# Spring Boot 3 Microservice Notes

- Spring Boot 3

- Integration Tests using Latest Testing Techniques

- Spring Cloud Feign for inter service communication

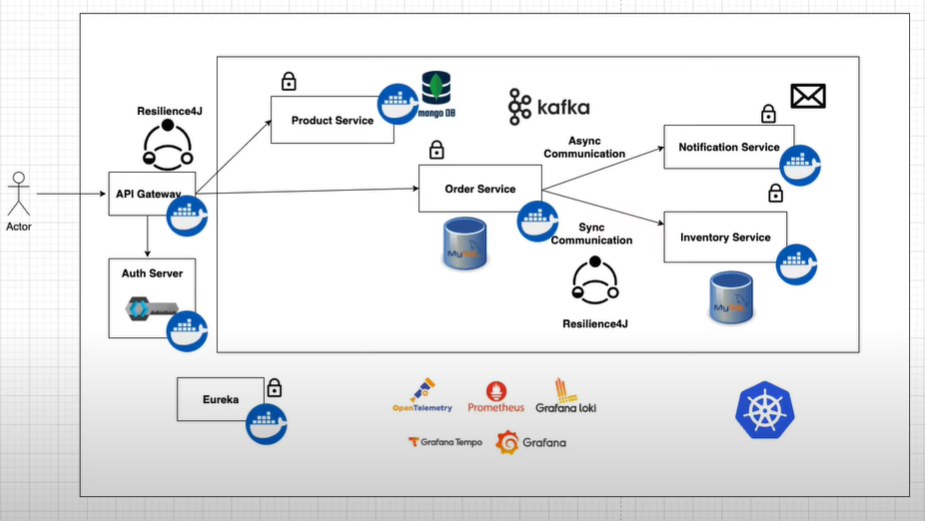
- API Gateway implementation using Spring Cloud Gateway MVC instead of Spring Cloud Gateway with Webflux

- Add Circuit Breaker Logic in API Gateway

- Security using latest Keycloak

- Distributed Tracing using Grafana Stack instead of Slueth

- Deployment using Kubernetes



4 Microservices:

1. Product Service: Product catalog, stored in MongoDB (NoSQL)

2. Order Service: handles orders, stored in MySQL (RDBMS)

3. Notification Service: stateless, no db, asynchronous using Kafka.

4. Inventory Service: handles inventory, synchronous using Resilience4J

- API Gateway: acts as the gateway to access microservices

- Auth Server: Keyloak, for app’s security.

- Eureka: server and client for registration of microservices

- Grafana: for monitoring and dashboard

- GrafanaLoko: for logs

- Grafana Tempo: for distributed tracing

- Prometheus: for metrics

- Containerized all services using Docker

- Run all Docker containers using Docker compose.

- Migrate all Docker compose workloads to Kurbenetes

Part 1:

- setup docker-compose.yml

- Started with product-service

Run the docker compose with this command: docker compose up -d

Part 2:

- setting up order-service

- something new with me is Flyway Migration dependency. It helps us version control database.

- we can write our database migration scripts under resources/db/migration. (This is automatically created when we have the pom dependency for flyway).

- naming convention of file should be “V<number>\_\_<name>.sql”

Part 3:

Microservice-to-Microservice Synchronous Communication:

A diagram of a process

Description automatically generated

- Learned about OpenFeign: helps with calling microservice-to-microservice.

Add this annnotation to register the app:

A screen shot of a computer program

Description automatically generated

This is how we register `inventory-service` to `order-service` to be able to call methods in inventory service inside order-service.

A screen shot of a computer program

Description automatically generated

\*\*Flyway migrations error: we can just run ./mvnw flyway:repair

- Integration testing using WireMock:

A screenshot of a computer

Description automatically generated

\*but I didn’t practice it in my code. Don’t have to.

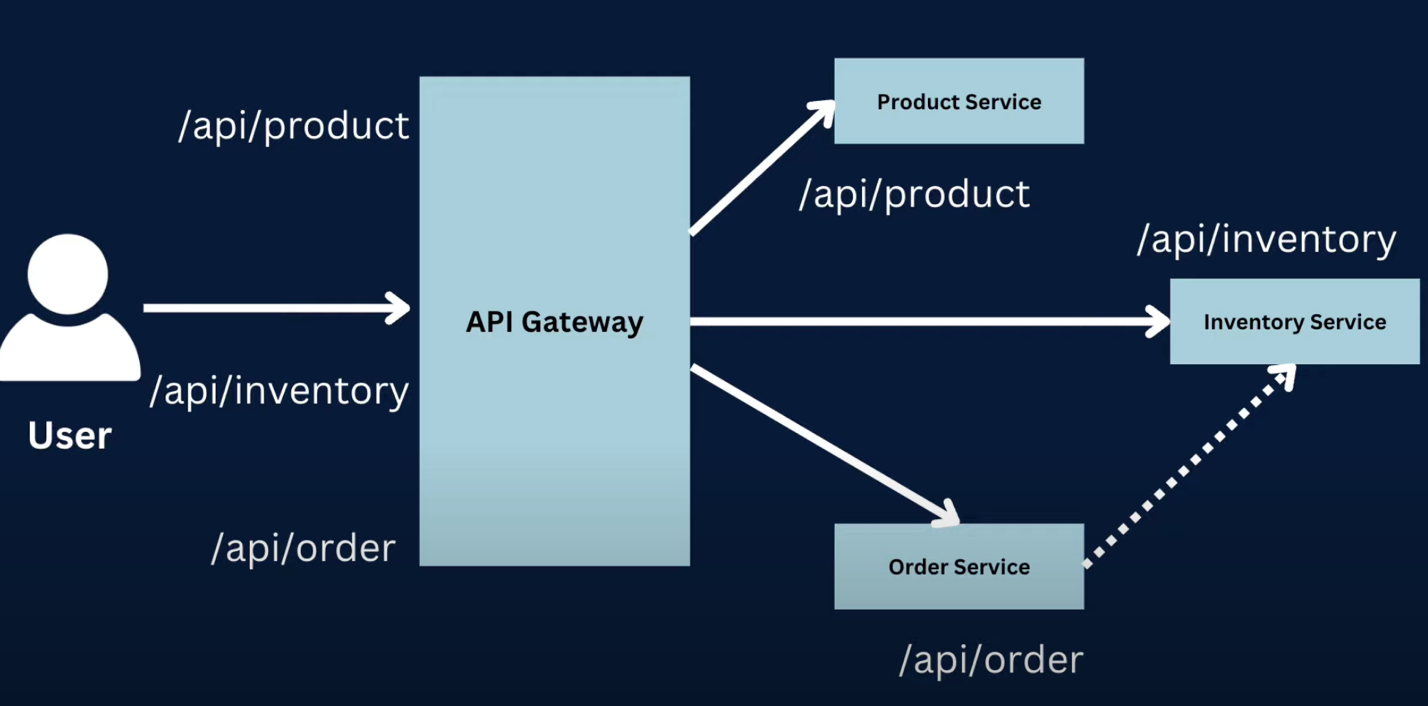
Part 5:

- API Gateway

- acts as an entry point for requests.

- forwards these request to the downstream microservices.

- commonly used in the distributed systems and microservices architecture.



# TO Dos:

* Better exception handling for better response body in Postman.
* Fix the logic of when the customer places an order, then we deduct the quantity to the inventory, ONLY IF the order is confirmed or shipped.
* Create api-gateway progmmatic code for routes
* Add order-service, inventory-service, and product-service configurations on config directory, then test calling
* Register eureka clients order-service, inventory-service, and product-service to eureka server (work on order-service and product-service not working and not being able to connect to mysql db or db).