Ch4. 간단한 어플을 실제로 배포해보기(개발 환경 부분)

생성일	@2022년 2월 2일 오후 4:10
태그	따라하며 배우는 도커와 CI환경

주제: 간단한 Nodejs 어플리케이션을 Docker로 배포하기

Issue1. Dockerfile 작성
Issue2. Package.json파일이 없다고 나오는 이유
Issue3. 생성한 이미지로 어플리케이션 실행 시 접근이 안 되는 이유
Issue4. 어플리케이션 소스 변경으로 재빌드할 때 효율적으로 하는 법
Issue5. Docker Volume

package.json

```
{
    "name": "docker_web_app",
    "version": "1.0.0",
    "description": "Node.js on Docker",
    "main": "server.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "dependencies": {
        "express": "^4.17.2"
    },
    "author": "",
    "license": "ISC"
}
```

server.js

```
const express = require('express');

const PORT = 8080;

//APP
const app = express();
app.get('/', (req, res) => {
    res.send("hi")
});

app.listen(PORT);
console.log("Server is running")
```

Dockerfile

```
FROM node:10

WORKDIR /usr/src/app

COPY package.json ./

RUN npm install

COPY ./ ./

CMD ["node", "server.js"]
```

Issue1. Dockerfile 작성

```
FROM node:10

RUN npm install

CMD ["node", "server.js"]
```

- Nodejs 앱을 도커 환경에서 실행하기 위해 이미지를 생성하여야 함 → DockerFile!
- 베이스 이미지는 alpine이 아닌 node 이미지를 사용
  - npm을 위한 파일이 필요하기에 npm이 들어있는 베이스 이미지인 node 이미지 사용
  - npm: Node.js로 만들어진 모듈을 웹에서 받아서 설치해주고 관리해주는 프로그램
- 노드 웹 서버를 작동시키기 위해 node + 엔트리 파일 입력

Issue2. Package.json파일이 없다고 나오는 이유

- npm install을 하여 어플리케이션에 필요한 종속성을 다운받으면 먼저 package.json을 보고 그곳에 명시된 종속성들을 다운받아서 설치
- 하지만 package.json이 컨테이너에 없기 때문에 **COPY를 이용해서 컨테이너 안으로 넣어주어야 한다.**
  - `COPY [복사할 파일] [복사될 경로]`
- 따라서 도커를 빌드하기 위해 필요한 모든 파일을 COPY를 통해 넣어주어야 한다.
  - `docker build -t [이미지이름]`

```
FROM node:10

COPY ./ ./

RUN npm install

CMD ["node", "server.js"]
```

```
FROM node:10

COPY ./ ./

RUN npm install

CMD ["node", "server.js"]
```

Issue3. 생성한 이미지로 어플리케이션 실행 시 접근이 안 되는 이유

- 어플리케이션을 실행할 때 네트워크도 로컬 네트워크에 있던 것을 컨테이너 내부에 있는 네트워크에 연결해야 한다.
- `docker run -p [로컬 포트]:[서버 포트] [이미지 이름]`

Issue4. 어플리케이션 소스 변경으로 재빌드할 때 효율적으로 하는 법

- 현재 노드 어플리케이션의 Root디렉토리에는 home, bin, dev 등의 파일들이 있다.
- 이 디렉토리에 대하여 workdir를 지정하지 않고 그냥 COPY를 하면 문제가 발생한다.
  - 원래 이미지에 있던 파일과 이름이 같다면 폴더가 중복이 되버려서 덮어씌워져버린다.
  - 모든 파일이 한 디렉토리에 들어가서 정리가 되지 않는다.
- 따라서 모든 어플리케이션을 위한 소스들은 WORK 디렉토리를 따로 만들어서 보관한다.

- `WORKDIR [디렉토리 경로]`

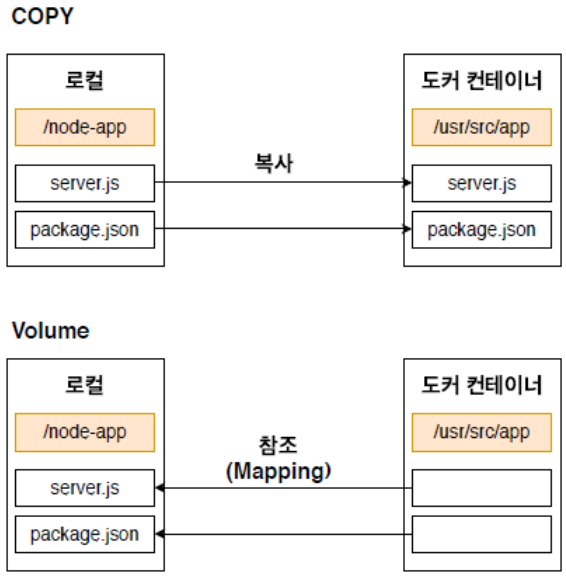
```
FROM node:10

WORKDIR /usr/src/app

COPY ./ ./

RUN npm install

CMD ["node", "server.js"]
```

- 뿐만 아니라 npm install 할 때 소스 코드에 조금의 변화만 생겨도 모듈 전체를 다시 다운받아야 하는 문제가 있다.
- 따라서 npm install 전에 package.json을 먼저 COPY해주어 소스 코드에 변화가 생길 때 모듈을 다시 받는 현상을 없앨 수 있다.

```
FROM node:10

WORKDIR /usr/src/app

COPY package.json ./

RUN npm install

COPY ./ ./

CMD ["node", "server.js"]
```

Issue5. Docker Volume

- 그럼에도 불구하고 소스를 변경할 때 마다 변경된 소스 부분은 COPY한 후 이미지를 다시 빌드를 해주고 컨테이너를 다시 실행하여야 변경된 소스가 반영이 된다.
- 이미지를 자주 빌드하는 것을 막기 위해 docker volume을 사용한다.



- 이렇게 volume을 이용해서 키면 빌드할 때 소스를 바꾸더라도 바꾼 코드가 적용이 된다.
- ex) `docker run -d -p 5000:8000 -v /usr/src/app/node_modules -v $(pwd):/usr/src/app jjongwoo27/nodejs`
  - 여기서 node_module은 호스트 디렉토리에 없기에 컨테이너에서 맵핑하지 않도록 하기 위해 명령어에 추가