Building a Secure and Scalable SOA System with API Gateway, NGINX, RabbitMQ and Kafka

Kaisser Thomas

January 4, 2026

1 Introduction

This tutorial presents a working Service-Oriented Architecture (SOA) system implemented using microservices and modern cloud-native technologies. The goal is to demonstrate:

- Secured REST APIs using JWT authentication
- Horizontal scalability via multiple API Gateways and NGINX load balancing
- Asynchronous communication using RabbitMQ
- Event streaming using Apache Kafka

The complete source code is available as a public GitHub repository.

2 System Overview

The system consists of the following components:

- **API Gateway** (Spring Cloud Gateway)
- **User Service** (Authentication and registration)
- **Notification Service**
- **RabbitMQ** (message broker)
- **Kafka** (event streaming platform)
- **NGINX** (load balancer)

All services are containerized using Docker and orchestrated via Docker Compose.

3 Securing the REST API with JWT

Authentication is handled by the User Service. When a user registers or logs in, a JSON Web Token (JWT) is generated.

3.1 JWT Flow

1. Client sends credentials to `/auth/register`
2. User Service validates input and creates the user
3. JWT token is generated and returned on login
4. Client includes token in `Authorization` header
5. API Gateway validates JWT for protected routes

3.2 Gateway JWT Filter

```
if (!authHeader.startsWith("Bearer ")) {
    exchange.getResponse().setStatusCode(HttpStatus.UNAUTHORIZED)
       ;
    return exchange.getResponse().setComplete();
}
jwtUtil.validateToken(token);
```

This ensures that only authenticated requests reach internal services.

4 Load Balancing with NGINX and API Gateways

To achieve scalability, two API Gateway instances are deployed.

4.1 NGINX Configuration

NGINX distributes incoming requests across gateway instances:

```
upstream api_gateway {
    server api-gateway-1:8080;
    server api-gateway-2:8080;
}
```

This enables fault tolerance and horizontal scaling.

4.2 Gateway Instance Identification

Each gateway injects its instance ID into the response header:

```
X-Gateway-Instance: gateway-1
```

This proves load balancing behavior in Postman or browser requests.

5 Asynchronous Communication with RabbitMQ

After successful registration, the User Service sends a message to RabbitMQ.

5.1 RabbitMQ Usage

- User Service publishes a message: `UserRegisteredEvent`
- Notification Service consumes the message
- Notification is processed asynchronously

```
rabbitTemplate.convertAndSend("user.exchange", "user.registered",
    event);
```

This decouples services and improves reliability.

6 Event Streaming with Kafka

Kafka is used to stream user-related events for analytics and auditing.

6.1 Kafka Flow

1. User Service publishes event to Kafka topic
2. Notification Service subscribes to the topic
3. Events are processed independently

```
kafkaTemplate.send("user-events", event);
```

Kafka enables replayable and scalable event processing.

7 End-to-End Registration Flow

The complete registration flow is:

1. Client sends `/auth/register` request
2. NGINX forwards request to an API Gateway
3. API Gateway validates JWT (or allows public endpoint)
4. User Service registers the user
5. RabbitMQ message is sent
6. Kafka event is published
7. Notification Service processes both

This demonstrates synchronous and asynchronous communication combined.

8 Containerization

All components are deployed using Docker Compose.

- Each service runs in its own container
- Internal networking is handled by Docker
- Ports are exposed only where required

This setup closely resembles real-world microservice deployments.

9 Conclusion

This tutorial demonstrated how to build a secure, scalable SOA system using:

- JWT-secured REST APIs
- Load balancing with NGINX
- Message-based communication with RabbitMQ
- Event streaming using Kafka

The architecture fulfills all assignment requirements and provides a solid foundation for further extensions.

10 Source Code

GitHub Repository: `https://github.com/ThomasKSSR/soa-project`