

RETAIL PROJECT

By Anviksha Singh

Anviksha.singh0110s@gmail.com

EMR Cluster Details

us-east-1.console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#cluster-detailsj-2M7EJ3IMVY5Q0

ML Assignment AWS

Amazon EMR

EMR Studio

EMR Serverless [New](#)

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on EKS

Virtual clusters

Help

What's new

Amazon EMR has launched a new console experience. [Learn more](#) or [switch to the new console](#)

EMR Serverless is now GA. With EMR Serverless, get the benefits of Amazon EMR such as open source compatibility, latest versions and performance optimized runtime for popular frameworks along with easy provisioning, quick job startup, automatic capacity management, and simple cost controls. [Get Started with EMR Serverless](#)

Clone Terminate AWS CLI export

Cluster: Anviksha_RetailProject14 **Waiting** Cluster ready to run steps.

Summary Application user interfaces Monitoring Hardware Configurations Events Steps Bootstrap actions

Summary

ID: j-2M7EJ3IMVY5Q0
Creation date: 2023-01-10 01:49 (UTC+5:30)
Elapsed time: 1 hour, 11 minutes
After last step completes: Cluster waits
Termination protection: Off [Change](#)
Tags: -- [View All](#) / [Edit](#)
Master public DNS: ec2-54-167-82-192.compute-1.amazonaws.com [Connect to the Master Node Using SSH](#)

Configuration details

Release label: emr-5.30.1
Hadoop distribution: Amazon 2.8.5
Applications: Sqoop 1.4.7, Spark 2.4.5, JupyterHub 1.1.0, Zeppelin 0.8.2, Livy 0.7.0, ZooKeeper 3.4.14
Log URI: s3://aws-logs-128629367978-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces

Persistent user interfaces [🔗](#): Spark history server, YARN timeline server
On-cluster user Not Enabled [Enable an SSH Connection](#)
interfaces [🔗](#):

Network and hardware

Availability zone: us-east-1b
Subnet ID: [subnet-062602ae303e2c05f](#)
Master: **Running** 1 m4.xlarge
Core: --
Task: --
Cluster scaling: Not enabled
Auto-termination: Not enabled

Security and access

Key name: Anvi
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Auto Scaling role: EMR_AutoScaling_DefaultRole
Visible to all users: All [Change](#)
Security groups for Master: [sg-02124be49ddc1bbd0](#) (ElasticMapReduce-master)
Security groups for Core & Task: [sg-0e0de56fd479bf139](#) (ElasticMapReduce-slave)

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

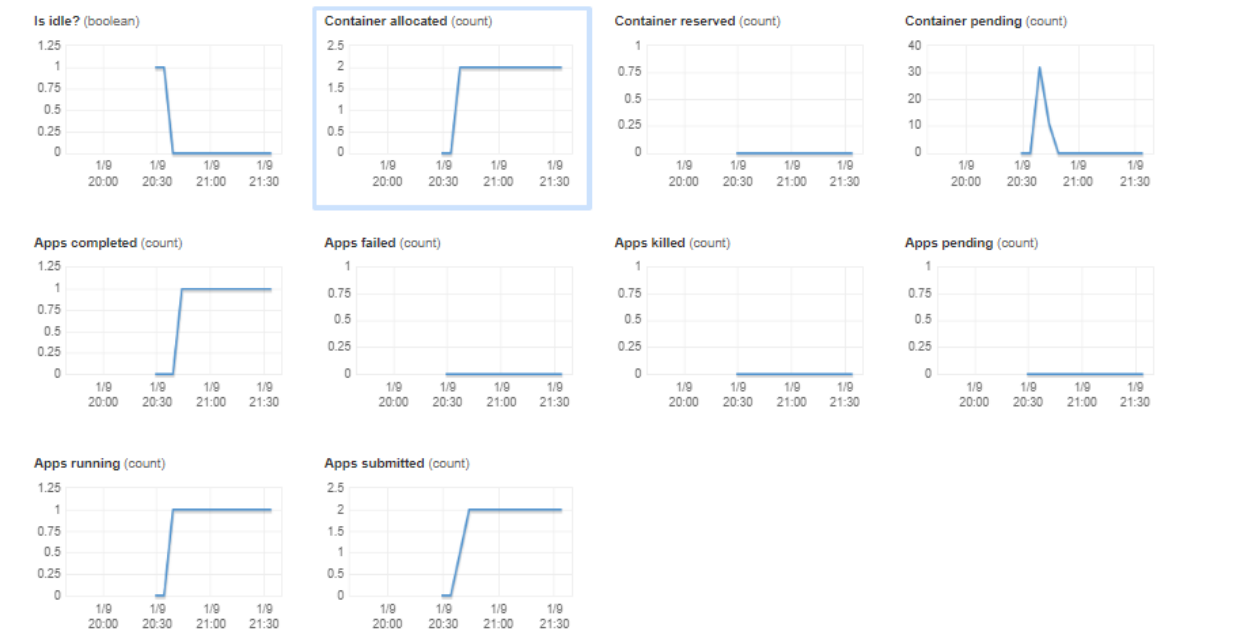
Submit

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 |
|---|-------|----------------|---------|-----------------------------|----------|------------------------|--------|
| <div>▶ application_1673296070272_0002</div> | 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

[illegible]

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 executed python file to read the data from kafka topic shared by the upgrad using the 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 x
1
2  # Import Required Lib and Dependencies
3  from pyspark.sql import SparkSession
4  from pyspark.sql.functions import *
5  from pyspark.sql.types import *
6  from pyspark.sql.functions import from_json
7  from pyspark.sql.window import Window
```

Starting a new Spark Session

```
spark-streaming.py x
8
9  spark = SparkSession \
10      .builder \
11      .appName("RetailByAnvil4") \
12      .getOrCreate()
13  spark.sparkContext.setLogLevel('ERROR')
14
```

Reading the data from Kafka topic given by upgrad

```
spark-streaming.py x
14
15  # Read raw_input from kafka topic shared via Upgrad
16  df = spark \
17      .readStream \
18      .format("kafka") \
19      .option("kafka.bootstrap.servers", "18.211.252.152:9092") \
20      .option("subscribe", "real-time-project") \
21      .option("auto.offset.reset", "earliest") \
22      .load()
23
```

Creating the Schema for the attributes that are expected to retrieved

```
# Define Schema to get the raw data in the required way
Schema = StructType() \
    .add("invoice_no", LongType()) \
    .add("country", StringType()) \
    .add("timestamp", TimestampType()) \
    .add("type", StringType()) \
    .add("total_items", IntegerType()) \
    .add("isOrder", IntegerType()) \
    .add("isReturn", IntegerType()) \
    .add("items", ArrayType(StructType([
        StructField("SKU", StringType()),
        StructField("title", StringType()),
        StructField("unit_price", FloatType()),
        StructField("quantity", IntegerType())
    ])))
```

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

def isAnOrder(type):
    if type=="ORDER":
        return 1
    else:
        return 0

def isAReturn(type):
    if type=="RETURN":
        return 1
    else:
```

Defining UDFS with Utility functions built in previous step

```
# Define the UDFs along with the utility functions
isOrder = udf(isAnOrder, IntegerType())
isReturn = udf(isAReturn, IntegerType())
total_item_count = udf(total_items, IntegerType())
total_cost = udf(total_items_cost, FloatType())

# Console Output
input = raw_input \
    .withColumn("total_items", total_item_count(r
    .withColumn("total_cost", total_cost(raw_input
    .withColumn("isOrder", isOrder(raw_input.type
    .withColumn("isReturn", isReturn(raw_input.ty
```

Calculate the required KPIs

```
# Calculate Time based KPIs
agg_time = input \
    .withWatermark("timestamp", "1 minutes") \
    .groupBy(window("timestamp", "1 minutes", "1 mi
    .agg(count("invoice_no").alias("OPM"),sum("tota
        avg("total_cost").alias("averageTransaction
        avg("isReturn").alias("rateOfReturn"))) \
    .select("window.start","window.end","totalSales

# Calculate Time and Country based KPIs
agg_time_country = input \
    .withWatermark("timestamp", "1 minutes") \
    .groupBy(window("timestamp", "1 minutes", "1 mi
    .agg(count("invoice_no").alias("OPM"),sum("tota
        avg("isReturn").alias("rateOfReturn"))) \
    .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/cp/") \
    .trigger(processingTime="1 minutes") \
    .start()

time_country.awaitTermination()
```

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

KPI and Console 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/
```