MTA New York Buses Trips
End-to-End Data Engineering Pipeline

Batch & Real time data

Presented by:

- Ahmed Rabie
- Diaa Abdel Tawab
- Mohanad Hossam
- Mohamed Ibrahim
- Mohamed Zaki
- Adel Ashraf

Introduction

Business means for our project

- **Improving Operational Efficiency:** By analyzing metrics like average delays, on-time performance (OTP), delayed trips percentage, and total alerts, the project helps identify underperforming routes and boroughs. This enables better resource allocation, such as prioritizing maintenance or rerouting during peak hours.
- **Enhancing Passenger Experience:** Real-time dashboards track active vehicles, arrival rates and delays per borough or route, allowing for timely alerts and predictions. This reduces wait times and frustration for riders, potentially increasing ridership and satisfaction.
- **Cost and Resource Optimization:** Monitoring alerts and performance trends (e.g., delays peaking around certain hours in boroughs) can help minimize operational costs by preventing escalations and optimizing fleet usage.

Target users

- City planners and government officials
- MTA operators and management
- Public transport users
- Transport agencies

Data Source

Transitland website (Batch data and APIs)

MTA New York City Transit (MTA)

Onestop ID	`o-dr5r-nyct`
Agencies	MTA New York City Transit MTA Bus Company
Locations	United States of America / New York United States of America / New Jersey / Newark United States of America / New York / New York
Website	`http://www.mta.info`
ID Crosswalk	US National Transit Database (NTD) ID: `20008`

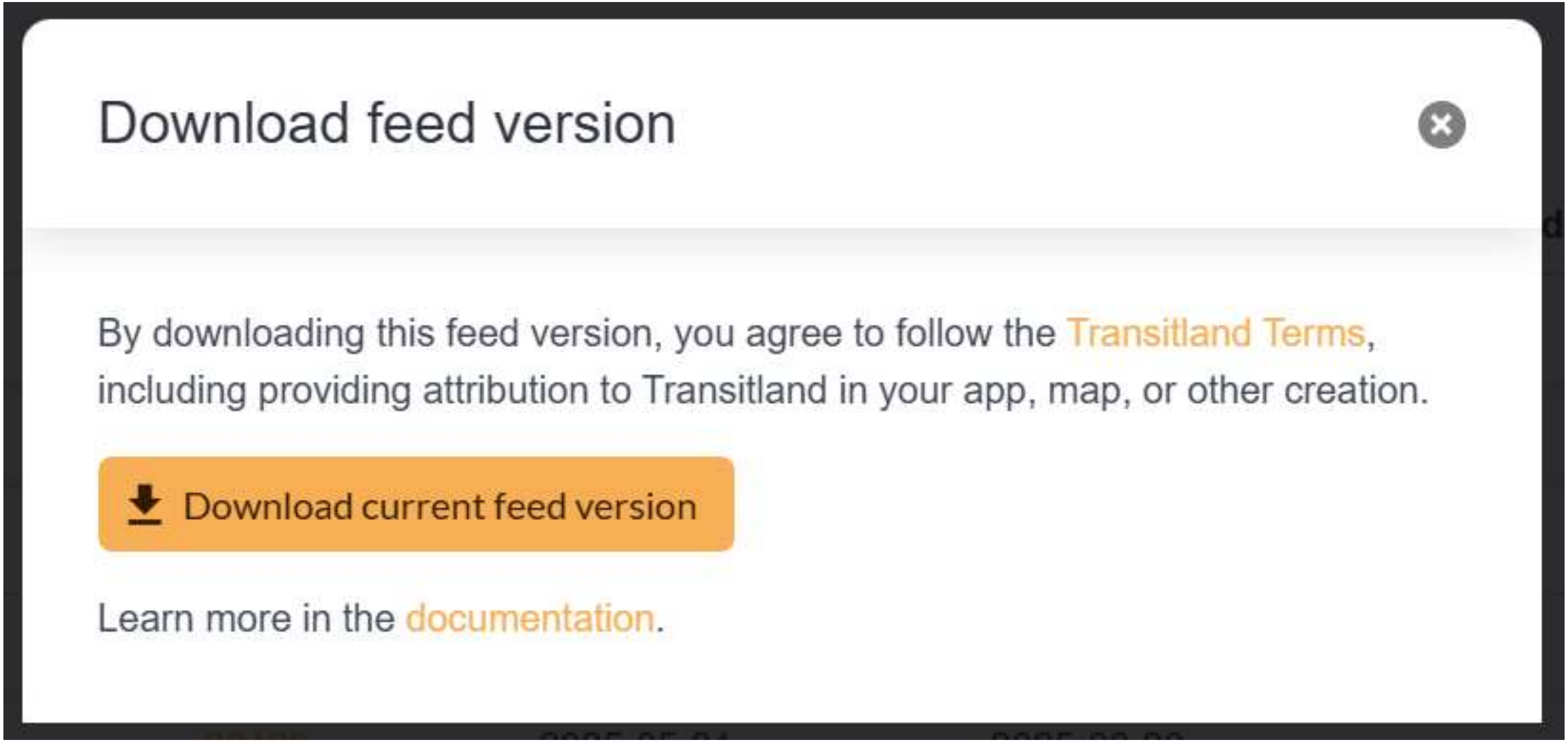
Source Feed(s)

Transitland fetches and imports data from one or more source feeds for each operator. Learn more about operators and source feeds in the Transitland documentation.

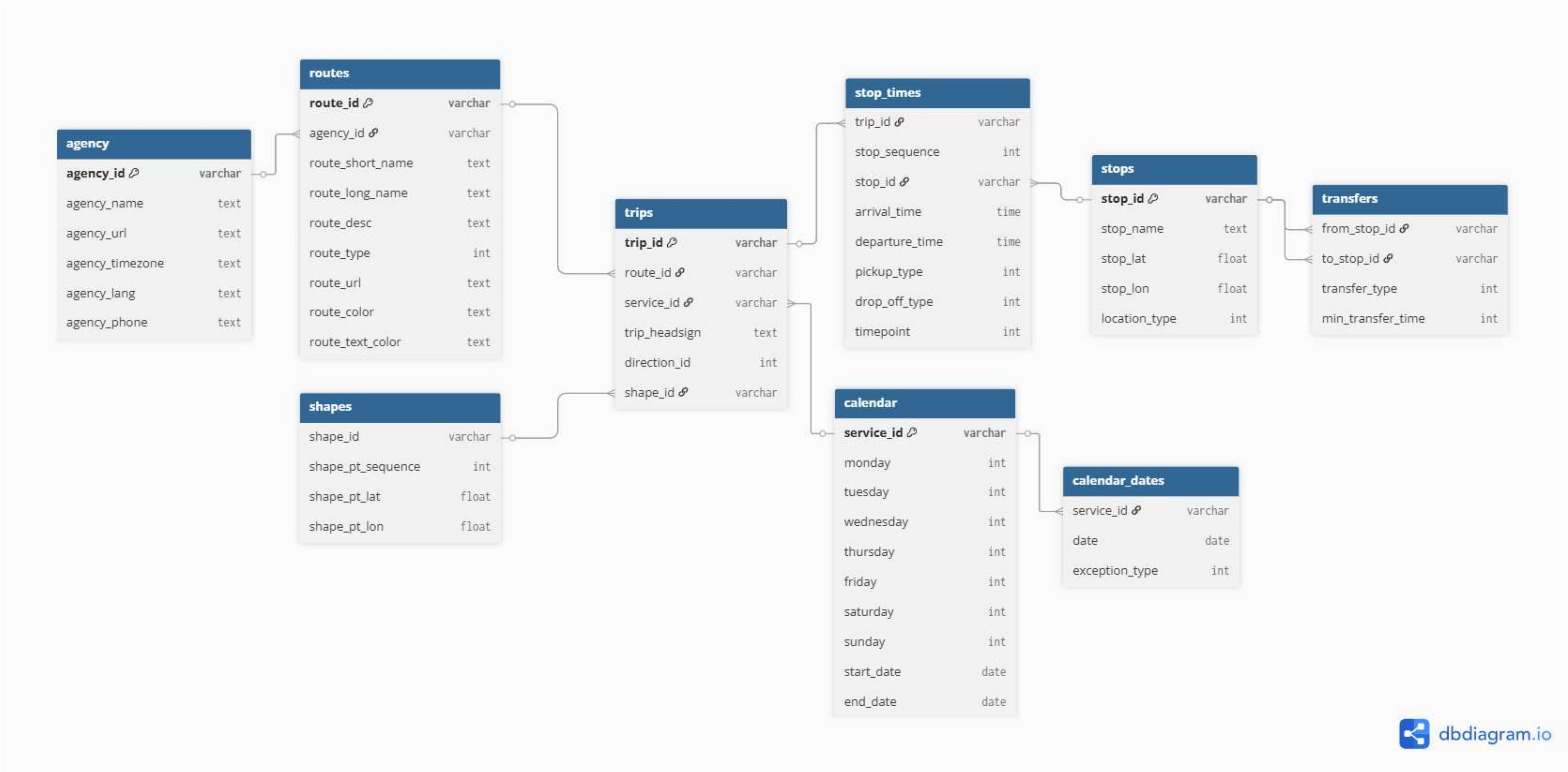
Source feed Onestop ID	Source spec	Association type	Matched GTFS agency	Links to view
f-dr5r-mtanyctbusstatenisland	GTFS	Associated Feed	✓ MTA New York City Transit	Feed Archived feed versions
f-dr72-mtanyctbusbronx	GTFS	Associated Feed	✓ MTA New York City Transit	Feed Archived feed versions
f-dr5r-mtanyctbusmanhattan	GTFS	Associated Feed	✓ MTA New York City Transit	Feed Archived feed versions
f-dr5x-mtanyctbusqueens	GTFS	Associated Feed	✓ MTA New York City Transit	Feed Archived feed versions
f-dr5r-mtabc	GTFS	Associated Feed	✓ MTA Bus Company	Feed Archived feed versions
f-dr5r-mtanyctbusbrooklyn	GTFS	Associated Feed	✓ MTA New York City Transit	Feed Archived feed versions
f-dr5r-nyctsubway	GTFS	Associated Feed	✓ MTA New York City Transit	Feed Archived feed versions
f-dr5r-mtanewyorkcitytransit	GTFS	Associated Feed		Feed Archived feed versions
f-mta~nyc~rt~alerts	GTFS Realtime	Associated Feed		Feed
f-mta~nyc~rt~subway~1~2~3~4~5~6~7	GTFS Realtime	Associated Feed		Feed
f-mta~nyc~rt~subway~a~c~e	GTFS Realtime	Associated Feed		Feed

Feed versions

Added	SHA1	Earliest date	Latest date	Imported	Active	Download
2025-07-10 (1 month ago)	dc9b1c…	2025-06-28	2025-08-30	✓✓	✓	
2025-07-03 (2 months ago)	92a467…	2025-06-28	2025-08-30			
2025-06-30 (2 months ago)	6db867…	2025-06-28	2025-08-30			
2025-06-25 (2 months ago)	c86829…	2025-06-28	2025-08-30			
2025-06-11 (2 months ago)	c96466…	2025-05-31	2025-06-28			
2025-05-29 (3 months ago)	1d5aea…	2025-05-31	2025-06-28			
2025-05-15 (3 months ago)	c7c997…	2025-01-29	2025-06-28			
2025-03-27 (5 months ago)	c90cd4…	2025-03-29	2025-06-28			
2025-02-15 (6 months ago)	c0cb9f…	2025-02-09	2025-03-29			


Download feed version
By downloading this feed version, you agree to follow the Transitland Terms, including providing attribution to Transitland in your app, map, or other creation.
Download current feed version
Learn more in the documentation.

Batch files



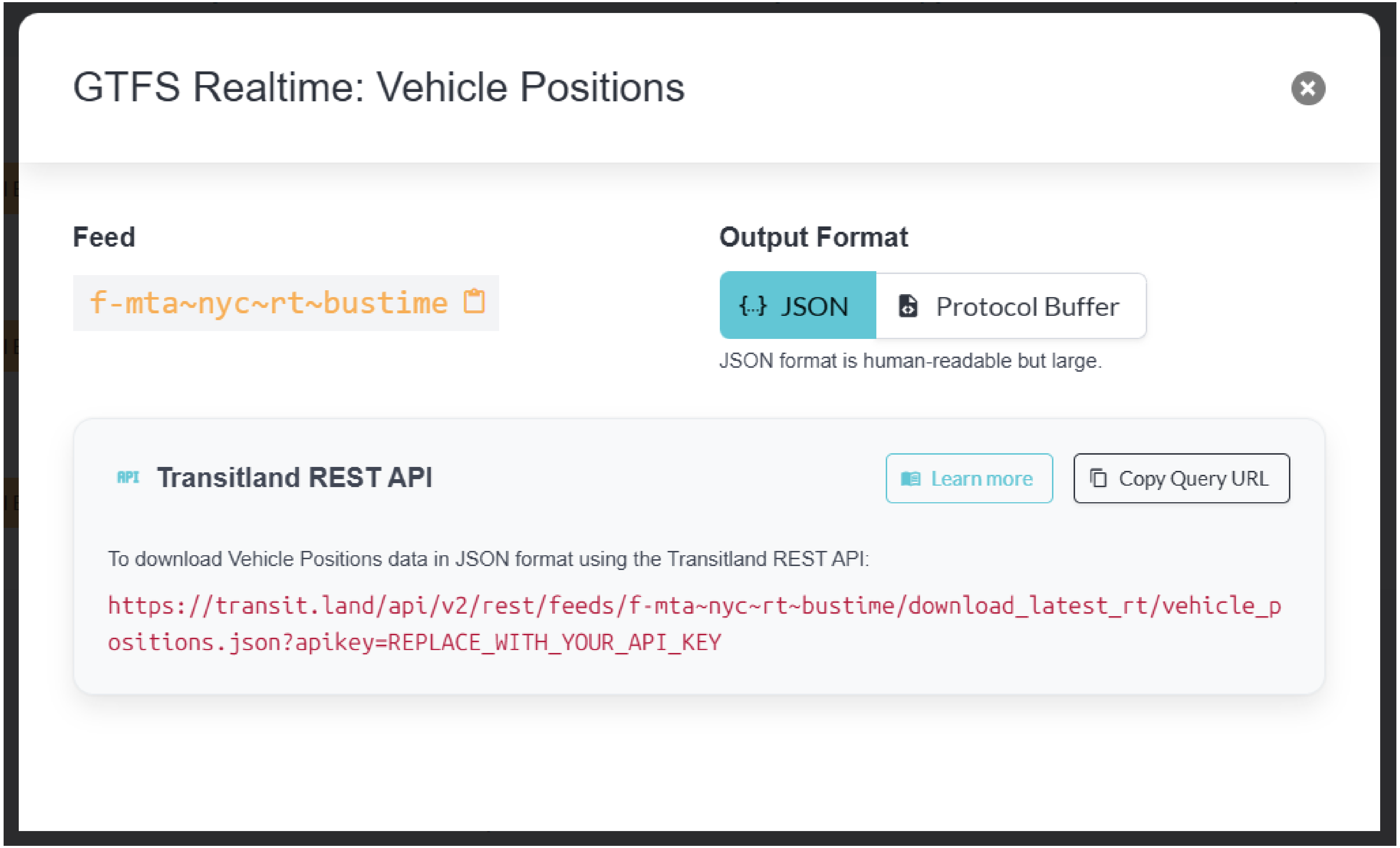
routes
route_id varchar
agency_id varchar
route_short_name text
route_long_name text
route_desc text
route_type int
route_url text
route_color text
route_text_color text
agency
agency_id varchar
agency_name text
agency_url text
agency_timezone text
agency_lang text
agency_phone text
trips
trip_id varchar
route_id varchar
service_id varchar
trip_headsign text
direction_id int
shape_id varchar
shapes
shape_id varchar
shape_pt_sequence int
shape_pt_lat float
shape_pt_lon float
stop_times
trip_id varchar
stop_sequence int
stop_id varchar
arrival_time time
departure_time time
pickup_type int
drop_off_type int
timepoint int
stops
stop_id varchar
stop_name text
stop_lat float
stop_lon float
location_type int
transfers
from_stop_id varchar
to_stop_id varchar
transfer_type int
min_transfer_time int
calendar
service_id varchar
monday int
tuesday int
wednesday int
thursday int
friday int
saturday int
sunday int
start_date date
end_date date
calendar_dates
service_id varchar
date date
exception_type int
dbdiagram.io

Streaming APIs

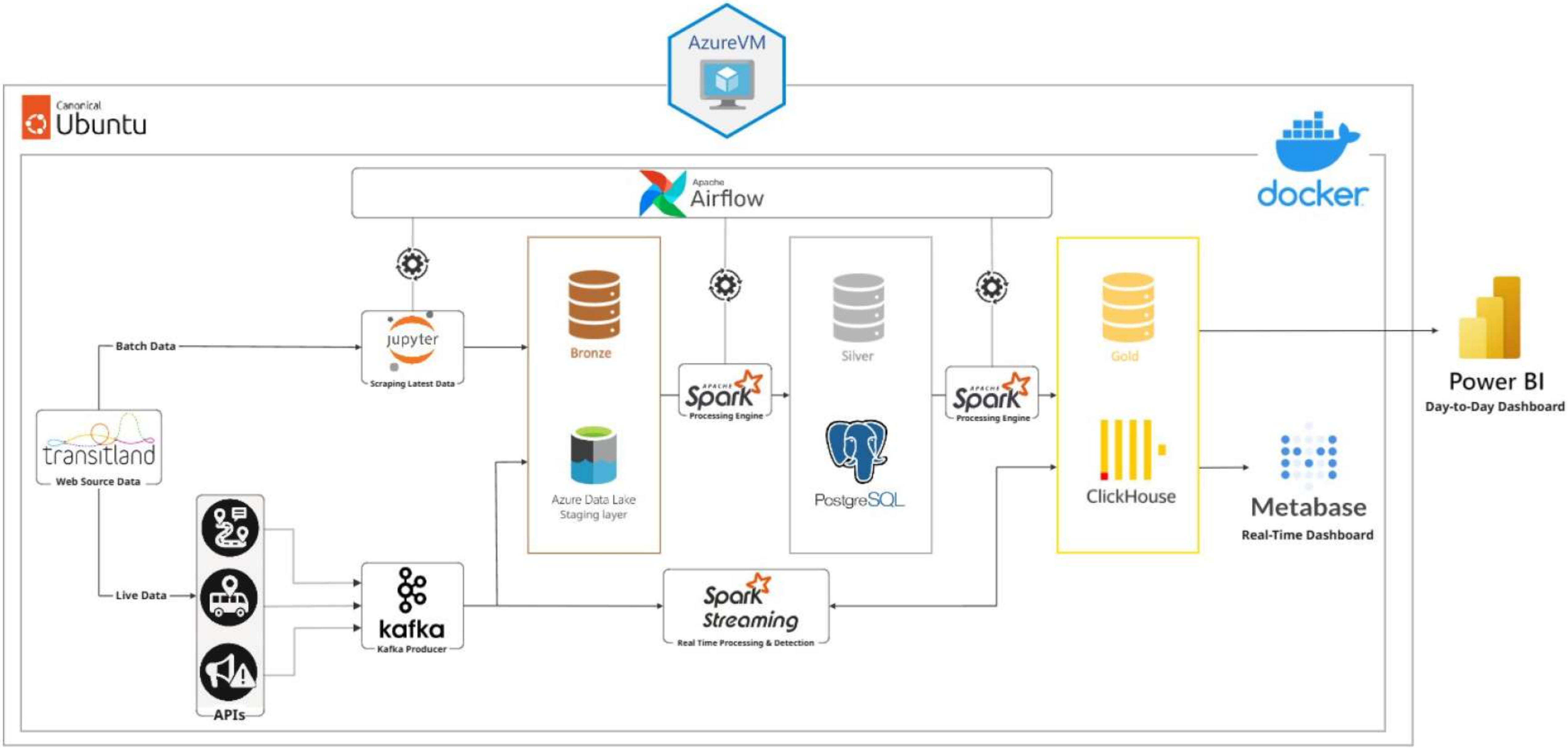
Vehicle Position API: actual GPS positions of buses in motion

Trip Updates Api: estimated delays, estimated arrival/departure per stop for a running trip

Alerts API: service disruptions (road closures, diversions, etc.)


GTFS Realtime: Vehicle Positions
Feed
f-mta~nyc~rt~bustime
Output Format
{...} JSON
Protocol Buffer
JSON format is human-readable but large.
API
Transitland REST API
Learn more
Copy Query URL
To download Vehicle Positions data in JSON format using the Transitland REST API:
https://transit.land/api/v2/rest/feeds/f-mta~nyc~rt~bustime/download_latest_rt/vehicle_positions.json?apikey=REPLACE_WITH_YOUR_API_KEY

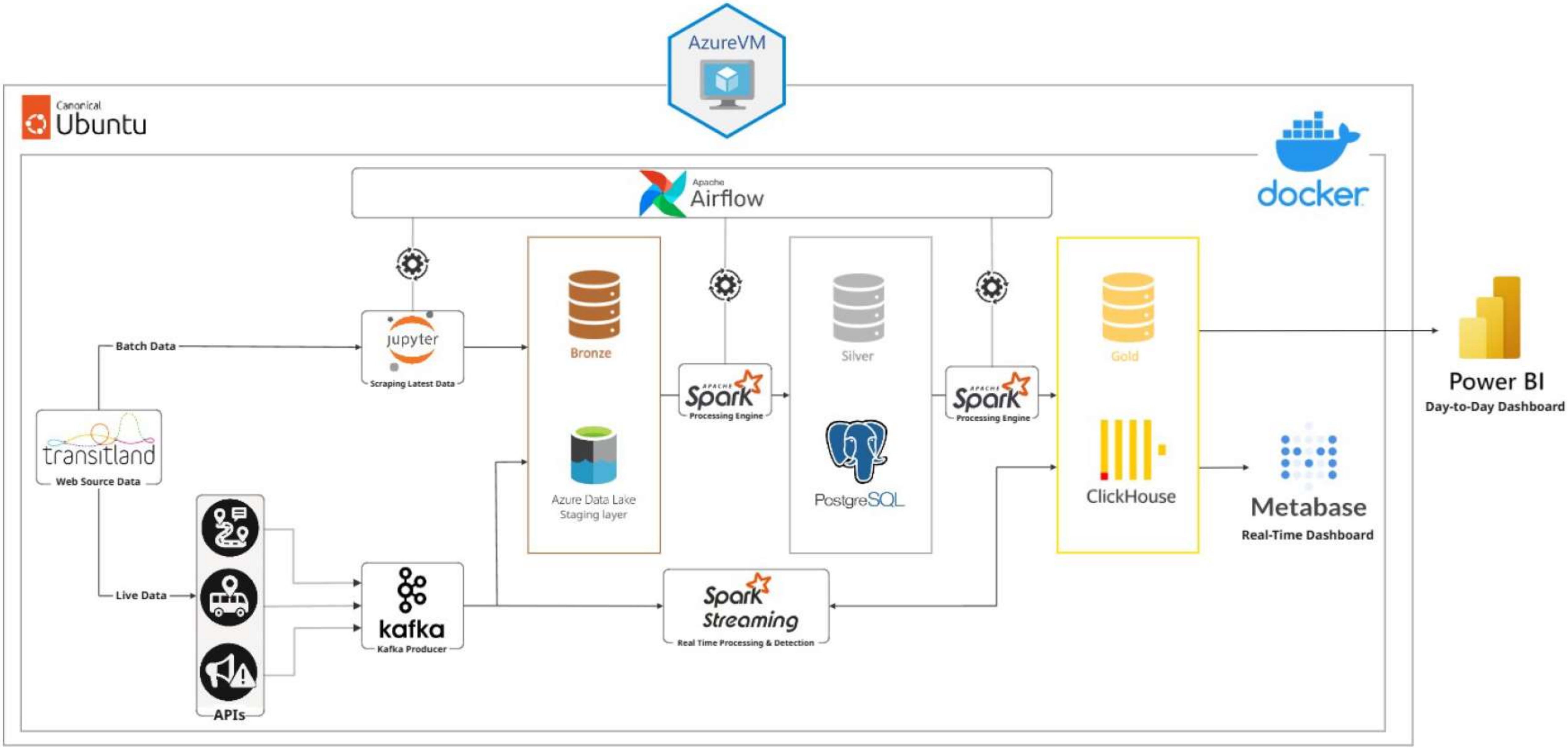
Architecture


AzureVM
Canonical
Ubuntu
docker
Apache
Airflow
Batch Data
jupyter
Scraping Latest Data
Bronze
Azure Data Lake
Staging layer
Spark
Processing Engine
Silver
PostgreSQL
Spark
Processing Engine
Gold
ClickHouse
Power BI
Day-to-Day Dashboard
Metabase
Real-Time Dashboard
transitland
Web Source Data
Live Data
APIs
kafka
Kafka Producer
Spark
Streaming
Real Time Processing & Detection

Azure VM is the cloud-based environment

Docker compose file connects all tools together on the VM

Batch data Path



AzureVM
Canonical Ubuntu
docker
Apache Airflow
Batch Data
Jupyter
Scraping Latest Data
transitland
Web Source Data
Live Data
APIs
kafka
Kafka Producer
Bronze
Azure Data Lake
Staging layer
Apache Spark
Processing Engine
Silver
PostgreSQL
Apache Spark
Processing Engine
Gold
ClickHouse
Spark Streaming
Real Time Processing & Detection
Power BI
Day-to-Day Dashboard
Metabase
Real-Time Dashboard

Batch Workflow: Airflow as the Engine

The engine behind all our batch workflows is Airflow, which runs on a fixed schedule — every day at 12:00 AM (NYC time).

Step 1: Data Extraction

We extract data from our website using **web scraping**. The reason we had to use scraping is that there's no API for batch processing — only a download button is available. Since we wanted to **automate** the process, web scraping was the only option.

On the website, there's a table for each NYC region. Every day, we scrape the website to check the **latest available update**.

Web scraping

Feed versions

Added	SHA1	Earliest date	Latest date	Imported	Active	Download
2025-07-10 (1 month ago)	dc9b1c…	2025-06-28	2025-08-30			
2025-07-03 (2 months ago)	92a467…	2025-06-28	2025-08-30			
2025-06-30 (2 months ago)	6db867…	2025-06-28	2025-08-30			
2025-06-25 (2 months ago)	c86829…	2025-06-28	2025-08-30			
2025-06-11 (2 months ago)	c96466…	2025-05-31	2025-06-28			
2025-05-29 (3 months ago)	1d5aea…	2025-05-31	2025-06-28			
2025-05-15 (3 months ago)	c7c997…	2025-01-29	2025-06-28			
2025-03-27 (5 months ago)	c90cd4…	2025-03-29	2025-06-28			
2025-02-15 (6 months ago)	c0cb9f…	2025-02-09	2025-03-29			
2025-01-01 (8 months ago)	25aedd…	2025-01-04	2025-03-29			
2024-12-09 (8 months ago)	348ac4…	2024-08-31	2025-01-04			
2024-08-29 (12 months ago)	dba68c…	2024-08-31	2025-01-04			
2024-06-27 (1 year ago)	64d403…	2024-06-29	2024-08-31			
2024-03-27 (over 1 year ago)	d91feb…	2024-03-30	2024-06-29			

XPath Tester

XPath ?

```
//table[@class='table is-striped is-fullwidth']/tbody/tr[1]
```

Text in Element

2025-07-10 (1 month ago)dc9b1c... 2025-06-282025-08-30Successfully imported

Need to download a website? Use Website Downloader

Rate Us:

Use Ctrl-Shift-X to toggle extension

Share sup@xpath-tester.help Restore account

Importance of Last Update Date

The most important part here is the **last update timestamp**, because it determines whether the batch pipeline should run or be skipped.

How this works:

We maintain a **JSON file** that stores the last update date for each region.

- Every day, we compare the website's latest update with the date in our JSON file.

If they **match**, it means there's no new update → we skip the batch pipeline except for the final task

Airflow Dag (without updates on data)

Batch Workflow: Data Processing

If they don't match, the workflow proceeds:

01

Scrape and Download

We scrape the site, trigger the **download button**, and download the data (as a **ZIP file**).

02

Unzip and Convert

We unzip it, and the extracted file comes in **.txt format**, which we then convert to **.CSV**.

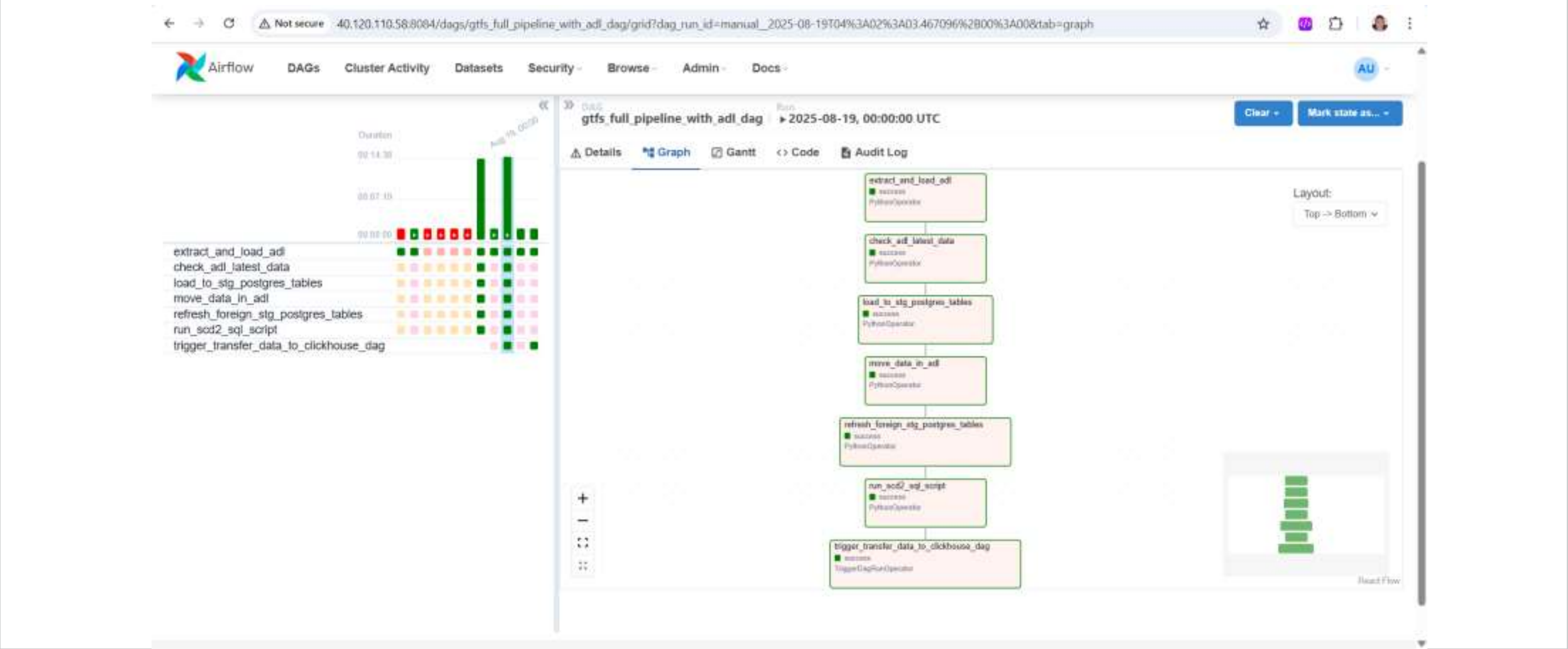
03

Upload to Azure Data Lake

Finally, we upload the entire file (all generated CSVs) to **Azure Data Lake**.

Why upload the whole file? → Because the file naming convention is very important: it includes the **region name, company name, and the upload date** from the website. All this logic is implemented in a **Jupyter notebook**, which Airflow is responsible for running.

Airflow Dag (with updates on data)



Step 2: Processing in Spark

Now the raw data is in **Azure Data Lake**. The next step: Airflow triggers a **Spark Jupyter notebook**, which extracts the data from Azure Data Lake, applies transformations (adding extra columns), and then loads it into the **Staging Database** hosted in **PostgreSQL**.

At this stage, only the **newly downloaded data** (i.e., data that is not already present in the historical database) is loaded into the staging area. The handling of the historical database

Implementing **SCD Type 2** (Slowly Changing Dimension)

A Data Warehouse isn't just about the latest snapshot; it's a historical archive. To truly understand trends and changes, we need to track how data evolves over time.

How We Apply SCD2 ?

We add three control columns to each batch record:

- `start_date`: When the record became active.
- `end_date`: When the record ceased to be current.
- `is_current`: A flag (True/False) indicating the latest version.

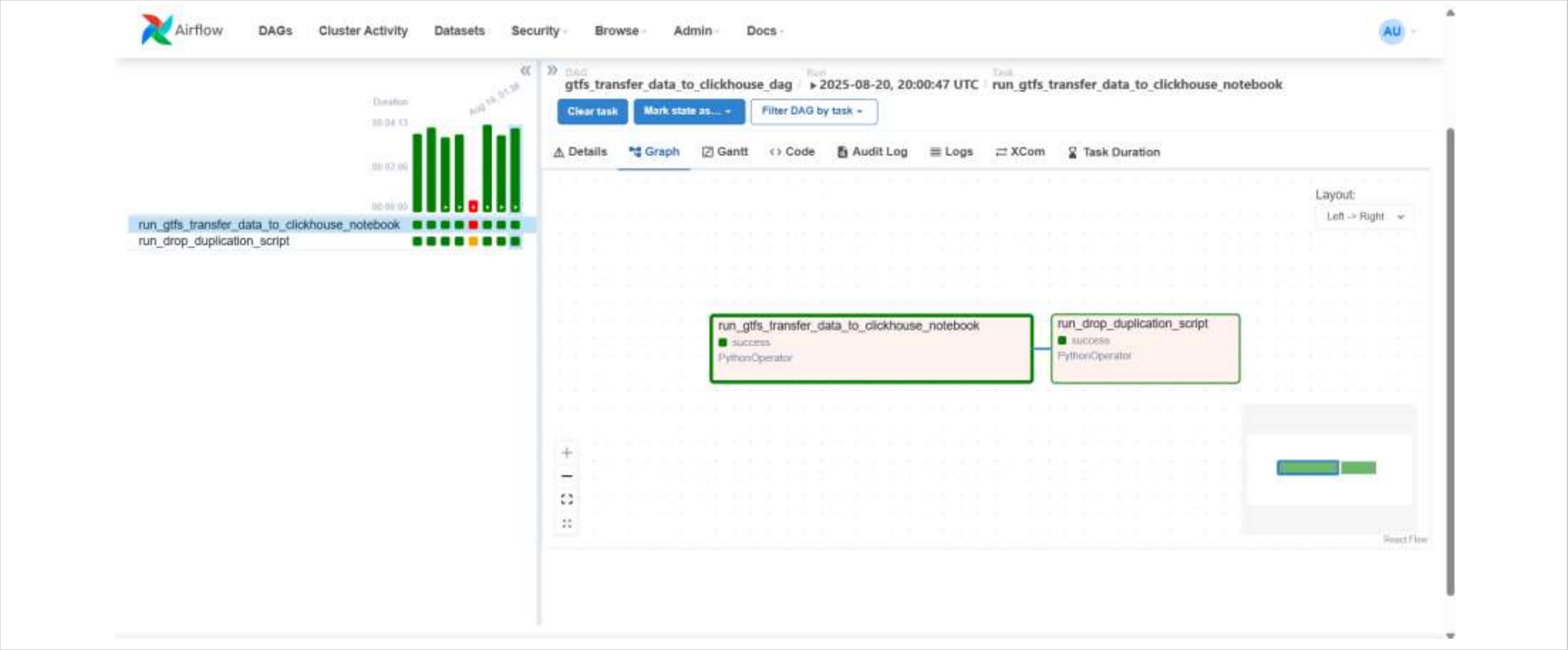
When a change happens:

- Expire old record (end_date + is_current = False)
- Insert new record (start_date, is_current = True)

Key Benefit

This approach enables precise historical analysis, allowing us to reconstruct data states at any past moment while still easily identifying the current valid record.

Transfer data from Postgres to ClickHouse



Daily Batch Data Transfer to ClickHouse

Moving large volumes of historical data repeatedly can strain systems.
Our approach focuses on efficiency and scalability by selectively transferring daily changes.

Why Daily Batches?

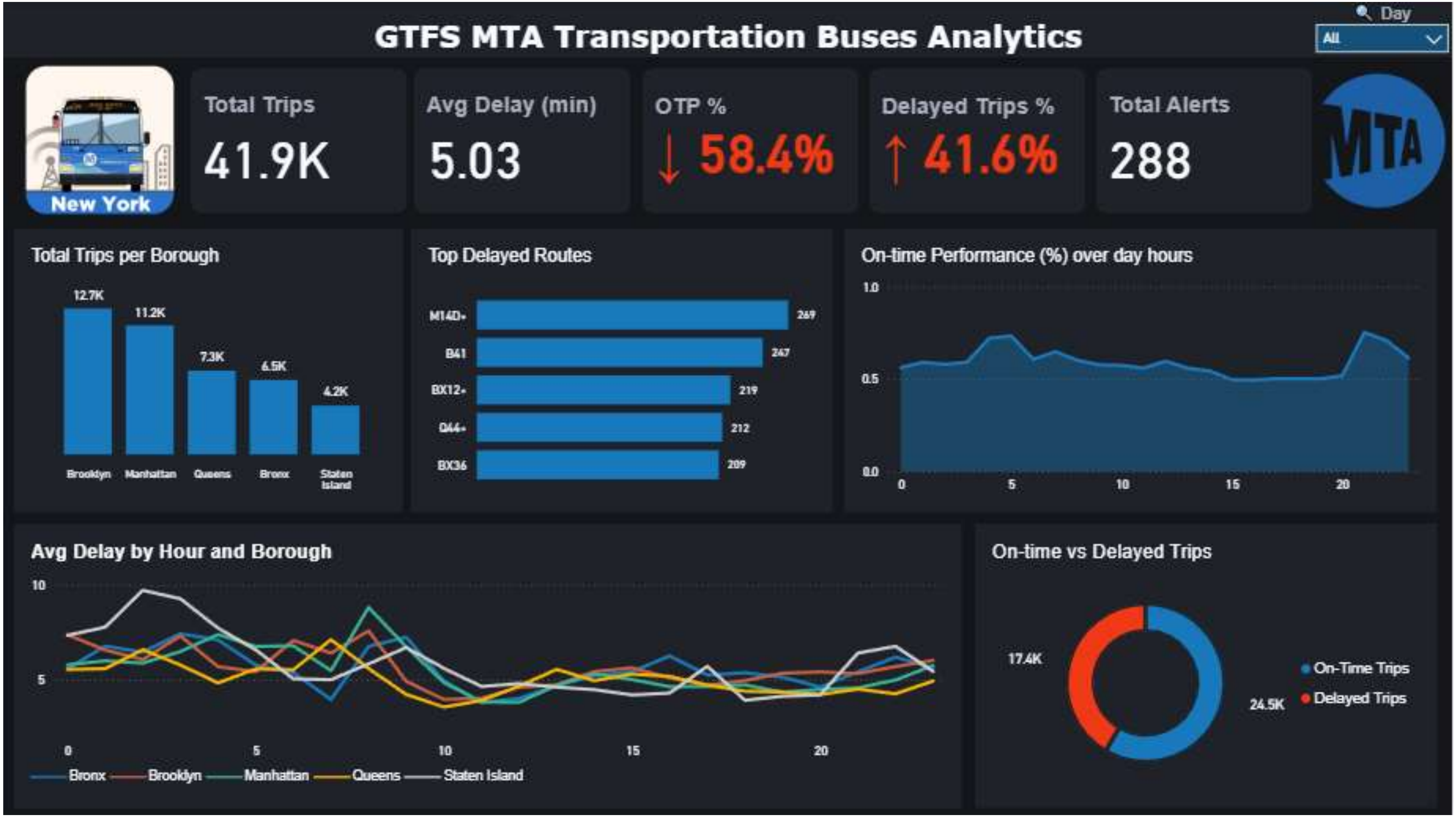
Instead of reloading our entire historical warehouse (Postgres) every day,
we only transfer the **daily batch data** for the current day to ClickHouse.

How it Helps ?

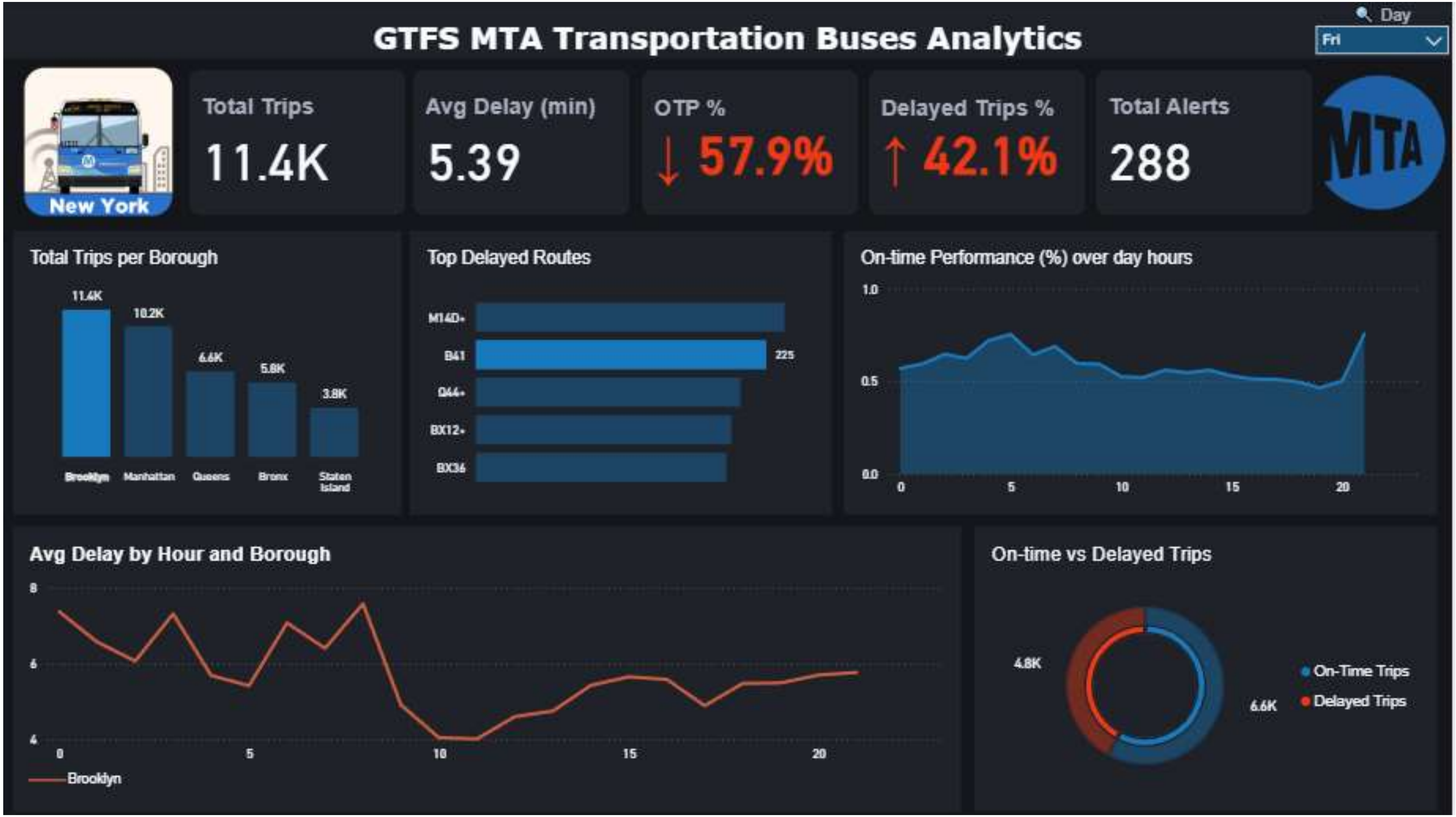
- **Optimized Spark Load:** Processes only today's data, not the full history.
- **Faster Comparison:** Quick reconciliation with real-time API data
- **Reduced Overhead:** Lower storage and processing costs in ClickHouse.

Day-to-Day Dashboard

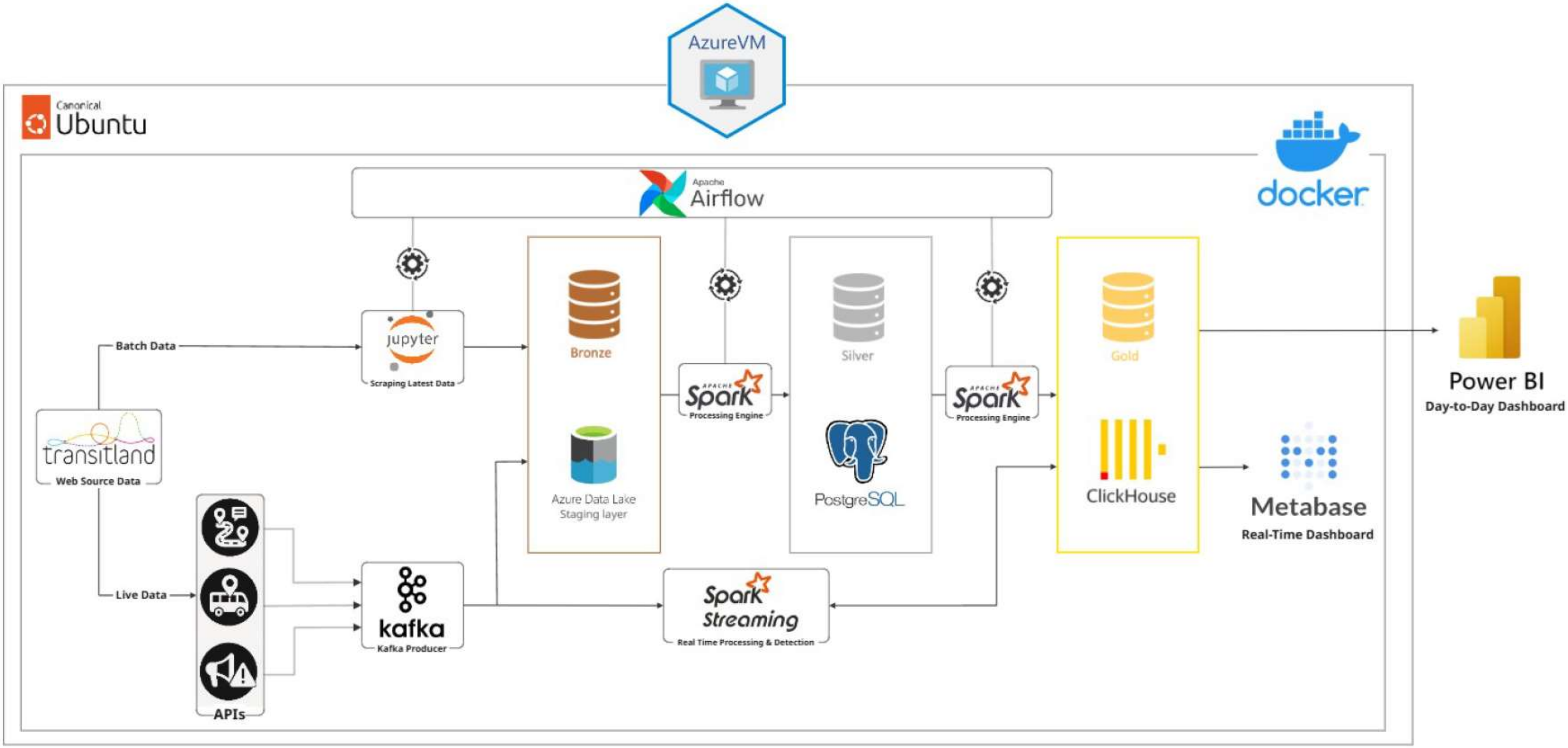
Day-to-Day Dashboard



Dashboard Filters



Streaming data Path


AzureVM
Canonical
Ubuntu
docker
Apache
Airflow
Batch Data
jupyter
Scraping Latest Data
transitland
Web Source Data
Live Data
APIs
kafka
Kafka Producer
Bronze
Azure Data Lake
Staging layer
APACHE
Spark
Processing Engine
Silver
PostgreSQL
APACHE
Spark
Processing Engine
Gold
ClickHouse
Spark
Streaming
Real Time Processing & Detection
Power BI
Day-to-Day Dashboard
Metabase
Real-Time Dashboard

Streaming Overview

- **Objective**

Build a comprehensive real-time data streaming pipeline for transit data processing using NYC MTA GTFS Realtime feeds.

- **Key Features**

Continuous ingestion of live transit data with real-time transformation, processing, and analytics-ready storage capabilities

- **Tech Stack**

Apache Kafka for message streaming, Spark Structured Streaming for processing, ClickHouse database for storage, Docker containerization, Metabase, Azure VM

Data Sources

Feed Type	Description	Frequency	Payload Size	Purpose
Trip Updates	Expected Arrival, Departure and Delays	~60 seconds	8-15 MB	Real-time monitoring
Vehicle Positions	GPS location updates	~60 seconds	0.7-1 MB	Geospatial tracking
Service Alerts	Disruptions and notifications	~60 seconds	~0.05 MB	Operational insights

GTFS Realtime Analysis

The GTFS Realtime APIs from TransitLand provide comprehensive transit data across three main feeds. Trip Updates deliver real-time schedule expectations. Vehicle Positions track GPS coordinates of active buses enabling geospatial analytics and route monitoring. Service Alerts communicate disruptions and operational changes with lightweight payloads but critical operational importance for transit management systems.

Kafka Setup

Topic Configuration

- gtfs-trip-updates topic with 1 partitions
- gtfs-vehicle-positions with 1 partitions
- gtfs-alerts with 1 partition

Partition Reasoning

- Data is processed quickly
- Only 1 message per topic every min
- No need for parallelism

Configuration

- Replication factor: 1
- Retention: 7 days
- Compression: snappy
- Max message: 25MB

Kafka Producer

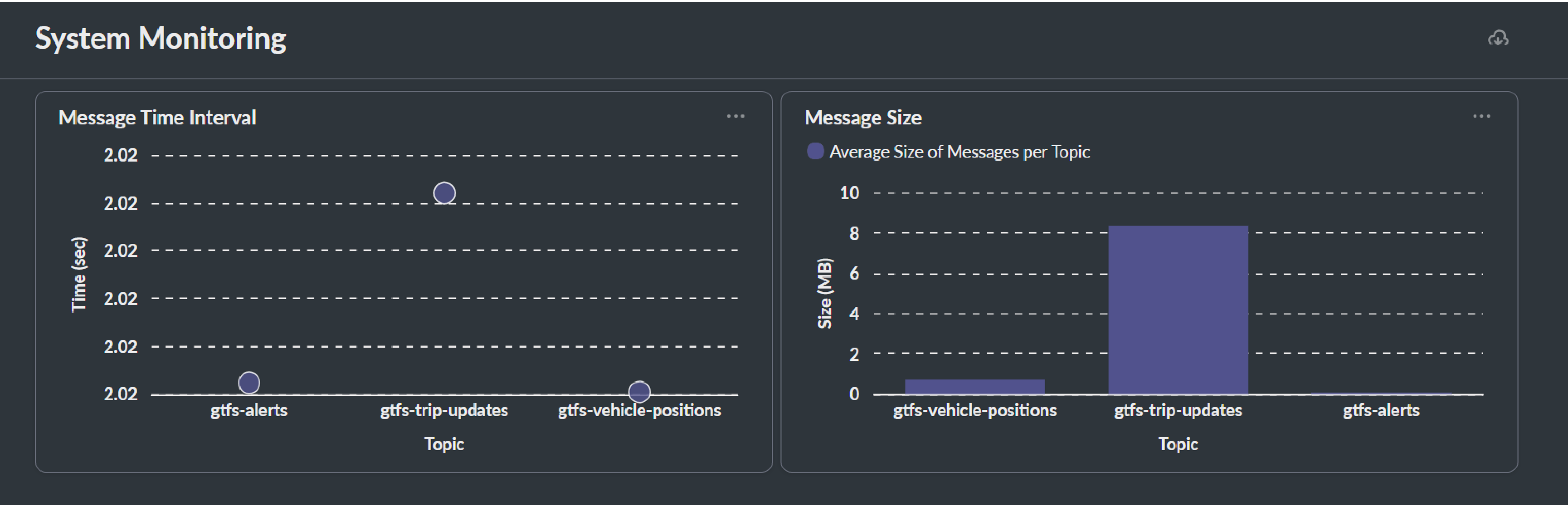
Core Responsibilities

Fetch API data every minute with automatic API key rotation to avoid rate limits.

Metrics Collection

Push JSON payloads to Kafka topics while logging update intervals and payload sizes to ClickHouse for monitoring.

Real-Time Metrics



Monitoring payload trends and intervals between new data.

Spark Streaming

Architecture Benefits

Spark Structured Streaming provides high-throughput, low-latency pipeline processing with built-in checkpointing and comprehensive fault tolerance mechanisms for reliable data processing

- High-throughput data processing capabilities
- Built-in checkpointing for fault tolerance
- Low-latency streaming pipeline architecture

Processing Flow

Comprehensive data transformation pipeline that consumes from Kafka topics, parses JSON payloads, applies business transformations, and writes processed data to ClickHouse storage

- Kafka consumption with offset management
- JSON parsing and schema validation
- Real-time transformation and ClickHouse writes

Alerts, Trip Updates, and Vehicle Positions Processing

1- JSON Parsing
Parse nested JSON structure

2- Schema Creation

3- Processing
Data Frame Explode and Flatten Technique

4- Storage Integration
Store processed alerts
in ClickHouse gtfs_alerts table

Vehicle Positions Monitoring

Information Technology Institute

1- JSON Parsing
Parse nested JSON structure

2- Schema Creation

3- Reading Scheduled Data From Clickhouse
stops, stop_times.

4- Join vehicle positions with static stops, stop_times.
Joining Streaming data with scheduled data

4- Calculate distance using Haversine.
To know the distance between vehicle and the stop.

5- Determine status (arrived if <100m else on_way), delay_seconds.

Challenges and Optimizations

1- Performance Tuning

Optimized Kafka configurations and Spark processing parameters for enhanced throughput and reduced latency

Due to RAM limitations on Azure VM, Kubernetes is needed to start needed containers only in the needed time.

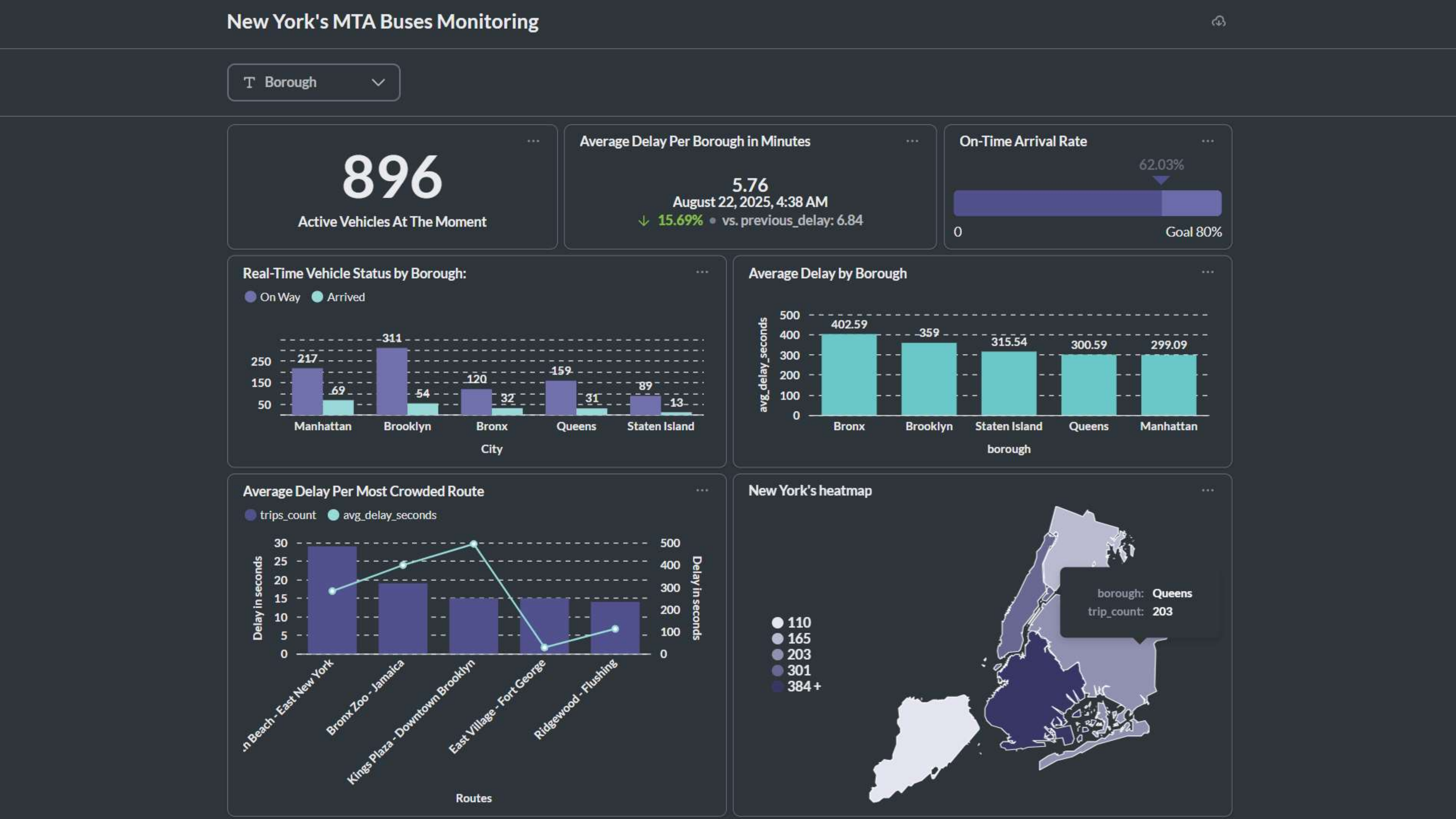
2- Reliability Engineering

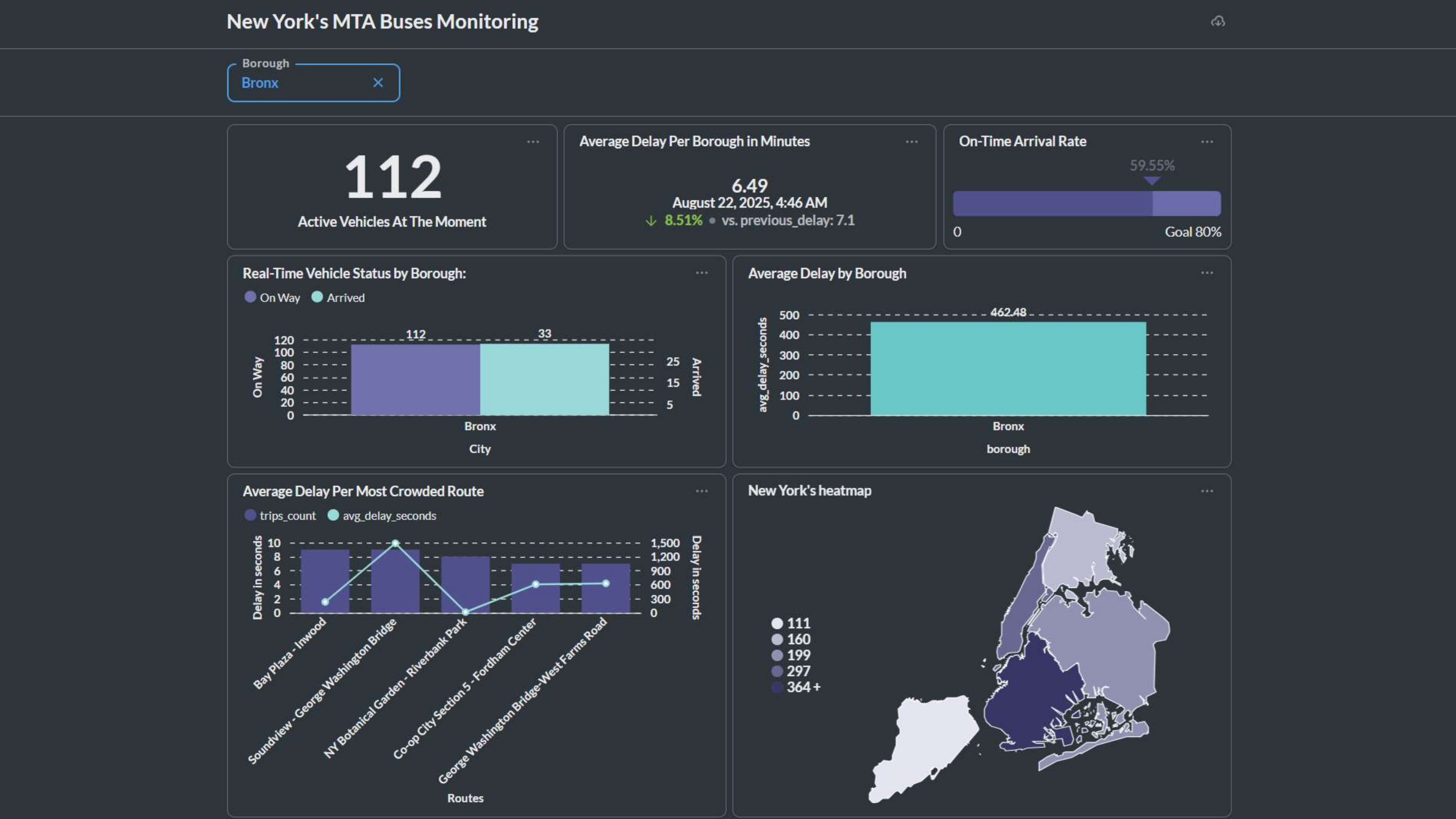
Implemented comprehensive error handling, retry mechanisms, and monitoring for production-grade reliability and fault tolerance

3- System Availability

Azure VM to ease the team work, system workload, and 24/7 monitoring

Real-time Dashboard


New York's MTA Buses Monitoring
Borough
896
Active Vehicles At The Moment
Average Delay Per Borough in Minutes
5.76
August 22, 2025, 4:38 AM
15.69% • vs. previous_delay: 6.84
On-Time Arrival Rate
62.03%
0
Goal 80%
Real-Time Vehicle Status by Borough:
On Way
Arrived
250
150
50
217
69
311
54
120
32
159
31
89
13
Manhattan
Brooklyn
Bronx
Queens
Staten Island
City
Average Delay by Borough
avg_delay_seconds
500
400
300
200
100
0
402.59
359
315.54
300.59
299.09
Bronx
Brooklyn
Staten Island
Queens
Manhattan
borough
Average Delay Per Most Crowded Route
trips_count
avg_delay_seconds
Delay in seconds
30
25
20
15
10
5
0
500
400
300
200
100
0
Delay in seconds
n Beach - East New York
Bronx Zoo - Jamaica
Kings Plaza - Downtown Brooklyn
East Village - Fort George
Ridgewood - Flushing
Routes
New York's heatmap
110
165
203
301
384 +
borough: Queens
trip_count: 203


New York's MTA Buses Monitoring
Borough
Bronx
112
Active Vehicles At The Moment
Average Delay Per Borough in Minutes
6.49
August 22, 2025, 4:46 AM
8.51% • vs. previous_delay: 7.1
On-Time Arrival Rate
59.55%
0
Goal 80%
Real-Time Vehicle Status by Borough:
On Way
Arrived
112
33
On Way
Arrived
Bronx
City
Average Delay by Borough
462.48
avg_delay_seconds
Bronx
borough
Average Delay Per Most Crowded Route
trips_count
avg_delay_seconds
Delay in seconds
Delay in seconds
Bay Plaza - Inwood
Soundview - George Washington Bridge
NY Botanical Garden - Riverbank Park
Co-op City Section 5 - Fordham Center
George Washington Bridge-West Farms Road
Routes
New York's heatmap
111
160
199
297
364 +

References

- Transitland: https://www.transit.land/
- OTP KPI: https://www.mta.info/document/145791
- Average distance between stops in NYC: https://www.mta.info/project/bus-network-redesign/about?utm_source=chatgpt.com

THANK YOU