Kafka (Serviço de messageria )

Explicação entendimento mensageria Aplicação orientada a evento

<https://www.youtube.com/watch?v=axJ_aHjyxQ8&list=PLTN1gMq8EHuIhoTzoDkwIXM2mgQQaYI6p>

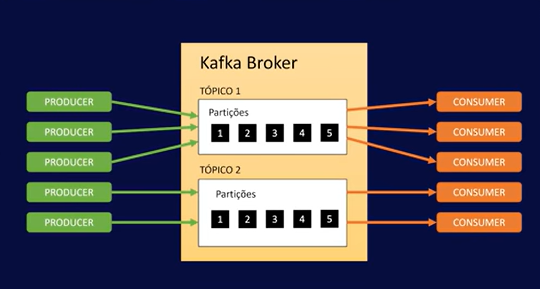
Pode ser feito de varias formas com varias API

Instalando o Kafka no Docker

<https://www.youtube.com/watch?v=ta8F4vy8SZI&list=PLTN1gMq8EHuIhoTzoDkwIXM2mgQQaYI6p&index=2>

Funcionamento e componentes do Kafka

<https://www.youtube.com/watch?v=uu6F6yvCXzc>



Producer = manda as msg

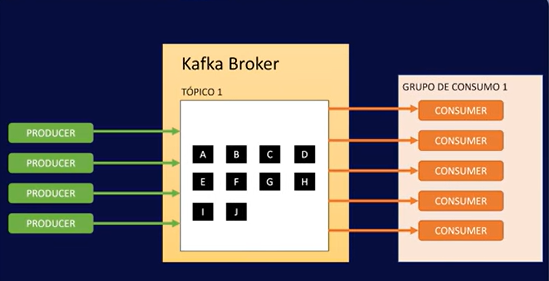
Consumer = pega as mensagens faz a leitura das msg informa para o Kafka que as msm foram processadas caso não informe vai ficar recebendo a msm msg sempre ate que informe registro é chamado de (ofSet)

Kafka = (chamado de Broker de msg) responsável pela inteligência e controle recebimento

Partições = A quantidade de partições define a quantidade de consumidores podem acessar e efetuar a leitura das msm tempo

Grupo de consumo = Kafka entrega a msm uma única vez para cada grupo de consumo

Producer envia a msg A,B,C,....



Consumer receber mais nenhum recebera a msg repetida conceito de fila

Diagrama

Descrição gerada automaticamente com confiança média

Quando vc inclui mais de 1 grupo de consumo muda o conceito de fila para tópico todos recebam a msm mensagem

Ex: criado 3 grupos de consumo

Diagrama

Descrição gerada automaticamente

Entrega do Kafka

Diagrama

Descrição gerada automaticamente

Exemplo pratico Cartão de credito

Producer = Usuario que passa o cartão e é enviado as informações (numero cartão, cod estabelecimento, valor)

Grupos Consumo = seria as aplicações que vai receber estas msg

grupo1(app que vai atualizar saldo do cartão )

grupo2(app que vai incluir o debito na fatura do cartão )

grupo3(app que vai registrar um credito para estabelecimento )

Partições mais de 1 consumer buscando as msm vc não garante a entrega em ordem fila

Zookeeper = gerenciador dos tópicos , ofSet , partições que estão vazias , quem pode ler a msg

Outra visão do Kafka

Diagrama

Descrição gerada automaticamente

Forma simples listeners busca informações em uma API e envia para outra API

Interface gráfica do usuário, Diagrama

Descrição gerada automaticamente

Kafka faz os armazenamentos em diferentes posições e Consumidor que é responsável para pegar de forma que o consumidor pega da posição 0 em diante caso ocorra algum erro ele consegue voltar, Kafka não tem nenhuma responsabilidade.

Diagrama, Tabela

Descrição gerada automaticamente

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

O **Apache Kafka** é uma plataforma distribuída de streaming de dados e mensageria, projetada para lidar com grandes volumes de dados em tempo real. Originalmente desenvolvido pela **LinkedIn** e agora mantido pela **Apache Software Foundation**, o Kafka tem ganhado popularidade por sua alta performance, escalabilidade e robustez em sistemas distribuídos.

**O que é o Apache Kafka?**

Apache Kafka é um **sistema de mensageria** que permite o **processamento de fluxos de dados em tempo real**. Ele é especialmente útil quando há a necessidade de processar, armazenar e mover grandes volumes de dados em tempo real, com baixa latência e alta disponibilidade.

**Conceitos Principais**

**1. Produtores (Producers)**

São os componentes responsáveis por enviar (ou publicar) dados para o Kafka. Um produtor envia mensagens para tópicos específicos, podendo ser qualquer aplicação que deseje enviar eventos ou dados para o Kafka. O produtor pode configurar a partição de um tópico ou deixar que o Kafka escolha.

**2. Consumidores (Consumers)**

São os componentes que **leem** as mensagens de um tópico. Um consumidor pode se inscrever em um ou mais tópicos e processar as mensagens à medida que são produzidas. O Kafka suporta a leitura de mensagens em paralelo, com consumidores distribuídos e agrupados em **grupos de consumidores**.

**3. Tópicos (Topics)**

No Kafka, as mensagens são categorizadas por tópicos. Cada **tópico** pode ter múltiplas partições, que distribuem as mensagens entre os nós do Kafka. Um tópico pode ser comparado a uma fila de mensagens em sistemas de mensageria tradicionais, mas com suporte à **alta paralelização**.

**4. Partições (Partitions)**

Cada tópico pode ser dividido em várias **partições**, que são unidades físicas de armazenamento dentro do Kafka. As partições ajudam a distribuir dados em vários brokers, permitindo **escalabilidade horizontal**. Cada partição tem uma ordem sequencial de mensagens.

**5. Brokers**

Um **broker** Kafka é um servidor que armazena dados e serve como ponto de comunicação para produtores e consumidores. O Kafka pode ser configurado para ter múltiplos brokers, o que aumenta sua **disponibilidade** e **escalabilidade**. Um **cluster** de Kafka é composto por vários brokers.

**6. Zookeeper**

O **Zookeeper** é utilizado pelo Kafka para **gerenciamento de clusters** e para garantir a **coerência de dados** entre os brokers. Embora o Kafka tenha começado a depender do Zookeeper, versões mais recentes do Kafka estão trabalhando para remover essa dependência, com o Kafka KRaft, uma nova arquitetura para gerenciamento de clusters sem o Zookeeper.

**7. Offsets**

Cada mensagem dentro de uma partição Kafka tem um **offset**, que é um número único que identifica a posição da mensagem dentro da partição. O consumidor lê mensagens em uma sequência ordenada, usando esse offset.

**Características e Vantagens do Apache Kafka**

* **Alta Escalabilidade**: O Kafka pode ser escalado facilmente para lidar com grandes volumes de dados. Ele é projetado para lidar com **milhões de mensagens por segundo** em grandes clusters distribuídos.
* **Durabilidade**: O Kafka armazena mensagens de forma persistente em disco e pode replicá-las para aumentar a resiliência e garantir a **durabilidade dos dados**.
* **Alta Disponibilidade**: Com a replicação de dados e a possibilidade de adicionar brokers de forma dinâmica, o Kafka é altamente disponível, mesmo em caso de falhas em brokers.
* **Desempenho em Tempo Real**: O Kafka é otimizado para entregar mensagens com **baixa latência**, ideal para processamento de dados em tempo real.
* **Processamento de Fluxos (Stream Processing)**: Kafka oferece suporte a **streaming de dados** com APIs como **Kafka Streams** e **ksqlDB**, que permitem processar dados diretamente no Kafka sem a necessidade de extrair para sistemas externos.
* **Tolerância a Falhas**: As mensagens no Kafka são replicadas em múltiplos brokers, garantindo que, em caso de falha de um broker, os dados ainda estarão disponíveis.
* **Desacoplamento**: O Kafka atua como um intermediário entre os produtores e consumidores de dados, permitindo um desacoplamento eficiente entre os componentes do sistema.

**Casos de Uso do Apache Kafka**

1. **Integração de Sistemas Distribuídos** O Kafka é frequentemente usado em **microserviços** e arquiteturas distribuídas para integrar diferentes serviços de forma eficiente. Em sistemas de microserviços, pode atuar como a espinha dorsal da comunicação entre os serviços.
2. **Processamento de Eventos em Tempo Real** Kafka é utilizado em **sistemas de eventos** e **arquiteturas orientadas a eventos**. Sistemas que precisam de **análise em tempo real**, como sistemas de **monitoramento**, **sistemas de recomendação** e **gestão de logs**, podem tirar proveito do Kafka.
3. **Monitoramento e Logs** É uma excelente solução para **coleta e processamento de logs**. Por exemplo, você pode enviar logs de sistemas distribuídos para o Kafka e usar consumidores para processá-los, analisá-los e gerar alertas ou relatórios em tempo real.
4. **Análise e Processamento de Big Data** Kafka é amplamente utilizado em pipelines de **big data**, alimentando sistemas como **Apache Spark**, **Apache Flink** ou **Apache Hadoop** para processamento em tempo real de grandes volumes de dados.
5. **Manejo de Fluxos de Dados em Tempo Real** Para sistemas que lidam com **dados em tempo real**, como sistemas de **finanças** (transações financeiras), **e-commerce** (eventos de compras, carrinhos de compra) ou **IoT** (fluxo de dados de dispositivos), o Kafka serve como um **mensageiro de dados**.

**Como Funciona o Apache Kafka em Termos Práticos**

1. **Configuração Básica de Produtores e Consumidores**:
   * O produtor envia mensagens para um **tópico**.
   * O consumidor lê as mensagens desse tópico.
   * As mensagens podem ser lidas em diferentes consumidores de forma independente e paralela, sem interferência.
2. **Garantias de Entrega**:
   * **Entrega Pelo Menos Uma Vez** (Atualmente, as mensagens são armazenadas e o consumidor pode ler a partir do **offset**).
   * **Entrega Exata Uma Vez** (Com configuração adequada, para garantir que uma mensagem não seja processada mais de uma vez).

**Como Integrar o Kafka com Java (Exemplo com Spring Boot)**

No Spring Boot, a integração com o Kafka pode ser realizada de maneira simples usando a dependência spring-kafka.

**1. Dependências Maven**

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

</dependency>

1. Configuração do Producer (Enviando Mensagens)
2. 

@EnableKafka

@Configuration

public class KafkaProducerConfig {

@Bean

public ProducerFactory<String, String> producerFactory() {

return new DefaultKafkaProducerFactory<>(producerConfigs());

}

@Bean

public KafkaTemplate<String, String> kafkaTemplate() {

return new KafkaTemplate<>(producerFactory());

}

private Map<String, Object> producerConfigs() {

Map<String, Object> config = new HashMap<>();

config.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092");

config.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class);

config.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class);

return config;

}

}

1. Produzindo Mensagens

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

@Service

public class KafkaProducerService {

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

public void sendMessage(String message) {

kafkaTemplate.send("topic-name", message);

}

}

1. Consumindo Mensagens

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

@Service

public class KafkaConsumerService {

@KafkaListener(topics = "topic-name", groupId = "group1")

public void listen(String message) {

System.out.println("Received message: " + message);

}

}

**Conclusão**

O Apache Kafka é uma ferramenta poderosa para sistemas que requerem **processamento de dados em tempo real**, **mensageria distribuída** ou **integração de sistemas complexos**. Ele permite escalabilidade horizontal, alta disponibilidade, persistência e processamento eficiente de grandes volumes de dados. Usando Kafka, você pode criar sistemas resilientes e altamente escaláveis que lidam com grandes fluxos de dados e eventos de maneira eficiente.