**Day4: CQRS Pattern**

**Lab Structure**

cqrs-kafka-springboot/

│

├── order-command-service // Spring Boot project for writing orders

├── order-query-service // Spring Boot project for reading orders

└── docker-compose.yml // Kafka & Zookeeper

**Prerequisites**

* IntelliJ IDEA
* Docker Desktop (for Kafka)
* Java 17+
* Spring Boot 3.x
* Postman or curl

**Step 1: Kafka Setup**

**docker-compose.yml in root folder**

version: '3.8'

services:

zookeeper:

image: confluentinc/cp-zookeeper:latest

ports:

- "2181:2181"

environment:

ZOOKEEPER\_CLIENT\_PORT: 2181

kafka:

image: confluentinc/cp-kafka:latest

ports:

- "9092:9092"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://localhost:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

Run the Kafka and Zookeeper from docker

docker-compose up -d

**Step 2: order-command-service – Producer (Write Side)**

**Create project using Spring Initializr (via IntelliJ)**

* **Dependencies**: Spring Web, Spring Kafka

**OrderCreatedEvent.java**

package com.commandservice.model;

import java.time.LocalDateTime;

import java.util.UUID;

public class OrderCreatedEvent {

private UUID orderId;

private String product;

private double price;

private LocalDateTime createdAt;

// Getters and setters

}

**KafkaProducerService.java**

package com.commandservice.kafka;

import com.commandservice.model.OrderCreatedEvent;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.springframework.kafka.core.KafkaTemplate;

import org.springframework.stereotype.Service;

@Service

public class KafkaProducerService {

private final KafkaTemplate<String, String> kafkaTemplate;

private final ObjectMapper objectMapper = new ObjectMapper();

public KafkaProducerService(KafkaTemplate<String, String> kafkaTemplate) {

this.kafkaTemplate = kafkaTemplate;

}

public void sendOrder(OrderCreatedEvent event) {

try {

String msg = objectMapper.writeValueAsString(event);

kafkaTemplate.send("order-topic", msg);

} catch (Exception e) {

e.printStackTrace();

}

}

}

**OrderController.java**

package com.commandservice.controller;

import com.commandservice.kafka.KafkaProducerService;

import com.commandservice.model.OrderCreatedEvent;

import org.springframework.web.bind.annotation.\*;

import java.time.LocalDateTime;

import java.util.UUID;

@RestController

@RequestMapping("/api/orders")

public class OrderController {

private final KafkaProducerService producerService;

public OrderController(KafkaProducerService producerService) {

this.producerService = producerService;

}

@PostMapping

public String createOrder(@RequestBody OrderCreatedEvent order) {

order.setOrderId(UUID.randomUUID());

order.setCreatedAt(LocalDateTime.now());

producerService.sendOrder(order);

return "Order sent to Kafka";

}

}

**application.yml**

server:

port: 8081

spring:

kafka:

bootstrap-servers: localhost:9092

**Step 3: order-query-service – Consumer (Read Side)**

**Create new Spring Boot project**

* **Dependencies**: Spring Web, Spring Kafka

**OrderCreatedEvent.java (same as in producer)**

Place this in com.queryservice.model

**KafkaConsumerService.java**

package com.queryservice.kafka;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.queryservice.model.OrderCreatedEvent;

import org.springframework.kafka.annotation.KafkaListener;

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.concurrent.CopyOnWriteArrayList;

@Service

public class KafkaConsumerService {

private final List<OrderCreatedEvent> orders = new CopyOnWriteArrayList<>();

private final ObjectMapper mapper = new ObjectMapper();

@KafkaListener(topics = "order-topic", groupId = "query-group")

public void consume(String message) {

try {

OrderCreatedEvent order = mapper.readValue(message, OrderCreatedEvent.class);

orders.add(order);

} catch (Exception e) {

e.printStackTrace();

}

}

public List<OrderCreatedEvent> getAllOrders() {

return orders;

}

}

**OrderQueryController.java**

package com.queryservice.controller;

import com.queryservice.kafka.KafkaConsumerService;

import com.queryservice.model.OrderCreatedEvent;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/orders")

public class OrderQueryController {

private final KafkaConsumerService consumerService;

public OrderQueryController(KafkaConsumerService consumerService) {

this.consumerService = consumerService;

}

@GetMapping

public List<OrderCreatedEvent> getOrders() {

return consumerService.getAllOrders();

}

}

**application.yml**

server:

port: 8082

spring:

kafka:

bootstrap-servers: localhost:9092

consumer:

group-id: query-group

auto-offset-reset: earliest

**Test Flow**

1. Run both services from IntelliJ.
2. Kafka must be running via Docker.
3. Use Postman:

**Create Order**

POST http://localhost:8081/api/orders

{

"product": "Laptop",

"price": 95000

}

**Get Orders**

GET http://localhost:8082/api/orders

**Summary**

* CommandService publishes to Kafka
* QueryService consumes and stores in memory
* CQRS: Write & read sides separated