MODULO 08 - EJERCICIO 05-A

ALEXIS YURI M.

Diseña un flujo de ingesta de datos en modalidad streaming utilizando una plataforma de mensajería asíncrona (Kafka, Flink o Spark Streaming), orientado a resolver la problemática de monitoreo en tiempo real.

1. Objetivo del Flujo.

El objetivo del flujo es detectar patrones de fraude en las transacciones de los clientes en tiempo real para mitigar pérdidas financieras.

Eventos a capturar: Se capturarán todos los eventos de transacciones, incluyendo:

- ID de la transacción.

- ID del cliente.

- Monto.

- Fecha y hora.

- Origen de la transacción (IP, dispositivo, etc.).

- Ubicación geográfica.

Detecciones en tiempo real: El sistema debe ser capaz de detectar y alertar sobre:

- Transacciones que superan un umbral de monto establecido.

- Patrones de gasto inusuales para un cliente específico.

- Transacciones simultáneas en ubicaciones geográficas distantes.

2. Elección de la Herramienta Principal.

Se elegirá Apache Kafka como la herramienta principal ya que actúa como un broker de mensajes distribuido, lo que le permite manejar un alto volumen de eventos de transacciones con **baja latencia** y **alta disponibilidad**. Su arquitectura basada en particiones permite el procesamiento paralelo y su capacidad de persistencia de datos asegura que los eventos no se pierdan, incluso si los consumidores fallan.

3. Esquema General del Flujo.

Origen de datos: Los eventos de transacción son generados por la API de pagos, la aplicación móvil y otros sistemas bancarios y se envían a Kafka.

Plataforma de mensajería: Los eventos se envían a un topic específico de Kafka, por ejemplo, transacciones\_fintech. Las transacciones se particionan por el ID de cliente para asegurar que todos los eventos de un mismo cliente se procesen en el mismo orden.

Procesamiento: Un motor de procesamiento de streaming (como Apache Spark Streaming o Apache Flink) consume los mensajes de Kafka. En esta etapa se realizan las siguientes tareas:

- Limpieza y validación: Se verifica que los datos de la transacción sean válidos y tengan el formato correcto.

- Detección de patrones: Se aplican modelos para identificar comportamientos sospechosos (ej. múltiples transacciones en un corto período de tiempo).

Destino final: Los eventos procesados se envían a diferentes destinos, dependiendo del resultado:

- Un sistema de alertas en tiempo real (ejemplo: correo electrónico) para las transacciones "marcadas" como fraudulentas.

- Una base de datos operativa para el análisis y la visualización de datos históricos.

- Un dashboard para que el equipo de analistas de fraude pueda monitorear las transacciones en tiempo real.

4. Beneficios del Streaming frente al Procesamiento Batch.

Detección Inmediata de Fraude: La principal ventaja es la **baja latencia**. El procesamiento en tiempo real permite detectar patrones de fraude en milisegundos, a diferencia del batch que procesa los datos al final del día, por ejemplo.

Experiencia del Cliente: Permite tomar **acciones inmediatas**, como bloquear una transacción sospechosa o enviar una notificación al cliente en el momento, lo que mejora la seguridad y la experiencia del usuario.

5. Flujo Paso a Paso.

Generación de Eventos: Cada vez que un cliente realiza una transacción, un evento de datos se genera desde la API de pagos o la aplicación móvil.

Ingesta en Kafka: El evento es inmediatamente enviado a Apache Kafka por un productor de datos hacia el topic transacciones\_fintech.

Procesamiento: Un grupo de consumidores de Apache Spark Streaming lee los eventos desde el topic y los procesa en micro-lotes.

Reglas de Negocio: Se ejecutan validaciones y se aplican algoritmos para detectar anomalías o patrones de fraude.

Destino y Alerta: Si un evento es "marcado" como fraudulento, se envía a un sistema de alertas. En caso contrario, se carga en una base de datos para análisis histórico.

Visualización: El equipo de analistas visualiza los datos en un dashboard para monitorear el estado de las transacciones en tiempo real.