Apartado 1
Jorge Pablo Ávila Gómez

1. **Localizar el archivo que contiene la configuración del servidor Zookeeper y levantar dicho servidor utilizando la configuración por defecto. ¿Qué puerto es el utilizado por defecto por Zookeeper?**

   El archivo de configuración del servidor Zookeeper se encuentra en: .\config\zookeeper.properties

   El servidor se levanta con el comando:

   .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

   El Puerto por defecto utilizado por Zookeeper es: 2181

2. **Localizar el archivo que contiene la configuración de un broker y levantar dicho broker. ¿Qué puerto es el utilizado por defecto por los brokers?**

   El archivo de configuración del bróker se encuentra en: .\config\server.properties

   El servidor se levanta con el comando:

   .\bin\windows\kafka-server-start.bat .\config\server.properties

   El Puerto por defecto utilizado para el bróker es: 9092

3. **Crear un topic nuevo con el nombre "initial", con una partición y un factor de replicación igual a 1.**

   El topic se crea con el comando:

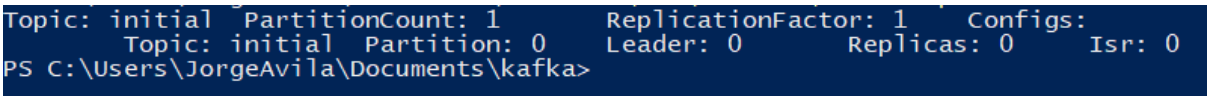
   .\bin\windows\kafka-topics.bat --create --zookeeper 192.168.1.100:2181 --replication-factor 1 --partitions 1 --topic initial

4. **Describir el topic "initial" y comentar los resultados que se obtienen.**

   El topic se describe con el comando:

   .\bin\windows\kafka-topics.bat --describe --zookeeper 192.168.1.100:2181 --topic initial

Aparecen los siguientes resultados:

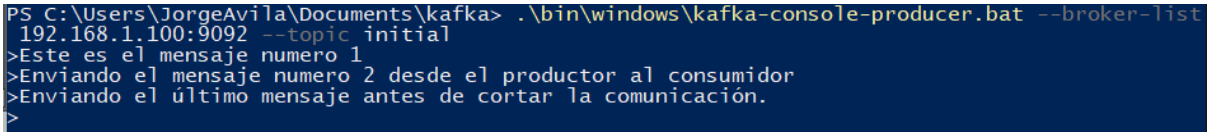


Indicando que hay 1 partición, y que el factor de replicación es 1 como se configuró en el ejercicio 3. En la partición cero el lider es el broker cero y hay una réplica en el broker cero. Isr (In-Sync Replica) indica las replicas que están sincronizadas con el lider. En este caso la replica del broker 0 está sincronizada con el lider, que es él mismo.

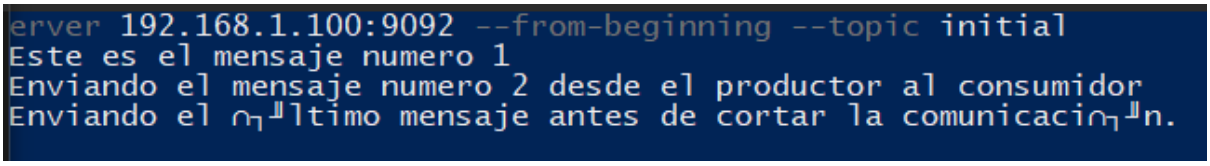
5. **Arrancar un productor sobre el topic "initial".**
   El productor se arranca con el siguiente comando:
   .\bin\windows\kafka-console-producer.bat --broker-list 192.168.1.100:9092 --topic initial

6. **Arrancar un consumidor sobre el topic "initial" que consuma todos los mensajes desde el principio.**
   El consumidor se arranca con el siguiente comando:
   .\bin\windows\kafka-console-consumer.bat --bootstrap-server 192.168.1.100:9092 --from-beginning --topic initial

7. **Generar algunos mensajes en el productor levantado en el ejercicio 5 y observar cómo se reciben en el consumidor levantado en el ejercicio 6. Comentar la salida obtenida. Detener a continuación el productor y el consumidor.**
   Se han enviado tres mensajes entre el productor y el consumidor. Aquí vemos los mensajes enviados por el productor:



   Y ahora vemos los mensajes cuando llegan al consumidor:



   Vemos que los dos primeros mensajes han llegado perfectamente, unos segundos después de que fuesen enviados. Por otro lado, el tercer mensaje presenta fallos en las letras con acento, por tanto, parece que no soporta el envío de cualquier carácter.

8. **Crear un topic nuevo con el nombre "testRep", con dos particiones y un factor de replicación igual a 2. ¿Qué sucede? ¿Cuál es la razón?**
   El topic se crea con el comando:
   .\bin\windows\kafka-topics.bat --create --zookeeper 192.168.1.100:2181 --replication-factor 2 --partitions 2 --topic testRep

Al intentar ejecutarlo da un error porque el factor de replicación (2) es mayor que el número de brokers (1). Deberíamos tener al menos dos brokers para poder crear este topic.

```
PS C:\Users\JorgeAvila\Documents\kafka> .\bin\windows\kafka-topics.bat --create --zookeeper 19
2.168.1.100:2181 --replication-factor 2 --partitions 2 --topic testRep
Error while executing topic command : Replication factor: 2 larger than available brokers: 1.
[2021-01-08 16:28:05,744] ERROR org.apache.kafka.common.errors.InvalidReplicationFactorExcepti
on: Replication factor: 2 larger than available brokers: 1.
 (kafka.admin.TopicCommand$)
```

9. **Detener el broker levantado en el ejercicio 2.**
   Se detiene el broker levantado anteriormente.

10. **Levantar tres brokers distintos: generar nuevos ficheros de configuración para cada uno de ellos indicando las modificaciones que han sido necesarias. Los identificadores de los brokers serán 0, 1 y 2.**
    Se han generado 3 ficheros de configuración, uno para cada broker. En cada archivo se han modificado las siguientes líneas. Para el broker0:

    broker.id=0
    listeners=PLAINTEXT://:9092
    log.dirs=/tmp/kafka-logs-0

    Para el broker1:

    broker.id=1
    listeners=PLAINTEXT://:9093
    log.dirs=/tmp/kafka-logs-1

    Para el broker2:

    broker.id=2
    listeners=PLAINTEXT://:9094
    log.dirs=/tmp/kafka-logs-2

    Le damos a cada broker una id única. También cada broker debe tener un número de puerto único. Por último, los archivos de log deben ser diferentes para cada broker.

11. **Describir el topic "initial" comentar los resultados que se obtienen.**
    El topic se describe con el comando:
    .\bin\windows\kafka-topics.bat --describe --zookeeper 192.168.1.100:2181 --topic initial
    Se obtienen los mismos resultados que en el ejercicio 4.
    Hay 1 partición, y el factor de replicación es 1 como se configuró en el ejercicio 3. En la partición cero el líder es el broker cero y hay una réplica en el broker cero.

12. **Crear un topic nuevo con el nombre "testOrder1", con 1 partición y factor de replicación igual a 3.**

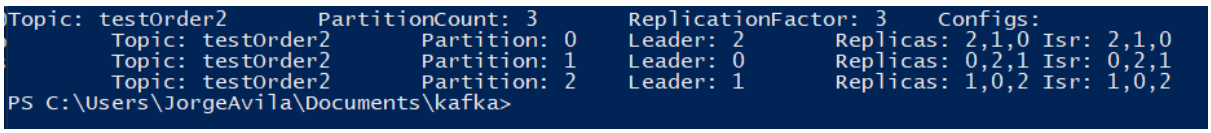
El topic se crea con el comando:
.\bin\windows\kafka-topics.bat --create --zookeeper 192.168.1.100:2181 --replication-factor 3 --partitions 1 --topic testOrder1

13. **Crear un topic nuevo con el nombre "testOrder2", con 3 particiones y factor de replicación igual a 3.**
    El topic se crea con el comando:
    .\bin\windows\kafka-topics.bat --create --zookeeper 192.168.1.100:2181 --replication-factor 3 --partitions 3 --topic testOrder2

14. **Describir los topics "testOrder1" y "testOrder2" y comentar los resultados que se obtienen.**
    El topic *testOrder1* se describe con el comando:
    .\bin\windows\kafka-topics.bat --describe --zookeeper 192.168.1.100:2181 --topic *testOrder1*
    Se obtiene el siguiente resultado:

    Vemos que tiene 1 partición y que el factor de replicación es 3. En la partición cero, el líder es el broker 0. Está replicado en los brokers 0, 1 y 2. Además, las réplicas de todos los brokers están sincronizadas (isr).

    El topic *testOrder2* se describe con el comando:
    .\bin\windows\kafka-topics.bat --describe --zookeeper 192.168.1.100:2181 --topic *testOrder2*
    Se obtiene el siguiente resultado:



    Vemos que tiene 3 particiones y que el factor de replicación es 3. En la partición cero, el líder es el broker 2, en la partición 1 el líder es el broker 0, y en la partición 2 el líder en el broker 1. En cada partición hay réplicas en los brokers 0, 1 y 2, y todas las réplicas están sincronizadas.

15. **Arrancar un productor sobre el topic "testOrder1 y un consumidor desde el inicio sobre el mismo topic. Introducir algunos mensajes y observar cómo se consumen. Detener el productor y el consumidor.**
    El productor se arranca con el comando: .\bin\windows\kafka-console-producer.bat --broker-list 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder1

El consumidor se arranca con el comando: .\bin\windows\kafka-console-consumer.bat --bootstrap-server 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --from-beginning --topic testOrder1

Los mensajes se envían y se reciben sin ningún problema desde el productor hasta el consumidor. De igual modo que en el ejercicio 7.

16. **Arrancar un productor sobre el topic "testOrder2" y un consumidor desde el inicio sobre el mismo topic. Introducir algunos mensajes y observar cómo se consumen. Detener el productor y el consumidor.**
    El productor se arranca con el comando: .\bin\windows\kafka-console-producer.bat --broker-list 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder2

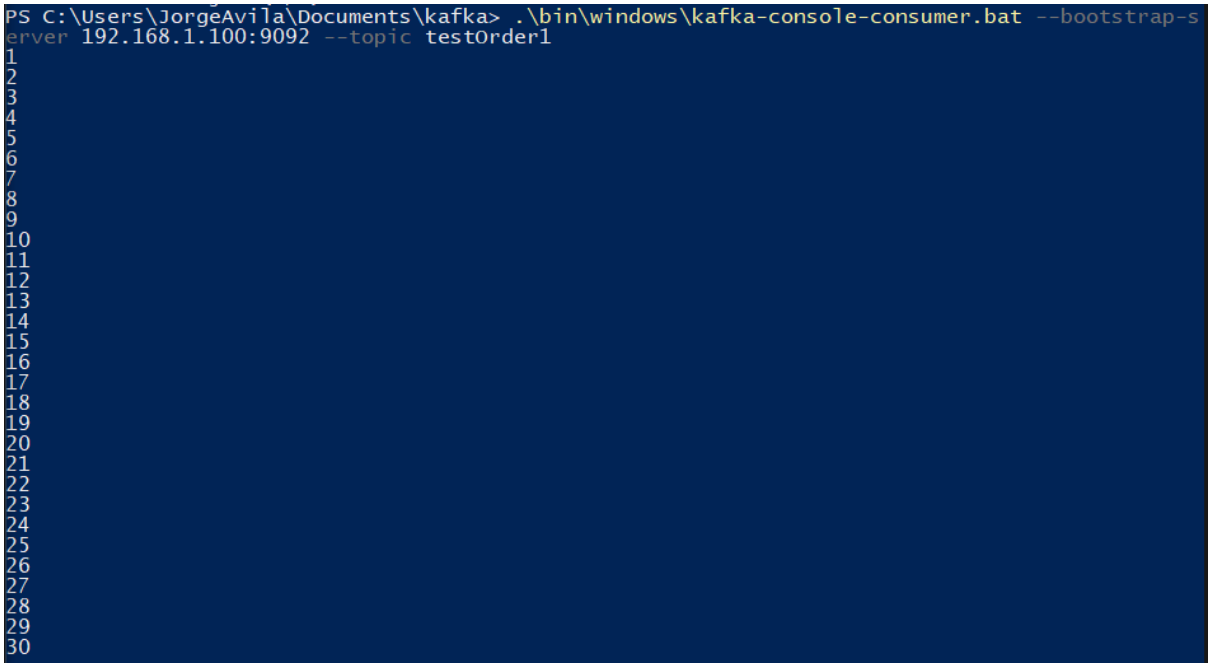
    El consumidor se arranca con el comando: .\bin\windows\kafka-console-consumer.bat --bootstrap-server 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --from-beginning --topic testOrder2

    Los mensajes se envían y se reciben sin ningún problema desde el productor hasta el consumidor. De igual modo que en el ejercicio 7.

17. **Arrancar un consumidor sobre el topic "testOrder1", sin que consuma desde el principio, únicamente los valores que le lleguen desde el nuevo arranque. En otra consola, generar una secuencia de números del 1 al 30 (utilizar el comando "seq") y enviársela mediante una tubería o pipe (indicada con el carácter "|") a un productor sobre el topic "testOrder1". Observar cómo se consume dicha secuencia en el consumidor.**
    El consumidor se arranca con el comando: .\bin\windows\kafka-console-consumer.bat --bootstrap-server 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder1

    Para enviar la secuencia de números en Windows se usa el siguiente comando: [Linq.Enumerable]::Range(1,30) | .\bin\windows\kafka-console-producer.bat --broker-list 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder1

    El consumidor recibe cada número en una línea diferente y ordenados.



18. **Arrancar un consumidor sobre el topic "testOrder2", sin que consuma desde el principio, únicamente los valores que le lleguen desde el nuevo arranque. En otra consola, generar una secuencia de números del 1 al 30 (utilizar el comando "seq") y enviársela mediante una tubería o pipe (indicada con el carácter "|") a un productor sobre el topic "testOrder2". Observar cómo se consume dicha secuencia en el consumidor.**
    El consumidor se arranca con el comando: .\bin\windows\kafka-console-consumer.bat --bootstrap-server 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder2

    Para enviar la secuencia de números en Windows se usa el siguiente comando: [Linq.Enumerable]::Range(1,30) | .\bin\windows\kafka-console-producer.bat --broker-list 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder2

    El consumidor recibe cada número en una línea diferente y ordenados, igual que en el ejercicio anterior.

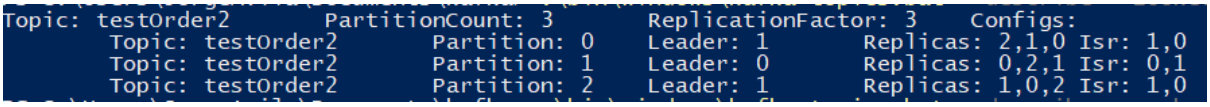
19. **¿Cuáles son las diferencias entre lo mostrado en el consumidor en los dos ejercicios anteriores? ¿A qué se deben?**
    Como se ha comentado en el foro de la asignatura desde al menos la versión 2.4, el modo por defecto implica la entrega ordenada. Por tanto, no se encuentran diferencias entre las ejecuciones de los dos ejercicios anteriores.

20. **Obtener los últimos offsets de los mensajes del topic "testOrder1" y "testOrder2". ¿Qué se observa? (Utilizar el comando kafka-run-class con la opción kafka.tools.GetOffsetShell).**

El comando para obtener los offset del topic testOrder1 es:
.\bin\windows\kafka-run-class.bat kafka.tools.GetOffsetShell --broker-list 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder1
El resultado es:
testOrder1:0:164
Esto quiere decir que en el topic testOrder1, el broker 0 ha procesado un total de 164 mensajes (la suma de todos los mensajes enviados en los ejercicios anteriores usando ese topic.)

El comando para obtener los offset del topic testOrder2 es:
.\bin\windows\kafka-run-class.bat kafka.tools.GetOffsetShell --broker-list 192.168.1.100:9092, 192.168.1.100:9093, 192.168.1.100:9094 --topic testOrder2
El resultado es:
testOrder2:0:70
testOrder2:1:67
testOrder2:2:37
Esto quiere decir que en el topic testOrder2, el broker 0 ha procesado un total de 70 mensajes, el broker 1 67, y el broker 2 37 mensajes. Vemos un ejemplo de computación distribuida usando esta plataforma.

21. **Detener de forma no planificada (utilizando Ctrl-C o cerrando la consola) el broker con identificador 2, y describir el topic "testOrder2". ¿Cuáles son las diferencias con la descripción que obtuvimos en el ejercicio 14? ¿A qué se deben?**
Tras parar el broker 2 con el siguiente comando se describe el topic testOrder2:
.\bin\windows\kafka-topics.bat --describe --zookeeper 192.168.1.100:2181 --topic *testOrder2*
Se obtiene el siguiente resultado:



La principal diferencia que vemos es que ahora el líder de la partición 0 es el broker 1, (originalmente era el broker 2).
Como el broker 2 ha dejado de funcionar, es necesario asignar un nuevo líder a las particiones en las que este era el líder. En este caso se ha asignado como nuevo líder el broker1, los lideres posibles son aquellos otros broker que tenían su replica sincronizada con la información del líder. (Los broker que aparecen en la lista isr. In-Sync Replica).
Vemos también que las replicas asignadas en cada broker no cambian. Pero las replicas que están sincronizadas si han cambiado. Vemos que solo los brokers 0 y 1 tienen replicas sincronizadas. Como el broker 2 no está conectado no se conoce su log y, por tanto, se considera que sus réplicas no están sincronizadas.

Comentario final sobre el apartado 1

Encuentro muy práctico la forma de este trabajo para familiarizarse con el funcionamiento de Kafka. No se han encontrado difíciles las preguntas y en general se sigue bien el desarrollo de las cuestiones y como deben contestarse.