

PRACTICA DE KAFKA

Inicio zookeeper:

```
spark@spark-virtualBox:~/kafka_2.11-2.1.0$ cd bin
spark@spark-virtualBox:~/kafka_2.11-2.1.0/bin$ ./zookeeper-server-start.sh ../co
nfig/zookeeper.properties
[2024-08-05 08:50:41,213] INFO Reading configuration from: ../config/zookeeper.p
roperties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-08-05 08:50:41,222] INFO autopurge.snapRetainCount set to 3 (org.apache.zo
ookeeper.server.DatadirCleanupManager)
[2024-08-05 08:50:41,222] INFO autopurge.purgeInterval set to 0 (org.apache.zook
eeper.server.DatadirCleanupManager)
[2024-08-05 08:50:41,222] INFO Purge task is not scheduled. (org.apache.zookeep
er.server.DatadirCleanupManager)
[2024-08-05 08:50:41,222] WARN Either no config or no quorum defined in config,
```

Inicio kafka:

```
[2024-08-05 08:58:39,337] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-30 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2024-08-05 08:58:39,337] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-36 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2024-08-05 08:58:39,337] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-42 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
[2024-08-05 08:58:39,338] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-48 in 0 milliseconds. (kafk
a.coordinator.group.GroupMetadataManager)
```

Creo el topic -> first_topicc.

```
spark@spark-virtualBox:~$ kafka-topics.sh --zookeeper 127.0.0.1:2181 --create --
topic first_topicc --partitions 3 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or under
score ('_') could collide. To avoid issues it is best to use either, but not bot
h.
Created topic "first_topicc".
```

Vemos que esta presente:

```
spark@spark-virtualBox:~$ kafka-topics.sh --zookeeper 127.0.0.1:2181 --list
__consumer_offsets
csv_topic
csv_topic1
csv_topic2
first_topic
first_topicc
```

```
spark@spark-virtualBox:~$ kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic fir
st_topicc --describe
Topic:first_topicc      PartitionCount:3        ReplicationFactor:1    Configs:
sr: 0                   Topic: first_topicc     Partition: 0            Leader: 0                Replicas: 0              I
sr: 0                   Topic: first_topicc     Partition: 1            Leader: 0                Replicas: 0              I
sr: 0                   Topic: first_topicc     Partition: 2            Leader: 0                Replicas: 0              I
```

```
spark@spark-virtualBox:~$ kafka-topics.sh --zookeeper 127.0.0.1:2181 --create --
topic second_topic --partitions 6 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or under
score ('_') could collide. To avoid issues it is best to use either, but not bot
h.
Created topic "second_topic".
```

```
spark@spark-virtualBox:~$ kafka-topics.sh --zookeeper 127.0.0.1:2181 --list
__consumer_offsets
csv_topic
csv_topic1
csv_topic2
first_topic
first_topicc
joa_json
joa_topic
joajson
joajson2
json_topic
json_topic1
new_joa_json
new_topic
prueba_topic2
prueba_topic3
second_topic
```

Insertamos mensajes en el topic:

```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092
--topic first_topicc
>Hola Mundo
>Aprendiendo kafka
>Más texto para seguir aprendiendo
>Otro mensajes
```

Producir mensajes con garantías de confirmación (acks=all)

```
>^Cspark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9
--topic first_topicc --producer-property acks=all
>Un mensaje que es confirmado
>Otro mensaje más
>Seguimos aprendiendo
```

Creamos mensajes en topic inexistente y vemos si se ha creado:

```
>^Cspark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9
--topic new_topic
>>>>Mensaje en topic nuevo inexistente
>Otro mensaje en este topic nuevo
>Un último mensaje
>^Cspark@spark-virtualBox:~$ kafka-topics.sh --zookeeper 127.0.0.1:2181 --topic
_topic --describe
Topic:new_topic PartitionCount:1 ReplicationFactor:1 Configs:
Topic: new_topic Partition: 0 Leader: 0 Replicas: 0 I
sr: 0
```

En una consola hacemos para que consuma mensajes enviado desde otra:

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc
Nuevo mensaje 1
Nuevo mensaje 2
Nuevo mensaje 3
```

Enviamos los mensajes desde otra terminal

```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topicc
>Nuevo mensaje 1
>Nuevo mensaje 2
>Nuevo mensaje 3
```

Tambien con el siguiente comando se ven todos los mensajes de una:

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc --from-beginning
> Nuevo mensaje 1
> Nuevo mensaje 2
> Nuevo mensaje 3
```

Ahora vamos a realizar un consumer groups.

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc --group my-first-application
```

```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topicc
```

```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topicc
>Mensaje 1
>Mensaje 2
>Mensaje 3

spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc --group my-first-application
Mensaje 1
Mensaje 2
Mensaje 3
```

Vamos enviando mensajes, como siempre, solo que ahora en otra terminal vamos a iniciar a un consumidor que pertenezca al grupo.

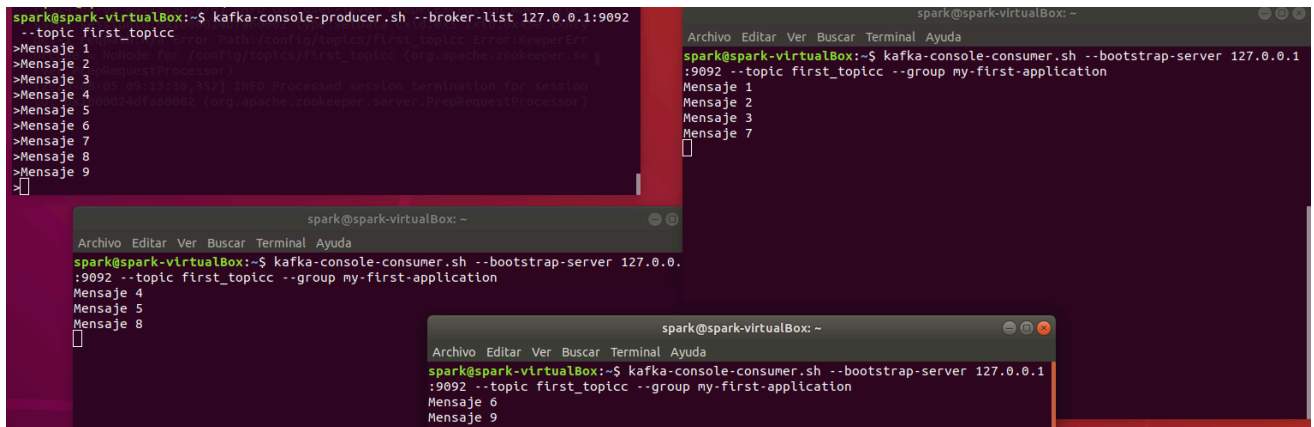
Vemos que kafka hace que los dos se comuniquen y solo lee el mensaje uno de los dos, así permite la paralelización:

```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topicc
>Mensaje 1
>Mensaje 2
>Mensaje 3
>Mensaje 4

[2024-08-05 10:55:12,701] INFO [GroupCoordinator 0]: Assignment received from leader for group my-first-application for generation 11 (kafka.coordinator.group.GroupCoordinator)

spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc --group my-first-application
Mensaje 1
Mensaje 2
Mensaje 3
```

Metemos otra consola y pasa lo mismo:



The image shows three terminal windows from a Spark virtual box. The top-left window shows a Kafka producer sending messages 1 through 9 to the topic 'first_topicc'. The top-right window shows a Kafka consumer receiving messages 1 through 7. The bottom window shows another Kafka consumer receiving messages 4 through 9. This demonstrates message distribution and offset management in a Kafka group.

```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092 --topic first_topicc
>Mensaje 1
>Mensaje 2
>Mensaje 3
>Mensaje 4
>Mensaje 5
>Mensaje 6
>Mensaje 7
>Mensaje 8
>Mensaje 9
^C
```

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc --group my-first-application
Mensaje 1
Mensaje 2
Mensaje 3
Mensaje 7
^C
```

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc --group my-first-application
Mensaje 4
Mensaje 5
Mensaje 8
^C
```

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topicc --group my-first-application
Mensaje 6
Mensaje 9
^C
```

- **Ejercicio 1. ¿Qué pasaría si cancelamos (utilizando Ctrl+C) uno de los consumidores (quedando 2) y seguimos enviando mensajes por el producer?**

Si se cancela uno de los consumidores en un grupo que tiene tres consumidores y se siguen enviando mensajes, los otros consumidores asumirán la carga de particiones del consumidor cancelado. Los mensajes nuevos serán leídos por los consumidores restantes, distribuyéndose entre ellos.

- **Ejercicio 2. ¿Qué pasaría si cancelamos otro de los consumidores (quedando ya solo 1) y seguimos enviando mensajes por el producer?**

Al cancelar el segundo consumidor en un grupo, solo quedará un consumidor, el cual manejará todas las particiones del topic. Este único consumidor leerá todos los mensajes nuevos que se envíen, procesándolos en su totalidad.

- **Ejercicio 3. ¿Qué sucede si lanzamos otro consumidor pero está vez de un grupo llamado my-second-application leyendo el topic desde el principio (--from-beginning)?**

Cuando se lanza un nuevo consumidor con un grupo diferente (**my-second-application**) y se configura la lectura desde el principio, este leerá todos los mensajes desde el inicio del topic. Esto se debe a que no tiene información previa sobre los offsets del topic.

- **Ejercicio 4. Cancela el consumidor y ¿Qué sucede si lanzamos de nuevo el consumidor pero formando parte del grupo my-secondapplication? ¿Aparecen los mensajes desde el principio? Pista -> kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topic --group my-second-application**

Si se cancela el consumidor del grupo `my-second-application` y luego se inicia uno nuevo con el mismo grupo, el nuevo consumidor continuará leyendo desde el último offset confirmado. No leerá desde el principio del topic, sino desde el último mensaje que no ha sido procesado.

• **Ejercicio 5. Cancela el consumer, a su vez aprovecha de enviar más mensajes utilizando el producer y de nuevo lanza el consumidor formando parte del grupo `my-second_application` ¿Cuál fue el resultado?**

Al reiniciar un consumidor del grupo `my-second-application` después de cancelar el anterior y enviar más mensajes, el nuevo consumidor leerá los mensajes nuevos que se enviaron después de la cancelación. Continuará desde el último offset confirmado, procesando solo los mensajes nuevos desde ese punto.

Ahora listamos los topics:

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --list
my-first-application
my-second-application
```

Vemos el offset por particiones:

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --describe --group my-first-application
```

TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	CONSU
MER-ID			HOST	CLIENT-ID	
first_topicc	1	7	7	0	consu
mer-1-ae2ad5f2-d6b6-4312-9683-c50e1d8690a5			/127.0.0.1	consumer-1	
first_topicc	0	6	6	0	consu
mer-1-02694743-16d0-472e-adfc-2be72cb47e76			/127.0.0.1	consumer-1	
first_topicc	2	7	7	0	consu
mer-1-d5091d2f-75f3-4d4c-8801-7567171183af			/127.0.0.1	consumer-1	
first_topic	0	3	3	0	-

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --describe --group my-second-application
```

TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	CONSU
MER-ID			HOST	CLIENT-ID	
first_topicc	0	6	6	0	consu
mer-1-2e96c839-5473-4f9b-81cb-c41a01f9d62b			/127.0.0.1	consumer-1	
first_topicc	1	7	7	0	consu
mer-1-2e96c839-5473-4f9b-81cb-c41a01f9d62b			/127.0.0.1	consumer-1	
first_topicc	2	7	7	0	consu
mer-1-2e96c839-5473-4f9b-81cb-c41a01f9d62b			/127.0.0.1	consumer-1	

Reseteo el offset del consumer group:

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --group my-first-application --reset-offsets --to-earliest --execute --topic first_topic
```

TOPIC	PARTITION	NEW-OFFSET
first_topic	0	0

Reinicio el consumer:

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic first_topic --group my-first-application
> Nuevo mensaje 1
> Nuevo mensaje 2
> Nuevo mensaje 3
```

Al reiniciar el consumidor, este reanudará la lectura de mensajes desde el último offset confirmado en el grupo de consumidores. Si el consumidor había sido detenido por un tiempo y luego reiniciado, empezará desde el punto donde se había detenido anteriormente, siempre y cuando el grupo de consumidores no haya sido reconfigurado o el offset no haya sido alterado. Si el grupo está inactivo o se han hecho cambios en los offsets, podría leer desde un punto diferente.

Listo el primer grupo:

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --describe --group my-first-application
```

TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	CONSUMER-ID
first_topic	0	3	3	0	consumer-1
first_topic	2	7	7	0	-
first_topic	1	7	7	0	-
first_topic	0	6	6	0	-

El parámetro **--shift-by** en el comando **kafka-consumer-groups.sh** se utiliza para ajustar los offsets de los consumidores en un grupo moviéndolos hacia adelante o hacia atrás en un número especificado de mensajes.

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --group my-first-application --reset-offsets --shift-by 2 --execute --topic first_topic
[2024-08-05 12:47:25,183] WARN New offset (5) is higher than latest offset for topic partition first_topic-0. Value will be set to 3 (kafka.admin.ConsumerGroupCommand$)

TOPIC          PARTITION  NEW-OFFSET
first_topic    0          3
```


Ejercicio final:

- Creo un nuevo topic "topic_app1"

```
spark@spark-virtualBox:~$ kafka-topics.sh --create --zookeeper 127.0.0.1:2181 --
replication-factor 1 --partitions 3 --topic topic_app1
WARNING: Due to limitations in metric names, topics with a period ('.') or under
score ('_') could collide. To avoid issues it is best to use either, but not bot
h.
Created topic "topic_app1".
```

- Creo un producer para que inserte mensajes:

```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092
--topic topic_app1
```

Inserto mensajes en el

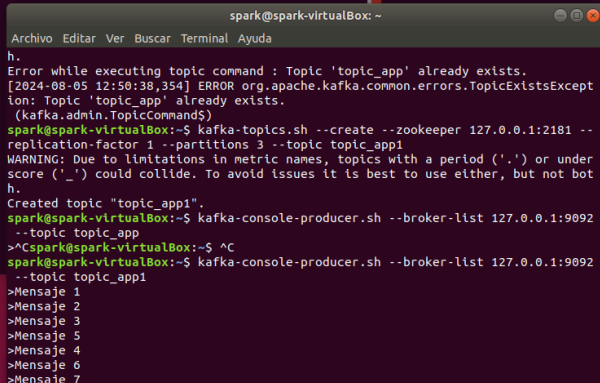
```
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092
--topic topic_app1
>Mensaje 1
>Mensaje 2
>Mensaje 3
```

- Creo dos consumer

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1
:9092 --topic topic_app1 --group my_app
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1
:9092 --topic topic_app1 --group my_app
```

- Interactúo con ellos

```
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1
:9092 --topic topic_app1 --group my_app
Mensaje 5
Mensaje 6
Mensaje 7
spark@spark-virtualBox:~$ kafka-console-consumer.sh --bootstrap-server 127.0.0.1
:9092 --topic topic_app1 --group my_app
Mensaje 4
```



```
spark@spark-virtualBox:~$ kafka-topics.sh --create --zookeeper 127.0.0.1:2181 --
replication-factor 1 --partitions 3 --topic topic_app1
WARNING: Due to limitations in metric names, topics with a period ('.') or under
score ('_') could collide. To avoid issues it is best to use either, but not bot
h.
Created topic "topic_app1".
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092
--topic topic_app1
>^C
spark@spark-virtualBox:~$ kafka-console-producer.sh --broker-list 127.0.0.1:9092
--topic topic_app1
>Mensaje 1
>Mensaje 2
>Mensaje 3
>Mensaje 5
>Mensaje 4
>Mensaje 6
>Mensaje 7
```

- Listo los topics y vemos que esta el "topic_app1"

```
spark@spark-virtualBox:~$ kafka-topics.sh --list --zookeeper 127.0.0.1:2181
__consumer_offsets
csv_topic
csv_topic1
csv_topic2
first_topic
first_topicc
joa_json
joa_topic
joajson
joajson2
json_topic
json_topic1
new_joa_json
new_topic
prueba_topic2
prueba_topic3
text_topic
text_topic2
text_topic3
topic_app
topic_app1
```

- Listo los grupos de consumo:

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --list
my_app
my-first-application
my-second-application
```

- Describo el grupo de consumo “my_app”

```
spark@spark-virtualBox:~$ kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --describe --group my_app
```

TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	CONSUMER-ID	HOST
topic_app1	0	4	4	0	consumer-1	/127.0.0.1
topic_app1	1	3	3	0	consumer-1	/127.0.0.1
topic_app1	2	2	2	0	consumer-1	/127.0.0.1
topic_app	2	5	5	0	-	-
topic_app	0	4	4	0	-	-
topic_app	1	4	4	0	-	-