Taller NoSQL: Telemetría con Python + InfluxDB

CE-3101 Bases de Datos — Guía práctica

Entrega limite: Martes 28 Octubre 7pm

1. Propósito del taller

Este taller busca conectar los conceptos teóricos de las bases de datos **NoSQL** con un caso práctico: la **recolección y visualización de métricas del sistema (telemetría)** usando Python y una base de datos de series de tiempo en la nube, **InfluxDB Cloud**.

Los estudiantes aprenderán a:

- Obtener métricas del sistema (CPU y memoria).
- Almacenar las métricas en una base de datos NoSQL en la nube.
- Visualizar los resultados en un **dashboard con gráficos en tiempo real**.

2. ¿Qué es InfluxDB y su relación con NoSQL?

InfluxDB es una base de datos **NoSQL orientada a series de tiempo (TSDB)** diseñada para almacenar datos con marcas temporales —como métricas de sensores, registros de rendimiento, o tráfico de red— de manera eficiente y escalable.

Características:

- No usa tablas ni esquema fijo como en las bases relacionales.
- Los datos se agrupan en **measurements**, con **tags** (etiquetas de texto) y **fields** (valores numéricos).
- Optimizada para **consultas por rango temporal** y escritura masiva.
- Compatible con dashboards en la nube (visualización rápida).

Relación con NoSQL: InfluxDB pertenece al ecosistema NoSQL porque:

1. No tiene un esquema rígido: cada punto puede tener diferentes campos.
2. Prioriza la velocidad de escritura y la consulta de agregaciones, no las relaciones.
3. Está orientada a la **temporalidad de los datos**, no a la integridad referencial.

Por qué es valioso este taller: Permite comprender cómo los modelos NoSQL aplican a contextos modernos como:

- Monitoreo de sistemas en tiempo real.
- IoT (Internet de las Cosas) y sensores distribuidos.
- Aplicaciones en la nube y microservicios que generan telemetría constante.

3. Requisitos previos

- Python 3.10 o superior.
- Cuenta gratuita en InfluxDB Cloud (https://www.influxdata.com/products/influxdb-cloud/).
- Librerías: `psutil`, `influxdb-client`, `python-dotenv`.

4. Crear entorno virtual (venv)

Un **venv** es un entorno de Python aislado donde se instalan dependencias sin afectar el sistema.

```
# Crear carpeta y entorno virtual
mkdir -p taller_nosql && cd taller_nosql
python -m venv .venv

# Activar entorno
source .venv/bin/activate    # (PowerShell: .venv\Scripts\Activate.ps1)

# Instalar dependencias
pip install --upgrade pip wheel
pip install psutil influxdb-client python-dotenv
```

5. Configuración en InfluxDB Cloud

5.1. Crear cuenta y bucket

1. Ingresa a https://cloud2.influxdata.com/. 2. Crea una cuenta con tu **correo institucional**. 3. En el menú lateral, abre **Data → Buckets** y crea un bucket llamado `telemetria`.

5.2. Obtener credenciales necesarias

Desde el menú **Load Data → Tokens → Generate API Token**, crea un token con permisos de lectura y escritura. Además, toma nota de:

- **URL:** dirección de tu instancia (ej. `https://us-east-1-1.aws.cloud2.influxdata.com`)
- **Organization (ORG):** nombre de tu organización (esquina superior derecha).
- **Bucket:** el contenedor de tus series (`telemetria`).
- **Token:** la clave de autenticación.

5.3. Archivo .env

Crea el archivo de variables de entorno `.env` en la raíz del proyecto:

```
INFLUX_URL=https://<tu-url>
INFLUX_ORG=<tu-org>
INFLUX_BUCKET=telemetria
INFLUX_TOKEN=<tu-token>
PERIOD_SEC=5
```

Nota: Nunca compartas este archivo ni lo subas a GitHub.

6. Lectura de métricas con Python

Para capturar datos del sistema usaremos `psutil`.

6.1. CPU (%)

Qué mide: el porcentaje de tiempo que la CPU está activa procesando tareas.
Llamada clave: `psutil.cpu_percent(interval=1)`
Ejemplo guía:

```
def leer_cpu():
    """Devuelve el uso de CPU (%) promedio en 1 segundo."""
    # TODO: implementar psutil.cpu_percent(interval=1)
    pass
```

6.2. Memoria (MB y %)

Qué mide: el uso de la memoria física (RAM) en megabytes y porcentaje.
Llamada clave: `psutil.virtual_memory()`
Ejemplo guía:

```
def leer_memoria():
    """Devuelve {"used_mb": float, "percent": float}."""
    # TODO: usar psutil.virtual_memory()
    # TODO: convertir mem.used a MB (mem.used / 1024**2)
    pass
```

7. Modelo de datos para InfluxDB

InfluxDB organiza la información como:

- **Measurement:** el tipo de dato registrado (ej. `sys_metrics`).
- **Tags:** texto para filtrar, como `hostname`, `os`.
- **Fields:** valores medibles: `cpu_percent`, `mem_percent`.
- **Timestamp:** instante en que se toma la métrica.

Pista:

```
from influxdb_client import Point, WritePrecision

def construir_punto(cpu, mem, hostname, osver):
    # TODO: crear Point("sys_metrics")
    #       agregar tags (hostname, os)
    #       agregar fields (cpu_percent, mem_used_mb, mem_percent)
    #       agregar timestamp actual
    pass
```

8. Flujo general del programa

1. Cargar configuración del archivo `.env`.
2. Identificar el equipo (`socket.gethostname()`).
3. Leer CPU y memoria periódicamente.
4. Construir y enviar el punto a InfluxDB.
5. Repetir en bucle cada `PERIOD_SEC` segundos.

9. Visualización en la nube (Data Explorer y Dashboard)

9.1. Ver datos en Data Explorer

1. Ingresa a https://cloud2.influxdata.com/.
2. Abre **Data Explorer**.
3. Selecciona el **bucket** `telemetria`.
4. Measurement: `sys_metrics`.
5. Fields: `cpu_percent`, `mem_percent`.
6. Aplica filtros por `hostname` si varios equipos reportan datos.

9.2. Crear un Dashboard en InfluxDB

El objetivo es construir un **Dashboard** con gráficos que muestren las métricas en tiempo real desde la nube.

1. En el menú lateral, selecciona **Dashboards → Create Dashboard**.
2. Asigna un nombre (por ejemplo: `Monitoreo del Sistema`).
3. Haz clic en **Add Cell → Visualization → Line Chart**.
4. En la consulta selecciona tu bucket (`telemetria`) y measurement (`sys_metrics`).
5. Elige los campos que quieras graficar: `cpu_percent` y `mem_percent`.
6. Ajusta el rango de tiempo (por ejemplo: últimos 15 minutos).
7. Guarda el panel con **Save to Dashboard**.

Ejemplo de Dashboard esperado:

Figura 1: Ejemplo de visualización en InfluxDB Cloud con métricas CPU y Memoria.

10. Extensión opcional: comparación entre equipos

Ejecuta el script en dos dispositivos distintos (o una VM adicional). En el dashboard, agrega una leyenda por `hostname` para comparar curvas.

Preguntas guía:

- ¿Qué máquina presenta mayor carga promedio?
- ¿Qué factores del hardware o software podrían explicarlo?

11. Solución de problemas comunes

- **401/403:** token inválido o sin permisos de escritura.
- **No aparecen puntos:** verifica el bucket y measurement correctos.
- **Error SSL:** añade `tls=True` al cliente InfluxDB.
- **psutil no instala:** actualiza `pip` y `wheel`.

12. Rúbrica

- (30 pts) Lectura correcta de CPU y memoria.
- (30 pts) Envío de datos a InfluxDB usando variables del `.env`.
- (25 pts) Dashboard funcional con gráfico de métricas en la nube.
- (15 pts) Reporte explicativo con observaciones.
- (+10 pts) Comparativa multi-host.

13. Referencias

- InfluxDB Cloud: https://www.influxdata.com/products/influxdb-cloud/
- InfluxDB Client para Python: https://github.com/influxdata/influxdb-client-python
- psutil: https://psutil.readthedocs.io/
- python-dotenv: https://github.com/theskumar/python-dotenv