[과제 1] 해설 및 참고자료

💡 과제 1에 대한 해설을 부록의 형태로 준비하였습니다. Dockerfile을 분석해보며 개념을 확실히 짚고 넘어갑시다 😀

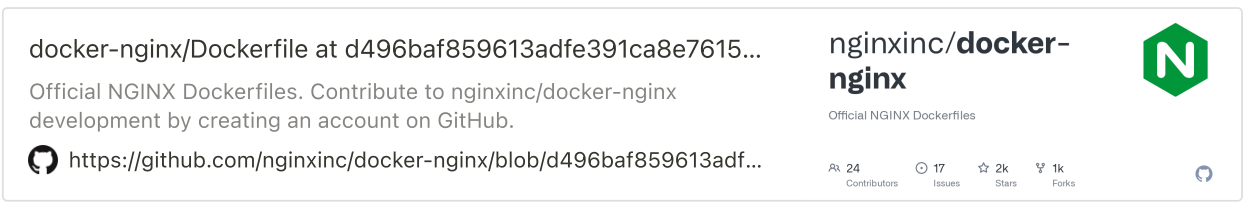
> 출처
> - **curl** : https://curl.se/

```
FROM nginx:alpine WORKDIR /usr/share/nginx/html RUN rm -rf ./* COPY ./* ./
ENTRYPOINT ["nginx", "-g", "daemon off;"] EXPOSE 80 HEALTHCHECK --interval=2s
--timeout=5s --retries=5 CMD curl --fail http://localhost:80/ || exit 1
```

1. 베이스 이미지 세팅

```
FROM nginx:alpine
```

alpine 리눅스 환경에서 구동되는 `nginx` 이미지를 베이스로 설정하였습니다. 작성한 `Dockerfile` 에는 따로 운영체제를 베이스로 삼지 않았는데 어떻게 alpine 리눅스가 설정된 것일까요? `nginx:alpine` 이미지를 빌드 한 `Dockerfile` 을 확인하면 그 비밀이 쉽게 풀립니다.



```
#
# NOTE: THIS DOCKERFILE IS GENERATED VIA "update.sh"
#
# PLEASE DO NOT EDIT IT DIRECTLY.
#
FROM alpine:3.14

LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>"

ENV NGINX_VERSION 1.21.3
ENV NJS_VERSION   0.6.2
ENV PKG_RELEASE   1

RUN set -x \
# create nginx user/group first, to be consistent throughout docker variants
    && addgroup -g 101 -S nginx \
    && adduser -S -D -H -u 101 -h /var/cache/nginx -s /sbin/nologin -G nginx -g nginx nginx \
    && apkArch="$(cat /etc/apk/arch)" \
    && nginxPackages=" \
        nginx=${NGINX_VERSION}-r${PKG_RELEASE} \
        nginx-module-xslt=${NGINX_VERSION}-r${PKG_RELEASE} \
        nginx-module-geoip=${NGINX_VERSION}-r${PKG_RELEASE} \
        nginx-module-image-filter=${NGINX_VERSION}-r${PKG_RELEASE} \
        nginx-module-njs=${NGINX_VERSION}.${NJS_VERSION}-r${PKG_RELEASE} \
    " \
```

`FROM` 에 alpine 리눅스 이미지가 세팅된 것을 Github 저장소에서 확인 가능합니다.

2. Working Directory 이동

```
WORKDIR /usr/share/nginx/html
```

nginx 기본 페이지 템플릿이 있는 `/usr/share/nginx/html` 로 이동합니다. 이는 기존에 존재하던 html 파일을 삭제하고 템플릿의 파일을 복사하기 위함입니다.

3. 기존 템플릿 삭제

```
RUN rm -rf ./*
```

현재 디렉토리(`./`)에 존재하는 모든 파일을 애스터리스크(`*`)를 이용하여 삭제합니다.

`rm` 은 파일/디렉토리를 삭제하는 리눅스 명령어이며, `-r` 은 디렉토리까지 삭제, `-f` 는 강제 삭제를 하기 위한 옵션입니다.

4. 템플릿 파일 복사

```
COPY ./* ./
```

`COPY` 명령어를 통해 `Dockerfile` 과 같은 경로에 위치한 파일과 디렉토리를 현재 설정된 Working Directory(`/usr/share/nginx/html`)로 복사하였습니다.

5. nginx 웹서버 구동

```
ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

`ENTRYPOINT` 를 통해 nginx 웹서버를 구동합니다. `daemon off;` 옵션을 주는 것은 컨테이너의 관점에서 어플리케이션이 `FOREGROUND` 에서 실행되어야 종료 상태가 되는 것을 막을 수 있기 때문입니다. 컨테이너 내부의 어플리케이션이 `BACKGROUND` 로 실행되는 경우 도커 엔진에서는 이를 프로세스 종료로 간주하고 컨테이너를 exited 상태로 만듭니다.

6. 포트 명시

```
EXPOSE 80
```

`80` 포트를 명시합니다. `http` 통신을 하는 포트이며, 실제로 포트포워딩을 해주기 위해서는 `docker run -p 80:80` 형태로 실행 시에 반드시 세팅해주어야 합니다. `Dockerfile` 내부의 `EXPOSE` 명령은 별도로 포트를 publish 하는 기능을 수행하지 않습니다.

7. 컨테이너 작동 상태 모니터링

```
HEALTHCHECK --interval=2s --timeout=5s --retries=5 CMD curl --fail
http://localhost:80/ || exit 1
```

`HEALTHCHECK` 를 통해 컨테이너 작동 상태를 모니터링 할 수 있습니다. 이를 위해서는 작동 상태를 점검하기 위한 수행 명령을 `CMD` 로 세팅해주어야 합니다. `curl` 은 `http://localhost:80/` 로 접속을 시도하는 커맨드라인 툴이며, HTTP 뿐만 아니라 TELNET, HTTPS, IMAP, POP3, SFTP, FTP 등의 다양한 프로토콜을 지원합니다. `--fail` 옵션을 추가하면 접속 실패에 대한 에러코드를 반환 합니다.