8. 복잡한 어플을 실제로 배포해보기 (개발 환경 부분)

출처: 인프런 - 따라하면 배우는 도커와 CI환경 / 8강

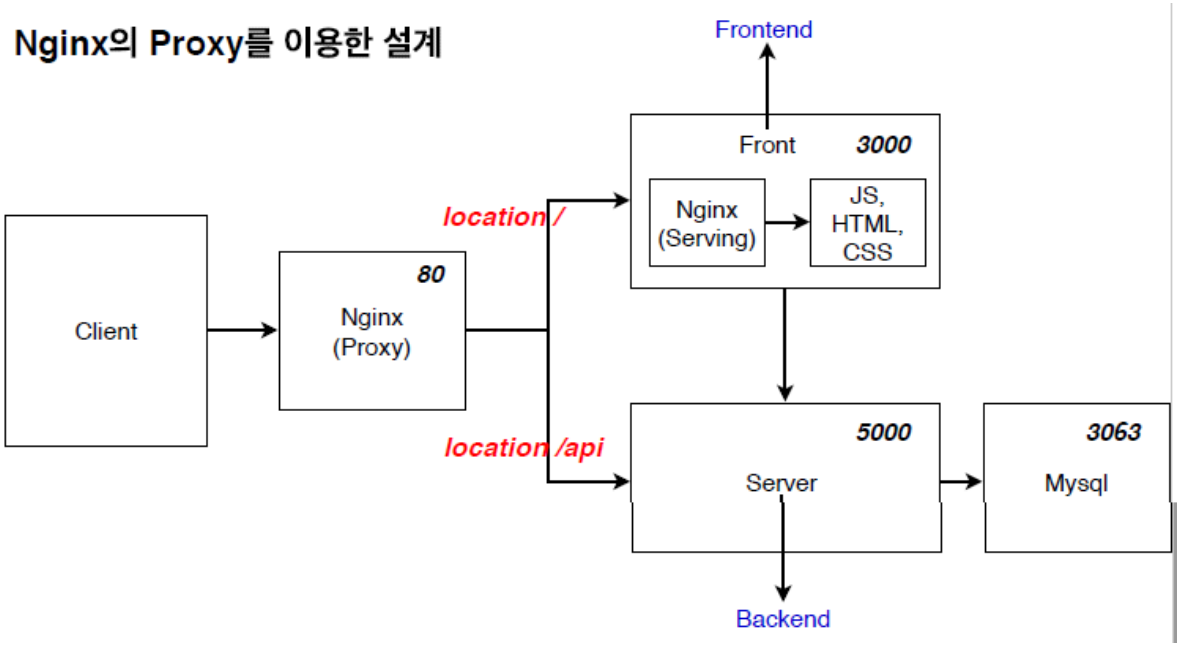
발제자: 이학림

이번 8강에서는 React, nodejs, nginx를 사용하는 어플을 만든 후, Docker compose까지 작성해보겠습니다. React, Nodejs는 이전 강의들과 유사하므로 차이점에 대해서만 다루겠습니다.

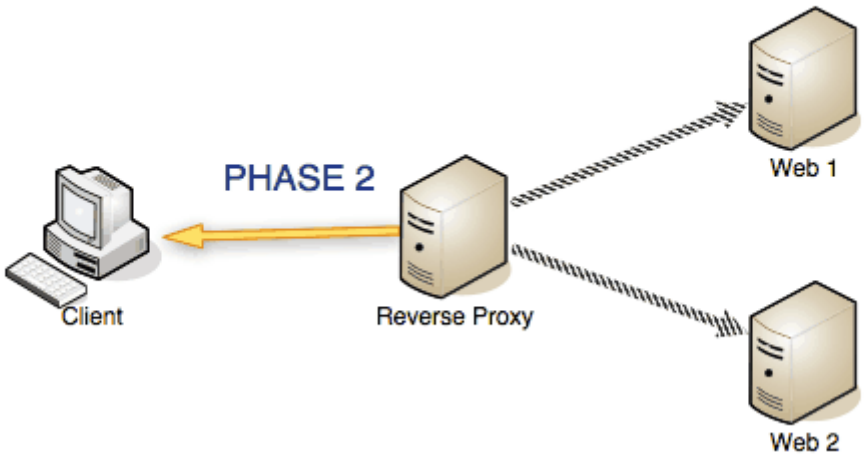
만들 어플 최종 결과

- 사용자가 단어를 입력하면 실시간으로 위에 출력되게 하는 어플.
- 웹을 닫았다켜도 이전의 결과를 계속해서 보여준다.

Nginx를 이용한 어플 설계



Nginx란?



- 동시접속 처리에 특화된 웹 서버 프로그램
- 기존에 널리 사용되던 Apache와 비교했을때 더 단순하고, 성능이 좋다.
- 정적파일 (HTML, CSS,, JAVASCRIPT, 이미지)를 웹 브라우저에 전송하는 역할
- 클라이언트의 요청을 Nginx(Reverse Proxy의 역할)가 받은 후, 진짜 서버에 요청을 보낸다. 요청을 배분하는 작업을 진행(비동기 방식 채택)

React.js

기본코드

```
import React, { useState, useEffect } from 'react'
import axios from 'axios';
import logo from './logo.svg';
import './App.css';

function App() {

  const [lists, setLists] = useState([])
  const [value, setValue] = useState("")

  useEffect(() => {
    //여기서 데이터베이스에 있는 값을 가져온다.

  }, [])
```

운영 도커파일

```
FROM node:alpine as builder
WORKDIR /app
COPY ./package.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx
EXPOSE 3000
COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/build  /usr/share/nginx/html
```

- 개발도커 파일과 달리, nginx에 정적파일을 제공해야하므로 default.conf를 제공해준다

개발 도커파일

```
FROM node:alpine

WORKDIR /app

COPY package.json ./

RUN npm install

COPY ./ ./

CMD [ "npm", "run", "start" ]
```

default.conf

```
server {
    listen 3000;

    location / {

        root /usr/share/nginx/html;

        index index.html index.html;

        try_files $uri  $uri/ /index.html;

    }
}
```

- nginx에 제공할 정적파일들을 담은 파일

Node js

기본코드

```
//필요한 모듈들을 가져오기
const express = require('express');
const bodyParser = require('body-parser');

// Express 서버를 생성
const app = express();

// json 형태로 오는 요청의 본문을 해석해줄수있게 등록
app.use(bodyParser.json());

// Express 서버 포트 5000에서 시작
app.listen(5000, () => {
  console.log('어플리케이션이 서버 5000번 포트에서 되었습니다.')
});
```

운영 도커파일

```
FROM node:alpine

WORKDIR /app

COPY ./package.json ./

RUN npm install

COPY . .

CMD ["npm", "run", "start"]
```

- 굳이 변경사항을 즉시 반영하는 기능이 필요없으므로 npm run start 사용

개발 도커파일

```
FROM node:alpine

WORKDIR /app

COPY ./package.json ./

RUN npm install

COPY . .

CMD ["npm", "run", "dev"]
```

- 개발 단계에서는 변경사항의 반영이 즉시 이루어져야하므로, nodemon 라이브러리를 사용하게 끔 dev 명령어 사용

mysql

기본코드

```
DROP DATABASE IF EXISTS myapp;

CREATE DATABASE myapp;
USE myapp;

CREATE TABLE lists (
  id INTEGER AUTO_INCREMENT,
  value TEXT,
  PRIMARY KEY (id)
);
```

my.conf

```
[mysqld]
character-set-server=utf8

[mysql]
default-character-set=utf8

[client]
default-character-set=utf8
```

- mysql에서 한글이 깨질수 있으므로 인코딩 문제를 해결해주는 파일

Nginx

default.config

도커파일

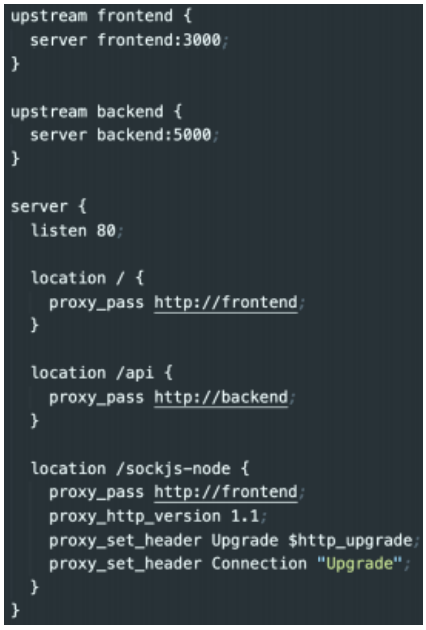
```
1  FROM mysql:5.7
2
3  ADD ./my.cnf /etc/mysql/conf.d/my.cnf
```

- mysql 이미지를 호출
- 인코딩 문제를 방지하기 위해 my.conf 파일을 제공

dockerfile

```
FROM nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

- nginx 이미지 호출
- proxy기능을 위해 default.config 파일을 제공



```
upstream frontend {
  server frontend:3000;
}

upstream backend {
  server backend:5000;
}

server {
  listen 80;

  location / {
    proxy_pass http://frontend;
  }

  location /api {
    proxy_pass http://backend;
  }

  location /sockjs-node {
    proxy_pass http://frontend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
  }
}
```

- nginx의 proxy기능을 위한 파일

Docker compose

```yaml
version: "3"
services:
  frontend:
    build:
      dockerfile: Dockerfile.dev
      context: ./frontend
    volumes:
      - /app/node_modules
      - ./frontend:/app
    stdin_open: true

  nginx:
    restart: always
    build:
      dockerfile: Dockerfile
      context: ./nginx
    ports:
      - "3000:80"

  backend:
    build:
      dockerfile: Dockerfile.dev
      context: ./backend
    container_name: app_backend
```

```
    volumes:
      - /app/node_modules
      - ./backend:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_ROOT_PASSWORD: johnahn777
      MYSQL_DATABASE: myapp
      MYSQL_PORT: 3306




  mysql:
    build: ./mysql
    restart: unless-stopped
    container_name: app_mysql
    ports:
      - "3306:3306"
    volumes:
      - ./mysql/mysql_data:/var/lib/mysql
      - ./mysql/sqls/:/docker-entrypoint-initdb.d/
    environment:
      MYSQL_ROOT_PASSWORD: "qweqweqwe"
      MYSQL_DATABASE: "myapp"
```

- front/backend, sql, nginx 에 대하여 docker compose 파일을 작성합니다
- 방법은 이전강의와 동일합니다