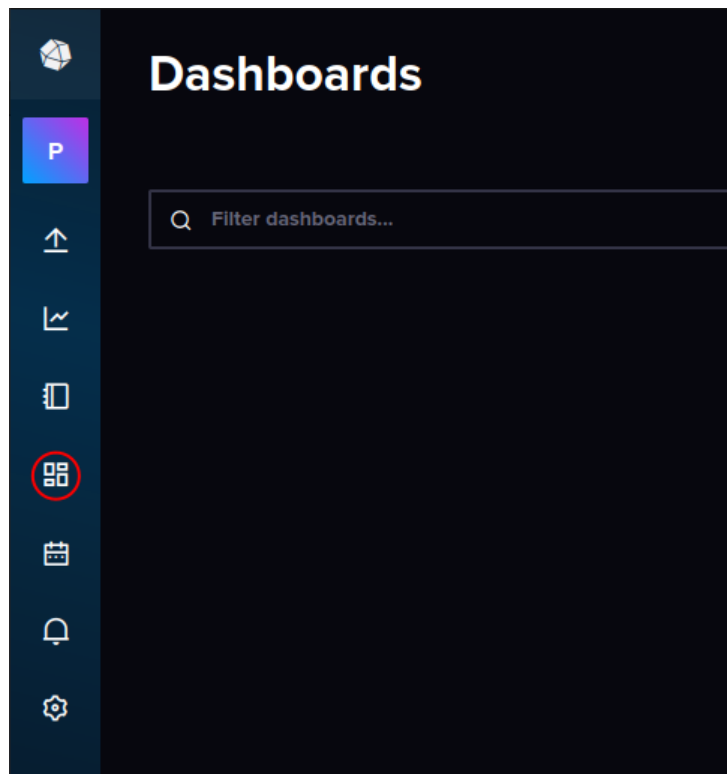


Figura 17: Dashboard.

Permite la visualización de varios gráficos de la pestaña Data Explorer al mismo tiempo.

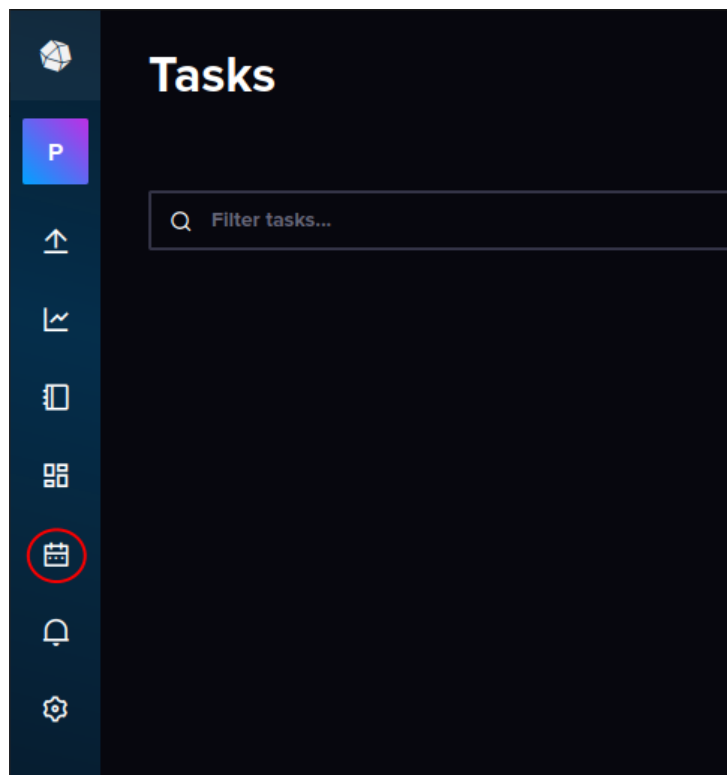


Figura 18: Task.

Crea Tareas que se ejecutarán a cierta hora del día. Para más información visita: <https://docs.influxdata.com/influxdb/v2/process-data/manage-tasks/create-task/#create-a-task-in-the-task-ui>

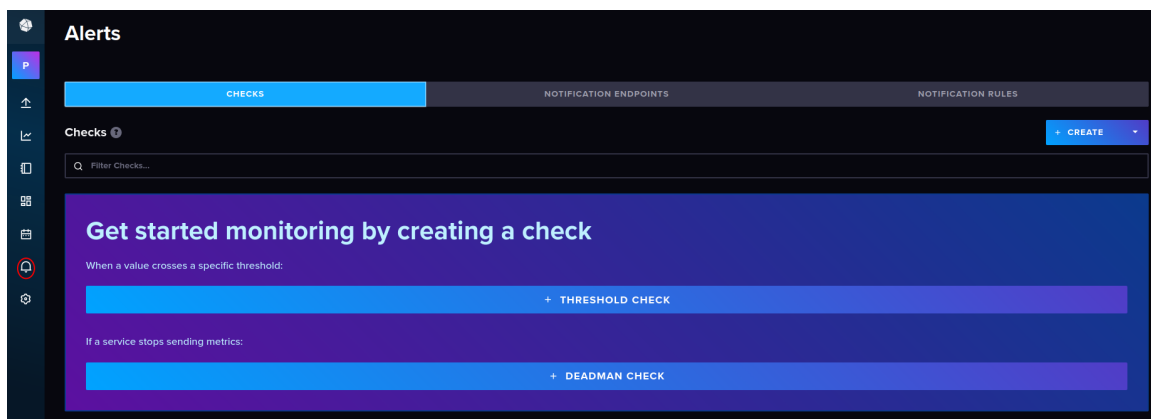


Figura 19: Alerta.

Permite la creación de alertas para notificar eventos, además permite ver el historial de dichas alertas. para más información visita: <https://docs.influxdata.com/influxdb/cloud/monitor-alert/>

3.3.5. Precauciones.

Influxdb no puede modificar la zona horaria, esto se tiene que tener en cuenta cuando se suban las medidas a la base de datos.

Influxdb no permite la subida de medidas que están después de la fecha actual.

4. Proyecto en influxdb.

Como ya fue dicho en el resumen, para poder entender el uso de influxdb y el cliente de python, realice un simulador de agente, en el cual veía como la temperatura fuera del mall podía afectar a la estadía y a las ventas de las tiendas dentro del mall.

Para ello hice un programa en Python que consta de 2 partes: la generación de las mediciones y el procesamiento de estos datos.

4.1. Generación de datos.

Para generar las mediciones necesarias par el procesamiento de los datos, cree dos programas en python llamados `gen_person.py` y `gen_temp.py`

gen_temp.py: Este programa genera y guarda las temperaturas dependiendo de la configuración dada en `config.json`, una vez generadas las temperaturas, las guarda en un json llamado `temperatura.json`.

```
{
  "temperature": [
    [
      "8:00",
      9.5
    ],
    [
      "8:01",
      9.9
    ],
    [
      "8:02",
      9.5
    ],
    [
      "8:03",
      10.0
    ],
    [
      "8:04",
      9.6
    ],
    [
      "8:05",
      9.9
    ],
  ],
}
```

Figura 20: Temperatura generada.

gen_person.py: Primero llama a gen_temp.py para luego generar las personas del mall, una vez terminada la generación de los nombres, empieza a simular el movimiento de las personas dentro del mall.

```
-----
Fecha          : 2025-01-01 14:22:00
Alida Peso Saez realizo compra en tienda 4a
Numero         : 82
-----
Fecha          : 2025-01-01 14:23:00
Nombre de cliente : Alida Peso Saez
Temperatura afuera : 5.6C
Probabilidad de salir : 13.53%
Tienda actual     : 4a
Hora            : 11:23
Numero          : 52
-----
Fecha          : 2025-01-01 14:24:00
Nombre de cliente : Alida Peso Saez
Temperatura afuera : 6.1C
Probabilidad de salir : 13.68%
Tienda actual     : 4a
Hora            : 11:24
Numero          : 45
-----
Fecha          : 2025-01-01 14:25:00
Nombre de cliente : Alida Peso Saez
Temperatura afuera : 5.9C
Probabilidad de salir : 13.62%
Tienda actual     : 4a
Hora            : 11:25
Numero          : 75
-----
Fecha          : 2025-01-01 14:26:00
Nombre de cliente : Alida Peso Saez
Temperatura afuera : 5.7C
Probabilidad de salir : 13.56%
Tienda actual     : 4a
Hora            : 11:26
Numero          : 74
-----
```

Figura 21: Extracto de la simulación de personas.

- 1.- Toma un nombre de la lista de nombres y le asigna una hora de entrada, probabilidad de cambiarse de tienda, probabilidad de compra y la posición inicial (“Entrada”).
- 2.- La persona empieza a moverse dentro del mall hasta llegar a una tienda.
- 3.- Una vez dentro de la tienda, se quedará dentro de ella hasta que se cambie, lo cual depende de la probabilidad de cambio. Mientras esté dentro de la tienda podrá comprar algún item dentro de la tienda (esta compra se subirá a influxdb).
- 4.- Si al cambiarse de tienda se mueve a una tienda en otro piso, tendrá un estado de “tránsito” por cada piso de diferencia.
- 5.- Cuando elija una tienda también puede elegir salir del mall, con lo cual se termina la simulación de la persona y comenzará con la siguiente.

Cada movimiento es guardado en influxdb, contiene: nombre, hora, tienda actual y la temperatura que había afuera del mall al realizar el movimiento.

4.2. Procesamiento de datos.

El procesamiento de datos consta de un solo programa en python llamado data_process.py, el cual descarga las medidas guardadas en influxdb y crea métricas, con las cuales se generarán gráficos que muestren la información.

El program consta de 6 métricas:

1.- Tiempo promedio por persona en mall:

Muestra el tiempo que cada persona estuvo en el mall y genera un gráfico que muestra cuántas personas se quedaron x cantidad de tiempo.

2.-Cantidad de personas por hora:

Muestra la cantidad de personas que estuvieron en el mall por hora y genera un gráfico que muestra dicha métrica.

3.-Ventas totales por tiendas:

Muestra las ventas totales por tienda que se hicieron a lo largo del día y genera un gráfico con estos datos.

4.- Ventas totales por hora:

Muestra la suma de las ventas de todas las tiendas que se hicieron por hora y genera un gráfico con estos datos.

5.- Ventas por hora por tienda:

Junta las dos métricas anteriores, generando un gráfico que muestra las ventas de las tiendas individualmente por hora, y genera una serie de gráficos que muestra las ventas por hora

6.- Personas por tiendas por minuto:

Muestra la cantidad de gente que hubieron por tienda por minuto, con lo cual se generan una serie de gráfico que muestra el total de personas que hubieron por minuto y los separa por hora

Para probar el proyecto visita: https://github.com/Acement/practica_influxdb


```

-----
Opcion 3

Tienda 1a vendio 32 productos
Tienda 1b vendio 27 productos
Tienda 1c vendio 42 productos
Tienda 1d vendio 51 productos
Tienda 2a vendio 48 productos
Tienda 2b vendio 56 productos
Tienda 2c vendio 43 productos
Tienda 2d vendio 38 productos
Tienda 3a vendio 39 productos
Tienda 3b vendio 45 productos
Tienda 3c vendio 31 productos
Tienda 3d vendio 47 productos
Tienda 4a vendio 39 productos
Tienda 4b vendio 37 productos
Tienda 4c vendio 48 productos
Tienda 4d vendio 40 productos
Tienda 5a vendio 39 productos
Tienda 5b vendio 34 productos
Tienda 5c vendio 43 productos
Tienda 5d vendio 39 productos

Ventas totales: 818
-----

```

Figura 22: Ejemplo de métrica nº 4.

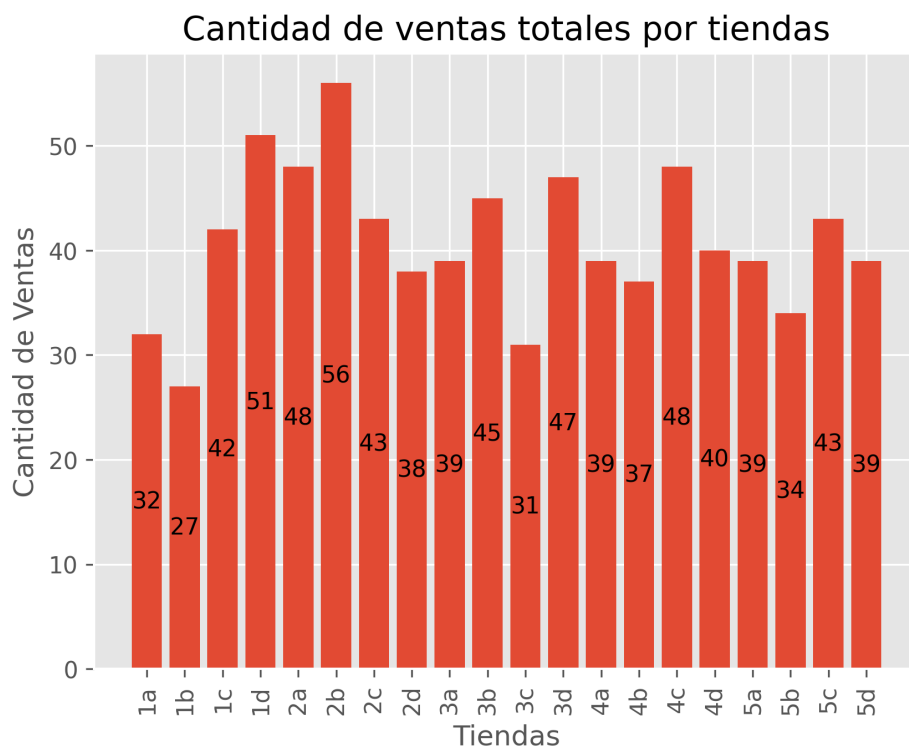


Figura 23: Gráfico generado con métrica nº 4.

5. Referencias.

Install Influxdb, influxdata, <https://docs.influxdata.com/influxdb/v2/install/>

Telegraf Documentation, influxdata, <https://docs.influxdata.com/telegraf/v1/>

The Complete Guide to InfluxDB 2, Rawkode Academy, 2021, <https://www.youtube.com/live/dIG8t67JcIY?si=ZNsLENA7IoznpI9h>

Monitorización de servidores con Grafana, Telegraf e InfluxDB, NullSafe Architect, 2019, <https://youtu.be/GLE71pIHUU8?si=WenV2vhJU0Q5K0-J>