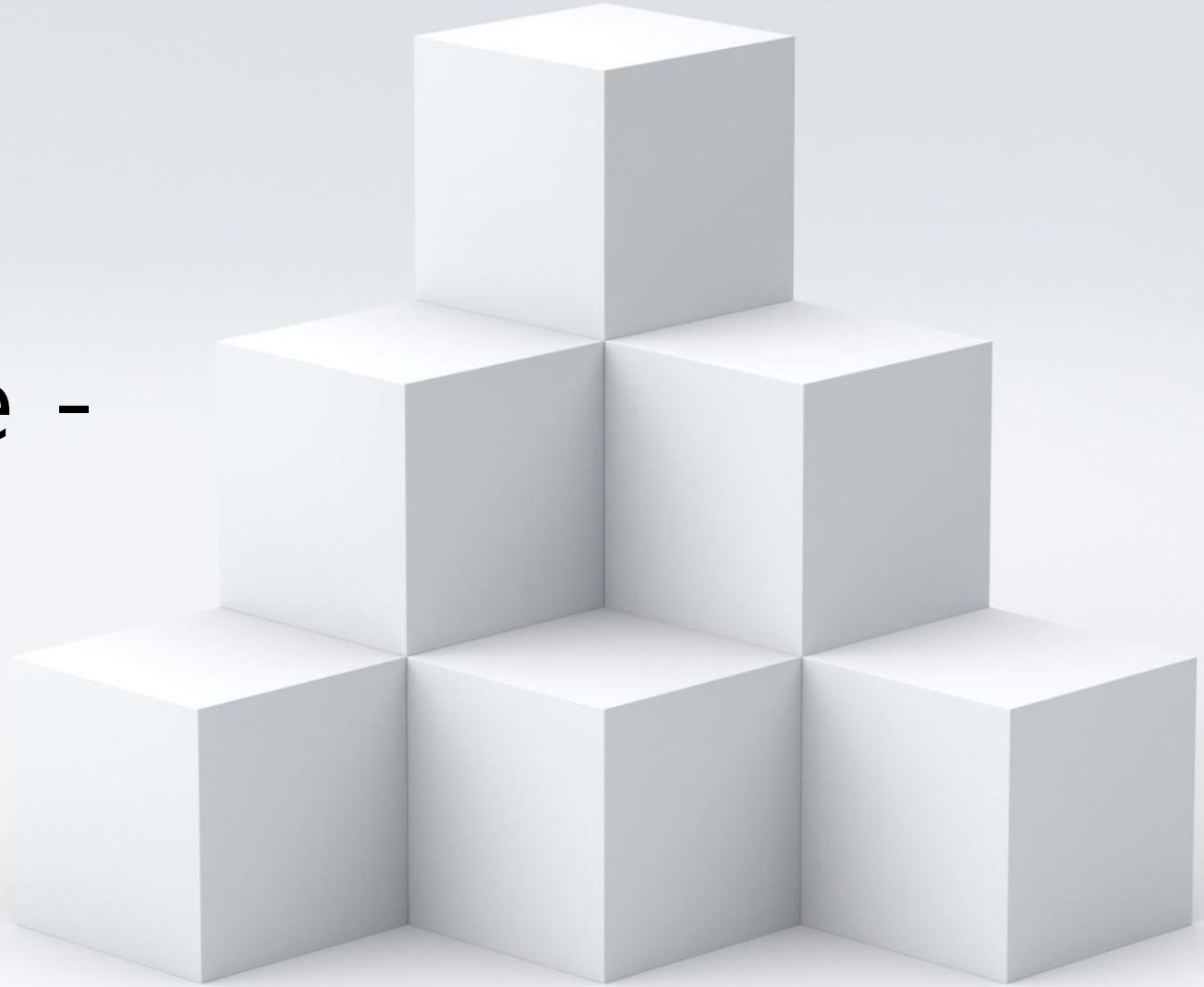
Software Engineering

- docker-compose -

Professor Han-gyoo Kim

2022



Monolithic App vs Microservices

- monolithic app 은 모든 기능이 단일 프로그램 안에 구현됨

  비현실적, 비효율, 매우 낮은 생산성

  비대한 app, 일부가 update되어도 전체 app이 re-deploy되어야 함, bug가 전체 app에 영향을 미침, 새로운 기술을 채택하기에 장벽이 있음

- microservices

  거의 모든 응용, 특히 Web 기반 App (front end + backend services including DB, login, search service, mail, etc.)

  각 단일 서비스가 서로 loosely coupled

  마이크로서비스들 사이에서의 Communication은?

  - 프로그램 속에서 arg 전달하는 대신에
  - HTTP 통신 프로토콜 등을 통한 데이터 교환 =>

    REST(ful) API (JSON)

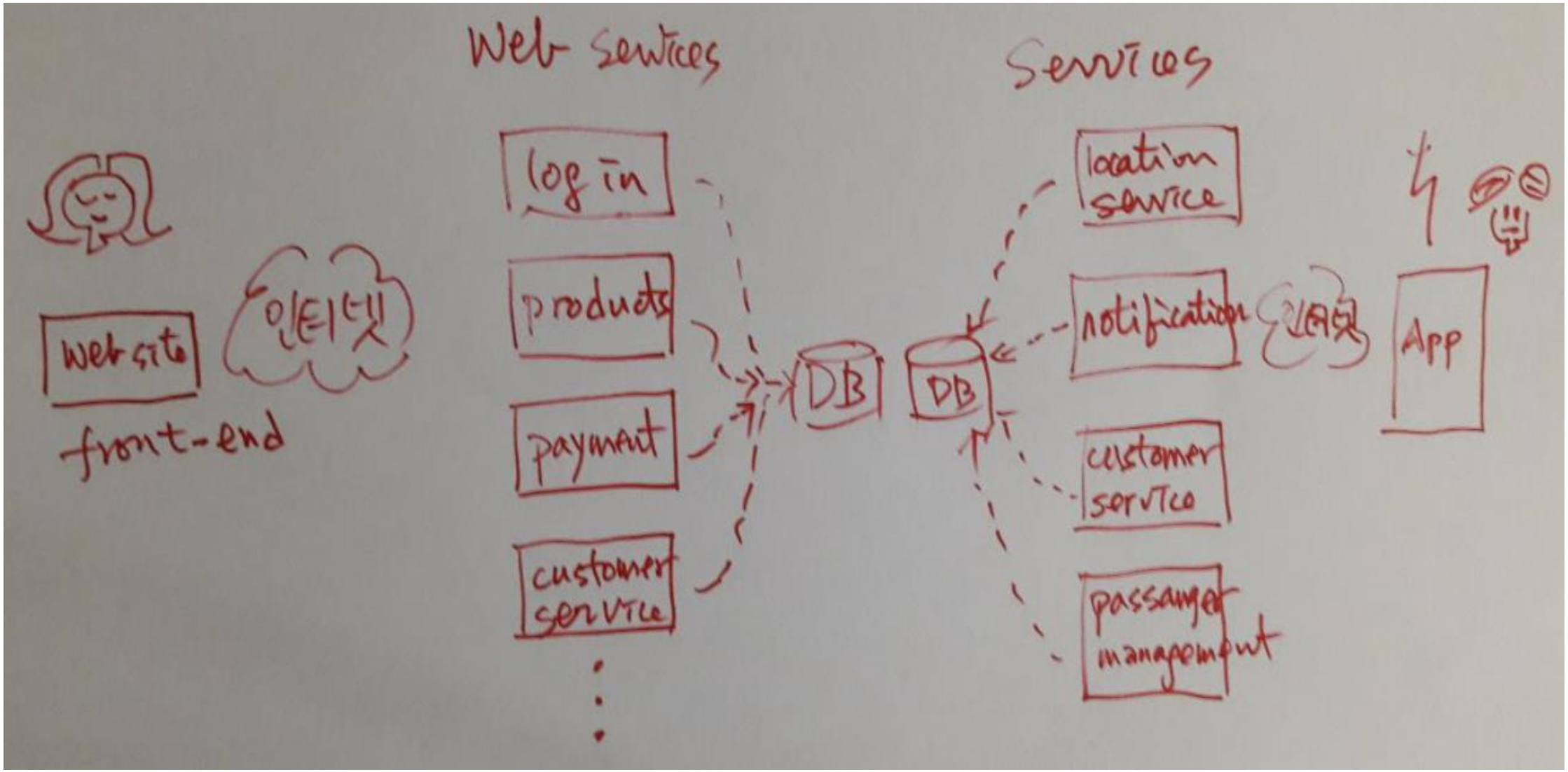
  - 각 service 들은 bug 관점에서 상호 독립적 <= 획기적, 서로 다른 기술로 구현

  Monolithic App의 단점이 모두 장점으로!!!

  Q) 성능은?

인터넷 상점

우버 서비스


Web services
Services
log in
products
paymant
customer service
DB
DB
location service
notification
customer service
passanger management
App
website
front-end

Docker-compose

- Interactive 방법 또는 dockerfile로 image를 만드는 것은 단일 컨테이너를 만드는 것
- 다수의 컨테이너를 만들고 연동시켜 동작하도록 배포하려면?

  => docker-compose 사용
- docker-compose = 다수의 docker container들을 규정하고 실행하고 관리하는 automation도구

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 4개의 space
    hello-world:
        image: hello-world:latest
```

```
docker-compose up –d
docker ps
```

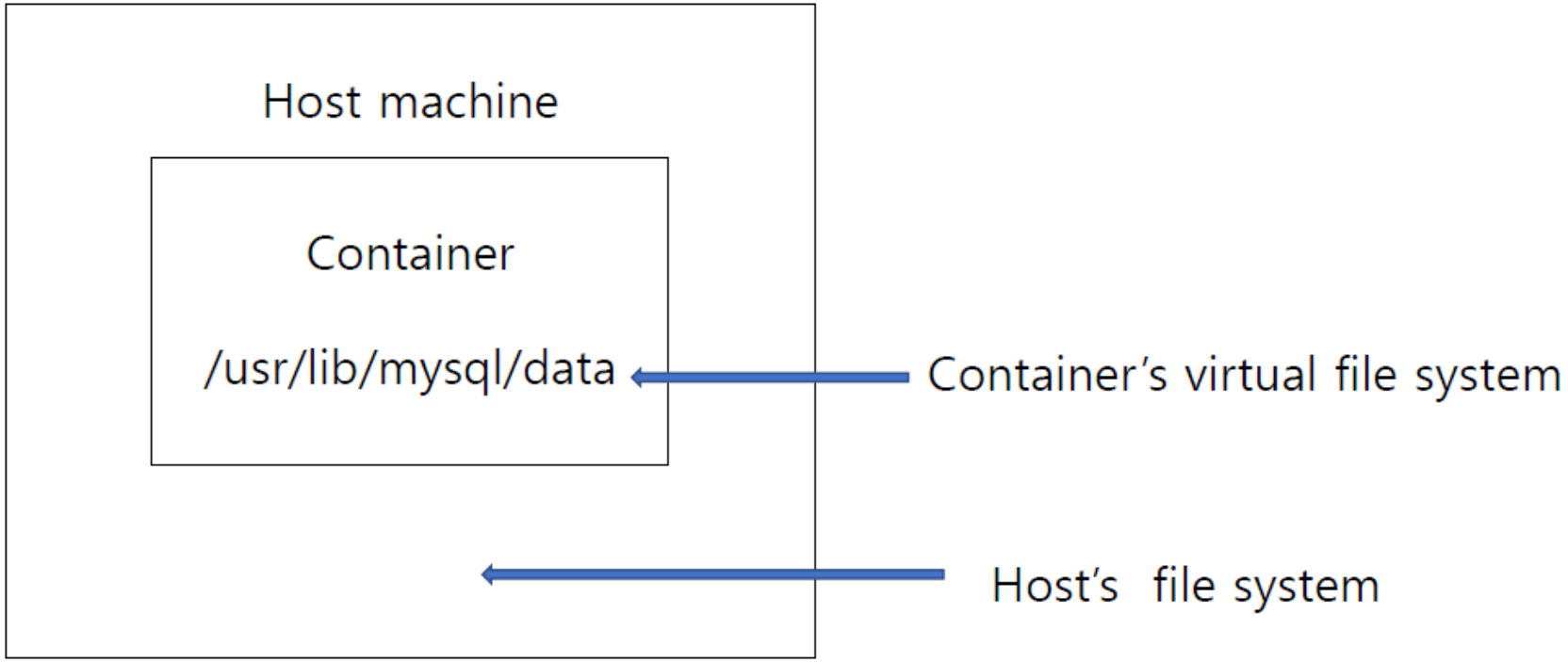
- 연습하기 - https://docs.docker.com/compose/gettingstarted/

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: {}
```

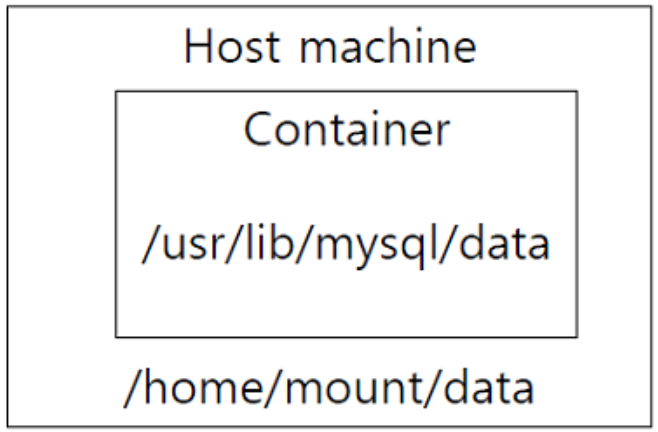
Docker volume



- 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

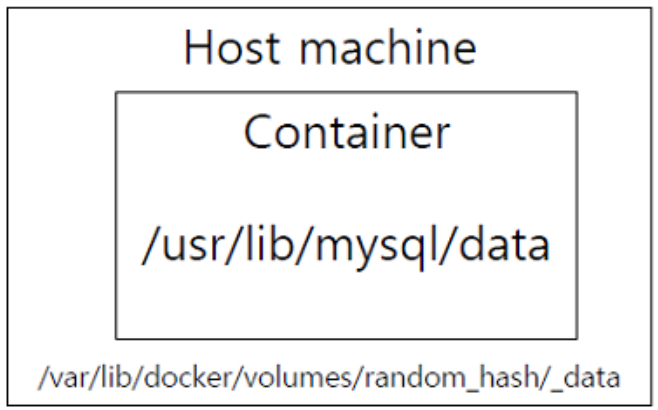
Three types of docker volumes

(1) host volume
docker run -v /home/mount/data:/var/lib/mysql/data



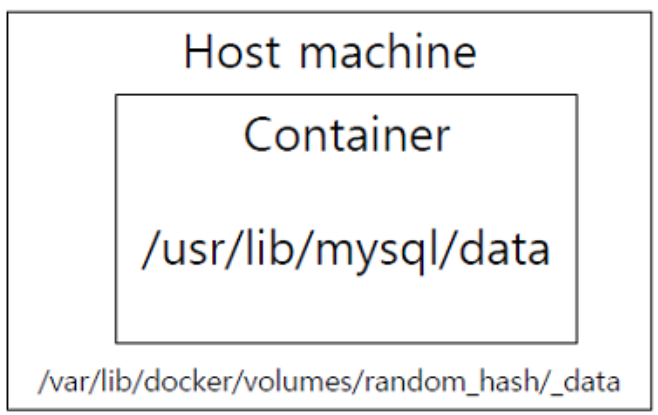
(2) anonymous volume
docker run -v /var/lib/mysql/data

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



(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)



```
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: {}
```