Docker 기초

DE 19기 유임성

목차 A table of contents

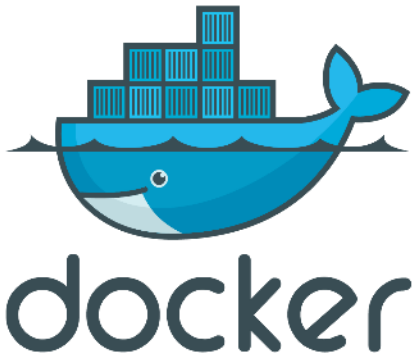
1

도커?

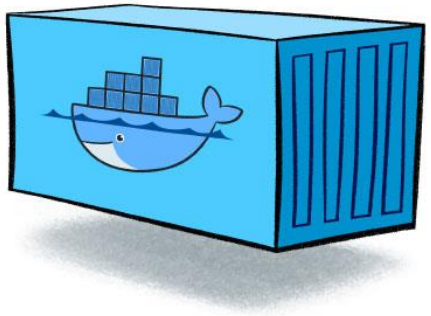
그게 뭔데
그거 어떻게 하는건데

1 도커

도커에 대해 알아봅시다



- Go 언어로 작성됨!
- 리눅스 컨테이너에 여러 기능을 추가한 오픈소스 프로젝트
- 차세대 클라우드 인프라 솔루션

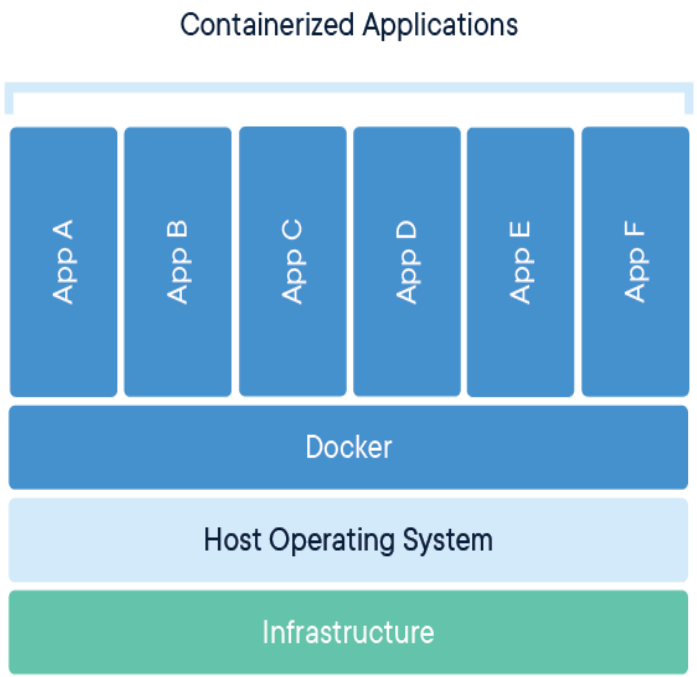


Container

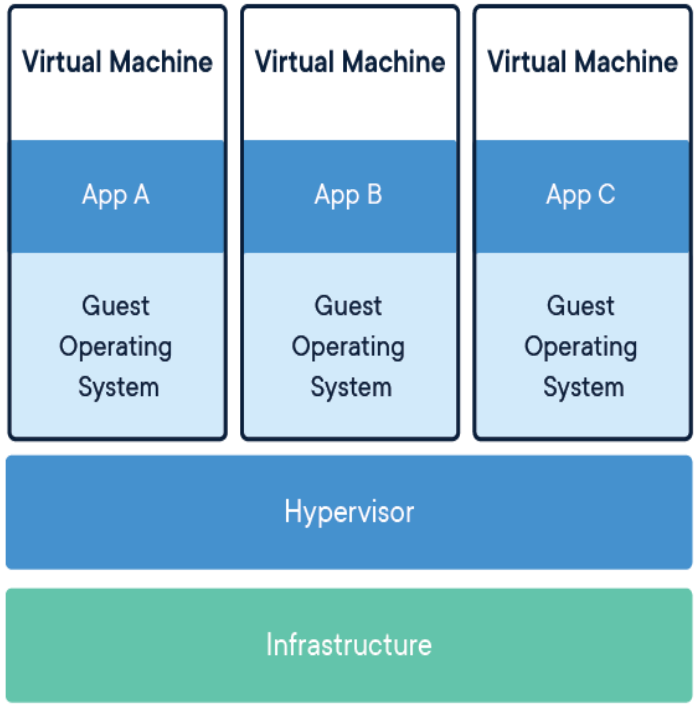
- 기존의 가상화 기술과는 다른 개념
- 규격화(정규화) 되어 있어 내용물을 이리저리 쉽게 옮길 수 있다
- 도커 엔진내에 작동하며 여러 개의 컨테이너가 존재할 수 있음

1 도커

도커 컨테이너와 가상 머신



docker



가상머신

가상머신

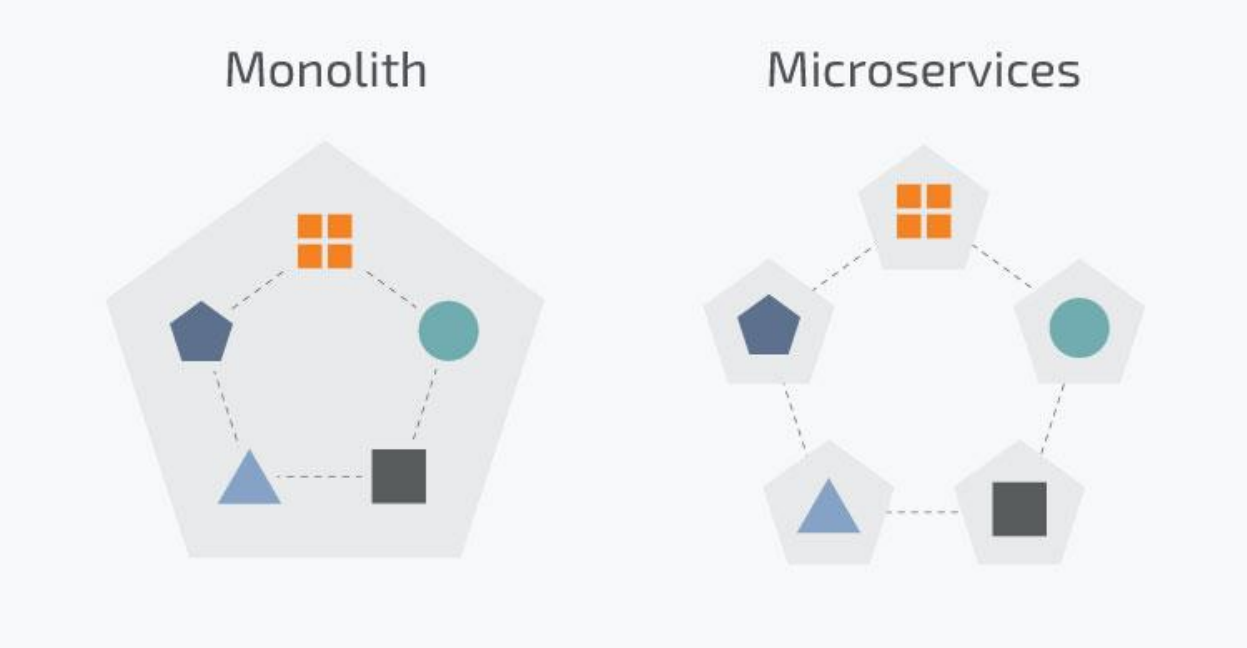
- 하이퍼바이저는 성능 손실의 원인
- 가상머신은 호스트와 하드웨어 수준의 자원을 공유
- 나머지 자원은 이미지 형태로 가지고 있어 무거움

docker

- 컨테이너는 프로세스 단위로 격리됨
- 컨테이너는 호스트와 커널, 라이브러리, 바이너리 같은 소프트웨어 수준의 자원을 공유
- 실행파일만 가지고 있어 가벼움

1 도커

도커의 장점



- 모놀리스는 여러 모듈이 한 프로그램 내에 같이 결합된 형태로 구성
- 유지보수 어려움
- 소규모 어플리케이션에서는 쓸만함

- 마이크로서비스는 여러 모듈이 독립된 형태로 구성
- 각 모듈의 관리가 쉬워진다
- 변화에 빠르게 대응할 수 있음(부하 관리)
- 이식성이 좋음
- 각 서비스들은 여러 개의 컨테이너를 가질 수 있음

- 내가 설정했던 개발환경, 소스코드들을 OS에 종속되지 않고 자유롭게 이식할 수 있다!

1 도커

도커의 기본 요소

이미지 : 컨테이너를 생성할 때 필요한 요소로써, 실행 파일과 각종 설정값을 가지고 있다.

컨테이너 : 이미지로부터 생성(복제)되며, 호스트와 분리된 존재 / 서비스에 필요한 최소단위

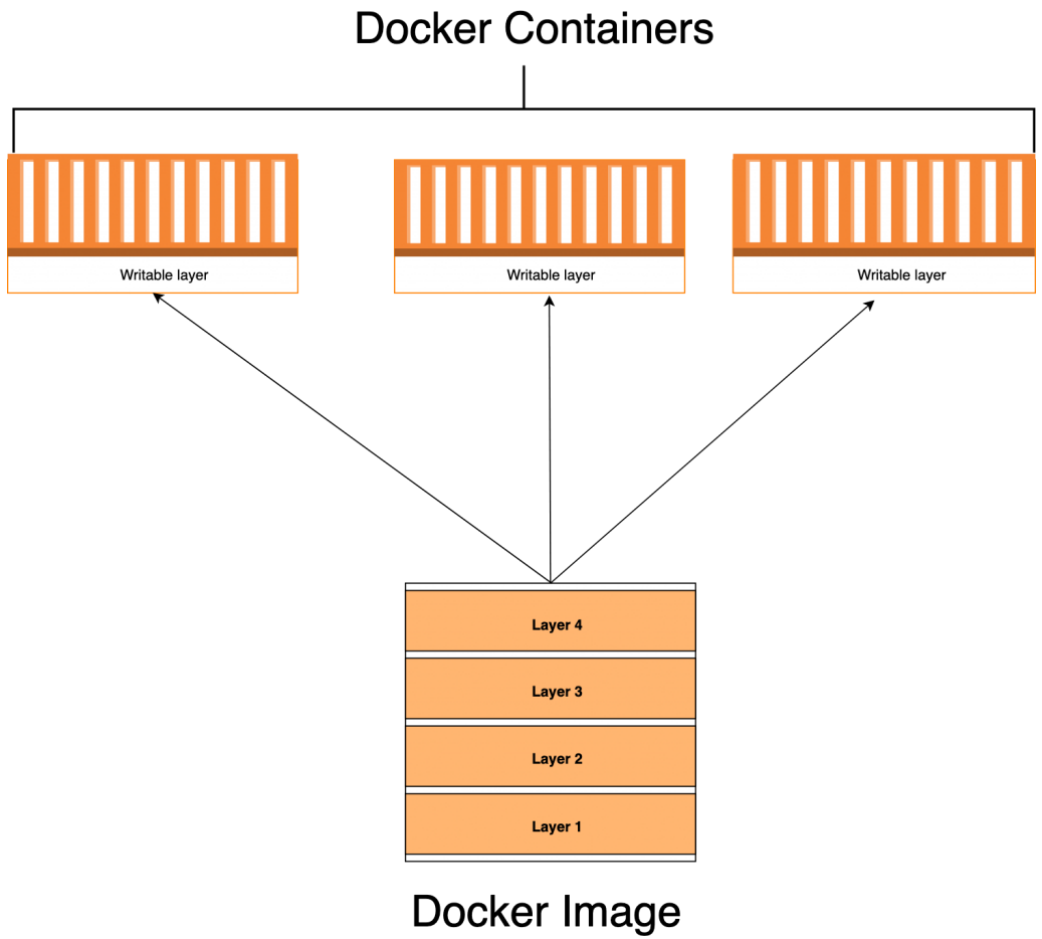
볼륨 : 컨테이너가 가지는 데이터, 관리하는 주인을 호스트나 볼륨 컨테이너, 도커 등 여러곳으로 설정할 수 있다.

네트워크 : 컨테이너가 외부와 통신할 수 있게 Bridge를 제공한다.

1 도커

이미지

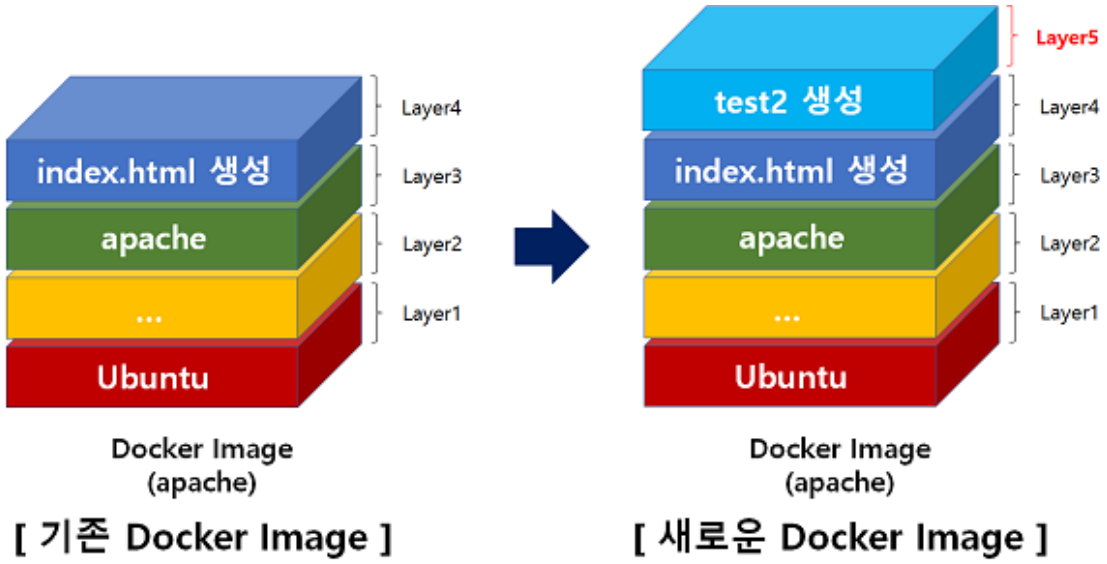
이미지 : 컨테이너를 생성할 때 필요한 요소로써, 실행 파일과 각종 설정값을 가지고 있다.

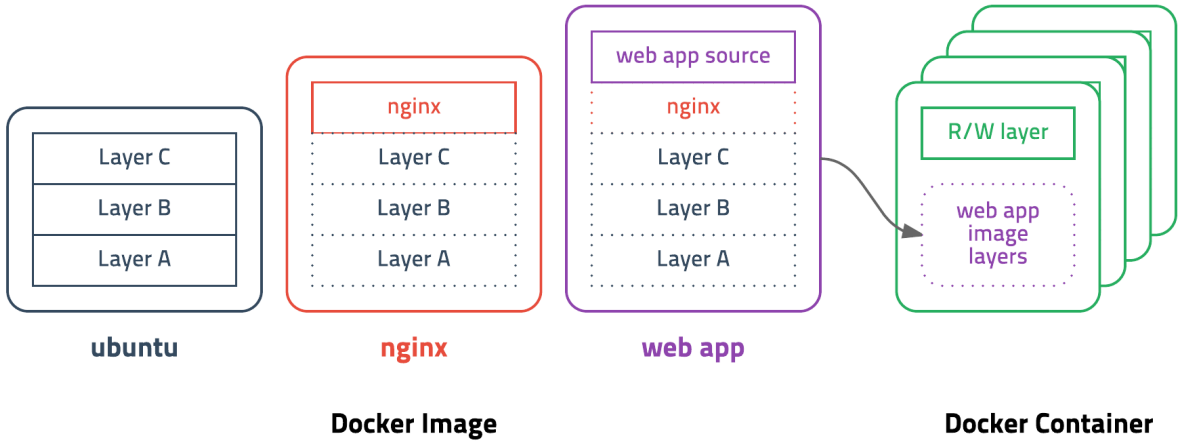


- 하나의 이미지로 여러 개의 컨테이너를 만들 수 있으며,
  컨테이너가 삭제되거나 상태가 변경되어도 이미지는 절대 변하지 않는다.
- 여러 개의 **Layer**로 구성되어있다.
- 나만의 컨테이너를 이미지화 할 수 있다.
- **Docker Hub**에서 이미지를 **push, pull** 할 수 있다.

1 도커

이미지 Layer

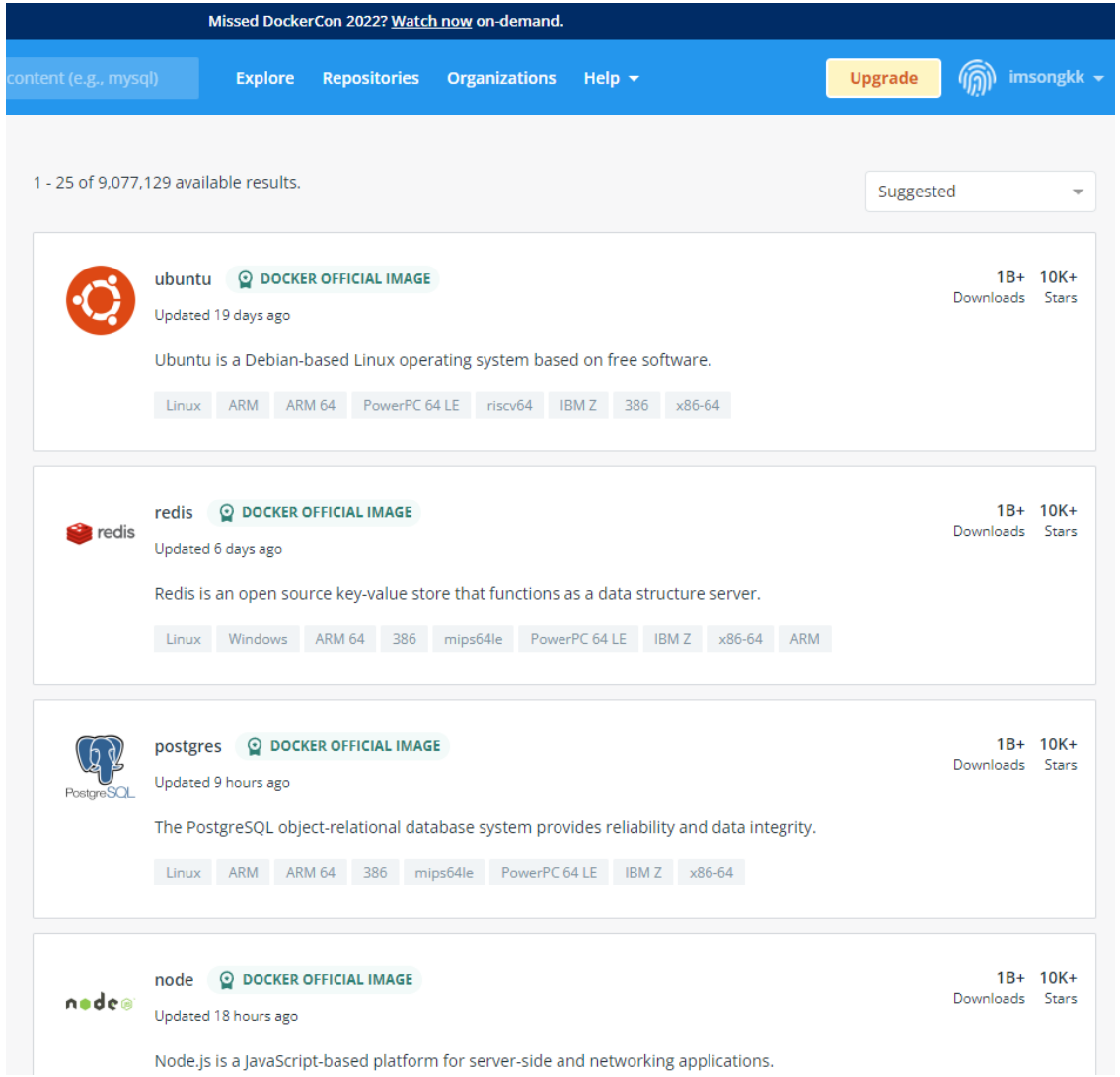


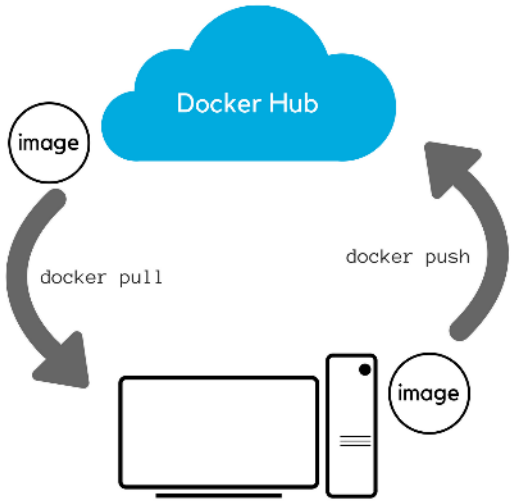


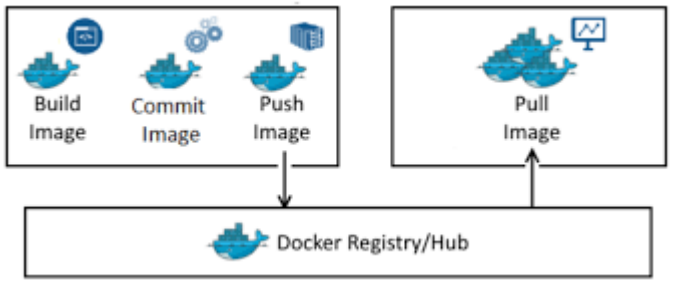
- 마치 git과 같이 동작한다!

1 도커

Docker Hub



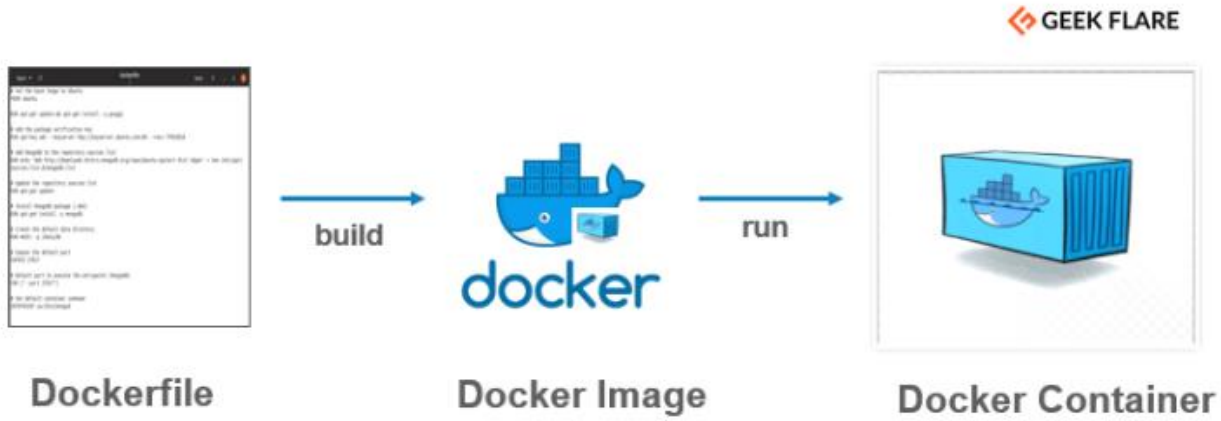




- github과 매우 유사, 도커 이미지를 **push & pull** 할 수 있다.

1 도커

Docker File



- 도커 이미지를 만들기 위한 설정 파일
- 도커 파일 내 한줄 한줄의 명령어가 이미지의 **Layer**가 된다.
- 베이스 이미지, 라이브러리 다운, 컨테이너 시작시 실행할 명령어
- 따라서 **Docker Hub**에 이미지 대신 **Docker File**을 올릴 수 있음

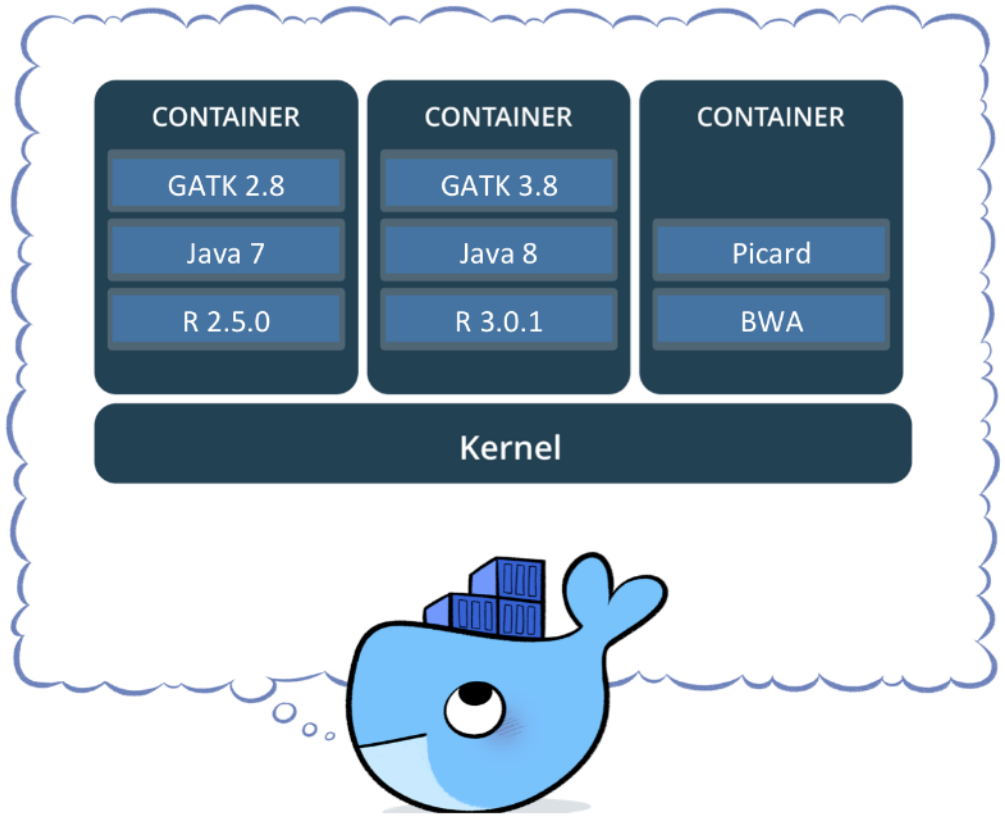
- 그냥 이미지 그대로 쓰면 안되나?

  - **Docker File**내 명령어를 보고 설치되는 라이브러리들을 명확히 볼 수 있음
  - 이미지 생성 자동화 **=>** 기존의 이미지에 여러 라이브러리를 설치후 새로운 이미지로 만드는 과정을 생각해보자
  - 만약 라이브러리 설치가 **git clone**으로 이루어진다면?

1 도커

컨테이너

컨테이너 : 이미지로부터 생성(복제)되며, 호스트와 분리된 존재 / 서비스에 필요한 최소단위



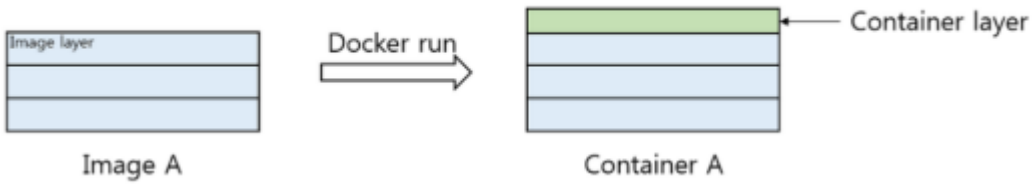
A container encapsulates
all the software dependencies
associated with running a program

Takes the guesswork out of running
pipelines on different platforms!

1 도커

볼륨

볼륨 : 컨테이너가 가지는 데이터, 관리하는 주인을 호스트나 볼륨 컨테이너, 도커 등 여러곳으로 설정할 수 있다.



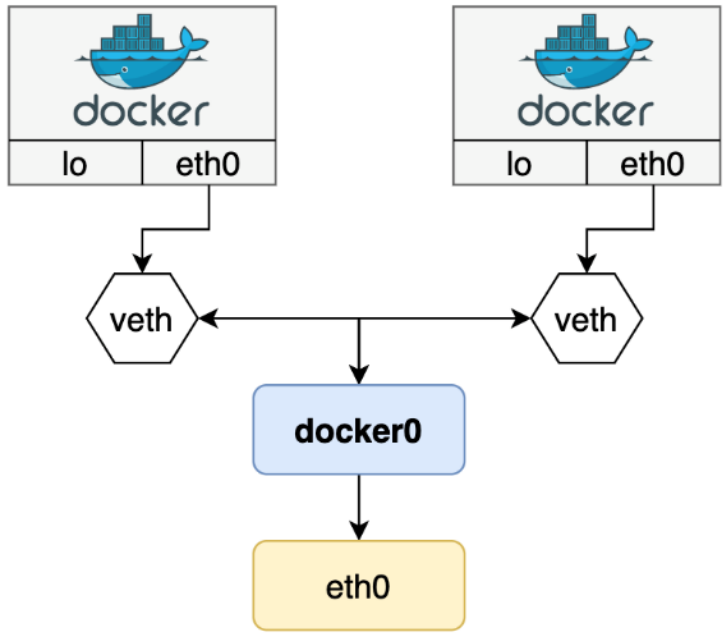
- Image Layer는 Read-Only
- Container Layer는 W/R 가능
- 이미지로부터 생성된 컨테이너가 변경한 데이터들을 관리
- 볼륨을 사용하지 않으면 컨테이너가 삭제되었을 때 데이터들도 같이 삭제된다.
- 컨테이너끼리 볼륨을 공유할 수도 있다.

- 볼륨을 활용하는 방법 3가지
  - 호스트와 볼륨을 공유한다
  - 볼륨 전용 컨테이너를 만든다
  - 도커 자체에서 볼륨을 관리한다

1 도커

네트워크

네트워크 : 컨테이너가 외부와 통신할 수 있게 **Bridge**를 제공한다.



- 내부 컨테이너끼리도 통신 가능
- 도커는 **docker0**라는 브릿지를 기본적으로 제공한다.
- 외부와의 통신은 포트를 이용한다

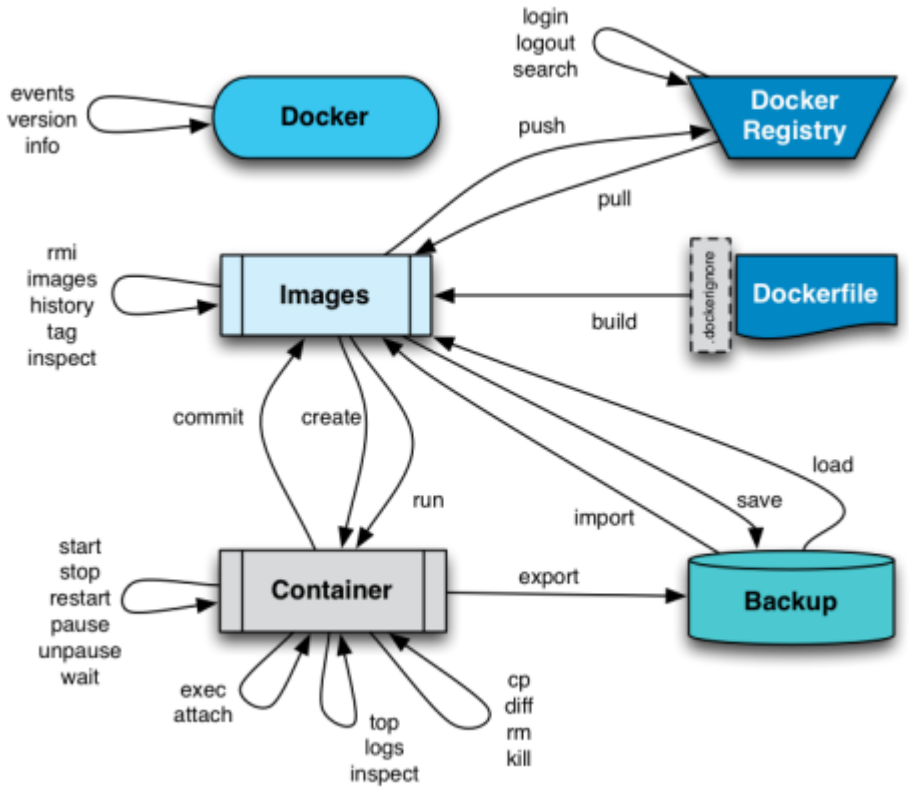
2

도커 커맨드

도커를 동작하게 만드는 명령어

2 도커 커맨드

도커 커맨드



docker images	: 로컬에 존재하는 이미지 리스트 출력
docker pull [maintainer]/[image](:[tag])	: 도커 이미지를 remot에서 pull
docker push [maintainer]/[image](:[tag])	: 도커 이미지를 remot로 push
docker run [options] ...	: 도커 이미지를 컨테이너로 실행
docker ps	: 현재 Running 상태인 컨테이너 목록 보기
docker stop [container_name]	: 현재 Running 상태인 컨테이너 중지
docker kill [container_name]	: 현재 Running 상태인 컨테이너 즉시 중지
docker start [container_name]	: 현재 Stopped 상태인 컨테이너 실행
docker attach [container_name]	: 실행중인 컨테이너 프로세스에 접속
docker detach	: 실행중인 컨테이너 프로세스에서 나오기
docker rm [container_name]	: 컨테이너 삭제
docker commit [options] ...	: 컨테이너를 이미지로 변환하여 저장
docker build [options] ...	: Dockerfile을 이용하여 이미지로 빌드하기

https://docs.docker.com/engine/reference/commandline/docker/

3

실습

간단한 과제

3 실습

실습 준비

도커 설치

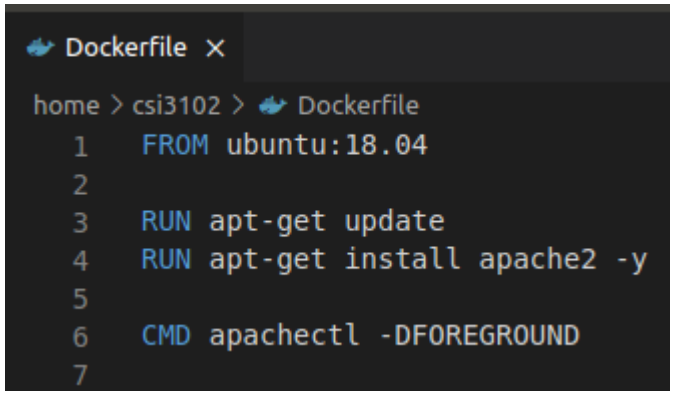
- windows : Install Docker Desktop on Windows | Docker Documentation
- mac : Install Docker Desktop on Mac | Docker Documentation
- linux : Install Docker Engine on Ubuntu | Docker Documentation

docker file

- FROM : 베이스 이미지 명시

- RUN : 도커 이미지가 생성되기 전에 수행할 쉘 명령어

- CMD : 컨테이너가 시작되었을 때 수행할 쉘 명령어



3 실습

이미지 빌드 및 배포해보기

과제 요약

- docker hub에 가입하기
- 도커 이미지 빌드해보기
  - Base 이미지 설정 후 1개 이상의 라이브러리 자유롭게 설치
    - ex) Hadoop, Spark, Python, MySql, Apache, ...
  - docker file, commit등 원하는 방법으로 빌드
  - 빌드 및 push과정을 스크린샷으로 제출해주세요

1

```
csi3102@csi3102:~$ sudo docker build -t mybuild:0.0 ./
[sudo] password for csi3102:
Sending build context to Docker daemon  473.3MB
Step 1/4 : FROM ubuntu:18.04
18.04: Pulling from library/ubuntu
40dd5be53814: Pull complete
```

```
Successfully built 237853ca2959
Successfully tagged mybuild:0.0
```

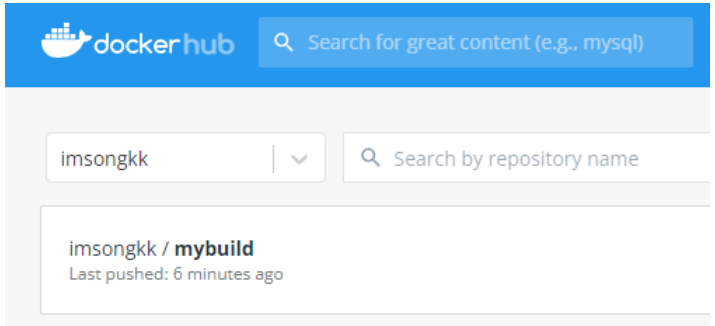
2

```
csi3102@csi3102:~$ sudo docker images
[sudo] password for csi3102:
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
mybuild      0.0       237853ca2959   4 minutes ago    200MB
```

3

```
csi3102@csi3102:~$ sudo docker push imsongkk/mybuild:0.0
The push refers to repository [docker.io/imsongkk/mybuild]
5e11fb4e784b: Pushed
602f1658c882: Pushed
3e549931e024: Pushed
0.0: digest: sha256:59c94808c35ca6f9ec1fcf901533ae6b659b4ba5724c34a5c556c
26afb265f94 size: 953
```

4



감사합니다