09. Getting started with Spark & Hadoop – client mode on local & cluster mode on yarn

Posted on November 15, 2018

This extends Setting up & getting started with Spark local mode with Sbt & Scala and Setting up & getting started with Hadoop on Mac.

The Spark job written in Scala will be reading the data from HDFS.

300+ Java Interview FAQs

300+ Java FAQs 🔥

16+ Java Key Areas Q&As

150+ Java Architect FAQs

80+ Java Code Quality Q&As

150+ Java Coding Q&As

300+ Big Data Interview FAQs

300+ Big Data FAQs 🔥

Tutorials - Big Data

TUT - ⏯ Starting Big Data

TUT - Starting Spark & Scala

1. CSV data in HDFS

```
Maths, 85
English, 94
Science, 75
```

2. Make sure HDFS & YARN services are up

You can check if the services are up with:

```
jps -lm
```

If the services are down start them with:

```
$ start-all.sh
```

3. Load the data on to HDFS from the local file system

```

$ hdfs dfs -mkdir -p /data/marks
$ hdfs dfs -put ./test.csv /data/marks

```

4. Spark job to read from HDFS

This is an extension to the SimpleSpark.java covered in the previous Spark tutorials in the series setting-up-getting-started-with-scala-sbt-spark-hadoop

```

package com.scalaproject

import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.Dataset;

```

TUT - Starting with Python
TUT - Kafka
TUT - Pig
TUT - Apache Storm
TUT - Spark Scala on Zeppelin
TUT - Cloudera
TUT - Cloudera on Docker
TUT - File Formats
TUT - Spark on Docker
TUT - Flume
TUT - Hadoop (HDFS)
TUT - HBase (NoSQL)
TUT - Hive (SQL)
TUT - Hadoop & Spark
TUT - MapReduce
TUT - Spark and Scala
TUT - Spark & Java
TUT - PySpark on Databricks
TUT - Zookeeper

```
object SimpleSpark {

  def main(args: Array[String]): Unit = {
     val spark = SparkSession.builder.appName("Sir
                              .config("spark.maste
                              .getOrCreate()

     spark.read
          .option("delimiter",",")
          .option("header","false")
          .csv("hdfs://localhost:9000/data/marks/t

     spark.stop()

  }
}
```

Within Eclipse you can "Run As" -> "Scala Application"

Output:

```
 //.........
18/11/14 23:35:01 INFO CodeGenerator: Code generat
18/11/14 23:35:01 INFO Executor: Finished task 0.0
18/11/14 23:35:01 INFO TaskSetManager: Finished ta
18/11/14 23:35:01 INFO TaskSchedulerImpl: Removed
18/11/14 23:35:01 INFO DAGScheduler: ResultStage 1
18/11/14 23:35:01 INFO DAGScheduler: Job 1 finishe
+-------+---+
|    _c0|_c1|
+-------+---+
|  Maths| 85|
|English| 94|
|Science| 75|
+-------+---+

18/11/14 23:35:01 INFO SparkUI: Stopped Spark web
18/11/14 23:35:01 INFO MapOutputTrackerMasterEndpo
18/11/14 23:35:01 INFO MemoryStore: MemoryStore cl
//.........

```

5. Package with sbt

```
~/projects/sbt-tutorial]$ sbt package

```

6. Spark-submit local & client mode

Spark-submit to run the master on the local & in client mode:

```
~/projects/sbt-tutorial]$ spark-submit --class com
```

You will get the same output:

```
2018-11-14 23:50:03 INFO  CodeGenerator:54 - Code
2018-11-14 23:50:03 INFO  Executor:54 - Finished t
2018-11-14 23:50:03 INFO  TaskSetManager:54 - Fin
2018-11-14 23:50:03 INFO  TaskSchedulerImpl:54 - R
2018-11-14 23:50:03 INFO  DAGScheduler:54 - Result
2018-11-14 23:50:03 INFO  DAGScheduler:54 - Job 1
+-------+---+
|    _c0|_c1|
+-------+---+
|  Maths| 85|
|English| 94|
|Science| 75|
+-------+---+

2018-11-14 23:50:03 INFO  AbstractConnector:318 -
2018-11-14 23:50:03 INFO  SparkUI:54 - Stopped Spa
2018-11-14 23:50:03 INFO  MapOutputTrackerMasterEn
2018-11-14 23:50:03 INFO  MemoryStore:54 - MemoryS
2018-11-14 23:50:03 INFO  BlockManager:54 - BlockM

```

6. Spark-submit yarn & cluster mode

Spark-submit to run the master on the yarn & in cluster mode:

Firstly, in the source code remove ".config("spark.master", "local")". The source code should look like:

```
package com.scalaproject
```

```
2
3  import org.apache.spark.sql.SparkSession;
4  import org.apache.spark.sql.Dataset;
5
6  object SimpleSpark {
7
8    def main(args: Array[String]): Unit = {
9       val spark = SparkSession.builder.appName("Sir
10                                .getOrCreate()
11
12      spark.read
13           .option("delimiter",",")
14           .option("header","false")
15           .csv("hdfs://localhost:9000/data/marks/t
16
17      spark.stop()
18
19   }
20 }
21
```

Package it:

```
1 ~/projects/sbt-tutorial]$ sbt package
2
```

Set the following environment variables (e.g. ~/.bash_profile or ~/.bashrc):

```
1 export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
2 export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

Run it on yarn in cluster mode:

```
1 ~/projects/sbt-tutorial]$ spark-submit --class com
2
```

Check http://localhost:8088/cluster for the output. Click on the id and then the logs.

Output:

```
+-------+---+
|    _c0|_c1|
+-------+---+
|  Maths| 85|
|English| 94|
|Science| 75|
+-------+---+

```

You can practice the Spark code that are covered in the Spark tutorials on Apache Zeppelin on Docker.

‹ 08. Setting up & getting started with Hadoop on Mac

10. Setting up & getting started with Hive ›

Disclaimer