Table of Contents

spark-3.1.2-bin-hadoop3.2.tgz
Hadoop 3.1.2

- Use the version specified above.

1. Install Spark in centos Linux – 60 Minutes

Use: VM Centos 7 64 bit. CLI

Start the VM or the container and open a terminal to the host using putty. Perform the following activities in the console.

Download the required software

Apache Spark – Version 3.0.1 – Prebuilt for Apache Hadoop 3.2 and later.
File : spark-3.0.1-bin-hadoop3.2.tgz
Url : https://spark.apache.org/downloads.html

Inflate all the software in /opt folder.

JDK installation☹) (jdk-8u45-linux-x64.tar.gz)
https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html

Extract the jdk and rename the folder to jdk.

```
#tar -xvf jdk-8u45-linux-x64.tar.gz -C /opt
#mv jdk* jdk
```

Untar spark-X.X.0-bin-hadoop.X.tar to /opt

```
# tar -xvf  spark-x-bin-hadoop.x.tgz.gz -C /opt/
```

It should create a folder inside spark.X, rename to spark folder.

```
mv  sparkX  spark
```

Set the JAVA_HOME and initialize the PATH variable.

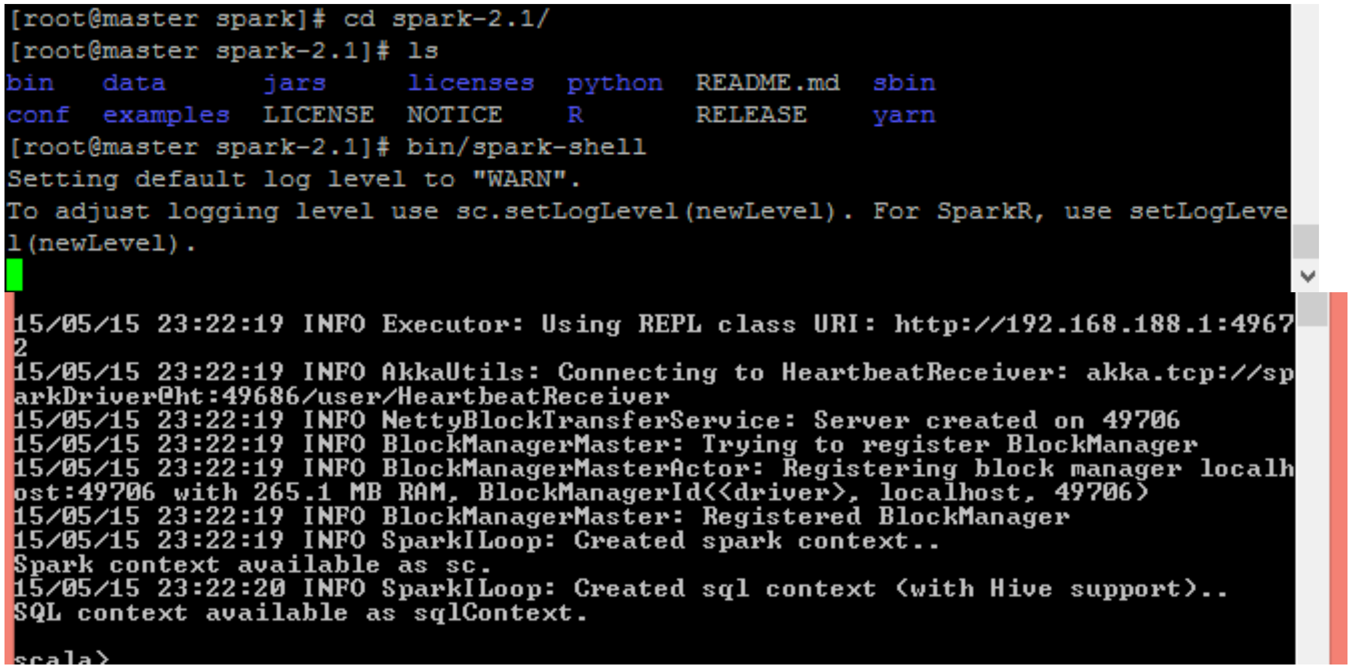
Open the profile and update with the following statements.

```
vi  ~/.bashrc
export  JAVA_HOME=/opt/jdk
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:/opt/spark/bin
```

Type bash, to initialize the profile.

Go to the installation directory:

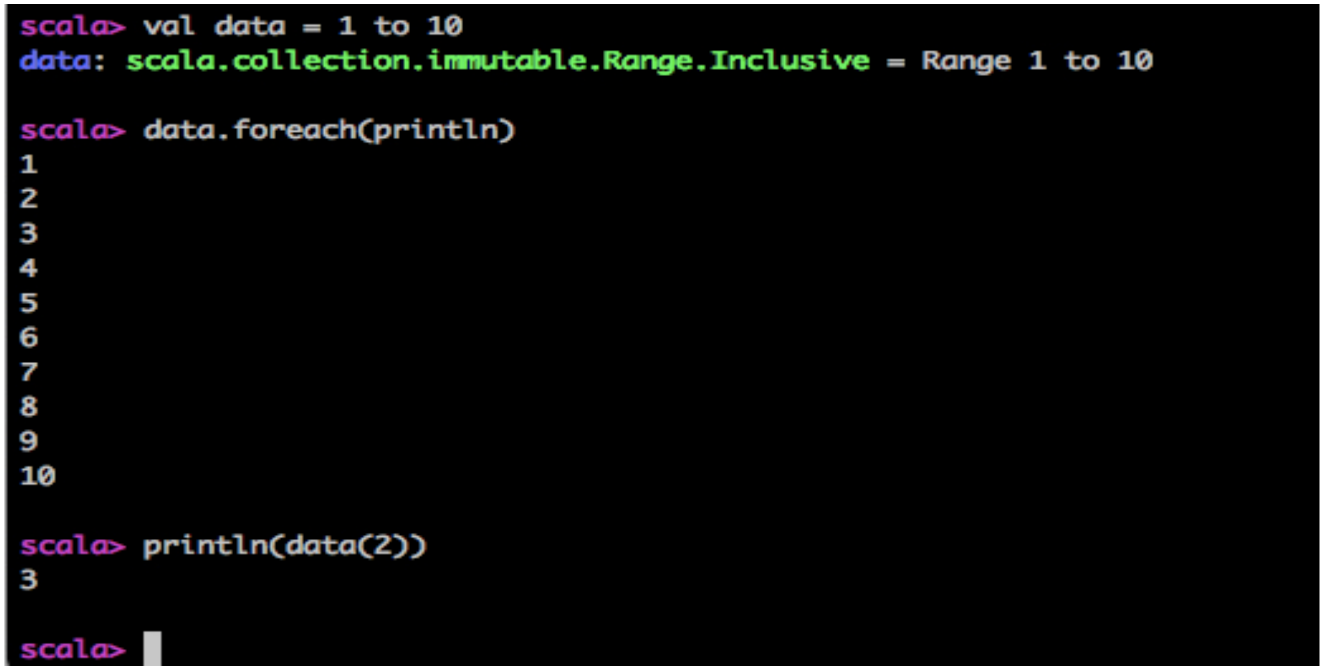
```
# cd /opt/spark/bin
execute
./spark-shell
```



```
[root@master spark]# cd spark-2.1/
[root@master spark-2.1]# ls
bin   data      jars     licenses  python   README.md  sbin
conf  examples  LICENSE  NOTICE    R        RELEASE    yarn
[root@master spark-2.1]# bin/spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).

15/05/15 23:22:19 INFO Executor: Using REPL class URI: http://192.168.188.1:4967
2
15/05/15 23:22:19 INFO AkkaUtils: Connecting to HeartbeatReceiver: akka.tcp://sp
arkDriver@ht:49686/user/HeartbeatReceiver
15/05/15 23:22:19 INFO NettyBlockTransferService: Server created on 49706
15/05/15 23:22:19 INFO BlockManagerMaster: Trying to register BlockManager
15/05/15 23:22:19 INFO BlockManagerMasterActor: Registering block manager localh
ost:49706 with 265.1 MB RAM, BlockManagerId(<driver>, localhost, 49706)
15/05/15 23:22:19 INFO BlockManagerMaster: Registered BlockManager
15/05/15 23:22:19 INFO SparkILoop: Created spark context..
Spark context available as sc.
15/05/15 23:22:20 INFO SparkILoop: Created sql context (with Hive support)..
SQL context available as sqlContext.

scala>
```

Enter the following command in the scala console to create a data set of 1…10 integers

```
#val data = 1 to 10
#data.foreach(println)
#println(data(2))
```

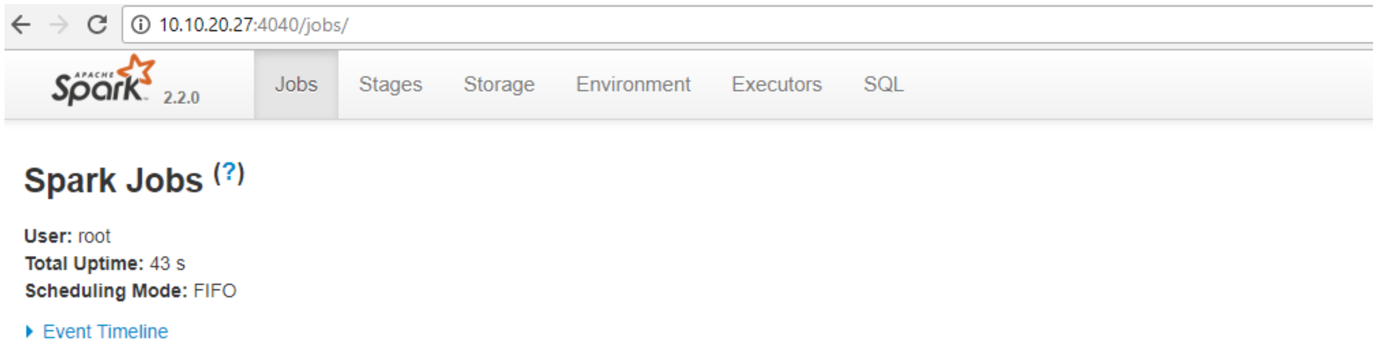


```
scala> val data = 1 to 10
data: scala.collection.immutable.Range.Inclusive = Range 1 to 10

scala> data.foreach(println)
1
2
3
4
5
6
7
8
9
10

scala> println(data(2))
3

scala> 
```

As seen above, you have initialized data variable with range from 1 to 10.
Then print its value using foreach method of scala.
You can also access the data using index.

Web UI of the spark shell can be accessed using the following URL. You can click on each tab and verify the screen.

http://ht:4040/jobs/



Spark Jobs (?)

User: root
Total Uptime: 43 s
Scheduling Mode: FIFO

Further details on this web UI will be discussed later.

You have successfully installed spark for local development.

--Lab Ends Here -----------------------------------

2. Exploring DataFrames using the Apache Spark Shell – Scala – 35 Minutes

Following features of Spark will be demonstrated here:

- Loading json file.
- Understand its schema
- Select required fields.
- Apply filter.

Start spark-shell

```
#spark-shell
```

Create a text file users.json which contains sample data as listed below in data folder:

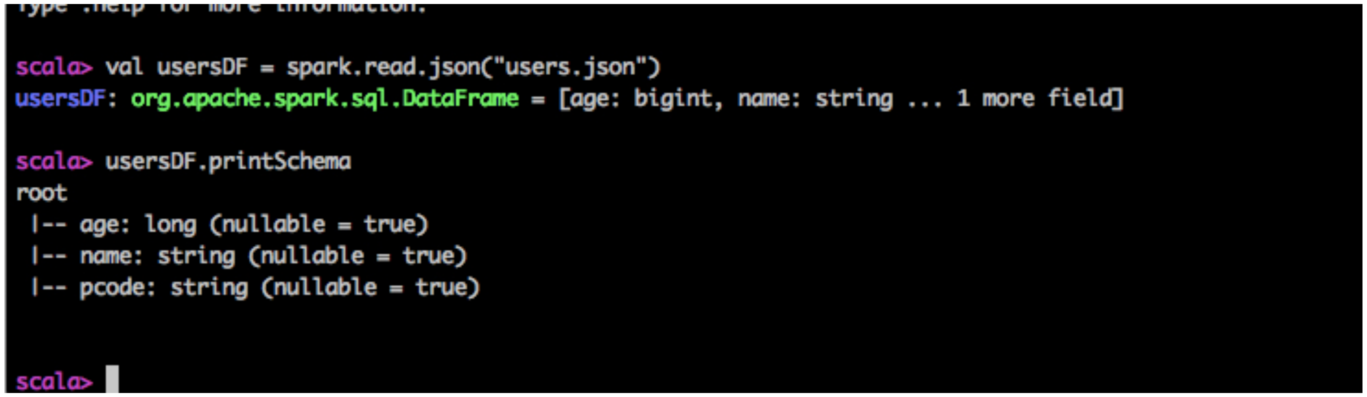
```
{"name":"Alice", "pcode":"94304"}
{"name":"Brayden", "age":30, "pcode":"94304"}
{"name":"Carla", "age":19, "pcode":"10036"}
{"name":"Diana", "age":46}
{"name":"Etienne", "pcode":"94104"}
```

Scala:

Initiate the spark-shell from the folder which you have created the above file.

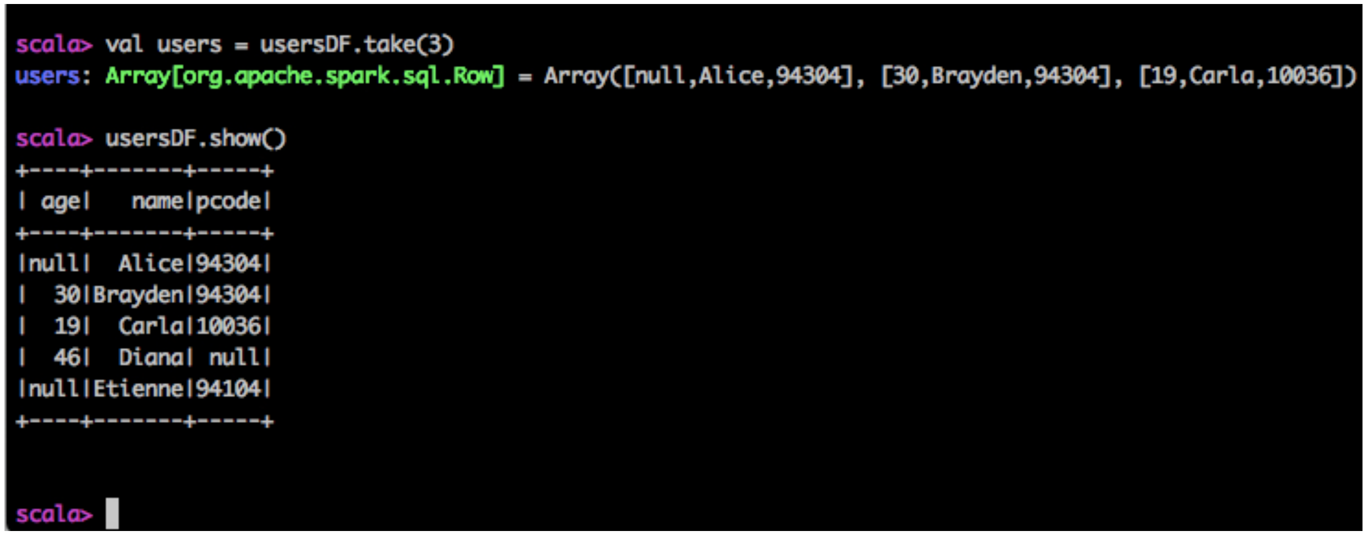
```
// Read the users json file as a dataframe.
val usersDF = spark.read.json("users.json")

// Find out the schema of the uploaded file
usersDF.printSchema()
```

As shown above, three fields will be displayed according to the json fields specified in the text file.



```
Type :help for more information.

scala> val usersDF = spark.read.json("users.json")
usersDF: org.apache.spark.sql.DataFrame = [age: bigint, name: string ... 1 more field]

scala> usersDF.printSchema
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
 |-- pcode: string (nullable = true)


scala> 
```

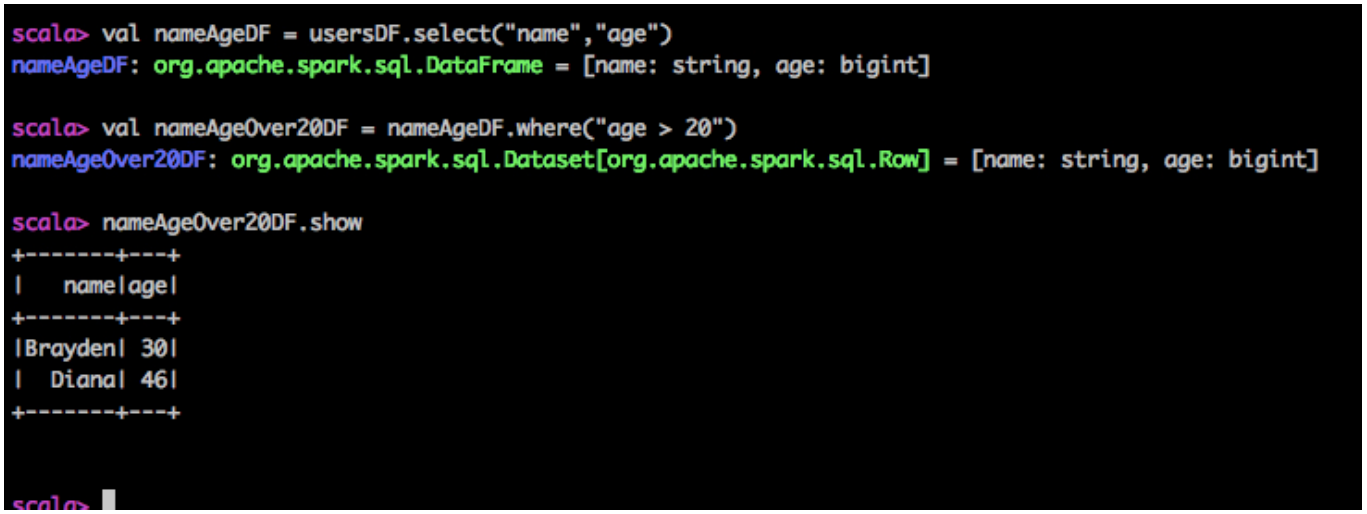
//Let us find out the first 3 records to have a sample data.
val users = usersDF.take(3)
usersDF.show()



```
scala> val users = usersDF.take(3)
users: Array[org.apache.spark.sql.Row] = Array([null,Alice,94304], [30,Brayden,94304], [19,Carla,10036])

scala> usersDF.show()
+----+-------+-----+
| age|   name|pcode|
+----+-------+-----+
|null|  Alice|94304|
|  30|Brayden|94304|
|  19|  Carla|10036|
|  46|  Diana| null|
|null|Etienne|94104|
+----+-------+-----+


scala> 
```

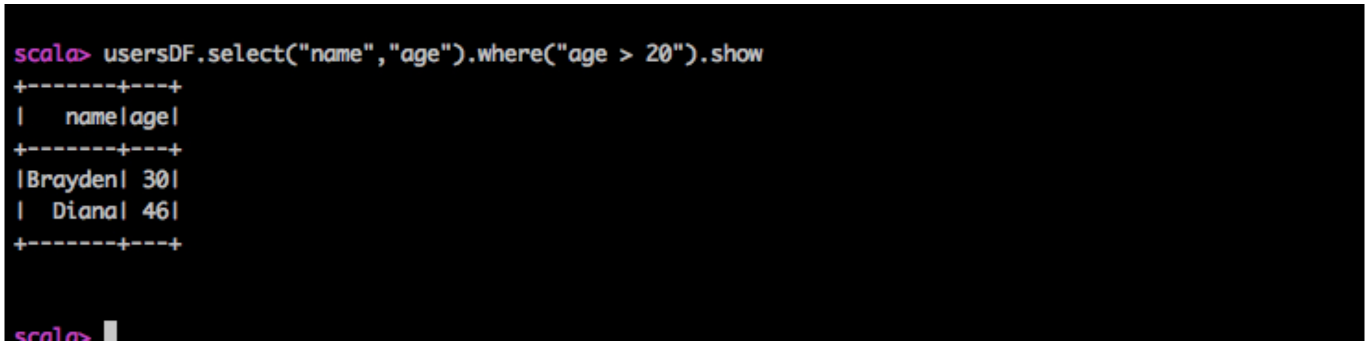
Out of the three fields, we are interested in only name and age fields. So, let us create a dataframe with only these two fields and apply a filter expression in which only person greater than 20 years are there in the dataframe.

```
val nameAgeDF = usersDF.select("name","age")
val nameAgeOver20DF = nameAgeDF.where("age > 20")
nameAgeOver20DF.show
```



```
scala> val nameAgeDF = usersDF.select("name","age")
nameAgeDF: org.apache.spark.sql.DataFrame = [name: string, age: bigint]

scala> val nameAgeOver20DF = nameAgeDF.where("age > 20")
nameAgeOver20DF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [name: string, age: bigint]

scala> nameAgeOver20DF.show
+-------+---+
|   name|age|
+-------+---+
|Brayden| 30|
|  Diana| 46|
+-------+---+


scala>
```

usersDF.select("name","age").where("age > 20").show



```
scala> usersDF.select("name","age").where("age > 20").show
+-------+---+
|   name|age|
+-------+---+
|Brayden| 30|
|  Diana| 46|
+-------+---+


scala>
```

You can also combine the functions as shown above. You will get the same result.

---------------------------------- Lab Ends Here---