PEC PROCESAMIENTO DE STREAMS E INGESTA DE DATOS

MIGUEL PÉREZ CARO

EJERCICIO KAFKA

Para la resolución de este ejercicio el primer paso es arrancar los servicios de zookeeper y de kafka. Para ello es necesario ejecutar los siguientes comandos en el directorio donde está instalado kafka: `/home/ubuntu/bigdata/apps/kafka/bin`

- `zookeeper-server-start.sh ../config/zookeeper.properties`
- `kafka-server-start.sh ../config/server.properties`

a) Creación de topics:

- **Crear un nuevo topic denominado pec-topic1-<NombreAlumno> con dos particiones y con factor de replicación 1.**

  Una vez arrancados los servicios necesarios, es muy sencillo crear el topic tal y como se muestra en la siguiente captura de pantalla:

```
(base) ubuntu@master-1:~/bigdata/apps/kafka/bin$ kafka-topics.sh --zookeeper 127
.0.0.1:2181 --topic pec-topic1-miguel --create --partitions 2 --replication-fact
or 1
Created topic "pec-topic1-miguel".
(base) ubuntu@master-1:~/bigdata/apps/kafka/bin$ kafka-topics.sh --zookeeper 127
.0.0.1:2181 --list
__consumer_offsets
pec-topic1-miguel
topic-key
topic-test
topicPrueba
topicTaxi
topicTest
topicTruck
topicname
tweetertopic
(base) ubuntu@master-1:~/bigdata/apps/kafka/bin$
```

- **¿Sería posible crearlo en el entorno que tienes con un factor de replicación 2? ¿Por qué?**

  En el entorno creado no se podría porque el número de replicaciones no puede ser superior al número de nodos en el clúster. Por lo tanto, como solo hay un nodo, el factor de replicación no puede ser superior a 1. Si se hubiera generado un clúster en docker con más nodos, si que se podría haber usado un factor de replicación superior.

b) Crea un productor que empiece a pasar información desde la línea de comandos al topic creado anteriormente y simular el envío de información.

Para la creación del productor solo hay que ejecutar un comando y ya se puede empezar a enviar información por la línea de comandos tal y como se muestra en la siguiente imagen:

```
(base) ubuntu@master-1:~/bigdata/apps/kafka/bin$ kafka-console-producer.sh
 --broker-list 127.0.0.1:9092 --topic pec-topic1-miguel
>Esto
>es
>una
>prueba
>para
>el
>ejercicio
>de
>la
>pec
>correspondiente
>a
>la
>parte
>de
>kafka
>
```

Se puede observar el comando usado para la creación del productor, que es:

```
kafka-console-producer.sh --broker-list 127.0.0.1:9092 –topic pec-topic1-miguel
```

Y los mensajes enviados, que se corresponden con la frase: Esto es una prueba para el ejercicio de la pec correspondiente a la parte de kafka.

c) Creación de consumidores:

- **Crear un consumidor que sea capaz de leer toda la información que contenga el topic creado.**

  Este consumidor se puede crear con el siguiente comando:

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 –topic pec-topic1-miguel --from-beginning
  ```

  Es necesario añadir al final el flag from beginning porque los mensajes se habían enviado antes de haber creado el consumidor, de forma que, si no se especificara que se desea obtener la información desde el inicio, solo recogería la información que se fuera enviando desde su creación, y habría que volver a enviar dichos mensajes para que fueran recogidos por el consumidor.

A continuación, se muestra una captura de pantalla del resultado:

```
(base) ubuntu@master-1:~/bigdata/apps/kafka/bin$ kafka-console-consumer.sh --
--bootstrap-server 127.0.0.1:9092 --topic pec-topic1-miguel --from-beginning
Esto
una
para
ejercicio
la
correspondiente
la
de
es
prueba
el
de
pec
a
parte
kafka
```

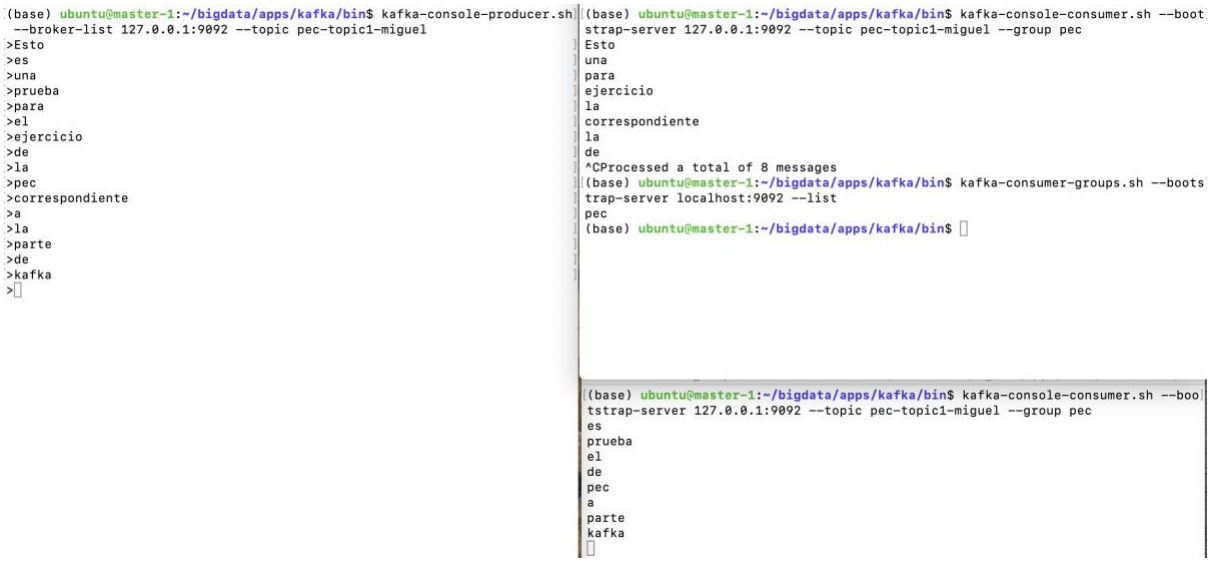
Se observa que se han recogido todos los mensajes enviados, pero están desordenados. Esto se debe a que, en primer lugar, la escritura en las particiones no sigue ningún orden específico, y simplemente se intenta balancear la carga, y a que la lectura es aleatoria, de forma que Kafka no sabe en que orden se han introducido los datos, y puede estar leyendo aleatoriamente de cada partición. El orden solo se garantiza a nivel de partición, por lo que si solo hubiera una partición si que se verían los datos ordenados.

- **Crear un grupo de consumidores con el suficiente número de ellos para que cada uno solo lea de una de las particiones del topic.**

  Para la realización de este último apartado se crea un grupo de consumidores que tendrá dos consumidores, uno para cada partición, lo cual se consigue ejecutando el siguiente comando:

  ```
  kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --
  topic pec-topic1-miguel --group pec
  ```

  A continuación, se muestran una captura de pantalla del proceso realizado:



A la izquierda de la imagen se ha generado un productor idéntico al del apartado b, enviando la misma información, y antes de que se hubiera enviado la información, se han puesto en marcha los dos consumidores del grupo que se ha denominado pec. Se puede observar como cada uno de ellos está leyendo de una partición, y como la distribución aleatoria que hace Kafka está enviando una palabra a una partición, y la siguiente palabra a la otra partición. También se listan los grupos de consumidores en la parte superior derecha de la imagen, apareciendo únicamente el que se acaba de generar.

Por último, es importante destacar que los datos también se podrían haber introducido en formato clave valor de forma que todos los datos de una misma clave se almacenan en una misma partición y así si que se consigue asegurar el orden a la hora de escribirlos y leerlos.