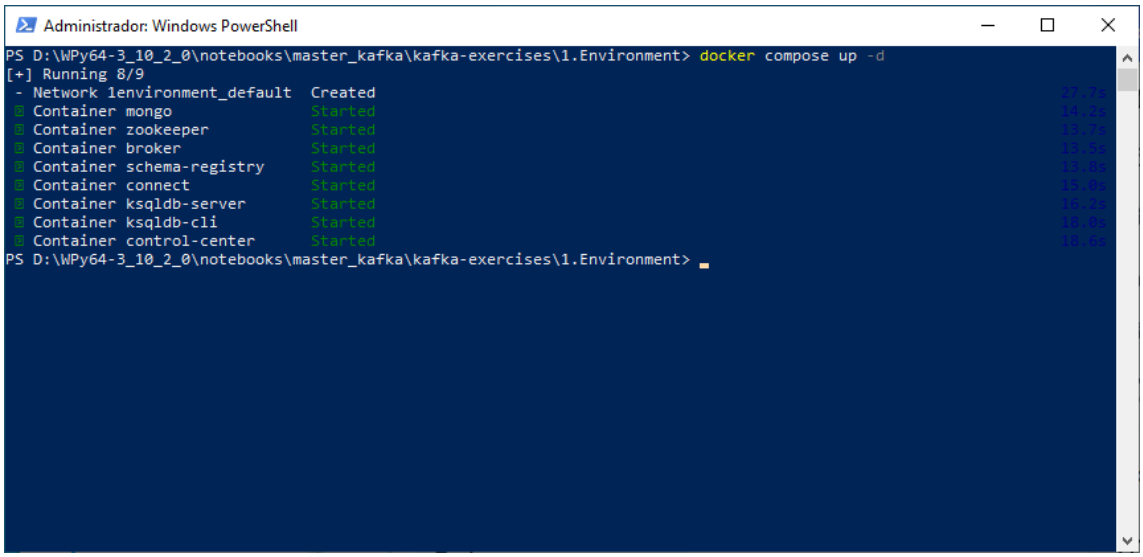
Manual de uso

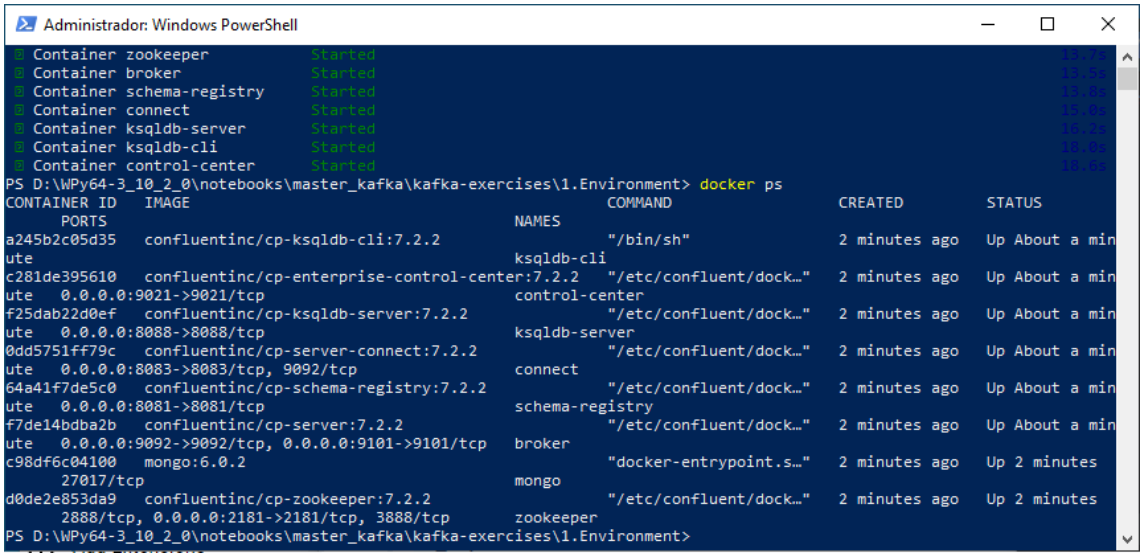
El proyecto se ha desarrollado en un sistema Windows, por lo que los comandos que se ejecutan durante el despliegue pueden variar ligeramente si se despliega en un sistema Ubuntu.

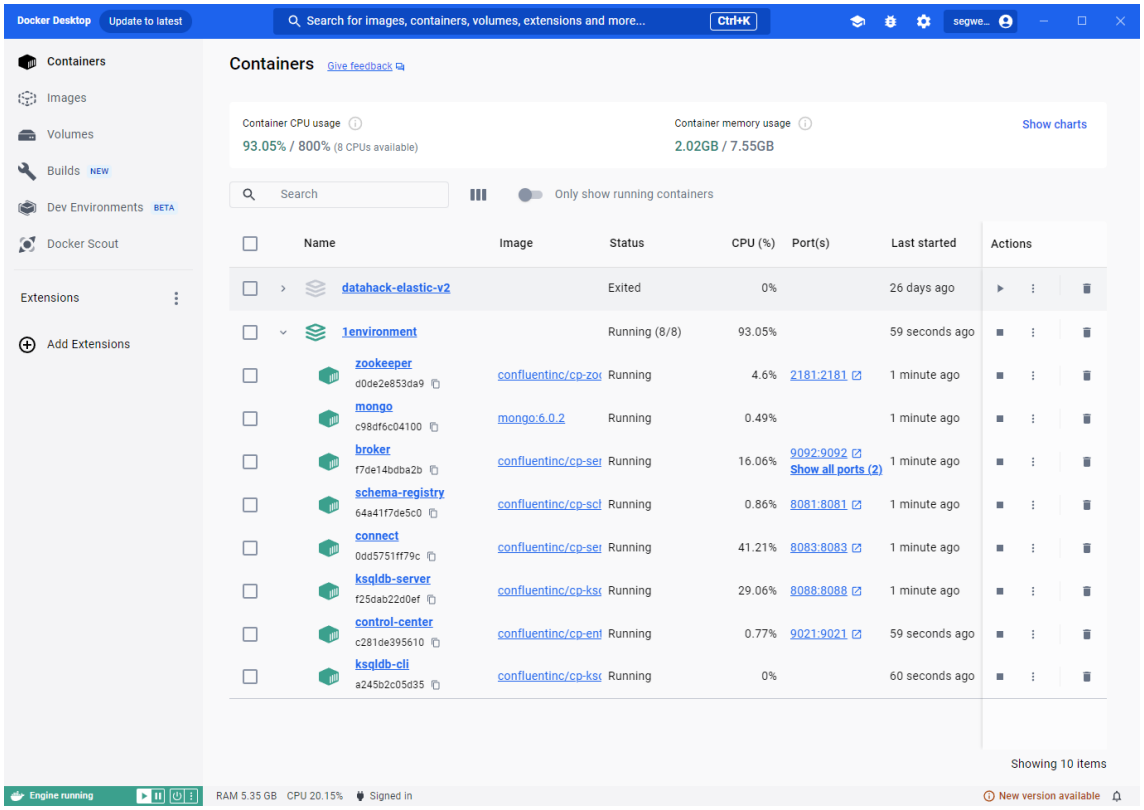
Levantar contenedores docker

1. Abrir una terminal donde se encuentre el docker-compose.yml para levantar el entorno y se ejecuta el comando **docker compose up –d**



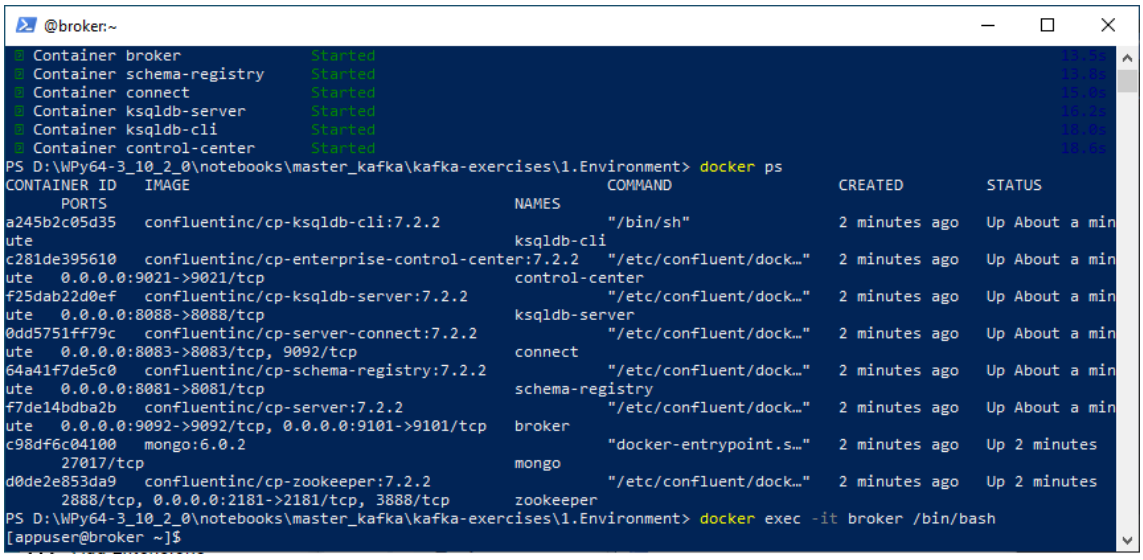
2. Para obtener el nombre y el puerto del bróker de Kafka dentro del entorno se ejecuta en consola el comando **docker ps.** En docker-desktop también permite obtener esta información:





Consola del bróker

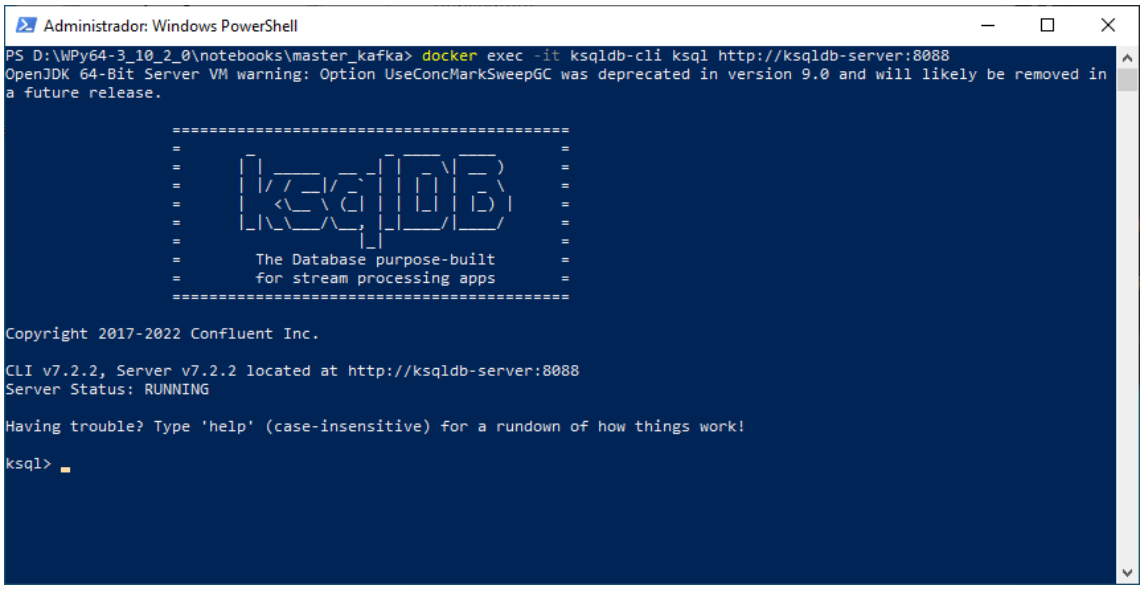
3. Habilitar una consola interactiva dentro del bróker mediante el comando **docker exec -it broker /bin/bash**



NOTA: Para ver la ubicación de los topics que se van creado se puede ejecutar el comando **ls -l /var/lib/kafka/data**

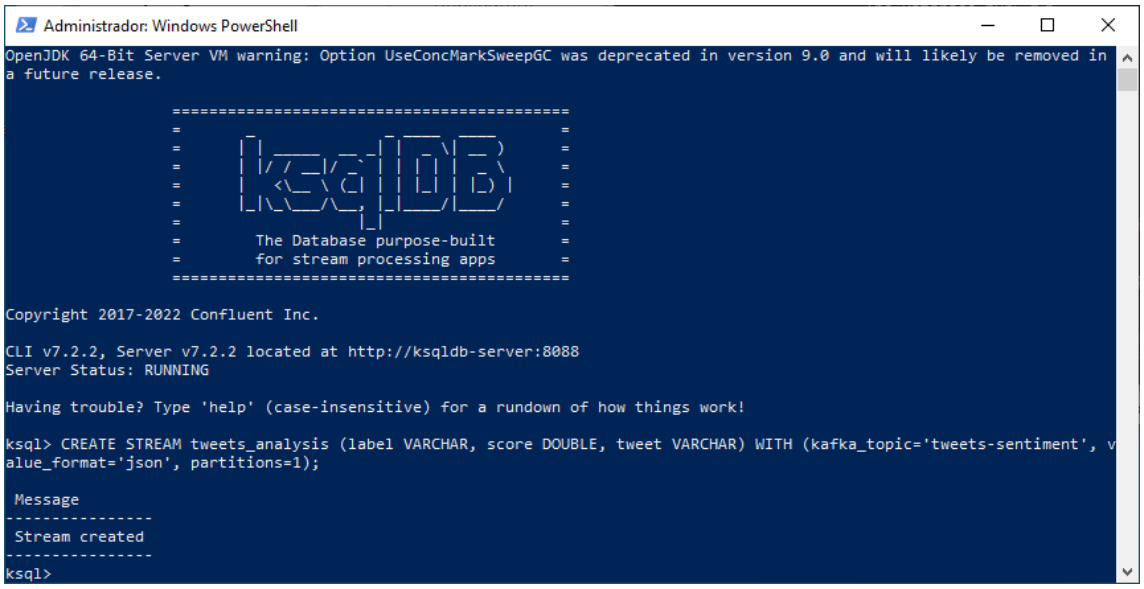
Consola de ksqlDB

4. Abrir otra consola interactiva para ksqlDB ejecutando el comando **docker exec -it ksqldb-cli ksql http://ksqldb-server:8088**



5. En la consola interactiva de ksqlDB se crea el stream por medio del cual se van a introducir los registros en ksqlDB. Para ello, se ejecuta el comando:

   CREATE STREAM tweets_analysis (label VARCHAR, score DOUBLE, tweet VARCHAR) WITH (kafka_topic='tweets-sentiment', value_format='json', partitions=1);

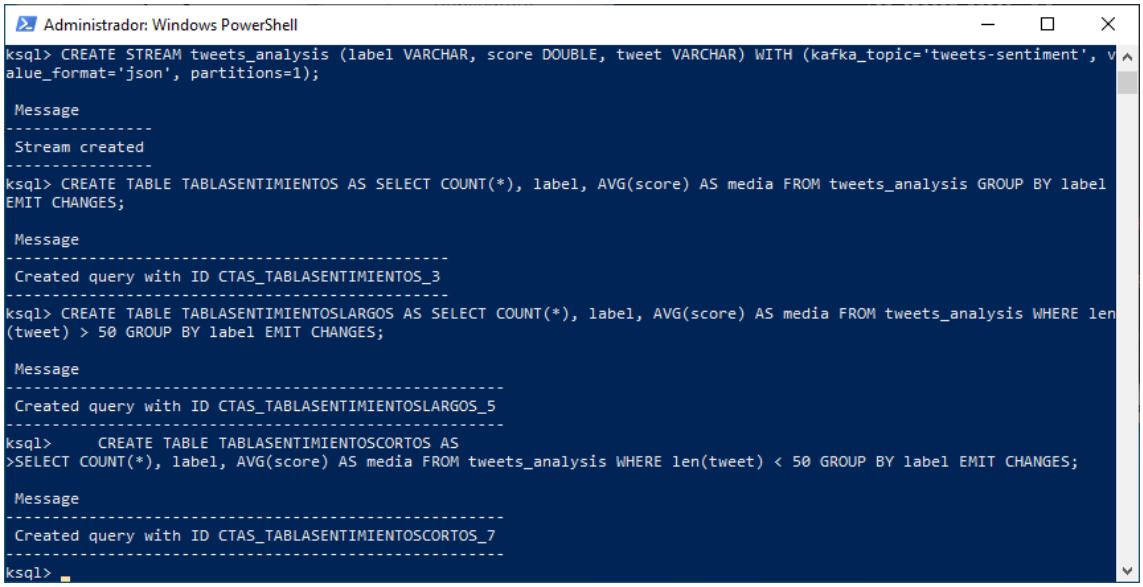


6. En la consola interactiva de ksqlDB se crea una tabla para contabilizar los registros positivos, negativos y neutros.
   CREATE TABLE TABLASENTIMIENTOS AS SELECT COUNT(*), label, AVG(score) AS media FROM tweets_analysis GROUP BY label EMIT CHANGES;

7. En la consola interactiva de ksqlDB se crean dos tablas más, una para mensajes largos y otras para mensajes cortos (50 caracteres). Los mensajes cortos pueden resultar ambiguos si no están en contexto. El número de caracteres se puede definir en el comando de creación de las tablas

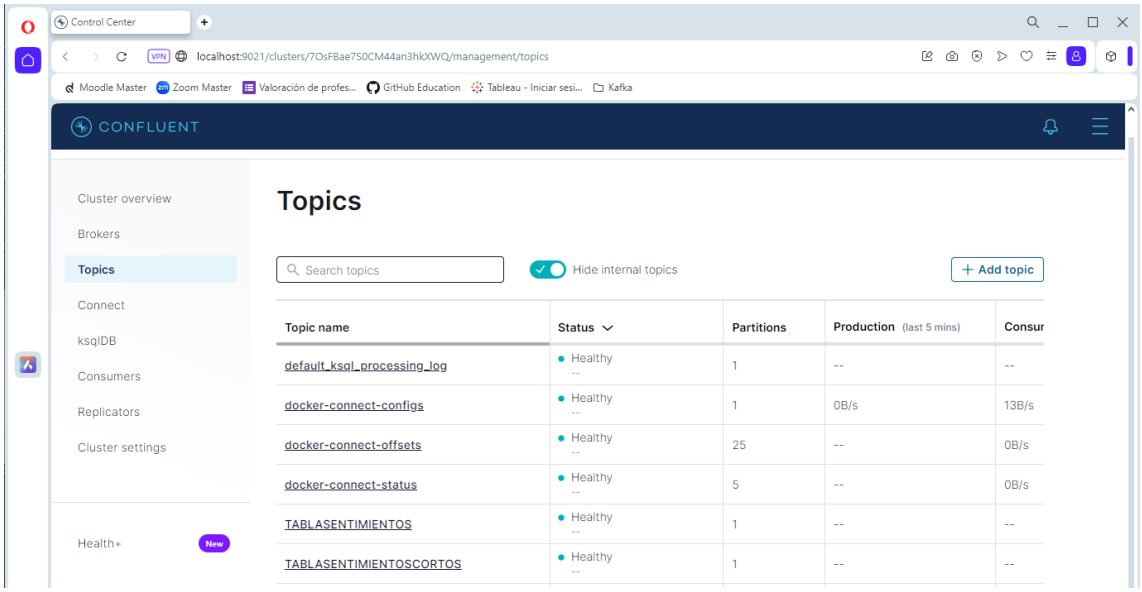
   CREATE TABLE TABLASENTIMIENTOSLARGOS AS SELECT COUNT(*), label, AVG(score) AS media FROM tweets_analysis WHERE len(tweet) > 50 GROUP BY label EMIT CHANGES;

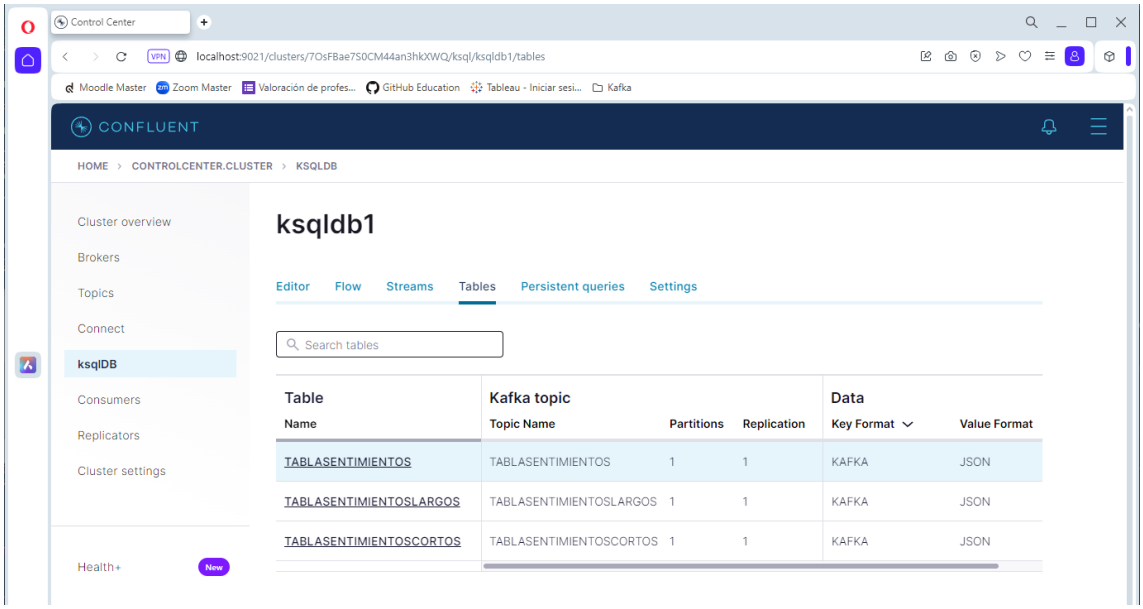
   CREATE TABLE TABLASENTIMIENTOSCORTOS AS SELECT COUNT(*), label, AVG(score) AS media FROM tweets_analysis WHERE len(tweet) < 50 GROUP BY label EMIT CHANGES;



Interfaz Control Center

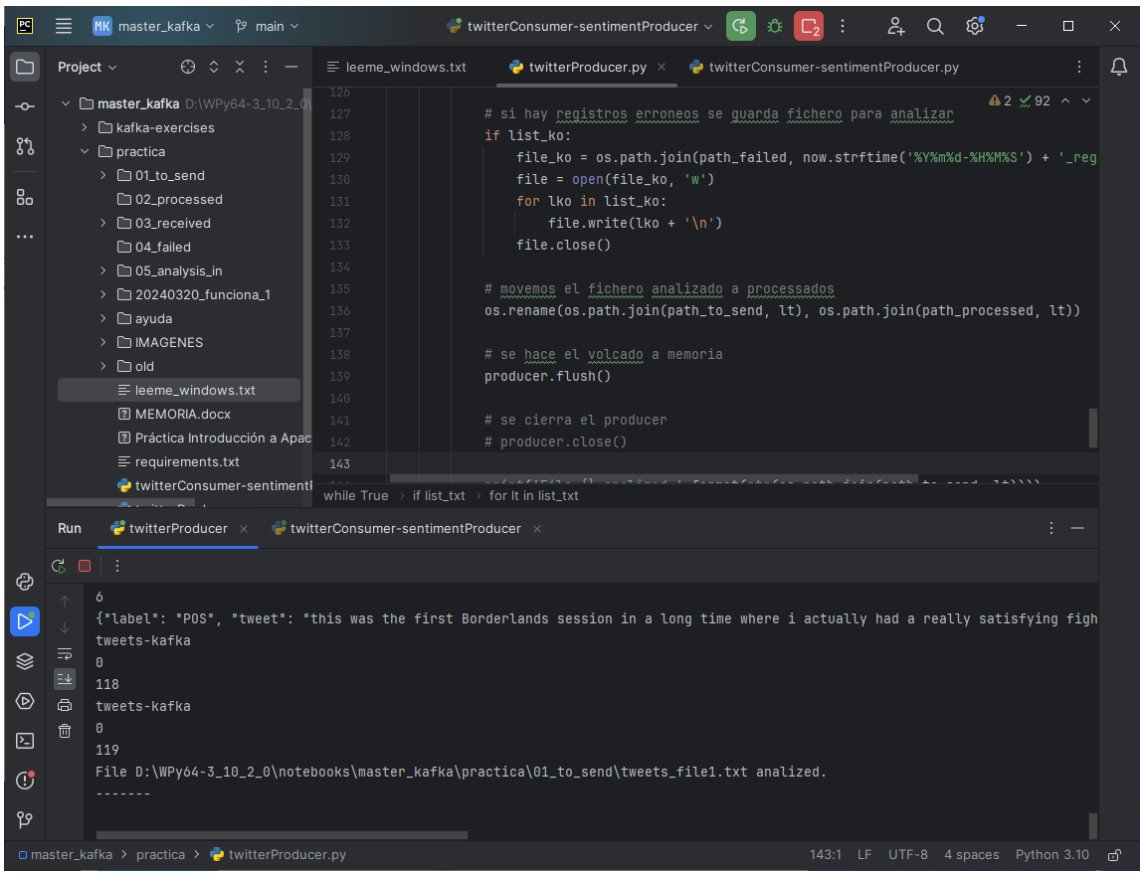
8. Se comprueba el estado de lo que se ha creado hasta el momento abriendo un navegador y escribiendo la siguiente ruta http://localhost:9021/





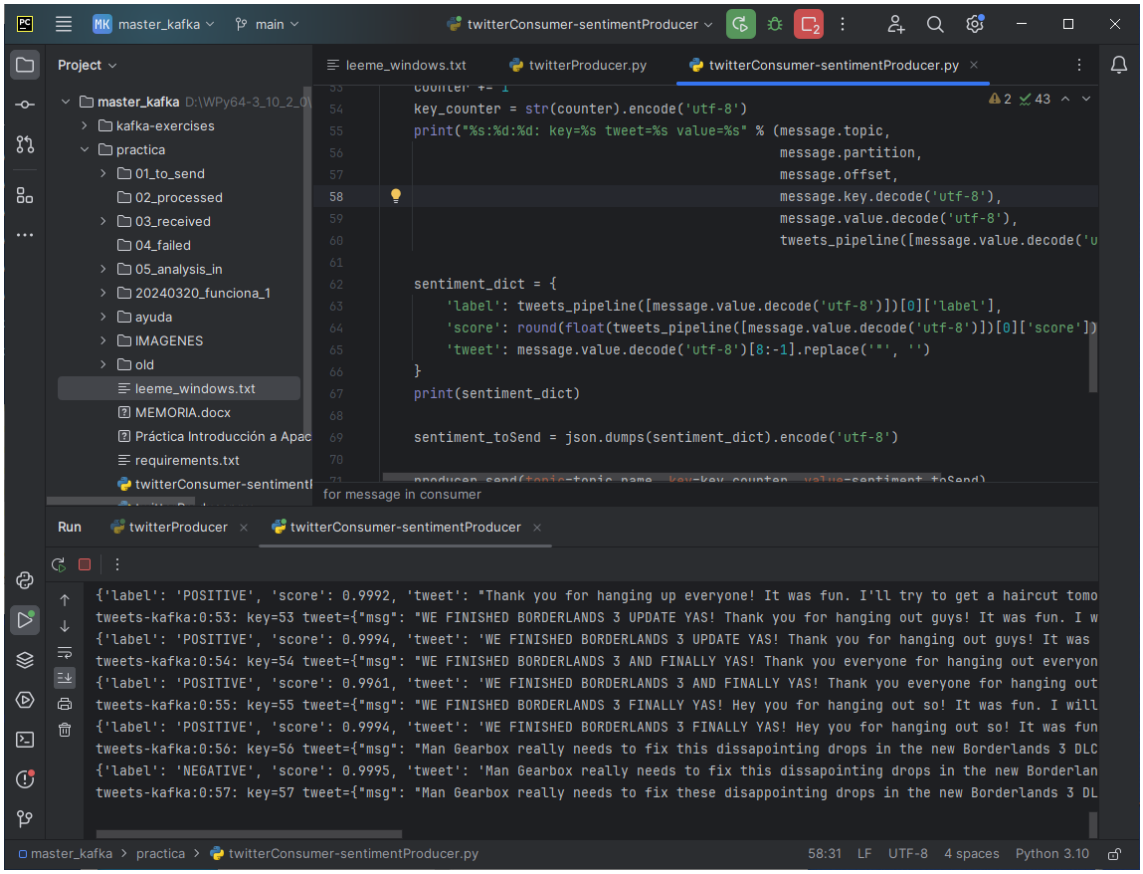
Producer y Consumer en Python

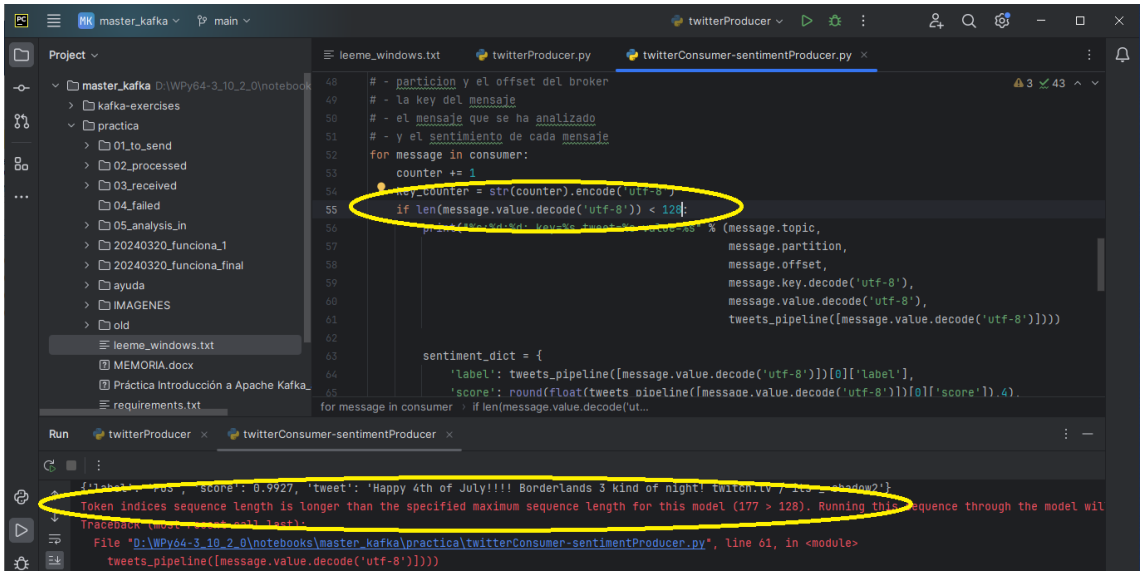
9. Se abre una consola en la ruta donde se encuentra el script Python del producer y se ejecuta el comando **python twitterProducer.py** (en Ubuntu **python3 twitterProducer.py**)



Se abre otra consola en la ruta donde se encuentra el script Python del consumer (que en este proyecto también será producer para ksqlDB) y se ejecuta el comando

python twitterConsumer-ksqldbProducer.py
(en Ubuntu **python3 twitterConsumer-ksqldbProducer.py**)





Para que se empiecen a procesar mensajes, hay que mover el fichero / los ficheros que se quiera desde la carpeta “ficheros_pruebas” a la carpeta “01_to_send”.

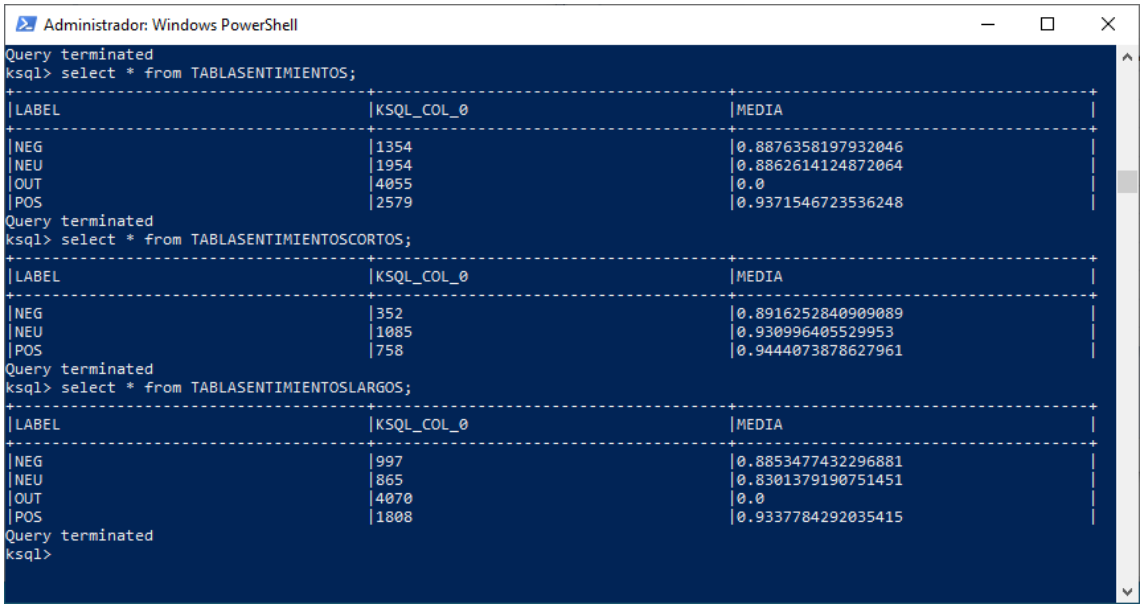
Visualización de datos en consola de ksqlDB

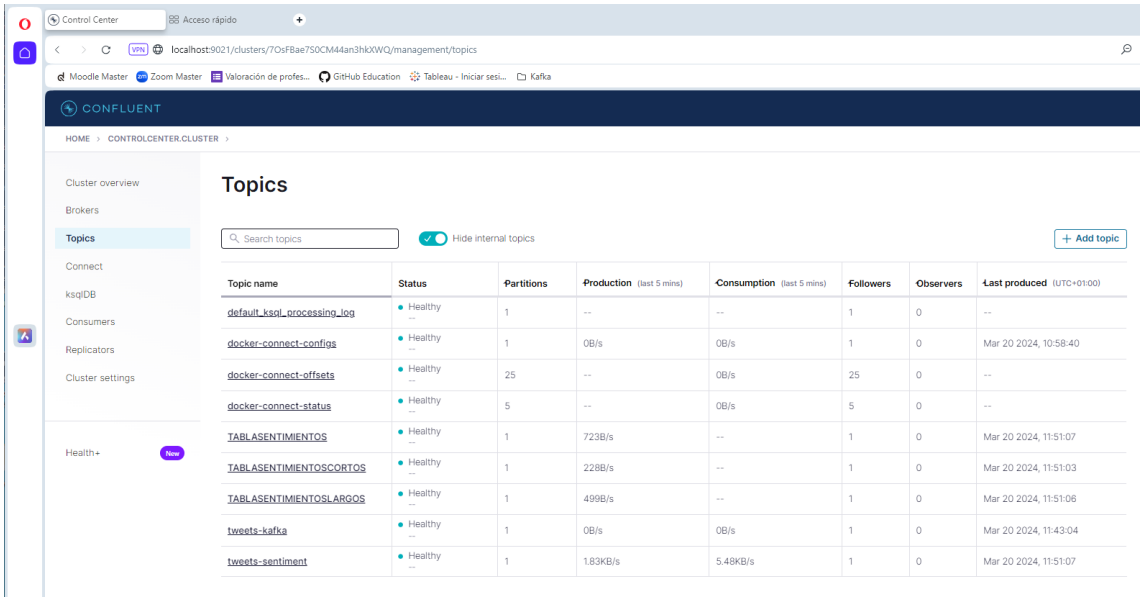
10. En la consola interactiva de ksqlDB se ejecutan las siguientes consultas para ir monitorizando la evolución de los sentimientos de los tweets analizados.

    select * from TABLASENTIMIENTOS;

    select * from TABLASENTIMIENTOSLARGOS;

    select * from TABLASENTIMIENTOSCORTOS;





Finalizar procesos

11. Para detener todos los procesos (para mantenimiento o configuración):
    - Ejecutar “Crtl+C” en las consolas donde se encuentran en ejecución los scripts Python.
    - Escribir “exit” en las consolas interactivas de ksqlDB y del bróker de Kafka.

Limpieza del despliegue

12. Finalmente, para limpiar el despliegue en la consola interactiva del bróker de Kafka ejecutar los siguientes comandos:
    - docker compose down
    - docker volumen prune (también se puede hacer desde docker-desktop).