MSA와 Service mesh

(주) 맨텍 이진현
jhlee@mantech.co.kr

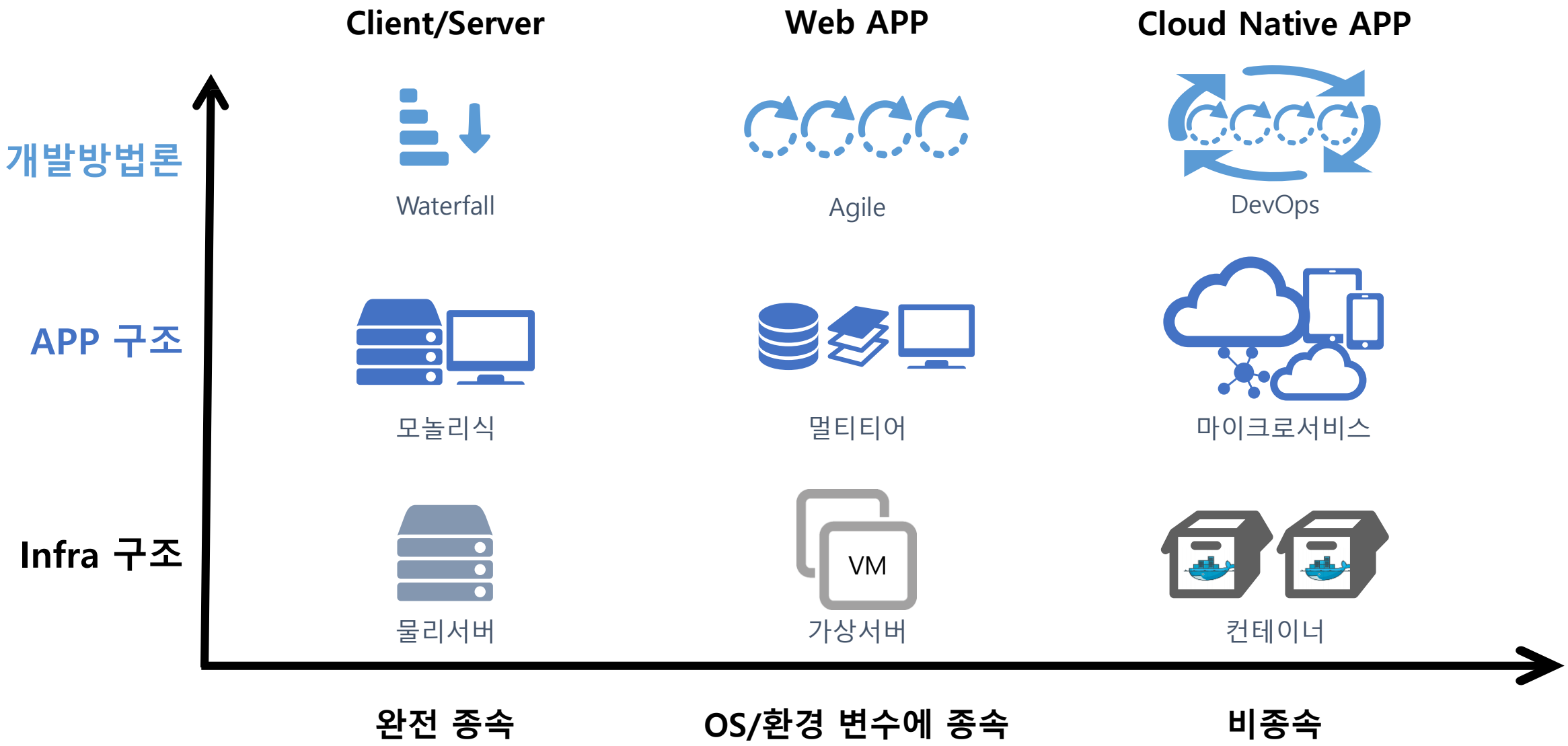
manTech
better solution

CONTENTS

1

“ MSA 개요

앱의 아키텍처는 종속성을 벗어나기 위한 방향으로 진화



MSA와 CNA는 왜 대두되게 되었는가?

속도

- ✓ 비지니스의 빠른 변화로 인한 애플리케이션의 잦은 배포 / 신속 배포 / 신속 롤백 필요

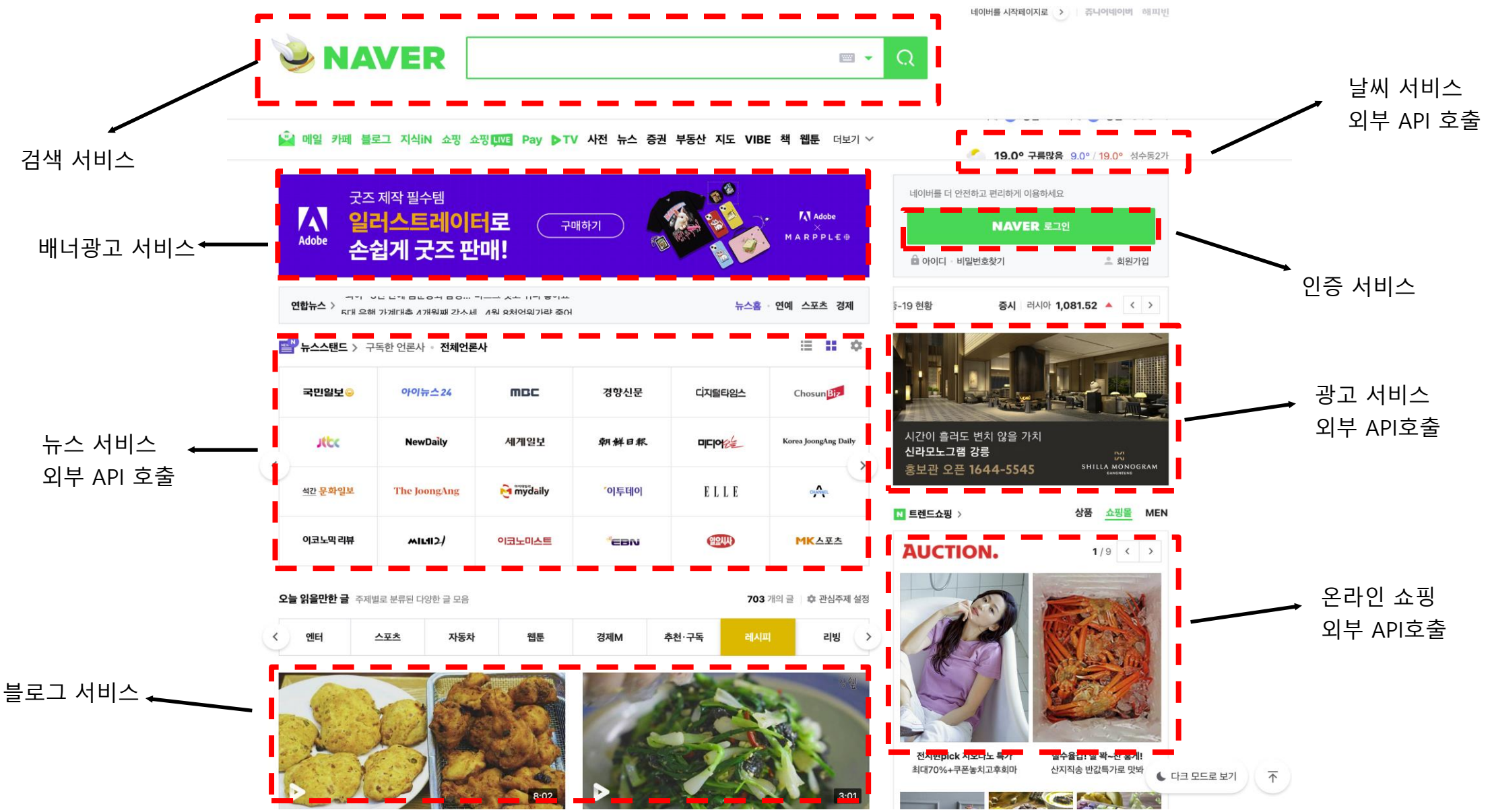
무중단

- ✓ 투명한 모니터링 / 장애 영향의 최소화 / 자동 복구 체계 필요
- ✓ 애플리케이션 배포, 롤백 시 중단 없는 서비스 요구

확장성

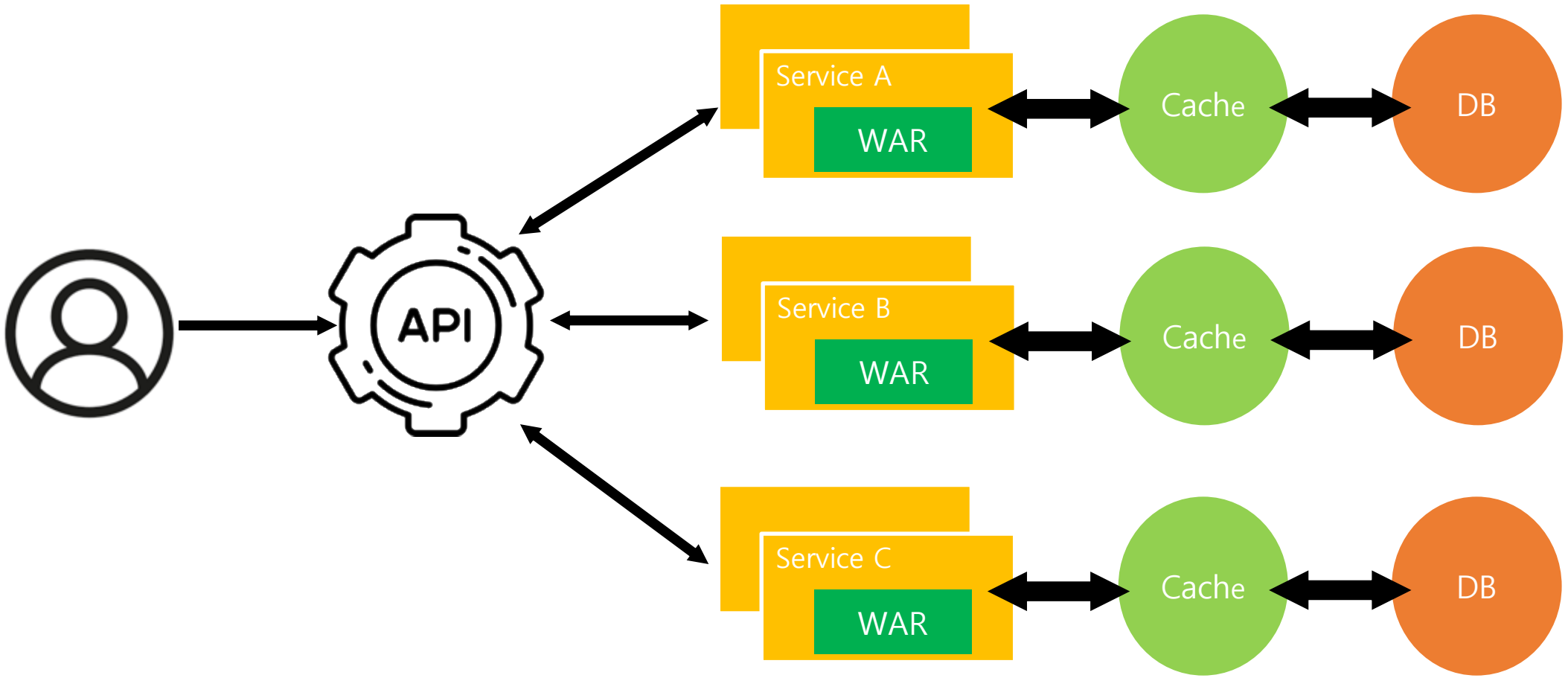
- ✓ 요구사항에 따른 수평적 확장
- ✓ 멀티 클라우드에 대한 투명한 확장 요구

MSA구조는 쉽게 접할 수 있다.

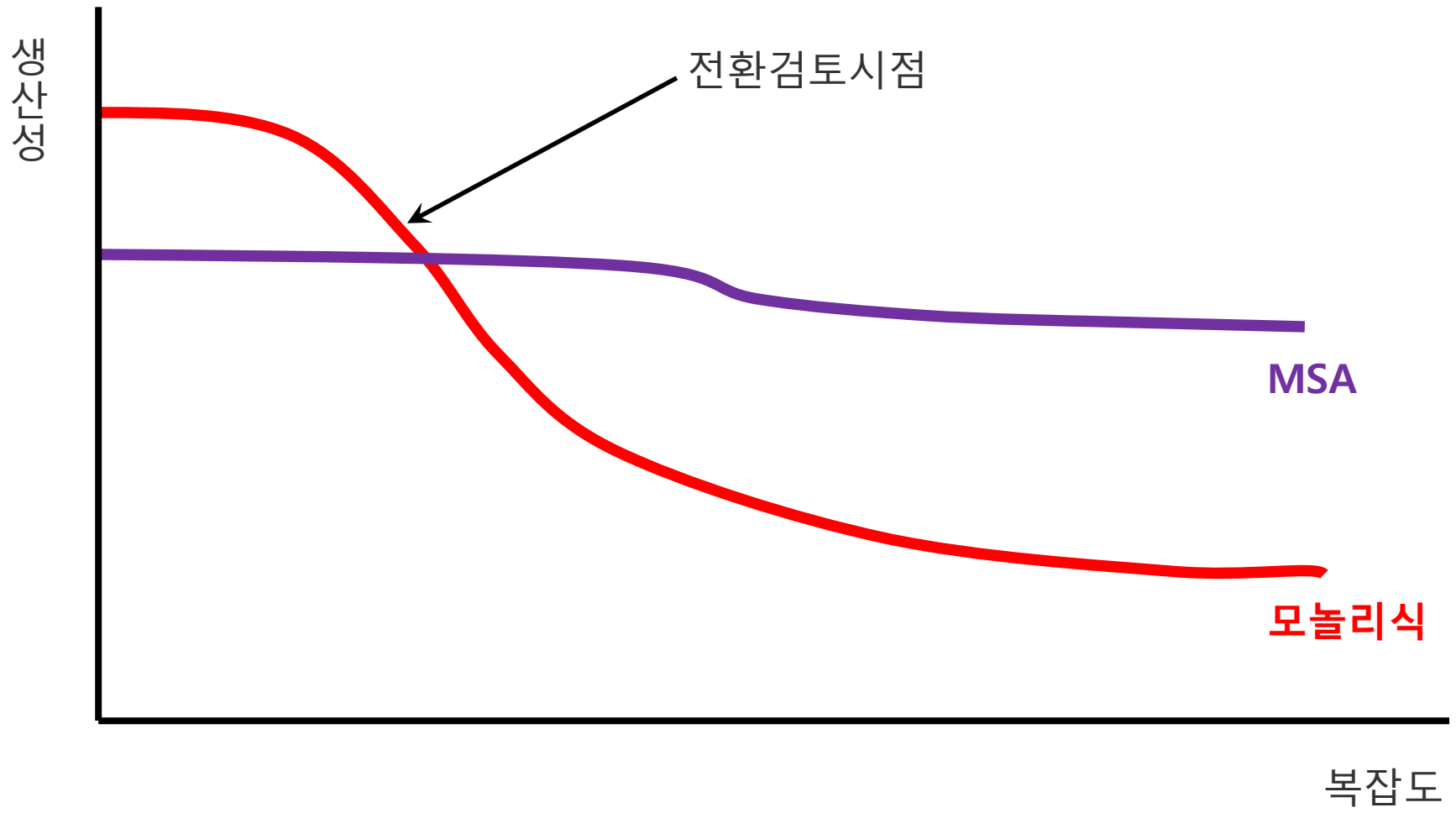


마이크로 서비스 아키텍처 란?

✓ 단일 서비스로 구성된 애플리케이션을 여러 서비스 단위로 작게 나누고, 서비스들끼리 서로 통신하는 형태의 아키텍처

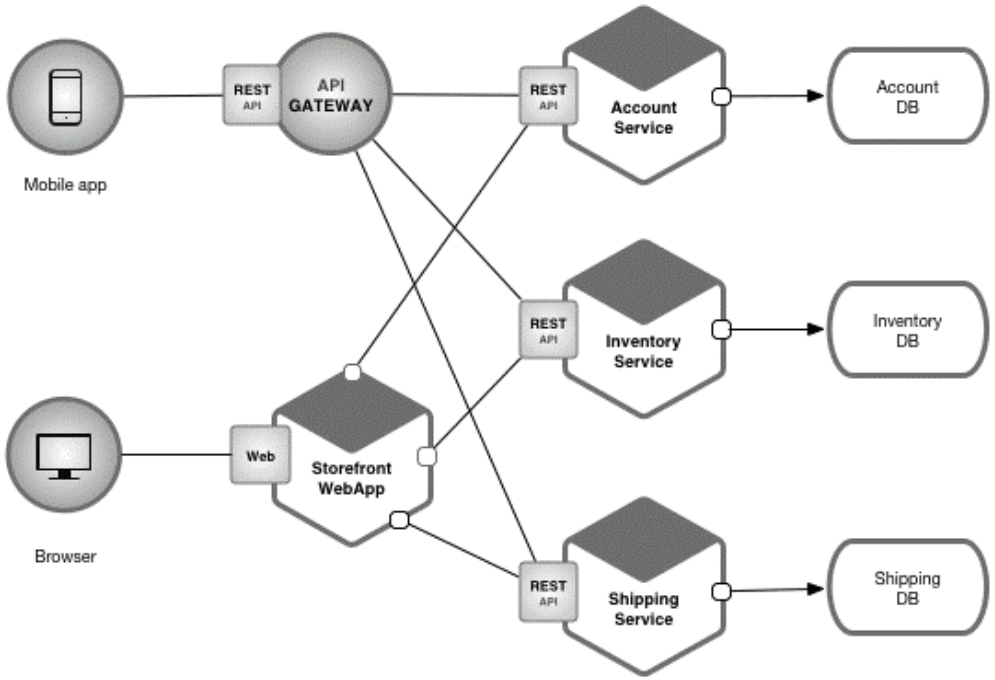


MSA로의 전환 시점?



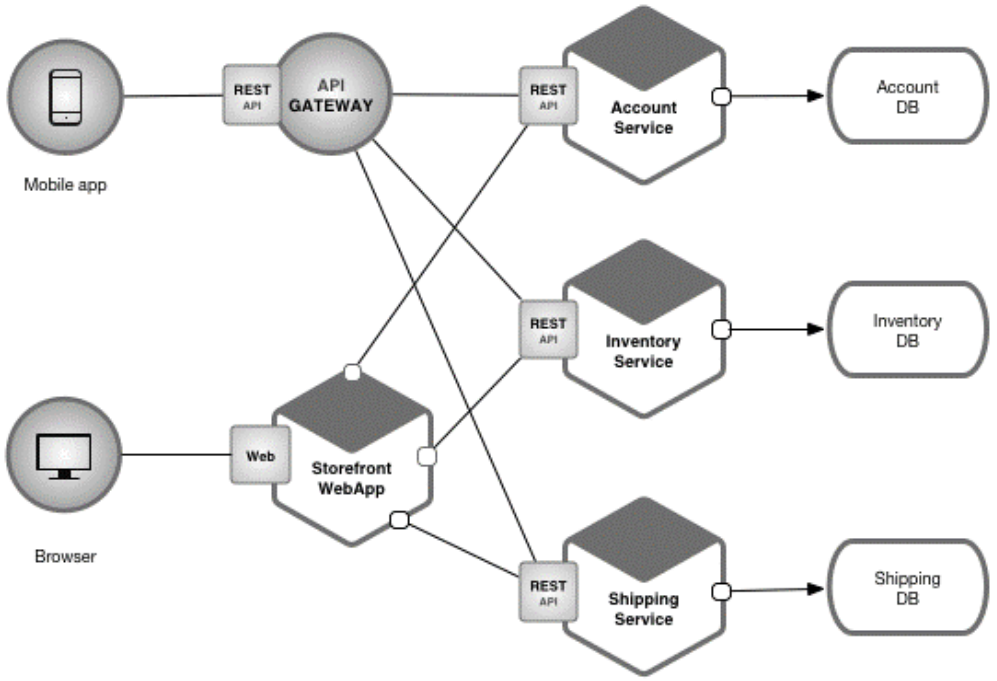
MSA의 특징

- ✓ 각 서비스 간 **느슨한** 결합 구조
- ✓ 각 서비스는 **독립적**으로 개발/배포 가능
- ✓ 서비스 간 **API**를 통한 통신 필요
- ✓ 독립적으로 **각각 데이터** 저장



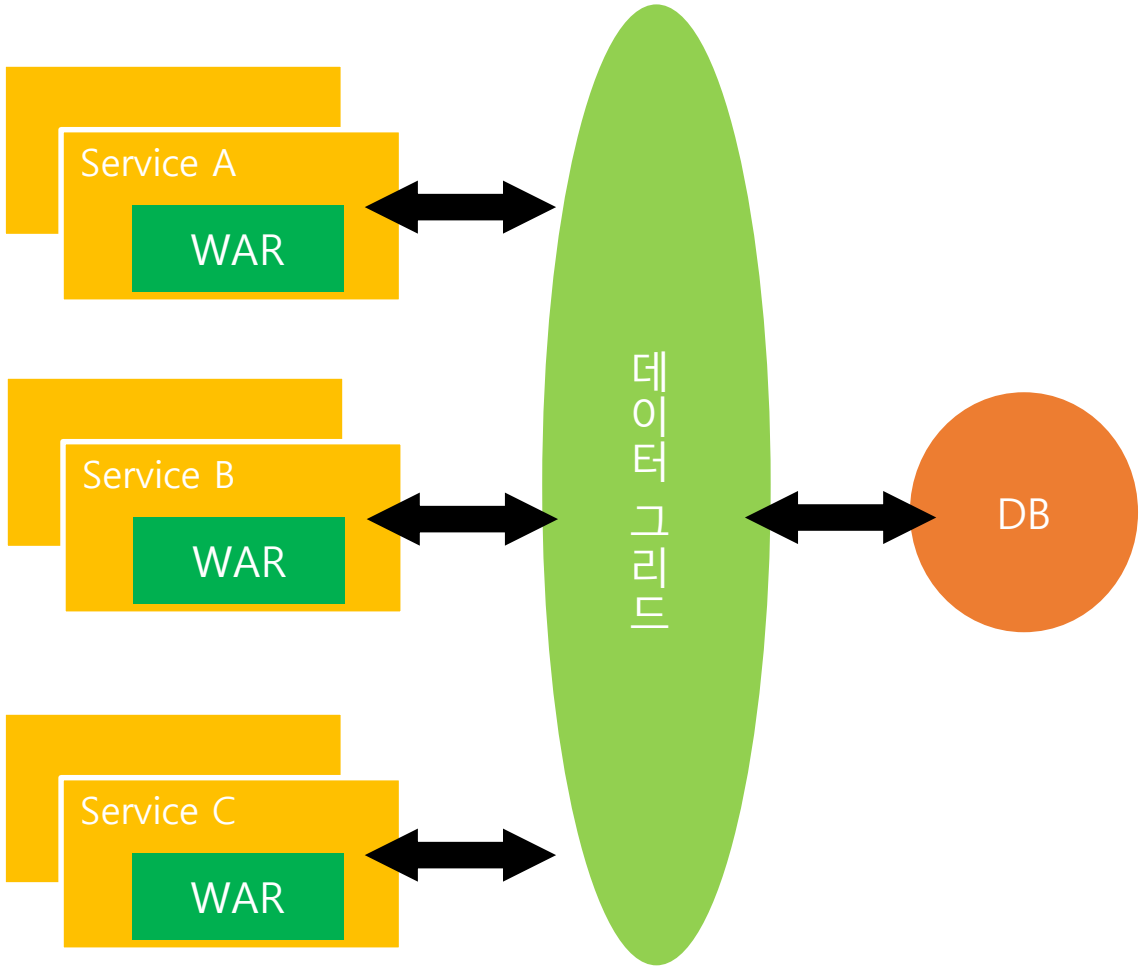
MSA의 장점과 단점

장점	✓ 각 서비스 별로 독립적 개발 가능 ✓ 각 서비스 별 독립적인 배포와 확장 가능 ✓ 서비스 별 특성에 맞는 언어, 인프라 사용 가능 ✓ 각 서비스의 장애가 전체 장애를 유발시키지 않음 ✓ 유지보수 매우 용이
단점	✓ 나누어진 서비스 간 통신 방법이 필요 ✓ 서비스 간 호출이 모놀리스보다 복잡 ✓ 모놀리스보다 성능 저하 ✓ 데이터 중복 발생하며, 정합성 보장이 쉽지 않음 ✓ 서비스 단위 테스트와 통합 테스트 정책 수립 필요



기존 환경을 MSA로 전환 시 도전과제와 살펴봐야할 사항들

- ✓ 기존 통합된 DB를 어떻게 나눌 것인가?
- ✓ 단일의 앱을 다수의 서비스 인스턴스로 나눌 시 세션의 공유는?
- ✓ 각 API 호출에 대한 Single entry point 처리는 어떻게?
- ✓ 여러 서비스로 나눌 만큼 큰 덩어리인가?
- ✓ 나눈 서비스별 조직 체계는?

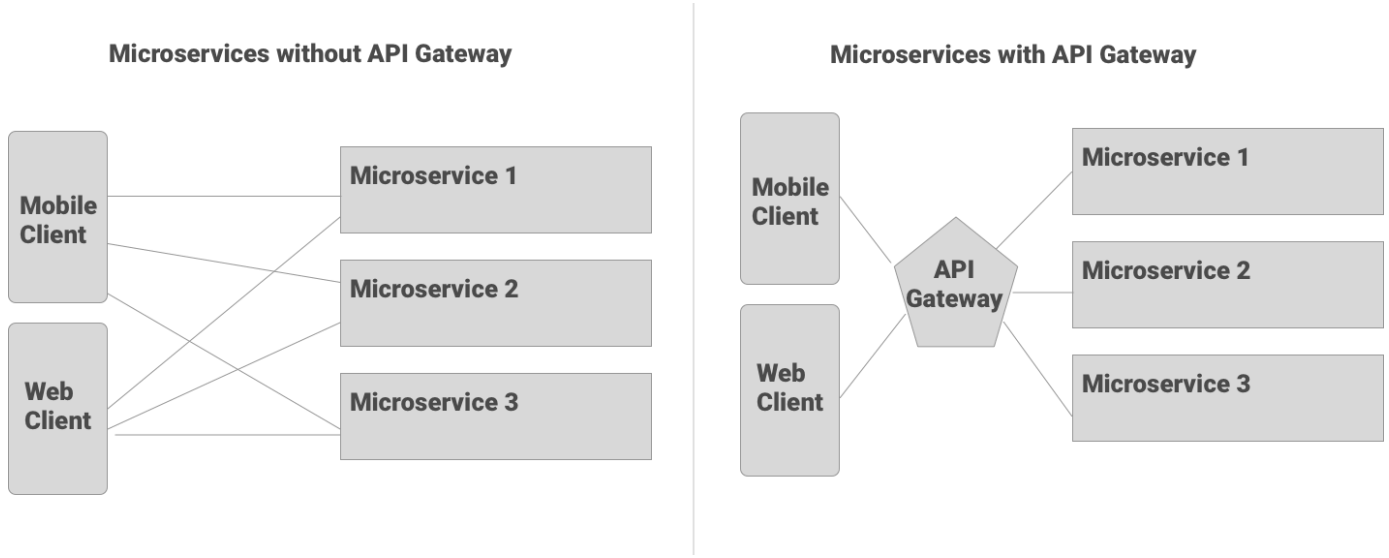


2

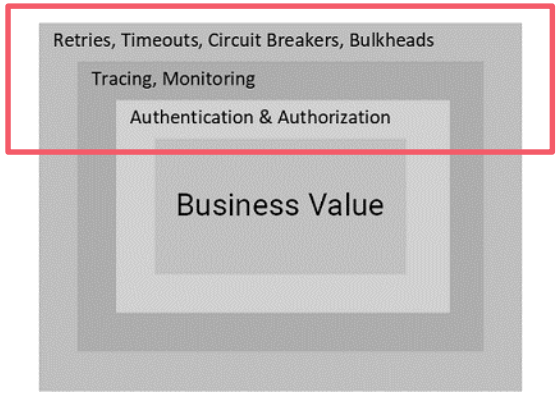
“서비스 메쉬 소개

API Gateway

✓ MSA간 서비스 호출과 라우팅을 처리하기 위한 단일 Entry point



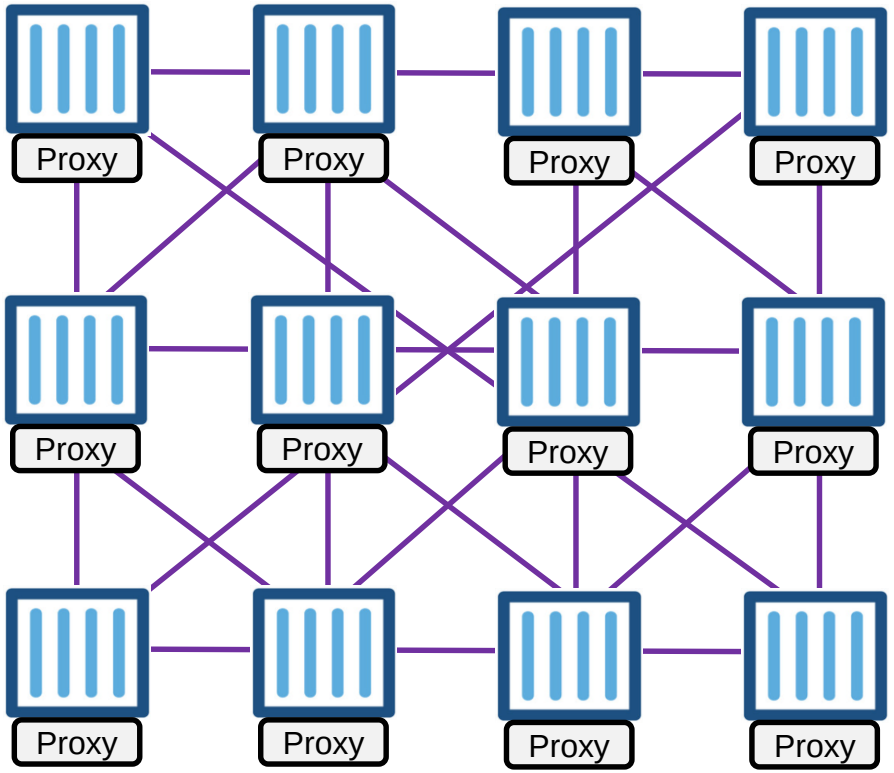
API Gateway에서 처리하고
개발자는 Business Value에 집중



인증 및 인가	entry point 단일화	라우팅	프로토콜 변환
- 요청에 대한 인증 및 인가 일괄 처리 - 개발자는 비즈니스 구현에 집중	- 모든 API에 대한 요청 단일화 - 모니터링과 같은 공통 기능 일괄 수행	- 조건에 따라 호출에 대한 서비스 분배 - 로드밸런싱	- 프록시 역할 수행 - 어플리케이션 변경 없이 프로토콜 변환

Service mesh

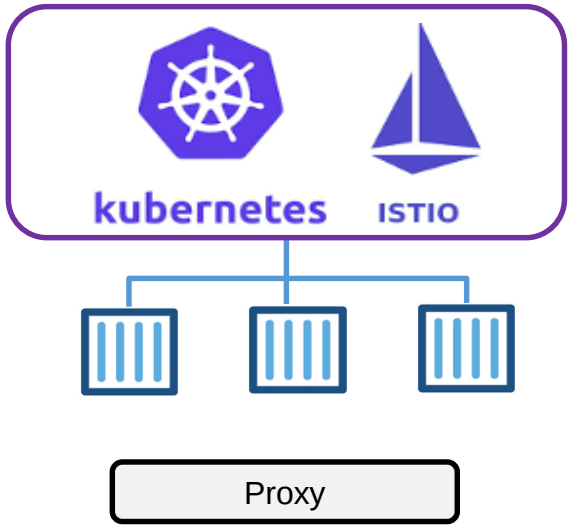
✓ K8S상 MSA로 쪼개진 POD들간 API호출을 위한 네트워크

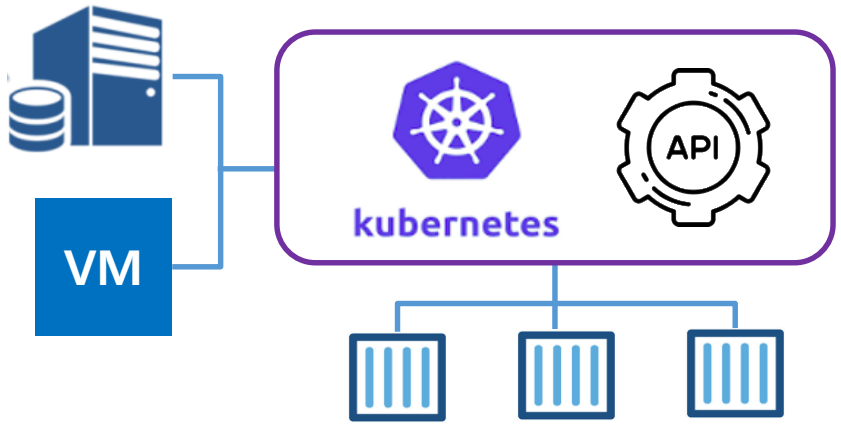


Service Mesh 기능

- Service Discovery
- Load Balancing
- Dynamic Request Routing
- Circuit Breaking
- Retry and Timeout
- TLS
- Distributed Tracing
- Metrics Collecting

API GW vs Service mesh

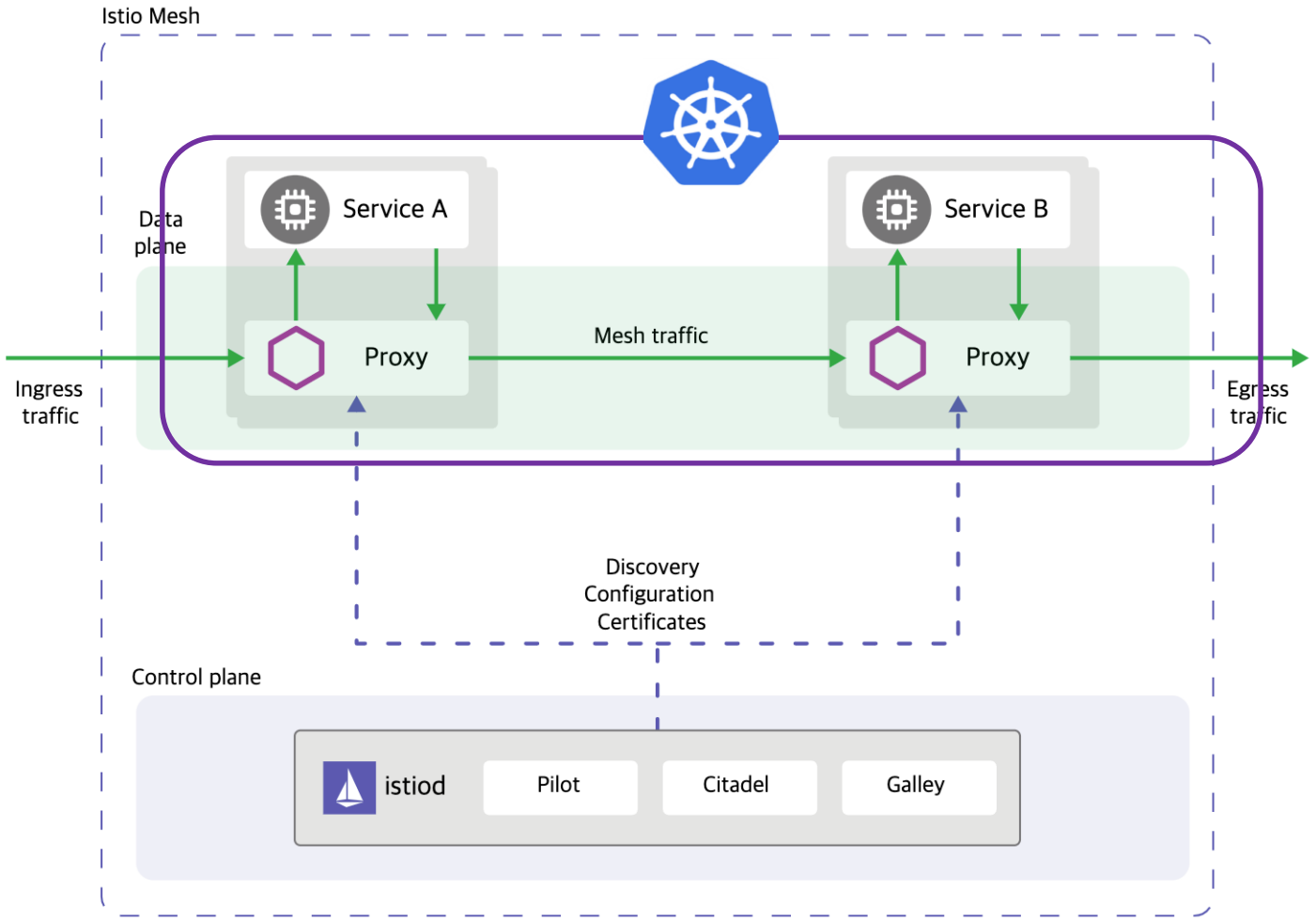




Service Mesh	API GW
내부 자원들 간 통신과 호출에 초점	내부와 외부 자원들 간 통신과 호출에 초점
네트워크 내 서비스 관리와 제어	서비스를 외부 인터넷에 노출
주로 east-west간	주로 north-south간
MSA들의 경계 내부에서 역할을 수행	MSA의 외부 경계에서 역할을 수행
분산형 아키텍처(No SPOF)	중앙 집중형 아키텍처(SPOF)

Istio의 Service mesh구성 아키텍처

- ✓ **Control Plain** : 트래픽을 제어하는 정책 및 구성에 따라 Proxy에게 설정 값을 전달하고 관리
- ✓ **Data Plain** : Proxy를 통해 MSA간 모든 네트워크 통신을 조정하고 제어

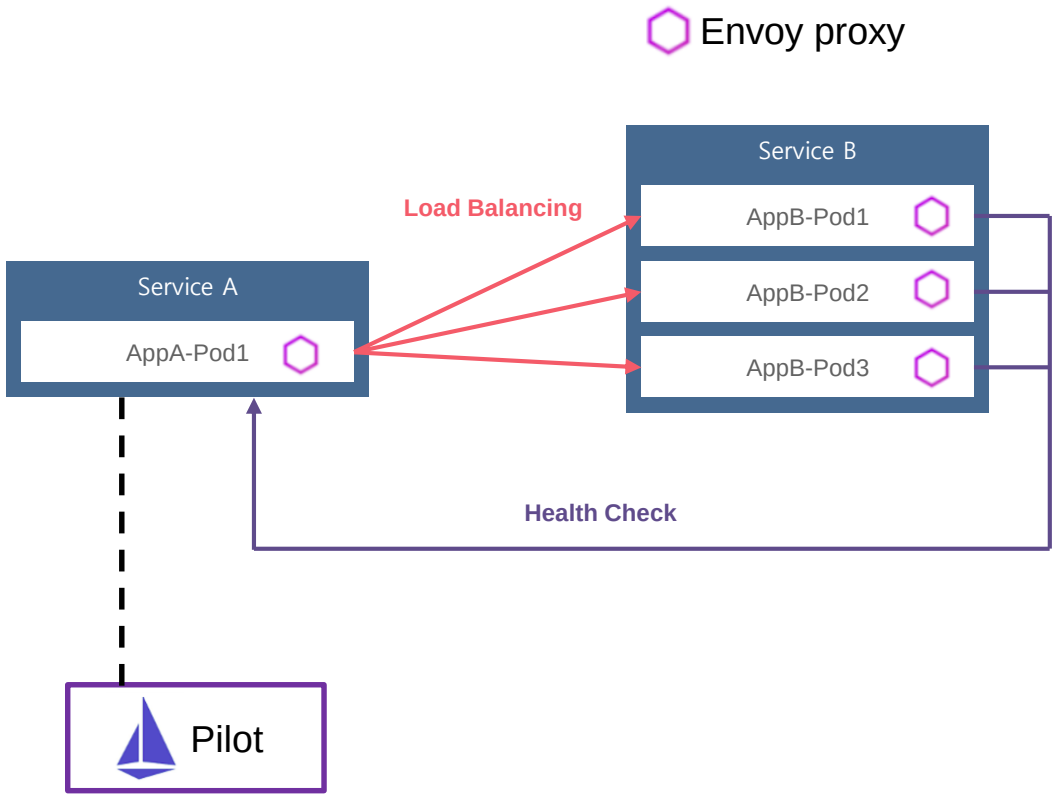


Control과 Data Plain의 구성요소

구성 요소	내용
Pilot	✓ Proxy에 대한 설정 관리 ✓ Traffic 관리 기능
Citadel	✓ 보안에 관련된 기능을 담당 ✓ TLS 암호화나 사용자 인증에 필요한 인증서 관리
Galley	✓ Istio configuration 체크
Envoy Proxy	✓ Pod에 sidecar 형태로 구성 ✓ 경량화된 L7 전용 Proxy ✓ Service discovery, Load balancing, TLS termination, Circuit breaker, Health checks, Retry, Timeout 등 다양한 기능을 수행

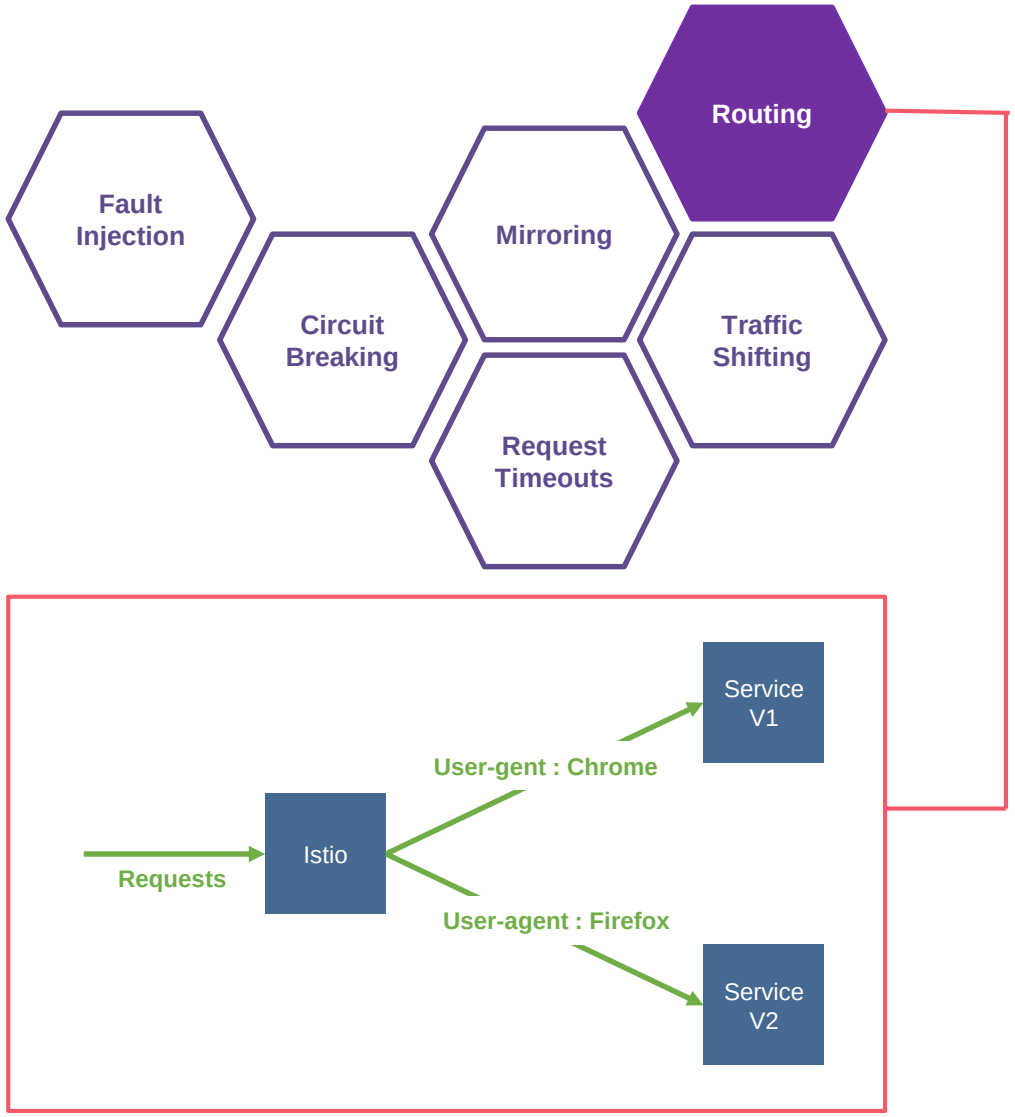
Health check과 Service discovery

- ✓ Istio의 Pilot은 대상 서비스가 여러 개의 인스턴스로 구성되어 있을 경우 로드밸런싱 수행
- ✓ 상태 체크를 통해 정상적인 POD로만 트래픽 라우팅



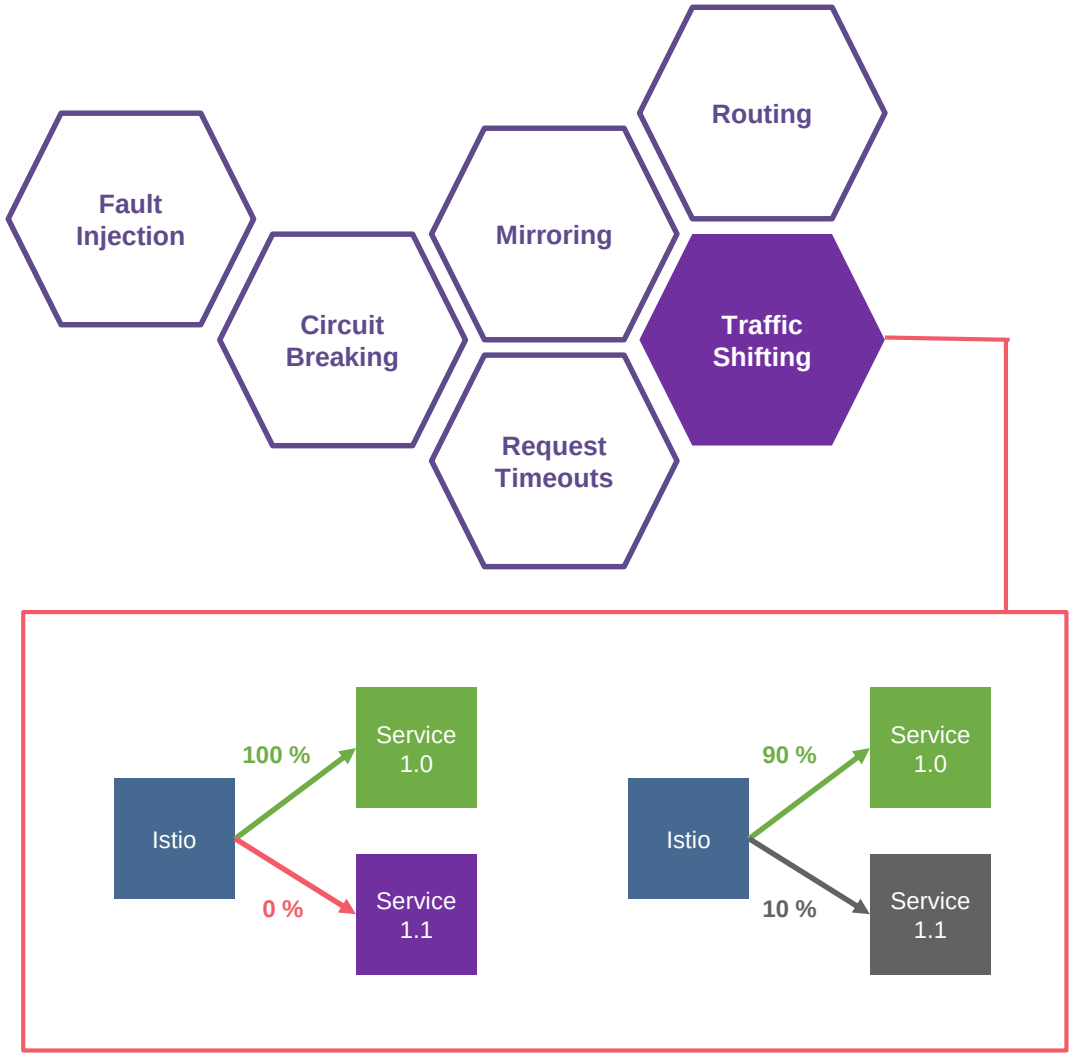
컨텐츠 기반 서비스 라우팅

- ✓ 단순히 커넥션 기반으로 트래픽을 라우팅하지 않고 요청한 컨텐츠 내용 기반으로 서비스 라우팅
- ✓ 사용자의 디바이스 종류에 따른 서비스 라우팅 처리가 용이해 짐



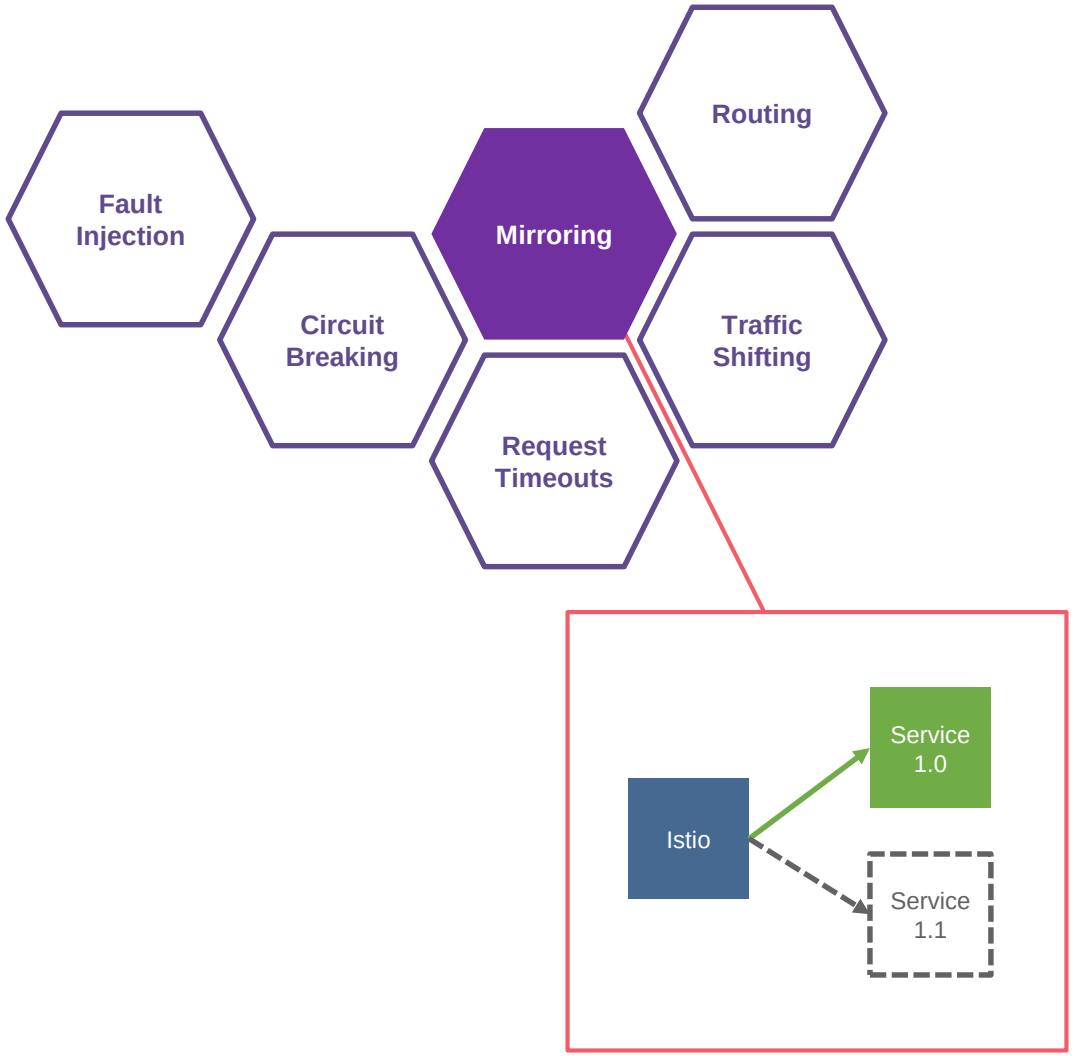
트래픽 분할

- ✓ 새로운 버전의 서비스 배포 시 기존 버전과 신규 버전에 대한 트래픽 양을 조절하여 테스트
- ✓ K8S의 롤링업데이트와 다름



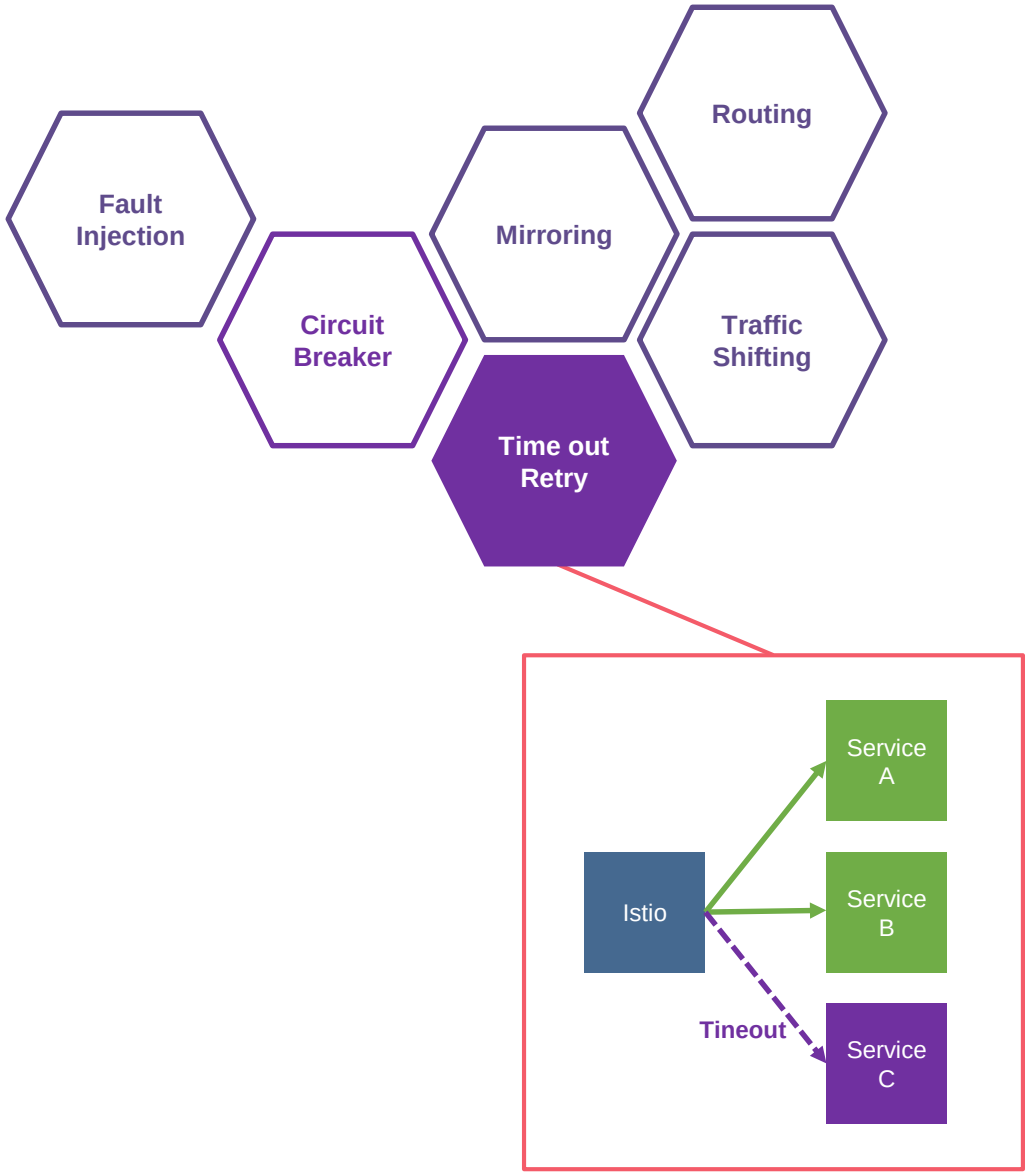
트래픽 미러링

- ✓ 서비스 요청을 신규 업데이트한 버전의 서비스쪽으로도 동일하게 보내어 실 사용자에 대한 Live test를 할 수 있음. 미러링된 서비스는 외부 응답을 하지는 않음



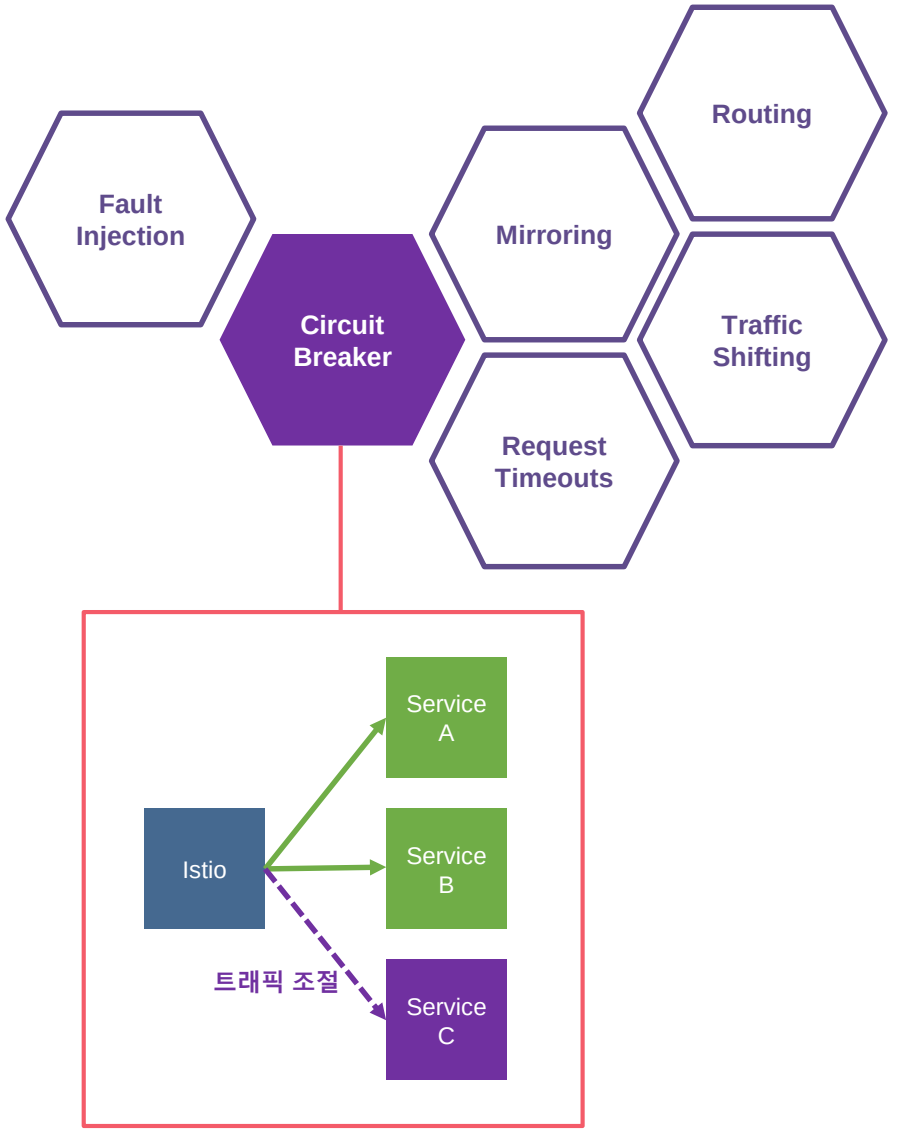
Time out

- ✓ 일정시간 응답이 오지 않으면 에러 처리
- ✓ 각 서비스 별과 Istio에서 설정할 수 있으며, 서비스 별 설정이 없을 경우 Istio의 값을 따름



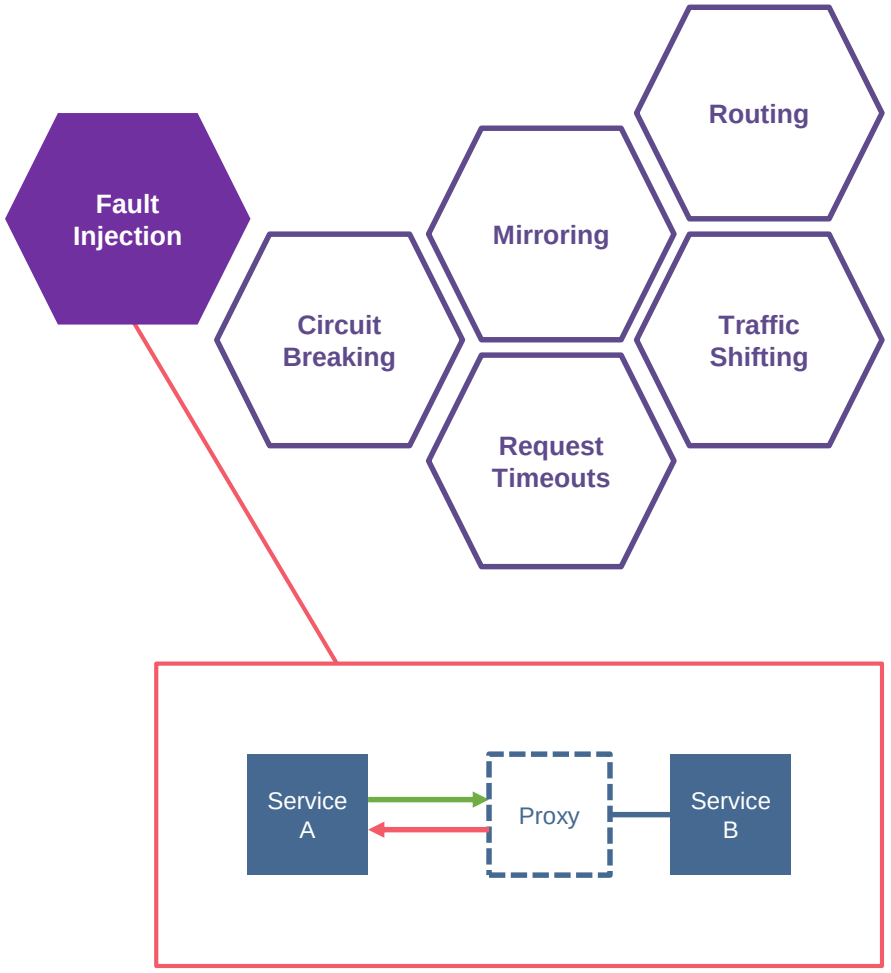
Circuit Breaker

- ✓ 특정 서비스쪽으로 과도한 트래픽이 몰릴 경우 임계치 이상의 요청을 거부하여 전체적인 서비스 장애를 사전에 방지
- ✓ 요청 거부 시 에러 메세지로 응답



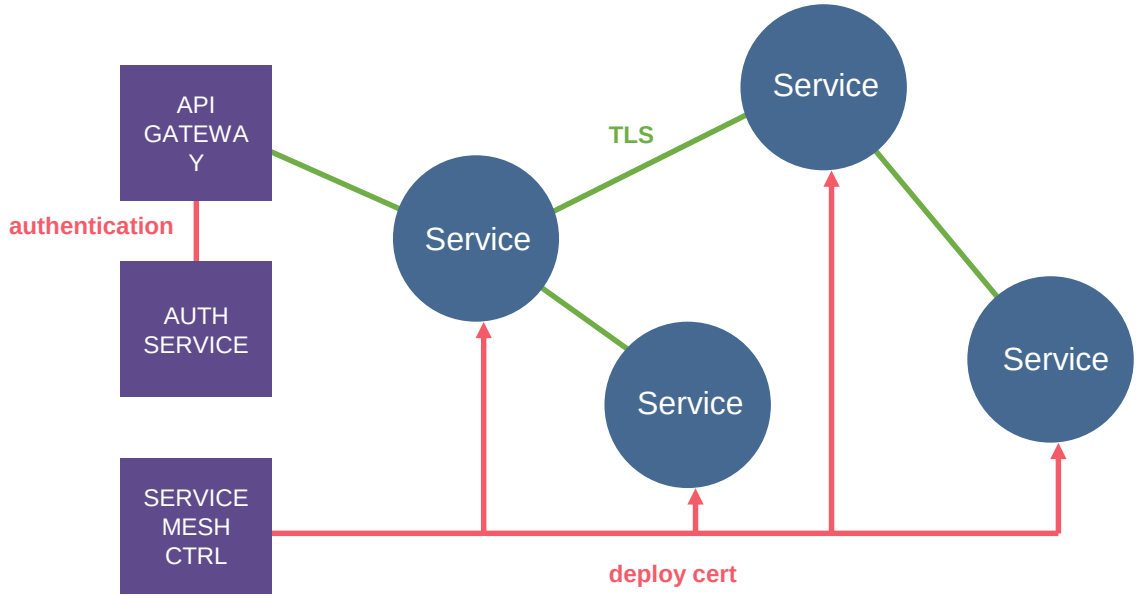
장애 유발

- ✓ 서비스 호출 시 강제로 장애를 유발시켜 장애 시 처리 로직이 정상으로 작동하는지 여부를 확인할 수 있음



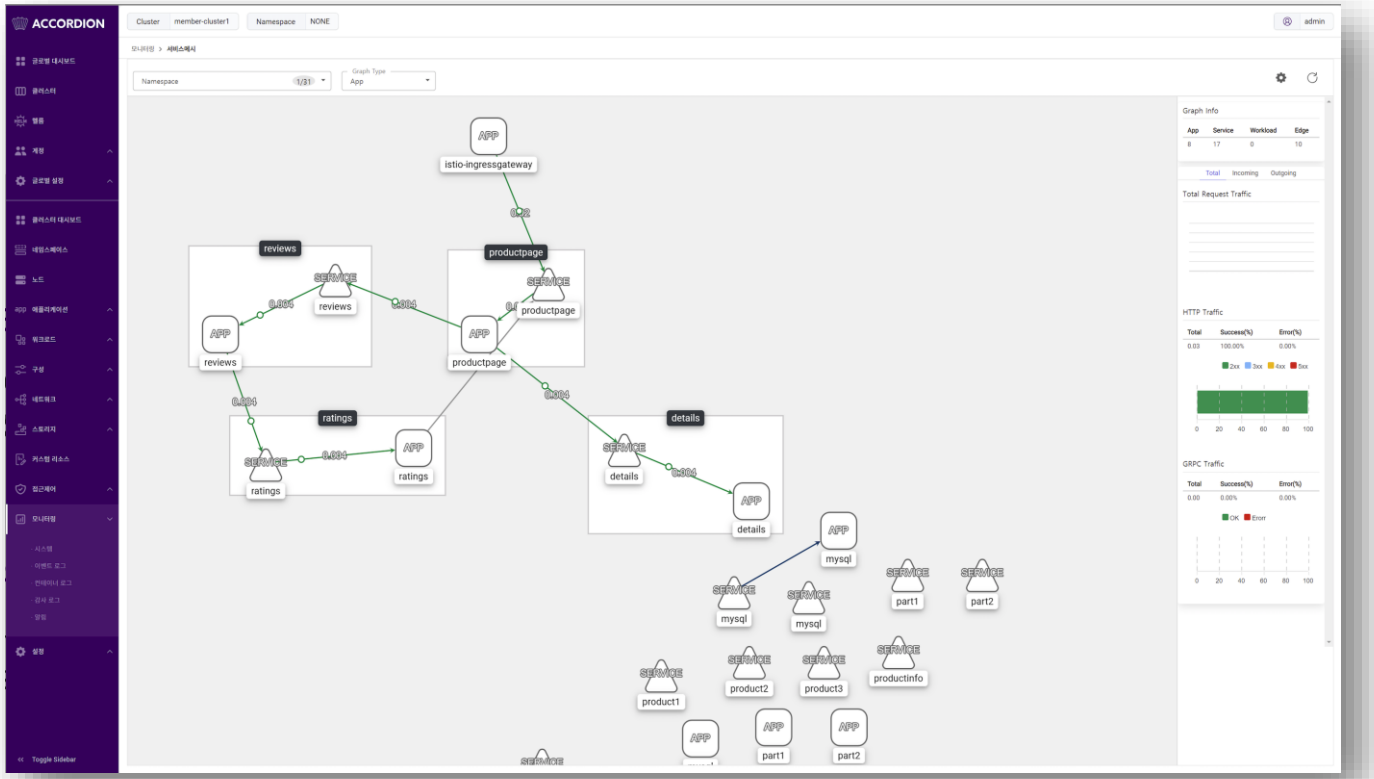
인증 및 인가

- ✓ 인증서를 이용하여 양방향 TLS인증을 이용하여 서비스와 서비스 간 인증 수행
- ✓ JWT토큰을 이용하여 서비스에 접근할 수 있는 클라이언트를 인증



모니터링

- ✓ MSA간 상관관계 가시화
- ✓ 서비스 간 네트워크 트래픽
- ✓ 권한 별 모니터링



Thank You