https://github.com/SaiHarshitM/Data-Architecture

EVERGREEN MARINE CORP.

- Sai Harshit Maddila

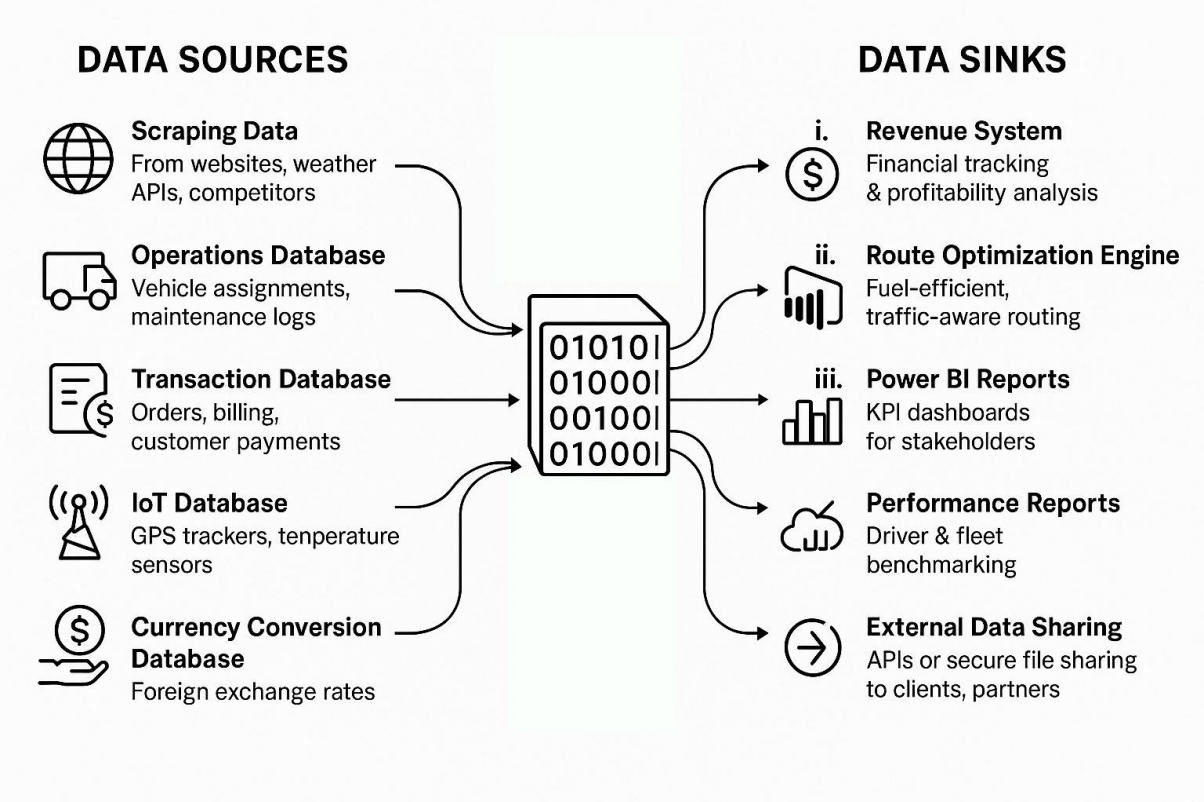
Contents

https://github.com/SaiHarshitM/Data-Architecture

OBJECTIVE:

Design a scalable, secure, and efficient data architecture for Evergreen Marine Corps that supports data-driven decision making, enhances operational efficiency, and promotes sustainable business growth.

1. Scalable: Design and implement a scalable cloud architecture to ensure efficient resource utilization, high availability, and adaptability to varying workloads.
2. Secure: Design and implement secure cloud architecture to protect data, ensure compliance, and safeguard against threats and vulnerabilities.
3. Efficient: Design and implement an efficient cloud architecture to optimize performance, reduce latency, and minimize operational costs.
4. Flexible: Support diverse data types (structured, semi-structured, unstructured) and integrate with multiple tools and platforms.



DATA SOURCES

1. Scraping Data

- *Nature:* Unstructured or semi-structured data scraped from carrier sites, weather updates, traffic APIs, Geopolitical think tanks, customer interactions, shipment tracking requests, and form submissions.
- *Format:* JSON, HTML, CSV, web stream
- *Usage:* Used to make routing decisions, delivery estimates, competitor benchmarking, enhance user experience, demand forecasting, and lead generation

2. Operations Database

- *Nature:* Structured data on vehicle assignments, maintenance, route logs, Delivery Schedules etc.
- *Format:* Relational tables (SQL).
- *Usage:* Supports fleet optimization, maintenance scheduling, and fuel efficiency tracking.

3. Transaction Database

- *Nature:* Structured data on shipment orders, billing, and customer payments.
- *Format:* SQL tables or NoSQL documents.
- *Usage:* Used for invoicing, financial reporting, and customer account reconciliation.

4. IoT Database

- *Nature:* Time-series telemetry data from GPS trackers, sensors, and onboard devices.
- *Format:* JSON, CSV, or time-series database formats.
- *Usage:* Enables real-time tracking, environmental monitoring (e.g., temperature-sensitive cargo), and performance analytics.

5. Currency Conversion Database

- *Nature:* Structured foreign exchange rate data relevant to international logistics.
- *Format:* JSON, XML, or SQL.
- *Usage:* Powers multi-currency invoicing, cost estimation, and budgeting across regions.

DATA SINKS

i. Revenue System

- *Purpose:* To calculate, track, and analyse all financial transactions across Evergreen's operations.
- *Details:* Ingests processed data from the transaction database, currency conversions, and customer billing to provide real-time visibility into revenue streams, cost margins, and profitability by route, region, or service type. Integrated with accounting tools for auditing and forecasting.

ii. Route Optimization Engine

- *Purpose:* To improve delivery efficiency and reduce operational costs by finding the most optimal routes.
- *Details:* Consumes enriched data from IoT sensors (e.g., location, traffic, load status), fleet operations, and external scraping sources (e.g., weather, road conditions). Uses algorithms or ML models to optimize routes dynamically for time, fuel, and resource allocation.

iii. Power BI Database & Reports

- *Purpose:* To provide business intelligence dashboards and reports for informed decision-making across departments.
- *Details:* Aggregates cleaned and modelled data from all sources—fleet data, revenue, IoT feeds, transactions—into tabular models and datasets in Power BI. Enables visual reporting on KPIs like delivery success rate, fleet utilization, shipment delays, and financial performance.

iv. Performance Reports

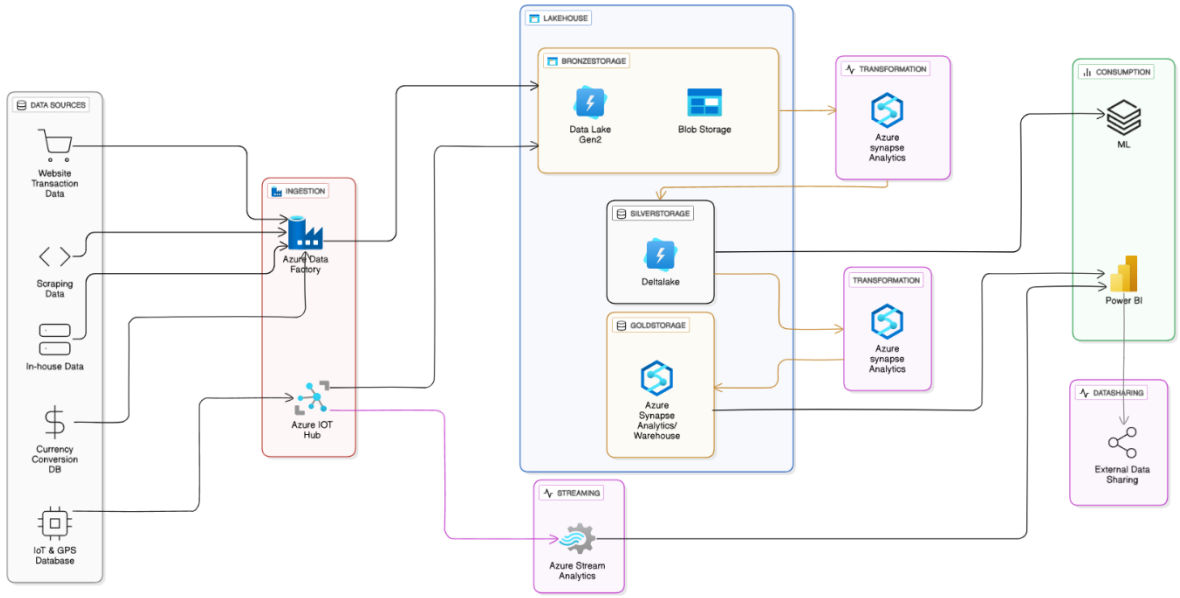
- *Purpose:* To evaluate and benchmark operational efficiency and overall logistics effectiveness.
- *Details:* Generated periodically (daily, weekly, monthly) using IoT metrics (e.g., idle time, fuel consumption), delivery logs, and SLA records. Used by operations managers to assess trends, detect bottlenecks, and drive performance improvements.

v. External Data Sharing

- *Purpose:* To securely share data with partners, clients, regulators, and third-party services.
- *Details:* Exports curated data subsets (e.g., shipment status, CO2 emissions, service reports) via APIs or secure file transfers. Data is often anonymized or filtered to protect sensitive business information. Used for client transparency, compliance reporting, and third-party analytics.

https://github.com/SaiHarshitM/Data-Architecture

Detailed Cloud Architecture: Layered Approach



The architecture is built on Microsoft Azure and is logically divided into four main layers: **Data Sources**, **Ingestion**, **Lake House** & **Transformation**, and **Consumption**. This layered approach ensures that data is systematically collected, stored, processed, and delivered.

Ingestion Layer

The Ingestion Layer is the entry point for all data into the platform. It is designed to handle both batch and real-time data streams.

- *Azure Data Factory (ADF):* This service is the backbone for **batch ingestion**. It is a cloud-based ETL/ELT service that orchestrates and automates data movement and transformation. ADF is ideal for pulling data from transactional databases, web-scraped sources, and in-house systems on a scheduled basis (or trigger). It manages the entire workflow from the various data sources into the Bronze Layer of the Lake House.
- *Azure IoT Hub:* This service is purpose-built for ingesting **real-time data streams**. It provides a secure and scalable solution for connecting, monitoring, and managing millions of IoT devices. The IoT Hub captures time-series telemetry data from GPS and other sensors on ships and vehicles like machinery health and temperature, routing this high-volume data directly into the streaming pipeline.

Lake House: A Tiered Storage Strategy

The Lake House is a modern data architecture that combines the low-cost storage of a data lake with the structure and management features of a data warehouse. Data is organized into three distinct layers, often referred to as Bronze, Silver, and Gold.

Bronze Layer (Raw Storage)

The Bronze Layer is the raw landing zone for all ingested data.

- *Azure Data Lake Gen2 & Blob Storage:* This layer stores all raw, unprocessed data exactly as it was ingested. This includes raw files from ADF pipelines and streaming data from IoT Hub. The data is kept in its original formats (e.g., CSV, JSON, Parquet), which is critical for historical tracking, data reprocessing, and establishing a complete data lineage.

Silver Layer (Cleaned & Conformed Data)

The Silver Layer is where the first stage of transformation and curation occurs.

- *Delta Lake:* This is an open-source storage layer that provides ACID (Atomicity, Consistency, Isolation, Durability) transactions on top of the data stored in the Data Lake. The Silver Layer stores data that has been cleansed, de-duplicated, and conformed to a consistent schema. This makes the data more reliable and ready for deeper analysis. Delta Lake also provides features like schema enforcement, time travel, and data versioning.

Gold Layer (Curated & Aggregated Data)

The Gold Layer is the final, highly refined layer of the Lake House.

- *Azure Synapse Analytics (Warehouse):* This is a powerful analytics service that brings together data warehousing and big data analytics. The Gold Layer stores highly curated, aggregated, and business-ready data. This data is optimized for fast query performance and is typically modeled in a star or snowflake schema to serve dashboards and BI tools like Power BI.

Detailed Data Flow: Batch and Stream data

The architecture handles two distinct types of data flow to process both scheduled and real-time data efficiently.

Batch Data Flow

This pipeline is designed for structured and semi-structured data that does not require real-time processing, such as transactional data, scraped information, and operational logs.

- *Ingestion: Azure Data Factory (ADF)* is the primary tool for this process. It pulls data from the **Operations Database** (SQL), **Transaction Database** (SQL/NoSQL), **Scraping Data** (JSON, HTML, CSV), and **Currency Conversion Database** (JSON, XML, SQL).
- *Storage (Bronze Layer):* The raw data is ingested by ADF and lands in **Azure Data Lake Gen2** and **Blob Storage**. The data is stored exactly as it was received, preserving its original format for a complete data history.
- *Curation (Silver Layer):* The data is then processed in the Silver Layer. Here, using **Azure Synapse Analytics (Spark Pools)**, the raw data is cleaned, validated, and de-duplicated. Data from different sources is conformed to a standardized schema, and a transactional layer is added using **Delta Lake**. For example, scraped data might be parsed from HTML into a structured JSON format, and transactional data might be checked for completeness.
- *Aggregation (Gold Layer):* The curated data is then aggregated and modeled in the Gold Layer, which is an **Azure Synapse Analytics (Warehouse)**. This is where data is restructured into star or snowflake schemas, summarizing key metrics for easy consumption.
- *Consumption:* From the Gold Layer, the data is served to various sinks, including **Power BI Reports** for dashboards, the **Route Optimizer** for strategic decisions, and the **Revenue System** for financial analysis.

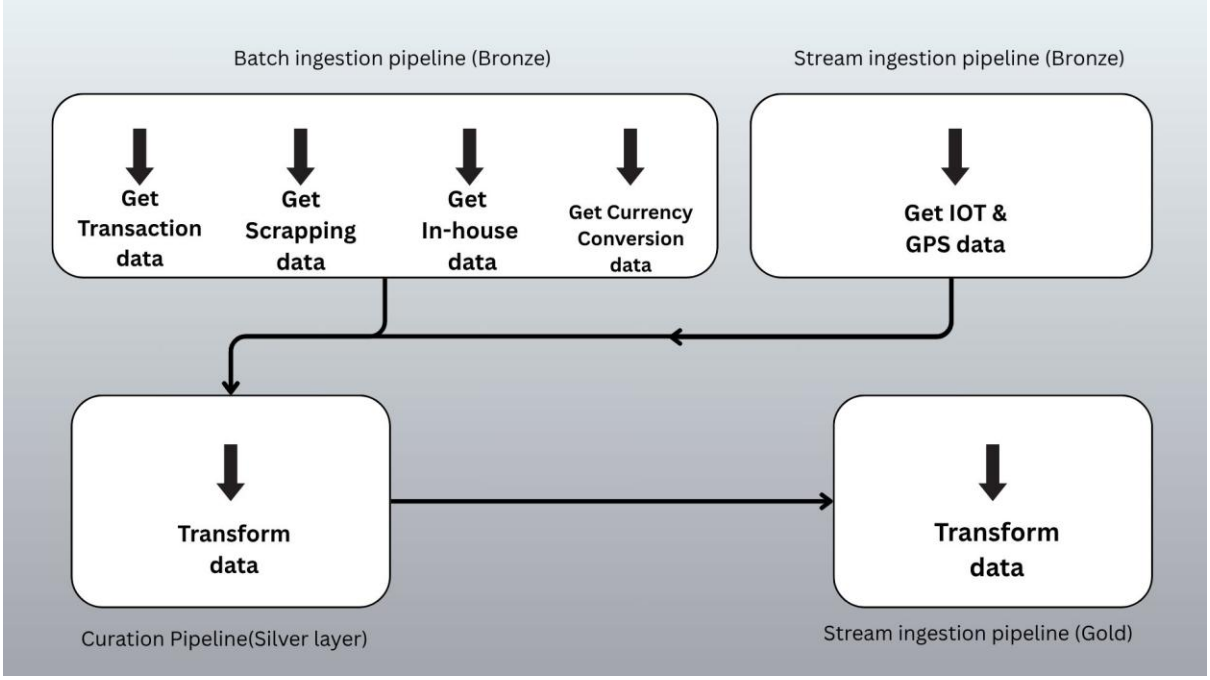
Stream Data Flow

This pipeline is specifically designed for high-velocity, time-sensitive data, primarily from IoT devices.

- *Ingestion: Azure IoT Hub* is the main entry point, securely ingesting real-time telemetry data from GPS trackers and various sensors. This data is in a JSON or time-series format.
- *Real-time Processing:* The data is then immediately fed into **Azure Stream Analytics**. This service processes the streaming data on the fly, performing simple transformations and aggregations in real-time.

- *Storage:* The IOT Hub data is then written directly to the **bronze Layer (Azure blob)** to be stored.
- *Consumption:* The **Azure Stream Analytics** data is then immediately fed into dashboards, for the 3rd party APIs to grab the data

Pipeline Design



Parent and Child Pipeline Design

The overall architecture can be viewed as two main parent pipelines: the **Batch Ingestion Pipeline** and the **Stream Ingestion Pipeline**. These parent pipelines are responsible for orchestrating the entire data flow from source to consumption.

Batch Ingestion Pipeline

This is the primary parent pipeline for all structured and semi-structured data that does not require real-time processing. Its main function is to gather data from various sources before any transformation occurs.

- **Parent Pipeline:** The "Batch Ingestion pipeline (Bronze)" is the parent pipeline.
- **Child Pipelines/Activities:** Within this parent pipeline, there are four distinct child activities. Each of these can be configured as a separate, reusable pipeline that runs as a step in the parent pipeline.
    - Get Transaction data: This child activity is responsible for pulling structured data from the Transaction Database.
    - Get Scraping data: This child activity ingests unstructured or semi-structured data from various external web sources.

- Get In-house data: This child activity retrieves structured data from the Operations Database.
- Get Currency Conversion data: This child activity fetches exchange rate data.

The parent pipeline's job is to ensure these four child activities run successfully, collecting all the necessary raw data and depositing it into the Bronze Layer of the data lake.

Stream Ingestion Pipeline

This is the parent pipeline dedicated to handling high-velocity, real-time data from IoT devices.

- **Parent Pipeline:** The "Stream ingestion pipeline (Bronze)" and the "Stream ingestion pipeline (Gold)" act as parent pipelines for their respective stages.
- **Child Activities:** The initial step, Get IOT & GPS data, is the child activity that listens to the stream from the Azure IoT Hub.

Pipeline Dependencies and Data Flow

The parent-child relationship extends to the data flow between layers, creating a clear dependency chain.

Batch Data Flow Dependency

The "Curation Pipeline (Silver layer)" is a subsequent pipeline that is **dependent** on the successful completion of the "Batch Ingestion pipeline (Bronze)". It acts like a child pipeline that is triggered only after the parent has finished.

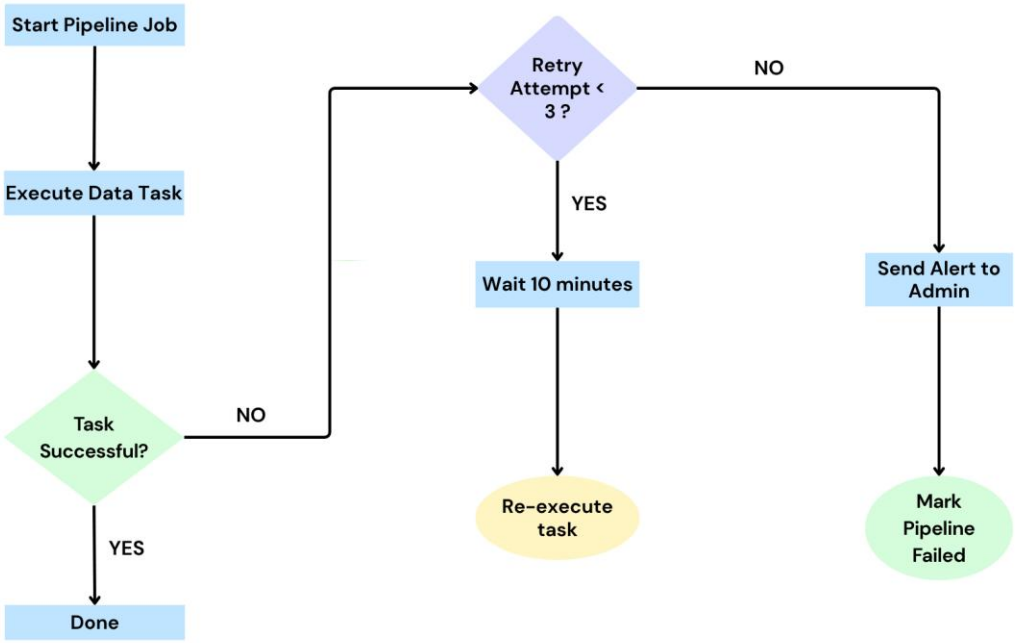
1. **Parent completion:** The "Batch Ingestion" parent pipeline first executes its four child activities to ingest all raw data into the Bronze layer.
2. **Child execution:** Once the Bronze layer is populated, the "Curation Pipeline" (Silver layer) is triggered. This pipeline's job is to take the raw data, perform cleaning, validation, and de-duplication, and then store the conformed data in the silver layer.

Stream Data Flow Dependency

The stream data flow follows a similar dependency model, though it is continuous rather than a triggered job. The stream ingestion pipeline takes this raw stream and performs real-time transformations and aggregations on the fly, immediately making the refined data available. This is a continuous dependency rather than a one-time trigger.

The parent-child pipeline structure is a robust and efficient way to manage a complex data platform, ensuring that data is processed in the correct order and that each step of the pipeline has a clear, defined responsibility.

Pipeline Failure Strategy



A critical component of this design is the automated failure handling strategy. The system is configured to be resilient to transient issues and to alert administrators when a persistent problem occurs.

1. **Start Pipeline Job:** The pipeline begins a data task.
2. **Execute Data Task:** The task performs its designated function (e.g., data ingestion, cleaning).
3. **Check for Success:** If the task is successful, the process completes.
4. **Failure and Retry:** If the tasks fail, the system checks if the number of **retry attempts is less than 3**.
5. **Wait and Re-execute:** If retries are available, the system waits for 10 minutes before re-executing the task. This is known as **exponential backoff**.
6. **Admin Alert:** If all three retry attempts fail, the system sends an alert to the data administrator and marks the pipeline as failed and is terminated. This ensures that persistent issues are addressed promptly.

Conclusion

This detailed cloud data platform architecture provides Evergreen Marine Corp with a state-of-the-art, end-to-end solution for managing its data. By leveraging a comprehensive set of Azure services and a tiered Lake House approach, the company can transform raw data into actionable insights, driving operational efficiency and supporting strategic decision-making. The scalable, secure, and flexible nature of this design positions the company for future growth and innovation in the logistics industry.