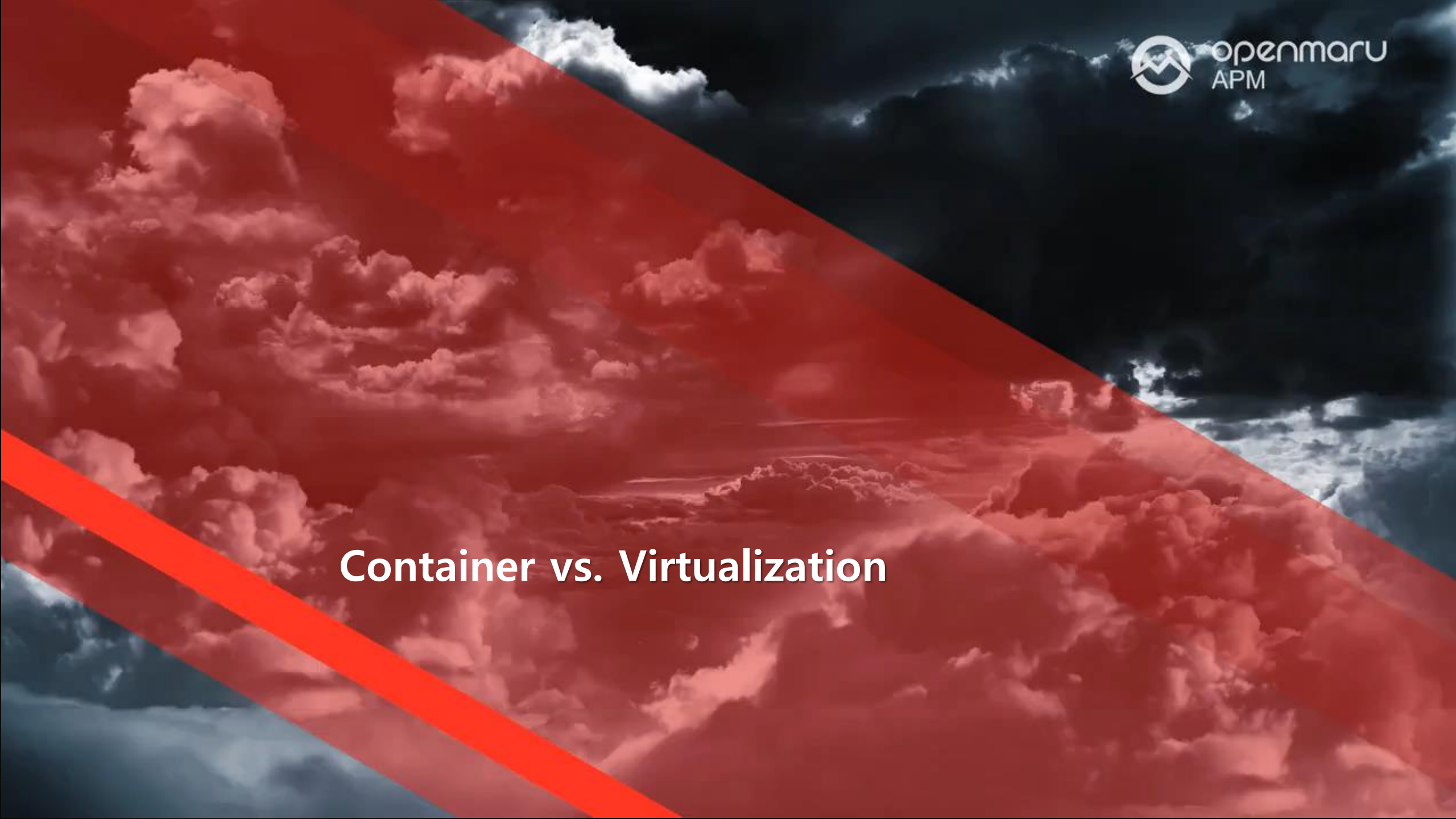
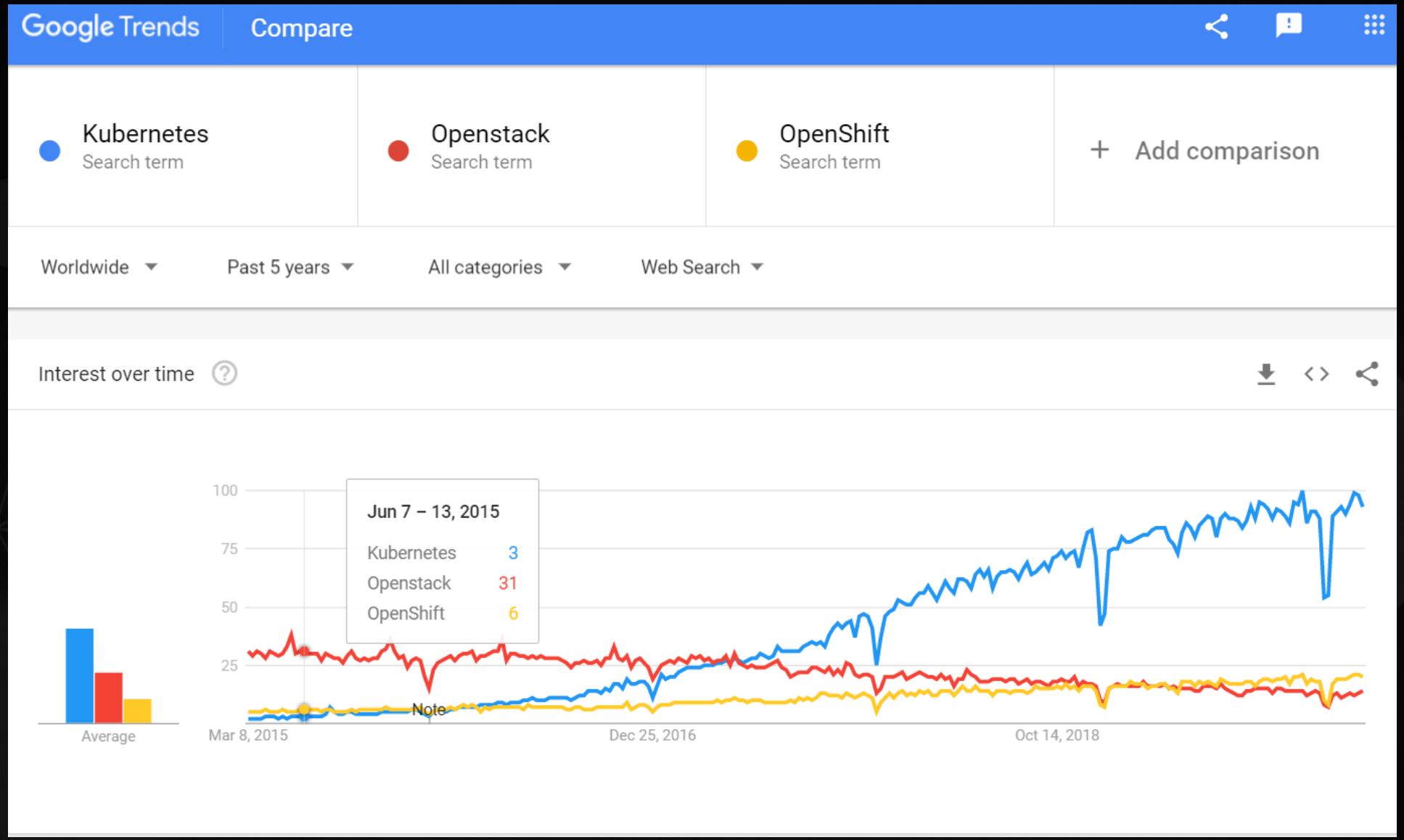

openmaru
APM
Container vs. Virtualization

가상화와 컨테이너의 차이?

Google Trends –
Kubernetes vs. OpenStack vs. OpenShift



컨테이너가 도입되기
전에는
제품별로 배에 올리고
내려야만 했습니다.

Jasmine Kim

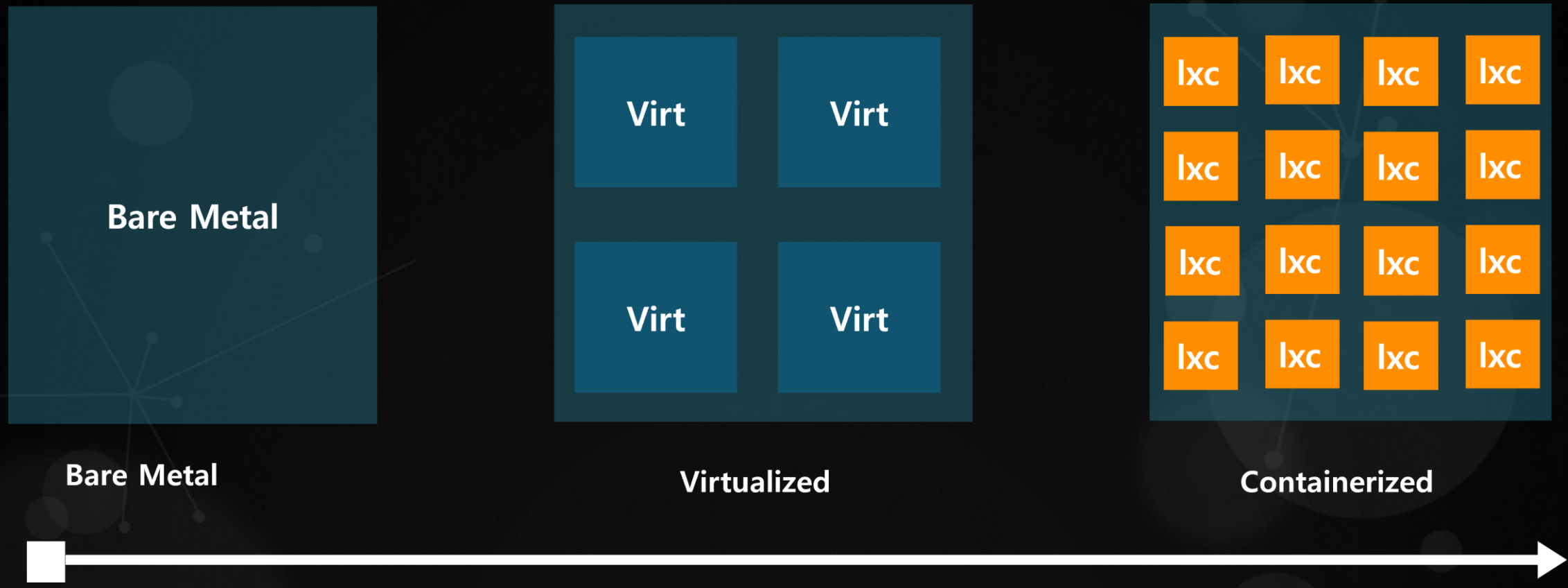
ad of Container Department


After Containerization
openmaru
APM
배에 컨테이너를 싣고
containerization

가상화 기술과 컨테이너 기술의 차이점

Container vs. Virtualization

Evolution of Infrastructure Architectures



openmaru

가상 머신

1개 애플리케이션을
위한 유휴 자원

!

VM VM VM VM

컨테이너

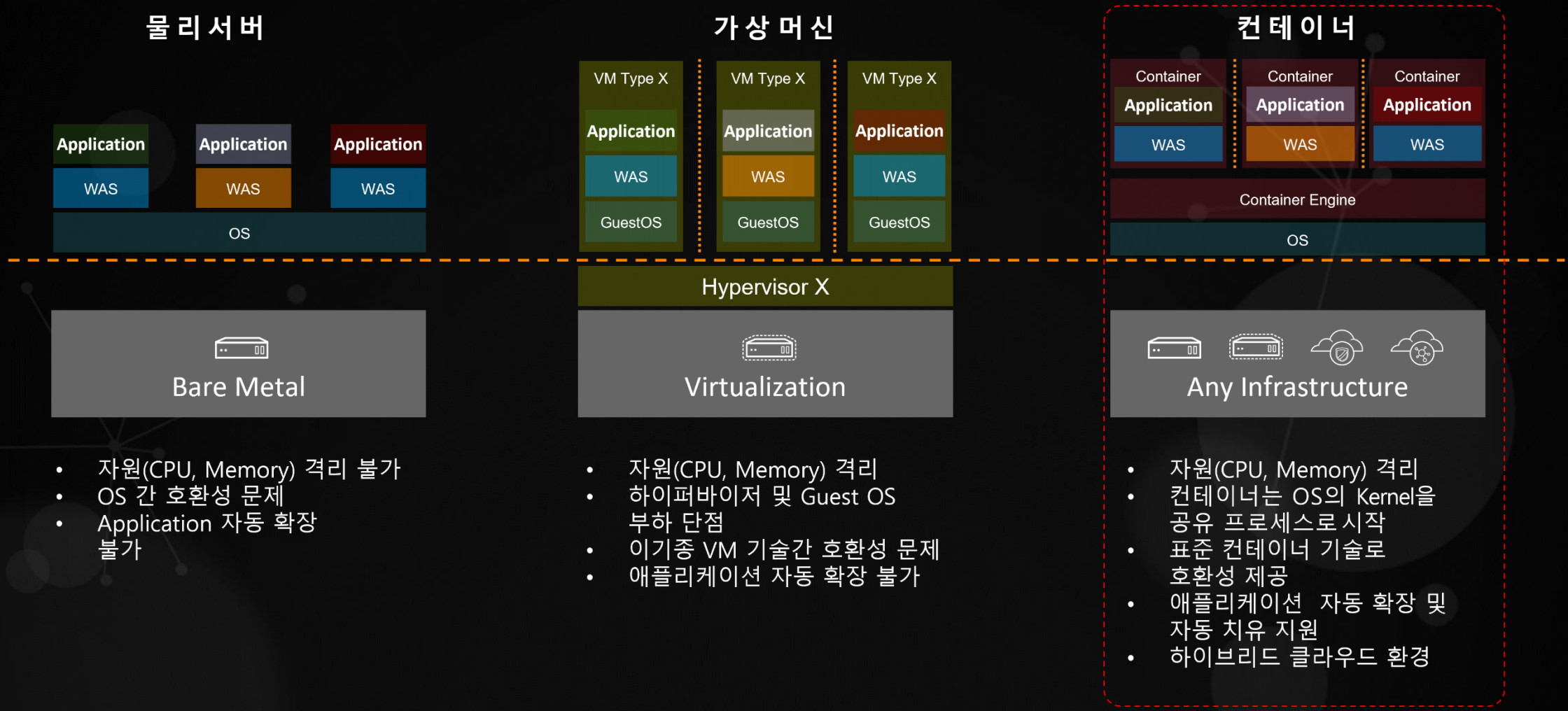
여유 자원

C C C
C C C C
C C C C

=

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

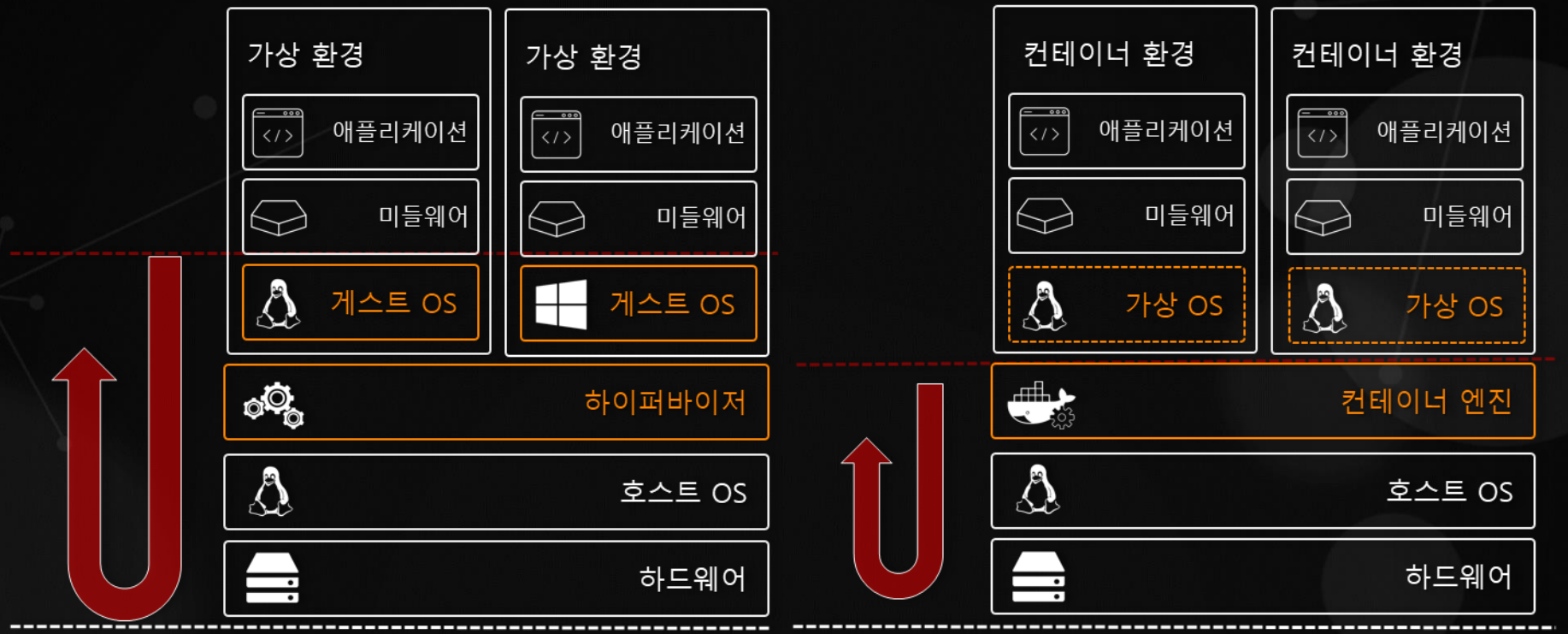
- 자원(CPU, Memory) 격리
- 컨테이너는 OS의 Kernel을 공유 프로세스로 시작
- 표준 컨테이너 기술로 호환성 제공
- 애플리케이션 자동 확장 및 자동 치유 지원
- 하이브리드 클라우드 환경

시작 시간 - Containers vs. VMs

- 하드웨어 가상화는 CPU, 메모리, 하드 디스크 등의 하드웨어를 가상화하고 있기 때문에 하드웨어 나 OS 부팅해야 부팅에 **분 단위 시간 소요**
- 컨테이너 형 가상화에서는 컨테이너 부팅 시 OS는 이미 시작하고 프로세스의 시작 만 할 **초 단위 시간**

오버헤드 - Containers vs. VMs

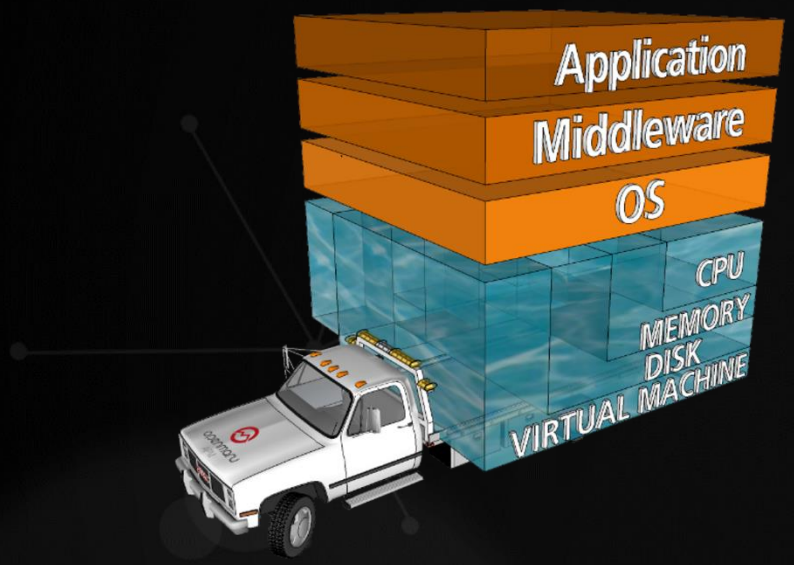
- OS에서 응용 프로그램을 작동하는 경우, 하드웨어 가상화에서는 **가상화 된 하드웨어 및 하이퍼바이저**를 통해 처리하기 때문에 물리적 시스템보다 처리에 **부가적인 시간 (오버 헤드)가 필요**
- 컨테이너 형 가상화 커널을 공유하고 **개별 프로세스가 작업**을 하는 것과 같은 정도의 시간 밖에 걸리지 않기 때문에 대부분 **오버 헤드가 없음**



VM 의 고질적인 문제점

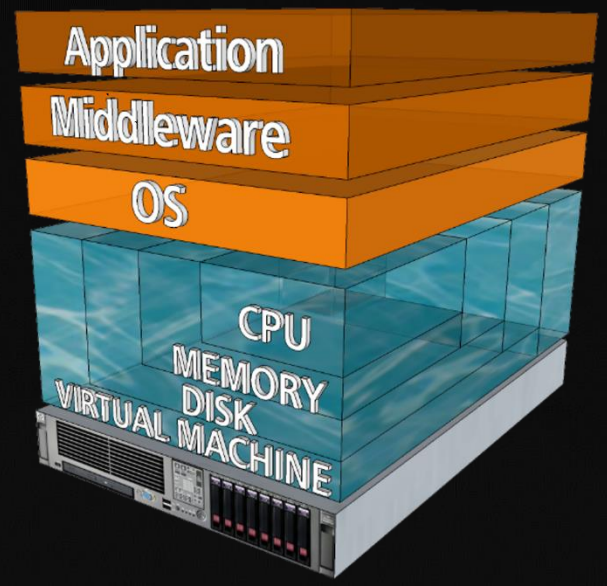
거대한 이미지 사이즈

- VM 을 템플릿 관리는 하지만 사이즈가 커서 재사용성을 높이기 어려움



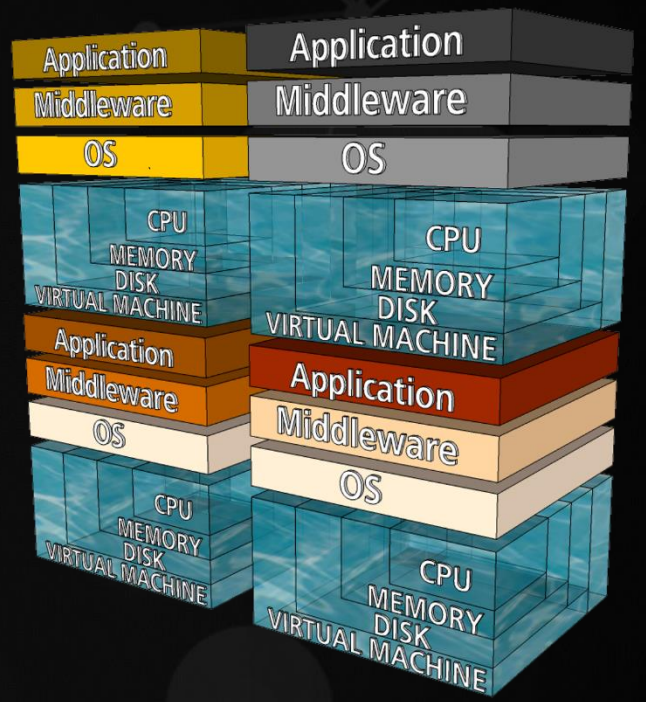
느린 시작 시간

- 부팅 시 Hypervisor – OS- 미들웨어 – 애플리케이션까지 실행 되어야함



VM 간의 환경 불일치

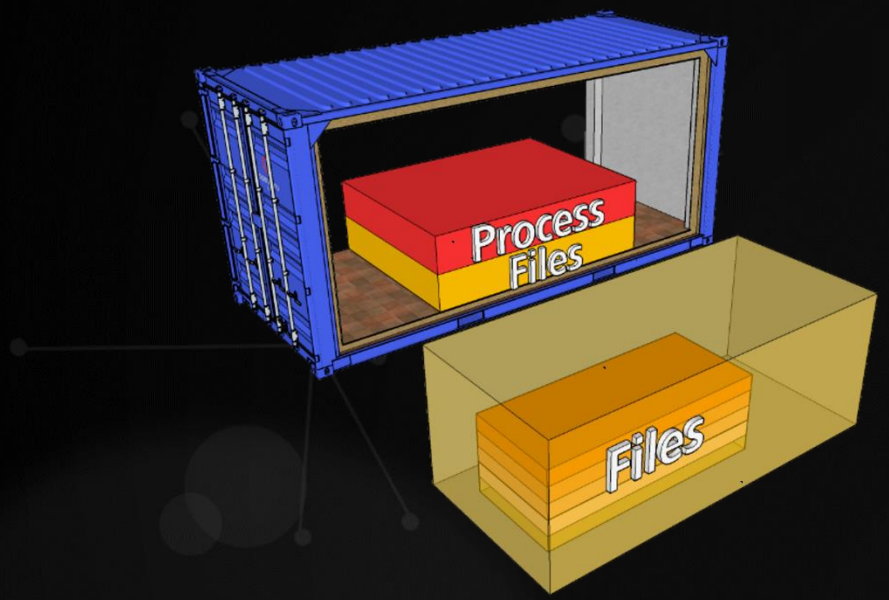
- VM 생성 후 개별로 변경 사항을 관리하기 때문에 VM 간 구성이나 환경이 불일치



컨테이너를 통한 VM 문제 해결

작은 이미지 사이즈

- 컨테이너는 레이어 개념으로 이미지에 파일을 추가/삭제하여 관리함
- 레이어 사이즈를 최적화하여 이미지 사이즈를 최소화



빠른 시작 시간

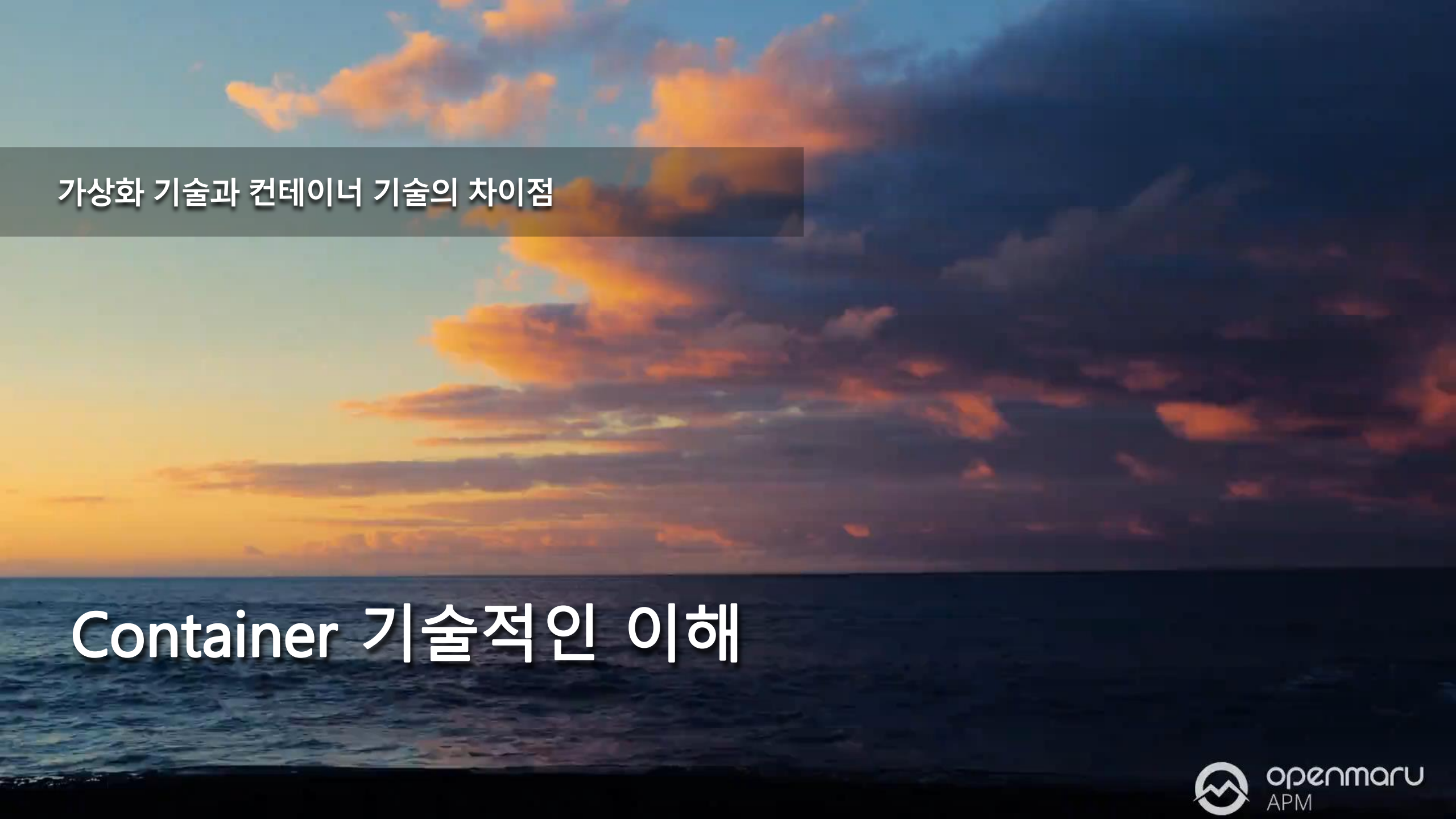
- 컨테이너는 분리된 프로세스 형식으로 OS 부팅이 필요 없기 때문에 부팅 시간을 최소화 할 수 있음

Container = Process

높은 이동성(Portability)

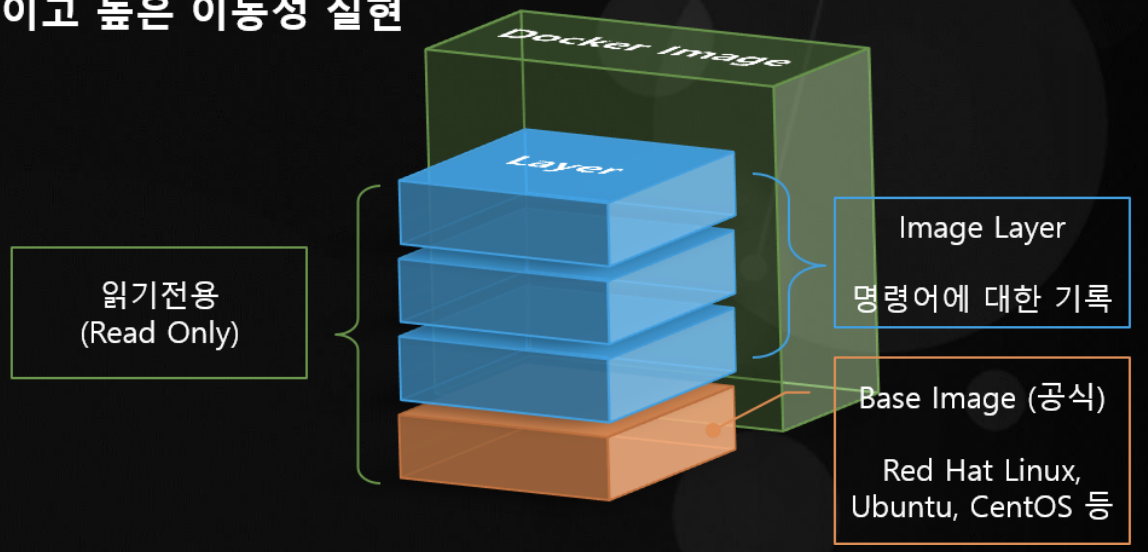
- 애플리케이션에 필요한 라이브러리나 의존 파일들을 이미지에 포함하기 때문에 환경에 의한 발행 되는 문제가 거의 없음

가상화와 컨테이너 비교 결과


가상화 기술과 컨테이너 기술의 차이점
Container 기술적인 이해
openmaru
APM

Container Image

- 컨테이너를 실행할 때 필요한 파일시스템
  - 이미지 레이어의 집합체
    - 파일 내용과 메타 정보를 포함
  - 레이어는 부모와 자식 관계
  - 변경분만 기록
  - Read Only (읽기 전용) 으로 쓸 수 없음
- 공통 레이어를 이미지 간에 공유
  - 디스크 용량을 줄이고 높은 이동성 실현

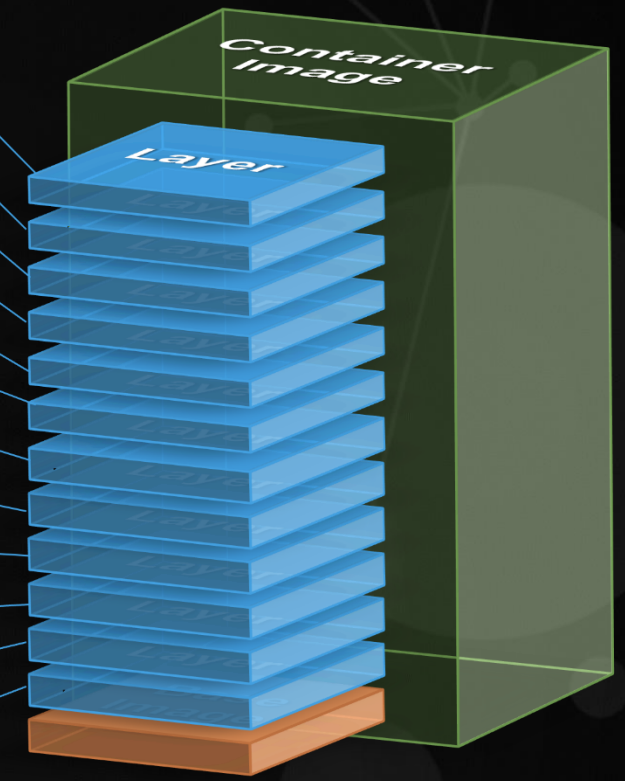


Container 이미지 구조 예시

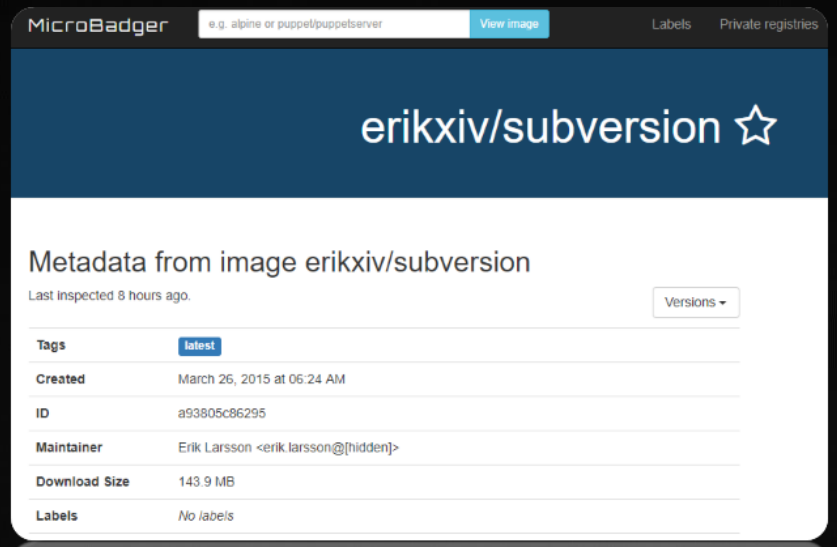
Container Image 구조

- **docker history <image id / name>**

	CREATED	CREATED BY	SIZE
a486da044a3f	10 weeks ago	/bin/sh -c # (nop) CMD ["nginx" "-g" "daemon o	0 B
3cb7f49c6bc4	10 weeks ago	/bin/sh -c # (nop) EXPOSE 443 /tcp 80 /tcp	0 B
42d2189f6cbe	10 weeks ago	/bin/sh -c # (nop) VOLUME [/var/cache/nginx]	0 B
6dda3f3a8c05	10 weeks ago	/bin/sh -c ln -sf /dev/stderr /var/log/nginx/	11 B
9108e25be489	10 weeks ago	/bin/sh -c ln -sf /dev/stdout /var/log/nginx/	11 B
72b67c8ad0ca	10 weeks ago	/bin/sh -c apt-get update &&	7.695 MB
e7e7a55e9264	10 weeks ago	/bin/sh -c # (nop) ENV NGINX_VERSION=1.9.4-1 ~ j	0 B
97df1ddba09e	3 months ago	/bin/sh -c echo "deb http://nginx.org/package	221 B
5dd2638d10a1	3 months ago	/bin/sh -c apt-key adv --keyserver hkp : // pgp.	1.997 kB
aface2a79f55	3 months ago	/bin/sh -c # (nop) MAINTAINER NGINX Container Mai	0 B
9a61b6b1315e	3 months ago	/bin/sh -c # (nop) CMD ["/bin/bash"]	0 B
902b87aaaec9	3 months ago	/bin/sh -c # (nop) ADD file:e1dd18493a216ecd0c	125.2 MB



Container Image에 대한 Layer 정보



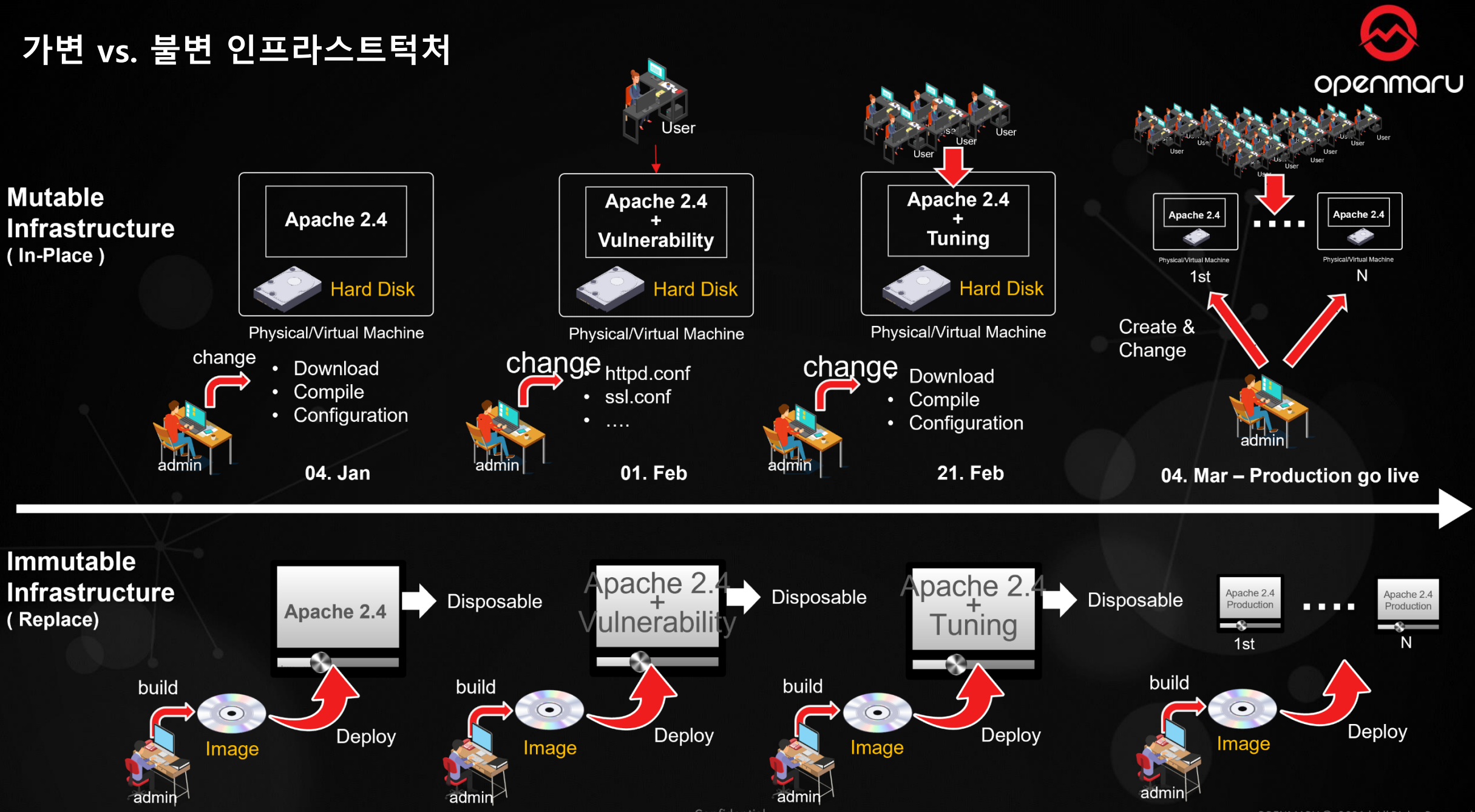
Source: https://microbadger.com/images/erikxiv/subversion

가상화 기술과 컨테이너 기술의 차이점

Immutable Infrastructure

openmaru APM

가변 vs. 불변 인프라스트럭처



컨테이너 - 애플리케이션 지향 인프라

- 컨테이너화는 데이터 센터를 머신 지향에서 애플리케이션 지향으로 전환
  - 애플리케이션 개발자와 운영팀에게 서버와 운영 체제에 대한 세부 사항을 추상화
  - 실행중인 애플리케이션과 개발자에 미치는 영향을 최소화하면서 새로운 하드웨어 지원과 운영 체제를 업그레이드하여 인프라 팀의 유연성 제공
  - 서버 CPU와 메모리 사용량과 같은 메트릭 정보 뿐만 아니라 애플리케이션에 연결하여 스케일 업, 머신 장애 또는 유지 보수 시에 애플리케이션 모니터링

Machine Centric Infrastructure

Application Centric Infrastructure

- Immutable Infrastructure 에 대한
- 자세한 내용은 Kubernetes 에서

Container Runtime
(CRI-O/ContainerD/Rkt)

GOOGLE 과 컨테이너

- **Google의 업무 방식**

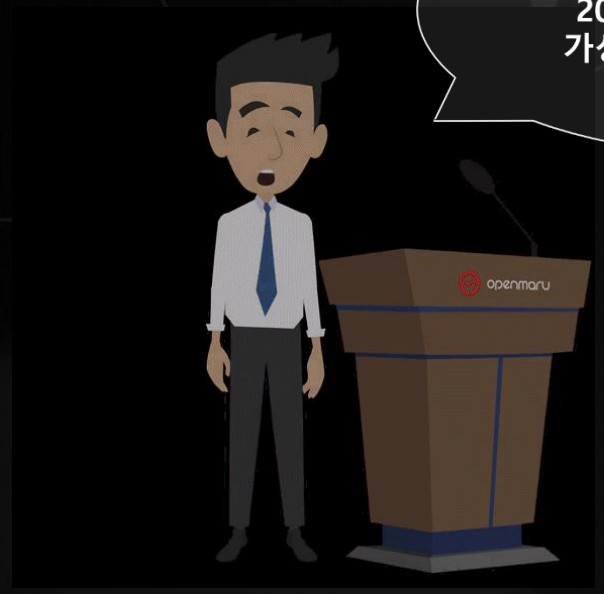
Gmail에서 YouTube, 검색에 이르기까지 Google의 모든 제품은 컨테이너에서 실행됩니다.

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/

컨테이너 기술의 역사



컨테이너의 역사는 1979 년에 chroot 로부터 시작합니다. 2000년 FreeBSD가 OS 가상화인 FreeBSD Jail을 발표합니다.

chroot → FreeBSD jail → Virtuozzo

2001 OpenVZ

2005

2008 LXC

Red Hat OpenShift Online

2011

Kubernetes

2013 docker

2014

RED HAT ENTERPRISE LINUX 7

2015 rkt

Red Hat OpenShift 3.0

OCI

2016

Moby moby project

2017

cri-o

2018

CNCF

Buildah

2019

Red Hat Enterprise Linux 8

Red Hat OpenShift 4.5

2020

skopeo

Podman

MIRANTIS 가 DOCKER ENTERPRISE 사업을 인수

- 미국 시간 2019년 11월 13일 Mirantis 는 Container 의 Container Enterprise 사업을 인수 발표
- 컨테이너 기술인 Container 와 그것의 기업용 상용 버전을 제공하는 미국의 Container 사는 Container Enterprise Platform 사업을 Mirantis 에 매각하기로 합의

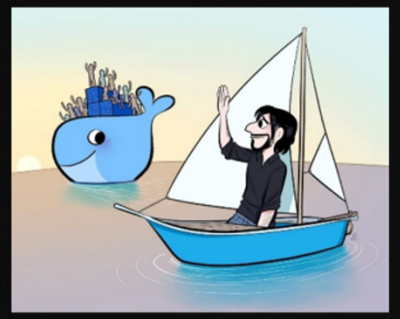


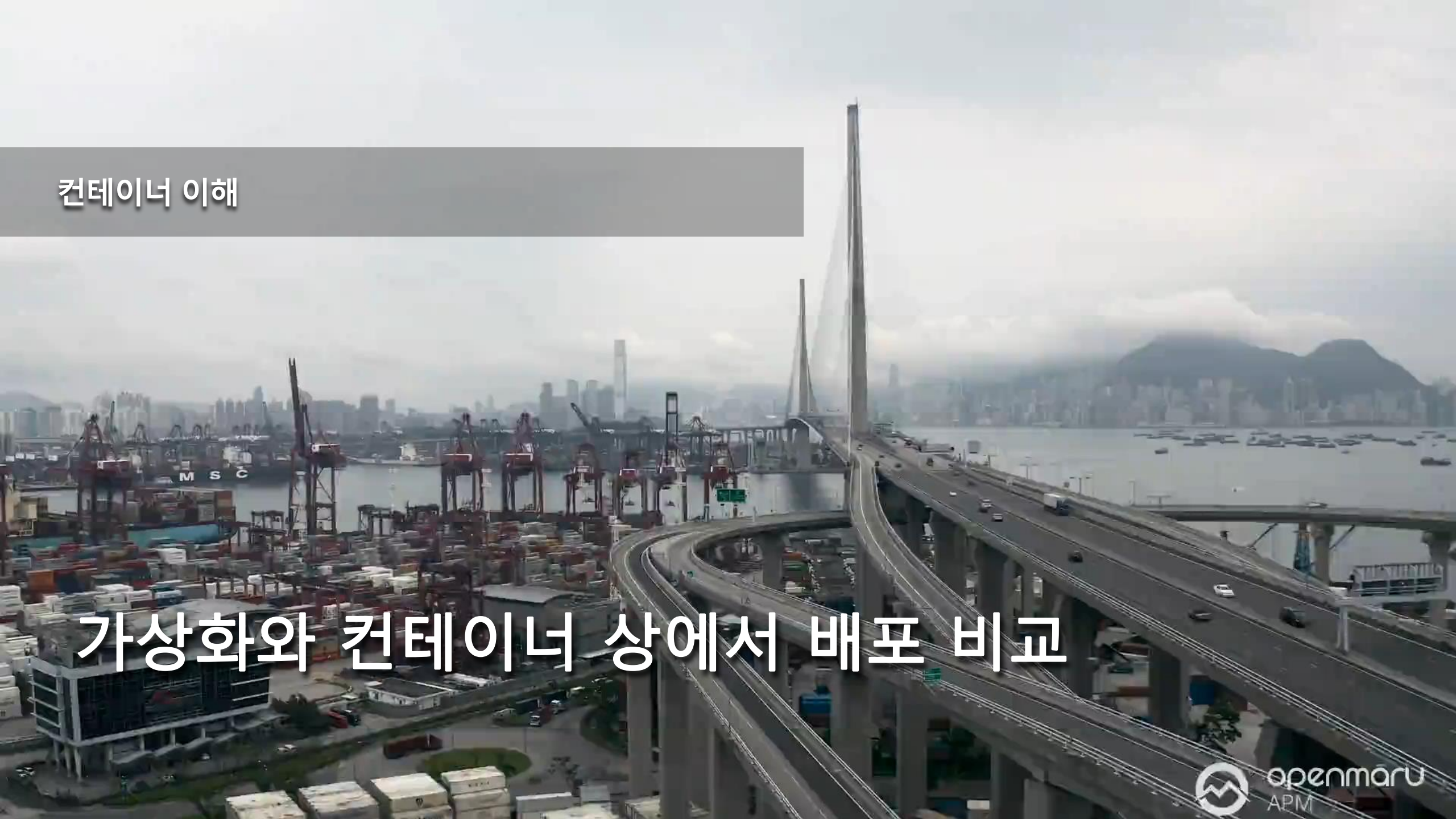
Image Source : https://www.docker.com/blog/au-revoir/

- Mirantis 의 직원은 450명 정도이며, 이번 인수를 통해 Docker 지원 300 명 정도가 조직에 새롭게 통합
- Docker 의 공동창업자 중 한명인 솔로몬 하이크스 최고기술책임자 (CTO) 는 2019년 11월 28일 회사 블로그를 통해 퇴사를 밝혔습니다.

KUBERNETES 1.20 부터 DOCKER 사용을 중단

- Kubernetes는 2020년 12월 v1.20 이후 Docker 지원 중단을 발표
- CRI-O, containerd를 지원할 수 있도록 설계되어 있음
- Dockerfile은 그대로 사용가능 함
  - Dockerfile은 OCI(Open Container Initiative)에서 표준화된 파일
  - OCI 표준 이미지는 CRI-O, containerd에서 동일하게 사용 가능

Source - https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/



가상화와 컨테이너 상에서 배포 비교

openmaru
APM

개발환경

application

Middleware

Windows

PC

테스트환경

application

Middleware

Linux

X86

운영환경

application

Middleware

Unix

IBM

개발자

운영자

openmaru APM

개발 환경

컨테이너

application

Middleware

Container Engine

Linux (Host)

Hardware

테스트 환경

컨테이너

application

Middleware

Container Engine

Linux (Host)

Hardware

컨테이너 이미지

운영 환경

컨테이너

application

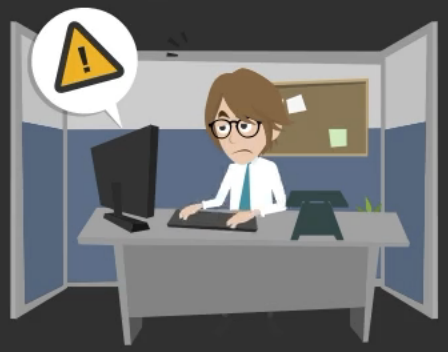
Middleware

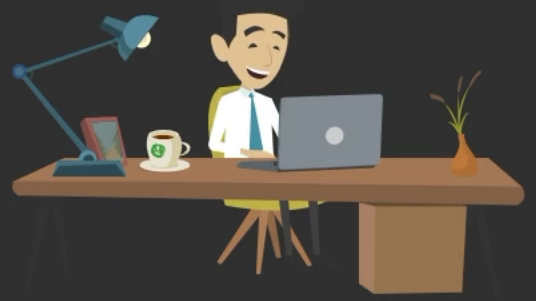
Container Engine

Linux (Host)

Hardware

모두 같은
컨테이너
에서 동작합니다.






openmaru