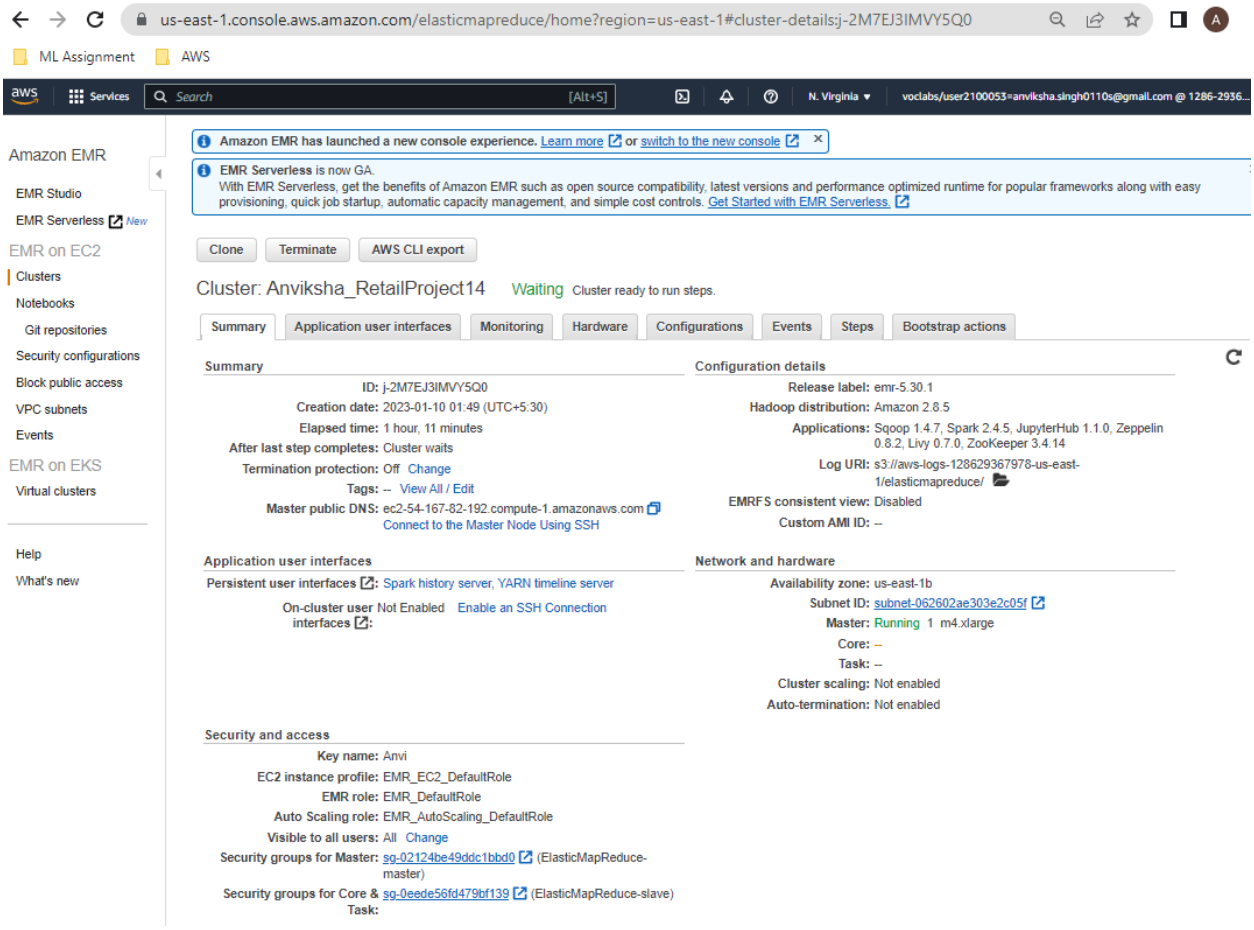
# RETAIL PROJECT


## By Anviksha Singh

Anviksha.singh0110s@gmail.com


## EMR Cluster Details

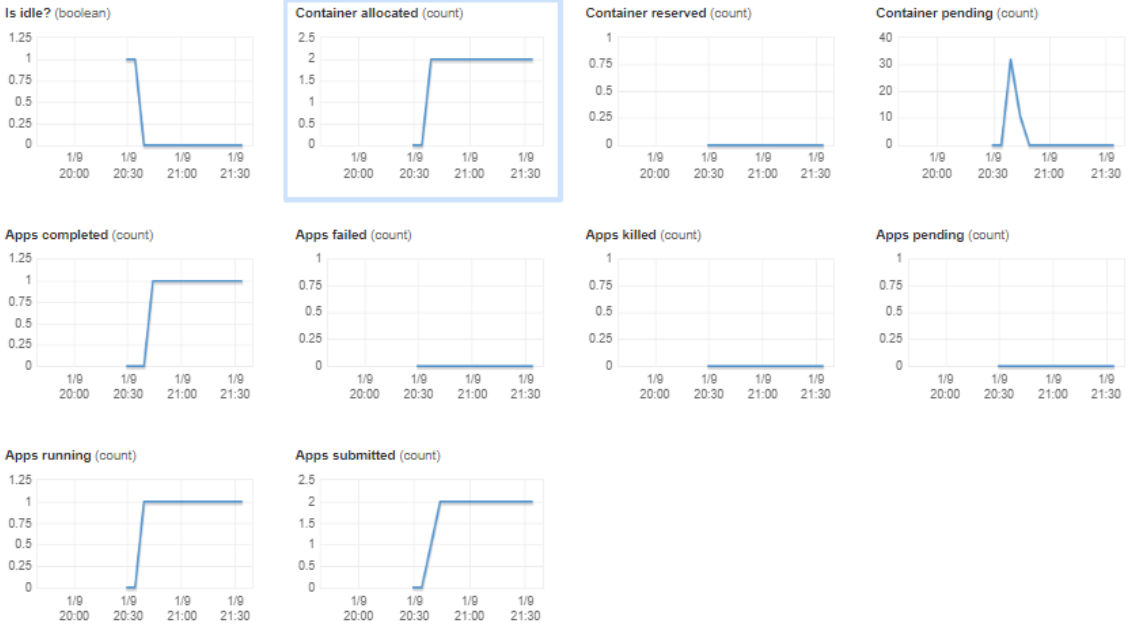


Cluster: Anviksha_RetailProject14 Waiting Cluster ready to run steps.

Summary | Application user interfaces | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Graph size: Large Start: 2 Hours Ago End: 0 Hours Ago Submit All graphs are displayed in the UTC time zone.

Cluster status | Node status | IO




## High-level application history

Amazon EMR collects information from YARN applications on your cluster and keeps a summary of historical information for seven days after applications have completed. Learn more

Effective January 23, 2023, the 7-day, high-level application history will be discontinued. Instead, connect to the Persistent Application user interfaces, which provide greater access to off-cluster application UIs and application logs for up to 30 days after the application ends. Learn more

**YARN applications (2)**

Filter: All applications | Filter applications ... | 2 applications (all loaded)

| Application ID | Type | Action | Status | Start time (UTC+5:30) | Duration | Finish time (UTC+5:30) | User |
|---|---|---|---|---|---|---|---|
| application_1673296070272_0002 | Spark | RetailByAnvi14 | Running | 2023-01-10 02:09 (UTC+5:30) | 50 min | | hadoop |

## EMR cluster launch via Putty and commands executed within it

**Force updating the cluster with all the packages and libraries**

```
[hadoop@ip-172-31-23-188 ~]$ sudo -i

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[root@ip-172-31-23-188 ~]# yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                | 3.7 kB     00:00
```

**Added the spark-streaming python file using WinSCP to Hadoop**

```
[hadoop@ip-172-31-23-188 ~]$ export SPARK_KAFKA_VERSION=0.10
[hadoop@ip-172-31-23-188 ~]$ ls
spark-streaming.py
[hadoop@ip-172-31-23-188 ~]$
```

## After this we have excuted python file to read the data from kafka topic shared by the upgrad using he below command

```
[hadoop@ip-172-31-23-188 ~]$ spark-submit --packages org.apache.spark:spark-sql-
kafka-0-10_2.11:2.4.5 spark-streaming.py 18.211.252.152 9092 real-time-project >
 Console-output.txt
```

**After the above command is executed we have received the console output in txt file and we have got seprate folders created for KPIs**

```
[hadoop@ip-172-31-23-188 ~]$ hadoop fs -ls
Found 5 items
drwxr-xr-x   - hadoop hadoop          0 2023-01-09 20:39 .sparkStaging
drwxr-xr-x   - hadoop hadoop          0 2023-01-09 21:13 TimeCountryKPIs
drwxr-xr-x   - hadoop hadoop          0 2023-01-09 21:13 TimeKPIs
```

## Snippets of the python code file

| | |
|---|---|
| **Import Required Libraries and Dependencies** | spark-streaming.py<br>1<br>2 # Import Required Lib and Dependencies<br>3 from pyspark.sql import SparkSession<br>4 from pyspark.sql.functions import *<br>5 from pyspark.sql.types import *<br>6 from pyspark.sql.functions import from_json<br>7 from pyspark.sql.window import Window |
| **Starting a new Spark Session** | spark-streaming.py<br>8<br>9 spark = SparkSession \<br>10 .builder \<br>11 .appName("RetailByAnvil4") \<br>12 .getOrCreate()<br>13 spark.sparkContext.setLogLevel('ERROR')<br>14 |
| **Reading the data from Kafka topic given by upgrad** | spark-streaming.py<br>14<br>15 # Read raw_input from kafka topic shared via Upgrad<br>16 df = spark \<br>17 .readStream \<br>18 .format("kafka") \<br>19 .option("kafka.bootstrap.servers","18.211.252.152:9092") \<br>20 .option("subscribe","real-time-project") \<br>21 .option("auto.offset.reset","earliest") \<br>22 .load() |
| **Creating the Schema for the attributes that are expected to retrieved** | # Define Schema to get the raw data in the required way<br>Schema = StructType() \<br>.add("invoice_no", LongType()) \<br>.add("country",StringType()) \<br>.add("timestamp", TimestampType()) \<br>.add("type", StringType()) \<br>.add("total_items",IntegerType())\<br>.add("isOrder",IntegerType()) \<br>.add("isReturn",IntegerType()) \<br>.add("items", ArrayType(StructType([<br>StructField("SKU", StringType()),<br>StructField("title", StringType()),<br>StructField("unit_price", FloatType()),<br>StructField("quantity", IntegerType())<br>])))) |

| | |
|---|---|
| **Adding the data in Raw Input using the select query** | `raw_input = df.select(from_json(col("value").cast("string"), Schem` |
| **Build Utility Functions – not all the functions are not here – please refer to the spark-streaming.py file for all the utility functions** | `# Build Utility functions for calculated attributes`<br><br>`def isAnOrder(type):`<br>`  if type=="ORDER":`<br>`  return 1`<br>`  else:`<br>`  return 0`<br><br>`def isAReturn(type):`<br>`  if type=="RETURN":`<br>`  return 1`<br>`  else:` |
| **Defining UDFS with Utility functions built in previous step** | `# Define the UDFs along with the utility functions`<br>`isOrder = udf(isAnOrder, IntegerType())`<br>`isReturn = udf(isAReturn, IntegerType())`<br>`total_item_count = udf(total_items, IntegerType())`<br>`total_cost = udf(total_items_cost, FloatType())`<br><br><br>`# Console Output`<br>`input = raw_input \`<br>`  .withColumn("total_items", total_item_count(r`<br>`  .withColumn("total_cost", total_cost(raw_inpu`<br>`  .withColumn("isOrder", isOrder(raw_input.type`<br>`  .withColumn("isReturn", isReturn(raw_input.ty` |
| **Calculate the required KPIs** | `# Calculate Time based KPIs`<br>`agg_time = input \`<br>`  .withWatermark("timestamp","1 minutes") \`<br>`  .groupby(window("timestamp", "1 minutes", "1 mi`<br>`  .agg(count("invoice_no").alias("OPM"),sum("tota`<br>`  avg("total_cost").alias("averageTransaction`<br>`  avg("isReturn").alias("rateOfReturn")) \`<br>`  .select("window.start","window.end","totalSales`<br><br>`# Calculate Time and Country based KPIs`<br>`agg_time_country = input \`<br>`  .withWatermark("timestamp", "1 minutes") \`<br>`  .groupBy(window("timestamp", "1 minutes", "1 mi`<br>`  .agg(count("invoice_no").alias("OPM"),sum("tota`<br>`  avg("isReturn").alias("rateOfReturn")) \`<br>`  .select("window.start","window.end","country", |

**Write the KPIs calculated in json format**

```
# Write Time based KPI values
time = agg_time.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "TimeKPIs/") \
    .option("checkpointLocation", "TimeKPIs/cp/") \
    .trigger(processingTime="1 minutes") \
    .start()


# Write Time and country based KPI values
time_country = agg_time_country.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "TimeCountryKPIs/") \
    .option("checkpointLocation", "TimeCountryKPIs
    .trigger(processingTime="1 minutes") \
    .start()

time_country.awaitTermination()
```

## Once the python file is executed we have the KPIs and Console

**KPI and Consol output goes to hadoop so we have to get it using the below command**

```
[hadoop@ip-172-31-23-188 ~]$ hadoop fs -get TimeKPIs/ ~/
[hadoop@ip-172-31-23-188 ~]$ hadoop fs -get TimeCountryKPIs/ ~/
[hadoop@ip-172-31-23-188 ~]$ ls
Console-output.txt  spark-streaming.py  TimeCountryKPIs  TimeKPIs
```

## In order to export these file we need to zip them for efficient migration to local system

```
[hadoop@ip-172-31-23-188 ~]$
[hadoop@ip-172-31-23-188 ~]$ zip -r output.zip ./TimeCountryKPIs/ ./TimeKPIs/
```