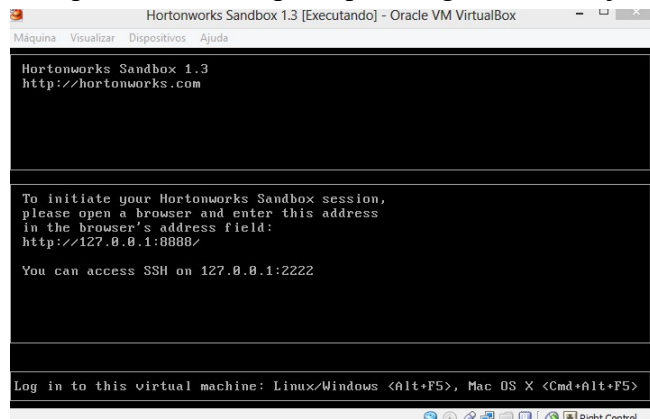


Professor: Cláudio Lúcio
Atividade: Utilizando o Spark-sql

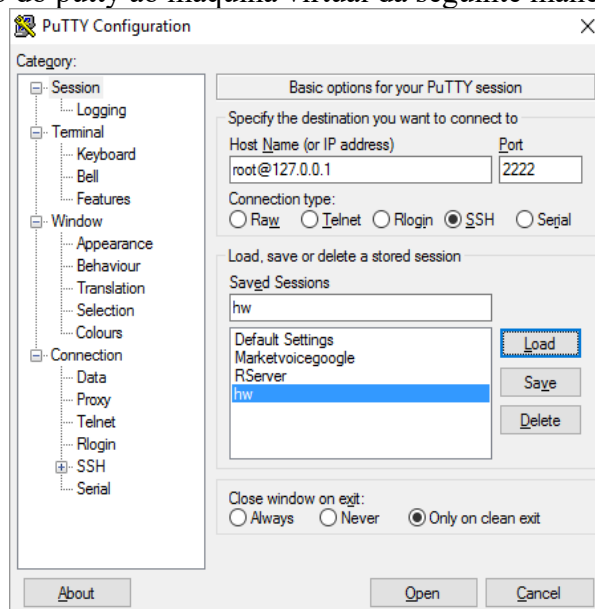
- 1) Para acesso ao Shell do Spark vamos usar a versão do Spark que vem na máquina virtual do HortonWorks:

Esta é a versão 2.3.2 do produto e pode ser obtida em:
<http://hortonworks.com/products/ Hortonworks-Sandbox/>

- 2) Inicialize a máquina virtual. Espere que a seguinte tela seja exibida:



- 3) A recomendação é usar um cliente para acesso SSH ao servidor do spark. Baixe a ferramenta putty.exe;
- 4) Configure o acesso do putty ao máquina virtual da seguinte maneira:



- 5) Clique em “Open” e então digite a senha do Root: hadoop

```
root@sandbox:~  
Using username "root".  
root@127.0.0.1's password:  
Last login: Fri Dec 23 16:56:09 2016 from 10.0.2.2  
[root@sandbox ~]#
```

- 6) Vamos fazer uma alteração para compartilhar as tabelas criadas entre os usuários no spark

```
cd /usr/hdp/2.3.2.0-2950/spark  
vi conf/spark-defaults.conf  
Aperte a tecla INSERT
```

Insira a última linha conforme abaixo:

```
spark.sql.hive.thriftServer.singleSession true
```

```
root@sandbox:/usr/hdp/2.3.2.0-2950/spark  
Generated by Apache Ambari. Tue Oct 27 12:41:01 2015  
  
spark.driver.extraJavaOptions -Dhdp.version=2.3.2.0-2950  
spark.history.kerberos.keytab none  
spark.history.kerberos.principal none  
spark.history.provider org.apache.spark.deploy.yarn.history.YarnHistoryProvider  
spark.history.ui.port 18080  
spark.yarn.am.extraJavaOptions -Dhdp.version=2.3.2.0-2950  
spark.yarn.applicationMaster.waitTries 10  
spark.yarn.containerLauncherMaxThreads 25  
spark.yarn.driver.memoryOverhead 384  
spark.yarn.executor.memoryOverhead 384  
spark.yarn.historyServer.address sandbox.hortonworks.com:18080  
spark.yarn.max.executor.failures 3  
spark.yarn.preserve.staging.files false  
spark.yarn.queue default  
spark.yarn.scheduler.heartbeat.interval-ms 5000  
spark.yarn.services org.apache.spark.deploy.yarn.history.YarnHistoryService  
spark.yarn.submit.file.replication 3  
spark.sql.hive.thriftServer.singleSession true  
~  
~  
~  
Type :quit<Enter> to exit Vim
```

Aperte a tecla ESC

Aperte a tecla ":" e digite "wq" e Aperte a tecla ENTER

Agora vamos ajustar os acessos aos diretórios do HDFS:

```
su hdfs  
hadoop fs -chmod -R 777 /user/hive/  
hadoop fs -chmod -R 777 /apps/hive/
```

- 7) Vamos interagir com a interface spark-sql do Spark, para tal vamos acessar o bin/spark-sql:
- 8) Digite os seguintes comandos:

```
cd /usr/hdp/2.3.2.0-2950/spark  
bin/spark-sql --packages com.databricks:spark-csv_2.10:1.1.0
```

```
root@sandbox:/usr/hdp/2.3.2.0-2950/spark
:: resolving dependencies :: org.apache.spark#spark-submit-parent;1.0
  confs: [default]
  found com.databricks#spark-csv_2.10;1.1.0 in central
  found org.apache.commons#commons-csv;1.1 in central
  found com.univocity#univocity-parsers;1.5.1 in central
:: resolution report :: resolve 353ms :: artifacts dl 12ms
  :: modules in use:
    com.databricks#spark-csv_2.10;1.1.0 from central in [default]
    com.univocity#univocity-parsers;1.5.1 from central in [default]
    org.apache.commons#commons-csv;1.1 from central in [default]
-----
|               | modules                | artifacts |
|               | search|dwnlded|evicted|| number|dwnlded|
-----+-----+-----+-----+-----+-----+-----+-----+-----+
| default       | 3    | 0    | 0    | 0    || 3    | 0    |
-----+-----+-----+-----+-----+-----+-----+
:: retrieving :: org.apache.spark#spark-submit-parent
  confs: [default]
  0 artifacts copied, 3 already retrieved (0kB/9ms)
SET spark.sql.hive.thriftServer.singleSession=true
SET spark.sql.hive.thriftServer.singleSession=true
SET spark.sql.hive.version=0.13.1
SET spark.sql.hive.version=0.13.1
spark-sql>
```

Caso ocorra um erro no download dos pacotes acima, vamos ter que executar os seguintes passos:

```
su root ou exit
cd /etc/spark/2.3.2.0-2950/0/
mkdir lib
cd lib

wget http://central.maven.org/maven2/com/databricks/spark-csv_2.10/1.5.0/spark-csv_2.10-1.5.0.jar

wget http://central.maven.org/maven2/org/apache/commons/commons-csv/1.2/commons-csv-1.2.jar

su hdfs
spark-sql --jars spark-csv_2.10-1.5.0.jar,commons-csv-1.2.jar
```

9) Veja que estamos usando também a conexão com o Hive. Desta forma conseguiremos acessar as tabelas que já estão no HIVE;

10) Experimente os comandos:

```
show tables;
SET spark.sql.hive.version=0.13.1
SET spark.sql.hive.version=0.13.1
spark-sql> show tables;
sample_07      false
sample_08      false
Time taken: 1.184 seconds, Fetched 2 row(s)
spark-sql>
```

Veja que o segundo parâmetro retornado indica se a tabela é temporária.

```
desc sample_07;
Time taken: 1.184 seconds, Fetched 2 row(s)
spark-sql> desc sample_07;
code      string  NULL
description string  NULL
total_emp int    NULL
salary    int    NULL
Time taken: 1.737 seconds, Fetched 4 row(s)
spark-sql>
```

```
select distinct total_emp,salary from sample_07;
```

```

120060 46320
5250 55490
499640 80460
20270 72150
103320 62500
100820 24000
1390260 23920
6620 78200
33950 42190
28890 145210
67700 73240
47210 97170
152870 87550
15780 37680
Time taken: 3.749 seconds, Fetched 823 row(s)
spark-sql>

```

- 11) Observe que ainda não usamos o Spark SQL, usamos apenas o Hive;
- 12) Vamos trabalhar com o SparkSQL
- 13) Vamos recriar a estrutura do json utilizado anteriormente, mas agora com um *data frame* , para isto vamos usar o seguinte comando:

```

CREATE TEMPORARY TABLE pessoasval USING org.apache.spark.sql.json
OPTIONS (path '/user/hue/pessoasval.json');

```

```

spark-sql> CREATE TEMPORARY TABLE pessoasval USING org.apache.spark.sql.json OPT
IONS (path '/user/hue/pessoasval.json');
Time taken: 0.238 seconds
spark-sql> [root@sandbox spark]#

```

- 14) Veja, agora, o comando para exibir as tabelas:

```

IONS (path '/user/hue/pessoasval.json');
Time taken: 0.662 seconds
spark-sql> show tables;
pessoasval      true
sample_07      false
sample_08      false
Time taken: 0.087 seconds, Fetched 3 row(s)
spark-sql>

```

- 15) Vamos consultar como o spark inferiu a estrutura do Json:

```

desc pessoasval;
sample_08      false
Time taken: 0.087 seconds, Fetched 3 row(s)
spark-sql> desc pessoasval;
dat      string
id       bigint
val      bigint
Time taken: 0.572 seconds, Fetched 3 row(s)
spark-sql>

```

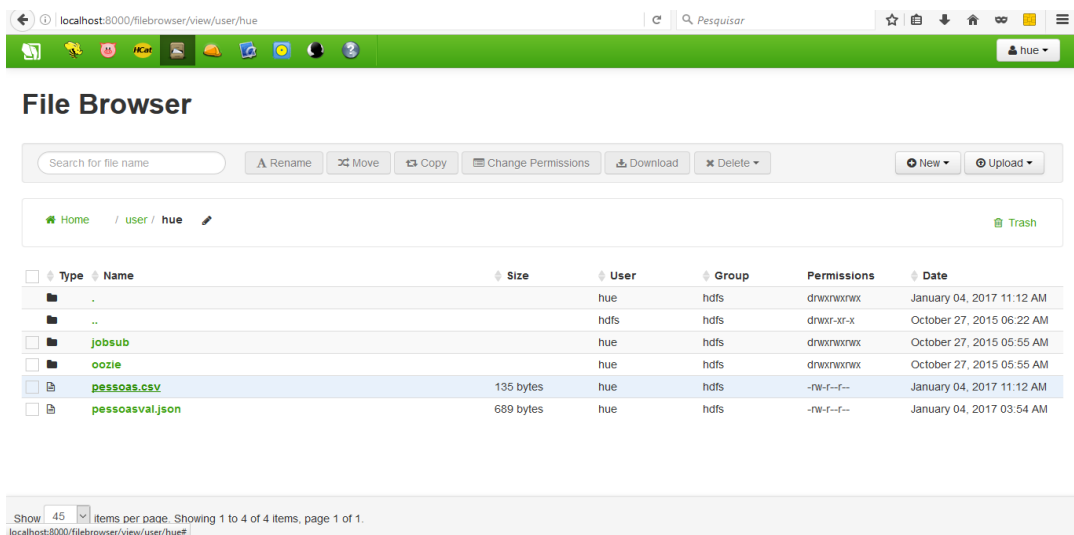
```

select * from pessoasval;

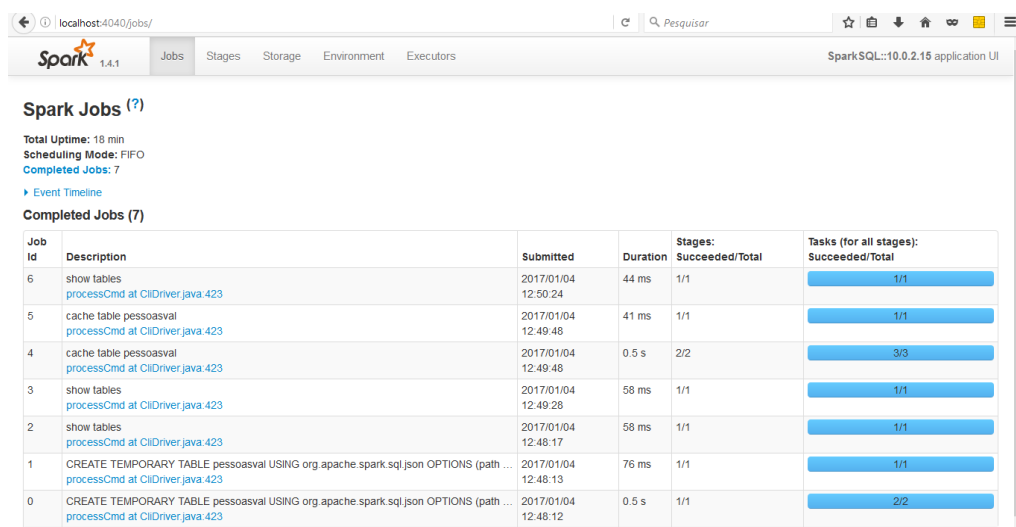
```

```
val    bigint
Time taken: 0.572 seconds, Fetched 3 row(s)
spark-sql> select * from pessoasval;
12/01/2006      1      45
04/06/2009      2      53
18/01/2012      4     345
12/01/2016      5     435
08/09/2015      8    2003
12/11/2000     12     100
12/01/2006     45     200
12/01/2006     13    99999
12/03/2006      1     405
04/09/2009      2     503
01/10/2012      4      35
12/12/2016      5      45
01/01/2015      8      23
02/01/2002     12      10
12/12/2006     45      20
12/01/2007     13    99999
Time taken: 0.23 seconds, Fetched 16 row(s)
spark-sql>
```

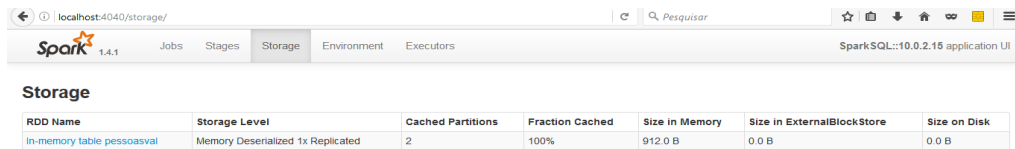
16) Vamos agora usar fazer um upload de um arquivo CSV para o HDFS: pessoas.csv



17) Acesse agora o UI SparkSQL e veja os jobs executados

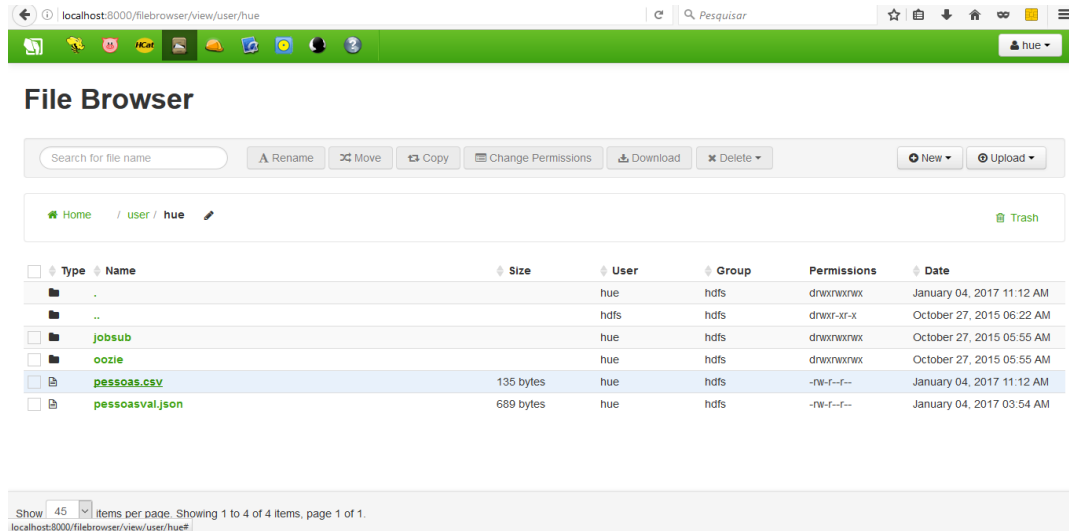


18) Veja a tabela que esta na aba 'Storage':



RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in ExternalBlockStore	Size on Disk
In-memory table pessoasval	Memory Deserialized 1x Replicated	2	100%	912.0 B	0.0 B	0.0 B

19) Vamos agora usar fazer um upload de um arquivo CSV para o HDFS: pessoas.csv



Type	Name	Size	User	Group	Permissions	Date
Folder	.		hue	hdfs	drwxrwxrwx	January 04, 2017 11:12 AM
Folder	..		hdfs	hdfs	drwxr-xr-x	October 27, 2015 06:22 AM
Folder	jobsub		hue	hdfs	drwxrwxrwx	October 27, 2015 05:55 AM
Folder	oozie		hue	hdfs	drwxrwxrwx	October 27, 2015 05:55 AM
File	pessoas.csv	135 bytes	hue	hdfs	-rw-r--r--	January 04, 2017 11:12 AM
File	pessoasval.json	689 bytes	hue	hdfs	-rw-r--r--	January 04, 2017 03:54 AM

20) Vamos então criar esta tabela neste fluxo de dados do spark-sql:

```
CREATE TEMPORARY TABLE pessoas USING com.databricks.spark.csv
OPTIONS (path "/user/hue/pessoas.csv", header "true");
```

21) Veja os campos da tabela:

```
desc pessoas;
Time taken: 0.252 seconds
spark-sql> desc pessoas;
id      string
nome    string
idade   string
gen      string
Time taken: 0.17 seconds, Fetched 4 row(s)
spark-sql>
```

22) Agora vamos criar uma consulta envolvendo as duas tabelas

```
select nome, idade, gen, sum(val) as val from pessoas inner join
pessoasval on pessoas.id=pessoasval.id group by nome, idade, gen;
```

```
>
> select nome, idade, gen, sum(val) as val from pessoas inner join pesso
asval on pessoas.id=pessoasval.id group by nome, idade, gen;
Albert  28      M      380
Simone  18      T      2026
Teste   38      T      199998
Gloria  43      F      556
Laura   33      F      480
Marta   45      F      110
Jairo   82      M      220
Bob      45      M      450
Time taken: 4.079 seconds, Fetched 8 row(s)
spark-sql> [root@sandbox spark]#
```

23) Agora vamos criar uma consulta envolvendo as duas tabelas e salvando em CSV no HDFS:

```
CREATE TABLE totpeessoasvall USING com.databricks.spark.csv OPTIONS  
(path "/user/hive/exemplo") AS select nome, idade, gen, sum(val) as  
val from pessoas inner join pessoasval on pessoas.id=pessoasval.id  
group by nome, idade, gen;
```