Warehouse Service System

System Design

Our simple Warehouse Service System consists of the two following microservices:

- Warehouse Service
- Central Service

Warehouse Service
A simple server application, listening to UDP protocol on the ports specified and forwarding the messages received to the Message Broker onto the specified message topic.

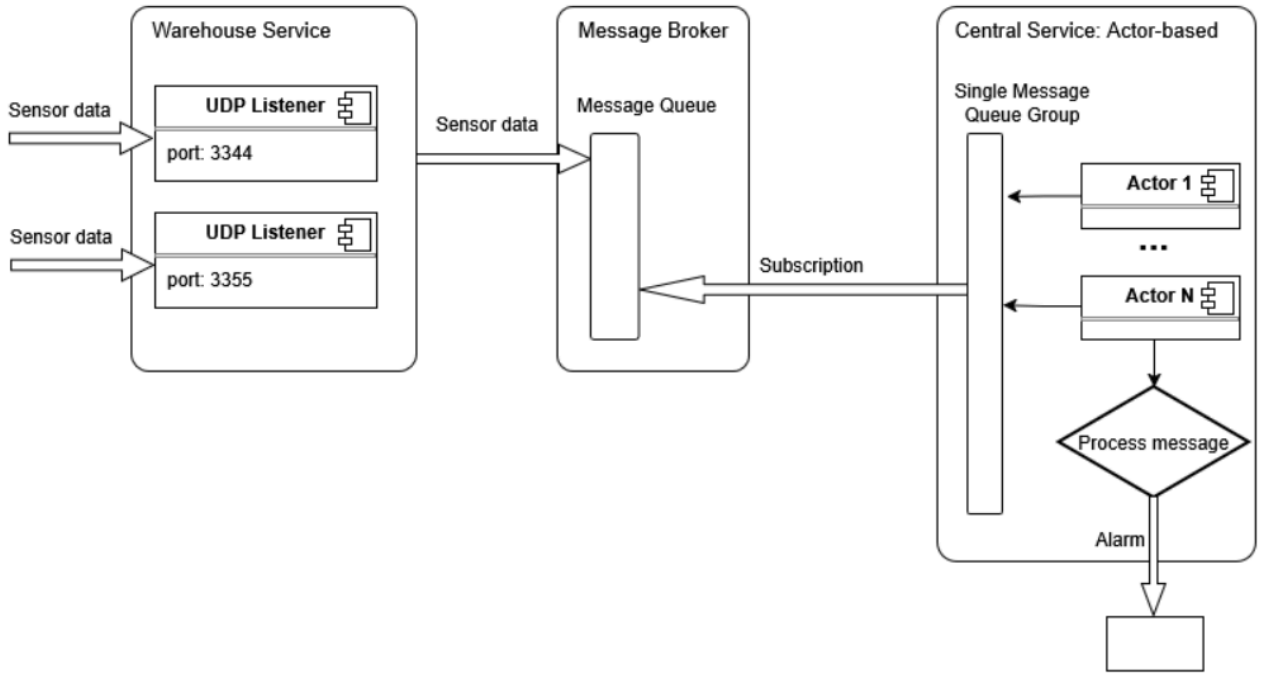
For simplicity, a decision was made not to use common popular frameworks. Instead Warehouse Service has been built using Java 21 Virtual Threads, which is practically one of the most lightweight solutions, yet works well from both I/O and scalability perspectives.

Central Service
A simple server application, subscribed to the aforementioned message topic, parsing and validating the messages received, comparing received values to the configured thresholds and logging alarms in the console if the value exceeds the threshold.

Experimentally, Central Service has been built as an Actor-based Java Akka application, given that Akka is also a lightweight, efficient thread-safe and scalability-proven framework.

Message send/receive
For communication between the two microservice applications above we are using NATS broker – a lightweight and one of the most performance-efficient message brokers.

Overview diagram



Sources/Tests

The source code for the two applications is located inside the warehouse-service and the central-service folders respectively. Key functionality has been covered with unit/integration tests in both projects.

Building/Running

Warehouse Service
Inside the warehouse-service directory
Build: `gradlew clean build`
Run: `java -jar build/libs/warehouse-service-1.0.0-all.jar`

Central Service
Inside the central-service directory
Build: `gradlew clean build`
Run: `java -jar build/libs/central-service-1.0.0-all.jar`

NATS
Run NATS locally using the following command to ensure it starts with JetStream:
`nats-server.exe -js`

Test UDP messages
You can use the **send-udp.bat** file located in the root directory to send UDP messages on Windows.