Real-time Streaming Bus Data with Kafka and Spark

by Andrea Zhang
2022 Aug

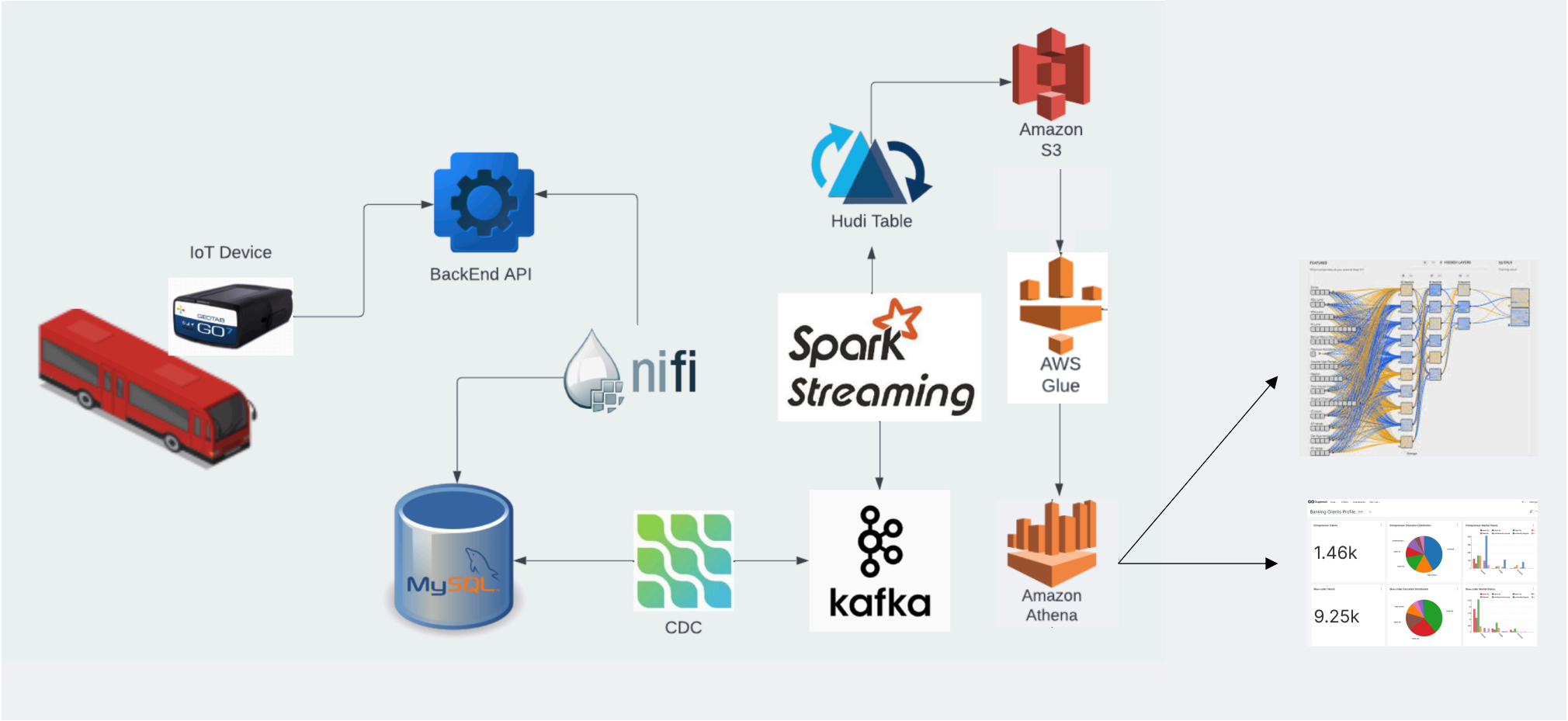
Project Motivation

Objective: The primary focus of this project is to build a real-time application that collects GPS data steamed by IoT devices located on Toronto TTC buses.

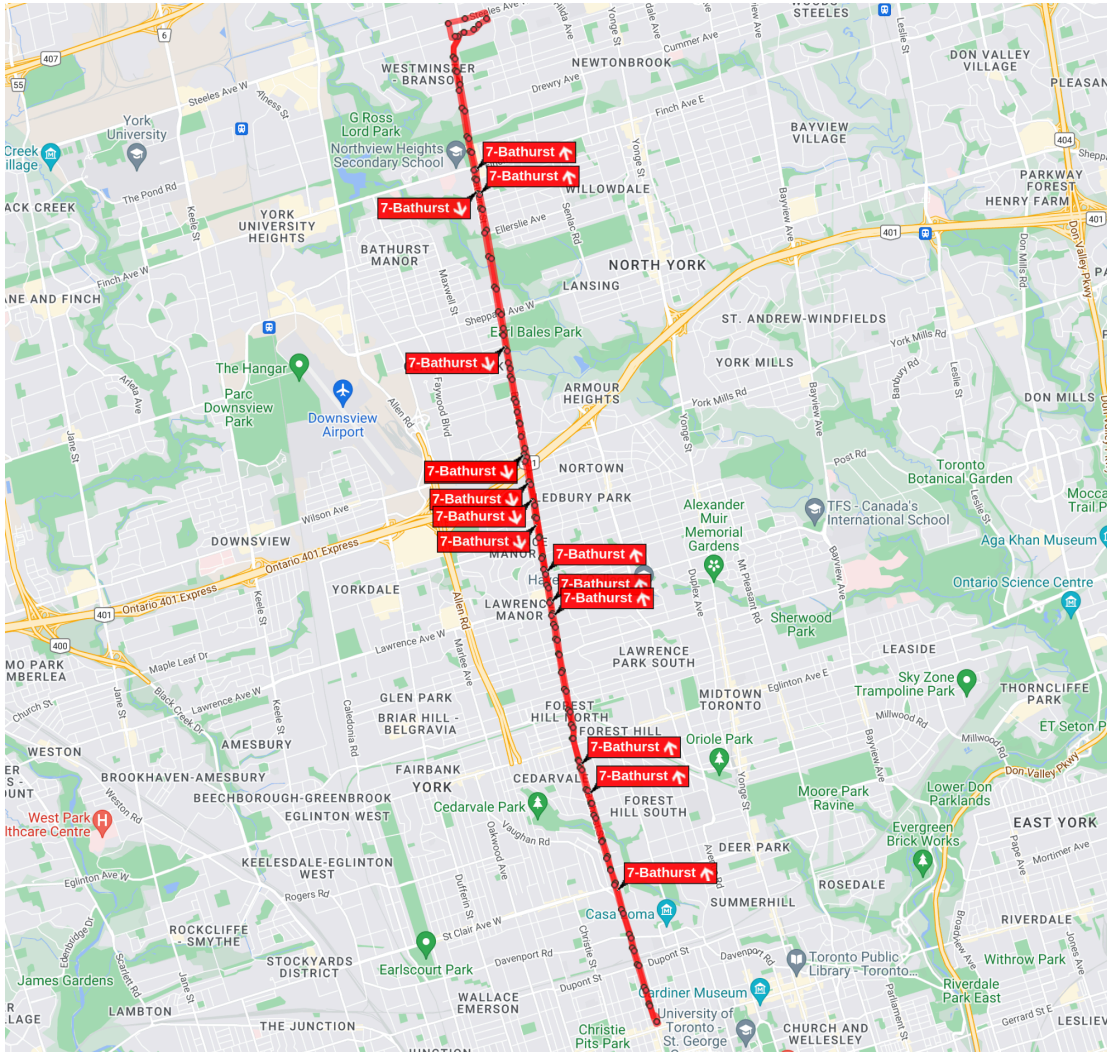
For this project, multiple big data applications were used, including Apache NiFi, CDC(Debezium), Kafka (MSK), Spark Structural Streaming, Docker, MySQL and the analytics layer using Amazon Athena.

Conceptual Data Streaming Architecture

Data Collection | Transformation | Storage | Analytics/ML



Data Collection – RestBus API



The data source used in the project is TTC bus data from an open API called [Rest Bus](#).

RESTBUS

A RESTful JSON API for the NextBus XML feed.

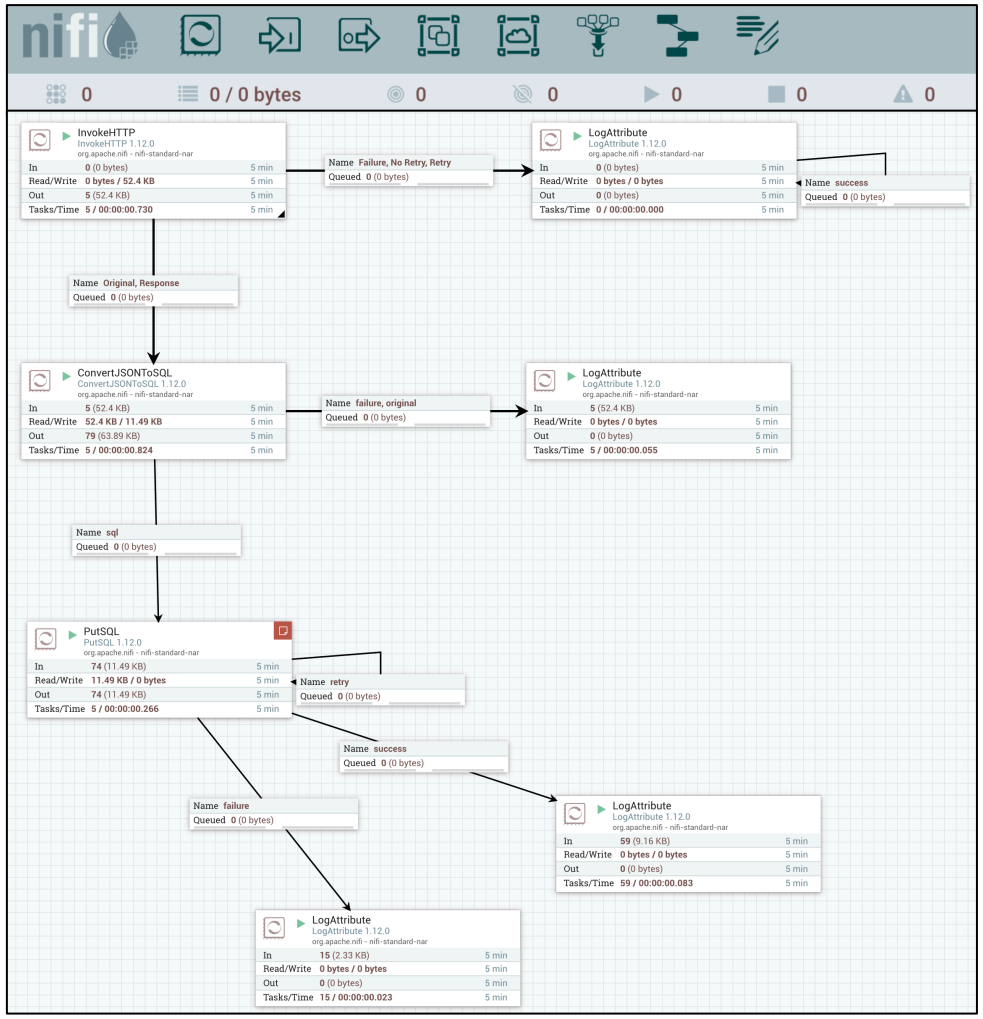
```
npm install restbus
```

DEMO GITHUB

Sample Data


```
[{"id":"7","title":"7-
Bathurst","_links":{"self":{"href":"http://restbus.info/api/agencies/ttc
/routes/7","type":"application/json","rel":"http://restbus.info/_links/r
el/full","rt":"route","title":"Full configuration for ttc route 7-
Bathurst."},"to":[{"href":"http://restbus.info/api/agencies/ttc/routes/7
","type":"application/json","rel":"http://restbus.info/_links/rel/full",
"rt":"route","title":"Full configuration for ttc route 7-
Bathurst."}],"from":[{"href":"http://restbus.info/api/agencies/ttc","typ
e":"application/json","rel":"via","rt":"agency","title":"Transit agency
ttc details."}
```

Data Transformation - Apache NiFi



Apache NiFi extracts data from the Bus API and save it to a MySQL database to make data available for the further consumption.

- ➢ Apache NiFi was set up using a Docker Container on an AWS EC2 instance
- ✓ **InvokeHTTP** processor collects data in JSON blobs. It is scheduled to get information from the Bus API every 30 seconds
- ✓ **ConvertJSONToSQL** processor converts JSON blobs into SQL inserts to update MySQL table
- ✓ **PutSQL** processor executes SQL statements that are generated by the ConvertJSONToSQL processor
- ✓ **LogAttribute** processors handle edge cases

Data Storage – MySQL

```
CREATE TABLE bus_status (
        record_id INT NOT NULL AUTO_INCREMENT,
        id INT NOT NULL,
        routeId INT NOT NULL,
        directionId VARCHAR(40),
        predictable BOOLEAN,
        secsSinceReport INT NOT NULL,
        kph INT NOT NULL,
        heading INT,
        lat REAL NOT NULL,
        lon REAL NOT NULL,
        leadingVehicleId INT,
        event_time DATETIME DEFAULT NOW(),
        PRIMARY KEY (record_id)
);
```

Bus data was constantly loaded into a MySQL table.

- ➢ MySQL was set up using a Docker Container on the AWS EC2 instance
- ➢ A MySQL table was created with a pre-defined schema to receive data
- ✓ The Bus data is imported from NiFi and stored in the MySQL table.
- ✓ The data in SQL database constantly gets updated and stored for further downstream processes.

```
mysql> select * from bus_status;
+-----------+------+---------+-------------+-------------+-----------------+-----+---------+------------+-------------+-----------------+---------------------+
| record_id | id   | routeId | directionId | predictable | secsSinceReport | kph | heading | lat        | lon         | leadingVehicleId | event_time          |
+-----------+------+---------+-------------+-------------+-----------------+-----+---------+------------+-------------+-----------------+---------------------+
|         1 | 8156 |       7 | 7_1_7       |           1 |              28 |  18 |     348 | 43.73719140 | -79.43381820 |            NULL | 2022-07-24 18:13:55 |
|         2 | 8178 |       7 | 7_0_7       |           1 |              28 |   0 |     168 | 43.70559560 | -79.42656040 |            NULL | 2022-07-24 18:13:55 |
|         3 | 8132 |       7 | 7_1_7       |           1 |              28 |  33 |      73 | 43.79323970 | -79.44185970 |            NULL | 2022-07-24 18:13:55 |
|         4 | 8138 |       7 | 7_1_7       |           1 |              28 |  28 |     349 | 43.77753710 | -79.44385090 |            NULL | 2022-07-24 18:13:55 |
|         5 | 8358 |       7 | 7_1_7       |           1 |              28 |  24 |     344 | 43.68229960 | -79.41795270 |            NULL | 2022-07-24 18:13:55 |
|         6 | 8389 |       7 | 7_0_7       |           1 |              28 |   0 |     181 | 43.70171730 | -79.42577360 |            NULL | 2022-07-24 18:13:55 |
|         7 | 8130 |       7 | 7_0_7       |           1 |              28 |   0 |     181 | 43.70195440 | -79.42571090 |            NULL | 2022-07-24 18:13:55 |
|         8 | 8384 |       7 | 7_1_7       |           1 |              28 |   0 |     230 | 43.79187920 | -79.44193110 |            NULL | 2022-07-24 18:13:55 |
|         9 | 8150 |       7 | 7_1_7       |           1 |              28 |   0 |     342 | 43.69939040 | -79.42483870 |            NULL | 2022-07-24 18:13:55 |
|        10 | 8360 |       7 | 7_0_7       |           1 |              28 |   0 |     263 | 43.79083100 | -79.44400010 |            NULL | 2022-07-24 18:13:55 |
|        11 | 8195 |       7 | 7_0_7       |           1 |              28 |   6 |     167 | 43.70161430 | -79.42572780 |            NULL | 2022-07-24 18:13:55 |
|        12 | 8382 |       7 | 7_0_7       |           1 |              28 |   5 |      75 | 43.73743050 | -79.43408200 |            NULL | 2022-07-24 18:13:55 |
|        13 | 8119 |       7 | 7_1_7       |           1 |              29 |  41 |     350 | 43.77991930 | -79.44451140 |            NULL | 2022-07-24 18:13:55 |
|        14 | 8156 |       7 | 7_1_7       |           1 |              28 |   3 |     251 | 43.73764020 | -79.43405470 |            NULL | 2022-07-24 18:14:25 |
```

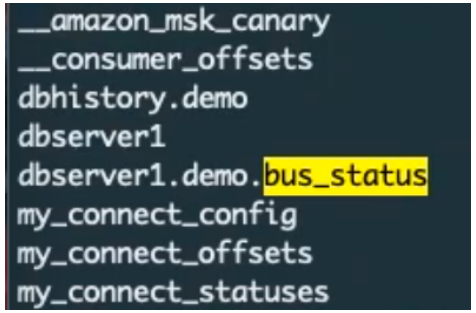
Data Distribution – CDC (Debezium)

Run Debezium using a Kafka tool called Connect to enable MySQL and Kafka Connection. Debezium helps monitoring changes in both MySQL target table and Kafka topics.

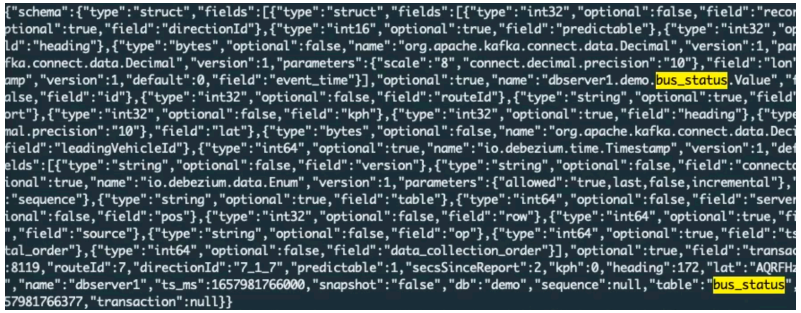
- ✓ Debezium reads data from MySQL databases and turn it into event streams, so applications can quickly react to each row-level change in the databases, such as inserts, updates, and deletes.
- ✓ Debezium pushes the data into a Kafka topic and monitors data changes in Kafka topics.

- ✓ Insert
- ✓ Update
- ✓ Delete

Kafka Topics



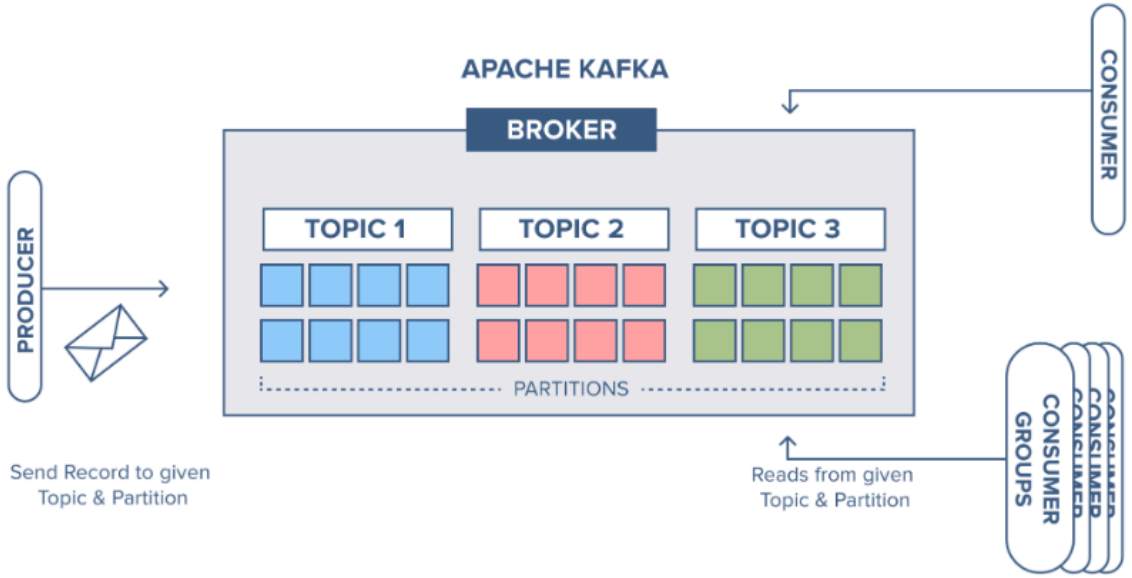
Check Data in Kafka Topic



Data Distributed Streaming – Kafka (MSK)

Kafka was used to transfer the Bus data from MySQL to Spark Streaming in a fast, fault-tolerance and high scalable way.

- ✓ The Kafka topic receives incoming messages in JSON blobs.
- ✓ Built-in partitioning
- ✓ Replication



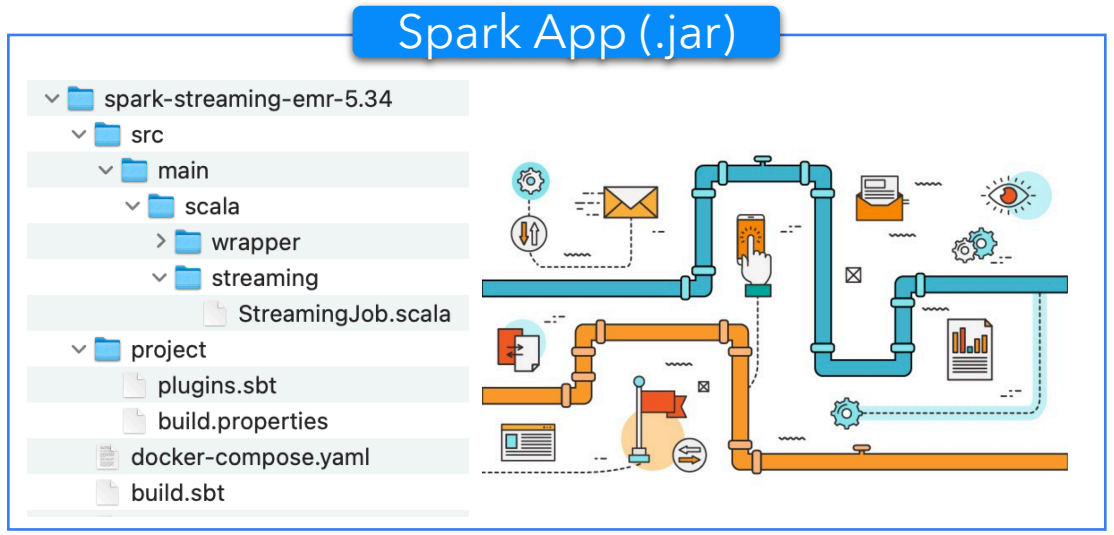
Kafka Topic Description

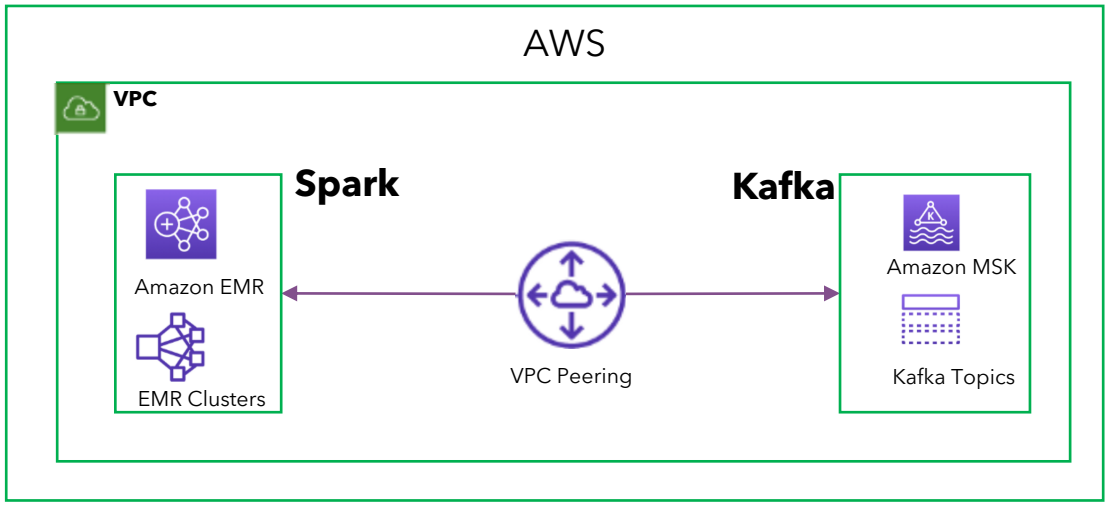
```
]$ bin/kafka-topics.sh --describe my_connect_config --boots
mp3s.1qo03c.c23.kafka.us-east-1.amazonaws.com:9092
fig        PartitionCount: 1       ReplicationFactor: 2
nect_config        Partition: 0    Leader: 2       Replicas
tuses      PartitionCount: 5       ReplicationFactor: 2
nect_statuses      Partition: 0    Leader: 2       Replicas
nect_statuses      Partition: 1    Leader: 1       Replicas
nect_statuses      Partition: 2    Leader: 3       Replicas
nect_statuses      Partition: 3    Leader: 2       Replicas
nect_statuses      Partition: 4    Leader: 1       Replicas
```

Data Streaming – Spark Streaming

Ingesting Data From Kafka with Spark Streaming

- Create an AWS EMR for Spark and enable EMR and MSK communication through CPV Perring.
- Compiling Spark Streaming Jobs Built in Scala. Upload .jar file to an S3 bucket.
- Spark-submit using the .jar file to the Spark (EMR cluster) master node.
- The Spark application ingests data from Kafka, transform the data according to a predefined schema (using an S3 file), and export data to an S3 bucket.





Data Streaming – Spark Streaming

Micro batches of data coming from the Kafka topic.

Micro-batch processing: divides incoming streams into small batches for processing.

```
-------------------------------------------
Batch: 189
-------------------------------------------
+-----------+------------+-------+----+---+---------+---------------+----------+-----------+---------+-------+---------------+
|directionId|event_time  |heading|id  |kph|lat      |leadingVehicleId|lon       |predictable|record_id|routeId|secsSinceReport|
+-----------+------------+-------+----+---+---------+---------------+----------+-----------+---------+-------+---------------+
|7_0_7      |1656546597000|281    |9004|0  |43.790825|null           |-79.444496|true       |4706     |7      |26             |
|7_1_7      |1656546597000|349    |9002|1  |43.76817 |null           |-79.441475|true       |4707     |7      |26             |
|7_1_7      |1656546597000|172    |9006|0  |43.66606 |null           |-79.411064|true       |4708     |7      |27             |
|7_0_7      |1656546597000|169    |8313|0  |43.737576|null           |-79.434235|true       |4709     |7      |26             |
|7_1_7      |1656546597000|349    |8395|31 |43.7115  |null           |-79.42788 |true       |4710     |7      |26             |
|7_0_7      |1656546597000|173    |9000|24 |43.719265|null           |-79.4298  |true       |4711     |7      |26             |
|null       |1656546597000|142    |8309|14 |43.741898|null           |-79.45461 |false      |4712     |7      |27             |
|7_0_7      |1656546597000|172    |8102|0  |43.666378|null           |-79.411125|true       |4713     |7      |26             |
|7_1_7      |1656546597000|349    |9012|0  |43.73826 |null           |-79.43412 |true       |4714     |7      |26             |
|7_0_7      |1656546597000|216    |8165|0  |43.790855|null           |-79.44464 |true       |4715     |7      |26             |
|7_1_7      |1656546597000|337    |9018|16 |43.68691 |null           |-79.419716|true       |4716     |7      |25             |
|7_0_7      |1656546597000|163    |9016|22 |43.768547|null           |-79.44178 |true       |4717     |7      |26             |
|7_0_7      |1656546597000|169    |8302|25 |43.70999 |null           |-79.42762 |true       |4718     |7      |26             |
|7_1_7      |1656546597000|349    |1370|39 |43.77536 |null           |-79.443245|true       |4719     |7      |26             |
|7_0_7      |1656546627000|281    |9004|0  |43.790825|null           |-79.444496|true       |4720     |7      |22             |
|7_1_7      |1656546627000|349    |9002|34 |43.770443|null           |-79.44208 |true       |4721     |7      |22             |
|7_1_7      |1656546627000|172    |9006|0  |43.66606 |null           |-79.411064|true       |4722     |7      |23             |
|7_0_7      |1656546627000|167    |8313|46 |43.736576|null           |-79.43387 |true       |4723     |7      |22             |
|7_1_7      |1656546627000|350    |8395|0  |43.71225 |null           |-79.42798 |true       |4724     |7      |22             |
|7_0_7      |1656546627000|169    |9000|4  |43.71797 |null           |-79.42948 |true       |4725     |7      |23             |
+-----------+------------+-------+----+---+---------+---------------+----------+-----------+---------+-------+---------------+
only showing top 20 rows
```

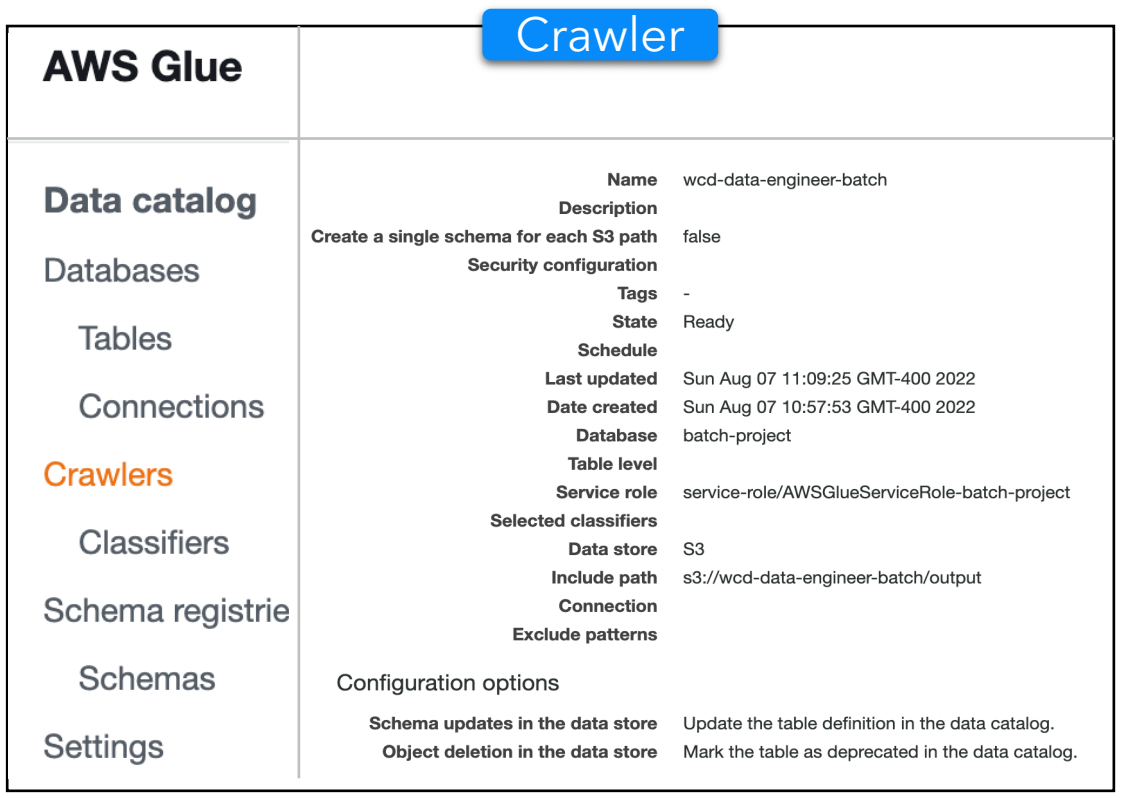
Data Integration - Glue

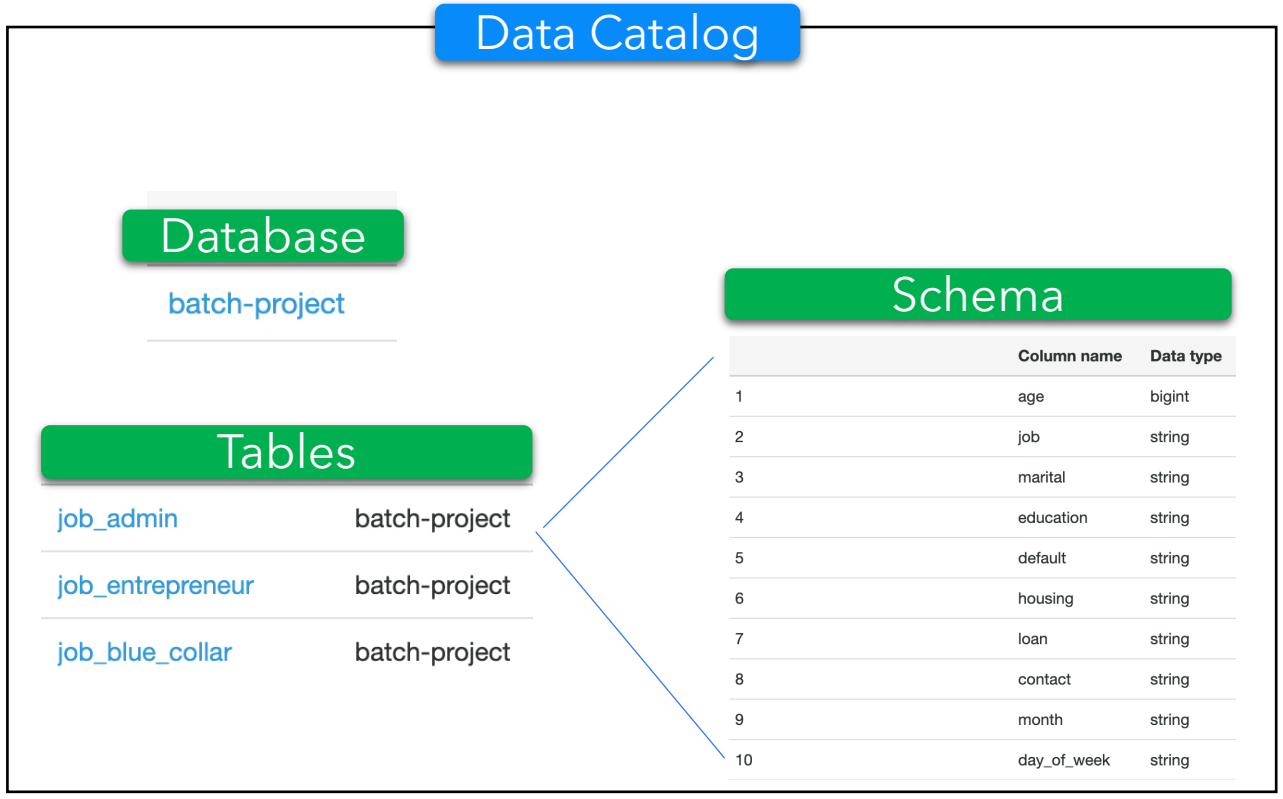
Glue provides serverless data integration that prepare and combine data for analytics.

The Glue data Catalog serves as a central metadata repository.

✓ Discover and search across multiple data sets using Glue data Catalog.

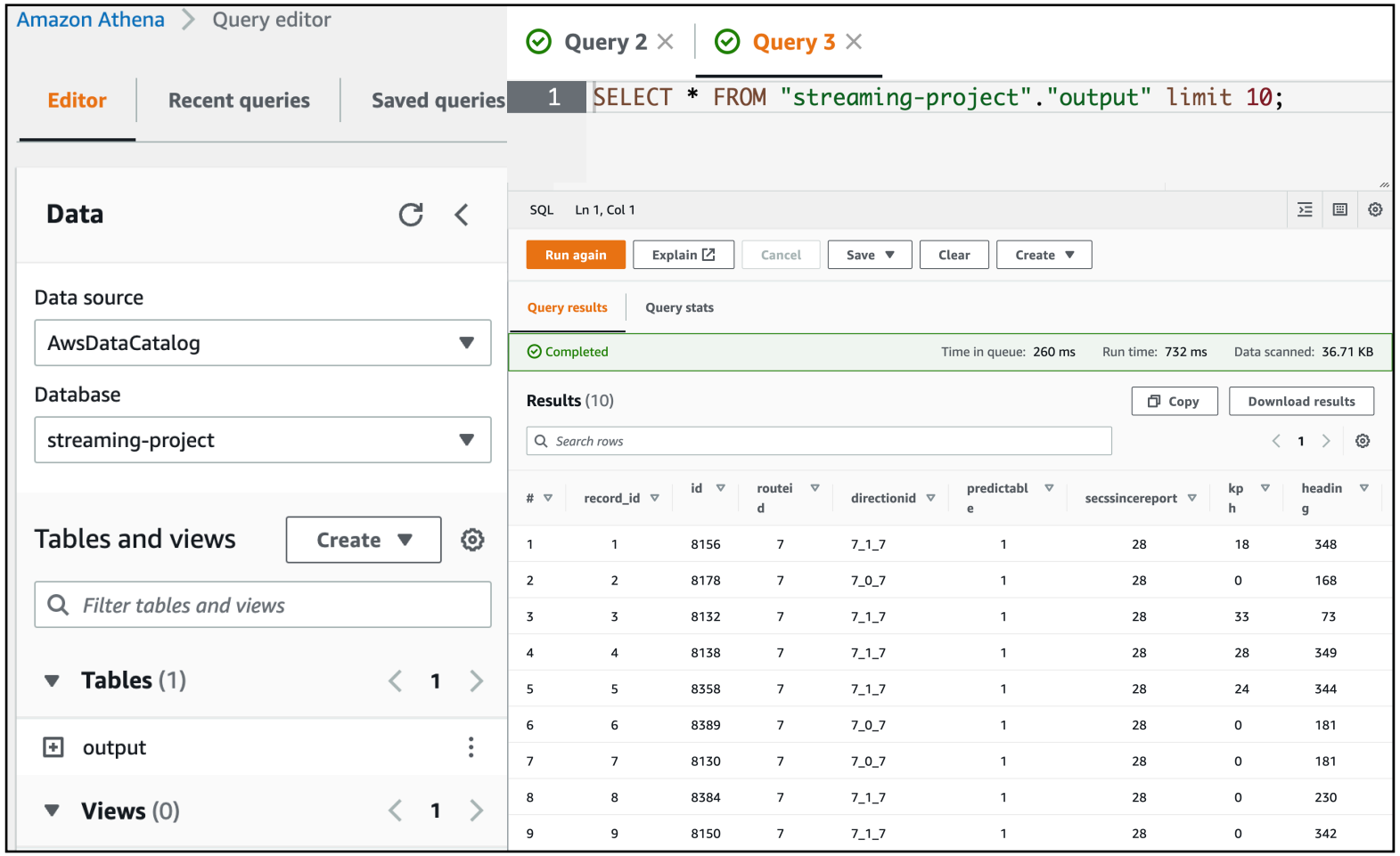
✓ Once the data is cataloged, it is immediately available for search and query using Amazon Athena.





Data Analytics – AWS Athena

Connect Glue Data Catalog and point to the streaming project database and enable querying the bus data instantly.



Use Cases

Monitoring

Optimization

Trip Planner

ML predictions