DIGITAL INNOVATION ONE

Aceleração Global Dev #4 everis

Criando pipelines de dados eficientes - Parte 2
Spark SQL - PySpark

Marco Antonio Pereira
Expert Data Architect

DIGITAL INNOVATION ONE

Objetivos da Aula

1. Apresentar Spark SQL

2. Usabilidade

3. Casos de Uso

DIGITAL INNOVATION ONE

Requisitos Básicos

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

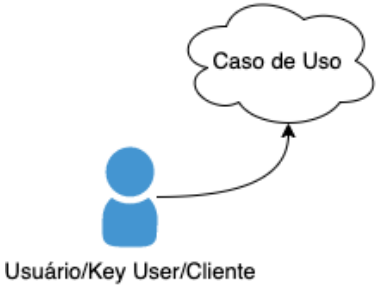
DIGITAL INNOVATION ONE

Parte 1: Falando sobre Dados

Criando pipelines de dados eficientes - Parte 2
Spark SQL

DIGITAL INNOVATION ONE

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.

DIGITAL INNOVATION ONE

Dados

Que tipo de informação podemos extrair?

Lógico:

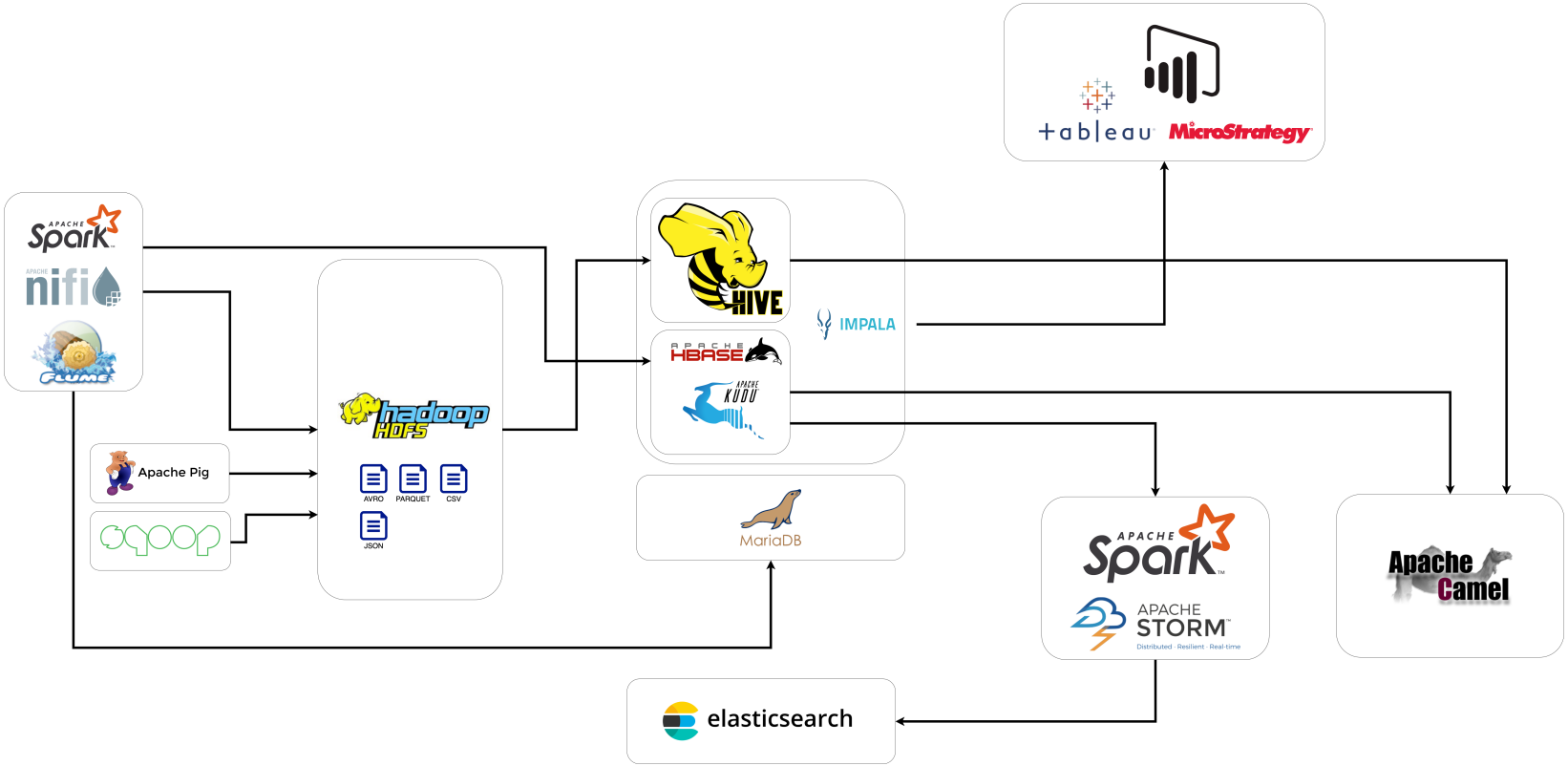
1. Procurar/mapear informações de produtos em alguma base de dados de produto
2. Criar histórico/base vegetativa de até 10 anos
3. Possível ingestão de dados Batch (dia -1 = dados de ontém até -10 anos)

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?

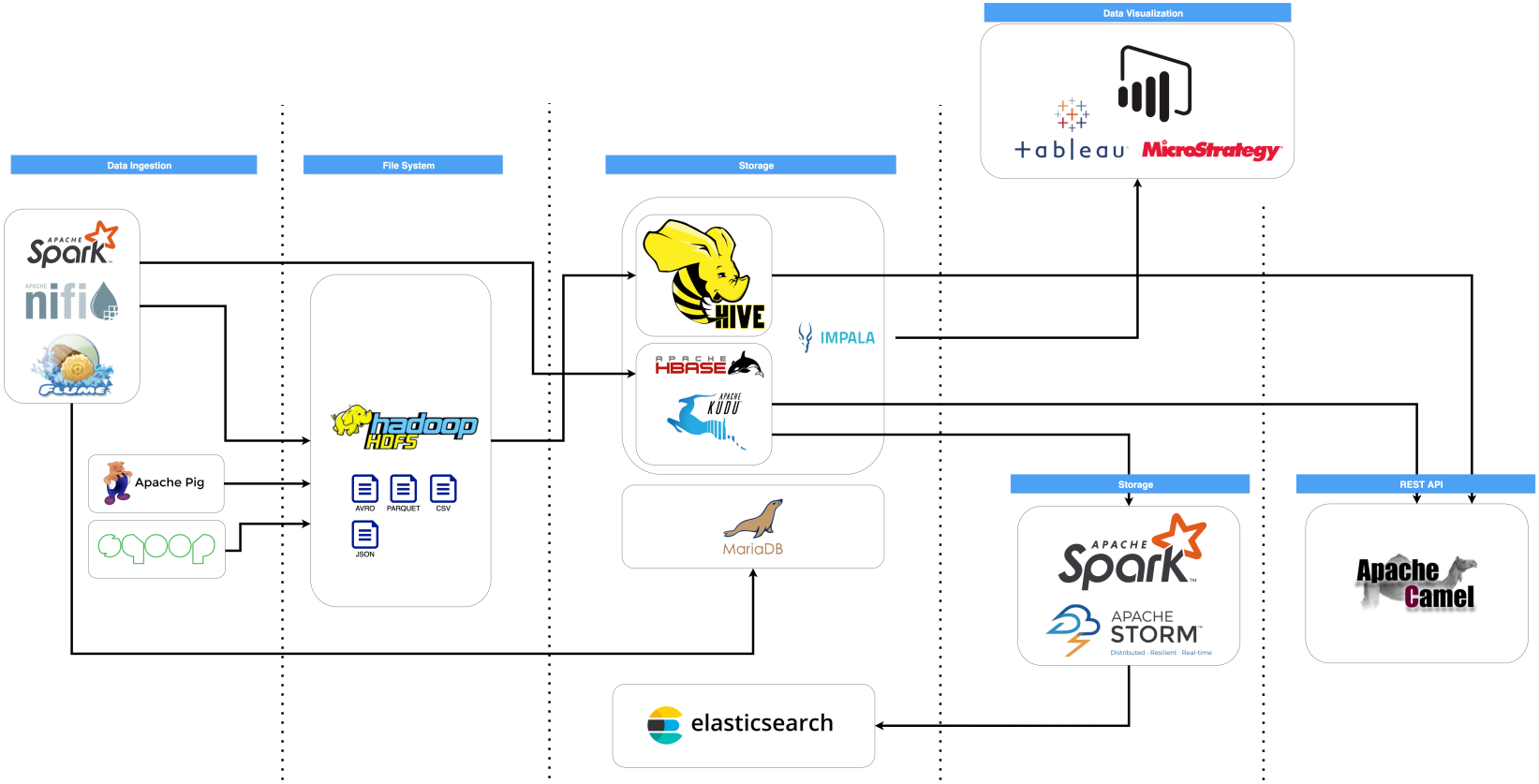
DIGITAL INNOVATION ONE

Arquitetura



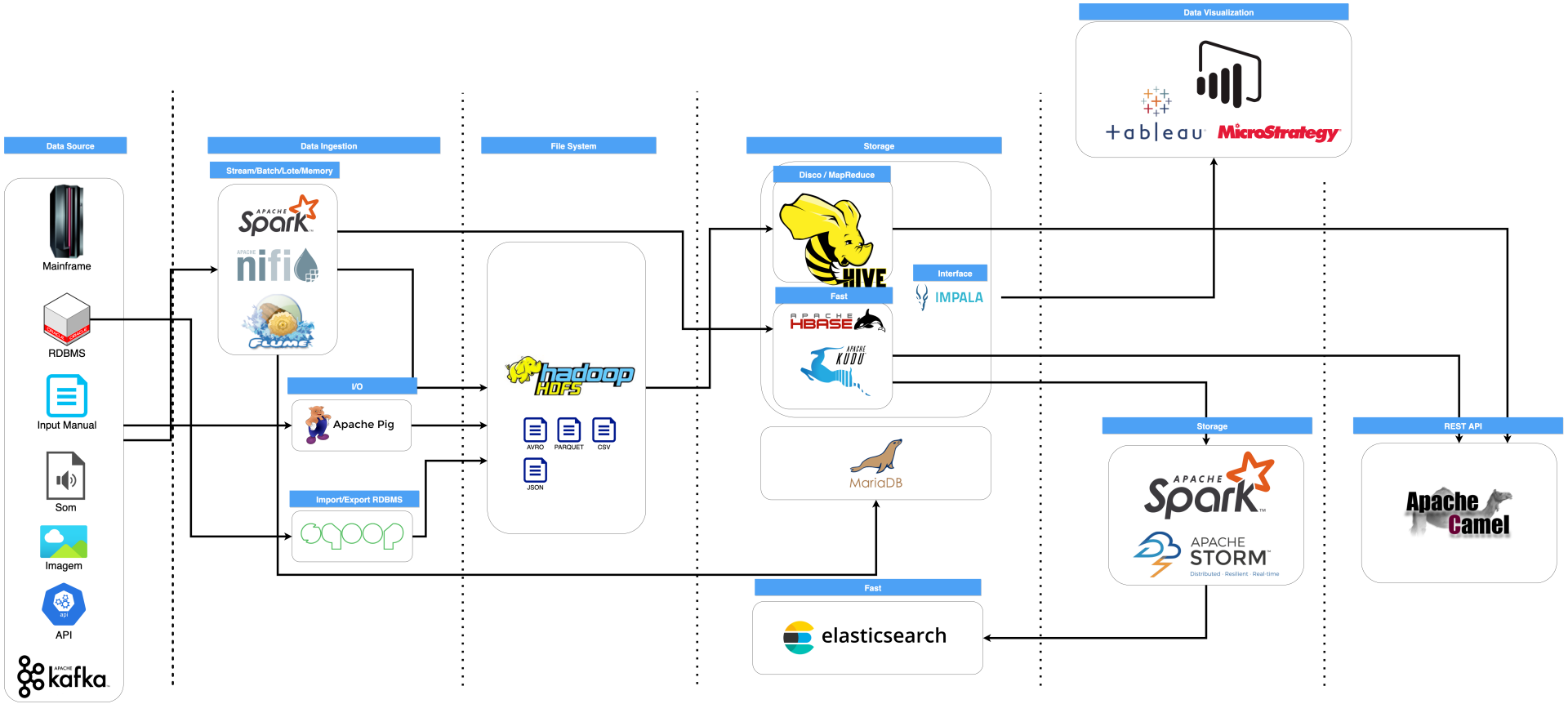
DIGITAL INNOVATION ONE

Arquitetura



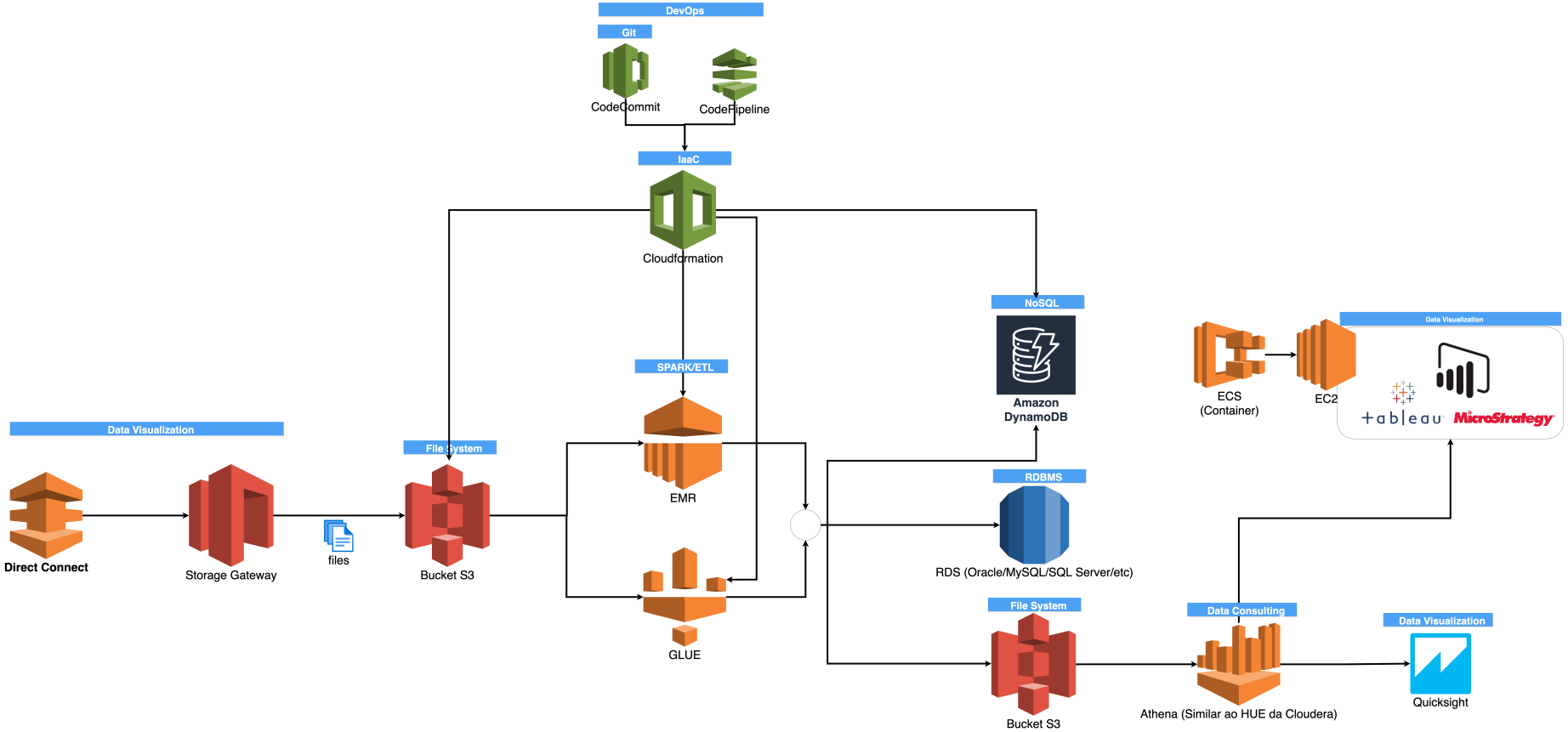
DIGITAL INNOVATION ONE

On Premises



DIGITAL INNOVATION ONE

Cloud (AWS)



DIGITAL INNOVATION ONE

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

DIGITAL INNOVATION ONE

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

DIGITAL INNOVATION ONE

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:

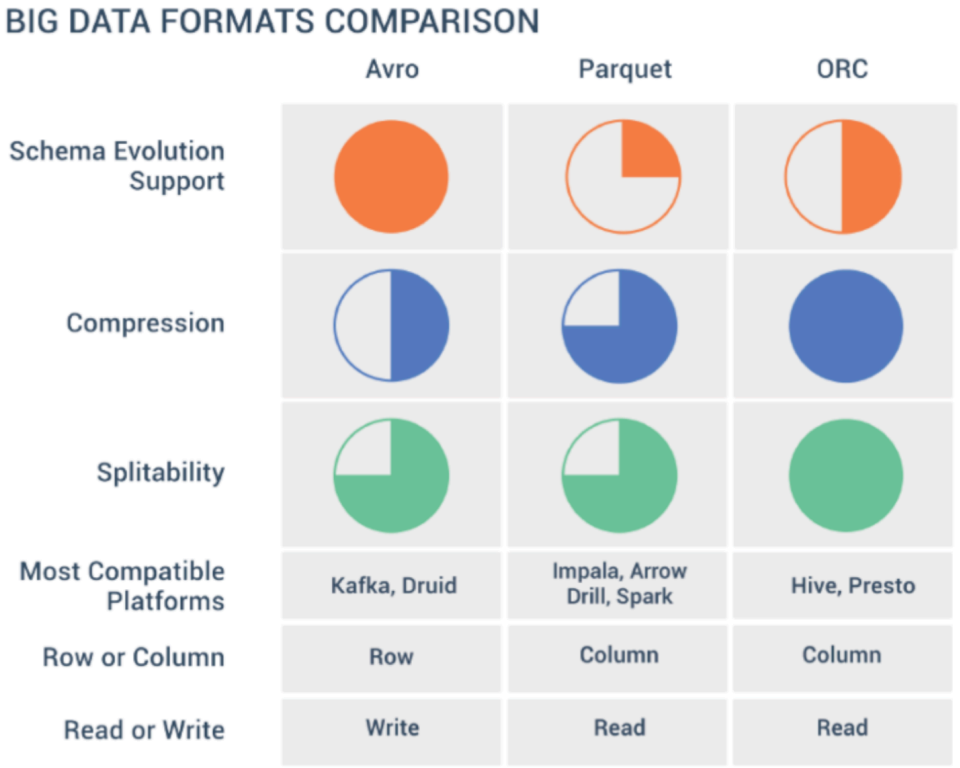
Optimized Row Columnar (ORC)

Avro

Parquet

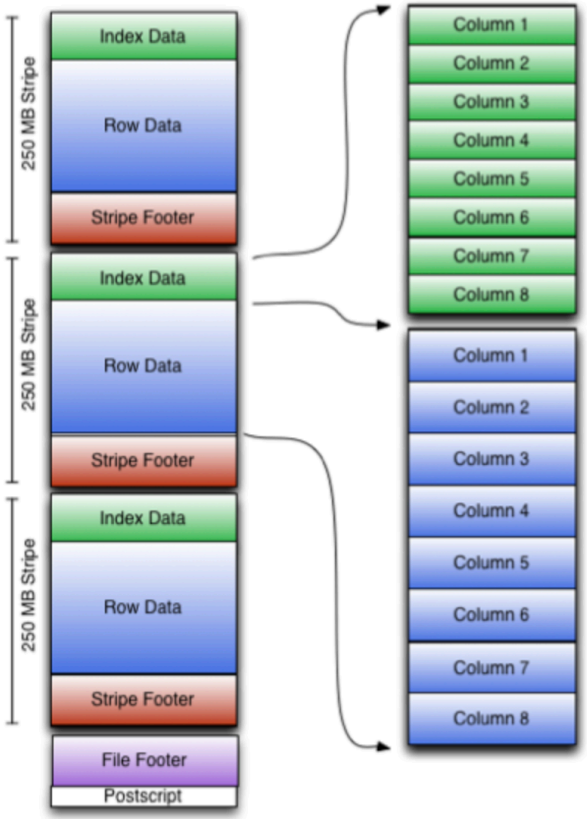
DIGITAL INNOVATION ONE

Tipos de armaz.



DIGITAL INNOVATION ONE

Parquet



DIGITAL INNOVATION ONE

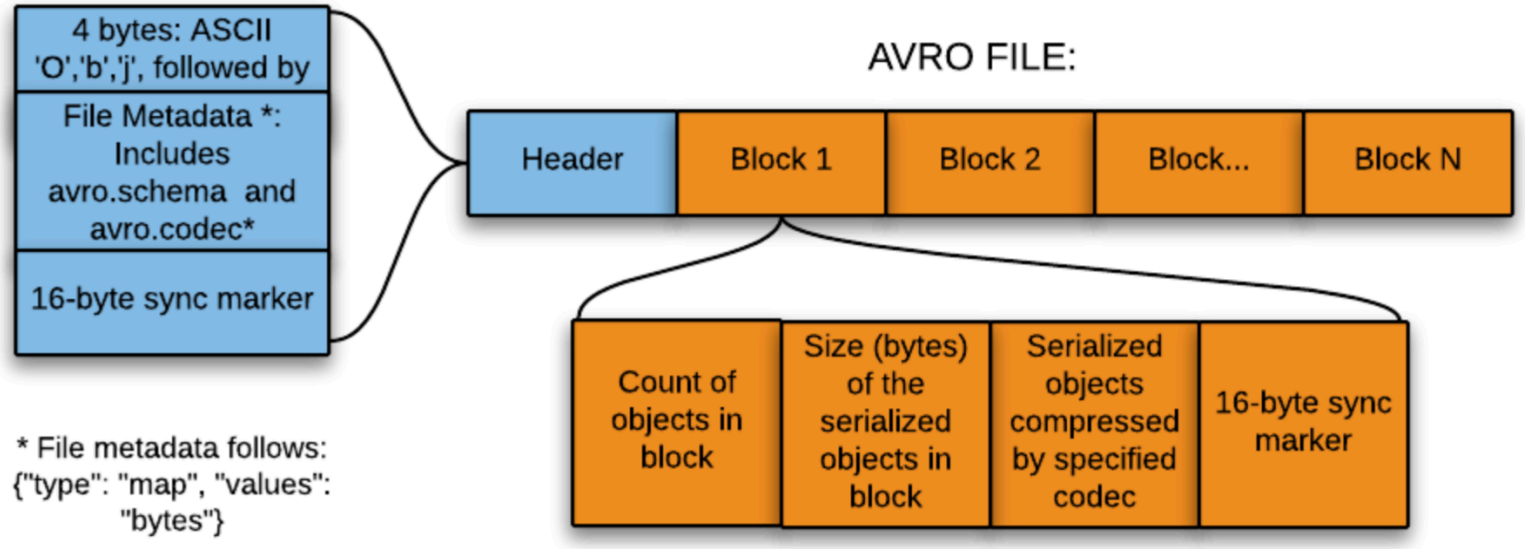
Parquet

- 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/

DIGITAL INNOVATION ONE

Avro



DIGITAL INNOVATION ONE

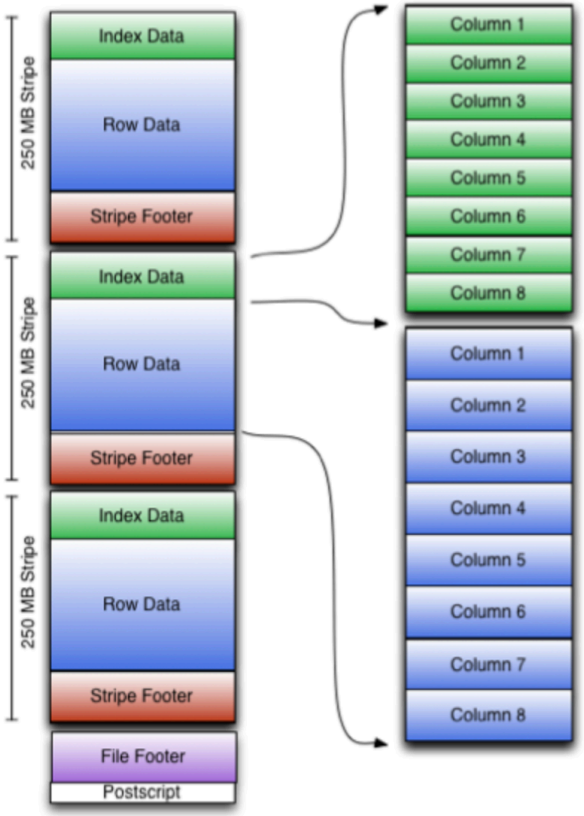
Avro

- 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/

DIGITAL INNOVATION ONE

ORC



DIGITAL INNOVATION ONE

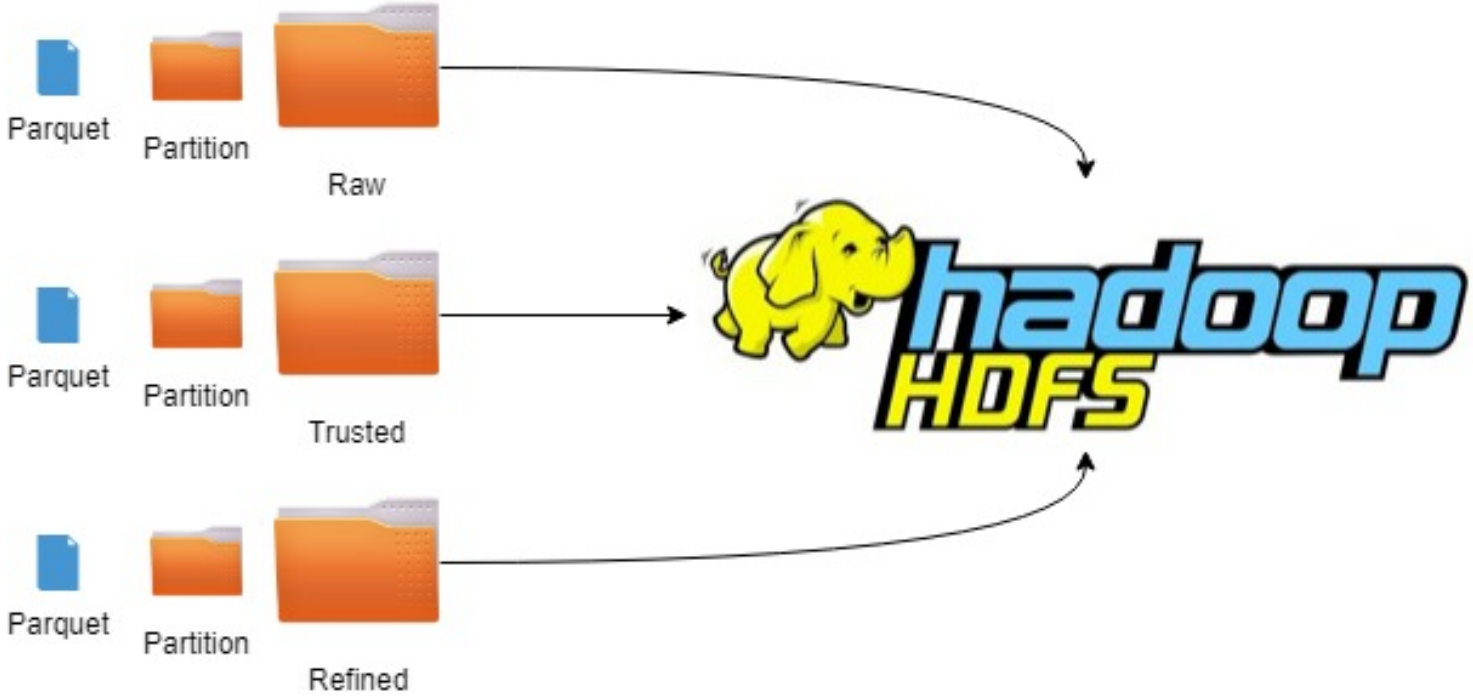
ORC

- 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

DIGITAL INNOVATION ONE

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.

DIGITAL INNOVATION ONE

HDFS

Referência dos Schemas, incluindo local/nodes com os arquivos

File 1

File 1 File 1 File 1

$ hdfs dfs -du -h /<path>

Size * Replication Factory

Arquivos físicos

Apache Derby

MariaDB

DIGITAL INNOVATION ONE

Parte 2: Spark SQL

Criando pipelines de dados eficientes - Parte 1

Spark Streaming

DIGITAL INNOVATION ONE

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()
```

DIGITAL INNOVATION ONE

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_temporiaria = "tempTableDataFrame1"
df1.createOrReplaceTempView(nome_tabela_temporiaria)
```

DIGITAL INNOVATION ONE

DataFrame

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

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

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

DIGITAL INNOVATION ONE

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_hur
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	7084	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()
```

DIGITAL INNOVATION ONE

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
```

DIGITAL INNOVATION ONE

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)
```

DIGITAL INNOVATION ONE

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)
```

DIGITAL INNOVATION ONE

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)
```

DIGITAL INNOVATION ONE

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)
```

DIGITAL INNOVATION ONE

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
```

DIGITAL INNOVATION ONE

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
```

DIGITAL INNOVATION ONE

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)
```

DIGITAL
INNOVATION
ONE

DataFrame

JOINS

DIGITAL INNOVATION ONE

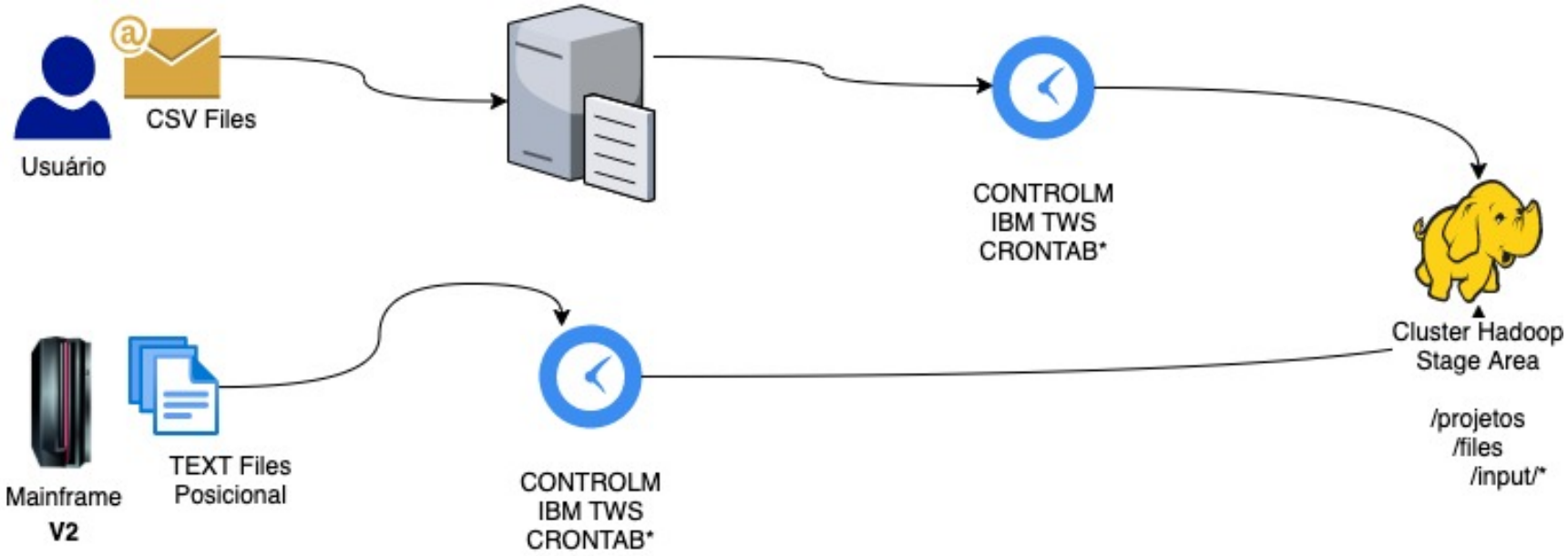
Parte 3: Casos de Uso

Criando pipelines de dados eficientes - Parte 1

Spark Streaming

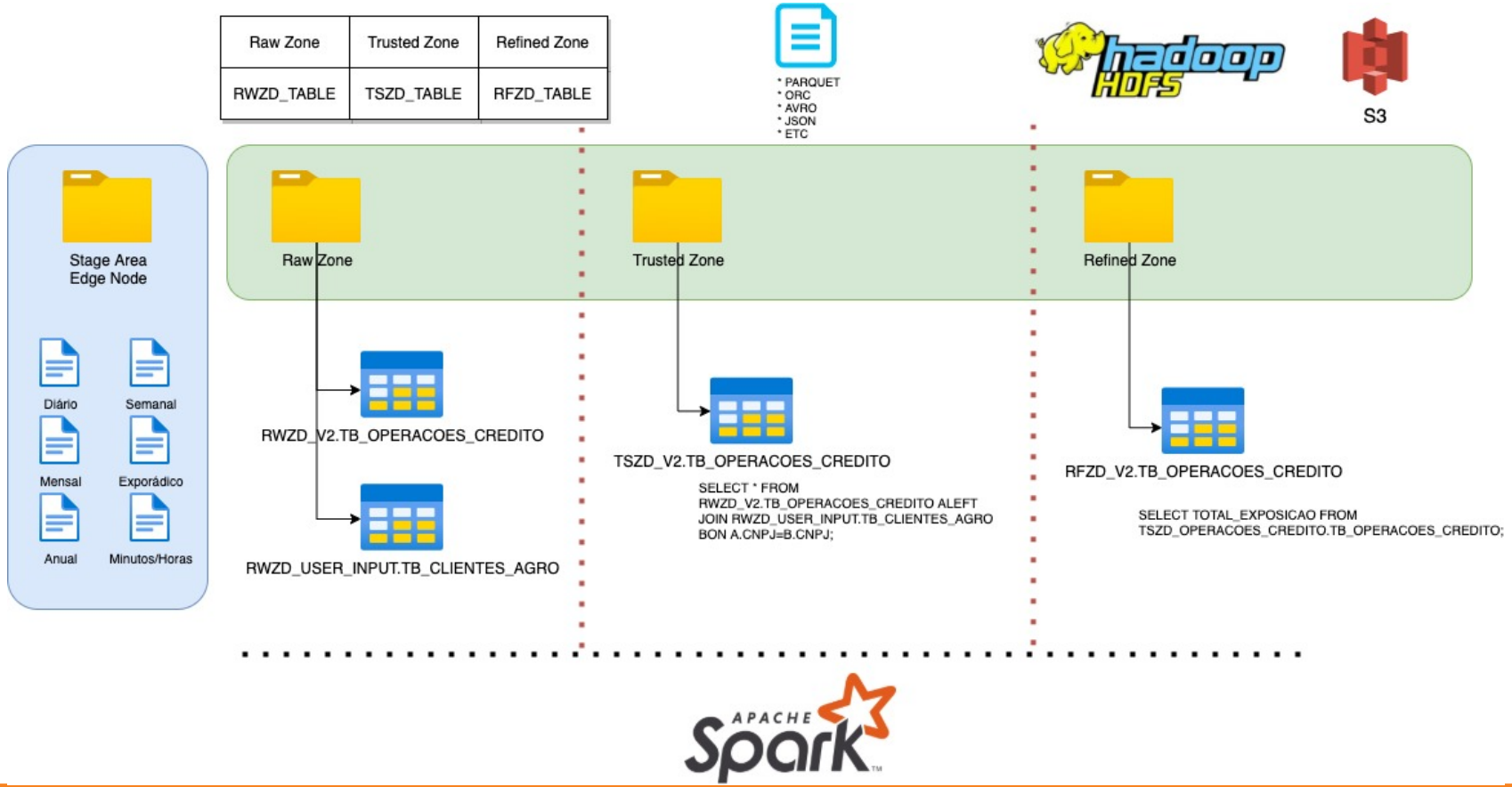
DIGITAL INNOVATION ONE

Caso de Uso 1



DIGITAL INNOVATION ONE

Caso de Uso 2



DIGITAL INNOVATION ONE

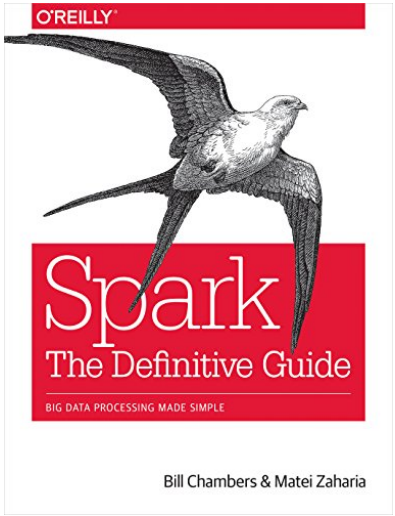
Parte 4: Encerramento

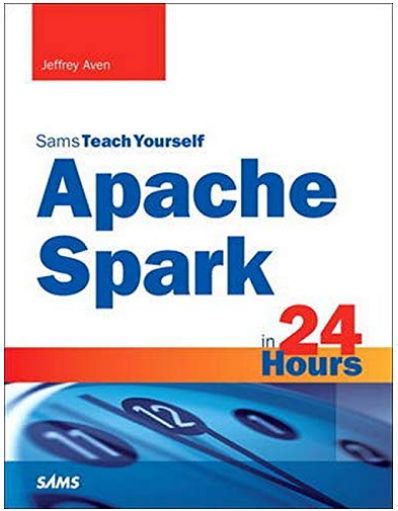
Criando pipelines de dados eficientes - Parte 1

Spark Streaming

DIGITAL INNOVATION ONE

Referências

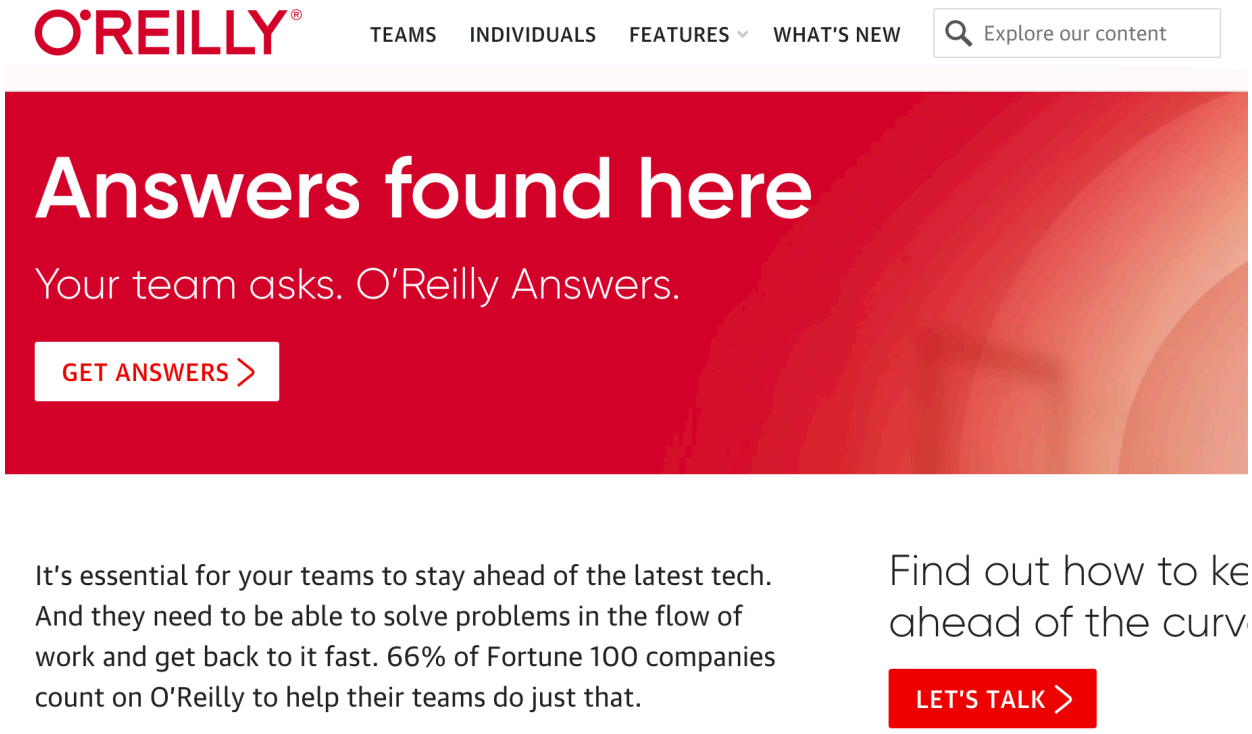




Documentação: https://spark.apache.org/docs/2.4.0/api/python/pyspark.sql.html

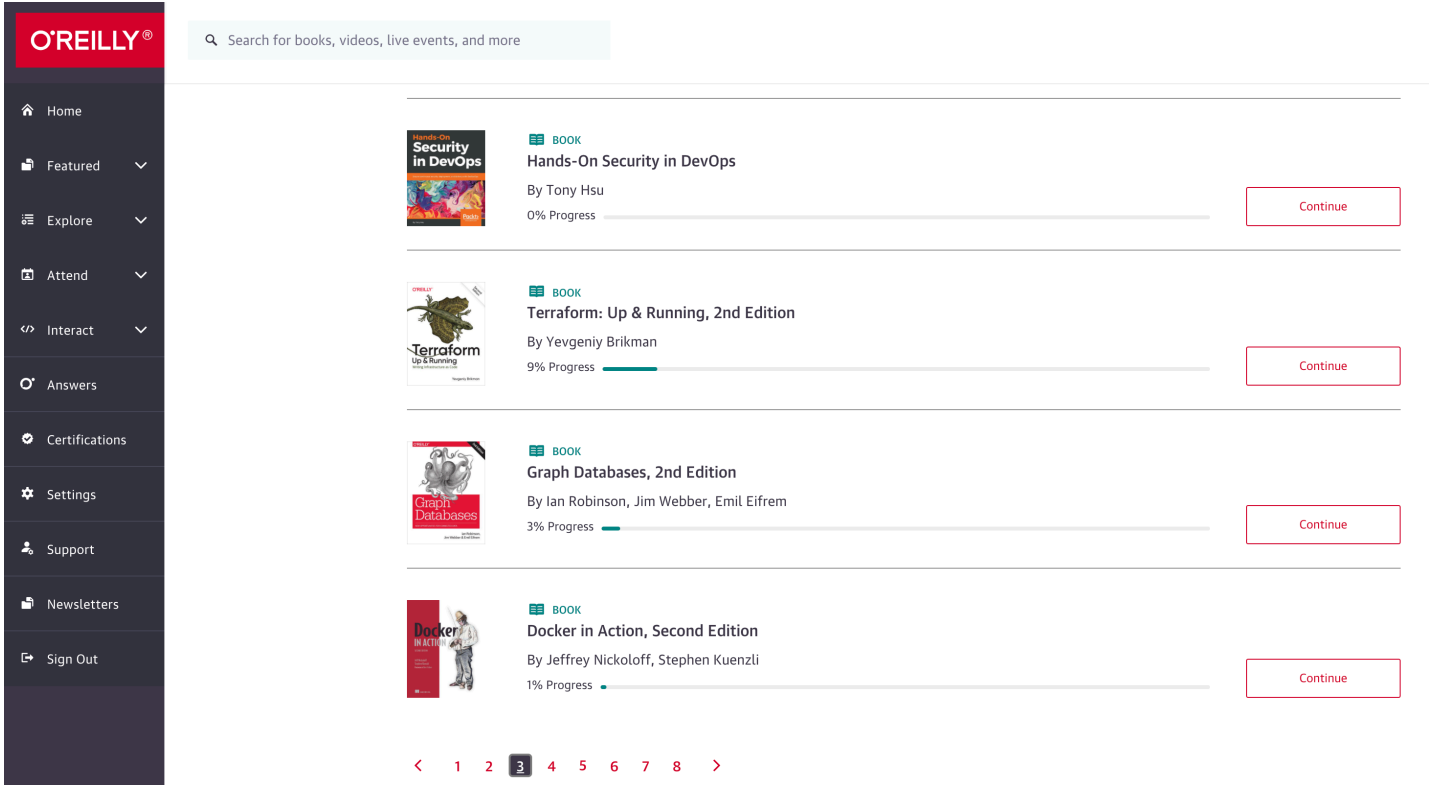
Referências

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



Referências

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



DIGITAL INNOVATION ONE

Referências

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

Preparation

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

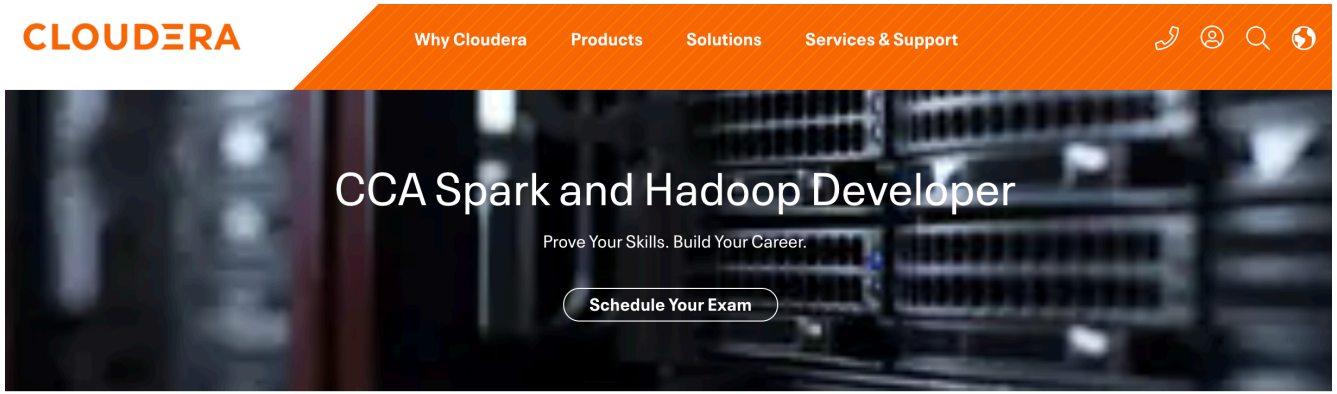
- DB 105 - Apache Spark Programming
- Quick Reference: Spark Architecture
- Future self-paced course on the Spark DataFrames API

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

DIGITAL INNOVATION ONE

Certificações

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



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**

DIGITAL INNOVATION ONE

Certificações

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

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 **certification@cloudera.com**

DIGITAL INNOVATION ONE

Certificações

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

Assessments

Databricks Certified Associate Developer for Apache Spark 3.0 – *Assessment*

$ 200.00 USD

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.

☆ VIEW

Databricks Certified Associate Developer for Apache Spark 2.4 – *Assessment*

$ 200.00 USD

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.

☆ VIEW

Databricks Certified Associate ML Practitioner for Apache Spark 2.4 – *Assessment*

$ 200.00 USD

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

☆ VIEW

Databricks Certified Professional Data Scientist – *Assessment*

$ 200.00 USD

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.

☆ VIEW

Databricks Certified Professional Data Engineer – *Assessment*

$ 200.00 USD

Coming soon (early 2021).

☆ VIEW

DIGITAL
INNOVATION
ONE

Dúvidas?

Criando pipelines de dados eficientes - Parte 1
Spark Streaming