제08장

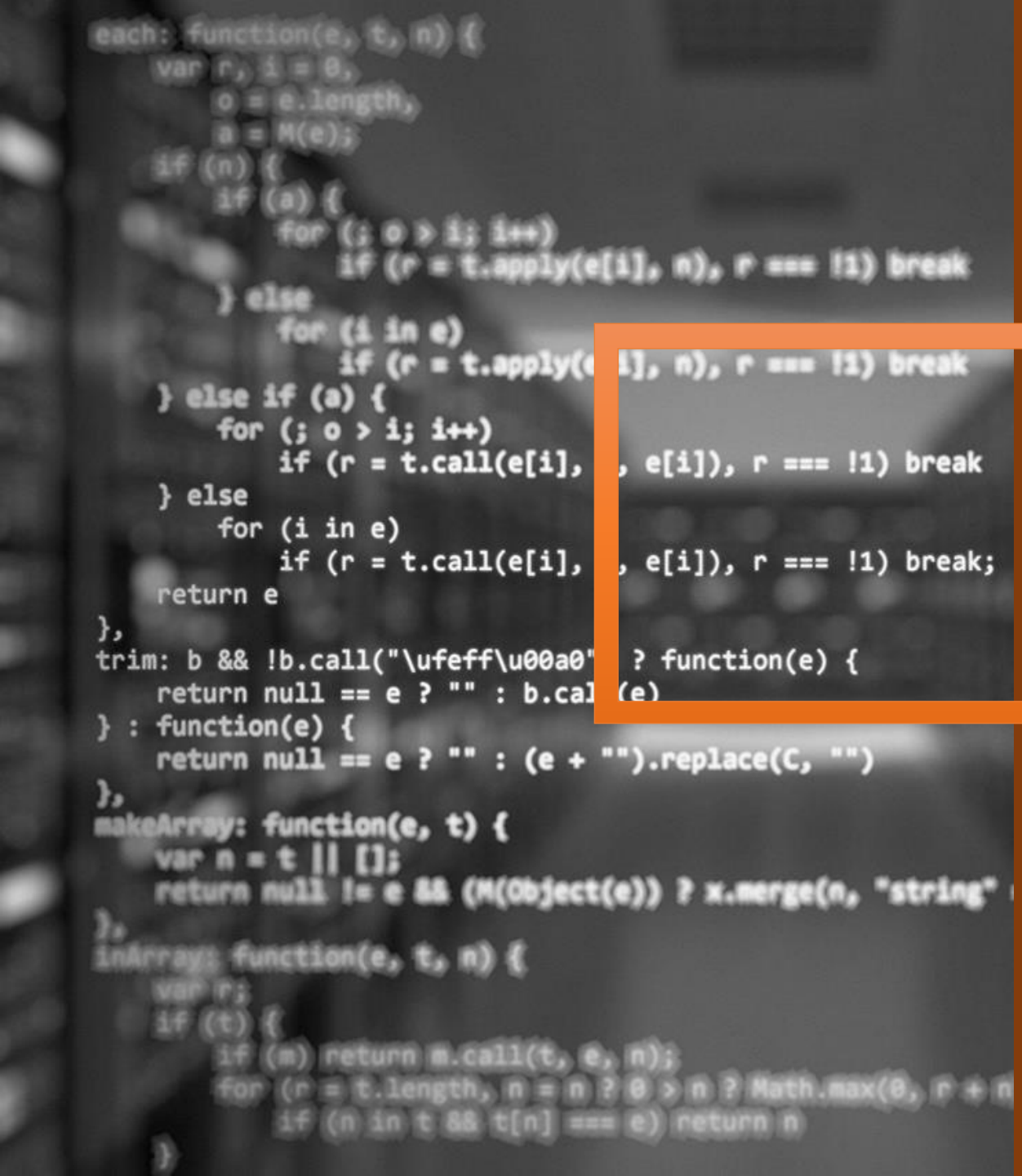
# 도커파일

DevOps

# 학습목표

1. 도커파일에 대해서 이해할 수 있다.
2. 도커파일에서 사용하는 명령어에 대해서 이해할 수 있다.
3. 도커파일을 이용해 도커 이미지를 빌드하고 이를 도커 허브에 업로드 할 수 있다.

# 목차



# 01. 도커파일

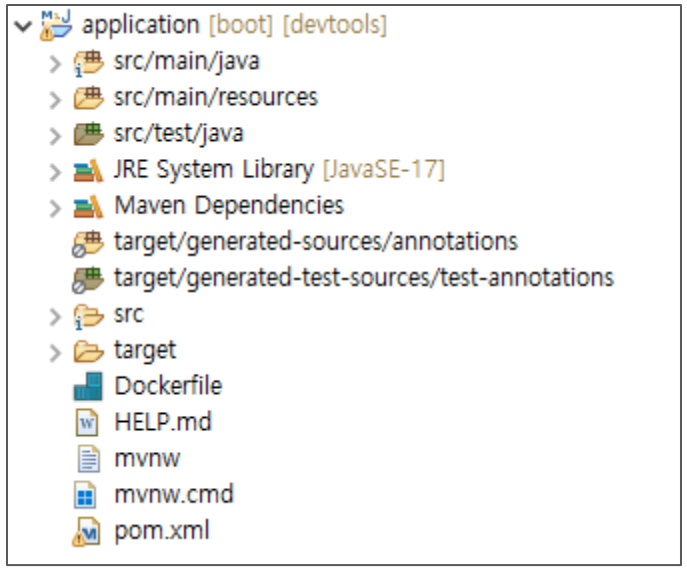
# 도커파일이란?

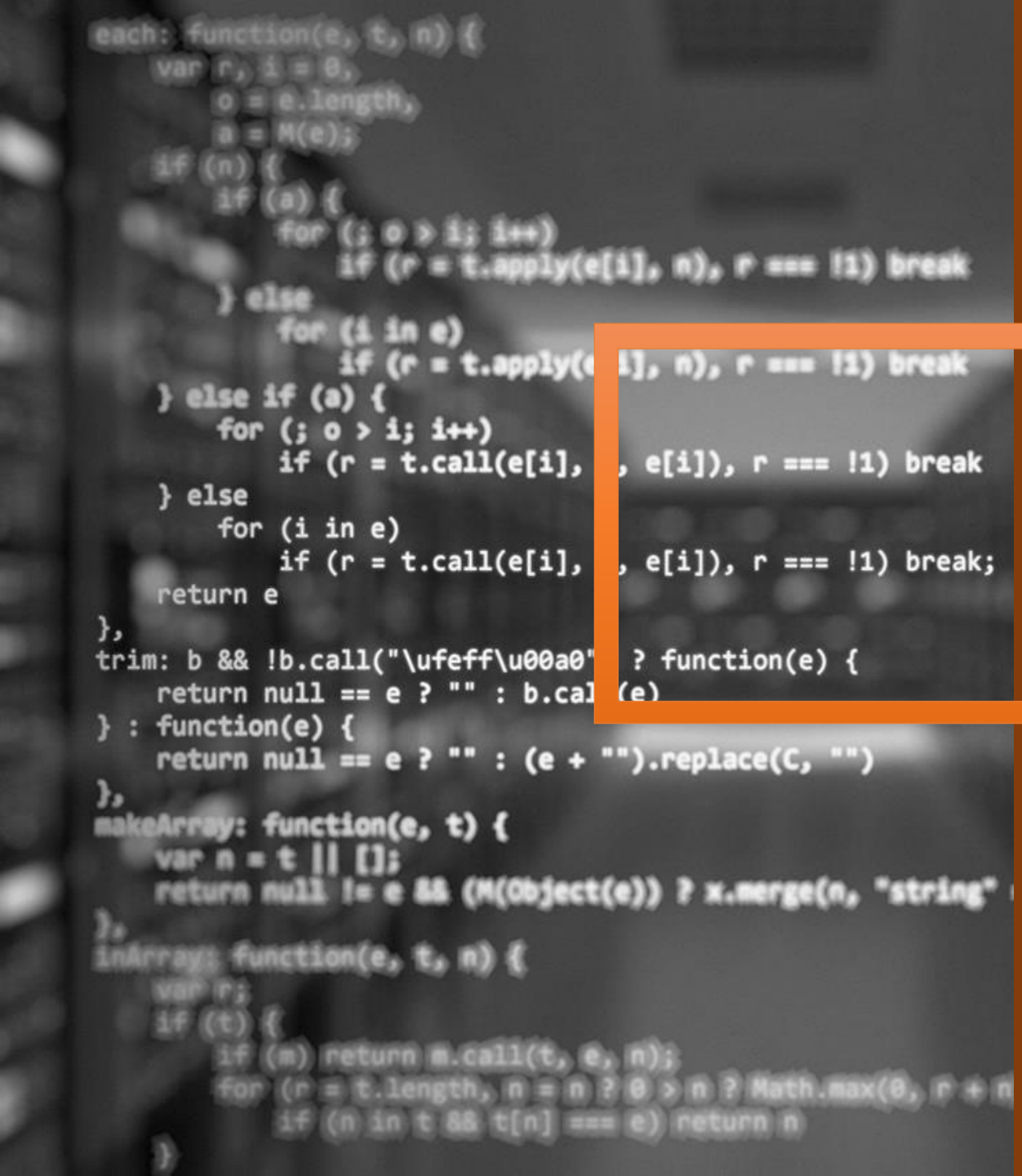
- Dockerfile
  - 도커 이미지를 만들 때 사용하는 텍스트 파일
  - 도커 파일 명령어를 이용해서 도커 컨테이너의 구성 정보를 작성해 놓은 파일
  - 확장자 없이 "Dockerfile"이라는 파일명을 사용
  - 다른 명칭을 사용하려면 docker image build 명령을 내릴 때 -f 옵션을 추가로 사용해야함

# 스프링 프로젝트와 도커파일

## 스프링 프로젝트와 도커파일

- 스프링 프로젝트를 도커 이미지로 만들기 위해서는 우선 스프링 프로젝트 내부에 Dockerfile을 생성해야 함
- 스프링 프로젝트의 최상위 디렉터리에 Dockerfile을 배치해야 함
- 스프링 프로젝트의 경우 Java 기반으로 동작하기 때문에 Dockerfile의 베이스 이미지는 Jdk로 설정해야 함





# 02. 도커파일 명령어

# Dockerfile 명령어

## FROM

- 설명
  - 베이스 이미지를 설정하는 커맨드
  - Official Image 사용을 권장함
  - Dockerfile의 시작은 반드시 FROM 이어야 함
- 형식
  - FROM 이미지명
- 예시
  - FROM ubuntu:22.04

# Dockerfile 명령어

## LABEL

- 설명
  - 이미지의 작성자, 제목, 버전, 설명 등의 정보를 작성하는 커맨드
  - 이미지의 상세 정보 확인 시 내용 확인 가능 (docker image inspect)

- 형식
  - LABEL KEY=VALUE

- 예시
  - LABEL title="제목" description="설명" version="버전"

# Dockerfile 명령어

## ENV

- 설명
  - 이미지 내부에서 사용할 환경 변수를 설정하는 커맨드
  - 설정한 환경 변수는 RUN, WORKDIR 명령에서 "$변수명" 형식으로 사용 가능
- 형식
  - ENV 변수명=값
- 예시
  - ENV MYSQL_DATABASE=db_menu

# Dockerfile 명령어

## ARG

- 설명
  - 이미지 빌드 과정에서 전달되는 값을 저장하는 변수를 선언하는 커맨드 (--build-arg 옵션)
  - 이미지 빌드 과정에서 값을 전달받지 않고 초기값을 지정할 수도 있음
  - 설정한 변수는 Dockerfile 내부에서 "${변수명}" 또는 "$변수명"으로 사용 가능
  - ENV와 달리 이미지가 빌드되는 동안에만 값이 유효함

- 형식
  - ARG 변수명
  - ARG 변수명=초기값

- 예시
  - ARG version
  - ARG port=9999

# Dockerfile 명령어

## EXPOSE

- 설명
  - 컨테이너가 사용할 포트를 지정하는 커맨드
  - TCP와 UDP 프로토콜 중 하나를 지정할 수 있음 (디폴트 TCP)
  - 컨테이너 내부 포트이므로 호스트에서 해당 포트로 컨테이너로 접근하려면 "docker run" 커맨드 시 -p 옵션으로 호스트의 포트 포워딩(Port Forwarding)을 해야 함

- 형식
  - EXPOSE 포트
  - EXPOSE 포트/프로토콜

- 예시
  - EXPOSE 8080
  - EXPOSE 80/tcp

# Dockerfile 명령어

## VOLUME

- 설명
  - 호스트의 특정 디렉터리를 컨테이너에 마운트할 때 사용하는 커맨드
  - VOLUME 명령으로 등록한 디렉터리는 "/var/lib/docker/volumes/{volume_name}"에 연결됨
- 형식
  - VOLUME 호스트디렉터리
- 예시
  - VOLUME /var/log

# Dockerfile 명령어

## USER

- 설명
  - 컨테이너의 기본 사용자를 설정하는 커맨드
  - 사용자가 생성된 상태여야 함
  - 사용자 생성을 위해서 RUN ["useradd", "사용자명"] 커맨드를 Dockerfile 내부에 작성해 둘 수 있음
- 형식
  - USER 사용자명
- 예시
  - USER teacher

# Dockerfile 명령어

## WORKDIR

- 설명
  - 컨테이너 상에서 작업 디렉터리를 이동하는 커맨드 (셸 커맨드 cd를 생각하면 됨)
  - WORKDIR 명령으로 작업 디렉터리를 이동하고 나면 그 이후에 등장하는
    모든 RUN, COPY, ADD, CMD, ENTRYPOINT 명령문은 이동한 디렉터리를 기준으로 실행함
  - 작업 디렉터리가 없으면 새로 생성함

- 형식
  - WORKDIR 이동경로

- 예시
  - WORKDIR /home/ubuntu

# Dockerfile 명령어

## RUN

- 설명
  - 이미지 빌드 과정에서 실행할 셸 커맨드를 등록하는 커맨드
  - 일반적으로 특정 패키지를 설치하기 위한 셸 커맨드 등록을 위해서 많이 사용함

- 형식
  - RUN 커맨드

- 예시
  - RUN apt update
  - RUN apt install -y curl
    또는
  - RUN apt update && apt install -y curl

# Dockerfile 명령어

## COPY

- 설명
  - 호스트의 파일이나 디렉터리를 도커 이미지 내부 디렉터리로 복사하는 커맨드
  - 도커 이미지 내부에 해당 디렉터리가 없는 경우에는 새로 생성함
- 형식
  - COPY 호스트경로 컨테이너경로
- 예시
  - COPY index.html /home/ubuntu/tomcat/webapps/ROOT/

# Dockerfile 명령어

## ADD

- 설명
  - 기본적으로 COPY 커맨드와 유사함
  - URL을 제시해서 외부에서 제공되는 파일을 복사할 수 있음
  - tar 파일인 경우 자동으로 압축을 풀어주는 기능을 가짐 (COPY 커맨드는 압축 해제 기능 없음)
- 형식
  - ADD 호스트경로 컨테이너경로
- 예시
  - ADD http://example.org/test.tar.gz /home/ubuntu/

# Dockerfile 명령어

## ENTRYPOINT

- 설명
  - 컨테이너 실행 시 항상 실행되어야 하는 커맨드를 등록할 때 사용하는 커맨드
  - 등록한 커맨드는 "docker run"시 실행되고 이 커맨드로 실행된 프로세스가 죽을 때 컨테이너도 함께 종료됨
- 형식
  - ENTRYPOINT ["커맨드", ["인자1", "인자2", ...]]
- 예시
  - ENTRYPOINT ["java", "-jar", "app.jar"]

# Dockerfile 명령어

## CMD

- 설명
  - 컨테이너를 실행할 때 디폴트로 사용할 커맨드를 등록할 때 사용하는 커맨드
  - ENTRYPOINT 명령과 함께 사용되는 경우에는 ENTRYPOINT 명령으로 지정된 커맨드에 넘길 인자들을 등록
- 형식
  - CMD [커맨드 [인자1, 인자2, ...]]
- 예시
  - CMD ["java", "-jar", "app.jar"]
    또는
  - ENTRYPOINT ["java"]
  - CMD ["-jar", "app.jar"]

# Dockerfile 명령어

## ENTRYPOINT와 CMD

- 설명
  - ENTRYPOINT 명령과 CMD 명령을 함께 사용하면 컨테이너를 실행할 때 인자의 유무에 따라 다른 실행을 유도할 수 있음
  - CMD 명령에 등록한 인자가 디폴트로 사용되지만 컨테이너 실행 시 인자가 전달되면 해당 인자가 사용됨

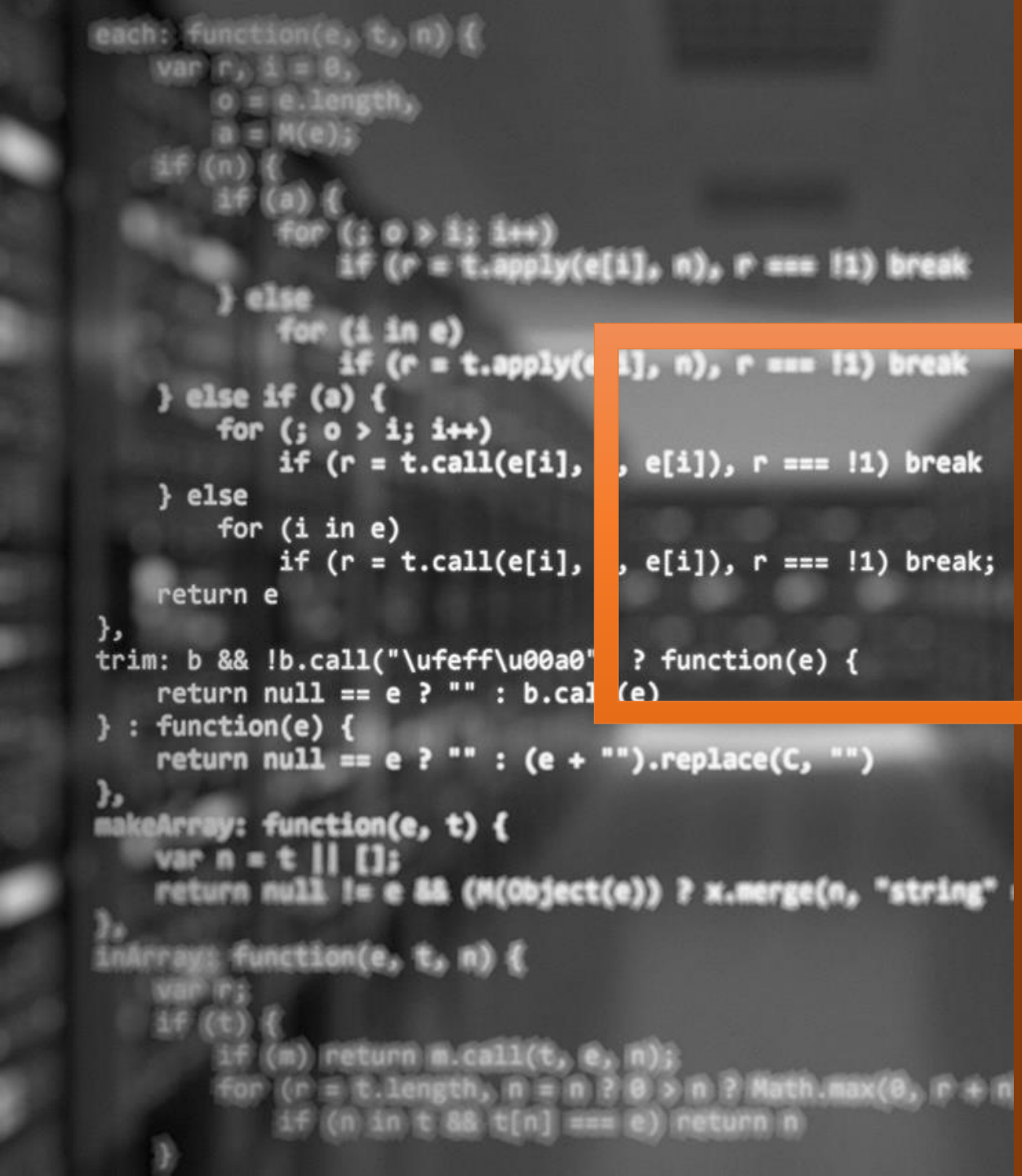
- Dockerfile

```
FROM ubuntu:latest
ENTRYPOINT ["echo"]
CMD ["Hello world"]
```

- docker run 커맨드 예시
  - $ docker run hello-image → "Hello world" 출력
  - $ docker run hello-image Hi → "Hi" 출력

# Dockerfile 명령어 유의사항

- 특징
  - 도커파일은 모든 명령어를 하나의 레이어(Layer)로 구성함
- 현상
  - 명령어가 많아지면 레이어(Layer)가 쌓여 메모리가 커지게 됨
- 해결
  - 방법1. 도커파일 명령어를 가급적 한 줄로 모아서 작성함
    - RUN apt update
    - RUN apt install -y openjdk
      대신
    - RUN apt update && apt install -y openjdk
  - 방법2. 마지막에 이미지 내부 임시 파일을 삭제하는 명령을 추가로 작성함
    - RUN apt update && apt install -y openjdk && rm -rf /var/lib/apt/lists/*



# 03. 도커 이미지 빌드 및 업로드

# 도커 이미지 빌드

- 도커 이미지 빌드
  - Dockerfile을 이용해서 도커 이미지를 생성하는 과정
  - docker image build 명령을 이용해 도커 이미지를 빌드할 수 있음

Dockerfile

Docker Image

# docker image build

## docker image build

- 설명
  - 도커파일을 이용해 도커 이미지를 만드는 명령
- 형식
  - docker image build [옵션] 도커파일경로
- 옵션

| 옵션 | 내용 |
|---|---|
| -t | 빌드할 도커 이미지의 이름과 태그를 설정함 |
| -f | Dokcerfile이 아닌 다른 이름을 사용한 경우 해당 도커파일명을 등록함 |
| --build-arg | 빌드 시 Dockerfile의 ARG에 전달할 인자와 값을 "인자=값" 형식으로 등록함 |

# docker image build

- 도커 이미지 빌드 예시

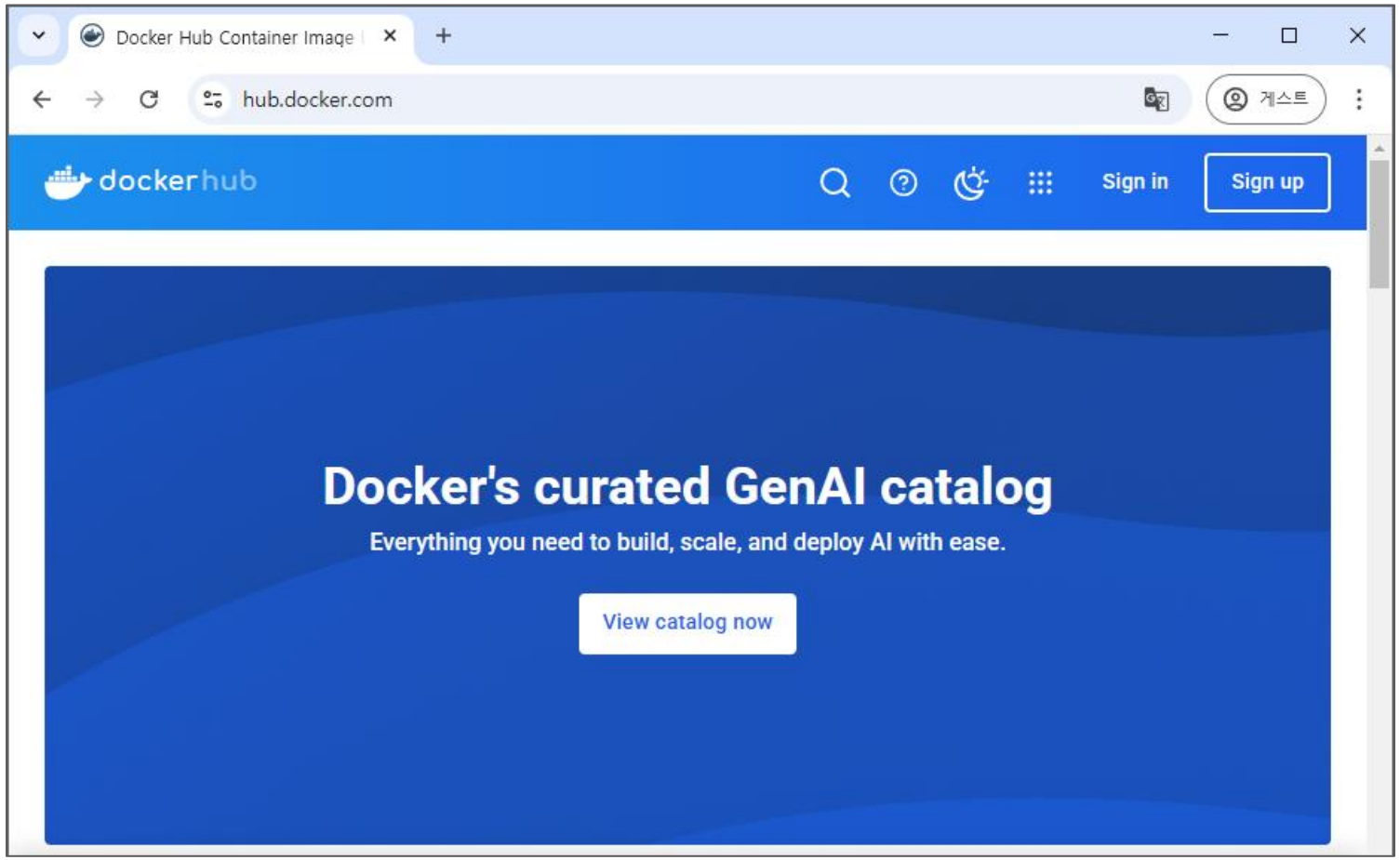
```
$ docker image build \
    -t 도커허브계정명/도커이미지명:태그 \
    --build-arg 변수명=값 \
    -f 도커파일명 \
    도커이미지경로
```

# Docker Hub

- 도커 허브
  - 개발자와 오픈소스 기여자가 생성하고자 하는 컨테이너의 이미지를 서로 공유할 수 있도록 구축된 사이트
  - 직접 생성한 이미지를 public 공개 저장소에 올려두면 다른 사람들도 해당 이미지를 내려 받아 사용할 수 있음
  - 팀이나 기업에서만 사용할 목적으로 private 저장소에 올려둘 수 있음

# Docker Hub

- 회원가입 및 로그인 필요
  - https://hub.docker.com/



# 도커 이미지 업로드

- **도커 이미지 업로드**
  - 도커 명령어를 이용해 도커 이미지 업로드 가능
  - 도커 허브에 로그인 후 도커 이미지 업로드 명령을 내림

- **명령**
  - $ docker login
  - $ docker image push 도커이미지명