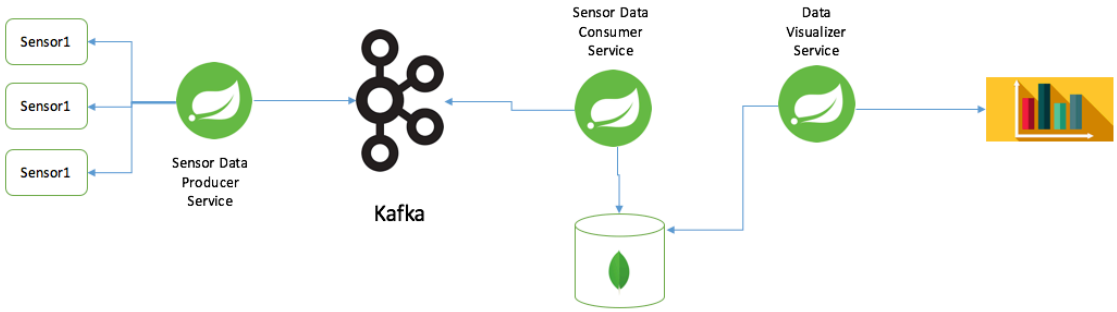
Processing IoT Device Data.



Approach:

Sensor Data Producer Service

1. Publish Data to the Kafka broker using Spring Boot micro service as a producer
    - Three asynchronous java process used to simulate the three IoT devices
    - Two Rest endpoints use to start and stop producing data through simulated devices
        i. /sensor/start
        ii. /sensor/stop
2. Using Spring micro service, produce these sensor data as a messages to the Kafka topic, Kafka use due to its ability to handle high throughput of streaming data

Sensor Data Consumer Service

1. Sensor Data Consumer Service, consume messages from Kafka topic persist in mongo database for later process.
2. If requires real time processing, it is better to use big data processing framework like Hadoop with some large data analytical tool such as Apache Spark that is not considered for the scope of this assessment.
3. Mongo like document DB selected here for easily persist the message template as it is and also the ability of horizontal scaling.
4. Better options like Cassendra, Couchbase can use over the mongo when comparing the features of scalability.

Data Visualizer Service

1. Process the data stored in a mongo database and expose via rest services
2. Those rest services are secured with spring security with JWT authentication.
3. JWT authentication configured from scratch.

Following Technologies used in the project

- Spring Boot
- Java 11
- Apache Kafka
- MongoDB
- Spring Security
- Spring Data
- Docker
- Maven

❖ Micro services architecture uses due to the scalability requirement, if required to handle high throughput of data we can scale the service horizontally in the containerized deployment environment like kubernetes.

Instruction to run the project.

- Unzip the project.
- Execute the build script(./build.sh) in the parent folder, change the permission if required.
- Create a database in mongo "sensor-data"
- Docker and Maven should be configured in the machine

API Documentation

Base URLs

Producer Service	http://localhost:8081
Consumer Service	http://localhost:8082
Visualization Service	http://localhost:8083

Rest APIs

Service	Operation	Endpoint	Method
Producer Service	Start Producing Data	/sensor/start	GET
Producer Service	Stop Producing Data	/sensor/stop	GET
Visualization Service	Register User	/register	POST
Visualization Service	Authenticate	/auth	POST
Visualization Service	Get Result	/data/sensor-reading	POST

1. http://localhost:8081/sensor/start
    a. Request : none

b. Response : Scheduling Started

2. http://localhost:8081/sensor/stop
   a. Request : none
   b. Response : Stop Producing Data

3. http://localhost:8083/register
   a. Request :

```
{
        "username":"janaka",
        "password":"password"
}
```

   b. Response :

```
{
  "id": "d5a28cc0-a20f-4ee9-8a49-cfad088e2d77",
  "username": "janaka"
}
```

4. http://localhost:8083/auth
   a. Request :

```
{
        "username":"janaka",
        "password":"password"
}
```

   b. Response :
      JWT token

5. http://localhost:8081/data/sensor-reading
   a. Request :

```
{
   "valueType": "MIN",
   "sensorType": "THERMO_METER"

}


   deviceType – THERMO_METER, FUEL_METER, HEART_RATE_METER
   queryType – MIN,MAX,AVERAGE
```

   a. Response:

```
{
    "valueType": "MAX",
    "sensorType": "THERMO_METER",
    "value": "78"
}
```

Limitations

- This project developed as a prototype to demonstrate the ability of data streaming using latest cloud technologies

- Only the minimum tools required to up and running the project are using here
- Docker level deployment use and not considered the micro services scalability

Encasements

- If required to build a production ready system to handle big amount of data produced by series of devices, it will be better to use MQTT bridge or proxy with Kafka
- Hadoop, Apache Spark, Apache storm, Couchbase, Apache Cassendra are better options to process big amount of streaming
- Different type of authentication mechanism such as oAuth, SSO, Auth2, SAML also a good fit to secure the endpoints.