

# Aceleração Global Dev #4 everis

## Criando pipelines de dados eficientes - Parte 1

### Spark Streaming

---

Marco Antonio Pereira  
Expert Data Architect

# Objetivos da Aula

**1.** Apresentar Apache Spark

**2.** StreamContext

**3.** Casos de Uso

# Requisitos Básicos

- ✓ Máquina Virtual com Apache Spark 2.4
- ✓ Conhecimentos básicos em Python ou Scala
- ✓ Paciência

# Parte 1: Sobre

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming

# O que é?

- Apache Spark é um projeto de processamento de dados distribuído de código aberto que foi iniciado em 2009
- Spark foi originalmente de código aberto
- O projeto Spark tem mais de 400 indivíduos contribuidores e committers de empresas como Facebook, Yahoo, Intel, Netflix, Databricks e outras
- Spark é escrito em Scala, que é construído em cima da Java Virtual Machine (JVM) e Java runtime
- Capaz de ser executado no Windows e também no Linux
- Mais de 500 organizações utilizam Spark

# O que é?

## **Spark como uma abstração**

O Spark permite que os desenvolvedores criem rotinas de processamento de dados complexas e de vários estágios, fornecendo uma API de alto nível e uma estrutura tolerante a falhas que permite aos programadores se concentrar na lógica em vez de questões ambientais ou de infraestrutura, como falha de hardware.

## **Spark é rápido, eficiente e escalonável**

O Spark implementa uma estrutura distribuída e tolerante a falhas na memória chamada Resilient Distributed Dataset (RDD) . O Spark maximiza o uso de memória em várias máquinas, melhorando o desempenho geral em ordens de magnitude. A reutilização dessas estruturas na memória pelo Spark o torna adequado para operações iterativas de aprendizado de máquina, bem como para consultas interativas.

# Aplicabilidade

- Operações de extração-transformação-carregamento (load) (ETL)
- Análise preditiva e aprendizado de máquina
- Operações de acesso a dados (como consultas e visualizações SQL)
- Mineração e processamento de texto
- Processamento de eventos em tempo real
- Aplicativos gráficos
- Reconhecimento de padrões
- Mecanismos de recomendação



# Programação

- Scala
- Python
- Java
- SQL
- R

# Programação

Exemplos RDD:

## Python

```
rddArquivo = sc.textFile ("hdfs://meucluster/data/arquivo.txt")
```

## Scala

```
val rddArquivo = sc.textFile("hdfs:// meucluster/data/arquivo.txt")
```

## Java

```
JavaRDD<String> rddArquivo = sc.textFile("hdfs://meucluster/data/arquivo.txt");
```

# Uso interativo

# PySpark:

```
[root@mycluster ~]# pyspark
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
15/11/17 00:16:08 WARN NativeCodeLoader: unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
Welcome to


$$\begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array} \begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array} \begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array}$$

version 1.3.1

Using Python version 2.6.6 (r266:84292, Jul 23 2015 15:22:56)
SparkContext available as sc, HiveContext available as sqlContext.
>>> █
```

# Uso interativo

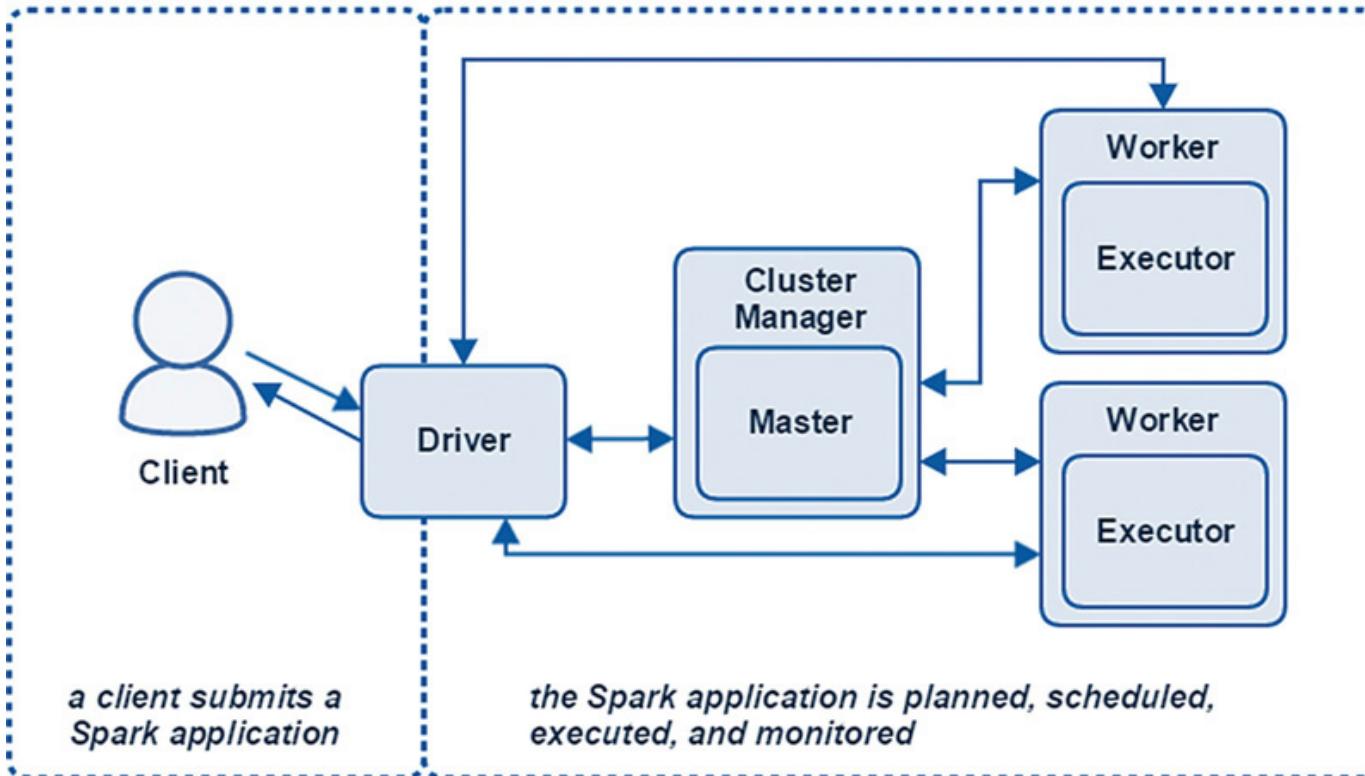
## Scala:

# Uso não-interativo

spark-submit:

```
[linux@user ~]$ spark-submit \
--master yarn \
--queue "SquadFI" \
--name "[Programa Spark 123] ETL 999" \
--driver-memory 2G \
--executor-memory 2G \
--executor-cores 1 \
--proxy-user hive \
--conf "spark.driver.maxResultSize=16g" \
--conf "spark.dynamicAllocation.enabled=true" \
--conf "spark.shuffle.service.enabled=true" \
--conf "spark.shuffle.service.port=7337" \
--conf "spark.dynamicAllocation.initialExecutors=10" \
--conf "spark.dynamicAllocation.minExecutors=10" \
--conf "spark.dynamicAllocation.maxExecutors=80" \
--conf "spark.yarn.driver.memoryOverhead=2000" \
--conf "spark.yarn.executor.memoryOverhead=2000" \
--driver-java-options "-Djavax.security.auth.useSubjectCredsOnly=false" \
--jars commons-csv-1.2.jar,spark-csv_2.11-1.5.0.jar \
Main.py <parametro 1> <parametro 2> <parametro N>
```

# Anatomia



# Anatomia

## DRIVER

A vida útil do aplicativo Spark começa (e termina) com o driver Spark. O driver Spark é o processo que os clientes usam para enviar aplicativos no Spark. O driver também é responsável por planejar e coordenar a execução do programa Spark e retornar o status e/ou resultados (dados) ao cliente.

O driver Spark é responsável por criar o SparkContext. O SparkContext, é referido como sc, é a instância do aplicativo que representa a conexão com o Master do Spark (e os Workers do Spark). O SparkContext é instanciado no início de um aplicativo Spark (incluindo os shells interativos) e é usado para todo o programa.

- Uma das principais funções do driver é planejar o aplicativo.
- O driver obtém a entrada de processamento do aplicativo e planeja a execução do programa.
- O driver pega todas as transformações solicitadas (operações de manipulação de dados) e ações (solicitações de saída ou um prompt para executar o programa) e cria um Gráfico Acíclico Direcionado (Directed Acyclic Graph - DAG).

# Anatomia

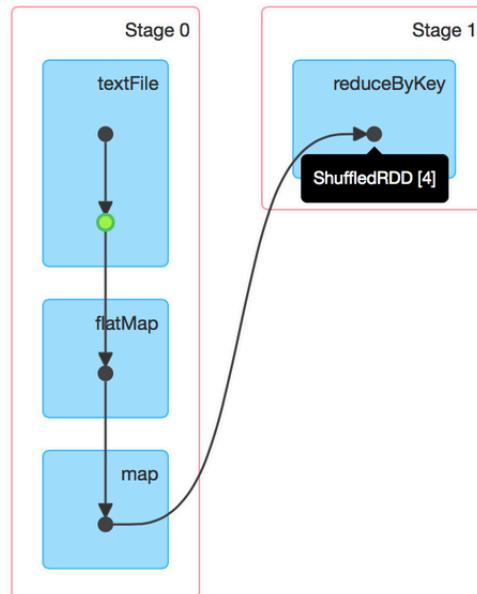
## Details for Job 0

### DAG

Status: SUCCEEDED

Completed Stages: 2

- ▶ Event Timeline
- ▼ DAG Visualization



# Anatomia

## CLUSTER MANAGER

O Cluster Manager é o processo responsável por monitorar os nós Workers e reservar recursos nesses nós mediante solicitação do Spark Master. O Spark Master, por sua vez, disponibiliza esses recursos de cluster para o Driver na forma de executores.

O Cluster Manager pode ser separado do processo Master. Esse é o caso ao executar o Spark no Mesos ou YARN.

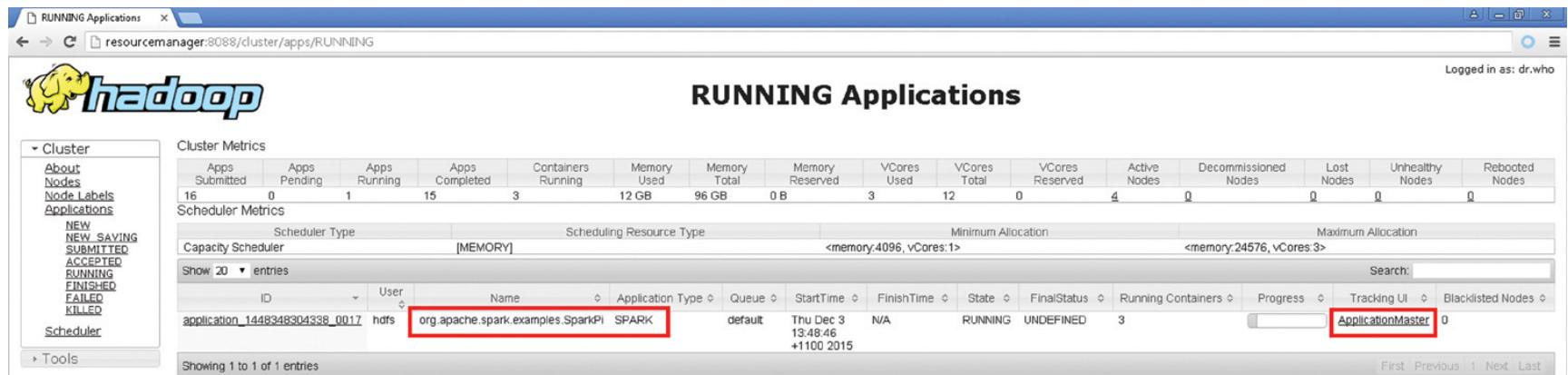
No caso do Spark rodando no modo Standalone, o processo Master também executa as funções do Cluster Manager. Efetivamente, ele atua como seu próprio gerenciador de cluster.

- O Cluster Manager em um aplicativo Spark distribuído é o processo que governa, monitora e reserva recursos na forma de contêineres em nós de worker de cluster (ou escravos). Esses contêineres são reservados mediante solicitação do Spark Master. O Cluster Manager no caso do Spark no YARN é o YARN ResourceManager.
- Um Spark Driver em execução no modo YARN envia um aplicativo ao ResourceManager e, em seguida, o ResourceManager designa um ApplicationsMaster para o aplicativo Spark.

# Anatomia

## CLUSTER MANAGER

Exemplo de um aplicativo Spark gerenciado pelo YARN mostrado no ResourceManager UI, normalmente disponível em [http://<resource\\_manager>:8088](http://<resource_manager>:8088)



The screenshot shows the "RUNNING Applications" page of the Hadoop ResourceManager UI. The URL is `resourcemanager:8088/cluster/apps/RUNNING`. The page title is "RUNNING Applications". On the left, there's a sidebar with "Cluster Metrics" and "Scheduler Metrics" sections, and a "Tools" section. The main area displays a table of running applications.

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Progress	Tracking UI	Blacklisted Nodes
application_1448348304338_0017	hdfs	org.apache.spark.examples.SparkPI	SPARK	default	Thu Dec 3 13:48:46 +1100 2015	N/A	RUNNING	UNDEFINED	3		<a href="#">ApplicationMaster</a>	0

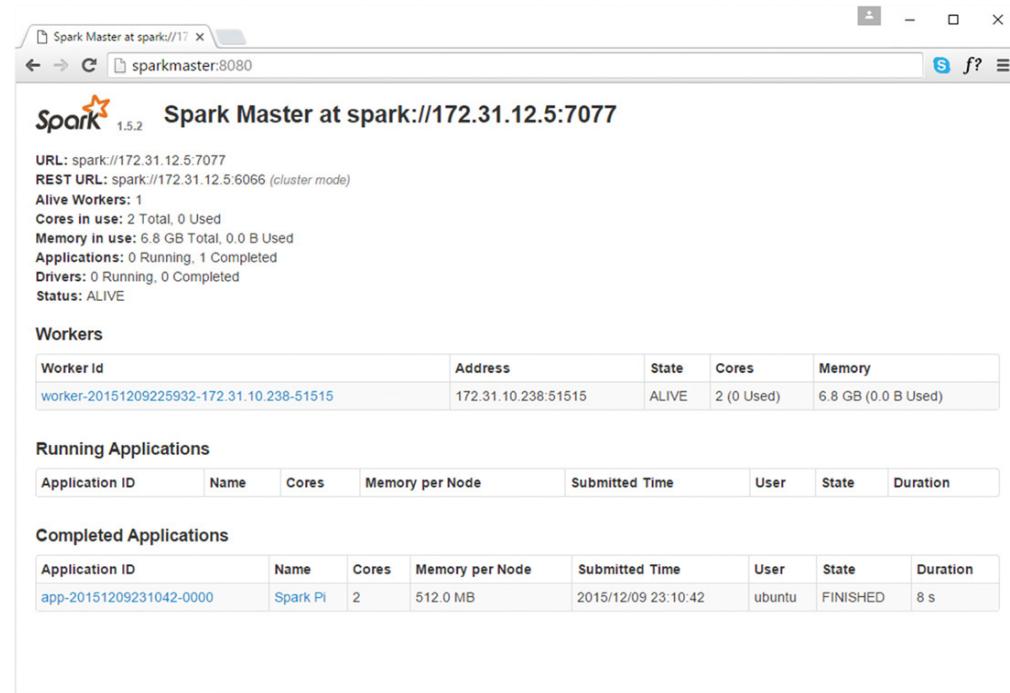
One row in the table is highlighted with a red box, specifically the row for application ID `application_1448348304338_0017`.

# Anatomia

## SPARK MASTER

O Spark master é o processo que solicita recursos no cluster e os disponibiliza para o driver Spark. Em qualquer modo de implantação, o mestre negocia recursos ou contêineres com nós de trabalho ou nós escravos e rastreia seu status e monitora seu progresso.

O processo mestre do Spark atende a uma interface de usuário da web na porta 8080 no host mestre:



The screenshot shows the Spark Master UI at `spark://172.31.12.5:7077`. The main statistics include:

- URL: `spark://172.31.12.5:7077`
- REST URL: `spark://172.31.12.5:6066 (cluster mode)`
- Alive Workers: 1
- Cores in use: 2 Total, 0 Used
- Memory in use: 6.8 GB Total, 0.0 B Used
- Applications: 0 Running, 1 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

The Workers section lists one worker:

Worker Id	Address	State	Cores	Memory
worker-20151209225932-172.31.10.238-51515	172.31.10.238:51515	ALIVE	2 (0 Used)	6.8 GB (0.0 B Used)

The Running Applications section shows one application:

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20151209231042-0000	Spark Pi	2	512.0 MB	2015/12/09 23:10:42	ubuntu	FINISHED	8 s

The Completed Applications section is empty.

# Anatomia

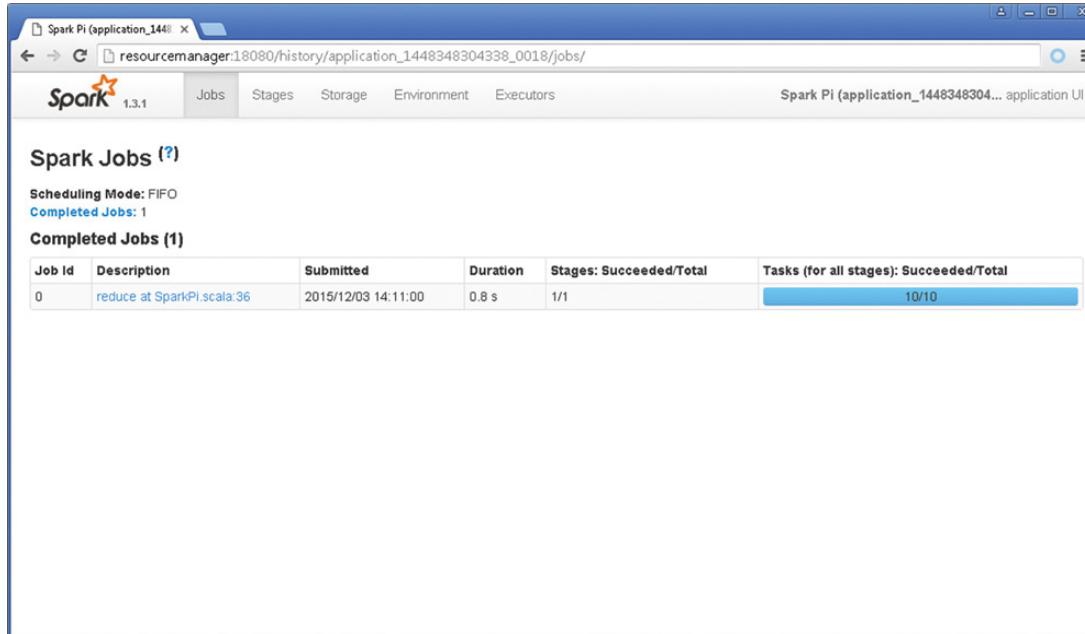
## SPARK MASTER

- O processo ApplicationsMaster que é instanciado pelo ResourceManager no envio do aplicativo Spark atua como o Spark Master. O Driver informa o ApplicationsMaster sobre os requisitos de seu executor para o aplicativo. O ApplicationsMaster, por sua vez, solicita containers (que são hospedados em NodeManagers) do ResourceManager para hospedar esses executores.
- O ApplicationsMaster é responsável por gerenciar esses containers (executors) durante o ciclo de vida do aplicativo Spark.
- O Driver coordena o estado do aplicativo e as transições de estágio de processamento.
- O próprio ApplicationsMaster é hospedado em uma JVM em um nó slave ou worker no cluster e é, na verdade, o primeiro recurso alocado para qualquer aplicativo Spark em execução no YARN.

# Anatomia

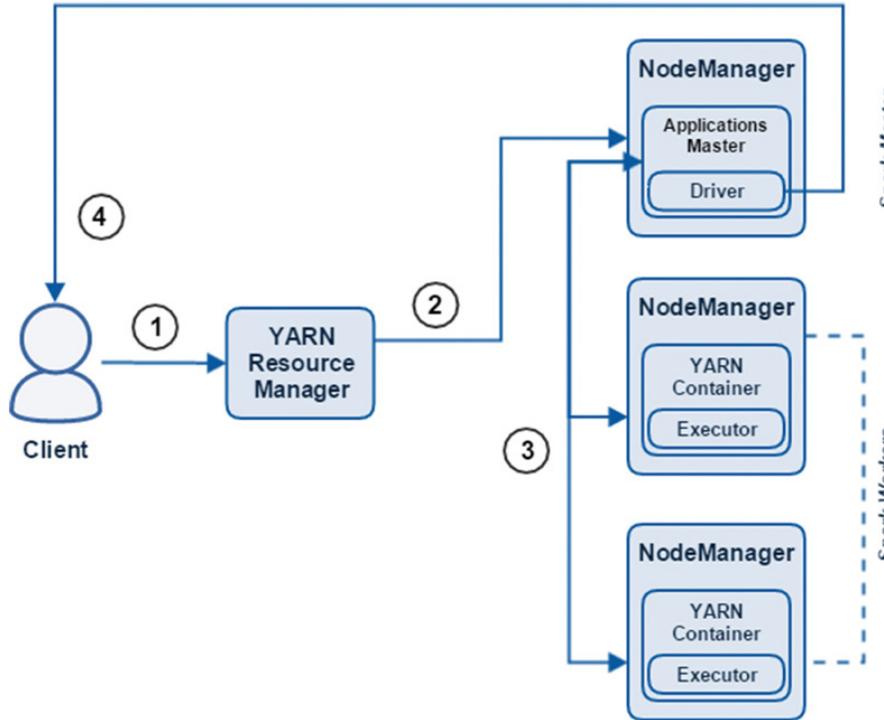
## SPARK MASTER

A UI do ApplicationsMaster para um aplicativo Spark é a UI do Spark Master. Isso está disponível como um link na UI do ResourceManager:



# Anatomia

## Modo Cluster



# Anatomia

## Modo Cluster (Yarn-Cluster)

1. O Client (uma chamada de processo do usuário via spark-submit) envia um aplicativo Spark para o Cluster Manager (o YARN ResourceManager).
2. O ResourceManager atribui um ApplicationsMaster (o Spark Master) para o aplicativo. O processo do Driver é criado no mesmo nó.
3. O ApplicationsMaster solicita containers para os Executors do ResourceManager. Os contêineres são atribuídos e os executors são gerados. O Driver então se comunica com os executors para organizar o processamento de tarefas e estágios do programa Spark.
4. O driver retorna o progresso, resultados e status para o cliente.

# Iniciando

[linux@user ~]\$ pyspark --master local

Option	Description
Local	Runs the application in local mode using one core.
local [n]	Runs the application in local mode using <i>n</i> cores. An asterisk (*) can be used as <i>n</i> to enable <i>all</i> cores on the system to be used.
local [n,m]	Runs the application in local mode using <i>n</i> cores, retrying failed tasks <i>m</i> times. An asterisk can be used as <i>n</i> to enable <i>all</i> cores on the system to be used.

## Modo não interativo (aplicativo em Spark-submit)

```
from pyspark import SparkContext
sc = SparkContext ("local [*]")
#o código do seu aplicativo ...
```

# Cloud

## AWS



EMR

Managed Hadoop Framework



AWS Glue DataBrew

Visual data preparation tool to clean and normalize data for analytics and machine learning

# Cloud

## AWS-EMR (Elastic MapReduce)

### Configuração de software

Versão emr-6.2.0



<input checked="" type="checkbox"/> Hadoop 3.2.1	<input type="checkbox"/> Zeppelin 0.9.0	<input type="checkbox"/> Livy 0.7.0
<input type="checkbox"/> JupyterHub 1.1.0	<input type="checkbox"/> Tez 0.9.2	<input type="checkbox"/> Flink 1.11.2
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 2.2.6-amzn-0	<input type="checkbox"/> Pig 0.17.0
<input checked="" type="checkbox"/> Hive 3.1.2	<input type="checkbox"/> Presto 0.238.3	<input type="checkbox"/> PrestoSQL 343
<input type="checkbox"/> ZooKeeper 3.4.14	<input type="checkbox"/> JupyterEnterpriseGateway 2.1.0	<input type="checkbox"/> MXNet 1.7.0
<input type="checkbox"/> Sqoop 1.4.7	<input checked="" type="checkbox"/> Hue 4.8.0	<input type="checkbox"/> Phoenix 5.0.0
<input type="checkbox"/> Oozie 5.2.0	<input checked="" type="checkbox"/> Spark 3.0.1	<input type="checkbox"/> HCatalog 3.1.2
<input type="checkbox"/> TensorFlow 2.3.1		



# Cloud

## AWS-EMR (Elastic MapReduce)

```
 .aws — hadoop@ip- — ssh - aws emr ssh --cluster-id j- — key-pair-file emr-cluster.p...
 (dataengineer) $ aws emr ssh --cluster-id j- — key-pair-file emr-cluste
 r.pem
 ssh -o StrictHostKeyChecking=no -o ServerAliveInterval=10 -i emr-cluster.pem hadoop@ec2- — .us-west-2.com
 [pute.amazonaws.com -t
 Warning: Permanently added 'ec2- — .us-west-2.compute.amazonaws.com, — (ECDSA) to the list of
 known hosts.
 Last login: —
 —| —|— )
 —| ( — /   Amazon Linux AMI
 ---| \---|--- |

 https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
 33 package(s) needed for security, out of 57 available
 Run "sudo yum update" to apply all updates.

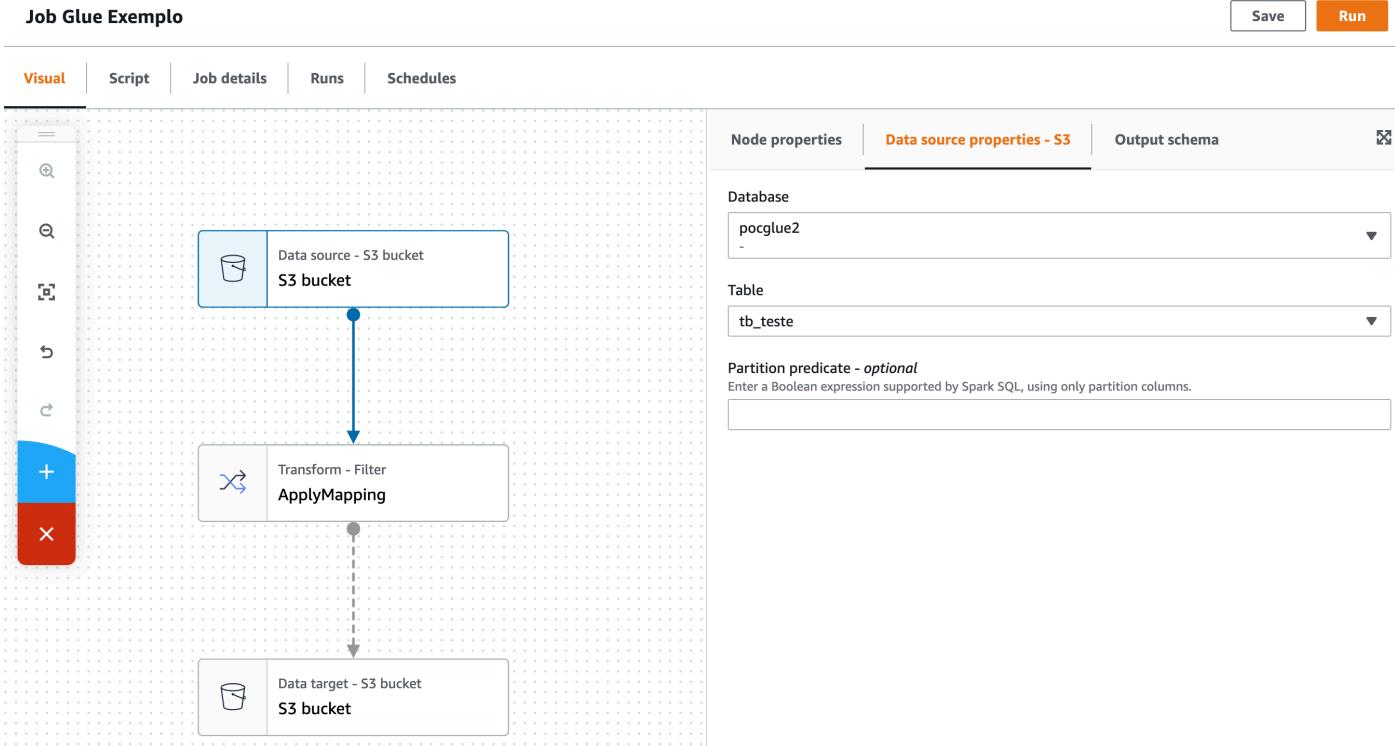
 EEEEEEEEEEEEEEEEEE MMMMMMM
 E:::::::::::E M:::::M      M:::::M R:::::R
 EE:::::EEEEEE:::E M:::::M      M:::::M R:::::RRRRRR:::::R
 E:::::E      EEEEE M:::::M      M:::::M RR:::::R      R:::::R
 E:::::E      M:::::M:::::M M:::::M:::::M R:::::R      R:::::R
 E:::::EEEEEEEEE  M:::::M M::::M M::::M M:::::M R:::::RRRRRR:::::R
 E:::::::::::E  M:::::M M:::::M:::::M M:::::M R:::::::::::RR
 E:::::EEEEEEEEE  M:::::M M:::::M M:::::M R:::::RRRRRR:::::R
 E:::::E      M:::::M M:::::M M:::::M R:::::R      R:::::R
 E:::::E      EEEEE M:::::M     MMM  M:::::M R:::::R      R:::::R
 EE:::::EEEEEEEEE:::E M:::::M      M:::::M R:::::R      R:::::R
 E:::::::::::E  M:::::M      M:::::M RR:::::R      R:::::R
 EEEEEEEEEEEEEEEEEE MMMMMMM RRRRRRRR RRRRRRR

 [hadoop@ip- ~]$
```

# Cloud

## AWS-GLUE

Job Glue Exemplo



# Cloud

## AWS-GLUE

### Job Glue Example

SaveRun[Visual](#)[Script](#)[Job details](#)[Runs](#)[Schedules](#)

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 import re
8
9 ## @params: [JOB_NAME]
10 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
11
12 sc = SparkContext()
13 glueContext = GlueContext(sc)
14 spark = glueContext.spark_session
15 job = Job(glueContext)
16 job.init(args['JOB_NAME'], args)
17 ## @type: DataSource
18 ## @args: [database = "pocglue2", table_name = "tb_teste", transformation_ctx = "DataSource0"]
19 ## @return: DataSource0
20 ## @inputs: []
21 DataSource0 = glueContext.create_dynamic_frame.from_catalog(database = "pocglue2", table_name = "tb_teste", transformation_ctx = "DataSource0")
22 ## @type: Filter
23 ## @args: [f = lambda row : (row["altura"] < 40), transformation_ctx = "Transform0"]
24 ## @return: Transform0
25 ## @inputs: [frame = DataSource0]
26 Transform0 = Filter.apply(frame = DataSource0, f = lambda row : (row["altura"] < 40), transformation_ctx = "Transform0")
27 ## @type: DataSink
28 ## @args: [connection_type = "s3", format = "json", connection_options = {"path": "s3://saida11/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
29 ## @return: DataSink0
30 ## @inputs: [frame = Transform0]
31 DataSink0 = glueContext.write_dynamic_frame.from_options(frame = Transform0, connection_type = "s3", format = "json", connection_options = {"path": "s3://sa..."}, transformation_ctx = "DataSink0")
32 job.commit()
```

# Cloud

## AWS-GLUE

Visual | Script | **Job details** | Runs | Schedules

Name  
Job Glue Exemplo

Description - *optional*  
  
Descriptions can be up to 2048 characters long.

IAM Role  
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.  
AWSGlueServiceRole-dfCrawler  
No description available.

Type  
The type of ETL job. This is set automatically based on the types of data sources you have selected.  
Spark

Glue version [Info](#)  
Glue 2.0 - Supports spark 2.4, Scala 2, Python 3

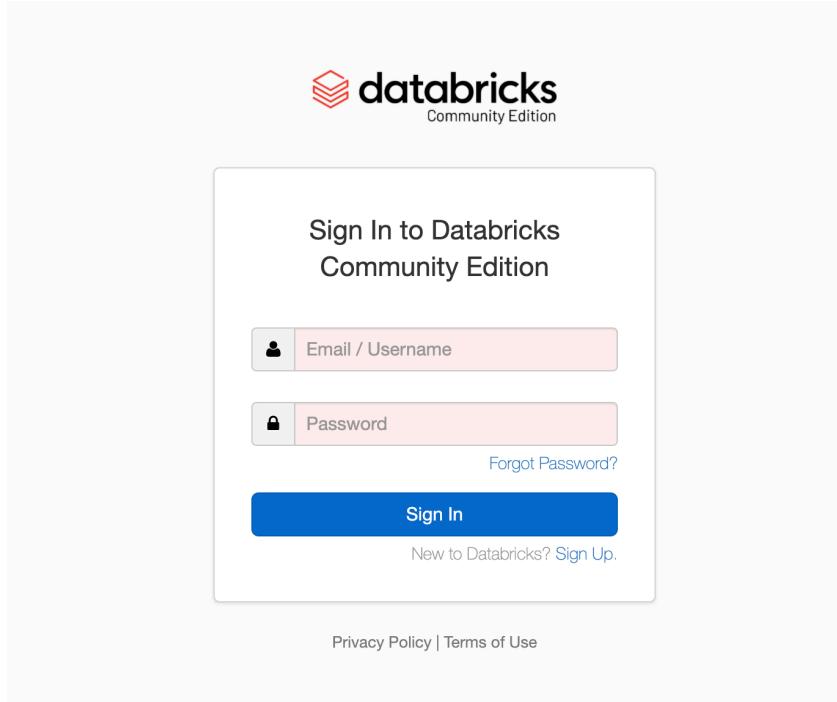
Language  
Python 3

Worker type  
Set the type of predefined worker that is allowed when a job runs.  
G.1X

Number of workers  
The number of workers of a defined workerType that are allocated when a job runs. The maximum number of workers you can define are 299 for G.1X, and 149 for G.2X.  
10

# Cloud

**Databricks Community (<https://community.cloud.databricks.com/>)**



# Cloud

**Databricks Community (<https://community.cloud.databricks.com/>)**

Welcome to  **databricks**



#### Explore the Quickstart Tutorial

Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.



#### Import & Explore Data

Quickly import data, preview its schema, create a table, and query it in a notebook.



#### Create a Blank Notebook

Create a notebook to start querying, visualizing, and modeling your data.

#### Common Tasks

-  New Notebook
-  Create Table
-  New Cluster
-  New Job
-  New MLflow Experiment
-  Import Library
-  Read Documentation

#### Recents

-  PythonNotebook
-  Estudos10
-  MassaTestePOC
-  MassaTestePOC\_HBase
-  Estudos9
-  Estudos7
-  Estudos8

#### What's new in v3.37

[View latest release notes](#)



# Cloud

**Databricks Community (<https://community.cloud.databricks.com/>)**

Create Notebook

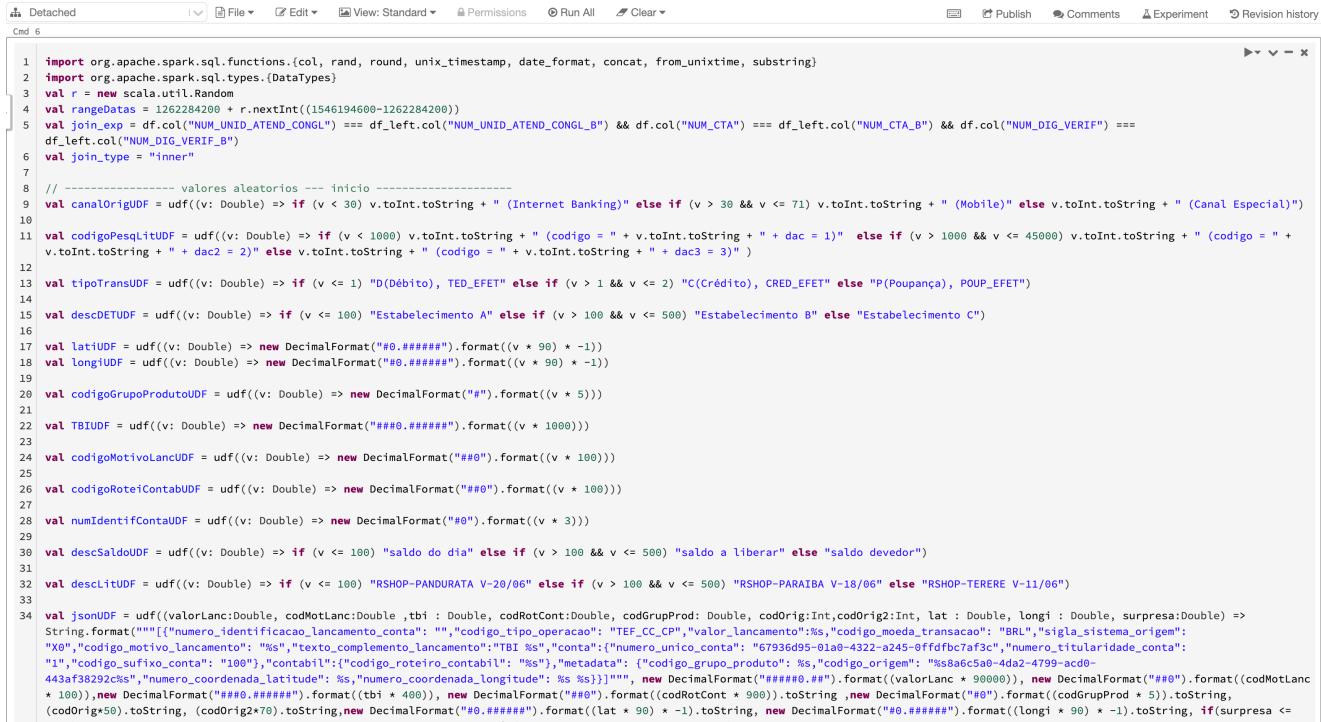
Name

Default Language

Cluster

# Cloud

## Databricks Community (<https://community.cloud.databricks.com/>)



```

1 import org.apache.spark.sql.functions.(col, rand, round, unix_timestamp, date_format, concat, from_unixtime, substring)
2 import org.apache.spark.sql.types.(DataTypes)
3 val r = new scala.util.Random
4 val rangeDatas = 1262284200 + r.nextInt((1546194600 - 1262284200))
5 val join_exp = df.col("NUM_UNID_ATEND_CONGL") === df_left.col("NUM_UNID_ATEND_CONGL_B") && df.col("NUM_CTA") === df_left.col("NUM_CTA_B") && df.col("NUM_DIG_VERIF") === df_left.col("NUM_DIG_VERIF_B")
6 val join_type = "inner"
7
8 // ----- valores aleatorios --- inicio
9 val canalOrigUDF = udf((v: Double) => if (v < 30) v.toInt.toString + " (Internet Banking)" else if (v > 30 & v <= 71) v.toInt.toString + " (Mobile)" else v.toInt.toString + " (Canal Especial)")
10
11 val codigoPesquisUDF = udf((v: Double) => if (v < 1000) v.toInt.toString + " (codigo = " + v.toInt.toString + " + dac = 1)" else if (v > 1000 && v <= 45000) v.toInt.toString + " (codigo = " + v.toInt.toString + " + dac2 = 2)" else v.toInt.toString + " (codigo = " + v.toInt.toString + " + dac3 = 3)" )
12
13 val tipoTransUDF = udf((v: Double) => if (v <= 1) "D(Débito), TED_EFET" else if (v > 1 && v <= 2) "C(Crédito), CRED_EFET" else "P(Poupança), POUPE_EFET")
14
15 val descDETUDF = udf((v: Double) => if (v <= 100) "Estabelecimento A" else if (v > 100 && v <= 500) "Estabelecimento B" else "Estabelecimento C")
16
17 val latiUDF = udf((v: Double) => new DecimalFormat("#,#####").format((v * 90) * -1))
18 val longiUDF = udf((v: Double) => new DecimalFormat("#,#####").format((v * 90) * -1))
19
20 val codigoGrupoProdutoUDF = udf((v: Double) => new DecimalFormat("#").format((v * 5)))
21
22 val TBIUDF = udf((v: Double) => new DecimalFormat("###0,#####").format((v * 1000)))
23
24 val codigoMotivoLancUDF = udf((v: Double) => new DecimalFormat("#0").format((v * 100)))
25
26 val codigoRoteiContabUDF = udf((v: Double) => new DecimalFormat("#0").format((v * 100)))
27
28 val numIdentifContaUDF = udf((v: Double) => new DecimalFormat("#0").format((v * 3)))
29
30 val descSaldoUDF = udf((v: Double) => if (v <= 100) "saldo d dia" else if (v > 100 && v <= 500) "saldo a liberar" else "saldo devedor")
31
32 val desclitUDF = udf((v: Double) => if (v <= 100) "RSHOP-PANDURATA V-20/06" else if (v > 100 && v <= 500) "RSHOP-PARAIBA V-18/06" else "RSHOP-TERERE V-11/06")
33
34 val jsonUDF = udf((valorLanc:Double, codMotLanc:Double ,tbi : Double, codRotCont:Double, codGrupProd: Double, codOrig:Int,codRig2:Int, lat : Double, longi : Double, surpresa:Double) =>
String.format("{{\"numero_identificacao_lancamento_conta\": \"\", \"codigo_tipo_operacao\": \"TEF_CC_CP\", \"valor_lancamento\":%s,\"codigo_moeda_transacao\": \"BRL\", \"sigla_sistema_origem\": \"X0\", \"codigo_moeda_lancamento\": \"%s\", \"texto_complemento_lancamento\": \"TBL %s\", \"conta\": \"numero_unico Conta\": \"67936d95-01a0-4322-a245-0ffdffbc7af3c\", \"numero_titularidade_conta\": \"1\", \"codigo_sufixo_conta\": \"108\", \"contabil\": {\"codigo_roteiro_contabil\": \"Rs\", \"metadata\": {\"codigo_grupo_produto\": \"%s\", \"codigo_origem\": \"58a8c65a0-4da2-4799-acd0-443af38292c8s\", \"numero_coordenada_latitude\": \"%s\", \"numero_coordenada_longitude\": \"%s %s\"}}}}\"", new DecimalFormat("###0.#####").format((codMotLanc * 100)), new DecimalFormat("###0.#####").format((tbi * 400)), new DecimalFormat("###0").format((codRotCont * 900)).toString(), new DecimalFormat("###0").format((codGrupProd * 5)).toString(), (codOrig*50).toString(), (codRig2*70).toString(),new DecimalFormat("###0.#####").format((lat * 90) * -1).toString(), new DecimalFormat("###0.#####").format((longi * 90) * -1).toString(), if(surpresa < 100) "0" else "1"))
    
```

# On Premises

## Cloudera

**cloudera manager** Clusters Hosts Diagnostics Audits Charts Backup Administration Search (Hotkey: /) Support admin

**Home** 30 minutes preceding November 3, 2015, 1:44 PM PST

Status All Health Issues Configuration ✗ 6 All Recent Commands Add Cluster

**Cluster 1** (CDH 5.5.0, Packages)

- Hosts ✗ 4
- FLUME-1
- HBASE-1
- HDFS-1 ✗ 1
- HIVE-1
- HUE-1 ✗ 1
- IMPALA-1
- KAFKA-1
- KS\_INDEXER-1
- KUDU-1
- MAPREDUCE-1
- OOZIE-1
- SOLR-1
- SPARK\_ON\_YA...
- SQOOP-1
- SQOOP\_CLIENT-1
- YARN-1
- ZOOKEEPER-1 ✗ 1

**Charts**

**Cluster CPU**



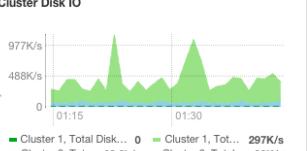
percent

0% 50% 100%

01:15 01:30

Cluster 1 2.4% Cluster 2 1.6%

**Cluster Disk IO**



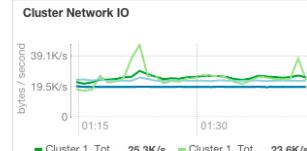
bytes / second

0 488K/s 977K/s

01:15 01:30

Cluster 1, Total Disk... 0 Cluster 1, Total... 297K/s Cluster 2, Tot... 68.3b/s Cluster 2, Total ... 99K/s

**Cluster Network IO**



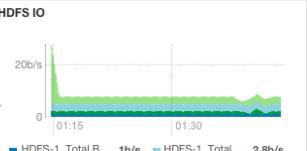
bytes / second

0 19.5K/s 39.1K/s

01:15 01:30

Cluster 1, Tot... 25.3K/s Cluster 1, Tot... 23.6K/s Cluster 2, Tot... 19.2K/s Cluster 2, Tot... 23.3K/s

**HDFS IO**



bytes / second

0 20b/s

01:15 01:30

HDFS-1, Total B... 1b/s HDFS-1, Total ... 2.8b/s HDFS-2, Total B... 1b/s HDFS-2, Total ... 2.8b/s

**Running MapReduce Jobs**



jobs

01:15 01:30

MAPREDUCE-1 0 MAPREDUCE-2 0

**Completed Impala Queries**



queries / second

01:15 01:30

IMPALA-1 0 IMPALA-2 0

# Parte 2: Streaming

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming

# Por que?

- Processamento de eventos em tempo real em sistemas de big data
- De sensores e processamento de dados de rede à detecção de fraudes e monitoramento de sites e muito mais
- Capacidade de consumir, processar e obter insights de fontes de dados de streaming

Os objetivos para Spark Streaming incluíam:

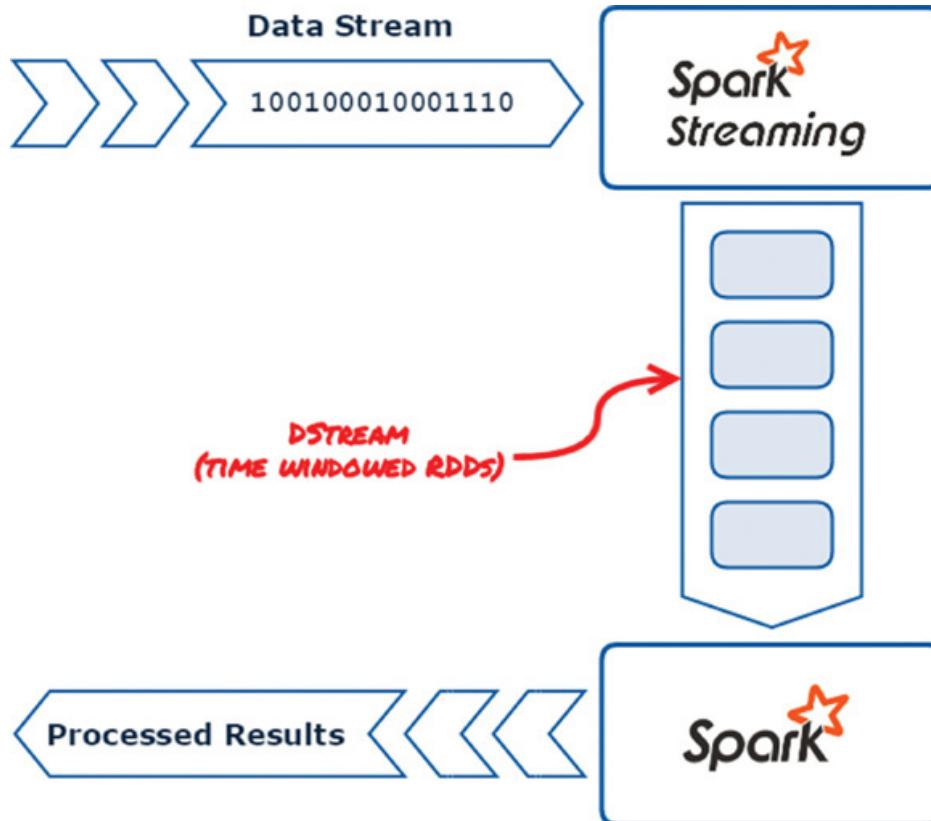
- Latência baixa (segunda escala)
- Processamento de evento único (e apenas uma vez)
- Escalabilidade linear
- Integração com Spark Core API

# Por que?

- Necessidade do processamento de eventos/fluxos como um componente-chave
- Processamento de eventos integrado com sua estrutura de lote baseada em RDD
- A abordagem Spark Streaming
- Spark Streaming apresenta o conceito de "discretized streams" (ou DStreams)
- DStreams são essencialmente lotes de dados armazenados em vários RDDs, cada lote representando uma janela de tempo (normalmente em segundos)
- Os RDDs resultantes podem ser processados usando a API Spark RDD principal e todas as transformações disponíveis



# Macro arq.



# Revisão

Assim como acontece com os pontos de entrada do programa SparkContext, SQLContext e HiveContext que discutímos antes, os aplicativos Spark Streaming também têm um ponto de entrada chamado StreamingContext.

- Aplicação Spark = Driver + grupo de Executors
- Driver executa o processo principal e cria um SparkContext que serve para coordenar a execução do seu job.
- Os executors são processos em execução nos work nodes responsável por executar tasks que o Driver atribuiu a ele.
- O cluster manager (Yarn, Mesos) é responsável pela alocação de recursos para seu aplicativo Spark.

O SparkContext é usado pelo processo do Spark Driver do seu aplicativo Spark para estabelecer uma comunicação com o cluster e o cluster manager (Yarn) para então coordenar e executar jobs.

SparkContext também permite o acesso a outros dois contextos, ou seja, SQLContext e HiveContext.

Para criar um SparkContext, primeiro você precisa criar uma configuração, chamada de SparkConf.

# Revisão

- SQLContext é o ponto de entrada para SparkSQL, que é um módulo Spark para processamento de dados estruturados.
- Depois que o SQLContext é inicializado, o usuário pode usá-lo para realizar várias operações "semelhantes a sql" em conjuntos de dados e dataframes.
- Para criar um SQLContext, primeiro você precisa instanciar um SparkContext.
- Se o seu aplicativo Spark precisa se comunicar com o Hive e você está usando o Spark <2.0, provavelmente precisará de um HiveContext.

# Revisão

# PySpark

```
from pyspark import SparkContext, HiveContext

conf = SparkConf().setAppName ('app').setMaster (master)
sc = SparkContext (conf)
hive_context = HiveContext (sc)
hive_context.sql ("select * from tableName limit 0")
```

# Revisão

// Scala – Acessando o Context

```
import org.apache.spark.{SparkConf, SparkContext}  
import org.apache.spark.sql.hive.HiveContext  
  
val sparkConf = new SparkConf().setAppName("app").setMaster("yarn")  
val sc = new SparkContext(sparkConf)  
val hiveContext = new HiveContext(sc)  
hiveContext.sql("select * from tableName limit 0")
```

# Revisão

O Spark 2.0 introduziu o `SparkSession` que substituiu essencialmente `SQLContext` e `HiveContext` e concede acesso imediato ao `SparkContext`.

Criar um Spark Session com suporte a Hive:

```
# PySpark
```

```
from pyspark.sql import SparkSession
spark_session = SparkSession.builder.enableHiveSupport().getOrCreate()
# Duas maneiras de acessar o contexto do spark a partir da sessão do spark
spark_context = spark_session._sc
spark_context = spark_session.sparkContext
```

# StreamContext

- O StreamingContext representa uma conexão a uma plataforma ou cluster Spark usando um SparkContext existente
- O StreamingContext é usado para criar os datasources de DStreams controlar a computação de streaming e as transformações de DStream.
- O StreamingContext também especifica o argumento batchDuration, que é um intervalo de tempo em segundos pelo qual os dados de streaming serão divididos em lotes.
- Depois de instanciar um StreamingContext, você criaria uma conexão com um fluxo de dados e definiria uma série de transformações a serem realizadas.
- O método start() ou ssc.start() é usado para acionar a os dados de entrada depois que um StreamingContext é estabelecido.
- O StreamingContext pode ser interrompido programaticamente usando ssc.stop()ou ssc.awaitTermination().

# StreamContext

- Discretized streams (DStreams) são o objeto de programação básico na API Spark Streaming
- Streams representam uma sequência contínua de RDDs que são criados a partir de um fluxo contínuo de dados
- DStreams podem ser criados a partir de fontes de dados de streaming, como soquetes TCP, sistemas de mensagens, APIs de streaming (como a API de streaming do Twitter) e muito mais.
- DStreams (como uma abstracção RDD) também pode ser produzido a partir de transformações realizadas na DStreams existentes (tais como map, flatMap e outras operações).

DStreams oferece suporte a dois tipos de operações:

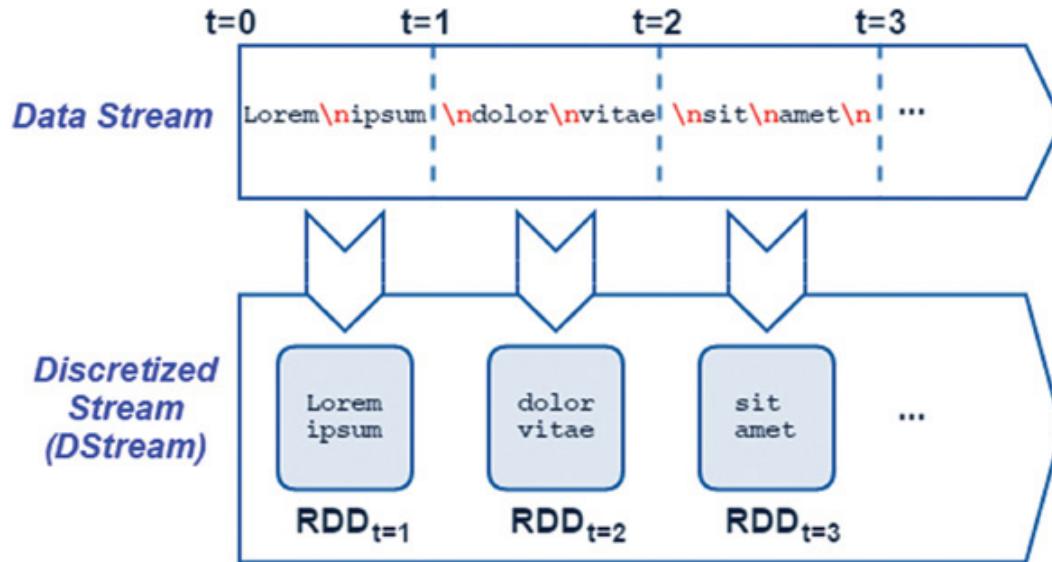
1. Imagem Transformações

2. Imagem Operações de saída

# StreamContext

DStreams são Lazy Evaluation assim como Spark RDD

Representação de um discretized stream, com cada intervalo  $t$  representando uma janela de tempo especificada pelo batchDuration argumento na instanciação de StreamingContext:



# StreamContext

- DStreams Source são definidos em um StreamingContext para um fluxo de dados de entrada especificado, da mesma forma que os RDDs são criados para uma fonte de dados de entrada em um SparkContext
- Muitas fontes de entrada de streaming comuns estão incluídas na API de streaming, como fontes para ler dados de um soquete TCP ou para ler dados enquanto eles estão sendo gravados no HDFS
- As fontes básicas de dados de entrada para a criação de DStreams são descritas aqui:  
**socketTextStream()**
- StreamingContext.socketTextStream (hostname, port, storageLevel = StorageLevel (True, True, False, False, 2))
- O método socketTextStream é usado para criar um DStream a partir de uma fonte TCP de entrada definida pelos argumentos hostname e port.
- Os dados recebidos são interpretados usando a encoding UTF8, com terminação de nova linha usada para definir novos registros.
- O storageLevel argumento que define o storage level padrão MEMORY\_AND\_DISK\_SER\_2

# StreamContext

```
from pyspark.streaming import StreamingContext
```

```
ssc = StreamingContext(sc, 1)
lines = ssc.socketTextStream('localhost', 9999)
counts = lines.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b)
counts.pprint()
ssc.start()
ssc.awaitTermination()
```

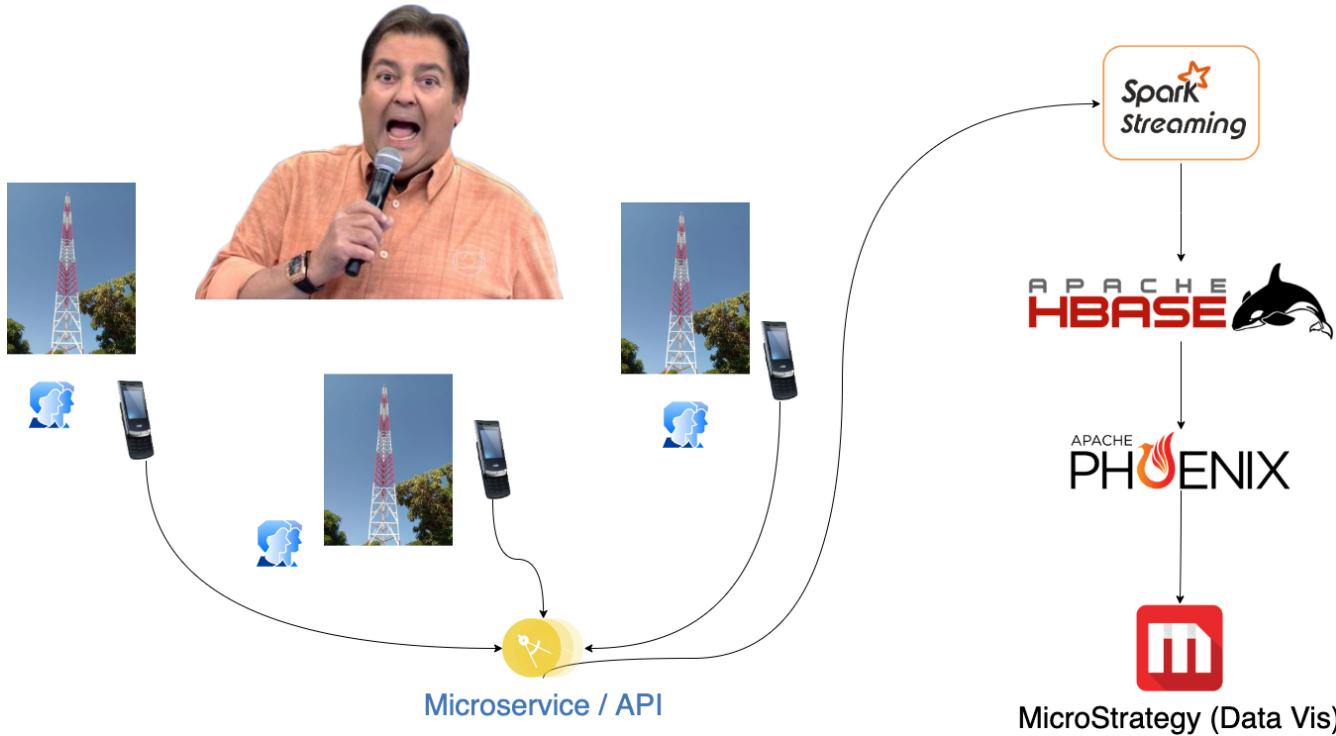
# Parte 3: Casos de Uso

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming



# Telecom

"Torpedão do Faustão"



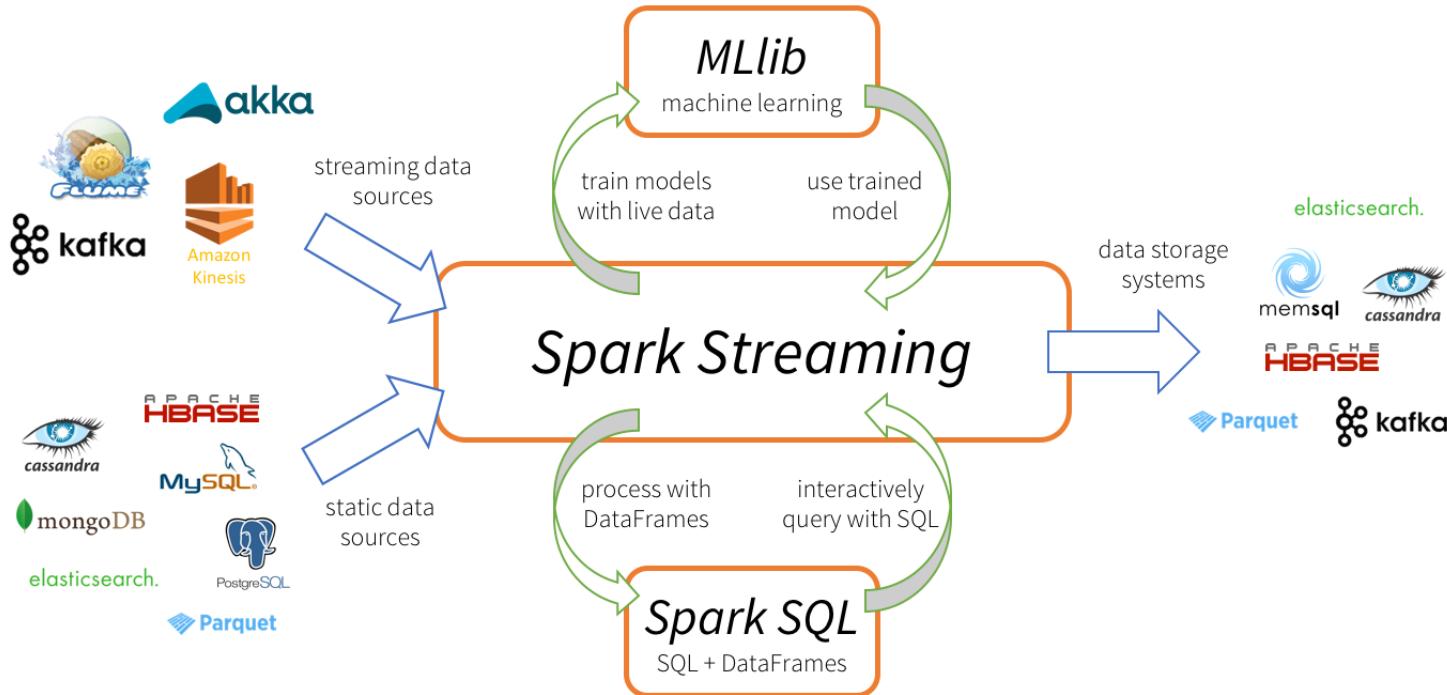


# Compatibilidade





# Compatibilidade





# eBay

## Apache Spark at **ebay**



eBay uses Apache Spark to provide targeted offers, enhance customer experience



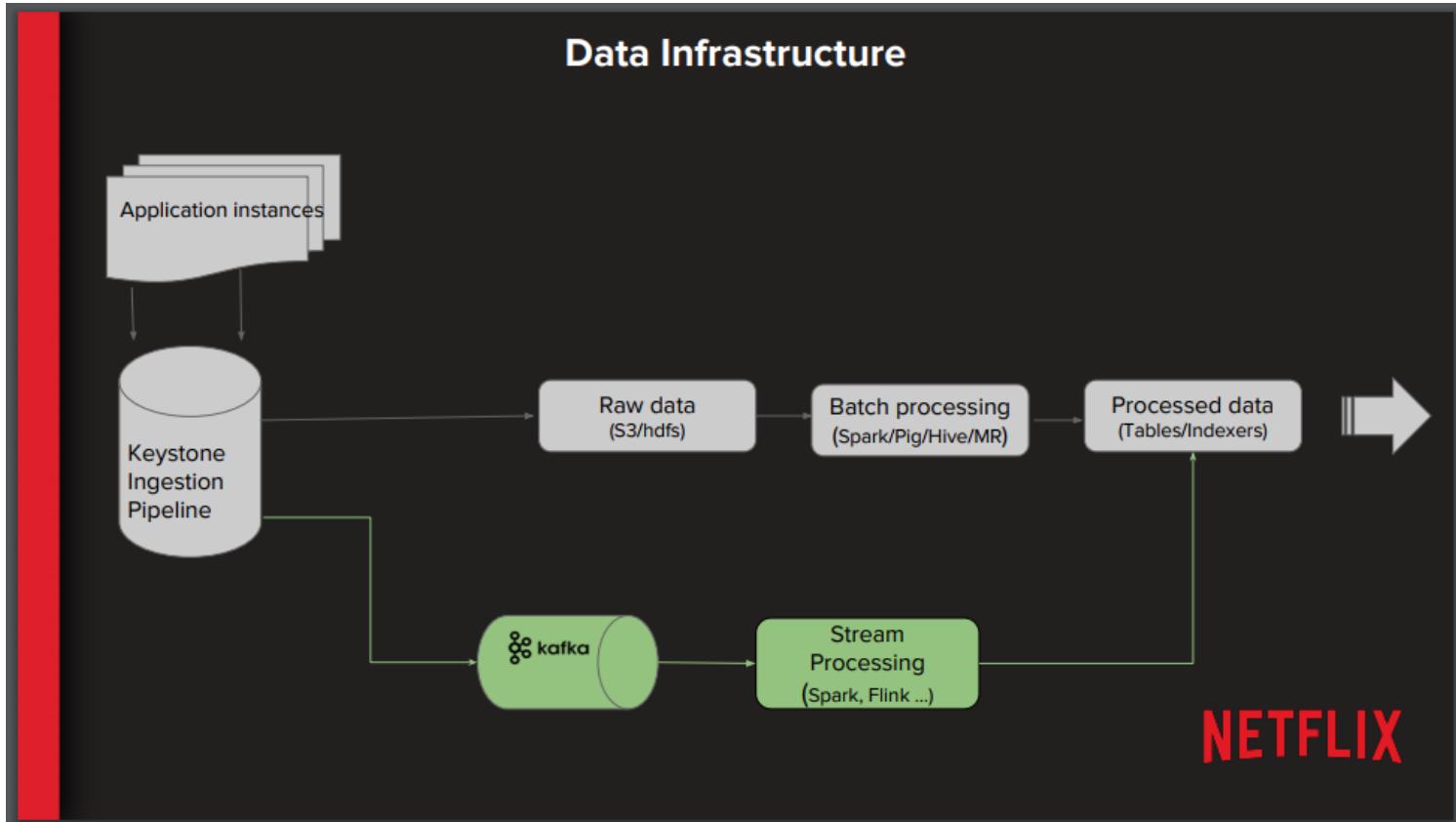
Apache Spark is leveraged at eBay through Hadoop YARN. YARN manages all the cluster resources to run generic tasks



eBay spark users leverage the Hadoop clusters in the range of 2000 nodes, 20,000 cores and 100TB of RAM through YARN



# eBay

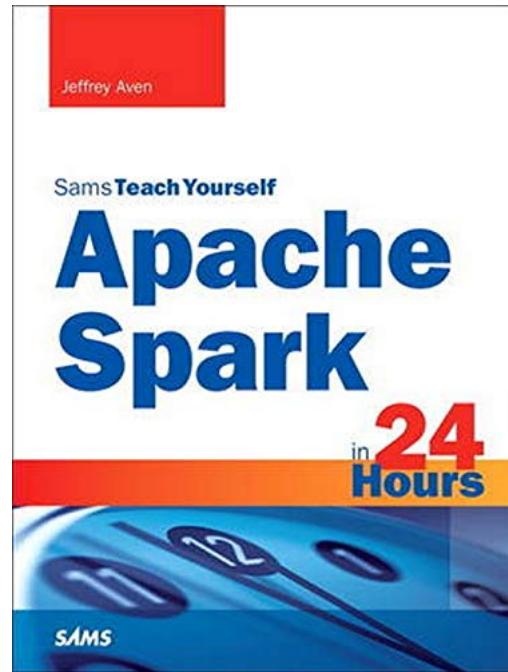
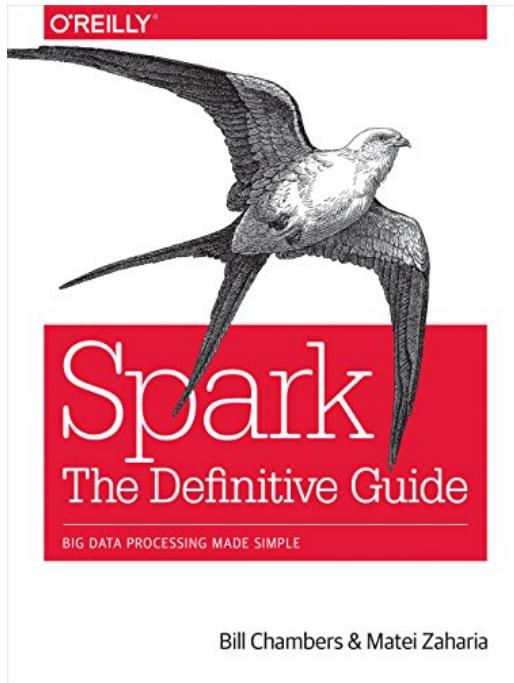


# Parte 4: Encerramento

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming

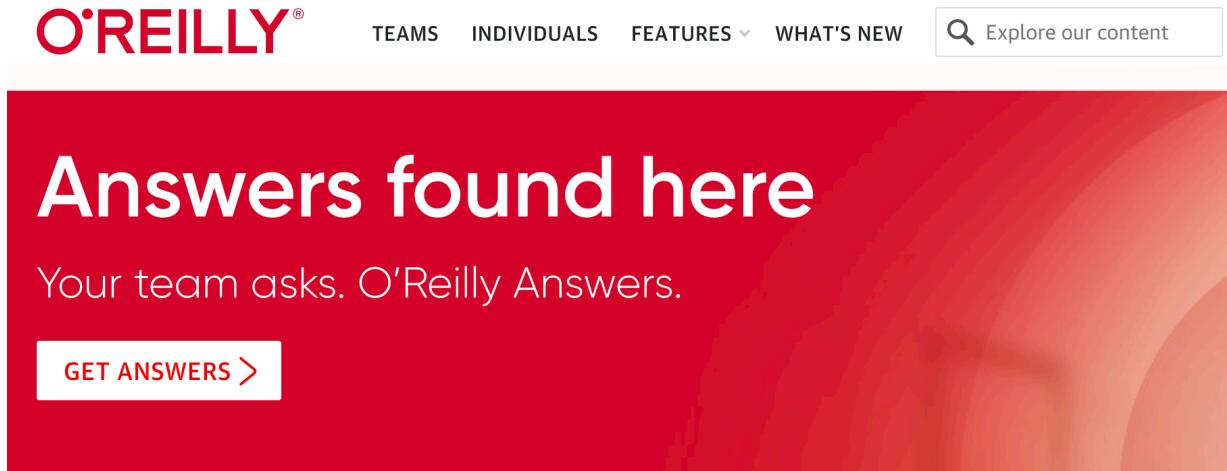


# Referências



# Referências

Safari Books: <https://www.safaribooksonline.com/>



The image shows the homepage of O'Reilly Answers. At the top, there's a navigation bar with the O'Reilly logo, 'TEAMS', 'INDIVIDUALS', 'FEATURES', 'WHAT'S NEW', and a search bar labeled 'Explore our content'. Below the navigation is a large red banner with the text 'Answers found here' and 'Your team asks. O'Reilly Answers.' followed by a 'GET ANSWERS >' button. The background of the main content area has abstract orange and red wavy patterns.

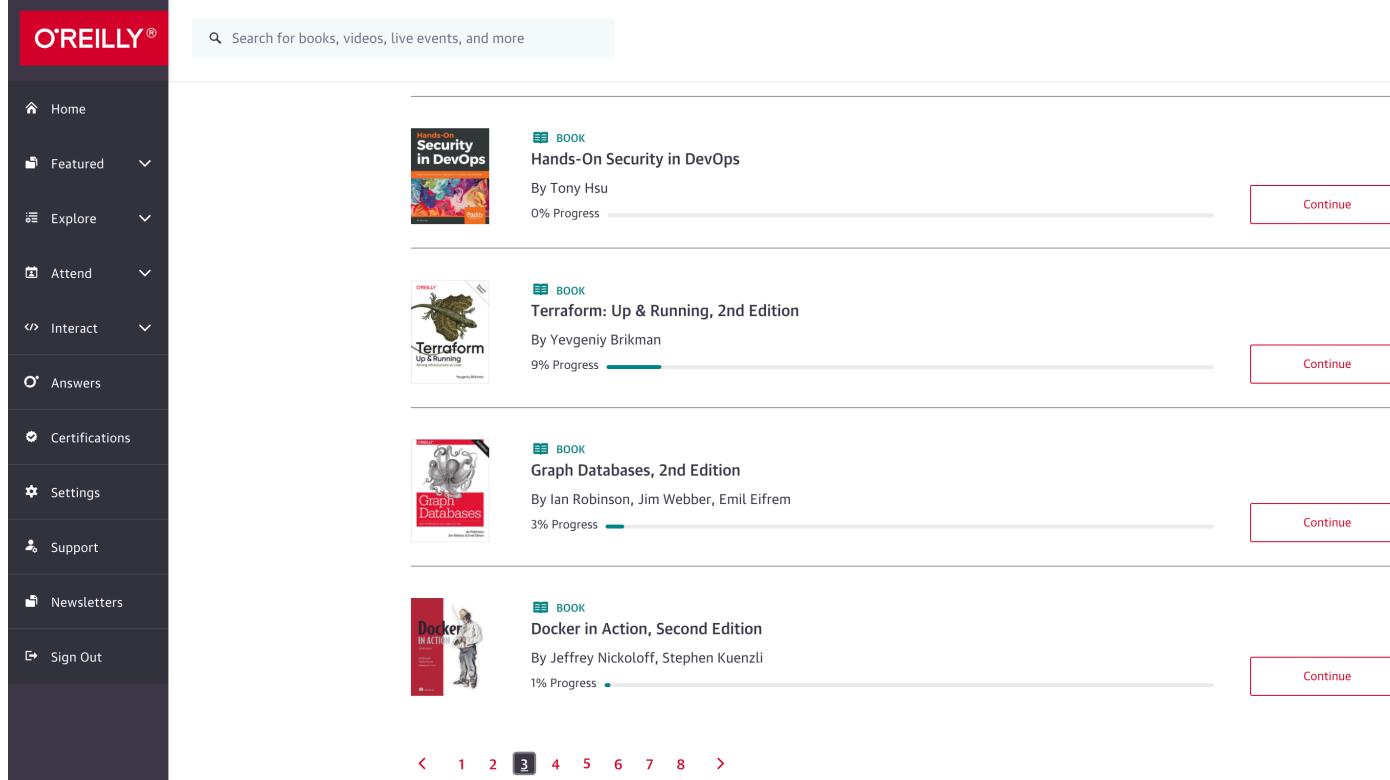
It's essential for your teams to stay ahead of the latest tech. And they need to be able to solve problems in the flow of work and get back to it fast. 66% of Fortune 100 companies count on O'Reilly to help their teams do just that.

Find out how to keep ahead of the curve

LET'S TALK >

# Referências

Safari Books: <https://www.safaribooksonline.com/>



The screenshot shows the Safari Books Online interface. On the left is a dark sidebar with the O'REILLY logo at the top and a search bar below it. The sidebar contains links for Home, Featured, Explore, Attend, Interact, Answers, Certifications, Settings, Support, Newsletters, and Sign Out. The main content area displays four book entries:

- Hands-On Security in DevOps** by Tony Hsu. Progress: 0%. Continue button.
- Terraform: Up & Running, 2nd Edition** by Yevgeniy Brikman. Progress: 9%. Continue button.
- Graph Databases, 2nd Edition** by Ian Robinson, Jim Webber, Emil Eifrem. Progress: 3%. Continue button.
- Docker in Action, Second Edition** by Jeffrey Nickoloff, Stephen Kuenzli. Progress: 1%. Continue button.

Pagination controls at the bottom include arrows for navigation and page numbers 1 through 8, with page 3 highlighted.

# Referências

<https://academy.databricks.com/exam/INT-ADAS-v2-CT>

## Preparation

The following Databricks courses should help you prepare for this exam:

- DB 105 - Apache Spark Programming
- Quick Reference: Spark Architecture
- Future self-paced course on the Spark DataFrames API

In addition, Sections I, II, and IV of *Spark: The Definitive Guide* should also be helpful in preparation.



# Certificações

Cloudera: <https://www.cloudera.com/about/training/certification/cca-spark.html>

The screenshot shows the Cloudera website for the CCA Spark and Hadoop Developer certification. The header features the Cloudera logo and navigation links for Why Cloudera, Products, Solutions, and Services & Support. Below the header is a large banner with a blurred server background. The banner text reads "CCA Spark and Hadoop Developer" and "Prove Your Skills. Build Your Career." A "Schedule Your Exam" button is visible. At the bottom of the banner, the text "CCA Spark and Hadoop Developer" is repeated.

CCA Spark and Hadoop Developer

## CCA Spark and Hadoop Developer Exam (CCA175)

- **Number of Questions:** 8-12 performance-based (hands-on) tasks on Cloudera Enterprise cluster. See below for full cluster configuration
- **Time Limit:** 120 minutes
- **Passing Score:** 70%
- **Language:** English
- **Price:** USD \$295

### Purchase

Watch a free [OnDemand course](#) to help prepare for your certification

Have questions? Read our [Certification FAQ](#)

[Verify a certification](#)

Contact us at [certification@cloudera.com](mailto:certification@cloudera.com)

# Certificações

Cloudera: <https://www.cloudera.com/about/training/certification/ccp-data-engineer.html>

**CLOUDERA**

Why Cloudera

Products

Solutions

Services & Support



## CCP Data Engineer

An experienced open-source developer who earns the Cloudera Certified Data Engineer credential is able to perform core competencies required to ingest, transform, store, and analyze data in Cloudera's CDH environment. The credential is earned after successfully passing the CCP Data Engineer Exam (DE575).

[Schedule your exam](#)

CCP Data Engineer

### CCP Data Engineer Exam (DE575)

- **Number of Questions:** 5-10 performance-based (hands-on) tasks on pre-configured Cloudera Enterprise cluster.
- **Time Limit:** 240 minutes
- **Passing Score:** 70%
- **Language:** English
- **Price:** USD \$400

[Purchase](#)

Have questions? Read our [Certification FAQ](#)

[Verify a Certification](#)

Contact us at [cetification@cloudera.com](mailto:cetification@cloudera.com)

# Certificações

Databricks: <https://academy.databricks.com/category/certifications>

## Assessments

### Databricks Certified Associate Developer for Apache Spark 3.0 - Assessment

The Databricks Certified Associate Developer for Apache Spark 3.0 certification exam assesses an understanding of the basics of the Spark architecture and the ability to apply the Spark DataFrame API to complete individual data manipulation tasks.

\$ 200.00 USD



VIEW

### Databricks Certified Associate Developer for Apache Spark 2.4 - Assessment

The Databricks Certified Associate Developer for Apache Spark 2.4 certification exam assesses an understanding of the basics of the Spark architecture and the ability to apply the Spark DataFrame API to complete individual data manipulation tasks.

\$ 200.00 USD



VIEW

### Databricks Certified Associate ML Practitioner for Apache Spark 2.4 - Assessment

The Databricks Certified Associate ML Practitioner for Apache Spark 2.4 certification exam assesses the understanding of and ability to apply machine learning techniques using the Spark ML library.

\$ 200.00 USD



VIEW

### Databricks Certified Professional Data Scientist - Assessment

The Databricks Certified Professional Data Scientist certification exam assesses the understanding of the basics of machine learning, the steps in the machine learning lifecycle, the understanding of basic machine learning algorithms and techniques, and the understanding of the basics of machine learning model management.

\$ 200.00 USD



VIEW

### Databricks Certified Professional Data Engineer - Assessment

Coming soon (early 2021).

\$ 200.00 USD



VIEW

# Dúvidas?

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming

# Aceleração Global Dev #4 everis

## Criando pipelines de dados eficientes - Parte 2

### Spark SQL - PySpark

---

Marco Antonio Pereira  
Expert Data Architect

# Objetivos da Aula

**1.** Apresentar Spark SQL

**2.** Usabilidade

**3.** Casos de Uso

# Requisitos Básicos

- ✓ Máquina Virtual com Apache Spark 2.4
- ✓ Conta na Community Databricks
- ✓ Conhecimentos básicos em Python ou Scala
- ✓ Persistência

# Parte 1: Falando sobre Dados

Criando pipelines de dados eficientes - Parte 2  
Spark SQL



# Dados



Quero criar um painel de movimentação de vendas por produto.

Neste painel, quero poder ver:

- O produto
- Preço do produto
- Quantidade vendida por produto
- Impostos/Taxas

Para comparar com um período anterior em até 10 anos  
em uma visão diária, mensal e anual

O balanço do dia atual é fechado às 7h do dia posterior.

# Dados

Que tipo de informação podemos extrair?

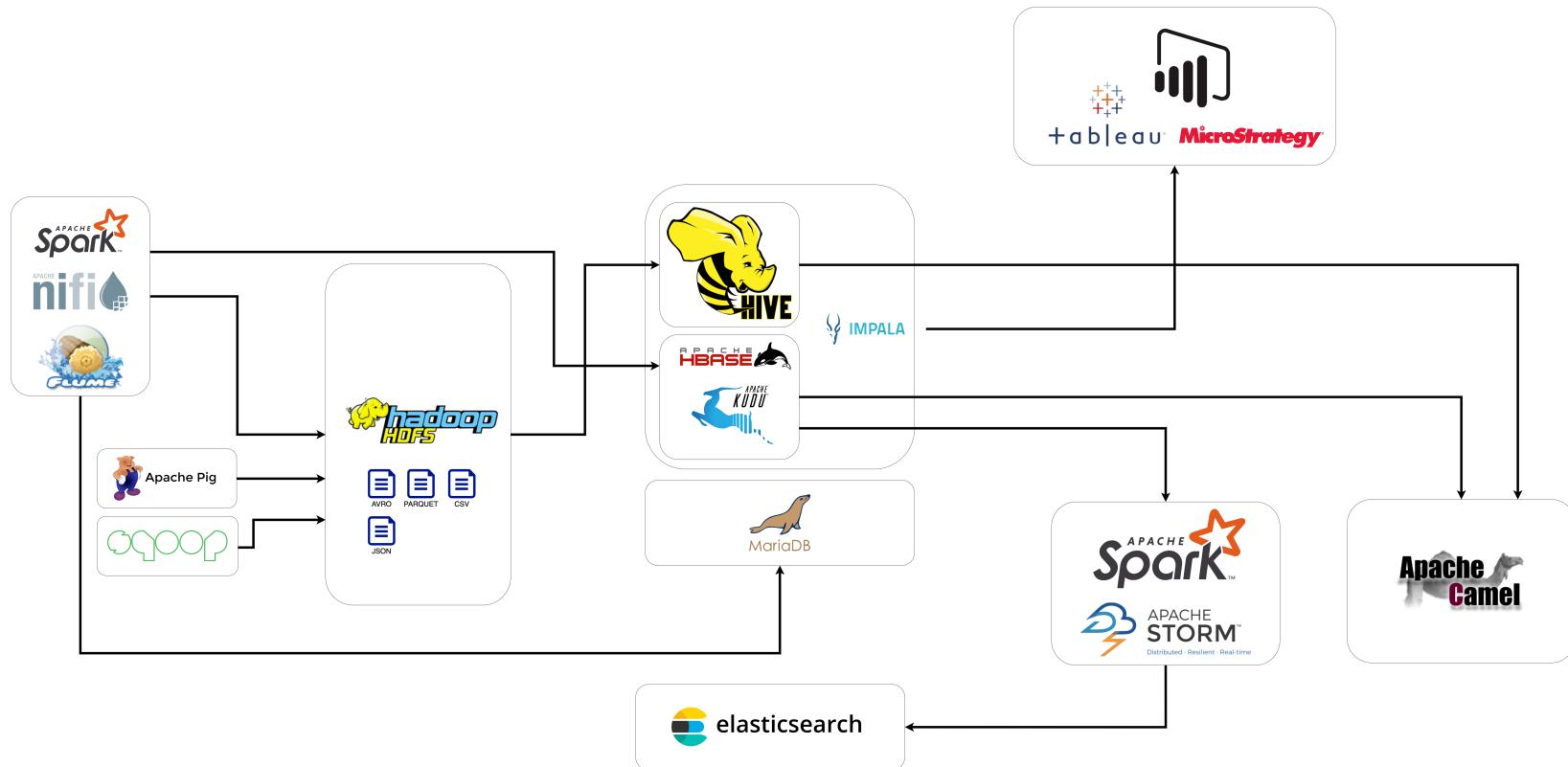
Lógico:

1. Procurar/mapear informações de produtos em alguma base de dados de produto
2. Criar histórico/base vegetativa de até 10 anos
3. Possível ingestão de dados Batch (dia -1 = dados de ontém até -10 anos)

Físico:

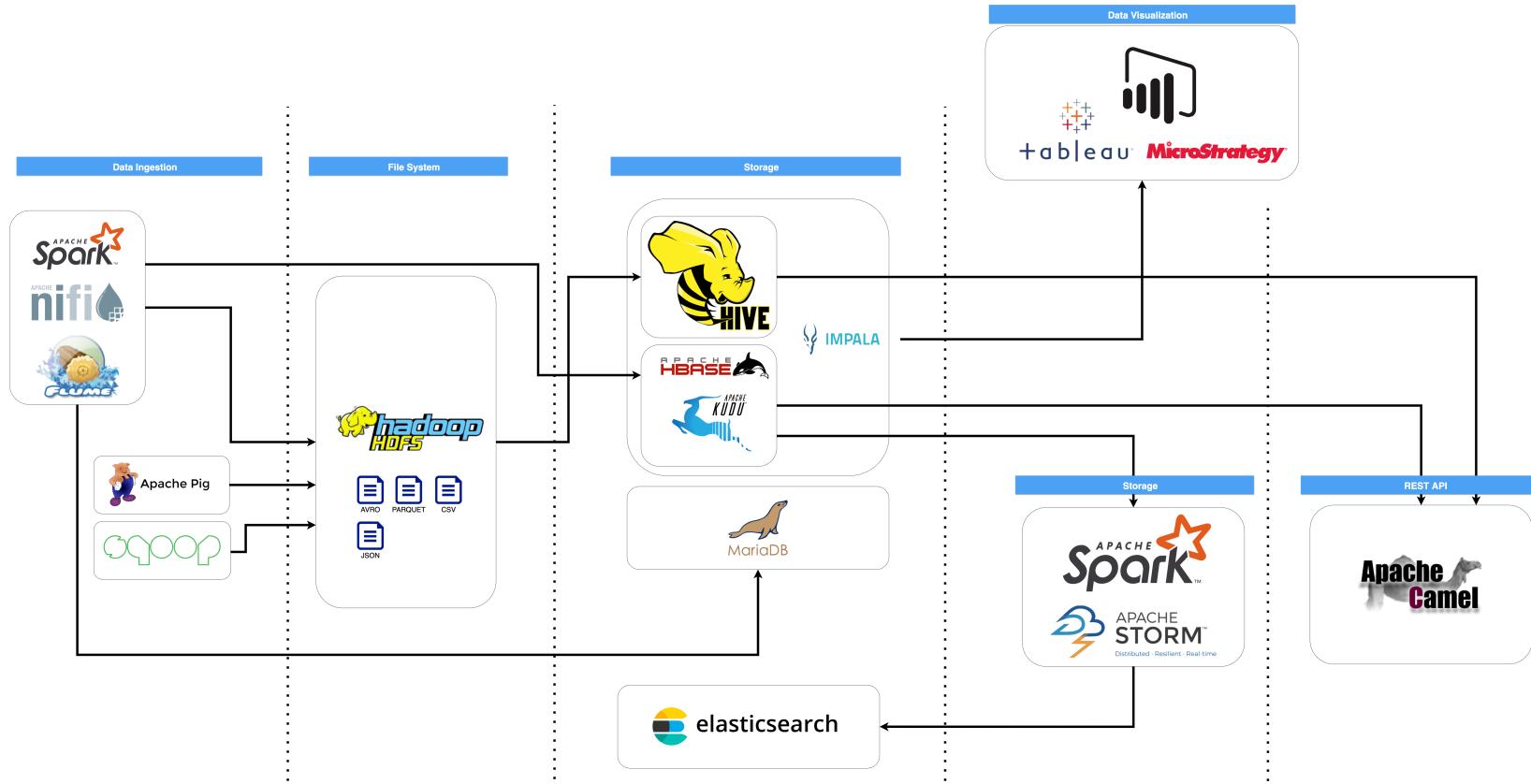
1. Qual a melhor estratégia de particionamento?
2. Qual banco de dados devo utilizar?
3. Qual tipo de arquivo de arquivo devo utilizar?
4. Qual tipo de compressão devo utilizar?

# Arquitetura



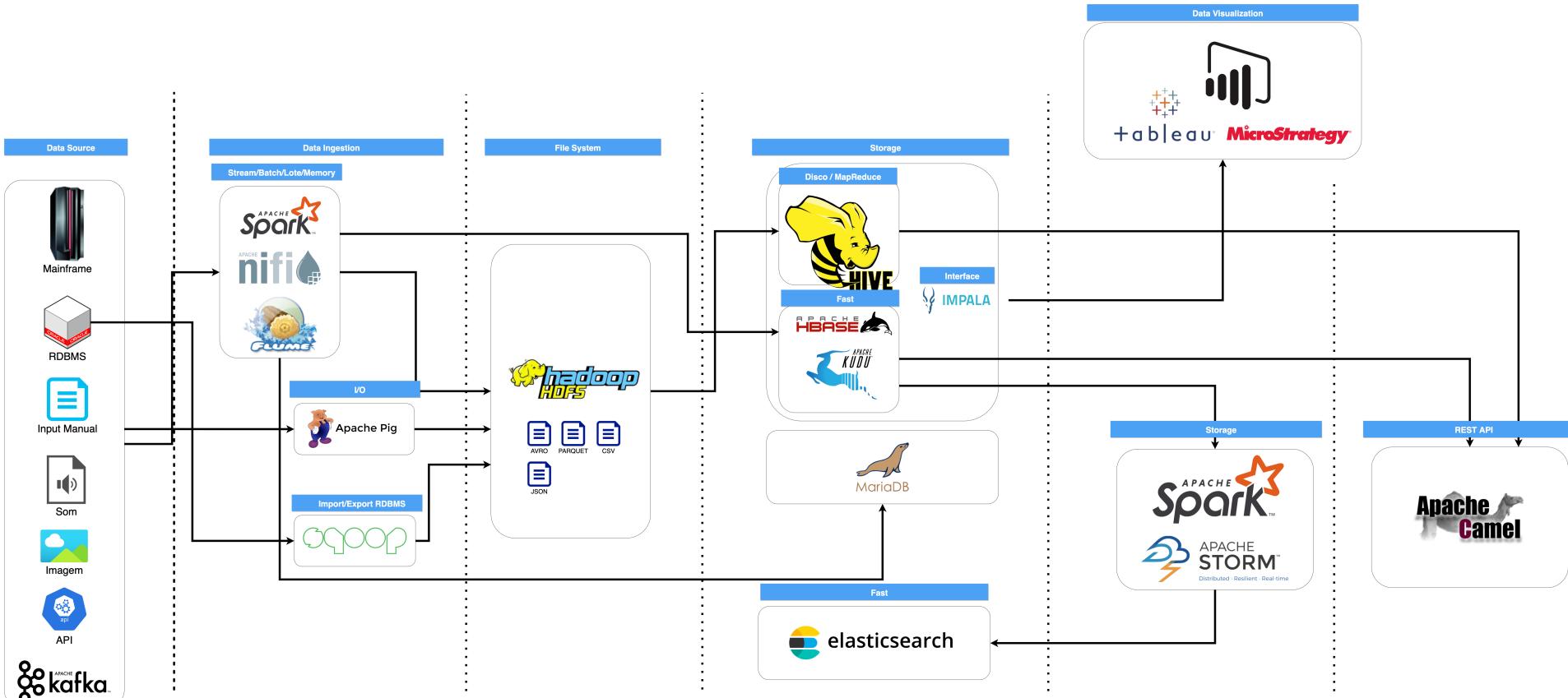


# Arquitetura



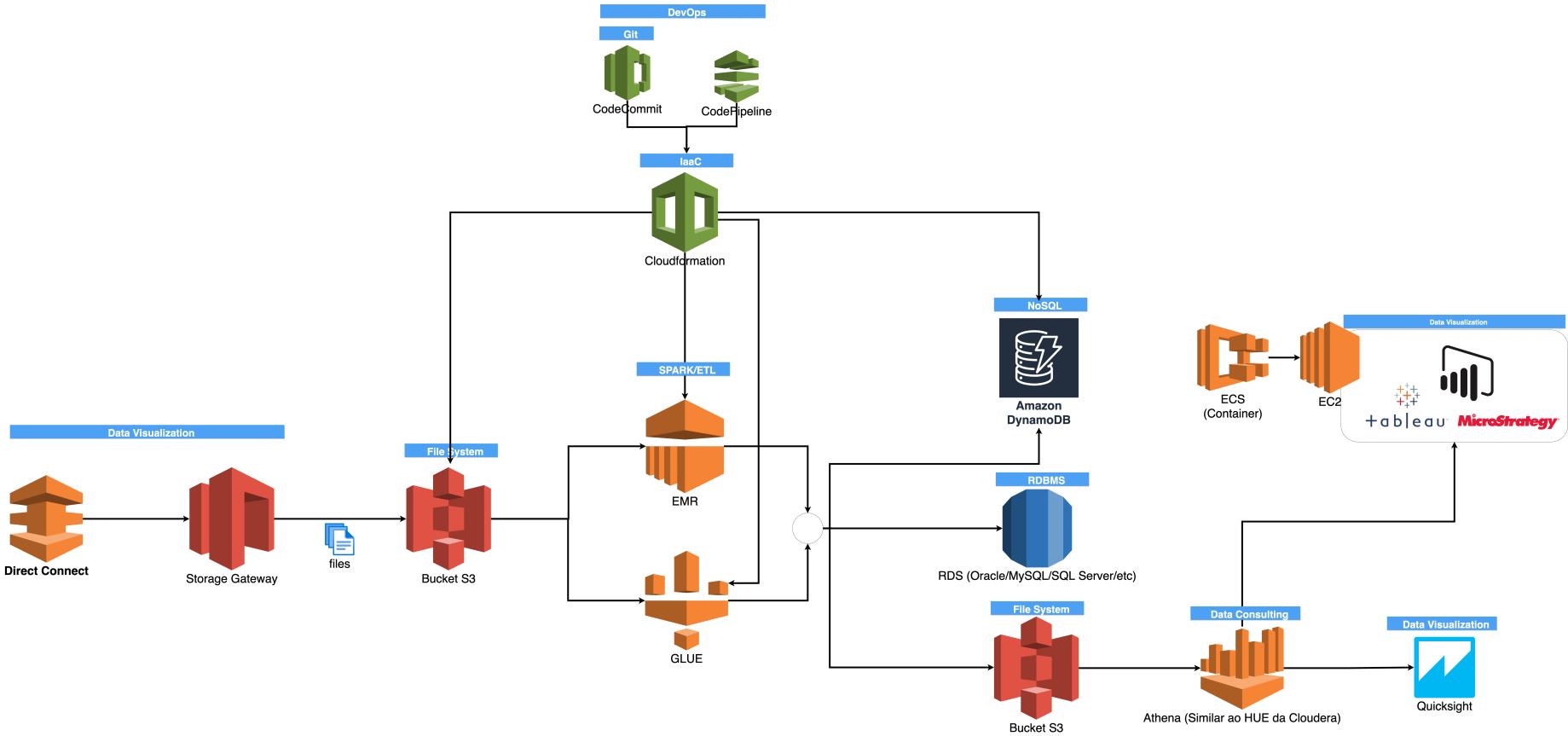


# On Premises





# Cloud (AWS)



# Cloud (AWS)

Guia de custo e configuração de máquinas: <https://aws.amazon.com/pt/ec2/pricing/on-demand/>

Spark:

Node	Máquina	CPU	Memória	Armazenamento	Custo/Hora
Namenode	m5d.4xlarge	16	64GB	2 x 300 SSD NVMe	1,44 USD
Datanode 1	m5d.4xlarge	16	64GB	2 x 300 SSD NVMe	1,44 USD
Datanode 2	m5d.4xlarge	16	64GB	2 x 300 SSD NVMe	1,44 USD
Datanode 3	m5d.4xlarge	16	64GB	2 x 300 SSD NVMe	1,44 USD
Datanode 4	m5d.4xlarge	16	64GB	2 x 300 SSD NVMe	1,44 USD

# Cloud (AWS)

RDS

Tipo	Máquina	Custo/Hora
RDS/Oracle	db.r5.2xlarge	3,1152 USD

DynamoDB: <https://aws.amazon.com/pt/dynamodb/pricing/on-demand/>

Tipo	Custo
gravação	1,875 USD por milhão de unidades de solicitação de gravação
leitura	0,375 USD por milhão de unidades de solicitação de gravação

CloudWatch (Logs): <https://aws.amazon.com/pt/cloudwatch/pricing/>

Tipo	Primeiras Primeiras 10.000 métricas
CloudWatch	0,30 USD / mês

# Camadas



RAW ZONE

Todo dado recebido pelos sistemas origens classificamos como Raw Zone. Realizamos ingestões nesta camada com todos os tipos STRING. O tipo STRING é o tipo mais genérico de todos. Ao realizarmos ingestões com o tipo String, garantimos que o dado AS-IS é armazenado. Caso arquivos provenientes do sistema origem cheguem com algum tipo de anormalidade tal como caracteres especiais, formatos inválidos, entre outros, o tipo STRING é flexível o suficiente para armazenar o dado como ele vem da origem.



TRUSTED ZONE

Na camada Trusted realizamos ingestões mais direcionada a regras de negócio. Após o cruzamento de tabelas do tipo Raw, realizamos transformações de CAST, por exemplo, realizamos JOIN's e aplicamos regras de negócio. A ingestão final deve contemplar os data-types corretos, ou seja, um campo valorado decimal em Raw é STRING porém aqui ele é Decimal.



REFINED ZONE

Na ultima camada, Refined, criamos visões para relatórios. Criamos tabelas do tipo Refined selecionando apenas campos específicos para as visões de relatório. Então se o resultado de uma tabela agregada e tipada, Trusted possuir várias informações sobre um determinado tema e precisamos mostrar no relatório um agrupamento menor de campos, criarmos Refined a partir de Trusted. Caso todos os campos precisem ser mostrados no Dashboard, entendemos que a "Trusted" é a "Refined".

Quando você está trabalhando em um ambiente de Big Data, existem vários formatos de dados. Os dados podem ser formados em um formato legível como arquivo JSON ou CSV, mas isso não significa que essa é a melhor maneira de realmente armazenar os dados.

Existem três formatos de arquivo otimizados para uso em clusters Hadoop:

Optimized Row Columnar (ORC)

Avro

Parquet

# Tipos de armaz.

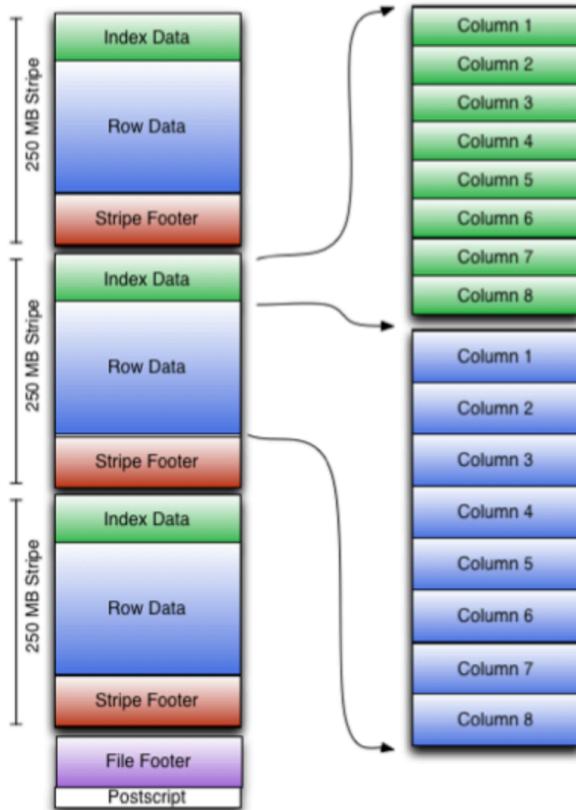
BIG DATA FORMATS COMPARISON



Source: Nexla analysis, April 2018



# Parquet



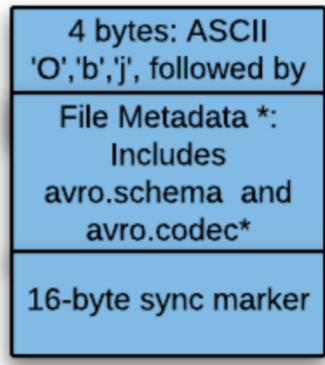
# Parquet

- Orientado por coluna (armazenar dados em colunas): os armazenamentos de dados orientados por coluna são otimizados para cargas de trabalho analíticas pesadas em leitura
- Altas taxas de compressão (até 75% com compressão Snappy)
- Apenas as colunas necessárias seriam buscadas / lidas (reduzindo a E / S do disco)
- Pode ser lido e escrito usando Avro API e Avro Schema

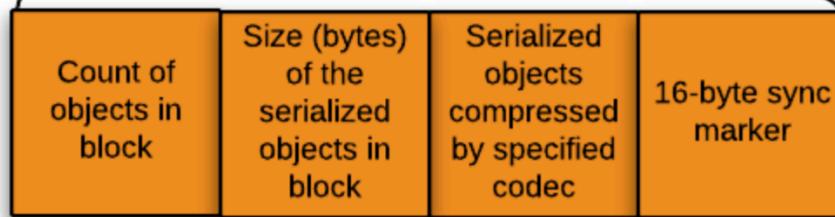
Referência: <http://parquet.apache.org/documentation/latest/>



# Avro



AVRO FILE:



\* File metadata follows:  
{"type": "map", "values":  
"bytes"}

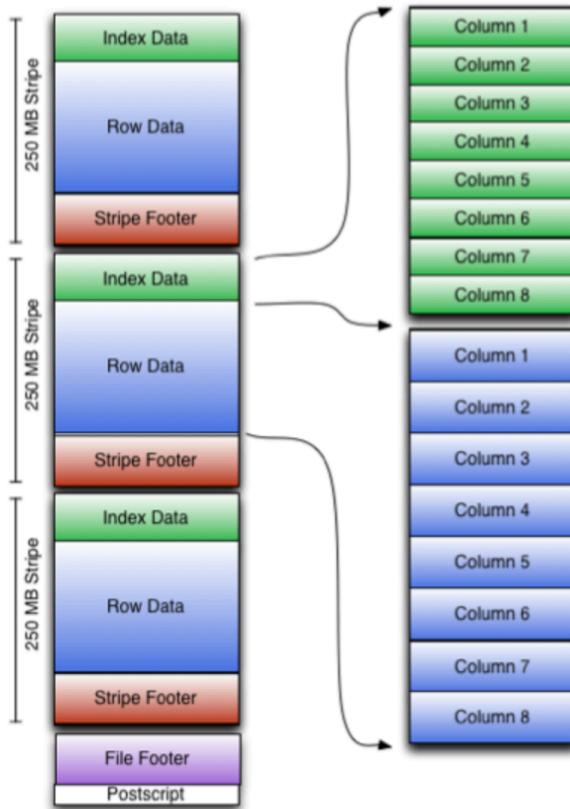
# Avro

- Com base em linha (armazenar dados em linhas): bancos de dados baseados em linha são melhores para cargas de trabalho transacionais pesadas de gravação
- Serialização de suporte
- Formato binário rápido
- Suporta compressão de bloco e divisível
- Evolução do esquema de suporte (o uso de JSON para descrever os dados, enquanto usa o formato binário para otimizar o tamanho do armazenamento)
- Armazena o esquema no cabeçalho do arquivo para que os dados sejam autodescritivos

Referência: <https://avro.apache.org/docs/1.10.1/>



# ORC



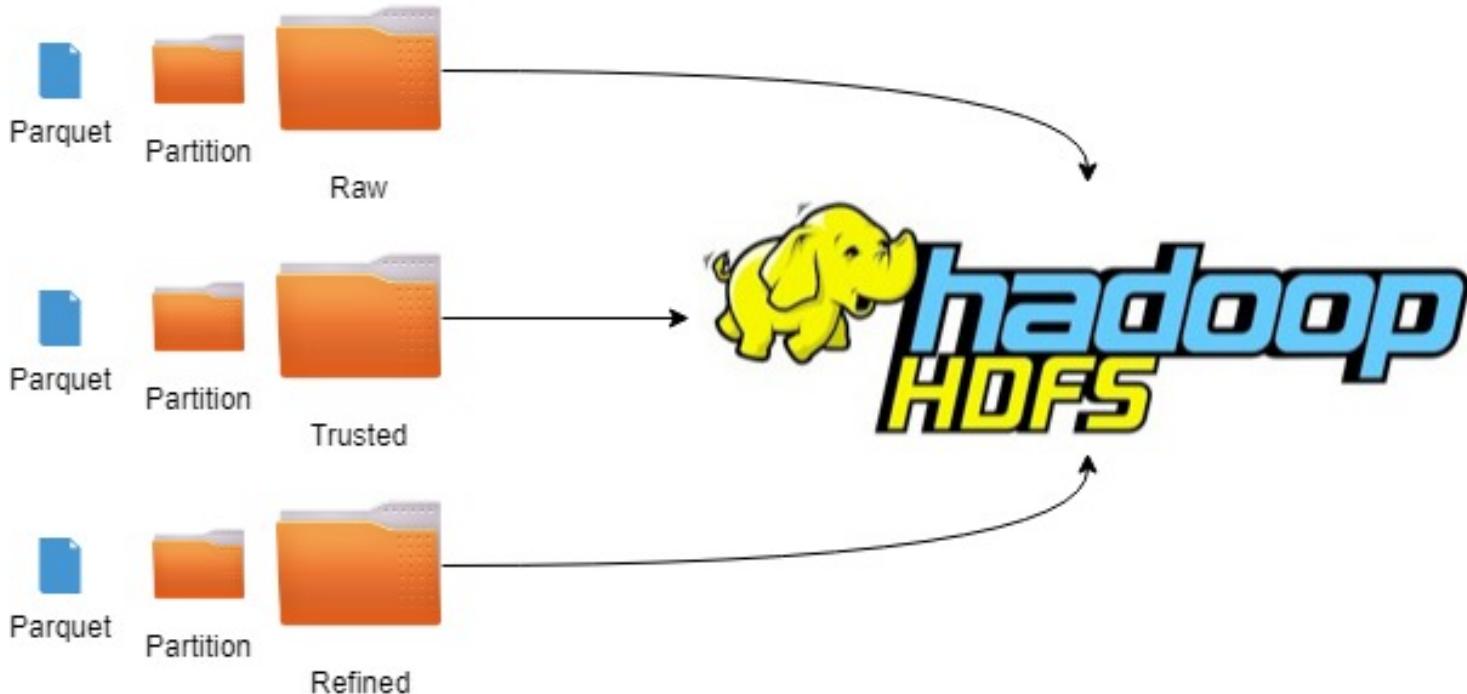
# ORC

- Orientado por coluna (armazenar dados em colunas): os armazenamentos de dados orientados por coluna são otimizados para cargas de trabalho analíticas pesadas em leitura
- Altas taxas de compressão (ZLIB)
- Suporte ao tipo Hive (datetime, decimal e os tipos complexos como struct, list, map e union)
- Metadados armazenados usando buffers de protocolo, que permitem adição e remoção de campos
- Compatível com HiveQL
- Suporte a Serialização

Referência: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC#LanguageManualORC-ORCfiles>



# Camadas



Cada uma das 3 camadas se tratam de subdiretórios dentro do sistema de arquivos distribuídos, HDFS, e podem ser mapeados através da propriedade LOCATION.



# HDFS



Referência dos Schemas, incluindo local/nodes com os arquivos



\$ hdfs dfs -du -h /<path>

Size \* Replication Factory

Arquivos físicos



Apache  
Derby



# Parte 2: Spark SQL

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming

# DataFrame

```
# Leitura de Dataframe
```

```
## Opção 1
```

```
df1 = spark.read.format("csv").option("header","true").load(path_dataset1)
```

```
## Opção 2
```

```
df1 = spark.read.csv(path_dataset1)
```

```
df1 = spark.read.option("header","true").option("inferSchema","true").csv(path_dataset1)
```

```
## Exibindo dataframe
```

```
df1.show()
```

# DataFrame

```
## Outras formas de leitura de arquivos com PySpark
```

```
path = "/../../arquivoXPTO"
```

```
# Criando um dataframe a partir de um JSON
```

```
dataframe = spark.read.json(path)
```

```
# Criando um dataframe a partir de um ORC
```

```
dataframe = spark.read.orc(path)
```

```
# Criando um dataframe a partir de um PARQUET
```

```
dataframe = spark.read.parquet(path)
```



# DataFrame

# Leitura de um RDD

```
rdd = sc.textFile(path_rdd)
```

#rdd.show() = Errado, não é possível exibir um SHOW() de um RDD, somente um Dataframe

```
rdd.collect()
```



# DataFrame

# Criando uma tabela temporária

```
nome_tabela_temporaria = "tempTableDataFrame1"  
df1.createOrReplaceTempView(nome_tabela_temporaria)
```

# DataFrame

```
nome_tabela_temporaria = "tempTableDataFrame1"  
df1.createOrReplaceTempView(nome_tabela_temporaria)
```

```
# Lendo a tabela temporaria opcao 1  
spark.read.table(nome_tabela_temporaria).show()
```

```
# Lendo a tabela temporaria opcao 2  
spark.sql("SELECT * FROM tempTableDataFrame1").show()
```

# DataFrame

# Visualização do Databricks

```
display(spark.sql("SELECT * FROM tempTableDataFrame1"))
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccinated_per_hundred
1	Argentina	ARG	2020-12-29	700	null	null	null	null	0	null
2	Argentina	ARG	2020-12-30	null	null	null	null	15656	null	null
3	Argentina	ARG	2020-12-31	32013	null	null	null	15656	0.07	null
4	Argentina	ARG	2021-01-01	null	null	null	null	11070	null	null
5	Argentina	ARG	2021-01-02	null	null	null	null	8776	null	null
6	Argentina	ARG	2021-01-03	null	null	null	null	7400	null	null
7	Argentina	ARG	2021-01-04	39599	null	null	null	6483	0.09	null
8	Argentina	ARG	2021-01-05	null	null	null	null	7004	null	null

Showing the first 1000 rows.

# DataFrame

```
# Scala
```

```
#import org.apache.spark.sql.functions._
```

```
# Python
```

```
from pyspark.sql.functions import col, column
```

```
# Usando function col ou column
```

```
df1.select(col("country"), col("date"), column("iso_code")).show()
```

```
# Usando selectExpr
```

```
df1.selectExpr("country", "date", "iso_code").show()
```

# DataFrame

```
# Scala import
# org.apache.spark.sql.types._

# Criando um Schema manualmente no PySpark
from pyspark.sql.types import *

dataframe_ficticio = StructType([
    StructField("col_String_1", StringType()),
    StructField("col_Integer_2", IntegerType()),
    StructField("col.Decimal_3", DecimalType())
])
dataframe_ficticio
```

# DataFrame

```
# Função para gerar Schema (campos/colunas/nomes de colunas)
"""

# Scala
org.apache.spark.sql.types._

def getSchema(fields : Array[StructField]) : StructType = {
    new StructType(fields)
}

"""

# PySpark
def getSchema(fields):
    return StructType(fields)

schema = getSchema([StructField("coluna1", StringType()), StructField("coluna2", StringType()), StructField("coluna3", StringType())])
```



# DataFrame

```
#Show
```

```
df1.show(2)
```

```
#Take
```

```
df1.take(2)
```



# DataFrame

# Gravando um novo CSV

```
path_destino="/FileStore/tables/CSV/"  
nome_arquivo="arquivo.csv"  
path_geral= path_destino + nome_arquivo  
df1.write.format("csv").mode("overwrite").option("sep", "\t").save(path_geral)
```



# DataFrame

# Gravando um novo JSON

```
path_destino="/FileStore/tables/JSON/"
nome_arquivo="arquivo.json"
path_geral= path_destino + nome_arquivo
df1.write.format("json").mode("overwrite").save(path_geral)
```



# DataFrame

# Gravando um novo PARQUET

```
path_destino="/FileStore/tables/PARQUET/"  
nome_arquivo="arquivo.parquet"  
path_geral= path_destino + nome_arquivo  
df1.write.format("parquet").mode("overwrite").save(path_geral)
```



# DataFrame

# Gravando um novo ORC

```
path_destino="/FileStore/tables/ORC/"
nome_arquivo="arquivo.orc"
path_geral= path_destino + nome_arquivo
df1.write.format("orc").mode("overwrite").save(path_geral)
```



# DataFrame

```
# Outros tipos de SELECT
```

```
#Diferentes formas de selecionar uma coluna
```

```
from pyspark.sql.functions import *

df1.select("country").show(5)
df1.select('country').show(5)
df1.select(col("country")).show(5)
df1.select(column("country")).show(5)
df1.select(expr("country")).show(5)
```

# DataFrame

```
# Define uma nova coluna com um valor constante
```

```
df2 = df1.withColumn("nova_coluna", lit(1))
```

```
# Adicionar coluna
```

```
teste = expr("total_vaccinations < 40")
```

```
df1.select("country", "total_vaccinations").withColumn("teste", teste).show(5)
```

```
# Renomear uma coluna
```

```
df1.select(expr("total_vaccinations as total_de_vacinados")).show(5)
```

```
df1.select(col("country").alias("pais")).show(5)
```

```
df1.select("country").withColumnRenamed("country", "pais").show(5)
```

```
# Remover uma coluna
```

```
df3 = df1.drop("country")
```

```
df3.columns
```

# DataFrame

```
# Filtrando dados e ordenando
```

```
# where() é um alias para filter().
```

```
# Seleciona apenas os primeiros registros da coluna "total_vaccinations"
```

```
df1.filter(df1.total_vaccinations > 55).orderBy(df1.total_vaccinations).show(2)
```

```
# Filtra por país igual Argentina
```

```
df1.select(df1.total_vaccinations, df1.country).filter(df1.country == "Argentina").show(5)
```

```
# Filtra por país diferente Argentina
```

```
df1.select(df1.total_vaccinations, df1.country).where(df1.country != "Argentina").show(5) # python type
```

# DataFrame

```
# Filtrando dados e ordenando
```

```
# Mostra valores únicos
```

```
df1.select("country").distinct().show()
```

```
# Especificando vários filtros em comando separados
```

```
filtro_vacinas = df1.total_vaccinations < 100
```

```
filtro_pais = df1.country.contains("Argentina")
```

```
df1.select(df1.total_vaccinations, df1.country, df1.vaccines).where(df1.vaccines.isin("Sputnik V", "Sinovac")).filter(filtro_vacinas).show(5)
```

```
df1.select(df1.total_vaccinations, df1.country, df1.vaccines).where(df1.vaccines.isin("Sputnik V",
```

```
"Sinovac")).filter(filtro_vacinas).withColumn("filtro_pais", filtro_pais).show(5)
```



# DataFrame

```
#####
```

Convertendo dados

```
#####
```

```
df5 = df1.withColumn("PAIS", col("country").cast("string").alias("PAIS"))
df5.select(df5.PAIS).show(2)
```

```
#####
```

Trabalhando com funções

```
#####
```

```
# Usando funções
df1.select(upper(df1.country)).show(3)
df1.select(lower(df1.country)).show(4)
```



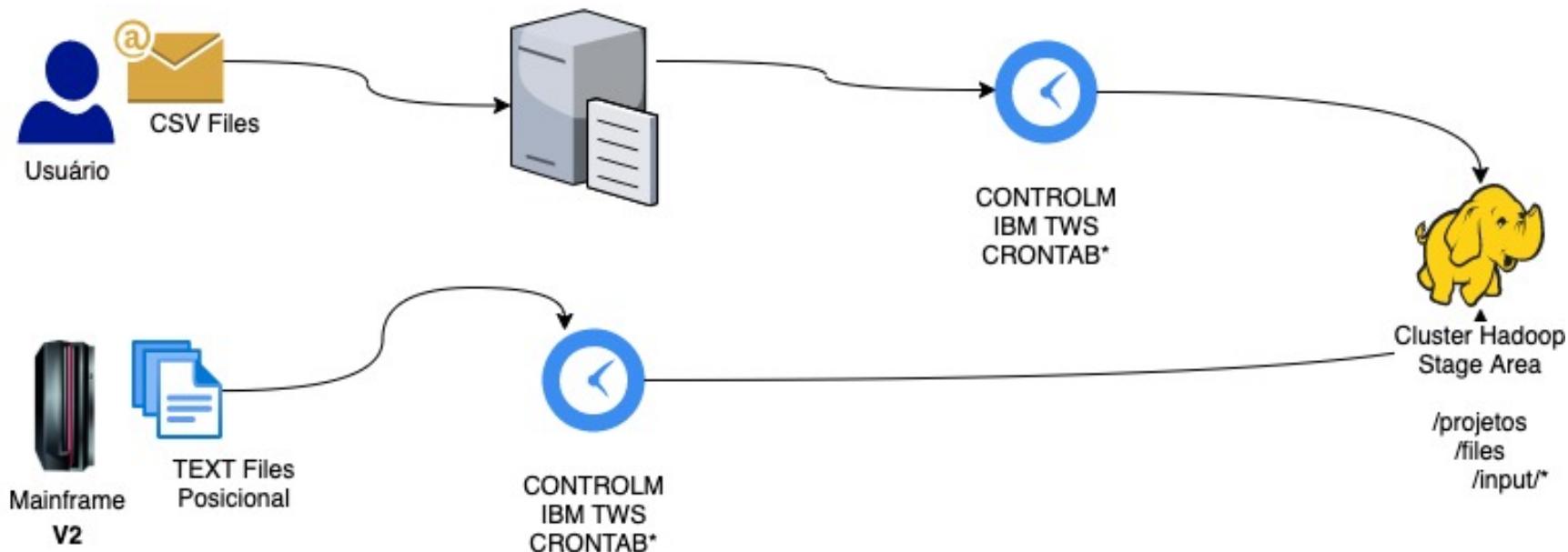
# DataFrame

JOINS

# Parte 3: Casos de Uso

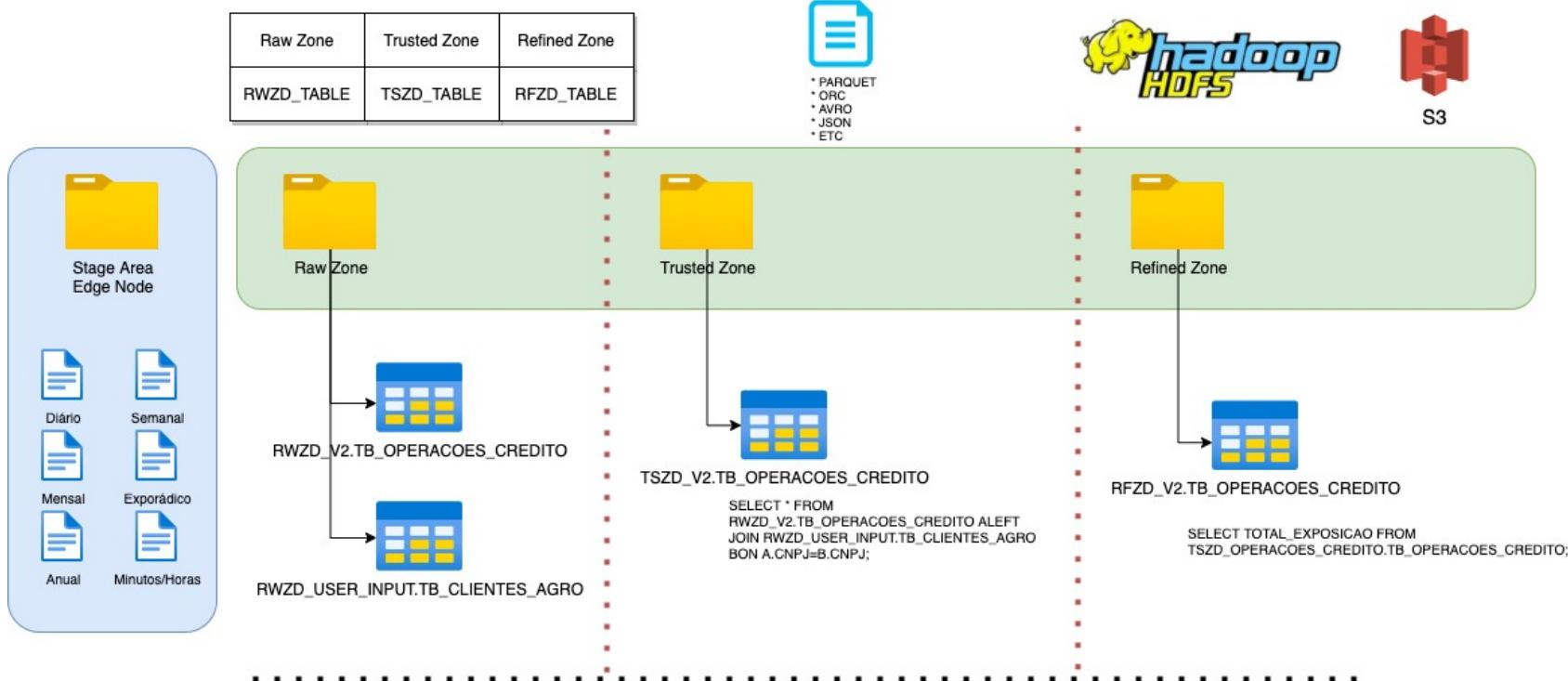
Criando pipelines de dados eficientes - Parte 1  
Spark Streaming

# Caso de Uso 1





# Caso de Uso 2

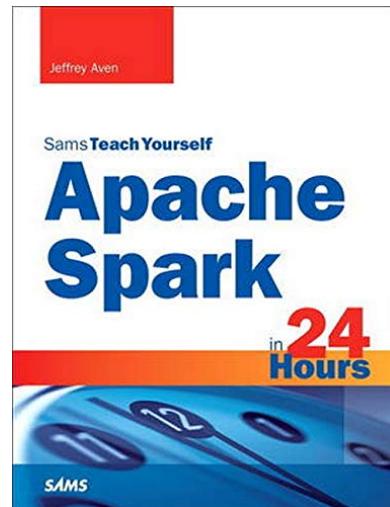
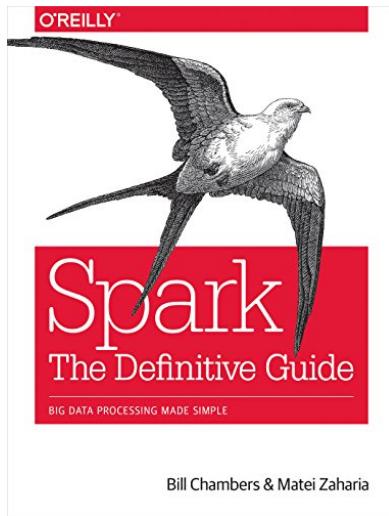


# Parte 4: Encerramento

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming



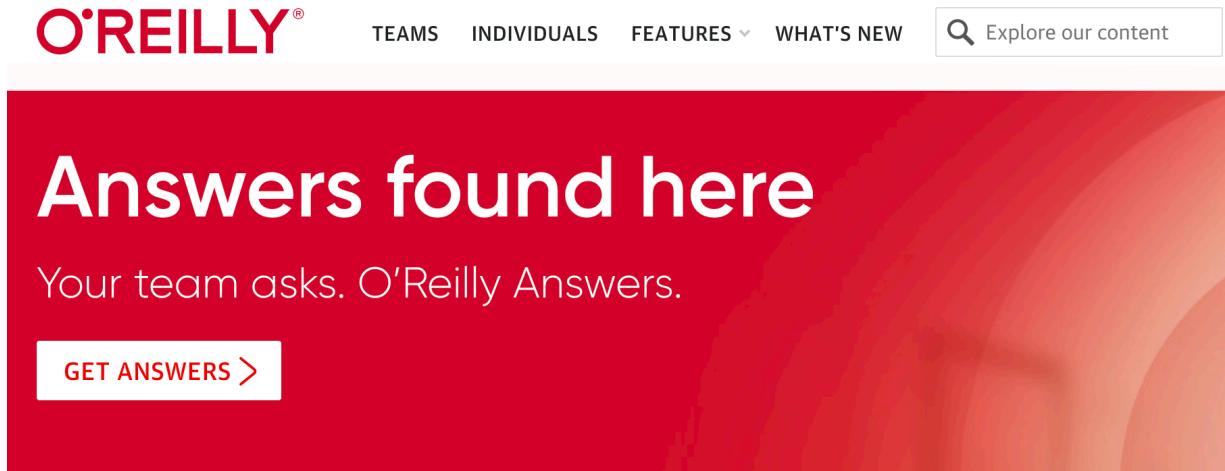
# Referências



Documentação: <https://spark.apache.org/docs/2.4.0/api/python/pyspark.sql.html>

# Referências

Safari Books: <https://www.safaribooksonline.com/>



The image shows the homepage of O'Reilly Answers. At the top, there's a navigation bar with the O'Reilly logo, 'TEAMS', 'INDIVIDUALS', 'FEATURES', 'WHAT'S NEW', and a search bar labeled 'Explore our content'. Below the navigation is a large red banner with the text 'Answers found here' in white, followed by 'Your team asks. O'Reilly Answers.' and a 'GET ANSWERS >' button. The background of the page features abstract orange and red wavy patterns.

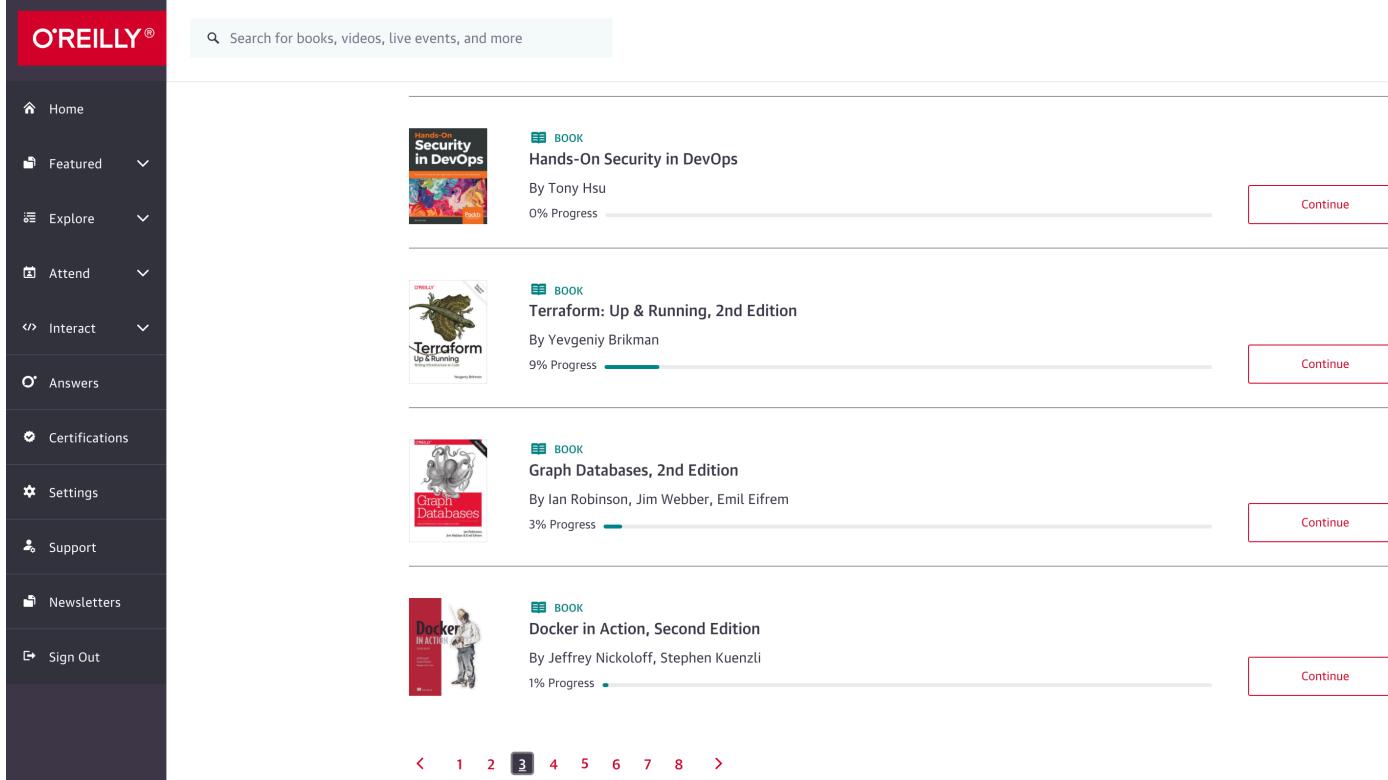
It's essential for your teams to stay ahead of the latest tech. And they need to be able to solve problems in the flow of work and get back to it fast. 66% of Fortune 100 companies count on O'Reilly to help their teams do just that.

Find out how to keep ahead of the curve

LET'S TALK >

# Referências

Safari Books: <https://www.safaribooksonline.com/>



The screenshot shows the Safari Books Online interface. On the left is a dark sidebar with the O'REILLY logo at the top and a search bar below it. The sidebar contains links for Home, Featured, Explore, Attend, Interact, Answers, Certifications, Settings, Support, Newsletters, and Sign Out. The main content area displays four book entries:

- Hands-On Security in DevOps** by Tony Hsu. Progress: 0%. Continue button.
- Terraform: Up & Running, 2nd Edition** by Yevgeniy Brikman. Progress: 9%. Continue button.
- Graph Databases, 2nd Edition** by Ian Robinson, Jim Webber, Emil Eifrem. Progress: 3%. Continue button.
- Docker in Action, Second Edition** by Jeffrey Nickoloff, Stephen Kuenzli. Progress: 1%. Continue button.

Pagination controls at the bottom include arrows for navigation and page numbers 1 through 8, with page 3 highlighted.

# Referências

<https://academy.databricks.com/exam/INT-ADAS-v2-CT>

## Preparation

The following Databricks courses should help you prepare for this exam:

- DB 105 - Apache Spark Programming
- Quick Reference: Spark Architecture
- Future self-paced course on the Spark DataFrames API

In addition, Sections I, II, and IV of *Spark: The Definitive Guide* should also be helpful in preparation.



# Certificações

Cloudera: <https://www.cloudera.com/about/training/certification/cca-spark.html>

The screenshot shows the Cloudera website for the CCA Spark and Hadoop Developer certification. The header features the Cloudera logo and navigation links for Why Cloudera, Products, Solutions, and Services & Support. Below the header is a large banner with a blurred server background. The banner text reads "CCA Spark and Hadoop Developer" and "Prove Your Skills. Build Your Career." A "Schedule Your Exam" button is visible. At the bottom of the banner, the text "CCA Spark and Hadoop Developer" is repeated.

CCA Spark and Hadoop Developer

## CCA Spark and Hadoop Developer Exam (CCA175)

- **Number of Questions:** 8-12 performance-based (hands-on) tasks on Cloudera Enterprise cluster. See below for full cluster configuration
- **Time Limit:** 120 minutes
- **Passing Score:** 70%
- **Language:** English
- **Price:** USD \$295

### Purchase

Watch a free [OnDemand course](#) to help prepare for your certification

Have questions? Read our [Certification FAQ](#)

[Verify a certification](#)

Contact us at [certification@cloudera.com](mailto:certification@cloudera.com)

# Certificações

Cloudera: <https://www.cloudera.com/about/training/certification/ccp-data-engineer.html>

**CLOUDERA**

Why Cloudera

Products

Solutions

Services & Support



## CCP Data Engineer

An experienced open-source developer who earns the Cloudera Certified Data Engineer credential is able to perform core competencies required to ingest, transform, store, and analyze data in Cloudera's CDH environment. The credential is earned after successfully passing the CCP Data Engineer Exam (DE575).

[Schedule your exam](#)

CCP Data Engineer

### CCP Data Engineer Exam (DE575)

- **Number of Questions:** 5-10 performance-based (hands-on) tasks on pre-configured Cloudera Enterprise cluster.
- **Time Limit:** 240 minutes
- **Passing Score:** 70%
- **Language:** English
- **Price:** USD \$400

[Purchase](#)

Have questions? Read our [Certification FAQ](#)

[Verify a Certification](#)

Contact us at [cetification@cloudera.com](mailto:cetification@cloudera.com)

# Certificações

Databricks: <https://academy.databricks.com/category/certifications>

## Assessments

### Databricks Certified Associate Developer for Apache Spark 3.0 - Assessment

The Databricks Certified Associate Developer for Apache Spark 3.0 certification exam assesses an understanding of the basics of the Spark architecture and the ability to apply the Spark DataFrame API to complete individual data manipulation tasks.

\$ 200.00 USD

 [VIEW](#)

### Databricks Certified Associate Developer for Apache Spark 2.4 - Assessment

The Databricks Certified Associate Developer for Apache Spark 2.4 certification exam assesses an understanding of the basics of the Spark architecture and the ability to apply the Spark DataFrame API to complete individual data manipulation tasks.

\$ 200.00 USD

 [VIEW](#)

### Databricks Certified Associate ML Practitioner for Apache Spark 2.4 - Assessment

The Databricks Certified Associate ML Practitioner for Apache Spark 2.4 certification exam assesses the understanding of and ability to apply machine learning techniques using the Spark ML library.

\$ 200.00 USD

 [VIEW](#)

### Databricks Certified Professional Data Scientist - Assessment

The Databricks Certified Professional Data Scientist certification exam assesses the understanding of the basics of machine learning, the steps in the machine learning lifecycle, the understanding of basic machine learning algorithms and techniques, and the understanding of the basics of machine learning model management.

\$ 200.00 USD

 [VIEW](#)

### Databricks Certified Professional Data Engineer - Assessment

Coming soon (early 2021).

\$ 200.00 USD

 [VIEW](#)

# Dúvidas?

Criando pipelines de dados eficientes - Parte 1  
Spark Streaming