ITI Gradutation Project

● Real-Time NYC Transit Monitoring & Analytics Platform ●

ITI Branch	Minia
Track	Data Engineering
Project Team	• AbdElrahman Mostafa • Moaaz omar • Yousef mahmoud • Maher Mahmoud • David Neil • Hussein Mohamed

---

1. ❓ Problem Statement

The Metropolitan Transportation Authority (MTA) data exists in two disjointed technical formats:
● Static schedules are delivered as **text files (batch)**
● Real-time updates arrive as highly efficient, but non-human-readable **GTFS-Realtime Protobuf messages**.

This architectural and format disconnect prevents efficient, real-time correlation between 'what is planned' and 'what is happening.' There is a critical need for a **distributed data pipeline** capable of seamlessly ingesting, decoding, and joining these two fundamentally different data sources into a unified, low-latency record for operational analytics.

---

2. 👥 Target Audience

The primary users of this platform are the teams within the **Metropolitan Transportation Authority (MTA)**:

- **Transit Operations Staff:** Requires immediate, high-confidence data on service health to make proactive decisions, reroute trains, or adjust staff assignments.
- **Performance Analytics Teams:** Needs a clean, reliable, and historically aligned dataset to conduct retrospective analysis, identify bottlenecks, and inform long-term planning and scheduling adjustments.

---

3. ⭐ End Goals

The ultimate measure of project success is defined by the following impacts for the MTA:

- **Operational Readiness:** Successfully implement a production-ready, low-latency **Lambda Architecture** that demonstrates the feasibility of handling MTA's complex data streams with open-source tools.
- **Provide a live dashboard:** enabling Operations Staff to identify and react to service interruptions *before* they result in major delays across the network.
- **Data Reliability:** Successfully parse the complex **GTFS-Realtime (protobuf) binary stream** and creating an enriched dataset that is historically accurate and ready for complex analytical queries.

4. 📈 Key Objectives of the Project

Objectives

- **Batch Pipeline:** Develop a pipeline to download, parse, and store the static GTFS schedule data (routes, stops, trips) into a data warehouse using a Medallion Architecture.
- **Streaming Pipeline:** Develop a pipeline to ingest, parse (from protobuf), and publish the live GTFS-Realtime feeds using Serverless AWS components.
- **Storage Layer:** Store the final, enriched data in a specialized time-series database.
- **Visualization:** Build a unified Grafana dashboard to visualize key performance indicators (KPIs) from both batch and streaming sources.

Key Performance Indicators (KPIs)

Batch Pipeline (Static Analysis):

KPI	Description
Total Routes / Stops / Trips	High-level network overview
Trips per Route	Route utilization comparison
Avg Trip Duration per Route	Identifies longest routes
Time Between Stops per Route	Stop frequency analysis
Top Stations by Trip Volume	Busiest station identification

Streaming Pipeline (Real-Time Monitoring):

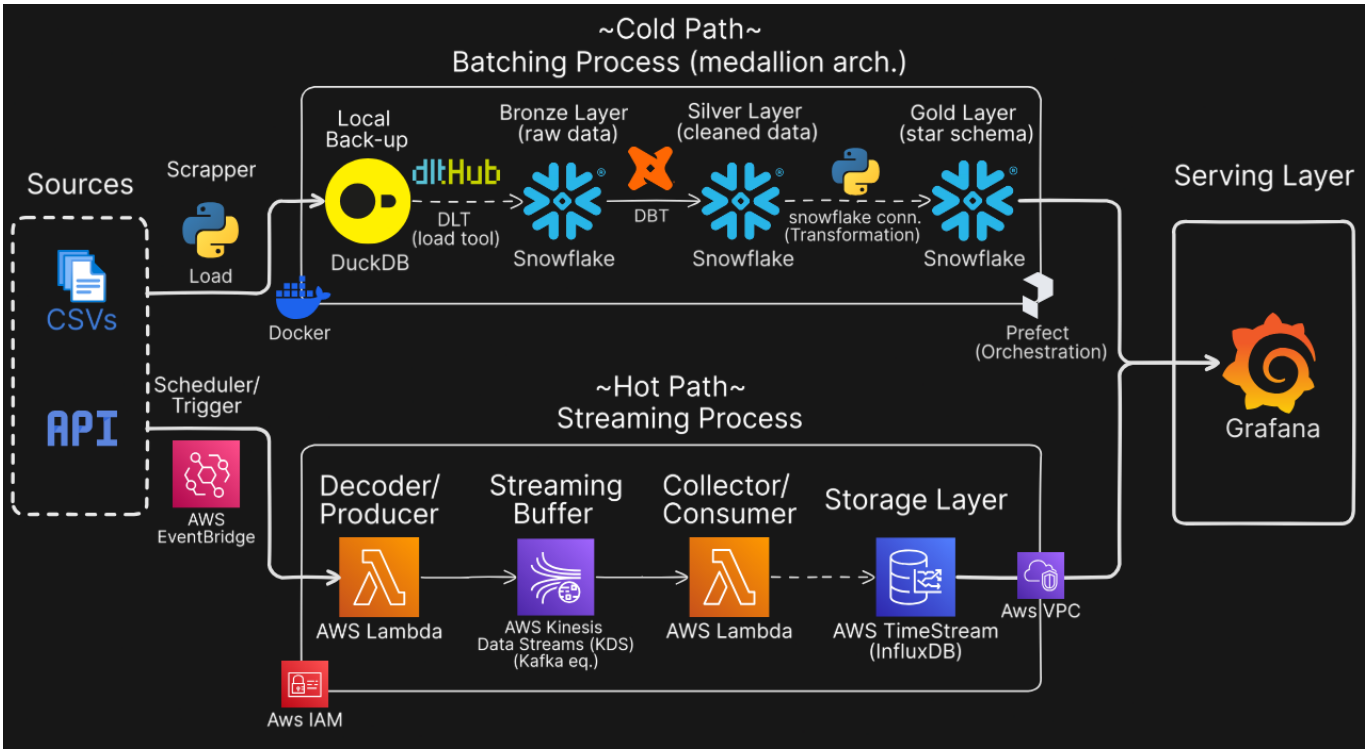
KPI	Description
Active Data Points (Last 1 Hour)	Volume of real-time updates
Avg Delay (Last 1 Hour)	Network-wide delay indicator
Active Alerts	Current service disruptions
Delays Over Time by Route	Trend analysis per line
Top 10 Delayed Stations	Hotspot identification
Route Performance (Delay/Route)	Per-route delay breakdown

---

5. 💾 Data Sources

Component	Source	Format	Purpose
Batch Data	MTA Static GTFS (data.gov)	ZIP file (TXT files)	Static context: Routes, Stops, Planned Trips
Streaming Data	MTA GTFS-Realtime API	Protocol Buffer (Protobuf)	Real-time vehicle positions, trip updates, and alerts

6. 🏗️ Proposed Technical Architecture



---

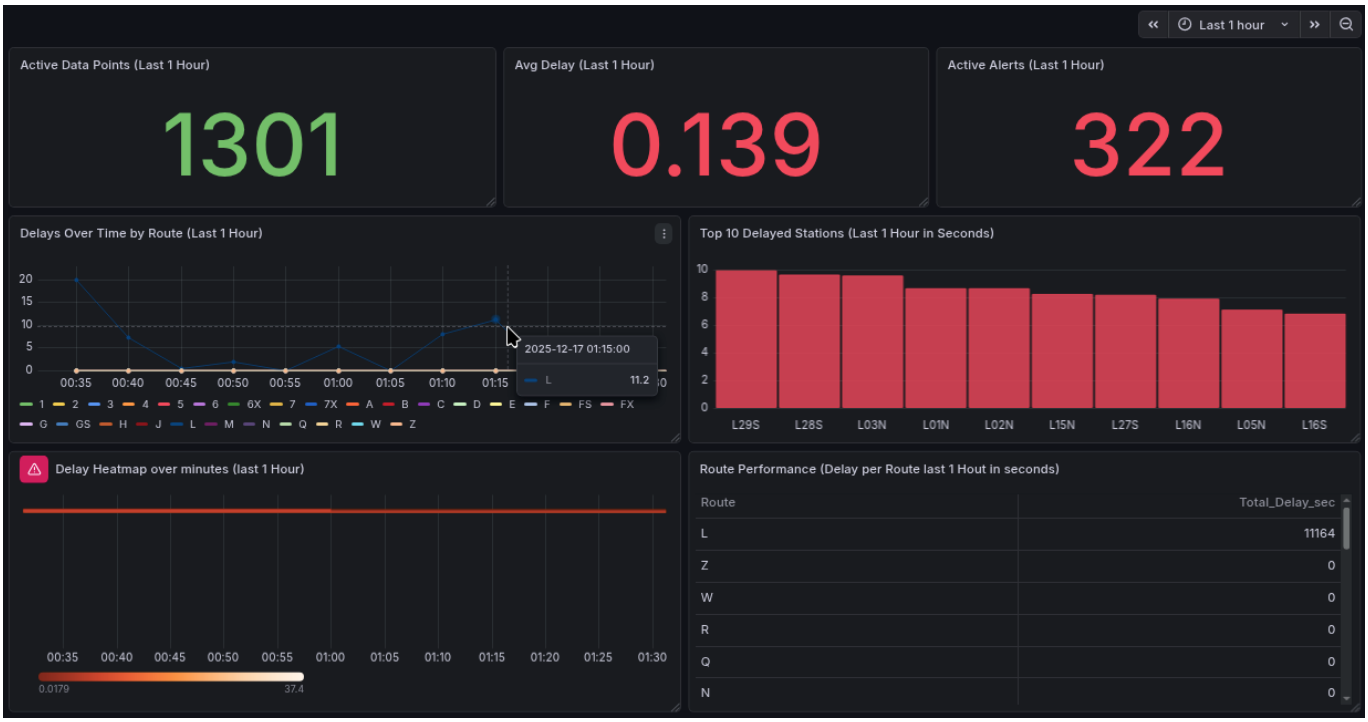
7. 💻 Technology Stack

Category	Component	Purpose
Project Management	Jira	Issue tracking, sprint planning, task delegation, and progress reporting
Data Sources	MTA GTFS Schedule	Static Batch Data (CSVs)
	MTA GTFS-Realtime API	Live Streaming Data (Protobuf)
Cloud Provider	AWS	Core infrastructure for streaming and orchestration
Security	AWS IAM	Access management and security policies
	AWS VPC	Network isolation and security for streaming components
Batch Ingestion	Python (Scrapper)	Initial extraction and loading logic
	Docker	Containerization for the load environment
	DuckDB	Local backup and staging database
	DLT (dltHub)	Load tool to push data from local to Cloud

Category	Component	Purpose
Batch Warehousing	Snowflake	**Bronze** (Raw), **Silver** (Cleaned), **Gold** (Star Schema) Layers
Batch Transformation	dbt (Data Build Tool)	Transformation logic: Bronze → Silver
	Python (Snowflake Connector)	Transformation logic: Silver → Gold
Batch Orchestration	Prefect	Orchestrating the entire batch pipeline flow
Streaming Trigger	AWS EventBridge	Scheduler/Trigger for serverless Lambda functions
Streaming Decoder/Producer	AWS Lambda	Decodes Protobuf and produces records to stream
Streaming Buffer	AWS Kinesis Data Streams	Real-time streaming buffer (Kafka equivalent)
Streaming Collector/Consumer	AWS Lambda	Consumes records from stream and writes to storage
Streaming Storage	AWS Timestream (InfluxDB)	Specialized Time-Series database for storage layer
Serving Layer	Grafana	Unified dashboard reading from Snowflake (Batch) and Timestream (Streaming)
Version Control	GitHub	Source code management

8. 📊 Final Dashboards

Batch Pipeline Dashboard



Streaming Pipeline Dashboard