□ Arquitectura de Apache Kafka: Explicación Básica

Es importante comprender un poco acerca de la arquitectura de **Apache Kafka** y entender, más o menos, cómo funciona por dentro. Si bien esta arquitectura puede parecer algo compleja al principio, haré mi mejor esfuerzo para explicar los conceptos de forma clara.

□ Arquitectura Productor-Consumidor

Lo primero que debemos entender es que la arquitectura de Apache Kafka está construida sobre el modelo **Productor-Consumidor**.

¿En qué consiste esta arquitectura?

- Un **productor** es quien genera o produce un **evento**.
- Un **consumidor** es quien recibe o consume ese evento.

El consumidor **siempre está en escucha activa**, observando si hay nuevos mensajes. En el momento que el productor genera un mensaje, el consumidor lo detecta inmediatamente y reacciona, realizando la acción correspondiente.

Esto permite una **comunicación asíncrona**, lo que significa que los consumidores no necesitan esperar de forma directa al productor: simplemente están atentos a los cambios o eventos nuevos.

□ Relación con Kafka

En **Apache Kafka**, implementamos este modelo usando componentes específicos para productores y consumidores.

Podemos tener múltiples productores y múltiples consumidores, sin ningún problema. Cada uno puede estar conectado a distintos flujos de información.

□ Ecosistema de Kafka

Para poder levantar un servidor de Kafka, necesitamos un servicio adicional llamado **Zookeeper**. Zookeeper viene incluido en los paquetes de Kafka, así que al descargar Kafka ya lo tenemos disponible. Solo debemos **inicializar Zookeeper antes de iniciar el servidor de Kafka**.

Estructura general de Kafka:

1. **Cluster**
   Un clúster de Kafka es el contenedor general que agrupa todos los elementos del ecosistema. Dentro del clúster, encontramos otros componentes clave.
2. **Brokers**
   Un **broker** es un nodo dentro del clúster que maneja los mensajes. Podemos tener **uno o más brokers** funcionando dentro de un mismo clúster. Estos brokers son responsables de recibir, almacenar y entregar los mensajes.
3. **Topics**
   Dentro de cada broker, encontramos los **topics**, que puedes imaginar como *temas de*

conversación. Cada topic es un canal lógico al que los productores envían mensajes y que los consumidores escuchan.

Ejemplos de topics:

- `confirmaciones`: mensajes relacionados con confirmaciones.
- `errores`: mensajes sobre errores o fallos del sistema.

4. **Particiones**
   Cada topic puede estar dividido en **particiones**. Estas particiones permiten que los mensajes se distribuyan internamente para optimizar el rendimiento y la escalabilidad del sistema. Las particiones almacenan físicamente los mensajes.
5. **Réplicas**
   Kafka permite tener **réplicas** de los mensajes, es decir, copias de seguridad. Por ejemplo, si tienes un mensaje en una partición de un broker, Kafka puede replicar ese mensaje en otro broker.

Esto garantiza **alta disponibilidad**: si un broker se cae, otra réplica puede tomar su lugar y mantener los mensajes disponibles. Kafka maneja esta replicación automáticamente, sin necesidad de intervención manual.

□ Flujo de trabajo típico

1. El **productor** envía un mensaje a un **topic** específico.
2. Los **consumidores** que están observando ese topic reciben el mensaje.
3. El consumidor reacciona al mensaje realizando la acción correspondiente.

□ Conclusión

La arquitectura de Kafka está basada en conceptos como el **modelo productor-consumidor**, **brokers**, **topics**, **particiones** y **réplicas**. Aunque puede parecer compleja al principio, al entender estos componentes fundamentales se vuelve mucho más clara.

Kafka es una plataforma poderosa para manejar flujos de datos en tiempo real, con gran rendimiento, tolerancia a fallos y escalabilidad.

□ Comandos Zookeeper y Kafka

Primero descargamos Apache Kafka (el comprimido):
https://kafka.apache.org/downloads

En la terminal debemos ubicarnos primero en el directorio de nuestra carpeta de Kafka.

▶□ Iniciar Zookeeper

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

▶☐ **Iniciar Kafka**

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

▶☐ **Crea un nuevo topic en el servidor de kafka (el puerto por defecto es 9092)**

```
./bin/kafka-topics.sh --create --topic {topic-name} --bootstrap-server {host}:9092

Otra forma (ejemplo):

./bin/kafka-topics.sh --create --topic topic-prueba --bootstrap-server localhost:9092 --partitions 3 --replication-factor 1
```

▶☐ **Decribir los detalles de un topic**

```
./bin/kafka-topics.sh --describe --topic {topic-name} --bootstrap-server {host}:9092
```

▶☐ **Listar todos los topics que existen dentro del broker**

```
./bin/kafka-topics.sh --list --bootstrap-server {host}:9092
```

▶☐ **Inicia una consola para ver mensajes de un topic específico (lo hace el cosumer)**

```
./bin/kafka-console-consumer.sh --topic {nombreTopic} --bootstrap-server {host}:9092
```

Otra forma (con ejemplo) para ver mensajes desde el inicio del topic:

```
./bin/kafka-console-consumer.sh --topic topic-prueba --bootstrap-server localhost:9092 --from-beginning
```

Otra forma (con un grupo de consumers):

```
./bin/kafka-console-consumer.sh --topic topic-prueba --bootstrap-server localhost:9092 --group my-group --from-beginning
```

Otra forma (en varias lineas):

```
./bin/kafka-console-consumer.sh \
  --bootstrap-server localhost:9092 \
  --topic topic-prueba \
  --from-beginning \
  --property print.key=true \
  --property print.partition=true
```

▶☐ **Inicia una consola para enviar mensajes a un topic específico (lo hace el producer)**

```
./bin/kafka-console-producer.sh --broker-list {host}:9092 --topic {topic-name}
```

Otra forma (con ejemplo) estableciendo que cada mensaje enviado tendrá el formato key:value, donde según la key se determinará a qué partición se enviará el mensaje y el value es el mensaje en sí:

```
./bin/kafka-console-producer.sh --broker-list localhost:9092 --
topic topic-prueba --property "parse.key=true" --property
"key.separator=:"
```

Para ver de un grupo de consumers cómo fueron consumiendo los mensajes del topic:

```
./bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 -
-describe --group my-group
```

 ¿Cómo se determina la partición?

Cuando produces un mensaje en Kafka con una clave (key), Kafka utiliza un particionador (partitioner) para decidir a qué partición enviar ese mensaje.

El particionador moderno de Kafka (desde Kafka 3.0+)

 Nombre:
DefaultPartitioner (org.apache.kafka.clients.producer.internals.DefaultPartitioner)

 ¿En qué se basa?
1. Si el mensaje tiene clave (key) → Usa el hash Murmur2
partition = murmur2(key) % numPartitions