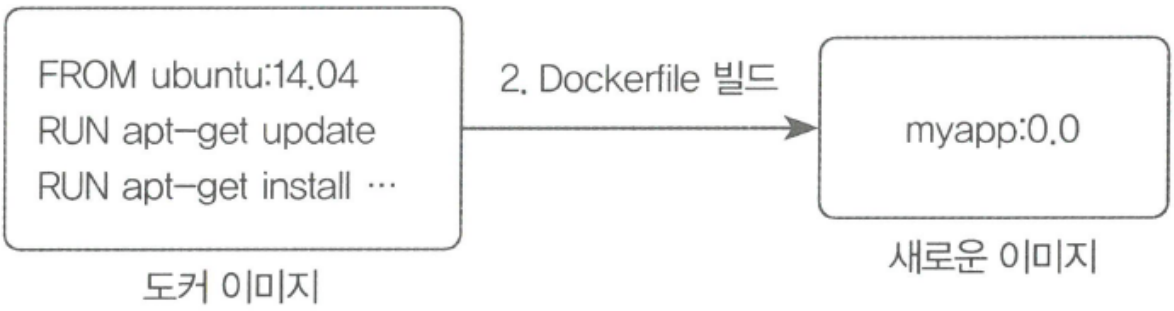
< Docker file & Docker Compose >

1. Docker file

도커 이미지를 만들 수 있는 파일이다. 도커 이미지를 생성할 수 있는 방법은 두가지가 있는데 한가지는 도커 컨테이너를 이미지로 커밋(commit)하는 것이다. 두번째 방법은 Dockerfile로 이미지를 생성하는 방법이다.



이미지를 생성하기 위해 컨테이너에 설치해야 하는 패키지, 추가해야 하는 소스코드, 실행해야 하는 명령어와 셸 스크립트 등을 하나의 파일에 기록해 두면 도커는 이 파일을 읽어 컨테이너에서 작업을 수행한 후 이미지로 만들어 낸다.

이러한 작업을 기록한 파일의 이름을 Dockerfile이라고 한다. 빌드 명령어는 Dockerfile을 읽어 이미지를 생성한다. 도커파일을 사용하면 직접 컨테이너를 생성하고 이미지로 커밋해야 하는 번거로움을 덜 수 있을 뿐 더러 깃과 같은 개발도구를 통해 애플리케이션의 빌드 및 배포를 자동화할 수 있다.

2. Dockerfile 작성

```
# server image는 ubunutu 18.04를 사용
FROM ubuntu:18.04
# Dockerfile 작성자
MAINTAINER Suejin

# image가 올라갔을 때 수행되는 명령어들
# -y 옵션을 넣어서 무조건 설치가 가능하도록 한다.
RUN \
    apt-get update && \
    apt-get install -y apache2
ADD test.html /var/www/html
WORKDIR /var/www/html

# apache가 기본적으로 80포트를 사용하기 때문에 expose를 이용해 apache server로 접근이 가능하도록 한다.
EXPOSE 80

# 컨테이너가 생성된 이후에 내부의 아파치 서버는 항상 실행중인 상태로 만들어준다.
# apachectl를 foreground(즉, deamon)상태로 돌아가도록 한다.
CMD ["apachectl", "-D", "FOREGROUND"]
```

- FROM: 생성할 이미지의 베이스가 될 이미지이다.
- RUN: 이미지를 만들기 위해 컨테이너 내부에서 명령어를 실행한다. 위 예제에서는 아파치 웹 서버가 설치된 이미지가 생성된다.
- ADD: 파일을 이미지에 추가한다. 추가하는 파일은 Dockerfile이 위치한 디렉터리인 컨텍스트에서 가져온다.
- WORKDIR: 명령어를 실행할 디렉터리이다.
- EXPOSE: Dockerfile의 빌드로 생성된 이미지에서 노출할 포트를 설정한다. 호스트의 포트와 바인딩 되는 것이 아니고 단지 컨테이너의 80번 포트를 사용할 것을 나타내는 것이다.
- CMD: 컨테이너가 시작될 때마다 실행할 명령어(커맨드)를 설정한다.

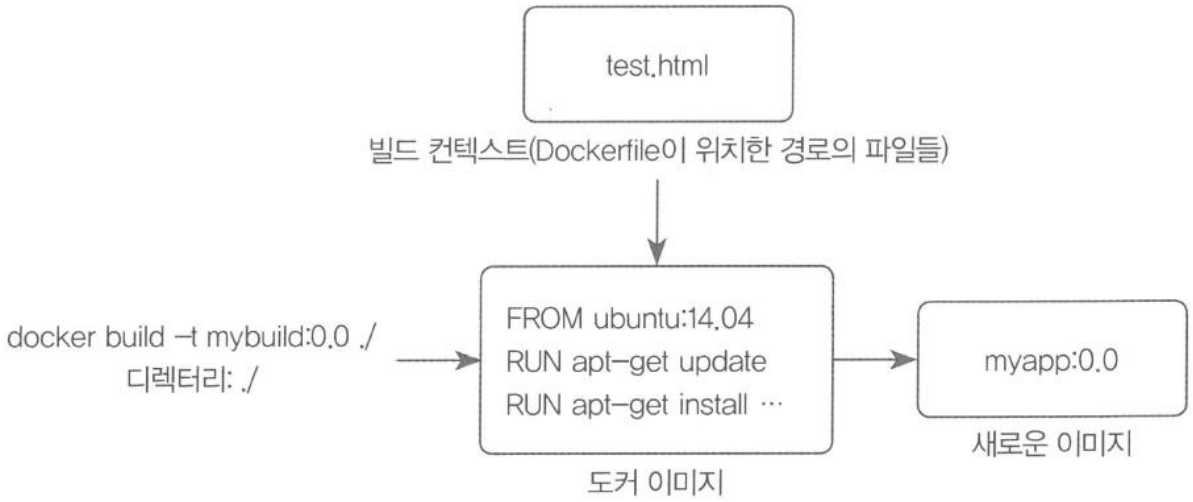
3. Dockerfile 빌드

1) 이미지 생성

```
root@test03:~/dockertest# docker build -t suejin:0.0 .
Sending build context to Docker daemon  16.38kB
Step 1/7 : FROM ubuntu:18.04
 ---> 8d5df41c547b
Step 2/7 : MAINTAINER Suejin
```

2) 컨테이너 생성

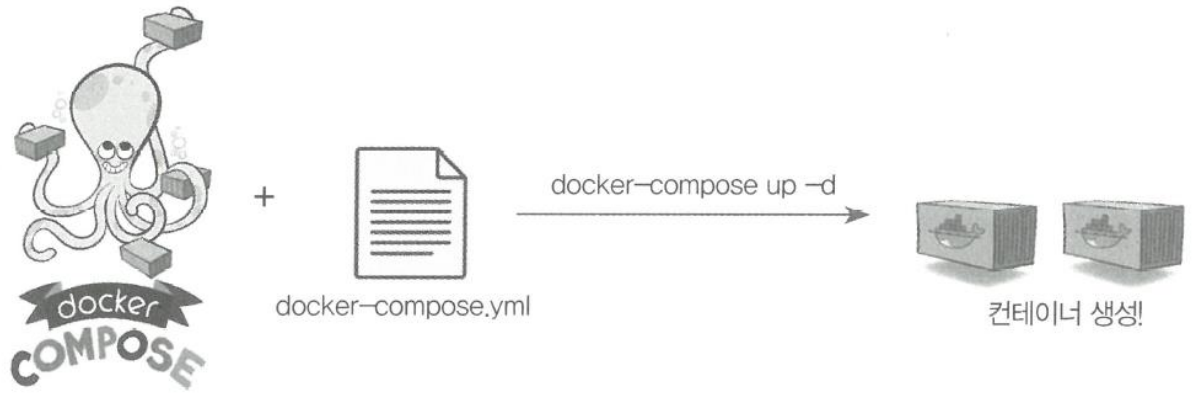
```
root@test03:~/dockertest# docker run -d -P --name suejintest suejin:0.0
856eff237d58c2c5324a6a4c9a6de37beb15edb5c80f61de48522c5a01562592
root@test03:~/dockertest# docker port suejintest
80/tcp -> 0.0.0.0:49153
80/tcp -> :::49153
```



이미지 빌드를 시작하면 도커는 가장 먼저 빌드 컨텍스트를 읽어 들인다. 빌드 컨텍스트는 이미지를 생성하는데 필요한 각종 파일, 소스코드, 메타데이터 등을 담고 있는 디렉터리를 의미한다. Dockerfile이 위치한 디렉터리가 빌드 컨텍스트가 된다.

4. Docker Compose

여러 개의 컨테이너를 하나의 서비스로 정의해 컨테이너 묶음으로 관리할 수 있는 작업 환경을 제공한다.



- Docker-compose.yaml

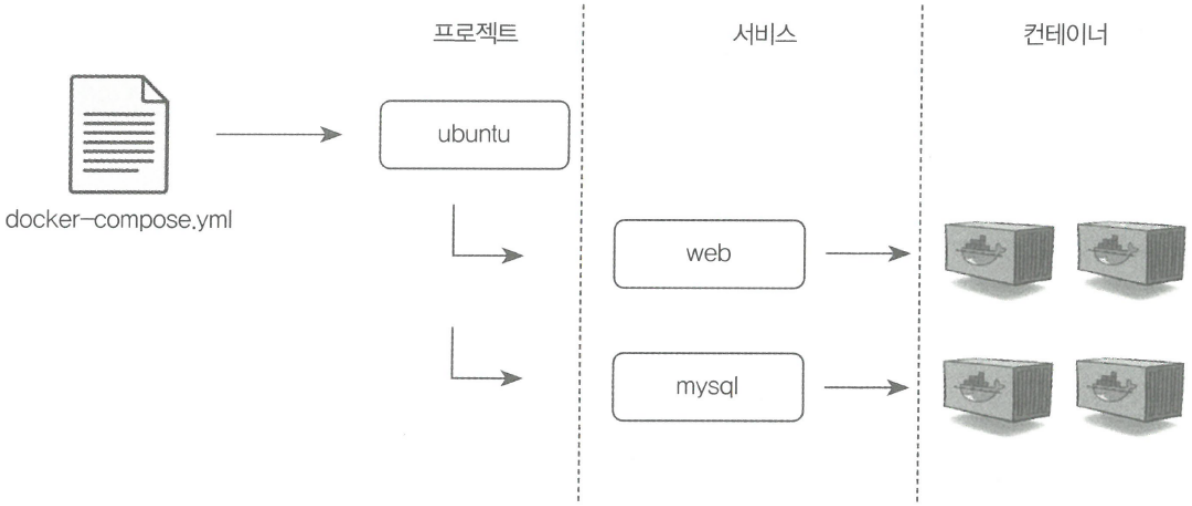
```
version: '3.0'
services:
  web:
    image: suejin:0.0
    ports:
    - "80:80"
    links:
      - mysql:db
    command: apachectl -DFOREGROUND
  mysql:
    image: suejin/mysql:latest
    command: mysqld
```

- Services : 서비스는 도커 컴포즈로 생성할 컨테이너 옵션을 정의한다. 이 항목에 쓰인 각 서비스는 컨테이너로 구현되며 하나의 프로젝트로서 도커 컴포즈에 의해 관리된다.
- Links: docker run 명령어의 –link와 같으며 다른 서비스에 서비스 명만으로 접근할 수 있도록 설정한다.
- Command : 컨테이너가 실행 될 때 수행할 명령어를 설정하며, docker run 명령어의 마지막에 붙는 커맨드와 같다.

```
root@test03:~/ubuntu# docker-compose up -d
```

[프로젝트 이름]_[서비스 이름]_[서비스 내에서 컨테이너의 번호]

Ex) ubuntu_mysql, web_1, 2



- Scale 명령어로 서비스 별로 컨테이너를 추가로 증가 시킬 수 있다.

```
root@test03:~/ubuntu# docker-compose scale mysql=2
WARNING: The scale command is deprecated. Use the up command with the --scale flag instead.
Starting root_mysql_1 ... done
Creating root_mysql_2 ... done
```

<Reference>

1. 시작하세요! 도커 쿠버네티스 입문 책
2. https://jjeongil.tistory.com/1588
3. https://yoonhoji.tistory.com/101