

Docker를 이용한
Container 활용

❖ 컨테이너와 도커

- 컨테이너란 리눅스 기반의 기술로써 서비스 구동에 필요한 파일들만으로 이루어진 컨테이너 이미지를 이용하여 동작시킨 가상서버. VM과 다르게 필요한 최소한의 파일들로 이루어져 용량이 작음.
- 도커란 컨테이너 이미지, 컨테이너의 생성, 제거 등을 위한 컨테이너 엔진
- 컨테이너 이미지 생성시 개발환경과 같은 환경으로 미리 구성하여 생성함으로써 해당 이미지를 이용해 생성하는 컨테이너는 항상 같은 환경으로 동작함. 즉 배포환경을 빠르게 구축할 수 있음.

컨테이너(Container) 기술 배경

- 컨테이너라는 개념은 2000년대 중반 리눅스에 내장된 LXC(Linux Container)기술로부터 처음 소개되기 시작. LXC는 단일 머신상에 여러 개의 독립된 리눅스 커널 컨테이너를 실행하기 위한 OS 레벨의 가상화 기법으로 컨테이너 개념의 시초

- 네트워크, 스토리지, 보안 등 각 영역마다 정책이 서로 다르기 때문에 컴퓨터 프로그램들은 환경이 바뀔 때마다 각종 오류가 발생. 이는 소프트웨어 개발자들의 오랜 골칫거리. 따라서 소프트웨어가 현재의 컴퓨팅 환경에서 다른 환경으로 이동하더라도 안정적으로 실행되도록 하기 위해 나온 개념이 바로 컨테이너

- 최근 클라우드 컴퓨팅에서는 컨테이너 기반 가상화가 기존의 하이퍼바이저 기반의 가상화 기술을 대체하며 각광받고 있음

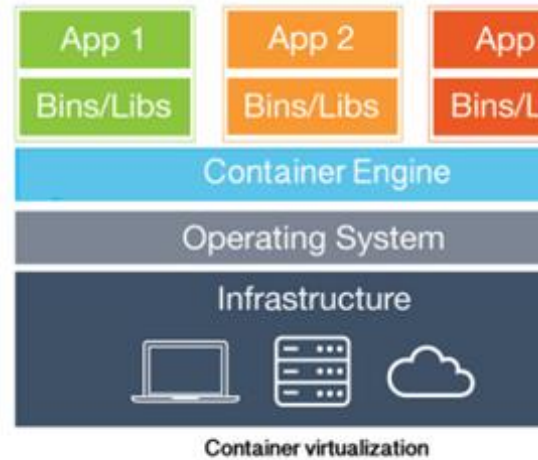
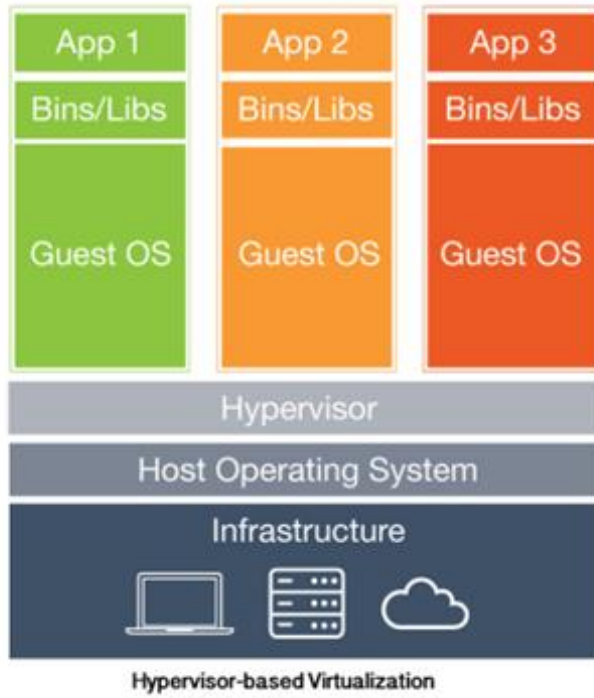
- 특히 구글은 Gmail, Google Drive를 포함한 모든 서비스를 컨테이너로 제공한다고 발표하였으며, 현재 자사의 컨테이너 플랫폼인 '쿠버네티스(Kubernetes)'를 통해 2014년부터 매주 20억개 이상의 컨테이너를 구동하고 있음

컨테이너 개념 및 구조

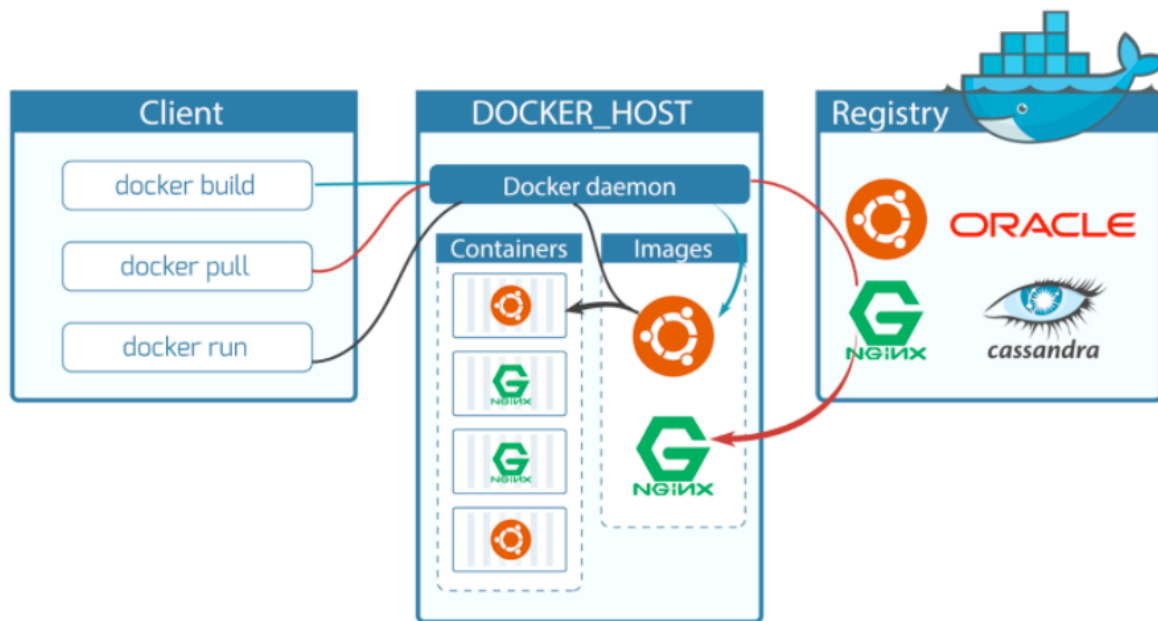
- 컨테이너는 모듈화되고 격리된 컴퓨팅 공간 또는 컴퓨팅 환경, 다시 말해 어플리케이션을 구동하는 환경을 격리한 공간을 의미
- 기본적으로 가상화를 위해 하이퍼바이저와 게스트 OS가 필요했던 것과는 달리, 컨테이너는 운영 체제를 제외하고 어플리케이션 실행에 필요한 파일만을 패키징(Packaging)한 형태. 그만큼 기존의 가상머신에 비해 가볍고 빠르게 동작이 가능
- '운영 체제의 커널(Kernel)'이 여러 격리된 사용자 공간 인스턴스를 갖출 수 있도록 하는 가상화 방식이기 때문에 'OS 레벨 가상화'라고 불림

*커널: 운영 체제의 핵심이 되는 컴퓨터 프로그램. 운영 체제의 다른 부분 및 응용 프로그램 수행에 필요한 여러 서비스를 제공하는 역할을 하며, 메모리나 저장장치 내에서 운영 체제의 주소 공간을 관리

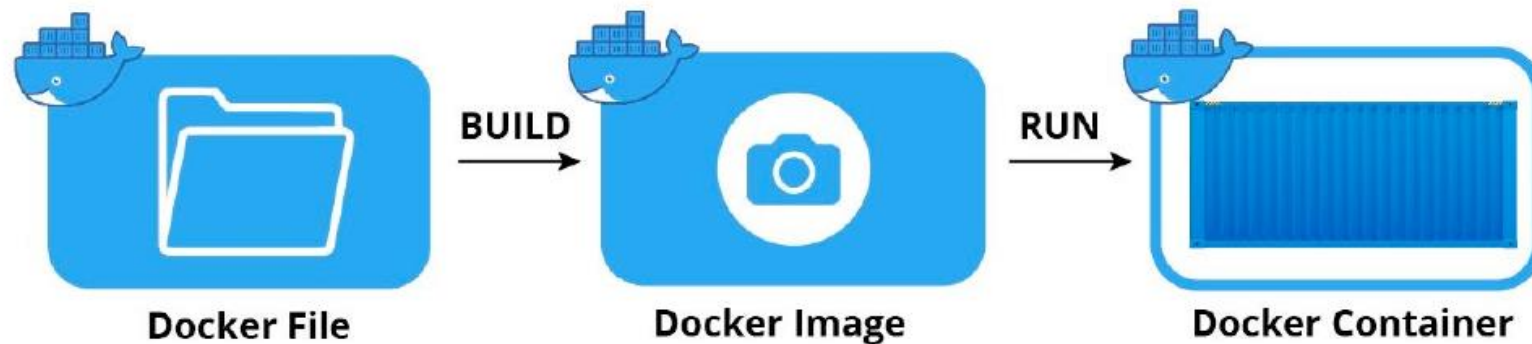
VM vs Container



도커 구성 요소



도커 이미지와 컨테이너



이미지 (Image) vs 컨테이너 (Container)
프로그램 (Program) vs 프로세스 (Process)
클래스 (Class) vs 인스턴스 (Instance)

도커 이미지 이름 구성

Registry	hub.docker.com
Repository/Image Name	goodeeacademy/nginx
Tag	latest

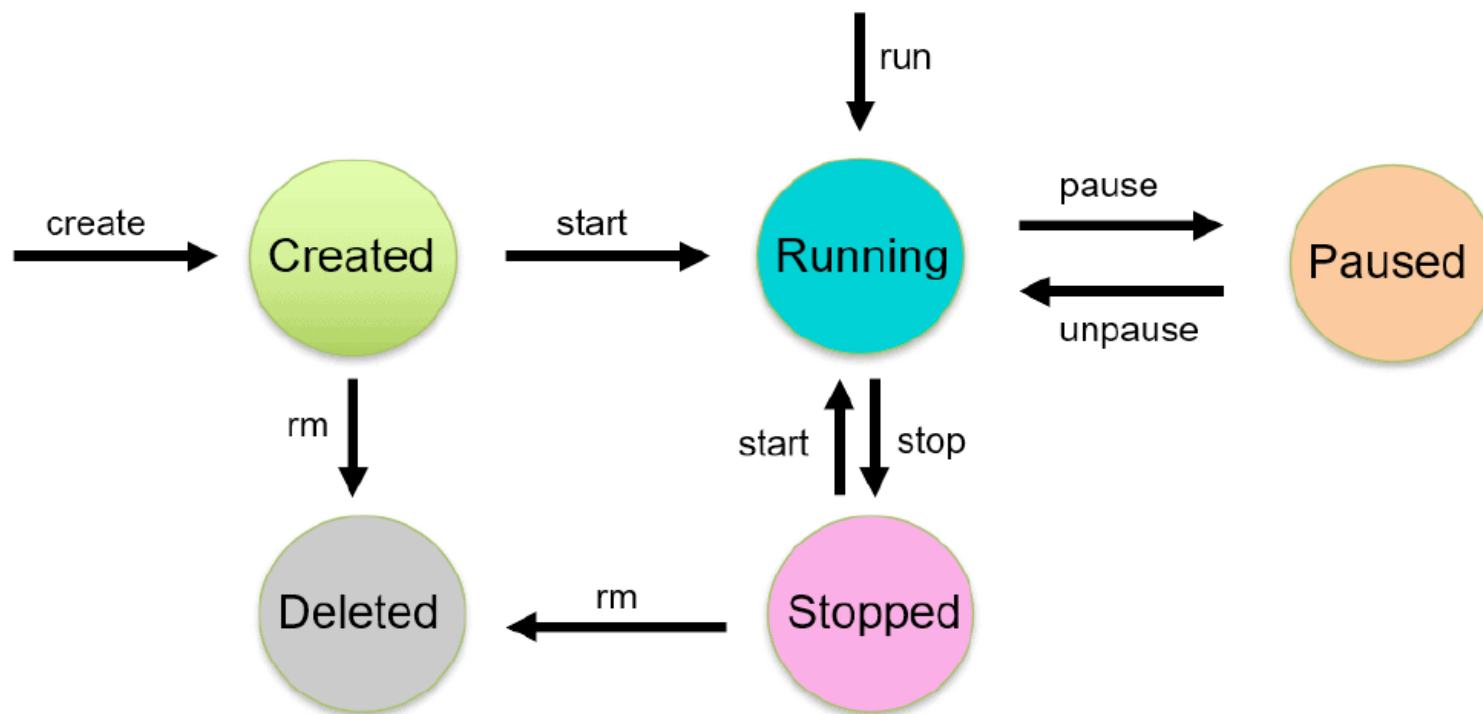
- 도커 이미지 Pull / Push 시에 Registry를 생략하면 기본 Registry인 도커 허브로 인식
- 도커 이미지 태그를 생략하면 최신 버전을 가리키는 latest로 인식

도커 이미지 저장소(Registry)

도커 이미지를 관리하기
위한 클라우드상의 공간

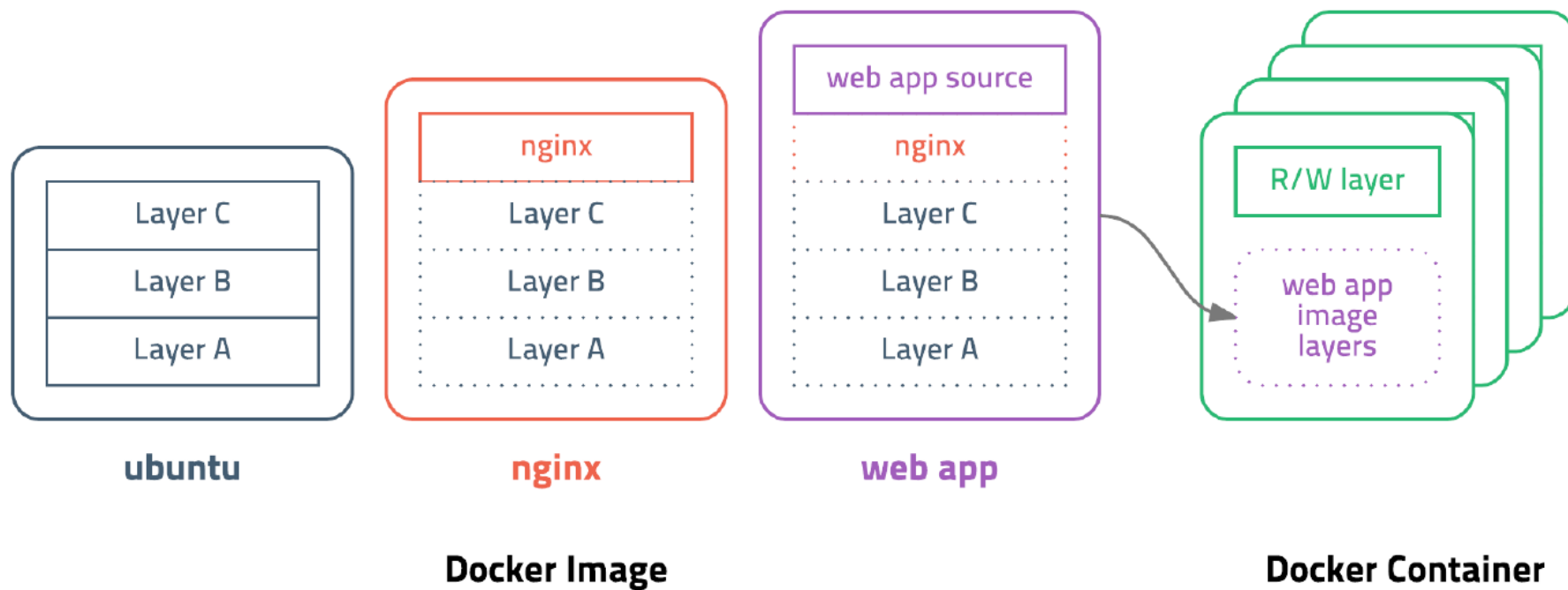


도커 컨테이너 라이프사이클

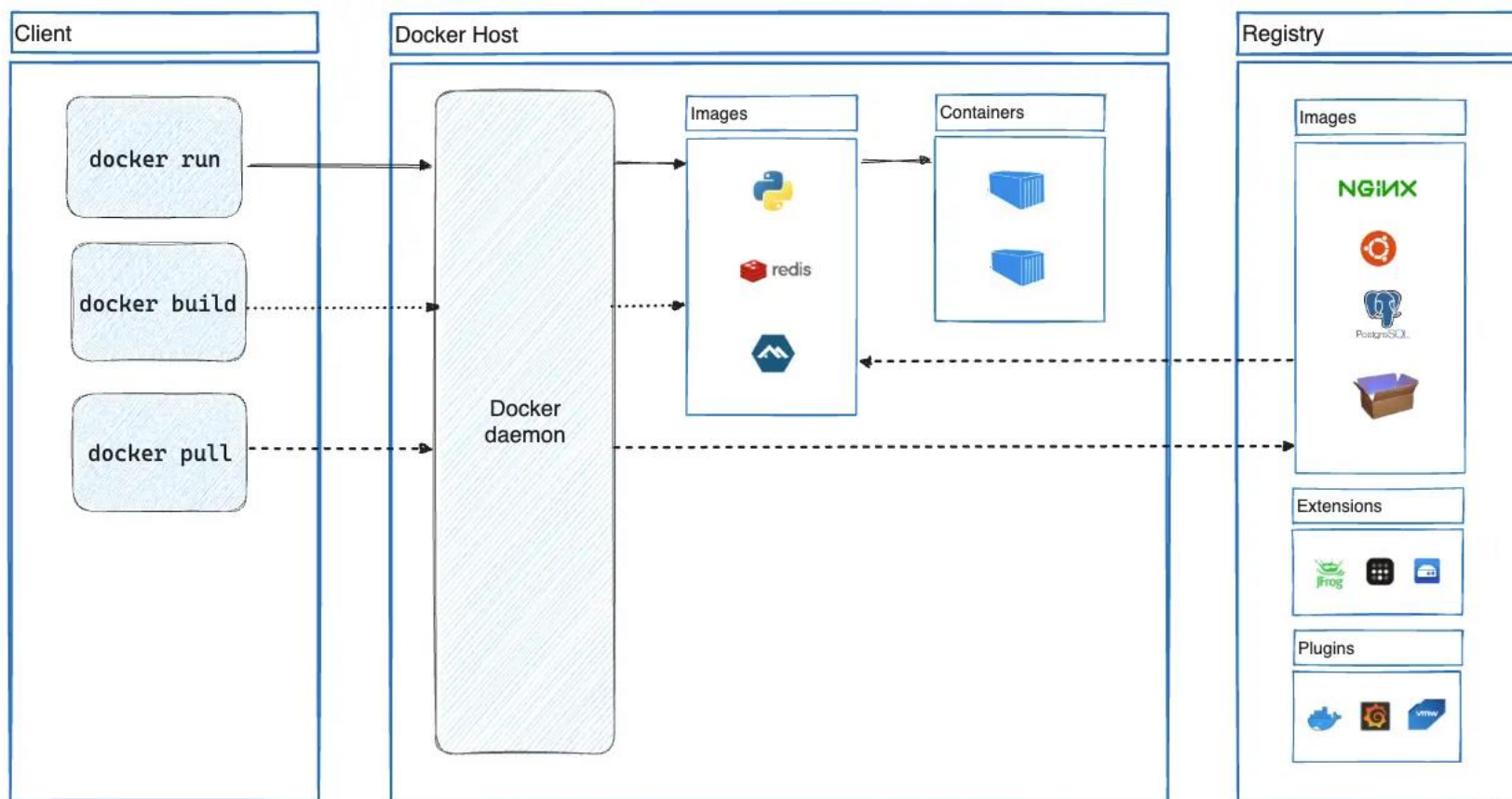


도커 이미지 Layer

도커 이미지 구조



도커 아키텍처



도커 설치하기

- <https://docs.docker.com/engine/install/>

Install Docker Engine

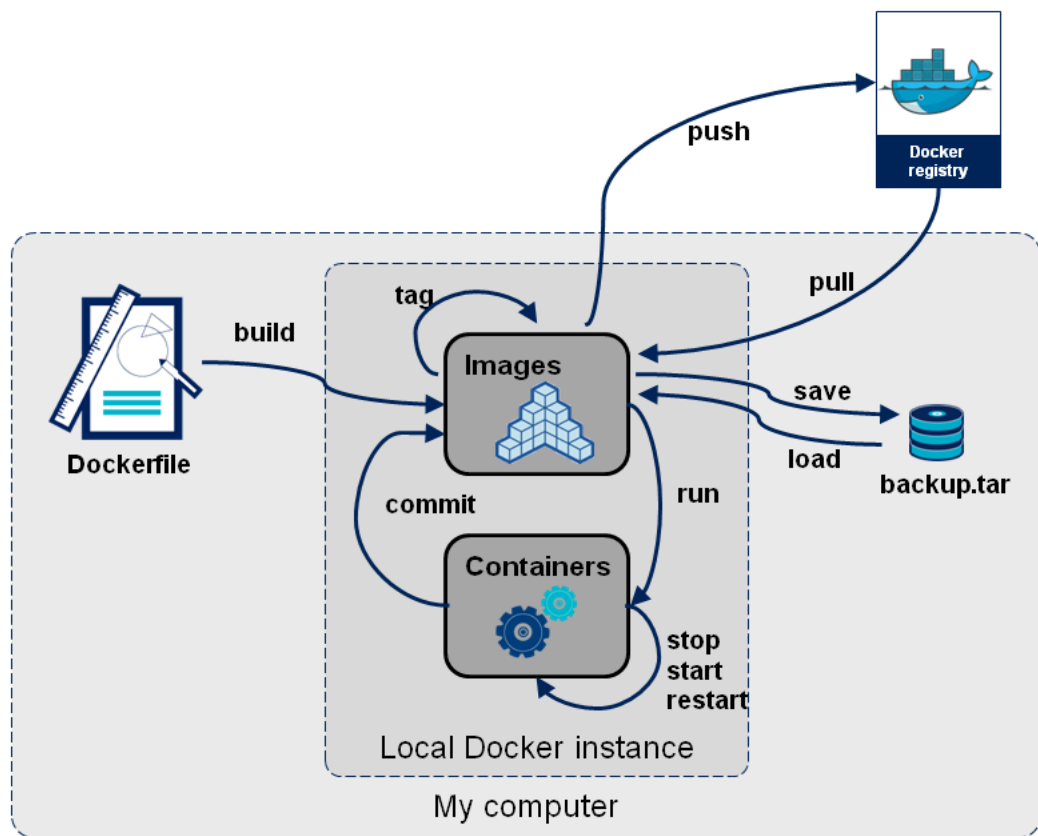
This section describes how to install Docker Engine on Linux, also known as Docker CE. Docker Engine is also available for Windows, macOS, and Linux, through Docker Desktop. For instructions on how to install Docker Desktop, see:

- [Docker Desktop for Linux](#)
- [Docker Desktop for Mac \(macOS\)](#)
- [Docker Desktop for Windows](#)

도커 설치 - ubuntu

- # 도커 공식 GPG key 추가하기:
- `sudo apt-get update`
- `sudo apt-get install ca-certificates curl`
- `sudo install -m 0755 -d /etc/apt/keyrings`
- `sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc`
- `sudo chmod a+r /etc/apt/keyrings/docker.asc`
- # 저장소 추가하기:
- `echo ₩`
- `"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu ₩`
- `$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | ₩`
- `sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`
- `sudo apt-get update`
- `sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`

도커 이미지 관련 명령



```
admin@gd-svr-1:~$ docker image --help
```

Usage: docker image COMMAND

Manage images

Commands:

- | | |
|------------------|--|
| build | Build an image from a Dockerfile |
| history | Show the history of an image |
| import | Import the contents from a tarball to create a filesystem image |
| inspect | Display detailed information on one or more images |
| load | Load an image from a tar archive or STDIN |
| ls | List images |
| prune | Remove unused images |
| pull | Pull an image or a repository from a registry |
| push | Push an image or a repository to a registry |
| rm | Remove one or more images |
| save
default) | Save one or more images to a tar archive (streamed to STDOUT by default) |
| tag | Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE |

도커 이미지 다운로드

- docker pull repository
 - docker pull ubuntu => tag를 생략하면 기본값으로 latest가 추가됨.
- docker pull repository:tag
 - docker pull ubuntu:20.04

도커 이미지 목록 확인

- docker image list
- docker images repository
 - docker images ubuntu

도커 이미지 삭제

- `docker rmi IMAGE`
- `docker image rm IMAGE`

도커 이미지 정보조회

- `docker inspect NAME|ID`

도커 컨테이너 관리

- 컨테이너 실행하기
 - `docker run [options] image [command] [arg ...]`
- 컨테이너 목록보기
 - `docker ps [options]`
 - `--all, -a` 상태와 관계없이 모든 컨테이너 표시
 - `--filter, -f` 지정된 조건에 맞는 컨테이너만 표시
 - `--size, -s` 전체 파일 사이즈도 표기

Option	Short	Description
<code>--name</code>		컨테이너의 이름 지정
<code>--detach</code>	<code>-d</code>	컨테이너를 백그라운드에서 실행
<code>--env</code>	<code>-e</code>	환경변수를 설정
<code>--env-file</code>		환경변수를 저장한 파일 설정
<code>--expose</code>		포트 또는 포트범위 노출
<code>--publish</code>	<code>-p</code>	컨테이너의 포트 공개
<code>--rm</code>		컨테이너가 종료되면 자동 삭제
<code>--interactive</code>	<code>-i</code>	STDIN을 활성화
<code>--tty</code>	<code>-t</code>	pseudo-tty를 할당
<code>--volume</code>	<code>-v</code>	볼륨 설정

도커 컨테이너 관리

- 컨테이너 로그 확인하기

`docker logs [options] name_or_id`
--follow, -f 계속 로그를 표시

- 컨테이너 접속하기

Attach

`docker attach [options] container`

현재 실행중인 컨테이너 터미널의 STDIN, STDOUT, STDERR에 접근

주의사항 : ctrl+c 또는 exit를 통해 프로세스를 종료하면 컨테이너도 함께 종료됨.

종료되지 않게 하려면 ctrl + p + q를 눌러 종료해야함.

Exec

`docker exec -it [options] id_or_name /bin/bash|sh|...`

현재 실행중인 컨테이너에서 새로운 명령을 실행함.

도커 컨테이너 관리

- 컨테이너 내부 명령 실행하기

`docker exec [options] CONTAINER COMMAND [ARG ...]`

- 컨테이너 종료/시작하기

`docker stop <NAME_OR_ID>`

`docker start <NAME_OR_ID>`

도커 컨테이너 관리

- 컨테이너 로그 확인하기

`docker logs [OPTIONS] <NAME | ID>`

`-f(--follow)` : 계속 로그를 표시합니다.

`-n(--tail) n` : 마지막 n개의 로그를 표시합니다.

- nginx 컨테이너를 실행하고 접속 테스트 후 마지막 로그 확인

```
$ docker run --rm -d -p 80:80 --name nginx nginx:latest
```

```
$ curl 127.0.0.1:80
```

```
$ docker logs --tail 1 nginx
```


도커 컨테이너 관리

- 컨테이너 접속하기

`docker attach CONTAINER`

현재 실행중인 컨테이너 터미널의 STDIN,STDOUT,STDERR에 접근

주의) ctrl+c를 통해 프로세스 종료시 컨테이너가 종료됨

컨테이너를 종료하지 않고 접속을 종료하기 위해 ctrl + p + q입력

`docker exec -it <NAME | ID> /bin/bash`

도커 컨테이너 관리

- 컨테이너 내부 명령 실행하기

`docker exec CONTAINER COMMAND`

`$ docker exec NAME|ID sh -c "pwd"`

- workdir에서 명령 실행

`$ docker exec --workdir /bin CONTAINER sh -c "pwd"`

도커 컨테이너 관리

- 컨테이너 종료 & 시작

\$ docker stop <NAME|ID>

\$ docker start <NAME|ID>

- 컨테이너 중지 & 재실행

\$ docker pause <NAME|ID>

\$ docker unpause <NAME|ID>

도커 컨테이너 관리

- 컨테이너 삭제

\$ docker rm CONTAINER

-f(--force) : 실행중인 컨테이너 강제 삭제

-v(--volumes) : 컨테이너의 anonymous volumes도 함께 삭제

- 종료된 모든 컨테이너 삭제

\$ docker rm \$(docker ps -a -q -f status=exited)

컨테이너 삭제

- 컨테이너 삭제

\$ docker rm CONTAINER

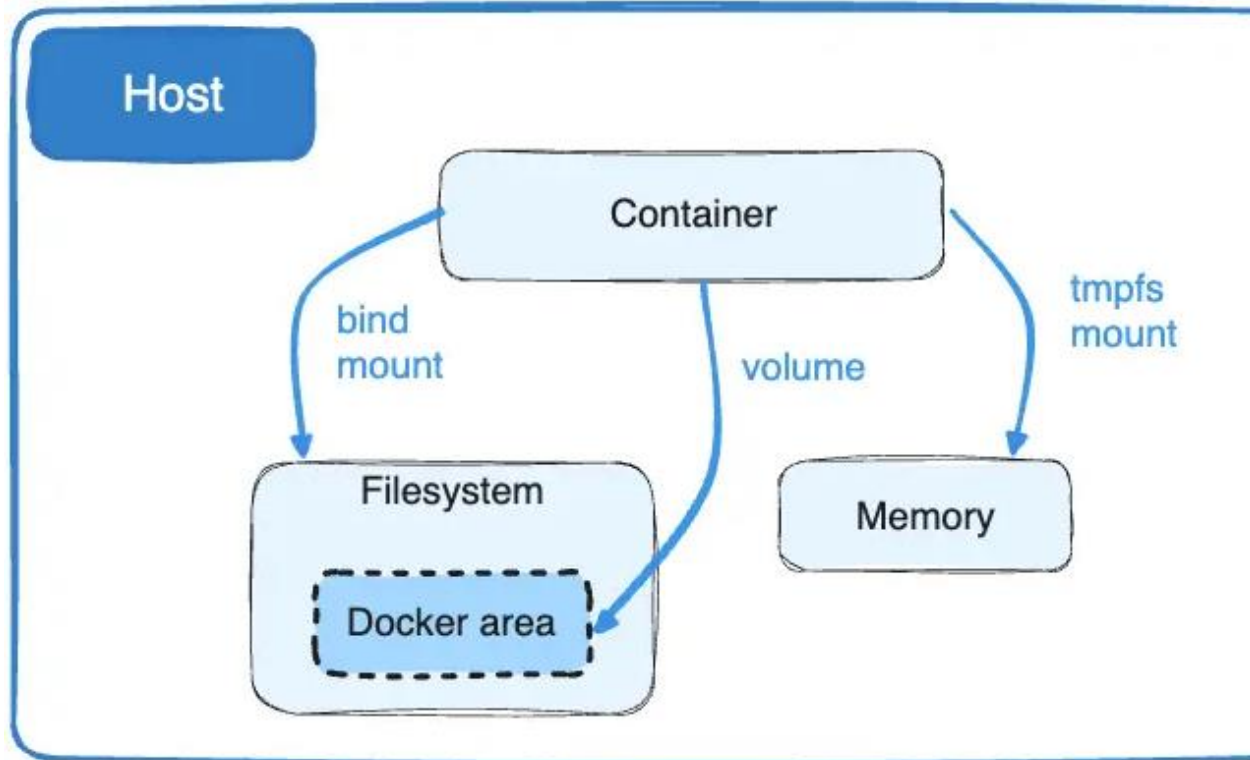
-f(--force) : 실행중인 컨테이너 강제 삭제

-v(--volumes) : 컨테이너의 anonymous volumes도 함께 삭제

- 종료된 모든 컨테이너 삭제

\$ docker rm \$(docker ps -a -q -f status=exited)

Docker 볼륨



도커볼륨은 컨테이너에서 생성,
재사용할 수 있고 호스트 운영체제
에서
직접 접근이 가능

또한 보존되어야하는 데이터를
유지(데이터 영속성과 지속성)하기
위한
메커니즘 제공

도커 볼륨 활용

- mariadb 최신버전 도커 이미지를 이용하여 컨테이너 생성.
컨테이너 생성시 ~/mariadb-vol 디렉토리를 컨테이너의 /var/lib/mysql 디렉토리와 연동되도록 설정하기

Dockerfile

- DockerFile은 도커이미지, 즉 필요로 하는 개발환경을 제공하기 위한 여러가지 명령어들의 집합체이다.

Instruction Description

FROM	사용할 Base 이미지를 설정합니다.
ARG	build 에 사용할 변수를 설정합니다.
ENV	환경 변수를 설정합니다.
ADD	파일 또는 폴더(directory)를 추가합니다.
COPY	파일 또는 폴더(directory)를 복사합니다.
LABEL	라벨을 추가합니다.
EXPOSE	포트를 외부에 노출합니다.
VOLUME	볼륨마운트를 생성합니다.
USER	User 와 Group 을 설정합니다.
WORKDIR	Working Directory 를 변경합니다.
RUN	명령어를 실행합니다.
CMD	기본 명령어를 정의합니다.
ENTRYPOINT	필수로 실행할 명령어를 정의합니다.

Dockerfile

FROM ubuntu:18.04

RUN apt-get update && apt-get install -y apache2 && \
apt-get clean -y

EXPOSE 80

CMD ["apache2ctl","-D","FOREGROUND"]

Dockerfile

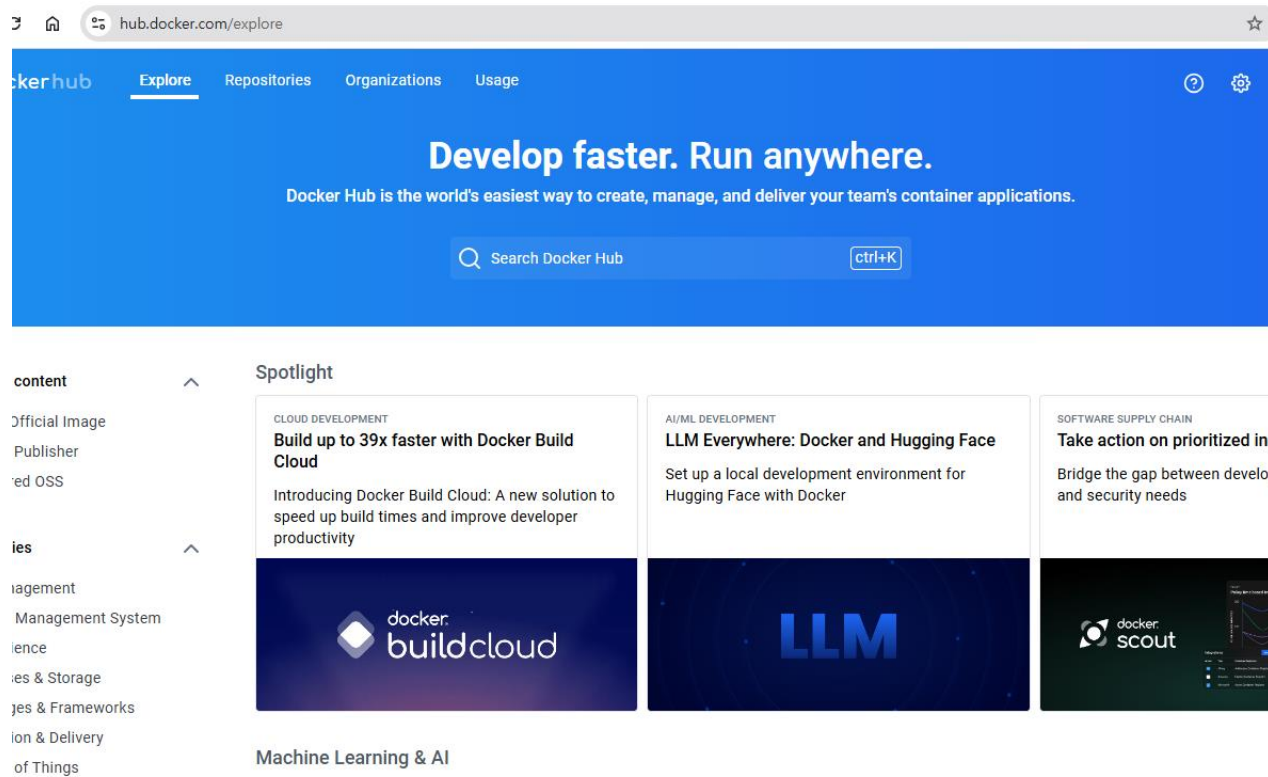
```
FROM openjdk:11
ARG VERSION
COPY target/guestbook-0.0.1-SNAPSHOT.jar /app/guestbook.jar
LABEL maintainer="YuTaek Kim<sunnykid7@gmail.com>" \#
      title="Guestbook App" \#
      version="$VERSION" \#
      description="This image is guestbook service"
ENV APP_HOME /app
EXPOSE 80
VOLUME /app/upload
WORKDIR $APP_HOME
ENTRYPOINT ["java"]
CMD ["-jar","guestbook.jar"]
```

Dockerfile로 이미지파일 만들기

현재 디렉토리의 Dockerfile을 이용하여 이미지 생성하기

```
docker image build -t 이미지명:태그 .
```

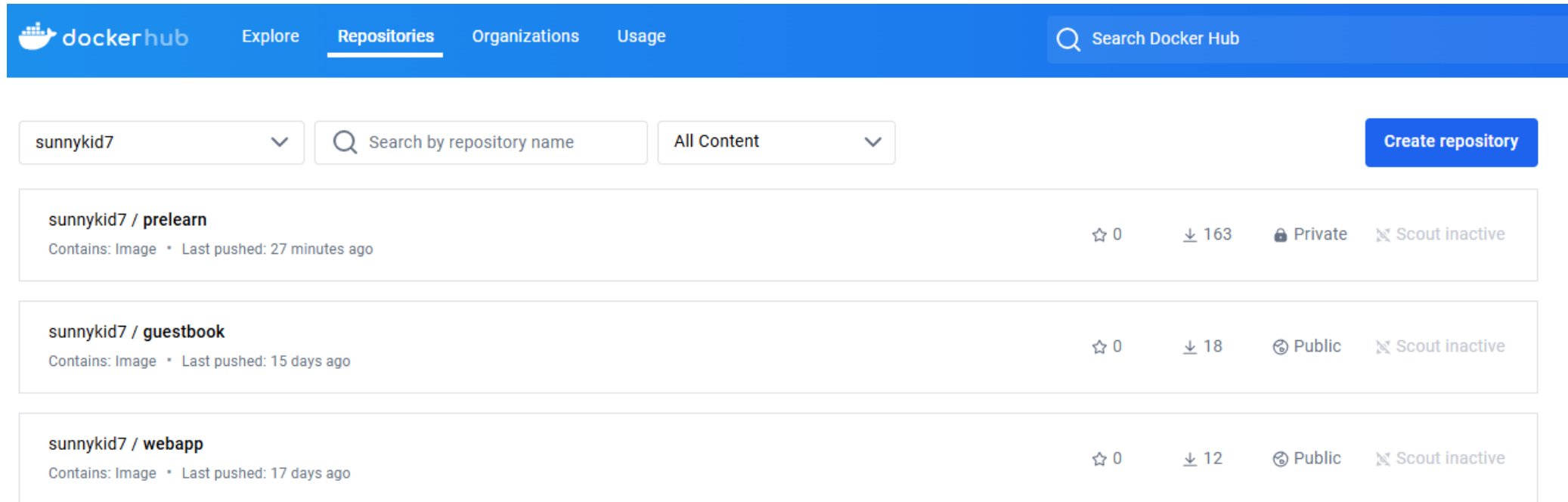
Dockerhub 사용하기



- 도커허브란 Github과 같이 다양한 도커이미지를 공유하기 위한 저장 공간으로 도커의 공식 허브 사이트임.

도커허브에 가입하고 저장소 생성하기


- 도커허브란 Github과 같이 다양한 도커이미지를 공유하기 위한 저장공간으로 도커의 공식 허브 사이트임.









The screenshot shows the Docker Hub interface. At the top is a blue navigation bar with the Docker Hub logo, links for Explore, Repositories (which is underlined), Organizations, and Usage, and a search bar labeled 'Search Docker Hub'. Below the navigation bar, there's a section for the user 'sunnykid7'. It includes a dropdown menu for the user name, a search bar labeled 'Search by repository name', and a dropdown menu for 'All Content'. To the right of these is a blue button labeled 'Create repository'. Below this section, there are three repository cards for the user 'sunnykid7':


- sunnykid7 / prelearn**: Contains: Image • Last pushed: 27 minutes ago. It has 0 stars, 163 downloads, is Private, and Scout is inactive.
- sunnykid7 / guestbook**: Contains: Image • Last pushed: 15 days ago. It has 0 stars, 18 downloads, is Public, and Scout is inactive.
- sunnykid7 / webapp**: Contains: Image • Last pushed: 17 days ago. It has 0 stars, 12 downloads, is Public, and Scout is inactive.

Gmail을 이용하여 가입한 경우 패스워드 변경하기


 **docker** EARLY ACCESS



   

 Repository settings for your personal namespace have moved. You can find these settings in Docker Hub via the Settings icon. [Go to repo settings](#) 

 **sunnykid7**
Joined July 6, 2020

General

Account information
Add your account information. 

Email
sunnykid7@gmail.com  VERIFIED 

Password

You can change your password by initiating a reset via email. [Reset password](#)

[Repositories](#) / [Create](#)

Using 1 of 1 private repositories.

Create repository

Namespace

sunnykid7

Repository Name *

testweb





Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

Using 1 of 1 private repositories. [Get more](#)

☒ **Public** 
Appears in Docker Hub search results

☐ **Private** 
Only visible to you

Cancel

Create

Pushing images

You can push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to replace `tagname` with your desired image repository tag.

CLI에서 도커허브에 로그인하고 이미지 업로드하기

- `docker login`
Username:
Password:
- `docker tag local-image:tagname new-repo:tagname`
- `docker push new-repo:tagname`