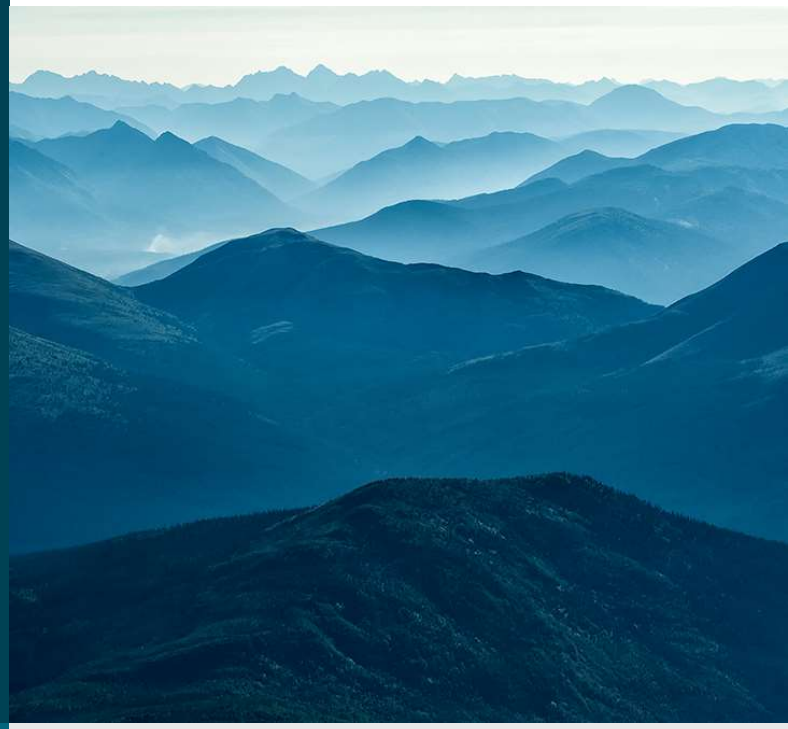
Engenharia de Dados Big Data

Aula 3 - Práticas de Ingestão e Manipulação de Dados



13/10/2021

Índice

Práticas de Ingestão

indra

1

Práticas de ingestão – Sqoop

Sqoop = SQL TO HADOOP

Práticas de ingestão – Sqoop

Ferramenta para transferir dados entre o Hadoop e banco de dados relacionais

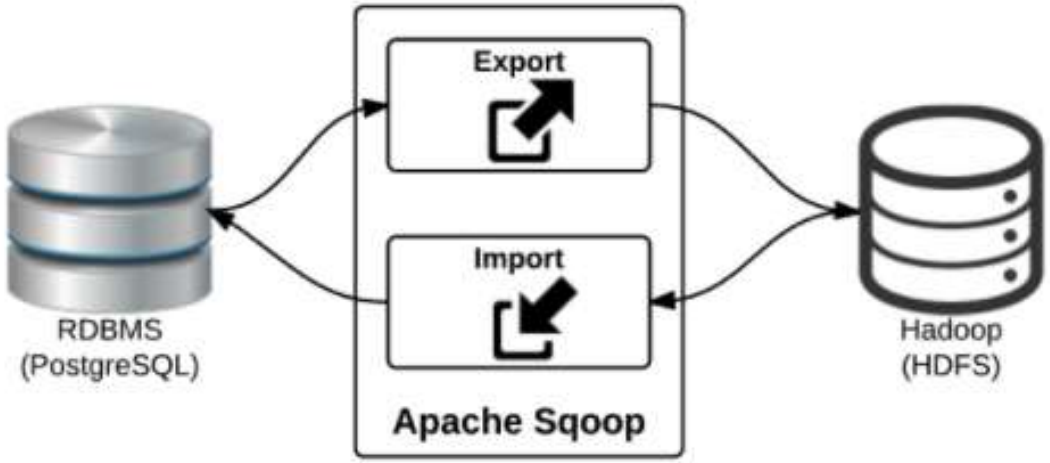
Sqoop = “SQL to Hadoop”

Importar dados
• Banco de dados relacional (RDBMS)

MySQL, SQL Server, Oracle
• Para o HDFS, Hive ou HBase

Transformar dados em MapReduce
• Execução em paralelo
• Tolerância a falhas

Exportar dados
• Armazenamento do Hadoop para um RDBMS



Práticas de ingestão – Sqoop Comandos

$ sqoop
- help
- version
- import
- import-all-tables
- export
- validation
- job
- metastore
- merge
- codegen
- create-hive-table
- eval
- list-databases
- list-tables

Práticas de ingestão – Sqoop Comandos

Informações para conexão
- Database type (MySQL, Oracle etc)
- Hostname
- Port number
- Database Name (list-databases)

Parâmetros
- --connect \ --username usuario \ --password senha

DB String de conexão
HSQLDB jdbc:hsqldb:*//
MySQL jdbc:mysql://
Oracle jdbc:oracle:*//
PostgreSQL jdbc:postgresql://

Práticas de ingestão – Sqoop Comandos

Listar Banco de Dados

• $ sqoop list-databases \
--connect jdbc:mysql://database \
--username usuario \
--password senha

Listar tabelas

• $ sqoop list-tables \
--connect jdbc:mysql://database/employees \
--username usuario \
--password senha

Práticas de ingestão – Sqoop

Sqoop eval

- Permite que os usuários executem consultas definidas pelo usuário nos respectivos servidores de banco de dados e visualizem o resultado no console.

Ex:

```
$ sqoop eval \
--connect jdbc:mysql://database/employees \
--username=root \
--password=secret \
--query "SELECT * FROM employees limit 15"
```

Práticas de ingestão – Sqoop

Sqoop import

- Importar dados do banco de dados RDBMS para o Hadoop HDFS.

```
sqoop import --table employees \
--connect jdbc:mysql://database/employees \
--username root \
--password secret \
--warehouse-dir /user/hive/warehouse/treinamento.db
```

Práticas de ingestão – Sqoop

Sqoop export

- Exportar dados de volta do HDFS para o banco de dados RDBMS.

```
sqoop export \
--connect jdbc:mysql://database/treinamento \
--username root \--password secret \
--export-dir /user/andre/data \
--fields-terminated-by ';' \
--table tb_dados_covid_mng
```

Práticas de ingestão – Kafka

Práticas de ingestão – Kafka

Plataforma de streaming distribuída
- Open Source

- Publicar e assinar streams de registros

- Fluxos de registros
  Processamento

- Tempo real
  Armazenamento

- Tolerante a falhas

Práticas de ingestão – Kafka

2010
- Originalmente desenvolvido pelo LinkedIn
  Necessidade de integração massiva de dados
  Conceito do Kafka surgido com Jay Kreps e sua equipe

2011
- Liberado como um projeto open-source

2012
- Apache Kafka

Práticas de ingestão – Kafka

Kafka é desenvolvido em Scala e Java

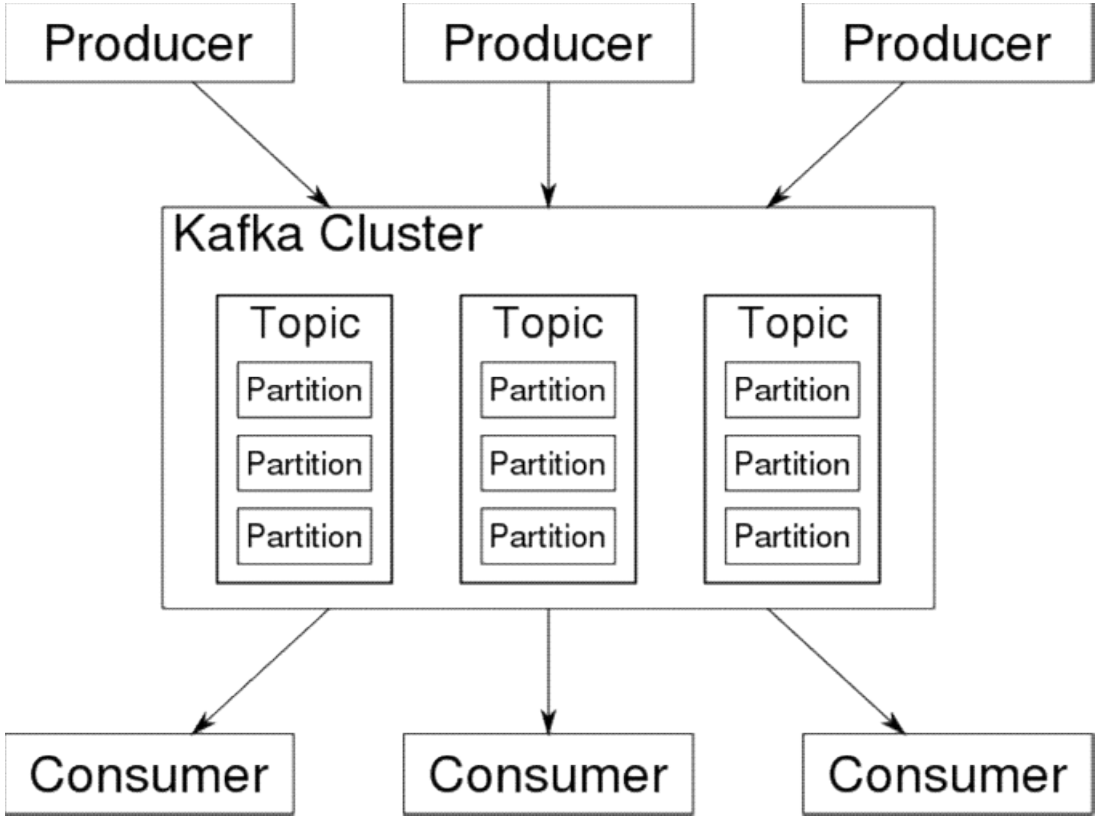
O Kafka é executado como um cluster em um ou mais servidores que podem abranger vários datacenters

O cluster Kafka armazena fluxos de registros em categorias denominadas tópicos

Cada registro consiste em uma chave, um valor e um registro de data e hora

Apache Kafka é um sistema para gerenciamento de fluxos de dados em tempo real, gerados a partir de web sites, aplicações e sensores

Práticas de ingestão – Kafka



Práticas de ingestão – Kafka

Producer API
• Permite que um aplicativo publique um fluxo de registros em um ou mais tópicos do Kafka

Consumer API
• Permite que um aplicativo assine um ou mais tópicos e processe o fluxo de registros produzidos para eles

Streams API
• Permite que um aplicativo transforme os fluxos de entrada em fluxos de saída

Connector API
• Permite criar e executar produtores ou consumidores reutilizáveis que conectam tópicos do Kafka a aplicativos ou sistemas de dados existentes

Práticas de ingestão – Kafka

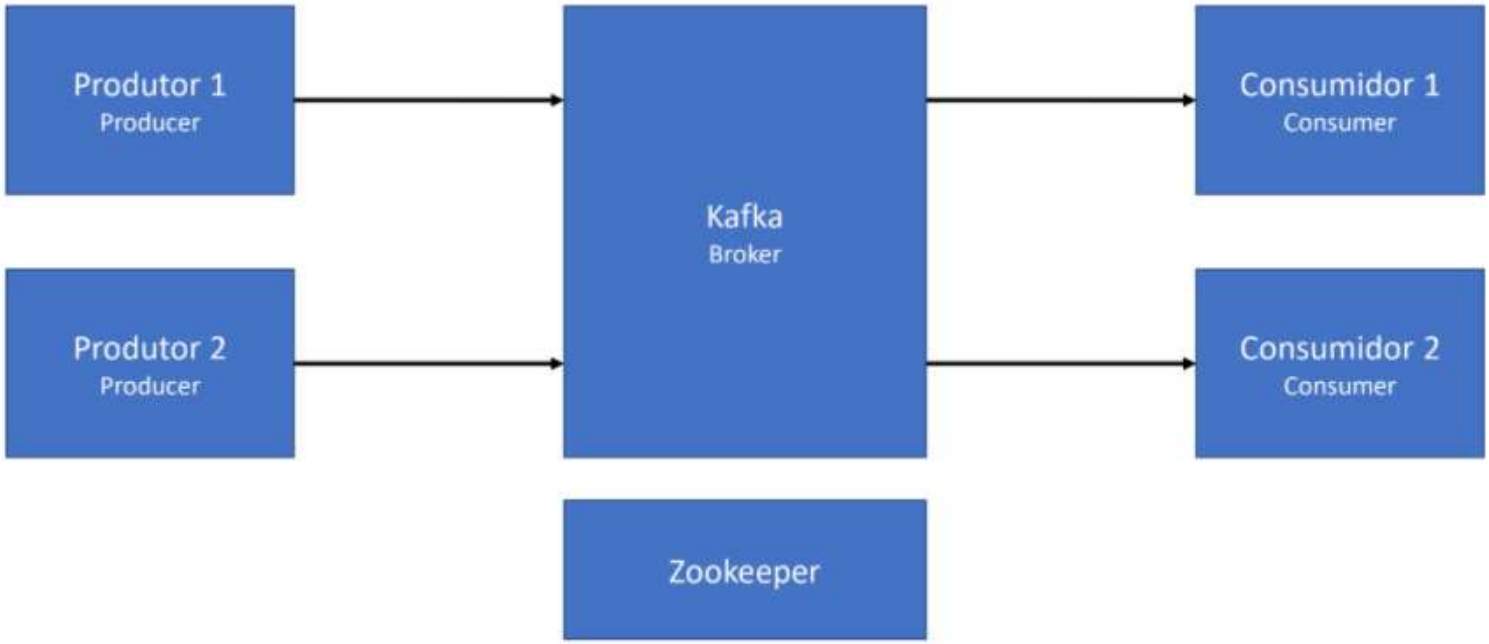
Producer API
• Permite que um aplicativo publique um fluxo de registros em um ou mais tópicos do Kafka

Consumer API
• Permite que um aplicativo assine um ou mais tópicos e processe o fluxo de registros produzidos para eles

Streams API
• Permite que um aplicativo transforme os fluxos de entrada em fluxos de saída

Connector API
• Permite criar e executar produtores ou consumidores reutilizáveis que conectam tópicos do Kafka a aplicativos ou sistemas de dados existentes

Práticas de ingestão – Kafka



Práticas de ingestão – Kafka

Produtores e consumidores

Produtores
- Publicar dados nos tópicos de sua escolha

Escolher qual registro atribuir a qual partição dentro do tópico
- Balancear a carga
- Chave no registro

Consumidores
- Receber os dados

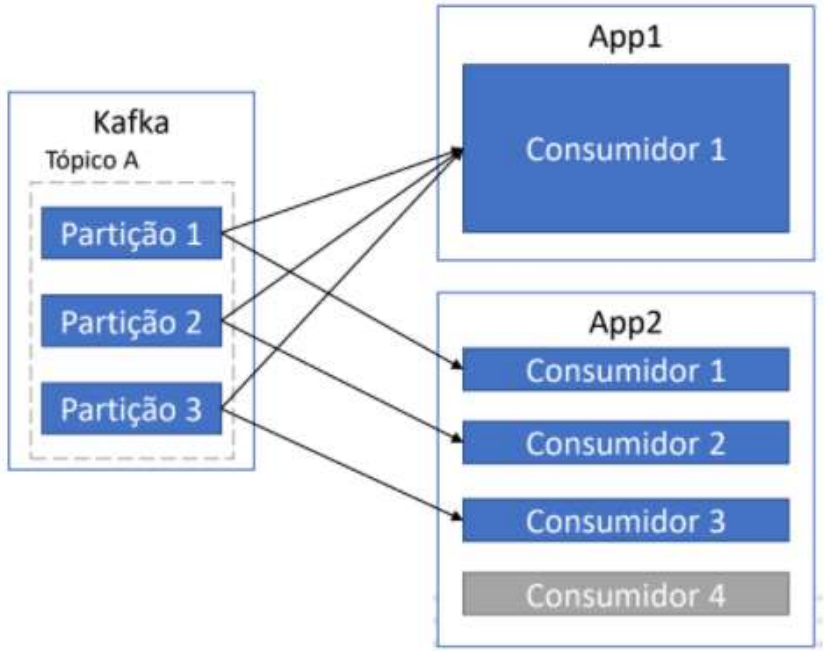
Cada registro publicado em um tópico
- Entregue aos consumidores dentro de grupo de consumidores

Práticas de ingestão – Kafka

Grupo de consumidores

Se todas as instâncias do consumidor tiverem no mesmo grupo de consumidores
• registros serão balanceados por carga

Se todas as instâncias do consumidor tiverem em grupos de consumidores diferentes
• Cada registro será transmitido para todos os processos



Práticas de ingestão – Kafka

Comandos básicos

kafka-topics --bootstrap-server localhost:9092 -list
ou
kafka-topics --zookeeper localhost:2181 -list

o Criar tópico
kafka-topics --bootstrap-server localhost:9092 --topic <nomeTópico>
--create --partitions 3 --replication-factor 1

o Descrever tópico
kafka-topics --bootstrap-server localhost:9092 --topic <nomeTópico> --describe

o Deletar tópico
kafka-topics --bootstrap-server localhost:9092 --topic <nomeTópico> --delete

Práticas de ingestão – Kafka

Comandos básicos

Enviar dados
kafka-console-producer --broker-list localhost:9092 --topic <nomeTópico>

kafka-console-producer --broker-list localhost:9092 --topic <nomeTópico>

Receber mensagens em tempo real
kafka-console-consumer -bootstrap-server localhost:9092 --topic <nomeTópico>

Receber mensagens desde a criação do tópico
kafka-console-consumer -bootstrap-server localhost:9092 --topic <nomeTópico> --from-beginning

Criar grupo de consumidores
kafka-console-consumer -bootstrap-server localhost:9092 --topic <nomeTópico> --group <nomeGrupo>

Processamento de Dados – Spark

Processamento de Dados – Spark

Plataforma de computação em cluster
- Suporte
  MapReduce
  Streaming

Análises interativas
- Execução em memória
- Compatível com Hadoop

Funciona com YARN
Acessar os dados
- HDFS
- Tabelas Hive
- Tabelas Hbase

o Linguagem: Scala, Java, Python e R

Processamento de Dados – Spark

Spark
• ETL e processamento em batch

Spark SQL
Consultas em dados estruturados

Spark Streaming
• Processamento de stream

Spark MLib
• Machine Learning

Spark GraphX
• Processamento de grafos

Ambiente Spark interativo

Inicialização no Cluster
• pyspark para Python
• spark-shell para Scala

Processamento de Dados – Spark

Representação de dados no spark
DataFrames

• Dados estruturados e semiestruturados em forma tabular
• Java, Scala e Python
• Operações

Transformação
Ação

Processamento de Dados – Spark

Dados suportados para leitura e escrita no DataFrame
- Text files
  CSV
  JSON

Plain text
- Binary format files

Apache Parquet (Muito utilizado)

Apache ORC
- Tables

Hive metastore
  JDBC
- Configurar outros tipos

Processamento de Dados – Spark

val <dataframe> = spark.read.format("<formato>").load("<arquivo>")

• <formato>

textFile(“arquivo.txt”)
csv(“arquivo.csv”)
jdbc(jdbcUrl, "bd.tabela", connectionProperties)
load ou parquet(“arquivo.parquet”)
table(“tabelaHive”)
json(“arquivo.json“)
orc(“arquivo.orc”)

• <arquivo>

“diretório/”
“diretório/*.log”
“arq1.txt, arq2.txt”
“arq*”

Processamento de Dados – Spark

Ação
- count: retorna o número de linhas
- first: retorna a primeira linha
- take(n): retorna as primeiras n linhas como um array
- show(n): exibe as primeiras n linhas da tabela
- collect: Trazer toda a informação dos nós do drive
- distincts: retorna os registros, removendo os repetidos
- write: salvar os dados
- printSchema() Visualizar a estrutura dos dados

  DataFrame sempre tem um esquema associado

Processamento de Dados – Spark

dadosDF.write.
• save(“arquivoParquet”)
• json("arquivoJson")
• csv(“arquivocsv”)
• saveAsTable("tableHive")
(/user/hive/warehouse)

scala> dadosDF.write.save(“outputData”)
/user/cloudera/outputData

scala> dadosDF.write. \
mode("append"). \
option("path","/user/root"). \
saveAsTable("outputData")


indra
At the core