MSA

01 MSA

MSA

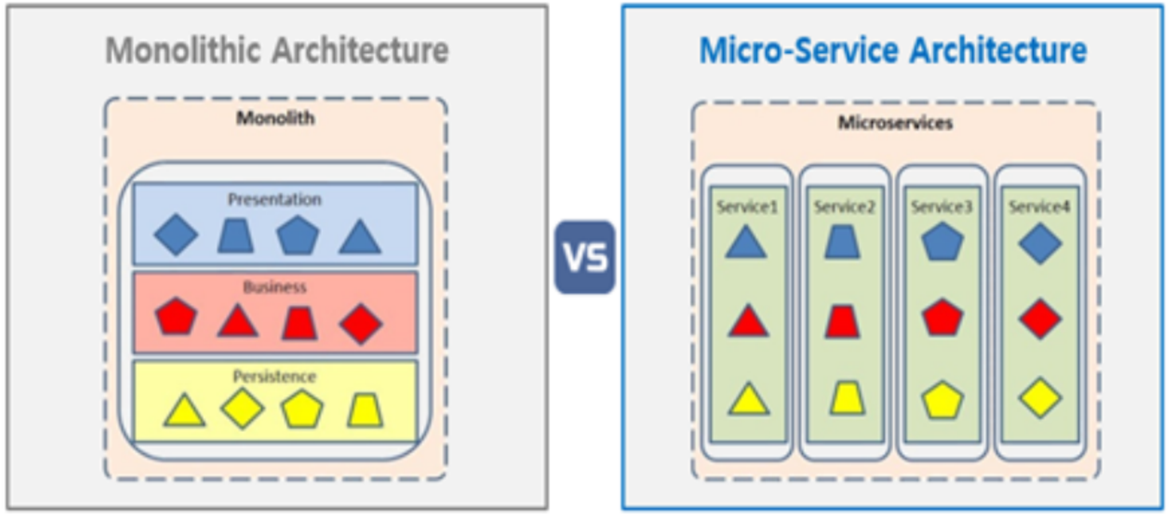
MicroService Architecture

여러 개의 작은 서비스로 구성되어 각 서비스가
독립적으로 개발되고 배포되는 구조

완전히 독립적으로 배포가 가능하고
다른 기술 스택이 사용 가능한 단일 사업 영역에 초점

01 MSA

등장 배경- Monolithic Architecture



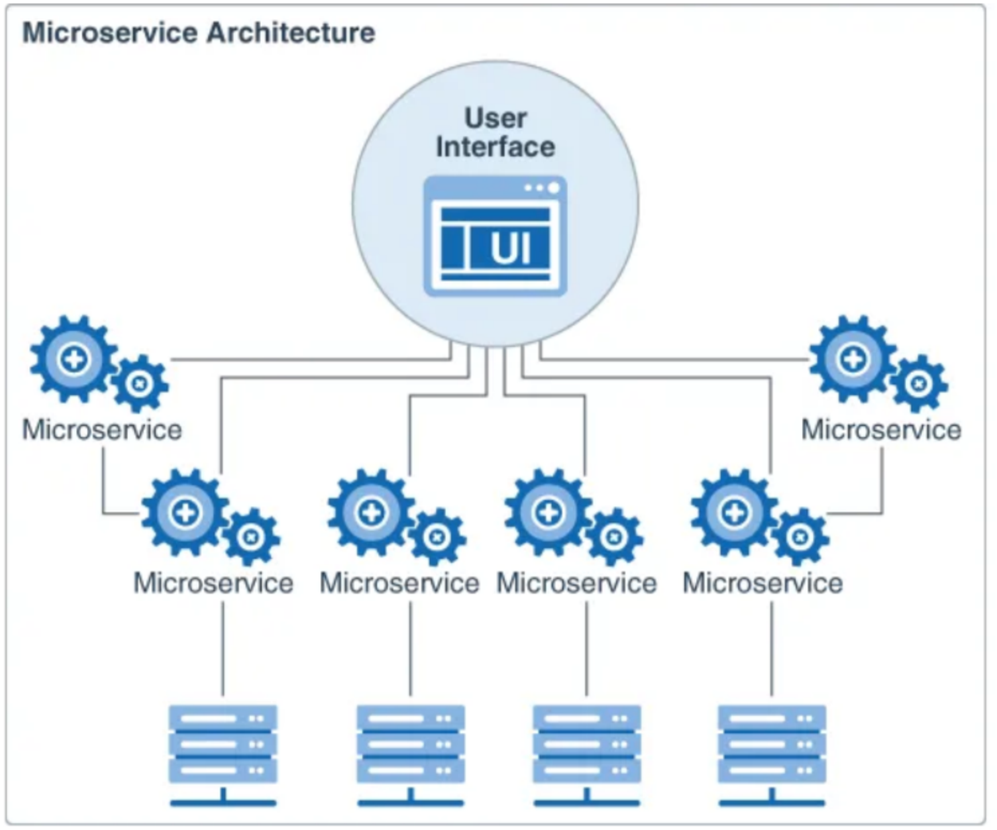
※ Micro-Service : 한 개 이상의 서비스(기능)로 Interface - Business Logic - Data 영역 모두 포함하여 독립적인 개발/수정/배포가 가능한 컴포넌트

Monolithic Architecture
소프트웨어의 모든 구성 요소가 한 프로젝트에 통합되어 있는 형태

한계점

- 부분 장애가 전체 서비스의 장애로 확대될 수 있다
- 부분적인 scale-out이 어렵다.
- 서비스의 변경이 어려고 수정시 장애의 영향 파악이 힘들다
- 배포 시간이 오래걸린다
- Framework와 언어에 종속적이다

01 MSA



특징

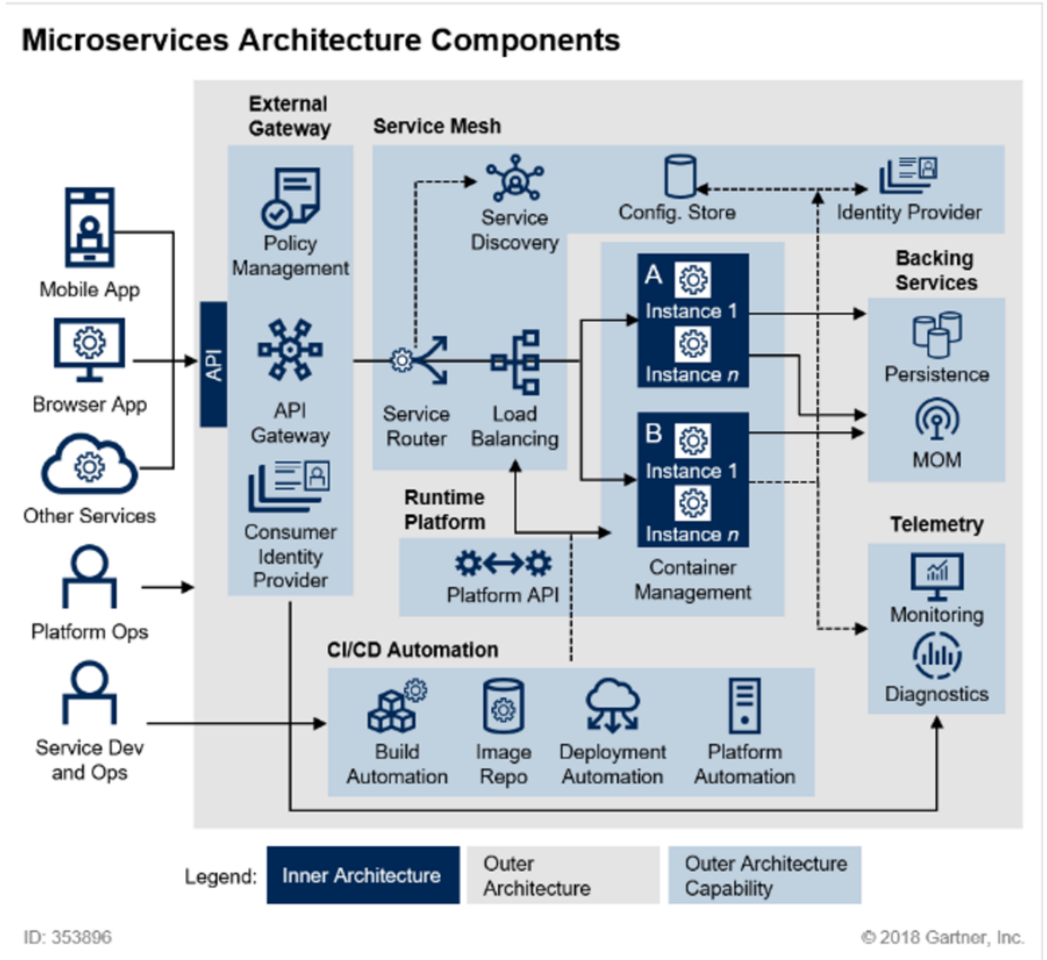
MSA는 API를 통해서만 상호 작용 할 수있어서 서비스의 end-point를 api 형태로 외부에 노출 시키고 실질적인 세부 사항은 모두 추상화한다.

MSA로 구성되어 있는 애플리케이션의 경우 전체 시스템이 분산 되어 있기 때문에 개발, 배포가 독립적으로 가능하고 확장성과 유지관리가 용이하다.

대규모 및 복잡한 프로젝트에 적합하다.

01 MSA

아키텍처의 구조



내부 아키텍처

고려사항
- 마이크로 서비스를 어떻게 정의할 것인가?
- DB 접근 구조를 어떻게 설계할 것인가?
- 마이크로 서비스 내 API를 어떻게 설계할 것인가?

내부 서비스를 어떻게 잘 쪼갤지에 대한 설계

외부 아키텍처

- 외부 게이트웨이
- Service Mesh
- Container Management
- Backing Service
- Telemetry
- CI/CD Automation

02 MSA와 Spring Cloud

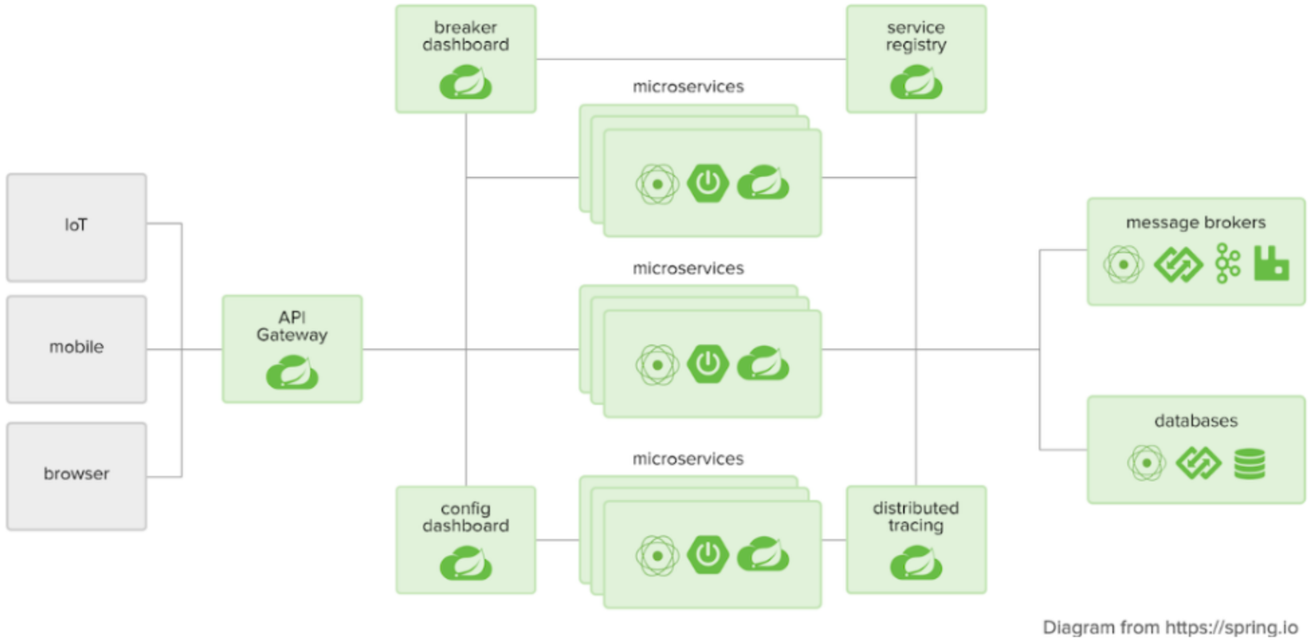


Diagram from https://spring.io

서비스 디스커버리, 로드 밸런싱, 분산 설정 관리, 분산 추적, API 게이트웨이 등 다양한 기능을 포함하고 통합과 확장이 가능하다.

Spring Cloud

분산 시스템을 구축하고 운영하기 위한 스프링 프레임 기반 라이브러리 모음

MSA를 구현할 때 주로 사용 된다.

02 MSA와 Spring Cloud

제공 기능

API 게이트웨이

: MSA에서 클라이언트와 백엔드 서비스 간의 통신을 중앙에서 관리하는 서버로,
클라이언트 애플리케이션이 서비스에 요청을 보낼 때 API 게이트웨이가
서비스 간 통신을 관리하고 모니터링을 한다.

02 MSA와 Spring Cloud

제공 기능

서비스 디스커버리

: 서비스들의 위치와 상태를 자동으로 찾고 관리하는 기능.

분산 설정

: MSA에서 필요한 설정을 중앙에서 관리하고 업데이트 할 수 있도록 하는 기능
서버에서 설정을 동적으로 가져와 사용하기 때문에 서비스가 실행 중일때 설정을 변경하고 즉시 적용 시킬 수 있다.

02 MSA와 Spring Cloud

제공 기능

로드 밸런싱

: 여러 서버에게 들어오는 트래픽을 균등하게 분배하여 부하를 분산시키는 기술
세션 유지, 스케일 아웃과 같은 기능을 통해 네트워크 트래픽을 관리

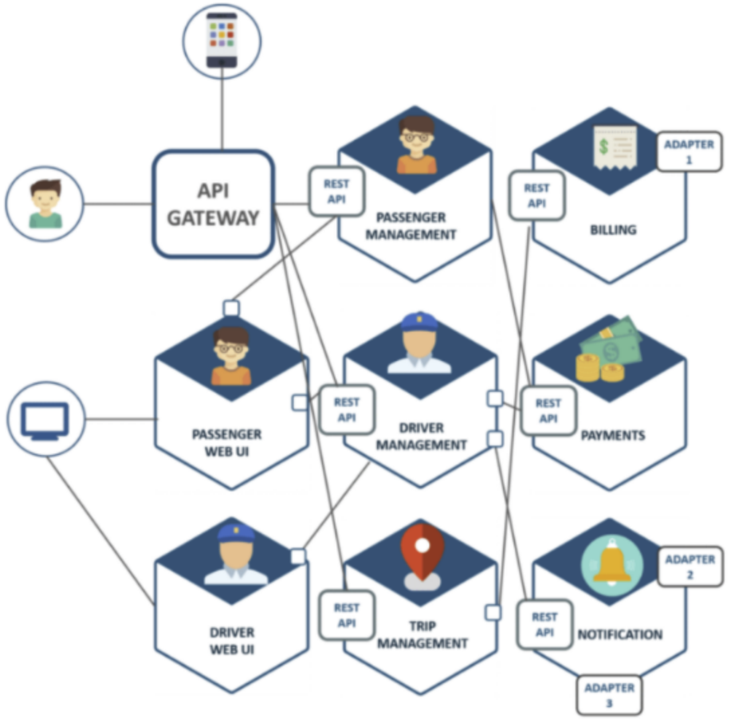
회로 차단기

: 서비스 간의 통신에서 발생 할 수 있는 장애에 대응하는 매커니즘
통신이 실패할 경우 일시적으로 연결을 차단해서 시스템에 영행을 최소화한다.

03 MSA 활용 사례

우버

미국의 종합 운송 기업. 택시, 물류/음식 배달 서비스 기능



기존 Monolithic Architecture에서 MSA로 이전함

도입후 변화된 점

운전자와 승객을 연결하는 API 게이트웨이가 등장함

API 게이트웨이로부터 각 가능들의 모든 내부 요청이 연결되어 있고
각각의 서비스는 독립적으로 분리되어 배포 가능하고 독립적으로 기능을 수행하게 되었음

요구 청구 서비스에 변화가 있을 때 요금 청구 서비스만 수정해서 배포하고 다른 서비스는 배포 할 필요가 없어지기 때문에 각각의 기능들을 독립적으로 적용 및 확장 가능

03 MSA 활용 사례

넷플릭스

미국의 멀티미디어 엔터테인먼트 OTT 기업

기존 Legacy 시스템은 모두 MSA로 전환해서 운영/개발 상의 효율성을 극대화 함

MSA를 선택 한 이유

과거 2007년도에 심각한 db 손상으로 인해 3일간 서비스 장애를 겪고난 후
신뢰성이 높고 수평 확장이 가능한 클라우드 시스템으로 이전할 필요성을 느낌

고가용성, 유연한 스케일링, 빠르고 쉬운 배포를 위해 MSA를 선택함

Netflix OSS

넷플릭스 측에서 MSA 전환 기술을 오픈소스로 공개

04 출처

https://wooaoe.tistory.com/57
https://mozzi-devlog.tistory.com/34
https://velog.io/@black_han26/Spring-Cloud-API-Gateway-%EA%B0%9C%EB%85%90-%EB%B0%8F-%EA%B8%B0%EB%8A%A5
https://steady-coding.tistory.com/595
https://m.blog.naver.com/ghdalswl77/222407195800