애플리케이션 소스 코드에서 도커 이미지까지

최 혁

dockerfile로 빌드 툴 배포하기

개발팀이 같은 빌드 툴을 사용하기 위한 빌드 서버를 구축하면 비용이 많이 든다. 도커를 사용해서 해결해보자

```
# AS 파라미터로 이름을 붙일 수 있다
FROM diamol/base AS build-stage
# RUN instruction은 빌드 중에 컨테이너 안에서 명령을 실행한 다음 결과를 이미지 레이어에 저장
RUN echo 'Building...' > /build.txt

FROM diamol/base AS test-stage
# --from 인자로 호스트 컴퓨터 파일시스템이 아닌 빌드 단계의 파일시스템에 접근
COPY --from=build-stage /build.txt /build.txt
RUN echo 'Testing...' >> /build.txt

FROM diabol/base
COPY --from=test-stage /build.txt /build.txt
CMD cat /build.txt
```

dockerfile로 빌드 툴 배포하기

1. 1단계 빌드에서 텍스트 파일 생성
2. 2단계 빌드에서 1단계에서 생성한 텍스트 파일을 복사
3. 3단계 빌드에서 2단계에서 생성한 텍스트 파일을 복사

- 여러 빌드로 나뉘어진 빌드를 멀티 스테이지 빌드라 한다.
- 각 빌드 단계는 서로 격리돼 있다.(빌드 단계별 기반 이미지가 다르기 때문)
- 한 단계라도 명령이 실패하면 전체 빌드가 실패한다.

애플리케이션 빌드 실전 예제: 자바 소스 코드

```
FROM diamol/maven AS builder

WORKDIR /usr/src/iotd
COPY pom.xml .
# 메이븐이 실행돼 필요한 의존 모듈을 내려받는다.
RUN mvn -B dependency:go-offline

# 도커 빌드가 실행중인 디렉터리에 포함된 모든 파일과 서브 디렉터리를 현재 이미지 내 작업 디렉터리로 복사
COPY . .
RUN mvn package

# app
FROM diamol/openjdk
WORKDIR /app
COPY --from=builder /usr/src/iotd/target/iotd-service-0.1.0.jar

EXPOSE 80
ENTRYPOINT ["java", "-jar", "/app/iotd-service-0.1.0.jar"]
```

애플리케이션 빌드 실전 예제: 자바 소스 코드

- 컨테이너는 컨테이너가 실행될 때 부여되는 가상 네트워크 내 가상 IP를 통해 서로 통신한다.

```
# 네트워크 생성
docker network create nat

# 이미지를 실행하며 80번 포트를 800번 포트로 바인딩하여 호스트 컴퓨터를 통해 공개하고,
# nat 네트워크에 컨테이너 접속
docker container run --name iotd -d -p 800:80 --network nat image-of-the-day
```

장점

- 도커만 깔려있으면 이 애플리케이션은 어느 환경에서도 실행될 수 있다.
- 최종적으로 생성되는 애플리케이션 이미지에 빌드 도구는 포함되지 않는다.
    - 도커 파일에 정의된 빌드 중 마지막 단계의 콘텐츠만 포함되기 때문이다.
    - 만약 포함하고 싶다면 최종 단계에서 명시적으로 해당 콘텐츠를 복사해야 한다.

애플리케이션 빌드 실전 예제: Node.js 소스 코드

```
FROM diamol/node AS builder

WORKDIR /src
COPY src/package/json .

RUN npm install

# app
FROM diamol/node

EXPOSE 80
CMD ["node", "server.js"]

WORKDIR /app
COPY --from=builder /src/node_modules/ /app/node_modules/
COPY src/ .
```

애플리케이션 빌드 실전 예제: Go 소스 코드

```
FROM diamol/golang as builder

COPY main.go .
RUN go build -o /server

# app
FROM diamol/base

ENV IMAGE_API_URL="http://iotd/image" \
    ACCESS_API_URL="http://accesslog/access-log"
CMD ["/web/server"]

WORKDIR web
COPY index.html .
COPY --from=builder /server .
RUN chmod +x server
```

멀티 스테이지 dockerfile 스크립트 이해하기

장점

1. 표준화
    - 운영체제와 모든 도구들의 버전, 빌드 과정이 도커 컨테이너 내부에서 이루어지기에 서버 관리 부담을 줄일 수 있다.
2. 성능 향상
    - 각 이미지 레이어가 캐싱되기에 90% 이상 빌드 단계에서 시간을 절약할 수 있다.
3. Dockerfile 스크립트를 통해 최종 산출물인 이미지를 작게 유지할 수 있다.
    - 최종 산출물인 이미지에 불필요한 도구를 빼버릴 수 있다(빌드 환경)
    - 이미지의 크기를 줄여 애플리케이션의 시작 시간을 단축할 수 있다.

연습 문제

```
FROM diamol/golang

WORKDIR web
COPY index.html .
COPY main.go .

RUN go build -o /web/server
RUN chmod +x /web/server

CMD ["/web/server"]
ENV USER=sixeyed
EXPOSE 80
```

- Go로 구현한 도커파일 스크립트를 최적화하여 15MB로 크기를 줄여라.
- 현재 도커파일 스크립트에 포함된 HTML 내용을 수정하면 일곱 단계의 빌드가 재수행된다.
    ㄴ> 최적화하여 HTML 파일을 수정하더라도 재수행되는 빌드 단계가 한 단계 되도록 하라.

연습문제 풀이

```
FROM diamol/golang

COPY main.go .

RUN go build -o /web/server

FROM diamlo/base

ENV USER=sixeyed
EXPOSE 80
CMD ["/web/server"]
RUN chmod +x /web/server

WORKDIR web
COPY --from=builder /server .
COPY index.html .
```

명령어 정리

docker network create [네트워크이름]