Spark Basics Walkthrough (01-spark-basics-lab.{scala,md}

This file gets you started in Spark in Scala.

Turn down the logging level

By default Spark versions before 2.0, many of Spark's logging parameters are set to `INFO` leading to extremely verbose messages printed to the console. In fact, if you don't turn these down, the prompt will soon scroll off the screen!

On Amazon EMR

If you are running a version of Spark before 2.0, edit `/usr/lib/spark/conf/log4j.properties` on the cluster with your text editor of choice (`emacs` , `vi` , `nano`) to selectively change `INFO` entries to `WARN` . To change them all easily:

```
sed -i.bak 's/INFO/WARN/' /usr/lib/spark/conf/log4j.properties
```

On Vagrant

If you are running a version of Spark before 2.0, edit `/vagrant/latest-spark/conf/log4j.properties` on the cluster with your text editor of choice (`emacs` , `vi` , `nano`) to selectively change `INFO` entries to `WARN` . To change them all easily:

```
sed -i.bak 's/INFO/WARN/' /vagrant/latest-spark/conf/log4j.properties
```

Launch Spark's interactive environment

Begin by starting the Pyspark shell from the Linux Shell command line

```
pyspark --master yarn-client         // start in client mode
```

At the `>>>` prompt, examine the Spark Context you were handed when you started Spark by typing `sc`. Throughout this lab, we'll show what you type immediately after the `>>>` prompt followed by what pyspark returns.

```
>>> sc
<SparkContext master=yarn appName=PySparkShell>
>>> sc.version
u'2.2.0'
>>> sc.appName
u'PySparkShell'
>>>
```

Play with an RDD

Now create a simple, small RDD by hand:

```
smallprimes = sc.parallelize(Array(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,
```

Try some basic operations on the RDD. Note that a function takes no arguments, no parentheses are needed after the function name.

```
>>> smallprimes = sc.parallelize([2, 3, 5, 7, 11, 13, 17, 19, 23,
... 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97])
>>> smallprimes.count()
25
>>> smallprimes.min()
2
>>> smallprimes.max()
97
>>> smallprimes.sum()
1060
>>> smallprimes.collect()
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 9
>>>
```

Read data from HDFS

A more realistic scenario is to create an RDD from data read from disk. Spark can natively access HDFS and

S3 in addition to the local file system. Try reading in the stock quote data from HDFS:

```
>>> rdd = sc.textFile("hdfs:///data/stocks-flat/input")
>>> rdd.first()
u'NASDAQ,AAIT,2015-06-22,35.299999,35.299999,35.299999,35.299999,300,35.299999'
>>> rdd.count()
2131092
>>>
```

Simple Transformations and Actions

Let's first filter the data set so we only see AAPL records:

```
>>> aapl = rdd.filter(lambda line: "AAPL" in line)
>>> aapl.count()
8706
>>>
```