

Practica de Kafka

Requerimientos

- Docker
- Postman

Configuración de Kafka

Dentro del directorio de Docker lanzamos lo siguientes comandos:

```
# Inicializamos los contenedores
docker-compose up

# Entramos al broker de kafka
docker exec -it broker bash

# Creamos los topics
kafka-topics --bootstrap-server localhost:29092 --topic tweets --create --
partitions 3 --replication-factor 1
kafka-topics --bootstrap-server localhost:29092 --topic tweets-hashtags-lang
--create --partitions 3 --replication-factor 1
kafka-topics --bootstrap-server localhost:29092 --topic tweets-trending-
topics --create --partitions 3 --replication-factor 1
```

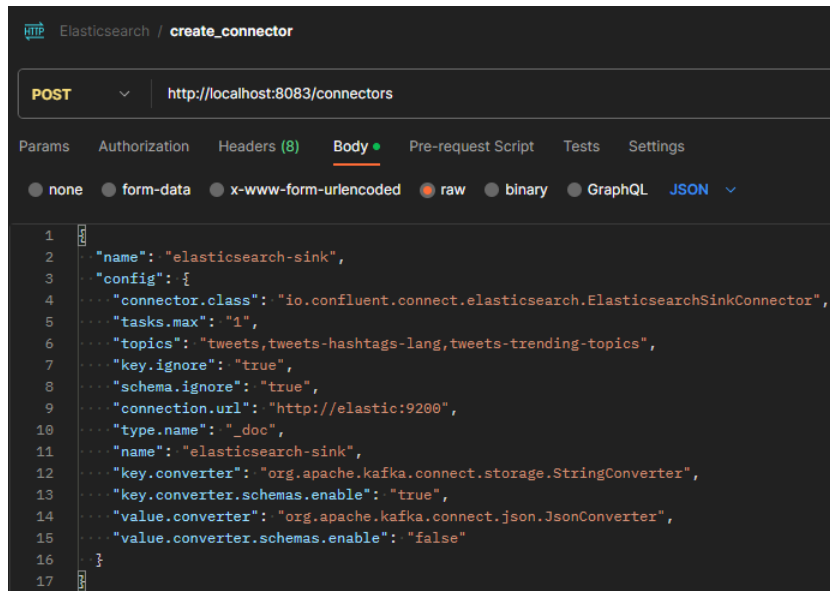
Lista de contenedores y puertos:

```
Kafka:          http://localhost:29092
Kafka Connect:  http://localhost:8083
ElasticSearch:  http://localhost:9200
Kibana:         http://localhost:5601
ksqldb-server:  http://localhost:8088
zookeeper:      http://localhost:2181
```

Configuración de Elasticsearch

Elasticsearch es la base de datos NoSQL elegida para guardar nuestros topics. Para habilitar este flujo entre Elasticsearch y Kafka es necesario configurar un conector con Kafka Connect y los índices:

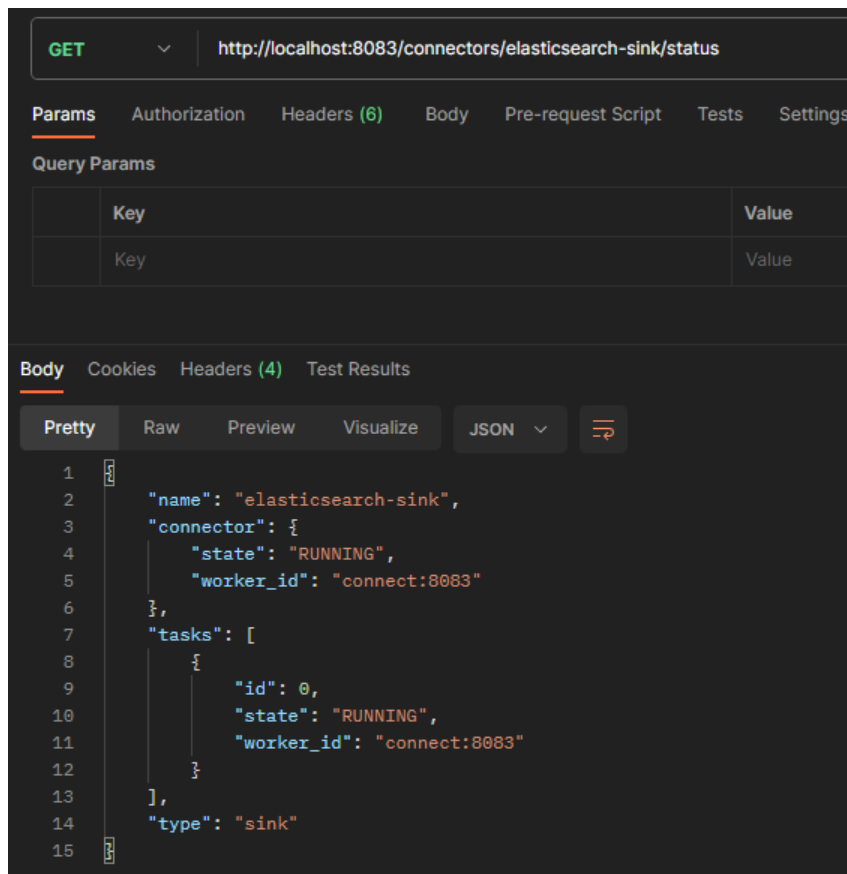
- Creamos el conector



The screenshot shows a REST client interface with the URL `http://localhost:8083/connectors` and the method `POST`. The body is set to `JSON` and contains the following configuration:

```
1 {
2   "name": "elasticsearch-sink",
3   "config": {
4     "connector.class": "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
5     "tasks.max": "1",
6     "topics": "tweets,tweets-hashtags-lang,tweets-trending-topics",
7     "key.ignore": "true",
8     "schema.ignore": "true",
9     "connection.url": "http://elastic:9200",
10    "type.name": "_doc",
11    "name": "elasticsearch-sink",
12    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
13    "key.converter.schemas.enable": "true",
14    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
15    "value.converter.schemas.enable": "false"
16  }
17 }
```

- Estado del conector



The screenshot shows a REST client interface with the URL `http://localhost:8083/connectors/elasticsearch-sink/status` and the method `GET`. The body is set to `JSON` and contains the following status information:

```
1 {
2   "name": "elasticsearch-sink",
3   "connector": {
4     "state": "RUNNING",
5     "worker_id": "connect:8083"
6   },
7   "tasks": [
8     {
9       "id": 0,
10      "state": "RUNNING",
11      "worker_id": "connect:8083"
12    }
13  ],
14   "type": "sink"
15 }
```

- Creación de índices para los topics

PUT ▼ http://localhost:9200/tweets

Params Authorization Headers (7) Body Pre-request Script

Query Params

	Key
	Key

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {  
2   "acknowledged": true,  
3   "shards_acknowledged": true,  
4   "index": "tweets"  
5 }
```

PUT ▼ http://localhost:9200/tweets-hashtags-lang

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	Key	Value
	Key	Value

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {  
2   "acknowledged": true,  
3   "shards_acknowledged": true,  
4   "index": "tweets-hashtags-lang"  
5 }
```

PUT <http://localhost:9200/tweets-trending-topics>

Params Authorization Headers (7) Body Pre-request Script

Query Params

Key
Key

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2    "acknowledged": true,
3    "shards_acknowledged": true,
4    "index": "tweets-trending-topics"
5  }

```

- Ver índices

Elasticsearch / **Indices**

GET http://localhost:9200/_cat/indices?v

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
<input checked="" type="checkbox"/> v		
Key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK

Pretty Raw Preview Visualize Text

```

1 health status index          uuid                                pri rep docs.count docs.deleted store.size pri.store.size
2 yellow open   tweets-hashtags-lang         e1kg7CNSTY6kGUDtuo168Q          1  1           0           0        208b        208b
3 green  open   .apm-custom-link             b25z5WXHR0KANTrDabluXQ          1  0           0           0        208b        208b
4 green  open   .kibana_task_manager_1       tf1gFmpTRf-FY6Y5n4XLbA          1  0           5          54       105.6kb       105.6kb
5 green  open   .apm-agent-configuration     d7PnGc0xTW0EiPgbUmeE9A          1  0           0           0        208b        208b
6 yellow open   tweets-trending-topics       Ta26Sg2dS7GUQGg9fVQ4Rg          1  1           0           0        208b        208b
7 yellow open   tweets                       04nwkGjMQ0epPfGCvI02Eg          1  1           0           0        208b        208b
8 green  open   .kibana_1                    xYpRMEEwSgm0gzCG600reA          1  0           9           0         2.1mb        2.1mb
9 green  open   .kibana-event-log-7.10.2-000001 Wc9VBeqhSnS5_XnN1EjLXg          1  0           1           0         5.6kb         5.6kb
10

```

Configuración de Java

Toda la configuración centraliza en el fichero Constants.java:

```
Constants.java X ProducerTweets.java *ConsumerTweets.java
1 package com.practice.library;
2
3 public class Constants {
4
5     public final static String BOOTSTRAP_SERVER      = "127.0.0.1:29092";
6     public final static String FILE_TWEETS          = "tweets.txt";
7     public final static String GROUP_ID             = "consumer-tweets";
8     public final static String HASHTAGS_LANG        = "es";
9     public final static int TRENDING_TOPICS_THRESHOLD = 2;
10
11     // Topics
12     public final static String TOPIC_TWEETS          = "tweets";
13     public final static String TOPIC_TWEETS_HASHTAGS_LANG = "tweets-hashtags-lang";
14     public final static String TOPIC_TWEETS_TRENDING_TOPICS = "tweets-trending-topics";
15 }
```

Productor

Por el lado del Productor, la carga de tweets se realiza por medio del fichero tweets.txt y el envío de tweets se realizará automáticamente a los topics definidos anteriormente:

```
Constants.java ProducerTweets.java X ConsumerTweets.java
48 properties.put(ProducerConfig.RETRIES_CONFIG, 3);
49 properties.put(ProducerConfig.BATCH_SIZE_CONFIG, 16384);
50 properties.put(ProducerConfig.BUFFER_MEMORY_CONFIG, 33554432);
51 properties.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
52 properties.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
53 properties.put(ProducerConfig.BootstrapServersConfig, Constants.BOOTSTRAP_SERVER);
54
55 File fileData = new File(Constants.FILE_TWEETS);
56
57 if (!fileData.exists()) {
58     log.error(Constants.FILE_TWEETS + " not found.");
59     return;
60 }
61
62 loadTweetsData(Constants.FILE_TWEETS);
63
64 // Local
65 // printHashtagLang();
66 // printTrendingTopicsCounter();
67 // printTrendingTopicsTweets();
68
69 // Kafka
70 sendTweets(properties);
71 sendHashtagsLang(properties);
72 sendTrendingTopics(properties);
73 }
74
```

Procedemos a ejecutarlo, si todo ha ido bien los mensajes se llegarán a Kafka y se almacenarán en Elasticsearch.

Consumidor

El consumidor está definido para leer un topic a la vez. Así pues, hay que configurar la línea marcada con la constante del topic que se quiera leer:

```
1 package com.practice.consumer;
2
3 import java.time.Duration;
4
5
6
7
8
9
10 public class ConsumerTweets {
11     private static final Logger log = LoggerFactory.getLogger(ConsumerTweets.class);
12     public static ObjectMapper objectMapper = new ObjectMapper();
13
14     public static void main(String[] args) {
15         log.info("-- Start Consumer");
16
17         String topic = Constants.TOPIC_TWEETS_HASHTAGS_LANG; // Configurar el topic
18
19         // create consumer configs
20         Properties properties = new Properties();
21         properties.setProperty(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, Constants.BOOTSTRAP_SERVER);
22         properties.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class.getName());
23         properties.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class.getName());
24         properties.setProperty(ConsumerConfig.GROUP_ID_CONFIG, Constants.GROUP_ID);
25         properties.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
26
27         // create consumer
28         KafkaConsumer<String, String> consumer = new KafkaConsumer<>(properties);
```

Procedemos a ejecutarlo:

```
1 package com.practice.consumer;
2
3 import java.time.Duration;
4
5
6
7
8
9
10 public class ConsumerTweets {
11     private static final Logger log = LoggerFactory.getLogger(ConsumerTweets.class);
12     public static ObjectMapper objectMapper = new ObjectMapper();
13
14     public static void main(String[] args) {
15         log.info("-- Start Consumer");
16
17         String topic = Constants.TOPIC_TWEETS_HASHTAGS_LANG;
18
19         // create consumer configs
20         Properties properties = new Properties();
21         properties.setProperty(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, Constants.BOOTSTRAP_SERVER);
22         properties.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class.getName());
23         properties.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class.getName());
24         properties.setProperty(ConsumerConfig.GROUP_ID_CONFIG, Constants.GROUP_ID);
25         properties.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
26
27         // create consumer
28         KafkaConsumer<String, String> consumer = new KafkaConsumer<>(properties);
29
30         // get a reference to the current thread
31         final Thread mainThread = Thread.currentThread();
32
33         // consume messages
34         while (true) {
35             ConsumerRecords<String, String> records = consumer.poll(Duration.ofMillis(100));
36             for (ConsumerRecord<String, String> record : records) {
37                 log.info("Received message: {} from topic: {}", record.value(), record.topic());
38             }
39         }
40     }
41 }
```

```
[main] INFO org.apache.kafka.clients.consumer.internals.SubscriptionState - [Consumer clientId=consumer-consumer-tweets-1, groupId=consumer-consumer-tweets-1, groupInstanceId=consumer-consumer-tweets-1] subscribed to topic=consumer-consumer-tweets-1, groupid=consumer-consumer-tweets-1, groupInstanceId=consumer-consumer-tweets-1
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"CREDENCIALFISICA"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"IBAB"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Paraguay"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"AydaAksel"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Hercal"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Overwatch2"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Grille"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"AyusoOsanidad"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"淄博"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"melevantosigo"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"杭州"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Revolucióntica"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Nehiri"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Argentina"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"CEO"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"BuenasTardes"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Adarasv19A"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"INA1"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"Crisis"}
[main] INFO com.practice.consumer.ConsumerTweets - {"tag":"LaPelotaEvolucion"}
```

Elasticsearch

Una vez configurado Elasticsearch y lanzado el Producer realizaremos una búsqueda de los topics:

HTTP Elasticsearch / search / **topic_search**

GET localhost:9200/tweets-hashtags-lang/_search

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "query": {
3     "match_all": {}
4   }
5 }
```

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON ▾

```
18   "_index": "tweets-hashtags-lang",
19   "_type": "_doc",
20   "_id": "tweets-hashtags-lang+0+22",
21   "_score": 1.0,
22   "_source": {
23     "tag": "Cripto"
24   }
25 },
26 {
27   "_index": "tweets-hashtags-lang",
28   "_type": "_doc",
29   "_id": "tweets-hashtags-lang+0+28",
30   "_score": 1.0,
31   "_source": {
32     "tag": "NoticiasDeChihuahua"
33   }
34 },
35 {
36   "_index": "tweets-hashtags-lang",
37   "_type": "_doc",
38   "_id": "tweets-hashtags-lang+0+13",
39   "_score": 1.0,
40   "_source": {
41     "tag": "Argentina"
42   }
43 },
```

Si todo ha ido bien veremos la lista de hashtags. El mismo proceso se realiza para leer los otros topics, hay que sustituir en la url el nombre del topic que queremos buscar:

```
GET localhost:9200/tweets-trending-topics/_search

{"query": {"match_all": {}}}
```

```
{
  "mentions": [
    {
      "start": 45,
      "end": 56,
      "id": "1476599262490284039",
      "username": "TOP100KPOP"
    }
  ],
  "annotations": [
    {
      "probability": 0.9729,
      "start": 17,
      "normalized_text": "BTS",
      "end": 19,
      "type": "Organization"
    }
  ]
},
{
  "edit_history_tweet_ids": [
    "1648411619104268289"
  ],
  "id": "1648411619104268289",
  "text": "I Vote #RM from #BTS for #TOP100KPOPMEMBERS \n@TOP100KPOP \n355",
  "lang": "en",
  "withheld": null
}
```

La API de búsqueda tiene mucho juego a la hora de aplicar queries de búsqueda.

Conclusión

He tenido ciertos problemas a la hora de realizar el filtrado de hashtags, si filtraba por prefijo de lenguaje “es” muchos de ellos estaban en tweets con lenguaje muy diferente al español. No utilice esta red social, pero deduzco que muchas de esas cuentas se crean con un lenguaje incorrecto. Tengo que confesar que no soy muy fan de las redes sociales y nunca he enviado un tweet en mi vida.

Por otro lado, yo soy programador .NET y al ser la práctica en Java me ha llevado más tiempo en realizarla por el aprendizaje del lenguaje y uso de Maven, pero ha estado bien aprender esta tecnología.

En definitiva, la práctica de Kafka ha sido realmente enriquecedora mí y mi carrera profesional, espero poder aplicar estos conocimientos en un proyecto real.