

TUTORIAL SPARK COM PYTHON MODO INTERATIVO E EDITOR

Prof. Dino Magri
professor.dinomagri@gmail.com

Esse documento tem por objetivo descrever os passos para rodar código em modo interativo (`yarn client`) e modo editor (`yarn cluster`).

O pré-requisito para realizar esse tutorial é ter Spark configurado junto com o YARN e Hadoop. Veja detalhes no arquivo **Configuração Ambiente Big Data**.

PASSO 1 – UTILIZANDO O MODO EDITOR NO SPARK

- 1) Abra um editor de texto de sua preferência (Gedit ou Sublime). Crie um arquivo chamado `minhaapp.py` na pasta `/home/labdata/Desktop/`. Utilize o comando `subl` para abrir o Sublime Text Editor ou o comando `gedit`.

```
cd /home/labdata/Desktop/  
subl minhaapp.py &
```

- 2) O primeiro passo é importar as bibliotecas necessárias para criar nossa aplicação.

```
from pyspark import SparkContext, SparkConf  
from pyspark.sql import SQLContext, Row
```

- 3) Temos que garantir que toda vez que nosso código rodar, ele seja executado como programa principal. Desta forma precisamos verificar se a variável especial `__name__` tem o valor: `"__main__"`.

```
if __name__ == "__main__":
```

- 4) Com as bibliotecas importadas, crie uma variável que irá receber a instância da Classe `SparkConf()`. Utilize um método chamado `setAppName`.

```
conf = SparkConf()  
conf.setAppName("MinhaAPP")
```

- 5) Pronto! Já temos as configurações mínimas definidas. Crie agora o contexto na variável `sc`. Utilize a variável `conf` como parâmetro da Instância `SparkConf()`.

```
sc = SparkContext(conf=conf)
```

- 6) Envie o arquivo que está em `/home/labdata/cluster-conf/data/pessoas.txt` para o HDFS. Abra o terminal e execute:

```
hdfs dfs -rm -r /user/labdata/pessoas.txt

hdfs dfs -put /home/labdata/cluster-conf-labdata/data/pessoas.txt
/user/labdata
```

- 7) Volte ao Sublime e carregue o arquivo localizado no HDFS com o método `textFile` do `SparkContext()`.

```
linhas = sc.textFile('hdfs://elephant:8020/user/labdata/pessoas.txt')
```

- 8) Agora para cada linha precisamos separar as colunas, como foi feito anteriormente

```
cols = linhas.map(lambda linha: linha.split(';'))
```

- 9) Se optarmos por criar o `DataFrame` passando como parâmetro a variável `cols`, o `Dataframe` será criado com os nomes das colunas gerados automaticamente (`_1`, `_2`, `_3` e assim sucessivamente). Para nomear as colunas, podemos utilizar a classe `Row`, que permite que os campos sejam acessados como atributos, e assim pode-se aplicar uma função `lambda` em cada elemento através do `map`.

```
dados = cols.map(lambda coluna: Row(nome=str(coluna[0]),
idade=int(coluna[1]), altura=float(coluna[2])))
```

- 10) Crie o `DataFrame` (`df`) passando por parâmetro a variável `dados`.

```
sqlContext = SQLContext(sc)
df = sqlContext.createDataFrame(dados)
```

- 11) Já temos nossos dados carregados. Para testar, crie uma consulta no formato SQL. O objetivo é selecionar pessoas com idade maior ou igual que 20 e menor ou igual que 30. Utilize o método `sql` disponível no objeto instanciado `sqlContext`. Porém, antes é necessário criar uma tabela temporária chamada `pessoas`.

```
df.registerTempTable("pessoas")

pessoas = sqlContext.sql("SELECT nome FROM pessoas WHERE idade >= 20 AND
idade <= 30")
```

- 12) Temos um `DataFrame`, que contém as pessoas com idade entre 20 e 30. Para imprimir os elementos, precisamos utilizar o método `collect()` e recuperar os dados.

```
nomes = pessoas.collect()
```

Agora é possível utilizar o código abaixo para imprimir os nomes.

```
print("Imprimindo os nomes: ")
for n in nomes:
    print(n.nome)

print("Existem {} pessoas entre 20 e 30 anos".format(len(nomes)))
```

OBS: Salve o arquivo minhaapp.py e feche o editor de texto.

13) Para executar a aplicação, utilize o comando `$SPARK_HOME/bin/spark-submit` para executar no cluster. Execute:

```
$SPARK_HOME/bin/spark-submit --master yarn --deploy-mode cluster
/home/labdata/Desktop/minhaapp.py
```

- a. Acesse <http://horse:8088/> e verifique o `FinalStatus` da execução.
- b. Verifique nos logs os nomes que apareceram.

Obs 1: Caso o `FinalStatus` não for de sucesso, acesse o Histórico e visualize os logs. Esse é o local onde terá as mensagens de erro que ocorreram na execução do seu código.

Obs 2: Caso não seja possível visualizar os logs via interface web, execute no terminal da VM Elephant:

```
yarn logs -applicationId <Application ID>
```

<Application ID> pode ser encontrado na interface <http://horse:8088/>.

Código Completo do arquivo minhaapp.py

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext, Row

if __name__ == "__main__":
    conf = SparkConf()
    conf.setAppName("MinhaAPP")

    sc = SparkContext(conf=conf)
    linhas = sc.textFile('hdfs://elephant:8020/user/labdata/pessoas.txt')
    cols = linhas.map(lambda linha: linha.split(';'))

    dados = cols.map(lambda coluna: Row(nome=str(coluna[0]), idade=int(coluna[1]), altura=float(coluna[2])))
    sqlContext = SQLContext(sc)

    df = sqlContext.createDataFrame(dados)
    df.registerTempTable("pessoas")

    pessoas = sqlContext.sql("SELECT nome FROM pessoas WHERE idade >= 20 AND idade <= 30")
    nomes = pessoas.collect()
```

```
print("Imprimindo os nomes: ")
for n in nomes:
    print(n.nome)
print("Existem {} pessoas entre 20 e 30 anos".format(len(nomes)))
```

PASSO 2 – MODO INTERATIVO

Para utilizar o modo interativo, iremos configurar o **PySpark** para rodar com o Jupyter Notebook.

1) Na VM Elephant, abra o arquivo `/home/labdata/.bashrc` e adicione as variáveis de ambientes definidas abaixo (caso esteja comentado, apenas remova o `#` do início de cada linha).

```
export HADOOP_CONF_DIR=/etc/hadoop/conf
export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce

export PYSPARK_DRIVER_PYTHON=/usr/bin/jupyter
export PYSPARK_DRIVER_PYTHON_OPTS="notebook --
NotebookApp.open_browser=True --NotebookApp.ip='*'"

export PYSPARK_SUBMIT_ARGS='--master yarn --deploy-mode client'
export SPARK_EXECUTOR_MEMORY="2G"
export SPARK_HIVE=true
```

2) Execute os comandos:

```
source /home/labdata/.bashrc

sudo pip3.6 install jupyter
```

PASSO 3 – VERIFICAR OS SERVIÇOS NAS VMS

Tenha certeza que todos os serviços estejam rodando em cada uma das VMs.

Na VM Elephant, execute o script abaixo para reiniciar todos os serviços em todas as VMs:

```
cd /home/labdata/cluster-conf-labdata

sudo ./scripts/restart_all_services.sh
```

PASSO 4 – UTILIZANDO O MODO INTERATIVO DO PYSPARK

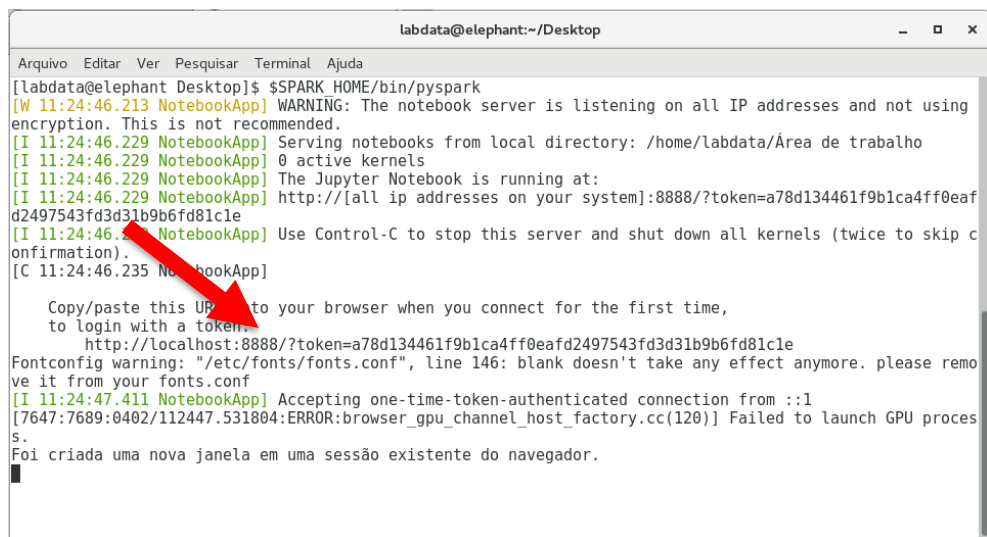
1) Adicione o arquivo `/home/labdata/cluster-conf-labdata/data/shakespeare.txt` não esteja no HDFS, adicione:

```
hdfs dfs -put /home/labdata/cluster-conf-labdata/data/shakespeare.txt
/user/labdata/shakespeare.txt
```

- 2) O primeiro passo é inicializar o **PySparkShell**. As configurações realizadas no Passo 2, servem para facilitar a execução do **PySparkShell** através do Jupyter. Acesse a pasta desejada e inicie o `pyspark`.

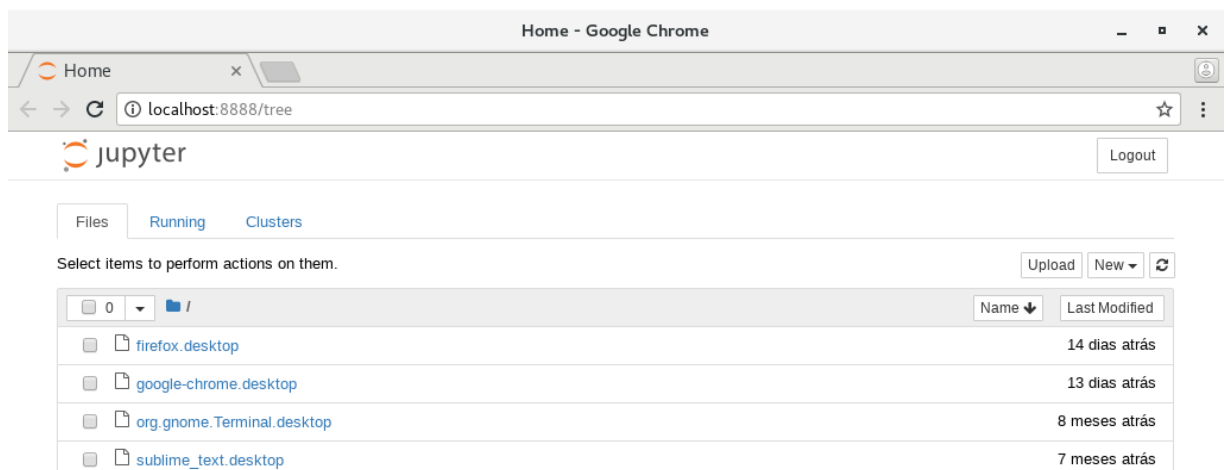
```
cd /home/labdata/Desktop/  
$SPARK_HOME/bin/pyspark
```

- 3) Uma tela parecida com a figura abaixo irá aparecer. Note que a porta que o Jupyter utiliza é a que definimos na variável de ambiente `PYSPARK_DRIVER_PYTHON_OPTS`.



```
labdata@elephant:~/Desktop  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
[labdata@elephant Desktop]$ $SPARK_HOME/bin/pyspark  
[W 11:24:46.213 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using  
encryption. This is not recommended.  
[I 11:24:46.229 NotebookApp] Serving notebooks from local directory: /home/labdata/Área de trabalho  
[I 11:24:46.229 NotebookApp] 0 active kernels  
[I 11:24:46.229 NotebookApp] The Jupyter Notebook is running at:  
[I 11:24:46.229 NotebookApp] http://[all ip addresses on your system]:8888/?token=a78d134461f9b1ca4ff0eaf  
d2497543fd3d31b9b6fd81c1e  
[I 11:24:46.229 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip c  
onfirmation).  
[C 11:24:46.235 NotebookApp]  
  
Copy/paste this URL into your browser when you connect for the first time,  
to login with a token:  
http://localhost:8888/?token=a78d134461f9b1ca4ff0eafd2497543fd3d31b9b6fd81c1e  
Fontconfig warning: "/etc/fonts/fonts.conf", line 146: blank doesn't take any effect anymore. please remo  
ve it from your fonts.conf  
[I 11:24:47.411 NotebookApp] Accepting one-time-token-authenticated connection from ::1  
[7647:7689:0402/112447.531804:ERROR:browser_gpu_channel_host_factory.cc(120)] Failed to launch GPU proces  
s.  
Foi criada uma nova janela em uma sessão existente do navegador.  
█
```

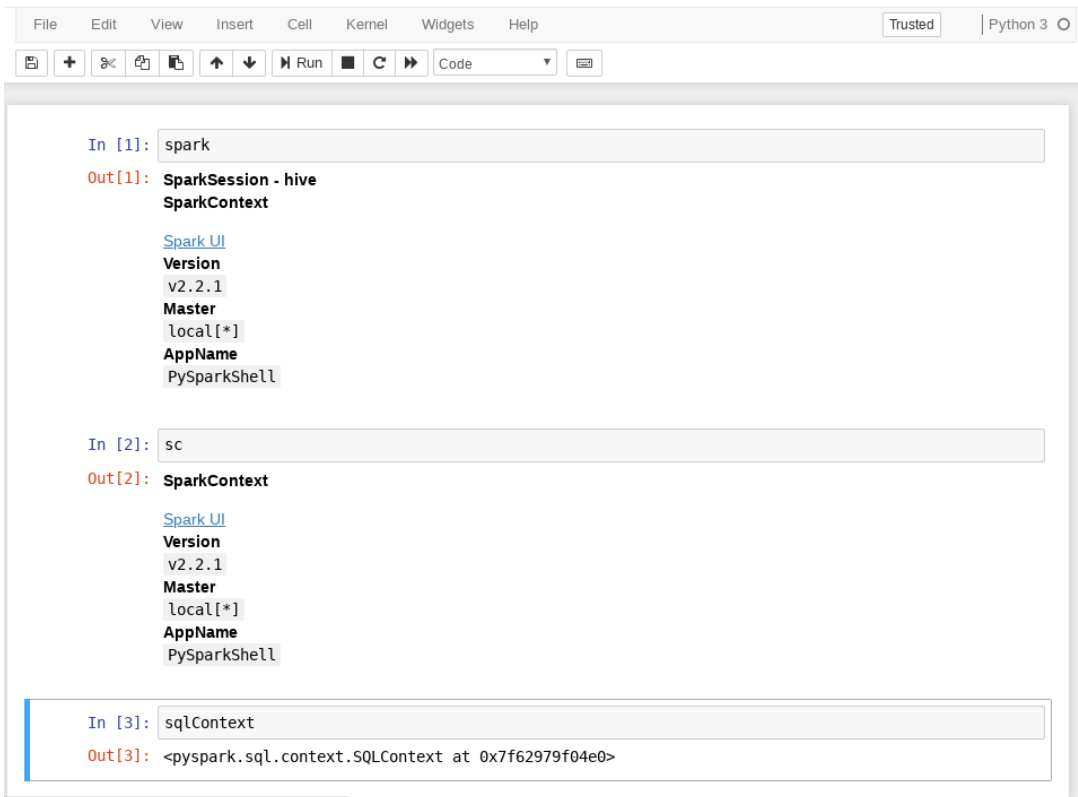
- 4) Abra o navegador e digite <http://localhost:8888/>. Uma página como a imagem abaixo deverá ser visualizada.



LEMBRE-SE - INICIE APENAS UM NOTEBOOK POR VEZ!!

- 5) Clique em `New` -> `Python 3` para criar um notebook.
- 6) Como iniciamos o PySparkShell, já temos algumas variáveis definidas que são inicializadas

pelo PySpark. No Jupyter aberto, imprima as três variáveis, `spark`, `sc` e `sqlContext`. A resposta deve ser parecida com a imagem abaixo:



```
In [1]: spark
Out[1]: SparkSession - hive
SparkContext

Spark UI
Version
v2.2.1
Master
local[*]
AppName
PySparkShell

In [2]: sc
Out[2]: SparkContext

Spark UI
Version
v2.2.1
Master
local[*]
AppName
PySparkShell

In [3]: sqlContext
Out[3]: <pyspark.sql.context.SQLContext at 0x7f62979f04e0>
```

7) Carregue o arquivo `README.md`

```
arquivo = sc.textFile('hdfs://elephant:8020/user/labdata/shakespeare.txt')
```

8) Utilizando o método `flatMap` iremos separar as palavras através do espaço vazio. Isso deverá ser feito em cada linha, utilizando a função `lambda`. Execute o código abaixo:

```
palavras = arquivo.flatMap(lambda linha: linha.split(" "))
```

9) Utilize o método `take` para recuperar os 5 primeiros elementos do RDD.

```
palavras.take(5)
```

10) Para as palavras salvas nos RDDs iremos criar uma tupla que terá a **palavra** e o valor **1**. Desta forma é possível reduzir pela chave e ter a soma total da quantidade de vezes que uma determinada palavra aparece.

```
mapeamento = palavras.map(lambda palavra: (palavra, 1))
mapeamento.take(5)
```

11) Utilize o método `reduceByKey` aplicando a função `lambda`.

```
contagem = mapeamento.reduceByKey(lambda v1, v2: v1 + v2)
```

```
contagem.take(5)
```

12) Utilize o método `count()` para realizar quantas palavras foram mapeadas.

```
print("Existem {} palavras distintas".format(contagem.count()))
```

13) Utilize o método `saveAsTextFile` para salvar o arquivo no HDFS. Verifique se o nome do arquivo já não existe. Caso exista, remova utilizando a linha de comando do Hadoop.

```
contagem.saveAsTextFile("hdfs://elephant:8020/user/labdata/resultado_contagem")
```

14) Abra o navegador e digite <http://elephant:4040/>. Acesse as páginas **Jobs** e **Stages** para verificar os processos e estágios executados no Spark.

15) Verifique também no HDFS, se o arquivo resposta foi distribuído corretamente.

IMPORTANTE: Finalize a notebook desta atividade. Clique em `File -> Close and Halt`. Acesse o terminal e finalize a execução do PySpark (Digite `Ctrl + C`).

PASSO 5 – UTILIZAR NOVAMENTE O MODO EDITOR

Caso seja necessário utilizar novamente o modo editor será necessário comentar as seguintes variáveis de ambiente no `/home/labdata/.bashrc`:

Está assim:

```
export PYSPARK_DRIVER_PYTHON=/usr/bin/jupyter
export PYSPARK_DRIVER_PYTHON_OPTS="notebook --
NotebookApp.open_browser=True --NotebookApp.ip='*'
export PYSPARK_SUBMIT_ARGS='--master yarn --deploy-mode client'
```

Deve ficar assim:

```
#export PYSPARK_DRIVER_PYTHON=/usr/bin/jupyter
#export PYSPARK_DRIVER_PYTHON_OPTS="notebook --
NotebookApp.open_browser=True --NotebookApp.ip='*'
#export PYSPARK_SUBMIT_ARGS='--master yarn --deploy-mode client'
```

Por fim, para não ter a necessidade de fazer o logoff e login ou reiniciar a máquina, execute os seguintes comandos no Terminal:

```
unset PYSPARK_DRIVER_PYTHON
unset PYSPARK_DRIVER_PYTHON_OPTS
unset PYSPARK_SUBMIT_ARGS
```