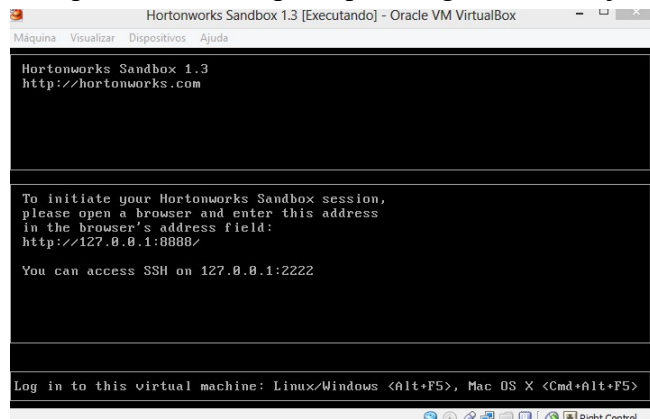


Professor: Cláudio Lúcio  
Atividade Verificando processos e RDD's

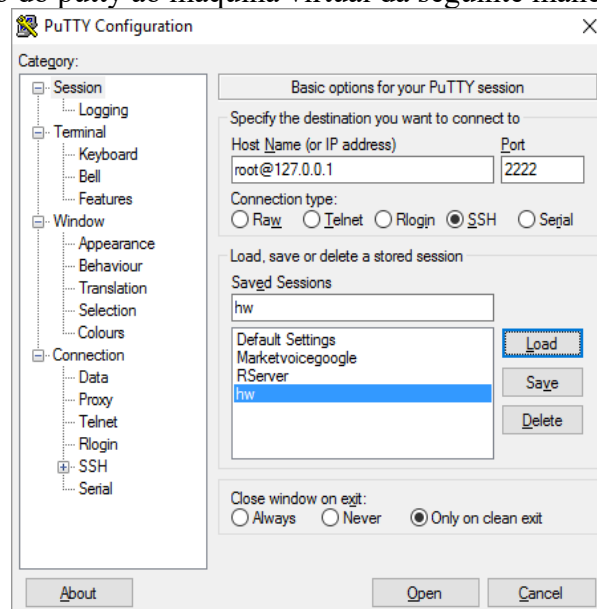
- 1) Para acesso ao Shell do Spark vamos usar a versão do Spark que vem na máquina virtual do HortonWorks:

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 ~]#
```

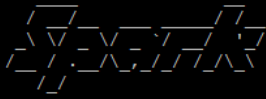
- 6) Vamos garantir que estamos com o usuário Admin do HDFS

```
sudo -u hdfs bash
```

- 7) Primeiramente vamos interagir com a interface python para o Spark, para tal vamos acessar o bin/pyspark:

- 8) Digite os seguintes comandos:

```
cd /usr/hdp/2.3.2.0-2950/spark/bin  
pyspark
```

```
root@sandbox:/usr/hdp/2.3.2.0-2950/spark/bin  
16/12/26 17:25:39 WARN SparkConf: The configuration key 'spark.yarn.applicationMaster.waitTries' has been deprecated as of Spark 1.3 and and may be removed in the future. Please use the new key 'spark.yarn.am.waitTime' instead.  
16/12/26 17:25:39 WARN SparkConf: The configuration key 'spark.yarn.applicationMaster.waitTries' has been deprecated as of Spark 1.3 and and may be removed in the future. Please use the new key 'spark.yarn.am.waitTime' instead.  
16/12/26 17:25:39 INFO Executor: Starting executor ID driver on host localhost  
16/12/26 17:25:40 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 53908.  
16/12/26 17:25:40 INFO NettyBlockTransferService: Server created on 53908  
16/12/26 17:25:40 INFO BlockManagerMaster: Trying to register BlockManager  
16/12/26 17:25:40 INFO BlockManagerMasterEndpoint: Registering block manager localhost:53908 with 265.4 MB RAM, BlockManagerId(driver, localhost, 53908)  
16/12/26 17:25:40 INFO BlockManagerMaster: Registered BlockManager  
Welcome to  
 version 1.4.1  
Using Python version 2.6.6 (r266:84292, Jul 23 2015 15:22:56)  
SparkContext available as sc, HiveContext available as sqlContext.  
>>>
```

- 9) Veja que estamos com a versão 1.4.1 do Spark;

- 10) Vamos inicialmente utilizar uma base de dados para testar alguns métodos

```
peessoas = []  
peessoas.append({'id':1,'nome':'Bob', 'idade':45,'gen':'M'})  
peessoas.append({'id':2,'nome':'Gloria', 'idade':43,'gen':'F'})  
peessoas.append({'id':4,'nome':'Albert', 'idade':28,'gen':'M'})  
peessoas.append({'id':5,'nome':'Laura', 'idade':33,'gen':'F'})  
peessoas.append({'id':8,'nome':'Simone', 'idade':18,'gen':'T'})  
peessoas.append({'id':12,'nome':'Marta', 'idade':45,'gen':'F'})  
peessoas.append({'id':45,'nome':'Jairo', 'idade':82,'gen':'M'})  
peessoas.append({'id':13,'nome':'Teste', 'idade':38,'gen':'T'})  
peessoasRdd=sc.parallelize(peessoas)
```

- 11) Utilizar o comando collect para vizualizar o conteúdo do RDD

```
peessoasRdd.collect()
```

12) Agora vamos fazer uma 'consulta' para verificar as pessoas com idade maior do que 30 anos

```
peessoas_M30 = pessoasRdd.filter(lambda x: x['idade']>30)
peessoas_M30.collect()
peessoas_M30.count()
```

13) Agora vamos fazer uma 'consulta' para verificar do sexo masculino

```
peessoas_GenM = pessoasRdd.filter(lambda x: x['gen'].upper() == 'M')
peessoas_GenM.collect()
peessoas_GenM.count()
```

14) Agora vamos verificar a quantidade de pessoas por sexo

```
peessoas_AggGen = pessoasRdd.map(lambda gen: gen['gen'])
peessoas_AggGen.collect()
peessoas_AggGen.countByValue()
```

15) Precisamos fazer a média da idade das pessoas. Para isto vamos usar a função *aggregate*:

```
seqOp = (lambda x,y: (x[0] + y['idade'],x[1] + 1))
combOp = (lambda x,y: (x[0] + y[0], x[1] + y[1]))
[soma,quantidade] = pessoasRdd.aggregate((0,0), seqOp, combOp)
media = soma/quantidade
media
```

16) Vamos fazer a união de pessoas acima de 30 mais pessoas do sexo masculino:

```
peessoas_GenM_30 = pessoas_M30.union(peessoas_GenM)
peessoas_GenM_30.collect()
```

17) Pessoas que estão nos dois critérios:

```
peessoas_M30_id = pessoas_M30.map(lambda x: x['id'])
peessoas_GenM_id = pessoas_GenM.map(lambda x: x['id'])
peessoas_GenM_And_30 = pessoas_M30_id.intersection(peessoas_GenM_id)
int_GenM_And_30 = pessoas_GenM_And_30.collect()
peessoas_GenM_And_30_full = pessoasRdd.filter(lambda x: x['id'] in int_GenM_And_30 )
peessoas_GenM_And_30_full.collect()
```

18) Seleção aleatório de duas pessoas do conjunto:

```
peessoasRdd.takeSample(True,2)
```

19) Importação de arquivo :

```
import json
peessoasValFile = sc.textFile("/user/hue/pessoasval.json")
peessoasValRdd = pessoasValFile.flatMap(lambda arq : (json.loads(arq)))
peessoasValRdd.collect()
```

20) Quantidade de ocorrências por id :

```
peessoasValRdd_id = pessoasValRdd.map(lambda x: x['id'])
peessoasValRdd_id.countByValue()
```