Table of Contents

Last Updated: 12 Feb 2022

1. Prerequisite

Using Docker:
Create the first container:

Let us create a common network for our nodes.

#docker network create --driver bridge spark-net

#docker run -it --name spark0 --hostname spark0 --privileged --network spark-net -v /Volumes/Samsung_T5/software/:/Software -v /Volumes/Samsung_T5/software/install/:/opt -v /Volumes/Samsung_T5/software/data/:/data -p 8080:8080 -p 7077:7077 -p 4040:4040 -p 8081:8081 -p 8090:8090 -p 8888:8888 centos:7 /usr/sbin/init

Download the required software

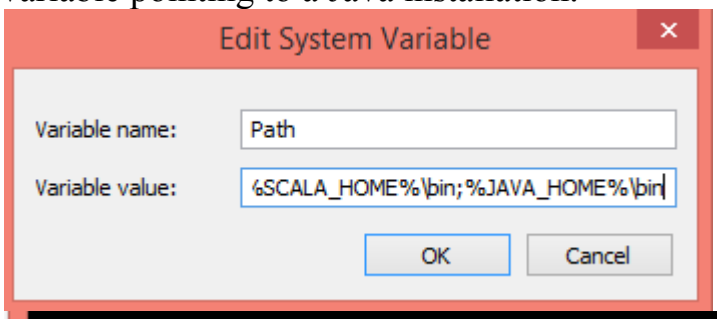
Apache Spark – Version 3.0.1 – Prebuilt for Apache Hadoop 3.2 and later.
File : spark-3.0.1-bin-hadoop3.2.tgz / **spark-3.2.1-bin-hadoop3.2.tgz**
Url : https://spark.apache.org/downloads.html

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

2. Exercise 1: Spark Installation: Windows:

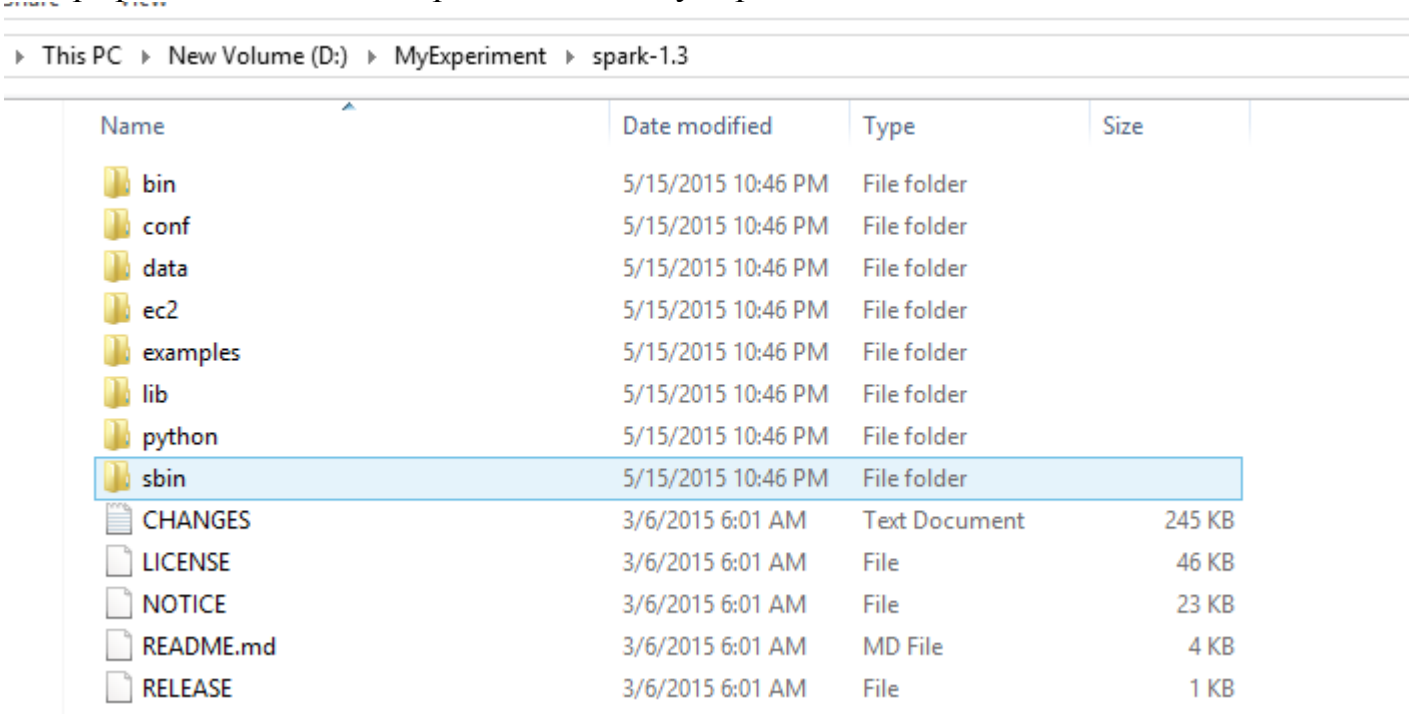
You need to install java on your system PATH, or the JAVA_HOME environment variable pointing to a Java installation.




```
C:\Users\henry>echo %JAVA_HOME%
D:\jdk1.8.0_45
```

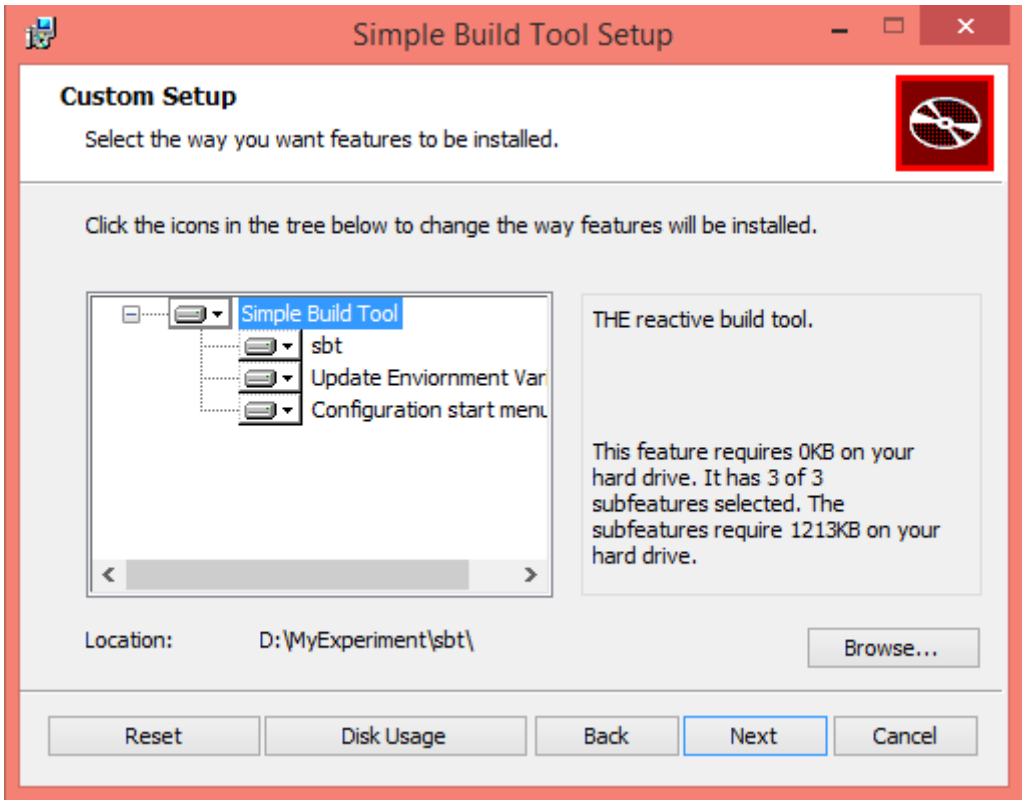
```
C:\Users\henry>java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
```

Spark runs on Java 6+ and Python 2.6+.
Unzip spark-1.3.0-bin-hadoop2.4.tar to D:\MyExperiment

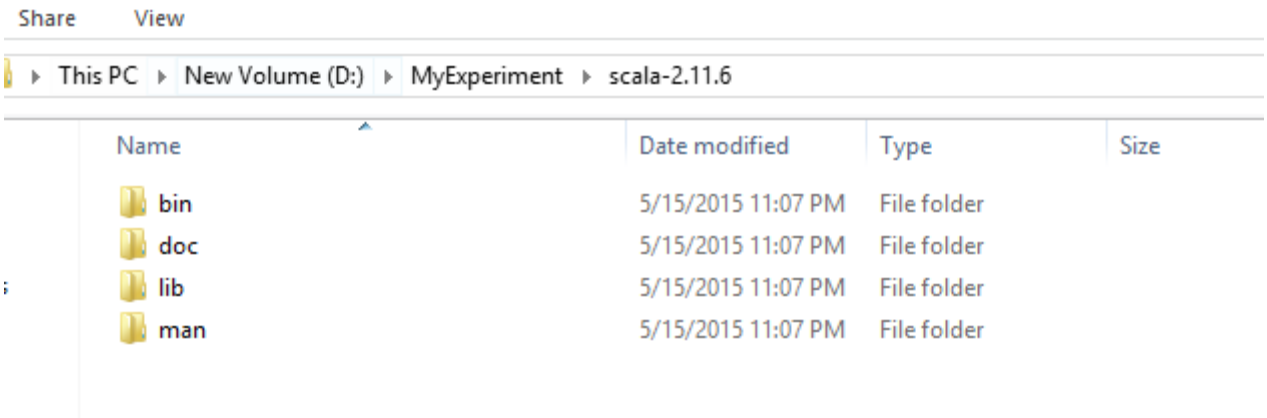


Install sbt.MSI & execute it.

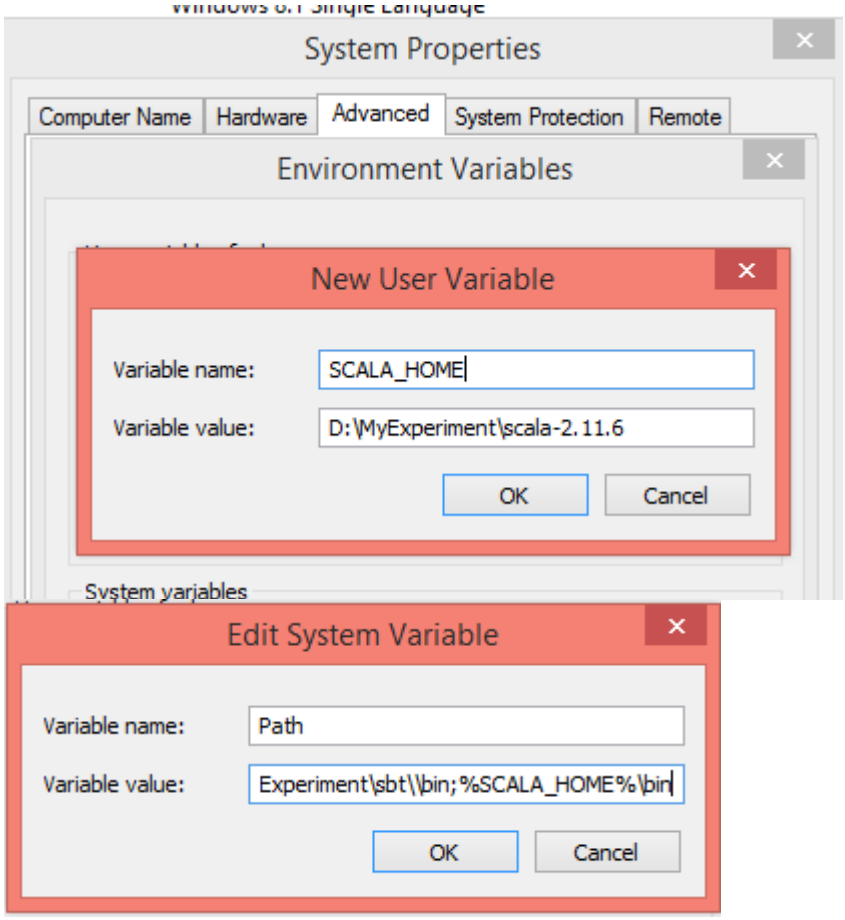
sbt-0.13.8.msi



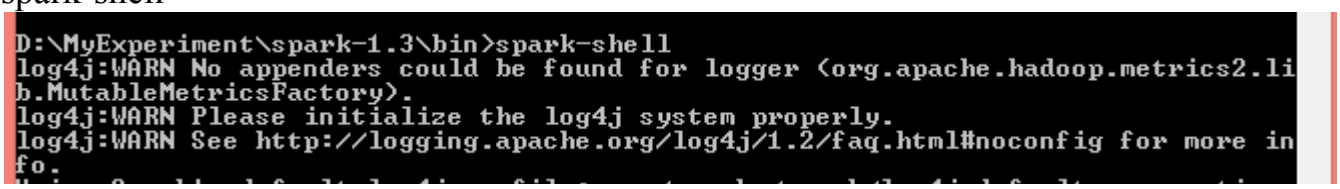
Untar scala-2.11.6



Set SCALA_HOME environment variable & set the PATH variable to the bin directory of scala.



change to the installation directory : D:\MyExperiment\spark-1.3>cd bin and execute spark-shell



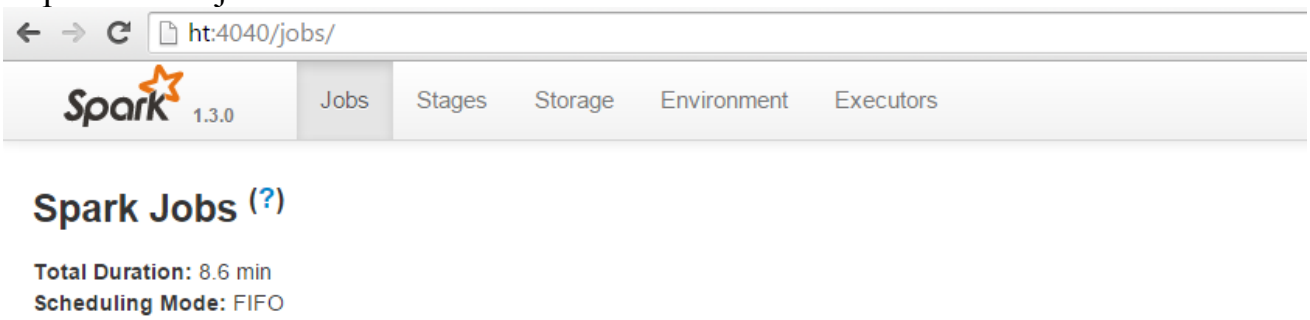
```
15/05/15 23:22:19 INFO Executor: Using REPL class URI: http://192.168.188.1:4967
2
15/05/15 23:22:19 INFO AkkaUtils: Connecting to HeartbeatReceiver: akka.tcp://sp
arkDriver@ht:49686/user/HeartbeatReceiver
15/05/15 23:22:19 INFO NettyBlockTransferService: Server created on 49706
15/05/15 23:22:19 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>
```

Create a data set of 1…10000 integers

val data = 1 to 10000

```
scala> val data = 1 to 10000
data: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8,
 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 1
07, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 1
23, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 1
39, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 1
55, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170...

scala>
```

http://ht:4040/jobs/



Spark Jobs (?)

Total Duration: 8.6 min
Scheduling Mode: FIFO

3. Install Spark in centos Linux – 60 Minutes(D)

Extract the jdk and rename the folder to jdk.

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

Extract 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 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

Install Python version 3.6.

Python 3.6 or above is required to run PySpark program.

#yum install -y python3

Type bash, to initialize the profile.

Go to the installation directory:

pyspark

Enter the following command in the pyspark console to count each alphabets.

// Code Begin. **********

//In the first two lines we are importing the Python libraries.

```
from operator import add
```

//Next we will create RDD from "Hello World" string:

```
data = sc.parallelize(list("Hello World"))
```

/*
Here we have used the object sc, sc is the SparkContext object which is created by pyspark before showing the console.

The parallelize() function is used to create RDD from String. RDD is also know as Resilient Distributed Datasets which is distributed data set in Spark. RDD process is done on the distributed Spark cluster.

Now with the following example we calculate number of characters and print on the console.

*/

```
counts = data.map(lambda x:
          (x, 1)).reduceByKey(add).sortBy(lambda x: x[1],
           ascending=False).collect()

for (word, count) in counts:
   print("{}: {}".format(word, count))
```

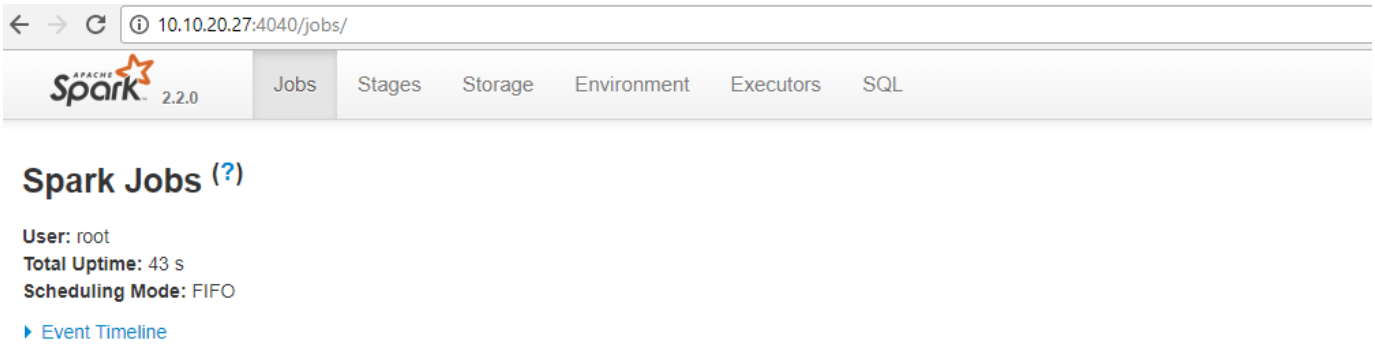
// Code Ends. **********

You should get the result as show below:

```
>>> for (word, count) in counts:
...     print("{}: {}".format(word, count))
...
l: 3
o: 2
d: 1
H: 1
 : 1
e: 1
W: 1
r: 1
>>>
```

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/



Further details on this web UI will be discussed later.

You have successfully installed spark for local development.
Note: You can start with default cores by the following command.
pyspark --master local[*]
----------Lab Ends Here ----------

4. Analysis Using Spark- Overview

Demonstrate this after installation lab
This Tutorial will demonstrate how we can use spark for data analytics - Overview.
Scenario - Upload the log file of a web access and find the following:
Which IP request url the most?
What is the max byte return from a server?
How many request receives from each client?

Let us intialize log entries, that will be used for analysis.

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:13 PM.

```
%pyspark
data = [('64.242.88.10 - - [07/Mar/2004:16:05:49 -0800] "GET
/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVariables HTTP/1.1"
401
    12846'),('64.242.88.10 - - [07/Mar/2004:16:06:51 -0800] "GET
/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1" 200 4523'
),('94.242.88.10 - - [07/Mar/2004:16:30:29 -0800] "GET
/twiki/bin/attach/Main/OfficeLocations HTTP/1.1" 401 12851')]
```

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:14 PM.

Convert the log entries into RDD

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:14 PM.

```
%pyspark
rdd = spark.sparkContext.parallelize(data)
```

FINISHED

Took 1 sec. Last updated by anonymous at July 04 2021, 1:17:15 PM.

Let us have a look how the RDD or data look like

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:15 PM.

```
%pyspark
rdd.collect()
```

```
['64.242.88.10 - - [07/Mar/2004:16:05:49 -0800] "GET
/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVariab
les HTTP/1.1" 401 12846', '64.242.88.10 - - [07/Mar/2004:16:06:51 -0800] "GET
/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1" 200 4523',
'94.242.88.10 - - [07/Mar/2004:16:30:29 -0800] "GET
/twiki/bin/attach/Main/OfficeLocations HTTP/1.1" 401 12851']
```

SPARK JOB FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:15 PM.

I would like to access the column using a name. So Let us intitialize RDD with a schema colum.

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:15 PM.

```
%pyspark
parts = rdd.map(lambda l: l.split(" "))
```

```
log = parts.map(lambda p: Row(ip=p[0], ts=(p[3]), atype = p[5], url=p[6] , code = p[8] ,
byte = p[9]))
```

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:16 PM.

Have I split the record fields correctly? Let us review the transformation

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:16 PM.

```
%pyspark
parts.collect()
```

```
[['64.242.88.10', '-', '-', '[07/Mar/2004:16:05:49', '-0800]', '"GET',
'/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVaria
bles', 'HTTP/1.1"', '401', '12846'], ['64.242.88.10', '-', '-',
'[07/Mar/2004:16:06:51', '-0800]', '"GET',
'/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2', 'HTTP/1.1"',
'200', '4523'], ['94.242.88.10', '-', '-', '[07/Mar/2004:16:30:29', '-0800]',
'"GET', '/twiki/bin/attach/Main/OfficeLocations', 'HTTP/1.1"', '401',
'12851']]
```

SPARK JOB FINISHED

Took 1 sec. Last updated by anonymous at July 04 2021, 1:17:17 PM.

I would like to use the Dataframe API. Let us convert from RDD to DF

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:17 PM.

```
%pyspark
# Infer the schema, and register the DataFrame as a table.
mylog = spark.createDataFrame(log)
```

SPARK JOB FINISHED

Took 1 sec. Last updated by anonymous at July 04 2021, 1:17:18 PM.

Have a view of our DF!

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:18 PM.

```
%pyspark
mylog.take(2)
```

```
[Row(atype=u'"GET', byte=u'12846', code=u'401', ip=u'64.242.88.10',
ts=u'[07/Mar/2004:16:05:49',
url=u'/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.Configuration
Variables'), Row(atype=u'"GET', byte=u'4523', code=u'200', ip=u'64.242.88.10',
ts=u'[07/Mar/2004:16:06:51',
url=u'/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2')]
```

SPARK JOB FINISHED

Took 1 sec. Last updated by anonymous at July 04 2021, 1:17:19 PM.

Now, we have the data in spark, let us answer the queries we have define in the first paragraphs.

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:19 PM.

Which IP request url the most?

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:19 PM.

```
%pyspark
```

```
from pyspark.sql.functions import col
mylog.groupBy("ip").count().orderBy(col("count").desc()).first()
Row(ip=u'64.242.88.10', count=2)
```

SPARK JOB FINISHED

Took 6 sec. Last updated by anonymous at July 04 2021, 1:17:26 PM.

What is the max byte return from a server?

FINISHED

Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:26 PM.

```
%pyspark
mylog.select("ip","byte").orderBy("byte").show()
mylog.select("ip","byte").orderBy("byte").first()
+------------+-----+ | ip| byte| +------------+-----+ |64.242.88.10|12846|
|94.242.88.10|12851| |64.242.88.10| 4523| +------------+-----+
Row(ip=u'64.242.88.10', byte=u'12846')
```

SPARK JOB FINISHED

Took 2 sec. Last updated by anonymous at July 04 2021, 1:17:28 PM.

How many request receives from each client?

FINISHED

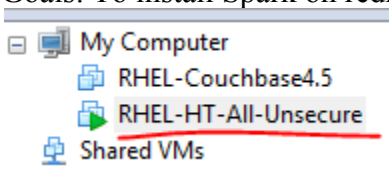
Took 0 sec. Last updated by anonymous at July 04 2021, 1:17:28 PM.

```
%pyspark
mylog.select("ip","url").groupBy("ip").count().show()
+------------+-----+ | ip|count| +------------+-----+ |94.242.88.10| 1|
|64.242.88.10| 2| +------------+-----+
```

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

5. Exercise : Spark Installation - Redhat Linux.

Goals: To install Spark on redhat linux OS.



Ensure to use latest software : spark-2.1.0-bin-hadoop2.7.tgz.tar
You need to replace with the specific version of software wherever relevant.

Create a folder /spark and un tar all related software inside this folder only. So that all installation happens in a single folder for better manageability.

#mkdir /spark

Untar spark-1.3.0-bin-hadoop2.4.tar to /spark

```
spark-1.3.0-bin-hadoop2.4/examples/src/main/resources/kv1.txt
[root@localhost software]# tar -xvf spark-1.3.0-bin-hadoop2.4.tgz -C /spark
```

it should create one folder inside /spark

```
[root@localhost spark]# ls
spark-1.3.0-bin-hadoop2.4
[root@localhost spark]#
```

Untar scala-2.11.6

tar -xvf scala-2.11.6.tgz -C /spark

```
[root@localhost software]#
[root@localhost software]# pwd
/mnt/hgfs/Spark/software
[root@localhost software]# ls
scala-2.11.6.tgz  spark-1.3.0-bin-hadoop2.4.tgz  winutils.exe
[root@localhost software]# tar -xvf scala-2.11.6.tgz -C /spark
```

Set the path and variable as follows: We are including Scala home and bin folder in the path variable.

vi ~/.bashrc
export SCALA_HOME=/spark/scala-2.11.6
export PATH=$PATH:$SCALA_HOME/bin

```
File  Edit  View  Terminal  Tabs  Help
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

export SCALA_HOME=/spark/scala-2.11.6
export PATH=$PATH:$SCALA_HOME/bin

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
~
```

Type bash

Install JDK as follows and set the PATH with environment as follows: (chose 32 or 64 bit depends on your server)

tar -xvf jdk-8u45-linux-x64.tar.gz -C /spark

```
[root@localhost software]# cd /spark
[root@localhost spark]# pwd
/spark
[root@localhost spark]# ls
jdk1.8.0_45  scala-2.11.6  spark-1.3.0-bin-hadoop2.4
[root@localhost spark]#
```

export JAVA_HOME=/spark/jdk1.8.0_45
export PATH=:$JAVA_HOME/bin :$PATH:$SCALA_HOME/bin

```
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

export SCALA_HOME=/spark/scala-2.11.6
export JAVA_HOME=/spark/jdk1.8.0_45
export PATH=$JAVA_HOME/bin:$PATH:$SCALA_HOME/bin
# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
~
```

Rename the installation folder as shown below: You need to change to installation directory before issuing this command.

#mv spark-2.1.0-bin-hadoop2.7/ spark-2.1/

change to the installation directory : \spark\spark-2.1>cd bin and execute ./spark-shell

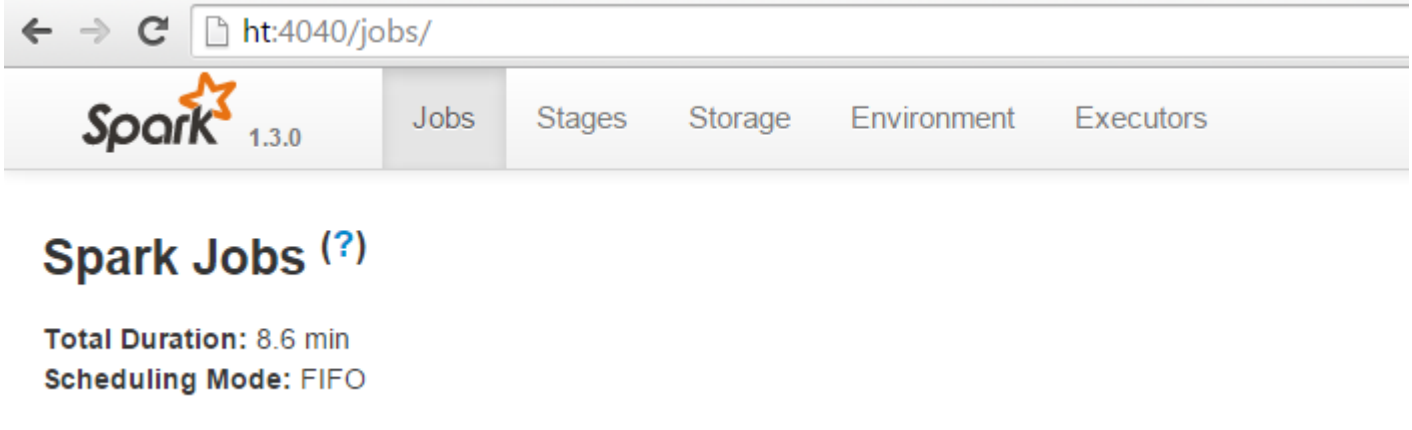
```
[root@master spark]# cd spark-2.1/
[root@master spark-2.1]# ls
bin   data      jars     licenses  python  README.md  sbin
conf  examples  LICENSE  NOTICE    R       RELEASE    yarn
[root@master spark-2.1]# bin/spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). 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…10000 integers

val data = 1 to 10000

```
scala> val data = 1 to 10000
data: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8,
 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 1
07, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 1
23, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 1
39, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 1
55, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170...

scala>
```

http://ht:4040/jobs/



6. Exploring DataFrames using pyspark – 35 Minutes

Following features of Spark will be demonstrated here:

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

Start spark-shell

#pysparkl
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)
print(users)
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.

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

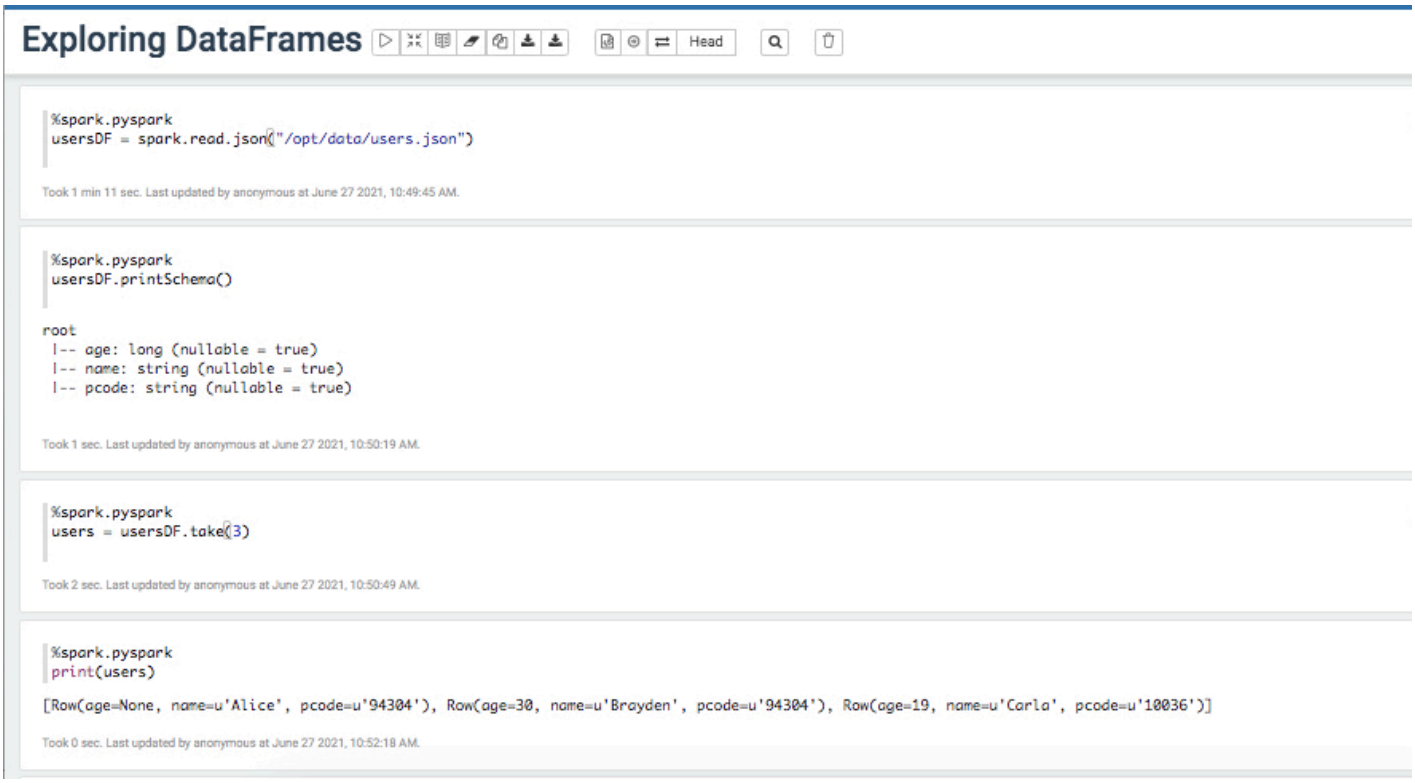
```
scala> val nameAgeDF = usersDF.select("name","age")
nameAgeDF: org.apache.spark.sql.DataFrame = [name: string, age: bigint]

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.

Zeppelin Ouput.



```
%spark.pyspark
usersDF.show()
```

```
+----+-------+-----+
| age|   name|pcode|
+----+-------+-----+
|null|  Alice|94304|
|  30|Brayden|94304|
|  19|  Carla|10036|
|  46|  Diana| null|
|null|Etienne|94104|
+----+-------+-----+
```

Took 1 sec. Last updated by anonymous at June 27 2021, 10:51:46 AM.

```
%spark.pyspark
nameAgeDF = usersDF.select("name","age")
```

Took 0 sec. Last updated by anonymous at June 27 2021, 10:53:04 AM.

```
%spark.pyspark
nameAgeOver20DF = nameAgeDF.where("age > 20")
```

Took 2 sec. Last updated by anonymous at June 27 2021, 10:53:16 AM.

```
%spark.pyspark
nameAgeOver20DF.show()
```

```
+-------+---+
|   name|age|
+-------+---+
|Brayden| 30|
|  Diana| 46|
+-------+---+
```

```
%spark.pyspark
usersDF.select("name","age").where("age > 20").show()
```

```
+-------+---+
|   name|age|
+-------+---+
|Brayden| 30|
|  Diana| 46|
+-------+---+
```

Took 1 sec. Last updated by anonymous at June 27 2021, 10:54:32 AM.

```
%spark.pyspark
```

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

7. Working with DataFrames and Schemas – 30 Minutes

You will understand the following:

- Defining schema and mapping to dataframe while loading json file.
- Saving the dataframe to json file.

Create an input text file /software/people.csv

```
pcode,lastName,firstName,age
02134,Hopper,Grace,52
94020,Turing,Alan,32
94020,Lovelace,Ada,28
87501,Babbage,Charles,49
02134,Wirth,Niklaus,48
```

Execute the following in the Spark-shell.

Import and define the Structure.

```
from pyspark.sql.types import *
columnsList = [
StructField("pcode", StringType()),
StructField("lastName", StringType()),
StructField("firstName", StringType()),
StructField("age", IntegerType())]
peopleSchema = StructType(columnsList)

// Specify the schema along with the loading instruction.
usersDF  =
spark.read.option("header","true").schema(peopleSchema).csv("people.csv")

usersDF.printSchema()
```

```
scala> val usersDF  =  spark.read.option("header","true").schema(peopleSchema).csv("people.csv")
usersDF: org.apache.spark.sql.DataFrame = [pcode: string, lastName: string ... 2 more fields]

scala> usersDF.printSchema()
root
 |-- pcode: string (nullable = true)
 |-- lastName: string (nullable = true)
 |-- firstName: string (nullable = true)
 |-- age: integer (nullable = true)
```

// As shown above, age is an Integer, which will be mapped to String by default in case its not define in the structure schema.

nameAgeDF = usersDF.select("firstname","age")
nameAgeDF.show()

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

scala> nameAgeDF.show()
20/09/24 03:06:55 WARN CSVHeaderChecker: CSV header does not conform to the schema.
 Header: LastName, age
 Schema: firstName, age
Expected: firstName but found: LastName
CSV file: file:///software/people.csv
+---------+---+
|firstName|age|
+---------+---+
|    Grace| 52|
|     Alan| 32|
|      Ada| 28|
|  Charles| 49|
|  Niklaus| 48|
+---------+---+


scala>
```

You can save the dataframe consisting of First Name and age to a file.

nameAgeDF.write.json("age.json")

Open a terminal and verify the file. It will create a folder by the name age.json and inside that output will be there.

```
[root@306633508e8b software]# more age.json/

*** age.json/: directory ***

[root@306633508e8b software]# cd age.json/
[root@306633508e8b age.json]# ls
_SUCCESS  part-00000-8c5ff49a-9c1f-4294-b9c3-7c52c4b20147-c000.json
[root@306633508e8b age.json]# more part-00000-8c5ff49a-9c1f-4294-b9c3-7c52c4b20147-c000.json
{"firstName":"Grace","age":52}
{"firstName":"Alan","age":32}
{"firstName":"Ada","age":28}
{"firstName":"Charles","age":49}
{"firstName":"Niklaus","age":48}
[root@306633508e8b age.json]#
```

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

8. Analyzing Data with DataFrame Queries – 40 Minutes

We will understand the following in this lab:

- Join
- Accessing Columns in DF

Use the following data for this lab:

/software/people.csv

```
pcode,lastName,firstName,age
02134,Hopper,Grace,52
94020,Turing,Alan,32
94020,Lovelace,Ada,28
87501,Babbage,Charles,49
02134,Wirth,Niklaus,48
```

Load the people data and fetch column, age in the following way:

```
peopleDF = spark.read.option("header","true").csv("people.csv")
peopleDF["age"]
peopleDF.age
peopleDF.select(peopleDF["age"]).show()
```

```
scala> val peopleDF = spark.read.option("header","true").csv("people.csv")
peopleDF: org.apache.spark.sql.DataFrame = [pcode: string, lastName: string ... 2 more

scala> peopleDF("age")
res8: org.apache.spark.sql.Column = age

scala> $"age"
res9: org.apache.spark.sql.ColumnName = age

scala> peopleDF.select(peopleDF("age")).show()
+---+
|age|
+---+
| 52|
| 32|
| 28|
| 49|
| 48|
+---+
```

Manipulate the column age i.e multiple age by 10.

peopleDF.select("lastName", peopleDF.age * 10).show()

```
scala> peopleDF.select($"lastName", $"age" * 10).show
+--------+----------+
|lastName|(age * 10)|
+--------+----------+
|  Hopper|     520.0|
|  Turing|     320.0|
|Lovelace|     280.0|
| Babbage|     490.0|
|   Wirth|     480.0|
+--------+----------+
```

You can chain the Queries.

peopleDF.select("lastName",(peopleDF.age * 10).alias("age_10")).show()

Perform aggregration

peopleDF.groupBy("pcode").count().show()

```
scala> peopleDF.select($"lastName",($"age" * 10).alias("age_10")).show()
+--------+------+
|lastName|age_10|
+--------+------+
|  Hopper| 520.0|
|  Turing| 320.0|
|Lovelace| 280.0|
| Babbage| 490.0|
|   Wirth| 480.0|
+--------+------+


scala> peopleDF.groupBy("pcode").count().show()
+-----+-----+
|pcode|count|
+-----+-----+
|87501|    1|
|94020|    2|
|02134|    2|
+-----+-----+
```

Next let us join two dataframes:

/software/people-no-pcode.csv

```
pcode,lastName,firstName,age
02134,Hopper,Grace,52
,Turing,Alan,32
94020,Lovelace,Ada,28
87501,Babbage,Charles,49
02134,Wirth,Niklaus,48
```

/software/pcodes.csv

```
pcode,city,state
02134,Boston,MA
94020,Palo Alto,CA
87501,Santa Fe,NM
60645,Chicago,IL
```

Load the people and code files in DF.

```
peopleDF = spark.read.option("header","true").csv("people-no-pcode.csv")
pcodesDF = spark.read.option("header","true").csv("pcodes.csv")
```

Peform Inner Join on pcode.

```
peopleDF.join(pcodesDF, "pcode").show()
```

```
scala> val peopleDF = spark.read.option("header","true").csv("people-no-pcode.csv")
peopleDF: org.apache.spark.sql.DataFrame = [pcode: string, lastName: string ... 2 more fields]

scala> peopleDF.join(pcodesDF, "pcode").show()
+-----+--------+---------+---+---------+-----+
|pcode|lastName|firstName|age|     city|state|
+-----+--------+---------+---+---------+-----+
|02134|  Hopper|    Grace| 52|   Boston|   MA|
|94020|Lovelace|      Ada| 28|Palo Alto|   CA|
|87501| Babbage|  Charles| 49| Santa Fe|   NM|
|02134|   Wirth|  Niklaus| 48|   Boston|   MA|
+-----+--------+---------+---+---------+-----+


scala>
```

Perform the Left outer join

peopleDF.join(pcodesDF,peopleDF["pcode"] == pcodesDF.pcode, "left_outer").show()

You can see null value in the pcode of the second row.

Joining on Columns with Different Names

/software/zcodes.csv

```
zip,city,state
02134,Boston,MA
94020,Palo Alto,CA
87501,Santa Fe,NM
60645,Chicago,IL
```

Join with the pcode and zip of the second file.

```
zcodesDF = spark.read.option("header","true").csv("zcodes.csv")
peopleDF.join(zcodesDF, peopleDF.pcode == zcodesDF.zip).show()
```

```
scala> peopleDF.join(zcodesDF, $"pcode" === $"zip").show
+-----+--------+---------+---+-----+---------+-----+
|pcode|lastName|firstName|age|  zip|     city|state|
+-----+--------+---------+---+-----+---------+-----+
|02134|  Hopper|    Grace| 52|02134|   Boston|   MA|
|94020|Lovelace|      Ada| 28|94020|Palo Alto|   CA|
|87501| Babbage|  Charles| 49|87501| Santa Fe|   NM|
|02134|   Wirth|  Niklaus| 48|02134|   Boston|   MA|
+-----+--------+---------+---+-----+---------+-----+
```

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

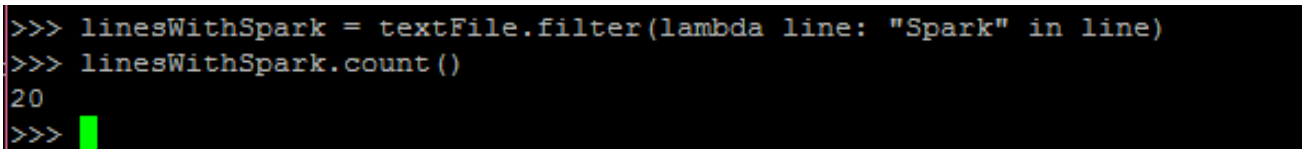
9. Interactive Analysis with pySpark – 30 Minutes

Optional : `export PYSPARK_PYTHON=python3.6 (In case you want to change the python version. Update in the bin\pyspark scripts of load environment sh file)`

Start it by running the following in the Spark directory:

./pyspark

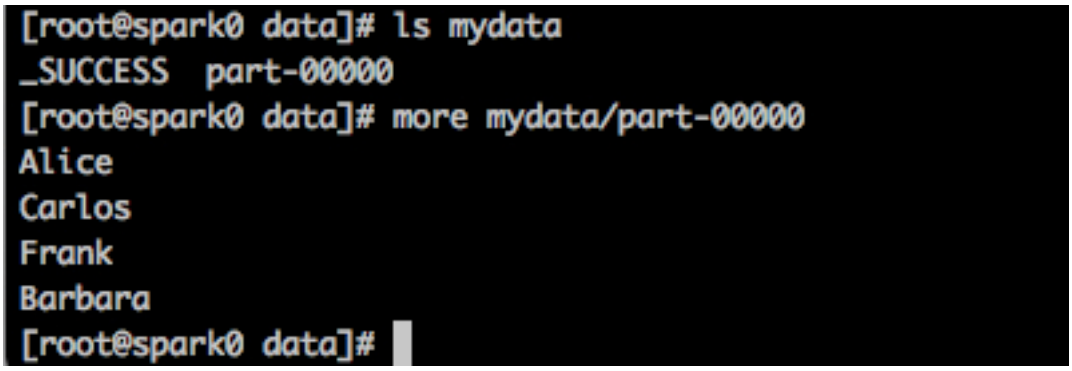
```
textFile = sc.textFile("README.md")
textFile.count()
textFile.first()
linesWithSpark = textFile.filter(lambda line: "Spark" in line)
linesWithSpark.count()
```



Create RDD from Collection.

```
myData = ["Alice","Carlos","Frank","Barbara"]
myRDD = sc.parallelize(myData)

for make in myRDD.collect(): print make
for make in myRDD.take(2): print make
myRDD.saveAsTextFile("mydata/")
```

You can verify the data in the folder mydata/



Add one more RDD
myData1 = ["Ram","Shyam","Krishna","Vishnu"]
myRDD1 = sc.parallelize(myData1)

create a new RDD by combining the second RDD to the previous RDD.

myNames = myRDD.union(myRDD1)

Collect and display the contents of the new myNames RDD.

for name in myNames.collect(): print name

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

10. Transformation and Action – RDD – 45 Minutes

Let's make a new RDD from the text of the README file in the Spark source directory: (SPARK_HOME/README.md)

```
textFile = sc.textFile("README.md")
```

```
scala> val textFile  = sc.textFile("/MyTrainingWork/Spark/README.md")
15/06/03 23:32:06 INFO MemoryStore: ensureFreeSpace(73391) called with curMem=18
1810, maxMem=278019440
15/06/03 23:32:06 INFO MemoryStore: Block broadcast_1 stored as values in memory
 (estimated size 71.7 KB, free 264.9 MB)
15/06/03 23:32:06 INFO MemoryStore: ensureFreeSpace(31262) called with curMem=25
5201, maxMem=278019440
15/06/03 23:32:06 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in
memory (estimated size 30.5 KB, free 264.9 MB)
15/06/03 23:32:06 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on l
ocalhost:1599 (size: 30.5 KB, free: 265.1 MB)
15/06/03 23:32:06 INFO BlockManagerMaster: Updated info of block broadcast_1_pie
ce0
15/06/03 23:32:06 INFO SparkContext: Created broadcast 1 from textFile at <conso
le>:21
textFile: org.apache.spark.rdd.RDD[String] = /MyTrainingWork/Spark/README.md Map
PartitionsRDD[3] at textFile at <console>:21
```

Let's start with a few actions:

```
textFile.count() // Number of items in this RDD
```

```
scala> textFile.count()
15/06/03 23:32:16 INFO FileInputFormat: Total input paths to process : 1
15/06/03 23:32:16 INFO SparkContext: Starting job: count at <console>:24
15/06/03 23:32:16 INFO DAGScheduler: Got job 0 (count at <console>:24) with 2 ou
tput partitions (allowLocal=false)
15/06/03 23:32:16 INFO DAGScheduler: Final stage: Stage 0(count at <console>:24)

15/06/03 23:32:16 INFO DAGScheduler: Parents of final stage: List()
15/06/03 23:32:16 INFO DAGScheduler: Missing parents: List()
15/06/03 23:32:16 INFO DAGScheduler: Submitting Stage 0 (/MyTrainingWork/Spark/R
EADME.md MapPartitionsRDD[3] at textFile at <console>:21), which has no missing
parents
tes result sent to driver
15/06/03 23:32:17 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in
 352 ms on localhost (1/2)
15/06/03 23:32:17 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in
 338 ms on localhost (2/2)
15/06/03 23:32:17 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have
all completed, from pool
15/06/03 23:32:17 INFO DAGScheduler: Stage 0 (count at <console>:24) finished in
 0.393 s
15/06/03 23:32:17 INFO DAGScheduler: Job 0 finished: count at <console>:24, took
 0.678076 s
res2: Long = 98

scala>
```

```
textFile.first() // First item in this RDD
```

```
md:0+1814
15/06/03 23:34:06 INFO Executor: Finished task 0.0 in stage 1.0 (TID 2). 1809 by
tes result sent to driver
15/06/03 23:34:06 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in
 10 ms on localhost (1/1)
15/06/03 23:34:06 INFO DAGScheduler: Stage 1 (first at <console>:24) finished in
 0.011 s
15/06/03 23:34:06 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have
all completed, from pool
15/06/03 23:34:06 INFO DAGScheduler: Job 1 finished: first at <console>:24, took
 0.019875 s
res3: String = # Apache Spark

scala>
```

Now let's use a transformation. We will use the filter transformation to return a new RDD with a subset of the items in the file.

```
linesWithSpark = textFile.filter(lambda line: "Spark" in line)
linesWithSpark.collect()
```

We can chain together transformations and actions:

```
textFile.filter(lambda line : "Spark" in line).count() // How many lines contain "Spark"?
```

```
15/06/03 23:35:17 INFO Executor: Finished task 0.0 in stage 2.0 (TID 3). 1830 by
tes result sent to driver
15/06/03 23:35:17 INFO TaskSetManager: Finished task 1.0 in stage 2.0 (TID 4) in
 12 ms on localhost (1/2)
15/06/03 23:35:17 INFO TaskSetManager: Finished task 0.0 in stage 2.0 (TID 3) in
 14 ms on localhost (2/2)
15/06/03 23:35:17 INFO DAGScheduler: Stage 2 (count at <console>:24) finished in
 0.014 s
15/06/03 23:35:17 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have
all completed, from pool
15/06/03 23:35:17 INFO DAGScheduler: Job 2 finished: count at <console>:24, took
 0.028214 s
res4: Long = 19

scala>
```

Let's say we want to find the line with the most words

```
textFile.map(lambda line : len(line.split(" "))). reduce(lambda a, b: a if (a > b) el
se b)
```

```
calhost, PROCESS_LOCAL, 1300 bytes)
15/06/03 23:36:03 INFO Executor: Running task 0.0 in stage 3.0 (TID 5)
15/06/03 23:36:03 INFO Executor: Running task 1.0 in stage 3.0 (TID 6)
15/06/03 23:36:03 INFO HadoopRDD: Input split: file:/MyTrainingWork/Spark/README
.md:0+1814
15/06/03 23:36:03 INFO HadoopRDD: Input split: file:/MyTrainingWork/Spark/README
.md:1814+1815
15/06/03 23:36:03 INFO Executor: Finished task 1.0 in stage 3.0 (TID 6). 1908 by
tes result sent to driver
15/06/03 23:36:03 INFO Executor: Finished task 0.0 in stage 3.0 (TID 5). 1908 by
tes result sent to driver
15/06/03 23:36:03 INFO TaskSetManager: Finished task 1.0 in stage 3.0 (TID 6) in
 15 ms on localhost (1/2)
15/06/03 23:36:03 INFO TaskSetManager: Finished task 0.0 in stage 3.0 (TID 5) in
 17 ms on localhost (2/2)
15/06/03 23:36:03 INFO DAGScheduler: Stage 3 (reduce at <console>:24) finished i
n 0.019 s
15/06/03 23:36:03 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have
all completed, from pool
15/06/03 23:36:03 INFO DAGScheduler: Job 3 finished: reduce at <console>:24, too
k 0.033852 s
res5: Int = 14

scala>
```

One common data flow pattern is MapReduce, as popularized by Hadoop. Spark can implement MapReduce flows easily:

```
wordCounts = textFile.flatMap(lambda line: line.split()).map(lambda word: (w
ord, 1)).reduceByKey(lambda a, b: a+b)
```

```
15/06/03 23:37:42 INFO ContextCleaner: Cleaned broadcast 6
15/06/03 23:37:42 INFO BlockManager: Removing broadcast 5
15/06/03 23:37:42 INFO BlockManager: Removing block broadcast_5_piece0
15/06/03 23:37:42 INFO MemoryStore: Block broadcast_5_piece0 of size 2163 droppe
d from memory (free 277729945)
15/06/03 23:37:42 INFO BlockManagerInfo: Removed broadcast_5_piece0 on localhost
:1599 in memory (size: 2.1 KB, free: 265.1 MB)
15/06/03 23:37:42 INFO BlockManagerMaster: Updated info of block broadcast_5_pie
ce0
15/06/03 23:37:42 INFO BlockManager: Removing block broadcast_5
15/06/03 23:37:42 INFO MemoryStore: Block broadcast_5 of size 3032 dropped from
memory (free 277732977)
15/06/03 23:37:42 INFO ContextCleaner: Cleaned broadcast 5
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[10] at reduceB
yKey at <console>:24

scala>
```

Here, we combined the flatMap, map and reduceByKey transformations to compute the per-word counts in the file as an RDD of (String, Int) pairs. To collect the word counts in our shell, we can use the collect action:

```
wordCounts.collect()
```

let's mark our linesWithSpark dataset to be cached:

```
15/06/03 23:39:02 INFO TaskSchedulerImpl: Removed TaskSet 6.0, whose tasks have
all completed, from pool
15/06/03 23:39:02 INFO DAGScheduler: Job 5 finished: collect at <console>:27, to
ok 0.346996 s
res7: Array[(String, Int)] = Array((package,1), (this,1), (Because,1), (Python,2
), (cluster.,1), (its,1), (run,1), (general,2), (YARN,,1), (have,1), (pre-built
,1), (locally.,1), (changed,1), (locally,2), (sc.parallelize(1,1), (only,1), (se
veral,1), (This,2), (basic,1), (first,1), (documentation,3), (Configuration,1),
(learning,,1), (graph,1), (Hive,2), (["Specifying,1), ("yarn-client",1), (page](
http://spark.apache.org/documentation.html),1), ([params]`.,1), (application,1),
 ([project,2), (prefer,1), (SparkPi,2), ((<http://spark.apache.org/>),1), (engine,
1), (version,1), (file,1), (documentation,,1), (MASTER,1), (example,3), (are,1),
 (systems.,1), (params,1), (scala>,1), (provides,1), (refer,2), (configure,1), (
Interactive,2), (distribution.,1), (can,6), (build,3), (when,1), (Apache,1), ...

scala>
```

```
linesWithSpark.cache()
linesWithSpark.count()
```

```
all completed, from pool
15/06/03 23:39:46 INFO DAGScheduler: Stage 7 (count at <console>:27) finished in
 0.056 s
15/06/03 23:39:46 INFO DAGScheduler: Job 6 finished: count at <console>:27, took
 0.070778 s
res9: Long = 19

scala>
```

Next, we will convert RDD to Dataframe.

Create a file /software/people.txt

```
02134,Hopper,Grace,52
94020,Turing,Alan,32
94020,Lovelace,Ada,28
87501,Babbage,Charles,49
02134 Wirth Niklaus 48
```

```
from pyspark.sql.types import *

from pyspark.sql import Row

rowRDD = sc.textFile("/opt/data/people.txt").map(lambda line :
line.split(",")).map(lambda values :
Row(values[0],values[1],values[2],values[3]))

// Display two records of the RDD.
rowRDD.take(2)


// Convert the RDD to data Frame.
myDF = rowRDD.toDF()

// Display two records of the data frame.
myDF.show(2)
```

```
scala> import org.apache.spark.sql.types._
import org.apache.spark.sql.types._

scala>  import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala> val mySchema = StructType(Array(  StructField("pcode", StringType),  StructField("lastName", StringType),  Struct
Field("firstName", StringType),  StructField("age", IntegerType)
     | ))
mySchema: org.apache.spark.sql.types.StructType = StructType(StructField(pcode,StringType,true), StructField(lastName,St
ringType,true), StructField(firstName,StringType,true), StructField(age,IntegerType,true))

scala> val rowRDD = sc.textFile("people.txt").map(line => line.split(",")).map(values =>
     | Row(values(0),values(1),values(2),values(3).toInt))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[12] at map at <console>:29

scala>

scala> val myDF = spark.createDataFrame(rowRDD,mySchema)
myDF: org.apache.spark.sql.DataFrame = [pcode: string, lastName: string ... 2 more fields]

scala> myDF.show(2)
```

Profile: Default
Command: /Users/henrypotsangbam/Applications/iTerm.app/Contents/MacOS/iTerm2

```
scala> myDF.show(2)
+-----+--------+---------+---+
|pcode|lastName|firstName|age|
+-----+--------+---------+---+
|02134|  Hopper|    Grace| 52|
|94020|  Turing|     Alan| 32|
+-----+--------+---------+---+
only showing top 2 rows
```

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

11. Work with Pair RDD – 30 Minutes

Lab Overview

In this activity, you will load SFPD data from a CSV file. You will create pair RDD and apply pair RDD operations to explore the data.

Scenario

Our dataset is a .csv file that consists of SFPD incident data from SF OpenData (https://data.sfgov.org/). For each incident, we have the following information:

Field	Description	Example Value
IncidentNum	Incident number	150561637
Category	Category of incident	NON-CRIMINAL
Descript	Description of incident	FOUND_PROPERTY
DayOfWeek	Day of week that incident occurred	Sunday
Date	Date of incident	6/28/15
Time	Time of incident	23:50
PdDistrict	Police Department District	TARAVAL

Resolution	Resolution	NONE
Address	Address	1300_Block_of_LA_PLAYA_ST
X	X-coordinate of location	-122.5091348
Y	Y-coordinate of location	37.76119777
PdID	Department ID	15056163704013

The dataset has been modified to decrease the size of the files and also to make it easier to use. We use this same dataset for all the labs in this course.

Load Data into Apache Spark

Objectives

- Launch the Spark interactive shell
- Load data into Spark
- Explore data in Apache Spark

4.1.1 Launch the Spark Interactive Shell

The Spark interactive shell is available in Scala or Python.

1. To launch the Interactive Shell, at the command line, run the following command:

```
pyspark --master local[2]
```

Note: To quit the Scala Interactive shell, use the command

```
Exit
```

4.1.2. Load Data into Spark

(copy all data files in sparkcluster)

To load the data we are going to use the `SparkContext` method `textFile`. The SparkContext is available in the interactive shell as the variable **`sc`**. We also want to split the file by the separator ",".

1. We define the mapping for our input variables. While this isn't a necessary step, it makes it easier to refer to the different fields by names.

```
val IncidntNum = 0
val Category = 1
val Descript = 2
val DayOfWeek = 3
val Date = 4
val Time = 5
val PdDistrict = 6
val Resolution = 7
val Address = 8
val X = 9
val Y = 10
val PdId = 11
```

2. To load data into Spark, at the Scala command prompt:

```
sfpdRDD =sc.textFile("sfpd.csv").map(_.split(","))
```

Lab 4.1.3 Explore data using RDD operations

What transformations and actions would you use in each case? Complete the command with the appropriate transformations and actions.

1. How do you see the first element of the inputRDD (sfpdRDD)?

```
sfpdRDD.first()
```

```
scala> sfpdRDD.first()
res4: Array[String] = Array(150599321, OTHER_OFFENSES, POSSESSION_OF_BURGLARY_TO
OLS, Thursday, 7/9/15, 23:45, CENTRAL, ARREST/BOOKED, JACKSON_ST/POWELL_ST, -122
.4099006, 37.79561712, 15059900000000)
```

2. What do you use to see the first 5 elements of the RDD?

```
sfpdRDD.take(5)
```

```
scala> sfpdRDD.take(5)
res6: Array[Array[String]] = Array(Array(150599321, OTHER_OFFENSES, POSSESSION_O
F_BURGLARY_TOOLS, Thursday, 7/9/15, 23:45, CENTRAL, ARREST/BOOKED, JACKSON_ST/PO
WELL_ST, -122.4099006, 37.79561712, 15059900000000), Array(156168837, LARCENY/TH
EFT, PETTY_THEFT_OF_PROPERTY, Thursday, 7/9/15, 23:45, CENTRAL, NONE, 300_Block_
of_POWELL_ST, -122.4083843, 37.78782711, 15616900000000), Array(150599321, OTHER
_OFFENSES, DRIVERS_LICENSE/SUSPENDED_OR_REVOKED, Thursday, 7/9/15, 23:45, CENTRA
L, ARREST/BOOKED, JACKSON_ST/POWELL_ST, -122.4099006, 37.79561712, 1505990000000
0), Array(150599224, OTHER_OFFENSES, DRIVERS_LICENSE/SUSPENDED_OR_REVOKED, Thurs
day, 7/9/15, 23:36, PARK, ARREST/BOOKED, MASONIC_AV/GOLDEN_GATE_AV, -122.4468469
, 37.77766882, 15059900000000), Array(156169067, LARCENY/THEFT, GRAND_THEFT_F...
```

3. What is the total number of incidents?

totincs = sfpdRDD.count()
print(totincs)

```
scala> sfpdRDD.count()
res8: Long = 383775
```

---

4. What is the total number of distinct resolutions?

totres = sfpdRDD.map(lambda inc : inc[7]).distinct().count()
totres

```
scala> sfpdRDD.map(inc=>inc(7)).distinct.count()
res11: Long = 17
```

---

5. List the distinct PdDistricts.

totresdistricts =sfpdRDD.map(lambda inc : inc[6]).collect()

totresdistricts

```
scala> sfpdRDD.map(inc=>inc(6)).distinct.collect()
res14: Array[String] = Array(INGLESIDE, SOUTHERN, PARK, NORTHERN, MISSION, RICHM
OND, TENDERLOIN, BAYVIEW, TARAVAL, CENTRAL)
```

---

12. Create & Explore PairRDD – 60 Minutes

In the previous activity we explored the data in the sfpdRDD. We used RDD operations. In this activity, we will create pairRDD to find answers to questions about the data.

Objectives

- Create pairRDD & apply pairRDD operations
- Join pairRDD

Create pair RDD & apply pair RDD operations

Start the spark shell if not done else skip the next command.

#pyspark

sfpdRDD = sc.textFile("sfpd.csv").map(lambda rec: rec.split(","))

1. Which five districts have the highest incidents?

Note: This is similar to doing a word count. First, use the **`map`** transformation to create a pair RDD with the key being the field on which you want to count. In this case, the key would be PdDistrict and the value is "1" for each count of the district.

To get the total count, use **`reduceByKey((a,b)=>a+b)`**.

To find the **top** 5, you have to do a sort in descending. However, sorting on the result of the reduceByKey will sort on the District rather than the count. Apply the **`map`** transformation to create a pairRDD with the count being the key and the value being the district.

Then use **`sortByKey`**(false) to specify descending order. To get the top 5, use **`take(5)`**.

Note that this is one way of doing it. There may be other ways of achieving the same result.

Step to be follow (Hints)

a. Use a **`map`** transformation to create pair RDD from sfpdRDD of the form: [(PdDistrict, 1)]

b. Use **`reduceByKey((a,b)=>a+b)`** to get the count of incidents in each district. Your result will be a pairRDD of the form [(PdDistrict, count)]

c. Use map again to get a pairRDD of the form [(count, PdDistrict)]

d. Use **`sortByKey(false)`** on [(count, PdDistrict)]

e. Use **`take(5)`** to get the top 5.

Answer:

top5Dists = sfpdRDD.filter(lambda incident: len(incident) == 14).map(lambda incident: (incident[6],len(incident))).reduceByKey(lambda x,y: x+y).map(lambda (city , inc) : (inc, city)).sortByKey(False).take(3)

```
scala> val top5Dists = sfpdRDD.map(incident=>(incident(6),1)).reduceByKey((x,y)
     | =>x+y).map(x=>(x._2,x._1)) .sortByKey(false).take(3)
top5Dists: Array[(Int, String)] = Array((73308,SOUTHERN), (50164,MISSION), (4687
7,NORTHERN))
```

Add filter to validate proper record i.e it should have exactly 14 fields only in each record.

2. Which five addresses have the highest incidents?
   a. Create pairRDD (map)
   b. Get the count for key (reduceByKey)
   c. Pair RDD with key and count switched (map
   d. Sort in descending order (sortByKey
   e. Use **`take(5)`** to get the top 5

Answer:

top5Adds = sfpdRDD.filter(lambda incident: len(incident) == 14).map(lambda incident : (incident[8],1)).reduceByKey(lambda x,y : x

+y).map(lambda (ad,count) : (count,ad)).sortByKey(False).take(5)

top5Adds

```
scala> val top5Adds = sfpdRDD.map(incident=>(incident(8),1)).reduceByKey((x,y)=>
x
     | +y).map(x=>(x._2,x._1)).sortByKey(false).take(5)
top5Adds: Array[(Int, String)] = Array((10852,800_Block_of_BRYANT_ST), (3671,800
_Block_of_MARKET_ST), (2027,1000_Block_of_POTRERO_AV), (1585,2000_Block_of_MISSI
ON_ST), (1512,16TH_ST/MISSION_ST))

scala>
```

3. What are the top **three** categories of incidents?

   Answer:

   top3Cat = sfpdRDD.filter(lambda incident: len(incident) == 14).map(lambda incident: (incident[1],1)).reduceByKey(lambda x,y : x+y).map(lambda (cat,count) : (count,cat)).sortByKey(False).take(3)

   top3Cat

```
scala> val top3Cat = sfpdRDD.map(incident=>(incident(1),1)).reduceByKey((x,y)=>
x+y).map(x=>(x._2,x._1)) .sortByKey(false).take(3)
top3Cat: Array[(Int, String)] = Array((96955,LARCENY/THEFT), (50611,OTHER_OFFENS
ES), (50269,NON-CRIMINAL))
```

4. . What is the count of incidents by district?

Answer:

num_inc_dist = sfpdRDD.filter(lambda incident: len(incident) == 14).map(lambda incident:(incident[6],1)).countByKey()

num_inc_dist

```
scala> val num_inc_dist = sfpdRDD.map(incident=>(incident(6),1)).countByKey()
num_inc_dist: scala.collection.Map[String,Long] = Map(SOUTHERN -> 73308, INGLESI
DE -> 33159, TENDERLOIN -> 30174, MISSION -> 50164, TARAVAL -> 27470, RICHMOND -
> 21221, NORTHERN -> 46877, PARK -> 23377, CENTRAL -> 41914, BAYVIEW -> 36111)
```

Caution! For large datasets, don't use countByKey.

Join Pair RDDs - TBD

This activity illustrates how joins work in Spark (python). There are two small datasets provided for this activity - J_AddCat.csv and J_AddDist.csv.

J_AddCat.csv - Category; Address			J_AddDist.csv - PdDistrict; Address		
1.	EMBEZZLEMENT	100_Block_of_JEFFERSON_ST	1.	INGLESIDE	100_Block_of_ROME_ST

2.	EMBEZZLEMENT	SUTTER_ST/MASON_ST	2.	SOUTHERN	0_Block_of_SOUTHPARK_AV
3.	BRIBERY	0_Block_of_SOUTHPARK_AV	3.	MISSION	1900_Block_of_MISSION_ST
4.	EMBEZZLEMENT	1500_Block_of_15TH_ST	4.	RICHMOND	1400_Block_of_CLEMENT_ST
5.	EMBEZZLEMENT	200_Block_of_BUSH_ST	5.	SOUTHERN	100_Block_of_BLUXOME_ST
6.	BRIBERY	1900_Block_of_MISSION_ST	6.	SOUTHERN	300_Block_of_BERRY_ST
7.	EMBEZZLEMENT	800_Block_of_MARKET_ST	7.	BAYVIEW	1400_Block_of_VANDYKE_AV
8.	EMBEZZLEMENT	2000_Block_of_MARKET_ST	8.	SOUTHERN	1100_Block_of_MISSION_ST
9.	BRIBERY	1400_Block_of_CLEMENT_ST	9.	TARAVAL	0_Block_of_CHUMASERO_DR
10.	EMBEZZLEMENT	1600_Block_of_FILLMORE_ST			

11.	EMBEZZLEMENT	1100_Block_of_SELBY_ST			
12.	BAD CHECKS	100_Block_of_BLUXOME_ST			

Based on the data above, answer the following questions:

1. Given these two datasets, you want to find the type of incident and district for each address. What is one way of doing this? (HINT: An operation on pairs or pairRDDs)

What should the keys be for the two pairRDDs?

[You can use joins on pairs or pairRDD to get the information. The key for the pairRDD is the address.]

2. What is the size of the resulting dataset from a join? Why?

Note: Look at the datasets above and estimate what a join on the two would return if they were joined on the key - from #1.

Remember that a join is the same as an inner join and only keys that are present in both RDDs are output.

[A join is the same as an inner join and only keys that are present in both RDDs are output. If you compare the addresses in both the datasets, you find that there are five addresses in common and

they are unique. Thus the resulting dataset will contain five elements. If there are multiple values for the same key, the resulting RDD will have an entry for every possible pair of values with that key from the two RDDs.]

3. If you did a right outer join on the two datasets with Address/Category being the source RDD,

Note: Look at the datasets above and estimate what a join on the two would return if they were joined on the key - from #1.

Remember that a right outer join results in a pair RDD that has entries for each key in the other pair RDD.

what would be the size of the resulting dataset? Why?

[A right outer join results in a pair RDD that has entries for each key in the other pairRDD. If the source RDD contains data from J_AddCat.csv and the “other” RDD is represented by J_AddDist.csv, then since “other” RDD has 9 distinct addresses, the size of the result of a right outer join is 9.]

4. If you did a left outer join on the two datasets with Address/Category being the source RDD,

Note: Look at the datasets above and estimate what a join on the two would return if they were joined on the key - from #1.

Remember that a left outer join results in a pair RDD that has entries for each key in the source pair RDD.

what would be the size of the resulting dataset? Why?

[A left outer join results in a pair RDD that has entries for each key in the source pairRDD. If thesource RDD contains data from J_AddCat.csv and the "other" RDD is represented by J_AddDist.csv,then since "source" RDD has 13 distinct addresses, the size of the result of a left outer join is 13.]

5. Load each dataset into separate pairRDDs with "address" being the key.

Note: once you load the text file, split on the "," and then apply the `map` transformation to create a pairRDD where address is the first element of the two-tuple.

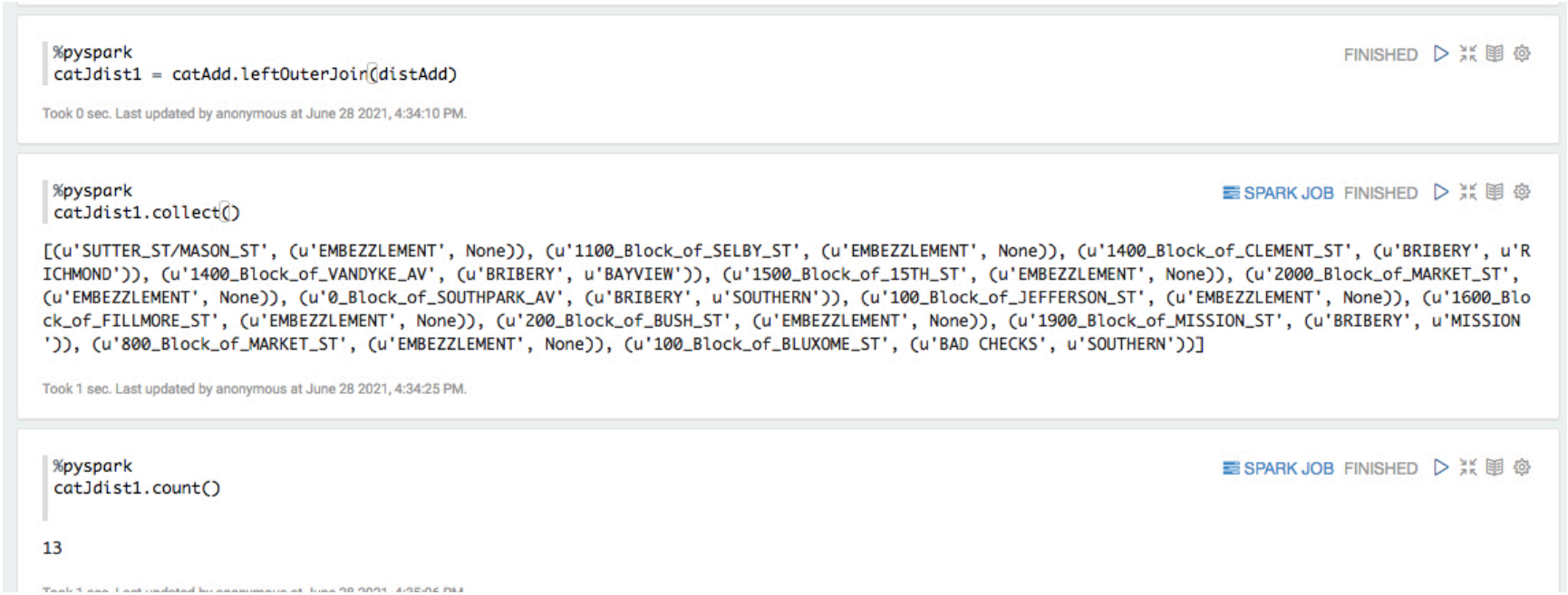
```
# catAdd = sc.textFile("/opt/data/J_AddCat.csv").map(lambda x : x.split(",")).map(lambda x : (x[1],x[0]))
# distAdd = sc.textFile("/opt/data/J_AddDist.csv").map(lambda x : x.split(",")).map(lambda x : (x[1],x[0]))
```

6. List the incident category and district for those addresses that have both category and district information. Verify that the size estimated earlier is correct.

```
catJdist = catAdd.join(distAdd)
catJdist.collect()
catJdist.count()
catJdist.take(5)
```

7. List the incident category and district for all addresses irrespective of whether each address has category and district information.

```
catJdist1 = catAdd.leftOuterJoin(distAdd)
catJdist1.collect()
catJdist1.count()
```



8. List the incident district and category for all addresses irrespective of whether each address has category and district information. Verify that the size estimated earlier is correct.

```
catJdist2 = catAdd.rightOuterJoin(distAdd)
catJdist2.collect()
catJdist2.count()
```

Explore Partitioning

In this activity we see how to determine the number of partitions, the type of partitioner and how to specify partitions in a transformation.

Objective

- Explore partitioning in RDDs

Explore partitioning in RDDs

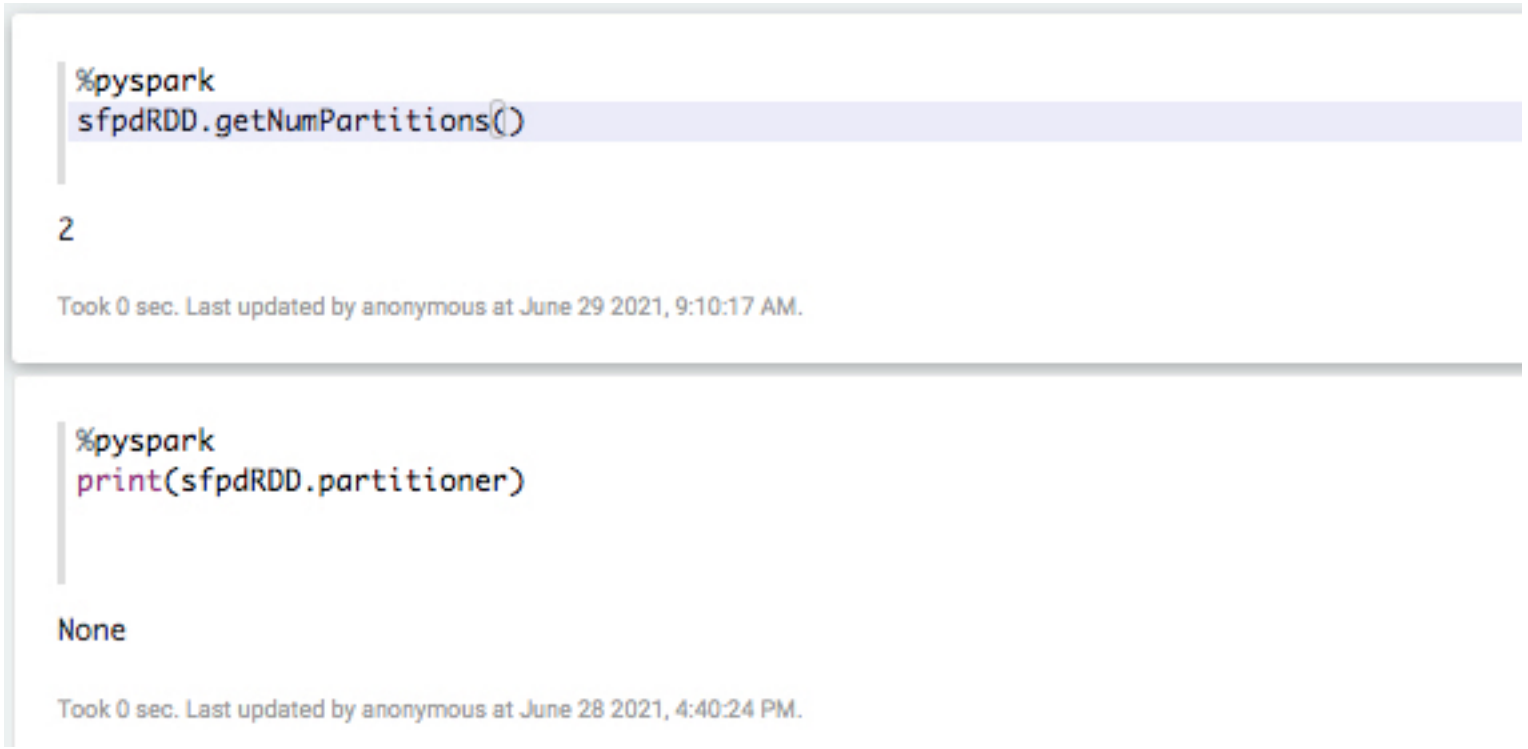
Note To find partition size: rdd.partitions.size (Scala); rdd.getNumPartitions() (in Python)

To determine the partitioner: rdd.partitioner (Scala);

Note: Ensure that you have started the spark shell with --master local[*] (Refer the First Lab)

1. How many partitions are there in the sfpdRDD?
   sfpdRDD.getNumPartitions()

2. How do you find the type of partitioner for sfpdRDD?
   `sfpdRDD.partitioner`



The above result will depend on the no of cores. In my case its 2 cores so 2.
If there is **no partitioner** the partitioning is not based upon characteristic of data but distribution is random and uniformed across nodes.

3. Create a pair RDD when only the length of the items is 14 per record that its 14 fields.

```
incByDists = sfpdRDD.filter(lambda rec : len(rec) == 14 ).map(lambda incident :
(incident(6),1)).reduceByKey(lambda x,y : x+y)
```

How many partitions does incByDists have?

```
incByDists.getNumPartitions()
```

What type of partitioner does incByDists have?

```
print(incByDists.partitioner)
```

Q: Why does incByDists have that partitioner?

A: reduceByKey automatically uses the Hash Partitioner

By default PySpark implementation uses *hash partitioning* as the partitioning function.

```
%pyspark
incByDists.getNumPartitions()
```

```
2
```

Took 0 sec. Last updated by anonymous at June 29 2021, 9:18:33 AM.

```
%pyspark
print(incByDists.partitioner)
```

```
<pyspark.rdd.Partitioner object at 0x7f5acb8ac0d0>
```

Took 0 sec. Last updated by anonymous at June 29 2021, 9:24:51 AM.

4. Add a map

   inc_map = incByDists.map(lambda (incidence, district) : (district,incidence))

   How many partitions does inc_map have? **`inc_map.getNumPartitions()`**

   What type of partitioner does incByDists have? print(incByDists.partitioner)

Q: Why does inc_map not have the same partitioner as its parent?

A: Transformations such as map() cause the new RDD to forget the parent's partitioning information because a map() can modify the key of each record.

```
%pyspark
inc_map = incByDists.map(lambda (incidence, district) : (district,incidence))
```

Took 0 sec. Last updated by anonymous at June 29 2021, 9:35:33 AM.

```
%pyspark
inc_map.getNumPartitions()
```

```
2
```

Took 0 sec. Last updated by anonymous at June 29 2021, 9:35:37 AM.

```
%pyspark
print(inc_map.partitioner)
```

```
None
```

Took 0 sec. Last updated by anonymous at June 29 2021, 9:36:06 AM.

5. Add groupByKey

```
inc_group = sfpdRDD.map(lambda incident: (incident[6],1)).groupByKey()
```

What type of partitioner does inc_group have? inc_group.partitioner

Q: Why does inc_group have this type of partitioner?

A: This is because groupByKey will automatically result in a hash partitioned RDD..

```
%pyspark
inc_group = sfpdRDD.map(lambda incident: (incident[6],1)).groupByKey()
```

Took 1 sec. Last updated by anonymous at June 28 2021, 4:52:39 PM.

```
%pyspark
inc_group.partitioner
```

```
<pyspark.rdd.Partitioner object at 0x7f42a6f8d8d0>
```

Took 0 sec. Last updated by anonymous at June 28 2021, 4:55:45 PM.

6. Create two pairRDD

```
catAdd = sc.textFile("/opt/data/J_AddCat.csv").map(lambda x : x.split(",")).map(lambda x :(x[1],x[0]))
distAdd = sc.textFile("/opt/data/J_AddDist.csv").map(lambda x : x.split(",")).map(lambda x : (x[1],x[0]))
```

7. You can specify the number of partitions when you use the join operation.

```
catJdist = catAdd.join(distAdd,8)
```

How many partitions does the joined RDD have? catJdist.getNumPartitions()

What type of partitioner does catJdist have? catJdist.partitioner

```
%pyspark
catAdd = sc.textFile("/opt/data/J_AddCat.csv").map(lambda x : x.split(",")).map(lambda x :(x[1],x[0]))
distAdd = sc.textFile("/opt/data/J_AddDist.csv").map(lambda x : x.split(",")).map(lambda x : (x[1],x[0]))
```

Took 0 sec. Last updated by anonymous at June 28 2021, 4:58:42 PM.

```
%pyspark
catJdist = catAdd.join(distAdd,8)
```

Took 1 sec. Last updated by anonymous at June 28 2021, 4:59:02 PM.

```
%pyspark
catJdist.getNumPartitions()
```

```
8
```

Took 0 sec. Last updated by anonymous at June 28 2021, 4:59:35 PM.

```
%pyspark
catJdist.partitioner
```

```
<pyspark.rdd.Partitioner object at 0x7f42a2454510>
```

Word Count Using Pair.

Finally, let us perform the word count application using the Pair data model.

```
counts = sc.textFile("/opt/data/J_AddCat.csv").flatMap(lambda line : line.split(',')).map(lambda word : (word,1))
result = counts.reduceByKey(lambda v1,v2 : v1+v2)
result.collect()
```

```
%pyspark
counts = sc.textFile("/opt/data/J_AddCat.csv").flatMap(lambda line : line.split(',')).map(lambda word : (word,1))
result = counts.reduceByKey(lambda v1,v2 : v1+v2)
result.collect()
```

SPARK JOB FINISHED

```
[(u'SUTTER_ST/MASON_ST', 1), (u'0_Block_of_SOUTHPARK_AV', 1), (u'1400_Block_of_CLEMENT_ST', 1), (u'1500_Block_of_15TH_ST', 1), (u'1100_Block_of_SELBY_ST', 1),
(u'100_Block_of_JEFFERSON_ST', 1), (u'1400_Block_of_VANDYKE_AV', 1), (u'1600_Block_of_FILLMORE_ST', 1), (u'EMBEZZLEMENT', 8), (u'200_Block_of_BUSH_ST', 1),
(u'BRIBERY', 4), (u'1900_Block_of_MISSION_ST', 1), (u'2000_Block_of_MARKET_ST', 1), (u'BAD CHECKS', 1), (u'800_Block_of_MARKET_ST', 1), (u'100_Block_of_BLUXOM
E_ST', 1)]
```

Took 1 sec. Last updated by anonymous at June 28 2021, 5:01:23 PM.

Errata: Lambda parameter errors. Sol – Remove the enclosing bracket.

```
>>> incByDists = sfpdRDD.filter(lambda rec : len(rec) == 14 ).map(lambda incident : (incident(6),1)).reduce
ByKey(lambda (x,y) : x+y)
  File "<stdin>", line 1
    incByDists = sfpdRDD.filter(lambda rec : len(rec) == 14 ).map(lambda incident : (incident(6),1)).reduce
ByKey(lambda (x,y) : x+y)
              ^
SyntaxError: invalid syntax
>>> incByDists = sfpdRDD.filter(lambda rec : len(rec) == 14 ).map(lambda incident : (incident(6),1)).reduce
ByKey(lambda x,y : x+y)
>>> incByDists.getNumPartitions()
2
>>> print(incByDists.partitioner)
```

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

13. Spark SQL – 30 Minutes

Spark SQL can convert an RDD of Row objects to a DataFrame, inferring the datatypes. Rows are constructed by passing a list of key/value pairs as kwargs to the Row class.

Copy the person.txt file to /software folder.

```
Henry,42
Rajnita,40
Henderson,14
Tiraj,5
```

Open a terminal from /software folder and enter the following command.

Using Spark SQL – RDD data structure. (Inferring the Schema Using Reflection)

#pyspark

```
// Create an RDD of Person objects and register it as a table.

peopleL = sc.textFile("person.txt")
peopleL.take(4)

parts = peopleL.map(lambda l: l.split(","))
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))

# Infer the schema, and register the DataFrame as a table.
schemaPeople = spark.createDataFrame(people)
schemaPeople.createOrReplaceTempView("people")


// SQL statements can be run by using the sql methods provided by sqlContext.

# SQL can be run over DataFrames that have been registered as a table.
kids = spark.sql("SELECT name FROM people WHERE age >= 1 AND age <= 9")


// The results of SQL queries are DataFrames and support all the normal RDD operations.

# The results of SQL queries are Dataframe objects.
# rdd returns the content as an :class:`pyspark.RDD` of :class:`Row`.
kidNames = kids.rdd.map(lambda p: "Name: " + p.name).collect()
for name in teenNames:
    print(name)
```

```
%pyspark
parts = peopleL.map(lambda l: l.split(","))
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))
```

Took 0 sec. Last updated by anonymous at June 29 2021, 12:58:01 PM.

```
%pyspark
# Infer the schema, and register the DataFrame as a table.
schemaPeople = spark.createDataFrame(people)
schemaPeople.createOrReplaceTempView("people")
```

Took 10 sec. Last updated by anonymous at June 29 2021, 12:59:54 PM.

```
%pyspark
# SQL can be run over DataFrames that have been registered as a table.
kids = spark.sql("SELECT name FROM people WHERE age >= 1 AND age <= 9")
```

Took 1 sec. Last updated by anonymous at June 29 2021, 1:06:28 PM.

```
%pyspark
# The results of SQL queries are Dataframe objects.
# rdd returns the content as an :class:`pyspark.RDD` of :class:`Row`.
kidNames = kids.rdd.map(lambda p: "Name: " + p.name).collect()
for name in teenNames:
    print(name)
```

```
Name: Tiraj
```

Create a temporary view:

```
# Register the DataFrame as a SQL temporary view
schemaPeople.createTempView("kids")
```

Display the output.

```
spark.sql("select * from kids").show()
```

```
%pyspark
# Register the DataFrame as a SQL temporary view
schemaPeople.createTempView("kids")
```

Took 0 sec. Last updated by anonymous at June 29 2021, 1:07:51 PM. (outdated)

```
%pyspark
spark.sql("select * from kids").show()
```

```
+---+---------+
|age|     name|
+---+---------+
| 42|    Henry|
| 40|  Rajnita|
| 14|Henderson|
|  5|    Tiraj|
+---+---------+
```

Show the Tables or View List.

spark.catalog.listTables('default')

or

spark.sql('show tables from default').show()

```
%pyspark
spark.catalog.listTables('default')
```

SPARK JOB FINISHED

```
[Table(name=u'kids', database=None, description=None, tableType=u'TEMPORARY', isTemporary=True), Table(name=u'people', database=None, description=None, tableT
ype=u'TEMPORARY', isTemporary=True)]
```

Took 4 sec. Last updated by anonymous at June 29 2021, 1:12:07 PM.

```
%pyspark
spark.sql('show tables from default').show()
```

FINISHED

```
+--------+---------+-----------+
|database|tableName|isTemporary|
+--------+---------+-----------+
|        |     kids|       true|
|        |   people|       true|
+--------+---------+-----------+
```

Took 0 sec. Last updated by anonymous at June 29 2021, 1:12:34 PM.

Using DataFrame and SQL:

/software/people.json

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

```
peopleDF = spark.read.format("json").load("people.json")
peopleDF.select("name", "age").write.format("parquet").save("namesAndAges.parquet")
peopleDF.select("name","age").show
```

```
%pyspark
peopleDF = spark.read.format("json").load("people.json")
```

Took 3 sec. Last updated by anonymous at June 29 2021, 1:18:22 PM.

```
%pyspark
peopleDF.select("name", "age").write.format("parquet").save("namesAndAges.parquet")
```

Took 6 sec. Last updated by anonymous at June 29 2021, 1:18:42 PM.

```
%pyspark
peopleDF.select("name","age").show()
```

```
+-------+----+
|   name| age|
+-------+----+
|Michael|null|
|   Andy|  30|
| Justin|  19|
+-------+----+
```

Took 1 sec. Last updated by anonymous at June 29 2021, 1:19:08 PM.

Query file directly with SQL.

```
sqlDF = spark.sql("SELECT * FROM parquet.`namesAndAges.parquet`")
```

sqlDF.show()

Perform some operations on the Dataframe.

In Python, it's possible to access a DataFrame's columns either by attribute (`df.age`) or by indexing (`df['age']`).

#convert the people RDD to Dataframe
df = people.toDF()

```
# Select everybody, but increment the age by 1
df.select(df['name'], df['age'] + 1).show()

# Select people older than 21
df.filter(df['age'] > 21).show()

# Count people by age
df.groupBy("age").count().show()
```

```
%pyspark
#convert the people RDD to Dataframe
df = people.toDF()
```

Took 0 sec. Last updated by anonymous at June 29 2021, 1:23:13 PM. (outdated)

```
%pyspark
df.select(df['name'], df['age'] + 1).show()
```

```
+---------+---------+
|     name|(age + 1)|
+---------+---------+
|    Henry|       43|
|  Rajnita|       41|
|Henderson|       15|
|    Tiraj|        6|
+---------+---------+
```

Took 1 sec. Last updated by anonymous at June 29 2021, 1:23:24 PM.

```
%pyspark
df.filter(df['age'] > 21).show()
```

```
+---+-------+
|age|   name|
+---+-------+
| 42|  Henry|
| 40|Rajnita|
+---+-------+
```

```
%pyspark
df.groupBy("age").count().show()
```

```
+---+-----+
|age|count|
+---+-----+
|  5|    1|
| 14|    1|
| 42|    1|
| 40|    1|
+---+-----+
```

Took 11 sec. Last updated by anonymous at June 29 2021, 1:23:57 PM.

Using aggregrate functions:

Determine the max and min age.

spark.sql("select min(age) from kids ").show()

```
%pyspark
spark.sql("select min(age) from kids ").show()
```

```
+--------+
|min(age)|
+--------+
|       5|
+--------+
```

Took 1 sec. Last updated by anonymous at June 29 2021, 2:01:51 PM.

```
%pyspark
spark.sql("select max(age) from kids ").show()
```

```
+--------+
|max(age)|
+--------+
|      42|
+--------+
```

Took 1 sec. Last updated by anonymous at June 29 2021, 2:02:16 PM.

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

14. Applications (Transformation) – Python – 30 Minutes

In this lab, let us develop a python application and submit to Spark for execution. Ensure that you have created a folder /spark/workspace for storing all the python code inside it.

Let us create a simple Spark application, SimpleApp.py:

```
"""SimpleApp.py"""
from pyspark import SparkContext

logFile = "/opt/spark/README.md"  # Should be some file on your system
sc = SparkContext("local", "Simple App")
logData = sc.textFile(logFile).cache()

numAs = logData.filter(lambda s: 'a' in s).count()
numBs = logData.filter(lambda s: 'b' in s).count()

print("Lines with a: %i, lines with b: %i" % (numAs, numBs))

sc.stop()
```

This program just counts the number of lines containing ‘a’ and the number containing ‘b’ in a text file. Note that you’ll need to replace YOUR_SPARK_HOME /logFile with the location where Spark is installed in your machine.

We can run this application using the bin/spark-submit script:

```
# Use spark-submit to run your application, change directory to /spark before executing the following command

$ spark-submit --master local[2] workspace/SimpleApp.py
```

```
[root@localhost spark]# ./spark-2.1.0/bin/spark-submit --master local[2] workspa
ce/SimpleApp.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
17/04/14 09:30:52 WARN Utils: Your hostname, localhost.localdomain resolves to a
 loopback address: 127.0.0.1; using 192.168.150.128 instead (on interface eth0)
17/04/14 09:30:52 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
```

You can view the job status with the following URl

http://192.168.150.128:4040/jobs/

```
 649 ms on localhost (executor driver) (1/1)
17/04/14 09:32:21 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have
all completed, from pool
Lines with a: 62, lines with b: 30
17/04/14 09:32:22 INFO SparkUI: Stopped Spark web UI at http://192.168.150.128:4
040
```

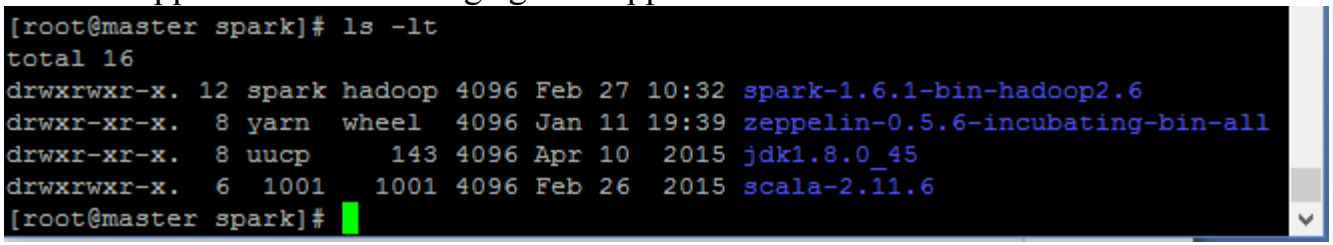
In this way, you can deploy any python application in the spark cluster.

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

15. Zeppelin Installation

You don't require spark installation in case you are using zeppelin-0.6.2-bin-all.tgz
Create a zeppelin user and switch to zeppelin user or if zeppelin user is already created then login as zeppelin.
Password should be life213
Use root credentials
groupadd hadoop
useradd -g hadoop zeppelin
passwd zeppelin

tar -xvf zeppelin-0.7.3-bin-all.tgz.gz -C /apps

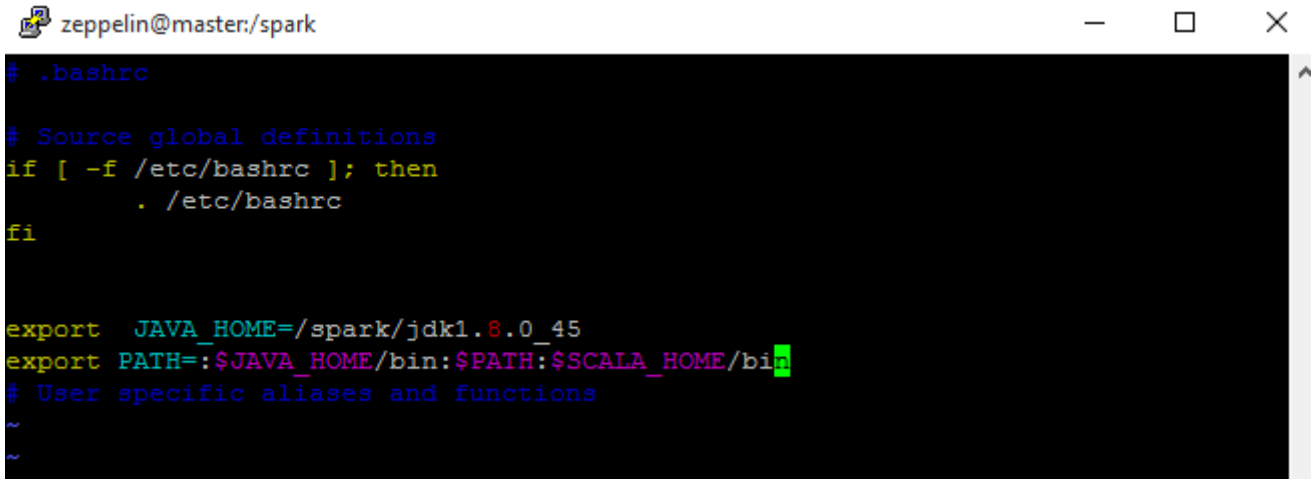


Change the folder owner/group to zeppelin
Note : chgrp -R hadoop zeppelin*
chown zeppelin:hadoop /apps/zeppelin* -R

```
[root@master spark]# chown zeppelin:hadoop zeppelin-0.5.6-incubating-bin-all/ -R
[root@master spark]# ls -lt
total 16
drwxrwxr-x. 12 spark    hadoop 4096 Feb 27 10:32 spark-1.6.1-bin-hadoop2.6
drwxr-xr-x.  8 zeppelin hadoop 4096 Jan 11 19:39 zeppelin-0.5.6-incubating-bin-a
ll
drwxr-xr-x.  8 uucp        143 4096 Apr 10  2015 jdk1.8.0_45
drwxrwxr-x.  6     1001   1001 4096 Feb 26  2015 scala-2.11.6
[root@master spark]#
```

```
su - zeppelin
whoami
```

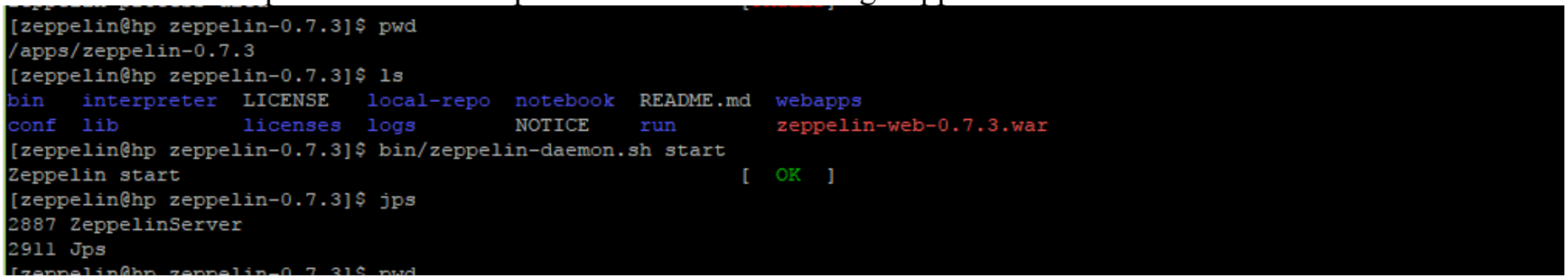
set the JAVA_HOME [vi ~/.bashrc]
export JAVA_HOME=/spark/jdk1.8.0_45
export PATH=:$JAVA_HOME/bin:$PATH:$SCALA_HOME/bin



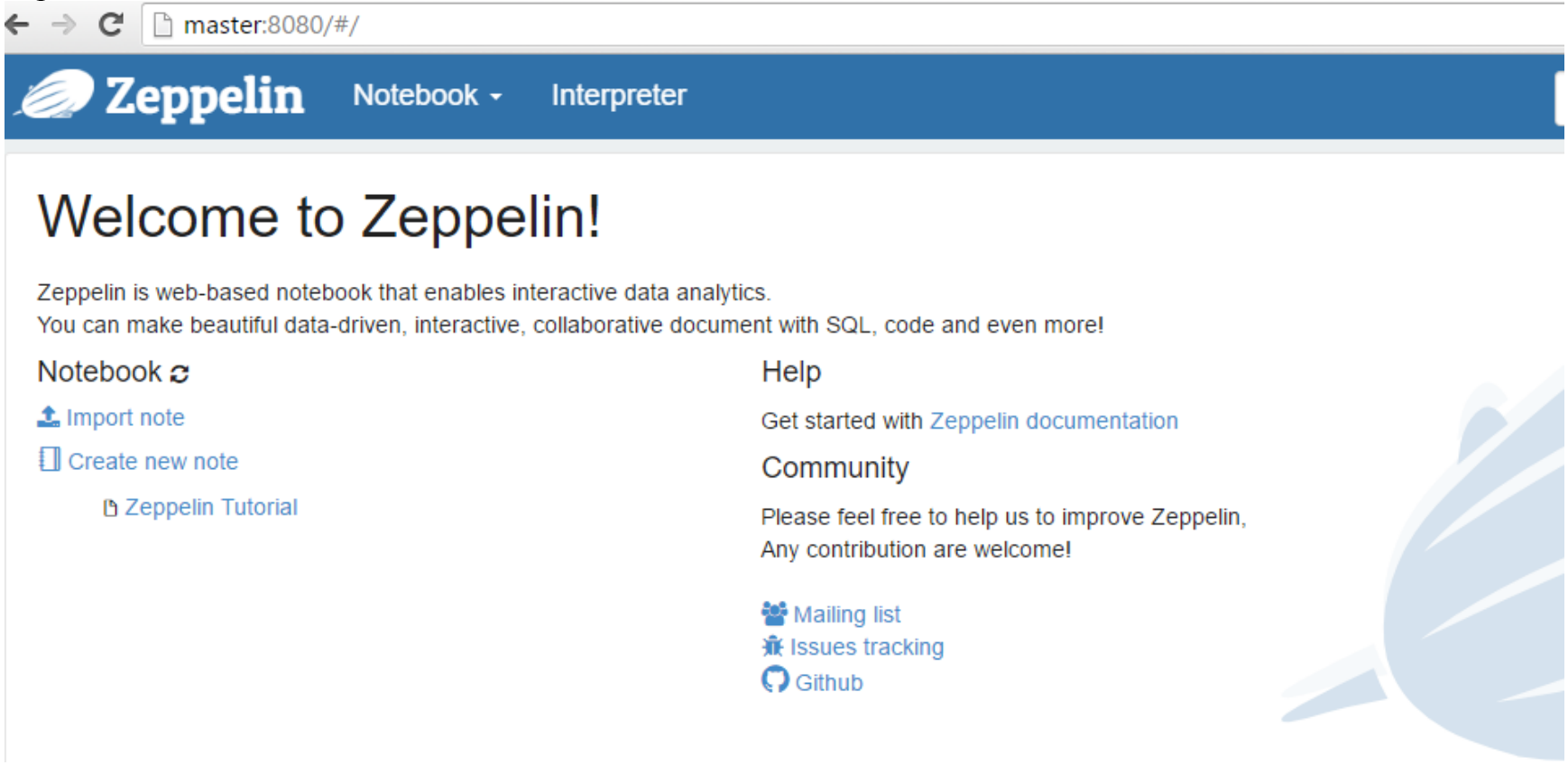
#cp zeppelin-site.xml.template zeppelin-site.xml

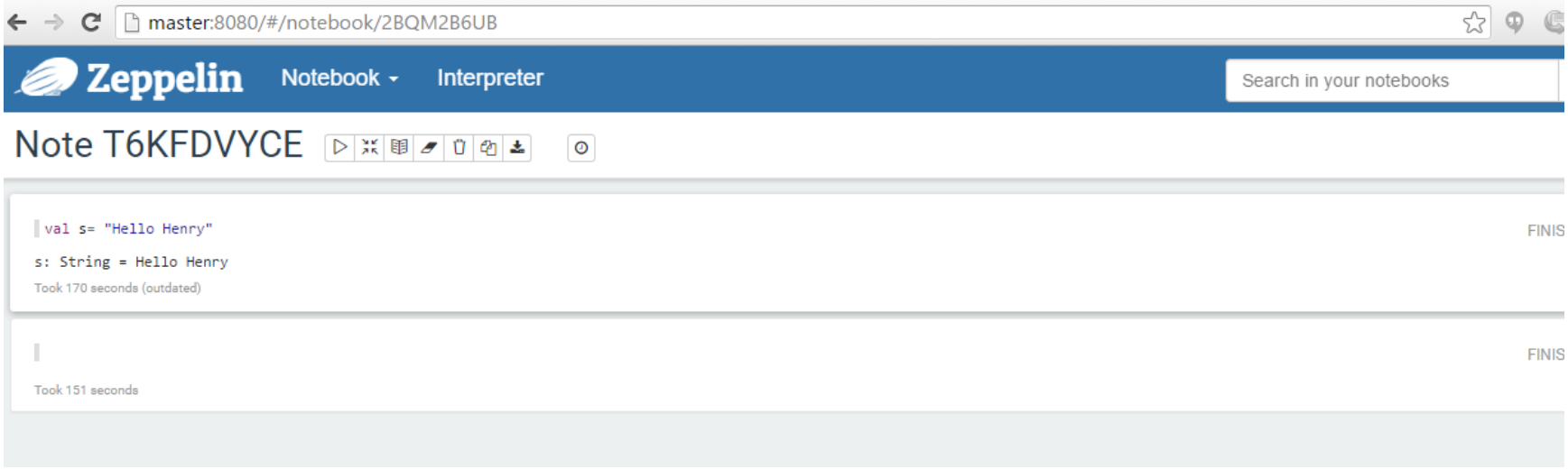
```
<property>
  <name>zeppelin.server.addr</name>
  <value>hp.tos.com</value>
  <description>Server address</description>
</property>

<property>
  <name>zeppelin.server.port</name>
  <value>8081</value>
  <description>Server port.</description>
</property>
```

Start Zeppelin
cd /spark/zeppelin-0.5.6-incubating-bin-all
bin/zeppelin-daemon.sh start
jps
Note: You don't required to start the spark cluster before starting Zeppelin

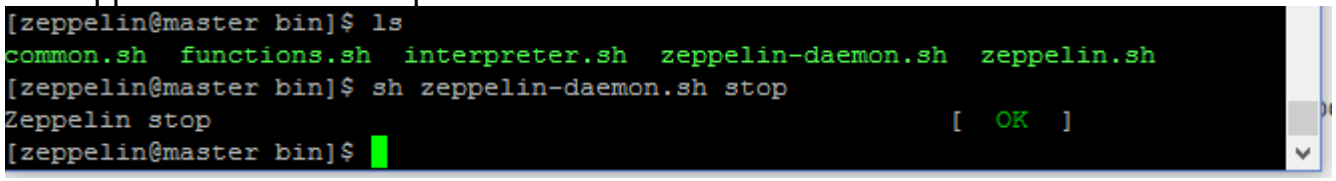


http://master:8081/#/




master:8080/#/notebook/2BQM2B6UB
Zeppelin
Notebook
Interpreter
Search in your notebooks
Note T6KFDVYCE
val s= "Hello Henry"
FINIS
s: String = Hello Henry
Took 170 seconds (outdated)
FINIS
Took 151 seconds

Stop Zeppelin
bin/zeppelin-daemon.sh stop



Let us connect Zeppelin with the existing Spark Cluster. You can start Spark in a single standalone cluster or Use any existing Spark Cluster.

In this lab let us use the single standalone spark cluster,
Let us set the Spark Metadata to be stored in derby Network so that more than one user can connect using spark shell.

Unzip the derby database and start as below: You can use zeppelin user account to start the derby.

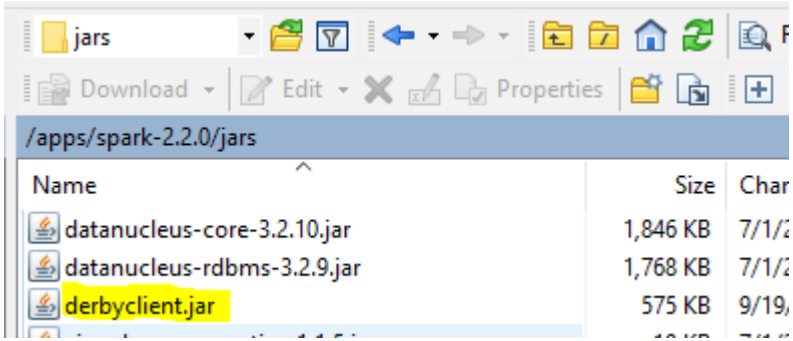
#tar -xzf db-derby-10.14.1.0-bin.tar.gz -C /apps
#mkdir db-derby-10.4.1.3-bin/data
#nohup /apps/db-derby-10.14.1.0-bin/bin/startNetworkServer -h 0.0.0.0 &

After the derby server, you need to modify the spark to refer the above database server. For this create a file hive-site.xml in the SPARK_HOME/conf folder and paste the following content,

```
<configuration>
        <property>
          <name>javax.jdo.option.ConnectionURL</name>
          <value>jdbc:derby://10.10.20.27:1527/metastore_db;create=true</value>
        </property>
```

```
    <property>
      <name>javax.jdo.option.ConnectionDriverName</name>
      <value>org.apache.derby.jdbc.ClientDriver</value>
    </property>

</configuration>
```

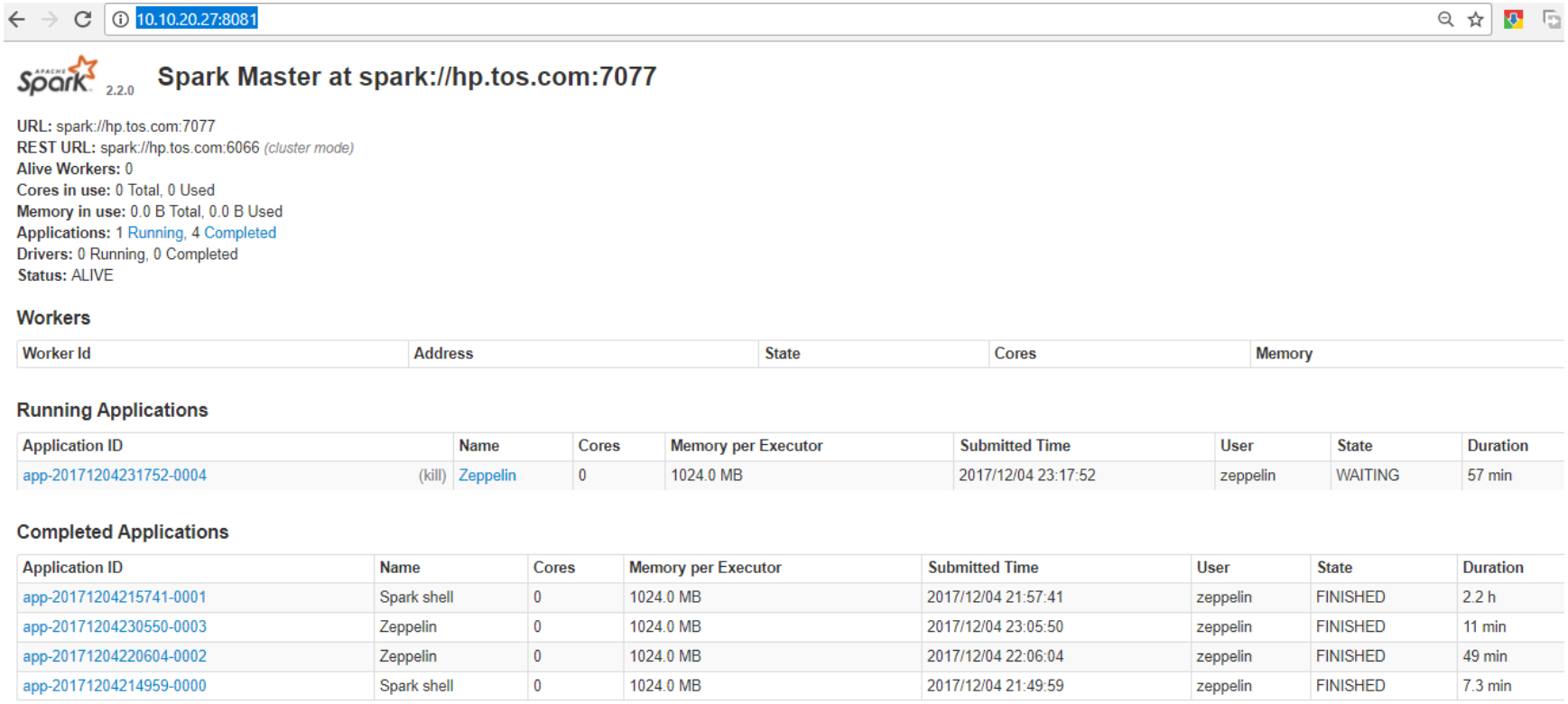
Remove the derby jar from the Spark installation jar folder and replace with the derby client jar.



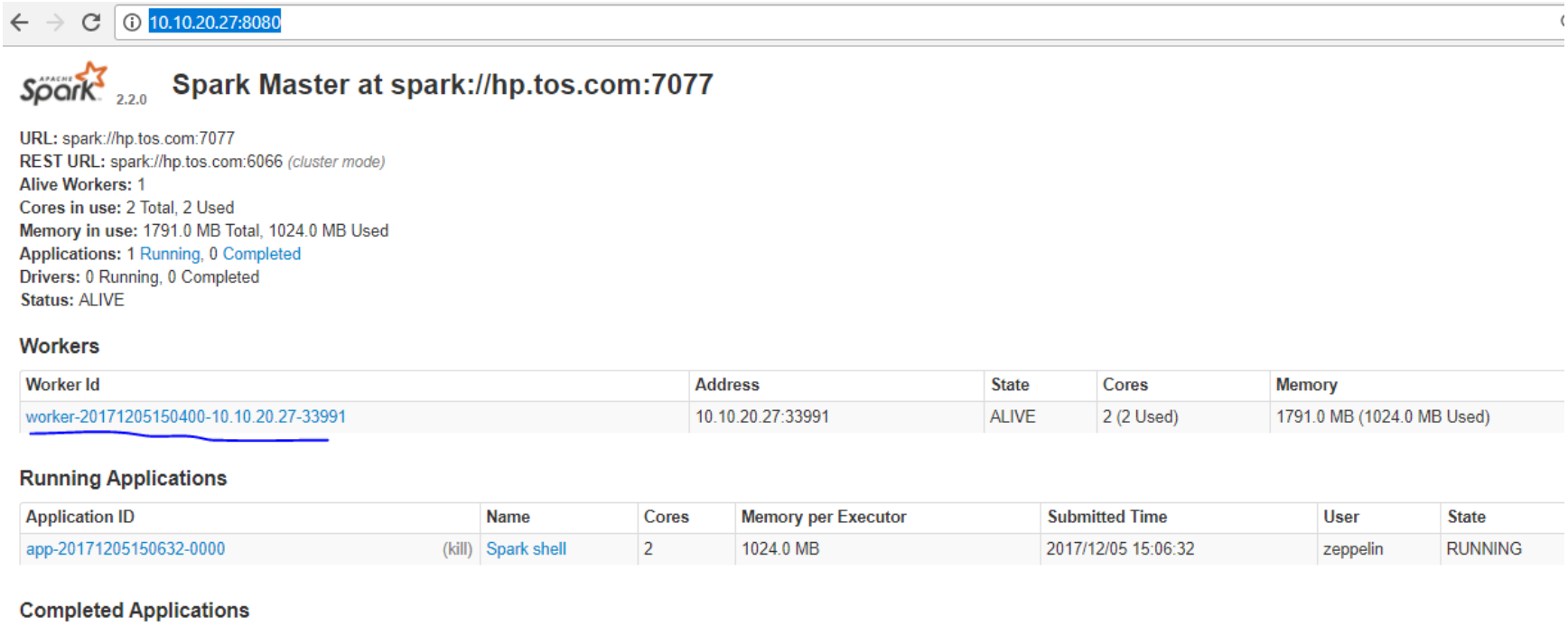
After that start the spark standalone cluster:

sbin/start-master.sh --webui-port 8081

At the end of the above step you should be able to access the Spark UI
http://10.10.20.27:8081/



Now, we have configured SPARK standalone cluster. We need to have atleast one worker node as shown below:
Add hp.tos.com in the slave file and start the slave with the following command
sh start-slave.sh spark://hp.tos.com:7077
http://10.10.20.27:8080/



Let us configure zeppelin to connect to it.

Stop the zeppelin if not done before. Use zeppelin user ID to execute the following command

#bin/zeppelin-daemon.sh stop

open the **zeppelin-env.sh** file present in **$zeppelin_HOME/conf** directory and provide the below specified configurations.

export MASTER=spark://hp.tos.com:7077
export SPARK_HOME=/apps/spark-2.2.0

Save the above file.

Let us activate admin user for logon to zeppelin web UI.

#cd /apps/zeppelin-0.7.3/conf

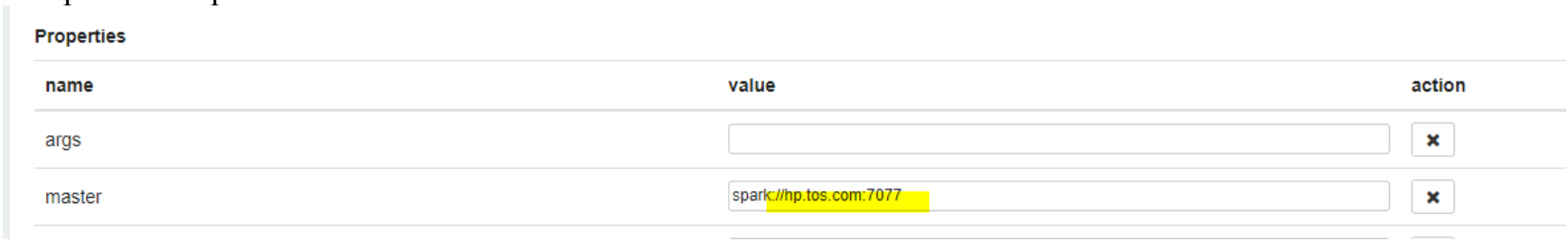
#cp shiro.ini.template shiro.ini
update the password of admin as shown below in the above ini file.
admin = life213

start the zeppelin using zeppelin user ID.
#bin/zeppelin-daemon.sh start

Now open your Zeppelin dashboard and go to the list of interprets and search for Spark interpreter.

Ensure that you modify the below parameters: You need to logon using admin/life213 credentials.

Interpreter → Spark → Edit



zeppelin.spark.useHiveContext	false

Save and then accept the prompt by clicking Ok button.

Create a New Notebook [Notebook → Create New Note → Enter Hello Spark as the name]

Enter sc and the click Run which is on the right corner. If everything is ok, you should be able to see as the above screenshot. Else review the log file in zeppelin installation folder.

If you want to configure python3, update the following parameters.

PYSPARK_PYTHON	python3	Python binary executable to use for PySpark in both driver an d workers (default is python2.7 if available, otherwise python). Property `spark.pyspark.python` take precedence if it is set
PYSPARK_DRIVER_PYTHON	python3	Python binary executable to use for PySpark in driver only (de fault is `PYSPARK_PYTHON`). Property `spark.pyspark.driver. python` take precedence if it is set

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

16. Using the Python Interpreter

To ensure Zeppelin uses that version of Python when you use the Python interpreter, set the zeppelin.python setting to the path to Anaconda.

Using the Python Interpreter

In a paragraph, use *%python* to select the **Python** interpreter and then input all commands.

The interpreter can only work if you already have python installed (the interpreter doesn't bring it own python binaries).

To access the help, type **help()**

%spark.pyspark PySparkInterpreter Provides a Python environment

%spark.r SparkRInterpreter Provides an R environment with SparkR support

%spark.sql SparkSQLInterpreter Provides a SQL environment

%spark.dep DepInterpreter Dependency loader

You should set your PYSPARK_DRIVER_PYTHON environment variable so that Spark uses Anaconda. You can get more information here:

https://spark.apache.org/docs/1.6.2/programming-guide.html

Export SPARK_HOME

In conf/zeppelin-env.sh, export SPARK_HOME environment variable with your Spark installation path.

For example,

export SPARK_HOME=/spark/spark-2.1.0

Install Anaconda2 and set in the Path Variable of root login. (vi ~/.bashrc)

export PATH="/spark/anaconda2/bin:$PATH"

18. PySpark-Example – Zeppelin.

```
%spark.pyspark sc
words = sc.textFile('file:///spark/spark-2.1.0/README.md')

 %pyspark
words.flatMap(lambda x: x.lower().split(' ')) \
 .filter(lambda x: x.isalpha()).map(lambda x: (x, 1)) \
 .reduceByKey(lambda a,b: a+b)

%pyspark
words.first()
```

```
%spark.pyspark sc

words = sc.textFile('file:///spark/spark-2.1.0/README.md')
```

Took 1 min 32 sec. Last updated by anonymous at April 15 2017, 8:14:32 PM.

```
%pyspark
words.flatMap(lambda x: x.lower().split(' ')) \
 .filter(lambda x: x.isalpha()).map(lambda x: (x, 1)) \
 .reduceByKey(lambda a,b: a+b)
```

```
PythonRDD[6] at RDD at PythonRDD.scala:48
```

Took 1 sec. Last updated by anonymous at April 15 2017, 8:17:15 PM.

```
%pyspark
words.first()
```

```
u'# Apache Spark'
```

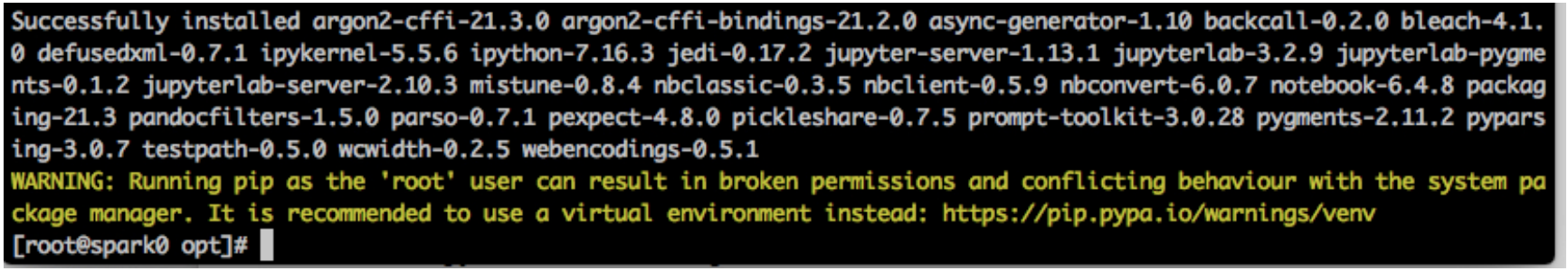
Took 0 sec. Last updated by anonymous at April 15 2017, 8:18:30 PM.

19.Installing Jupyter for Spark – 35 Minutes.(D)

You can install Jupter using pip or anaconda.

Using pip – We will use this for our lab.

```
#yum install epel-release
#yum -y install python-pip

Using pip
#yum install python3-pip -y
#pip3 install --upgrade setuptools
#pip3 install --upgrade pip
#pip3 install jupyterlab
```



To run the notebook:
#jupyter notebook --generate-config

Start Jupyter in the different port.
#jupyter notebook --generate-config
[Writing default config to: /root/.jupyter/jupyter_notebook_config.py]

Update the following properties.
#vi ~/.jupyter/jupyter_notebook_config.py

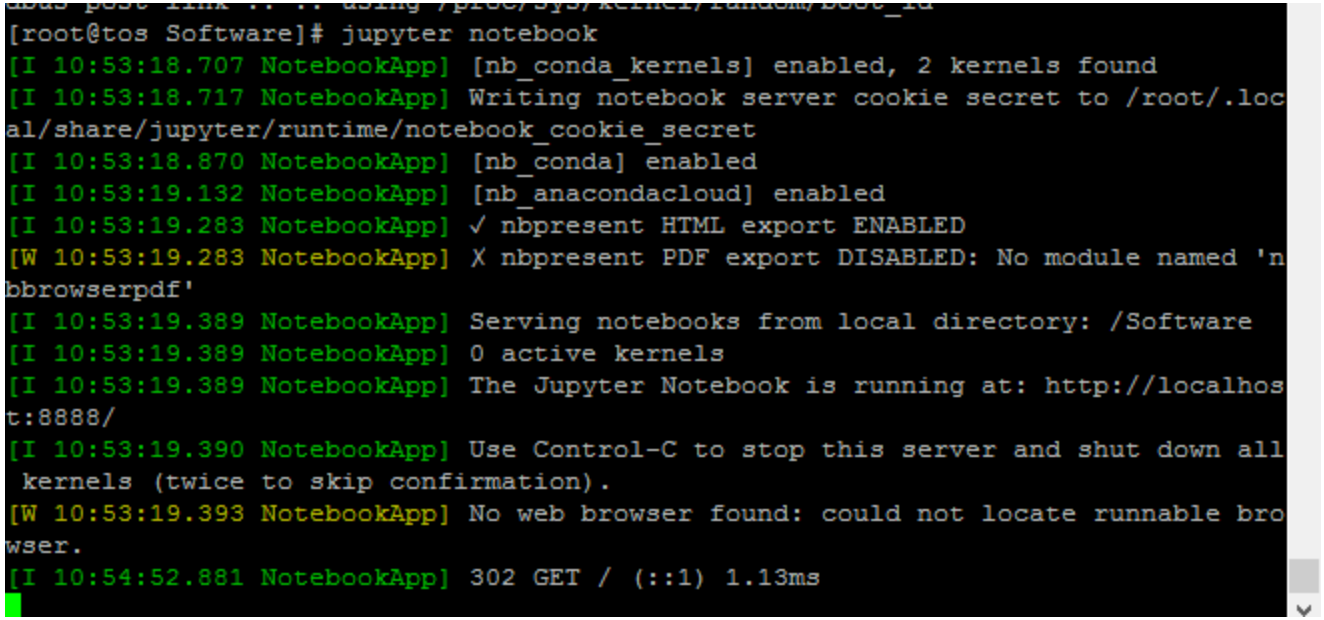
c.NotebookApp.ip = 'sparko'
c.NotebookApp.open_browser = False
c.NotebookApp.allow_remote_access = True
c.NotebookApp.allow_root = True

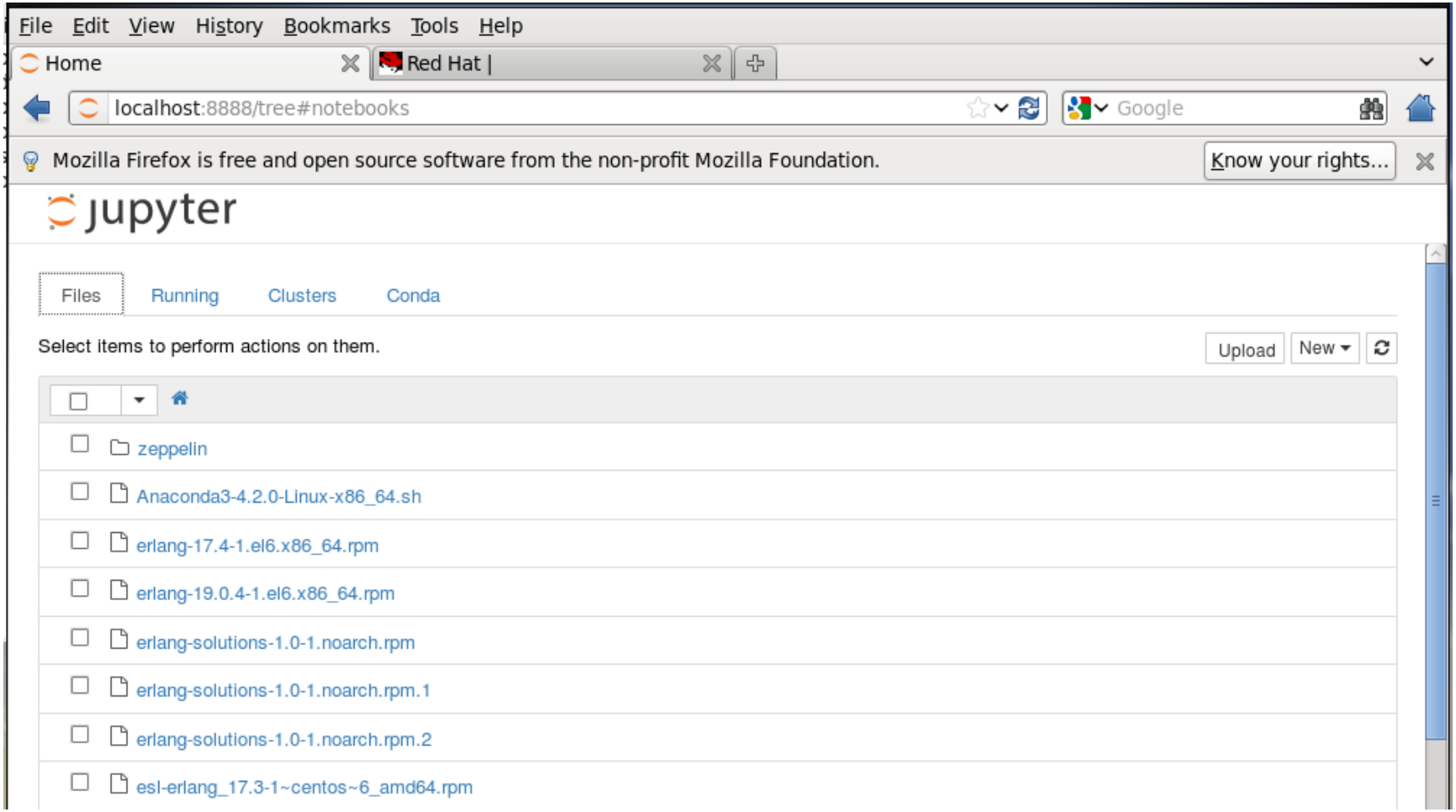
If you want to update the port no.

```
## The port the notebook server will listen on (env: JUPYTER_PORT).
#  Default: 8888
 c.NotebookApp.port = 8200
```

#jupyter notebook

Copy the URL in the web browser.


[root@tos Software]# jupyter notebook
[I 10:53:18.707 NotebookApp] [nb_conda_kernels] enabled, 2 kernels found
[I 10:53:18.717 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
[I 10:53:18.870 NotebookApp] [nb_conda] enabled
[I 10:53:19.132 NotebookApp] [nb_anacondacloud] enabled
[I 10:53:19.283 NotebookApp] ✓ nbpresent HTML export ENABLED
[W 10:53:19.283 NotebookApp] X nbpresent PDF export DISABLED: No module named 'nbbrowserpdf'
[I 10:53:19.389 NotebookApp] Serving notebooks from local directory: /Software
[I 10:53:19.389 NotebookApp] 0 active kernels
[I 10:53:19.389 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
[I 10:53:19.390 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 10:53:19.393 NotebookApp] No web browser found: could not locate runnable browser.
[I 10:54:52.881 NotebookApp] 302 GET / (::1) 1.13ms



Let us configure Notebook for Pyspark.

Update PySpark driver environment variables: add these lines to your **~/.bashrc** (or **~/.zshrc**) file.

```
export PYSPARK_DRIVER_PYTHON=jupyter

export PYSPARK_DRIVER_PYTHON_OPTS='notebook'
```

Restart your terminal and launch PySpark again:

```
$ pyspark
```

Now, this command should start a Jupyter Notebook in your web browser. Create a new notebook by clicking on ‘New’ > ‘Notebooks Python [default]’.

Copy and paste Pi calculation script and run it by pressing Shift + Enter.

// Code Begin

```
import random
num_samples = 100000000

def inside(p):
  x, y = random.random(), random.random()
  return x*x + y*y < 1

count = sc.parallelize(range(0, num_samples)).filter(inside).count()

pi = 4 * count / num_samples
print(pi)

sc.stop()
```

// Code Ends.

In [1]:
```
import random
num_samples = 100000000

def inside(p):
  x, y = random.random(), random.random()
  return x*x + y*y < 1

count = sc.parallelize(range(0, num_samples)).filter(inside).count()

pi = 4 * count / num_samples
print(pi)

sc.stop()
```

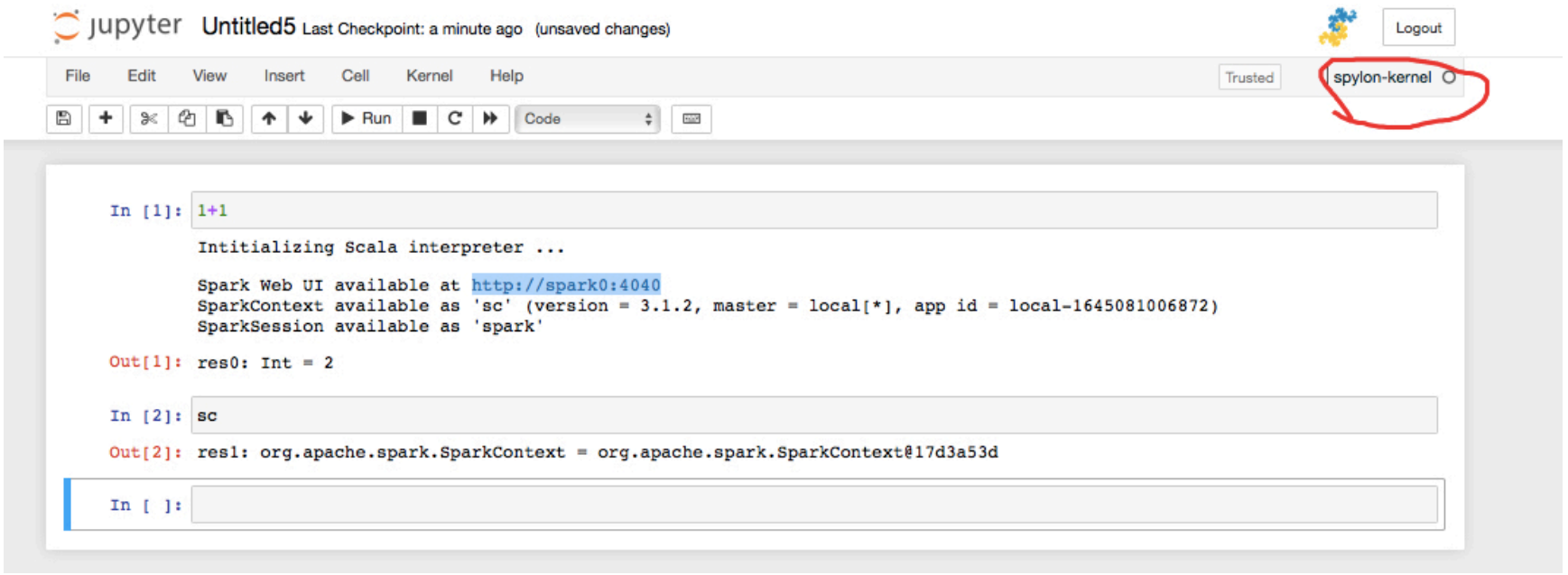
```
3.14156176
```

You have successfully configured pyspark on Jupyter.

Configure scala on Jupyter.

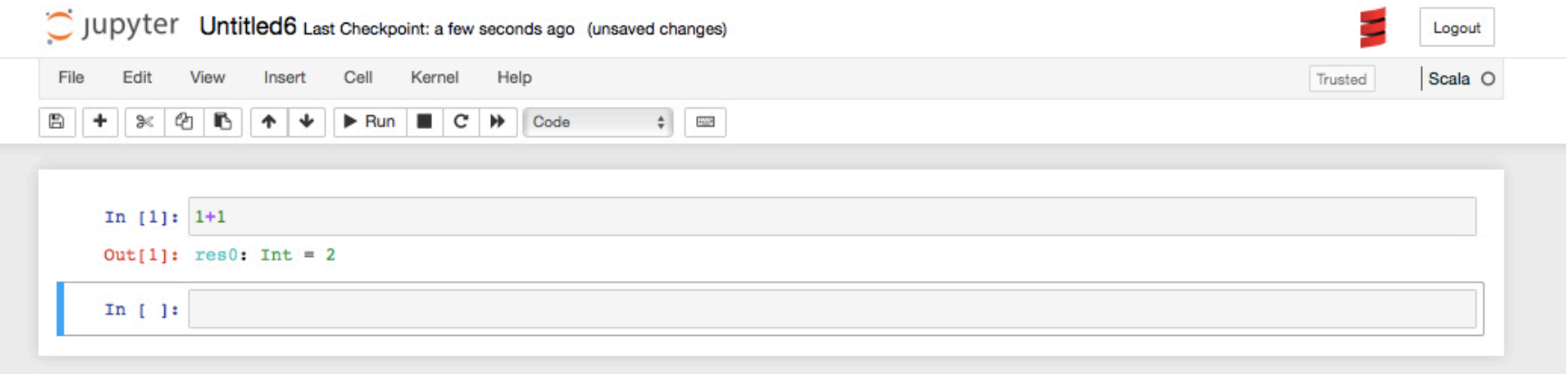
Option I: Using spylon – Execute Scala spark program.

#pip3 install spylon-kernel
```
#python3 -m spylon_kernel install

# export SPARK_HOME=/opt/spark

#jupyter notebook
```



Option 2: using Scala in Jupyter

https://almond.sh/docs/quick-start-install



Or using Anaconda: (Optional)

Anaconda 4.2.0
For Linux
Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.
Changelog
Download the installer
Optional: Verify data integrity with MD5 or SHA-256 _ More info
In your terminal window type one of the below and follow the instructions:
Python 3.5 version
bash Anaconda3-4.2.0-Linux-x86_64.sh
Python 2.7 version
bash Anaconda2-4.2.0-Linux-x86_64.sh
NOTE: Include the "bash" command even if you are not using the bash shell.

```
[root@tos Software]# bash
[root@tos Software]# sh Anaconda3-4.2.0-Linux-x86_64.sh

Welcome to Anaconda3 4.2.0 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

```
Anaconda3 will now be installed into this location:
/root/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/root/anaconda3] >>> /anaconda3
```

```
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /root/.bashrc ? [yes|no]
[no] >>> yes

Prepending PATH=/anaconda3/bin to PATH in /root/.bashrc
A backup will be made to: /root/.bashrc-anaconda3.bak


For this change to become active, you have to open a new terminal.

Thank you for installing Anaconda3!

Share your notebooks and packages on Anaconda Cloud!
Sign up for free: https://anaconda.org

[root@tos Software]#
```

#bash

Run the following command to install Jupyter.
#conda install jupyter

```
[root@tos Software]# conda install jupyter
Fetching package metadata .......
Solving package specifications: ..........

Package plan for installation in environment /anaconda3:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-env-2.6.0            |                0          502 B
    requests-2.12.4            |            py35_0         800 KB
    pyopenssl-16.2.0           |            py35_0          70 KB
    conda-4.3.7                |            py35_0         491 KB
    ------------------------------------------------------------
                                           Total:         1.3 MB
```

```
The following packages will be UPDATED:

    conda:     4.2.9-py35_0  --> 4.3.7-py35_0
    pyopenssl: 16.0.0-py35_0 --> 16.2.0-py35_0
    requests:  2.11.1-py35_0 --> 2.12.4-py35_0

Proceed ([y]/n)? y

Fetching packages ...
conda-env-2.6. 100% |################################| Time: 0:00:00 129.08 kB/s
requests-2.12. 100% |################################| Time: 0:00:11  70.63 kB/s
pyopenssl-16.2 100% |################################| Time: 0:00:01  53.66 kB/s
conda-4.3.7-py 100% |################################| Time: 0:00:04 104.02 kB/s
Extracting packages ...
[      COMPLETE      ]|##################################################| 100%
Unlinking packages ...
[      COMPLETE      ]|##################################################| 100%
Linking packages ...
[      COMPLETE      ]|##################################################| 100%
dbus post-link :: /etc/machine-id not found ..
dbus post-link :: .. using /proc/sys/kernel/random/boot_id
[root@tos Software]#
```

Anaconda 4.2.0
For Windows
Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.
Changelog
Download the installer
Optional: Verify data integrity with MD5 or SHA-256 More info
Double-click the **.exe** file to install Anaconda and follow the instructions on the screen

Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.
Congratulations, you have installed Jupyter Notebook.

Reference:
https://www.sicara.ai/blog/2017-05-02-get-started-pyspark-jupyter-notebook-3-minutes
https://community.hortonworks.com/articles/75551/installing-and-exploring-spark-20-with-jupyter-not.html
https://medium.com/@bogdan.cojocar/how-to-run-scala-and-spark-in-the-jupyter-notebook-328a80090b3b

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

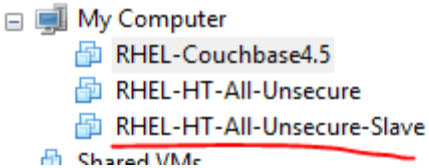
20. Spark Standalone Cluster(VM) – 60 Minutes.

You will learn to configure two nodes Spark Standalone Cluster.

Create another instance of the VM by copying the VM folder or making clone. You need to shutdown the VM for that.
If you use clone, then specify as shown below. Hints: Select the parent virtual machine and select VM > Manage > Clone.

It will take few minutes depending on your VM size.

At the end, you will get an additional VM as shown below: (ex)



You need to power on both VM at this step.

Add another node i.e slave node to the Cluster. Change the hostname of the slave node to slave.

Logon to the slave VM. Your slave node should be as shown below.

```
[root@slave sbin]#
[root@slave sbin]# hostname
slave
[root@slave sbin]#
```

Ensure to provide entries in /etc/hosts of both the VMs, so that it can communicate each other's using hostname.

192.168.188.178 master
192.168.188.174 slave

Note: Replace the IP of your machine accordingly.

Logon to the first VM i.e master to configure password less connection between the nodes.

SSH access

The root user on the master must be able to connect

- to its own user account on the master – i.e. ssh master in this context and not necessarily ssh localhost – and
- to the root user account on the slave via a password-less SSH login.

You have to add the root@master‘s public SSH key (which should be in $HOME/.ssh/id_rsa.pub) to the authorized_keys file of root@slave (in this user’s$HOME/.ssh/authorized_keys).

The following steps will ensure that root user can ssh to its own account without password in all nodes.
Let us generate the keys for user root on the master node, to make sure that root user on master can ssh to slave nodes without password.

```
$ su - root
$ ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa

$cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ ssh master
```

You need to accept the key for the first time.
You can do this manually or use the following SSH command: to copy public key to all the slave nodes.

```
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub root@slave
```

Accept it, yes and supply the root password for the last time.

This command will prompt you for the login password for user root on slave, then copy the public SSH key for you, creating the correct directory and fixing the permissions as necessary.

The final step is to test the SSH setup by connecting with user root from the master to the user account root on the slave. This step is also needed to save slave‘s host key fingerprint to the root@master‘s known_hosts file.

So, connecting from master to master…
$ ssh master

And from master to slave.
$ ssh slave

```
[root@master sbin]# ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
dd:87:61:51:6e:f9:1e:88:70:a0:ff:f6:92:e5:7f:fb root@master
The key's randomart image is:
+--[ RSA 2048]----+
|          .  ...  |
|         . .  o . |
|        . . .o +  |
|         ..oo.+.. |
|         S...o....|
|           .  o ..|
|            o+   .|
|           .o..  .|
|             ...oE|
+-----------------+
[root@master sbin]# scp
usage: scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-l limit] [-o ssh_option] [-P port] [-S program]
           [[user@]host1:]file1 ... [[user@]host2:]file2
[root@master sbin]# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
[root@master sbin]# ssh master
Last login: Thu Sep  1 23:45:46 2016 from localhost
[root@master ~]# ssh-copy-id -i $HOME/.ssh/id_rsa.pub root@slave
root@slave's password:
Now try logging into the machine, with "ssh 'root@slave'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

[root@master ~]# ssh slave
Last login: Thu Sep  1 23:45:46 2016 from localhost
[root@slave ~]#
```

Go to SPARK_HOME/conf/ and create new file with name spark-env.sh on the master node
*There will be **spark-env.sh.template** in same folder and this file gives you detail on how to declare various environment variables.*

Now we will give the IP address of Master.
SPARK_LOCAL_IP=192.168.188.173
SPARK_MASTER_HOST={IP Address}

```
# Options read by executors and drivers running inside the cluster
# - SPARK_LOCAL_IP, to set the IP address Spark binds to on this node
SPARK_LOCAL_IP=192.168.188.173
# - SPARK_PUBLIC_DNS, to set the public DNS name of the driver program
# - SPARK_CLASSPATH, default classpath entries to append
# - SPARK_LOCAL_DIRS, storage directories to use on this node for shuffle and RD
```

```
# Options for the daemons used in the standalone deploy mode
# - SPARK_MASTER_HOST, to bind the master to a different IP address or hostname
SPARK_MASTER_HOST=192.168.188.173
# - SPARK_MASTER_PORT / SPARK_MASTER_WEBUI_PORT, to use non-default ports for th
e master
```

// Note: You need to specify IP only.

On master node; Create SPARK_HOME/conf/slaves and enter the following entries.
Worker process will be started in both the nodes.

```
[root@master conf]# more slaves
slave
master
[root@master conf]#
```

Then go to Spark HOME_DIRECTORY/sbin and run following command from terminal

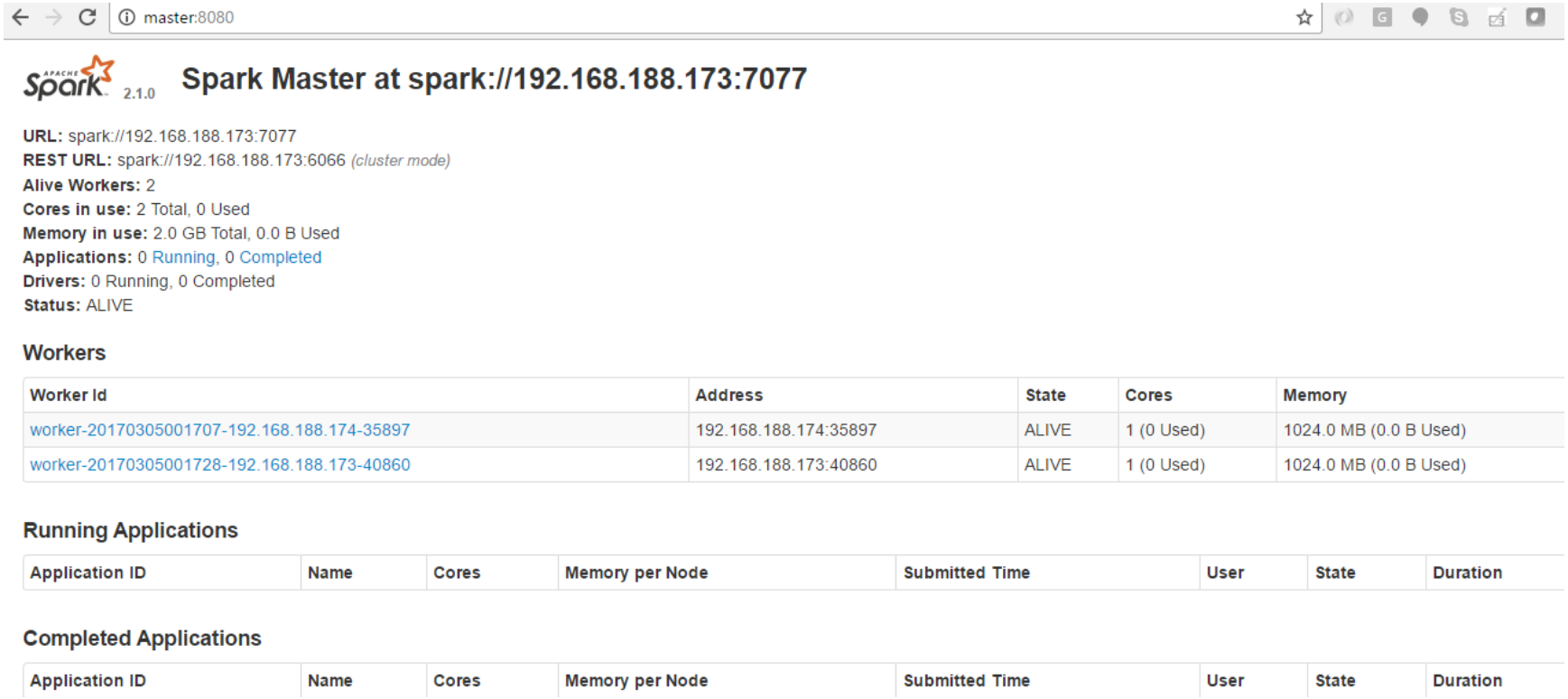
#sh sbin/start-all.sh

```
[root@master sbin]# sh start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /spark/spark-2.1/logs
/spark-root-org.apache.spark.deploy.master.Master-1-master.out
slave: starting org.apache.spark.deploy.worker.Worker, logging to /spark/spark-2
.1/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-slave.out
master: starting org.apache.spark.deploy.worker.Worker, logging to /spark/spark-
2.1/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-master.out
master: failed to launch: nice -n 0 /spark/spark-2.1/bin/spark-class org.apache.
spark.deploy.worker.Worker --webui-port 8081 spark://192.168.188.173:7077
master: full log in /spark/spark-2.1/logs/spark-root-org.apache.spark.deploy.wor
ker.Worker-1-master.out
```

This will start a standalone master server.

http://master:8080/

You can access the Spark Master with the above URL.



Once started, the master will print out a spark://HOST:PORT URL for itself, which you can use to connect workers to it, or pass as the "master" argument to SparkContext. You can also find this URL on the master's web UI, which is http://localhost:8080 by default.

Once you have started a worker, look at the master's web UI (http://localhost:8080 by default). You should see the new node listed there, along with its number of CPUs and memory (minus one gigabyte left for the OS).
In our case its having 2 nodes.

Then you can verify the Node using the master UI:

Workers

Worker Id	Address	State	Cores	Memory
worker-20170305001707-192.168.188.174-35897	192.168.188.174:35897	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20170305001728-192.168.188.173-40860	192.168.188.173:40860	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

Let us execute Spark shell using Cluster mode:

You can perform from any client/server node i.e windows desktop client.

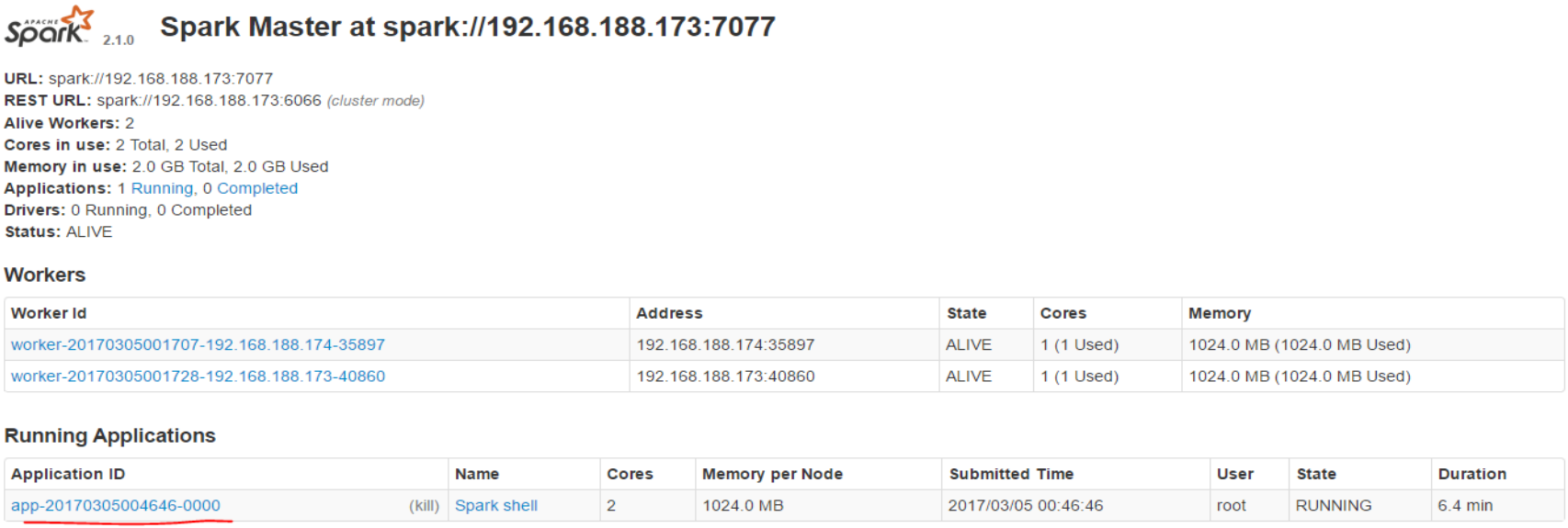
Connecting an Application to the Cluster

To run an application on the Spark cluster, simply pass the spark://IP:PORT URL of the master as to the SparkContext constructor.

To run an interactive Spark shell against the cluster(Specify the master IP), run the following command from the bin folder:
We are executing from the slave node.

#pyspark --conf spark.executor.memory=512mb --conf spark.executor.cores=1 --master spark://spark0:8090

You can verify the application execution from the web UI



Spark 2.1.0 **Spark Master at spark://192.168.188.173:7077**

URL: spark://192.168.188.173:7077
REST URL: spark://192.168.188.173:6066 *(cluster mode)*
Alive Workers: 2
Cores in use: 2 Total, 2 Used
Memory in use: 2.0 GB Total, 2.0 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20170305001707-192.168.188.174-35897	192.168.188.174:35897	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)
worker-20170305001728-192.168.188.173-40860	192.168.188.173:40860	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20170305004646-0000 (kill)	Spark shell	2	1024.0 MB	2017/03/05 00:46:46	root	RUNNING	6.4 min

Let's make a new RDD from the text of the README file in the Spark source directory:

```
textFile = sc.textFile("README.md")
```

```
scala> val textFile  = sc.textFile("/MyTrainingWork/Spark/README.md")
15/06/03 23:32:06 INFO MemoryStore: ensureFreeSpace(73391) called with curMem=18
1810, maxMem=278019440
15/06/03 23:32:06 INFO MemoryStore: Block broadcast_1 stored as values in memory
 (estimated size 71.7 KB, free 264.9 MB)
15/06/03 23:32:06 INFO MemoryStore: ensureFreeSpace(31262) called with curMem=25
5201, maxMem=278019440
15/06/03 23:32:06 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in
memory (estimated size 30.5 KB, free 264.9 MB)
15/06/03 23:32:06 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on l
ocalhost:1599 (size: 30.5 KB, free: 265.1 MB)
15/06/03 23:32:06 INFO BlockManagerMaster: Updated info of block broadcast_1_pie
ce0
15/06/03 23:32:06 INFO SparkContext: Created broadcast 1 from textFile at <conso
le>:21
textFile: org.apache.spark.rdd.RDD[String] = /MyTrainingWork/Spark/README.md Map
PartitionsRDD[3] at textFile at <console>:21
```

Let's start with a few actions:

textFile.count() *// Number of items in this RDD*

```
scala> textFile.count()
15/06/03 23:32:16 INFO FileInputFormat: Total input paths to process : 1
15/06/03 23:32:16 INFO SparkContext: Starting job: count at <console>:24
15/06/03 23:32:16 INFO DAGScheduler: Got job 0 (count at <console>:24) with 2 ou
tput partitions (allowLocal=false)
15/06/03 23:32:16 INFO DAGScheduler: Final stage: Stage 0(count at <console>:24)

15/06/03 23:32:16 INFO DAGScheduler: Parents of final stage: List()
15/06/03 23:32:16 INFO DAGScheduler: Missing parents: List()
15/06/03 23:32:16 INFO DAGScheduler: Submitting Stage 0 (/MyTrainingWork/Spark/R
EADME.md MapPartitionsRDD[3] at textFile at <console>:21), which has no missing
parents
```

```
tes result sent to driver
15/06/03 23:32:17 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in
 352 ms on localhost (1/2)
15/06/03 23:32:17 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in
 338 ms on localhost (2/2)
15/06/03 23:32:17 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have
all completed, from pool
15/06/03 23:32:17 INFO DAGScheduler: Stage 0 (count at <console>:24) finished in
 0.393 s
15/06/03 23:32:17 INFO DAGScheduler: Job 0 finished: count at <console>:24, took
 0.678076 s
res2: Long = 98

scala>
```

textFile.first() *// First item in this RDD*

```
.md:0+1814
15/06/03 23:34:06 INFO Executor: Finished task 0.0 in stage 1.0 (TID 2). 1809 by
tes result sent to driver
15/06/03 23:34:06 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in
 10 ms on localhost (1/1)
15/06/03 23:34:06 INFO DAGScheduler: Stage 1 (first at <console>:24) finished in
 0.011 s
15/06/03 23:34:06 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have
all completed, from pool
15/06/03 23:34:06 INFO DAGScheduler: Job 1 finished: first at <console>:24, took
 0.019875 s
res3: String = # Apache Spark

scala>
```

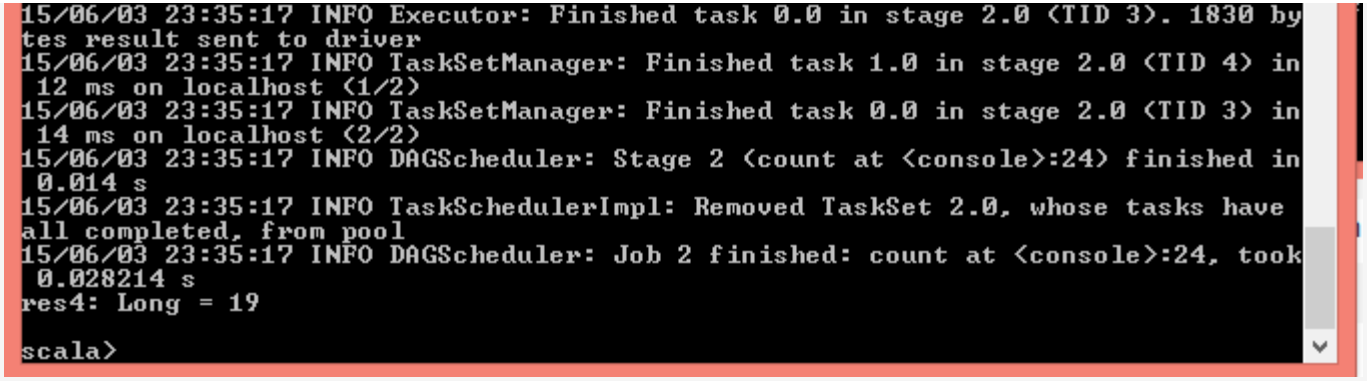
Now let's use a transformation. We will use the filter transformation to return a new RDD with a subset of the items in the file. Use collect to display the output.

```
linesWithSpark = textFile.filter(lambda line : "Spark" in line)
```

```
linesWithSpark.collect()
```

We can chain together transformations and actions:

```
textFile.filter(lambda line : "Spark" in line).count()  // How many lines contain "Spark"?
```



Let's say we want to find the line with the most words

```
textFile.map(lambda line : len(line.split(" "))). reduce(lambda a, b: a if (a > b) else b)
```

Stop the spark process. Hints [sbin/stop-all.sh]

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

21. Spark Two Nodes Cluster Using Docker – 90 Minutes

Prerequisites:

- Install Docker
- Pull Centos Image.

Steps for deploying two node spark cluster in docker:

- Create two containers using the above image
  - Spark0
  - Spark1
- Create a network to connect between these two nodes.

  You should pull the following images from docker hub.(centos:7)

Let us create a common network for our two nodes cluster.

#docker network create --driver bridge spark-net

Create first node, spark0 container

#docker run -it --name spark0 --hostname spark0 --privileged -p 8080:8080 -p 7077:7077 -p 4040:4040 -p 8081:8081 -p 8090:8090 centos:7 /usr/sbin/init

Docker command with Host folder mounted. (Skip)

docker run -it --name spark0 --hostname spark0 --privileged --network spark-net -v /Volumes/Samsung_T5/software/:/Software -v /Volumes/Samsung_T5/software/install/:/opt -v /Volumes/Samsung_T5/software/data/:/data -p 8080:8080 -p 7077:7077 -p 4040:4040 -p 8081:8081 -p 8090:8090 centos:7 /usr/sbin/init

To connect a **running** container to an existing user-defined bridge, use the docker network connect command. If the spark0 is already started without the network being attached execute the following else skip the following step.

```
$ docker network connect spark-net spark0
```

Start the first container:

#docker start spark0

Connect to you first node:

docker exec -it spark0 /usr/bin/bash

Start the second node and perform the installation as specify in the first lab.

#docker run -dit --name spark1 -p 8082:8081 -p 4041:4040 --network spark-net --entrypoint /bin/bash centos:7

or

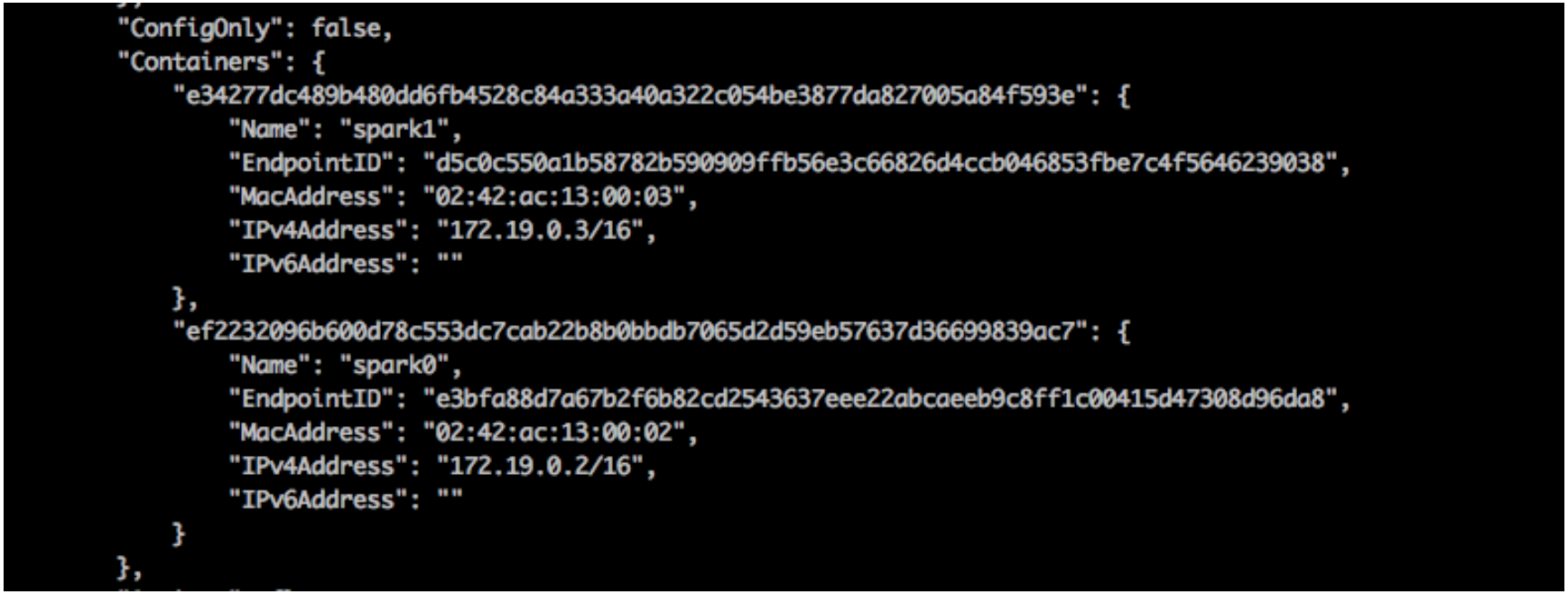
Docker command with Host folder mounted. (Skip)

docker run -it --name spark1 --hostname spark1 --privileged --network spark-net -v /Users/henrypotsangbam/Documents/Docker:/opt -p 4041:4040 -p 8082:8081 centos:7 /usr/sbin/init

Confirm the Network status:

Inspect the network and find the IP addresses of the two containers

```
#docker network inspect spark-net
```

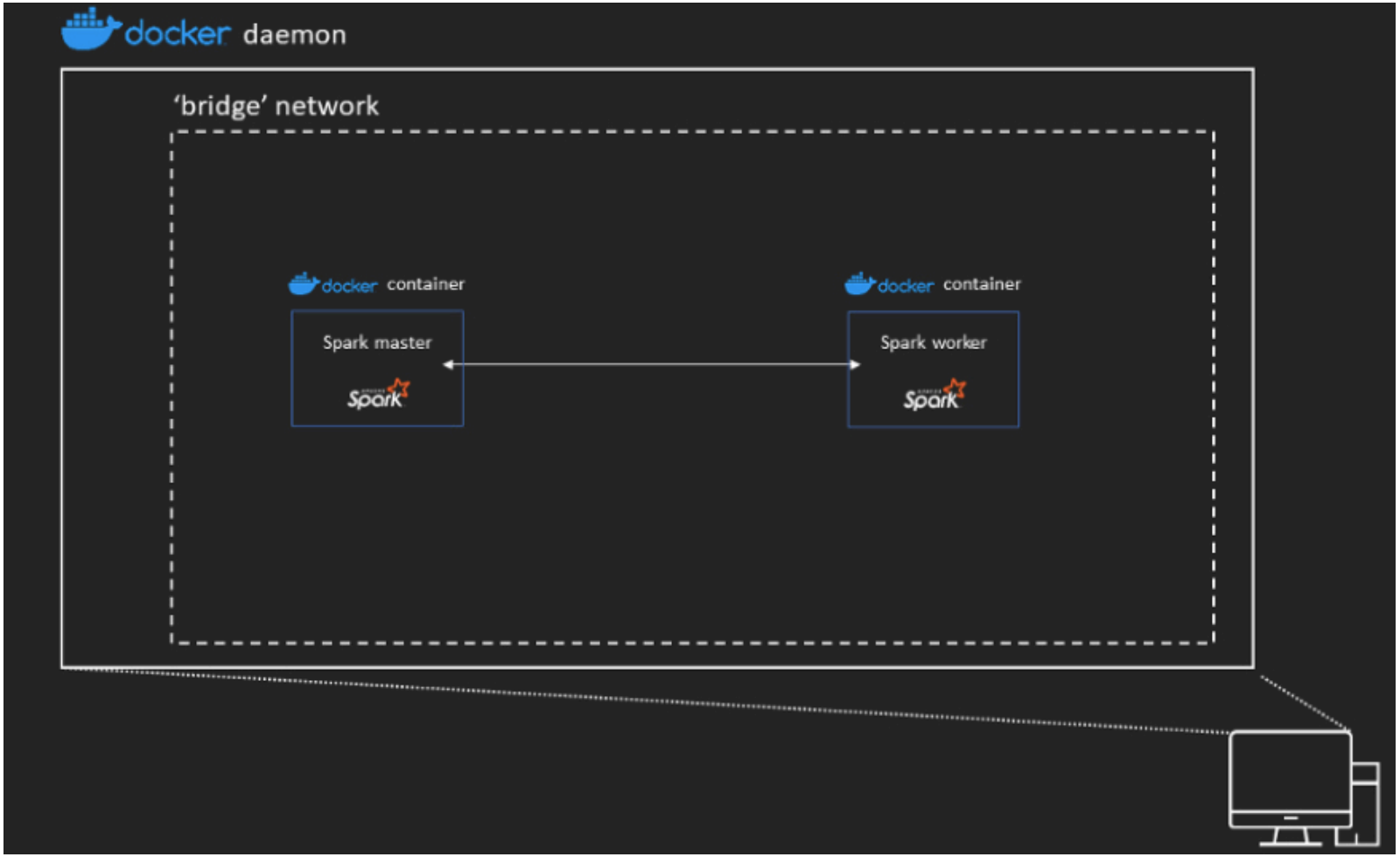


Ensure to substitute IPs accordingly.

Attach to the spark-master container and test its communication to the spark-worker container using both it's IP address and then using its container name.
ping spark0

```
[root@ef2232096b60 /]# ping spark0
PING spark0 (172.19.0.2) 56(84) bytes of data.
64 bytes from ef2232096b60 (172.19.0.2): icmp_seq=1 ttl=64 time=0.062 ms
64 bytes from ef2232096b60 (172.19.0.2): icmp_seq=2 ttl=64 time=0.104 ms
64 bytes from ef2232096b60 (172.19.0.2): icmp_seq=3 ttl=64 time=0.062 ms
^C
--- spark0 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2060ms
rtt min/avg/max/mdev = 0.062/0.076/0.104/0.019 ms
[root@ef2232096b60 /]# ping spark1
PING spark1 (172.19.0.3) 56(84) bytes of data.
64 bytes from spark1.spark-net (172.19.0.3): icmp_seq=1 ttl=64 time=0.202 ms
64 bytes from spark1.spark-net (172.19.0.3): icmp_seq=2 ttl=64 time=0.204 ms
^C
--- spark1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.202/0.203/0.204/0.001 ms
#[root@ef2232096b60 /]#
```

Architecture



master – sparko and Slave – spark1

Let us start the Spark master.

Install the spark. You can refer the first lab.

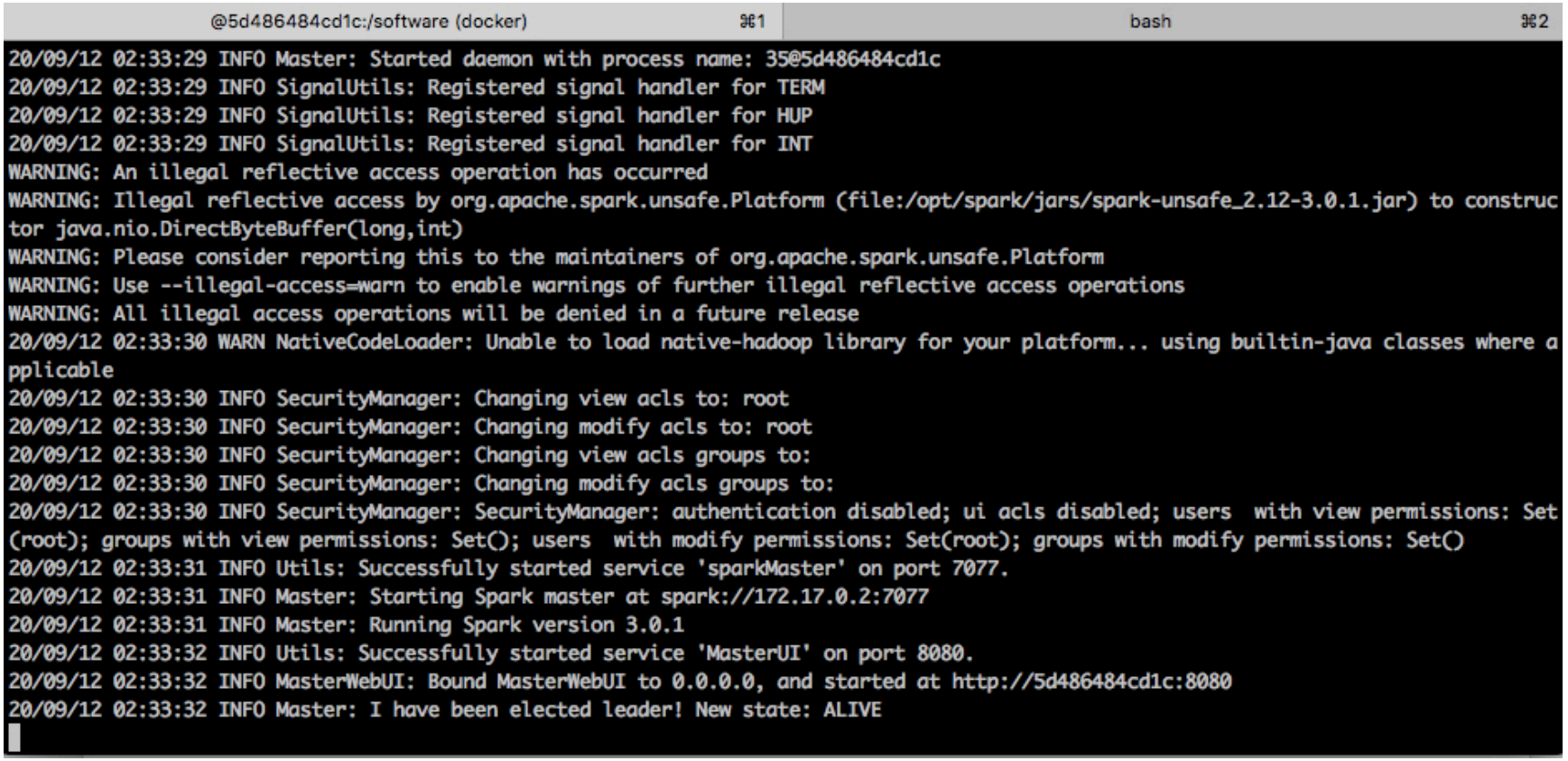
Execute the following in Spark0

Setup a Spark master node

Change the Master Port to 8090, there is issue when it is executed in 7077 in docker environment.

Execute on the spark0 – Terminal.

```
#export PATH=$PATH:/opt/spark/bin
#export SPARK_MASTER_PORT=8090
#spark-class org.apache.spark.deploy.master.Master
```

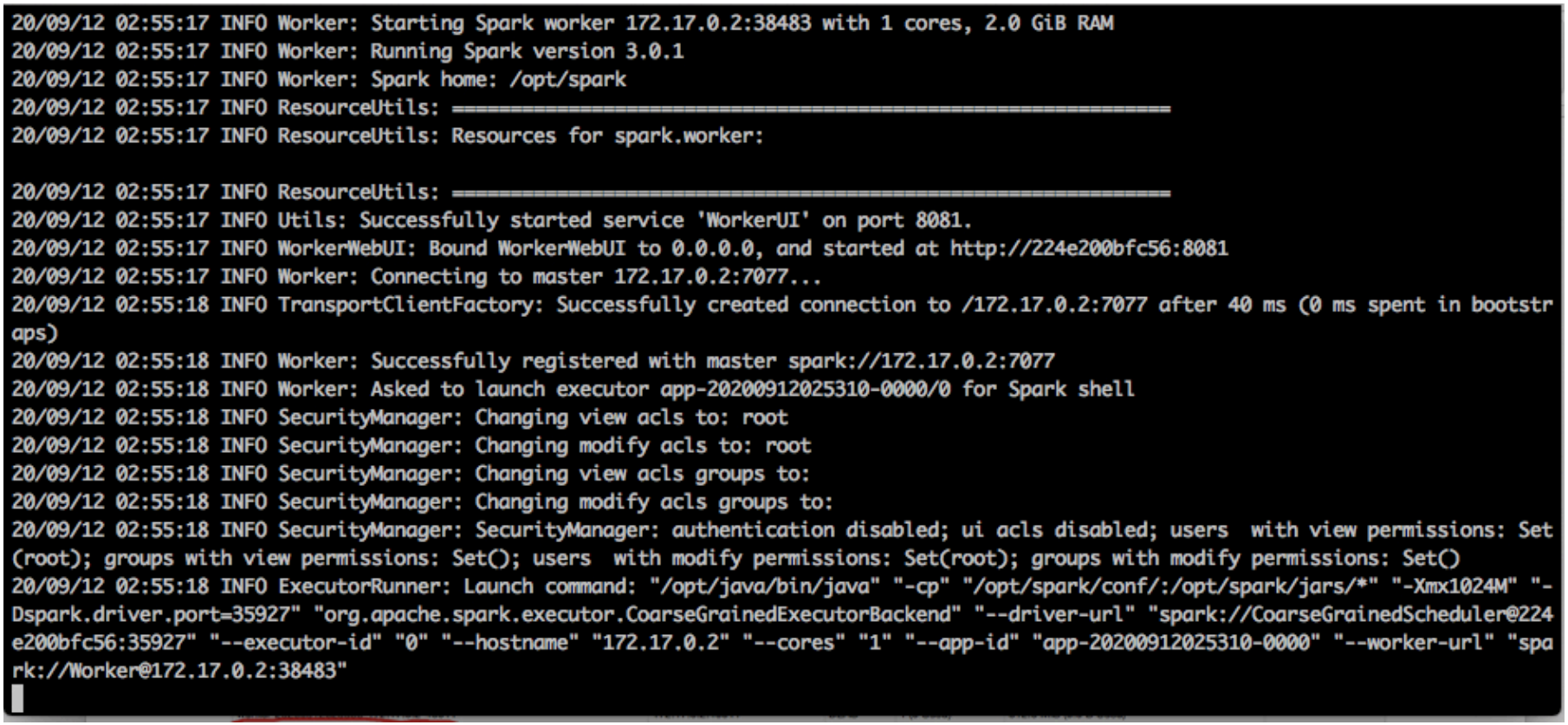


Start a worker process On Node 0: In a separate terminal.

Attach a worker node to the cluster, execute the following in the spark0 container.

#export PATH=$PATH:/opt/spark/bin

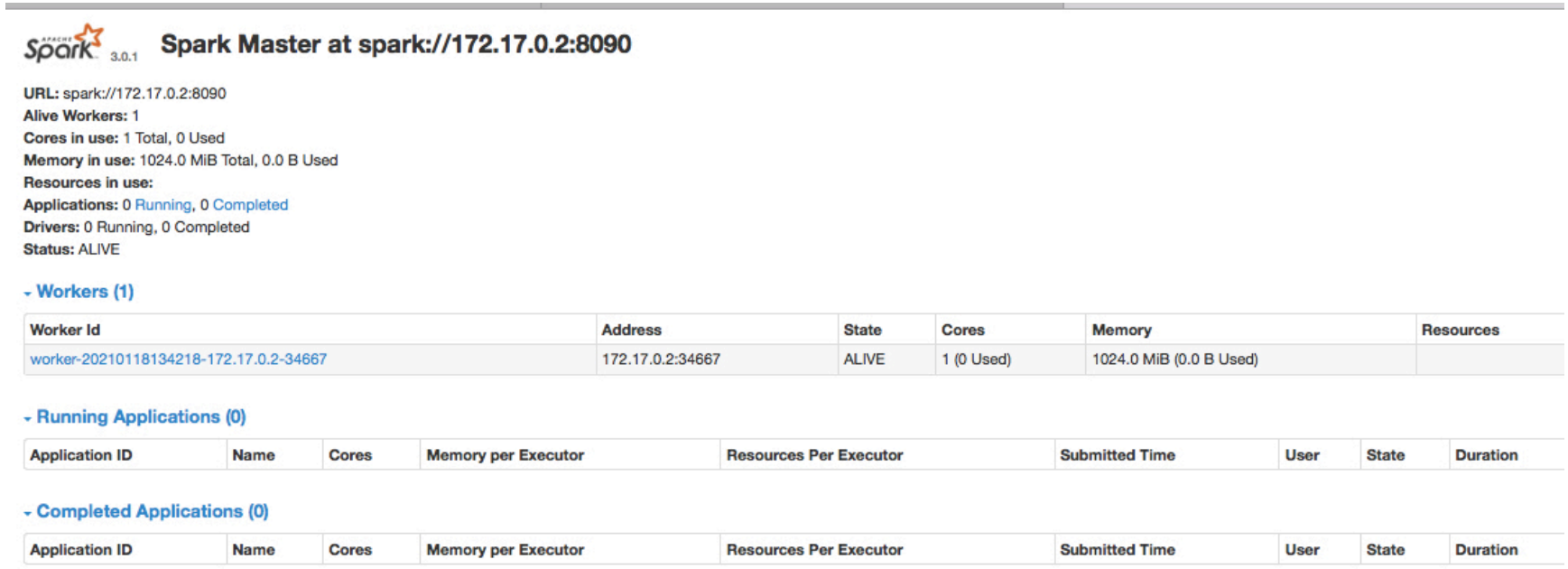
```
#spark-class org.apache.spark.deploy.worker.Worker -c 1 -m 1G spark://172.17.0.2:8090
```



If unable to connect to localhost replace it with the container IP or the container alias i.e sparko.

Refresh the web ui, ensure that you can see a worker as shown below.

http://127.0.0.1:8080



Spark 3.0.1 **Spark Master at spark://172.17.0.2:8090**

URL: spark://172.17.0.2:8090
Alive Workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 1024.0 MiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210118134218-172.17.0.2-34667	172.17.0.2:34667	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Start the Node on spark1

Ensure that you have install spark in the Spark1 too.

#docker exec -it spark1 bash

spark-class org.apache.spark.deploy.worker.Worker -c 1 -m 1G spark://172.17.0.2:8090

At the end of this step, you should have 2 worker nodes as shown below:

Open a pyspark shell and connect to the Spark cluster

Let us execute Spark shell using Cluster mode:

You can perform from any client/server node i.e windows desktop client.

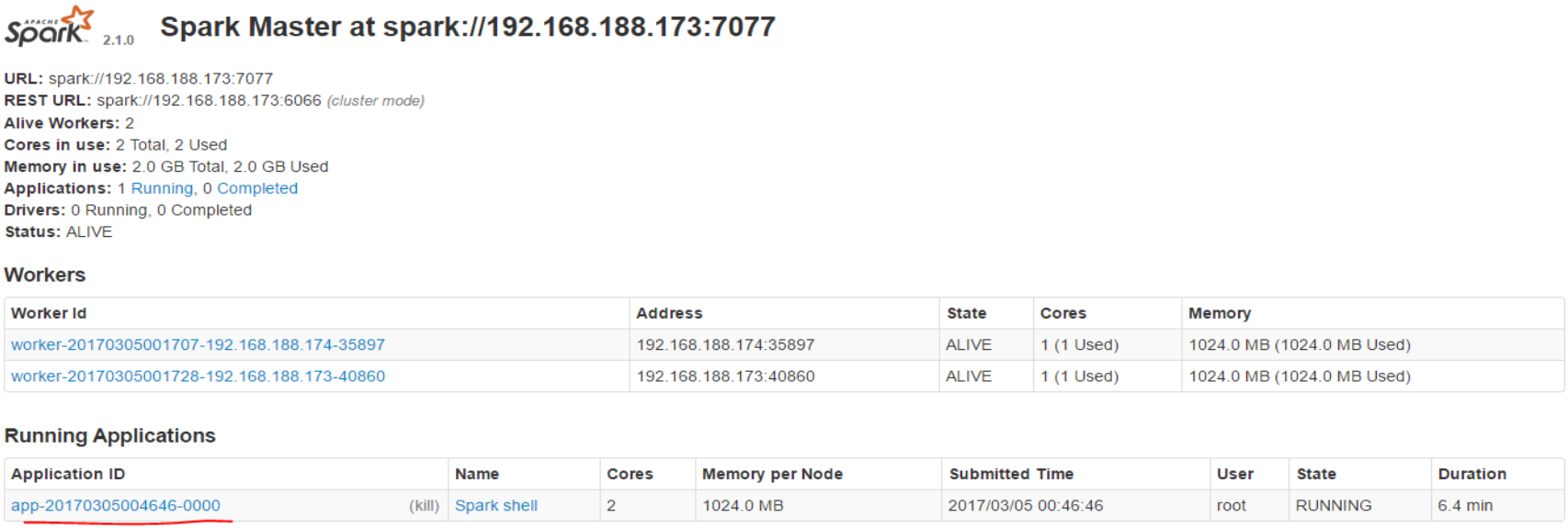
Connecting an Application to the Cluster

To run an application on the Spark cluster, simply pass the spark://IP:PORT URL of the master as to the SparkContext constructor.

To run an interactive Spark shell against the cluster(Specify the master IP), run the following command from the bin folder:
We are executing from the slave node.

#pyspark --conf spark.executor.memory=512mb --conf spark.executor.cores=1 --master spark://spark0:8090

You can verify the application execution from the web UI



Spark 2.1.0 **Spark Master at spark://192.168.188.173:7077**

URL: spark://192.168.188.173:7077
REST URL: spark://192.168.188.173:6066 *(cluster mode)*
Alive Workers: 2
Cores in use: 2 Total, 2 Used
Memory in use: 2.0 GB Total, 2.0 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20170305001707-192.168.188.174-35897	192.168.188.174:35897	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)
worker-20170305001728-192.168.188.173-40860	192.168.188.173:40860	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20170305004646-0000 (kill)	Spark shell	2	1024.0 MB	2017/03/05 00:46:46	root	RUNNING	6.4 min

Let's make a new RDD from the text of the README file in the Spark source directory:

```
textFile = sc.textFile("README.md")
```

```
scala> val textFile  = sc.textFile("/MyTrainingWork/Spark/README.md")
15/06/03 23:32:06 INFO MemoryStore: ensureFreeSpace(73391) called with curMem=18
1810, maxMem=278019440
15/06/03 23:32:06 INFO MemoryStore: Block broadcast_1 stored as values in memory
 (estimated size 71.7 KB, free 264.9 MB)
15/06/03 23:32:06 INFO MemoryStore: ensureFreeSpace(31262) called with curMem=25
5201, maxMem=278019440
15/06/03 23:32:06 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in
memory (estimated size 30.5 KB, free 264.9 MB)
15/06/03 23:32:06 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on l
ocalhost:1599 (size: 30.5 KB, free: 265.1 MB)
15/06/03 23:32:06 INFO BlockManagerMaster: Updated info of block broadcast_1_pie
ce0
15/06/03 23:32:06 INFO SparkContext: Created broadcast 1 from textFile at <conso
le>:21
textFile: org.apache.spark.rdd.RDD[String] = /MyTrainingWork/Spark/README.md Map
PartitionsRDD[3] at textFile at <console>:21
```

Let's start with a few actions:

textFile.count() *// Number of items in this RDD*

```
scala> textFile.count()
15/06/03 23:32:16 INFO FileInputFormat: Total input paths to process : 1
15/06/03 23:32:16 INFO SparkContext: Starting job: count at <console>:24
15/06/03 23:32:16 INFO DAGScheduler: Got job 0 (count at <console>:24) with 2 ou
tput partitions (allowLocal=false)
15/06/03 23:32:16 INFO DAGScheduler: Final stage: Stage 0(count at <console>:24)

15/06/03 23:32:16 INFO DAGScheduler: Parents of final stage: List()
15/06/03 23:32:16 INFO DAGScheduler: Missing parents: List()
15/06/03 23:32:16 INFO DAGScheduler: Submitting Stage 0 (/MyTrainingWork/Spark/R
EADME.md MapPartitionsRDD[3] at textFile at <console>:21), which has no missing
parents
```

```
tes result sent to driver
15/06/03 23:32:17 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in
 352 ms on localhost (1/2)
15/06/03 23:32:17 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in
 338 ms on localhost (2/2)
15/06/03 23:32:17 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have
all completed, from pool
15/06/03 23:32:17 INFO DAGScheduler: Stage 0 (count at <console>:24) finished in
 0.393 s
15/06/03 23:32:17 INFO DAGScheduler: Job 0 finished: count at <console>:24, took
 0.678076 s
res2: Long = 98

scala>
```

textFile.first() *// First item in this RDD*

```
.md:0+1814
15/06/03 23:34:06 INFO Executor: Finished task 0.0 in stage 1.0 (TID 2). 1809 by
tes result sent to driver
15/06/03 23:34:06 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in
 10 ms on localhost (1/1)
15/06/03 23:34:06 INFO DAGScheduler: Stage 1 (first at <console>:24) finished in
 0.011 s
15/06/03 23:34:06 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have
all completed, from pool
15/06/03 23:34:06 INFO DAGScheduler: Job 1 finished: first at <console>:24, took
 0.019875 s
res3: String = # Apache Spark

scala>
```

Now let's use a transformation. We will use the filter transformation to return a new RDD with a subset of the items in the file. Use collect to display the output.

```
linesWithSpark = textFile.filter(lambda line : "Spark" in line)
```

```
linesWithSpark.collect()
```

We can chain together transformations and actions:

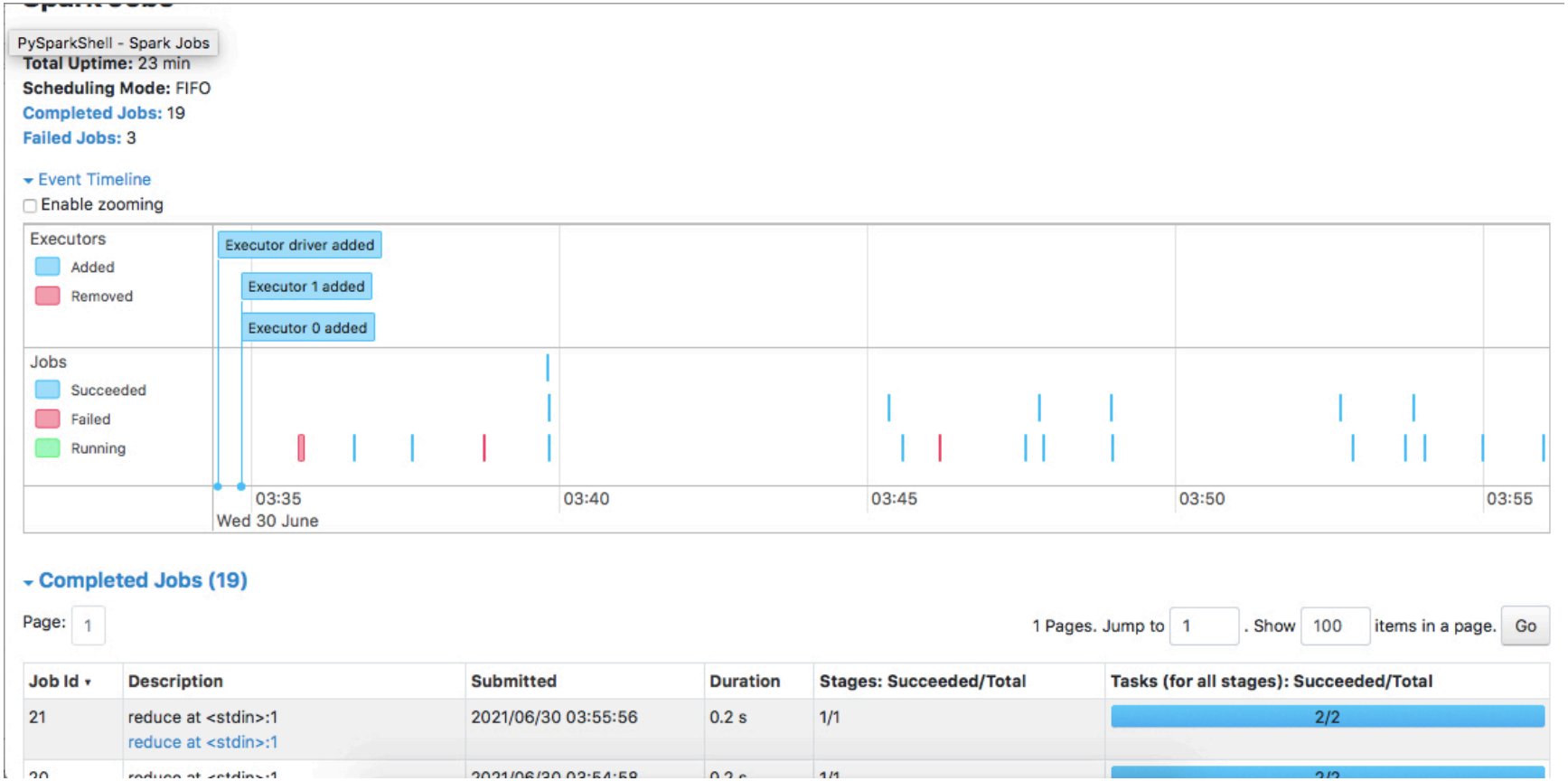
```
textFile.filter(lambda line : "Spark" in line).count()  // How many lines contain "Spark"?
```

```
15/06/03 23:35:17 INFO Executor: Finished task 0.0 in stage 2.0 (TID 3). 1830 by
tes result sent to driver
15/06/03 23:35:17 INFO TaskSetManager: Finished task 1.0 in stage 2.0 (TID 4) in
 12 ms on localhost (1/2)
15/06/03 23:35:17 INFO TaskSetManager: Finished task 0.0 in stage 2.0 (TID 3) in
 14 ms on localhost (2/2)
15/06/03 23:35:17 INFO DAGScheduler: Stage 2 (count at <console>:24) finished in
 0.014 s
15/06/03 23:35:17 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have
all completed, from pool
15/06/03 23:35:17 INFO DAGScheduler: Job 2 finished: count at <console>:24, took
 0.028214 s
res4: Long = 19

scala>
```

Let's say we want to find the line with the most words

```
textFile.map(lambda line : len(line.split(" "))). reduce(lambda a, b: a if (a > b) else b)
```

Refer the web UI for the Job details as shown below.



The console should display the result as shown below:

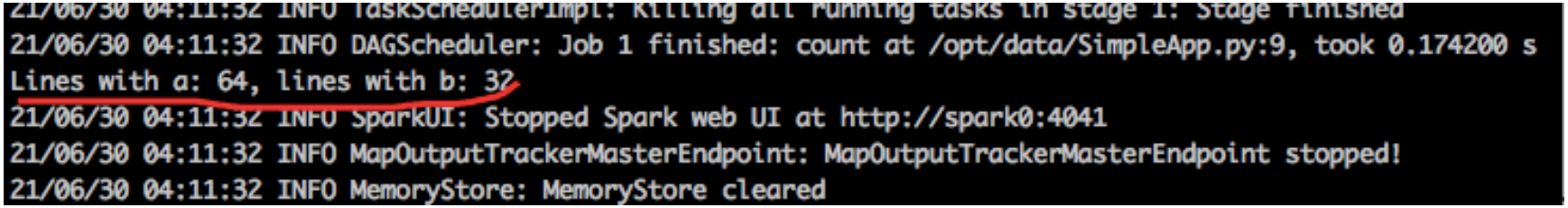
Hints: (For submitting job in cluster)

You need to complete the **Lab - Running a Spark Application** before going ahead. Submit from the /software folder, since the input file exist in that folder.

You need to copy the data files in both the nodes. However, for our lab its already present in the spark folder.

#spark-submit --master spark://spark0:8090 workspace/SimpleApp.py --conf spark.executor.memory=256mb

Verify the output.



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

https://towardsdatascience.com/diy-apache-spark-docker-bb4f11c10d24

22. Launching on a Cluster: Hadoop YARN – 150 Minutes

In this lab, you will deploy Spark Application on Hadoop Cluster.

Configure and install YARN.
One Node will be exclusively running YARN,
Options for deploying Hadoop:

- Copy a VM and rename to HadoopVM-YARN.
- Create another container

Install YARN – On the New Node.

By now, you should have two VM on your workstation or Two containers:

For Me, the first VM hostname is hp.com and the second one is ht.com. Ensure to follow the same nomencleature to avoid confusion. Its very important.

Host/VM	Container	Remarks
hp.com	Hadoopo	YARN Services
ht.com	Sparko	Spark Client or Spark.

Using Docker:
#docker run -it --name hadoop0 --privileged -p 8088:8088 -p 9870:9870 -p 9864:9864 -p 8032:8032 -p 8188:8188 -p 8020:8020 --hostname hadoop0 centos:7 /usr/sbin/init

Verify the hostname and the /etc/hosts. You need to enter each IP and host name as shown above. Update details accordingly in your local machine.

```
[root@hp ~]# hostname
hp.com
[root@hp ~]# more /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
#127.0.0.1               hp.com localhost
#::1                     hp.com localhost6

192.168.188.134 ht.com
192.168.188.136 hp.com localhost
[root@hp ~]#
```

You should see in your window as follows for ht.com. Changes the IP as your system.

```
[root@ht ~]# bash
[root@ht ~]# hostname
ht.com
[root@ht ~]# more /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
#127.0.0.1               localhost.localdomain localhost
#::1             localhost6.localdomain6 localhost6
192.168.188.134 ht.com localhost
192.168.188.136 hp.com
[root@ht ~]#
```

All the below commands should be executed on hp.com (Hadoop Node) unless specify. We are configuring YARN cluster now.

Change directory to the location where you have the software.

Download Location: (hadoop-3.2.2.tar.gz)
https://hadoop.apache.org/releases.html

Untar as follows:
tar -xvf hadoop-X.tar.gz -C /opt

tar -xvf jdk-8u40-linux-i586.tar.gz -C /opt

Rename the folder:

#mv hadoo* hadoop
#mv jdk* jdk
--- Install JDK and set Java Home (Copy the bin file in the YARN folder) , To include JAVA_HOME for all bash users , make an entry in /etc/profile.d as follows:

echo "export JAVA_HOME=/opt/jdk/" > /etc/profile.d/java.sh

Unpack the downloaded Hadoop distribution. In the distribution, edit the file /opt/hadoop/`etc/hadoop/hadoop-env.sh` to define some parameters as follows:

```
# set to the root of your Java installation

export JAVA_HOME=/opt/jdk
```

cd /opt/hadoop

Try the following command:

```
$ bin/hadoop
```

You need to modify some setting as follows: Replace with your hostname accordingly.

Use the following:

etc/hadoop/core-site.xml:

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://hadoop0:8020</value>
    </property>
</configuration>
```

etc/hadoop/hdfs-site.xml:

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

Setup passphraseless ssh

Now check that you can ssh to the localhost without a passphrase:

```
yum -y install openssh-server openssh-clients
systemctl start sshd
```

Change the passwd using passwd command.

```
$ ssh localhost
```

If you cannot ssh to localhost without a passphrase, execute the following commands:

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa

$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

$ chmod 0600 ~/.ssh/authorized_keys
```

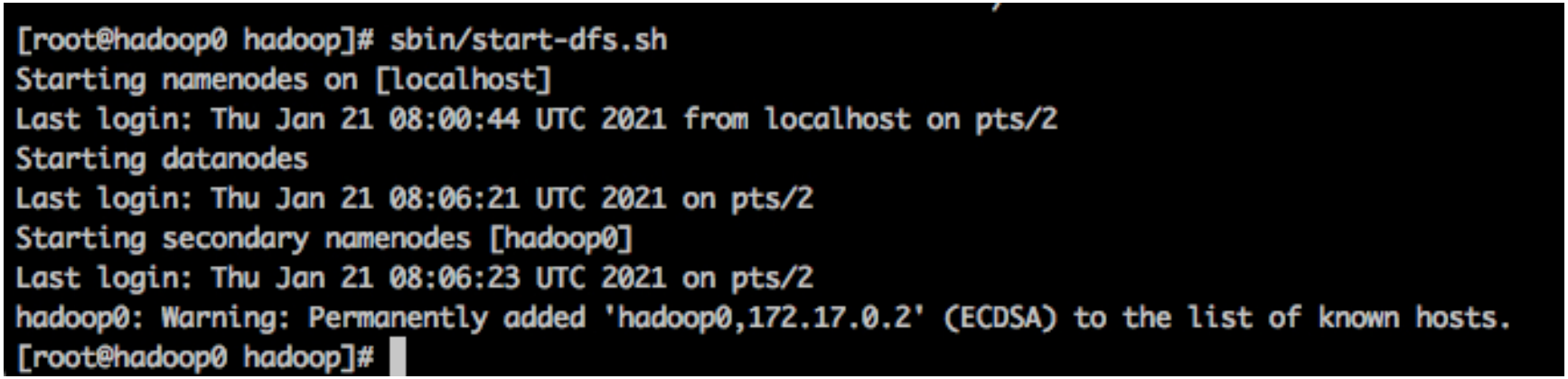
```
export HDFS_NAMENODE_USER="root"
export HDFS_DATANODE_USER="root"
export HDFS_SECONDARYNAMENODE_USER="root"
export YARN_RESOURCEMANAGER_USER="root"
export YARN_NODEMANAGER_USER="root"
```

Format the filesystem:

```
$ bin/hdfs namenode -format
```

Start NameNode daemon and DataNode daemon:

```
$ sbin/start-dfs.sh
```



1. Browse the web interface for the NameNode; by default it is available at:
   - NameNode - `http://localhost:9870/`
2. Make the HDFS directories required to execute MapReduce jobs:

```
3.   $ bin/hdfs dfs -mkdir /user
```

```
  $ bin/hdfs dfs -mkdir /user/root
```

You can run a MapReduce job on YARN in a pseudo-distributed mode

Start HDFS and YARN on the YARN Node: hp.com

1. Configure parameters as follows:

   `etc/hadoop/mapred-site.xml`:

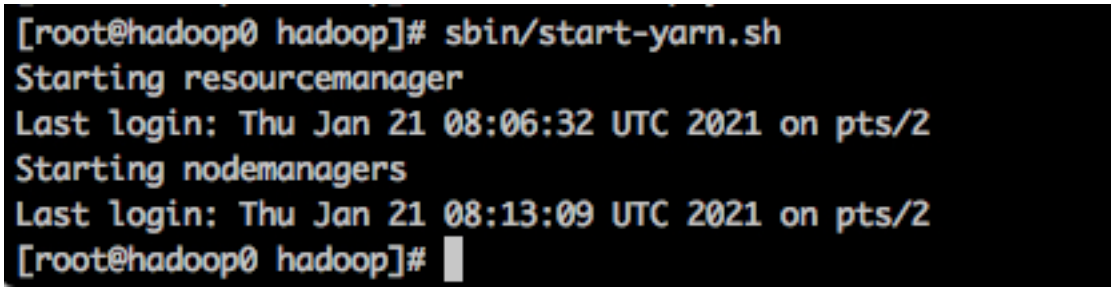
```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*
</value>
    </property>
</configuration>
```

etc/hadoop/yarn-site.xml:

```
<configuration>
<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HAD
OOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
    </property>
</configuration>
```

2. Start ResourceManager daemon and NodeManager daemon:

```
$ sbin/start-yarn.sh
```



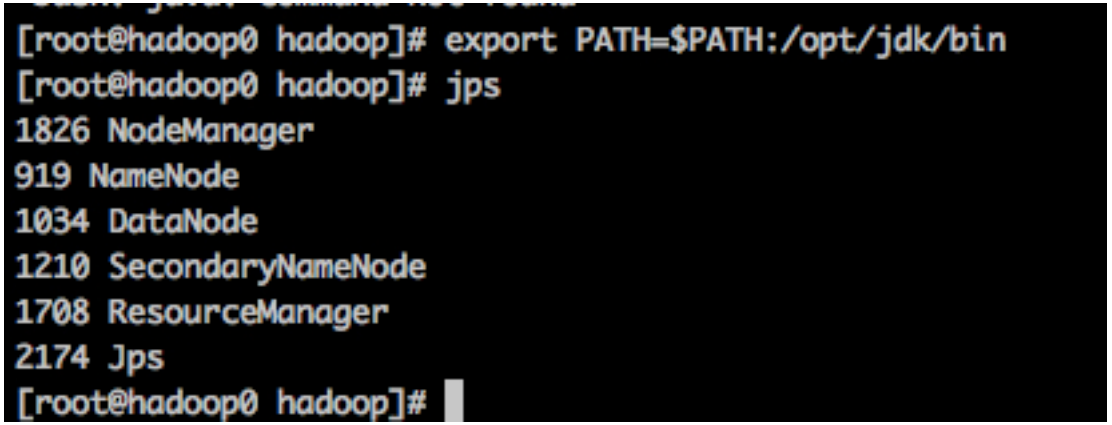
3. Browse the web interface for the ResourceManager; by default it is available at:

   - ResourceManager - `http://localhost:8088/`

4. When you're done, you can stop the daemons with: Optional.

```
$ sbin/stop-yarn.sh
```

#verify with jps , all the above services should be there.
jps



Hadoop Node - Let us set some environment and path variable as follows: (vi ~/.bashrc)

```
export HADOOP_HOME=/opt/hadoop
export PATH=$HADOOP_HOME/bin:$PATH
```

Let us create a temporary folder, that will be used for working space for the YARN cluster.

```
hadoop fs -mkdir /tmp
hadoop fs -chmod -R 1777 /tmp
hadoop fs -mkdir /tmp/in
hadoop fs -ls /tmp
```

You can get the README.md file from the Software folder, which is provided along with the training.

```
hadoop fs -copyFromLocal /opt/hadoop/README.txt /tmp/in
hadoop fs -ls /tmp/in
```

```
hadoop fs -mkdir  /tmp/spark-events
```

Congrats! You have successfully configure Yarn Cluster.

If you are using docker, join the hadoop0 and spark0 containers in a single network:

```
#docker network connect spark-net hadoop0
```

Verify it:

```
#docker network inspect spark-net
```

```
        "Containers": {
            "303b681436449f211ee3f828ec09af683947652bd1db52c68288e6ac75d71e04": {
                "Name": "spark0",
                "EndpointID": "56cfeed9c11107e578a688ddab77af96e72c1063ebd2631bd801629f328b17e2",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            },
            "c88c1c8cb01a6aedc1832d1bf3d3cb22dec209f45a5749ace85a66d1d1d0cf12": {
                "Name": "hadoop0",
                "EndpointID": "3c030821125613a7bf5c1cea99f510144a01220b9cc38391bbeb71274076c824",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Start the Spark VM/spark container, if it's not yet started, i.e ht.com and issue the following command.
Logon to the machine using telnet. You can use the VM, share folder options to copy the file to the VM from your workstation or else copy using the mouse from your workstation to the folder specify.

Let us configure hadoop client setting on the Spark VM --> ht.com or spark0 node.

Compress the Hadoop folder on the Hadoop0 instances or Hadoop node.
#cd /opt
#tar -cvzf hadoop.tar hadoop/

Copy the compressed hadoop folder to the spark0 or spark node and uncompressed in /opt folder.
[Docker:
copy from hadoop to host : docker cp hadoop0:/opt/hadoop.tar .
copy from host to spark : docker cp hadoop.tar spark0:/opt/]

#tar -xvf hadoop.tar

Update yarn-site.xml on the Spark Node.

```
<property>
        <name>yarn.resourcemanager.hostname</name>
        <value>hadoop0</value>
        <description>The hostname of the RM.</description>
    </property>
<property>
        <name>yarn.resourcemanager.address</name>
        <value>hadoop0:8032</value>
```

```
    <description>The hostname of the RM.</description>
  </property>
```

Create the following folder on HDFS cluster and copy the file in the folder.
hdfs dfs -mkdir -p /opt/spark
hdfs dfs -copyFromLocal /opt/spark/README.md /opt/spark/

You can verify the file with the following command. This command will display the content of the file in the HDFS cluster.

hdfs dfs -cat /opt/spark/README.md

Open a command console to execute the Spark jobs to submit on YARN.

Ensure that you install python on Hadoop node:
yum install -y python3

Export Hadoop and Yarn environment variable pointing to the Spark Node – Hadoop conf file.

Create a python app, SimpleApp.py with the following code:

// ---------- Code Begin ----------//

```
"""SimpleApp.py"""
from pyspark.sql import SparkSession

logFile = "/opt/spark/README.md"  # Should be some file on your system

spark   = SparkSession.builder.appName("Python App").getOrCreate()
logData = spark.read.text(logFile).cache()

numAs = logData.filter(logData.value.contains('a')).count()
numBs = logData.filter(logData.value.contains('b')).count()

print("Lines with a: %i, lines with b: %i" % (numAs, numBs))

spark.stop()
```

// ---------- Code Ends ----------//

The above code reads the README.md file from HDFS cluster and counts the line containing ‘a’ and ‘b’ respectively.

```
#export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop
#export YARN_CONF_DIR=/opt/hadoop/etc/hadoop
#./spark-submit  --master yarn --deploy-mode cluster --executor-memory 512m --num-executors 2 SimpleApp.py
```

or client side.

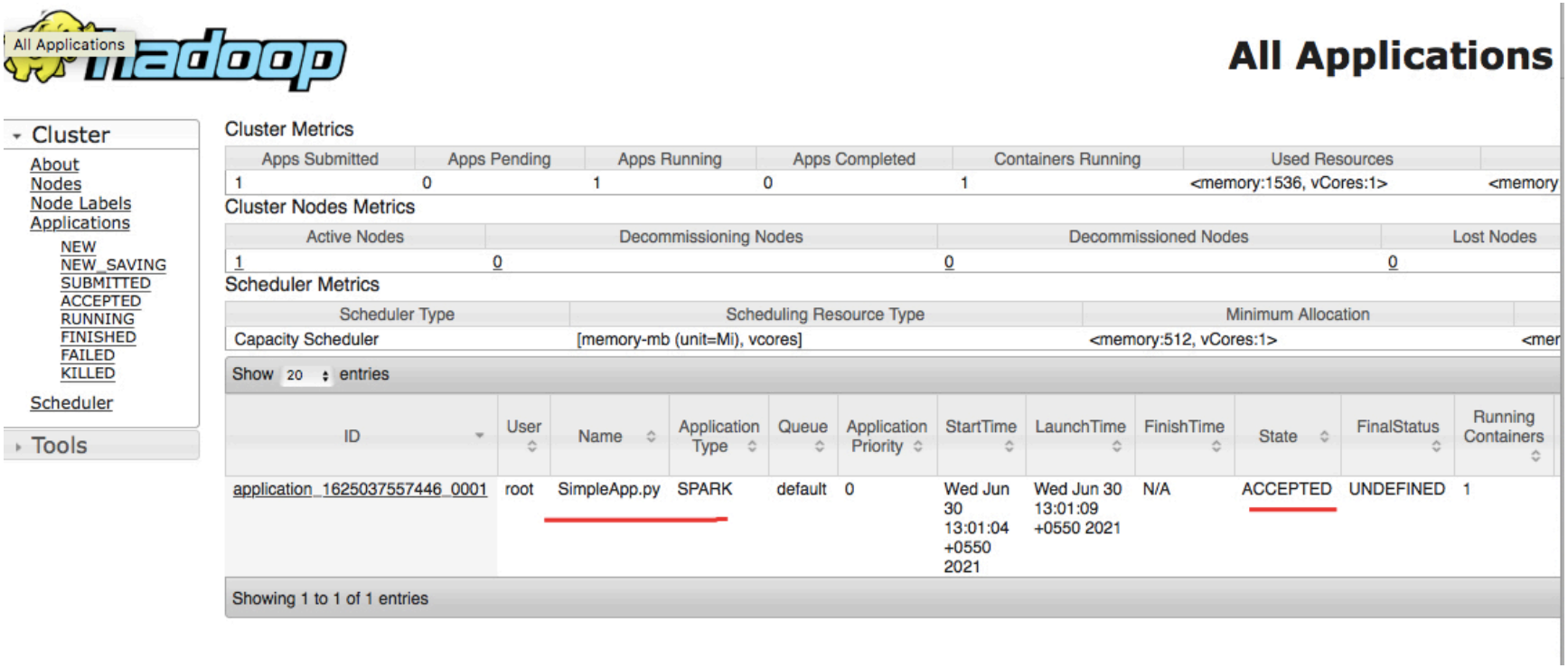
```
#./spark-submit  --master yarn --deploy-mode client /opt/data/SimpleAppP.py
```

Ensure that `HADOOP_CONF_DIR` or `YARN_CONF_DIR` points to the directory which contains the (client side) configuration files for the Hadoop cluster.

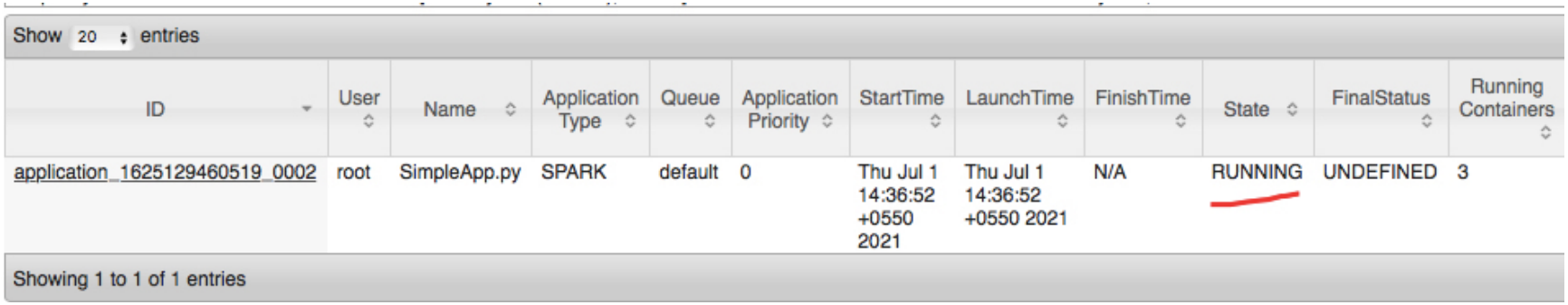
Note --> Application py/Jar should be in the local file system, In folder is in HDFS and Out should not be present in the HDFS,it will be automatically created.

You can verify the progress of the application using http://hp.com:8088/cluster

Click on Cluster-- >Application --> accepted



Wait for few moment and verify on Running tab.

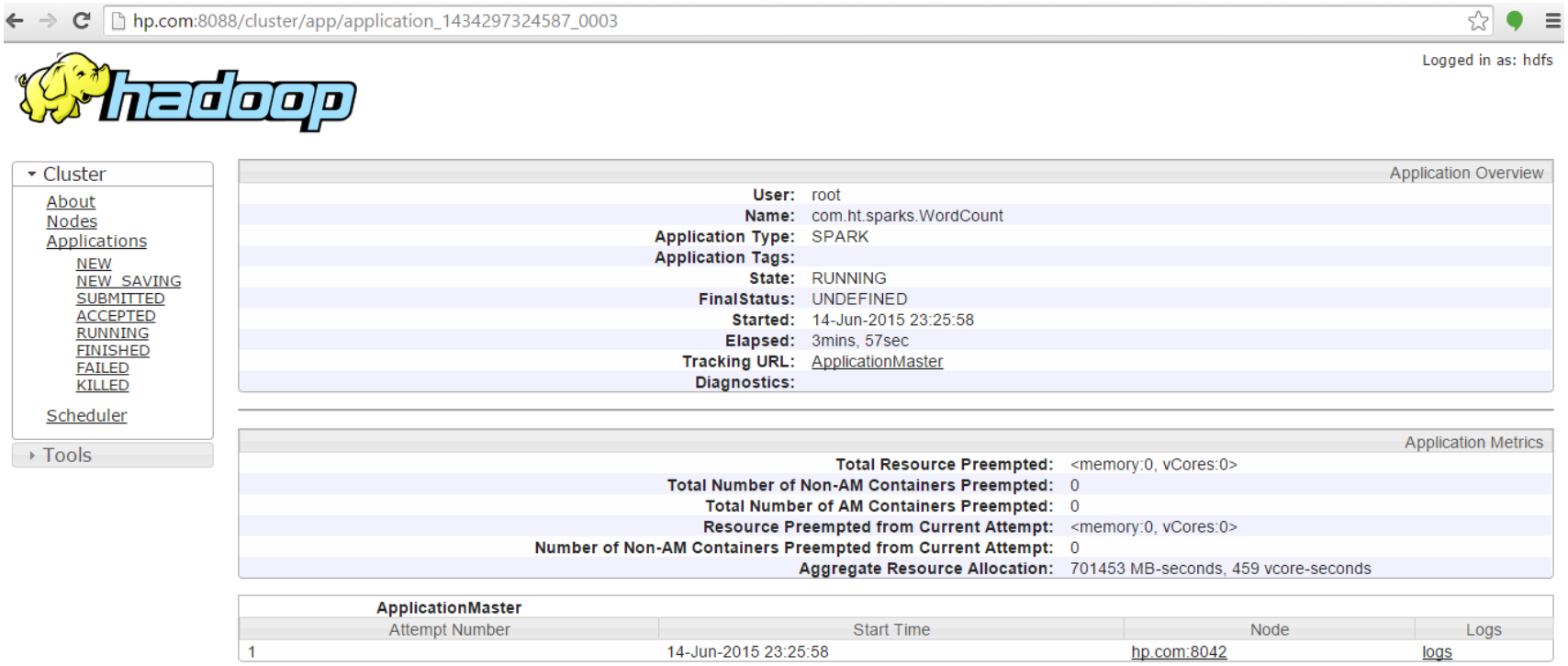


Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers
application_1625129460519_0002	root	SimpleApp.py	SPARK	default	0	Thu Jul 1 14:36:52 +0550 2021	Thu Jul 1 14:36:52 +0550 2021	N/A	RUNNING	UNDEFINED	3

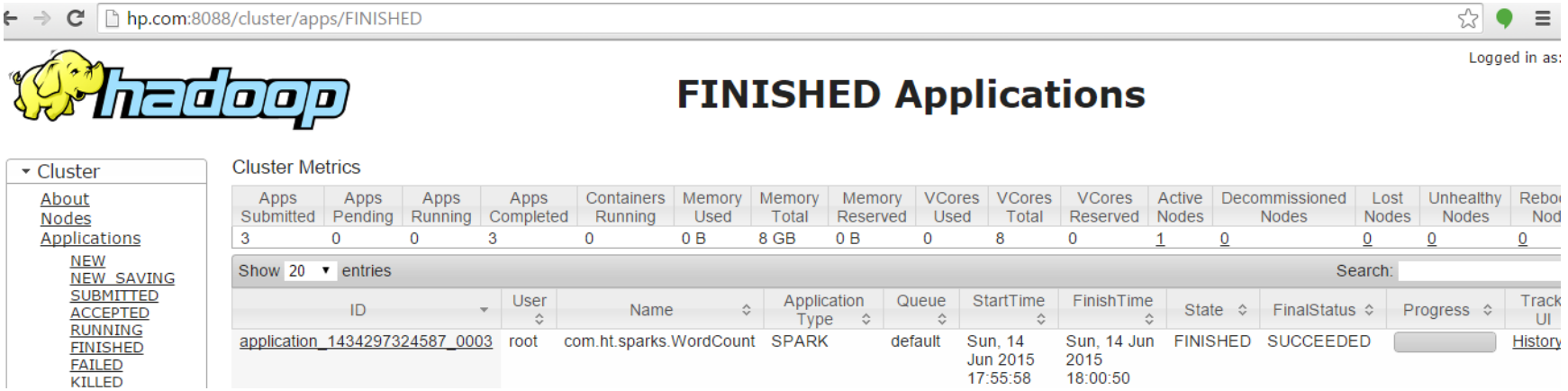
Showing 1 to 1 of 1 entries

Click on the Application ID --> Logs



Logs --> Stderr to get details as follows:

After sometimes click on Finished, after the console exit the program.



On the job submit console.

```
2021-01-22 16:36:11,589 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:12,662 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:13,765 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:14,896 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:16,011 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:17,296 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:18,725 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:19,914 INFO yarn.Client: Application report for application_1611331123305_0002 (state: RUNNING)
2021-01-22 16:36:21,041 INFO yarn.Client: Application report for application_1611331123305_0002 (state: FINISHED)
2021-01-22 16:36:21,401 INFO yarn.Client:
         client token: N/A
         diagnostics: N/A
         ApplicationMaster host: hadoop0
         ApplicationMaster RPC port: 38603
         queue: default
         start time: 1611333080086
         final status: SUCCEEDED
         tracking URL: http://hadoop0:8088/proxy/application_1611331123305_0002/
         user: root
2021-01-22 16:36:23,265 INFO util.ShutdownHookManager: Shutdown hook called
2021-01-22 16:36:23,442 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-7b20b446-2b75-4550-9e2d-c84ad6
7c9b3b
2021-01-22 16:36:23,532 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-89770943-febc-4bdb-bf6e-404103
88026d
[root@303b68143644 bin]#
```

Since we have print the output. You can verify from the log fie only.

Go to the User Log folders of the hadoop installation. (Job ID and the container ID will be different in your machine, update acordingly)

#cd /opt/hadoop/logs/userlogs
#more application_1625129460519_0004/container_1625129460519_0004_01_000001/stdout

```
[root@hadoop0 userlogs]# pwd
/opt/hadoop/logs/userlogs
[root@hadoop0 userlogs]# more application_1625129460519_0004/container_1625129460519_0004_01_000001/stdout
Lines with a: 64, lines with b: 32
[root@hadoop0 userlogs]#
```

Task -> Let us write the output into file instead of writing in console.

Create a file SimpleAppP.py and update with the following code. It only writes the dataframe which has the line having 'a' in it.
//------------------------- Code Begin ---------------------

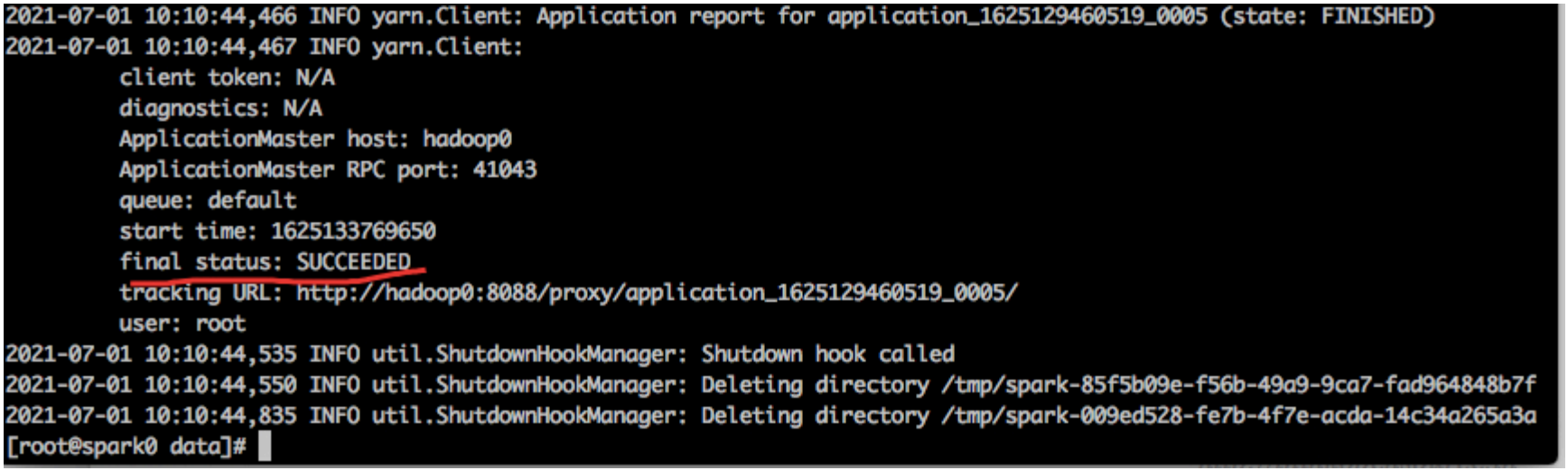
```
"""SimpleApp.py"""
from pyspark.sql import SparkSession

logFile = "/opt/spark/README.md"  # Should be some file on your system

spark   = SparkSession.builder.appName("Python App").getOrCreate()
logData = spark.read.text(logFile).cache()

numAs = logData.filter(logData.value.contains('a')).count()
numBs = logData.filter(logData.value.contains('b')).count()
```

```
#print("Lines with a: %i, lines with b: %i" % (numAs, numBs))

myDF = logData.filter(logData.value.contains('a'))
myDF.write.save("mydatap")
spark.stop()

//------------------------- Code Ends ----------------------
```

Execute the program again.

#spark-submit --master yarn --deploy-mode cluster --executor-memory 512m --num-executors 2 SimpleAppP.py

At the end of the execution, you should have the following output in the console.



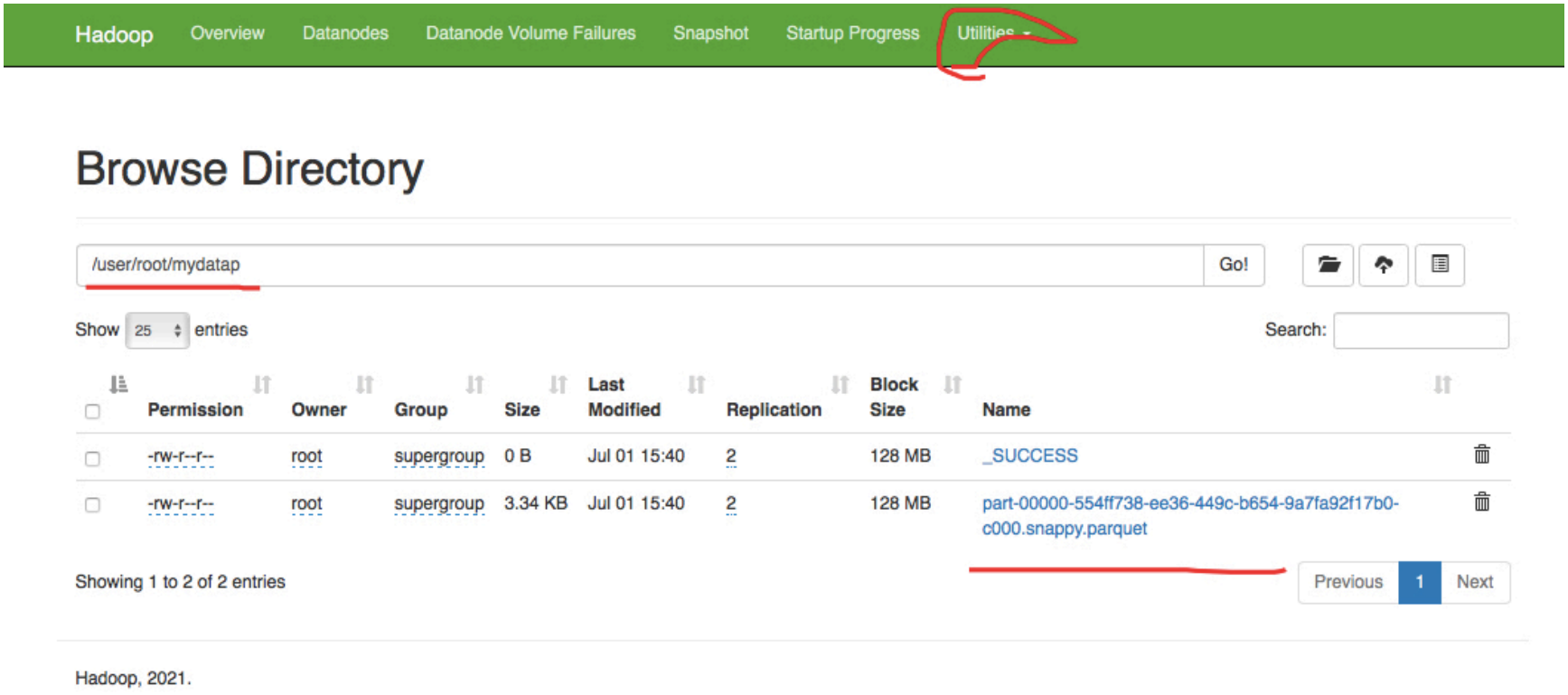
You can verify the output file in the hdfs as shown below:
#hdfs dfs -ls -R /user/root/mydatap
 # hdfs dfs -cat /user/root/mydatap/part-00000-554ff738-ee36-449c-b654-9a7fa92f17b0-c000.snappy.parquet

It's a binary parquet file.

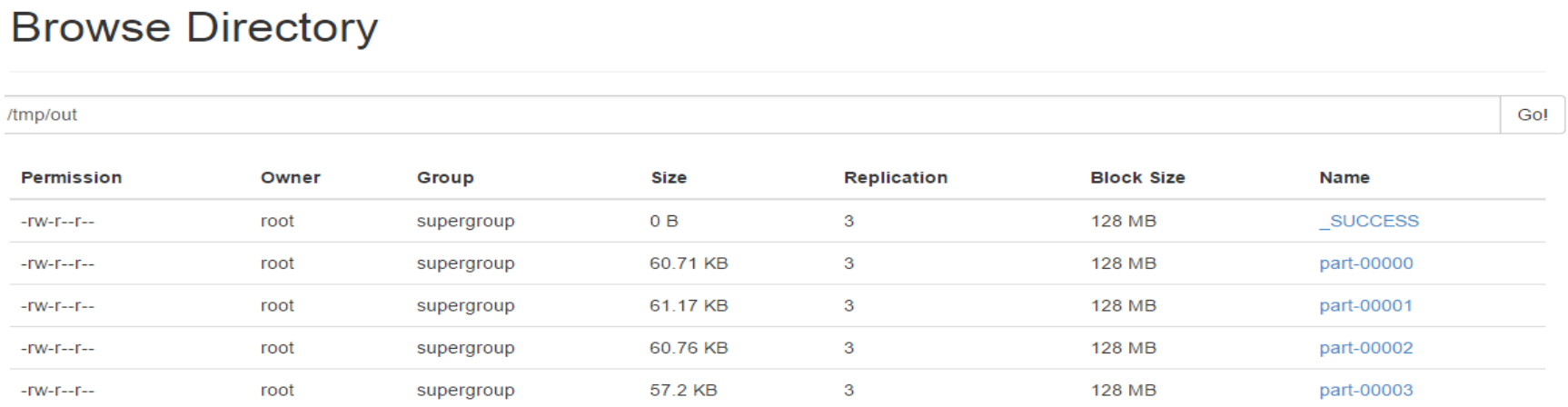
Great! You have successfully executed spark job in YARN cluster.
Let us verify the output now using HDFS browser.

http://localhost:9870/explorer.html#/

Click Utilities --> Browse the file system → /user/root/output (Enter in the Directory) - Go



Click on the above mark file.



Let us view one of the output. click on part-0001 --> Download.

Congrats!.

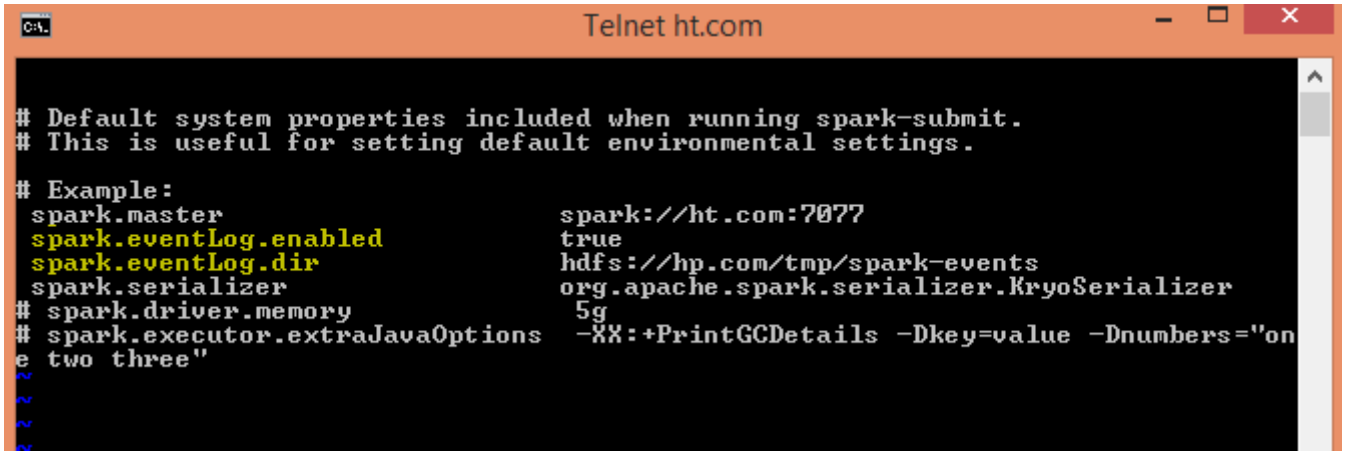
---------- End of Lab----------

Errata:

1) 15/06/14 21:50:46 INFO storage.BlockManagerMasterActor: Registering block manager ht.com:50475 with 267.3 MB RAM, BlockManagerId(<driver>, ht.com, 50475)
15/06/14 21:50:46 INFO storage.BlockManagerMaster: Registered BlockManager
Exception in thread "main" java.lang.IllegalArgumentException: Wrong FS: file:/tmp/spark-events/application_1434297324587_0001.inprogress, expected: hdfs://hp.com at org.apache.hadoop.fs.FileSystem.checkPath(FileSystem.java:643)
at org.apache.hadoop.hdfs.DistributedFileSystem.getPathName(DistributedFileSystem.java:191)
at org.apache.hadoop.hdfs.DistributedFileSystem.access$000(DistributedFileSystem.java:102)
at org.apache.hadoop.hdfs.DistributedFileSystem$22.doCall(DistributedFileSystem.java:1266)
at org.apache.hadoop.hdfs.DistributedFileSystem$22.doCall(DistributedFileSystem.java:1262)
at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.java:81)
at org.apache.hadoop.hdfs.DistributedFileSystem.setPermission(DistributedFileSystem.java:1262)
at org.apache.spark.scheduler.EventLoggingListener.start(EventLoggingListener.scala:128)

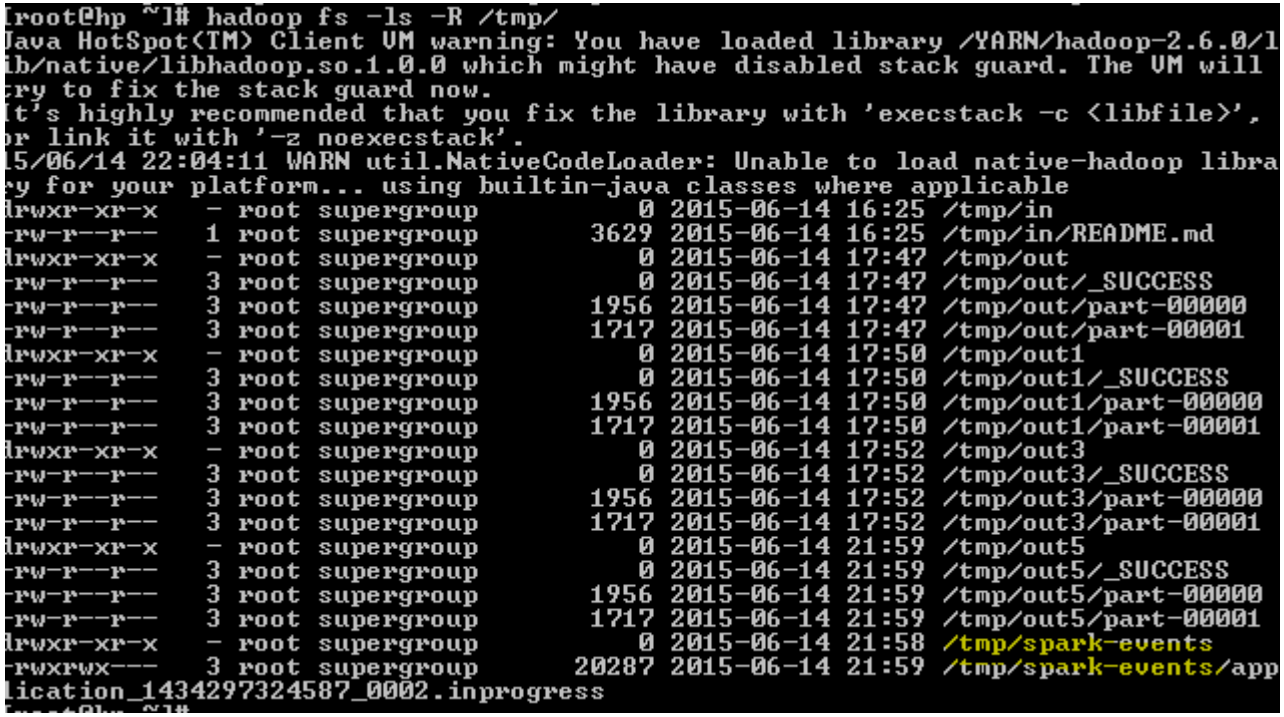
```
15/06/14 21:50:46 INFO storage.BlockManagerMasterActor: Registering block manage
r ht.com:50475 with 267.3 MB RAM, BlockManagerId(<driver>, ht.com, 50475)
15/06/14 21:50:46 INFO storage.BlockManagerMaster: Registered BlockManager
Exception in thread "main" java.lang.IllegalArgumentException: Wrong FS: file:/t
mp/spark-events/application_1434297324587_0001.inprogress, expected: hdfs://hp.c
om
        at org.apache.hadoop.fs.FileSystem.checkPath(FileSystem.java:643)
        at org.apache.hadoop.hdfs.DistributedFileSystem.getPathName(DistributedF
ileSystem.java:191)
        at org.apache.hadoop.hdfs.DistributedFileSystem.access$000(DistributedFi
leSystem.java:102)
        at org.apache.hadoop.hdfs.DistributedFileSystem$22.doCall(DistributedFil
eSystem.java:1266)
        at org.apache.hadoop.hdfs.DistributedFileSystem$22.doCall(DistributedFil
eSystem.java:1262)
        at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkRes
olver.java:81)
        at org.apache.hadoop.hdfs.DistributedFileSystem.setPermission(Distribute
dFileSystem.java:1262)
        at org.apache.spark.scheduler.EventLoggingListener.start(EventLoggingLis
tener.scala:128)
        at org.apache.spark.SparkContext.<init>(SparkContext.scala:399)
        at org.apache.spark.api.java.JavaSparkContext.<init>(JavaSparkContext.sc
ala:61)
```

Solution: Verify default configuration of the spark installation. /spark/spark-1.3.0-bin-hadoop2.4/conf/spark-defaults.conf



Check that spark.eventLog.dir is begin with hdfs:// and the /tmp/spark-events is already created in HDFS system
You can verify on the YARN cluster only.

Verify using : hadoop fs -ls -R /tmp/



2) 15/06/14 00:41:20 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/06/14 21:58:27 INFO yarn.Client: Application report for application_1434297324587_0002 (state: ACCEPTED)
Indefinite loop of above
Solution: Ensure that all configuration file are map to appropriate hostname and hostname to ip details are modified in the hosts file , Increase the resources and wait for few minutes to convert the status to running.

```
Yarn-site.xml
yarn.scheduler.minimum-allocation-mb: 256m
yarn.scheduler.increment-allocation-mb: 256m
```

```
<property>
<name>yarn.nodemanager.resource.memory-mb</name>
<value>4096</value> <!-- 4 GB -->
</property>
<property>
<name>yarn.scheduler.minimum-allocation-mb</name>
<value>512</value> <!-- 1 GB -->
</property>
<property>
<name>yarn.scheduler.increment-allocation-mb</name>
<value>256</value> <!-- 1 GB -->
</property>
```

yarn-site.xml – Spark Node. (Specify the hostname of the Hadoop RM Node as shown below)

```
<property>
      <name>yarn.resourcemanager.hostname</name>
      <value>hadoop0</value>
      <description>The hostname of the RM.</description>
   </property>
<property>
      <name>yarn.resourcemanager.address</name>
      <value>hadoop0:8032</value>
      <description>The hostname of the RM.</description>
```

</property>

3) If unhealthy nodes are being displayed in the cluster URL with 1/1 local-dirs are bad: /tmp/hadoop-yarn/nm-local-dir;

Solution: delete the folder using the following command and ensure that nodes are healthy before proceeding forward, you can restart if require

hadoop fs -rm -fr /tmp/hadoop-yarn/*

Try the following command:
yarn application -list
yarn application -kill [application name]

You can view jps in YARN cluster when spark job is executing. It will start Coarse* processes.

Even, cluster mode works

```
15/06/14 23:43:46 INFO yarn.Client: Application report for application_143429732
4587_0004 (state: FINISHED)
15/06/14 23:43:46 INFO yarn.Client:
         client token: N/A
         diagnostics: N/A
         ApplicationMaster host: hp.com
         ApplicationMaster RPC port: 0
         queue: default
         start time: 1434305512588
         final status: SUCCEEDED
         tracking URL: http://hp.com:8088/proxy/application_1434297324587_0004/A
         user: root
[root@ht spark-1.3.0-bin-hadoop2.4]# ./bin/spark-submit --master yarn-cluster -
-class com.ht.sparks.WordCount  --num-executors 3 --executor-cores 1  /spark/Sp
arkWC-0.0.1-SNAPSHOT.jar /tmp/in /tmp/out7
```

Issue:

```
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:08 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 2 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:09 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 3 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:10 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 4 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:11 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 5 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:12 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 6 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:13 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 7 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:14 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 8 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:15 INFO Client: Retrying connect to server: 0.0.0.0/0.0.0.0:8032. Already tried 9 time(s); retry policy is Retry
UpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
21/01/23 05:38:15 INFO RetryInvocationHandler: java.net.ConnectException: Your endpoint configuration is wrong; For more detail
s see:  http://wiki.apache.org/hadoop/UnsetHostnameOrPort, while invoking ApplicationClientProtocolPBClientImpl.getClusterMetri
cs over null after 1 failover attempts. Trying to failover after sleeping for 42760ms.
```

Verify the ip of the resource manager and set the home directory appropriately.
Try updating the following setting in yarn-site.xml of Hadoop and spark node too..

```
<property>
        <name>yarn.resourcemanager.hostname</name>
        <value>hadoop0</value>
        <description>The hostname of the RM.</description>
    </property>
<property>
        <name>yarn.resourcemanager.address</name>
        <value>hadoop0:8032</value>
        <description>The hostname of the RM.</description>
    </property>
```

---

23. Jobs Monitoring : Using Web UI. – 45 Minutes

Execute the Job as specified in the Spark standalone cluster lab/ Using Docker before proceeding ahead.

- Start the Two Nodes Spark Cluster
- Copy the input file in both the nodes, if you are executing on the Standalone cluster.

Execute the following job:

File names.json contains:

```
{"firstName":"Grace","lastName":"Hopper"}
{"firstName":"Alan","lastName":"Turing"}
{"firstName":"Ada","lastName":"Lovelace"}
{"firstName":"Charles","lastName":"Babbage"}
```

Enter the following commands one at a time.

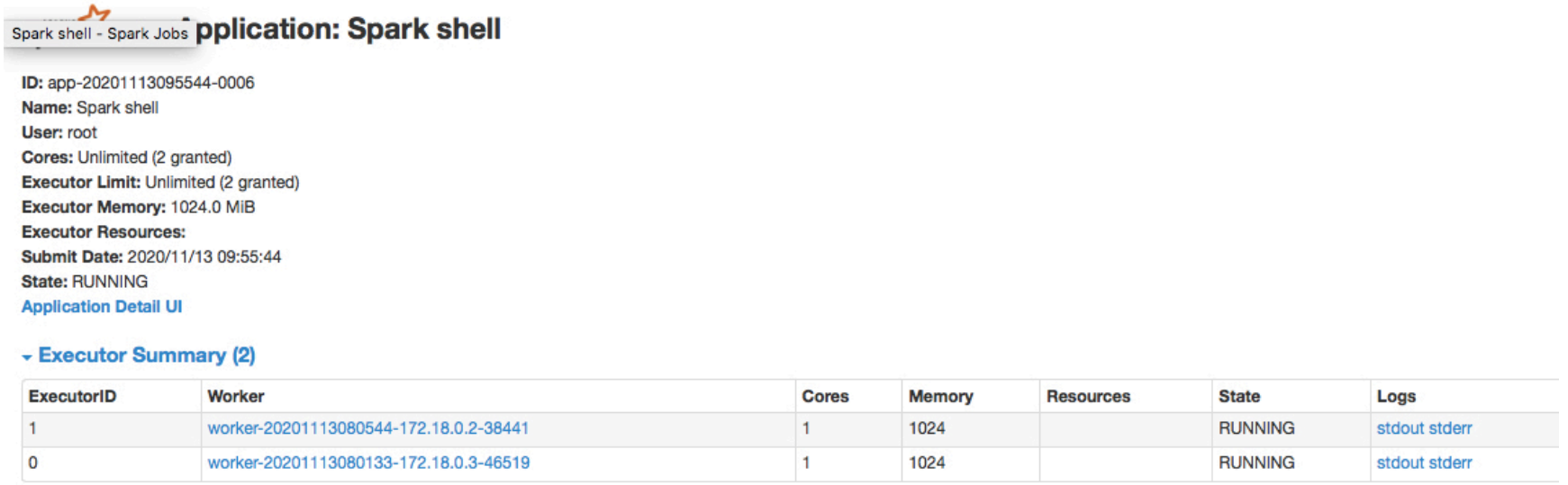
```
#pyspark --master spark://spark0:8090
ip="/software/names.json"
op="/software/nameop"
spark.sparkContext.setLogLevel("WARN")
peopleDF = spark.read.json(ip)     // One Job
namesDF = peopleDF.select("firstName","lastName")
namesDF.write.option("header","true").csv(op)    // Second Job
```

Access the Spark Cluster web console.

http://127.0.0.1:8080

You should have two worker nodes as shown below.

Click on Application ID → Application Detail UI

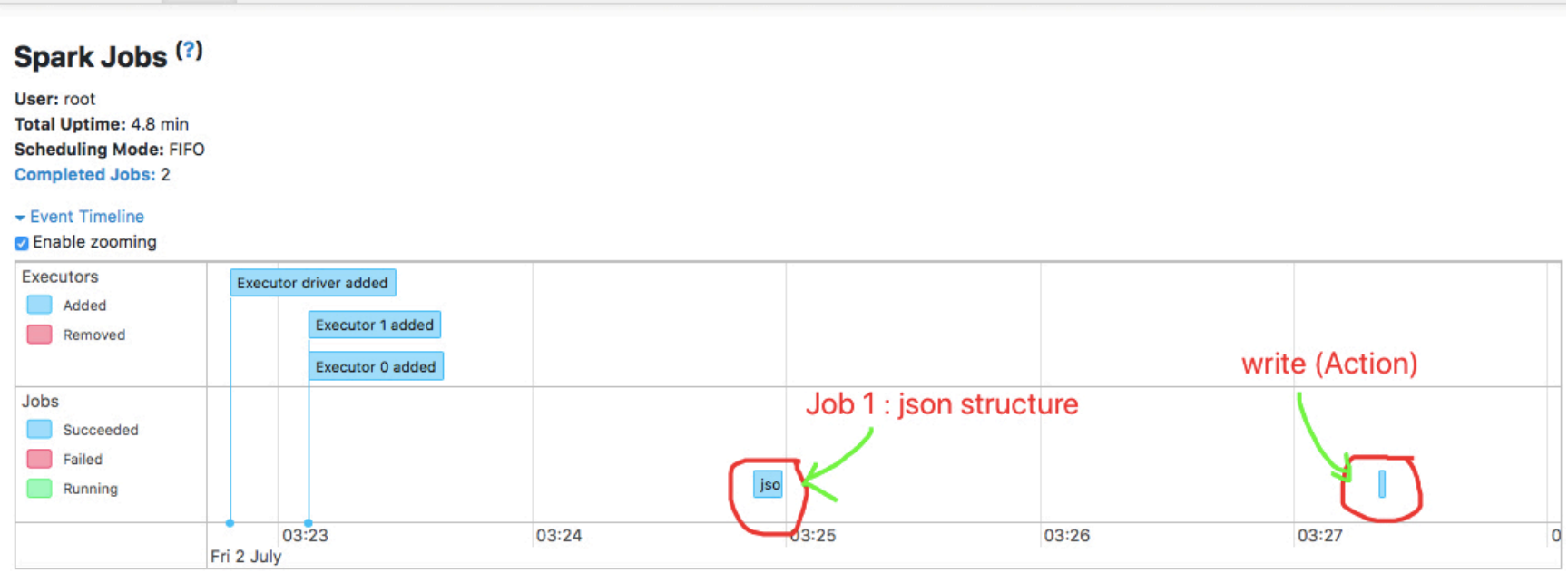


You can verify the Jobs details Using Spark UI: http://master:4040/jobs/

web UI comes with the following tabs (which may not all be visible at once as they are lazily created on demand, e.g. Streaming tab):

- Jobs
- Stages
- Storage with RDD size and memory use
- Environment
- Executors
- SQL

The **Jobs Tab** shows status of all Spark jobs in a Spark application

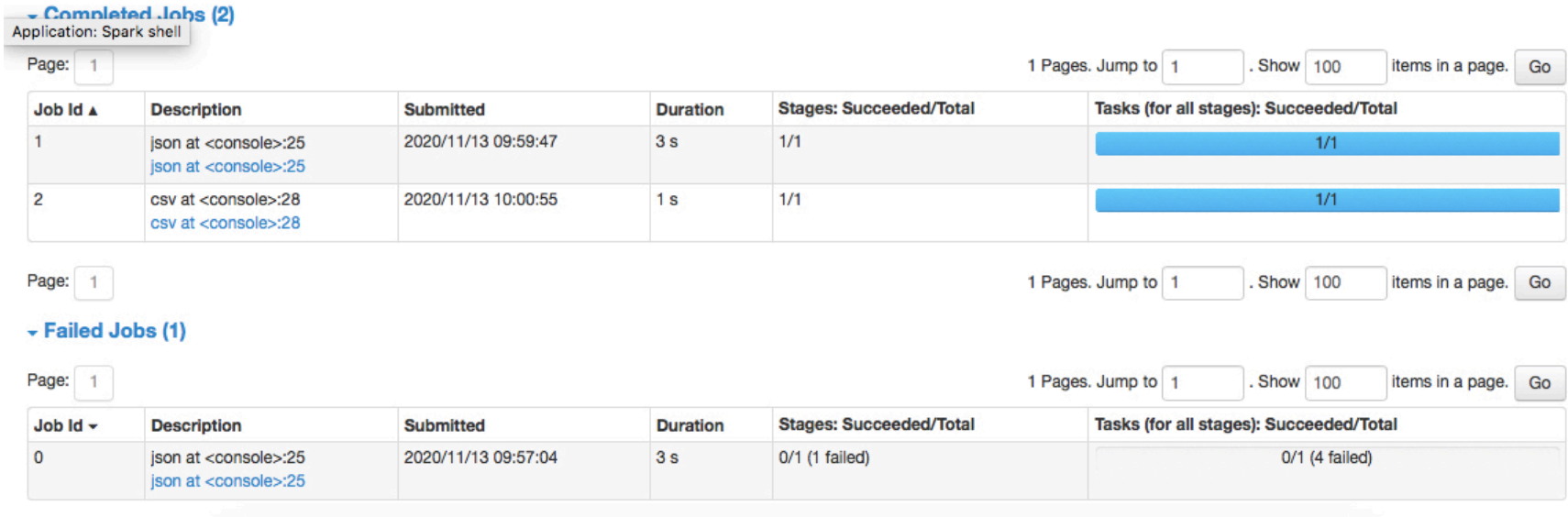


Display the timeline of Executor being added in the Job. When the Job Get completed etc.
In the above, you can verify that 2 executors have been added.

1 Job Succeded i.e (Reading Json file)
2 Job Succeeded i.e write action.

When you hover over a job in Event Timeline not only you see the job legend but also the job is highlighted in the Summary section.

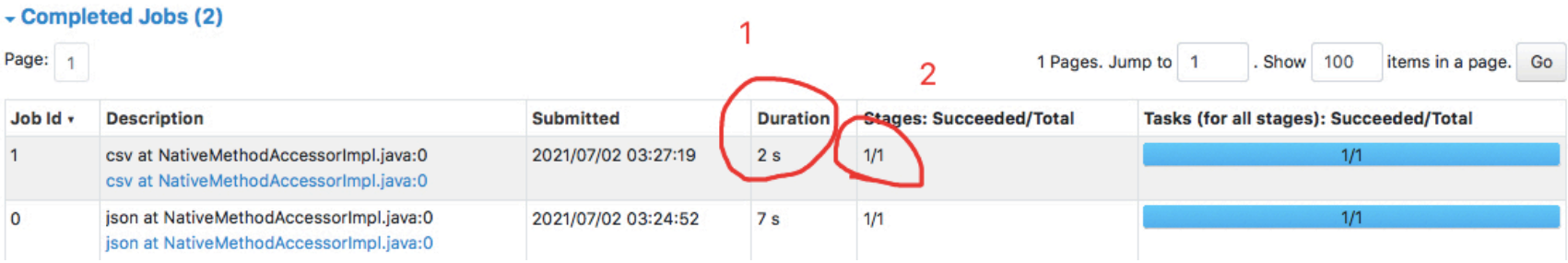
The Event Timeline section shows not only jobs but also executors.
You can verify the completed and failed jobs too:


Completed Jobs (2)
Application: Spark shell
Page: 1
1 Pages. Jump to 1 . Show 100 items in a page. Go
Job Id ▲ | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total
1 | json at <console>:25 json at <console>:25 | 2020/11/13 09:59:47 | 3 s | 1/1 | 1/1
2 | csv at <console>:28 csv at <console>:28 | 2020/11/13 10:00:55 | 1 s | 1/1 | 1/1
Page: 1
1 Pages. Jump to 1 . Show 100 items in a page. Go
Failed Jobs (1)
Page: 1
1 Pages. Jump to 1 . Show 100 items in a page. Go
Job Id ▾ | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total
0 | json at <console>:25 json at <console>:25 | 2020/11/13 09:57:04 | 3 s | 0/1 (1 failed) | 0/1 (4 failed)

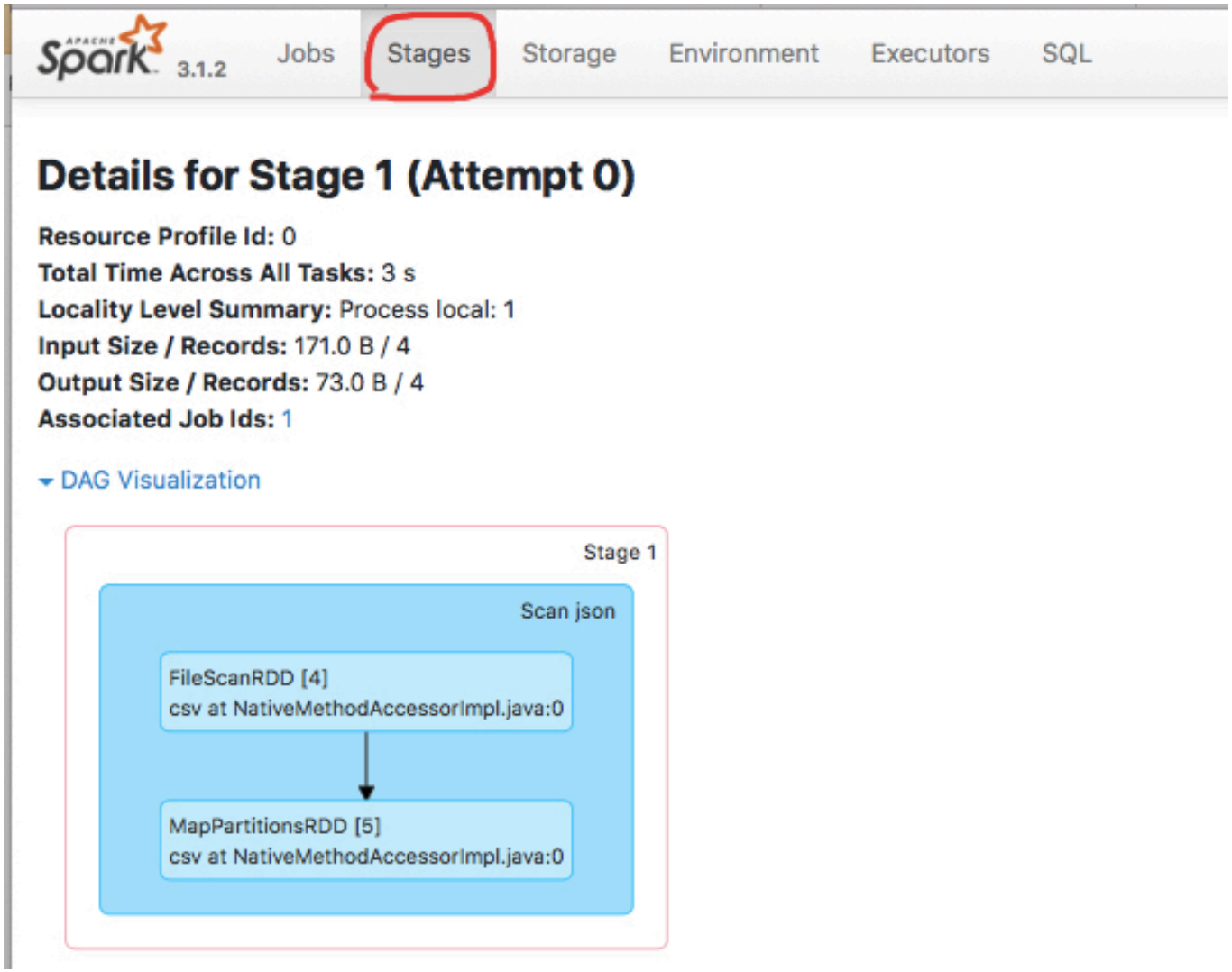
You can verify some of the following Queries from this console?

How long the Job takes to execute? – (1: Here it is 2 s.)

How Many Statges and Tasks created per Job? (2: 1 stages and only one task)

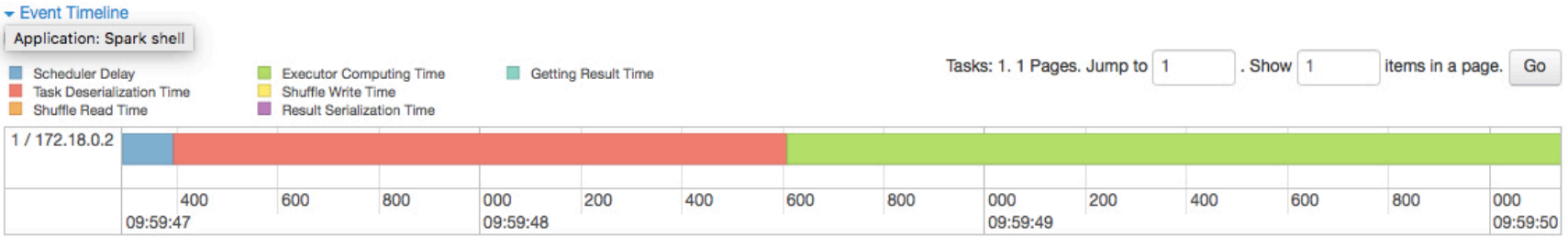


Click on On job (1 – write action or job) -> for details on Description → stages to view DAG - When you click a job in All Jobs Page page, you see the **Details for Job** page.



As seen above, there is 1 stage in the Job. It scans the file and create a map partition RDD.

It shows the DAG graph and the stage details.
Scroll down to view the Event timeline.



Summary Metrics for 1 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	2 s	2 s	2 s	2 s	2 s
GC Time	68.0 ms	68.0 ms	68.0 ms	68.0 ms	68.0 ms
Input Size / Records	171 B / 5	171 B / 5	171 B / 5	171 B / 5	171 B / 5

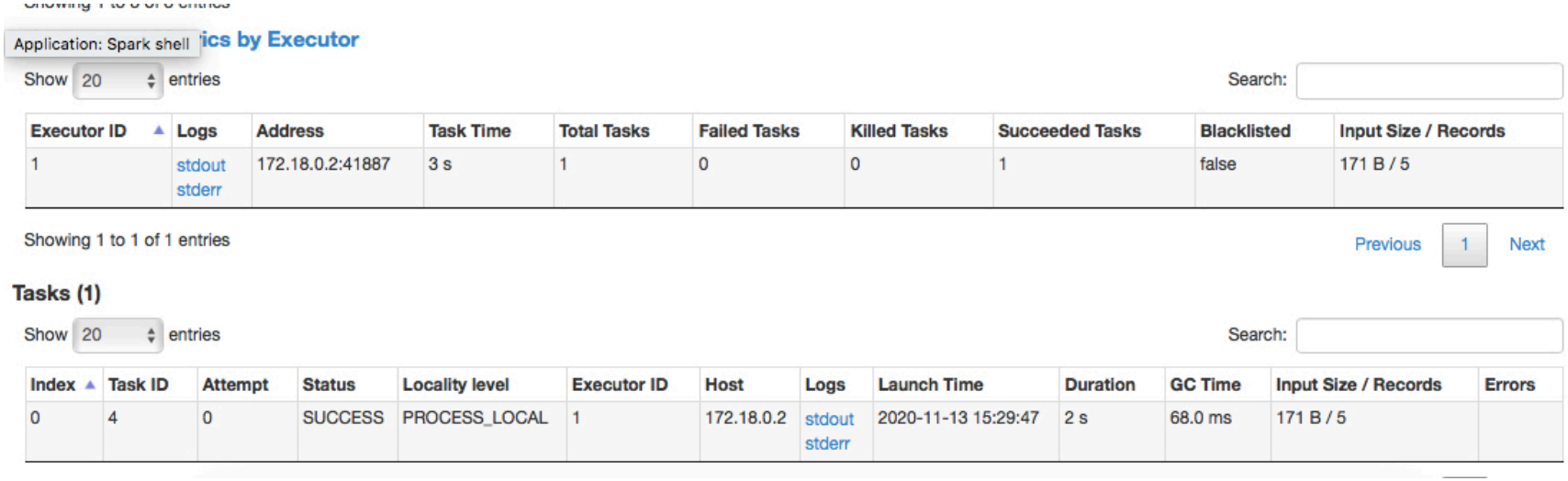
Showing 1 to 3 of 3 entries

Questions:

How long does the scheduler take to schedule the task? (Check the Light Blue Block)
How long does the task deserialization take? (Check the Orange Block)
What is the actual computation time? (Verify the Green Color block)

In ideal scenario, Green should take most of the computing resources.
You can also verify how much each task took to excute. It will provide some information about the health of the computing task.

Scroll down to view the task metrics: which task take majority of the time etc. How much byte consume or output by each task and its statistics.



You can verify from the above, Locality_Level column – the data is process locally. i.e data locality.

Executors Tab

Stats of each executor, how much time it spent on GC etc. How much data get shuffle?
Executors tab in web UI shows

Summary

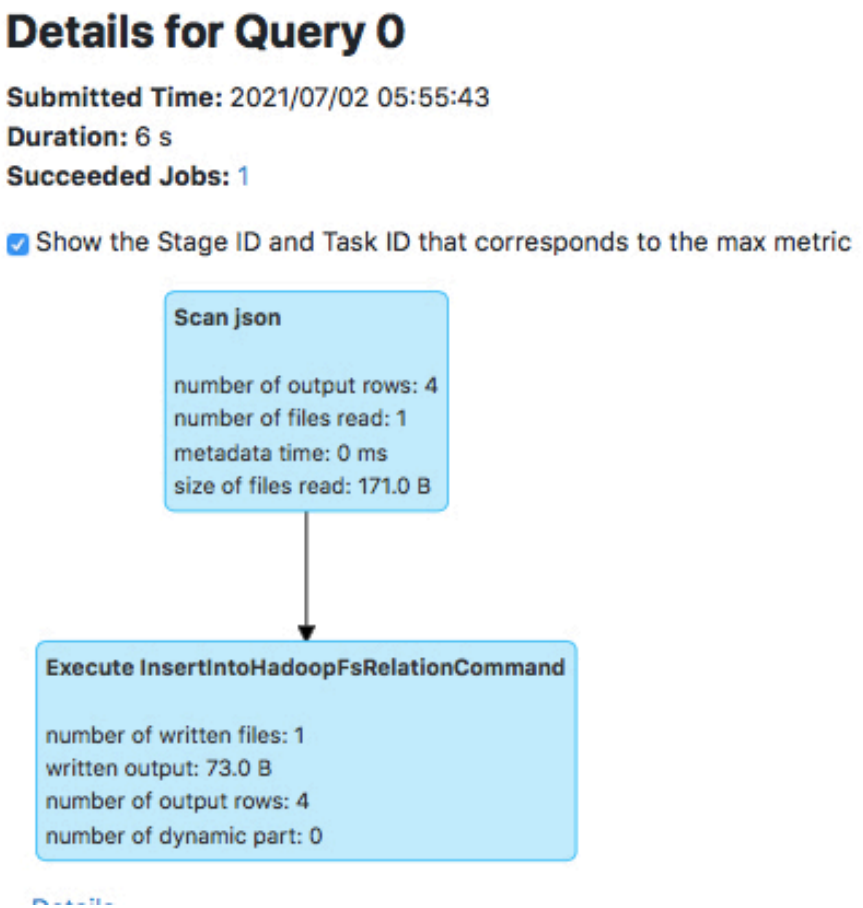
	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(3)	1	23.4 KB / 1.3 GB	0.0 B	2	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Dead(4)	2	46.8 KB / 1.7 GB	0.0 B	4	0	0	7	7	1.0 min (6 s)	21 KB	0.0 B	0.0 B
Total(7)	3	70.2 KB / 3 GB	0.0 B	6	0	0	7	7	1.0 min (6 s)	21 KB	0.0 B	0.0 B

Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
driver	192.168.188.173:54112	Active	1	23.4 KB / 434 MB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B		Thread Dump
0	192.168.188.173:54931	Dead	1	23.4 KB / 434 MB	0.0 B	1	0	0	4	4	37 s (6 s)	9.5 KB	0.0 B	0.0 B	stdout stderr	Thread Dump
1	192.168.188.174:34320	Dead	1	23.4 KB / 434 MB	0.0 B	1	0	0	3	3	25 s (0.3 s)	11.5 KB	0.0 B	0.0 B	stdout stderr	Thread Dump
2	192.168.188.174:46874	Dead	0	0.0 B / 434 MB	0.0 B	1	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
3	192.168.188.173:34382	Dead	0	0.0 B / 434 MB	0.0 B	1	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
4	192.168.188.174:44548	Active	0	0.0 B / 434 MB	0.0 B	1	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump

Click on SQl tab



You can view the statistics of the SQL – How much Rows read and how much bytes etc.

Verify the plan (Scroll down the SQL tab.)

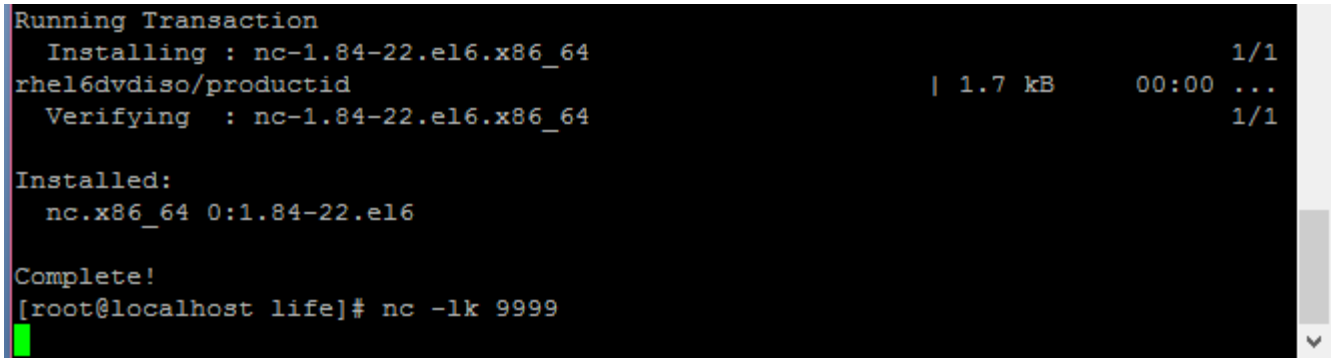
▾ Details

```
== Physical Plan ==
Execute InsertIntoHadoopFsRelationCommand (2)
+- Scan json  (1)


(1) Scan json
Output [2]: [firstName#7, lastName#8]
Batched: false
Location: InMemoryFileIndex [file:/opt/data/names.json]
ReadSchema: struct<firstName:string,lastName:string>

(2) Execute InsertIntoHadoopFsRelationCommand
Input [2]: [firstName#7, lastName#8]
Arguments: file:/opt/data/op, false, CSV, [header=true, path=op], ErrorIfExists, [firstName, lastName]
```

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

24.Spark Streaming With Socket – 45 Minutes

You can execute using pyspark cli console or zeppelin
Open a terminal and perform the following activities.
Install netcat utility
yum install nc.x86_64
or yum install nc

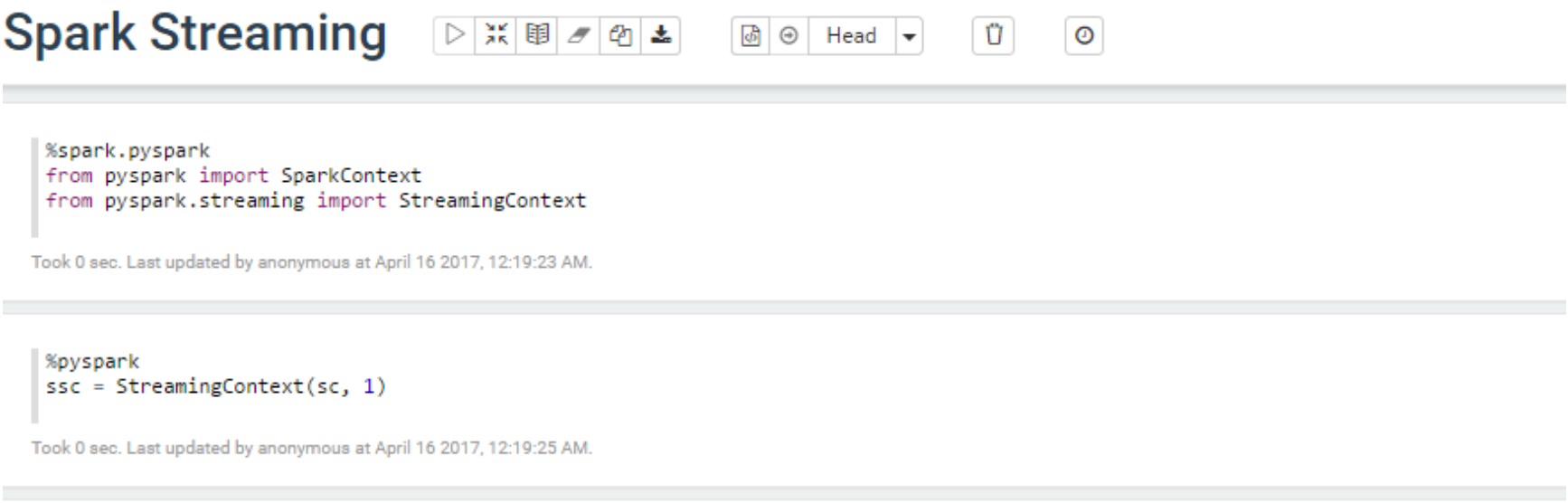


You will first need to run Netcat (a small utility found in most Unix-like systems) as a data server by using
$ nc -lk 9999
Then, any lines typed in the terminal running the netcat server will be counted and printed on screen every second. It will look something like the following.

Type the following command in pyspark or Zeppelin.

First, we import StreamingContext, which is the main entry point for all streaming functionality. We create a local StreamingContext with two execution threads, and batch interval of 1 second.

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
# Create a local StreamingContext with two working thread and batch interval of 5 second
ssc = StreamingContext(sc, 5)
```




```
// Create a DStream that will connect to hostname:port, like localhost:9999
lines = ssc.socketTextStream("localhost", 9999)
```

This lines DStream represents the stream of data that will be received from the data server. Each record in this DStream is a line of text. Next, we want to split the lines by space into words.

```
# Split each line into words
words = lines.flatMap(lambda line: line.split(" "))
```

flatMap is a one-to-many DStream operation that creates a new DStream by generating multiple new records from each record in the source DStream. In this case, each line will be split into multiple words and the stream of words is represented as the words DStream. Next, we want to count these words.

Count each word in each batch
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda a, b: a + b)

Print the first ten elements of each RDD generated in this DStream to the console
wordCounts.pprint()

The words DStream is further mapped (one-to-one transformation) to a DStream of (word, 1) pairs, which is then reduced to get the frequency of words in each batch of data.
Finally, wordCounts.pprint() will print a few of the counts generated every second.

Note that when these lines are executed, Spark Streaming only sets up the computation it will perform when it is started, and no real processing has started yet. To start the processing after all the transformations have been setup, we finally call

ssc.start() # Start the computation
ssc.awaitTermination() # Wait for the computation to terminate

```
%pyspark
counts = lines.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b)
counts.pprint()
ssc.start()
ssc.awaitTermination()
```

```
-------------------------------------------
```

You will be able to see the following in the zeppelin console

```
-------------------------------------------
Time: 2017-04-15 11:51:16
-------------------------------------------
-------------------------------------------
Time: 2017-04-15 11:51:17
-------------------------------------------
-------------------------------------------
Time: 2017-04-15 11:51:18
-------------------------------------------
(u'we', 1)
(u'where', 1)
(u'going', 1)
(u'now?', 1)
(u'are', 1)
-------------------------------------------
Time: 2017-04-15 11:51:19
-------------------------------------------
```

Counting each words occurrence in 1 second.

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

25.Spark Streaming With Kafka – 60 Minutes

Goals: To integrate Apache Kafka with Spark Using Streaming API.

Steps to be performed:

- Install Kafka and start it along with producer.
- Write python program to connect to Kafka and get data from Kafka using streaming
- Execute it to fetch the data.

Use the Spark VM

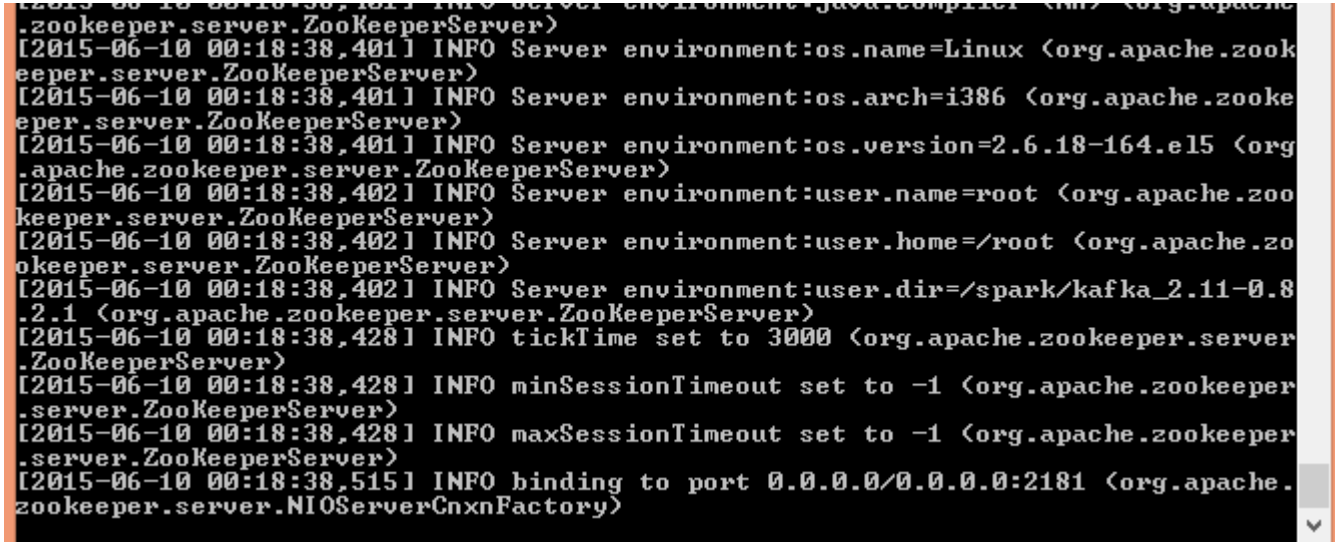
Install kafka:

tar -xvf kafka_2.12-2.8.0.tar -C /spark
cd /spark/kafka_2.11-0.8.2.1

```
[root@ht kafka_2.11-0.8.2.1]# pwd
/spark/kafka_2.11-0.8.2.1
[root@ht kafka_2.11-0.8.2.1]# ls
bin  config  libs  LICENSE  NOTICE
[root@ht kafka_2.11-0.8.2.1]#
```

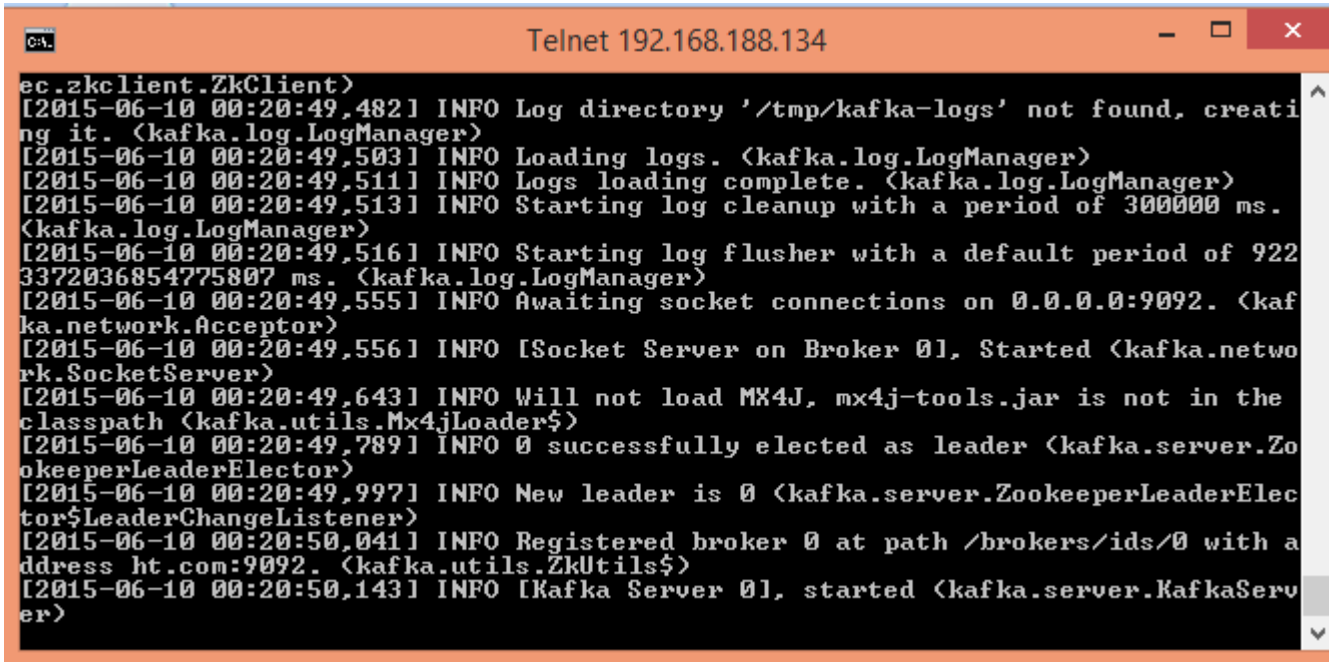
to Start the server, start zookeeper first using a terminal

bin/zookeeper-server-start.sh config/zookeeper.properties



Now start the Kafka server on other terminal

bin/kafka-server-start.sh config/server.properties

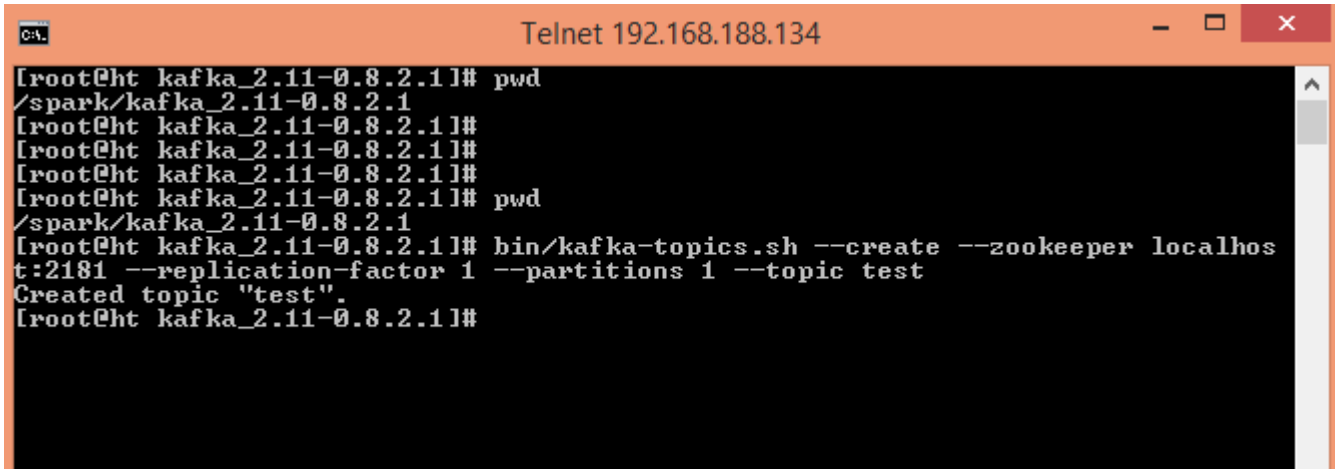


Keep both the console open.

Create a topic

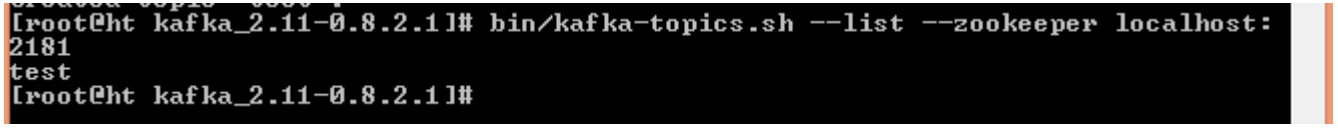
Let's create a topic named "test" with a single partition and only one replica: Open one more terminal

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test



We can now see that topic if we run the list topic command:

> bin/kafka-topics.sh --list --zookeeper localhost:2181



Send some messages

// Open a new console, change directory to the kafka installation folder

```
test
[root@ht kafka_2.11-0.8.2.1]# bin/kafka-console-producer.sh --broker-list local
host:9092 --topic test
[2015-06-10 00:25:31,582] WARN Property topic is not valid (kafka.utils.Verifiab
leProperties)
I am sending a message
```

bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test

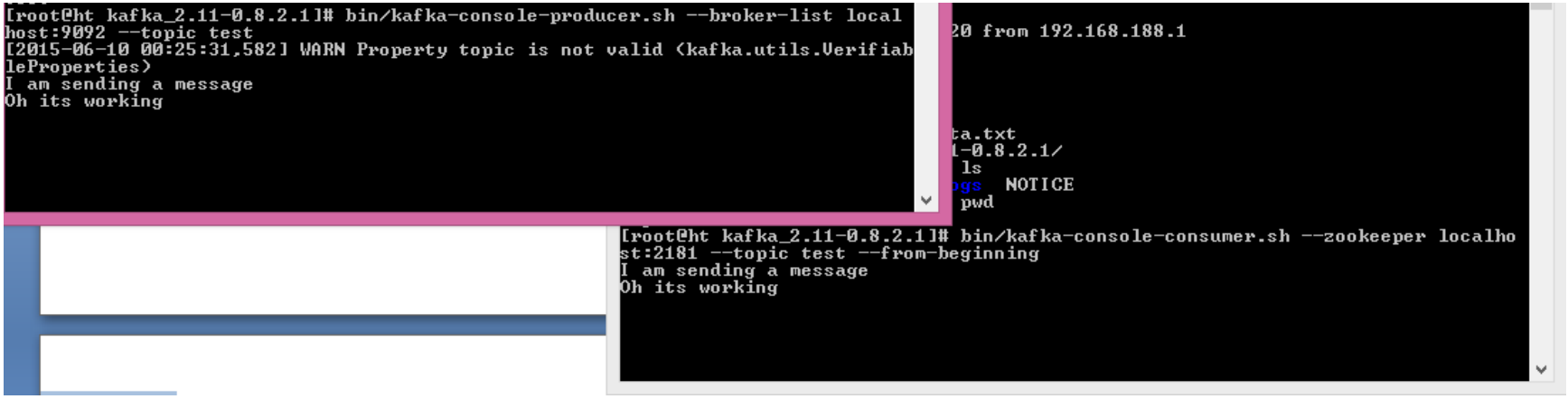
// type some message

Start a consumer

// Open a new console, change directory to the kafka installation folder
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

```
[root@spark0 kafka]# bin/kafka-console-consumer.sh --bootstrap-server localhost:9092  --topic tes
t --from-beginning
Hi How are you?
Hi How are you?
```

Try sending some information from the producer to consumer.



You have successfully configure Kafka, Now let us fetch information using spark.

Do not close any of the above window.

Update the following code in kafka.py file.

---------- Code Begin ----------
```
from __future__ import print_function

import sys

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

# Initializing Spark session
spark = SparkSession.builder.appName("MySparkSession").getOrCreate()
spark.sparkContext.setLogLevel('ERROR')
# Setting parameters for the Spark session to read from Kafka
bootstrapServers = "spark0:9092"
subscribeType ="subscribe"
topics = "test"

# Streaming data from Kafka topic as a dataframe
lines = spark.readStream.format("kafka").option("kafka.bootstrap.servers",
bootstrapServers).option(subscribeType,
 topics).load()

# Expression that reads in raw data from dataframe as a string
```

```
and names the column "words"
```

```
lines = lines.selectExpr("CAST(value AS STRING) as words")
```

```
query = lines.writeStream.outputMode("append").format("console").start()
# Terminates the stream on abort
query.awaitTermination()
```

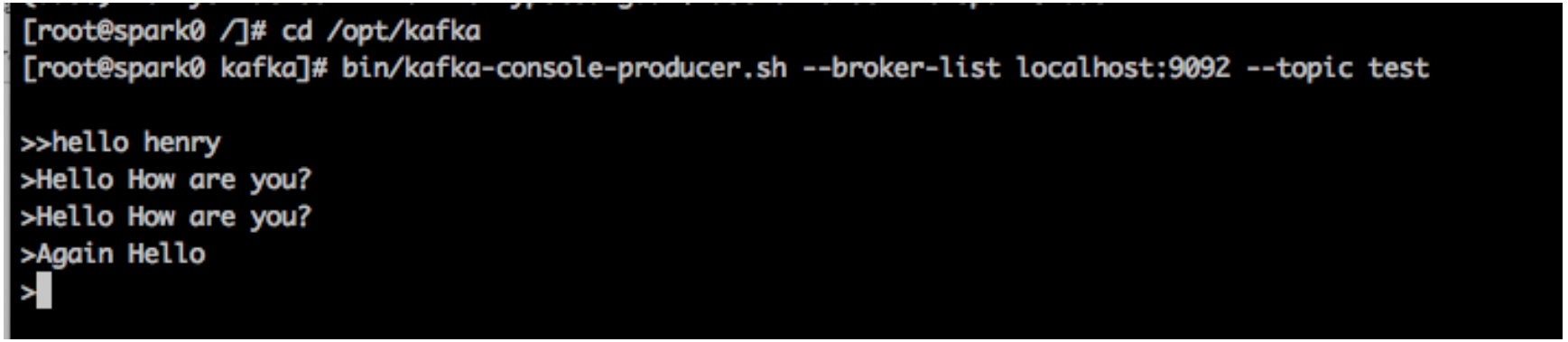
---------- Code Ends ----------

Submit the job with the following parameters:

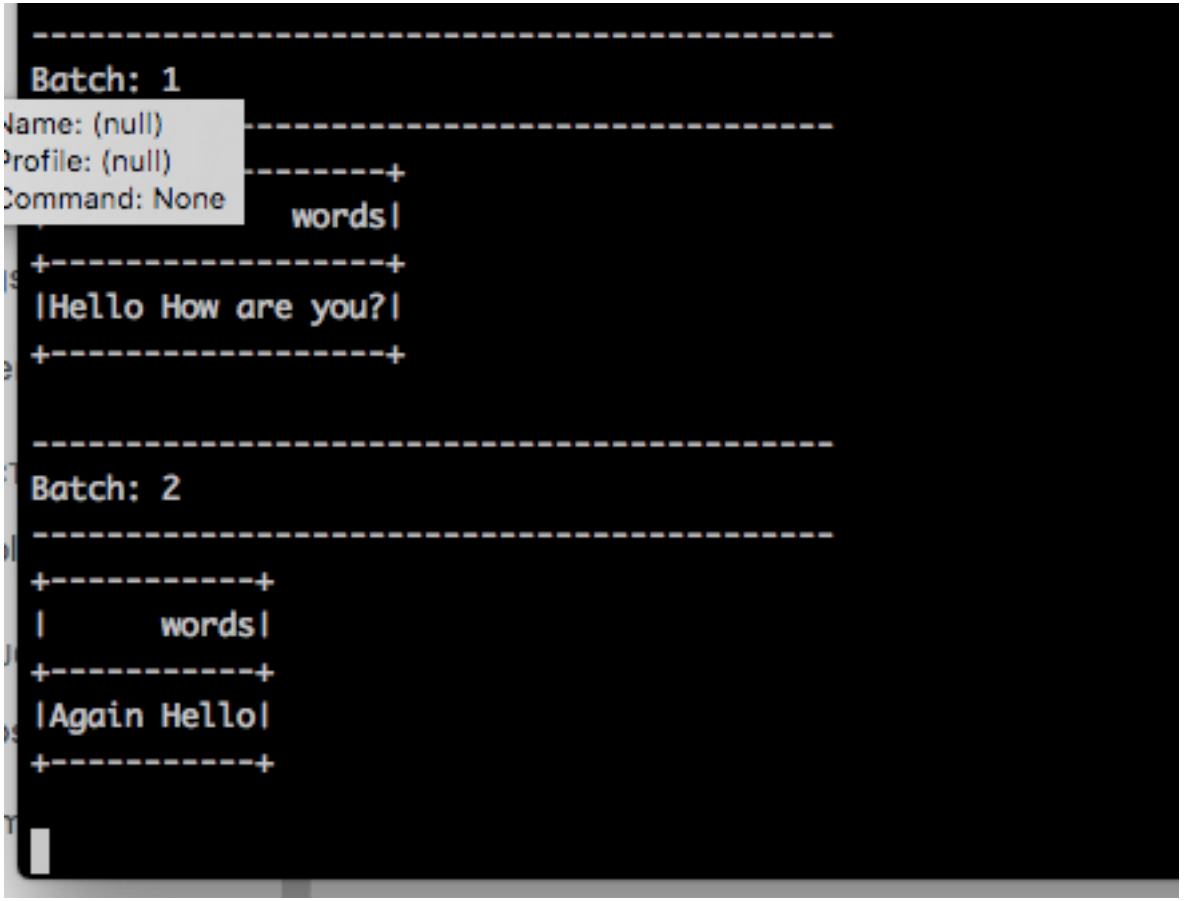
#spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 --master local[2] --conf "spark.dynamicAllocation.enabled=false" kafka.py

```
[root@spark0 data]# spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 --master local[2] --c
                  amicAllocation.enabled=false" kafka_wordcount.py
Name: (null)      tings :: url = jar:file:/opt/spark/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
Profile: (null)   che set to: /root/.ivy2/cache
Command: None
The jars for the packages stored in: /root/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-5e26bf52-2ac0-43da-90df-3793adc16dbb;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.1.2 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.1.2 in central
        found org.apache.kafka#kafka-clients;2.6.0 in central
        found com.github.luben#zstd-jni;1.4.8-1 in central
        found org.lz4#lz4-java;1.7.1 in central
        found org.xerial.snappy#snappy-java;1.1.8.2 in central
        found org.slf4j#slf4j-api;1.7.30 in central
        found org.spark-project.spark#unused;1.0.0 in central
        found org.apache.commons#commons-pool2;2.6.2 in central
downloading https://repo1.maven.org/maven2/org/apache/spark/spark-sql-kafka-0-10_2.12/3.1.2/spark-sql-kafka-0-10_2
.12-3.1.2.jar ...
        [SUCCESSFUL ] org.apache.spark#spark-sql-kafka-0-10_2.12;3.1.2!spark-sql-kafka-0-10_2.12.jar (712ms)
downloading https://repo1.maven.org/maven2/org/apache/spark/spark-token-provider-kafka-0-10_2.12/3.1.2/spark-token
```

Type some words on the Producer console.



Whatever you push to the Kafka should be written on the console as shown below.



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

26. Streaming with State Operations- Python – 60 Minutes

In this lab, we will aggregate the word across a given window time interval.

To run this on your local machine, you need to first run a Netcat server.
$ nc -lk 9999

We will be sending message to the above netcat server and then apply transformation to it using Spark Streaming.

Open a pyspark terminal.

#pyspark

Type the following code one by one in the console:

```
from __future__ import print_function
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
```

The above commands import the necessary required packages for execution of Spark Streaming application.

#Initialize the Streamming context using the pre created spark context, sc. The interval is 5 5 seconds. i.e micro batch of 5 mseconds.

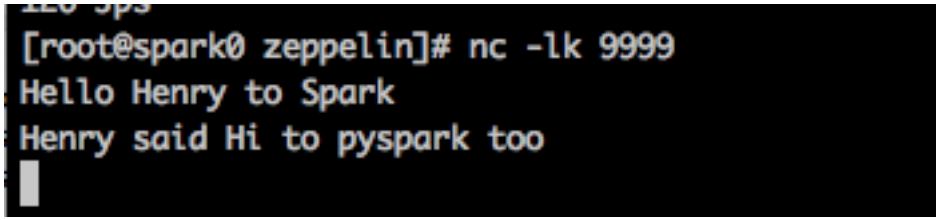
```
ssc = StreamingContext(sc, 5)

# Specify the checkpoint directory, since we are going to maintain state across the batch.
ssc.checkpoint("checkpoint")

# RDD with initial state (key, value) pairs
initialStateRDD = sc.parallelize([(u'hello', 1), (u'world', 1)])
```

```
#Define the function that will update the state of the count.
def updateFunc(new_values, last_sum): \
    return sum(new_values) + (last_sum or 0)

# Open a connection to the netcat server to retrieve the incoming messages.
lines = ssc.socketTextStream("localhost",9999)

# Apply the transformation logic to count the word. Here, we are using flatMap, since it can return list of objects.

running_counts = lines.flatMap(lambda line: line.split(" "))\
                          .map(lambda word: (word, 1))\
                          .updateStateByKey(updateFunc, initialRDD=initialStateRDD)

# Print out the calculation on the terminal.

running_counts.pprint()

# start the streaming context and wait for termination from external.
ssc.start()
ssc.awaitTermination()
```

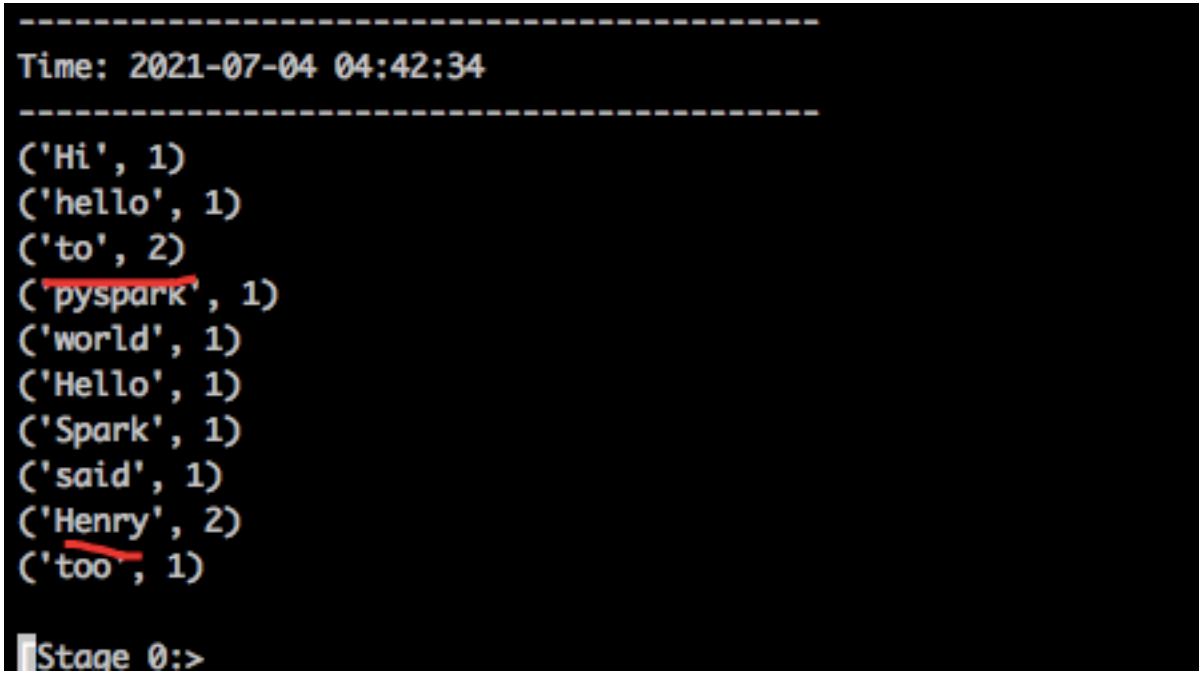
Snippets:

```
>>> ssc = StreamingContext(sc, 1)
Name: (null)
Profile: (null)
Command: None
kpoint("checkpoint")
tateRDD = sc.parallelize([(u'hello', 1), (u'world', 1)])
teFunc(new_values, last_sum): \
... return sum(new_values) + (last_sum or 0)
...
>>> lines = ssc.socketTextStream('localhost',9999))
```

On the telnet terminal, Type sentences like following with some interval between the sentences.



Observe the out put on the pyspark console.



As you can observe that “Henry” word has occurred twice across the data records. Thus state has been maintained.

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

27. Spark – Hive Integration – 45 Minutes

When working with Hive, one must instantiate SparkSession with Hive support, including connectivity to a persistent Hive metastore, support for Hive serdes, and Hive user-defined functions. Users who do not have an existing Hive deployment can still enable Hive support.

use spark.sql.warehouse.dir to specify the default location of database in warehouse

Open a pyspark terminal (Using CLI only)

#pyspark --conf spark.sql.catalogImplementation=hive

Execute the following code.

```
from os.path import abspath

from pyspark.sql import SparkSession
from pyspark.sql import Row

# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('/opt/spark/spark-warehouse')

spark = SparkSession \
    .builder \
    .appName("Python Spark SQL Hive integration example") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("spark.sql.catalogImplementation", "hive") \
    .enableHiveSupport() \
    .getOrCreate()

spark.sql("CREATE TABLE IF NOT EXISTS src (key INT, value STRING) USING hive")
```

spark.sql("LOAD DATA LOCAL INPATH '/opt/spark/examples/src/main/resources/kv1.txt' INTO TABLE src")

```
# Queries are expressed in HiveQL
spark.sql("SELECT * FROM src").show()
# +---+-------+
# |key|  value|
# +---+-------+
# |238|val_238|
# | 86| val_86|
# |311|val_311|
# ...

# Aggregation queries are also supported.
spark.sql("SELECT COUNT(*) FROM src").show()
# +--------+
# |count(1)|
# +--------+
# |    500 |
# +--------+

# The results of SQL queries are themselves DataFrames and support all normal functions.
sqlDF = spark.sql("SELECT key, value FROM src WHERE key < 10 ORDER BY key")

# The items in DataFrames are of type Row, which allows you to access each column by
ordinal.
stringsDS = sqlDF.rdd.map(lambda row: "Key: %d, Value: %s" % (row.key, row.value))
for record in stringsDS.collect():
    print(record)
# Key: 0, Value: val_0
# Key: 0, Value: val_0
```

```
# Key: 0, Value: val_0
# ...

# You can also use DataFrames to create temporary views within a SparkSession.
Record = Row("key", "value")
recordsDF = spark.createDataFrame([Record(i, "val_" + str(i)) for i in range(1, 101)])
recordsDF.createOrReplaceTempView("records")

# Queries can then join DataFrame data with data stored in Hive.
spark.sql("SELECT * FROM records r JOIN src s ON r.key = s.key").show()
```

Execution Output:

```
[root@spark0 data]# pyspark --conf spark.sql.catalogImplementation=hive
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
21/07/03 10:09:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using built
in-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/07/03 10:10:04 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Python version 3.6.8 (default, Nov 16 2020 16:55:22)
Spark context Web UI available at http://spark0:4041
Spark context available as 'sc' (master = local[*], app id = local-1625307004585).
SparkSession available as 'spark'.
>>> from os.path import abspath
>>>
>>> from pyspark.sql import SparkSession
>>> from pyspark.sql import Row
```

Profile: (null)
Command: None

```
>>> spark.sql("LOAD DATA LOCAL INPATH '/opt/spark/examples/src/main/resources/kv1.txt' INTO TABLE src")
21/07/03 10:11:29 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
DataFrame[]
>>> spark.sql("SELECT * FROM src").show()
+---+-------+
|key|  value|
+---+-------+
|238|val_238|
| 86| val_86|
|311|val_311|
| 27| val_27|
|165|val_165|
|409|val_409|
|255|val_255|
|278|val_278|
| 98| val_98|
```

```
>>> spark.sql("SELECT COUNT(*) FROM src").show()
+--------+
|count(1)|
+--------+
|     500|
+--------+

>>> sqlDF = spark.sql("SELECT key, value FROM src WHERE key < 10 ORDER BY key")
>>> stringsDS = sqlDF.rdd.map(lambda row: "Key: %d, Value: %s" % (row.key, row.value))
>>> for record in stringsDS.collect():
...     print(record)
...
Key: 0, Value: val_0
Key: 0, Value: val_0
Key: 0, Value: val_0
Key: 2, Value: val_2
Key: 4, Value: val_4
Key: 5, Value: val_5
Key: 5, Value: val_5
Key: 5, Value: val_5
```

```
>>> Record = Row("key", "value")
>>> recordsDF = spark.createDataFrame([Record(i, "val_" + str(i)) for i in range(1, 101)])
>>> recordsDF.createOrReplaceTempView("records")
>>> spark.sql("SELECT * FROM records r JOIN src s ON r.key = s.key").show()
+---+------+---+------+
|key| value|key| value|
+---+------+---+------+
|  2| val_2|  2| val_2|
|  4| val_4|  4| val_4|
|  5| val_5|  5| val_5|
|  5| val_5|  5| val_5|
|  5| val_5|  5| val_5|
|  8| val_8|  8| val_8|
|  9| val_9|  9| val_9|
| 10|val_10| 10|val_10|
| 11|val_11| 11|val_11|
| 12|val_12| 12|val_12|
| 12|val_12| 12|val_12|
| 15|val_15| 15|val_15|
| 15|val_15| 15|val_15|
```

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

28. Spark integration with Hive Cluster (Local Mode)

Pre requisites: Spark Installation and Hive Local Mode installation.

You can refer Hive Lab for Hive Local Installation.

In this lab, we will configure to use Apache Spark with Hive.

export SPARK_HOME=/opt/spark
export HIVE_HOME=/opt/hive_local

Set HIVE_HOME and SPARK_HOME accordingly.

29.Annexure & Errata:

Quit Scala CLI :- :q
Error: No suitable device found: no device found for connection 'System eth0'.
You could try removing/renaming /etc/udev/rules.d/70-persistent-net.rules and reboot. A new file will be created, sometimes that fixes this kind of problems.

You could also add a file "/etc/sysconfig/network-scripts/ifcfg-eth1", can you bring up eth1 then?

Changing hostname:

- /etc/hosts
- vi /etc/sysconfig/network
- sysctl kernel.hostname=slave
- bash

Unable to start spark shell with the following error

```
  at org.apache.hadoop.hive.ql.session.SessionState.createSessionDirs(SessionState.java:554)
  at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:508)
  ... 85 more
Caused by: java.net.ConnectException: Connection refused
  at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
  at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:717)
  at org.apache.hadoop.net.SocketIOWithTimeout.connect(SocketIOWithTimeout.java:206)
  at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:531)
  at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:495)
  at org.apache.hadoop.ipc.Client$Connection.setupConnection(Client.java:614)
```

```
  at org.apache.hadoop.ipc.Client$Connection.setupIOstreams(Client.java:712)
  at org.apache.hadoop.ipc.Client$Connection.access$2900(Client.java:375)
  at org.apache.hadoop.ipc.Client.getConnection(Client.java:1528)
  at org.apache.hadoop.ipc.Client.call(Client.java:1451)
```

Solution: unset HADOOP_CONF_DIR and unset HADOOP_HOME

https://jaceklaskowski.gitbooks.io/mastering-apache-spark/

issue: Cannot assign requested address

```
17/03/15 21:44:15 ERROR spark.SparkContext: Error initializing SparkContext.
java.net.BindException: Cannot assign requested address: Service 'sparkDriver' f
ailed after 16 retries (starting from 0)! Consider explicitly setting the approp
riate port for the service 'sparkDriver' (for example spark.ui.port for SparkUI)
to an available port or increasing spark.port.maxRetries.
        at sun.nio.ch.Net.bind0(Native Method)
        at sun.nio.ch.Net.bind(Net.java:433)
        at sun.nio.ch.Net.bind(Net.java:425)
        at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:
223)
```

Solution:

- include host to ip entry in /etc/hosts --> 192.168.188.178 master
- update in conf/spark-env.sh

    - SPARK_LOCAL_IP=127.0.0.1
    - SPARK_MASTER_HOST=192.168.188.178

- comment the following in conf/spark-defaults.conf
    - # spark.master spark://master:7077

Unable to start spark shell:

Caused by: java.io.IOException: Error accessing /opt/jdk/jre/lib/ext/._cld*.jar
Such kind of error resolve by removing all the hidden file. .* (You can determine the file by running #ls -alt). List all the hidden dot file and remove it.

Yum repo config

mkdir -p /redhatimg
mount -o loop,ro /mnt/hgfs/MyExperiment/rhel-server-6.4-x86_64-dvd.iso /redhatimg

Create an entry in /etc/fstab so that the system always mounts the DVD image after a reboot.
/mnt/hgfs/MyExperiment/rhel-server-6.4-x86_64-dvd.iso /redhatimg iso9660 loop,ro 0 0

/etc/yum.repos.d/rheldiso.repo

[rhel6dvdiso]
name=RedHatOS DVD ISO
mediaid=1359576196.686790
baseurl=file:///redhatimg
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release