

Introdução ao Spark

Advanced Institute for Artificial Intelligence – AI2

<https://advancedinstitute.ai>



Background

O importante é não parar de questionar.

“

Aubert Einstein

Referências

- ❑ [Learning Apache Spark with Python \(Link\)](#) ou
- ❑ [Learning Apache Spark with Python \(.pdf\)](#)
- ❑ [Spark with Python \(PySpark\) Tutorial](#)

Apresentação



Apresentação

□ **História do Spark:**

- *Google File System (GFS), MapReduce (MR) e Bigtable*

□ **Resultou em um sistema complexo, composto por:**

- **Hadoop Distributed File System (HDFS):** É a camada de armazenamento do Hadoop.
- **Map-Reduce:** É a camada de processamento de dados do Hadoop.
- **YARN:** É a camada de gerenciamento de recursos do Hadoop

□ **Solução:**

- Hive criado pelo facebook (DW SQL sobre HDFS)

Apresentação

□ O que é PySpark?

- **PySpark** Ferramenta de Processamento de Dados (Não é *Data Storage*)
- Muito mais veloz que qualquer ferramenta de processamento de dados
- *Spark* tem sua arquitetura voltada a processar dados!

Apresentação

□ O que é PySpark?

- **PySpark** é uma biblioteca Spark escrita em Python para executar aplicativos Python usando recursos do Apache Spark.
- Com essa ferramenta, podemos executar aplicativos paralelamente num *cluster* distribuído. (Entendam *cluster* como um conjunto de servidores e computadores, já os nós seriam cada computador individualmente.)
- Em outras palavras, **PySpark** é uma *API* para *Apache Spark*, que é um mecanismo de processamento analítico para aplicativos poderosos de processamento de dados distribuídos e aprendizado de máquina em grande escala.

Apresentação

- **Spark possui suporte para várias linguagens:**
 - Muitas bibliotecas votadas ao processamento de dados

Scala 

Python 

Java 

R 

SQL 

Apresentação

□ O que é PySpark?

- Basicamente, **PySpark** foi escrito em **Scala** e, posteriormente, devido algumas adaptações voltadas para a indústria, a *API* **PySpark** foi lançada para *Python* usando **Py4J**.
- **Py4J** é uma biblioteca em *Java* integrada ao **PySpark** que permite que o *python* faça interface dinâmica com objetos *Java Virtual Machine - JVM*.
- Para executarmos o **PySpark** então, é preciso também instalar o *Java* junto com o *Python* e o *Apache Spark*.

Por que usar PySpark?

□ Vamos pensar...

- É do nosso conhecimento que a geração de informações de diversas fontes como, celular, rede social, e-mail e etc, estão aumentando cada dia mais;
- Com essa elevação constante, as aplicações em *Data Science* e a construção de modelos de *Machine Learning* de forma eficiente e rápida começaram a virar um grande desafio;
- As ferramentas que estamos acostumados a usar em nossos projetos do dia a dia (como *Pandas* por exemplo), começam a apresentar perda de performance e até mesmo, em alguns casos, não dar mais conta do recado.

Por que usar PySpark?

□ Vamos pensar...

- Surge então a necessidade de trabalhar com ferramentas capazes manipular projetos contendo uma carga constante e volumosa de dados, ou seja, que nos permitam extrair informação de conjunto de dados volumosos sem perdermos performance e, de preferência, sem elevar os custos do projeto
- Para isso temos então as soluções de *Big Data*, como o *Apache Spark*, que iremos começar a trabalhar.
- **PySpark** é amplamente utilizado nas comunidades de *Data Science* e *Machine Learning*, pois existe uma infinidade de bibliotecas usadas e escritas em *Python*, incluindo *NumPy* e *TensorFlow*.

Apresentação

- **Spark é uma ferramenta que opera em Cluster**
 - Computadores em rede com funções distintas
- Rede de Computadores

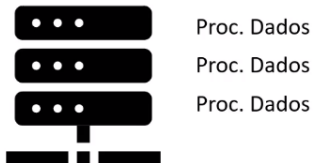


Apresentação

□ Spark é uma ferramenta que opera em Cluster

- Rede de computadores com as mesmas funções.

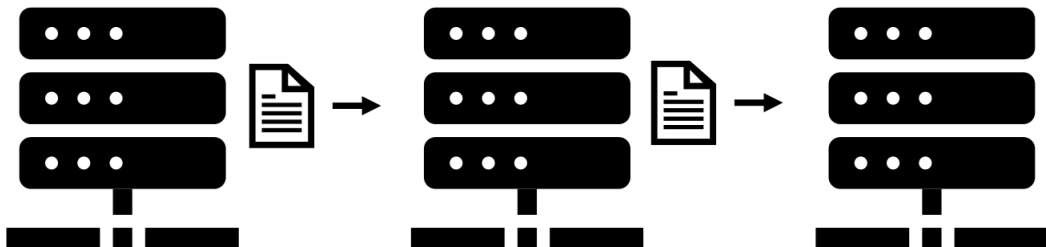
- Rede de Computadores - Cluster



Apresentação

□ Replicação

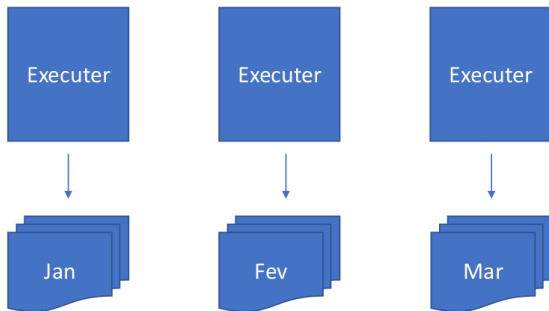
- Dados são copiados entre os nós do cluster. Isso traz o benefício de, entre outras coisas, tolerância a falhas.



Apresentação

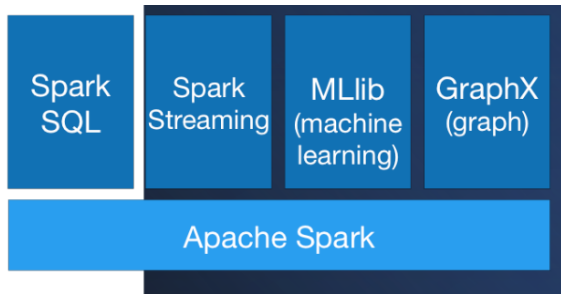
❑ Particionamento e trabalho em paralelo

- Como o custo computacional é muito grande quando aplicado a grandes quantidade de dados, realizar o **particionamento** tornou-se uma excelente saída.



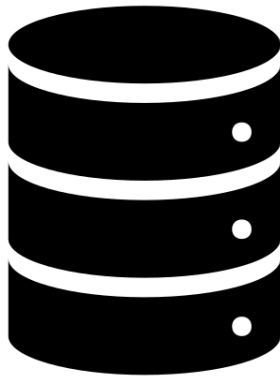
□ Componentes

- Machine learning
- SQL (Spark SQL)
- Processamento em Streaming
- Processamento em grafos (GraphX)



□ Spark SQL

- Permite ler dados tabulares de várias fontes como CSV, Json, Parquet, ORC e etc.
- Pode usar sintaxe SQL



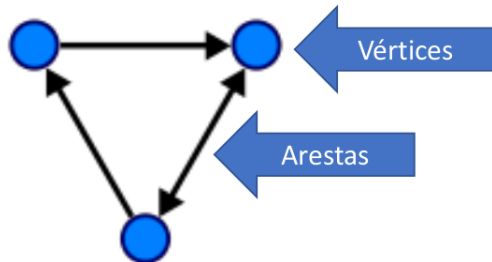
□ **Spark Structured Streaming**

- Dados estruturados;
- Novos registros adicionados ao final da tabela.



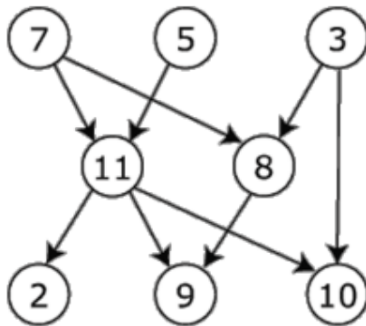
□ Grafos acíclicos dirigidos

- Spark Constrói Gráficos Acíclicos Dirigidos.
- Novos registros adicionados ao final da tabela.



□ Acíclicos Dirigidos

- Acíclicos: não tem ciclo
- Dirigidos: tem uma direção



Estrutura

□ Driver:

- Inicializa SparkSession, solicita recursos computacionais do Cluster Manager, transforma as operações em DAGs, distribui estas pelos executors

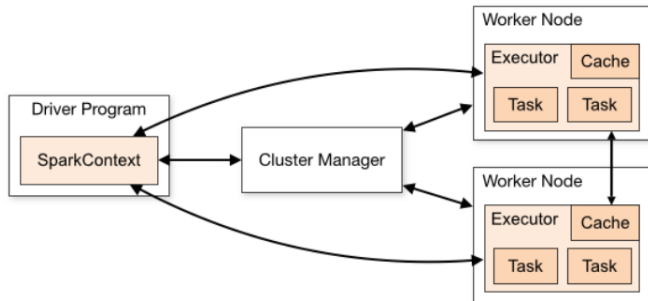
□ Manager:

- Gerencia os recursos do cluster. Quatro possíveis: built-in standalone (padrão), YARN, Mesos e Kubernetes

□ Executor:

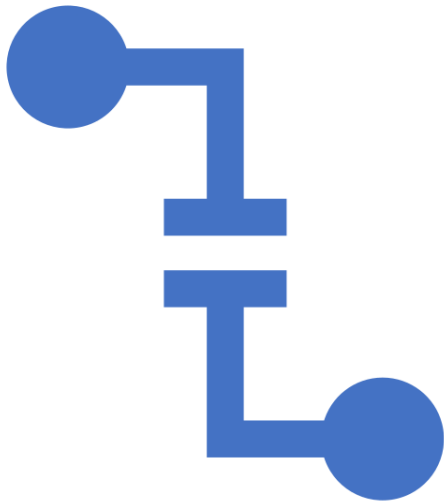
- Roda em cada nó do cluster executando as tarefas

Estrutura



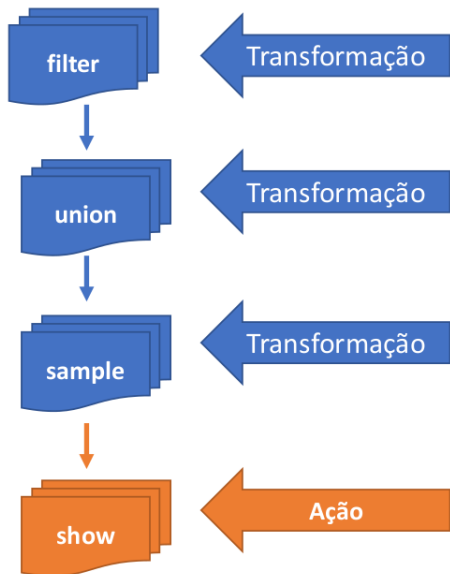
□ Transformações e Ações

- Um data frame é imutável: traz tolerância a falha
- Uma transformação gera um novo data frame.
- O processamento de transformação de fato só ocorre quando há uma Ação: *Lazy Evaluation*



□ Lazy Evaluation

- Na prática, para cada etapa dessas realizadas, o Spark não faz nada, apenas quando solicita para mostrar os dados (uma ação), então nesse momento o Spark realiza cada um dos processos
- É dessa forma que o Spark trabalha para otimização dos processos, através de planos de otimização das tarefas a serem realizadas.



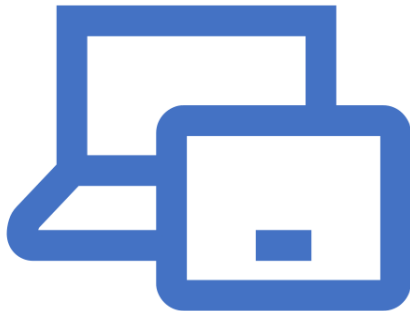
□ Transformações e Ações

- Essa tabela apresenta os processos de transformações e ações que o Spark utiliza durante seu processo.

Transformações	Ações
map	
filter	reduce
flatMap	collect
mapPartitions	
mapPartitionsWithIndex	count
sample	first
union	
intersection	take
distinct	
groupByKey	takeSample
reduceByKey	takeOrdered
aggregateByKey	
sortByKey	saveAsTextFile
join	saveAsSequenceFile
cogroup	
cartesian	saveAsObjectFile
pipe	
coalesce	countByKey
repartition	
repartitionAndSortWithinPartitions	foreach

□ Transformações Narrow e Wide

- **Narrow:** Os dados necessários estão em uma mesma partição
- **Wide:** Os dados necessários estão em mais de uma partição

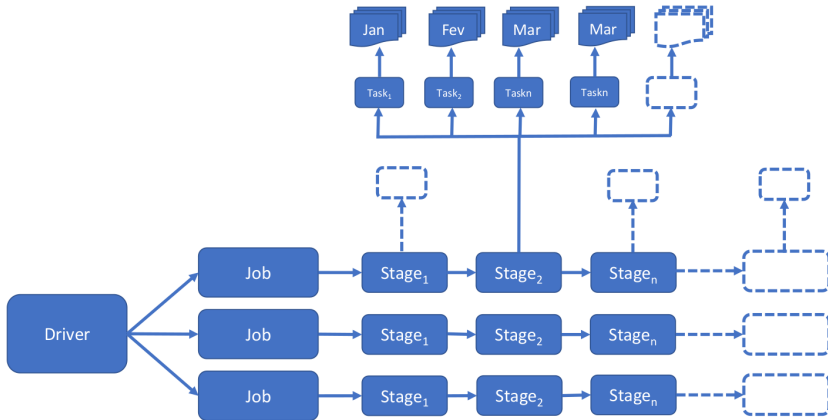


□ Componentes

- **Job:** Tarefa
- **Stage:** Divisão do Job
- **Task:** Menor unidade de trabalho. Uma por núcleo e por partição

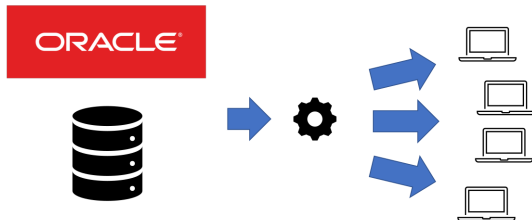


Diagrama Simplificado do Processo



Formato dos dados (Big Data)

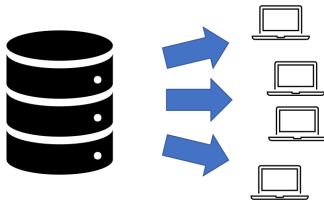
- ❑ **Quais os tipos de dados que mais utilizamos em nossas análises?**
 - Jason, CSV, TXT, XML ...
 - Esses tipos de dados ocupam muito espaço, pois não são performáticos nem auxiliam nos processos de busca e consultas.
- ❑ Oracle: Com seus tipos de dados em formato Proprietário



Formato dos dados (Big Data)

□ Armazenamento de dados modernos

- Possuem formatos abertos
- Dados são armazenados para que qualquer ferramenta possa acessar os dados diretamente, sem a necessidade de qualquer ferramenta que gerencie esses dados.



□ Principais Formatos para Big Data

- **Parquet:** Apache Parquet é um formato de armazenamento colunar disponível em todos os projetos que pertencem ao ecossistema Hadoop, independente do modelo de processamento, framework ou linguagem usada.



□ Principais Formatos para Big Data

- **Avro:** O Avro é um sistema de serialização de dados de software livre que ajuda na troca de dados entre sistemas, linguagens de programação e estruturas de processamento.
- Ajuda a definir um formato binário para seus dados, bem como mapeá-lo para a linguagem de programação escolhida.



❑ Principais Formatos para Big Data

- **Apache ORC:** O Apache ORC

(*Optimized Row Columnar*) é um formato de armazenamento de dados orientado a colunas gratuito e de código aberto do ecossistema Apache Hadoop;

- Semelhante a outros formatos de arquivo de armazenamento em colunas disponíveis no ecossistema Hadoop, é compatível com a maioria das estruturas de processamento de dados no ambiente Hadoop.



Principais Formatos para Big Data

☐ Formatos

- Armazéns de dados modernos tendem a armazenar dados em formatos "desacoplados" de ferramentas e abertos;
- Formatos binários, compactados;
- Suportam Schema;

☐ Podem ser particionados entre discos (ou clusteres):

- Redundância, ou seja, cópia dos dados espalhados pelos *clusteres*.
- Paralelismo, ou seja, executar diferentes processos em paralelo sobre o mesmo conjunto de dados.

Principais Formatos para Big Data

□ Mas... qual escolher???

- Em geral **ORC** é mais eficiente na criação (escrita) e na compressão
- **Parquet** tem melhor performance na consulta (leitura)
- O ideal é fazer um *benchmark*, ou seja, comparar de forma eficiente a performance entre as técnicas utilizadas.

Preparando o ambiente

- Abaixo temos alguns artigos que apresentam um passo a passo da instalação do Spark em **Windows** e **Ubuntu**
- **Windows**
 - [PySpark Windows](#)
- **Ubuntu**
 - [PySpark Linux](#)

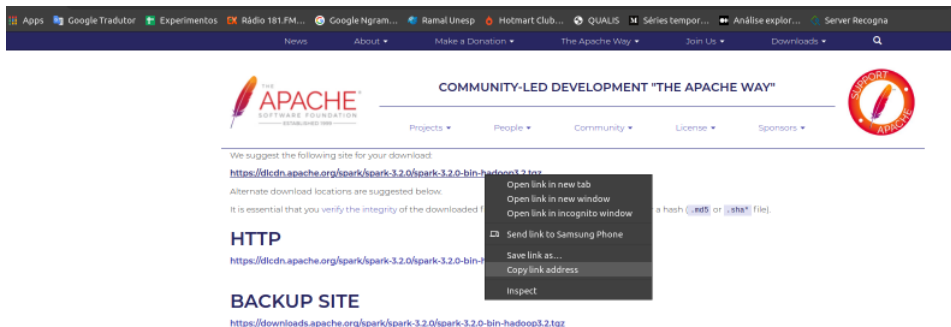
Preparando o ambiente

```
1  # buscando atualizacoes
2  sudo apt update
3
4  # atualizando
5  sudo -y upgrade
6
7  # instalando o Java
8  sudo apt install curl mlocate default-jdk -y
```

Preparando o ambiente

❑ Instalando Spark

- Spark atualizado
- link para o apache



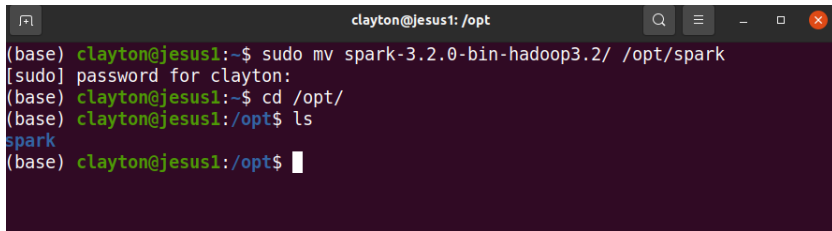
Preparando o ambiente

❑ Instalando Spark

```
clayton@jesus1: ~  
(base) clayton@jesus1:~$ wget https://dlcdn.apache.org/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz  
--2021-12-22 10:55:37-- https://dlcdn.apache.org/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz  
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644  
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 300965906 (287M) [application/x-gzip]  
Saving to: 'spark-3.2.0-bin-hadoop3.2.tgz'  
  
spark-3.2.0-bin-had 100%[=====>] 287,02M  91,2MB/s   in 3,1s  
  
2021-12-22 10:55:40 (91,2 MB/s) - 'spark-3.2.0-bin-hadoop3.2.tgz' saved [300965906/300965906]  
  
(base) clayton@jesus1:~$
```

Preparando o ambiente

- O passo seguinte é extrair os arquivos do download com o comando `tar -zxvf`
- Poderíamos deixar a pasta extraída aqui mesmo porém, motivados pela boa prática, vamos movê-la para o diretório `/opt`



```
clayton@jesus1: /opt
(base) clayton@jesus1:~$ sudo mv spark-3.2.0-bin-hadoop3.2/ /opt/spark
[sudo] password for clayton:
(base) clayton@jesus1:~$ cd /opt/
(base) clayton@jesus1:/opt$ ls
spark
(base) clayton@jesus1:/opt$
```


Definindo a variável de ambiente

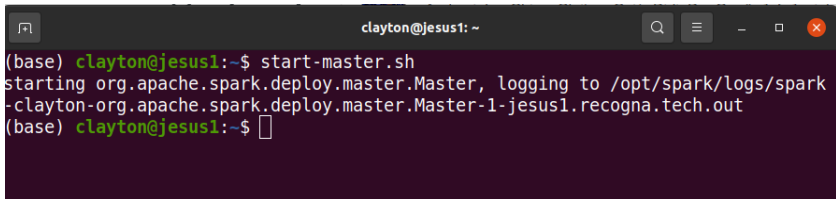
□ Definindo a variável de ambiente

- No diretório raiz, abrimos um editor de textos e editamos o arquivo através do comando:

```
1      sudo nano ~/.bashrc
2
3      # editando o file:
4      export SPARK_HOME=/opt/spark
5      export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
6
7      # atualizando o bash
8      source ~/.bachrc
```

Preparando o ambiente

- Iniciando Spark no modo Standalone:

A terminal window with a dark background and light text. The window title is 'clayton@jesus1: ~'. The prompt is '(base) clayton@jesus1:~\$'. The user has entered 'start-master.sh'. The output shows the Spark master starting, logging to '/opt/spark/logs/spark-clayton-org.apache.spark.deploy.master.Master-1-jesus1.recogna.tech.out'. The prompt returns to '(base) clayton@jesus1:~\$' with a cursor.

```
(base) clayton@jesus1:~$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark
-clayton-org.apache.spark.deploy.master.Master-1-jesus1.recogna.tech.out
(base) clayton@jesus1:~$
```

- Agora que temos nosso ambiente inicialmente configurado, podemos testá-lo através do comando:
 - **Digitamos no terminal:** localhost:8080

← → localhost:8080

Apps Google Tradutor Experimentos Rádio 181.FM... Google Ngram... Ramal Unesp Hotmart Club... QUALIS M Séries

Spark Master at spark://clayton-note:7077

URL: spark://clayton-note:7077
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

▼ Workers (0)

Worker Id	Address	State	Cores
-----------	---------	-------	-------

▼ Running Applications (0)

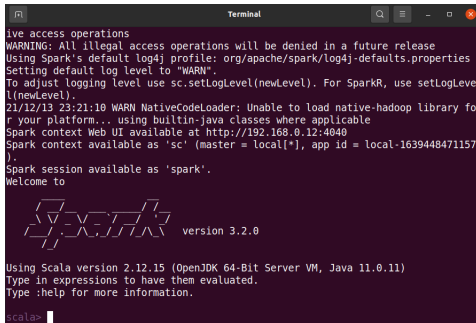
Application ID	Name	Cores	Memory per Executor	Resources Per Executor
----------------	------	-------	---------------------	------------------------

▼ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor
----------------	------	-------	---------------------	------------------------

Workspace do Spark:

- ❑ Executamos no terminal o seguinte comando:
 - `/opt/spark/sbin/start-slave.sh spark://localhost:7077`
- ❑ Em seguida, podemos executar o **Spark** direto no terminal:
 - `spark-shell`



```
ive access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/12/13 23:21:10 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://192.168.0.12:4040
Spark context available as 'sc' (master = local[*], app id = local-1639448471157).
Spark session available as 'spark'.
Welcome to

  ____              __
 / _  /__  ___  ___/_/
/_  /_  /_  /_  /_  /_
/_  /_  /_  /_  /_  /_
/_  /_  /_  /_  /_  /_
/_  /_  /_  /_  /_  /_
/_  /_  /_  /_  /_  /_
/_  /_  /_  /_  /_  /_
/_  /_  /_  /_  /_  /_
/_  /_  /_  /_  /_  /_

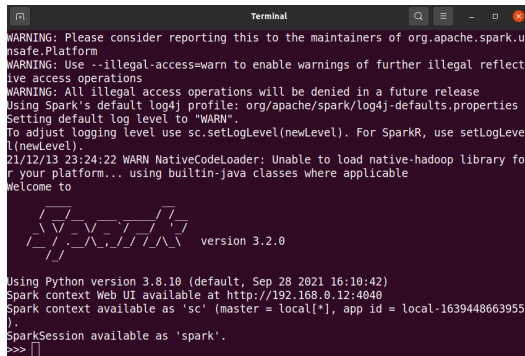
version 3.2.0

Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 11.0.11)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Workspace do Spark:

- ❑ O comando executado nos trouxe o terminal do **Scala**, mas utilizaremos o **Pyspark** em nossos trabalhos:
 - pyspark



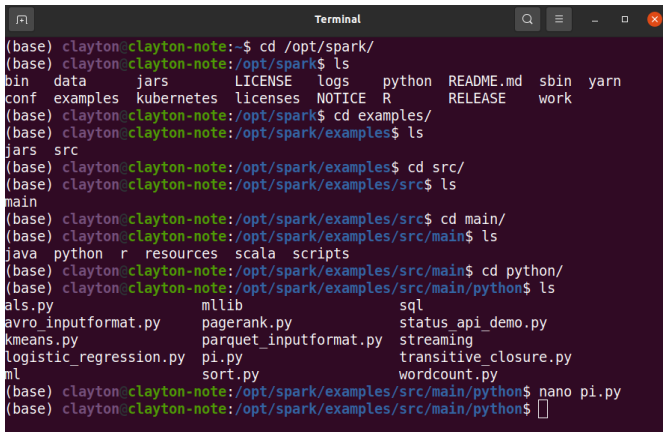
```
Terminal
WARNING: Please consider reporting this to the maintainers of org.apache.spark.
UnsafePlatform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
21/12/13 23:24:22 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Welcome to

      _/ _\ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/ _/
      \_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_
      version 3.2.0

Using Python version 3.8.10 (default, Sep 28 2021 16:10:42)
Spark context Web UI available at http://192.168.0.12:4040
Spark context available as 'sc' (master = local[*], app id = local-1639448663955
).
SparkSession available as 'spark'.
>>> □
```

Rodando exemplos no Spark:

- ❑ Acessando o diretório do spark:



```
Terminal
(base) clayton@clayton-note:~$ cd /opt/spark/
(base) clayton@clayton-note:/opt/spark$ ls
bin  data  jars  LICENSE  logs  python  README.md  sbin  yarn
conf  examples  kubernetes  licenses  NOTICE  R  RELEASE  work
(base) clayton@clayton-note:/opt/spark$ cd examples/
(base) clayton@clayton-note:/opt/spark/examples$ ls
jars  src
(base) clayton@clayton-note:/opt/spark/examples$ cd src/
(base) clayton@clayton-note:/opt/spark/examples/src$ ls
main
(base) clayton@clayton-note:/opt/spark/examples/src$ cd main/
(base) clayton@clayton-note:/opt/spark/examples/src/main$ ls
java  python  r  resources  scala  scripts
(base) clayton@clayton-note:/opt/spark/examples/src/main$ cd python/
(base) clayton@clayton-note:/opt/spark/examples/src/main/python$ ls
als.py  mllib  sql
avro_inputformat.py  pagerank.py  status_api_demo.py
kmeans.py  parquet_inputformat.py  streaming
logistic_regression.py  pi.py  transitive_closure.py
ml  sort.py  wordcount.py
(base) clayton@clayton-note:/opt/spark/examples/src/main/python$ nano pi.py
(base) clayton@clayton-note:/opt/spark/examples/src/main/python$
```

Exercícios

- Verifique as informações apresentadas no diretório de Exercícios do GitHub
 - Bom trabalho!!!