



# MSA

☰ 주차	28주차
📅 스터디 일자	@2024/05/23

## 마이크로서비스 아키텍처 (Microservices Architecture, MSA)

마이크로서비스 아키텍처에 대한 정확한 정의는 없지만, 마이크로서비스란 작고 독립적으로 배포 가능한 각각의 기능을 수행하는 서비스로 구성된 프레임워크라고 할 수 있다.

마이크로서비스는

완전히 독립적으로 배포가 가능하고, 다른 기술 스택(개발 언어, 데이터베이스 등)이 사용 가능한 단일 사업 영역에 초점을 둔다.

## MSA 의 특징

### 1. 낮은 서비스 간의 결합도

- 각 서비스는 독립적으로 개발, 배포, 확장될 수 있다.
- 최소한의 중앙집중식 시스템을 유지하며, 분산 시스템을 구축한다.

## 2. 독립된 기술 스택과 데이터 스토리지

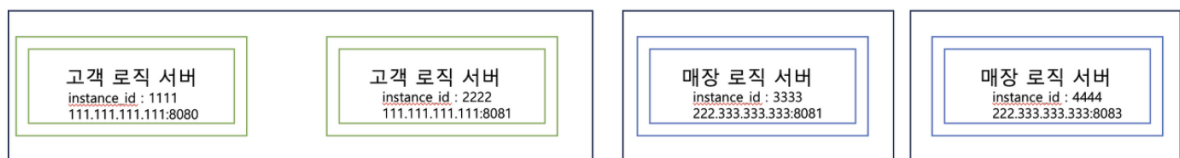
- 독립적으로 어떤 기능에 충실하여 개발하게 되기에, 굳이 같은 기술 스택을 사용하지 않고 Spring, Node.js, ASP.NET 등 **각각 다른 프레임워크**를 통해 개발할 수 있다.
- 이에 따른 데이터 스토리지도 공유하지 않고 **각각 운영하여** 의존성을 낮추고 **낮은 결합도**를 가질 수 있다.

### But !

데이터베이스의 공통 관심 영역 즉, 여러 마이크로서비스가 **동일한 비즈니스 엔터티**를 조작하고 서로 간의 일관성을 유지해야 하는 경우 데이터 동기화가 필요할 수 있다.

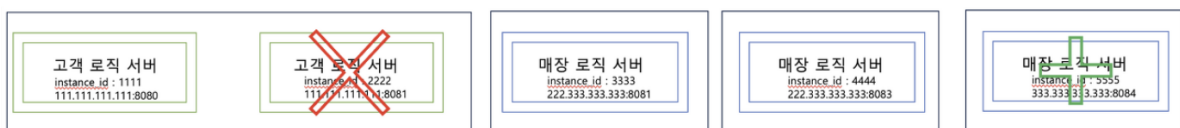
⇒ 이 때 **Apache Kafka** 와 같은 메시지 브로커를 이용하면 동기화가 가능하다.

## 3. 여러 개의 서비스 인스턴스



- 각각의 비즈니스 로직을 담당하는 서비스 인스턴스들이 여러 개가 생성되어 운영된다.
- 트래픽 처리를 위해서 **같은 비즈니스 로직을 처리하는 인스턴스** 또한 여러 개 생성할 수 있다.

## 4. 스케일 아웃(Scale out)



- 서비스의 부하에 따라 **수평적으로 서비스 인스턴스**를 확장하거나 축소할 수 있다.

## 5. Service Discovery (Naming Server)

- 각각의 인스턴스들의 IP와 포트 주소를 일일이 관리하기는 어려운 작업이며, IP와 Port가 변경되고, 인스턴스가 수시로 추가되거나 제거될 수 있다.

MSA에서는

동적인 환경에서 여러 개의 서비스 인스턴스를 관리해주는 **Naming Server**의 역할이 필요하다.

- Service Discovery는 각 서비스 인스턴스들을 추가 혹은 제거하고, 검색 및 **Health Check(상태 추적)**의 기능을 한다.

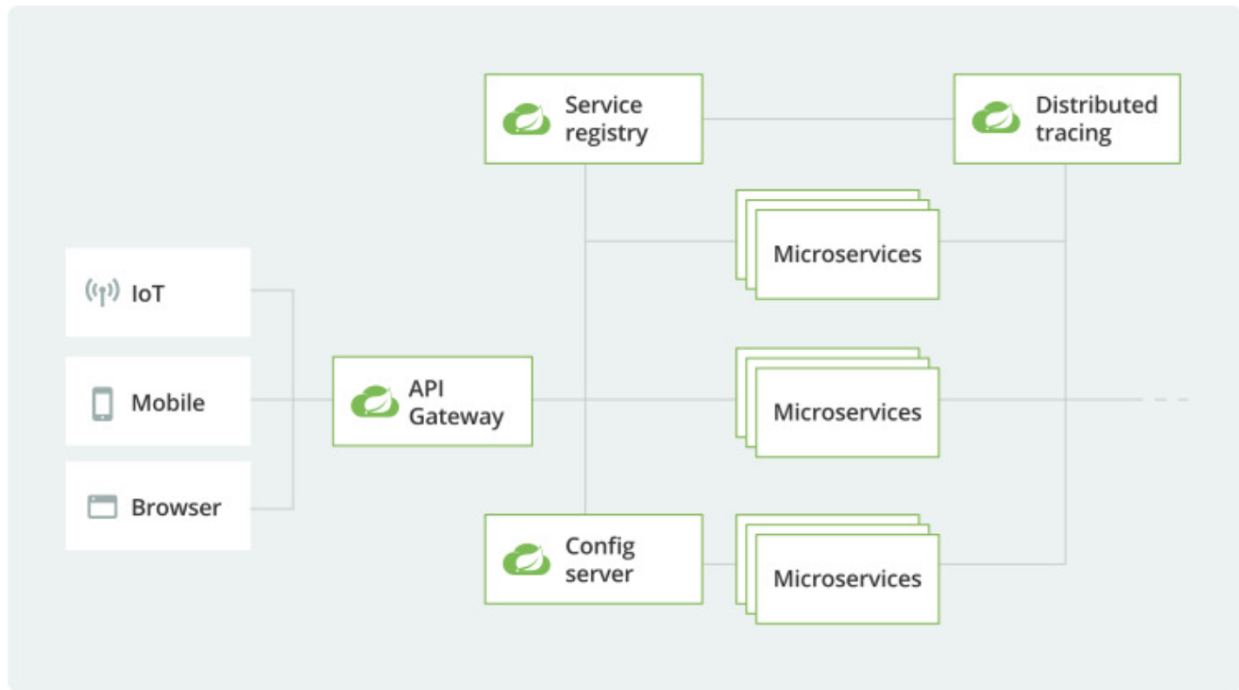
⇒ 이를 통해 클라이언트가 서비스에 접근할 때 필요한 정보를 제공한다.

## 6. Gateway와 Load Balancer

- Gateway
  - 클라이언트는 **단일 진입점**을 통해서 MSA의 인스턴스들에 접근 가능하며, Gateway는 Naming Server에 등록된 인스턴스의 정보를 기반으로 클라이언트의 요청을 **라우팅**해주는 역할을 한다.
- Load Balancer
  - 서비스의 트래픽을 여러개로 분산된 인스턴스에 나누어 주는 역할을 수행한다.
  - Load Balancer의 역할은 서비스 인스턴스의 상태를 모니터링하고, 이를 기반으로 트래픽을 분산시킨다.

# Spring Cloud

Spring Cloud는 **분산 시스템**과 **MSA**를 쉽게 개발하고 구축하기 위한 Spring 기반의 프레임워크 모음이다.



## Spring Cloud API Gateway

- 마이크로서비스 아키텍처에서 Gateway로서 **클라이언트의 단일 진입점 역할**을 한다.
- 인증, 권한 부여, 로드 밸런싱, 라우팅, 트래픽 제어 등의 기능을 제공한다.

## Spring Cloud Config Server

- 설정 관리를 중앙에서 효과적으로 관리할 수 있게 해주는 서버 역할을 한다.
- 각 마이크로서비스는 Config Server를 통해 자체적인 설정을 동적으로 가져올 수 있으며, 설정의 중앙 집중 관리로 유지보수 및 변경 관리를 용이하게 한다.

## Spring Service Registry (Naming Server)

- MSA에서 서비스 인스턴스의 등록과 검색을 담당하는 서버 역할을 한다.
- 서비스 인스턴스의 등록과 해제를 관리하며, 서비스 디스커버리를 통해 클라이언트는 특정 서비스 인스턴스를 동적으로 찾을 수 있다.
- 대표적인 예 : Netflix Eureka

## Distributed Tracing

- 마이크로서비스 아키텍처에서 여러 서비스 간의 트랜잭션을 추적하고 모니터링하는 기술
- 대표적인 도구 : Zipkin, Jaeger

# Spring Cloud 맛보기

## Spring Cloud 의존성

- Spring Cloud Netflix Eureka 클라이언트 의존성 추가한 예

```
implementation 'org.springframework.cloud:spring-cloud-starter-eureka'
```

### 💡 Eureka 란?

Eureka는 마이크로서비스들의 정보를 레지스트리에 등록할 수 있도록 하고 마이크로서비스의 동적인 탐색과 로드밸런싱을 제공한다.

## 게이트 웨이 (.yml) 설정

```
spring:
  application:
    name: gateway
  cloud:
    gateway:
      routes:
        - id: customer
          uri: lb://CUSTOMER
          predicates:
            - Path=/customer/**
        - id: coupon
          uri: lb://COUPON
          predicates:
            - Path=/coupon/**
```

## 유레카 서버

```
@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(EurekaServerApplication.class, args)
    }
}
```


## 서비스 서버

```

@SpringBootApplication
@EnableDiscoveryClient
public class CouponApplication {
    public static void main(String[] args) {
        SpringApplication.run(CouponApplication.class, args);
    }
}

```

## 유레카 서버 접속 시 (localhost:{설정된 포트})


HOME LAST 1000 SINCE STARTUP

### System Status

Environment	test	Current time	2023-07-28T19:51:07 +0900
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	8
		Renews (last min)	0

### DS Replicas

### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
COUPON	n/a (2)	(2)	UP (2) - localhost:coupon:8085, localhost:coupon:8081
CUSTOMER	n/a (1)	(1)	UP (1) - localhost:customer:8080
GATEWAY	n/a (1)	(1)	UP (1) - localhost:gateway:8000

출처

[https://seokbin.tistory.com/15#1.1 Spring Cloud%3F-1](https://seokbin.tistory.com/15#1.1_Spring_Cloud%3F-1)

<https://sjh9708.tistory.com/119>

<https://9hyuk9.tistory.com/100>