



| POSGRADOS |

Maestría en

Ingeniería de software

Asignatura:

Patrones de Integración Empresarial

Tarea Nro. 03:

Integración con Kafka

Autor(es):

Cabascango Garcia Amanda Elizabeth

Calo Catota Carlos Edison

Fuentes Espinoza Pablo Gustavo

Guaman Guaman Saul German

Guerra Campuzano Cesar Hugo

Rengel Rivera Mateo Santiago

Vela Moya Christian Eduardo

Apache Kafka

Apache Kafka es una plataforma de mensajería distribuida y basada en logs de eventos diseñada para manejar grandes volúmenes de datos en tiempo real. Fue desarrollada originalmente por LinkedIn y posteriormente donada a la Apache Software Foundation. Kafka es ampliamente utilizado en arquitecturas de microservicios, procesamiento de flujos de datos y sistemas de integración de eventos.

Arquitectura y Componentes Claves.

La arquitectura de Kafka se basa en los siguientes componentes fundamentales:

- **Producer (Productor)**
Los productores son las aplicaciones o sistemas que envían datos a Kafka. Estos pueden enviar mensajes a uno o múltiples topics dentro de un clúster de Kafka. Los productores pueden configurar estrategias de balanceo de carga, asegurando que los datos se distribuyan equitativamente entre las particiones de un topic.
- **Broker (Intermediario)**
Los brokers son los servidores que conforman un clúster de Kafka y se encargan de almacenar y gestionar los mensajes. Cada broker maneja una o varias particiones de un topic y permite la distribución de carga y replicación de datos. Un clúster de Kafka generalmente está compuesto por múltiples brokers para garantizar disponibilidad y tolerancia a fallos.
- **Consumer (Consumidor)**
Los consumidores son aplicaciones que leen los mensajes desde Kafka. Estos consumidores pueden agruparse en grupos de consumidores, donde cada uno lee mensajes de distintas particiones para garantizar procesamiento paralelo y balanceo de la información.
- **Topic (Tópico)**
Un topic es una categoría o canal donde se publican los mensajes. Los consumidores se suscriben a los topics para recibir los eventos publicados por los productores. Cada topic puede dividirse en varias particiones para mejorar la escalabilidad.
- **Partitions (Particiones)**
Las particiones permiten dividir los topics en segmentos distribuidos entre los brokers. Esto permite escalabilidad horizontal, ya que cada partición puede ser procesada de manera independiente por distintos consumidores dentro de un grupo.

- **Zookeeper**

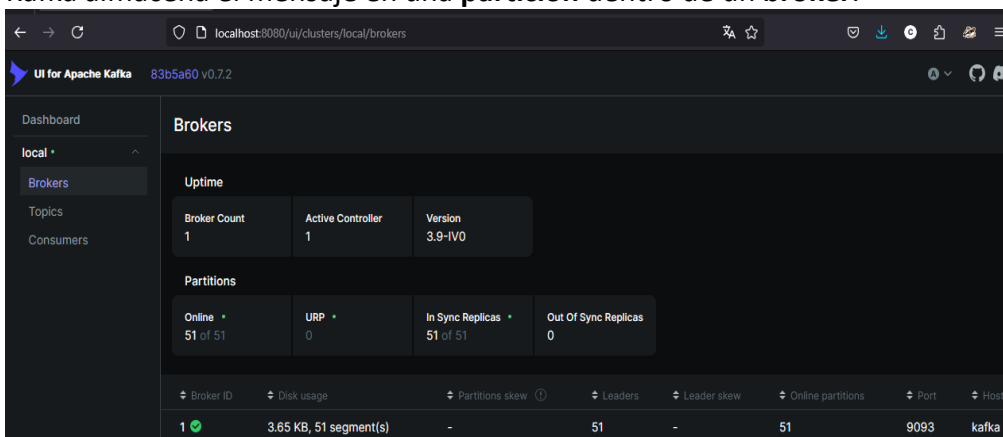
Apache Zookeeper es un servicio utilizado por Kafka para la gestión de metadatos, como el control de líderes de particiones y la configuración del clúster.

Flujo de Trabajo en Kafka

1. Un **productor** publica un mensaje en un **topic**.

```
PS C:\Pruebas UPS\T3-Kafka-NetCore\KafkaProducer> dotnet run
Escribe un mensaje para enviarlo a Kafka (o 'salir' para terminar):
> Hola desde producer..
```

2. Kafka almacena el mensaje en una **partición** dentro de un **broker**.



Uptime			
Broker Count	Active Controller	Version	
1	1	3.9-IV0	

Partitions			
Online	URP	In Sync Replicas	Out Of Sync Replicas
51 of 51	0	51 of 51	0

Broker ID	Disk usage	Partitions skew	Leaders	Leader skew	Online partitions	Port	Host
1	3.65 KB, 51 segment(s)	-	51	-	51	9093	kafka

3. Un **consumidor** lee el mensaje desde la partición.

```
PS C:\Pruebas UPS\T3-Kafka-NetCore\KafkaConsumer> dotnet run
Escuchando mensajes de test-topic...
Mensaje recibido: Hola desde producer..
```

4. Kafka mantiene el mensaje por un tiempo determinado, permitiendo relectura si es necesario.

Casos de Uso en Integración Empresarial

Kafka se usa en una variedad de aplicaciones empresariales, incluyendo:

- Procesamiento en tiempo real: Sistemas de monitoreo, detección de fraudes, procesamiento de logs.
- Integración de microservicios: Comunicación entre microservicios mediante eventos.
- Análisis de datos y ETL: Ingesta de datos para lagos de datos y procesamiento en tiempo real.
- Mensajería y notificaciones: Servicios de notificación en grandes plataformas.

Comparación con RabbitMQ

Característica	Apache Kafka	RabbitMQ
Modelo de mensajería	Basado en logs distribuidos, optimizado para transmisión de eventos	Basado en colas de mensajes, diseñado para enrutamiento y procesamiento de tareas
Persistencia y rendimiento	Almacena mensajes en disco con replicación, alta velocidad de lectura	Mensajes en memoria por defecto, permite persistencia, pero con menor rendimiento
Escalabilidad	Escalabilidad horizontal mediante particiones distribuidas en brokers	Escalabilidad con clustering y federación, pero con más configuraciones
Casos de uso	Procesamiento de datos en tiempo real, streaming de eventos, big data	Colas de mensajes para transacciones empresariales y procesamiento distribuido
Orden de entrega	Garantizado dentro de cada partición	Garantizado por cola de mensajes
Tiempo de retención	Configurable, puede mantener mensajes por tiempo indefinido	Mensajes eliminados tras ser consumidos, a menos que se configuren colas durables
Latencia	Baja latencia para transmisión continua	Optimizado para baja latencia en tareas transaccionales
Complejidad de configuración	Más compleja, requiere configuración de particiones y Zookeeper	Más simple de configurar e implementar

Repositorio

Comparto la [URL](#) de un repositorio en GitHub que contiene un ejemplo práctico en .NET Core, donde se implementan servicios de productor y consumidor para trabajar con Kafka dockerizado. El repositorio incluye un archivo README con instrucciones detalladas para ejecutar el ejercicio.

Conclusión

Apache Kafka es una solución robusta y escalable para la transmisión de eventos en tiempo real, superando a otras tecnologías como RabbitMQ en escenarios de alto volumen de datos. Su modelo basado en logs lo hace ideal para sistemas distribuidos y microservicios.

Referencias

Apache Kafka: <https://kafka.apache.org/>

Confluent Kafka: <https://www.confluent.io/>

Apache Zookeeper: <https://zookeeper.apache.org/>

RabbitMQ: <https://www.rabbitmq.com/>