Microservices

- Breaking down the single unit into different smaller module.
- Loosely couples
- Easy to deploy

ADVANTAGES
- It can’t affect other services when service is down
- Can use different DB or different programming language
- Easy to scale

Microservices Communication

1. Synchronous
- Rest Template
- Web Client
- Spring Cloud Open Feign

2. Asynchronous
- Apache Kafka
- RabbitMQ

Service Registry & Discovery Server
- Spring Cloud Netflix Eureka
- Registering microservices to eureka server
- Keeps the track of port number, url, name

Eureka Server: 8761

First go to maven-> select service -> lifecycle -> package (double click)
Now check target folder to see jar file
In terminal to run instances
Go to project folder and run
java -jar -Dserver.port=8082 target/department-service-0.0.1-SNAPSHOT.jar

API Gateway
- Routing

Config Server
- Keeping the configuration file in GIT

Distributed Tracing Using Spring Cloud Sleuth and Zopkin
- Depricated spring boot 3

Circuit Breaker using Resilience4J
- Fallback method

- Circuit breaker
- Retry