Apache Spark Streaming Handbook

Sasikumar Venkatesh

June 19, 2020

1 Basic Socket Streaming

In this section we will be experimenting a simple basic TCP/IPv4 Socket Streaming.

What is a Socket?

A Socket is a channel for communiccation between client and server machines. This happens over TCP protocol and basically created as follows.

Syntax: IP-Address:Port

Example: 127.0.0.1:8000

Lets run a simple socket on the port 8000 on our local machine

```
$ nc -lk 8000

//Lets Send Some Message
10
20
30
40
10
20
30
40
```

Consumer Program in Spark-Streaming

```
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._ // not necessary
    since Spark 1.3

// Create a local StreamingContext with two working thread and batch
    interval of 1 second.
// The master requires 2 cores to prevent a starvation scenario.

val conf = new
    SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")
val ssc = new StreamingContext(conf, Seconds(1))

// Create a DStream that will connect to hostname:port, like
    localhost:8000
val lines = ssc.socketTextStream("localhost", 8000)

// Split each line into words
val words = lines.flatMap(_.split(" "))

import org.apache.spark.streaming.StreamingContext._ // not necessary
    since Spark 1.3
// Count each word in each batch
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)

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

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

However, this program cannot be run on the spark-shell we need to run this in a compiler rather than the interpreter.

Lets setup the IDE for running the program, we will use Eclipse. To download and anage Spark and Spark Streaming Jars we will use Apache Maven.

1. Create a new Maven Project in Eclipse

   File ->New Project ->Create a Simple Project

   Enter Group Id: com.example

   Enter Artificat Id: Spark-Streaming-Example

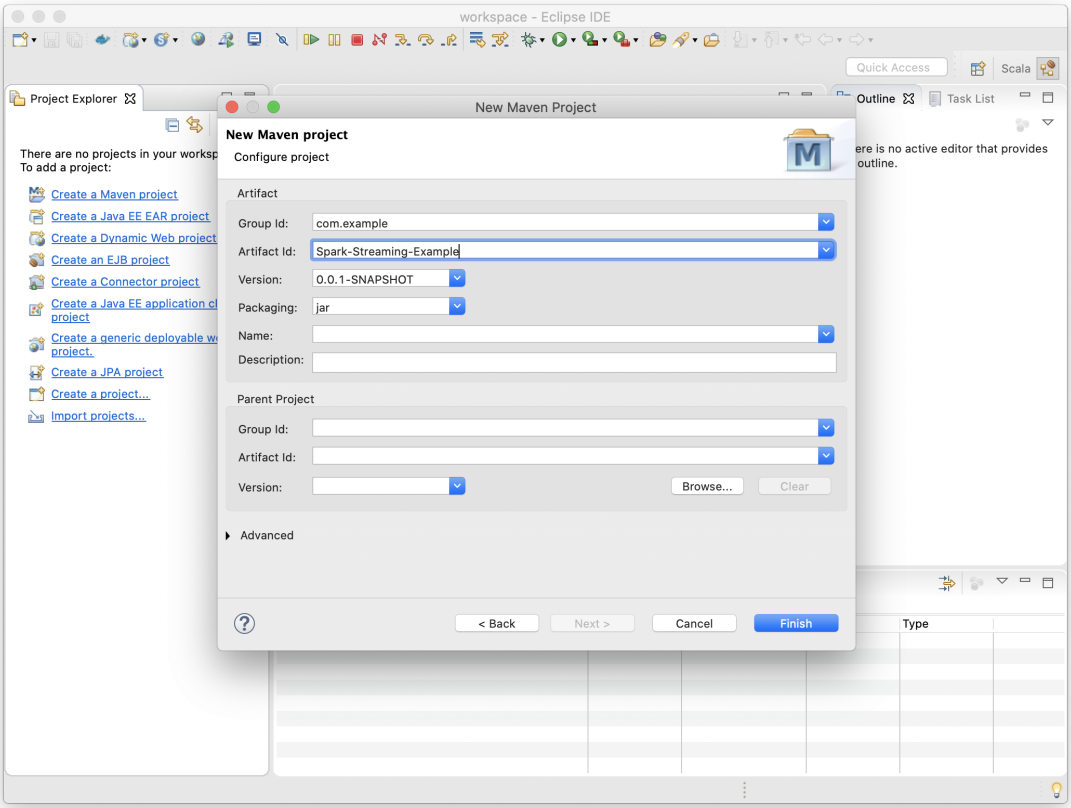

Figure 1: Creating a Maven Projectl

2. Make sure the Scala Plugin in Installed. If not, goto Market Place and install scala plugin.

3. Add Scala Nature to the Project Righ Click on the Project and then Add Scala Nature.

4. Rename the src/main/java source folder into src/main/scala similarly src/test/java to src/test/scala.

5. Add the below dependencies into *pom.xml* file.

```
<dependencies>
   <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.12</artifactId>
      <version>2.4.5</version>
   </dependency>
   <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-streaming_2.12</artifactId>
      <version>2.4.5</version>
   </dependency>
</dependencies>
```

6. Create a new Packge under src/main/scala

   Name: com.example.spark.streaming

7. Create a new Scala Object under com.example.spark.streaming

   Name: NetworkWordCount.scala

8. Add the Consumer Program to the main function of the NetworkWordCount.scala.

```
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._ //
    not necessary since Spark 1.3


object NetworkWordCount{
   def main(args: Array[String]): Unit {

      // Create a local StreamingContext with two working
          thread and batch interval of 1 second.
      // The master requires 2 cores to prevent a
          starvation scenario.

      val conf = new
          SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")
      val ssc = new StreamingContext(conf, Seconds(1))

      // Create a DStream that will connect to
          hostname:port, like localhost:8000
      val lines = ssc.socketTextStream("localhost", 8000)

      // Split each line into words
      val words = lines.flatMap(_.split(" "))

      // Count each word in each batch
      val pairs = words.map(word => (word, 1))
      val wordCounts = pairs.reduceByKey(_ + _)

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

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

}
```

9. Lets run the above program in Eclipse by Right Click ->Run As Scala Application. The output looks as follows.