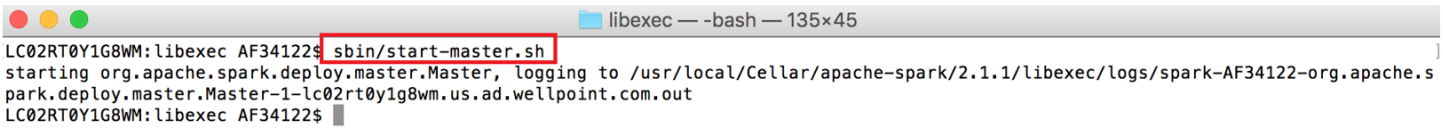
Spark Streaming with Standard Streaming Scripts

Spark Streaming allows us to have application based streaming. This will help us to track statistics about pages in real time, perform real-time analytics, collect events to train machine learning models or automatically detect anomalies.

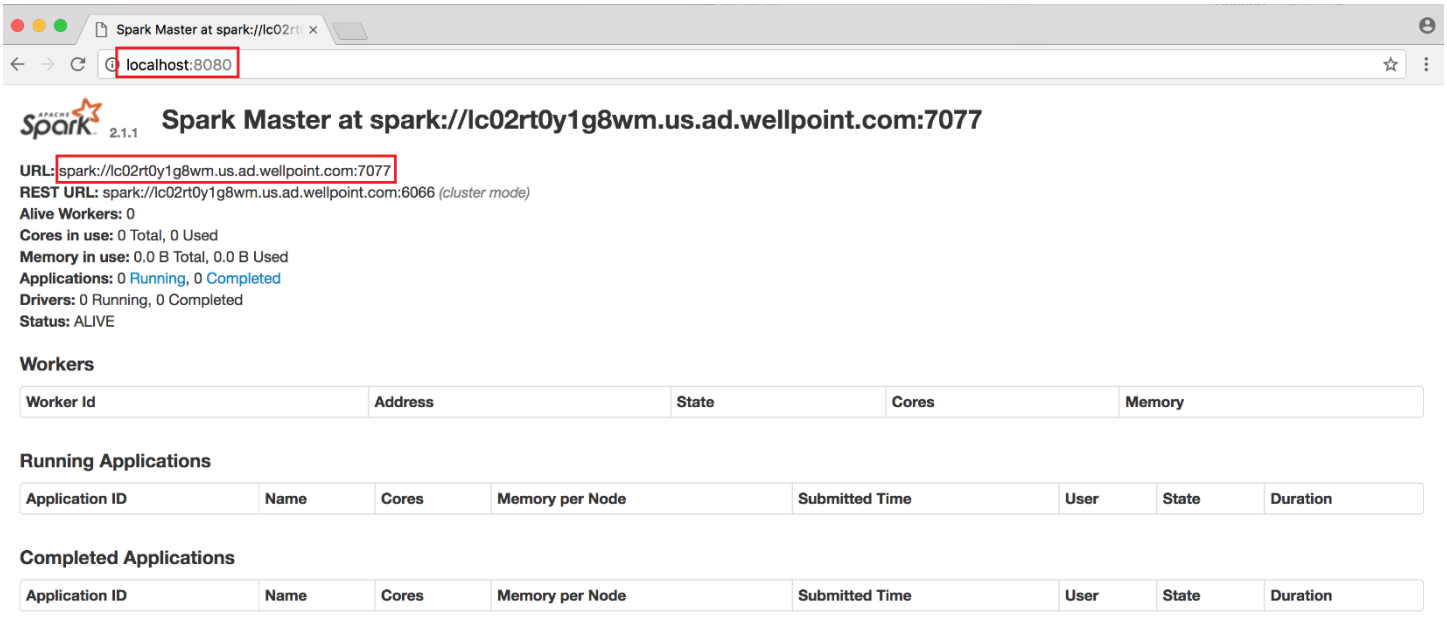
In this project, we will work with the standard streaming scripts provided by Spark on the Standalone cluster and test its efficiency.

Starting the Clusters

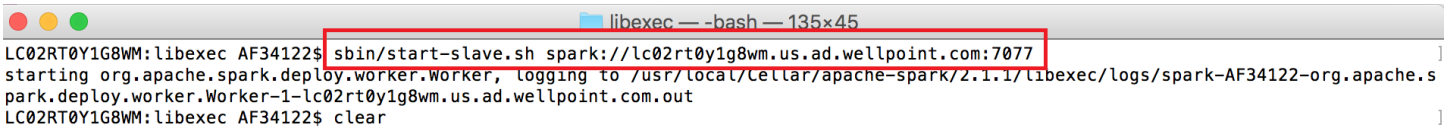
1. First, we need to start the Master node of the Spark Standalone Cluster.



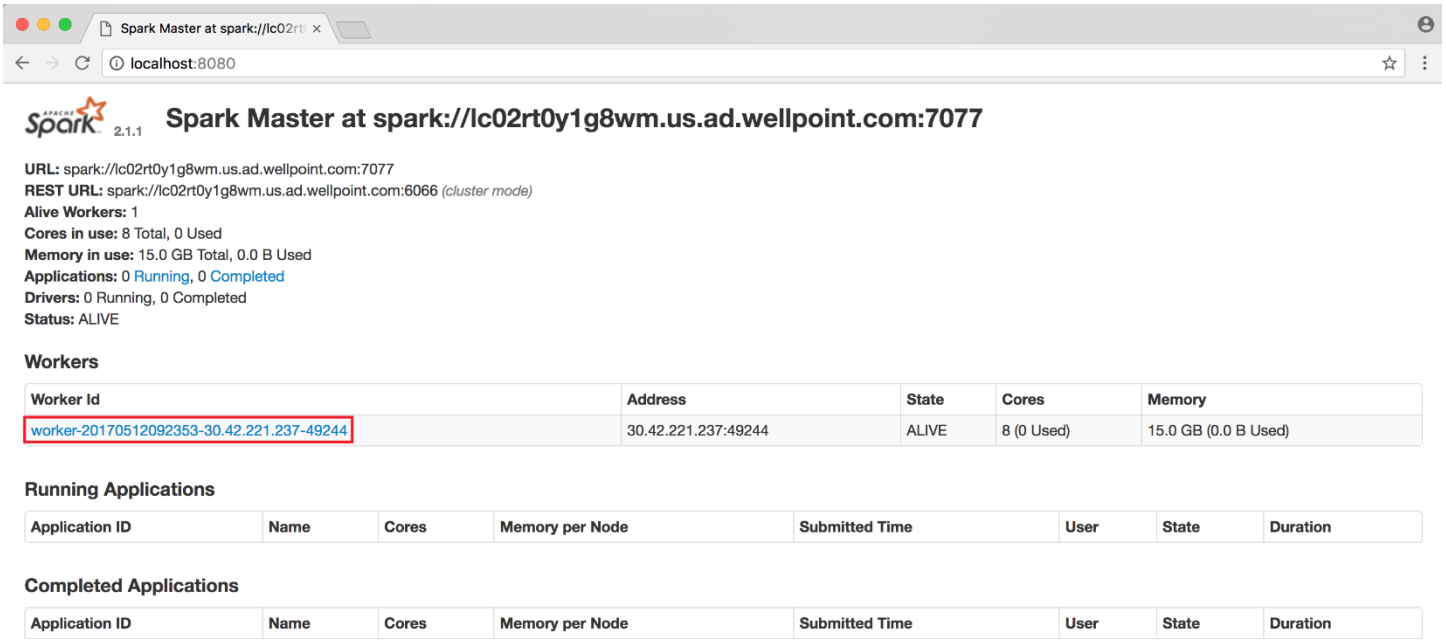
2. We can check if the Master node has started by opening a web browser and opening the address "localhost:8080". We see the Screen showing the details of the Master Node. Keep in mind the URL of the Master Node as this will be required later.



3. Next start the Slave Node for the Cluster. Include the URL for the Master Node as a parameter.



4. We can verify if the node has properly initiated in the same address.



Starting the Streaming Program

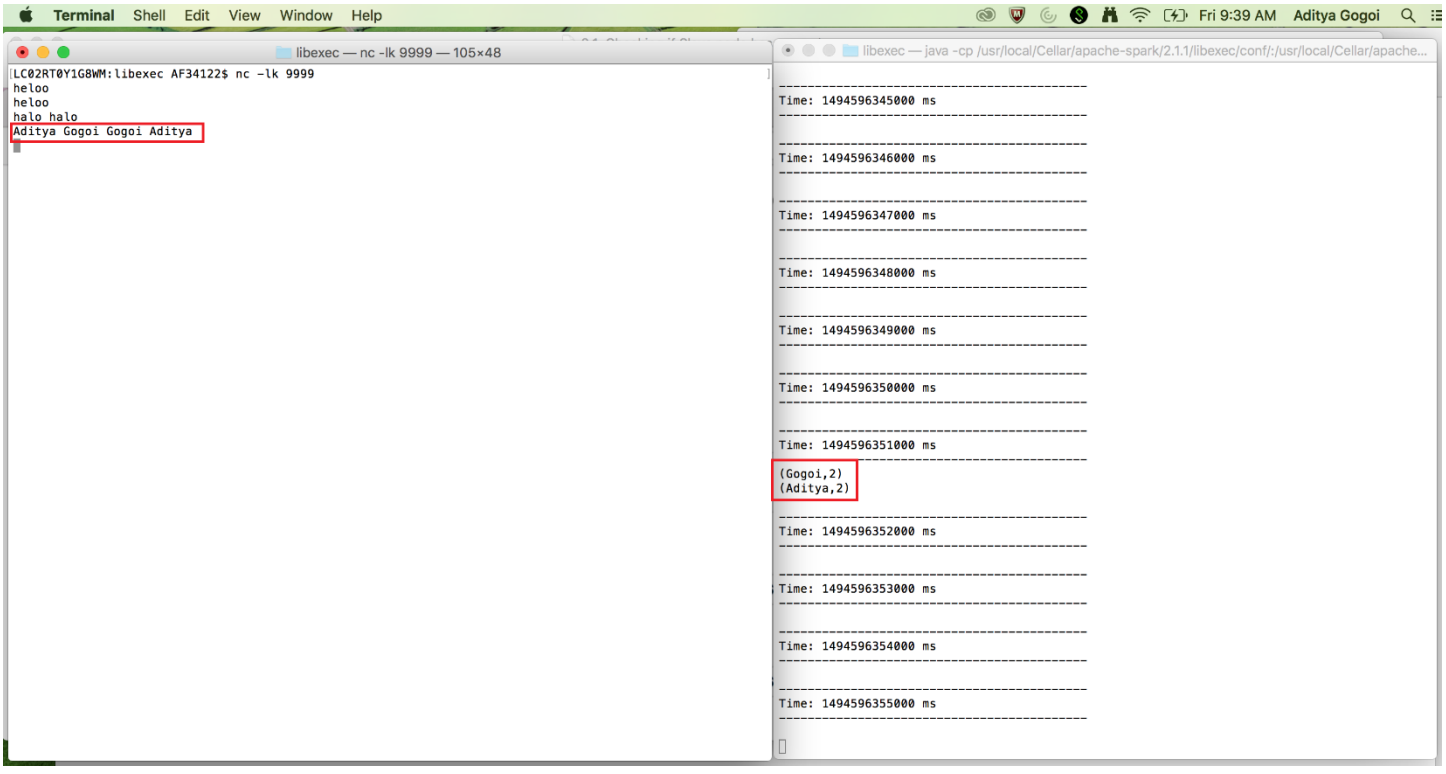
1. Run netcap command on Port 9999. I am using Mac OS the command may vary with the OS. This command basically allows us to send data to a specific port.

2. On a different terminal, from the run-example folder of Spark, start the ‘streaming.NetworkWordCount’ with the local port of 9999, the same port where we are running netcap. You will see timestamps after every second on the second, the timestamps being in the format of seconds since the Unix epoch in 1970.

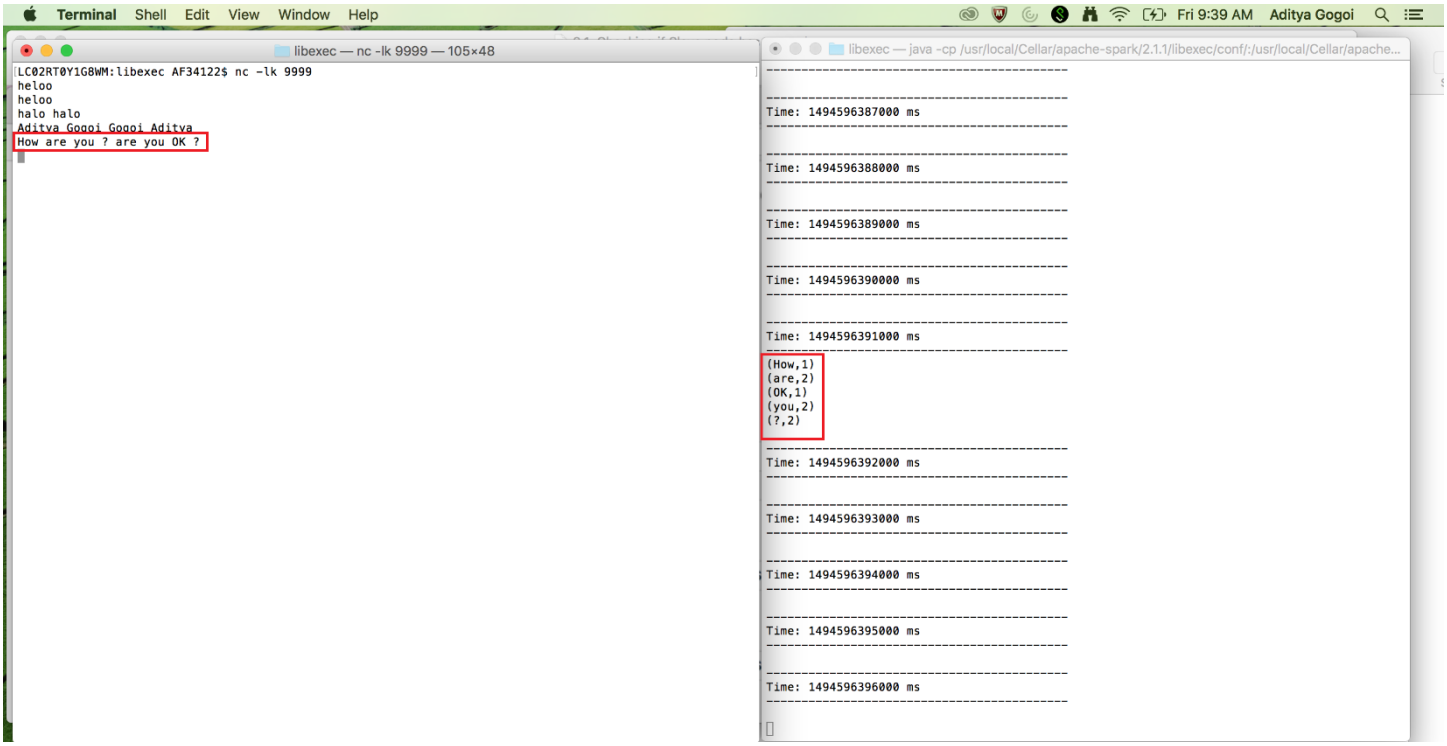
```
libexec — java -cp /usr/local/Cellar/apache-spark/2.1.1/libexec/conf/:/usr/local/Cellar/apache...
LC02RT0Y1G8WM:libexec AF34122$ bin/run-example streaming.NetworkWordCount localhost 9999
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
17/05/12 09:38:22 INFO StreamingExamples: Setting log level to [WARN] for streaming example
. To override add a custom log4j.properties to the classpath.
17/05/12 09:38:22 WARN NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
-------------------------------------------
Time: 1494596305000 ms
-------------------------------------------

-------------------------------------------
Time: 1494596306000 ms
-------------------------------------------

-------------------------------------------
Time: 1494596307000 ms
-------------------------------------------

-------------------------------------------
Time: 1494596308000 ms
-------------------------------------------

-------------------------------------------
Time: 1494596309000 ms
-------------------------------------------

-------------------------------------------
Time: 1494596310000 ms
-------------------------------------------

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

Testing the Streaming

1. Keeping both the windows side-by-side, start entering a few messages on the netcap command window. First we will enter a few words on a line. We will notice that every time we enter the data and press enter, Spark not only gathers the data in that second, it also reduces it into groups with word count.



2. Next we will just make another try with a coherent sentence. We see that Spark Streaming easily parses the sentence into its component words and reduces it to its wordcount.



We can see that Spark Streaming is a powerful tool which provides streaming capabilities on any application or network. This proves useful when we have a constant stream of data and we need insights on it on a regular time interval.