Docker & Kubernetes

도커와 쿠버네티스 기초 _____ 최용호

발표자

- 최용호
- 넥슨 코리아
- 자바카페 운영진
- AWSKRUG 판교 소모임 운영진

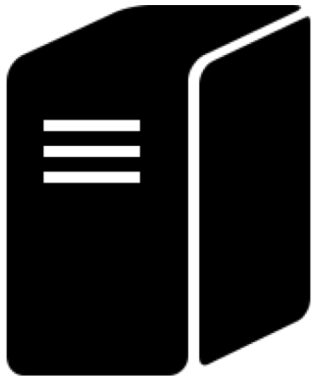
알아볼 내용

- Docker
- Docker compose
- Kubernetes

Docker가 좋은 이유 #1

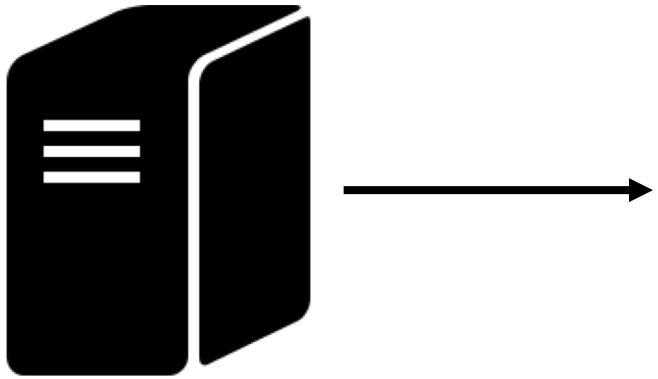
재현성
(Reproducibility)

어플리케이션 서비스 환경 구축



서버

어플리케이션 서비스 환경 구축



서버

SVN 서버 (Ubuntu) 구축 가이드

최용호 [yongho1037]님이 작성, 11월 22, 2018에 최종 변경

1. Subversion 패키지 설치

```
sudo aptitude install subversion
```

2. 그룹 설정

```
sudo addgroup svn
sudo gpasswd -a "$USER" svn
```

3. SVN 서버가 될 호스트에 Repository 생성

```
sudo mkdir -p /svn/repository
```

   a. 디렉토리 경로는 임의로 설정 했기 때문에 필요 시 수정 가능하다.
   b. 해당 디렉토리가 최상위 디렉토리가 되고 실제 SVN Repository들을 하위에 생성한다.

4. 생성한 Repository 최상위 디렉토리에 SVN 사용자 그룹과 권한 추가

```
sudo chgrp -R svn /svn/repository
sudo chmod -R 770 /svn/repository
```

5. 실제 원격에서 SVN Update를 수행할 Repository를 생성한다.

```
svnadmin create /svn/repository/my_project
```

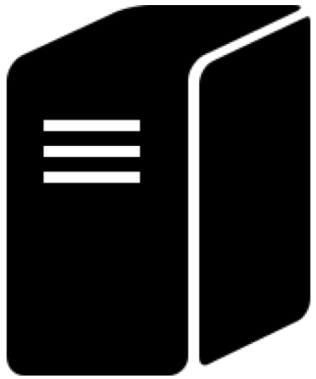
   a. create 명령 수행 후 Repository에 필요한 기본 디렉토리들과 파일들이 생성된다.

6. svnserve.conf 설정파일을 수정하여 SVN Repository에 접근할 수 있는 권한을 설정한다.

```
vi /svn/repository/my_project/conf/svnserve.conf
```

```
anon-access = none
auth-access = read
password-db = passwd
```

어플리케이션 서비스 환경 구축



서버

데이터베이스

SVN 서버 (Ubuntu) 구축 가이드

최용호 [yongho1037]님이 작성, 11월 22, 2018에 최종 변경

1. Subversion 패키지 설치

```
sudo aptitude install subversion
```

2. 그룹 설정

```
sudo addgroup svn
sudo gpasswd -a "$USER" svn
```

3. SVN 서버가 될 호스트에 Repository 생성

```
sudo mkdir -p /svn/repository
```

   a. 디렉토리 경로는 임의로 설정 했기 때문에 필요 시 수정 가능하다.
   b. 해당 디렉토리가 최상위 디렉토리가 되고 실제 SVN Repository들을 하위에 생성한다.

4. 생성한 Repository 최상위 디렉토리에 SVN 사용자 그룹과 권한 추가

```
sudo chgrp -R svn /svn/repository
sudo chmod -R 770 /svn/repository
```

5. 실제 원격에서 SVN Update를 수행할 Repository를 생성한다.

```
svnadmin create /svn/repository/my_project
```

   a. create 명령 수행 후 Repository에 필요한 기본 디렉토리들과 파일들이 생성된다.

6. svnserve.conf 설정파일을 수정하여 SVN Repository에 접근할 수 있는 권한을 설정한다.

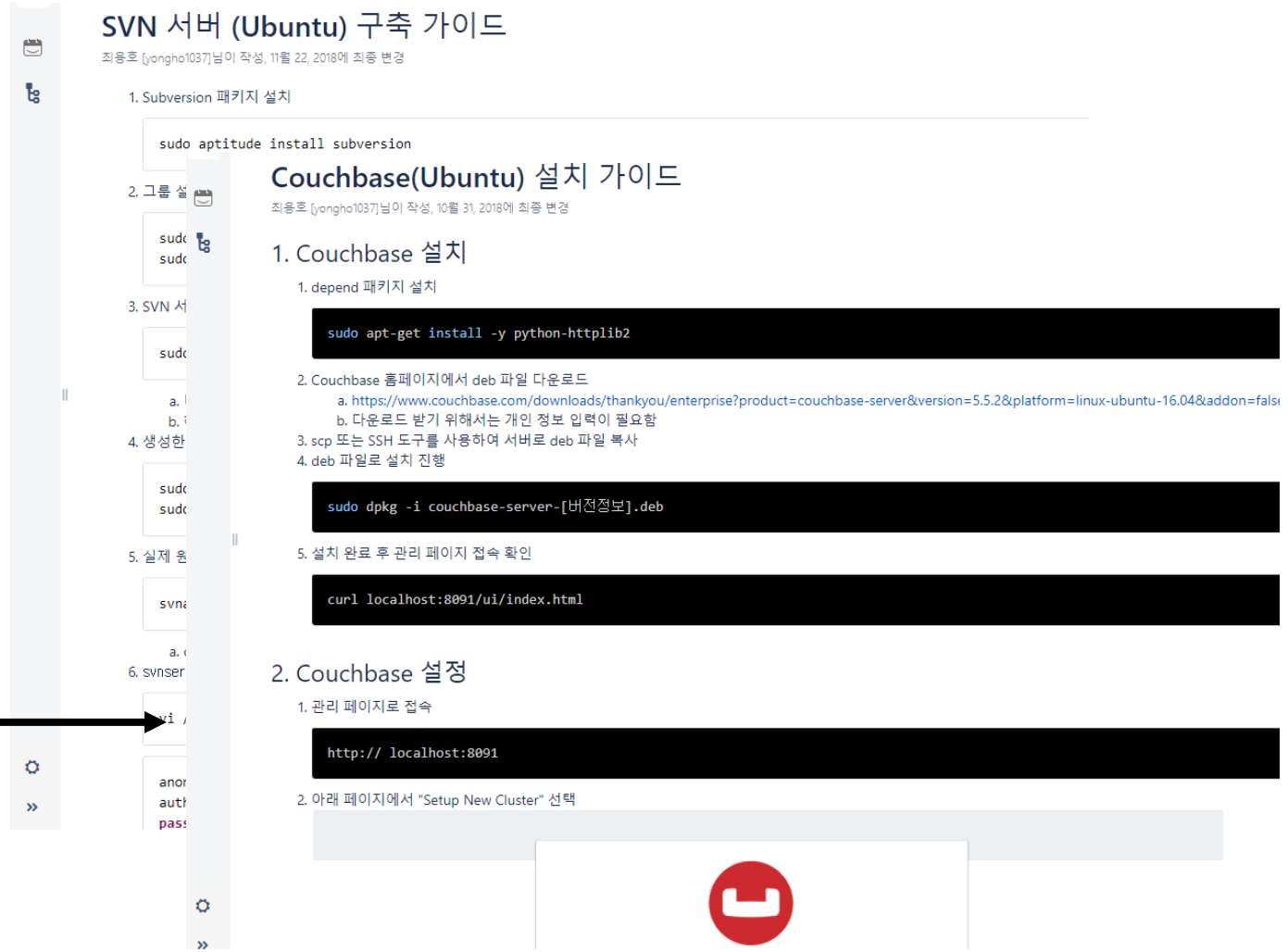
```
vi /svn/repository/my_project/conf/svnserve.conf
```

```
anon-access = none
auth-access = read
password-db = passwd
```

어플리케이션 서비스 환경 구축

서버

데이터베이스



SVN 서버 (Ubuntu) 구축 가이드

최용호 [yongho1037]님이 작성, 11월 22, 2018에 최종 변경

1. Subversion 패키지 설치

```
sudo aptitude install subversion
```

Couchbase(Ubuntu) 설치 가이드

최용호 [yongho1037]님이 작성, 10월 31, 2018에 최종 변경

1. Couchbase 설치

1. depend 패키지 설치

```
sudo apt-get install -y python-httplib2
```

2. Couchbase 홈페이지에서 deb 파일 다운로드
   a. https://www.couchbase.com/downloads/thankyou/enterprise?product=couchbase-server&version=5.5.2&platform=linux-ubuntu-16.04&addon=fals
   b. 다운로드 받기 위해서는 개인 정보 입력이 필요함
3. scp 또는 SSH 도구를 사용하여 서버로 deb 파일 복사
4. deb 파일로 설치 진행

```
sudo dpkg -i couchbase-server-[버전정보].deb
```

5. 설치 완료 후 관리 페이지 접속 확인

```
curl localhost:8091/ui/index.html
```

2. Couchbase 설정

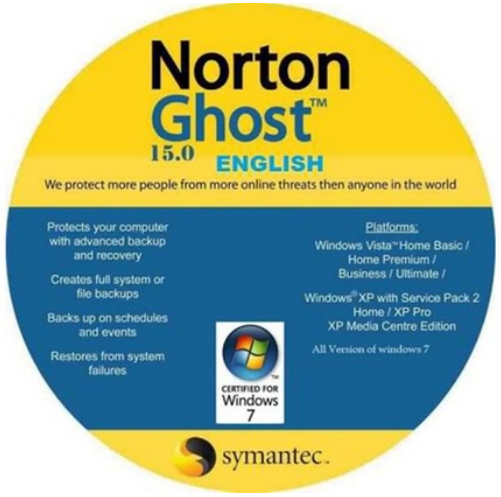
1. 관리 페이지로 접속

```
http:// localhost:8091
```

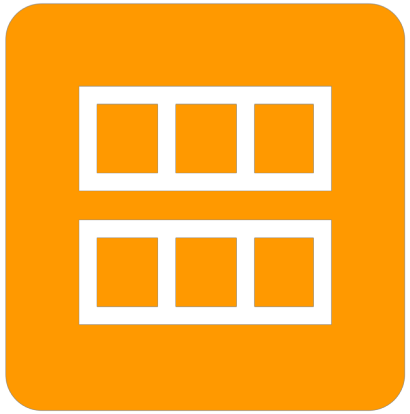
2. 아래 페이지에서 "Setup New Cluster" 선택

필요한 것들을 미리 만들어 놓을 순 없을까?

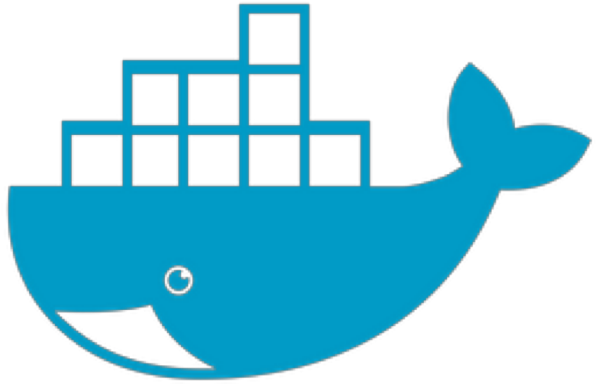
이미지



노턴 고스트

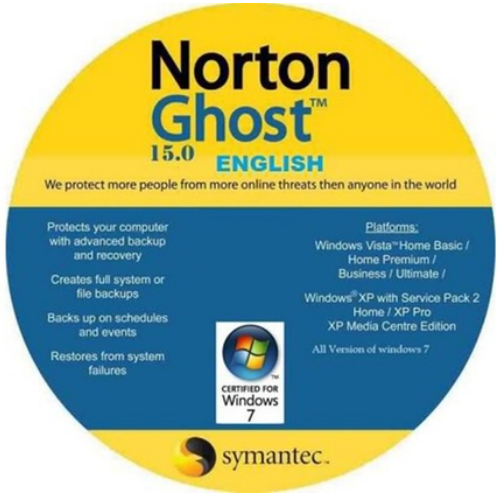


AWS AMI



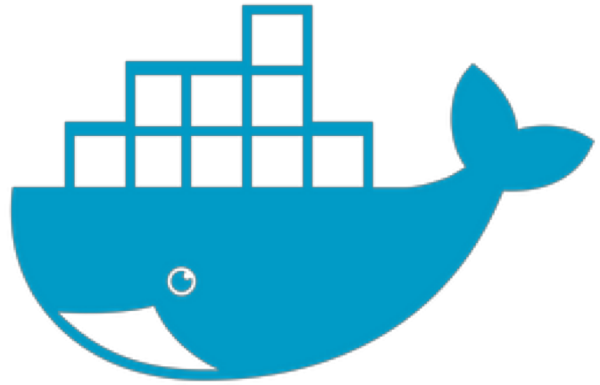
도커 이미지

이미지

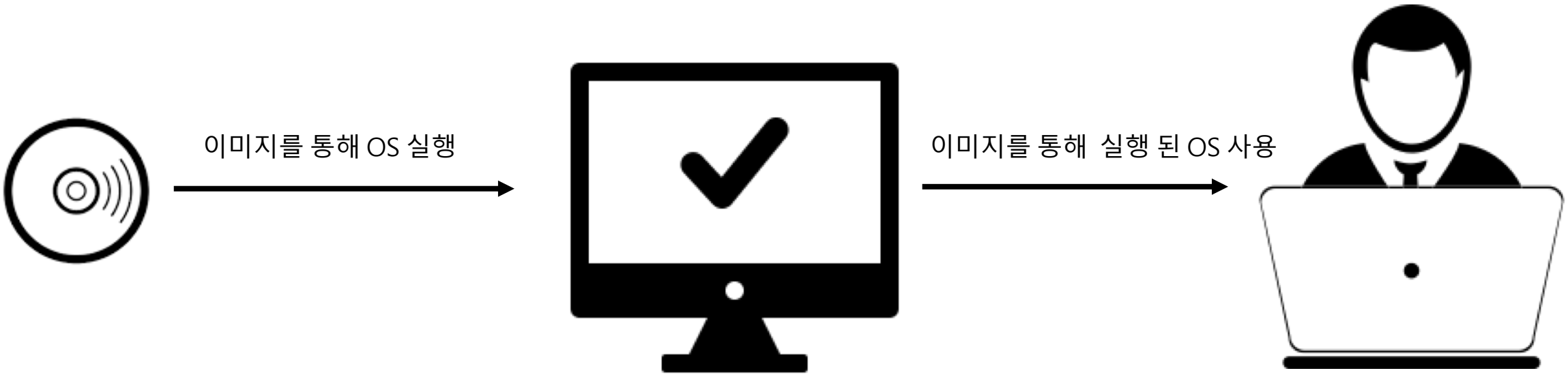


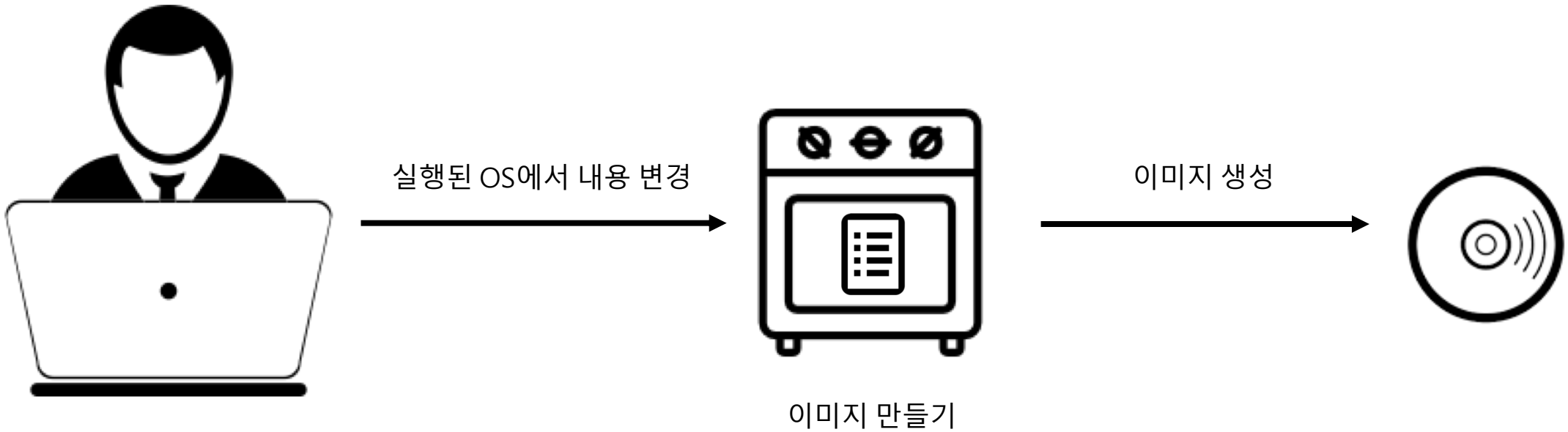
노턴 고스트

AWS AMI



도커 이미지

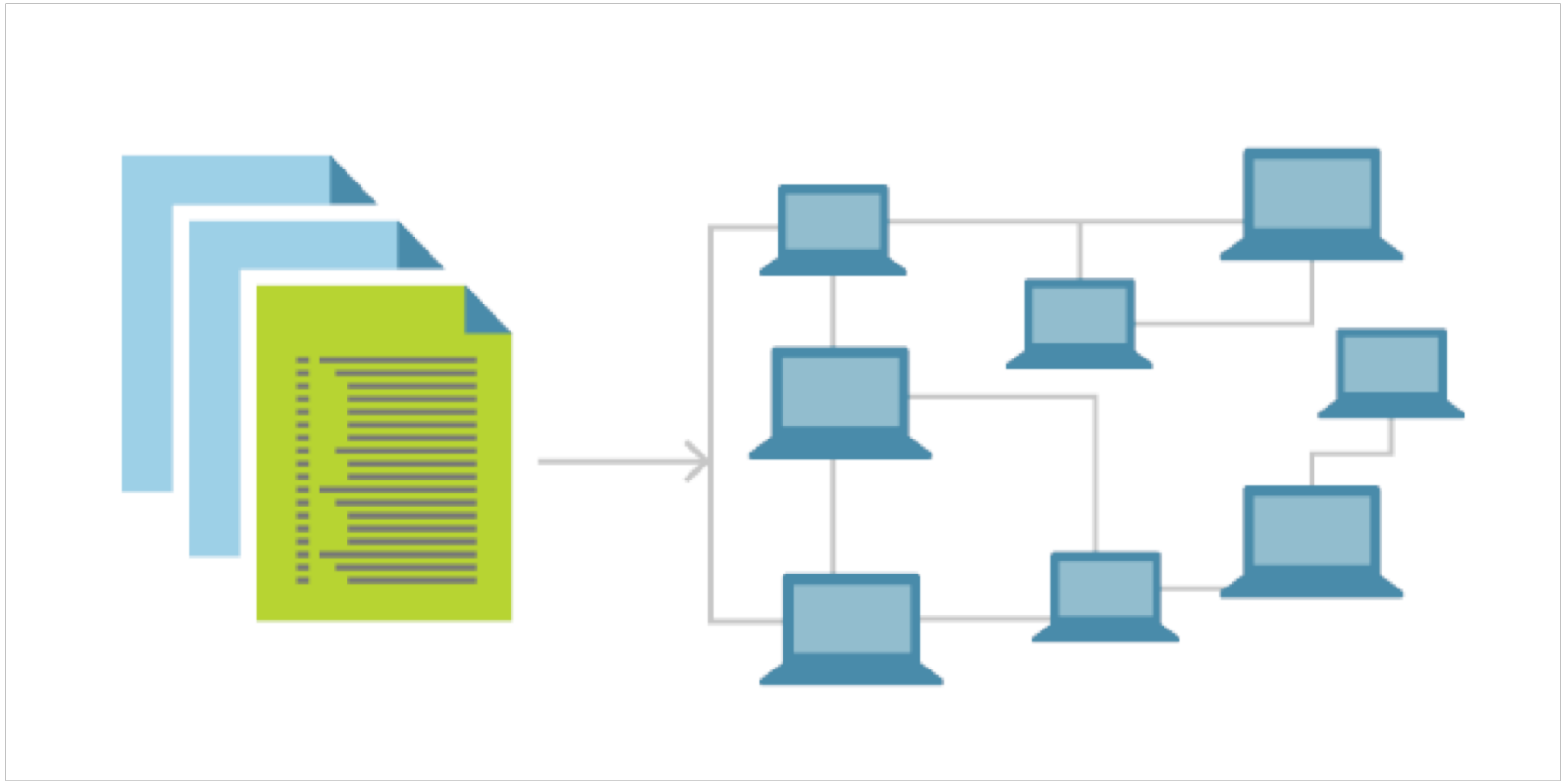

이미지를 통해 OS 실행
이미지를 통해 실행 된 OS 사용


실행된 OS에서 내용 변경
이미지 생성
이미지 만들기

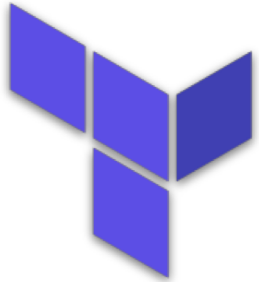

뭐가 변경된거지!!??

이미지의 내용도 확인하고 버전관리도 하려면 어떻게 해야할까?

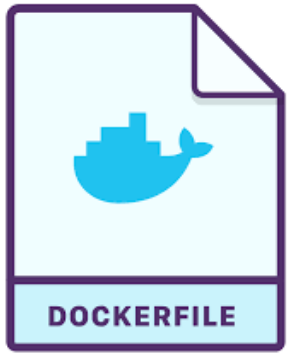
이미지 내용을 코드화

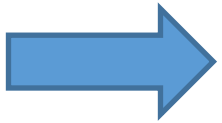


코드화를 위해 이미 많은 도구들이 존재



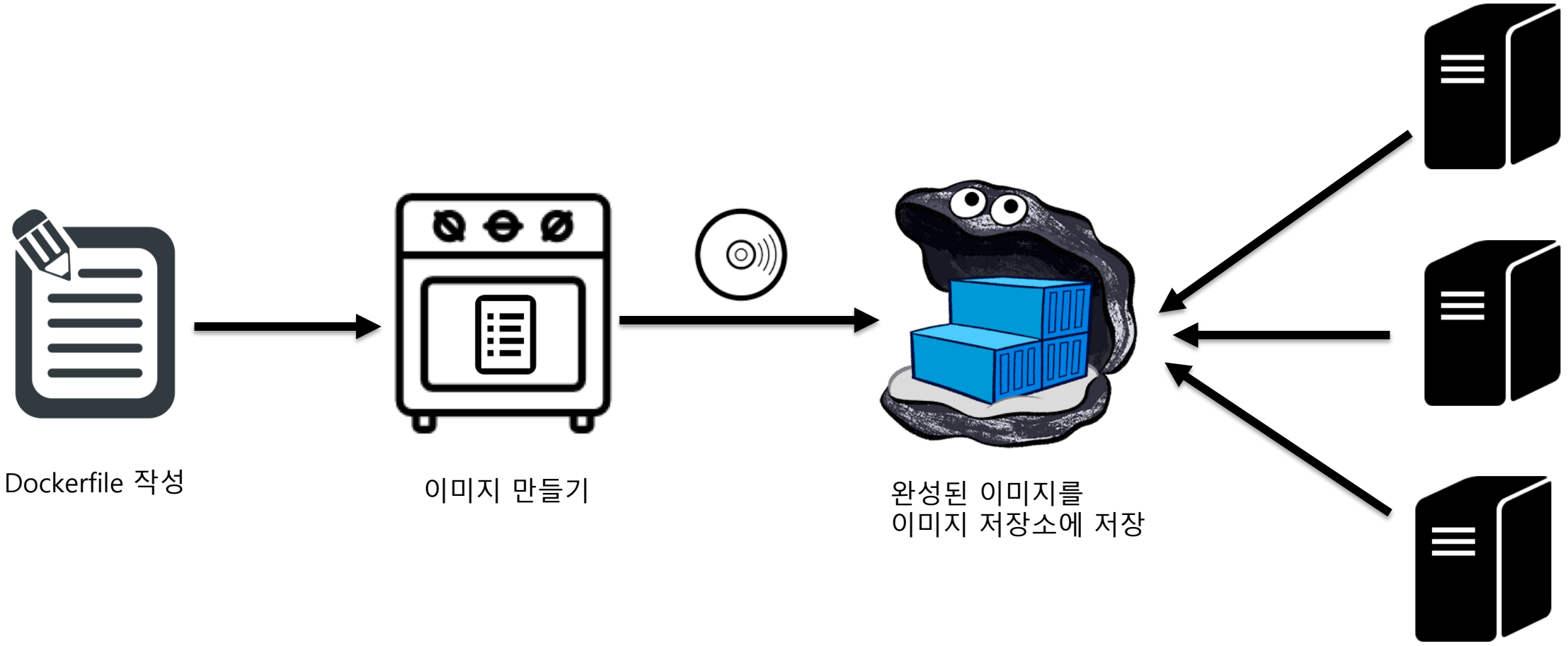
도커는 Dockerfile을 사용하여 코드화





```
FROM tomcat:8.0-jre8
MAINTAINER Server Team

RUN echo 'deb http://http.debian.net/debian jessie-backports main'
RUN rm -rf /var/lib/apt/lists/*
RUN apt-get clean && apt-get update -y && apt-get install -y opens

# SSH 관련 설정 (사용자 계정 생성 및 sudo 권한 부여)
RUN adduser --disabled-password --gecos "" hive  \
    && echo 'hive:gkdlqm' | chpasswd \
    && adduser hive sudo \
    && echo 'hive ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers \
    && mkdir /var/run/sshd

# 시간 설정
RUN rm /etc/localtime && ln -s /usr/share/zoneinfo/UTC /etc/localt

# 언어 설정
RUN sed -i -e 's/# ko_KR.UTF-8 UTF-8/ko_KR.UTF-8 UTF-8/' /etc/loca
    echo 'LANG="ko_KR.UTF-8 UTF-8"'>/etc/default/locale && \
    dpkg-reconfigure --frontend=noninteractive locales && \
    echo 'export LANG=ko_KR.utf-8' >> /etc/bash.bashrc
```

도커를 사용하면?



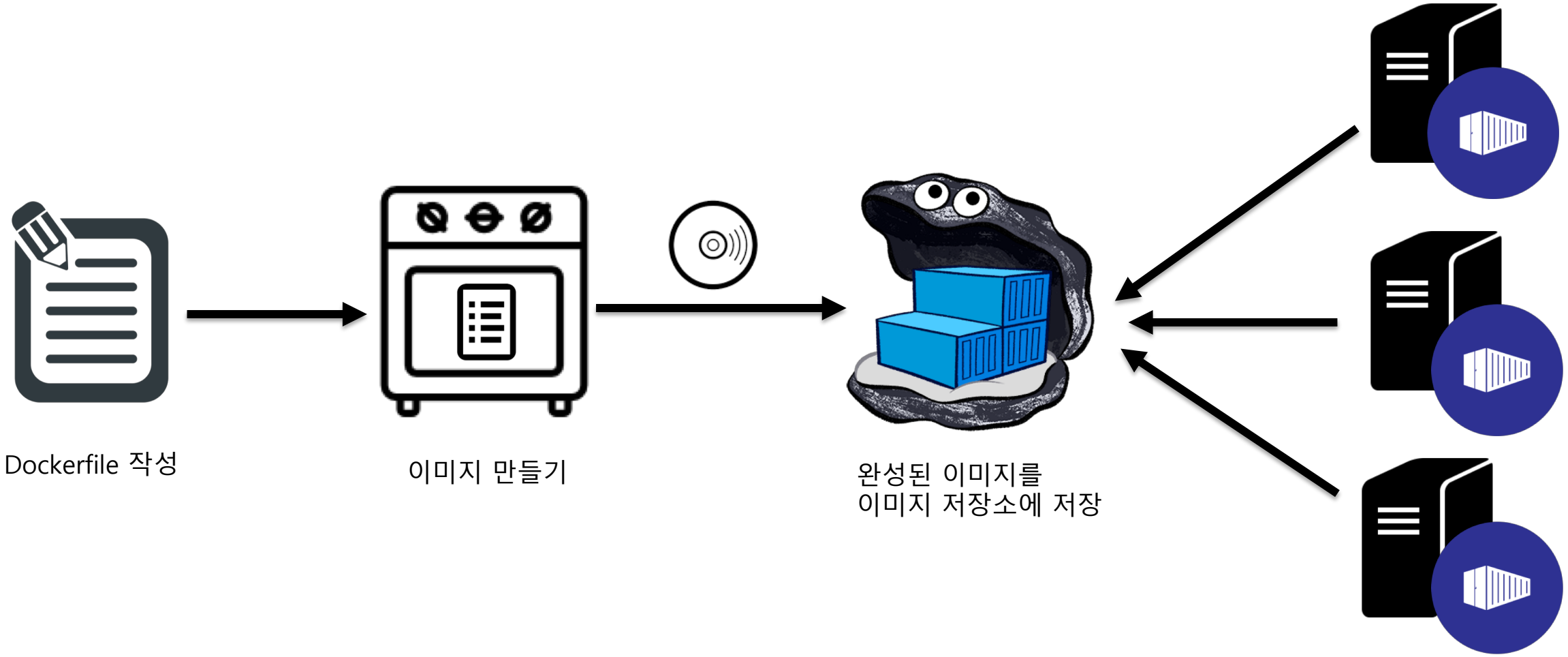
Dockerfile 작성

이미지 만들기

완성된 이미지를
이미지 저장소에 저장

각 서버에서 이미지 저장소에서
이미지를 다운로드 받아서 구동

도커를 사용하면?



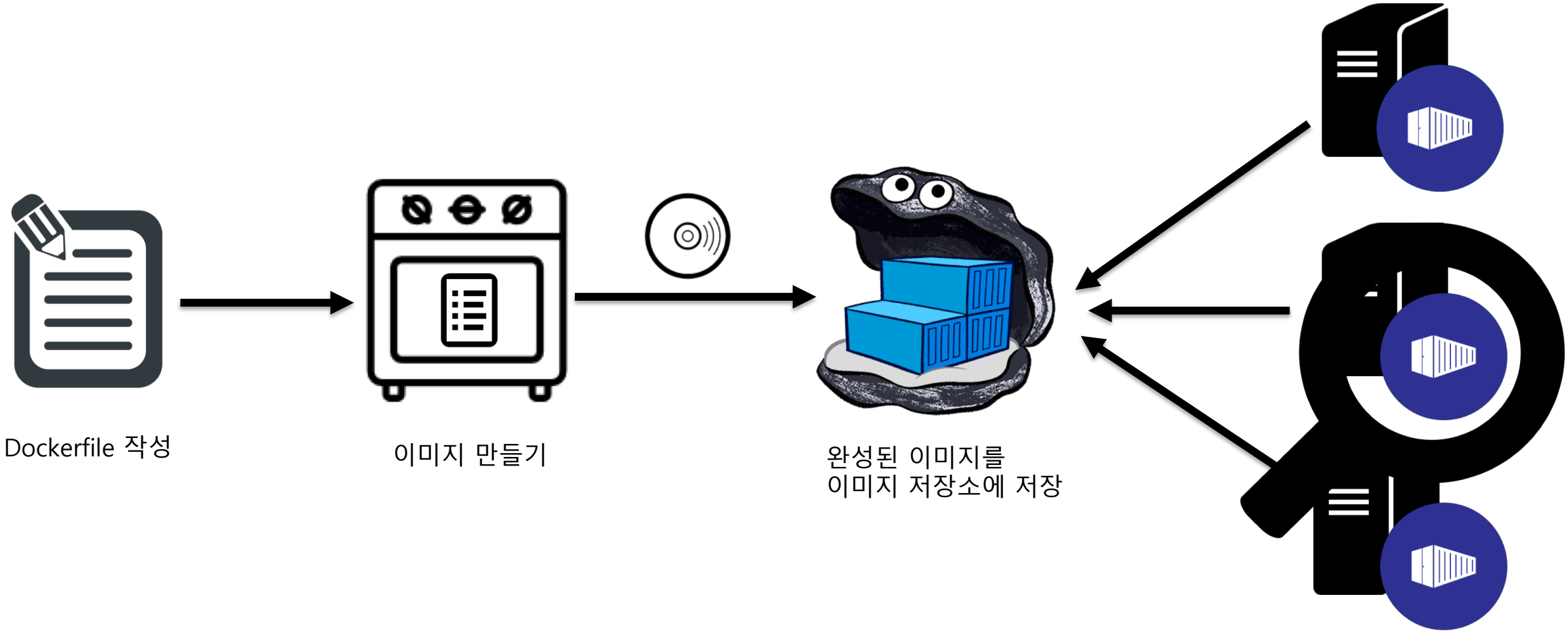
Dockerfile 작성

이미지 만들기

완성된 이미지를
이미지 저장소에 저장

이미지는 컨테이너 형태로 구동

도커를 사용하면?



Dockerfile 작성

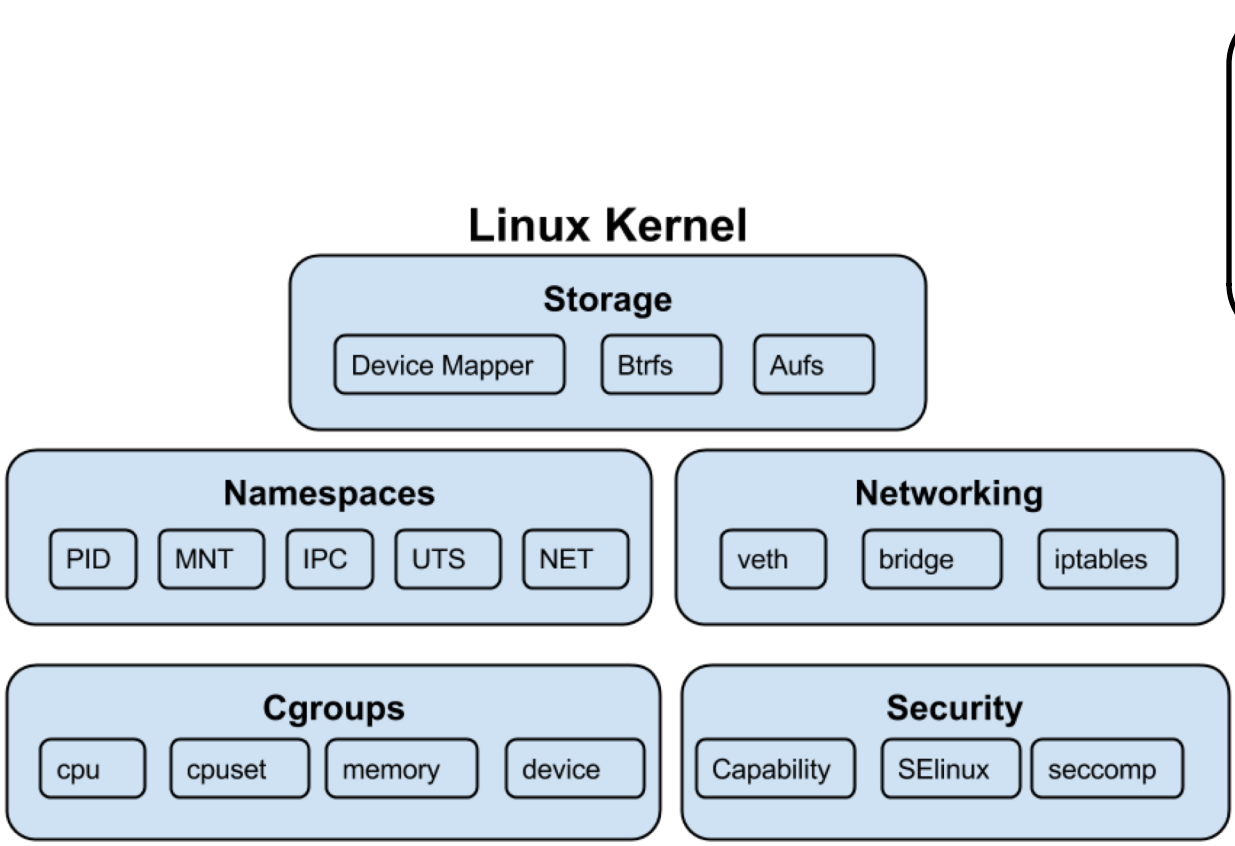
이미지 만들기

완성된 이미지를
이미지 저장소에 저장

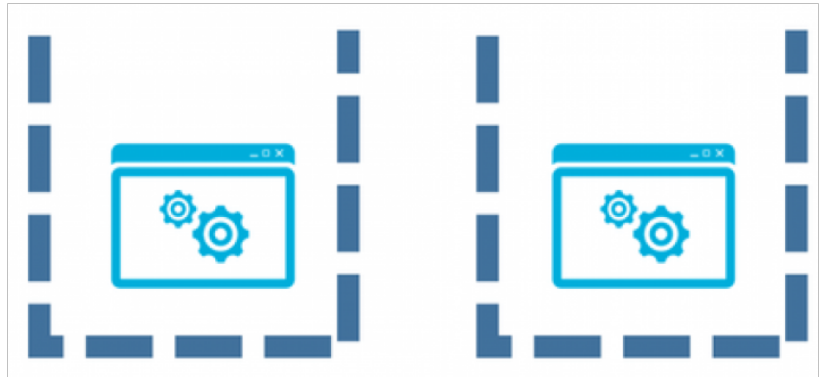
이미지는 컨테이너 형태로 구동

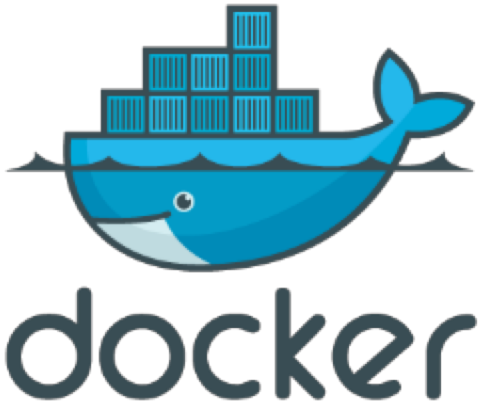
컨테이너란?

VS



리눅스 기술을 사용하여
선박의 컨테이너처럼
프로세스가 사용하는
자원을 격리





이미 존재했지만 사용하기 어려웠던 리눅스 기술들을 도커가 잘 포장하여 제공

Linux Kernel

Storage

Device Mapper | Btrfs | Aufs

Namespaces

PID | MNT | IPC | UTS | NET

Networking

veth | bridge | iptables

Cgroups

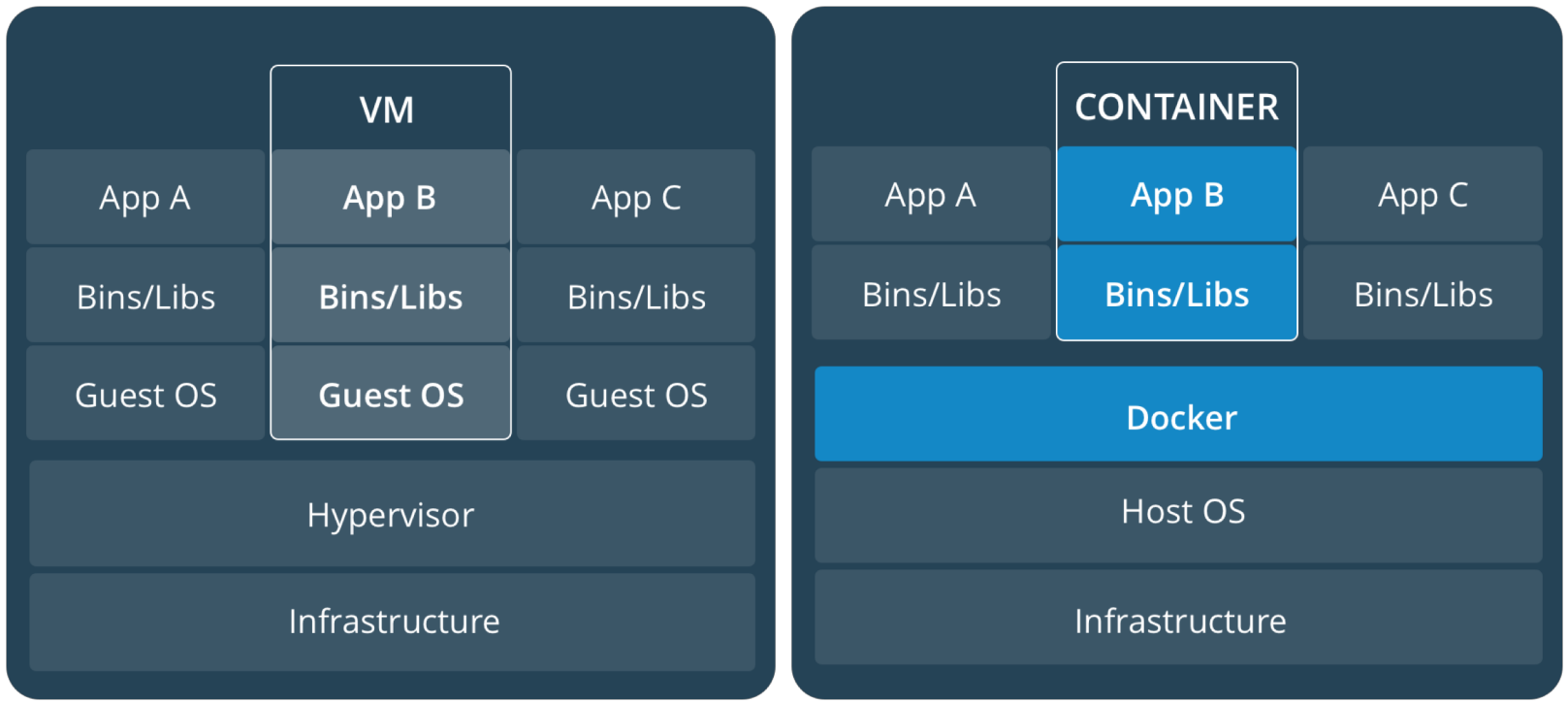
cpu | cpuset | memory | device

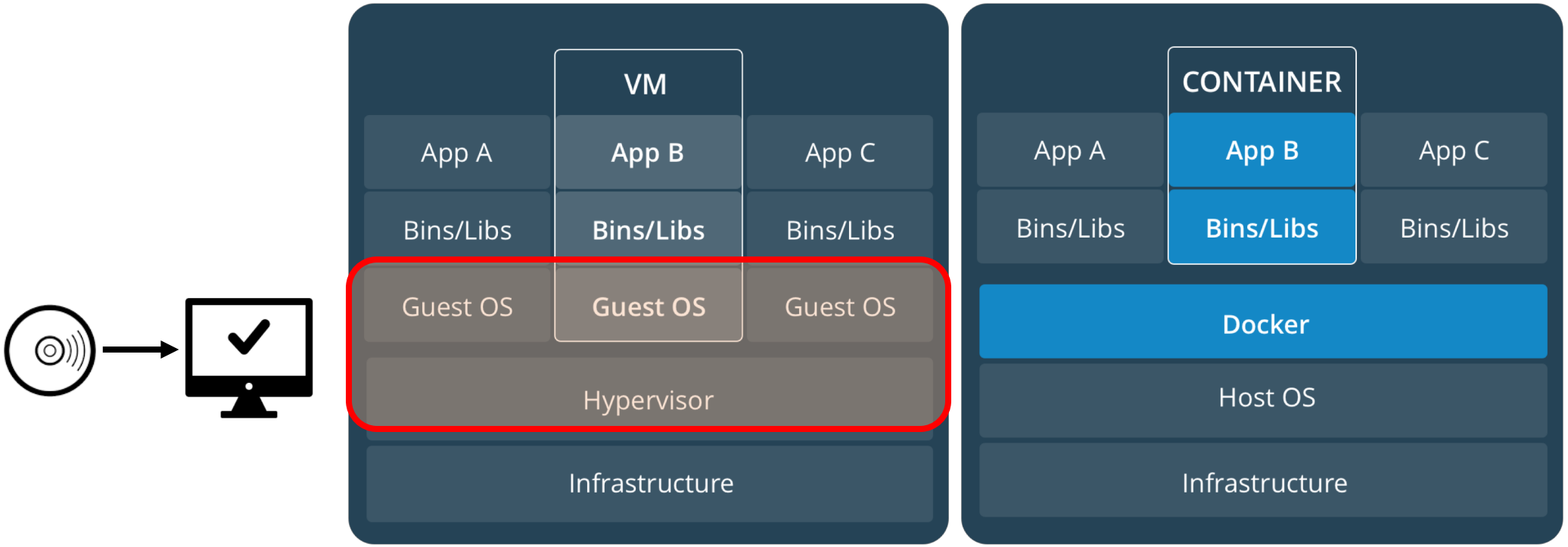
Security

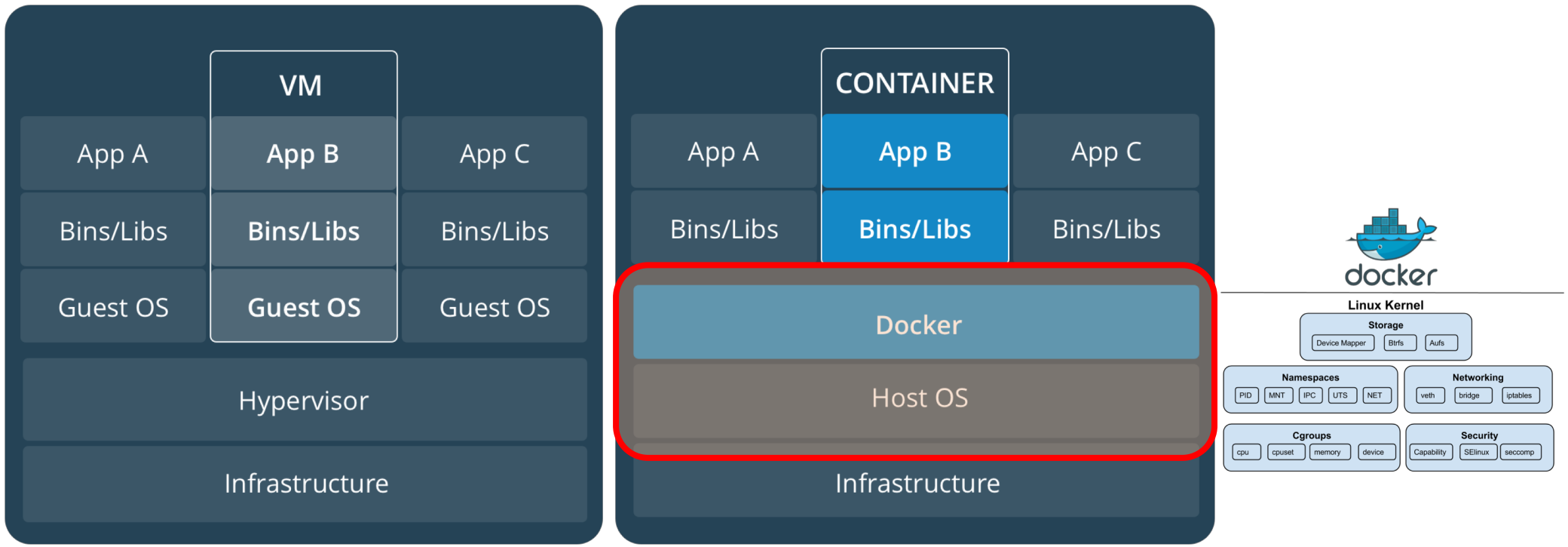
Capability | SElinux | seccomp

Docker가 좋은 이유 #2

가볍다
(lightweight)

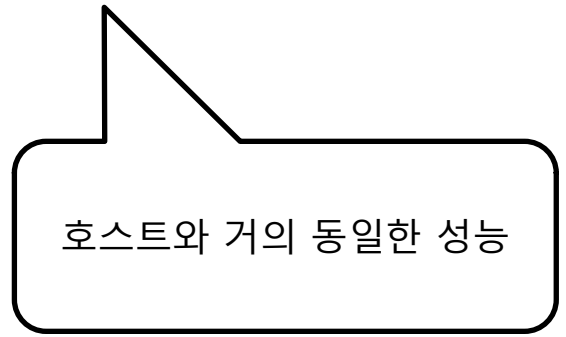

VM
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
CONTAINER
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Host OS
Infrastructure


VM
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
CONTAINER
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Host OS
Infrastructure


VM
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
CONTAINER
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Host OS
Infrastructure
docker
Linux Kernel
Storage
Device Mapper
Btrfs
Aufs
Namespaces
PID
MNT
IPC
UTS
NET
Networking
veth
bridge
iptables
Cgroups
cpu
cpuset
memory
device
Security
Capability
SElinux
seccomp

Benchmark

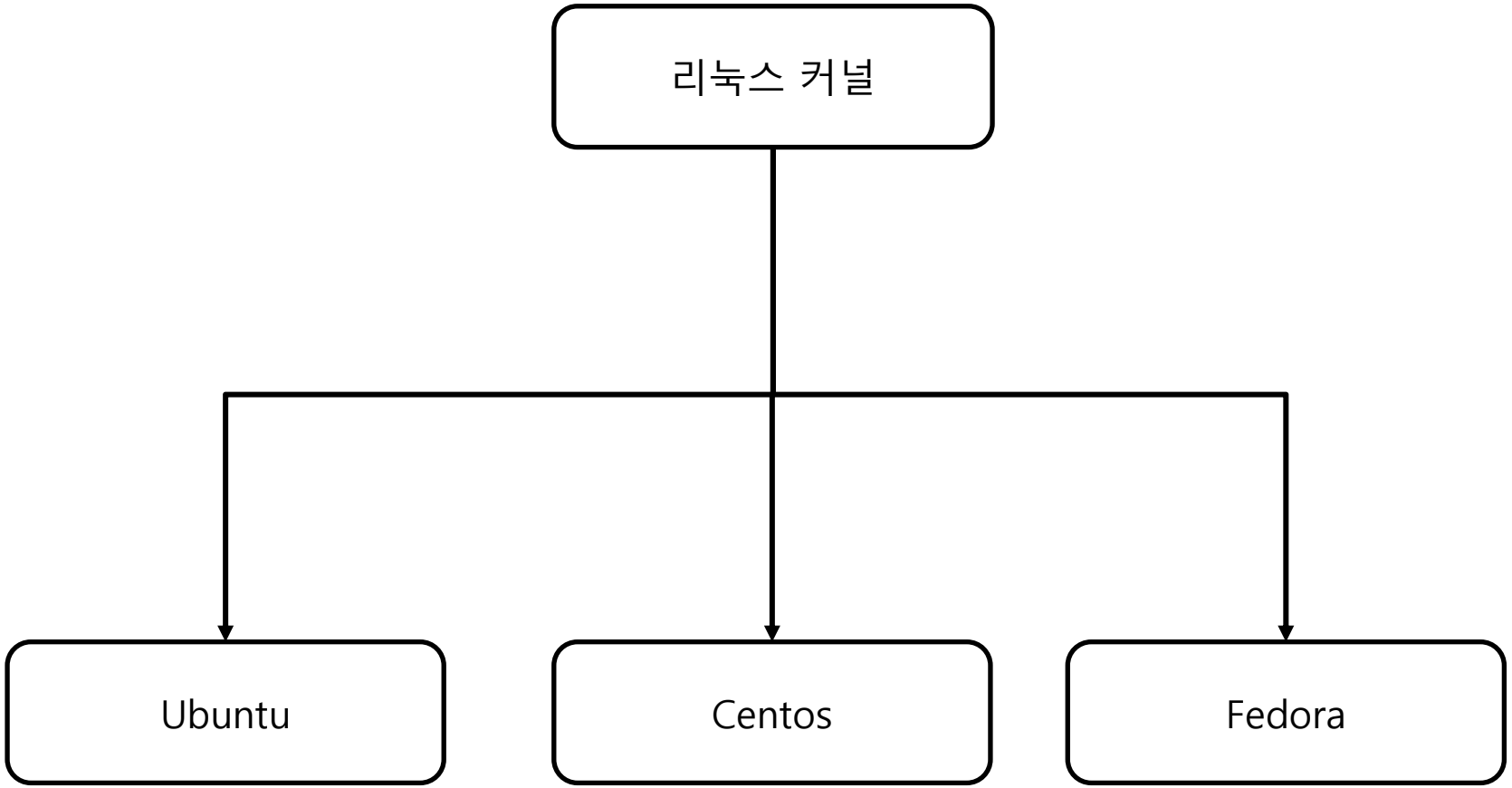
	성능 측정 도구	호스트	Docker
CPU	sysbench	1	0.9945
메모리 쓰기	sysbench	1	0.9826
메모리 읽기	sysbench	1	1.0025
디스크 I/O	dd	1	0.9811
네트워크	iperf	1	0.9626

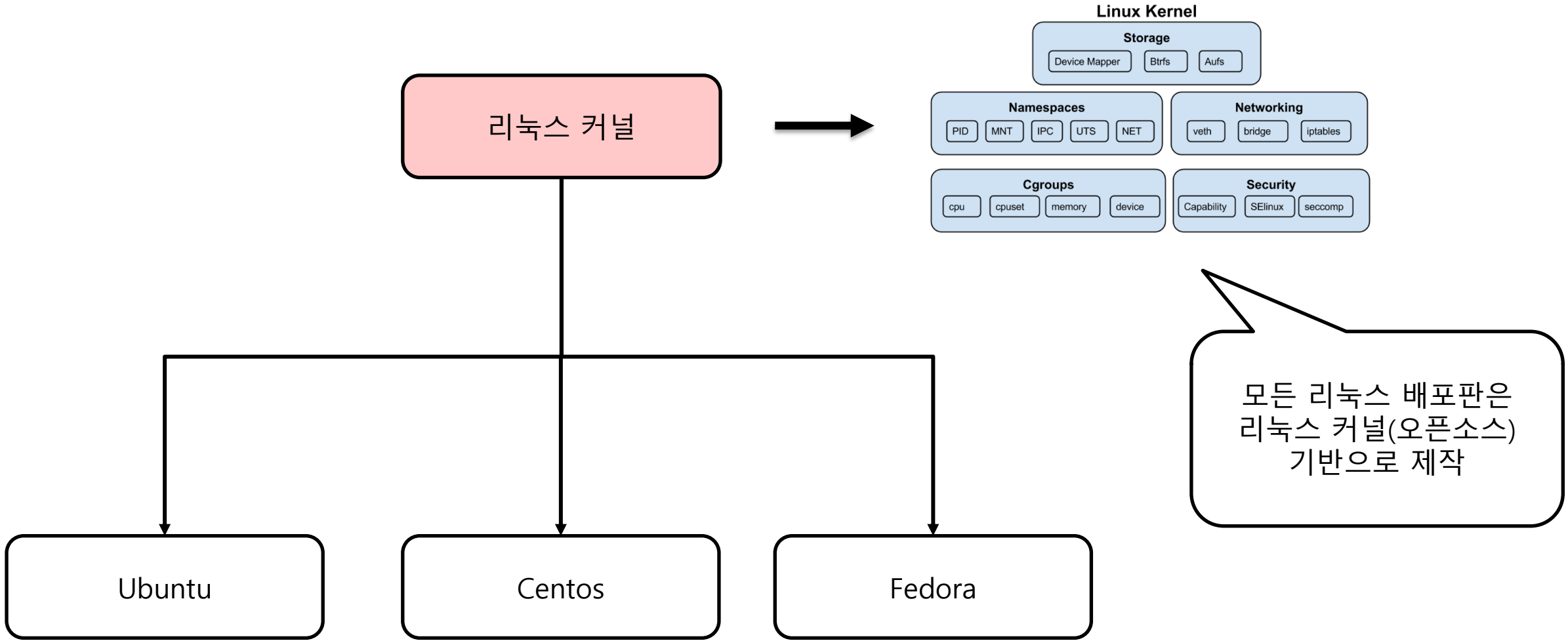


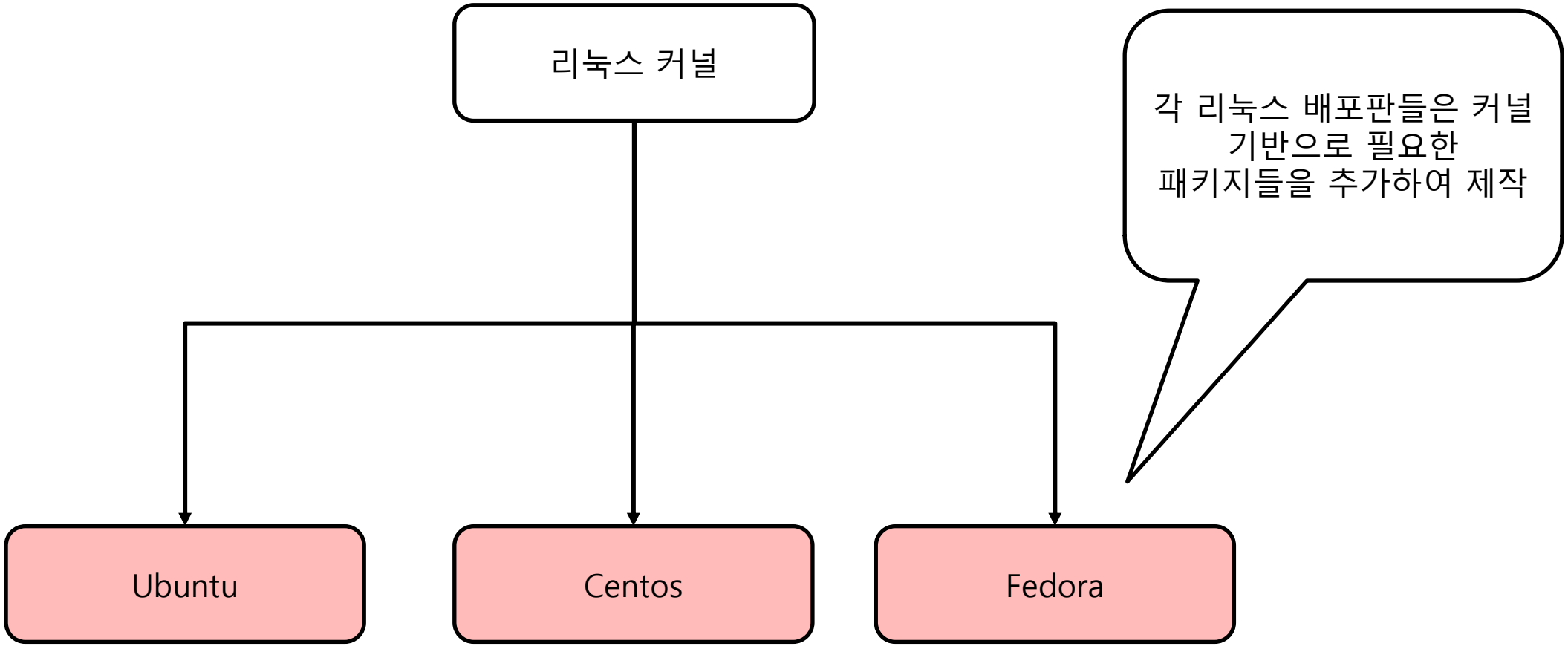
Docker가 좋은 이유 #3

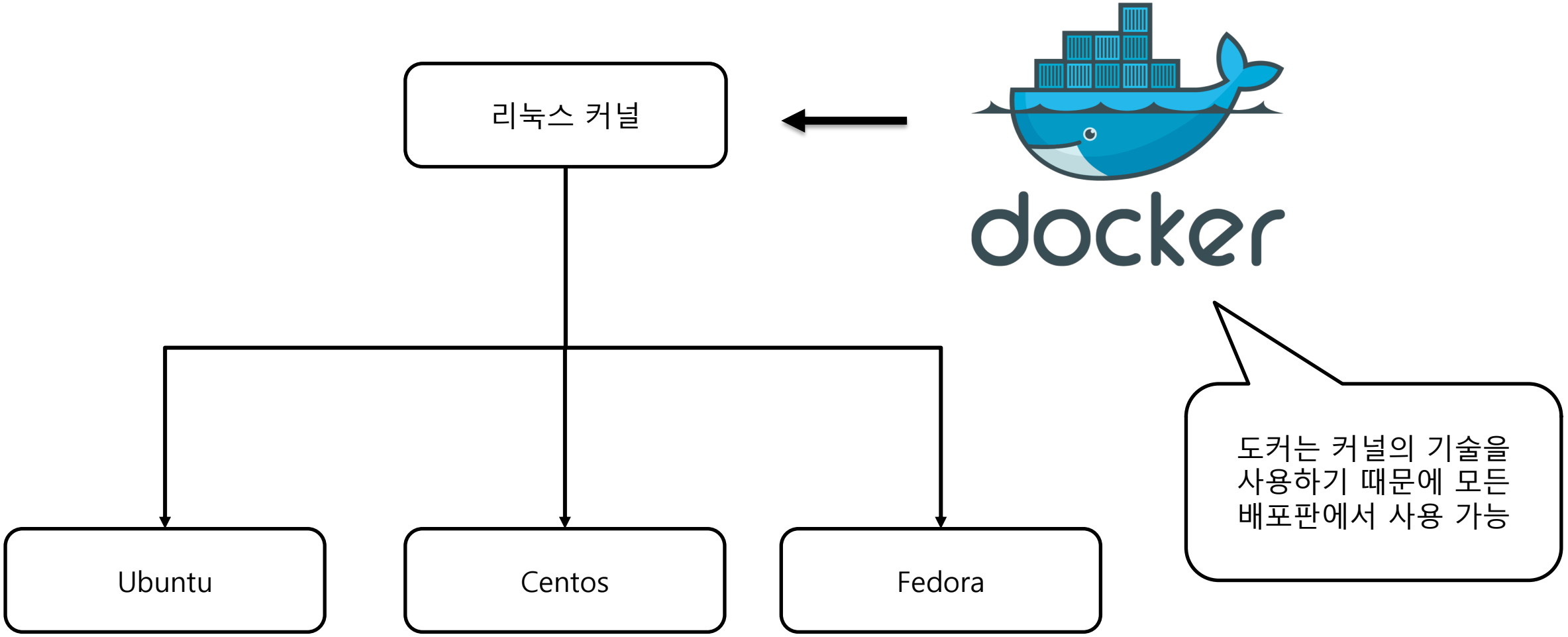
이식성
(Portability)

리눅스 기반의 호스트라면
동일한 환경 보장


리눅스 커널
Ubuntu
Centos
Fedora


리눅스 커널
Linux Kernel
Storage
Device Mapper
Btrfs
Aufs
Namespaces
PID
MNT
IPC
UTS
NET
Networking
veth
bridge
iptables
Cgroups
cpu
cpuset
memory
device
Security
Capability
SElinux
seccomp
모든 리눅스 배포판은
리눅스 커널(오픈소스)
기반으로 제작
Ubuntu
Centos
Fedora


리눅스 커널
각 리눅스 배포판들은 커널
기반으로 필요한
패키지들을 추가하여 제작
Ubuntu
Centos
Fedora

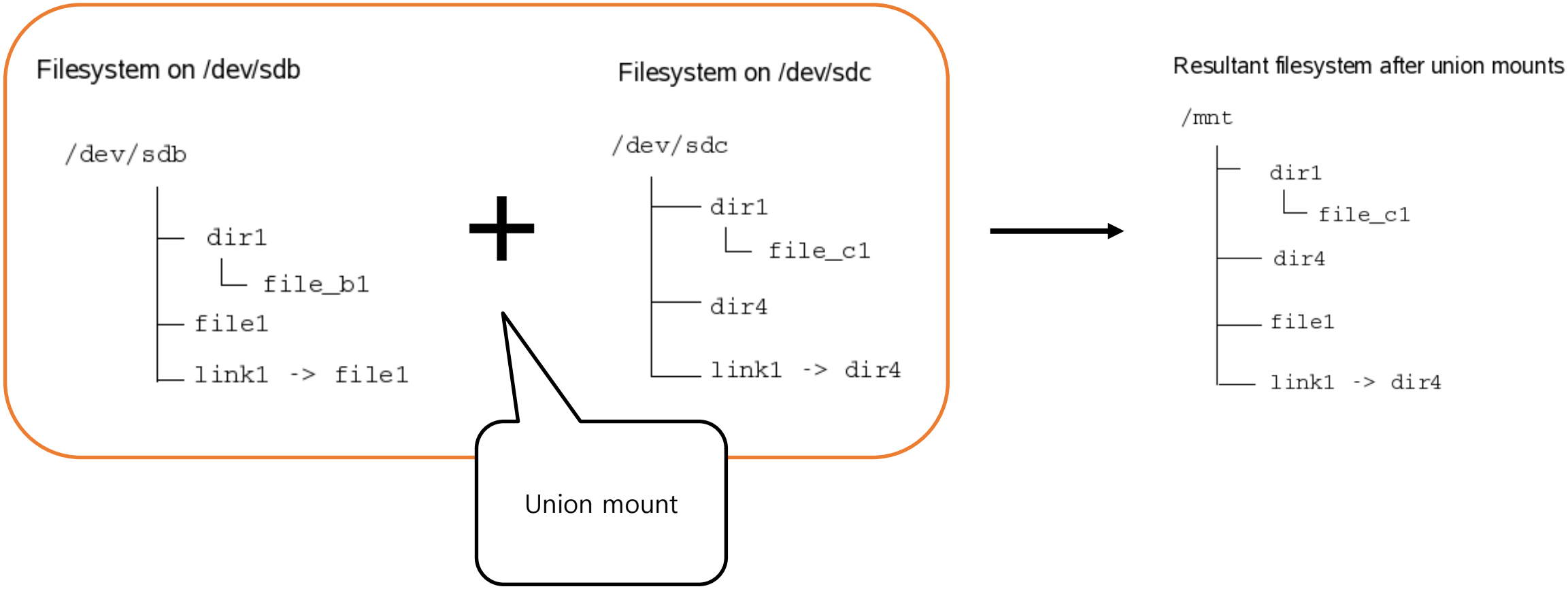

리눅스 커널
docker
Ubuntu
Centos
Fedora
도커는 커널의 기술을
사용하기 때문에 모든
배포판에서 사용 가능

파일시스템은 주의가 필요

AUFS (Advanced multi-layered Unification File System)

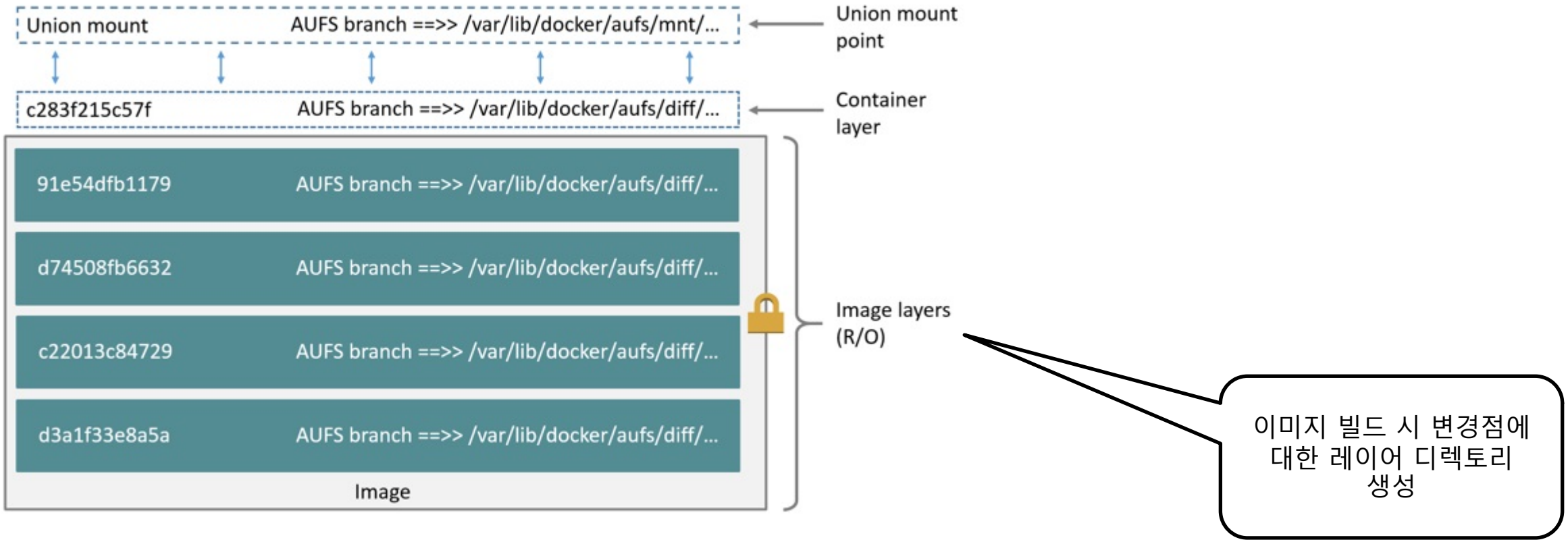
- Union File System
- Ubuntu 도커 이미지, 레이어를 관리
- Linux 커널 버전 4.0 이상이고, 최신버전의 도커(v18.06 이상)를 사용하는 경우 overay2 스토리지 드라이버 사용 권장

Union File System

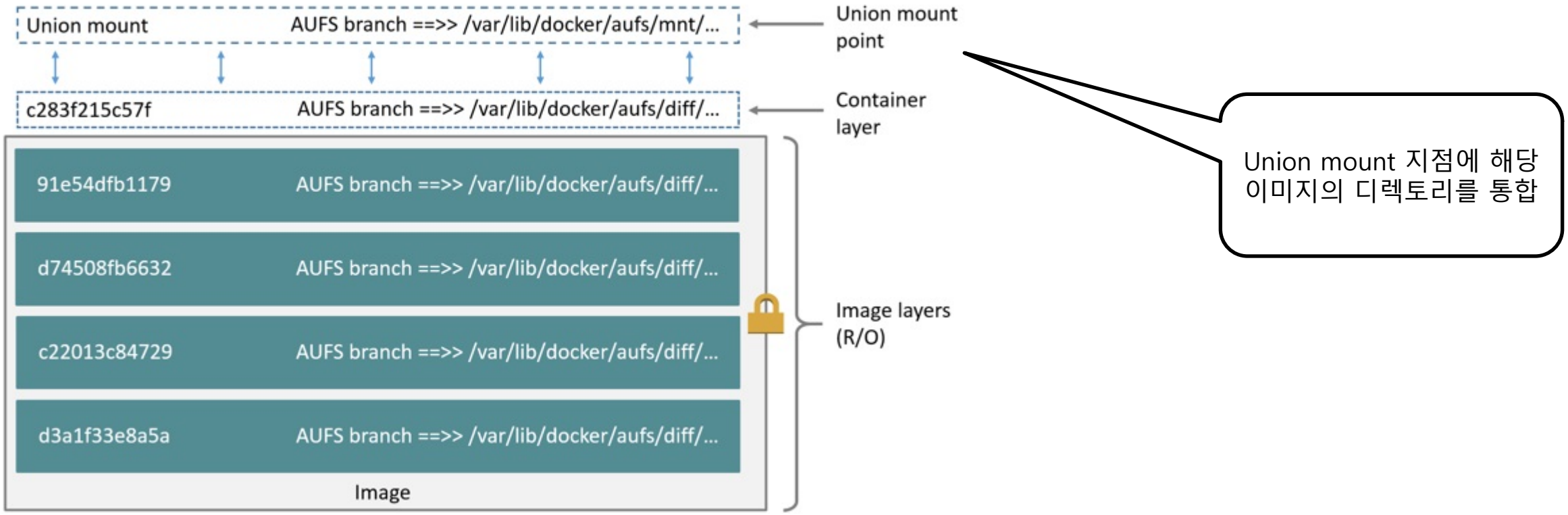


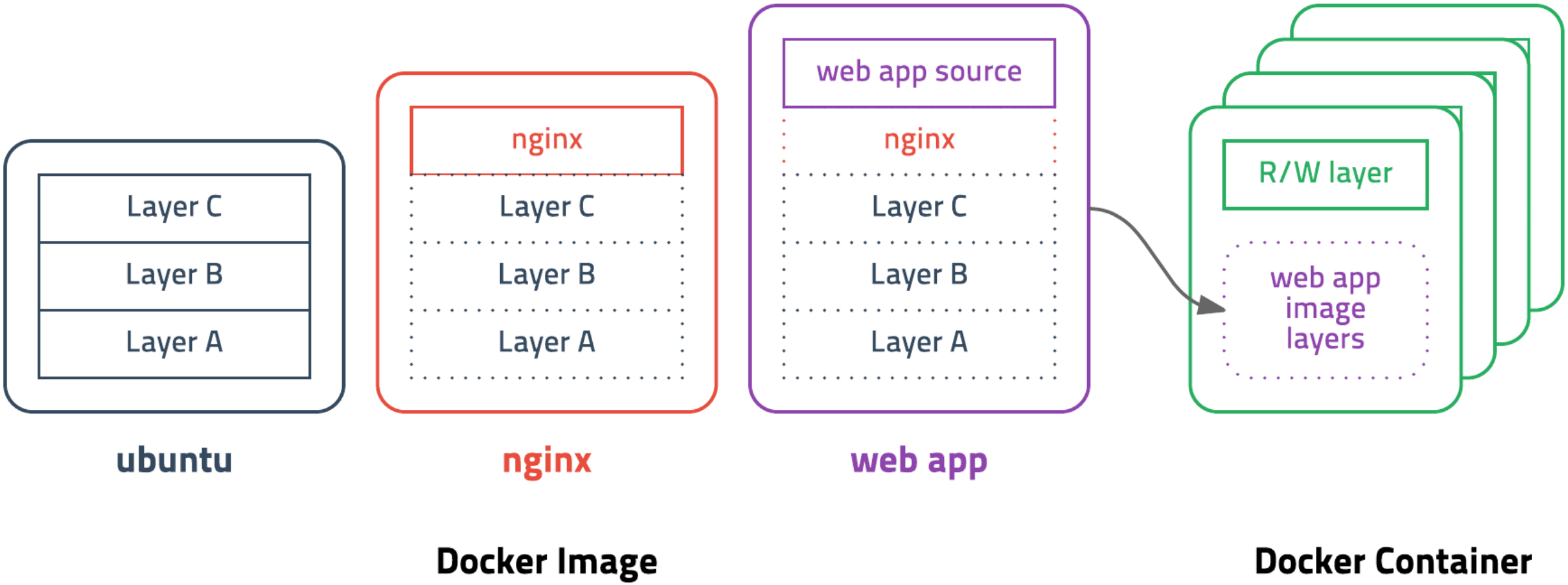
출저 : https://zetawiki.com/wiki/%EC%9C%A0%EB%8B%88%EC%98%A8_%EB%A7%88%EC%9A%B4%ED%8A%B8

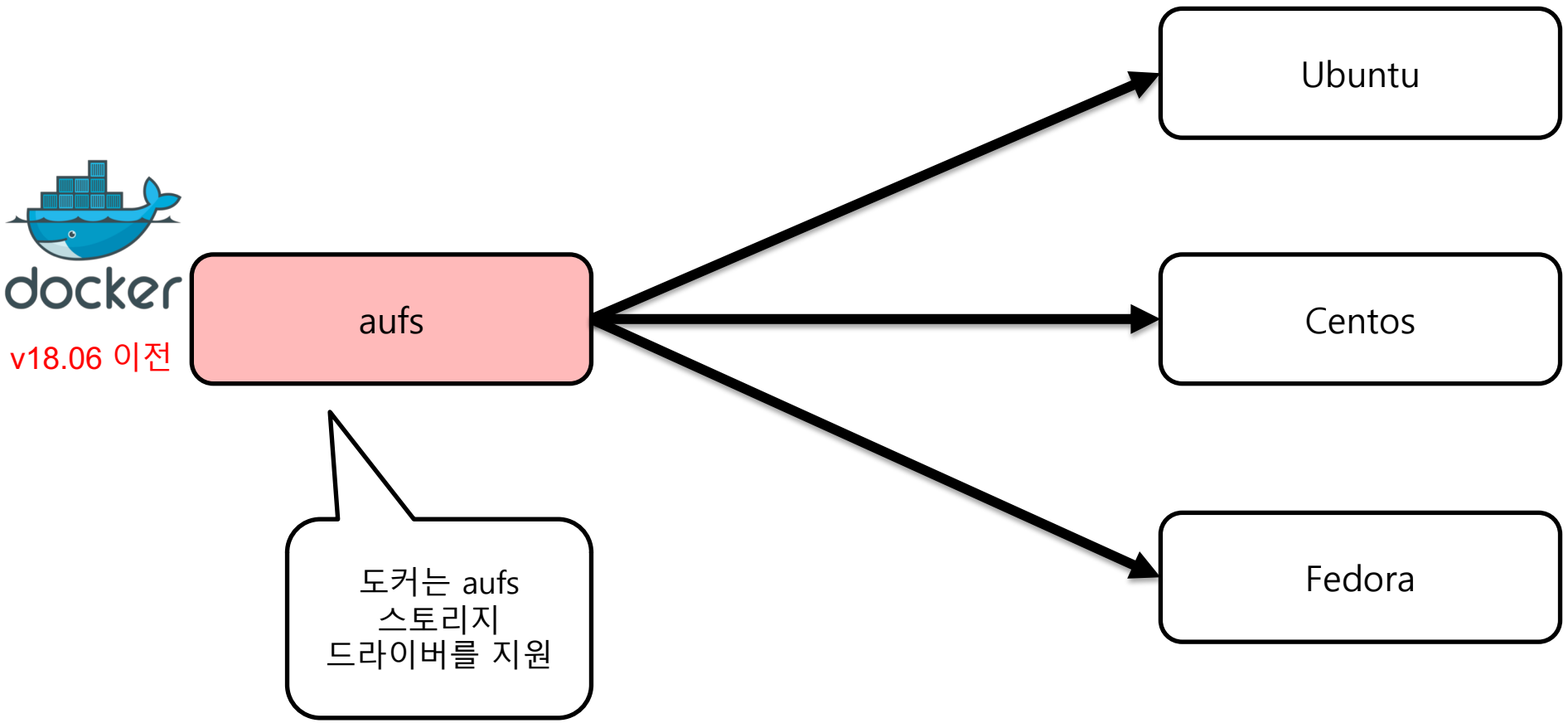
AUFS (Advanced multi-layered Unification File System)

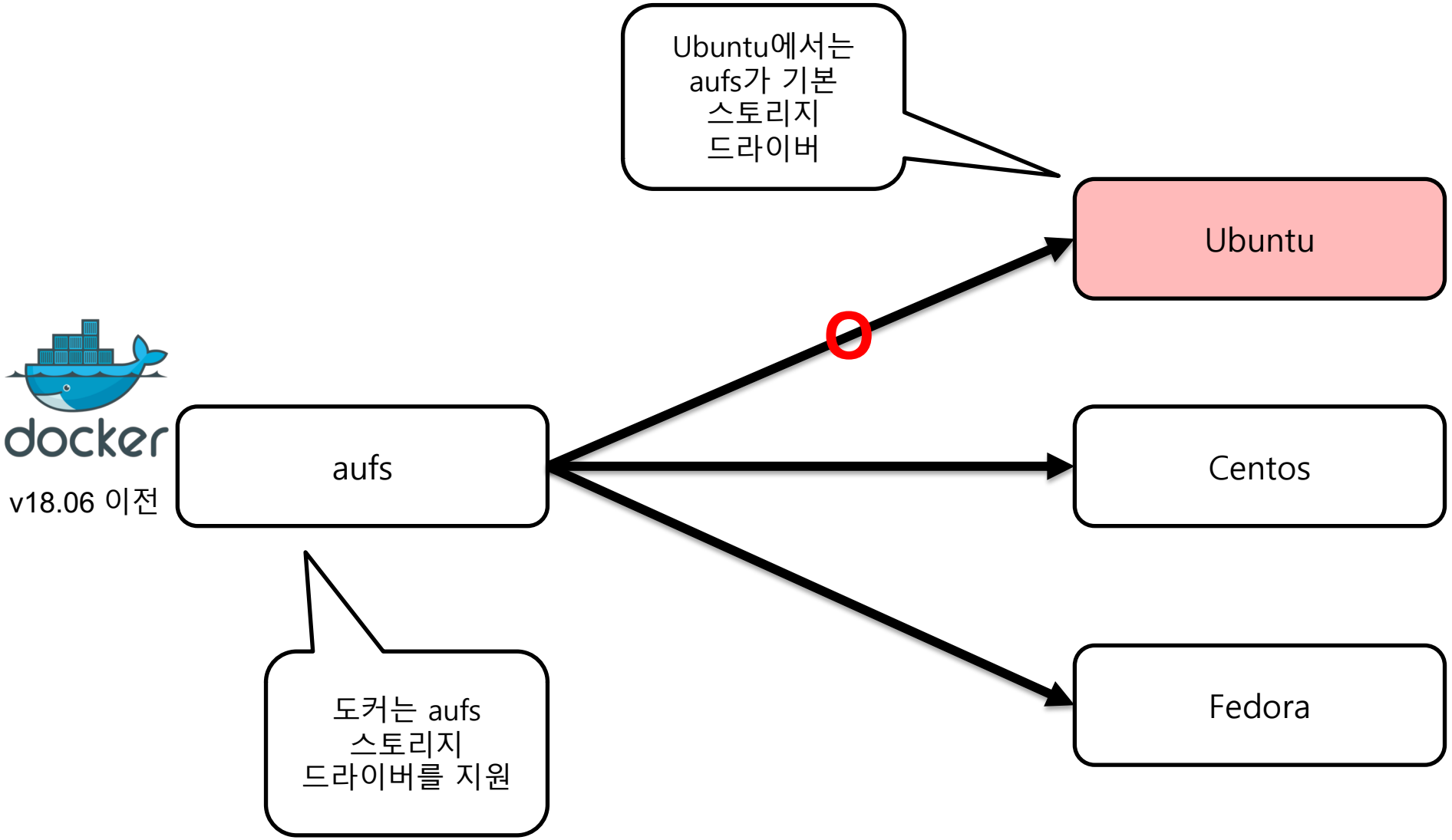


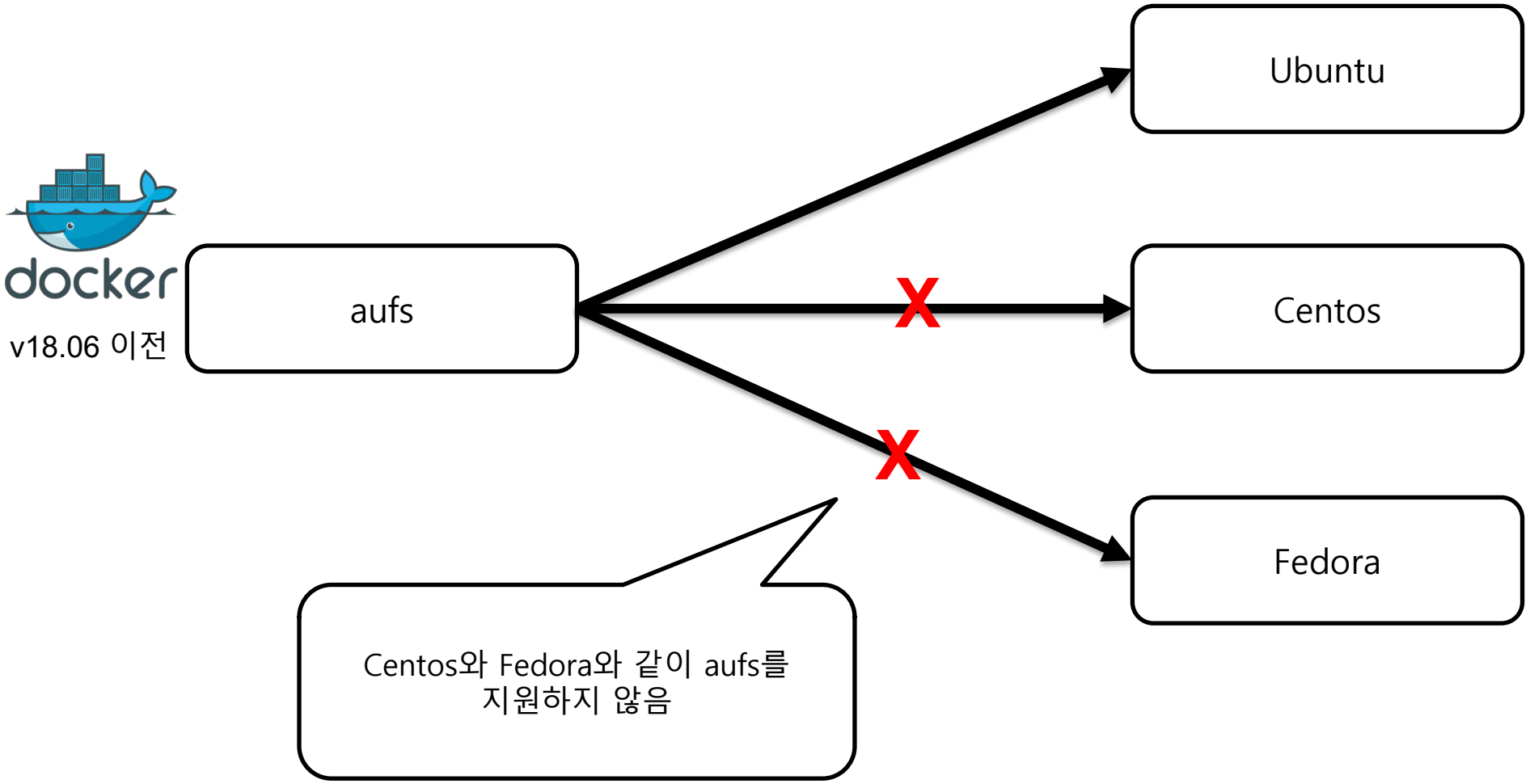
AUFS (Advanced multi-layered Unification File System)

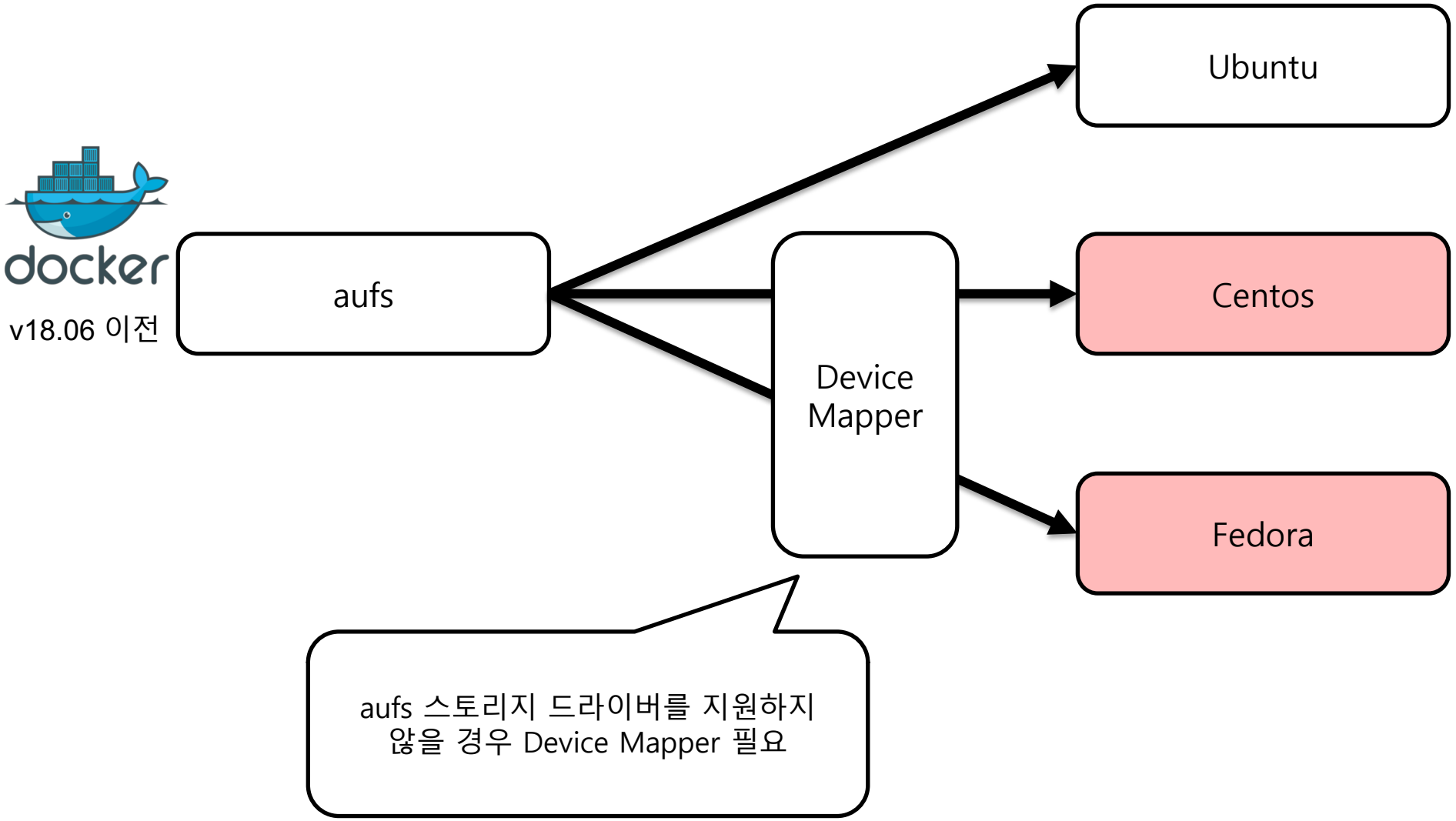


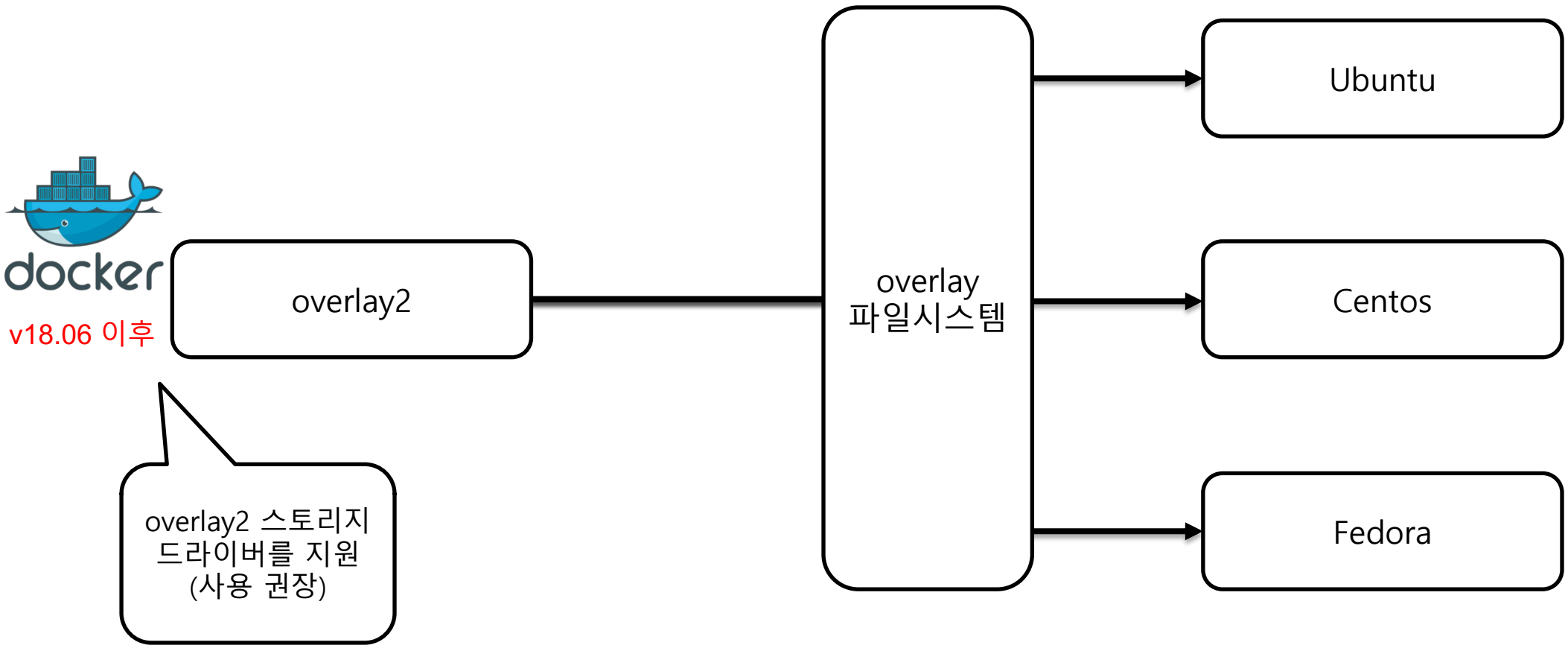

Layer C
Layer B
Layer A
ubuntu
nginx
Layer C
Layer B
Layer A
nginx
web app source
nginx
Layer C
Layer B
Layer A
web app
R/W layer
web app image layers
Docker Image
Docker Container

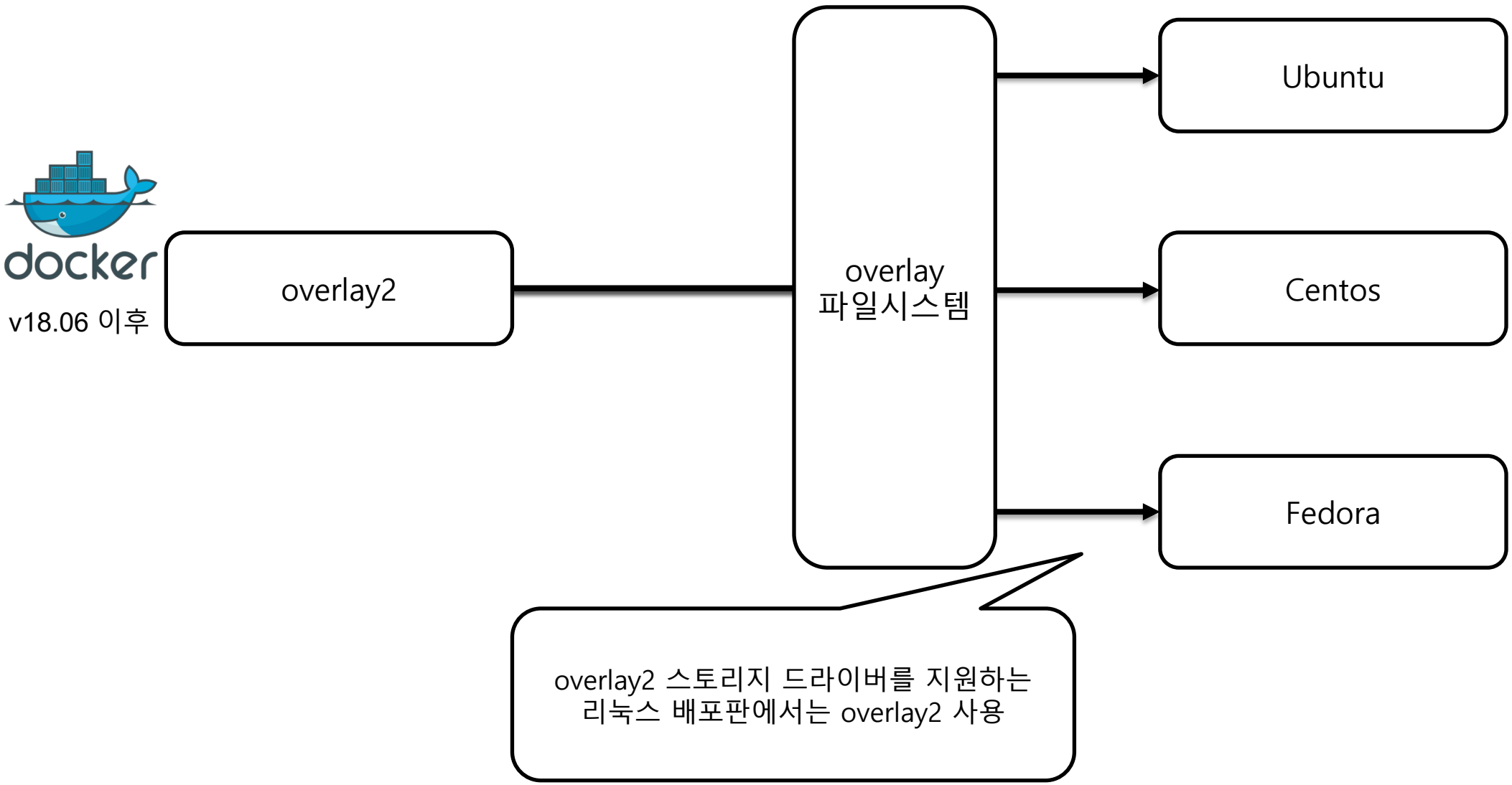

docker
v18.06 이전
aufs
Ubuntu
Centos
Fedora
도커는 aufs
스토리지
드라이버를 지원


Ubuntu에서는
aufs가 기본
스토리지
드라이버
Ubuntu
O
docker
v18.06 이전
aufs
Centos
Fedora
도커는 aufs
스토리지
드라이버를 지원


docker
v18.06 이전
aufs
Ubuntu
Centos
Fedora
X
X
Centos와 Fedora와 같이 aufs를
지원하지 않음


docker
v18.06 이전
aufs
Device Mapper
Ubuntu
Centos
Fedora
aufs 스토리지 드라이버를 지원하지 않을 경우 Device Mapper 필요


docker
v18.06 이후
overlay2
overlay2 스토리지 드라이버를 지원 (사용 권장)
overlay 파일시스템
Ubuntu
Centos
Fedora


docker
v18.06 이후
overlay2
overlay
파일시스템
Ubuntu
Centos
Fedora
overlay2 스토리지 드라이버를 지원하는
리눅스 배포판에서는 overlay2 사용

```
// List of drivers that should be used in an order
priority = "btrfs,zfs,overlay2,aufs,overlay,devicemapper,vfs"

// FsNames maps filesystem id to name of the filesystem.
FsNames = map[FsMagic]string{
        FsMagicAufs:        "aufs",
        FsMagicBtrfs:       "btrfs",
        FsMagicCramfs:      "cramfs",
        FsMagicEcryptfs:    "ecryptfs",
        FsMagicExtfs:       "extfs",
        FsMagicF2fs:        "f2fs",
        FsMagicGPFS:        "gpfs",
        FsMagicJffs2Fs:     "jffs2",
        FsMagicJfs:         "jfs",
        FsMagicNfsFs:       "nfs",
        FsMagicOverlay:     "overlayfs",
        FsMagicRAMFs:       "ramfs",
        FsMagicReiserFs:    "reiserfs",
        FsMagicSmbFs:       "smb",
        FsMagicSquashFs:    "squashfs",
        FsMagicTmpFs:       "tmpfs",
        FsMagicUnsupported: "unsupported",
        FsMagicVxFS:        "vxfs",
        FsMagicXfs:         "xfs",
        FsMagicZfs:         "zfs",
}
```

[도커가 지원하는 스토리지 드라이버]

OS Distribution (x86_64)	Enterprise Engine	Storage Driver
RHEL 7.4	18.09.x	overlay2, devicemapper
RHEL 7.5	18.09.x	overlay2
RHEL 7.6	18.09.x	overlay2
SLES 12 SP2	18.09.x	overlay2, btrfs
SLES 12 SP3	18.09.x	overlay2,btrfs
Ubuntu 16.04	18.09.x	overlay2, aufs
Ubuntu 18.04	18.09.x	overlay2,aufs
CentOS 7.6	18.09.x	overlay2, devicemapper
Windows Server 2016	18.09.x	windowsfilter
Windows Server, version 1709	18.09.x	windowsfilter
Windows Server, version 1803	18.09.x	windowsfilter
Windows Server, version 2019	18.09.x	windowsfilter

[OS별 지원하는 스토리지 드라이버]

```
// List of drivers that should be used in an order
priority = "btrfs,zfs,overlay2,aufs,overlay,devicemapper,vfs"

// FsNames maps filesystem id to name of the filesystem.
FsNames = map[FsMagic]string{
        FsMagicAufs:        "aufs",
        FsMagicBtrfs:       "btrfs",
        FsMagicCramfs:      "cramfs",
        FsMagicEcryptfs:    "ecryptfs",
        FsMagicExtfs:       "extfs",
        FsMagicF2fs:        "f2fs",
        FsMagicGPFS:        "gpfs",
        FsMagicJffs2Fs:     "jffs2",
        FsMagicJfs:         "jfs",
        FsMagicNfsFs:       "nfs",
        FsMagicOverlay:     "overlayfs",
        FsMagicRAMFs:       "ramfs",
        FsMagicReiserFs:    "reiserfs",
        FsMagicSmbFs:       "smb",
        FsMagicSquashFs:    "squashfs",
        FsMagicTmpFs:       "tmpfs",
        FsMagicUnsupported: "unsupported",
        FsMagicVxFS:        "vxfs",
        FsMagicXfs:         "xfs",
        FsMagicZfs:         "zfs",
}
```

도커가 지원하는 파일시스템 우선 순위

OS Distribution (x86_64)	Enterprise	Storage Driver
RHEL		
RHEL		
SLES 12 SP2	18.09.x	overlay2, btrfs
SLES 12 SP3	18.09.x	overlay2,btrfs
Ubuntu 16.04	18.09.x	overlay2, aufs
Ubuntu 18.04	18.09.x	overlay2,aufs
CentOS 7.6	18.09.x	overlay2, devicemapper
Windows Server 2016	18.09.x	windowsfilter
Windows Server, version 1709	18.09.x	windowsfilter
Windows Server, version 1803	18.09.x	windowsfilter
Windows Server, version 2019	18.09.x	windowsfilter

[도커가 지원하는 스토리지 드라이버]

[OS별 지원하는 스토리지 드라이버]

```
// List of drivers that should be used in an order
priority = "btrfs,zfs,overlay2,aufs,overlay,devicemapper,vfs"

// FsNames maps filesystem id to name of the filesystem.
FsNames = map[FsMagic]string{
        FsMagicAufs:        "aufs",
        FsMagicBtrfs:       "btrfs",
        FsMagicCramfs:      "cramfs",
        FsMagicEcryptfs:    "ecryptfs",
        FsMagicExtfs:       "extfs",
        FsMagicF2fs:        "f2fs",
        FsMagicGPFS:        "gpfs",
        FsMagicJffs2Fs:     "jffs2",
        FsMagicJfs:         "jfs",
        FsMagicNfsFs:       "nfs",
        FsMagicOverlay:     "overlayfs",
        FsMagicRAMFs:       "ramfs",
        FsMagicReiserFs:    "reiserfs",
        FsMagicSmbFs:       "smb",
        FsMagicSquashFs:    "squashfs",
        FsMagicTmpFs:       "tmpfs",
        FsMagicUnsupported: "unsupported",
        FsMagicVxFS:        "vxfs",
        FsMagicXfs:         "xfs",
        FsMagicZfs:         "zfs",
}
```

[도커가 지원하는 스토리지 드라이버]

OS Distribution (x86_64)	Enterprise Engine	Storage Driver
RHEL 7.4	18.09.x	overlay2, devicemapper
RHEL 7.5	18.09.x	overlay2
RHEL 7.6	18.09.x	overlay2
SLES 12 SP2	18.09.x	overlay2, btrfs
SLES 12 SP3	18.09.x	overlay2,btrfs
Ubuntu 16.04	18.09.x	overlay2, aufs
Ubuntu 18.04	18.09.x	overlay2,aufs
CentOS 7.6	18.09.x	overlay2, devicemapper
Windows Server 2016	18.09.x	windowsfilter
Windows Server, version 1709	18.09.x	windowsfilter
Windows Server, version 1803	18.09.x	windowsfilter
Windows Server, version 2019	18.09.x	windowsfilter

[OS별 지원하는 스토리지 드라이버]

Red Hat 계열은
overlay2와
devicemapper 지원

```
// List of drivers that should be used in an order
priority = "btrfs,zfs,overlay2,aufs,overlay,devicemapper,vfs"

// FsNames maps filesystem id to name of the filesystem.
FsNames = map[FsMagic]string{
        FsMagicAufs:        "aufs",
        FsMagicBtrfs:       "btrfs",
        FsMagicCramfs:      "cramfs",
        FsMagicEcryptfs:    "ecryptfs",
        FsMagicExtfs:       "extfs",
        FsMagicF2fs:        "f2fs",
        FsMagicGPFS:        "gpfs",
        FsMagicJffs2Fs:     "jffs2",
        FsMagicJfs:         "jfs",
        FsMagicNfsFs:       "nfs",
        FsMagicOverlay:     "overlayfs",
        FsMagicRAMFs:       "ramfs",
        FsMagicReiserFs:    "reiserfs",
        FsMagicSmbFs:       "smb",
        FsMagicSquashFs:    "squashfs",
        FsMagicTmpFs:       "tmpfs",
        FsMagicUnsupported: "unsupported",
        FsMagicVxFS:        "vxfs",
        FsMagicXfs:         "xfs",
        FsMagicZfs:         "zfs",
}
```

[도커가 지원하는 스토리지 드라이버]

OS Distribution (x86_64)	Enterprise Engine	Storage Driver
RHEL 7.4	18.09.x	overlay2, devicemapper
RHEL 7.5	18.09.x	overlay2
RHEL 7.6	18.09.x	overlay2
SLES 12 SP2	18.09.x	overlay2, btrfs
SLES 12 SP3	18.09.x	overlay2,btrfs
Ubuntu 16.04	18.09.x	overlay2, aufs
Ubuntu 18.04	18.09.x	overlay2,aufs
CentOS 7.6	18.09.x	overlay2, devicemapper
Windows Server 2016	18.09.x	windowsfilter
Windows Server, version 1709	18.09.x	windowsfilter
Windows Server, version 1803	18.09.x	windowsfilter
Windows Server, version 2019	18.09.x	windowsfilter

Ubuntu는 overlay2와 aufs 지원

[OS별 지원하는 스토리지 드라이버]

```
// List of drivers that should be used in an order
priority = "btrfs,zfs,overlay2,aufs,overlay,devicemapper,vfs"

// FsNames maps filesystem id to name of the filesystem.
FsNames = map[FsMagic]string{
        FsMagicAufs:        "aufs",
        FsMagicBtrfs:       "btrfs",
        FsMagicCramfs:      "cramfs",
        FsMagicEcryptfs:    "ecryptfs",
        FsMagicExtfs:       "extfs",
        FsMagicF2fs:        "f2fs",
        FsMagicGPFS:        "gpfs",
        FsMagicJffs2Fs:     "jffs2",
        FsMagicJfs:         "jfs",
        FsMagicNfsFs:       "nfs",
        FsMagicOverlay:     "overlayfs",
        FsMagicRAMFs:       "ramfs",
        FsMagicReiserFs:    "reiserfs",
        FsMagicSmbFs:       "smb",
        FsMagicSquashFs:    "squashfs",
        FsMagicTmpFs:       "tmpfs",
        FsMagicUnsupported: "unsupported",
        FsMagicVxFS:        "vxfs",
        FsMagicXfs:         "xfs",
        FsMagicZfs:         "zfs",
}
```

[도커가 지원하는 스토리지 드라이버]

OS Distribution (x86_64)	Enterprise Engine	Storage Driver
RHEL 7.4	18.09.x	overlay2, devicemapper
RHEL 7.5	18.09.x	overlay2
RHEL 7.6	18.09.x	overlay2
SLES 12 SP2	18.09.x	overlay2, btrfs
SLES 12 SP3	18.09.x	overlay2,btrfs
Ubuntu 16.04	18.09.x	overlay2, aufs
Ubuntu 18.04	18.09.x	overlay2,aufs
CentOS 7.6	18.09.x	overlay2, devicemapper
Windows Server 2016	18.09.x	windowsfilter
Windows Server, version 1709	18.09.x	windowsfilter
Windows Server, version 1803	18.09.x	windowsfilter
Windows Server, version 2019	18.09.x	windowsfilter

[OS별 지원하는 스토리지 드라이버]

CentOS는 overlay2와 devicemapper 지원

```
// List of drivers that should be used in an order
priority = "btrfs,zfs,overlay2,aufs,overlay,devicemapper,vfs"

// FsNames maps filesystem id to name of the filesystem.
FsNames = map[FsMagic]string{
        FsMagicAufs:        "aufs",
        FsMagicBtrfs:       "btrfs",
        FsMagicCramfs:      "cramfs",
        FsMagicEcryptfs:    "ecryptfs",
        FsMagicExtfs:       "extfs",
        FsMagicF2fs:        "f2fs",
        FsMagicGPFS:        "gpfs",
        FsMagicJffs2Fs:     "jffs2",
        FsMagicJfs:         "jfs",
        FsMagicNfsFs:       "nfs",
        FsMagicOverlay:     "overlayfs",
        FsMagicRAMFs:       "ramfs",
        FsMagicReiserFs:    "reiserfs",
        FsMagicSmbFs:       "smb",
        FsMagicSquashFs:    "squashfs",
        FsMagicTmpFs:       "tmpfs",
        FsMagicUnsupported: "unsupported",
        FsMagicVxFS:        "vxfs",
        FsMagicXfs:         "xfs",
        FsMagicZfs:         "zfs",
}
```

[도커가 지원하는 스토리지 드라이버]

OS Distribution (x86_64)	Enterprise Engine	Storage Driver
RHEL 7.4	18.09.x	overlay2, devicemapper
RHEL 7.5	18.09.x	overlay2
RHEL 7.6	18.09.x	overlay2
SLES 12 SP2	18.09.x	overlay2, btrfs
SLES 12 SP3	18.09.x	overlay2,btrfs
Ubuntu 16.04	18.09.x	overlay2, aufs
Ubuntu 18.04	18.09.x	overlay2,aufs
CentOS 7.6	18.09.x	overlay2, devicemapper
Windows Server 2016	18.09.x	windowsfilter
Windows Server, version 1709	18.09.x	windowsfilter
Windows Server, version 1803	18.09.x	windowsfilter
Windows Server, version 2019	18.09.x	windowsfilter

[OS별 지원하는 스토리지 드라이버]

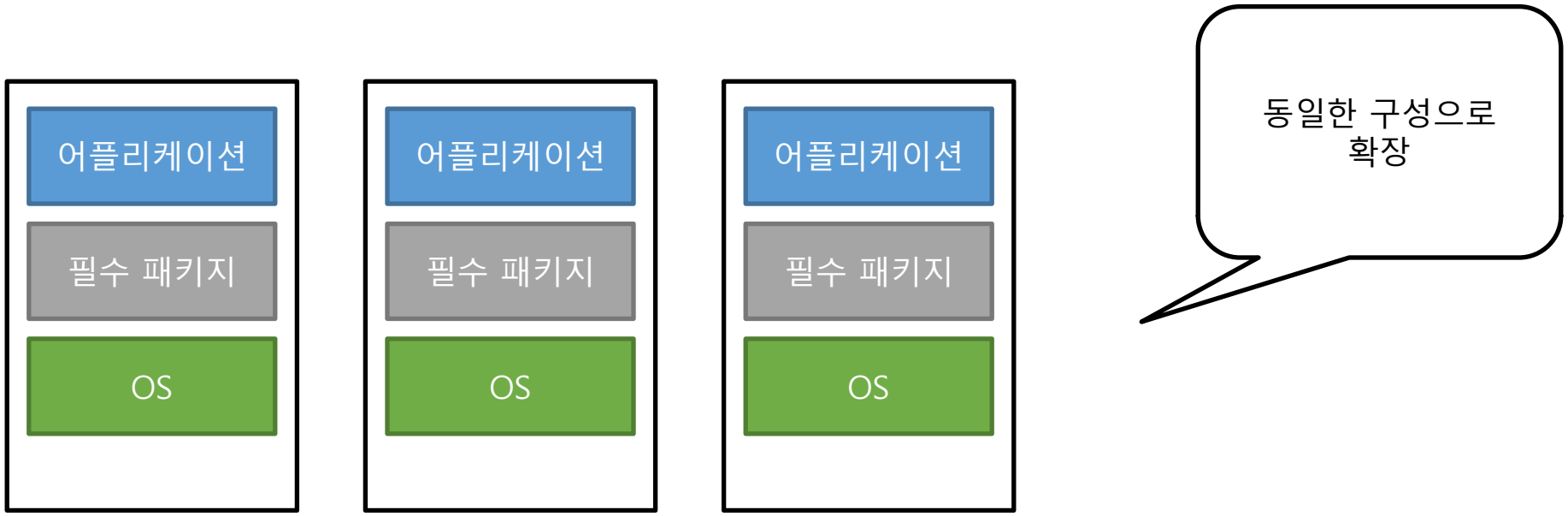
overlay2 지원 버전이 아닌 경우
aufs를 기본 파일시스템으로 사용하는 Ubuntu가 적합

Docker가 좋은 이유 #4

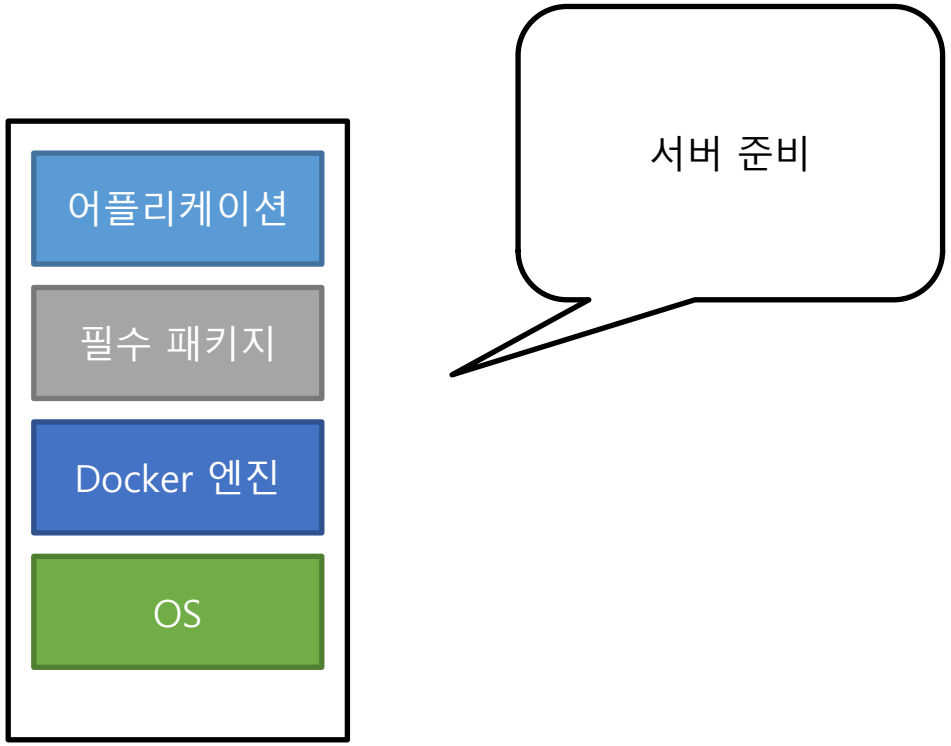
비용 절감

물리 서버 사용하는 경우

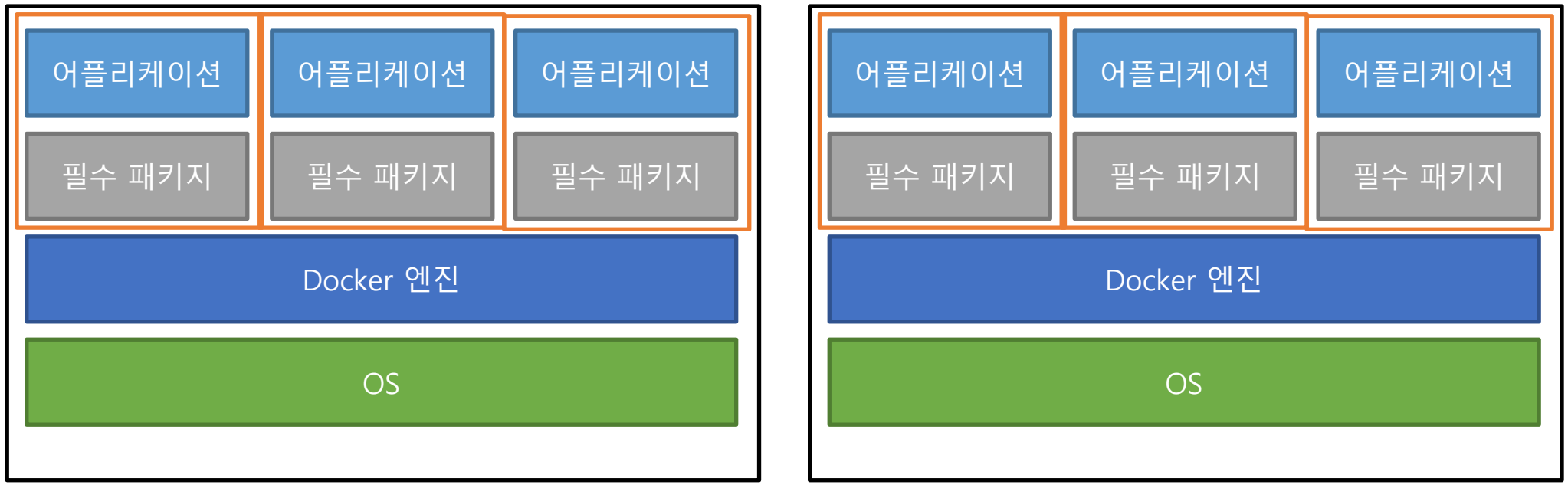
물리 서버 사용하는 경우



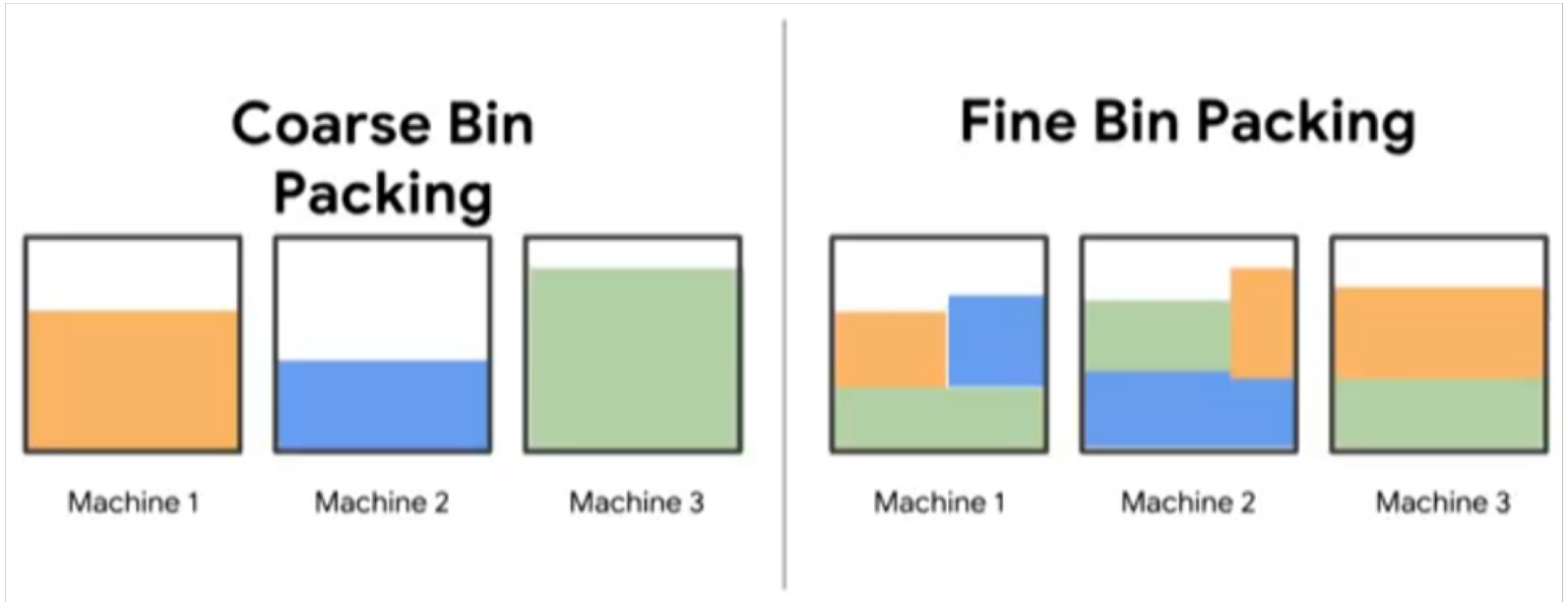
컨테이너를 사용하는 경우



컨테이너를 사용하는 경우



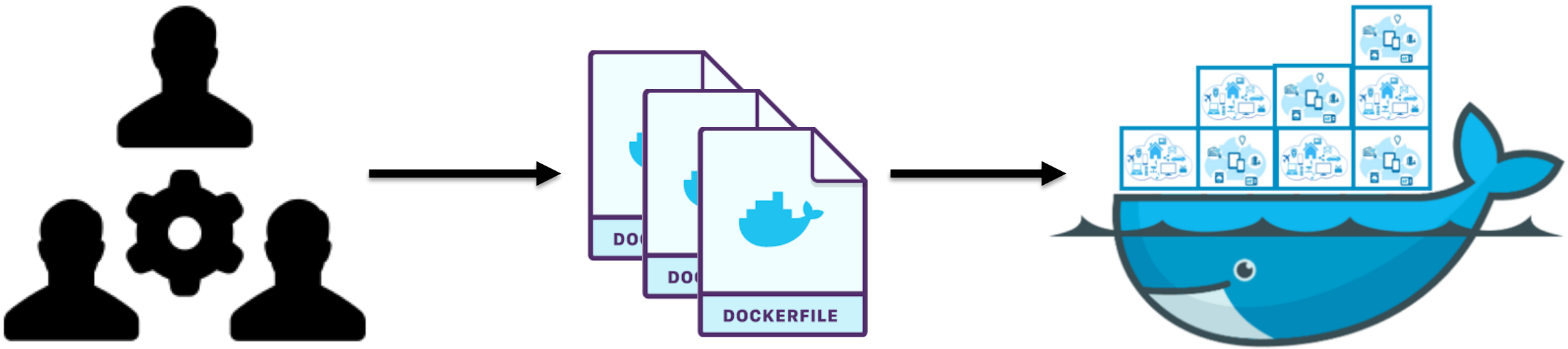
자원을 효율적으로 사용


Coarse Bin Packing
Machine 1
Machine 2
Machine 3
Fine Bin Packing
Machine 1
Machine 2
Machine 3

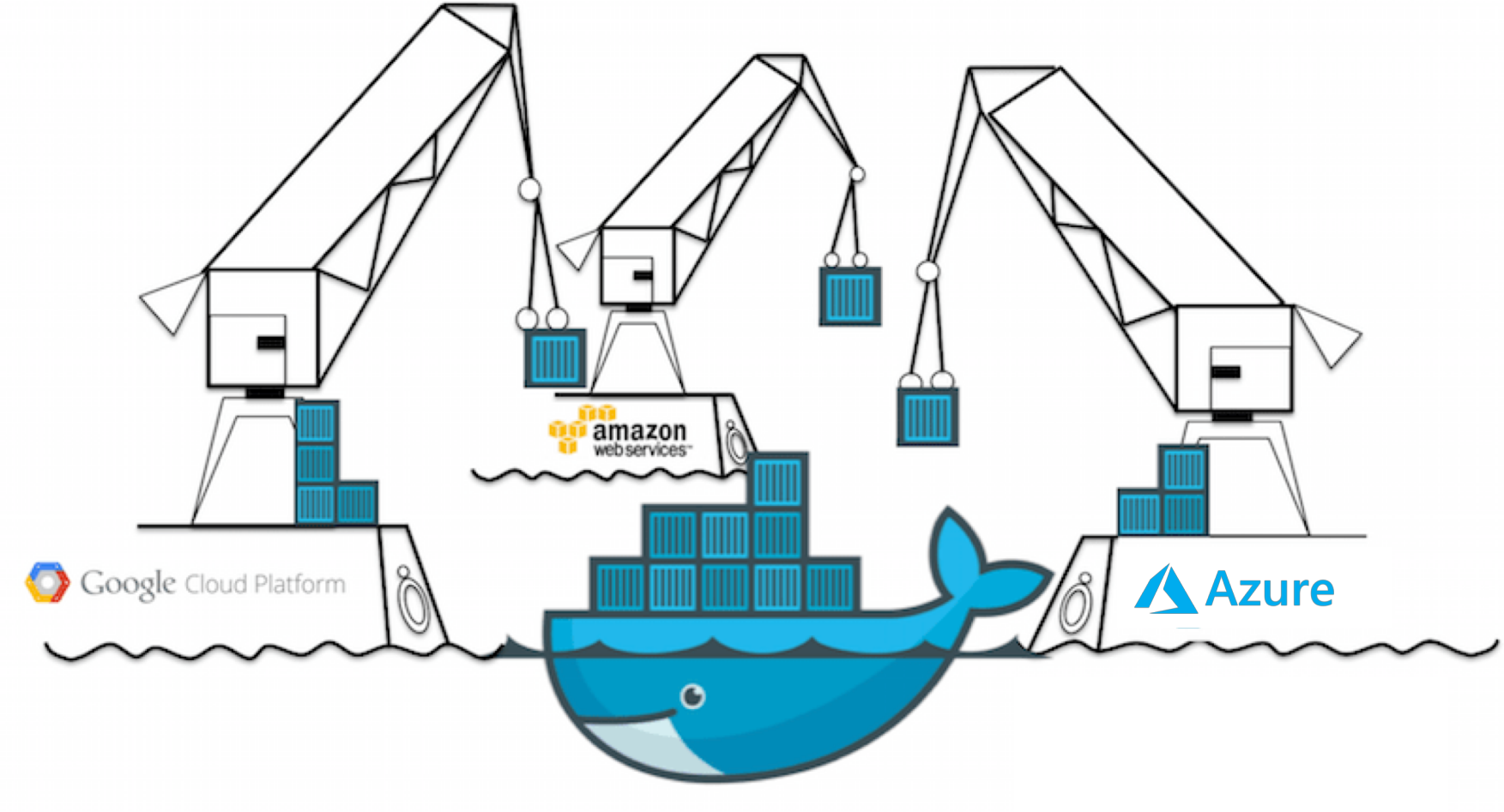
Docker가 좋은 이유 #5

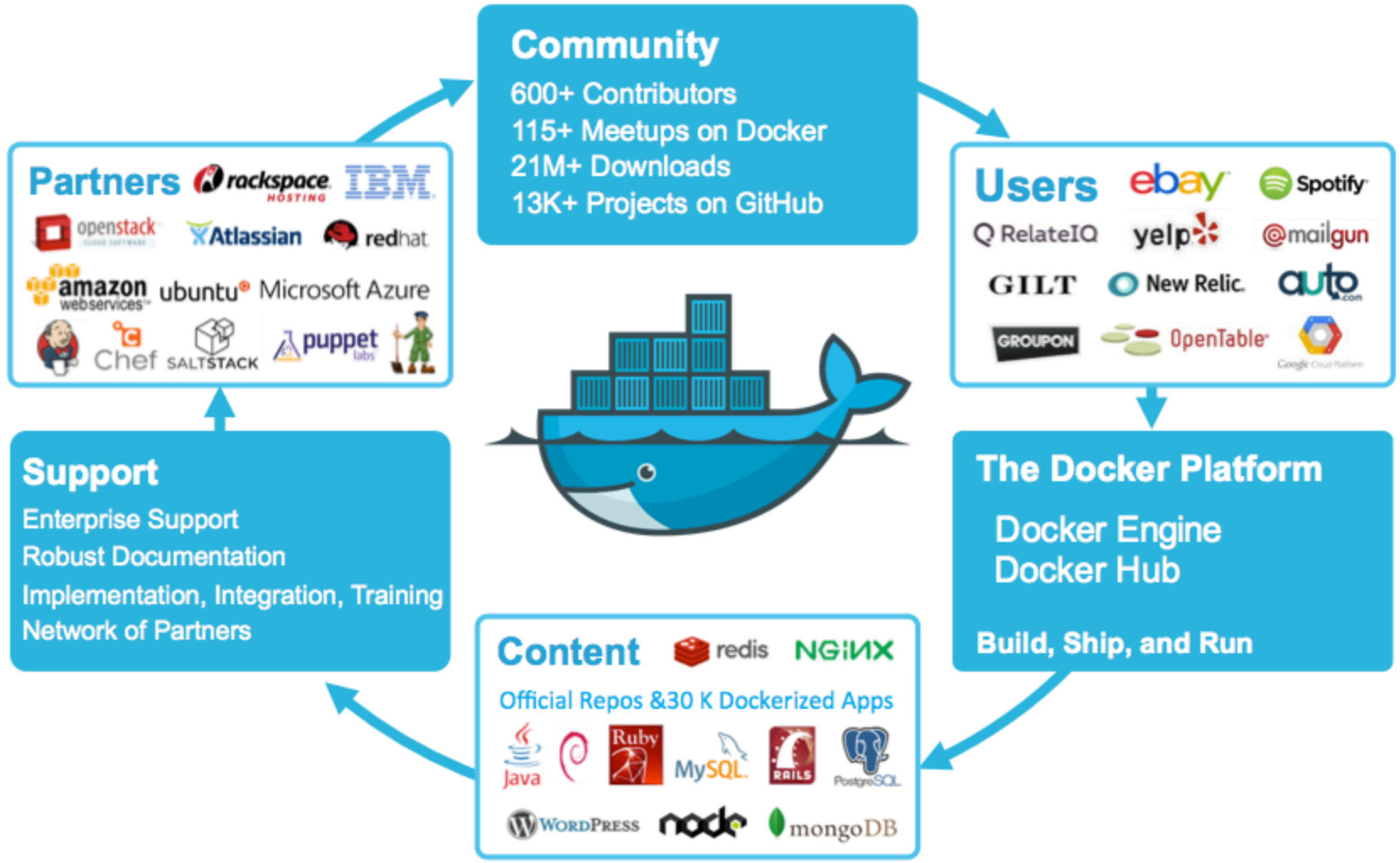
표준화
(Standardized)

협업



도커 기반 서비스 지원




Community
600+ Contributors
115+ Meetups on Docker
21M+ Downloads
13K+ Projects on GitHub
Partners
rackspace HOSTING
IBM
openstack
Atlassian
redhat
amazon webservices
ubuntu
Microsoft Azure
Chef
SALTSTACK
puppet labs
Users
ebay
Spotify
RelateIQ
yelp
mailgun
GILT
New Relic
auto.com
GROUPON
OpenTable
Google Cloud Platform
Support
Enterprise Support
Robust Documentation
Implementation, Integration, Training
Network of Partners
The Docker Platform
Docker Engine
Docker Hub
Build, Ship, and Run
Content
redis
NGINX
Official Repos &30 K Dockerized Apps
Java
Ruby
MySQL
RAILS
PostgreSQL
WordPress
node
mongoDB

docker
검색어

+ 비교

전 세계 ▾ 지난 5년 ▾ 모든 카테고리 ▾ 웹 검색 ▾

시간 흐름에 따른 관심도 변화

Docker 명령어

이미지 관리

- docker images
- docker build
- docker tag
- docker commit
- docker push
- docker rmi

실행

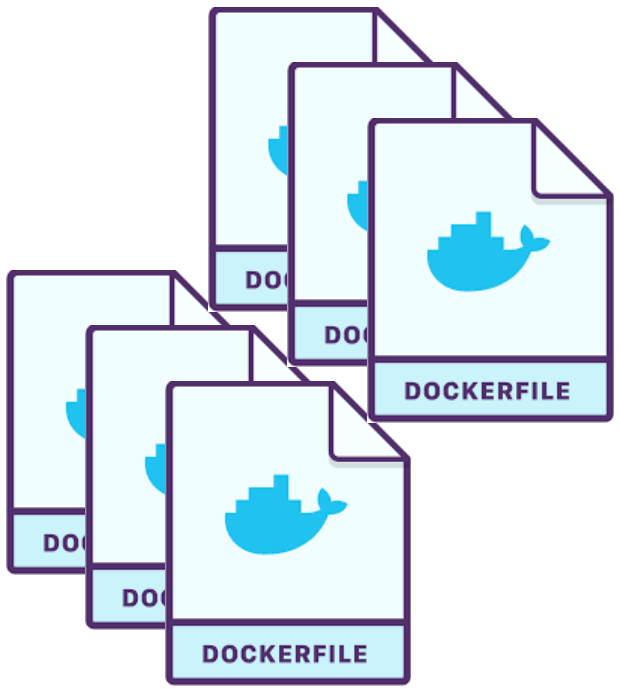
- docker ps
- docker run
- docker start
- docker stop

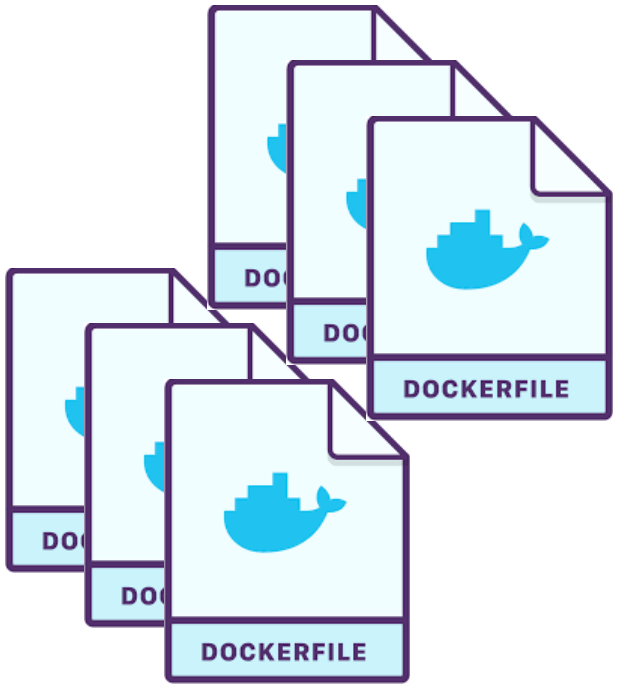
디버그

- docker exec
- docker inspect
- docker logs

실습

Docker 실습 #1,2,3


DOCKERFILE
DOCKERFILE




```
docker run -dit  -p 8080:80
             -v "$PWD":/usr/local/apache2/htdocs/ httpd:2.4
```

```
docker run -it --rm -p 8888:8080 tomcat:8.0
```

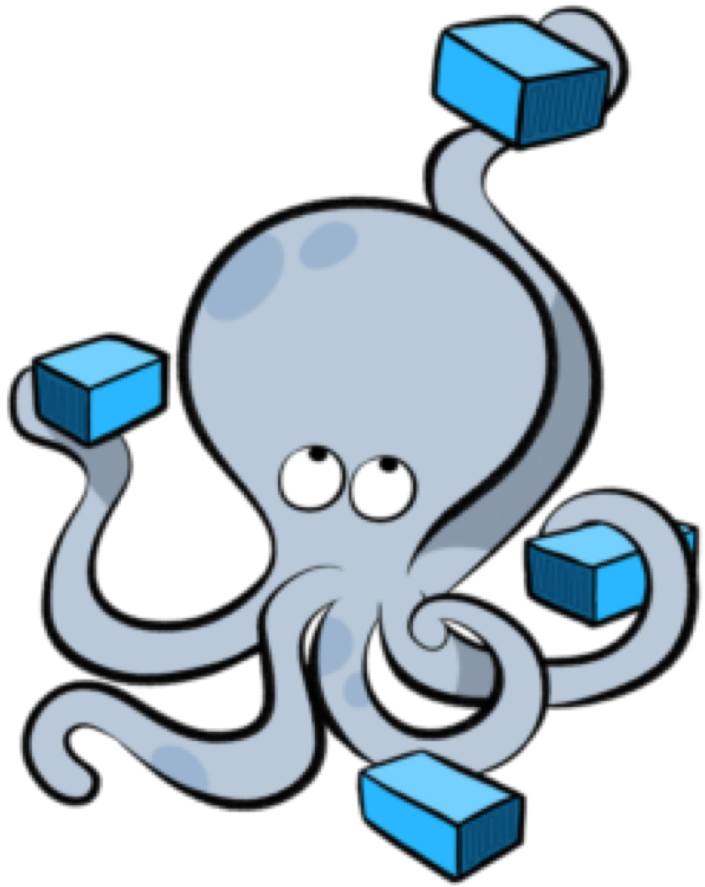
```
docker run --name some-redis -d redis
```

```
docker run --name some-mongo -d mongo:tag
```

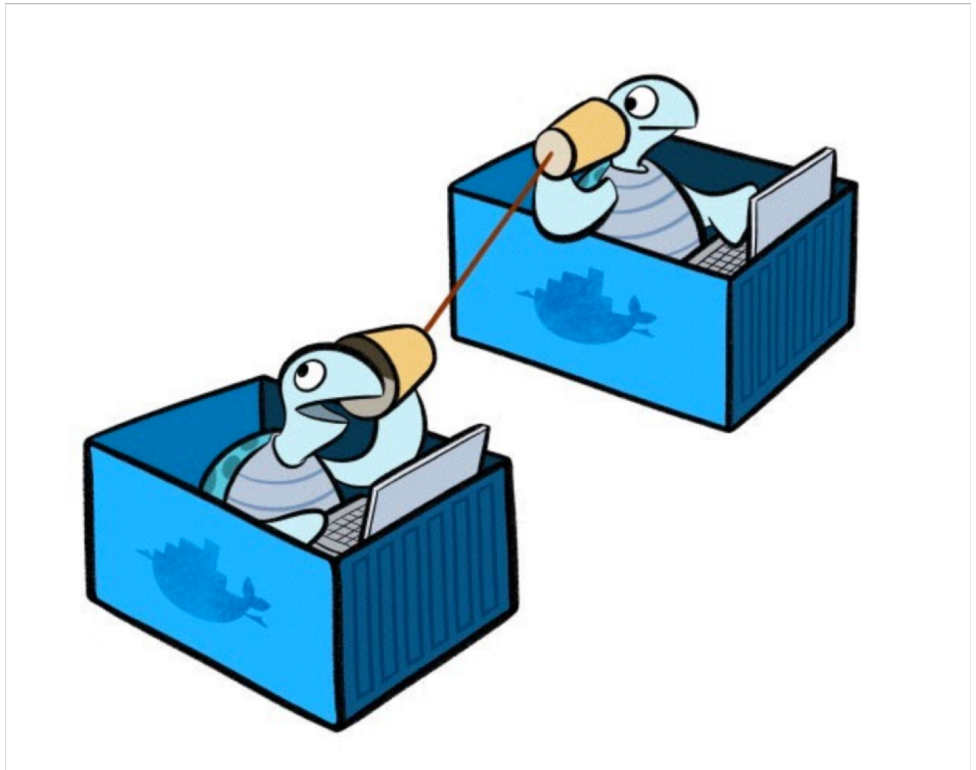
```
docker run --name some-mysql
             -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

```
docker run --name some-nginx -v /some/content:/usr/share/nginx/html:ro -d nginx
```

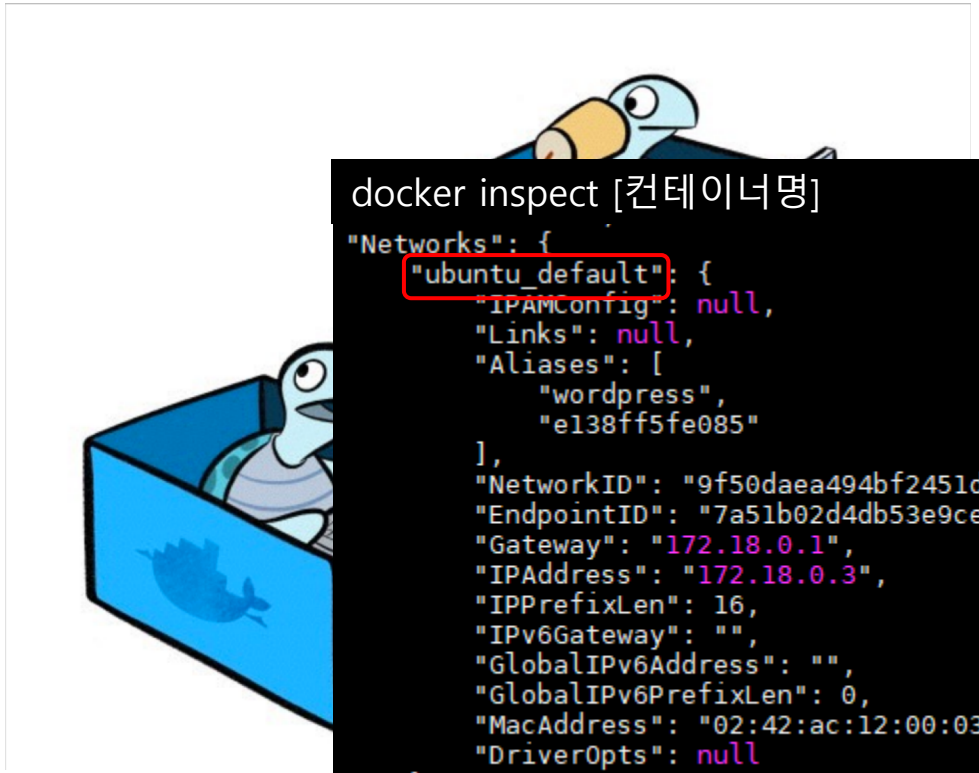
Docker Compose



```yaml
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
   web:
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



```yaml
version: '3.3'

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
  web:
    depends_on:
      - wordpress
    image: nginx
    ports:
      - "80:80"
    volumes:
      - default.conf:/etc/nginx/conf.d/default.conf
```



docker inspect [컨테이너명]

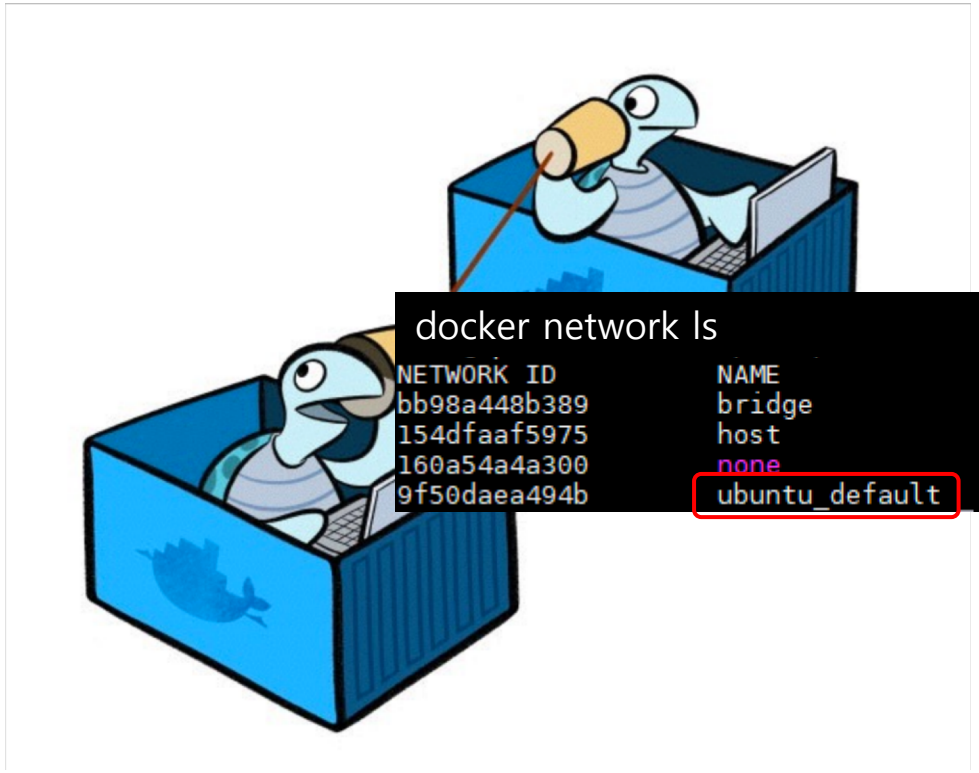

```
"Networks": {
    "ubuntu_default": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": [
            "wordpress",
            "e138ff5fe085"
        ],
        "NetworkID": "9f50daea494bf2451d409ee028360cd042a54a42f06082840589l4ff3bc6caf0",
        "EndpointID": "7a51b02d4db53e9cea2808f7a93fa2112f53045fb286150aed0956313089634e",
        "Gateway": "172.18.0.1",
        "IPAddress": "172.18.0.3",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:12:00:03",
        "DriverOpts": null
    }
}
```


```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
```

s

press

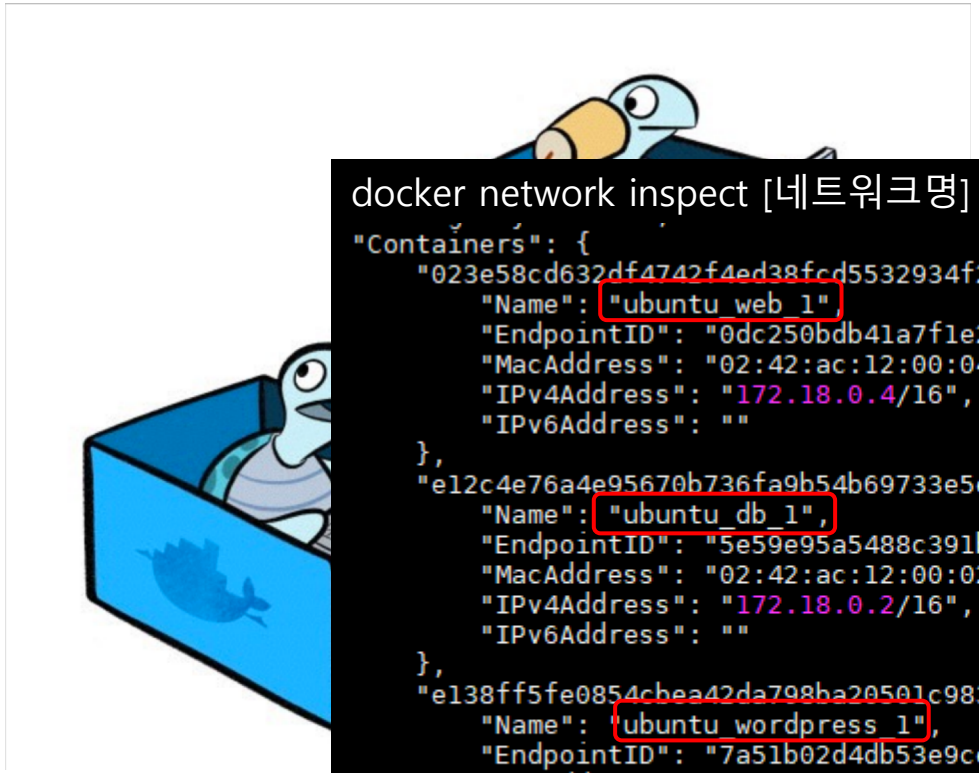
```
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



docker network ls

```
NETWORK ID          NAME                DRIVER              SCOPE
bb98a448b389        bridge              bridge              local
154dfaaf5975        host                host                local
160a54a4a300        none                null                local
9f50daea494b        ubuntu_default      bridge              local
```

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress



     ress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
   web:
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



docker network inspect [네트워크명]


```
"Containers": {
    "023e58cd632df4742f4ed38fcd5532934f2d17a0613c5289f4829d4373231228": {
        "Name": "ubuntu_web_1",
        "EndpointID": "0dc250bdb41a7f1e23e69d14f90dd29683e1f309dee90871f6be7a58bba3bb65",
        "MacAddress": "02:42:ac:12:00:04",
        "IPv4Address": "172.18.0.4/16",
        "IPv6Address": ""
    },
    "e12c4e76a4e95670b736fa9b54b69733e5e354a14baec42f4ac736c2e5b2b3e7": {
        "Name": "ubuntu_db_1",
        "EndpointID": "5e59e95a5488c391b9d9c7c3cf094164d09a8d221a81536329055c4dce11d786",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
    },
    "e138ff5fe0854cbea42da798ba20501c9836d2cfdf0b55ce8aac6689d74e0c01": {
        "Name": "ubuntu_wordpress_1",
        "EndpointID": "7a51b02d4db53e9cea2808f7a93fa2112f53045fb286150aed0956313089634e",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
    }
},
```


```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
```

```
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```

컨테이너 포트가 외부에 어떻게 노출 될까?

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
```

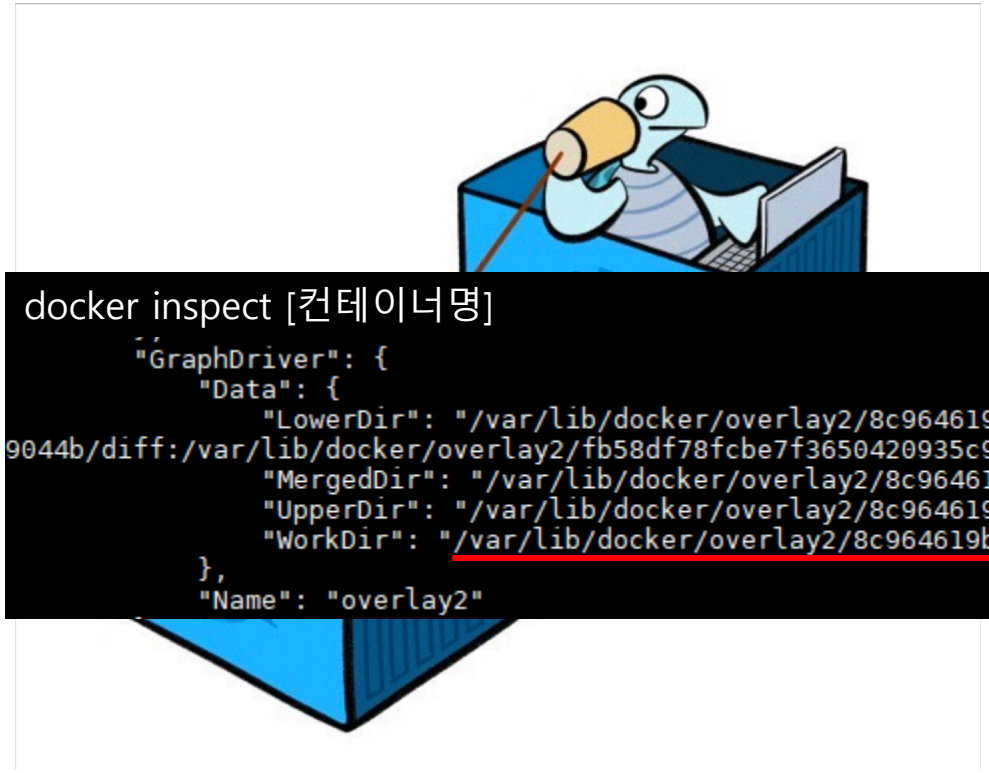
iptables –t nat –L –n

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0            0.0.0.0/0            ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER     all  --  0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  172.18.0.0/16        0.0.0.0/0
MASQUERADE  all  --  172.17.0.0/16        0.0.0.0/0
MASQUERADE  tcp  --  172.18.0.3           172.18.0.3           tcp dpt:80
MASQUERADE  tcp  --  172.18.0.4           172.18.0.4           tcp dpt:80

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:8000 to:172.18.0.3:80
DNAT       tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:80 to:172.18.0.4:80
```

```
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```

볼륨은 어떻게 될까?

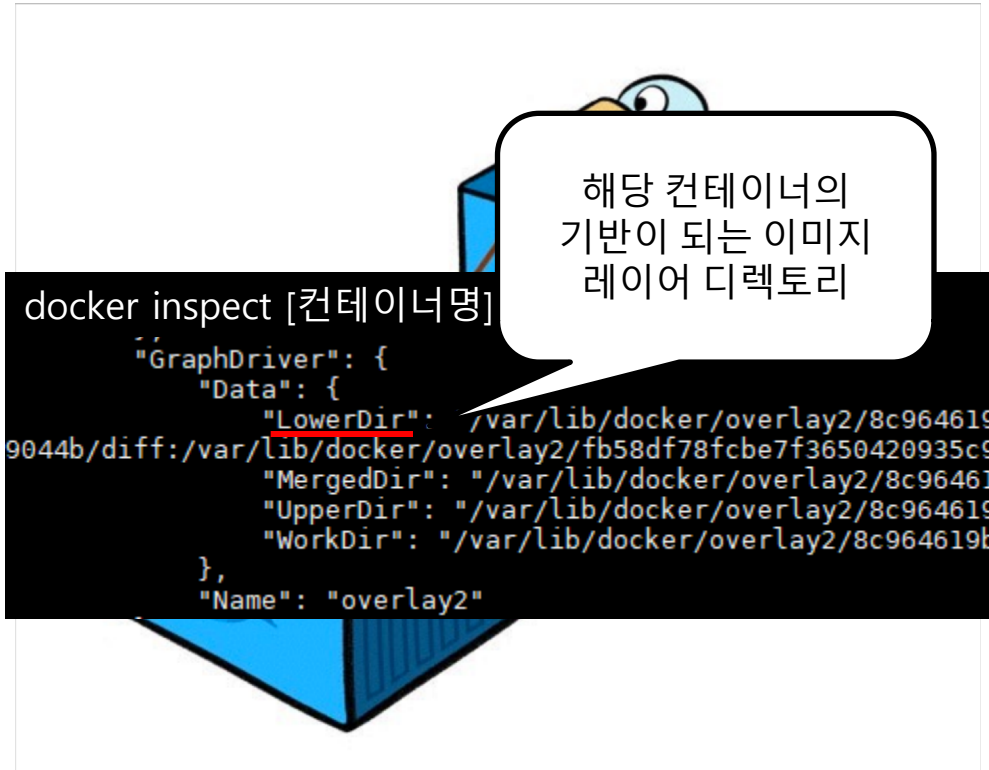
(앞서 배운 내용 복습)



docker inspect [컨테이너명]


```
        "GraphDriver": {
            "Data": {
                "LowerDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d-init/diff:/var
9044b/diff:/var/lib/docker/overlay2/fb58df78fcbe7f3650420935c92ec86c8b9b63aac9f8747f455c0a8ed9732c5c/diff:/var/lib/docker/overlay2/69
                "MergedDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/merged",
                "UpperDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/diff",
                "WorkDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/work"
            },
            "Name": "overlay2"
```

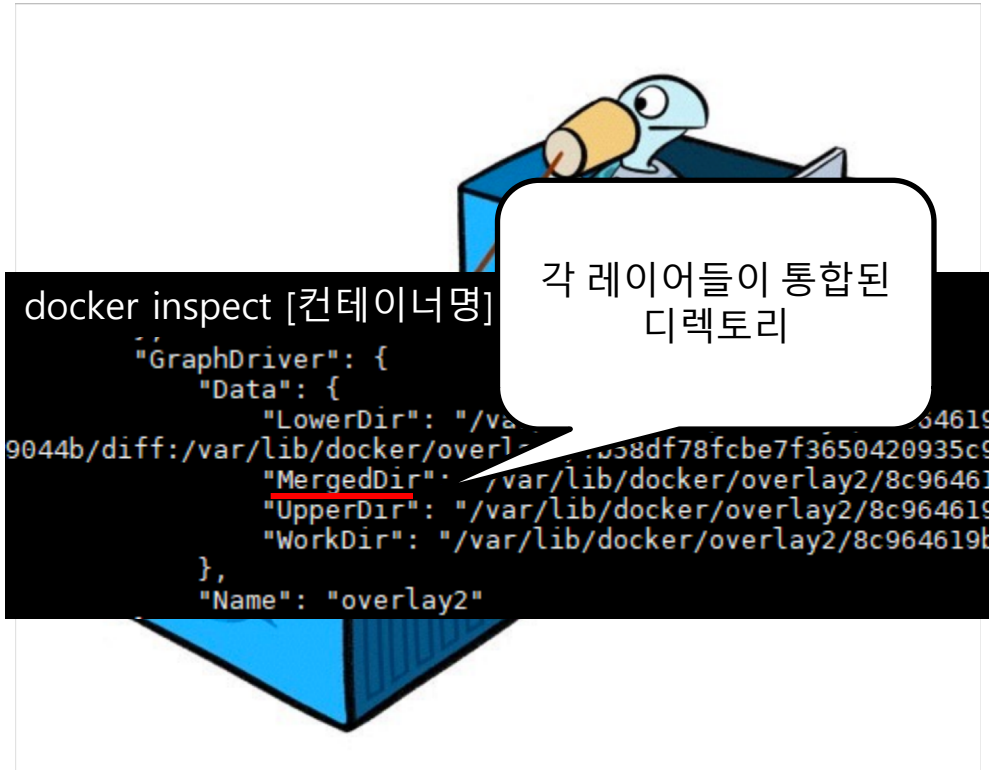

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
   web:
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



docker inspect [컨테이너명]

```
        "GraphDriver": {
            "Data": {
                "LowerDir":  /var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d-init/diff:/var/
9044b/diff:/var/lib/docker/overlay2/fb58df78fcbe7f3650420935c92ec86c8b9b63aac9f8747f455c0a8ed9732c5c/diff:/var/lib/docker/overlay2/69
                "MergedDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/merged",
                "UpperDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/diff",
                "WorkDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/work"
            },
            "Name": "overlay2"
```

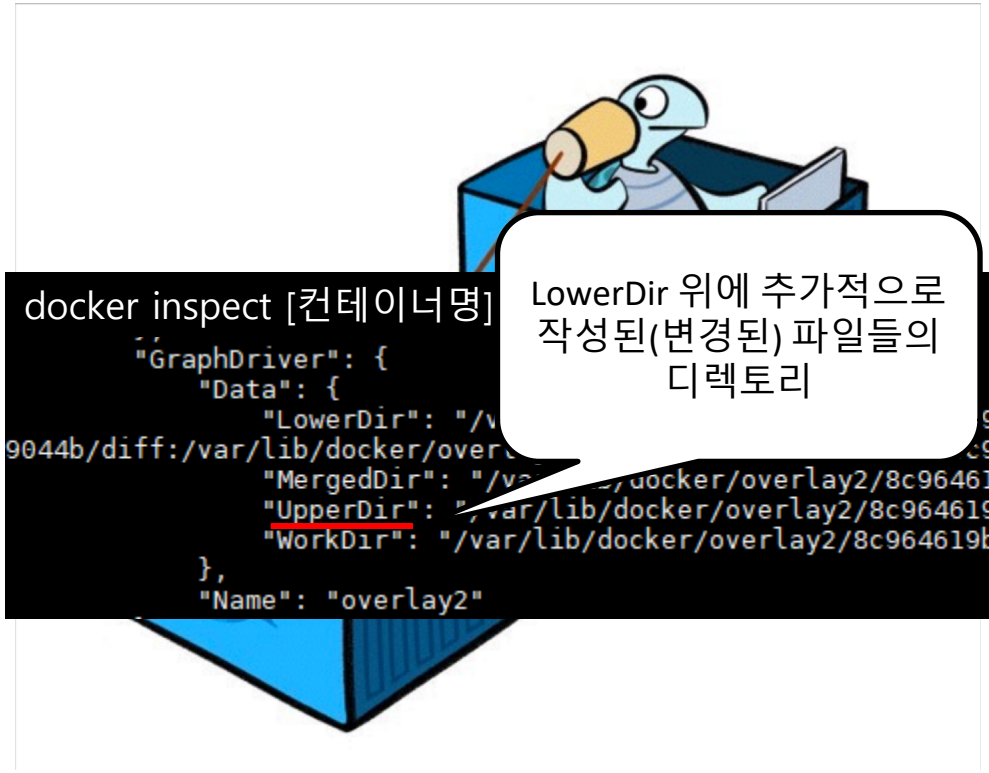
```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
   web:
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



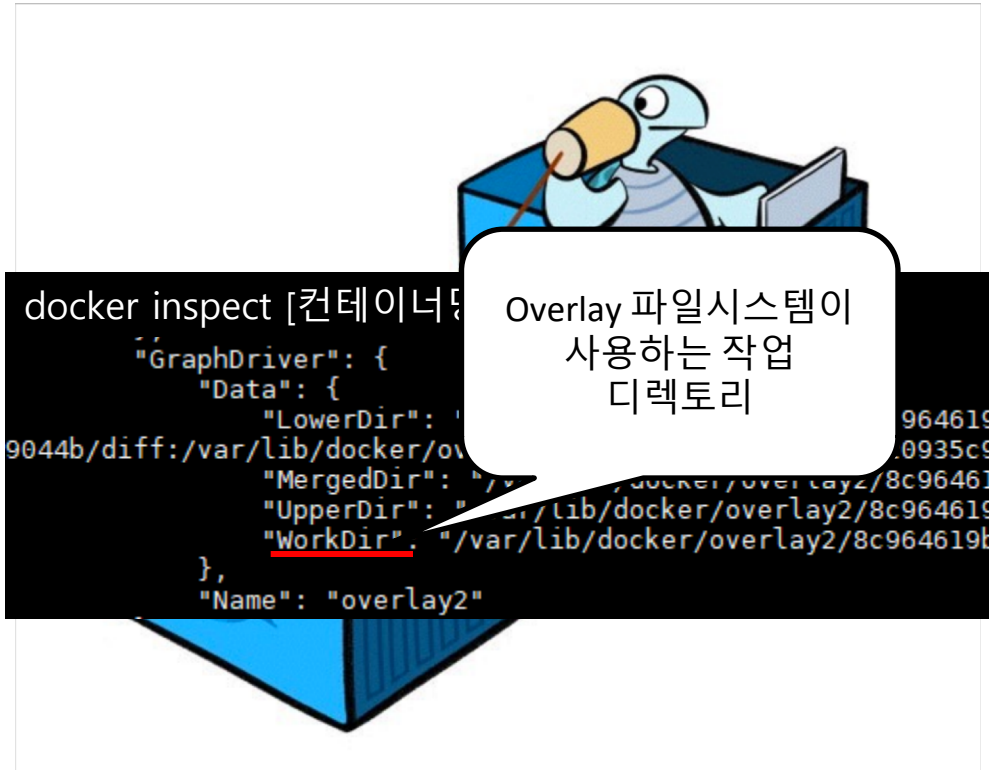
docker inspect [컨테이너명]

```
        "GraphDriver": {
            "Data": {
                "LowerDir": "/va                                    64619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d-init/diff:/var
9044b/diff:/var/lib/docker/overl                    38df78fcbe7f3650420935c92ec86c8b9b63aac9f8747f455c0a8ed9732c5c/diff:/var/lib/docker/overlay2/69
                "MergedDir":  /var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/merged",
                "UpperDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/diff",
                "WorkDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/work"
            },
            "Name": "overlay2"
```

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
   web:
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```




```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
   web:
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```

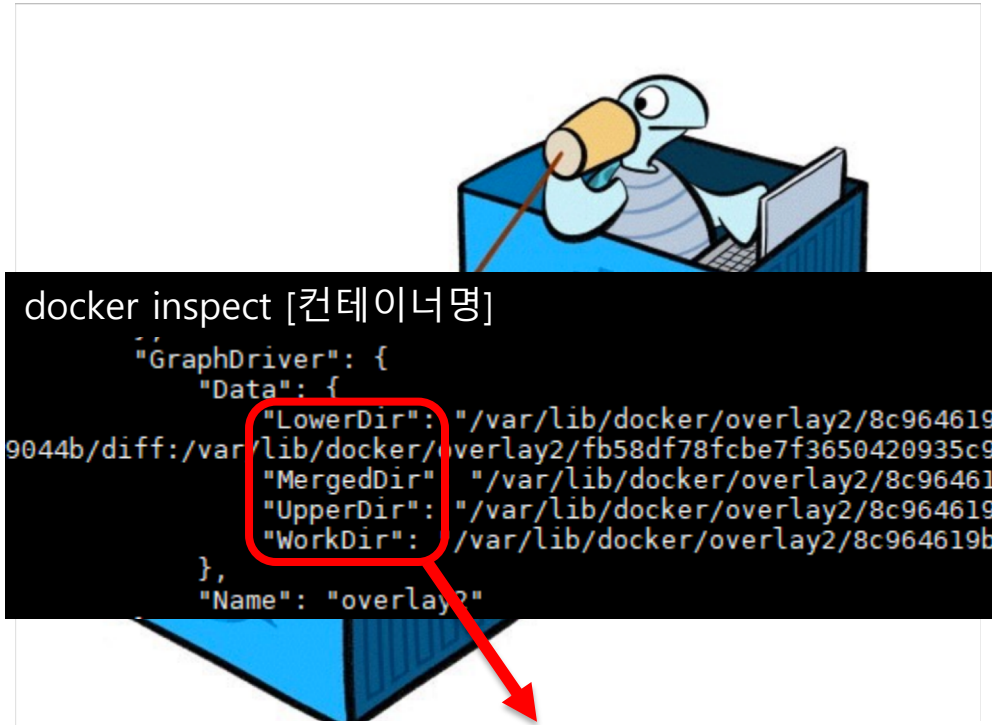


docker inspect [컨테이너명]

```
        "GraphDriver": {
            "Data": {
                "LowerDir": "
964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d-init/diff:/var
9044b/diff:/var/lib/docker/ov
0935c92ec86c8b9b63aac9f8747f455c0a8ed9732c5c/diff:/var/lib/docker/overlay2/69
                "MergedDir": "/v
/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/merged",
                "UpperDir": "
/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/diff",
                "WorkDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/work"
            },
            "Name": "overlay2"
```

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
   web:
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```

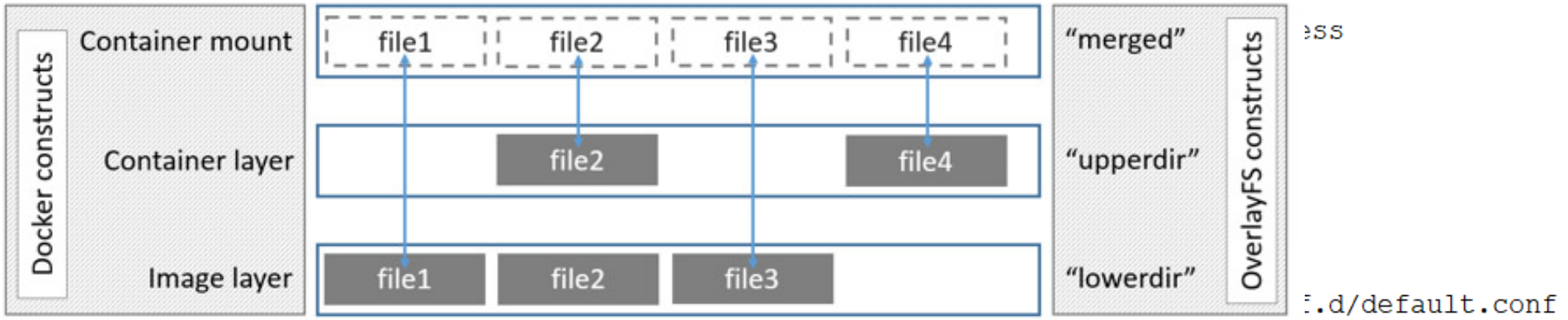
```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
```



docker inspect [컨테이너명]

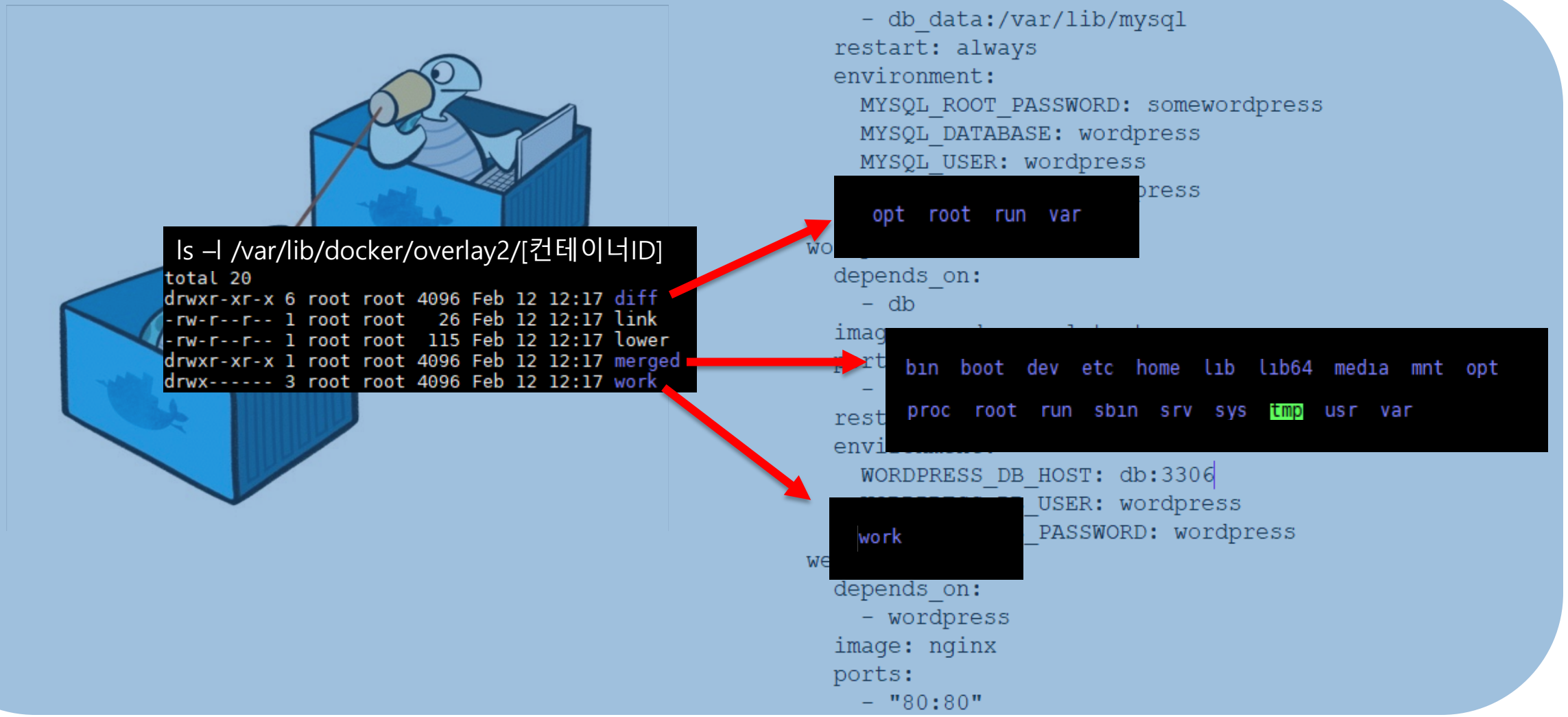
```
        "GraphDriver": {
            "Data": {
                "LowerDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d-init/diff:/var
9044b/diff:/var/lib/docker/overlay2/fb58df78fcbe7f3650420935c92ec86c8b9b63aac9f8747f455c0a8ed9732c5c/diff:/var/lib/docker/overlay2/69
                "MergedDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/merged",
                "UpperDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/diff",
                "WorkDir": "/var/lib/docker/overlay2/8c964619bdefa6cba1bf77175c6e84b6b99ed4b9ceee2adc804662c38759405d/work"
            },
            "Name": "overlay2"
```

```
     environment:
       WORDPRESS_DB_HOST: db:3306
```

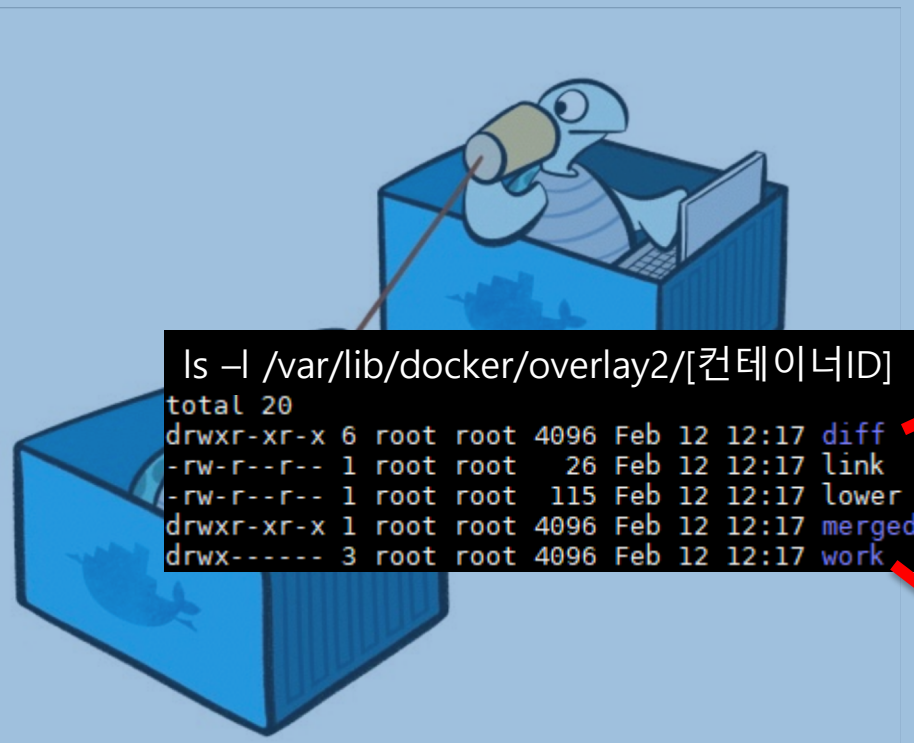


ess

f.d/default.conf




```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
                                    press
   wo
     depends_on:
       - db
     imag
     p   rt
       -
     rest
     envi
       WORDPRESS_DB_HOST: db:3306
                          _USER: wordpress
                          _PASSWORD: wordpress
   we
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



```
ls –l /var/lib/docker/overlay2/[컨테이너ID]
total 20
drwxr-xr-x 6 root root 4096 Feb 12 12:17 diff
-rw-r--r-- 1 root root   26 Feb 12 12:17 link
-rw-r--r-- 1 root root  115 Feb 12 12:17 lower
drwxr-xr-x 1 root root 4096 Feb 12 12:17 merged
drwx------ 3 root root 4096 Feb 12 12:17 work
```

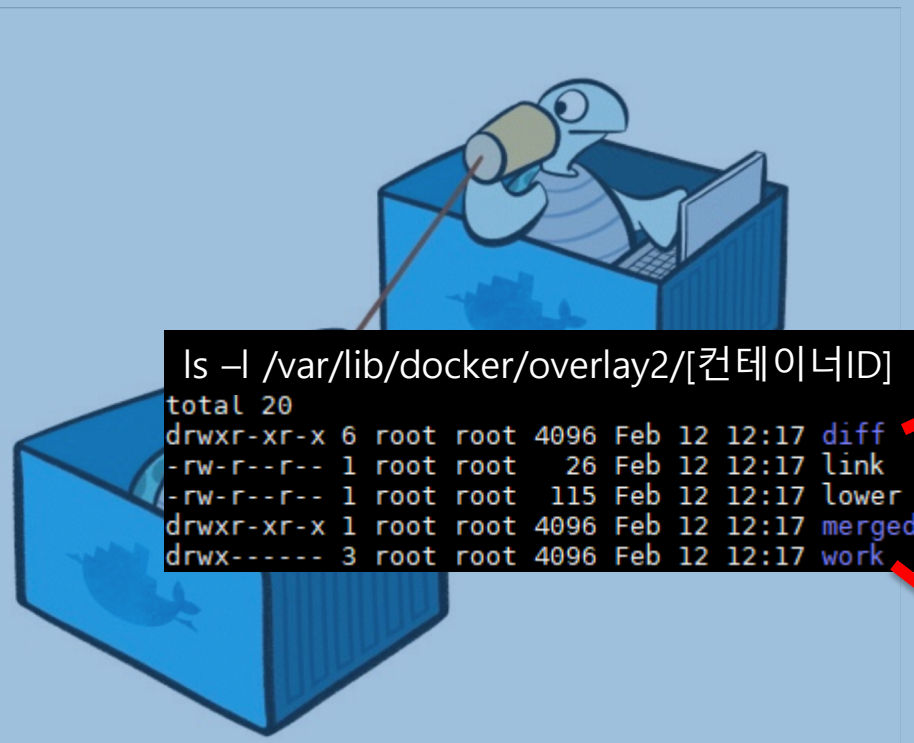
```
opt  root  run  var
```

파일 시스템 레이어의
변경된 데이터

```
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys  tmp  usr  var
```

```
work
```

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somew
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
     depends_on:
       - db
     WORDPRESS_DB_HOST: db:3306
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



```
ls –l /var/lib/docker/overlay2/[컨테이너ID]
total 20
drwxr-xr-x 6 root root 4096 Feb 12 12:17 diff
-rw-r--r-- 1 root root   26 Feb 12 12:17 link
-rw-r--r-- 1 root root  115 Feb 12 12:17 lower
drwxr-xr-x 1 root root 4096 Feb 12 12:17 merged
drwx------ 3 root root 4096 Feb 12 12:17 work
```

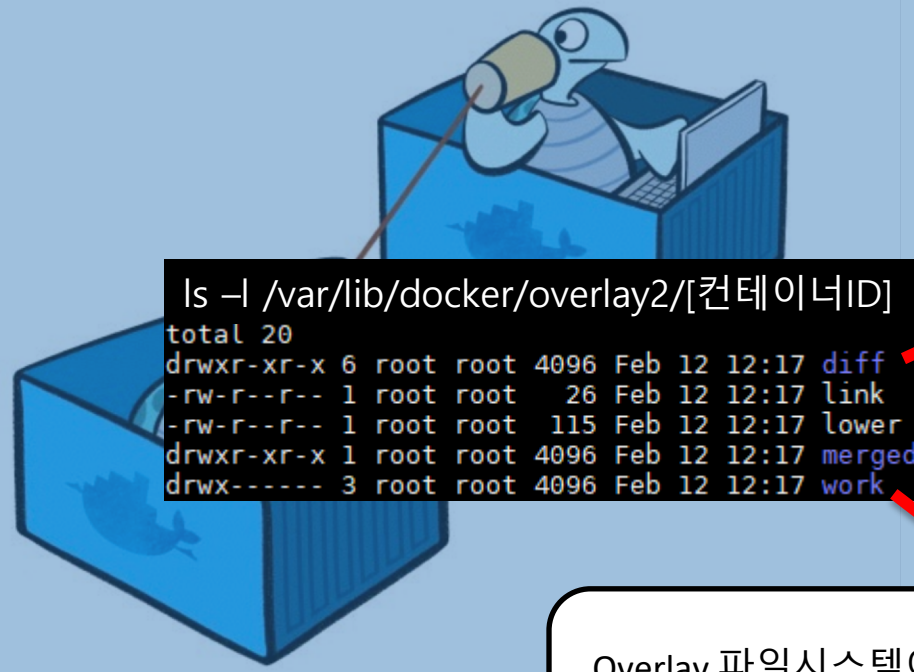
```
opt  root  run  var
```

```
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys  tmp  usr  var
```

```
work
```

통합된 디렉토리

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somev
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
     depends_on:
       - db
     WORDPRESS_DB_HOST: db:3306
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```



```
ls –l /var/lib/docker/overlay2/[컨테이너ID]
total 20
drwxr-xr-x 6 root root 4096 Feb 12 12:17 diff
-rw-r--r-- 1 root root   26 Feb 12 12:17 link
-rw-r--r-- 1 root root  115 Feb 12 12:17 lower
drwxr-xr-x 1 root root 4096 Feb 12 12:17 merged
drwx------ 3 root root 4096 Feb 12 12:17 work
```

```
opt  root  run  var
```

```
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys  tmp  usr  var
```

```
work
```

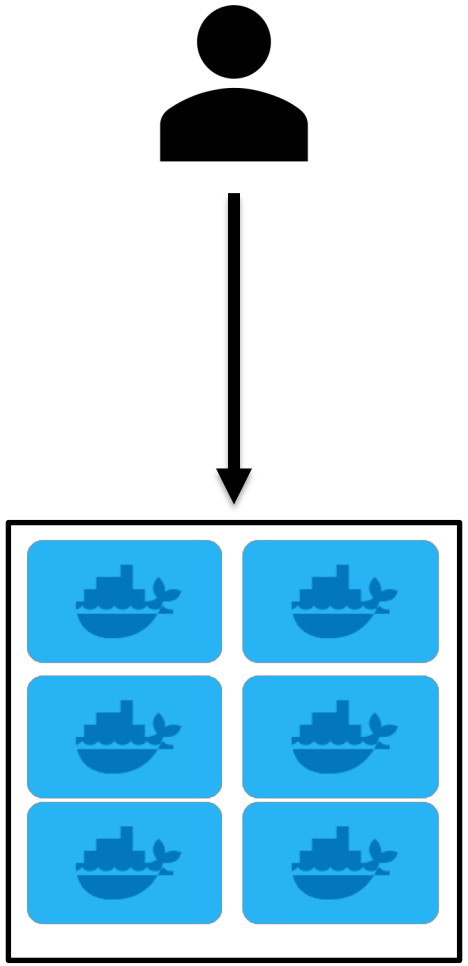
Overlay 파일시스템이 사용하는 작업 디렉토리

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
     depends_on:
       - db
     WORDPRESS_DB_HOST: db:3306
     depends_on:
       - wordpress
     image: nginx
     ports:
       - "80:80"
     volumes:
       - default.conf:/etc/nginx/conf.d/default.conf
```

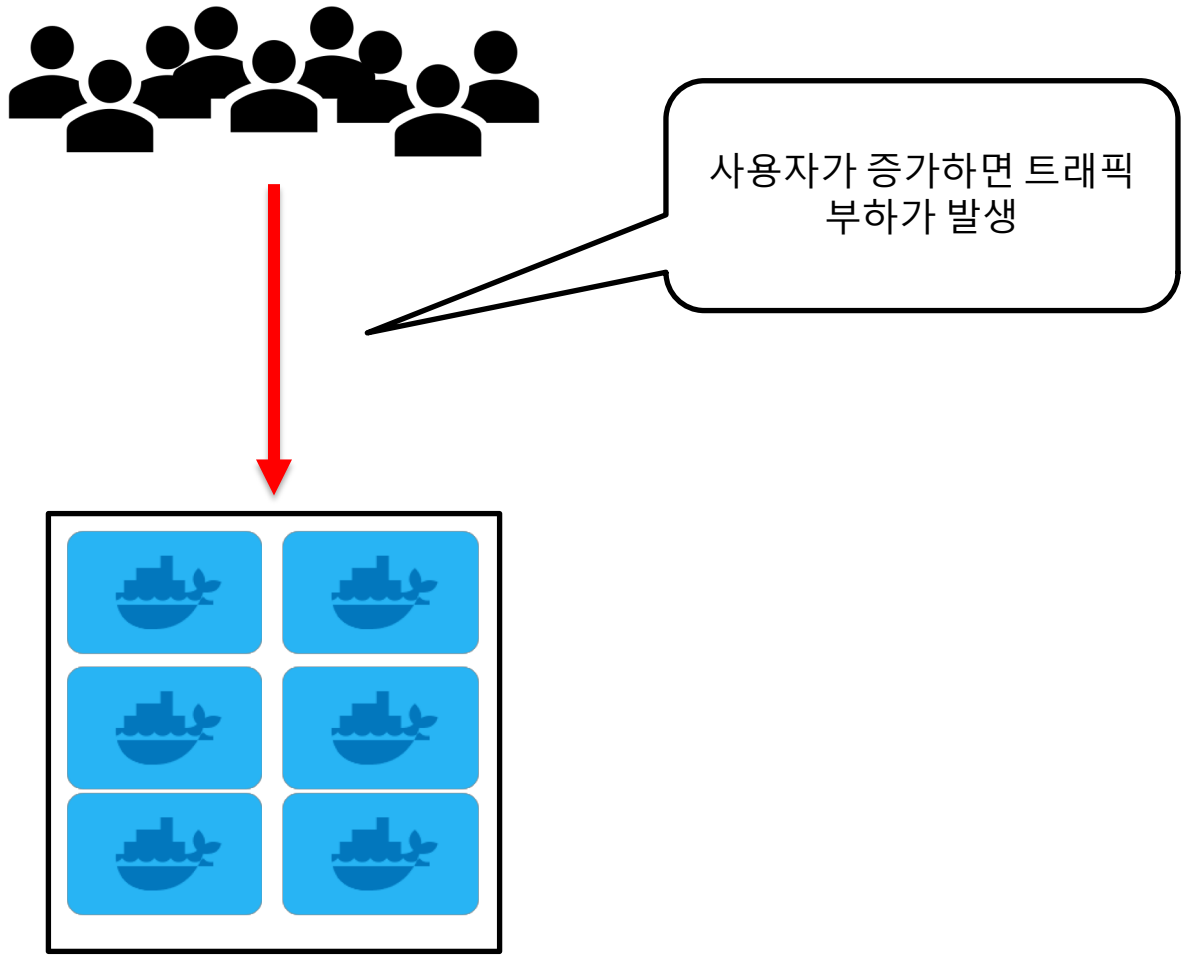
실습

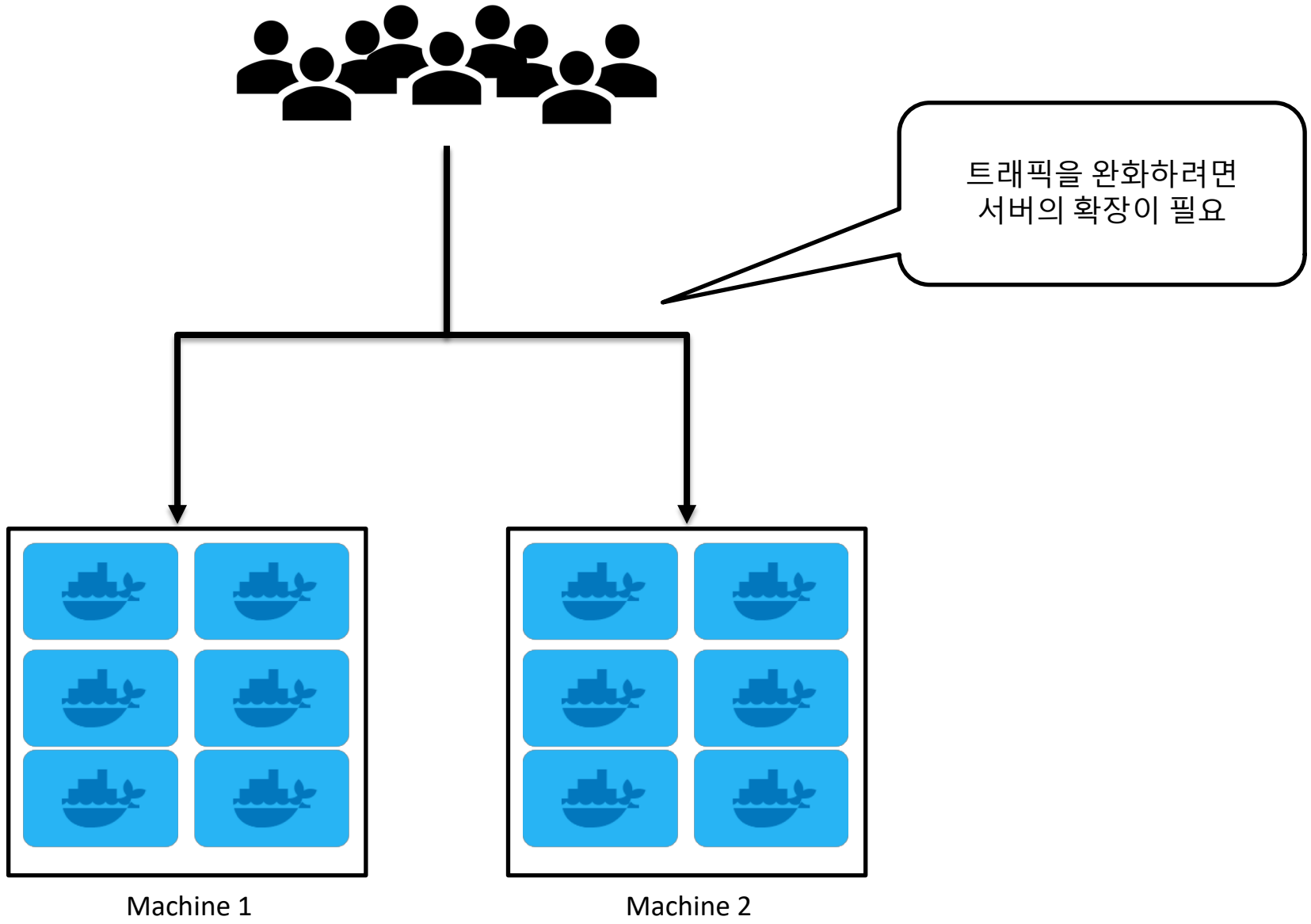
Docker-compose 실습 #1,2

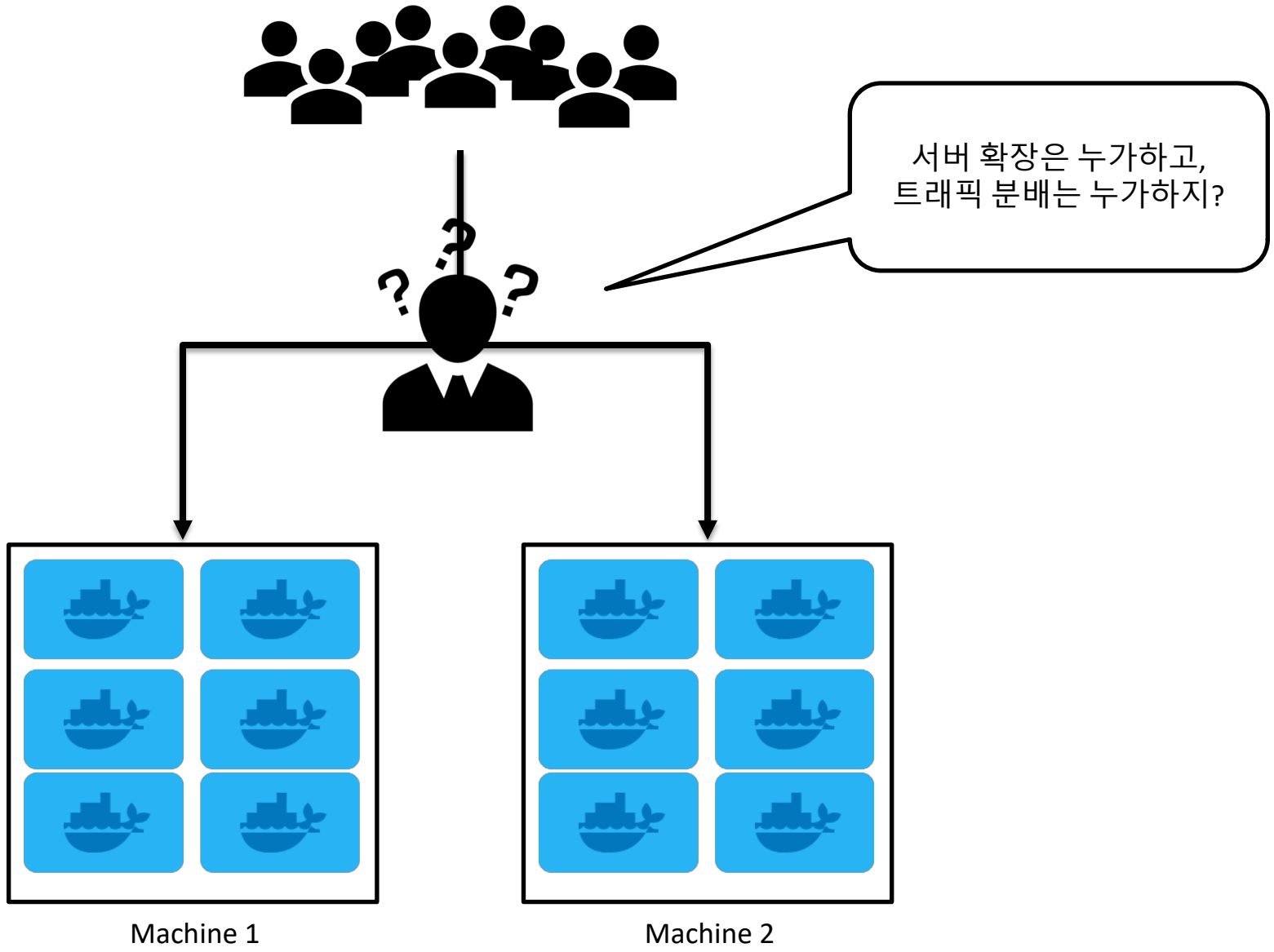
실제 서비스에 도커를 활용하려면?

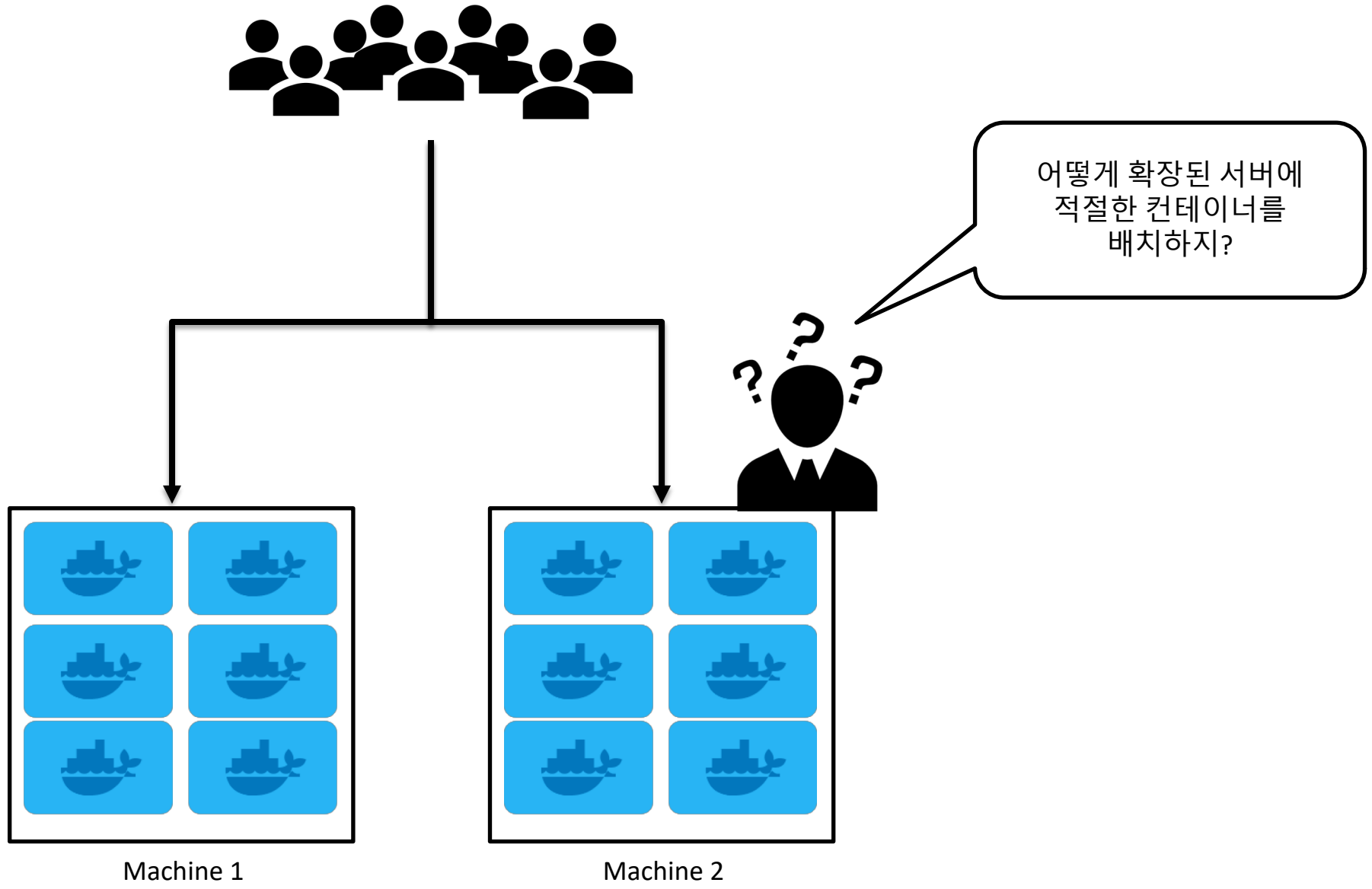


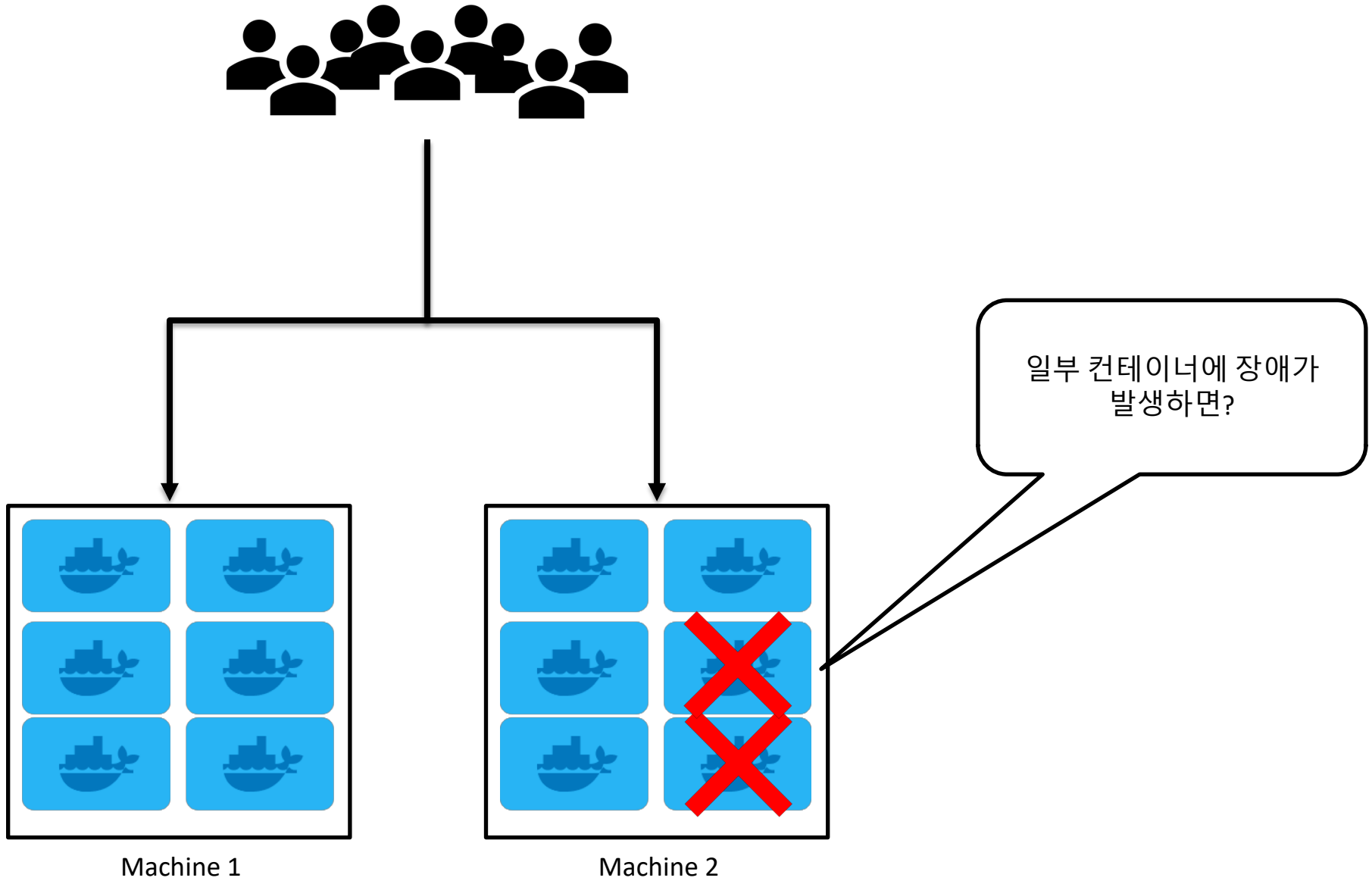
Machine 1

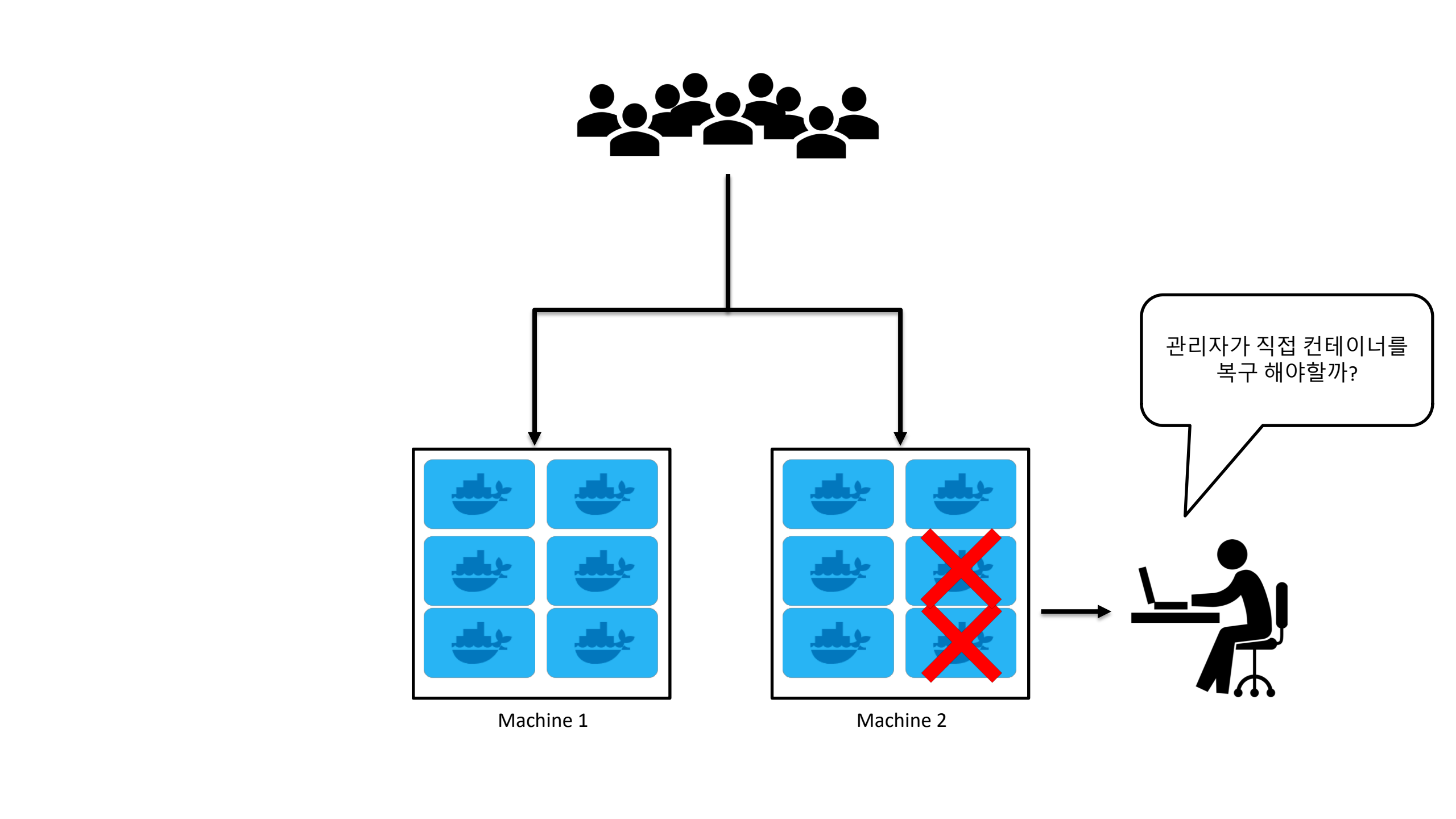

사용자가 증가하면 트래픽
부하가 발생
Machine 1

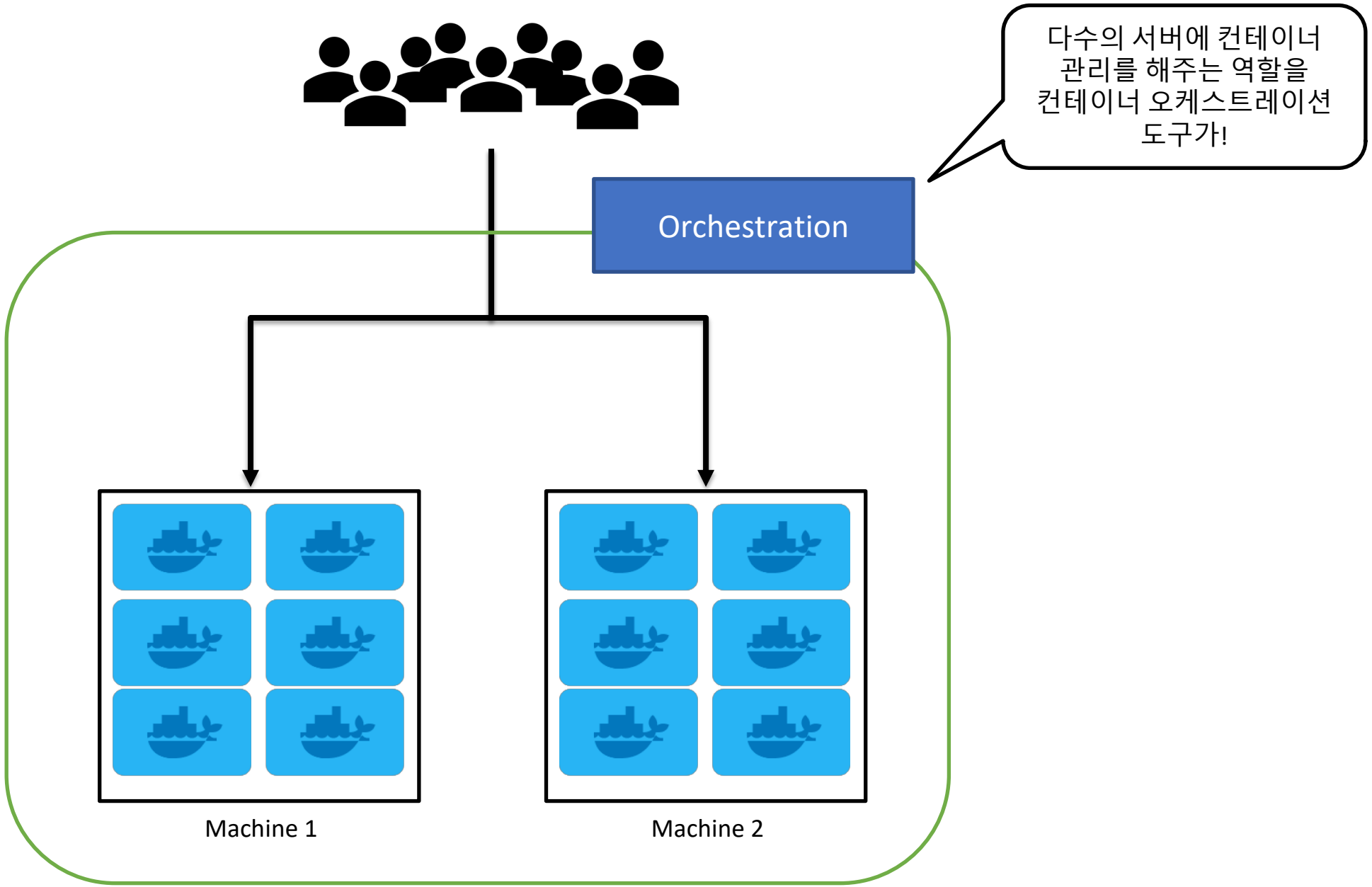

트래픽을 완화하려면
서버의 확장이 필요
Machine 1
Machine 2


서버 확장은 누가하고,
트래픽 분배는 누가하지?
Machine 1
Machine 2


어떻게 확장된 서버에
적절한 컨테이너를
배치하지?
Machine 1
Machine 2


일부 컨테이너에 장애가
발생하면?
Machine 1
Machine 2


관리자가 직접 컨테이너를
복구 해야할까?
Machine 1
Machine 2


다수의 서버에 컨테이너 관리를 해주는 역할을 컨테이너 오케스트레이션 도구가!
Orchestration
Machine 1
Machine 2

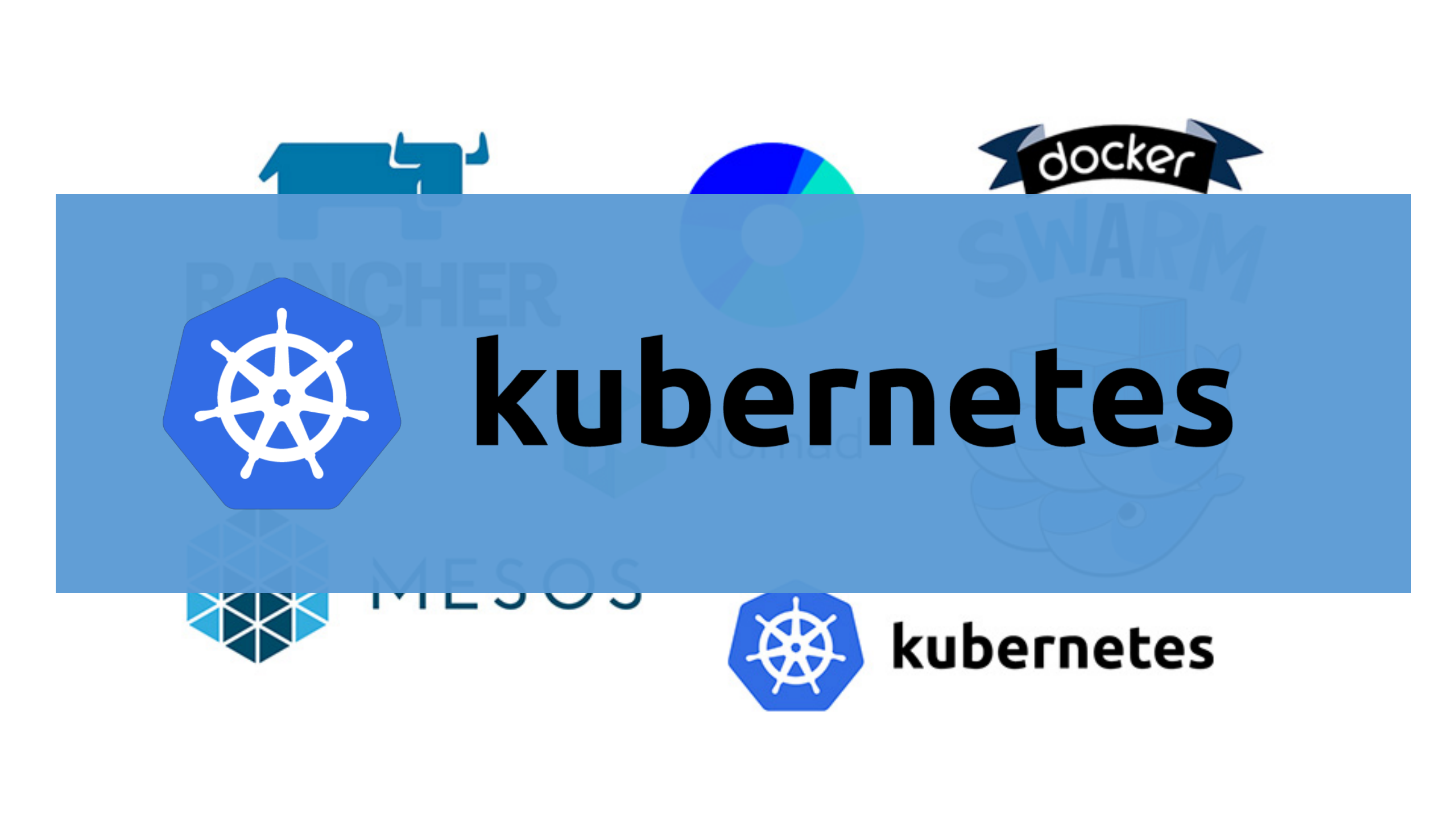

RANCHER

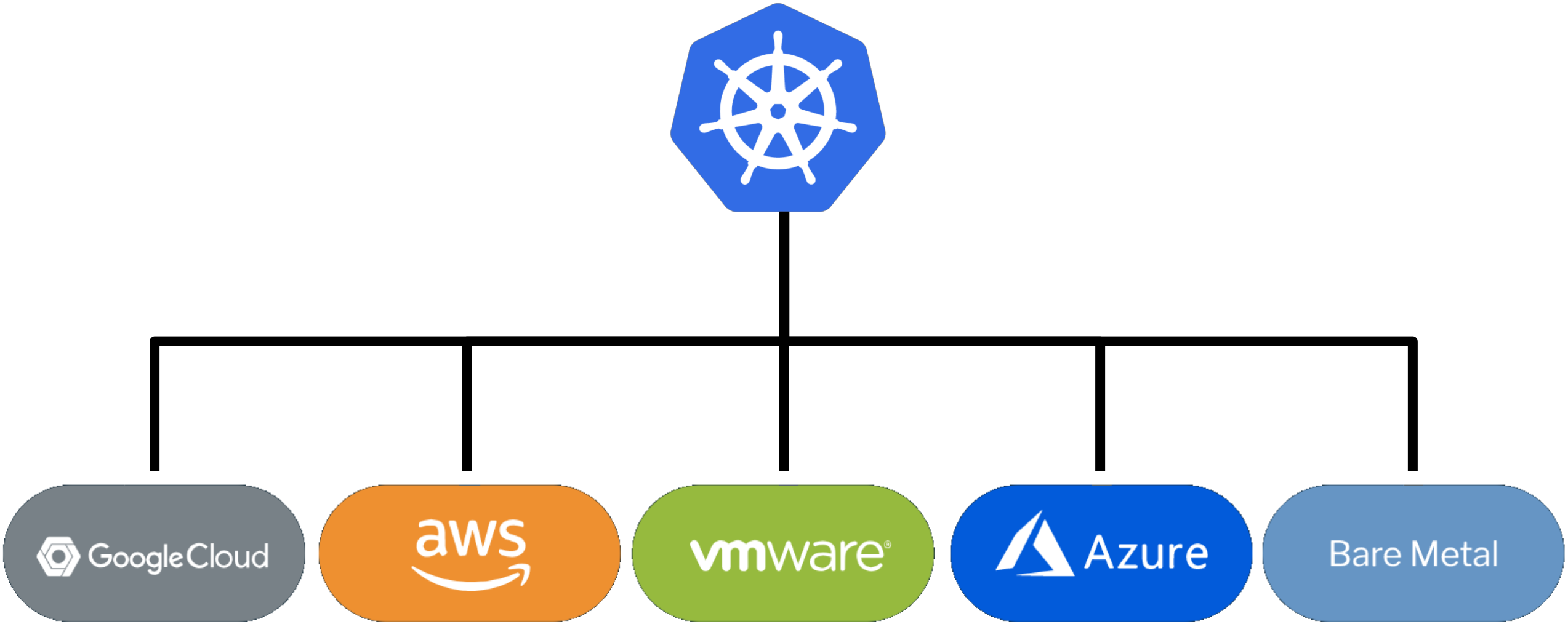

docker
SWARM


Nomad

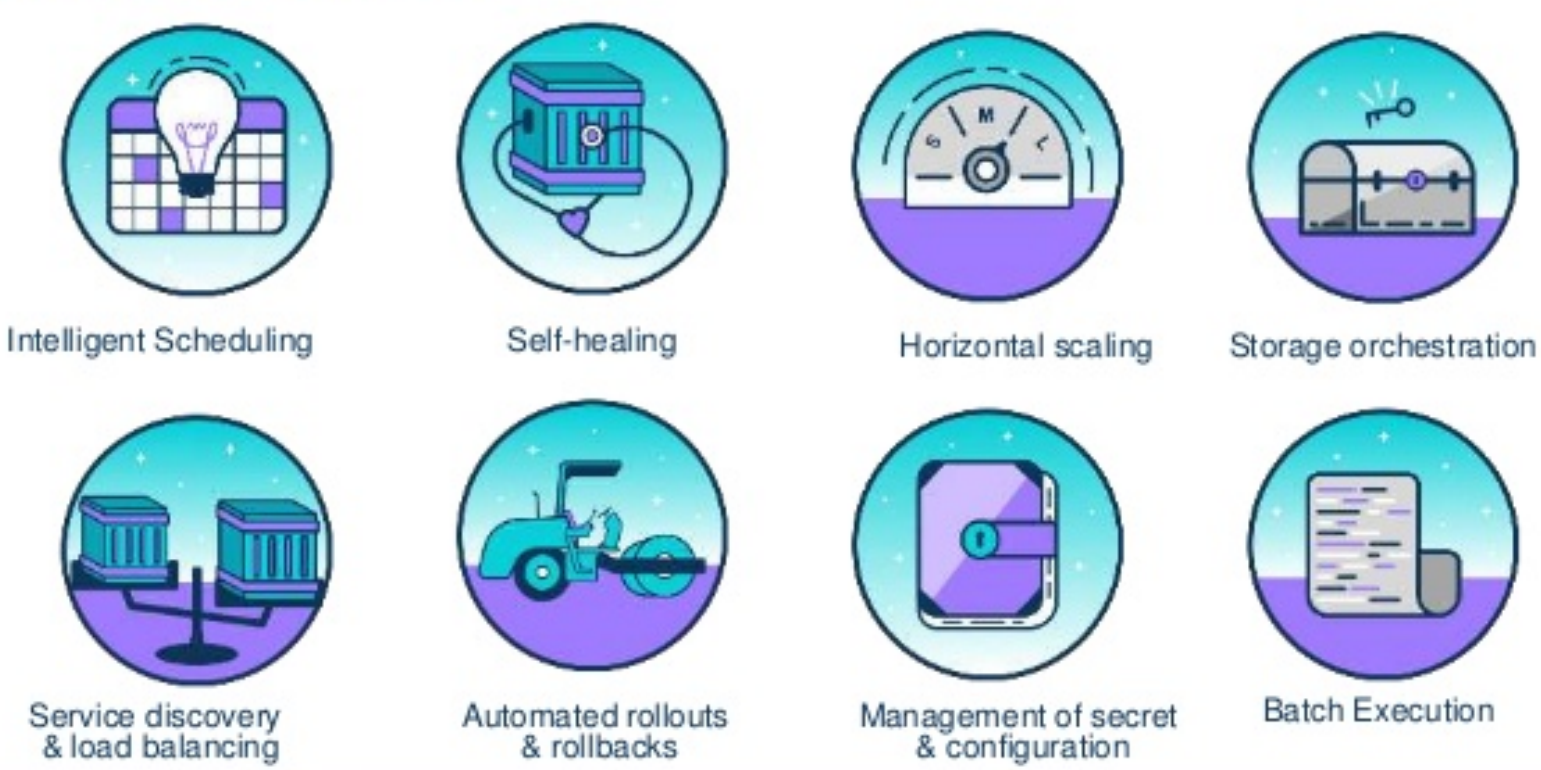

MESOS


kubernetes


kubernetes
docker
RANCHER
SWARM
MESOS
kubernetes


Google Cloud
aws
vmware
Azure
Bare Metal

쿠버네티스 기능



출저 : IBM

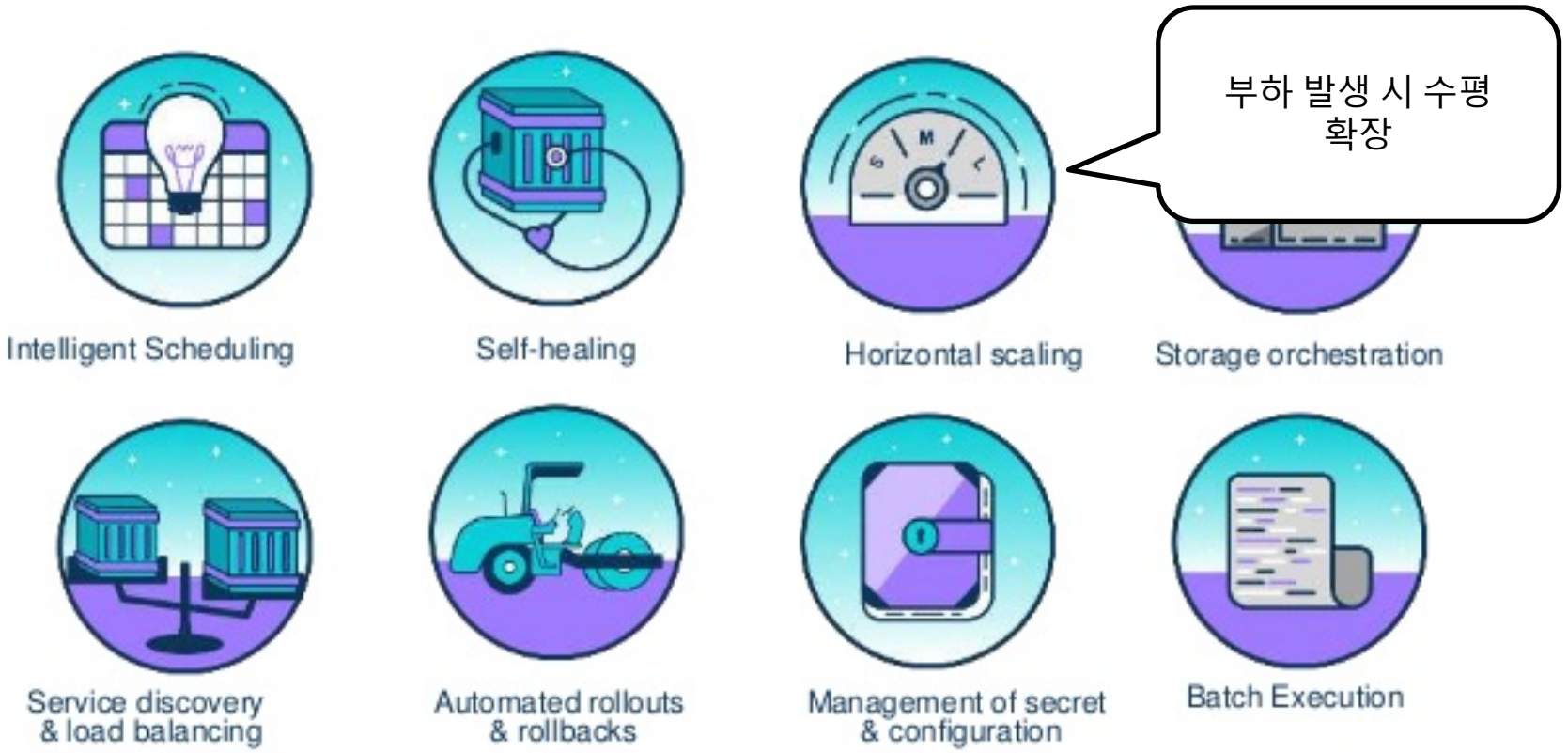
쿠버네티스 기능

출저 : IBM

쿠버네티스 기능

출저 : IBM

쿠버네티스 기능



출저 : IBM

쿠버네티스 기능

출저 : IBM

쿠버네티스 기능

출저 : IBM

쿠버네티스 기능

출저 : IBM

쿠버네티스 기능

출저 : IBM

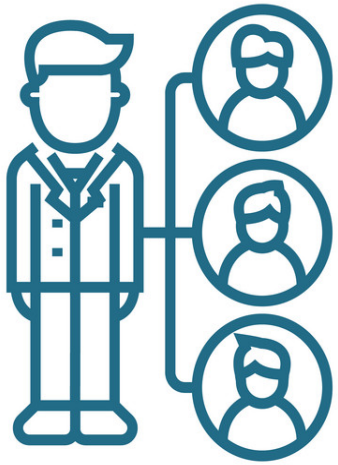
쿠버네티스 기능

출저 : IBM

쿠버네티스 구조

실장님

업무 지시



협업 부서

업무 요청

팀장님

업무 지시 및 분배

팀원들

관리자

명령 전달

서비스 사용자

서비스 요청

master

명령 전달

트래픽 분배

node

node

node

node

관리자

명령 전달

서비스 사용자

서비스 요청

- Pod 스케줄링 및 이벤트 처리
- Pod 실행 및 중지
- Pod 스케일링
- Pod 업데이트

master

명령 전달

트래픽 분배

node

node

node

node

관리자

명령 전달

master

명령 전달

트래픽 분배

node

node

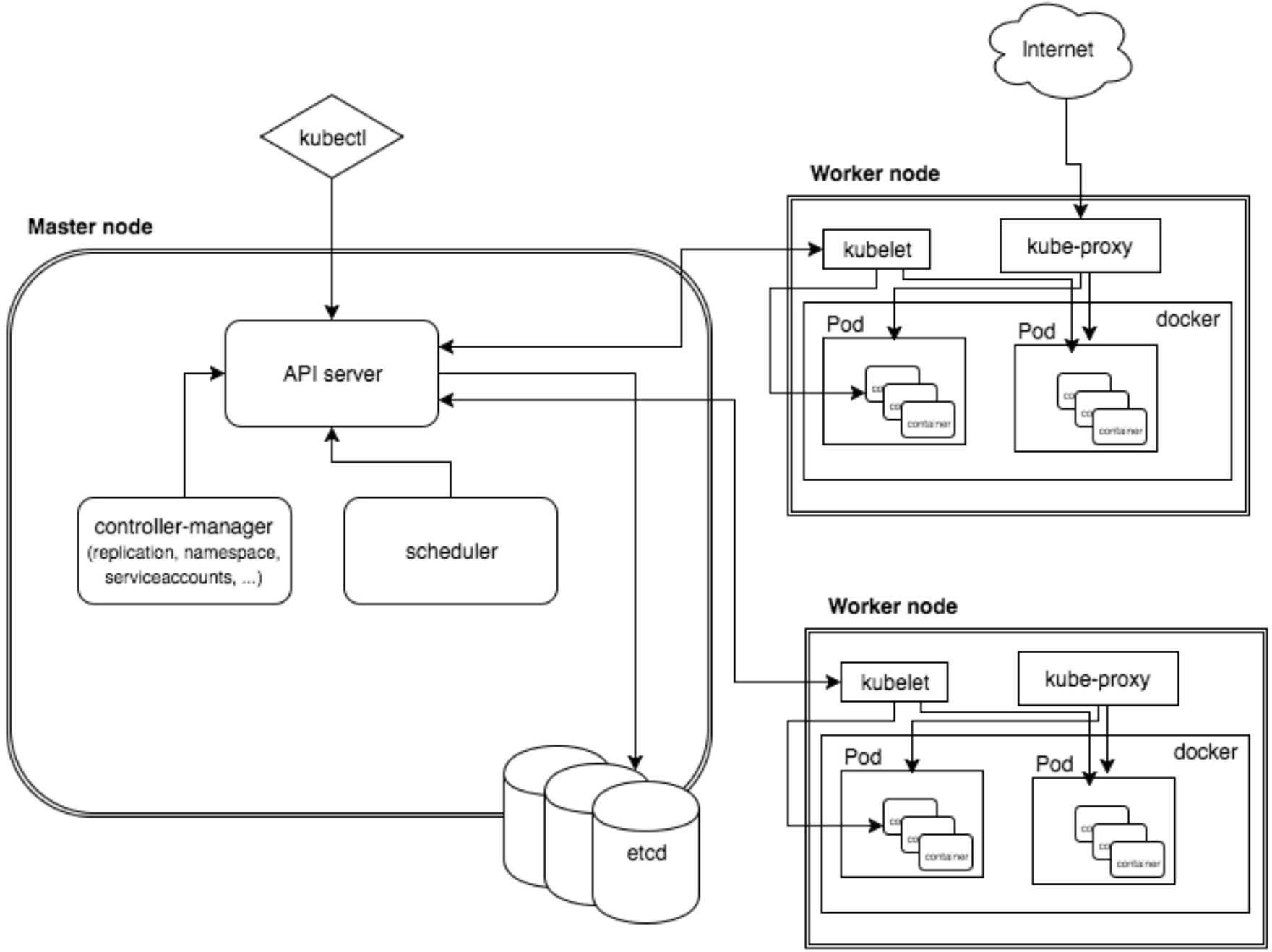
node

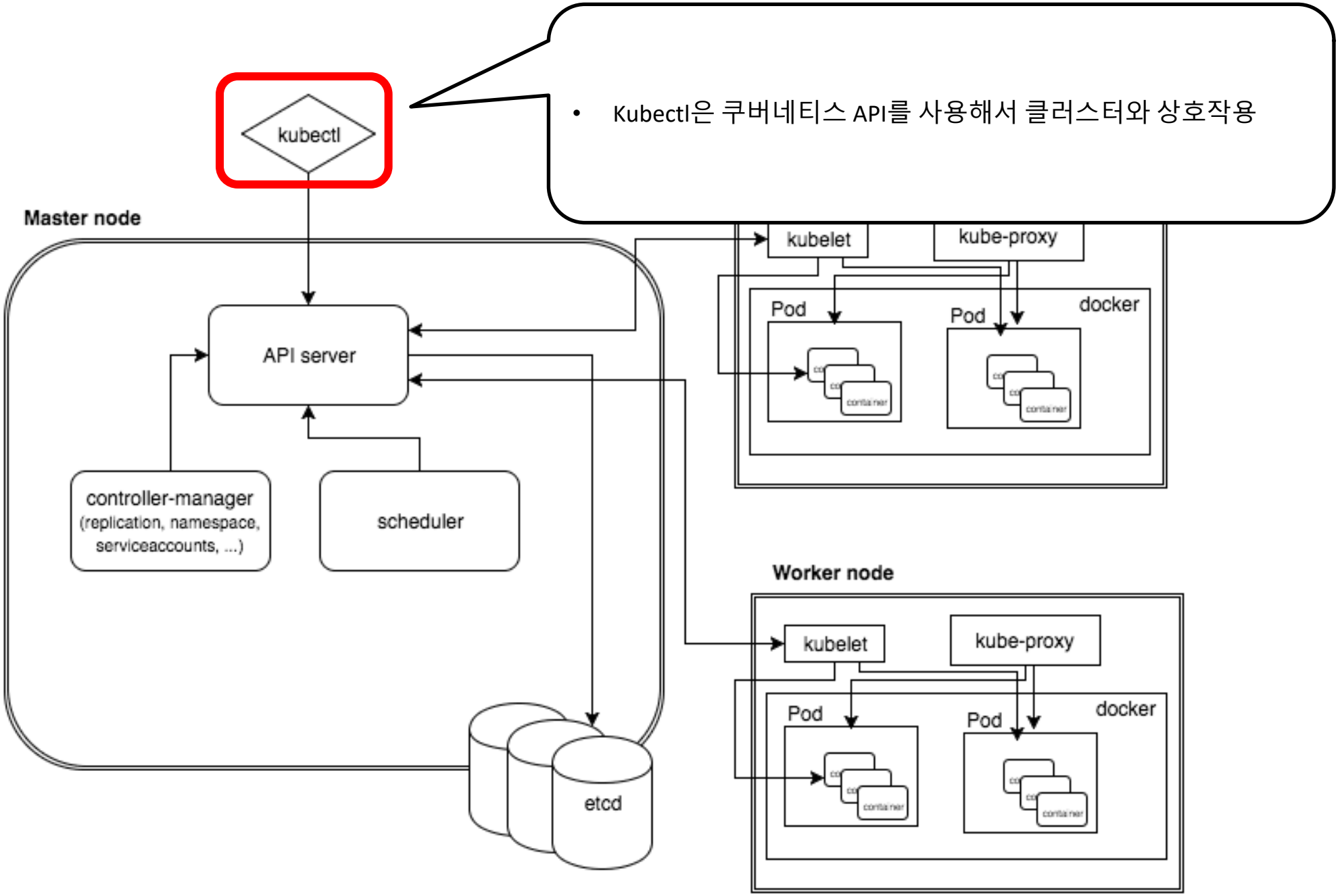
node

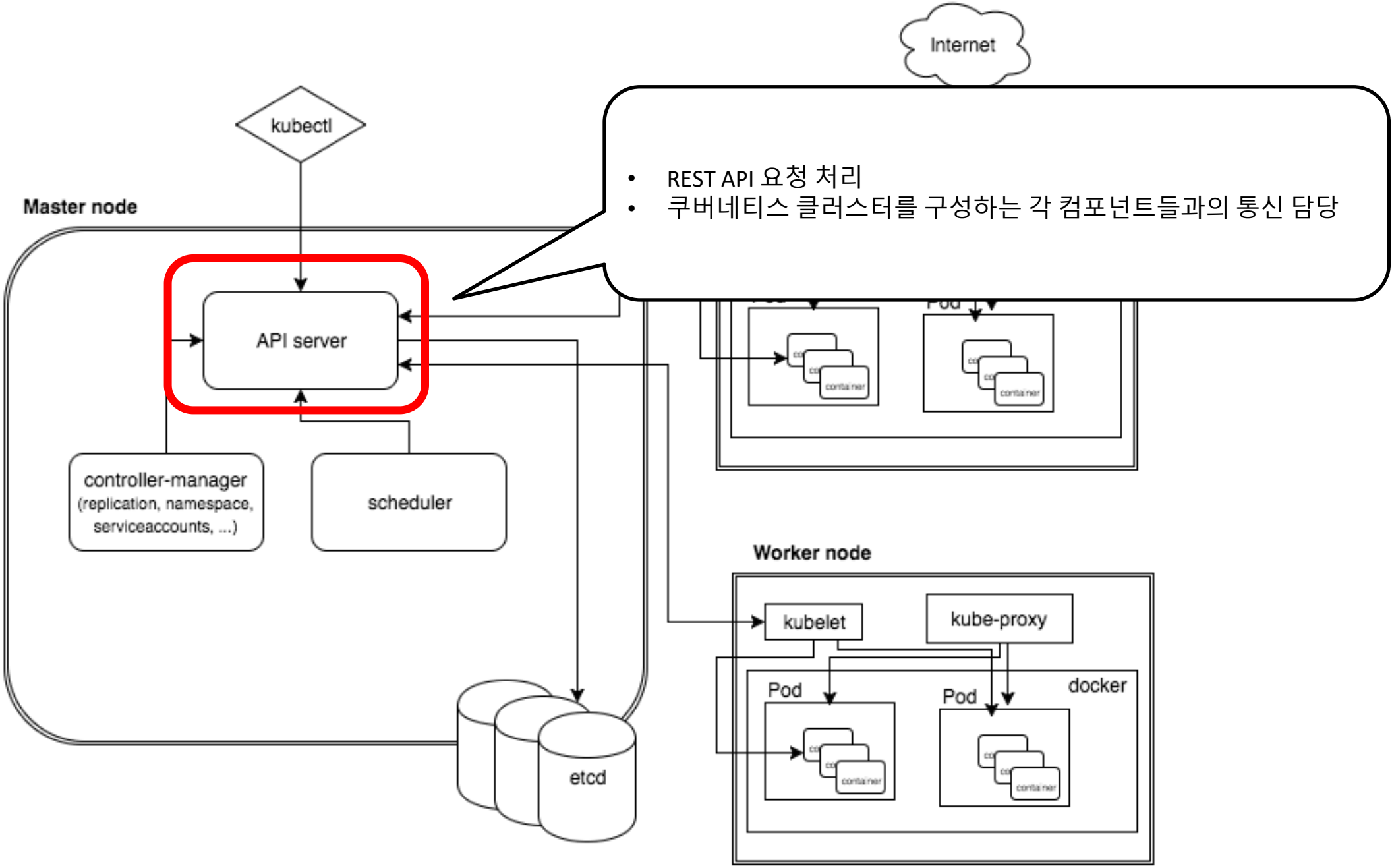
서비스 사용자

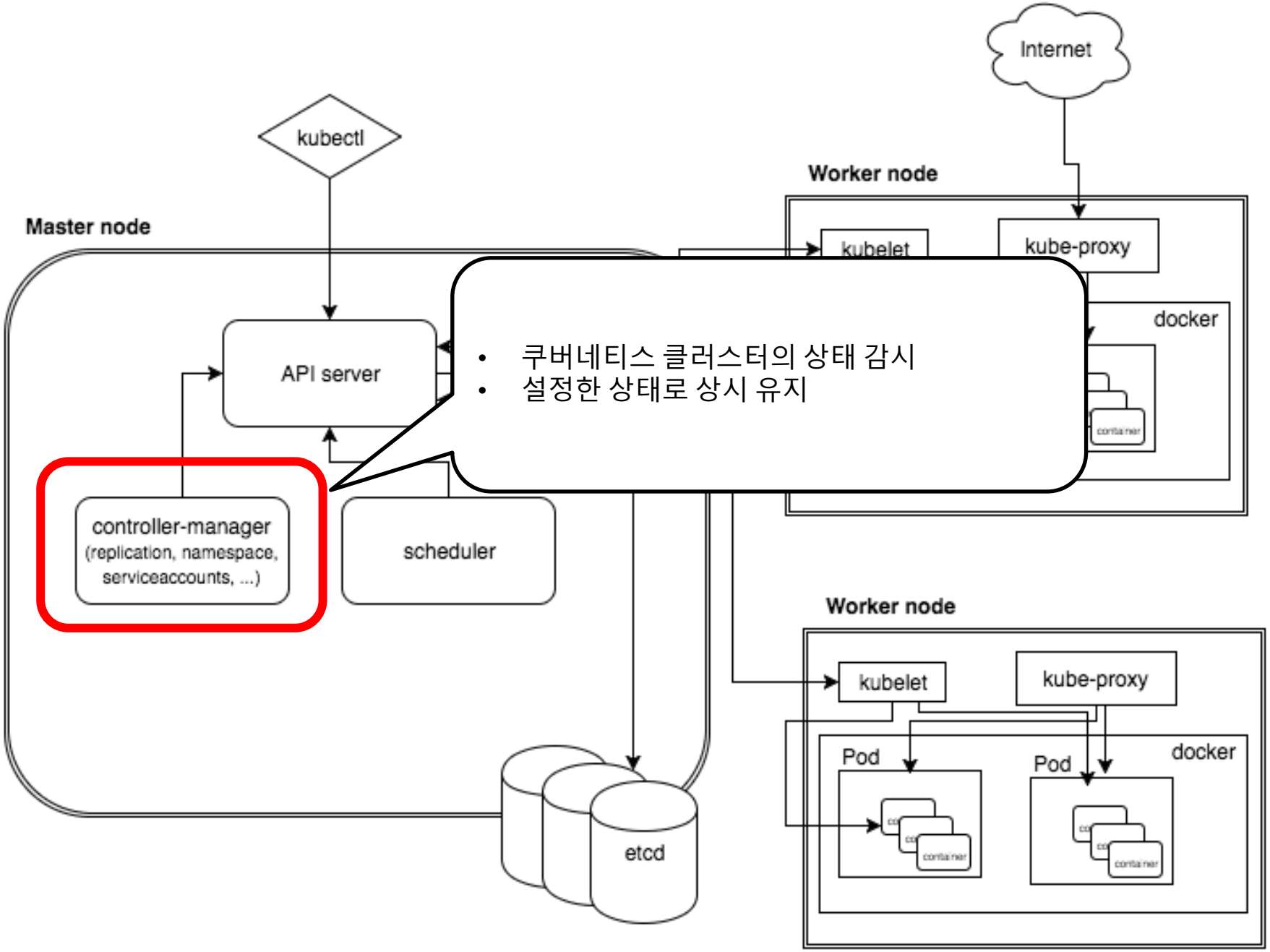
서비스 요

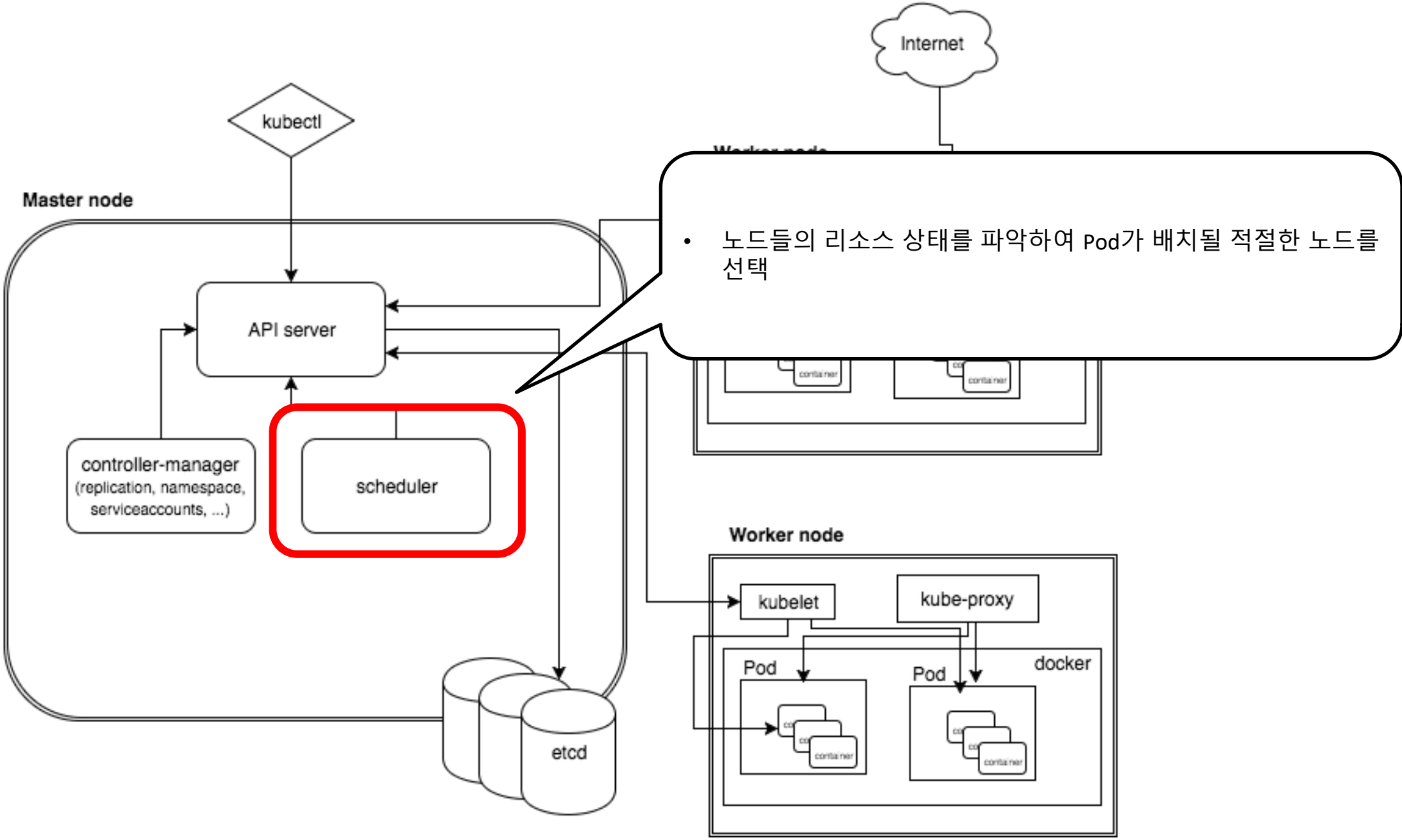
- 단일 호스트를 의미
- Pod를 실행
- kubelet과 kube proxy등 여러 쿠버네티스 컴포넌트를 실행


Internet
kubectl
Worker node
Master node
kubelet
kube-proxy
Pod
Pod
docker
API server
container
container
controller-manager
(replication, namespace,
serviceaccounts, ...)
scheduler
Worker node
kubelet
kube-proxy
Pod
Pod
docker
container
container
etcd

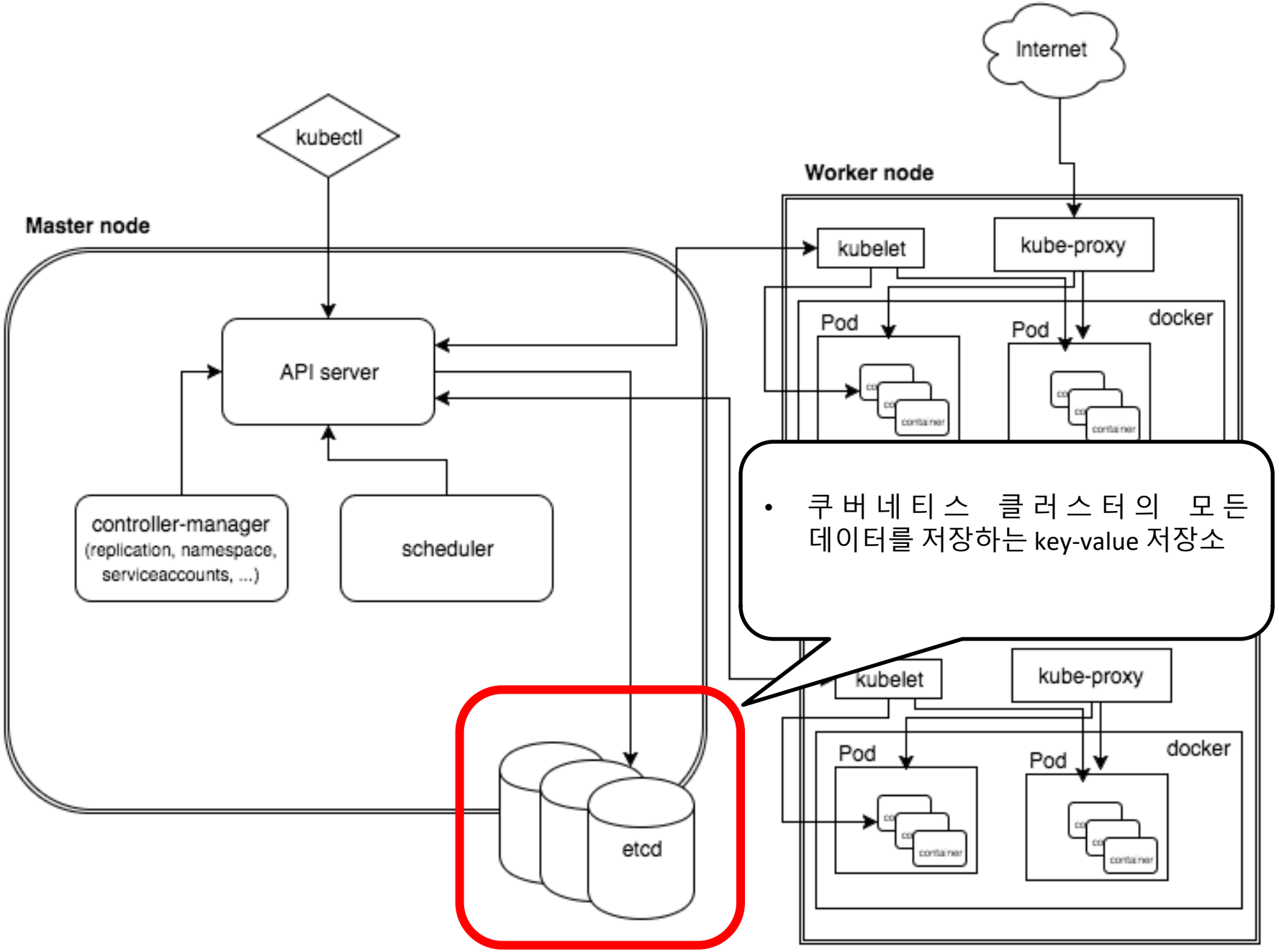

kubectl
• Kubectl은 쿠버네티스 API를 사용해서 클러스터와 상호작용
Master node
API server
controller-manager
(replication, namespace,
serviceaccounts, ...)
scheduler
etcd
kubelet
kube-proxy
Pod
Pod
docker
container
container
Worker node
kubelet
kube-proxy
Pod
Pod
docker
container
container


Internet
kubectl
Master node
API server
controller-manager
(replication, namespace,
serviceaccounts, ...)
scheduler
etcd
Worker node
kubelet
kube-proxy
Pod
Pod
docker
container
container
container
container
container
container
• REST API 요청 처리
• 쿠버네티스 클러스터를 구성하는 각 컴포넌트들과의 통신 담당

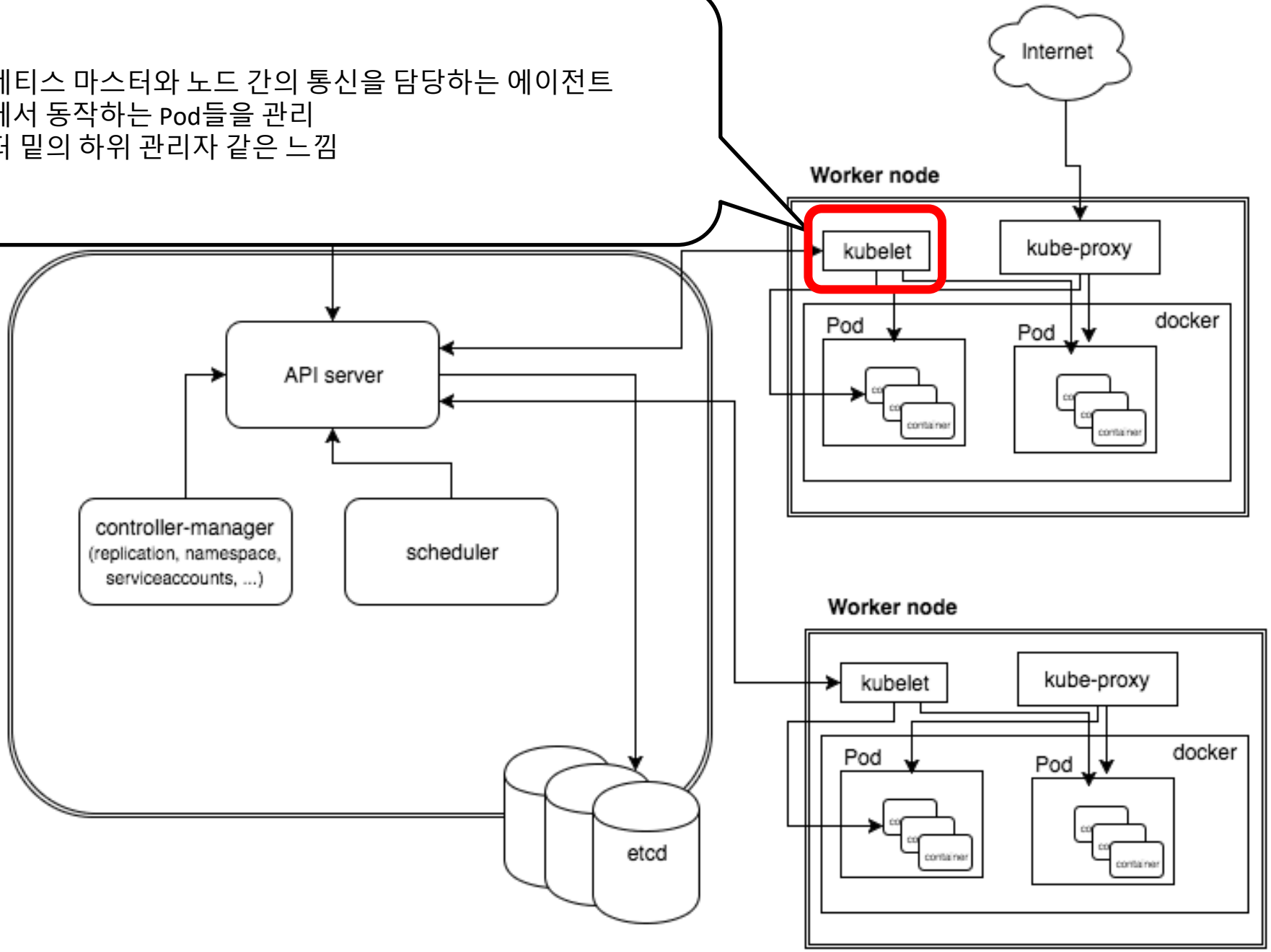

Internet
kubectl
Worker node
Master node
kubelet
kube-proxy
docker
API server
• 쿠버네티스 클러스터의 상태 감시
• 설정한 상태로 상시 유지
controller-manager
(replication, namespace,
serviceaccounts, ...)
scheduler
Worker node
kubelet
kube-proxy
Pod
Pod
docker
etcd

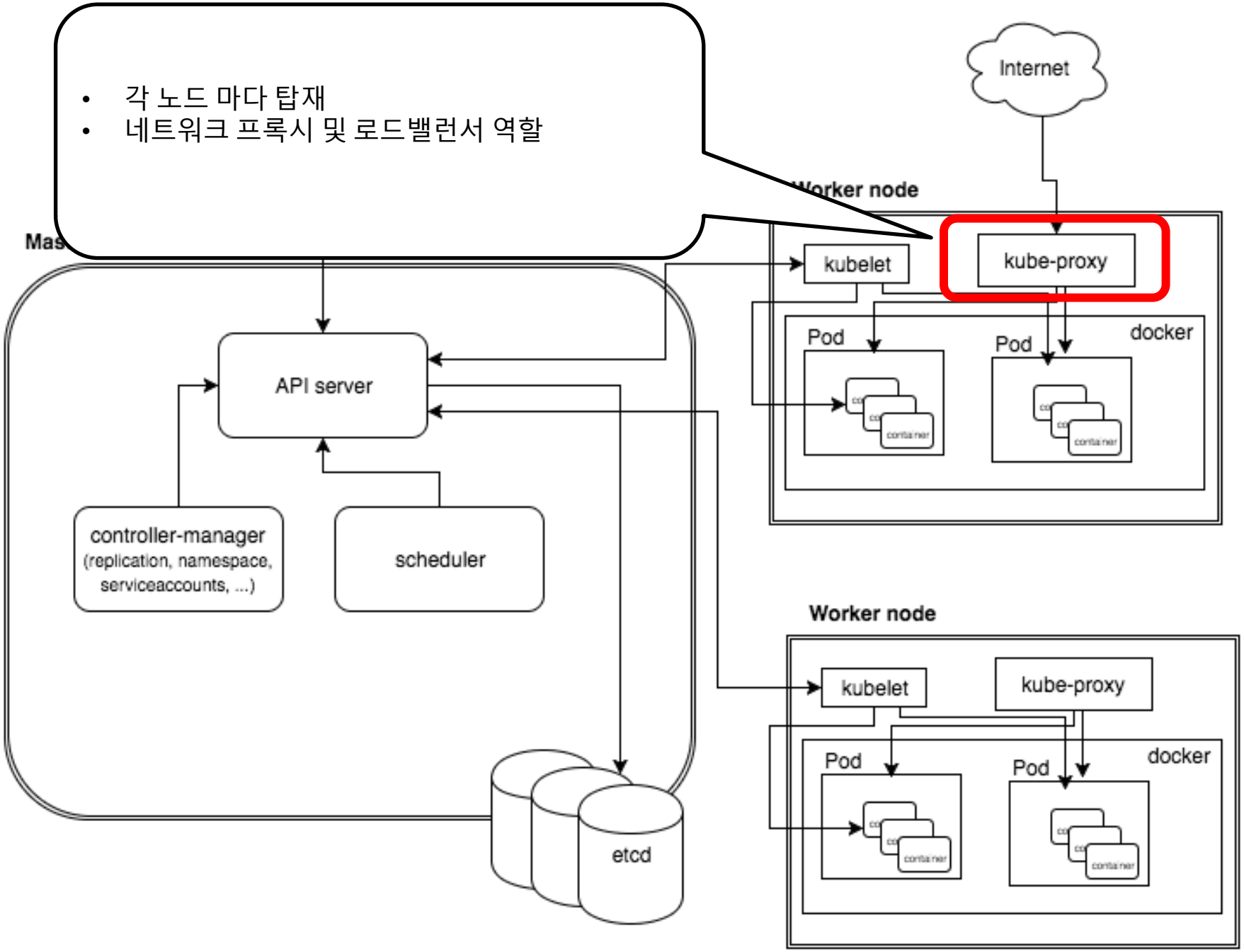


- 노드들의 리소스 상태를 파악하여 Pod가 배치될 적절한 노드를 선택

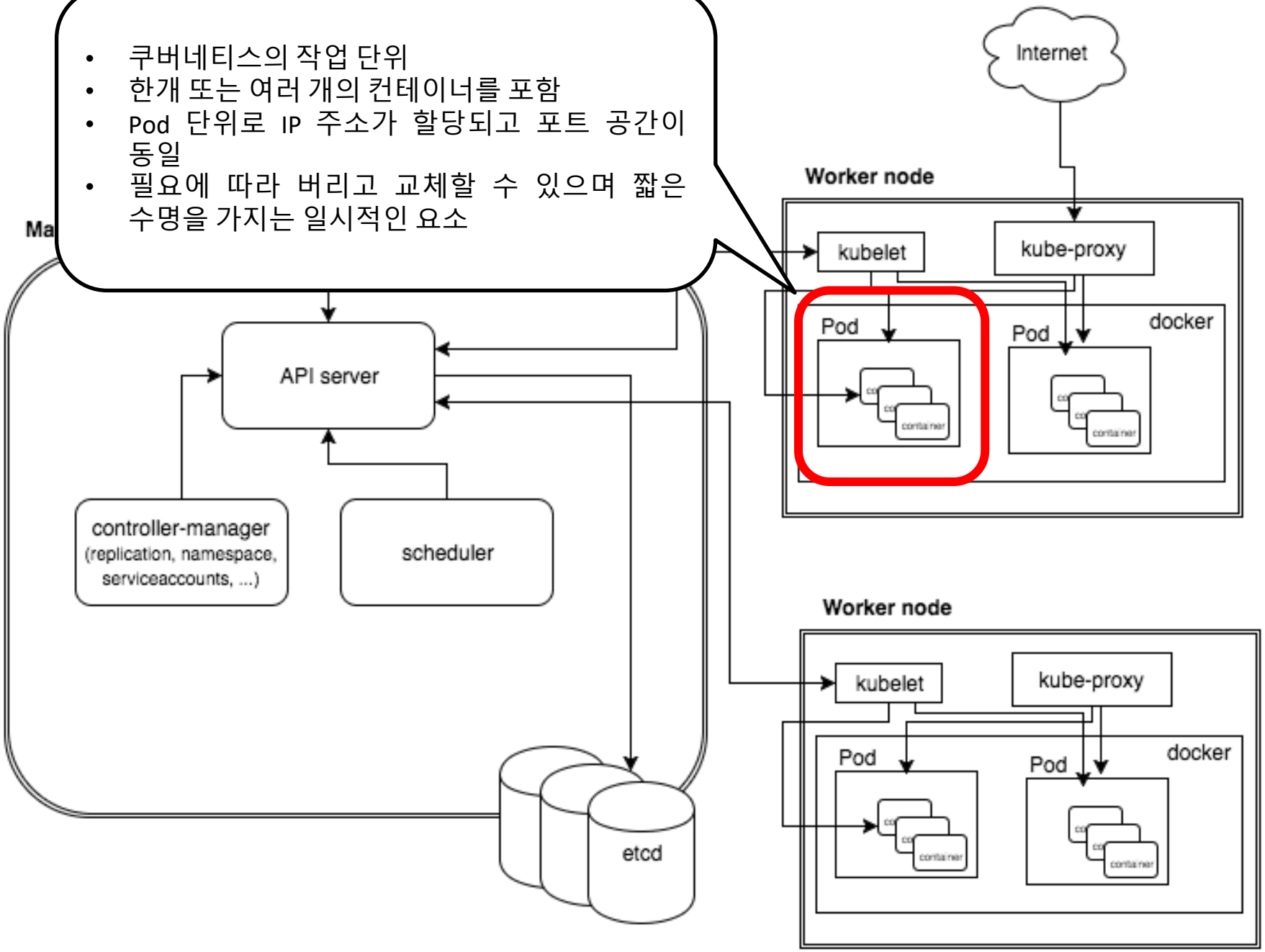

Internet
kubectl
Worker node
Master node
kubelet
kube-proxy
Pod
Pod
docker
API server
controller-manager
(replication, namespace,
serviceaccounts, ...)
scheduler
etcd
• 쿠버네티스 클러스터의 모든 데이터를 저장하는 key-value 저장소

- 쿠버네티스 마스터와 노드 간의 통신을 담당하는 에이전트
- 노드에서 동작하는 Pod들을 관리
- 마스터 밑의 하위 관리자 같은 느낌

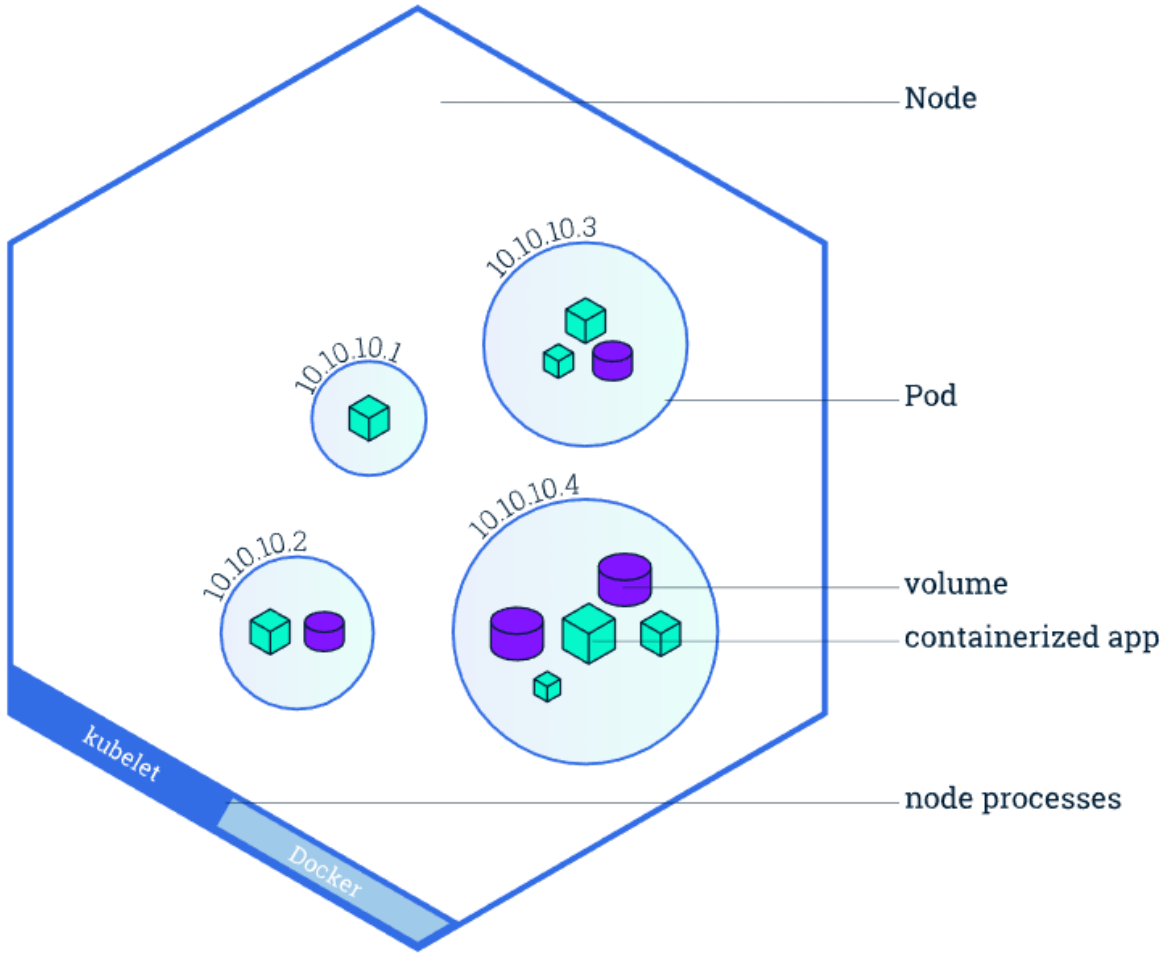



각 노드 마다 탑재
네트워크 프록시 및 로드밸런서 역할
Internet
Worker node
kubelet
kube-proxy
Pod
Pod
docker
container
container
API server
controller-manager
(replication, namespace,
serviceaccounts, ...)
scheduler
etcd
Worker node
kubelet
kube-proxy
Pod
Pod
docker
container
container

- 쿠버네티스의 작업 단위
- 한개 또는 여러 개의 컨테이너를 포함
- Pod 단위로 IP 주소가 할당되고 포트 공간이 동일
- 필요에 따라 버리고 교체할 수 있으며 짧은 수명을 가지는 일시적인 요소



Node 내 Pod 구성


Node
10.10.10.3
10.10.10.1
Pod
10.10.10.4
10.10.10.2
volume
containerized app
kubelet
Docker
node processes

복잡하니까 이것만 기억합시다!

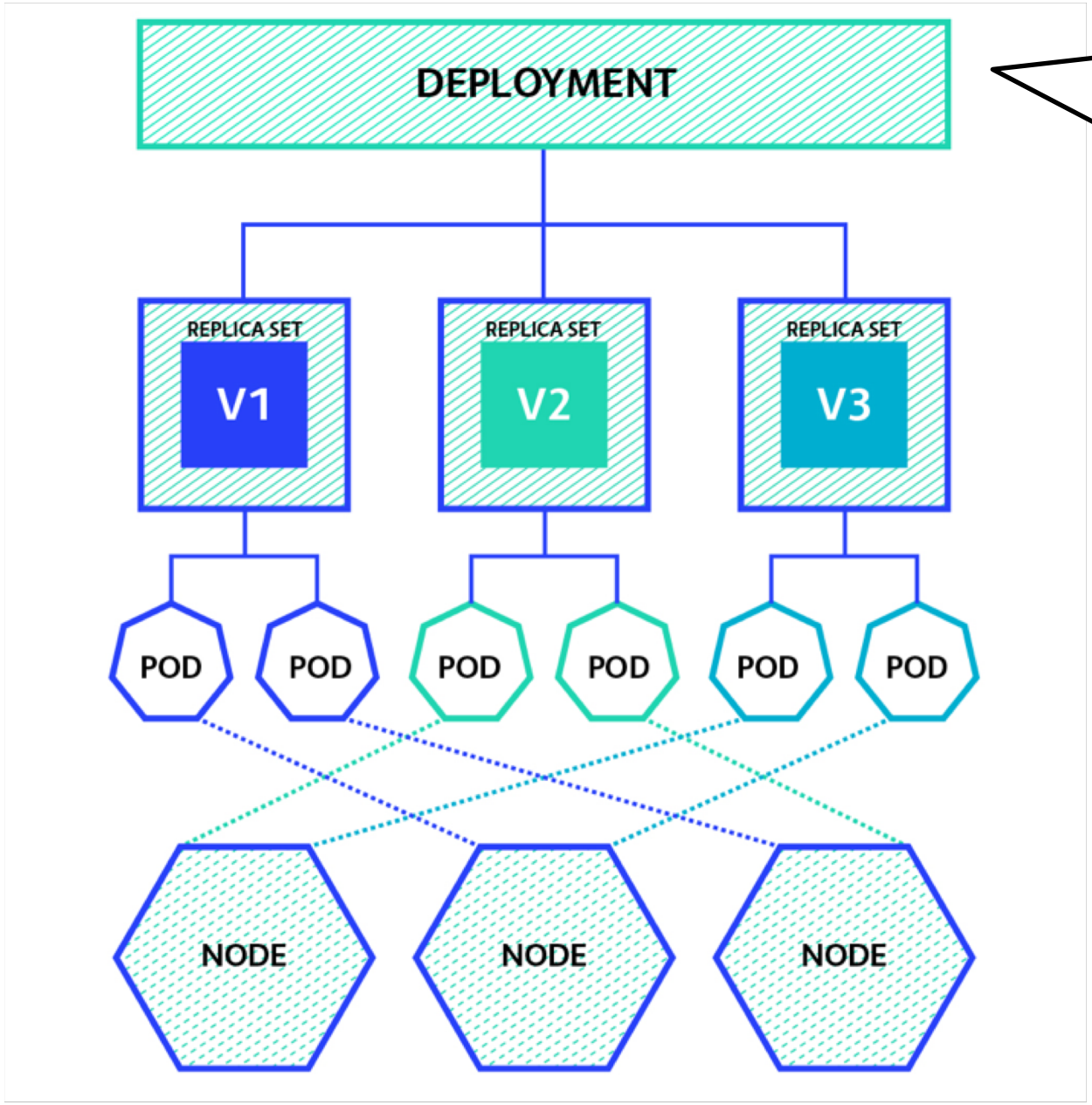
쿠버네티스 클러스터는
마스터 노드와 워커노드로 구성!

쿠버네티스 클러스터에서 작업의 단위는 **Pod**!

쿠버네티스 클러스터에 명령은 kubectl로!

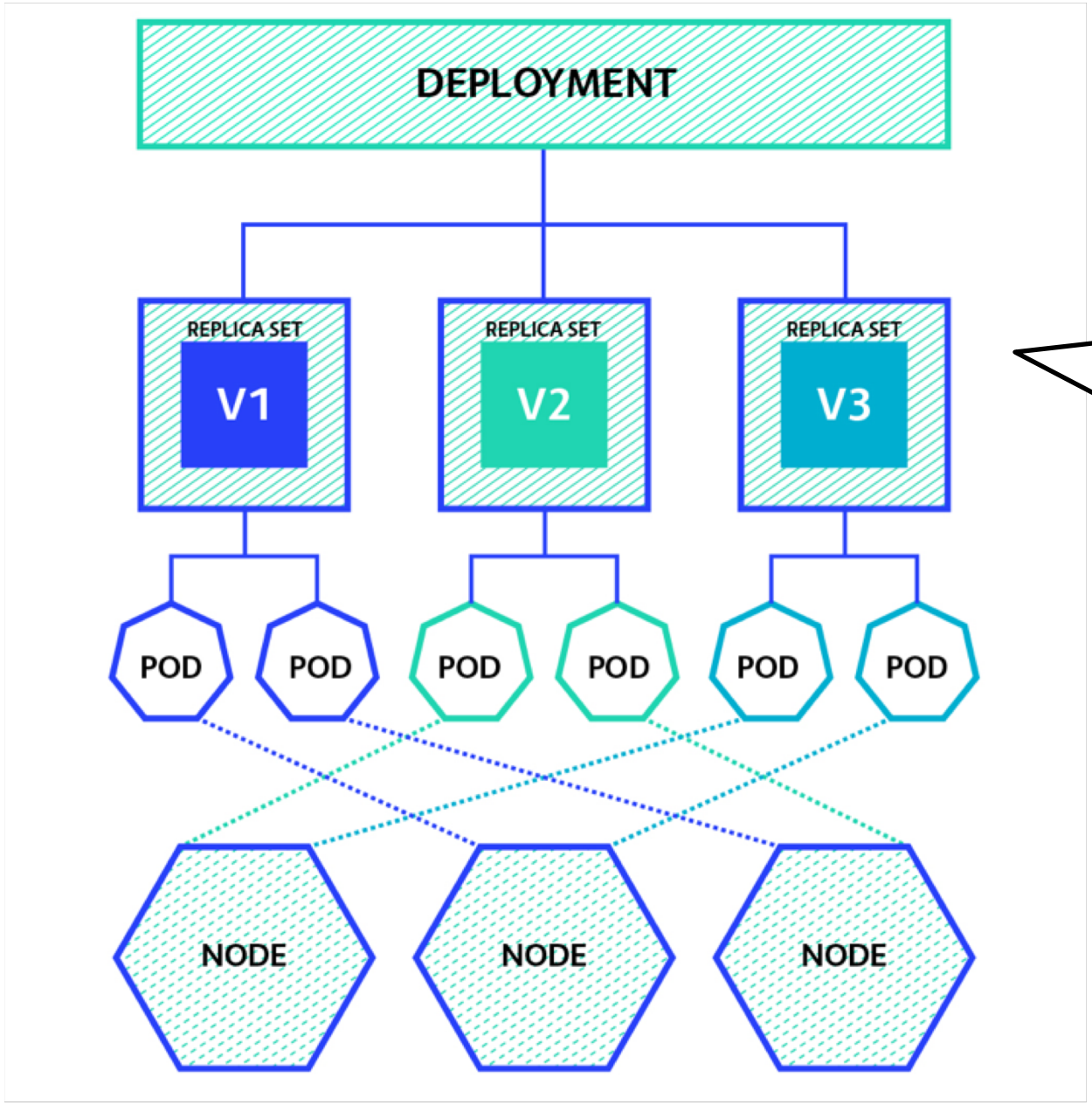
실습

Kubernetes 실습 #1



Deployment 설정에는 Pod를 어떻게 구성하고 업데이트를 수행할지 등에 대해 정의

출저 : https://thenewstack.io/kubernetes-deployments-work/



Pod 가 만 들 어 지 면 replicaset 는 지속적으로 Pod를 모니터링하고, Pod가 다운되거나 삭제되면 적절한 노드에 다시 생성

출저 : https://thenewstack.io/kubernetes-deployments-work/

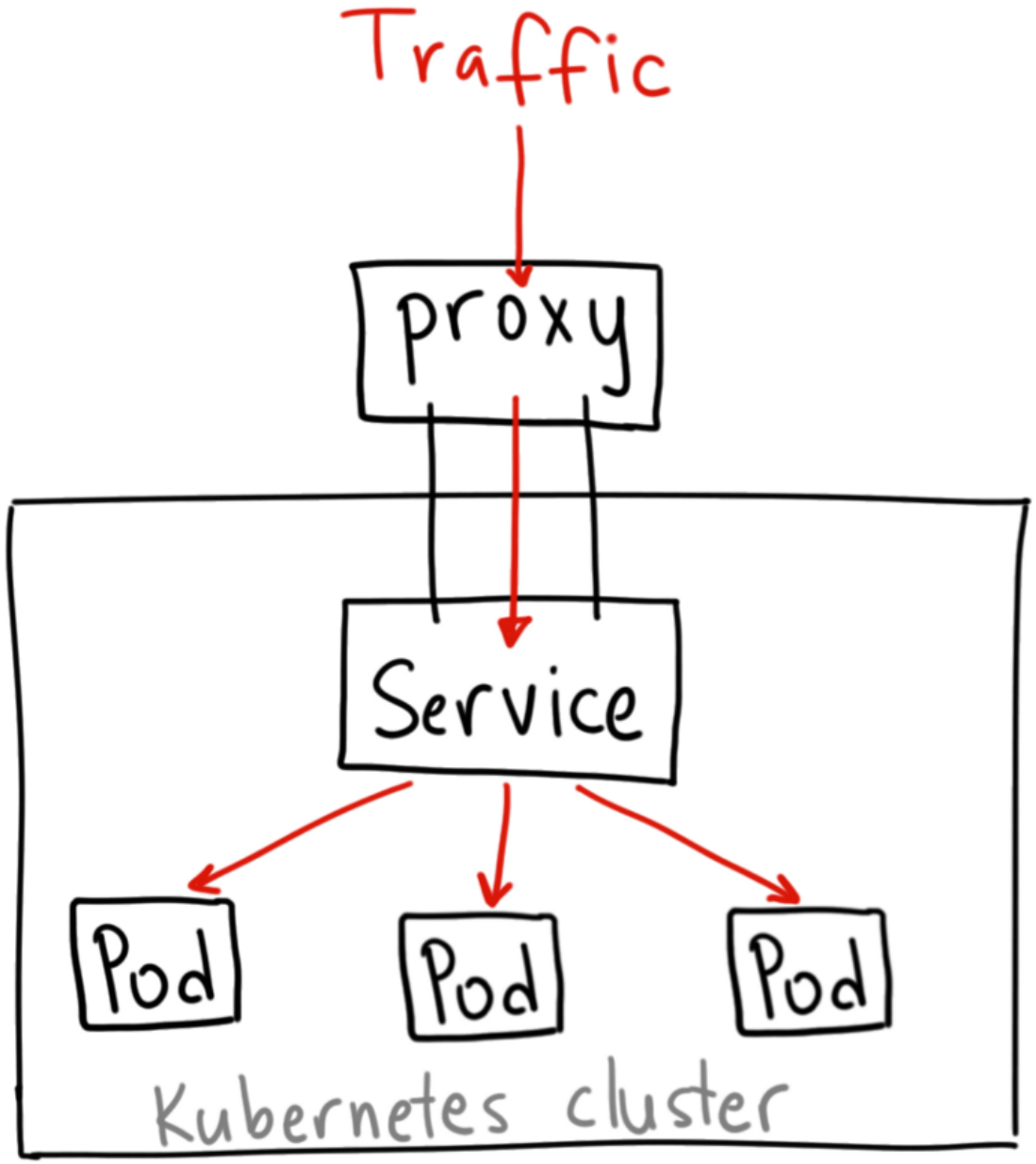
서비스

- 지금까지의 Pod, Node로는 외부에서의 접속이 불가능
- 서비스를 사용해서 쿠버네티스 클러스터 외부로 어플리케이션을 노출
  - 사용자가 외부 자원이나 가상 IP를 통해 서비스에 접근 가능
- OSI 7계층의 3계층(TCP/UDP)에서 동작
- 서비스에 속한 Pod들은 LabelSelect로 판단됨
- 로드밸런싱 및 서비스 디스커버리 수행

서비스에서 IP 노출 방법

ClusterIP

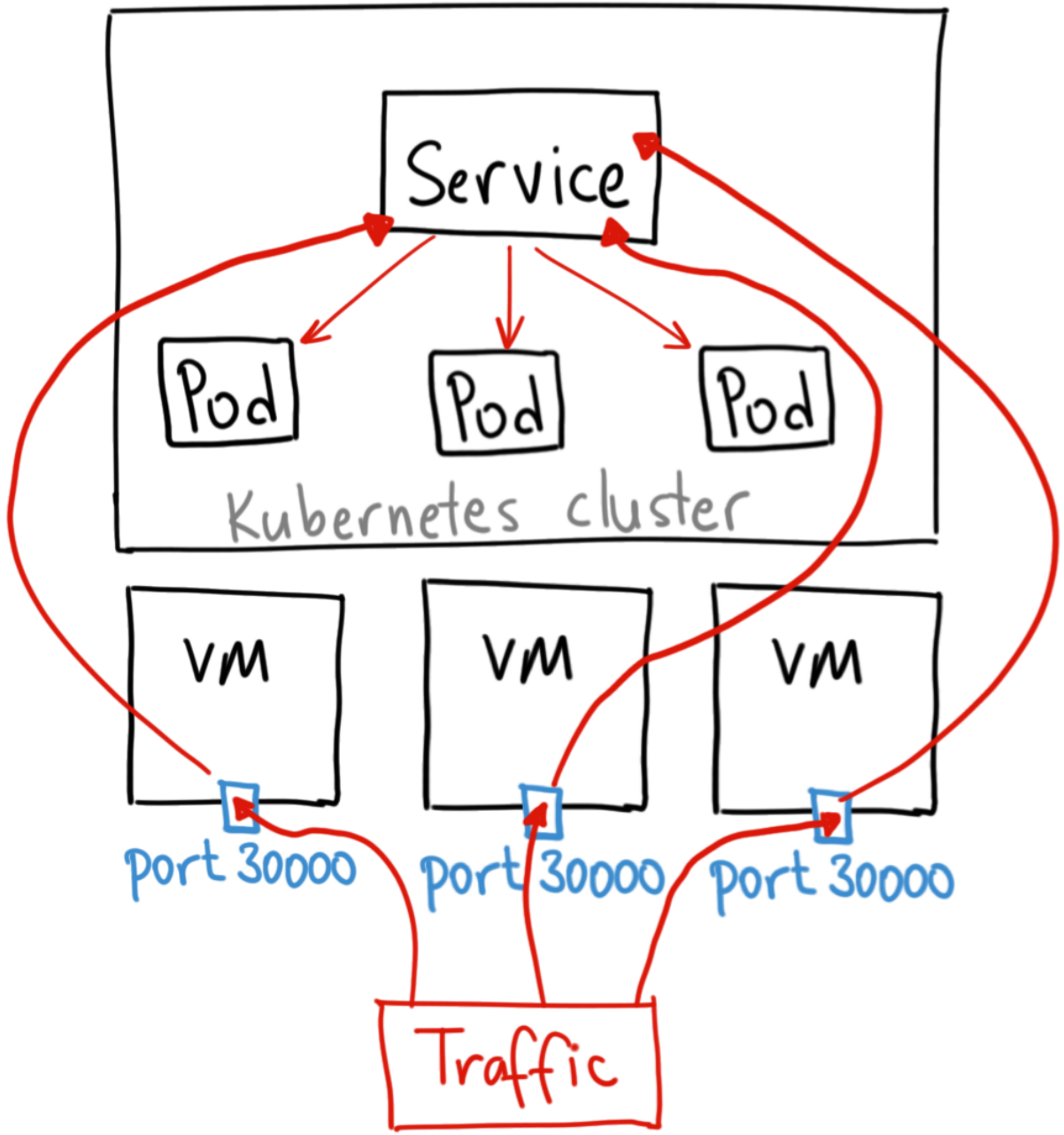
- Cluster 내부 통신
- 외부에서 액세스 불가능
- kube proxy를 통해 외부 오픈 가능



출저 : https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0

NodePort

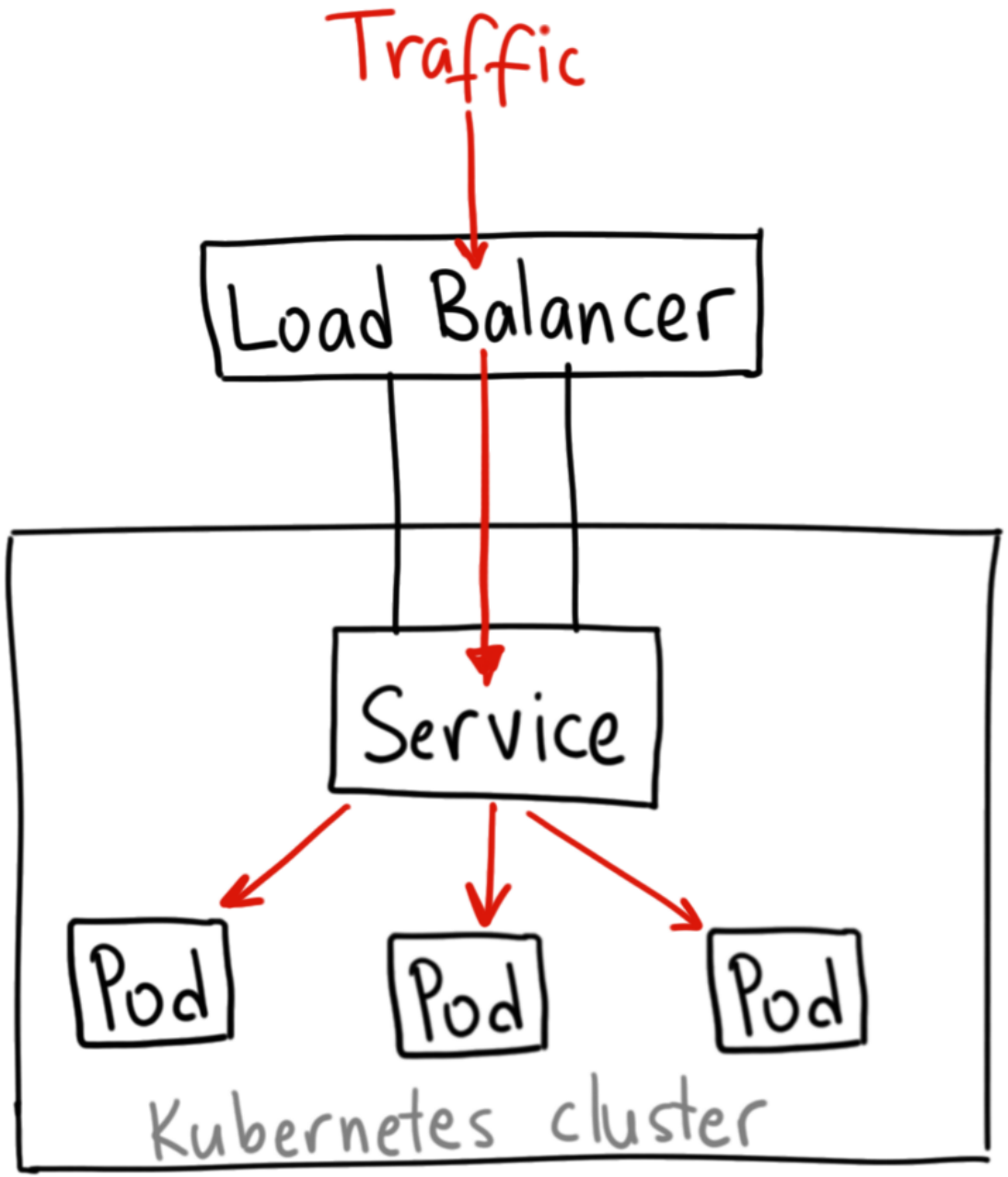
- NAT를 사용하여 각 노드의 Port를 오픈
- 30000~32767 범위의 포트 할당
- 명시적으로 지정하지 않으면 범위 내에서 랜덤 할당



출저 : https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0

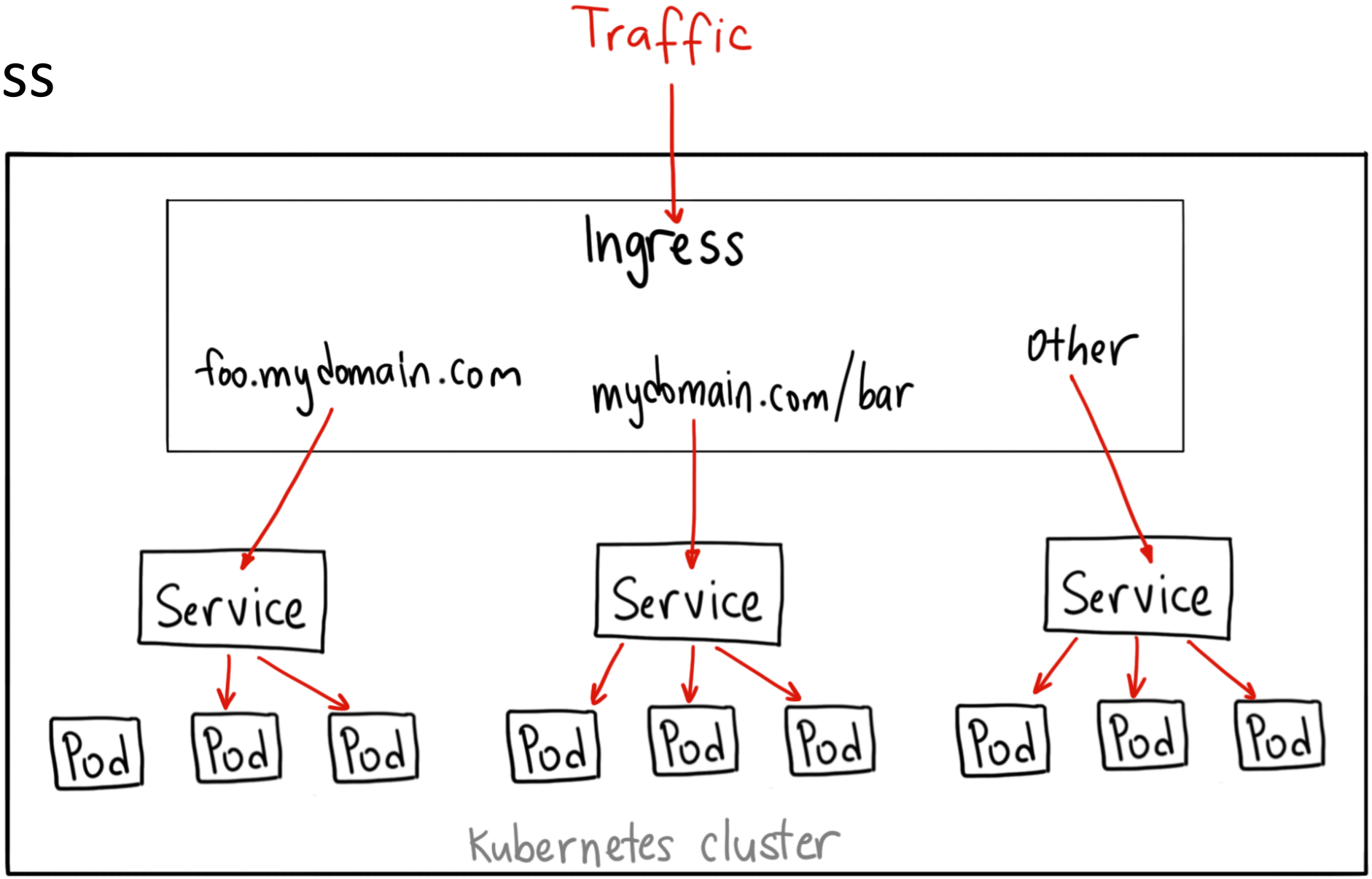
LoadBalancer

- External 로드밸런서를
  만들어서 고정된 외부 IP 배정



출저 : https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0

Ingress



출저 : https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0

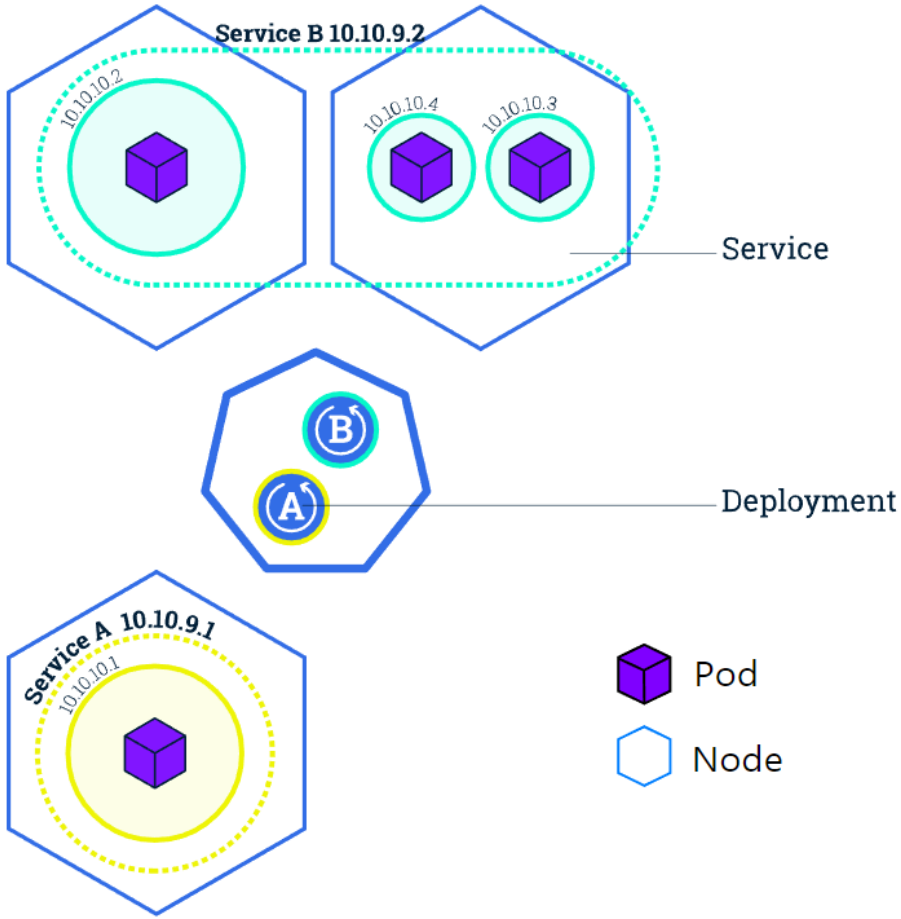
라벨 (label)

- key-value 쌍의 객체 집합이며 객체의 식별에 사용
- Pod를 그룹화할 때 사용
- 복제 컨트롤러(Replication Controller), 복제 집합(Replica set) 등에서 사용

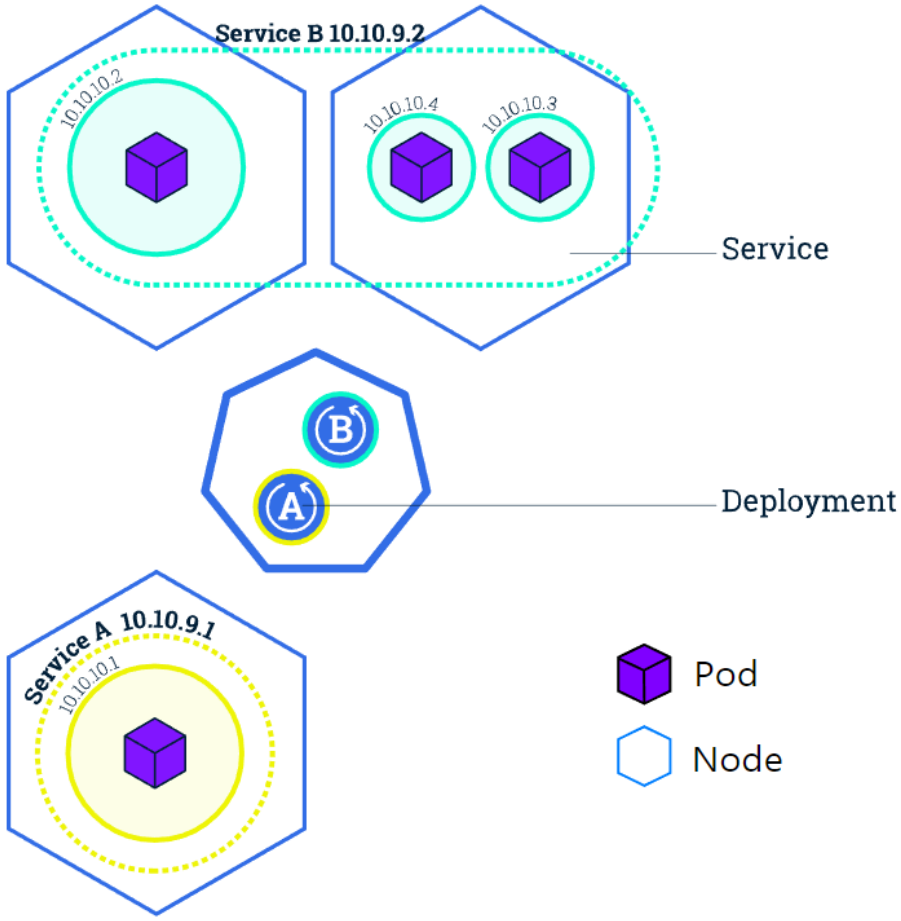
라벨 셀렉터

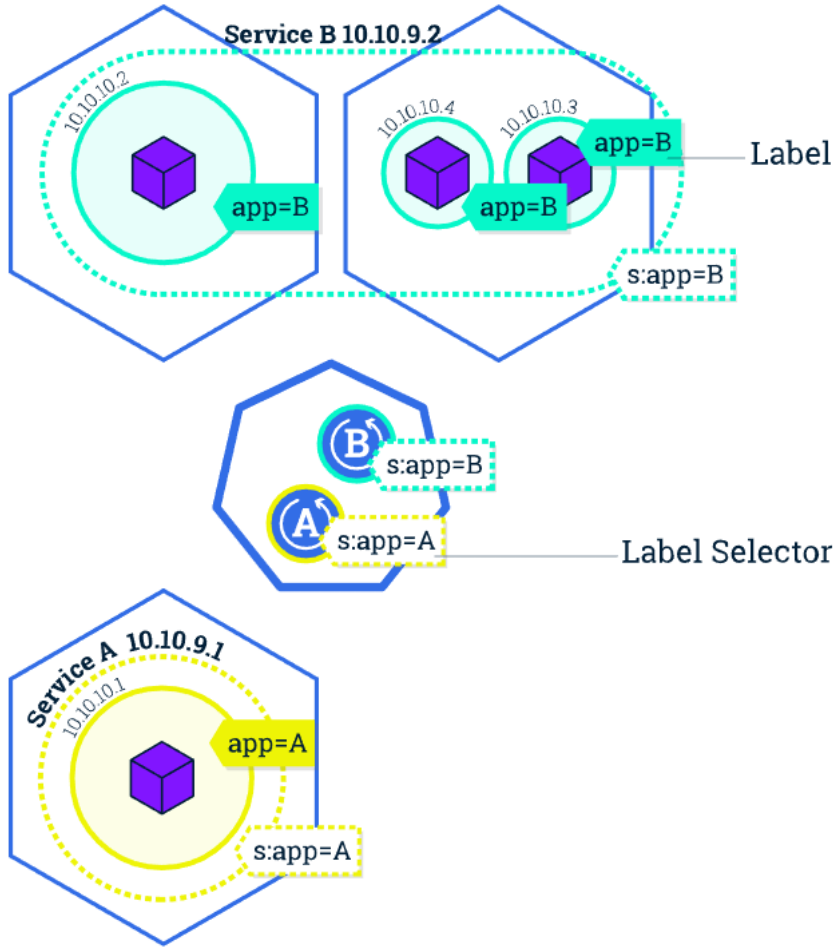
- 라벨을 기반으로 객체를 선택
- 동일한 라벨이 부여된 객체들을 연산자를 통해 일괄 선택 가능
  - ex) 라벨 키가 role이고 값이 webserver인 객체 선택
    - role = webserver
      - = 또는 ==, != 을 통해 라벨에 따른 객체 선택
    - 복수의 조건은 쉼표로 구분
      - role = webserver, application != foo
    - 집합은 다양한 값을 기반으로 객체 선택
      - role in (webserver, backend)

서비스와 라벨


Service B 10.10.9.2
10.10.10.2
10.10.10.4
10.10.10.3
Service
B
A
Deployment
Service A 10.10.9.1
10.10.10.1
Pod
Node

서비스와 라벨


Service B 10.10.9.2
10.10.10.2
10.10.10.4
10.10.10.3
Service
B
A
Deployment
Service A 10.10.9.1
10.10.10.1
Pod
Node

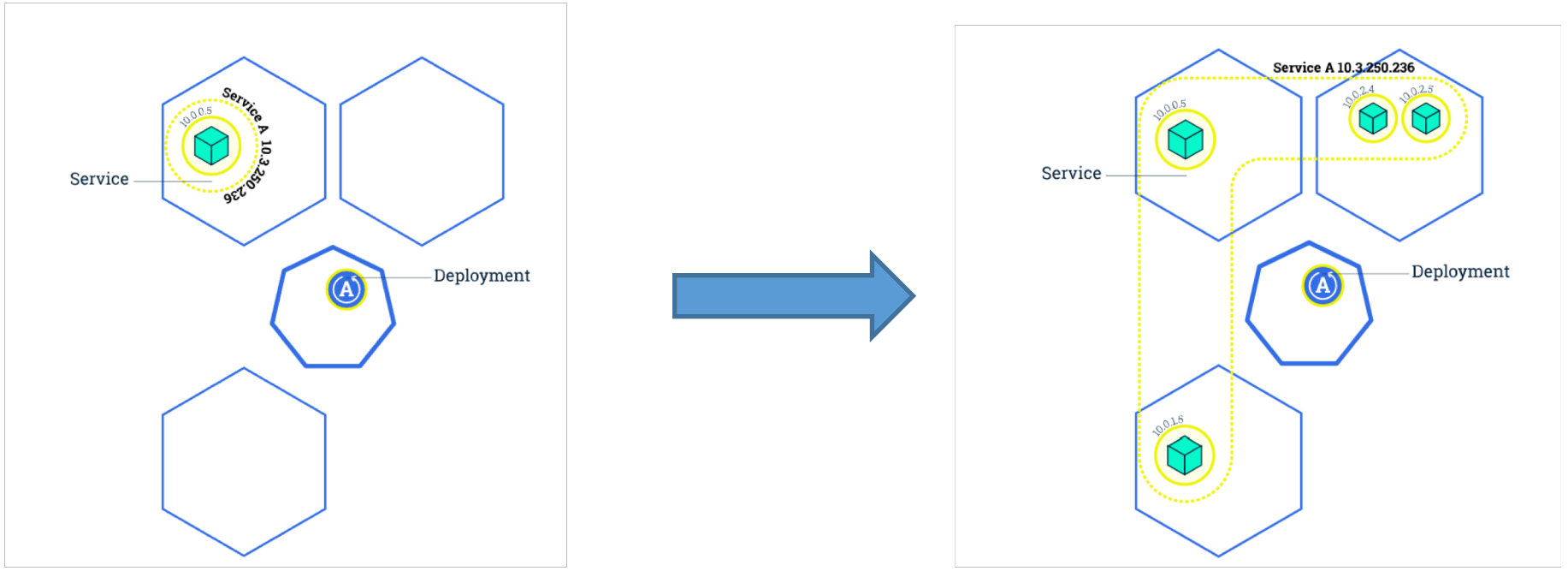

Service B 10.10.9.2
10.10.10.2
app=B
10.10.10.4
10.10.10.3
app=B
app=B
Label
s:app=B
B
s:app=B
A
s:app=A
Label Selector
Service A 10.10.9.1
10.10.10.1
app=A
s:app=A

복제 컨트롤러와 복제 집합

- 라벨 셀렉터로 식별된 Pod 그룹을 관리하고, 특정 수만큼 실행 중인지 항상 확인
- 복제 컨트롤러는 이름의 동일 여부로 구성원을 확인
- 복제 집합은 집합 기반 선택을 사용
- 지정한 수의 Pod가 항상 실행되도록 보장
  - 문제가 발생하여 Pod가 일정 수준 이하로 내려가면 쿠버네티스는 새로운 인스턴스를 시작

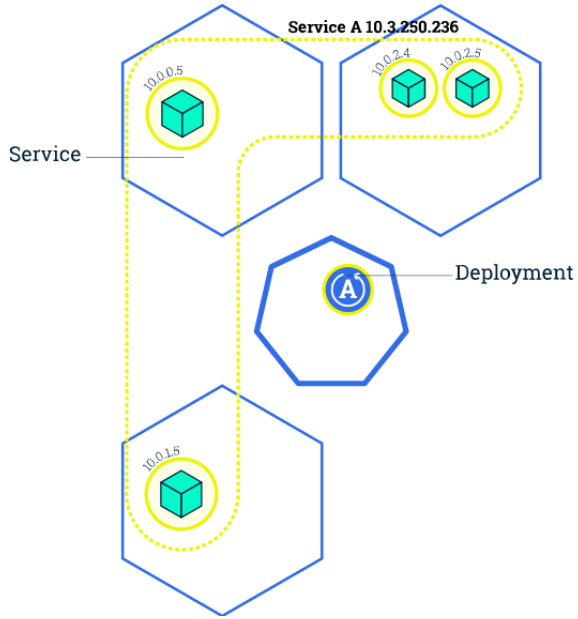
스케일링

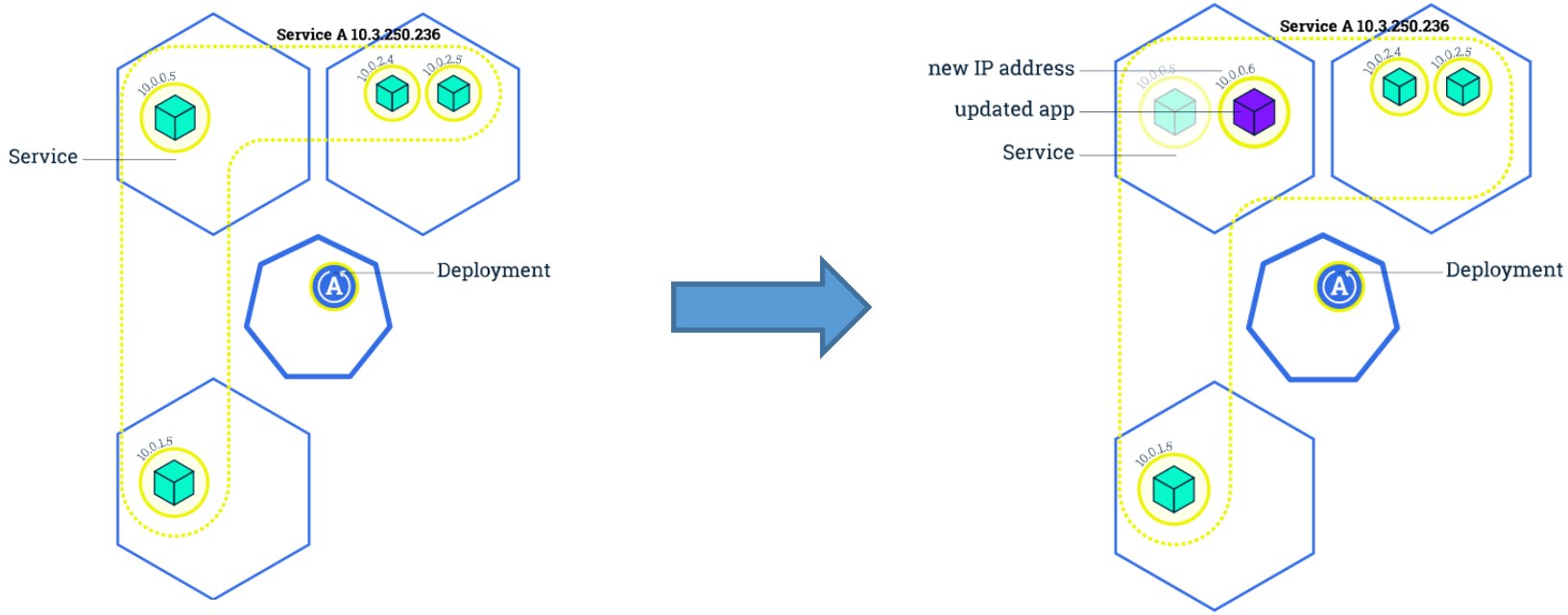
- kubectl을 사용해서 스케일링
- 트래픽이 증가하면 자동으로 스케일 아웃 수행
- 마스터가 알아서 노드에 Pod 확장 및 축소

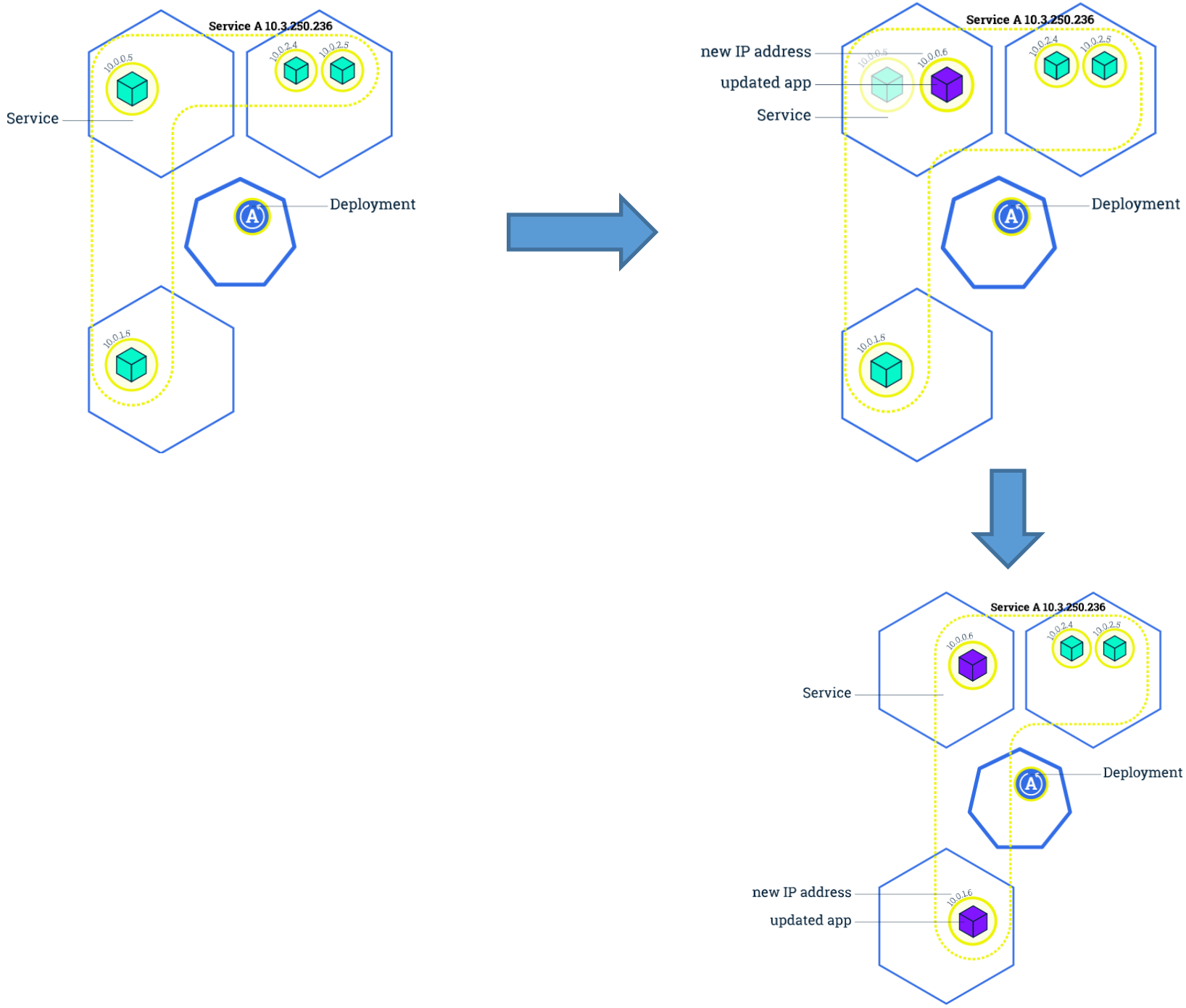


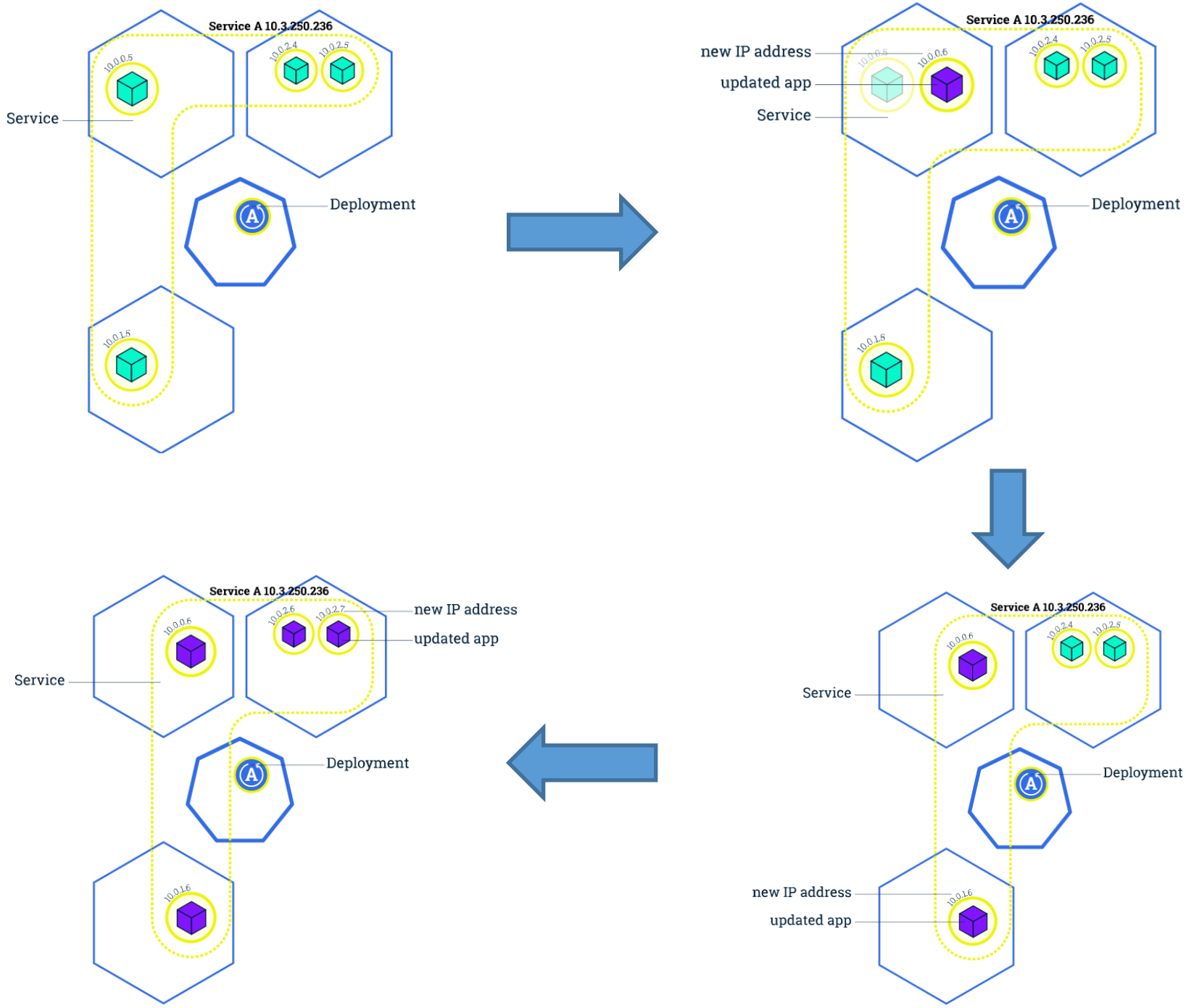
롤링 업데이트

- Kubectl을 사용해서 수행
- 무정지 배포
- 점진적으로 Pod를 하나씩 업데이트


Service A 10.3.250.236
10.0.0.5
10.0.2.4
10.0.2.5
Service
Deployment
10.0.1.5


Service A 10.3.250.236
10.0.0.5
10.0.2.4
10.0.2.5
Service
Deployment
10.0.1.5
new IP address
updated app
Service
Service A 10.3.250.236
10.0.0.5
10.0.0.6
10.0.2.4
10.0.2.5
Deployment
10.0.1.5


Service A 10.3.250.236
10.0.0.5
10.0.2.4
10.0.2.5
Service
Deployment
10.0.1.5
new IP address
updated app
Service
Service A 10.3.250.236
10.0.0.5
10.0.0.6
10.0.2.4
10.0.2.5
Deployment
10.0.1.5
Service A 10.3.250.236
10.0.0.6
10.0.2.4
10.0.2.5
Service
Deployment
new IP address
updated app
10.0.1.6


Service A 10.3.250.236
10.0.0.5
10.0.2.4
10.0.2.5
Service
Deployment
10.0.1.5
new IP address
updated app
10.0.0.6
10.0.2.6
10.0.2.7
10.0.1.6

실습

Kubernetes 실습 #2

Thank you