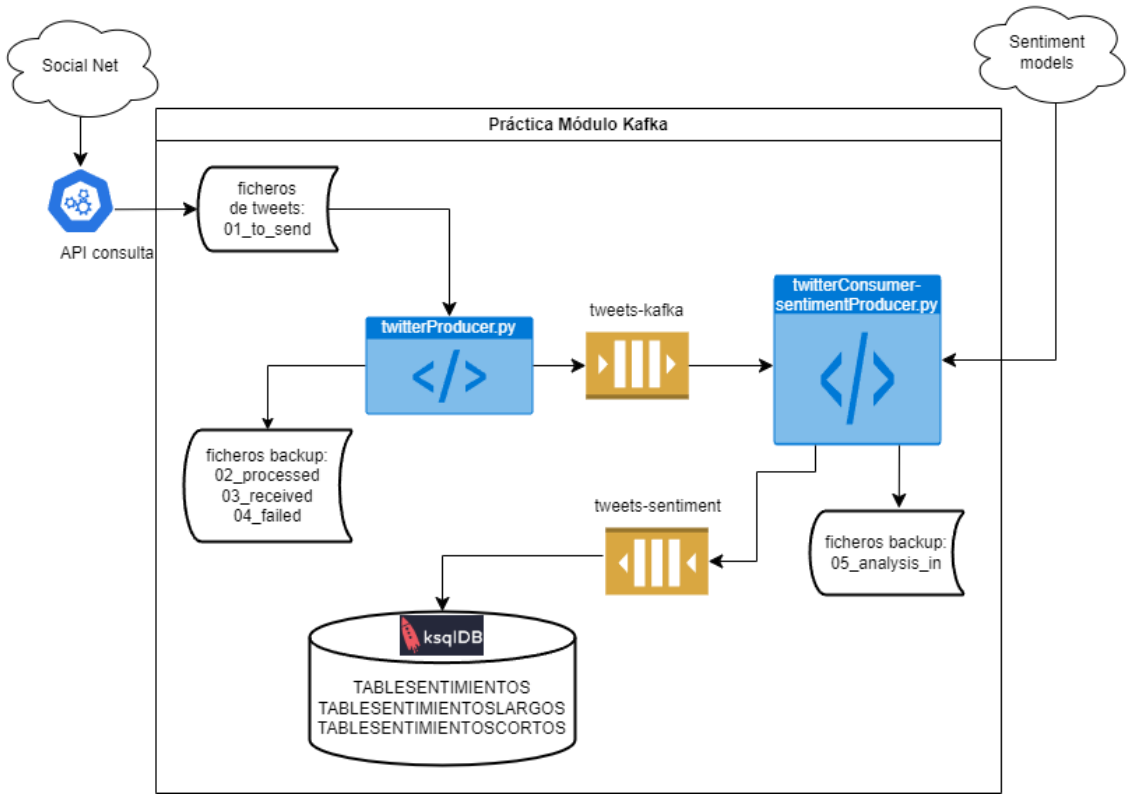
Documento de diseño

Diagrama de arquitectura de componentes



Descripción de componentes

Contenedores Docker

Contenedor	Función	Usado
zookeeper	Servidor de código abierto que permite la coordinación distribuida de diferentes procesos.	Sí
broker	Contenedor para levantar un nodo dentro de un clúster de Kafka. Cada bróker es responsable de almacenar datos en particiones y manejar la comunicación con los productores y consumidores.	Sí
schema-registry	Contenedor que proporciona un repositorio centralizado para administrar y validar esquemas para datos de mensajes.	Sí
connect	Contenedor para implementar y ejecutar Kafka Connect que permite conectar sistemas externos con un clúster Kafka.	Por dependencia
control-center	Confluent Control Center es una herramienta basada en web para administrar y monitorizar Apache Kafka.	Sí

	Proporciona una interfaz de usuario que permite visualizar de forma rápida el estado del clúster, observar y controlar mensajes, topics y esquemas, y desarrollar y ejecutar consultas ksqlDB.	
ksqldb-server	Contenedor que incluye el motor y la API Rest.	Por dependencia
ksqldb-cli	Contenedor que proporciona una interfaz por línea de comandos que permite a los usuarios interactuar con el motor de ksqlDB Server a través de la API Rest de ksqlDB Server.	Sí

Scripts Python

twitterProducer.py

Su función principal es leer los ficheros con tweets de una determinada ruta y enviarlos por la cola "tweets-kafka".

También es el encargado de almacenar en diferentes rutas (a modo de backup) la información que recibe y envía:

- 02_processed: Almacena los ficheros que ha leído de origen.
- 03_received: Almacena los mensajes que se han enviado.
- 04_failed: Almacena los registros de los ficheros de origen que han dado error de lectura.

twitterConsumer-sentimentProducer.py

Tiene dos funciones principales:

- consumir los mensajes procedentes de la cola "tweets-kafka".
- enviar a la cola "tweets-sentiment" el resultado de aplicar un modelo de sentimientos a los tweets.

También es el encargado de almacenar (a modo de backup) la información envía.

Colas Kafka

El propósito de una cola de mensajes es ayudar al flujo de mensajes de manera fiable, donde los productores y los receptores de los mensajes no necesitan interactuar con la cola de mensajes al mismo tiempo.

En este caso se ha optado por usar dos colas de Kafka:

- tweets-kafka: Se encarga de transmitir los mensajes de la red social desde unos ficheros de origen hasta los diferentes consumidores. En este caso, sólo existe un único consumidor, que será el encargado de analizar los mensajes para determinar el sentimiento de los mismos.
- tweets-sentiment: Se encarga de transmitir los mensajes etiquetados con un sentimiento (determinado por un modelo de análisis) hasta una base de datos ksqlDB.

ksqlDB

ksqlDB es una base de datos para streaming de eventos de código abierto, basada en Kafka Streams y que simplifica el procesamiento de flujos de datos.

Para ello, integra dos componentes del ecosistema Kafka (Kafka Streams y Kafka Connect) en un único sistema y ofrecer una interfaz SQL para interactuar con ambos componentes.