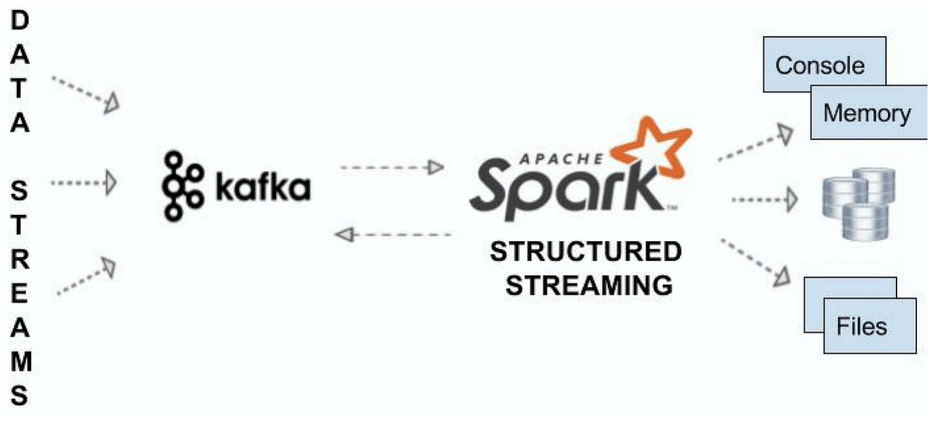


Assignment 3: Build a real-time data ingestion and processing pipeline for traffic events

Due Date: 2025-03-30 11:55 PM

🚨 **Warning** 🚨

Using ChatGPT or any Large Language Model (LLM) for code generation is strictly prohibited. Any such case will be considered plagiarism and will be dealt with accordingly. Ensure that all submitted work is original and adheres to academic integrity guidelines.

1. Problem Statement: Real-Time Traffic Monitoring System

You are tasked with developing a **real-time traffic monitoring system** for a smart city. The system will:

1. Ingest live traffic sensor data from different road sensors using Kafka.
2. Process the data in real-time using PySpark Structured Streaming.
3. Perform data quality validation to ensure that:
   - No missing or corrupted records exist.
   - The sensor data is within valid ranges.
   - Duplicates are handled properly.
4. Aggregate traffic patterns to analyze trends (e.g., sudden speed drops, high congestion)
5. Write the final processed data back to kafka topics for real-time dashboards.

2. Dataset & Input Format

Your pipeline will simulate real-time streaming data from multiple traffic sensors. Each Kafka message represents a traffic event in JSON format, with the following structure:

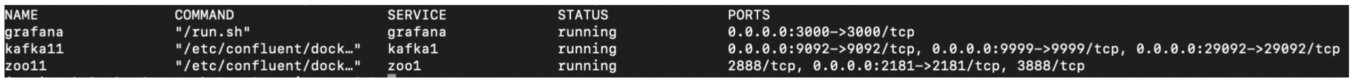
```json
{
    "sensor_id": "S123",
    "timestamp": "2025-03-15T14:25:10",
    "vehicle_count": 15,
    "average_speed": 45.6,
    "congestion_level": "LOW"
}
```

Field Descriptions:

- sensor_id: Unique identifier of the traffic sensor.
- timestamp: Event time of the recorded traffic.
- vehicle_count: Number of vehicles detected in the last second.
- average_speed: Average speed of passing vehicles.
- congestion_level: Categorized as **LOW, MEDIUM, HIGH**.

3. Prerequisites

1. Please use the setup we followed in lab 5, we will be using one additional service (Grafana) for this assignment to visualize our real-time analysis. The updated docker-compose.yml file is attached. After running the docker compose, you should have 3 running containers as shown below. *Your container names could be different.*



```
NAME                COMMAND                  SERVICE             STATUS              PORTS
grafana             "/run.sh"                grafana             running             0.0.0.0:3000->3000/tcp
kafka11             "/etc/confluent/dock…"   kafka1              running             0.0.0.0:9092->9092/tcp, 0.0.0.0:9999->9999/tcp, 0.0.0.0:29092->29092/tcp
zoo11               "/etc/confluent/dock…"   zoo1                running             2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp
```

2. The web UI for grafana is available at localhost:3000. The default credentials are admin/admin. You will need to enable the Kafka plugin from the data sources in the UI before creating the dashboards.
3. For students who did not complete the lab, please follow the steps 2 & 3 to have a smooth kafka cluster and pyspark libraries in place.
4. Please note that stopping or killing your containers would lose your progress so be careful. You can use `docker-compose -f zk-single-kafka.yml down -v` to stop your containers.
5. IMPORTANT: If you have a macbook, please contact the course staff immediately when you face a problem with grafana. We will walk you through an alternate approach.

Part 1: Kafka Producer (3 Marks)

Write a **Kafka Producer** that simulates traffic data:

1. Generate random **traffic events** every second.

2. Ensure at least **5 different sensors** send data to Kafka.
3. Publish messages to a Kafka topic named **"traffic_data"**.
4. Use Python's random library for data simulation.

Part 2: PySpark Structured Streaming (5 Marks)

Consume and process streaming data using **PySpark Structured Streaming**:

1. Read data from Kafka topic "traffic_data".
2. Convert binary Kafka messages into readable JSON format.
3. Perform traffic analysis given below.

Analysis Tasks:

1. Compute Real-Time Traffic Volume per Sensor (1 Mark)
   - Calculate the total number of vehicles detected every 5 minutes per sensor.
2. Detect Congestion Hotspots in Real Time (1 Mark)
   - Identify sensors where `congestion_level == "HIGH"` for 3 consecutive time windows.
3. Calculate the Average Speed per Sensor with Windowing (1 Mark)
   - Compute average vehicle speed per sensor over a 10-minute rolling window.
4. Identify Sudden Speed Drops (Anomaly Detection) (1 Mark)
   - Detect if speed drops by more than 50% within 2 minutes for any sensor.
5. Find the Busiest Sensors in the Last 30 Minutes (1 Mark)
   - Identify top 3 sensors with the highest total vehicle count in the last 30 minutes.

Part 3: Data Quality Checks (4 Marks)

Implement the following data quality validations:

1. Missing Values: Remove or flag records with null sensor_id or timestamp.
2. Range Validation: Ensure vehicle_count >= 0 and average_speed > 0.
3. Duplicate Handling: Remove duplicate events by using deduplication logic.
4. Schema Validation: Enforce correct data types using PySpark schema.

Hint: Use .dropDuplicates(["sensor_id", "timestamp"]) and .filter(col("vehicle_count") >= 0)

Part 4: Output Storage & Visualization (3 Marks)

Now we have to send analysis results to Kafka and visualize them in Grafana.

Task 1: Write Streaming Analysis Results to Kafka (1 Mark)

- Write aggregated streaming data (traffic volume, avg speed) to Kafka topic `traffic_analysis`.
- Use structured JSON format for easy parsing.
- Ensure correct partitioning so messages are evenly distributed.

Task 2: Configure Grafana for Real-Time Kafka Visualization (1 Mark)

- Add Kafka (`traffic_analysis` topic) as a Grafana data source.
- Create real-time visualizations given below to monitor streaming results.

Task 3: Create the Following Dashboards in Grafana (Submit 2/4 dashboards)

1. **Traffic Volume Over Time** (Bar Chart)
   - X-Axis: Time Window (5 min)
   - Y-Axis: Total Vehicle Count
   - Data: `$.total_count`
   - Goal: Monitor traffic trends per sensor.
2. **Average Speed per Sensor** (Time-Series Line Chart)
   - X-Axis: Time Window (10 min)
   - Y-Axis: Average Speed (km/h)
   - Data: `$.avg_speed`
   - Goal: Identify speed trends across different roads.
3. **Congestion Hotspots** (Heatmap)
   - X-Axis: Sensor ID
   - Y-Axis: Congestion Level (LOW, MEDIUM, HIGH)
   - Data: `$.congestion_level`
   - Goal: Identify busiest roads in real-time.
4. **Top 3 Busiest Sensors (Leaderboard)**
   - Metric: Total vehicle count (last 30 min)
   - Sorted in descending order.
   - Goal: Identify most congested areas.

5. Submission Guidelines

- Submit via **ZIP file**.

Required Files

- **`producer.py`** – Kafka Producer (simulated sensor data).
- **`streaming.py`** – PySpark Streaming (data processing & Kafka output).
- **`docker-compose.yml`** – IF ONLY you edited the provided compose file
- **`grafana_dashboard.json`** – (Optional) Exported Grafana dashboard.

- **Screenshots** – Grafana dashboard showing real-time data.
- `architecture.png` – Architecture diagram showing the end to end pipeline (reference slide #68 in Module 4 slides of class lectures)

6. Helpful Resources

- https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/index.html
- https://grafana.com/grafana/plugins/hamedkarbasi93-kafka-datasource/
- https://blog.devgenius.io/pyspark-and-kafka-streaming-reading-and-writing-data-in-different-formats-149c78b54f9
- Understanding Dashboards in Grafana | Panels, Visualizations, Queries, and Transformations