Big Data Processing with EMR

Overview

In this project, you will learn how to process big data using **AWS EMR (Elastic MapReduce)**. You will work with a dataset from a **car rental marketplace**, leveraging **Spark on EMR** to process and transform raw data stored in **Amazon S3**. You will also integrate AWS **Glue, Athena, and Step Functions** to create a data pipeline.

Objectives

By the end of this lab, you will be able to:

1. **Understand AWS EMR** and how it fits in a big data ecosystem.
2. **Process raw data from S3 using Spark on EMR.**
3. **Transform datasets to derive key business metrics.**
4. **Use AWS Glue Crawlers and Athena** to analyze processed data.
5. **Automate data workflows with AWS Step Functions.**

Step 1: Understanding the Data

You are working with a car rental marketplace that provides vehicles for rent. The data consists of four datasets:

- **Vehicles Dataset:** Contains all available rental vehicles.
- **Users Dataset:** Contains user sign-up information.
- **Locations Dataset:** Contains master data for all rental locations.
- **Rental Transactions Dataset:** Contains records of vehicle rentals, including:
  - Rental start and end time
  - Pickup and drop-off locations
  - Vehicle ID
  - Total amount paid

All datasets are stored in **Amazon S3** in raw format.

Step 2: Setting Up AWS EMR Cluster

To process the data, you need to set up an **EMR Cluster**

Step 3: Processing Data with Spark on EMR

You will run two Spark jobs to transform the raw datasets and extract meaningful insights:

Spark Job 1: Vehicle and Location Performance Metrics

This job calculates key metrics by location and vehicle type:

- **Revenue per location**
- **Total transactions per location**
- **Average, max, and min transaction amounts**
- **Unique vehicles used at each location**
- **Rental duration and revenue by vehicle type**

Spark Job 2: User and Transaction Analysis

This job analyzes user engagement and transaction trends:

- **Total transactions per day**
- **Revenue per day**
- **User-specific spending and rental duration metrics**
- **Maximum and minimum transaction amounts**

Both jobs will **write the transformed data back to S3** in **Parquet format**.

Step 4: Creating AWS Glue Crawlers

Once the transformed data is in S3, use **AWS Glue Crawlers** to infer the schema and create a **Glue Data Catalog**

Step 5: Querying Data with Athena

1. Open **AWS Athena** and select the **Glue Data Catalog** as the source.
2. Use **SQL queries** to analyze the transformed data.
3. Example queries:
    - Find the **highest revenue-generating location.**
    - Find the **most rented vehicle type.**
    - Identify **top-spending users.**

Step 6: Automating the Pipeline with AWS Step Functions

To streamline the workflow, use **AWS Step Functions** to automate the process:

1. **Spin up an EMR cluster.**
2. **Run Spark jobs to process data.**
3. **Trigger Glue Crawler** after processing.

4. **Query data in Athena automatically.**
5. **Terminate the EMR cluster after job completion.**

Key Performance Indicators (KPIs)

After running the Spark jobs, you will derive the following KPIs:

Location and Vehicle Performance:

- **Total revenue per location**
- **Total transactions per location**
- **Average transaction amount per location**
- **Max/min transaction amount per location**
- **Unique vehicles used per location**
- **Rental duration metrics by vehicle type**

User and Transaction Metrics:

- **Total daily transactions and revenue**
- **Average transaction value**
- **User engagement metrics** (total transactions, total revenue per user)
- **Max and min spending per user**
- **Total rental hours per user**

Evaluation Criteria:

- Correct implementation of **ETL pipeline**.
- Proper **data validation** and error handling.
- Efficient computation of **KPIs** and storage in S3.
- Code **readability, efficiency, and best practices**.
- Well-structured documentation for ease of use and troubleshooting.