Learning Kubernetes

당통 소개

경력

변현창

연락처 : dangtong@gmail.com

2002 : BookZon
백앤드 개발

2010 : 삼성SDS (DBA)

2017 : Oracle Korea
미들웨어 솔루션 컨설턴트

2018 : 발란
CTO / Lead Software Developer

2018 : SAS Korea
AI/ML 분석 플랫폼 엔지니어

2020 : TIBCO Korea
Data Science and Analytics 엔지니어

2021 : 발란
VP of Engineering(겸 데이터플랫폼 실장)

2022 : Red Hat Application Architect

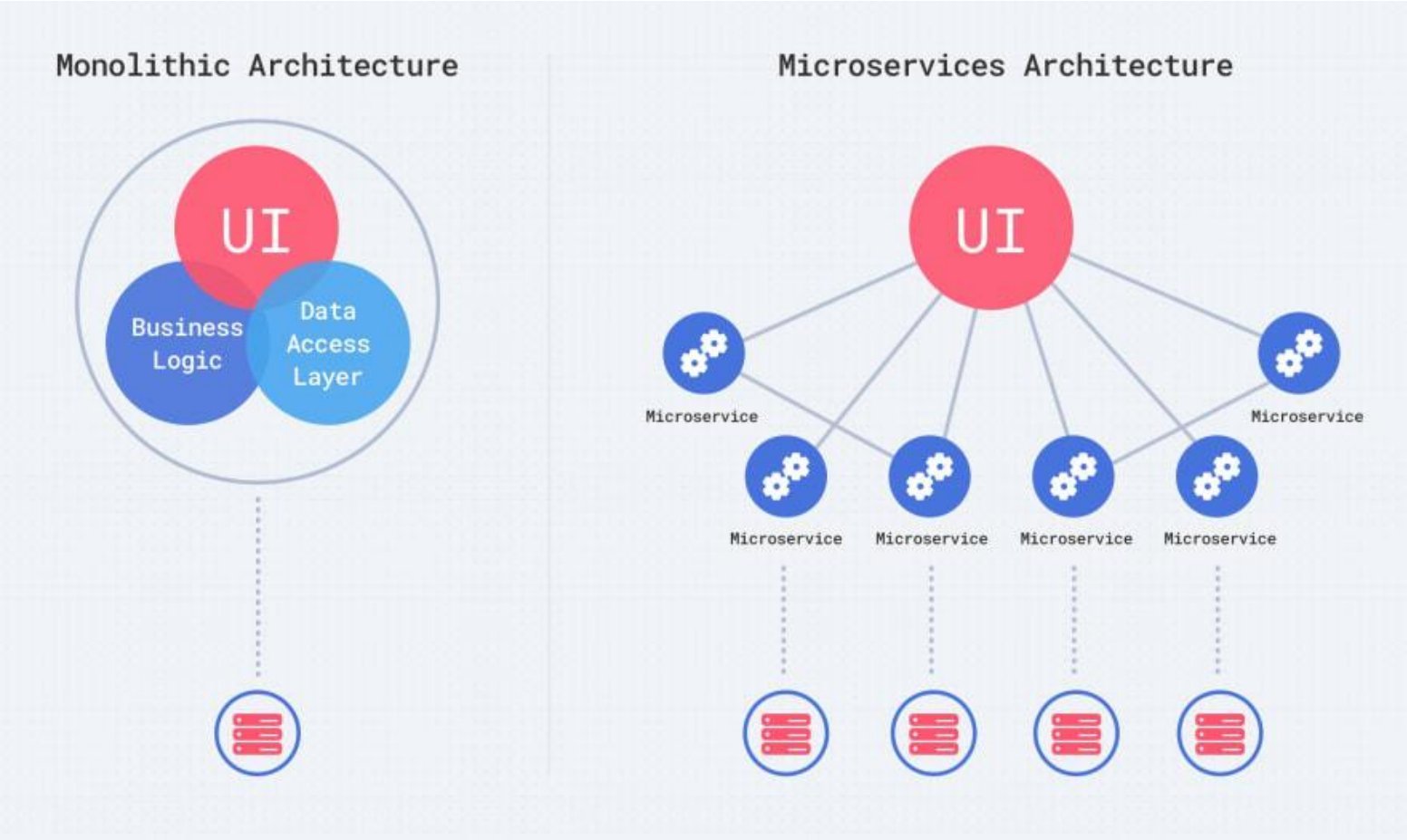
강의

- Python 으로 배우는 머신러닝/딥러닝 기초 (5주)
- React와 Node.js 로 배우는 마이크로 서비스 아키텍쳐 (5주)
- 도커 와 쿠버네티스 (5주)
- Laravel 프레임워크로 배우는 Modern PHP (5주)

01. 마이크로 서비스 아키텍쳐

마이크로 서비스 아키텍처(MSA)란?

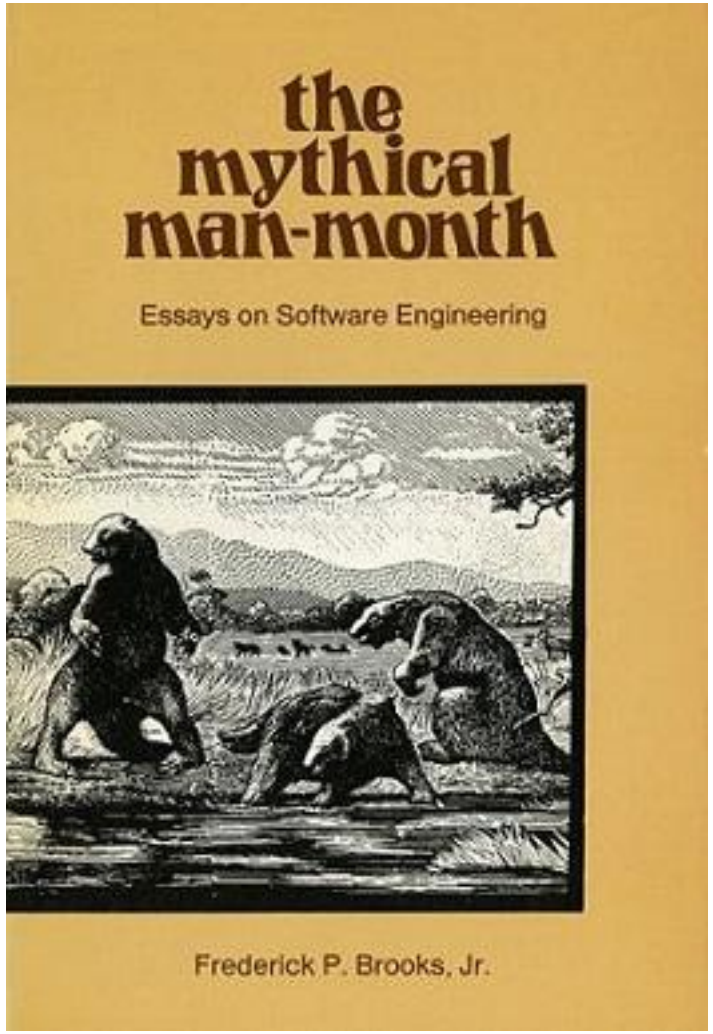
Loosely coupled and High-cohesion service oriented architecture with bounded context



SOA VS MSA

Component	SOA	Micro-Service
서비스간 통신	SOAP, WS* (Heavy Protocol)	REST, gRPC (lightweight Protocol)
데이터	전역 모델, 공유 DB	개별 데이터 모델 및 DB
주요사례	대규모 모놀리틱 애플리케이션	작은 서비스

브룩스의 법칙 (Brooks's Law)



브룩스의 법칙

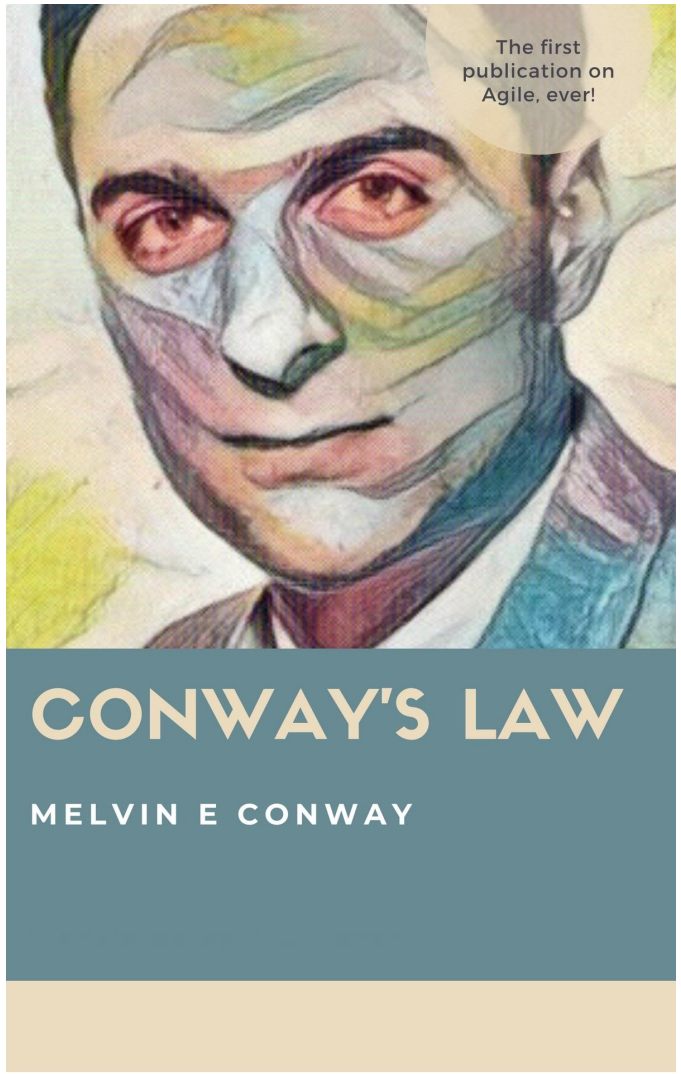
소프트웨어 개발 프로젝트에 인력을 더 투입하면, 개발은 더 늦어진다.

adding human resources to a late software project makes it later

맨먼스(Man-Month) 단위를 사용해 일정을
추정하는 것은 잘못된 사고 방식이다.

N명인 팀의 소통 오버헤드는 $O(n^2)$ 으로 증가한다

콘웨이의 법칙 (Conway's Law)



콘웨이의 법칙

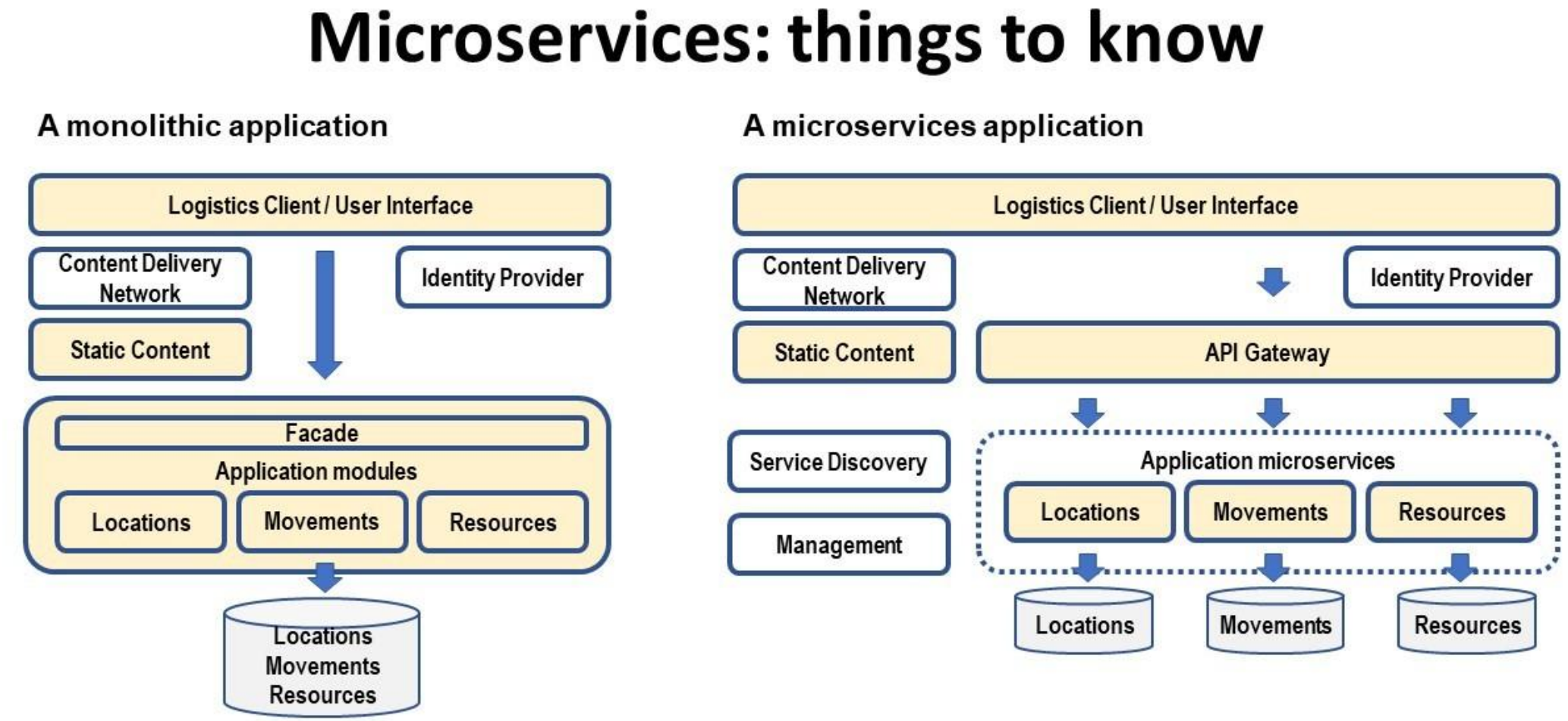
어플리케이션 아키텍처는 그것을 개발하는 조직 구조를 그대로 반영한다.

organizations, who design systems, are constrained to produce designs which are copies of the communication structures of these organizations

역 콘웨이의 전략

조직의 구조가 마이크로 서비스에 적용 되도록 조직을 설계

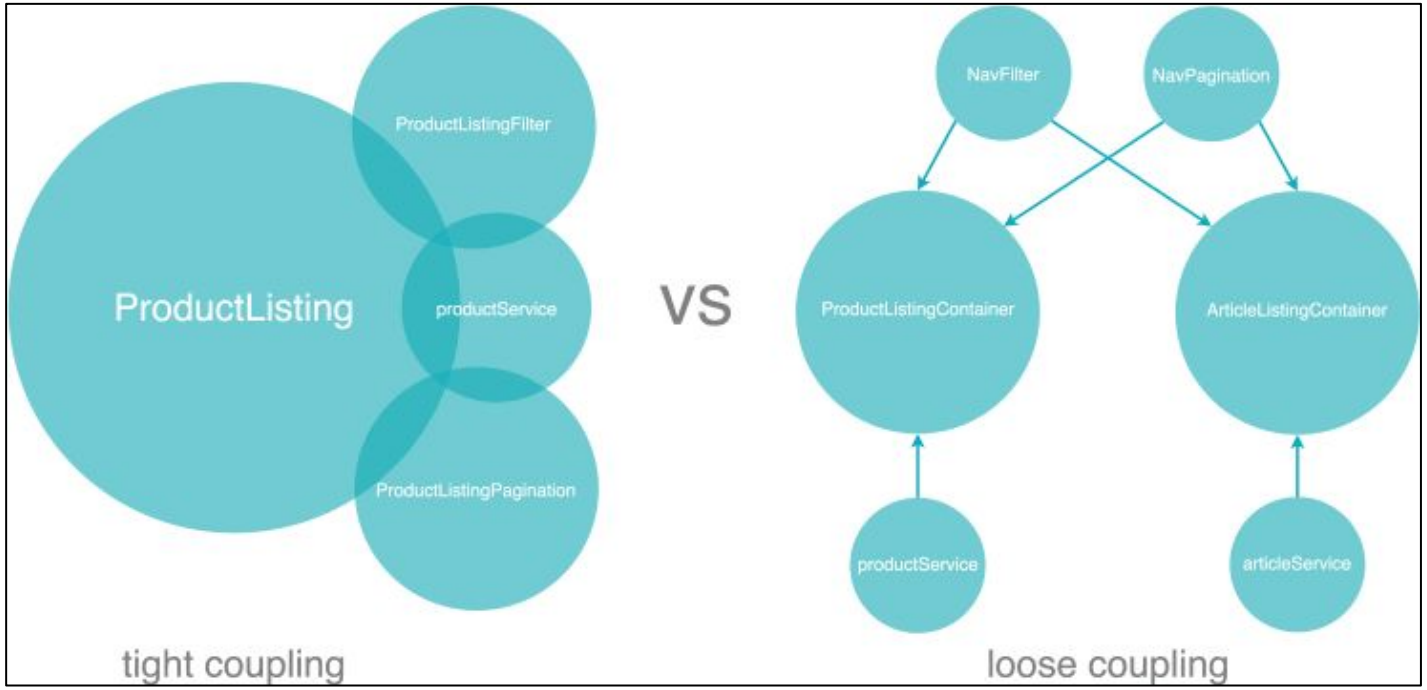
Monolith and MSA 구체적 사례



Loosely Coupled 란?

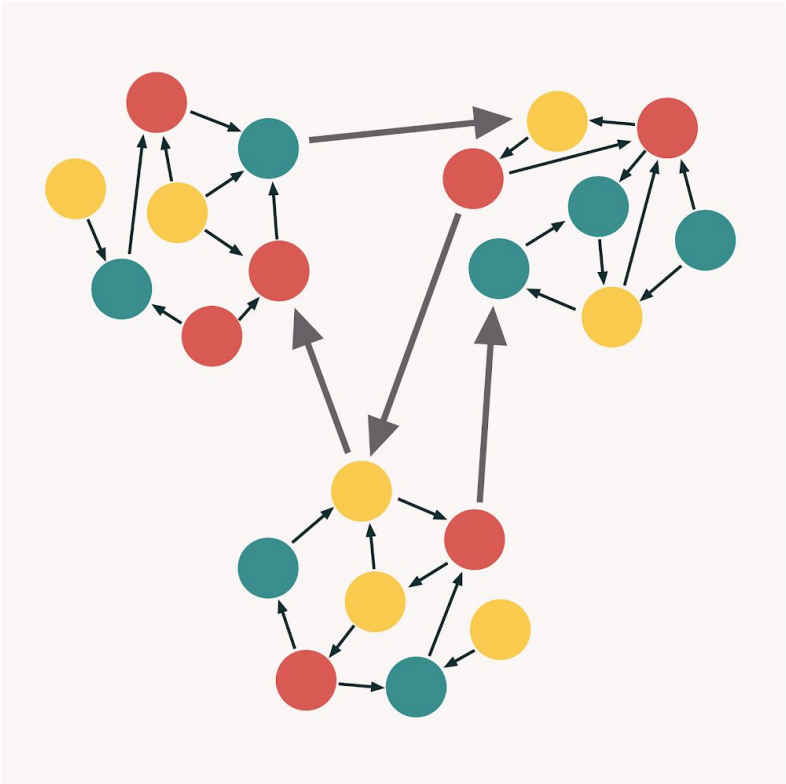
Loosely Coupled (느슨한 결합)

- 하나의 서비스가 변경 될 때 다른 서비스가 변경되는 일이 없다
- 시스템의 그 어떤 부분도 추가 변경한 필요 없이 특정 서비스를 변경하고 바로 배포 할 수 있는 것
- 강한 결합은 변경을 더 어렵게 하고 시험 가능성(testability)을 지연시킨다.



High Cohesion(응집력) 이란?

도메인은 다수의 경계가 있는 컨텍스트로 구성 됩니다. Bounded Context 는 명료한 경계에 의해 강제된 구체적인 책임 을 구분 합니다.



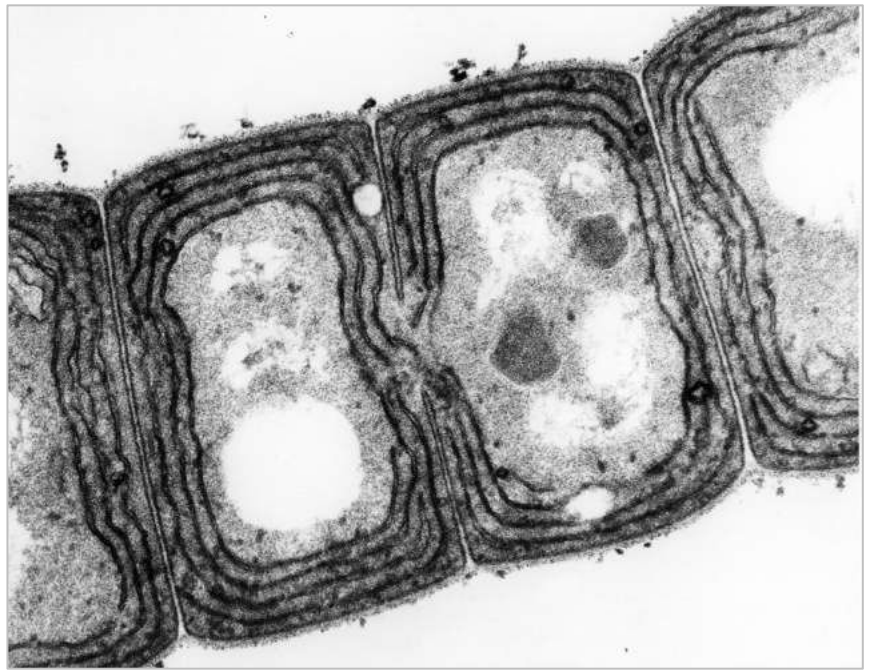
단일 책임 원칙(Single Responsible) 하에 하나의 마이크로 서비스 안에는 함께 변경 되는 것들은 같은 곳에 모으게 됩니다.

설계 시 주의 할 점은 업무 도메인을 정의 하고. 해당 업무 도메인 안에서 변경을 고려해 각각의 마이크로 서비스를 분리 해야 합니다.

Bounded Context(결정 경계) 란?

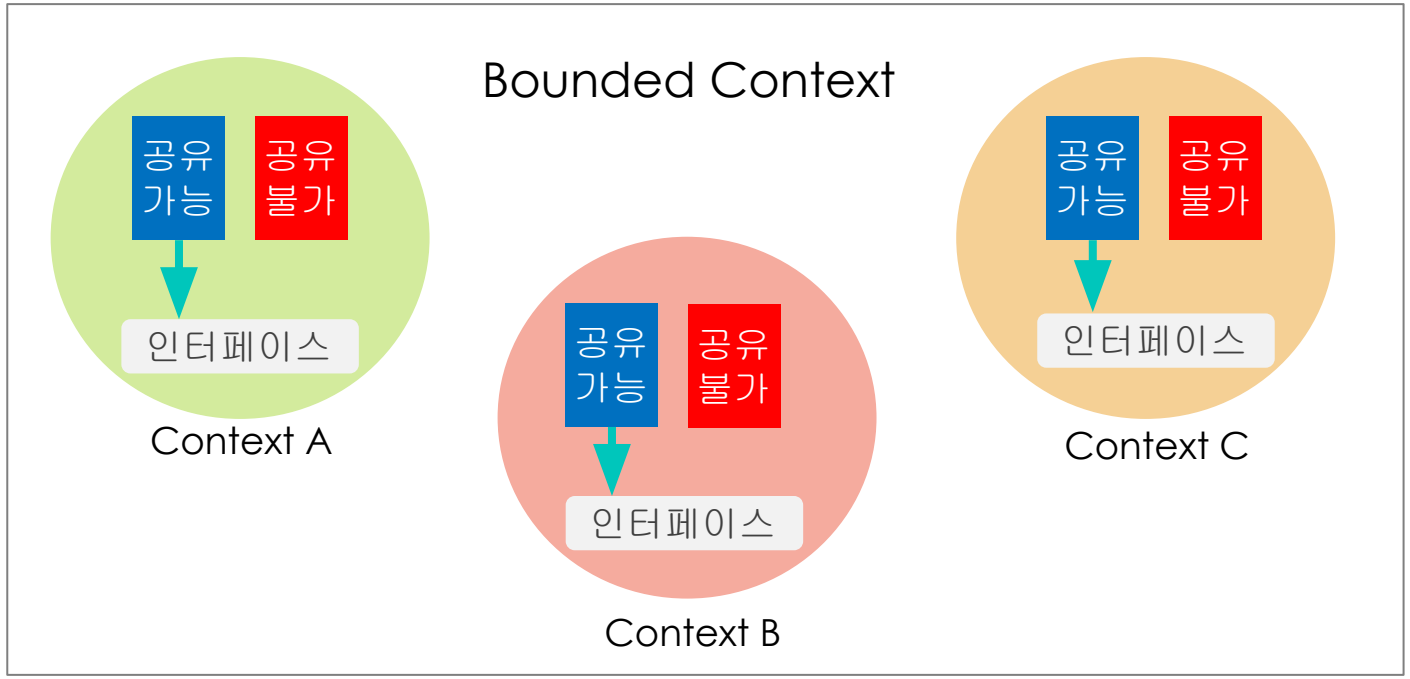
도메인은 다수의 경계가 있는 컨텍스트로 구성 됩니다. **Bounded Context** 는 명료한 경계에 의해 강제된 구체적인 책임 을 구분 합니다.

생물학

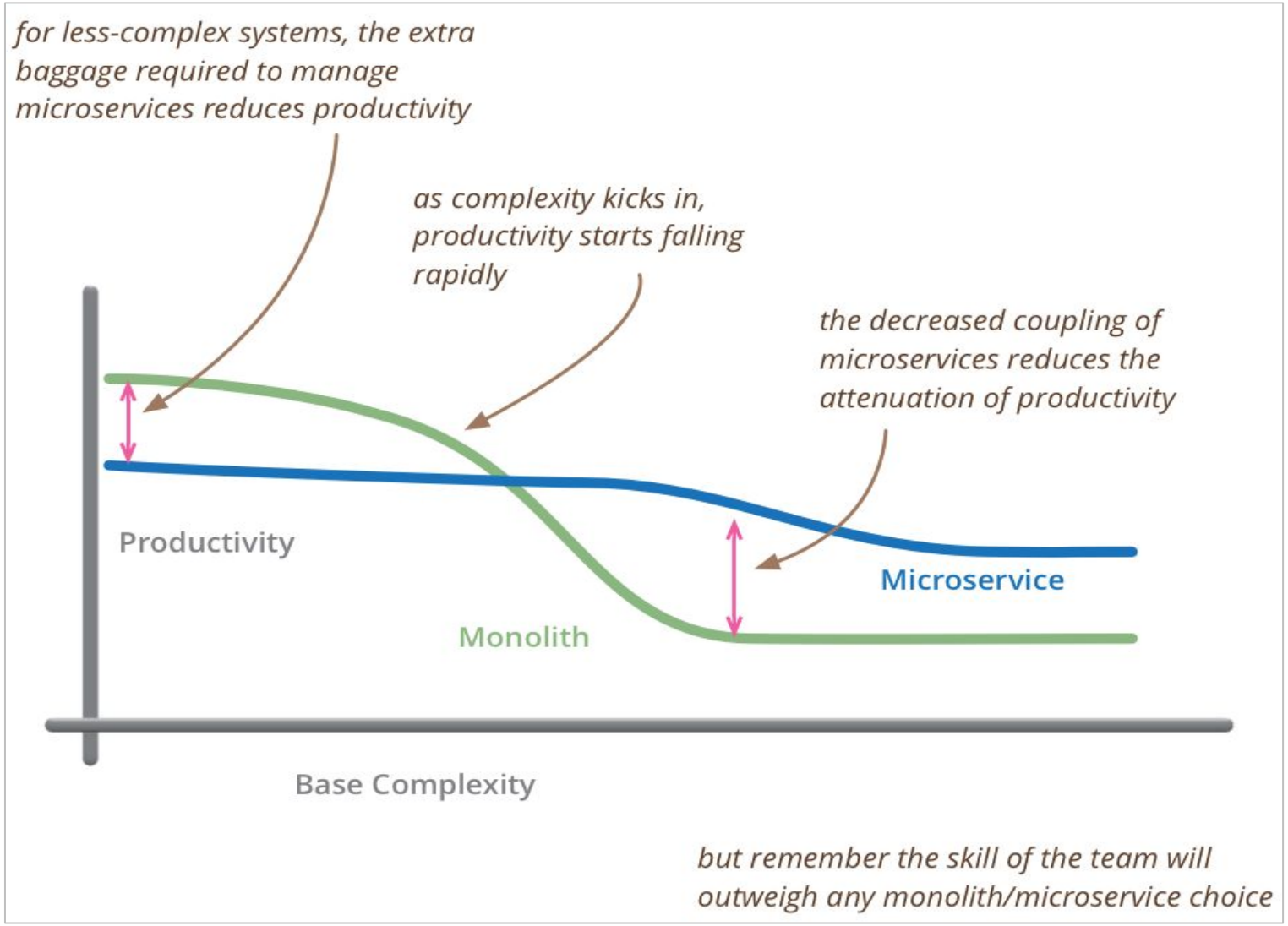


세포막에서 세포의 내부에 존재할 수 있는 것과 외부에 존재해야 하는 것을 정의하고 어떤 물질이 세포막을 통과할 수 있는지 결정

마이크로 서비스. 아키텍쳐



Monolith VS Microservice

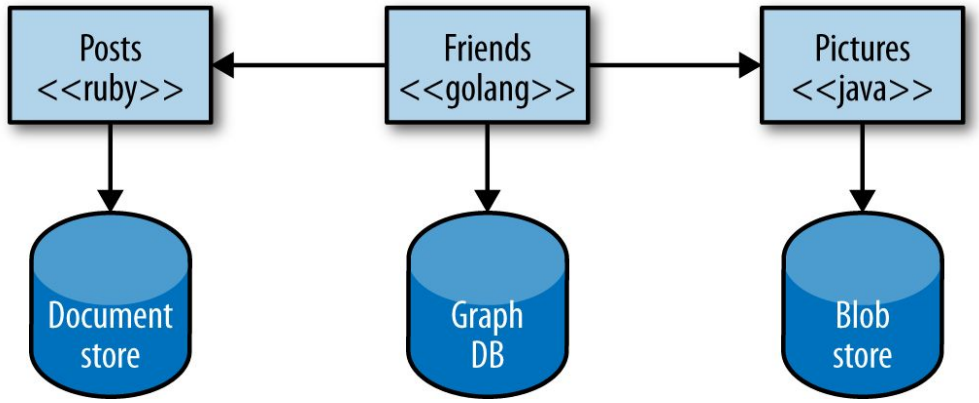


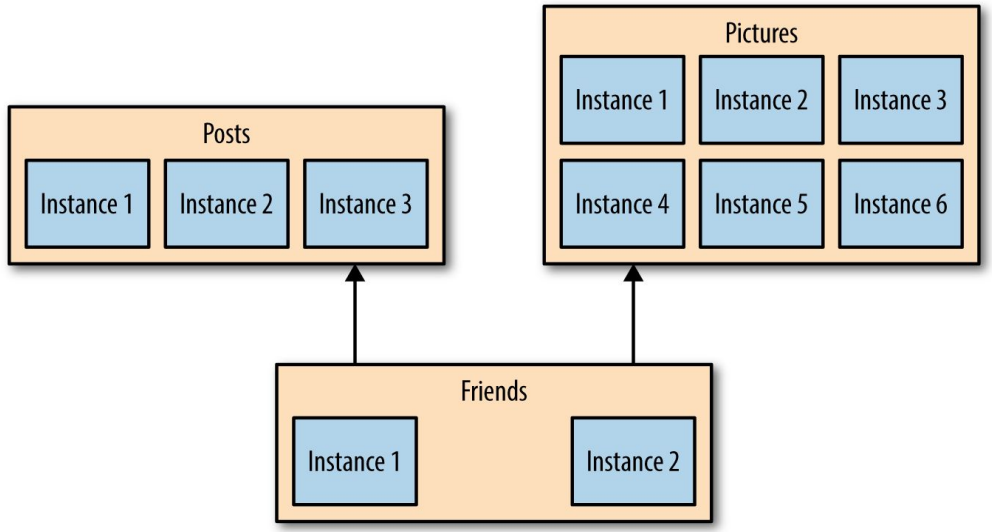
Monolith 의 경우 업무가 커짐에 따라 코드에 대한 **생산성이 감소**하고, **복잡성은 증가** 합니다.

마이크로 서비스 아키텍쳐(MSA)의 특징

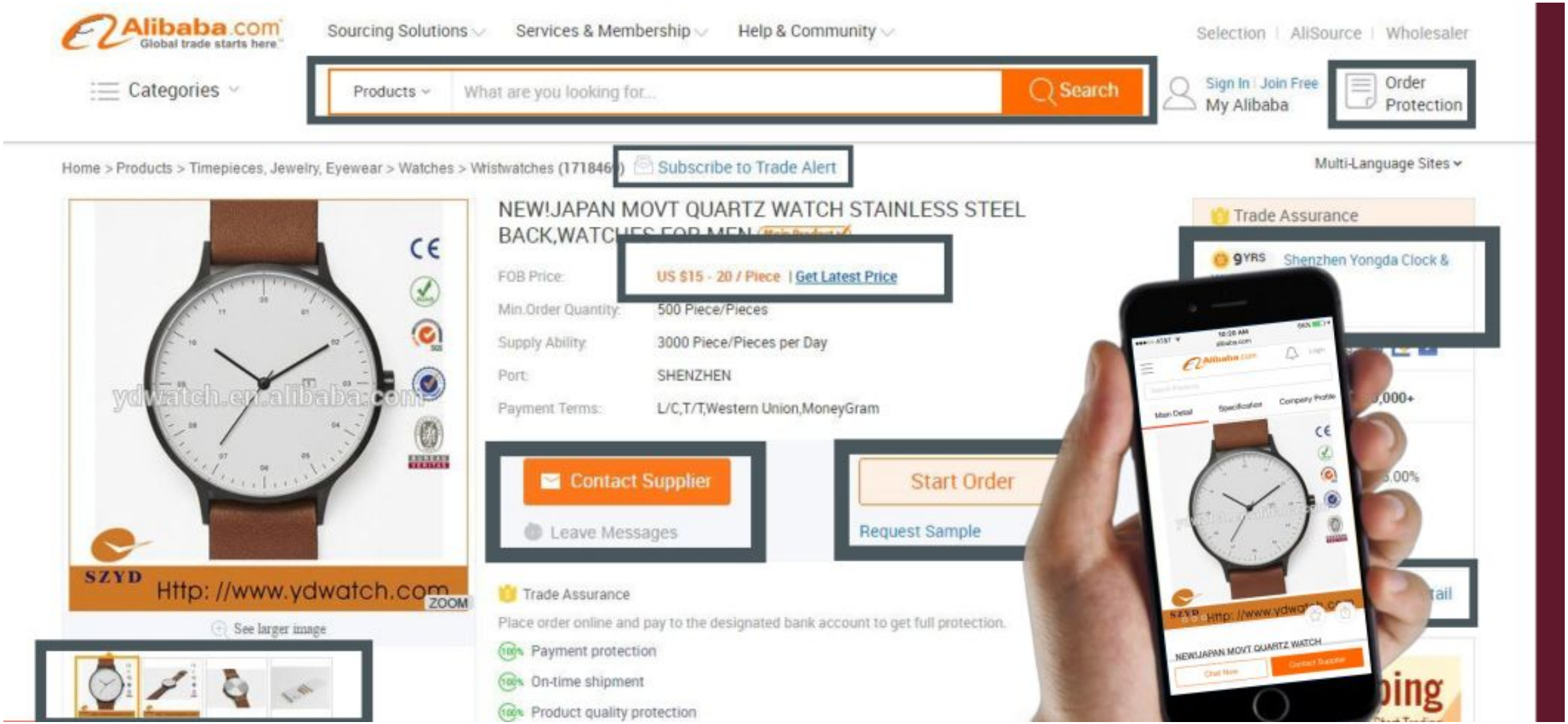
마이크로 서비스의 특징들

- 작고 한 가지 일을 잘하도록 설계
- 자율성 (Autonomous)
- 기술 이기종성 (Tech Heterogeneity)
- 회복성 (Resilience)
- 확장성 (Scaling)
- 배포 용이성 (Easy of Deployment)

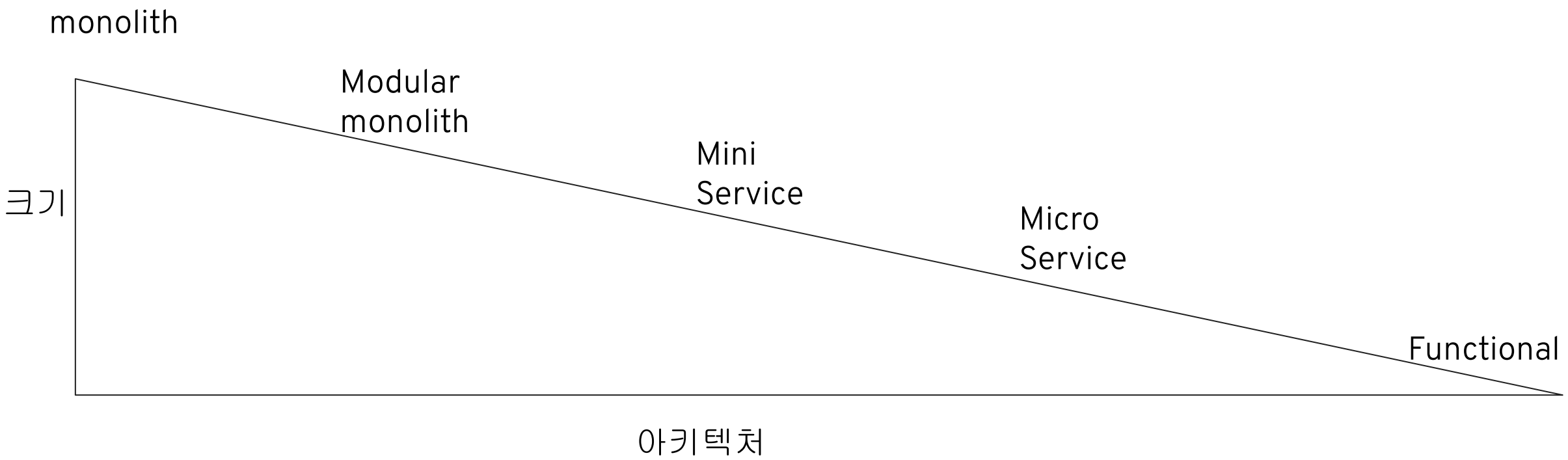




마이크로서비스 아키텍쳐 예



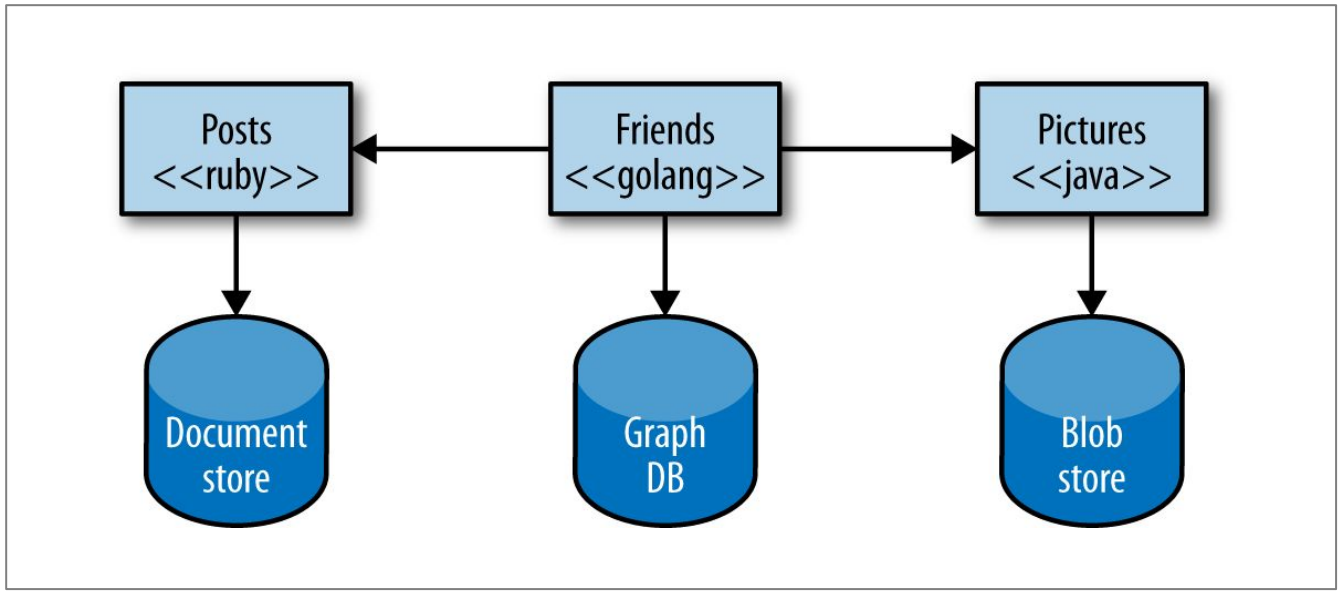
다양한 아키텍처


monolith
Modular
monolith
Mini
Service
Micro
Service
Functional
크기
아키텍처

마이크로 서비스 아키텍쳐의 특징 및 이점

1. 기술 이기종성(Hetrogeneous)

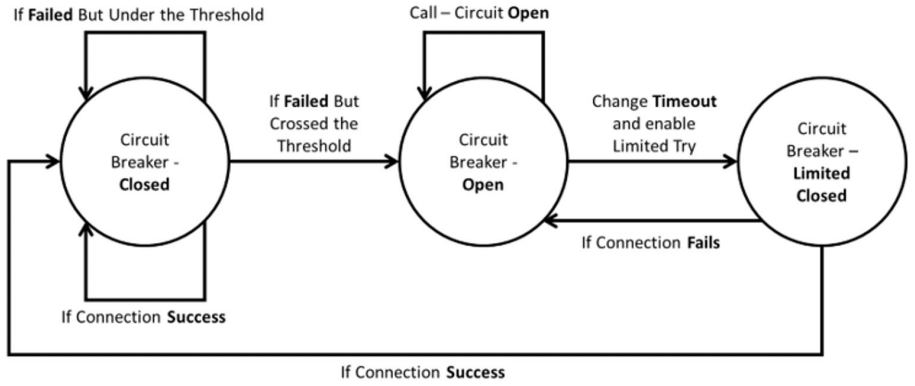
- 각각의 서비스가 다른 기술을 사용하여 구현 될 수 있다.
- 서비스에 맞는 기술을 채택 할 수 있음(성능, 데이타 모델, 구현용이성 등)
- 너무 많은 기술 스택은 오히려 부담

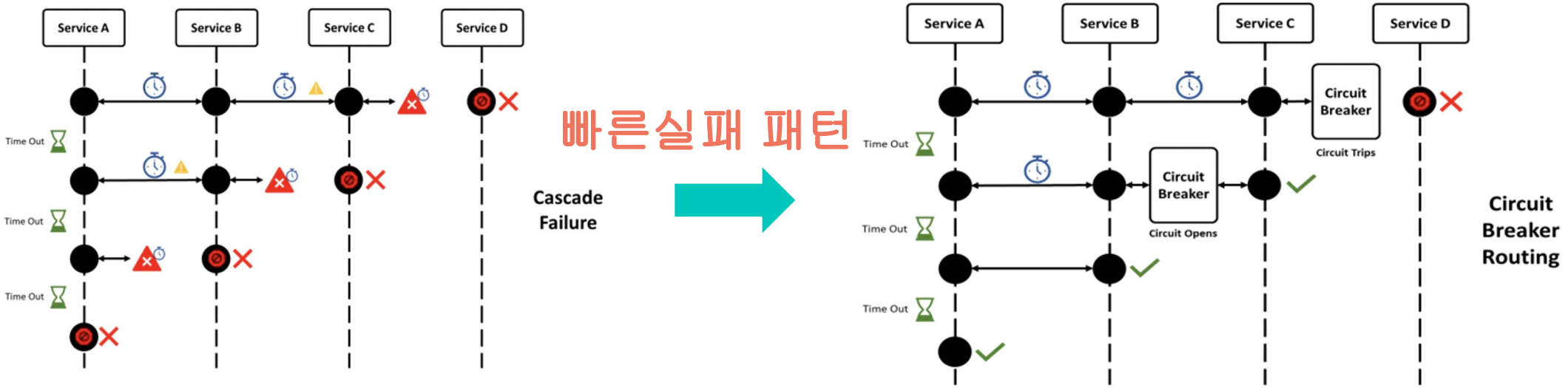


마이크로 서비스 아키텍쳐의 특징 및 이점 - 회복성

2. 회복성 (Resilience)

- 하나의 서비스 장애가 다른 서비스로 전이 되지 않음
- 전체 장애를 차단 하고 기능은 저하시킴
- 분산 기술에 대한 깊은 이해 필요



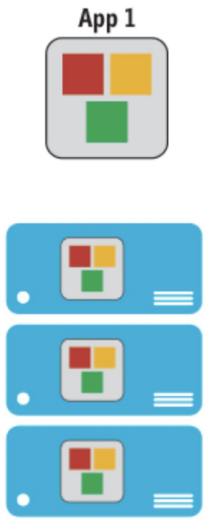


마이크로 서비스 아키텍쳐의 특징 및 이점 - 기술 이기종성

3. 확장성 (Scale Out)

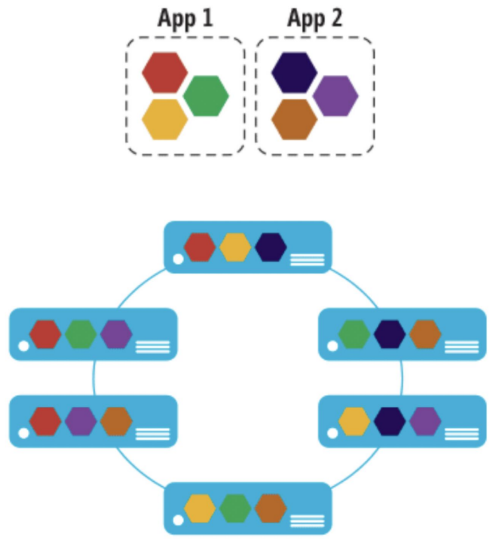
- 하나의 서비스 장애가 다른 서비스로 전이 되지 않음
- 전체 장애를 차단 하고 기능은 저하시킴
- 분산 기술에 대한 깊은 이해 필요
- 전통적인 ScaleOut 과 다름

Traditonal ScaleOut



APP 단위 배포 및 확장

Microservice ScaleOut

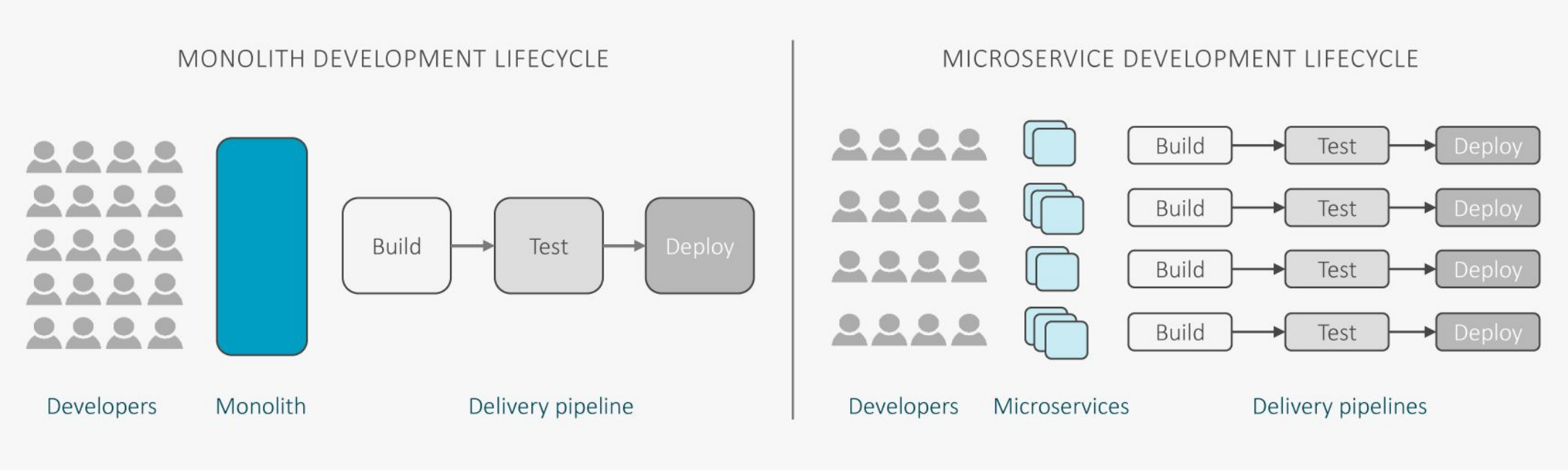


마이크로 서비스단위 배포 및 확장

마이크로 서비스 아키텍쳐의 특징 및 이점 - 기술 이기종성

4. 배포 용이성 (Easy Deployment)

- 전체 빌드(Build) 가 아닌 서비스 단위의 빌드 및 배포
- 테스트 코드 또는 TDD 필요
- 컨테이너 환경에 매우 적합
- 다같이 야근 안해도 됨



마이크로 서비스의 단점

업무 도메인 구분이 어렵다(경험 부족)

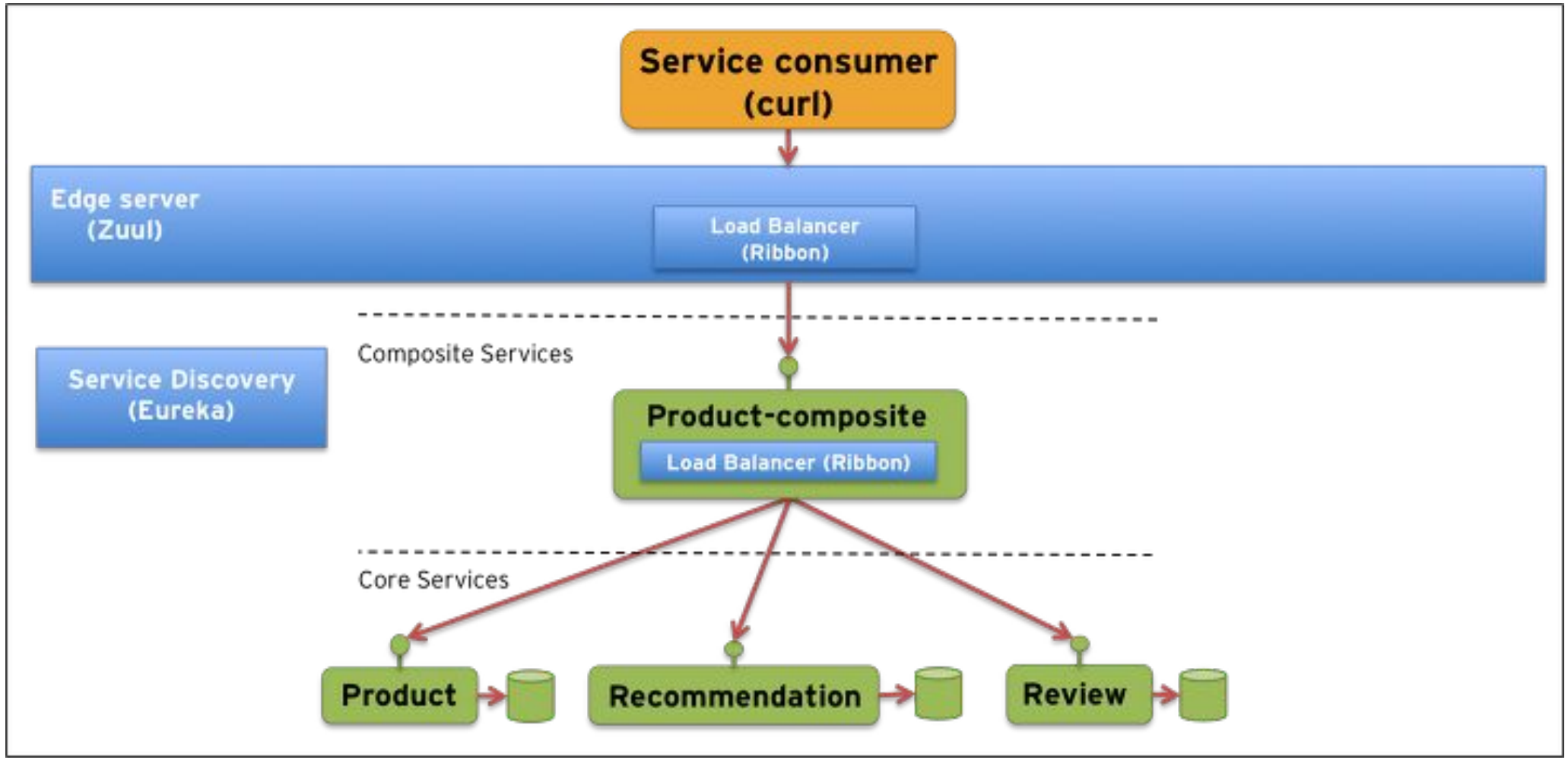
- 업무 도메인을 구분하여, 잘게 분리 하는 것이 생각보다 경험을 필요로 함.
- 초기 스타트업 에서는 MSA 권장하지 않음. (어떤 업무가 어떻게 나뉘게 될지 모름)
- 개발 인력의 기본적인 능력이 매우 중요

운영하기 힘들다

- 작은 서비스들이 흩어져 있기 때문에 모니터링이 쉽지 않음.
- 산재된 로그를 분석 하기가 힘들다
- 운영자가 마이크로 서비스를 구성하는 컴포넌트에 대해 정말 잘 알아야 함 (장애 대처 중요)

도커 등의 컨테이너 기술과. 쿠버네티스등 컨테이너 오케스트레이션이 대안으로 떠오른 이유

마이크로 서비스 아키텍쳐 OSS 예시



마이크로 서비스 Netflix OSS vs Kubernetes

Microservices Concern	Spring Cloud & Netflix OSS	Kubernetes
Configuration Management	Config Server, Consul, Netflix Archaius	Kubernetes ConfigMap & Secrets
Service Discovery	Netflix Eureka, Hashicorp Consul	Kubernetes Service & Ingress Resources
Load Balancing	Netflix Ribbon	Kubernetes Service
API Gateway	Netflix Zuul	Kubernetes Service & Ingress Resources
Service Security	Spring Cloud Security	-
Centralized Logging	ELK Stack (LogStash)	EFK Stack (Fluentd)
Centralized Metrics	Netflix Spectator & Atlas	Heapster, Prometheus, Grafana
Distributed Tracing	Spring Cloud Sleuth, Zipkin	OpenTracing, Zipkin
Resilience & Fault Tolerance	Netflix Hystrix, Turbine & Ribbon	Kubernetes Health Check & resource isolation
Auto Scaling & Self Healing	-	Kubernetes Health Check, Self Healing, Autoscaling
Packaging, Deployment & Scheduling	Spring Boot	Docker/Rkt, Kubernetes Scheduler & Deployment
Job Management	Spring Batch	Kubernetes Jobs & Scheduled Jobs
Singleton Application	Spring Cloud Cluster	Kubernetes Pods

Spring-Cloud and Kubernetes

Capability
DevOps Experience
Auto Scaling & Self Healing
Resilience & Fault Tolerance
Distributed Tracing
Centralized Metrics
Centralized Logging
API Gateway
Job Management
Singleton Application
Load Balancing
Service Discovery
Configuration Management
Application Packaging
Deployment & Scheduling
Process Isolation
Environment Management
Resource Management
Operating System
Virtualization
Hardware, Storage, Networking

Spring Cloud

Kubernetes

IaaS

Capability	Spring Cloud with Kubernetes
DevOps Experience	Self Service, multi-environment capabilities
Auto Scaling & Self Healing	Pod/Cluster Autoscaler, HealthIndicator(actuator), Scheduler
Resilience & Fault Tolerance	HealthIndicator(actuator), Hystrix, HealthCheck, Process Check
Distributed Tracing	Zipkin, Jaeger
Centralized Metrics	Heapster, Prometheus, Grafana
Centralized Logging	EFK, ELK
Load Balancing	Ribbon, Service
Service Discovery	Eureka, Service
Configuration Management	Externalized Configuration, ConfigMap, Secret
Application Packaging	Spring Boot Maven, Spring Boot Gradle
Deployment & Scheduling	Deployment Strategy, A/B, Canary, Scheduler Strategy
Process Isolation	Pods
Environment Management	Namespaces, Authorizations
Resource Management	CPU and Memory limits, Namespace resource quotas
IaaS	Cloud IaaS, VMware, OpenStacks

스프링 클라우드 컴포넌트를 코드 수정 거의 없이
쿠버네티스
기능으로 대체 하는 컴포넌트
https://github.com/spring-cloud/spring-cloud-kubernetes