Heart-rate Monitor App

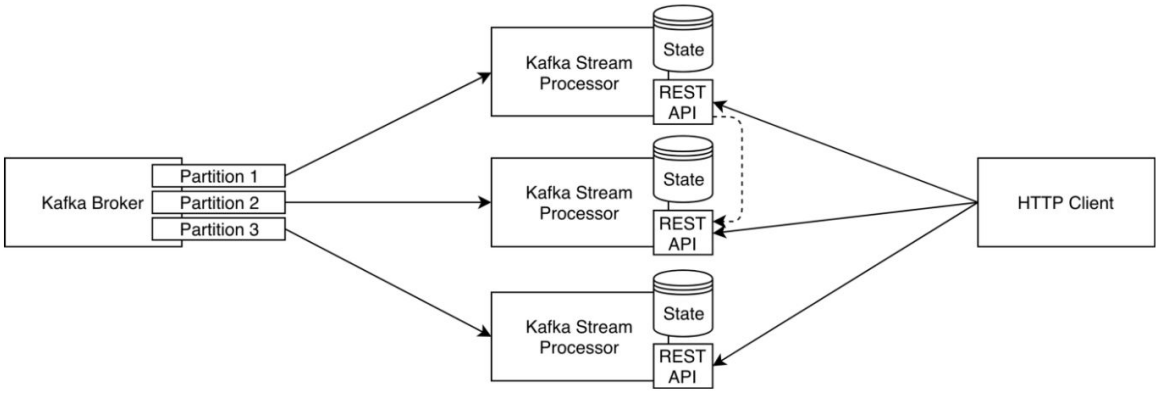
June 30 , 2020

Application Overview

This application continuously processes the message collected from multiple iot devices and exposes REST endpoint to access the metrics by individual device or collection of devices.

Architecture

This application is built on Kafka Streams Interactive Queries. Below is the bird's eye view of the architecture.



Kafka Streams API - Kafka Streams is a client library for building applications and microservices, where the input and output data are stored in Kafka clusters.

Jetty - Servlet Container for serving the rest end-points.

Jersey - RESTful Web Services framework is open source, production quality, framework for developing RESTful Web Services

Maven - To compile, build and packaging.

Details

The data from the iot devices are stored, processed and stored in the State Stores of Kafka.

Rest end-points are created to expose the data stored in the store using interactive query features of kafka streams.

Below are the key components:

- **Heart-rate Data Producer Service** — Pushes *simulated data to kafka on regular intervals and pushes to a topic created in Kafka*
- **Data Consumer Service** — Listens to a topic and processes the data and generates different metrics like count, minimum, maximum on the data received and stores them in the state stores.
- **REST endp-points -** These endpoints fetch data from the store using interactive queries exposed by kafka streams. These endpoints are created using Jersey implementation of JAX-RS and hosted on Jetty - an embedded servlet container. The endpoints are secured and can be accessed using a bearer token which can be generated from the authentication end-point.

Improvements to consider:

- The topics are created on a single kafka instance on a single partition. Using service discovery, stores spanned across multiple kafka instances can be fetched and processed.
- Producers can be easily tweaked to produce data to multiple topics
- Data is represented as Strings for simplicity. Complex data can be processed using avro and schema registry.
- Application is packaged as a single unit for simplicity. Producers and consumers and rest services can be decoupled and run independently and can be scaled horizontally.
- Authentication layer can be replaced with OAuth2 or SAML or using API keys