

ENMILOCALFUNCIONA

THOUGHTS, STORIES AND IDEAS.



Aprendiendo Apache Kafka (Parte 6) : Ejemplo de Múltiples Brokers con 1 Topic de 1 partición y con replicación

Publicado por Víctor Madrid el 03 August 2020

Arquitectura de Soluciones

Kafka

event-driven

En este **sexto artículo** de la serie "**Aprendiendo Apache Kafka**" se va a detallar un **ejemplo práctico** de una **configuración de múltiples brokers/nodos** donde se usará un topic (con la partición básica o por defecto) con **configuración de replicación**, así vamos a poder seguir entendiendo como funcionan en "real" los conceptos explicados en los anteriores artículos.



A modo de recordatorio pongo los enlaces a los artículos anteriores :

- [Introducción](#)
- [Conceptos Básicos](#) donde se trataron : Zookeeper, Broker, Clúster, Mensaje, Esquema, Topic, Partición y Offset
- [Conceptos Básicos para Desarrollo](#) donde se trataron : Connect, Streams, Pr...

Subscribe

- **Instalación, Configuración y Ejemplo Práctico Básico** donde se trataron: instalación/configuración básica, gestión básica de topics y un pequeño ejemplo práctico.
- **Configuración de múltiples Brokers** donde se trataron: instalación/configuración básica de múltiples brokers.
- **Infraestructura de Kafka con Docker** donde se trato la instalación/configuración básica de múltiples brokers con diferentes "escenarios basados" en contenedores Docker.

El caso que utilizaremos como ejemplo práctico consistirá en :

- *Arrancar el Zookeeper*
- *Arrancar 4 brokers (nodos) en el clúster*
- *Definir un topic (tema) con una partición y con replicación*
- *Utilizar un productor "por línea de comandos"*
- *Utilizar variantes de comportamientos de consumo "por línea de comandos"*
- Variante 1: Un sólo consumidor
- Variante 2: Dos consumidores CON lectura desde el inicio
- Variante 3: Dos consumidores SIN lectura desde el inicio
- Variante 4: Dos consumidores CON el mismo grupo de consumo
- *Enviar mensajes desde el productor y ver que comportamiento tiene en las diferentes variantes*
- Comprobar el funcionamiento de replicación

Este artículo esta dividido en 3 partes:

- **1. Configuración:** Detalles sobre la configuración requerida para la realización del ejercicio práctico.
- **2. Ejemplo Práctico:** Implementación del ejercicio práctico que deberá cumplir los objetivos marcados.
- **3. Conclusiones:** Opinión sobre los resultados obtenidos.

1. Configuración

En este apartado se enseñará como preparar el entorno para el ejercicio práctico.

Ayuda :

Esta preparación se puede realizar sobre una instalación específica como la que se enseña en los anteriores artículos o bien hacer uso de una instalación basada en contenedores para acelerar el desarrollo

Si quieres saber más sobre como montar la infraestructura de Kafka con contenedores Docker mira el artículo **Infraestructura de Kafka con Docker**

Para facilitar el desarrollo haremos uso de la infraestructura mediante contenedores y para ello haremos uso del siguiente fichero "docker-compose": **zk-single-kafka-multiple-4**

Contenido del fichero:

```

version: '3'

services:

  zookeeper-1:
    image: confluentinc/cp-zookeeper:5.5.0
    hostname: zookeeper-1
    environment:
      ZOOKEEPER_SERVER_ID: 1
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
      ZOOKEEPER_LOG4J_ROOT_LOGLEVEL: INFO
      # ZOOKEEPER_LOG4J_LOGGERS: "INFO,CONSOLE,ROLLINGFILE" -- No Include
    ports:
      - "2181:2181"
    volumes:
      - ./zk-single-kafka-multiple-4/zookeeper-1/conf:/etc/kafka/
      - ./zk-single-kafka-multiple-4/zookeeper-1/logs:/logs
      - ./zk-single-kafka-multiple-4/zookeeper-1/data:/var/lib/zookeeper/data
      - ./zk-single-kafka-multiple-4/zookeeper-1/data-log:/var/lib/zookeeper/log/

  kafka-1:
    image: confluentinc/cp-kafka:5.5.0
    hostname: kafka-1
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: "zookeeper-1:2181"
      KAFKA_ADVERTISED_LISTENERS: LISTENER_DOCKER_INTERNAL://kafka-1:29092,LISTENER_DOCKER_EXTERNAL://${DOCKER_HOST_IP:-127.0.0.1}:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
        LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
      KAFKA_LOG4J_ROOT_LOGLEVEL: INFO
      # KAFKA_LOG4J_LOGGERS:
      "kafka.controller=INFO,kafka.producer.async.DefaultEventHandler=INFO,state.change.logger=INFO"
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    ports:
      - "9092:9092"
      - "29092:29092"
    volumes:
      - ./zk-single-kafka-multiple-4/kafka-1/logs:/var/log/kafka
      - ./zk-single-kafka-multiple-4/kafka-1/data:/var/lib/kafka/data
    depends_on:
      - zookeeper-1

  kafka-2:
    image: confluentinc/cp-kafka:5.5.0
    hostname: kafka-2
    environment:
      KAFKA_BROKER_ID: 2
      KAFKA_ZOOKEEPER_CONNECT: "zookeeper-1:2181"
      KAFKA_ADVERTISED_LISTENERS: LISTENER_DOCKER_INTERNAL://kafka-2:29093,LISTENER_DOCKER_EXTERNAL://${DOCKER_HOST_IP:-127.0.0.1}:9093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
        LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
      KAFKA_LOG4J_ROOT_LOGLEVEL: INFO
      # KAFKA_LOG4J_LOGGERS:
      "kafka.controller=INFO,kafka.producer.async.DefaultEventHandler=INFO,state.change.logger=INFO"
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    ports:
      - "9093:9093"
      - "29093:29093"
    volumes:
      - ./zk-single-kafka-multiple-4/kafka-2/logs:/var/log/kafka
      - ./zk-single-kafka-multiple-4/kafka-2/data:/var/lib/kafka/data
    depends_on:
      - zookeeper-1

  kafka-3:
    image: confluentinc/cp-kafka:5.5.0
    hostname: kafka-3
    environment:
      KAFKA_BROKER_ID: 3
      KAFKA_ZOOKEEPER_CONNECT: "zookeeper-1:2181"
      KAFKA_ADVERTISED_LISTENERS: LISTENER_DOCKER_INTERNAL://kafka-3:29094,LISTENER_DOCKER_EXTERNAL://${DOCKER_HOST_IP:-127.0.0.1}:9094
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:

```

```

LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
KAFKA_LOG4J_ROOT_LOGLEVEL: INFO
# KAFKA_LOG4J_LOGGERS:
"kafka.controller=INFO,kafka.producer.async.DefaultEventHandler=INFO,state.change.logger=INFO"
KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
ports:
- "9094:9094"
- "29094:29094"
volumes:
- ./zk-single-kafka-multiple-4/kafka-3/logs:/var/log/kafka
- ./zk-single-kafka-multiple-4/kafka-3/data:/var/lib/kafka/data
depends_on:
- zookeeper-1

kafka-4:
image: confluentinc/cp-kafka:5.5.0
hostname: kafka-4
environment:
KAFKA_BROKER_ID: 4
KAFKA_ZOOKEEPER_CONNECT: "zookeeper-1:2181"
KAFKA_ADVERTISED_LISTENERS: LISTENER_DOCKER_INTERNAL://kafka-
4:29095,LISTENER_DOCKER_EXTERNAL://{DOCKER_HOST_IP:-127.0.0.1}:9095
KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
KAFKA_LOG4J_ROOT_LOGLEVEL: INFO
# KAFKA_LOG4J_LOGGERS:
"kafka.controller=INFO,kafka.producer.async.DefaultEventHandler=INFO,state.change.logger=INFO"
KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
ports:
- "9095:9095"
- "29095:29095"
volumes:
- ./zk-single-kafka-multiple-4/kafka-4/logs:/var/log/kafka
- ./zk-single-kafka-multiple-4/kafka-4/data:/var/lib/kafka/data
depends_on:
- zookeeper-1

```

2. Ejemplo Práctico

En este apartado se va a detallar la realización de un ejercicio práctico mediante el uso de un productor y un consumidor por consola, es decir, mediante el uso de la línea de comandos.

Nota : Recordar que los comandos utilizados cambian si usas Windows o Linux/Mac

Estos son los pasos a seguir :

2.1. Arrancar la Infraestructura

Comando para arrancar el escenario

```
docker-compose -f zk-single-kafka-multiple-4.yml up -d
```

- -f : Indica el fichero que ejecutará
- -d : Ejecuta los contenedores en modo background

Comando para parar el escenario

```
docker-compose -f zk-single-kafka-multiple-4.yml down
```

Se pueden utilizar las herramientas de soporte que se enseñan en el artículo [Instalación, Configuración y Ejemplo Práctico Básico](#)

Se verificará que se ha arrancado correctamente (1 Zookeeper + 4 Brokers Kafka).

■ Nota : Recordar eliminar los volúmenes si se quiere reiniciar toda la configuración

2.2. Crear el Topic requerido

Se ha establecido como nombre : **"topic-basic-replication-test"**

Este apartado se divide en varios pasos:

Paso 1: Verificar que el topic no existe previamente

```
kafka-topics --list --zookeeper localhost:2181
```

Paso 2: Crear un topic con una partición y CON replicación

Hay que ejecutar el siguiente comando :

```
kafka-topics --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--zookeeper** : Establece la dirección del Zookeeper con la que trabajará
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")
- **--replication-factor** : Si Kafka se está ejecutando en un clúster, esto determina en cuántos brokers se replicará una partición (En este caso 3)
- **--partitions** : Define cuántas particiones habrá en un topic (En este caso 1)

Detalles:

- Se utilizará una única partición para almacenar mensajes -> NO habrá distribución de fragmentos entre sus brokers (No hay paralelización del trabajo)
- Como el factor de replicación es 3, se consigue que la partición tenga la siguiente estructura de distribución en los 4 brokers : 1 partición leader en uno de brokers (que se elegirá aleatoriamente), 2

particiones réplicas en 2 de los brokers y el otro broker que falta, estará libre de tener nada relacionado con la partición

- Como el nº de brokers es superior al de réplicas con partición habrá un broker que permanecerá inactivo para trabajar con ese topic, hasta que quizás alguno de los brokers activos falle

```
➤ kafka-topics --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic topic-basic-replication-test
Created topic topic-basic-replication-test.
➤
```

Verificar que se ha creado correctamente.

Paso 3: Verificar que el topic existe en el ámbito de Zookeeper

```
kafka-topics --list --zookeeper localhost:2181
```

Paso 4: Verificar los detalles del topic creado

```
kafka-topics --describe --zookeeper localhost:2181 --topic topic-basic-replication-test
```

Importante

Cada vez que se crea un topic con una configuración particular, la infraestructura de brokers utilizada y ciertos aspectos de su configuración puede implicar cambios en los brokers a la hora de asignar la partición leader y las réplicas.

Por lo que quizás, tu ejemplo puede que no cumpla 100% lo que se muestra aquí. Pero tranquilo el cambio será mínimo y lo verás en seguida.

```
➤ kafka-topics --describe --zookeeper localhost:2181 --topic topic-basic-replication-test
Topic: topic-basic-replication-test PartitionCount: 1 ReplicationFactor: 3 Configs:
Topic: topic-basic-replication-test Partition: 0 Leader: 1 Replicas: 1,2,3 Isr: 1,2,3
➤
```

Se puede verificar que el topic indica (para este caso) que :

- El broker leader es el que tiene el id igual a 1 y que la partición leader que tiene asignada es la 0 (además de ser una única partición, no se trabaja con el valor 1 sino con el valor 0).
- En este caso también se indica que existen réplicas de la partición en los brokers con id 2 y 3. También se puede ver su cumplimiento en el ISR.
- Hay que tener en cuenta que el broker con id 4 está activo en el clúster pero que no interviene en el trabajo con este topic -> Estará disponible para caídas

Según esto los brokers quedarían así para mi ejecución :

- El broker con id 1 y puerto 9092 es LEADER
- El broker con id 2 y puerto 9093 es REPLICA
- El broker con id 3 y puerto 9094 es REPLICA

- El broker con id 4 y puerto 9095 está activo pero NO tiene nada

2.3. Ejecución de variantes

En este apartado se van a detallar diferentes variantes del ejercicio

2.3.1. Variante 1: Un sólo consumidor

Paso 1: Arrancar un productor por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-producer --broker-list localhost:9092,localhost:9093 --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--broker-list** : Establece la lista de brokers con la que trabajara
- Se le ha indicado que trabaje con el broker con puerto 9092
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")

Verificar que se encuentra a la espera de enviar mensajes.

Paso 2: Arrancar un consumidor por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--bootstrap-server** : Establece la dirección de los Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")
- **--from-beginning** : Mostrará el contenido del topic desde el inicio

Verificar que se encuentra a la espera de recibir los mensajes.

Paso 3: Enviar un mensaje del productor al consumidor

Escribir desde la consola del productor "Esto es una prueba 1 para la variante 1" y enviar.

Verificar que se muestra el mensaje desde la consola del consumidor

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic topic-basic-replication-test  
Esto es una prueba 1 para la variante 1
```

2.3.2. Variante 2: Dos consumidores CON lectura desde el inicio

Paso 1 : Arrancar un productor por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-producer --broker-list localhost:9092,localhost:9093 --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--broker-list** : Establece la lista de Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")

Verificar que se encuentra a la espera de enviar mensajes.

Escribimos desde la consola del productor "Esto es una prueba 1 para la variante 2" y enviar.

Paso 2 : Arrancar un consumidor 1 por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--bootstrap-server** : Establece la dirección de los Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")
- **--from-beginning** : Mostrará el contenido del topic desde el inicio de su contenido

Actualmente solo se han utilizado 2 de los 4 servidores establecidos.


```
~> kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic topic-basic-replication-test
Esto es una prueba 1 para la variante 1
Esto es una prueba 1 para la variante 2
█
```

Verificar que se muestran los mensajes (muestra los mensajes que se han enviado desde el principio del topic debido al parámetro `--from-beginning`) y que posteriormente queda a la escucha

- Se muestra "Esto es una prueba 1 para la variante 1" resultante de la anterior variante
- Y "Esto es una prueba 1 para la variante 2" resultante de usar su productor

Escribimos desde el productor el mensaje "Esto es una prueba 2 para la variante 2" y lo enviamos.

Verificamos que aparece este mensaje "Esto es una prueba 2 para la variante 2" en el consumidor.

```
~> kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic topic-basic-replication-test
Esto es una prueba 1 para la variante 1
Esto es una prueba 1 para la variante 2
Esto es una prueba 2 para la variante 2
█
```

Paso 3: Arrancar un consumidor 2 por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic
topic-basic-replication-test
```

Los parámetros utilizados son :

- **--bootstrap-server** : Establece la dirección de los Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")
- **--from-beginning** : Mostrará el contenido del topic desde el inicio

Actualmente sólo se han utilizado 2 de los 4 servidores establecidos.

```
~> kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic topic-basic-replication-test
Esto es una prueba 1 para la variante 1
Esto es una prueba 1 para la variante 2
Esto es una prueba 2 para la variante 2
█
```

Verificar que se muestran todos mensajes (muestra los mensajes que se han enviado desde el principio del topic debido al parámetro `--from-beginning`) y que posteriormente queda a la escucha.

Nota : Muestra los mismos mensajes que el consumidor 1



The image shows two terminal windows side-by-side. Both windows have the title 'kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093'. The top window is labeled '#3' and the bottom window is labeled '#4'. Both windows show the same output: 'kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --from-beginning --topic topic-basic-replication-test', followed by three lines of messages: 'Esto es una prueba 1 para la variante 1', 'Esto es una prueba 1 para la variante 2', and 'Esto es una prueba 2 para la variante 2'.

2.3.3. Variante 3: Dos consumidores SIN lectura desde el inicio

Paso 1: Arrancar un productor por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-producer --broker-list localhost:9092,localhost:9093 --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--broker-list** : Establece la lista de Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")

Verificar que se encuentra a la espera de enviar mensajes los mensajes.

Escribimos el mensaje "Esto es una prueba 1 para la variante 3" y lo enviamos.

Paso 2: Arrancar un consumidor 1 por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--bootstrap-server** : Establece la dirección de los Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")

Actualmente solo se han utilizado 2 de los 4 servidores establecidos.

Verificar que NO se muestra ningún mensaje.

Escribimos desde el productor el mensaje "Esto es una prueba 2 para la variante 3" y lo enviamos.

Verificamos que aparece sólo este mensaje "Esto es una prueba 2 para la variante 3" en el consumidor.

```
~ kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test
Esto es una prueba 2 para la variante 3
```

Paso 3: Arrancar un consumidor 2 por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--bootstrap-server** : Establece la dirección de los Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")

Actualmente sólo se han utilizado 2 de los 4 servidores establecidos

Verificar que NO se muestra ningún mensaje de los enviados anteriormente.

Escribimos desde el productor el mensaje "Esto es una prueba 3 para la variante 3" y lo enviamos.

Verificamos que aparece el mensaje "Esto es una prueba 3 para la variante 3" en los consumidores.

Consumidor 1

```
~ kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test
Esto es una prueba 2 para la variante 3
Esto es una prueba 3 para la variante 3
```

Consumidor 2

```
~ kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test
Esto es una prueba 3 para la variante 3
```

2.3.4. Variante 4: Dos consumidores CON el mismo grupo de consumo

Paso 1: Arrancar un productor por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-producer --broker-list localhost:9092,localhost:9093 --topic topic-basic-replication-test
```

Los parámetros utilizados son :

- **--broker-list** : Establece la lista de Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")

Verificar que se encuentra a la espera de enviar mensajes los mensajes.

Escribimos el mensaje "Esto es una prueba 1 para la variante 4" y lo enviamos.

Paso 2: Arrancar un consumidor 1 por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test --consumer-property group.id=myconsumergroup
```

Los parámetros utilizados son :

- **--bootstrap-server** : Establece la dirección de los Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-test")
- **--consumer-property group.id**: Se le ha configurado un grupo de consumo

Actualmente solo se han utilizado 2 de los 4 servidores establecidos.

Verificar que NO muestra ningún mensaje.

Paso 3: Arrancar un consumidor 2 por consola

Hay que ejecutar el siguiente comando :

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test --consumer-property group.id=myconsumergroup
```

Los parámetros utilizados son :

- **--bootstrap-server** : Establece la dirección de los Brokers con los que trabajara
- Se le ha indicado que trabaje con los brokers con puertos 9092 y 9093 (Replica y Leader respectivamente)
- **--topic** : Establece el nombre del topic (En este caso "topic-basic-replication-test")
- **--from-beginning** : Mostrará el contenido del topic desde el inicio

Actualmente solo se han utilizado 2 de los 4 servidores establecidos.

Verificar que NO muestra ningún mensaje.

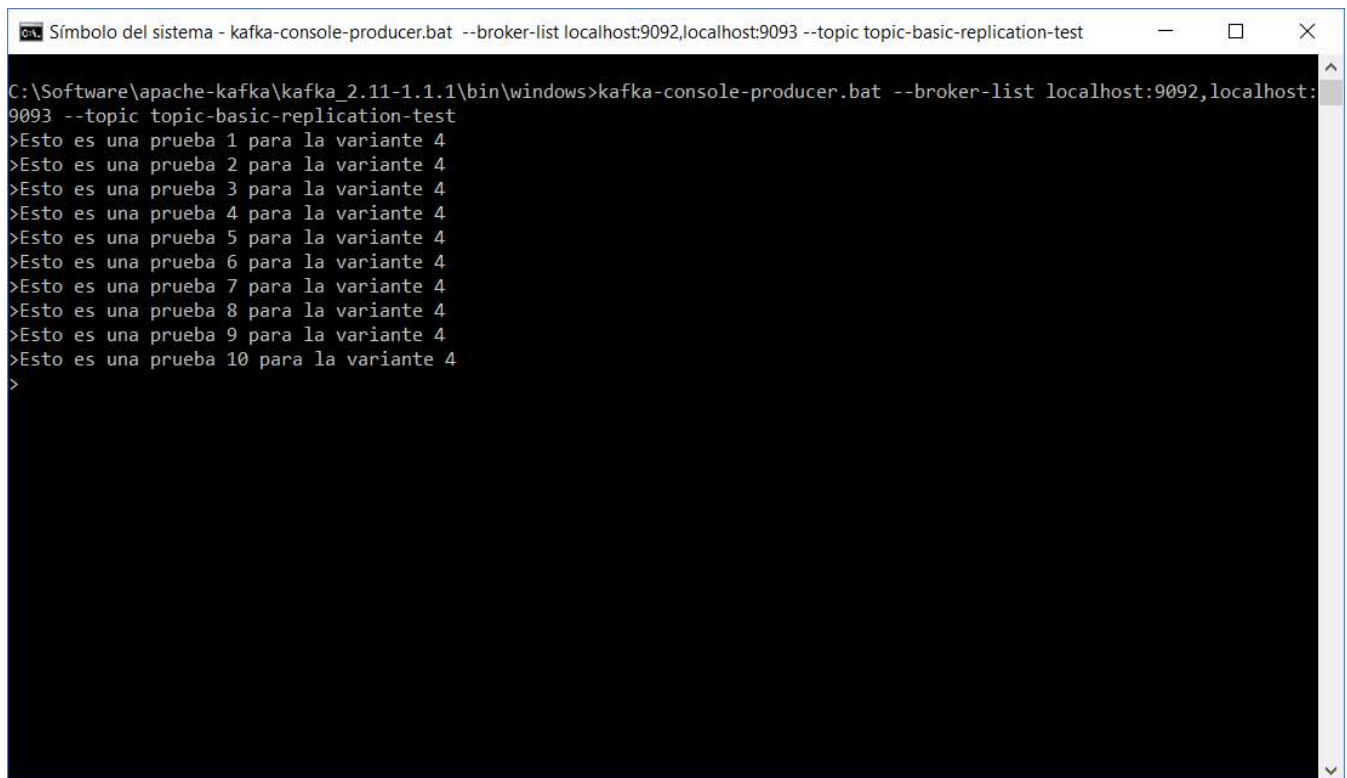
Paso 4: Escribir diferentes mensajes desde el productor

Escribimos el mensaje "Esto es una prueba 2 para la variante 4" y lo enviamos.

Escribimos el mensaje "Esto es una prueba 3 para la variante 4" y lo enviamos

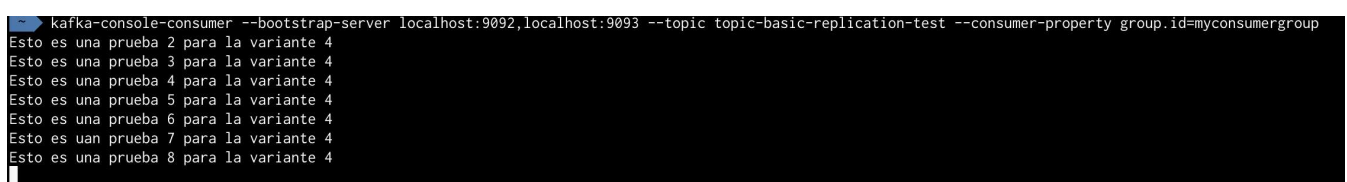
...

Escribimos el mensaje "Esto es una prueba 10 para la variante 4" y lo enviamos.



Verificamos que uno de los consumidores no se muestra nada y en el otro se muestran todos los mensajes.

En este caso el consumidor 1



En este caso el consumidor 2

```
kafka-console-consumer --bootstrap-server localhost:9092,localhost:9093 --topic topic-basic-replication-test --consumer-property group.id=myconsumergroup
```

Por lo tanto, se puede ver que un único consumidor del grupo esta asignado a escuchar desde la única partición leader.

2.3.5. Comprobar el funcionamiento de la replicación

En este apartado se va a detallar cómo se asigna automáticamente otro de los broker activos con partición réplica para contener la partición leader tras la pérdida del broker leader.

Paso 1: Verificamos el broker que contiene la partición leader

Para ello ejecutamos el comando **--describe** sobre el **topic**.

```
kafka-topics --describe --zookeeper localhost:2181 --topic topic-basic-replication-test
```

```
kafka-topics --describe --zookeeper localhost:2181 --topic topic-basic-replication-test
Topic: topic-basic-replication-test PartitionCount: 1 ReplicationFactor: 3 Configs:
Topic: topic-basic-replication-test Partition: 0 Leader: 1 Replicas: 1,2,3 Isr: 1,2,3
```

Se puede ver en la imagen que la partición leader se encuentra en el broker con id 1 y resto de ids se corresponden con brokers réplicas (2 y 3).

Paso 2: Paramos el broker que contiene la partición leader

Para ello paramos la ejecución desde la consola o bien con el comando **kill -9** para el caso de la instalación normal.

Para el caso de contenedores ejecutamos el comando

```
docker ps
```

```

CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
3996c4a54899   confluentinc/cp-kafka:5.5.0        "/etc/confluent/dock_  49 minutes ago Up 49 minutes 0.0.0.0:9095->9095/tcp, 0.0.0.0:29095->29095/tcp cp-zookeeper-550_kafka-4_1
37ef8eff3549   confluentinc/cp-kafka:5.5.0        "/etc/confluent/dock_  49 minutes ago Up 49 minutes 0.0.0.0:9092->9092/tcp, 0.0.0.0:29092->29092/tcp cp-zookeeper-550_kafka-1_1
880a4b428108   confluentinc/cp-kafka:5.5.0        "/etc/confluent/dock_  49 minutes ago Up 49 minutes 0.0.0.0:9093->9093/tcp, 9092/tcp, 0.0.0.0:29093->29093/tcp cp-zookeeper-550_kafka-2_1
60519d55ee60   confluentinc/cp-kafka:5.5.0        "/etc/confluent/dock_  49 minutes ago Up 49 minutes 0.0.0.0:9094->9094/tcp, 9092/tcp, 0.0.0.0:29094->29094/tcp cp-zookeeper-550_kafka-3_1
f96ea75a9953   confluentinc/cp-zookeeper:5.5.0    "/etc/confluent/dock_  49 minutes ago Up 49 minutes 2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp cp-zookeeper-550_zookeeper-1_1

```

Localizamos el contenedor correspondiente con el id 1 -> 9092

Para el caso de contenedores ejecutamos el comando

```
docker stop <CONTAINER_ID>
```

Paso 3: Verificamos la situación actual del topic

Para ello ejecutamos el comando --describe sobre el topic.

```
kafka-topics --describe --zookeeper localhost:2181 --topic topic-basic-replication-test
```

```
~/Workspace/personal/enmilocalfunciona-kafka/infrastructure/environment/custom > master • kafka-topics --describe --zookeeper localhost:2181 --topic topic-basic-replication-test
Topic: topic-basic-replication-test PartitionCount: 1 ReplicationFactor: 3 Configs:
Topic: topic-basic-replication-test Partition: 0 Leader: 2 Replicas: 1,2,3 Isr: 2,3
```

Se puede verificar que la nueva partición leader se encuentra en el broker con id 2 y que el valor del id 1 ha desaparecido del ISR.

Por lo tanto ya tenemos el primer ejemplo de replicación frente a un fallo.

Si volviéramos a realizar las variantes obtendríamos el mismo resultado pero el LEADER sería el nuevo BROKER.

3. Conclusiones

En este artículo hemos podido empezar a probar la plataforma de forma un poco más seria, con múltiples brokers y con diferentes situaciones. Además todo el ejemplo ha estado enfocado en explicar el concepto de replicación y cómo afecta durante el funcionamiento de un caso de uso "real" (la caída del broker leader durante su uso).

Todavía lo podemos complicar más pero de momento no ha estado nada mal para iniciarnos. Además hemos podido configurar un topic y habituarnos más en el uso de la herramienta proporcionada para su gestión.

Se puede decir que esto ya es un "hola mundo" de Kafka ;-).

Si te ha gustado, ¡síguenos en [Twitter](#) para estar al día de próximos posts!

Autor

VÍCTOR MADRID

Líder Técnico de la Comunidad de Arquitectura de Soluciones en atSistemas. Aprendiz de mucho y maestro de nada. Técnico, artista y polifacético a partes iguales ;-)

COMPARTE



ALSO ON EN MI LOCAL FUNCIONA

Observabilidad usando OpenTelemetry

hace 5 meses • 1 comentario

OpenTelemetry es el nuevo estándar de observabilidad avalado por la CNCF. ...

Crea tu carrusel de imágenes con CSS ...

hace un año • 7 comentarios

Como desarrollador habrás usado "slider" o carruseles en varias ocasiones, ...

Acelerando los desarrollos con ...

hace 10 meses

En este segundo artículo se va a mostrar cómo poder disponer de un ...

Re ef

hac

El i

nu

un

10 Comentarios

1

Acceder ▼

G

Únete a la conversación...

INICIAR SESIÓN CON

O REGISTRARSE CON DISQUS ?

Nombre

- ♡

Comparte

Mejores

Más nuevos

Más antiguos
- L

luis

hace un año

Alguien sabe como usar Kafka y logstash ayuuuuuda!

0

0

Responder • Comparte ›
- JOSE ELIAS SANABRIA MARTINEZ