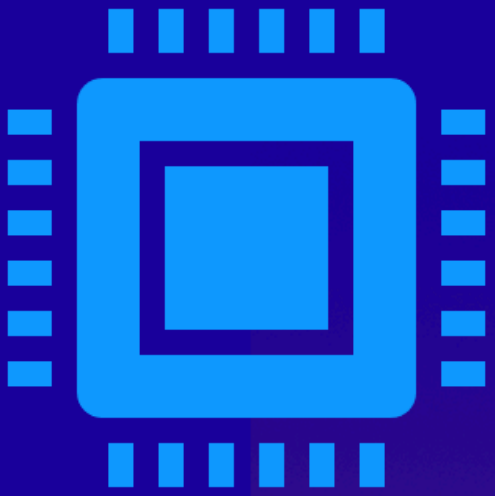
IOT
Proyecto
2025

Sistema de Monitoreo IoT en Tiempo Real con Apache Kafka



ECOTEC

Introducción

El Internet de las Cosas (IoT) conecta dispositivos físicos como sensores y actuadores a sistemas distribuidos capaces de procesar grandes volúmenes de datos. En este proyecto se implementa un sistema de monitoreo en tiempo real utilizando **Apache Kafka** para el transporte de eventos, **InfluxDB** para almacenamiento de métricas y **Grafana** para visualización.
La finalidad es demostrar cómo un entorno distribuido puede recibir información de múltiples sensores, procesarla de manera eficiente y mostrar resultados inmediatos para la toma de decisiones.

Objetivos

Objetivo General

Diseñar e implementar un sistema IoT distribuido que permita monitorear sensores en tiempo real, procesar los datos y generar visualizaciones y alertas.

Objetivos Específicos

- Configurar un entorno dockerizado con Kafka, InfluxDB y Grafana.
- Desarrollar un productor en Python que simule sensores enviando datos a Kafka.
- Implementar un consumidor que procese los mensajes y los almacene en InfluxDB.
- Visualizar métricas en Grafana para validar el funcionamiento del sistema.
- Evaluar la escalabilidad y tolerancia a fallos del sistema.

Tecnologías Utilizadas

- **Apache Kafka**: plataforma de mensajería distribuida para transmisión de eventos.
- **Python/Node.js**: scripts de productores y consumidores.
- **InfluxDB**: base de datos orientada a series temporales para métricas.
- **Grafana**: herramienta de visualización en tiempo real.
- **Docker Compose**: orquestación de contenedores para simplificar la implementación.

Arquitectura del Sistema

El sistema de monitoreo IoT en tiempo real se organiza en varias capas que trabajan de manera distribuida:

1. **Dispositivo IoT (Sensor simulado)**
    - Genera datos de temperatura, humedad u otras métricas.
    - Envía los datos mediante un **Kafka Producer**.
2. **Kafka Broker**
    - Recibe los eventos de los productores.
    - Garantiza la entrega ordenada y confiable de los mensajes.
3. **Kafka Consumer**
    - Procesa los mensajes recibidos.
    - Transforma los datos y los envía a la base de datos.
4. **InfluxDB**
    - Almacena las métricas en formato de series temporales.
    - Permite consultas eficientes sobre datos históricos.
5. **Grafana**
    - Se conecta a InfluxDB.
    - Genera dashboards y visualizaciones en tiempo real.

Flujo resumido:
Sensor → Producer → Kafka Broker → Consumer → InfluxDB → Grafana

Componentes Dockerizados

Para simplificar la implementación, cada servicio se ejecuta en un contenedor independiente:

- **Zookeeper**: Servicio requerido por Kafka para coordinar brokers.
- **Kafka**: Broker principal para transmisión de eventos.
- **InfluxDB**: Base de datos de métricas.
- **Grafana**: Visualización de datos en dashboards.
- **Python Producer**: Script que simula sensores y envía datos a Kafka.
- **Python Consumer**: Script que recibe mensajes y los almacena en InfluxDB.

Características Clave

- Escalabilidad horizontal: se pueden añadir más consumidores sin afectar el sistema.
- Relectura de mensajes: Kafka permite volver a procesar datos históricos.
- Visualización en tiempo real: dashboards dinámicos en Grafana.

Desarrollo paso a paso en Linux para el proyecto Kafka + InfluxDB + Grafana

Preparación del entorno en Linux

Requisitos

- **Sistema:** Linux (Debian/Ubuntu recomendado)
- **Herramientas:** Docker, Docker Compose, Git, Python 3.10+

Instalación de Docker y Docker Compose

- **Actualizar paquetes:**
    - sudo apt update && sudo apt upgrade -y

```
cristo1256@server01:~$ sudo apt update && sudo apt upgrade -y

[sudo] password for cristo1256:
```

- **Instalar Docker:**
    - sudo apt install -y ca-certificates curl gnupg

```
cristo1256@server01:~$ sudo apt install -y ca-certificates curl gnupg
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20240203).
curl ya está en su versión más reciente (8.5.0-2ubuntu10.6).
gnupg ya está en su versión más reciente (2.4.4-2ubuntu17.3).
fijado gnupg como instalado manualmente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 117 no actualizados.
cristo1256@server01:~$
```

    - sudo install -m 0755 -d /etc/apt/keyrings
    - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

```
cristo1256@server01:~$ sudo install -m 0755 -d /etc/apt/keyrings
cristo1256@server01:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
cristo1256@server01:~$
```

    - echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release; echo $VERSION_CODENAME) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
cristo1256@server01:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release; echo $VERSION_CODEN
AME) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
cristo1256@server01:~$
```

    - sudo apt update
    - sudo apt install -y docker-ce docker-ce-cli containerd.io

```
cristo1256@server01:~$ sudo apt install -y docker-ce docker-ce-cli containerd.io
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz slirp4netns
Paquetes sugeridos:
  cgroupfs-mount | cgroup-lite docker-model-plugin
Se instalarán los siguientes paquetes NUEVOS:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz slirp4netns
0 actualizados, 9 nuevos se instalarán, 0 para eliminar y 117 no actualizados.
Se necesita descargar 105 MB de archivos.
Se utilizarán 437 MB de espacio de disco adicional después de esta operación.
Des:1 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.29-1~ubuntu.24.04~noble [31,9 MB]
Des:2 http://ec.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65,6 kB]
Des:3 http://ec.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63,8 kB]
Des:4 http://ec.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34,9 kB]
Des:5 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:28.5.2-1~ubuntu.24.04~noble [16,0 MB]
Des:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:28.5.2-1~ubuntu.24.04~noble [19,8 MB]
52% [6 docker-ce 5.505 kB/19,8 MB 28%]
```

- **Agregar tu usuario al grupo docker (evita sudo):**
    - sudo usermod -aG docker $USER

```
cristo1256@server01:~$ sudo usermod -aG docker $USER
cristo1256@server01:~$
```

    - Cierra sesión y vuelve a entrar, o ejecuta: newgrp docker

```
cristo1256@server01:~$ newgrp docker
cristo1256@server01:~$
```

- **Instalar Docker Compose (plugin):**
    - sudo apt install -y docker-compose-plugin
    - Verifica: docker compose version

```
cristo1256@server01:~$ sudo apt install -y docker-compose-plugin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
docker-compose-plugin ya está en su versión más reciente (2.40.3-1~ubuntu.24.04~noble).
fijado docker-compose-plugin como instalado manualmente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 117 no actualizados.
cristo1256@server01:~$ docker compose version
Docker Compose version v2.40.3
cristo1256@server01:~$
```

- **Instalar Python y dependencias:**
    - sudo apt install -y python3 python3-pip git

```
Des:57 http://ec.archive.ubuntu.com/ubuntu noble-updates/universe amd64 python3-pip all 24.0+dfsg-1ubuntu1.3 [1.3
Descargados 74,0 MB en 10s (7.782 kB/s)
Extrayendo plantillas para los paquetes: 100%
(Leyendo la base de datos ... 127731 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../dpkg_1.22.6ubuntu6.5_amd64.deb ...
Desempaquetando dpkg (1.22.6ubuntu6.5) sobre (1.22.6ubuntu6.1) ...
Configurando dpkg (1.22.6ubuntu6.5) ...
(Leyendo la base de datos ... 127731 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../00-libpython3.12t64_3.12.3-1ubuntu0.8_amd64.deb ...
Desempaquetando libpython3.12t64:amd64 (3.12.3-1ubuntu0.8) sobre (3.12.3-1ubuntu0.7) ...
Preparando para desempaquetar .../01-python3.12_3.12.3-1ubuntu0.8_amd64.deb ...
Desempaquetando python3.12 (3.12.3-1ubuntu0.8) sobre (3.12.3-1ubuntu0.7) ...
Preparando para desempaquetar .../02-libpython3.12-stdlib_3.12.3-1ubuntu0.8_amd64.deb ...
Desempaquetando libpython3.12-stdlib:amd64 (3.12.3-1ubuntu0.8) sobre (3.12.3-1ubuntu0.7) ...
Preparando para desempaquetar .../03-python3.12-minimal_3.12.3-1ubuntu0.8_amd64.deb ...
Desempaquetando python3.12-minimal (3.12.3-1ubuntu0.8) sobre (3.12.3-1ubuntu0.7) ...
Preparando para desempaquetar .../04-libpython3.12-minimal_3.12.3-1ubuntu0.8_amd64.deb ...
Desempaquetando libpython3.12-minimal:amd64 (3.12.3-1ubuntu0.8) sobre (3.12.3-1ubuntu0.7) ...
Progreso: [  5%] [########.....................................................................................
```

Estructura de proyecto

- **Carpetas:**
    - iot-kafka/
        - docker-compose.yml
        - producer/
            - requirements.txt
            - producer.py
        - consumer/
            - requirements.txt
            - consumer.py
- **Crear carpeta base:**
    - mkdir -p iot-kafka/producer iot-kafka/consumer && cd iot-kafka

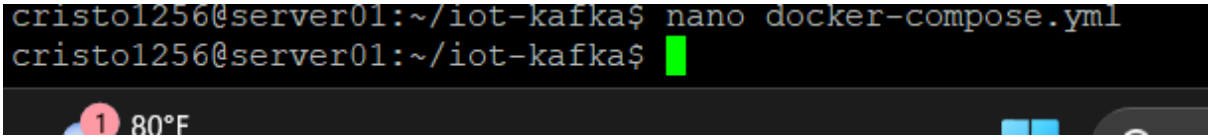
```
cristo1256@server01:~$ mkdir -p iot-kafka/producer iot-kafka/consumer && cd iot-kafka
cristo1256@server01:~/iot-kafka$
```

Docker Compose inicial

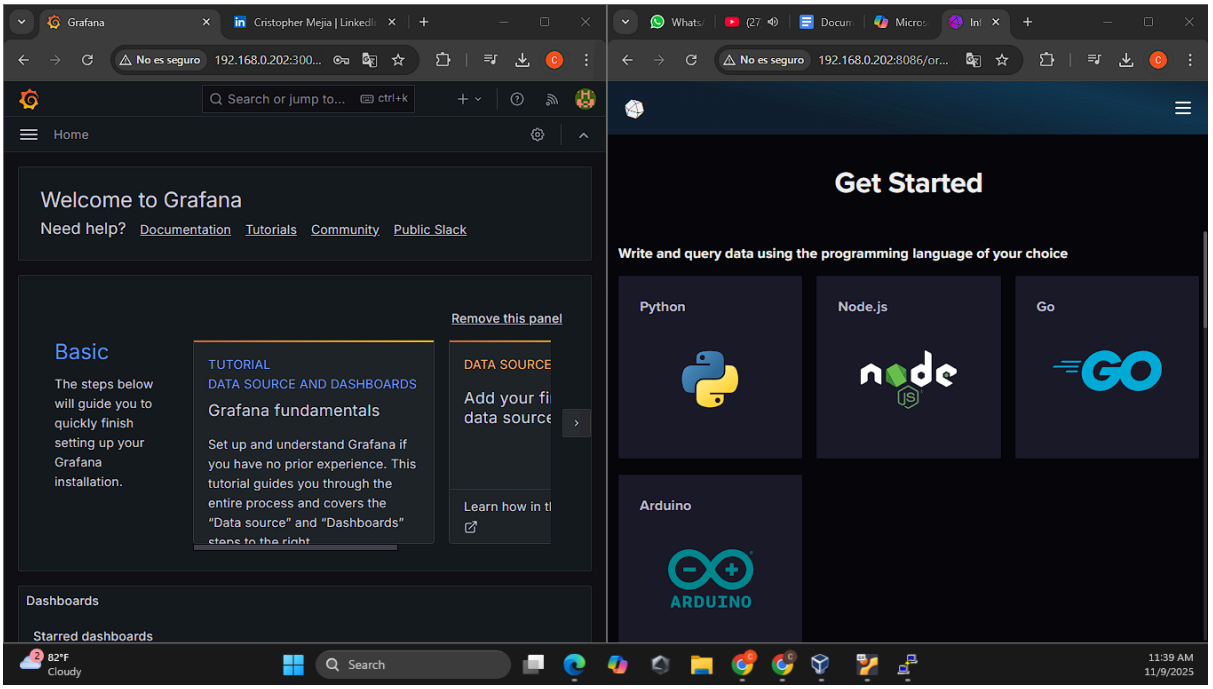
Crea docker-compose.yml con servicios para Zookeeper, Kafka, InfluxDB y Grafana.

```
version: "3.9"

services:
  zookeeper:
    image: bitnami/zookeeper:3.9
    environment:
      - ZOOKEEPER_CLIENT_PORT=2181
      - ZOOKEEPER_TICK_TIME=2000
    ports:
      - "2181:2181"
    networks:
      - iot-net

  kafka:
    image: bitnami/kafka:3.7
    depends_on:
      - zookeeper
    environment:
      - KAFKA_BROKER_ID=1
      - KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
      - KAFKA_LISTENERS=PLAINTEXT://:9092
      - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka:9092
      - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1
    ports:
      - "9092:9092"
```

```
    networks:
      - iot-net

  influxdb:
    image: influxdb:2.7
    ports:
      - "8086:8086"
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=admin123
      - DOCKER_INFLUXDB_INIT_ORG=ecotec
      - DOCKER_INFLUXDB_INIT_BUCKET=iot_metrics
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=ecotec-token-123
    volumes:
      - influx-data:/var/lib/influxdb2
    networks:
      - iot-net

  grafana:
    image: grafana/grafana:10.4.2
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=admin123
    depends_on:
      - influxdb
    networks:
      - iot-net

networks:
  iot-net:

volumes:
  influx-data:
```



- **Levantar servicios:**
  - docker compose up -d
- **Validar estados:**
  - docker compose ps



- **Accesos:**
  - **InfluxDB UI:** http://localhost:8086 (admin / admin123)
  - **Grafana:** http://localhost:3000 (admin / admin123)

Productor en Python (simulación de sensores)

Dependencias del productor

Crea producer/requirements.txt:

kafka-python-ng==2.0.2

Instala:

- sudo apt install python3-kafka
- pip3 install -r requirements.txt --break-system-packages

```
cristo1256@server01:~/iot-kafka/producer$ nano requirements.txt
cristo1256@server01:~/iot-kafka/producer$ pip3 install -r producer/requirements.txt
```

```
cristo1256@server01:~/iot-kafka/producer$ pip3 install -r requirements.txt --break-system-packages
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: kafka-python==2.0.2 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 1)) (2.0.2)
cristo1256@server01:~/iot-kafka/producer$
```

Código del productor

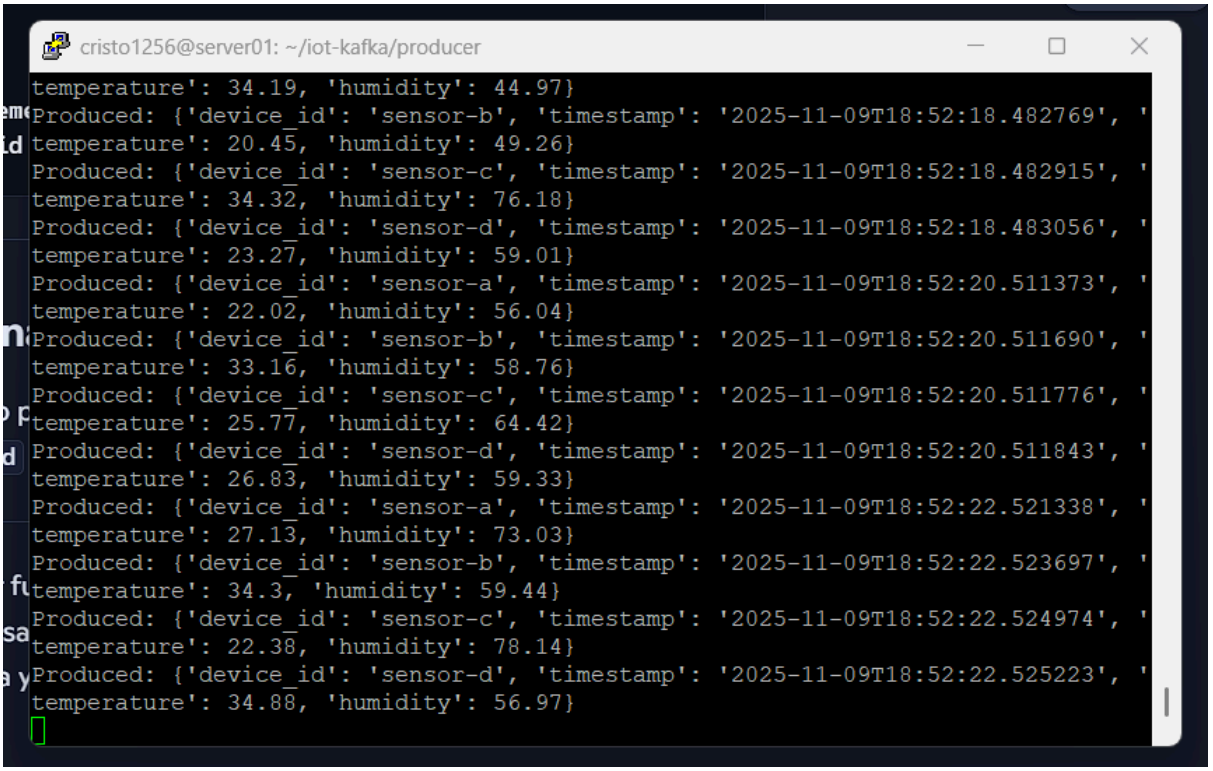
Crea producer/producer.py:

```
cristo1256@server01:~/iot-kafka/producer$ nano producer.py
cristo1256@server01:~/iot-kafka/producer$
```

```
import json
import random
import time
from datetime import datetime
from kafka import KafkaProducer

TOPIC = "iot.sensors"
BROKER = "localhost:9092"  # Si corres dentro de Docker, usa "kafka:9092"

def create_producer():
    return KafkaProducer(
        bootstrap_servers=BROKER,
        value_serializer=lambda v: json.dumps(v).encode("utf-8"),
        acks="all",
        retries=5
    )

def generate_sensor_data(device_id: str):
    return {
        "device_id": device_id,
        "timestamp": datetime.utcnow().isoformat(),
        "temperature": round(random.uniform(20.0, 35.0), 2),
        "humidity": round(random.uniform(40.0, 80.0), 2)
    }

def main():
    producer = create_producer()
    devices = ["sensor-a", "sensor-b", "sensor-c"]

    print(f"Sending messages to topic '{TOPIC}' on {BROKER} ...")
    while True:
        for d in devices:
            payload = generate_sensor_data(d)
            producer.send(TOPIC, value=payload)
            print("Produced:", payload)
        producer.flush()
        time.sleep(2)

if __name__ == "__main__":
    main()
```

- **Crear el tópico (opcional, Kafka lo crea automáticamente):**
    - docker exec -it $(docker ps --filter name=kafka --format "{{.ID}}") /opt/bitnami/kafka/bin/kafka-topics.sh --create --topic iot.sensors --bootstrap-server kafka:9092 --replication-factor 1 --partitions 3
- **Ejecutar productor (desde tu host Linux):**
    - python3 producer/producer.py

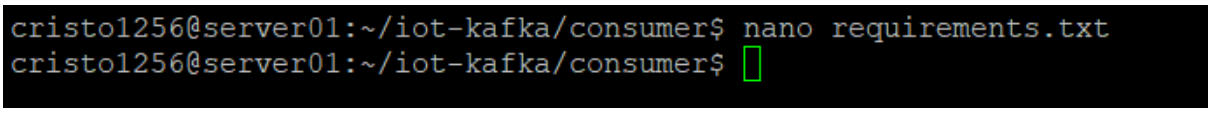
Tip: Si ejecutas el productor dentro de un contenedor, cambia BROKER = "kafka:9092".



Consumidor en Python (almacenamiento en InfluxDB)

Dependencias del consumidor

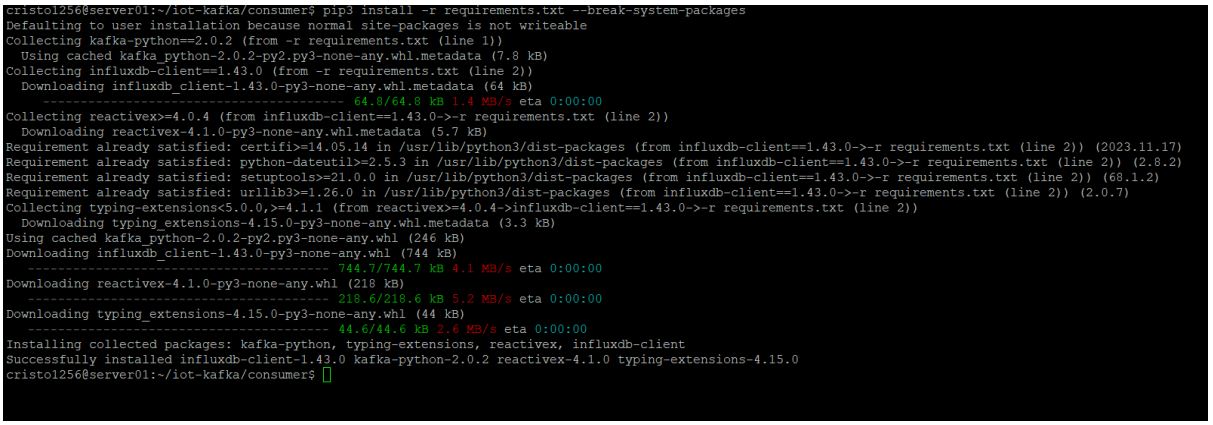
Crea consumer/requirements.txt:



kafka-python-ng==2.0.2
influxdb-client==1.43.0

Instala:

- pip3 install -r consumer/requirements.txt



Variables de InfluxDB

- **URL:** http://localhost:8086
- **Org:** ecotec
- **Bucket:** iot_metrics
- **Token:** ecotec-token-123

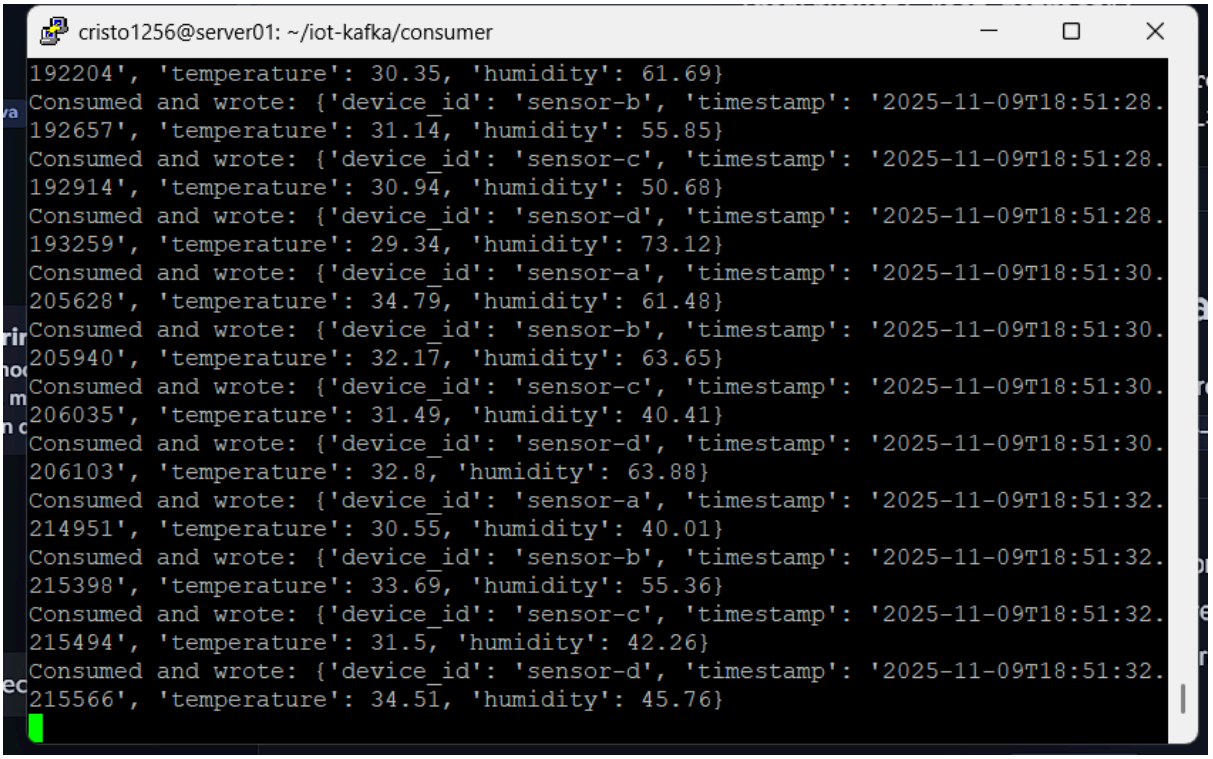
Código del consumidor

Crea consumer/consumer.py:

```
import json
from kafka import KafkaConsumer
from influxdb_client import InfluxDBClient, Point, WriteOptions

TOPIC = "iot.sensors"
BROKER = "localhost:9092"

INFLUX_URL = "http://localhost:8086"
INFLUX_TOKEN = "ecotec-token-123"
INFLUX_ORG = "ecotec"
INFLUX_BUCKET = "iot_metrics"

def create_consumer():
    return KafkaConsumer(
        TOPIC,
        bootstrap_servers=BROKER,
        auto_offset_reset="earliest",
        enable_auto_commit=True,
        value_deserializer=lambda v: json.loads(v.decode("utf-8")),
        group_id="iot-consumers"
    )

def create_influx_writer():
    client = InfluxDBClient(url=INFLUX_URL, token=INFLUX_TOKEN, org=INFLUX_ORG)
    write_api = client.write_api(write_options=WriteOptions(batch_size=1000, flush_interval=1000))
    return client, write_api

def to_point(data: dict) -> Point:
    p = Point("sensor_metrics") \
        .tag("device_id", data["device_id"]) \
        .field("temperature", float(data["temperature"])) \
        .field("humidity", float(data["humidity"]))
    return p

def main():
    consumer = create_consumer()
    client, write_api = create_influx_writer()

    print(f"Consuming from '{TOPIC}' and writing to InfluxDB bucket '{INFLUX_BUCKET}'...")
```

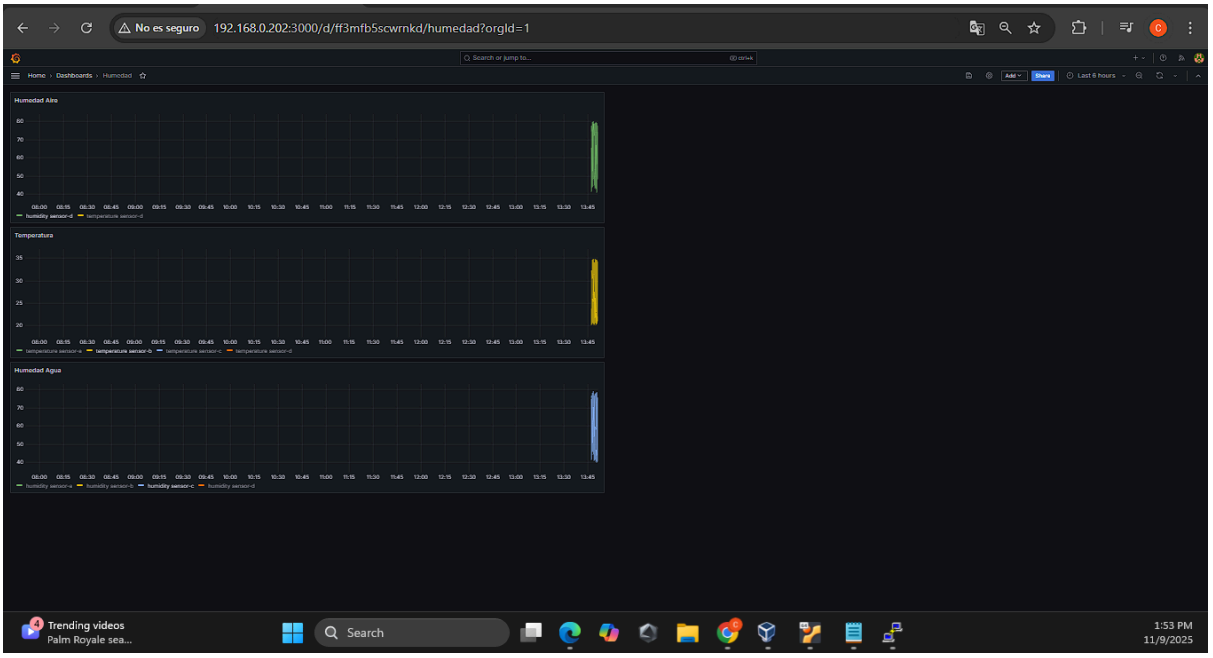
```
    for msg in consumer:
        data = msg.value
        try:
            point = to_point(data)
            write_api.write(bucket=INFLUX_BUCKET, org=INFLUX_ORG, record=point)
            print("Consumed and wrote:", data)
        except Exception as e:
            print("Write error:", e)

    client.close()

if __name__ == "__main__":
    main()
```

- **Ejecutar consumidor (desde tu host Linux):**
    - python3 consumer/consumer.py

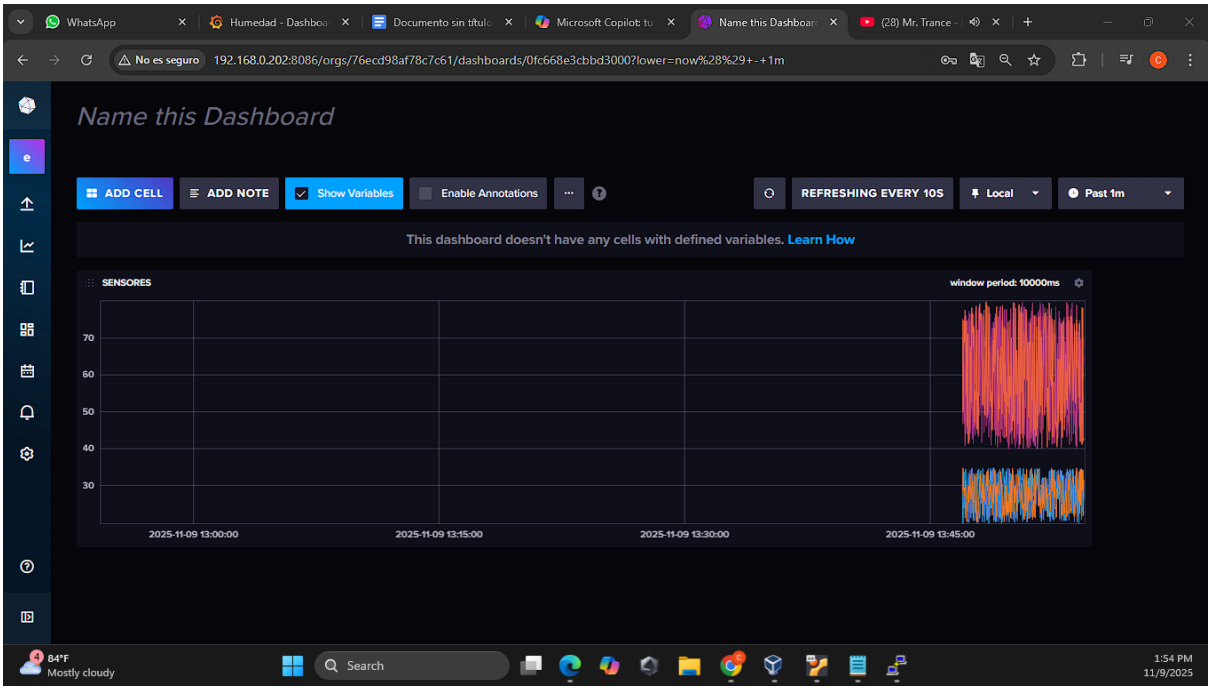


Configurar Grafana para visualizar

- **Entrar a Grafana:** http://localhost:3000 (admin / admin123)
- **Agregar datasource:**
    - **Tipo:** InfluxDB
    - **Query language:** Flux
    - **URL:** http://influxdb:8086
    - **Auth:** Token ecotec-token-123
    - **Organization:** ecotec
    - **Default Bucket:** iot_metrics



Crear dashboard y panel:



Query de temperatura:

```
from(bucket: "iot_metrics")
  |> range(start: -15m)
  |> filter(fn: (r) => r._measurement == "sensor_metrics")
  |> filter(fn: (r) => r._field == "temperature")
```

Query de humedad:

```
from(bucket: "iot_metrics")
  |> range(start: -15m)
  |> filter(fn: (r) => r._measurement == "sensor_metrics")
  |> filter(fn: (r) => r._field == "humidity")
```

Query de humedad aire:

```
from(bucket: "iot_metrics")
```

```
|> range(start: -15m)
|> filter(fn: (r) => r._measurement == "sensor_metrics")
|> filter(fn: (r) => r.device_id == "sensor-d")
```