BIG DATA HADOOP AND SPARK DEVLOPMENT

ASSIGNMENT 25

Table of Contents:

BIG DATA HADOOPAND SPARK DEVELOPMENT

1. Introduction

In this assignment, the given tasks are performed and Output of the tasks are recorded in the form of Screenshots.

2. Objective

This Assignment consolidates the deeper understanding of the Session – 25 Bigdata Ecosystem Integration

3. Problem Statement

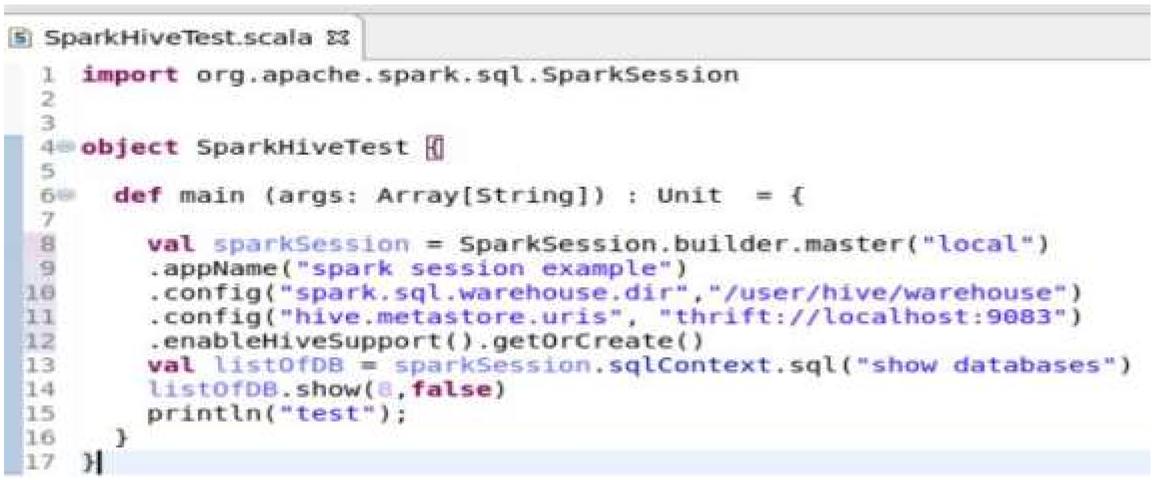
- Task 1
  - As discussed in class integrate Spark Hive
- Task 2
  - As discussed in class integrate Spark Hbase
- Task 3
  - As discussed in class integrate Spark Kafka

4. Expected Output

- Task 1

  - **As discussed in class integrate Spark Hive**

    - Open acadgild VM and start all daemons – Hadoop + Hbase using below commands. - Start-all.sh - Start-hbase.sh
    - Now open Eclipse in VM and create a project named “TestScalaProject” and import the source code “SparkHiveTest.scala”



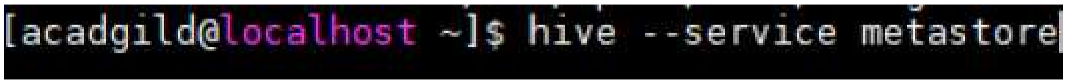
SparkHiveTest.scala

```
import org.apache.spark.sql.SparkSession


object SparkHiveTest {

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

    val sparkSession = SparkSession.builder.master("local")
    .appName("spark session example")
    .config("spark.sql.warehouse.dir","/user/hive/warehouse")
    .config("hive.metastore.uris", "thrift://localhost:9083")
    .enableHiveSupport().getOrCreate()
    val listOfDB = sparkSession.sqlContext.sql("show databases")
    listOfDB.show(8,false)
    println("test");
  }
}
```

- Resolve all the compilation errors from external JARs which are required and then execute the class as a Scala Application.

- Copy hive-site.xml file from $HIVE_HOME/conf to $SPARK_HOME/conf.

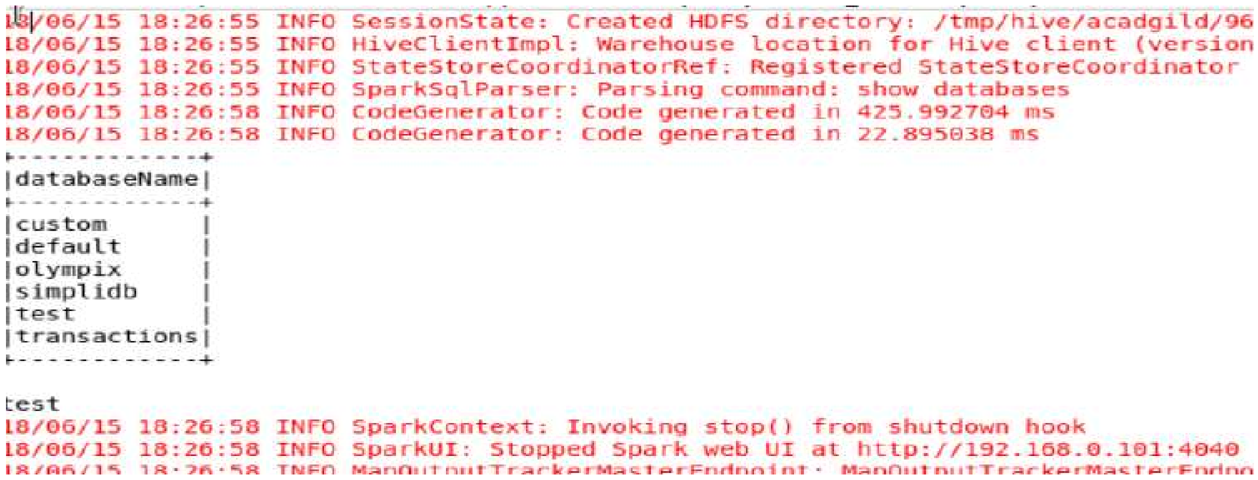
- Add the following property to hive-site.xml on Spark side:
  <property>
  <name>hive.metastore.uris</name>
  <value>thrift://localhost:9083</value>
  <description>password for connecting to mysql server</description>
  </property>
  This property helps to create the connection between Spark and Hive.
- Start hive metastore using the command – “hive –service metastore”



```
[acadgild@localhost ~]$ hive --service metastore
```

Make sure mysql service is running or else execute the command “sudo service mysqld start”.

- Run the class in eclipse as “Scala Application”.



```
18/06/15 18:26:55 INFO SessionState: Created HDFS directory: /tmp/hive/acadgild/96
18/06/15 18:26:55 INFO HiveClientImpl: Warehouse location for Hive client (version
18/06/15 18:26:55 INFO StateStoreCoordinatorRef: Registered StateStoreCoordinator
18/06/15 18:26:55 INFO SparkSqlParser: Parsing command: show databases
18/06/15 18:26:58 INFO CodeGenerator: Code generated in 425.992704 ms
18/06/15 18:26:58 INFO CodeGenerator: Code generated in 22.895038 ms
+------------+
|databaseName|
+------------+
|custom      |
|default     |
|olympix     |
|simplidb    |
|test        |
|transactions|
+------------+

test
18/06/15 18:26:58 INFO SparkContext: Invoking stop() from shutdown hook
18/06/15 18:26:58 INFO SparkUI: Stopped Spark web UI at http://192.168.0.101:4040
18/06/15 18:26:58 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpo
```

Here we could see the list of databases in Hive using Apache Spark – Hadoop Integration.

Task 2:

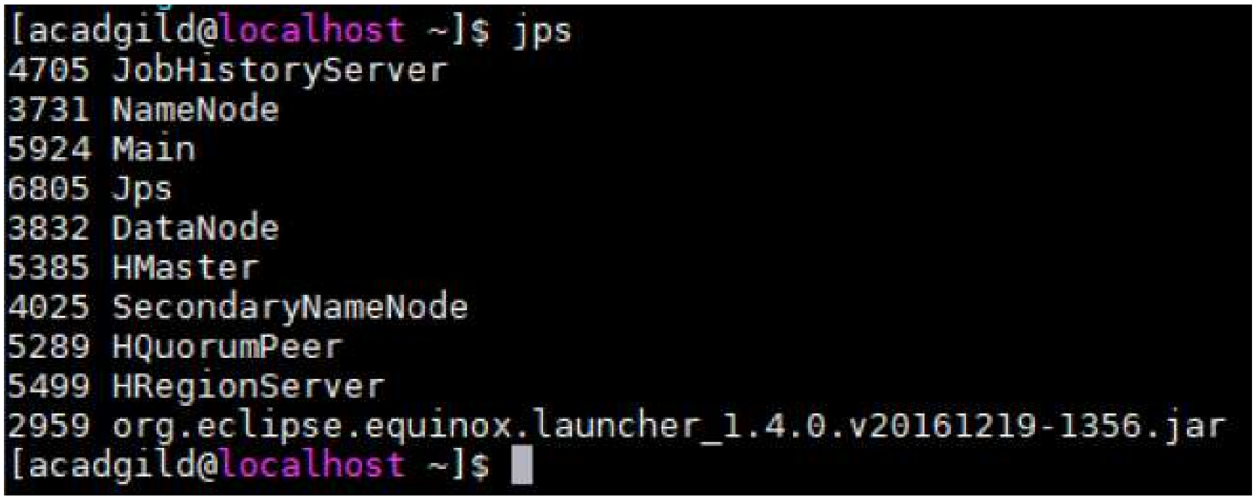
- As discussed in class integrate Spark Hbase

Follow the below steps to proceed for Spark Hbase integration

- Open acadgild VM and start all daemons – Hadoop + Hbase using below commands.

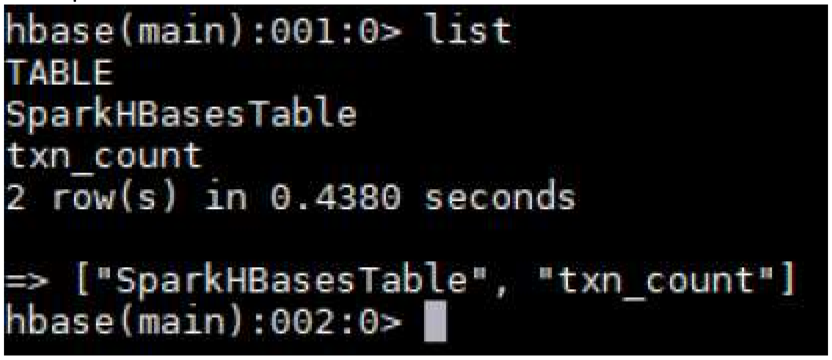
  - Start-all.sh
  - Start-hbase.sh
  - Mr-jobhistory-daemon.sh start history server

  Check all daemons are running as in below screenshot.



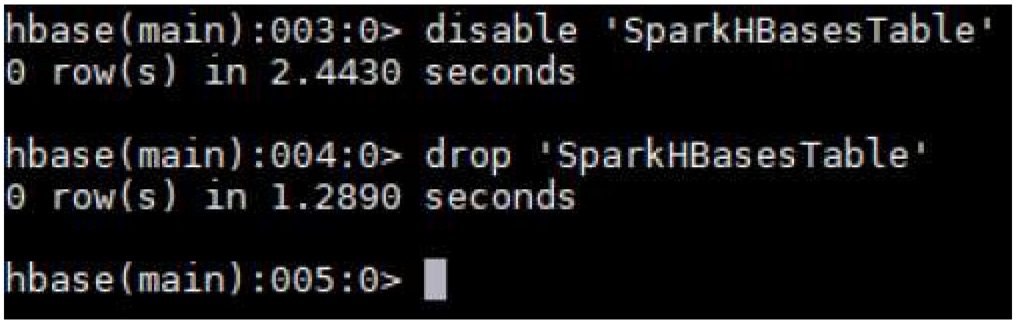
```
[acadgild@localhost ~]$ jps
4705 JobHistoryServer
3731 NameNode
5924 Main
6805 Jps
3832 DataNode
5385 HMaster
4025 SecondaryNameNode
5289 HQuorumPeer
5499 HRegionServer
2959 org.eclipse.equinox.launcher_1.4.0.v20161219-1356.jar
[acadgild@localhost ~]$
```

- Open HBase shell and execute the list command.



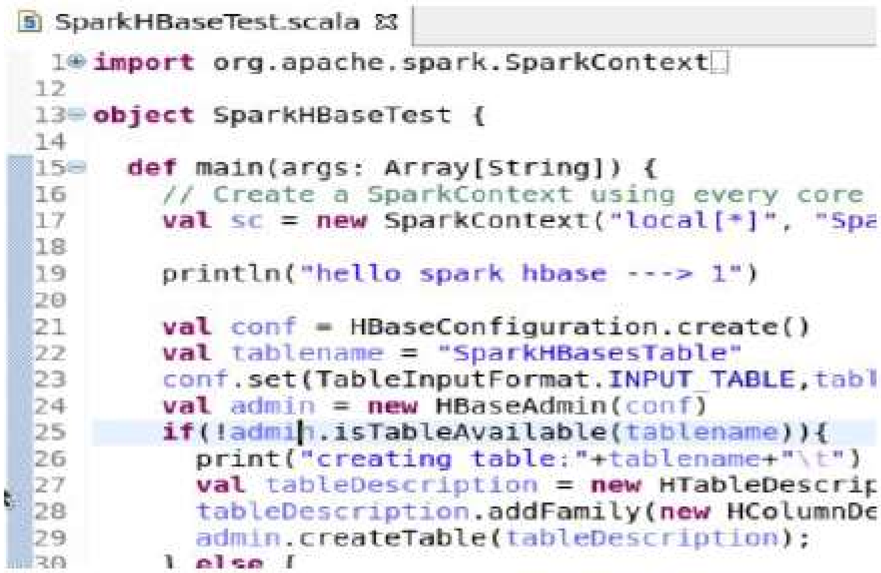
```
hbase(main):001:0> list
TABLE
SparkHBasesTable
txn_count
2 row(s) in 0.4380 seconds

=> ["SparkHBasesTable", "txn_count"]
hbase(main):002:0>
```

- Disable and drop the SparkHBasesTable.



```
hbase(main):003:0> disable 'SparkHBasesTable'
0 row(s) in 2.4430 seconds

hbase(main):004:0> drop 'SparkHBasesTable'
0 row(s) in 1.2890 seconds

hbase(main):005:0> 
```

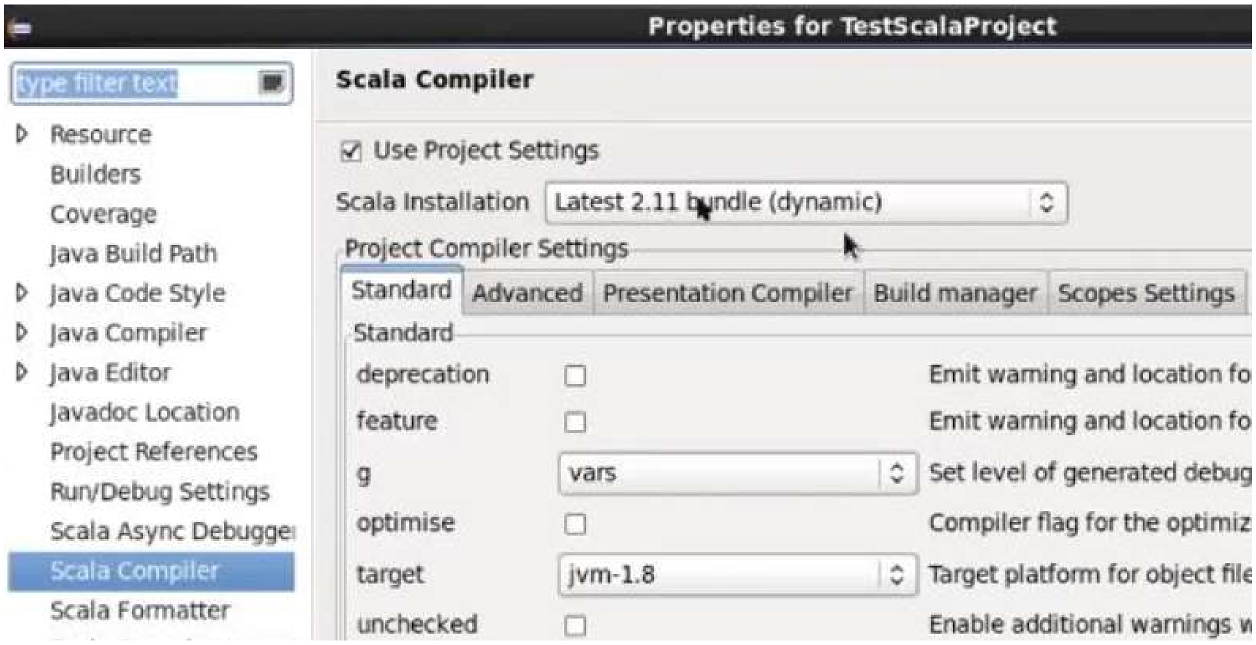
- Now open Eclipse in VM and create a project named “TestScalaProject” and import the source code “SparkHBaseTest.scala”



SparkHBaseTest.scala

```
import org.apache.spark.SparkContext

object SparkHBaseTest {

  def main(args: Array[String]) {
    // Create a SparkContext using every core
    val sc = new SparkContext("local[*]", "Spa

    println("hello spark hbase ---> 1")

    val conf = HBaseConfiguration.create()
    val tablename = "SparkHBasesTable"
    conf.set(TableInputFormat.INPUT_TABLE,tabl
    val admin = new HBaseAdmin(conf)
    if(!admin.isTableAvailable(tablename)){
      print("creating table:"+tablename+"\t")
      val tableDescription = new HTableDescrip
      tableDescription.addFamily(new HColumnDe
      admin.createTable(tableDescription);
    } else {
```

- Resolve all the error by adding on the external JARs from Spark folder and make sure the Scala compiler version is "Latest 2.11 bundle (dynamic)".



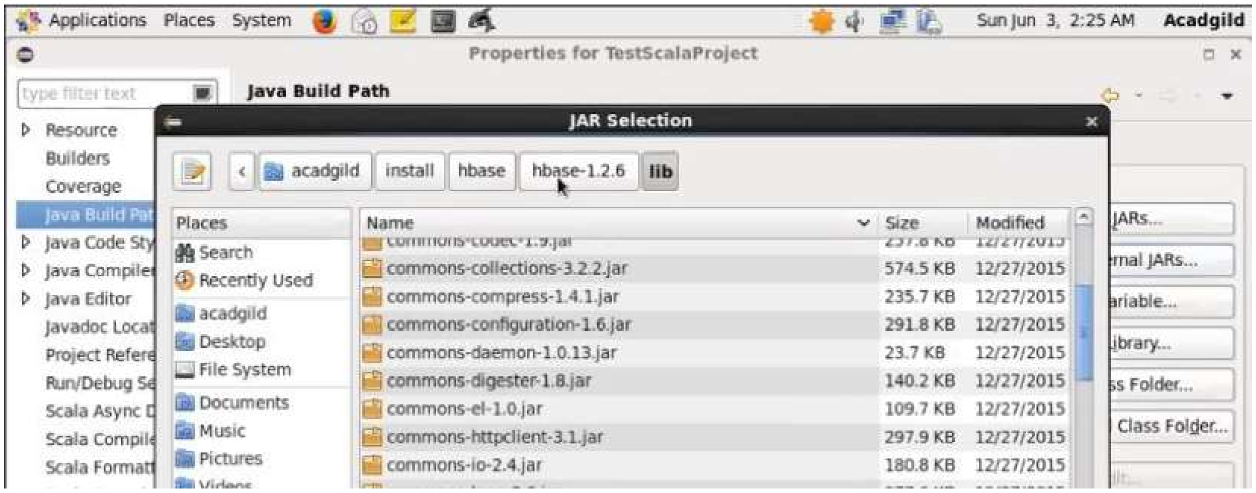
Scala Compiler

☑ Use Project Settings

Scala Installation: Latest 2.11 bundle (dynamic)

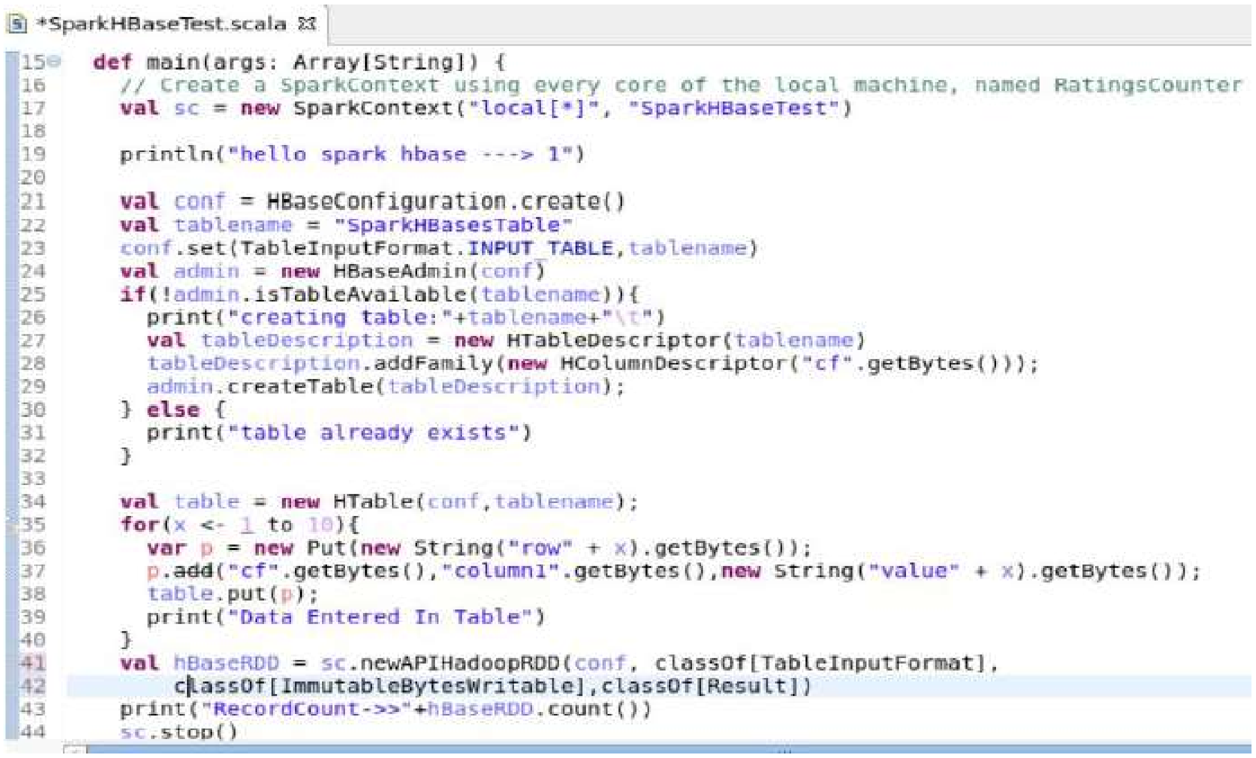
deprecation	☐	Emit warning and location fo
feature	☐	Emit warning and location fo
g	vars	Set level of generated debug
optimise	☐	Compiler flag for the optimiz
target	jvm-1.8	Target platform for object file
unchecked	☐	Enable additional warnings w

JAR details from this path:



Name	Size	Modified
commons-collections-3.2.2.jar	574.5 KB	12/27/2015
commons-compress-1.4.1.jar	235.7 KB	12/27/2015
commons-configuration-1.6.jar	291.8 KB	12/27/2015
commons-daemon-1.0.13.jar	23.7 KB	12/27/2015
commons-digester-1.8.jar	140.2 KB	12/27/2015
commons-el-1.0.jar	109.7 KB	12/27/2015
commons-httpclient-3.1.jar	297.9 KB	12/27/2015
commons-io-2.4.jar	180.8 KB	12/27/2015

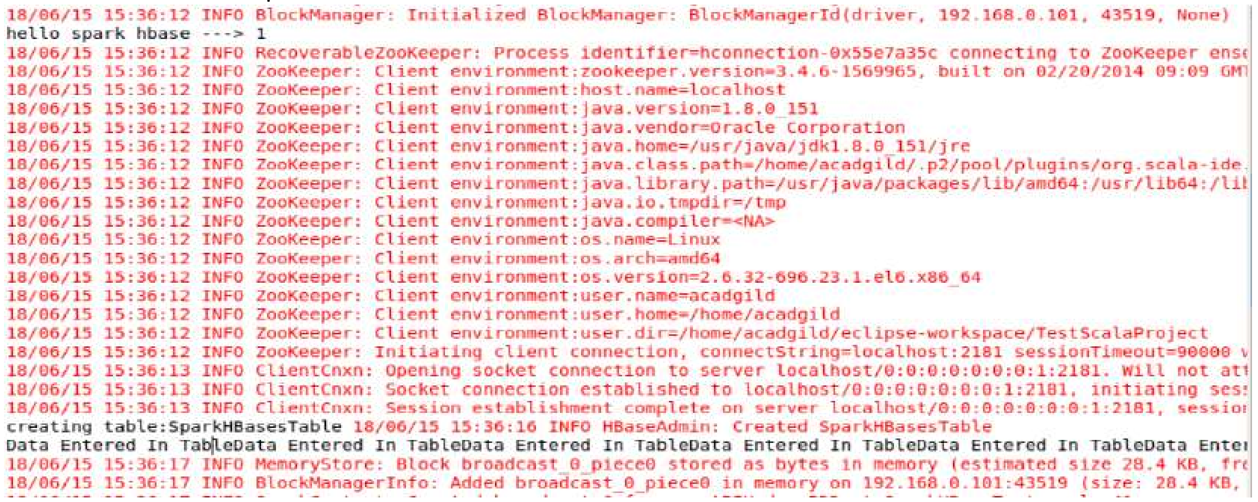
- Now run the class "SparkHBaseTest.scala" as Scala application.



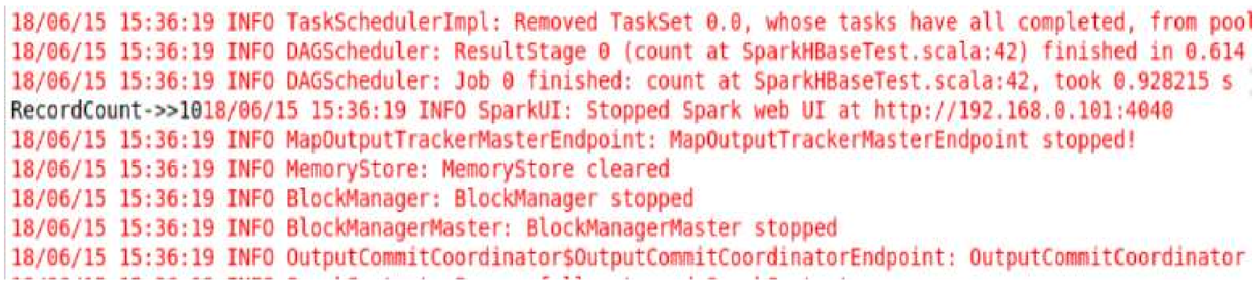
*SparkHBaseTest.scala

```scala
  def main(args: Array[String]) {
    // Create a SparkContext using every core of the local machine, named RatingsCounter
    val sc = new SparkContext("local[*]", "SparkHBaseTest")

    println("hello spark hbase ---> 1")

    val conf = HBaseConfiguration.create()
    val tablename = "SparkHBasesTable"
    conf.set(TableInputFormat.INPUT_TABLE,tablename)
    val admin = new HBaseAdmin(conf)
    if(!admin.isTableAvailable(tablename)){
      print("creating table:"+tablename+"\t")
      val tableDescription = new HTableDescriptor(tablename)
      tableDescription.addFamily(new HColumnDescriptor("cf".getBytes()));
      admin.createTable(tableDescription);
    } else {
      print("table already exists")
    }

    val table = new HTable(conf,tablename);
    for(x <- 1 to 10){
      var p = new Put(new String("row" + x).getBytes());
      p.add("cf".getBytes(),"column1".getBytes(),new String("value" + x).getBytes());
      table.put(p);
      print("Data Entered In Table")
    }
    val hBaseRDD = sc.newAPIHadoopRDD(conf, classOf[TableInputFormat],
      classOf[ImmutableBytesWritable],classOf[Result])
    print("RecordCount->>"+hBaseRDD.count())
    sc.stop()
```

- Above code – creates a new table and enters input values to the table in HBase using Spark API. Below are the output screens for the same.



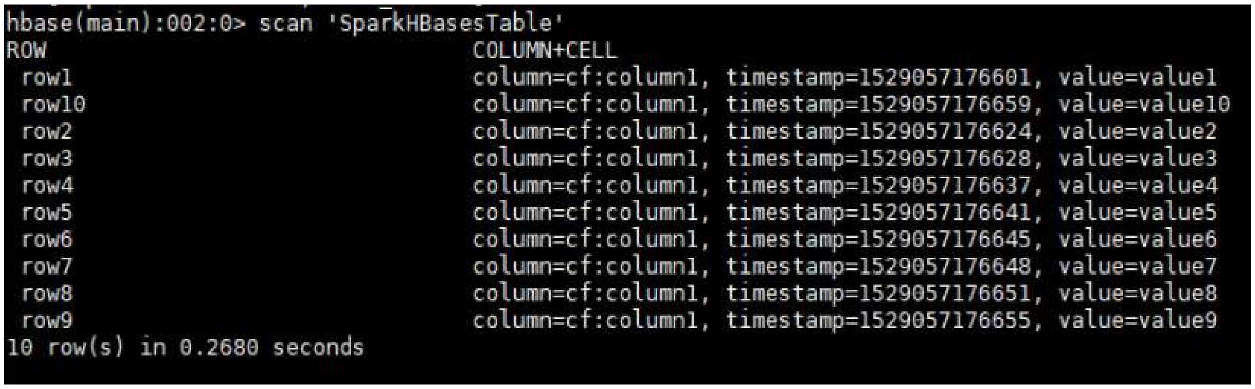
```
18/06/15 15:36:12 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 192.168.0.101, 43519, None)
hello spark hbase ---> 1
18/06/15 15:36:12 INFO RecoverableZooKeeper: Process identifier=hconnection-0x55e7a35c connecting to ZooKeeper ens
18/06/15 15:36:12 INFO ZooKeeper: Client environment:zookeeper.version=3.4.6-1569965, built on 02/20/2014 09:09 GM
18/06/15 15:36:12 INFO ZooKeeper: Client environment:host.name=localhost
18/06/15 15:36:12 INFO ZooKeeper: Client environment:java.version=1.8.0_151
18/06/15 15:36:12 INFO ZooKeeper: Client environment:java.vendor=Oracle Corporation
18/06/15 15:36:12 INFO ZooKeeper: Client environment:java.home=/usr/java/jdk1.8.0_151/jre
18/06/15 15:36:12 INFO ZooKeeper: Client environment:java.class.path=/home/acadgild/.p2/pool/plugins/org.scala-ide.
18/06/15 15:36:12 INFO ZooKeeper: Client environment:java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/li
18/06/15 15:36:12 INFO ZooKeeper: Client environment:java.io.tmpdir=/tmp
18/06/15 15:36:12 INFO ZooKeeper: Client environment:java.compiler=<NA>
18/06/15 15:36:12 INFO ZooKeeper: Client environment:os.name=Linux
18/06/15 15:36:12 INFO ZooKeeper: Client environment:os.arch=amd64
18/06/15 15:36:12 INFO ZooKeeper: Client environment:os.version=2.6.32-696.23.1.el6.x86_64
18/06/15 15:36:12 INFO ZooKeeper: Client environment:user.name=acadgild
18/06/15 15:36:12 INFO ZooKeeper: Client environment:user.home=/home/acadgild
18/06/15 15:36:12 INFO ZooKeeper: Client environment:user.dir=/home/acadgild/eclipse-workspace/TestScalaProject
18/06/15 15:36:12 INFO ZooKeeper: Initiating client connection, connectString=localhost:2181 sessionTimeout=90000 v
18/06/15 15:36:13 INFO ClientCnxn: Opening socket connection to server localhost/0:0:0:0:0:0:0:1:2181. Will not at
18/06/15 15:36:13 INFO ClientCnxn: Socket connection established to localhost/0:0:0:0:0:0:0:1:2181, initiating ses
18/06/15 15:36:13 INFO ClientCnxn: Session establishment complete on server localhost/0:0:0:0:0:0:0:1:2181, sessio
creating table:SparkHBasesTable 18/06/15 15:36:16 INFO HBaseAdmin: Created SparkHBasesTable
Data Entered In TableData Entered In TableData Entered In TableData Entered In TableData Entered In TableData Ente
18/06/15 15:36:17 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 28.4 KB, fr
18/06/15 15:36:17 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on 192.168.0.101:43519 (size: 28.4 KB,
```



```
18/06/15 15:36:19 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
18/06/15 15:36:19 INFO DAGScheduler: ResultStage 0 (count at SparkHBaseTest.scala:42) finished in 0.614
18/06/15 15:36:19 INFO DAGScheduler: Job 0 finished: count at SparkHBaseTest.scala:42, took 0.928215 s
RecordCount->>1018/06/15 15:36:19 INFO SparkUI: Stopped Spark web UI at http://192.168.0.101:4040
18/06/15 15:36:19 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/06/15 15:36:19 INFO MemoryStore: MemoryStore cleared
18/06/15 15:36:19 INFO BlockManager: BlockManager stopped
18/06/15 15:36:19 INFO BlockManagerMaster: BlockManagerMaster stopped
18/06/15 15:36:19 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator
```

Here we have got the output in the eclipse console.

Now check the same in the terminal using hbase shell.



```
hbase(main):002:0> scan 'SparkHBasesTable'
ROW                                  COLUMN+CELL
 row1                                column=cf:column1, timestamp=1529057176601, value=value1
 row10                               column=cf:column1, timestamp=1529057176659, value=value10
 row2                                column=cf:column1, timestamp=1529057176624, value=value2
 row3                                column=cf:column1, timestamp=1529057176628, value=value3
 row4                                column=cf:column1, timestamp=1529057176637, value=value4
 row5                                column=cf:column1, timestamp=1529057176641, value=value5
 row6                                column=cf:column1, timestamp=1529057176645, value=value6
 row7                                column=cf:column1, timestamp=1529057176648, value=value7
 row8                                column=cf:column1, timestamp=1529057176651, value=value8
 row9                                column=cf:column1, timestamp=1529057176655, value=value9
10 row(s) in 0.2680 seconds
```

- Task 3
    - **As discussed in class integrate Spark Kafka**

Program which runs the word count program by reading the contents from kafka and run in spark.

//imports required for the program

```
import com.test.schema.ContactType;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.function.*;
import org.apache.spark.streaming.Durations;
import org.apache.spark.streaming.api.java.JavaDStream;
import org.apache.spark.streaming.api.java.JavaInputDStream;
import org.apache.spark.streaming.api.java.JavaPairDStream;
import org.apache.spark.streaming.api.java.JavaStreamingContext;
import org.apache.spark.streaming.kafka010.ConsumerStrategies;
import org.apache.spark.streaming.kafka010.KafkaUtils;
import org.apache.spark.streaming.kafka010.LocationStrategies;
import scala.Tuple2; import java.util.*;

public class SparkKafka10 {
public static void main(String[] args) throws Collection topics = Arrays.asList("WordCount");
```

//setting the spark configuration with local master and setting the appname
//as “SparkKafkaWordCount”

```
SparkConf conf = new
SparkConf().setMaster("local[2]").setAppName("SparkKafkaWordCount");
```

//Read messages in batch of 30 seconds in realtime, by using console
//producer

```
JavaStreamingContext jssc = new JavaStreamingContext(conf, Durations.seconds(30));
```

// Start reading messages from Kafka and get DStream

```
final JavaInputDStream<ConsumerRecord> stream = KafkaUtils.createDirectStream(jssc,
LocationStrategies.PreferConsistent(), ConsumerStrategies.Subscribe(topics,kafkaParams));
```

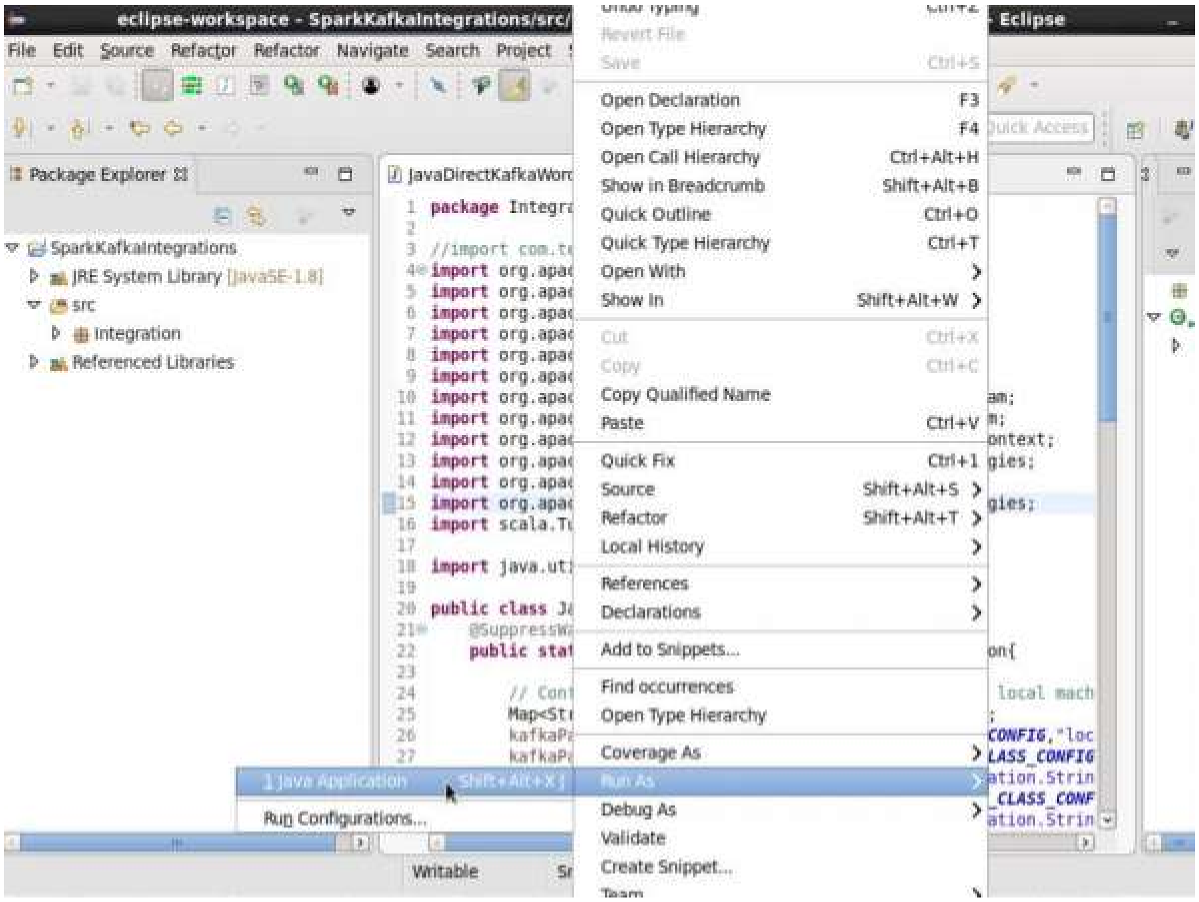
// Read value of each message from Kafka and return it

```
JavaDStream lines = stream.map(new Function<ConsumerRecord, String>() { @Override public
String call(ConsumerRecord kafkaRecord) throws Exception { return kafkaRecord.value(); } });
```

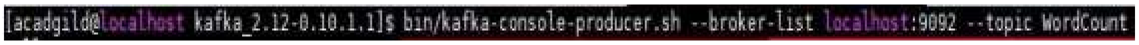
// Break every message into words and return list of words

```
JavaDStream words = lines.flatMap(new FlatMapFunction() { @Override public Iterator
call(String line) throws Exception { return Arrays.asList(line.split(" ")).iterator(); } });

// Take every word and return Tuple with (word,1)
JavaPairDStream wordMap = words.mapToPair(new PairFunction() {
@Override public Tuple2 call(String word) throws Exception { return new Tuple2<>(word,1); }
});

// Count occurrence of each word
JavaPairDStream wordCount = wordMap.reduceByKey(new Function2() {
@Override public Integer call(Integer first, Integer second) throws Exception {
return first+second; } });

//Print the word count
wordCount.print();
jssc.start();
jssc.awaitTermination();
}
}
```

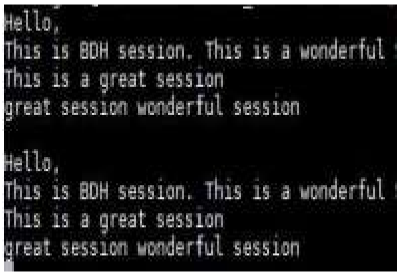
We execute the program in eclipse as shown below:



After executing the run command in eclipse, we open the console producer in the terminal, and input the data as shown below:

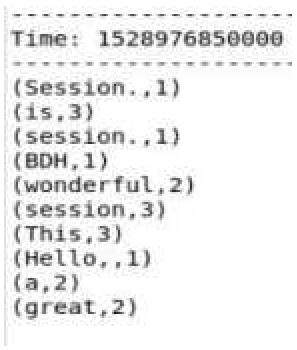


```
[acadgild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic WordCount
```



```
Hello,
This is BDH session. This is a wonderful
This is a great session
great session wonderful session

Hello,
This is BDH session. This is a wonderful
This is a great session
great session wonderful session
```

After 30 seconds, we can see the word count for 30second batch of data. We can see the output in eclipse as shown below:



```
--------------------
Time: 1528976850000
--------------------
(Session.,1)
(is,3)
(session.,1)
(BDH,1)
(wonderful,2)
(session,3)
(This,3)
(Hello,,1)
(a,2)
(great,2)
```