ARA0168
TÓPICOS DE BIG DATA EM PYTHON

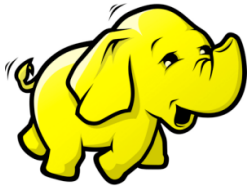
Aula 5 – Ecossistema *Hadoop*: Apache Sparks

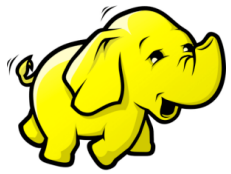
Universidade Estácio de Sá

Prof. Simone Gama

simone.gama@estacio.br

Uma das principais atribuições do **Arquiteto de Dados** é a "criação de uma visão *end-to-end* do fluxo de dados / pipeline de dados".
Autor desconhecido

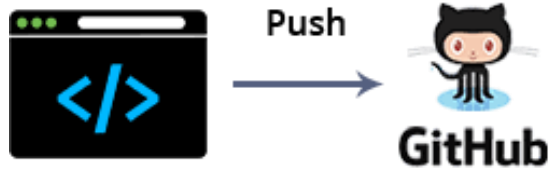




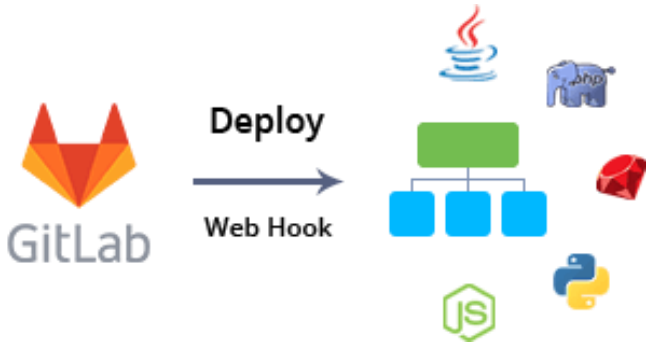
Ecossistema *Hadoop*: Definições

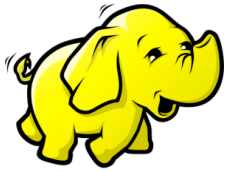
Deploy

Na engenharia de dados, quando falamos em **disponibilização de códigos em um servidor** ou em **uma aplicação na nuvem**, estamos falando de um ***deploy***.



Ou seja, é a tarefa de **exportar o programa** da sua máquina para um servidor — com o objetivo de colocá-lo em produção.



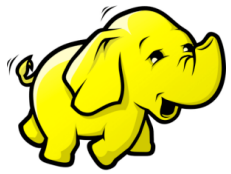


Ecossistema *Hadoop*: Definições

Pipeline (Dados)

Um ***pipeline*** tem como objetivo **mover os dados de um lugar para outro**, e geralmente é similar ao conceito de **ETL** (*extract*, *transform*, *load*).

Em um *pipeline* de dados, é possível ter uma série de processos ou serviços em sequência — que realizam a extração dos dados da fonte para o destino (que pode ser um *data lake*, por exemplo).



Ecossistema *Hadoop*: Definições

Pipeline (Dados) – Leituras Recomendadas

[1] – Princípios de Boas Práticas em *Pipelines*. Link: Sete princípios para pipelines de dados confiáveis | by Ricardo Pinto | Data Hackers | Medium

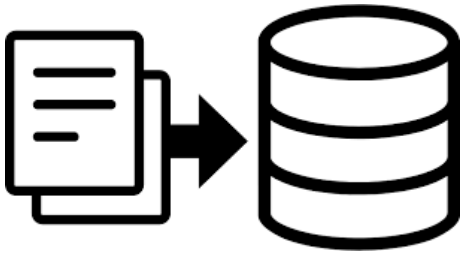
[2] - *Scalable Efficient Big Data Pipeline Architecture*. Link: Scalable Efficient Big Data Pipeline Architecture – Machine Learning for Developers (ml4devs.com)

Ecossistema *Hadoop*: Definições

Dataset

A *Dataset* is a set or collection of data.

This set is normally presented in a tabular pattern. Every column describes a particular variable. And each row corresponds to a given member of the data set, as per the given question. This is a part of data management.



Ecossistema *Hadoop*: Definições

Kubernetes or K8s

Os ***kubernetes***, ou "***k8s***", é uma plataforma *open-source* que automatiza as operações dos containers Linux.

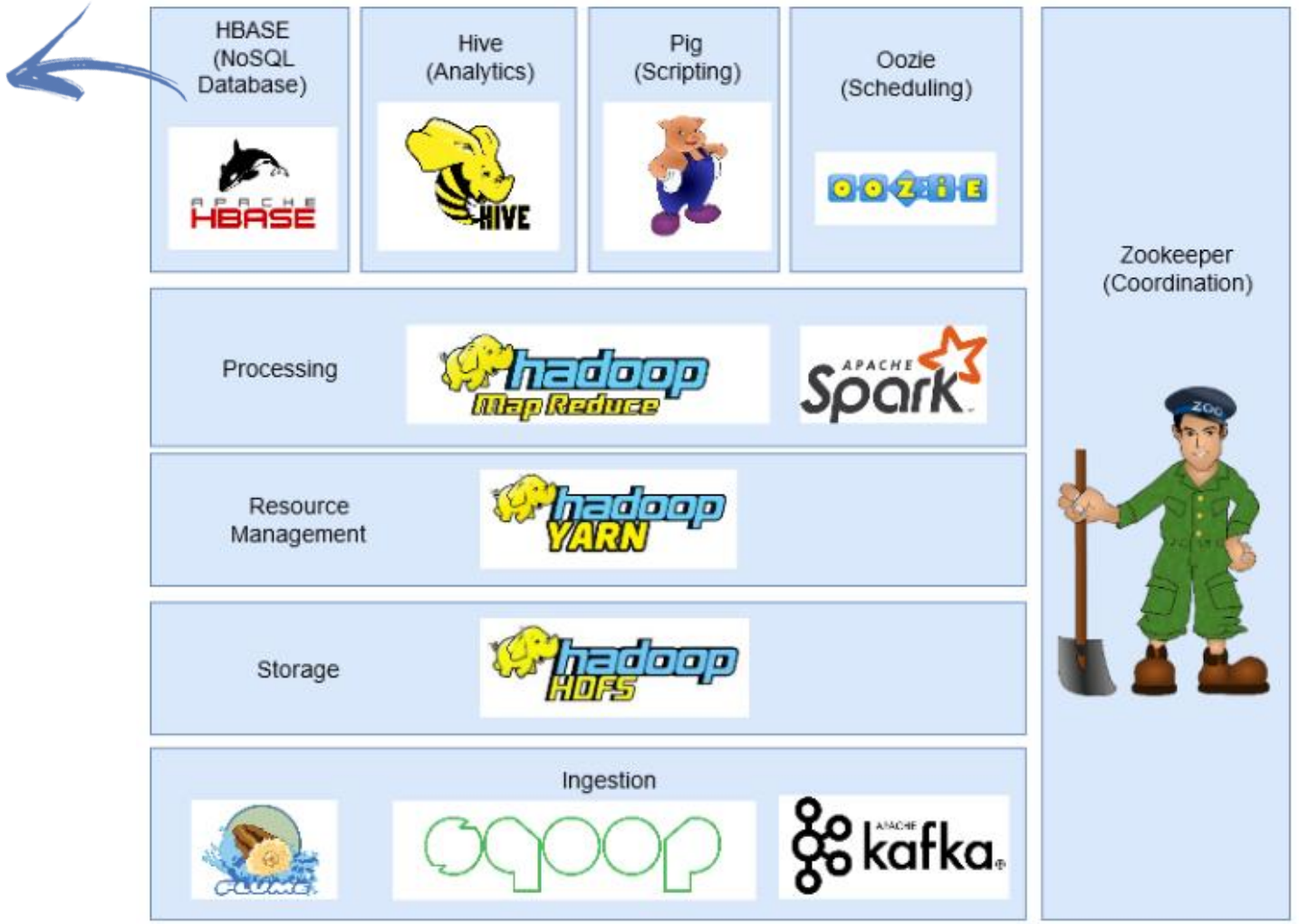
O Kubernetes elimina grande parte dos processos manuais necessários para implantar e escalar as aplicações em ***containers***.

Foi originalmente projetado pelo Google e agora é mantido pela *Cloud Native Computing Foundation*. Funciona com uma variedade de ferramentas de conteinerização, incluindo **Docker**.[1,2]

[1]Fonte: Kubernetes – Wikipédia, a enciclopédia livre (wikipedia.org)
[2]Fonte: O que é Kubernetes (K8s)? (redhat.com)

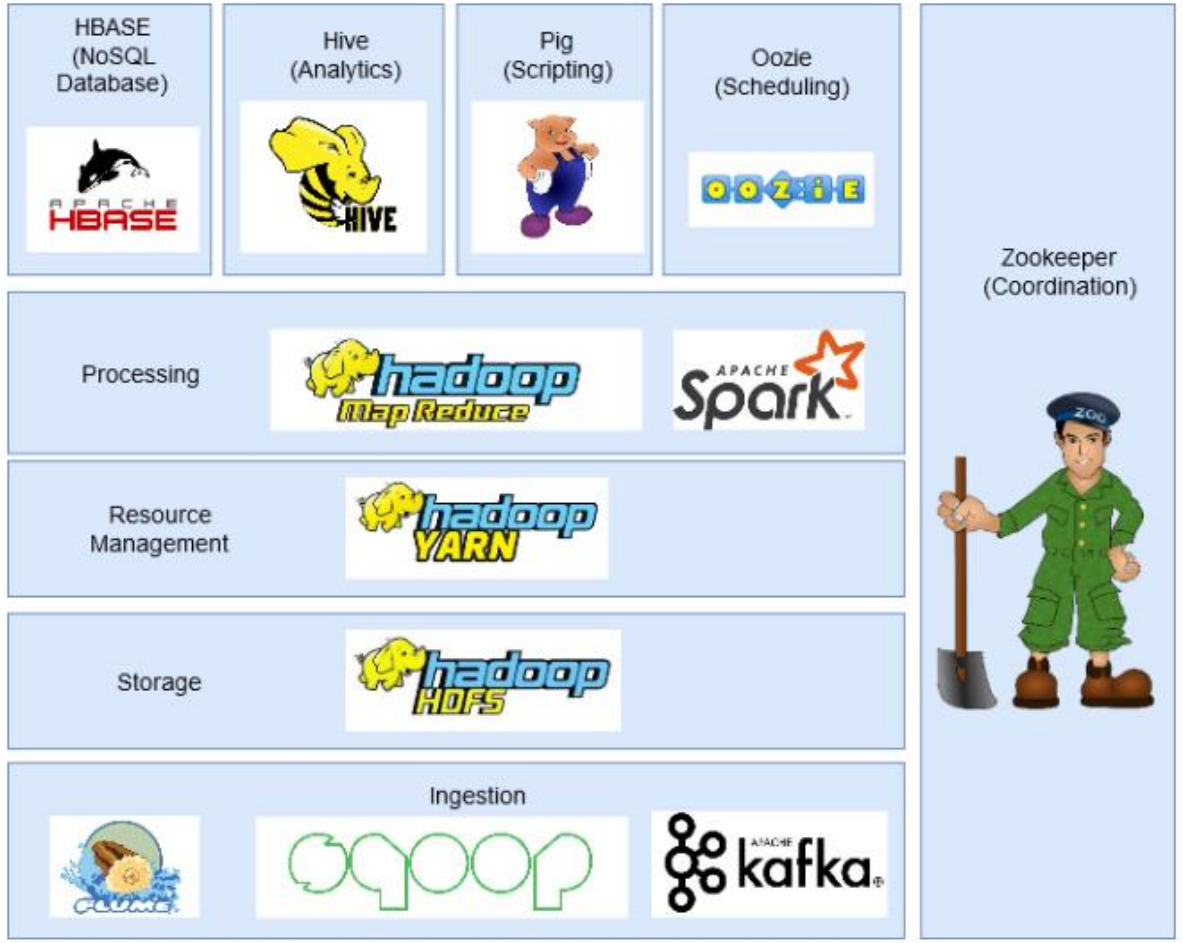
Ecossistema *Hadoop*

Ecossistema *Hadoop*



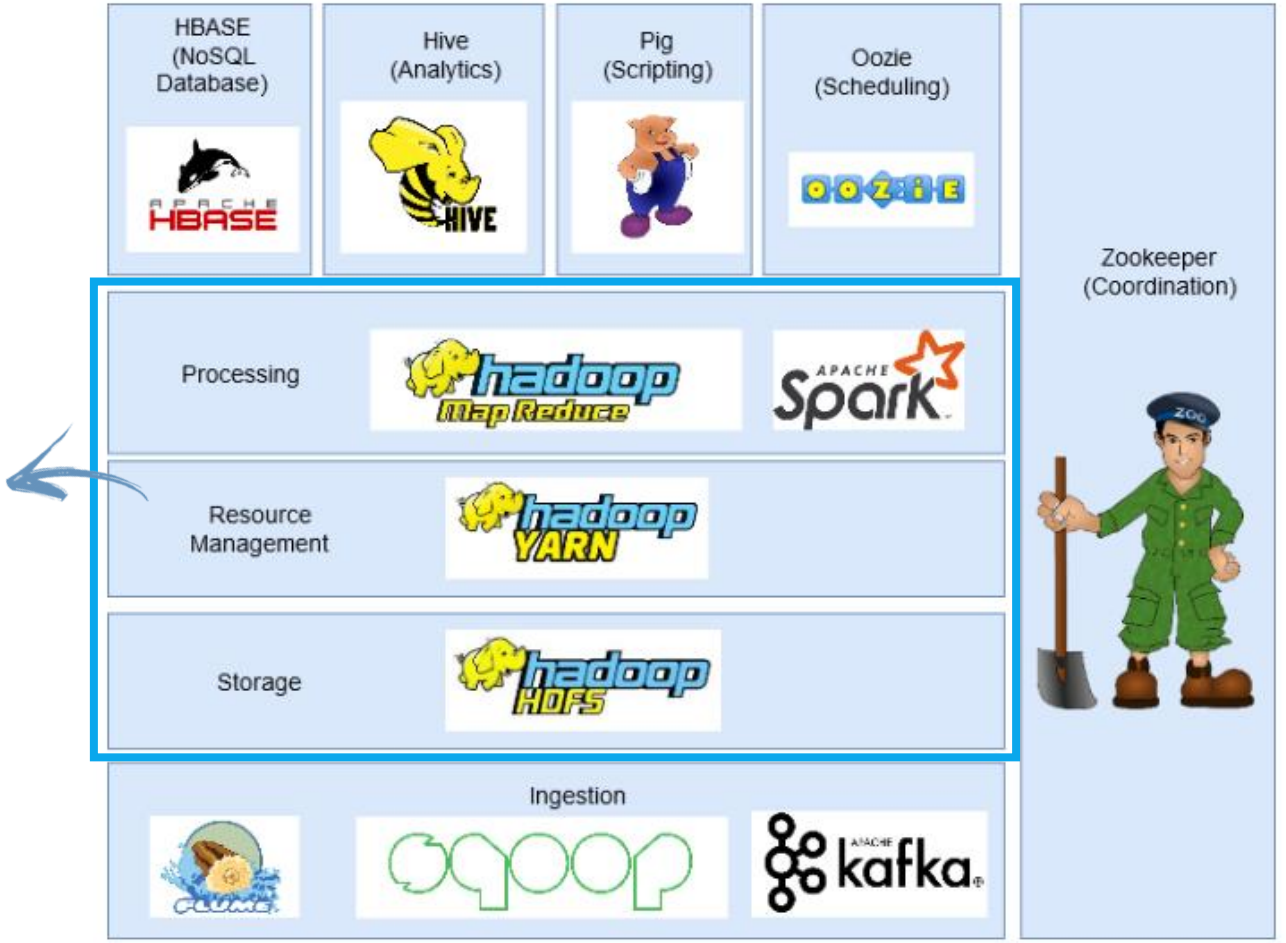
Ecossistema *Hadoop*

Dependendo da **Regra de Negócios** aplicada ou de outros requisitos, o **Ecossistema *Hadoop*** pode ser composto dos mais diversos componentes.



Ecossistema *Hadoop*

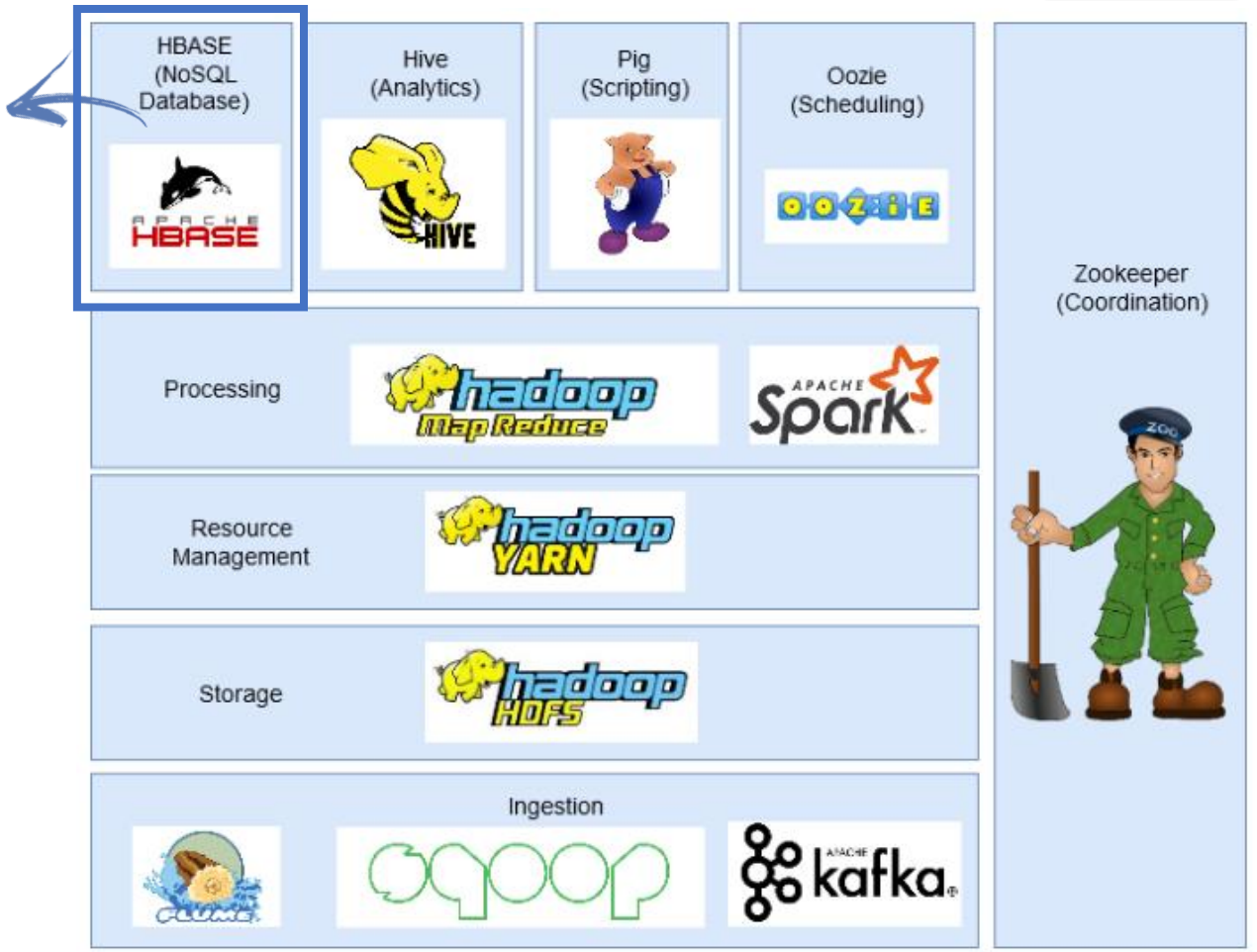
Estamos aqui...



Ecossistema *Hadoop*

HBASE

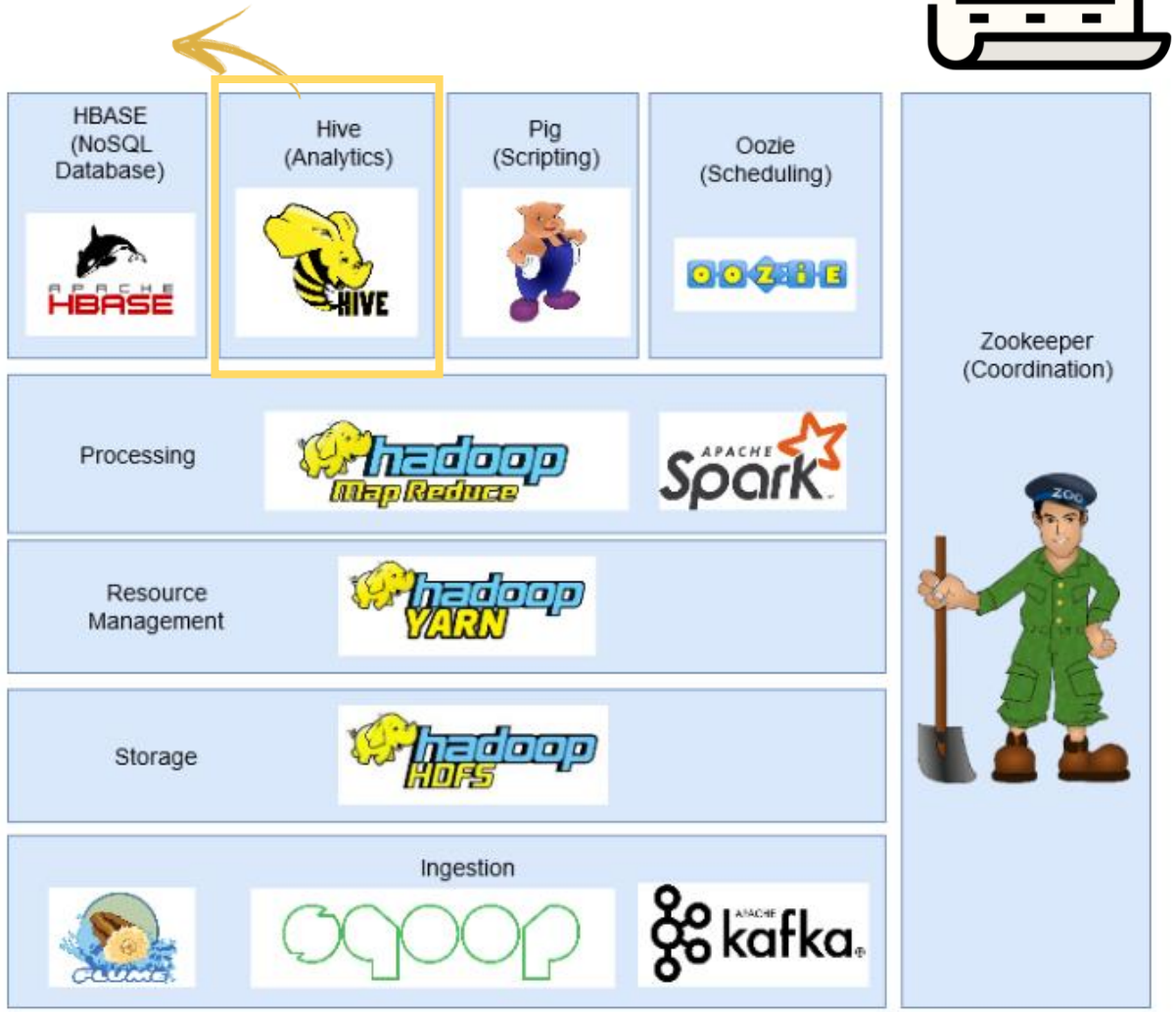
O HBase é um **banco de dados NoSQL**, *open-source* com estrutura colunar. Roda com o HDFS e pode trabalhar com diversos formatos de dados, permitindo o processamento em tempo real e randômico de leitura/gravação nos dados. A função principal dele é hospedar grandes tabelas – bilhões de linhas x milhões de colunas – sobre clusters de hardware comum, muito semelhante ao **HDFS**.



Ecossistema *Hadoop*

Hive

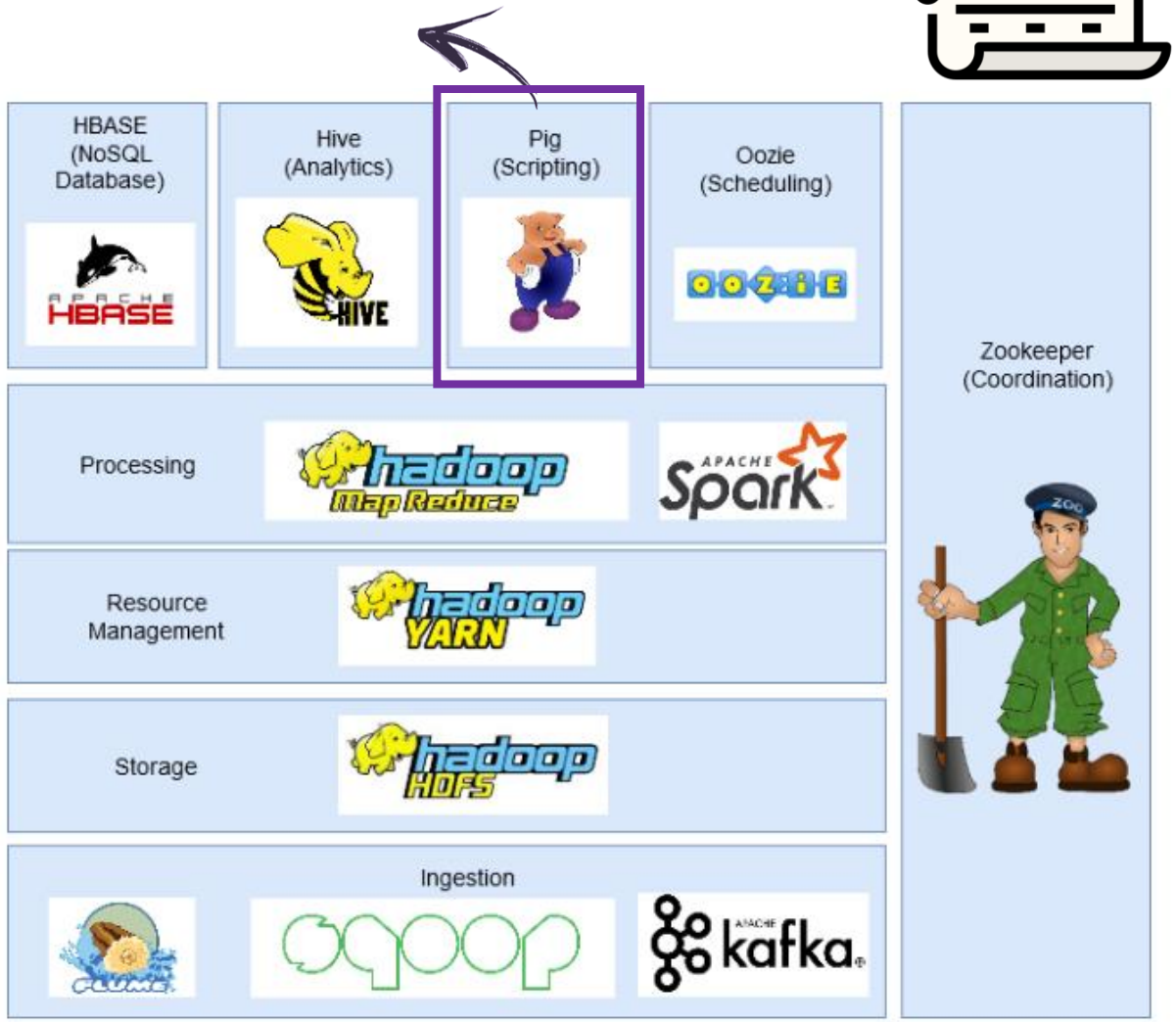
O Hive é um **sistema de data warehouse** distribuído desenvolvido pelo facebook. Permite de maneira simples, ler, gravar e gerenciar grandes arquivos no HDFS. Tem sua própria linguagem de consulta chamda HQL – Hive Querie Language – que é muito similar à linguagem SQL. Simplifica a escrita de funções MapReduce usando a linguagem HQL.



Ecossistema *Hadoop*

Apache Pig

Esse componente do ecossistema Hadoop originou-se de um projeto desenvolvido pelo Yahoo! por volta de 2006 porque sentiram a necessidade de ter um ferramenta para executarem *jobs MapReduce* de maneira *ad-hoc*, ou mais simplificada para os usuários. O Pig foi desenvolvido para analisar grandes conjuntos de dados e simplifica a escrita de funções *map reduce* ou *Spark*.



Ecossistema *Hadoop*

Oozie

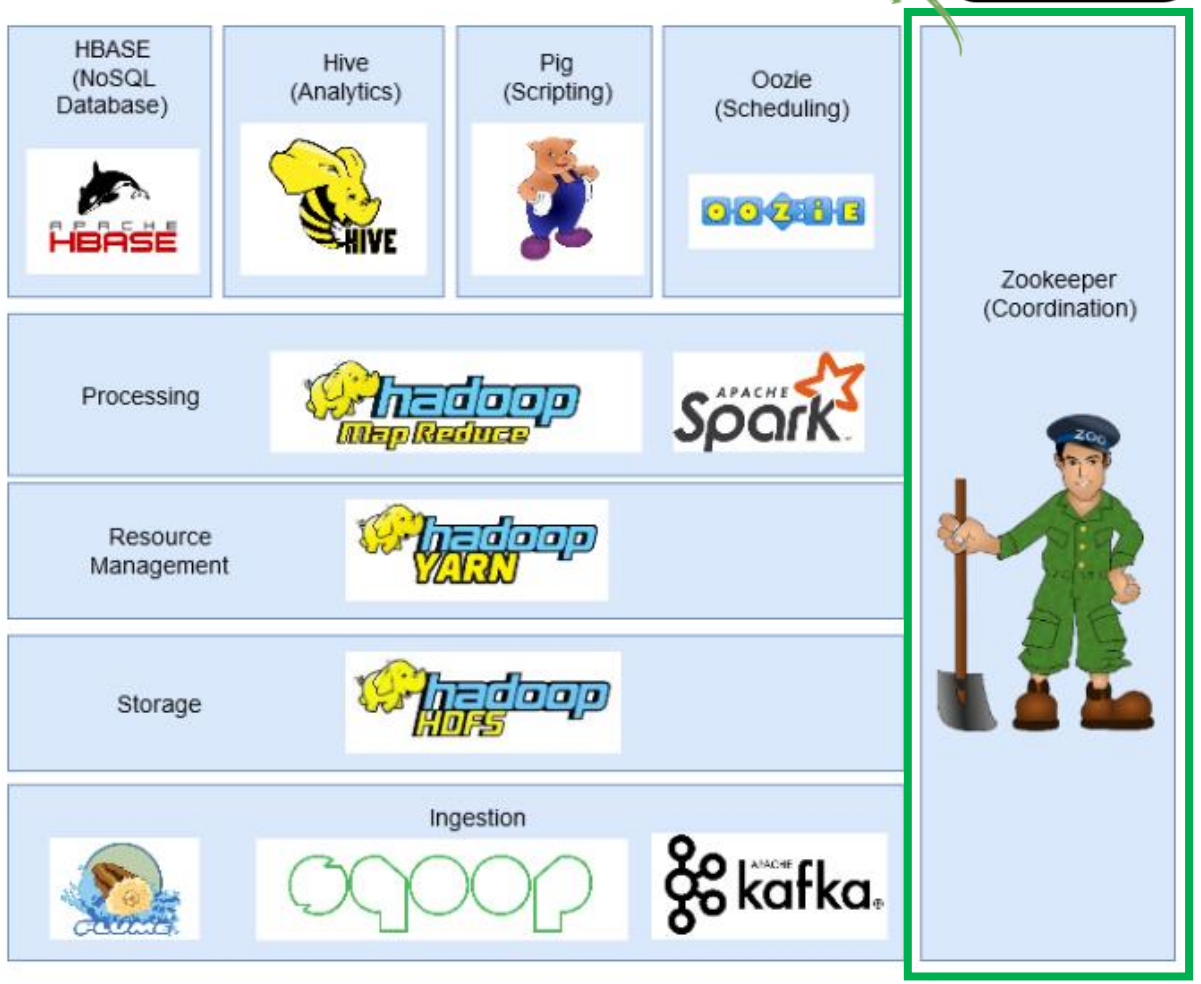
O Oozie é um agendador de *workflows* que permite os usuários fazerem o agendamento dos jobs desenvolvidos em várias plataformas, como MapReduce, Pig, Hive e etc.
Usando o **Oozie** é possível criar um job que possa chamar de maneira orquestrada, outros jobs ou pipelines de dados, seja de maneira sequencial ou paralela para executar uma determinada tarefa.

Ecossistema *Hadoop*

Zookeeper

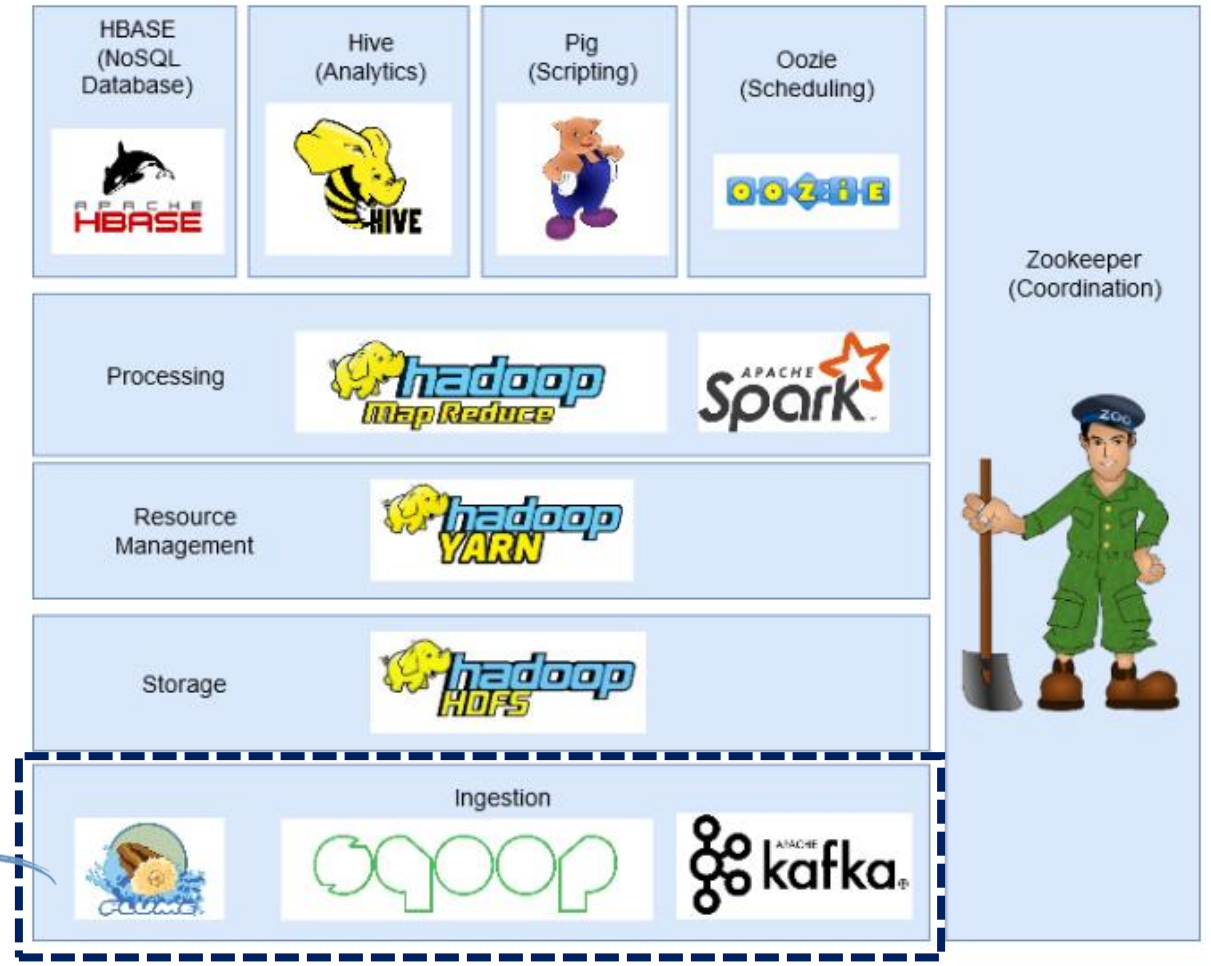
Um ambiente Hadoop é bastante desafiador. **Sincronizar**, **coordenar** e **manter as configurações** de um ambiente de *cluster Hadoop* exige bastante esforço, para resolver esse problema entra em cena o Zookeeper.

Ele é um software *open-source*, distribuído, e com um serviço centralizado para manter as informações de configuração.



Ecossistema *Hadoop*: Etapas Big Data

Etapas de um processo de Big Data

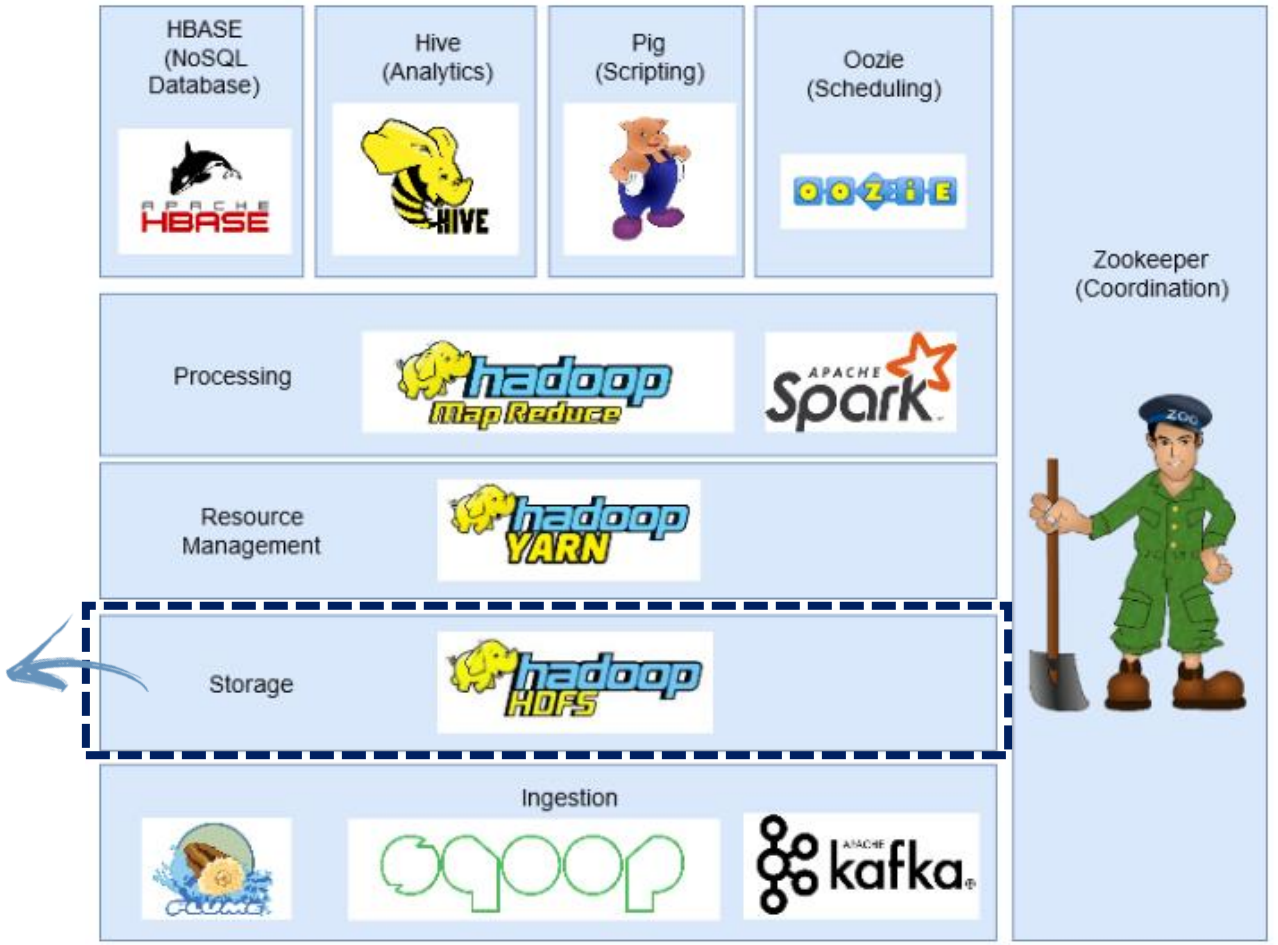


Flume, Kafka, Sqoop são utilizados para fazer a **ingestão de dados** no *HDFS*.

Ecossistema *Hadoop*: Etapas Big Data

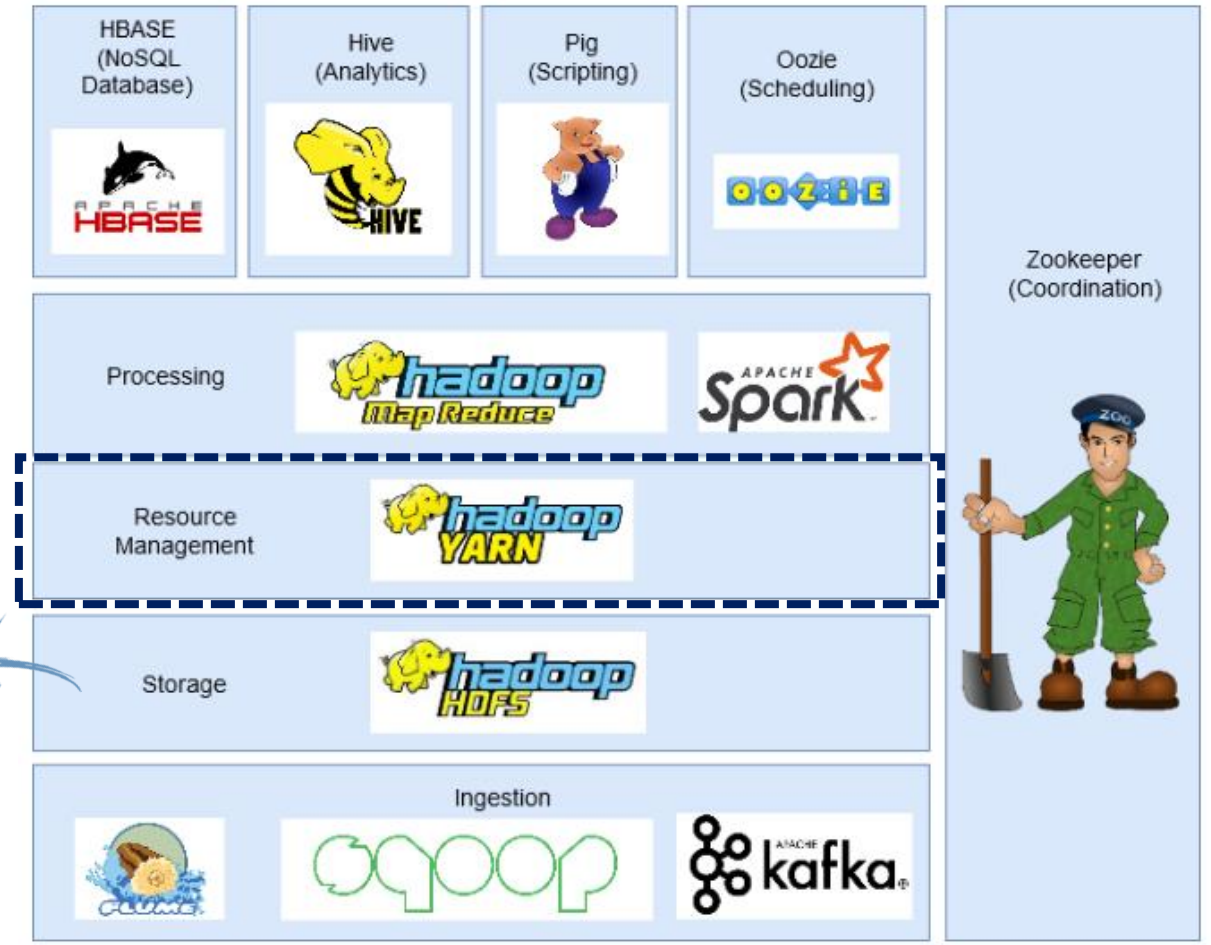
Etapas de um processo de Big Data

O **YARN** ou ***Yet Another Resource Negotiator*** gerencia os recursos no cluster e as aplicações no *Hadoop*. Em resumo, ele **coordena como as aplicações são executadas**.



Ecossistema *Hadoop*: Etapas Big Data

Etapas de um processo de Big Data

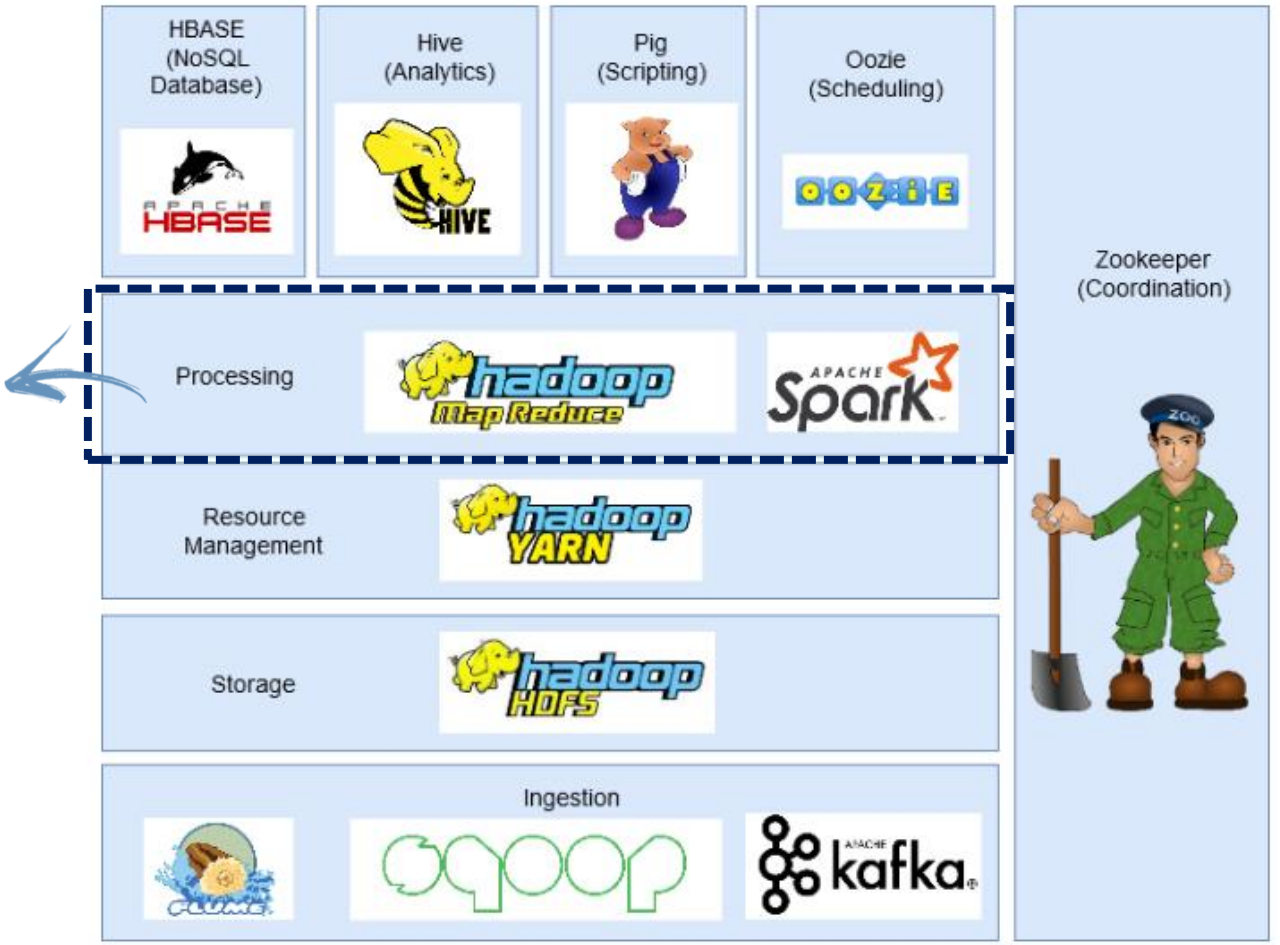


O HDFS é a unidade de armazenamento do *Hadoop*.

Ecossistema *Hadoop*: Etapas Big Data

Etapas de um processo de Big Data

O **MapReduce** e **Spark** são usados para processar os dados no HDFS e executar várias outras tarefas, principalmente **redução de dados**.

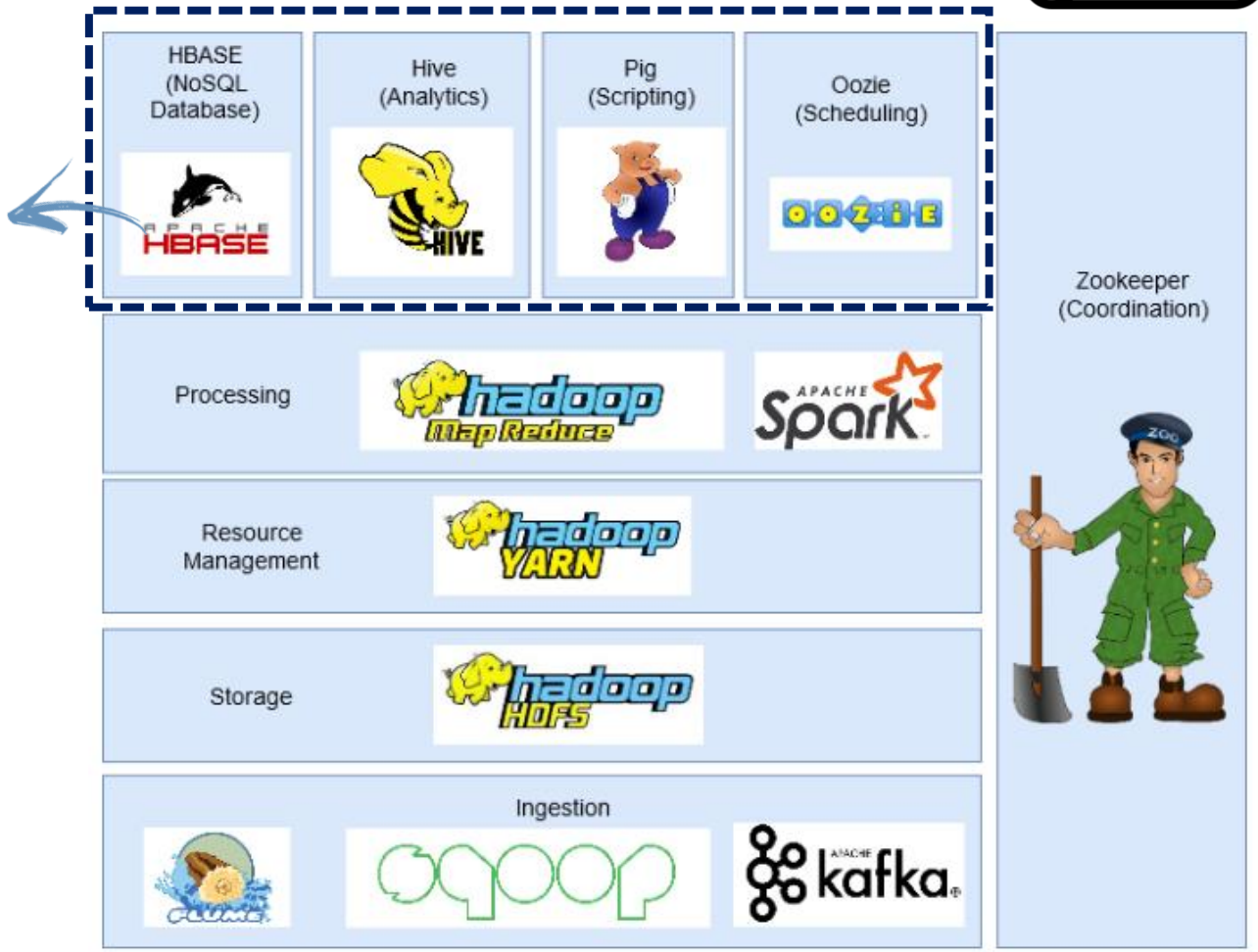


Ecossistema *Hadoop*: Etapas Big Data

Etapas de um processo de Big Data

Pig, **Hive** e **Spark** são utilizados para **analisar dados** processados pelo *MapReduce* e *Sparks*.

Oozie ajuda a agendar as tarefas.



ARA0168
TÓPICOS DE BIG DATA EM PYTHON

5.1 – Apache Sparks

Universidade Estácio de Sá

Prof. Simone Gama

simone.gama@estacio.br

Apache Sparks

O Apache Spark é um *framework* de código aberto de computação distribuída para uso geral, usado para aplicações de Big Data.

O *Spark* é muito mais eficiente do que o Hadoop para **gerenciar** e **processar** tarefas devido à utilização de cache de memória e algoritmo de processamento otimizado.

Apache Spark™ - Unified Engine for large-scale data analytics

Apache Sparks

Ele fornece APIs de desenvolvimento para as linguagens de programação Python, Java, Scala e R. Além disso, o Spark fornece recursos para o **desenvolvimento de aplicações** de aprendizado de máquina, SQL, análise gráfica e bibliotecas de *streaming*.

Apache Sparks

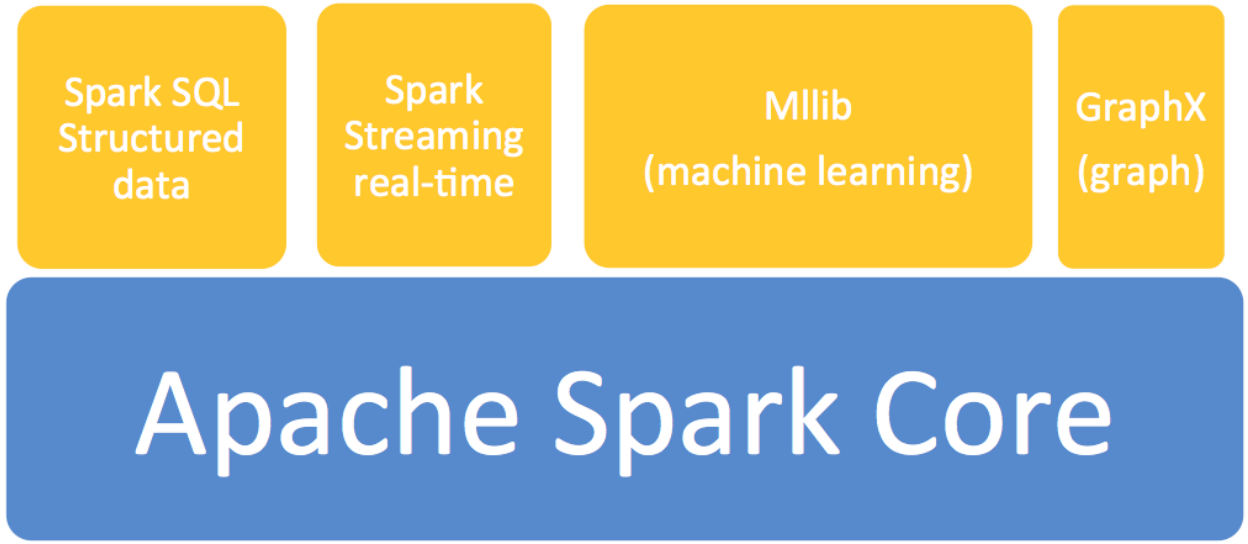
É uma **extensão** do modelo de programação já conhecido do Apache Hadoop – MapReduce – que facilita o desenvolvimento de aplicações de processamento de grandes volumes de dados.

Apache Sparks

Estude na Sala de Aula Virtual sobre as **aplicações** e **cenários** adequados e inadequados de uso do **Apache Spark**!

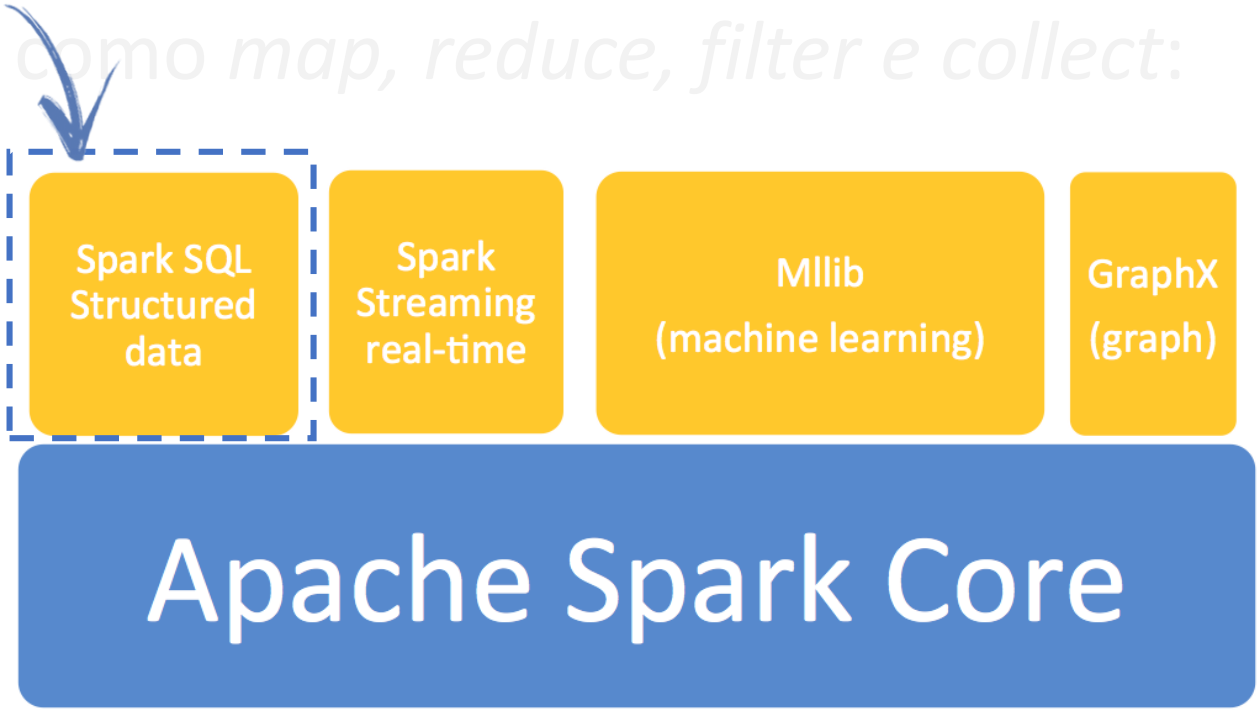
Apache Sparks: Componentes

O **Spark** possui vários componentes para diferentes tipos de processamento, todos construídos no **Spark Core**, que é o componente que oferece as funções básicas para as funções de processamento, como *map, reduce, filter e collect*:



Apache Sparks: Componentes

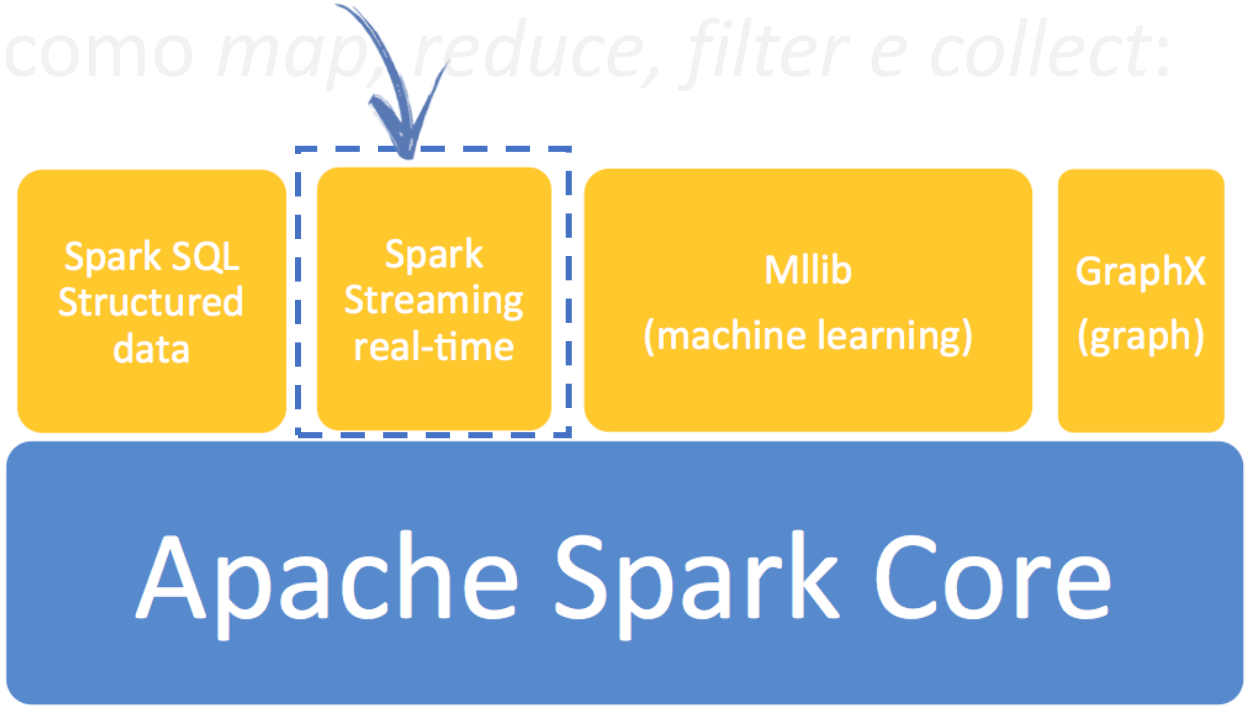
SparkSQL para usar SQL em *queries* e processar os dados no Spark.



Apache Sparks: Componentes

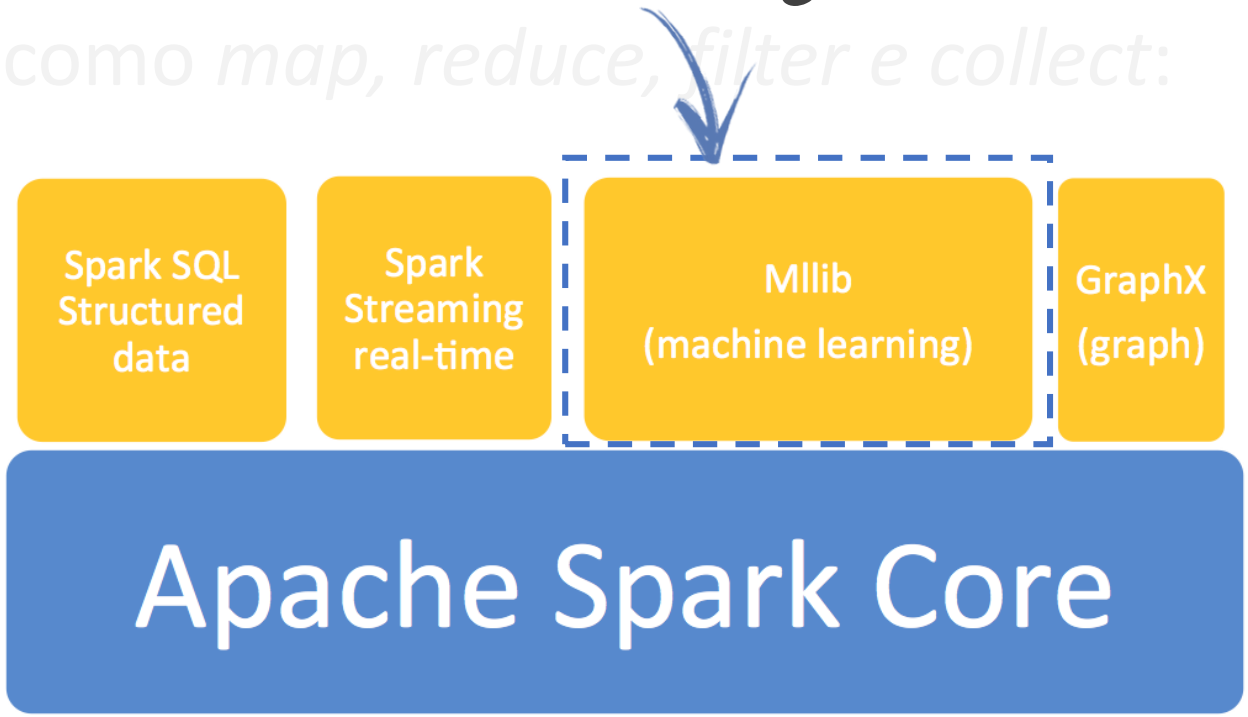
O Spark possui vários componentes para diferentes tipos de p Spark Core, que é o c sicas para as funções de processamento, como *map, reduce, filter e collect*:

Spark Streaming para processamento em **real-time**.

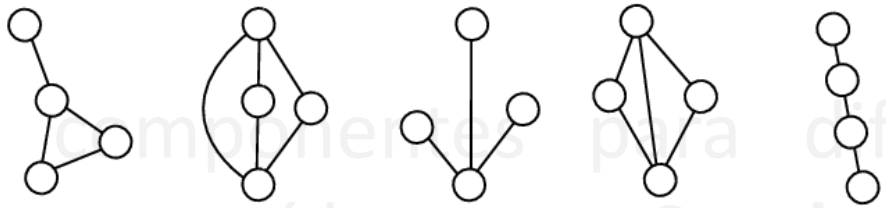


Apache Sparks: Componentes

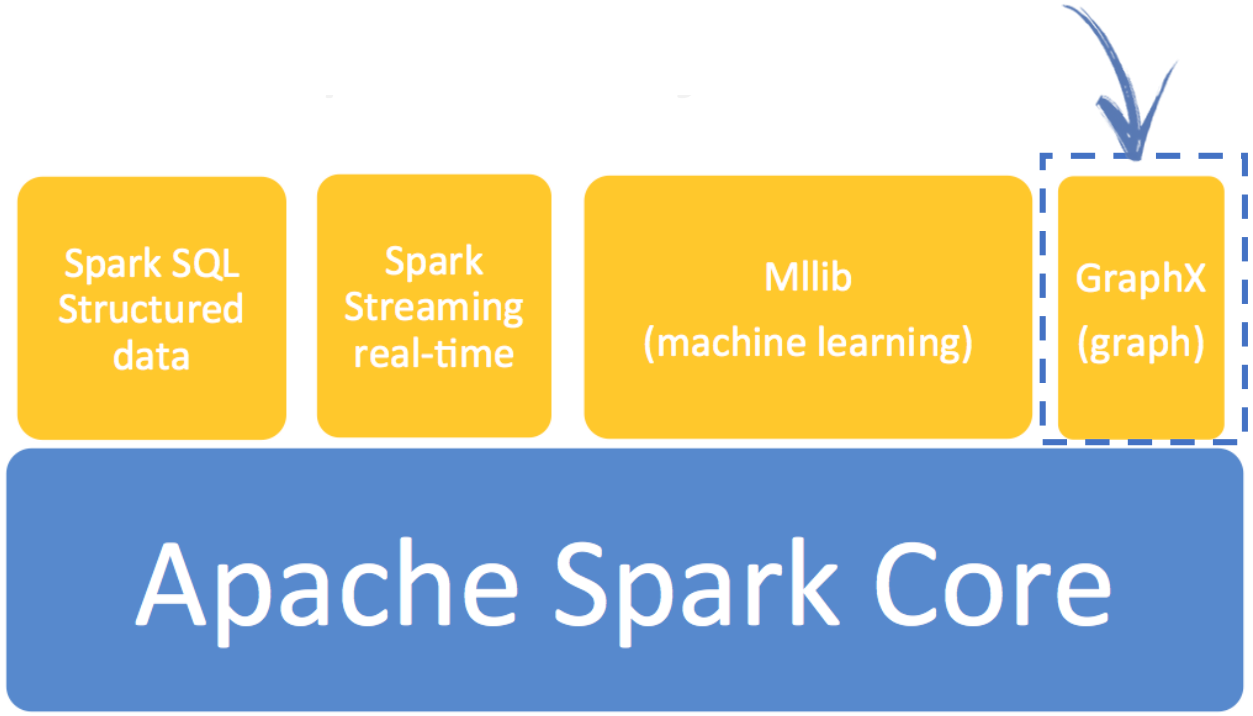
MLlib, que é uma biblioteca de *machine learning*, com diferentes algoritmos para várias atividades, como *clustering*.



Apache Sparks: Componentes

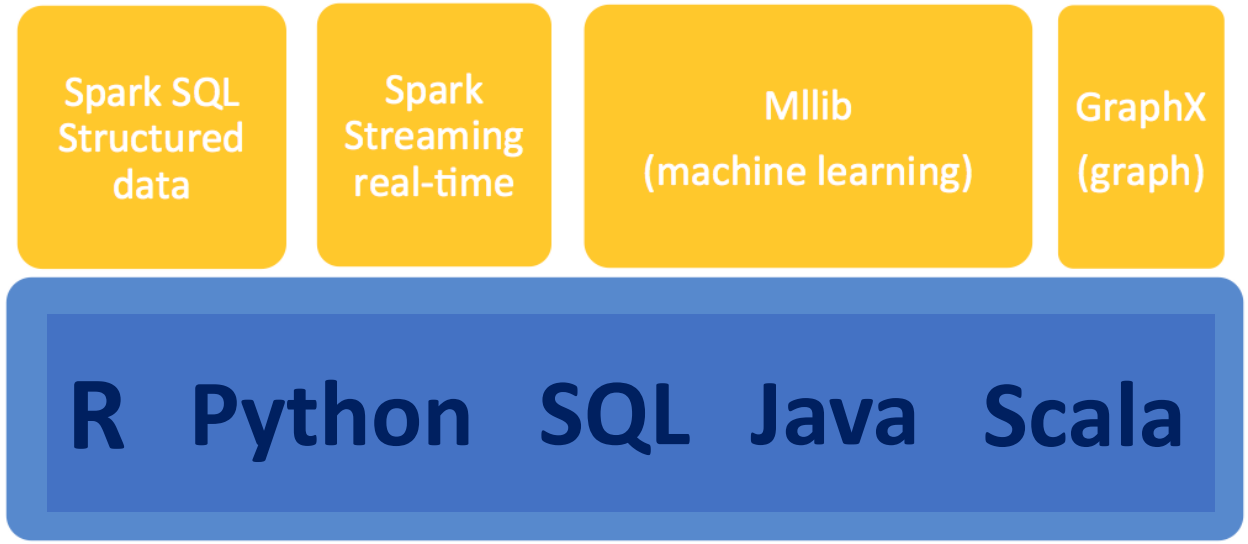


GraphX, que executa processamento sobre grafos.



Apache Sparks: Componentes

O **Spark** possui vários componentes para diferentes tipos de processamento, todos construídos no **Spark Core**, que é o componente que oferece as funções básicas para as funções de processamento, como *map, reduce, filter e collect*:

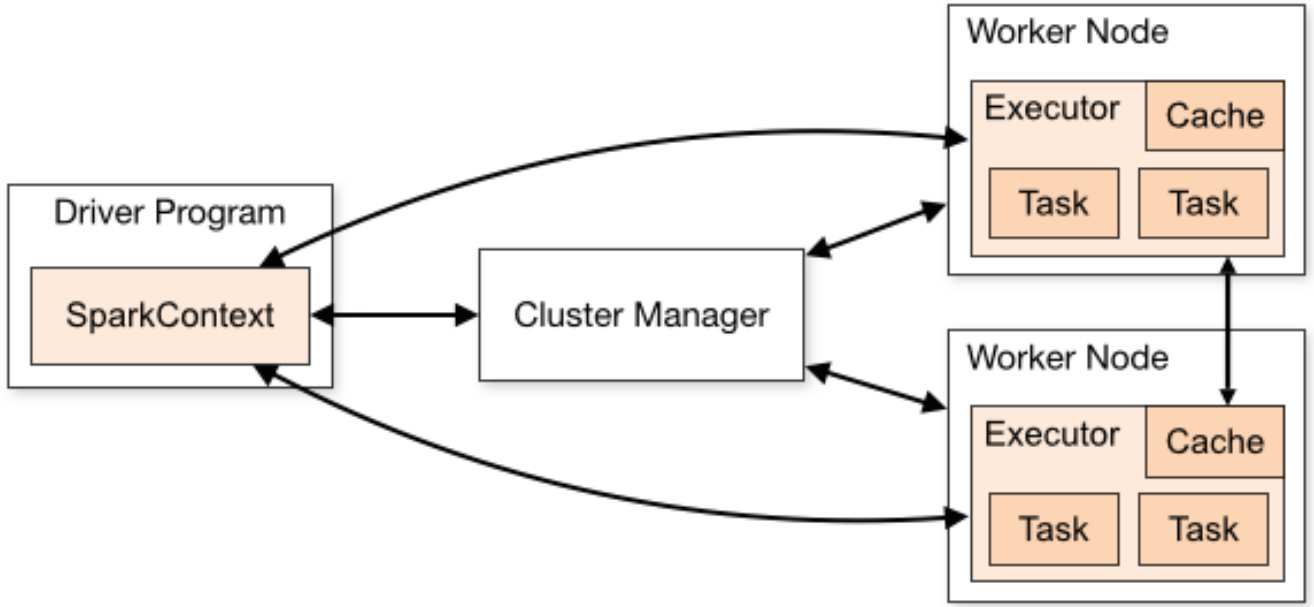


Apache Sparks: Spark's Architecture

A **arquitetura** das aplicações de **Spark** é constituída por três partes principais:

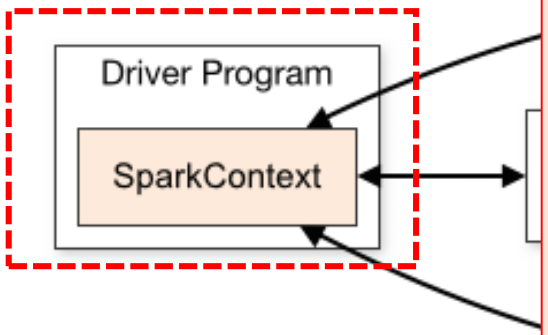
Apache Sparks: Spark's Architecture

A **arquitetura** das aplicações de **Spark** é constituída por três partes principais:



Apache Sparks: Spark's Architecture

A **arquitetura** das aplicações de **Spark** é constituída por três partes principais:

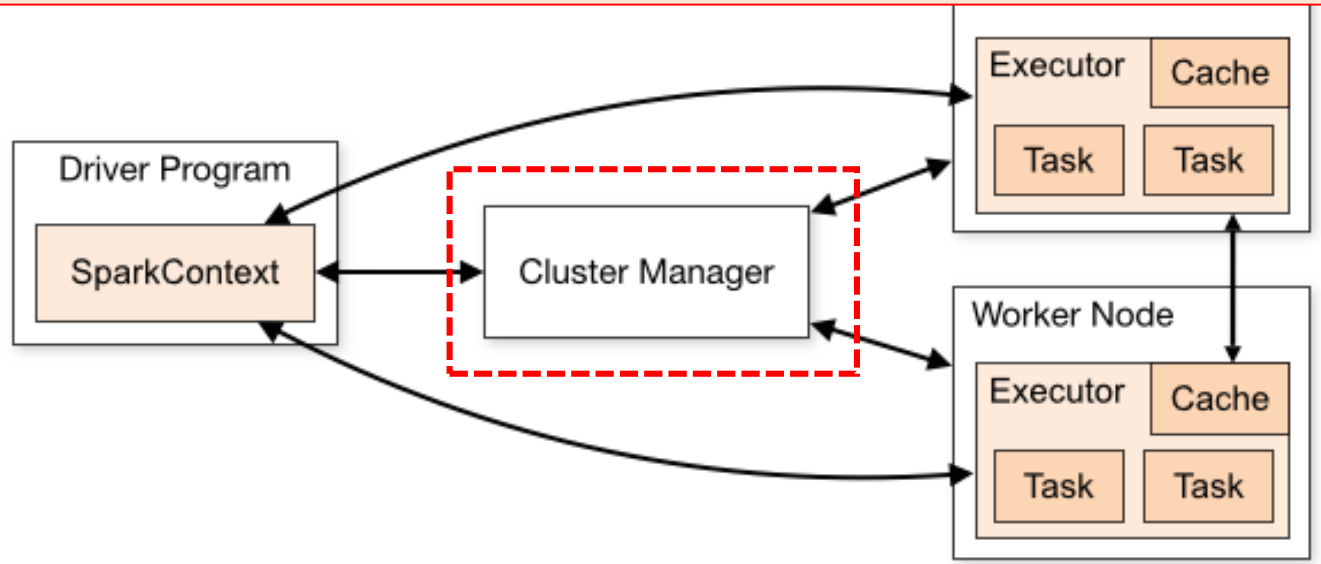


O *Driver Program* é a principal aplicação que **gera a criação** e aquele que **executa o processamento** definido pelos programadores.

O *Cluster Manager* é um componente opcional que é necessário apenas se o Spark for executado de forma distribuída.

É responsável por administrar as máquinas que serão usadas como *Workers.*

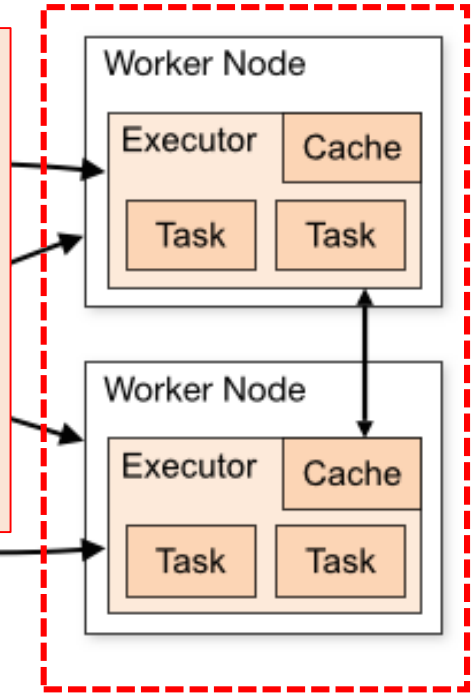
A a por três pa



Apache Sparks: Spark's Architecture

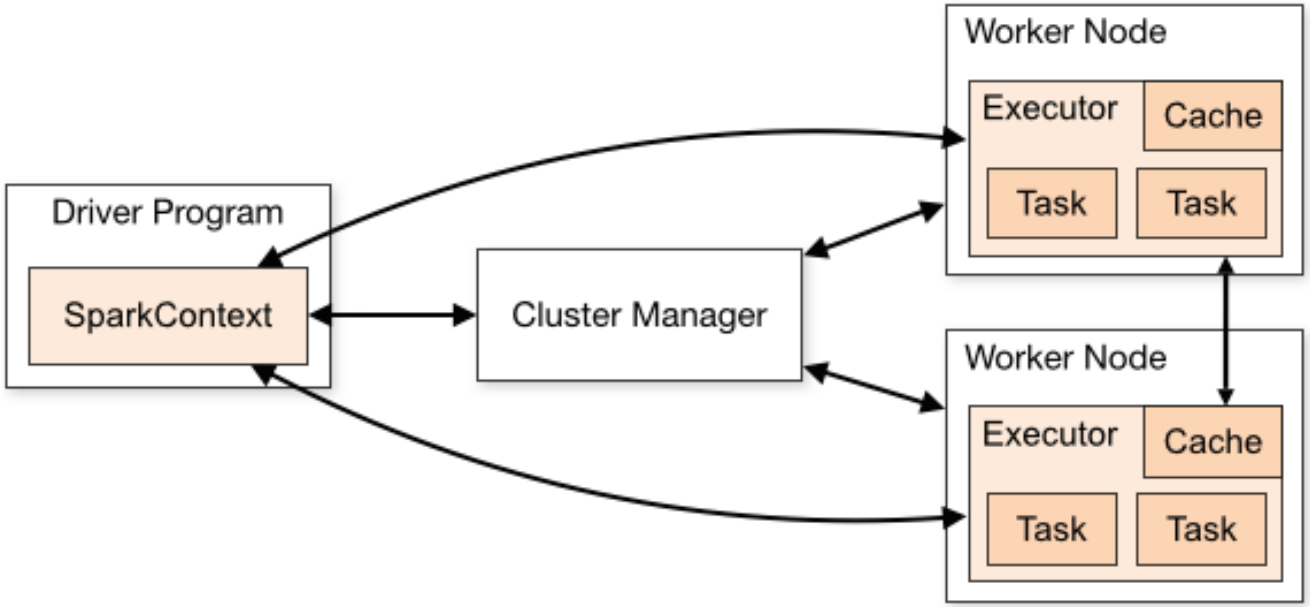
A **arquitetura** das aplicações de **Spark** é constituída por três partes principais:

Os *Workers* executam as tarefas enviadas pelo *Driver Program*. Se o Spark for executado em modo local, a máquina terá ambas as funções de *Driver Program* e *Worker*.



Apache Sparks: Spark's Architecture

A **arquitetura** das aplicações de **Spark** é constituída por três partes principais:



ARA0168
TÓPICOS DE BIG DATA EM PYTHON

5.1.1 – Interfaces Sparks

Universidade Estácio de Sá

Prof. Simone Gama

simone.gama@estacio.br

Apache Sparks: Interfaces do Sparks

Além da arquitetura, é importante conhecer os principais componentes do modelo de programação do Spark, ou também conhecidos como Interfaces do Spark.

Existem três conceitos fundamentais que são utilizados em todas as aplicações desenvolvidas:

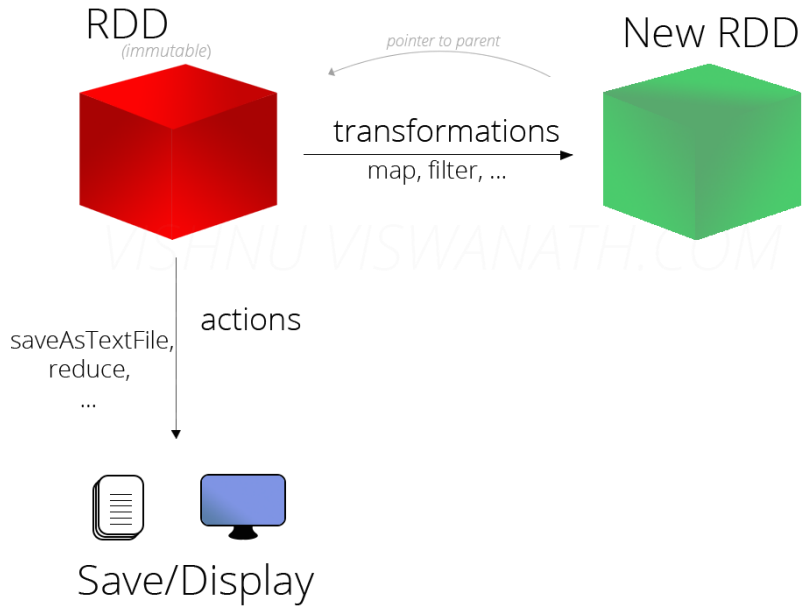
1. **Resilient Distributed Datasets (RDD).**
2. **DataFrame (ou Operações).**
3. **Dataset;**

Apache Sparks: Interfaces do Sparks

1.Resilient Distributed Datasets (RDD)

Abstraem um conjunto de objetos distribuídos no *cluster*, geralmente executados na memória principal. Estes podem estar armazenados em sistemas de arquivo tradicional, no **HDFS** e em alguns Banco de Dados NoSQL, como Cassandra.

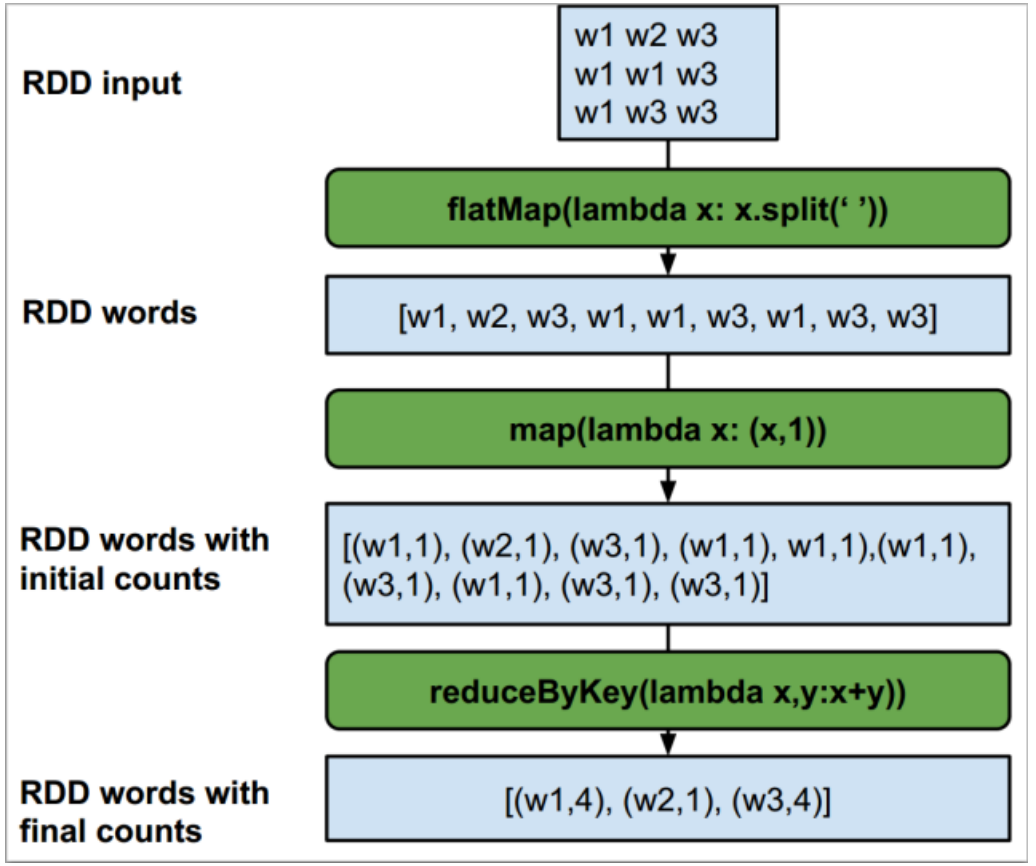
Ele é o objeto principal do modelo de programação do Spark, pois são nesses objetos que **serão executados os processamentos dos dados**.



Apache Sparks: Interfaces do Sparks

1.Resilient Distributed Datasets (RDD)

Exemplo simples de processamento de dados na interface RDD, de contagem de palavras em arquivos localizados no HDFS do Hadoop.

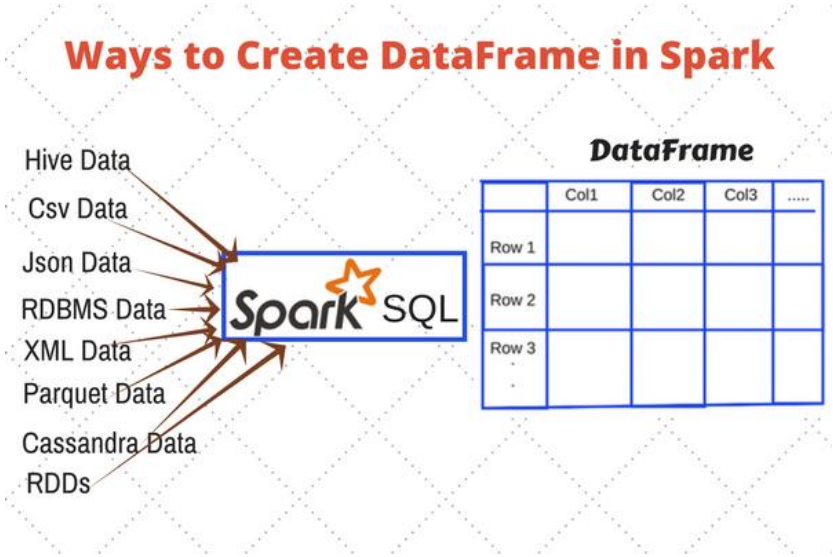


Apache Sparks: Interfaces do Sparks

2. DataFrame

Um *DataFrame* é uma estrutura de dados rotulada bidimensional com colunas de tipos potencialmente diferentes.
Considere um DataFrame como uma **planilha**, uma **tabela SQL** ou um **dicionário** de objetos de série.

A biblioteca Pandas é uma ferramenta amplamente usada nesse ambiente do Sparks.



Apache Sparks: Interfaces do Sparks

2. DataFrame

Exemplo de um *DataFrame* criado pela biblioteca pysparks no Python.

```
# import pyspark class Row from module sql
from pyspark.sql import *

# Create Example Data - Departments and Employees

# Create the Departments
department1 = Row(id='123456', name='Computer Science')
department2 = Row(id='789012', name='Mechanical Engineering')
department3 = Row(id='345678', name='Theater and Drama')
department4 = Row(id='901234', name='Indoor Recreation')

# Create the Employees
Employee = Row("firstName", "lastName", "email", "salary")
employee1 = Employee('michael', 'armbrust', 'no-reply@berkeley.edu', 100000)
employee2 = Employee('xiangrui', 'meng', 'no-reply@stanford.edu', 120000)
employee3 = Employee('matei', None, 'no-reply@waterloo.edu', 140000)
employee4 = Employee(None, 'wendell', 'no-reply@berkeley.edu', 160000)
employee5 = Employee('michael', 'jackson', 'no-reply@neverla.nd', 80000)

# Create the DepartmentWithEmployees instances from Departments and Employees
departmentWithEmployees1 = Row(department=department1, employees=[employee1, employee2])
departmentWithEmployees2 = Row(department=department2, employees=[employee3, employee4])
departmentWithEmployees3 = Row(department=department3, employees=[employee5, employee4])
departmentWithEmployees4 = Row(department=department4, employees=[employee2, employee3])

print(department1)
print(employee2)
print(departmentWithEmployees1.employees[0].email)
```

Apache Sparks: Interfaces do Sparks

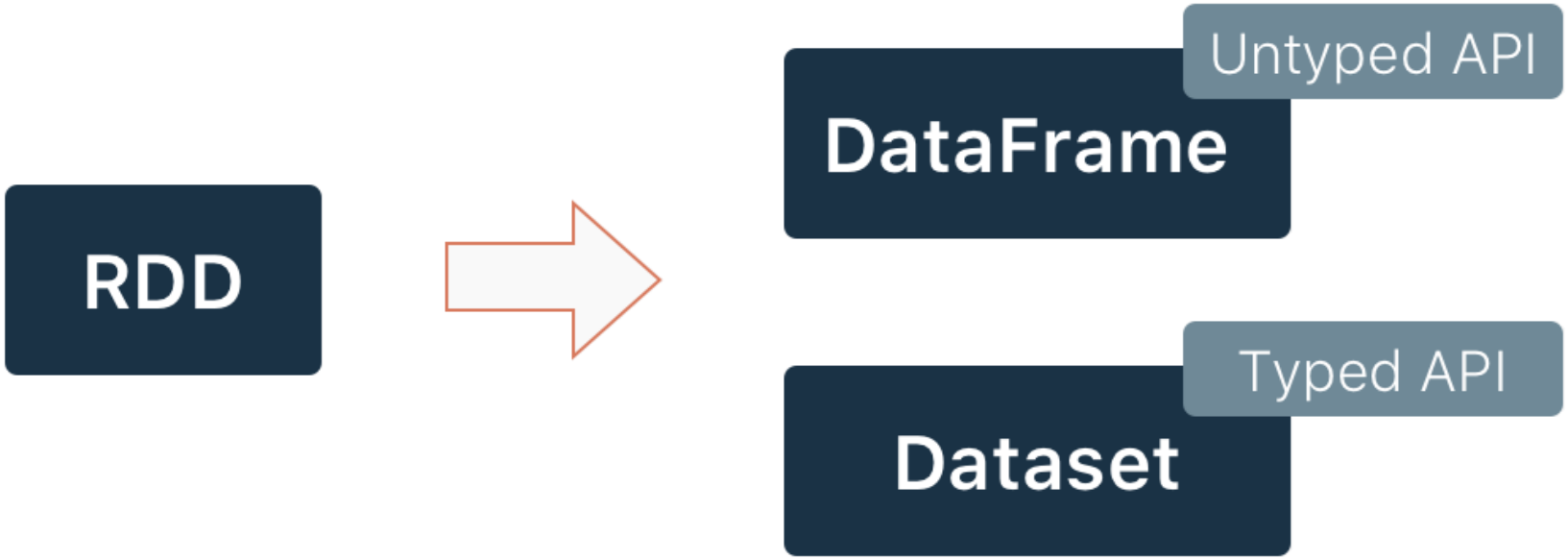
3. DataSet

É uma combinação de *DataFrame* e RDD. Ele fornece a interface tipada que está disponível nos RDDs, bem como a conveniência do *DataFrame*. A API do **Dataset** está disponível nas linguagens Java e Scala.

O programador, tendo um bom domínio de processamento de dados em *RDD* e *Dataframe*, dependo da regra de negócio, apenas o *DataSet* pode substituir as etapas anteriores.

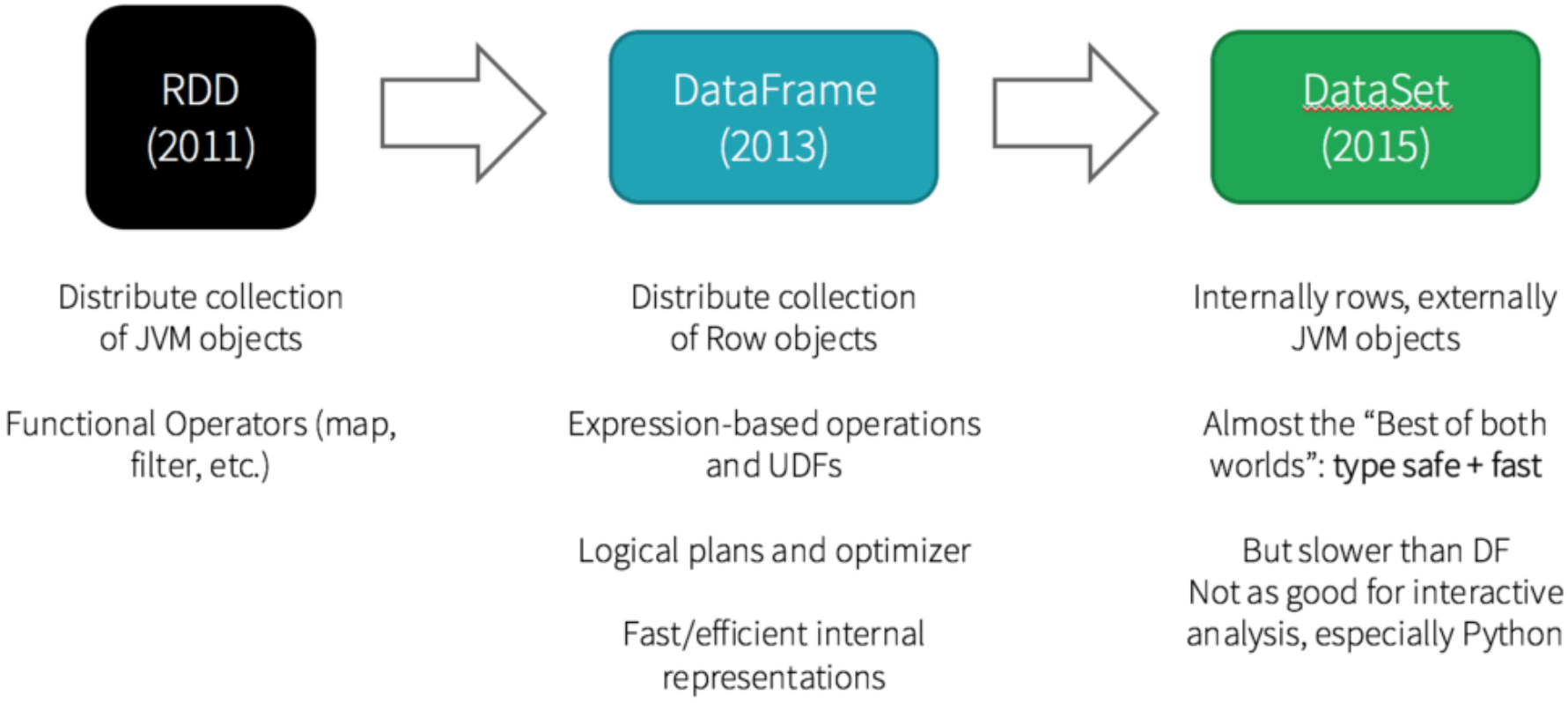
Apache Sparks: Interfaces do Sparks

Resumo do RDD x DataFrame x Dataset



Apache Sparks: Interfaces do Sparks

Histórico das API's Sparks



Apache Sparks: Interfaces do Sparks

Material (extremamente) importante sobre API's Sparks

[1] History and APIs Sparks. Link (Inglês): Resilient Distributed Dataset (RDD) – Databricks

[2] Apache Spark com Java. Introdução Básica. Link:

- Apache Spark: introdução (devmedia.com.br)
- Apache Spark Tutorial with Examples - Spark by {Examples} (sparkbyexamples.com)

ARA0168
TÓPICOS DE BIG DATA EM PYTHON

5.1.2 – Operações RDD

Universidade Estácio de Sá

Prof. Simone Gama

simone.gama@estacio.br

Apache Sparks: Operações RDD

Transformações RDD do Sparks

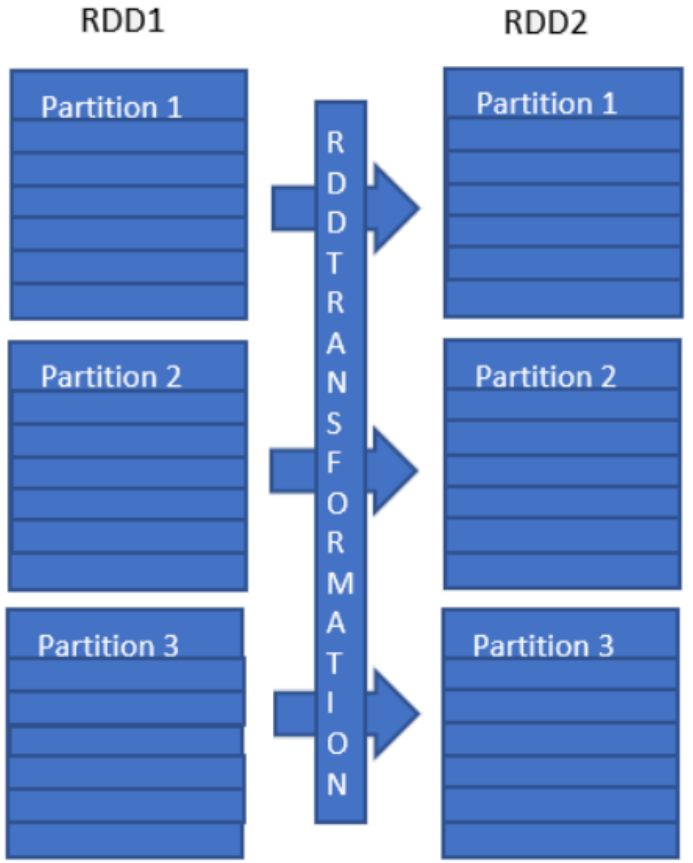
RDD Transformations are **Spark operations when executed on RDD**, it results in a single or multiple new RDD's.

Since RDD are immutable in nature, transformations always create new RDD without updating an existing one hence, this creates an **RDD lineage.**

Apache Sparks: Operações RDD

Transformações RDD do Sparks: Narrow Transformation

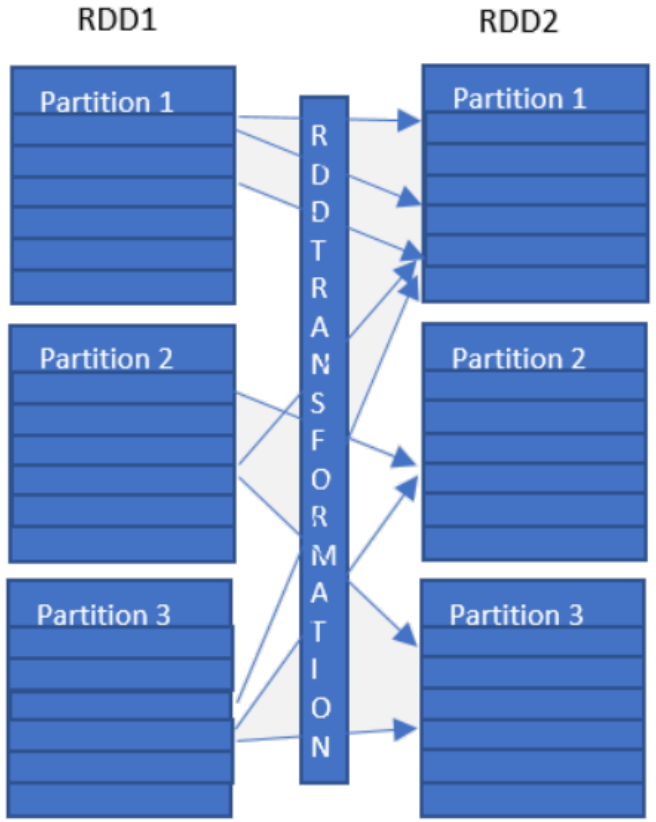
Narrow transformations are the result of `map()` and `filter()` functions and these compute data that live on a single partition meaning there will not be any data movement between partitions to execute narrow transformations.



Apache Sparks: Operações RDD

Transformações RDD do Sparks: Wider Transformation

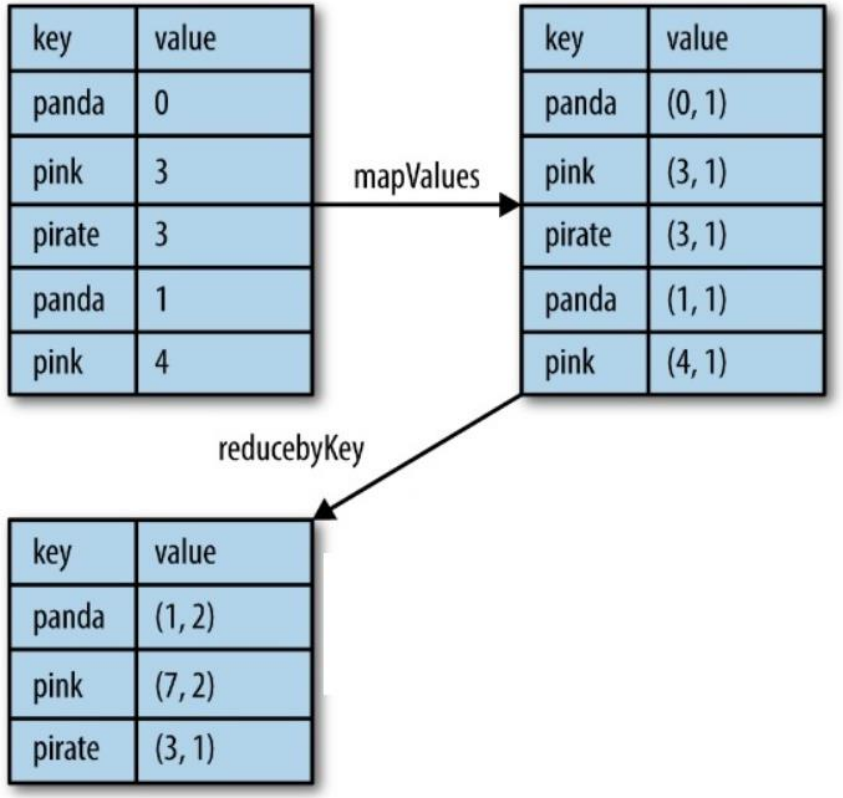
Wider transformations are the result of `groupByKey()` and `reduceByKey()` functions and these compute data that live on many partitions meaning there will be data movements between partitions to execute wider transformations. **Since these shuffles the data, they also called shuffle transformations.**



Apache Sparks: Operações RDD

Transformações RDD do Sparks: Wider Transformation

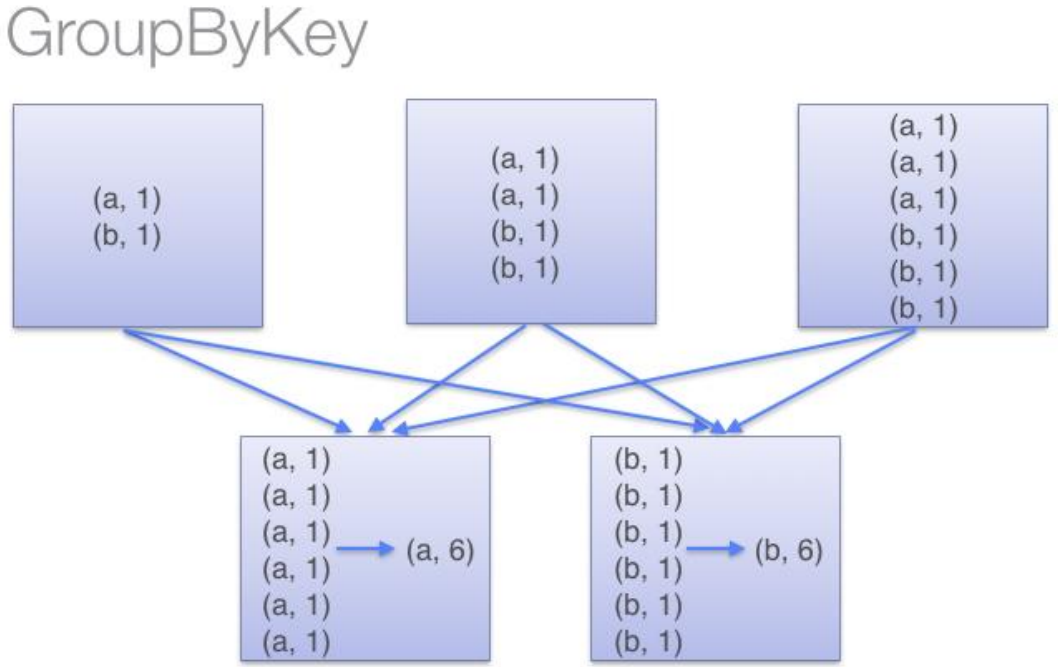
Exemplo 1 de **Transformação de grande escala:**



Apache Sparks: Operações RDD

Transformações RDD do Sparks: Wider Transformation

Exemplo 2 de **Transformação de grande escala:**



ARA0168
TÓPICOS DE BIG DATA EM PYTHON

5.2 – PySparks

Universidade Estácio de Sá

Prof. Simone Gama

simone.gama@estacio.br

PySparks: Definitions

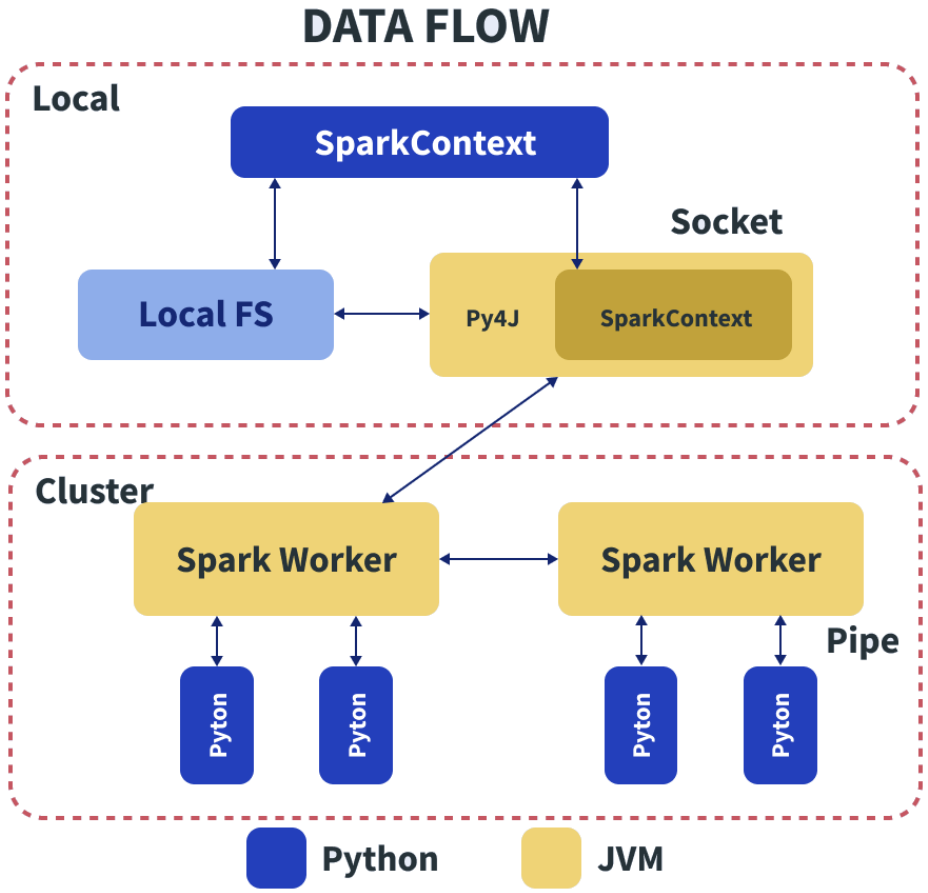
Muitos cientistas de dados e engenheiro de dados que utilizam o Apache Spark preferem se utilizar do **PySpark** para criar seus ***pipelines de dados***.

É uma interface bastante conveniente e sem dúvida um dos motivos da popularidade do Apache Spark, e que por muitas vezes parece "magia" como *spark* **escrito em Scala** que **roda na JVM** interage com **Python** transparentemente.

PySparks: Definitions

PySparks

O Pyspark é construído “em cima” da API Java, onde ele é apenas uma fina camada de software Python **que repassa as chamadas de funções** para o core Java.



PySparks: Definitions

PySparks - Performance

- Se a operação pode ser expressada com **chamadas de funções nativas** esta seria melhor opção, grande característica do Python.
- Caso seja necessário implementar funções customizadas então as funções definidas pelo usuários vetorizadas ou **pandas UDFs** é a opção mais performática[1] e estas podem ser de **3x até 100x** mais rápidas que as função definidas pelo usuário que não são vetorizadas em Python.
- Funções definidas pelo usuário não vetorizadas devem ser evitadas e utilizadas apenas como último recurso.

[1]Fonte: Introducing Pandas UDF for PySpark - The Databricks Blog

PySparks: Definitions

PySparks - Instalação

Pacotes de Instalação Pip

```
pip install pyspark
```

PySparks: Definitions

PySparks - Ferramentas

Como auxílio ao PySparks, existem ferramentas fundamentais em Python para auxiliar na produção de chamadas de função, dentre elas:

- Pandas
- Numpy
- Pydot (visualização)
- Scikit-learn (*Machine Learnig*)
- NLTK (Linguagem Natural)
- Regex

ARA0168
TÓPICOS DE BIG DATA EM PYTHON

5.2.1 – Python tools for function calls in PySpark

Universidade Estácio de Sá

Prof. Simone Gama

simone.gama@estacio.br

PySparks: Python tools for PySparks

Numpy

A **biblioteca NumPy** (*Numerical Python*) é uma biblioteca Python *open-source* bem popular e usada para aplicativos de computação científica e Big Data, que consiste em objetos de matriz multidimensionais e uma coleção de rotinas para processar essas matrizes.

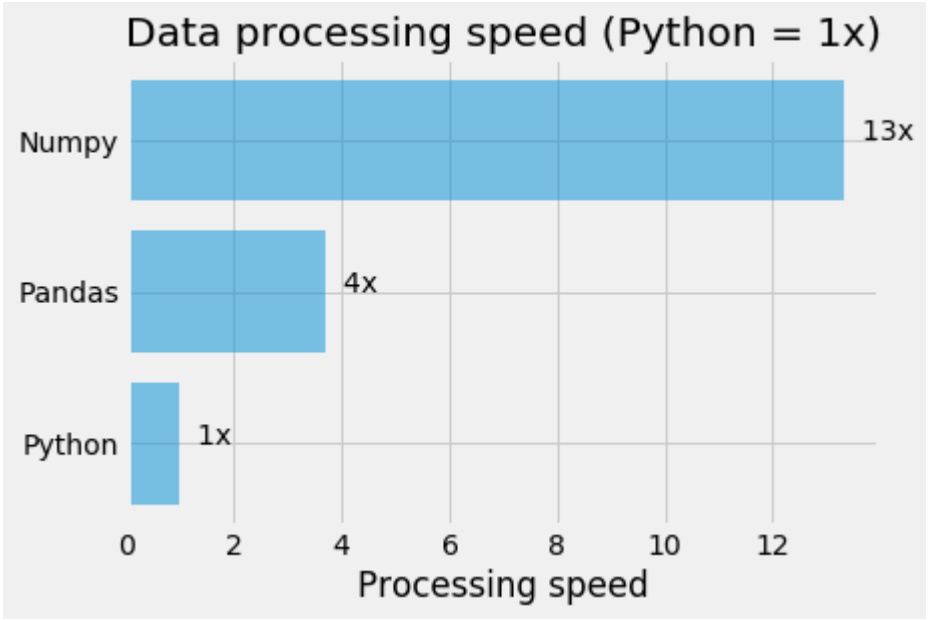
Documentação Numpy: numpy.ndarray.ndim — NumPy v1.23 Manual

PySparks: Python tools for PySparks

Numpy - Performance

NumPy foi escrito (em sua maior parte) **em linguagem C**, que é uma linguagem de baixo nível, o que torna a biblioteca **extremamente veloz**, e escondendo toda sua complexidade em um módulo Python simples de utilizar.

Outra diferença é que o NumPy **armazena os dados em memória**, ao contrário das listas em Python, de modo que as funções possam acessá-los e manipulá-los de maneira muito mais eficiente.



Fonte: Tutorial NumPy: os primeiros passos em computação numérica e tratamento de dados - ABRACD - ASSOCIAÇÃO BRASILEIRA DE CIÊNCIA DE DADOS

PySparks: Python tools for PySparks

Numpy - Instalação

Standard Python distribution doesn't come bundled with the NumPy module, hence you need to install NumPy using a popular Python package installer, **`pip`**.

```
# Install NumPy using PIP
pip install numpy
```

CONDA®

```
# Install NumPy using Conda
conda install numpy
```

PySparks: Python tools for PySparks

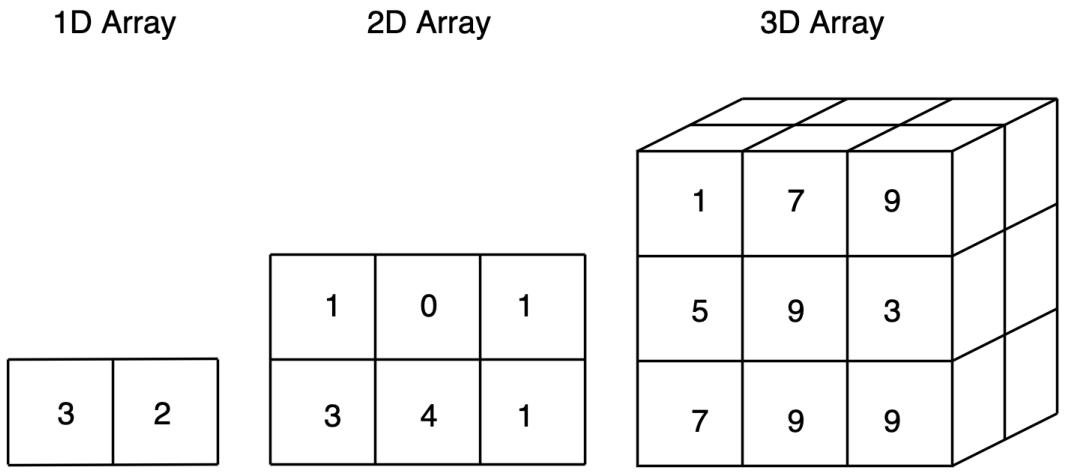
Numpy – Criando arrays com NumPy

Arrays são o tipo de objeto básico do NumPy. São como listas em Python, que contém valores armazenados em posições.

Valores →	**10**	**14**	**20**	**9**	**16**	**22**
Índices →	0	1	2	3	4	5

PySparks: Python tools for PySparks

Numpy – Criando arrays com NumPy



Um *array* também pode ser **multidimensional**, não se limitando apenas à 3D, dimensão máxima que o ser humano consegue compreender.

PySparks: Python tools for PySparks

Numpy – Criando arrays com NumPy

Example 1:

```
import numpy as np

# list in Python
data = [1, 2, 3, 4, 5, 6]

# list with parameter for NumPy array function
array = np.array(data)

print(type(array))
print(array)
```

Result:

```
<class 'numpy.ndarray'>
[1 2 3 4 5 6]
```

PySparks: Python tools for PySparks

Numpy – Criando arrays com NumPy

Example 2:

```
import numpy as np

# nested list
data = [[1, 2, 3], [4, 5, 6]]

# list 2D
array = np.array(data)

print(type(array))
print(array)
```

Result:

```
<class 'numpy.ndarray'>
[[1 2 3]
 [4 5 6]]
```

PySparks: Python tools for PySparks

Numpy – Atributos de arrays

Example 3:

```
import numpy as np

# nested list
data = [[1, 2, 3], [4, 5, 6]]

# list 2D
array = np.array(data)

print(array.ndim)
print(array.shape)
```

PySparks: Python tools for PySparks

Numpy – Atributos de arrays

Example 3:

```
import numpy as np

# nested list
data = [[1, 2, 3], [4, 5, 6]]

# list 2D
array = np.array(data)

print(array.ndim)
print(array.shape)
```

ndim fornece o número de dimensões do array.

shape nos diz quantos elementos temos em cada dimensão.

PySparks: Python tools for PySparks

Numpy – Atributos de arrays

Example 3:

```
import numpy as np

# nested list
data = [[1, 2, 3], [4, 5, 6]]

# list 2D
array = np.array(data)

print(array.ndim)
print(array.shape)
```

Result:

```
2
(2, 3)
```

PySparks: Python tools for PySparks

Numpy – Atributos de arrays: Geek

Example 4:

```
import numpy as geek

arr = [[1, 2, 3], [4, 5, 6]]

dimension = geek.ndim(arr)

print(dimension)
```

PySparks: Python tools for PySparks

Numpy – Gerando arrays: Geek

Example 5:

```
import numpy as np

print(np.zeros(10))

print(np.ones(10))

print(np.arange(20))
```

PySparks: Python tools for PySparks

Numpy – Gerando arrays: Geek

Example 5:

```
import numpy as np

print(np.zeros(10))

print(np.ones(10))

print(np.arange(20))
```

Result:

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
[ 0  1  2  3  4  5  6  7  8  9 10 11
12 13 14 15 16 17 18 19]
```

PySparks: Python tools for PySparks

Numpy – Gerando arrays

Example 6: Criando um array de zeros:

```
import numpy as np

zeros = np.zeros((3, 4))
print(zeros)
```

Result:

```
[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]
```

PySparks: Python tools for PySparks

Numpy – Gerando arrays

Example 7: Criando um array de nº aleatórios:

```
import numpy as np

random_nums = np.random.rand(2, 3)
print(random_nums)
```

Result:

```
[[0.99219663 0.98559045 0.23112187]
[0.54128177 0.43171111 0.37973207]]
```

Apache Sparks: Operações RDD

Simulando Operações de Transformações de Grande Escala RDD do Spark com Python

Questão 1: Sejam **três dataset's** em txt. Leia os três arquivos e gere um dicionário **datasetReduce** com a contagem **total** das palavras contidas nos 3 arquivos.

```
coração paixão amor paixão
alegria tristeza amor amor
coração alegria tristeza
alegria paixão
```

dataset1.txt

```
coração paixão amor paixão
alegria tristeza amor amor
coração alegria tristeza
alegria paixão
```

dataset2.txt

```
coração paixão amor paixão
alegria tristeza amor amor
coração alegria tristeza
alegria paixão
```

dataset3.txt

```
{'coração': 6, 'paixão': 9, 'amor':9, 'alegria':9, 'tristeza':6}
```

datasetReduce

Dúvidas?

Bibliografia

- FACELI, Katti. **Inteligência Artificial Uma abordagem de aprendizado de máquina.** 2ª Ed. Rio de Janeiro: LTC, 2021. Disponível em:

Bibliografia Auxiliar

- *Kubernetes Google*: How Kubernetes came to be: A co-founder shares the story | Google Cloud Blog
- Etapas Ecossistema Hadoop: Hadoop Para Leigos - Os 11 Componentes Do Ecossistema - Analyticsbr
- Apache Sparks:
  - Spark RDD Transformations with examples - Spark by {Examples} (sparkbyexamples.com)
  - 2. Why Spark with Python ? — Learning Apache Spark with Python documentation (runawayhorse001.github.io)