

**DARFT: Spark Story POT Phase 2**

**Use Case 1 - 4**

**Spark Story POT Phase 2**

**Simple Spark job to Dell ECS S3**

The goal is to create a simple Spark job written in Scala, containerize in docker and run through the kubernetes orchestrator. Scala application will read the TPC generated files on the Travelers Dell ECS S3 and join them together and produce result file back to Dell ECS S3.

This documentation will contain a few case usages that have been tried.

Case 1: Scala application

Case 2: Zeppelin

Case 3: Curl connection to Dell EMC ECS S3

Case 4: Perl Script pull files then run Spark application to join the files and Perl script to push back the files

Case 5: Convert Perl script to Scala and run it all through Scala

**Set Spark 2.3.2 environment in K0007048.travp.net**

The setup should be already completed in Simple Spark Story POT Phase 1. Please find the below SharePoint URL  “Spark Story POT Phase 1”

<http://teams.trv.net/cop/kx_ebits/Data_Platforms/Hadoop/Hadoop%20Documentation/Spark2/Spark%20POT%20Phase%201/SimpleSparkStory.docx?d=w152dd2c7b32d4448bc4739e50d4372e9>

**Prerequisite:**

Java 1.8.0

Scala 2.11.8

Spark 2.3.2

SBT – Scala Built Tool

#Login into K0007048 as sudo user

#Login as **sudo user sudo -u support -i**

#Export the following commands to use Spark

export SPARK\_HOME=/opt/spark   
unset JAVA\_HOME

**Case 1: Scala application connected to Dell ECS S3**

ECSMerge.scala

/share\_data/dockertest/spark\_test/ ECSMerge.scala

import java.net.URI

import com.emc.ecs.spark.sql.sources.s3.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.SQLContext

object ECSMerge{

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

System.loadLibrary("com.emc.ecs:spark-ecs-connector\_2.11:1.4.2")

val spark = SparkSession.builder().master("local").appName("SparkTest").getOrCreate()

val endpointUri = new URI("http://10.40.200.11:80/")

val credential = ("ebia-vol-test-user", "qsUJepZHhevNxhyqfXeM97vJ188hkdvajjmga5/b")

val df = spark.read.bucket(endpointUri, credential, "ebia-vol-test", withSystemMetadata = false)

df.registerTempTable("jack\_temp")

val result = spark.sql("SELECT \* FROM jack\_temp")

result.show()

}

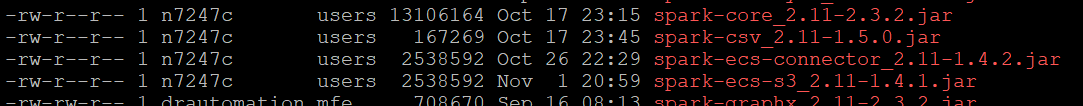
}

Testing the ECSMerge.scala using spark-shell

Step 1:

export SPARK\_HOME=/opt/spark   
unset JAVA\_HOME

Download the EMC ECS connector and put it in /share\_data/spark-2.3.2-bin-hadoop2.6/jars





Run spark-shell

Now paste the code from ECSMerge.scala line by line to test in spark-shell

import java.net.URI

import com.emc.ecs.spark.sql.sources.s3.\_

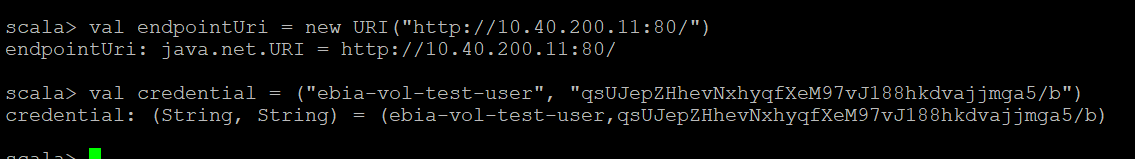
import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.SQLContext

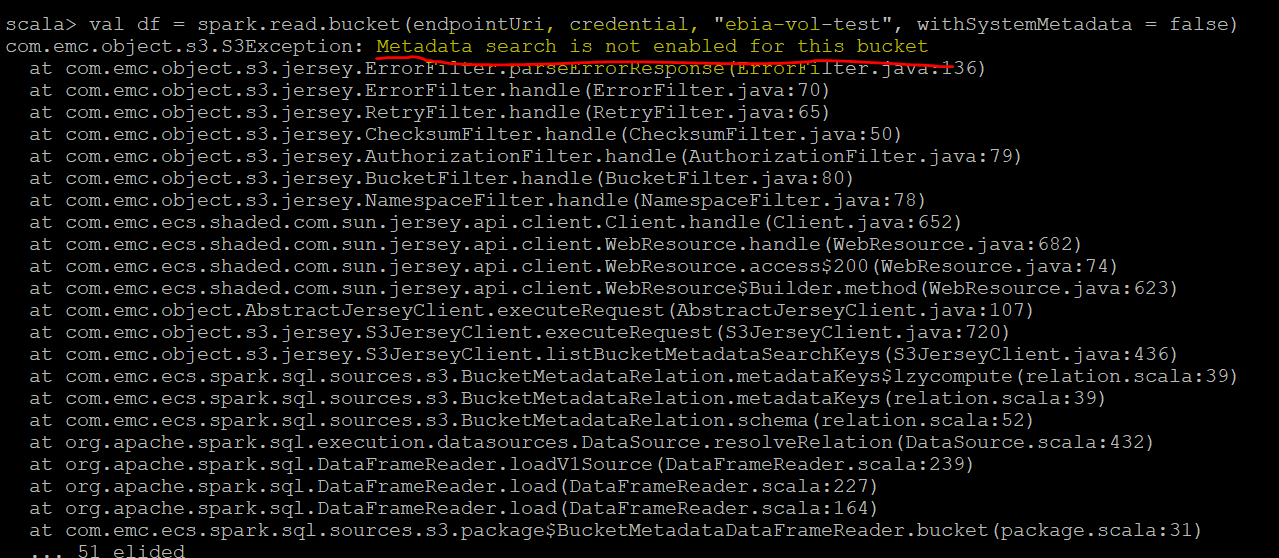


val endpointUri = new URI("http://10.40.200.11:80/")

val credential = ("ebia-vol-test-user", "qsUJepZHhevNxhyqfXeM97vJ188hkdvajjmga5/b")



val df = spark.read.bucket(endpointUri, credential, "ebia-vol-test", withSystemMetadata = false)



Case 1 has come to a blocker. I’ve come to find out that Travelers built this bucket to have search Meta Data set on unable. The solution to fixing this is to enable Metadata search. However, the ECS S3 bucket cannot be configure after it’s been deployed. Another solution is to build a new ECS S3 and set the configuration to have Metadata search.

Above screen shot is the cause of the code not being able to access the files and read the files.

Reference: https://github.com/EMCECS/spark-ecs-connector

**Case 2: Zeppelin**

docker pull djannot/docker-zeppelin

docker run -d -p 8080:8080 it --net=host djannot/docker-zeppelin

docker exec -it djannot/docker-zeppelin bash

apt-get update

apt-get install vim

Go to /usr/hadoop-3.0.0-alpha1/lib and run the command below to download the Amazon S3 SDK:

wget <http://central.maven.org/maven2/com/amazonaws/aws-java-sdk/1.7.4/aws-java-sdk-1.7.4.jar>

After that, go to /usr/hadoop-3.0.0-alpha1/etc/hadoop and edit the core-site.xml file as follows:

<configuration>

<property>

<name>fs.s3a.impl</name>

<value>org.apache.hadoop.fs.s3a.S3AFileSystem</value>

</property>

<property>

<name>fs.s3a.access.key</name>

<value>ebia-vol-test-user</value>

</property>

<property>

<name>fs.s3a.secret.key</name>

<value>qsUJepZHhevNxhyqfXeM97vJ188hkdvajjmga5/b</value>

</property>

<property>

<name>fs.s3a.endpoint</name>

<value>https://ecslab.dvllb.travp.net</value>

</property>

<property>

<name>fs.s3a.connection.ssl.enabled</name>

<value>enabled</value>

</property>

<property>

<name>fs.s3a.signing-algorithm</name>

<value>S3SignerType</value>

</property>

</configuration>

Finally go to /usr/zeppelin and execute the bin/zeppelin.sh command to start Apache Zeppelin. The Zeppelin UI can now be accessed with a web browser on port 8080. This approach blocker is that Traveler’s fire wall access permission isn’t allowing Zeppelin to run on any of our port.

Reference: https://github.com/djannot/ecs-p3/blob/master/spark/spark.md

**Case 3: Curl connection to Dell EMC ECS S3**

This approach is a documentation based on EMC ECS using “curl”. For case 3 I am trying to find connectivity through the Dell ECS bucket. These are some of the steps I’ve tried and errors that I’ve encountered.

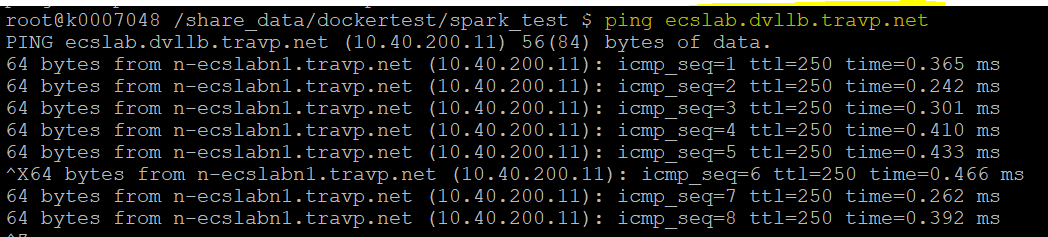
Ping status

curl http://ecslab.dvllb.travp.net/?ping

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><PingList xmlns="http://s

3.amazonaws.com/doc/2006-03-01/"><PingItem><Name>LOAD\_FACTOR</Name><Value>1</Val

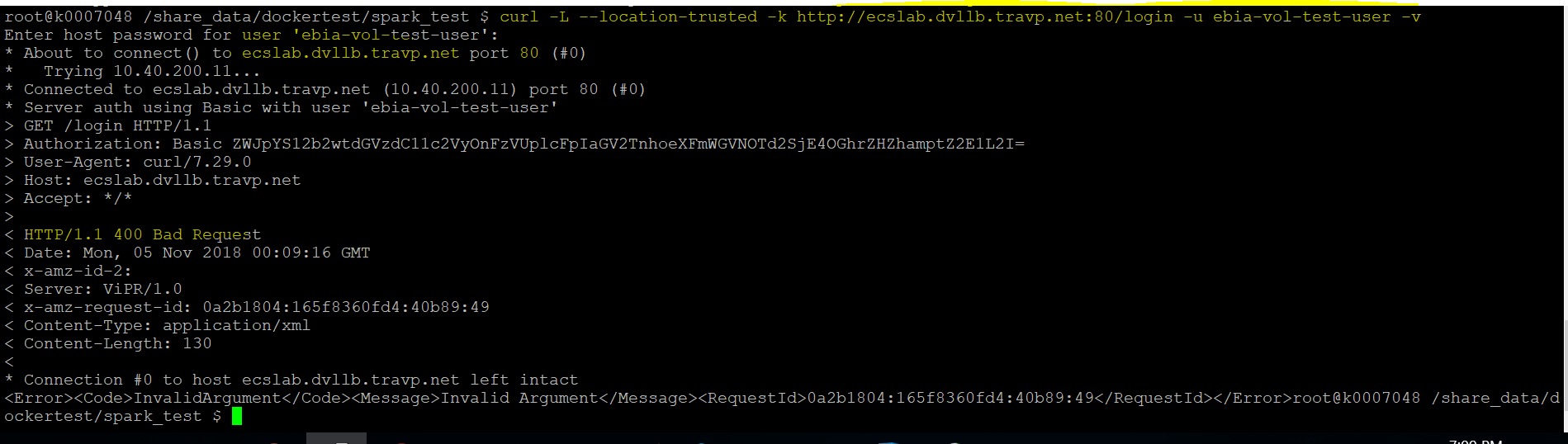
ue></PingItem><PingItem><Name>MAINTENANCE\_MODE</Name><Status>OFF</Status><Text>D



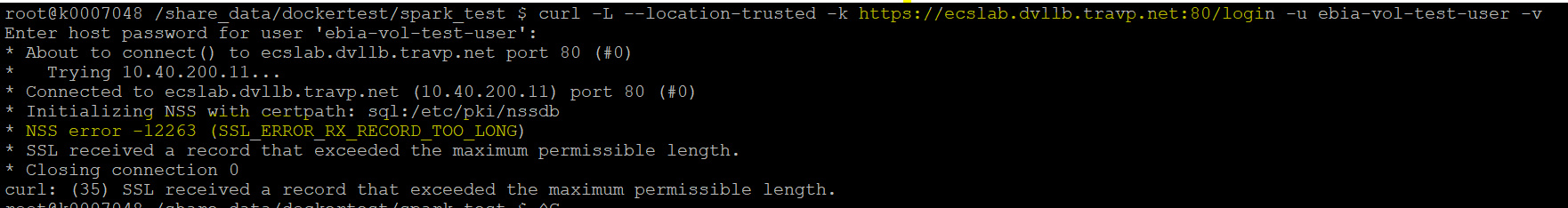
Curl

curl -L --location-trusted -k http://10.40.200.11:80/login -u ebia-vol-test-user -v

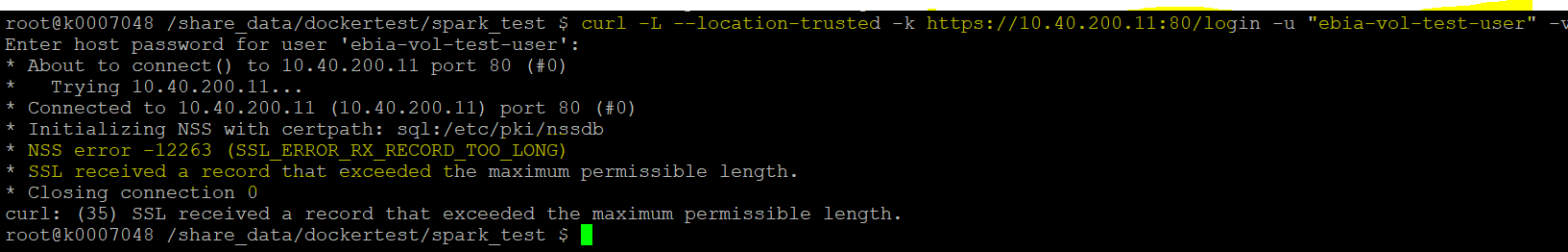


curl -L --location-trusted -k http://ecslab.dvllb.travp.net:80/login -u ebia-vol-test-user -v

curl -L --location-trusted -k https://ecslab.dvllb.travp.net:80/login -u ebia-vol-test-user -v



curl -L --location-trusted -k https://10.40.200.11:80/login -u "ebia-vol-test-user" -v



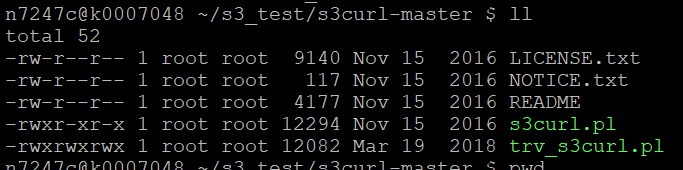
Reference: https://www.emc.com/collateral/TechnicalDocument/docu86295.pdf

Reference to page 115

**Case 4: Perl Script pull files then run Spark application to join the files and Perl script to push back the files**

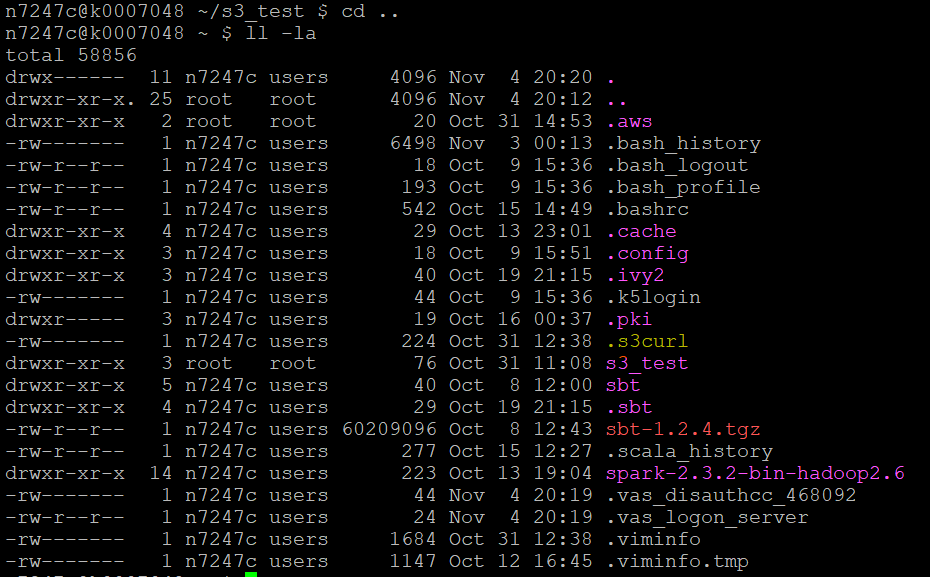
Step 1: Pull the jack\_script.out and gd\_lyrics.out using the perl script trv\_s3curl.pl

cd /home/n7247c/s3\_test/s3curl-master



Step 2: You will have put the Access key and Secret key inside your /home directory. This help minimizes other to have access to the ECS S3 for security reason.

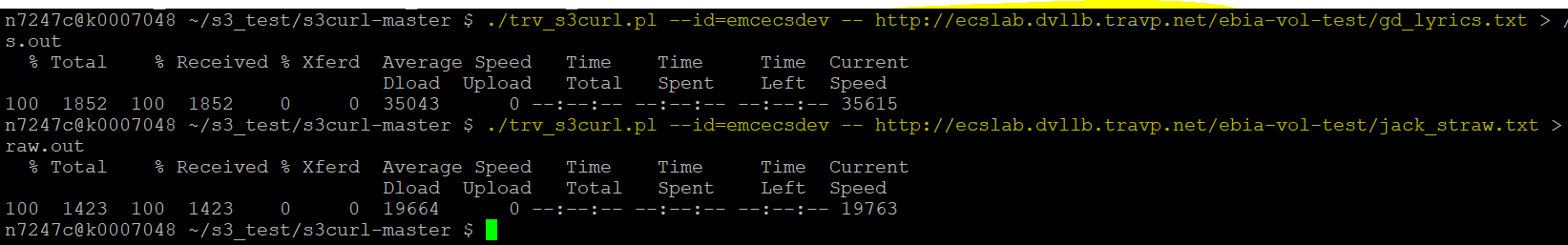
cd /home/n7247c/.s3curl

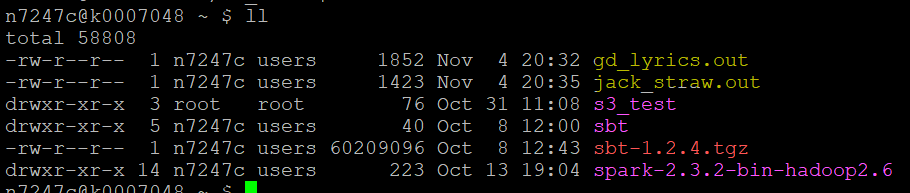


Step 3: Pull the files jack\_script.out and gd\_lyrics.out by using the following command

./trv\_s3curl.pl --id=emcecsdev -- http://ecslab.dvllb.travp.net/ebia-vol-test/gd\_lyrics.txt > /home/n7247c/gd\_lyrics.out

./trv\_s3curl.pl --id=emcecsdev -- http://ecslab.dvllb.travp.net/ebia-vol-test/jack\_straw.txt > /home/n7247c/jack\_straw.out





**EcsMerge.scala**/share\_data/dockertest/spark\_test/ecss3/EcsMerge.scalaimport java.io.\_import scala.io.Sourceimport sys.process.\_

object EcsMerge{

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

//this is my new code (tested it in saprk-shell and its working)

//Load the jack\_straw.out

val file\_jack = Source.fromFile("/home/n7247c/jack\_straw.out").getLines

//Load the gd\_lyrics.out

val file\_gd = Source.fromFile("/home/n7247c/gd\_lyrics.out").getLines

//loading the two files to be added

val added\_files = file\_jack.toList ++ file\_gd.toList

//Output the new added file to a specific location or path

val target\_path = new File("/share\_data/dockertest/spark\_test/ecss3/outPutResult")

//Instantiate a Reader

val bw = new BufferedWriter(new FileWriter(target\_path))

//Add the files in a loop and close when the loop once it's done reading.

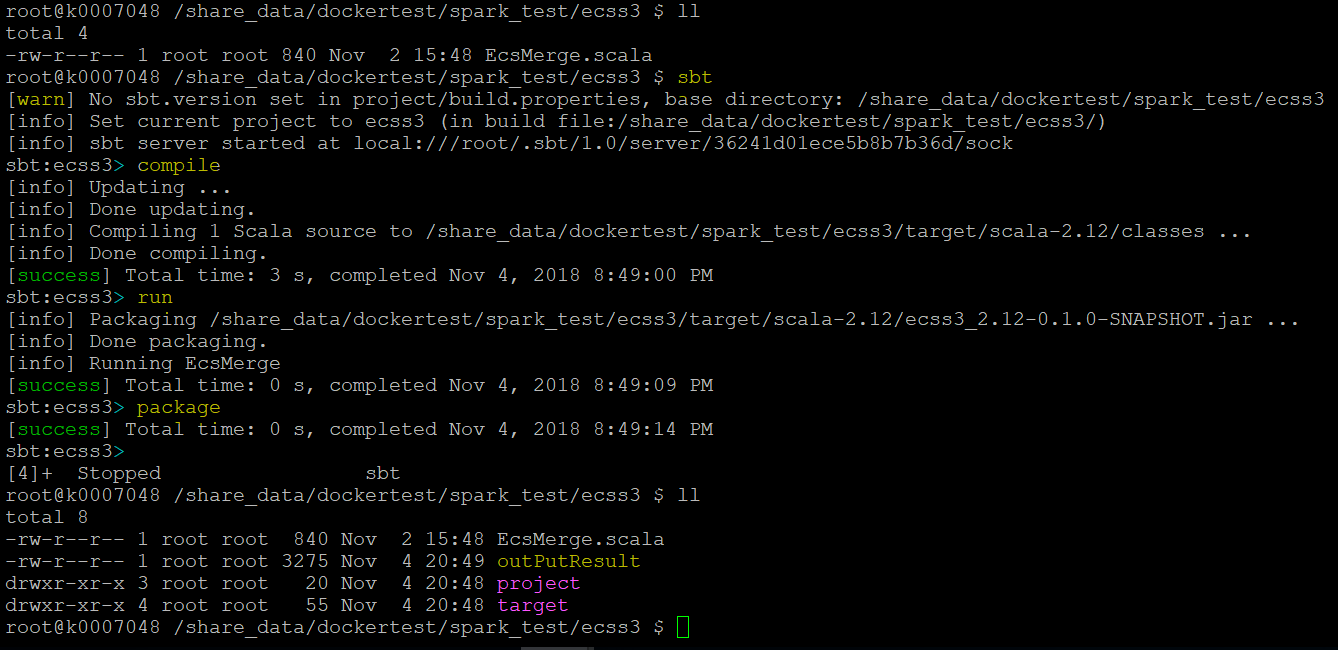
added\_files.foreach(p => bw.write(p + "\n"))

bw.close

}

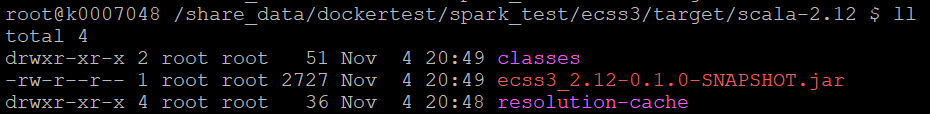
}

Step 4: Use SBT to compile **EcsMerge.scala**



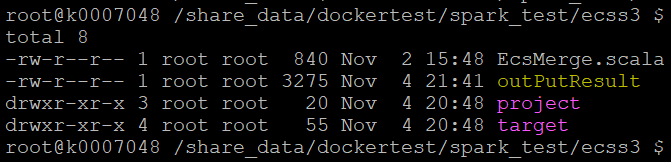
Step 5: We can test the scala application by running the spark-submit on the code using the jar package from SBT.

cd /share\_data/dockertest/spark\_test/ecss3/target/scala-2.12/ ecss3\_2.12-0.1.0-SNAPSHOT.jar

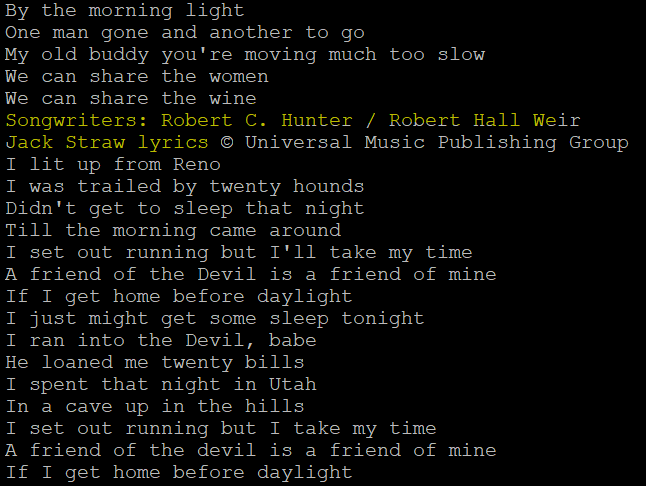


Step 6: Use spark-submit

spark-submit --class EcsMerge --master local[\*] /share\_data/dockertest/spark\_test/ecss3/target/scala-2.12/ecss3\_2.12-0.1.0-SNAPSHOT.jar

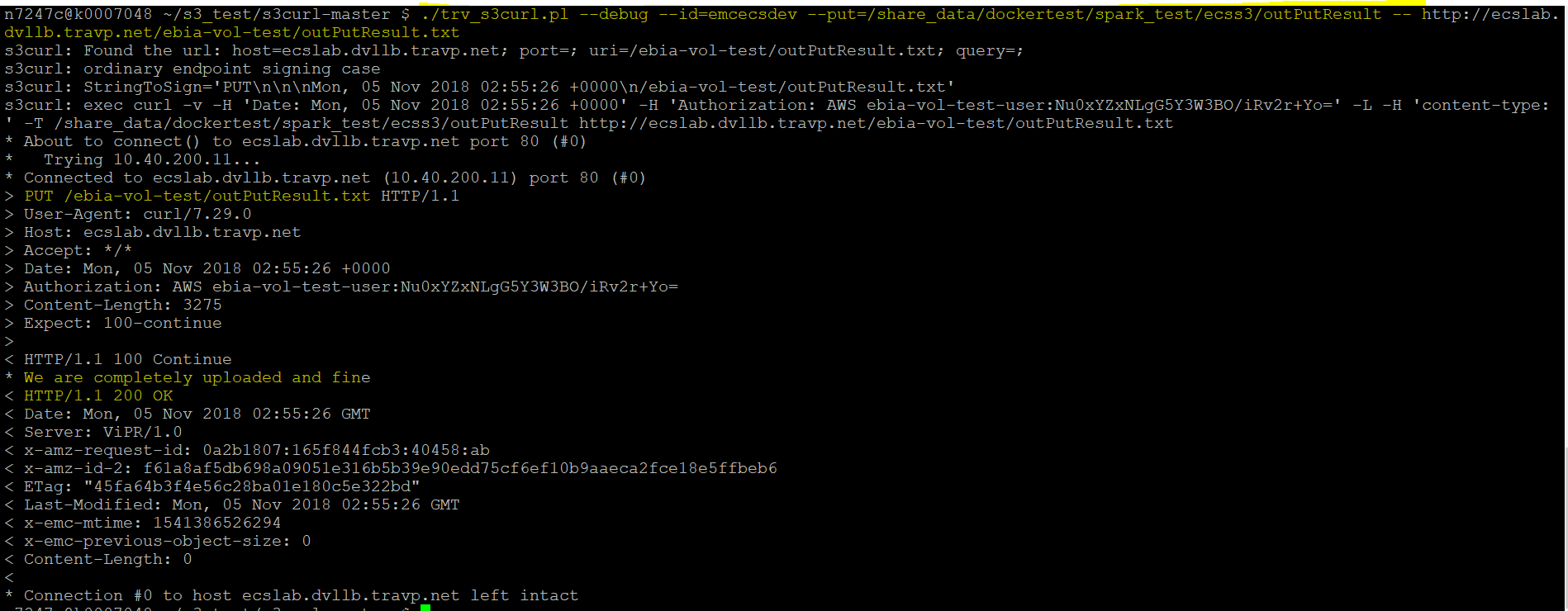


Step 7: view outPutResult



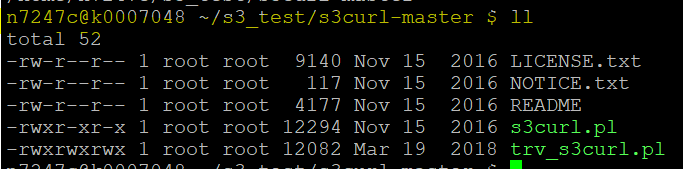
Step 8: Push the outPutResult back to Dell ECS S3

./trv\_s3curl.pl --debug --id=emcecsdev --put=/share\_data/dockertest/spark\_test/ecss3/outPutResult -- <http://ecslab.dvllb.travp.net/ebia-vol-test/outPutResult.txt>

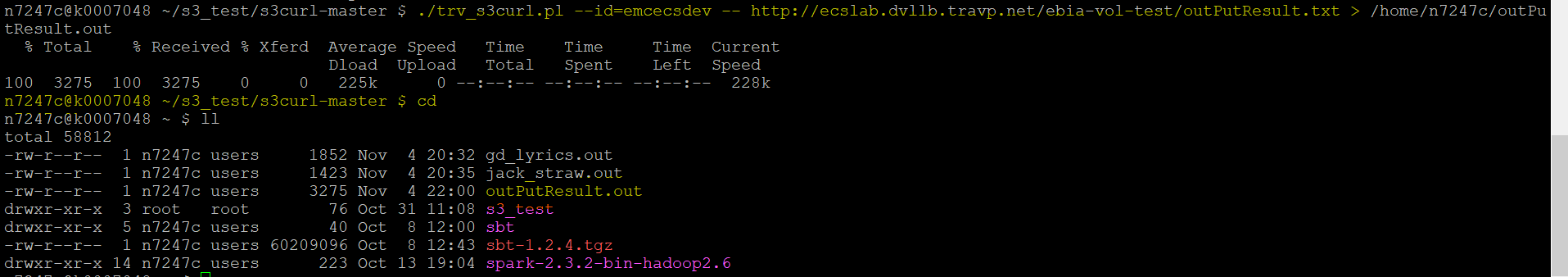


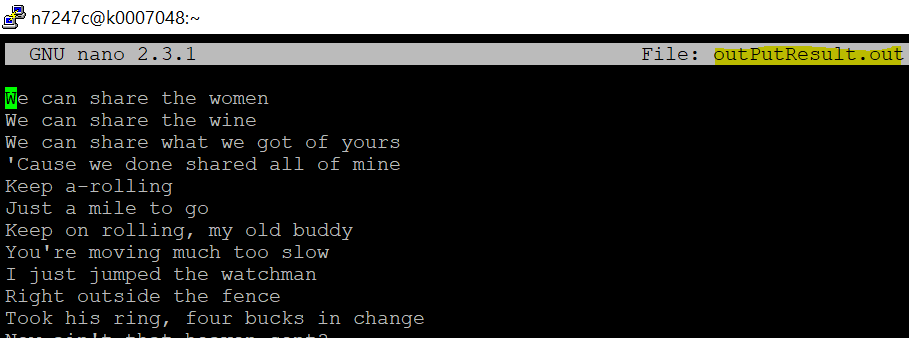
Step 9: To confirm the file is there, pull the file again using the perl script.

cd /home/n7247c/s3\_test/s3curl-master



./trv\_s3curl.pl --id=emcecsdev -- http://ecslab.dvllb.travp.net/ebia-vol-test/outPutResult.txt > /home/n7247c/outPutResult.out





Case 4 seem to be a success. However, it uses two part to this to able to achieve the goal. Using Perl script to get the file from Dell ECS S3 and then run spark job. Then put the file back into Dell ECS S3.

I am still working to find a way around to achieve Case 5. Case 5 goal is to convert Perl script to Scala and run it all through Scala. The goal is to have everything in one application to run the entire process.