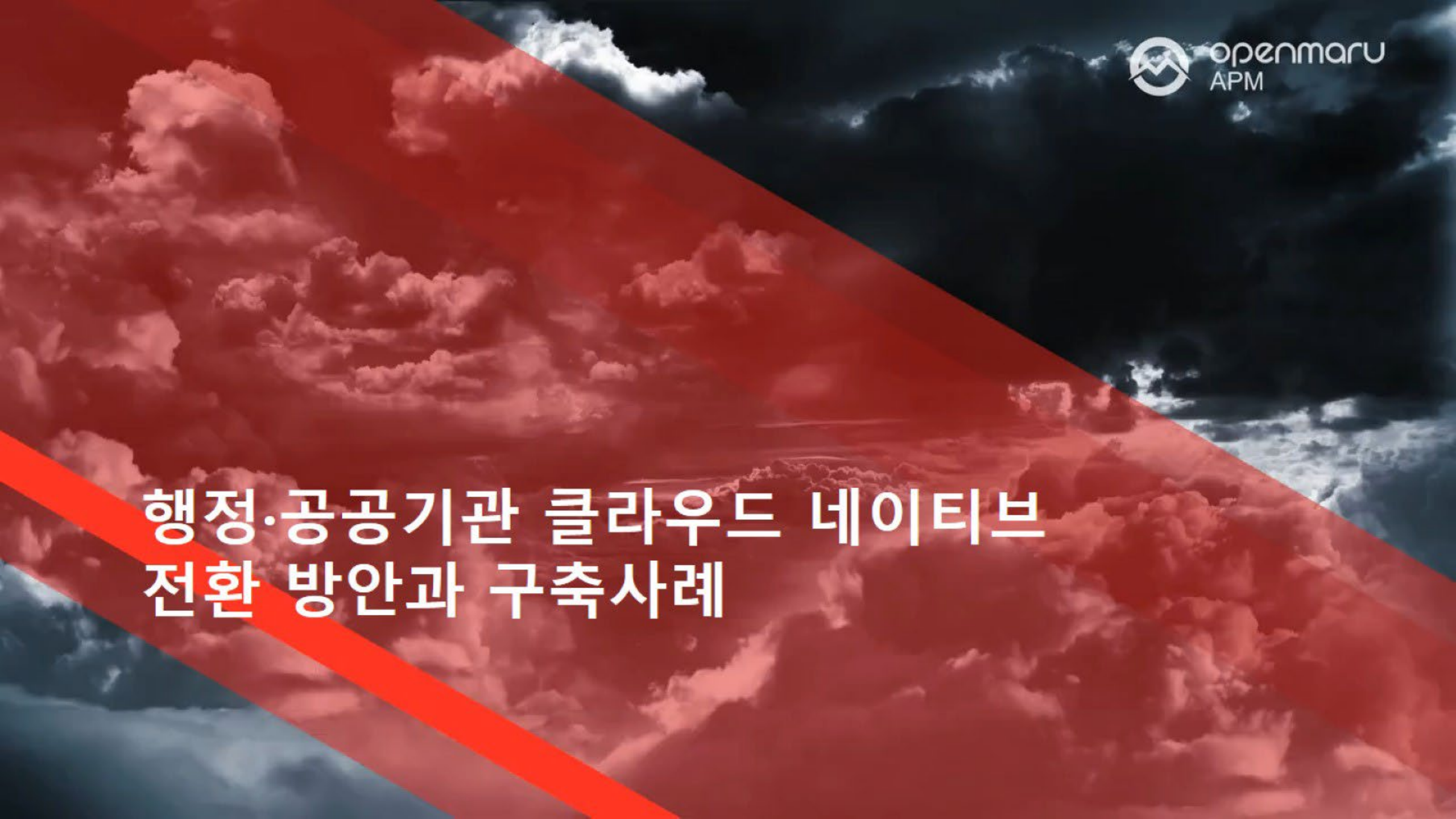

openmaru
APM
행정·공공기관 클라우드 네이티브
전환 방안과 구축사례

디지털플랫폼정부 인프라 구축

4 디지털플랫폼정부 인프라 구축

4-1 공공부문 클라우드 정착

◆ **행정·공공기관의 정보시스템을 클라우드로 본격 전환하여 디지털플랫폼정부 서비스를 신속·유연하게 안정적으로 제공**

○ 각 기관이 **개별적으로 운영 중인 정보시스템을 클라우드로 본격 전환**

※ '23년 공공기관 대상 활용모델 지원 → ~'27년 행정·공공기관 활용모델 확대

- (**다양한 활용모델***) 유사한 영역의 **업무군별 활용모델 확대** 및 **기관의 특성과 여건을 고려한 다양한 형태의 활용모델 적용 지원**

  * 2개 이상의 클라우드 서비스를 동시에 활용하는 멀티클라우드, 하이브리드 클라우드 등

  ※ 지자체 활용모델 시범사업 : 클라우드서비스 제공기업(CSP)과 경남, 세종, 제주 등 지자체가 협력하여 광역지자체 단위로 지자체 정보시스템의 클라우드 전환 추진 중

- (**클라우드 최적화**) **MSA*** 기법의 도입 효과를 극대화하는 업무를 중심으로 **클라우드 네이티브 구조의 설계**를 적용하여 **단계적 전환**

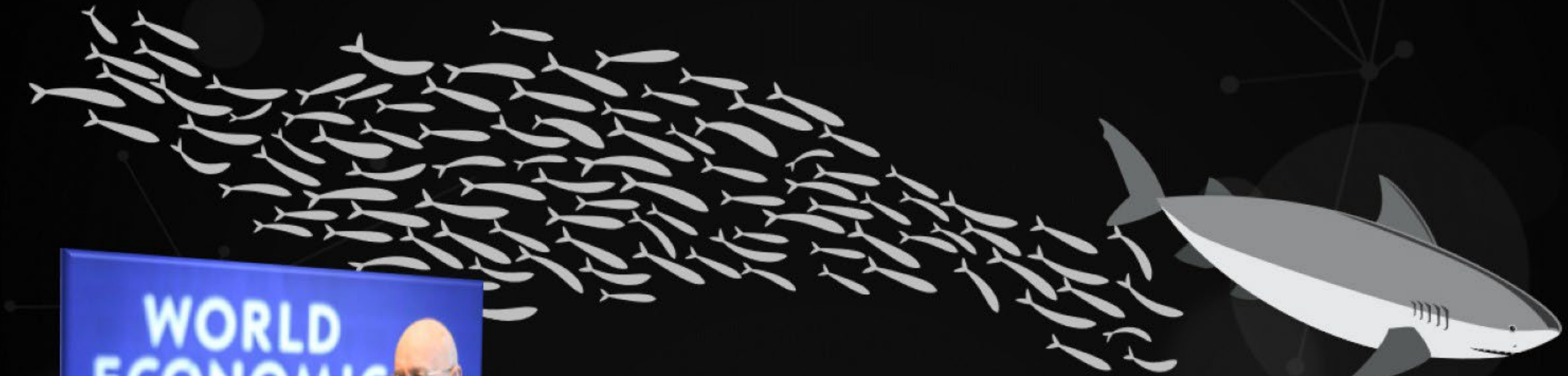
  * MicroService Architecture : 하나의 큰 애플리케이션을 여러 개의 작은 애플리케이션 단위로 쪼개어 손쉽게 변경·조합이 가능하도록 만든 아키텍처

  ※ 민간클라우드 전환을 위해 심층 컨설팅을 추진 중인 과기정통부 소관 시스템을 대상으로 선도적으로 시범 전환('23년)

행정안전부의 디지털플랫폼정부 추진계획(2023년~2027년)
- https://www.mois.go.kr/frt/sub/a06/b04/egovVision/screen.do

The fast fish eats the slow fish

"In the new world, it is not the **big fish** which eats the **small fish**,
it's the **fast fish** which eats the **slow fish**."



4 차 산업혁명의 이해 (Mastering the Fourth Industrial Revolution) – 세계 경제 포럼
Klaus Schwab, Founder and Executive Chairman of the World Economic Forum

openmaru APM

Business Disruptor

airbnb

- **자신의 방이나 집, 별장** 등 사람이 지낼 수 있는 모든 공간을 임대할 수 있으며, 192개국 3만 4800여 개 장소에서 200만여 개의 객실에 대한 숙박을 중개
- Airbnb는 **부동산**을 소유하지 않음

facebook

- 전 세계 14억 9천 만명 이상의 월 활동 사용자가 활동 중인 세계 최대의 **소셜 네트워크 서비스**
- **자체 생산 콘텐츠 없이도** 뉴스, 사진, 비디오 제공

UBER

- 승객이 스마트폰 앱을 이용해 차량을 호출하면 우버와 계약한 기사가 자기 차량을 몰고 와 목적지까지 데려다 주는 '**주문형 개인 기사 서비스**'
- **Uber는 보유한 차량이 없음**

Alibaba Group

- 중국의 전자상거래업체로 '세계에서 가장 큰 쇼핑몰' 서비스
- **Alibaba는** 보유한 **상품재고 없이** 전자상거래

Netflix 클라우드 네이티브 성공 사례

Netflix는 클라우드 네이티브 성공사례로,
전세계 확산 서비스를 제공하여 사용자에게 고화질 동영상 서비스를 안정적으로 제공

넷플릭스는 매년 전세계 인터넷 트래픽의
15% 이상을 차지하고 있음

2021년에는 이미 2억 400만명
(3년 만에 2배 증가)의 구독자가 생겼고,
매 4분기 마다 5백만명이
200개 이상의 국가에서 구독 中

NETFLIX
OSS
Netflix
Open SourceSoftware
Center

spring boot
+
NETFLIX

기술팀이 8년간 노력해온 결과

인프라를 자체 센터에서
Public 클라우드 이전

모놀리틱 프로그램을 작게 관
리할 수 있는 마이크로서비
스 아키텍처로 변경

애플리케이션 함수 실행 서비
스하는 서버리스 컴퓨팅 과 및
백엔드 아키텍처를 제공

클라우드 네이티브 국내·외 선도 도입 사례

선진 사례

AWS

빠른 배포 구현

수 천개 팀(자율적 DevOps팀) X 마이크로서비스 아키텍처
X 지속적 배포(CD) X 다양한 개발 환경

넷플릭스

가입자 대상서비스 확대

Netflix Open Source Software Center

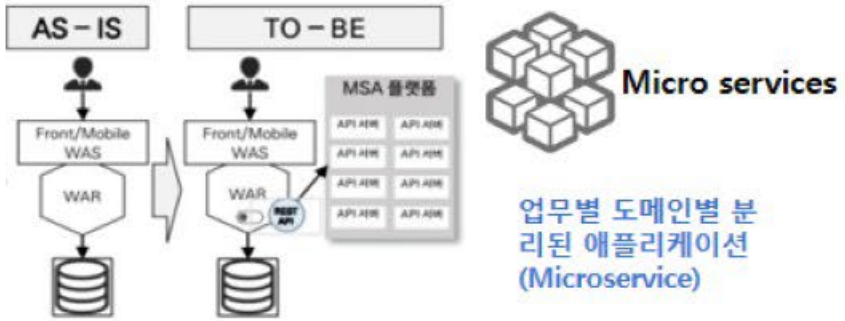
KAKAO

계열사 신규서비스 확대 및 빠른 출시 사례

카카오의 애자일 문화, 일하는 방식 관리를 위한
전담팀 및 개발플랫폼 운영

11번가

서비스 분리를 통한 점진적 MSA전환



openmaru

Development Process

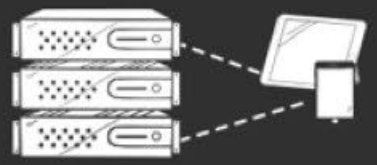
WATERFALL

AGILE

DEVOPS

Application Architecture

MONOLITHIC



N-TIER

MICROSERVICES

Deployment & Packaging

PHYSICAL SERVERS

VIRTUAL SERVERS

CONTAINERS

Application Infrastructure

DATA CENTER

HOSTED

CLOUD


클라우드 네이티브 개념과 기술요소들
Cloud Native

클라우드 네이티브를 저해하는 요인

- 생각과 시스템을 클라우드 네이티브 하게 전환하지 못하면, 클라우드라도 개선이 없음

클라우드를 임대하여 사용하는 것일 뿐

- 가상머신과 스토리지를 임대하여 사용하고 있을 뿐, 기존의 인프라와 다르지 않음

클라우드 특징에 맞게 설계하고 운영 하지 않음

- 클라우드 특성을 이해하지 못하고 기존 인프라를 단순히 대체하여 설계
- 비용 부분에서만 정액제 클라우드로 전환하였으나, 벤더 종속성과 비용만 높아짐

인프라만 클라우드 일 뿐 조직은 그대로

- 기존의 개발팀과 운영팀이 수행하던 역할과 프로세스 그대로 운영

클라우드로 전환했으나 구인난과 고비용 구조로 더 큰 문제

- 클라우드를 이용하고 있음에도 불구하고 수작업 프로세스에서 벗어나지 못함
- 운영 인력 부족과 업무 효율성을 개선하지 못함

클라우드 전환 클라우드 이민과 클라우드 네이티브

Cloud Immigrant
(클라우드 이민 단계)

Cloud Native
(클라우드 네이티브 단계)

기존 어플리케이션

AS-IS
시스템

클라우드 네이티브
어플리케이션

- 기존 애플리케이션 그대로
- 기존 배포 방법

어플리케이션

어플리케이션

어플리케이션실행영역

마이크로서비스

- 기존 애플리케이션 또는 MSA 로 전환
- 컨테이너 이미지 배포

- 기존 운영 환경과 방법 그대로
- 시스템 S/W 설치와 라이선스
- 비용과 빌링 방법 변경
- 가상화 이미지로 구성

미들웨어/
OS/DB

가상화

IaaS (인프라)

미들웨어/
OS/DB

HW

컨테이너

PaaS (플랫폼)

- 시스템 S/W 컨테이너 이미지제공 (번들, OS, Java, WAS)
- 하이퍼바이저와 Guest OS 불필요

Cloud Immigrant vs. Cloud Native

구분	Cloud Immigrant	Cloud Native
서비스 모델	가상화 기반 IaaS (Infrastructure As A Service)	컨테이너 기반 PaaS (Platform As A Service)
디자인	On Premise 에 구축된 시스템을 클라우드로 이전하여 운영	시작 단계부터 클라우드의 장점인 민첩성, 확장성 그리고 이동성을 최대한 활용할 수 있도록 설계
구현	특정 클라우드 벤더에 의존적인 설정이 있어 구축에 시간이 걸림	어떤 클라우드 환경에서도 빠르고 효율적으로 전환 (Portability)
확장성	애플리케이션 업데이트가 수작업이기 때문에 장시간의 다운타임일 필요하고 Scale In/Out 이 어려움	컨테이너와 MSA 기반으로 서비스에 영향을 주지 않고, 업데이트가 필요한 서비스만 변경할 수 있으며, 서비스 단위의 Scale In/out 지원
비용	애플리케이션이 커질 수록 인프라 비용이 상승	인프라 부분의 종속성이 없어 비용이 저렴
유지보수	버전관리, 설치 그리고 구성관리가 수작업이며 복잡함	CI (Continuous Integration) / CD (Continuous Delivery)

CNCF Cloud Native Definition v1.0

openmaru

클라우드 네이티브 기술을 사용하는 조직은 현대적인 퍼블릭, 프라이빗, 그리고 하이브리드 클라우드와 같이 동적인 환경에서 확장성 있는 애플리케이션을 만들고 운영할 수 있다.

컨테이너, 서비스 메시, 마이크로서비스, 불변의 인프라스트럭처, 그리고 선언적 API가 전형적인 접근 방식에 해당한다.

이 기술은 회복성이 있고, 관리 편의성을 제공하며, 가시성을 갖는 느슨하게 결합된 시스템을 가능하게 한다.

견고한 자동화와 함께 사용하면, 엔지니어는 영향이 큰 변경을 최소한의 노력으로 자주, 예측 가능하게 수행할 수 있다.

Cloud Native Computing Foundation은 **벤더 중립적인 오픈소스 프로젝트 생태계**를 육성하고 유지함으로써 해당 패러다임 채택을 촉진한다.

우리 재단은 최신 기술 수준의 패턴을 대중화하여 이런 혁신을 누구나 접근 가능하도록 한다.

Source:https://github.com/cncf/toc/blob/master/DEFINITION.md#%ED%95%9C%EA%B5%AD%EC%96%B4

Cloud Native Computing Foundation

- **Non-profit, part of the Linux Foundation; founded Dec 2015**

- **Platinum members:**

Alibaba Cloud, aws, cisco, Dell Technologies, docker, FUJITSU, Google Cloud, HUAWEI, IBM Cloud, intel, JD.COM, MESOSPHERE, Microsoft Azure, ORACLE, Pivotal, redhat, SAMSUNG SDS, SAP, vmware

Cloud Native ERA

- 애플리케이션을 실행하기 위한 최적의 인프라 최적 솔루션 중 하나
  - 컨테이너 기술
    - 애플리케이션이 동작하기 위한 운영 환경을 함께 패키징
    - 개발자를 위한 이미지 빌드/배포 용이성
    - 빠른 애플리케이션 실행과 낮은 오버헤드

Develper Experience장점

- Kubernetes(컨테이너 오케스트레이션)
  - 애플리케이션 실행을 먼저 생각할 때
  - 어떤 인프라를 만들지 주축으로 설계된 인프라 기반

Reconcilliation model의 정교함

Cloud Native

컨테이너 기술

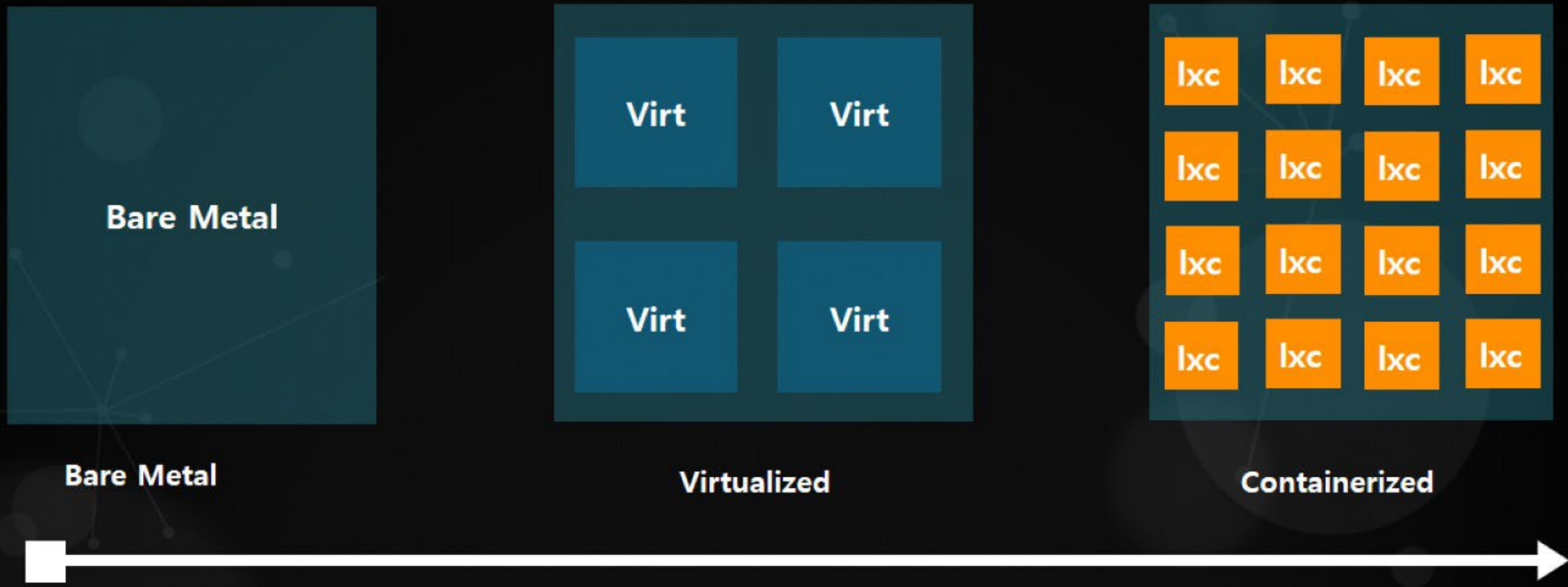
IT Infrastructure 에 대한 고민들

- IT 운영 부담을 줄이려고 하는 개발팀,

  시스템 운영팀, 네트워크 운영팀, DEVOPS 팀

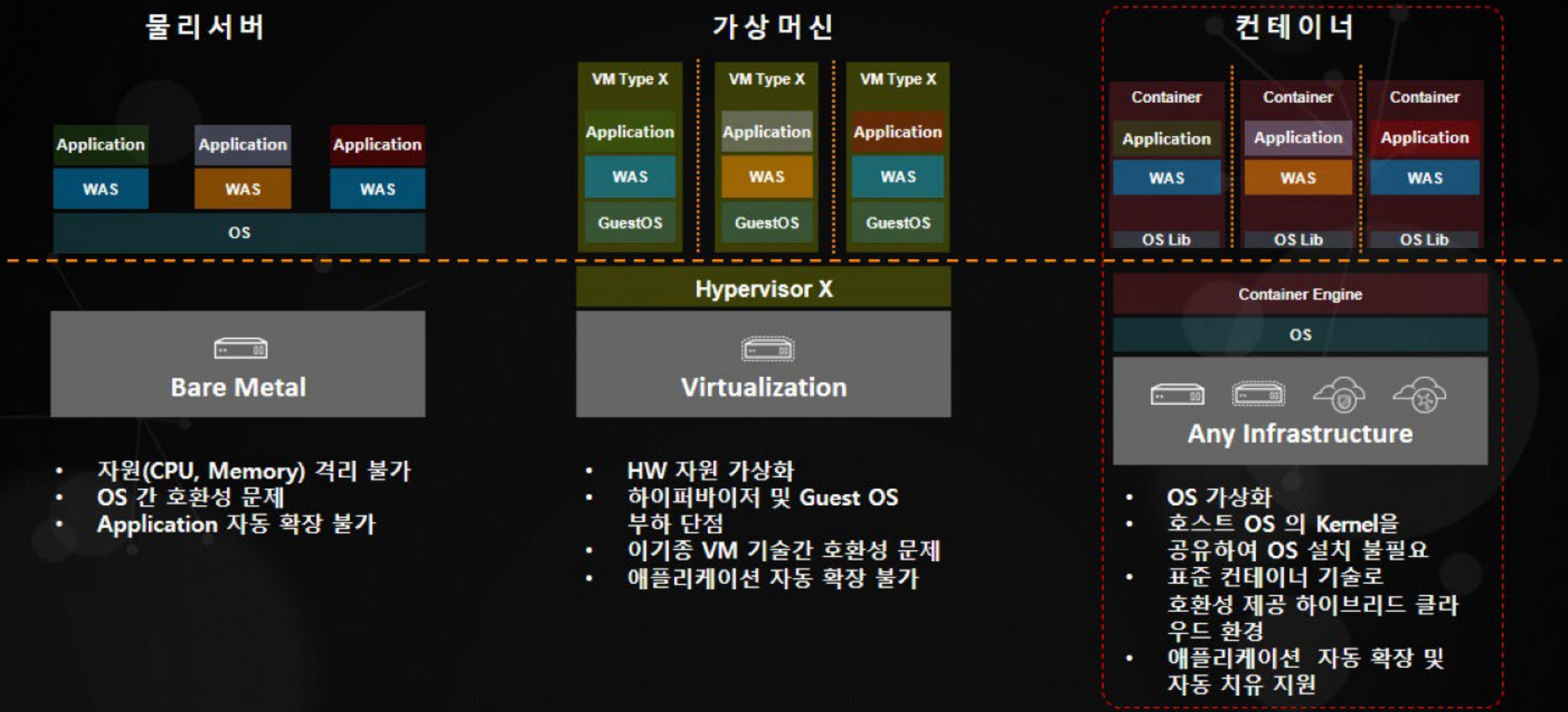
  - 애플리케이션 변경 요건이 많고,하루에도 여러 번 배포
  - 대규모 이벤트로 인한 부하에 대응할 수 있는 시스템 요구
  - 빈번한 설정 변경과 시스템 작업으로 인한 이력 관리
  - 복잡한 작업 절차에 대한 자동화
  - 비즈니스 요구에 적합한 배포 정책 요구와 롤백 방안

Evolution of Infrastructure Architectures



WHY CONTAINER ?

- 자원 효율성, 자원 격리, 호환성, Auto Scaling, DevOps, MSA, 관리 편의성



- 자원(CPU, Memory) 격리 불가
- OS 간 호환성 문제
- Application 자동 확장 불가

- HW 자원 가상화
- 하이퍼바이저 및 Guest OS 부하 단점
- 이기종 VM 기술간 호환성 문제
- 애플리케이션 자동 확장 불가

- OS 가상화
- 호스트 OS 의 Kernel을 공유하여 OS 설치 불필요
- 표준 컨테이너 기술로 호환성 제공 하이브리드 클라우드 환경
- 애플리케이션 자동 확장 및 자동 치유 지원

오버헤드 - Containers vs. VMs

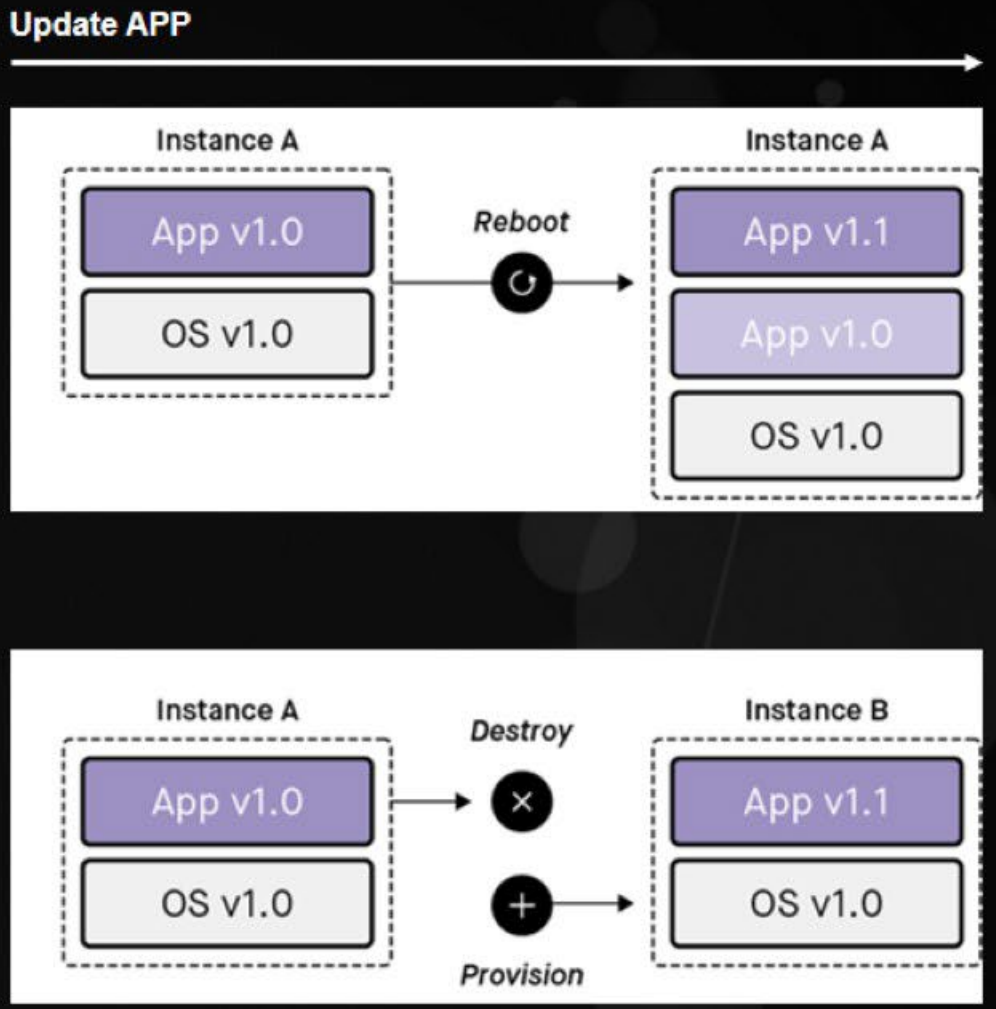
- OS에서 응용 프로그램을 작동하는 경우, 하드웨어 가상화에서는 가상화 된 하드웨어 및 하이퍼바이저를 통해 처리하기 때문에 물리적 시스템보다 처리에 부가적인 시간 (오버 헤드)가 필요
- 컨테이너 형 가상화 커널을 공유하고 개별 프로세스가 작업을 하는 것과 같은 정도의 시간 밖에 걸리지 않기 때문에 대부분 오버 헤드가 없음

Mutable vs. Immutable Infrastructure (가변 vs. 불변)

openmaru

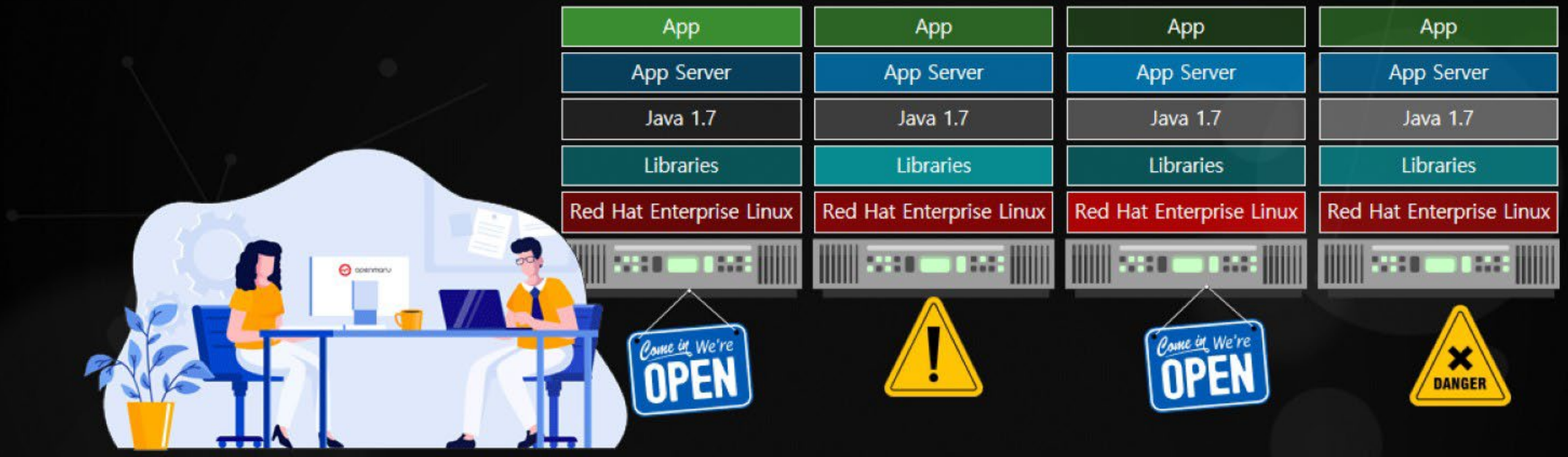
Mutable Infrastructure (물리서버, 가상화)

Immutable Infrastructure (컨테이너)



Configuration Drift

- 컨피그레이션 드리프트 (**Configuration Drift**)
  - 손으로 직접 수정한 임시 수정/업데이트와 전반적인 엔트로피(**entropy**) 증가로 인해 인프라의 서버들이 시간이 갈수록 점점 서로 다른 상태가 되는 현상
  - 장비의 라이프사이클 동안 초기 설정으로부터 멀어지고(**drift**) 다른 장비들 과도 서로 달라짐



Pets vs Cattle

openmaru

Pets

Cattle


Cloud Native 기술을 통한 Open Hybrid Cloud 구현
Kubernetes
& Cloud Native
openmaru
APM

GOOGLE 과 컨테이너

- **Google의 업무 방식**

Gmail에서 YouTube, 검색에 이르기까지 Google의 모든 제품은 컨테이너에서 실행됩니다.

개발팀은 컨테이너화를 통해 더욱 신속하게 움직이고, 효율적으로 소프트웨어를 배포하며 전례 없는 수준의 확장성을 확보할 수 있게 되었습니다. **Google은 매주 수십억 개가 넘는 컨테이너를 생성**합니다. **지난 10여 년간 프로덕션 환경에서 컨테이너화된 워크로드를 실행하는 방법에 관해 많은 경험**을 쌓으면서 Google은 커뮤니티에 계속 이 지식을 공유해 왔습니다.

초창기에 cgroup 기능을 Linux 커널에 제공한 것부터 **내부 도구의 설계 소스를 Kubernetes 프로젝트로 공개**한 것까지 공유의 사례는 다양합니다. 그리고 이 전문 지식을 Google Cloud Platform으로 구현하여 개발자와 크고 작은 규모의 회사가 최신의 컨테이너 혁신 기술을 쉽게 활용할 수 있도록 하였습니다.

Source - https://cloud.google.com/containers/

About Kubernetes

- 쿠버네티스(K8s)는 컨테이너화된 애플리케이션을 자동으로 배포, 스케일링 및 관리해주는 오픈소스 소프트웨어
- 쿠버네티스”, “쿠베르네테스”,“K8s”, “쿠베”, “쿠버”, “큐브”라고 부르며
- Go로 작성된 오픈 소스 , 오픈소스 S/W (Apache License 2.0) 라이선스
- 리눅스 재단 (Linux Foundation)산하 Cloud Native Computing Foundation (CNCF) 에서 관리
- 구글에서 개발하고 설계한 플랫폼으로서 사내에서 이용하던 컨테이너 클러스터 관리 도구인 “Borg”의 아이디어를 바탕으로 개발

> "Kubernetes is open source-a contrast to Borg and Omega, which were developed as purely Google-internal systems. "
>
> *- Borg, Omega, and Kubernetes*

컨테이너는 클라우드 에서 Java 와 같이 벤더 종속성 해제

openmaru

2000 년 – Java 를 통한 Vendor Lock-In 해제

OS	hp UX	AIX	solaris	Linux	Windows
CPU	Intanium	Power	Sparc	X86	X86
Vendor	hp	IBM	ORACLE	THE LINUX FOUNDATION	Microsoft

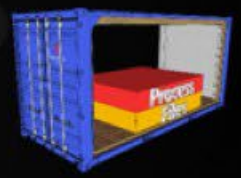
→ Java

2020 년 – 컨테이너와 Kubernetes 를 통한 Vendor Lock-In 해제

Write Container, run anywhere!

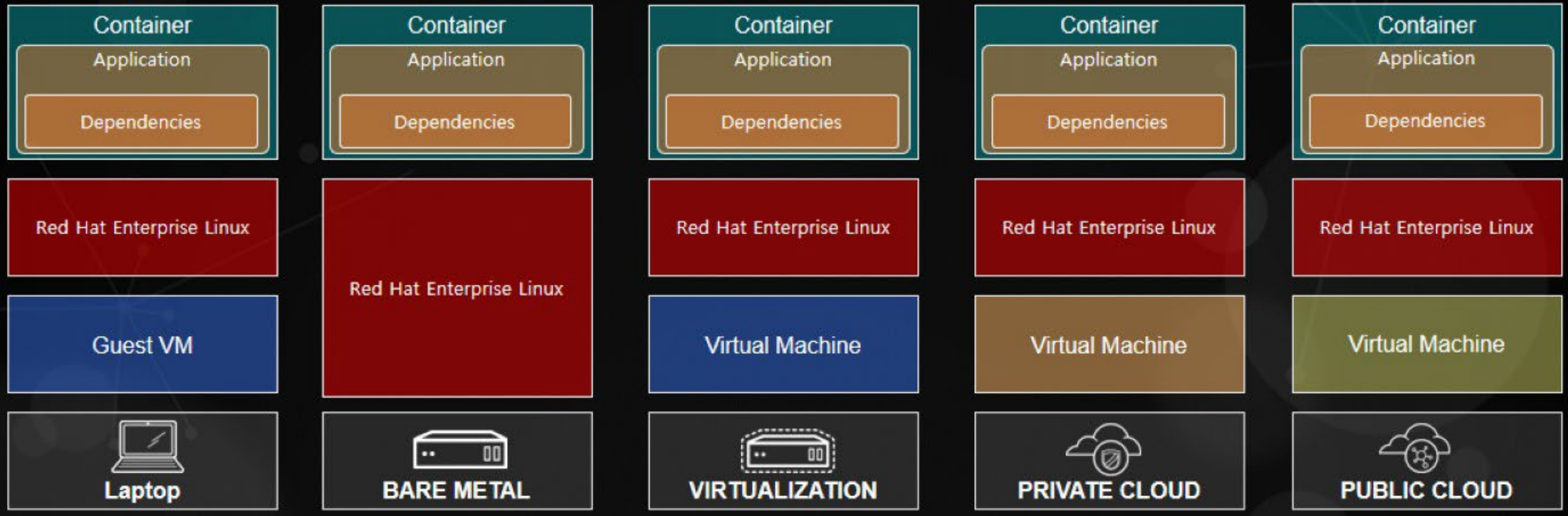
Hypervisor	Xen Project	KVM	Hyper-V	Xen Project	Xen Project
Public Cloud	aws	Google Cloud	Microsoft Azure	ORACLE	IBM Cloud

→ kubernetes



컨테이너의 이동성

- Linux 커널 구조를 이용하고있다 = Linux 움직이는 모든 환경에서 이동성



Open Source Software

"누군가가 나의 등잔의
심지에서 불을 붙여가도
내 등잔의 불은 여전히
빛나고 있습니다."

미국의 정치가 토머스 제퍼슨

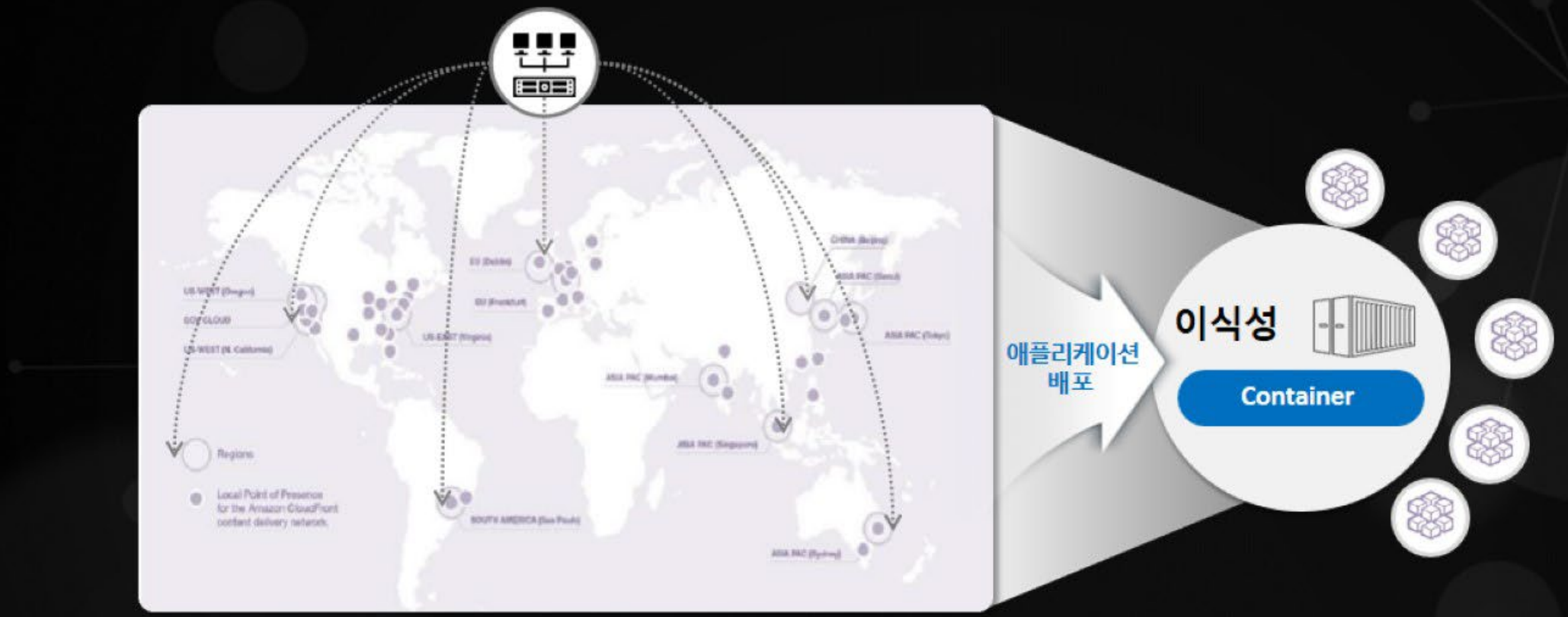
openmaru
APM

Cloud Native 기술을 통한 Open Hybrid Cloud 구현

데모로 이해하는 클라우드 네이티브
- 하이브리드 클라우드 데모

클라우드 네이티브 특징

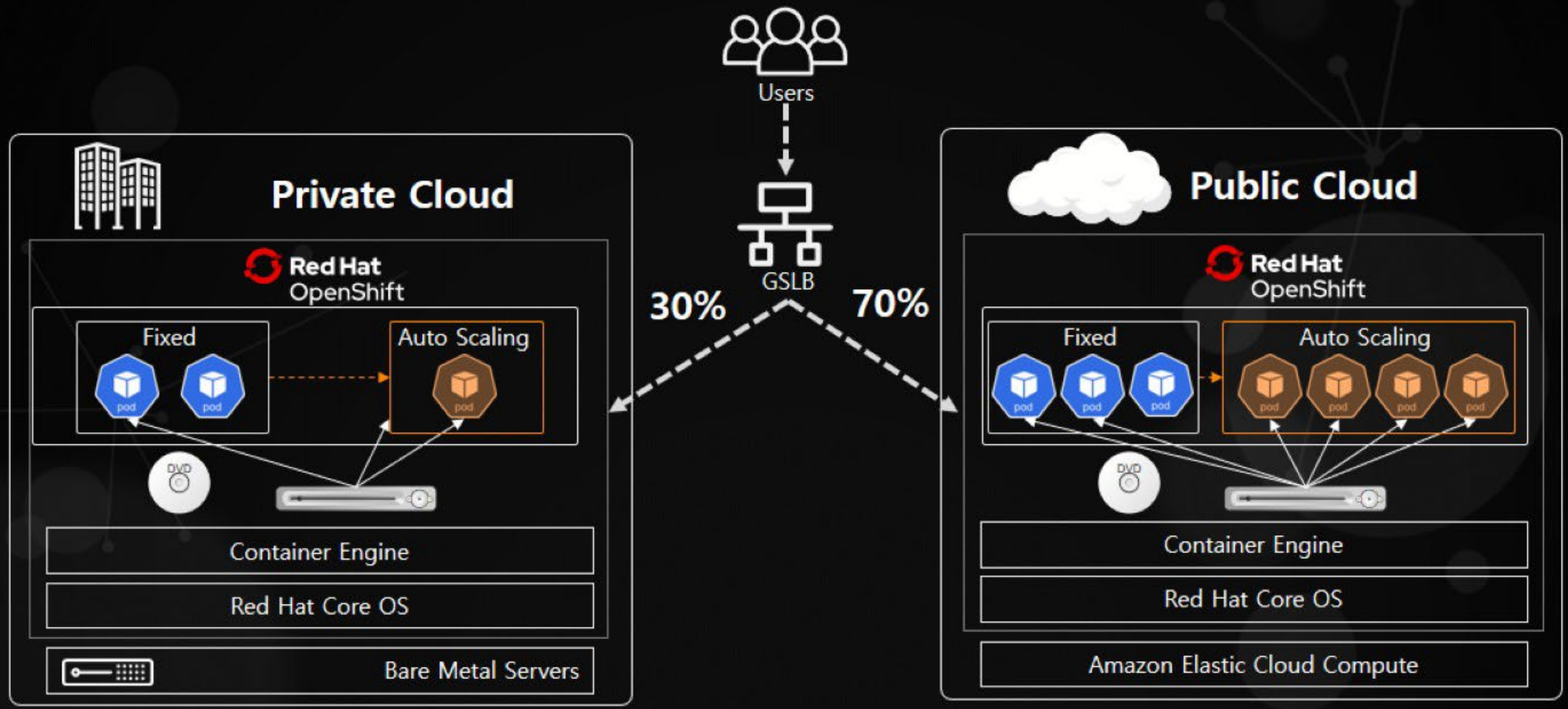
- 클라우드 네이티브는 작고, 가볍고 손쉽게 배포
- 클라우드 환경에서의 →서비스 배포는 전세계에 한번의 클릭으로 애플리케이션 배포 →전세계를 상대로 서비스 가능



Source : 클라우드 네이티브 추진 시 고려사항 (교육 교재)
클라우드 네이티브 기반 행정·공공 서비스 확산 지원 - 한국지능정보사회진흥원

하이브리드 클라우드 데모 – 사용자 증가 자동 확장

- 하이브리드 클라우드로 운영되는 포탈을 접속하는데 자동확장이 필요한 사용자 폭주상태를 가정
- 내부 클라우드과 외부 클라우드에 모두 동일한 홈페이지 서비스를 하고 프라이빗 30% vs. 퍼블릿 70% 로 운영 중



하이브리드 클라우드 데모 – 전자정부 F/W 포탈에 대한 글로벌 서버 부하 분산

- **GSLB (Global Server Load Balancing)을 통한 업무 부하 분산 데모**
- **전자정부 F/W 포탈 서비스를 내부와 외부 클라우드에서 동시 운영**

https://portal.egov.openmaru.io

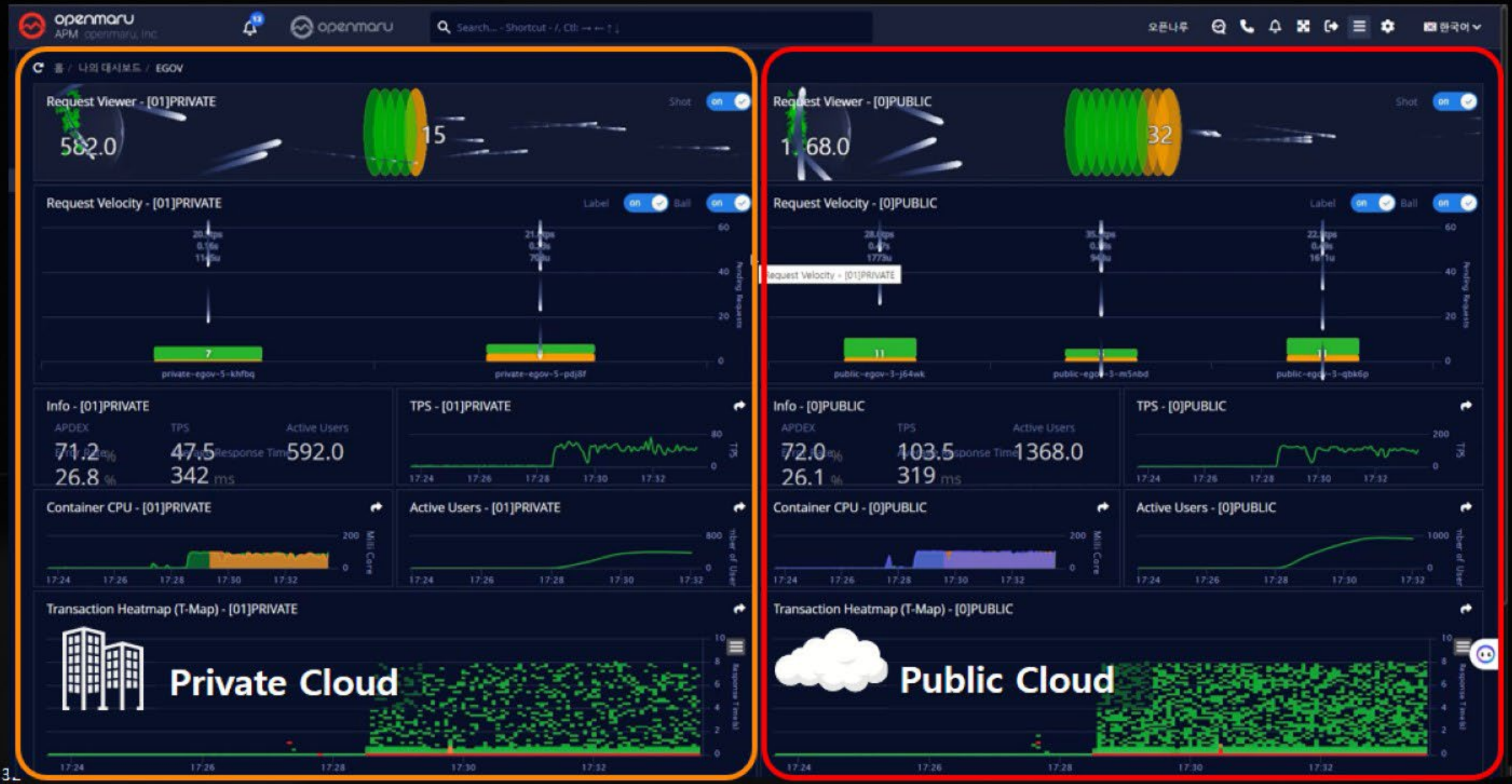
30%

70%

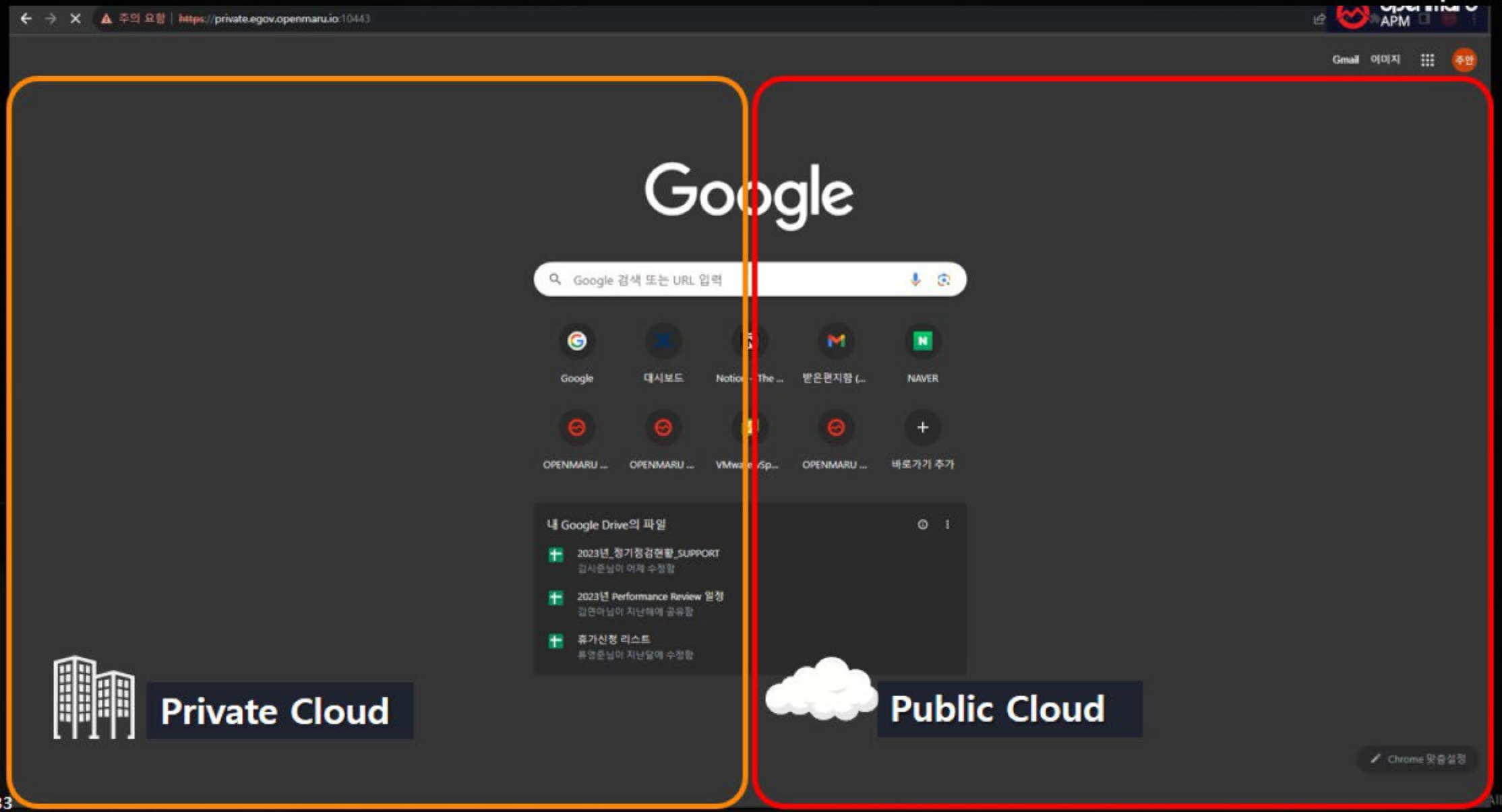
https://private.egov.openmaru.io

https://public.egov.openmaru.io

하이브리드 클라우드 데모 – 자동 확장 이전

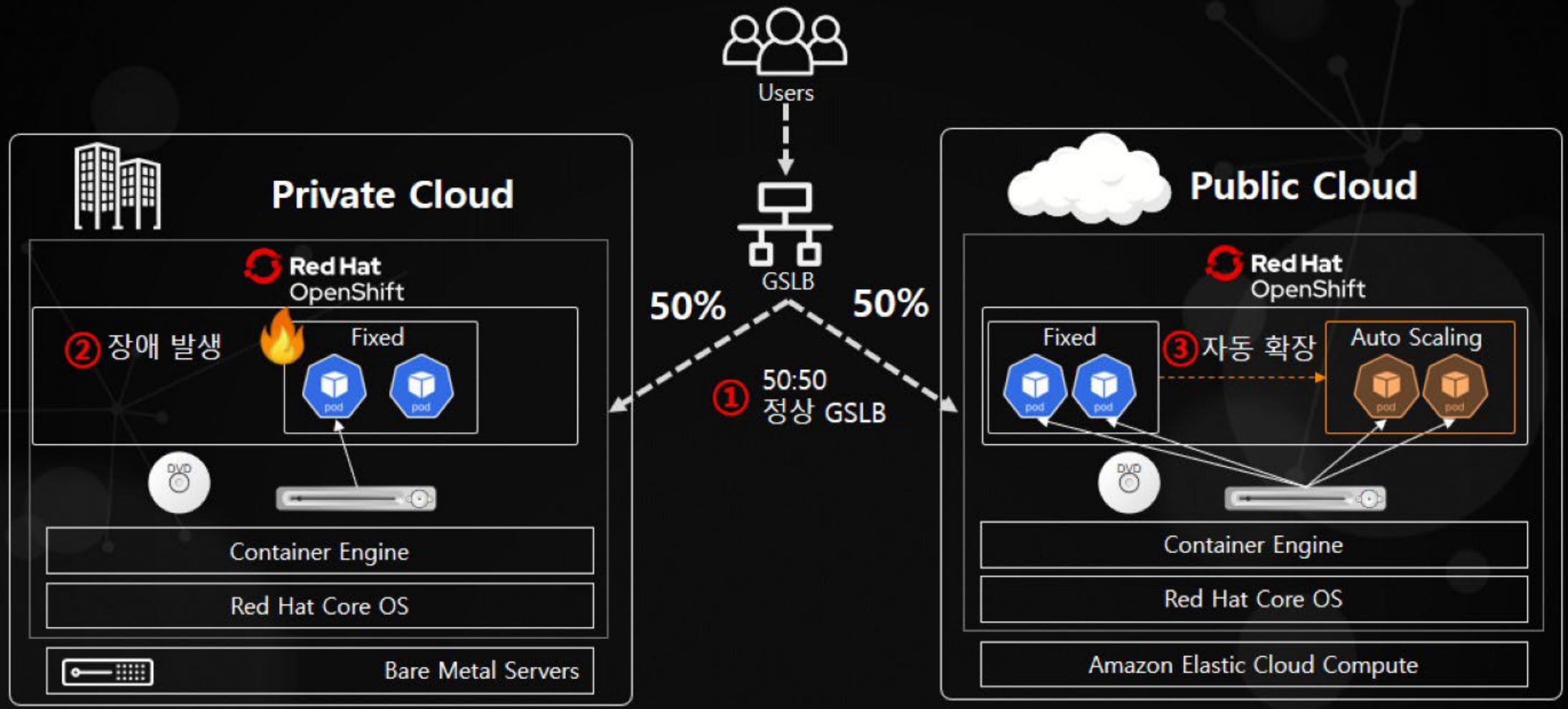


하이브리드 클라우드 데모 – 자동 확장 이후

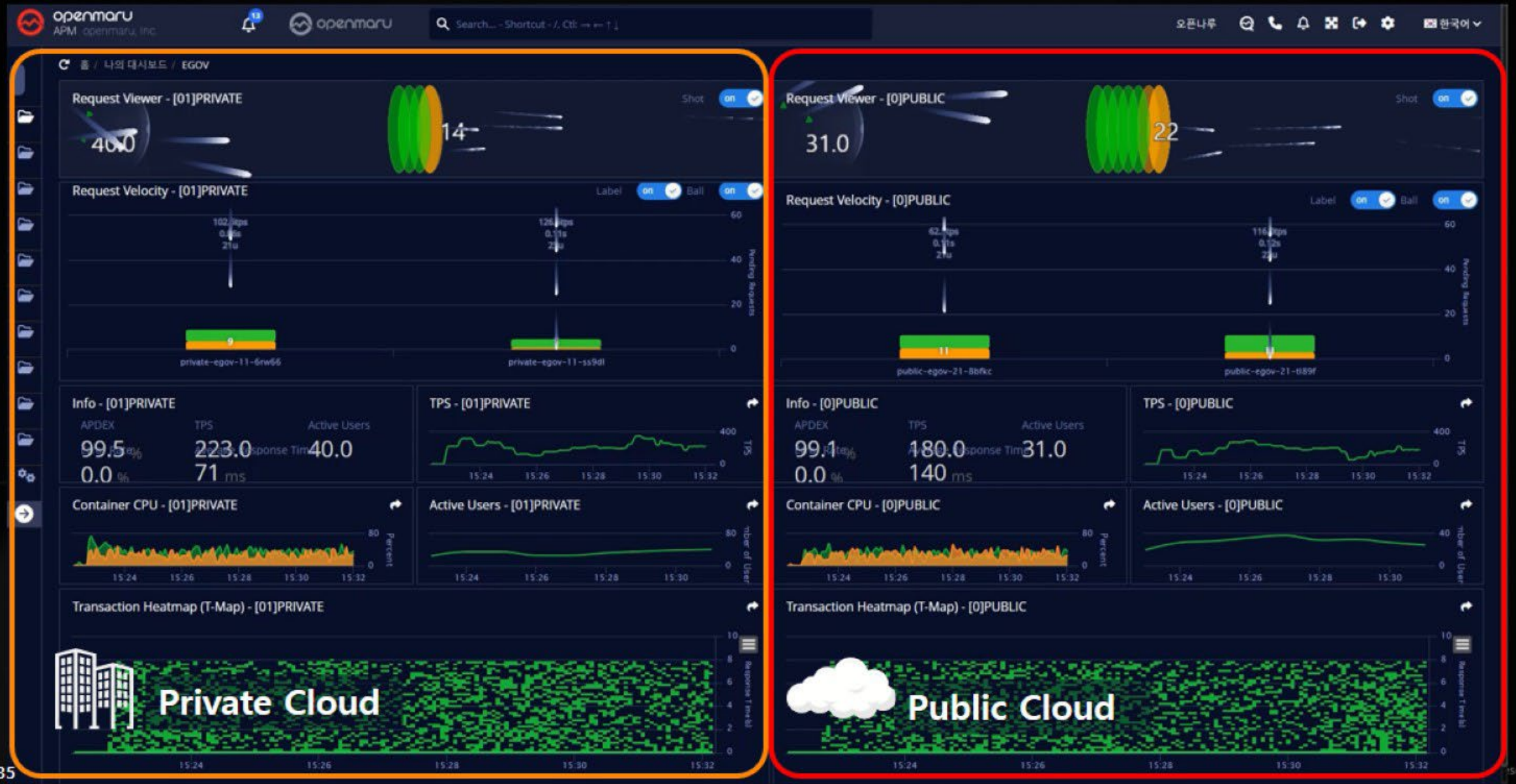


하이브리드 클라우드 데모 – Active Active DR

- 하이브리드 클라우드로 운영되는 포탈을 접속하는데 자동확장이 필요한 사용자 폭주상태를 가정
- 내부 클라우드과 외부 클라우드에 모두 동일한 홈페이지 서비스를 하고 프라이빗 30% vs. 퍼블릿 70% 로 운영 중



하이브리드 클라우드 데모 – 장애 발생 이전 Active-Active GSLB



하이브리드 클라우드 데모 – Public Cloud 에서 부하에 따른 자동확장

Cloud Native 기술을 통한 Open Hybrid Cloud 구현

어떤 공공업무에 클라우드 네이티브의 적용이 가능할까요?

클라우드 네이티브 대상 업무선정 방향 (전문가 의견)

클라우드 네이티브 업무는 학계, 업체, 정부정책을 반영하여 대상을 선정할 수 있습니다.

서비스 복잡도가 높은 시스템

- 마틴파울러(2015, 최초 용어정의) : "마이크로서비스는 복잡한 시스템에서 유용할 때 MSA전환"

+ 정부정책

명확한 경계가 가능한 시스템

- 샘뉴먼(2019, 저서) : "해당분야를 제대로 이해하지 못해 적절한 경계를 찾기 어렵다면 MSA전환 불리"

더 이상 확장할 수 없는 한계지점에 도달한 시스템

- 수잔파울러(2019, 저서) : "확장성 한계로 인해 심각한 안정 문제 발생하여, 개발생산성·효율성 저하 시 MSA 전환"

Source : 클라우드 네이티브 추진 시 고려사항 (교육 교재)
클라우드 네이티브 기반 행정·공공 서비스 확산 지원 - 한국지능정보사회진흥원

클라우드 네이티브 대상 업무선정 방향 (전문가 의견)

- 클라우드 네이티브 업무는 학계, 업체, 정부정책을 반영하여 대상을 선정할 수 있습니다.

정부내 전환가능 업무 식별

✔ SRM(서비스참조모델)의 전환가능 업무

정부내지원 서비스 →	업무변화 少 (적용 불리)
공통기술 서비스 →	시스템 연계 多 (적용 유리)
대국민 서비스 →	업무변화 多/ 폭주성 高 (적용 유리)

기관·시스템특성을 반영한 업무선정

✔ 시스템 특성 반영 (시스템복잡성)

규모 大 × 부서 多 × 시스템 메뉴 多

※국가및기초자치 단체226, 공공기관 338개기관업무대상

✔ 정부정책 특성 반영 (제도개선이많은업무)

A기관	B기관	C기관	D기관

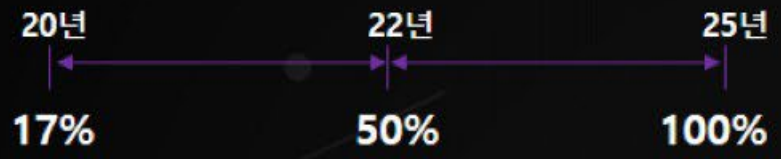
Source : 클라우드 네이티브 추진 시 고려사항 (교육 교재)
클라우드 네이티브 기반 행정·공공 서비스 확산 지원 - 한국지능정보사회진흥원

클라우드 네이티브 적용 검토

- 공공 클라우드 전면 전환에 따른 클라우드 네이티브 상호 운용성 확보하여 서비스 간 연결

공공클라우드 전면 전환 사업진행

✔ 공공기관 전면전환 비율 (목표)

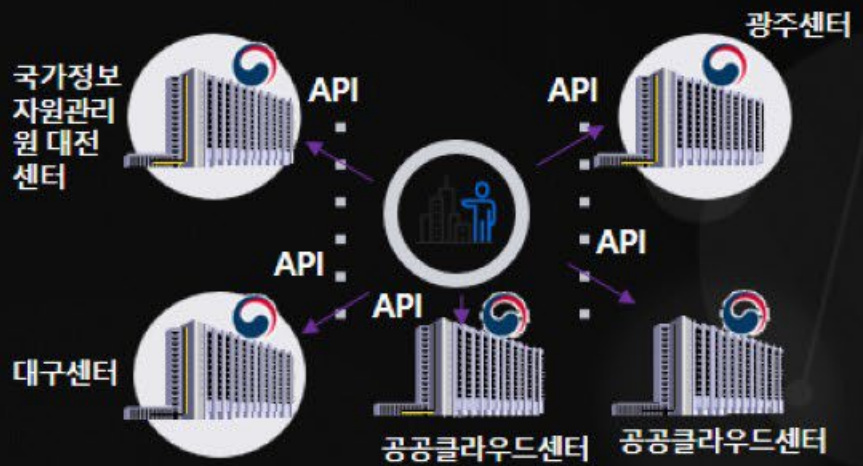


✔ 공공 (G-클라우드, 자체), 민간클라우드 센터

공공 클라우드센터	민간 클라우드센터
54%	46%

공공클라우드 센터 상호운용성 확보

✔ 디지털정부 서비스 개발환경인 클라우드 표준 플랫폼으로 고도화 필요



※ 마이크로서비스아키텍처에서는API로멀티센터의서비스를통합제공

Source : 클라우드 네이티브 추진 시 고려사항 (교육 교재)
클라우드 네이티브 기반 행정·공공 서비스 확산 지원 - 한국지능정보사회진흥원

정보시스템 자가 진단 체크리스트

현행 정보시스템에 대한 체크리스트를 도출하였으며, 6개 이상 "Y"응답시, 클라우드 네이티브 도입이 필요한 것으로 판단할 수 있습니다.

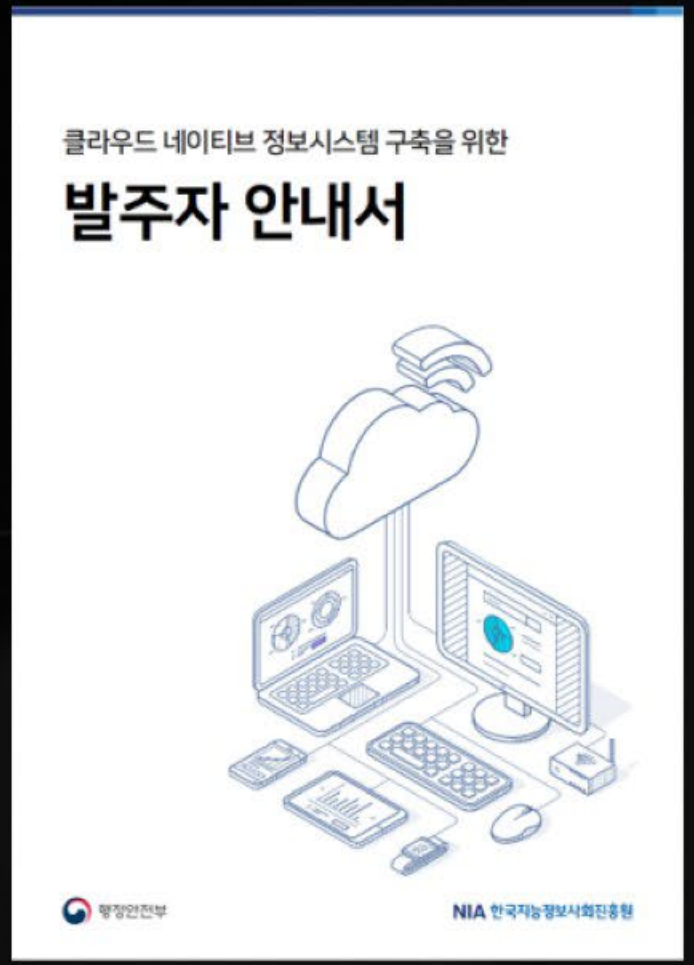
▌정보시스템 자가진단 체크리스트 예시

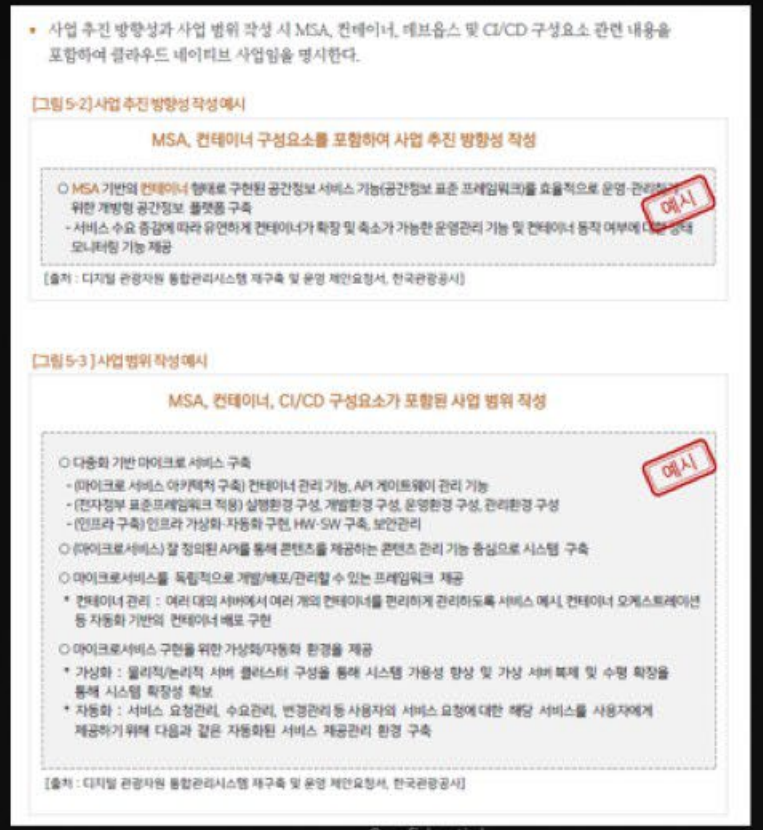
구분(목표)	자가진단항목	답변
안정적 서비스 운영	1 • 초기개발비의 약 15%이상을 매년 추가개발 및 유지보수 비용으로 사용하고 있습니까?	✓
	2 • 다양한 원인에 의한 장애 발생 시 장애복구(예.시스템증설, 업그레이드 등)를 위해 서비스를 중단한 적이 있습니까?	✓
	3 • 특정시점(년, 월, 주, 시)에 트래픽이 증가로 접속지연으로 불만이 제기된 적이 있습니까?	
업무 및 기술 변화 대응	4 • 수시로 정책, 업무 요건 등의 변화에 따른 요구사항에 대해 신속한 대응이 필요합니까?	
	5 • 디지털 신기술(빅데이터, AI, 블록체인, IoT등) 적용 및 다양한 언어 및 다양한 오픈소스에 대한 요구사항 반영이 필요합니까?	✓
	6 • 소규모 서비스 단위로 기능과 DB의 명확한 분리가 가능하고, 독립적 단위로 실행이 가능합니까? (공통 기능 및 데이터 사용, 타 시스템과의 연계성, 서비스 의존관계 등 확인)	✓
개발 품질 향상	7 • 시스템 개발 및 운영시 개발 및 운영 조직의 분리에 따라 의사소통, 개발 및 배포 지연 등의 문제가 존재합니까?	✓
	8 • 소스코드의 복잡성으로 서비스 확장이 곤란하여 서비스 분리 및 소스코드 개선이 필요합니까?	✓
개발기간	9 • 개발된 SW를 형상관리 시스템에 커밋 후 개발계, 검증계, 운영계 서버에서 빌드, 테스트, 배포하는 과정에 빌드·테 스트·배포 도구를 사용하지 않거나 부분적으로 사용하고 있습니까?	
	10 • 현행 시스템의 배포주기를 단축하고 싶습니까?	

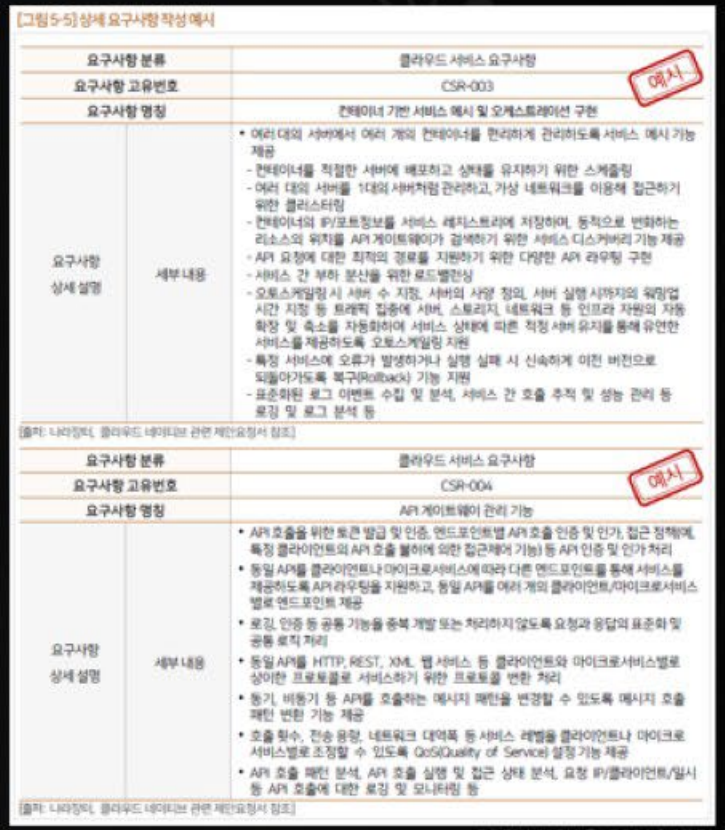
6개 이상 "YES" 응답 시 도입 검토

발주자 안내서 - 클라우드 네이티브 정보시스템 구축

- 발주기관에서 클라우드 기반 정보화 사업을 기획하고 발주하기 전에 발주자 안내서를 통해 클라우드 네이티브의 개념 , 주요 기술 등을 이해하고 도입 적합성을 검토

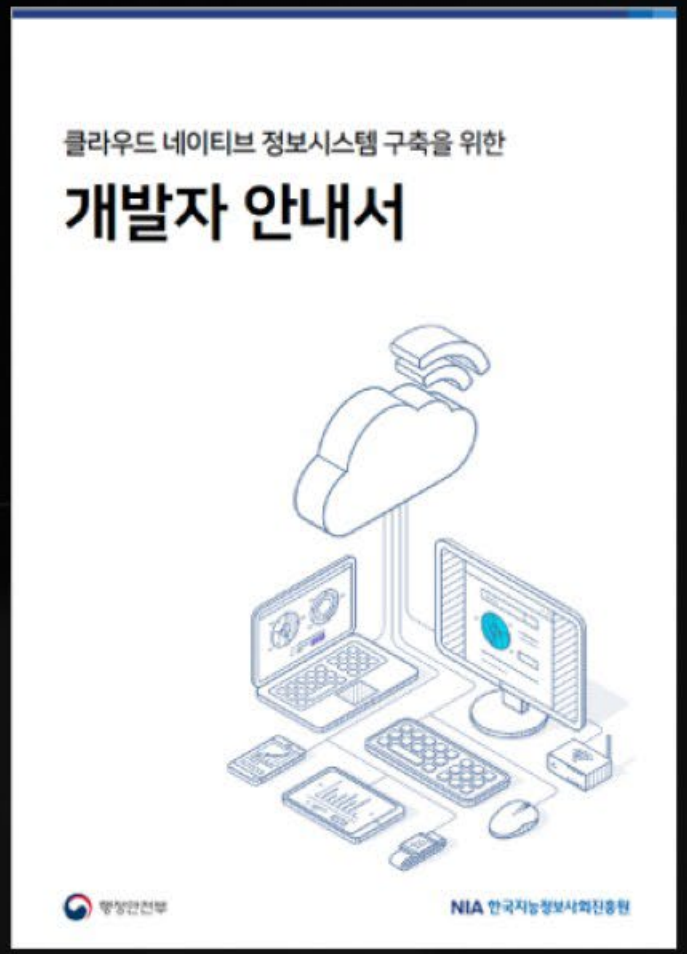


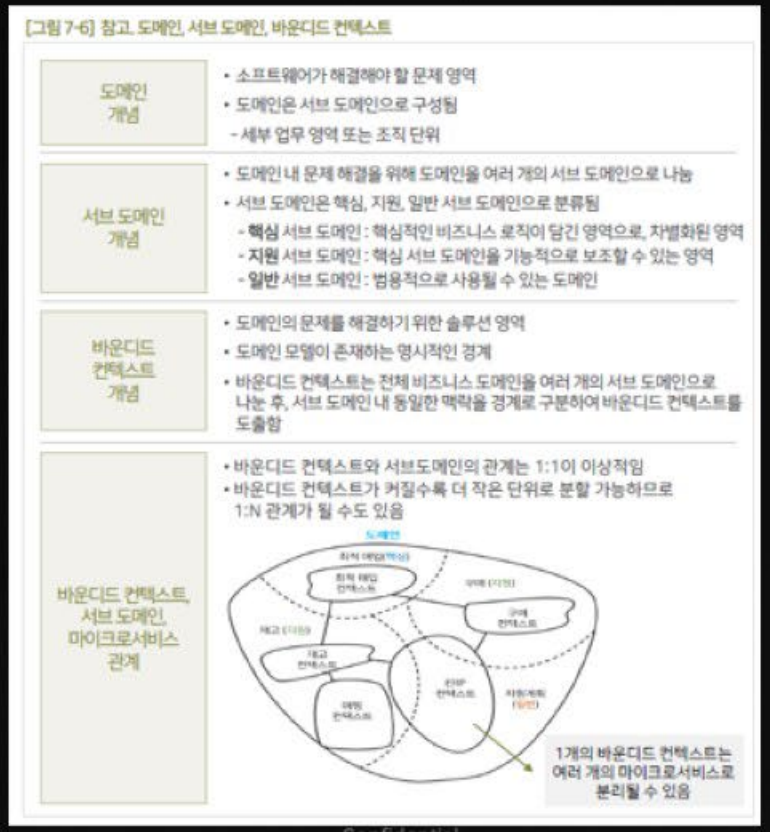


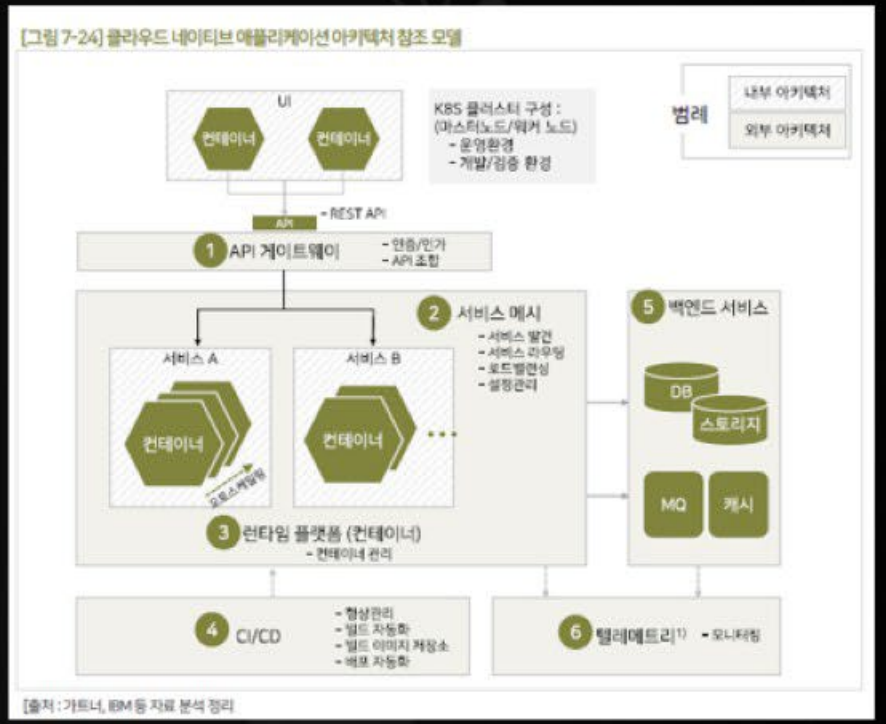


개발자 안내서 - 클라우드 네이티브 정보시스템 구축

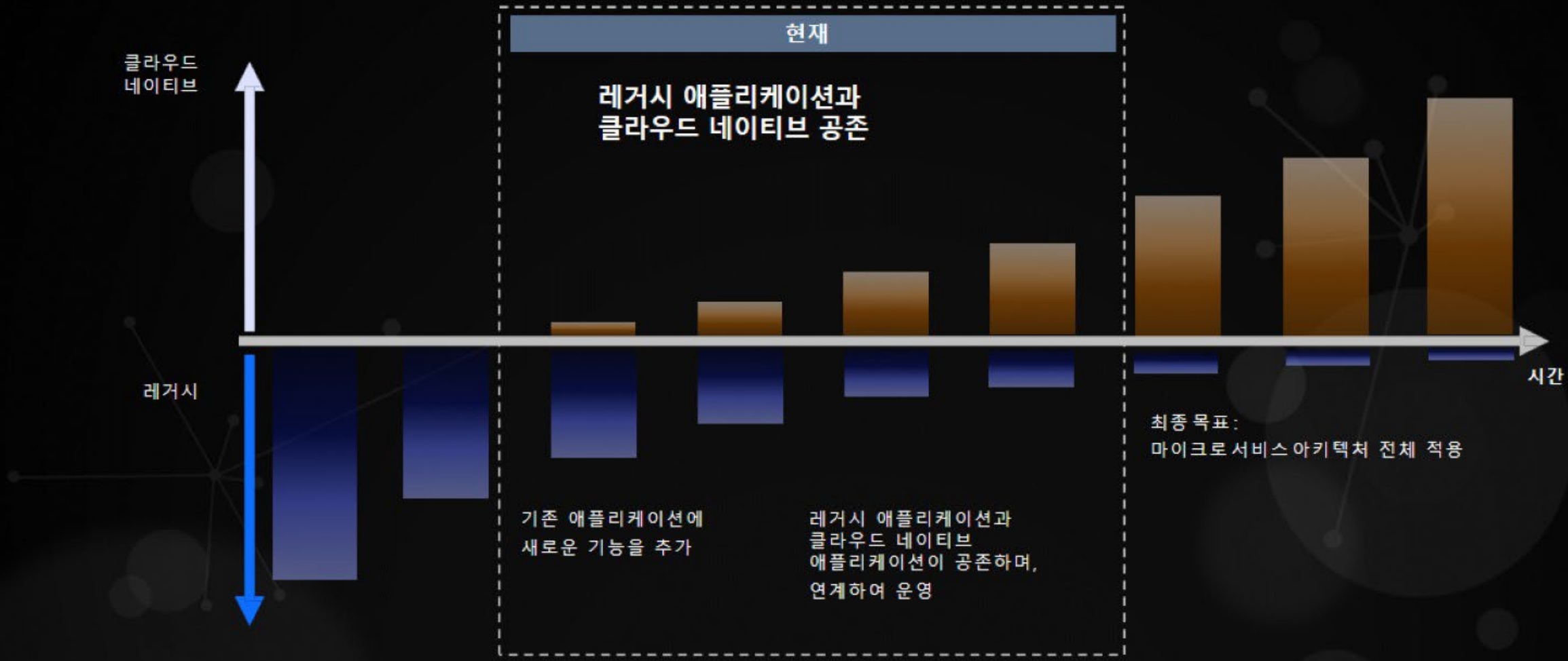
- 클라우드 관련 정보화 사업을 준비하는 중앙행정기관 , 지방자치단체 , 공공기관 등 발주자와 클라우드 네이티브 정보시스템 구축 및 운영 사업에 참여하거나 관심이 있는 개발자







레거시와 클라우드 네이티브 애플리케이션 전환



Source : IBM

Cloud Native Summary

클라우드에 대한 개발팀과 운영팀의 고민들

- **Cloud Native Computing**은 클라우드의 특성과 장점을 적용하여 구성된 컴퓨팅 환경으로, 인프라, 플랫폼, 어플리케이션/서비스와 개발, 운영, 관리의 전체 영역을 대상으로 함

서비스는 대박인데, 맨날 시스템 장애에 변경사항만 엄청 나네!!

개발을 빨리해도 , 1주일에 1번 배포인데 무슨 소용?

개발/운영 하기도 바쁜데, 오픈소스 스터디는 어떻게?

사용자와 시스템이 늘어 나면 당연히 운영 인력을 충원 해야지 !!!

인프라에 투자한다고 서비스가 좋아지나? 안정된 서비스가 중요하지 !!!

개발팀은 책임도 못질 오픈소스만 자꾸 할까? 제대로 사용하려면 몇 년은 공부 해야지!!

애플리케이션 → 클라우드 네이티브 애플리케이션/서비스

기존 운영환경 → 클라우드 네이티브 운영 환경

혁신적인 IT 환경 구축
IT 조직의 운영 비용 절감과 비즈니스에 대한 민첩성 증대

클라우드에 최적화된 표준화
통합로그, 통합모니터링, 배포 자동화, 소스형상관리, 환경구성 표준화 등

PaaS
개발팀에서 스스로 시스템 S/W 를 설치/구성, 개발에만 집중

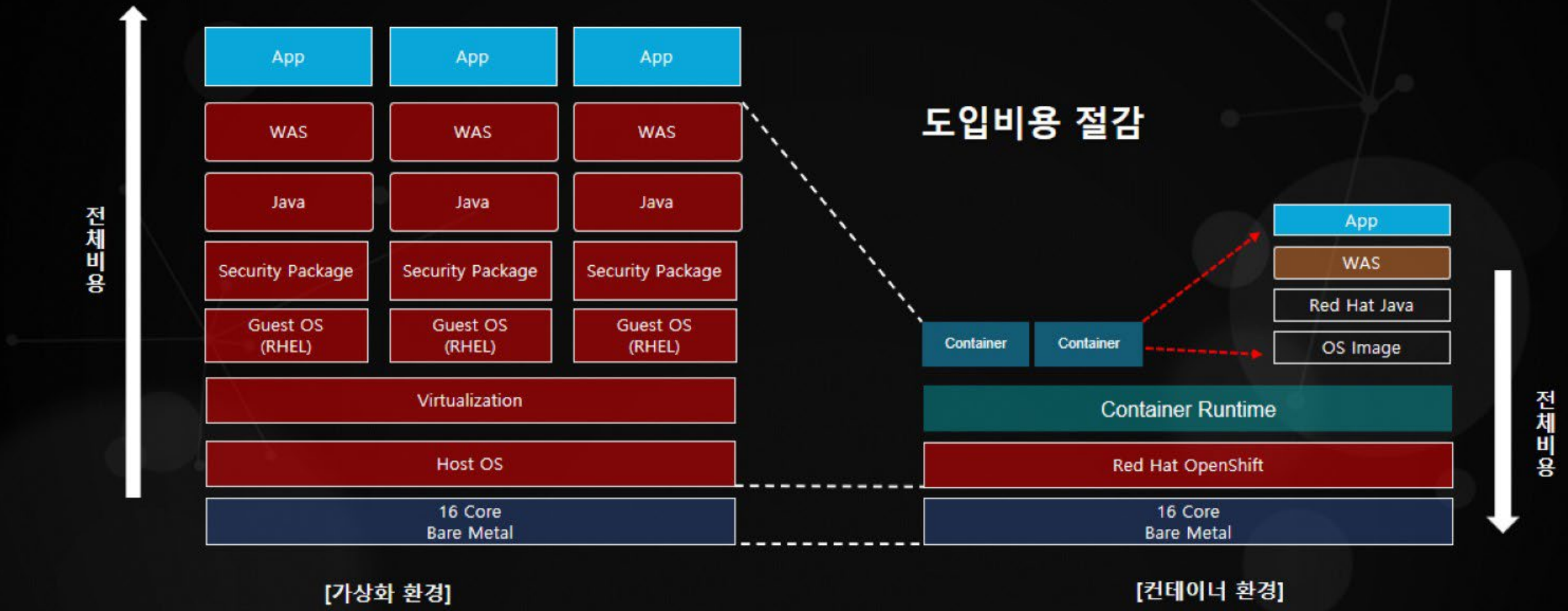
MSA
서비스 변경이 필요하면 언제든지 배포 (1일 10회 이상)

운영자동화
사용자가 폭증하더라도 인력에 의존하지 않고 안정성 있는 서비스


컨테이너 기술의 차이점
인프라 운영의 변화
openmaru
APM

가상화 VS 컨테이너 비교 – 비용적인 측면

- 가상화 대비 Guest OS 유지보수, 라이선스, 관리비용 제거
- 서버 접근제어를 비롯한 보안 솔루션 제거



Maker (OpenShift) vs. Taker (... ?)

openmaru

Maker

오픈소스 리더, 커미터

Red Hat

okd

Vs

Taker

오픈소스 기술지원

안정성,성능, 보안 등 품질 관리는?

구축은 했지만, 운영은 누가?

유지보수 (Patch)

오픈 소스 S/W 의 Maker 와 Taker

openmaru

구분	Maker	Taker
제품 개발 방법	• 오픈소스로 만들고 책임도 지겠습니다.	• 만들지는 않았지만 오픈소스라 동일한 제품입니다.
비즈니스 형태	• 오픈소스 커뮤니티를 리딩하며, Full Time 엔지니어를 채용하여 오픈소스 S/W 제품을 개발	• 오픈소스 S/W 를 가져와 단순 설치나 기술지원
판매 방식	• 오픈소스는 소유가 불가능 -> 서브스크립션	• 인력 중심의 인건비
주요 제품	• Linux, JBoss, OpenShift, OpenStack 등	• CentOS, Apache , Tomcat , Redis , Kubernetes
조직	• SW 제품 개발과 유지를 위한 전체 조직 체계 운영 • Product PM, Committer, QA, Engineer, Support 등	• 설치 중심의 기술지원 인력
기술지원	• 미션크리티컬 부분 - 성능, 보안 , 안정성 (엔진 중심) • 장기 제품 라이프사이클 지원	• 설치/구성 과 UI 커스터마이징 중심 • 성능, 보안, 안정성 그리고 제품에 대한 책임은 한계
라이선스 책임	• 도입 제품에 대한 오픈소스 라이선스 책임	• 없음
로드맵	• 제품의 로드맵 및 기능 개선/ 검토	• 제품 로드맵에 참여하기 어려움
개발자	• 컨설팅, 교육, 파트너 체계	• 설치/구성 엔지니어
Vendor	• Red Hat , Enterprise DB	• 단순 오픈소스 기술 지원


Application Performance Management
감사합니다.
openmaru
APM


openmaru