Solutions

[**https://github.com/mevikas4/TradeStore-Consumer**](https://github.com/mevikas4/TradeStore-Consumer)

[**https://github.com/mevikas4/TradeStore-Producer**](https://github.com/mevikas4/TradeStore-Producer)

I have developed a Java application using Spring Boot and Kafka that integrates with both SQL and NoSQL databases by building a producer-consumer model Using Kafka. A REST endpoint can trigger a Kafka producer to send an event, and a Kafka consumer will then process that event and store the data in both databases.

SQL Database (e.g., H2 Database): Stores structured, transactional data, such as Trade details. This layer uses Spring Data JPA.

NoSQL Database (e.g., MongoDB): Manages unstructured or event-based data, such as Orders. This layer uses Spring Data MongoDB.

To develop a trade store that organizes and stores each trade in a specific sequence.

I have used kafka topic with single partition - Kafka guarantees message order within a single partition.

Steps to Create a Topic:

cd /path/to/kafka/bin

.\kafka-topics.sh --create --topic my\_first\_topic1 --bootstrap-server localhost:9093 --partitions 1 --replication-factor 1

Added Unit tests using (JUnit, Mockito)

PlantUML design diagrams

Class diagram

This diagram illustrates the main application components and their relationships with the SQL and NoSQL database layers.

Sequence diagram:

This diagram visualizes how a REST call leads to a Kafka event and subsequent database operations.

Deployment pipeline using GitHub Actions

A CI/CD pipeline can be defined in a .github/workflows/pipeline.yml file to automate building, testing, and security scanning.

When Storing Trade in Trade store added below checks

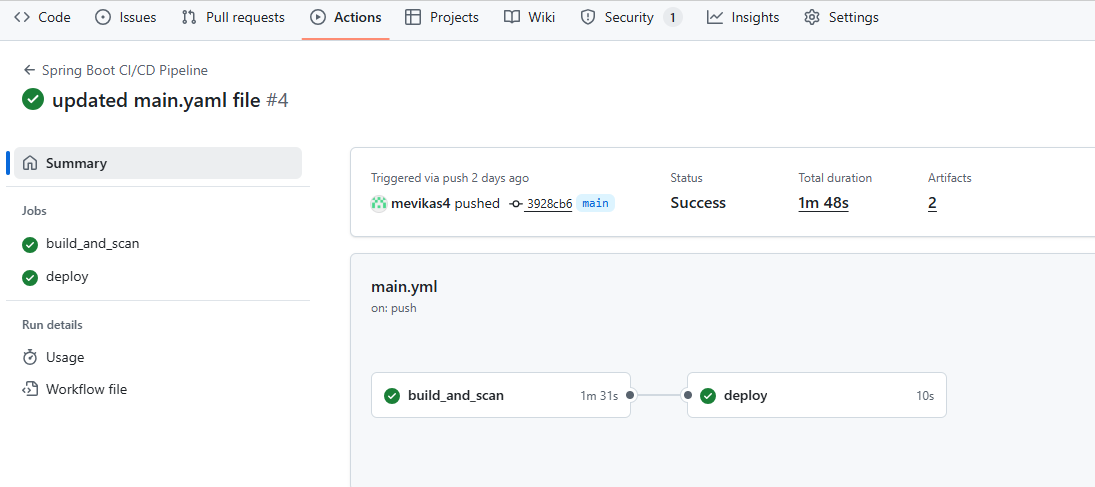
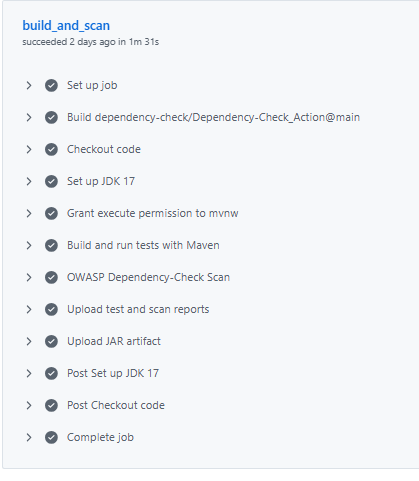
If a trade with a lower version is received during transmission, the store will reject it and generate an exception. Trades with the same version will replace the current record.

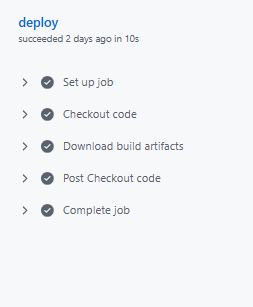
The store must reject any trade that has a maturity date earlier than today's date.

When a trade's maturity date is surpassed, the store should automatically mark the trade as expired.

Added Scheduler for Trade expiry

@Scheduled(fixedRate = 10000) *// Runs every 5 seconds* public void runFixedRateTask() {  
 System.*out*.println("Fixed Rate Task executed!");  
 tradeService.updateExpiredTrades();  
 }  
  
*/\* @Scheduled(cron = "0 0 10 \* \* ?") // Runs every day at 10:00 AM  
 public void runCronTask() {  
 System.out.println("Cron Task executed!");  
 tradeService.updateExpiredTrades();  
 }\*/*



**# TradeStore-Consumer application**

cd C:\kafka\kafka\_2.12-2.5.0>

.\bin\windows\zookeeper-server-start.bat config\zookeeper.properties

check zookeeper is up and running - netstat -ano | findstr 2181

after making sure that zookeper is up, open another CMD in the downloaded folder and paste the command below:

.\bin\windows\kafka-server-start.bat config\server.properties

http://localhost:8083/h2-console/

for NO SQL Database - Used Mongo DB Atlas

Cluster0

mevikas4\_db\_user

JYSb3ydEKOCJAKk9

mongodb+srv://mevikas4\_db\_user:JYSb3ydEKOCJAKk9@cluster0.ghgk1bh.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

http://localhost:8084/order

{

"tradeId": "1",

"version": 1,

"counterPartyId": "Vikas ji",

"bookId": "B2",

"maturityDate": "2025-09-09",

"createdDate": "2025-09-06",

"expired": false

}

In case use as rest controler

http://localhost:8080/trade/1

http://localhost:8080/trade

{

"tradeId": "1",

"version": 1,

"counterPartyId": "Vikas Nidhi ji",

"bookId": "B2",

"maturityDate": "2025-09-07",

"createdDate": "2025-09-06",

"expired": false

}

**# TradeStore-Producer application**

http://localhost:8080/send

{

"tradeId": "2",

"version": 1,

"counterPartyId": "Vikas Nidhi ji",

"bookId": "B2",

"maturityDate": "2025-09-06",

"createdDate": "2025-09-06",

"expired": false

}