PALABRAS CLAVE:
Mensajería Asíncrona, Kafka para grandes volúmenes, Topic & Particiones, Suscripción, consumidor, Productor, colas.

1. **Diferencias entre mensajería síncrona y asíncrona**

   Mensajería **síncrona**: El **remitente espera** una **respuesta** inmediata **después** de **enviar** un **mensaje**. Ejemplo: llamadas **HTTP**.
   Mensajería **asíncrona**: El remitente **envía** un **mensaje sin esperar respuesta** inmediata; los **mensajes** se **procesan** de **forma independiente**.

2. **¿Qué es Kafka?**

   Es un **sistema** de **mensajería asíncrona** basado en el modelo **publicador-consumidor**, el **publicador envía** un mensaje, y el **consumidor** se **suscribe** para dicho mensaje

3. **¿Para qué se utiliza Kafka?**

   Se utiliza para **recopilar grandes volúmenes de datos**, realizar **análisis** en **tiempo real** y **procesar flujos** de datos en **tiempo real**. Está **diseñada** para **administrar** los **flujos** de datos de **varias fuentes** y **enviarlos** a **distintos usuarios**

4. **Ventajas de Kafka**

   - Es un **sistema robusto**
   - **Fácilmente escalable**, pues **añadiendo nodos** se puede ampliar
   - **Gestiona** con **facilidad grandes volúmenes** de datos
   - **Almacena datos en disco**, la información se **guarda** de forma **duradera**
   - **Eficiente**, pues usa **paralelismo**
   - **Fiable**, pues usa **replicación** en **todos** los **nodos** que usa

5. **¿Cuáles son los componentes principales de Kafka?**

   - **Tópico: Agrupación** de **registros**. Se **compone** de 1 o más **particiones**
   - **Partición: Secuencia ordenada** de **mensajes**, contiene un **identificador único** llamado **offset.**
   - **Segmentos: Subdivisiones** de **particiones** que **almacenan** datos de forma **duradera**.
   - **Intermediario (Nodo):** Aloja **tópicos** y **recibe mensajes** de producers, y **permite** a los consumers **obtener** dichos mensajes
   - **Zookeeper:** Coordina la **configuración** y **sincronización** entre **nodos**.
   - **Clúster: Agrupación** de **intermediarios**, que se **comunican** mediante **zookeepers**
   - **Productor:** Envía registros a los tópicos de manera asíncrona, y eligen a qué partición envían el mensaje
   - **Consumidor:** Lee registros de los tópicos, y **puede leer** en **cualquier punto** de una **partición** proporcionando el **offset**

- **Grupo de Consumidores:** Conjunto de consumidores que comparten la carga.

6. **¿Cómo distribuye Kafka los mensajes en las particiones?**

   Los datos se envían a una partición si se cumplen estas reglas:
   - El **producer** un nº de **partición**.
   - Si un registro no tiene ID de partición pero tiene **clave**, por lo que se **elige** la **partición** en base al **valor hash** de esa **clave**.
   - Si no se cumplen estas reglas, Kafka **asigna** la **partición** en base a una **estrategia** de **turnos**.

7. **¿Cuál es la función de las particiones en Kafka?**

   Al dividir los **topics** en particiones se pueden **distribuir** entre diferentes **brokers**, mejorando la **escalabilidad** y la **fiabilidad**, y permitiendo manejar datos en **paralelo**.

8. **¿Kafka garantiza el orden de los mensajes?**

   Sí, **dentro** de una **partición**, los **mensajes mantienen** el **orden** en que se **producen**.

9. **¿Los mensajes se almacenan indefinidamente?**

   No, se almacena según un retention time o hasta que alcance el tamaño configurado.

10. **¿Cómo se gestiona la información si un nodo falla?**

    Al escribir una partición, un intermediario o nodo se asocia esa partición y se convierte en líder de ella, mientras que los demás intermediarios son seguidores. Todos los nodos contienen copias de la partición, por lo que si el líder falla, se elige un seguidor como líder de esa partición, y se mantiene la partición.

11. **Diferencias entre Kafka y RabbitMQ**

    RabbitMQ está enfocado en la **mensajería tradicional por colas**, un **mensaje** se **consume** y luego **desaparece** (en principio). **Diseñado** para la **entrega rápida** y **confiable**.

    Kafka es **publicación-suscripción** con **topics** y **particiones**. Varios **consumidores** pueden **leer** los **mismos datos** sin **borrarlos**. Manejo de **flujos de datos grandes** y **continuos**.

1. **¿Qué es RabbitMQ?**

   Gestor de colas para enviar y recibir mensajes entre aplicaciones.

2. **¿Qué es una cola en RabbitMQ?**

   Un **buffer** donde los **mensajes** se **almacenan** hasta que un **consumidor** los **procesa**.

3. **¿Cuáles son los tipos de intercambios en RabbitMQ y cómo funcionan?**

   Los tipos de intercambios en RabbitMQ son:

   - **Directo (Direct):** Enruta mensajes a las colas cuya **clave de enlace coincide exactamente** con la **clave** de **enrutamiento** del **mensaje**.
   - **Fanout: Distribuye** los **mensajes** a **todas** las **colas vinculadas** al **intercambio**. Todos los **consumidores** que están **escuchando** en estas **colas recibirán** una **copia** del **mensaje**.
   - **Topic:** Permite el **enrutamiento flexible** mediante **patrones** y **comodines** en las **claves** de **enrutamiento**.
   - **Headers:** Enruta mensajes **basándose** en los **atributos** del **encabezado** del **mensaje**, **no** en la **clave de enrutamiento**.

4. **¿Qué es una clave de enrutamiento (routing key) y cómo se utiliza?**

   Una clave de enrutamiento es un **identificador** que el **intercambio utiliza** para **decidir** cómo **dirigir** un **mensaje** a las **colas correspondientes**. Es una especie de **"dirección del mensaje"** que ayuda a filtrar y enrutar mensajes basándose en reglas predefinidas.

5. **¿Qué es AMQP y cuál es su relación con RabbitMQ?**

   AMQP (**Advanced Message Queuing Protocol**) es un **protocolo** estándar para **mensajería asincrónica** que **utiliza RabbitMQ**. Este protocolo **define** cómo los mensajes se **transmiten** entre **productores** y **consumidores**, incluyendo la **gestión** de **conexiones**, **colas**, **tipo** de **intercambios** y la **clave** de **enrutamiento**.

6. **¿Qué sucede si un consumidor se desconecta mientras hay mensajes en la cola?**

   Si un consumidor se desconecta, los **mensajes permanecen** en la cola **hasta** que el **consumidor** se **reconecta** o hasta que **otro consumidor** disponible los **procesa**. **RabbitMQ** garantiza la **persistencia** de los mensajes **si** están **configurados** como "**duraderos**".

7. **¿En qué situaciones utilizarías Kafka o RabbitMQ?**

   Kafka es ideal para manejar **grandes volúmenes de datos**, mientras que **RabbitMQ** puede ser una gran opción en **procesamiento** de **pequeños paquetes** de **mensajes**