

# 기업 생존 전략인 디지털 트랜스포메이션 (DX)의 성공 사례와 클라우드 네이티브

# 디지털 트랜스포메이션 (DX : Digital Transformation )은?

- Digital Transformation 에서 “Trans”를 “X”로 표기하여 DX 로 표기함



**4차 산업혁명 시대**를 맞아 국민의 일상에 **디지털 전환**이 가속화되고 있고,

이에 적합한 대국민 서비스 제공을 위해

**공공서비스의 클라우드 전환이 추진**되고 있습니다.

클라우드 전환 및 도입 효과를 높이기 위해

**단순한 기술 인프라 위주의 클라우드** 도입보다 클라우드

환경에 최적화된 새로운 형태의 **클라우드 네이티브 정보시스템 구축**이

필요합니다. 즉, 기존의 크고,

**단일한 서비스 구조를 마이크로서비스 아키텍처로 구현**하

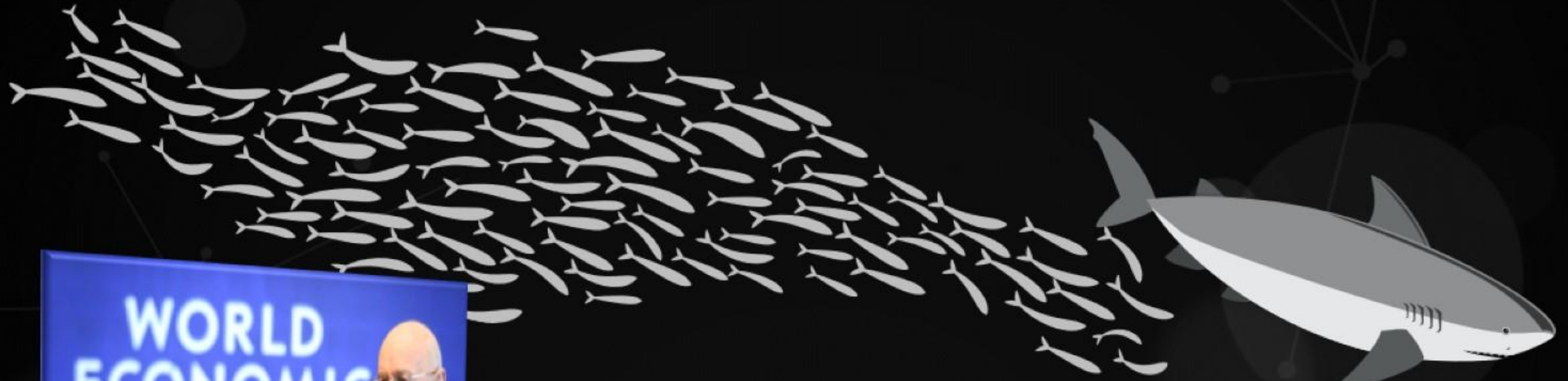
여 개발, 배포, 운영함으로써 빠르고 안정적인 할 수 있습니다.

*클라우드 네이티브 정보시스템을 구축을 위한 발주자 안내서-  
행정안전부, 한국지능정보사회진흥원*



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**

# Business Disruptor



- 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 부동산을 소유하지 않음



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



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



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

ALL CHECKED IN!



ONLINE STORE



“**메르세데스 벤츠**는 이제 자동차 기업이 아닌 소프트웨어 기업이며 자동차는 궁극의 웨어러블이다.

- 올라 칼레니우스, 메르세데스 벤츠 그룹 대표



“우리는 클라우드 네이티브로 기술을 재구축하면서 조직 운영 방식도 바꿨다.

- 넷플릭스 클라우드 부문 부사장 이즈라일레브스키



“**골드만삭스**는 **정보기술(IT)** 회사다.

- 골드만삭스 CEO 데이비드 리브킨



“인간 중심의 **디지털 전략**이 미래 **리테일 산업**의 성공의 열쇠가 될 것입니다.

- Kevin Johnson, CEO of Starbucks



## 클라우드 네이티브 성공사례

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



**NETFLIX**

넷플릭스는 매년 전세계 인터넷 트래픽의  
15% 이상을 차지하고 있음

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

**NETFLIX**  
**OSS**  
Netflix  
OpenSourceSoftware  
Center



## 기술팀이 8년간 노력해온 결과



인프라를 자체 센터에서  
Public 클라우드 이전



모놀리틱 프로그램을 작게  
관리할 수 있는 마이크로서비스  
아키텍처로 변경



애플리케이션 함수 실행  
서비스하는 서버리스 컴퓨팅 및  
백엔드 아키텍처를 제공

# 클라우드 네이티브 국내·외 선도 도입 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



## 선진 사례

## KAKAO

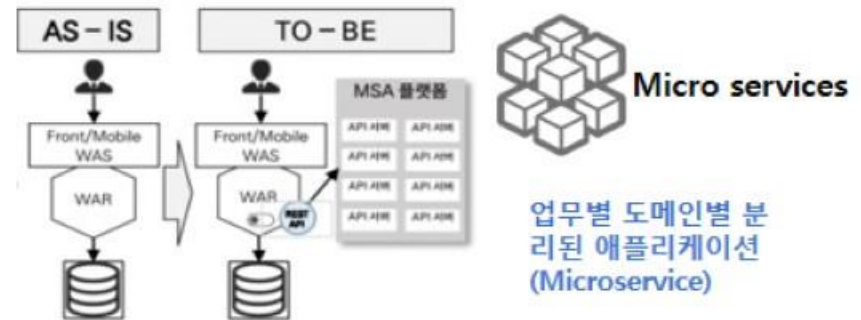
### 계열사 신규서비스 확대 및 빠른 출시 사례

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



## 11번가

### 서비스 분리를 통한 점진적 MSA 전환





# SoR 에서 SoE 로 전환

## Systems of Record (SoR)

|              | Mainframe   | Mini         | PC                   |
|--------------|-------------|--------------|----------------------|
| Time Frame   | 1960 - 1975 | 1975-1992    | 1992-2001            |
| Data Types   | Batch       | Dept Process | Documents            |
| First Movers | IBM, Unisys | DEC, Compaq  | Microsoft, Dell, IBM |

## Systems of Engagement (SoE)

|              | Internet          | Mobile/Cloud            | Connected    |
|--------------|-------------------|-------------------------|--------------|
| Time Frame   | 2001-2009         | 2010-2015               | 2016-        |
| Data Types   | Web Pages         | User Interactions       | IOT / AI     |
| First Movers | Google, Microsoft | Facebook, Amazon, Apple | Airbnb, Uber |

데이터를 기록하는 시스템

신뢰성/안전성/정확성/데이터 중심

Waterfall

Java & .NET

독점 기술 중심/ 물리&가상 환경

모노리식

웹기반

일괄/실시간 처리

역할

포커스

어프로치

프레임웍

기술

아키텍처

어플리케이션

트랜잭션

연결·관계·상호작용을 만드는 시스템

즉시성/유연성/사용자 중심

DevOps & Agile

경량/다양한 프로그램 언어

오픈 / 클라우드

마이크로 서비스

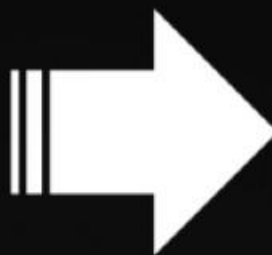
Mobile

리얼타임, 탄력적인 인프라

## 기존 IT



- 물리 서버와 PC
- 5년 이상 시스템 사용 기간
- 폭포수형 개발
- 운용의 우선 사항 "안정성"
- 시스템 부문은 시스템에 전념
- 대형 IT 벤더가 기술을 주도



## 새로운 IT



- 클라우드와 모바일
- 몇 개월 사용 후 폐기
- Agile, DevOps
- 운용의 우선 사항 "민첩성"
- 시스템 부문도 비즈니스에 초점
- 사용자 주도 기술, 오픈 소스 주도

## Development Process



WATERFALL



AGILE



DEVOPS



## Application Architecture



MONOLITHIC



N-TIER



MICROSERVICES



## Deployment & Packaging



PHYSICAL SERVERS



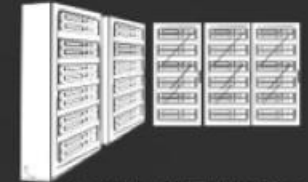
VIRTUAL SERVERS



CONTAINERS



## Application Infrastructure



DATA CENTER



HOSTED



CLOUD



Cloud Native



디지털 플랫폼 정부와 클라우드 네이티브

# 디지털플랫폼정부 인프라 TF 가이드 라인

| 순번 | TF명  | 이행과제   |
|----|--|--|
| 12 | <p><b>DPG 기반 TF</b></p> <ul style="list-style-type: none"> <li>▪ [담당] 오종훈 위원</li> <li>▪ [담당] 이한주 위원, 송상호·이동범·김준범·정재웅 전문위원</li> </ul> | <ul style="list-style-type: none"> <li>- 본격적인 클라우드 네이티브 전환을 위해 준비해야할 구체화된 과제를 검토 및 도출하고, 관계 부처 및 민간기업 등과 협의하여 실행방안 마련</li> <li>- 디지털플랫폼정부 구현을 위한 표준화 및 가이드라인 정립을 위해 과제 발굴 및 검토, 관계부처·관련 민간 등과 협의 추진</li> </ul> <p style="text-align: center;">〈 「디지털플랫폼정부 실현계획」 상 관련 과제 〉</p> <div style="border: 1px dashed black; padding: 5px;"> <ul style="list-style-type: none"> <li>▶ 1.3.8 클라우드 네이티브 적용 의무화 및 클라우드 네이티브 전환 지원</li> <li>▶ 1.3.9 DPG 구현을 위한 표준화 및 가이드라인 개발</li> </ul> </div>   |
| 13 | <p><b>DPG 허브 TF</b></p> <ul style="list-style-type: none"> <li>▪ [담당] 송호철 위원</li> <li>▪ [담당] 유병선·전재남·심승배·김병무 전문위원</li> </ul>         | <ul style="list-style-type: none"> <li>- DPG 허브, 데이터레이크, 초거대 AI 등을 통해 민·관 데이터·서비스를 융합·분석하여 새로운 가치 창출에 기여 할 수 있는 과제를 발굴하고 혁신 테스트베드 등을 통해 개발·테스트·개념검증 (PoC) 등 실증 추진</li> </ul> <p style="text-align: center;">〈 「디지털플랫폼정부 실현계획」 상 관련 과제 〉</p> <div style="border: 1px dashed black; padding: 5px;"> <ul style="list-style-type: none"> <li>▶ 1.3.1 데이터·서비스 융합을 위한 통합플랫폼(DPG 허브) 구축</li> <li>▶ 1.3.2 공공·민간 데이터 융합 인프라 마련</li> <li>▶ 1.3.3 초거대 AI 인프라 마련</li> <li>▶ 1.3.4 공통기능 활용을 위한 범부처 공용서비스 빌딩블록 구축</li> <li>▶ 1.3.5 혁신서비스를 구매·사용하는 마켓플레이스 제공</li> <li>▶ 1.3.6 혁신서비스의 개발환경을 제공하는 플레이그라운드 조성</li> <li>▶ 1.3.7 혁신서비스를 실제환경에서 검증(Field-Trial)·구현 지원</li> </ul> </div> |

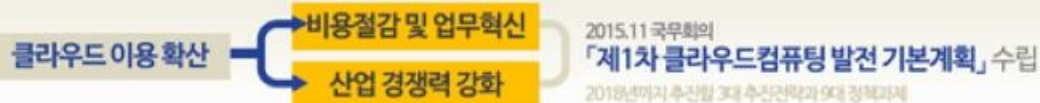
ZDNET 디플정위원회, 16개 TF체제로 전면 개편  
 - <https://zdnet.co.kr/view/?no=20230529120005>

## 클라우드 네이티브 전환 로드맵 수립 설명회

- 행정안전부, 2023년 5월 3일

### 국내 클라우드 정책(1차 기본계획)

2021년 클라우드 선도국가 도약을 위한 발걸음



### 「제1차 클라우드컴퓨팅 발전 기본계획」 3대 추진전략 + 9대 정책과제

#### 공공부문의 선제적인 클라우드 도입

- 1 공공부문의 민간 클라우드 이용 활성화
- 2 지능정부 구현을 위한 정부 클라우드 전환 가속화

#### 민간부문 클라우드 이용 확산

- 3 안전한 클라우드 이용 환경 마련
- 4 클라우드 친화적인 제도개선
- 5 중소기업 및 산업의 혁신지원

#### 클라우드 산업성장 생태계 조성

- 6 클라우드 기술 경쟁력 강화
- 7 클라우드 서비스 해외진출 촉진
- 8 클라우드 전문 인력 양성
- 9 클라우드 데이터센터 경쟁력 강화

1차 클라우드 컴퓨팅 발전 기본계획 당시 정부의 클라우드 방향성(출처: NIA)

### 공공 클라우드 네이티브 전환 로드맵

|       |  |
|-------|--|
| 추진 배경 | <ul style="list-style-type: none"> <li>• 정부 재정투자 방향 변화</li> <li>• 보안인증제 개편</li> <li>• 신기술 보편화 등</li> </ul>                                 |
| 추진 환경 | <ul style="list-style-type: none"> <li>• 행정안전부 전환 사업 예산 축소</li> <li>• 정부 정책 "민간 클라우드 우선 이용"</li> <li>• "클라우드 네이티브 우선 적용"으로 발전 등</li> </ul> |
| 추진 방향 | <ul style="list-style-type: none"> <li>• 범정부 정보자원 등록 및 관리시스템에 등록된 <b>모든 시스템의 클라우드 네이티브 전환</b></li> </ul>                                   |
| 전환기간  | <ul style="list-style-type: none"> <li>• 2024년부터 2030년까지 7개년 추진</li> </ul>   |
| 추진 목표 | <ul style="list-style-type: none"> <li>• 2023년 시범 사업</li> <li>• 2024년 10% 달성</li> <li>• 2025년 30% 달성</li> <li>• 2026년 이후 70% 달성</li> </ul> |

# 디지털플랫폼정부는 '클라우드 네이티브'로 구축하는 게 핵심이다



**디지털플랫폼정부는** 초거대AI를 비롯한 디지털 기술을 적극 도입, 칸막이를 없애고 '원팀 정부'로 거듭나는 것을 목표로한다. 이를 위한 **공공 플랫폼을 단순 클라우드 전환이 아닌, 클라우드를 클라우드답게 쓸 수 있도록 '클라우드 네이티브'로 구축하는 게 이번 정책의 핵심**이다. 공공SW(소프트웨어) 사업 고질병을 극복할 해법을 제시하면서 각종 미래 IT 산업 육성의 요람이 될 것으로 기대된다.

전자신문 etnews Conference allshowTV Engli

경제·금융 전자·모빌리티 통신·미디어·게임 소재·부품 SW·보안 산업·에너지·환경 품

## [기획]행안부, 클라우드 네이티브 확산 위한 지원사업 추진

발행일: 2022-06-29 15:00 | 시점: 2022-06-30 | 18면

클라우드 네이티브는 클라우드 성숙도 단계 중 최고 단계로, 클라우드의 기능과 장점을 최대한 활용해 애플리케이션을 구축·실행하는 것을 의미한다.

## [테크&포커스] 정부·공공 시스템도 '클라우드 최화'... '디플정'의 도전

입력: 2023-04-16 16:07 | 평용현 기자

IT 폰트 N 기자 구

尹정부 핵심과제 청사진 공개  
민첩한 개발·유연한 확장 가능  
초거대 AI 등 디지털기술 도입  
홈택스 등 사이트 통합도 추진

### 디지털플랫폼정부 기대 효과(2026년)

|   |  |
|---|--|
| <b>대국민 서비스 강화</b> <ul style="list-style-type: none"> <li>공공서비스 1500여종 연계·통합</li> <li>해택 일리미 총 1021종 제공</li> <li>정부서류 제로화로 연 2조원 절감</li> </ul>             | <b>민·관 성장 플랫폼 구축</b> <ul style="list-style-type: none"> <li>Sea5 기업 1단계 육성</li> <li>AI 유니콘 기업 5개 육성</li> <li>DPG 수출 연 20억달러 달성</li> </ul>                |
| <b>정부·공공 시스템 혁신</b> <ul style="list-style-type: none"> <li>공공부문 종이 사용량 50% 감축</li> <li>대상 시스템 70% 클라우드 네이티브 전환</li> <li>공학-기초로 이원화된 지자체 시스템 통합</li> </ul> | <b>사이버보안·개인정보보호</b> <ul style="list-style-type: none"> <li>주요 분야 사이버데이터 유통체계 구축</li> <li>제로트러스트 등 새로운 보안체계 도입</li> <li>신기술 공공 적용, 보안산업 경쟁력 강화</li> </ul> |

### 모놀리식 아키텍처와 마이크로서비스 아키텍처(MSA) 차이

\*실용성을 갖춘 모놀리식 아키텍처와 달리 MSA는 각 서비스가 병행·분산돼 유연성과 가용성이 뛰어나다.

## 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 설계를 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>



# Native

- 네이티브(Native)의 사전적 의미는 '선천적인', '본래' 등이다.
- 클라우드 네이티브는 "클라우드가 '클라우드 다룰 수 있도록' 애플리케이션을 구축, 실행하는 방식"

'어린이 또는 성인이 되어 언어를 배운 것이 아닌 태어나서 부터 특정 언어를 사용해 온 사람'

## 네이티브 스피커



'어린이 또는 성인이 되어 스마트폰을 접한 것이 아닌 유아기부터 스마트폰을 사용해 온 사람'


## 스마트폰 네이티브



'애플리케이션을 계획/설계할 때부터 클라우드 특징과 장점을 기반으로 개발/운영'

## 클라우드 네이티브



A wide-angle photograph of a cable-stayed bridge spanning a body of water at sunset. The sun is low on the horizon, creating a bright, shimmering reflection on the water. The bridge's two tall, slender pylons are silhouetted against the golden sky, with numerous stay cables fanning out to support the deck. In the background, a prominent, pointed skyscraper stands out against the horizon. The sky is filled with soft, wispy clouds, and the overall atmosphere is serene and dramatic.

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)

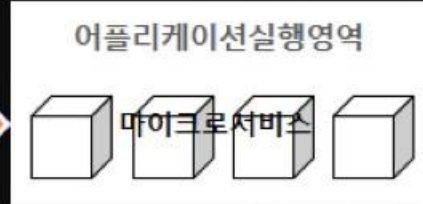
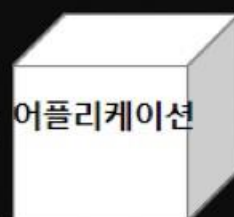


### 기존 어플리케이션

### AS-IS 시스템

### 클라우드 네이티브 어플리케이션

- 기존 어플리케이션 그대로
- 기존 배포 방법



- 기존 어플리케이션 또는 MSA 로 전환
- 컨테이너 이미지 배포

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



- 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 )                          |

# Cloud Immigrant 와 Cloud Native 애플리케이션 비교

| 분류          | Cloud Immigrant 애플리케이션 | Cloud Native 애플리케이션         |
|-------------|------------------------|-----------------------------|
| 실행 환경       | 가상화 중심                 | 컨테이너 중심                     |
| 구조          | Monolithic             | Microservice                |
| 확장          | Scale Up (수직 확장)       | Scale Out (수평 확장)           |
| 결합          | 크고 조밀 결합               | 느슨하게 & 서비스 기반               |
| 인프라 의존성     | 인프라 의존                 | 인프라 독립적으로 이동성 보장            |
| Delivery 방법 | 폭포수형으로 장기간 개발          | Agile & Continuous Delivery |
| 빌드 배포       | 수작업, 긴시간               | CI/CD, 짧은 시간과 지속적 배포        |
| 개발 도구       | 로컬 IDE 개발 도구           | 클라우드 기반의 지능형 개발 도구          |
| 조직구조        | 사일로화 된 개발, 운영, 보안 팀    | DevSecOps, NoOps 또는 협업      |

# CNCF Cloud Native Definition v1.0

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.



# Cloud Native Computing Foundation

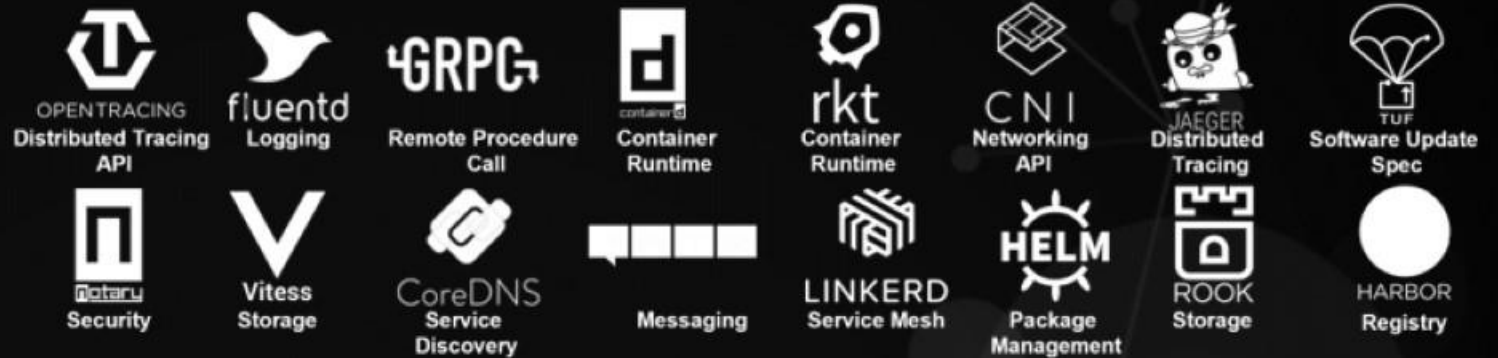


- Non-profit, part of the Linux Foundation; founded Dec 2015

## Graduated



## Incubating



- Platinum members:



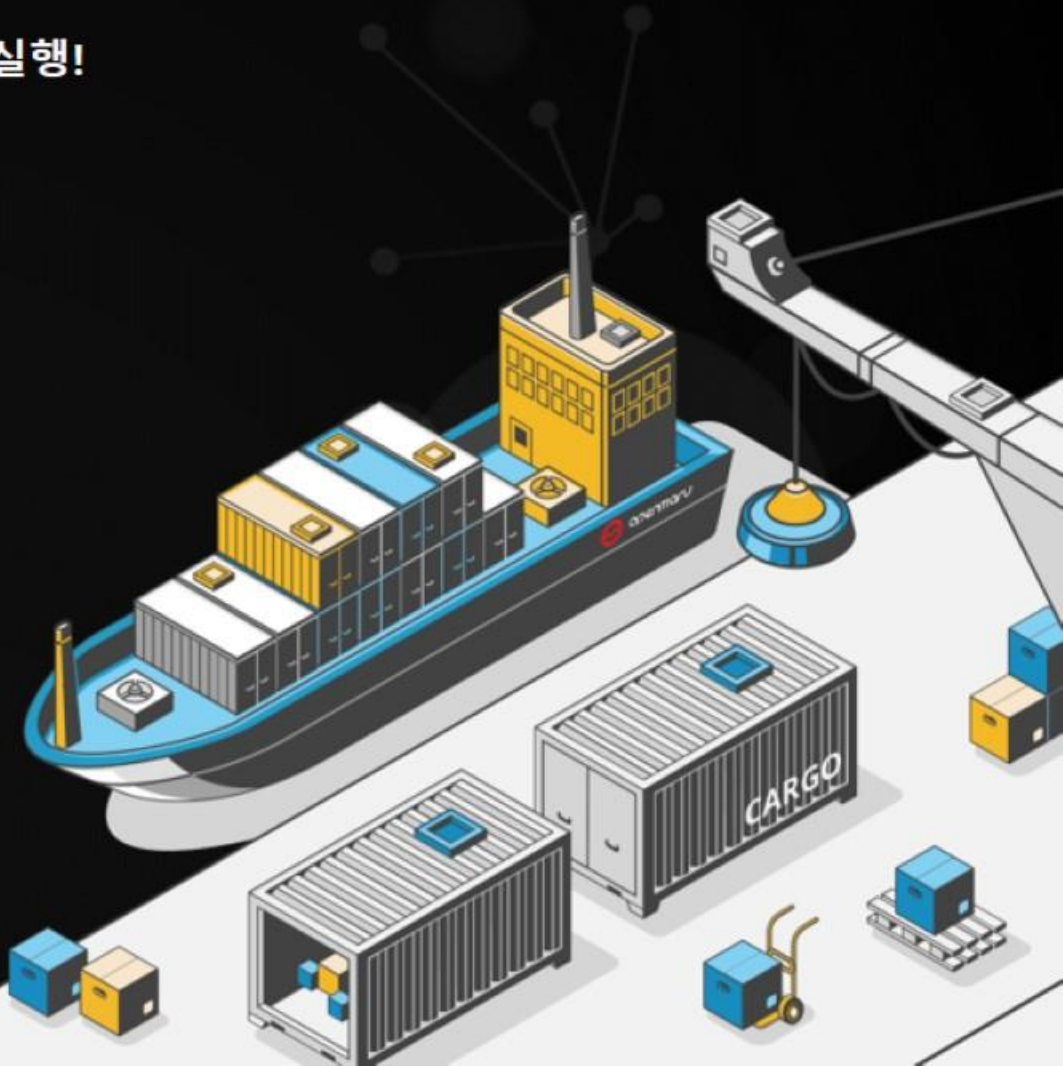
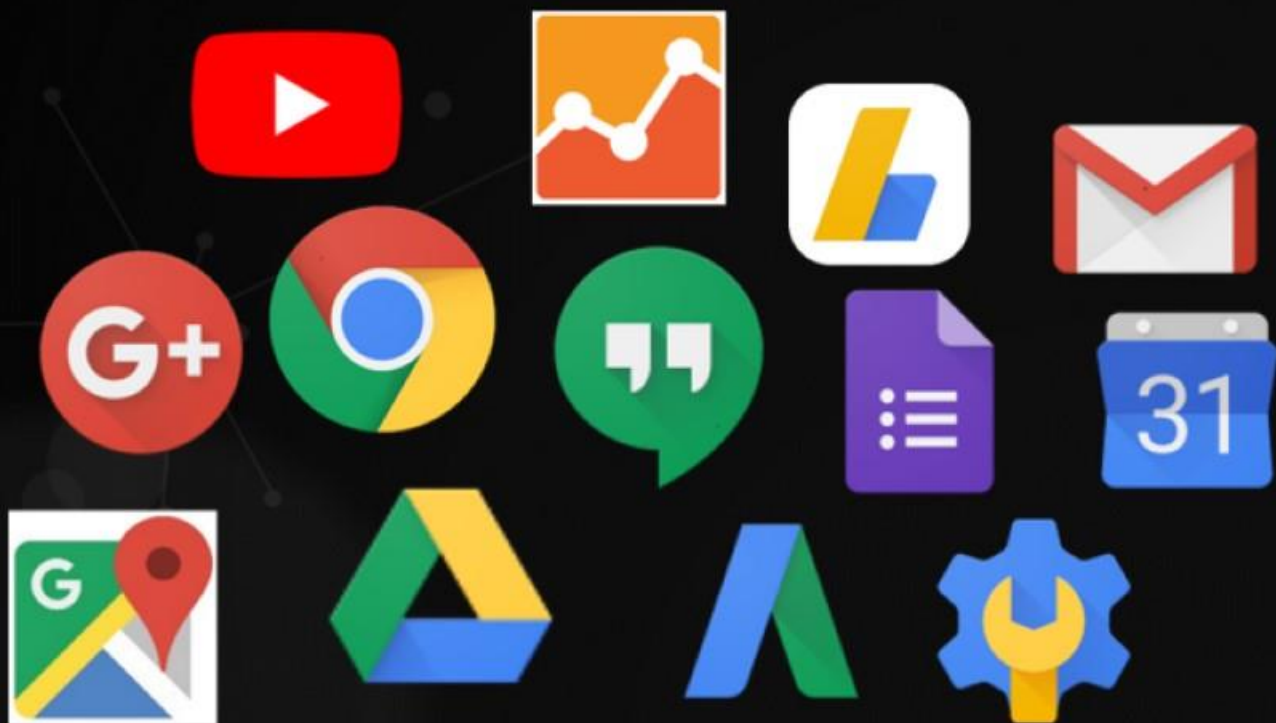


Cloud Native 기술을 통한 Open Hybrid Cloud 구현

# Google , Container, Kubernetes & Cloud Native

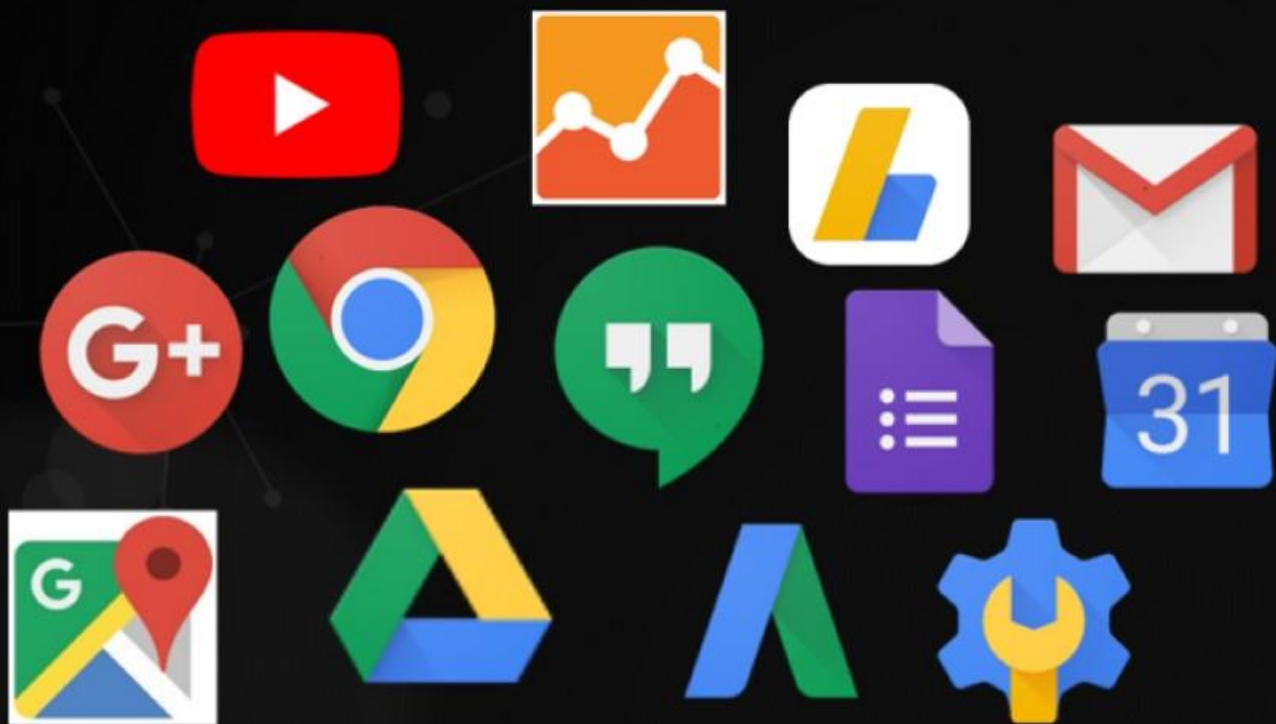
# Google 의 모든 서비스는 [ ] 에서 실행

- Gmail , 검색, 지도 ...
- MapReduce , GFS , Colossus ...
- Google Compute Engine 가상 머신도 [ ] 에서 실행!
- 매주 20 억개 이상의 [ ] 를 실행 중



# Google 의 모든 서비스는 **컨테이너** 에서 실행

- Gmail , 검색, 지도 ...
- MapReduce , GFS , Colossus ...
- Google Compute Engine 가상 머신도 **컨테이너** 에서 실행!
- 매주 20 억개 이상의 **컨테이너** 를 실행 중



# GOOGLE 과 컨테이너

- Google의 업무 방식

Gmail에서 YouTube, 검색에 이르기까지 Google의 모든 제품은 컨테이너에서 실행됩니다.

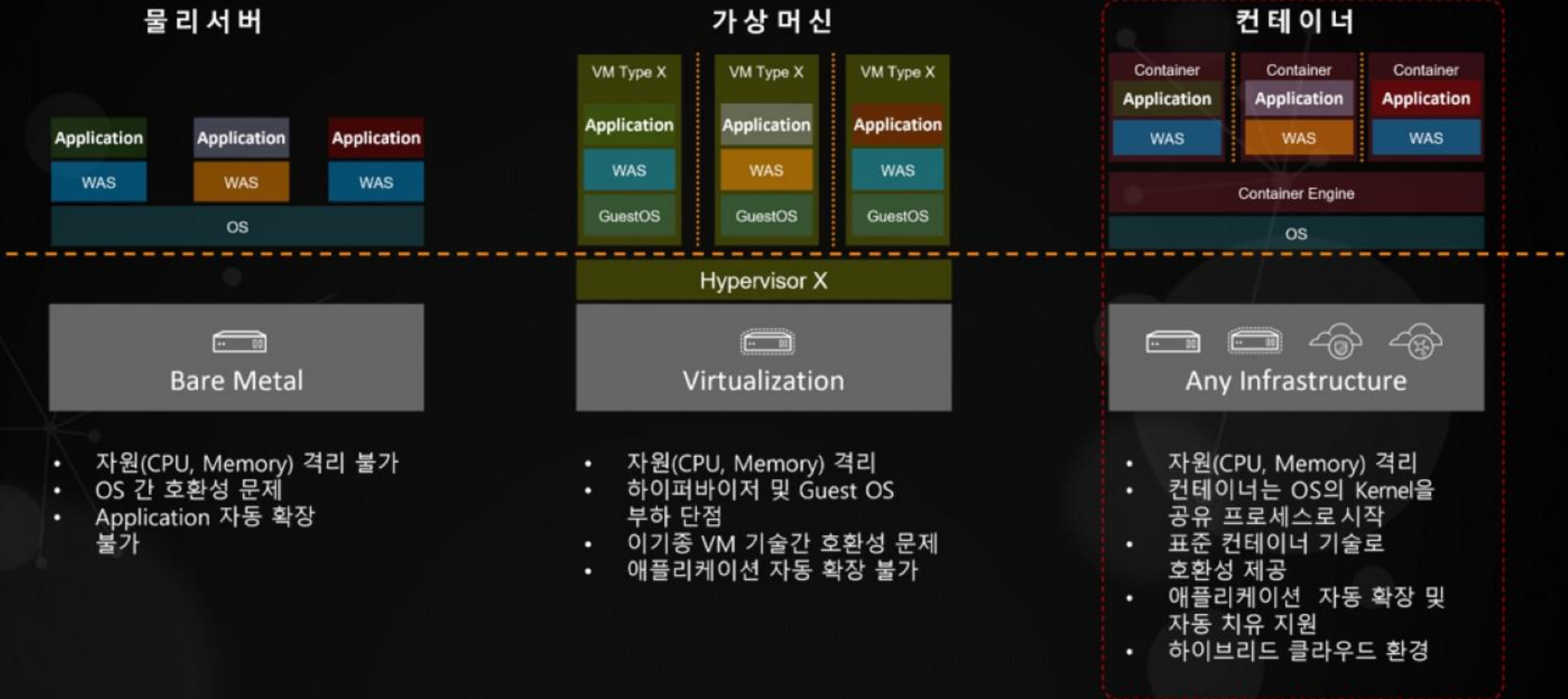
개발팀은 컨테이너화를 통해 더욱 신속하게 움직이고, 효율적으로 소프트웨어를 배포하며 전례 없는 수준의 확장성을 확보할 수 있게 되었습니다. Google은 매주 수십억 개가 넘는 컨테이너를 생성합니다. 지난 10여 년간 프로덕션 환경에서 컨테이너화된 워크로드를 실행하는 방법에 관해 많은 경험을 쌓으면서 Google은 커뮤니티에 계속 이 지식을 공유해 왔습니다.

초창기에 cgroup 기능을 Linux 커널에 제공한 것부터 내부 도구의 설계 소스를 Kubernetes 프로젝트로 공개한 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.



# 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

- 자원(CPU, Memory) 격리
- 하이퍼바이저 및 Guest OS 부하 단점
- 이기종 VM 기술간 호환성 문제
- 애플리케이션 자동 확장 불가

- 자원(CPU, Memory) 격리
- 컨테이너는 OS의 Kernel을 공유 프로세스로 시작
- 표준 컨테이너 기술로 호환성 제공
- 애플리케이션 자동 확장 및 자동 치유 지원
- 하이브리드 클라우드 환경

# How does one build apps for the cloud?

 KVM  Google Cloud

 Xen Project

 aws

 Hyper-V

 Microsoft Azure

 Xen Project

 Alibaba Cloud

 Xen Project

 IBM Cloud

Hypervisor

Public Cloud

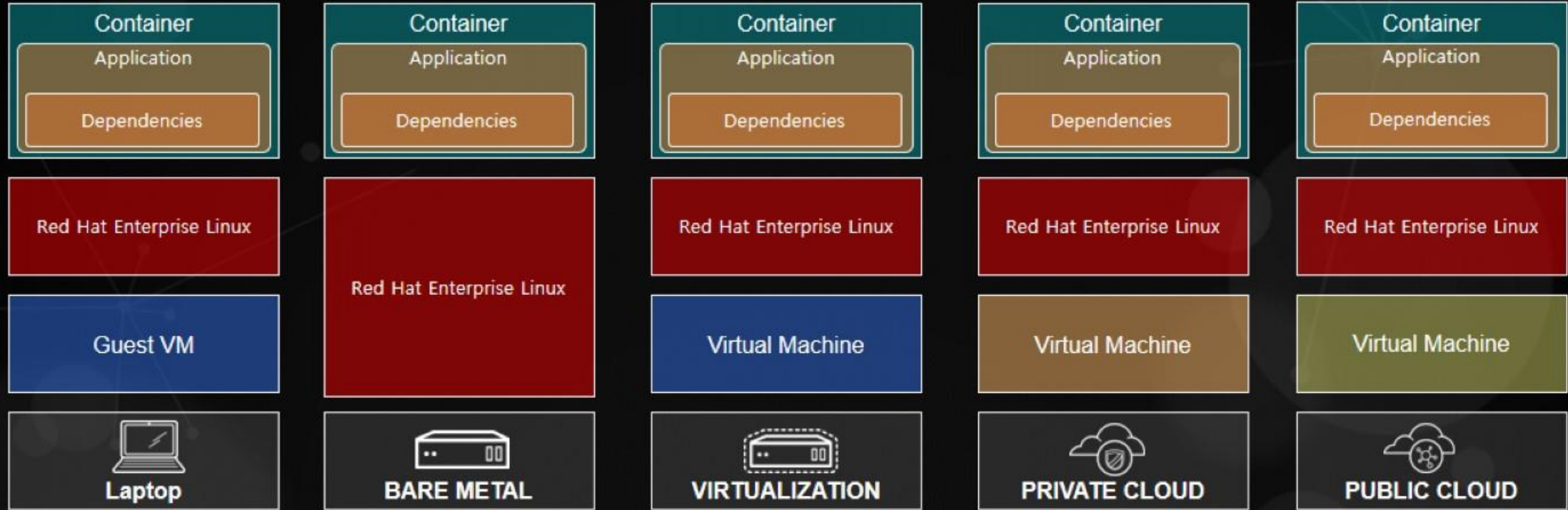


Virtual Machine

**Write once,  
run anywhere?**

# 컨테이너의 이동성

- Linux 커널 구조를 이용하고있다 = Linux 움직이는 모든 환경에서 이동성





## About Kubernetes

- 쿠버네티스(K8s)는 컨테이너화된 애플리케이션을 자동으로 배포, 스케일링 및 관리해주는 오픈소스 소프트웨어
- 쿠버네티스", "쿠베르네티스", "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

## 2000 년 - Java 를 통한 Vendor Lock-In 해제



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





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

*미국의 정치가 토머스 제퍼슨*

Cloud Native 기술을 통한 Open Hybrid Cloud 구현

클라우드 네이티브는  
어떻게 발전하고 있는가?

# 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

Developer Experience 장점

- Kubernetes(컨테이너 오케스트레이션)

- 애플리케이션 실행을 먼저 생각할 때
- 어떤 인프라를 만들지 주축으로 설계된 인프라 기반

Reconcillation model의 정교함

2016년 ~  
클라우드 네이티브

~2000년  
물리서버

2001~2009년  
가상화 기술 1세대

2010~2015년  
가상화 기술 2세대

# 클라우드 네이티브 정의

## 클라우드 네이티브 란?

### 클라우드 네이티브 (형용사/명사)

클라우드 컴퓨팅의 장점을 최대한 활용할 수 있는  
(효율적인 자원이용, 탄력적 수요 대응 등)  
정보시스템 분석·설계·구현 및 실행하는 환경

### 클라우드 네이티브 애플리케이션

클라우드 환경에서 실행되는 애플리케이션



## CNCF (Cloud Native Computing Foundation) v1.0

클라우드 네이티브 전환할 수 있는 기술 정의 및 오픈 소스를 관리하는 단체

- 퍼블릭, 프라이빗, 하이브리드 클라우드 환경에서 확장성 있는 애플리케이션
- 컨테이너, 서비스 메시(Mesh), 마이크로서비스(Micro Service) 인프라 구조, 선언적 API로 접근
- 자동화, 회복성, 편리성, 가시성을 갖는 느슨하게 결합된 시스템 (개발 및 실행 환경)
- 엔지니어는 최소한의 수고로, 영향력이 크고, 예측 가능한 변경을 할 수 있는 기술 정의

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

# 클라우드 네이티브 정의

- 클라우드 네이티브(Cloud Native)는 “클라우드의 장점을 최대한 활용하여 정보시스템을 구축 및 실행하는 환경”
- 2015년 최초로 클라우드 네이티브라는 용어를 사용한 리눅스 재단은 CNCF(Cloud Native Computing Foundation) 재단을 설립하여 클라우드 네이티브 관련 기술을 정의하고 오픈소스를 관리



Source - 클라우드 네이티브 정보시스템 구축을 위한발주자 안내서

# 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

# How does one build apps for the cloud?

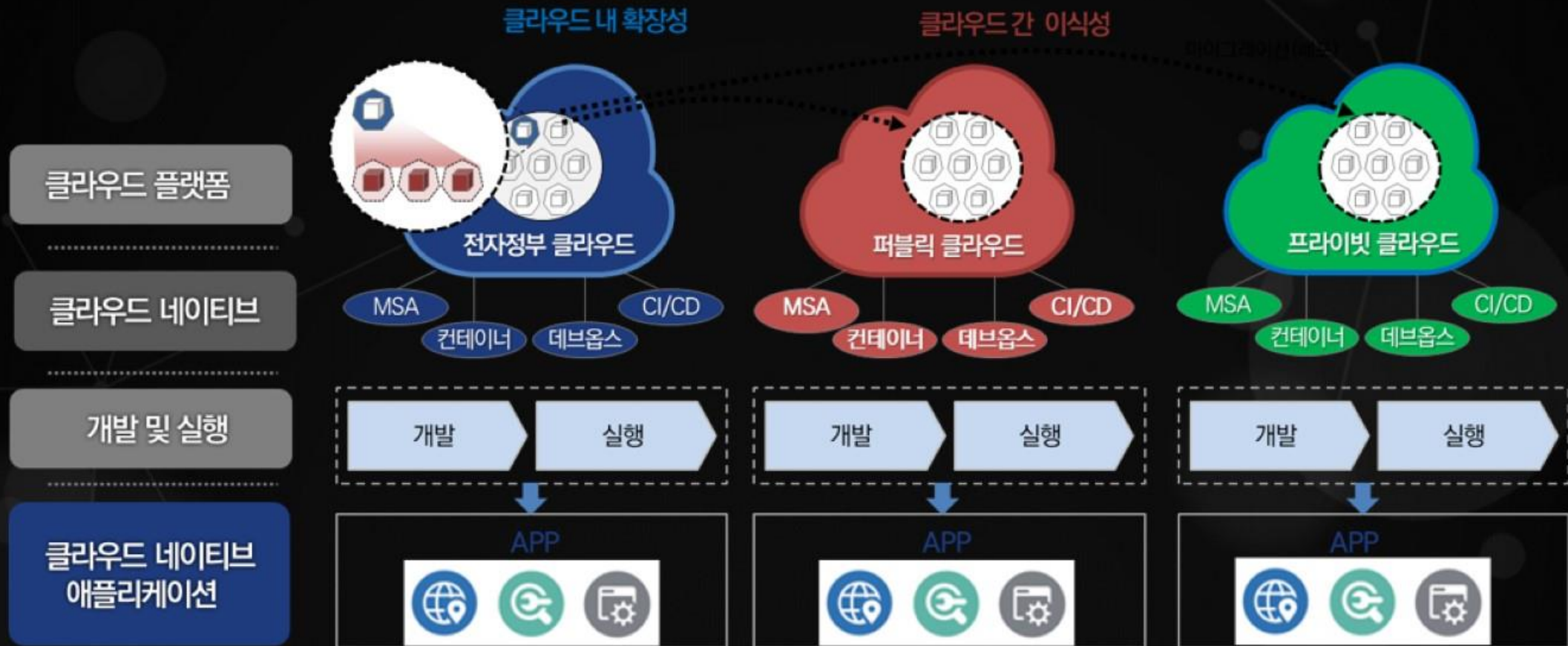
- 가상화 기반 IaaS Cloud





# 클라우드 네이티브

- 클라우드 네이티브 애플리케이션은 클라우드 네이티브 환경에서 개발, 실행되는 애플리케이션
- 프라이빗, 퍼블릭 및 하이브리드 클라우드 환경 전체에 지속적인 개발과 자동화된 관리 환경을 제공하기 위해 설계된 애플리케이션이므로 클라우드 내에서 확장이 가능하고, 어떤 클라우드에서도 이식이 가능하다



Source - 클라우드 네이티브 정보시스템 구축을 위한발주자 안내서

# 클라우드 네이티브 변화



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

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 MSA 전환”



## 명확한 경계가 가능한 시스템

- 샘뉴먼(2019, 저서): “해당분야를 제대로 이해하지 못해 적절한 경계를 찾기 어렵다면 MSA 전환 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”

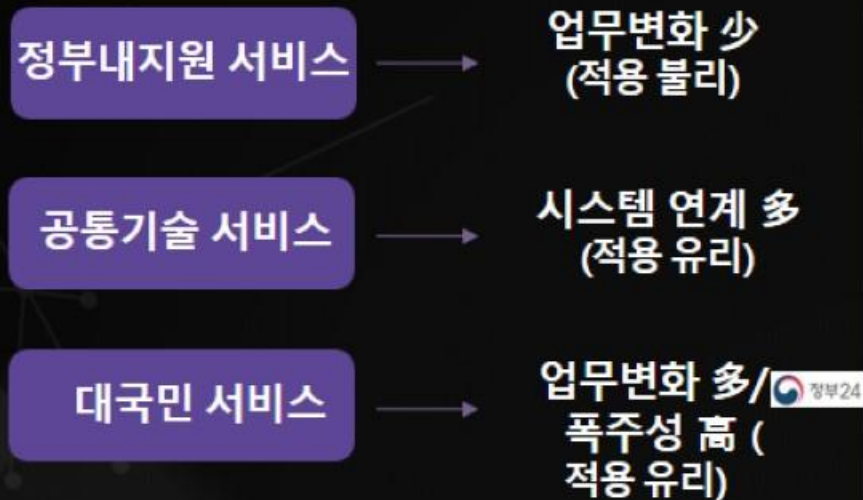
# 클라우드 네이티브 대상 업무선정 방향 (전문가 의견)

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



## 정부내 전환가능 업무 식별

- SRM, CRM, ERP의 전환가능 업무



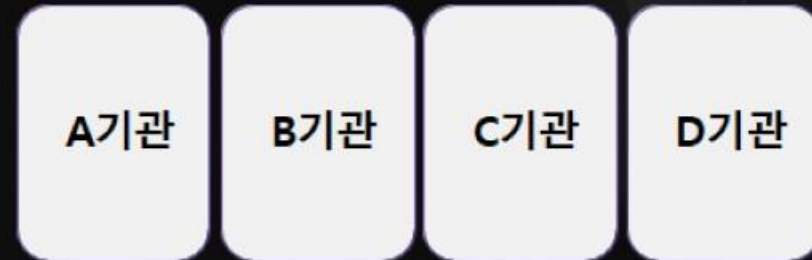
## 기관·시스템특성을 반영한 업무선정

- 시스템 특성 반영 (시스템복잡성)



※ 국가및기초자치 단체226, 공공기관 338개기관업무대상

- 정부정책 특성 반영 (제도개선이많은업무)



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

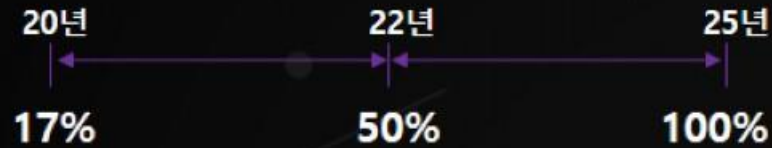
# 클라우드 네이티브 적용 검토

- 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  
54%

민간 클라우드센터  
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(예시스템중설, 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, IoT 등) 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-테스트-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

클라우드 네이티브 정보시스템 구축을 위한

## 발주자 안내서



- 사업 추진 방향성과 사업 범위 작성 시 MSA, 컨테이너, 데브옵스 및 CI/CD 구성요소 관련 내용을 포함하여 클라우드 네이티브 사업임을 명시한다.

### [그림 5-2] 사업 추진 방향성 작성 예시

#### MSA, 컨테이너 구성요소를 포함하여 사업 추진 방향성 작성

- MSA 기반의 컨테이너 형태로 구현된 공간정보 서비스 기능(공간정보 표준 프레임워크)을 효율적으로 운영 관리하기 위한 개발형 공간정보 플랫폼 구축
- 서비스 수요 증감에 따라 유연하게 컨테이너가 확장 및 축소가 가능한 운영관리 기능 및 컨테이너 동작 여부에 따른 상태 모니터링 기능 제공

[출처: 디지털 권역자원 통합관리시스템 재구축 및 운영 제안요청서, 한국관광공사]

### [그림 5-3] 사업 범위 작성 예시

#### MSA, 컨테이너, CI/CD 구성요소가 포함된 사업 범위 작성

- 다중화 기반 마이크로 서비스 구축
  - (마이크로 서비스 아키텍처 구축) 컨테이너 관리 기능, API 게이트웨이 관리 기능
  - (전자정부 표준프레임워크 적용) 실행환경 구성, 개발환경 구성, 운영환경 구성, 관리환경 구성
  - (인프라 구축) 인프라 가상화 자동화 구현, HW-SW 구축, 보안관리
- 클라우드 서비스 및 정의된 API를 통해 콘텐츠를 제공하는 콘텐츠 관리 기능 중심으로 시스템 구축
- 마이크로 서비스를 독립적으로 개발/배포/관리할 수 있는 프레임워크 제공
  - 컨테이너 관리: 여러 대의 서버에서 여러 개의 컨테이너를 편리하게 관리하도록 서비스 메시 컨테이너 오케스트레이션 등 자동화 기반의 컨테이너 배포 구현
- 마이크로 서비스 구현을 위한 가상화/자동화 환경을 제공
  - 가상화: 물리적/논리적 서버 클러스터 구성을 통해 시스템 가용성 향상 및 가상 서버 복제 및 수명 확장을 통해 시스템 확장성 확보
  - 자동화: 서비스 요청관리, 수요관리, 변경관리 등 사용자의 서비스 요청에 대한 해당 서비스를 사용자에게 제공하기 위해 다음과 같은 자동화된 서비스 제공관리 환경 구축

[출처: 디지털 권역자원 통합관리시스템 재구축 및 운영 제안요청서, 한국관광공사]

### [그림 5-5] 상세 요구사항 작성 예시

| 요구사항 분류    | 클라우드 서비스 요구사항               |   |
|------------|-----------------------------|---|
| 요구사항 고유번호  | CSR-003                     |   |
| 요구사항 명칭    | 컨테이너 기반 서비스 메시 및 오케스트레이션 구현 |   |
| 요구사항 상세 설명 | 세부 내용                       | <ul style="list-style-type: none"> <li>여러 대의 서버에서 여러 개의 컨테이너를 편리하게 관리하도록 서비스 메시 기능 제공</li> <li>컨테이너를 적절한 서버에 배포하고 상태를 유지하기 위한 스케줄링</li> <li>여러 대의 서버를 1대의 서버처럼 관리하고, 가상 네트워크를 이용해 접근하기 위한 클러스터링</li> <li>컨테이너의 IP/포트정보를 서비스 메시 스트러티에 저장하며, 동적으로 변화하는 라스스의 위치를 API 게이트웨이가 검색하기 위한 서비스 디스커버리 기능 제공</li> <li>API 요청에 대한 최적의 경로를 지능하기 위한 다양한 API 라우팅 구현</li> <li>서비스 간 부하 분산을 위한 로드밸런싱</li> <li>오로스케일링 시 서버 수 지정, 서버의 사양 정의, 서버 실행 시작과 워밍업 시간 지정 등 프래틱 입출에 서버 스토리지, 네트워크 등 인프라 자원의 자동 확장 및 축소를 자동화하여 서비스 상태에 따른 적정 서버 용자를 통해 유연한 서비스를 제공하도록 오로스케일링 지원</li> <li>특정 서비스에 오류가 발생하거나 실행 실패 시 신속하게 이전 버전으로 되돌아가도록 복구(Rollback) 기능 지원</li> <li>표준화된 로그 이벤트 수집 및 분석, 서비스 간 호출 추적 및 성능 관리 등 로깅 및 로그 분석 등</li> </ul> |

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]

| 요구사항 분류    | 클라우드 서비스 요구사항   |  |
|------------|-----------------|--|
| 요구사항 고유번호  | CSR-004         |  |
| 요구사항 명칭    | API 게이트웨이 관리 기능 |  |
| 요구사항 상세 설명 | 세부 내용           | <ul style="list-style-type: none"> <li>API 호출을 위한 토큰 발급 및 인증, 엔드포인트별 API 호출 인증 및 인가, 접근 정책에 특정 클라이언트와 API 호출 불허에 의한 접근 제어 기능 등 API 인증 및 인가 처리</li> <li>동일 API를 클라이언트나 마이크로 서비스에 따라 다른 엔드포인트를 통해 서비스를 제공하도록 API 라우팅을 지원하고, 동일 API를 여러 개의 클라이언트/마이크로 서비스 별로 엔드포인트 제공</li> <li>로그, 인증 등 공통 기능을 중복 개발 또는 처리하지 않도록 요청과 응답의 표준화 및 공통 로직 처리</li> <li>동일 API를 HTTP, REST, XML, 웹 서비스 등 클라이언트와 마이크로 서비스별로 상이한 프로토콜로 서비스하기 위한 프로토콜 변환 처리</li> <li>동기, 비동기 등 API를 호출하는 메시지 패턴을 변경할 수 있도록 메시지 호출 패턴 변환 기능 제공</li> <li>호출 횟수, 전송 용량, 네트워크 대역폭 등 서비스 레벨을 클라이언트나 마이크로 서비스별로 조정할 수 있도록 QoS/Quality of Service 설정 기능 제공</li> <li>API 호출 패턴 분석, API 호출 실행 및 접근 상태 분석, 요청 IP/클라이언트/일시 등 API 호출에 대한 로깅 및 모니터링 등</li> </ul> |

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]



# 개발자 안내서 - 클라우드 네이티브 정보시스템 구축

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

클라우드 네이티브 정보시스템 구축을 위한  
**개발자 안내서**

중앙안전부  
NIA 한국지능정보사회진흥원

[그림 7-6] 참고, 도메인, 서브 도메인, 바운디드 컨텍스트

|                                      |  |
|--------------------------------------|--|
| <b>도메인 개념</b>                        | <ul style="list-style-type: none"> <li>소프트웨어가 해결해야 할 문제 영역</li> <li>도메인은 서브 도메인으로 구성됨</li> <li>세부 업무 영역 또는 조직 단위</li> </ul>  |
| <b>서브 도메인 개념</b>                     | <ul style="list-style-type: none"> <li>도메인 내 문제 해결을 위해 도메인을 여러 개의 서브 도메인으로 나눔</li> <li>서브 도메인은 핵심, 지원, 일반 서브 도메인으로 분류됨</li> <li>핵심 서브 도메인: 핵심적인 비즈니스 로직이 담긴 영역으로, 차별화된 영역</li> <li>지원 서브 도메인: 핵심 서브 도메인을 기능적으로 보조할 수 있는 영역</li> <li>일반 서브 도메인: 범용적으로 사용할 수 있는 도메인</li> </ul> |
| <b>바운디드 컨텍스트 개념</b>                  | <ul style="list-style-type: none"> <li>도메인의 문제를 해결하기 위한 솔루션 영역</li> <li>도메인 모델이 존재하는 명시적인 경계</li> <li>바운디드 컨텍스트는 전체 비즈니스 도메인을 여러 개의 서브 도메인으로 나눔 후, 서브 도메인 내 동일한 맥락을 경계로 구분하여 바운디드 컨텍스트를 도출함</li> </ul>   |
| <b>바운디드 컨텍스트, 서브 도메인, 마이크로서비스 관계</b> | <ul style="list-style-type: none"> <li>바운디드 컨텍스트와 서브도메인의 관계는 1:1이 이상적임</li> <li>바운디드 컨텍스트가 커질수록 더 작은 단위로 분할 가능하므로 1:N 관계가 될 수도 있음</li> </ul> <p>1개의 바운디드 컨텍스트는 여러 개의 마이크로서비스로 분리될 수 있음</p>   |

[그림 7-24] 클라우드 네이티브 애플리케이션 아키텍처 참조 모델

범례: 내부 아키텍처, 외부 아키텍처

K8S 클러스터 구성: (마스터노드/워커 노드)  
- 운영환경  
- 개발/검증 환경

1 API 게이트웨이  
- 인증/인가  
- API 조합

2 서비스 메시  
- 서비스 발견  
- 서비스 리우팅  
- 로드밸런싱  
- 설정관리

3 런타임 플랫폼 (컨테이너)  
- 컨테이너 관리

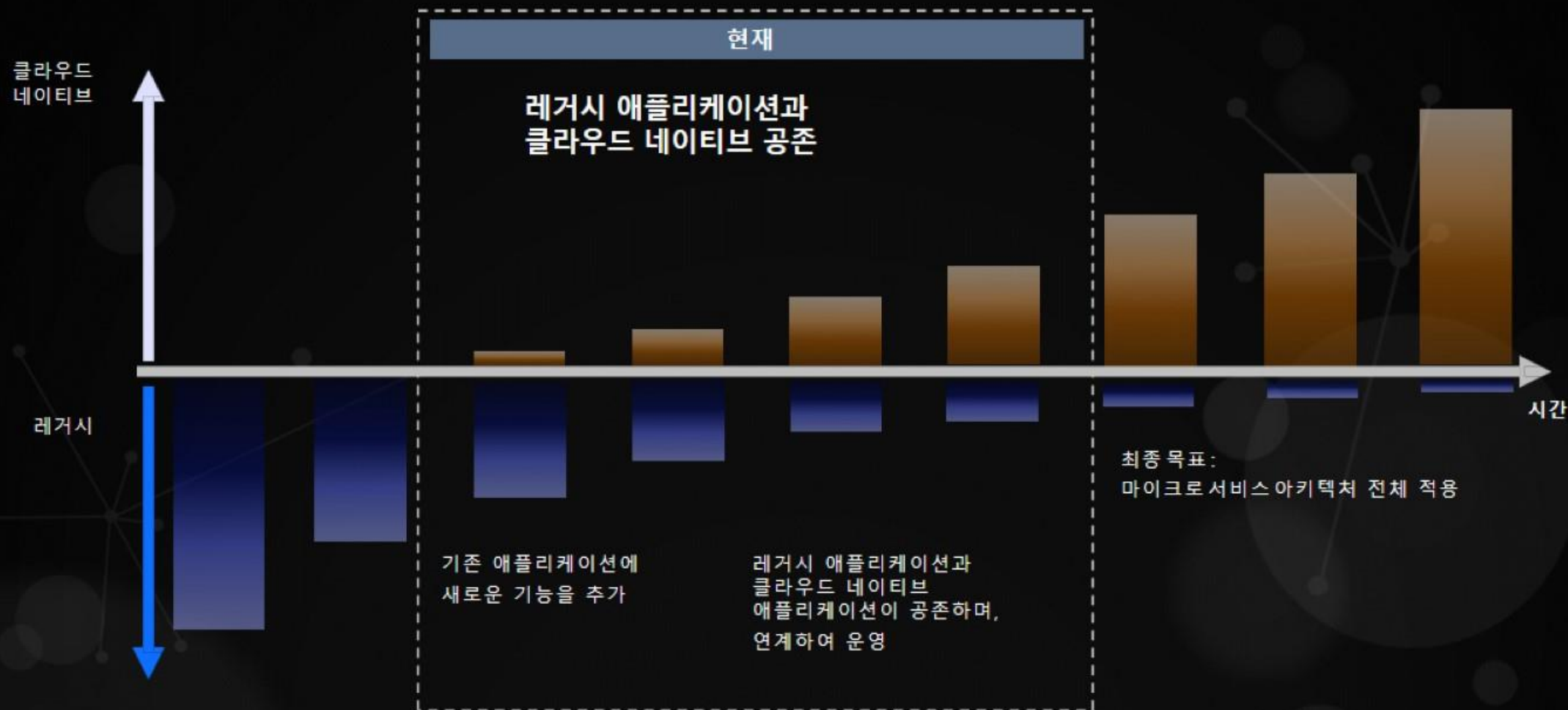
4 CI/CD  
- 형상관리  
- 빌드 자동화  
- 빌드 이미지 저장소  
- 배포 자동화

5 백엔드 서비스  
- DB  
- 스토리지  
- MQ  
- 캐시

6 텔레메트리  
- 모니터링

[출처: 가트너, IBM 등 자료 분석 정리]

# 레거시와 클라우드 네이티브 애플리케이션 전환



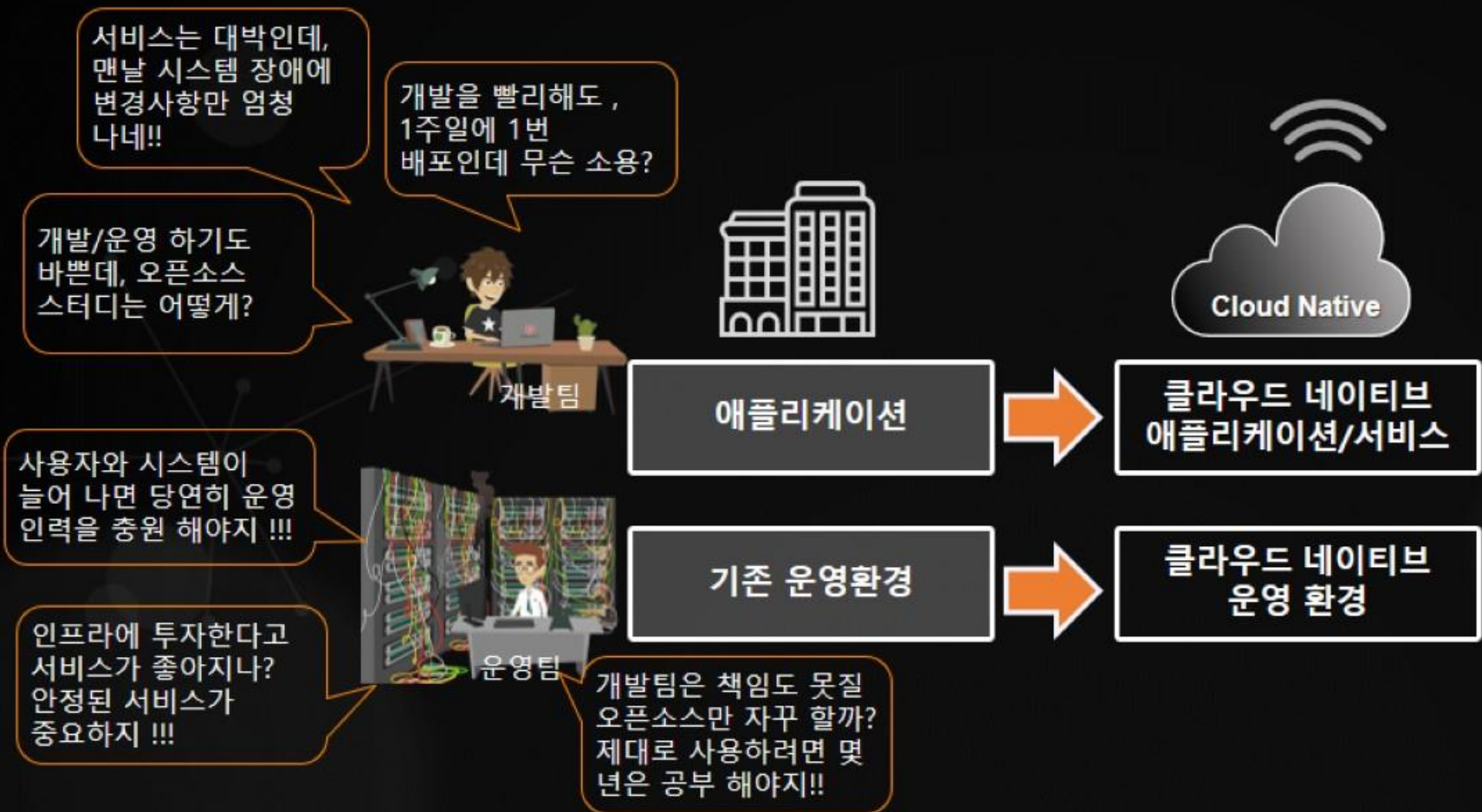
Source : IBM



# 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



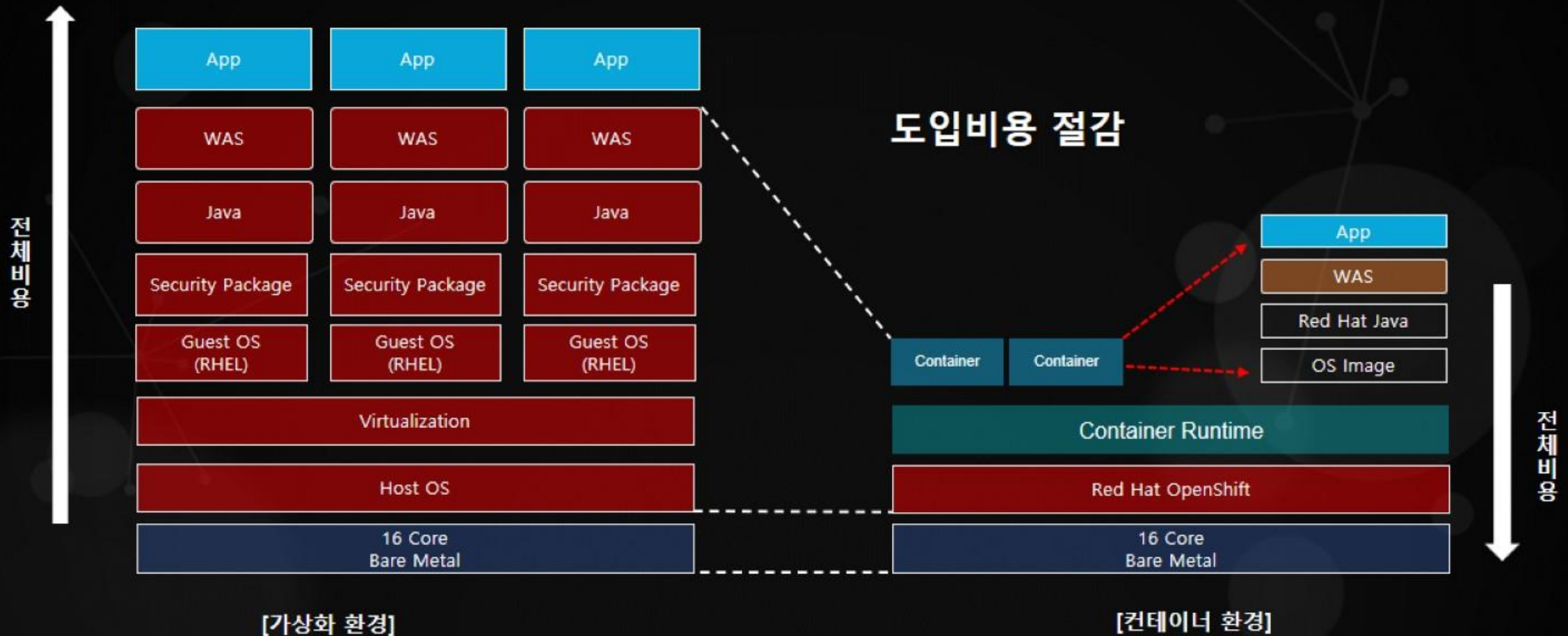
- 혁신적인 IT 환경 구축**  
IT 조직의 운영 비용 절감과 비즈니스에 대한 민첩성 증대
- 클라우드에 최적화된 표준화**  
통합로그, 통합모니터링, 배포 자동화, 소스형상관리, 환경구성 표준화 등
- PaaS**  
개발팀에서 스스로 시스템 S/W 를 설치/구성, 개발에만 집중
- MSA**  
서비스 변경이 필요하면 언제든지 배포 (1일 10회 이상)
- 운영자동화**  
사용자가 폭증하더라도 인력에 의존하지 않고 안정성 있는 서비스

컨테이너 기술의 차이점

인프라 운영의 변화

# 가상화 VS 컨테이너 비교 - 비용적인 측면

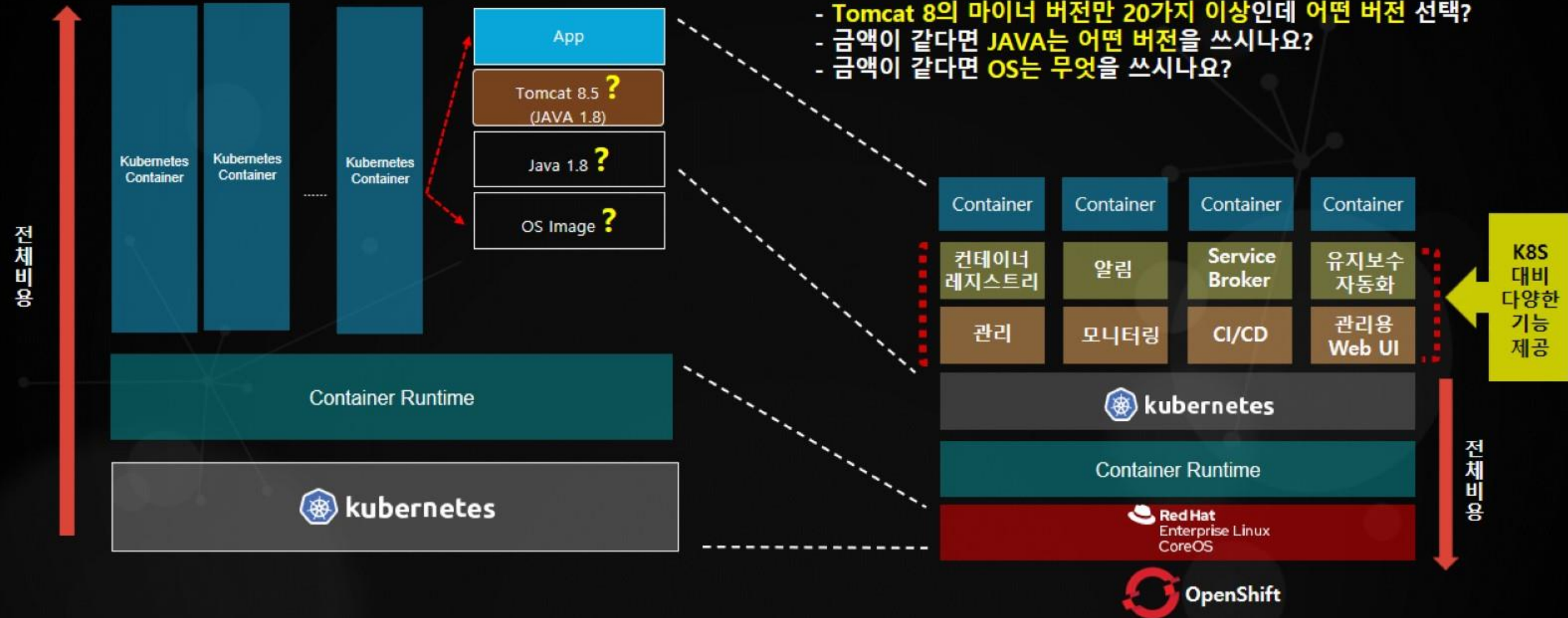
- 가상화 대비 Guest OS 유지보수, 라이선스, 관리비용 제거
- 서버 접근제어를 비롯한 보안 솔루션 제거



# 운영 환경에 부족한 Kubernetes vs 완벽한 OpenShift

**Openshift**는 Tomcat/JAVA/OS 이외에도 많은 오픈소스 유지보수 제품들이 포함되어 있습니다.

- Tomcat 8의 마이너 버전만 20가지 이상인데 어떤 버전 선택?
- 금액이 같다면 JAVA는 어떤 버전을 쓰시나요?
- 금액이 같다면 OS는 무엇을 쓰시나요?



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





# 오픈 소스 S/W 의 Maker 와 Taker

| 구분       | Maker  | Taker   |
|----------|--|---|
| 제품 개발 방법 | <ul style="list-style-type: none"> <li>• 오픈소스로 만들고 책임도 지겠습니다.</li> </ul>   | <ul style="list-style-type: none"> <li>• 만들지는 않았지만 오픈소스라 동일한 제품입니다.</li> </ul>                                    |
| 비즈니스 형태  | <ul style="list-style-type: none"> <li>• 오픈소스 커뮤니티를 리딩하며, Full Time 엔지니어를 채용하여 오픈소스 S/W 제품을 개발</li> </ul>                                  | <ul style="list-style-type: none"> <li>• 오픈소스 S/W 를 가져와 단순 설치나 기술지원</li> </ul>                                    |
| 판매 방식    | <ul style="list-style-type: none"> <li>• 오픈소스는 소유가 불가능 -&gt; 서브스크립션</li> </ul>   | <ul style="list-style-type: none"> <li>• 인력 중심의 인건비</li> </ul>  |
| 주요 제품    | <ul style="list-style-type: none"> <li>• Linux, JBoss, OpenShift, OpenStack 등</li> </ul>   | <ul style="list-style-type: none"> <li>• CentOS, Apache , Tomcat , Redis , Kubernetes</li> </ul>                  |
| 조직       | <ul style="list-style-type: none"> <li>• SW 제품 개발과 유지를 위한 전체 조직 체계 운영</li> <li>• Product PM, Committer, QA, Engineer, Support 등</li> </ul> | <ul style="list-style-type: none"> <li>• 설치 중심의 기술지원 인력</li> </ul>  |
| 기술지원     | <ul style="list-style-type: none"> <li>• 미션크리티컬 부분 - 성능, 보안, 안정성 (엔진 중심)</li> <li>• 장기 제품 라이프사이클 지원</li> </ul>                             | <ul style="list-style-type: none"> <li>• 설치/구성 과 UI 커스터마이징 중심</li> <li>• 성능, 보안, 안정성 그리고 제품에 대한 책임은 한계</li> </ul> |
| 라이선스 책임  | <ul style="list-style-type: none"> <li>• 도입 제품에 대한 오픈소스 라이선스 책임</li> </ul>   | <ul style="list-style-type: none"> <li>• 없음</li> </ul>  |
| 로드맵      | <ul style="list-style-type: none"> <li>• 제품의 로드맵 및 기능 개선/ 검토</li> </ul>  | <ul style="list-style-type: none"> <li>• 제품 로드맵에 참여하기 어려움</li> </ul>  |
| 개발자      | <ul style="list-style-type: none"> <li>• 컨설팅, 교육, 파트너 체계</li> </ul>  | <ul style="list-style-type: none"> <li>• 설치/구성 엔지니어</li> </ul>  |
| Vendor   | <ul style="list-style-type: none"> <li>• Red Hat , Enterprise DB</li> </ul>  | <ul style="list-style-type: none"> <li>• 단순 오픈소스 기술 지원</li> </ul>   |

Application Performance Management

감사합니다.



openmaru  
APM



openmaru