📄 Technical Solution Document: Lakehouse Architecture for E-Commerce Transactions

---

1. Introduction

Purpose

This document describes the architecture and implementation of a Lakehouse system on AWS for managing and analyzing e-commerce transactional data using Delta Lake, AWS Glue, and Amazon Athena.

Scope

The system handles the full lifecycle of data ingestion, transformation, storage, and querying of structured e-commerce datasets. It ensures high reliability, referential integrity, and operational automation.

Definitions, Acronyms, and Abbreviations

- **ETL**: Extract, Transform, Load
- **CI/CD**: Continuous Integration / Continuous Deployment
- **S3**: Amazon Simple Storage Service
- **Glue**: AWS data processing service
- **Delta Lake**: Storage layer providing ACID transactions
- **Athena**: Serverless SQL query engine
- **DWH**: Data Warehouse

References

- Project README
- “Building a Lakehouse Using PySpark Delta Tables & S3” (PDF)

Overview

The system is based on a Lakehouse design that leverages AWS Glue for scalable data processing, Delta Lake for ACID-compliant storage, and Amazon Athena for analytics. Orchestration is performed using Step Functions, and all components are integrated using a CI/CD pipeline via GitHub Actions.

---

2. Architectural Representation

Description of Architectural Style

- **Architectural Style**: Lakehouse on AWS
- **Design Pattern**: Data Pipeline, ETL with ACID layer
- **Structure**: Modular and event-driven with stateless processing

---

3. Architectural Goals and Constraints

Goals

- Build a scalable, modular data processing pipeline
- Guarantee data quality and schema compliance
- Enable quick analytics via Delta Lake and Athena
- Ensure automated and reproducible workflows

Constraints

- Use AWS-native services
- Deploy only on AWS cloud
- S3 as the primary storage layer
- Real-time orchestration not required, but batch freshness must be reliable

---

4. Use-Case View

Use-Case Model

Primary Use-Case: Ingest and process CSV transactional data and expose it for analytical querying.

Use-Case Realizations

1. **Raw File Detected → Glue ETL → Delta Format → Athena Queryable**
2. **Failed Job → Log + Optional Alert**

3. **Post-success → Archive Raw File**

---

5. Logical View

Major Logical Components

- **Data Ingestion Handler**
- **ETL Job Runner (Glue Job using Spark)**
- **Delta Table Writer**
- **Metadata Updater (Glue Crawler)**
- **Athena Query Layer**

---

6. Process View

Concurrent Processes

- AWS Step Functions execute concurrent Glue Jobs per dataset
- Each job independently processes orders, order_items, and products

---

7. Deployment View

Mapping of Components

- **S3**: Raw, Processed, and Archived zones
- **Glue Jobs**: Run Spark-based ETL
- **Step Functions**: Orchestrate ETL workflow
- **Athena**: Query Delta Lake tables

---

8. Implementation View

Development Environment

- Python 3.x
- PySpark

- Delta Lake
- AWS SDK (boto3)
- GitHub Actions for CI/CD

Configuration Management

- Defined via pyproject.toml and requirements.txt
- .github/workflows/ for CI/CD

---

9. Data View

Schema Definitions

- **Product**: product_id (PK), department_id, ...
- **Orders**: order_id (PK), order_num, ...
- **Order Items**: id (PK), order_id (FK), product_id (FK), ...

Check assets/ in repo for full ERD

Storage & Retrieval

- Delta Lake format with schema enforcement
- Partitioning on date
- Deduplication logic before write

Integrity & Security

- Referential integrity checks across datasets
- Logging of bad records
- IAM-based S3 access controls

---

10. Size and Performance

Performance Benchmarks

Scalability Constraints

- Dependent on AWS Glue job scaling and S3 throughput

- Horizontal scalability via partitioned processing

---

11. Quality

Attributes

- **Usability**: SQL access via Athena
- **Reliability**: ACID guarantees with Delta
- **Maintainability**: Modular ETL code
- **Security**: IAM roles + secure S3 access
- **Testability**: Full test suite via Pytest

---

12. Appendices

Revision History

Version	Date	Author	Change Description
1.0	2025-04-17	Marzuk	Initial Draft