Technical Solution Document

Big Data Processing with AWS EMR

---

1. Introduction

1.1 Purpose

This document outlines the technical solution for processing big data using AWS Elastic MapReduce (EMR). The solution involves transforming raw datasets from a car rental marketplace stored in Amazon S3, using Apache Spark on EMR, and integrating AWS Glue, Athena, and Step Functions to create an automated data pipeline.

1.2 Scope

This solution is designed to:

- **Process raw data from Amazon S3 using Spark on EMR.**
- **Transform datasets to derive key business metrics.**
- **Store transformed data efficiently in Amazon S3 in Parquet format.**
- **Automate schema inference using AWS Glue Crawlers.**
- **Enable querying of processed data using AWS Athena.**
- **Automate the entire workflow using AWS Step Functions.**

1.3 Definitions, Acronyms, and Abbreviations

- **AWS EMR (Elastic MapReduce): A cloud-based big data platform for processing large-scale datasets.**
- **Apache Spark: A distributed computing framework for big data processing.**
- **AWS Glue: A fully managed ETL (Extract, Transform, Load) service.**
- **Amazon Athena: A serverless interactive query service for analyzing data in S3.**
- **AWS Step Functions: A workflow automation service.**

1.4 References

- **AWS EMR Documentation: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-what-is.html**

- **AWS Glue Documentation:** https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html
- **Amazon Athena Documentation:** https://docs.aws.amazon.com/athena/latest/ug/what-is.html

1.5 Overview

This document details the architectural representation, use cases, logical and deployment views, performance considerations, and quality attributes of the solution.

---

2. Architectural Representation

2.1 Description of Architectural Style and Design Patterns

- **The solution follows a data pipeline architecture.**
- **Uses batch processing with Apache Spark for data transformation.**
- **Incorporates a serverless approach with AWS Glue and Athena for data management and querying.**
- **Event-driven orchestration using AWS Step Functions.**

2.2 Architectural Goals and Constraints

- **Scalability: The solution should scale with increasing data volumes.**
- **Performance: Spark jobs should process data efficiently.**
- **Cost Optimization: Use on-demand EMR clusters and S3 for storage.**
- **Security: Implement IAM roles and policies for secure data access.**

---

3. Use-Case View

3.1 Use-Case Model

Actors:

- **Data Engineer**
- **Data Analyst**

Use Cases:

1. **Process Data with EMR: A Data Engineer triggers the EMR cluster to process raw data.**
2. **Analyze Processed Data: A Data Analyst queries the transformed data using Athena.**
3. **Automate Pipeline: AWS Step Functions orchestrate the entire workflow.**

3.2 Use-Case Realizations

- **Data Engineer uploads raw data → Spark jobs process data → Transformed data stored in S3 → Glue Crawlers update schema → Analysts query with Athena.**

---

4. Logical View

4.1 Overview of Major Components

- **Amazon S3: Stores raw and processed data.**
- **AWS EMR: Runs Spark jobs to transform data.**
- **AWS Glue: Crawls and catalogs processed data.**
- **Amazon Athena: Queries transformed data.**
- **AWS Step Functions: Automates data processing workflow.**

---

5. Process View

5.1 Concurrent Processes and Synchronization

- **Step 1: EMR processes data in parallel Spark jobs.**
- **Step 2: Transformed data is written back to S3 in Parquet format.**
- **Step 3: AWS Glue Crawlers run asynchronously to infer schema.**
- **Step 4: Athena allows interactive querying of processed data.**

---

6. Deployment View

6.1 Mapping of Software Components to Hardware

Component	Deployment Environment
Amazon S3	Cloud Storage
AWS EMR	Managed Spark Cluster
AWS Glue	Serverless Data Catalog
Amazon Athena	Serverless Query Engine
AWS Step Functions	Serverless Workflow Orchestration

---

7. Implementation View

7.1 Development Environment

- Programming Language: Python (for Spark jobs, AWS SDK automation).
- Frameworks: Apache Spark, AWS SDK (Boto3).
- Configuration Management: AWS IAM roles and policies for secure access.

7.2 Configuration Management and Versioning

- Use GitHub for source code versioning.
- Store ETL scripts in S3 for deployment.

---

8. Data Integrity and Security

- Implement AWS IAM roles for data access control.
- Use S3 bucket policies to restrict unauthorized access.

---

9. Size and Performance Considerations

9.1 Performance Benchmarks

- EMR cluster size: 3-Node cluster (m5.xlarge)
- Processing time: ~15 minutes for 10GB dataset
- Query execution time (Athena): <5 seconds for standard queries

9.2 Scalability Constraints

- **Auto-scaling EMR cluster based on workload.**
- **Optimize Spark jobs for parallel execution.**

---

10. Quality Attributes

10.1 Key Quality Attributes

- **Usability: Analysts can query data with Athena using SQL.**
- **Reliability: Redundant data storage in S3.**
- **Security: IAM roles for controlled access.**
- **Maintainability: Modular ETL scripts for easy updates.**

---

11. Appendices

11.1 Glossary

- **ETL: Extract, Transform, Load**
- **Parquet: Optimized columnar storage format**

11.3 Revision History

Version	Date	Author	Changes
1.0	04/04/2025	Marzuk	Initial Draft