**Build a Transit Data Application**

In an age where transportation data is integral to smart city planning, understanding the intricacies of a public transit system becomes vital. My project focuses on the Massachusetts Bay Transportation Authority's (MBTA) Route 1, aiming to capture, analyze, and visualize real-time data related to this route. Utilizing the MBTA API, I'll store and monitor this data in a MySQL database within a Docker container. This will further enable change data capture (CDC) techniques, allowing our application to track database modifications and replicate those changes to a MongoDB database. By analyzing the collected data over 1 hour (due to RAM limitations), I seek to discern patterns, primarily centered on average transit times and bus speeds. The process is comprehensive, demanding meticulous screenshot submissions and data analysis, conducted in a Jupyter Notebook format.

A Docker network called MBTANetwork was created. All Docker containers were associated with this network.

A screenshot of a computer screen

Description automatically generated

The mysqlDocker folder was unzipped and opened. A screenshot was provided to confirm that the mysqlDocker folder was successfully opened.

A screenshot of a computer

Description automatically generated

Using VS Code, the MBTA.sql file was opened. A screenshot was presented to verify that the MBTA.sql file was successfully opened.

A screenshot of a computer

Description automatically generated

The following code was run in a Jupyter Notebook:

A screenshot of a computer

Description automatically generated

The MBTA API was analyzed using this code. A decision was made on which additional fields returned by the service should be included in the table. Additional fields were added to the mbta\_buses table in the MBTA.sql file. A screenshot was presented showing the successful addition of at least five more fields to the mbta\_buses table.

A screenshot of a computer

Description automatically generated

The Dockerfile location was navigated to from a shell prompt, and a Docker command was run to produce a Docker image named mysqlmbtamasterimg. A Docker container named mysqlserver was created and linked to the previously established MBTANetwork network.

A screenshot of a computer screen

Description automatically generated

A screenshot was offered to verify the successful creation of the mysqlserver Docker container.

A screenshot of a computer

Description automatically generated

A MongoDB Docker container named some-mongo was set up for CDC. It was ensured that this container was part of the MBTANetwork.

A screen shot of a computer

Description automatically generated

A screenshot was provided to confirm the successful creation of the some-mongo Docker container.

A screenshot of a computer

Description automatically generated

The Module16ProjectFlask.zip folder was unzipped on the local machine and then opened using VS Code.

A screenshot of a computer program

Description automatically generated

The code in the mysqldb.py file was adjusted to incorporate all the columns defined in the mbta\_buses SQL table.

A screen shot of a computer

Description automatically generated

The MBTAApiClient.py file was altered to parse all the columns defined in the mbta\_buses SQL table.

A screen shot of a computer program

Description automatically generated

The Mapbox access token was inserted into the index.html file from the Module16ProjectFlask.zip folder.

A screenshot of a computer program

Description automatically generated

The server.py file was modified to initialize the buses list by making an API call to the MBTA database.

A screenshot of a computer program

Description automatically generated

The server.py file was run from VS Code.

A screen shot of a computer

Description automatically generated

A browser was directed to localhost:3000.

A map of a city

Description automatically generated

The DebeziumCDC.zip folder was unzipped on the local machine and opened using VS Code.

A screenshot of a computer program

Description automatically generated

A Docker image named debeziummodule16 was crafted from the DebeziumCDC.zip folder.

A screenshot of a computer program

Description automatically generated

The Debezium Docker container was established and associated with the MBTANetwork network.

A screenshot of a computer screen

Description automatically generated

The nano text editor was installed within the shell of the Debezium Docker container.

A screenshot of a computer program

Description automatically generated

The MongoDB.java file was navigated to and edited to modify the insertRecord method.

A computer screen shot of a blue screen

Description automatically generated

The handleChangeEvent method in the DebeziumListener.java file was adjusted to insert a record into the MongoDB database.

A screenshot of a computer screen

Description automatically generated

The Maven SpringBoot application was run from the Debezium shell prompt.

A screenshot of a computer program

Description automatically generated

The MongoDB database's population was verified, and the javamaven container was created and shown running in a screenshot.

A screenshot of a computer

Description automatically generated

The /java-quick-start/src/main/java/com/mongodb/quickstart directory was navigated to, and its files were listed.

A screenshot of a computer

Description automatically generated

Using the nano text editor, the ReadCDC.java file was created in the specified directory, and specific code was copied into this file.

A computer screen shot of a blue screen

Description automatically generated

From the bash command prompt, while in the /java-quick-start directory, the ReadCDC.java class was executed.

A screenshot of a computer program

Description automatically generated

After an exhaustive process of setting up Docker networks, containers, accessing and modifying data from the MBTA API, and ensuring a robust data pipeline from MySQL to MongoDB through CDC, I have successfully managed to capture a snapshot of Route 1's dynamics over a 1-hour window. By integrating various technologies and platforms, I've not only achieved our data capture and storage objectives but have also paved the way for valuable insights into the route's performance. My subsequent analysis using the Jupyter Notebook will further solidify my understanding, answering essential questions about transit times, bus speeds, and more. This project underscores the power of combined tools in generating insights from real-time data, and it sets a benchmark for similar analyses in the future.