# Docker 실습 3, 4 - Docker Image

- 0. Docker Image
- 1. Dockerfile
  - 1) Dockerfile 만들기
  - 2) 기본 명령어
  - 3) 간단한 Dockerfile 작성해보기
- 2. Docker build from Dockerfile
- 3. Docker Image 저장소
  - 1) Docker Registry
  - 2) Docker Hub

# 0. Docker Image

- · Docker image:
  - 。 어떤 애플리케이션에 대해서,
  - 。 단순히 애플리케이션 코드뿐만이 아니라.
  - 。 그 애플리케이션과 dependent 한 모든 것을 함께 패키징한 데이터
- Dockerfile
  - 사용자가 도커 이미지를 쉽게 만들 수 있도록, 제공하는 템플릿

### 1. Dockerfile

### 1) Dockerfile 만들기

• Dockerfile 이라는 이름으로 빈 file 을 하나 만들어봅니다.

```
# home 디렉토리로 이동합니다.
$ cd $HOME

# docker-practice 라는 이름의 폴더를 생성합니다.
$ mkdir docker-practice

# docker-practice 폴더로 이동합니다.
$ cd docker-practice

# Dockerfile 이라는 빈 파일을 생성합니다.
$ touch Dockerfile
```

```
# 정상적으로 생성되었는지 확인합니다.
$ ls
```

# 2) 기본 명령어

- Dockerfile 에서 사용할 수 있는 기본적인 명령어에 대해서 하나씩 알아보겠습니다.
  - 지금 모든 사용법을 자세히 알아야 할 필요는 없으며, 필요한 경우에 구글링할 수
     있을 정도면 충분합니다.

#### FROM

Dockerfile 이 base image 로 어떠한 이미지를 사용할 것인지를 명시하는 명령어입니다.

```
FROM <image>[:<tag>] [AS <name>]
# 예시
FROM ubuntu
FROM ubuntu:18.04
FROM nginx:latest AS ngx
```

#### COPY

○ <src> 의 파일 혹은 디렉토리를 <dest> 경로에 복사하는 명령어입니다.

```
# 예시
COPY a.txt /some-directory/b.txt
COPY my-directory /some-directory-2
```

#### RUN

。 명시한 커맨드를 도커 컨테이너에서 실행하는 것을 명시하는 명령어입니다.

```
RUN <command>
RUN ["executable-command", "parameter1", "parameter2"]
# 예시
RUN pip install torch
RUN pip install -r requirements.txt
```

#### CMD

- 。 명시한 커맨드를 도커 컨테이너가 **시작될 때**, 실행하는 것을 명시하는 명령어입니다.
  - 비슷한 역할을 하는 명령어로 ENTRYPOINT 가 있지만, 아직은 그 차이를 구분하기 어려울 수 있으므로 이번 강의에서는 ENTRYPOINT 에 대한 설명은 생략하겠습니다.
- 하나의 Docker Image 에서는 하나의 **CMD** 만 실행할 수 있다는 점에서 **RUN** 명령 어와 다릅니다.

```
CMD <command>
CMD ["executable-command", "parameter1", "parameter2"]
CMD ["parameter1", "parameter2"] # ENTRYPOINT 와 함께 사용될 때
# 예시
CMD python main.py
CMD
```

#### WORKDIR

- 이후 작성될 명령어를 컨테이너 내의 어떤 디렉토리에서 수행할 것인지를 명시하는 명령어입니다.
- 。 해당 디렉토리가 없다면 생성합니다.

```
WORKDIR /path/to/workdir
# 예시
WORKDIR /home/demo
```

#### • ENV

컨테이너 내부에서 지속적으로 사용될 environment variable 의 값을 설정하는 명령어입니다.

```
ENV <key> <value>
ENV <key>=<value>

# 예시

# default 언어 설정
RUN locale-gen ko_KR.UTF-8
ENV LANG ko_KR.UTF-8
ENV LANGUAGE ko_KR.UTF-8
ENV LANGUAGE ko_KR.UTF-8
```

#### EXPOSE

- 컨테이너와 연결할 포트/프로토콜을 지정할 수 있습니다.
- protocol 을 지정하지 않으면 TCP 가 디폴트로 설정됩니다.

```
EXPOSE <port>
EXPOSE <port>/<protocol>
# 예시
EXPOSE 8080
```

# 3) 간단한 Dockerfile 작성해보기

• vi Dockerfile 혹은 vscode 등 본인이 사용하는 편집기로 Dockerfile 을 열어 다음과 같이 작성해줍니다.

```
# base image 를 ubuntu 18.04 로 설정합니다.
FROM ubuntu:18.04

# apt-get update 명령을 실행합니다.
RUN apt-get update

# DOCKER CONTAINER 가 시작될 때, "Hello" 를 출력합니다.
CMD ["echo", "Hello"]
```

### 2. Docker build from Dockerfile

• docker build 명령어로 Dockerfile 로부터 Docker Image 를 만들어봅니다.

```
$ docker build --help
# 자세한 옵션들에 대한 설명은 생략
# Dockerfile 이 있는 경로에서 다음 명령을 실행합니다.
$ docker build -t my-image:v1.0.0 .
```

- 설명
  - (현재 경로에 있는 Dockerfile 로부터)
  - my-image 라는 **이름**과 v1.0.0 이라는 **태그**로 **이미지**를
  - 。 빌드하겠다라는 명령어

• 정상적으로 이미지 빌드되었는지 확인해보겠습니다.

```
# grep : my-image 가 있는지를 잡아내는 (grep) 하는 명령어
$ docker images | grep my-image
```

• 그럼 이제 방금 빌드한 my-image:v1.0.0 이미지로 docker 컨테이너를 **run** 해보겠습니다.

```
$ docker run my-image:v1.0.0
# Hello 가 출력되는 것 확인
```

# 3. Docker Image 저장소

## 1) Docker Registry

- 공식 문서
  - https://docs.docker.com/registry/
- 간단하게 도커 레지스트리를 직접 띄워본 뒤에, 방금 빌드한 my-image:v1.0.0 을 도커 레지스트리에 push 해보겠습니다.
- Docker Registry 는 이미 잘 준비된 도커 컨테이너가 존재하므로, 쉽게 사용할 수 있습니다.
- docker registry 를 띄워봅니다.

```
$ docker run -d -p 5000:5000 --name registry registry
$ docker ps
# 정상적으로 registry 이미지가 registry 라는 이름으로 생성되었음을 확인할 수 있습니다.
# localhost:5000 으로 해당 registry 와 통신할 수 있습니다.
```

my-image 를 방금 생성한 registry 를 바라보도록 tag 합니다.

```
$ docker tag my-image:v1.0.0 localhost:5000/my-image:v1.0.0
$ docker images | grep my-image
# localhost:5000/my-image:v1.0.0 로 새로 생성된 것을 확인할 수 있습니다.
```

• my-image 를 registry 에 push 합니다. (업로드합니다.)

```
$ docker push localhost:5000/my-image:v1.0.0
```

• 정상적으로 push 되었는지 확인합니다.

```
# localhost:5000 이라는 registry 에 어떤 이미지가 저장되어 있는지 리스트를 출력하는 명령
$ curl -X GET http://localhost:5000/v2/_catalog

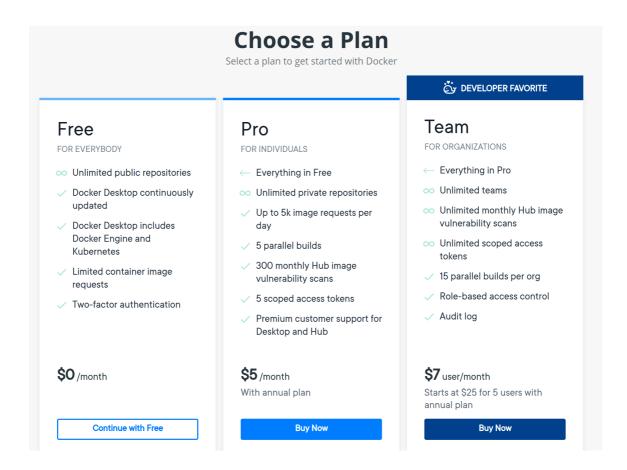
# 출력 : {"repositories":["my-image"]}

# my-image 라는 이미지 네임에 어떤 태그가 저장되어있는지 리스트를 출력하는 명령
$ curl -X GET http://localhost:5000/v2/my-image/tags/list

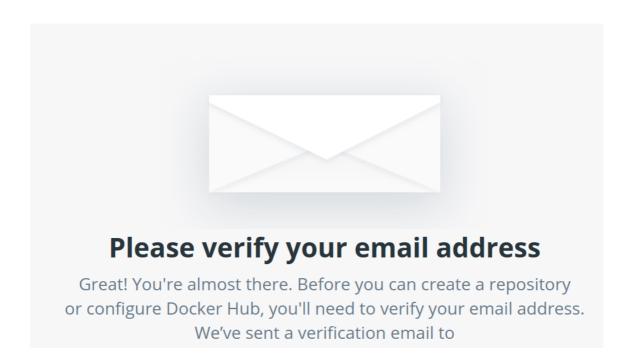
# 출력 : {"name":"my-image","tags":["v1.0.0"]}
```

### 2) Docker Hub

- 회원 가입
  - <u>hub.docker.com</u>
- · Choose a Plan
  - Free



#### • Email 인증



#### Docker login

```
$ docker login
# username, password 입력
# Login Succeeded!
```

• Docker Hub 를 바라보도록 tag 생성

```
$ docker tag my-image:v1.0.0 username/my-image:v1.0.0
# docker tag <image_name>:<tag_name> <user_name>/<image_name>:<tag>
```

• Docker image push to Docker Hub

```
$ docker push username/my-image:v1.0.0
# docker push <user_name>/<image_name>:<tag>
```

- Docker hub 의 본인 계정에서 업로드한 이미지 확인
  - https://hub.docker.com/repositories