PROGRAMACIÓN

Buceando profundamente en Kafka

Vivek Chaudhary Seguir
3 de agosto de 2020 · 6 min de lectura ★

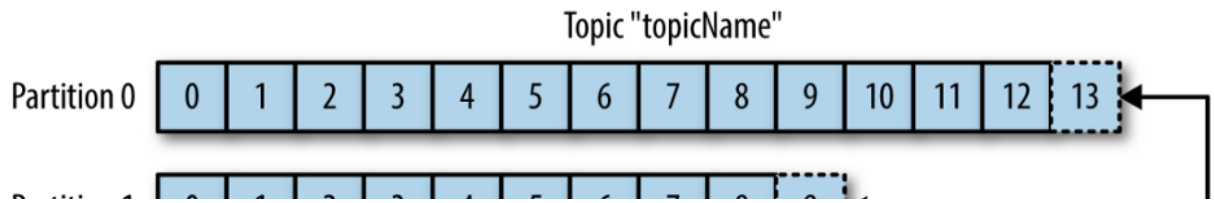
El objetivo de este blog es desarrollar una mayor comprensión de los conceptos de Apache Kafka, como Temas, Particiones, Consumidores y Grupos de consumidores. Los conceptos básicos de Kafka se han tratado en mi **artículo anterior** .

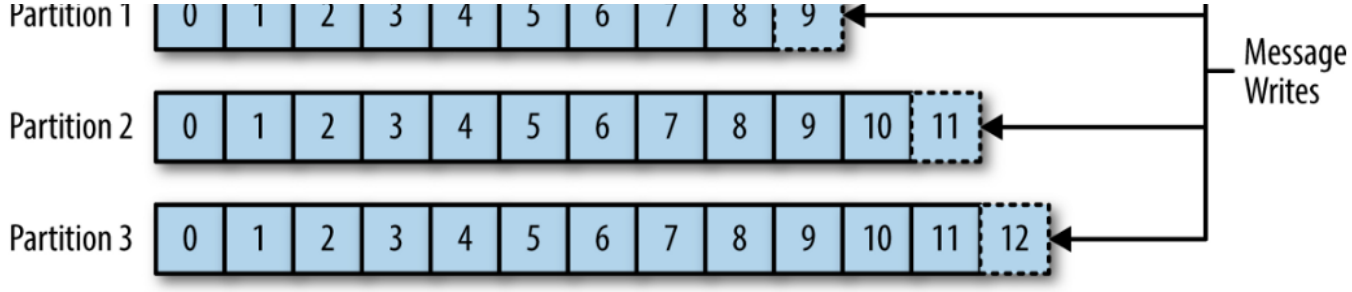
Temas y particiones de Kafka

Como sabemos, los mensajes en Kafka se clasifican o almacenan dentro de Temas. En términos simples, el tema se puede interpretar como una tabla de base de datos. El interior de Kafka Topics está dividido en particiones. Las particiones nos permiten paralelizar un tema al dividir los datos de un tema en múltiples corredores, agregando así una esencia de paralelismo al ecosistema.

Entre bastidores

Los mensajes se escriben en una partición de una manera de solo agregar y los mensajes se leen desde una partición de principio a fin, manierismo FIFO. Cada mensaje dentro de una partición se identifica mediante un valor entero llamado *desplazamiento*. Un *desplazamiento* es un orden secuencial inmutable de mensajes, mantenido por Kafka. Anatomía de un tema con múltiples particiones:





Tema particionado

> *El número secuencial en forma de matriz es el valor de compensación mantenido por Kafka*

Algunos puntos clave:

1. El orden de los mensajes se mantiene a nivel de partición, no a través del tema.
2. Los datos escritos en la partición son inmutables y no se pueden actualizar.
3. Cada mensaje en el agente de Kafka es una colección de temas de mensajes, partición, desplazamiento, clave y valor.
4. Cada partición tendrá un líder que se encargará de las operaciones de lectura / escritura en la partición.

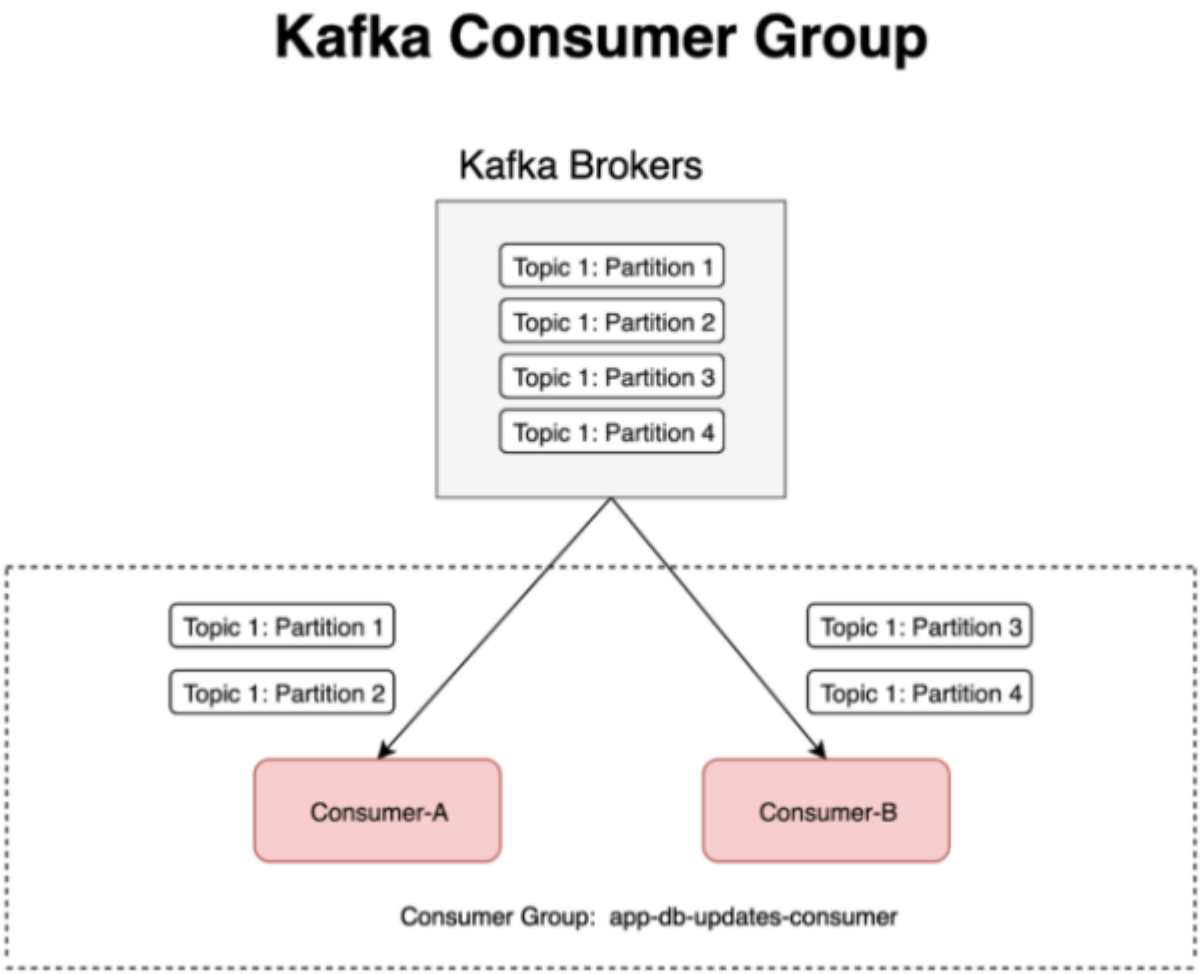
Productores y consumidores

Los productores crean nuevos mensajes y escriben los mismos a los corredores de Kafka, independientemente de la partición. Sin embargo, un productor puede escribir en una partición seleccionada con la ayuda de la clave del mensaje y la partición que generará un hash de clave y lo mapeará a la partición seleccionada.

Los consumidores leen mensajes de los corredores de Kafka. Un consumidor puede suscribirse a uno o más temas de Kafka y leer los mensajes en formato FIFO. Un consumidor realiza un seguimiento de los mensajes ya leídos con la ayuda de la compensación. Cada mensaje en una partición específica tiene un desplazamiento, que es un valor entero secuencial. Con la ayuda de la compensación, un consumidor puede detener o leer mensajes sin perder su posición.

Consumer trabaja como parte del **Consumer Group.** Un grupo de consumidores es básicamente una colección de uno o más consumidores que trabajan colectivamente en

paralelo para consumir mensajes de particiones de temas. De esta forma, se mejora la escalabilidad horizontal del consumo de mensajes. Además, si un consumidor falla, los otros miembros de los grupos de consumidores reequilibran la partición que se está consumiendo para reemplazar al miembro faltante.

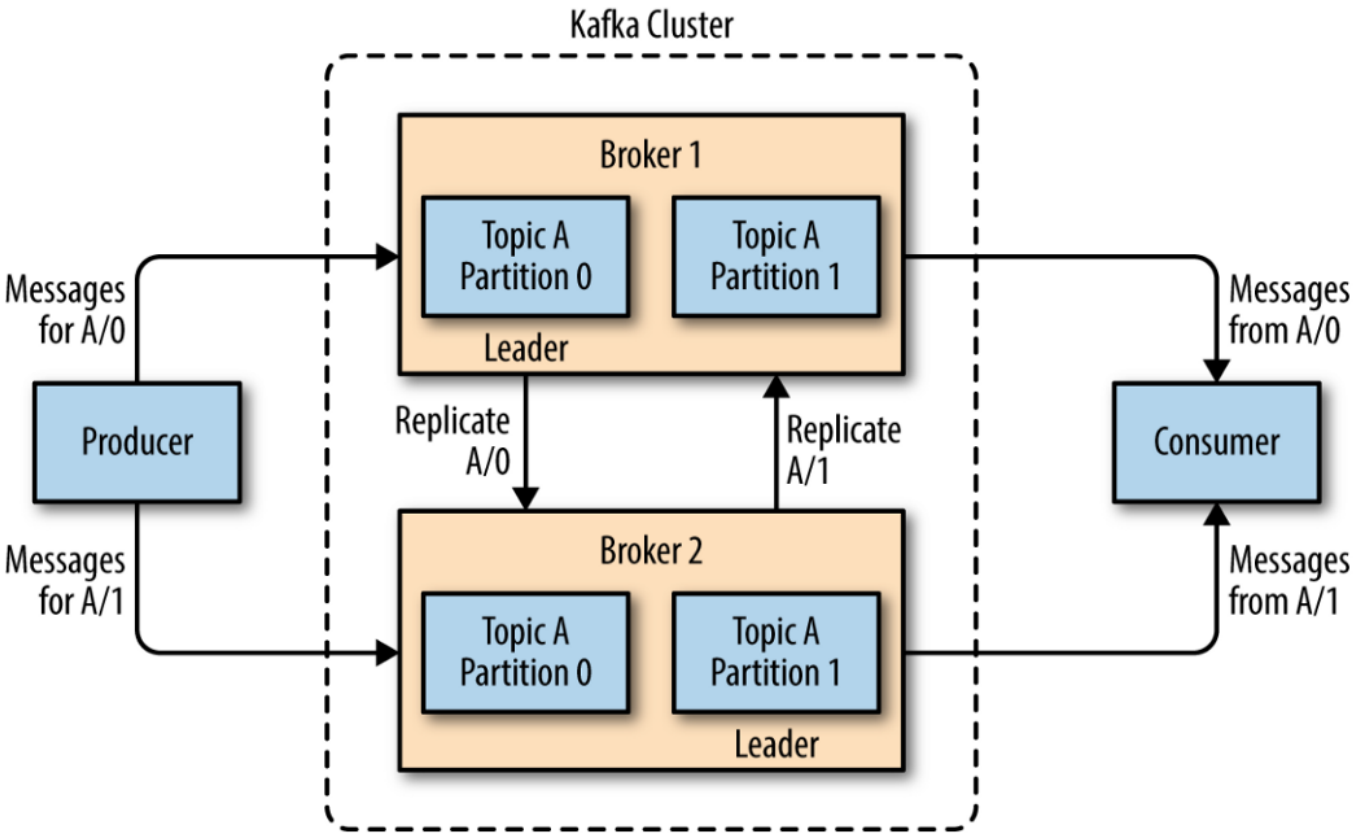


Corredores y particiones

Kafka Brokers opera dentro de un clúster. Dentro de un grupo de corredores, se elegirá automáticamente a un corredor como *controlador*. El controlador es responsable de las operaciones de administración, como la asignación de particiones al intermediario, la supervisión de fallas del intermediario, etc.

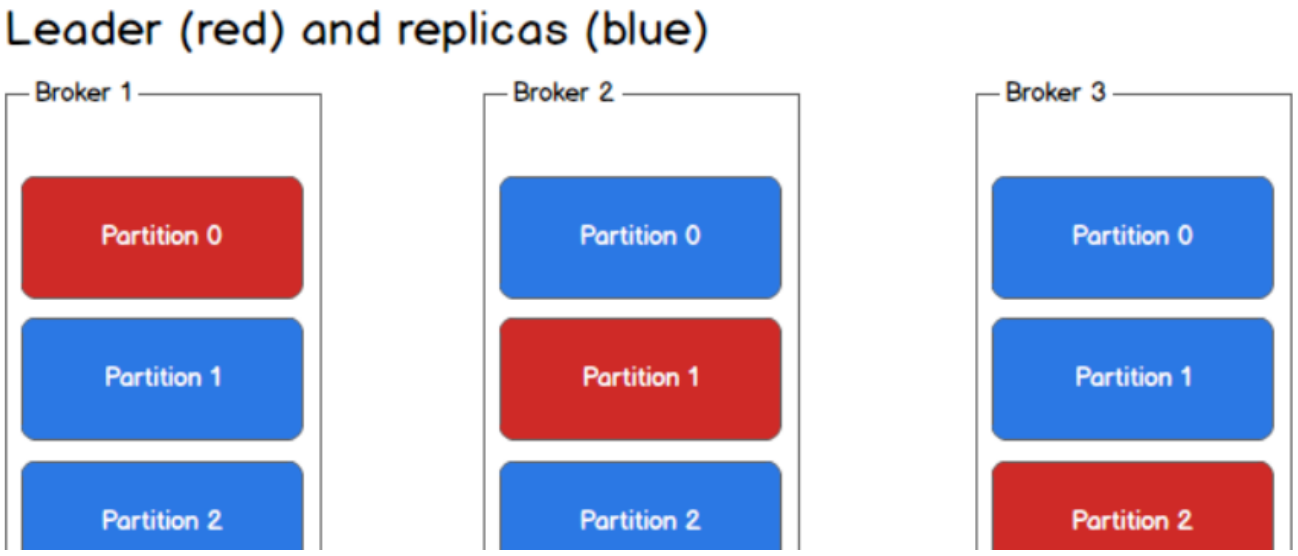
Cada corredor tiene varias particiones de tema y cada una de estas particiones puede ser un líder o una réplica para ese tema. Se asigna una partición a varios intermediarios, lo que da como resultado la replicación de la partición. Esto proporciona redundancia de

mensajes en una partición para que otro corredor asuma el liderazgo en caso de que el líder caiga.



Proceso de replicación

Todas las operaciones de lectura / escritura de un tema pasan por el líder del tema y, en consecuencia, el líder actualiza las réplicas con los nuevos mensajes. En caso de que un líder falle, una réplica se convierte en un nuevo líder.

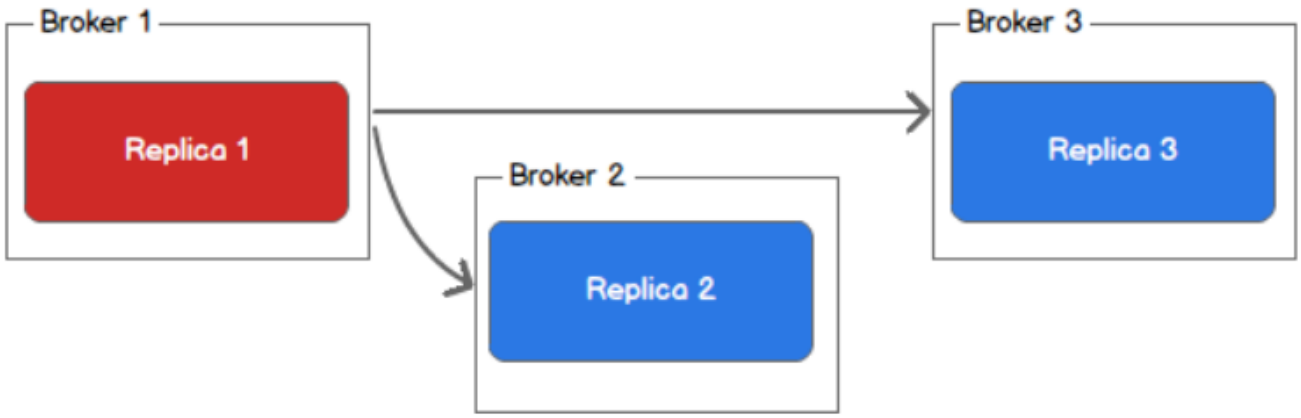


Líder y réplicas

Gestión de fallos

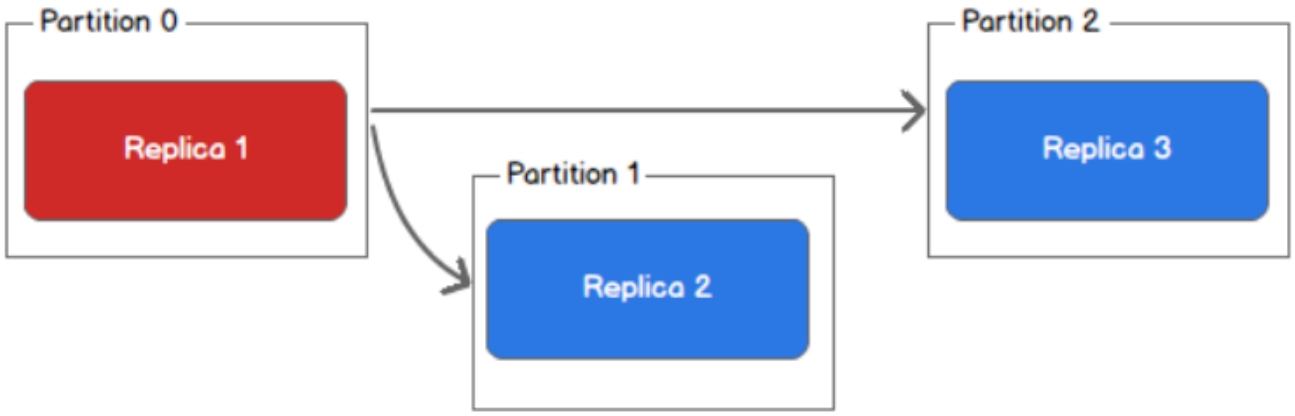
Como sabemos, los mensajes se envían al líder de la partición, y el líder es responsable de escribir mensajes en su réplica. Una vez actualizada, cada réplica reconoce al líder que ha recibido el mensaje y está sincronizado ahora.

Leader in Red escribe a **Replicas** in Blue



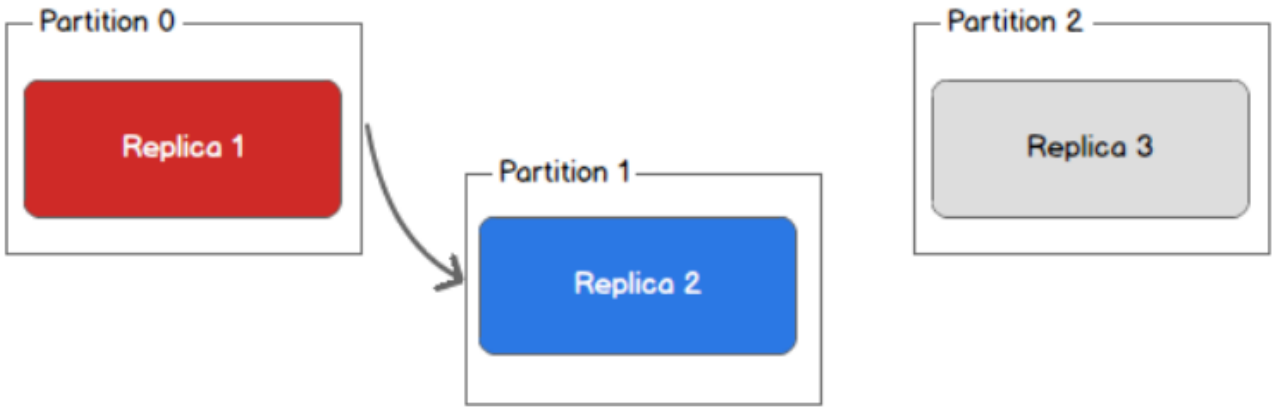
En un escenario ideal, cuando todos los corredores están disponibles, los productores y consumidores pueden escribir y leer del líder sin problemas. Pero ¿y si algo sale mal?

Un líder falla mientras aún existen réplicas sincronizadas. En tal escenario, el controlador de Kafka detectará la falla y elegirá un nuevo líder del grupo de réplicas. Este proceso lleva algo de tiempo y hasta que el error LeaderNotAvailable es visible.



Fallo del líder (rojo)

Todo parece controlable, pero ¿y si las réplicas caen? Los mensajes nuevos ya no se replicarán en una réplica fallida, por lo que se desincronizarán con el líder. Por ejemplo, la réplica 3 se cae, dejó de recibir mensajes y no estará sincronizada.

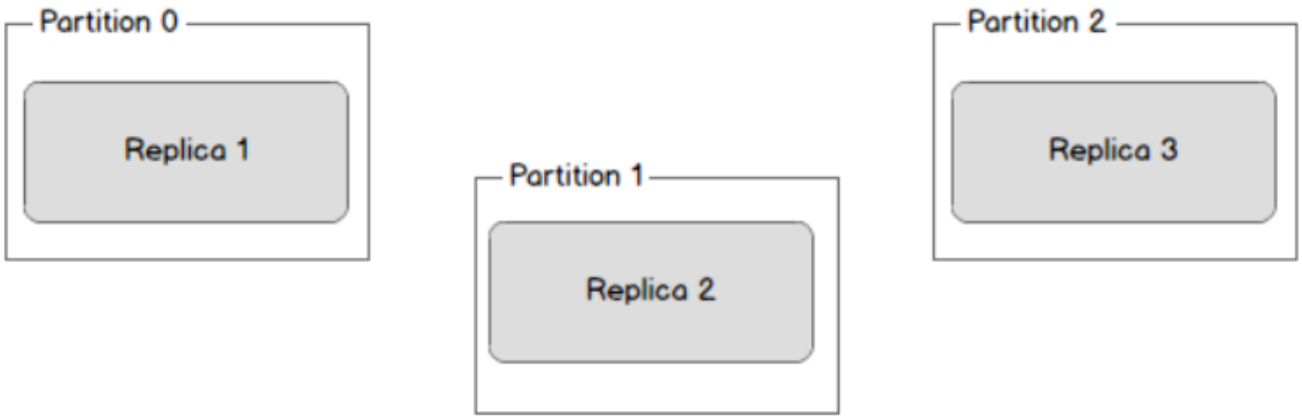


Réplica 3 abajo

Del mismo modo, cuando la Replica 2 se cae, dejará de recibir mensajes. En este momento, todavía tenemos una réplica en funcionamiento que es líder.

Réplica 2 abajo

Y luego Leader baja. Peor de los casos.



Todas las réplicas caídas

Dado este estado, puede haber dos posibles soluciones. Primero, espere hasta que el líder se levante y comience a recibir mensajes y todas las réplicas estén sincronizadas con el líder. En segundo lugar, seleccione un nuevo corredor como nuevo líder, pero este nuevo corredor no tendrá los datos hasta el punto en que el líder original cayó y los nuevos mensajes enviados hasta el momento en que el nuevo líder elegido se perderá, por lo que no estará sincronizado. sin embargo, tendrá nuevos mensajes una vez elegidos.

Coherencia del cliente de Kafka

Un consumidor puede leer solo los mensajes que se han escrito en todas las réplicas. Hay tres formas de aportar coherencia como consumidor:

1. **Como máximo una vez** : este escenario ocurre cuando se alcanza el intervalo de confirmación y Kafka tiene que confirmar la última compensación de lectura, pero el consumidor aún no ha completado el procesamiento de los mensajes y se bloquea en el medio. Entonces, cuando el consumidor se reinicia, recibirá mensajes del último desplazamiento comprometido y puede perder algunos mensajes.
2. **Al menos una vez:** este escenario ocurre cuando el consumidor procesa un mensaje y envía el mensaje a su tienda local y se bloquea en ese punto, pero Kafka no pudo

tener la oportunidad de enviar el desplazamiento al corredor porque el intervalo de confirmación no ha pasado. . Entonces, cuando el consumidor se reinicia, entregará los mensajes del último desplazamiento. La entrega de mensajes duplicados podría ocurrir en tal escenario. Prácticamente se sigue este camino.

3. **Exactamente una vez:** mientras procesa el mensaje, realiza un seguimiento de cada mensaje enviando el desplazamiento junto con un mensaje a un sistema transaccional. En caso de que un consumidor falle, leerá la última compensación comprometida del sistema transaccional y comenzará a leer después de ese punto. Esto no conduce a la duplicación de datos ni a la pérdida de datos, pero puede provocar una disminución del rendimiento.

Resumen:

El blog cubre algunos temas de Kafka en profundidad, particiones, proceso de replicación de mensajes, gestión de fallas, etc. Espero que el blog sea útil para desarrollar una mayor comprensión de Apache Kafka.

Gracias

Vivek Chaudhary

Python de Kafka Kafka Apache Kafka Transmisión