End-to-End Data Engineering Pipeline Documentation

1. Project Overview

This project implements a complete end-to-end data engineering pipeline for ingesting vehicle CAN bus data, processing it on AWS, and making it available for analytics and reporting.

The objective is to build a scalable, event-driven, production-grade data platform using modern cloud services.

2. Architecture Overview

The architecture follows a modular, cloud-native design using AWS services.

Core components include Amazon S3 for storage, EventBridge for event routing, AWS Lambda for lightweight processing, AWS Glue for ETL, Glue Data Catalog for metadata, Athena for querying, and CloudWatch for monitoring.

Data flows from raw ingestion to curated analytics-ready datasets.

3. Data Sources

Source data originates from vehicle ECUs via CAN Bus systems.

Data is captured using CAN Logger devices and stored in MDF/MF4 or structured formats such as JSON or CSV before ingestion.

The data includes telemetry such as speed, RPM, temperature, and diagnostic codes.

4. Ingestion Layer

Data files are uploaded into an Amazon S3 raw bucket.

Each successful upload generates an ObjectCreated event which is routed via EventBridge and triggers Lambda-based validation and routing logic.

This ensures automated ingestion without manual intervention.

5. Storage Design

The storage layer follows a zone-based design:

- Raw Zone: Original unprocessed files

- Processed Zone: Cleaned and standardized data

- Curated Zone: Analytics-ready datasets

S3 partitioning is applied using date and source identifiers for performance optimization.

6. Data Processing & Transformations

AWS Glue jobs perform ETL processing including schema normalization, filtering invalid records, handling nulls, deduplication, and converting files into columnar formats such as Parquet.

Business rules are applied during transformation to ensure data quality.

7. Data Schema & Catalog

AWS Glue Data Catalog maintains table definitions and schemas.

This allows Athena and other analytics tools to query data using SQL.

Schemas include fields such as vehicle_id, timestamp, signal_name, and signal_value.

8. Orchestration & Workflow

EventBridge controls orchestration. File arrival triggers downstream processing automatically.

Retry mechanisms and controlled execution ensure reliability.

9. Monitoring & Logging

Amazon CloudWatch captures logs, metrics, and alarms across Lambda, Glue, and EventBridge.

Alerts are configured for failures to ensure proactive monitoring.

10. Error Handling & Recovery

Failures are logged and captured. Failed records are isolated for reprocessing.

The pipeline supports idempotent execution to avoid duplicate processing.

11. Security & Access Control

IAM roles enforce least-privilege access.

S3 encryption is enabled and secure bucket policies are implemented.

Access to datasets is controlled using fine-grained permissions.

12. Performance & Scalability

The system is designed to scale automatically using serverless services.

Partitioning, optimized file formats, and distributed processing ensure performance at scale.

13. Cost Considerations

Storage lifecycle policies are applied to control costs.

Serverless components minimize idle resource expenses.

14. Deployment & Environment Setup

The pipeline can be deployed across dev, test, and production environments using consistent configurations.

Infrastructure can be automated using IaC tools.

15. Testing Strategy

Sample datasets and validation queries are used to test transformations.

Data quality checks validate schema consistency and completeness.

16. Known Limitations

Real-time streaming is currently not implemented.

Extremely large binary formats require optimization.

17. Future Enhancements

Integration with real-time services like Kinesis.

Advanced analytics dashboards and ML model integration.

18. Runbook (Operations Guide)

Operators can monitor pipelines via CloudWatch.

In case of failure, logs indicate root cause and jobs can be safely re-run.

19. Appendix

Includes sample schemas, example queries, and configuration references.