SPARK STREAMING

Spark streaming helps in processing streams of data in form Micro-Batches with the help streams method. There are several types of stream sources like Socket, Files, Kafka etc.

We will be demonstrating Structured stream processing in this hands-On.

Copy the **streamsdata** folder the script file [**streamtest-file.sh**] to the desktop of VM machine.

In this exercise, you will read and process streaming data from a set of files.

Display Streaming Data to the Console

We will read data from a file-based stream and display the results to the console. The query in this section is very simple—it does not transform the data, and simply outputs the data it receives "as-is."

1. Review the test data you will be using in this exercise. It contains information about device activations on Loudacre's cellular network in JSON format. The data is in **streamsdata/activations_stream/** directory which you have copied on the Desktop of VM machine.

2. In a terminal session, set up a directory to contain the data files that Spark will read. Set the file permissions to allow your application to access the files. Do not copy any data into the directory yet.

   [root@saispark ~]# **mkdir -p /tmp/devsh-streaming**

   [root@saispark ~]# **chmod +wr /tmp/devsh-streaming**

   Note: The directory from which Spark will load data must exist before the you create the DataFrame based on the data.

3. If you currently have a Spark shell running in a terminal session, exit it. This exercise environment's resources are not sufficient to run a streaming application, so start a new Python or Scala Spark shell running locally instead of on the cluster.

On Scala

```
[root@saispark ~]# spark-shell    --master  local[2]
```

On Python

```
[root@saispark ~]# pyspark      --master    local[2]
```

4. Define a schema for the structure of the input data.

On Scala

```
scala> import org.apache.spark.sql.types._
scala> val activationsSchema = StructType( List(
StructField("acct_num", IntegerType),
StructField("dev_id", StringType),
StructField("phone", StringType),
StructField("model", StringType)))
```

On Python

```
pyspark> from pyspark.sql.types import *
pyspark> activationsSchema = StructType([
StructField("acct_num", IntegerType()),
StructField("dev_id", StringType()),
StructField("phone", StringType()),
StructField("model", StringType())])
```

5. Create a streaming DataFrame by reading the data you reviewed above.

On Scala

```
scala> val activationsDF =
spark.readStream.schema(activationsSchema).json("file:/t
mp/devsh-streaming/")
```

On Python

```
pyspark> activationsDF =
spark.readStream.schema(activationsSchema).json("file:/t
mp/devsh-streaming/")
```

6. Display the streaming DataFrame's schema to confirm that it is set up correctly.

   On Scala

   scala> **activationsDF.printSchema**

   On Python

   scala> **activationsDF.printSchema()**

7. Confirm that the DataFrame's isStreaming property is set. [Must return True]

   On Scala

   scala> **activationsDF.isStreaming**

   On Python

   scala> **activationsDF.isStreaming()**

8. Start a streaming query that displays results to the console. Use append mode to display the first several records in each new input stream micro-batch. Set the truncate option so that you will be able to see all the data in each record.

   On Scala

   scala> **val activationsQuery = activationsDF.writeStream.outputMode("append"). format("console").option("truncate","false").start**

   On Python

   pyspark> **activationsQuery = activationsDF.writeStream.outputMode("append"). format("console").option("truncate","false").start()**

   The query will not display any output yet, because no files are available to read yet.

9. Open a new terminal window. Run a test script to copy the test data files into the streaming directory at a rate of one per second.
[root@saispark ~]# **cd Desktop**

Type in continuous line giving space after streamtest-file.sh
[root@saispark Desktop]# **./streamtest-file.sh /root/Desktop/streamsdata/activations_stream/ /tmp/devsh-streaming/**

The script will display the names of the files as it copies them.

Note: Spark keeps track of files that have been previously read by each query. If you need to re-run the script later to test the same query, Spark will ignore any files that were previously copied.

10. Return to your Spark shell and confirm that the query is displaying console output displaying data from each batch. Note that the first batch processed and displayed will always be empty.

```
-------------------------------------------
Batch: 46
-------------------------------------------
+--------+------------------------------------+----------+--------------------+
|acct_num|dev_id                              |phone     |model               |
+--------+------------------------------------+----------+--------------------+
|29729   |68462601-1703-4cf0-b76b-81be9f9bc643|4247261326|Titanic 1000        |
|31807   |dbd6d2aa-3d89-4765-9d44-ac3c73027e19|7025440135|Sorrento F41L       |
|49004   |00b0c391-f7af-413e-8aa3-d4465a47d23e|7757169431|Sorrento F10L       |
|103891  |5c4d0cf8-270f-40f7-a9b5-273f2f594433|5107087452|Titanic 1000        |
|102577  |5cfed3e5-9615-43f6-899c-57036b1ecbc3|3107745454|Sorrento F20L       |
|4696    |ef427014-5b3a-4a09-b490-1ed0d87cf361|6196669339|Sorrento F41L       |
|28793   |f07235d9-43d8-4d50-8824-dcee47839185|7751121147|Sorrento F41L       |
|59074   |bd4dcf19-07a0-4f25-99ac-7cedb1cf718b|8185919378|iFruit 2            |
|95279   |3de4469f-961d-4ac7-a0b7-a25035e01483|2094012680|Sorrento F41L       |
|100308  |9ceb921d-6fe3-417d-8a7a-8a50af066b7f|7756923736|Sorrento F41L       |
|35200   |4f9bb4ac-1e2c-4a38-9e6b-8d0952da988d|2091173899|MeeToo 3.1          |
|94116   |3c34d4f8-4112-43bd-b819-4b14598196bf|5030103089|Sorrento F41L       |
|77934   |862e5929-ad0f-4ace-89d0-c79bf36fea79|5308415641|Ronin Novelty Note 2|
|1737    |b68598db-3102-4da1-90aa-9decbf661c59|6500219384|Sorrento F41L       |
|103147  |8accb504-3a6f-44ea-b2d2-91f1a30fc41c|9284758305|Sorrento F31L       |
|124066  |a98de1c2-7f28-4f41-8c62-9f4694b5517f|6264362235|Sorrento F41L       |
|107653  |94b63677-93f5-4640-a354-fdaba57f8b9e|9286779182|Sorrento F00L       |
|94771   |3e76847f-6119-4172-83a3-6f5b95c5f28c|6267279686|iFruit 5            |
|90652   |22d355fb-f7dd-49b8-b79c-2bd7b708d049|8185854632|Sorrento F00L       |
|28984   |20495444-b39b-49b1-9835-c6e630495235|7075238411|Sorrento F41L       |
+--------+------------------------------------+----------+--------------------+
only showing top 20 rows
```

11. When you are done, stop the stream by entering
activationsQuery.stop()

Note: You will not see a prompt while the shell is displaying output, but you can enter commands in the shell anyway.

Terminate the test script in the second terminal window using **Ctrl+C.**