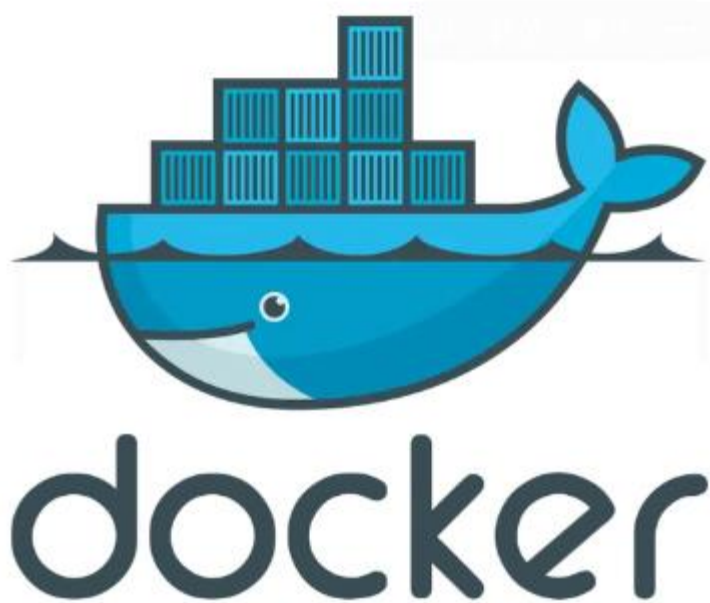


Docker

도커란?

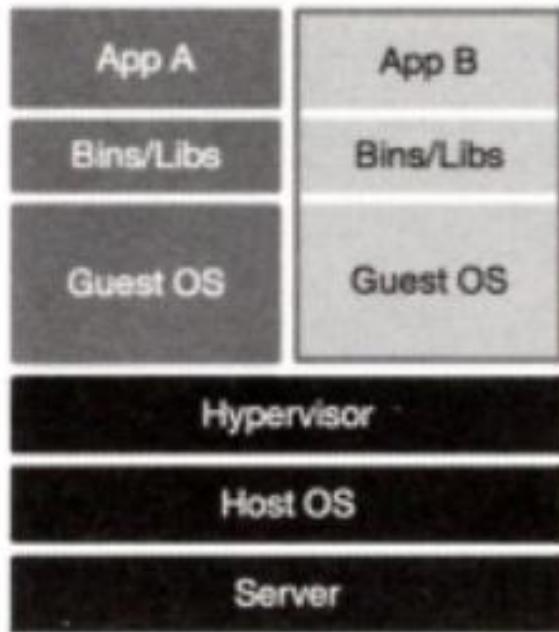


- 2013년 3월 Docker, Inc에서 출시한 오픈소스 플랫폼
- 복잡한 리눅스 애플리케이션을 컨테이너로 묶어서 실행 가능
- 고래는 서버에서 여러 개의 컨테이너를 실행하고, 이미지 저장과 배포를 의미

도커란?

- 애플리케이션을 신속하게 구축, 테스트, 배포할 수 있는 플랫폼
- 소프트웨어를 컨테이너라는 표준화된 유닛으로 패키징
- **컨테이너**: 라이브러리, 시스템 도구, 코드 등 소프트웨어 실행에 필요한 모든 것을 포함
→ 도커는 컨테이너 환경에서 독립적으로 애플리케이션을 실행 할 수 있도록 도와주는 도구

가상화



기존의 가상화 기술 : 하이퍼바이저를 이용해 여러 개의 운영체제를 하나의 호스트에서 생성해 사용하는 방식

게스트 OS: 하이퍼바이저에 의해 생성되고 관리되는 운영체제

VMware, VirtualBox같은 가상머신은 호스트 OS위에 게스트 OS전체를 가상화하여 사용하는 방식

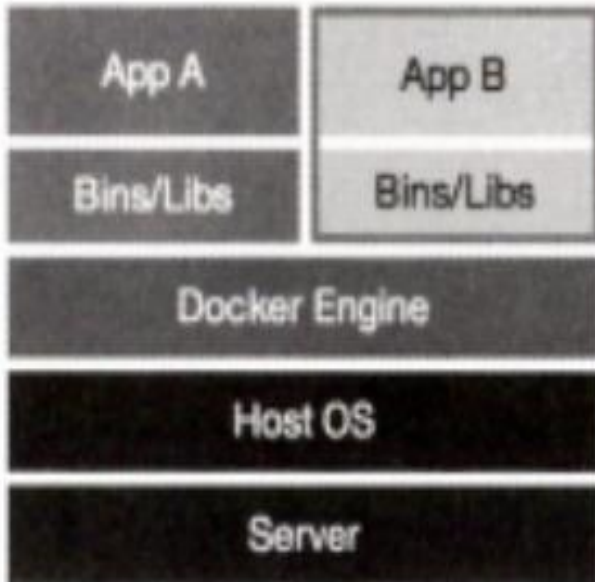
가상머신의 단점



가상화하는 작업은 하이퍼바이저를 거치기에 성능에 손실 발생

게스트 OS를 사용하기 위한 라이브러리, 커널 등을 포함하기에 가상 머신을 배포하기 위한 이미지로 만들었을 때 이미지의 용량이 커짐

도커 컨테이너



컨테이너: 격리된 공간에서 프로세스가 동작하는 기술

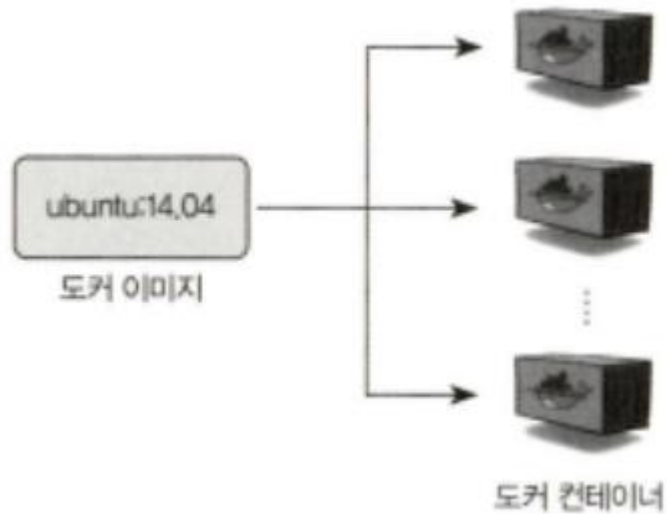
OS가상화가 아닌 프로세스를 격리하는 방식으로 작동

커널은 호스트의 커널을 공유하고, 컨테이너 안에 애플리케이션을 구동하는 데 필요한 라이브러리 및 실행 파일만 존재하기에 이미지의 용량도 작은 편

컨테이너의 특징

1. 서버에 여러 컨테이너를 실행하면 독립적으로 실행되어 가벼운 가상머신을 사용하는 느낌이 듦
2. 실행 중인 컨테이너에 접속하여 명령어를 입력할 수 있음
3. apt-get이나 yum등 운영체제에서 사용하는 패키지 매니저를 통해 설치할 수 있고, 사용자도 추가하고 프로세스를 백그라운드로 실행할 수 있음
4. CPU나 메모리 사용량을 제한할 수 있음
5. 새로운 컨테이너를 만드는데 매우 빠름

이미지



서비스 운영에 필요한 서버 프로그램, 소스코드, 컴파일된 실행 파일을 묶은 형태

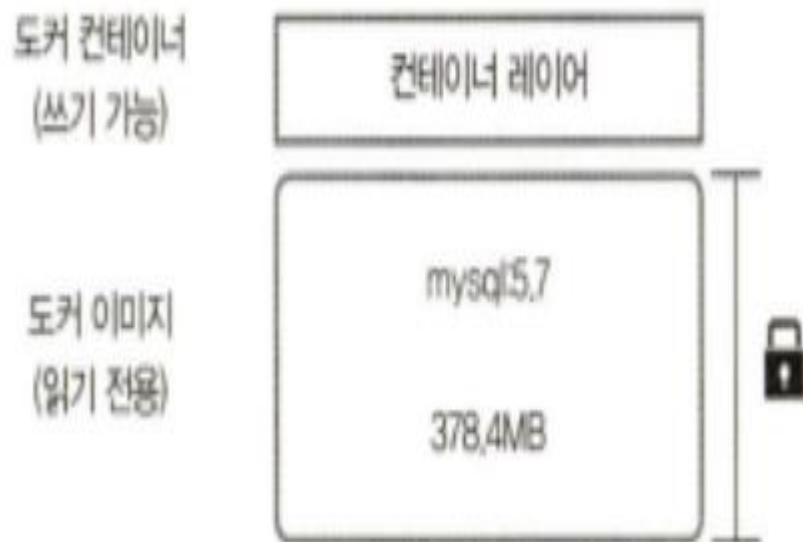
컨테이너 실행에 필요한 파일과 설정을 포함하고 있는 것으로 상태값을 가지지 않고 변하지 않음

EX) mysql 이미지: mysql을 실행하는데 필요한 파일, 실행 명령어, 포트 정보 등을 가짐

컨테이너는 이미지를 실행한 상태

-> 운영체제로 치면 이미지는 실행 파일, 컨테이너는 프로세스

이미지



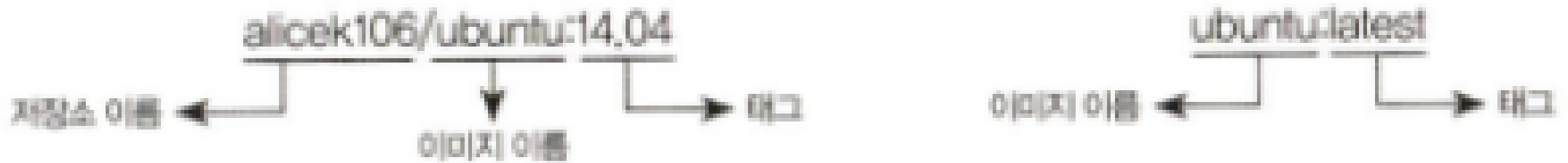
추가되거나 변하는 값은 컨테이너에 저장

같은 이미지에서 여러 개의 컨테이너를 만들 수 있고 컨테이너의 상태가 바뀌거나 삭제되어도 이미지는 변하지 않고 그대로 남아있음

이미지로 컨테이너를 생성하면 컨테이너의 변경 사항만 별도로 저장해 정보를 보존

이미 생성된 이미지는 변하지 않음

이미지 이름



저장소: 이미지가 저장된 장소, 생략 가능

이미지 이름: 해당 이미지가 어떤 역할을 하는지, 위 예시는 우분투 컨테이너 생성 이미지

태그: 이미지 버전 관리, 생략하면 태그를 latest로 인식
latest는 최신버전

이미지 특징

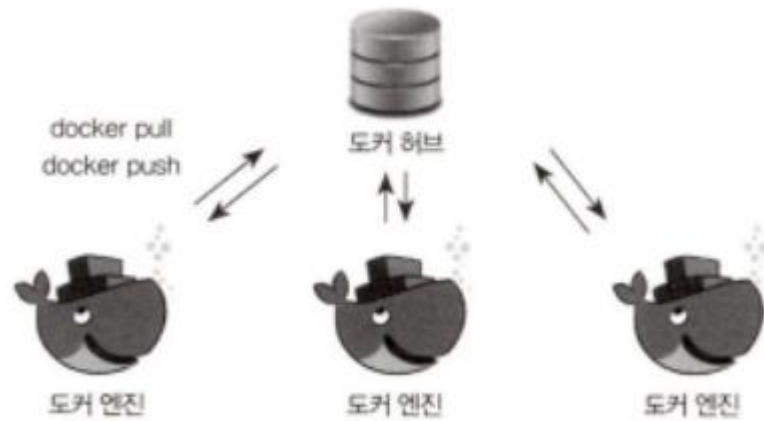
도커의 이미지는 가상머신에 비해 작은 용량

하나의 이미지를 통해 여러 컨테이너를 생성, 컨테이너를 삭제해도 이미지는 변하지 않음

이미지는 Docker hub를 통해 버전 관리 및 배포 가능

Dockerfile이라는 파일로 이미지를 만듦

Docker Hub



이미지를 무료로 관리해줌


<https://hub.docker.com/>

Docker Hub

아이디 생성

← → ↻ hub.docker.com

Missed DockerCon 2022? [Watch now on-demand.](#)

 Search for great content (e.g., mysql)

Explore Pricing Sign In **Register**

Build and Ship any Application Anywhere


Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.

Get Started Today for Free

Already have an account? [Sign In](#)


Docker ID

Email

Password 


☐ Send me occasional product updates and announcements.

☐ I agree to the [Subscription Service Agreement](#), [Privacy Policy](#) and [Data Processing Terms](#).

☐ 로봇이 아닙니다.  reCAPTCHA
개인정보 보호 · 약관

Sign Up

Docker Hub





Create a Docker ID.

Already have an account? [Sign In](#)

☒ Send me occasional product updates and announcements.


☒ I agree to the [Subscription Service Agreement](#), [Privacy Policy](#) and [Data Processing Terms](#).

 로봇이 아닙니다.


reCAPTCHA
개인정보 보호 · 약관

[Sign Up](#)


아이디 생성



Enter Your Password

[Edit](#)

Password

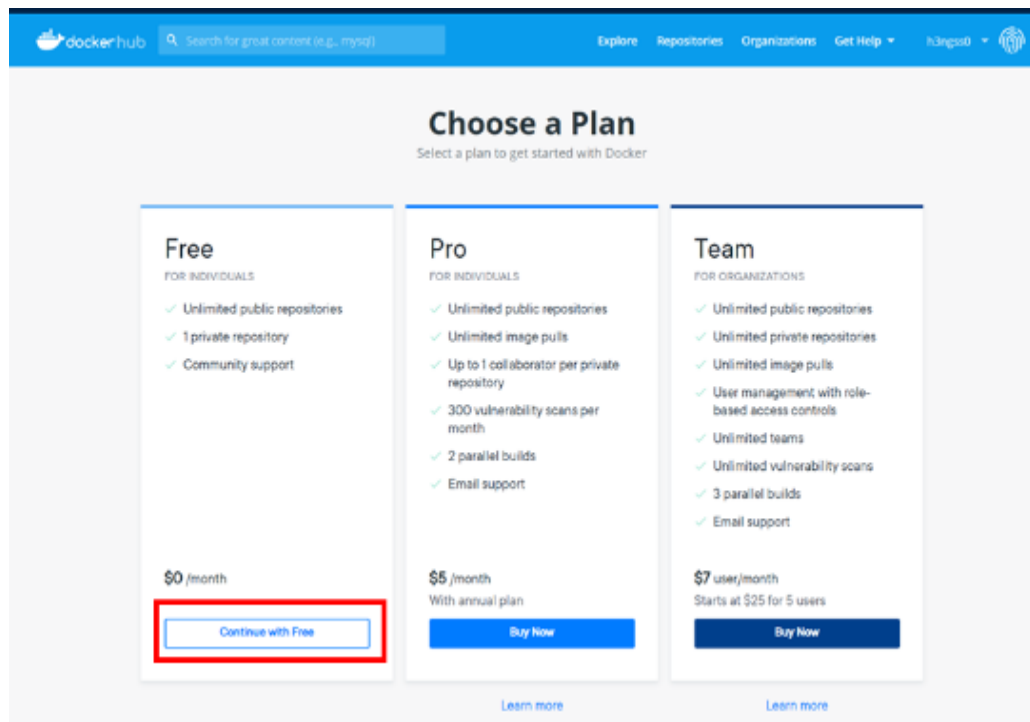


[Forgot password?](#)

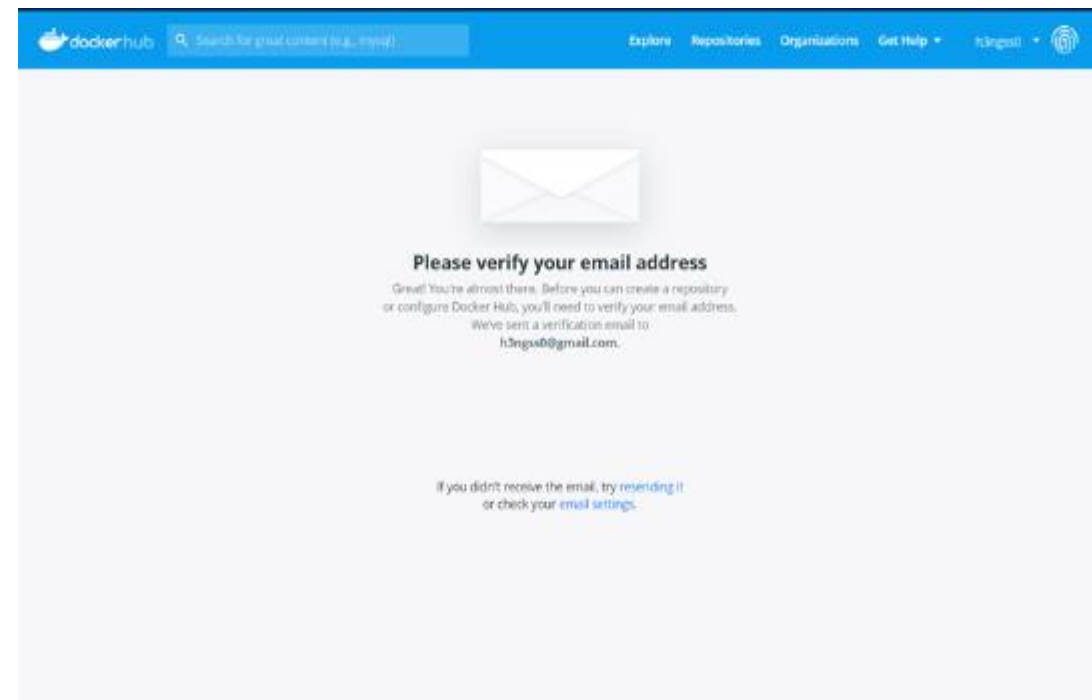
[Continue](#)

로그인

Docker Hub

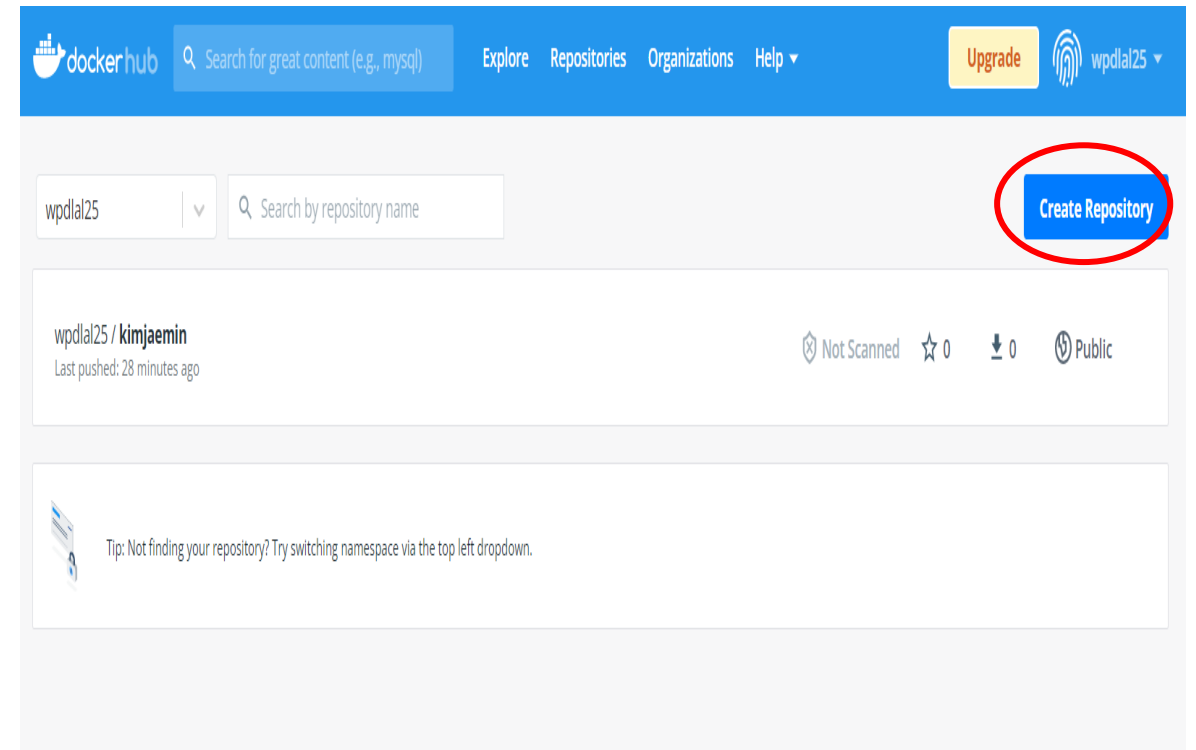
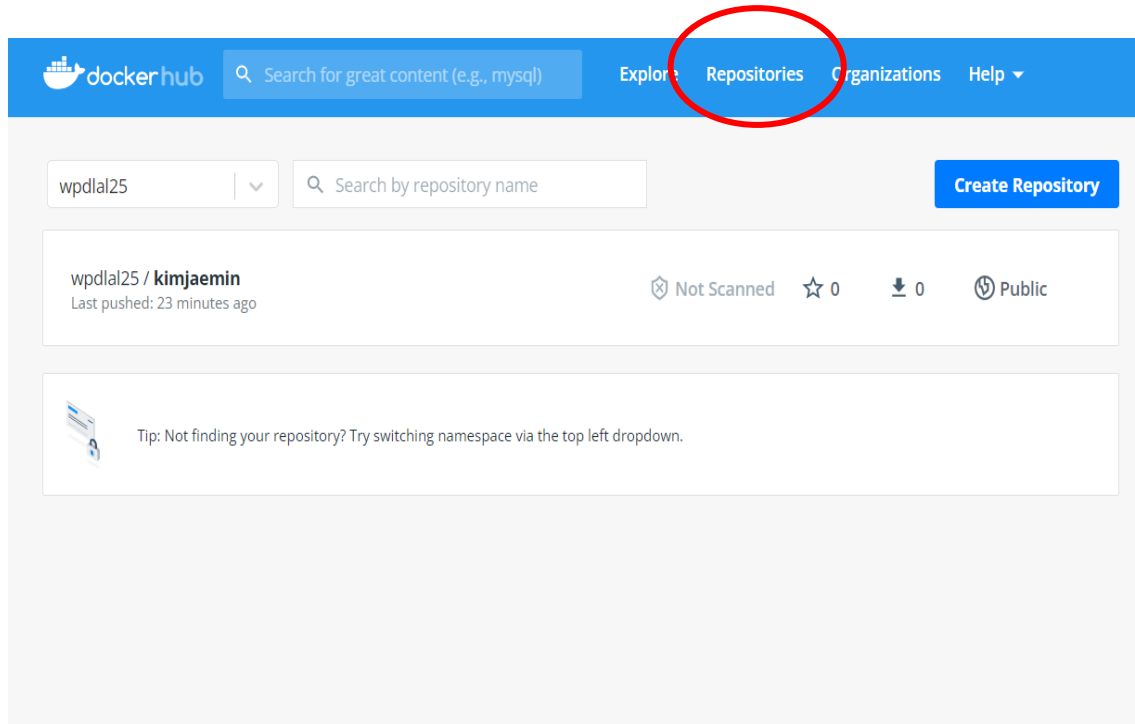


무료 플랜 선택



이메일 인증

Docker Hub



저장소 생성

Docker Hub

Repositories > Create

Create Repository

wpdlal25

123213

Description

Visibility

Using 0 of 1 private repositories. [Get more](#)

☒ Public

Appears in Docker Hub search results

☐ Private

Only visible to you

Cancel

Create

저장소 이름 잘 기억해 두세요!

wpdlal25 / kimjaemin

Description

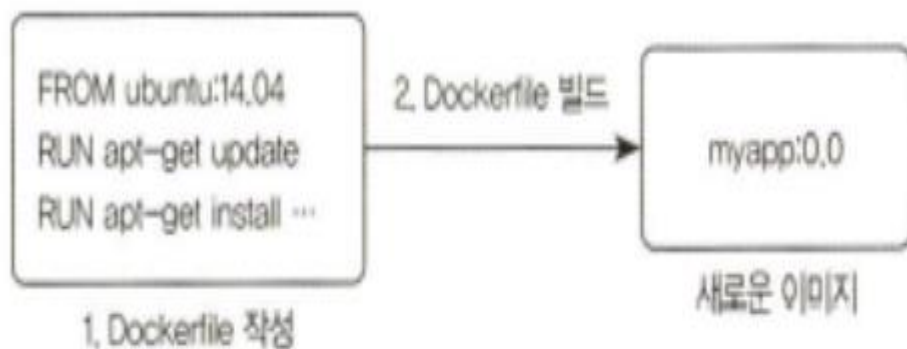
This repository does not have a description

Last pushed: 34 minutes ago

Public은 누구나 접근 가능한 저장소, private는 자신만 접근 가능
공개저장소는 무제한 생성, 사설 저장소는 1개만 생성 가능

저장소 생성 완료

Dockerfile



도커는 이미지를 만들기 위해 Dockerfile이라는 파일에 DSL(Domain Specific Language) 언어를 이용해 이미지를 생성 과정을 적음

서버에서 프로그램을 설치하려고 할 때 Dockerfile을 통해 관리

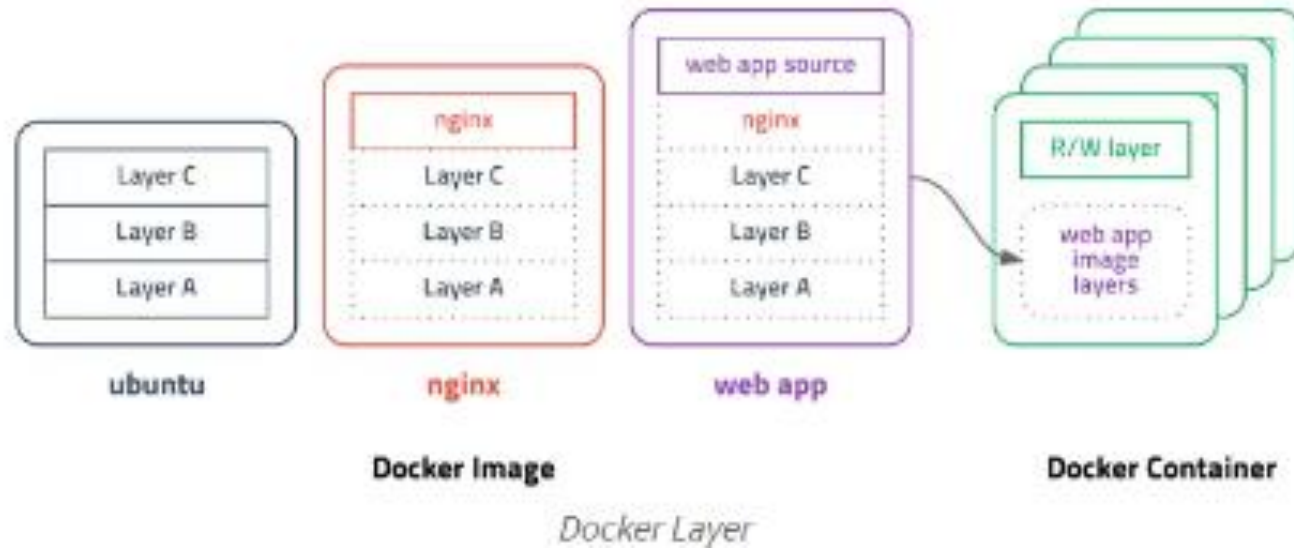
레이어 저장 방식

도커 이미지는 컨테이너를 실행하기 위한 정보를 가지고 있기에 용량이 작진 않음(수백 메가)

기존 이미지에 파일 하나 추가했다고 수백 메가를 다시 다운 받기엔 비효율
→ 레이어 사용

이미지는 여러 개의 읽기 전용 레이어로 구성, 파일이 추가, 수정 되면 새로운 레이어 생성

레이어 저장 방식



ubuntu이미지가 A + B + C라면 nginx이미지는 A + B + C + nginx, web app이미지는 A + B + C + nginx + web app source레이어로 구성

web app source를 수정하면 A, B, C, nginx를 제외한 web app source만 다운 받으면 되기에 효율적

도커 설치

1. 리눅스(일반 설치)

업데이트 및 http 패키지 설치

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl gnupg lsb-release
```

GPG키 및 저장소 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o  
/usr/share/keyrings/docker-archive-keyring.gpg
```

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]  
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee  
/etc/apt/sources.list.d/docker.list > /dev/null
```

도커 설치

1. 리눅스(일반 설치)

도커 엔진 설치

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-\$\(uname -s\)-\$\(uname -m\)" -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

도커 설치

2. 리눅스(자동 설치 스크립트)

```
curl -fsSL https://get.docker.com/ | sudo sh
```

자동 설치 스크립트보단 일반설치 추천

도커 설치

맥

<https://docs.docker.com/desktop/install/mac-install/>
사이트 들어가서 본인에게 맞는 칩셋 선택해 설치

Install Docker Desktop on Mac

Estimated reading time: 8 minutes

Update to the Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than \$10 million USD in annual revenue) now requires a paid subscription.

Welcome to Docker Desktop for Mac. This page contains information about Docker Desktop for Mac system requirements, download URLs, instructions to install and update Docker Desktop for Mac.

Download Docker Desktop for Mac

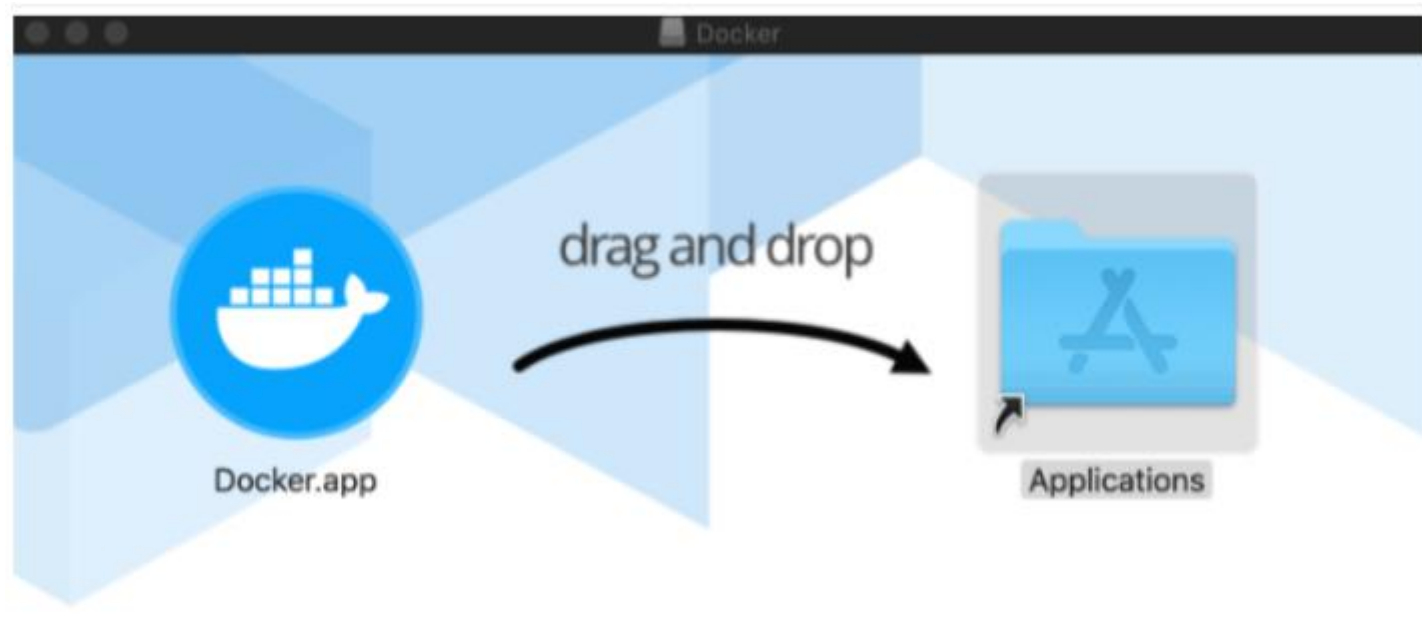
Mac with Intel chip

Mac with Apple chip

도커 설치

맥

다운 받은 파일을 클릭하면 아래와 같은 화면이 나옴
이때 Applications 폴더로 해당 app을 이동시킴



도커 설치

맥

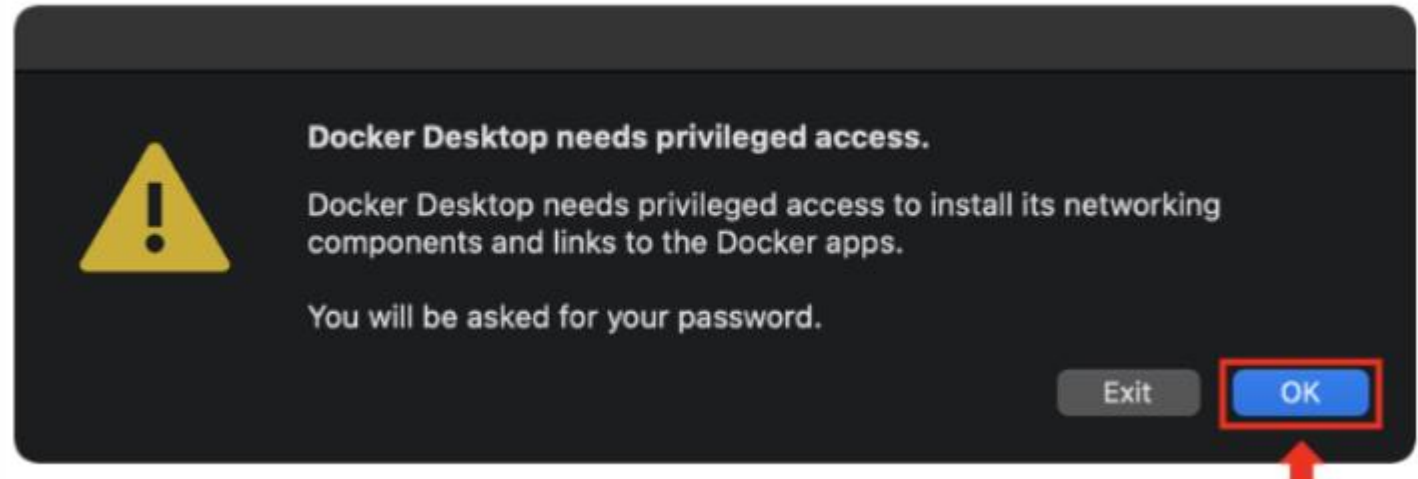
설치된 앱 실행

1

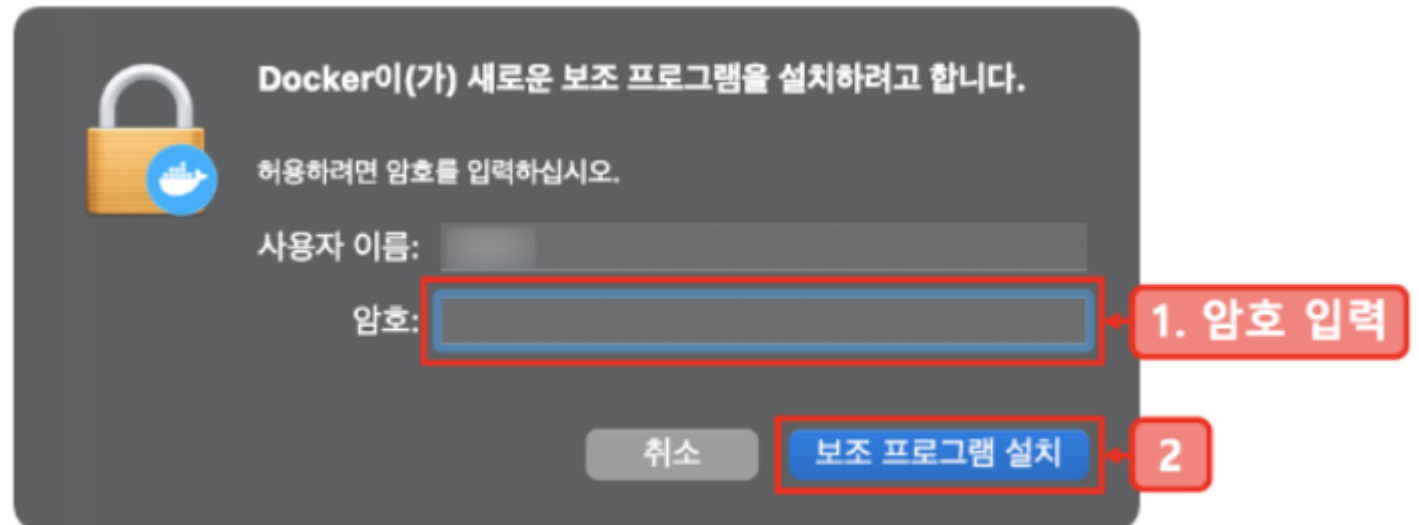


2

액세스 권한 ok



3



도커 설치

설치 버전 확인

```
sudo docker version
```

Client와 server 정보가 정상적으로 출력 되면 설치 완료

도커 명령어 실습

Docker는 기본적으로 root권한 필요

sudo 입력하지 않는 방법

1. 처음부터 root 계정으로 로그인

2. sudo su 명령을 사용하여 root 계정으로 전환

```
ubuntu@server:~$ sudo su  
root@server:/home/ubuntu#
```

3. 사용자를 docker 그룹에 추가

sudo suermod -aG docker \${USER} #현재 접속중인 사용자에게 권한주기

sudo service docker restart

도커 명령어 형식: docker + 명령어

도커 명령어 실습

search 명령으로 이미지 검색

명령어: docker search <이미지 이름>

Docker hub에서 이미지 검색 해보기

-> docker search ubuntu

```
root@server:/home/ubuntu# docker search ubuntu
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
ubuntu	Ubuntu is a Debian-based Linux operating sys...	14648	[OK]	
websphere-liberty	WebSphere Liberty multi-architecture images ...	286	[OK]	
ubuntu-upstart	DEPRECATED, as is Upstart (find other proces...	112	[OK]	
neurodebian	NeuroDebian provides neuroscience research s...	92	[OK]	
ubuntu/nginx	Nginx, a high-performance reverse proxy & we...	55		
open-liberty	Open Liberty multi-architecture images based...	53	[OK]	
ubuntu-debootstrap	DEPRECATED; use "ubuntu" instead	46	[OK]	
ubuntu/apache2	Apache, a secure & extensible open-source HT...	39		
ubuntu/mysql	MySQL open source fast, stable, multi-thread...	36		

도커 명령어 실습

pull 명령으로 이미지 받기

명령어: `docker pull <이미지 이름>:<태그>`

Docker hub에서 우분투 이미지(최신 버전)받아 보기
-> `docker search ubuntu:latest`

```
root@server:/home/ubuntu# docker pull ubuntu:latest
latest: Pulling from library/ubuntu
405f018f9d1d: Pull complete
Digest: sha256:b6b83d3c331794420340093eb706a6f152d9c1fa51b262d9bf34594887c2c7ac
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
root@server:/home/ubuntu#
```

도커 명령어 실습

images 명령으로 이미지 목록 출력하기

명령어: docker images

모든 이미지 출력
-> docker images

```
root@server:/home/ubuntu# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    27941809078c   7 weeks ago   77.8MB
root@server:/home/ubuntu#
```

도커 명령어 실습

run 명령으로 컨테이너 생성하기

명령어: `docker run <옵션> <이미지 이름> <실행할 파일>`

이미지를 컨테이너로 생성한 뒤 bash셸 실행 해보기

-> `docker run -i -t --name hello ubuntu /bin/bash`

(컨테이너 내부에 들어가기 위해 /bin/bash, -it옵션으로 키보드 입력, --name으로 컨테이너 이름 hello)

ls 명령으로 컨테이너 내부를 둘러본 뒤 exit를 입력하여 bash셸에서 빠져나오기

-> `ls`

-> `exit`

셸: 커널과 사용자 사이를 이어 명령어를 처리

도커 명령어 실습

```
root@server:/home/ubuntu# docker run -i -t --name hello ubuntu /bin/bash
root@178e5b2b6a27:/# ls
bin    dev    home  lib32  libx32  mnt    proc  run    srv    tmp    var
boot  etc    lib   lib64  media   opt    root  sbin   sys    usr
```

```
root@178e5b2b6a27:~# exit
exit
root@server:/home/ubuntu#
```

컨테이너 내부에서 나가기

나가면서 정지: 셀에 exit 입력 or Ctrl + D
정지하지 않고 나가기: Ctrl + P + Ctrl + Q

도커 명령어 실습

옵션

옵션	설명
<code>-d</code>	컨테이너를 백그라운드에서 실행 (Detached Mode)
<code>-p</code>	호스트 포트와 컨테이너 내부의 포트를 바인드한다.
<code>-v</code>	컨테이너 내부의 디렉토리를 호스트로 마운트 (연결) 한다.
<code>-e</code>	컨테이너에서 사용되는 환경변수를 설정한다.
<code>--name</code>	컨테이너의 이름을 설정한다.
<code>--rm</code>	컨테이너가 종료될 경우 컨테이너 자체를 삭제한다.
<code>-it</code>	터미널 입력을 위한 옵션. <code>-i</code> 옵션과 <code>-t</code> 옵션은 주로 함께 사용된다.
<code>-w</code>	WORKDIR 를 설정한다.

도커 명령어 실습

ps 명령으로 컨테이너 목록 확인하기

명령어: docker ps <옵션>

모든 컨테이너 목록 출력

-> docker ps -a

(-a 옵션이 없으면 정지된 컨테이너는 출력되지 않음)

```
root@server:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS          NAMES
178e5b2b6a27   ubuntu    "/bin/bash"             About a minute ago    Exited (0) 24 seconds ago          hello
root@server:/home/ubuntu#
```

```
root@server:/home/ubuntu# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS          NAMES
root@server:/home/ubuntu#
```

도커 명령어 실습

start 명령으로 컨테이너 시작하기

명령어: `docker start <컨테이너 이름 or 컨테이너 id>`

방금 정지한 컨테이너 다시 시작

-> `docker start hello`

컨테이너 목록 확인

-> `docker ps`

```
root@server:/home/ubuntu# docker start hello
hello
root@server:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
178e5b2b6a27   ubuntu   "/bin/bash"             2 minutes ago Up 7 seconds   hello
root@server:/home/ubuntu#
```

도커 명령어 실습

restart 명령으로 컨테이너 시작하기

명령어: `docker restart <컨테이너 이름 or 컨테이너 id>`

os 재부팅처럼 컨테이너를 다시 시작
-> `docker restart hello`

도커 명령어 실습

attach 명령으로 컨테이너 접속하기

명령어: `docker attach <컨테이너 이름 or 컨테이너 id>`

방금 시작한 컨테이너에 접속해보기

-> `docker attach hello`

Ctrl + P + Ctrl + Q를 사용해 컨테이너를 정지하지 않고 빠져나오기

```
root@server:/home/ubuntu# docker attach hello
root@178e5b2b6a27:/# read escape sequence
root@server:/home/ubuntu#
```

도커 명령어 실습

exec 명령으로 외부에서 컨테이너 안의 명령 실행하기

명령어: `docker exec <컨테이너 이름 or 컨테이너 id> <명령> <매개변수>`

/bin/bash를 통하지 않고 외부에서 컨테이너안의 명령 실행해 보기

-> `docker exec hello echo "Hello World"`

컨테이너가 실행 되고 있는 상태에서만 사용가능

echo: 문자열 출력 명령어

```
root@server:/home/ubuntu# docker exec hello echo "Hello World"
Hello World
root@server:/home/ubuntu#
```

도커 명령어 실습

stop 명령으로 컨테이너 정지하기

명령어: `docker stop <컨테이너 이름 or 컨테이너 id>`

실행된 컨테이너 목록 출력 -> `docker ps`

컨테이너 정지 -> `docker stop hello`

실행된 컨테이너 목록 출력 -> `docker ps`

```
root@server:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS          NAMES
178e5b2b6a27   ubuntu   "/bin/bash"             10 minutes ago  Up 6 minutes           hello
root@server:/home/ubuntu# docker stop hello
hello
root@server:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS          NAMES
root@server:/home/ubuntu#
```


도커 명령어 실습

rm 명령으로 컨테이너 삭제하기

명령어: `docker rm <컨테이너 이름 or 컨테이너 id>`

생성된 컨테이너 삭제하기-> `docker rm hello`

모든 컨테이너 목록 출력 -> `docker ps -a`

```
root@server:/home/ubuntu# docker rm hello
hello
root@server:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
root@server:/home/ubuntu#
```

도커 명령어 실습

rmi 명령으로 이미지 삭제하기

명령어: `docker rmi <이미지 이름>:<태그>`

태그를 지정하지 않으면 같은 이름의 이미지 모두 삭제

이미지삭제하기-> `docker rmi ubuntu:latest`

이미지 목록 출력하기-> `docker images`

```
root@server:/home/ubuntu# docker rmi ubuntu:latest
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:b6b83d3c331794420340093eb706a6f152d9c1fa51b262d9bf34594887c2c7ac
Deleted: sha256:27941809078cc9b2802deb2b0bb6feed6c236cde01e487f200e24653533701ee
Deleted: sha256:a790f937a6aea2600982de54a5fb995c681dd74f26968d6b74286e06839e4fb3
root@server:/home/ubuntu# docker images
REPOSITORY    TAG        IMAGE ID      CREATED      SIZE
root@server:/home/ubuntu#
```

도커 허브 실습

도커 허브 로그인 하기

-> docker login

아까 도커 허브 페이지에서 설정한 username과 password 입력하면 로그인 완료

```
root@server:/home/ubuntu# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: wpdlal25
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@server:/home/ubuntu#
```

도커 허브 실습

hello-world 이미지 받기 -> `docker pull hello-world`

이미지 목록 확인 -> `docker images`

hw라는 이름으로 컨테이너 만들기 -> `docker run --name hw hello-world`

```
root@server:/home/ubuntu# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:53f1bbee2f52c39e41682ee1d388285290c5c8a76cc92b42687eecf38e0af3f0
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
root@server:/home/ubuntu# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world    latest    feb5d9fea6a5   10 months ago  13.3kB
root@server:/home/ubuntu# docker run --name hw hello-world
```

도커 허브 실습

컨테이너 커밋하기(컨테이너 상에서 작업한 내용은 컨테이너가 삭제되면 사라지기에 작업한 내용을 이미지로 커밋하는 과정)

-> docker commit hw [wpdlal25/kimjaemin:1](#)

본인의 저장소 이름:태그
태그는 원하는 대로 작성

이미지 목록 확인 -> docker images

이미지 도커 허브 저장소에 업로드-> docker push wpdlal25/kimjaemin:1

도커 허브 확인->



The screenshot shows the Docker Hub interface for the repository `wpdlal25 / kimjaemin`. It includes a description field (currently empty), a 'Last pushed' timestamp of 'a few seconds ago', and a 'Public View' button. Below this, the 'Docker commands' section provides the command `docker push wpdlal25/kimjaemin:tagname` for pushing a new tag. The 'Tags and Scans' section indicates that the repository contains 1 tag(s) and shows a table with columns for TAG, OS, PULLED, and PUSHED. The table lists one tag with OS 'linux' and a push time of 'a few seconds ago'. At the bottom, there are links for 'See all' and 'Go to Advanced Image Management'.

wpdlal25 / kimjaemin

Description
This repository does not have a description

Last pushed: a few seconds ago

Docker commands
To push a new tag to this repository,

```
docker push wpdlal25/kimjaemin:tagname
```

Tags and Scans
This repository contains 1 tag(s).

VULNERABILITY SCANNING - DISABLED [Enable](#)

TAG	OS	PULLED	PUSHED
1	linux	---	a few seconds ago

[See all](#) [Go to Advanced Image Management](#)

도커 허브 실습

hw 컨테이너 삭제-> `root@server:/home/ubuntu# docker rm hw`

컨테이너 목록 확인-> `root@server:/home/ubuntu# docker ps -a`

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5633c0b6cc1d	wpdlal25/kimjaemin	python3	5 minutes ago	Up 5 minutes		hw

이미지 목록 확인-> `root@server:/home/ubuntu# docker images`

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
wpdlal25/kimjaemin	1	5633c0b6cc1d	5 minutes ago	13.3kB
hello-world	latest	feb5d9fea6a5	10 months ago	13.3kB

이미지id를 사용해 이미지 삭제(id는 꼭 다 쓸 필요 없음)-> `root@server:/home/ubuntu# docker rmi 5633c`

`root@server:/home/ubuntu# docker rmi feb5d9f`

이미지 목록 확인-> `root@server:/home/ubuntu# docker images`

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	feb5d9fea6a5	10 months ago	13.3kB

도커 허브 실습

저장소에서 이미지 다운 하기-> `docker pull <저장소에서 다운받을 이미지이름>`

이미지 목록 확인 -> `docker images`

이미지 삭제-> `docker rmi <이미지 id>`

```
root@server:/home/ubuntu# docker pull wpdlal25/kimjaemin:1
1: Pulling from wpdlal25/kimjaemin
2db29710123e: Pull complete
Digest: sha256:d7d3e1a7cba5a3429fefcfc29b411580f9be25027fcc7d4ba8b97daf8dd33ab5
Status: Downloaded newer image for wpdlal25/kimjaemin:1
docker.io/wpdlal25/kimjaemin:1
root@server:/home/ubuntu# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
wpdlal25/kimjaemin  1           5633c0b6cc1d     10 minutes ago  13.3kB
root@server:/home/ubuntu# docker rmi 5633c0
```

공지사항

다음주 카프카 실습때 도커 사용하니 지우시면 안됩니다!