

Apache Kafka com Java - Gerenciamento de Filas.

**Parte I: Fundamentos do Apache Kafka**

**[[](https://www.linkedin.com/in/chmulato/)](https://www.linkedin.com/in/chmulato/)**

**[Christian Mulato](https://www.linkedin.com/in/chmulato/)**

Desenvolvedor Java Sênior | Especialista em Back-end | Jakarta, Spring Boot, REST APIs, Docker | Engenheiro Químico

30 de junho de 2025

**Introdução**

Esta parte apresenta os conceitos essenciais do Apache Kafka, sua arquitetura, principais componentes e comandos básicos para quem está começando.

**O que é Apache Kafka?**

Apache Kafka é uma plataforma distribuída de streaming de eventos, projetada para alta performance, escalabilidade e tolerância a falhas. É amplamente utilizada para processamento de dados em tempo real, integração entre sistemas e pipelines de dados.

**Conceitos Principais**

* Broker: Servidor Kafka responsável por armazenar e entregar mensagens.
* Topic: Canal lógico onde as mensagens são publicadas e consumidas.
* Partition: Subdivisão de um tópico para escalabilidade e paralelismo.
* Producer: Aplicação que envia mensagens para o Kafka.
* Consumer: Aplicação que lê mensagens do Kafka.
* Consumer Group: Grupo de consumidores que compartilham a leitura de partições.
* Offset: Posição sequencial de uma mensagem dentro de uma partição.

**Arquitetura Básica**

1. Producers publicam mensagens em tópicos.
2. Brokers armazenam as mensagens.
3. Consumers leem as mensagens dos tópicos.
4. O Kafka garante alta disponibilidade e escalabilidade por meio de partições e replicação.

**Instalação Rápida com Docker**

docker-compose up -d

**Comandos Essenciais**

* Criar um tópico:

kafka-topics --create --topic meu-topico --bootstrap-server localhost:9092 --partitions 3 --replication-factor 1

* Produzir mensagens:

kafka-console-producer --topic meu-topico --bootstrap-server localhost:9092

* Consumir mensagens:

kafka-console-consumer --topic meu-topico --from-beginning --bootstrap-server localhost:9092

**Exercícios Práticos**

1. Suba o ambiente Kafka localmente.
2. Crie tópicos com diferentes números de partições.
3. Produza e consuma mensagens usando o terminal.
4. Experimente criar múltiplos consumidores em um mesmo grupo.

**Recursos Recomendados**

* [**Documentação Oficial do Apache Kafka**](https://kafka.apache.org/documentation/)
* Livro: Kafka: The Definitive Guide (O'Reilly)

**Exemplo Java: Producer e Consumer Simples**

A seguir, você encontra exemplos didáticos de um Producer e um Consumer em Java, ideais para quem está começando a experimentar o Apache Kafka na prática. Os arquivos completos estão disponíveis em: parte1-fundamentos/src/main/java/SimpleProducer.java e parte1-fundamentos/src/main/java/SimpleConsumer.java.

**Como executar os exemplos**

1. **Garanta que o Kafka está rodando em** localhost:9092

Utilize o docker-compose.yml fornecido na pasta parte1-fundamentos/ para subir o ambiente local rapidamente:

docker-compose up -d

2. **Crie o tópico** meu-topico se necessário

Execute o comando abaixo para criar o tópico no seu cluster Kafka local:

docker exec -it <nome\_do\_container\_kafka> kafka-topics --bootstrap-server localhost:9092 --create --topic meu-topico --partitions 1 --replication-factor 1

Substitua <nome\_do\_container\_kafka> pelo nome real do container Kafka em execução (ex: kafka ou kafka1).

3. **Compile e execute os exemplos Java usando Maven**

O projeto já possui um pom.xml pronto na pasta parte1-fundamentos com todas as dependências necessárias. Basta rodar:

mvn compile

mvn exec:java -Dexec.mainClass=SimpleProducer

mvn exec:java -Dexec.mainClass=SimpleConsumer

O **SimpleProducer** envia uma mensagem de exemplo para o tópico, e o **SimpleConsumer** consome e imprime as mensagens recebidas.

**Producer Java — Enviando uma mensagem**

O Producer é responsável por publicar mensagens em um tópico Kafka. Veja um exemplo básico:

import org.apache.kafka.clients.producer.KafkaProducer;

import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;

public class SimpleProducer {

public static void main(String[] args) {

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092");

props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");

props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

try (KafkaProducer<String, String> producer = new KafkaProducer<>(props)) {

producer.send(new ProducerRecord<>("meu-topico", "mensagem de exemplo"));

System.out.println("Mensagem enviada!");

}

}

}

**Consumer Java — Lendo mensagens do tópico**

O Consumer é responsável por ler as mensagens publicadas em um tópico. Veja um exemplo básico:

import org.apache.kafka.clients.consumer.ConsumerRecords;

import org.apache.kafka.clients.consumer.KafkaConsumer;

import org.apache.kafka.clients.consumer.ConsumerRecord;

import java.util.Collections;

import java.util.Properties;

public class SimpleConsumer {

public static void main(String[] args) {

Properties props = new Properties();

props.put("bootstrap.servers", "localhost:9092");

props.put("group.id", "grupo-exemplo");

props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

try (KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props)) {

consumer.subscribe(Collections.singletonList("meu-topico"));

ConsumerRecords<String, String> records = consumer.poll(java.time.Duration.ofSeconds(5));

for (ConsumerRecord<String, String> record : records) {

System.out.printf("Recebido: %s%n", record.value());

}

}

}

}

**Dica:** Você pode modificar os exemplos para enviar e consumir múltiplas mensagens, testar diferentes tópicos ou experimentar com múltiplos consumidores para entender o funcionamento dos consumer groups.

Esses exemplos são apenas para fins didáticos e funcionam em ambientes locais com o Kafka rodando no padrão (localhost:9092).

**Exercícios Práticos**

Para praticar e aprofundar os conceitos desta parte, consulte também o arquivo auxiliar:

* parte1-fundamentos/exercicios-parte1.md — Exercícios de fundamentos, comandos básicos, experimentação inicial e espaço para anotações.

**Código-Fonte e Exemplos**

Todo o conteúdo, exemplos práticos e arquivos de configuração deste artigo estão disponíveis no repositório oficial do projeto no GitHub:

[**🔗**](https://github.com/chmulato/kafka-java-mastery) [**github.com/chmulato/kafka-java-mastery**](http://github.com/chmulato/kafka-java-mastery)

**Acesse, explore e contribua!**