Document Creation Date
2024.03.10

Document Version
1.0

openmaru

클라우드 네이티브로의 여정

OPENMARU Inc.
오픈마루 주식회사

Platform As A Service

클라우드 네이티브 도입을 고려해야하는 이유?

클라우드 핵심 개념 : Architecture & Model

openmaru

Architecture

클라우드 네이티브

vs.

클라우드 기반

Service Model

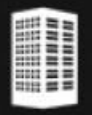
IaaS : Infrastructure As A Service

PaaS : Platform As A Service

SaaS : Software As A Service

Delivery Model

Public Cloud



Private Cloud

Hybrid Cloud

Cloud Delivery Model

openmaru

구분	클라우드 네이티브	구조	특징	벤더
	**Public Cloud**	클라우드 서비스 업체가 인터넷을 통해 컴퓨팅 리소스를 제공하고, 서버의 유지 관리	• 여러 기업이 하나의 클라우드 인프라를 이용 (Multi-Tenant) • 더 적은 구축 비용 • 더 적은 유지 보수	• AWS • Azure • Google Cloud • Oracle • Alibaba
	**Private Cloud**	기업이 클라우드 서버를 독점 아키텍처를 이용하여 자사의 데이터 센터 운영	• 하나의 기업에 하나의 인프라스트럭처 (싱글 테넌트) • On Premise 하드웨어 • 고객이 인프라 관리	• HPE • VMWare • Dell EMC • IBM • Red Hat
	**Hybrid Cloud**	On Premise 인프라, 프라이빗 클라우드 과 퍼블릭 클라우드의 혼합 컴퓨팅 환경	• 데이터 처리를 더 • 개인 및 제어 가능 • 버스트 기능 • 기존 시스템을 함유 할 수있다	• Red Hat • AWS • Azure • Google Cloud

클라우드 서비스 모델

openmaru

On-Premise

- Application
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Network

- 고객은 인프라 제공, 유지 및 애플리케이션 호스팅 모두 책임

Infrastructure-as-a-Service (IaaS)

- Application
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Network

- 공급 업체는 인터넷을 통해 컴퓨팅 인프라를 제공
- 예 : AWS EC2, MSFT Azure

Platform-as-a-Service (PaaS)

- Application
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Network

- 애플리케이션 개발을 위한 플랫폼을 제공
- 공급 업체는 서버, 스토리지, 네트워크를 관리
- 고객은 애플리케이션 관리
- 예 : Salesforce Lightning, Heroku

Software-as-a-Service (SaaS)

- Application
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Network

- 인터넷을 통해 제공되는 소프트웨어
- 공급 업체가 소프트웨어 구축, 유지, 운영
- 예 : G-suite, Microsoft 365

범례 : 기업 고객 관리 클라우드 공급자 관리

클라우드 네이티브 란?

- 클라우드의 이점을 최대로 활용할 수 있도록 애플리케이션을 구축하고 실행하는 방식

DevOps (SRE)

애플리케이션 **개발-운영 간의 협업 프로세스**를 자동화하는 것을 말하며 결과적으로 **애플리케이션의 개발과 개선 속도향상**

- 속도와 안정성
- 협업 강화
- 문화
- 플랫폼 필요성 대두
- OnDemand Service
- SRE

Continuous Delivery

- **지속적인 통합**(CI)은 개발자가 작업한 **코드를 자동으로 테스트하고 통합**
- **지속적인 배포**(CD)는 코드를 리포지토리에 업로드하고, **서비스 배포로 릴리즈까지 자동화**

- Agile
- 짧고 지속적 반복
- 지속적 통합/배포/제공
- 플랫폼/애플리케이션 배포 자동화

Microservices

애플리케이션을 구성하는 서비스들을 독립적인 **작은 단위로 분해하여** 구축하고 각 구성 요소들을 **네트워크로 통신하는 아키텍처로** 서비스 안정성과 확장성(scaling)을 지원

- API First
- 도메인 주도 설계
- 독립적 확장 가능
- 보다 빠르고 독립적인 배포
- 간편한 개발 및 유지관리

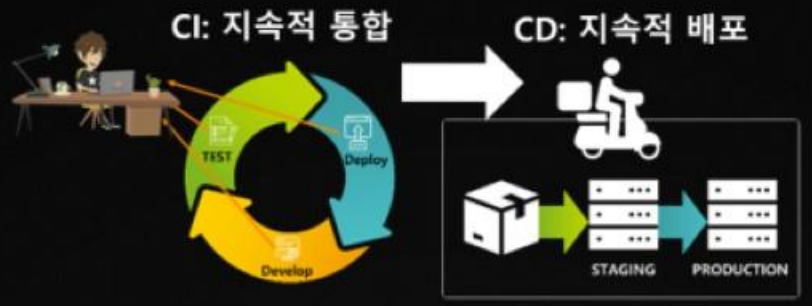
Containers

가상화 기술 중 하나로, **시스템을 가상화 하는 것이 아니라** 애플리케이션을 구동할 수 있는 컴퓨팅 작업을 패키징하여 **OS를 가상화** 한 것입니다.

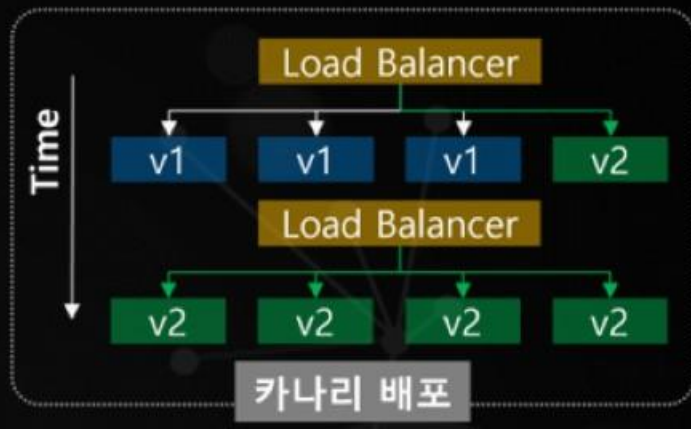
- 컨테이너 오케스트레이션
- 불변의 인프라스트럭처
- 변경이 아닌 폐기 후 생성
- 일관된 환경 유지

클라우드 네이티브 기반 환경의 장점

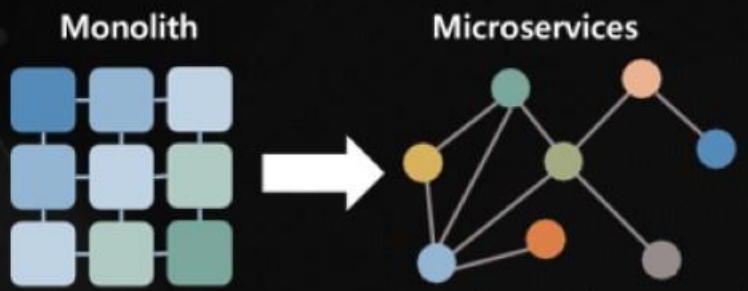
컨테이너 기반의 빠른 개발환경



신속한 개발과 편리한 배포



서비스 무중단



MSA개발에 적합한 환경

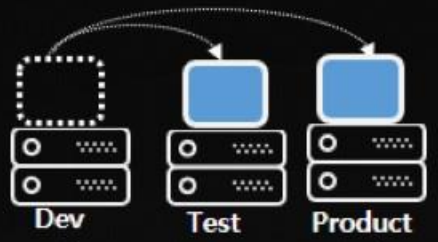
DevOps기반의 민첩한 개발

Cloud Native Computing으로 전환 효과

- **Cloud Native Computing** 환경은 클라우드가 제공하는 민첩성, 가용성, 확장성의 장점을 어플리케이션/서비스의 개발, 운영, 관리에 적용하여 기존 컴퓨팅 환경을 최적화 함

On Demand Delivery

필요한 컴퓨팅 자원을 즉시 제공



Consistency & Continuous

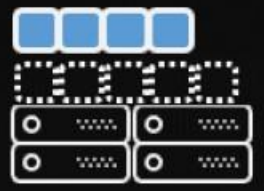
이미지 기반으로 구성, 배포 효율화
개발과 운영 환경의 일관성

Rolling Update

업그레이드 또는 패치 시
다운 타임은 제로 또는 최소화

Self Recovery

- 비정상 애플리케이션 재시작
- 노드의 장애 발생시
  정상 서버 노드로 자동 재배치



Application Scaling

VM 단위가 아닌 어플리케이
단위의 오토스케일링

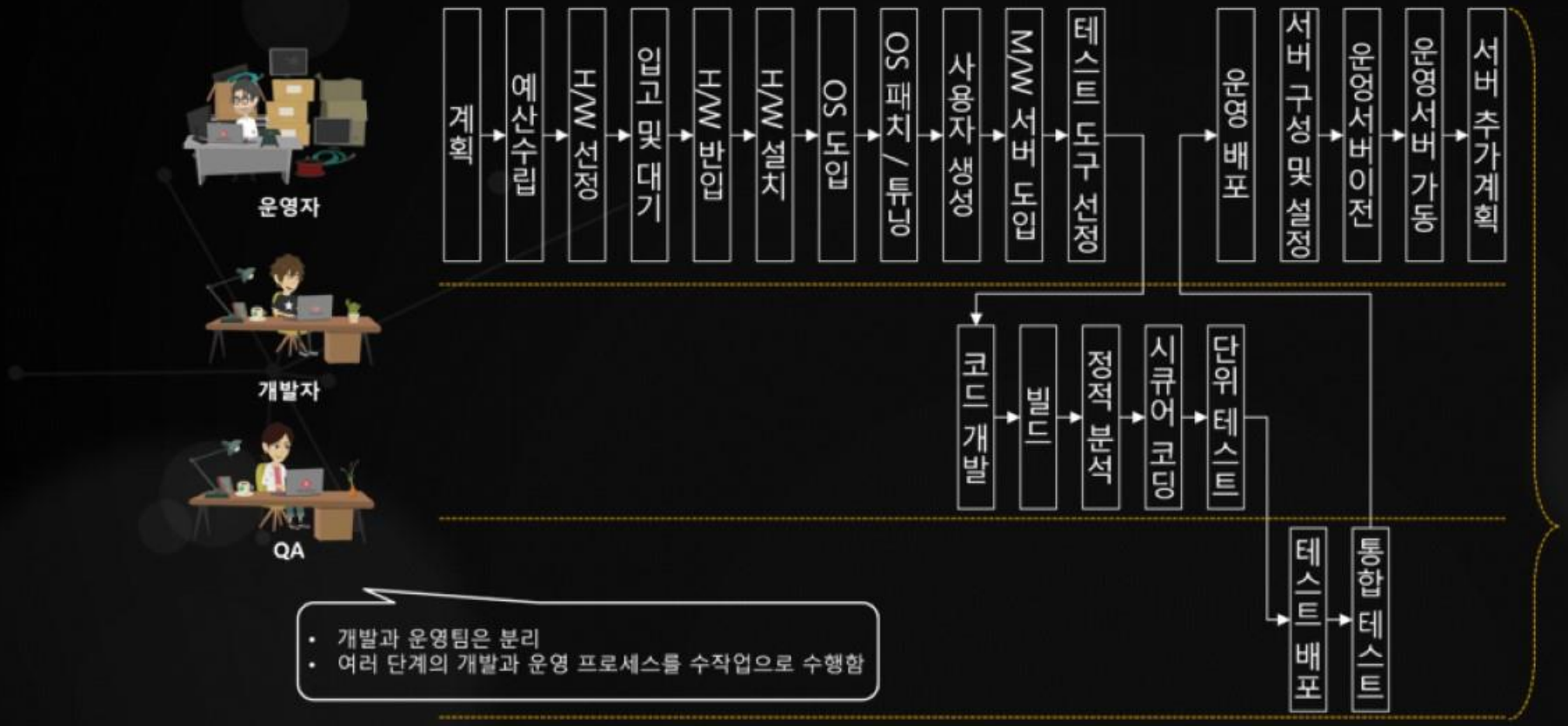
Portable

멀티/하이브리드 클라우드 기반
어플리케이션/서비스 운영

개발팀과 운영팀 누구의 잘못인가? 불행의 시작

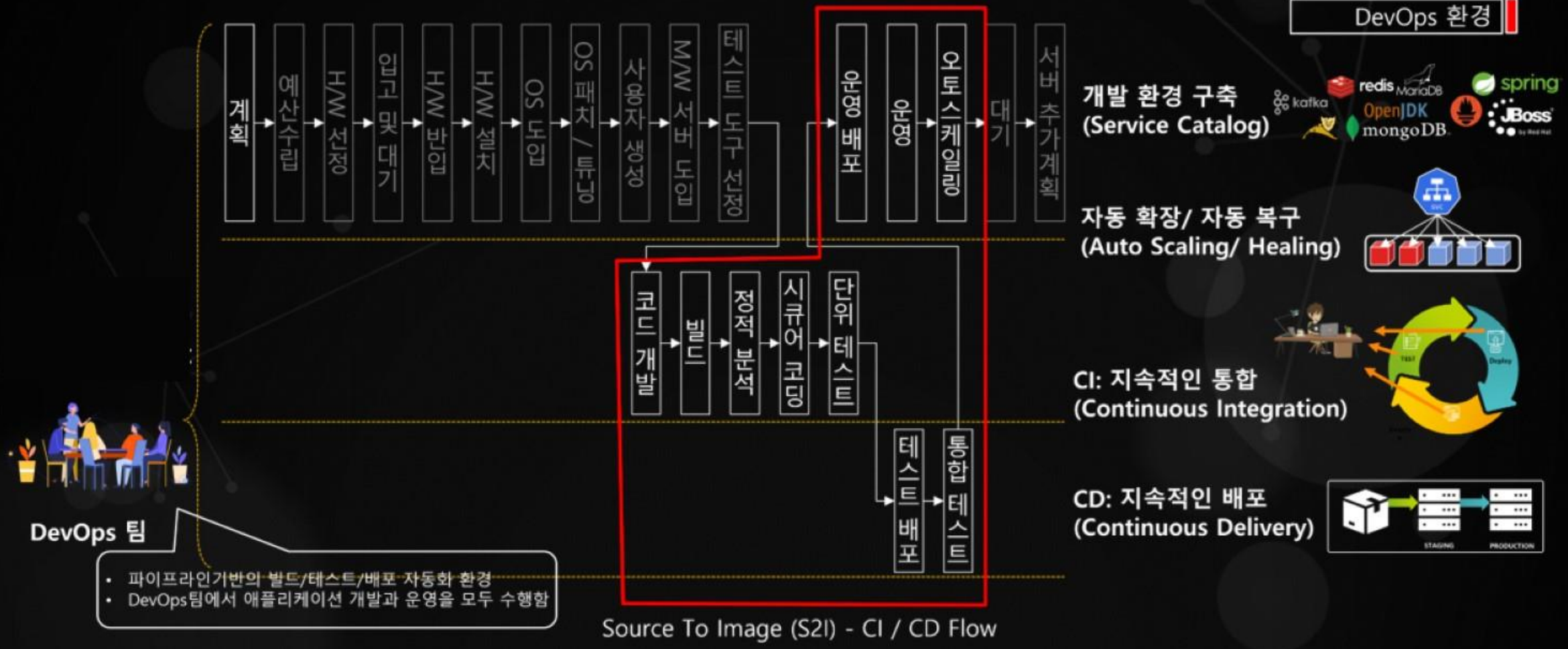
- 하드웨어 도입부터, OS, 미들웨어, 빌드/배포, 기타 인프라 환경 등 복잡한 과정
- 관리 서버 증가

기존개발 운영 환경



컨테이너 자동화 도구를 이용한 프로세스

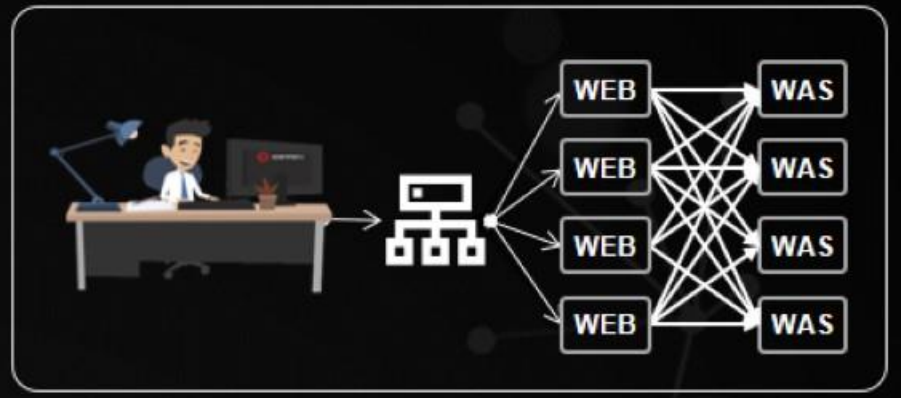
- 파이프라인 기반의 자동화 환경을 이용하여 신속한 개발 및 운영 환경 구축
- 필요에 따라 구축과 삭제가 편리



시스템 운영의 현실

웹시스템의 요구사항과 특징

- Scale-out 형 인프라로 계층 형 아키텍처
- 가상화와 클라우드환경에 적합하며 인스턴스 개수가 많으며 노드 간 연결성이 높음
- 장애 민감하며 신속한 장애 복구와 재발 방지가 중요
- 설치되는 관련 소프트웨어가 많으며 환경 검증이 필요



시스템 운영 이슈

- 시스템 환경의 불일치(Dev/Stage/Prod, 서버별)
- 긴 배포 시간
- 수 작업으로 인한 Human Errors
- IT Agility 부족으로 인한 운영팀 축소
  - 개발팀에서 직접 IT 인프라 운영
- 문제점의 발견과 조치에 많은 시간 소요

시스템 비대화로 작업 폭증과 인력 부족 어떻게 할까요?

장애의 65 %는 Human Error이며,시스템 복잡도와 난이도 증가

시스템 운용 업무의 45 %는 정기적으로 수행해야하는 반복 작업

운영 효율화를 통한 비용 절감의 요구

시스템의 대규모 화

높은 수준의 엔지니어 부족

지속적인 시스템 통합 요구

동일한 작업 반복

운영 품질 향상

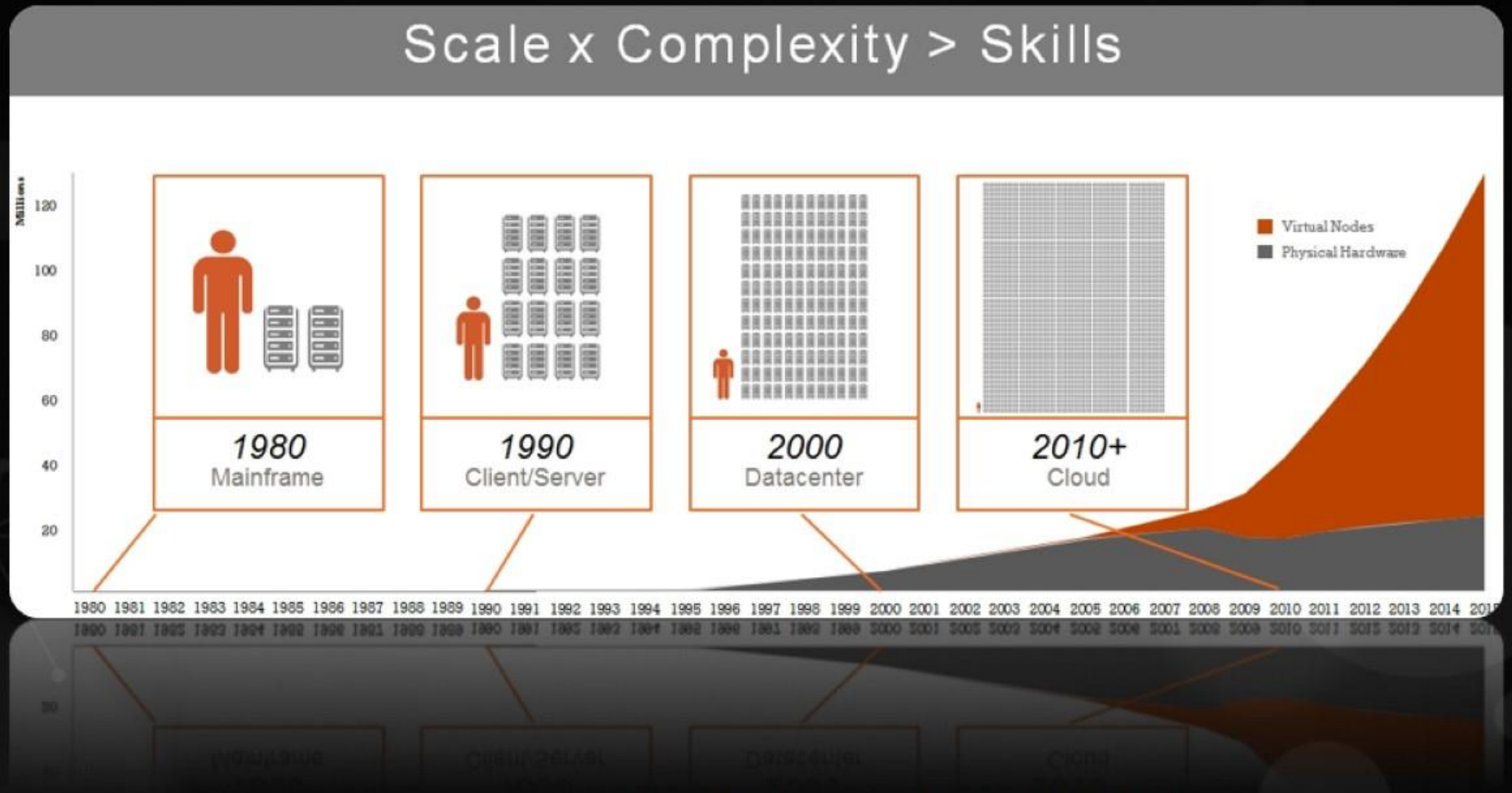
운영 비용 (TCO) 절감 요구

업무 확대와 관련 데이터양의 비약적인 증가

가상화, 클라우드 등 다양한 운영 환경의 증가와 관리 효율화 요구

운영 품질에 대한 지속적인 향상 요청

Increasing scale and complexity means we need admin automation



Opscode gets more venture dough for its Chef
From - http://goo.gl/dLcjS

OpenShift 을 통한 IT 인프라 운영 자동화

- IT 인프라의 대규모화, 고도화에 따라 IT 장비에 대한 환경설정 및 정보 취합이 복잡하고 어려움
- 작업 계획시간과 현장 작업 시간의 증가와 휴먼 에러의 증가

Before OpenShift

- 시스템 운영을 위한 관리 작업 증가
- 현장 작업 시간 증가
- Human Error 증가

After OpenShift

- 운영 기술 표준화를 통한 준비 시간 및 작업 시간 감소
- 시스템 일괄 설정 작업 시간 단축
- 시스템을 통한 작업으로 Human Error감소

PaaS Cloud 혜택

- OpenShift 을 적용하여 민첩성 높은 서비스를 제공

작업 공수 절감

기존 수동으로 해왔 던 작업을 자동화하여 작업 공정 및 납기 단축

운영 품질 향상

관리자의 개입을 최소로 자동화하여 작업 품질을 균일하게 유지

시스템 운영의 표준화 촉진

- 자동화 및 버전 관리 함으로써 시스템 운영 정책 및 업무 표준화

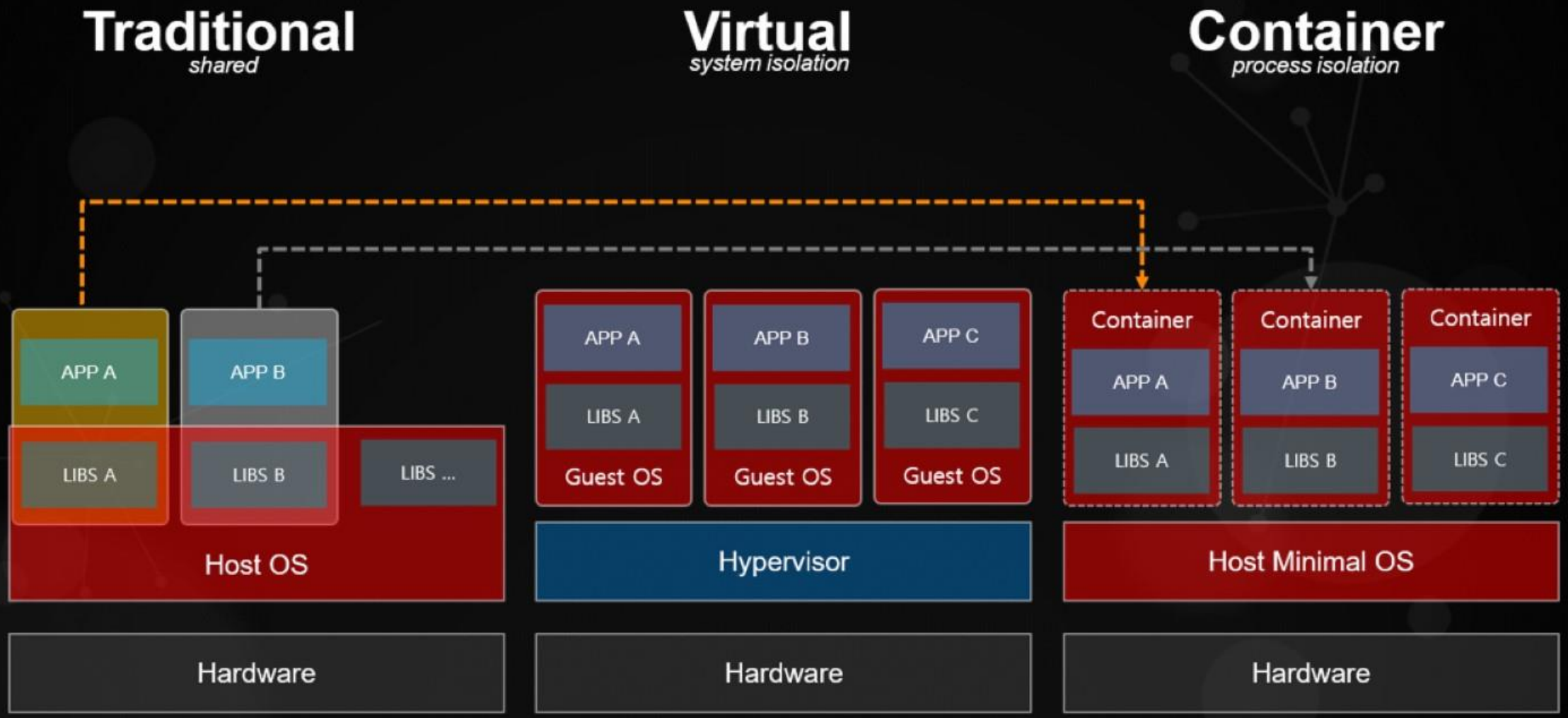
작업 통제 강화

작업 작업을 자동화함으로써 내부 통제 및 보안 측면에서의 효과를 기대

Platform As A Service

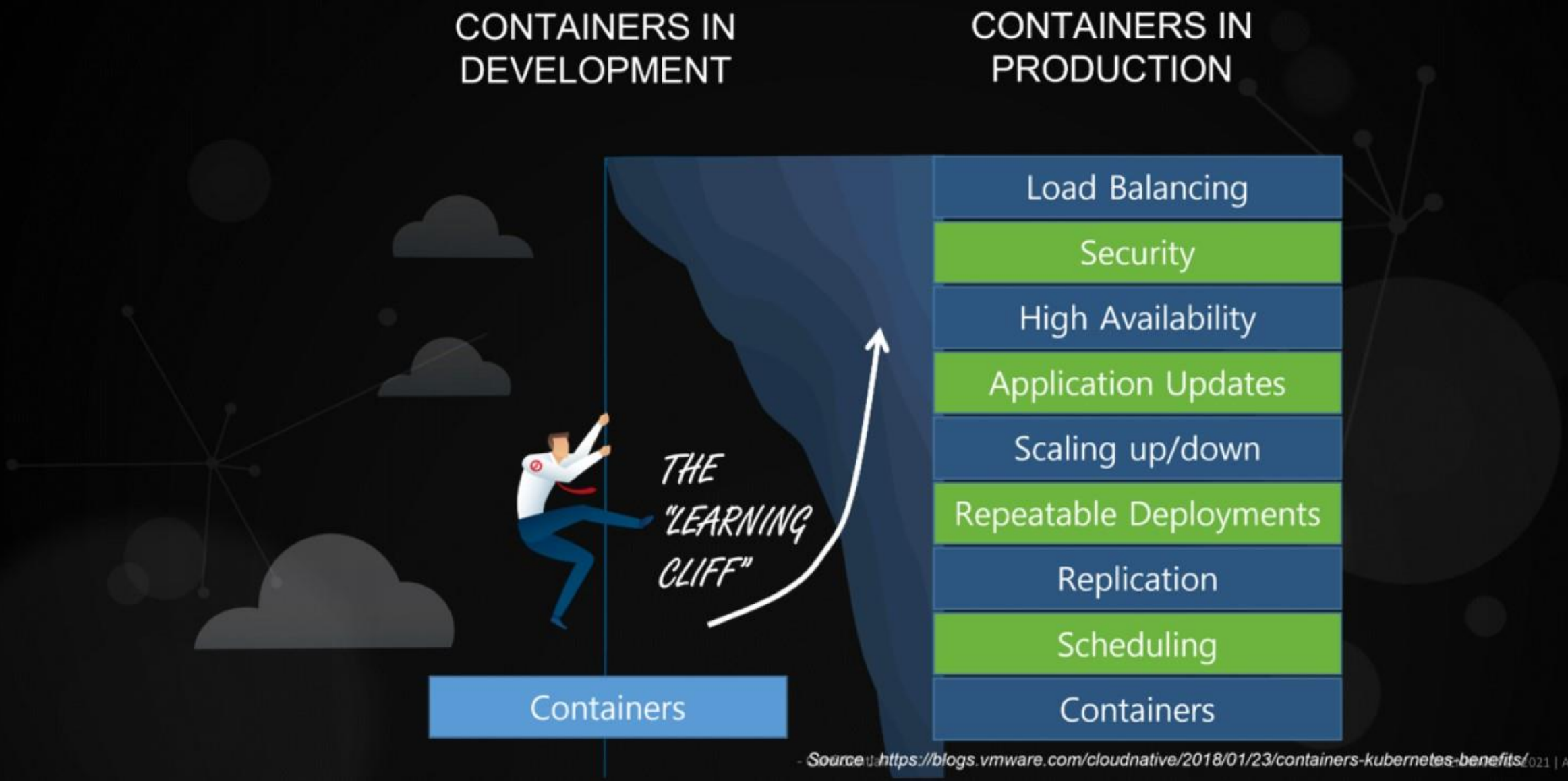
클라우드 네이티브의 기반 기술, 컨테이너

Evolution



Containers and Kubernetes: The Time Is Now

openmaru



Source: https://blogs.vmware.com/cloudnative/2018/01/23/containers-kubernetes-benefits/

About Kubernetes

openmaru

쿠버네티스(k8s) 는 컨테이너화된 애플리케이션을 자동으로
배포, 스케일링 및 관리해 주는 오픈소스 시스템

- 그리스어로 키잡이의 어원
- " Helmsman (키잡이) " 그리스어.
- " Governor (지사) " 의 어원

- k8s (ubernete -> 8) called k-eight-s

- Originally designed by Google
  (Borg -> Omega -> Kubernetes)

- Go 로 작성된 오픈 소스,
  OSS (Apache License 2.0)

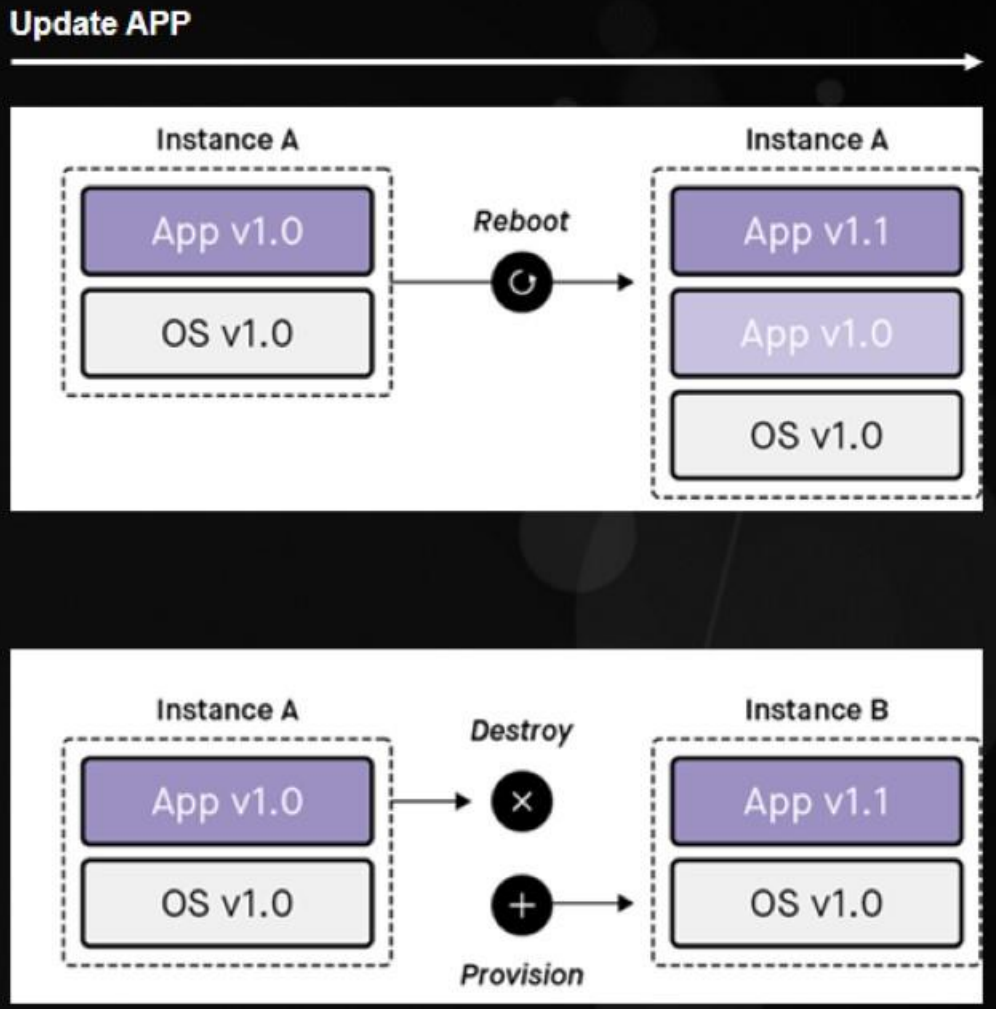
kubernetes

Mutable vs. Immutable Infrastructure (가변 vs. 불변)

openmaru

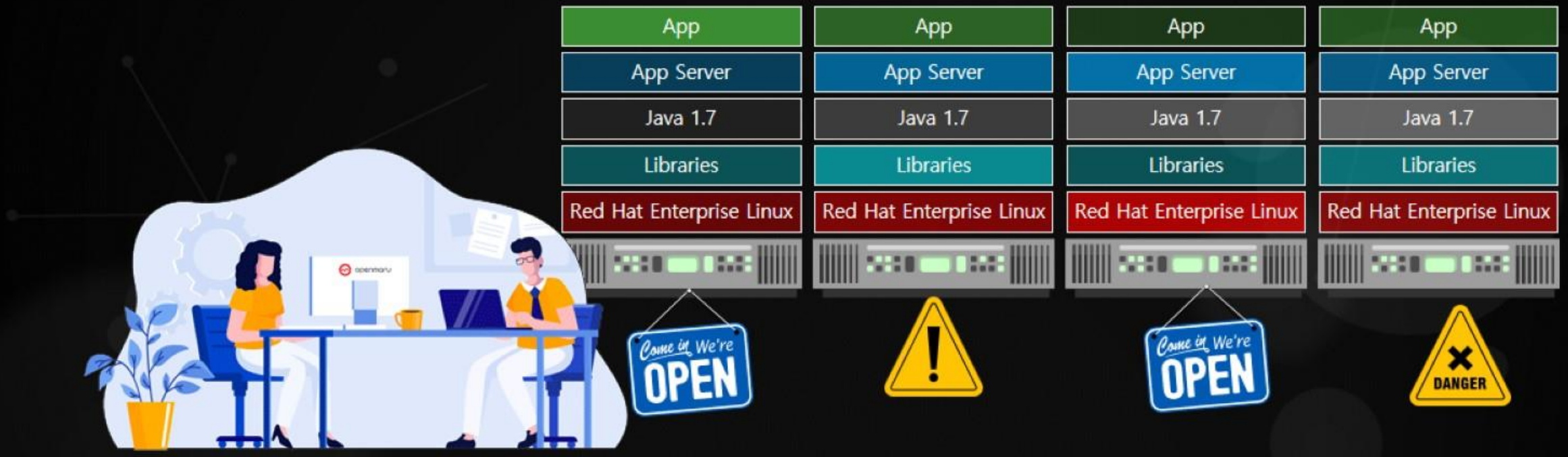
Mutable Infrastructure (물리서버, 가상화)

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

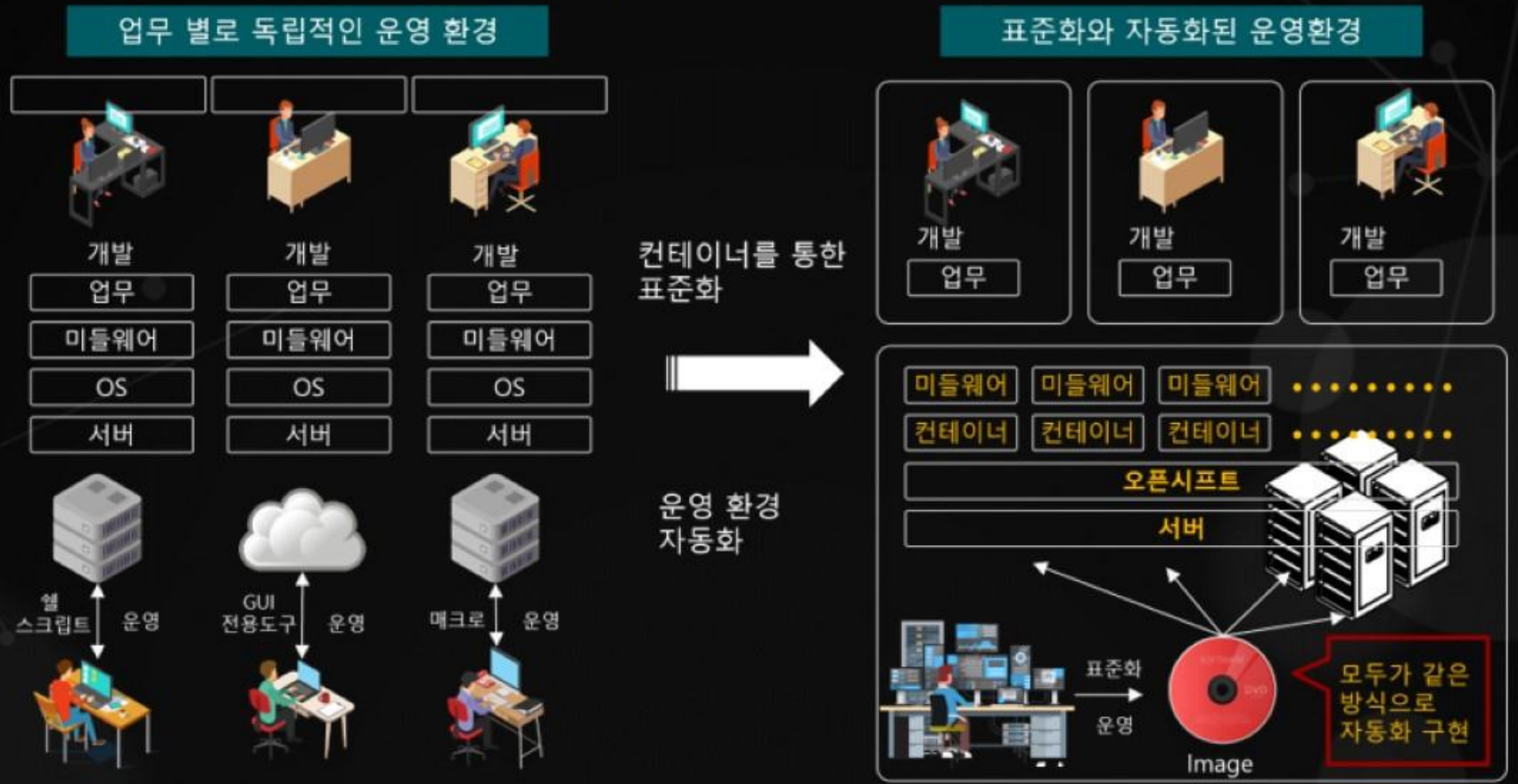


Platform As A Service

클라우드 네이티브의 특징으로 인하여 작업이 어떻게 바뀔까요?

OpenShift 를 통한 개발환경 구축

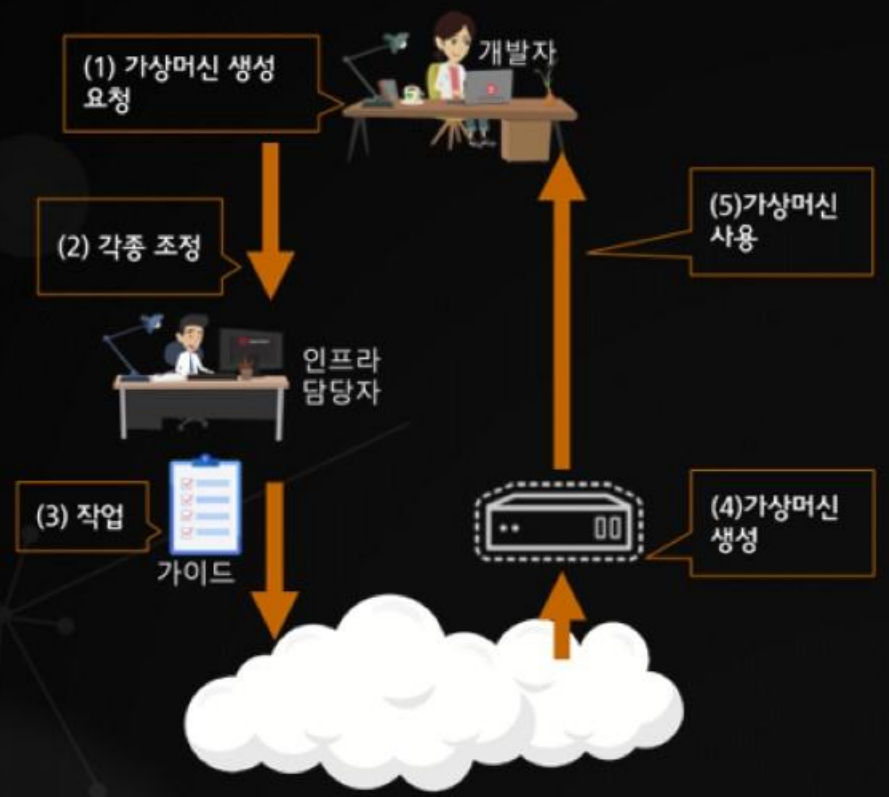
- 시스템 마다 절차와 관리도구가 다르고 수작업에 의한 운영
- PaaS를 통한 시스템 운영 환경의 변화



서버 자원 요청 작업 비교

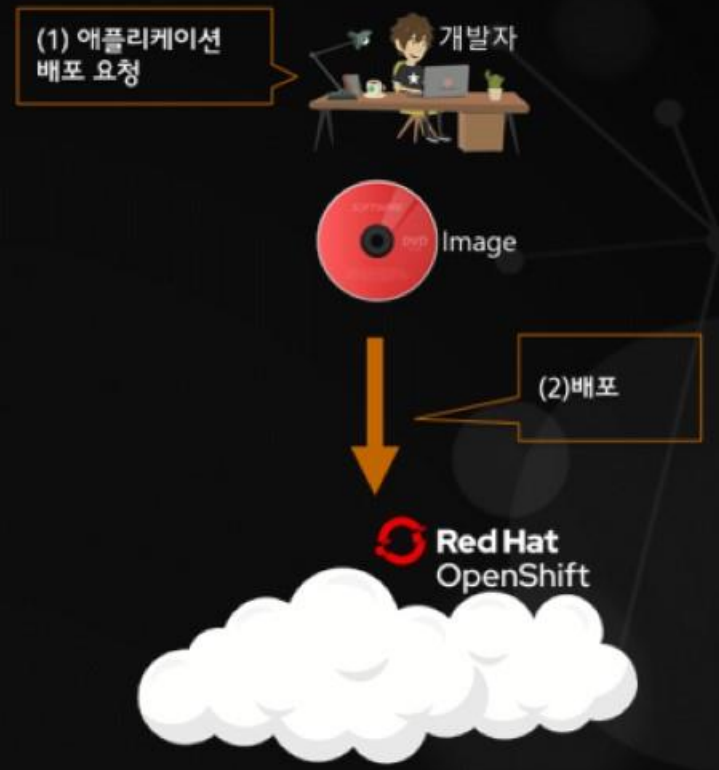
openmaru

Before



- ✓ 사람에 의한 조정과 작업
- ✓ Human Error 와 품질의 차이

After



- ✓ 테스트까지도 자동화
- ✓ 일정한 품질 유지

운영환경 패치 작업 비교

openmaru

Before

업무 시간 외 작업

21:00 사전 백업 — 인프라 담당자 — 가이드

23:00 패치 적용 — 인프라 담당자 — 가이드

3:00 동작 확인 — 인프라 담당자 — 체크 리스트

5:00 작업완료 메일 — 인프라 담당자

확인 시 문제가 있으면 롤백

- ✓ 작업하는 동안 상주 지원하며 시간 소요
- ✓ 오류 확인 및 롤백에 대한 위험요소
- ✓ 품질의 차이

After

업무 시간 중

10:00 이미지 배포

10:20 롤링 업그레이드 — Red Hat OpenShift — 지정 시간에 시작

10:30 동작 확인 — 인프라 담당자 — 체크 리스트

10:40 작업완료 메일 — 인프라 담당자

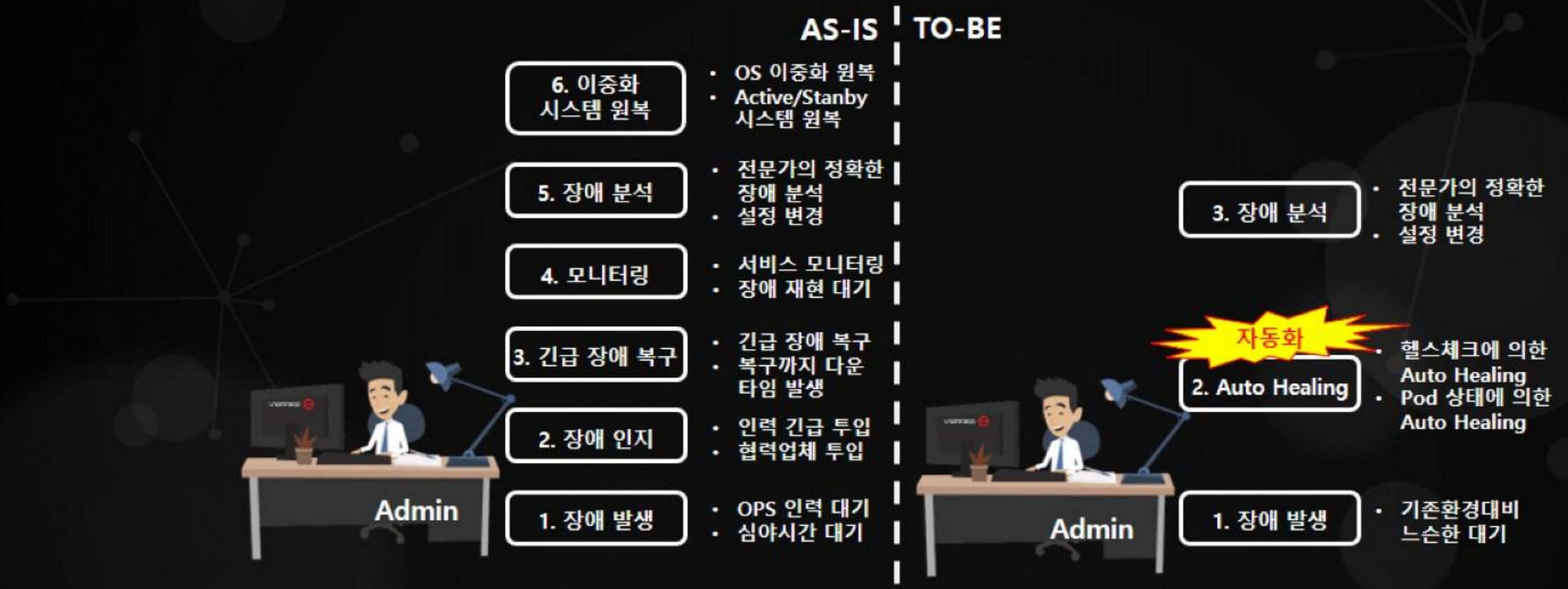
- ✓ 비상주 지원과 유비보수 시간 단축
- ✓ Human Error 차단
- ✓ 일정한 품질 유지

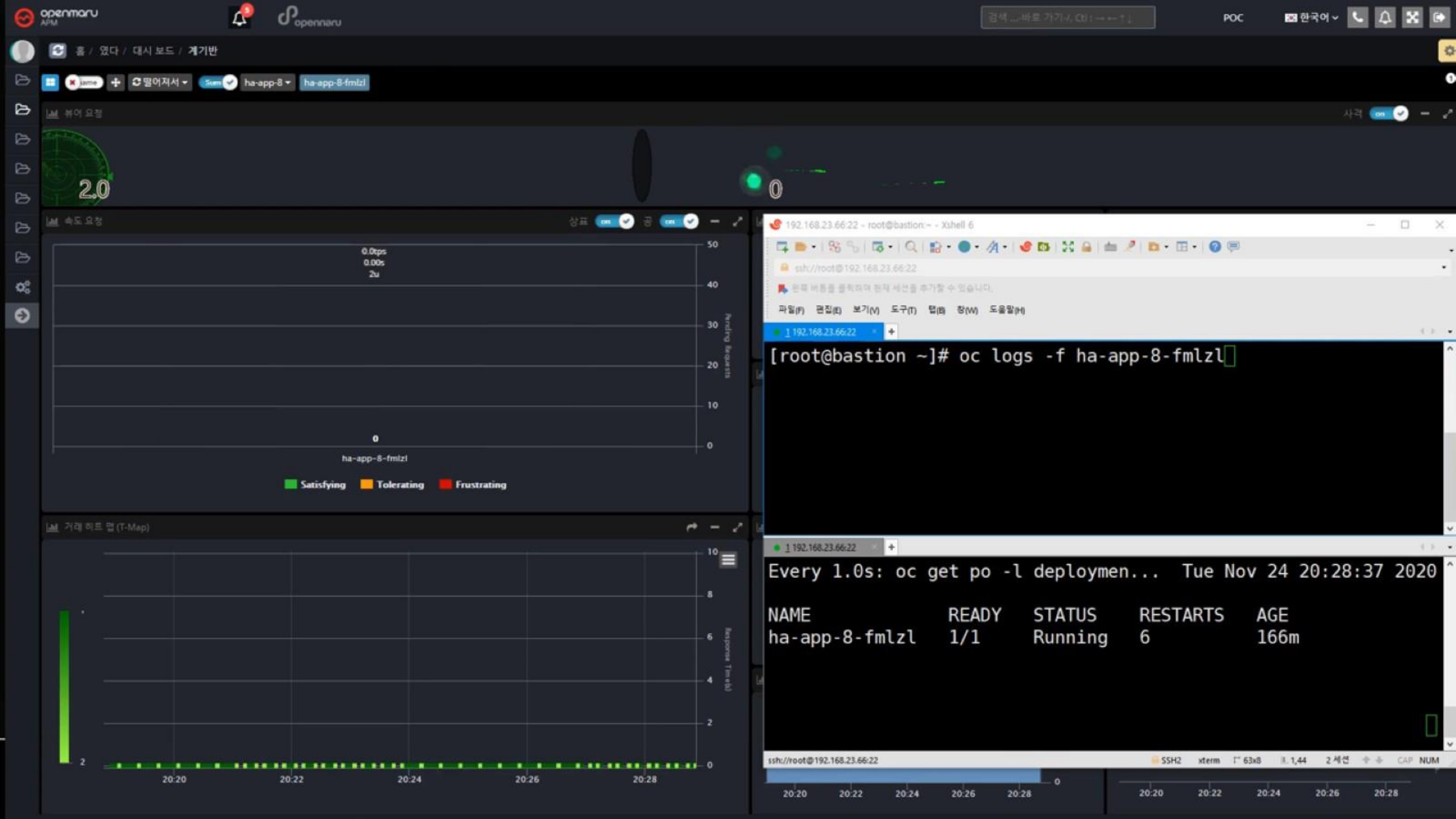
장애 상황 운영관리 비교

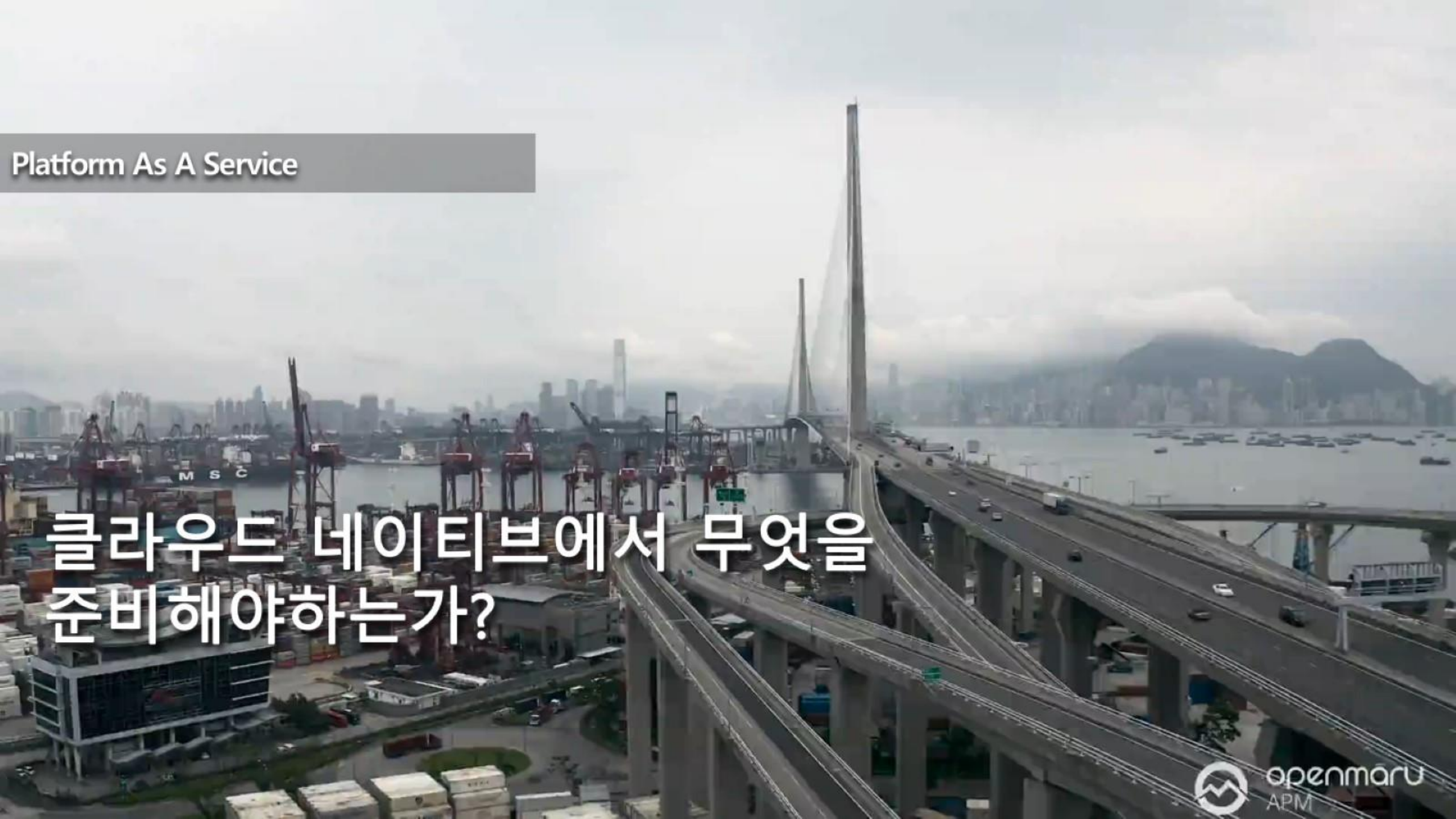
openmaru

인프라 팀(장애상황)

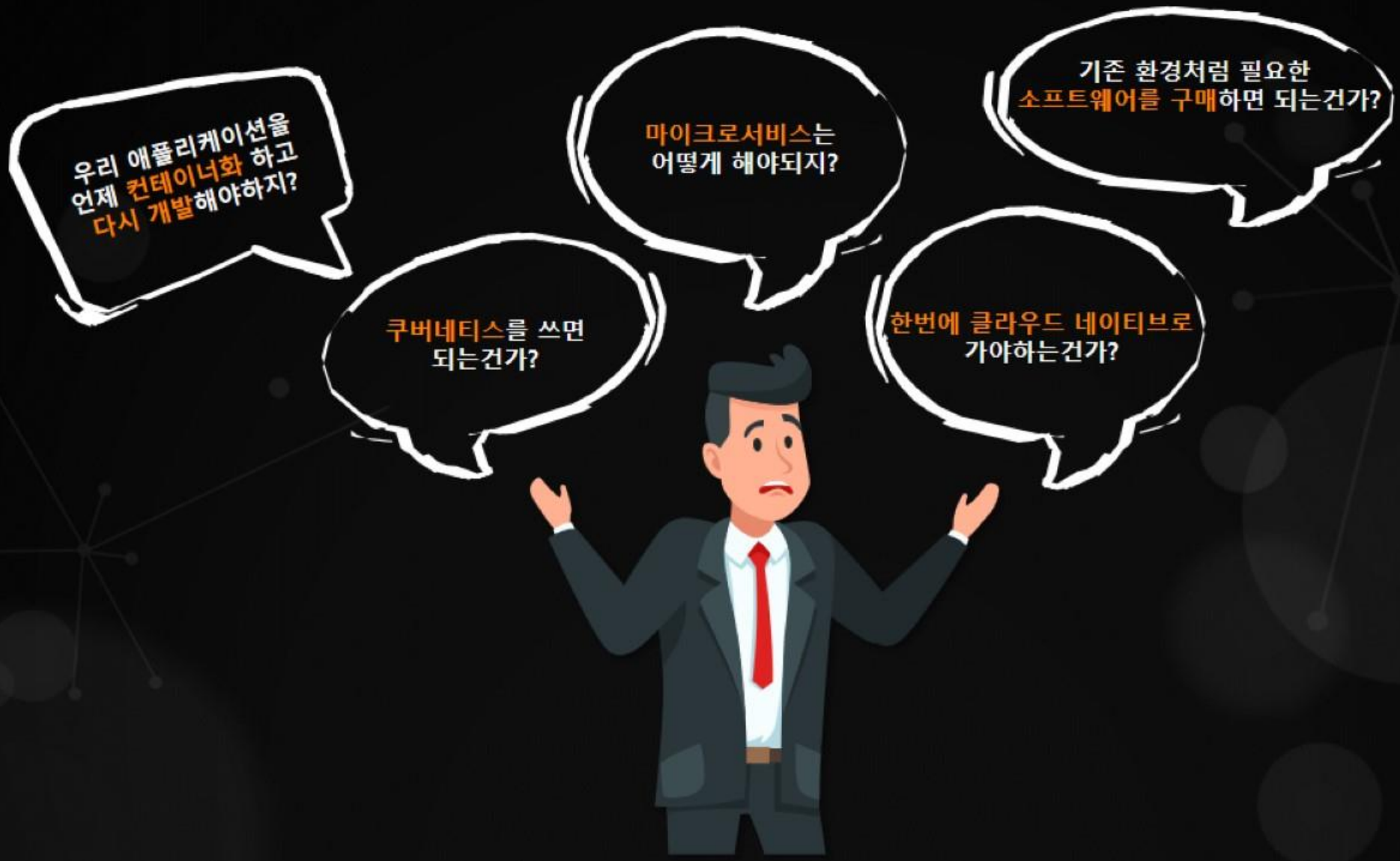
- 기존 환경은 장애 발생을 대비한 인력들이 상시 대기해야 함
- 야간 장애 발생시 **복구까지 긴 시간 소요, 서비스 다운 타임** 발생
- 컨테이너 환경은 Auto Healing으로 인해 장애 발생시 바로 자동 복구




openmaru
APM
POC
한국어
홈 / 였다 / 대시 보드 / 계기판
ha-app-8
ha-app-8-fmlzl
뷰어 요청
2.0
0
속도 요청
0.0tps
0.00s
2u
ha-app-8-fmlzl
Satisfying
Tolerating
Frustrating
거래 히트 맵 (T-Map)
20:20
20:22
20:24
20:26
20:28
192.168.23.66:22 - root@bastion:~ - Xshell 6
[root@bastion ~]# oc logs -f ha-app-8-fmlzl
Every 1.0s: oc get po -l deploymen... Tue Nov 24 20:28:37 2020
NAME READY STATUS RESTARTS AGE
ha-app-8-fmlzl 1/1 Running 6 166m

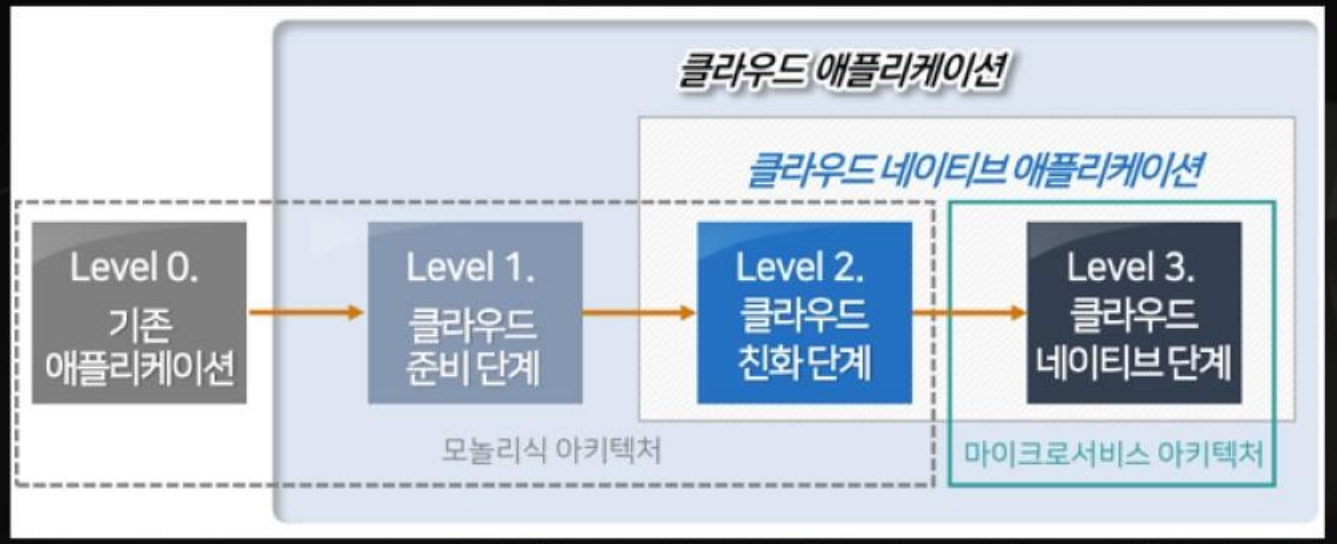

Platform As A Service
클라우드 네이티브에서 무엇을
준비해야하는가?
openmaru
APM

클라우드 네이티브를 가로막는 두려움



클라우드 애플리케이션 성숙도 단계

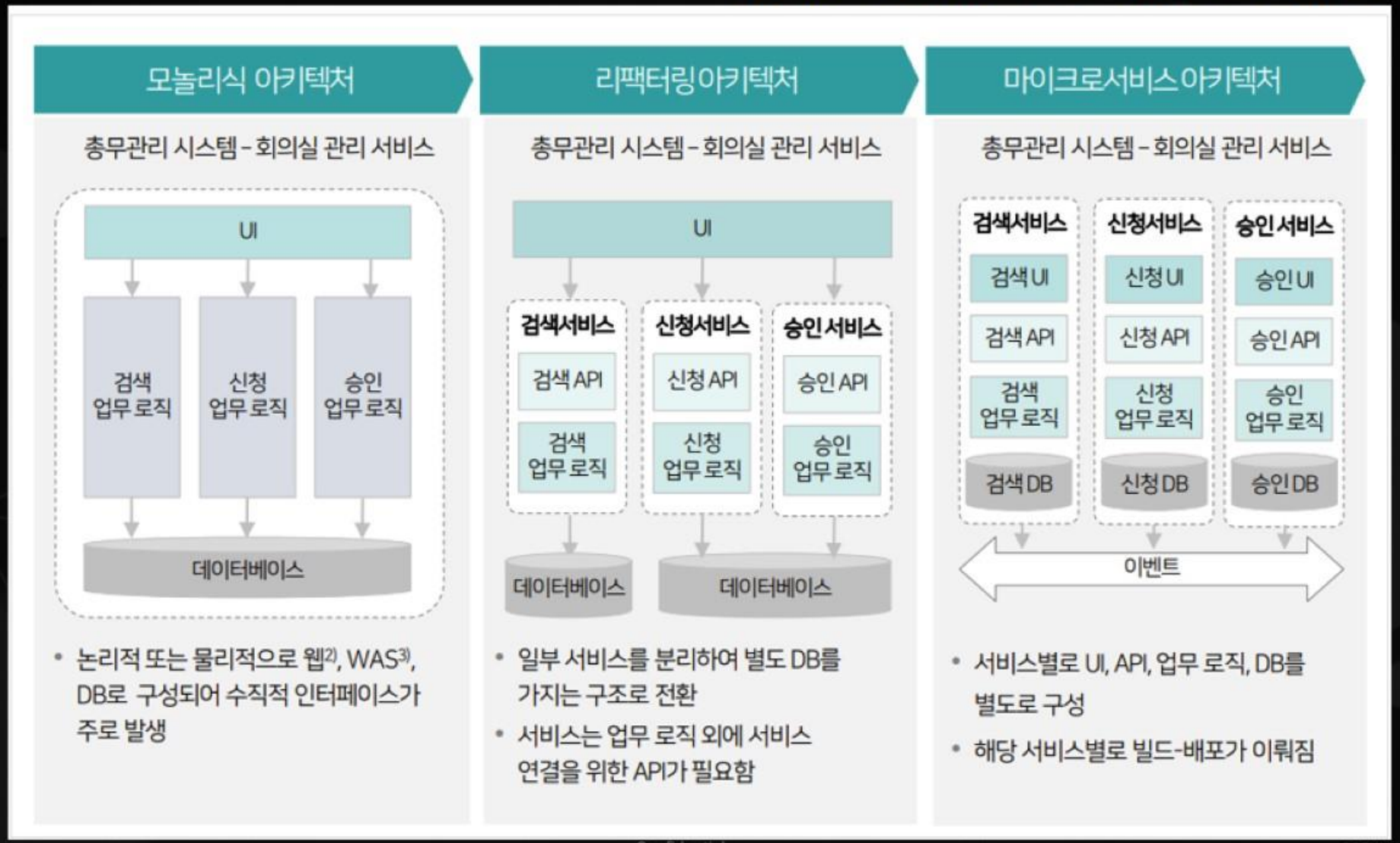
- 클라우드 네이티브는 컨테이너 기반의 PaaS로 시작하여, CI/CD, MSA 모두 포함된 단계
- Lv1. 클라우드 준비 단계 → Lv2. 클라우드 친화 단계 → Lv3. 클라우드 네이티브 단계
  - PaaS와 컨테이너를 도입하는 클라우드 친화 단계
  - 데브옵스, CI/CD, MSA를 적용하는 클라우드 네이티브 단계



한국지능정보사회진흥원 클라우드 네이티브 발주자 안내서

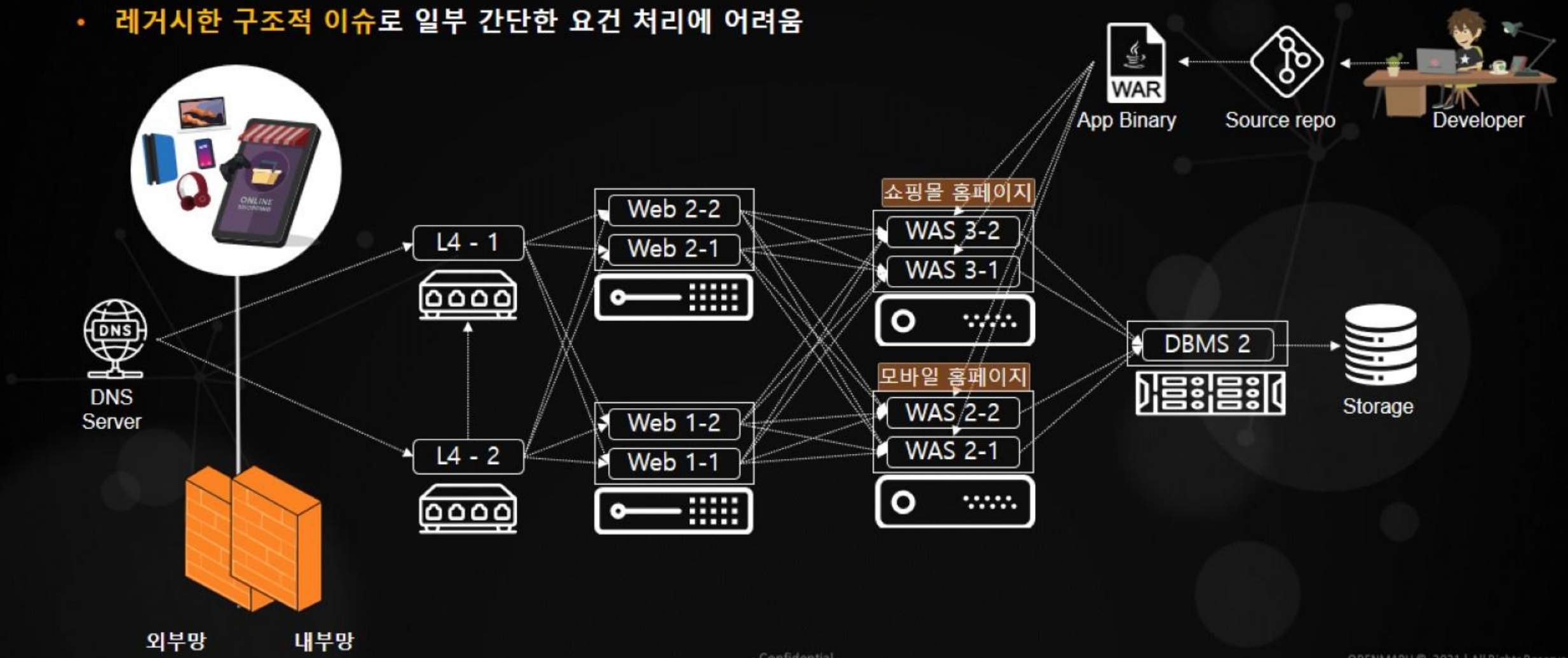
마이크로서비스 아키텍처로의 전환 예시

한국지능정보사회진흥원 클라우드 네이티브 발주자 안내서



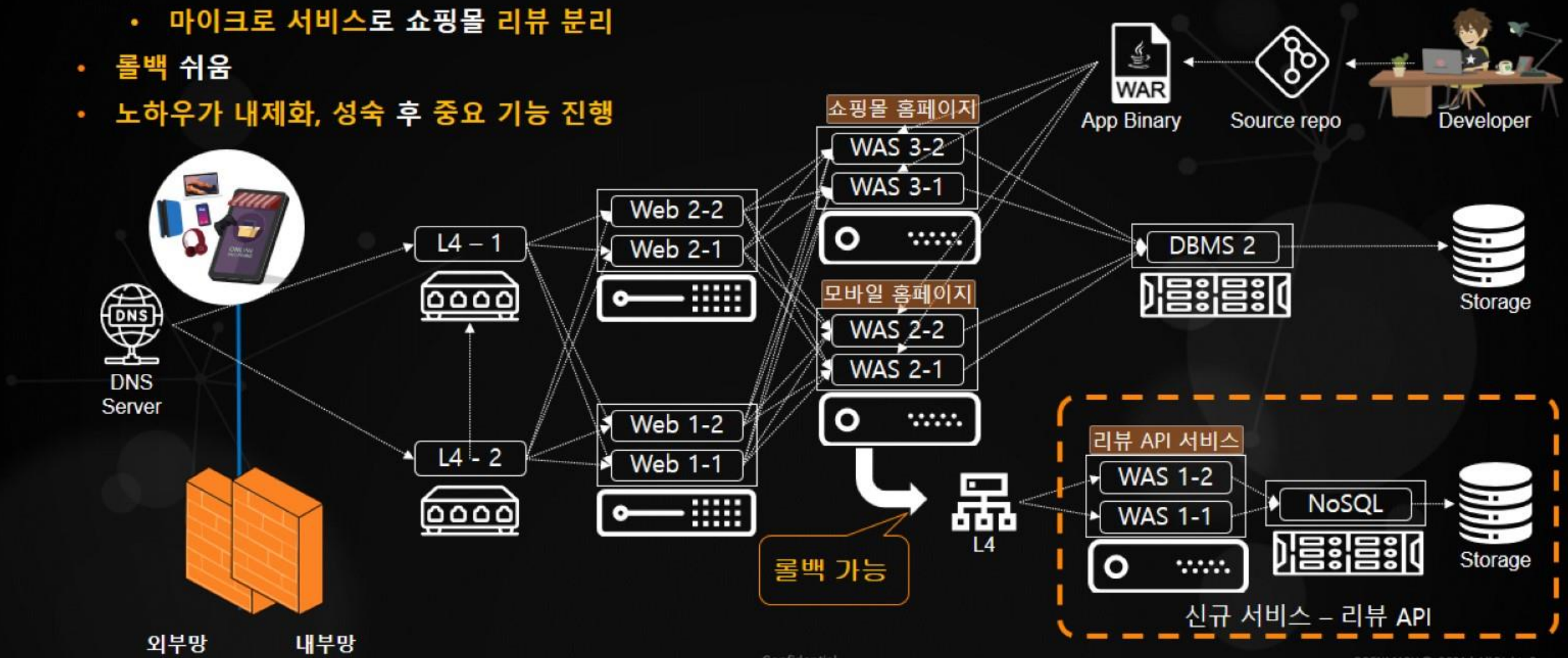
일반화 되어 있는 모놀리틱 아키텍처

- 웹서버, WAS 서버, 데이터베이스 의 역할 별로 티어를 나눈 3 티어 구조
  - 각 티어 별로 수동 확장과 관리, 오토스케일링 및 자동화 부족
- 레거시한 구조적 이슈로 일부 간단한 요건 처리에 어려움

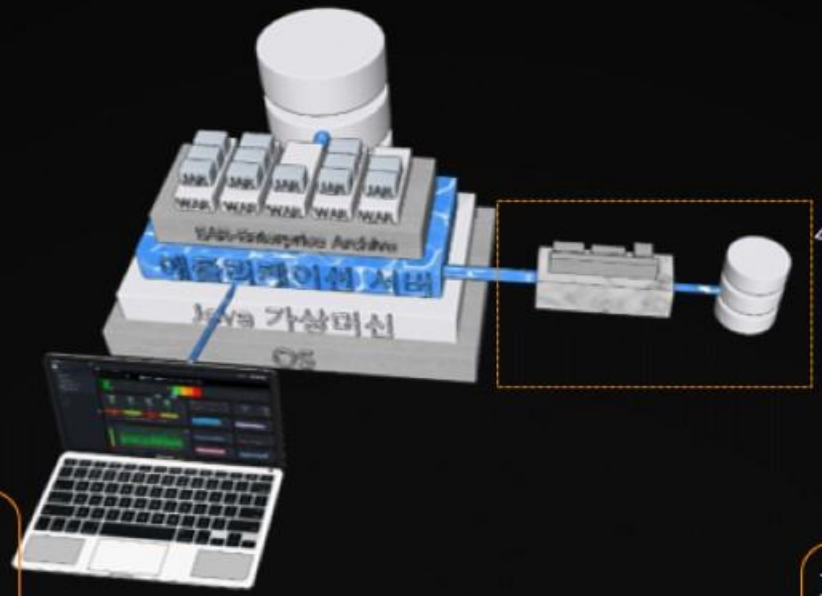


Microservice Pilot – 리뷰API 서비스 도입

- 수많은 기능으로 동작하는 기존 서비스에서 쉬운 부분부터 API로 분리
  - 재설계 시 리스크 부담으로 시도 어려움
  - 마이크로 서비스로 쇼핑몰 리뷰 분리
- 롤백 쉬움
- 노하우가 내제화, 성숙 후 중요 기능 진행



이슈 - Microservice Pilot – 리뷰API 서비스 도입



호출 빈도가 높으며, 독립적으로 동작하는 모듈을 API 서비스로 독립 – 리뷰API

자원(VM) 및 OS/WAS 설치 요청합니다.

빌드/배포 자동화도 해주세요.

내부 API 용 L4 가 필요 해요.

개발팀

전문 엔지니어 일정 확인 해 볼께요.

자원이 여유 있는지 확인 해볼께요.

내부 L4는 없어요.

운영팀

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

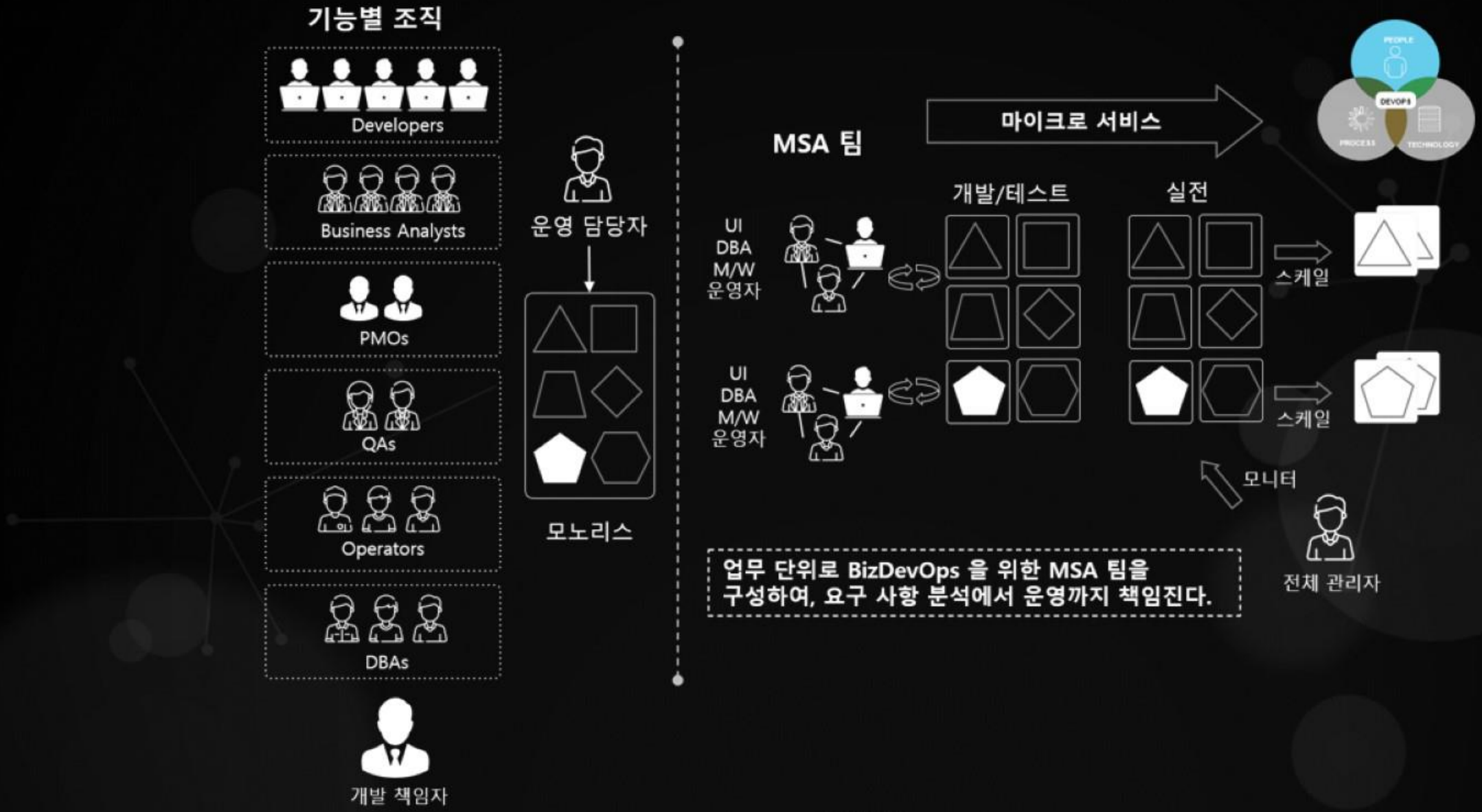
인프라만 클라우드 일 뿐 조직은 그대로

- 기존의 개발팀과 운영팀이 수행하던 역할과 프로세스 그대로 운영

클라우드로 전환했으나 구인난과 고비용 구조로 더 큰 문제

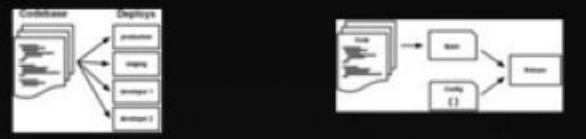
- 클라우드를 이용하고 있음에도 불구하고 수작업 프로세스에서 벗어나지 못함
- 운영 인력 부족과 업무 효율성을 개선하지 못함

시스템 개발과 운영 방법론의 혁신 DevOps



Cloud Native Application(애플리케이션 현대화) 도전!

- 클라우드의 이점을 최대로 활용할 수 있도록 애플리케이션을 구축하고 실행하는 방식
- 신속한 개발과 클라우드 확장성 확보를 위한 클라우드 네이티브 애플리케이션 개발은 필수
  - SaaS 12-Factor, Cloud, MSA, Container, CI/CD 등 DevOps 중요
  - https://landscape.cncf.io/



12-Factor applications

Independent, autonomous, stateless, processes

Configuration sources, service references

Application as composition of polyglot services

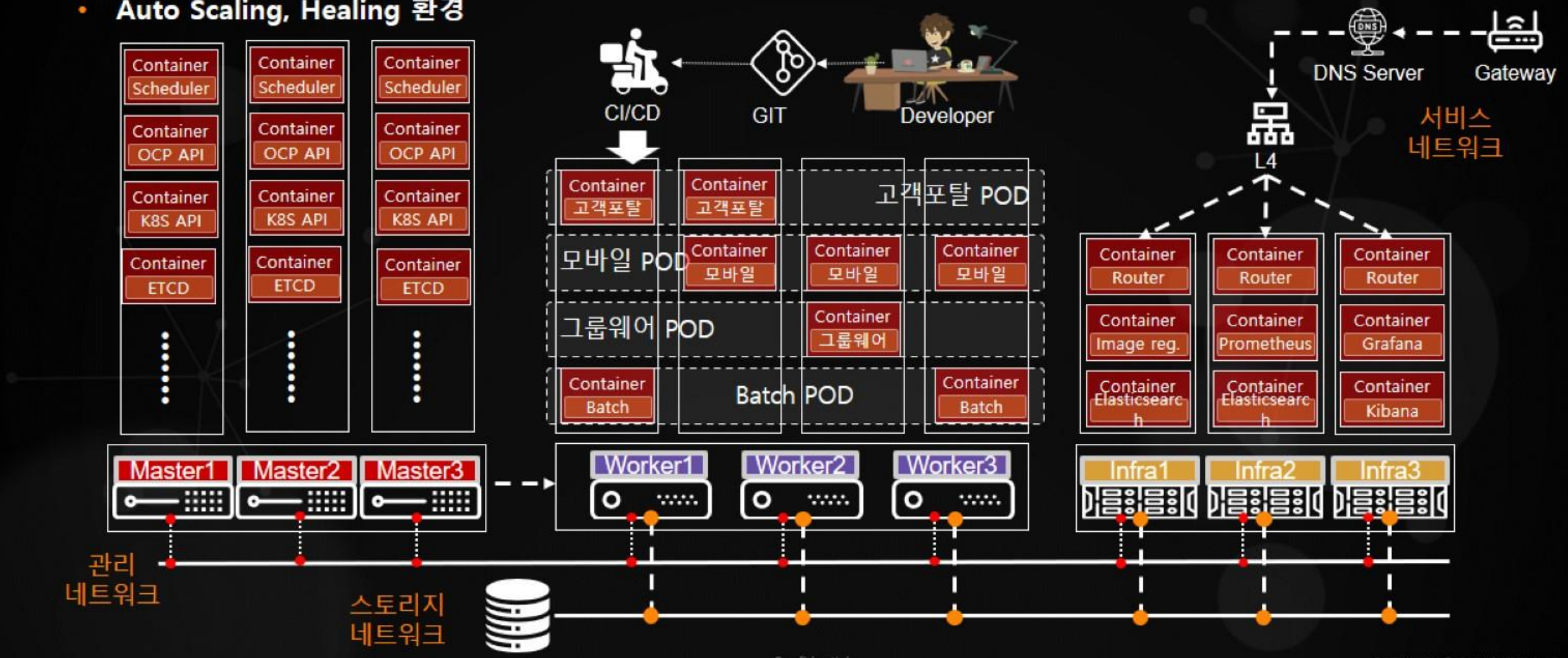
Build, package, deploy, observe

Platform As A Service

클라우드 네이티브를 위한 플랫폼, PaaS

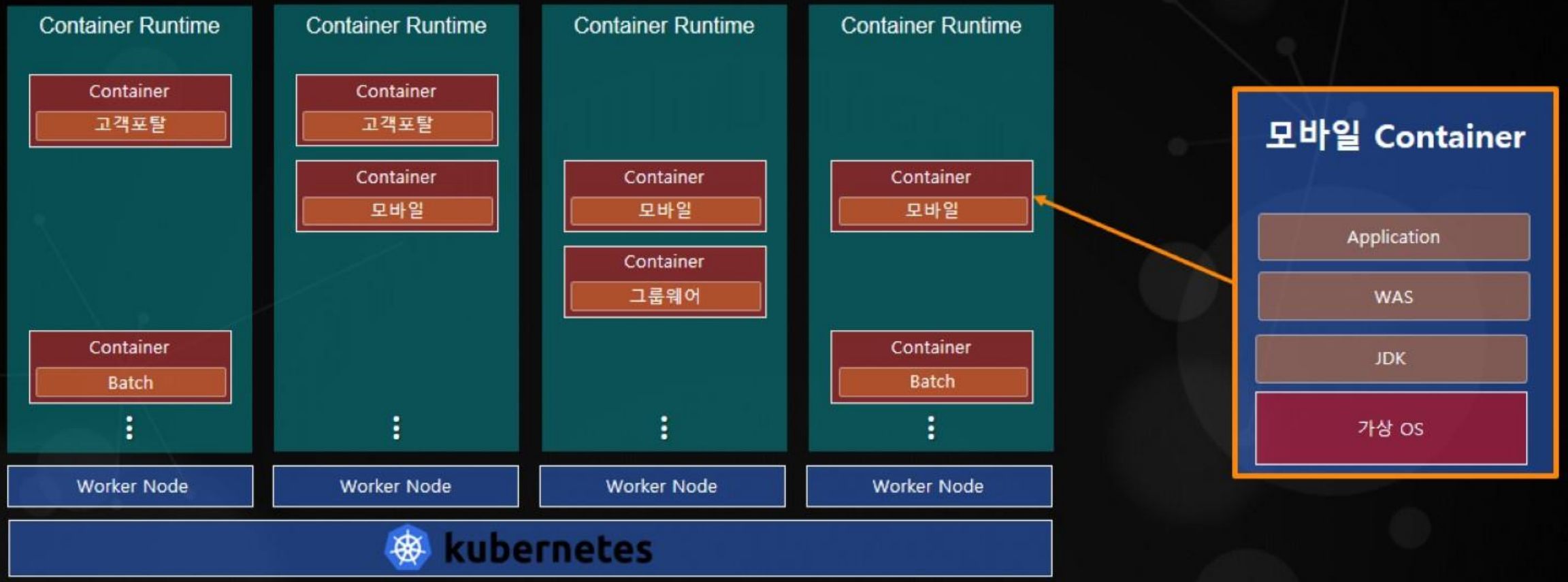
클라우드 네이티브 – 인프라 구성

- 서버별로 업무가 나뉘는 것이 아니고, 스케쥴러에 의해 적재 적소 서버에 애플리케이션 배치
- **Auto Scaling, Healing 환경**



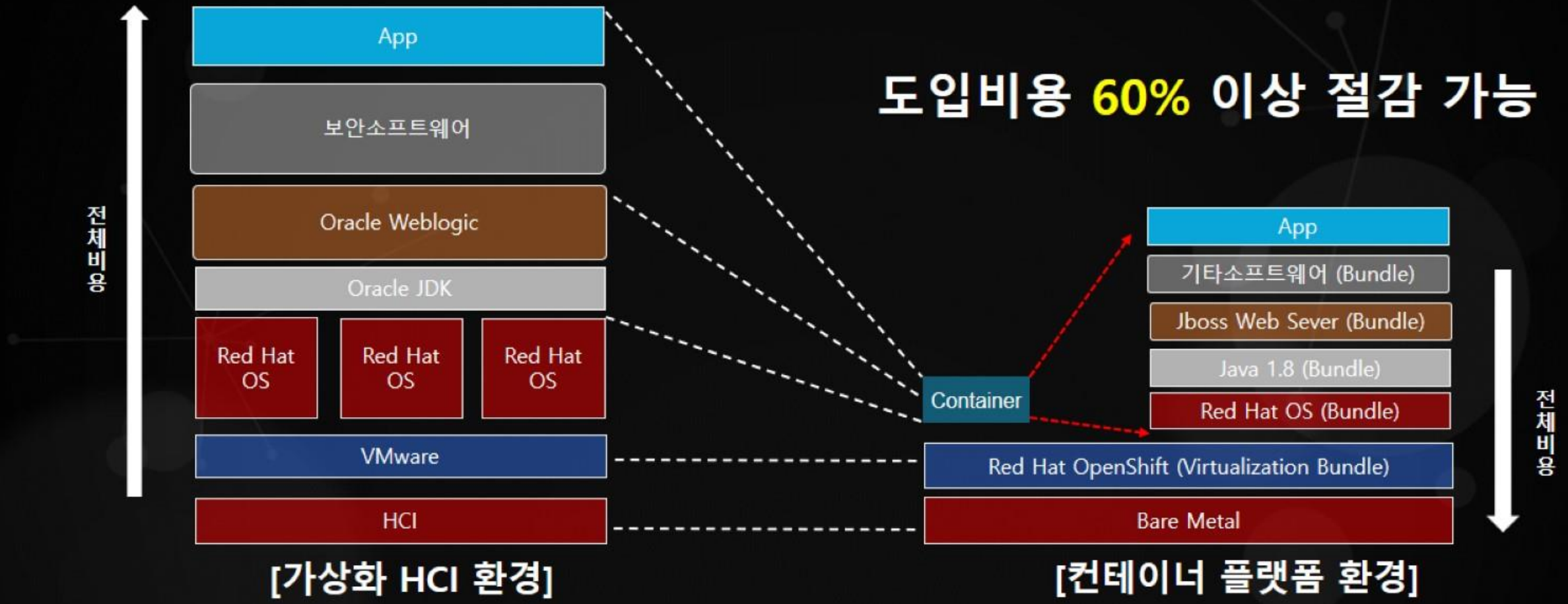
컨테이너 환경의 소프트웨어 컴포넌트 요약 구성

- **Container는 가상OS, JDK, WAS, Application이 포함된 Image를 기반으로 기동됨.**



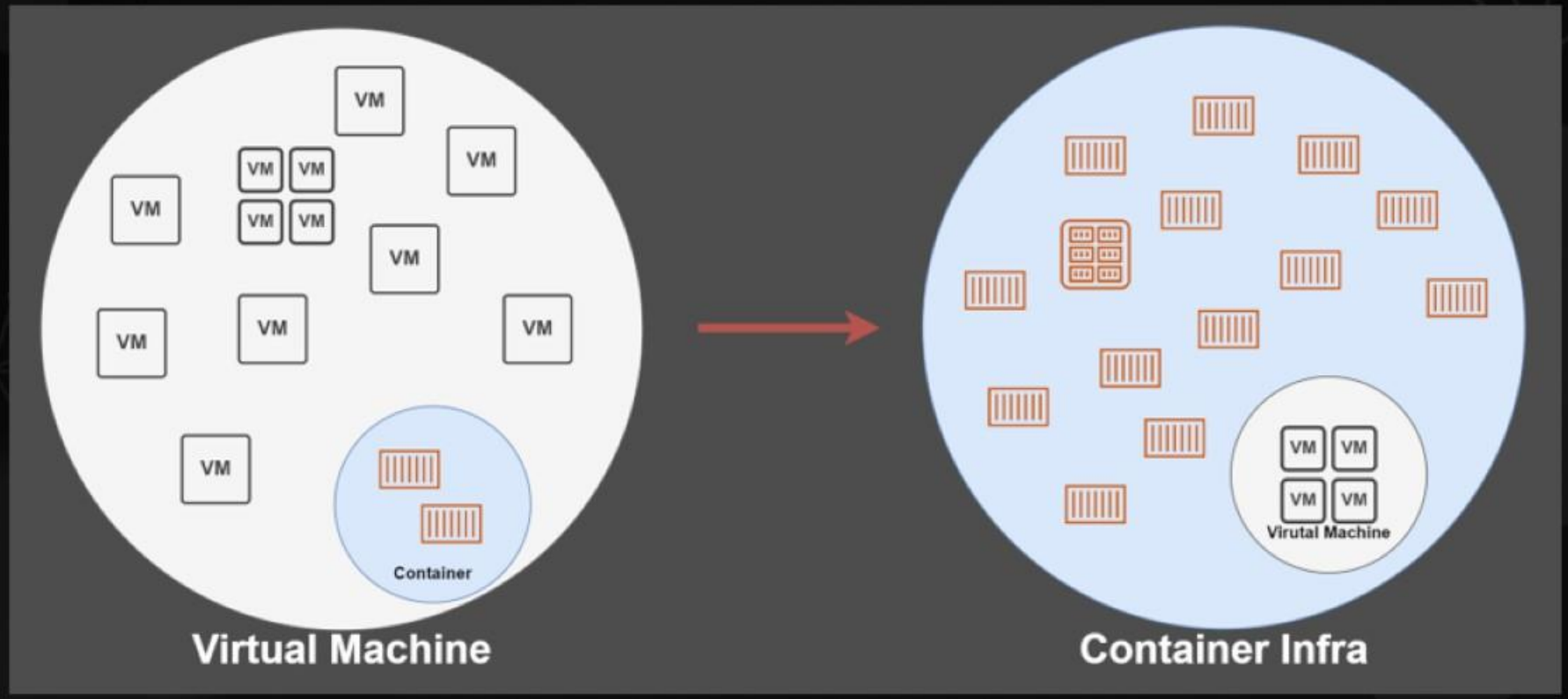
HCI 가상화 VS 컨테이너 플랫폼 금액 비교(2)

- 컨테이너 환경에서의 가상화소프트웨어 / WAS / 보안소프트웨어 비용 제거
- 1 Bare Metal (1 Worker Node - 2 socket / up to 64 cores) 1 copy 기준은 2 Core 1 Copy 기준 대비 90% 비용 절감



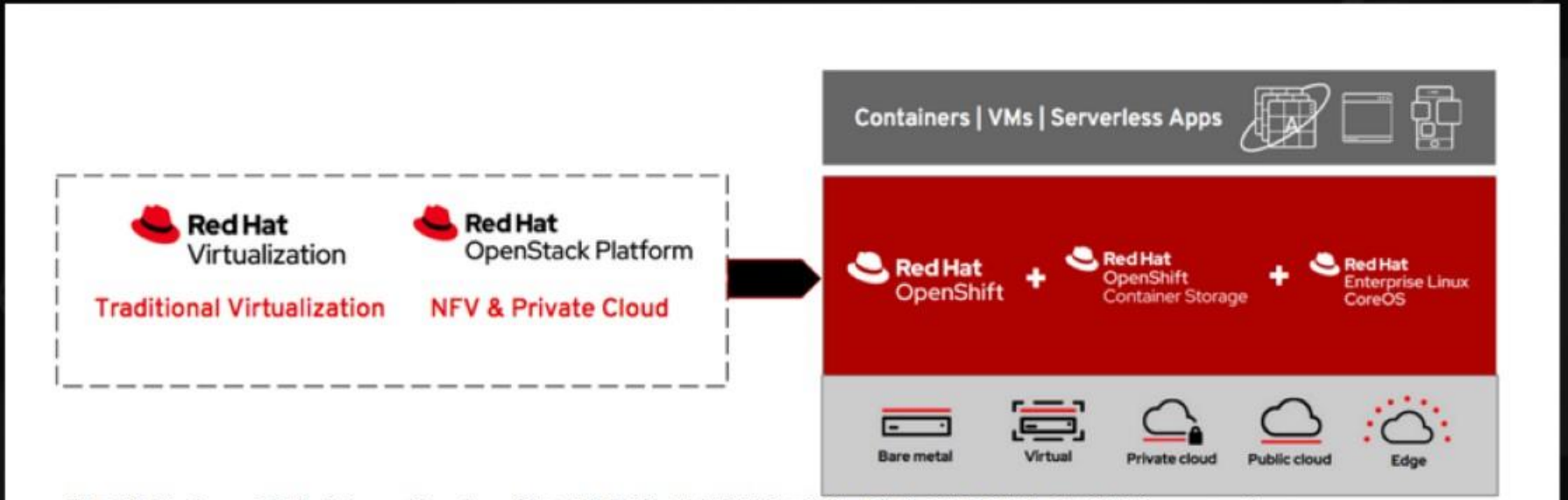
앞으로의 VM과 Container의 비중 변화

- 기존 환경에서 VM의 대다수는 애플리케이션을 기동 시키기 위한 서버
- Container Infra에서는 서비스 애플리케이션은 모두 Container로 기동!
- VM 비중이 축소됨에 따라 Container Infra에서 VM을 기동한다면? 기동하기 위한 솔루션은?



OpenShift Virtualization

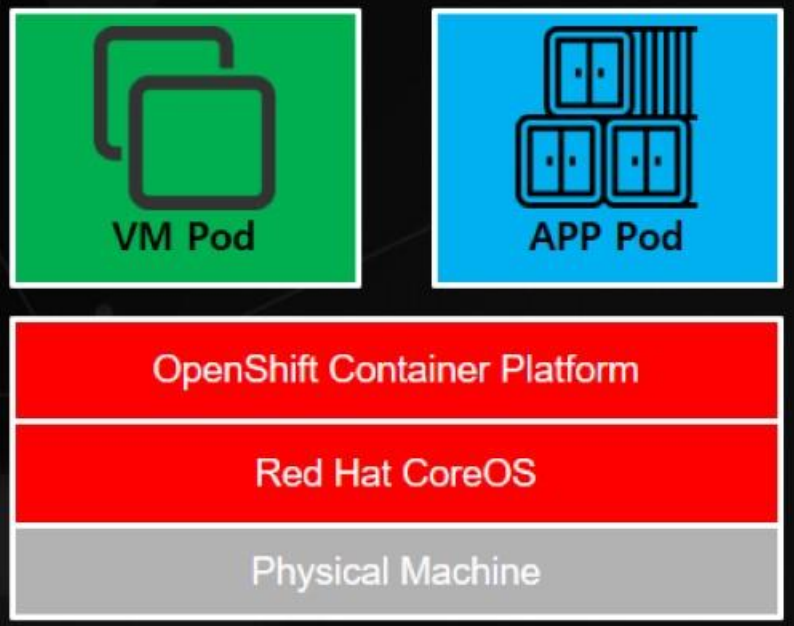
- OpenShift Virtualization을 통해 컨테이너 네이티브 아키텍처에서 가상 머신, 컨테이너를 통합 관리
- OpenShift에서 가상 머신과 컨테이너를 모두 사용함에 따라 애플리케이션을 컨테이너 네이티브에 적합하게 개발
- 컨테이너 기반 애플리케이션과 동일한 플랫폼에서 VM 기반 워크로드를 운영
- Windows 혹은 별도 솔루션 같은 Containerize가 어려운 자원들에 대한 지속 운영 가능



레드햇이 OpenShift Virtualization으로 제공할 가상화의 미래, 현재 전통적인 가상화 integration
https://cloud.redhat.com/blog/blog-openshift-virtualization-whats-new-with-virtualization-from-red-hat

OpenShift Virtualization

- Kubernetes 네이티브방식을 사용하여 가상 머신을 실행하고관리하기위해 KubeVirt를 기본으로 OpenShift에 구축된 가상화 API 및 런타임
- KubeVirt 프로젝트는 Red Hat의 주도하에 개발



- Kubernetes 시스템으로직접 컨테이너화할 수 없는 애플리케이션구성 요소를 전환하는방법을 제공
- VM 리소스를컨테이너네이티브로연결 및 사용
- 가상 머신은 기존 애플리케이션 컨테이너와 동일한 Red Hat OpenShift 노드에서 병렬로 실행
- VMware vSphere및 Red Hat Virtualization 가상 머신을 포함한 기존 가상 머신 가져 오기 및 복제 지원
- k8s 오케스트레이션, 관리를 통한 성숙하고 안정적인 KVM 기반 가상화

OpenShift Virtualization의 구매비용?

- OpenShift Virtualization은 별도의 제품이 아닌 기능
- OpenShift Container Platform과 OpenShift Container Engine에 번들된 기능
- OpenShift 운영시 별도의 서브스크립션, 제품 구매비용 발생하지 않음
- OCP에 Red Hat Enterprise Linux (RHEL) Virtual Datacenter Subscription이 포함되어 있기 때문에 RHEL Guest OS 역시 추가 금액이 발생하지 않음.

Question: Is OpenShift Virtualization a product?	**Answer:** OpenShift Virtualization is a feature, not a product. It is based on the upstream open source KubeVirt project and is available to download as a Red Hat OpenShift operator. More information on how to get and install the OpenShift Virtualization operator can be found in the OpenShift Virtualization documentation.
Question: How will OpenShift Virtualization be made available?	**Answer:** OpenShift Virtualization is a feature of Red Hat OpenShift Container Platform and Red Hat OpenShift Kubernetes Engine. It is not an add-on or a separate product. The OpenShift Virtualization operator must be installed to access the feature. All current and future subscribers receive OpenShift Virtualization as part of their Red Hat OpenShift subscription. OpenShift Virtualization is a feature of Red Hat OpenShift Container Platform and Red Hat OpenShift Kubernetes Engine. It is not an add-on or a separate product. The OpenShift Virtualization operator must be installed to access the feature. All current and future subscribers receive OpenShift Virtualization as part of their Red Hat OpenShift subscription.

https://www.redhat.com/en/resources/openshift-virtualization-faq

Virtual Machines · Red Hat Ope

주의 요함 | https://console-openshift-console.apps.ocp4.ov.com/k8s/ns/comodo/kubevirt.io~v1~VirtualMachine

openmaru APM

Red Hat OpenShift Container Platform

31 admin

Administrator

Home

Operators

Workloads

Virtualization

Overview

VirtualMachines

Templates

Networking

Storage

Builds

Observe

Compute

User Management

Administration

Project: comodo

Virtual Machines

Launch Migration Tool | Create

With Wizard

With YAML

No virtual machines found

See the templates tab to quickly create a virtual machine from the available templates.

Create virtual machine

Learn how to use virtual machines

OpenShift내에 가상머신 생성

console-openshift-console.apps.ocp4.ov.com/k8s/ns/comodo/.../wizard?namespace=como...

Platform As A Service

클라우드 네이티브를 위한 고려사항

PaaS를 운영할 때 필요한 인프라 리스트

- 고객분들의 오해 : 플랫폼이니까 PaaS만 설치만 하면 된다?

분류	이유	역할	비고
DNS	• Cloud 에 신규서비스 추가 시 배포와 함께 서비스가 제공 되어야함 • 플랫폼 외부 네트워크에서 접속 시 도메인 기반의 애플리케이션 통신	• OpenShift Router Domain 등록	• New_App.apps.example.co.kr
L4	• 모든 Node가 Active 형태로 고가용성으로 구성됨에 따라 VIP, 헬스체크, 부하분산 필요	• Master Node, Infra Node VIP 및 로드밸런싱	
GIT	• 소스코드 저장방식의 고도화된 브랜치 전략을 가져갈 수 있음 • 소스코드만으로 컨테이너 이미지까지 빌드되는 OpenShift S2I 기능을 100% 활용	• OpenShift S2I(Source-To-Image)를 위한 소스 저장소	
Storage	• 애플리케이션들이 NAS처럼 활용 할 수 있는 공유 스토리지 • EFK, Monitoring, Image registry 등 OpenShift Infra 자원이 사용할 스토리지 공간	• 서비스가 이용할 공유 스토리지 및 OpenShift 운영 데이터 저장소	
Network	• 컨테이너 플랫폼을 구성하는 인프라 파드들과 애플리케이션 파드들도 기존 환경과는 달리 매우 많은 수가 기동되는 환경임에 따라 넓은 대역폭이 권고	• 10G 네트워크	

클라우드 네이티브 환경의 애플리케이션 모니터링 어려움

openmaru

6. Monitor, log and troubleshoot from the start

The construction of applications from a set of microservice Legos considerably complicates how to monitor and troubleshoot systems and their performance. Various microservices often trigger a cascade of events that leads to an application failure. To minimize failures -- which aren't a *maybe*, but a reality -- incorporate monitoring and troubleshooting into microservices design.

https://www.techtarget.com/searchitoperations/tip/Follow-these-6-steps-to-deploy-microservices-in-production

- 마이크로 서비스 아키텍처일수록 **모니터링 방법이 상당히 복잡**해 집니다.
- 마이크로 서비스 아키텍처에 **모니터링과 트러블 슈팅 방법이 고려**되어야 합니다.

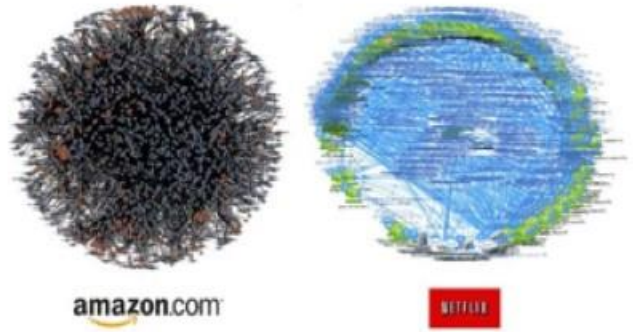
- Uber는 2014년 말 에 4,000개가 넘는 독점 마이크로 서비스와 점점 더 많은 수의 오픈 소스 시스템이 모니터링 시스템에 문제를 제기했다고 보고
- 컨테이너 인프라는 모니터링 시스템이 필수적인 환경
- **마이크로 서비스에 특화된 모니터링 시스템이 필요**

2. Microservices instead of a monolith

Following a microservice architecture, a typical monolith application would be broken down into a dozen or more microservices, each one potentially running its own programming language and database, each one independently deployed, scaled and upgraded.

Uber for example reported in late 2014 over 4,000 proprietary microservices and a growing number of open source systems which posed a challenge for their monitoring system.

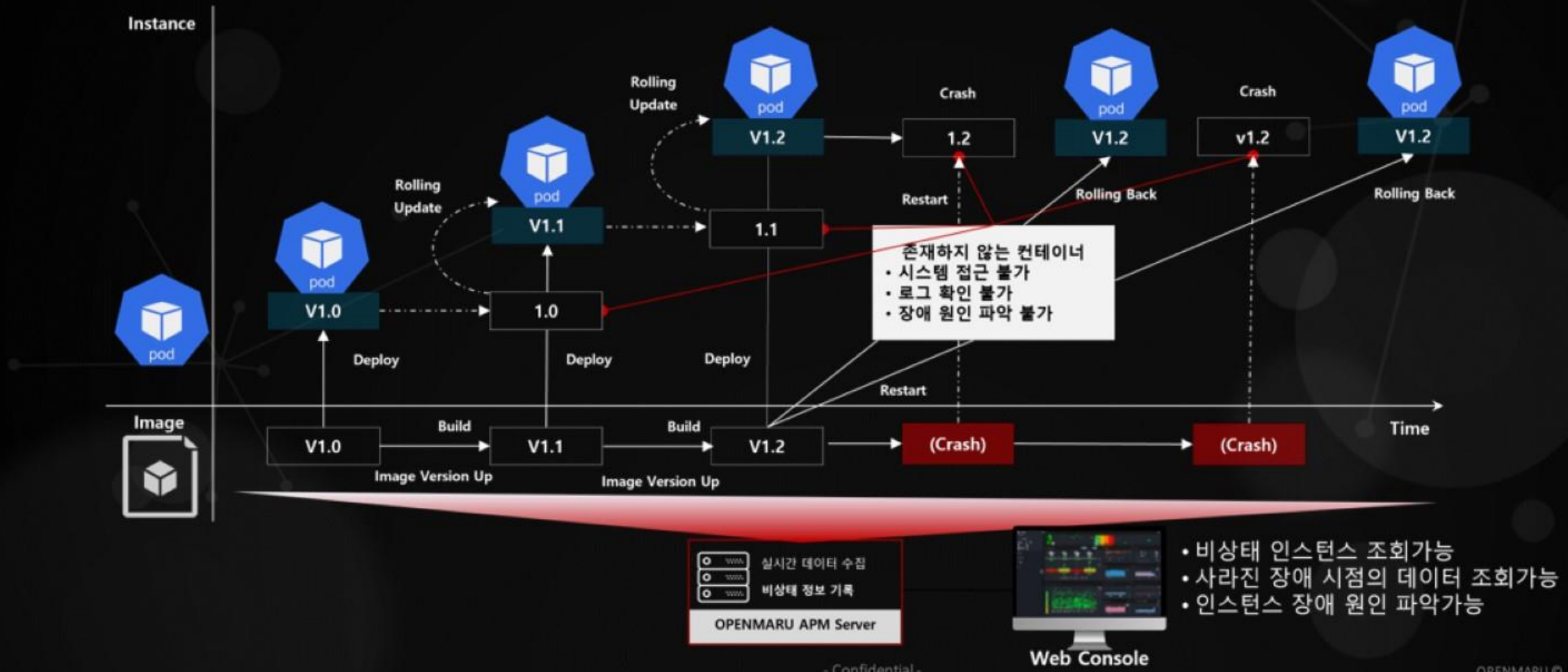
The Challenge: A surge in the number of discrete components you need to monitor.



https://www.infoq.com/articles/microservice-monitoring-right-way

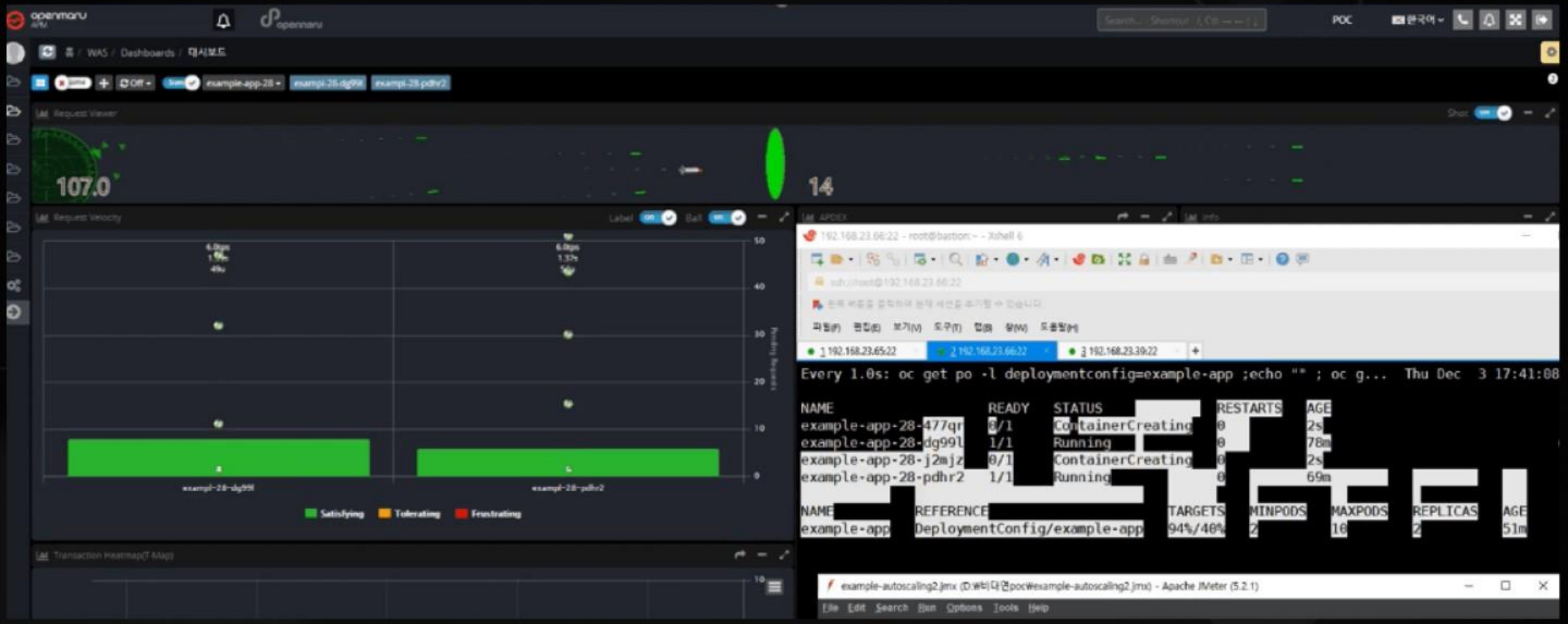
컨테이너는 무상태 (Immutable) 인프라스트럭처로 상태를 저장하지 않음

- PaaS 환경에서 컨테이너가 중지된 후 장애원인 파악을 위한 방법을 제공
- APM 에서 사라진 컨테이너에 대한 정보를 보관하여 장애원인을 파악할 수 있음



컨테이너 환경에서의 애플리케이션 모니터링

- **Auto Scaling으로 Container(Pod)가 유연하게 Scale Out/In이 발생하는 환경**
- **클라우드 네이티브 애플리케이션을 트러블 슈팅할 수 있는 기능이 있는 모니터링 시스템이 필요함**




openmaru

제품 / 서비스에 관한 문의

- 콜 센터 : 02-469-5426 (휴대폰 : 010-2243-3394)
- 전자 메일 : sales@opennaru.com