Streaming Tweets con Kafka.

Contenido

[Recursos útiles 1](#_Toc204589337)

[Objetivo de la práctica 2](#_Toc204589338)

[Guion de partida 2](#_Toc204589339)

[Creación del entorno en Docker 3](#_Toc204589340)

[Prueba de Kafka 4](#_Toc204589341)

[Programación en Pyhton 4](#_Toc204589342)

[Bearer Twitter – X 4](#_Toc204589343)

[Productor 4](#_Toc204589344)

[Consumidor 5](#_Toc204589345)

[Comandos útiles para depurar 5](#_Toc204589346)

[Entregables de la práctica 6](#_Toc204589347)

# Recursos útiles

Recuerde el directorio de recursos compartidos disponible en:

<https://nascorformacion0-my.sharepoint.com/:f:/g/personal/juan_pinuela_docente_nascorformacion_com/EtGSBITnqFpJp6-XpG6REOQBFycP_G9zgcB89jxPymjV2Q?e=IBIEER>

En este directorio se encuentra la carpeta “Kafka” con los comandos más relevantes.

# Objetivo de la práctica

El En esta práctica, los alumnos levantarán un entorno completo de Apache Kafka usando Docker, crearán un entorno virtual de Python 3.10 y desarrollarán sobre él un pipeline básico de streaming que:

* Obtenga tweets en tiempo real usando la Twitter API.
* Los envíe a un topic Kafka.
* Lea esos tweets desde Kafka y los almacene en Elasticsearch.
* Visualice los datos en Kibana.

# Guion de partida

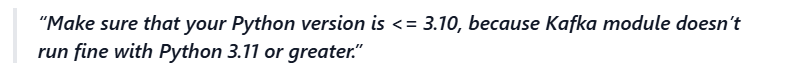
En esta práctica partimos del siguiente articulo:

<https://blog.praveshsudha.com/apache-kafka-project-real-time-twitter-streaming-with-python>

Que hace referencia al siguiente repositorio:

<https://github.com/Pravesh-Sudha/twitter-streams>

En el propio artículo se hace referencia al siguiente hecho:



Por eso es necesario crear un VENV en Python, en nuestro caso con VSCode e instalar la vesión 3.10. Podemos probar tanto esta versión como la correcta instalación de dependencias con el siguiente código:

**Creación entorno virtual e instalación de dependencias**

python3.10 -m venv kafka-env  
source kafka-env/bin/activate # (Windows: kafka-env\Scripts\activate)

pip install kafka-python tweepy requests

**check\_env.py**

import sys

import kafka

print("Python version:", sys.version)

print("Kafka version:", kafka.\_\_version\_\_)

# Creación del entorno en Docker

En el repositorio indicado se dan las siguientes instrucciones para arrancar los distintos contenedores necesarios para la práctica (una vez clonado o descargado el repositorio)

* **Zookeeper**

docker run -d --name zookeeper -p 2181:2181 zookeeper

* **Kafka**

docker run -d --name kafka \

  --link zookeeper \

  -p 9092:9092 \

  -e KAFKA\_ZOOKEEPER\_CONNECT=zookeeper:2181 \

  -e KAFKA\_ADVERTISED\_LISTENERS=PLAINTEXT://localhost:9092 \

  -e KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR=1 \

**confluentinc/cp-kafka**

* **ElasticSearch**

docker run -d --name elasticsearch -p 9200:9200 -e "discovery.type=single-node" elasticsearch:7.17.10

* **Kibana**

docker run -d --name kibana --link elasticsearch -p 5601:5601 kibana:7.17.10

Nosotros ya tenemos un stack ElasticSearch + Kibana de prácticas anteriores por lo que sólo será necesario crear un **docker-compose.yml con Zookeeper y Kafka.**

version: '3.8'

services:

zookeeper:

image: confluentinc/cp-zookeeper:7.5.0

container\_name: zookeeper

environment:

ZOOKEEPER\_CLIENT\_PORT: 2181

ZOOKEEPER\_TICK\_TIME: 2000

ports:

- "2181:2181"

kafka:

image: confluentinc/cp-kafka:7.5.0

container\_name: kafka

depends\_on:

- zookeeper

ports:

- "9092:9092"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://localhost:9092

KAFKA\_LISTENERS: PLAINTEXT://0.0.0.0:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

Una vez creado o descargado del directorio compartido el fichero Docker-compose.yml arrancamos con:

docker-compose up -d

# Prueba de Kafka

Probamos el funcionamiento de Kafka ejecutando el script proporcionado para crear un topic de nombre “twitter-stream” según el fichero:  
./create-kafka-topic.sh

Y validamos que existe con:  
docker exec -it kafka kafka-topics --list --bootstrap-server kafka:9092

# Programación en Pyhton

## Bearer Twitter – X

En producer.py, reemplaza:  
BEARER\_TOKEN = "tu\_token\_de\_twitter"

Por el momento puedes usar el del docente sabiendo que será regenerado al finalizar la sesión y deberás pedir uno nuevo:

#Twitter API credentials

BEARER\_TOKEN = "AAAAAAAAAAAAAAAAAAAAAC%2BHgwAAAAAAyp0eUiZuf1OIUF8EQtRZIJRPL4c%3D5fOLn5xS32JMVJssJBSEygBvPiGNseOHCBvZTmtxMya49jPgO2"

## Productor

1. Ejecutar el productor (recoge tweets y los envía a Kafka):  
   python producer.py
2. Ver mensajes en Kafka con un consumer manual:  
   docker exec -it kafka kafka-console-consumer --bootstrap-server kafka:9092 --topic twitter-stream --from-beginning

## Consumidor

1. Ejecutar el consumer (lee de Kafka y envía a Elasticsearch):  
   python consumer.py
2. Verificar en ElasticSearch con Postman:

<http://localhost:9200/twitter/_search>

1. Visualizar en Kibana:

<http://localhost:5601>

## Comandos útiles para depurar

* Listar topics:

docker exec -it kafka kafka-topics --list --bootstrap-server kafka:9092

* Ver mensajes en tiempo real:

docker exec -it kafka kafka-console-consumer --bootstrap-server kafka:9092 --topic twitter-stream --from-beginning

* Contar mensajes en el topic:

docker exec -it kafka kafka-run-class kafka.tools.GetOffsetShell --broker-list kafka:9092 --topic twitter-stream --time -1

(Devuelve el offset actual = nº de mensajes)

* Ver índices en Elasticsearch:

<http://localhost:9200/_cat/indices?v>

# Entregables de la práctica

Al finalizar la práctica, se deberá entregar:

* Evidencias de la configuración del entorno
* Captura de pantalla de la ejecución de docker ps mostrando los contenedores Kafka, Zookeeper, Kafdrop, Elasticsearch y Kibana corriendo.
* Captura del comando docker exec -it kafka kafka-topics --list --bootstrap-server kafka:9092 donde aparezca el topic twitter-stream.
* Evidencias del flujo de datos
* Pantallazo del producer.py en ejecución mostrando tweets enviados a Kafka.
* Pantallazo del consumer.py procesando mensajes desde Kafka y confirmando envío a Elasticsearch (líneas “Received tweet” y “Elasticsearch response”).
* Validación en los sistemas de destino
* Captura de pantalla de una consulta en Postman o curl a http://localhost:9200/twitter/\_search?pretty=true mostrando tweets indexados.
* Captura de Kibana (Discover o Dashboard) donde se vean tweets visualizados.