# Nginx 이용해 프로젝트 배포하기

**교육장에서 제공되는 교육자료는 외부 반출 금지입니다. 블로그 업로드, 요약하여 게시, 타인에게 공유 등의 행위를 하지 말아주세요.**



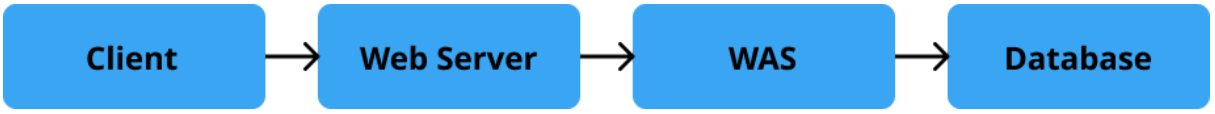
## Step1️⃣ - Web Sever 와 WAS 이해하기

`Web Server(웹 서버)`란 클라이언트의 요청을 받아 주로 `정적 파일`(파일, 이미지)을 전달하는 서버입니다. 여기서 정적 파일(static file)이란 javascript, css, image 파일과 같이 “변경되지 않는 파일”을 의미합니다.

대표적인 웹 서버의 종류로는 Apache, Nginx, IIS 가 있습니다.

`WAS(Web Application Server)`란 클라이언트의 요청에 대해 `동적인 데이터를 처리`할 수 있는 서버입니다. 물론 WAS가 정적 파일도 처리할 수는 있습니다. WAS는 데이터를 데이터베이스에 저장하여 서버를 구축합니다. (참고! 실무에서는 WAS 를 Web Server 라고 부르기도 합니다. 둘을 명확히 구분하고자 할 때 WAS라고 지칭합니다.)

대표적인 WAS 의 종류로는 Tomcat, Websphere, JEUS 등이 있습니다. 우리가 배운 Node.js 역시도 WAS로 볼 수 있습니다.




▼ ***Q. WAS로도 정적 파일 처리가 가능하다면 Web Server는 왜 사용하나요?***

WAS는 로그인, 회원가입 등 데이터를 다루는 해야할 작업이 많습니다. 따라서 정적 파일을 보여주는 일은 Web Server가 담당하고, WAS는 데이터베이스 접근 및 데이터 처리 로직을 담당하여 서버 부하를 방지해야 합니다.

그렇다면 Web Server 종류 중 가볍고 높은 성능을 보여주는 `Nginx`를 사용하여 **React 프로젝트를 배포**해보겠습니다.

# Step2️⃣ - 프로젝트 코드 변경하기

## .env.production 파일 생성하기

react 배포 환경에서 사용될 env 파일을 생성하고, 환경 변수를 작성해주세요. 이 때 변수의 이름에는 반드시 `REACT_APP_` 이 들어가야 합니다.

.env.production 파일은 배포 환경에서 사용됩니다. 즉, `npm run build` 명령어를 입력한 경우 자동으로 .env.production 파일의 환경 변수를 사용합니다.

```
REACT_APP_DB_HOST="http://ip_address:backend_port_num"
REACT_APP_MODE="product"

ex)
REACT_APP_DB_HOST="http://3.36.66.93:8080"
REACT_APP_MODE="product"
```

### App 컴포넌트에서 axios 요청 주소 변경하기

```
await axios.get(`${process.env.REACT_APP_DB_HOST}/api/todos`)
await axios.post(
      `${process.env.REACT_APP_DB_HOST}/api/todo`,
      newItem,
    );
await axios.patch(
      `${process.env.REACT_APP_DB_HOST}/api/todo/${targetItem
      targetItem,
    );
await axios.delete(
      `${process.env.REACT_APP_DB_HOST}/api/todo/${targetItem
    );
```

#### 👩‍🔬 [참고] React의 환경 변수

리액트에서는 실행 환경에 따라 자동으로 환경변수를 지정합니다. 여러분들이 해야할 것은 환경변수 파일을 잘 만들어주는 것 입니다. 환경변수 파일 명은 아래를 참고합니다.

- `.env`: 기본 환경 변수 설정.
- `.env.local`: 로컬 환경 설정.
- `.env.development`, `.env.production`: 개발, 배포 환경 별 설정.
- `.env.development.local`, `.env.production.local`: 환경 별 설정의 로컬 오버라이드.

**리액트 환경 변수 우선 순위**

1. `npm start` 명령어 이용 시 env 적용 우선순위 (개발 환경)

   **.env.development.local > .env.development > .env.local > .env**

2. `npm run build` 명령어 이용 시 env 적용 우선순위 (배포 환경)

   **.env.production.local > .env.production > .env.local > .env**

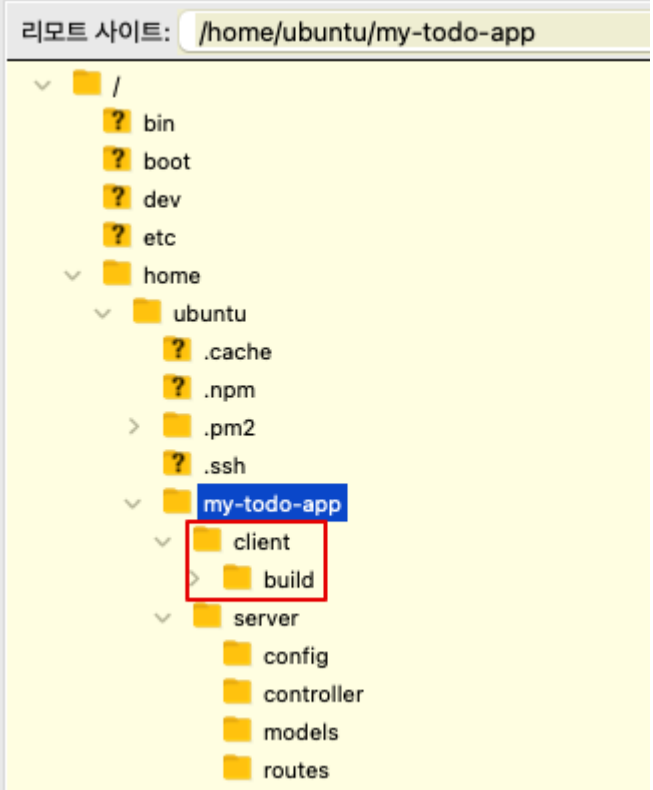
# Step3️⃣ - React 프로젝트 빌드하기

### React 프로젝트 빌드

```
npm run build
# build 폴더가 생성됨 (1~2분 소요)
# build 폴더에 배포에 필요한 모든 파일이 있음. 배포 용량을 줄이기 위해 공
```

### 빌드 폴더를 클라우드 서버에 업로드

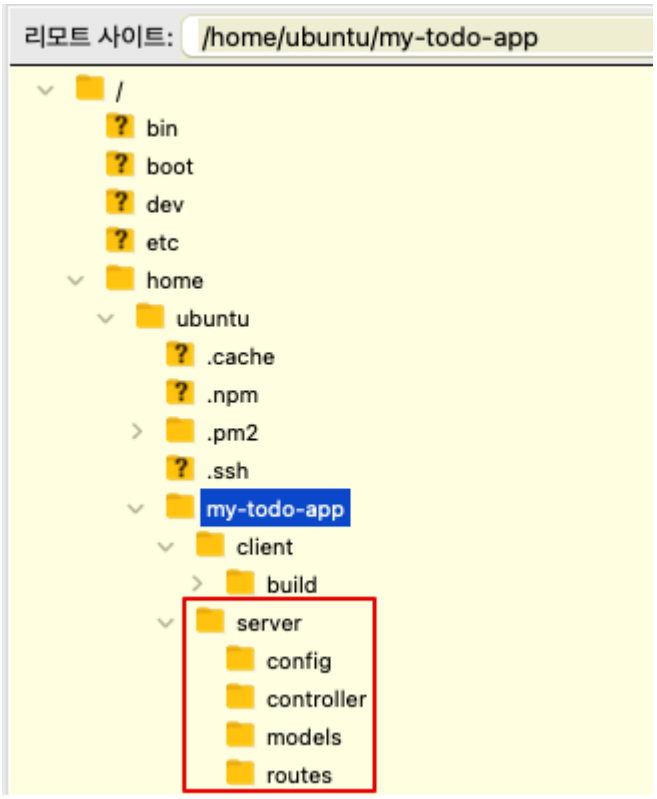
~ (home, /home/ubuntu) 경로에 프로젝트명 폴더(my-todo-app)를 생성하고, client 폴더 아래에 build 폴더를 업로드 해주세요.




# Step4️⃣ - Express 프로젝트 배포하기

### server/ 폴더 업로드

~ (home, /home/ubuntu) 경로에 생성한 프로젝트명 폴더(my-todo-app)에 server 폴더를 업로드 해주세요.




## MySQL 접속 후 데이터베이스 및 테이블 생성

클라우드 서버의 MySQL 접속 후 데이터베이스 및 테이블을 생성해 주세요!

(참! MySQL 은 외부에서 root 유저로의 접근을 허용하지 않기 때문에 새로운 유저 생성 후 권한 부여도 해주셔야겠죠?)

```
-- 데이터베이스 생성
create database codingon default character set utf8 collate u

-- 데이터베이스 선택
USE codingon;

-- 테이블 생성
DROP TABLE IF EXISTS todo;
CREATE TABLE todo (
  id INT NOT NULL PRIMARY KEY auto_increment,
  title VARCHAR(100) NOT NULL,
  done BOOLEAN NOT NULL DEFAULT false
);
```

## pm2 실행

```
npm install pm2 # pm2 모듈 설치 (이미 설치되어 있다면 생략 가능)

pm2 start app.js # status가 online 상태라면 실행중
pm2 monit # 프로세스 감시
pm2 list # 프로세스 목록 확인

pm2 stop [name] # name으로 프로세스 중단 (status가 stopped로 변경)
pm2 stop [id] # id로 프로세스 중단
pm2 kill # pm2 자체를 종료

pm2 log # 로그 확인 (재시동 자동화시 에러 메세지 확인시 유용)
```

# Step5️⃣ - Nginx 설치하기

Putty(Windows), Terminal(MacOS)를 이용해 클라우드 서버(AWS EC2, Naver Cloud Platform 서버 등)에 접속해주세요.

## Ubuntu 에 Nginx 설치하기

```
sudo apt-get update # 설치 가능한 패키지 리스트 업데이트 (이전에 작성한
sudo apt-get install nginx
```

설치가 완료되면 /etc/nginx 경로에 파일들이 생성됩니다.

## Nginx 설정하기

우리가 만드는 설정과 겹칠 수 있기에 원본 파일을 다른 경로에 복원 후, default 기본 설정 파일을 삭제하겠습니다.

```
# 원본 복원
sudo cp -r /etc/nginx/sites-available/ /etc/nginx/sites-avail
sudo cp -r /etc/nginx/sites-enabled/ /etc/nginx/sites-enabled

# default 기본 설정 파일 삭제
sudo rm /etc/nginx/sites-available/default
sudo rm /etc/nginx/sites-enabled/default
```

## 프로젝트 설정 파일 생성

여러분 프로젝트의 build 결과물을 배포하기 위한 설정 파일(.conf)을 생성하겠습니다.

```
sudo vi /etc/nginx/sites-available/프로젝트명.conf
ex) sudo vi /etc/nginx/sites-available/my-todo-app.conf
```

## .conf 파일 수정하기

vim 사용법은 첨부한 링크를 참고해 빠르게 익히고 오세요! 직전에 생성한 설정파일(.conf)을 아래와 같이 수정해주세요.

```
server {
        listen 80;
        location / {
                root build_파일_업로드_경로;
                index index.html index.htm;
                try_files $uri /index.html;
        }
}

ex)
server {
        listen 80;
        location / {
                root /home/ubuntu/my-todo-app/client/build;
                index index.html index.htm;
                try_files $uri /index.html;
        }
}
```

## 실행할 수 있도록 링크 파일 만들기

```
sudo ln -s /etc/nginx/sites-available/프로젝트명.conf /etc/ngin
ex) sudo ln -s /etc/nginx/sites-available/my-todo-app.conf /e
```

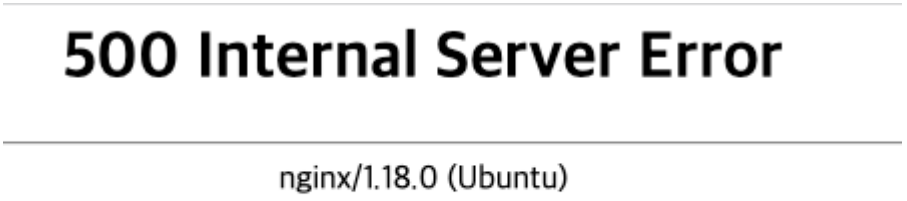
## Nginx 재가동

```
sudo systemctl stop nginx # 웹 서버 중단
sudo systemctl start nginx # 웹 서버 가동
sudo systemctl status nginx # 웹 서버 동작 상태 확인
```

# Step6️⃣ - 배포 링크 접속하기

브라우저에서 클라우드 서버의 http://ip-address:80(http://ip-address와 동일)로 접속한다면, 아래와 같이 500 에러가 발생합니다!

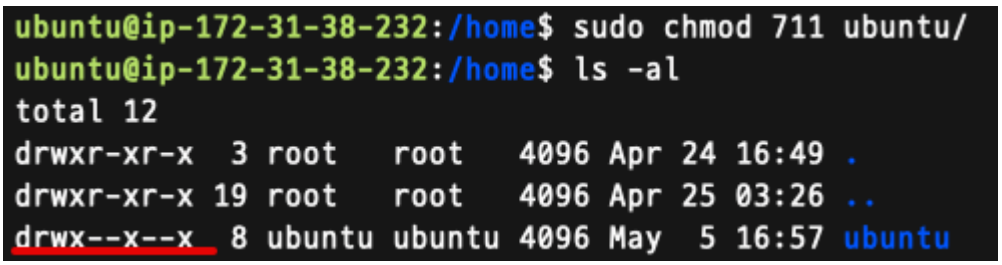
여기까지 왔다면 설정파일(.conf)을 올바르게 읽어낸 것입니다. 다만, AWS EC2에서는 /home/ubuntu/프로젝트_경로/build 디렉토리까지 접근할 때, ubuntu 외부 에서 실행(x) 권한이 없어서 발생하는 문제입니다.




## 권한 수정하기

현재 home/ 디렉토리 아래의 유저명(ubuntu/) 폴더의 권한이 750 이므로 이를 711로 변경해주겠습니다.

```
cd /home # home 디렉토리 이동
ls -al # home 디렉토리 목록 자세히 보기 (ubuntu/ 폴더의 권한이 750임)
sudo chmod 711 ubuntu/ # ubuntu 폴더 권한 711로 변경
ls -al # ubunbu 디렉토리 권한 변경 확인 가능
```



이제 다시 브라우저에서 ip 주소로 접속해 볼까요? 축하합니다! 무사히 프로젝트 배포를 마쳤습니다 🥳