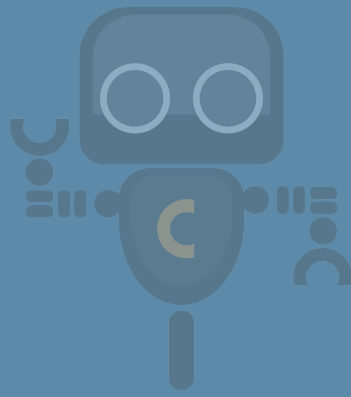
Fundamentos e Arquitetura de Bigdata, Banco de dados não relacionais

Aula 3



Francisco Nauber Bernardo Gois

Universidade Estadual do Ceará

2 de março de 2

Instrutor

F. Nauber Bernardo Gois

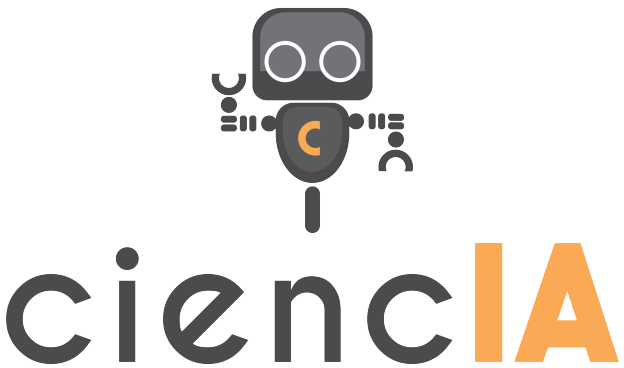
Dsc. Informática Aplicada
Lider de Aprendizado de Máquina na Secretaria de Saúde do Ceará
Engenheiro de Aprendizado de Máquina
Professor da UFC (2018-2019)
Analista de Desenvolvimento Serpro (2004-2018)

Instrutor

https://www.linkedin.com/in/n

https://www.linkedin.com
/in/naubergois/
Envie recomendações, depoimentos e competÊncias

Instrutor



https://www.youtube.com/can

https://www.youtube.com/
canaldaciencia
Instagram: @canaldaciencia

Índice

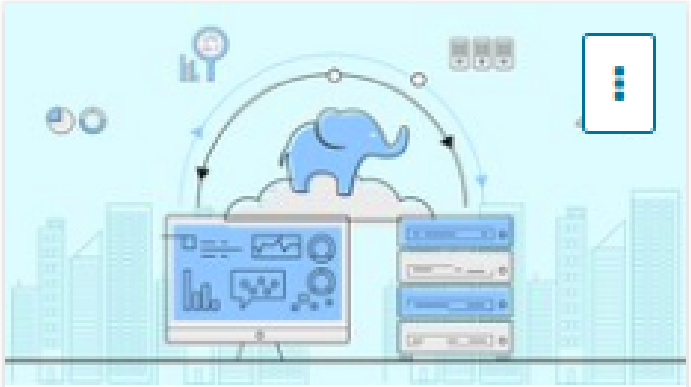
Índice

The Ultimate Hands-On Hadoop - Tame your Big Data!

Sundog Education by Frank Kane, Founder, Sundog...

6% Complete

Deixe uma classificação



Learn Big Data: The Hadoop Ecosystem Masterclass

Edward Viaene, DevOps, Cloud, Big Data Specialist

8% Complete

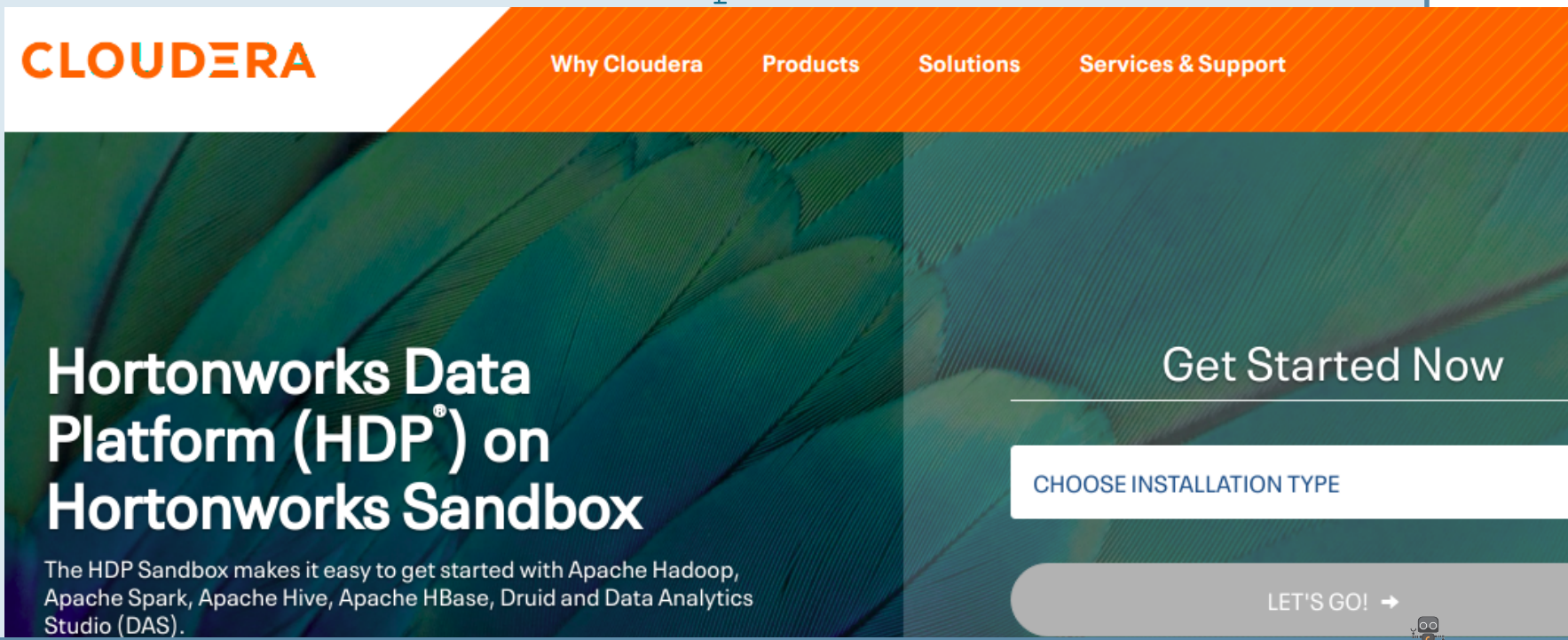
Sua classificação

Cursos Recomendados

Material do Curso

HDP 2.5 sandbox image

Baixe a imagem no endereço `https://www.cloudera.com/downloads/hortonworks-sandbox/hdp.html`

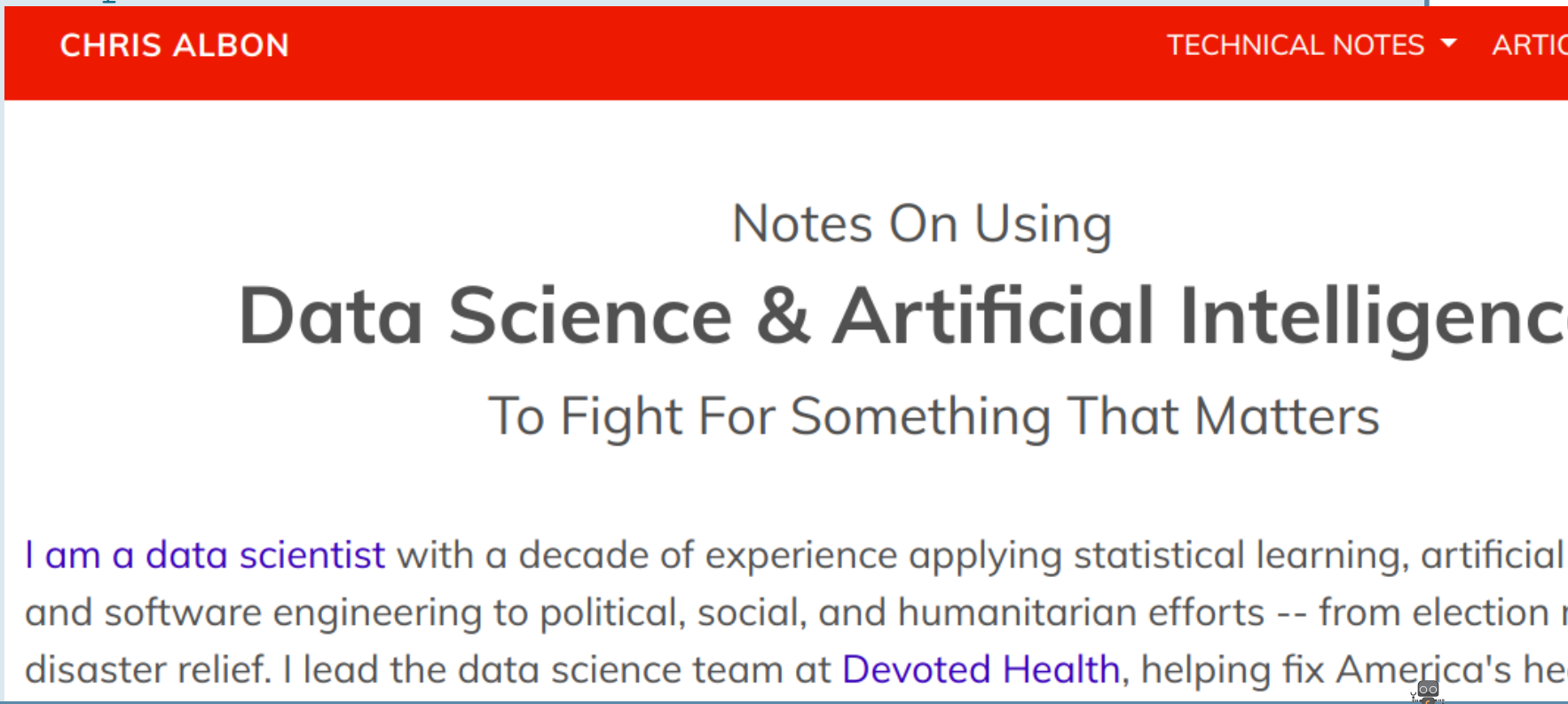


Material do Curso

Chris Albon

Curso de Aprendizado de Máquina

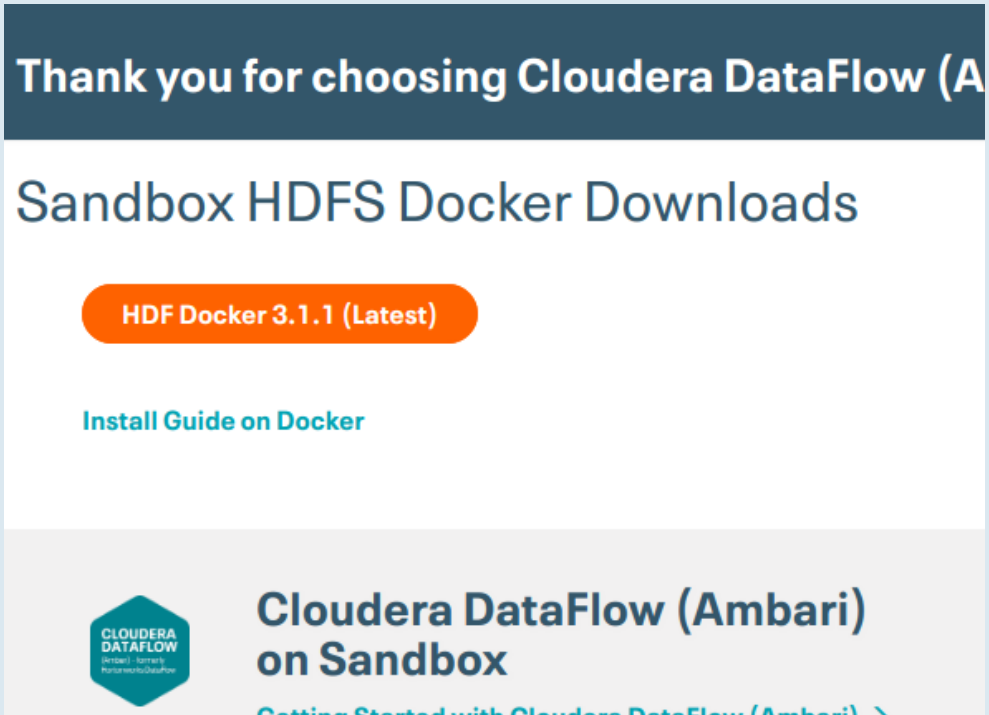
https://chrisalbon.com/

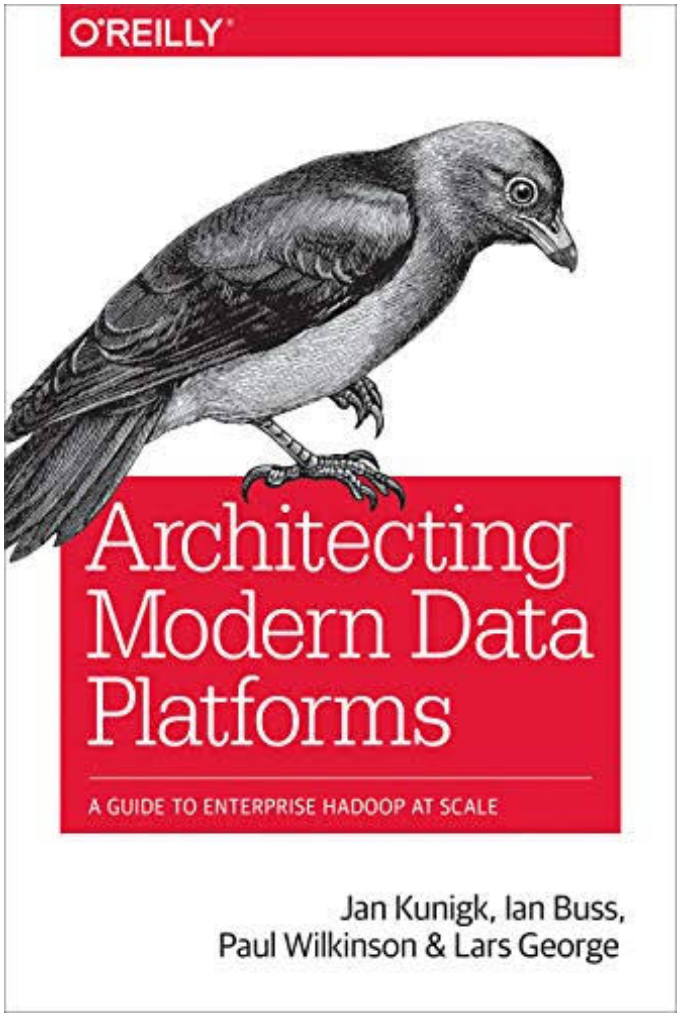


Material do Curso

HDP 2.5 sandbox image

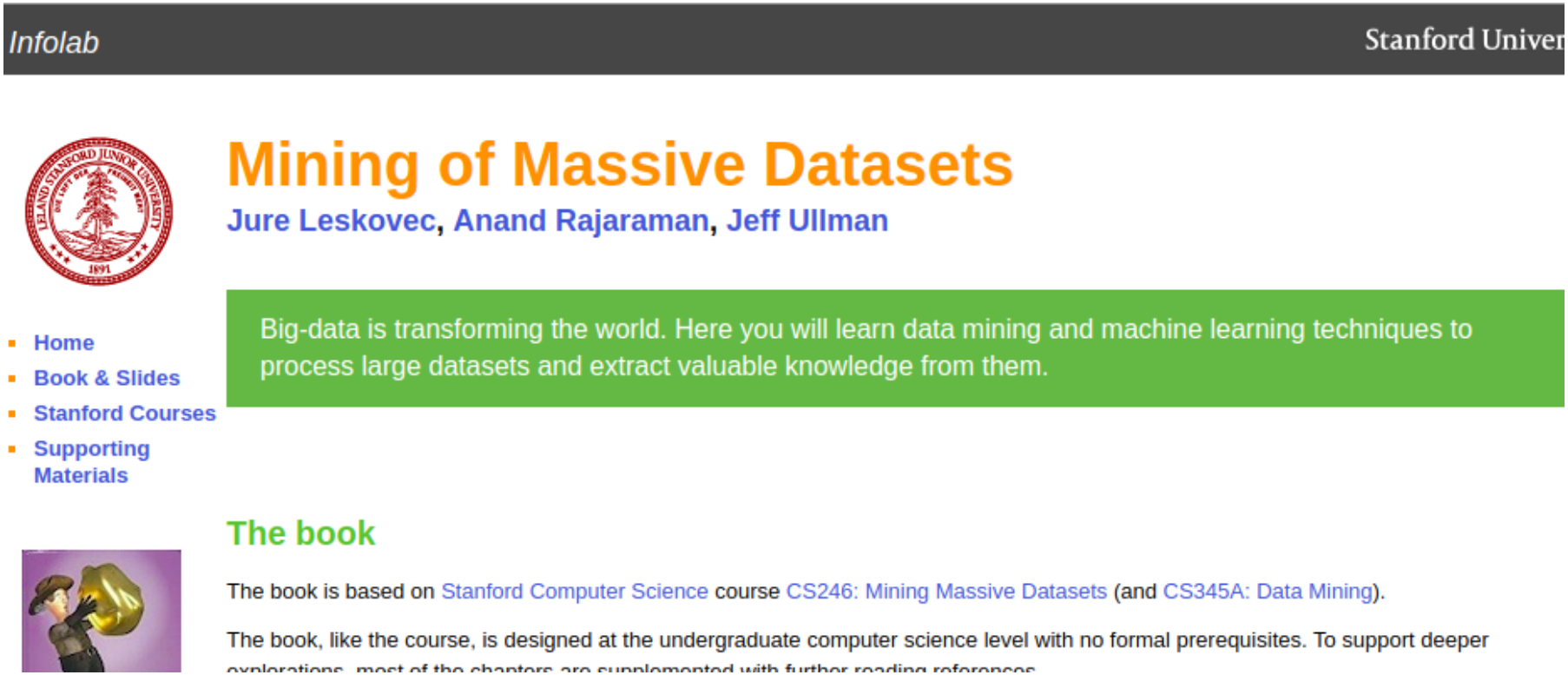
Baixe a imagem no endereço `https://www.cloudera.com/downloads/hortonworks-sandbox/hdf.html?utm_source=`





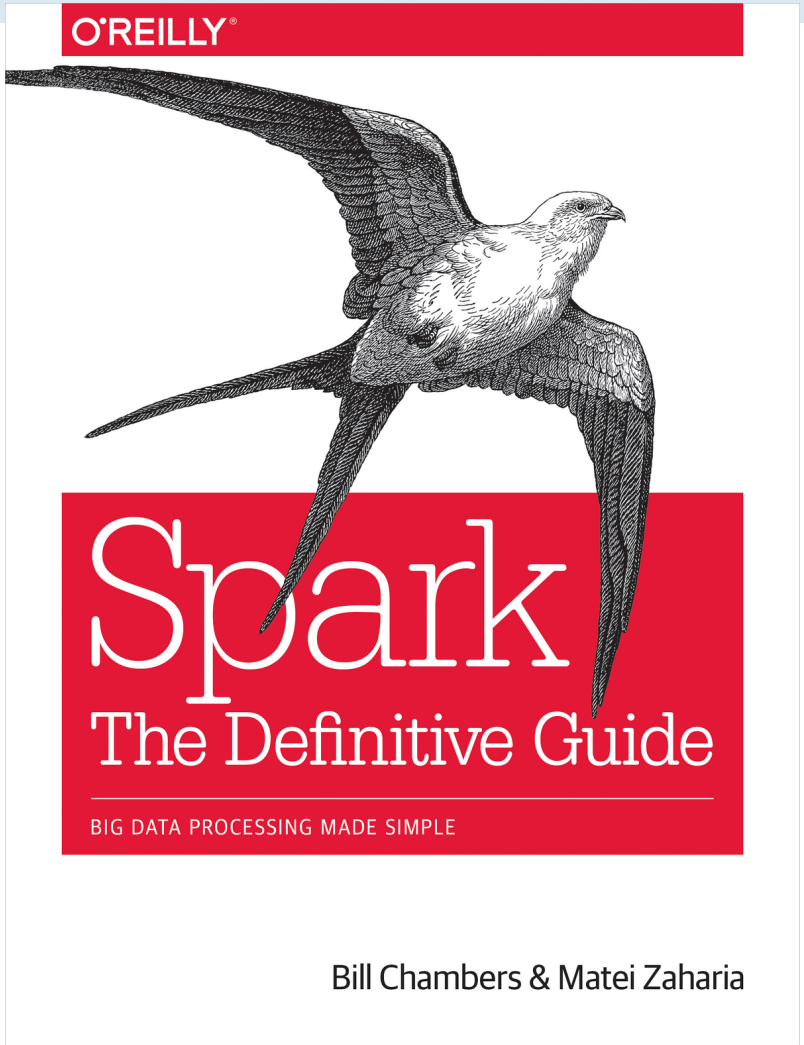
Livro Recomendado

Livro



http://mmds.org/

Livro


O'REILLY
Spark
The Definitive Guide
BIG DATA PROCESSING MADE SIMPLE
Bill Chambers & Matei Zaharia


APACHE
Spark
TM

Spark

Conceito

O Apache Spark é um mecanismo de computação unificado e um conjunto de bibliotecas para processamento paralelo de dados em clusters de computadores [CZ18]

Instalando spark

Descompactando arquivo do Spark

```
cd ~/Downloads
tar -xf spark-2.2.0-bin-hadoop2.7.tgz
cd spark-2.2.0-bin-hadoop2.7.tgz
```

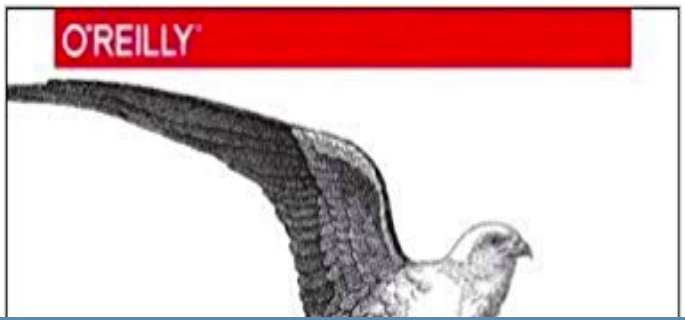
data	add missing images	2 years ago
project-templates	Added dependency for Spark ML	2 years ago
.gitignore	first pass	3 years ago
README.md	Update README.md	2 years ago
license.md	first pass	3 years ago

README.md

Spark: The Definitive Guide

This is the central repository for all materials related to Spark: The Definitive Guide by Bill Chambers and Matei Zaharia.

This repository is currently a work in progress and new material will be added over time.



Lightning-fast unified analytics engine

Download | Libraries ▾ | Documentation ▾ | Examples | Community ▾ | Developers ▾ | Apache Software Found

Download Apache Spark™

1. Choose a Spark release: 3.0.0-preview2 (Dec 23 2019) ▾
2. Choose a package type: Pre-built for Apache Hadoop 2.7 ▾
3. Download Spark: spark-3.0.0-preview2-bin-hadoop2.7.tgz
4. Verify this release using the 3.0.0-preview2 signatures, checksums and project release KEYS.

Note that, Spark is pre-built with Scala 2.11 except version 2.4.2, which is pre-built with Scala 2.12.

Latest Preview Release

Preview releases, as the name suggests, are releases for previewing upcoming features. Unlike nightly packages, preview releases

Latest News

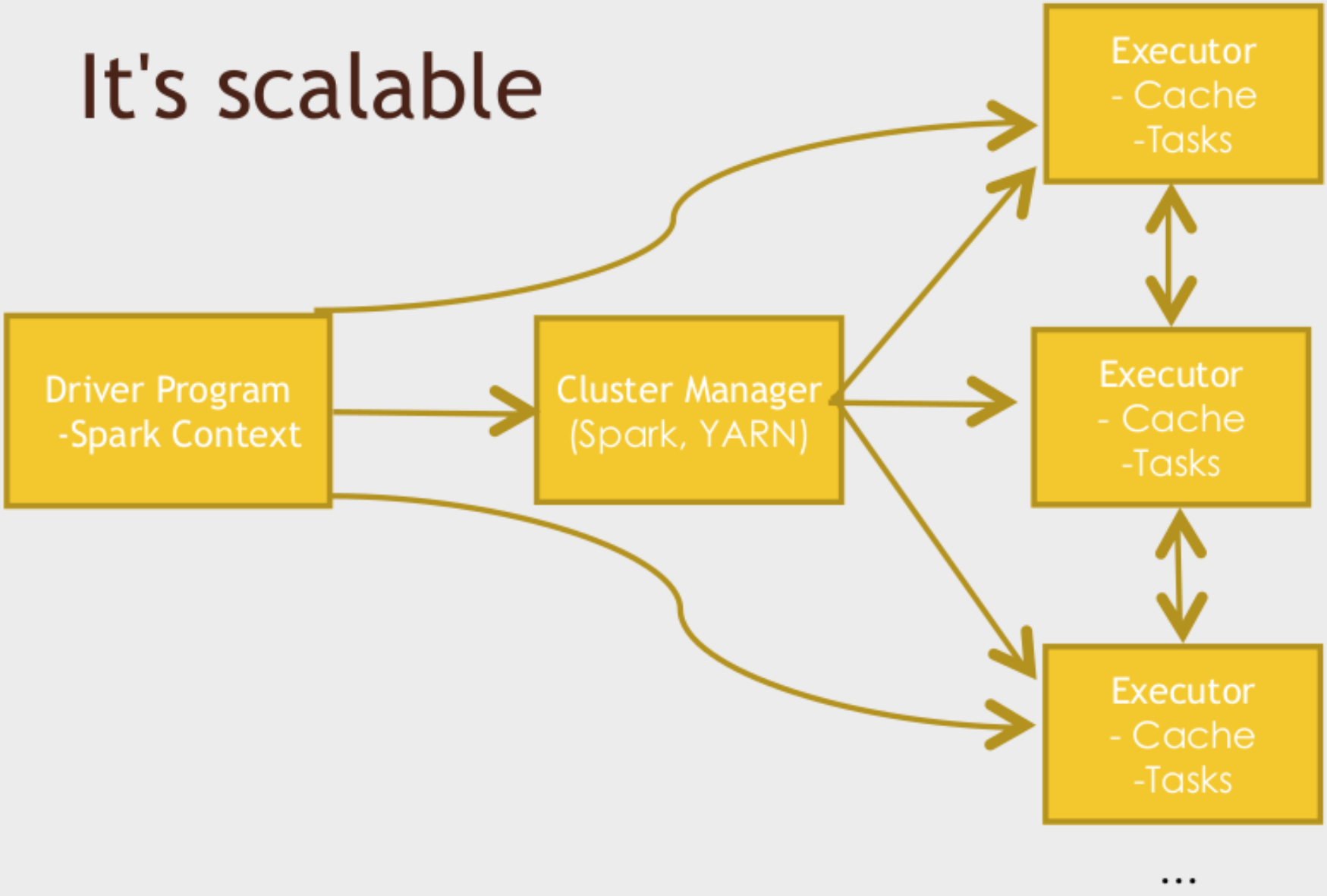
Spark 2.4.5 released (Feb 08, 2

Preview release of Spark 3.0 (
2019)

Preview release of Spark 3.0 (
2019)

Spark 2.3.4 released (Sep 09, 2

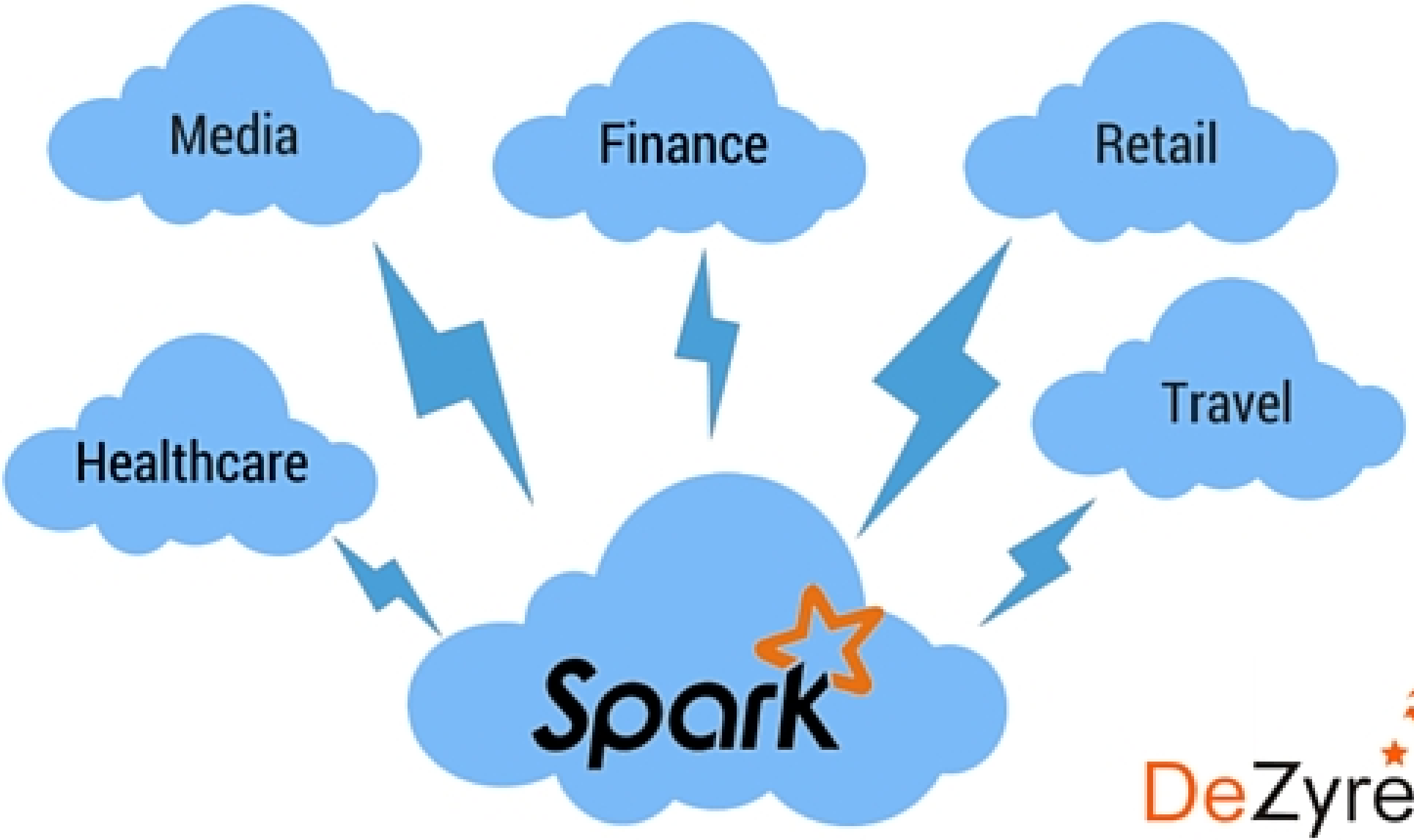
It's scalable




MySQL
elastic
PostgreSQL
kafka
APACHE
Spark
redis
and more...

It's fast

- "Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk."
- DAG Engine (directed acyclic graph) optimizes workflows

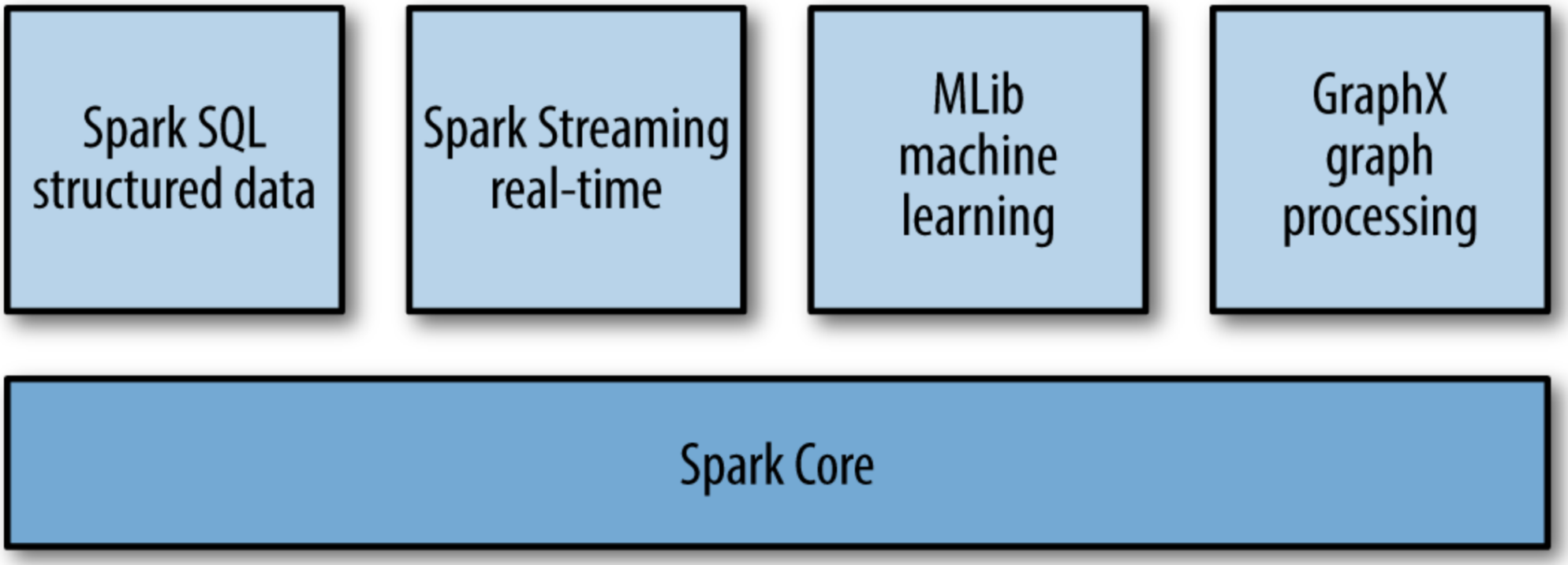

Media
Finance
Retail
Healthcare
Travel
Spark
DeZyre

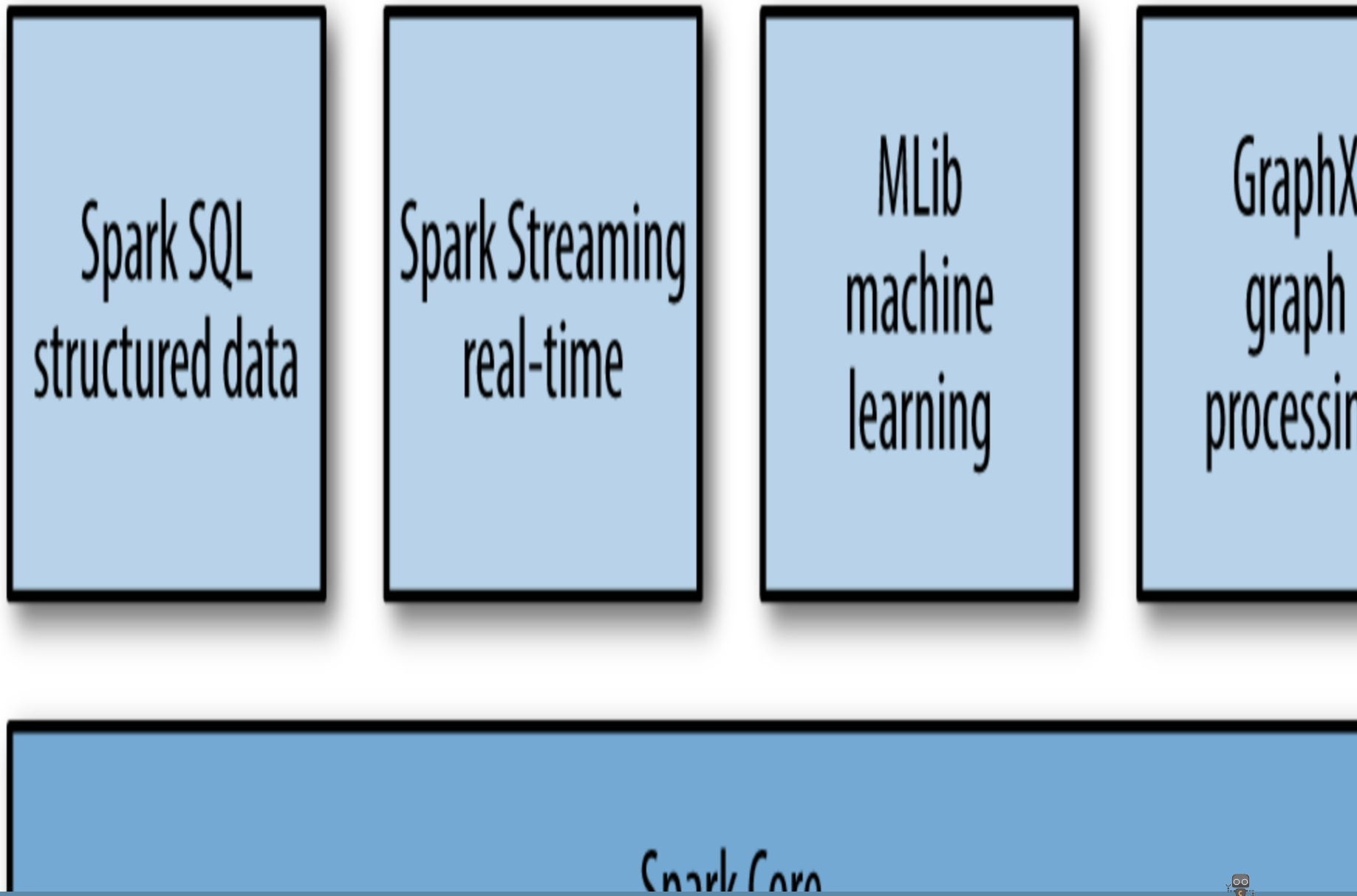
It's hot

- Amazon
- Ebay: log analysis and aggregation
- NASA JPL: Deep Space Network
- Groupon
- TripAdviser
- Yahoo
- Many others:
  https://cwiki.apache.org/confluence/display/SPARK/Powered+By+Spark

It's not that hard

- Code in Python, Java, or Scala
- Built around one main concept: the Resilient Distributed Dataset (RDD)


Spark SQL
structured data
Spark Streaming
real-time
MLib
machine
learning
GraphX
graph
processing
Spark Core


Spark SQL
structured data
Spark Streaming
real-time
MLib
machine
learning
GraphX
graph
processin

FEAR NOT

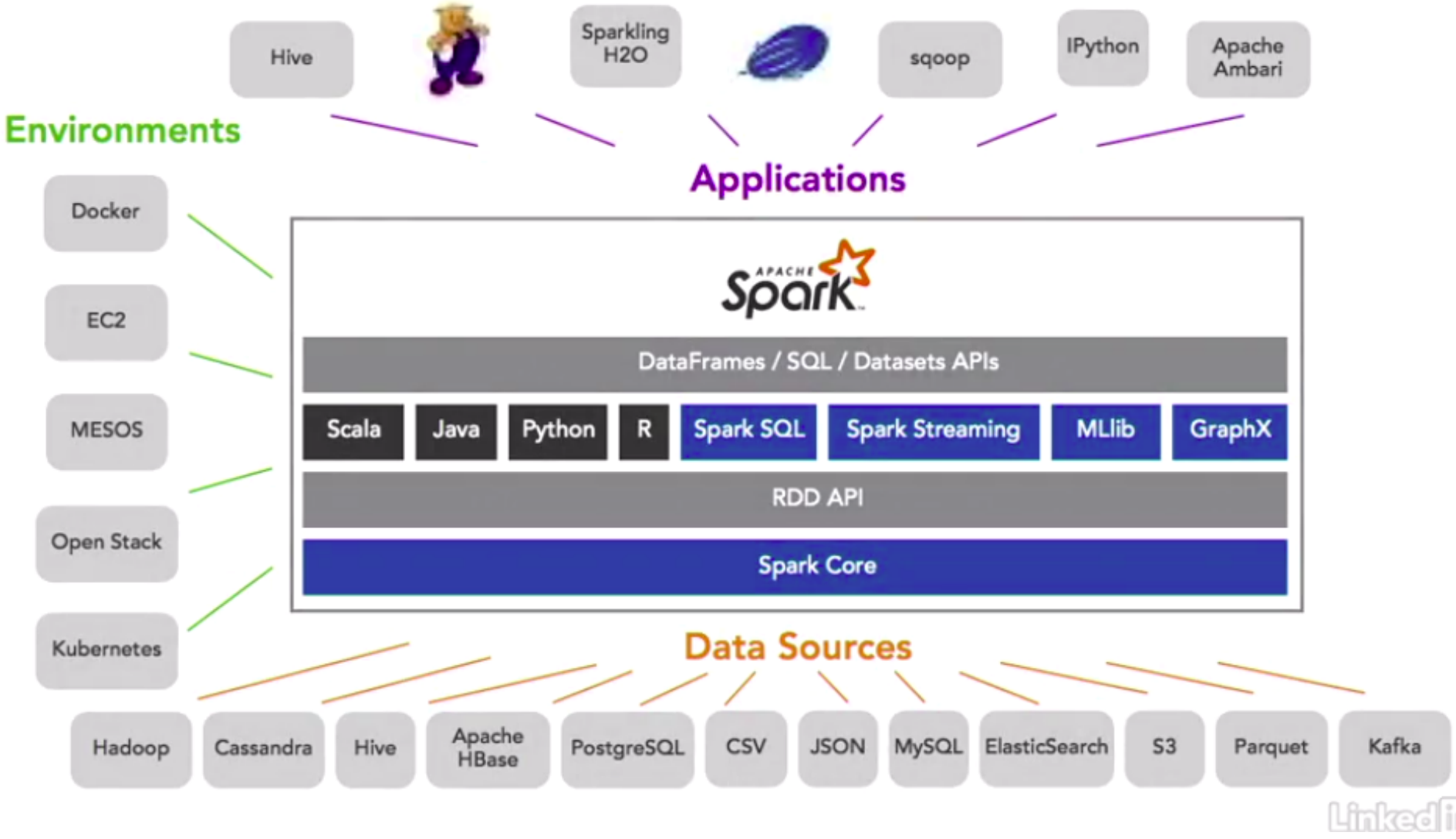
- Scala code in Spark looks a LOT like Python code.

Python code to square numbers in a data set:

```
nums = sc.parallelize([1, 2, 3, 4])
squared = nums.map(lambda x: x * x).collect()
```

Scala code to square numbers in a data set:

```
val nums = sc.parallelize(List(1, 2, 3, 4))
val squared = nums.map(x => x * x).collect()
```


Hive
Sparkling H2O
sqoop
IPython
Apache Ambari
Environments
Applications
Docker
EC2
MESOS
Open Stack
Kubernetes
APACHE Spark
DataFrames / SQL / Datasets APIs
Scala
Java
Python
R
Spark SQL
Spark Streaming
MLlib
GraphX
RDD API
Spark Core
Data Sources
Hadoop
Cassandra
Hive
Apache HBase
PostgreSQL
CSV
JSON
MySQL
ElasticSearch
S3
Parquet
Kafka

Spark Programming Languages

Language	Details
Python	Simplest
Scala	Functional and scalable
Java	Enterprise
R	For machine learning

SparkSession Objects

SQLContext (sc)

- Available in Spark 1.x

spark

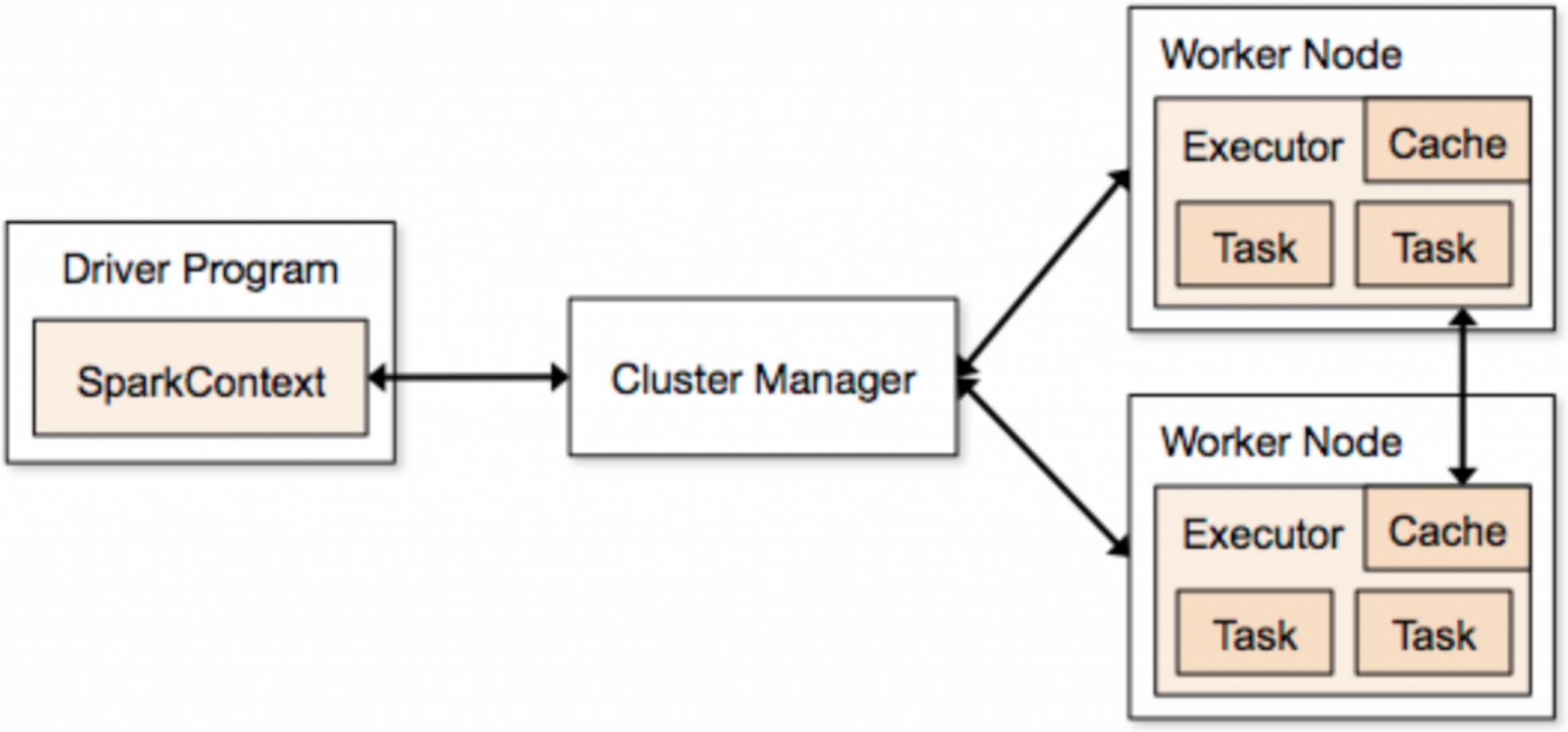
- Available in Spark 2.0

RDD

- Resilient
- Distributed
- Dataset

Creating RDD's

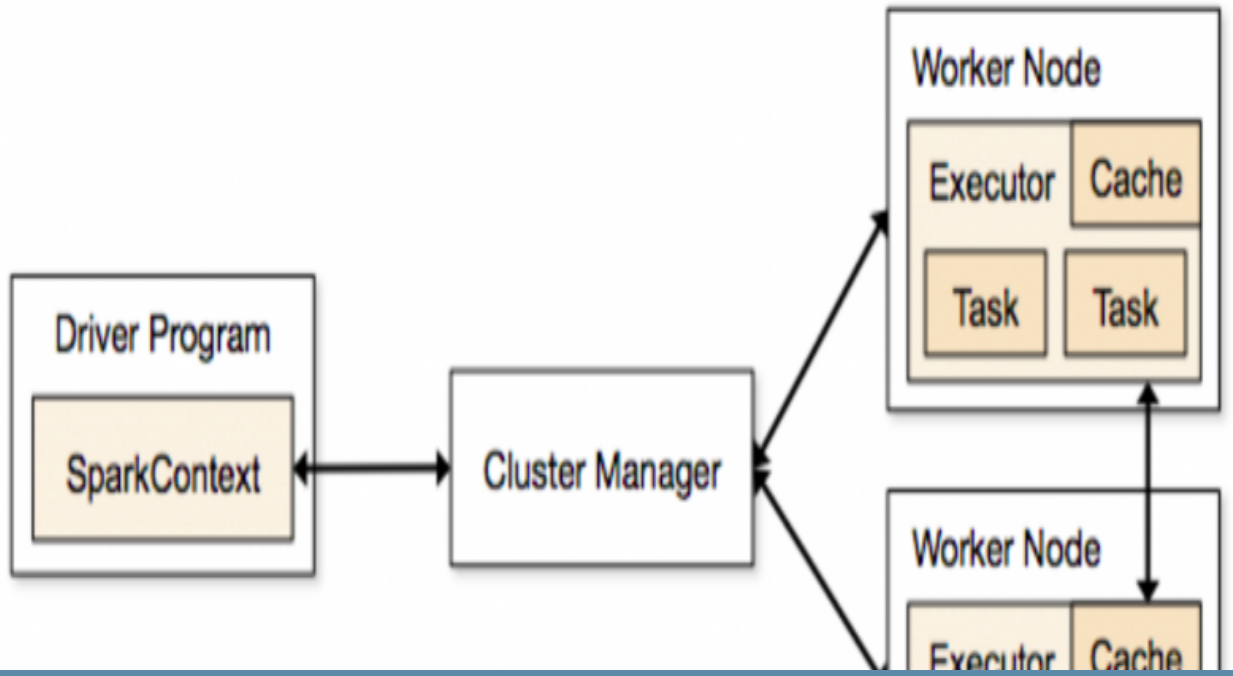
- nums = parallelize([1, 2, 3, 4])
- sc.textFile("file:///c:/users/frank/gobs-o-text.txt")
  - *or s3n:// , hdfs://*
- hiveCtx = HiveContext(sc) rows = hiveCtx.sql("SELECT name, age FROM users")
- Can also create from:
  - *JDBC*
  - *Cassandra*
  - *HBase*
  - *Elastisearch*
  - *JSON, CSV, sequence files, object files, various compressed formats*


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task

Spark Runtime Components

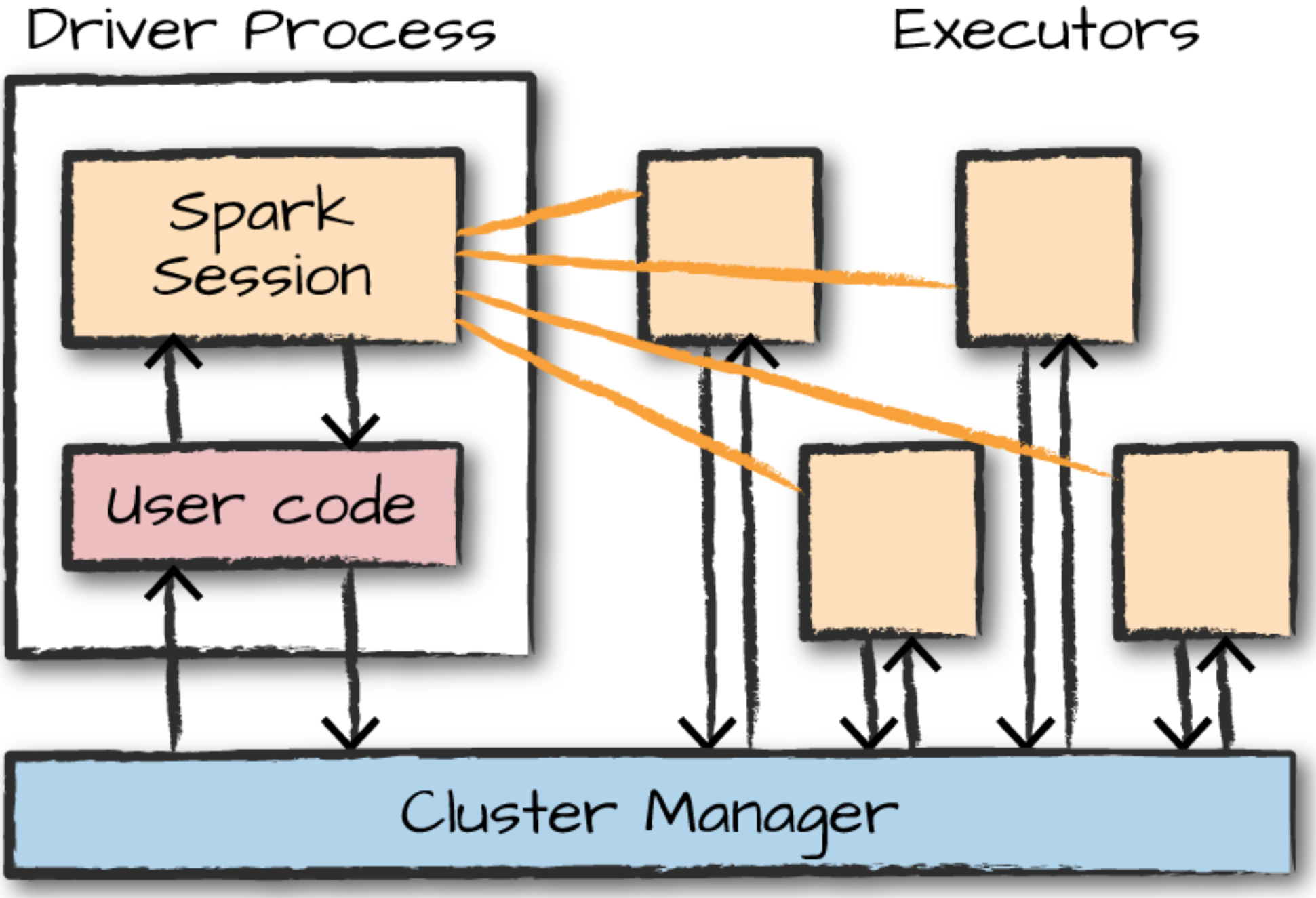
NOVEMBER 4, 2018 • VIPIN CHADHA • APACHE SPARK, SPARK CONCEPTS

Spark Cluster has various components and sub-systems which are running in the background when a job is being executed. All of these components below. Don't worry if you have trouble imagining how these are created or implemented in code. That will become clear in the later entries.

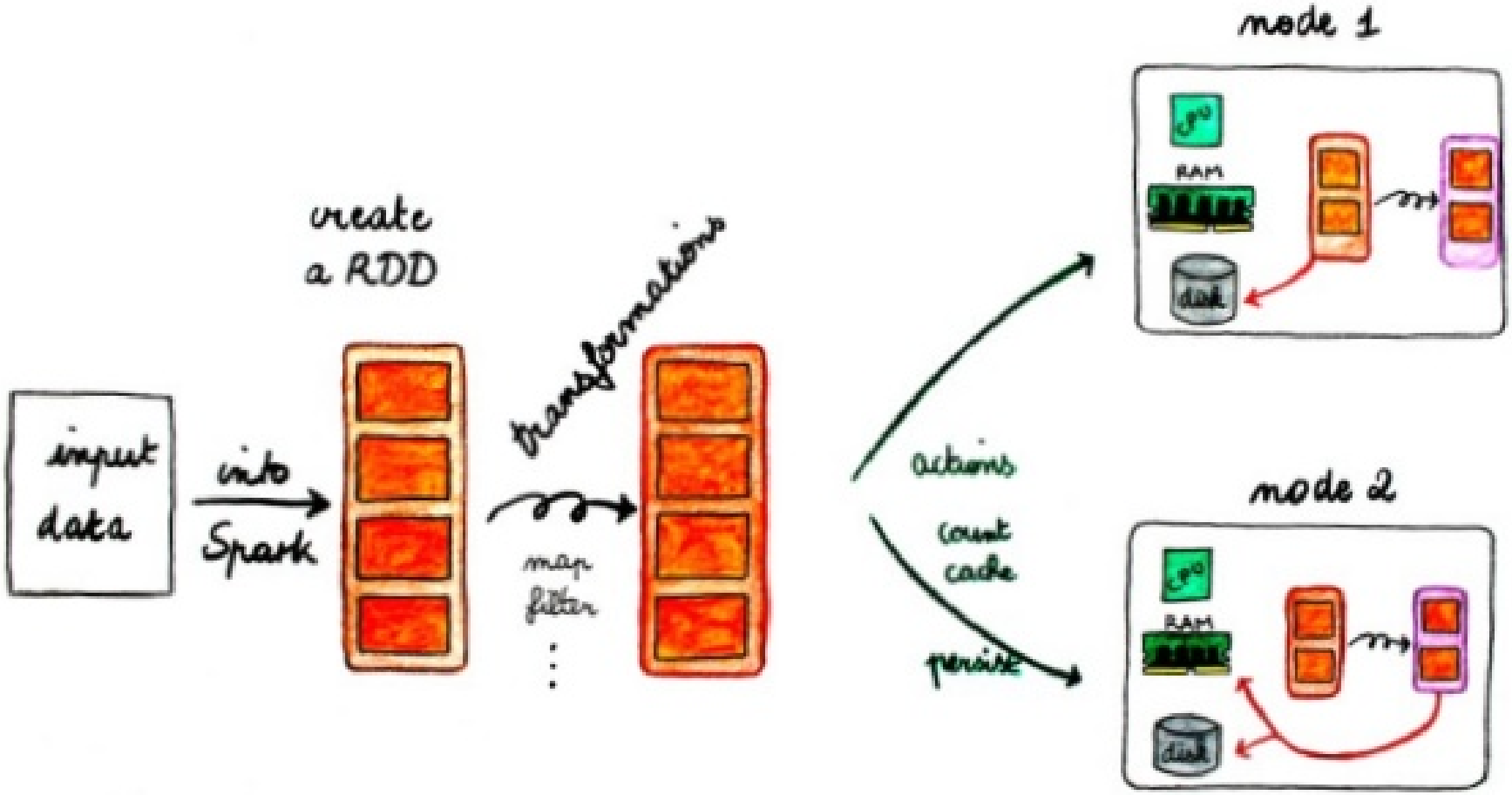


MY RECENT POSTS

- Airflow – XCOM
- Redis, Docker & Raspberry PI
- Spark & Redis
- Redis – Data Structures – Introductio
- Redis – Getting Started
- Airflow – Scale-out with Redis and Ce

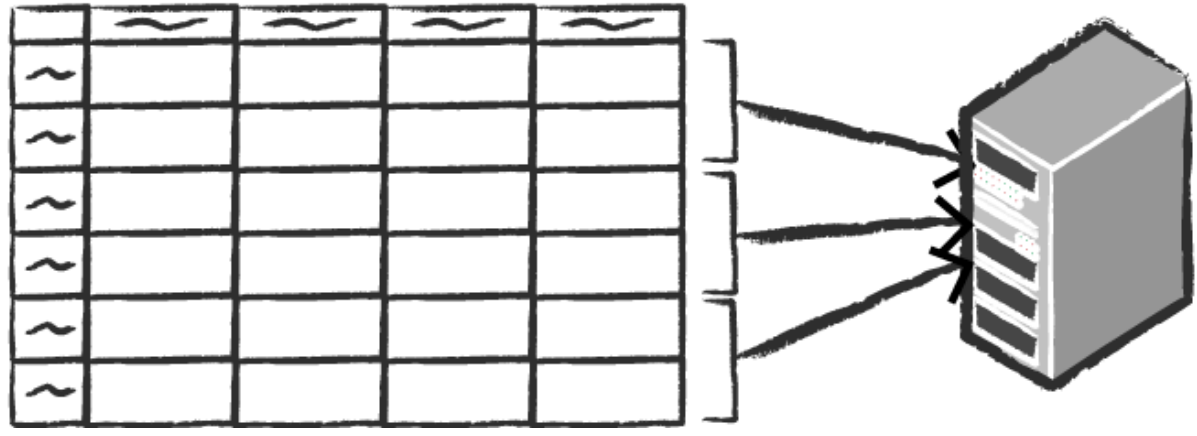

Driver Process
Executors
Spark Session
User code
Cluster Manager

Dataflow


input data
into Spark
create a RDD
transformations
map
filter
actions
count
cache
persist
node 1
node 2
CPU
RAM
disk

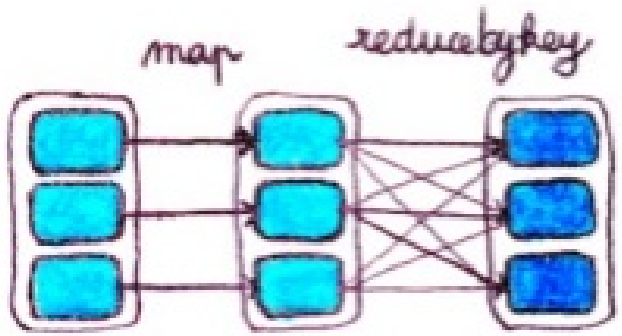
Spreadsheet on
a single machine

Table or Data Frame
partitioned across servers
in a data center

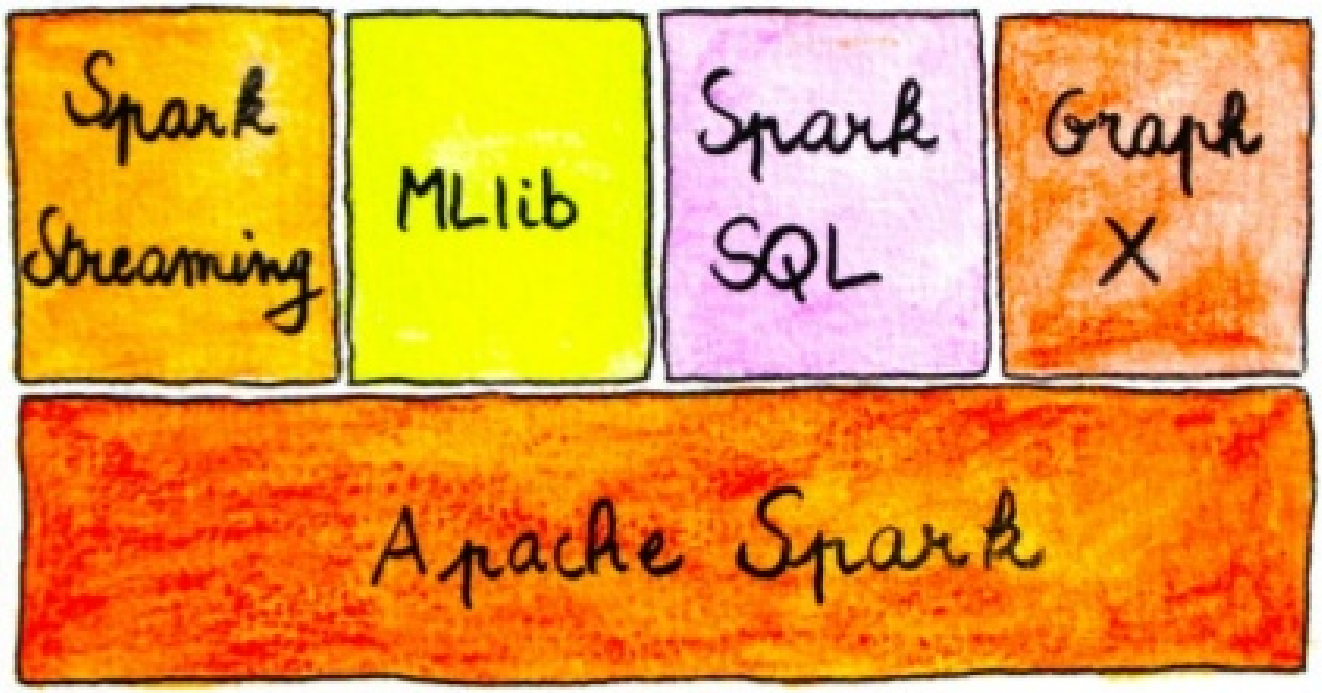


Example : Wordcount

```
// create configuration for Spark and the context
val conf = new SparkConf()
                .setAppName("Spark word count")
                .setMaster("local")

val sc = new SparkContext(conf)

// load the data
val data = sc.textFile("filepath/wordcount.txt")

// map then reduce step
val wordCounts = data.flatMap(line => line.split("\\s+"))
                     .map(word => (word, 1))
                     .reduceByKey(_ + _)
// persist the data
wordCounts.cache()
```

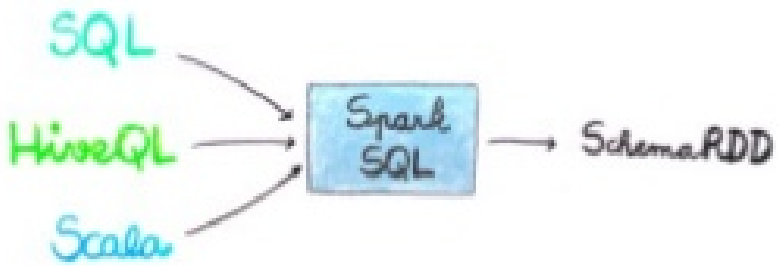


Spark ecosystem



Spark SQL

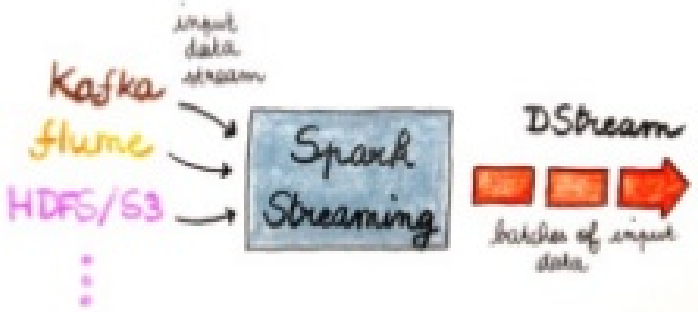
unifies access to structured data




```
// create a sql context from the Spark context
val sqlContext = new SQLContext(sc);

// load data and create an RDD
val tweets = sqlContext.jsonFile(pathToFile);
// register tweets as a table to operate on it later
tweets.registerAsTable("tweet");


// make sql request on RDD
val nb = sqlContext.sql("SELECT user, COUNT(*) AS c FROM tweet " +
                                    "WHERE user <> '' " +
                                    "GROUP BY user " +
                                    "ORDER BY c ");
```

Spark Streaming

makes it easy to build scalable fault-tolerant streaming applications




```
// create a java streaming context and define the window
val jssc = new StreamingContext(conf, Durations.seconds(10))

// create our DStream (sequence of RDD)
val tweetsStream = TwitterUtils.createStream(jssc, StreamUtils.getAuth())

// find all user
val tweetUser = tweetsStream.map(tweetStatus => tweetStatus.getUser())
```

The SparkContext

- Created by your driver program
- Is responsible for making RDD's resilient and distributed!
- Creates RDD's
- The Spark shell creates a "sc" object for you

Sessão do Spark

Sessão do spark

```
spark
```

https://community.cloud.databricks.com/

Dataframe do Spark

Criando Dataframe com números

```
myRange = spark.range(1000).toDF("number")
```

https://community.cloud.databricks.com/

Dataframe do Spark

Filtrando Dataframe

```
divisBy2 = myRange.where("number % 2 = 0")
```

https://community.cloud.databricks.com/

Dataframe do Spark

Lendo dados de um csv

```
flightData2015 = spark\
.read\
.option("inferSchema", "true")\
.option("header", "true")\
.csv("/data/flight-data/csv/2015-summary.csv")
```

https://community.cloud.databricks.com/

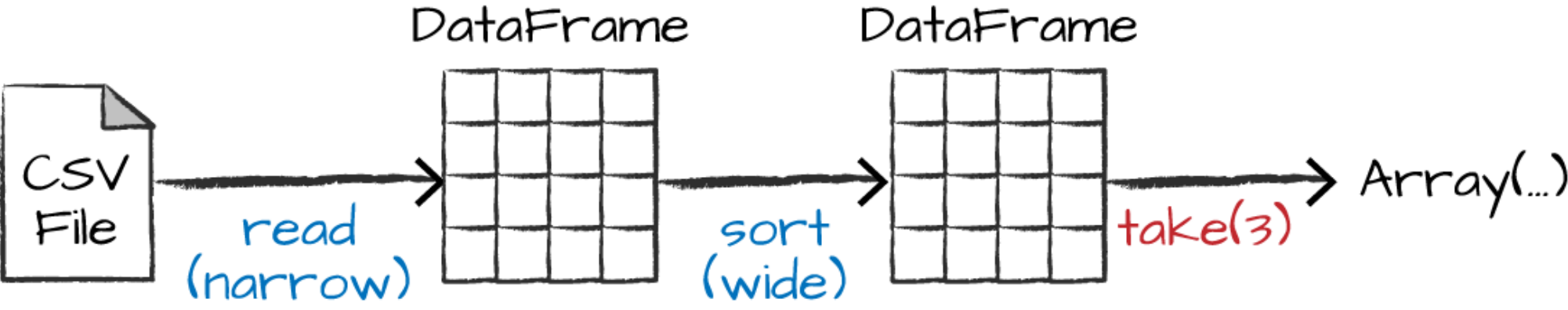

CSV
File
read
DataFrame
take(n)
Array(row(...),Row(...))

Dataframe do Spark

Lendo dados de um csv

```
df.take(3)
```

https://community.cloud.databricks.com/


DataFrame
DataFrame
CSV File
read (narrow)
sort (wide)
take(3)
Array(...)

Dataframe do Spark

Lendo dados de um csv

```
df.sort("count").explain()
```

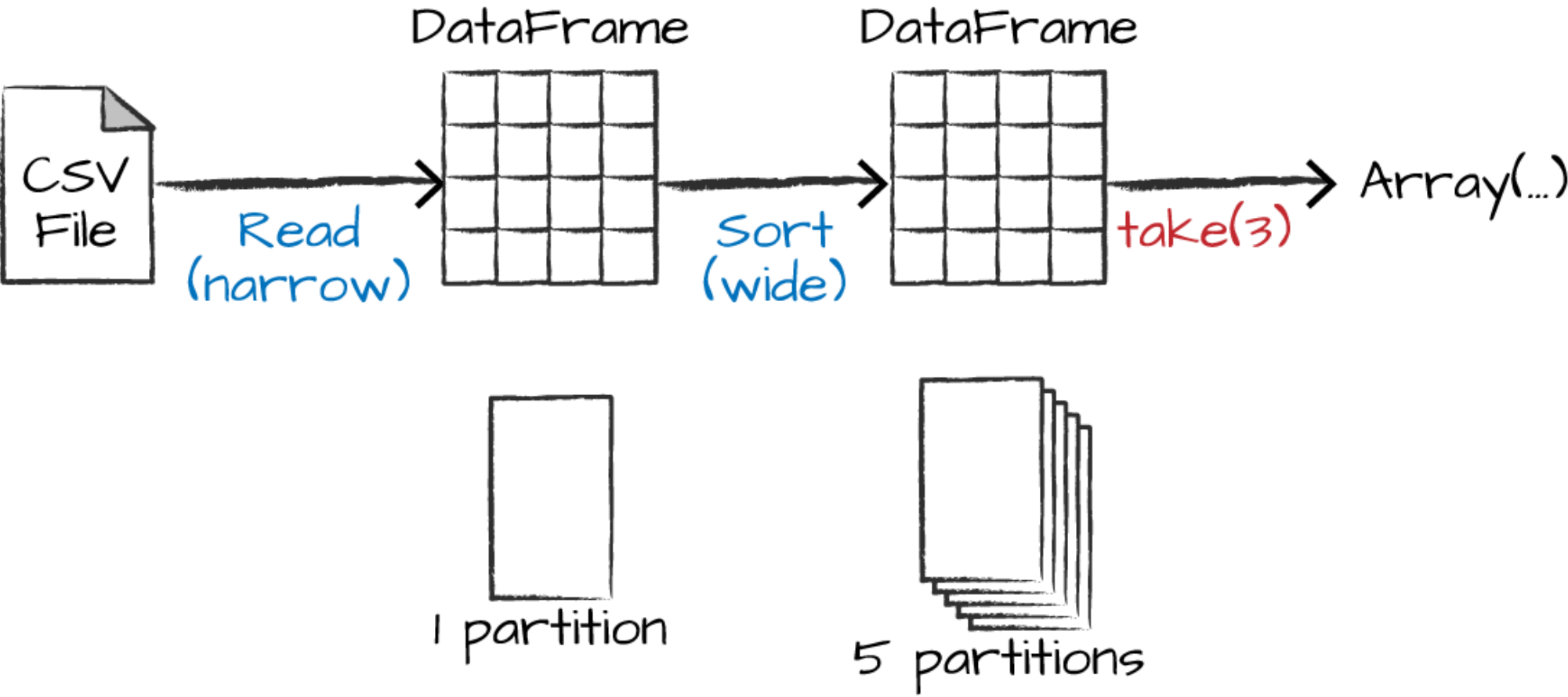
https://community.cloud.databricks.com/

Dataframe do Spark

Definindo numero de partições

```
spark.conf.set("spark.sql.shuffle.partitions", "5")
df.sort("count").take(2)
```

https://community.cloud.databricks.com/


DataFrame
DataFrame
CSV File
Read (narrow)
Sort (wide)
take(3)
Array(...)
1 partition
5 partitions

Dataframe do Spark

Comandos SQL

```
sqlWay = spark.sql("""
select * from `2015_voo`
""").show()
```

https://community.cloud.databricks.com/

Dataframe do Spark

Comandos SQL

```
sqlWay = spark.sql("""
SELECT DEST_COUNTRY_NAME, count(1)
FROM `2015_voo`
GROUP BY DEST_COUNTRY_NAME
""").orderBy("count(1)", ascending=False).show()
```

https://community.cloud.databricks.com/

Dataframe do Spark

Comandos SQL

```
dataFrameWay =df\
   .groupBy("DEST_COUNTRY_NAME")\
   .count().orderBy("count",ascending=False).show()
```

https://community.cloud.databricks.com/

Dataframe do Spark

Comandos SQL

```
sqlWay = spark.sql("""
SELECT DEST_COUNTRY_NAME, count(1)
FROM `2015_voo`
GROUP BY DEST_COUNTRY_NAME
""").orderBy("count(1)", ascending=False)
sqlWay.explain()
```

Dataframe do Spark

Comandos SQL

```
== Physical Plan ==
Sort [count(1)#291L DESC NULLS LAST], true, 0
+- Exchange rangepartitioning(count(1)#291L DESC NULLS L
   +- *(2) HashAggregate(keys=[DEST_COUNTRY_NAME#136], f
      +- Exchange hashpartitioning(DEST_COUNTRY_NAME#136
         +- *(1) HashAggregate(keys=[DEST_COUNTRY_NAME#1
            +- *(1) FileScan csv [DEST_COUNTRY_NAME#136]
```

Dataframe do Spark

Comandos SQL

```
from pyspark.sql.functions import max
df.select(max("count")).take(1)
```

Dataframe do Spark

Comandos SQL

```
maxSql = spark.sql("""
SELECT DEST_COUNTRY_NAME, sum(count) as destination_tota
FROM 2015_voo
GROUP BY DEST_COUNTRY_NAME
ORDER BY sum(count) DESC
LIMIT 5
""")

maxSql.show()
```

Dataframe do Spark

Comandos SQL

```
df=df.withColumn("count_int", df["count"].cast("Int"))
```

Dataframe do Spark

Comandos SQL

```
df\
   .groupBy("DEST_COUNTRY_NAME")\
   .sum("count_int")\
   .withColumnRenamed("sum(count_int)", \
   "destination_total")\
   .sort(desc("destination_total"))\
   .limit(5)\
   .explain()
```

Dataframe do Spark

Lendo arquivos com wildcard

```
staticDataFrame = spark.read.format("csv")
        .option("header", "true")
        .option("inferSchema", "true")\
        .load("/FileStore/tables/2011*.csv")
```

Dataframe do Spark

Criando visão temporária

```
staticDataFrame.createOrReplaceTempView("retail_data")
staticSchema = staticDataFrame.schema
```

Dataframe do Spark

Realizando operações

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

staticDataFrame\
.selectExpr(    "CustomerId",
"(UnitPrice * Quantity) as total_cost",
"InvoiceDate")\
.groupBy(    col("CustomerId"),
window(col("InvoiceDate"), "1 day"))\
.sum("total_cost")\
.show(5)
```

Dataframe do Spark

Realizando operações de Streaming

```
streamingDataFrame = spark.readStream\
.schema(staticSchema)\
.option("maxFilesPerTrigger", 1)\
.format("csv")\
.option("header", "true")\
.load("/FileStore/tables/2011*.csv")
```

Dataframe do Spark

Realizando operações de Streaming

```
streamingDataFrame.isStreaming
```

Dataframe do Spark

Realizando operações de Streaming

```
purchaseByCustomerPerHour = streamingDataFrame\
.selectExpr(    "CustomerId",

"(UnitPrice * Quantity) as total_cost",

"InvoiceDate")\

.groupBy(    col("CustomerId"),
window(col("InvoiceDate"), "1 day"))\
.sum("total_cost")
```

Dataframe do Spark

Escrevendo dados na memória

```
purchaseByCustomerPerHour.writeStream\
.format("memory")\
.queryName("customer_purchases")\
.outputMode("complete")\
.start()
```

Dataframe do Spark

Lendo dados da memória

```
# in Python
spark.sql("""  SELECT *  FROM customer_purchases  ORDER
""")\
.show(5)
```

Dataframe do Spark

Escrevendo dados para o console

```
purchaseByCustomerPerHour.writeStream\
.format("console")\
.queryName("customer_purchases_2")\
.outputMode("complete")\
.start()
```

Dataframe do Spark

Escrevendo dados para o console

```
# in Python
from pyspark.sql.functions import date_format, col
preppedDataFrame = staticDataFrame\
.na.fill(0)\
.withColumn("day_of_week", date_format(col("InvoiceDate"
.coalesce(5)
```

Spak-submit

Spark Submit é uma ferramenta de linha de comando que permite executar o código em um cluster

```
./bin/spark-submit \
--class org.apache.spark.examples.SparkPi \
--master local \
./examples/jars/spark-examples_2.11-2.2.0.jar 10
```

Spak-submit

Spark Submit é uma ferramenta de linha de comando que permite executar o código em um cluster

```
./bin/spark-submit \
--master local \
./examples/src/main/python/pi.py 10
```

Criando script spark

Retornando pior avaliação de filmes

```
from pyspark import SparkConf, SparkContext

# This function just creates a Python "dictionary" we ca
# use to convert movie ID's to movie names while printin
# the final results.
def loadMovieNames():
    movieNames = {}
    with open("ml-100k/u.item") as f:
        for line in f:
            fields = line.split('|')
            movieNames[int(fields[0])] = fields[1]
    return movieNames
```

Criando script spark

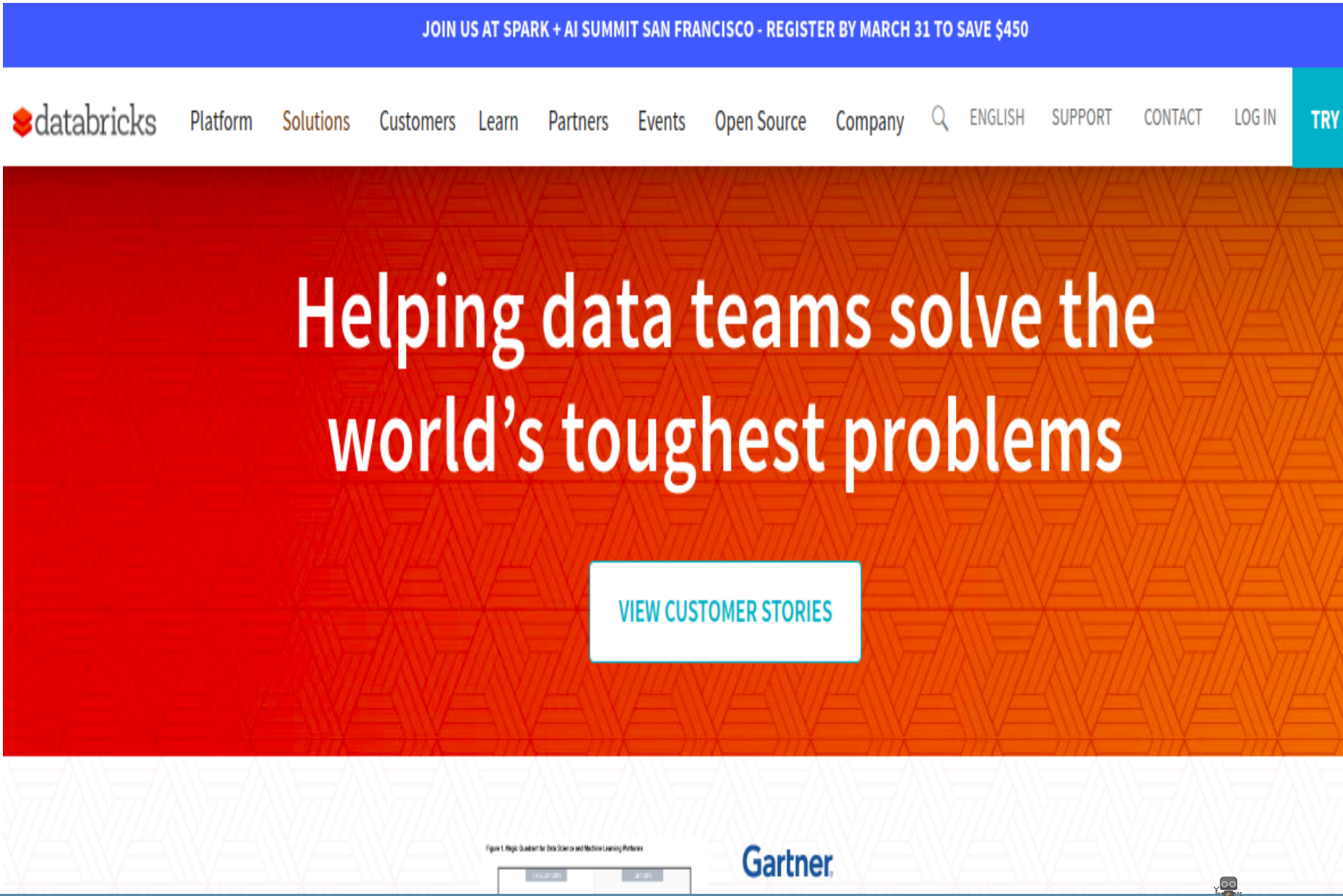
Retornando pior avaliação de filmes

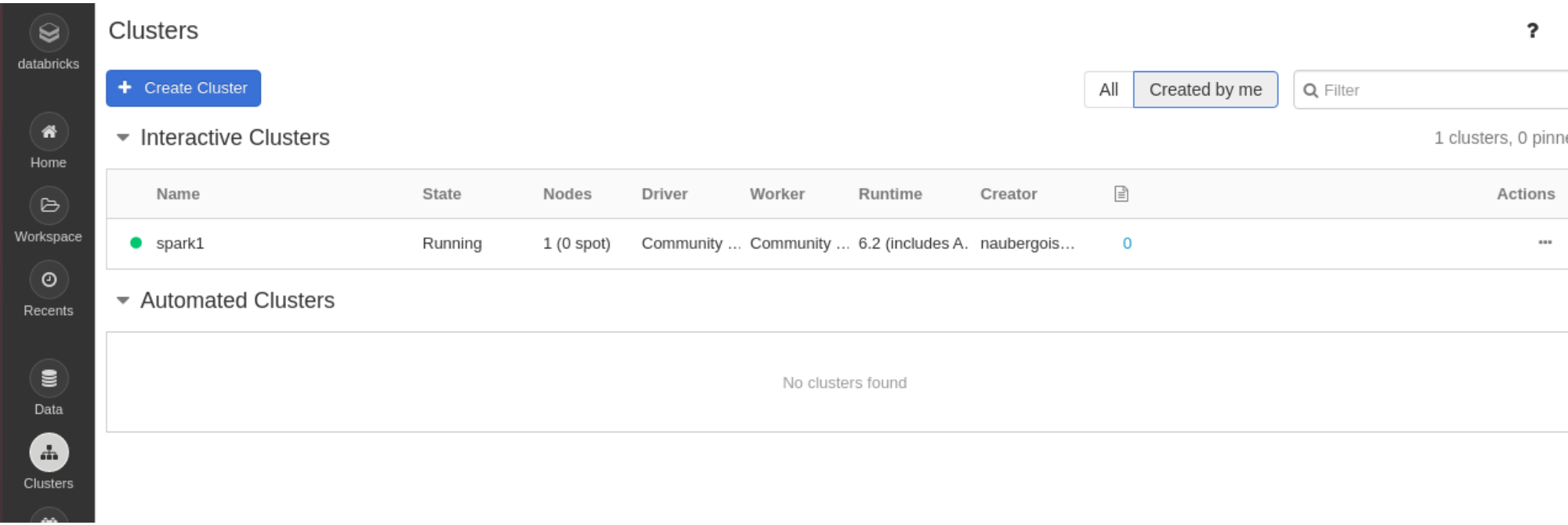
```
if __name__ == "__main__":
    # The main script - create our SparkContext
    conf = SparkConf().setAppName("WorstMovies")
    sc = SparkContext(conf = conf)

    # Load up our movie ID -> movie name lookup table
    movieNames = loadMovieNames()

    # Load up the raw u.data file
    lines = sc.textFile("hdfs:///user/maria_dev/ml-100k/

    # Convert to (movieID, (rating, 1.0))
    movieRatings = lines.map(parseInput)
```

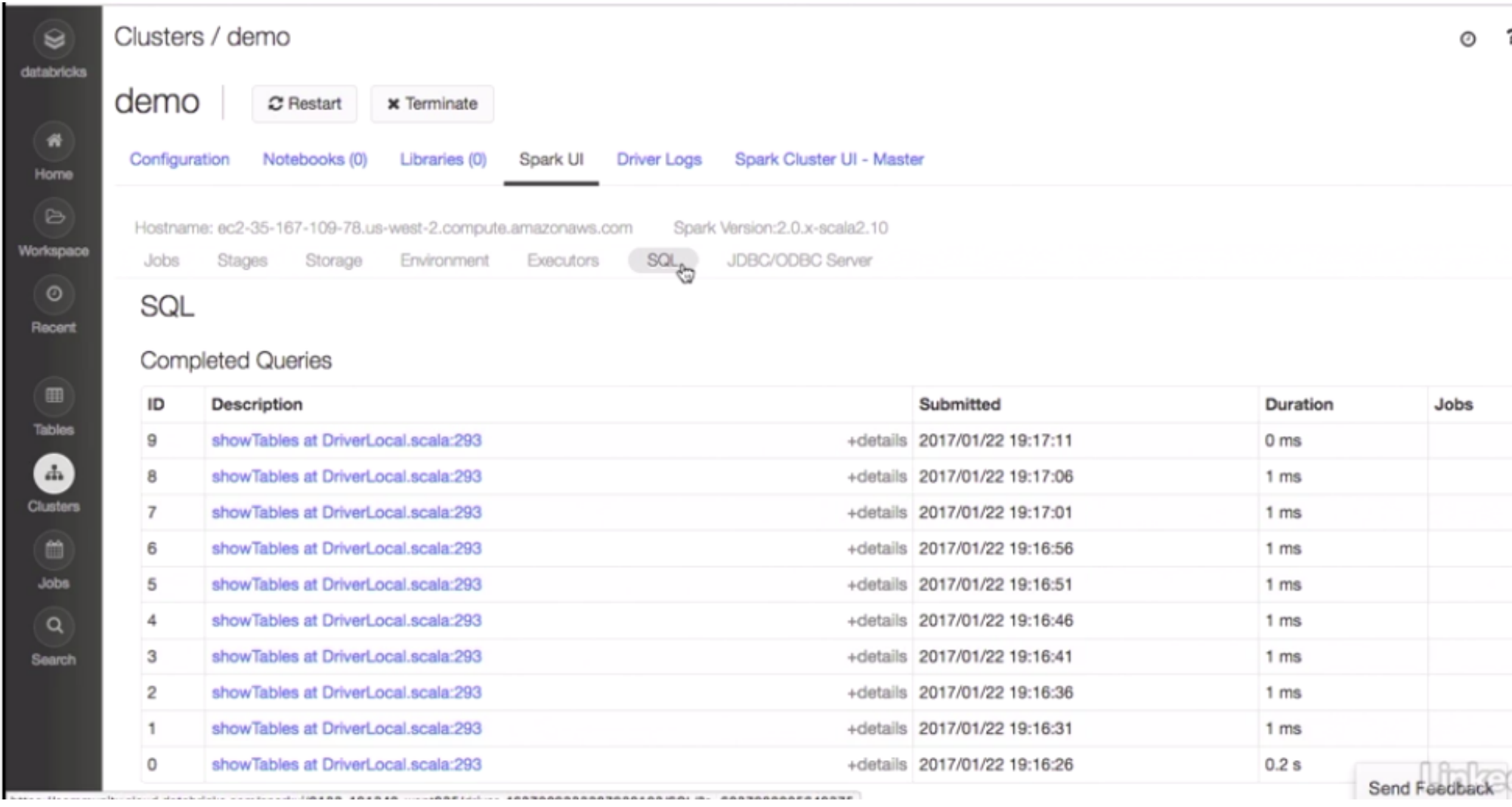
Criando script spark

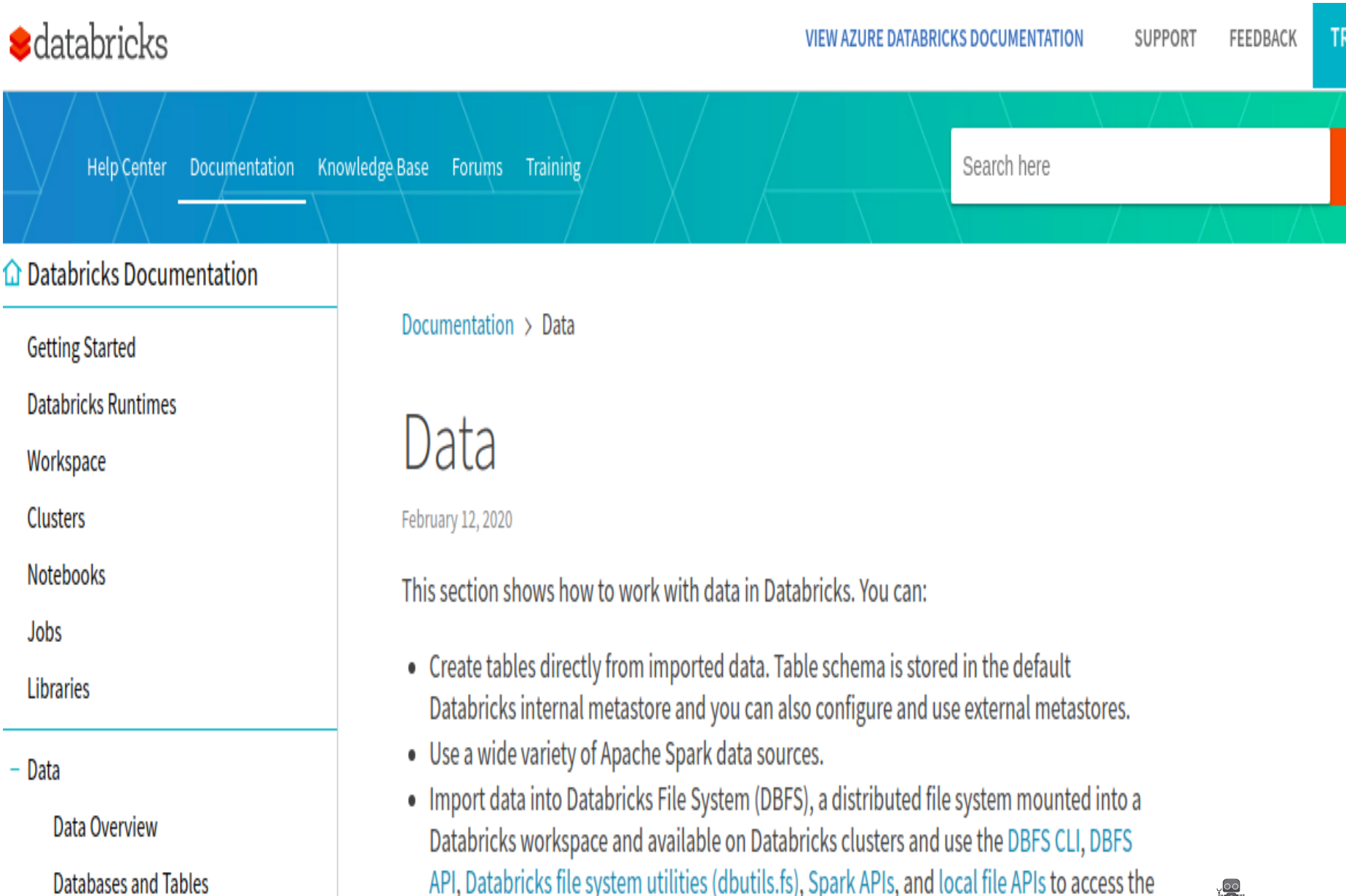
Retornando pior avaliação de filmes

```
spark-submit LowestRatedMovieSpark.py
```


JOIN US AT SPARK + AI SUMMIT SAN FRANCISCO - REGISTER BY MARCH 31 TO SAVE $450
databricks
Platform
Solutions
Customers
Learn
Partners
Events
Open Source
Company
ENGLISH
SUPPORT
CONTACT
LOG IN
TRY
Helping data teams solve the world's toughest problems
VIEW CUSTOMER STORIES
Gartner


databricks
Home
Workspace
Recents
Data
Clusters
Clusters
?
Create Cluster
All
Created by me
Filter
Interactive Clusters
1 clusters, 0 pinn
Name
State
Nodes
Driver
Worker
Runtime
Creator
Actions
spark1
Running
1 (0 spot)
Community ...
Community ...
6.2 (includes A.
naubergois...
0
Automated Clusters
No clusters found


databricks
Home
Workspace
Recent
Tables
Clusters
Jobs
Search
Clusters / demo
demo
Restart
Terminate
Configuration
Notebooks (0)
Libraries (0)
Spark UI
Driver Logs
Spark Cluster UI - Master
Hostname: ec2-35-167-109-78.us-west-2.compute.amazonaws.com
Spark Version:2.0.x-scala2.10
Jobs
Stages
Storage
Environment
Executors
SQL
JDBC/ODBC Server
SQL
Completed Queries
ID
Description
Submitted
Duration
Jobs
9 showTables at DriverLocal.scala:293 +details 2017/01/22 19:17:11 0 ms
8 showTables at DriverLocal.scala:293 +details 2017/01/22 19:17:06 1 ms
7 showTables at DriverLocal.scala:293 +details 2017/01/22 19:17:01 1 ms
6 showTables at DriverLocal.scala:293 +details 2017/01/22 19:16:56 1 ms
5 showTables at DriverLocal.scala:293 +details 2017/01/22 19:16:51 1 ms
4 showTables at DriverLocal.scala:293 +details 2017/01/22 19:16:46 1 ms
3 showTables at DriverLocal.scala:293 +details 2017/01/22 19:16:41 1 ms
2 showTables at DriverLocal.scala:293 +details 2017/01/22 19:16:36 1 ms
1 showTables at DriverLocal.scala:293 +details 2017/01/22 19:16:31 1 ms
0 showTables at DriverLocal.scala:293 +details 2017/01/22 19:16:26 0.2 s
Send Feedback


databricks
VIEW AZURE DATABRICKS DOCUMENTATION
SUPPORT
FEEDBACK
Help Center
Documentation
Knowledge Base
Forums
Training
Search here
Databricks Documentation
Getting Started
Databricks Runtimes
Workspace
Clusters
Notebooks
Jobs
Libraries
Data
Data Overview
Databases and Tables
Documentation › Data
Data
February 12, 2020
This section shows how to work with data in Databricks. You can:
Create tables directly from imported data. Table schema is stored in the default Databricks internal metastore and you can also configure and use external metastores.
Use a wide variety of Apache Spark data sources.
Import data into Databricks File System (DBFS), a distributed file system mounted into a Databricks workspace and available on Databricks clusters and use the DBFS CLI, DBFS API, Databricks file system utilities (dbutils.fs), Spark APIs, and local file APIs to access the

Conclusão

Você esta apto a:

Compreender o que é o Spark Aprender noções básicas do Spark

Bibliografia I

Bill Chambers and Matei Zaharia, Spark: The definitive guide: Big data processing made simple, ”O’Reilly Media, Inc.”, 2018.