Delivery Report

On

Build reference implementation using Event Sourcing DB and Integration events using Kafka

Infrastructure setup

1. Java, Docker
2. Zookeeper, Apache Kafka, Kafka UI
3. Postgres, PgAdmin
4. EventStoreDB

Source Code

Developed and attached here

Replay functions

The PaymentCommandHandler and PaymentEventHandler classes both read all the events for a payment from the PaymentEventStore and create a new instance of the Payment aggregate using the events. This is effectively a replay of all the events for the payment up to the current point in time.

Integration events in kafka

The PaymentIntegrationEventSender class is responsible for sending integration events to a Kafka topic. When a new integration event is generated, the send() method of the PaymentIntegrationEventSender class serializes the event data and creates a ProducerRecord object, which is then sent to the Kafka topic using the KafkaProducer.send() method.

Audit History in event sourcing db

the PaymentEventStore class is responsible for interacting with the event store to read and write events. The append() method of the PaymentEventStore class appends a new event to the event store for a given payment. The readEvents() method of the PaymentEventStore class reads all events for a given payment from the event store.

When an event is appended to the event store, it is stored with a unique identifier, a timestamp, and any metadata associated with the event. This information provides a complete audit trail of all the changes that have occurred in the system.

Run the project

1. Download & install OpenJDK 11 (LTS) at AdoptOpenJDK.
2. Download and install Docker and Docker Compose.
3. Build Java project and Docker image
   ./gradlew clean build jibDockerBuild -i
4. Run Kafka, ksqlDB and event-sourcing-app
   docker-compose up -d --scale event-sourcing-app=2

Microservice documentation

Payment service

URL: http://localhost:8080/payment/

Method: POST

Request body:


```
{
    "accountId": "22770803-38f4-4594-aec2-4c74918f7111",
    "status": "PAYMENT_SUCCESSFUL",
    "account": [
        {
            "accountId": "22770803-38f4-4594-aec2-4c74918f7111",
            "address": "Address 1",
            "name": "Account 1",
            "balance_due": 1234.56
        },
        {
            "accountId": "33770803-38f4-4594-aec2-4c74918f7111",
            "address": "Address 2",
            "name": "Account 2",
            "balance_due": 4567.56
        }
    ]
}
```


Response:


```
{
    "paymentId": "f89aef7b-6338-49f0-8ba5-cfc52a4422ca"
}
```


Code: 202

Payment Resolution service

URL: http://localhost:8080/payment/{paymend_id}

Method: PATCH

Request body:


```
{
    "accountId": "11770803-38f4-4594-aec2-4c74918f7111",
    "revision":0,
    "amount": "123456.78"
}
```


Response:

Code: 202