

# Spark – Big Data Processing

## Aula 2



**Semantix<sup>®</sup>**

All about data

# Quem sou eu?



## Rodrigo Augusto Rebouças

Engenheiro de dados da Semantix  
Instrutor do Semantix Academy

### Contatos

[rodrigo.augusto@semantix.com.br](mailto:rodrigo.augusto@semantix.com.br)  
[linkedin.com/in/rodrigo-reboucas](https://www.linkedin.com/in/rodrigo-reboucas)



# Jupyter Notebook

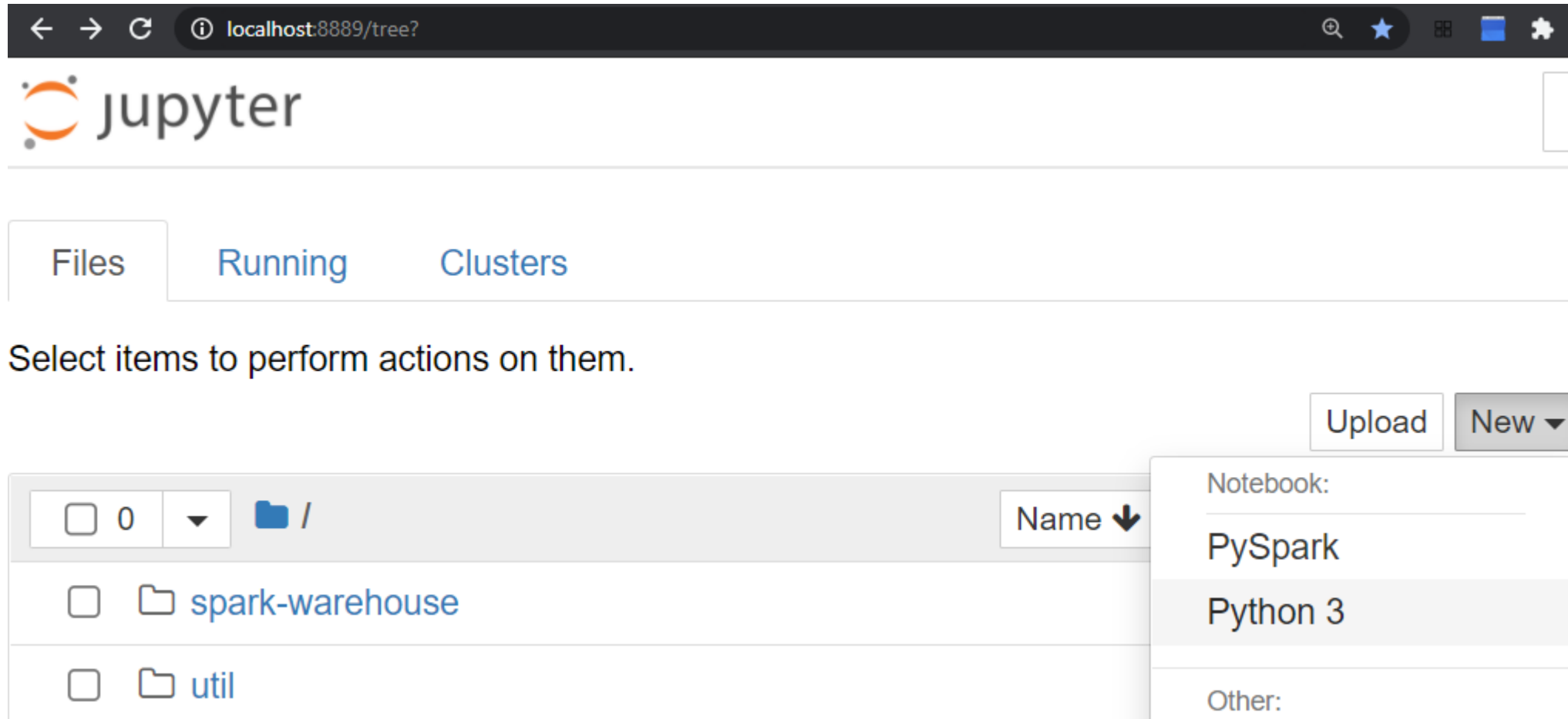
- **Conceitos**

# Projeto Jupyter

- Desenvolver softwares
  - Open-Source
  - Open-Standards
  - Serviços para computação interativa em diversas linguagens
- Jupyter Notebook
  - Ambiente computacional web
  - Criar documentos
    - Código ativo
    - Equações
    - Graficos
    - Texto
- <https://jupyter.org/>



# Jupyter Notebook



The screenshot shows the Jupyter Notebook web interface in a browser. The address bar indicates the URL is `localhost:8889/tree?`. The Jupyter logo is visible in the top left. Below the logo, there are three tabs: "Files" (selected), "Running", and "Clusters". A message states "Select items to perform actions on them." Below this, there are buttons for "Upload" and "New". The "New" button has been clicked, opening a dropdown menu. The menu lists "Notebook:" with two options: "PySpark" and "Python 3" (which is highlighted). Below this, there is an "Other:" section. In the background, the file browser shows a directory structure with a root "/" and two subdirectories: "spark-warehouse" and "util".

localhost:8889/tree?

jupyter

Files Running Clusters

Select items to perform actions on them.

Upload New

0 /

Name

spark-warehouse

util

Notebook:

PySpark

Python 3

Other:

# Jupyter Notebook

- Projeto PySpark
  - Comandos em Python: `print("seja bem vindo")`
  - Comandos em Spark: `spark`
  - Comandos em Linux: `!pwd`
  - Extensão do arquivo: `.ipynb` (IPython Notebook)
- Atalhos
  - Executar a célula: `ctrl Enter`
  - Executar a célula e criar uma nova célula: `shift Enter`
  - Alterar a célula para o tipo código: `y`
  - Alterar a célula para o tipo markdown: `m`
  - Deletar uma célula: `dd`
  - Desfazer o deletar a célula: `z`
  - Ajuda: `h`

In [5]: `print("seja bem vindo")`

seja bem vindo

In [6]: `spark`

Out[6]: **SparkSession - hive**  
**SparkContext**

[Spark UI](#)

**Version**

v2.4.1

**Master**

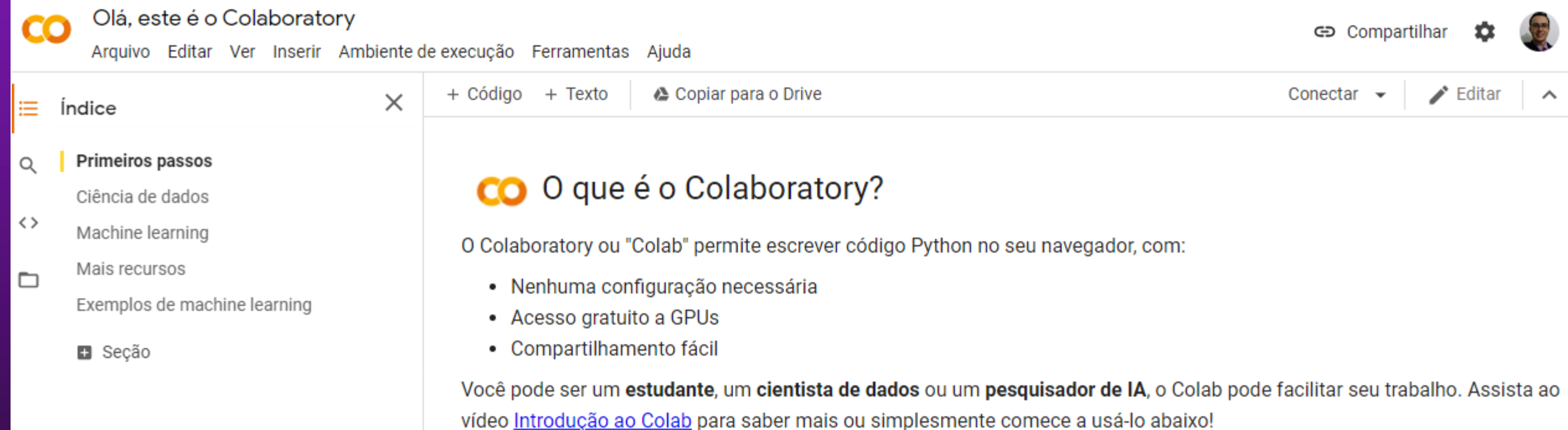
local[\*]

**AppName**

pyspark-shell

# Google Colab – Alternativa para Jupyter Notebook

○ <https://colab.research.google.com/>



The screenshot displays the Google Colaboratory (Colab) web interface. At the top, the Google Colab logo is followed by the text "Olá, este é o Colaboratory". To the right of this text are links for "Compartilhar" (Share), a settings gear icon, and a user profile picture. Below the header, a navigation bar includes "Arquivo" (File), "Editar" (Edit), "Ver" (View), "Inserir" (Insert), "Ambiente de execução" (Runtime environment), "Ferramentas" (Tools), and "Ajuda" (Help). On the left side, there is a sidebar with a search icon and a list of categories: "Índice" (Index), "Primeiros passos" (First steps), "Ciência de dados" (Data science), "Machine learning", "Mais recursos" (More resources), "Exemplos de machine learning", and a "Seção" (Section) button. The main content area features the Google Colab logo and the heading "O que é o Colaboratory?". Below this, a paragraph states: "O Colaboratory ou 'Colab' permite escrever código Python no seu navegador, com:". This is followed by a bulleted list: "• Nenhuma configuração necessária" (No configuration necessary), "• Acesso gratuito a GPUs" (Free access to GPUs), and "• Compartilhamento fácil" (Easy sharing). At the bottom of the main area, a paragraph says: "Você pode ser um **estudante**, um **cientista de dados** ou um **pesquisador de IA**, o Colab pode facilitar seu trabalho. Assista ao vídeo [Introdução ao Colab](#) para saber mais ou simplesmente comece a usá-lo abaixo!" (You can be a **student**, a **data scientist**, or an **AI researcher**, Colab can facilitate your work. Watch the video [Introduction to Colab](#) to know more or simply start using it below!).

# Sessão no Spark



# Sessão Spark

- Criar sessão Spark
  - Configurar nós
  - Configurar o projeto
- Classe SparkSession fornece o acesso às funcionalidades do spark
  - sql - Executa as consultas Spark SQL
  - catalog - Gerenciar tabelas
  - read - função para ler dados de um arquivo ou outra fonte de dados
  - conf - objeto para gerenciar configurações de Spark
  - sparkContext - Core Spark API

```
spark = SparkSession \
    .builder \
    .appName("Python") \
    .getOrCreate()
```

# Log

- Spark usa log4j para logging
  - Configurações do log em conf/log4j.properties
- Avaliação dos níveis de log
  - TRACE
  - DEBUG
  - INFO (default level in Spark applications)
  - WARN (default level in Spark shell)
  - ERROR
  - FATAL
  - OFF
- Setar o nível do log
  - `spark.sparkContext.setLogLevel("INFO")`

# API Catalog



# API Catalog

- Scala

- spark.catalog
  - listDatabases
  - setCurrentDatabase(nomeBD)
  - listTables
  - listColumns(nomeTabela)
  - dropTempView(nomeView)

- Python

- spark.catalog
  - listDatabases
  - setCurrentDatabase(nomeBD)
  - listTables
  - listColumns(nomeTabela)
  - dropTempView(nomeView)

# API Catalog - Exemplo

## Scala

```
scala> val tabDF = spark.sql("select * from bdtest.user").show(10)
```

```
scala> spark.catalog.listDatabases.show()
```

```
scala> spark.catalog.setCurrentDatabase("bdtest")
```

```
scala> spark.catalog.listTables.show()
```

```
scala> val tabDF = spark.sql("select * from user").show(10)
```

# API Catalog - Exemplo

## Python

```
tab_df = spark.sql("select * from bdtest.user").show(10)
```

```
spark.catalog.listDatabases()
```

```
spark.catalog.setCurrentDatabase("bdtest")
```

```
spark.catalog.listTables.show()
```

```
tab_df = spark.sql("select * from user").show(10)
```

# Exercícios – Preparar os dados e o ambiente

1. Configurar o jar do spark para aceitar o formato parquet em tabelas Hive
  - `curl -O https://repo1.maven.org/maven2/com/twitter/parquet-hadoop-bundle/1.6.0/parquet-hadoop-bundle-1.6.0.jar`
  - `docker cp parquet-hadoop-bundle-1.6.0.jar jupyter-spark:/opt/spark/jars`
2. Baixar os dados dos exercícios do treinamento no diretório spark/input (volume no Namenode)
  - `cd input`
  - `sudo git clone https://github.com/rodrigo-reboucas/exercises-data.git .`
3. Verificar o envio dos dados para o namenode
4. Criar no hdfs a seguinte estrutura: `/user/rodrigo/data`
5. Enviar todos os dados do diretório input para o hdfs em `/user/rodrigo/data`

# Exercícios – Testar o Jupyter Notebook

1. Criar o arquivo do notebook com o nome teste\_spark.ipynb
2. Obter as informações da sessão de spark (spark)
3. Obter as informações do contexto do spark (sc)
4. Setar o log como INFO.
5. Visualizar todos os banco de dados com o catalog
6. Ler os dados "hdfs://namenode:8020/user/rodrigo/data/juros\_selic/juros\_selic.json" com uso de Dataframe
7. Salvar o Dataframe como **juros** no formato de tabela Hive
8. Visualizar todas as tabelas com o catalog
9. Visualizar no hdfs o formato e compressão que está a tabela juros do Hive
10. Ler e visualizar os dados da tabela juros, com uso de Dataframe no formato de Tabela Hive
11. Ler e visualizar os dados da tabela juros , com uso de Dataframe no formato Parquet





# Semantix<sup>®</sup>

All about data

[contato@semantix.com.br](mailto:contato@semantix.com.br)

[www.semantix.com.br](http://www.semantix.com.br)