**InfluxDB**

**¿Qué es InfluxDB?**

Se trata de un **sistema de gestión de bases de datos** creada por los desarrolladores de InfluxData, Inc. Es un software gratuito y de código abierto, aunque posee una versión comercial llamada **InfluxDB Enterprise**, con la que los desarrolladores ofrecen mantenimiento y servidores dedicados a los clientes que lo adquieran.

InfluxDB ha sido creado en **Go**, el lenguaje de programación desarrollado por Google, y que hoy en día es muy popular en todo tipo de aplicaciones.

**¿Por qué usar InfluxDB?**

InfluxDB fue creado con el objetivo de almacenar **series temporales**. Las bases de datos que se gestionan son llamadas **Time Series Data Bases (TSDB)**, cuya principal finalidad es la de almacenar un flujo continuo de datos, extraídos comúnmente de instrumentos o herramientas de recolección de métricas (Termómetros, sensores, IoT…).

InfluxDB es capaz de procesar rápidamente todos esos datos y almacenarlos en una base de datos TSDB, que tienden a ser bastante simples, con dos o tres columnas.

Ejemplo de una TSDB:

|  |  |  |
| --- | --- | --- |
| **Hora** | **Sensor** | **Valor** |
| 29/11/2021-19:09:56 | Sensor\_1 | 10,23 |
| 29/11/2021-19:09:57 | Sensor\_2 | 10,29 |
| 29/11/2021-19:10:43 | Sensor\_1 | 11,01 |

Cabe destacar que lo que caracteriza una **TSDB** es que almacena un campo **time** con la hora o fecha en la que se ha recogido la métrica, aunque no sea un campo obligatorio.

**¿Cómo almacena los datos?**

InfluxDB tiene una estructura que separa el contenido de las bases de datos en dos conceptos: **Tags** y **fields:**

Por una parte, los **tags o etiquetas** son usados como *metadatos* que ayudan a organizar las tablas, cumpliendo la función de índices. Un ejemplo sería la columa **Sensor** del ejemplo anterior, ya que no contiene ningún valor por sí mismo, pero viene emparejada con el **valor** que representa a esa **Hora**.

Por otra parte, tenemos los **fields o campos.** Almacenan los valores que querremos evaluar en un futuro, es decir, la **métrica** que nos interesa guardar.

Una opción de subir nuevas métricas a InfluxDB es en formato **JSON**, que será utilizado en esta práctica para actualizar la base de datos. Se envía el diccionario al cliente de InfluxDB indicando los tags y fields que queramos añadir.

**Aplicación y cliente en Docker**

**Objetivo**

El objetivo del sistema creado será disponer de una base de datos con InfluxDB y utilizar un pequeño cliente improvisado creado en Python para crear y leer datos de la base de datos.

El sistema simula una especie de “almacen de comentarios”, en el que puedes añadir una **frase o comentario** junto a una palabra descriptiva sobre el **tema** del comentario.

**Sistema creado**

Se lanzará un contenedor Docker con una Imagen de [InfluxDB](https://hub.docker.com/_/influxdb), a la que accederá nuestra aplicación cliente desde otro contenedor a través del puerto 8086. Para ejecutar el cliente se debe llamar a la aplicación encapsulada dentro del contenedor cliente:

**docker exec -it client influx\_client**

Con ello ejecutaras la aplicación en modo interactivo de docker. Una vez en ella, tendrás dos opciones: **Escribir comentario**, con la que añadirás datos a la BBDD y **leer comentarios**, con lo que tendrás acceso a todos los comentarios almacenados.

Como extensión, se podrán guardar los comentarios leídos en un archivo de texto, que serán guardados en un volumen ubicado en **/tmp/influx\_comentarios/** del host.

**Código generado**

**Dockerfile**

Por un lado, tenemos el **Dockerfile**, encargado de crear nuestro cliente a partir de una imagen de Python:3.

La imagen será subida al repositorio en <https://hub.docker.com/r/xafer337/influx_client>.

FROM python:3 # Imagen base de Python 3

WORKDIR /usr/local/bin/ # Cambiamos el directorio en el que trabajamos a uno en la variable de entorno $PATH

RUN pip install influxdb # Instalamos el cliente Python para comunicarnos con InfluxDB

COPY influx\_client . # Copiamos el código del cliente Python que hemos creado en la imagen. Se adjunta el código del programa comentado

RUN chmod +x influx\_client # Otorgamos permisos de ejecución a la aplicación cliente. En el programa cliente especificamos en la primera línea que debe ejecutarse con Python.

**Docker-compose**

En el **docker-compose** lanzaremos nuestro cliente y una instancia de InfluxDB, en contenedores separados.

version: '3' # Versión de docker-compose

services:

influxDB: # Contenedor con InfluxDB

image: influxdb:1.8.10 # Imagen provista de InfluxDB

container\_name: influxDB # Nombre identificativo para el contenedor

environment: # Variables de entorno usadas por InfluxDB

- INFLUXDB\_ADMIN\_USER=admin # Nombre del usuario administrador

- INFLUXDB\_ADMIN\_PASSWORD=admin # Contraseña del usuario administrador

- INFLUXDB\_DB=influx\_db # Base de datos base que se creará

volumes: # Almacenamiento persistente de la base de datos

- data\_influx:/var/lib/influxdb # Almacenamos el directorio de datos de InfluxDB

client: # Contenedor con la aplicación cliente

image: xafer337/influx\_client:v1.0.0 # Imagen creada por el Dockerfile

container\_name: client # Nombre identificativo para el contenedor

tty: true # Activar una pseudo terminal para evitar el cierre del contenedor

volumes: # Directorio en el que se almacenarán los comentarios leídos

- /tmp/influx\_comentarios/:/tmp/influx\_comentarios/

volumes:

data\_influx: # Volumen de almacenamiento persistente de InfluxDB