

# 영화 추천 프론트엔드

## 동적 프록시 (Dynamic Proxy) 소개

동적 프록시는 프론트엔드 개발에서 API 요청을 효율적으로 관리하고 보안을 강화하는 중요한 기술입니다. 영화 추천 서비스에서는 사용자의 요청을 백엔드 서버로 안전하게 전달하고, 캐싱과 로드밸런싱을 통해 더 나은 성능을 제공하는 역할을 합니다.

# 1.동적 Proxy

동적 Proxy는 React 애플리케이션의 개발 환경에서 API 요청을 효율적으로 처리하기 위한 중요한 설정입니다. 프론트엔드와 API 백엔드 간의 원활한 통신을 가능하게 하며, CORS(Cross-Origin Resource Sharing) 이슈를 해결하는데 도움을 줍니다.

이는 특히 프론트엔드 개발 과정에서 백엔드 서버와의 통신을 단순화하고, 로컬 개발 환경에서의 API 테스트를 더욱 효율적으로 만들어줍니다.

## 동적 Proxy란 무엇인가?

동적 Proxy는 React 애플리케이션이 백엔드 서버와 통신할 때, 보안 문제를 해결하고 요청을 효율적으로 전달하기 위해 사용됩니다. React는 프론트엔드 애플리케이션으로, 데이터를 처리하기 위해 백엔드 서버와 연결이 필요합니다. 이때, package.json 파일에 Proxy 설정을 추가하면 React 개발 서버가 자동으로 요청을 백엔드 서버로 전달합니다.

그러나 package.json에 백엔드 서버의 IP 주소를 노출할 경우, 외부에 민감한 정보가 노출될 위험이 있습니다. 이를 방지하기 위해 동적 Proxy를 활용하면, **애플리케이션 실행 시점**에 Proxy 설정을 입력하여 백엔드 서버의 IP를 숨길 수 있습니다.

## 동적 Proxy의 작동 방식

- package.json에 백엔드 IP를 하드코딩하지 않고, 실행 시점에 환경변수로 Proxy 주소를 전달합니다.
- 이를 통해 백엔드 서버의 정보를 숨길 수 있어 보안이 강화됩니다.

## 2.nginx

Nginx는 Igor Sysoev가 개발한 오픈 소스 웹 서버 소프트웨어입니다

## Nginx란 무엇인가?

Nginx는 React 애플리케이션의 정적 파일(HTML, CSS, JavaScript)을 효율적으로 제공하고, 백엔드 서버와의 요청을 관리할 수 있도록 도와주는 웹 서버입니다. React 애플리케이션은 빌드된 정적 파일로 구성되며, 이러한 파일들을 브라우저에 제공하는 역할을 Nginx가 수행합니다. 또한, 동적 Proxy와의 결합을 통해 API 요청을 백엔드 서버로 전달하는 작업도 처리할 수 있습니다.

## Nginx의 주요 역할

1. **React 정적 파일 제공:** 빌드된 React 애플리케이션 파일을 브라우저에 빠르게 전달합니다.
2. **동적 Proxy와의 결합:** API 요청을 백엔드 서버로 전달하여 프론트엔드와 백엔드 간 통신을 원활하게 만듭니다.
3. **80번 포트를 통한 서비스 제공:** Nginx는 기본적으로 80번 포트에서 작동하며, 사용자가 브라우저를 통해 애플리케이션에 접근할 수 있도록 설정됩니다.

## React 프로젝트에서의 동작 흐름

React 프로젝트에서 동적 Proxy와 Nginx를 사용하는 과정을 단계적으로 설명하면 다음과 같습니다.

### 1. 동적 Proxy 사용 전

- package.json에 Proxy를 다음과 같이 설정합니다:

```
json                                                                    Copy code
{
  "proxy": "http://123.456.789.0:5000"
}
```

### 동적 Proxy 사용 후

- 애플리케이션 실행 시 다음과 같이 Docker 환경변수를 사용하여 Proxy 설정을 전달합니다

```
bash                                                                    Copy code

docker run -e BACKEND_URL="http://123.456.789.0:8080" my_app
```

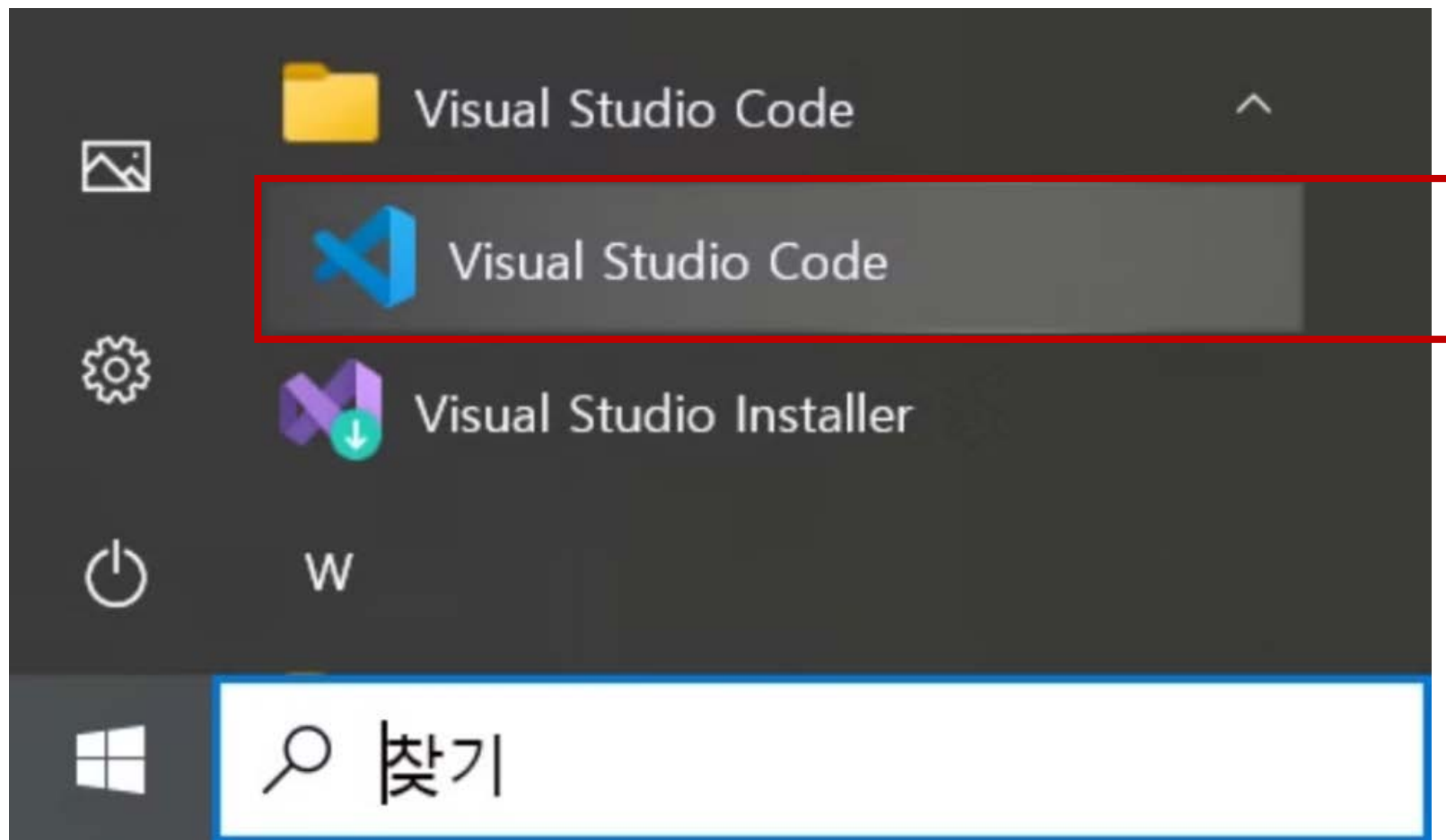
- 실행 시점에 Proxy를 설정함으로써 IP 주소가 코드에 하드코딩되지 않아 보안이 강화됩니다.

## 4. package.json 수정

보안 강화를 위해 package.json 파일에서 백엔드 URL 포트를 제거합니다. 이는 민감한 정보가 소스 코드에 직접 노출되는 것을 방지하고, 배포 환경에 따라 유연하게 설정을 변경할 수 있도록 합니다. 이러한 수정은 프로젝트의 보안성을 향상시키고 환경 변수를 통한 동적 설정을 가능하게 합니다.

# package.json 수정

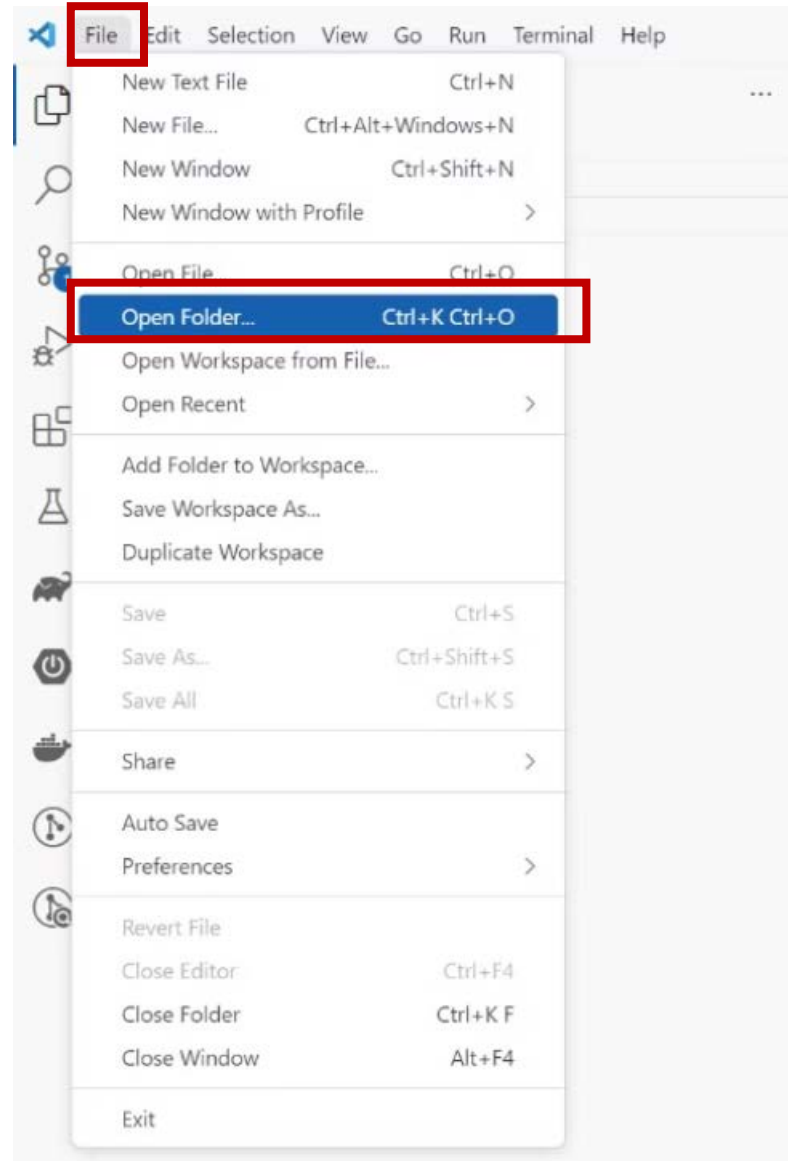
프로젝트를 GitHub로 Push 하기 위해서 Visual Studio Code를 실행 합니다





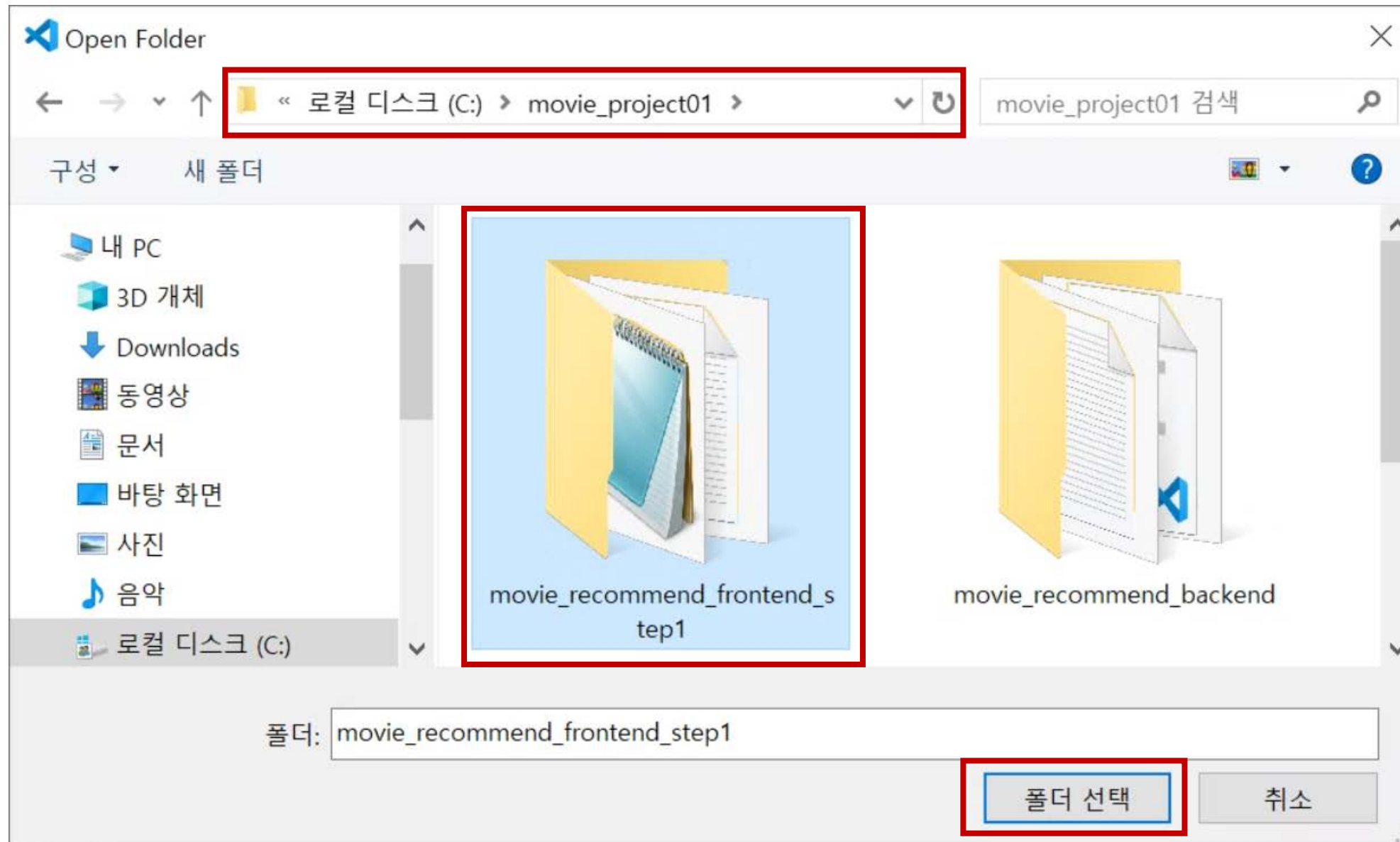
# package.json 수정

Open Folder를 선택 합니다



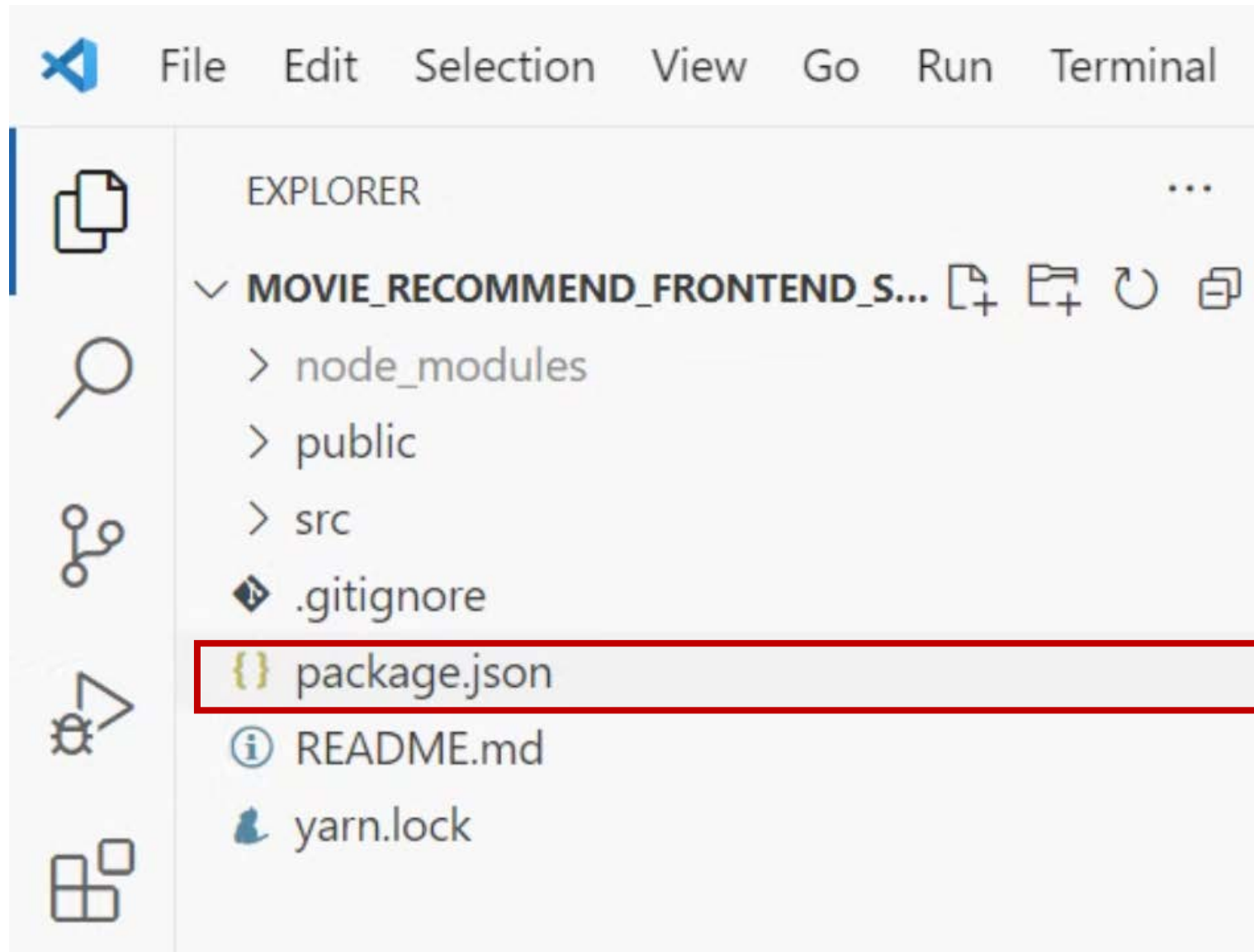
# package.json 수정

c:\mvie\_project01\movie\_recomment\_frontend\_step1 를 선택합니다



# package.json 수정

package.json 을 선택 합니다



# package.json 수정

proxy 설정을 삭제 합니다

```
package.json
package.json > ...

1 {
2   "proxy": "http://localhost:8080",
3
4   "name": "movie_recommend_frontend",
5   "version": "0.1.0",
6   "private": true,
7   "dependencies": {
8     "@testing-library/jest-dom": "^5.14.1",
9     "@testing-library/react": "^13.0.0",
10    "@testing-library/user-event": "^13.2.1",
11    "axios": "^1.7.9",
12    "react": "^19.0.0",
13    "react-dom": "^19.0.0",
14    "react-scripts": "5.0.1",
15    "web-vitals": "^2.1.0"
16  },
```

← 삭제 합니다

백엔드 아이피와 포트 삭제

Ctrl+S 눌러서 저장

## 5. Dockerfile 작성

Dockerfile은 React 애플리케이션을 실행 가능한 컨테이너로 변환하는 설정 파일입니다. 이 파일을 통해 React 애플리케이션을 빌드하고, nginx 웹 서버에 효율적으로 배포할 수 있습니다. 최적화된 프로덕션 환경을 구성하여 안정적인 서비스 운영이 가능해집니다.

# Dockerfile 작성

파일 URL : [https://raw.githubusercontent.com/gyeongnamit/movie\\_recommend\\_frontend\\_step2/refs/heads/main/Dockerfile](https://raw.githubusercontent.com/gyeongnamit/movie_recommend_frontend_step2/refs/heads/main/Dockerfile) 에 접속 합니다

```
# 파일명: Dockerfile
# Docker 이미지를 생성하는 설정 파일입니다.

# 1단계: React 애플리케이션 빌드
# Node.js 16 버전의 이미지 사용
# - Node.js는 JavaScript 실행 환경으로, React 애플리케이션을 빌드하는 데 필요합니다.
FROM node:16 AS builder

# 작업 디렉터리를 /app으로 설정
# - 모든 명령어는 /app 디렉터리에서 실행됩니다.
WORKDIR /app

# 의존성 설치
# - 프로젝트의 의존성 목록(package.json과 yarn.lock)을 복사합니다.
COPY package.json yarn.lock ./

# 필요한 패키지 설치 및 axios 추가
# - yarn install: package.json에 정의된 패키지를 설치합니다.
# - yarn add axios: axios(HTTP 요청 라이브러리)를 추가로 설치합니다.
RUN yarn install && yarn add axios

# 현재 디렉터리의 모든 파일을 컨테이너로 복사
# - 소스 코드와 구성 파일을 복사합니다.
COPY . .

# React 애플리케이션 빌드
# - 소스를 압축 및 최적화하여 배포 가능한 상태로 만듭니다.
RUN yarn build

# 2단계: nginx로 배포 설정
# - nginx는 웹 서버로, React 빌드 결과물을 제공하는 역할을 합니다.
FROM nginx:1.23-alpine

# nginx 템플릿 파일을 위한 작업 디렉터리 설정
# - nginx 설정 파일(template)을 저장할 디렉터리를 작업 디렉터리로 설정합니다.
WORKDIR /etc/nginx/templates

# 패키지 설치 및 설정
# - nginx 구동을 위한 추가 패키지 설치
RUN apk update && apk add --no-cache bash gettext && rm -rf /var/cache/apk/*
# bash : 쉘 스크립트 실행을 위한 패키지
# gettext : envsubst 명령어를 제공하는 패키지로, 환경 변수 치환을 지원합니다.

# 빌드한 파일 복사
# - React 애플리케이션의 빌드 결과물을 nginx의 기본 경로에 복사합니다.
COPY --from=builder /app/build /usr/share/nginx/html

# nginx 설정 템플릿 복사
# - 환경 변수 치환을 지원하는 템플릿 파일을 복사합니다.
COPY nginx.conf.template /etc/nginx/templates/nginx.conf.template

# 시작 스크립트 복사
# - nginx 설정을 적용하고 서버를 시작하는 스크립트를 복사합니다.
COPY entrypoint.sh /entrypoint.sh

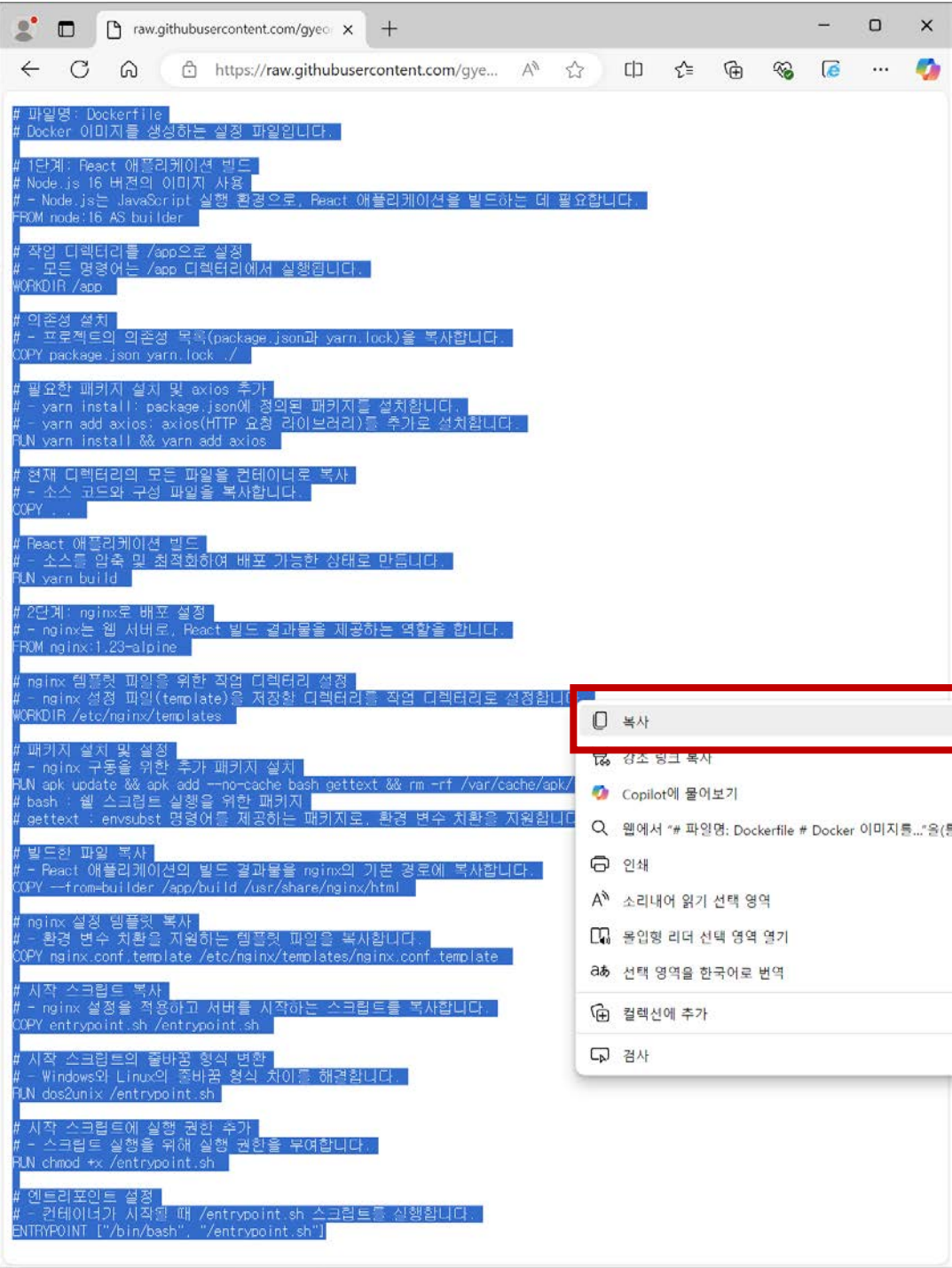
# 시작 스크립트의 줄바꿈 형식 변환
# - Windows와 Linux의 줄바꿈 형식 차이를 해결합니다.
RUN dos2unix /entrypoint.sh

# 시작 스크립트에 실행 권한 추가
# - 스크립트 실행을 위해 실행 권한을 부여합니다.
RUN chmod +x /entrypoint.sh

# 엔트리포인트 설정
# - 컨테이너가 시작될 때 /entrypoint.sh 스크립트를 실행합니다.
ENTRYPOINT ["/bin/bash", "/entrypoint.sh"]
```

# Dockerfile 작성

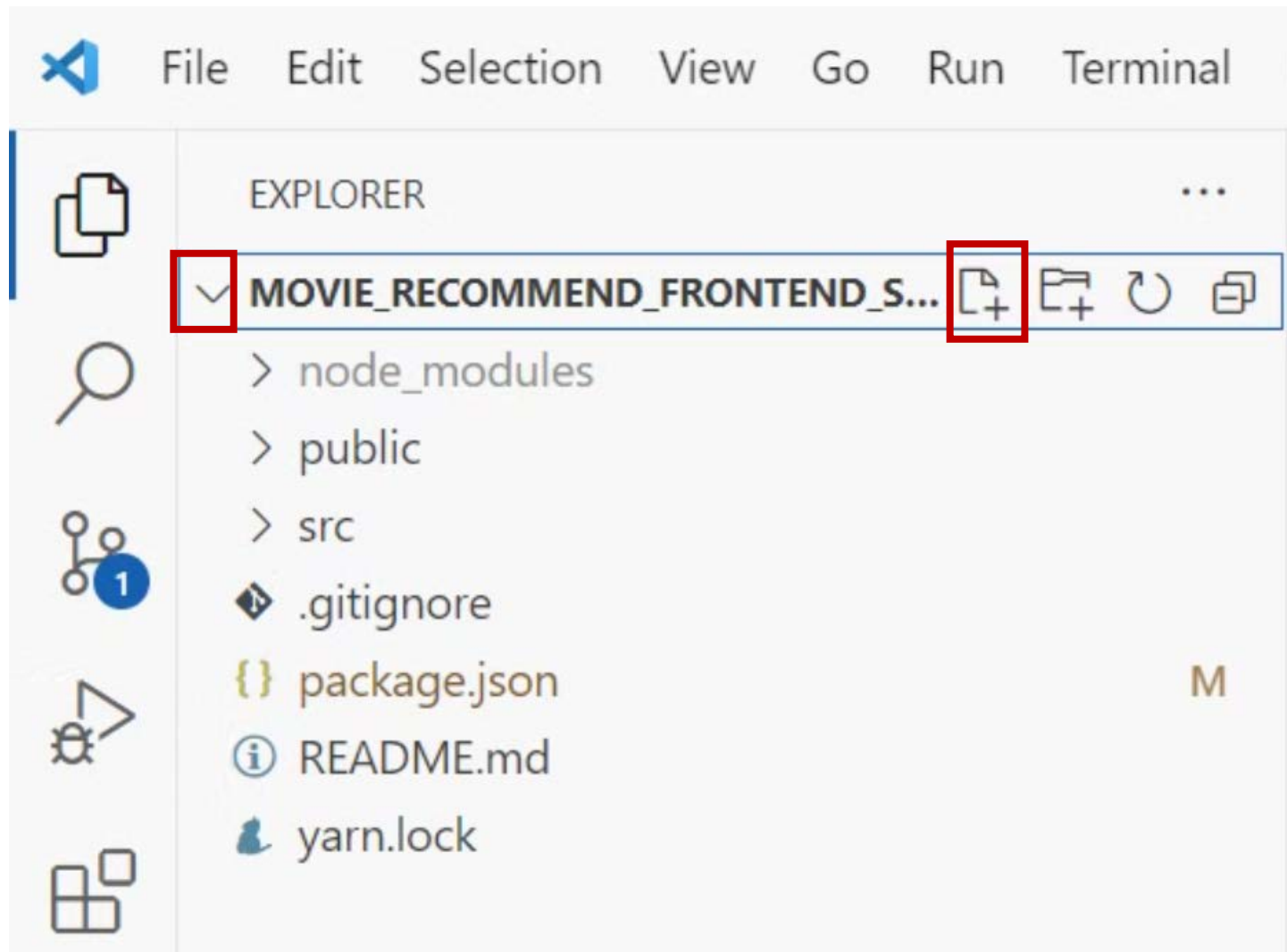
드래그 해서 모든 내용을 선택 하고 복사합니다





# Dockerfile 작성

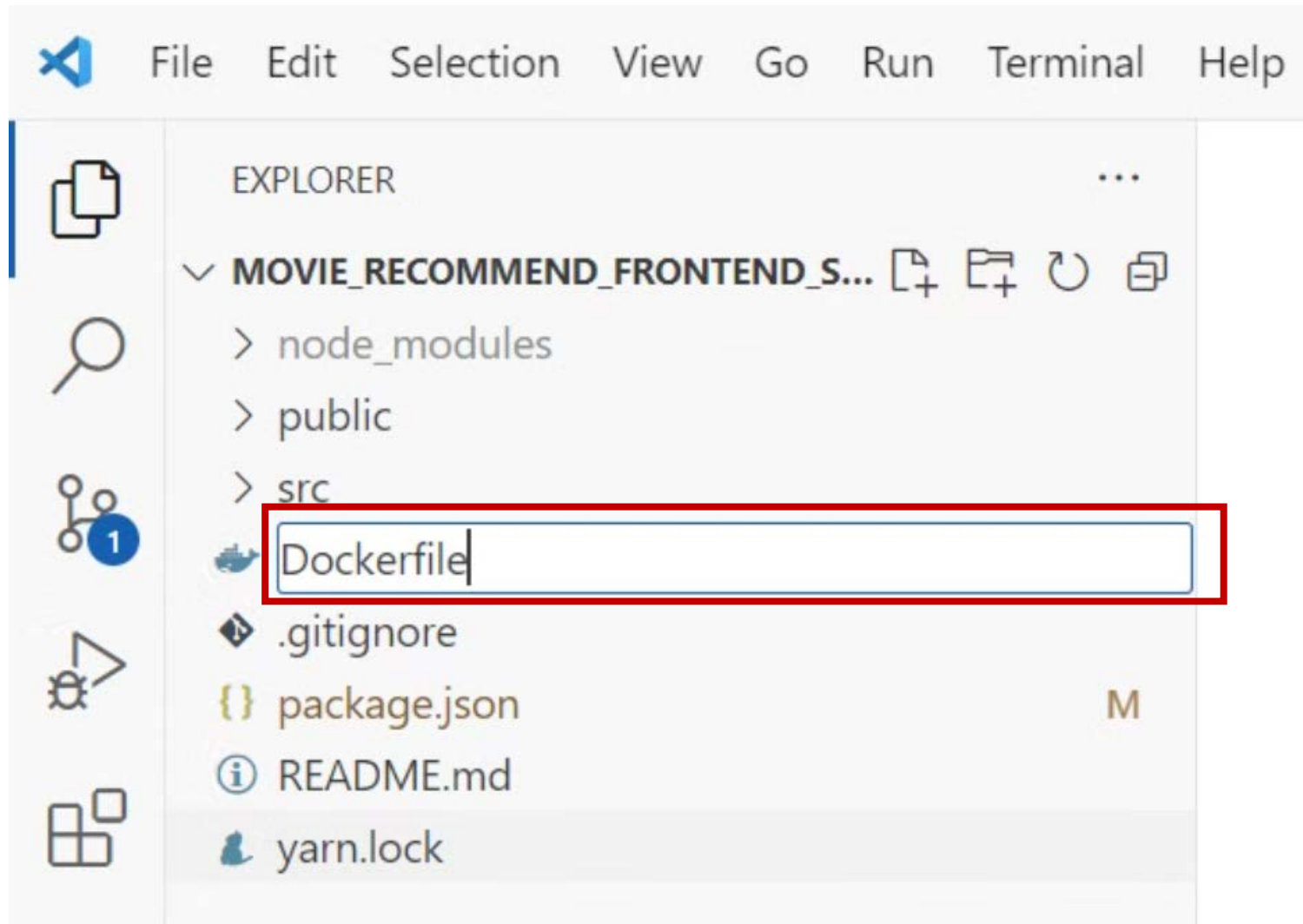
프로젝트 옆의 화살표를 선택하고 + 버튼을 클릭합니다





# Dockerfile 작성

Dockerfile 을 추가 합니다



# Dockerfile 작성

Dockerfile에 15페이지에서 복사한 내용을 붙여 넣고 Ctrl+S 를 눌러 저장 합니다

```

Dockerfile > ...
You, 1 hour ago | 1 author (You)
1 # 파일명: Dockerfile
2 # Docker 이미지를 생성하는 설정 파일입니다.
3
4 # 1단계: React 애플리케이션 빌드
5 # Node.js 16 버전의 이미지 사용
6 # - Node.js는 JavaScript 실행 환경으로, React 애플리케이션을 빌드하는 데 필요함
7 FROM node:16 AS builder
8
9 # 작업 디렉터리를 /app으로 설정
10 # - 모든 명령어는 /app 디렉터리에서 실행됩니다.
11 WORKDIR /app
12
13 # 의존성 설치
14 # - 프로젝트의 의존성 목록(package.json과 yarn.lock)을 복사합니다. You, 1
15 COPY package.json yarn.lock ./
16
17 # 필요한 패키지 설치 및 axios 추가
18 # - yarn install: package.json에 정의된 패키지를 설치합니다.
19 # - yarn add axios: axios(HTTP 요청 라이브러리)를 추가로 설치합니다.
20 RUN yarn install && yarn add axios
21
22 # 현재 디렉터리의 모든 파일을 컨테이너로 복사
23 # - 소스 코드와 구성 파일을 복사합니다.
24 COPY . .
25
26 # React 애플리케이션 빌드
27 # - 소스를 압축 및 최적화하여 배포 가능한 상태로 만듭니다.
28 RUN yarn build
29
30 # 2단계: nginx로 배포 설정
31 # - nginx는 웹 서버로, React 빌드 결과물을 제공하는 역할을 합니다.
32 FROM nginx:1.23-alpine
33
34 # nginx 템플릿 파일을 위한 작업 디렉터리 설정
35 # - nginx 설정 파일(template)을 저장할 디렉터리를 작업 디렉터리로 설정합니다.
36 WORKDIR /etc/nginx/templates
37
38 # 패키지 설치 및 설정
39 # - nginx 구동을 위한 추가 패키지 설치
40 RUN apk update && apk add --no-cache bash gettext && rm -rf /var/cache/apk/*
41 # bash : 웹 스크립트 실행을 위한 패키지
42 # gettext : envsubst 명령어를 제공하는 패키지로, 환경 변수 치환을 지원합니다.
43
44 # 빌드한 파일 복사
45 # - React 애플리케이션의 빌드 결과물을 nginx의 기본 경로에 복사합니다.
46 COPY --from=builder /app/build /usr/share/nginx/html
47
48 # nginx 설정 템플릿 복사
49 # - 환경 변수 치환을 지원하는 템플릿 파일을 복사합니다.
50 COPY nginx.conf.template /etc/nginx/templates/nginx.conf.template
51
52 # 시작 스크립트 복사
53 # - nginx 설정을 적용하고 서버를 시작하는 스크립트를 복사합니다.
54 COPY entrypoint.sh /entrypoint.sh
55
56 # 시작 스크립트의 줄바꿈 형식 변환
57 # - Windows와 Linux의 줄바꿈 형식 차이를 해결합니다.
58 RUN dos2unix /entrypoint.sh
59
60 # 시작 스크립트에 실행 권한 추가
61 # - 스크립트 실행을 위해 실행 권한을 부여합니다.
62 RUN chmod +x /entrypoint.sh
63
64 # 엔트리포인트 설정
65 # - 컨테이너가 시작될 때 /entrypoint.sh 스크립트를 실행합니다.
66 ENTRYPOINT ["/bin/bash", "/entrypoint.sh"]
67

```

# Dockerfile 설명

**Dockerfile**은 프로그램이 동작하는 환경을 만드는 레시피입니다.  
프로그램이 필요로 하는 도구와 설정 방법을 적어둔 파일입니다.

예를 들어, **햄버거 가게를 연다고** 생각해 봅시다.

- **Dockerfile**은 햄버거를 만드는 **조리법**입니다.
- 조리법에 따라 **재료 준비, 조리, 포장**이 이루어지죠.
- 이 파일을 실행하면 **햄버거(프로그램)**를 만드는 과정이 자동으로 진행됩니다.

# Dockerfile 설명

## 비유를 통한 요약

이 과정을 햄버거 가게 운영에 비유하면 다음과 같습니다.

### 1. 1단계 - 반죽 만들기 (React 준비)

- 반죽을 준비하고, 햄버거 속 재료를 미리 만들어 둡니다.

### 2. 2단계 - 포장하고 진열하기 (nginx 배포)

- 햄버거를 포장해서 진열대(서버)에 놓고, 손님을 맞을 준비를 합니다.

### 3. 마지막 단계 - 가게 문 열기

- 가게 문을 열고 손님들이 들어와서 햄버거(웹사이트)를 볼 수 있게 합니다.

# Dockerfile 설명

## 1단계: React 애플리케이션 준비 (반죽 만들기)

이 단계에서는 웹사이트에 필요한 파일을 미리 준비하는 과정을 수행합니다.  
이를 반죽을 준비하는 과정에 비유할 수 있습니다.

1. Node.js라는 도구를 사용합니다.

```
vbnet
```

 Copy code

```
FROM node:16 AS builder
```

- **Node.js**는 JavaScript 기반의 웹사이트 개발 도구입니다.
- 이 코드에서는 **Node.js 16버전**을 사용하도록 지정합니다.
- **AS builder**는 해당 단계를 반죽 준비 단계라고 이름 붙이는 역할을 합니다.

# Dockerfile 설명

## 2.작업 디렉터리 설정

```
bash
```

 Copy code

```
WORKDIR /app
```

- 반죽을 준비할 **작업 공간(폴더)**을 생성합니다.
- 이 폴더 안에서 앞으로의 작업이 이루어집니다.

# Dockerfile 설명

## 3.필요한 프로그램을 설치합니다

```
csharp
```

 Copy code

```
COPY package.json yarn.lock ./
```

- **package.json**과 **yarn.lock** 파일에는 프로그램에 필요한 도구 목록이 작성되어 있습니다.
- 이를 바탕으로 필요한 도구들이 설치됩니다.

# Dockerfile 설명

## 4. 프로그램 설치 및 추가 도구 설치

```
csharp
```

 Copy code

```
RUN yarn install && yarn add axios
```

- **yarn install** 명령어를 사용하여 필요한 프로그램을 설치합니다.
- 추가로 **axios**라는 도구를 설치합니다.
  - **axios**는 데이터를 주고받을 때 자주 사용되는 도구입니다.



# Dockerfile 설명

## 5. 프로그램 코드 복사

```
sql
```

 Copy code

COPY . .

- 현재 폴더의 모든 파일을 작업 공간으로 복사합니다.

# Dockerfile 설명

## 6.웹사이트를 빌드(구성)합니다

```
RUN yarn build
```

 Copy code

- React 애플리케이션을 웹사이트에 배포할 수 있도록 **포장된 상태**로 준비합니다.
- 이 과정에서 HTML, CSS, JavaScript와 같은 파일들이 생성됩니다.

# Dockerfile 설명

## 2단계: nginx로 배포 설정 (포장하고 진열하기)

이 단계에서는 완성된 웹사이트 파일을 배포하기 위해  
오븐에 구워 포장하고, 가게에 진열하는 과정을 수행합니다.

### 1. nginx 이미지 사용

CSS

 Copy code

```
FROM nginx:1.23-alpine
```

- **nginx**는 웹사이트를 인터넷에 배포하는 역할을 합니다.
- 용량이 작고 빠른 **경량 버전**을 사용합니다.

# Dockerfile 설명

## 2. 작업 공간을 설정합니다.

```
bash
```

 Copy code

```
WORKDIR /etc/nginx/templates
```

- nginx 설정 파일이 들어갈 폴더를 설정합니다.

# Dockerfile 설명

## 3. 필요한 도구 설치

sql

 Copy code

```
RUN apk update && apk add --no-cache bash gettext && rm -rf /var/cache/apk/*
```

- 이 명령어는 프로그램 실행을 위한 여러 도구들을 설치합니다.
  - **bash**: 명령어 실행 도구.
  - **gettext**: 환경 변수 값을 자동으로 변환해주는 도구.
- 설치 후 필요 없는 파일들은 삭제하여 용량을 최적화합니다.

# Dockerfile 설명

4. 완성된 파일을 복사합니다.

bash

 Copy code

```
COPY --from=builder /app/build /usr/share/nginx/html
```

- 1단계에서 빌드된 웹사이트 파일을 복사합니다.
- 복사된 파일은 nginx가 사용자에게 보여줄 **웹사이트 공간**에 배치됩니다.

# Dockerfile 설명

## 5.nginx 설정 파일 추가

arduino

 Copy code

```
COPY nginx.conf.template /etc/nginx/templates/nginx.conf.template
```

- nginx의 동작 방식을 설정한 **템플릿 파일**을 복사합니다.
- 이 설정에는 웹사이트에 접속하는 사용자 수나 연결 방식 등의 규칙이 포함되어 있습니다.

# Dockerfile 설명

## 6.시작 스크립트 복사 및 실행 권한 부여

dockerfile

 Copy code

```
COPY entrypoint.sh /entrypoint.sh  
RUN dos2unix /entrypoint.sh  
RUN chmod +x /entrypoint.sh
```

- entrypoint.sh 복사:
  - nginx 설정을 적용하고 서버를 시작하는 명령어가 저장된 스크립트 파일을 복사합니다.
- 형식 변환(dos2unix):
  - Windows에서 작성한 파일은 줄바꿈 방식이 Linux와 다르므로 변환합니다.
- 실행 권한 추가:
  - 스크립트가 실행될 수 있도록 실행 권한을 부여합니다.



# Dockerfile 설명

## 3. 문을 열고 손님을 맞이할 준비

### 1. 80번 포트 개방

 Copy code


```
EXPOSE 80
```

- **80번 포트**는 웹사이트 접속을 위한 기본 포트입니다.
- 이 포트를 통해 사용자가 웹사이트에 접속할 수 있습니다.

# Dockerfile 설명

## 2.서버 시작 명령어 실행

dockerfile

 Copy code

```
ENTRYPOINT ["/bin/bash", "/entrypoint.sh"]
```

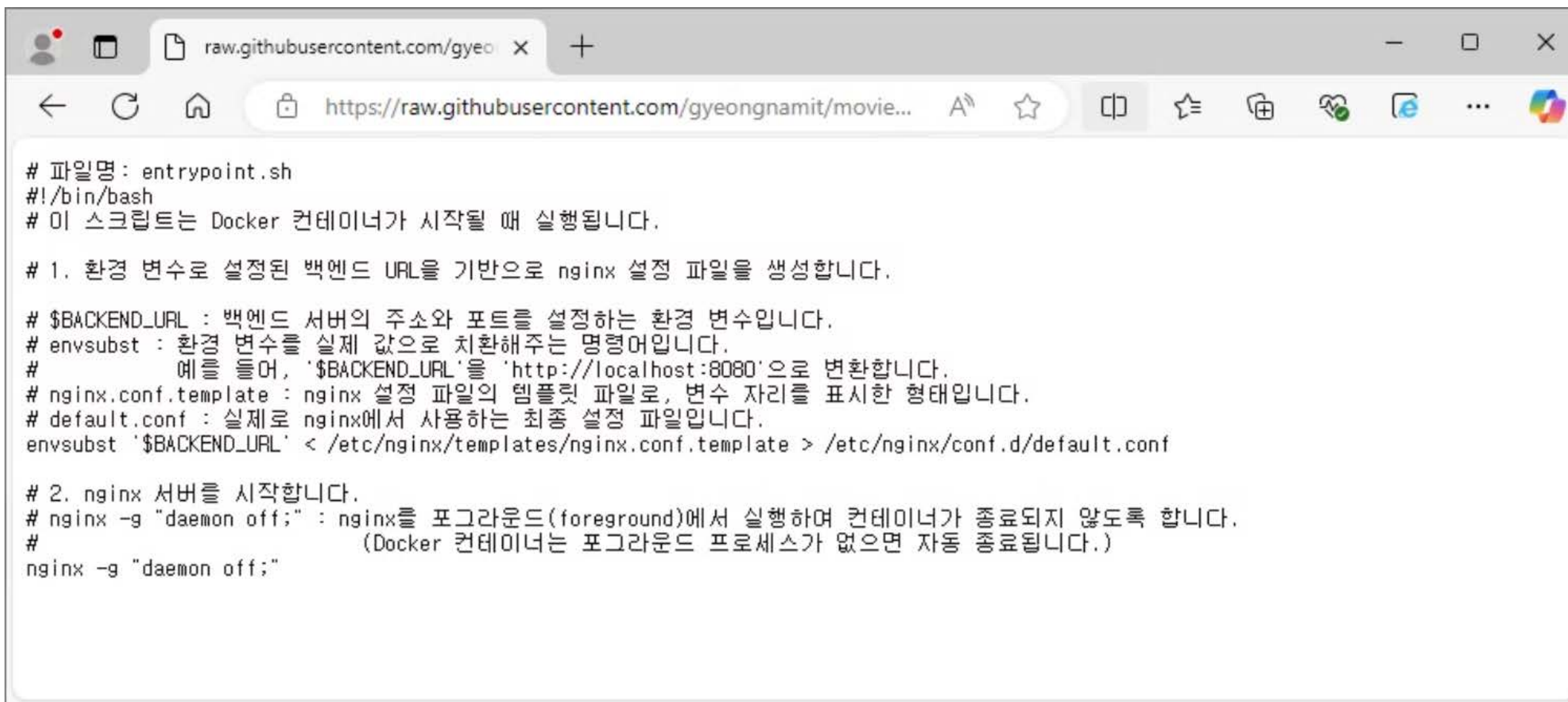
- 시작 명령어 설정:
  - 컨테이너가 실행될 때 자동으로 /entrypoint.sh 스크립트를 실행합니다.
  - 이 스크립트는 nginx 서버를 시작하는 역할을 합니다.

## 6.entrypoint.sh 구현

entrypoint.sh는 Docker 컨테이너가 시작될 때 자동으로 실행되는 핵심 스크립트 파일입니다.

# entrypoint.sh 구현

[https://raw.githubusercontent.com/gyeongnamit/movie\\_recommend\\_frontend\\_step2/refs/heads/main/entrypoint.sh](https://raw.githubusercontent.com/gyeongnamit/movie_recommend_frontend_step2/refs/heads/main/entrypoint.sh) 접속 합니다



```
# 파일명: entrypoint.sh
#!/bin/bash
# 이 스크립트는 Docker 컨테이너가 시작될 때 실행됩니다.

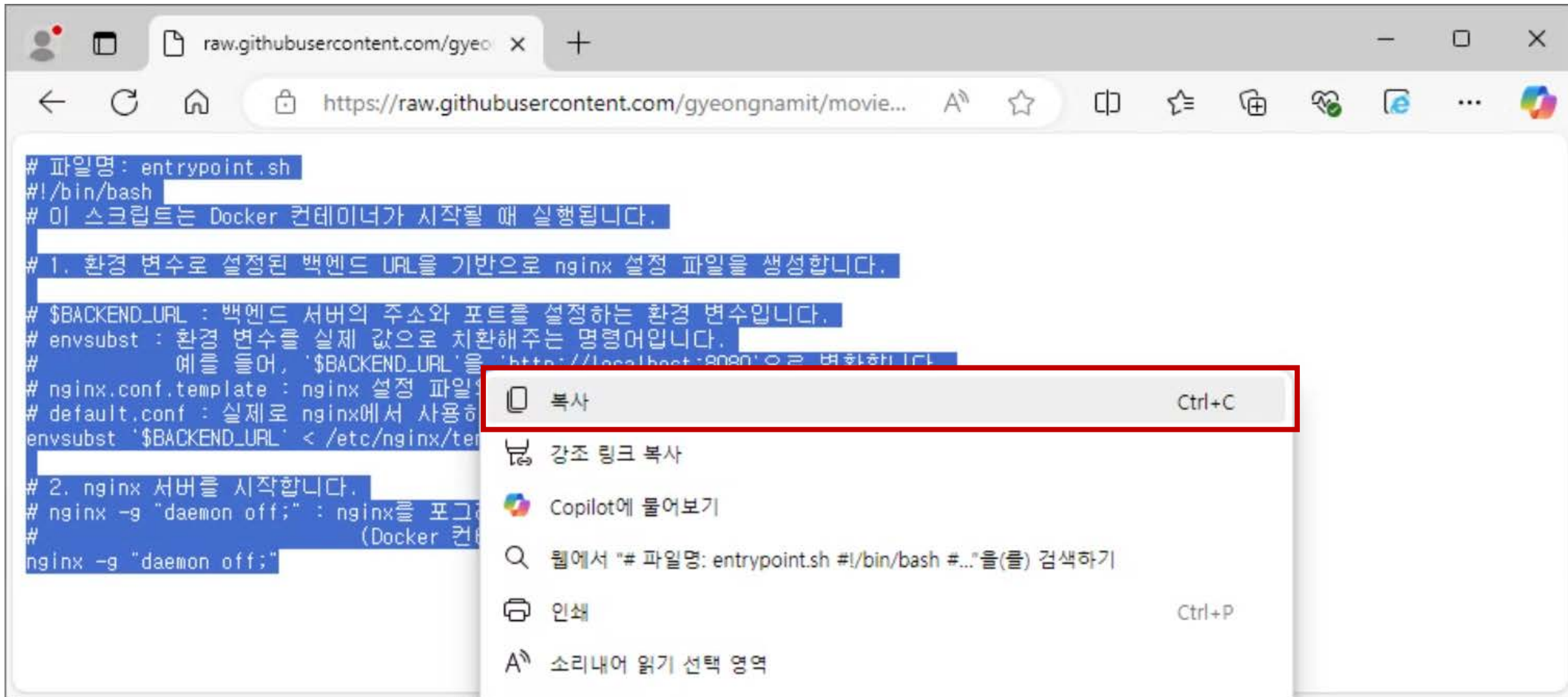
# 1. 환경 변수로 설정된 백엔드 URL을 기반으로 nginx 설정 파일을 생성합니다.

# $BACKEND_URL : 백엔드 서버의 주소와 포트를 설정하는 환경 변수입니다.
# envsubst : 환경 변수를 실제 값으로 치환해주는 명령어입니다.
#           예를 들어, '$BACKEND_URL'을 'http://localhost:8080'으로 변환합니다.
# nginx.conf.template : nginx 설정 파일의 템플릿 파일로, 변수 자리를 표시한 형태입니다.
# default.conf : 실제로 nginx에서 사용하는 최종 설정 파일입니다.
envsubst '$BACKEND_URL' < /etc/nginx/templates/nginx.conf.template > /etc/nginx/conf.d/default.conf

# 2. nginx 서버를 시작합니다.
# nginx -g "daemon off;" : nginx를 포그라운드(foreground)에서 실행하여 컨테이너가 종료되지 않도록 합니다.
#                           (Docker 컨테이너는 포그라운드 프로세스가 없으면 자동 종료됩니다.)
nginx -g "daemon off;"
```

# entrypoint.sh 구현

드래그 해서 모든 내용을 선택한 후 복사 합니다



The screenshot shows a web browser window with the address bar displaying `https://raw.githubusercontent.com/gyeongnamit/movie...`. The main content area shows a text file named `entrypoint.sh` with the following content:

```
# 파일명: entrypoint.sh
#!/bin/bash
# 이 스크립트는 Docker 컨테이너가 시작될 때 실행됩니다.

# 1. 환경 변수로 설정된 백엔드 URL을 기반으로 nginx 설정 파일을 생성합니다.

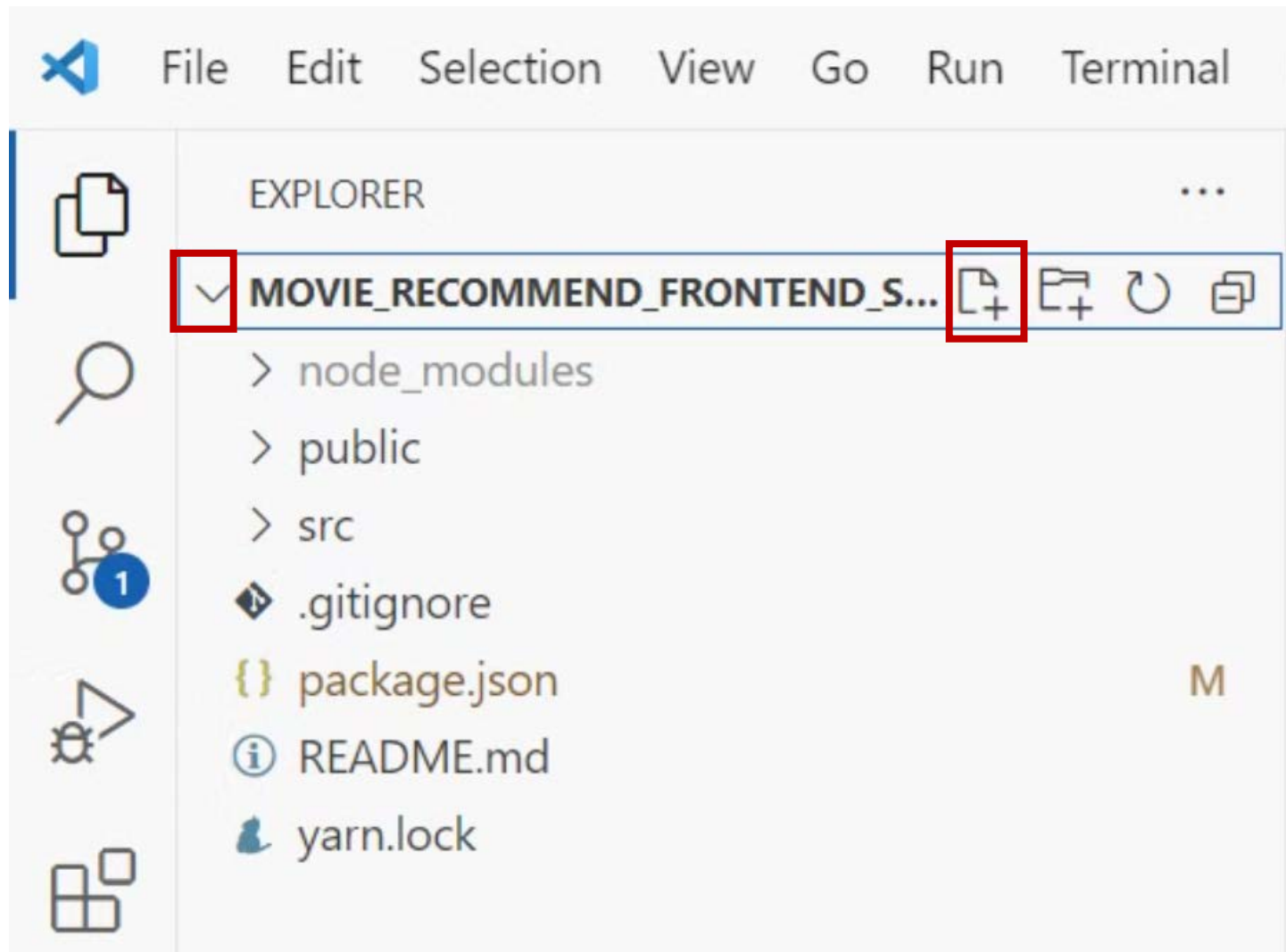
# $BACKEND_URL : 백엔드 서버의 주소와 포트를 설정하는 환경 변수입니다.
# envsubst : 환경 변수를 실제 값으로 치환해주는 명령어입니다.
# 예를 들어, '$BACKEND_URL'을 'http://localhost:8080'으로 변환합니다.
# nginx.conf.template : nginx 설정 파일
# default.conf : 실제로 nginx에서 사용하는 기본 설정 파일
envsubst '$BACKEND_URL' < /etc/nginx/template/nginx.conf.template > /etc/nginx/conf.d/default.conf

# 2. nginx 서버를 시작합니다.
# nginx -g "daemon off;" : nginx를 포그라운드에서 실행합니다. (Docker 컨테이너에서 사용)
nginx -g "daemon off;"
```

A context menu is open over the code, with the '복사' (Copy) option highlighted. The menu also includes '강조 링크 복사' (Copy link), 'Copilot에 물어보기' (Ask Copilot), '웹에서 "# 파일명: entrypoint.sh #!/bin/bash #..."을(를) 검색하기' (Search for "# filename: entrypoint.sh #!/bin/bash #" on the web), '인쇄' (Print), and '소리내어 읽기 선택 영역' (Select area to read aloud).

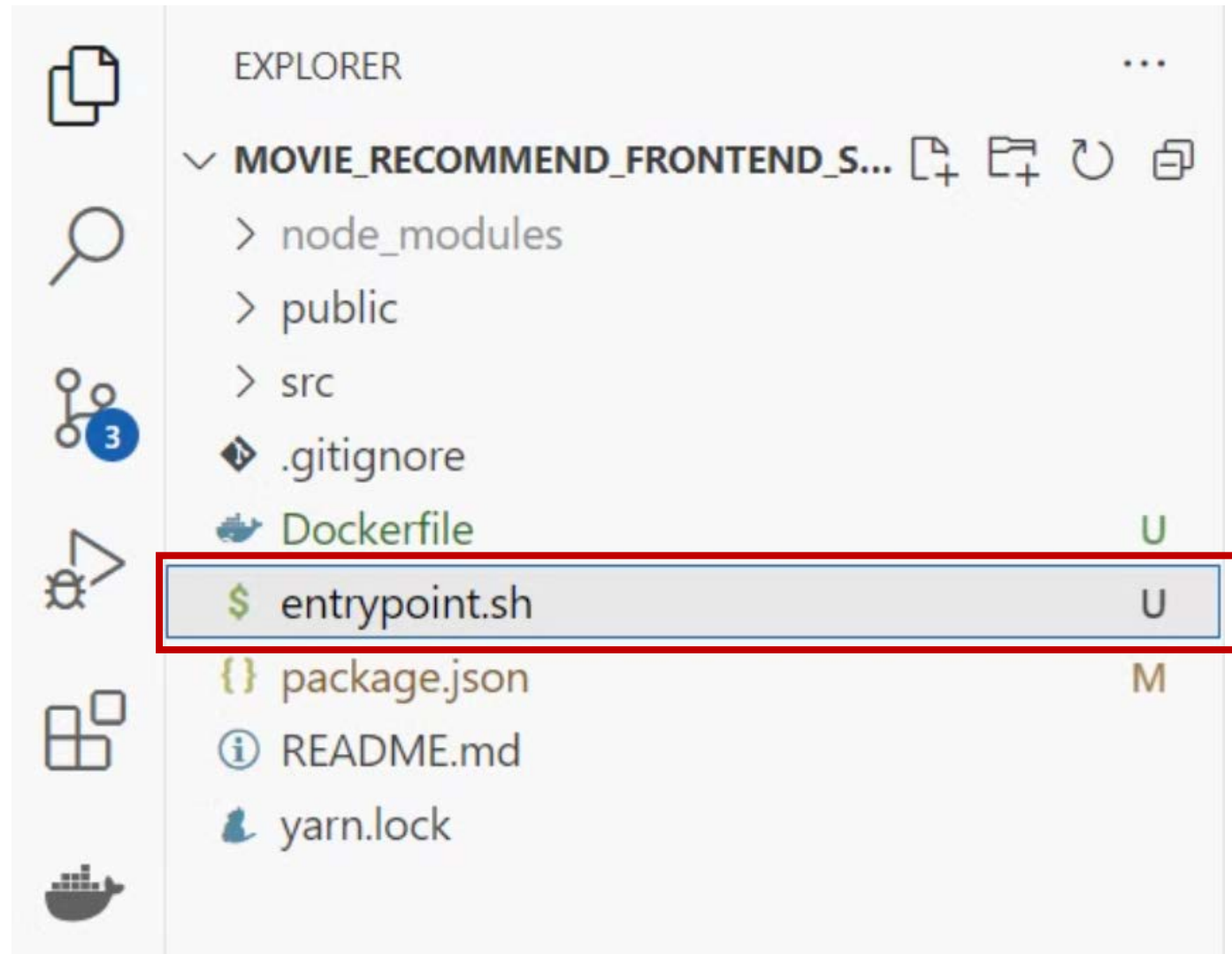
# entrypoint.sh 구현

프로젝트를 선택하고 + 버튼을 클릭합니다



# entrypoint.sh 구현

entrypoint.sh 파일을 추가 합니다





# entrypoint.sh 구현

entrypoint.sh 파일에 37 페이지에서 복사한 내용을 붙여 넣고 Ctrl+S를 눌러서 저장 합니다

```
$ entrypoint.sh U X
$ entrypoint.sh
1  # 파일명: entrypoint.sh
2  #!/bin/bash
3  # 이 스크립트는 Docker 컨테이너가 시작될 때 실행됩니다.
4
5  # 1. 환경 변수로 설정된 백엔드 URL을 기반으로 nginx 설정 파일을 생성합니다.
6
7  # $BACKEND_URL : 백엔드 서버의 주소와 포트를 설정하는 환경 변수입니다.
8  # envsubst : 환경 변수를 실제 값으로 치환해주는 명령어입니다.
9  #           예를 들어, '$BACKEND_URL'을 'http://localhost:8080'으로 변환합니다.
10 # nginx.conf.template : nginx 설정 파일의 템플릿 파일로, 변수 자리를 표시한 형태입니다.
11 # default.conf : 실제로 nginx에서 사용하는 최종 설정 파일입니다.
12 envsubst '$BACKEND_URL' < /etc/nginx/templates/nginx.conf.template > /etc/nginx/conf.d/default.conf
13
14 # 2. nginx 서버를 시작합니다.
15 # nginx -g "daemon off;" : nginx를 포그라운드(foreground)에서 실행하여 컨테이너가 종료되지 않도록 합니다.
16 #           (Docker 컨테이너는 포그라운드 프로세스가 없으면 자동 종료됩니다.)
17 nginx -g "daemon off;"
```



# entrypoint.sh 설명

entrypoint.sh는 **Docker** 컨테이너가 시작될 때 자동으로 실행되는 명령어 모음 파일입니다.

쉽게 말해, **컴퓨터가 켜질 때 자동으로 실행되는 시작 버튼** 역할을 합니다.

## 이 파일은 어떤 일을 하나요?

이 파일은 크게 두 가지 일을 합니다.

1. **nginx 서버 설정을 준비합니다.** (웹사이트 접속 설정)
2. **nginx 서버를 실행합니다.** (웹사이트를 사용자에게 보여주기 시작)

# entrypoint.sh 설명

전체 과정을 요약해 봅시다.

이 파일은 웹사이트 서비스 준비 및 실행 과정을 자동화합니다.

## 1. nginx 설정 파일 만들기

- 환경 변수 값을 채워서 설정 파일을 완성합니다.
- 예: '전화번호 빈 칸'을 실제 값으로 채웁니다.

## 2. nginx 서버 실행

- nginx 서버가 계속 동작하도록 설정합니다.
- 예: 가게 문을 열고 손님이 계속 올 수 있도록 합니다.

# entrypoint.sh 설명

## 쉽게 비유로 다시 정리

이 파일의 동작을 **햄버거 가게 운영**에 비유하면 다음과 같습니다.

### 1. 1단계 - 설정 준비 (메뉴판 작성)

- 메뉴판의 빈 칸(전화번호)을 채웁니다.
- 채운 메뉴판을 가게에 걸어둡니다.

### 2. 2단계 - 가게 문 열기 (서버 시작)

- 문을 열고 손님을 맞을 준비를 합니다.
- 문을 닫지 않고 계속 열어 둡니다.

# entrypoint.sh 설명

1단계: nginx 설정 파일 만들기 (웹사이트 설정 준비)

환경 변수란?

1. 환경 변수는 프로그램이 동작할 때 필요한 값을 저장하는 곳입니다. 예를 들어, BACKEND\_URL은 웹사이트가 데이터를 가져올 백엔드 서버 주소를 저장합니다. 이 주소는 웹사이트와 서버가 서로 통신하는 전화번호 같은 역할을 합니다.

## 1. 명령어 설명

```
javascript Copy code  
  
envsubst '$BACKEND_URL' < /etc/nginx/templates/nginx.conf.template > /etc/nginx/conf.d/def
```

### 1. envsubst

- 환경 변수 값을 실제 값으로 바꿔주는 명령어입니다.
- 예: '\$BACKEND\_URL' → '<http://localhost:8080>'으로 변환합니다.

### 2. nginx.conf.template

- nginx 설정의 **템플릿 파일**입니다.
- 템플릿 파일에는 '변수 자리표(\$BACKEND\_URL)'만 적혀 있고,
- 실제 값은 나중에 채워집니다.

### 3. default.conf

- 최종 완성된 nginx 설정 파일입니다.
- 이 파일은 실제로 nginx가 사용할 설정 정보입니다.

## 비유

- **템플릿 파일(nginx.conf.template)**: 빈 칸이 있는 계약서 양식.
- **환경 변수(\$BACKEND\_URL)**: 빈 칸에 채울 주소나 전화번호.
- **최종 설정 파일(default.conf)**: 빈 칸이 채워진 최종 계약서.

이 명령어는 **계약서를 완성하는 과정**과 같습니다.

웹사이트가 어떤 서버와 연결될지 정리해 줍니다.

# entrypoint.sh 설명

## 2단계: nginx 서버 실행 (문 열기)

### 명령어 설명

arduino

 Copy code

```
nginx -g "daemon off;"
```

- **nginx**
  - 웹사이트를 서비스하는 도구(서버)입니다.
  - **가게 문을 열고 손님을 맞을 준비를 시작하는 역할**을 합니다.
- **-g "daemon off;"**
  - nginx 서버가 백그라운드(숨겨진 상태)가 아니라,
  - **앞에서 계속 실행되도록 설정**합니다.
  - 이 명령어가 없으면, 문을 연 뒤 바로 가게가 문을 닫습니다.
  - 즉, 서버가 멈추지 않고 계속 실행되도록 유지합니다.

## 6. nginx.conf.template 구현

nginx.conf.template은 Nginx 웹 서버의 설정을 정의하는 템플릿 파일입니다. 이 파일은 프록시 설정, 서버 포트, 백엔드 연결 등 핵심적인 웹 서버 설정을 포함하고 있습니다. 환경 변수를 사용하여 유연한 설정이 가능하며, 실제 운영 환경에서 최종 nginx.conf 파일로 변환됩니다.

# nginx.conf.template 구현

[https://raw.githubusercontent.com/gyeongnamit/movie\\_recommend\\_frontend\\_step2/refs/heads/main/nginx.conf.template](https://raw.githubusercontent.com/gyeongnamit/movie_recommend_frontend_step2/refs/heads/main/nginx.conf.template) 접속 합니다

```
# 파일명 : Dockerfile
# Docker 이미지를 생성하는 설정 파일입니다.

# 1단계 : React 애플리케이션 빌드
# Node.js 16 버전의 이미지 사용
FROM node:16 AS builder
# 작업 디렉터리를 /app으로 설정
WORKDIR /app

# 의존성 설치
# package.json과 yarn.lock 파일을 복사
COPY package.json yarn.lock ./
# 필요한 패키지 설치 및 axios 추가
RUN yarn install && yarn add axios
# 현재 디렉터리의 모든 파일을 컨테이너로 복사
COPY . .
# React 애플리케이션 빌드
RUN yarn build

# 2단계 : nginx로 배포 설정
# 경량 nginx 이미지를 사용합니다.
FROM nginx:1.23-alpine
# nginx 템플릿 파일을 위한 작업 디렉터리 설정
WORKDIR /etc/nginx/templates

# 패키지 설치 및 설정
RUN apk update && apk add --no-cache bash gettext && rm -rf /var/cache/apk/*
# bash : 쉘 스크립트 실행을 위한 패키지
# gettext : envsubst 명령어를 제공하는 패키지

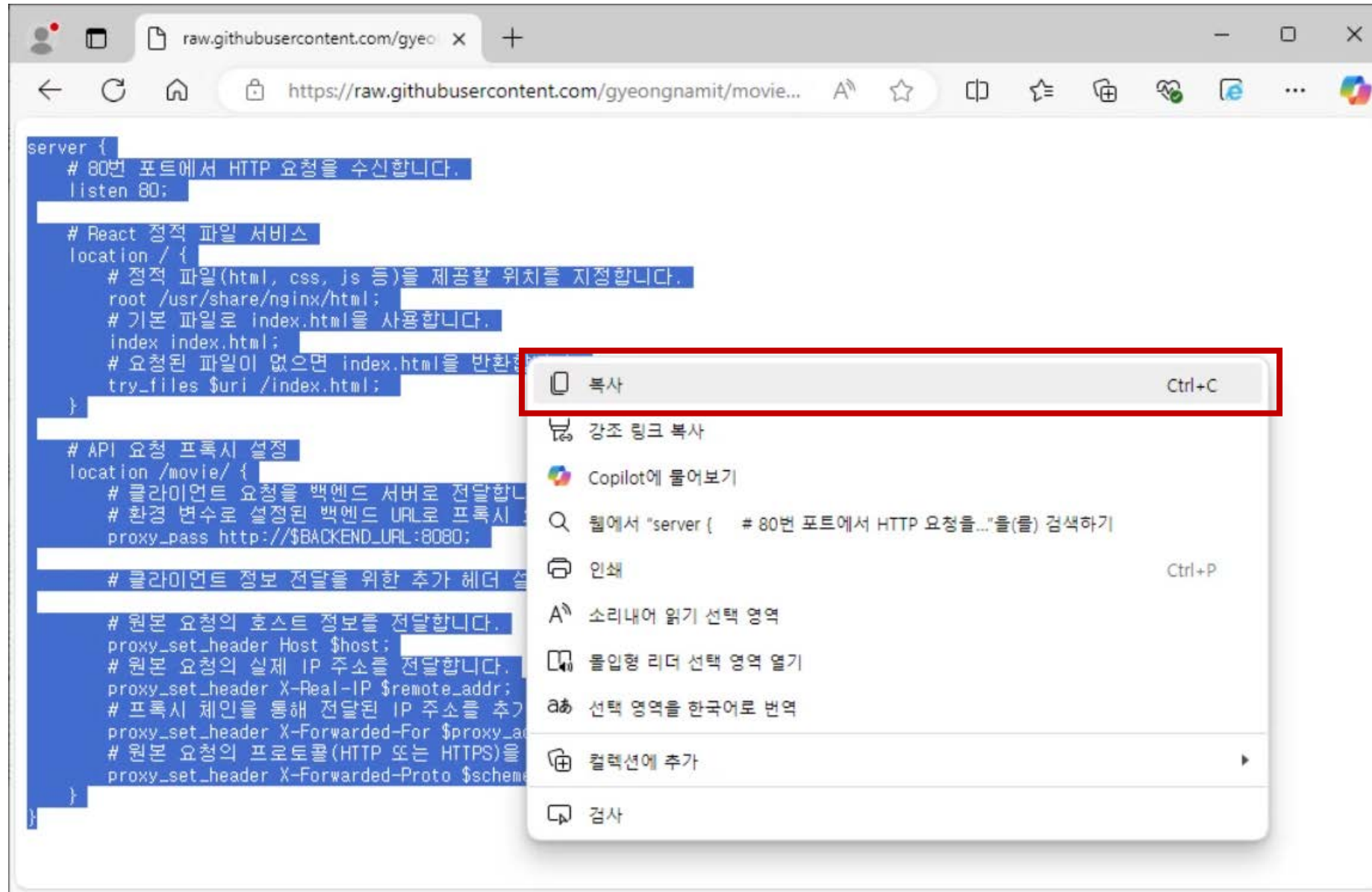
# 빌드한 파일 복사
# 빌드된 정적 파일을 nginx 루트 디렉터리에 복사
COPY --from=builder /app/build /usr/share/nginx/html
# nginx 설정 템플릿 복사
COPY nginx.conf.template /etc/nginx/templates/nginx.conf.template
# 시작 스크립트 복사
COPY entrypoint.sh /entrypoint.sh
# 시작 스크립트에 실행 권한 추가
RUN chmod +x /entrypoint.sh

# 컨테이너의 80번 포트를 호스트에 노출
EXPOSE 80
# 컨테이너 시작 시 entrypoint.sh 실행
CMD ["/entrypoint.sh"]
```



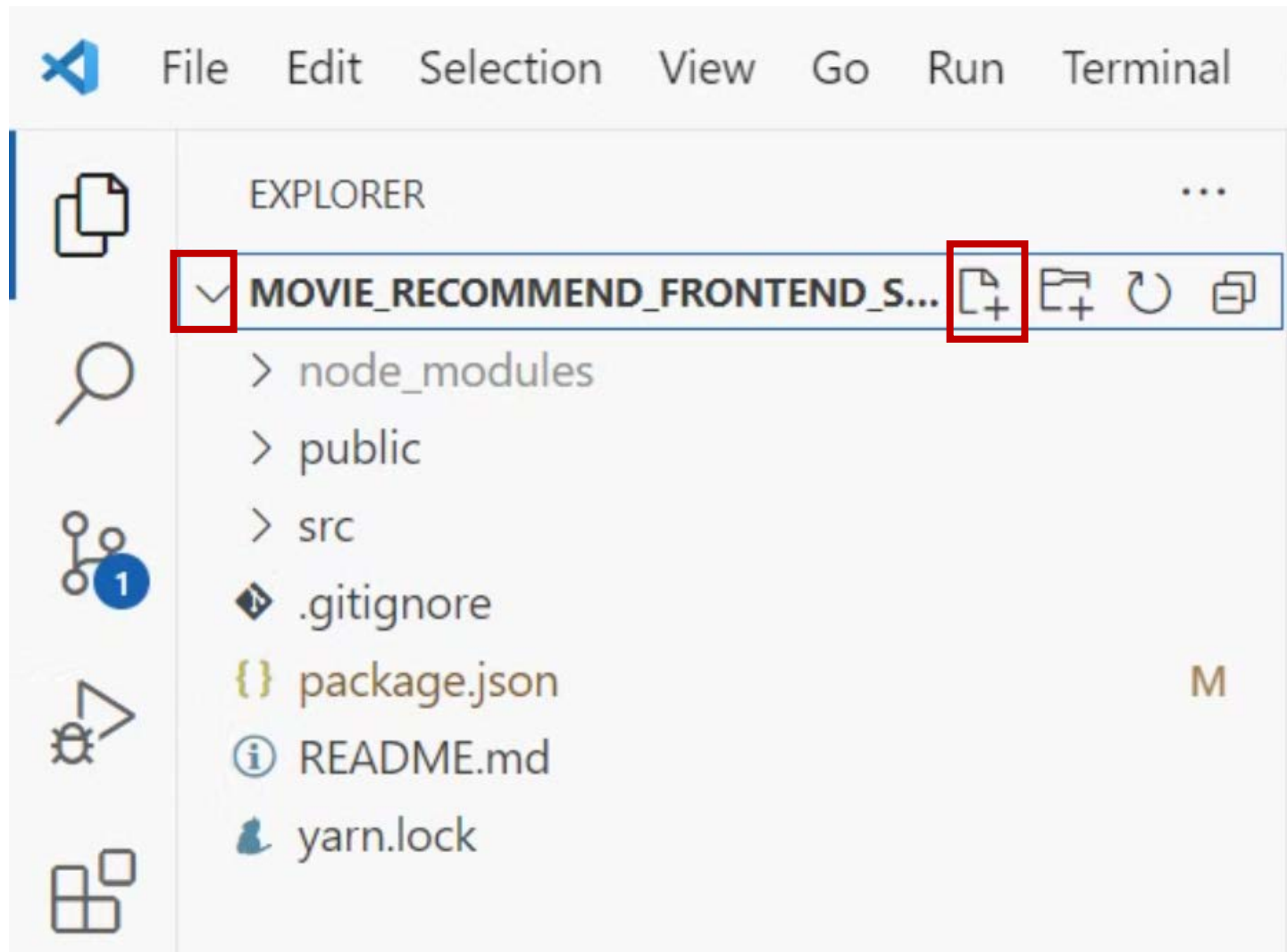
# nginx.conf.template 구현

파일의 전체 내용을 드래그 해서 선택 한 후 복사 합니다



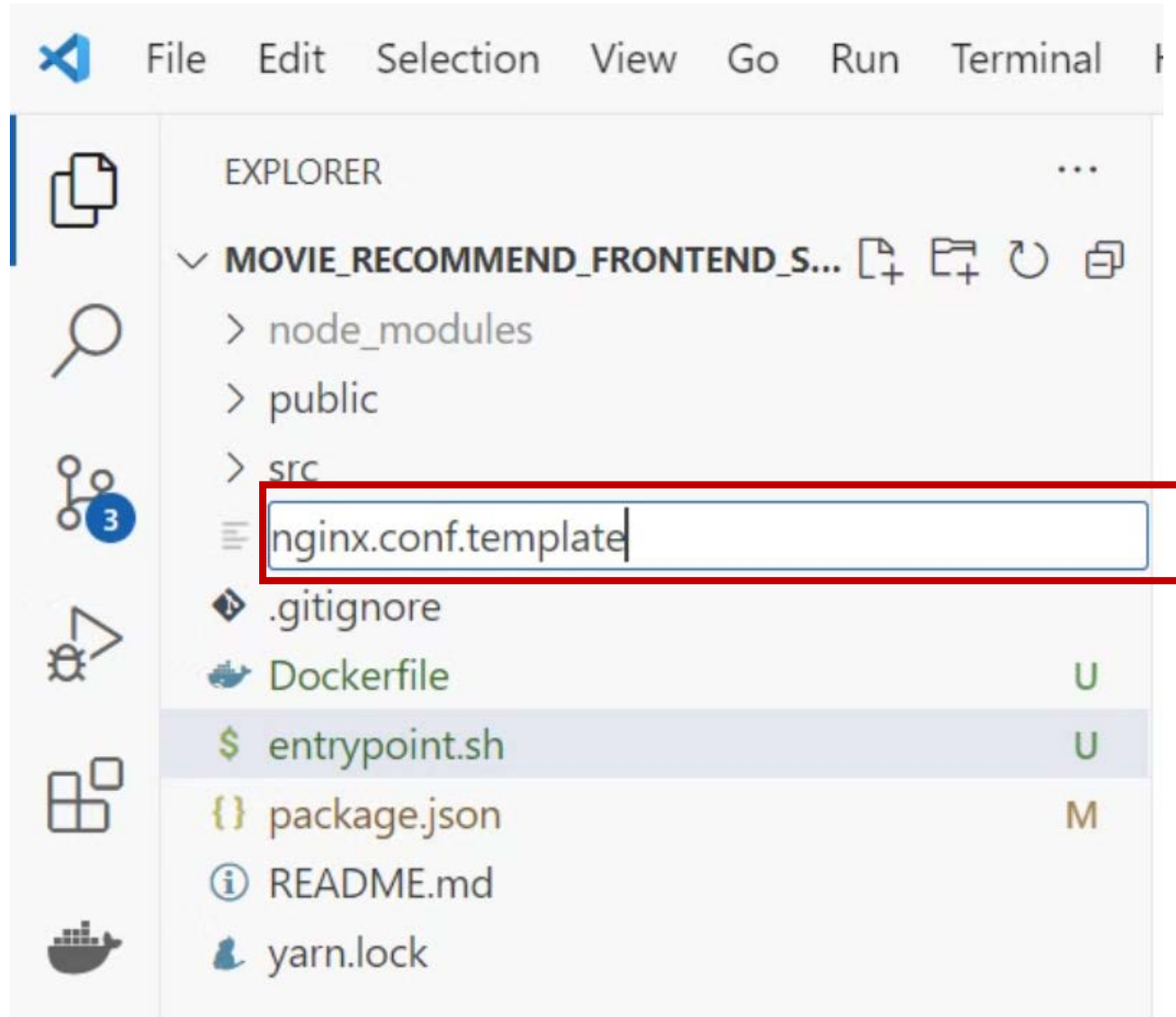
# nginx.conf.template 구현

프로젝트를 선택하고 + 버튼을 클릭합니다



# nginx.conf.template 구현

nginx.conf.template 파일을 추가 합니다



# nginx.conf.template 구현

nginx.conf.template 49 페이지에서 복사한 내용을 붙여 넣고 Ctrl+S 를 눌러서 저장합니다

```
nginx.conf.template U X
nginx.conf.template
1  server {
2      # 80번 포트에서 HTTP 요청을 수신합니다.
3      listen 80;
4
5      # React 정적 파일 서비스
6      location / {
7          # 정적 파일(html, css, js 등)을 제공할 위치를 지정합니다.
8          root /usr/share/nginx/html;
9          # 기본 파일로 index.html을 사용합니다.
10         index index.html;
11         # 요청된 파일이 없으면 index.html을 반환합니다.
12         try_files $uri /index.html;
13     }
14
15     # API 요청 프록시 설정
16     location /movie/ {
17         # 클라이언트 요청을 백엔드 서버로 전달합니다.
18         # 환경 변수로 설정된 백엔드 URL로 프록시 요청을 보냅니다.
19         proxy_pass http://$BACKEND_URL:8080;
20
21         # 클라이언트 정보 전달을 위한 추가 헤더 설정
22
23         # 원본 요청의 호스트 정보를 전달합니다.
24         proxy_set_header Host $host;
25         # 원본 요청의 실제 IP 주소를 전달합니다.
26         proxy_set_header X-Real-IP $remote_addr;
27         # 프록시 체인을 통해 전달된 IP 주소를 추가합니다.
28         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
29         # 원본 요청의 프로토콜(HTTP 또는 HTTPS)을 전달합니다.
30         proxy_set_header X-Forwarded-Proto $scheme;
31     }
32 }
```

# nginx.conf.template 설명

## 전체 과정 요약

### 1. 웹사이트 파일 보여주기 (메뉴판 제공)

- 손님이 가게에 들어오면, 기본 메뉴판(index.html)을 보여줍니다.
- 파일이 없더라도 빈 메뉴판 대신 기본 메뉴판을 보여줍니다.

### 2. 데이터 요청 처리 (주방 연결)

- 손님이 메뉴에서 특정 요리를 요청하면, 주방(서버)에서 데이터를 가져옵니다.
- 백엔드 서버 주소는 환경 변수(**\$BACKEND\_URL**)에 따라 자동 설정됩니다.

### 3. 추가 정보 전달 (주문서 작성)

- 요청한 손님의 위치, 경로, 방식 등 추가 정보를 전달하여 정확한 처리 결과를 보장합니다.

# nginx.conf.template 설명

## 비유로 다시 설명

이 설정을 **햄버거 가게 운영**에 비유하면 다음과 같습니다.

### 1. 손님이 가게에 들어오면 메뉴판을 보여줍니다.

- 파일 위치를 설정하고, 기본 메뉴판(index.html)을 제공합니다.
- 메뉴가 없을 경우, 기본 메뉴판을 대신 보여줍니다.

### 2. 손님이 특정 요리를 주문하면 주방에 요청을 전달합니다.

- `/movie/` 요청은 주방(백엔드 서버)에서 데이터를 처리합니다.
- 예: "햄버거 주문서를 주방에 전달하고 결과를 받아옵니다."

### 3. 주문서에 추가 메모를 적어서 주방에 전달합니다.

- 손님의 위치, 경로, 요청 방식 등 추가 정보를 함께 보냅니다.
- 예: "손님 테이블 번호와 요청 사항을 주방에 전달합니다."

# nginx.conf.template 설명

웹사이트와 서버 간의 연결 방법을 설정하는 내용입니다.

쉽게 말해, 웹사이트를 서비스하는 가게의 운영 규칙을 정하는 것입니다.

1. **웹사이트 파일 제공** – 손님(사용자)에게 웹사이트를 보여줍니다.
2. **요청 처리(프록시)** – 손님이 메뉴(데이터)를 요청하면, 주방(서버)에서 가져옵니다.

# nginx.conf.template 설명

## 1단계: 웹사이트 파일 제공 (웹페이지 보여주기)

이 부분은 웹사이트의 메인 화면을 보여주는 설정입니다.

arduino

 Copy code

```
server {  
    listen 80;
```

- **80번 포트**에서 요청을 받습니다.
- **80번 포트**는 웹사이트가 기본적으로 사용하는 문입니다.
- (예: 손님들이 가게로 들어올 수 있는 문)



# nginx.conf.template 설명

bash

 Copy code

```
location / {  
    root /usr/share/nginx/html;  
    index index.html;  
    try_files $uri /index.html;  
}
```

## 1.웹사이트 파일 위치 지정

- **root /usr/share/nginx/html;**
  - **웹사이트 파일**이 저장된 폴더 위치입니다.
  - 손님(사용자)에게 보여줄 메뉴판(파일)이 여기에 있습니다.

## 2.기본 파일 지정

- **index index.html;**
  - 사용자가 접속하면 기본으로 **index.html** 파일을 보여줍니다.
  - 예: 식당에 들어오면 기본 메뉴판을 자동으로 보여주는 것과 같습니다.

## 3.파일 없을 때 처리

- **try\_files \$uri /index.html;**
  - 사용자가 요청한 파일이 없으면 **index.html**을 대신 보여줍니다.
  - 예: 메뉴판에 없는 요리를 주문해도 기본 메뉴판을 다시 보여줍니다.

# nginx.conf.template 설명

## 2단계: API 요청 처리 (주방 연결)

이 부분은 웹사이트에서 필요한 데이터를 백엔드 서버에서 가져오는 설정입니다.

```
bash
```

 Copy code

```
location /movie/ {  
    proxy_pass http://$BACKEND_URL:8080;
```

### 1. 요청 경로 설정

- **location /movie/:**
  - '/movie/'로 시작하는 요청을 처리합니다.
  - 예: 손님이 영화 정보를 요청하는 메뉴를 선택하는 것과 같습니다.

### 2. 백엔드 서버로 연결

- **proxy\_pass http://\$BACKEND\_URL:8080;**
  - 사용자의 요청을 **백엔드 서버**로 전달합니다.
  - 백엔드 서버가 데이터를 처리한 후 결과를 사용자에게 반환합니다.
  - 예: 웨이터가 주방에 주문을 전달하고 요리를 받아오는 과정입니다.

# nginx.conf.template 설명

## 3단계: 추가 정보 전달 (요청 정보 보내기)

다음 부분은 요청에 대한 **추가 정보**를 백엔드 서버로 전달하는 설정입니다.  
이는 웨이터가 주방에 주문서를 전달할 때 추가 메모를 적는 것과 같습니다.

```
bash
```

 Copy code

```
proxy_set_header Host $host;
```

- 원본 요청의 **호스트 정보**를 전달합니다.
- 예: 손님의 식탁 번호를 주방에 알려주는 역할입니다.

# nginx.conf.template 설명

bash

 Copy code

```
proxy_set_header X-Real-IP $remote_addr;
```

- **사용자의 실제 IP** 주소를 전달합니다.
- 예: 손님의 실제 위치 정보를 주방에 전달합니다.

# nginx.conf.template 설명

bash

 Copy code

```
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

- **중간 서버를 거친 IP 주소 목록**을 추가로 전달합니다.
- 예: 손님이 여러 중간 지점을 거쳐 왔다면, 그 경로를 기록합니다

# nginx.conf.template 설명

bash

 Copy code

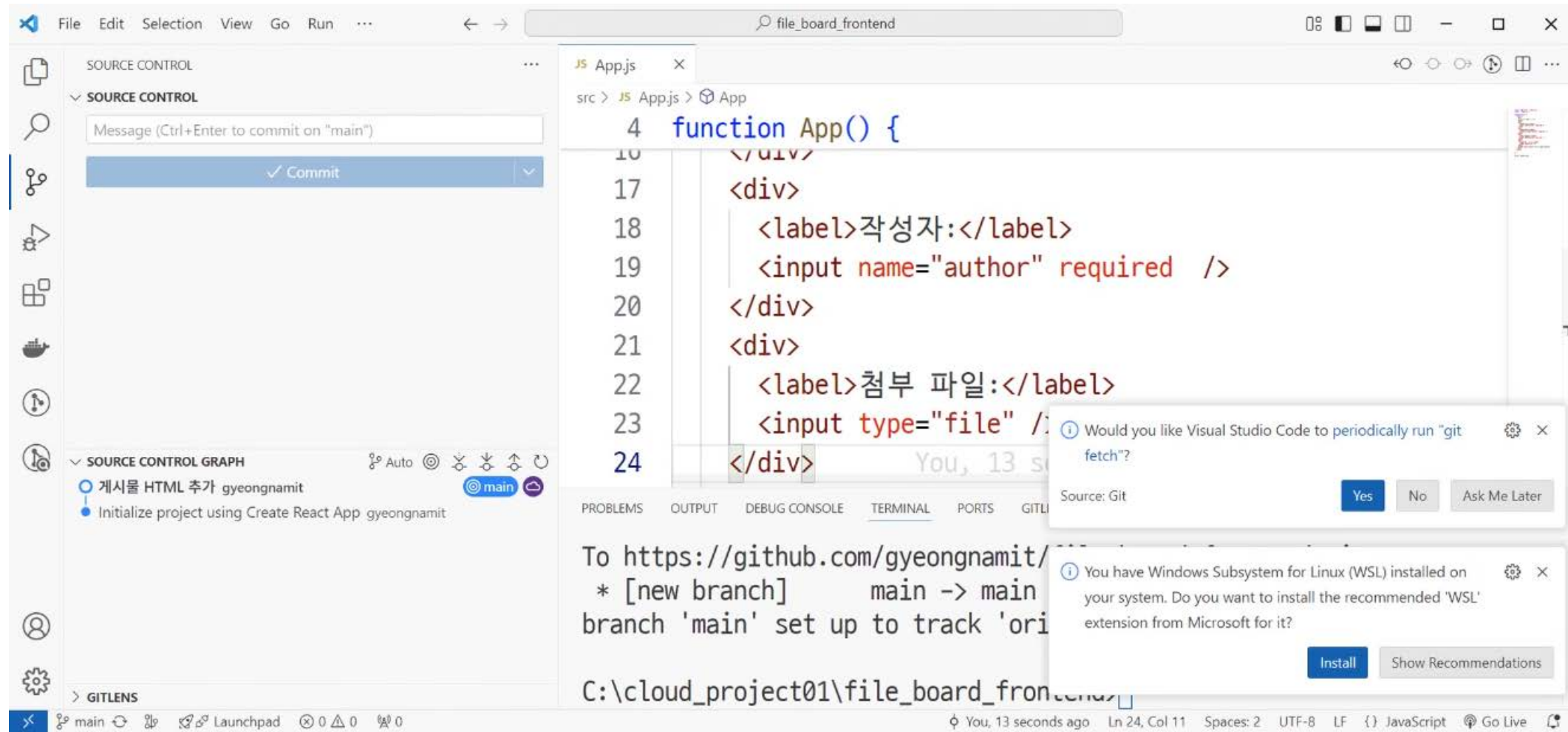
```
proxy_set_header X-Forwarded-Proto $scheme;
```

- 요청 프로토콜(HTTP 또는 HTTPS) 정보를 전달합니다.
- 예: 손님이 전화로 주문했는지, 직접 방문했는지를 기록하는 것과 비슷합니다.

## 7. Visual Studio Code 종료

# Visual Studio Code 종료

Visual Studio Code를 종료 합니다



클릭