
openmaru
APM
클라우드 네이티브 환경으로 변화하였을 때의 특징

목차

- Confidential -

OPENMARU © 2022 | All Rights Reserved.

Platform As A Service

개발/운영팀은 왜 클라우드 네이티브를 도입하는 걸까요?

클라우드 네이티브 란?

openmaru Red Hat

- 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

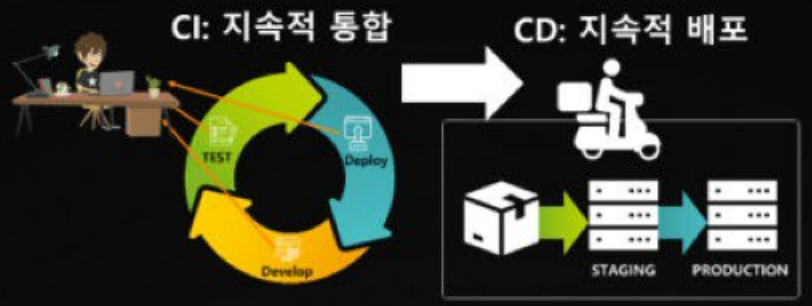
가상화 기술 중 하나로, **시스템을 가상화 하는 것이 아니라** 애플리케이션을 구동할 수 있는 컴퓨팅 작업을 패키징하여 **OS를 가상화** 한 것입니다.

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

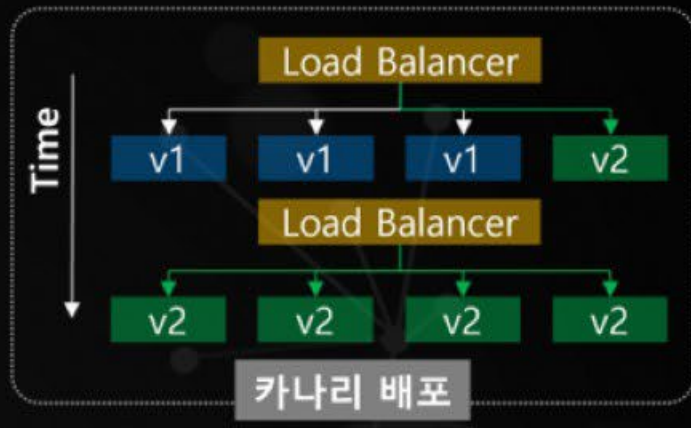
OPENMARU © 2022 | All Rights Reserved.

클라우드 네이티브 기반 환경의 장점

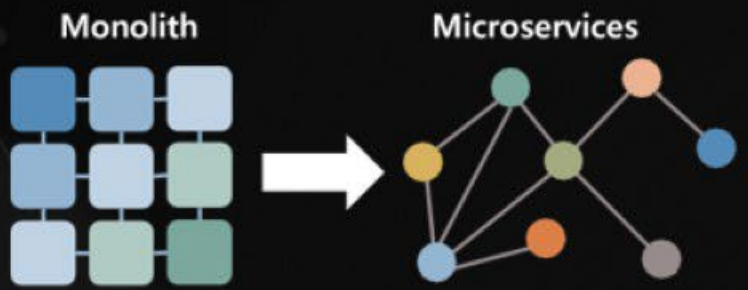
컨테이너 기반의 빠른 개발환경



신속한 개발과 편리한 배포



서비스 무중단



MSA개발에 적합한 환경

DevOps기반의 민첩한 개발

OPENMARU © 2022 | All Rights Reserved.

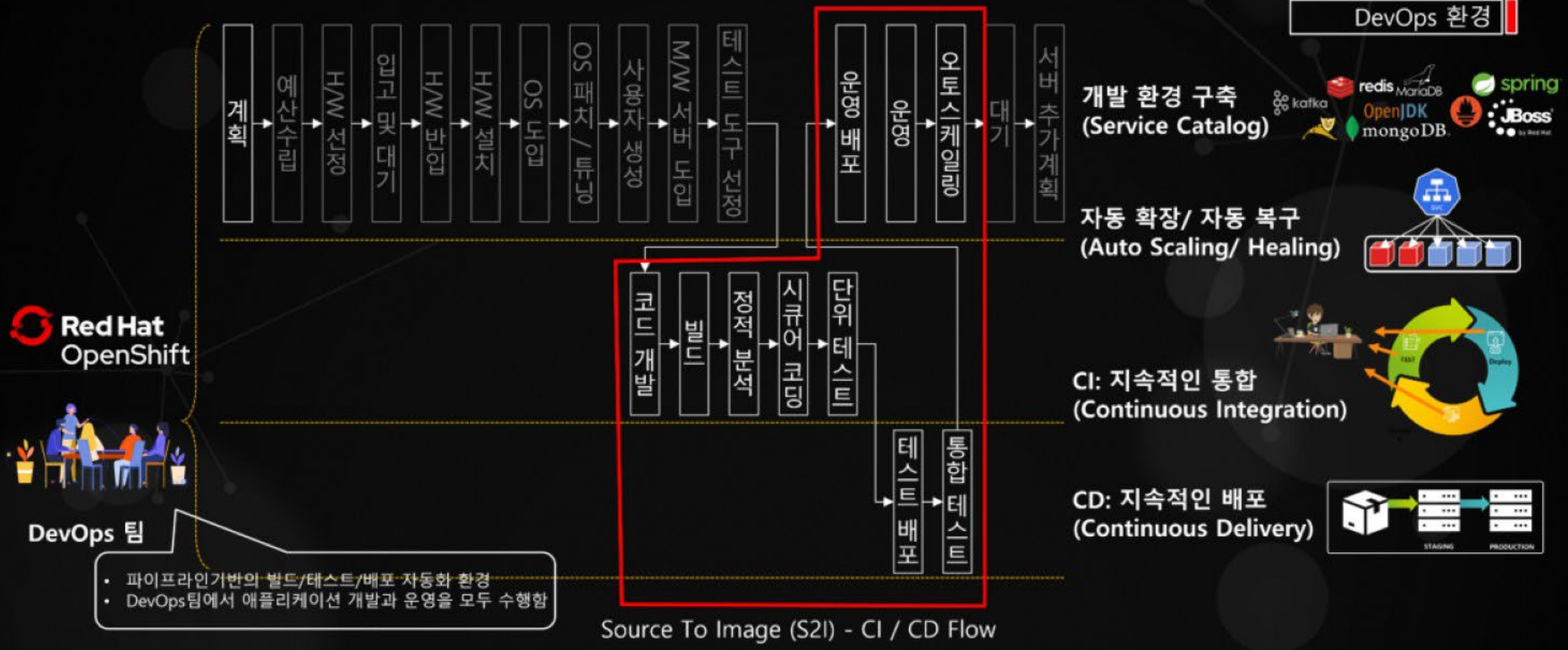
개발팀과 운영팀 누구의 잘못인가? 불행의 시작

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

OPENMARU © 2022 | All Rights Reserved.

컨테이너 자동화 도구를 이용한 프로세스

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



OPENMARU © 2022 | All Rights Reserved.

10억 명 이상의 클라우드 서비스

openmaru

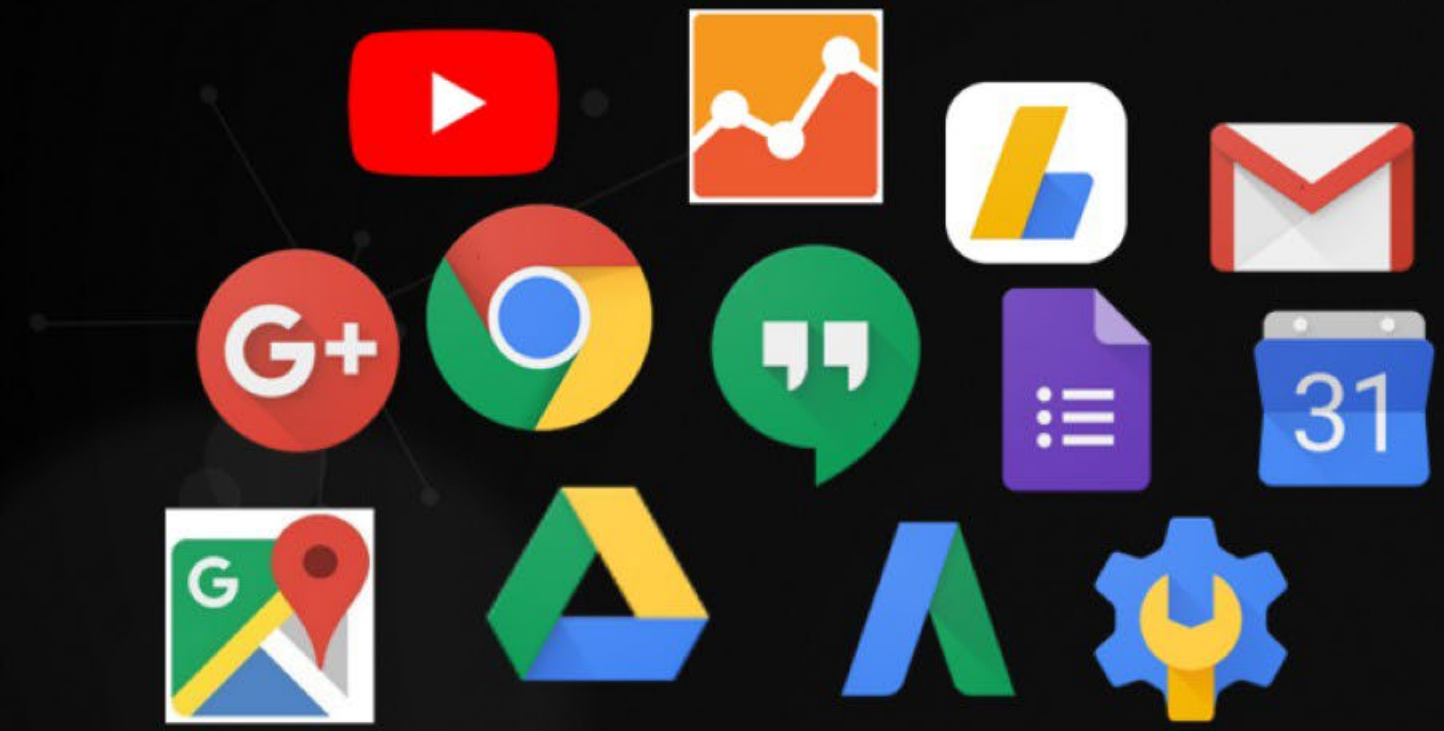
Google Cloud

- 2조 / 매년 - 구글 검색
- 650억/ 매년 - Google PlayStore 앱 다운로드
- 10억 명/ 매월 - 모바일 Chrome 브라우저.
- 2억 명/ 매월 - Google 포토

Google 는 모두 컨테이너에서 실행

openmaru Red Hat

- Gmail , 검색, 지도 ...
- MapReduce , GFS , Colossus ...
- Google Compute Engine 가상 머신도 컨테이너에서 실행!
- 매주 20 억개 이상의 컨테이너를 실행 중



- Confidential -

GOOGLE 과 컨테이너

openmaru Red Hat

- **Google의 업무 방식**

Gmail에서 YouTube, 검색에 이르기까지 Google의 모든 제품은 컨테이너에서 실행됩니다.

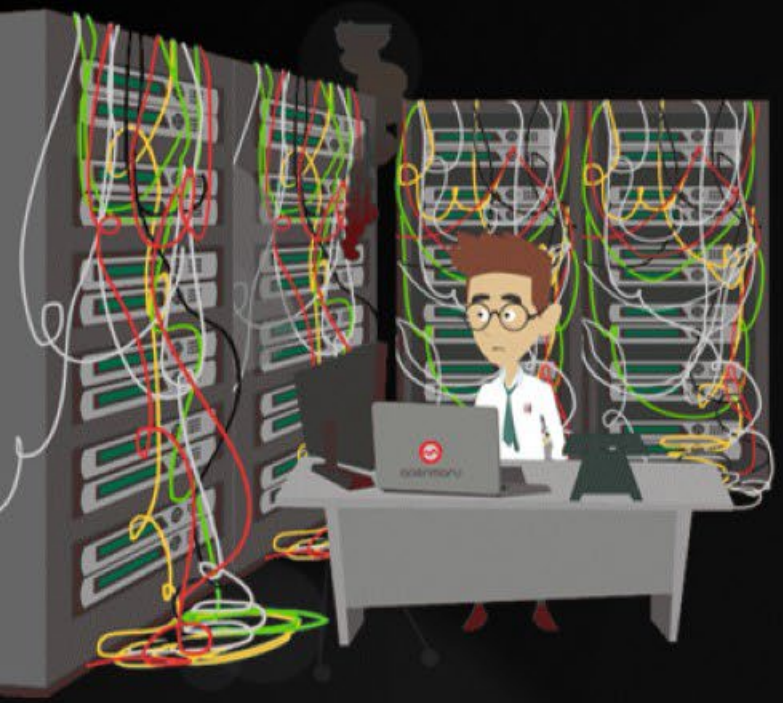
개발팀은 컨테이너화를 통해 더욱 신속하게 움직이고, 효율적으로 소프트웨어를 배포하며 전례 없는 수준의 확장성을 확보할 수 있게 되었습니다. **Google은 매주 수십억 개가 넘는 컨테이너를 생성**합니다. **지난 10여 년간 프로덕션 환경에서 컨테이너화된 워크로드를 실행하는 방법에 관해 많은 경험**을 쌓으면서 Google은 커뮤니티에 계속 이 지식을 공유해 왔습니다.

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/

시스템 비대화로 작업 폭증과 인력 부족 어떻게 할까요?

openmaru Red Hat



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

OPENMARU © 2022 | All Rights Reserved.

클라우드 네이티브 기반을 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

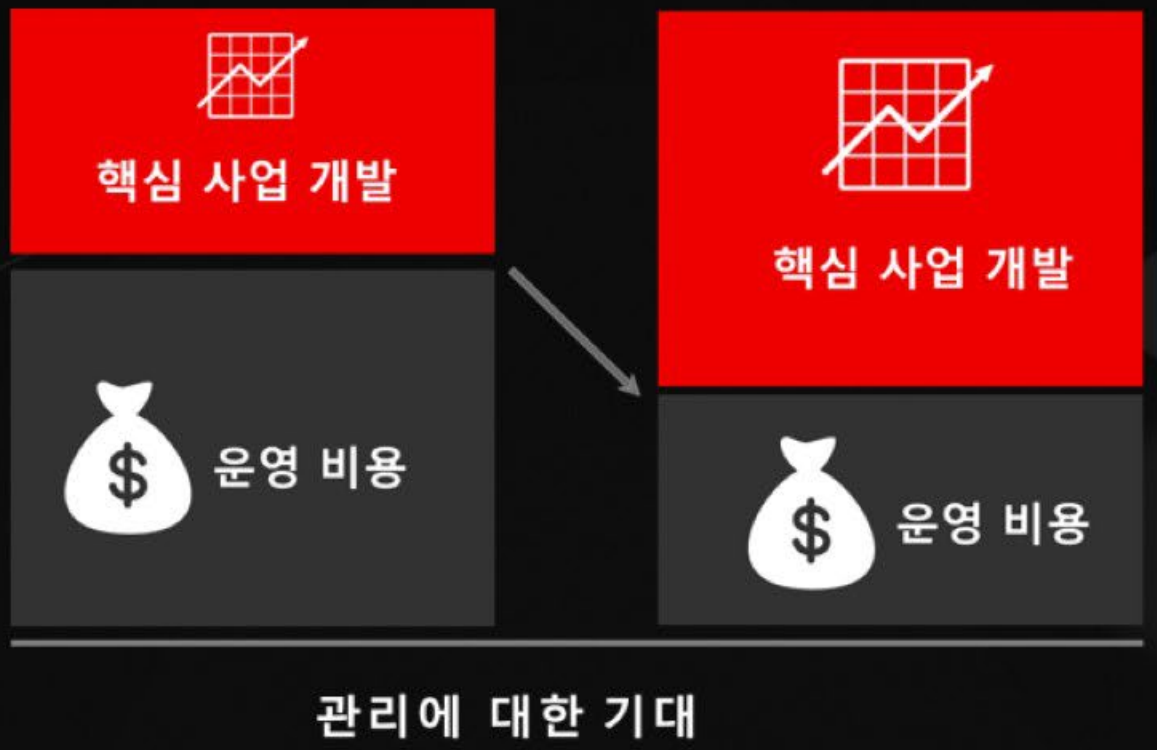
- 운영 기술 표준화를 통한 준비 시간 및 작업 시간 감소
- 시스템 일괄 설정 작업 시간 단축
- 시스템을 통한 작업으로 Human Error감소

OPENMARU © 2022 | All Rights Reserved.

PaaS 를 통한 사업 비용 절감

openmaru Red Hat

- 매니지드 서비스로 운영 비용을 줄이고 귀중한 인적 자원을 핵심 사업의 개발에 집중

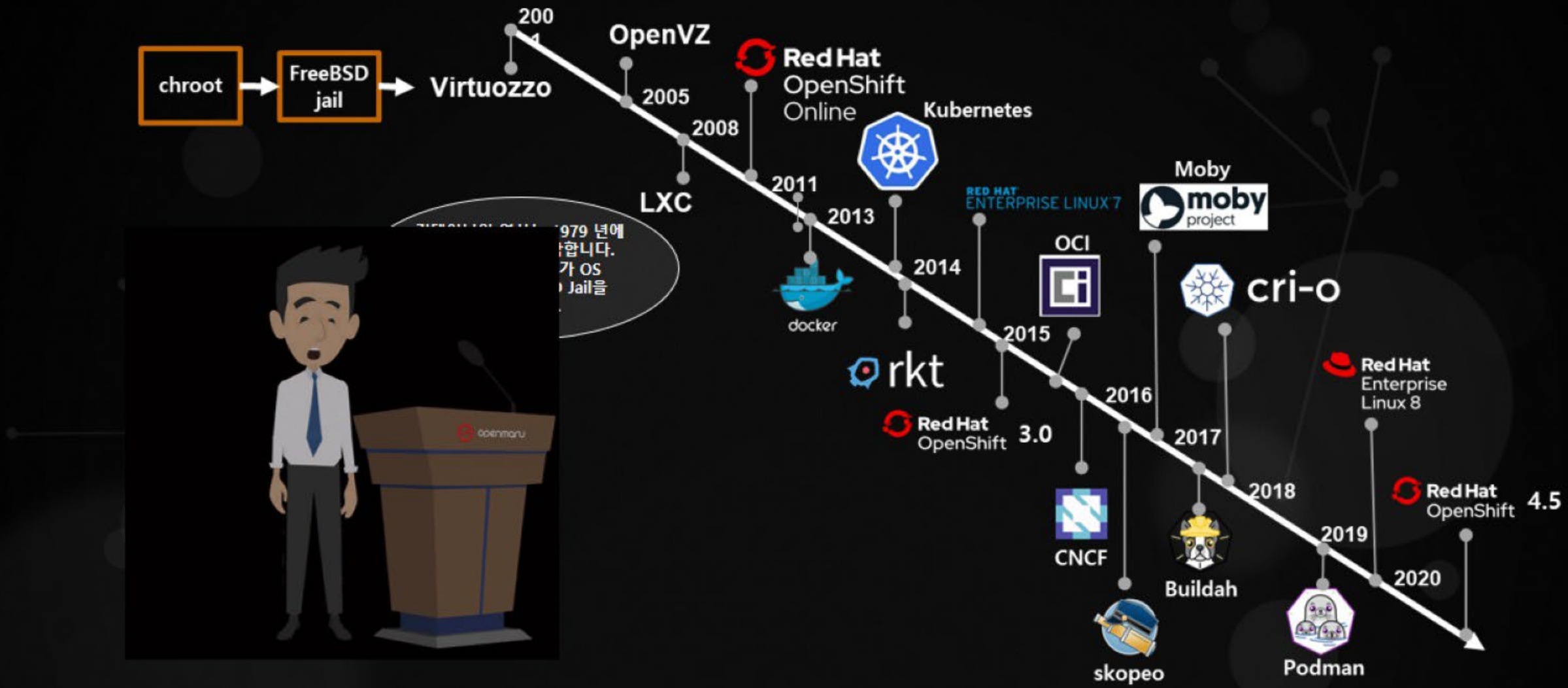


- Confidential -

OPENMARU © 2022 | All Rights Reserved.

Container와 Kubernetes의 역사

openmaru Red Hat

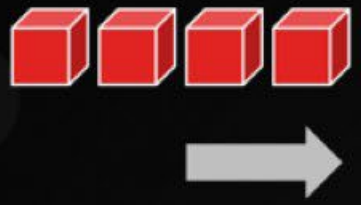


OPENMARU © 2022 | All Rights Reserved.

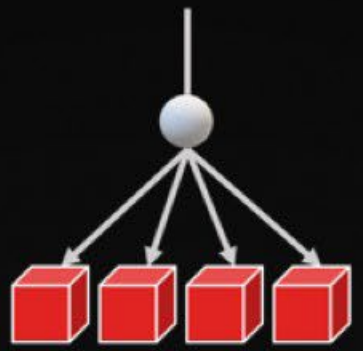
Kubernetes 주요 기능

openmaru Red Hat

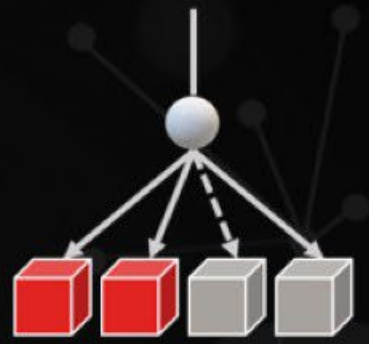
Scale Out / In



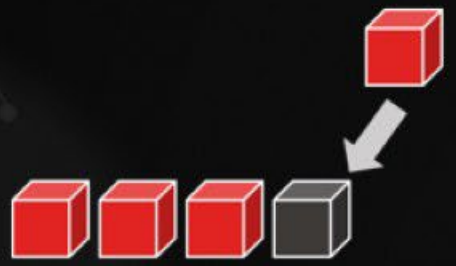
Load Balancer



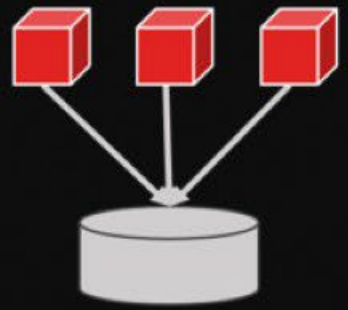
Rolling Update



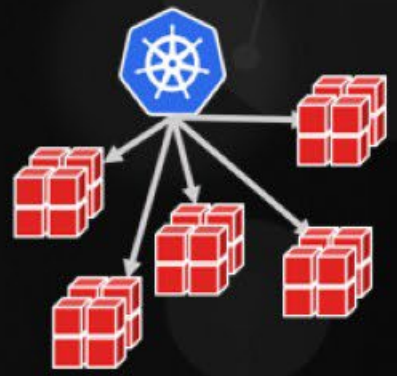
Auto Healing



Persistence Volume



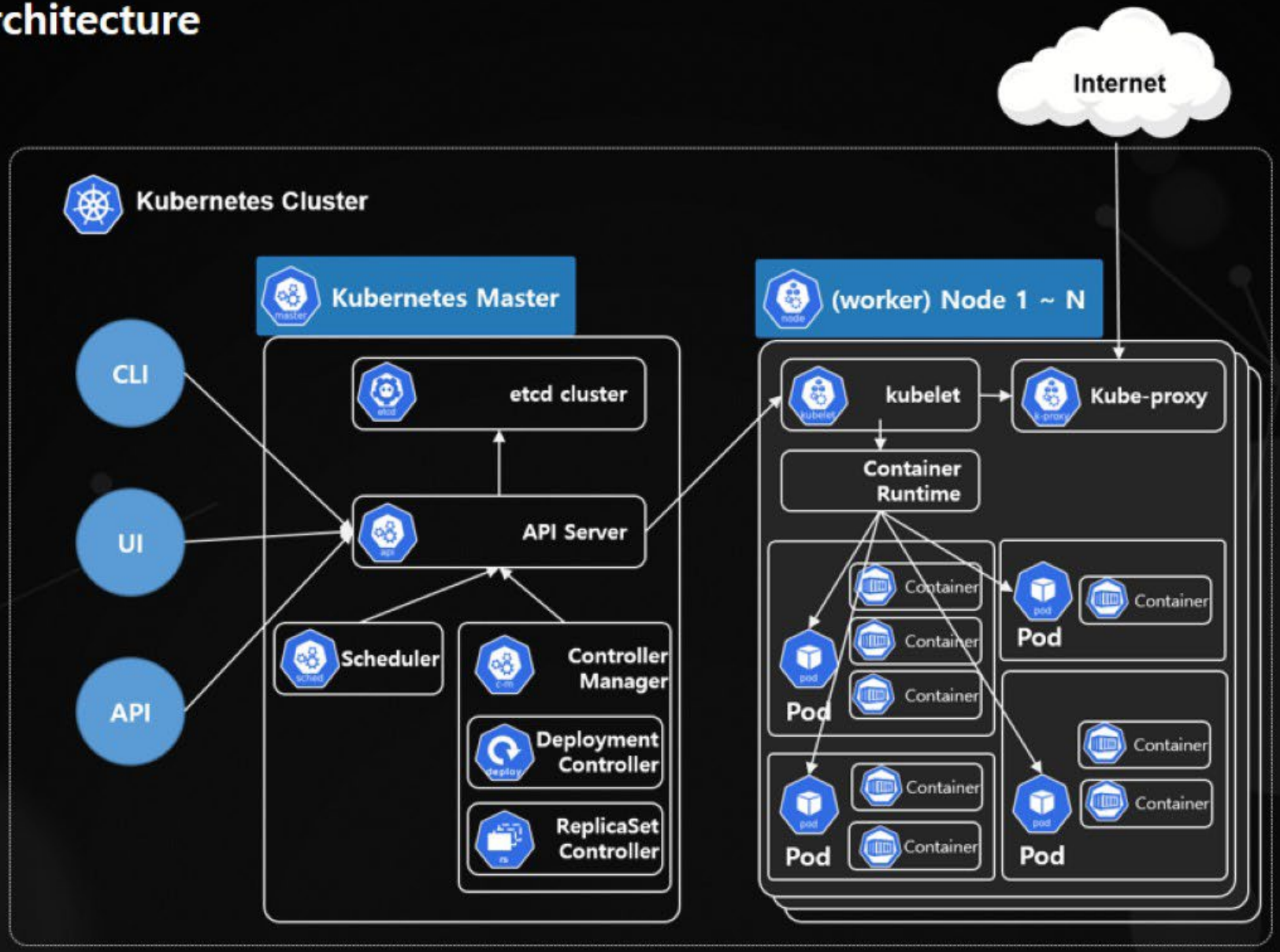
Container Orchestration



- Confidential -

OPENMARU © 2022 | All Rights Reserved.

Kubernetes Architecture



OPENMARU © 2022 | All Rights Reserved.

클라우드 네이티브 기반의 PaaS 도입 혜택

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

OPENMARU © 2022 | All Rights Reserved.

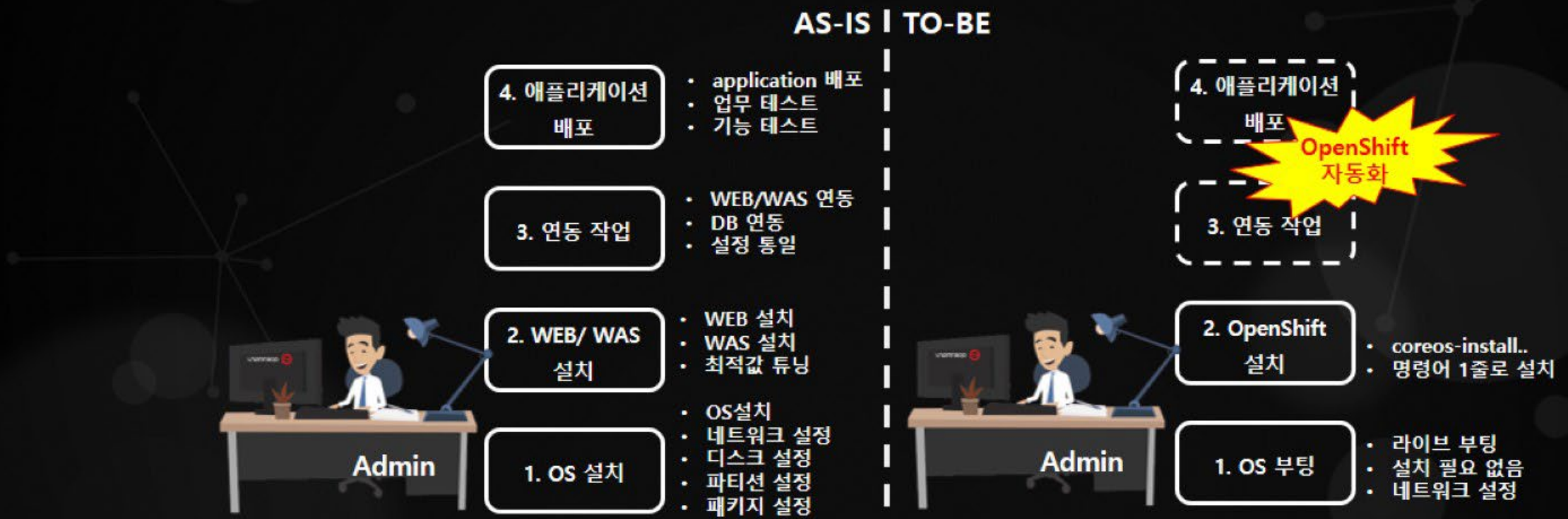
Platform As A Service

클라우드 네이티브의 특징으로 인하여 작업이 어떻게 바뀔까요?

서버 확장 운영관리 비교

인프라 팀(확장성)

- 기존 환경을 확장하기 위해서는 WEB/WAS 설치 및 연동 모든 구간 수작업 필요
- 기존 환경과 동일한 설정을 수작업으로 유지해야함
- 컨테이너 환경은 커맨드 기준 2~3줄의 Worker Node 추가만 하면 작업 완료



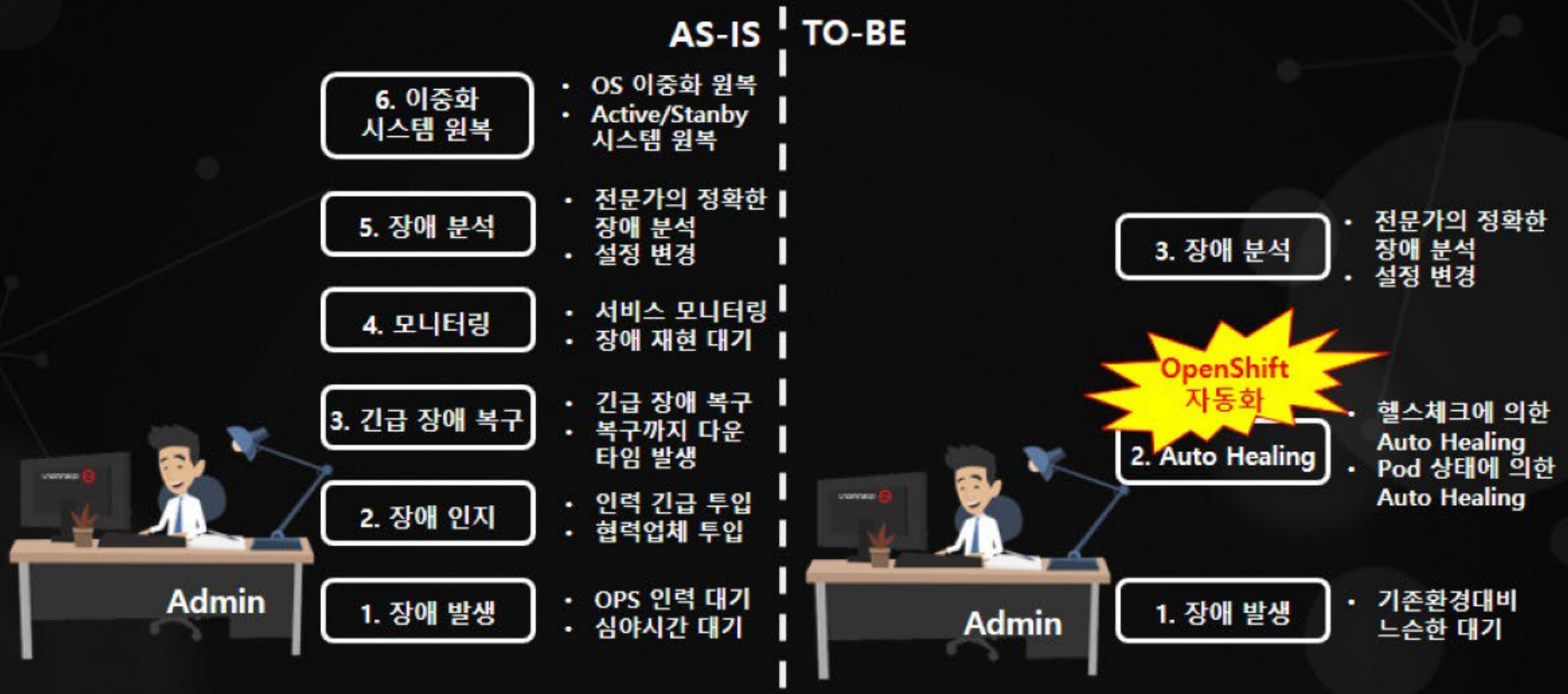
OPENMARU © 2022 | All Rights Reserved.

장애 상황 운영관리 비교

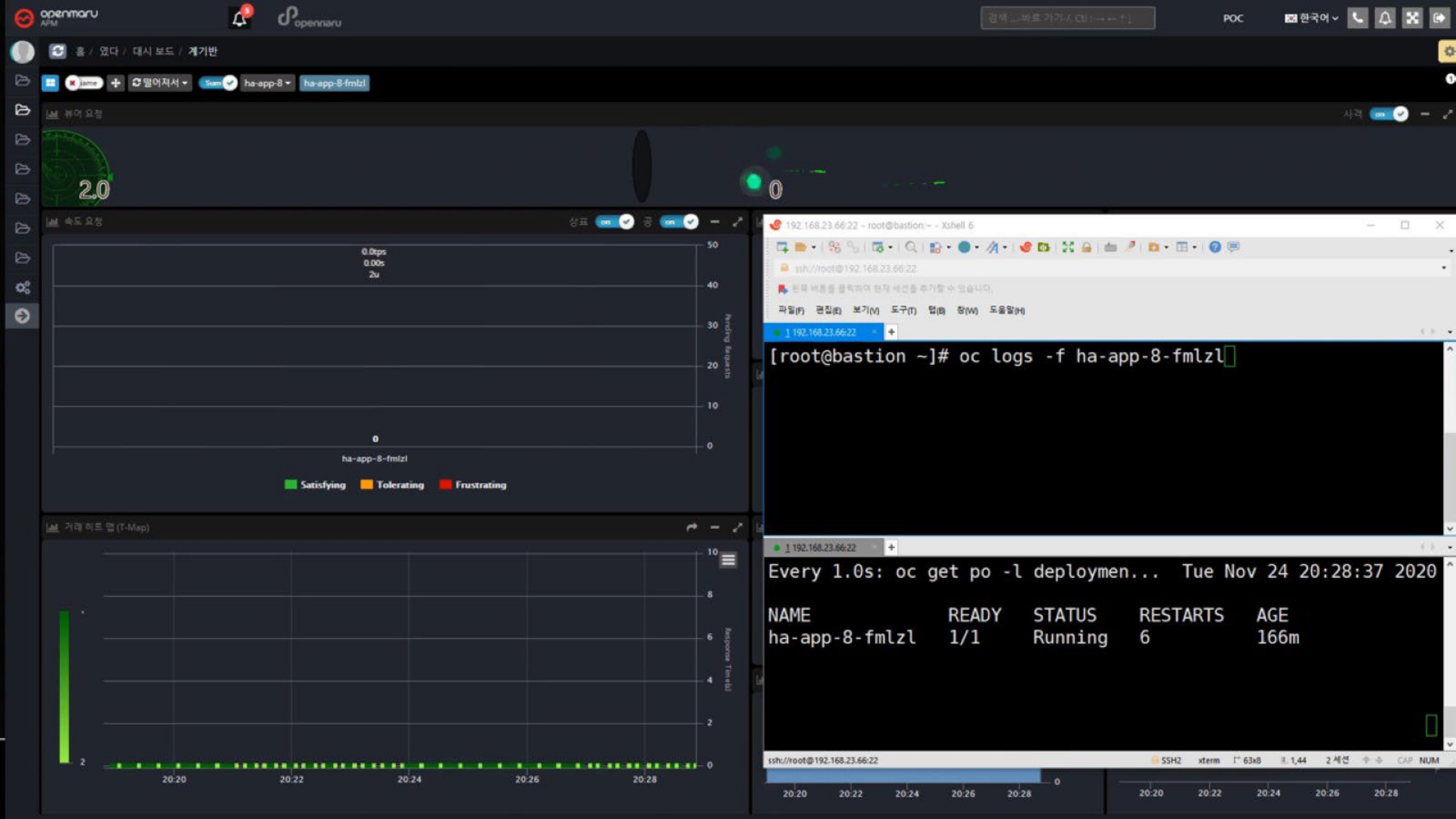
openmaru Red Hat

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 © 2022 | All Rights Reserved.

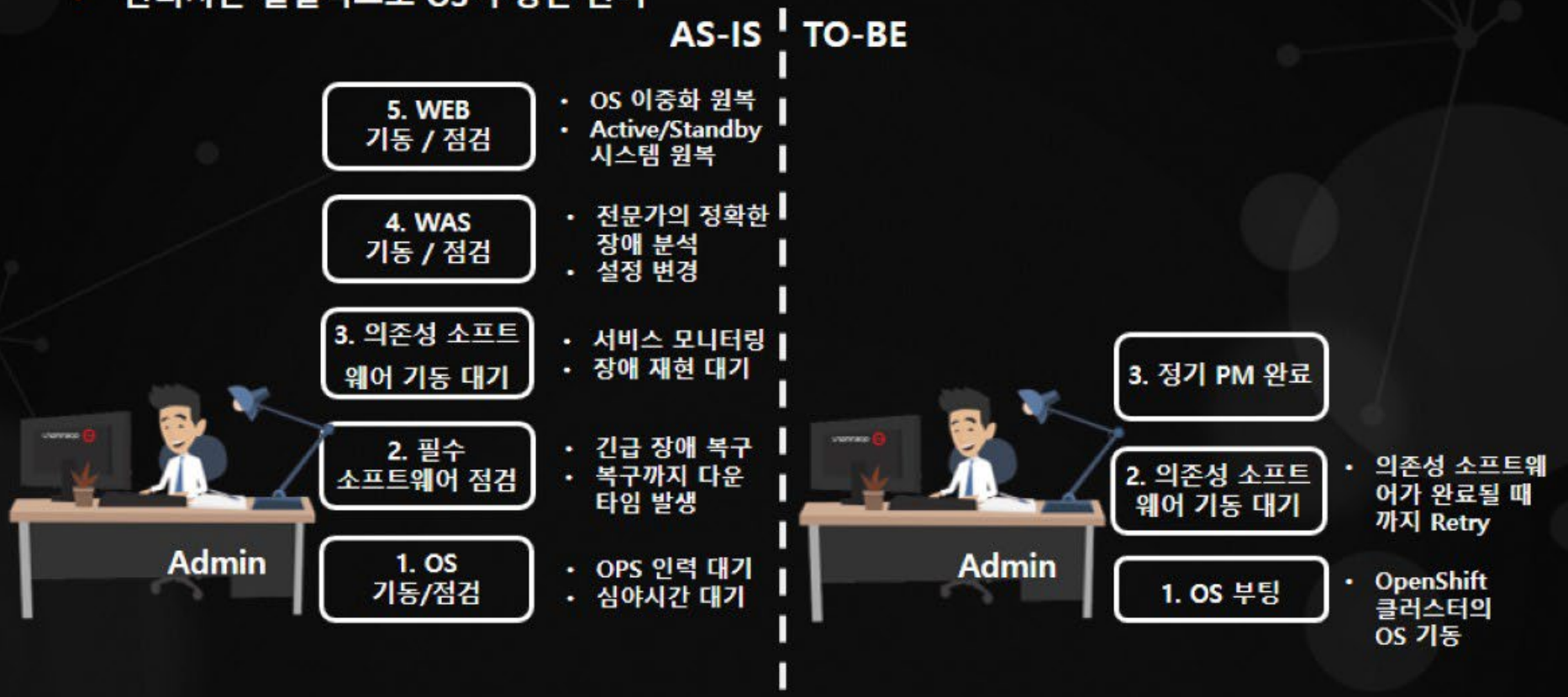

openmaru
APM
opennaru
POC
한국어
홈 / 였다 / 대시 보드 / 계기반
멀어지서
ha-app-8
ha-app-8-fmlzl
뷰어 요청
사격
2.0
0
속도 요청
상표
공
0.0tps
0.00s
2u
50
40
30
20
10
0
Pending Requests
0
ha-app-8-fmlzl
Satisfying
Tolerating
Frustrating
거래 히트 맵 (T-Map)
10
8
6
4
2
0
Response Time (s)
2
20:20
20:22
20:24
20:26
20:28
192.168.23.66:22 - root@bastion:~ - Xshell 6
ssh://root@192.168.23.66:22
파일(F) 편집(E) 보기(V) 도구(T) 탭(B) 창(W) 도움말(H)
1 192.168.23.66:22
[root@bastion ~]# oc logs -f ha-app-8-fmlzl
1 192.168.23.66:22
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
ssh://root@192.168.23.66:22
SSH2 xterm 63x8 1,44 2 세션
CAP NUM
0
20:20
20:22
20:24
20:26
20:28

정기 PM 운영관리 비교

openmaru Red Hat

인프라 팀(정기 PM)

- 기존 환경은 소프트웨어에 따라 기동 순서가 있음
- 의존성 소프트웨어(ex: DataBase)가 정상 기동 될 때 까지 관리자는 작업 대기
- 컨테이너 환경은 의존성 소프트웨어로 인해 기동 실패하더라도 성공 시 까지 Retry
- 관리자는 실질적으로 OS 부팅만 관여

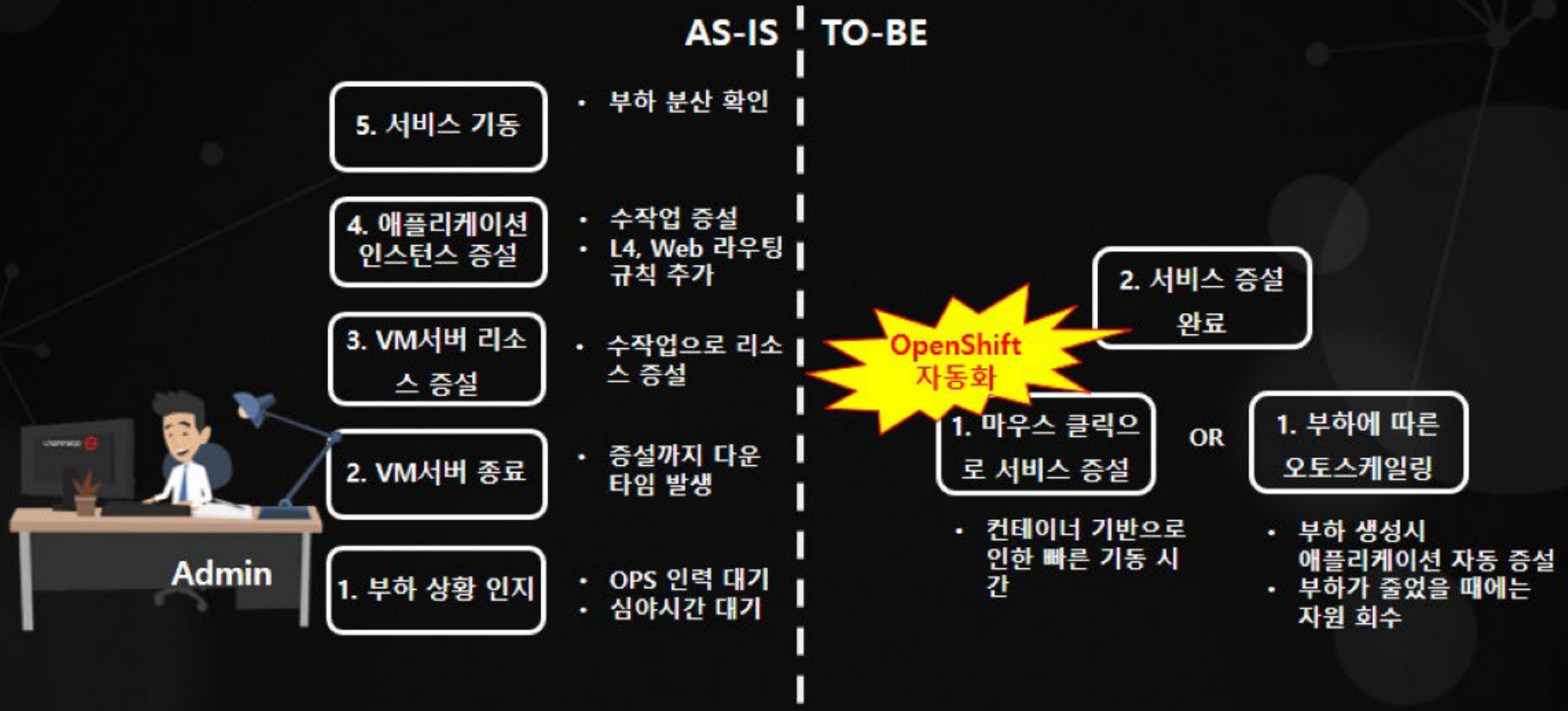


OPENMARU © 2022 | All Rights Reserved.

부하 발생 및 부하 예정시 운영 관리 비교

인프라 팀(부하 상황)

- 부하가 예고된 경우 수작업으로 Scale-Up 진행 → 애플리케이션 인스턴스도 늘려줘야 함.
- 예고되지 않은 부하의 경우 서비스의 사용자 만족도 하락
- 컨테이너 기반의 클라우드 네이티브 환경은 부하에 따라 자동으로 컨테이너(애플리케이션 인스턴스) 증설

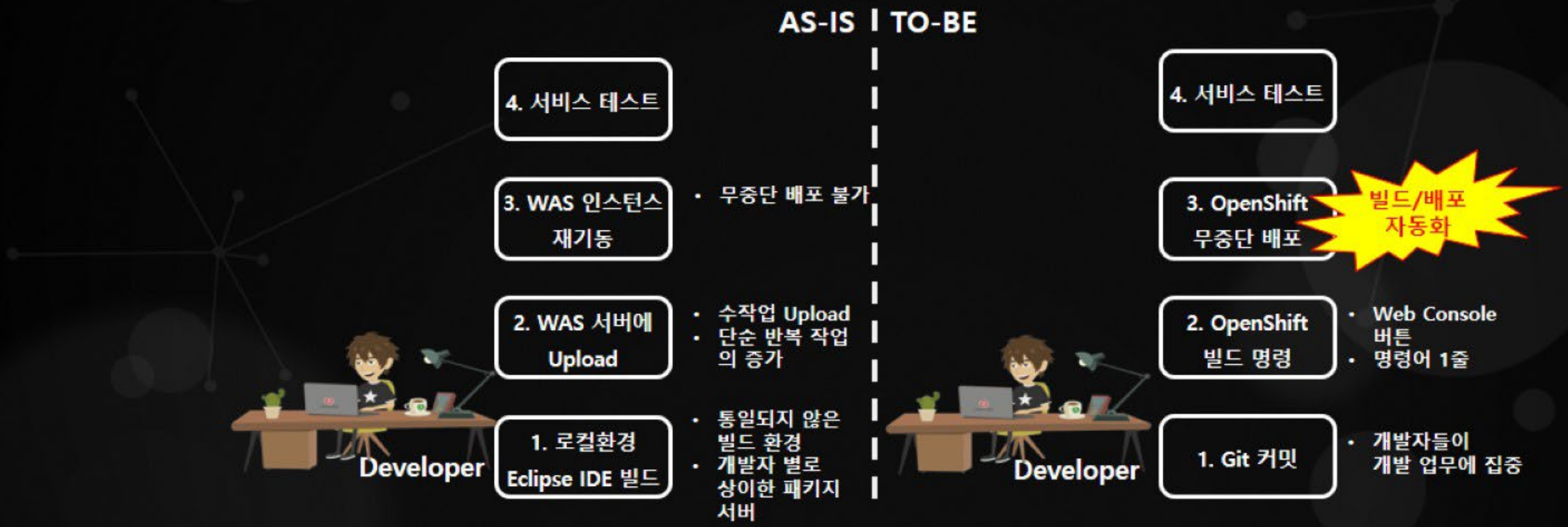


OPENMARU © 2022 | All Rights Reserved.

빌드/배포 운영관리 비교

개발 팀(빌드/배포)

- 기존 환경은 애플리케이션을 빌드, 서버에 업로드, 재기동 모두 수작업으로 진행
- 무중단 배포가 불가능하였으며 배포작업시 휴먼 에러 발생가능성이 높음
- 컨테이너 환경은 빌드 명령 혹은 Console 버튼으로 빌드, 배포까지 완전 자동화
- OpenShift에 대한 전문 지식이 없어도 빌드 배포 가능



OPENMARU © 2022 | All Rights Reserved.

애플리케이션 Rollback 운영관리 비교

openmaru Red Hat

개발 팀(Rollback)

- 애플리케이션 롤백 또한 자동화 되지 않은 수작업으로 진행
- 기존 환경의 빌드 배포와 같은 방식
- 컨테이너 환경은 애플리케이션 배포 실패 시 자동으로 정상 버전으로 롤백
- 관리자의 개입으로 명령어 한 줄로 특정 버전의 애플리케이션 버전으로 롤백 가능

AS-IS		TO-BE	
4. 서비스 테스트		3. 서비스 테스트	
3. WAS 인스턴스 재기동	• 무중단 배포 불가 • 신속한 롤백 불가	2. 롤백 완료	
2. WAS 서버에 Upload	• 수작업 Upload • 단순 반복 작업의 증가	1. 기존의 정상 애플리케이션 롤백	OR 1. 관리자의 롤백 명령
1. 롤백 애플리케이션 빌드	• 롤백을 위한 애플리케이션 빌드 필요	• 신규 버전 애플리케이션 배포 실패시 자동 롤백	• 특정 버전으로 롤백

OpenShift 자동화

OPENMARU © 2022 | All Rights Reserved.

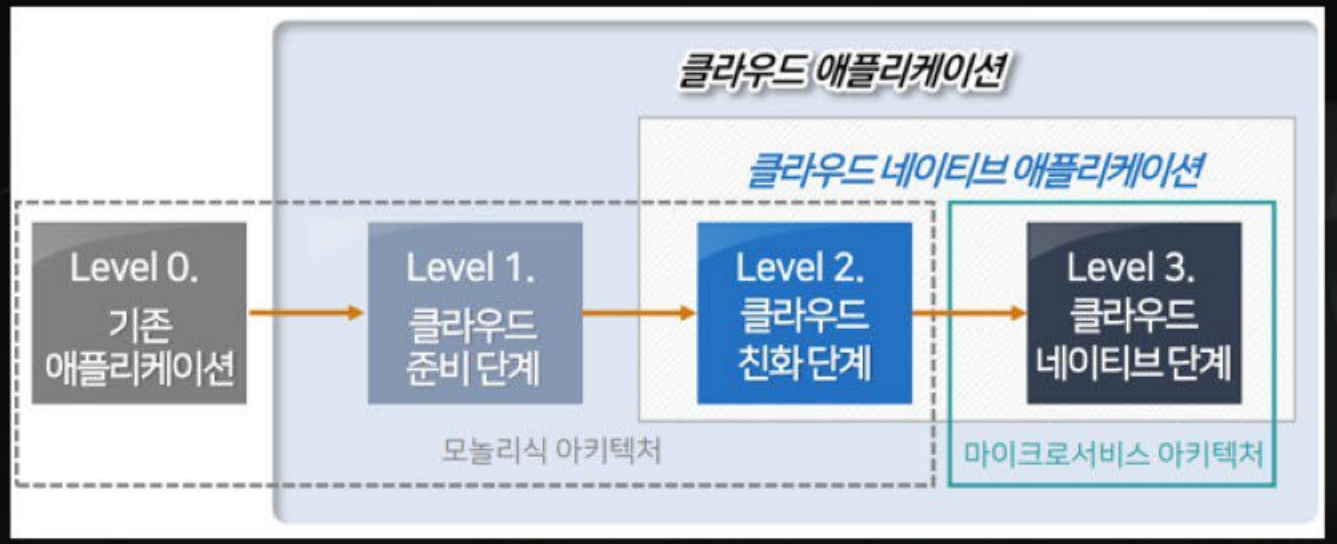

Platform As A Service
클라우드 네이티브 환경에서 해야할 것.
openmaru
APM

클라우드 네이티브를 가로막는 두려움

OPENMARU © 2022 | All Rights Reserved.

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



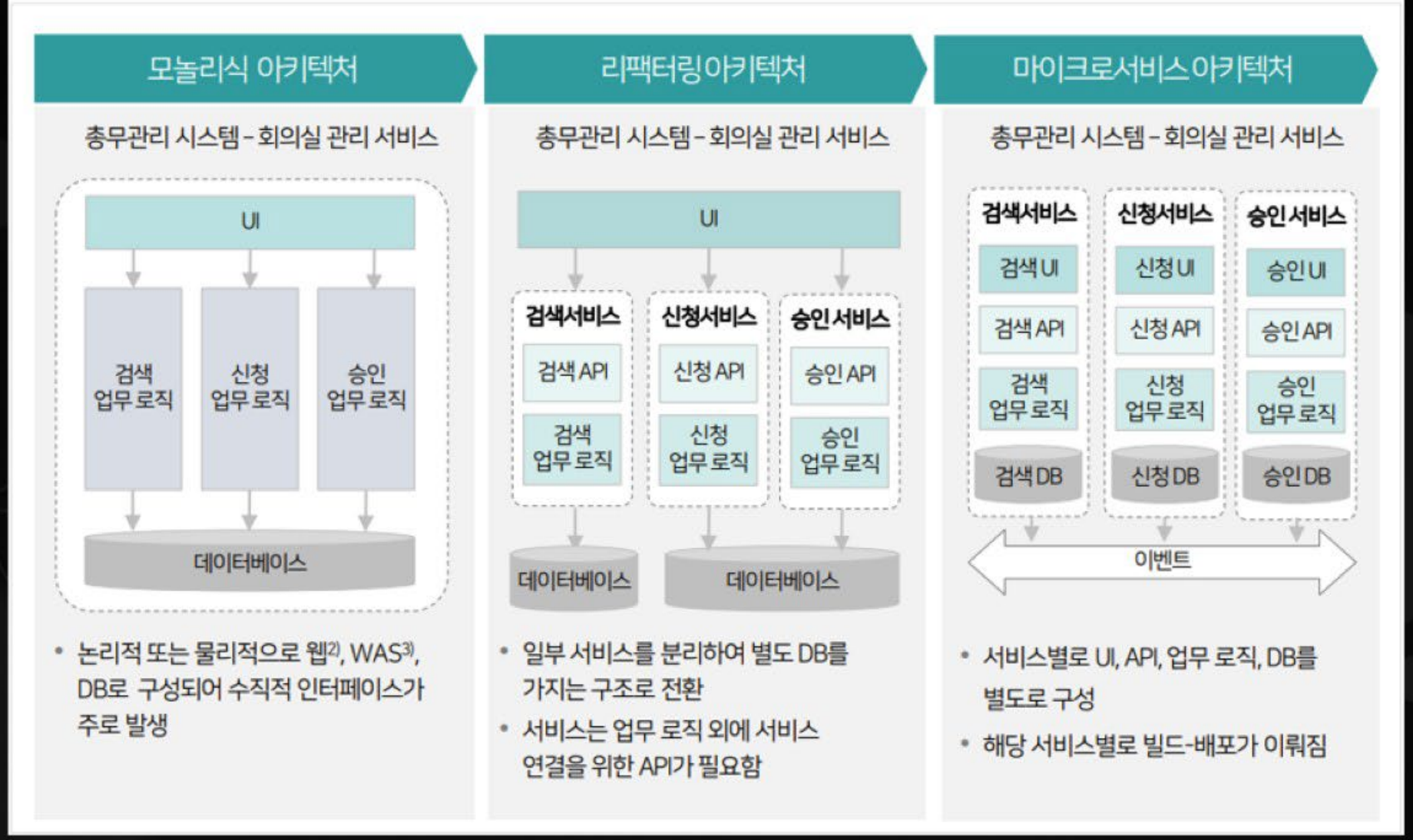
한국지능정보사회진흥원 클라우드 네이티브 발주자 안내서

OPENMARU © 2022 | All Rights Reserved.

마이크로서비스 아키텍처로의 전환 예시

openmaru Red Hat

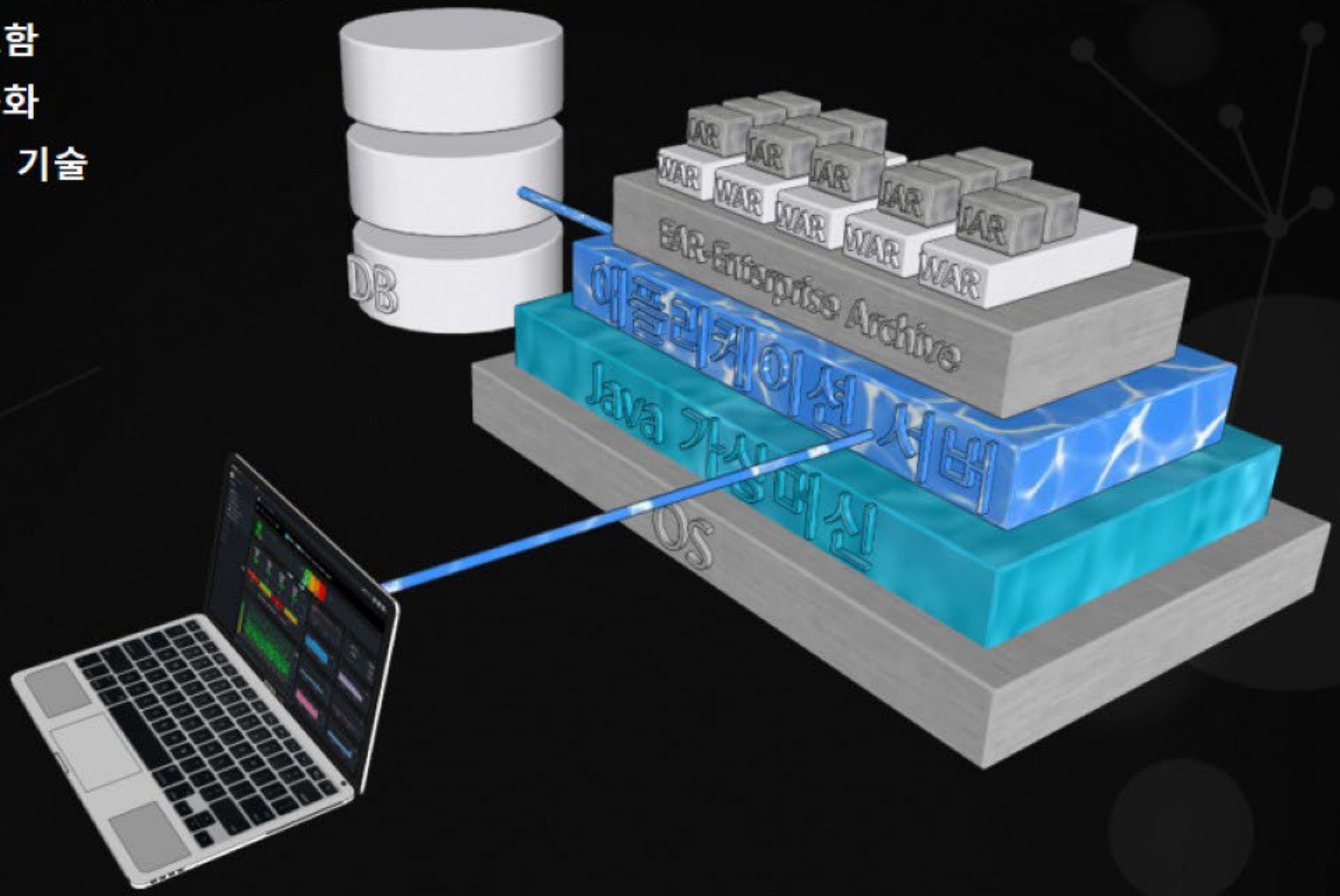
한국지능정보사회진흥원 클라우드 네이티브 발주자 안내서



OPENMARU © 2022 | All Rights Reserved.

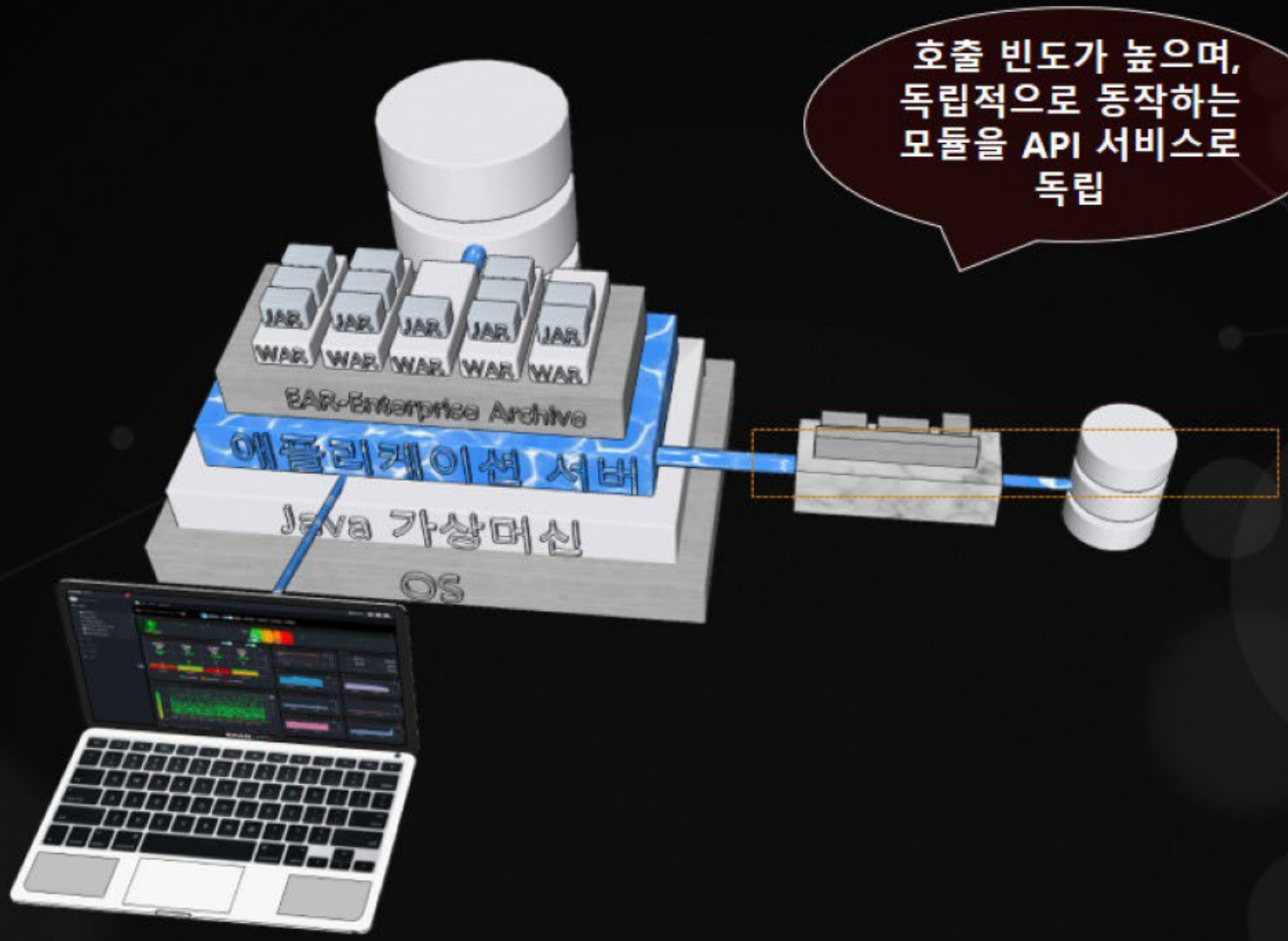
Monolith Architecture

- 모노리스 아키텍처의 특징
  - 견고함
  - 표준화
  - 단일 기술



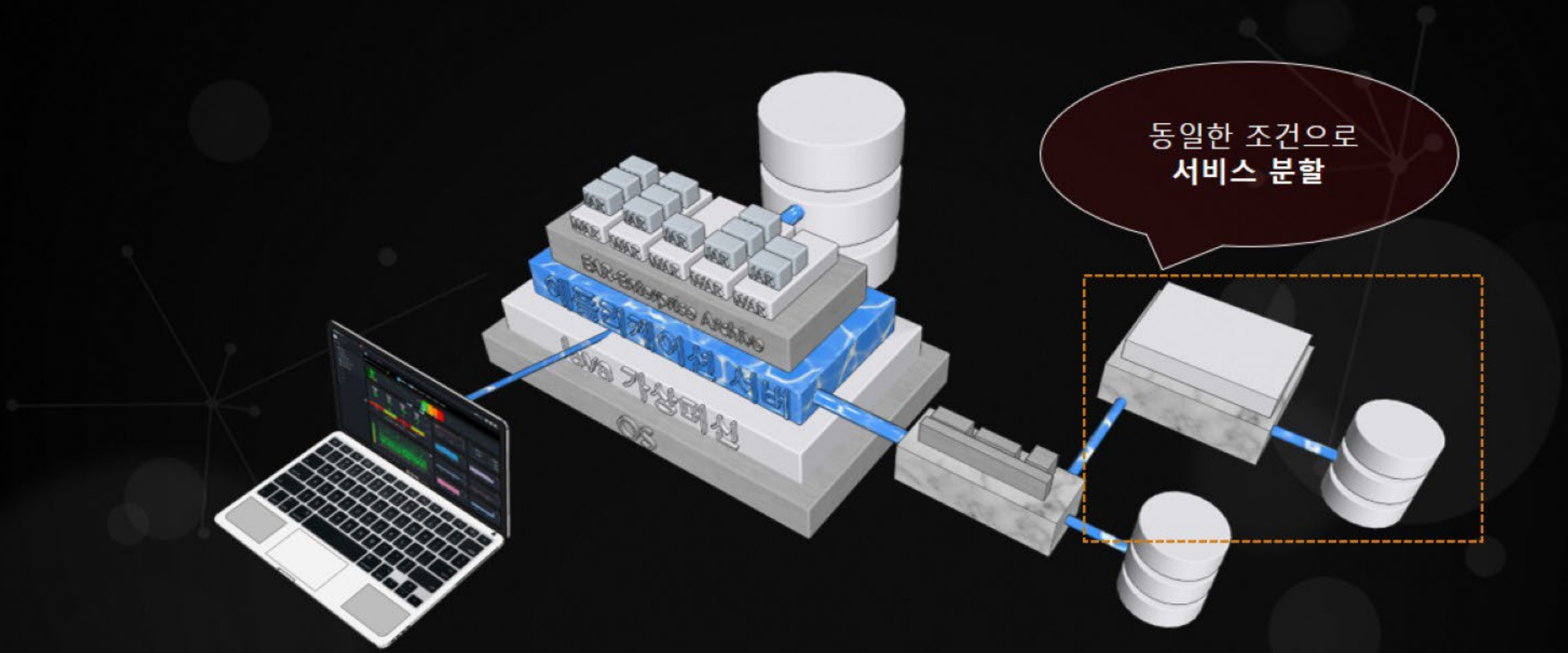
OPENMARU © 2022 | All Rights Reserved.

Monolith → Microservices – Phase 1



OPENMARU © 2022 | All Rights Reserved.

모노리스에서 마이크로서비스로 전환 – Phase 2



OPENMARU © 2022 | All Rights Reserved.

모노리스에서 마이크로서비스로 전환 – Phase 3

OPENMARU © 2022 | All Rights Reserved.

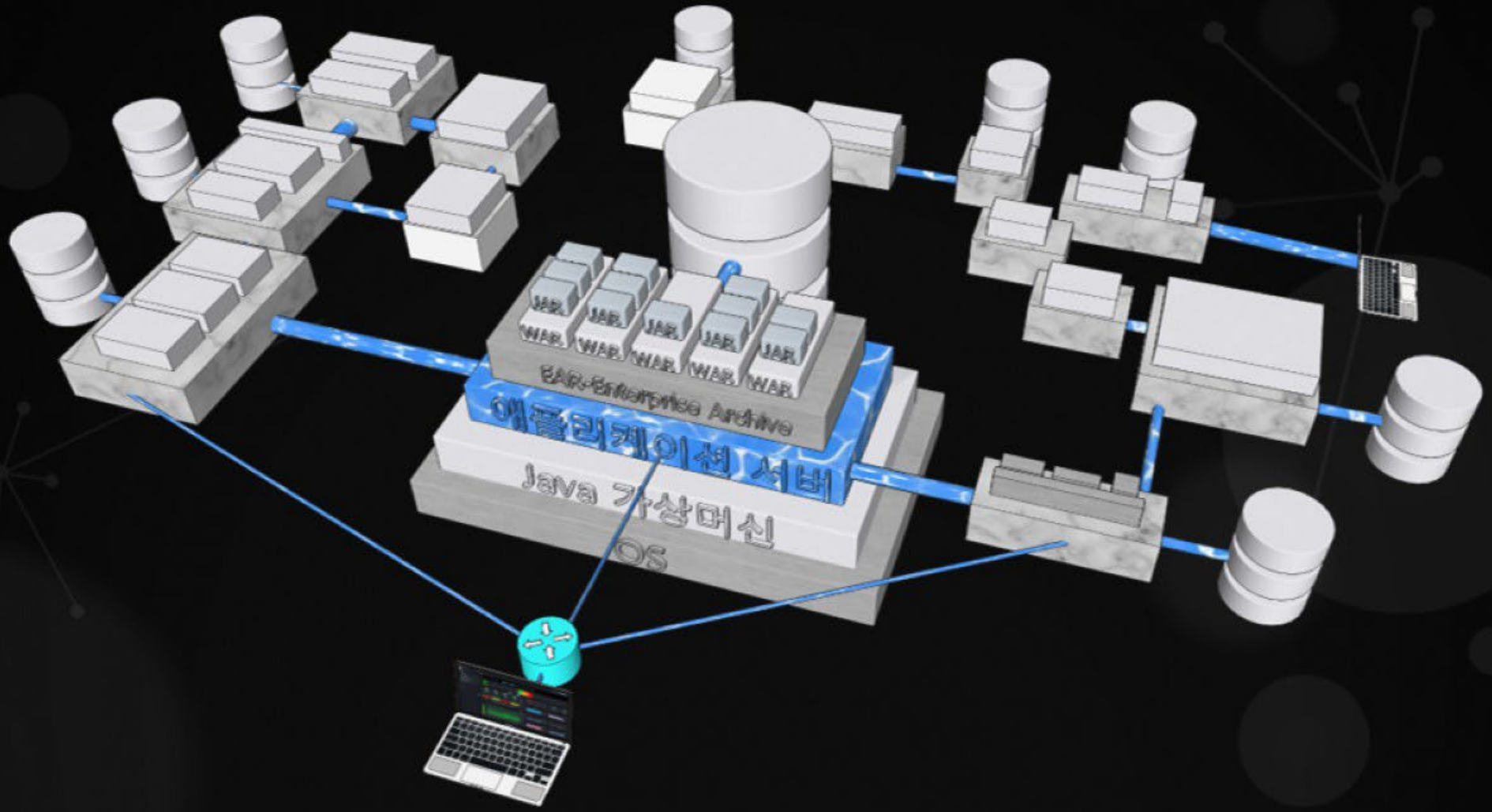
모노리스에서 마이크로서비스로 전환 – Phase 4

- Confidential -

OPENMARU © 2022 | All Rights Reserved.

모노리스에서 마이크로서비스로 전환 – Phase 5

- 개선을 반복하여 최적의 마이크로 서비스를 제공



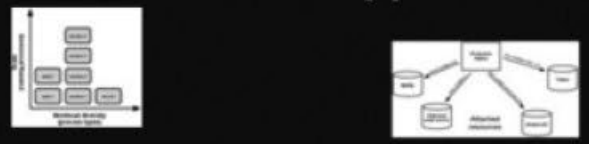
OPENMARU © 2022 | All Rights Reserved.

Cloud Native Application(애플리케이션 현대화) 도전!

openmaru Red Hat

- 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

OPENMARU © 2022 | All Rights Reserved.

The 12 Factor App (1/2)

- **I. 코드베이스**
  - 버전 관리되는 하나의 코드베이스로 여러 곳에 배포
- **II. 종속성**
  - 의존 관계를 명시적으로 선언하고 분리
  - 환경에 의존하지 않도록 함
- **III 설정**
  - 설정을 환경 변수에 저장하기
- **IV. 백엔드 서비스**
  - 백엔드 서비스를 연결된 리소스로 취급
- **V. 빌드 릴리스, 실행 (Build, release, run)**
  - 빌드, 릴리스, 실행 3 단계를 엄격하게 분리
- **VI 프로세스**
  - 응용 프로그램을 하나 또는 여러 개의 독립적인 프로세스로 실행
- **VII. 포트 바인딩**
  - 포트 바인딩을 통해 서비스를 공개

OPENMARU © 2022 | All Rights Reserved.

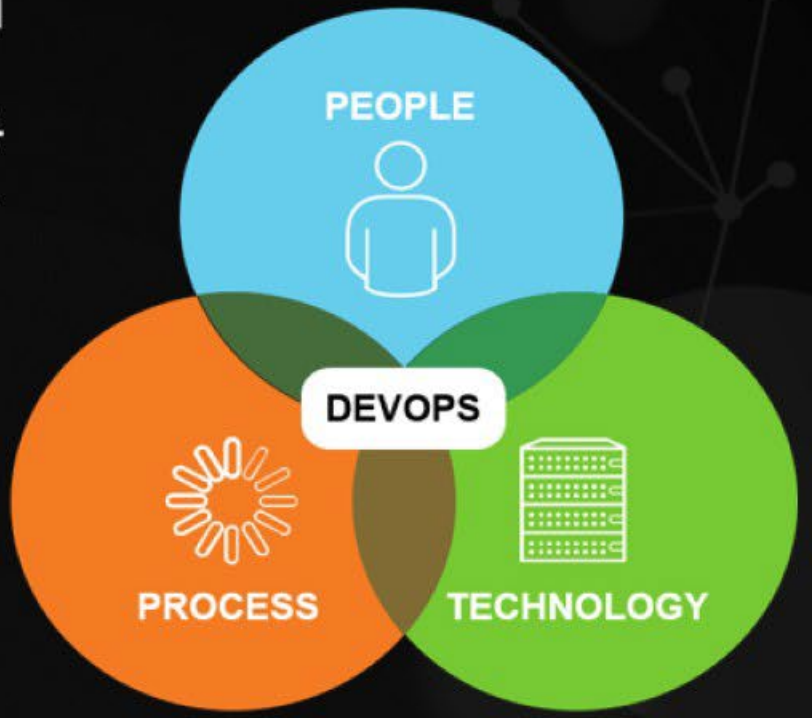
The 12 Factor App (2/2)

- **VIII. 동시성**
  - 프로세스 모델에 따라 확장
- **IX. 폐기 용이성**
  - 빠른 시작이 가능하며, Graceful Shutdown시 서비스에 영향을 미치지 않도록 함
- **X. 개발 / 운영 일치**
  - 개발 준비 프로덕션 환경과 최대한 일치된 상태를 유지
  - CI / CD (Continuous Integration/Continuous Delivery)환경이 갖춰져 있어야 함
- **XI. 로그**
  - 로그를 이벤트 스트림으로 취급함
  - 중앙 집권적인 서비스를 통해 로그 이벤트를 수집하고, 인덱싱하여 분석하는 환경이 가능해야 함
- **XII. 관리 프로세스**
  - 관리 작업을 일회성 프로세스로 실행

OPENMARU © 2022 | All Rights Reserved.

DevOps 란 무엇인가?

- 개발 (Development) 과 운영 (Operations) 의 합성어
- 개발담당자 및 운영 담당자의 협력 개발방법론
- DevOps사업 부문을 추가 한 BizDevOps 라는 단어로도 확산



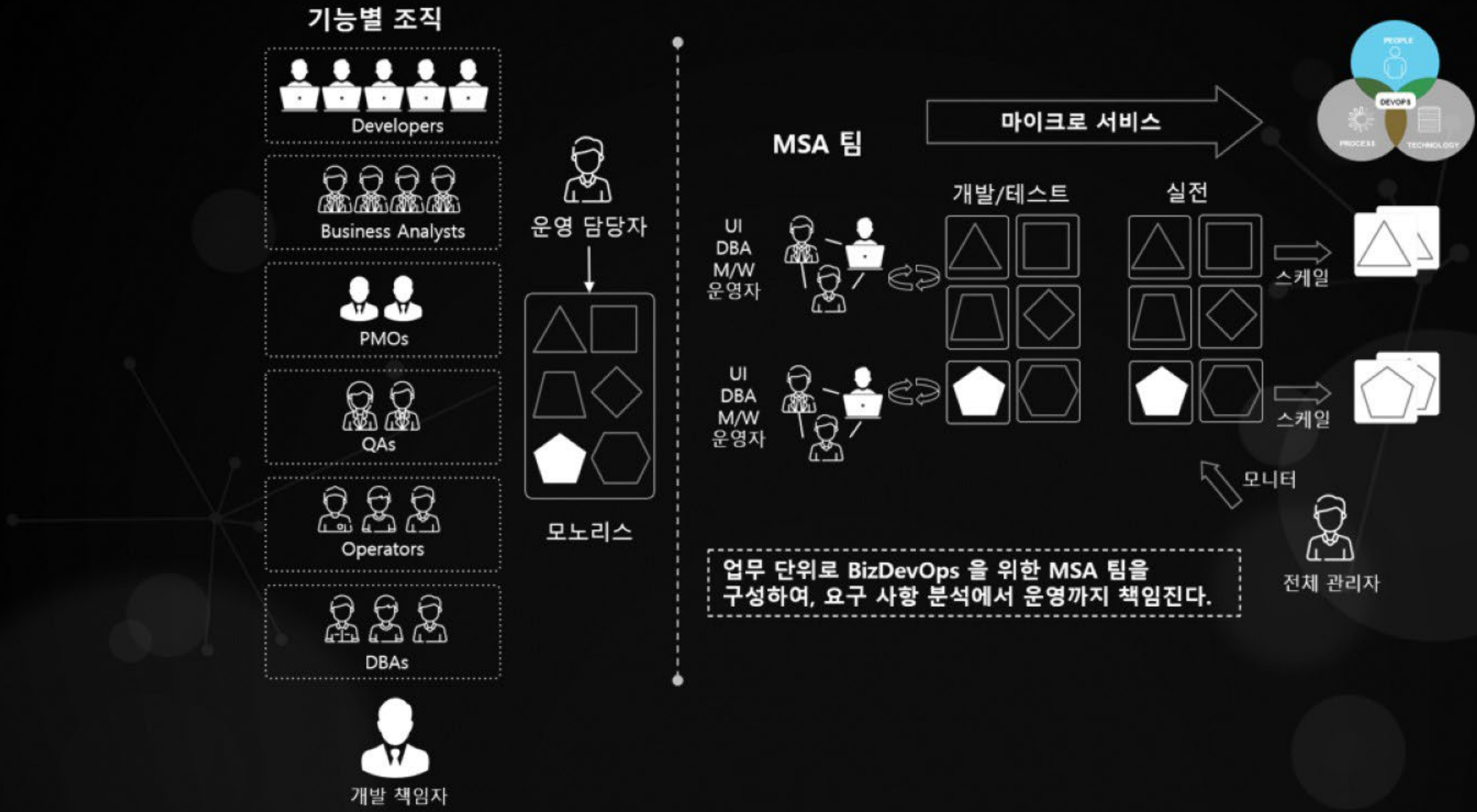
DevOps 목표

아이디어에서 구현까지의 기간을 압도적 단축

- 사용자 요구의 조기 실현
- 개발 기간 단축으로 비용 절감
- 짧은 주기의 혁신에 의한 품질 향상
  - 린스타트업 핵심
  - 디지털 트랜스 포메이션 필수 도구

OPENMARU © 2022 | All Rights Reserved.

시스템 개발과 운영 방법론의 혁신 DevOps



OPENMARU © 2022 | All Rights Reserved.

레거시 빌드 배포 환경의 해결 해야하는 과제들

설계와 구축보다, 테스트와 배포에 시간이 더 소요됨

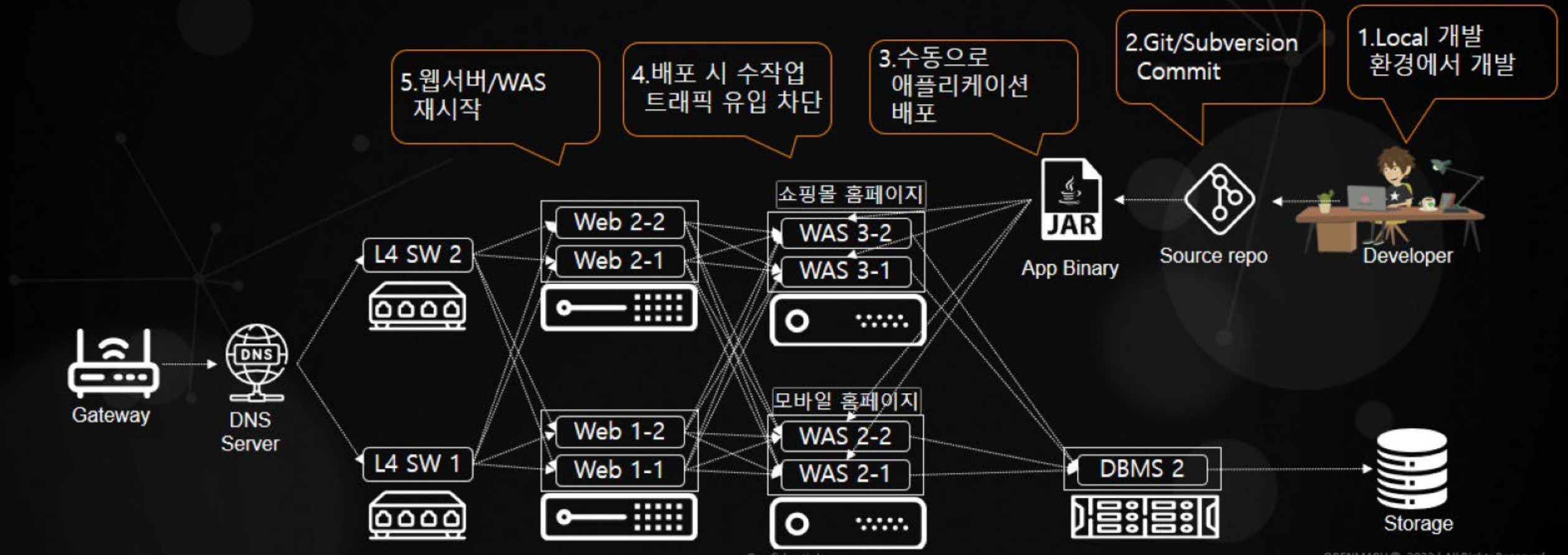
수작업으로 배포하면서 사람의 실수로 인한 문제가 발생

IT 개발과 운영을 위한 자동화 도구들이 없는 경우

OPENMARU © 2022 | All Rights Reserved.

기존 레거시에서 빌드/배포

- 수작업을 통한 복잡하고 반복적인 배포 플로우
- 시스템에 경험이 많은 배포 전문가가 집중에서 배포
- 애플리케이션 원복시에도 배포만큼의 수작업이 필요함



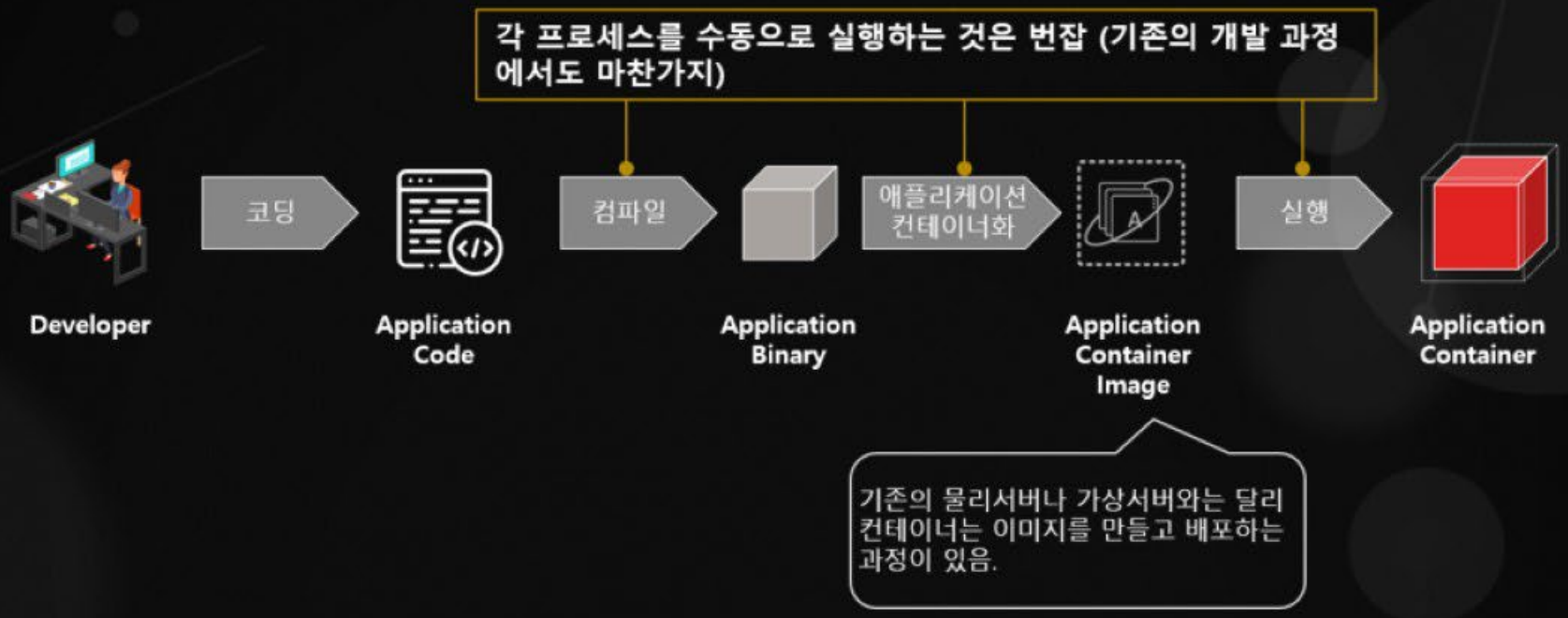
- Confidential -
OPENMARU © 2022 | All Rights Reserved.

컨테이너 기반의 애플리케이션 빌드 배포 프로세스 예

openmaru Red Hat

①

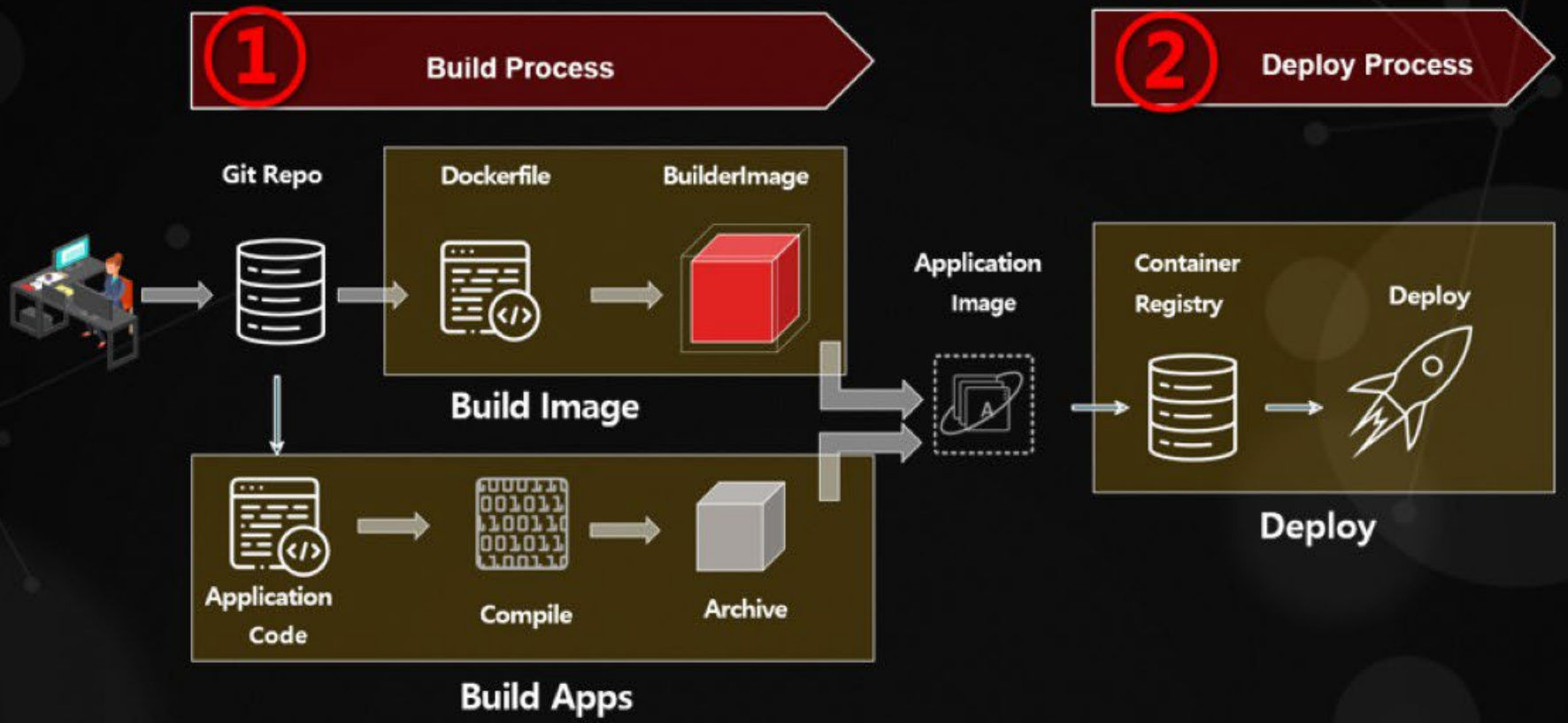
- 컨테이너 기반에서 애플리케이션을 개발하는 경우, 프로그래밍 테스트 단계의 일반적인 절차
- 컨테이너 이미지화 하여 개발 생산성 향상
  - 손쉬운 애플리케이션 환경 구성
  - 개발/스테이징/운영 환경의 차이 제거



OPENMARU © 2022 | All Rights Reserved.

Build & Deploy Application

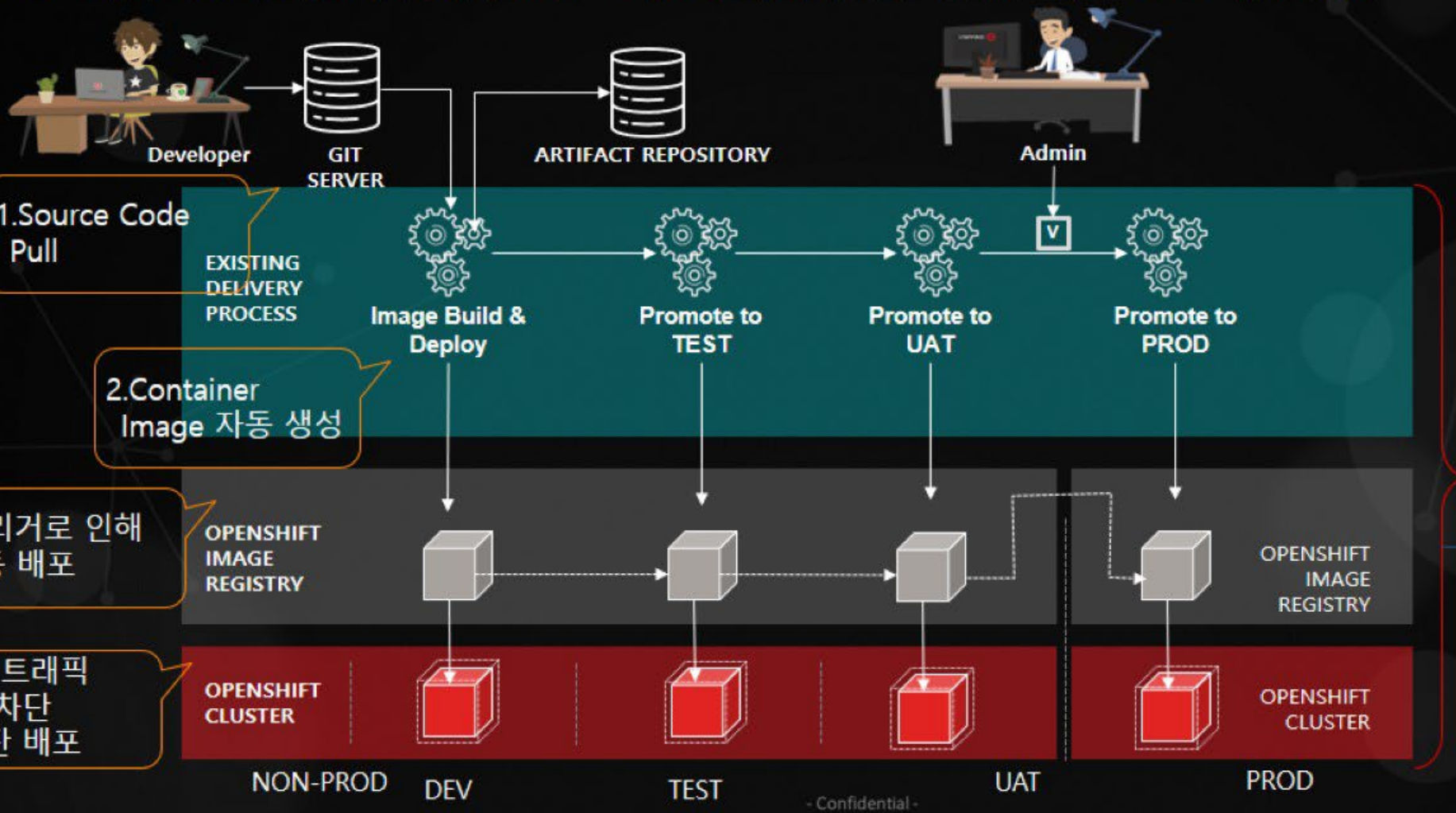
- Build Process 소스 코드 빌드 및 기반 이미지 빌드로 구성
- 애플리케이션 실행은 " Build Process "와" Deploy Process "로 구성



OPENMARU © 2022 | All Rights Reserved.

컨테이너 기반 자동 빌드/배포

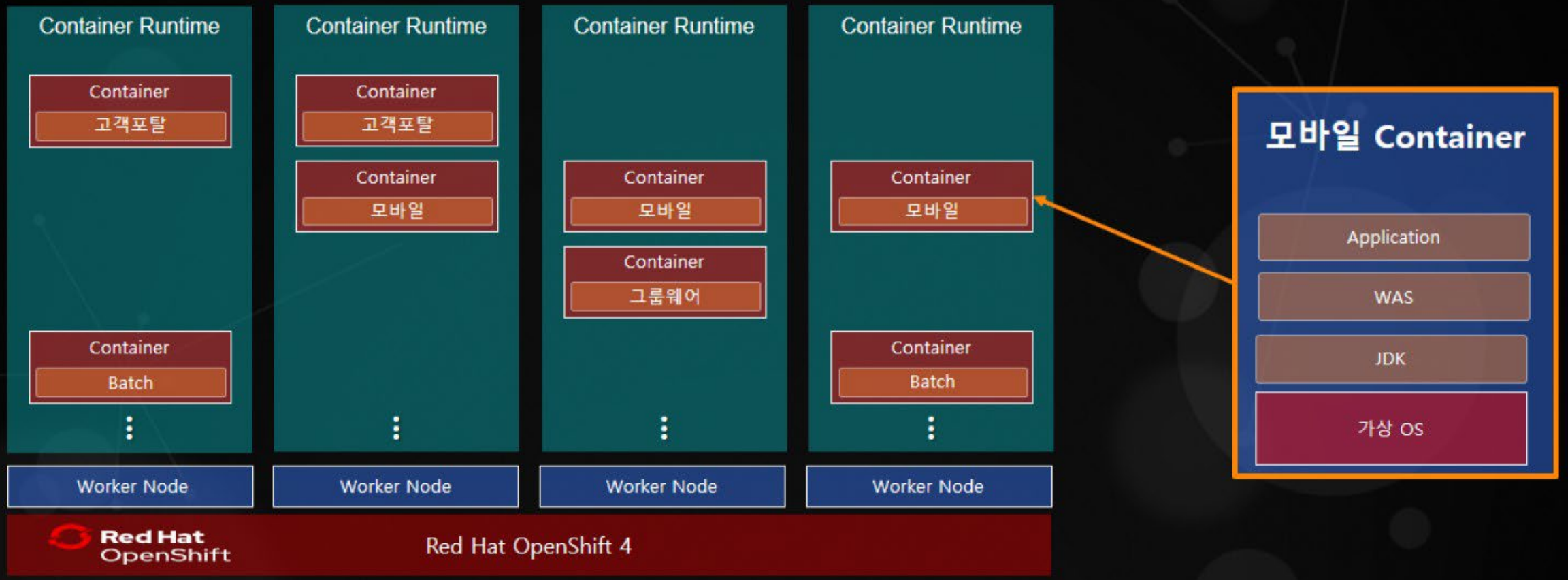
- 신속한 개발과 효율적인 운영 환경 구축
- 코드 개발 이외에 개발자/관리자가 빌드 배포에 기여하는 수작업은 "빌드 명령" 밖에 없음



- Confidential -
OPENMARU © 2022 | All Rights Reserved.

컨테이너 환경의 소프트웨어 컴포넌트 요약 구성

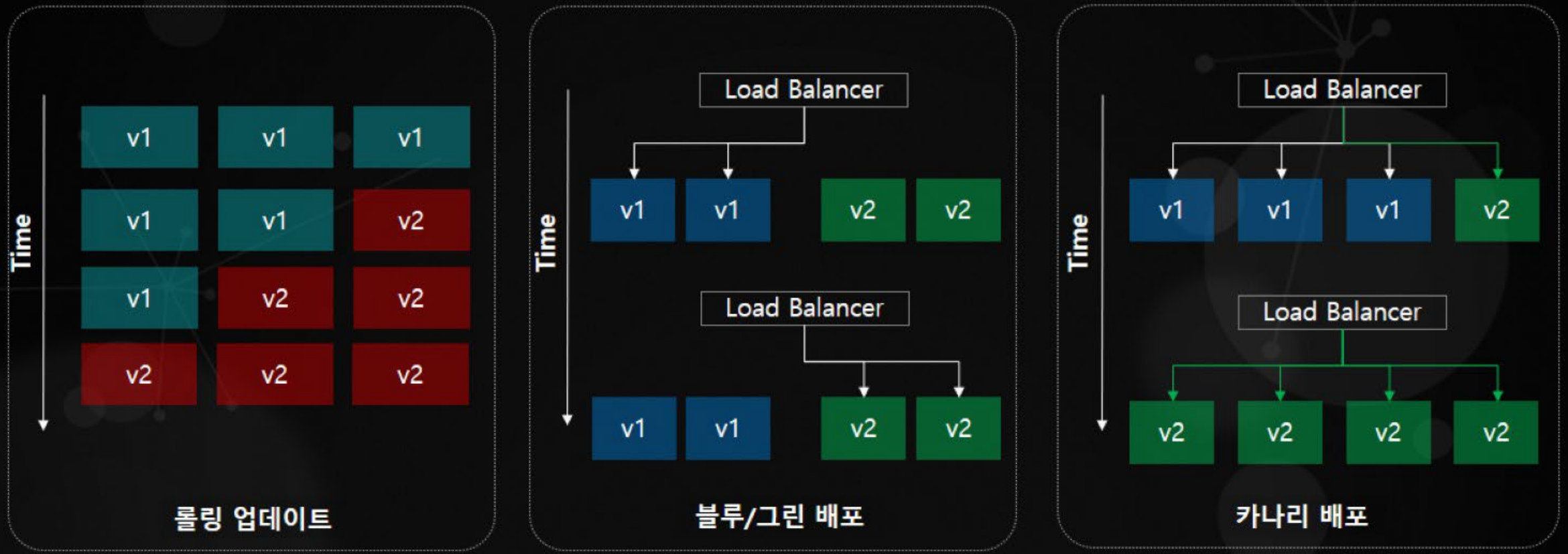
- **Container는 가상OS, JDK, WAS, Application이 포함된 Image를 기반으로 기동됨.**



OPENMARU © 2022 | All Rights Reserved.

무중단 뿐만 아닌 고도화된 배포 방식

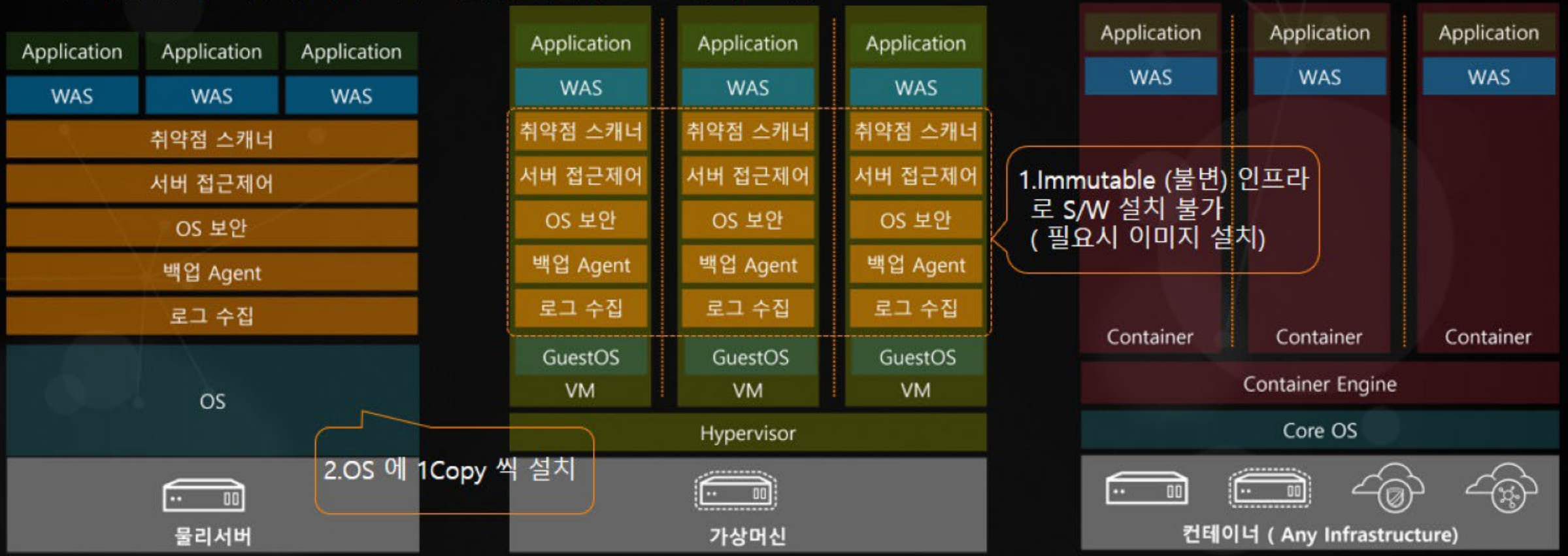
- 롤링 업데이트 - 점차 새로운 버전으로 변경
- 블루/그린 배포 – 블루 버전을 유지한채로 그린 버전을 테스트 (신속한 롤백)
- 카나리 배포 – 소규모로 검증 후 전체 반영 (신속한 롤백)



OPENMARU © 2022 | All Rights Reserved.

레거시 환경에서 필요한 서버 보안

- OS 보안, 서버 접근제어 등 보안 5종 S/W를 OS 설치
  - OS 마다 설치하기 때문에 물리서버는 1Copy이고, 가상화는 VM 개수 만큼 설치
- 서비스에 따라 부가적인 보안 소프트웨어 필요(개인정보 검출, 비정형 암호화 등)
- 법적인 근거로 인해 서버보안 – 전자금융법, ISMS 등의 요건



OPENMARU © 2022 | All Rights Reserved.

클라우드 네이티브 환경에서 필요한 보안

왜 AS-IS 환경의 보안들이 필요하지 않을까?

실질적으로 컨테이너환경에서 필요한 보안들

- 컨테이너 실행 권한에 대한 보안 : SCC
  - Container breakout
- 취약한 Container Image 사용
  - 악성코드, 채굴 프로그램이 포함된 이미지
- Role Base Access Controle : RBAC
  - 사용자별 필요 권한 부여
- 컨테이너 플랫폼의 Audit 로깅
  - EFK
- 신뢰할 수 있는 컨테이너 런타임 보안
  - Capabilities, SELinux, Seccomp & Userremap
- 컨테이너간의 네트워크 격리
  - Network Policy

OPENMARU © 2022 | All Rights Reserved.

클라우드 네이티브 환경 보안 솔루션 설치

OPENMARU © 2022 | All Rights Reserved.

아직도 WAS BMT 나 POC를 하시나요?

- WAS 의 가용성/확장성/성능/신뢰성 등의 기능은 플랫폼으로 이전
- 더 가볍고 더 빠르고, 자동화에 친화적인 WAS 로 전환

WAS 제품 BMT

ORACLE WEBLOGIC

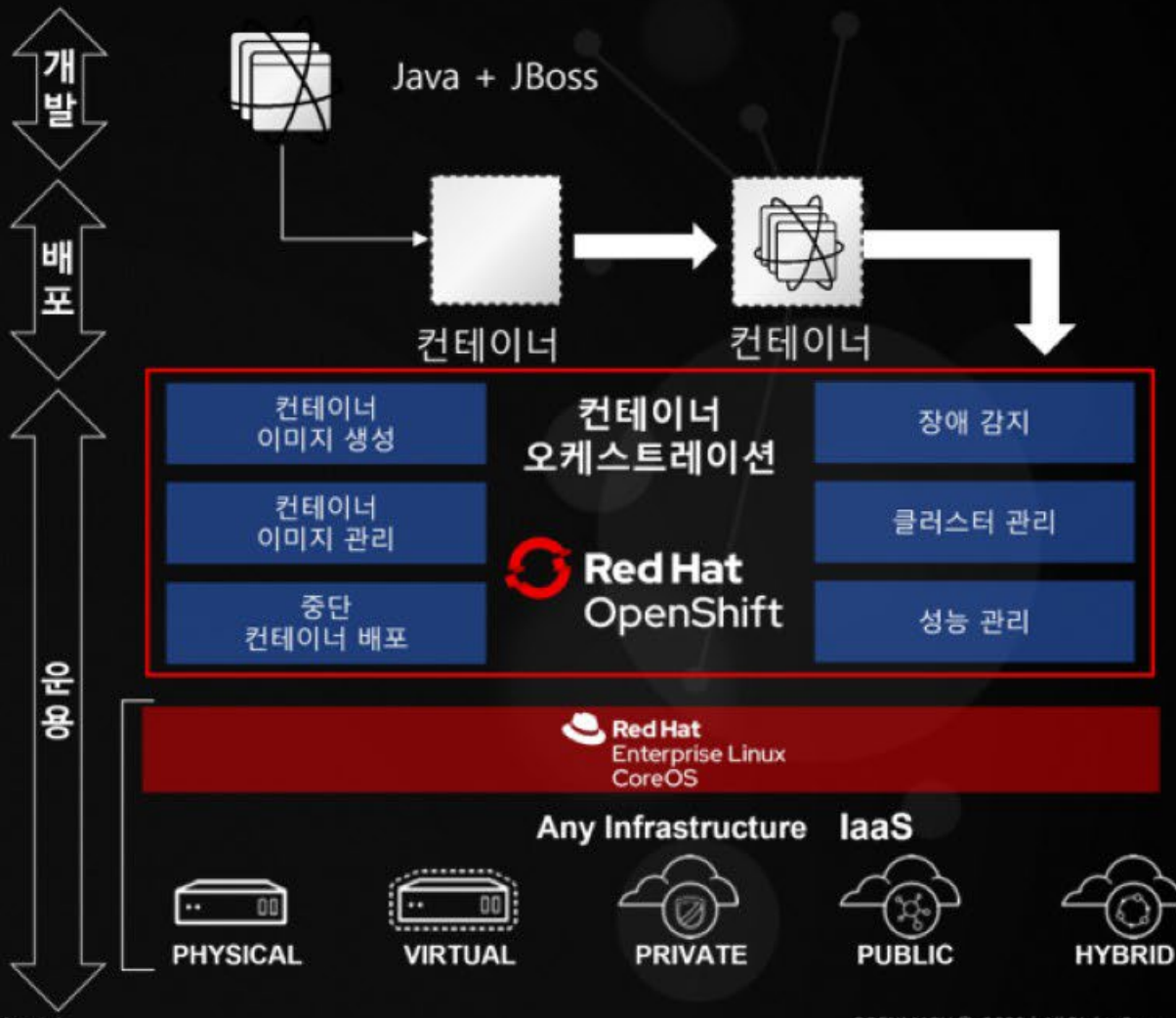
JBoss by Red Hat

WebSphere

Tmax JEUS

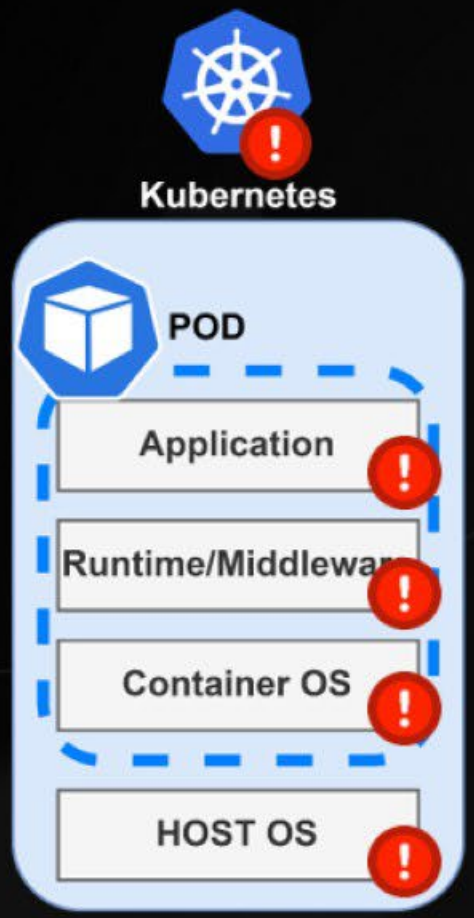
RASP 에 의한 WAS평가

- 신뢰성 (Reliability):
  - 과부하 테스트
- 가용성 (Availability):
  - 일부 장애 발생시 전체 시스템 영향 최소화
- 확장성 (Scalability):
  - 자원 추가에 따른 선형적인 성능 개선
- 성능 (Performance):
  - TPS, 응답시간 등 평가
- 보안 기능 (Security)
  - (RBAC 등)
- WAS 도구 평가 (Manageability)
  - Admin Console



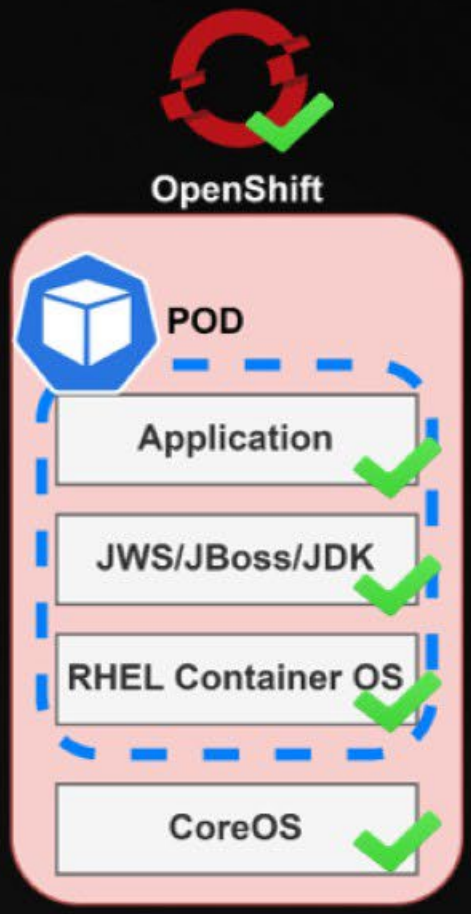
OPENMARU © 2022 | All Rights Reserved.

클라우드 네이티브 기반의 PaaS는 OS/WAS/Runtime를 제공



Kubernetes + DIY Stack

- 신뢰할 수 없는 컨테이너 이미지
  - 컨테이너 보안 업데이트 X
- QA팀을 통한 안정화 테스트 X
- HOST OS 유지보수
  - 업그레이드
  - 패치
  - 트러블슈팅
  - 구매비용 발생
- Runtime/Middleware 유지보수
  - 업그레이드
  - 패치
  - 트러블슈팅
  - 구매비용 발생



OpenShift Container Platform

- 신뢰할 수 있는 컨테이너 이미지
  - Quay, red hat registry
- QA팀을 통한 안정화 테스트
- Red Hat Core OS
  - 업그레이드 지원
  - 버그 패치 지원
  - 트러블슈팅 지원
  - 컨테이너 특화 OS
  - 구매비용 발생 X
- OpenJDK / JBoss Web Server, EAP
  - 업그레이드 지원
  - 패치 지원
  - 트러블슈팅 지원
  - 구매비용 발생 X
    (EAP의 경우 구매비용 발생)

OPENMARU © 2022 | All Rights Reserved.

클라우드 네이티브 환경의 애플리케이션 모니터링 어려움

openmaru Red Hat

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.

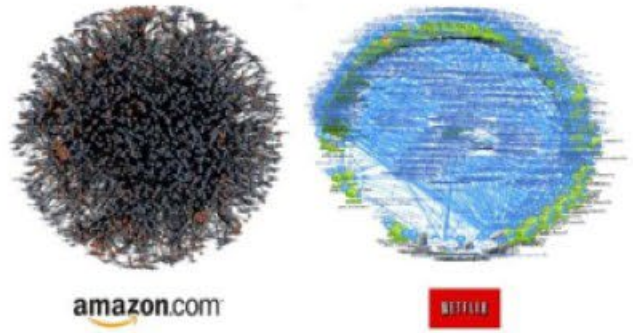
- Uber는 2014년 말 에 4,000개가 넘는 독점 마이크로 서비스와 점점 더 많은 수의 오픈 소스 시스템이 모니터링 시스템에 문제를 제기했다고 보고
- 컨테이너 인프라는 모니터링 시스템이 필수적인 환경
- **마이크로 서비스에 특화된 모니터링 시스템이 필요**

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.

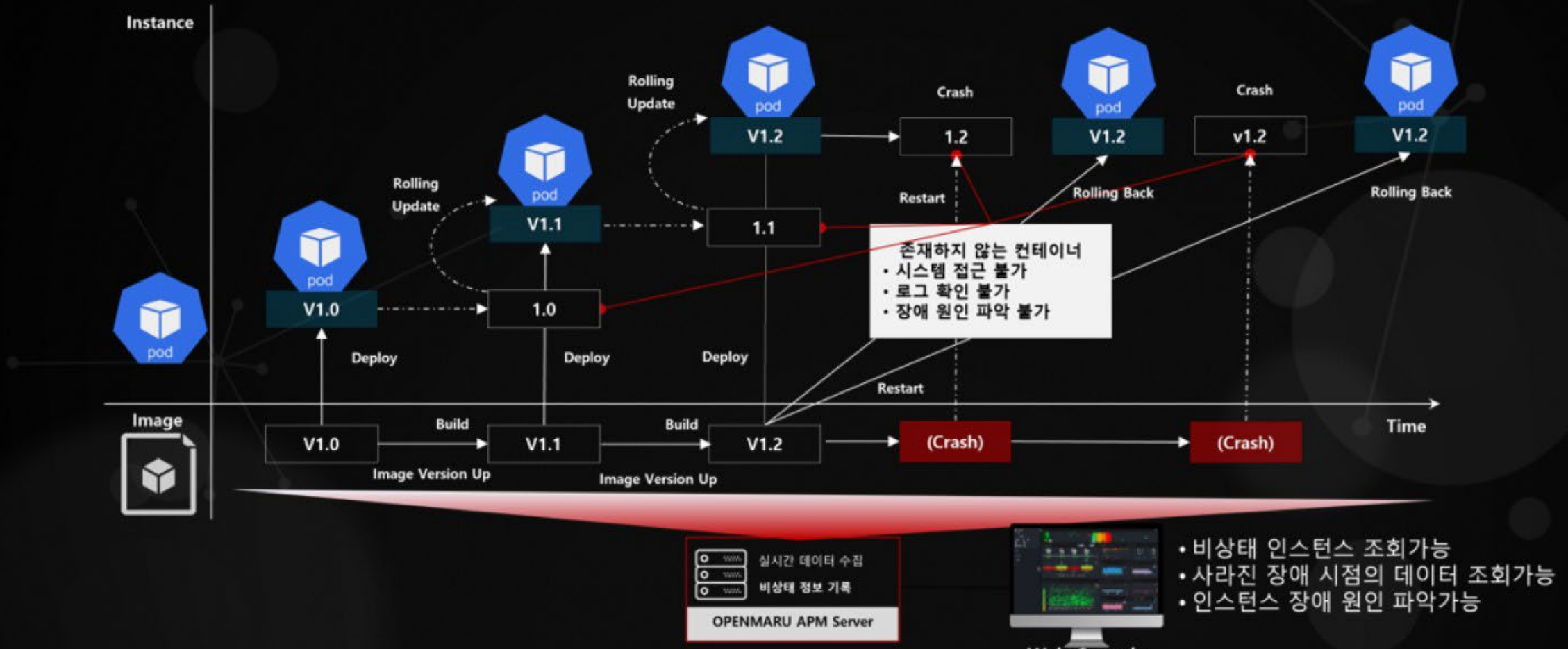


https://www.infoq.com/articles/microservice-monitoring-right-way

OPENMARU © 2022 | All Rights Reserved.

컨테이너는 무상태 (Immutable) 인프라스트럭처로 상태를 저장하지 않음

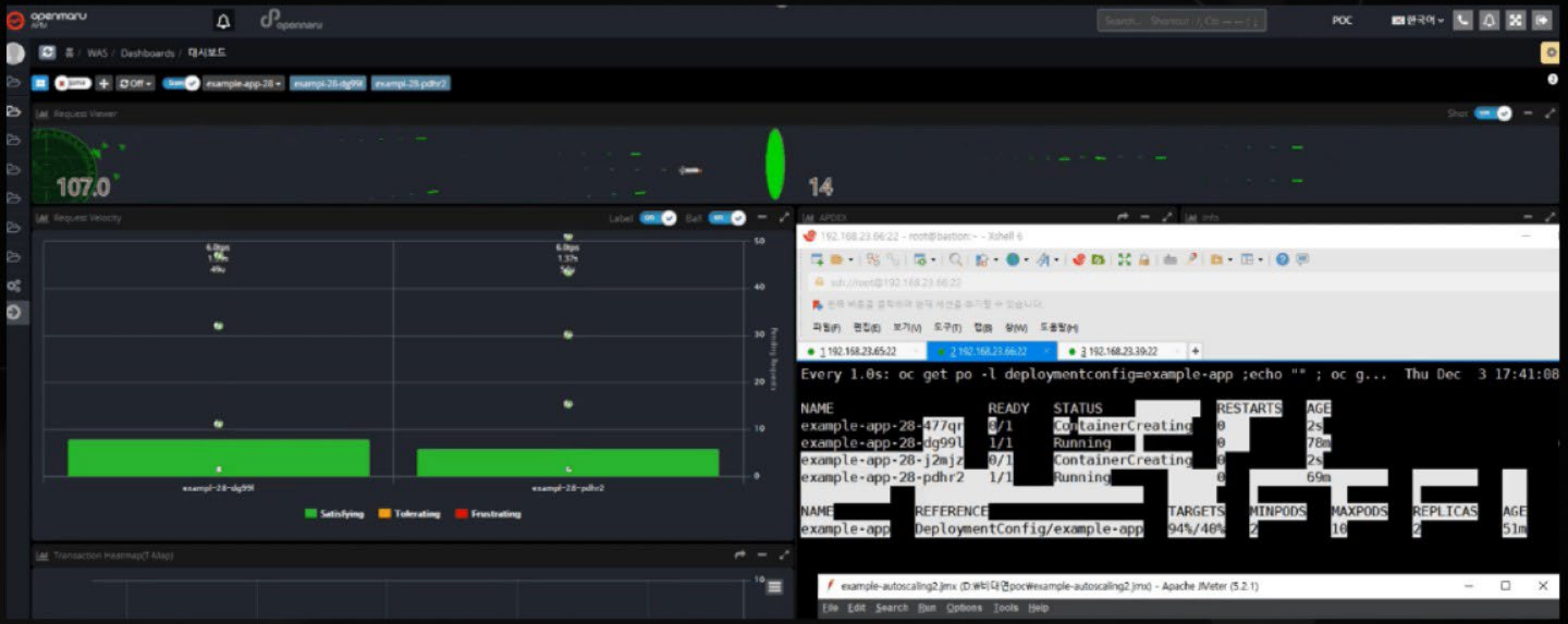
- PaaS 환경에서 컨테이너가 중지된 후 장애원인 파악을 위한 방법을 제공
- APM 에서 사라진 컨테이너에 대한 정보를 보관하여 장애원인을 파악할 수 있음



OPENMARU © 2022 | All Rights Reserved.

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 © 2022 | All Rights Reserved.


openmaru

제품 / 서비스에 관한 문의

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