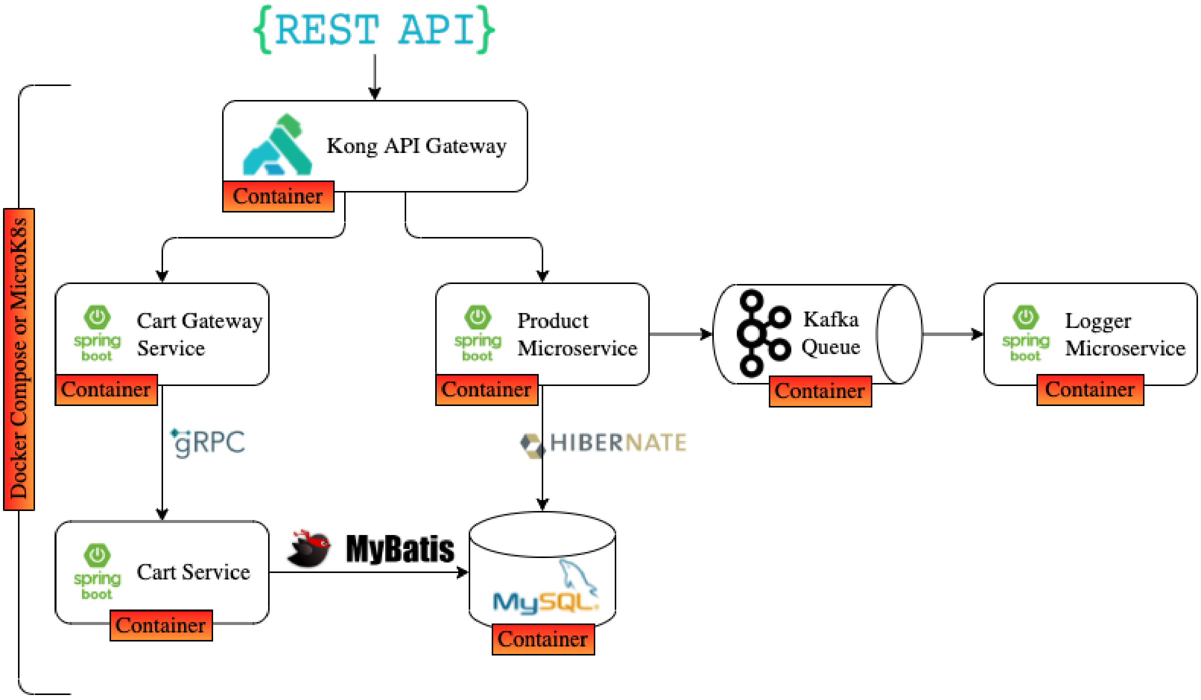
POC Application

Activity

Build microservices using spring boot including java 8 / 11 features. Kong is used as the API gateway that routes the requests to either CartGatewayService or ProductMicroservice. ProductMicroservice communicates with LoggerMicroservice via Apache Kafka; while CartGatewayService communicates with CartService using gRPC calls.

Architecture



[1]

Product Microservice

- Spring boot microservice exposing rest endpoints for Products (listing ,addition of products & booking of products)
- Accepts orders.
- Broadcast all the activities to Kafka for the logger Microservice to log
- Database Used is MySQL
- Hibernate is the ORM framework used to interact with MySQL.

Activity Logger

- Spring boot microservice to process all the activity logs.
- which listens for events and stores in NoSql dB.

Cart Gateway Microservice

- Spring boot microservice exposing rest endpoints for Cart related operations (create new cart, update cart by addition or deletion of products from the cart, get cart, etc).
- Invokes the corresponding methods of CartService using gRPC calls.

Cart Microservice

- Spring boot Microservice exposing methods (not REST endpoints) for Cart (listing, addition and deletion of products).
- Database Used is MySQL.
- MyBatis is the ORM framework used to interact with MySQL.

Requirement

- Product Microservice should expose POST, GET and PUT / PATCH API's.
- Cart Gateway Service should expose POST, GET and DELETE API's.
- Handling Validations: API's request validations should be performed and valid error/exception should be returned.
- Exception Handling: Microservices should have a proper exception and error handling across.
- Health Check: Microservices should include health check framework/ dependency.
- Swagger Documentation: Leverage swagger for API doc.
- Logging and Security: Use necessary implementation for logging and security (APO, Interceptors, Service mesh).
- Containerization: Use dockers to containerize applications. All the Microservices,Kafka, DB, and API Gateway should be running on Docker and can be on the same network using Docker Compose or deployed in MicroK8s.

[1]

- Additional folders / substructures can be assumed.
- Necessary tables can be assumed.
- Coding standards and Microservice best practices to be followed.

Code Structure

Microservice Code Structure should be as follows

```
 -      Controller

   -SampleController

    -DTO

      -RequestDTO

      -ResponseDTO

 -Service

       -SampleService

 -ServiceImpl

       -SampleImpService

 -DTO

      -SampleDTO

-Repository

    -SampleRepository

-Entity

-Exceptions

- Util

- Config
```

Deliverables

- Technical / Design document (design pattern and other considerations made).
- Github link (source code along with docker composer or K8 yaml files).

Note

- **API Gateway is optional (can run on a container for context-based routing).**
- **Install SonarLint extension to your IDE.**

Reference Documents:

- **SpringBoot integration with Apache Kafka - https://www.baeldung.com/spring-kafka**
- **Integrating REST APIs and gRPC within microservice architecture - https://techdozo.dev/restful-api-gatewat-with-grpc/**
- **Using MyBatis - https://www.baeldung.com/spring-mybatis**
- **Kong API Gateway - https://www.baeldung.com/kong**