Serverless EV ETL Pipeline

A serverless, event-driven data pipeline for processing Electric Vehicle (EV) charging session data using AWS services. This project demonstrates how to build a cost-efficient and scalable ETL (Extract, Transform, Load) workflow using serverless technologies and Infrastructure as Code (IaC) via Terraform.

Architecture Overview

Component	Description
Amazon S3 (Landing Zone)	Stores raw CSV files containing EV session data uploaded from data sourc
AWS Lambda (Trigger)	Automatically triggered upon file upload in S3; initiates the ETL process.
AWS Glue (Transformation Layer)	Cleans, validates, and transforms the raw data into partitioned Parquet forn
Amazon S3 (Curated Zone)	Stores transformed Parquet data in a structured, query-friendly format.
Amazon Athena (Query Layer)	Enables SQL-based querying and analysis of curated EV data directly from
Terraform	Manages and provisions all infrastructure resources in a reproducible, decla

Key Features

• Serverless and event-driven architecture — no manual triggers required.
• Infrastructure as Code (IaC) using Terraform.
• Cost-efficient — only pay for compute time when jobs run.
• Scalable — automatically adapts to data volume.
• Domain-specific design — optimized for EV charging session datasets.

Evaluation and Review

Strengths
■ Clear architecture using AWS serverless services.
■ Infrastructure as Code with Terraform for reproducibility.
■ Serverless, scalable, and cost-efficient design.
■ Educational and domain-focused implementation.
Limitations
■■ Documentation lacks deep operational details.
■■ No integrated monitoring or alerting (CloudWatch).
■■ No schema evolution or data validation pipeline.
■■ Missing IAM security and encryption best practices.
■■ No automated testing or CI/CD workflow.

Verdict

This project is a strong proof-of-concept implementation of a serverless ETL pipeline for EV data processing. It effectively demonstrates an end-to-end workflow using AWS-native tools. To make it production-ready, enhancements in monitoring, schema evolution, security, and testing are recommended.

Future Enhancements

- Add schema evolution and data validation.
- Integrate CloudWatch metrics and alerts.
- Include CI/CD for automated deployments.
- Optimize Glue job partitioning for performance.
- Add IAM least privilege and encryption policies.
- Implement cost monitoring and lifecycle management.