
opennaru
Web System based on Docker
KHAN
apm

opennaru

Application Performance Management

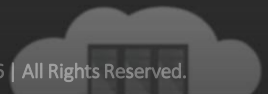
Change brings opportunity.

IT Evolution

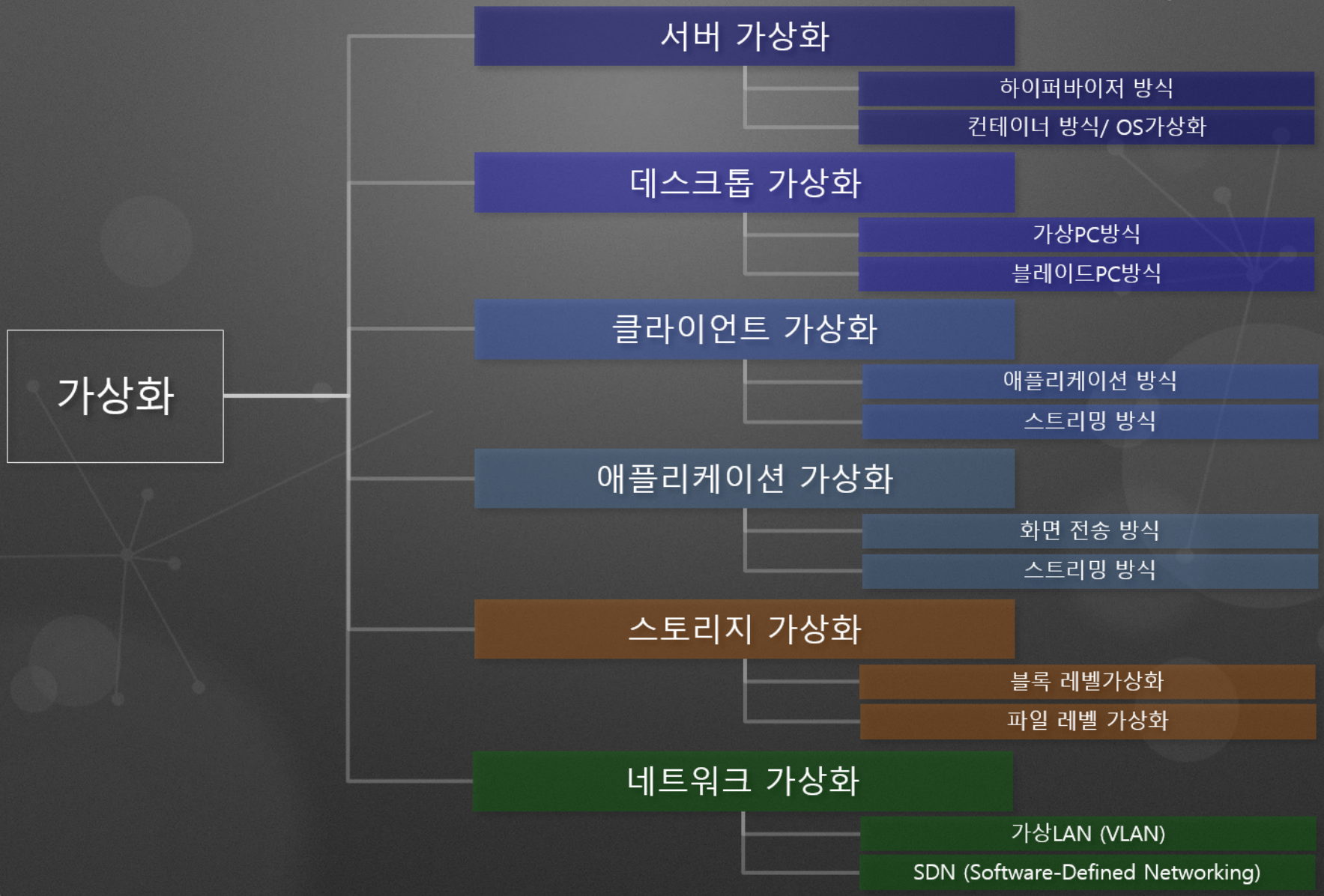
opennaru

KHAN [apm]

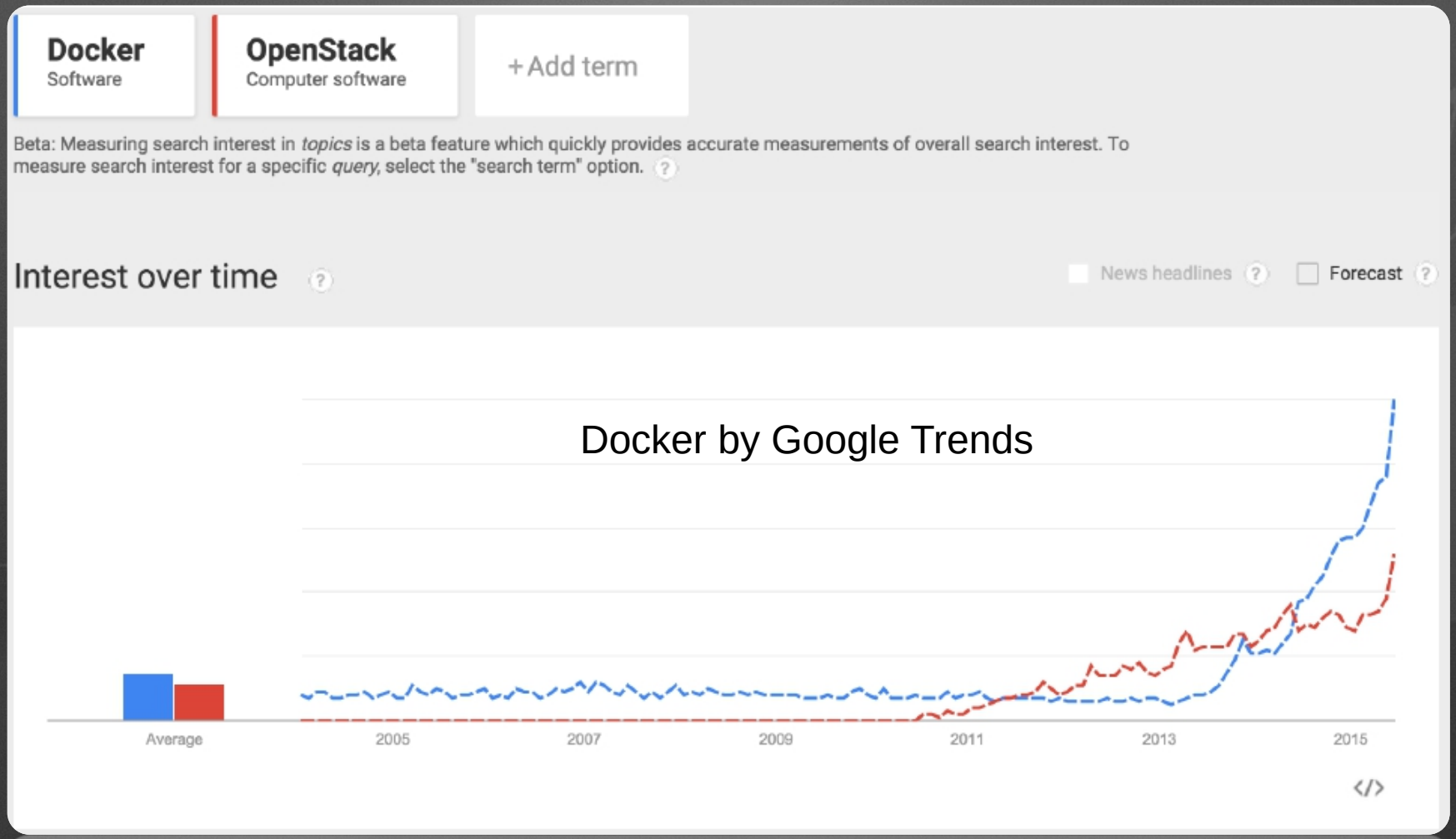
Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
WATERFALL	MONOLITHIC	PHYSICAL SERVERS	DATA CENTER
AGILE	N-TIER	VIRTUAL SERVERS	HOSTED
DEVOPS	MICROSERVICES	CONTAINERS	CLOUD



시스템 컴포넌트 별 가상화

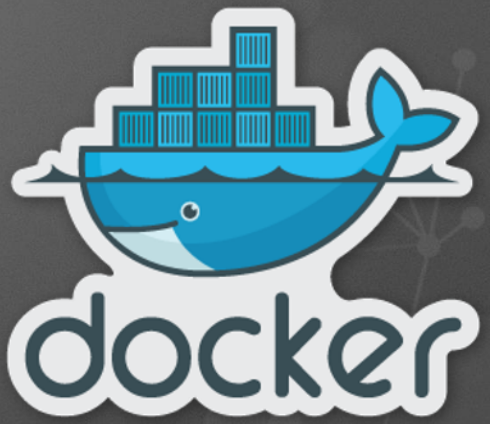


Docker by Google Trends



Container 시대를 향하여

opennaru

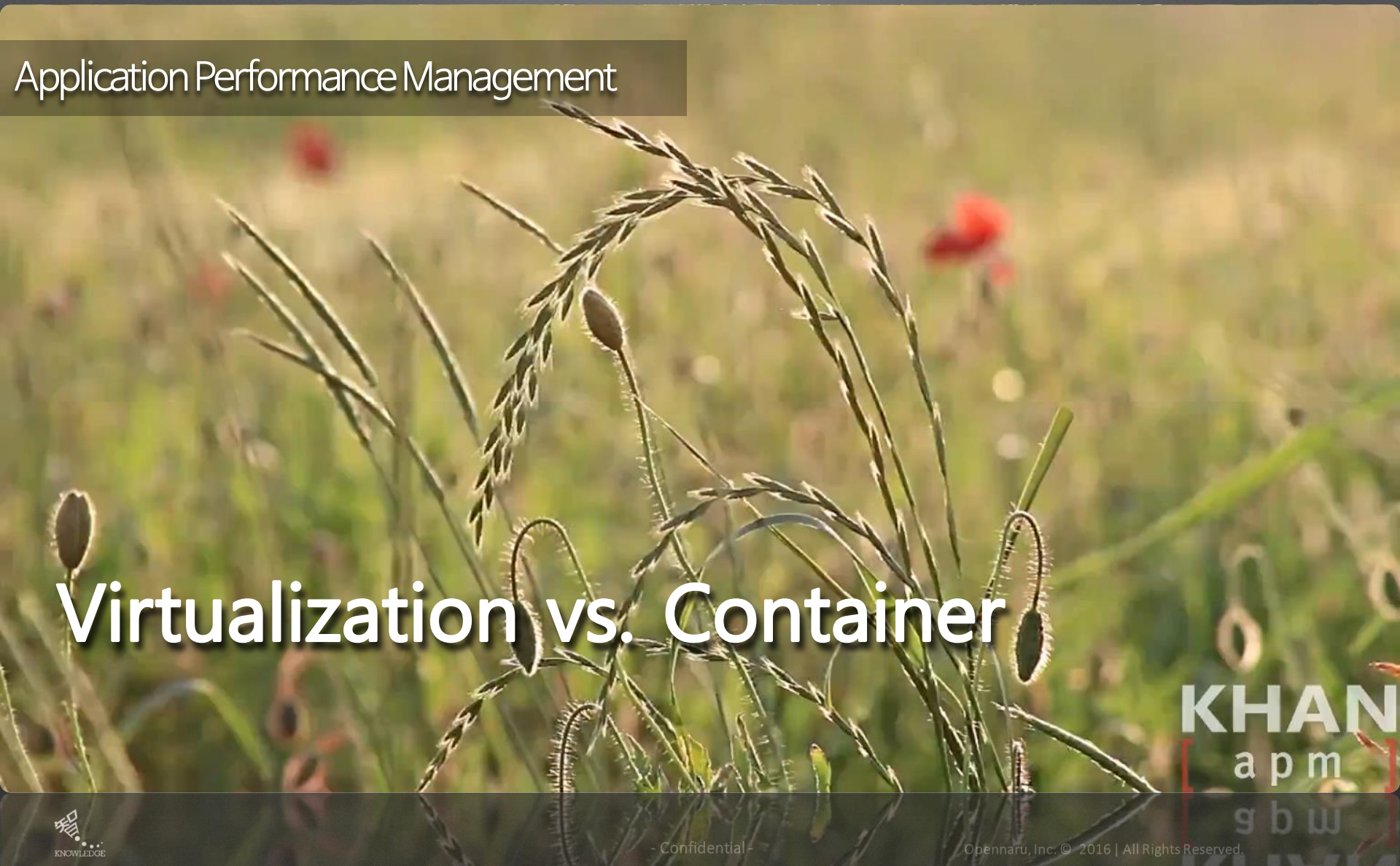


1. 컨테이너란 무엇인가? 가상화와의 차이는 무엇인가?
   - ☐ 기존의 가상화와 컨테이너는 무엇이 다른지?
   - ☐ 컨테이너에는 어떤 구조로 되어 있는 건가?
   - ☐ Linux OS만 있으면 컨테이너를 바로 사용하나?
2. 컨테이너 적용시의 기대효과?
   - ☐ 컨테이너를 사용하면 무엇이 좋은 것인가?
   - ☐ 컨테이너는 무조건 좋은 건가? 단점은 무엇인가?
   - ☐ 컨테이너를 적용하기에 좋은 시스템과 그렇지 않은 시스템은?
3. 컨테이너를 운영 환경에서 사용하려면? 최근 화제인 컨테이너 플랫폼이란?
   - ☐ 컨테이너기반 시스템을 구축 하는 방법은?
   - ☐ 컨테이너는 실제 운영환경에서 사용할 만큼 성숙된 기술인가?
   - ☐ 컨테이너 플랫폼은 무엇이 있으며 향후 발전 방향은?

opennaru

Application Performance Management

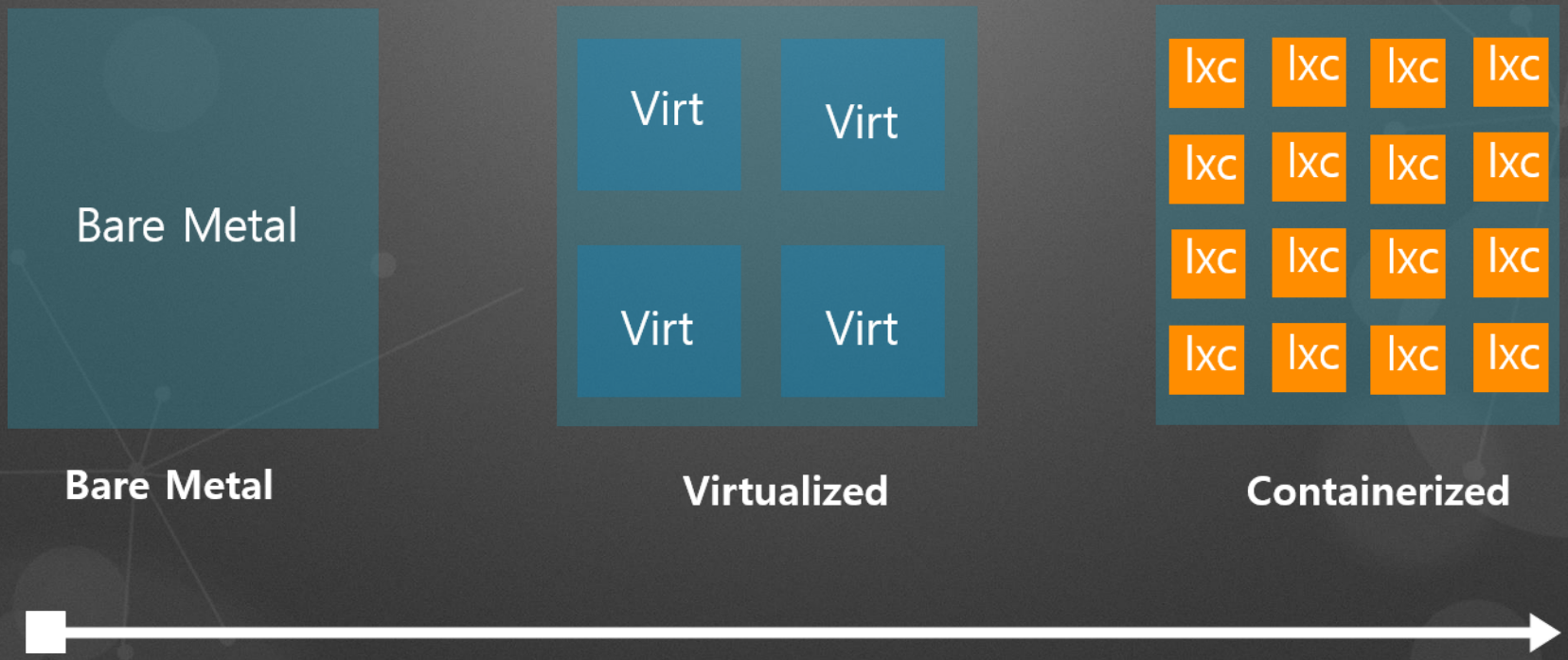
Virtualization vs. Container



KHAN
apm
gbw

Evolution of Infrastructure Architectures

opennaru



하드웨어 가상화와 OS 가상화 비교

opennaru

- 서버용 가상화 기술로 VMWare, RHV 나 KVM등의 가상머신모니터(VMM)/하이퍼바이저에 의한 "하드웨어 가상화"
- 하드웨어와 리소스를 가상머신에 할당하기 위해서는 하드웨어 전체가 가상화되어야 함

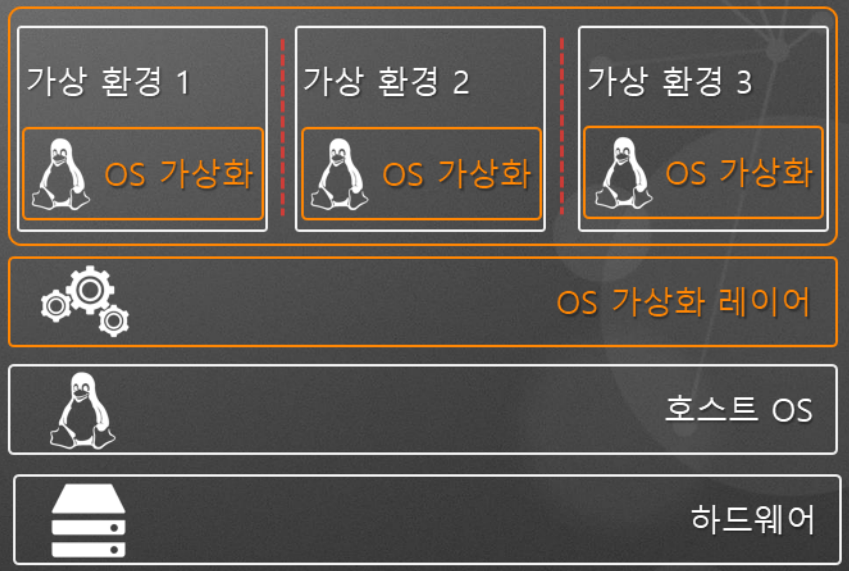
하이퍼바이저에 의한 하드웨어 가상화

호스트 OS와 게스트OS가 다를 경우도 가능

디스크/메모리 소비 큼

구성 자유도가 높은 가상화 기술

- 가상화라기보다는 격리된 가상적인 OS환경을 제공



OS 가상화

호스트OS와 게스트OS 동일

디스크/메모리 소비 작음

가볍고 휴대 성이 높은 가상화 기술

컨테이너 (OS 가상화) 비교

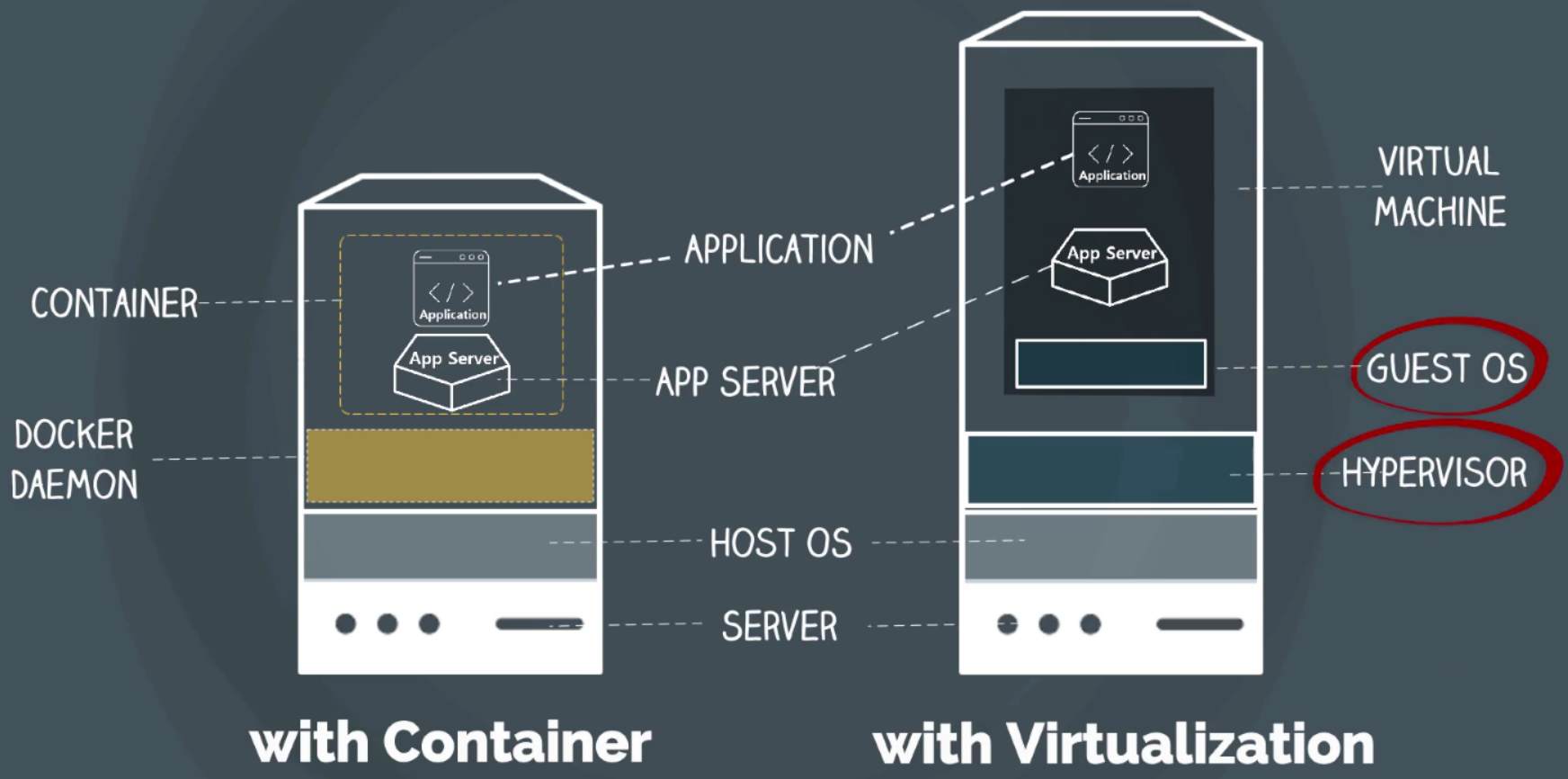
opennaru

	LXD	Docker	LXC	OpenVZ	Virtuozzo
Release	2015	2013	2008	2005	2001
개발사	Canonical	Docker Inc.	IBM, Parallels, Canonical	OpenVZ Community	Parallels
지원OS	Linux	Linux, Windows, MacOS	Linux	Linux	Linux, Windows
가격	무상	무상	무상	무상	유상
형태	오픈소스	오픈소스	오픈소스	오픈소스	독점

KNOWLEDGE

Container Vs. Virtualization

opennaru



Containers vs. VMs

opennaru

VM 난립(sprawl) 현상과 오버헤드

<하이퍼바이저에 의한 하드웨어 가상화>

- 가상머신은 쉽게 만들 수 있기 때문에 사용하지 않는 가상머신들이 무질서하게 증가하여 VM 난립 현상 발생
- 중복 된 OS는 하드 디스크 공간을 낭비하고, 라이선스와 관리를 개별적으로 수행
- 애플리케이션을 운영하기 위해 여러 OS (게스트 OS – 하이퍼바이저 - 호스트 OS)를 거치기 때문에 더 많은 프로세스 파워를 소비하여 오버 헤드의 증대

Docker vs. 가상화

opennaru

	컨테이너 형 가상화 (Docker)	하이퍼 바이저 형 가상화 (VMWare ESXi)	호스트 형 가상화 (Linux KVM)
가상 머신	OS를 호스트 OS와 공유하기 때문에 VM 마다 OS 설치를 할 필요는 없다	VM마다 OS 설치	VM마다 OS 설치
지원 OS	• Linux • Windows	• Windows, Linux • 일부 Unix도 지원	• Windows, Linux • 일부 Unix도 지원
부팅 시간	OS 설치 불필요하기 때문에 사용 시작까지의 시간이 매우 짧음	초기 구축 시에는 네트워크 OS 설치 등의 작업이 발생하기 때문에 이용 개시까지의 시간이 많이 소요	초기 구축 시에는 네트워크 OS 설치 등의 작업이 발생하기 때문에 이용 개시까지의 시간이 많이 소요
네트워크	호스트 측에 작성된 Docker 전용 NIC와 통신	• 네트워크의 생성이 가능 • VM에 임의의 숫자 vNIC를 부여 가능	• 네트워크의 생성이 가능 • VM에 임의의 숫자 vNIC를 부여 가능
자원	표준에서는 HDD 자원을 지정할 수 없다. CPU, 메모리에 대한 자원 할당 지정 가능	CPU, 메모리, HDD의 자원 할당을 지정	CPU, 메모리, HDD의 자원 할당을 지정
오버 헤드	컨테이너는 호스트 OS에서 보면 하나의 프로세스이며, 오버 헤드는 거의 없음	VM에서 기기까지의 액세스 경로를 하이퍼바이저 뿐이므로 호스트 형 가상화에 비해 오버 헤드가 적은	VM에서 기기까지의 액세스 경로가 다른 가상화 기술에 비해 길기 때문에 비교했을 경우에는 가장 오버 헤드가 높음

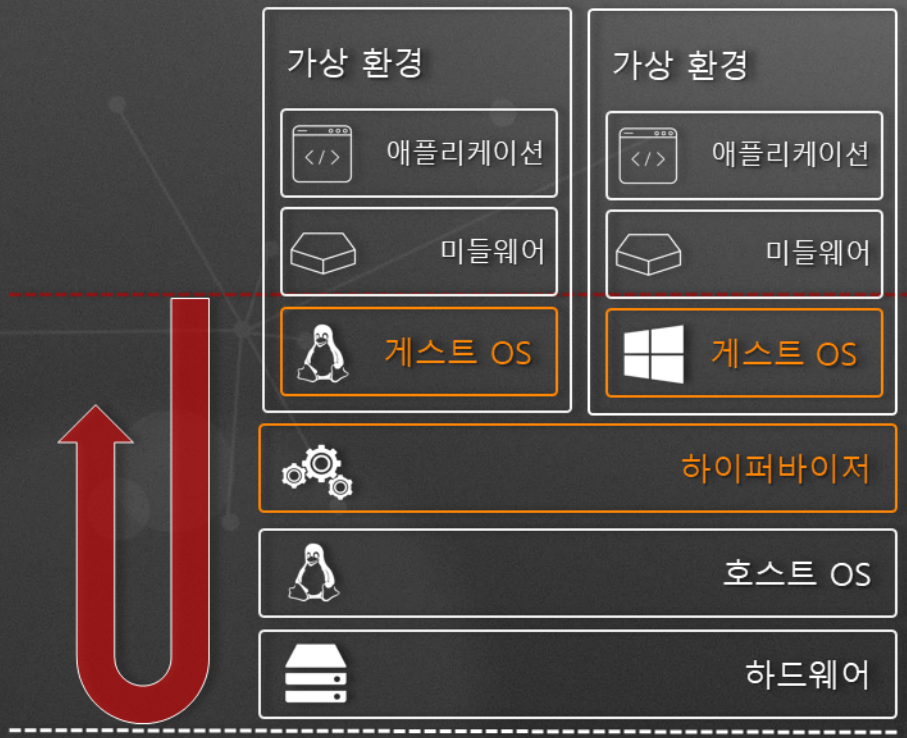
시작 시간 - Containers vs. VMs

opennaru

- 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

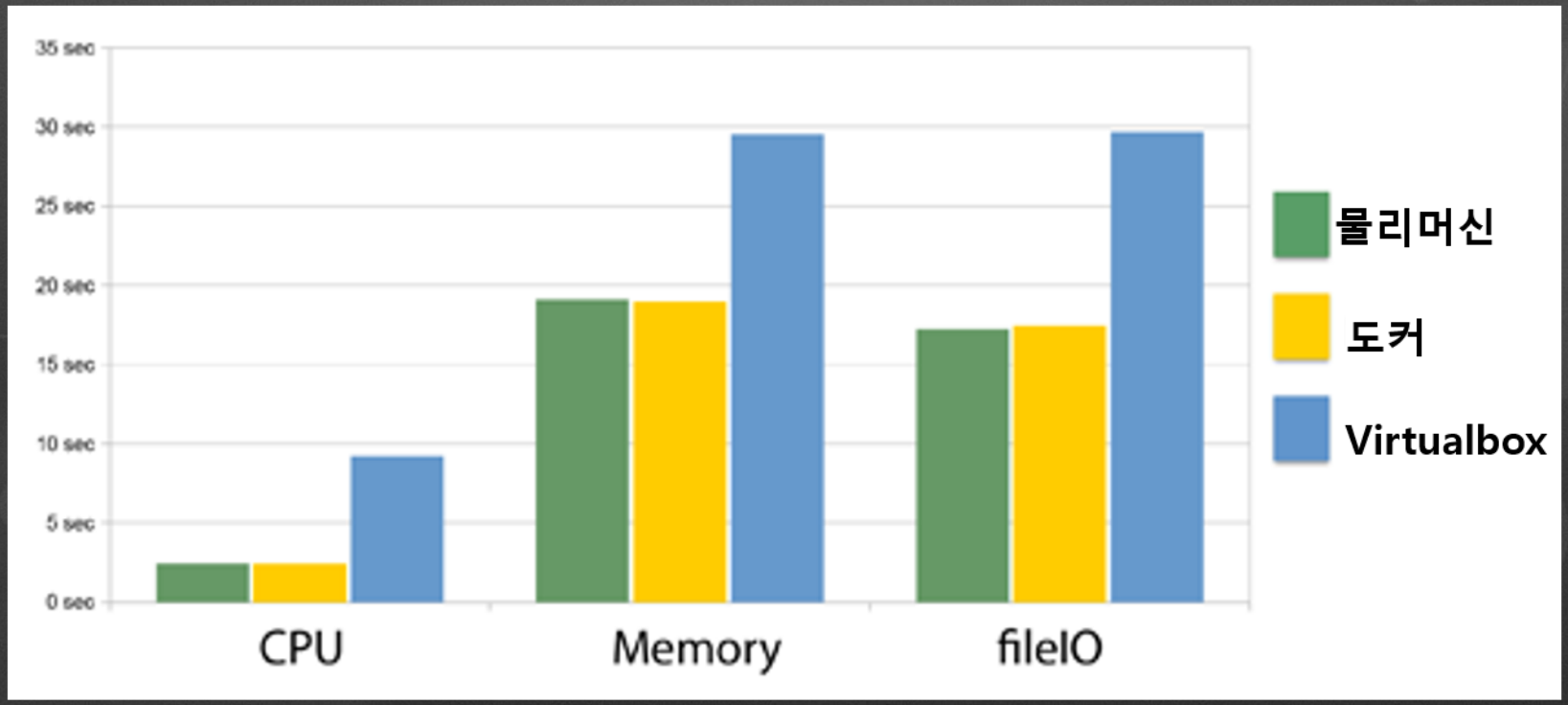
- OS에서 응용 프로그램을 작동하는 경우, 하드웨어 가상화에서는 **가상화 된 하드웨어 및 하이퍼바이저**를 통해 처리하기 때문에 물리적 시스템보다 처리에 **부가적인 시간 (오버 헤드)가 필요**
- 컨테이너 형 가상화 커널을 공유하고 **개별 프로세스가 작업**을 하는 것과 같은 정도의 시간 밖에 걸리지 않기 때문에 대부분 **오버 헤드가 없음**



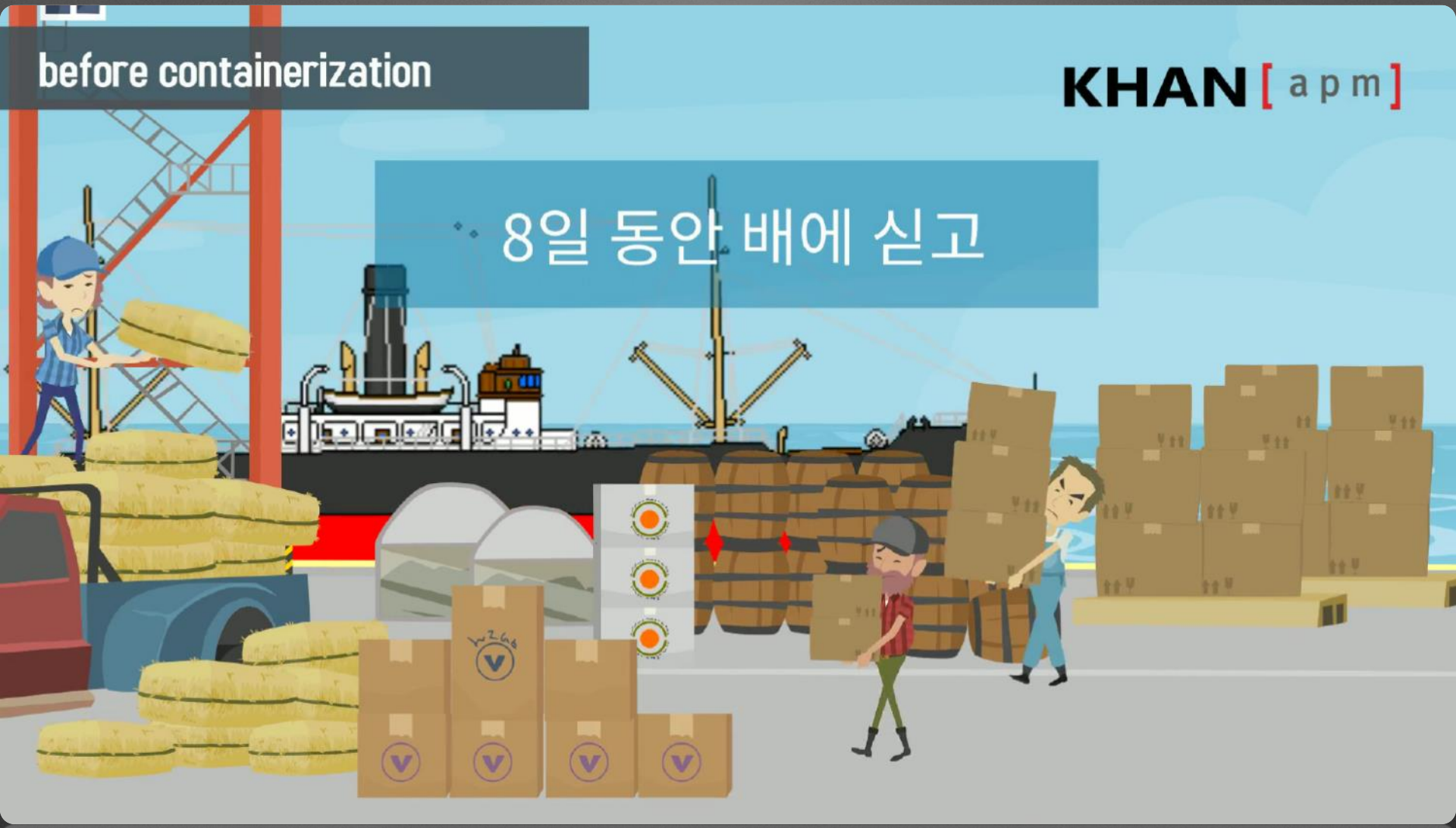
성능 - Containers vs. VMs

opennaru

- "sysbench"라는 벤치 마크 도구를 사용하여 성능 측정
- 물리적 시스템과 컨테이너 형 가상화 성능은 모든 항목에서 거의 같은 결과
- 하드웨어 가상화는 메모리, 파일 IO는 약 2 배, CPU는 약 5 배의 시간
- 물리 머신과 비교해도 성능 저하가 거의 없음



Before Container



After Container

opennaru

Application Performance Management

Container 개요

KHAN
apm
gbw

Docker 란?

opennaru

- 고래 등에 올려진 컨테이너 박스들 처럼
- 프로그램과 실행에 필요한 것들을 컨테이너에 Shipping,
- 컨테이너를 손쉽게 이동해서,
- 어디서나 간단하게 실행할 수 있는
- 도구와 환경을 제공하는 오픈 소스 플랫폼.

Build, Ship and Run
Any App, Anywhere

History of Container

opennaru

2000:
JAILS ADDED
TO FREEBSD

2003:
SELINUX ADDED TO
LINUX MAINLINE

2006:
GENERIC PROCESS
CONTAINERS

2008:
KERNEL AND USER
NAMESPACES

2013:
RED HAT ENTERPRISE
LINUX

2008:
LINUX CONTAINER
PROJECT (LXC)

2013:
DOTCLOUD BECOMES
DOCKER

2015:
RHEL ATOMIC HOST

2015:
RHT CONTAINER
PLATFORMS

2001:
LINUX -VSERVER
PROJECT

2005:
FULL RELEASE OF
SOLARIS ZONES

2007:
GPC RENAMED
CONTROL GROUPS

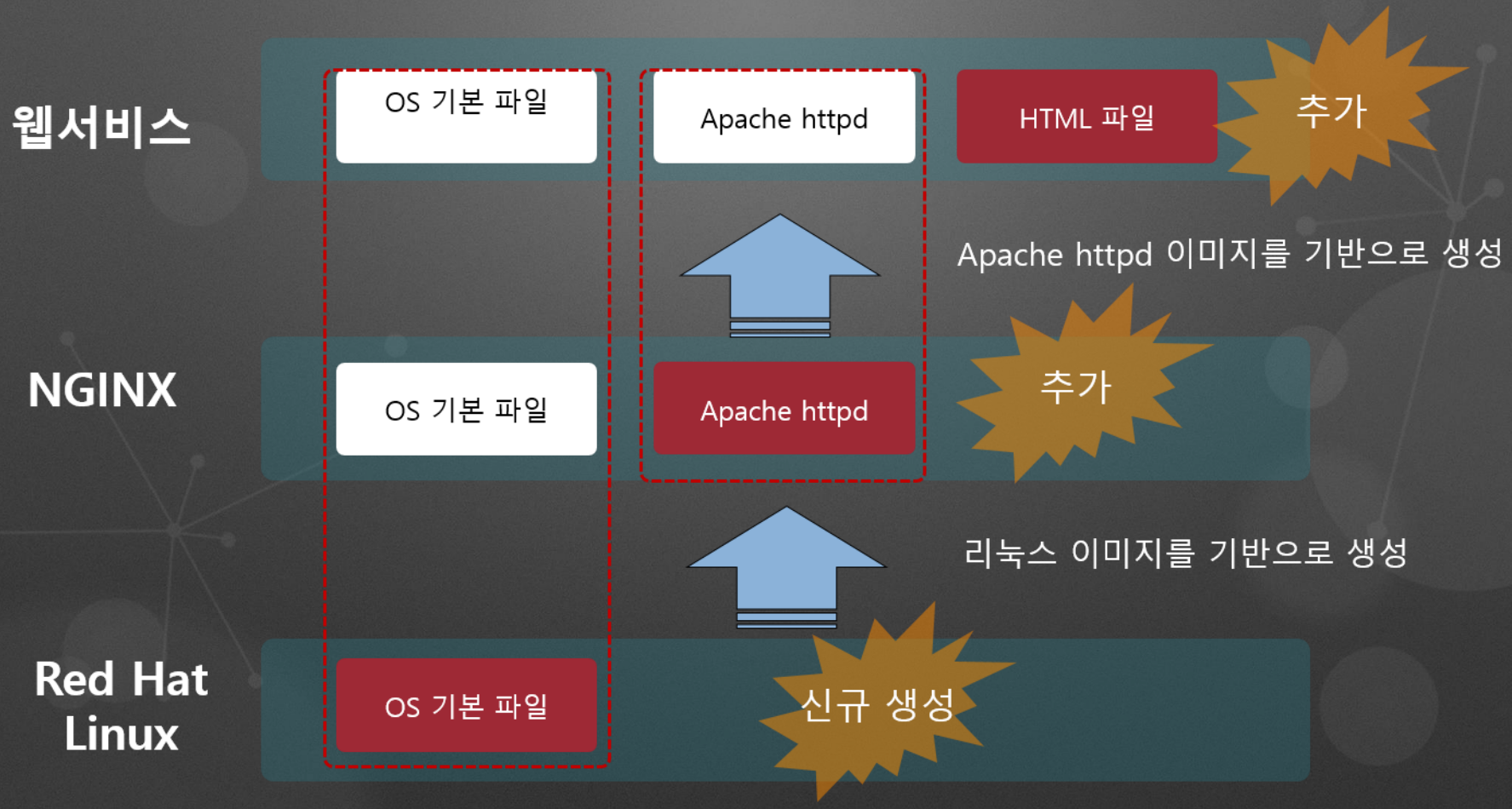
2013:
DOT CLOUD PYCON
LIGHTNING TALK

2014:
GOOGLE KUBERNETES

2015:
STANDARDS VIA OCI
AND CNCF

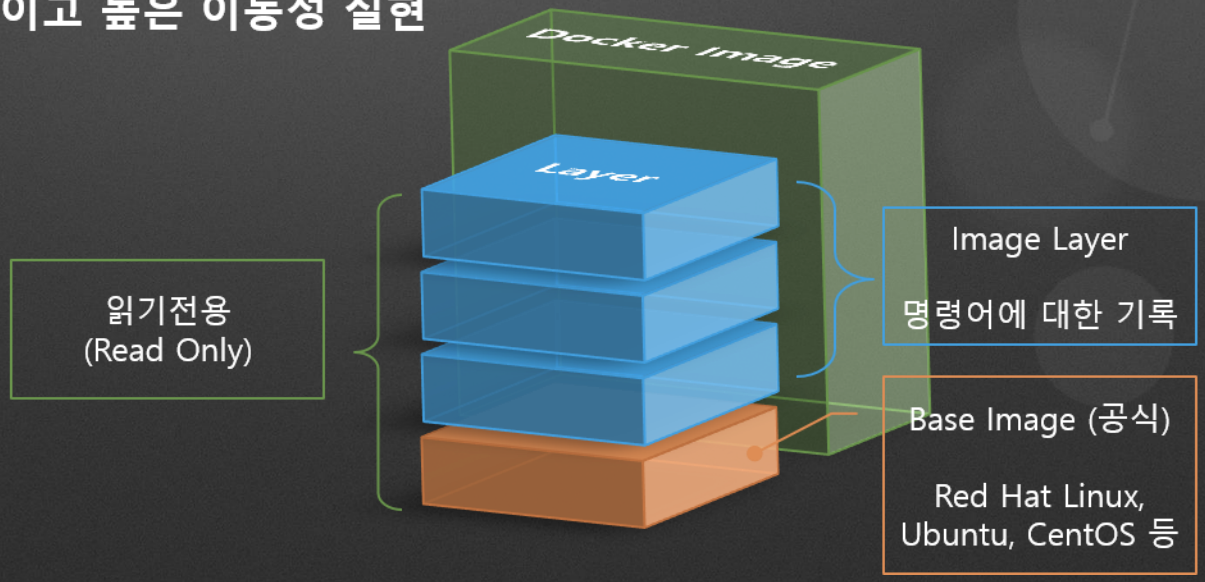
2000
2005
2010
2015

Docker 이미지 구조 예시



Docker Image

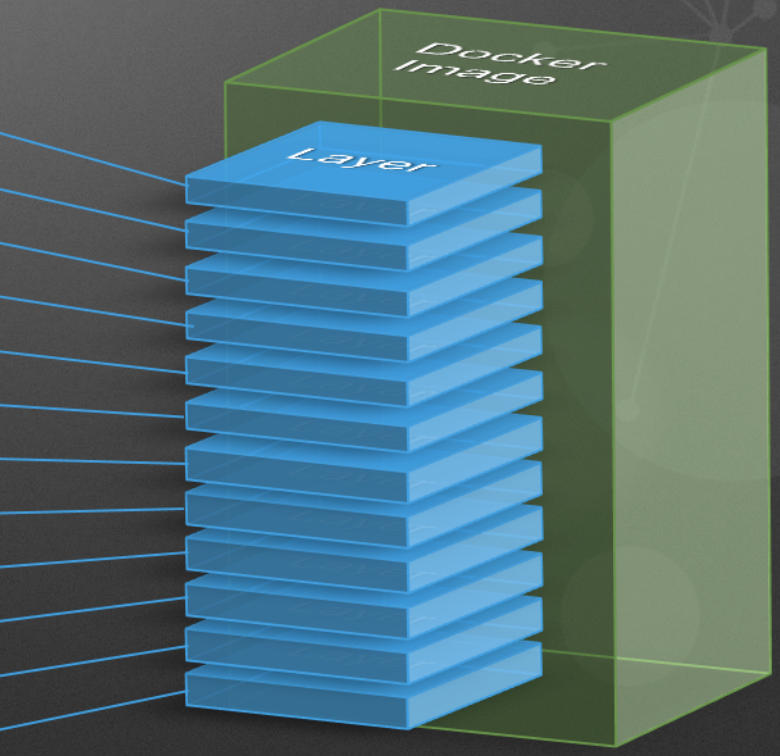
- 컨테이너를 실행할 때 필요한 파일시스템
  - 이미지 레이어의 집합체
    - 파일 내용과 메타 정보를 포함
  - 레이어는 부모와 자식 관계
  - 변경분만 기록
  - Read Only (읽기 전용) 으로 쓸 수 없음
- 공통 레이어를 이미지 간에 공유
  - 디스크 용량을 줄이고 높은 이동성 실현



Docker Image 구조

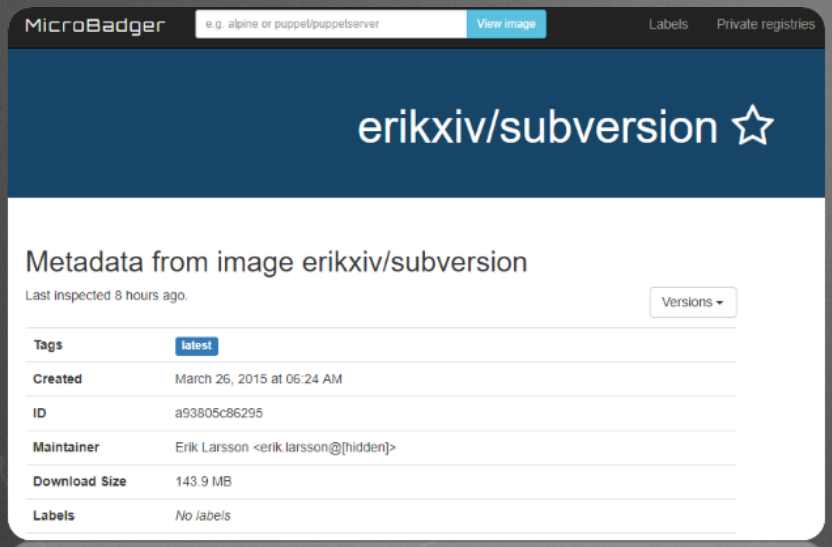
- **docker history <image id / name>**

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 Docker 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



Docker Image에 대한 Layer 정보

opennaru

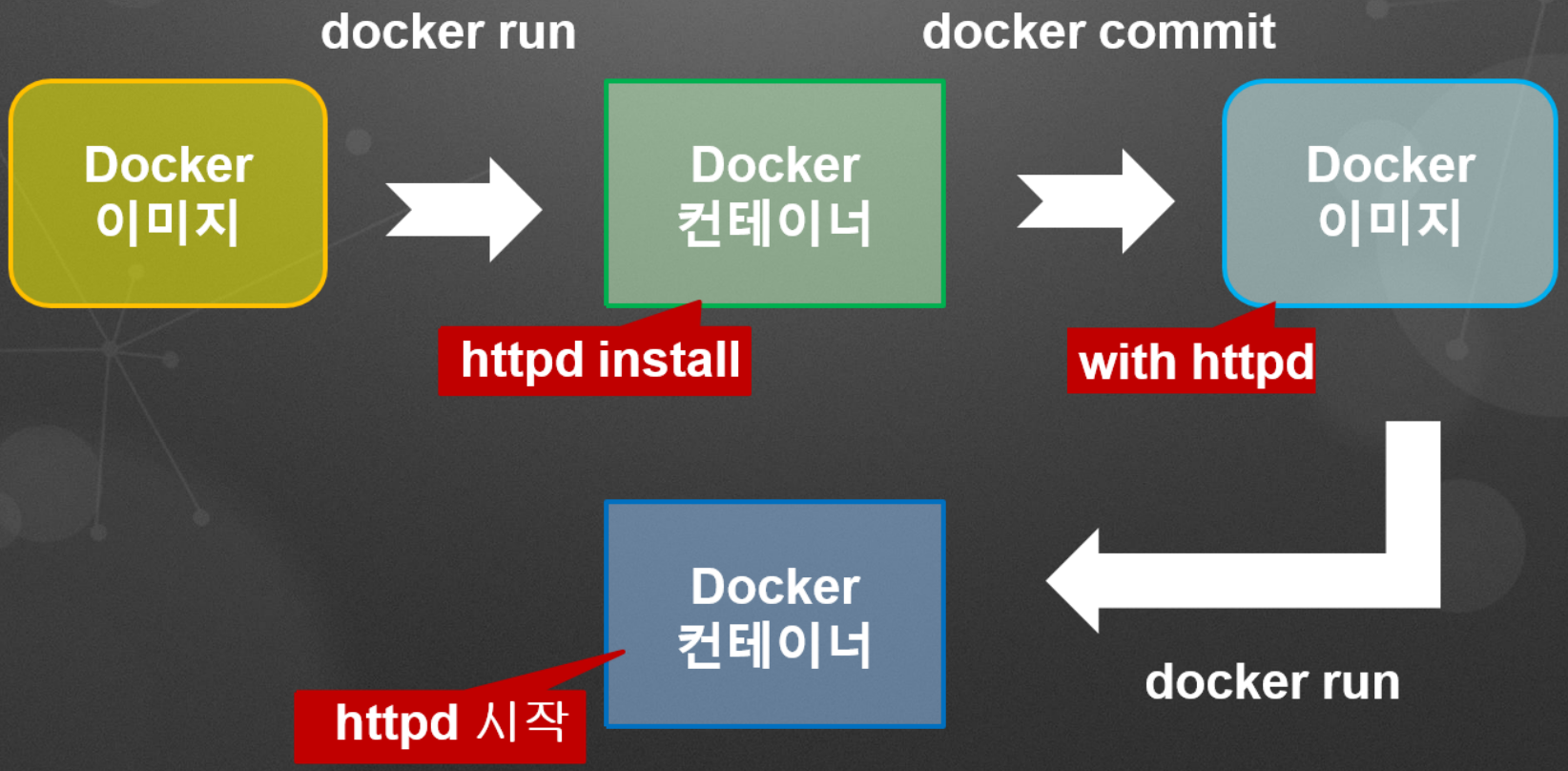


Source: https://microbadger.com/images/erikxiv/subversion

사용자 Docker 이미지 만들기

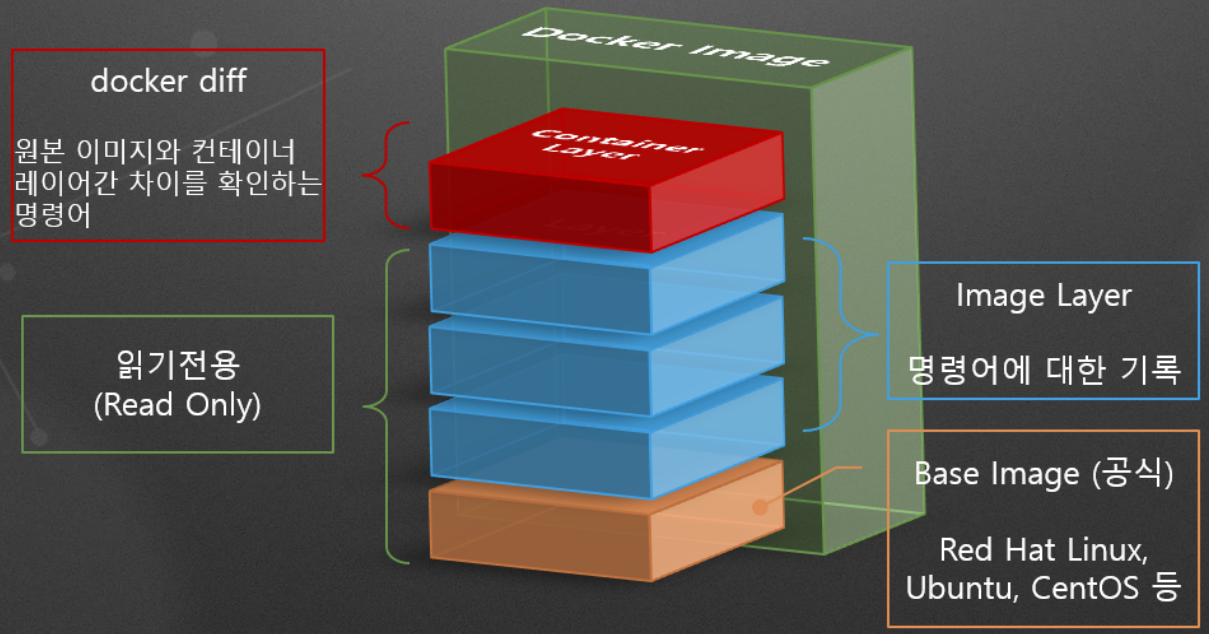
opennaru

- Docker 이미지는 Docker 컨테이너 생성의 기반이 되는 이미지
- 대표적으로 Red Hat Linux, ubuntu , centos 등이 이미지화 됨
- Docker 컨테이너는 한 Docker 이미지를 기반 작성된 각각의 애플리케이션 환경 제공

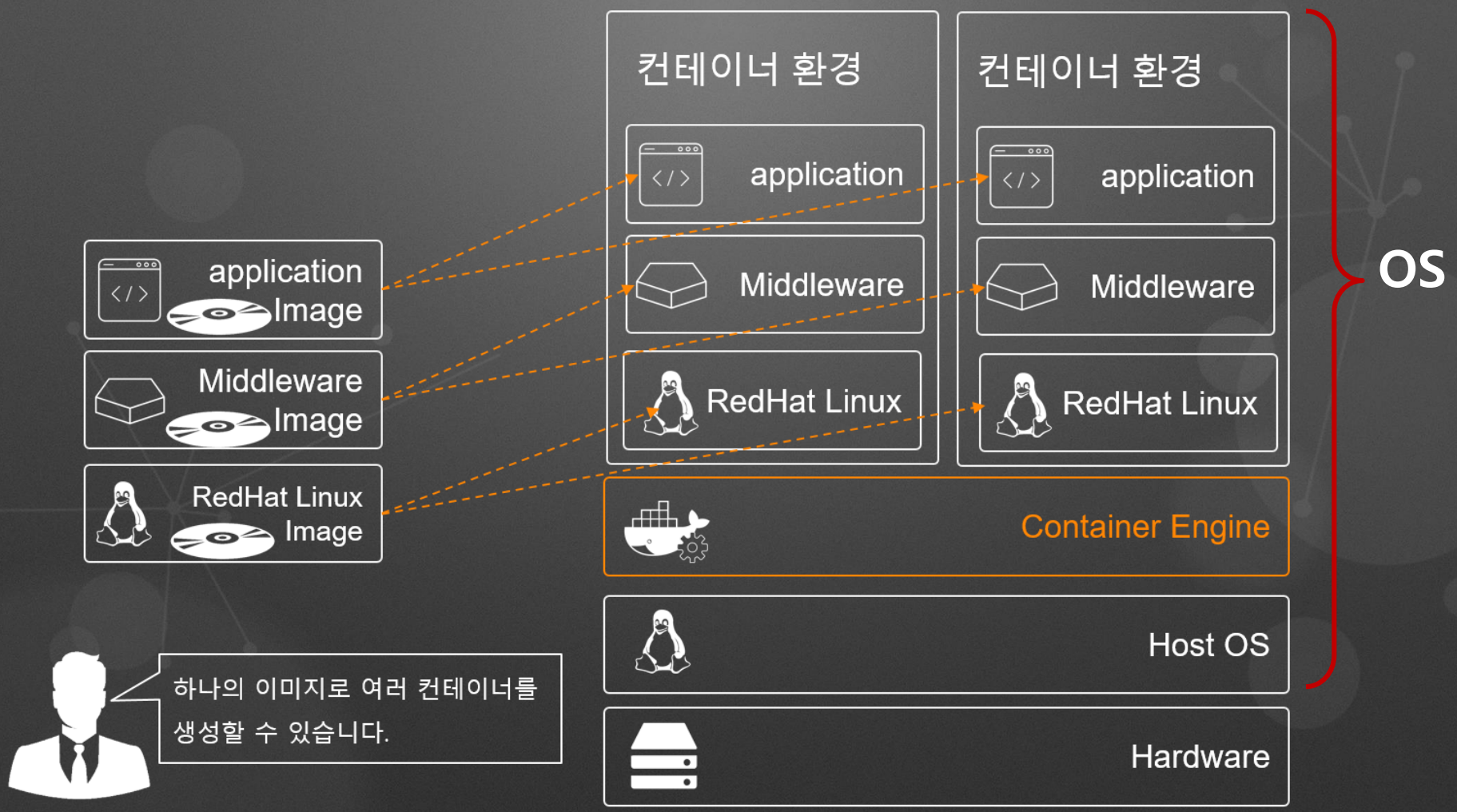


Docker Container 실행

- 새로운 이미지 레이어 자동 할당
- 이미지를 기반으로 독립 한 프로세스로 실행 (컨테이너 상태)
- docker run <opts> <image : tag>

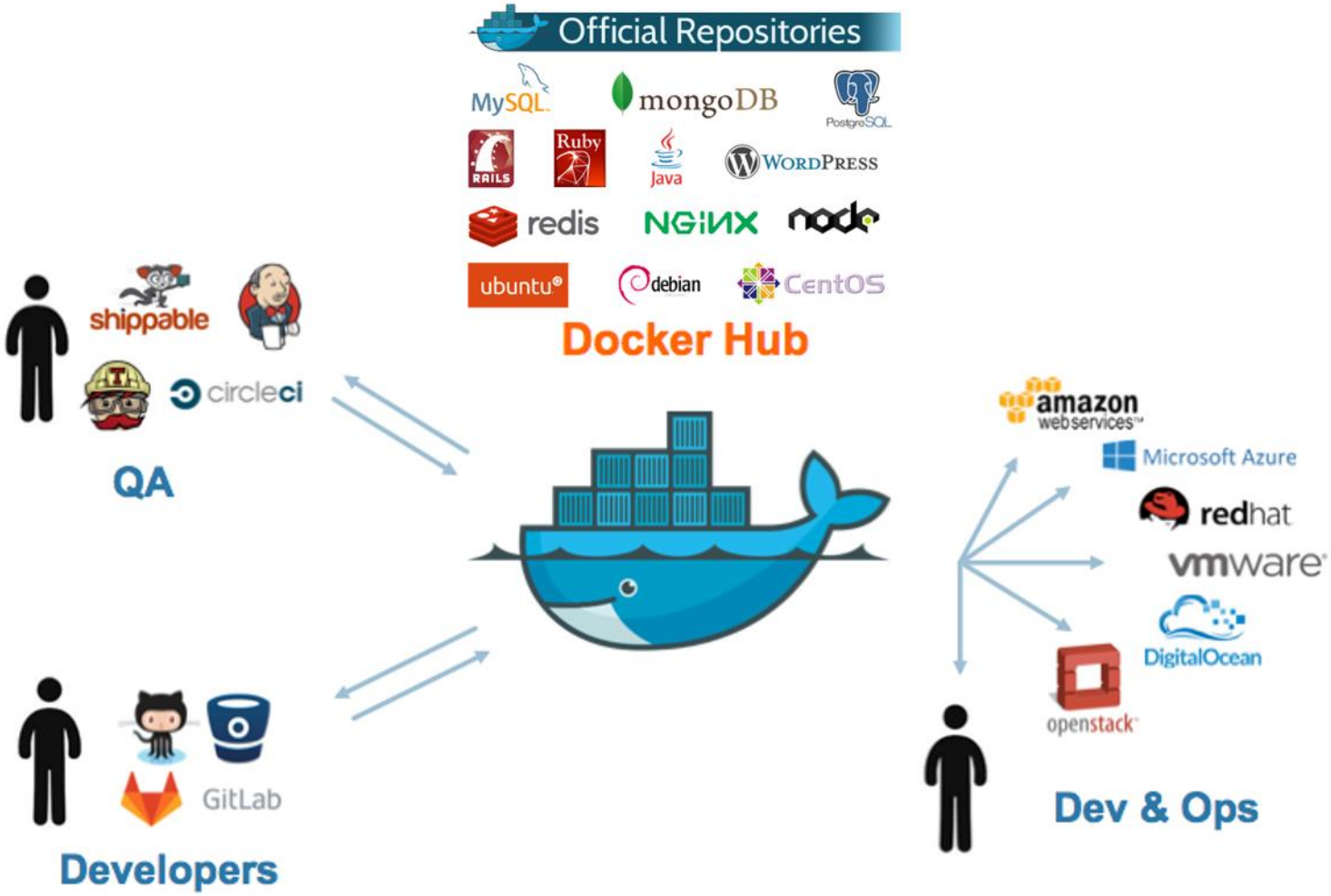


컨테이너의 동작



Docker Hub

- Docker는 Linux 커널을 사용하고 있는 환경에 있다면 어디서나 작동하기 때문에 플랫폼을 의식하지 않고 사용 (Linux 만)



opennaru

Application Performance Management

Container 기반 운영환경

KHAN
[apm]

물리환경에서 가상화와 컨테이너 환경까지

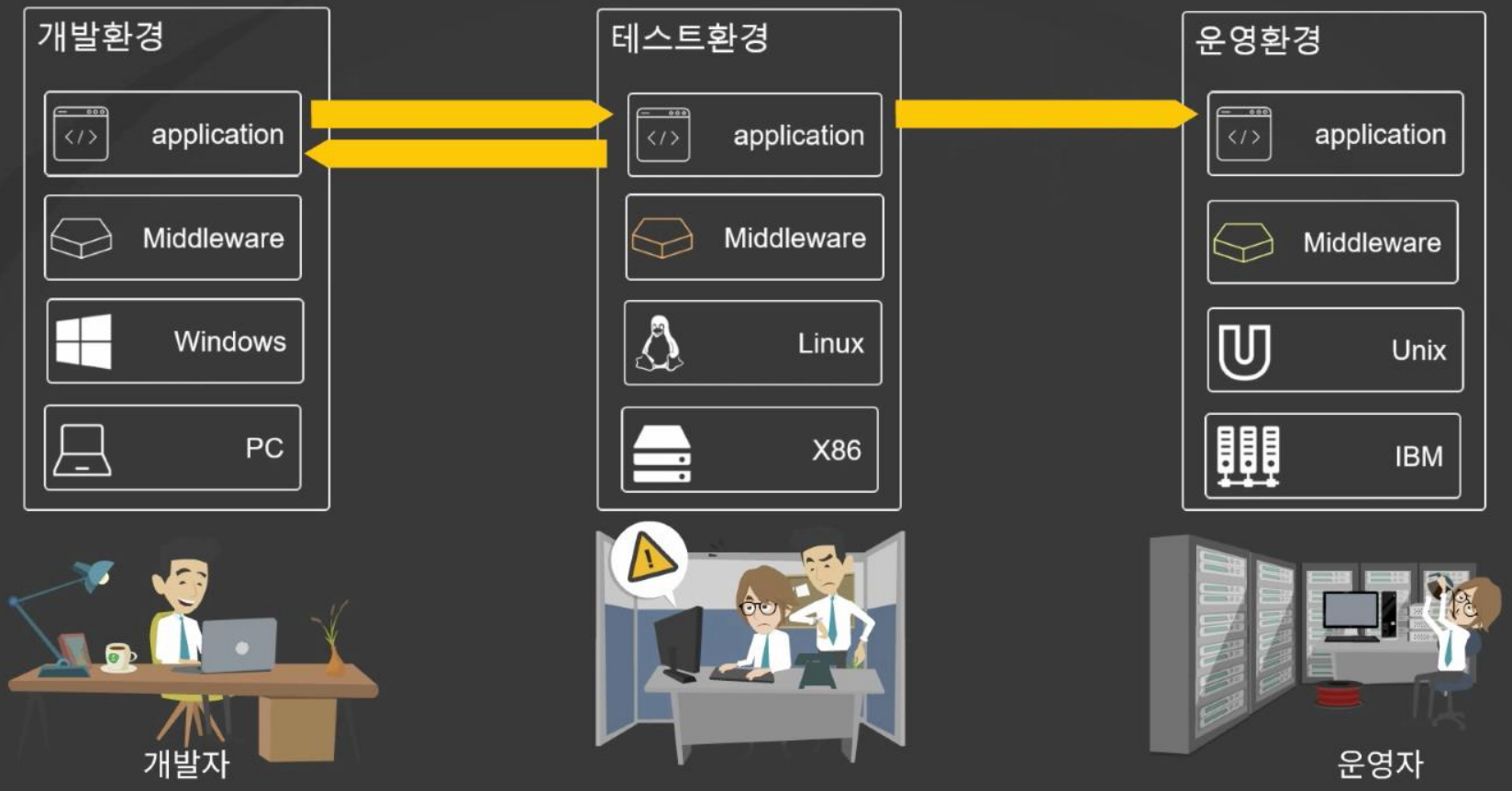
opennaru

KHAN [apm]

애플리케이션 배포 방법의 변화 – 물리 환경

opennaru

KHAN [apm]



애플리케이션 배포 방법의 변화 – 컨테이너 환경

opennaru

KHAN [apm]

Deployment & Packaging

opennaru

KHAN [apm]

Deploying Applications Faster

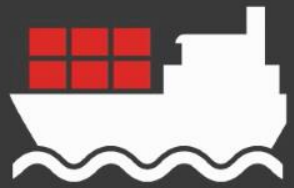
27시간

PHYSICAL SERVERS

12 분

VIRTUAL SERVERS

10 초



CONTAINERS

opennaru

Application Performance Management

Red Hat Openshift

Google에서는 모든 것이 컨테이너로 움직이고 있다

opennaru

Gmail, 검색, 맵, ...

MapReduce, 배치, ...

GFS, Colossus, ...

Google Cloud Platform도!

VM이 컨테이너로 움직이고 있다

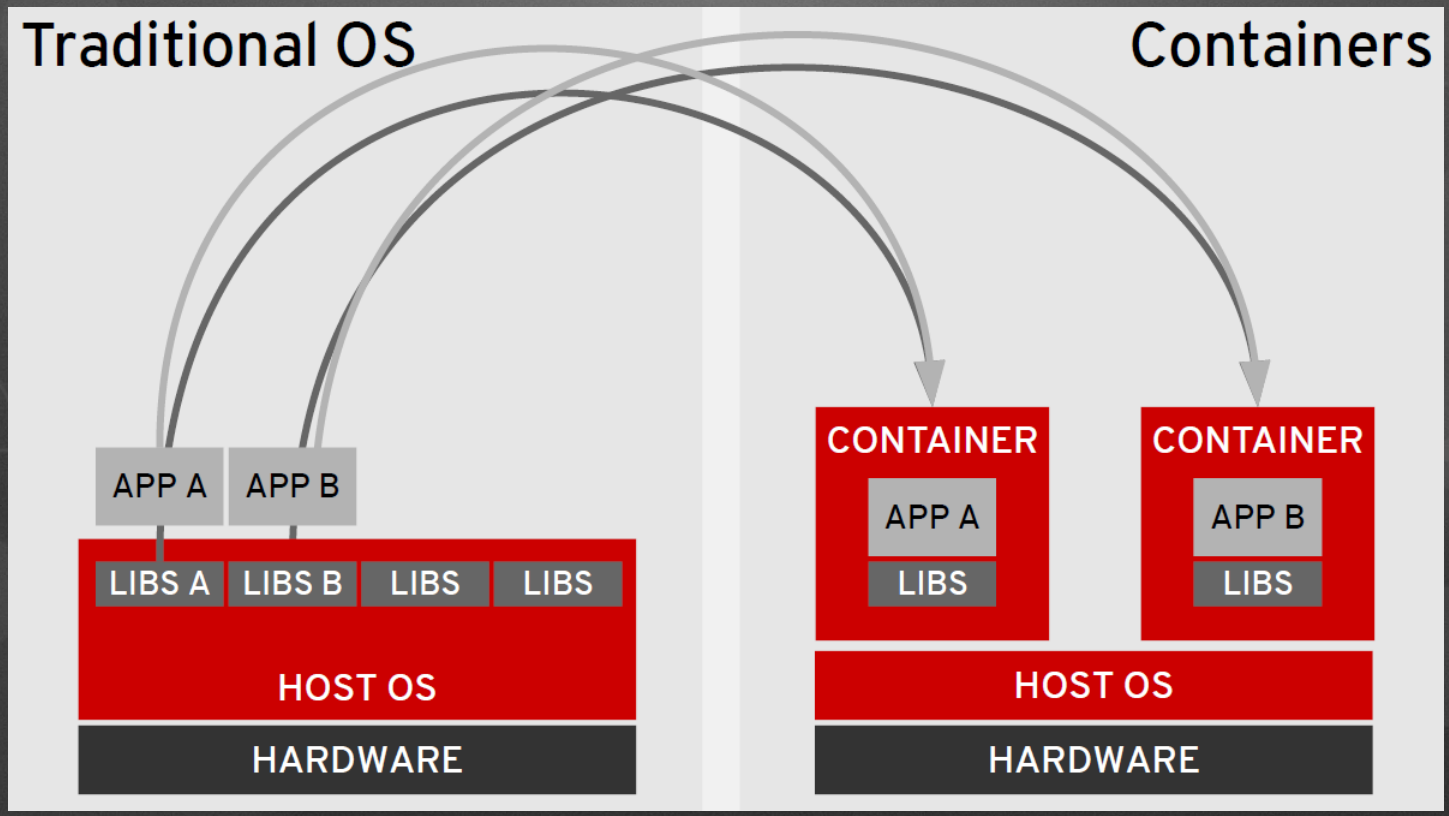
매주 20억 개 이상의 컨테이너를 기동하고 있다

Google

TRADITIONAL OS VS. CONTAINERS

opennaru

Packaged dependencies = faster boot times + greater portability



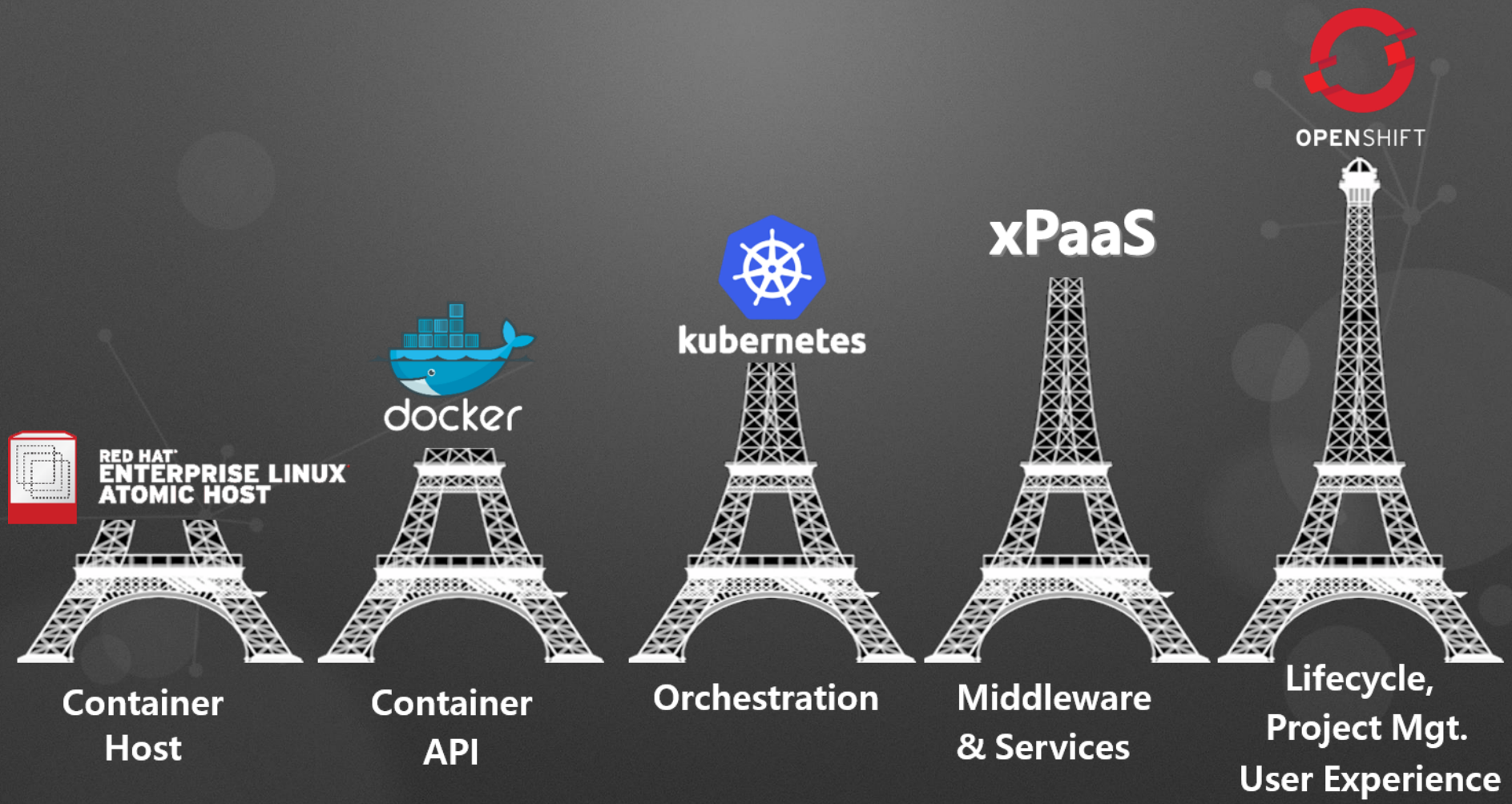
컨테이너 오케스트레이션 - Kubernates

opennaru

- 다수의 클라우드와 온프레미스 환경을 지원함
- 구글의 컨테이너 운영 경험으로 만들어진 프로젝트
- GO 언어로 작성됨
- Opensource Software
- 서버를 관리하기 보단 애플리케이션을 관리함

OVERVIEW: OPENSHIFT 3 Components

opennaru



OpenShift 가 제공하는 가치

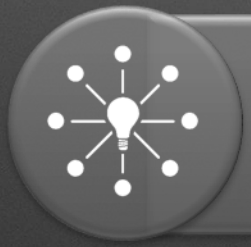
opennaru

신속한 어플리케이션 배치와 DevOps 환경

- 생산성의 높은 시스템 개발 환경
- 민첩한 어플리케이션 운영 환경 제공해
- 비즈니스 기여

폭발적인 Openshift 도입 증가

- 14 개의 업종에서 70 이상의 고객사에서 도입
- OpenShift 를 이용한 DevOps 환경 확산



Opensource 를 기반으로한 기술 혁신

- Red Hat 은 OpenShift 커뮤니티 뿐만이 아니라, 관련한 Docker 나 Kubernetes ,Project Atomic 등 다수의 기술 혁신을 선도

엔터프라이즈 PaaS 플랫폼

- 기업에서 안심하고 사용할 수 있는 DevOps플새폼
- **OS 이외의 미들웨어를 제공**
- **진정한 의미의 오픈 하이브리드 클라우드 환경**

OpenShift V3

opennaru

- PaaS에 필요한 기능 추가
  - 사용자 관리, 인증
  - 네트워크 분리
  - 소스에서 부터 배포 까지
- Docker, Kubernetes 등합

OPENSHIFT

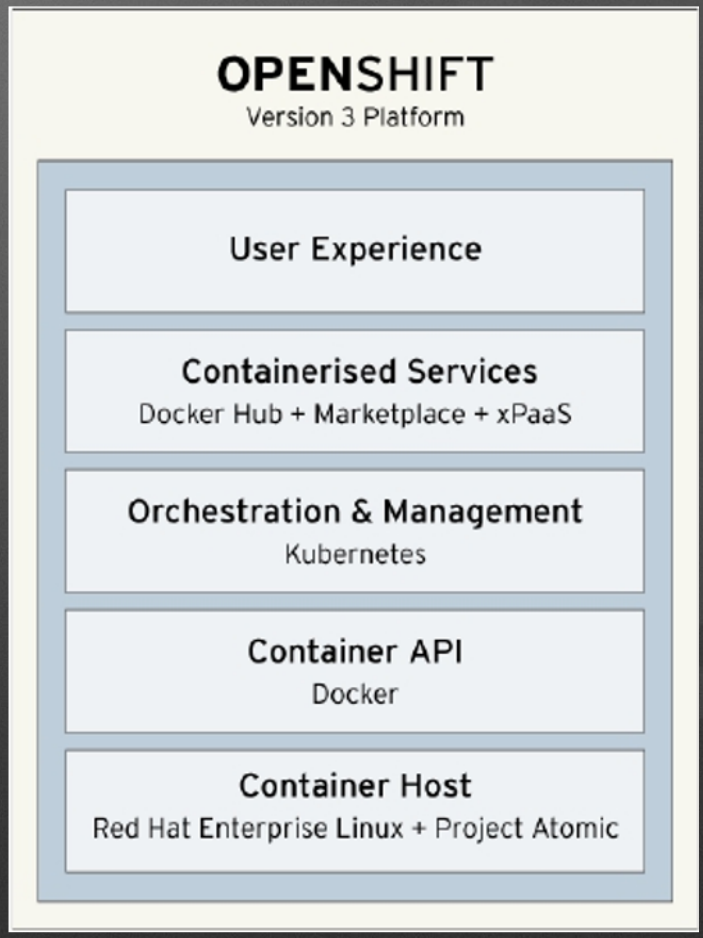
- Web UI, 네트워크 관리, 사용자 관리
- Jenkins 연계
- 소스부터 서비스 구축 등의 서비스

kubernetes

- 프록시, 로드밸런스 제공
- 컨테이너 라이프 사이클 관리
- 컨테이너를 조합해 서비스구성

docker

- 컨테이너 파일 포맷
- Linux 컨테이너 인터페이스



opennaru

Application Performance Management

KHAN
[apm]

감사합니다.

KHAN
[apm]

제품이나 서비스에 관한 문의

콜 센터 : 02-469-5426 (휴대폰 : 010-2243-3394)

전자 메일 : sales@opennaru.com