STREAMLINE REAL TIME INGESTION AND SERVING WITH DLT AND ONLINE TABLES

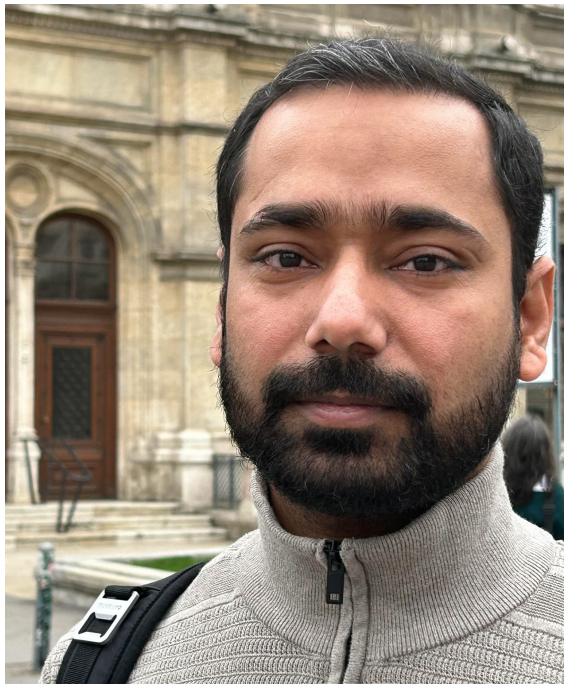
Magnus Johannesson

Solutions Architect and Data Engineer

- 30 years in the IT industry
- Developer, application architect, integration architect, enterprise architect
- 13 years in analytics
- 5 years working with Databricks
- Self-employed consultant since 2017

Sanjeev Kumar

Specialist Solution Architect



- 12 years in the IT industry
- Data Engineer
- With Databricks for 2 years

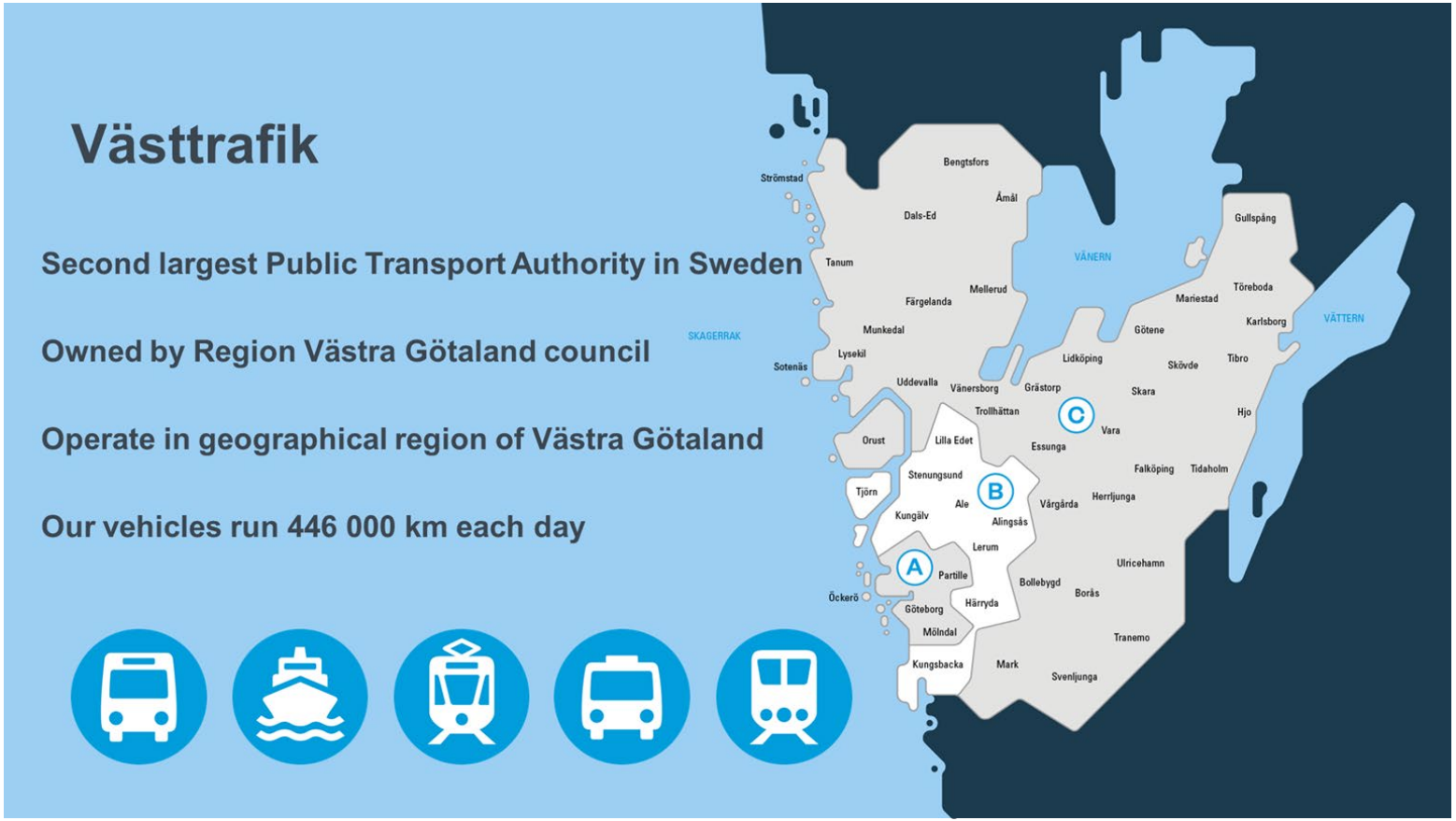
STREAMLINE REAL TIME INGESTION AND SERVING WITH DLT AND ONLINE TABLES

Agenda

- Västtrafik: Our business
- Use case and problem description
- Architecture
- Pipeline Implementation
- Future improvements
  - Online Store – Serving
- Demo
- Key takeaways

VÄSTTRAFIK

Västtrafik


Västtrafik
Second largest Public Transport Authority in Sweden
Owned by Region Västra Götaland council
Operate in geographical region of Västra Götaland
Our vehicles run 446 000 km each day
SKAGERRAK
VÄNERN
VÄTTERN
Strömstad
Bengtsfors
Åmål
Dals-Ed
Gullspång
Tanum
Mellerud
Färgelanda
Töreboda
Mariestad
Karlsborg
Munkedal
Götene
Lysekil
Lidköping
Skövde
Tibro
Sotenäs
Uddevalla
Vänersborg
Grästorp
Skara
Trollhättan
Hjo
Vara
Orust
Lilla Edet
Essunga
Falköping
Tidaholm
Stenungsund
Tjörn
Ale
Vårgårda
Herrljunga
Kungälv
Alingsås
Lerum
Ulricehamn
Partille
Bollebygd
Borås
Öckerö
Göteborg
Härryda
Mölndal
Tranemo
Kungsbacka
Mark
Svenljunga
A
B
C

Västtrafik

40 partner companies

470 employees

9,000 employees including our partners

SEK 10.8 billion turnover

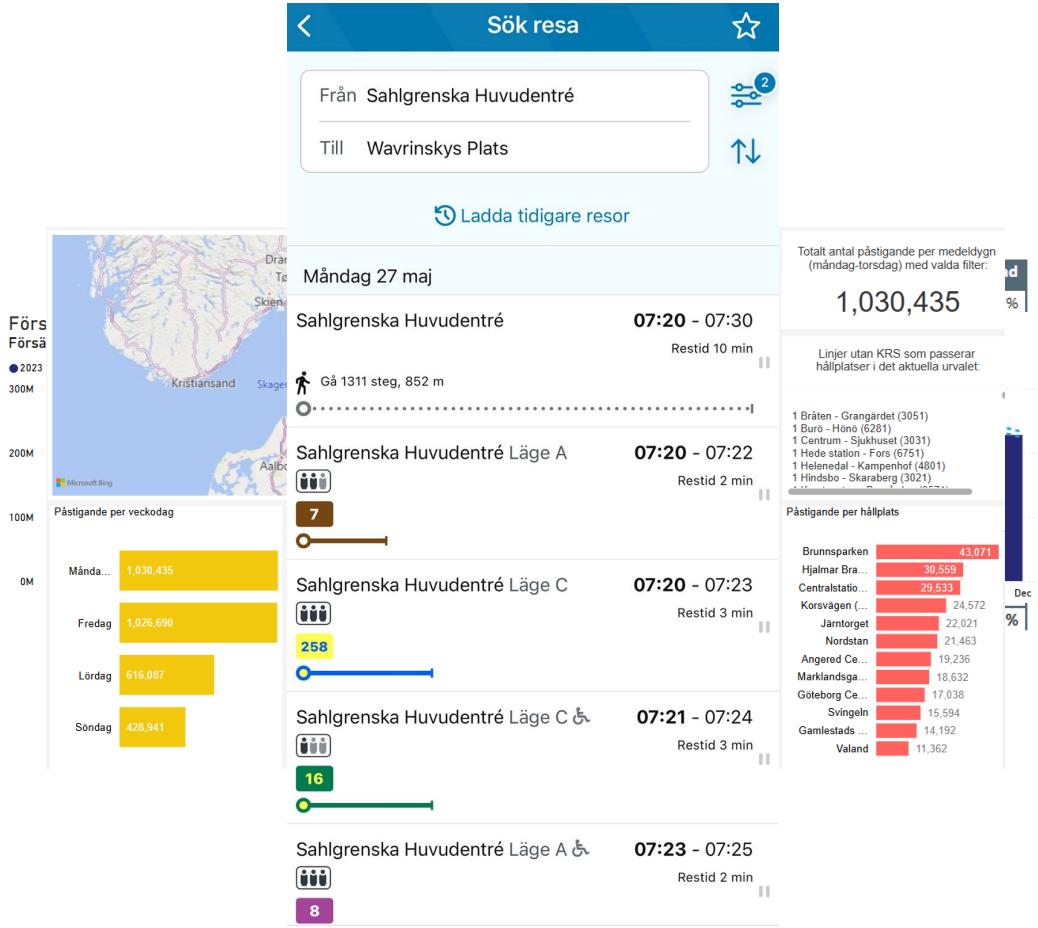
301 million trips, 2022

Pushing development together with **49** municipalities.

västtrafik

Some of our data

- Planned traffic data
- Real-time data
- Automatic passenger data
- Ticket data
  - Sales
  - Validations
  - Ticket inspections

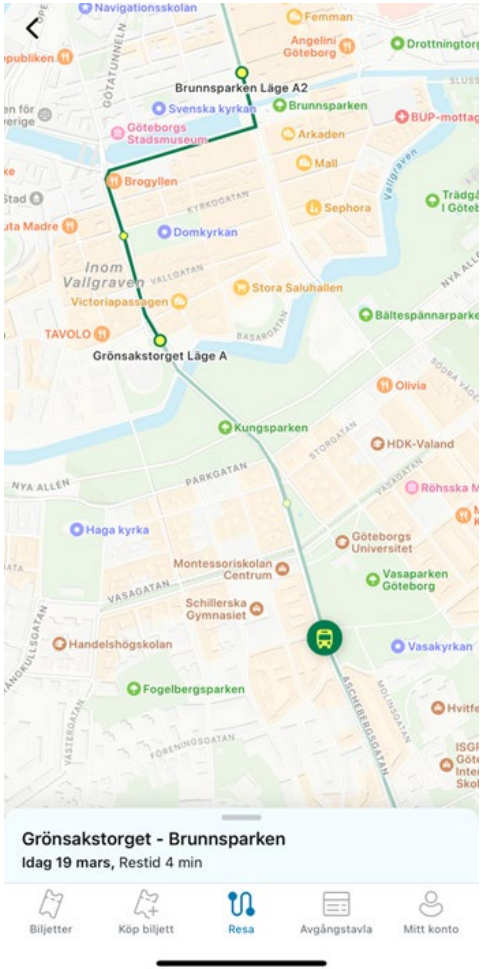


USE CASE

Use Case

Real-time position of Västtrafik's vehicles

- Trams & buses
  - GPS position, vehicle ID, line, stops, etc.
- Trains
  - Only GPS position and internal vehicle ID
- Proof of Concept (PoC)
  - Enrich using existing data
  - Test streaming data in Databricks
- Target latency 10 sec



Project Goals

Improve **Customer Satisfaction**

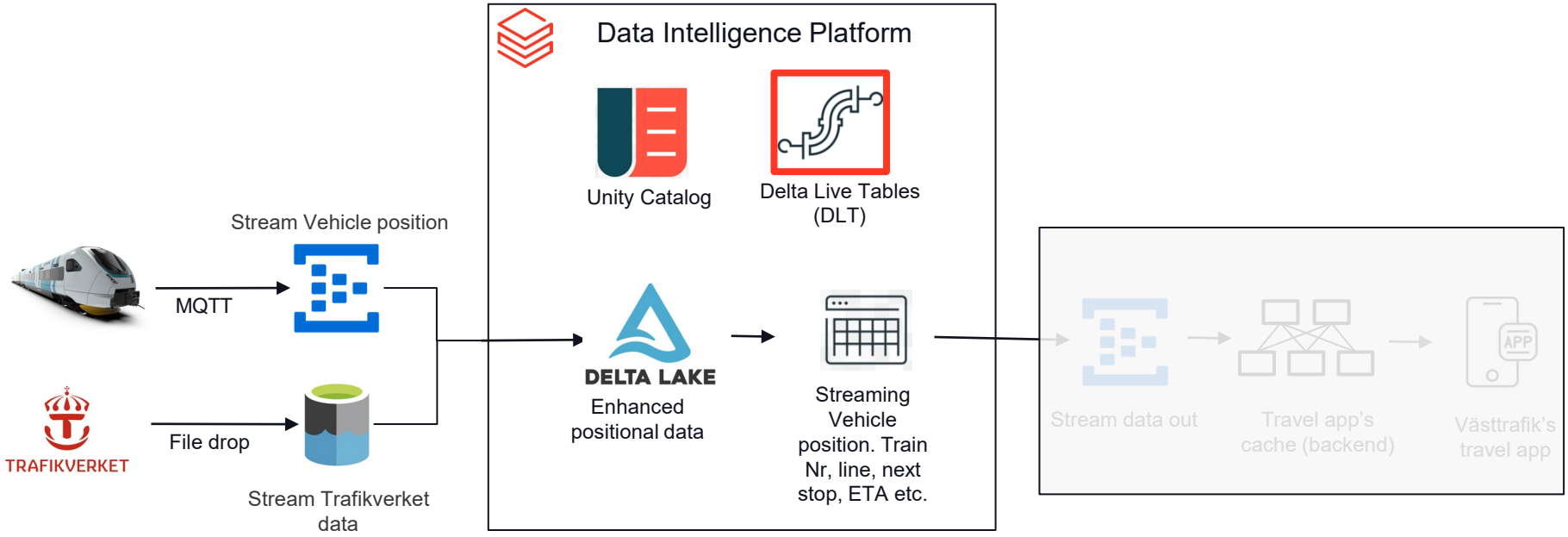
Ease of **development** and **Maintainability**

Stream data ≈ 10 sec latency

ARCHITECTURE

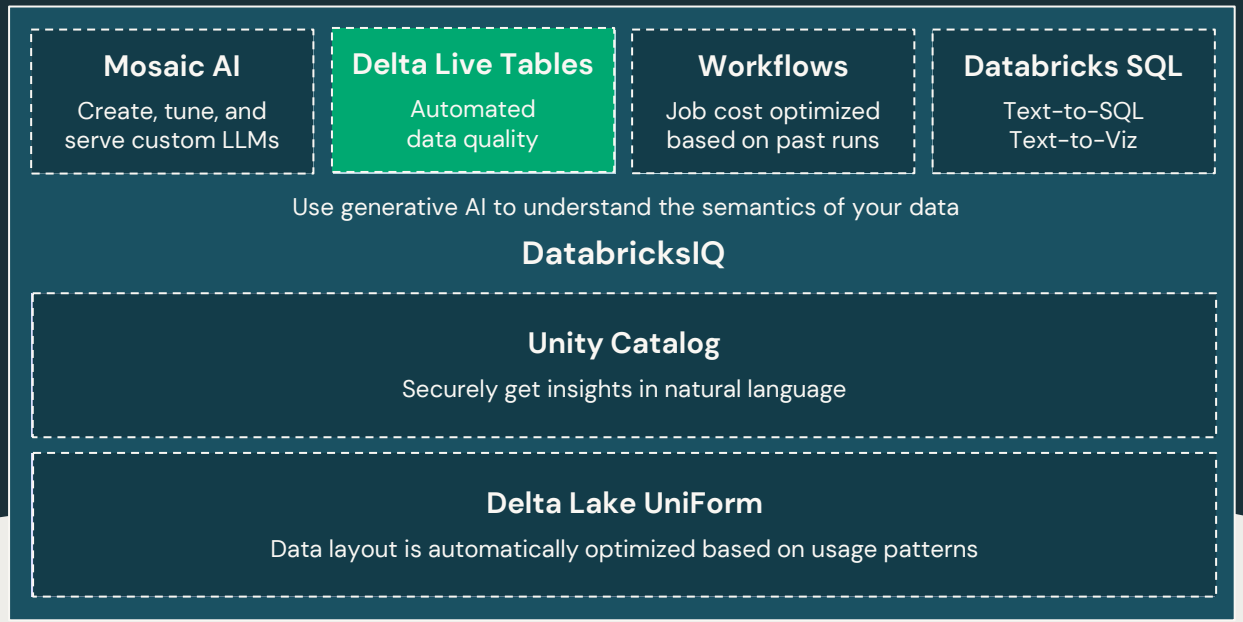
Architecture

Streaming with Delta Live Tables



WHY THIS ARCHITECTURE?

Databricks Data Intelligence Platform



Open Data Lake

All Raw Data
(Logs, Texts, Audio, Video, Images)

Delta Live Tables

The declarative way to do ETL on the lakehouse

```
CREATE STREAMING TABLE raw_data
AS SELECT *
FROM cloud_files ("/raw_data",
"json")

CREATE MATERIALIZED VIEW clean_data
AS SELECT …
FROM LIVE.raw_data
```

Accelerate ETL development

Declare **SQL or Python** and DLT automatically orchestrates the DAG, handles retries, changing data

Automatically manage your infrastructure

Automates complex tedious activities like **recovery, auto-scaling, and performance optimization**

Ensure high data quality

Deliver reliable data with built-in **quality controls, testing, monitoring, and enforcement**

Unify batch and streaming

Get the simplicity of SQL with freshness of streaming with one **unified API**

PIPELINE IMPLEMENTATION

Data Metrics

Streaming

Arrival & Departure ≈ **500 files/hr**

Streaming

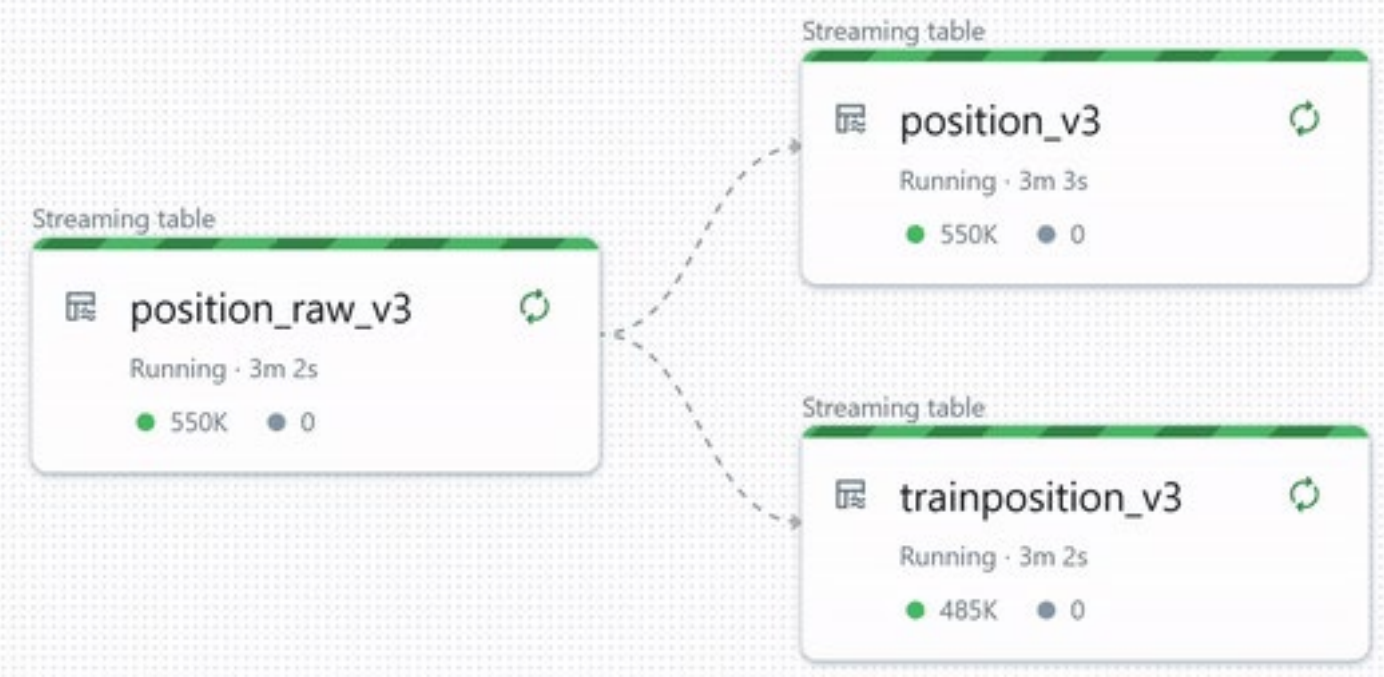
Positions ≈ **400 000 pos/hr**

Batch

Vehicle and Planned Traffic - **Daily**

DLT Streaming Pipeline 1

DLT Streaming Pipeline 2



Implementation

DLT Pipeline 1

- Train Arrival and Departure data is streamed as files.

DLT Pipeline 2

- Train Positions data is streamed from Event hub.
- Joined with Batch data i.e. Vehicle and Planned Traffic, and creates final Train Positions table.

DLT for both **Batch and Streaming** Data

Dependency management

Auto Optimizations

OPTIMIZATIONS

Top Optimization

Optimize for latency

- We started with ~ 2 minutes of latency.

Architecture

Reduced
Data Hops

Data layout

Enabled
Deletion Vectors

Cluster

Increased
Minimum Workers

Cluster

Enabled
PHOTON

- Final outcome was ~ 10 seconds latency.

Event hub Optimizations

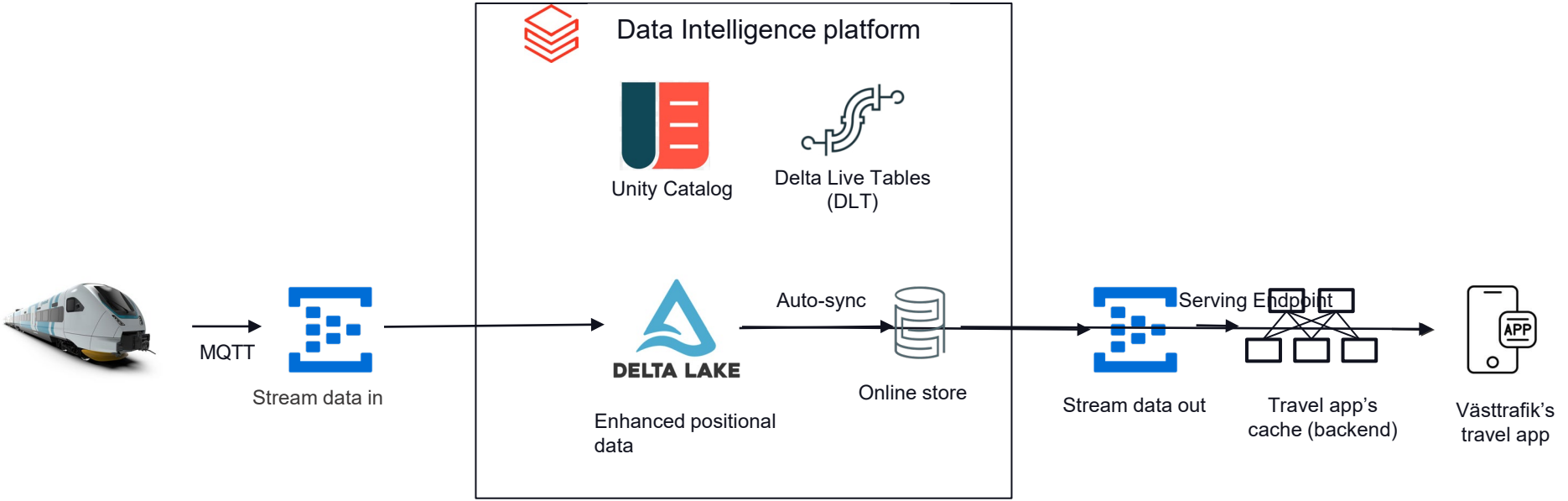
- Event hub: Kafka interface has better read speeds.
- Use `maxOffsetsPerTrigger` for rate limiting.

- Kafka fan-out : Set `minPartitions` to increase the read throughput if cluster has more compute.
- Partitioning
  - Too many : Increase the number of CPUs to match no of partitions
  - Too few : You may not be able to handle more load due to TU limits

Future Improvements

Proposed Architecture

Introducing Online Tables



Databricks Online Tables

Data Serving from Lakehouse

- Complexity
  - Online store infrastructure management
  - Data pipeline management
- Performance
  - Lookup latency
  - Data freshness
  - Dynamic scaling
- Cost
  - Cost of publishing data to online store

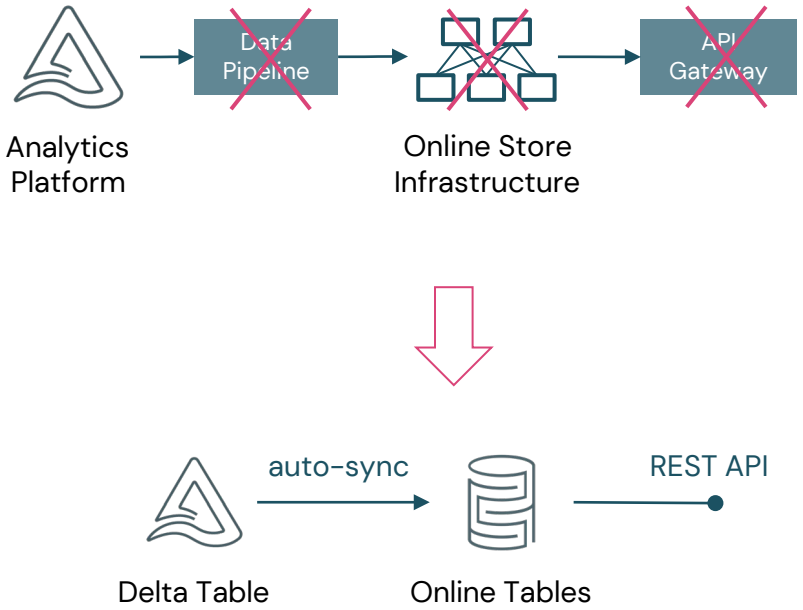
Databricks Online Tables

Simplicity

- Fully managed serverless online store
- Fully managed data synchronization pipeline
- Unified governance with Unity Catalog
- Zero-configuration Serving

Speed

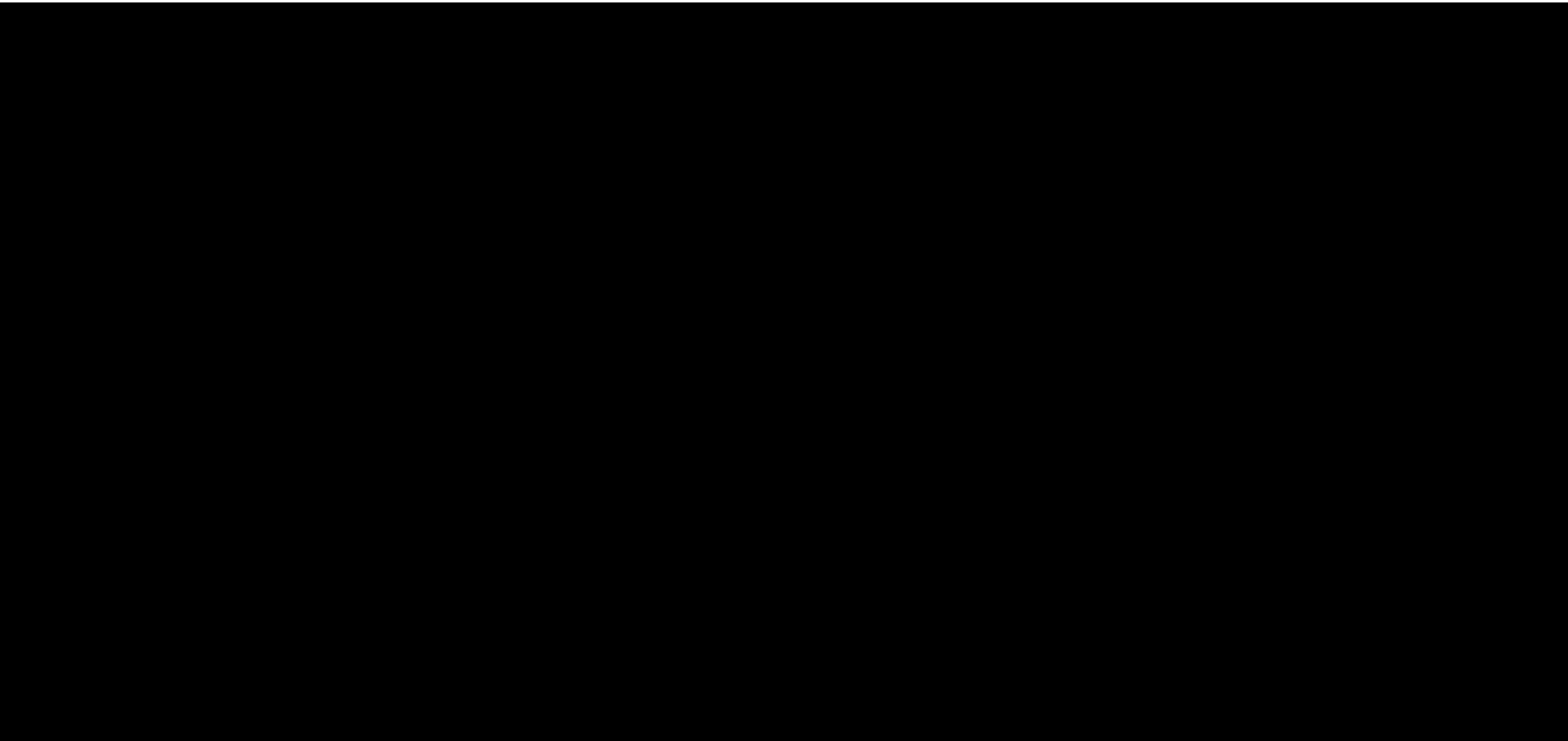
- Lookup latency (< 10 milliseconds)



DEMO

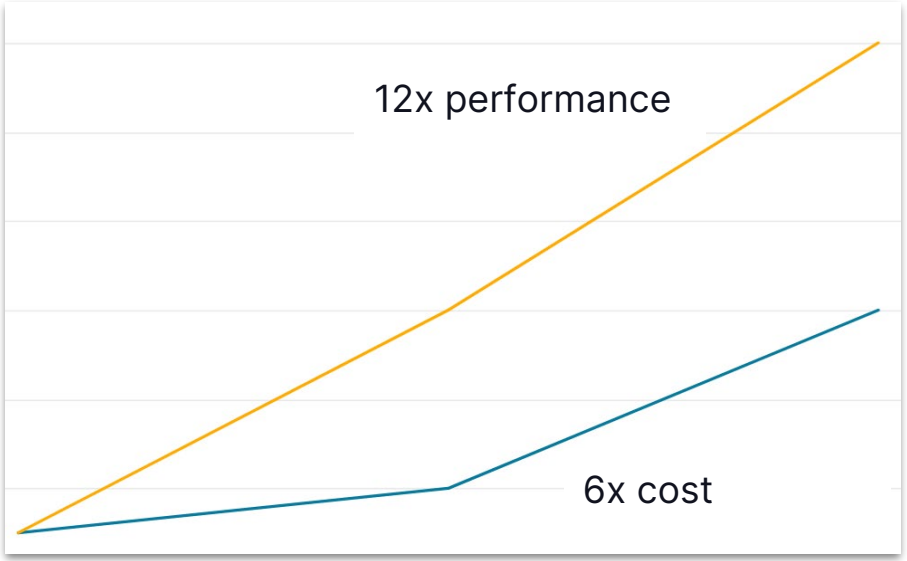
Serving with Online Table

Pre-recorded demo



Cost

- Compute cost affected due to multiple DLT pipelines and latency needs
- Storage cost affected due to continuous reads and writes
  - Depends on chosen redundancy
- Tune the cluster
  - Cheaper node type
  - Max worker nodes



Key takeaways

DLT for pipelining and **Online Tables** for Serving

Reached Desired latency **of 10s**

Ease of **Development** and **Maintainability**

QUESTIONS ?