Manual de operación

Prerequisitos

Docker

Sistema Ubntu

- Ir a la página https://docs.docker.com/engine/install/ubuntu/ y seguir las instrucciones de la sección "*Install using the apt repository*"
- Abrir consola en la máquina Ubuntu e instalar docker-compose:
  sudo apt-get install docker-compose-plugin

Sistema Windows

- Ir a la página https://docs.docker.com/desktop/install/windows-install/ y seguir las instrucciones de la sección "*Install Docker Desktop on Windows*"

Python

Sistema Ubuntu

Ir a la página https://phoenixnap.com/kb/how-to-install-python-3-ubuntu y seguir las instrucciones.

Sistema Windows

Existen varias opciones para instalar Python en entorno Windows:

- Básico: Ir a la página https://www.python.org/downloads/windows/ y seguir las instrucciones.
- Conda: Ir a la página https://conda.io/projects/conda/en/latest/user-guide/install/index.html
- WinPython: Ir a la página https://sourceforge.net/projects/winpython/files/ y seguir las instrucciones

Librerías

- kafka-python: https://pypi.org/project/kafka-python/ Obligatoria
- transformers: https://pypi.org/project/transformers/ Obligatoria
- emoji: https://pypi.org/project/emoji/ Recomendada
-

Para instalar estas librerías:

Ubuntu	Windows
pip3 install kafka-python	pip install kafka-python
pip3 install transformers	pip install transformers
pip3 install emoji	pip install emoji

Git

Ir a la página https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Instalación-de-Git y seguir las instrucciones.

Código fuente

Para descargar el código fuente se recomienda abrir una consola en la ruta donde se quiere desplegar la solución y ejecutar:

git clone https://github.com/segweba/practica_kafka.git

docker-compose.yml

Fichero para el despliegue de los contenedores Docker. Este es que se ha facilitado en el módulo y se ha decidido no modificarlo, aunque hay contenedores que no se han usado.

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 broker 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. 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.	Sí
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
mongodb	Contenedor de base de datos que proporciona alta disponibilidad y fácil escalabilidad.	No

twitterProducer.cfg

Fichero que contiene los parámetros de configuración del script twitterProducer.py.

```
{
    "version": "0.1",
    "server": "127.0.0.1:9092",
    "topic_name": "tweets-kafka",
    "folders": ["01_to_send","02_processed","03_received","04_failed"],
    "time_inter_files": "10",
    "time_standby": "5"
}
```

El parámetro “time_standby” establece cada cuanto tiempo (en segundos) se chequea el contenido de la carpeta “01_to_send”.

El parámetro “time_inter_files” establece el intervalo de tiempo (en segundos) entre lecturas de ficheros.

twitterProducer.py

Su función principal es leer los ficheros con tweets de una ruta y enviarlos a una cola de Kafka. Al ejecutarse, éste monitoriza el contenido de la carpeta de entrada (“01_to_send”). En el momento en el que detecta un fichero lo procesa.

twitterConsumer-sentimentProducer.cfg

Fichero con los parámetros de configuración de twitterConsumer-sentimentProducer.py.

```
{
    "version": "0.1",
    "model_0": "sentiment-analysis",
    "model_1": "finiteautomata/bertweet-base-sentiment-analysis",
    "model_2": "",
    "in_use": "1",
    "size_limit": "128",
    "topic_consumer": "tweets-kafka",
    "group_name": "sentiment-group",
    "server": "127.0.0.1:9092",
    "offset_type": "earliest",
    "topic_producer": "tweets-sentiment",
    "folders": ["05_analysis_in"],
    "regs_in_file": "100"
}
```

Los parámetros “model_X” incluyen diferentes tipos de modelos de análisis de sentimientos.

Mediante el parámetro “in_use” se identifica el modelo que se va a usar.

El parámetro “size_limit” establece la longitud máxima del token soportada por el modelo de sentimientos.

El parámetro “regs_in_file” establece el número de registros que se van a incluir en cada fichero de backup.

El parámetro “folders” incluye el nombre de la carpeta donde se guardan los ficheros de backup.

twitterConsumer-sentimentProducer.py

Script que aplica el modelo de sentimientos a los tweets y envía el resultado por medio de otra cola Kafka a ksqlDB. Al ejecutarse, éste se suscribe al topic definido en el parámetro “topic_consumer” y aplica el análisis de sentimientos a los mensajes recibidos. El resultado de este análisis se envía a ksqlDB por medio del topic “topic_producer”.

ficheros_pruebas

La carpeta incluye algunos ficheros usados para probar la solución. Estos ficheros habrá que moverlos a la carpeta “01_to_send” para simular la entrada de información procedente de la API de la red social.

requirements.txt

Este fichero no es necesario para el despliegue, pero se incluye para informar del entorno Python sobre el cual se ha desarrollado este proyecto.