Kafka POC

**Problem statement**

A real time analytics capability in Vantage for data collected from streaming data sources. Latency (most important) and throughput (secondary to latency).

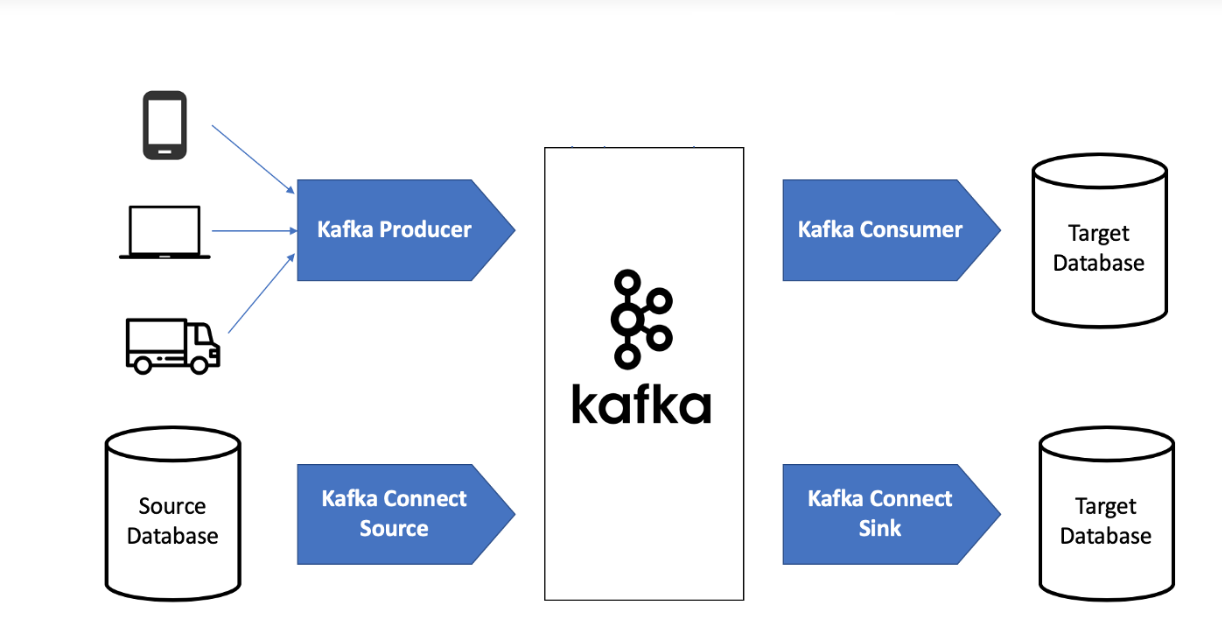
**Using custom connector with Kafka**

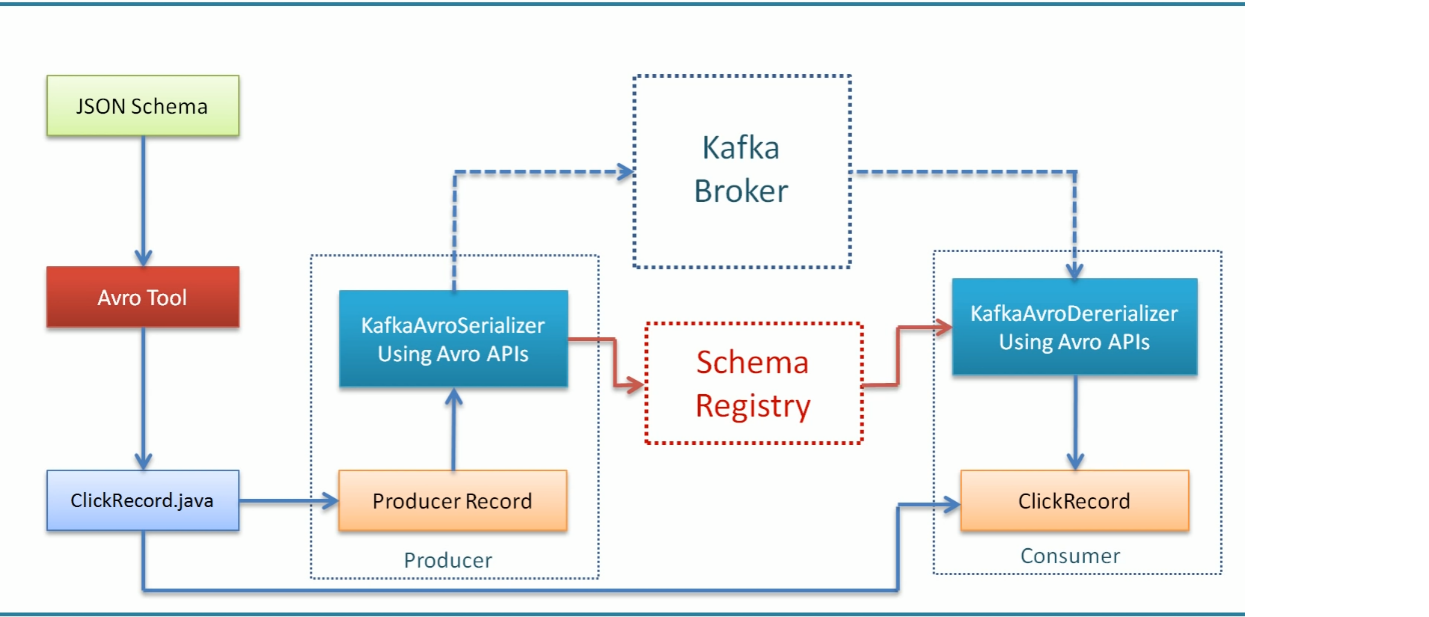
This option is to build a custom kafka connector to consume from a Kafka topic and deliver the data to Teradata using TeraJDBC driver.

Challenges

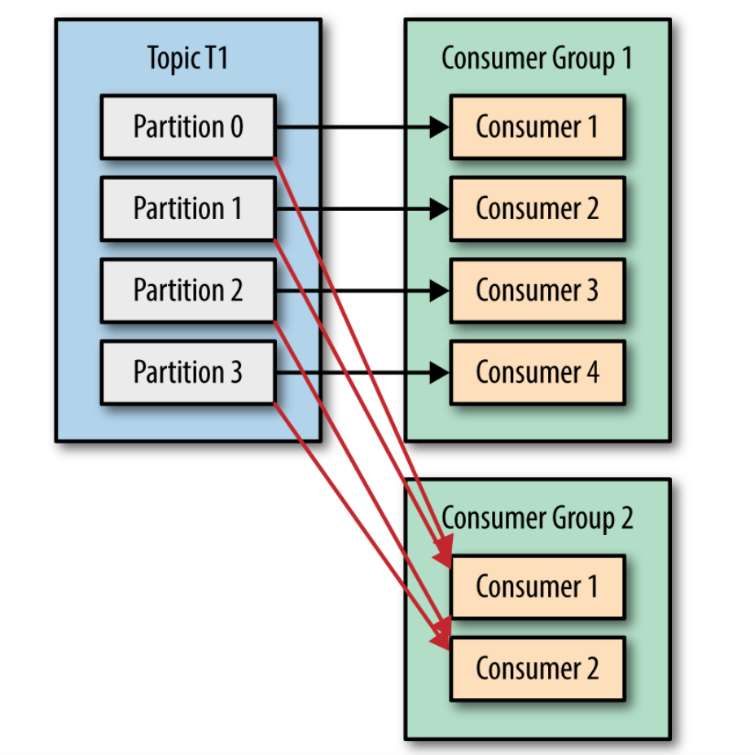
* Highly available and scalable model is required to meet the streaming data requirement
* Manage the schema for ingestion to Teradata table. The data schema could be different for different data **pipeline**. A pipeline describes the flow of data from the origin system to destination systems and defines how to transform the data along the way.
* Classical problem of meeting consumer rate of consuming to meet the producer’s rate of production.

Design





* The scalable Kafka consumer group will be created, and it will subscribe to a Kafka topic.
* The messages are replicated and persist in Kafka cluster
  + If consumer goes down and upon restart will not miss the messages.
  + If broker goes down, the message is replicated across cluster. Thus, the message is not lost.
* The scaling happens at both ends
  + Multiple producers can producers can write to same topic
  + Multiple consumers can consume from same topic
* Consumer group model



* Consumer rebalance use-case

If the consumer stops sending heartbeats for long enough, its session will time out and the group coordinator will consider it dead and trigger a rebalance. If a consumer crashed and stopped processing messages, it will take the group coordinator a few seconds without heartbeats to decide it is dead and trigger the rebalance. During those seconds, no messages will be processed from the partitions owned by the dead consumer. When closing a consumer cleanly, the consumer will notify the group coordinator that it is leaving, and the group coordinator will trigger a rebalance immediately,

* Data schema management
  + Schema management solves the problem where **Data consumers may not understand data producers** i.e. lack of structure makes consumption of data challenging.
  + Confluent Kafka (commercial version) uses a HA, scalable schema registry cluster to manage the schema.
  + Schema Registry is a distributed storage layer for Avro Schemas which uses Kafka as its underlying storage mechanism
  + It provides a RESTful interface for storing and retrieving Avro schemas
* Data serialization and deserialization
  + Avro schema, for example, to serialize into bytes, and deserialize these bytes back into the object.
* Advantages of streaming with Kafka connectors, connect Sink API allows you to leverage the ecosystem of existing Kafka Connectors out there to perform your streaming ETL
  + Kafka Connect runtime offers automatic balancing of the work, auto-failover, dynamic scaling (up or down), fault-tolerance, in-built offset management for auto-recovery
  + Kafka Connect integrates with the schema registry to capture schema information from sources if present.

What does the POC achieve

* Create a custom Kafka consumer to connect to a Kafka topic and ingest data to Teradata database
* The JSON data schema is used to serialize and deserialize the data
* AVRO used as serialization and deserialization library
* A custom producer will push messages (JSON schema serialized with AVRO) to Kafka and the Consumer deserializes the data and push it to a table in database
* The schema and table are pre-created in Teradata database

System requirements

* Vantage 16.10.00.12
* Kafka cluster

Technology stack

* Confluent 5.1
* TeraJDBC 16.10.00.10

Performance benchmark

Next steps

* Improve using Kafka connector instead of Kafka consumer
* Deploy on AWS and process the message

Feedback / comments