# Guia de operación detallada “practicakafka”

Contenido

[Guia de operación detallada “practicakafka” 1](#_Toc161901845)

[Introducción 2](#_Toc161901846)

[Requisitos del sistema 2](#_Toc161901847)

[Software utilizado 2](#_Toc161901848)

[Estructura de directorios del sistema 2](#_Toc161901849)

[Preparación del entorno – Instaladorsistema.sh 3](#_Toc161901850)

[Comprobación de que todo está preparado. 6](#_Toc161901851)

[Componentes del sistema 8](#_Toc161901852)

## Introducción

Con este documento se intenta explicar y guiar en la configuración del sistema “practicakafka”, sistema de análisis de sentimiento de comentarios de Twitter.

## Requisitos del sistema

- 16Gb de memoria RAM.

- 50Gb de Disco Duro.

- CPU i5 o superior.

## Software utilizado

- Windows 10 Pro o superior.

- Docker Desktop 4.27.1 o superior.

- Terminal 1.19.10573 o superior.

- Ubuntu 22.04.3 LTS

- Python3

- Librería de Python Textblob

## Estructura de directorios del sistema

+ practicakafka/

+ LEEME.txt

+ Instaladorsistema.sh

+ kafkaconnect/

+ tweets/

- connect-distributed.properties

- connect-file-source.json

- delete-file-connector.sh

- post-file-connector.sh

- twitter.txt

+ entorno/

- docker-compose.yml

+ doc/

- Manual de operación detallada.docx

- Manual de uso detallado.docx

- Diagrama arquitectura de componentes.pdf

- Diagrama de contexto.pdf

- Diagrama de contenedores.pdf

+ consumerproducerAPI/

- ConsumerAnalizador.py

- ConsumerProcesador.py

- ConsumerComprobador.py

+logs/ (Este directorio se creará con Instaladorsistema.sh si no existe).

- ConsumerAnalizador.log (Este fichero se creará con Instaladorsistema.sh si no existe).

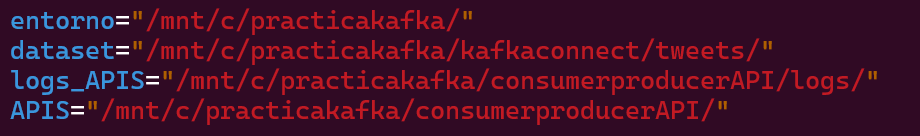
- ConsumerProcesador.log (Este fichero se creará con Instaladorsistema.sh si no existe).

## Preparación del entorno – Instaladorsistema.sh

Este script se encarga de arrancar, instalar y configurar el entorno "practicakafka", durante el proceso de instalación en la terminal que se ha ejecutado Instaldorsistema.sh va a ir mostrando información detallada de lo que va haciendo el instalador.

\***IMPORTANTE**\*

Hay que tener en cuenta que en este script hay tres variables de entorno que deben ser configuradas por el usuario para una correcta instalación del entorno. Estas variables son:

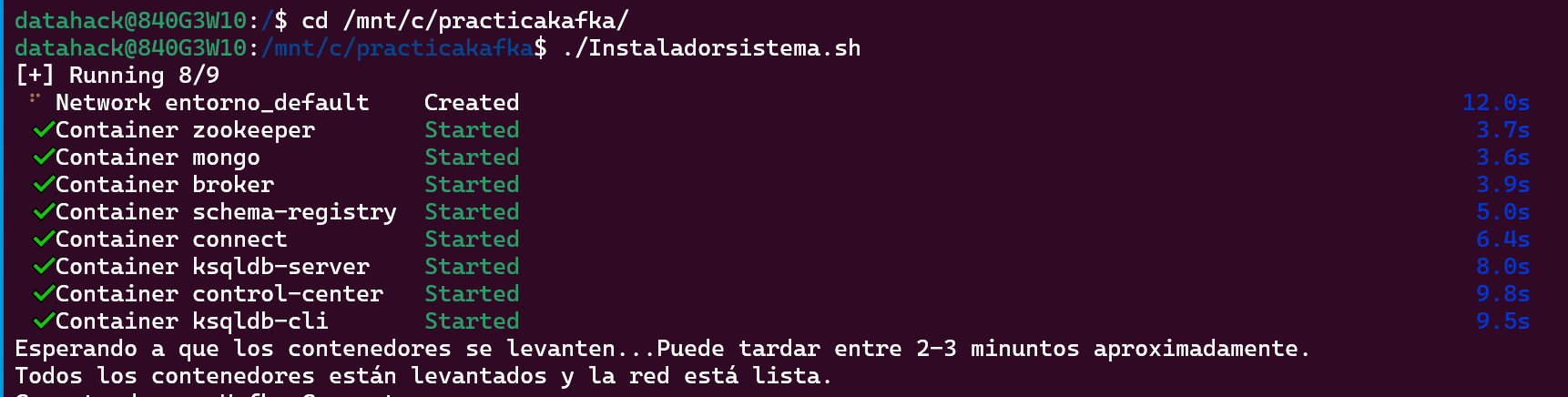


En la imagen se muestra la configuración por defecto ya que directorio practicakafka se copia en c:, al abrir un terminal WSL (Ubuntu) esa ruta equivale a /mnt/c/

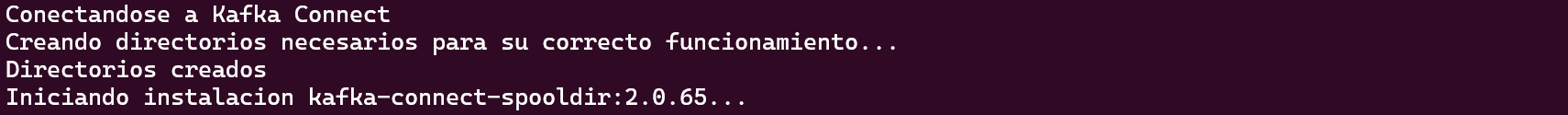
*c: = /mnt/c/*

Así que dependiendo de donde se copie el entrono "practicakafka" debe de cambiarse la ruta de las variables.

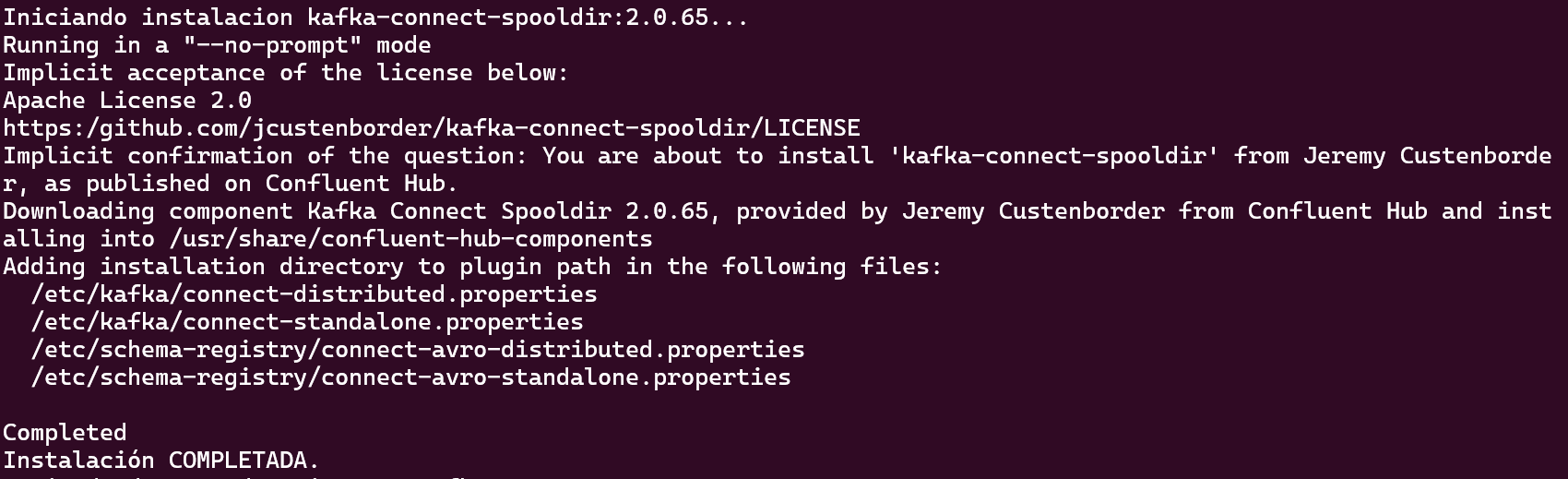
Cuando se ejecuta “Instaladorsistema.sh”, se pasa a detallar lo que muestra el instalador en pantalla:



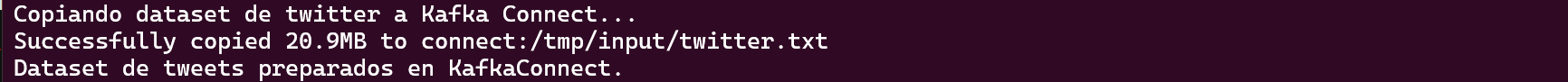
* El programa espera a que estén todos los contenedores levantados.



* Se conecta al contenedor de Kafka connect para configurar los directorios necesarios para su funcionamiento e instalar “confluent-hub install --no-prompt jcustenborder/kafka-connect-spooldir:2.0.65”.



* Copia desde kafkaconnet/tweets el dataset de tweets que vamos a analizar en el sistema.



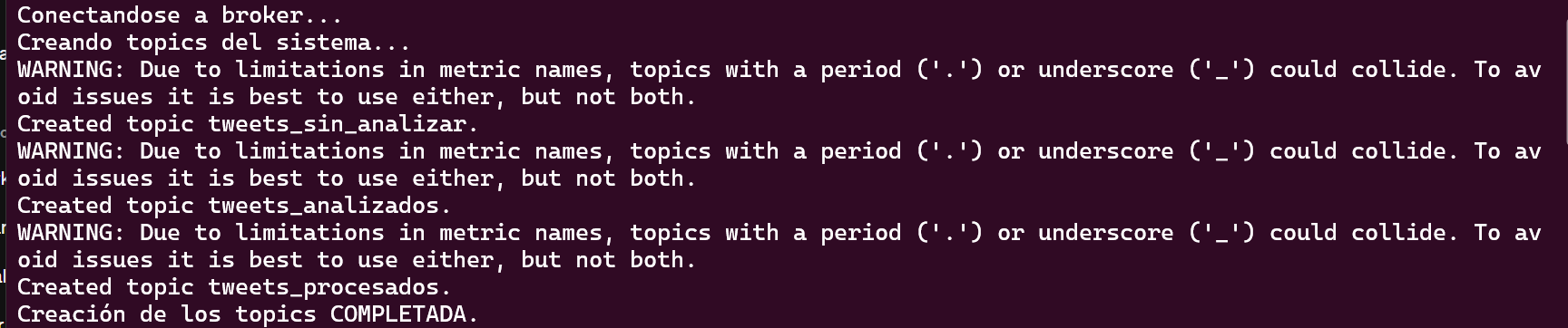
* Reinicia el contenedor de kakfa connect para que funcione el paquete instalado en los pasos previos.



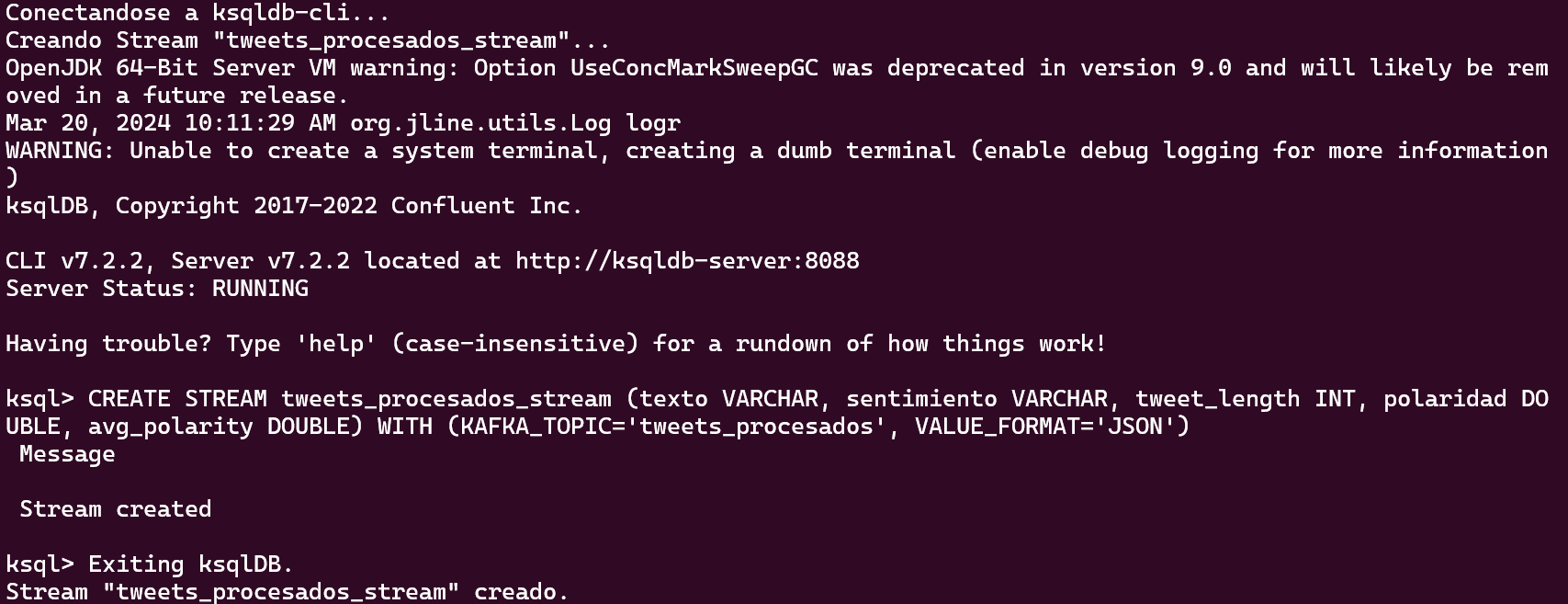
* Se comprueba que los procesos Python (Analizador.py y Procesador.py), no estén arrancados, porque como contienen los productores y consumidores se crearían los topics y luego a la hora de crear los topics puede llevar a conflicto.

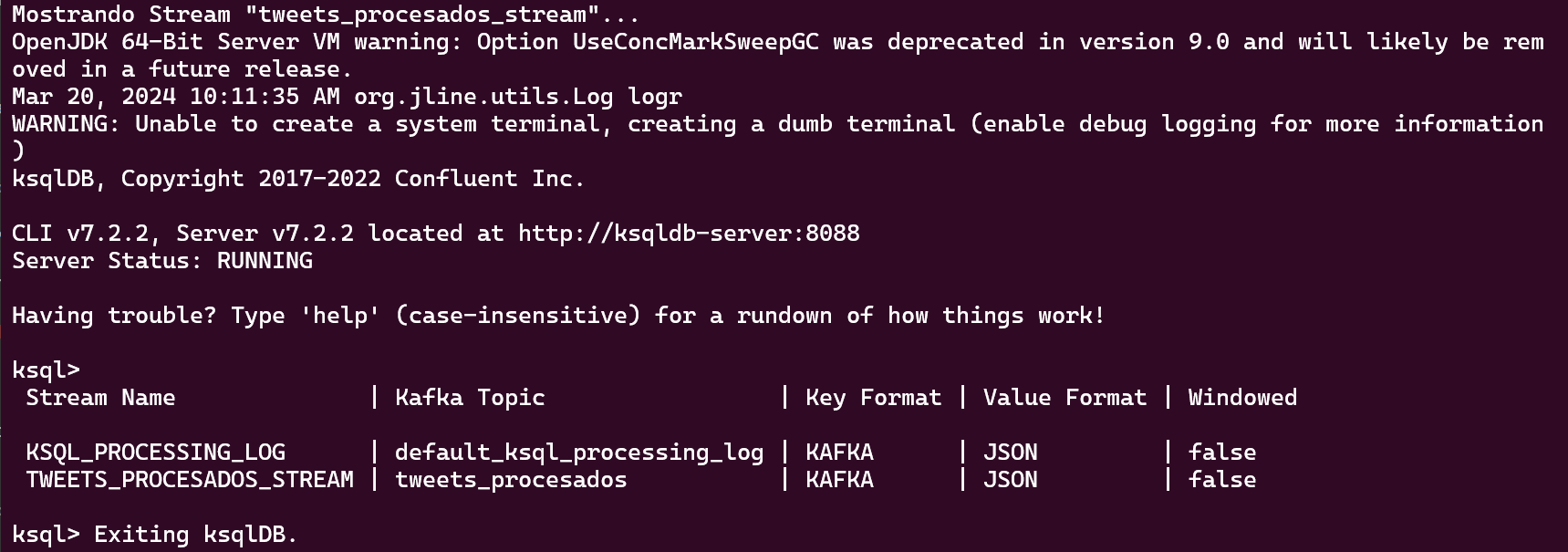


* Se conecta al broker y se crean los topics.

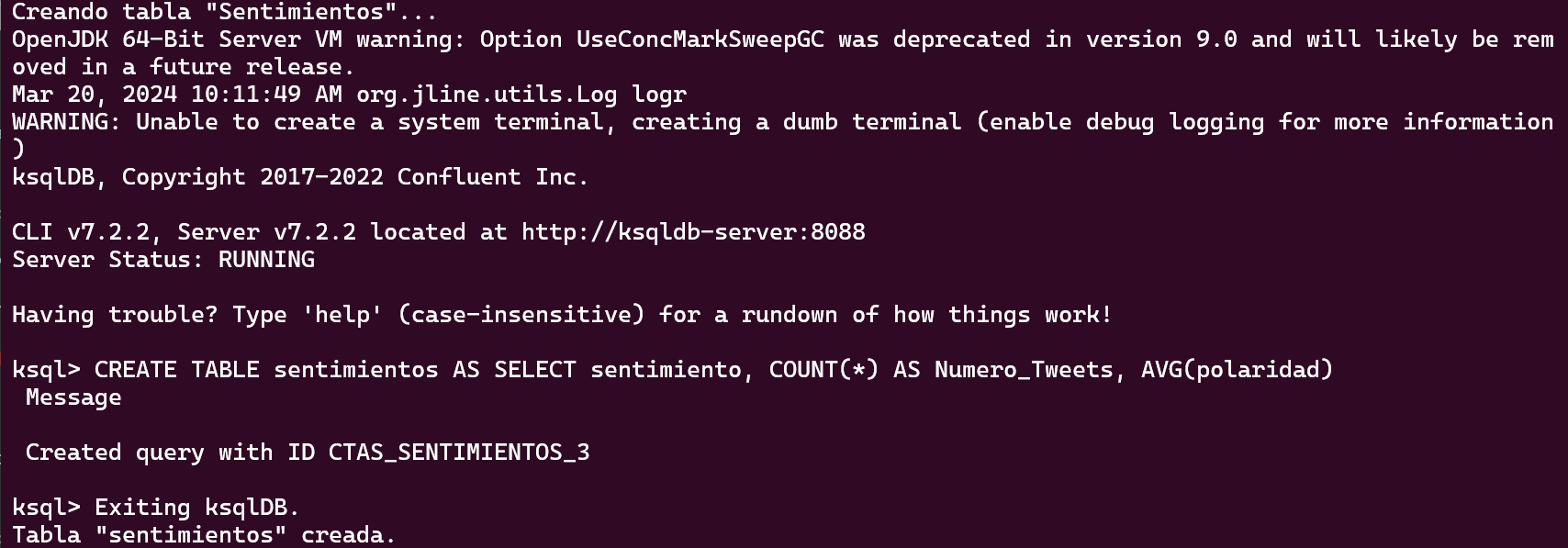


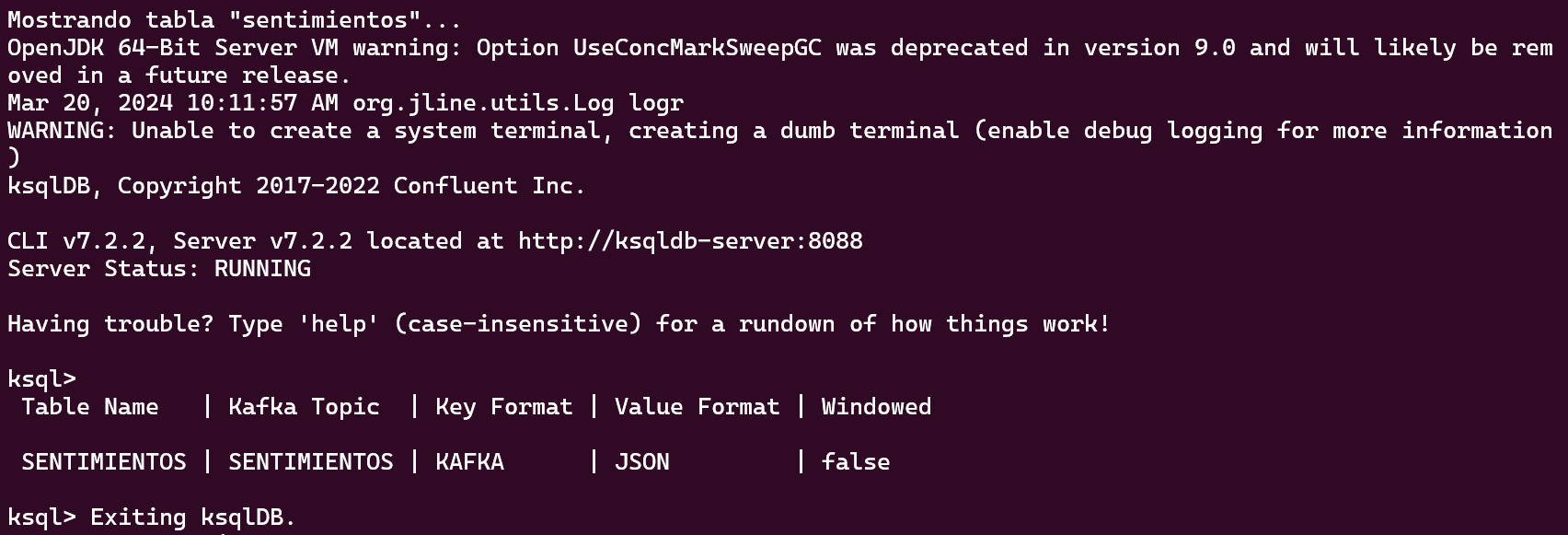
* Se conecta a ksqldb-cli y crear el stream y lo muestra.





* Desde el ksqldb crea la tabla y la muestra.





* Se comprueba que la librería de Python texblob está instalada en el host donde corren los procesos Python.

Si no está instalado, la instala.



Si está instalado lo muestra.



* Se comprueba que el directorio logs de los procesos Python estén creados.

Si no está creado lo crea.

Si está credo lo comunica.



* Arranca lo procesos Python (APIS).



Por último, indica el despliegue ha finalizado.

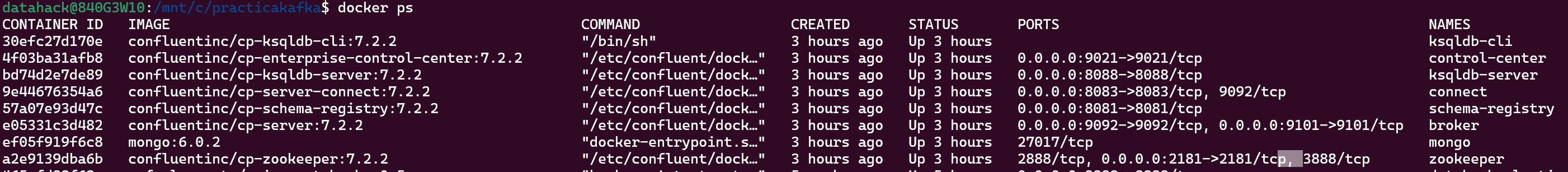


## Comprobación de que todo está preparado.

Para comprobar que toda la instalación y configuraciones ha salido correctamente se pueden comprobar siguiendo los pasos que se detallan:

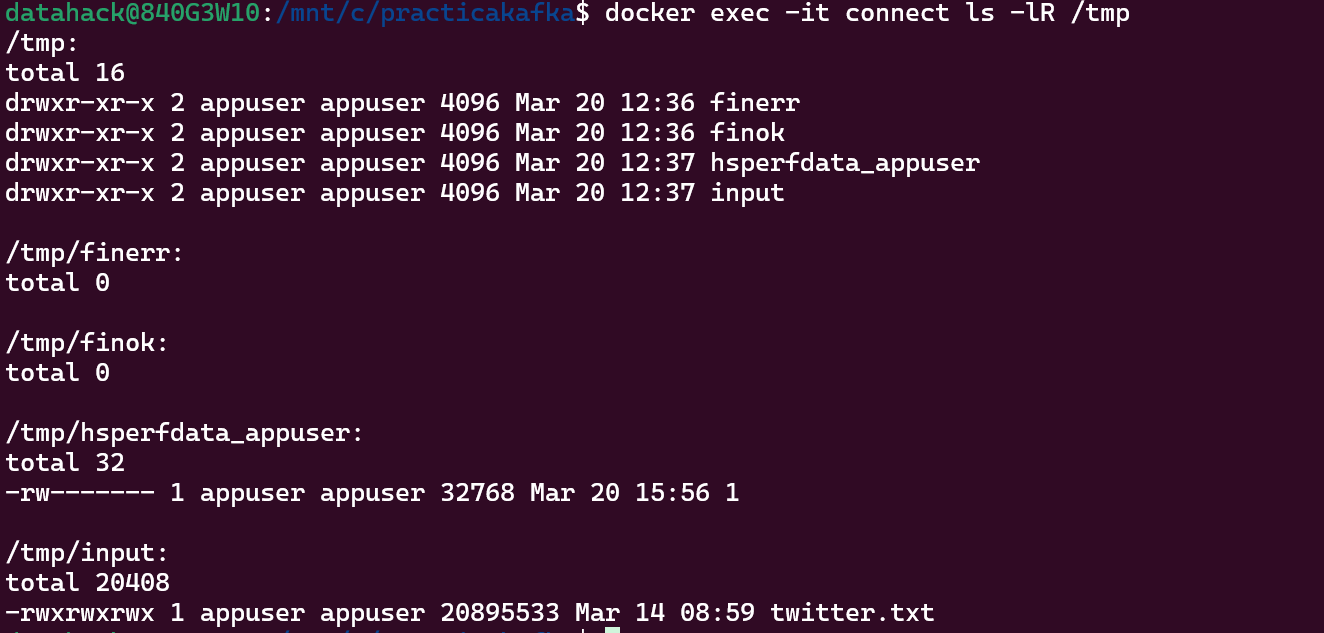
* Comprobar que los contenedores se han levantado, ejecutar:

docker ps



* Comprobar que los directorios y el dataset son correctos en el contenedor connect, ejecutar:

docker exec -it connect ls -lR /tmp/



* Comprobar que se ha reiniciado adecuadamente el contenedor connect, ejecutar:

docker logs connect –f

Comprobar que esta siguiente línea en el log: "INFO Kafka Connect started (org.apache.kafka.connect.runtime.Connect)"

Ctl+c

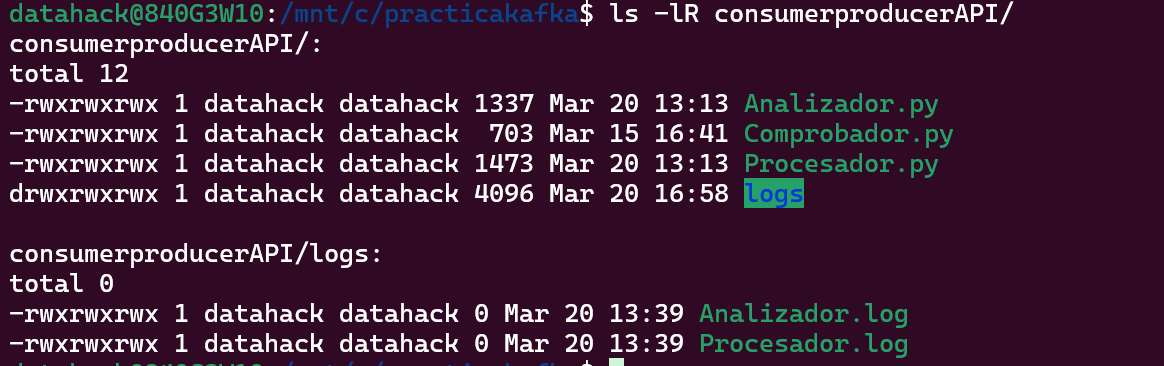
* Comprobar que los topics se ha creado correctamente, ejecutar:

docker exec -it broker kafka-topics --list --bootstrap-server localhost:9092 | grep tweets



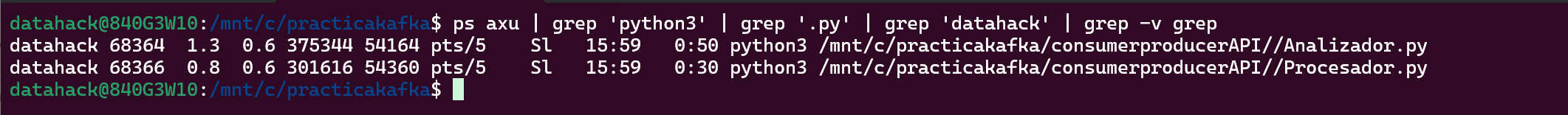
* Comprobar que se han creado el directorio logs, ejcutar:

ls -lR consumerproducerAPI/

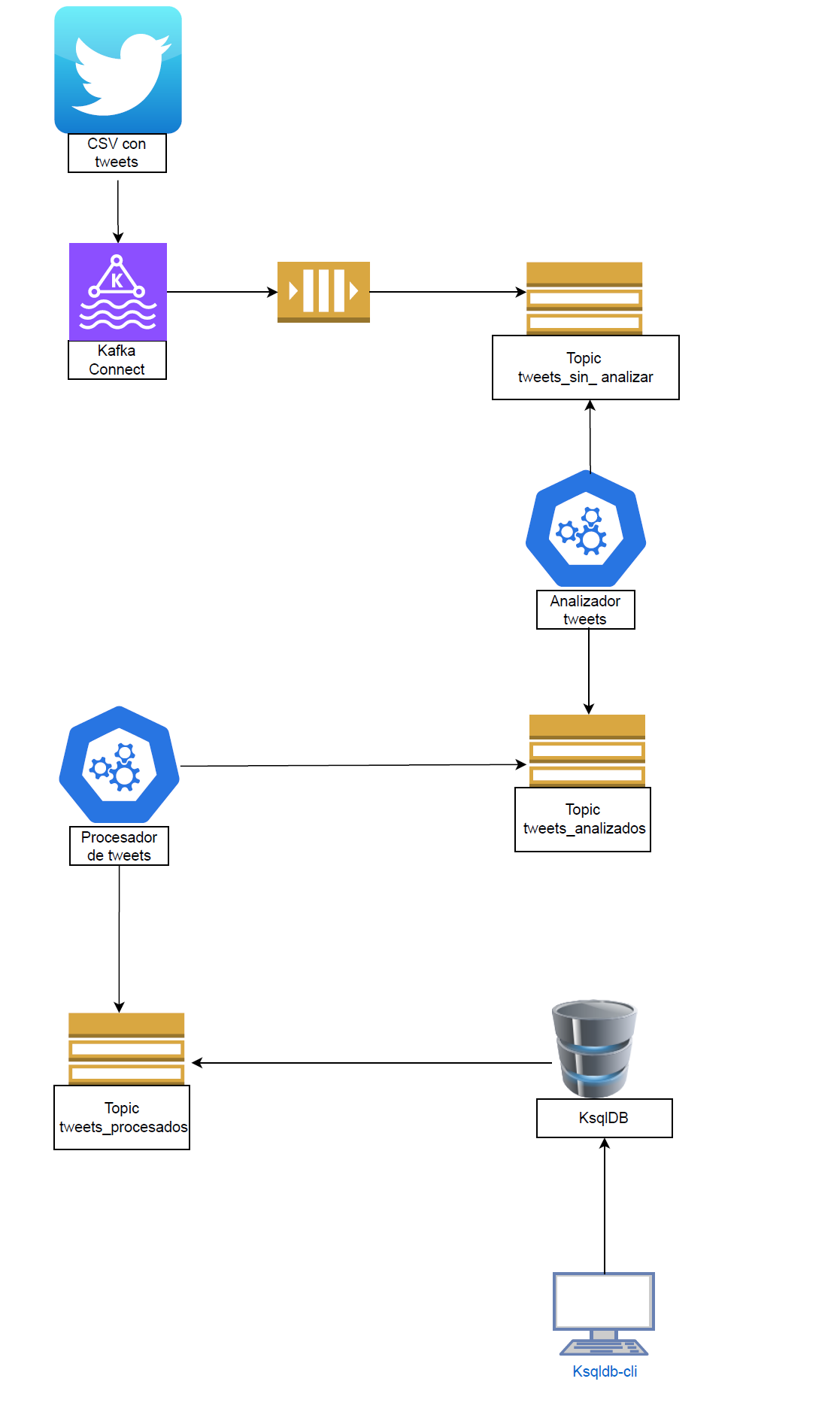


* Comprobar que han arrancado correctamente las aplicaciones API, ejecutar:

ps axu | grep 'python3' | grep '.py' | grep 'datahack' | grep -v grep



## Componentes del sistema



Kafka Connect

Es el contenedor proporcionado en el docker-compose.yml que se encarga de insertar el dataset twitter.txt con 190.000 tweets aproximadamente en el sistema para su posterior análisis y procesado.

Topics

Se usan tres topics en el sistema:

* Tweets\_sin\_analizar: Es donde produce los tweets con el formato .json Kafka Connect para realizar el análisis de sentimiento de cada comentario. Y también de donde los consume Analizador.py para su posterior análisis.
* Tweets\_analizados: Es donde produce Analizador.py los tweets ya analizados para realizar el procesado posterior. Y también de donde los consume Procesador.py para calcular la media de polaridad, cuenta el número de caracteres en cada comentario y añade una clave con un id único por cada tweet.
* Tweets\_procesados: Es donde produce los tweets ya procesados Procesador.py. Y También de donde los consume ksqlDB.

Procesos Python (APIS)

Tenemos dos APIS:

* Analizador.py:
  + Limpia cada tweet de caracteres que pueden generar problemas a la hora de analizarlos.
  + Tiene un consumer configurado que se alimenta del topic tweets\_sin\_analizar.
  + Tiene un producer configurado que deja los tweets analizados en el topic tweets\_analizados.
  + Analiza el sentimiento de cada tweet analizado en base a la librería Textblob.



* Procesador.py
  + Cuenta el número de caracteres por tweet procesado.
  + Tiene un consumer configurado que se alimenta del topic tweets\_analizados.
  + Tiene un producer configurado que deja los tweets analizados en el topic tweets\_procesados.
  + Calcula la polaridad y la media de cada tweet procesado.
  + Genera el mensaje final que se va a mandar al topic.
  + Añade una clave id a cada tweet para que tenga un identificador único.



KsqlDB

Es el contenedor proporcionado en el docker-compose.yml que a través de un stream y una tabla de agregación que se han creado previamente en el “instaladorsistema.sh” lee los tweets del topic tweets\_procesados y se pueden realizar búsquedas según se insertan los tweets a tiempo real con ksqldb-cli.