Productor de Tweets de fichero

Con los cambios sobre la API de Twitter, los nuevos registros necesitarán una cuenta de pago para consumir tweets de la versión 2 de la API

Por esta razón, si este es tu caso, el curso ofrece una alternativa que consiste en usar el fichero tweets.txt. Este fichero contiene más de 10000 tweets extraídos directamente de la API de twitter y volcados en un fichero de texto plano con la misma estructura.

Nos permitirá hacer pruebas como si fuera la fuente original.

Para ello, vamos a hacer algunas modificaciones en nuestro productor de Tweets para que use este fichero.

Implementación del productor

A continuación tienes el código completo para la clase ProductorTweetsFichero:

```
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.io.*;
import java.util.Properties;

public class ProductorTweetsFichero {

   public final static String TOPIC_NAME = "rawtweets";
   public static ObjectMapper objectMapper = new ObjectMapper();

   public static void main (String[] args){

       Properties props = new Properties();
       props.put("acks", "1");
       props.put("retries", 3);
       props.put("batch.size", 16384);
       props.put("buffer.memory", 33554432);
       props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
       props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
       props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");

       final KafkaProducer<String, String> prod = new KafkaProducer<>(props);

       try (BufferedReader br = new BufferedReader(new FileReader("tweets.txt"))) {
           String line;
           while ((line = br.readLine()) != null) {
               JsonNode root;
               try {
                   root = objectMapper.readTree(line);
                   JsonNode hashtagsNode = root.path("entities").path("hashtags");
                   if (!hashtagsNode.toString().equals("")) {
                       String value = root.toString();
```

```
                        String lang = root.path("lang").toString();
//                        System.out.println(value);
                        prod.send(new ProducerRecord<>(ProductorTweetsFichero.TOPIC_NAME, lang, value));
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Como ves, tenemos que definir el nombre del topic, en mi caso voy a usar *rawtweets*.

```
public final static String TOPIC_NAME = "rawtweets";
```

El siguiente paso consiste en configurar las propiedades del productor. Aquí le indicamos los valores del ACK, reintentos, tamaños de batch y memoria, los servidores de bootstrap y los serializadores. En este caso usaremos de tipo string para almacenar el contenido del tweet en formato JSON.

```
      Properties props = new Properties();
       props.put("acks", "1");
       props.put("retries", 3);
       props.put("batch.size", 16384);
       props.put("buffer.memory", 33554432);
       props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
       props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
       props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
```

Después, inicializamos el productor de Kafka en la variable llamada *prod*.

```
final KafkaProducer<String, String> prod = new KafkaProducer<>(props);
```

Ahora, debemos leer nuestro fichero tweets.txt. Para ello, indicamos a java la ubicación en donde se encuentra el fichero.

En nuestro caso, queremos que el productor escriba en Kafka el contenido de los tweets que contengan al menos un hashtag. También, usaremos el idioma para agrupar más adelante los hashtags en función del idioma y del número de ocurrencias.

Para el envío a kafka, parsearemos la información de cada línea, que está en formato JSON.

Por último, debemos escribir el bloque de código que se va a encargar de escribir el tweet en Kafka.

Antes de escribir en Kafka, comprobaremos que el tweet contiene algún hashtag. Si no es así, descartaremos este registro y pasaremos al siguiente.

En el caso de que contenga algún hashtag, procederemos a escribir mediante el productor de Kafka un nuevo registro. Aquí, a la función *send*, le indicamos un nuevo objeto de tipo producer record con tres argumentos: el nombre del topic, la clave, que en este caso va a ser el idioma y por último el texto del tweet.

Prueba del productor

Si queremos comprobar que funciona correctamente, podemos comentar la línea que se encarga de enviar la escritura a Kafka, ya que no tenemos levantado el clúster, y añadir en su lugar una escritura por consola:

```
System.out.println(value);
```

Esta línea nos imprimirá cada tweet que enviaríamos a Kafka.

Lo ejecutamos dándole a run -> *ProductorTweetsFichero*. Recordad el argumento con el token de twitter

Vemos que en consola nos aparece una serie de tweets.

Con esto, ya tendríamos implementado el primer componente que producirá los tweets a kafka.