

# 도커 쿠버네티스

**20201105**

**angelo.davinci**

# 1. 인프라 배경

# 크게 2가지 관점



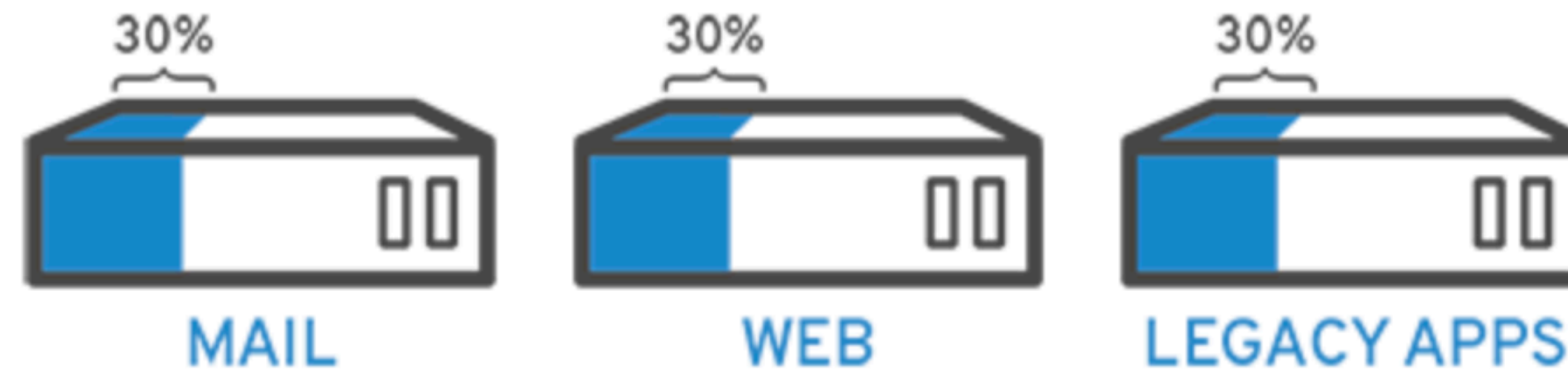
물리머신 당 1태스크	물리머신 당 다중 태스크
각 태스크끼리 영향이 없음	한 태스크로 인해 서버가 뺨으면 모든 태스크가 죽음
관리가 용이	내 태스크의 작업(개발/배포/버저닝 등)이 다른 태스크에 영향을 끼칠 수 있음
독립적인 환경	환경이 꼬일 수 있는 가능성 높아짐 (ex 메일은 java 1.4, 앱은 java 1.7)
기타 Micro Service Architecture가 주는 장점들	그렇지 않았을때의 단점

## 크게 2가지 관점



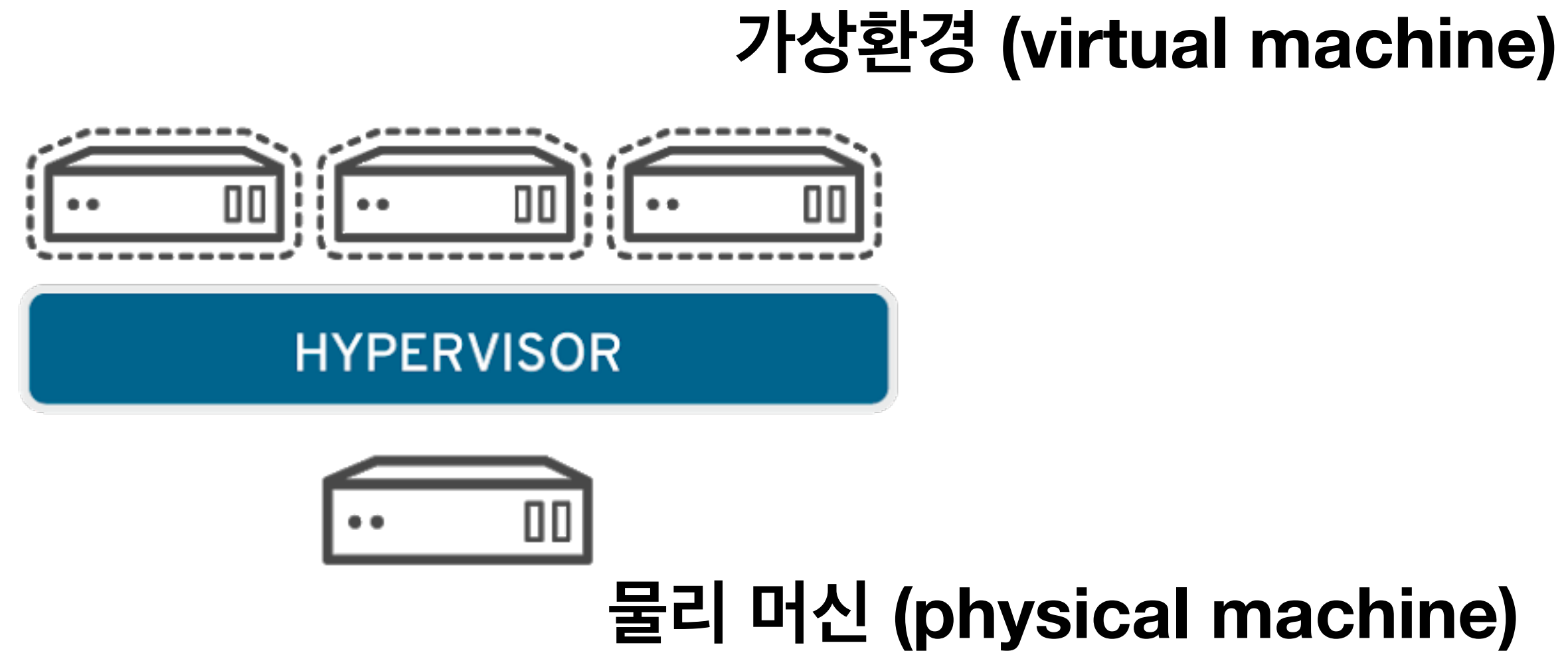
then,  
물리 머신 1대 당 1 태스크가 국룰

# But!



- 물리 장비가 비쌘 ㅌㅌ
  - 근데 장비 당 1태스크로 구성 시 자원을 100% 활용하지 못하는 경우도 많음 (돈낭비 자원낭비)
  - 스케일링이 힘들
- (앗? 사용량이 오르네? -> 스펙 정하고, 서버 주문하고, 배달오고, 세팅하고, 어플리케이션 올리고 ..)
- 밴더(vendor)에 종속적 (웹서비스가 IBM 서버면 IBM 서버로 주문해야 호환가능요~ ㅌㅌ)

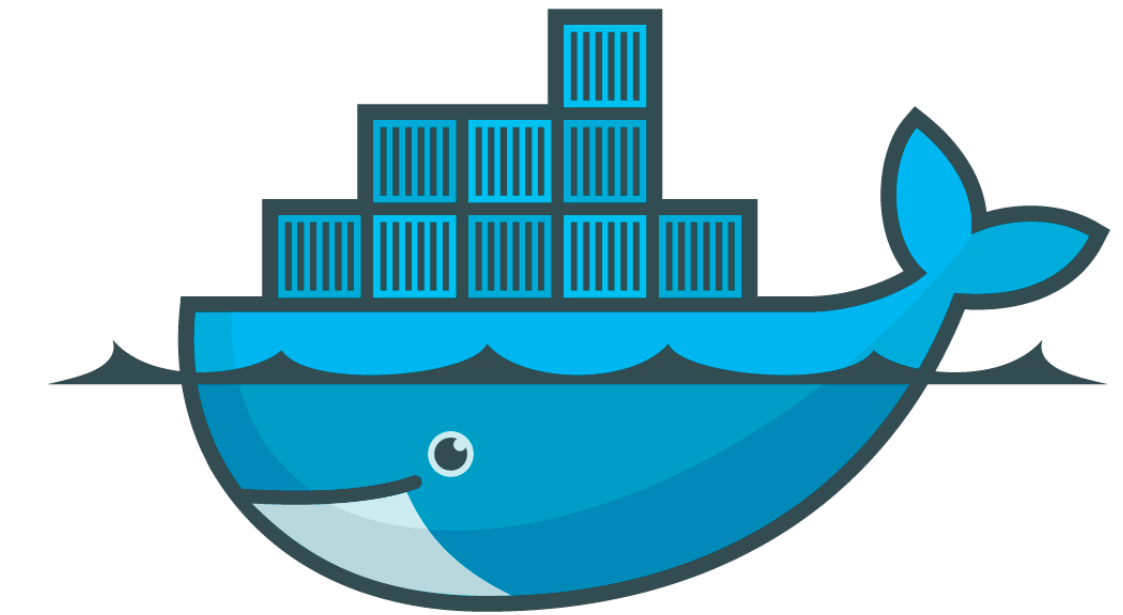
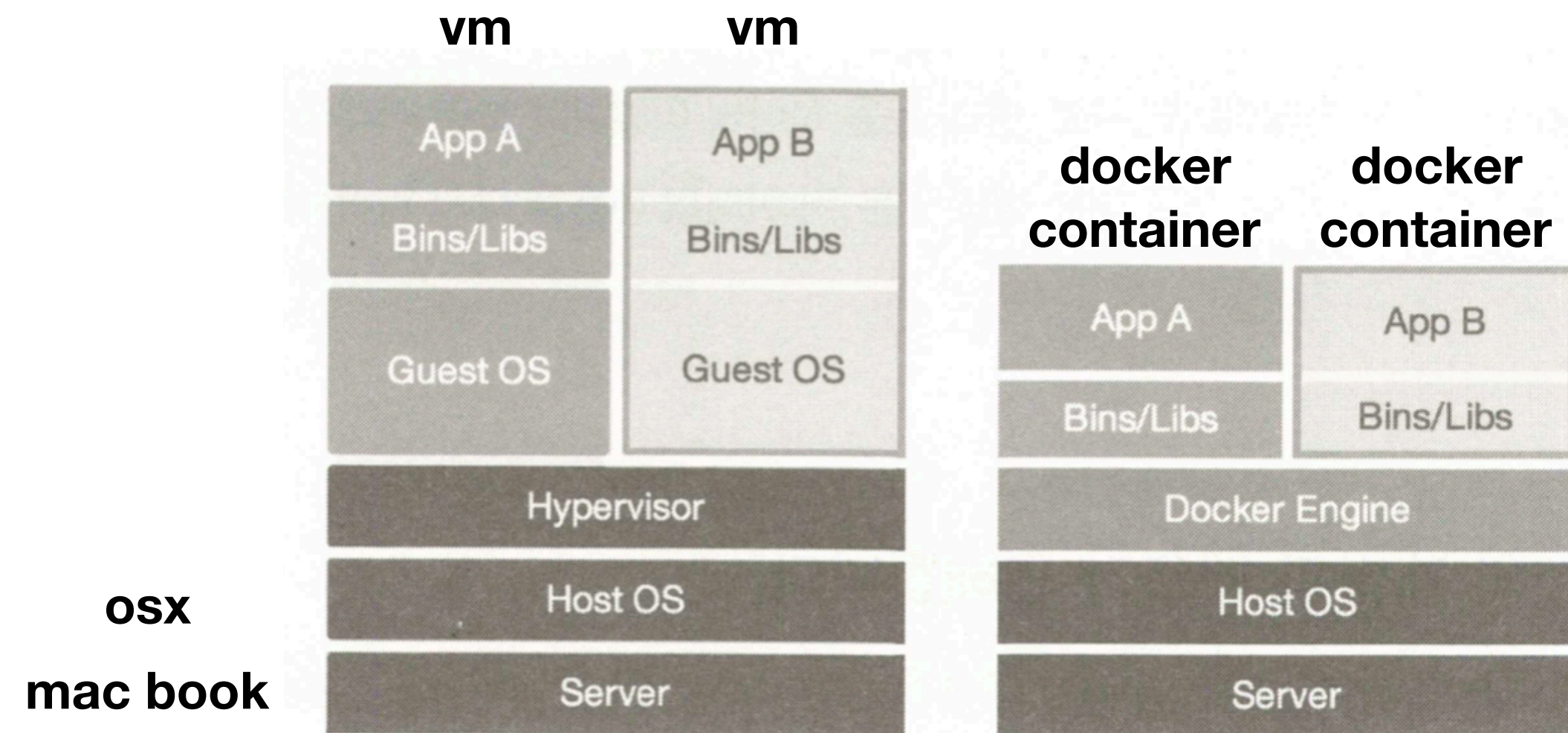
# 가상화로 해결



- 1개의 물리 머신에, 개별 태스크를 가상환경으로 구축하여 필요한만큼 올리는 형태로 자원활용률 up!
- 태스크 환경을 이미지 파일로 관리할 수 있어서 쉽고 빠르게 스케일링 가능
- 태스크 환경 뻘나면 날리고 이미지 다시 올리면 됨 ^^ (따부따부썰)
- 개발환경과 동일한 환경을 서버에 그대로 올릴 수 있음 (로컬에선 잘됐는데 서버에선 왜 안돼 ㄴㄴ)
- **태스크 환경을 파일시스템으로 관리**할 수 있다는 개념이 핵심

## 2. Docker

# VM과 Docker



- 도커 이전에는 Hypervisor라는 가상화 기술 기반의 Virtual Machine이 사실상 표준
- **VM은 os에 필요한 library, kernel을 전부 포함**하고 있어서 크기가 크기 때문에 어플리케이션으로 배포하기에는 부담스러움 (🐷..TT)
- 반면 **docker는 host의 kernel을 공유해 사용**하고, container 안에는 어플리케이션 구동에 필요한 실행파일만 존재하기 때문에 image로 만들어도 가벼움

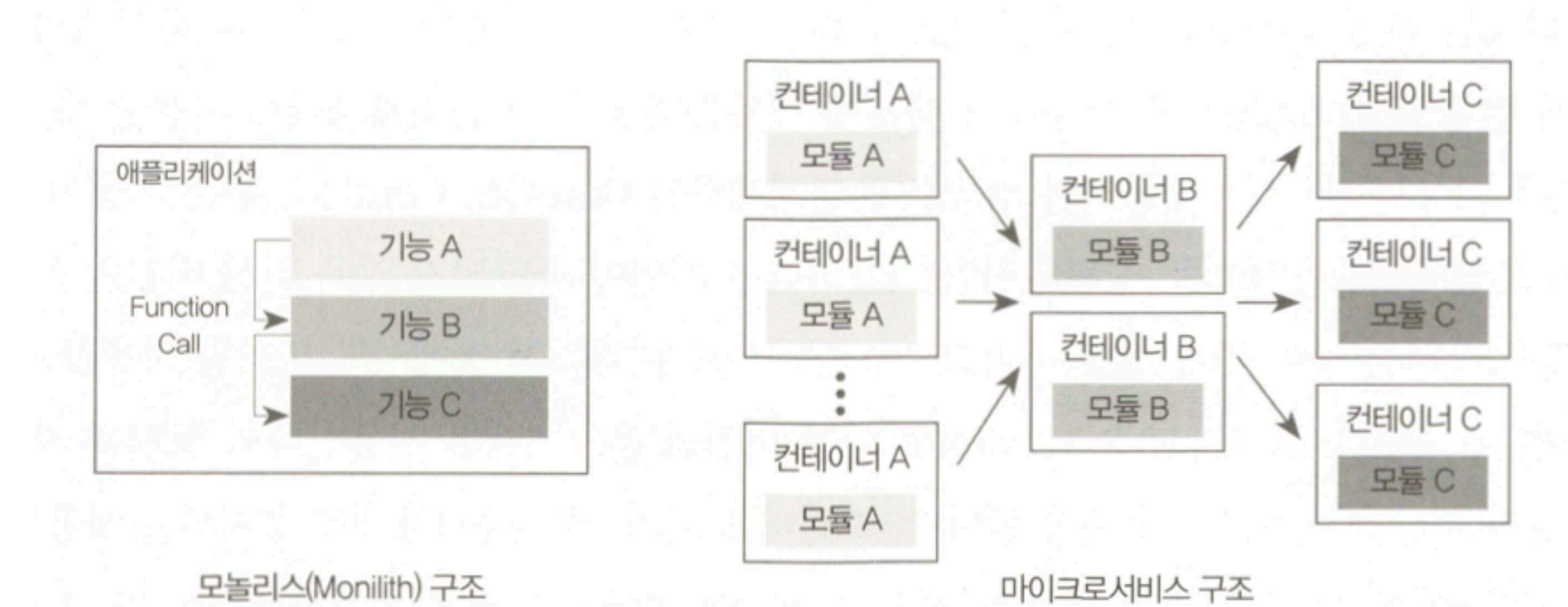


# Docker를 시작해야하는 이유 1

## 어플리케이션 개발과 배포가 편함

- 컨테이너 안에서 지지고 볶고 망가뜨려도 host os에는 no 영향, 컨테이너 다시 만들면 됨!
- 컨테이너 내부에서 여러작업 마친 뒤 운영 환경에 배포하려고 하면, 컨테이너를 '도커 이미지'로 만들어서 올리면 그대로 실행가능
- 심지어 이미지가 가벼워 ><

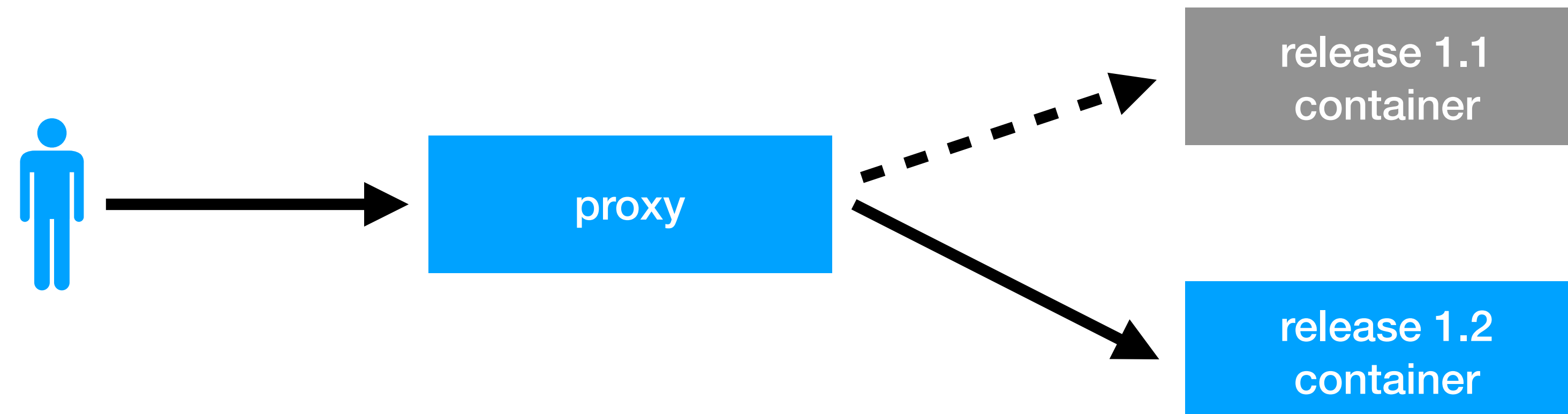
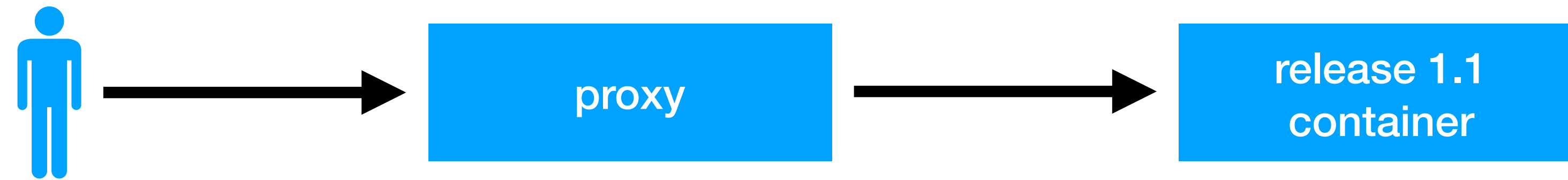
## Docker를 시작해야하는 이유 2



여러 어플리케이션의 독립성과 확장성이 높아짐

- MSA에서 각 태스크들을 컨테이너화 시키면 독립적인 환경을 제공하기 쉬움
- 부하가 많은 태스크들에 컨테이너를 늘려주는 형태로 쉽게 스케일업 가능

## Docker를 시작해야하는 이유 2



무중단 배포 참쉽죠~



### 3. 설치

# 설치하기 전에..

- 도커는 원래 linux 기반임
- 때문에 Oracle VirtualBox라는 가상화 기술 베이스의 도커 툴박스라는 환경 위에 설치했는데
- 최근에는 linuxkit이라는 툴을 이용해서 최소화 된 리눅스 커널을 탑재하여 해당 커널을 사용
- 우리는 linuxkit 기반의 도커를 설치하겠음~~

<https://github.com/linuxkit/linuxkit>

# 설치

## Get Docker

Stable
The Stable version is fully baked and tested, and comes with the latest GA release of Docker.
<a href="#">Get Docker Desktop for Mac (Stable)</a>

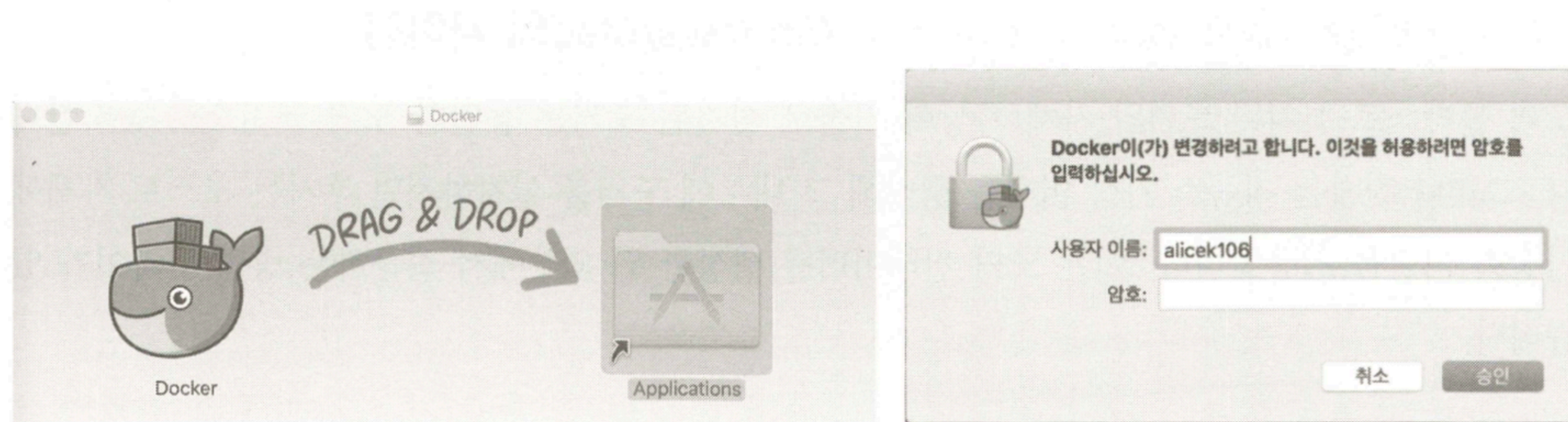
## Installation

### Main system requirements

- Docker Desktop - macOS must be version 10.14 or newer, i.e. Mojave (10.14) or Catalina (10.15).
- Mac hardware must be a 2010 or a newer model.
- View all macOS system requirements [here](#).

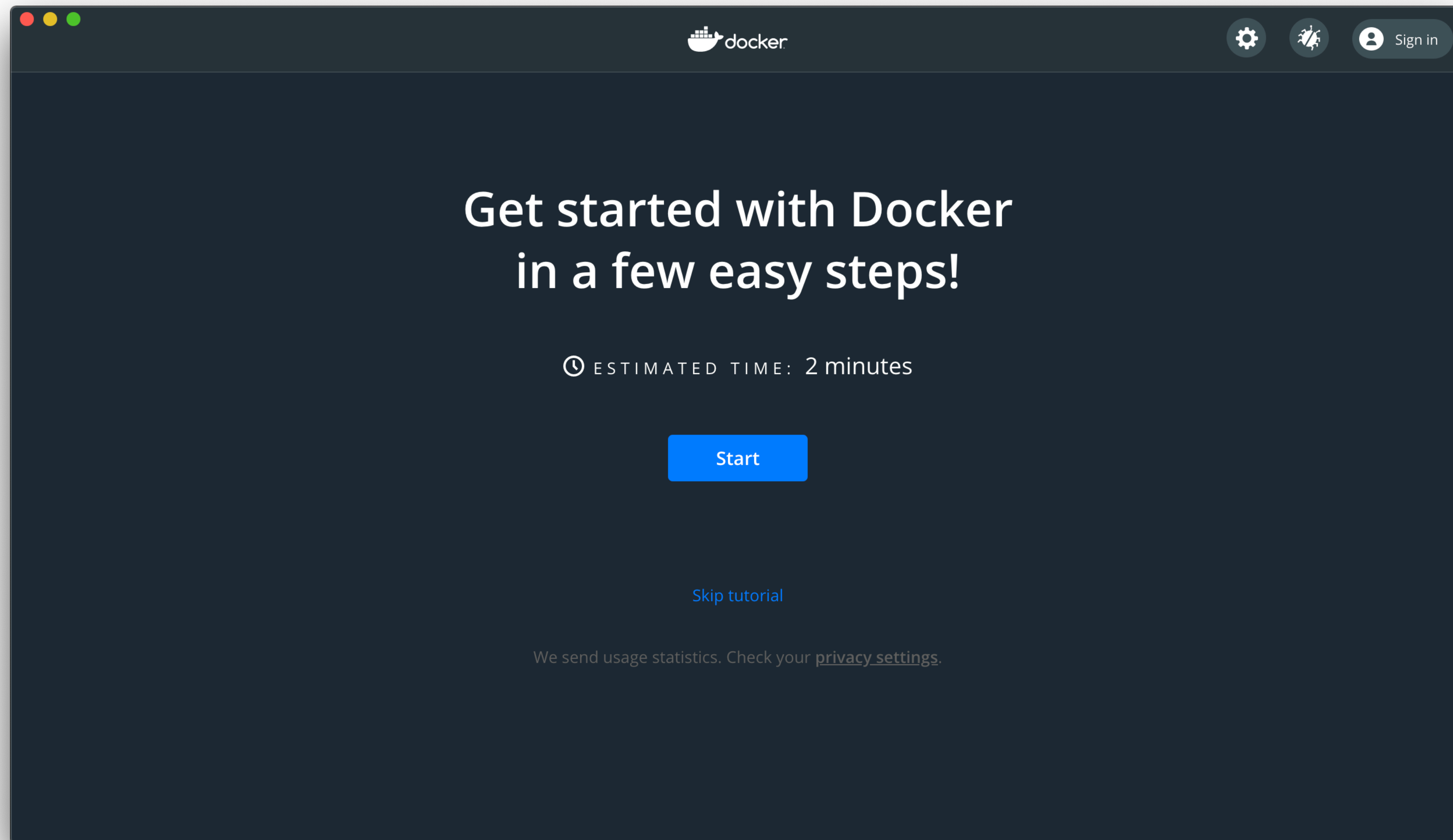
<https://hub.docker.com/editions/community/docker-ce-desktop-mac/>

# 설치



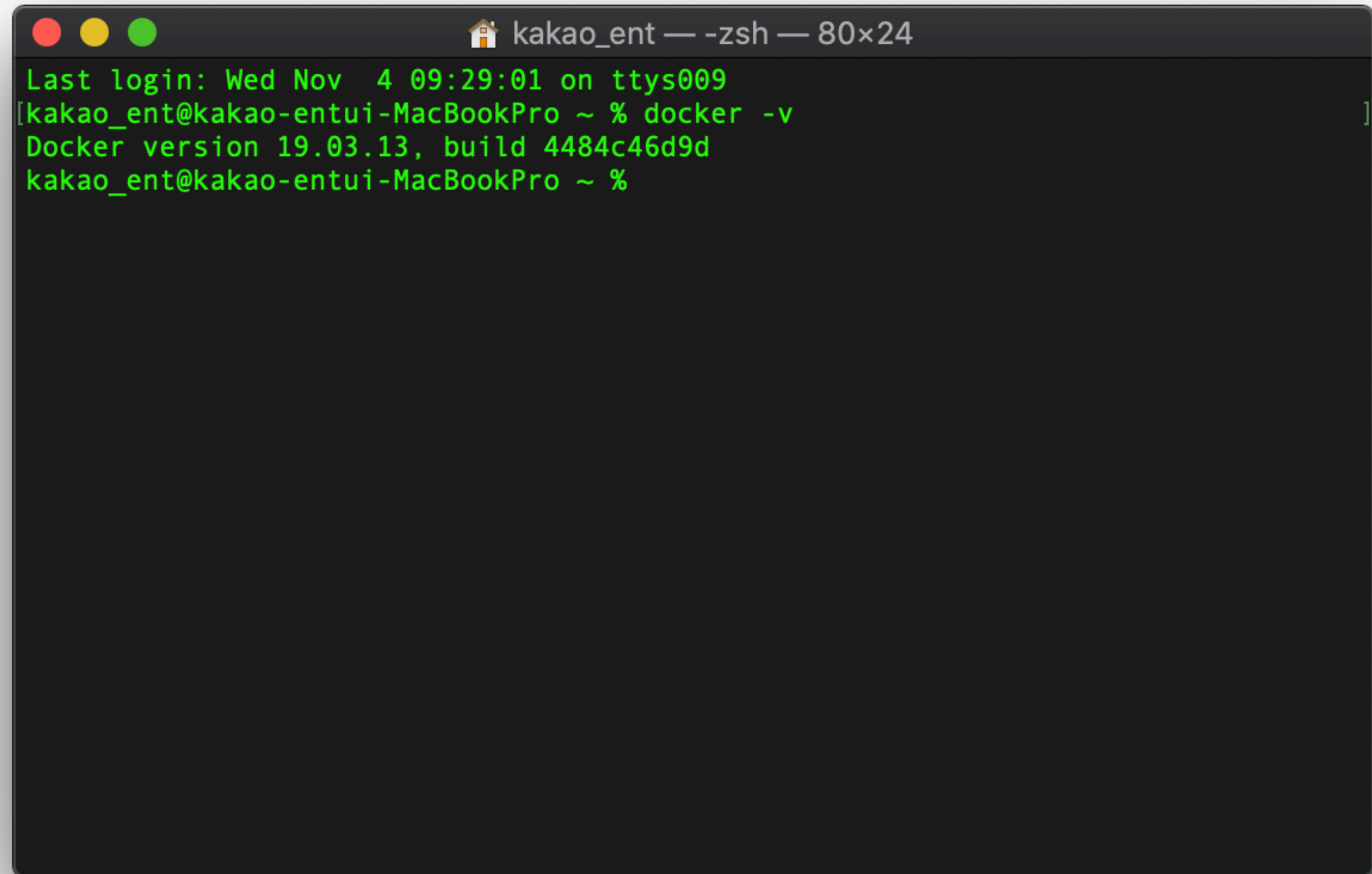


# 설치





# 설치

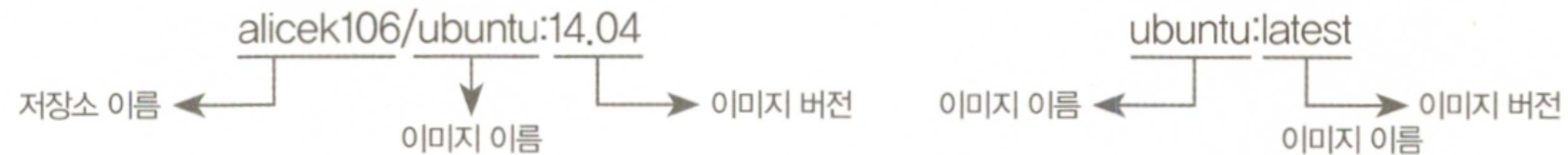


```
kakao_ent — -zsh — 80x24
Last login: Wed Nov  4 09:29:01 on ttys009
[kakao_ent@kakao-entui-MacBookPro ~ % docker -v ]
Docker version 19.03.13, build 4484c46d9d
kakao_ent@kakao-entui-MacBookPro ~ %
```

## 4. 개념

# 도커 이미지

- 컨테이너를 생성할 때 필요한 요소
- 가상 머신을 생성할 때 사용하는 iso 파일과 비슷한 개념
- docker hub에서 github repo처럼 필요한 이미지를 내려받아 사용할 수 있음 (docker pull ubuntu 하면 ubuntu 환경이 뿔!)
- [저장소 이름]/[이미지 이름]:[태그] 형태로 구성



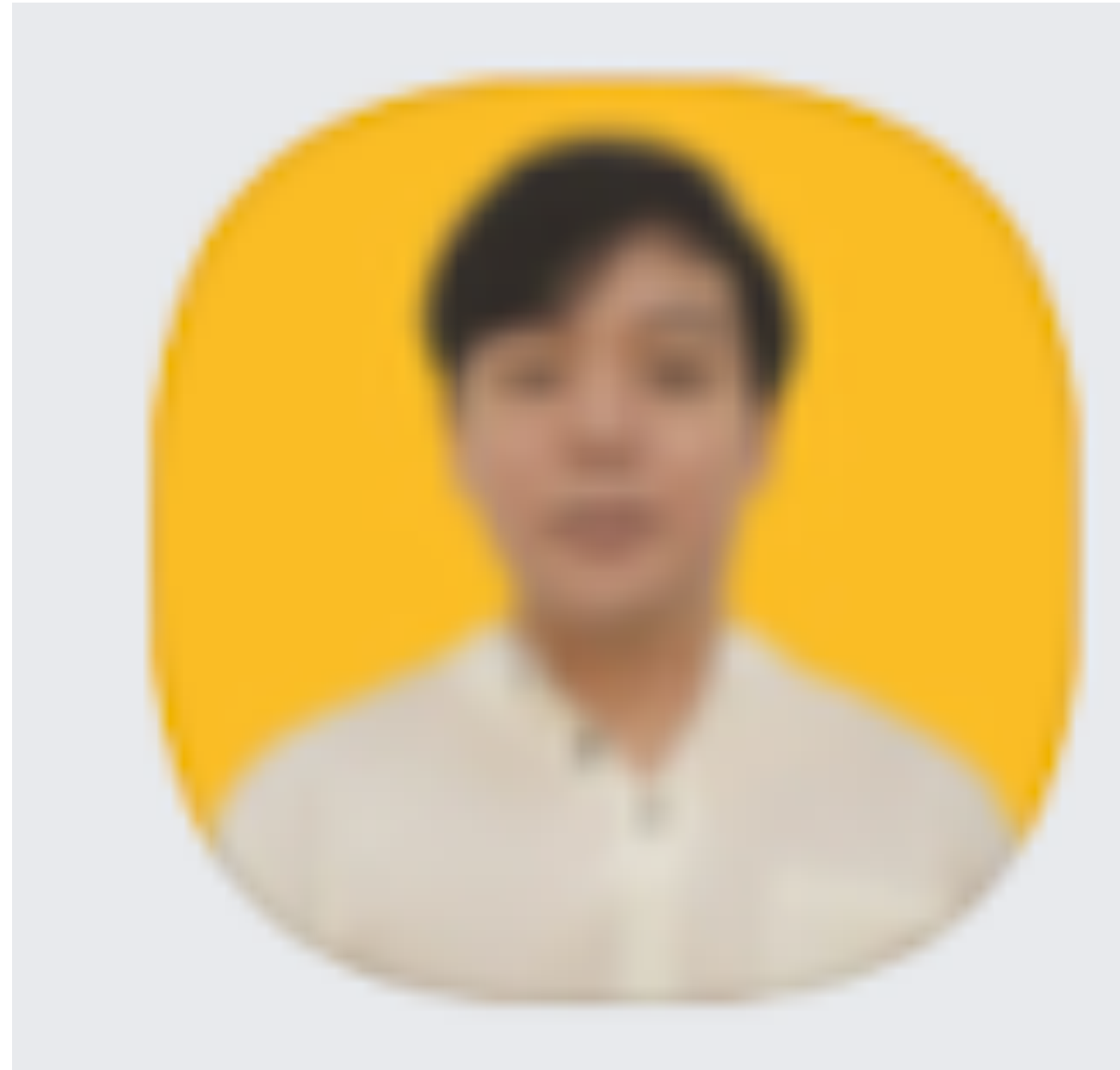
# 도커 컨테이너

- 도커 이미지는 ubuntu, centos 등 운영체제부터, 아파치, nginx, MySQL등 갖가지 어플리케이션이 존재
- 이러한 이미지로 컨테이너를 생성하면 이미지의 목적에 맞는 파일시스템, 격리된 자원/네트워크를 사용할 수 있음
- ubuntu 도커 이미지로 두개의 컨테이너를 생성하고, A에는 MySQL을, B에는 nginx를 설치해도 서로 영향 x
- 이렇게 나만의 컨테이너를 구축해서 이미지로 만들면 고거시 개발/배포~



# What's next?

**docker container, image 실습!!!!  
with kong**



## 5. 부록

# 도커 내부 동작에 관한 개념 이야기

<https://www.plesk.com/blog/business-industry/docker-containers-explained/>