

Project Name: Real-Time Vehicle GPS Tracker

Owner: Sri Sivakumar Ramar

Platform: Databricks + Delta Lake + Power BI

Type: Real-time analytics with geospatial visualization

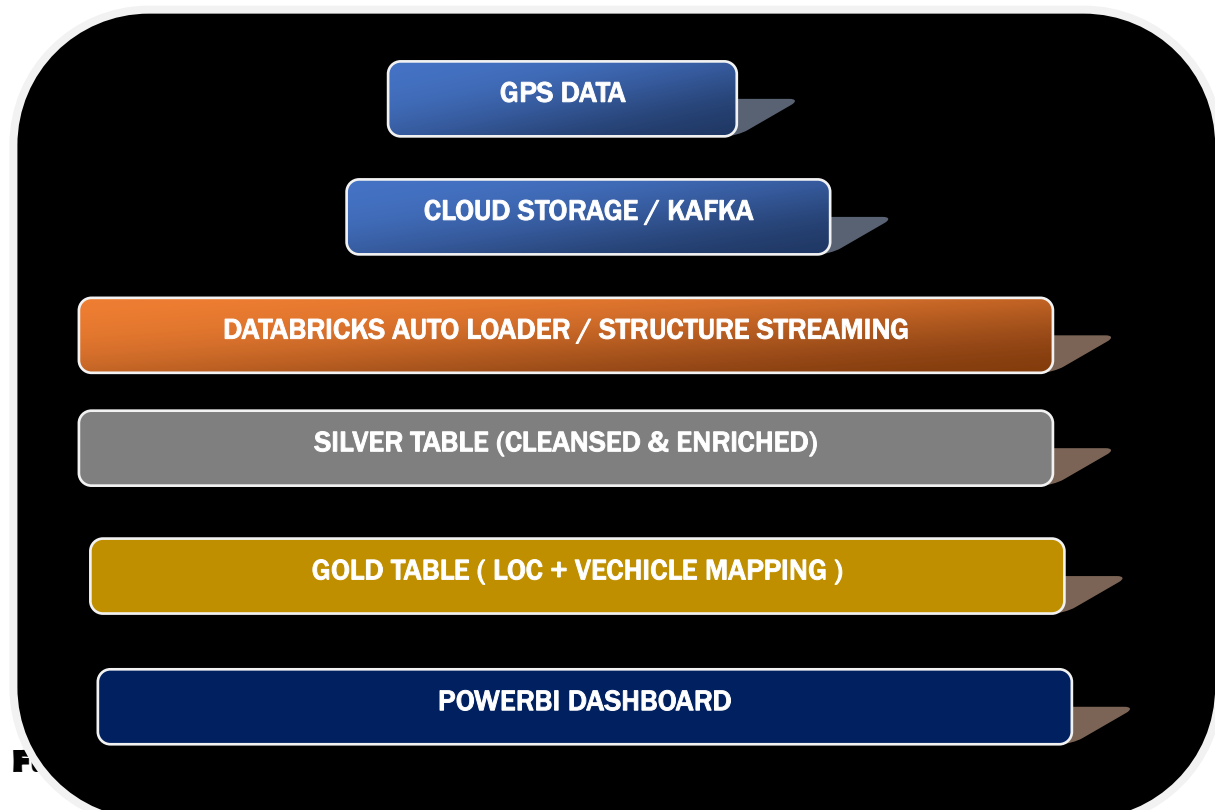
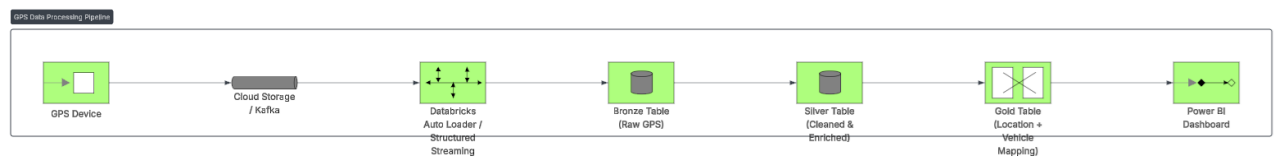
Use Case: Track a specific vehicle's GPS coordinates over time

Objective: This project implements a **real-time vehicle tracking system** using the Databricks ecosystem to process and analyze live GPS data. GPS signals from vehicles are ingested through streaming sources into cloud storage or Kafka, and then processed in Databricks using Auto Loader and Structured Streaming.

The data flows through Bronze, Silver, and Gold Delta Lake layers for raw capture, cleansing, enrichment, and aggregation. The final curated data is visualized in Power BI through interactive dashboards, enabling users to monitor vehicle locations, speed, and movement history in real time.

The solution is scalable, secure, and supports future enhancements like geofencing and alerting, making it suitable for fleet management, logistics, and transportation analytics.

1. Architecture Overview



Databricks: GPS Logger

- Ingest GPS data in real-time
- Schema should include: vehicle_id, latitude, longitude, timestamp, speed, direction

Ingestion Options

- **Option 1:** Azure Event Hub / Kafka
- **Option 2:** REST API posting to cloud storage (e.g., Azure Blob, S3)

Here's code to simulate how vehicle GPS data **is collected by an** HTTP API, IoT Hub, or Kafka, **and then written as** files into cloud storage (e.g., Azure Blob Storage or AWS S3). You can adapt this depending on your source. Below are examples for each source:

1. HTTP API to Cloud Storage (Python Flask + Azure Blob)

Simulates GPS data pushed from a device to your API, which writes to cloud storage.

Python Code (Flask API → Azure Blob)

```
python
CopyEdit

from flask import Flask, request, jsonify
from datetime import datetime
import json
from azure.storage.blob import BlobServiceClient
import uuid

app = Flask(__name__)

# Azure Blob Storage credentials
connect_str =
"DefaultEndpointsProtocol=https;AccountName=SAK;AccountKey=YOUR_KEY;EndpointSuffix=core.windows.net"
container_name = "gps-landing"
blob_service_client = BlobServiceClient.from_connection_string(connect_str)

@app.route('/upload_gps', methods=['POST'])
def upload_gps():
    data = request.json
    timestamp = datetime.utcnow().strftime("%Y%m%d_%H%M%S")
    file_name = f"gps_{data['vehicle_id']}_{timestamp}_{uuid.uuid4().hex[:6]}.json"

    # Convert to JSON string
    blob_data = json.dumps(data)

    # Upload to Azure Blob
    blob_client = blob_service_client.get_blob_client(container=container_name,
blob=file_name)
    blob_client.upload_blob(blob_data, overwrite=True)

    return jsonify({"status": "uploaded", "file": file_name}), 200
```

Databricks: GPS Logger

```
if __name__ == '__main__':  
    app.run(debug=True, port=5000)
```

Example Input

```
bash  
CopyEdit  
curl -X POST http://localhost:5000/upload_gps -H "Content-Type: application/json" -d '{  
    "vehicle_id": "VH001",  
    "latitude": 12.9716,  
    "longitude": 77.5946,  
    "timestamp": "2025-07-10T10:00:00Z",  
    "speed": 45.5,  
    "direction": "NE"  
}'
```

2. Kafka Consumer → Cloud Storage (Spark Streaming)

Use this if GPS data is pushed to a Kafka topic and needs to be saved as files.

Python Kafka Producer: Push GPS Data

```
from kafka import KafkaProducer  
import json  
import time  
from datetime import datetime  
import random
```

```
from kafka import KafkaProducer  
import json  
import time  
from datetime import datetime  
import random
```

Set up Kafka producer

```
producer = KafkaProducer(  
    bootstrap_servers='localhost:9092',  
    value_serializer=lambda v: json.dumps(v).encode('utf-8')  
)
```

Simulate GPS data

```
def generate_gps_data(vehicle_id):  
    return {  
        "vehicle_id": vehicle_id,  
        "latitude": round(random.uniform(12.9, 13.0), 6),  
        "longitude": round(random.uniform(77.5, 77.7), 6),  
        "timestamp": datetime.utcnow().isoformat(),  
        "speed": round(random.uniform(20, 60), 2),  
        "direction": random.choice(["N", "S", "E", "W", "NE", "NW", "SE", "SW"])  
    }
```

Databricks: GPS Logger

Continuously push data to Kafka topic

```
while True:
    gps_data = generate_gps_data("VH001")
    producer.send("gps_topic", gps_data)
    print("Sent:", gps_data)
    time.sleep(5) # simulate GPS update every 5 seconds
```

Kafka Topic Setup

Create Kafka topic (optional if auto.create.topics.enable=true)

```
kafka-topics.sh --create --topic gps_topic --bootstrap-server localhost:9092 --partitions 1 --
replication-factor 1
```

Output (Kafka messages)

Kafka will receive JSON messages like:

```
json
CopyEdit
{
  "vehicle_id": "VH001",
  "latitude": 12.973456,
  "longitude": 77.594321,
  "timestamp": "2025-07-10T10:30:00Z",
  "speed": 48.6,
  "direction": "NE"
}
```

PySpark Code (Databricks)

```
python
CopyEdit
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StringType, DoubleType, TimestampType
```

```
schema = StructType() \
    .add("vehicle_id", StringType()) \
    .add("latitude", DoubleType()) \
    .add("longitude", DoubleType()) \
    .add("timestamp", StringType()) \
    .add("speed", DoubleType()) \
    .add("direction", StringType())
```

```
kafka_df = (
```

Databricks: GPS Logger

```
spark.readStream
  .format("kafka")      # Listens to the Kafka topic gps_topic
  .option("kafka.bootstrap.servers", "kafka-broker:9092")
  .option("subscribe", "gps_topic")
  .load()
)

parsed_df = kafka_df.select(from_json(col("value").cast("string"),
schema).alias("data")).select("data.*")

parsed_df.writeStream \
  .format("json") \
  .option("path", "/mnt/gps/landing/") \
  .option("checkpointLocation", "/mnt/gps/checkpoints/kafka") \
  .outputMode("append") \
  .start()
```

This writes GPS data from Kafka to /mnt/gps/landing/ where cloudFiles can pick it up.

3. IoT Hub (Azure) → Blob Storage (Auto with Event Grid)

No manual code needed — use **Azure Event Hub or IoT Hub routing to Blob Storage**.

Steps:

1. Send GPS data from the IoT device to Azure IoT Hub
2. Use **IoT Hub routing** to send messages to Azure Blob Storage
3. Configure **storage container path** as /gps/landing/
4. Auto Loader in Databricks will consume new files

If you need, I can give an ARM template or device simulation code for Azure IoT as well.

Summary: Which to Use?

Source	Code You Need	Use Case
HTTP POST API	Flask + Azure Blob SDK (above)	Direct device → server ingest
Kafka	Spark Streaming consumer	Event bus or real-time platform
IoT Hub	Azure routing + no code	Azure-based IoT ecosystem

3. Databricks Pipeline Design

3.1 Auto Loader for Real-Time Ingestion (Bronze Layer)

```
python
CopyEdit
from pyspark.sql.functions import input_file_name

raw_df = (
```

Databricks: GPS Logger

```
spark.readStream
  .format("cloudFiles")
  .option("cloudFiles.format", "json")
  .load("/mnt/gps/landing/")
)

raw_df.writeStream \
  .format("delta") \
  .option("checkpointLocation", "/mnt/gps/checkpoints/bronze") \
  .outputMode("append") \
  .table("bronze.gps_raw")
```

3.2 Data Cleansing & Enrichment (Silver Layer)

```
python
CopyEdit
from pyspark.sql.functions import col, to_timestamp

bronze_df = spark.readStream.table("bronze.gps_raw")

silver_df = bronze_df.select(
    col("vehicle_id"),
    col("latitude").cast("double"),
    col("longitude").cast("double"),
    to_timestamp("timestamp").alias("event_time"),
    col("speed").cast("double"),
    col("direction")
).filter("latitude IS NOT NULL AND longitude IS NOT NULL")

silver_df.writeStream \
  .format("delta") \
  .option("checkpointLocation", "/mnt/gps/checkpoints/silver") \
  .outputMode("append") \
  .table("silver.gps_cleaned")
```

3.3 Aggregation & Geohash (Gold Layer)

```
python
CopyEdit
from pyspark.sql.functions import hour, date_format

silver_df = spark.readStream.table("silver.gps_cleaned")

gold_df = silver_df.select(
    "vehicle_id", "latitude", "longitude", "event_time", "speed", "direction"
).withColumn("date", date_format("event_time", "yyyy-MM-dd")) \
  .withColumn("hour", hour("event_time"))

gold_df.writeStream \
  .format("delta") \
```

Databricks: GPS Logger

```
.option("checkpointLocation", "/mnt/gps/checkpoints/gold") \  
.outputMode("append") \  
.table("gold.gps_aggregated")
```

4. Power BI Integration

Steps:

1. Install **Power BI Desktop**
 2. Connect to Databricks using:
 - **ODBC connector** or
 - **Databricks SQL endpoint**
 3. Load data from gold.gps_aggregated
 4. Use **Map visualizations** (Azure Map / ArcGIS Map):
 - Latitude, Longitude, Vehicle ID, Timestamp, Speed
 5. Add filters:
 - Vehicle selector
 - Time range
 - Speed filter
-

5. Functional Requirements

Requirement	Description
GPS Feed	Real-time GPS feed from vehicle(s)
Tracking	Track vehicle path over time
Filtering	Filter data by time, vehicle ID
Alerting	Detect high speed or geofence breaches (future enhancement)
Reporting	Interactive Power BI dashboard with maps

6. Non-Functional Requirements

Category	Description
Scalability	Stream multiple vehicle data concurrently
Latency	Ingestion + transformation < 1 min
Availability	> 99% via Databricks jobs & Auto Loader
Security	Secrets stored in Databricks Secret Scope; RBAC for tables
Observability	Job monitoring using job_runtime_log Delta Table
Data Quality	Null checks, coordinate validation, schema enforcement
Audit	Audit logs on write streams and access logs
Cost Optimization	Use Photon/Delta caching for performance tuning

7. Validation & Testing

Test Area	Strategy
Schema Validation	expectations in DLT (optional)
Real-time latency	Time diff between source and Power BI
Null/malformed values	Assert non-null, lat/lon range checks
Power BI visuals	Check if location plots correctly update by filters
Job failure	Retry + alert via email/webhook

8. Monitoring Jobs (Optional)

```
python
CopyEdit
from datetime import datetime, timezone






log_df = spark.createDataFrame([
    {"job_id": "gps_tracking",
     "run_id": "some_id",
     "start_time": datetime.now(timezone.utc).isoformat(),
     "duration_minutes": 2.5,
     "detected_at": datetime.now(timezone.utc).isoformat()}
])

log_df.write.mode("append").saveAsTable("monitoring.job_runtime_log")
```

9. Folder Structure

```
bash
CopyEdit
/mnt/gps/
├── landing/           # Raw GPS JSON from device
├── checkpoints/
│   ├── bronze/
│   ├── silver/
│   └── gold/
├── notebooks/
│   ├── 01_bronze_ingest.py
│   ├── 02_silver_transform.py
│   └── 03_gold_aggregate.py
├── dashboards/
└── gps_tracker.pbit
```

10. Next Steps

-  Define Databricks jobs & clusters
 -  Secure secrets for Power BI + storage
 -  Schedule the pipeline using Databricks Jobs
 -  Publish Power BI dashboard to workspace
 -  Optional: Add geofencing and alerts
-

FINAL NOTE:

This solution enables real-time GPS vehicle tracking by streaming location data from Kafka into Databricks using Structured Streaming.

Databricks: GPS Logger

The data is parsed, stored in Delta tables, and visualized in Power BI for monitoring vehicle movement, speed, and direction. It supports scalable ingestion, geospatial analytics, and future enhancements like geofencing and alerts.