Docker로 워드프레스 블로그 구축하기

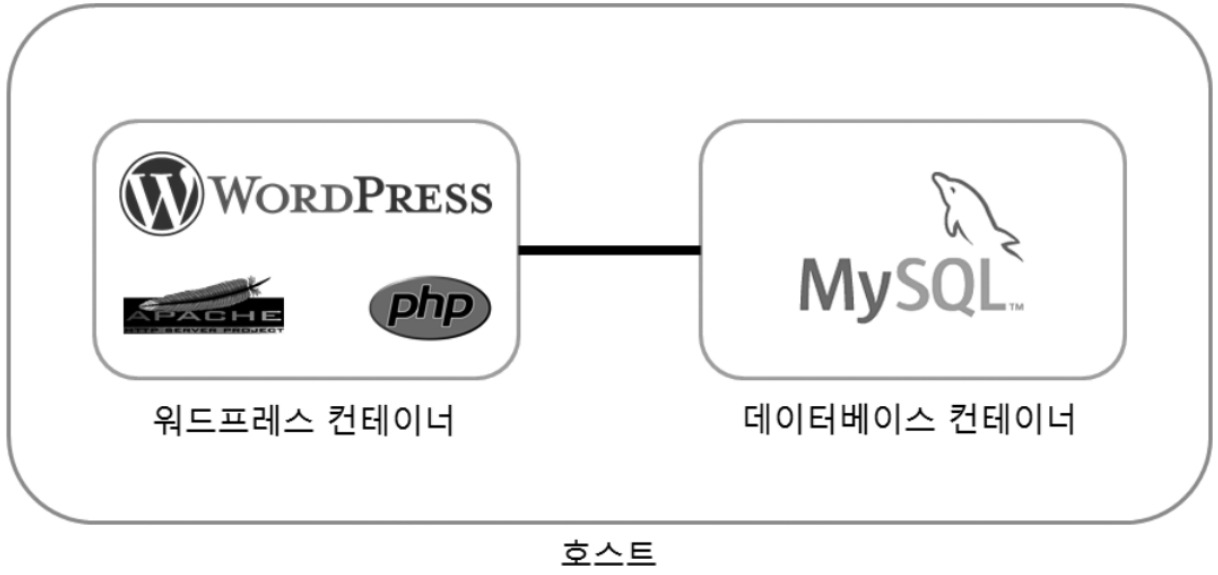
목차

Docker로 워드프레스 블로그 구축하기

워드프레스는 오픈 소스 설치형 블로그입니다. 이미 Docker Hub에 공식 이미지가 올라와 있지만, 이 장에서는 Docker로 직접 구축하는 방법을 알아보겠습니다.
먼저 워드프레스 이미지와 데이터베이스 이미지 두 개를 만듭니다.

- 워드프레스 이미지: 웹 서버로 사용할 Apache를 설치합니다. 그리고 워드프레스가 PHP로 작성되어 있으므로 PHP도 설치합니다.
- 데이터베이스 이미지: 워드프레스가 MySQL 데이터베이스를 사용하므로 MySQL을 설치합니다.

워드프레스 컨테이너에서 데이터베이스 컨테이너를 사용할 수 있도록 컨테이너를 생성할 때 docker run 명령의 --link 옵션으로 연결합니다.



1. 워드프레스 Dockerfile 작성하기

먼저 워드프레스 Docker 이미지를 생성합니다. wordpress 디렉터리를 생성하고 다음 내용을 Dockerfile로 저장합니다.

```
~$ mkdir wordpress
~$ cd wordpress
```

~/wordpress/Dockerfile

```
FROM ubuntu:14.04

RUN apt-get update
RUN apt-get install -y apache2 php5 php5-mysql mysql-client wget

WORKDIR /var/www
RUN wget http://ko.wordpress.org/wordpress-4.0-ko_KR.tar.gz -O - | tar -xz

WORKDIR /etc/apache2/sites-enabled
RUN sed -i "s/\/var\/www\/html/\/var\/www\/wordpress/g" 000-default.conf

WORKDIR /var/www/wordpress
RUN mv wp-config-sample.php wp-config.php
RUN sed -i "s/'database_name_here'/'wp'/g" wp-config.php && \
    sed -i "s/'username_here'/'root'/g" wp-config.php && \
    sed -i "s/'password_here'/getenv('DB_ENV_MYSQL_ROOT_PASSWORD')/g" wp-config.php && \
    sed -i "s/'localhost'/'db'/g" wp-config.php

ADD entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

ENTRYPOINT /entrypoint.sh
```

저는 우분투 14.04에 apt-get으로 필요한 패키지를 설치하도록 구성하였습니다.

- FROM으로 **ubuntu:14.04**를 기반으로 이미지를 생성하도록 설정합니다.
- `apt-get update`로 패키지 목록을 최신 상태로 업데이트한 뒤 **apache2**, **php5**, **php5-mysql**, **mysql-client**, **wget**을 설치합니다.
- `wget`으로 **/var/www** 디렉터리에 워드프레스 소스 파일을 다운로드한 뒤 압축을 해제합니다.
- `sed`로 **/etc/apache2/sites-enabled** 디렉터리의 000-default.conf 파일의 내용을 수정합니다. 웹 서버 기본 디렉터리를 **/var/www/html**에서 **/var/www/wordpress**로 바꿔서 워드프레스 소스를 사용할 수 있도록 합니다.
- **/var/www/wordpress** 디렉터리의 **wp-config-sample.php** 파일을 **wp-config.php** 파일로 이름을 바꾼 뒤 `sed`로 DB 설정을 수정합니다.
  - **DB_NAME**에 **wp**를 설정합니다.
  - **DB_USER**에 **root**를 설정합니다.
  - **DB_PASSWORD**에 환경 변수의 **DB_ENV_MYSQL_ROOT_PASSWORD**를 사용하도록 설정합니다. `docker run` 명령의 `--link` 옵션으로 컨테이너를 연결했을 때 연결한 컨테이너의 환경 변수는 `<별칭>_ENV_<환경 변수>` 형식입니다. 우리는 컨테이너를 연결할 때 별칭을 **db**로 하고, 데이터베이스 컨테이너에서 환경 변수는 **MYSQL_ROOT_PASSWORD**를 사용할 것이기 때문에 **DB_ENV_MYSQL_ROOT_PASSWORD**가 됩니다.
  - 컨테이너를 연결할 때 별칭을 **db**로 할 것이므로 **DB_HOST**에 **db**를 설정합니다.
- entrypoint.sh 파일을 추가한 뒤 실행할 수 있도록 권한을 설정합니다.
- ENTRYPOINT에 /entrypoint.sh 파일을 설정하여 컨테이너가 시작되었을 때 스크립트 파일을 실행합니다.

1. 워드프레스 Dockerfile 작성하기

다음 내용을 entrypoint.sh로 저장합니다.

~/wordpress/entrypoint.sh

```
#!/bin/sh

mysql -h db -uroot -p$DB_ENV_MYSQL_ROOT_PASSWORD -e "create database wp"
apachectl -DFOREGROUND
```

- 워드프레스는 미리 MySQL 데이터베이스를 생성해주어야 합니다. 따라서 `mysql` 명령으로 db에 접속한 뒤 wp 데이터베이스를 생성합니다. 사용자 계정은 root이고 비밀번호는 환경 변수의 DB_ENV_MYSQL_ROOT_PASSWORD를 활용합니다.
- Apache 웹 서버를 foreground로 실행합니다. 여기서 Apache를 foreground로 실행하지 않으면 `docker run -d`로 컨테이너를 생성해도 바로 정지되므로 주의합니다.

`docker build` 명령으로 이미지를 생성합니다.

```
~/wordpress$ sudo docker build --tag wordpress .
```

2. MySQL 데이터베이스 Dockerfile 작성하기

이제 데이터베이스 이미지를 생성합니다. mysql 디렉터리를 생성하고 다음 내용을 Dockerfile로 저장합니다.

```
~$ mkdir mysql
~$ cd mysql
```

~/mysql/Dockerfile

```
FROM ubuntu:14.04

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update
RUN echo "mysql-server mysql-server/root_password password" | debconf-set-selections
RUN echo "mysql-server mysql-server/root_password_again password" | debconf-set-selections
RUN apt-get install -y mysql-server

WORKDIR /etc/mysql
RUN sed -i "s/127.0.0.1/0.0.0.0/g" my.cnf

ADD entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
EXPOSE 3306

ENTRYPOINT /entrypoint.sh
```

- ENV로 환경 변수 **DEBIAN_FRONTEND**에 **noninteractive**를 반드시 설정합니다. `apt-get`으로 MySQL 패키지를 설치하면 사용자가 직접 root 비밀번호를 입력하는 부분이 나옵니다. 하지만, Docker 이미지를 생성할 때는 입력을 할 수가 없으므로 **noninteractive**를 설정하여 사용자 입력 없이 넘어가 야합니다.
- `apt-get update`로 패키지 목록을 최신 상태로 업데이트합니다.
- **mysql-server mysql-server/root_password password**를 `debconf-set-selections`에 설정하여 **noninteractive**로 넘어갔던 부분에 비밀번호 설정을 적용합니다. **password** 뒤에 실제로 사용할 비밀번호를 입력해도 되지만, 우리는 `docker run` 명령에서 `-e` 옵션으로 비밀번호를 설정할 것이므로 아무것도 입력하지 않습니다.
- **mysql-server mysql-server/root_password_again password**도 위와 동일합니다.
- `apt-get install`로 **mysql-server** 패키지를 설치합니다.
- `sed`로 **/etc/mysql** 디렉터리의 **my.cnf** 파일 내용을 수정합니다. **bind-address = 127.0.0.1** 부분을 **bind-address = 0.0.0.0**으로 수정합니다. 이 부분을 수정하지 않으면 외부에서 MySQL에 접속할 수 없습니다.
- entrypoint.sh 파일을 추가한 뒤 실행할 수 있도록 권한을 설정합니다.
- EXPOSE에 **3306**을 설정하여 **3306**번 포트에 접속할 수 있도록 합니다.
- ENTRYPOINT에 /entrypoint.sh 파일을 설정하여 컨테이너가 시작되었을 때 스크립트 파일을 실행합니다.

2. MySQL 데이터베이스 Dockerfile 작성하기

다음 내용을 entrypoint.sh로 저장합니다.

~/mysql/entrypoint.sh

```
#!/bin/bash

if [ -z $MYSQL_ROOT_PASSWORD ]; then
  exit 1
fi

mysql_install_db --user mysql > /dev/null

cat > /tmp/sql <<EOF
USE mysql;
FLUSH PRIVILEGES;
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
UPDATE user SET password=PASSWORD("$MYSQL_ROOT_PASSWORD") WHERE user='root';
EOF

mysqld --bootstrap --verbose=0 < /tmp/sql
rm -rf /tmp/sql

mysqld
```

- 환경 변수에 **MYSQL_ROOT_PASSWORD**가 없으면 데이터베이스를 실행하지 않고 빠져나옵니다.
- `mysql_install_db`로 데이터베이스 파일을 설치합니다.
- MySQL **root** 계정의 비밀번호를 설정하는 SQL문을 **/tmp/sql** 파일로 저장합니다. 비밀번호는 환경 변수의 **MYSQL_ROOT_PASSWORD**에 저장된 값을 사용합니다. Dockerfile에서 비밀번호를 설정하지 않고 이곳에서 비밀번호를 설정하는 이유는 `docker run` 명령의 `-e` 옵션으로 비밀번호를 설정하기 위해서입니다. `-e` 옵션으로 설정한 환경 변수 값은 CMD, ENTRYPOINT에서만 사용할 수 있습니다.
- `mysqld`에 `--bootstrap` 옵션을 설정하고 **/tmp/sql** 파일을 입력하여 **root** 계정의 비밀번호를 설정합니다. 그리고 **/tmp/sql** 파일은 삭제합니다.
- 마지막으로 `mysqld`를 실행합니다. Apache와 마찬가지로 MySQL도 foreground로 실행합니다.

`docker build` 명령으로 이미지를 생성합니다.

```
~/mysql$ sudo docker build --tag mysql .
```

3. 워드프레스와 데이터베이스 컨테이너 생성하기

워드프레스와 데이터베이스 이미지 준비가 끝났으니 컨테이너를 생성합니다.

```
$ sudo docker run -d --name db -e MYSQL_ROOT_PASSWORD=examplepassword mysql
$ sudo docker run -d --name example-wp -p 80:80 --link db:db wordpress
```

- 데이터베이스 컨테이너를 생성할 때 `-e` 옵션을 사용하여 **MYSQL_ROOT_PASSWORD**에 사용할 **root** 계정의 비밀번호를 설정합니다.
- 워드프레스 컨테이너를 생성할 때 `--link` 옵션을 사용하여 **db** 컨테이너를 **db** 별칭으로 연결합니다. 그리고 `-p` 옵션을 사용하여 외부에서 80번 포트에 접근할 수 있도록 설정합니다.