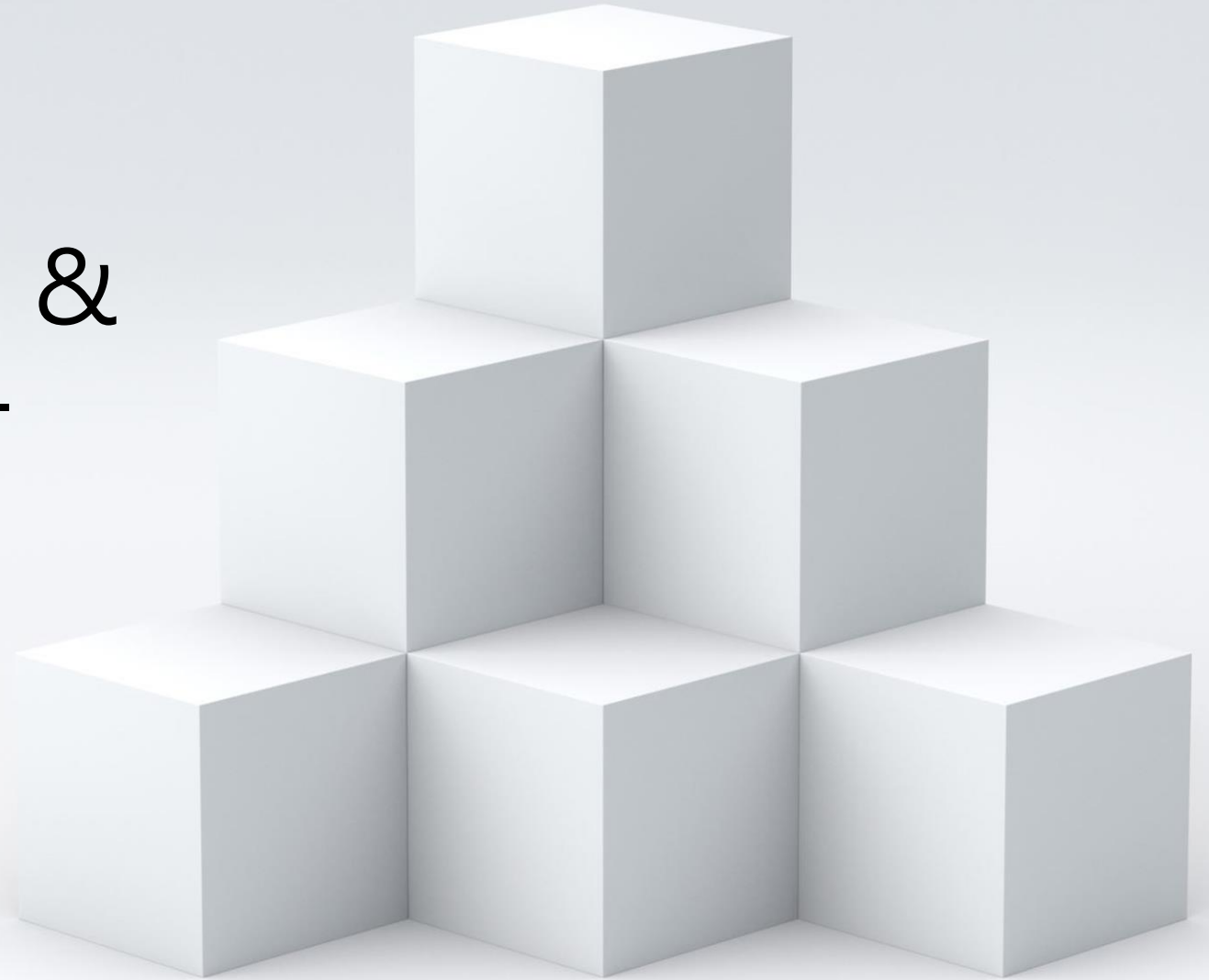


# Software Engineering

- docker-compose & docker volume -

Professor Han-gyoo Kim

2022



# Install docker-compose on Ubuntu

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

```
sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-
`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

설치 확인은

```
docker-compose --version
```

- Test 해 보기

```
mkdir test; cd test; vim docker-compose.yml
```

version: '3.9' // compose file version (docker version 20.10.7은 3.9로 보임, 하위 버전 사용 OK)

services: // indentation에 매우 주의 필요 – version 3이상부터는 **4**개의 space

```
hello-world:
```

```
  image: hello-world:latest
```

```
docker-compose up -d
```

```
docker ps
```

- 연습하기 - <https://docs.docker.com/compose/gettingstarted/>

# Dockerhub에서 제공하는 예제들

- <https://docs.docker.com/samples/>
- <https://docs.docker.com/samples/wordpress/>
- Wordpress 와 mySQL DB container를 각각 만들어 연동하는 예제
- EC2 기계에 실습을 위한 디렉토리 만들고 그 안에서  
docker-compose.yml (또는 .yaml) 파일을 만듦 (다음 페이지)
- EC2 기계의 inbound rule을 변경 (예에서는 TCP 8000포트)
- Docker-compose up -d
- 웹 브라우저에 url <http://localhost:8000> 입력

version: "3.9"

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: somewordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

volumes:

- wordpress\_data:/var/www/html

ports:

- "8000:80" // port forwarding, EC2 기계 inbound rule에 반영해야 함

restart: always

environment:

WORDPRESS\_DB\_HOST: db

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

WORDPRESS\_DB\_NAME: wordpress

volumes:

db\_data: {}

wordpress\_data: {}

version: "3.9"

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql // **named volume**

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: somewordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

volumes:

- wordpress\_data:/var/www/html // **named volume**

ports:

- "8000:80"

restart: always

environment:

WORDPRESS\_DB\_HOST: db

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

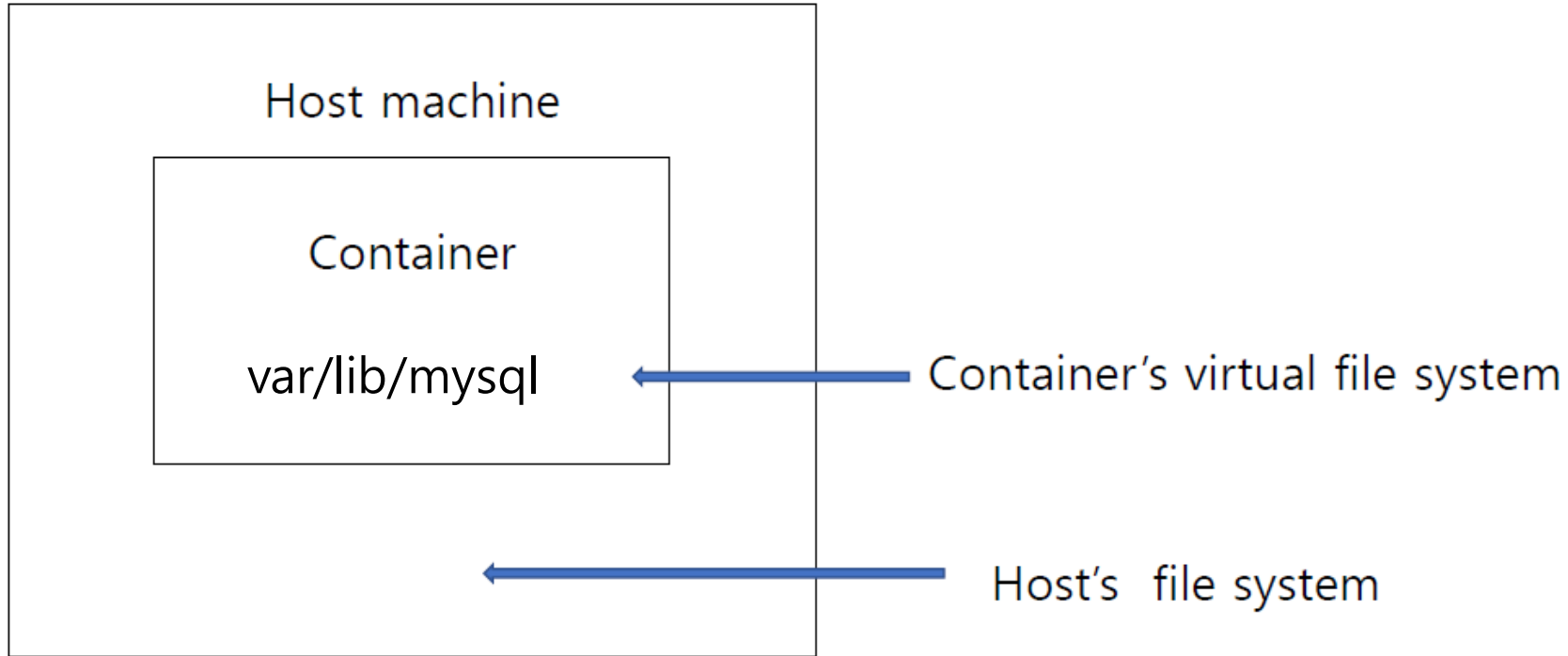
WORDPRESS\_DB\_NAME: wordpress

volumes: // **named volumes**

db\_data: {}

wordpress\_data: {}

# Docker volume

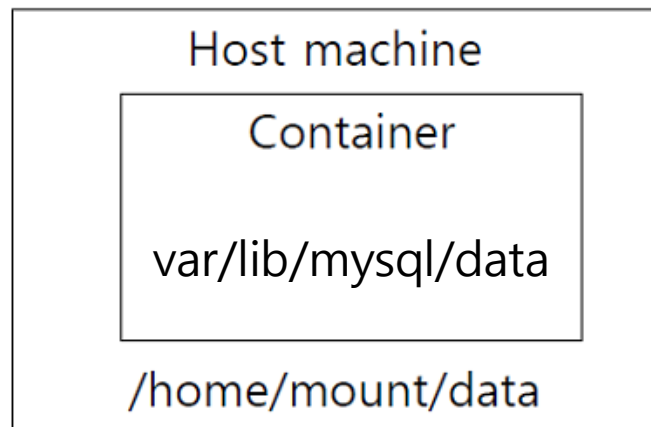


- 호스트 기계의 on-disk persistent filesystem을 컨테이너의 in-memory virtual filesystem에 mount하여 컨테이너가 종료되어도 데이터가 사라지지 않고 호스트 기계의 파일시스템에 저장되어 다시 동일 컨테이너가 시작되어 과거 데이터를 사용할 수 있도록 제공된 기능
- DB를 사용하는 컨테이너의 경우에는 필수적

# Three types of docker volumes

## (1) host volume

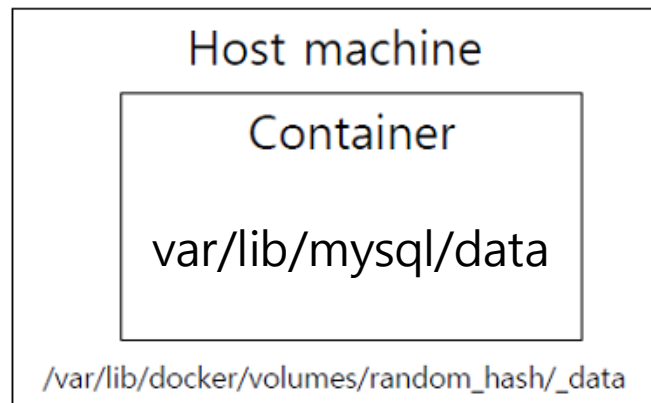
`docker run -v /home/mount/data:/var/lib/mysql/data`



## (2) anonymous volume

`docker run -v /var/lib/mysql/data`

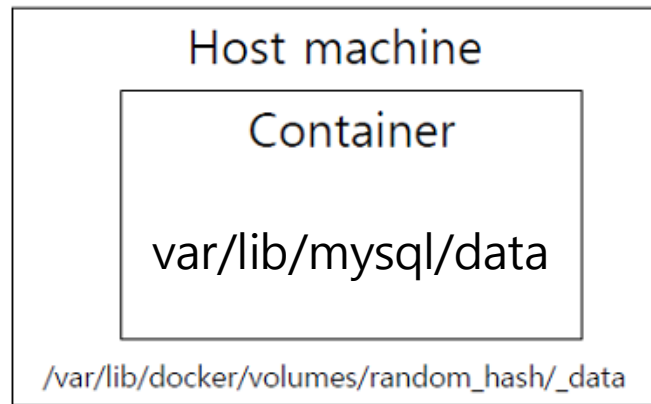
(호스트 기계의 볼륨을 지정하지 않으면 docker가 호스트 기계의 정해진 경로에 폴더를 생성. 이때 매 폴더마다 hash하여 경로 이름을 달리 함)



## (3) named volume

`docker run -v name:/var/lib/mysql/data`

(docker가 생성하게 하되 생성된 볼륨에 이름을 붙여서 여러 컨테이너가 이름을 통해 공유할 수 있도록 하며 가장 많이 사용되는 종류)



# Docker volume

- <https://docs.docker.com/storage/volumes/>
- [https://docs.docker.com/engine/reference/commandline/volume\\_create/](https://docs.docker.com/engine/reference/commandline/volume_create/)

```
docker volume create my-vol
```

```
docker volume ls
```

```
docker volume inspect my-vol
```

```
docker run -it -d -v my-vol:/world busybox /bin/sh
```

```
docker exec -it <container id> /bin/sh
```

```
// busybox 속에서
```

```
cd /world; vi 1.txt; cat /world/1.txt
```

```
// 두 번째 container 실행
```

```
docker run -it -d -v my-vol:/world2 busybox /bin/sh
```

```
docker exec -it <container id> /bin/sh
```

```
// busybox 속에서 cat /world2/1.txt로 my-volume에 저장된 것 확인
```

```
docker volume rm my-vol
```



# Run docker-compose for wordpress example

```
docker-compose up -d
```