

# BACKEND CHALLENGE

PRE-ONBOARDING

5월. 컨테이너 기반 서버 관리 경험으로 면접 뽐내기

2024.05.08 (수)

# 교육 대상

- 커리어를 시작하는 주니어 개발자

## 챌린지 참가 자격

- 주니어 커리어 시작을 희망하는 누구나 참여 가능합니다.
- 이직을 희망하는 주니어도 누구나 참여 가능합니다.
- 학습 커리큘럼은 사전 미션을 기반으로 구성하였으며, 강의 시간을 통해 함께 역량을 발전시켜갑니다.
- 취업 / 이직을 목표로 하는 분들은 적극적으로 도전해 보세요.

# 들어가기

누군가에게 도커를 설명한다면?

어디에 위치하고 있는가?

- 가상화 기술 중, 컨테이너 가상화 방식

어떻게 사용하는가?

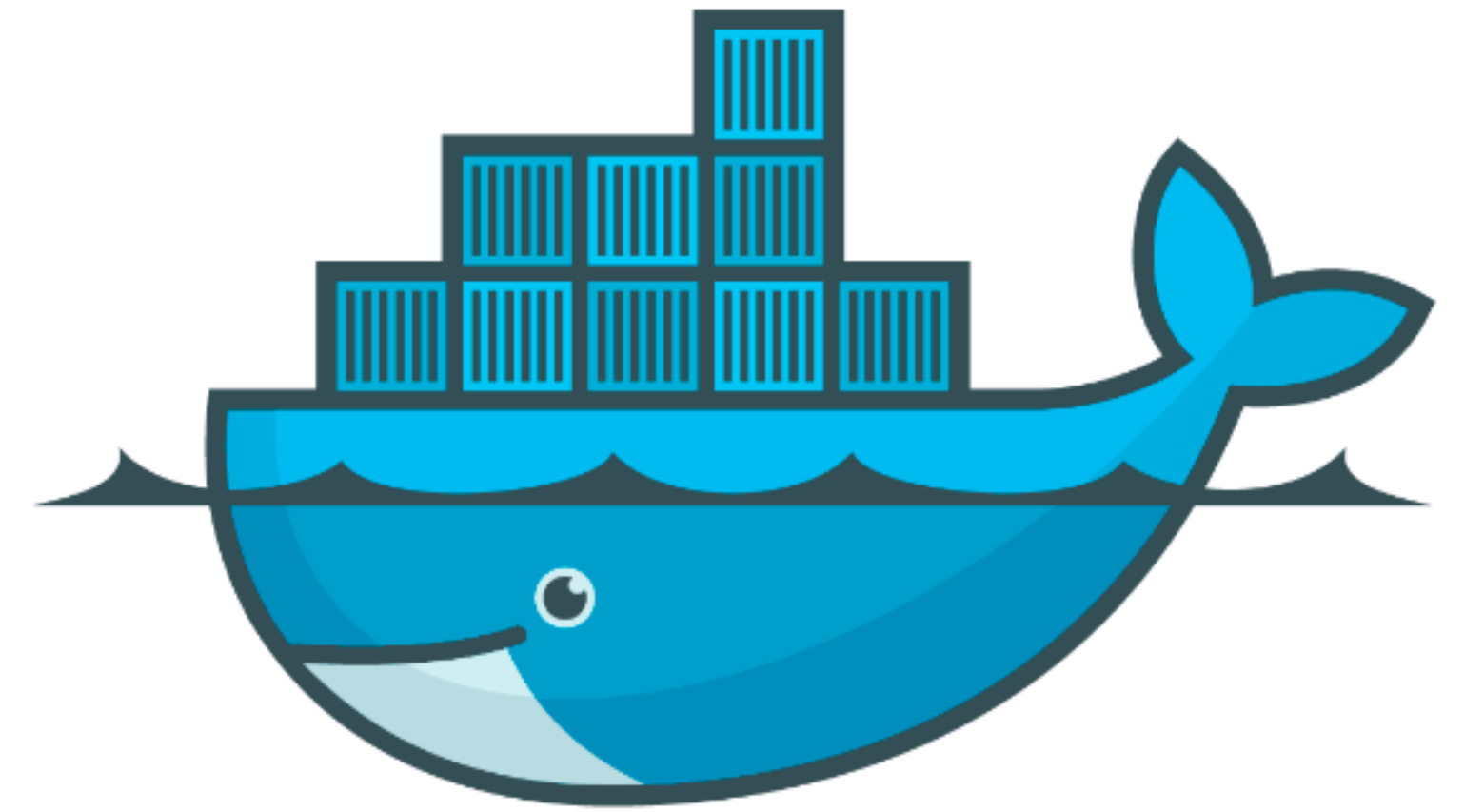
- 도커 설치, 도커 cli, 도커 파일 -> 도커 이미지 -> 도커 컨테이너

왜 만들어 졌는가?

- [강의 중 알아가봅시다. 🙌🙌]

역할은 무엇인가?

- [강의 중 알아가봅시다. 🙌🙌]



# 들어가기

## 누군가에게 도커를 설명한다면?

### 어디에 위치하고 있는가?

- 가상화 기술 중, 컨테이너 가상화 방식

### 어떻게 사용하는가?

- 도커 설치, 도커 cli, 도커 파일 -> 도커 이미지 -> 도커 컨테이너

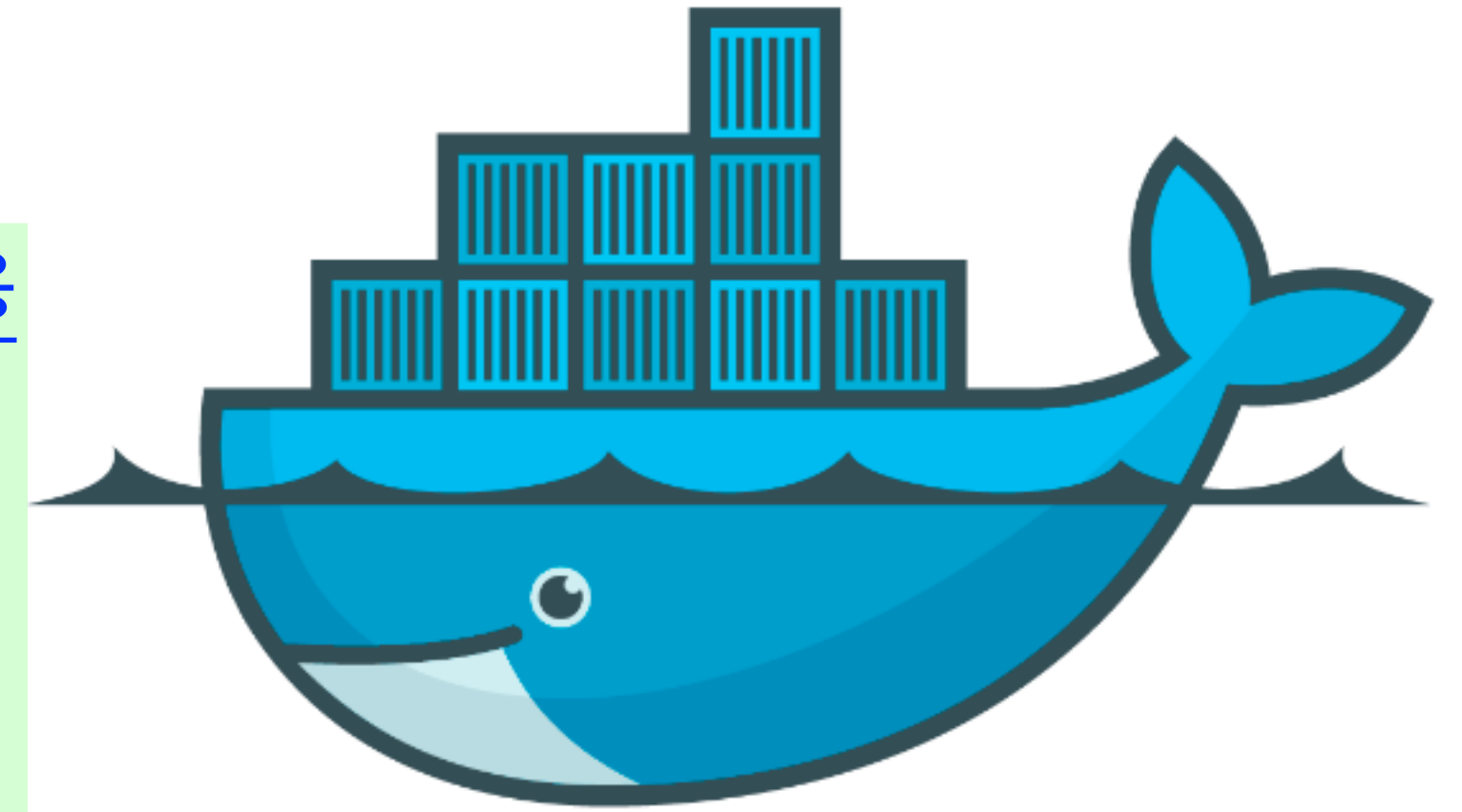
### 왜 만들어 졌는가?

- [강의 중 알아가봅시다. 🙌🙌]

### 역할은 무엇인가?

- [강의 중 알아가봅시다. 🙌🙌]

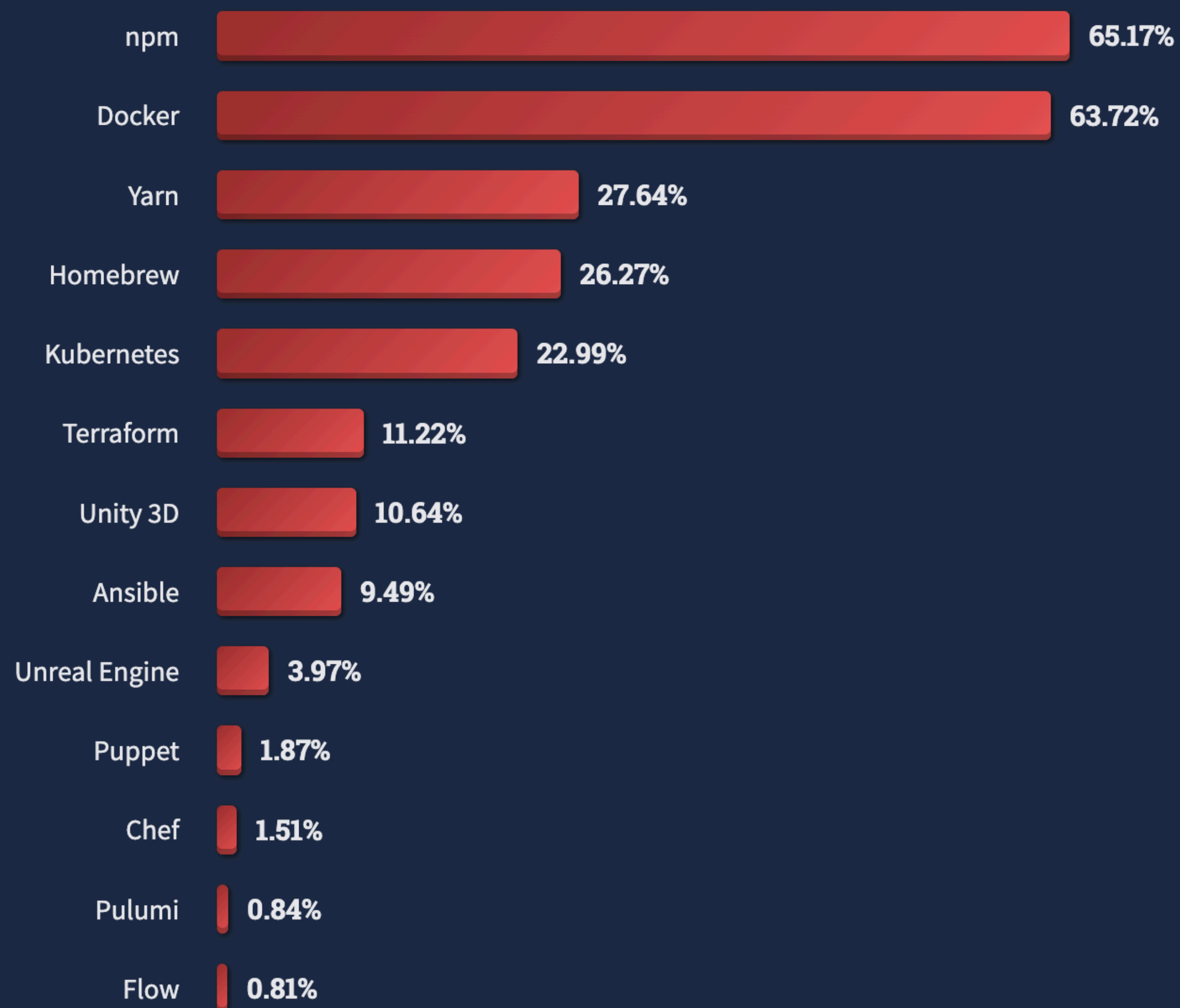
 [사전 미션 내용](#)





# Git 만큼 기본 적인 개발 툴이 된 Docker

stackoverflow 2022 developer survey



## Technology

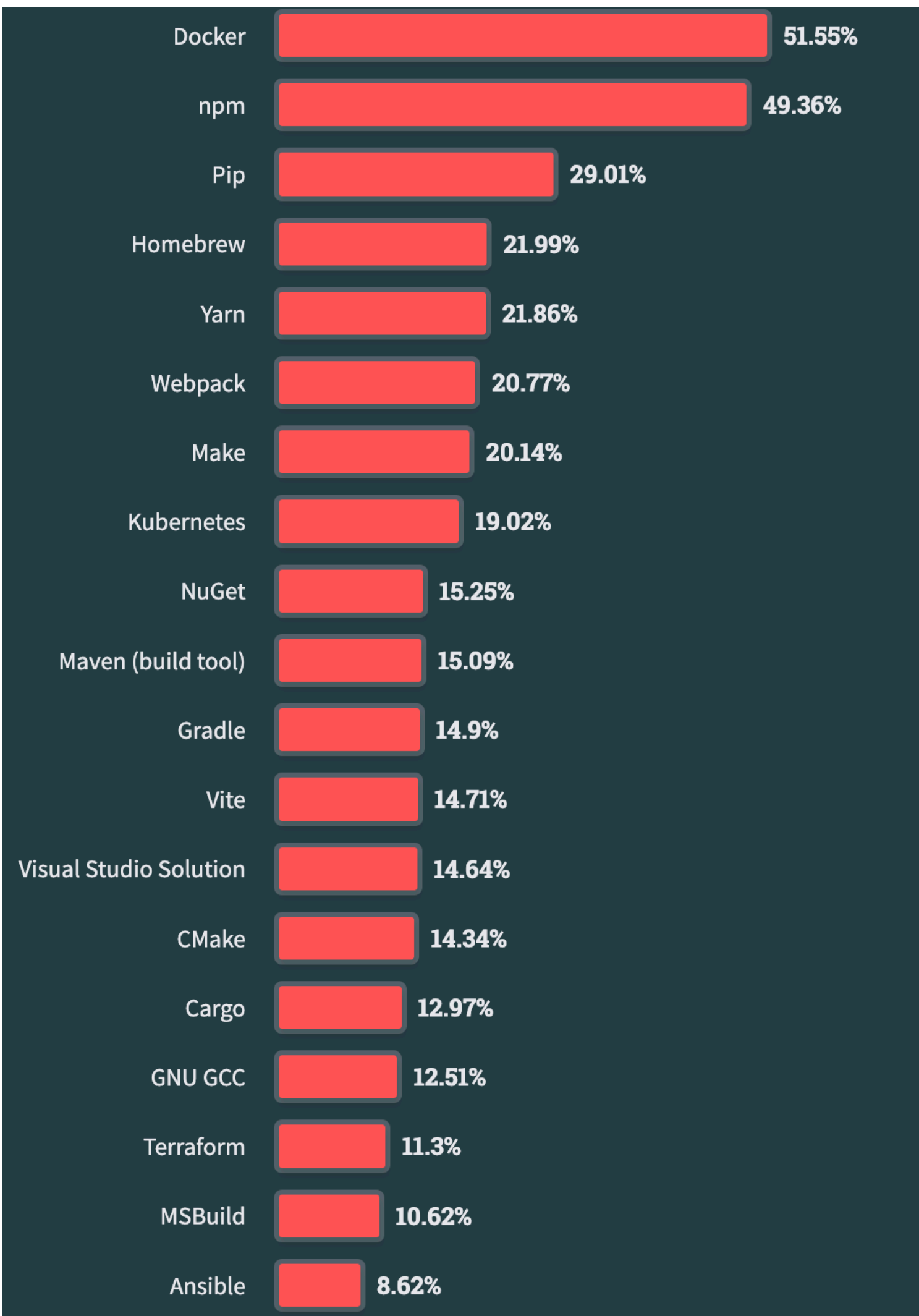
Most popular technologies

Last year we saw Git as a fundamental tool to being a developer. This year it appears that Docker is becoming a similar fundamental tool for Professional Developers, increasing from 55% to 69%.

People learning to code are more likely to be using 3D tools than Professional Developers - Unity 3D (23% vs 8%) and Unreal Engine (9% vs 3%) - teaching themselves skills for 3D VR and AR.

# Git 만큼 사랑 받는 개발 툴이 된 Docker

stackoverflow 2023 developer survey



## Technology Most popular technologies

This year, Docker is the top-used other tool amongst all respondents (53%) rising from its second place spot last year.

People learning to code are more likely to be using npm or Pip than Docker (50% and 37% respectively vs. 26%). Both are used alongside languages that are popular with students (JavaScript and Python respectively).

올해 Docker는 작년 2위에서 상승해 전체 응답자의 53%로 가장 많이 사용되는 도구가 되었습니다.

코딩을 배우는 사람들은 Docker보다 npm 또는 Pip을 더 많이 사용하는 경향이 있는데,

npm은 50%, Pip은 37% 사용률로 나타났습니다.

두 도구는 학생들에게 인기가 있는 언어인 JavaScript와 Python과 함께 사용되고 있습니다.

# 커리큘럼

**[제 1 강]** 컨테이너 기술에 대해서 알아보고, Docker의 기본 개념과 사용법에 대해 알아보자!

**[제 2 강]** 로컬 환경에서 도커를 활용해보자!

**[제 3 강]** 도커를 활용하는 클라우드 서비스에 대해 알아보자!

**[제 4 강]** 도커를 활용하여 나만에 클라우드 백엔드 서버를 만들어 보자!

# 커리큘럼

**[제 1 강]** 컨테이너 기술에 대해서 알아보고, Docker의 기본 개념과 사용법에 대해 알아보자!

## 이론편 I

- 가상화 기술
- 컨테이너 기술

## 이론편 II

- 도커의 개념과 구조

## 실습편

- 도커 설치
- 도커 파일 작성, 도커 이미지를 만들고, 컨테이너를 실행해보기



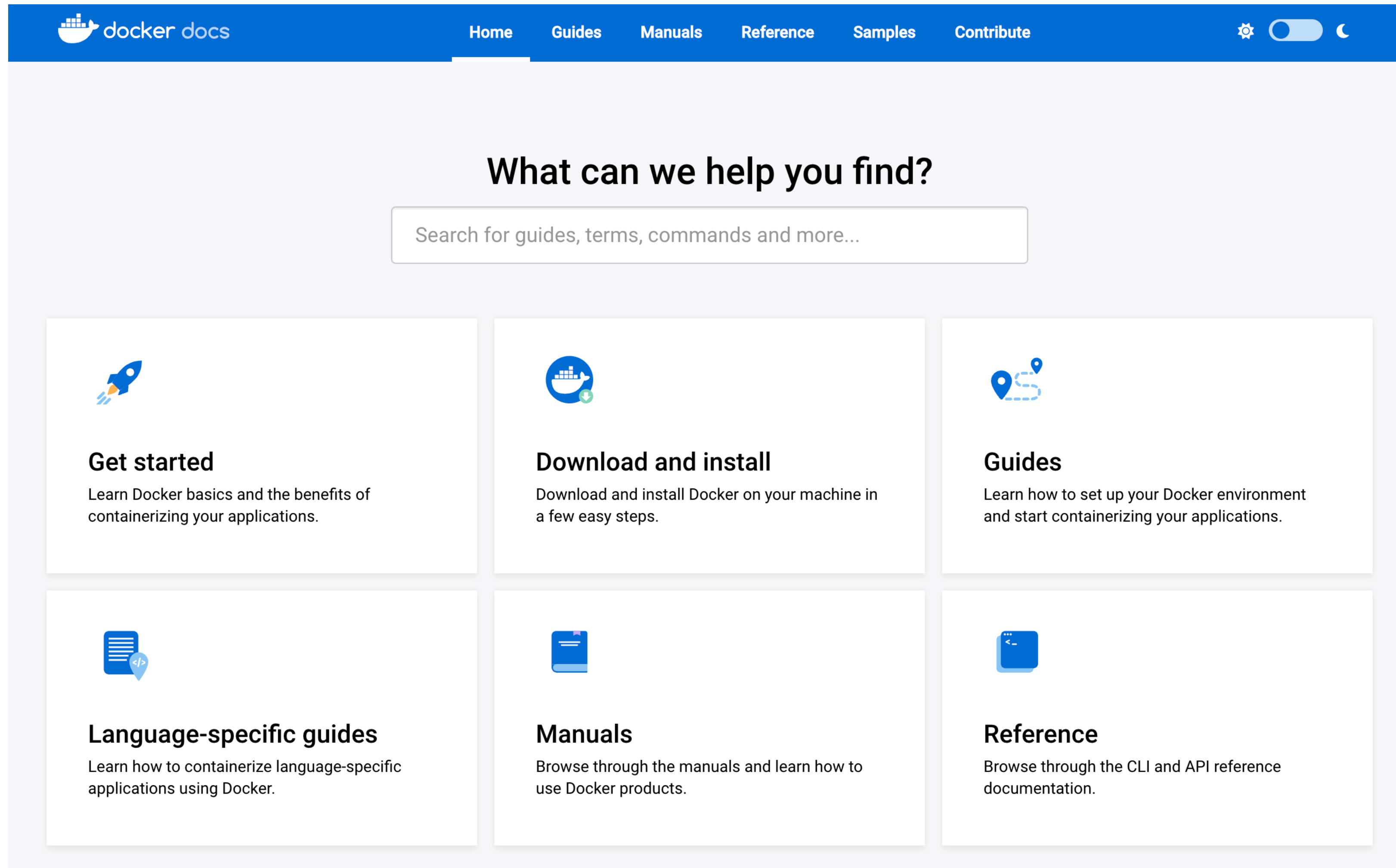
# Docker 란 무엇일까?

컨테이너 기술과 가상화 기술에 대하여...

## 이론편 I

# Docker란 무엇일까?

## 도커 공식 홈페이지 훑어보기



# Docker 란 무엇일까?

## 도커 공식 홈페이지 훑어보기

- open platform 이다.
- 어플리케이션을 인프라에서 분리해준다.
- 신속하다.
- 인프라를 어플리케이션을 관리하는 것 처럼 관리 할 수 있다.
- 코드 배포에 용의하다.

# Docker 란 무엇일까?

- 컨테이너 기반 가상화 도구
- 애플리케이션을 컨테이너라는 단위로 격리하여 실행하고 배포하는 기술

# Container 란 무엇일까?

- 컨테이너는 가상화 기술 중 하나
- 호스트 운영체제 위에 여러 개의 격리된 환경을 생성
- 각각의 컨테이너 안에서 애플리케이션을 실행



# 가상화 (Virtualization) 기술이란 무엇일까?

- 하드웨어 리소스(프로세서, 메모리, 저장소 등)를 추상화 하는 것
- 가상화의 예
  - 메모리 가상화
  - 하드웨어 가상화(Hypervisor 기반)
  - 컨테이너 가상화
  - 네트워크 가상화

# 가상 머신이란 무엇일까?

하나의 물리적인 컴퓨터 자원(CPU, 메모리, 저장장치 등)을

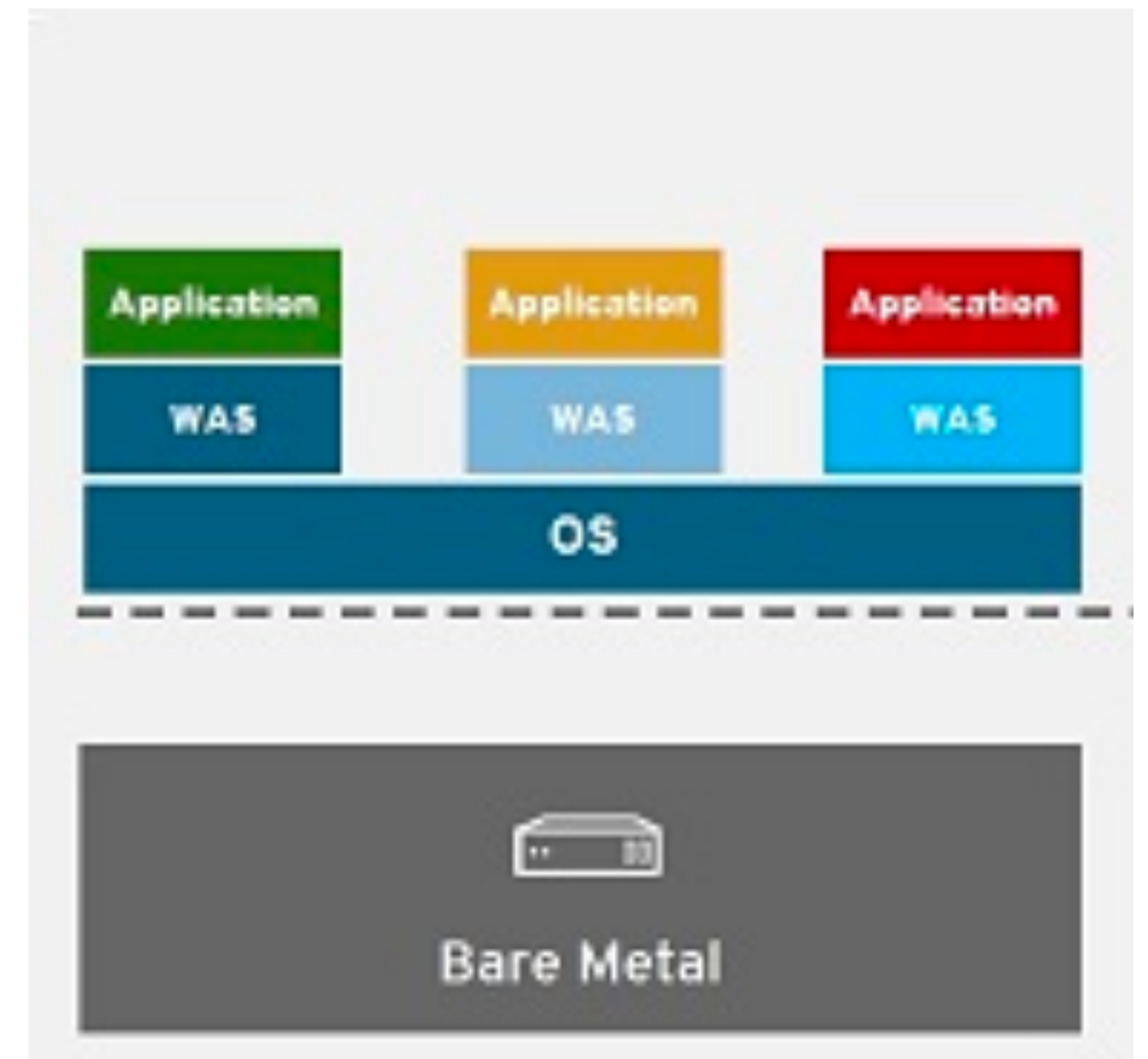
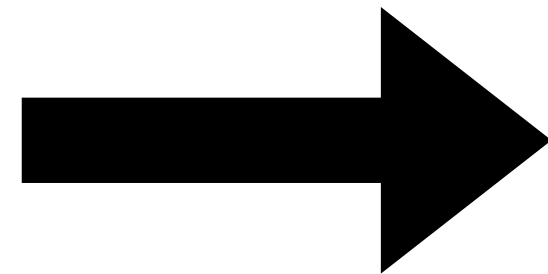
가상적으로 분할하여 여러 개의 가상 컴퓨터 환경을 만들어 내는 기술

이를 통해 물리적인 컴퓨터 자원을 더욱 효율적으로 사용할 수 있으며,

서버나 애플리케이션 등을 운영하는데 있어 유연성과 안정성을 제공합니다.

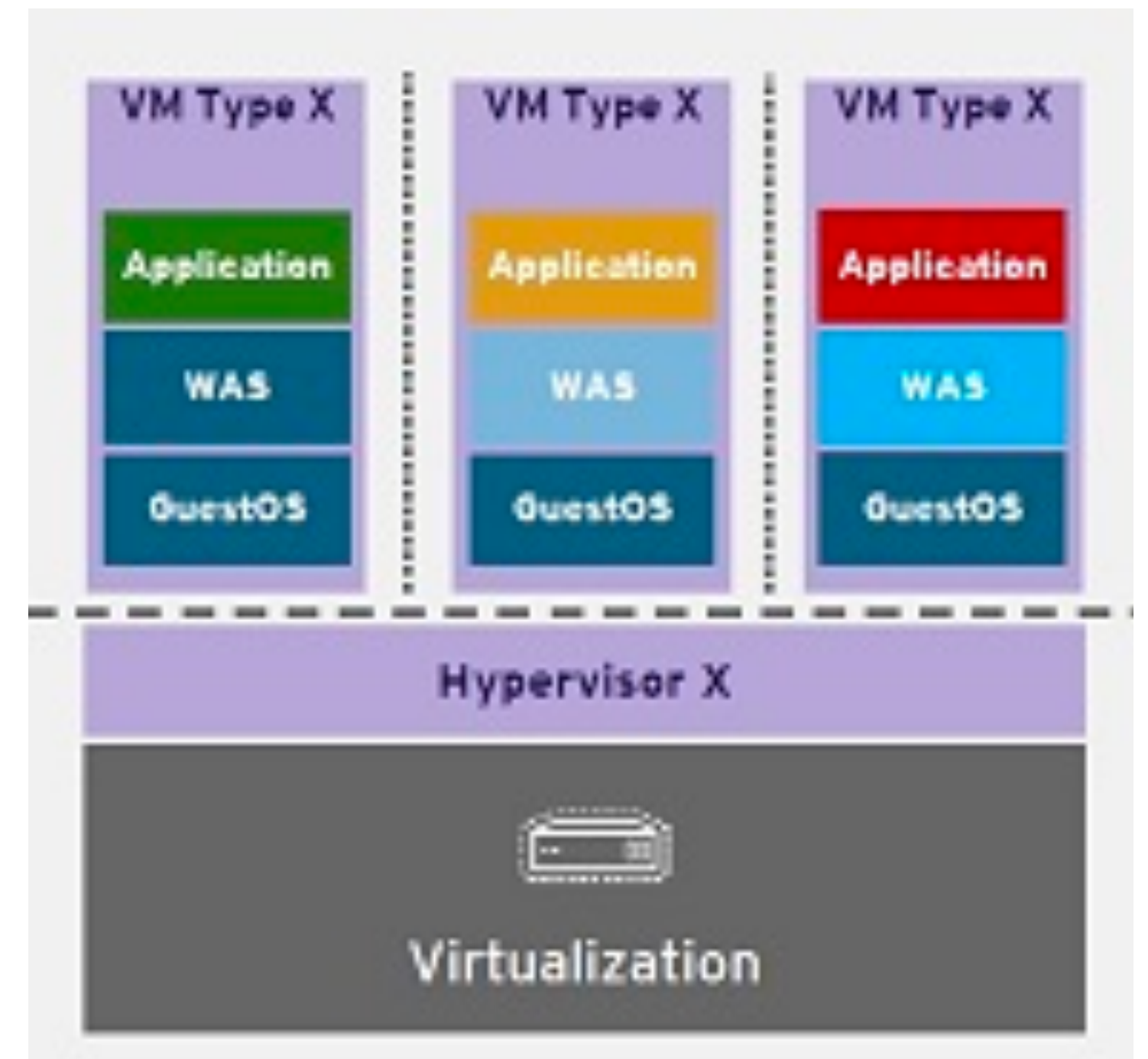
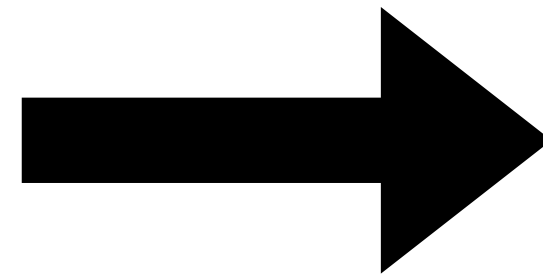
# 가상 머신이란 무엇일까?

고성능 서버를 조금 더 안정적이고 효율적으로 사용할 수 있을까?



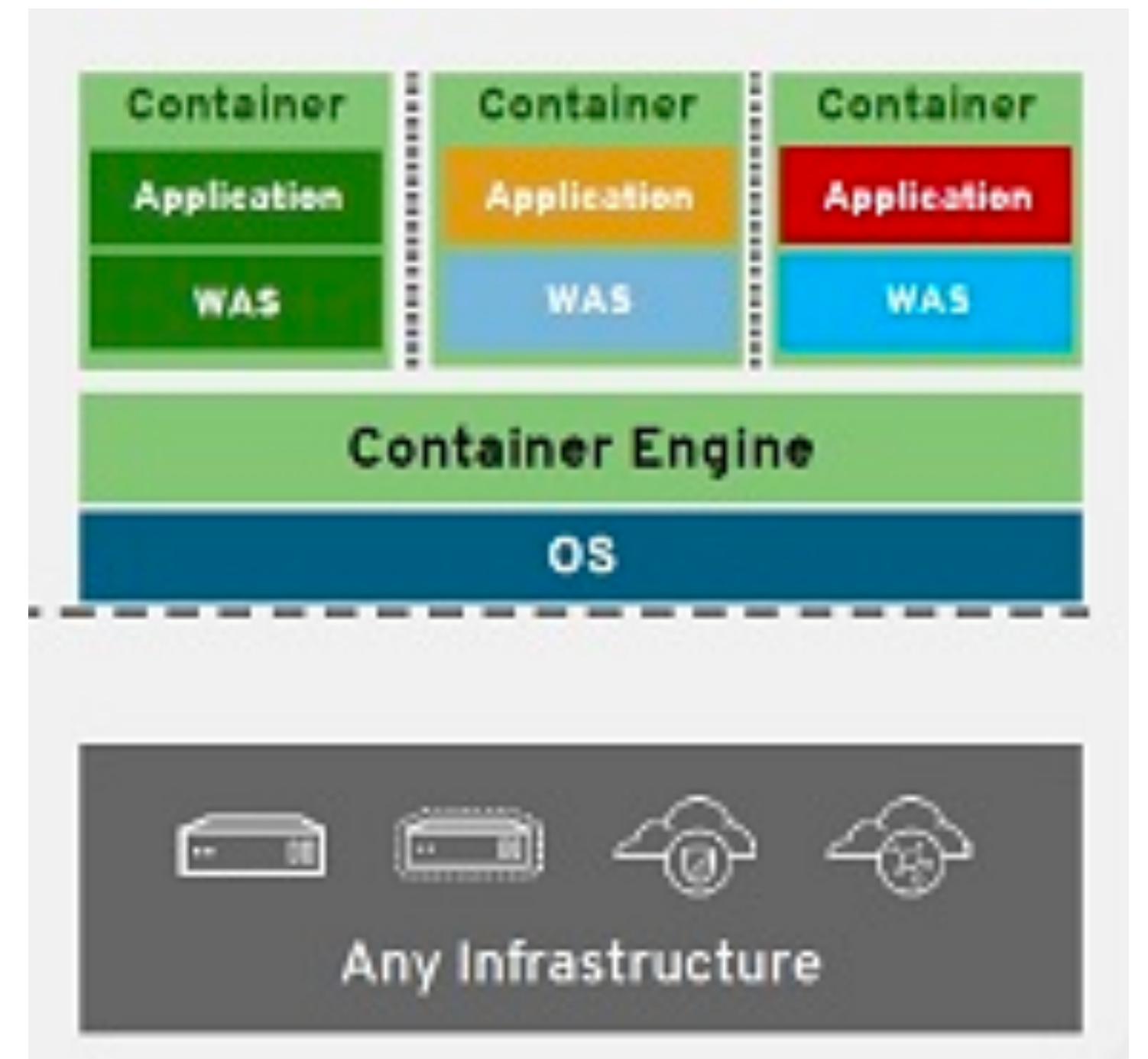
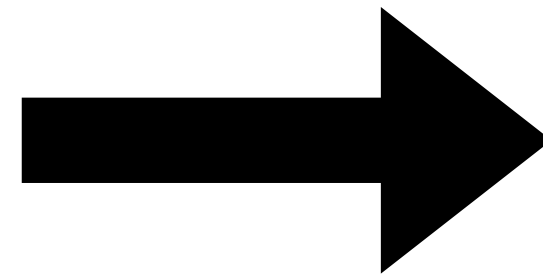
# 가상 머신이란 무엇일까?

고성능 서버를 조금 더 안정적이고 효율적으로 사용할 수 있을까?



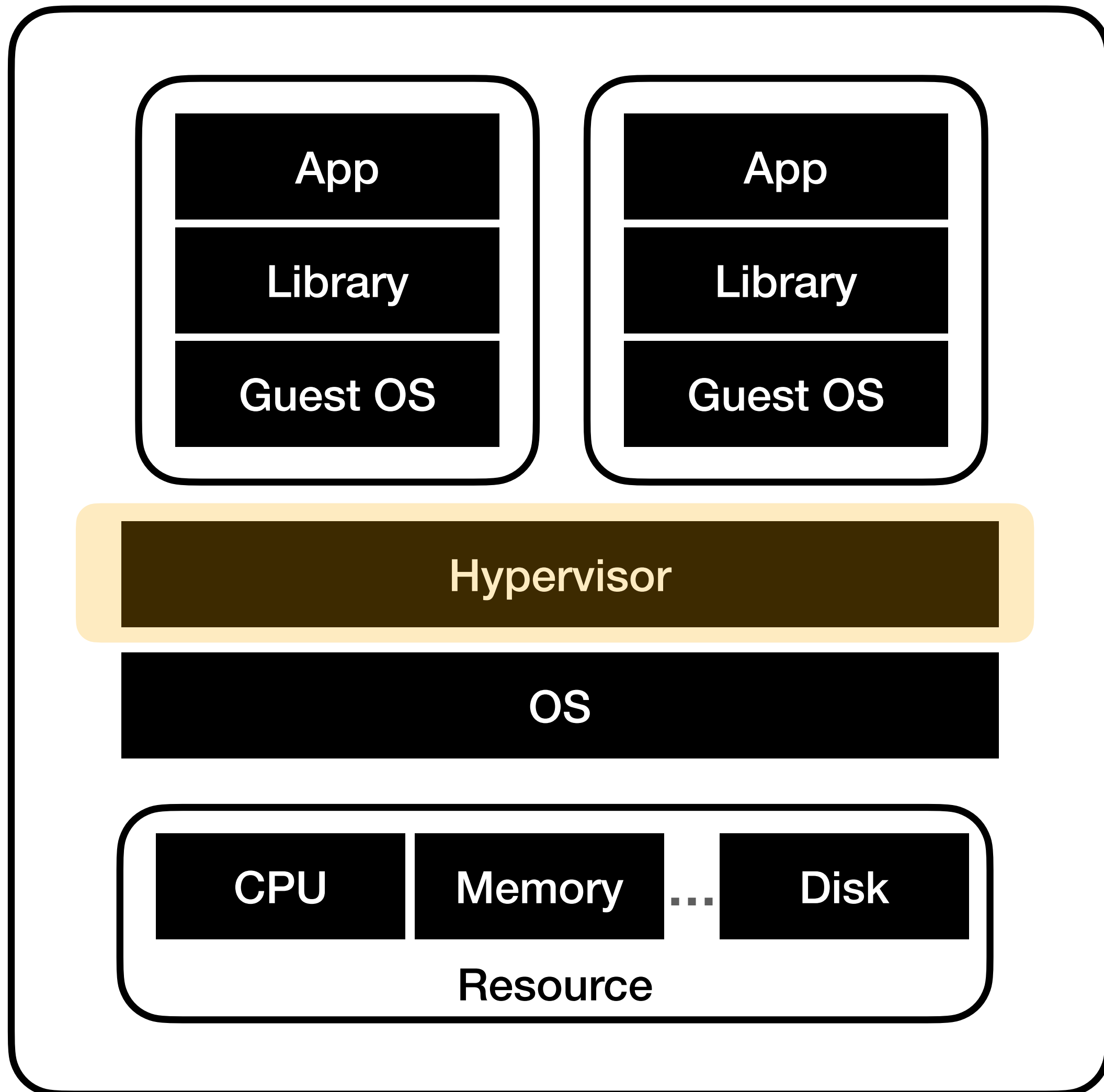
# 가상 머신이란 무엇일까?

고성능 서버를 조금 더 안정적이고 효율적으로 사용할 수 있을까?





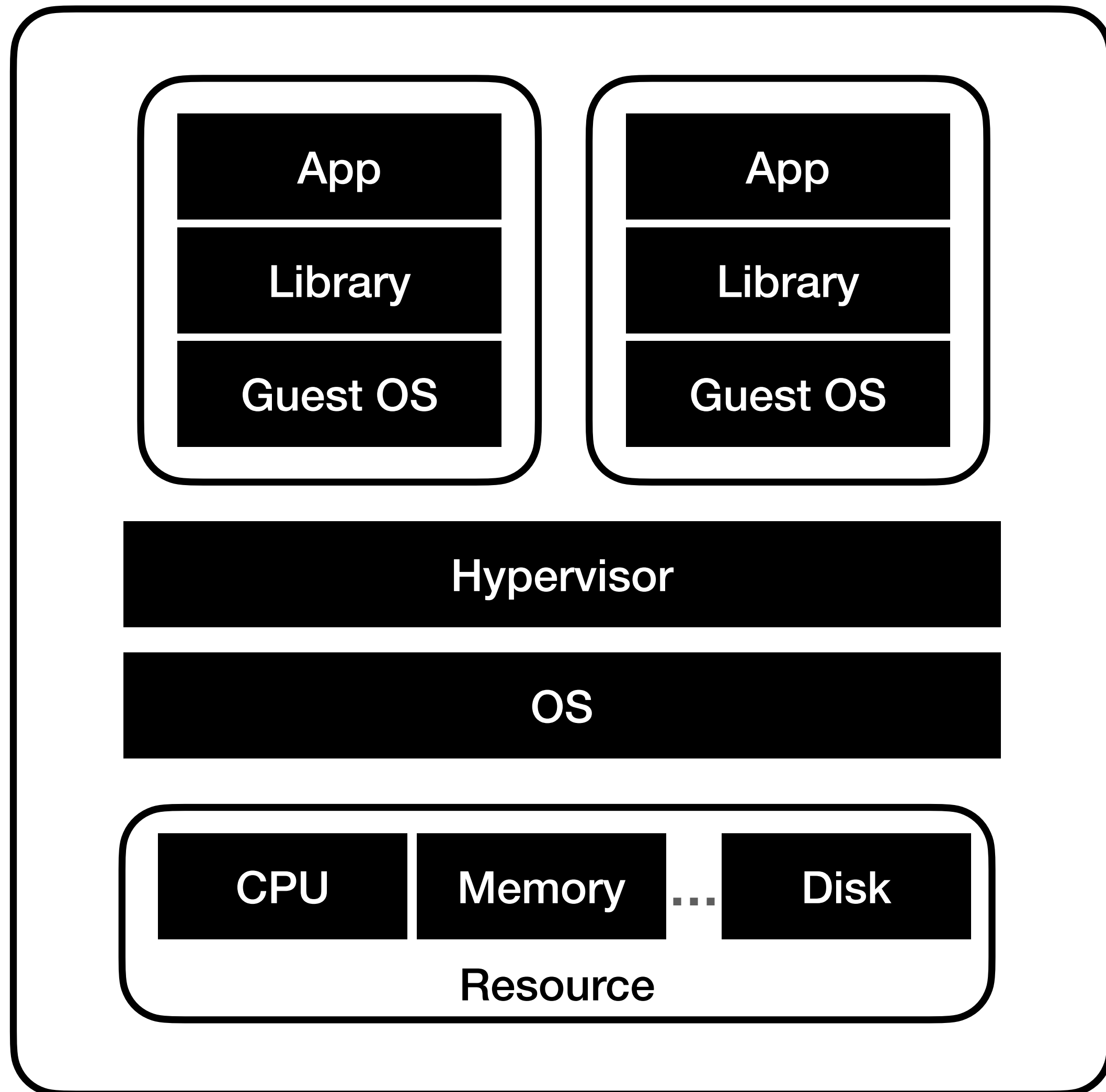
# 가상 머신이란 무엇일까?



## 하이퍼바이저 (Hypervisor) 란?

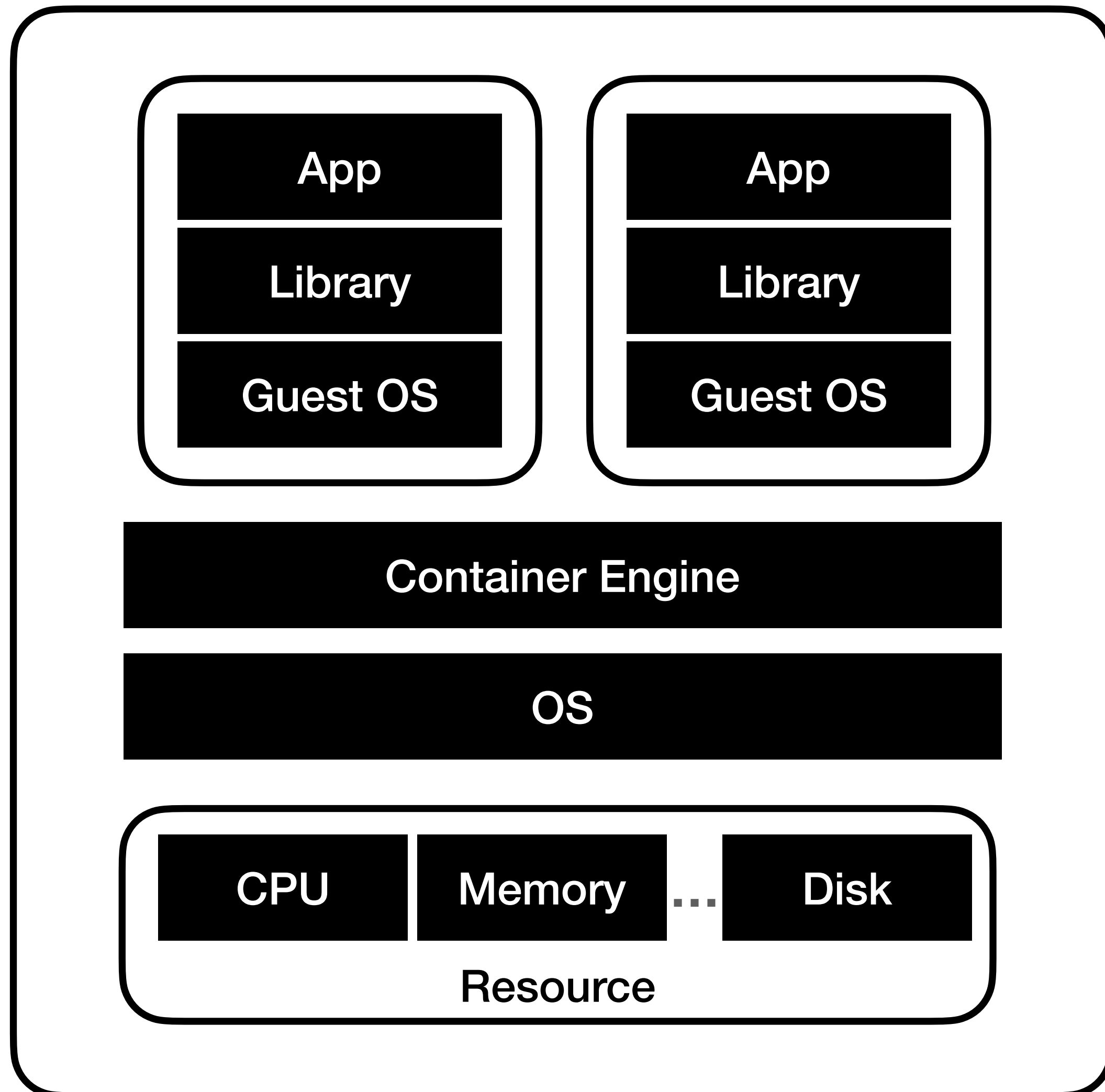
- 가상 머신(Virtual Machine, VM)을 생성하고 구동하는 소프트웨어
- OS에 자원을 할당 및 조율
- OS들의 요청을 번역하여 하드웨어에 전달

# 가상 머신이란 무엇일까?

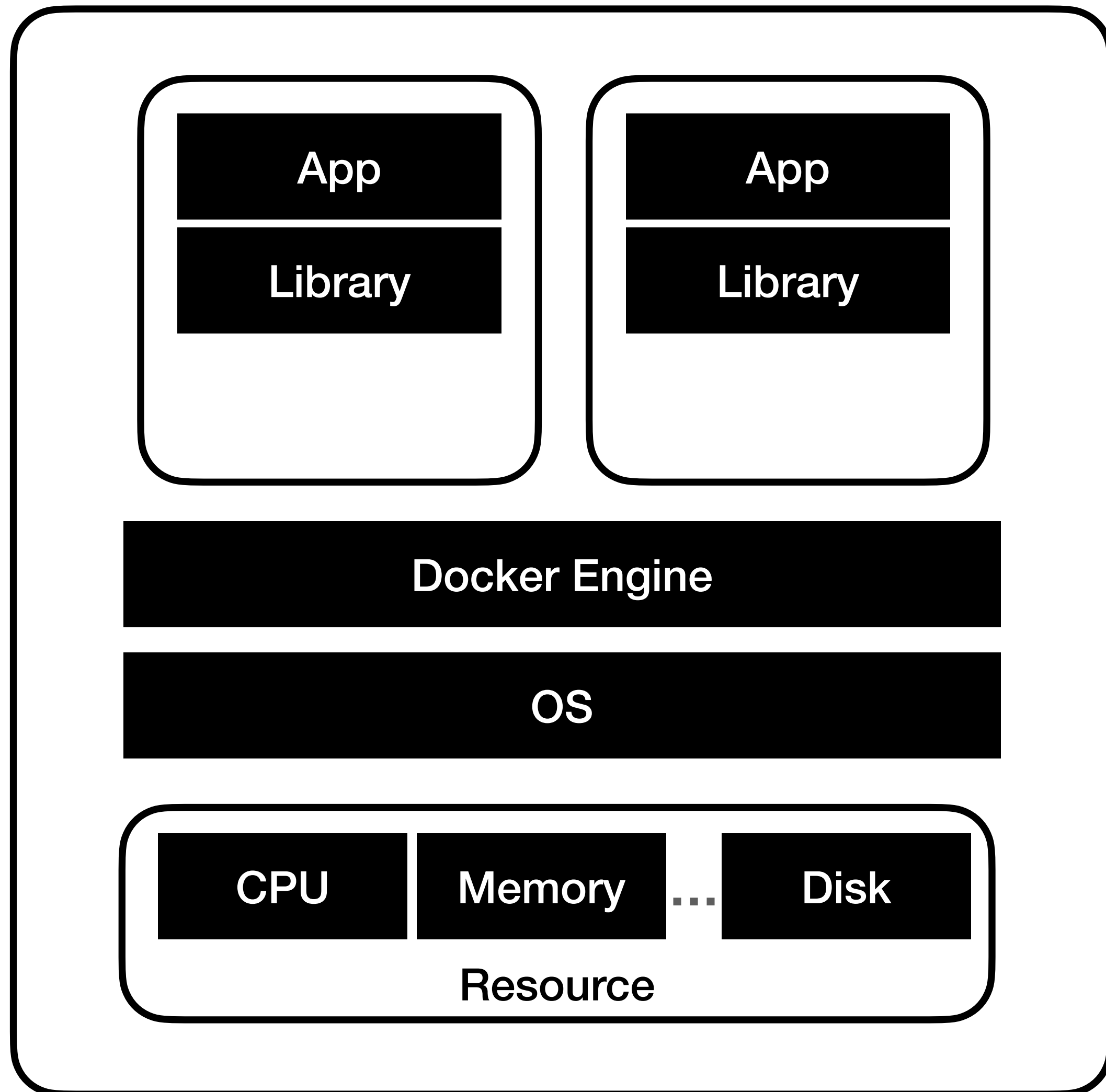


vmware®

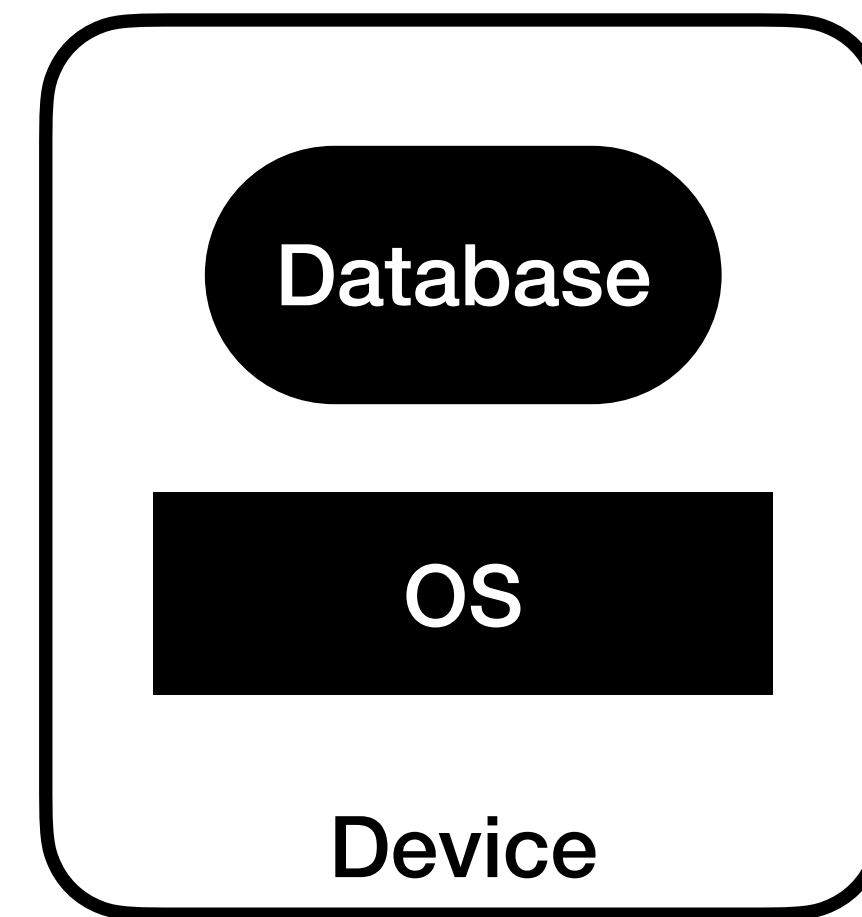
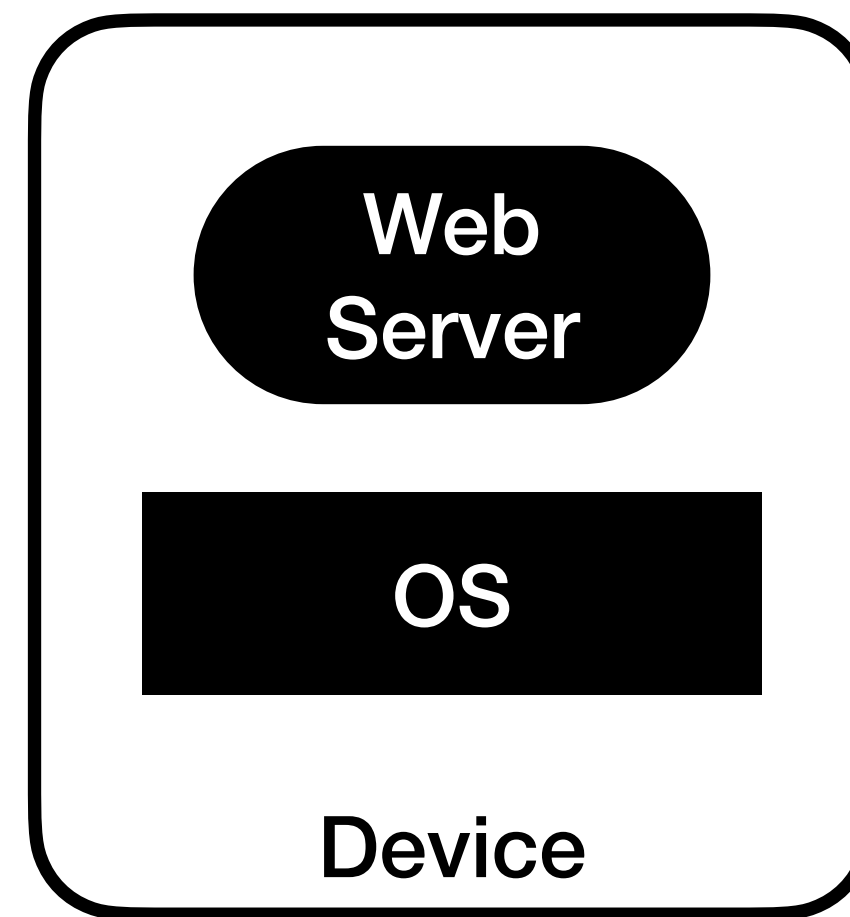
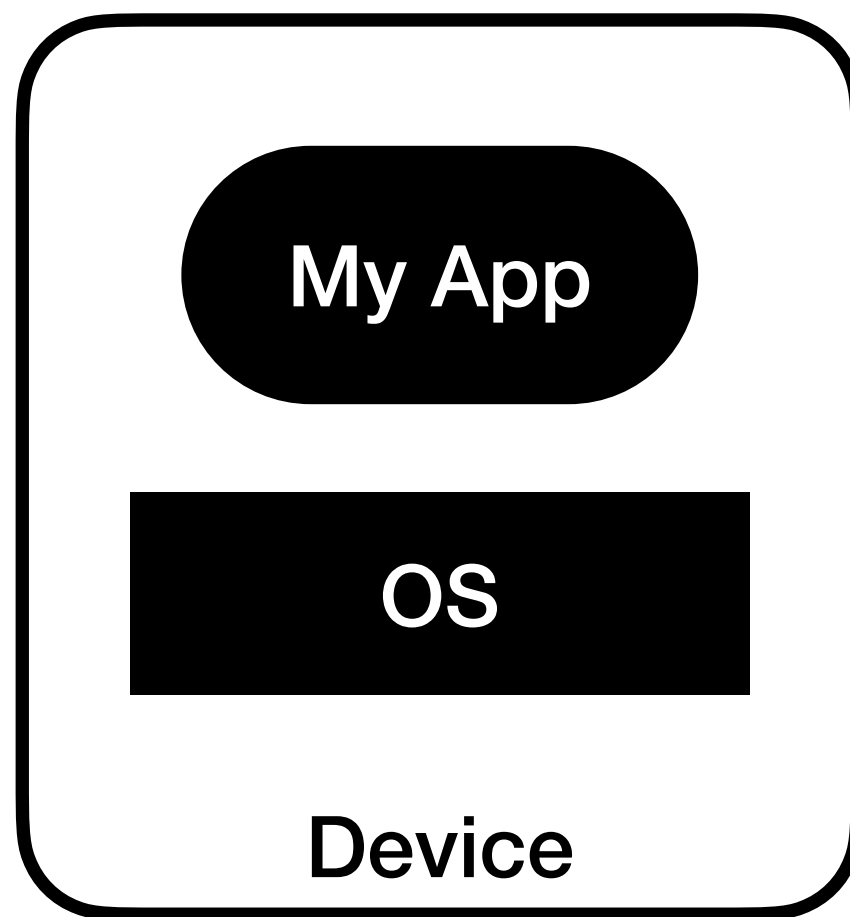
# Container란 무엇인가?



# Container란 무엇인가?

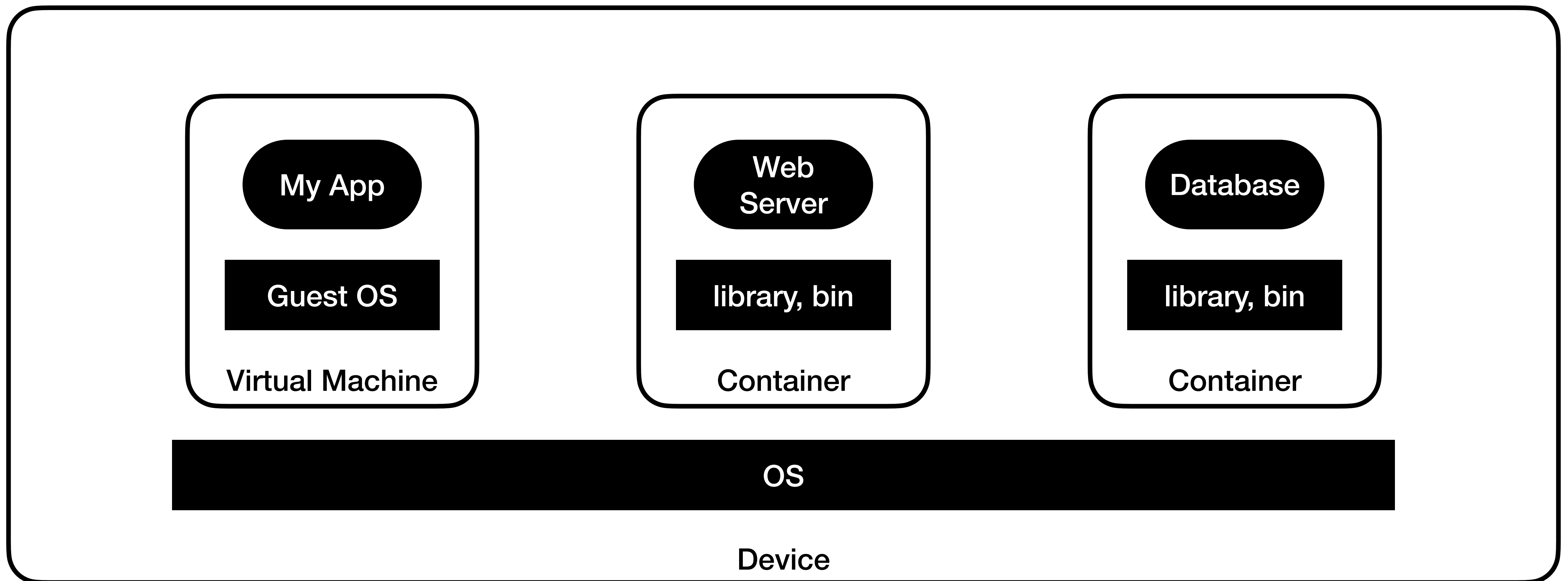


# 활용 예제

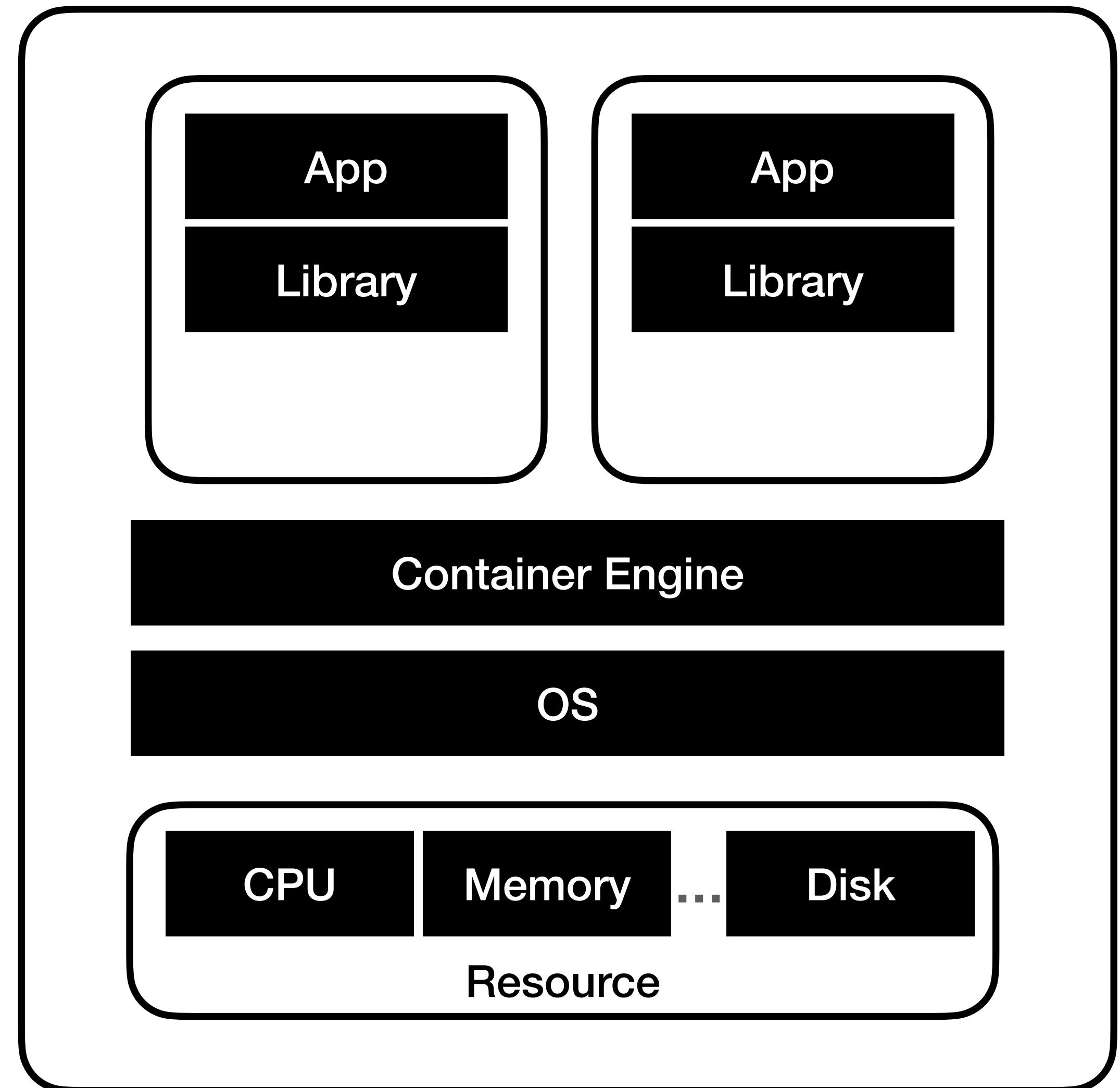
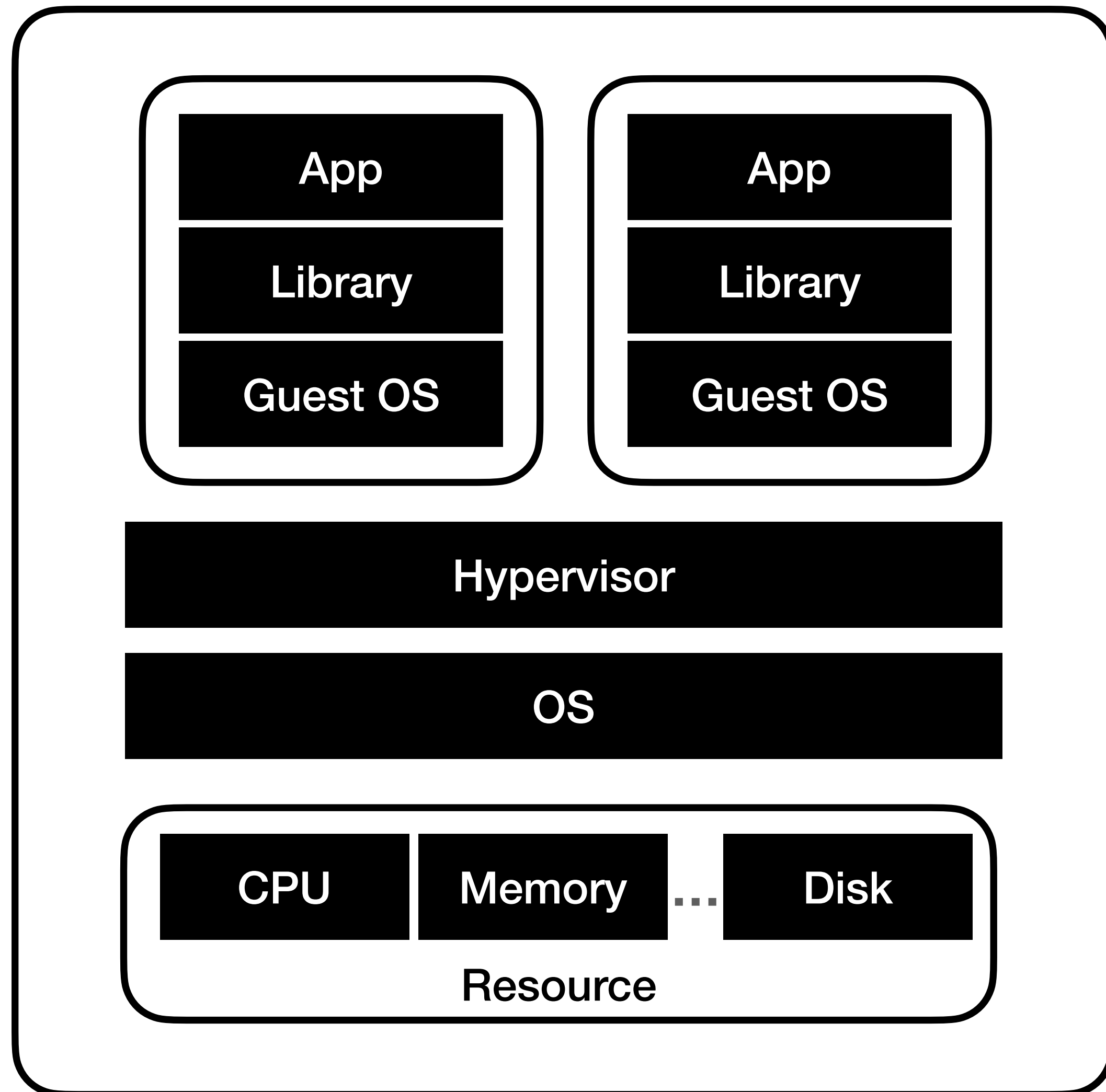




# 활용 예제



# Virtual Machine VS Container



# 컨테이너 기반 특징

- 리눅스 커널의 기능을 사용하여 만들어짐
  - chroot: 파일 시스템을 격리
  - namespace: 프로세스 격리
  - cgroup: 하드웨어 자원 격리
- 프로세스 단위의 격리 환경

🤔 질문: 도커는 리눅스에서만 사용해야하나요?

# Docker 란 무엇일까?

- 컨테이너 기반 가상화 도구
  - 리눅스 컨테이너 기술인 LXC(Linux Containers) 기반
- 애플리케이션을 컨테이너라는 단위로 격리하여 실행하고 배포하는 기술
- 다양한 운영체제에서 사용할 수 있으며,  
컨테이너화된 애플리케이션을 손쉽게 빌드, 배포, 관리할 수 있는 다양한 기능을 제공
- 위 기능들을 통해 애플리케이션을 빠르게 개발하고, 효율적으로 배포, 관리할 수 있음

# 가상화 (Virtualization) 기술이란 무엇일까?

도커 관련 밈





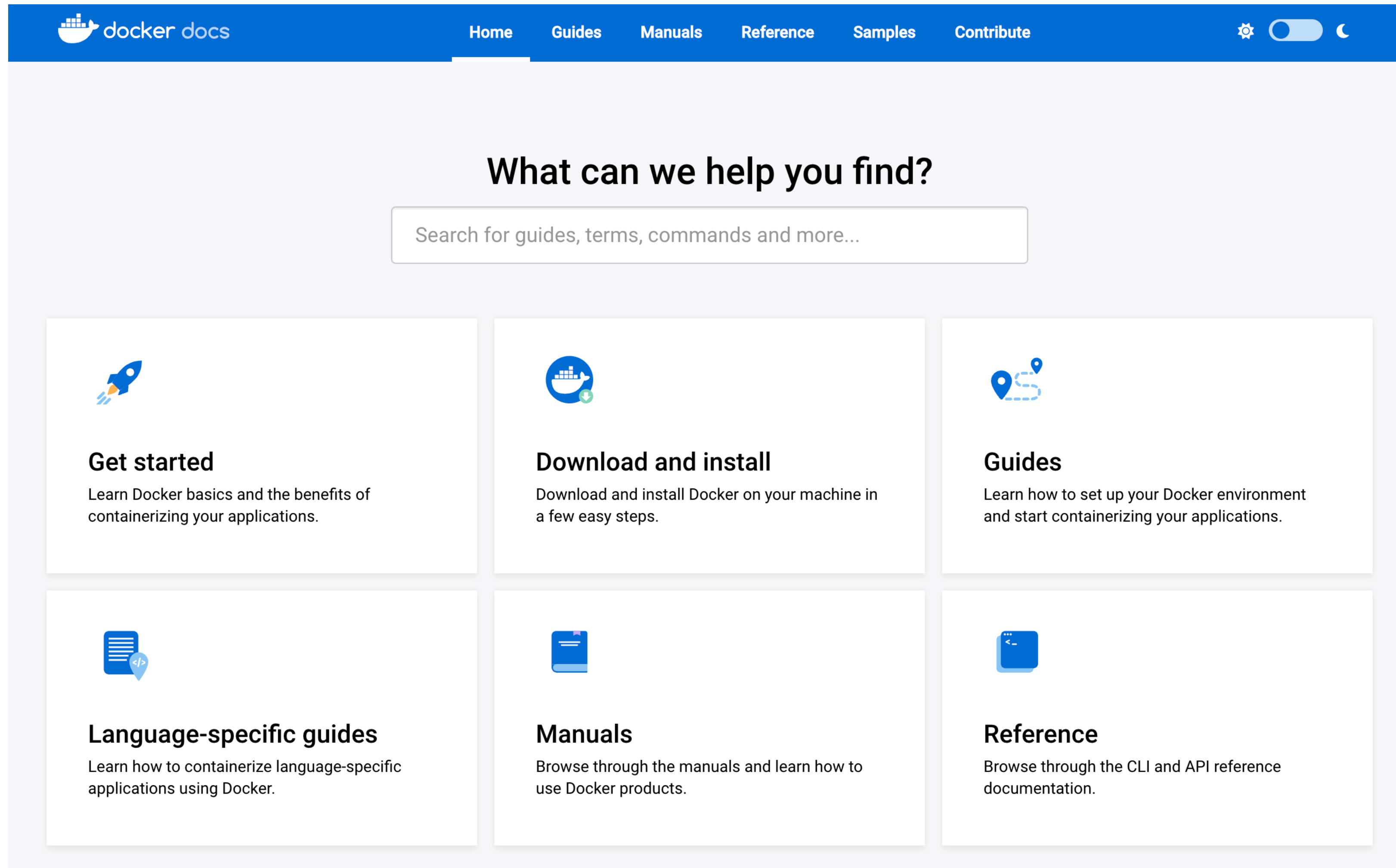
# Docker란 무엇일까?

Docker의 내부 구조에 대하여...

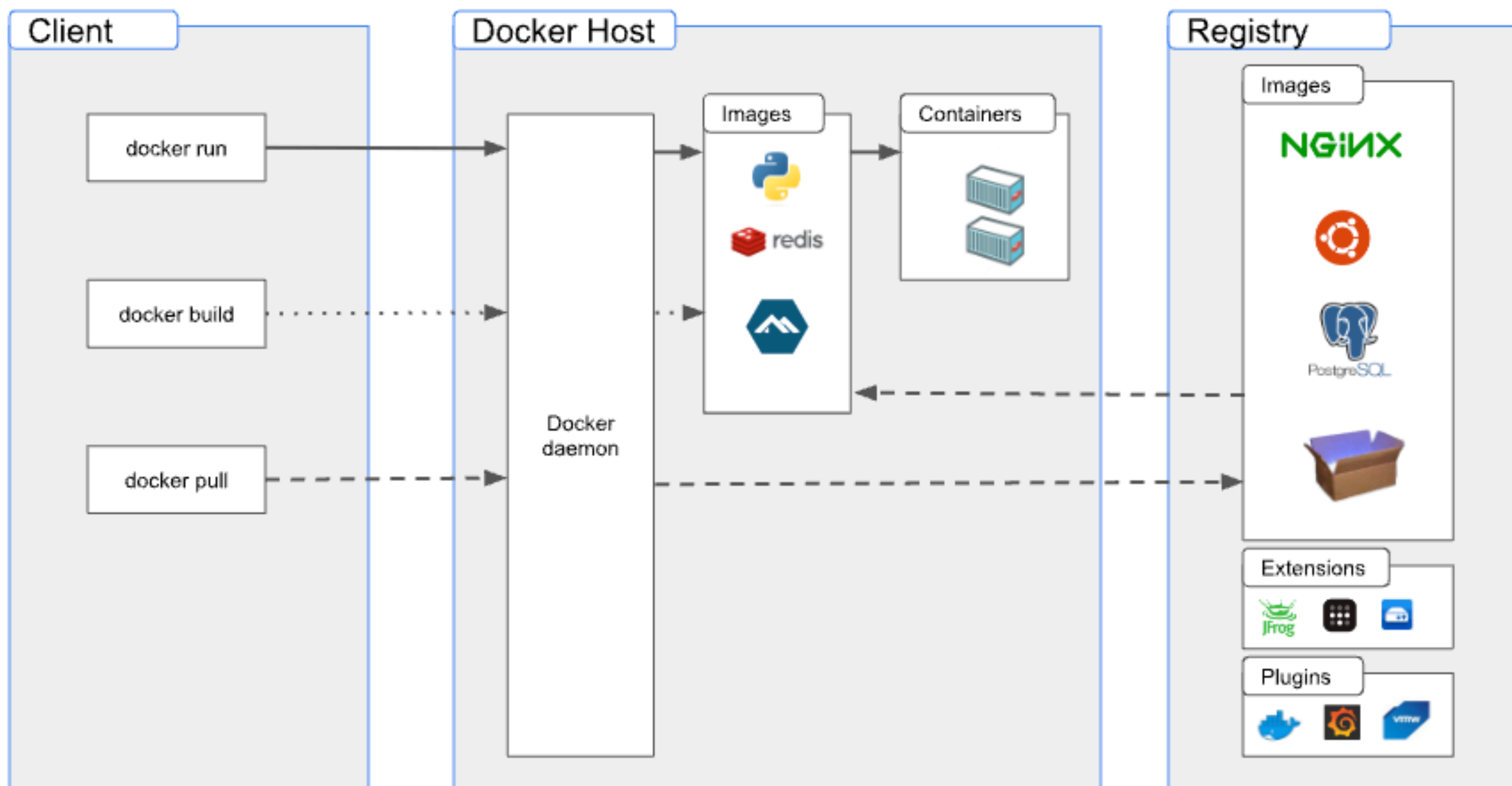
## 이론편 II

# Docker 란 무엇일까?

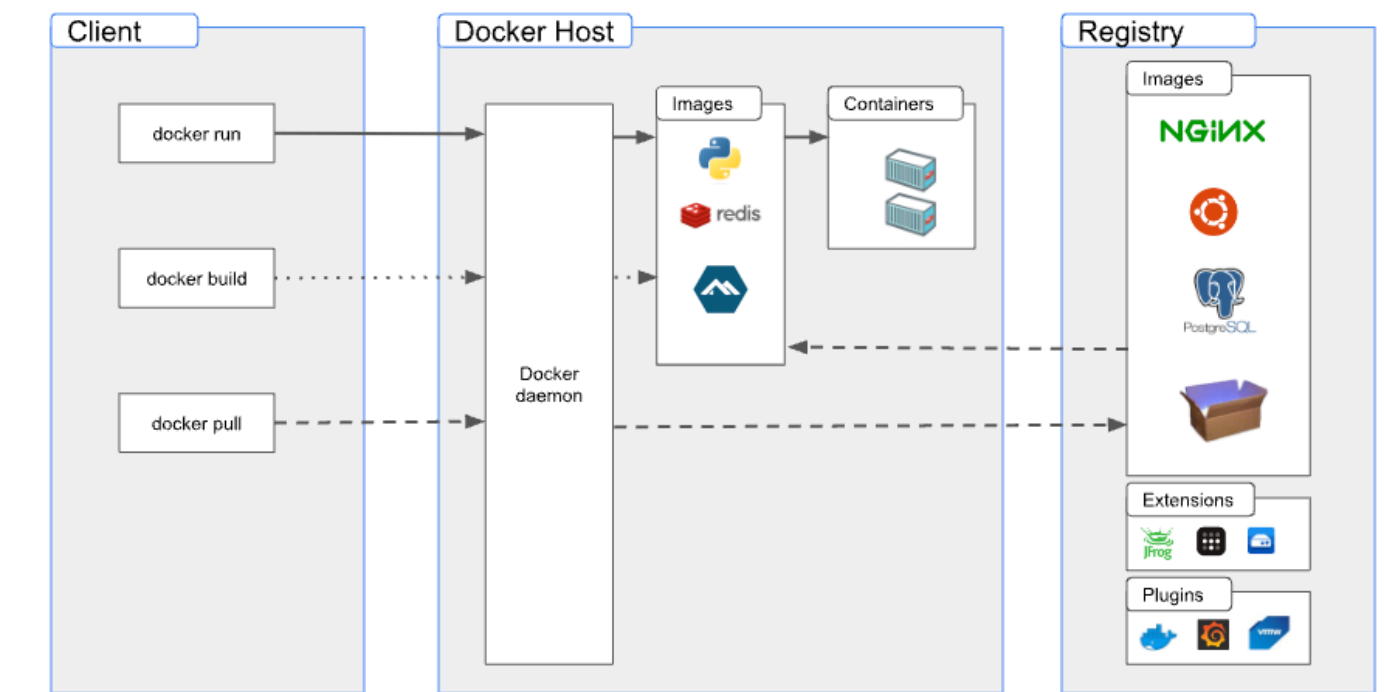
## 도커 공식 홈페이지 훑어보기



# Docker Architecture

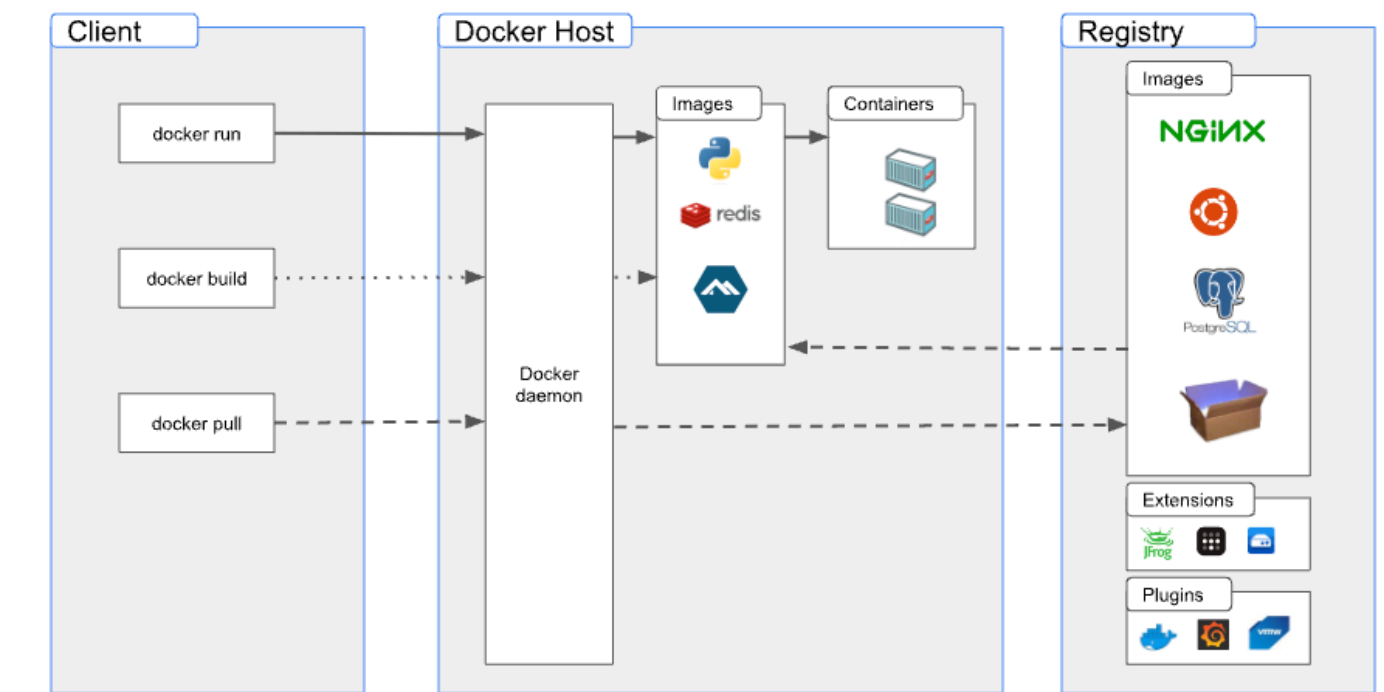


# Docker Architecture



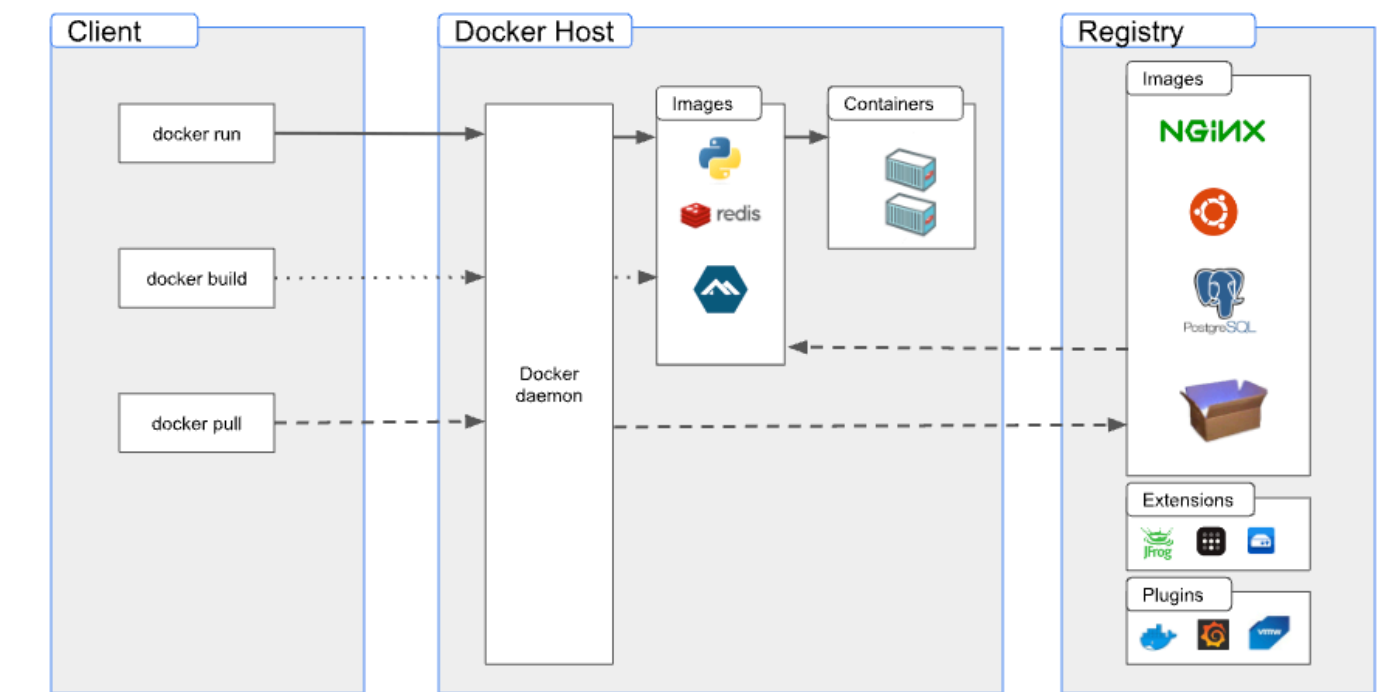
- 도커 데몬(Docker daemon = dockerd)
  - 도커 엔진의 핵심 구성 요소
  - 도커 호스트에서 컨테이너를 관리하고 실행하는 역할
  - 컨테이너를 생성, 시작, 중지, 삭제하는 등의 작업을 수행
  - 컨테이너 이미지를 관리하고
  - 외부에서 이미지를 다운로드하고 빌드하는 작업을 수행

# Docker Architecture



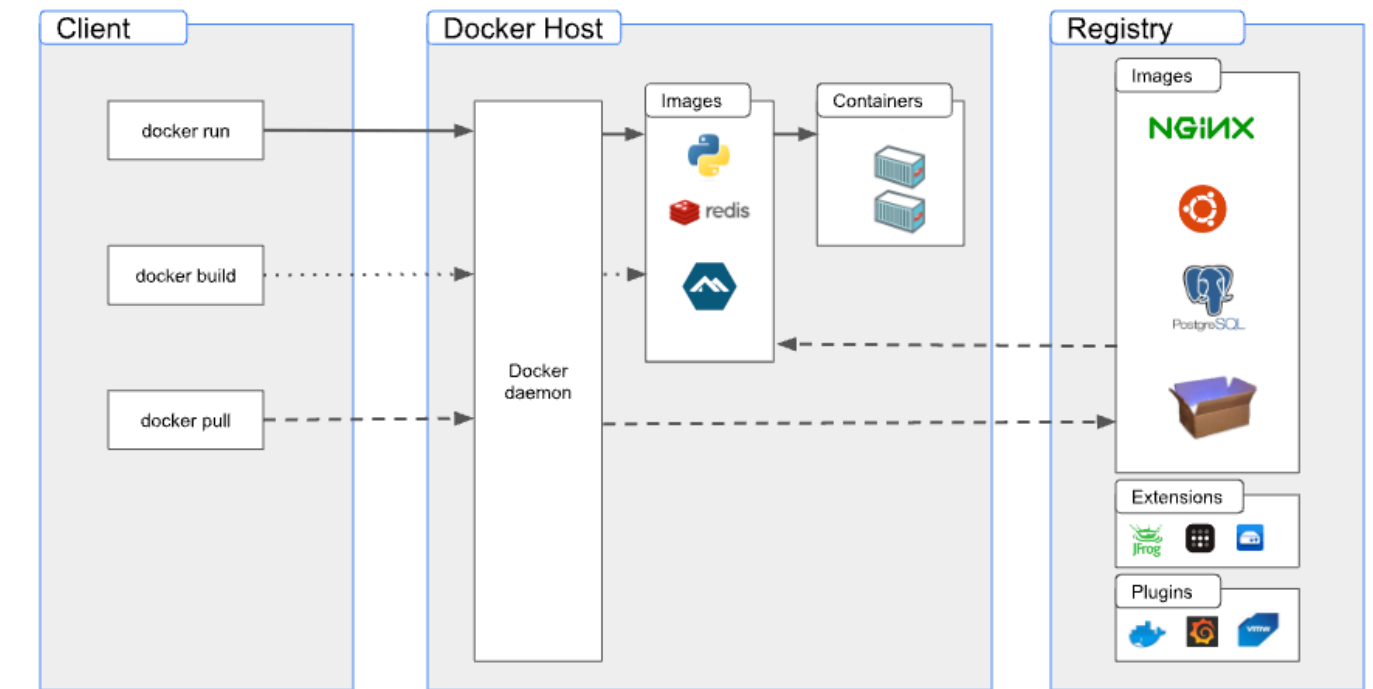
- 도커 클라이언트 (Docker Client)
  - Docker와 상호 작용
  - docker 명령어를 사용하면 Docker daemon으로 보내어 실행

# Docker Architecture

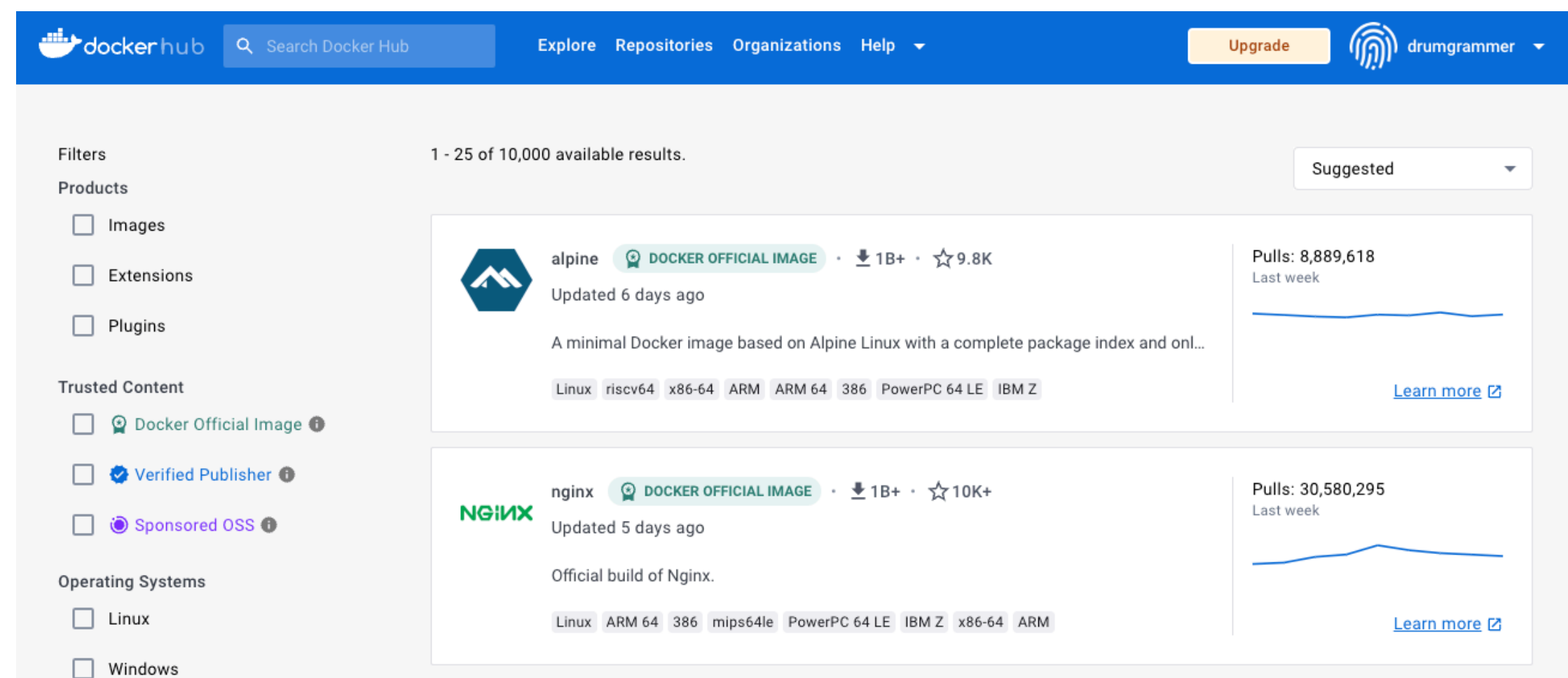


- 도커 오브젝트 (Docker Object)
  - 도커 이미지 (Docker Image)
    - 도커 컨테이너를 만들기 위한 읽기 전용 템플릿
  - 도커 컨테이너 (Docker Container)
    - 한 도커 이미지의 실행 가능한 인스턴스
    - 애플리케이션을 실행하기 위한 모든 파일과 설정 정보를 포함하는 패키지

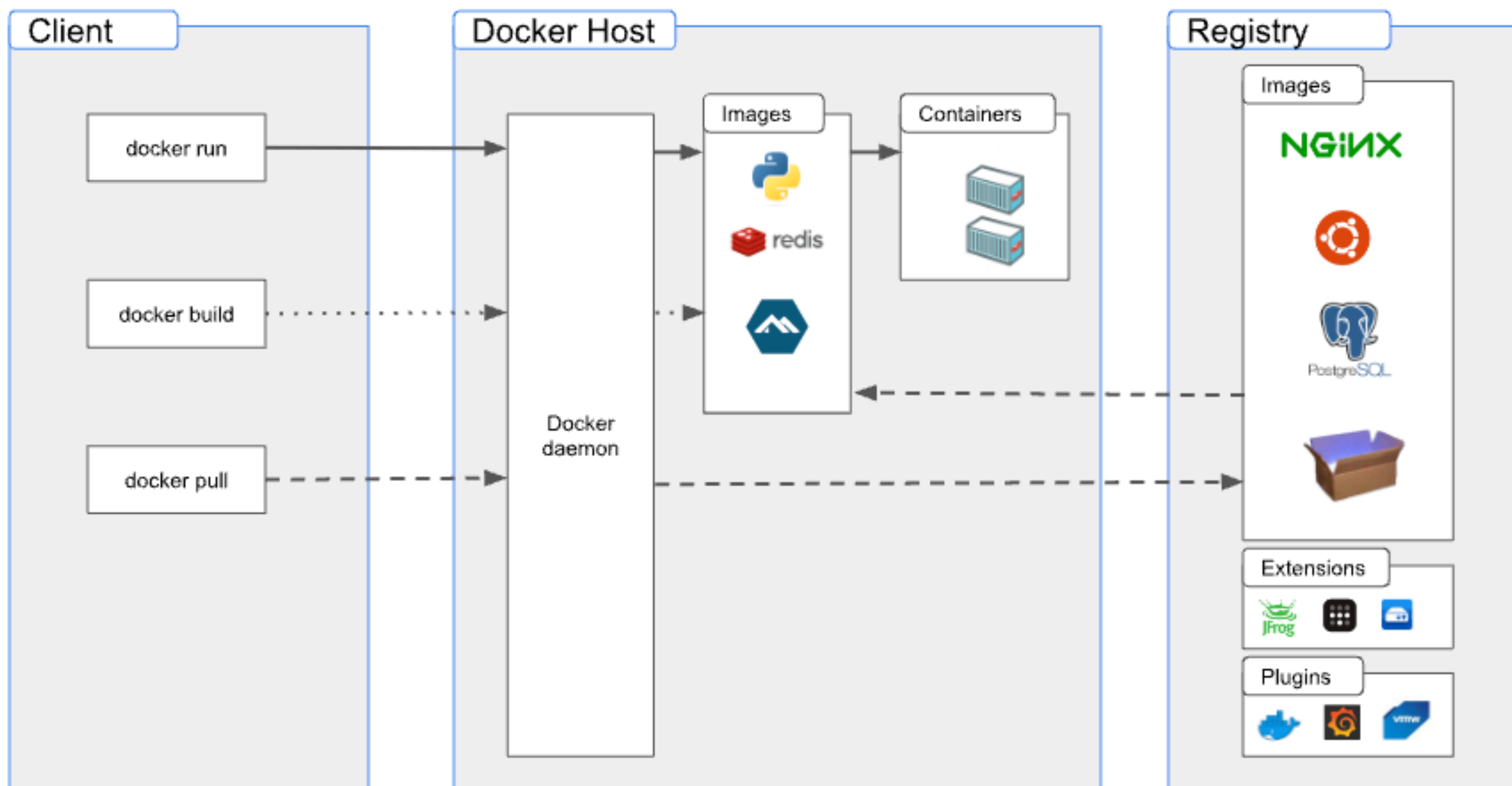
# Docker Architecture



- 도커 레지스트리 (Docker Registries)
- 도커 이미지 (Docker Image) 를 관리하고 저장하는 곳
- Docker hub: 디폴트 레지스트리, 누구나 접근 가능한 공개형 저장소



# Docker Architecture





App Store

Program

Process

Docker Hub



docker pull

Image



docker run


Container


# Docker 사용해보자!

## 실습편

# Docker 설치하기

## 도커 공식 홈페이지 - Get Docker

 docker docs

 Search the docs

[Home](#)

[Guides](#)

[Manuals](#)

[Reference](#)

[Samples](#)

[Contribute](#)

[Home](#) / [Guides](#) / [Get Docker](#)

Docker overview

[Get Docker](#)

Get started

Docker Desktop hands-on guides

Language-specific guides

Develop with Docker

Deploy your app to the cloud


Run your app in production

Educational resources

## Get Docker


Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

You can download and install Docker on multiple platforms. Refer to the following section and choose the best installation path for you.




### Docker Desktop for Mac

A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.




### Docker Desktop for Windows

A native Windows application which delivers all Docker tools to your Windows computer.



### Docker Desktop for Linux

A native Linux application which delivers all Docker tools to your Linux computer.

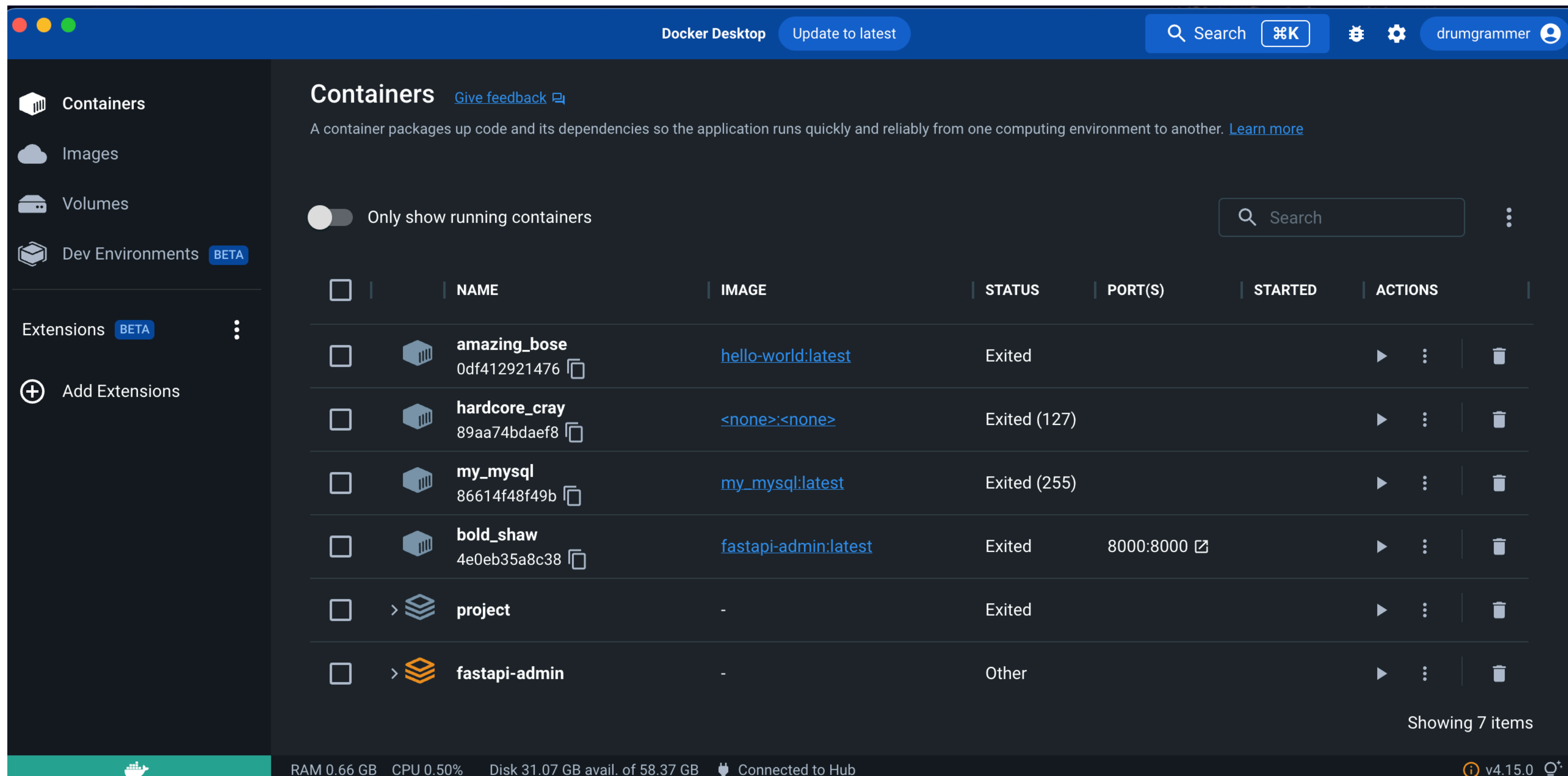
 **Note**

If you're looking for information on how to install Docker Engine, see [Docker Engine installation overview](#).

공식 설치 메뉴얼: <https://docs.docker.com/desktop/install/mac-install/>

# Docker 실행하기

## Docker Desktop Dashboard - Docker에서 제공하는 GUI 환경



# Docker 실행하기

## Terminal 환경에서 확인하기 - Docker CLI

```
→ ~ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS              PORTS                               NAMES
fece6ff9e9f5   fastapi-admin  "uvicorn examples.ma..." 8 months ago   Restarting (3) 41 seconds ago      0.0.0.0:6379->6379/tcp             fastapi-admin_app_1
adabd9d580c6   redis         "docker-entrypoint.s..." 11 months ago   Up 4 hours                                     path2unicorn_redis_1

→ ~ docker images
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
hello-world         latest      8b8df8492a06  7 weeks ago   7.46MB
<none>              <none>      353eb2624a5d  7 weeks ago   7.46MB
testcontainers/ryuk 0.3.4       5d28cedc492d  8 months ago   11.5MB
my_mysql            latest      38b3476c07cf  8 months ago   497MB
mysql-allowed-many-connections latest      38b3476c07cf  8 months ago   497MB
fastapi-admin       latest      77390b585967  8 months ago   301MB
mariadb             10         7185139442f4  14 months ago  394MB
redis               latest      6f72dd2e7b80  14 months ago  107MB
rabbitmq            3-management-alpine d6aa79d5b205  14 months ago  160MB
ubuntu              14.04      7304c635fe52  17 months ago  187MB
testcontainers/ryuk 0.3.3       146fb8d54138  17 months ago  11.5MB
mysql/mysql-server  8.0.26     f7f3acd8a80c  20 months ago  497MB
```



# Docker 실행하기

## Docker CLI

docker docs

Search the docs

Home

Guides

Manuals

Reference

Samples

Contribute

🏠

Reference

Command-line reference

Docker CLI (docker)

Docker run reference

Reference documentation

Command-line reference

Docker CLI (docker)

Docker run reference

Use the Docker command line

docker (base command)

docker attach

docker build

docker builder

docker buildx

docker checkpoint

docker commit

docker compose

docker config

docker container

docker context

docker cp

docker create

docker diff

docker events

docker exec

docker export

Docker run reference

Docker runs processes in isolated containers. A container is a process which runs on a host. The host may be local or remote. When an operator executes `docker run`, the container process that runs is isolated in that it has its own file system, its own networking, and its own isolated process tree separate from the host.

This page details how to use the `docker run` command to define the container’s resources at runtime.

General form

The basic `docker run` command takes this form:

```
$ docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
```

The `docker run` command must specify an *IMAGE* to derive the container from. An image developer can define image defaults related to:

- detached or foreground running
- container identification
- network settings
- runtime constraints on CPU and memory

With the `docker run [OPTIONS]` an operator can add to or override the image defaults set by a developer. And, additionally, operators can override nearly all the defaults set by the Docker runtime itself. The operator’s ability to override image and Docker runtime defaults is why *run* has more options than any other `docker` command.

To learn how to interpret the types of `[OPTIONS]`, see *Option types*.

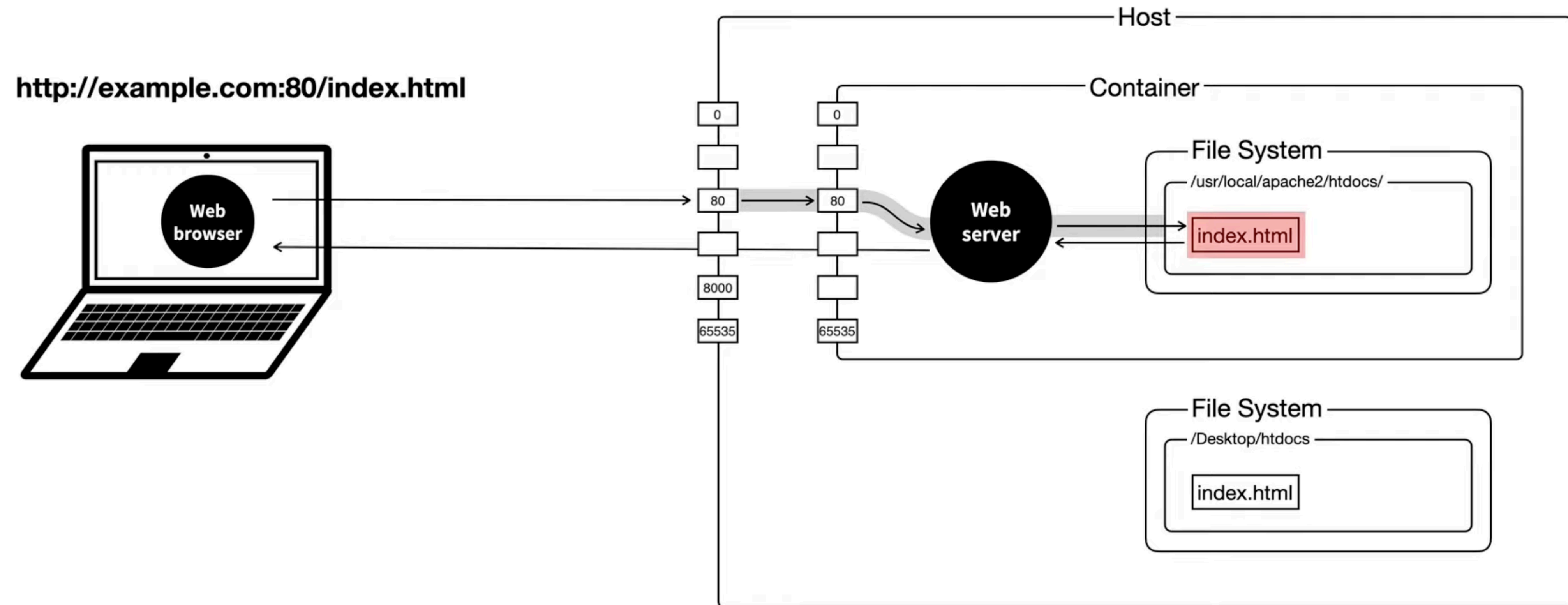
Note

Depending on your Docker system configuration, you may be required to preface the `docker run` command with `sudo`. To avoid having to use `sudo` with the `docker` command, your system administrator can create a Unix group called `docker` and add users to it. For more information about this configuration, refer to the Docker installation documentation for your operating system.

# Docker CLI

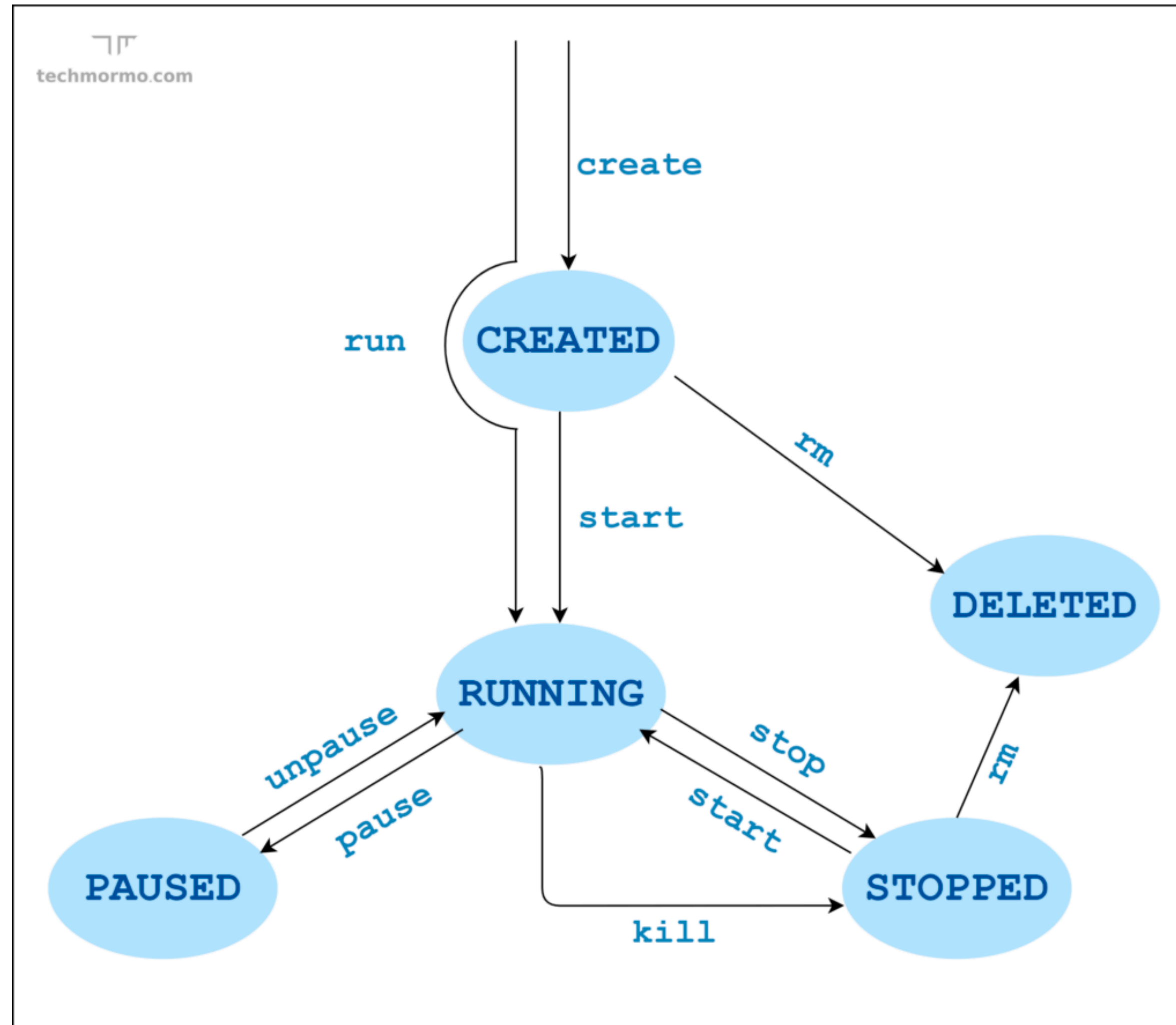
- **Download an image from a registry**
  - `docker pull [OPTIONS] NAME[:TAG|@DIGEST]`
- **List images**
  - `docker images [OPTIONS] [REPOSITORY[:TAG]]`
- **Create and run a new container from an image**
  - `docker run [OPTIONS] IMAGE [COMMAND] [ARG...]`
- **Stop one or more running containers**
  - `docker stop [OPTIONS] CONTAINER [CONTAINER...]`
- **Fetch the logs of a container**
  - `docker logs [OPTIONS] CONTAINER`
- **Remove one or more containers**
  - `docker rm [OPTIONS] CONTAINER [CONTAINER...]`
- **Remove one or more images**
  - `docker rmi [OPTIONS] IMAGE [IMAGE...]`

# Docker Network

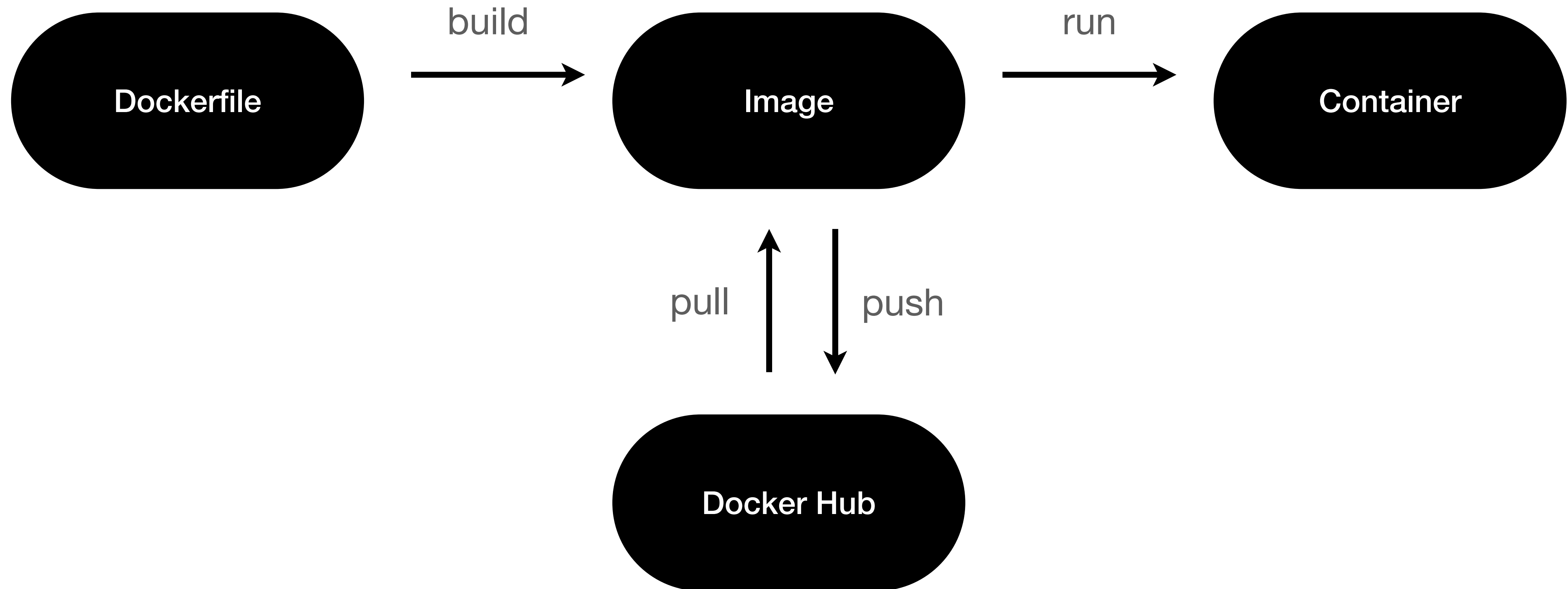




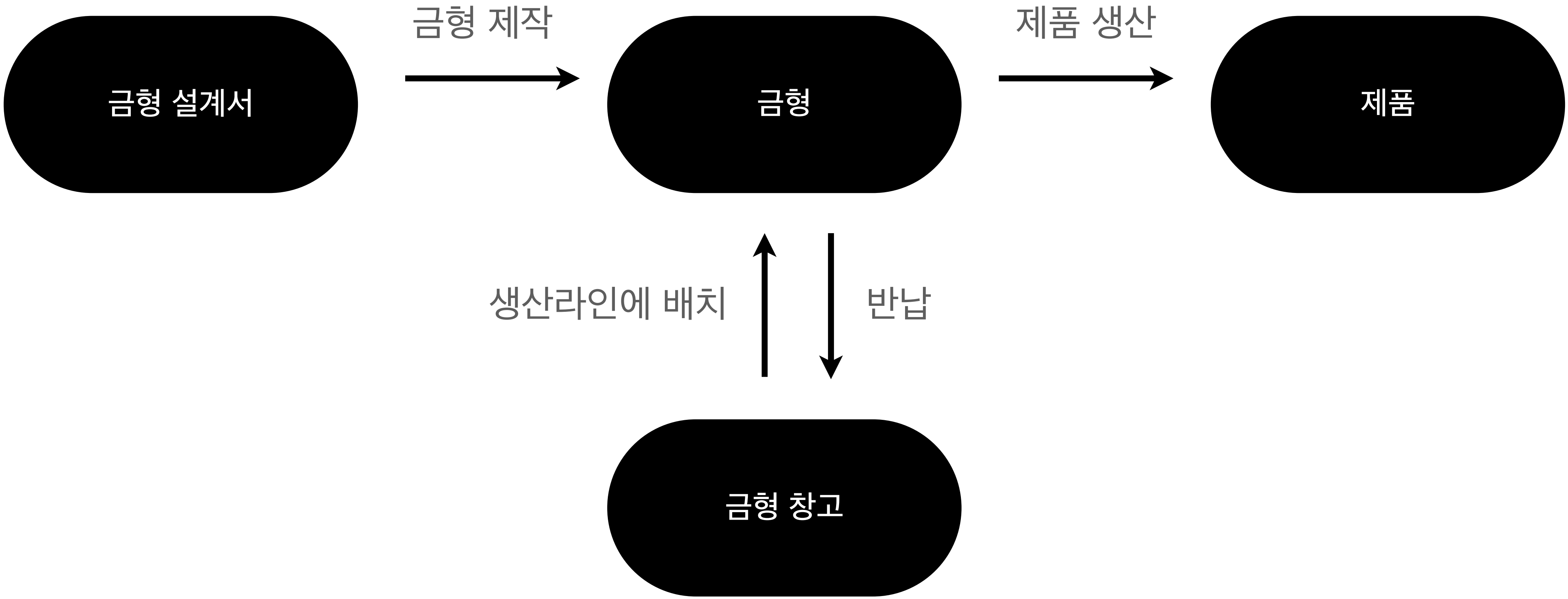
# The Container Lifecycle



# Dockerfile



# Dockerfile



**수고하셨습니다!**