02_Dockerfile_code

- Docker 위에서 구현했던 것들(Wordpress, php, mysql, autoindex)을 Dockerfile과 쉘스크립트를 작성하자.
- 과제에서 기본적으로 지켜야할 것들
  - 서버 구성에 필요한 모든 파일을 srcs 폴더에 넣어야 한다.
  - WordPress 웹 사이트에 필요한 모든 파일은 srcs 폴더에 있어야 합니다.
  - Dockerfile 파일은 repository의 root에 있어야 하며, container를 빌드할 것 이다.
    - docker-compose은 사용할 수 없다.
- 생각할 것들
  - 어떤 파일들을 srcs 폴더에 넣어주어야 하는가?
  - server를 구축하기 위해 입력했던 명령어들을 dockerfile과 쉘스크립트에 어떻게 나눠 넣을 것인가?
- 구글링과 많은 블로그, 그 중에서도 daehyunlee 님과 jasekim 님의 블로그를 참고하면서 작성을 했다.
- Dockerfile : 프로젝트 구축에 필수적인 패키지를 설치하는 명령어를 작성한다.
  - 설치항목
    - nginx
    - php7.3-fpm
    - mariadb-server, php-mysql, php-mbstring
    - openssl
    - vim
    - crul
- srcs : 서버 구성에 필요한 모든 파일을 넣어준다.
  - nginx, mysql, blowfish 등의 설정파일은 사용자가 직접 작성하기 복잡한 파일이므로 srcs 폴더에 넣어주자.
  - nginx 서버의 ssl, autoindex, redirection 설정파일
    - 위치 : etc/nginx/sites-available/default
  - mysql의 wordpress DB 설정파일
    - 위치 : var/www/html/wordpress/wp-config.php
  - blowfish 설정파일
    - 위치 : var/www/html/phpmyadmin/config.inc.php

    - run.sh 실행파일
- 쉘스크립트(run.sh) : 그외 프로젝트 설정을 위해 작성했던 명령어를 적어준다.

Dockerfile

- container 내부의 파일을 hostOS의 srcs 폴더로 옮기기

```
docker cp etc/nginx/sites-available/default ./srcs
docker cp var/www/html/phpmyadmin/config.inc.php ./srcs
#docker cp var/www/html/wordpress/wp-config.php ./srcs
```

```
# 1
FROM  debian:buster

# 2
ENV   AUTOINDEX on

# 3
ARG   WP_NAME=wordpress
ARG   WP_USER=hkwon
ARG   WP_PWD=hkwon

# 4
LABEL maintainer="hkwon@student.42seoul.kr"

# 5
RUN   apt-get -y update; apt-get -y upgrade

# 6
RUN   apt-get -y install nginx openssl php-fpm mariadb-server php-mysql php-mbstring vim curl

# 7
COPY  ./srcs/* ./

# 8
RUN   openssl req -newkey rsa:4096 -days 365 -nodes -x509 \
    -subj "/C=KR/ST=Seoul/L=Gaepo/O=42Seoul/OU=hkwon/CN=localhost" \
    -keyout localhost.dev.key -out localhost.dev.crt; \
    mv localhost.dev.crt etc/ssl/certs/; \
    mv localhost.dev.key etc/ssl/private/; \
    chmod 600 etc/ssl/certs/localhost.dev.crt etc/ssl/private/localhost.dev.key; \
    cp /default /etc/nginx/sites-available/default

# 9
RUN   curl -O https://wordpress.org/latest.tar.gz; \
    tar -xzf latest.tar.gz -C var/www/html/; \
    chown -R www-data:www-data /var/www/html/wordpress; \
    cp var/www/html/wordpress/wp-config-sample.php var/www/html/wordpress/wp-config.php; \
    sed -i "s/database_name_here/${WP_NAME}/g" /var/www/html/wordpress/wp-config.php; \
    sed -i "s/username_here/${WP_USER}/g" /var/www/html/wordpress/wp-config.php; \
    sed -i "s/password_here/${WP_PWD}/g" /var/www/html/wordpress/wp-config.php

# 10
RUN   curl -O https://files.phpmyadmin.net/phpMyAdmin/5.0.4/phpMyAdmin-5.0.4-all-languages.tar.gz; \
    tar -xzf phpMyAdmin-5.0.4-all-languages.tar.gz -C var/www/html; \
    mv var/www/html/phpMyAdmin-5.0.4-all-languages var/www/html/phpmyadmin; \
    cp /config.inc.php var/www/html/phpmyadmin/
```

```
# 11
RUN   service mysql start; \
    mysql -e "CREATE DATABASE ${WP_NAME}; \
    USE ${WP_NAME}; \
    CREATE USER '${WP_USER}'@'localhost' IDENTIFIED BY '${WP_PWD}'; \
    GRANT ALL PRIVILEGES ON wordpress.* TO '${WP_USER}'@'localhost' WITH GRANT OPTION; \
    FLUSH PRIVILEGES;"

# 12
RUN   chown -R www-data:www-data /var/www/html/

# 13
EXPOSE  80 443

# 14
ENTRYPOINT ["/bin/bash", "-C", "/run.sh"]
```

- #1
  - FROM : 어떤 image를 기반으로 시작하는지 정의한다.
    - debian:buster(OS 중 하나)를 이용해서 image를 만들 것이다.
- #2
  - ENV : 환경변수 설정
    - autoindex를 runtime시 설정하여 사용가능 하지만 run할 때 환경변수를 설정하여 사용.
- #3
  - ARG : Dockerfile 내부 변수 선언
    - wordpress 안에 wp-config.php file의 값을 할당하여 관리하기 위해 사용.
- #4
  - LABEL : 생성된 image에 대해 여러가지 정보를 담는다.
    - 이미지의 버전 정보, 작성자, 코멘트 등 이미지 상세 정보를 작성해두기 위해 사용.

```
docker inspect --format="{{ .Config.Labels }}" <컨테이너명>
```

- #5
  - RUN : Shell script 또는 명령 실행
    - 패키지 관리자 설치
- #6
  - 서버 구축에 필요한 패키지 설치
    - nginx : Docker 위에 OS로 Debian:buster를 올린 후 웹서버로 사용한다.
    - openssl : ssl위에서 http가 동작하게 만들어 준다.

- php-fpm : nginx만으로 동적페이지를 구성할 수 없어 동적페이지로 사용할 수 있게 도와준다.

  ▼ php와 php-fpm

  - php
    - php란 동적 페이지를 위해 필요한 기술. 대표적인 서버 사이드 스크립트 언어.
    - wordpress 웹 페이지 또한 php기반으로 php가 설치되어야 정상적으로 동작이 가능하다.
  - php-fpm(FastCGI Process Manger)
    - 일반 CGI보다 빠른 처리가 가능한 FastCGI.
    - php-fpm을 통해 nginx와 php를 연동시켜 웹 서버가 정적 content 뿐만 아니라 동적 conten를 다룰 수 있게 만드는 것.
    - CGI: Common Gateway Interface
      - 두 컴퓨터(서버-클라이언트) 사이의 HTML 등의 언어를 양방향으로 번역해주는 것.
      - nginx는 웹 서버이기 때문에 정적 content밖에 다루지 못한다.
      - 동적 페이지를 구현하기 위해서는 웹 서버 대신 동적 content를 읽은 후 html로 변환시켜 웹 서버에 다시 전달 해주는 외부 프로그램이 필요하다. → php module
      - 이러한 연결 과정의 방법 또는 규약을 정의한 것.
- mariadb-server : 과제에서는 mySQL을 이용하라고 하지만, debian buster에서는 mariaDB만을 지원한다.
- php-mysql : php에서 mysql에 접근할 수 있게 해준다.
- php-mbstring : 한국어, 중국어, 일본어와 같은 multibyte를 처리할 수 있게 해준다.
- vim : 파일을 읽고 수정하기위해 사용.
- curl : http, url을 통해 경로의 데이터를 가져오기 위해 사용.

- #7
  - srcs 폴더 안에 있는 파일(host OS에서 관리)을 container안에 넣어준다.
    - nginx 서버의 ssl, autoindex, redirection 설정파일.
    - blowfish 설정파일.
    - wordpress DB 설정파일도 넣어주었다가 ARG 변수를 이용해 DB 설정파일을 관리하여 제외하였다.
- #8
  - openssl

- req : 인증서 요청 및 인중서 생성 유틸.
- -newkey : 개인키를 생성하기 위한 옵션.
- days 365 : 인증서의 유효기간.
- -nodes : 개인키를 암호화하지 않기 위한 옵션. 이 옵션이 없을 경우 최소 4자의 암호를 입력해 주어야한다.
- -x509 : 인증서 요청 대신 자체 서명된 인증서를 출력하는 옵션. 테스터 인증서 혹은 자체 서명된 root CA를 생성하는데 사용한다.
- -keyout <키 파일 이름> : 키 파일 이름을 지정해 키 파일 생성.
- -out <인증서 이름> : 인증서 이름을 지정해 인증서 생성.
- -subj"" : subject를 입력하기 위한 옵션.

  ▼ Option

  - CN(Command Name) : 일반 이름(인증서 고유 이름), 대부분의 인증기관 CA에서는 SSL인증서 신청 시 도메인명을 CN을 지정.
  - O(Organization) : 기관명.
  - OU(Organization Unit) : 회사/기관 내의 '사업부, 부분, 부서, 본부, 과, 팀' 정도.
  - L(City/Locality) : 시/도.
  - S(State/County/Region) : 구/군.
  - ST(Street) : 나머지 상세 주소. (OV, EV 인증시 필요)
  - C(Country) : 국가를 나타내는 ISO 코드를 지정. 한국은 KR, 미국은 US 등 2자리 코드.
- 권한설정 : 보안 및 인증 관련 파일을 root만 허용하도록 한다.
- 수정한 default file을 옮겨준다.

  ▼ default

  - 여기 중요한 부분이 80번 port로 들어오면 443번 port로 redirection 시켜주는 것이다.
  - 301 Moved Permanently는 영구적인 URL 리다이렉션을 위해 사용된다.
    - 참고1
    - 참고2
  - $host$request_uri
    - nginx의 변수로 https://localhost/wordpress로 접속했다면 host = localhost, requset_uri = /wordpress가 된다.

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
server {
  listen 80 default_server;
  listen [::]:80 default_server;

  return 301 https://$host$request_uri;
}

# SSL configuration
#
server {

  listen 443 ssl default_server;
  listen [::]:443 ssl default_server;

  # ssl setting
  ssl on;
  ssl_certificate /etc/ssl/certs/localhost.dev.crt;
  ssl_certificate_key /etc/ssl/private/localhost.dev.key;

  # set root dic
  root /var/www/html;

  # Add index.php to the list if you are using PHP
  # autoindex
  index index.html index.htm index.php #index.nginx-debian.html;

  server_name ft_server;

  # /를 기준으로 request_uri에 해당하는 directory를 찾는다.
  # 기준은 location의 /로 지정
  location / {
    # try_files로 $uri에 해당하는 file과 directory를 찾는다.
    # file을 먼저 찾으며 file과 directory가 없다면 404 error.
    # directory는 있으나 위에 index로 지정한 file이 없다면 403 error.
    # atuoindex를 통해 uri로 접근한 directory에 index 목록에 해당하는 file이
    # 없다면 리스트 형태로 directory와 file을 보여준다.
    autoindex off;
    try_files $uri $uri/ =404;
  }

  # php
  location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:var/run/php/php7.3-fpm.sock;
```

```
  }
}


# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
# listen 80;
# listen [::]:80;
#
# server_name example.com;
#
# root /var/www/example.com;
# index index.html;
#
# location / {
#   try_files $uri $uri/ =404;
# }
#}
```

- #9
  - wordpress
    - php로 작성된 홈페이지를 제작, 관리하는 오픈소스 도구.
    - curl을 이용해 웹 서버로 부터 파일을 다운받은 후 var/www/html/ 에 압축을 풀어준다.
      - -O : 파일명을 수정하지않고 다운받을 수 있게 해준다.
    - chown : 권한 설정으로 wordpress의 유저, 그룹을 www-data로 설정한다.
      - nginx.conf 파일을 보면 user로 www-data로 작성되어 있다.
    - wp-config-sample.php 파일을 wp-config.php로 이름을 바꿔주고 sed 명령어를 통해 mysql 설정부분을 선언해준 변수로 바꿔준다.
- #10
  - phpmyadmin
    - mySQL을 웹 상에서 관리할 수 있게 php로 작성된 오픈소소 도구.
    - curl을 이용해 웹 서버로 부터 파일을 다운받은 후 var/www/html/ 에 압축을 풀어준다.
      - -O : 파일명을 수정하지않고 다운받을 수 있게 해준다.
    - phpmyadmin으로 폴더명을 변경해주고 srcs 폴더에 있는 config.inc.php 파일을 옮겨준다.
- #11
  - mySQL
    - 오픈소스 데이터베이스.
    - 선언한 변수를 통해 database와 user를 생성한다.

- #12
  - /var/www/html이 소유자가 root로 등록되어 있어 phpmyadmin에서 temp directory를 생성하지 못하는 이슈가 있으므로 권한을 바꿔준다.
- #13
  - 어떤 포트를 사용할 지 알려 container 내부에서 네트워크 포트를 수신 대기 상태로 만든다.
  - 처음에는 CMD로 사용을 했지만 container를 실행할 때 추가 인자 값을 주면 변경이 되므로 ENTRYPOINT를 이용해 반드시 수행되도록 해준다.
  - RUN vs CMD vs ENTRYPOINT
    - https://blog.leocat.kr/notes/2017/01/08/docker-run-vs-cmd-vs-entrypoint
    - https://bluese05.tistory.com/77

Shell script

- shell 명령어를 한줄씩 순차적으로 읽어 실행되도록 작성된 프로그램.
- .sh의 확장자를 가진다.
- 작성방법
  - 헤더
    - 다음 내용을 파일 상단에 쓰면 뒤에 나오는 코드를 모두 bash 명령어로 인식한다.

```
#!/bin/bash
```

  - 개행
    - 모든 명령어는 개행을 기준으로 순차적으로 실행된다.
  - 주석 : #을 사용한다.
  - 변수
    - =를 이용하여 선언하고 ${변수명}으로 사용한다.
    - =는 공백없이 붙여써야한다.
    - ""로 감싸서 사용하면 더 안전하다. (문자열에 공백을 포함한 값을 이용할 수 있기 때문)
  - 추가 문법 참고
- 실행방법

```
$ ./filename
$ sh filename
$ bash filename
```

- run.sh
  - nginx, mysql, php-fpm을 실행하기전 set_autoindex.sh 을 실행시켜 환경변수로 AUTOINDEX값이 들어오면 조건에 맞게 default 을 수정한다.
  - daemon off : nginx 서버를 foreground에서 실행할 수 있게 해준다. container안에서 상시 작동 중으로 만든다.
  - Docker background 실행과 daemon off
    - Docker background 실행
    - nginx daemon off

```
#!/bin/bash

#/bin/bash -C /set_autoindex.sh
bash set_autoindex.sh

service nginx start
service mysql start
service php7.3-fpm start
bash
# nginx -g "daemon off;"
```

- set_autoindex.sh
  - Autoindex를 환경변수를 통해 on/off를 시켜주었다.
  - 참고

```
#!/bin/bash

if [ "$AUTOINDEX" = "on" ]; then
  echo "autoindex[on]"
  sed -i "s/autoindex off;/autoindex on;/g" /etc/nginx/sites-available/default
else
  echo "autoindex[off]"
  sed -i "s/autoindex on;/autoindex off;/g" /etc/nginx/sites-available/default
fi
```