



SMCH 2025 IT Conference



NULL4U

Cloud Native: Changing the paradigm of the internet.

Speaker / Presenter by **Kim Jun Young**

Sessions

Session 00 Presenter Introduction

Session 01 Hello, Null4U !

 by Null4U

Session 02 클라우드 컴퓨팅과 MSA, 그리고 Kubernetes가
도대체 뭐길래 웹 배포의 패러다임을 바꿨을까?

 by KJY

Session 03 AWS — 데이터베이스를 직접 구축하는 시대는 지났다.

 by KJY

Key Points of Sessions 1-3: What is **AWS** and **Kubernetes**?

Sessions

Session	04	Operator 패턴 — Kubernetes는 어떻게 단순 배포 도구를 넘어 클라우드 네이티브 생태계를 바꿀 수 있었을까?	_____ by KJY
Session	05	Istio — Kubernetes 서비스 메시의 핵심	_____ by KJY
Session	06	<i>Extra) 프로그래밍 7-8년차가 전해드리는 경험담과 팁들</i>	_____ by KJY

Key Points of Sessions 4-5: Deep Dive into **Kubernetes** and **AWS VPC**

Session 00. Presenter Introduction



Kim Jun Young *@yulmwu*

세명컴퓨터고등학교 스마트보안솔루션과 2학년

교내 클라우드 컴퓨팅/DevOps 전공동아리 Null4U 부장

- github.com/yulmwu
- Discord @rlawnsdud
- 010-2980-1336
- normal8781@gmail.com

DevOps/Backend 기술 블로그 운영

<https://velog.io/@yulmwu>



Session 00. Presenter Introduction



<https://vlog.io/@yulmwu>



개별/익명 Q&A (질문) 양식

세부적인 기술에 대한 설명보단 표면적인 개념을 주로 설명합니다.

이에 대한 자세한 내용은 **전공 동아리 세미나** 또는 **블로그**를 참고 해주시길 바랍니다.

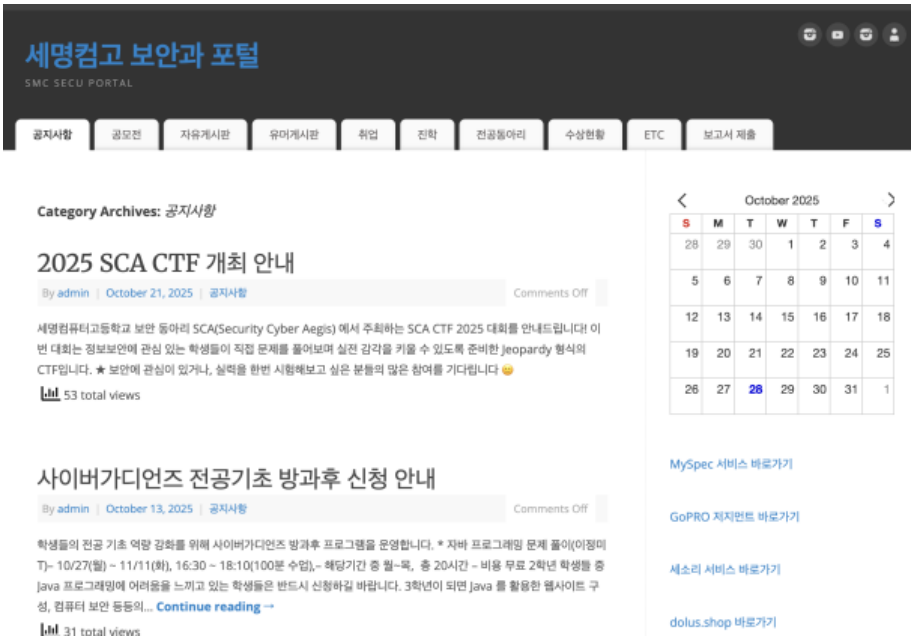
Session 01

Hello, Null4U !

“ 그래서, Null4U가 뭘 하는 동아리죠? 🤔 ”

Session 1. Hello Null4U ! (Feat. 그래서, Null4U가 뭘 하는 동아리죠?)

" 개발, 클라우드 컴퓨팅 구축 및 운영 "



보안과 포털



세소리 프로젝트



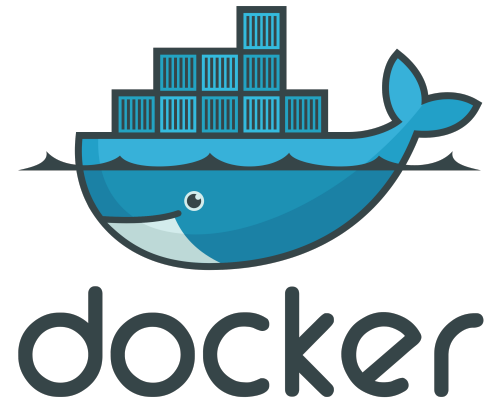
지방기능경기대회(클라우드 종목)

등등.. (동아리 외 개인 프로젝트 제외)

" 어떤걸 공부하나요? "

Session 1. Hello Null4U ! (Feat. 그래서, Null4U가 뭘 하는 동아리죠?)

" 어떤걸 공부하나요? "



kubernetes



Terraform

등등...

" Cloud Native "

+ 자격증, 풀스택 개발, 클라우드 보안, CI/CD 등등 가리지 않음

Session 1. Hello Null4U ! (Feat. 그래서, Null4U가 뭘 하는 동아리죠?)

" 무슨 자격증을 취득해볼 수 있죠? "

정보처리기능사, 정보기기운용기능사, 리눅스마스터 (국내)

AWS Certification (CCP 등), Kubernetes Certification (CKA 등)

Session 1. Hello Null4U ! (Feat. 그래서, Null4U가 뭘 하는 동아리죠?)

그 외에... 전문가 특강

2025 클라우드

 배경 지식

 Terraform

 Container Orchestration(EKS, Kubernetes)

 1과제 (1)

 2025-09-28 2차시

(고급 강의) (IaC with Terraform, AWS Elastic Kubernetes Service ...)

등등

Session 02

클라우드 컴퓨팅과 MSA, 그리고  **kubernetes** 가
도대체 뭐길래 웹 배포의 패러다임, 그리고 생태계를 바꿨을까?

Session 2 ToC

01	• google.com은 어떻게 우리의 검색을 처리하고 응답할까? • 창업을 하고 서버 운영자가 되어보자. • 우리에게엔 클라우드라는 선택지가 남아있다. (Feat. AWS)	클라우드와 AWS
02	• 개발자는 어떻게 실제 서비스에 사용할 수 있도록 애플리케이션을 배포할까? • 컨테이너, 개발 및 운영 생태계를 바꾸다.	컨테이너와 도커
03	• 결제 서비스에 문제가 생겼다, 이를 어찌지? • 그래서, MSA가 뭔데?	MicroService Architecture
04	• 컨테이너는 언제든지 죽고 살아날 수 있다. • 수 많은 컨테이너를 어떻게 관리하지?	Kubernetes
05	• 이제는 한가지 기술만을 요구하는 시대가 아니다.	DevOps

Session 2-1. google.com은 어떻게 우리의 검색을 처리하고 응답할까?



🔍 세명컴퓨터고등학교



Session 2-1. google.com은 어떻게 우리의 검색을 처리하고 응답할까?

세명컴퓨터고등학교 - Google 검색

google.com/search?q=세명컴퓨터고등학교&rlz=1C5CHFA_enKR1016KR1016&oq=세&gs_lcrp=EgZjaHJvbWUqDAgBECMYJxiABBiBTIGCAAQRRg5MgwIARAJGccYg...

Google

세명컴퓨터고등학교

AI 모드 전체 이미지 동영상 뉴스 쇼핑 지도 더보기 도구

나무위키

https://namu.wiki › 세명컴퓨터고등학교

세명컴퓨터고등학교

서울특별시 은평구 불광동에 있는 IT(공업계열) 특성화고등학교이다. 현재 학교법인 세명학원 소속으로 되어있다.[5] 학교 이름을 줄여서 세명컴고, 세컴고, ...

학교알리미

https://www.schoolinfo.go.kr › Pneiss_b01_s0

세명컴퓨터고등학교 학교정보

학생수 : 445명, 교원수 : 68명, 설립구분 : 사립, 설립유형 : 단설, 설립일자 : 1974년 04월 29일, 대표번호 : 02-354-1323, 팩스 : 02-388-1381, 주소 : 서울특별시 ...

Instagram · semyeong_com_hs

팔로어 660명 이상

세명컴퓨터고등학교 (@semyeong_com_hs)

세명컴퓨터고등학교 공식 계정. Official Instagram of Semyeong Computer High school. ▾ 홈페이지 & 유튜브 & 진로 체험 및 설명회 신청 링크 ... Photo by 세명컴퓨터 ...

Wikipedia

https://ko.wikipedia.org › wiki › 세명컴퓨터고등학교

세명컴퓨터고등학교

세명컴퓨터고등학교(世明컴퓨터高等學校)는 대한민국 서울특별시 은평구 불광동에 위치한 사립 고등학교이다. 세명컴퓨터고등학교 世明컴퓨터高等學校

사진 보기

시골순대국

세명컴퓨터고등학교

대한민국 서울특별시의 고등학교

웹사이트

경로

리뷰

저장

공유

전화 걸기

세명컴퓨터고등학교는 대한민국 서울특별시 은평구 불광동에 위치한 사립 고등학교이다. 위키백과

주소: 대한민국 서울특별시 은평구 통일로92가길 33

창립: 1970년

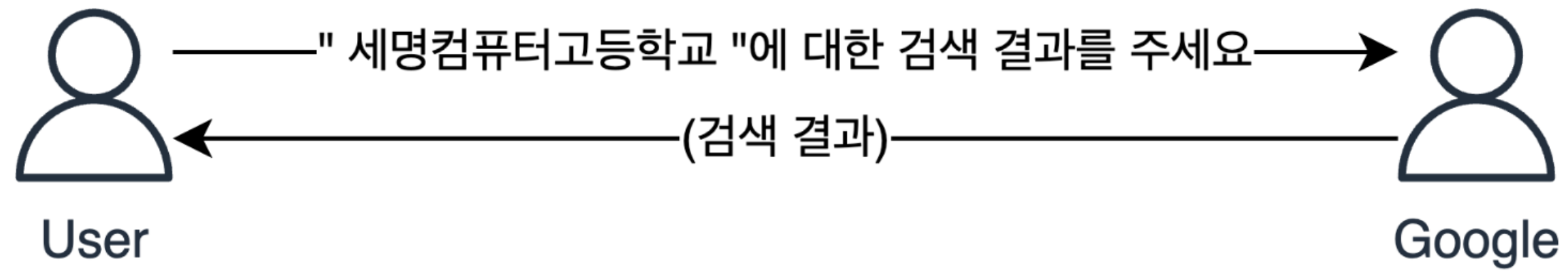
모회사: 대원교육재단

전화번호: +82 2-354-1321 ext. 4

교장·여지배

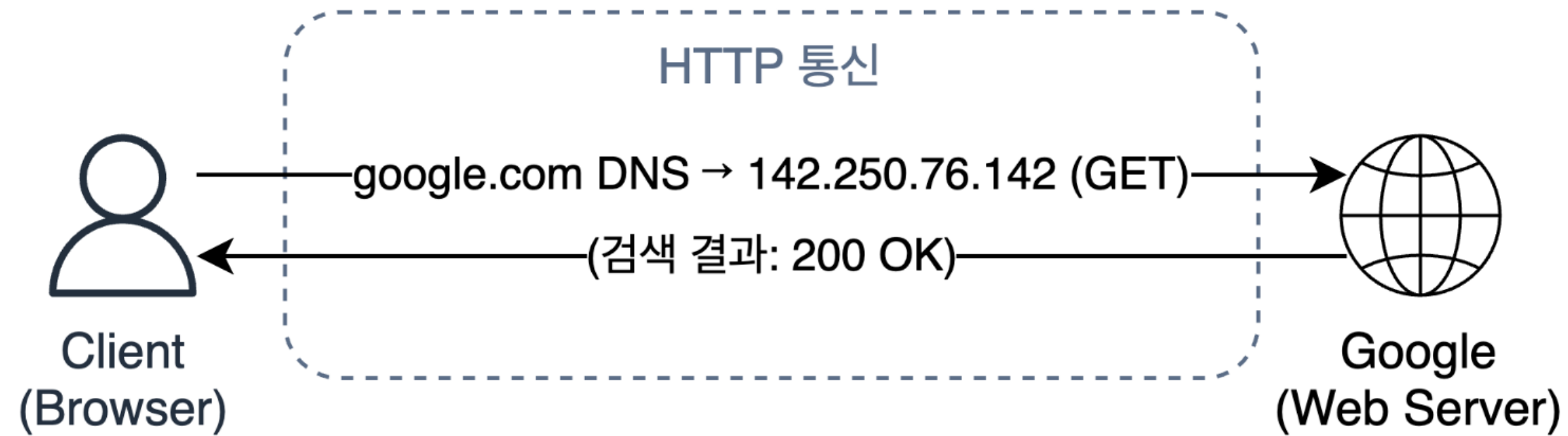
" 무슨 원리로 검색을 하면 이런 결과가 나올까 ? "

Session 2-1. google.com은 어떻게 우리의 검색을 처리하고 응답할까?



요청(Request)과 응답(Response)

Session 2-1. google.com은 어떻게 우리의 검색을 처리하고 응답할까?



이러한 주고 받기를 HTTP 통신
트래픽을 보내는 쪽은 **클라이언트**(브라우저), 받아서 처리하는 쪽이 (웹)**서버(Server)**

Session 2-1. 창업을 하고 서버 운영자가 되어보자.

서버를 운영하기 위해선 근본적으로 **서버가 필요**

직접 서버 컴퓨터를 구축하려면? ⇒ 서버(컴퓨터) 구매 및 펌웨어, OS 등을 하나하나 다루고 주기적인 유지보수 필요

⇒ 초기 인프라 구축 비용 큼(네트워크 구축, UPS 등의 전기 설비까지 포함)

⇒ 유지보수도 어렵고 보안 솔루션을 직접 만들어야함

⇒ 성능 확대 시 규모 확장도 어려움 (새로 서버를 사야함) = 확장성 문제

⇒ 인터넷 데이터센터(IDC) 자체 비용도 만만치 않음(전기세, 인터넷 사용료, 냉각 시스템 등등)

(흔히 "서버실"이라고 부르는 곳 IDC, 출처: 한국경제)



Session 2-1. 창업을 하고 서버 운영자가 되어보자.

" 집에 있는 컴퓨터로 서버 만들면 되는거 아니가요? "

⇒ 서비스 규모에 따라 다르겠지만, 일반적인 실제 서비스라면 PC방 모든 컴퓨터를 풀로 가동해도 감당 불가능함
+ 분산 컴퓨팅, 안정적인 스토리지 구축(DB, File System..), 네트워크 토폴로지 구축, UPS 등 전기 설비, 시설(냉방, 소방 등등..)

(물론 MAU가 적은 소규모 서비스라면 그런 컴퓨터나 라즈베리파이 등을 활용하여 서버를 구축하는 경우도 있음)

⇒ **On premise**, 기업이 자체적으로 서버와 같은 인프라를 직접 소유하고 관리하는 방식

다만 보안/법적 문제, 마이그레이션 문제 등으로 인해 온 프레미스를 고집하는 기업이나 기관도 많음
(대표적으로 거의 대부분의 정부 기관 서비스)

(애플은 백엔드 프레임워크도 자체적으로 개발해서 씬 = 전자정부프레임워크)

Session 2-1. 우리에게 클라우드라는 선택지가 남아있다. — AWS

" 서버 컴퓨터를 구매하는 것이 아닌 업체에서 빌리면 되지 않을까? " ⇒ 클라우드(Cloud)

인터넷을 통해 업체 등에서 컴퓨팅 서버, 스토리지(DB) 등을 빌려 사용하는 것

⇒ 빌린 만큼 요금을 청구 (On Demand)



Amazon Web Services



Google Cloud Platform



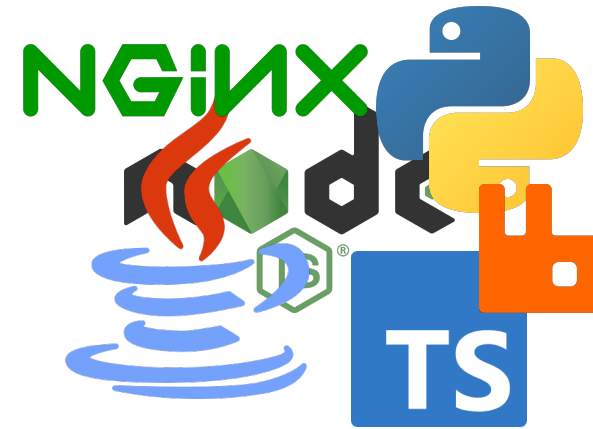
Microsoft Azure



그러한 클라우드 컴퓨팅 서비스 제공 업체 중 1위 "AWS"

Session 2-2. 개발자는 어떻게 실제 서비스에 사용할 수 있도록 애플리케이션을 배포할까?

솔루션 A. 서버에 직접 접속하여 코드 수정하기



(복잡한 애플리케이션 구조)

서버 접속 > 코드 수정 > 애플리케이션 관리 등등, 복잡함 + 유지보수에 효율적이지 못함, 기존의 의존성 충돌 등

Session 2-2. 개발자는 어떻게 실제 서비스에 사용할 수 있도록 애플리케이션을 배포할까?

솔루션 B. 하이퍼바이저(가상머신, VM) 사용하기

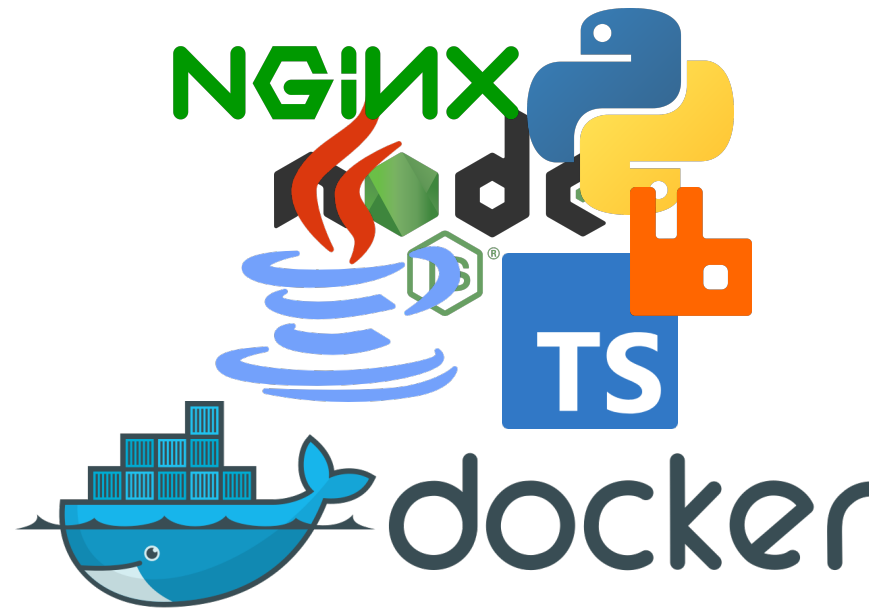


가상머신 사용: 독립된 환경에 분리되어 충돌 적음, 효율적인 자원 활용, 환경 분리 용이(보안 이점)

단점: 무거움(OS 자체를 가상화) ⇒ 같은 컴퓨터에 필요한 라이브러리, 패키지, 아키텍트만 여러개 올려둘 순 없을까?

Session 2-2. 개발자는 어떻게 실제 서비스에 사용할 수 있도록 애플리케이션을 배포할까?

솔루션 C. 컨테이너(Container) 기술 사용하기



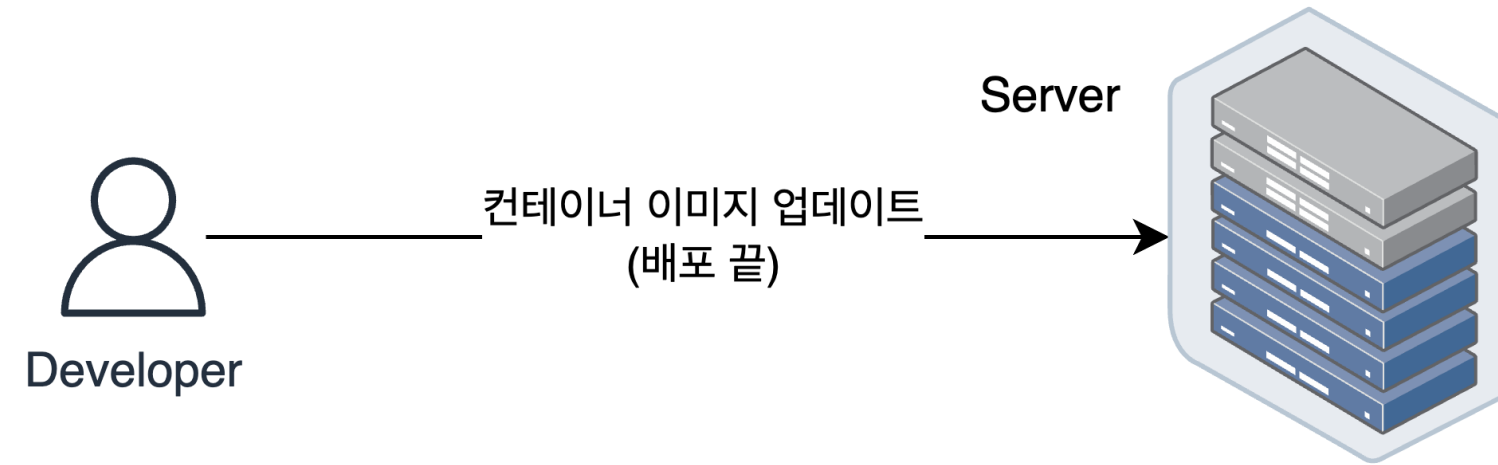
(도커, Containerd 등의 컨테이너 기술)

컨테이너 사용: 필요한 애플리케이션/런타임/라이브러리만 패키징(이미지), 하나의 호스트 OS(커널) 공유

어디서든 똑같이 실행된다, VM보다 빠르게 확장가능, 가벼움 등

Session 2-2. 컨테이너, 개발 및 운영 생태계를 바꾼다.

컨테이너 기술의 사용으로 서버에 애플리케이션을 배포하기가 매우 쉬워짐



실행 환경의 일관성 확보 및 배포 자동화 및 속도 향상

그런데 하나의 웹 서버 애플리케이션과 하나의 서버로 동작할 수 있을텐데 굳이 쓸 필요가 있을까?

Session 2-3. 결제 서비스에 문제가 생겼다, 이를 어찌지?

어제 릴리즈했던 백엔드에서 라이브러리 이슈로 특정 클라이언트 대상으로 결제 미들웨어 부분에서 에러가 발생합니다.

3
오후 3:18

김 팀장님

음, 일단 다른 업무 잠시 미뤄두고, 모니터링하면서 빠른 원인 보고 부탁드립니다.

3
오후 3:19

최 대리님

방금 해당 라이브러리 자체에서 발생하는 제로데이 버그라고 보고가 나와서 일단 이전 버전으로 롤백해야 할 것 같습니다.

3
오후 3:22

김 팀장님

애플리케이션 전체를 롤백하는데 2~3 시간 정도 걸리는데 그럼 SLA에 위배됩니다. 최대한 롤백하지 않는 선에서 조치가 가능할까요?

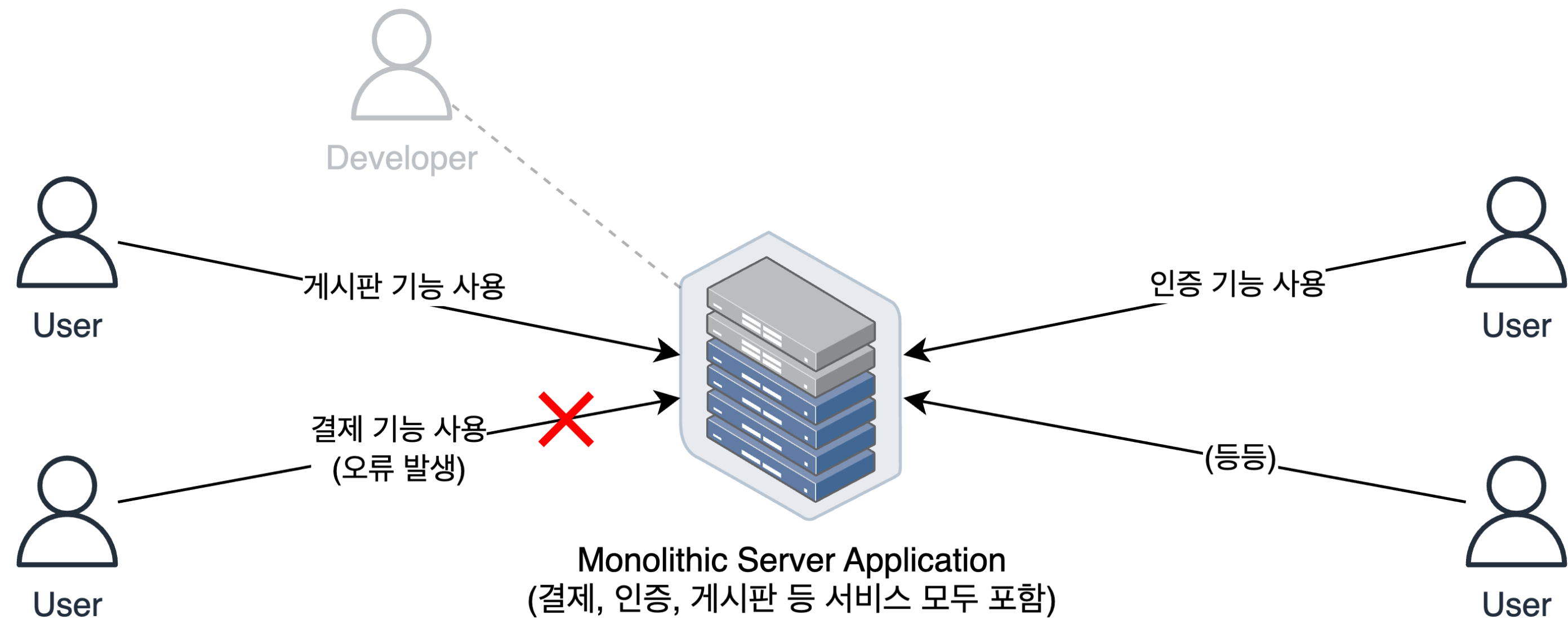
4
오후 3:23

백엔드 서버 애플리케이션에서 문제가 발생

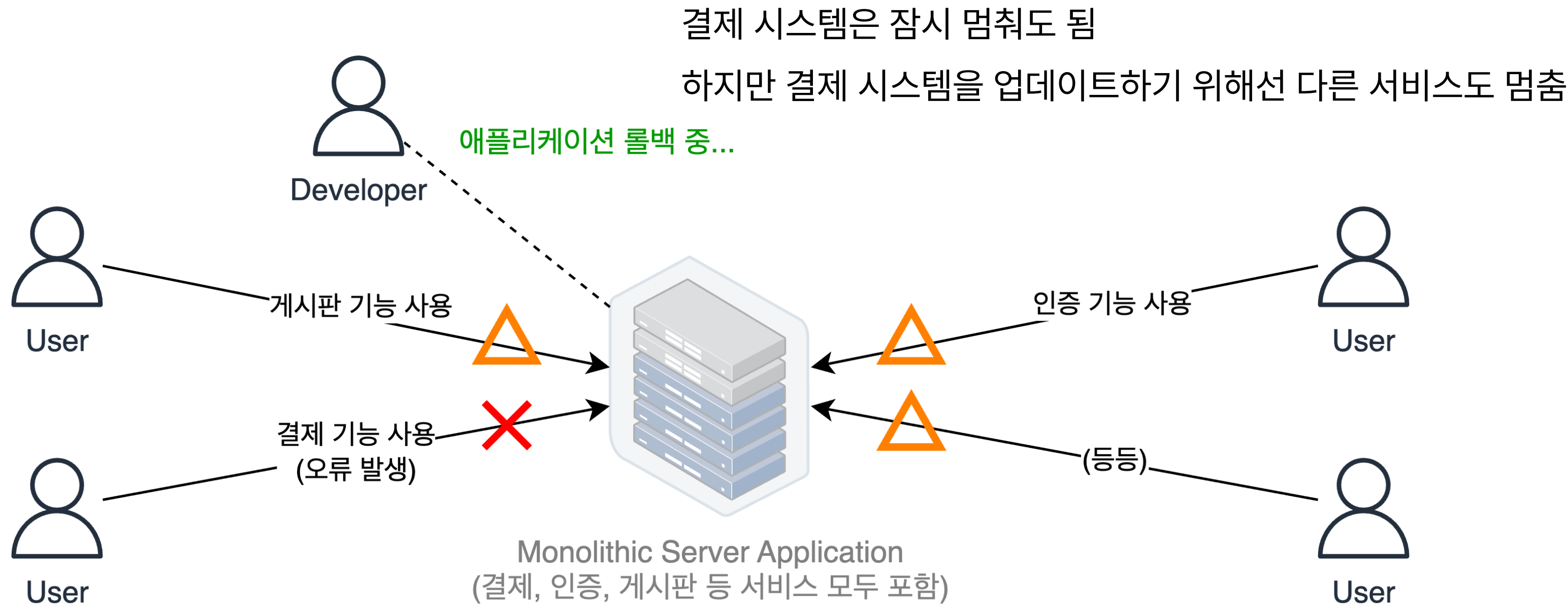
⇒ 이전 버전으로 돌리기엔 애플리케이션이 방대하여 시간 문제로 인해 이전 버전으로 롤백하기 어려운 상황

만약 각 서비스를 분리하여 개별적인 애플리케이션으로 운영했다면 어땠을까?

Session 2-3. 그래서, MSA가 뭔데?

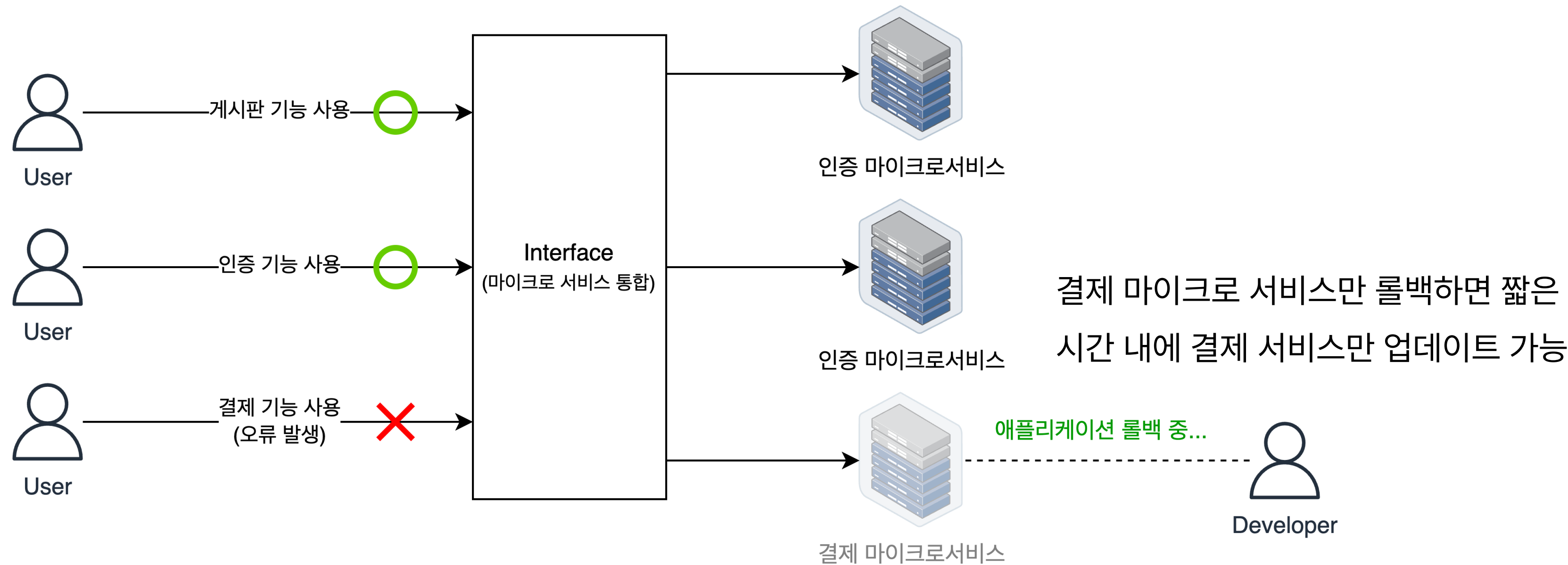


Session 2-3. 그래서, MSA가 뭔데?



" 모놀리식(Monolithic) 아키텍처 "

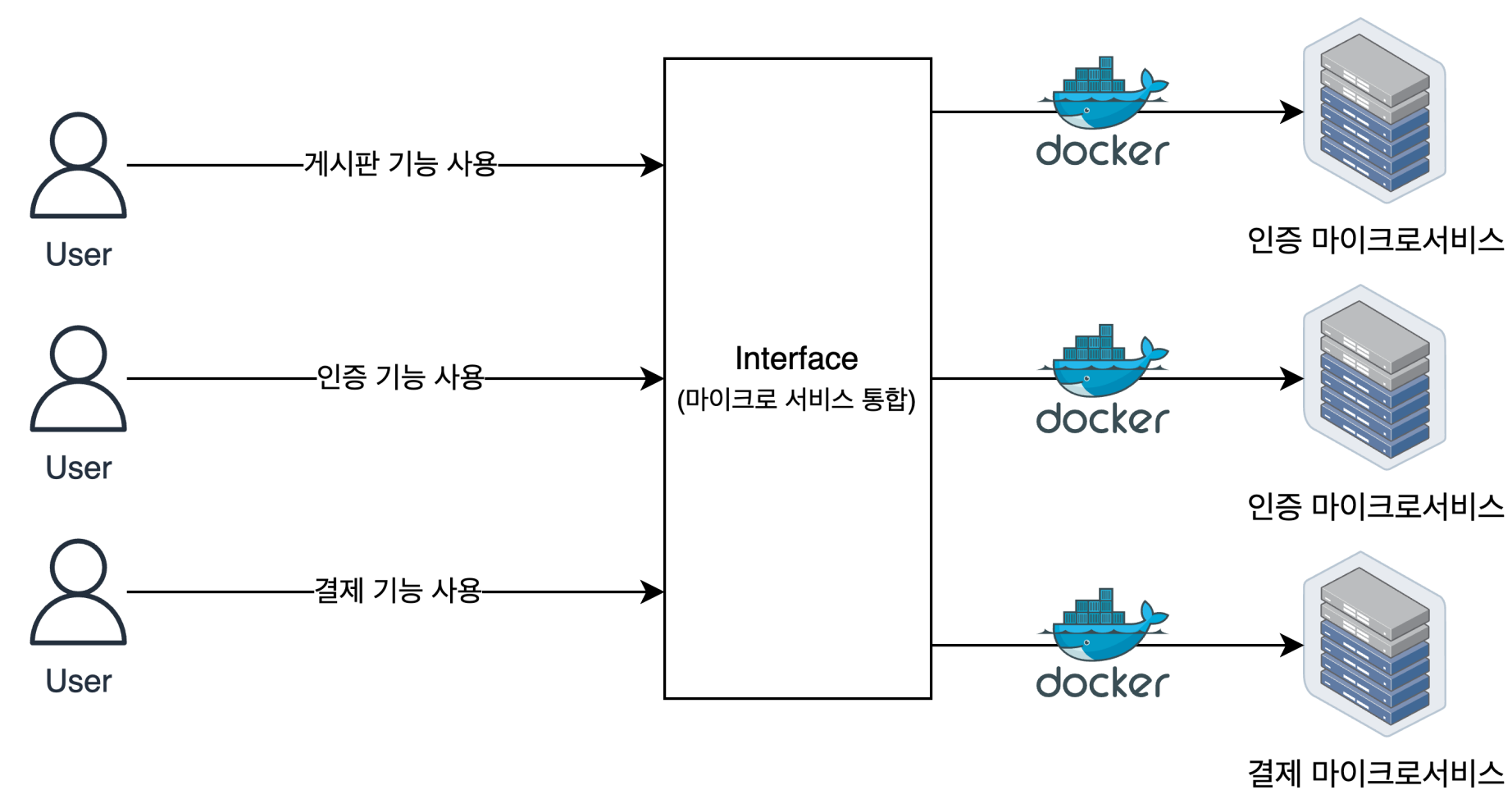
Session 2-3. 그래서, MSA가 뭔데?



" Micro Service Architecture (MSA) "

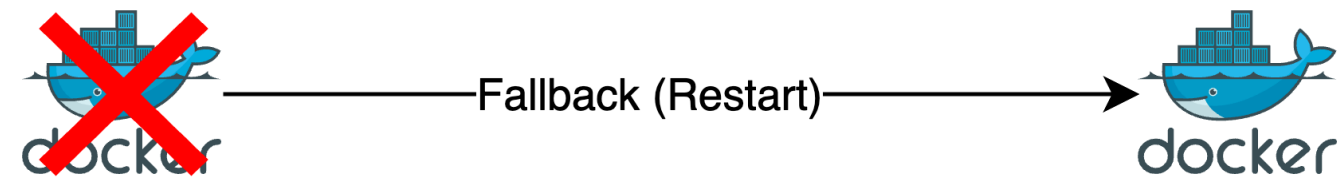
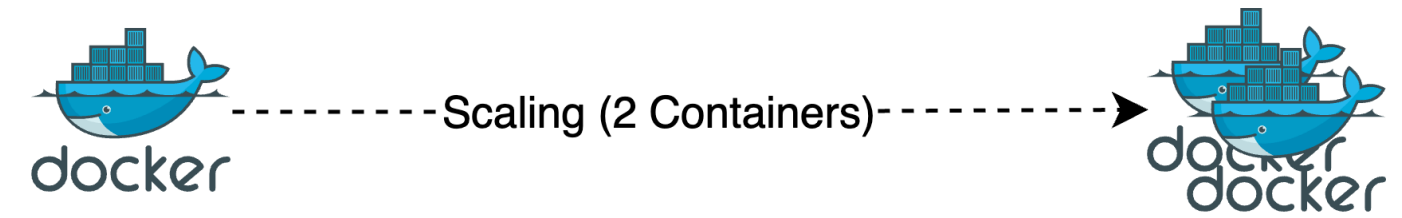
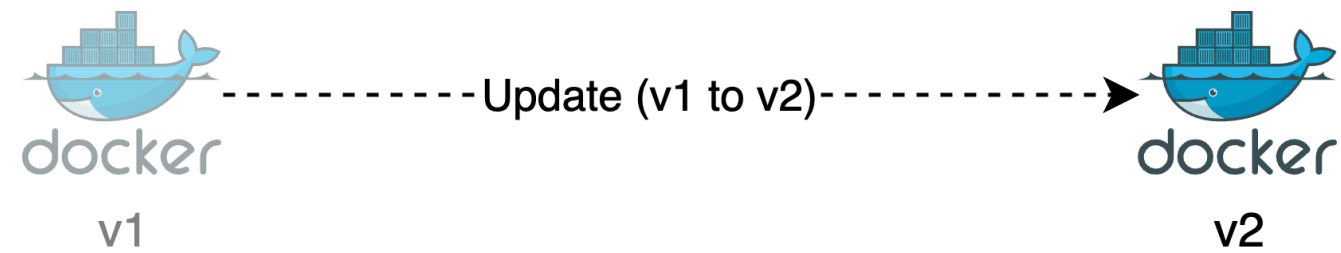
Session 2-3. 그래서, MSA가 뭔데?

각 마이크로 서비스가 개별적인 애플리케이션으로 동작하기 때문에 컨테이너로 배포하면 효율적임



수 많은 마이크로 서비스를 컨테이너로 효율적이게 운영
(Amazon과 Netflix의 마이크로서비스 간 서비스 메시 연결 구조)

Session 2-4. 컨테이너는 언제든지 죽고 살아날 수 있다.



목적이나 필요에 따라 새롭게 시작되거나 종료됨

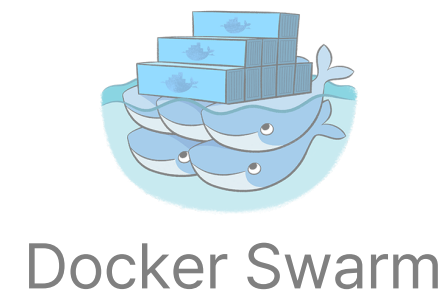
" 컨테이너 ≡ 일회성 소모품(Disposable) "

Session 2-4. 수 많은 컨테이너를 어떻게 관리하지?



실제 서비스 환경에선 수백~수천개의 컨테이너가 동작함

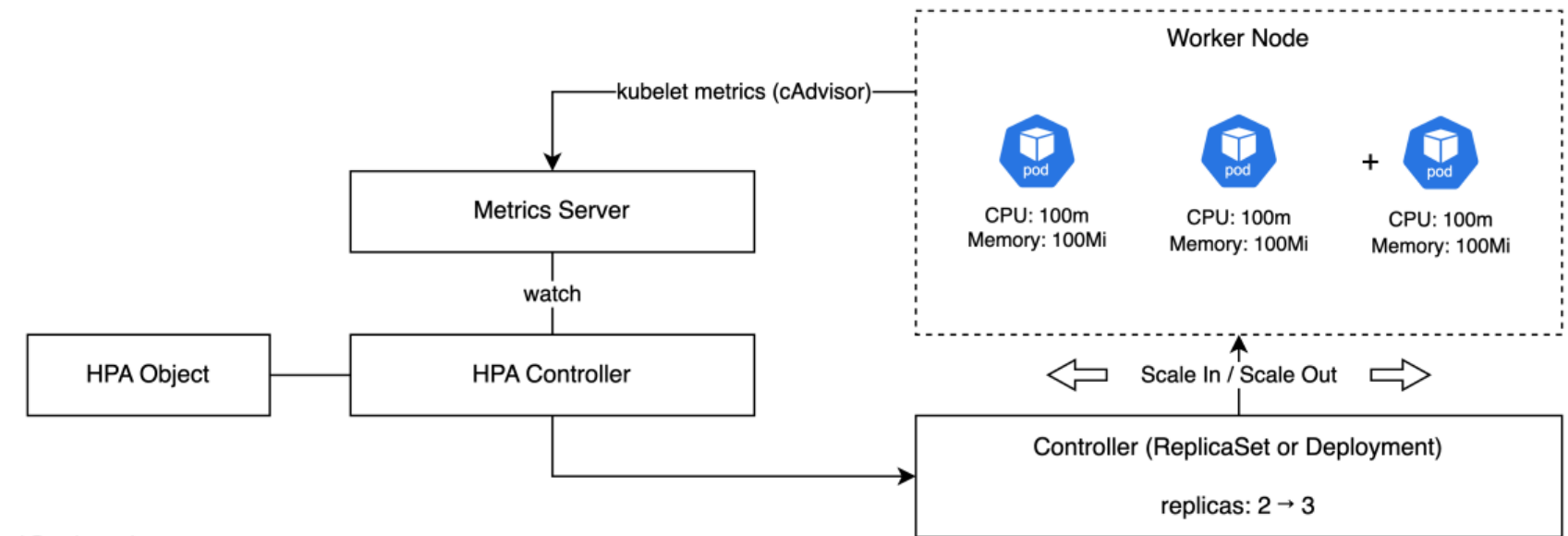
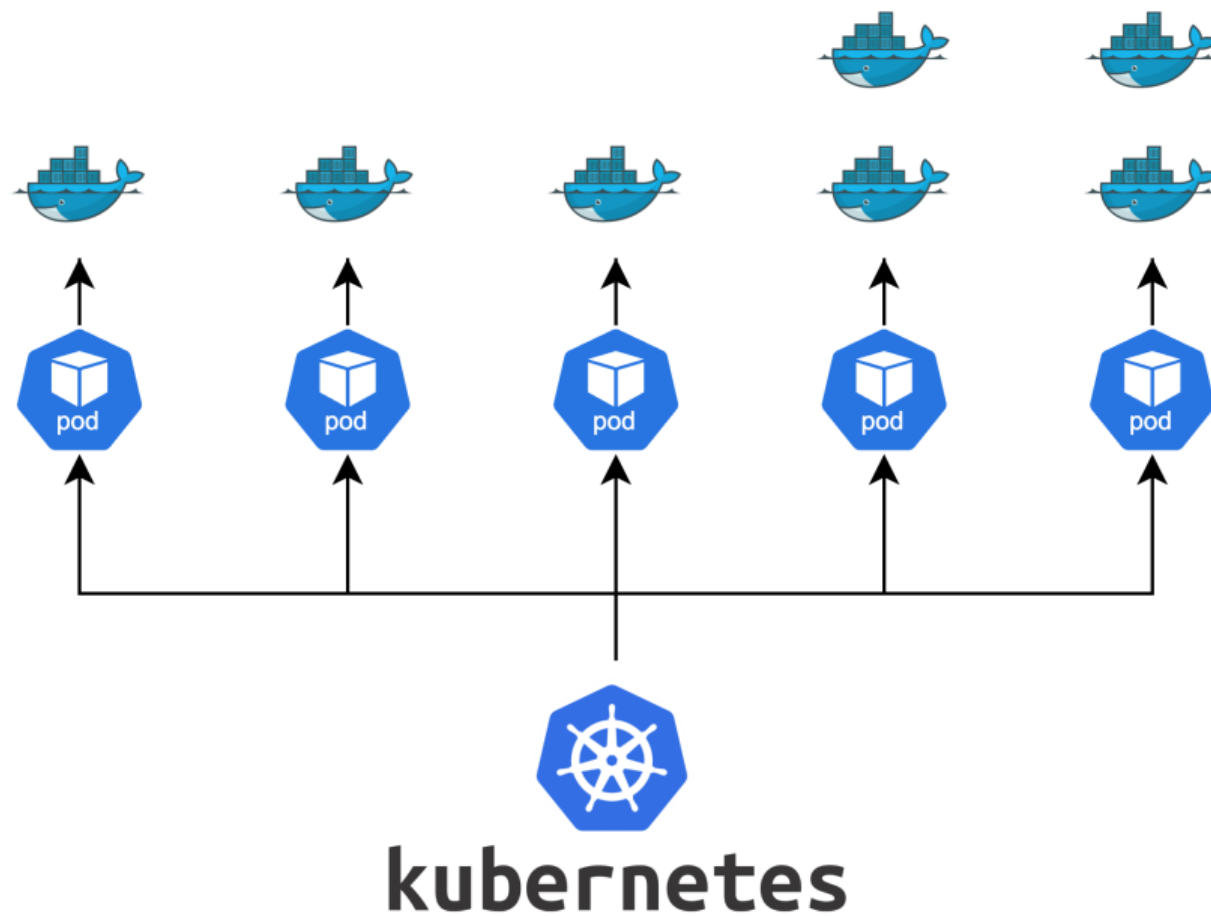
컨테이너를 자동으로 관리하기 위한 **오케스트레이션 도구** 필요



Session 2-4. 수 많은 컨테이너를 어떻게 관리하지?



컨테이너화 된 애플리케이션의 배포, 확장, 네트워킹(로드밸런싱, DNS 등)을 자동화하는 컨테이너 오케스트레이션 플랫폼



by Kim Jun Young (@yulmwu)

파드: 기본 단위, 1개 이상의 컨테이너 이미지를 가짐

노드: 파드를 실행하는데 필요한 컴퓨팅 리소스 단위 (워커노드)

클러스터: 여러 노드의 논리적인 집합

서비스: 파드들에 대해 안정적인 네트워크 접근을 제공함

Session 2-4. 수 많은 컨테이너를 어떻게 관리하지?

A **survey** of 689 IT professionals published this week by the Cloud Native Computing Foundation (CNCF) finds 80% of respondents work for IT organizations that have deployed Kubernetes in a production environment, with another 13% piloting or actively testing the platform.

출처: CNCF

" IT 기업 중 80% 정도가 쿠버네티스를 사용하고 있거나 사용했던 경험이 있음 "

Session 2-4. 수 많은 컨테이너를 어떻게 관리하지?



AWS Elastic Kubernetes Service (EKS)



Google Kubernetes Engine (GKE)



Azure Kubernetes Service (AKS)

클라우드에서 제공하는 쿠버네티스 환경

⇒ 설치/클러스터 관리 등 필요 없이 배포에만 신경 쓰면 됨



완전 관리형 서비스
(Fully-Managed)

Session 2-5. 이제는 한가지 기술만을 요구하는 시대가 아니다.



새로운 기능 만들었는데 배포하고 출시해주세요.

FE/BE 개발팀

안정성을 위해서 추가적인 테스트가 필요할 것 같은데요..



에이...
전 싫어요

인프라 운영팀



유저

신기능 나온다면서 언제나와요??????



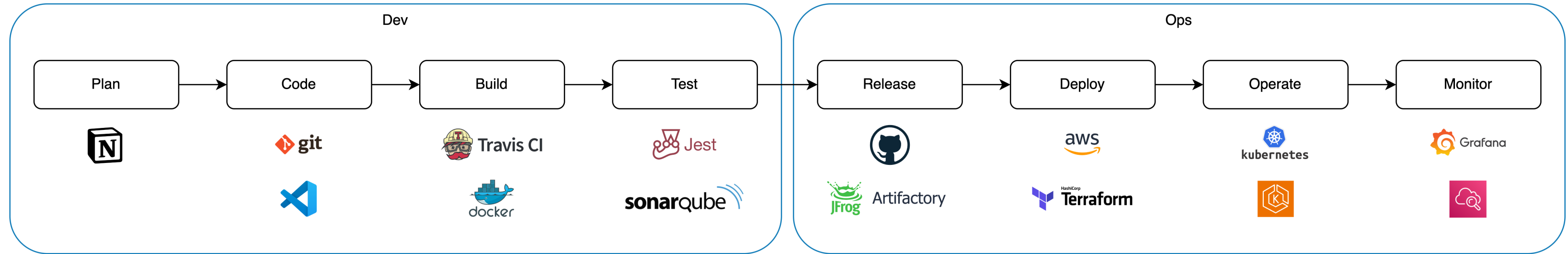
유저

이거 버그나는데요???

(서비스 종료)

Session 2-5. 이제는 한가지 기술만을 요구하는 시대가 아니다.

DevOps = Development(개발) + Operations(인프라 운영) + α



단순히 개발 + 인프라 운영 뿐만 아니라 CI/CD, 테스트, 모니터링, 보안 등 통합



장점: 빠른 개발부터 배포, 커뮤니케이션/협업 쉬워짐, 자동화 등

단점: 러닝 커브, 자동화 의존 등

Session 2-5. 이제는 한가지 기술만을 요구하는 시대가 아니다.

● kubernetes

검색어

● devops

검색어

+ 비교 추가

전 세계 ▼

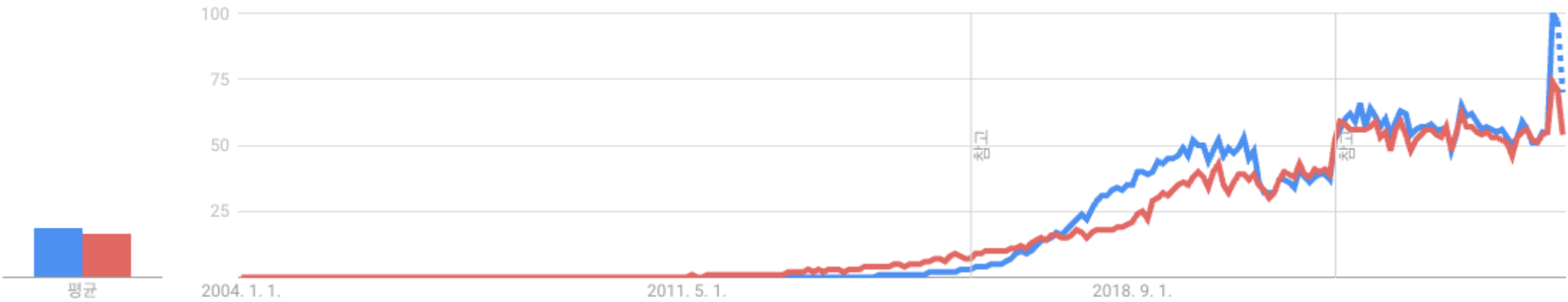
2004 - 현재 ▼

모든 카테고리 ▼

웹 검색 ▼

시간 흐름에 따른 관심도 변화 ⓘ

↓ <> 🔗



Session 03

 — 데이터베이스를 직접 구축하는 시대는 지났다.

Session 3 ToC

- | | | | |
|----|--|-------|---------------|
| 01 | ● 국가정보자원관리원 화재 사고의 G 드라이브, 왜 복구할 수 없었을까? | ————— | 서비스 분산의 중요성 |
| 02 | ● 데이터는 곧 기업의 생명이다. | ————— | SLA |
| 03 | ● 모든걸 AWS가 대신 해준다. | ————— | AWS 완전관리형 서비스 |

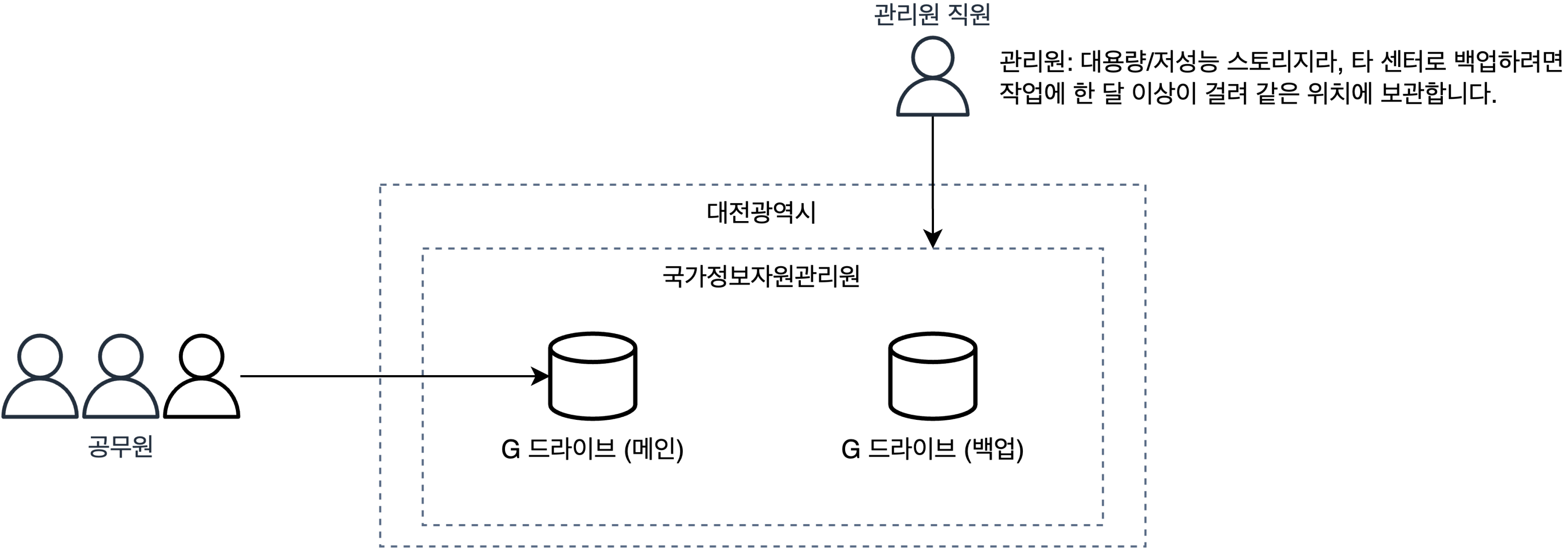
Session 3. 국가정보자원관리원 화재 사고의 G 드라이브, 왜 복구할 수 없었을까?



2025년 9월 26일에 발생하였던 국가정보관리원화재 사고

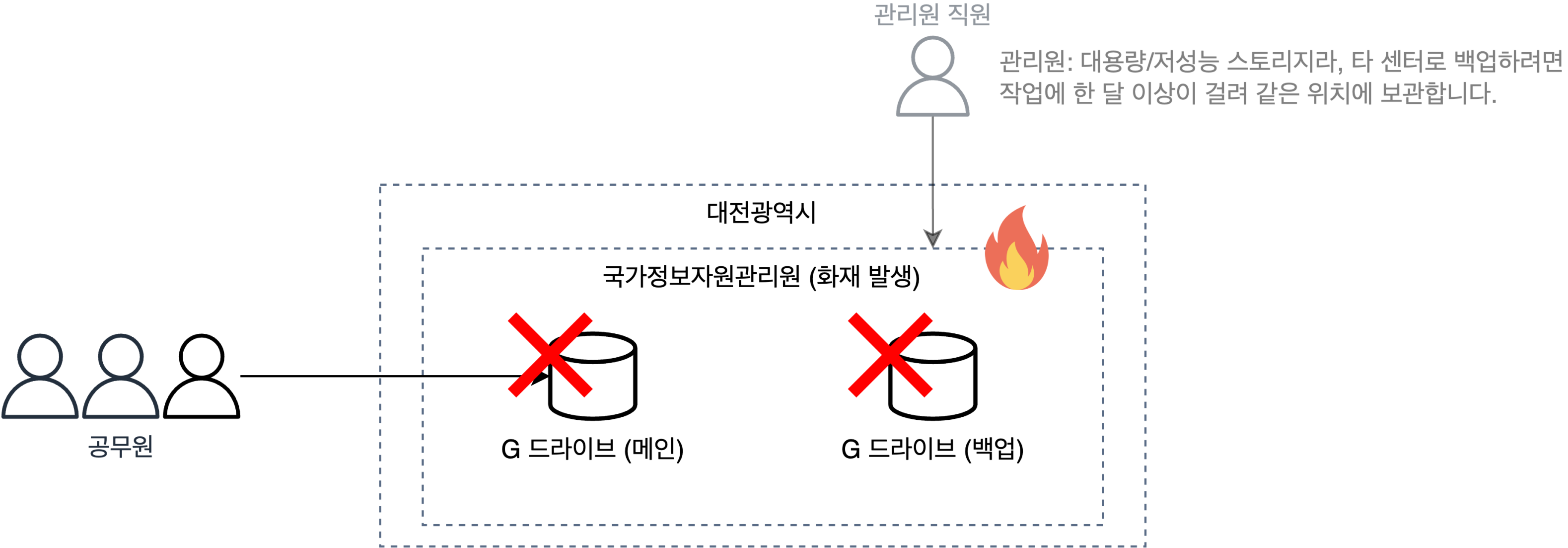
이때 공무원들이 사용하는 일명 "G드라이브"가 화재로 인해 전량 소실됨

Session 3. 국가정보자원관리원 화재 사고의 G 드라이브, 왜 복구할 수 없었을까?



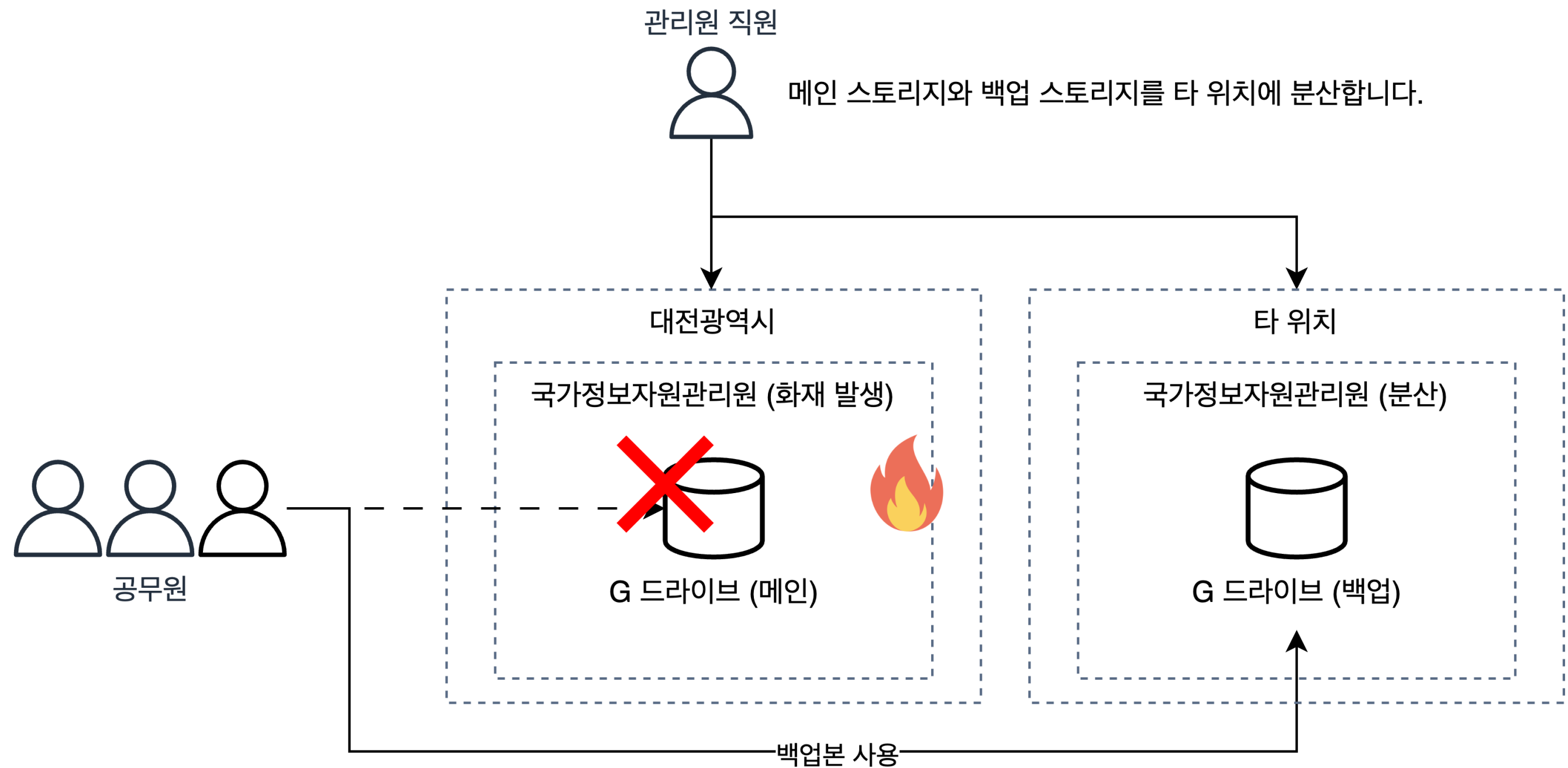
공무원들이 사용하는 G드라이브의 백업본(Replica)를 오래걸린다는 이유로 같은 위치에 보관하고 있었음

Session 3. 국가정보자원관리원 화재 사고의 G 드라이브, 왜 복구할 수 없었을까?



관리원 화재로 인해 메인으로 사용하는 스토리지(G드라이브)와 백업본 모두 소실된 상태
⇒ 만약 백업본이 다른 곳(위치)에 있었다면?

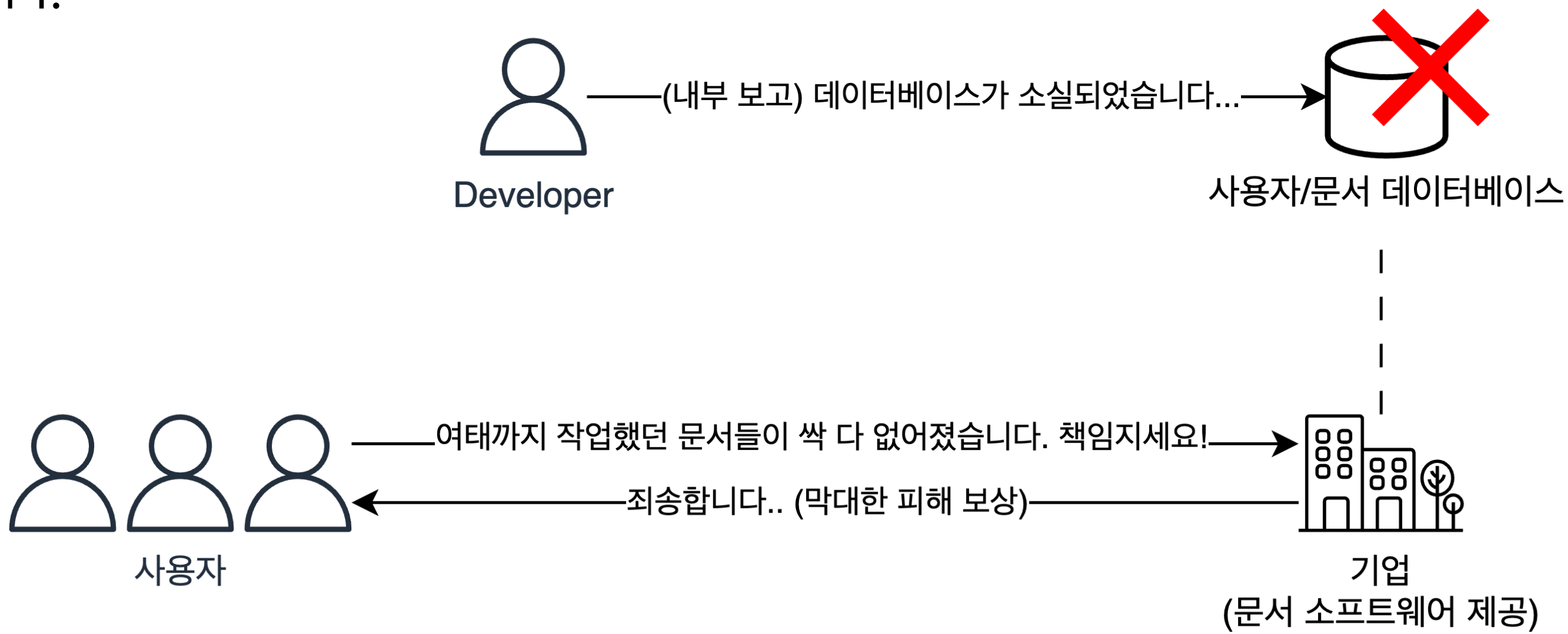
Session 3. 국가정보자원관리원 화재 사고의 G 드라이브, 왜 복구할 수 없었을까?



다른 위치에 보관했었다면 백업본을 사용하면서 복구가 가능했었음

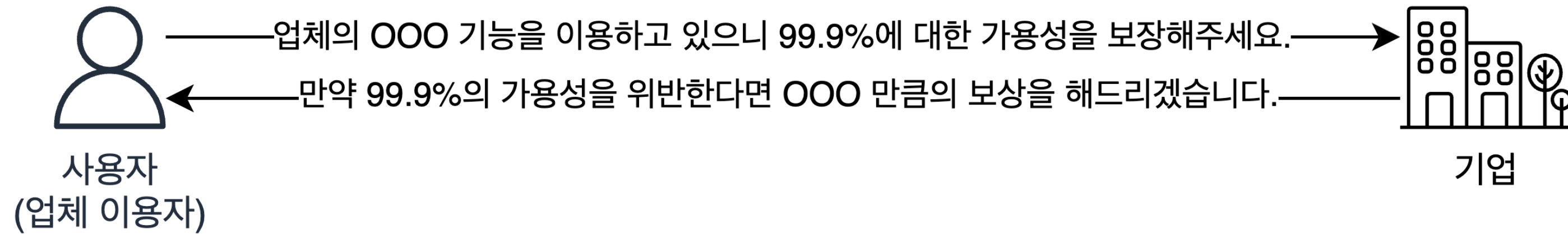
Session 3. 데이터는 곧 기업의 생명이다.

Fiction 1.



(실제로 이런 상황이 발생한다면 기업 자체가 한 순간에 없어짐)

Session 3. 데이터는 곧 기업의 생명이다.

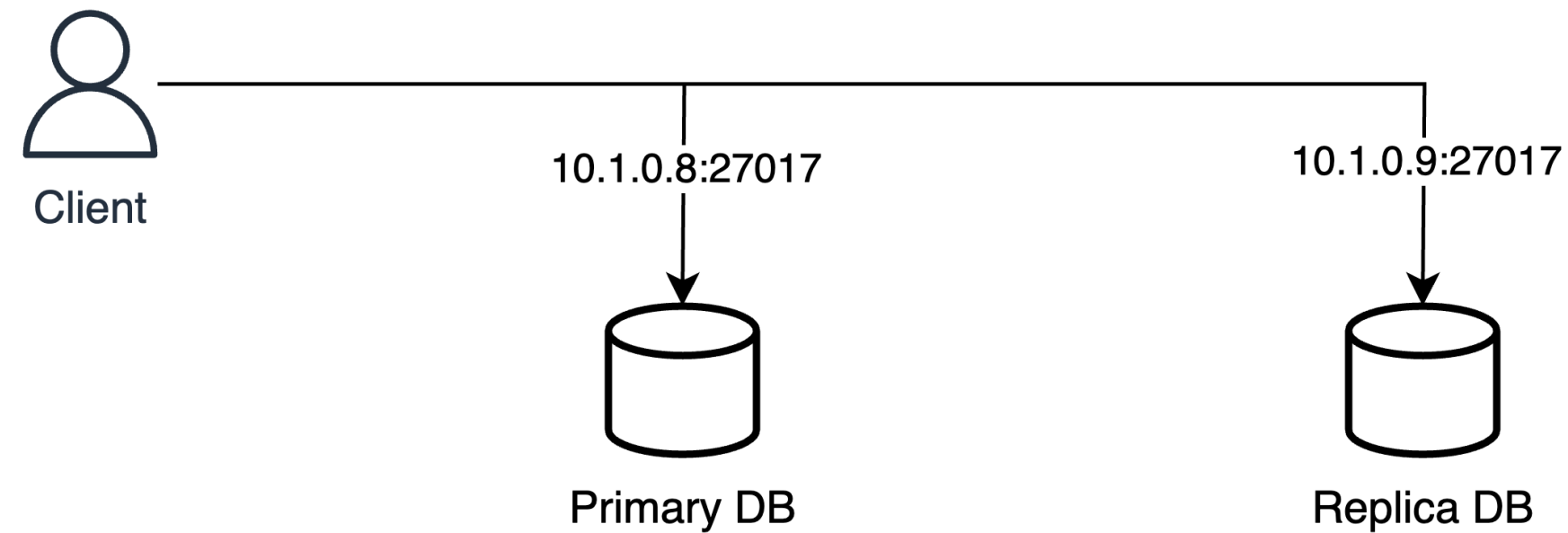


Fiction 1과 같은 상황을 최대한 방지하기 위해 클라우드/호스팅 업체와 이를 이용하는 기업(사용자) 간의 **SLA(서비스 수준 계약, Service Level Agreement)**을 계약함. (아웃 소싱)

예시) 계약된 가용성 %를 보장하기 위해 데이터를 분산하여 저장

Session 3. 모든걸 AWS가 대신 해준다.

그럼 어떻게 데이터(스토리지, DB)를 분산하여 저장할까?



분산된 DB에 접근하기 위해 각 DB의 IP(엔드포인트나 진입점)를 따로 사용, 유지보수 하기 매우 어려움

이런 작업을 자동으로 해주는 건 없을까?

Session 3. 모든걸 AWS가 대신 해준다.



S3



DynamoDB
(NoSQL, HTTP Based)



RDS
(Aurora, MySQL, PostgreSQL ...)



ElastiCache
(In Memory, Redis ...)



EFS



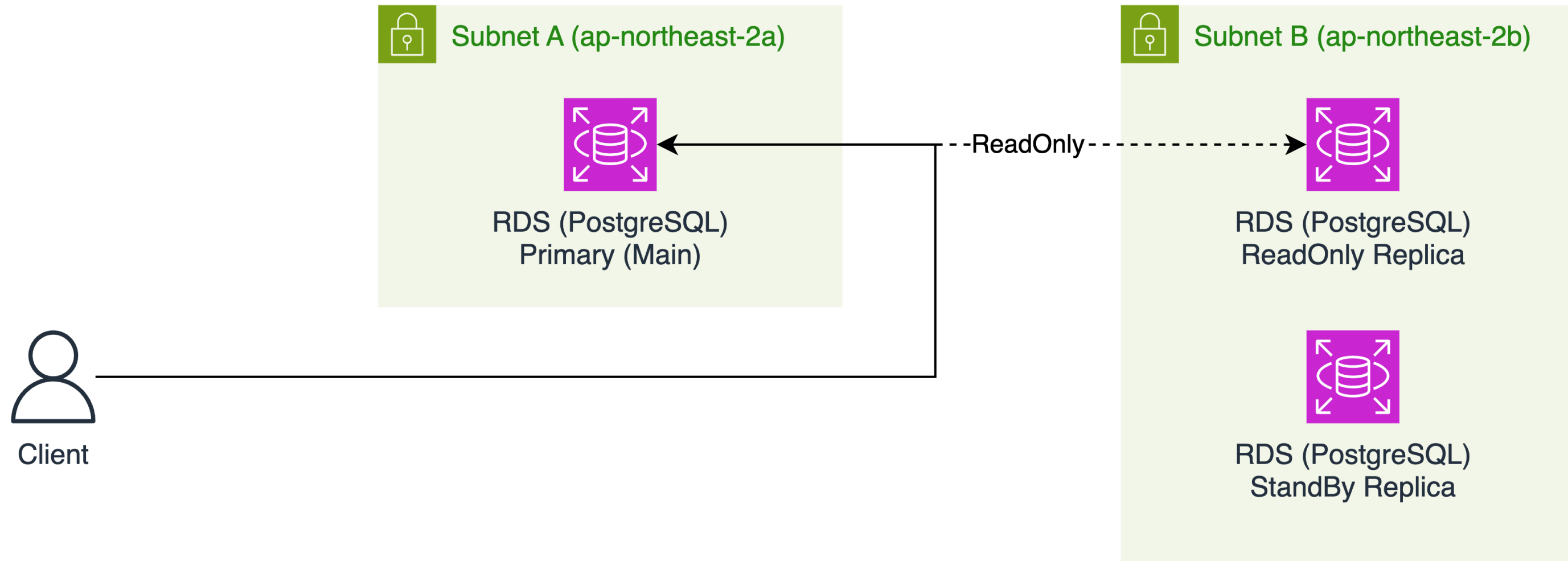
DocumentDB
(MongoDB)

등등..

위 서비스들은 스토리지/데이터베이스 운영을 AWS에서 담당 (**AWS 완전 관리형 서비스**)

그 중 다른 AZ(지역)에 자동으로 분산(백업이나 ReadOnly 등의 Replica) 지원

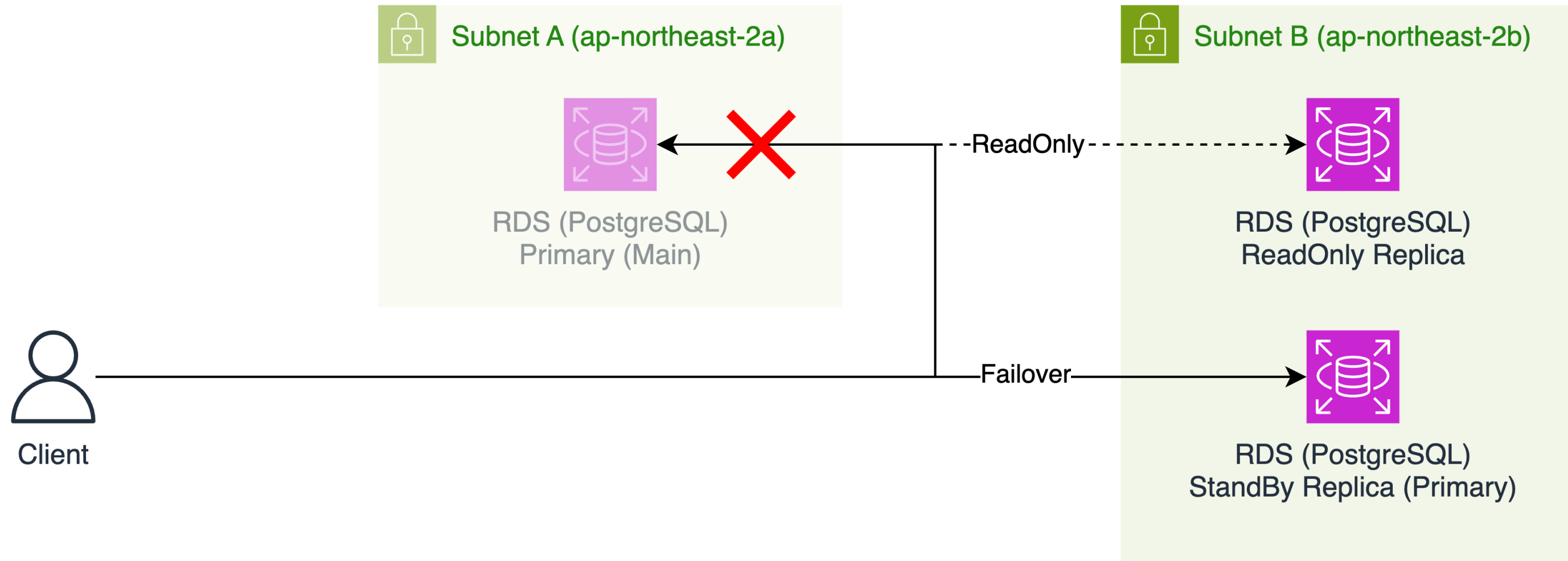
Session 3. 모든걸 AWS가 대신 해준다.



평상시엔 Read/Write 모두 가능한 메인 DB, 성능 분산을 위해 빠른 성능이 필요 없을 경우 ReadOnly DB 접근

(Main의 데이터를 Replica에 분산하는데 시간이 걸리기 때문에 바로 업데이트 되지 않음)

Session 3. 모든걸 AWS가 대신 해준다.



만약 서브넷 A (ap-northeast-2a AZ)에 문제가 생겼을 경우 StandBy Replica로 Primary 승격(Failover)
이 과정(DB 프로비저닝, Fallback/Failover 처리, 분산 처리 등)을 AWS가 완전 관리형으로 자동화

Session 3. 모든걸 AWS가 대신 해준다.

를 이용하면 데이터베이스를 직접 설정하고 수동으로 분산/Fallback 처리를 직접 구현할 필요가 없음

+ AWS 내부적으로 SLA 보장

Aurora and RDS > 대시보드

Aurora and RDS

대시보드

데이터베이스

쿼리 편집기

성능 개선 도우미

스냅샷

Amazon S3에서 내보내기

자동 백업

예약 인스턴스

프록시

서브넷 그룹

파라미터 그룹

옵션 그룹

사용자 지정 엔진 버전

제로 ETL 통합

이벤트

이벤트 구독

권장 사항 0

인증서 업데이트

리소스

새로 고침

Asia Pacific (Seoul) 리전에서 다음의 Amazon RDS 리소스를 사용하고 있습니다(사용량/할당량).

DB 인스턴스 (0/40)

파라미터 그룹 (3)

할당된 스토리지 (0 TB/100 TB)

기본값 (3)

인스턴스와 스토리지에는 Neptune과 DocumentDB가 포함됩니다.DB 인스턴스 제한 증가

사용자 지정 (0/100)

DB 클러스터 (0/40)

옵션 그룹 (1)

예약 인스턴스 (0/40)

기본값 (1)

스냅샷 (0)

사용자 지정 (0/20)

수동

서브넷 그룹 (2/50)

DB 클러스터 (0/100)

지원되는 플랫폼 VPC

DB 인스턴스 (0/100)

기본 네트워크 none

자동

자동

DB 클러스터 (0)

DB 인스턴스 (0)

최근 이벤트 (0)

이벤트 구독 (0/20)

데이터베이스 생성

Amazon Relational Database Service(RDS)를 사용하면 클라우드에서 관계형 데이터베이스를 쉽게 설정, 운영 및 조정할 수 있습니다.

Amazon S3의 백업을 사용하여 새 Aurora MySQL 및 MySQL 데이터베이스를 복원하고 생성할 수 있습니다.

데이터베이스 생성

S3에서 복원

참고: DB 인스턴스가 Asia Pacific (Seoul) 리전에서 시작됩니다.

Aurora 및 RDS 살펴보기

이 활동에서는 데이터베이스를 생성하는 방법을 배우게 됩니다. 자습서 시작을 선택하여 시작하세요.

예상 시간 2~5분

자습서 시작

권장 서비스

사용량 패턴이 유사한 AWS 고객을 기반으로, 사용자의 워크플로에 적합한 기타 서비스

EFS

EC2용 관리형 파일 스토리지

Amazon Connect

Amazon Connect는 규모와 상관없이 고객 참여를 지원하는 고객 센터입니다.

Amazon AppFlow

Amazon AppFlow는 앱을 통합하고 코드 없이 데이터 흐름을 자동화합니다.

AWS Firewall Manager

방화벽 규칙의 중앙 관리

AWS Well-Architected Tool

AWS Well-Architected Tool을 사용하여 모범 사례를 배우고 워크로드의 상태를 측정 및 개선

간단한 구성으로 데이터베이스 구성 및 On Demand 요금제로 사용한 만큼만 요금 지불

Session 3. 모든걸 AWS가 대신 해준다.

다만 주의할 점, 데이터베이스는 한번 구축하고 서비스를 하게 될 경우

다른 인프라, 또는 DBMS로 마이그레이션하기 매우 어려운 주제임. (예: 온 프레미스 → 클라우드)

(기술적, 비용적, 법적 문제 등등..)

(컴퓨팅 인프라는 클라우드로 마이그레이션 했지만 DB만 기존의 온 프레미스 환경에서 레거시하게 운영하는 기업도 있음)

때문에 초기 인프라 설계를 효율적으로, 상황에 맞게 설계하고 향후까지 생각하는 것이 중요

Session 04



Operator 패턴 — Kubernetes는 어떻게 단순 배포 도구를 넘어,
클라우드 네이티브 생태계를 바꿀 수 있었을까? (*Feat. Kubernetes의 설계 철학*)

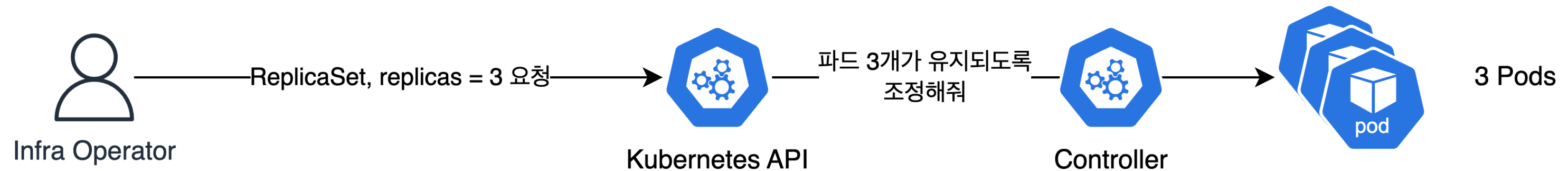
Session 4 ToC

01	• 쿠버네티스는 어떻게 컨테이너(또는 파드)를 오케스트레이션 할까?	Controller
02	• 더 이상 복잡한 인프라를 수동으로 관리 할 필요가 없다.	Operator

Session 4. 쿠버네티스는 어떻게 컨테이너(또는 파드)를 오케스트레이션 할까?

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 3
```

선언형(Declarative)

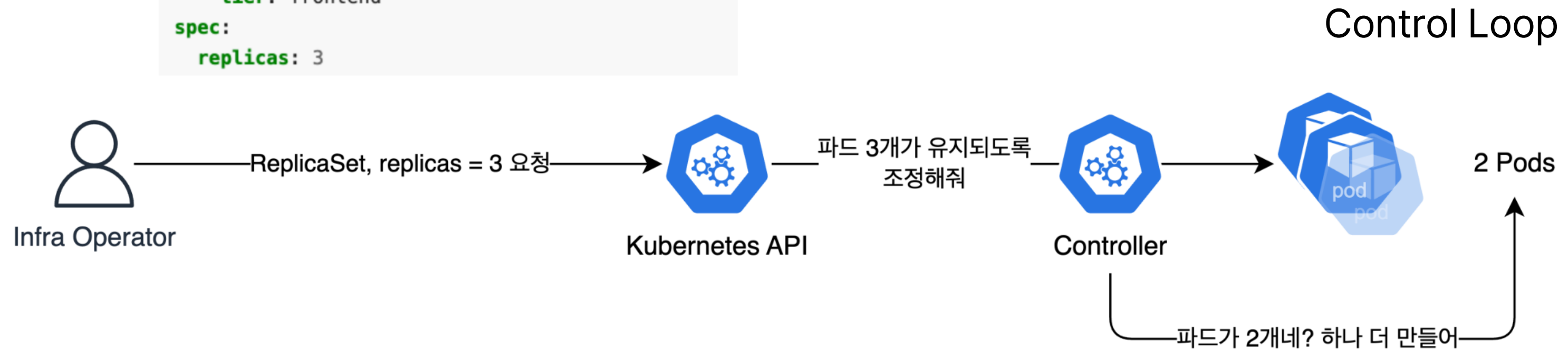


ReplicaSet 오브젝트(파드 복제) 상태 선언, Kubernetes API를 통해 구성 및 Controller가 컨트롤

만약 파드가 2개가 되어 선언된 상태와 맞지 않을 경우?

Session 4. 쿠버네티스는 어떻게 컨테이너(또는 파드)를 오케스트레이션 할까?

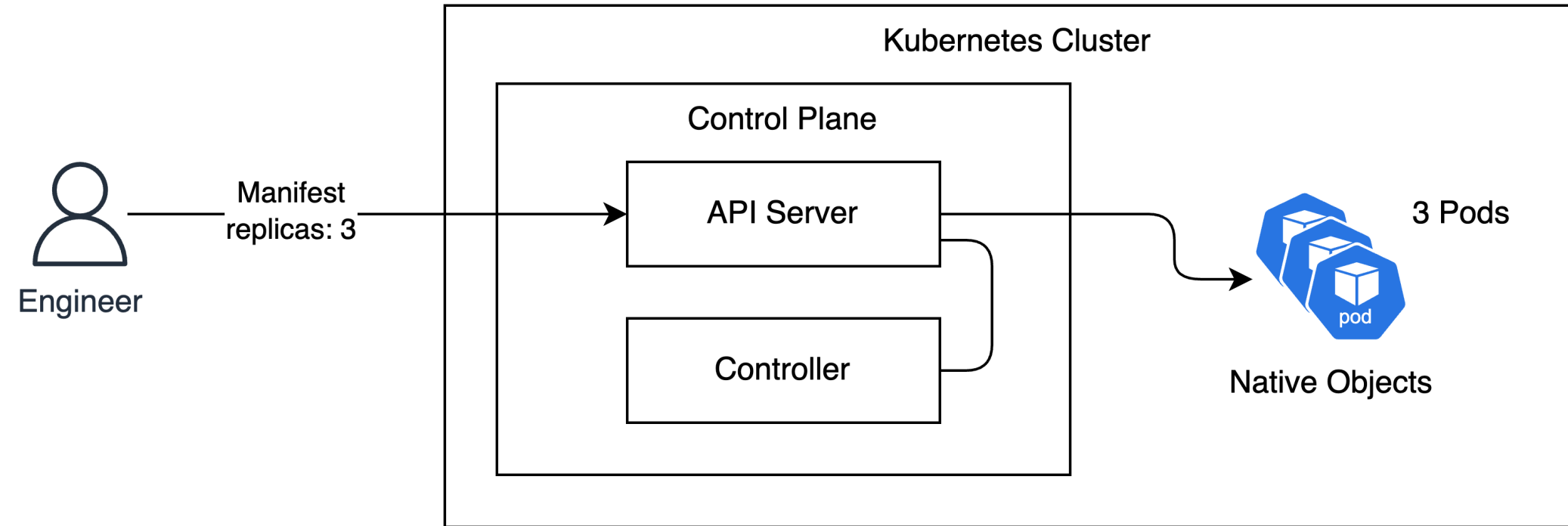
```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 3
```



⇒ Controller가 다시 파드가 3개가 되도록 조정(Reconcile) 한다.

" Controller Pattern "

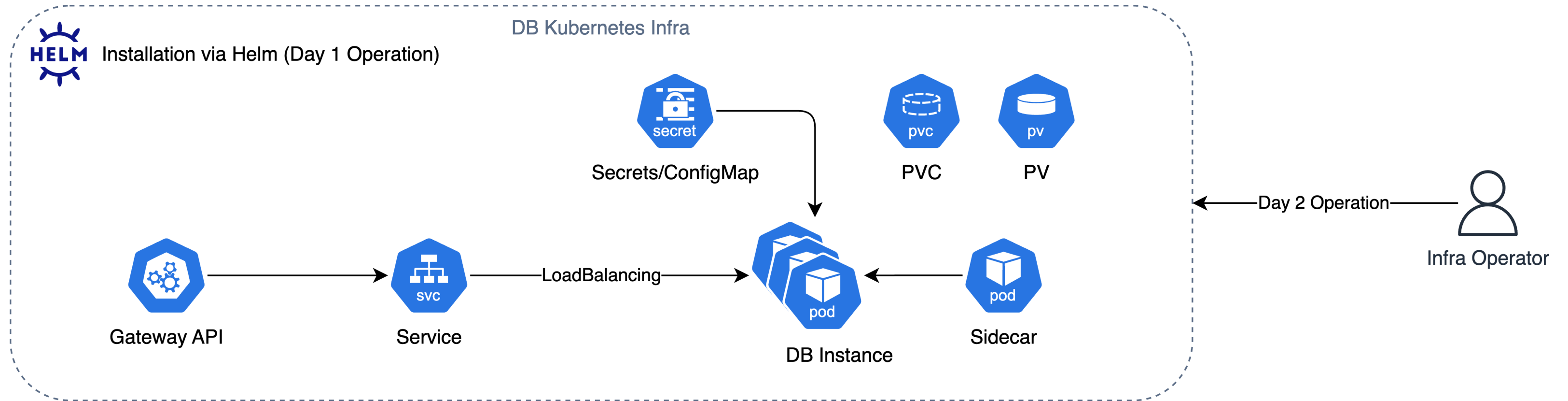
Session 4. 쿠버네티스는 어떻게 컨테이너(또는 파드)를 오케스트레이션 할까?



네이티브 오브젝트(Pod, Deployment, Service, Ingress 등)로만 구성하면 문제가 없음

하지만 복잡한 상태의 Stateful 애플리케이션이나, 설정이 자주 변경되어
ConfigMap 등을 하나하나 수정 후 Deployment/파드를 수동으로 재시작을 해야 할 경우? ⇒ 네이티브 오브젝트로만
운영하기엔 어려움

Session 4. 쿠버네티스는 어떻게 컨테이너(또는 파드)를 오케스트레이션 할까?

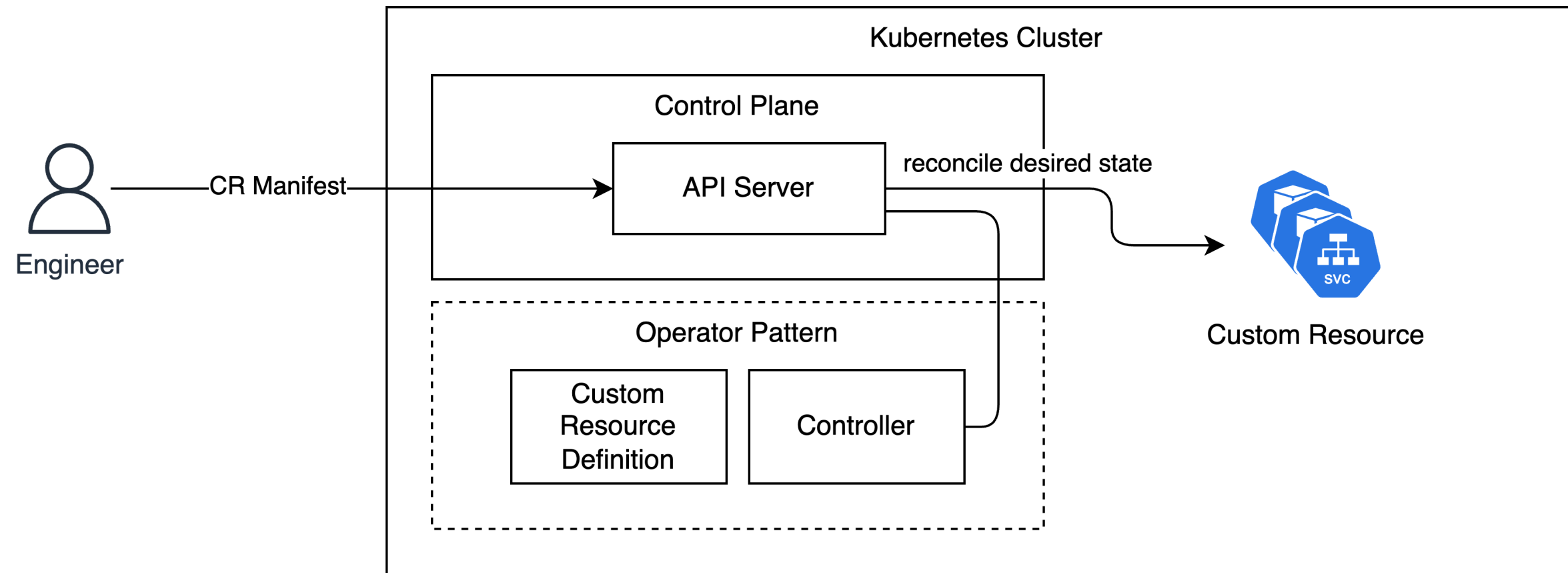


(물론 실제로 쿠버네티스 등으로 DB를 배포하는 경우는 거의 없음)

Day 1 Operation(설계, 배포)는 Helm, Kustomize 등을 통해 가능
하지만 **Day 2 Operation**(유지보수, 모니터링, 장애 대응=Failover 등의 실제 운영)는 인프라 운영자가 담당

" 이를 자동화할 순 없을까? "

Session 4. 더 이상 복잡한 인프라를 수동으로 관리 할 필요가 없다.

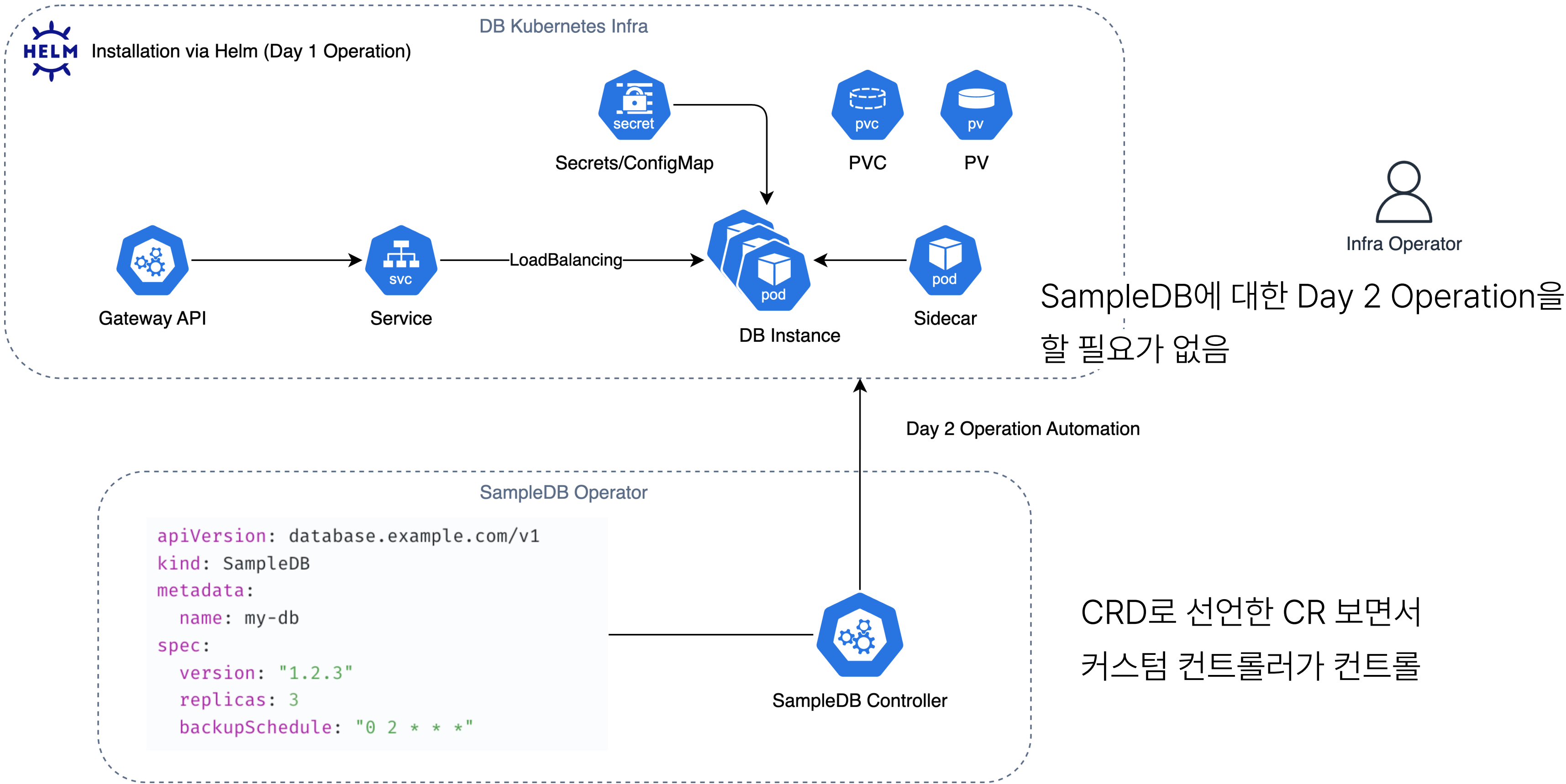


Operator — 운영자(Operator)의 **Operational Knowledge**를 컨트롤러 자체로 옮기자

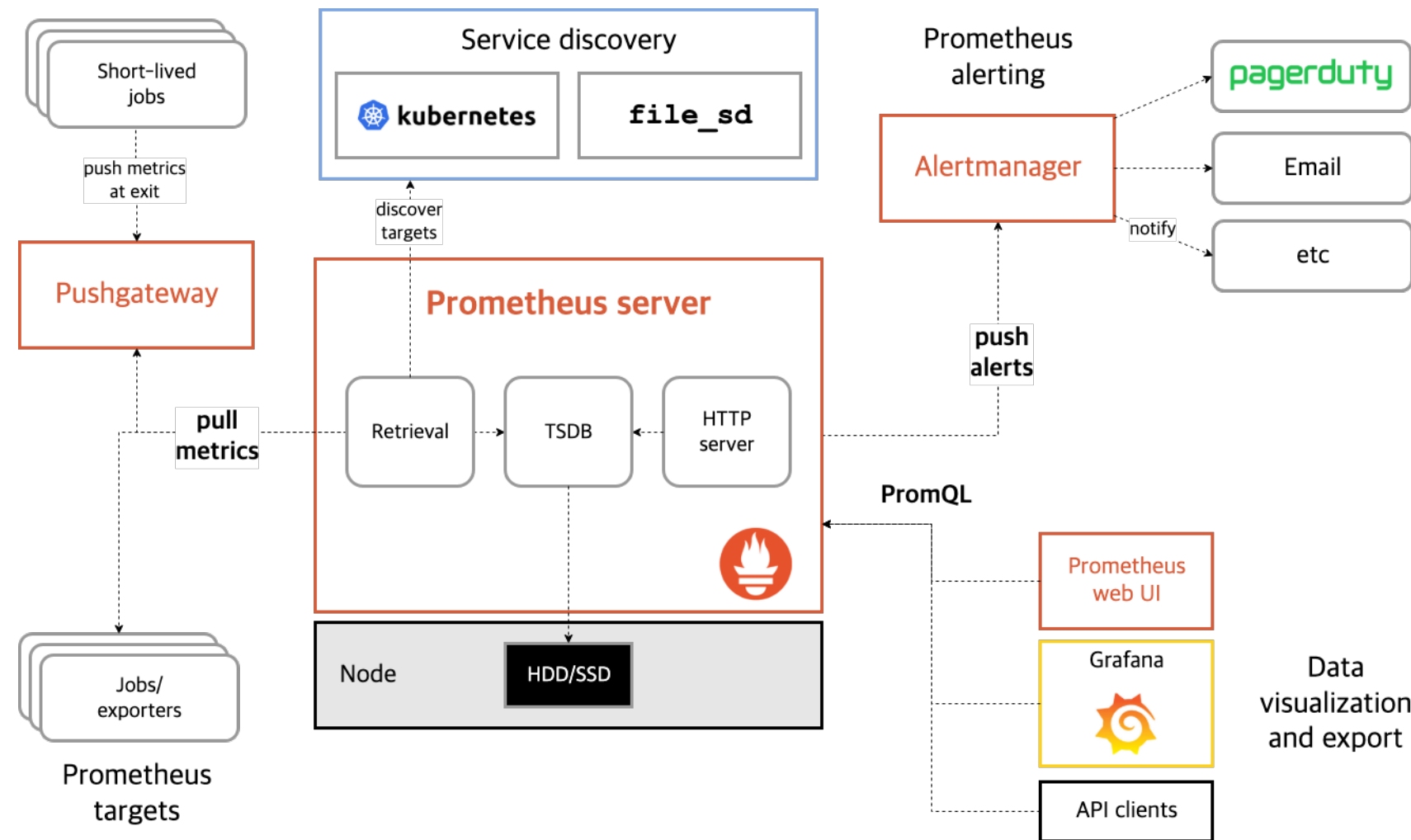
⇒ **CRD**(Custom Resource Definition), **커스텀 컨트롤러**

Operational Knowledge = 운영 시 필요한 메뉴얼이나 노하우, 장애 발생 시 Failover, 백업/복구, 버전 업그레이드 등의 방법론 등

Session 4. 더 이상 복잡한 인프라를 수동으로 관리 할 필요가 없다.



Session 4. 더 이상 복잡한 인프라를 수동으로 관리 할 필요가 없다.



출처: Prometheus

모니터링 시스템 Prometheus,
내부적으로 매우 복잡한 구조를 가졌음

Helm을 통해 설치하는 가능하지만
설정 변경 시 수동으로 다시 구성해야함

⇒ Operator를 지원하기 때문에 Helm을 통해 Operator만 설치하면 CRD로 쉬운 구성 가능
+ 거기에 맞게 자동으로 컨트롤러가 Reconile(조정)

Session 4. 더 이상 복잡한 인프라를 수동으로 관리 할 필요가 없다.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

```
helm install kps prometheus-community/kube-prometheus-stack \
  -n monitoring -f kps-values.yaml
```

```
•> kubectl get all -n monitoring
```

NAME	READY	STATUS	RESTARTS	AGE
pod/alertmanager-kps-kube-prometheus-stack-alertmanager-0	2/2	Running	0	59s
pod/kps-grafana-5bdb744bf4-k2djg	3/3	Running	0	61s
pod/kps-kube-prometheus-stack-operator-68cdddc4c7-lrpqs	1/1	Running	0	61s
pod/kps-kube-state-metrics-85c65fb7b6-25fdz	1/1	Running	0	61s
pod/kps-prometheus-node-exporter-qfj7w	1/1	Running	0	61s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/alertmanager-operated	ClusterIP	None	<none>	9093/TCP,9094/TCP,9094/UDP	59s
service/kps-grafana	ClusterIP	10.100.252.222	<none>	80/TCP	61s
service/kps-kube-prometheus-stack-alertmanager	ClusterIP	10.96.141.214	<none>	9093/TCP,8080/TCP	61s
service/kps-kube-prometheus-stack-operator	ClusterIP	10.98.96.73	<none>	443/TCP	61s
service/kps-kube-prometheus-stack-prometheus	ClusterIP	10.104.44.168	<none>	9090/TCP,8080/TCP	61s
service/kps-kube-state-metrics	ClusterIP	10.107.255.103	<none>	8080/TCP	61s
service/kps-prometheus-node-exporter	ClusterIP	10.105.133.207	<none>	9100/TCP	61s

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
daemonset.apps/kps-prometheus-node-exporter	1	1	1	1	1	kubernetes.io/os=linux	61s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/kps-grafana	1/1	1	1	61s
deployment.apps/kps-kube-prometheus-stack-operator	1/1	1	1	61s
deployment.apps/kps-kube-state-metrics	1/1	1	1	61s

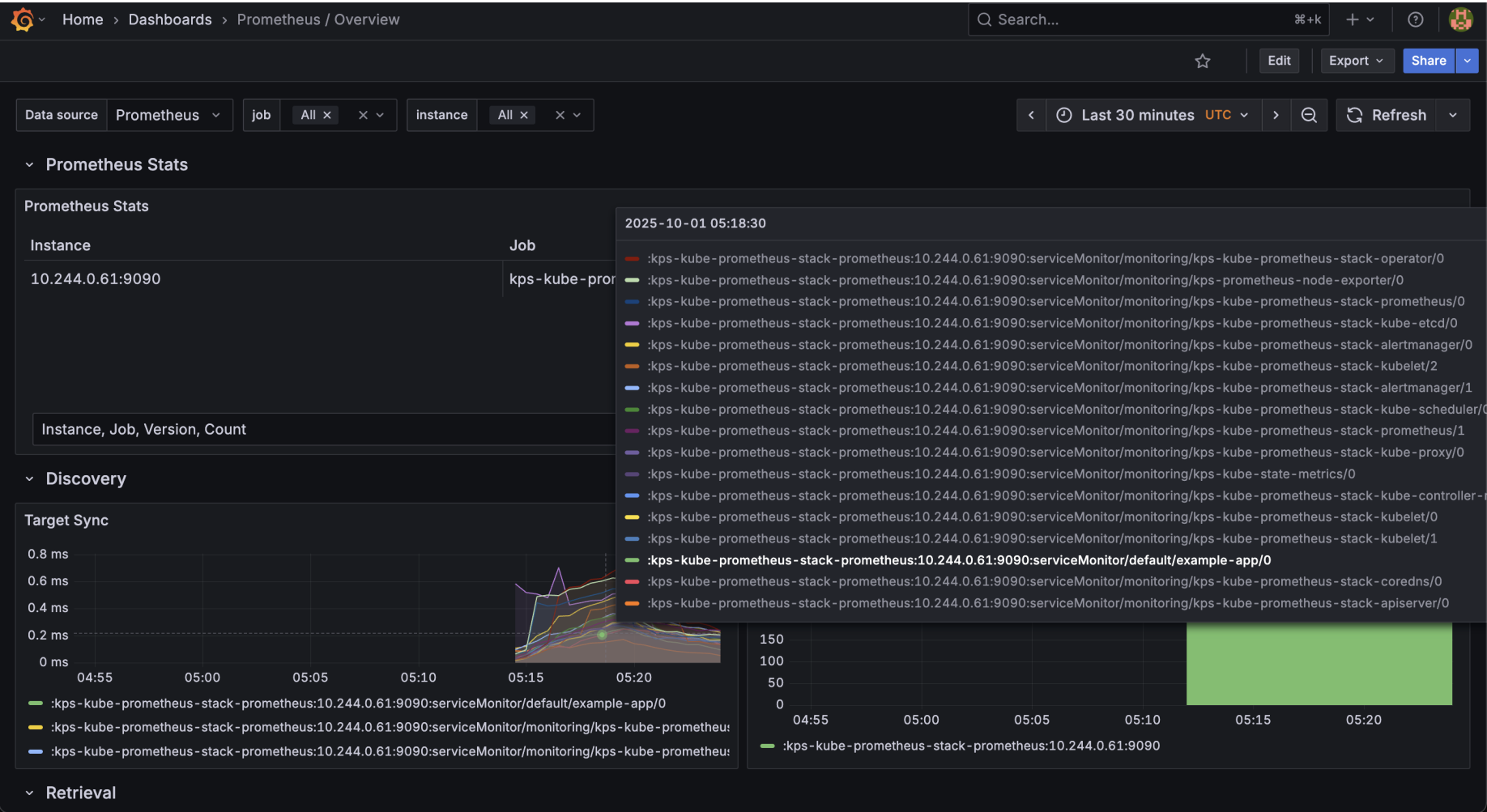
NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/kps-grafana-5bdb744bf4	1	1	1	61s
replicaset.apps/kps-kube-prometheus-stack-operator-68cdddc4c7	1	1	1	61s
replicaset.apps/kps-kube-state-metrics-85c65fb7b6	1	1	1	61s

NAME	READY	AGE
statefulset.apps/alertmanager-kps-kube-prometheus-stack-alertmanager	1/1	59s

(Prometheus 설치 및 Operator 구성 끝)

Session 4. 더 이상 복잡한 인프라를 수동으로 관리 할 필요가 없다.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: example-app
  namespace: default
  labels:
    release: kps
    app: example-app
spec:
  selector:
    matchLabels:
      app: example-app
  namespaceSelector:
    matchNames: ["default"]
  endpoints:
    - port: http
      interval: 30s
      path: /metrics
```



특정 Service에 대한 Service Monitor CRD 적용

Prometheus 컨트롤러를 통한 Service Monitor 적용 자동화

⇒ Kubernetes Operator가 없었다면 전부 수동으로 했어야할 작업 (운영 시 비효율적)

Session 05

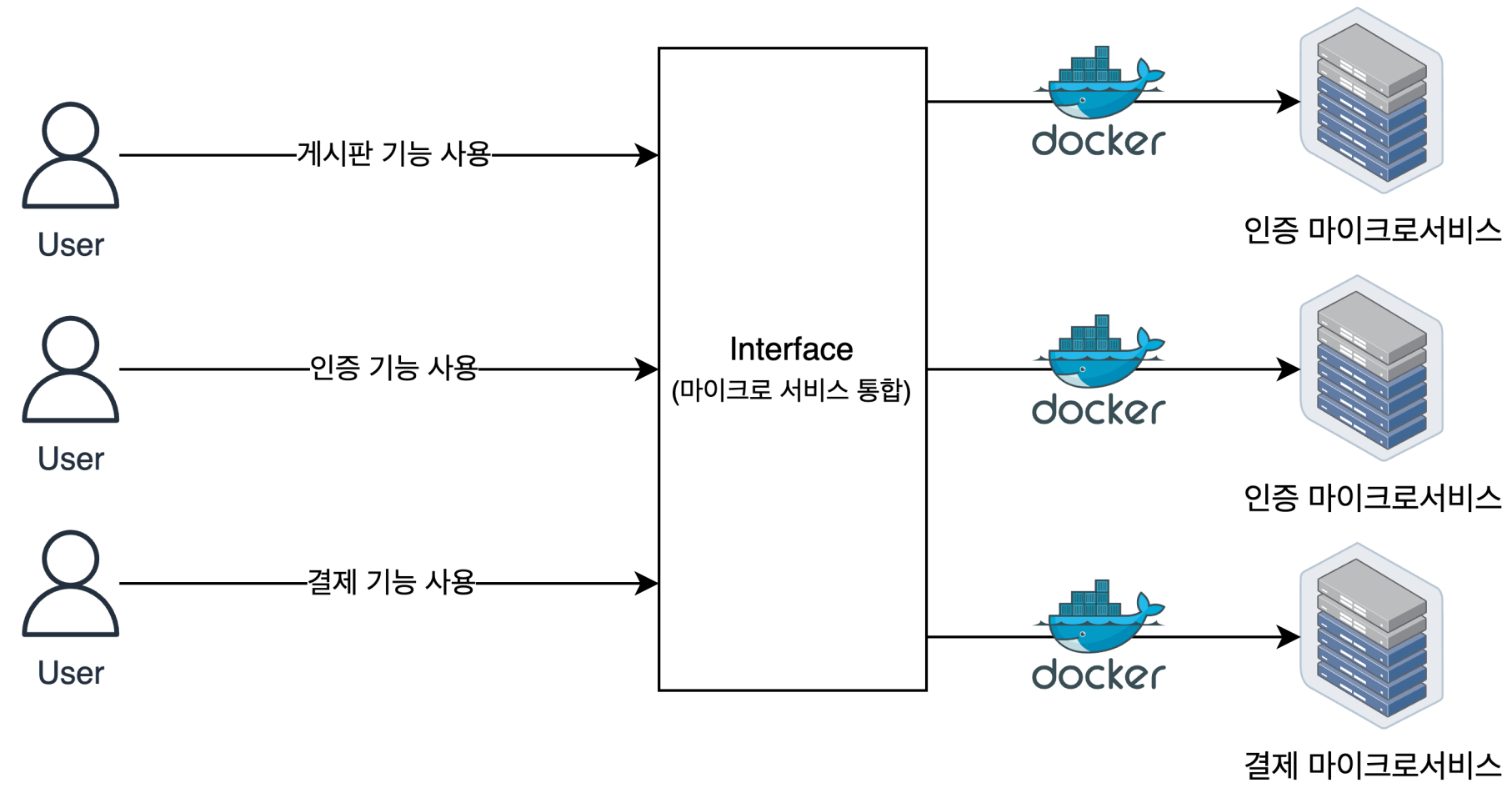


Istio — Kubernetes 서비스 메시의 핵심

Session 5 ToC

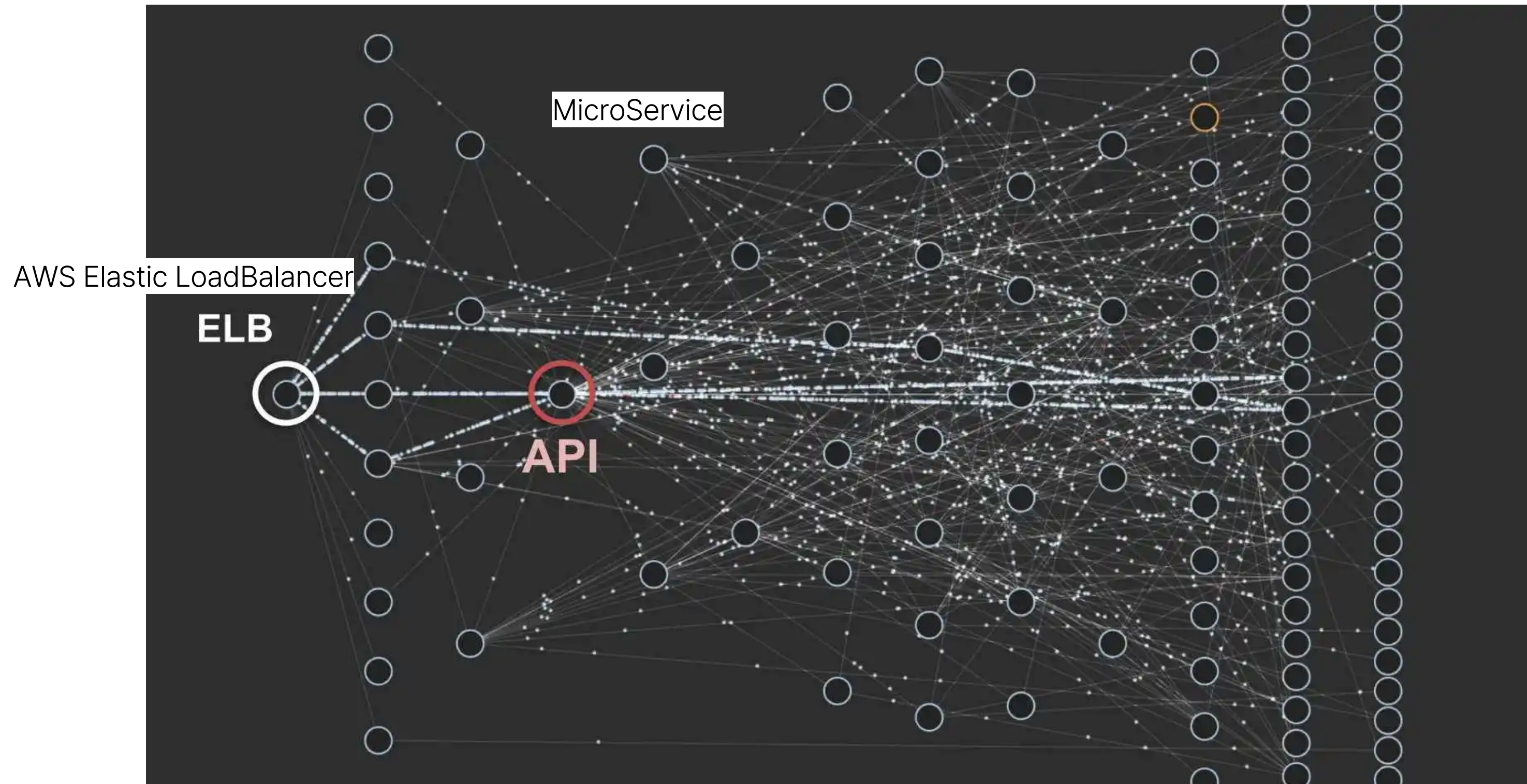
- | | | | |
|----|-------------------------------|-------|-------------|
| 01 | • MSA는 생각 이상으로 복잡하다. | ————— | 서비스 메시의 필요성 |
| 02 | • 우리의 트래픽은 과연 안전하게 목적지로 도착할까? | ————— | TLS와 mTLS |
| 03 | • Istio를 통한 서비스 메시 구성하기 | | |

Session 5. MSA는 생각 이상으로 복잡하다.



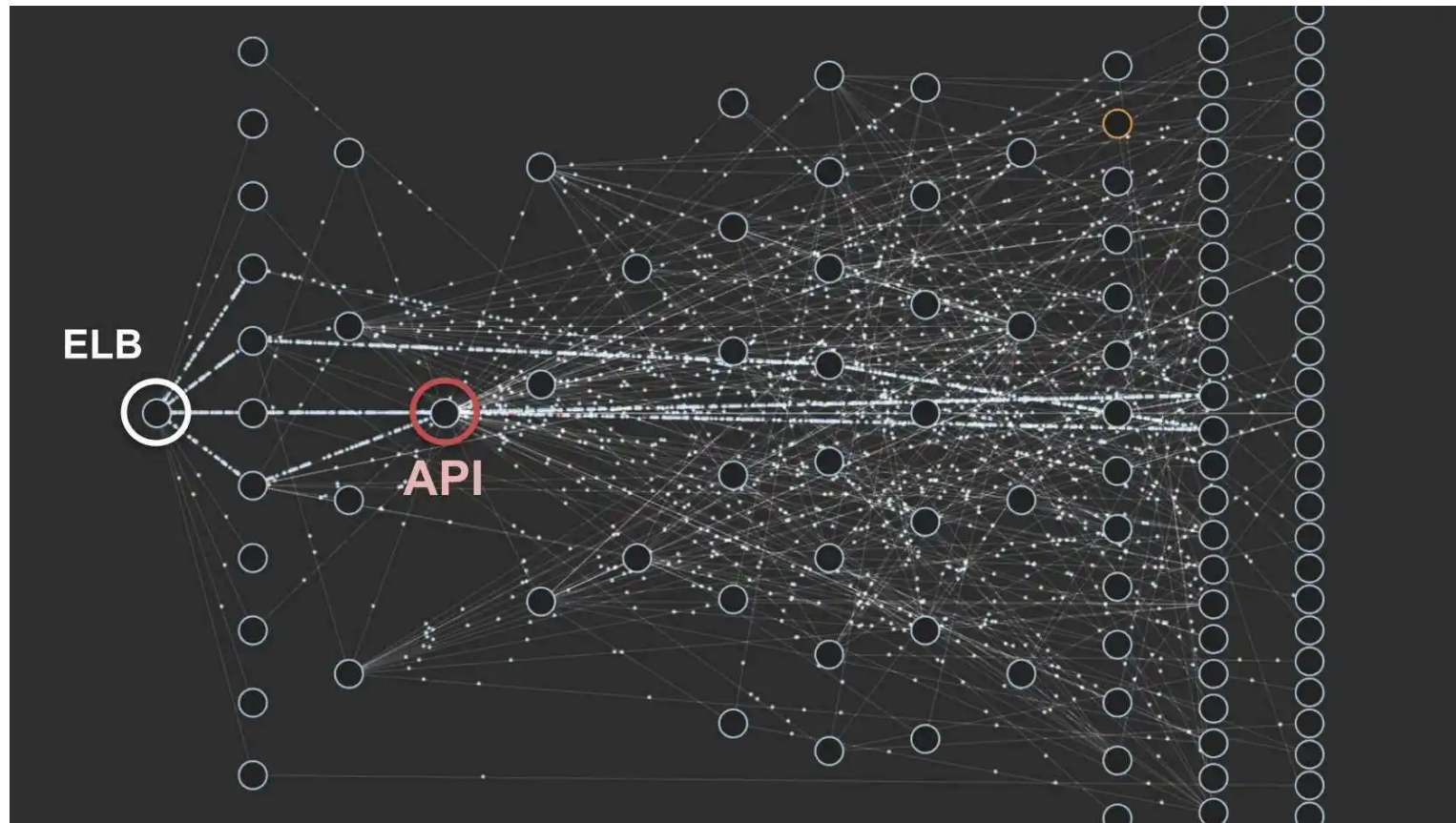
MicroService Architecture, 실제로 이렇게 간단할까?

Session 5. MSA는 생각 이상으로 복잡하다.



출처: InfoQ 2017 - A Netflix Guide to Microservices

Session 5. MSA는 생각 이상으로 복잡하다.



MSA 구조에서 서비스 간의 통신을 관리하기 위한 인프라 계층

" Service Mesh "

복잡한 마이크로 서비스를 서비스 메시를 통해 제어하고
모니터링이나 보안 구성, 서비스 디스커버리 및 로드밸런싱 가능

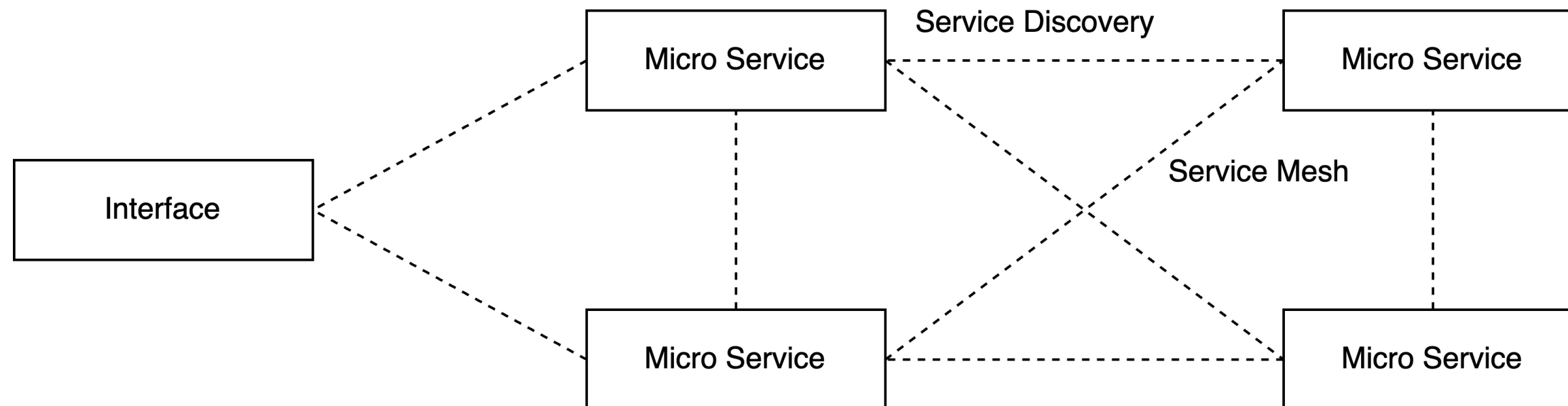
Session 5. MSA는 생각 이상으로 복잡하다.

쿠버네티스 = MSA 구조에 특화된 컨테이너 오케스트레이션 플랫폼

" 쿠버네티스 내부적으로 서비스 메시를 지원하지 않나요? "

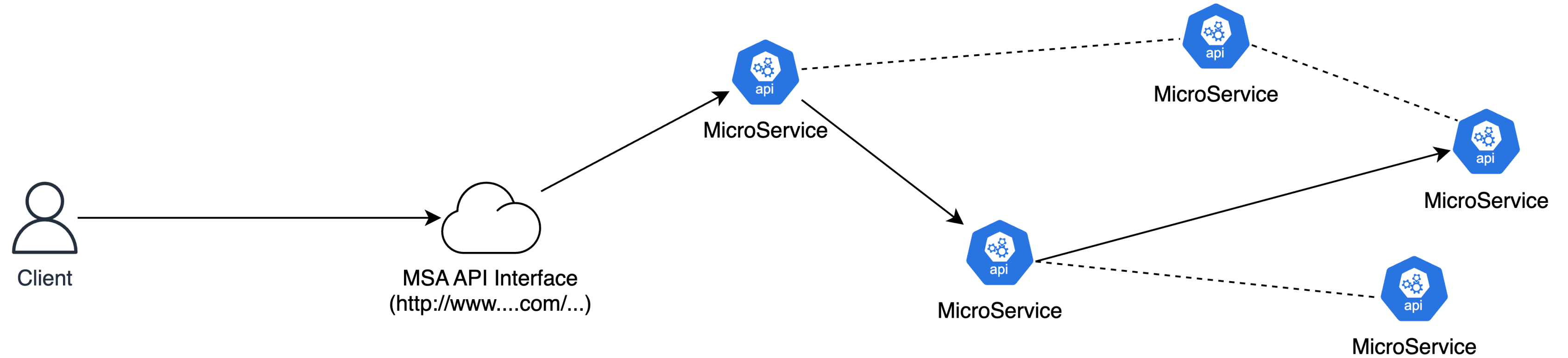
CoreDNS, kube-proxy, NetworkPolicy 등의 기능으로 기본적인 통신 및 디스커버리는 가능

⇒ 다만 L4 네트워킹 중심에 세부적인 mTLS 적용, 모니터링(Observability), AuthZ/AuthN 또는 L7(HTTP/HTTPS) 레벨의 트래픽을 세부적으로 다루기엔 어려움



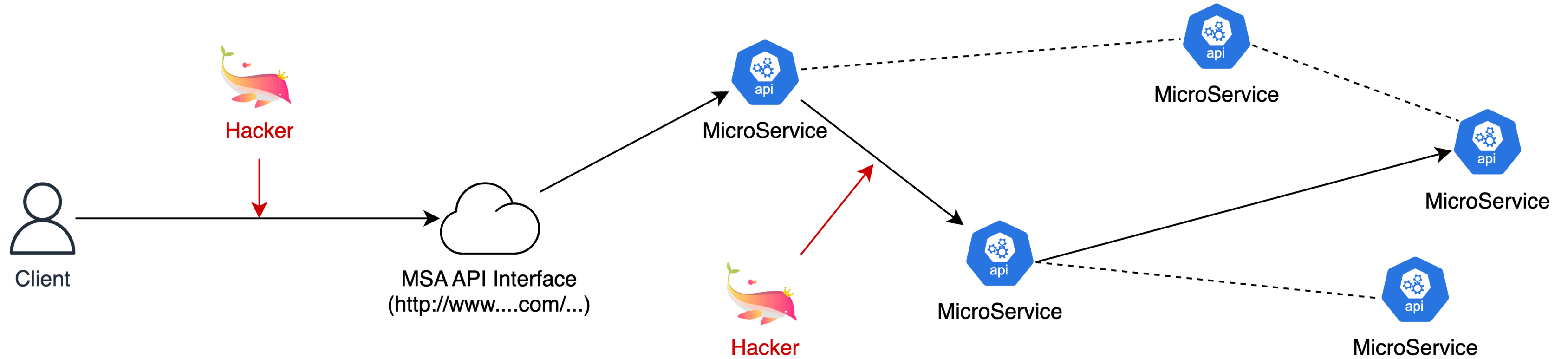
TLS, mTLS ?

Session 5. 우리의 트래픽은 과연 안전하게 목적지로 도착할까?



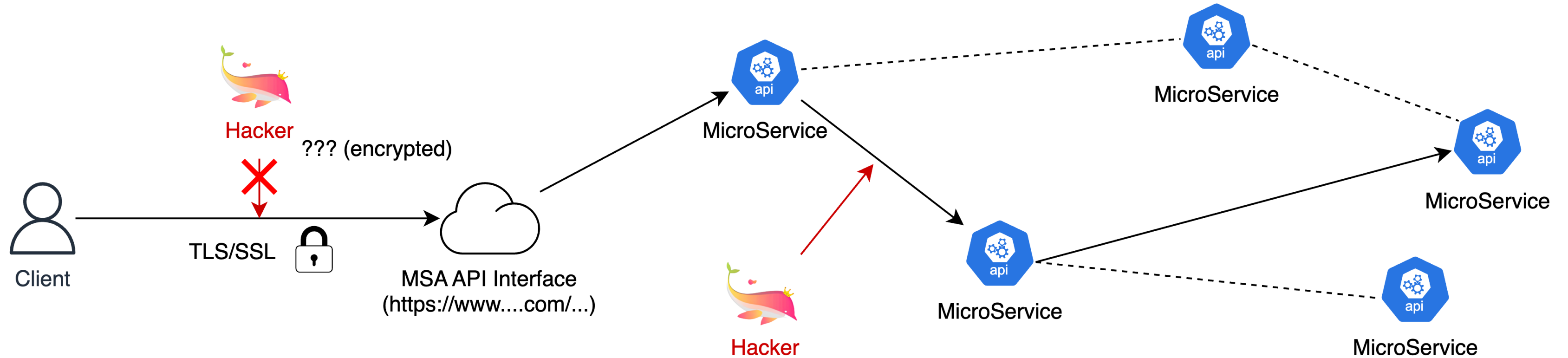
통신에 있어 아무런 보안도 없는 상황

Session 5. 우리의 트래픽은 과연 안전하게 목적지로 도착할까?



통신에 있어 아무런 암호화를 적용하지 않았기 때문에 중간에서 가로채기 가능
(중간자 공격 등)

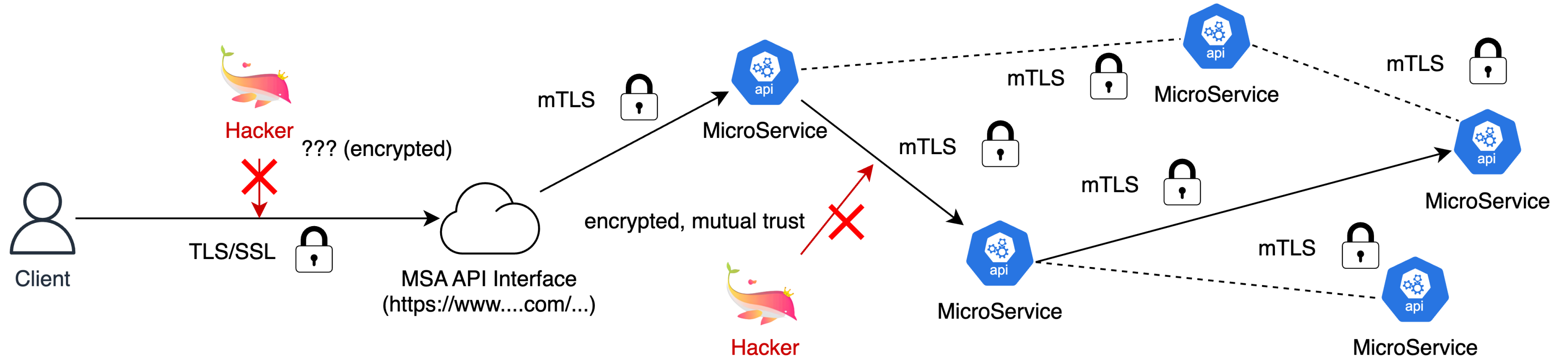
Session 5. 우리의 트래픽은 과연 안전하게 목적지로 도착할까?



서버에서 제공되는 Public Key로 암호화(TLS Handshake, 이후엔 세션키로 암호화) (HTTPS)

" 하지만 신뢰하는 트래픽만 받아야 하는 마이크로 서비스에선? " (TLS는 단방향 신뢰)

Session 5. 우리의 트래픽은 과연 안전하게 목적지로 도착할까?



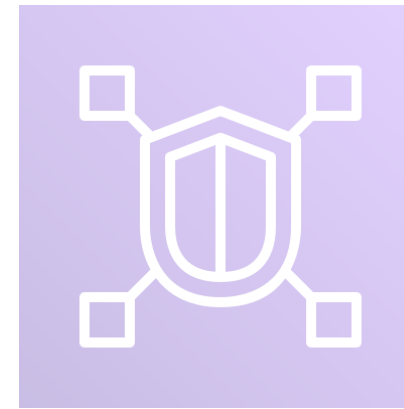
양방향으로 인증서를 내면 암호화 + 신뢰성까지 보장됨 ⇒ **mTLS (mutual TLS)**

" 그런데 수 많은 마이크로 서비스 연결마다 mTLS를 하나하나 설정해줘야 하나요? "

Session 5. Istio를 통한 서비스 메시 구성하기



(mTLS 기본 적용)



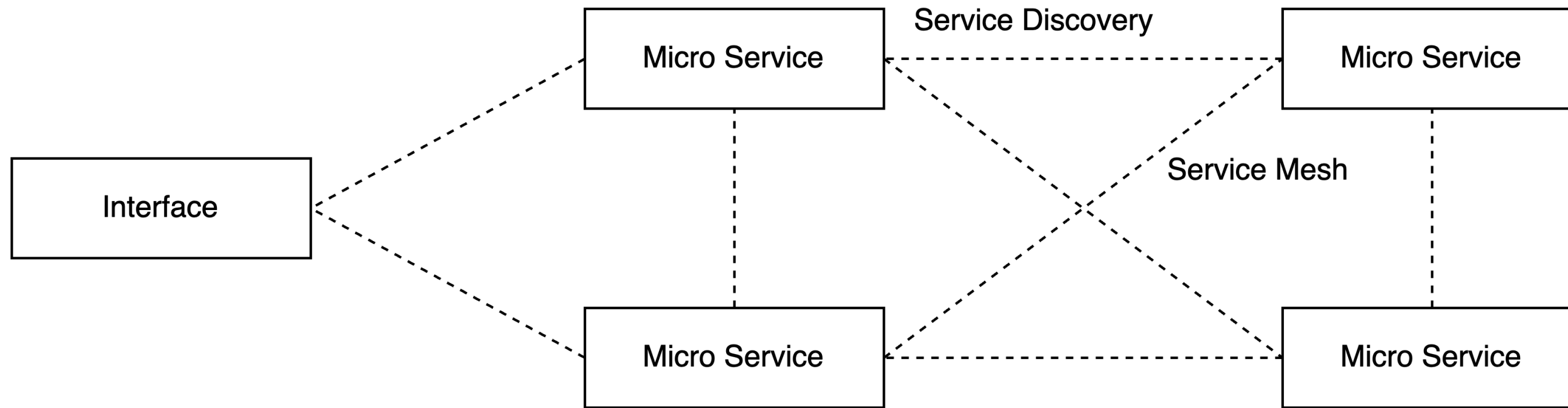
AWS
VPC Lattice

(시간 관계상 생략, 추후 전공 동아리에서 발표 예정)

Istio = 오픈소스 서비스 메시 (Envoy Sidecar 패턴 사용)

AWS VPC Lattice = 다중 VPC/계정, 온프레미스 환경에서 동작 가능한 AWS 완전 관리형 서비스 메시
(또는 AppMesh)

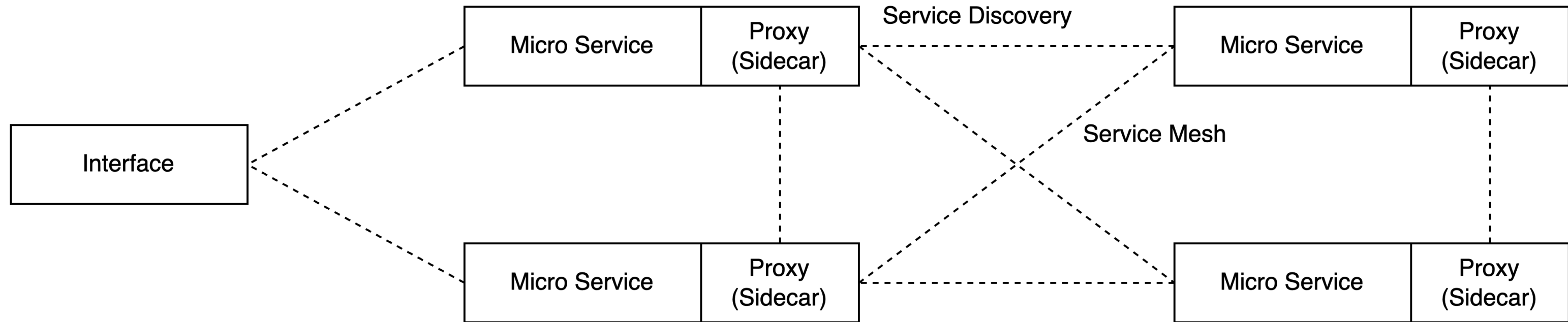
Session 5. Istio를 통한 서비스 메시 구성하기



어떻게 마이크로 서비스에 붙어서 서비스 메시를 구성할까?

1. 클러스터 외부에서 완전관리형으로 네트워킹 담당 및 서비스 메시 구성 (VPC Lattice)
2. 각 마이크로 서비스(파드)에 사이드카(Sidecar) 형태로 프록시 붙이기

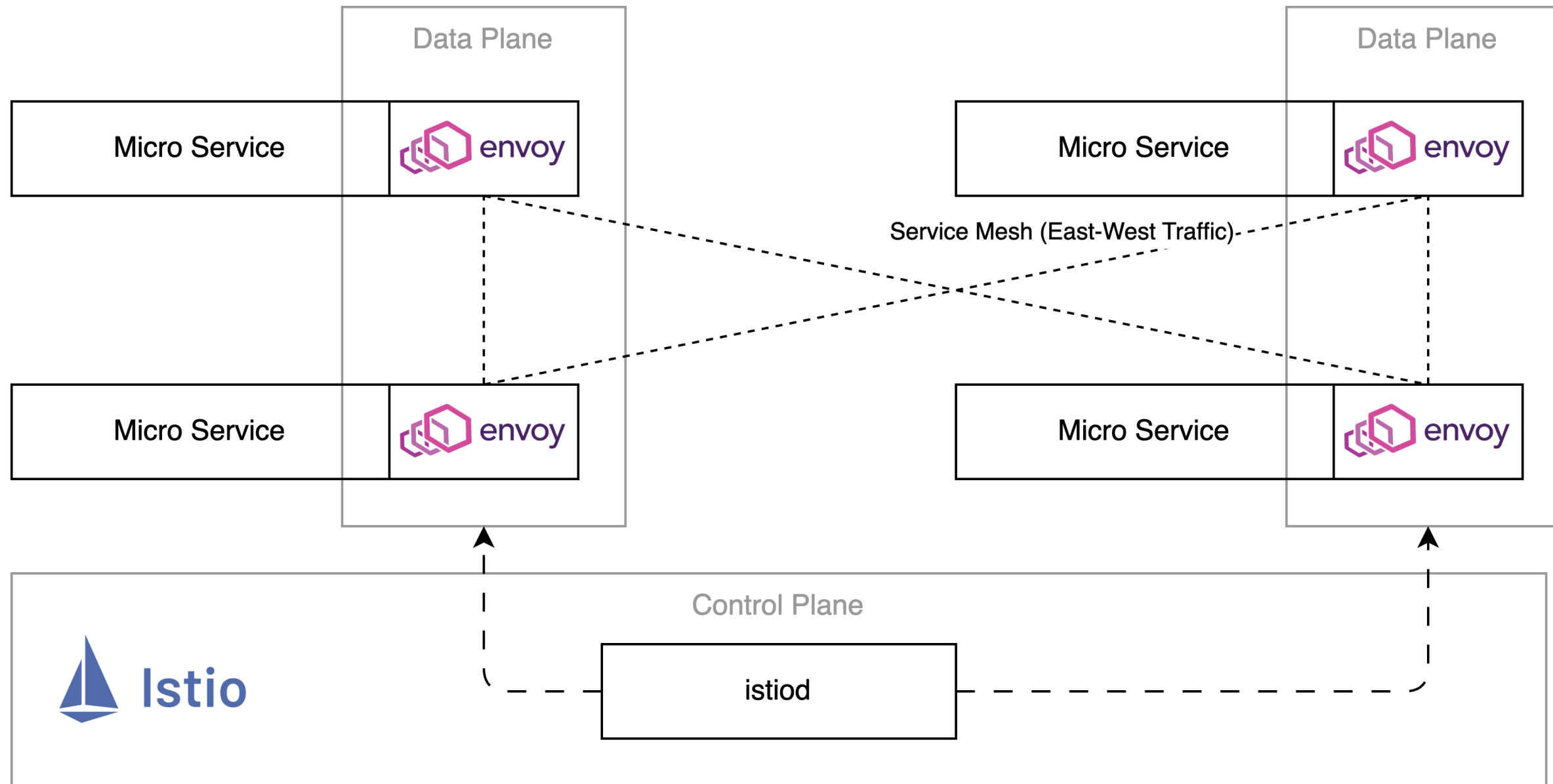
Session 5. Istio를 통한 서비스 메시 구성하기



마이크로 서비스(파드) 옆에 프록시를 붙이고 해당 프록시에 네트워킹 기능을 맡김

⇒  **envoy** 그리고 전체적인 Envoy Proxy를 컨트롤하는게 **Istio**

Session 5. Istio를 통한 서비스 메시 구성하기



Envoy Proxy 관리 외에도 Istio Gateway Controller를 지원하여 East-West(내부) 트래픽 외에도 North-South(외부) 트래픽까지 담당 가능

Session 5. Istio를 통한 서비스 메시 구성하기



Secure by default

기본적으로 서비스 간 네트워킹에서 mTLS로 암호화되어 통신,
세부적인 제어 정책으로 Zero-Trust를 기본으로 지원

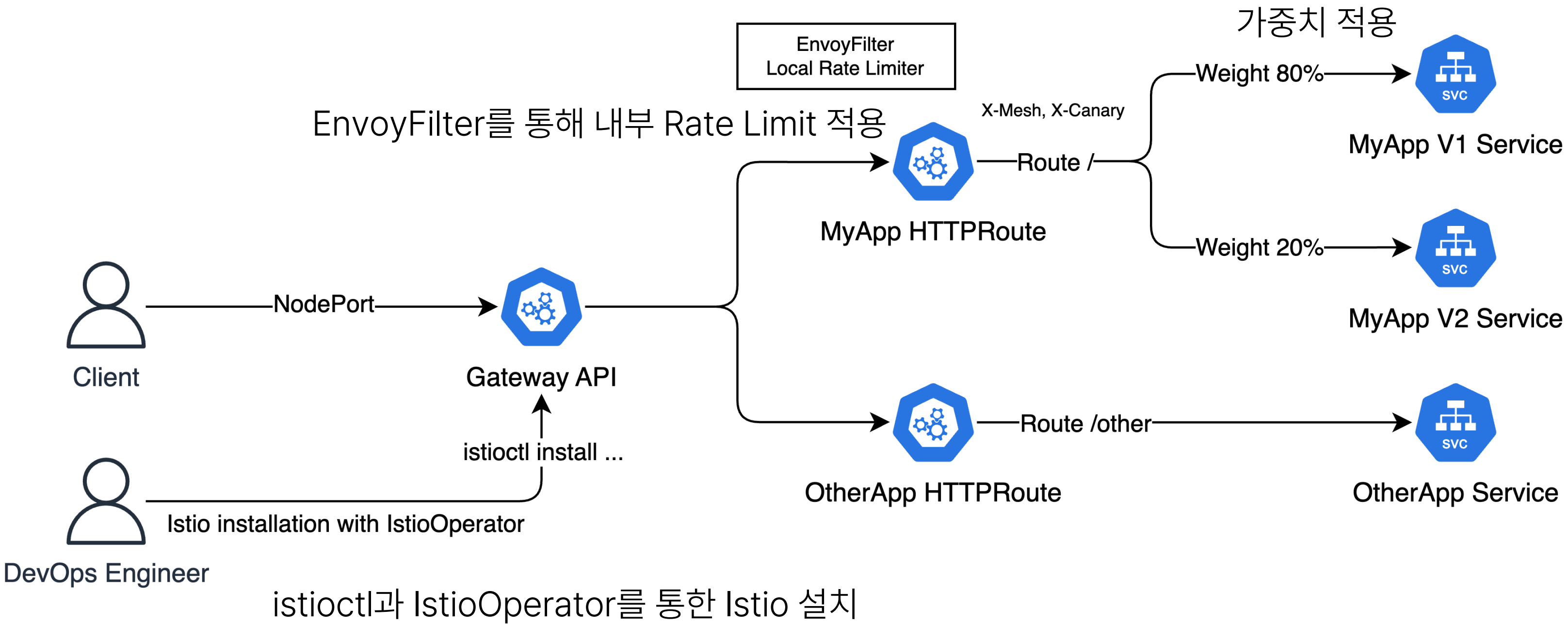
Increase observability

서비스 간 네트워크 흐름을 실시간으로 모니터링,
상세한 Metrics와 로그를 제공

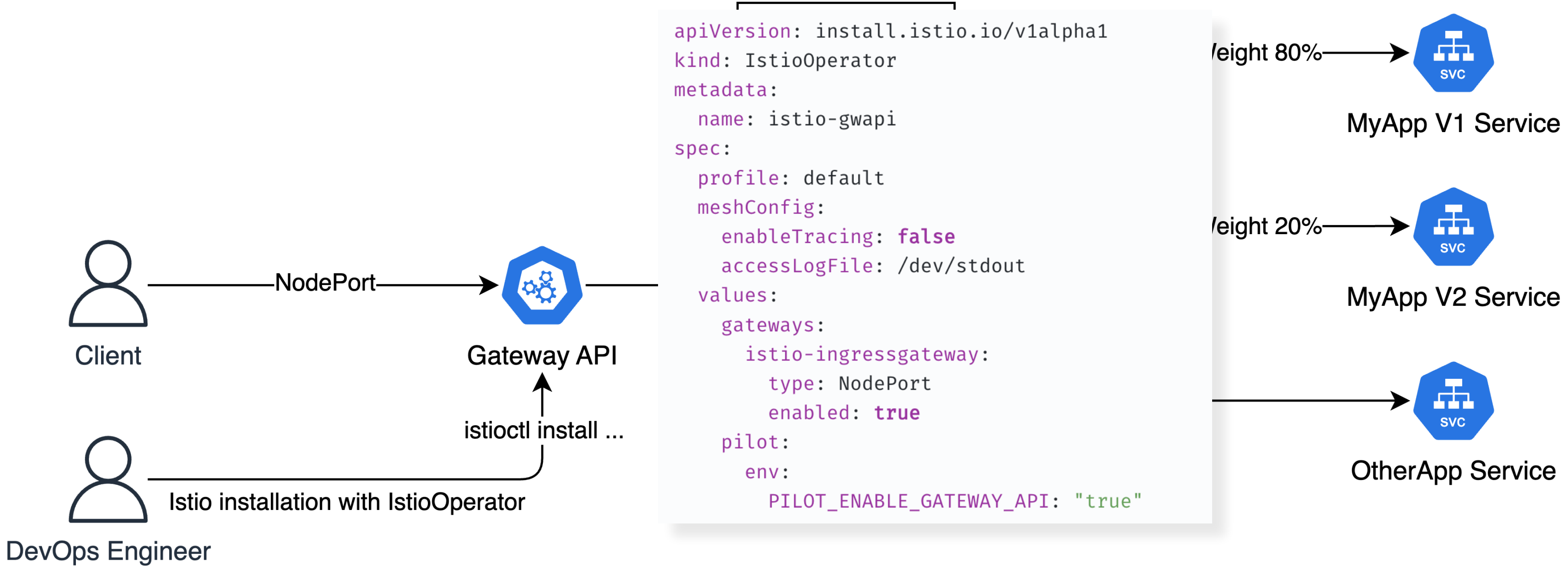
Manage traffic

Sidecar 프록시(Envoy)를 통해 서비스 레벨의 트래픽을 제어,
A/B 테스트와 카나리 배포 등을 쉽게 구성 가능

Session 5. Istio를 통한 서비스 메시 구성하기

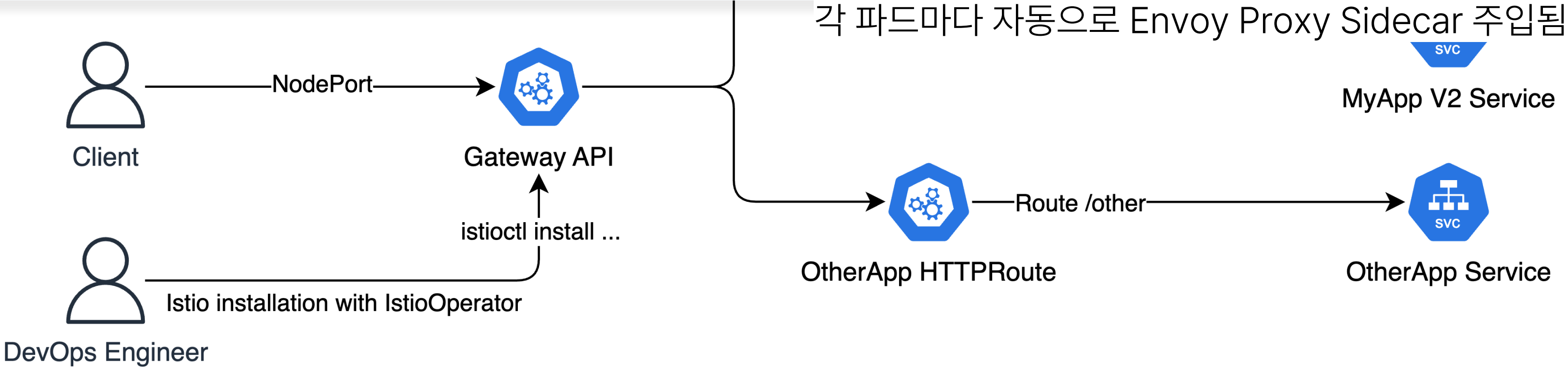


Session 5. Istio를 통한 서비스 메시 구성하기

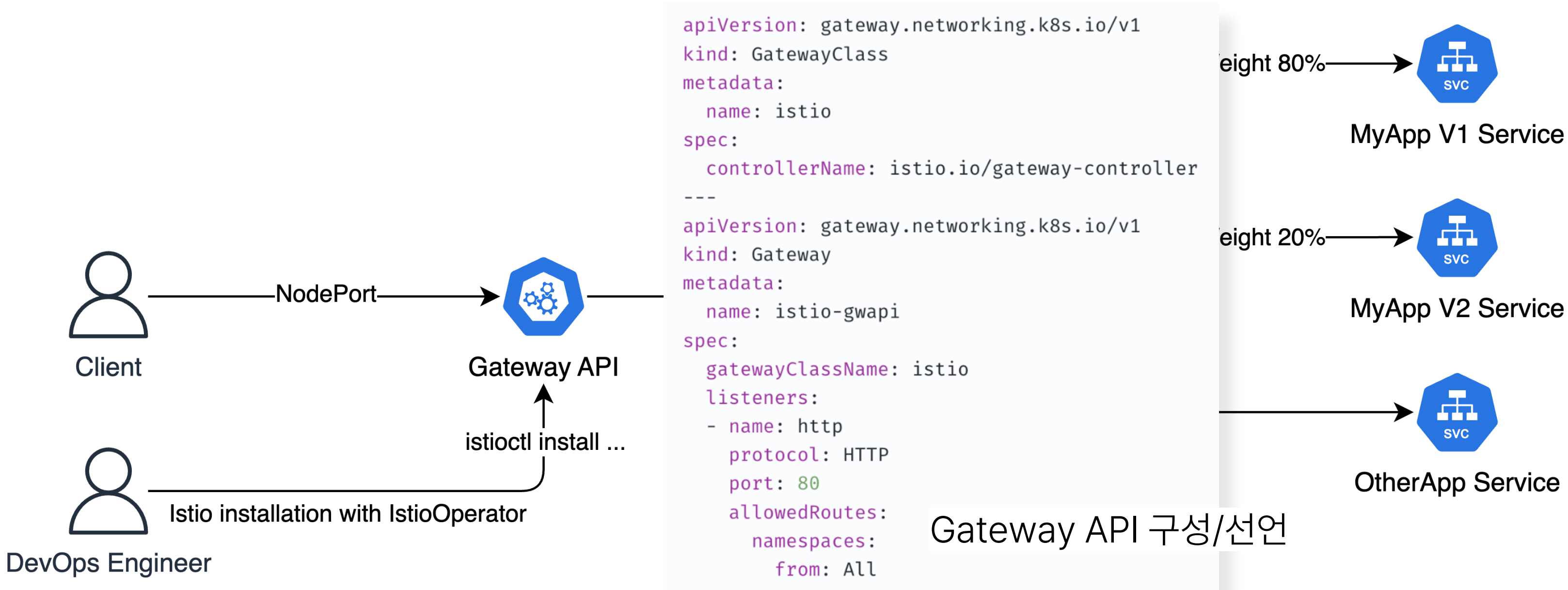


Session 5. Istio를 통한 서비스 메시 구성하기

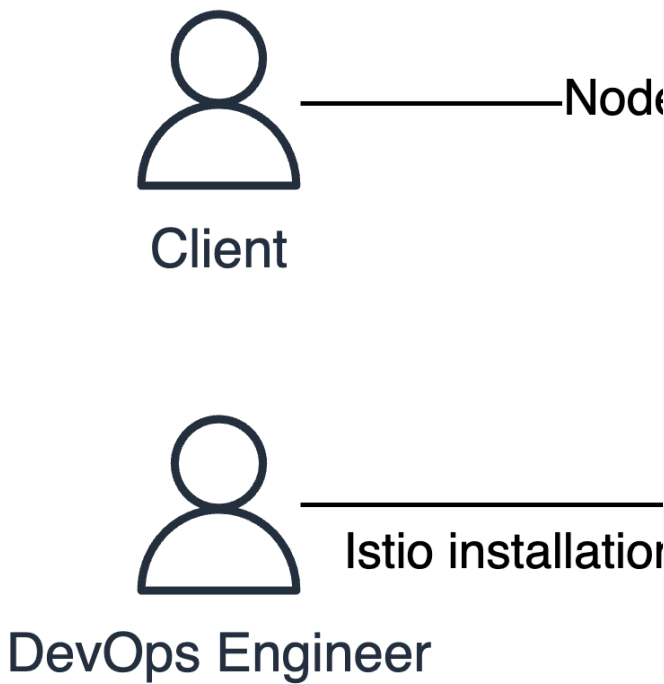
```
Init Containers:
  istio-init:
    Container ID:  containerd://cc8d6b07b277c8c7d2175e65c9c3d720fc5129288c8ccb5252ed4d348d56761a
    Image:         docker.io/istio/proxyv2:1.27.2
    Image ID:      docker.io/istio/proxyv2@sha256:b00a23cb37e7b8e422b57e617c1bb7304955e368308b5c166e38f0444e0f5a08
    Port:          <none>
    Host Port:     <none>
```



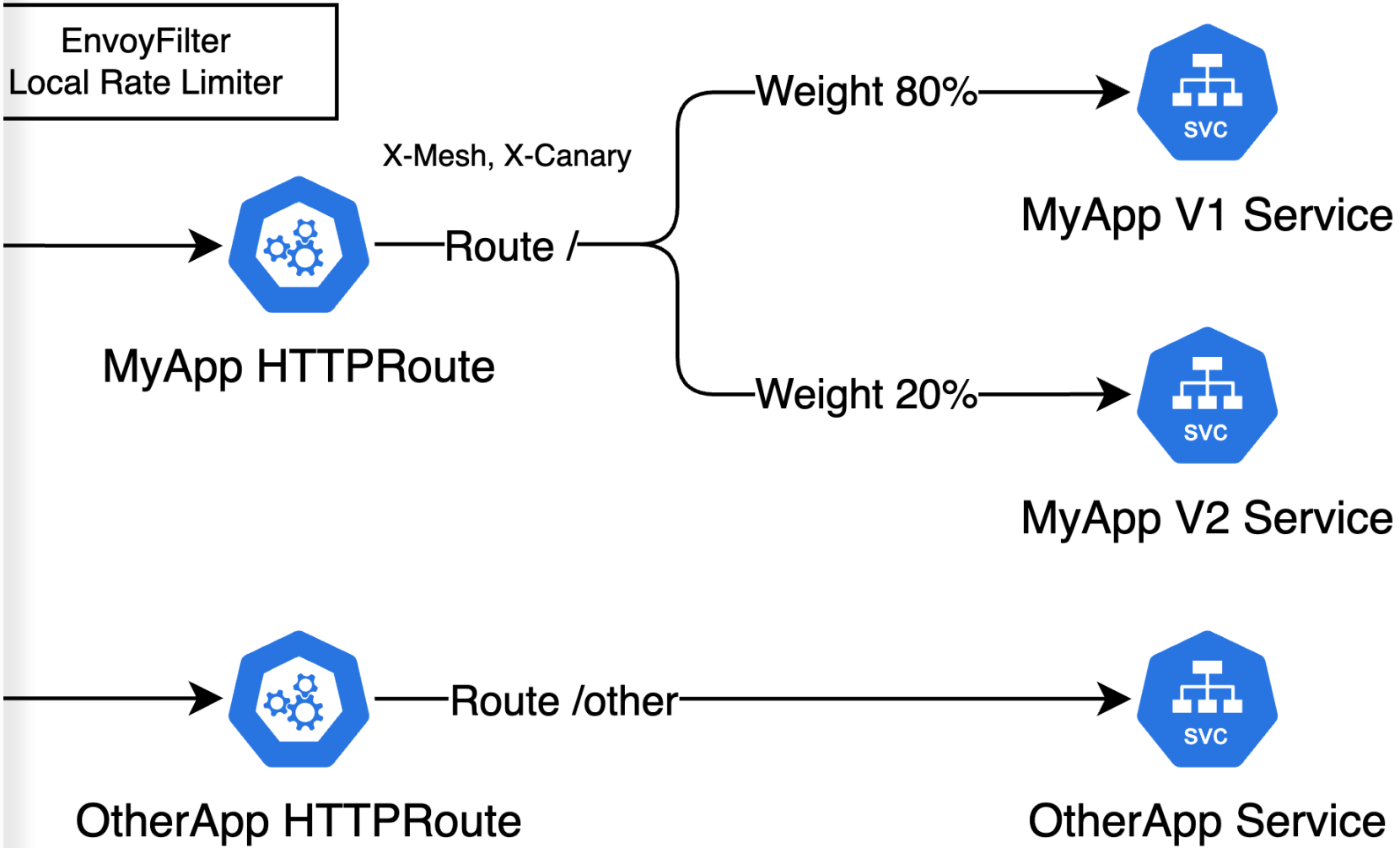
Session 5. Istio를 통한 서비스 메시 구성하기



Session 5. Istio를 통한 서비스 메시 구성하기



```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: myapp-route
spec:
  parentRefs:
  - kind: Gateway
    name: istio-gwapi
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /
    backendRefs:
    - name: myapp-v1
      port: 80
      weight: 80
    - name: myapp-v2
      port: 80
      weight: 20
  filters:
  - type: RequestHeaderModifier
    requestHeaderModifier:
      add:
      - name: X-Mesh
        value: istio
      set:
      - name: X-Canary
        value: "v2-20pct"
  - type: ResponseHeaderModifier
    responseHeaderModifier:
      add:
      - name: X-Mesh
        value: istio
      set:
      - name: X-Canary
        value: "v2-20pct"
```



HTTPRoute(Gateway, Service HTTP 연결) 및 가중치, 커스텀 헤더 추가

Session 5. Istio를 통한 서비스 메시 구성하기

```
/Users/user [ky0422@MacBookPro] [9:39]
> curl -v http://15.164.182.29:31134
* Trying 15.164.182.29:31134...
* Connected to 15.164.182.29 (15.164.182.29) port 31134
> GET / HTTP/1.1
> Host: 15.164.182.29:31134
> User-Agent: curl/8.7.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< x-powered-by: Express
< content-type: text/html; charset=utf-8
< content-length: 19
< etag: W/"13-wu88cbqLodmLX52dC0F9P6wgZgA"
< date: Tue, 21 Oct 2025 00:39:23 GMT
< x-envoy-upstream-service-time: 2
< server: istio-envoy
< x-canary: v2-20pct
< x-mesh: istio
<
* Connection #0 to host 15.164.182.29 left intact
Hello from myapp v1%
```

Envoy 관련 헤더 및 커스텀 헤더 추가 (x- 접두사)

Session 5. Istio를 통한 서비스 메시 구성하기

```
import requests
import time

url = "http://15.164.182.29:31134"
count = 100
v1 = v2 = 0
```

```
for i in range(count):
    try:
        r = requests.get(url, timeout=3)
        text = r.text.strip()
        if "v1" in text:
            v1 += 1
        elif "v2" in text:
            v2 += 1
    except requests.RequestException:
        pass
    time.sleep(0.1)

total = v1 + v2
print(f"v1: {v1}, v2: {v2}")
if total > 0:
    print(f"v1: {v1*100/total:.1f}% / v2: {v2*100/total:.1f}%")
else:
    print("No valid responses.")
```

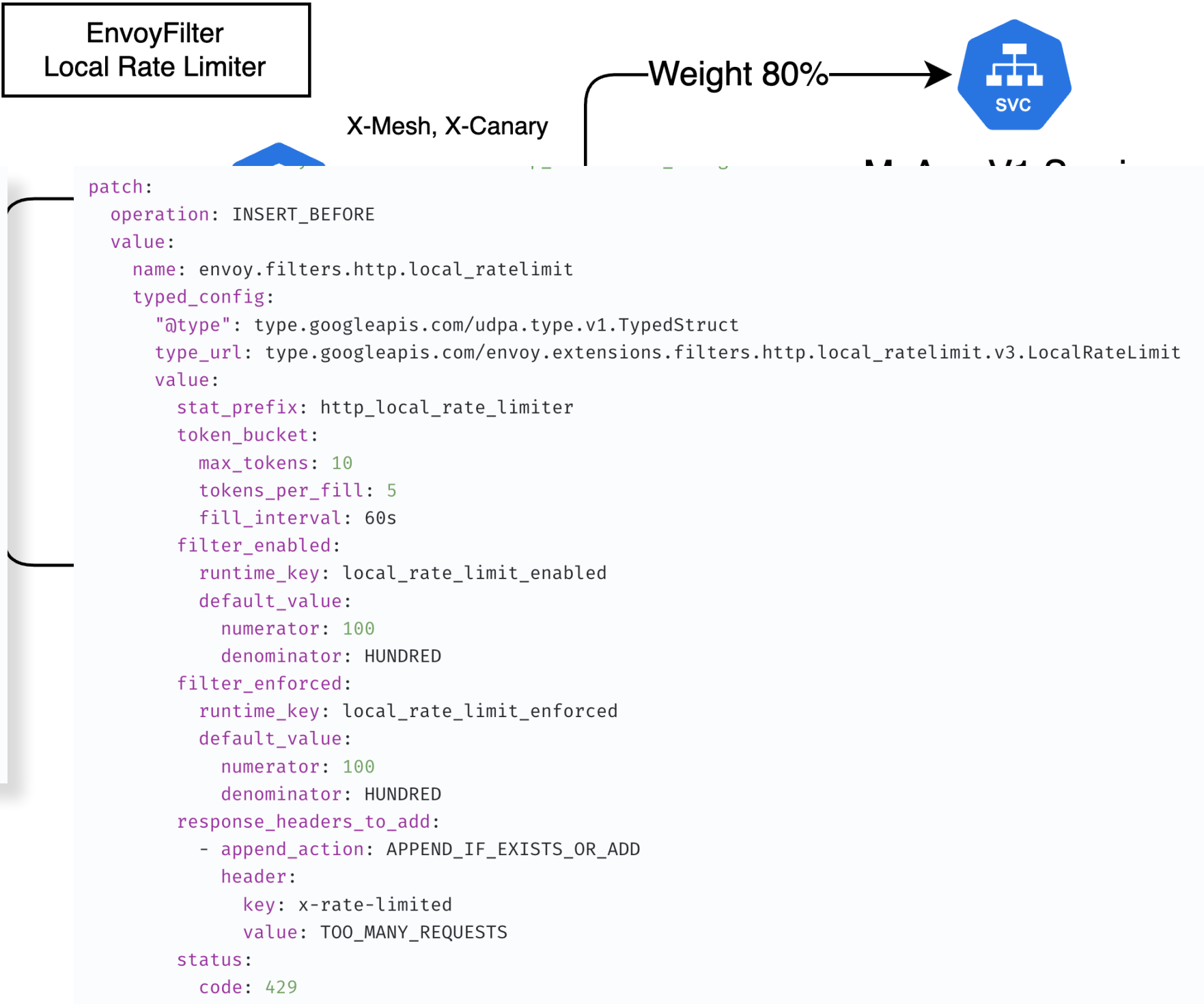
```
> python3 check_canary.py
```

```
v1: 79, v2: 21
```

```
v1: 79.0% / v2: 21.0%
```

설정된 가중치에 맞게 로드밸런싱 됨

Session 5. Istio를 통한 서비스 메시 구성하기



EnvoyFilter — LocalRateLimiter 적용

Session 5. Istio를 통한 서비스 메시 구성하기

```
seq 1 100 | \
xargs -I{} -P100 curl -s -o /dev/null -w "%{http_code}\n" \
    http://3.34.180.38:32165 | \
sort | uniq -c && \
curl -I http://3.34.180.38:32165
```

```
20 200
80 429
HTTP/1.1 429 Too Many Requests
x-rate-limited: TOO_MANY_REQUESTS
content-length: 18
content-type: text/plain
date: Tue, 21 Oct 2025 05:55:11 GMT
server: istio-envoy
x-envoy-upstream-service-time: 0
x-canary: v2-20pct
x-mesh: istio
```

초반 토큰 버킷에 토큰 10개, 10개 소비 이후 429 반환
(v1, v2 각각 적용돼서 20개의 요청만이 200 OK)

Istio + Envoy를 통해 서비스 메시지를 구성하였기 때문에
필터링을 하지 않는 한 모든 메시지에 적용 및 자동으로 mTLS 적용



👋 Good Bye, Null4U in the SMCH 2025 IT Conference

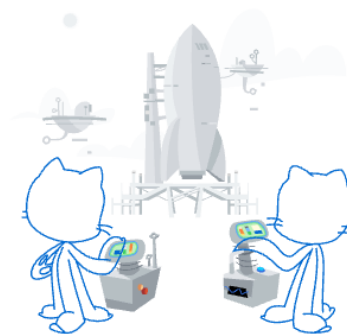
Session 06

보안과 학생들을 위한 — 프로그래밍 7-8년차가 공유하는 경험담과 팁들

Contribution activity

December 2019

🚀 Joined GitHub



@yulmwu

Joined GitHub
on December 22, 2019

2025

2024

2023

2022

2021

2020

2019

국가기술훈격증
정보처리기능사

한국산업인력공단

이름 김준영
직무 정보기술
합격 2023.09.20

HRDK

Session 6 ToC

- 01 • 절대 늦지 않았다.
- 02 • 자신이 하는 일의 목적을 알고, 작은 목표를 세우자.
- 03 • 책 한권만 읽은 사람이 더 무섭다.
- 04 • 파기 시작한 우물, 파는김에 깊게 파자.
- 05 • 학교에서 배우는 내용은 배우는 이유가 충분히 있다.

—— 자기개발

- 06 • 따라하지만 말고, "왜?"를 일상화하자.
- 07 • 프로젝트를 두려워하거나 망설이지 말자.
- 08 • 모든 것이 포트폴리오이다.
- 09 • 기록하는 습관을 가지자.

—— 습관

Session 6 ToC

10 • AI를 절대적으로 믿지 말자. — AI

11 • Agent를 활용하자.

12 • AI를 통한 바이브 코딩, 한가지 습관만 바꾸면 실력으로 돌아온다.

13 • 구글은 모든 것을 알고있다. (트러블 슈팅과 공부 위한 올바른 검색법) — 개발

14 • 언어 하나를 깊게 파자.

15 • 코딩은 외우는게 아니다, 읽고 로직으로 접근하자.

16 • 인간은 도구를 사용하는 동물이다.

17 • 개발에서 가장 어려운 것은 최적화이다.

18 • 버전 관리는 선택이 아닌 필수다.

Session 6 ToC

19	• 학생 혜택을 적극적으로 누리자. (Github Education)	—— 혜택
20	• 전공동아리도 우리에게 제공된 혜택이다.	
21	• 클린 코드는 협업의 기본이다.	—— 협업
22	• 개발자의 능력은 개발 능력과 커뮤니케이션 능력으로 나뉜다.	
23	• 인맥을 적극적으로 활용하자.	
24	• 온라인 개발자 커뮤니티도 인맥을 만들 수 있다.	

Session 6. 굳이 금같은 시간을 소비하며 24가지의 조언을 드리는 이유는..

OO아, 그거 공부 할땐 ~~부터 하는게 좋ㅇ.. (말 끊기) 어찌라고 내 맘대로 할거임

1:1로 대화하면서 말하면 듣지를 않음 ⇒ '나중에 세미나 등에서 시간 잡아서 말해봐야지...'

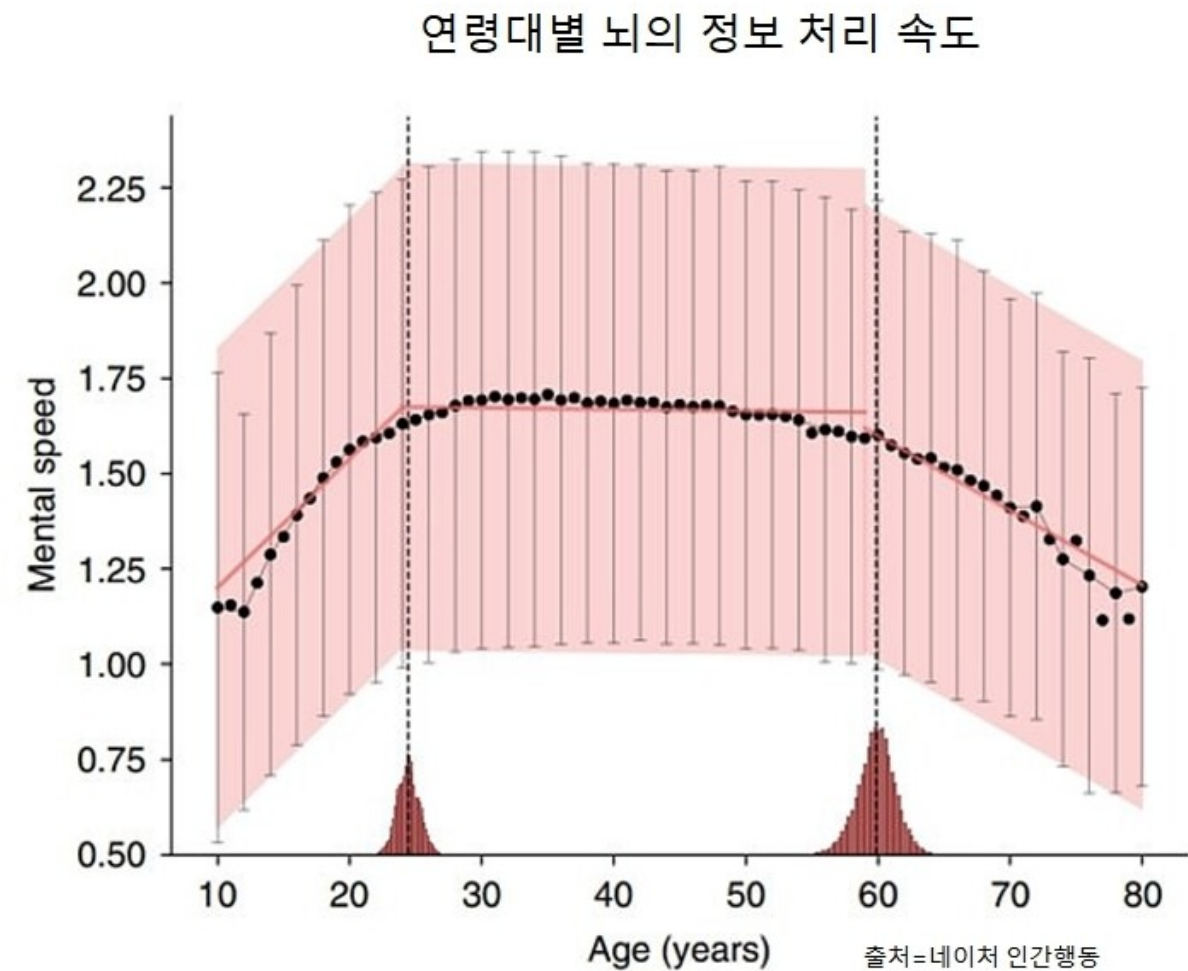
누군가에게 조언할 짬은 아니지만

지금까지 여러 친구들을 보면서 저마다의 이유로 답답했던 적이 많았음..

그래도 나름 전공 실력이 있다고 생각하였기에 드리는 최소한의 조언

Session 6. 자기개발 — 절대 늦지 않았다.

"누구는 초등학생, 중학생때 부터 해왔다는데 가망이 없다..." (해보려는 의지도 없이 포기)



⇒ 지금 시기에 시작해도 꾸준히만 한다면 충분히 가능성 있음

가장 어려운 것: 끈기, 포기하지 않는다는 마음

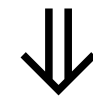
"나랑 안맞던데.."

⇒ 라고 말한 학생 중 90%가 최소한의 노력도 없었음 (핑계)

"오히려 초등학생/중학생 초반 때 부터 해왔다는게 특이한 케이스"

Session 6. 자기개발 — 자신이 하는 일의 목적을 알고, 작은 목표를 세우자.

(목표 없음) ⇒ "쿠버네티스 배워야겠다.." ⇒ "근데 이걸 내가 배워서 뭘 하는데?"



(목표: 웹 애플리케이션 배포) ⇒ "쿠버네티스 배워야겠다.." ⇒ DevOps 개발자로서 나아감

목표가 없다면 배운다는 이유도 없을테고, 이는 곧 발전을 하기 어렵다는 의미

" 하지만 너무 큰 목표를 세운다면... "

Session 6. 자기개발 — 자신이 하는 일의 목적을 알고, 작은 목표를 세우자.

(목표: 취업하기) ⇒ 뭐부터 해야할까? 어떻게..?
 내가 취업을 할 수 있을까? ⇒ 발전 없음

큰 목표(취업)을 배경으로 깔아두고, 단기적이고 작은 목표를 여러개/여러번 가지자.

(목표: 게시판 만들기) ⇒ 이렇게, ~~를 사용하여 만들자. ⇒ 점차적으로 발전
완성 후 포트폴리오로 활용하기

Session 6. 자기개발 — 책 한권만 읽은 사람이 더 무섭다.



Blog: *Kubernetes는 내부 런타임을 Docker(Dockershim)를 사용합니다. (작성일: 2019년 XX..)*



쿠버네티스를 공부하는 학생: 음.. 그렇구나. ("쿠버네티스 기본 런타임 = 도커" 각인)

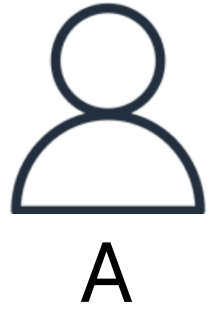
컨테이너 런타임

참고: Dockershim은 쿠버네티스 릴리스 1.24부터 쿠버네티스 프로젝트에서 제거되었다. 더 자세한 내용은 [Dockershim 제거 FAQ](#)를 참고한다.

(쿠버네티스는 1.24부터 내부적으로 containerd를 기본 런타임으로 사용)

⇒ 여러 자료(책, 블로그, 논문, 유튜브 등)를 보면서 **교차 검증 필수** (AI도 마찬가지)

Session 6. 자기개발 — 파기 시작한 우물, 파는김에 깊게 파자.



" 뭘 공부해야 할까? " (웬만한건 다 한번씩 해봄)



React 공부
(useState 까지)



TypeScript 공부
(union 까지)



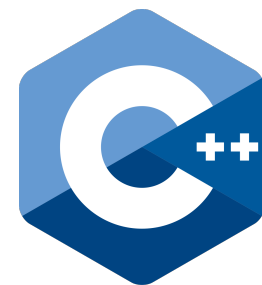
NodeJS 공부
(Express 까지)



쿠버네티스 공부
(Service에서 포기)



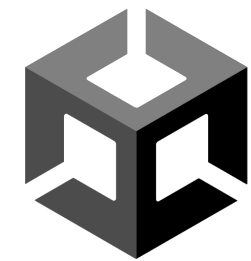
Java Spring 공부
(MVC 까지)



C++ 공부
(클래스 까지)



AWS 공부
(ECS에서 포기)



Unity 공부
(Collider2D에서 포기)

Session 6. 자기개발 — 파기 시작한 우물, 파는김에 깊게 파자.



A

" 뭘 공부해야 할까? " (웬만한건 다 한번씩 해봄)



(필자)

" 일단 학교에서 배우는 Java/JavaScript/SQL/보안 공부나 열심히 해라... " (조언)



A

" 다 알고있는데? 그걸 왜함 " (수행평가 반타작) (실화)

기본기도 안되어 있는데 목표도 없이 이것저것 해보다 보니 제대로 할 수 있는게 하나도 없는 상황

제대로, 최소 하나라도 깊게 우물을 파서 자신만의 스킬을 만들어야 함

(뭘 해야할지 모르겠다면 학교에서 배우는 전공에 우선적으로 집중)

Session 6. 자기개발 — 파기 시작한 우물, 파는김에 깊게 파자.



A

" 도커부터 공부해야지... 아니다 쿠버네티스 공부할래.. 컨테이너가 뭐지? " (실화)

공부를 할땐 수학 공부처럼 **하위 개념을 공부하고 상위 개념을 공부해야 함**

⇒ 순서를 자기 마음대로, 또는 공부 순서를 몰라서 상위 개념부터 공부하려는 사람이 너무 많음 (로드맵의 존재 이유)

구글에 검색하여 **책이나 문서, 개념서** 등의 목차만 봐도 공부해야 할 순서를 쉽게 알 수 있음

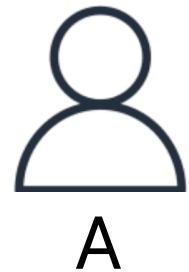


혹은 **roadmap.sh** 서비스를 활용할 (개발 관련 거의 대부분의 로드맵이 있음)

Session 6. 자기개발 — 학교에서 배우는 내용은 배우는 이유가 충분히 있다.

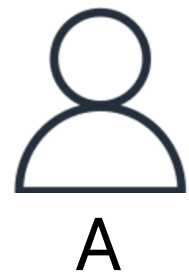


" 일단 학교에서 배우는 Java/JavaScript 등등 전공 공부나 열심히 해라... " (조언)



" 그 뻔한걸 왜 함? " (수행평가 반타작) " 난 빨리 타입스크립트 공부나 해보고 싶은데? " (자신감)

↓ 공부 시작



' 타입스크립트 공부하려니깐 뭐가 뭔지 하나도 모르겠네... 타입? 인터페이스? —... '

⇒ 학교에서 배웠던 내용, 즉 기본기가 없어 공부하기가 어려운 상황

학교에서 배우는 내용은 **기본기**로, 나중에 하나도 빠짐 없이 사용/응용하게 되는 상황이 오게 됨

Session 6. 습관 — 따라하지만 말고, "왜?"를 일상화하자.



선생님

" Module not found 오류가 났을땐 pip install scapy로 설치하면 됩니다. "



' scapy가 무슨 라이브러리지길래 이 코드에서 사용되는거지?' (검색해보기)

모르는 것, 또는 에러를 "직접" 찾아보면서 해결하고 찾아보는 것이 학습하는데 가장 큰 도움이 됨
(100% 확신, 보기만 하는게 아닌 "직접" 해보는게 핵심임)



Session 6. 습관 — 프로젝트를 두려워하거나 망설이지 말자.



' 게시판 프로젝트를 해보고 싶은데.. 뭐부터 해야할지 막막하고 어렵다.. '

⇒ 그러한 프로젝트를 진행하는데 돈이 들지 않음, 필요한 것은 의지와 시간 뿐

프로젝트 = 포트폴리오

배웠던 내용을 활용하면서 공부까지 가능

프로젝트는 장기적으로 계획하는 것이 베스트, 중요한건 **꾸준히**, 그리고 **성실하게**

Session 6. 습관 — 모든 것이 포트폴리오이다.



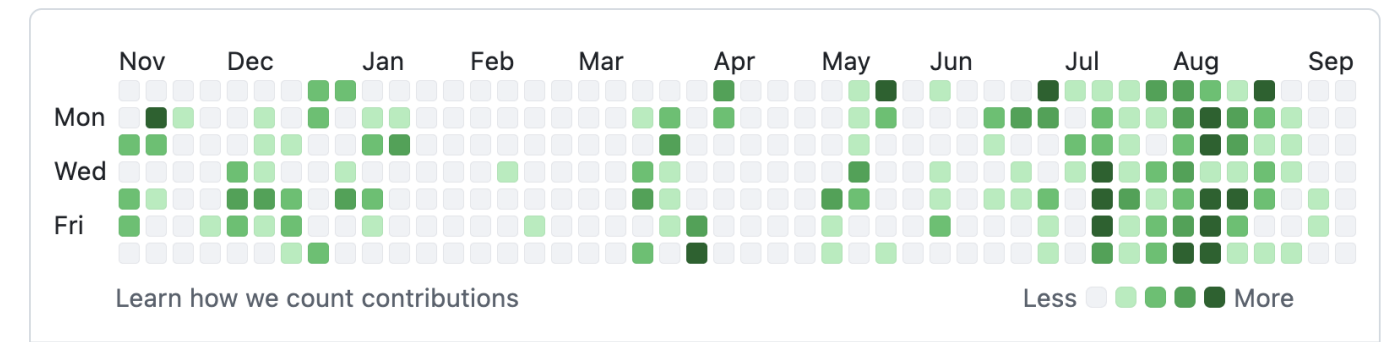
"그럼 프로젝트만 좀 하고 깃허브에 대충 올려두면 되는 거임?"

⇒ 포트폴리오엔 프로젝트만 있진 않음

- 블로그 / 전공 관련 유튜브 운영
- 대회(CTF 등), 컨퍼런스, 세미나 참여
- 타 프로젝트 기여 또는 협업
- (깃허브 등) 활동(커밋 등) 내역
- 발표, 보고서 자료(프레젠테이션 등)
- 자격증
- 학교 생활기록부 등등

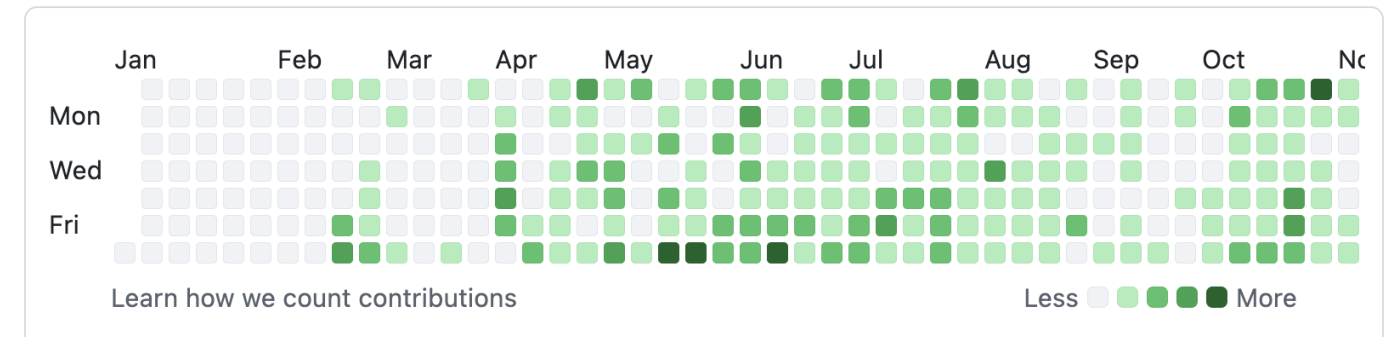
1,113 contributions in the last year

Contribution settings ▾



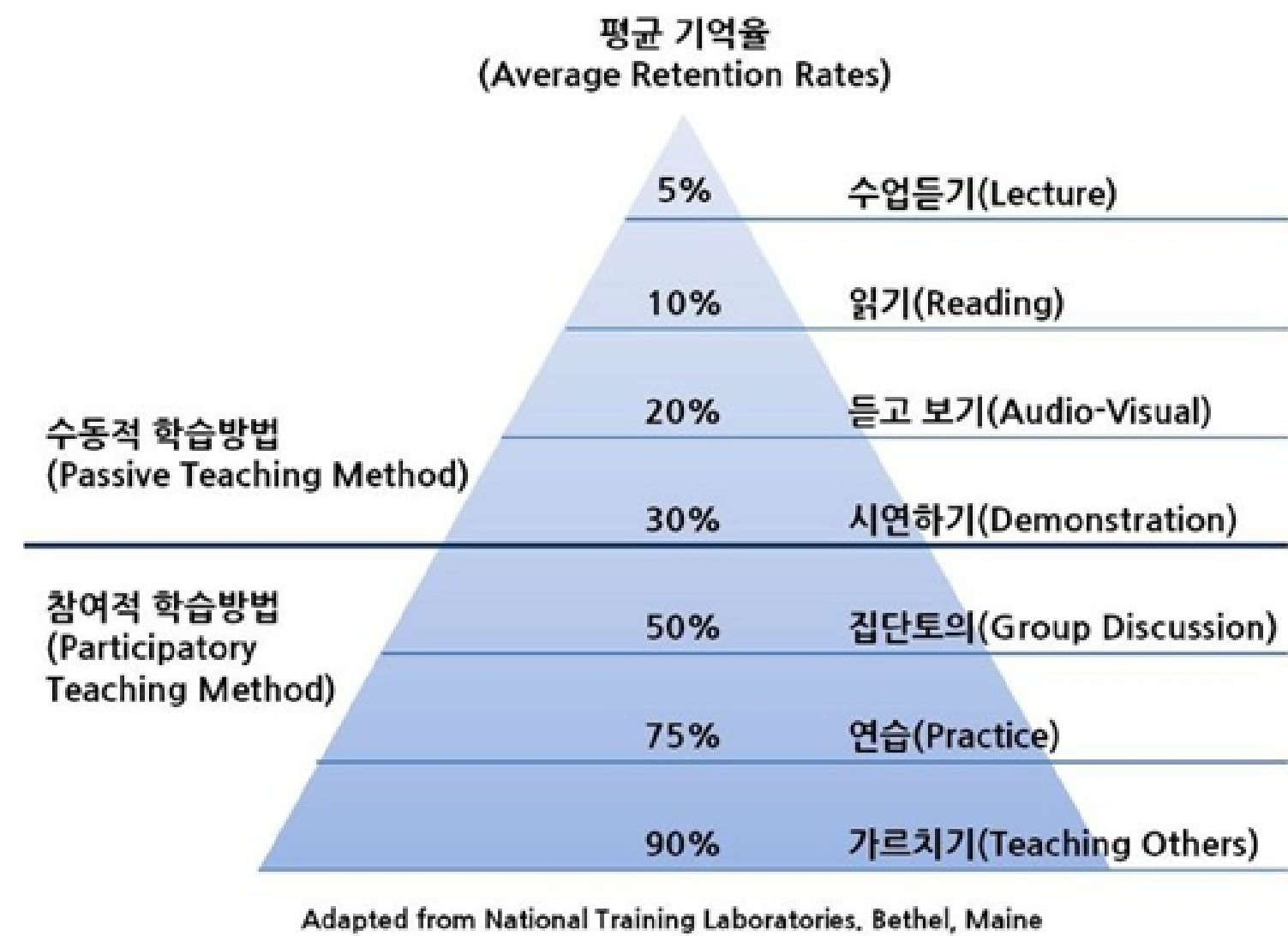
2,559 contributions in 2022

Contribution settings ▾



⇒ 했던 모든 전공 활동이 포트폴리오에 포함, 다만 그 활동을 유의미한 결과물로 남겨야 함

Session 6. 습관 — 기록하는 습관을 가지자.



직접 가르치기? ⇒ 글로 기록하기



또는 블로그 플랫폼(Tistory, Naver Blog, Velog ...)

단, 중요한 것은 단순한 개념/요점 정리가 아닌
누군가에게 성의껏 가르친다는 마인드로

"배웠던 내용을 직접 가르치는 것이 기억율이 가장 높다."

Session 6. AI — AI를 절대적으로 믿지 말자.

istio PILOT_DISABLE_GATEWAY_API option

ChatGPT와 같은 AI 및 구글 AI 요약 등 모두 포함

AI 모드 전체 이미지 쇼핑 동영상 뉴스 짧은 동영상 더보기 도구

이것을 찾으셨나요? istio **PILOT_DISABLE_GATEWAY_API** option

◆ AI 개요

The `PILOT_DISABLE_GATEWAY_API` option was previously a way to disable Istio's controller for the Kubernetes Gateway API, but it has been deprecated and removed. Istio now intends for the Gateway API to be the default for traffic management. [↗](#)

In older versions of Istio, you could set this environment variable on the `istiod` deployment to prevent it from managing Gateway API resources. This would allow you to continue using only the original Istio `Gateway` and `VirtualService` resources. [↗](#)

이런 옵션은 존재 하지도 않았음

하지만 AI는 존재했었으나 현재는

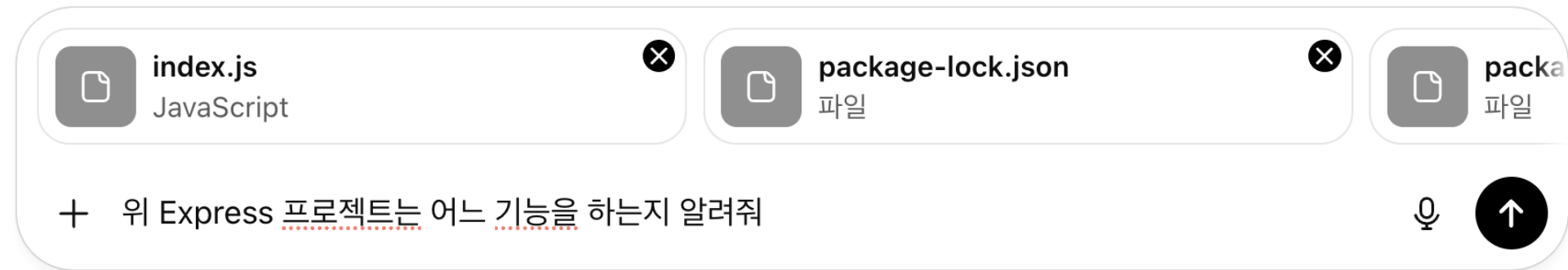
Deprecated 되었다고 잘못된 정보 제공

AI는 잘못된 정보를 학습할 수 있고, 과거의 정보를 현재의 정보처럼 꾸며 제공하는 등

100% 신뢰하면 안되는 도구

만약 시간에 따라 자주 변동되는 대상이라면 AI 사용 시 "웹 검색" 기능을 활용

Session 6. AI — Agent를 활용하자.



단순히 파일을 읽고 코드를 바탕으로 분석을 해주기 때문에 비효율적
여기에 MCP를 사용하여 내 프로젝트 워크스페이스에 직접 액세스하게 한다면?

Agent를 통해 자율적으로 액세스하여 더욱 자세한 분석 가능
(예: 명령어/코드 실행, 코드 작성 ⇒ **바이브코딩**, 단 큰 코드베이스라면 토큰 소비로 어려움)



ChatGPT Codex, Gemini CLI, Copilot Agent, Cursor(IDE), Continue(Ext & API) ...

Session 6. AI — AI를 통한 바이브 코딩, 한가지 습관만 바꾸면 실력으로 돌아온다.



GPT야, 전공 과목 실습으로 (생략) 이러한 기능을 가진 프로그램을 만들어야 하는데 만들어줘



(3초 경과) 제시한 조건대로 main.py와 lib/...에 코드를 작성하였습니다. 실행까지 해드릴까요?

비종사자, 관련 업계나 진학/취업을 희망하지 않는 경우 ⇒ **괜찮음** (이해 안되어도 지장 없음)

전문가/전공자/높은 실력, 업계 종사자 등 ⇒ **괜찮음** (이미 다 알고있는 내용)

코딩이나 기술을 공부하고 있거나 아직 잘 모르지만 취업이나 진학을 희망하는 경우 (대부분)

⇒ **문제 발생** (이해가 안되지만 AI가 코딩을 다 해주니 **AI에 의존하게 됨**)

Session 6. AI — AI를 통한 바이브 코딩, 한가지 습관만 바꾸면 실력으로 돌아온다.

코딩이나 기술을 공부하고 있거나 아직 잘 모르지만 취업이나 진학을 희망하는 경우 (대부분)

⇒ 문제 발생 (이해는 안되지만 AI가 코딩을 다 해주니 AI에 의존하게 됨)

다만 **이해를 하면서** AI를 적절하게 사용하면 됨 (AI를 사용하지 말라는 소리가 아님)



너가 작성해준 코드에서 @가 들어간 문법을 사용하였는데, 이건 뭐고 왜 쓰는거지?



파이썬에서 "@가 들어간 문법"은 데코레이터를 의미하며, 이는 ...(생략)

" AI가 작성해준 코드의 로직이나 의도/의미 등을 이해하는 것이 중요 "

AI가 아닌 선생님이 알려주시는 "수업"에서도 해당됨

Session 6. AI — AI를 통한 바이브 코딩, 한가지 습관만 바꾸면 실력으로 돌아온다.

계속 질문을 하게 될 경우 토큰이 소비되어 나중에 AI가 맥락(Context) 판단이 안되기 시작함

그런데 왜 아직 -1을 하는지 모르겠습니다. Node의 IP는 하나만 제외하면 되는거 아니니까?



VS VPC CNI의 ENI 구조와 **primary/secondary IP** 개념을 이해해야 완전히 납득됩니다

메시지 편집

그럴 땐 "메시지 편집" 등의 기능을 활용하면 됨 (지원하지 않는 경우도 있긴 함)

Session 6. 개발 — 구글은 모든 것을 알고있다. (트러블 슈팅과 공부 위한 올바른 검색법)



" 파이썬에서 At(@) 기호는 무슨 역할인가요? "



" 쿠버네티스의 장점이 뭐길래 사용하는건가요? "



사실 이런식의 검색은
찾고자하는 정보를 찾기 어려움

완성된 문법의 문장으로 검색하기 보단 "**키워드**"를 활용할 것



" 파이썬 At(@) 기호 역할 "



" 쿠버네티스 장점 및 사용 이유 "

Session 6. 개발 — 구글은 모든 것을 알고있다. (트러블 슈팅과 공부 위한 올바른 검색법)

더 나아가 한글이 아닌 **영어**로 검색, 국내 네이버가 아닌 **구글**을 이용

(네이버는 자료=데이터베이스가 구글에 비해 턱없이 부족함)



" python at symbol "



" kubernetes coredns crashloopbackoff "



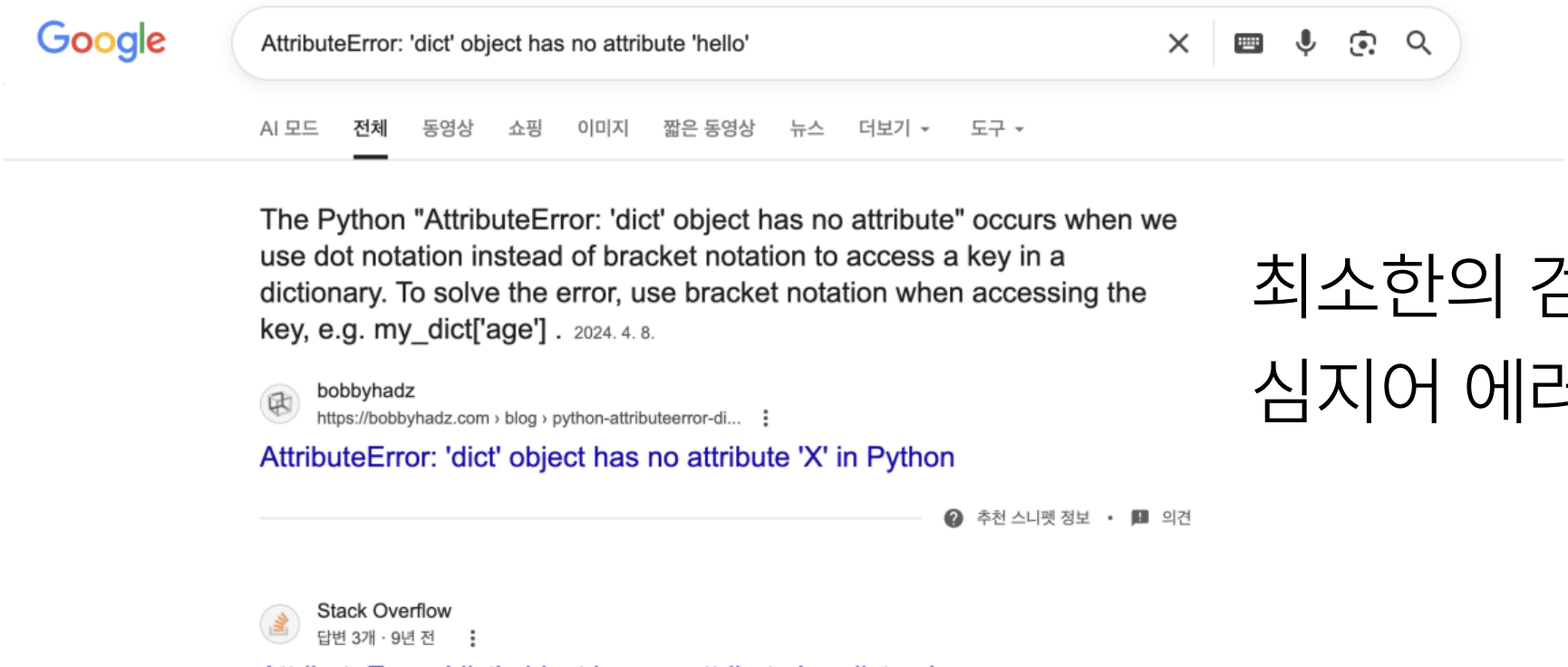
" java map class example "

등등..

Session 6. 개발 — 구글은 모든 것을 알고있다. (트러블 슈팅과 공부 위한 올바른 검색법)

```
> python3 test.py
Traceback (most recent call last):
  File "/Users/workspace5/sqltndjq/test.py", line 5, in <module>
    print(test.hello())
           ^^^^^^^^^
AttributeError: 'dict' object has no attribute 'hello'
```

에러가 발생한다면 그 에러 메시지를 구글에 복사한다면 95%의 확률로 해결 방법(트러블 슈팅)이 나옴
다짜고짜 AI에게 물어보는 습관 보단 검색부터 해보는 것이 오히려 더 유리함



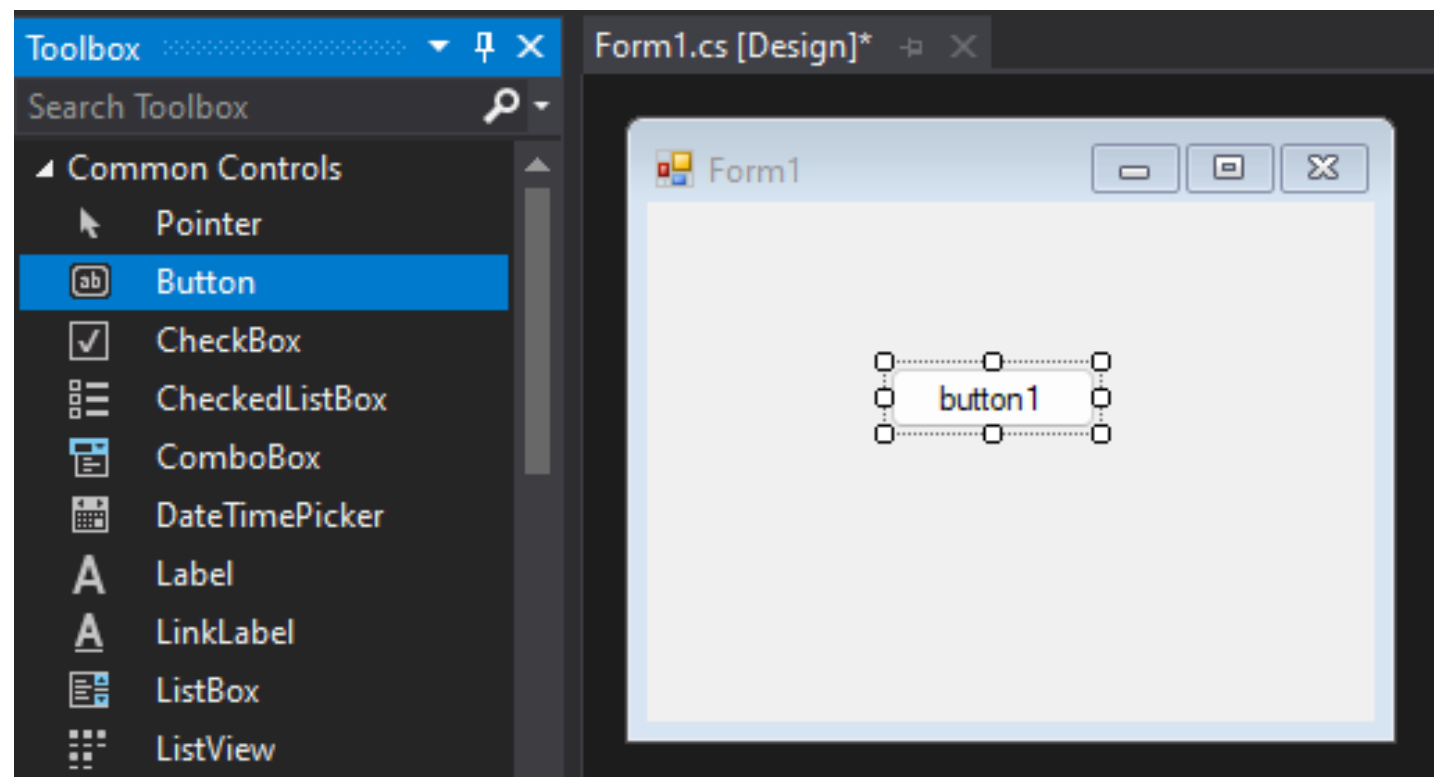
최소한의 검색도 안해보고 도와달라는 사람이 너무 많음...
심지어 에러 메시지도 안읽음.. (핑프)

Session 6. 개발 — 언어 하나를 깊게 파자.

때는 2018~2019년...



GUI 앱을 만들어보고 싶은데.. 어떻게 만들 수 있을까?



(지금은 WPF에 밀려 안쓰는 기술)

Windows Form

.NET C# 언어를 사용하는 GUI 프레임워크

⇒ 첫 프로그래밍 언어 " **C#** "

(마이크로소프트에서 자바 견제하려고 만든 객체지향 언어)

Session 6. 개발 — 언어 하나를 깊게 파자.

C# 이후 JavaScript/TypeScript/NodeJS만 주구장창 사용했음

⇒ " 파이썬, 자바 공부는 언제부터 했음? " (1학년때 부터 수행평가 때문에 단골 질문)

⇒ 따로 제대로 공부해본적은 없음 (안함)

						학년	학기	수 상 명	등급(위)
1	정보·통신	데이터베이스 프로그래밍	3	100/73.6(18.3)	A(35)	1	1	교과우수상(음악, 프로그래밍, 컴퓨터 보안)	
								미래인재 코딩 챌린지	은상(2위)
							2	미래인재 정보보안 챌린지	은상(2위)
1	정보·통신	응용 프로그래밍 개발	4	100/63.1(16.9)	A(35)	2	1	선행상	
1	정보·통신	응용 프로그래밍 화면 구현	4	100/77.4(15.6)	A(35)			교과우수상(음악, 프로그래밍, 정보 처리와 관리, 컴퓨터 보안)	
1	정보·통신	응용 프로그래밍 화면 구현	4	100/77.4(15.6)	A(35)			교과우수상(데이터베이스 프로그래밍, 응용 프로그래밍 개발, 응용 프로그래밍 화면 구현)	

" 그런데 어떻게 프로그래밍 수행평가 만점이 나왔냐..?? 학과 포털 털었음? "

Session 6. 개발 — 언어 하나를 깊게 파자.

... JavaScript/TypeScript/NodeJS를 사용했던 입장에서

 **Java** ⇒ JS/TS 모두 객체지향 언어라 어렵지 않았음

 **Python** ⇒ 들여쓰기 문법만 좀 익숙해지면 그 무엇보다 쉬운 언어

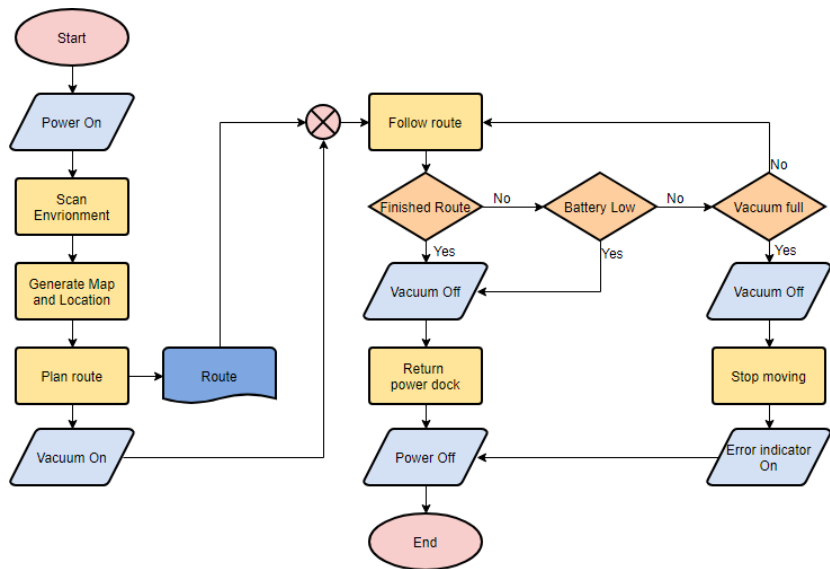
그 외에 C/C++, Rust, Go, Ruby 등의 언어도 특성(스마트포인터, GoRoutine 등)만 공부하면 쉽게 다루기 가능

1~2개의 언어(추천: Java, JavaScript, C/C++)를 깊게 공부하면 다른 언어는 쉽게 배울 수 있음

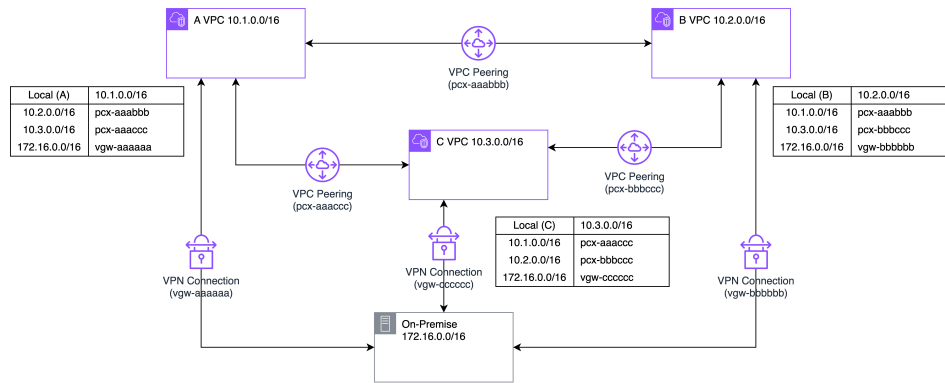
중요한건 그 언어들을 통해 **논리적으로 코드를 작성하는 것**

Session 6. 개발 — 코딩은 외우는게 아니다, 읽고 로직으로 접근하자.

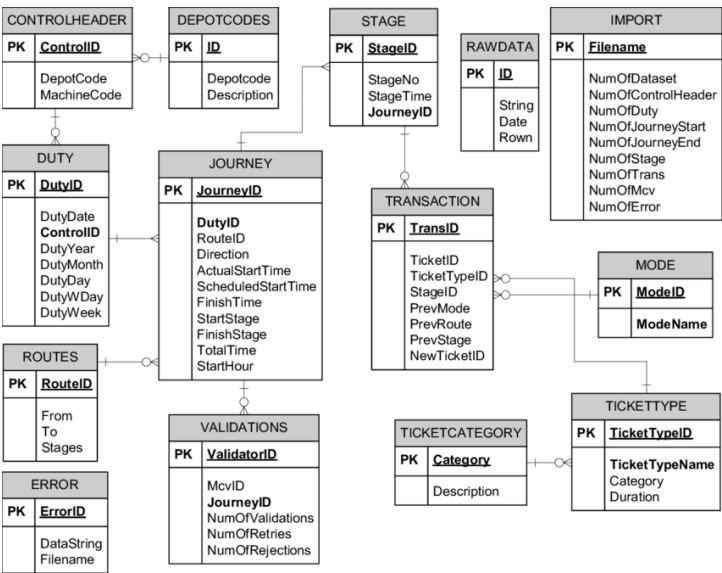
계속 "논리(로직)"을 강조하는 이유 ? ⇒ 그만큼 중요하기 때문



Flow Chart?



Architecture Diagram/Topology?



DB ERD?

```
103 func getFirstFoundEnvTestBinaryDir() string {
104     basePath := filepath.Join(".", "..", "bin", "k8s")
105     entries, err := os.ReadDir(basePath)
106     if err != nil {
107         logf.Log.Error(err, "Failed to read directory", "path", basePath)
108         return ""
109     }
110     for _, entry := range entries {
111         if entry.IsDir() {
112             return filepath.Join(basePath, entry.Name())
113         }
114     }
115     return ""
116 }
```

Programming?

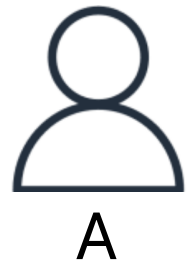
⇒ 이해하고 다루기 위해선 논리력이 필요

이런건 (대부분) 절대 외워서 하는 것이 아님

Session 6. 개발 — 인간은 도구를 사용하는 동물이다.



" ... 너는 왜 Visual Studio Code에 언어 확장 안쓰고 맨땅에 코딩을 하나? " (설치해줌)

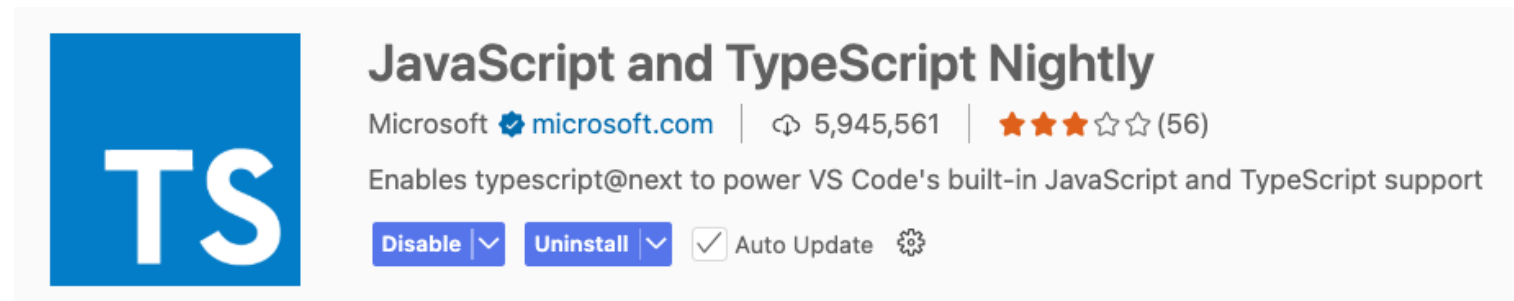


..!!! " 이걸 왜 지금 알려줌?? " (에러 바로바로 나오고 어떤 변수/함수 쓸 수 있는 전부 나옴)

IDE (통합 개발 환경, Integrated Development Environment)

Visual Studio, JetBrains IntelliJ 등

Visual Studio Code는 IDE가 아닌 텍스트 에디터지만
확장(Extensions)을 통해 IDE 처럼 사용 가능



(자세한건 직접 사용해보는걸 추천)

Session 6. 개발 — 개발에서 가장 어려운 것은 최적화이다.

블로그(Velog) 탐색 도중 굉장히 흥미롭게 본 포스팅... (원본 글을 못찾음)

(4초가 걸리던 DB 쿼리를 최적화하여 0.01초로 줄였다는 후기의 글)

⇒ N+1 문제와 같은 단순한 것 외에도 스키마 개편, 정규화 등등...

(기업에서 왜 야근하면서 최적화를 목숨걸고 하는지 알게된 글이었음)

최적화를 위해선?

알고리즘부터 시작하여 전체적인 구조/코드베이스 이해, 기술의 퍼포먼스 등 자세하게 알 필요가 있음

코드를 빠르고 안정성있게 작성하는 것도 중요하지만 기업에선 "**최적화 능력**"도 요구한다..

Session 6. 개발 — 버전 관리는 선택이 아닌 필수다.

 main_v2_final.py	오늘 오전 8:21	844바이트	Python 스크립트
 main_v2_real_final_last.py	오늘 오전 8:22	1KB	Python 스크립트
 main_v2_real_final.py	오늘 오전 8:22	680바이트	Python 스크립트
 main_v2.py	오늘 오전 8:21	510바이트	Python 스크립트
 main_v3_dev.py	오늘 오전 8:22	1KB	Python 스크립트
 main.py	오늘 오전 8:21	170바이트	Python 스크립트

⇒ 버전 관리 필요



커밋(변경 사항을 저장)

브랜치(개발 라인 분리: Dev, Prod...)

병합(Merge) 등등..

⇒ 모르면 협업 절대 불가능



깃 레포지토리 저장소 (또는 GitLab)

(깃허브는 Github Actions라는 CI/CD 도구까지 지원)

Session 6. 개발 — 버전 관리는 선택이 아닌 필수다.

Commits on Oct 23, 2025	
Deprecate esModuleInterop and allowSyntheticDefaultImports (default to true) (#62567) andrewbranch authored 5 days ago · ✓ 30 / 31	Verified f450c1b <>
Fix TS2783 false positive for union types in object spread expressions (#62656) 🗨 Copilot and RyanCavanaugh authored 5 days ago · ✓ 30 / 31	Verified 542b095 <>
Fixed a crash when parsing invalid decorator on await expression (#62659) Andarist authored 5 days ago · ✓ 30 / 31	Verified cbc2059 <>
Commits on Oct 21, 2025	
add allowJs default value description (#62611) manbearwiz authored last week · ✓ 30 / 31	Verified af3a377 <>
Fix fourslash tests (#62641) andrewbranch authored last week · ✓ 30 / 31	Verified 84f4856 <>
Bump the github-actions group across 1 directory with 2 updates (#62632) 🗨 dependabot[bot] authored last week · ✗ 28 / 31	Verified 16a52ed <>
Commits on Oct 18, 2025	
Handle more child types when generating navigation tree items for export default (#62620) Andarist authored last week · ✓ 30 / 31	Verified d8aafb3 <>

이렇게 버전 관리를 하면 PullRequest, Branch Merge 등의 협업 시 매우 효율적
이전 버전(커밋)으로 되돌리기 가능 (revert) 등등...

개발 업계에서 깃을 모른다... ⇒ 취업 안할게요~

Session 6. 혜택 — 학생 혜택을 적극적으로 누리자. (Github Education)

Pro 최고 인기

무제한 코드 작성 및 더 많은 모델에 액세스할 수 있는 채팅.

\$10^{USD}
매월 또는 매년 \$100

30일 동안 무료 사용

^ Free에 포함된 사양:

- ✓ 무제한 에이전트 모드 및 GPT-5 mini와의 채팅¹
- ✓ 무제한 코드 작성
- ✓ 코드 검토, Claude Sonnet 4, GPT-5, Gemini 2.5 Pro 등 지원
- ✓ Copilot Free보다 6배 더 많은 프리미엄 요청으로 최신 모델 사용 및 더 많은 구매 옵션 제공²
- ✓ 코딩 에이전트

학생, 교사, 인기 있는 오픈 소스 프로젝트의 유지 담당자에 한하여 무료. [자세히 알아보기](#)

Github Copilot Pro (월 10달러)

⇒ GPT5 (Thinking 포함), Claude Sonnet 4, Gemini 2.5 Pro 등 포함
ChatGPT Plus(월 약 3만원) 살짝 아래 정도 되는 성능 (충분함)

" 사실 우리는 무료로 사용할 수 있음 "

(고등학생, 대학생 한정)

Session 6. 혜택 — 학생 혜택을 적극적으로 누리자. (Github Education)

Github Education — Student Pack 신청 시 Github Copilot Pro 외에도 다양한 혜택이 있음
(<https://education.github.com/pack> 참고)

Students

Join a community where learning meets doing, with free access to the same tools professional developers use, including GitHub Copilot Pro and Codespaces.

[Learn more >](#)



GitHub Education



Education Benefits

You have a current student coupon applied. Find more information on your benefits [here!](#)

[Start an application](#)

학생증 및 학교 이메일(smc.sen.hs.kr)만 있으면 신청 가능

Session 6. 혜택 — 전공동아리도 우리에게 제공된 혜택이다.

" 전공동아리, 보니까 하는 것도 없는 것 같은데 저녁 제공 외에 혜택이 있긴 한가요? "

동아리에 따라 AWS/DreamHack 계정, 장비 지원, 대회 지원, 전문가 특강 등등...



가장 중요한 것은 **선후배와의 접점** ⇒ 동아리 시간 외엔 교류 할 시간이 마땅치 않음

(전공 활동 관련 1차적인 인맥을 쌓을 곳이 전공동아리가 거의 유일)

Session 6. 협업 — 클린 코드는 협업의 기본이다.

```
1 function a(x) {  
2     if (x) {  
3         var t = 0;  
4         for (var i = 0; i < x.length; i++) {  
5             if (x[i] != null && x[i] != undefined) {  
6                 t = t + x[i];  
7             }  
8         }  
9         return t;  
10    } else {  
11        return null;  
12    }  
13 }  
14  
15 var r = a([1,2,3,4,null,undefined,5])  
16 console.log(r);
```

- 함수/변수 명 a, x, t ... 등 의미를 알 수 없음
- if 조건 중첩
- var 사용 (스코프 문제가 있을 수 있음)
- 코드 스타일(들여쓰기 등)이 일관성 없음

협업, 리팩토링(코드 개선) 시 읽기/해석의 어려움, 버그 발생 가능성 높음

나중에 다시보면 본인도 뭘 한건지 모름

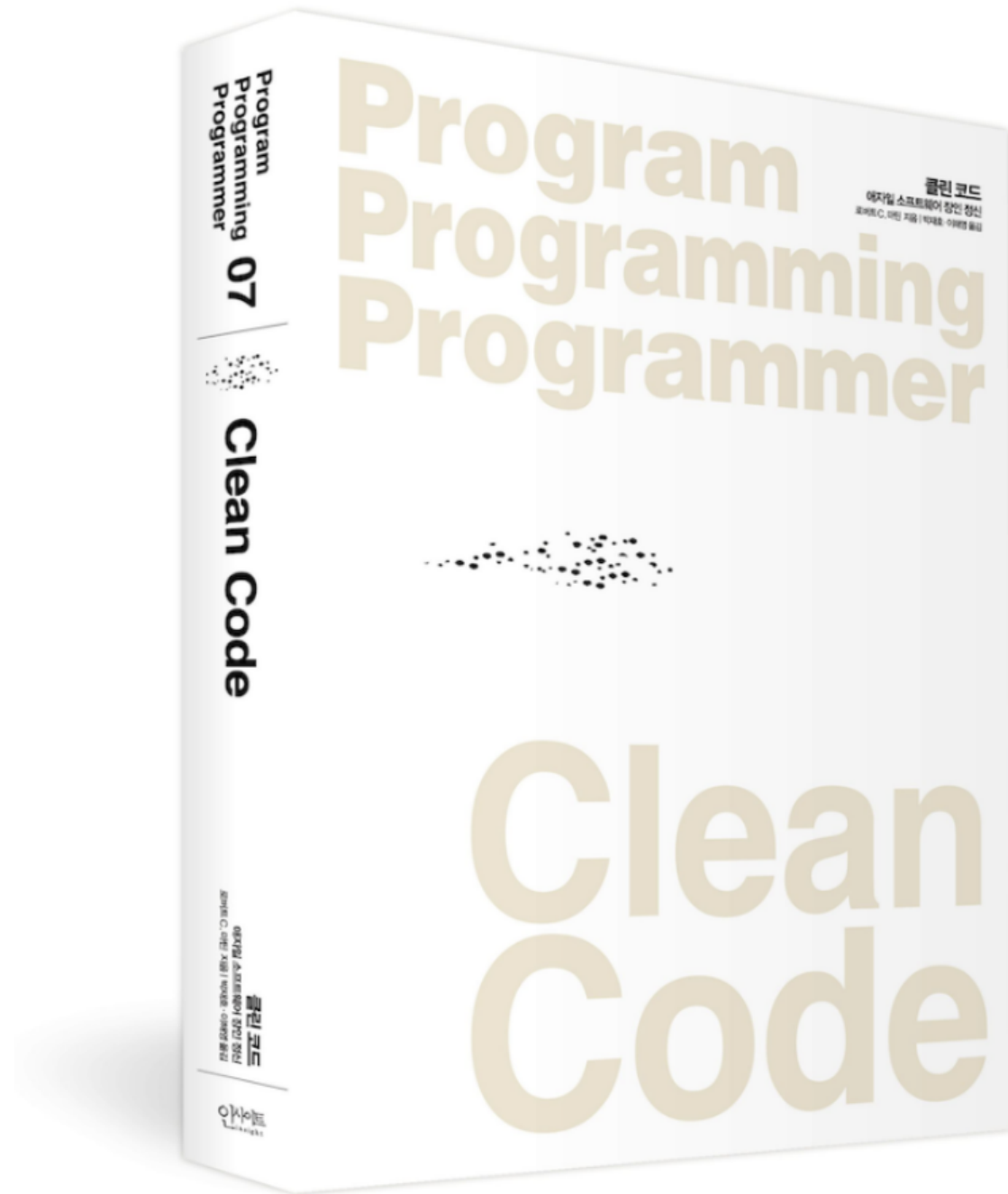
Session 6. 협업 — 클린 코드는 협업의 기본이다.

```
1  const sumNumbers = (numbers) => {  
2    if (!Array.isArray(numbers)) return null;  
3  
4    return numbers  
5      .filter((n) => typeof n === 'number')  
6      .reduce((acc, n) => acc + n, 0);  
7  };  
8  
9  const result = sumNumbers([1, 2, 3, 4, null, undefined, 5]);  
10 console.log(result);
```

- 명확한 함수/변수 명
- 맨 앞단에 Guard로 매개변수 유효성 체크
- const, let 등의 모던한 문법 사용(var 사용 자제)
- 함수형 요소(filter, reduce 등) 사용
- 일관성 있는 코드 스타일(Prettier 등 포맷터 활용)

당장 코딩을 할 때 클린하게 작성하지 않아도
나중에 리팩토링 시 **클린 코드로** 리팩토링하는 것을 추천

Session 6. 협업 — 클린 코드는 협업의 기본이다.



(객체지향 원칙 자세히 다룸, 개발에 관심이 많다면 한번쯤 읽어보시길 추천)

⇒ 물론 중간중간 애매하거나 "이건 좀.." 하는 내용도 있으니 참고만 할 것

프로그래밍에 클린 코드가 있다면 레이어 분리(CI/CD, 인프라, 구조론 등)에
중점을 둔 "클린 아키텍처"도 추천

Session 6. 협업 — 개발자의 능력은 개발 능력과 커뮤니케이션 능력으로 나뉜다.



개발 팀장

" 김OO 사원, 이번에 작성했던 비즈니스 로직 훌륭하던데, 다음번 발표 때 소개하면 좋을 것 같아. "



김OO 사원

" ㄴ..네... 알겠습니다...다.."



김OO 사원

" 제가 작성했던게.. 그러니깐.. 어... (...)"

... ' 김OO 사원은 실력은 대단한데 커뮤니케이션 능력이 참 아쉬워... '

단순히 개발을 넘어, 협업에서 정보 전달, 조율, 논의/미팅 등의
사회성을 요구하는 **커뮤니케이션 능력** 또한 매우 중요함

Session 6. 협업 — 인맥을 적극적으로 활용하자.



' 작성했던 코드에 대해 코드 리뷰를 받고 싶은데, 개발에 관심있는 지인이 없네.. '



' 이번 대회에 꼭 참여하고 입상하고 싶은데, 같이 할 사람이 없어서 못하겠어.. '



' 취업/진학에 관련하여 피드백이나 조언을 듣고 싶은데, 인맥이 부족해서 어려워.. '

인맥은 학교 동문/선후배, 직장 동료/선후배 외에도 외부 컨퍼런스/세미나

또는 온라인(Linkdin, 블로그, 커뮤니티 등)에서도 만들 수 있음

⇒ 현재 가장 쉬운 방법은 전공 동아리에서 친구, 또는 선후배와의 인맥 쌓기

Session 6. 협업 — 온라인 개발자 커뮤니티도 인맥을 만들 수 있다.

오프라인에서 인맥을 쌓기가 꺼려지는 상황이라면...



등의 온라인 커뮤니티를 활용하는 것도 하나의 방법

(단, 온라인 익명성의 특성상 조심해야 할 부분도 있긴 함)

그래도 가장 좋은 것은 오프라인에서 인맥을 만드는 것 (온라인은 잘못된 길로 빠질 가능성이 있음)

Session Ending

" 경험을 압축하는 알고리즘은 존재하지 않는다 " — AWS

무엇이든 경험이 절대적으로 중요함 = 시작, 시도해보지 않으면 경험은 존재하지 않음

일단 시도, 그리고 최소한의 노력이라도 해보는 것이 전공 실력을 키우는데 유일한 진입점

Session Ending



Korea Cloud Native Community for Students

Cloud Native: Changing the paradigm of the internet.

📍 Korea, South ✉ normal8781@gmail.com

We are not bound to any specific cloud service vendor and pursue all technologies related to **Cloud Native** and **Cloud Computing**.

"**학생**" 중심의 클라우드, DevOps 커뮤니티/스터디 그룹 기획 중 (장기 프로젝트, 인원 모집이 가장 큰 문제..)

⇒ 정기 모임, 세미나 발표, 프로젝트 공유, 상호 간 피드백, 커뮤니케이션 등등...

⇒ 특정 학교/학과에 종속되지 않고 열정있는 **학생**이라면 누구나 참여 가능, 수평적인 조직 문화 지향

비주류 경향으로 인해 이루어질 가능성은 현실적으로 높지 않지만..

관심이 있으시다면 언제든지 연락 주시면 감사드리겠습니다.

Cloud Native: Changing the paradigm of the internet.

THANKS FOR YOUR ATTENTION

Speaker / Presenter by Kim Jun Young

Contact 010-2980-1336



개별/익명 Q&A (질문) 양식