



Data Science Academy

www.datascienceacademy.com.br

Data Lake – Design, Projeto e Integração

Desenvolvendo Kafka Consumer em Java Produzindo e Consumindo Dados em Tempo Real



Nas aulas anteriores você aprendeu como criar uma aplicação Java para produzir mensagens e enviá-las ao Kafka. Agora você encontra um exemplo de aplicação Java para consumir as mensagens.

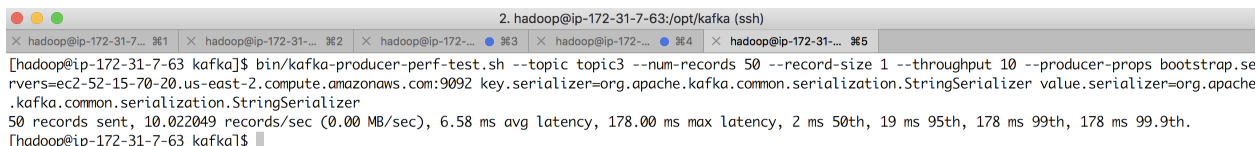
A aplicação está no script em anexo. Abra na sua IDE e edite as informações do servidor e nome do tópico. Execute o programa, da mesma forma que fez com o Producer.

Para ver um exemplo mais interessante, você pode usar um script fornecido pelo Kafka que produz mensagens em tempo real. Esse script é usado para testes de performance com o Kafka. Para usá-lo, abra um terminal e digite o comando conforme abaixo (considerando, claro, que seu Kafka Cluster está ativo):



```
2. hadoop@ip-172-31-7-63:/opt/kafka (ssh)
[hadoo@ip-172-31-7-63 kafka]$ bin/kafka-producer-perf-test.sh --topic topic3 --num-records 50 --record-size 1 --throughput 10 --producer-props bootstrap.servers=ec2-52-15-70-20.us-east-2.compute.amazonaws.com:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringSerializer
```

```
bin/kafka-producer-perf-test.sh --topic topic3 --num-records 50 --record-size 1 --throughput 10 --producer-props bootstrap.servers=ec2-52-15-70-20.us-east-2.compute.amazonaws.com:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringSerializer
```



```
2. hadoop@ip-172-31-7-63:/opt/kafka (ssh)
[hadoo@ip-172-31-7-63 kafka]$ bin/kafka-producer-perf-test.sh --topic topic3 --num-records 50 --record-size 1 --throughput 10 --producer-props bootstrap.servers=ec2-52-15-70-20.us-east-2.compute.amazonaws.com:9092 key.serializer=org.apache.kafka.common.serialization.StringSerializer value.serializer=org.apache.kafka.common.serialization.StringSerializer
50 records sent, 10.022049 records/sec (0.00 MB/sec), 6.58 ms avg latency, 178.00 ms max latency, 2 ms 50th, 19 ms 95th, 178 ms 99th, 178 ms 99.9th.
[hadoo@ip-172-31-7-63 kafka]$
```

O comando é longo, então tenha atenção. O comando cria um Producer, que você pode usar para gerar grandes volumes de dados em tempo real e testar suas aplicações de consumo. Altere o nome do servidor para o nome do seu servidor Kafka. Fique à vontade para aumentar o número de registros no parâmetro num-records. Ao executar o comando, o Kafka produz mensagens e enviar ao broker. Sua aplicação Java então consome as mensagens, conforme tela abaixo:

The screenshot displays the IntelliJ IDEA IDE with the following components:

- Project View (Left):** Shows the project structure for 'Cap07Producer'. The 'src/main/java' directory contains 'KafkaConsumerApp' and 'KafkaProducerApp'. The 'resources' directory is also visible.
- Code Editor (Center):** Displays the 'KafkaConsumerApp.java' file. The code includes imports for 'org.apache.kafka.common.*', 'org.apache.kafka.clients.consumer.*', and 'java.util.*'. It defines a 'KafkaConsumerApp' class with a 'main' method. Inside the 'main' method, a 'Properties' object is created and populated with various configuration properties for a Kafka consumer, including bootstrap servers, serializers, session timeout, auto.offset.reset, connections.max.idle.ms, enable.auto.commit, and exclude.internal.topics.
- Run View (Bottom):** Shows the output of the application. It displays a series of log messages indicating the consumer's progress, such as 'Tópico: topic3, Partição: 0, Offset: 89, Key: null, Value: S'.
- Terminal (Bottom):** Shows the command 'Run' and the output of the application.

Esse exemplo de aplicação pode ser usado para consumir os dados no Kafka e levar para armazenamento no HDFS ou processamento com o Spark, de acordo com o seu objetivo final.

Altere o programa e experimente outras opções. Você pode por exemplo, consumir os dados em pequenos lotes (batches) apenas alterando o código para consumir os dados em intervalos de tempo. As possibilidades são muitas.

Bons estudos!