Comprehensive Data Pipeline Solutions Using Spark, Kafka, and AWS

Ronak Sengupta

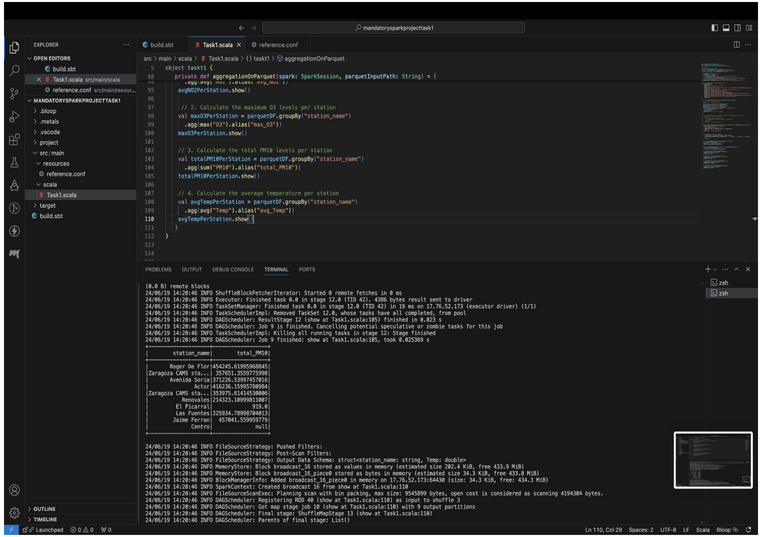
Senior Software Engineer 2

Payoda Technologies

Scenario 1 - Data Pipeline for performing Data conversions and performing aggregation operations

Description: In this project, I designed a data pipeline to analyze air quality data by seamlessly reading from AWS S3, writing to AWS Keyspaces for storage, converting and storing as Parquet format in S3, and performing crucial aggregation operations, designed by Apache Spark for efficient data processing.

Screenshot of aggregate operations



mandatorysparkprojecttask1
build.sbt
Task1.scala
reference.conf
src > main > scala > Task1.scala > {} task1 > aggregationOnParquet
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Ln 110, Col 29

```scala
object task1 {
  private def aggregationOnParquet(spark: SparkSession, parquetInputPath: String) = {
    avgNO2PerStation.show()

     // 2. Calculate the maximum O3 levels per station
    val maxO3PerStation = parquetDF.groupBy("station_name")
      .agg(max("O3").alias("max_O3"))
    maxO3PerStation.show()

    // 3. Calculate the total PM10 levels per station
    val totalPM10PerStation = parquetDF.groupBy("station_name")
      .agg(sum("PM10").alias("total_PM10"))
    totalPM10PerStation.show()

    // 4. Calculate the average temperature per station
    val avgTempPerStation = parquetDF.groupBy("station_name")
      .agg(avg("Temp").alias("avg_Temp"))
    avgTempPerStation.show()
  }
}
```

```
(0.0 B) remote blocks
24/06/19 14:20:46 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
24/06/19 14:20:46 INFO Executor: Finished task 0.0 in stage 12.0 (TID 42). 4386 bytes result sent to driver
24/06/19 14:20:46 INFO TaskSetManager: Finished task 0.0 in stage 12.0 (TID 42) in 19 ms on 17.76.52.173 (executor driver) (1/1)
24/06/19 14:20:46 INFO TaskSchedulerImpl: Removed TaskSet 12.0, whose tasks have all completed, from pool
24/06/19 14:20:46 INFO DAGScheduler: ResultStage 12 (show at Task1.scala:105) finished in 0.023 s
24/06/19 14:20:46 INFO DAGScheduler: Job 9 is finished. Cancelling potential speculative or zombie tasks for this job
24/06/19 14:20:46 INFO TaskSchedulerImpl: Killing all running tasks in stage 12: Stage finished
24/06/19 14:20:46 INFO DAGScheduler: Job 9 finished: show at Task1.scala:105, took 0.025369 s
+------------------+-----------------+
|      station_name|       total_PM10|
+------------------+-----------------+
|     Roger De Flor|454245.61995968845|
|Zaragoza CAMS sta...| 357651.3559775998|
|     Avenida Soria|371226.53997457016|
|             Actur|416236.15995780984|
|Zaragoza CAMS sta...|353975.61414530006|
|         Renovales|214323.10999811007|
|        El Picarral|            919.0|
|       Las Fuentes|225934.78998704013|
|       Jaime Ferran| 457041.559959779|
|            Centro|             null|
+------------------+-----------------+

24/06/19 14:20:46 INFO FileSourceStrategy: Pushed Filters:
24/06/19 14:20:46 INFO FileSourceStrategy: Post-Scan Filters:
24/06/19 14:20:46 INFO FileSourceStrategy: Output Data Schema: struct<station_name: string, Temp: double>
24/06/19 14:20:46 INFO MemoryStore: Block broadcast_16 stored as values in memory (estimated size 202.4 KiB, free 433.9 MiB)
24/06/19 14:20:46 INFO MemoryStore: Block broadcast_16_piece0 stored as bytes in memory (estimated size 34.3 KiB, free 433.8 MiB)
24/06/19 14:20:46 INFO BlockManagerInfo: Added broadcast_16_piece0 in memory on 17.76.52.173:64430 (size: 34.3 KiB, free: 434.3 MiB)
24/06/19 14:20:46 INFO SparkContext: Created broadcast 16 from show at Task1.scala:110
24/06/19 14:20:46 INFO FileSourceScanExec: Planning scan with bin packing, max size: 9545899 bytes, open cost is considered as scanning 4194304 bytes.
24/06/19 14:20:46 INFO DAGScheduler: Registering RDD 40 (show at Task1.scala:110) as input to shuffle 3
24/06/19 14:20:46 INFO DAGScheduler: Got map stage job 10 (show at Task1.scala:110) with 9 output partitions
24/06/19 14:20:46 INFO DAGScheduler: Final stage: ShuffleMapStage 13 (show at Task1.scala:110)
24/06/19 14:20:46 INFO DAGScheduler: Parents of final stage: List()
```

keyspace table screenshot & DAG Output

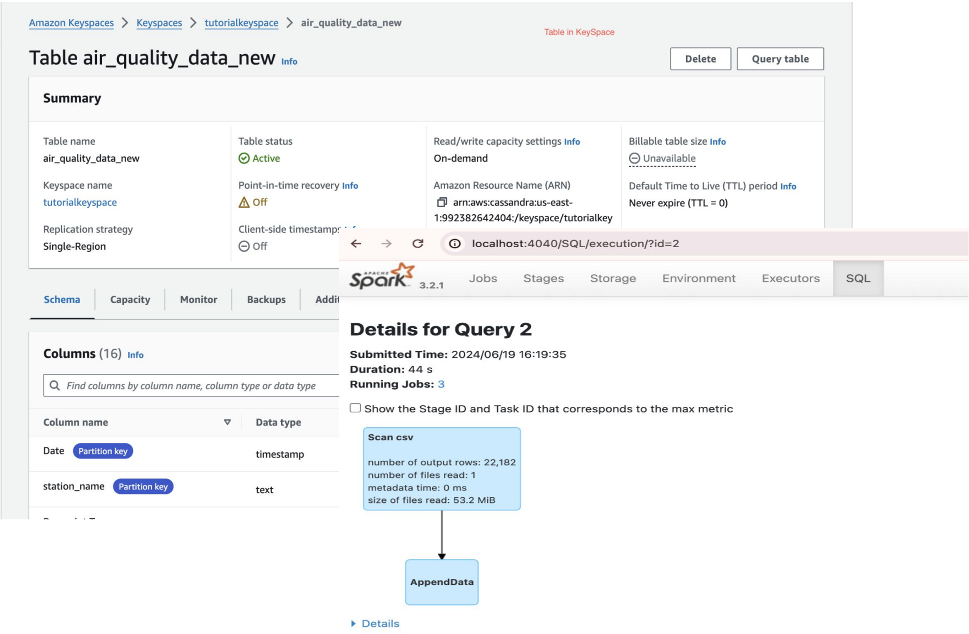


Table in KeySpace

Table air_quality_data_new Info

Summary

Table name: air_quality_data_new

Table status: Active

Read/write capacity settings Info: On-demand

Billable table size Info: Unavailable

Keyspace name: tutorialkeyspace

Point-in-time recovery Info: Off

Amazon Resource Name (ARN): arn:aws:cassandra:us-east-1:992382642404:/keyspace/tutorialkey

Default Time to Live (TTL) period Info: Never expire (TTL = 0)

Replication strategy: Single-Region

Client-side timestamps: Off

Columns (16) Info

Find columns by column name, column type or data type

Column name		Data type
Date	Partition key	timestamp
station_name	Partition key	text

localhost:4040/SQL/execution/?id=2

Details for Query 2

Submitted Time: 2024/06/19 16:19:35
Duration: 44 s
Running Jobs: 3

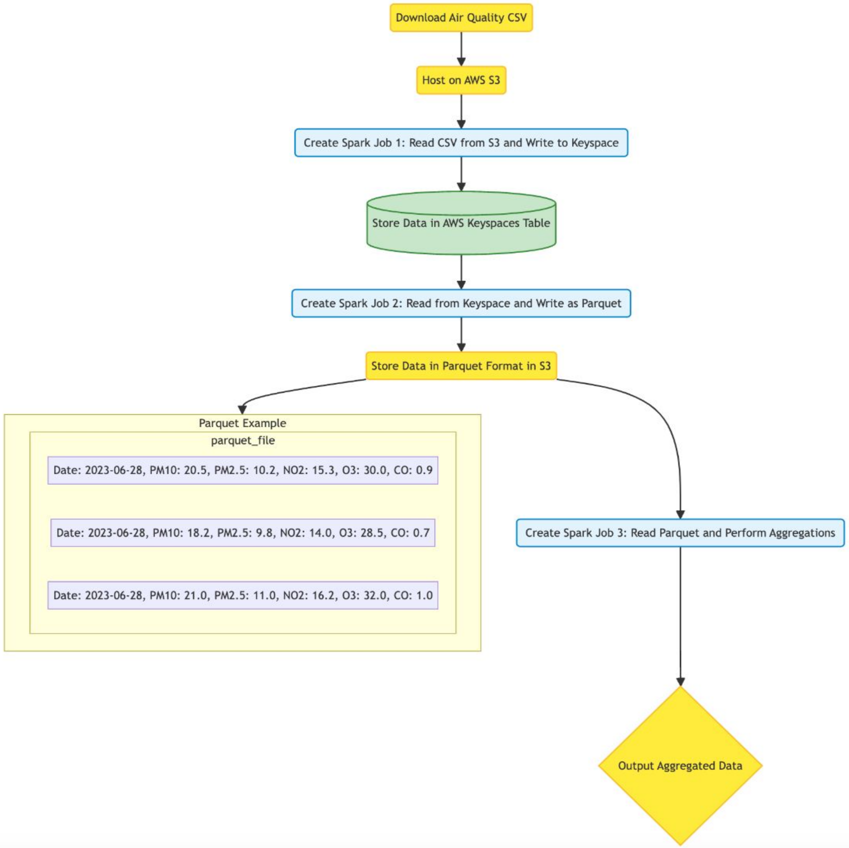
□ Show the Stage ID and Task ID that corresponds to the max metric

Scan csv

number of output rows: 22,182
number of files read: 1
metadata time: 0 ms
size of files read: 53.2 MiB

AppendData

Flow Diagram - Scenario 1:



Download Air Quality CSV

Host on AWS S3

Create Spark Job 1: Read CSV from S3 and Write to Keyspace

Store Data in AWS Keyspaces Table

Create Spark Job 2: Read from Keyspace and Write as Parquet

Store Data in Parquet Format in S3

Parquet Example

parquet_file

Date: 2023-06-28, PM10: 20.5, PM2.5: 10.2, NO2: 15.3, O3: 30.0, CO: 0.9

Date: 2023-06-28, PM10: 18.2, PM2.5: 9.8, NO2: 14.0, O3: 28.5, CO: 0.7

Date: 2023-06-28, PM10: 21.0, PM2.5: 11.0, NO2: 16.2, O3: 32.0, CO: 1.0

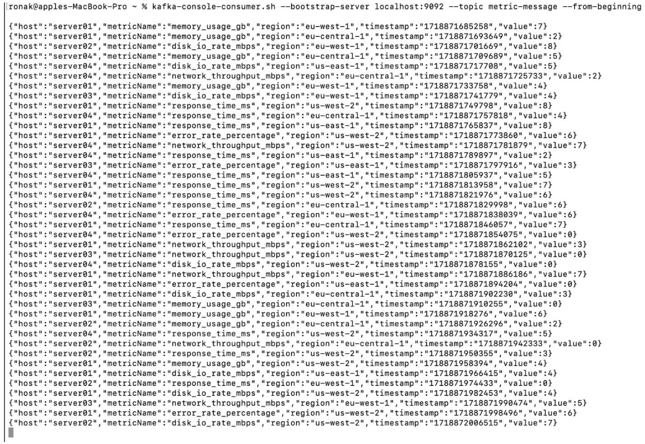
Create Spark Job 3: Read Parquet and Perform Aggregations

Output Aggregated Data

Scenario 2 - Streaming JOB using Kafka , Spark and Protobuf

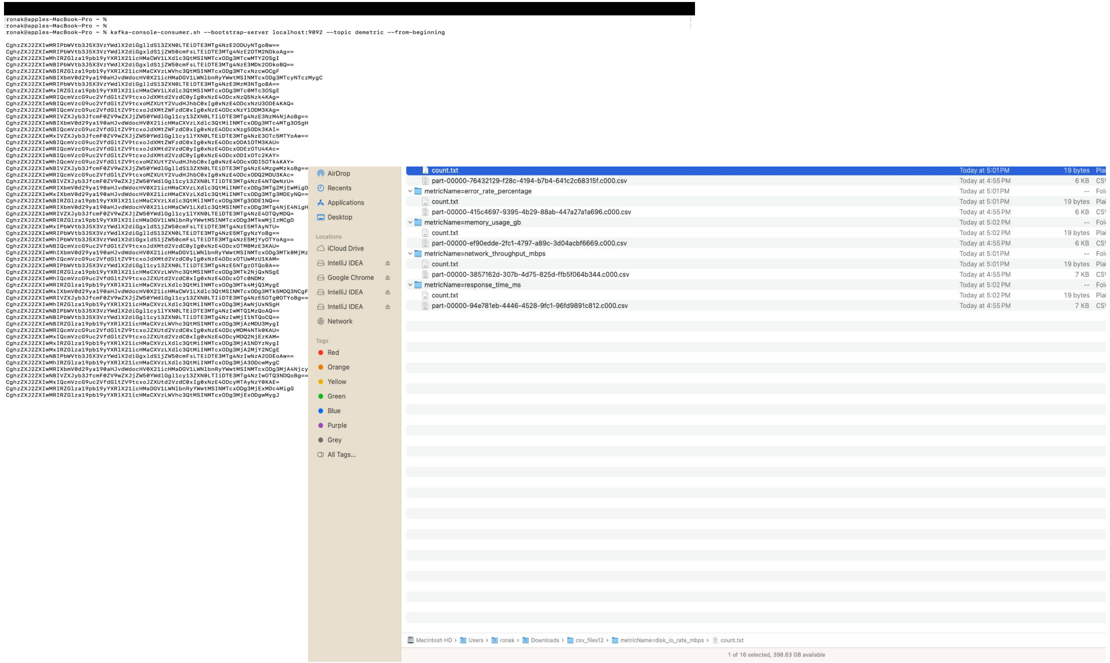
Description: In Scenario 2, a streaming pipeline using AKKA, Apache Spark, Protobuf, Spark Streaming, and Apache Kafka was implemented. An AKKA microservice generates JSON server metrics data posted to Kafka every 5 seconds. Spark Streaming converts JSON to Protobuf and publishes to another Kafka topic, then deserializes Protobuf to CSV files based on metric type. Spark Jobs aggregate these CSV files for comprehensive analysis, leveraging Spark and Kafka for scalable, fault-tolerant data processing.

Screenshot of akka kafka topic



```
ronak@apples-MacBook-Pro ~ % kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic metric-message --from-beginning

{"host":"server01","metricName":"memory_usage_gb","region":"eu-west-1","timestamp":"1718871685258","value":7}
{"host":"server01","metricName":"memory_usage_gb","region":"eu-central-1","timestamp":"1718871693649","value":2}
{"host":"server02","metricName":"disk_io_rate_mbps","region":"eu-west-1","timestamp":"1718871701669","value":8}
{"host":"server04","metricName":"memory_usage_gb","region":"eu-central-1","timestamp":"1718871709689","value":5}
{"host":"server04","metricName":"disk_io_rate_mbps","region":"us-east-1","timestamp":"1718871717708","value":5}
{"host":"server04","metricName":"network_throughput_mbps","region":"eu-central-1","timestamp":"1718871725733","value":2}
{"host":"server01","metricName":"memory_usage_gb","region":"eu-west-1","timestamp":"1718871733758","value":4}
{"host":"server03","metricName":"disk_io_rate_mbps","region":"eu-west-1","timestamp":"1718871741779","value":4}
{"host":"server01","metricName":"response_time_ms","region":"us-west-2","timestamp":"1718871749798","value":8}
{"host":"server04","metricName":"response_time_ms","region":"eu-central-1","timestamp":"1718871757818","value":4}
{"host":"server01","metricName":"response_time_ms","region":"us-east-1","timestamp":"1718871765837","value":8}
{"host":"server01","metricName":"error_rate_percentage","region":"us-west-2","timestamp":"1718871773860","value":6}
{"host":"server04","metricName":"network_throughput_mbps","region":"us-west-2","timestamp":"1718871781879","value":7}
{"host":"server04","metricName":"response_time_ms","region":"us-east-1","timestamp":"1718871789897","value":2}
{"host":"server03","metricName":"error_rate_percentage","region":"us-east-1","timestamp":"1718871797916","value":3}
{"host":"server04","metricName":"response_time_ms","region":"us-east-1","timestamp":"1718871805937","value":5}
{"host":"server01","metricName":"response_time_ms","region":"us-west-2","timestamp":"1718871813958","value":7}
{"host":"server04","metricName":"response_time_ms","region":"us-west-2","timestamp":"1718871821976","value":6}
{"host":"server02","metricName":"response_time_ms","region":"eu-central-1","timestamp":"1718871829998","value":6}
{"host":"server04","metricName":"error_rate_percentage","region":"eu-west-1","timestamp":"1718871838039","value":6}
{"host":"server01","metricName":"response_time_ms","region":"eu-central-1","timestamp":"1718871846057","value":7}
{"host":"server04","metricName":"error_rate_percentage","region":"us-west-2","timestamp":"1718871854075","value":0}
{"host":"server01","metricName":"network_throughput_mbps","region":"us-west-2","timestamp":"1718871862102","value":3}
{"host":"server03","metricName":"network_throughput_mbps","region":"us-west-2","timestamp":"1718871870125","value":0}
{"host":"server04","metricName":"disk_io_rate_mbps","region":"us-west-2","timestamp":"1718871878155","value":0}
{"host":"server01","metricName":"network_throughput_mbps","region":"eu-west-1","timestamp":"1718871886186","value":7}
{"host":"server01","metricName":"error_rate_percentage","region":"us-east-1","timestamp":"1718871894204","value":0}
{"host":"server01","metricName":"disk_io_rate_mbps","region":"eu-central-1","timestamp":"1718871902230","value":3}
{"host":"server03","metricName":"memory_usage_gb","region":"eu-central-1","timestamp":"1718871910255","value":0}
{"host":"server01","metricName":"memory_usage_gb","region":"eu-west-1","timestamp":"1718871918276","value":6}
{"host":"server02","metricName":"memory_usage_gb","region":"eu-central-1","timestamp":"1718871926296","value":2}
{"host":"server04","metricName":"response_time_ms","region":"us-west-2","timestamp":"1718871934317","value":5}
{"host":"server02","metricName":"network_throughput_mbps","region":"eu-central-1","timestamp":"1718871942333","value":0}
{"host":"server02","metricName":"response_time_ms","region":"us-west-2","timestamp":"1718871950355","value":3}
{"host":"server01","metricName":"memory_usage_gb","region":"us-west-2","timestamp":"1718871958394","value":4}
{"host":"server01","metricName":"disk_io_rate_mbps","region":"us-east-1","timestamp":"1718871966415","value":4}
{"host":"server02","metricName":"response_time_ms","region":"eu-west-1","timestamp":"1718871974433","value":0}
{"host":"server01","metricName":"disk_io_rate_mbps","region":"us-west-2","timestamp":"1718871982453","value":4}
{"host":"server03","metricName":"network_throughput_mbps","region":"eu-west-1","timestamp":"1718871990474","value":5}
{"host":"server01","metricName":"error_rate_percentage","region":"us-west-2","timestamp":"1718871998496","value":6}
{"host":"server02","metricName":"disk_io_rate_mbps","region":"us-west-2","timestamp":"1718872006515","value":7}
```

Screenshot of binary data(protobuf format) & output csv screenshots



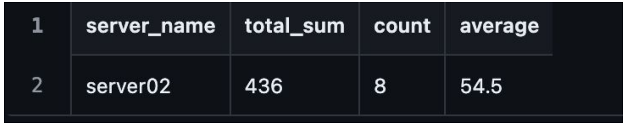
[redacted]

```
ronak@apples-MacBook-Pro ~ %
ronak@apples-MacBook-Pro ~ %
ronak@apples-MacBook-Pro ~ % kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic demetric --from-beginning
```

```
CghzZXJ2ZXIwMRIPbWVtb3J5X3VzYWdlX2diGglldS13ZXN0LTEiDTE3MTg4NzE2ODUyNTgoBw==
CghzZXJ2ZXIwMRIPbWVtb3J5X3VzYWdlX2diGgxldS1jZW50cmFsLTEiDTE3MTg4NzE2OTM2NDkoAg==
CghzZXJ2ZXIwMhIRZGlza19pb19yYXRlX21icHMaCWV1LXdlc3QtMSINMTcxODg3MTcwMTY2OSgI
CghzZXJ2ZXIwNBIPbWVtb3J5X3VzYWdlX2diGgxldS1jZW50cmFsLTEiDTE3MTg4NzE3MDk2ODkoBQ==
CghzZXJ2ZXIwNBIRZGlza19pb19yYXRlX21icHMaCXVzLWVhc3QtMSINMTcxODg3MTcxNzcwOCgF
CghzZXJ2ZXIwNBIXbmV0d29ya190aHJvdWdocHV0X21icHMaDGV1LWNlbnRyYWwtMSINMTcxODg3MTcyNTczMygC
CghzZXJ2ZXIwMRIPbWVtb3J5X3VzYWdlX2diGglldS13ZXN0LTEiDTE3MTg4NzE3MzM3NTgoBA==
CghzZXJ2ZXIwMxIRZGlza19pb19yYXRlX21icHMaCWV1LXdlc3QtMSINMTcxODg3MTc0MTc3OSgE
CghzZXJ2ZXIwMRIQcmVzcG9uc2VfdGltZV9tcxoJdXMtd2VzdC0yIg0xNzE4ODcxNzQ5Nzk4KAg=
CghzZXJ2ZXIwNBIQcmVzcG9uc2VfdGltZV9tcxoMZXUtY2VudHJhbC0xIg0xNzE4ODcxNzU3ODE4KAQ=
CghzZXJ2ZXIwMRIQcmVzcG9uc2VfdGltZV9tcxoJdXMtZWFzdC0xIg0xNzE4ODcxNzY1ODM3KAg=
CghzZXJ2ZXIwMRIVZXJyb3JfcmF0ZV9wZXJjZW50YWdlGgl1cy13ZXN0LTIiDTE3MTg4NzE3NzM4NjAoBg==
CghzZXJ2ZXIwNBIXbmV0d29ya190aHJvdWdocHV0X21icHMaCXVzLXdlc3QtMiINMTcxODg3MTc4MTg3OSgH
CghzZXJ2ZXIwNBIQcmVzcG9uc2VfdGltZV9tcxoJdXMtZWFzdC0xIg0xNzE4ODcxNzg5ODk3KAI=
CghzZXJ2ZXIwMxIVZXJyb3JfcmF0ZV9wZXJjZW50YWdlGgl1cy1lYXN0LTEiDTE3MTg4NzE3OTc5MTYoAw==
CghzZXJ2ZXIwNBIQcmVzcG9uc2VfdGltZV9tcxoJdXMtZWFzdC0xIg0xNzE4ODcxODA1OTM3KAU=
CghzZXJ2ZXIwMRIQcmVzcG9uc2VfdGltZV9tcxoJdXMtd2VzdC0yIg0xNzE4ODcxODEzOTU4KAc=
CghzZXJ2ZXIwNBIQcmVzcG9uc2VfdGltZV9tcxoJdXMtd2VzdC0yIg0xNzE4ODcxODIxOTc2KAY=
CghzZXJ2ZXIwMhIQcmVzcG9uc2VfdGltZV9tcxoMZXUtY2VudHJhbC0xIg0xNzE4ODcxODI5OTk4KAY=
CghzZXJ2ZXIwNBIVZXJyb3JfcmF0ZV9wZXJjZW50YWdlGglldS13ZXN0LTEiDTE3MTg4NzE4MzgwMzkoBg==
CghzZXJ2ZXIwMRIQcmVzcG9uc2VfdGltZV9tcxoMZXUtY2VudHJhbC0xIg0xNzE4ODcxODQ2MDU3KAc=
CghzZXJ2ZXIwNBIVZXJyb3JfcmF0ZV9wZXJjZW50YWdlGgl1cy13ZXN0LTIiDTE3MTg4NzE4NTQwNzU=
CghzZXJ2ZXIwMRIXbmV0d29ya190aHJvdWdocHV0X21icHMaCXVzLXdlc3QtMiINMTcxODg3MTg2MjEwMigD
CghzZXJ2ZXIwMxIXbmV0d29ya190aHJvdWdocHV0X21icHMaCXVzLXdlc3QtMiINMTcxODg3MTg3MDEyNQ==
CghzZXJ2ZXIwNBIRZGlza19pb19yYXRlX21icHMaCXVzLXdlc3QtMiINMTcxODg3MTg3ODE1NQ==
CghzZXJ2ZXIwMRIXbmV0d29ya190aHJvdWdocHV0X21icHMaCWV1LXdlc3QtMSINMTcxODg3MTg4NjE4NigH
CghzZXJ2ZXIwMRIVZXJyb3JfcmF0ZV9wZXJjZW50YWdlGgl1cy1lYXN0LTEiDTE3MTg4NzE4OTQyMDQ=
CghzZXJ2ZXIwMRIRZGlza19pb19yYXRlX21icHMaDGV1LWNlbnRyYWwtMSINMTcxODg3MTkwMjIzMCgD
CghzZXJ2ZXIwMxIPbWVtb3J5X3VzYWdlX2diGgxldS1jZW50cmFsLTEiDTE3MTg4NzE5MTAyNTU=
CghzZXJ2ZXIwMRIPbWVtb3J5X3VzYWdlX2diGglldS13ZXN0LTEiDTE3MTg4NzE5MTgyNzYoBg==
CghzZXJ2ZXIwMhIPbWVtb3J5X3VzYWdlX2diGgxldS1jZW50cmFsLTEiDTE3MTg4NzE5MjYyOTYoAg==
CghzZXJ2ZXIwNBIQcmVzcG9uc2VfdGltZV9tcxoJdXMtd2VzdC0yIg0xNzE4ODcxOTM0MzE3KAU=
CghzZXJ2ZXIwMhIXbmV0d29ya190aHJvdWdocHV0X21icHMaDGV1LWNlbnRyYWwtMSINMTcxODg3MTk0MjMz
CghzZXJ2ZXIwMhIQcmVzcG9uc2VfdGltZV9tcxoJdXMtd2VzdC0yIg0xNzE4ODcxOTUwMzU1KAM=
CghzZXJ2ZXIwMRIPbWVtb3J5X3VzYWdlX2diGgl1cy13ZXN0LTIiDTE3MTg4NzE5NTgzOTQoBA==
CghzZXJ2ZXIwMRIRZGlza19pb19yYXRlX21icHMaCXVzLWVhc3QtMSINMTcxODg3MTk2NjQxNSgE
CghzZXJ2ZXIwMhIQcmVzcG9uc2VfdGltZV9tcxoJZXUtd2VzdC0xIg0xNzE4ODcxOTc0NDMz
CghzZXJ2ZXIwMRIRZGlza19pb19yYXRlX21icHMaCXVzLXdlc3QtMiINMTcxODg3MTk4MjQ1MygE
CghzZXJ2ZXIwMxIXbmV0d29ya190aHJvdWdocHV0X21icHMaCWV1LXdlc3QtMSINMTcxODg3MTk5MDQ3NCgF
CghzZXJ2ZXIwMRIVZXJyb3JfcmF0ZV9wZXJjZW50YWdlGgl1cy13ZXN0LTIiDTE3MTg4NzE5OTg0OTYoBg==
CghzZXJ2ZXIwMhIRZGlza19pb19yYXRlX21icHMaCXVzLXdlc3QtMiINMTcxODg3MjAwNjUxNSgH
CghzZXJ2ZXIwNBIPbWVtb3J5X3VzYWdlX2diGgl1cy1lYXN0LTEiDTE3MTg4NzIwMTQ1MzQoAQ==
CghzZXJ2ZXIwNBIPbWVtb3J5X3VzYWdlX2diGgl1cy13ZXN0LTIiDTE3MTg4NzIwMjI1NTQoCQ==
CghzZXJ2ZXIwNBIRZGlza19pb19yYXRlX21icHMaCXVzLWVhc3QtMSINMTcxODg3MjAzMDU3MygI
CghzZXJ2ZXIwMRIQcmVzcG9uc2VfdGltZV9tcxoJZXUtd2VzdC0xIg0xNzE4ODcyMDM4NTk0KAU=
CghzZXJ2ZXIwMxIQcmVzcG9uc2VfdGltZV9tcxoJZXUtd2VzdC0xIg0xNzE4ODcyMDQ2NjEzKAM=
CghzZXJ2ZXIwMxIRZGlza19pb19yYXRlX21icHMaCXVzLXdlc3QtMiINMTcxODg3MjA1NDYzNygI
CghzZXJ2ZXIwMxIRZGlza19pb19yYXRlX21icHMaCXVzLXdlc3QtMiINMTcxODg3MjA2MjY2NCgE
CghzZXJ2ZXIwNBIPbWVtb3J5X3VzYWdlX2diGgxldS1jZW50cmFsLTEiDTE3MTg4NzIwNzA2ODEoAw==
CghzZXJ2ZXIwMhIRZGlza19pb19yYXRlX21icHMaCXVzLXdlc3QtMiINMTcxODg3MjA3ODcwMygC
CghzZXJ2ZXIwMRIXbmV0d29ya190aHJvdWdocHV0X21icHMaDGV1LWNlbnRyYWwtMSINMTcxODg3MjA4Njcy
CghzZXJ2ZXIwNBIVZXJyb3JfcmF0ZV9wZXJjZW50YWdlGgl1cy13ZXN0LTIiDTE3MTg4NzIwOTQ3NDQoBg==
CghzZXJ2ZXIwMxIQcmVzcG9uc2VfdGltZV9tcxoJZXUtd2VzdC0xIg0xNzE4ODcyMTAyNzY0KAE=
CghzZXJ2ZXIwMRIRZGlza19pb19yYXRlX21icHMaDGV1LWNlbnRyYWwtMSINMTcxODg3MjExMDc4MigG
CghzZXJ2ZXIwMRIRZGlza19pb19yYXRlX21icHMaCXVzLWVhc3QtMSINMTcxODg3MjExODgwMygJ
```

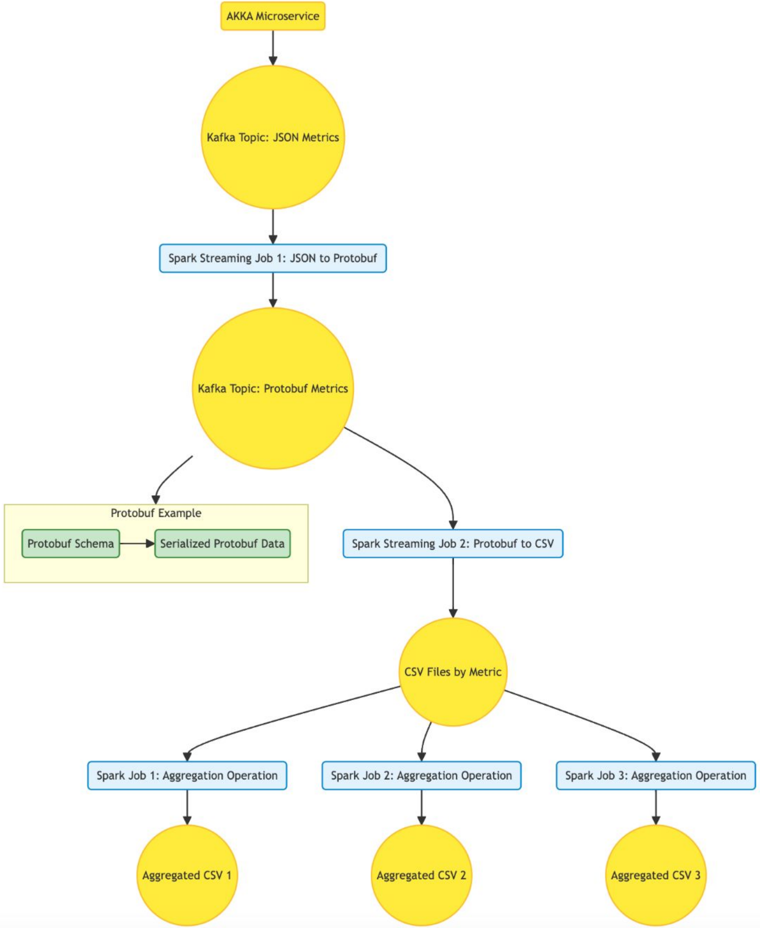
```
    count.txt                                                  Today at 5:01 PM   19 bytes  Pla
    part-00000-76432129-f28c-4194-b7b4-641c2c68315f.c000.csv   Today at 4:55 PM       6 KB  CS
  ⌄ metricName=error_rate_percentage                           Today at 5:01 PM         --  Fol
    count.txt                                                  Today at 5:01 PM   19 bytes  Plai
    part-00000-415c4697-9395-4b29-88ab-447a27a1a696.c000.csv   Today at 4:55 PM       6 KB  CS
  ⌄ metricName=memory_usage_gb                                 Today at 5:02 PM         --  Fol
    count.txt                                                  Today at 5:02 PM   19 bytes  Plai
    part-00000-ef90edde-2fc1-4797-a89c-3d04acbf6669.c000.csv   Today at 4:55 PM       6 KB  CS
  ⌄ metricName=network_throughput_mbps                         Today at 5:01 PM         --  Fol
    count.txt                                                  Today at 5:01 PM   19 bytes  Plai
    part-00000-3857162d-307b-4d75-825d-ffb5f064b344.c000.csv   Today at 4:55 PM       7 KB  CS
  ⌄ metricName=response_time_ms                                Today at 5:02 PM         --  Fol
    count.txt                                                  Today at 5:02 PM   19 bytes  Plai
    part-00000-94e781eb-4446-4528-9fc1-96fd9891c812.c000.csv   Today at 4:55 PM       7 KB  CS

Macintosh HD › Users › ronak › Downloads › csv_files12 › metricName=disk_io_rate_mbps › count.txt

1 of 16 selected, 398.63 GB available
```

Aggregate Output



1	server_name	total_sum	count	average
2	server02	436	8	54.5

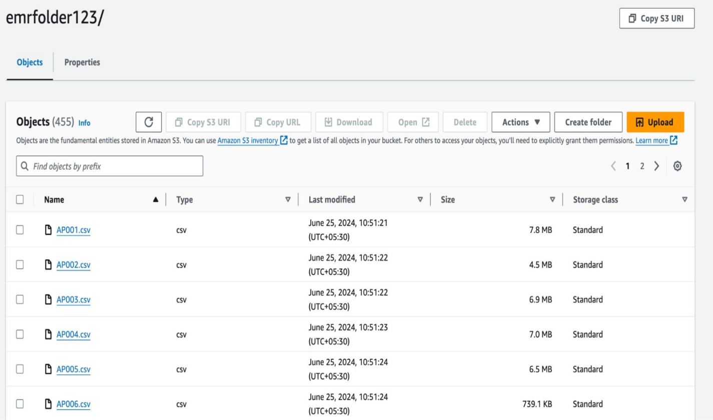
Flow Diagram - Scenario 2:


AKKA Microservice
Kafka Topic: JSON Metrics
Spark Streaming Job 1: JSON to Protobuf
Kafka Topic: Protobuf Metrics
Protobuf Example
Protobuf Schema
Serialized Protobuf Data
Spark Streaming Job 2: Protobuf to CSV
CSV Files by Metric
Spark Job 1: Aggregation Operation
Spark Job 2: Aggregation Operation
Spark Job 3: Aggregation Operation
Aggregated CSV 1
Aggregated CSV 2
Aggregated CSV 3

Scenario 3 - Executing Jobs on EMR

Description: - Scenario 3 utilizes AWS EMR, RDBMS, S3 Bucket, Spark, and Scala for data processing. It begins with setting up an EMR Cluster and Studio with a notebook. The dataset from Kaggle is stored in an S3 bucket, processed using an EMR Notebook to convert files to Parquet format and store them in another S3 bucket. Ten Spark Jobs on EMR perform operations like filtering and aggregation on the Parquet data, with results stored in MySQL tables for structured analysis and management.

Screenshot of S3 bucket:



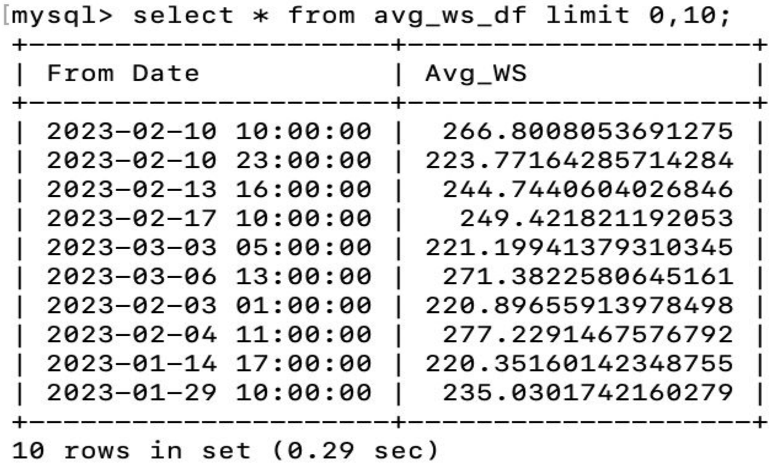
emrfolder123/

Objects (455)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

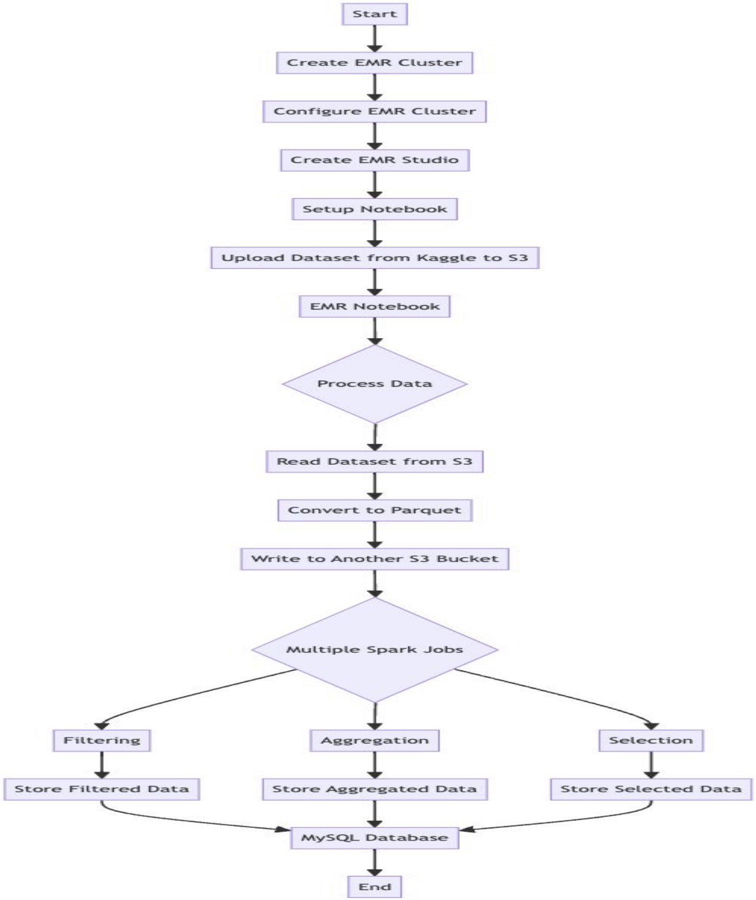
☐	Name ▲	Type	Last modified	Size	Storage class
☐	AP001.csv	csv	June 25, 2024, 10:51:21 (UTC+05:30)	7.8 MB	Standard
☐	AP002.csv	csv	June 25, 2024, 10:51:22 (UTC+05:30)	4.5 MB	Standard
☐	AP003.csv	csv	June 25, 2024, 10:51:22 (UTC+05:30)	6.9 MB	Standard
☐	AP004.csv	csv	June 25, 2024, 10:51:23 (UTC+05:30)	7.0 MB	Standard
☐	AP005.csv	csv	June 25, 2024, 10:51:24 (UTC+05:30)	6.5 MB	Standard
☐	AP006.csv	csv	June 25, 2024, 10:51:24 (UTC+05:30)	739.1 KB	Standard

Spark Job filtration screenshot



```
mysql> select * from avg_ws_df limit 0,10;
+---------------------+--------------------+
| From Date           | Avg_WS             |
+---------------------+--------------------+
| 2023-02-10 10:00:00 |  266.8008053691275 |
| 2023-02-10 23:00:00 | 223.77164285714284 |
| 2023-02-13 16:00:00 |  244.7440604026846 |
| 2023-02-17 10:00:00 |   249.421821192053 |
| 2023-03-03 05:00:00 | 221.19941379310345 |
| 2023-03-06 13:00:00 |  271.3822580645161 |
| 2023-02-03 01:00:00 | 220.89655913978498 |
| 2023-02-04 11:00:00 |  277.2291467576792 |
| 2023-01-14 17:00:00 | 220.35160142348755 |
| 2023-01-29 10:00:00 |  235.0301742160279 |
+---------------------+--------------------+
10 rows in set (0.29 sec)
```

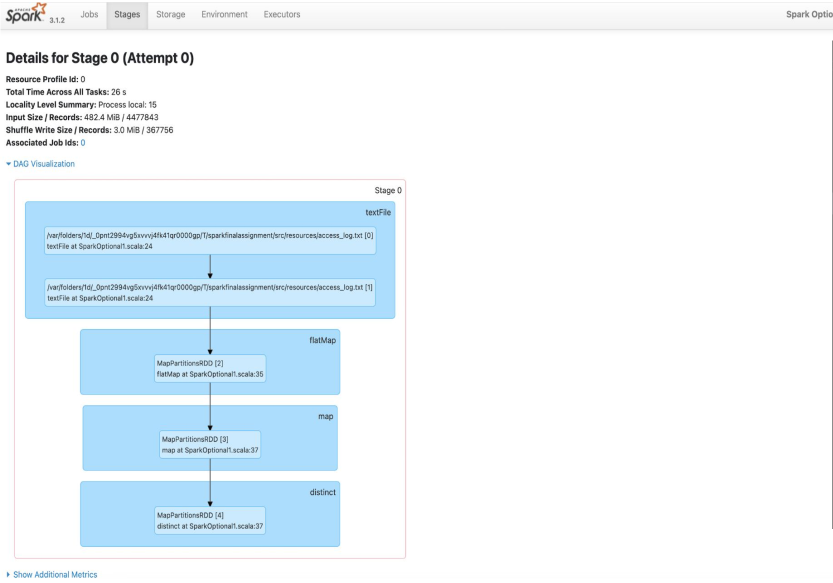
Flow Diagram of Scenario - 3



Optional Scenario 1

Description: Optional Scenario 1 uses RDDs in Apache Spark to process access log data from Kaggle in text format. Steps include reading the log file, converting it to RDDs, and extracting IP addresses, URLs, response statuses, and bytes sent. Key operations include finding unique IPs, grouping URLs by 200 status, counting 4xx responses, identifying requests > 5000 bytes, determining URLs with the most requests, and finding URLs with the most 404 errors. These operations offer insights into traffic patterns, response statuses, and error occurrences in the log data.

Screenshot of DAG output



Details for Stage 0 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 26 s
Locality Level Summary: Process local: 15
Input Size / Records: 482.4 MiB / 4477843
Shuffle Write Size / Records: 3.0 MiB / 367756
Associated Job Ids: 0

▾ DAG Visualization

Stage 0

textFile
/var/folders/1d/_0pnt2994vg5xvvvj4fk41qr0000gp/T/sparkfinalassignment/src/resources/access_log.txt [0]
textFile at SparkOptional1.scala:24

/var/folders/1d/_0pnt2994vg5xvvvj4fk41qr0000gp/T/sparkfinalassignment/src/resources/access_log.txt [1]
textFile at SparkOptional1.scala:24

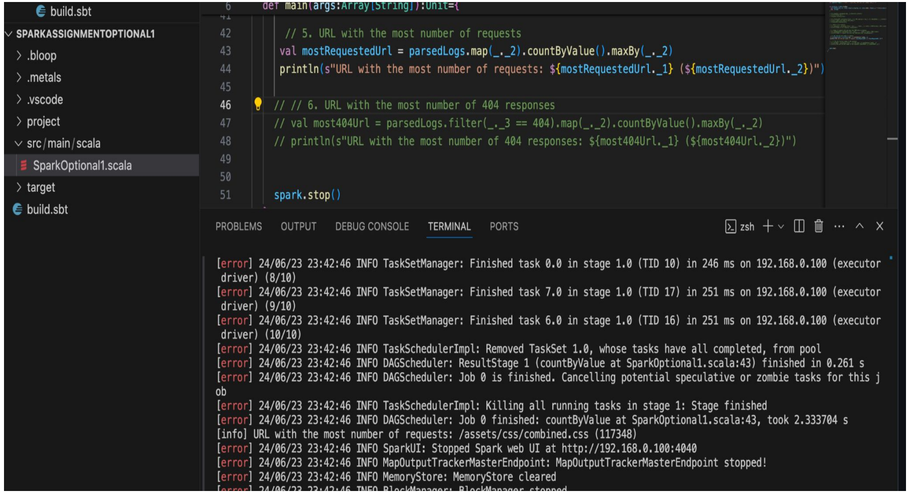
flatMap
MapPartitionsRDD [2]
flatMap at SparkOptional1.scala:35

map
MapPartitionsRDD [3]
map at SparkOptional1.scala:37

distinct
MapPartitionsRDD [4]
distinct at SparkOptional1.scala:37

▸ Show Additional Metrics

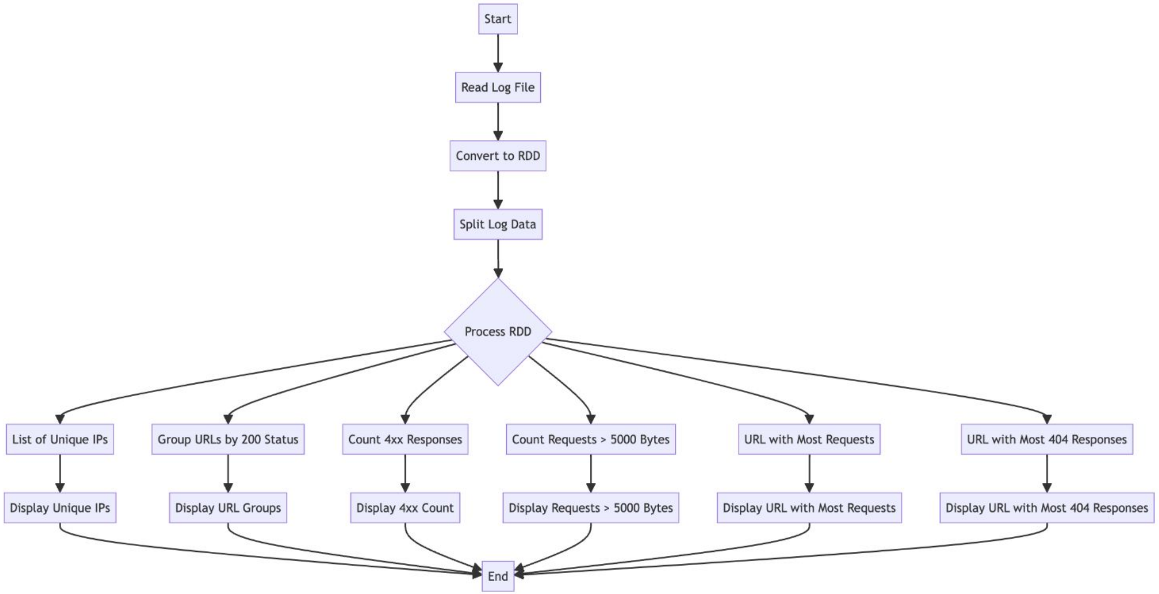
Screenshot of spark job operation



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

     // 5. URL with the most number of requests
    val mostRequestedUrl = parsedLogs.map(_._2).countByValue().maxBy(_._2)
    println(s"URL with the most number of requests: ${mostRequestedUrl._1} (${mostRequestedUrl._2})")

  // // 6. URL with the most number of 404 responses
  // val most404Url = parsedLogs.filter(_._3 == 404).map(_._2).countByValue().maxBy(_._2)
  // println(s"URL with the most number of 404 responses: ${most404Url._1} (${most404Url._2})")


  spark.stop()
```

```
[error] 24/06/23 23:42:46 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 10) in 246 ms on 192.168.0.100 (executor
 driver) (8/10)
[error] 24/06/23 23:42:46 INFO TaskSetManager: Finished task 7.0 in stage 1.0 (TID 17) in 251 ms on 192.168.0.100 (executor
 driver) (9/10)
[error] 24/06/23 23:42:46 INFO TaskSetManager: Finished task 6.0 in stage 1.0 (TID 16) in 251 ms on 192.168.0.100 (executor
 driver) (10/10)
[error] 24/06/23 23:42:46 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
[error] 24/06/23 23:42:46 INFO DAGScheduler: ResultStage 1 (countByValue at SparkOptional1.scala:43) finished in 0.261 s
[error] 24/06/23 23:42:46 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this j
ob
[error] 24/06/23 23:42:46 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
[error] 24/06/23 23:42:46 INFO DAGScheduler: Job 0 finished: countByValue at SparkOptional1.scala:43, took 2.333704 s
[info] URL with the most number of requests: /assets/css/combined.css (117348)
[error] 24/06/23 23:42:46 INFO SparkUI: Stopped Spark web UI at http://192.168.0.100:4040
[error] 24/06/23 23:42:46 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
[error] 24/06/23 23:42:46 INFO MemoryStore: MemoryStore cleared
[error] 24/06/23 23:42:46 INFO BlockManager: BlockManager stopped
```

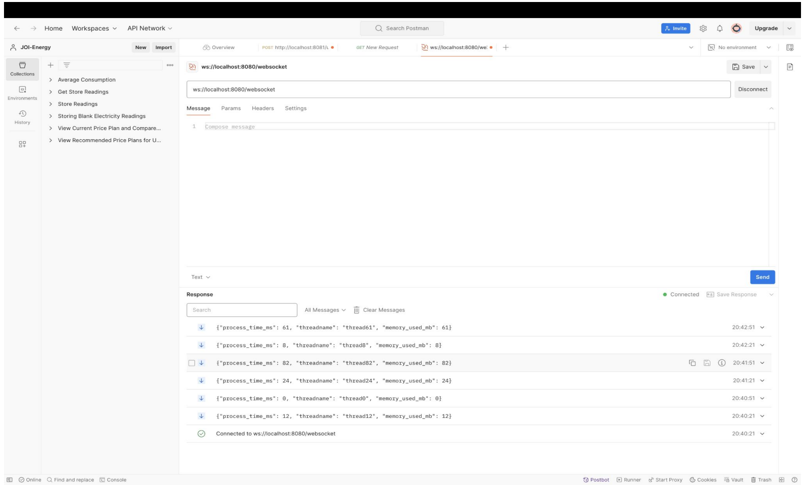
Flow Diagram - Optional 1



Optional Scenario 5

Description: Optional Scenario 5 involves creating a WebSocket using the Akka framework to produce JSON objects every 30 seconds with process time, thread name, and memory used. A Spark Streaming job consumes this WebSocket data, writes it to a Kafka topic in AVRO format, and another Spark Streaming job performs aggregation operations on the consumed data.

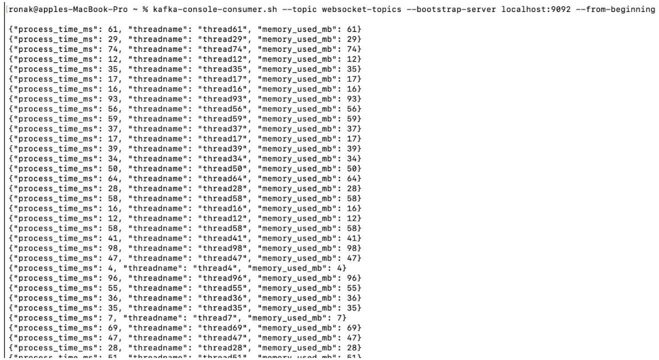
Screenshot of postman websocket connection



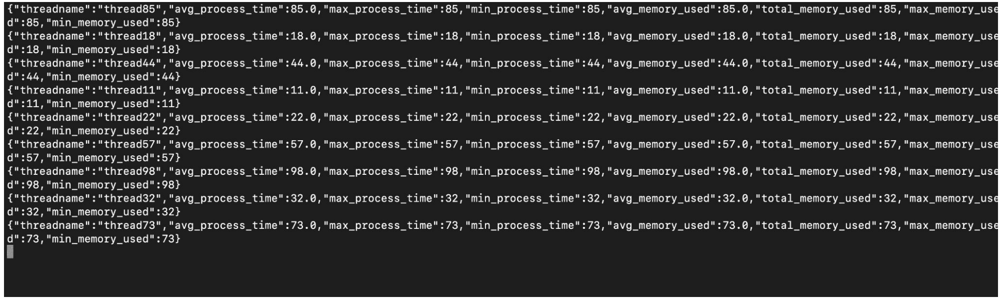
ws://localhost:8080/websocket

```
{"process_time_ms": 61, "threadname": "thread61", "memory_used_mb": 61}    20:42:51
{"process_time_ms": 8, "threadname": "thread8", "memory_used_mb": 8}       20:42:21
{"process_time_ms": 82, "threadname": "thread82", "memory_used_mb": 82}    20:41:51
{"process_time_ms": 24, "threadname": "thread24", "memory_used_mb": 24}    20:41:21
{"process_time_ms": 0, "threadname": "thread0", "memory_used_mb": 0}       20:40:51
{"process_time_ms": 12, "threadname": "thread12", "memory_used_mb": 12}    20:40:21
Connected to ws://localhost:8080/websocket                                 20:40:21
```

kafka consuming websocket data & aggregation output

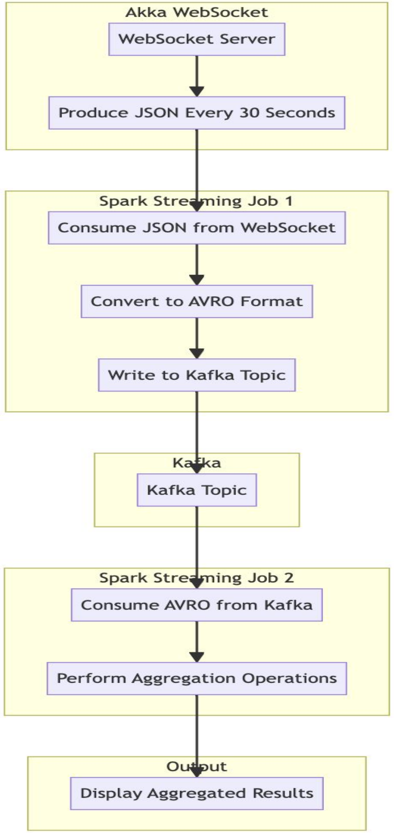


```
ronak@apples-MacBook-Pro ~ % kafka-console-consumer.sh --topic websocket-topics --bootstrap-server localhost:9092 --from-beginning

{"process_time_ms": 61, "threadname": "thread61", "memory_used_mb": 61}
{"process_time_ms": 29, "threadname": "thread29", "memory_used_mb": 29}
{"process_time_ms": 74, "threadname": "thread74", "memory_used_mb": 74}
{"process_time_ms": 12, "threadname": "thread12", "memory_used_mb": 12}
{"process_time_ms": 35, "threadname": "thread35", "memory_used_mb": 35}
{"process_time_ms": 17, "threadname": "thread17", "memory_used_mb": 17}
{"process_time_ms": 16, "threadname": "thread16", "memory_used_mb": 16}
{"process_time_ms": 93, "threadname": "thread93", "memory_used_mb": 93}
{"process_time_ms": 56, "threadname": "thread56", "memory_used_mb": 56}
{"process_time_ms": 59, "threadname": "thread59", "memory_used_mb": 59}
{"process_time_ms": 37, "threadname": "thread37", "memory_used_mb": 37}
{"process_time_ms": 17, "threadname": "thread17", "memory_used_mb": 17}
{"process_time_ms": 39, "threadname": "thread39", "memory_used_mb": 39}
{"process_time_ms": 34, "threadname": "thread34", "memory_used_mb": 34}
{"process_time_ms": 50, "threadname": "thread50", "memory_used_mb": 50}
{"process_time_ms": 64, "threadname": "thread64", "memory_used_mb": 64}
{"process_time_ms": 28, "threadname": "thread28", "memory_used_mb": 28}
{"process_time_ms": 58, "threadname": "thread58", "memory_used_mb": 58}
{"process_time_ms": 16, "threadname": "thread16", "memory_used_mb": 16}
{"process_time_ms": 12, "threadname": "thread12", "memory_used_mb": 12}
{"process_time_ms": 58, "threadname": "thread58", "memory_used_mb": 58}
{"process_time_ms": 41, "threadname": "thread41", "memory_used_mb": 41}
{"process_time_ms": 98, "threadname": "thread98", "memory_used_mb": 98}
{"process_time_ms": 47, "threadname": "thread47", "memory_used_mb": 47}
{"process_time_ms": 4, "threadname": "thread4", "memory_used_mb": 4}
{"process_time_ms": 96, "threadname": "thread96", "memory_used_mb": 96}
{"process_time_ms": 55, "threadname": "thread55", "memory_used_mb": 55}
{"process_time_ms": 36, "threadname": "thread36", "memory_used_mb": 36}
{"process_time_ms": 35, "threadname": "thread35", "memory_used_mb": 35}
{"process_time_ms": 7, "threadname": "thread7", "memory_used_mb": 7}
{"process_time_ms": 69, "threadname": "thread69", "memory_used_mb": 69}
{"process_time_ms": 47, "threadname": "thread47", "memory_used_mb": 47}
{"process_time_ms": 28, "threadname": "thread28", "memory_used_mb": 28}
{"process_time_ms": 51, "threadname": "thread51", "memory_used_mb": 51}
```



```
{"threadname":"thread85","avg_process_time":85.0,"max_process_time":85,"min_process_time":85,"avg_memory_used":85.0,"total_memory_used":85,"max_memory_use
d":85,"min_memory_used":85}
{"threadname":"thread18","avg_process_time":18.0,"max_process_time":18,"min_process_time":18,"avg_memory_used":18.0,"total_memory_used":18,"max_memory_use
d":18,"min_memory_used":18}
{"threadname":"thread44","avg_process_time":44.0,"max_process_time":44,"min_process_time":44,"avg_memory_used":44.0,"total_memory_used":44,"max_memory_use
d":44,"min_memory_used":44}
{"threadname":"thread11","avg_process_time":11.0,"max_process_time":11,"min_process_time":11,"avg_memory_used":11.0,"total_memory_used":11,"max_memory_use
d":11,"min_memory_used":11}
{"threadname":"thread22","avg_process_time":22.0,"max_process_time":22,"min_process_time":22,"avg_memory_used":22.0,"total_memory_used":22,"max_memory_use
d":22,"min_memory_used":22}
{"threadname":"thread57","avg_process_time":57.0,"max_process_time":57,"min_process_time":57,"avg_memory_used":57.0,"total_memory_used":57,"max_memory_use
d":57,"min_memory_used":57}
{"threadname":"thread98","avg_process_time":98.0,"max_process_time":98,"min_process_time":98,"avg_memory_used":98.0,"total_memory_used":98,"max_memory_use
d":98,"min_memory_used":98}
{"threadname":"thread32","avg_process_time":32.0,"max_process_time":32,"min_process_time":32,"avg_memory_used":32.0,"total_memory_used":32,"max_memory_use
d":32,"min_memory_used":32}
{"threadname":"thread73","avg_process_time":73.0,"max_process_time":73,"min_process_time":73,"avg_memory_used":73.0,"total_memory_used":73,"max_memory_use
d":73,"min_memory_used":73}
```

Flow Diagram - Optional Task 5



THANK YOU