개발작업이 이루어지는 원리

웹 서비스를 개발할 때, 먼저 개발용 컴퓨터에서 코딩한 후 테스트에서 이상이 없으면 서버에서 프로그래밍을 실행, 즉 '배포'를 한다.

이 때 개발용 컴퓨터와 서버 모두에 환경을 동일하게 맞춰줘야 한다. (불편함 1)

또한 개발용 컴퓨터에서 여러 서비스를 만들어야 한다면, 모든 서비스 간의 개발 환경을 일치시키거나, 환경 설정을 매번 복잡하게 바꿔줘야 한다. (불편함 2)

해결 방법

1. 가상환경(Virtualization)

내 OS 안에 또 다른 OS를 설치하여 환경을 각각 분리하는 것

개발 환경들을 확실히 분리해주긴 하지만, 각각의 가상환경에 컴퓨터 자원을 영구적으로 할당해야하기 때문에, 컴퓨터 성능과 자원의 손실이 크다.

2. Docker

도커의 컨테이너는 각각 분리된 환경이지만, 컴퓨터 자원을 서로 공유할 수 있기 때문에 컴퓨터 자원의 낭비가 크지 않다.

도커를 사용하는 개발자는 각 컨테이너들이 어떻게 설계되고, 지정한 곳에 어떻게 설치가 되며, 어떤 업무를 수행할지 하나하나 명시하기 때문에, 다른 컴퓨터 환경에서도 같은 환경을 구성할 수 있다.

기본 용어 정리

1. 컨테이너(Container)

컨테이너는 호스트 머신의 다른 프로세스와 독립된 하나의 프로세스를 의미합니다.

2. 이미지(Image)

이미지는 컨테이너를 찍어내기 위한 틀이며, 어플리케이션을 실행하기 위한 모든 파일시스템을 포함합니다.

3. 도커파일(Dockerfile)

도커파일은 명령어들로 이루어진 스크립트 파일이며, 이미지를 생성(빌드)하기 위해 사용됩니다.

Dockerfile을 사용하여 이미지와 컨테이너 만들기

먼저 다음과 같은 도커파일을 만들어봅니다.

```
FROM node:10-alpine                 # 사용할 이미지
WORKDIR /app                        # 작업 폴더 지정
COPY . .                            # 현재 폴더의 파일을 WORKDIR로 복사
RUN yarn install --production       # 명령어 실행 (새로운 이미지 생성)
CMD ["node", "/app/src/index.js"]   # 명령어 실행 (default 명령 설정)
```

이미지를 만들기 위해서 `docker build` 명령어를 사용합니다.

```
docker build -t docker-101 .
```

생성한 이미지를 통해 새로운 컨테이너를 만듭니다.

```
docker run -dp 3000:3000 docker-101
```

각 flag의 설명은 다음과 같습니다.

- `-d` detached mode의 사용, background에서 동작
- `-p 3000:3000` host의 3000번 포트를 컨테이너의 3000번 포트에 연결
- `docker-101` 이미지명

Docker Hub에 이미지 푸시하기

Docker Hub에 로그인 후, Repository를 생성합니다. 이후 3가지 단계(로그인, 태그, 푸시)를 따라서 이미지를 푸시할 수 있습니다.

1. 로그인하기

```
docker login -u {YOUR-USER-NAME}
```

2. 태그하기

```
docker tag {IMAGE-NAME} {YOUR-USER-NAME}/{REPO-NAME}
```

3. 푸시하기

```
docker push {YOUR-USER-NAME}/{REPO-NAME}
```

볼륨(Volume)

볼륨은 컨테이너의 파일시스템 경로를 호스트 머신과 연결하는 기능입니다. 볼륨은 크게 두 가지 타입이 있습니다.

Named Volume

named volume을 사용하면, 도커가 지정한 공간에 데이터를 저장하여, 컨테이너가 재시작되어도 데이터를 보존할 수 있습니다.

1. `docker volume create` 명령어를 사용하여 볼륨을 생성합니다.

```
docker volume create todo-db
```

2. 컨테이너를 실행할 때, `-v` 플래그를 사용하여 볼륨을 지정합니다.

```
docker run -dp 3000:3000 -v todo-db:/etc/todos docker-101
```

bind mount

bind mount를 사용하면, 우리는 호스트 머신의 mountpoint를 직접적으로 컨트롤할 수 있습니다.

1. -v, COPY의 차이

COPY는 RUN처럼 이미지를 생성하는 과정에서 해당 이미지 안에 특정 파일을 미리 넣어두는 것

volume은 CMD처럼, 컨테이너가 생성되어 실행될 때 그 내부의 폴더를 외부의 것과 연결하는 것

3. docker compose