UDACITY

‹ Return to "Data Streaming Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Optimizing Public Transportation

REVIEW

CODE REVIEW

HISTORY

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

This was a really challenging project and you have shown lots of expertise. Sure you have learned a lot and we encourage you to keep up with this hard work. Have a nice day and good luck forward.

Regarding your question, it might be css issue, but to check you can add debug logs in your server.py to check whether you are able to listen to all the topic properly.

Kafka Producer

Using the Kafka Topics CLI, topics appear for arrivals on each train line in addition to the turnstiles for each of those stations.

Good work, topic names are meaningful. Just FYI, you should use the dot (.) and underscore both at once in topic name as it may collide internally.
Resource link:
https://stackoverflow.com/questions/37062904/what-are-apache-kafka-topic-name-limitations
https://github.com/apache/kafka/blob/2.3.0/core/src/main/scala/kafka/admin/TopicCommand.scala#L147-L148 -- line:148

Using the Kafka Topics CLI, messages continuously appear for each station on the train line, for both arrivals and turnstile actions.

Good, Messages are being auto routed and all expected fields are present.

Using the Schema Registry API, a schema is visible for arrivals and turnstile events.

Ans; Required fields are produced and have the appropriate types as defined in the solution schema.

Kafka Consumer

Stations, status, and weather data appear and update in the Transit Status UI.

on_assign callback is used for topic subscription.
All direct Kafka interaction should be contained in just the consumer.py code, and not leaked into the model files.

All Blue, Green, and Red Line stations appear in the Transit Status UI.

Not relying on boot-order and chance. ✓
Explicitly set consumer offsets to the beginning. ✓

Kafka REST Proxy

Using the kafka-console-consumer, weather messages are visible in the weather topic and are regularly produced as the simulation runs.

Good job, temperature and status are included with correct data type.

Using the Kafka Schema Registry REST API, a schema is defined for the weather topic.

Good, Content-Type header is appropriately set to the Avro type in your code.

Kafka Connect

Using the kafka-console-consumer, all stations defined in Postgres are visible in the stations topic.

Using the Kafka Connect REST API, the Kafka Connect configuration is configured to use JSON for both key and values.

Using the Schema Registry REST API, the schemas for stations key and value are visible.

Using the Kafka Connect REST API, the Kafka Connect configuration uses an incrementing ID, and the ID is configured to be `stop_id`.

Connector type is incrementing and based on stop_id.

Faust Streams

A consumer group for Faust is created on the Kafka Connect Stations topic.

Good job, you have correctly set the faust_out topic and in server.py as well.

Data is ingested in the `Station` format and is then transformed into the `TransformedStation` format.

Good work, Correct value type is used for the ingestion topic.

A topic is present in Kafka with the output topic name the student supplied. Inspecting messages in the topic, every station ID is represented.

Station ID is used as the table key, and set to the transformed station model.

KSQL

Using the KSQL CLI, turnstile data is visible in the table `TURNSTILE`.

Good you have correctly set the table name.

Using the KSQL CLI, verify that station IDs have an associated `count` column.

you should set the count as a column instead of `turnstile_count` as you are fetching `count` from Kafka message in the consumer.

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review