I want to build real time public transit tracking system that solves the problem of getting real time updates about delays and arrival of public transit. I want to use databricks, spark, kafka for this. tell me how can this be used in free version to complete the project

**Gandhar said:**

You **can absolutely build a real-time public transit tracking system** using **Databricks (Community Edition), Apache Spark, and Kafka**, even on free or local setups. Here’s a complete breakdown of **how to implement this project using free-tier tools** or local alternatives:

**✅ Project Overview**

**Goal:** Provide real-time updates on delays and estimated arrival times for public transit (buses, trains, etc.).

**🔧 Core Components & Free Tools**

| **Component** | **Purpose** | **Free Option** |
| --- | --- | --- |
| **Kafka** | Real-time data ingestion (e.g., GPS feeds) | Run Kafka locally via Docker or Confluent Platform (free) |
| **Spark** | Real-time stream processing | Run locally or on Databricks Community Edition |
| **Databricks (CE)** | Data pipeline orchestration + Notebooks | Free Community Edition (no real-time jobs, but used for logic dev, batch sim, dashboards) |
| **PostgreSQL / Delta Lake** | Store processed data for querying | PostgreSQL (local/free cloud tier) or Databricks Delta tables |
| **Dashboard (Optional)** | Display real-time status updates | Streamlit, Power BI, Grafana (local setup) |

**⚙️ System Architecture**

plaintext

CopyEdit

[GPS/Transit Feed (Simulated or Public API)]

|

[Producer]

(Python + Kafka)

|

[Kafka Broker] (local Docker)

|

[Spark Structured Streaming]

|

[Transform & ETA logic]

|

[Storage Layer]

(PostgreSQL / Delta Lake / Parquet)

|

[Dashboard/API]

(Streamlit / Power BI / REST API)

**🧪 Step-by-Step Implementation (Using Free Tools)**

**1. Kafka Setup (Free, Local)**

* Use Docker or Confluent CLI to spin up a **local multi-broker Kafka setup**
* Simulate real-time GPS data (e.g., CSV replay, public transit API like GTFS-RT)
* Kafka topics: bus\_position, train\_delay, route\_update

**2. Spark Streaming Job**

* Run a **Spark Structured Streaming** job (locally or on Databricks CE for simulation) to consume Kafka topics:

python

CopyEdit

spark.readStream.format("kafka").option("subscribe", "bus\_position")...

* Apply transformations:
  + Parse coordinates & timestamp
  + Calculate delays/ETA using current vs scheduled time
  + Add geospatial mapping (optional)

**3. Databricks (Community Edition)**

* Simulate batch ingestion (not real-time) for development/testing
* Use Databricks Notebooks to:
  + Develop transformation logic
  + Visualize sample route and delay data
  + Store as Delta tables

Note: Databricks CE **does not support real-time streaming**. Use it to develop batch/stream logic, test pipelines, visualize results.

**4. Storage**

* Option 1: PostgreSQL (free, local)
* Option 2: Delta Lake (if using Databricks)
* Write structured, enriched data for:
  + Delay reports
  + Historical performance
  + Real-time lookups (via API/UI)

**5. Visualization (Free)**

* Use **Streamlit** to build a lightweight real-time dashboard
* Or use **Grafana** with PostgreSQL plugin
* Or build REST API with **Flask/FastAPI** for integration into apps

**📚 Sample Public Transit Data Sources (Free)**

* **GTFS Realtime**: https://gtfs.org/reference/realtime/v2/
* City APIs:
  + New York MTA
  + LA Metro
  + Chicago CTA
  + Toronto TTC

**🚀 Project Tips**

* Use **faker + scheduler** to simulate GPS pings if no real feed
* Enable **checkpointing** and **watermarking** in Spark to handle late events
* Build **geo-fencing** logic to trigger alerts near stops
* Use **Kafka consumer lag** and **Spark UI** to monitor performance

**🔚 Summary: Free Stack Recommendation**

| **Component** | **Tool (Free Version)** |
| --- | --- |
| Kafka | Local Docker / Confluent CLI |
| Stream Processor | Apache Spark (local) |
| Dev Platform | Databricks CE for testing logic |
| Storage | PostgreSQL / Delta / Parquet |
| Visualization | Streamlit / Grafana / Power BI |