

# 영화 추천 프론테 엔드 실행(Docker)

DevOps 엔지니어의 업무인 배포를 실습합니다

# 1.백엔드 프론트엔드와 도커

도커를 사용하여 영화 추천 프론트엔드를 배포하는 방법에 대해 알아보겠습니다.



# 백엔드는 요리사

1

## 요리사로서의 백엔드

백엔드는 음식을 만드는 요리사입니다.

2

## 레시피와 코드

요리사는 특정 레시피에 따라 음식을 만듭니다. 이 레시피가 바로 백엔드 코드입니다.

3

## 일관성 있는 환경의 중요성

하지만 요리사가 음식을 어디에서든 똑같이 만들려면, 요리 도구와 재료가 정확히 준비되어 있어야 하죠.



# 프론트엔드는 식당의 홀

식당에 들어갔을 때 손님(사용자)은 먼저 홀에 앉습니다.

홀에는 메뉴판(웹사이트)이 놓여 있고, 예쁜 인테리어(디자인)가 되어 있습니다.

손님이 메뉴판을 보고 음식을 주문하는 과정이 바로 **프론트엔드**의 역할입니다.

예: 웹사이트에서 버튼을 누르고 정보를 입력하는 것과 같습니다.

프론트엔드는 손님이 음식을 편리하게 주문하고, 식사를 즐기기 편하게 만들어 주는 부분입니다.

쉽게 말해, 프론트엔드는 손님과 직접 상호작용하는 서비스의 '얼굴'입니다.



# 도커는 이동형 식당

## 도커의 역할

도커는 백엔드(요리사) 프론트엔드(홀)가 음식을 만들고 서비스하기 위해 필요한 모든 도구, 재료, 환경을 담아 두는 이동형 식당입니다.

## 일관성 보장

이 식당은 어디로 옮기든 요리사가 같은 음식을 만들고 서비스 할 수 있도록 필요한 모든 것을 포함하고 있습니다.

## 포장 상자의 내용물

필요한 조리기구(라이브러리), 식재료(Java, 데이터베이스 등), 식당 메뉴, 식당 의자, 테이블 등이 식당에 있습니다.

## 2. Dockerfile, Docker Image, Docker Container

이 세 가지 요소는 Docker의 핵심 구성 요소로, 애플리케이션의 일관된 개발과 배포를 가능하게 합니다.

# Dockerfile

## 설계 스케치

Dockerfile은 식당을 어떻게 구축할지 설명하는 간략한 설계 스케치입니다.

## 필요한 것들

필요한 가구, 장비(오븐, 싱크대 등), 재료(칼, 접시 등)를 간략히 명시합니다.

## 설계도 생성

이 스케치를 바탕으로 실제 식당을 구현하는 완성된 설계도(Docker Image)를 생성합니다.

# Docker Image

## Docker Image

Docker Image는 Dockerfile 스케치를 바탕으로 만들어진 상세하고 완성된 식당 설계도입니다.

## 상세한 설계

식당의 모든 세부 사항을 포함하며 실제 동작 가능한 식당을 설계 합니다

## 실제 식당 생성

실제 식당(Docker Container) 생성에 사용됩니다.

# Docker Container

## 실제 식당

Docker Container 는 Docker Image 설계도에 따라 구축된 실제 식당입니다.

## 요리 및 작업

요리(애플리케이션 실행) 및 기타 작업을 수행할 수 있습니다.

## 생성 및 삭제

작업 종료 후 컨테이너를 삭제하고 필요시 언제든지 새 컨테이너를 생성할 수 있습니다.



# Dockerfile, Docker Image, Docker Container

## Dockerfile

"식당 설계 스케치"와 같습니다. 식당을 설계하기 위해 필요한 기본 아이디어와 재료를 적어놓은 초안입니다.

## Docker Image

"완성된 식당 설계도"입니다. 스케치를 바탕으로 정확히 완성된 식당 설계도입니다

## Docker Container

"실제로 사용 중인 식당"입니다. 설계도를 기반으로 만들어진 실제 식당으로, 요리를 하거나 작업을 수행하는 공간입니다.



### 3.Docker Image (설계도) 만들고 Docker hub (도서관) 에 등록

도커 이미지를 만든 후 도커 허브에 등록하는 과정입니다.

## 3.1 Docker Hub

Docker Hub는 Docker Image(식당 설계도)를 저장하고 공유하는 온라인 저장소입니다

# Docker Hub

Docker Hub는 Docker Image(식당 설계도)를 저장하고 공유하는 온라인 저장소입니다. 개발자들은 자신이 만든 이미지를 여기에 올려 공유하고, 다른 이들은 필요한 이미지를 가져다 사용할 수 있습니다.

이번 프로젝트에서는 영화 추천 프론트엔드를 Docker Image로 만들어 Docker Hub에 저장합니다.

- **설계도 공유 및 재사용:** 누구나 쉽게 Docker Image를 사용할 수 있습니다.
- **공식 이미지 제공:** Java, Python, Node.js 등 다양한 프로그래밍 환경의 공식 이미지를 제공합니다.
- **무료 저장 및 다운로드:** Docker Hub는 이미지 저장 및 다운로드를 무료로 제공합니다.

## 3.2 Docker Hub 회원 가입

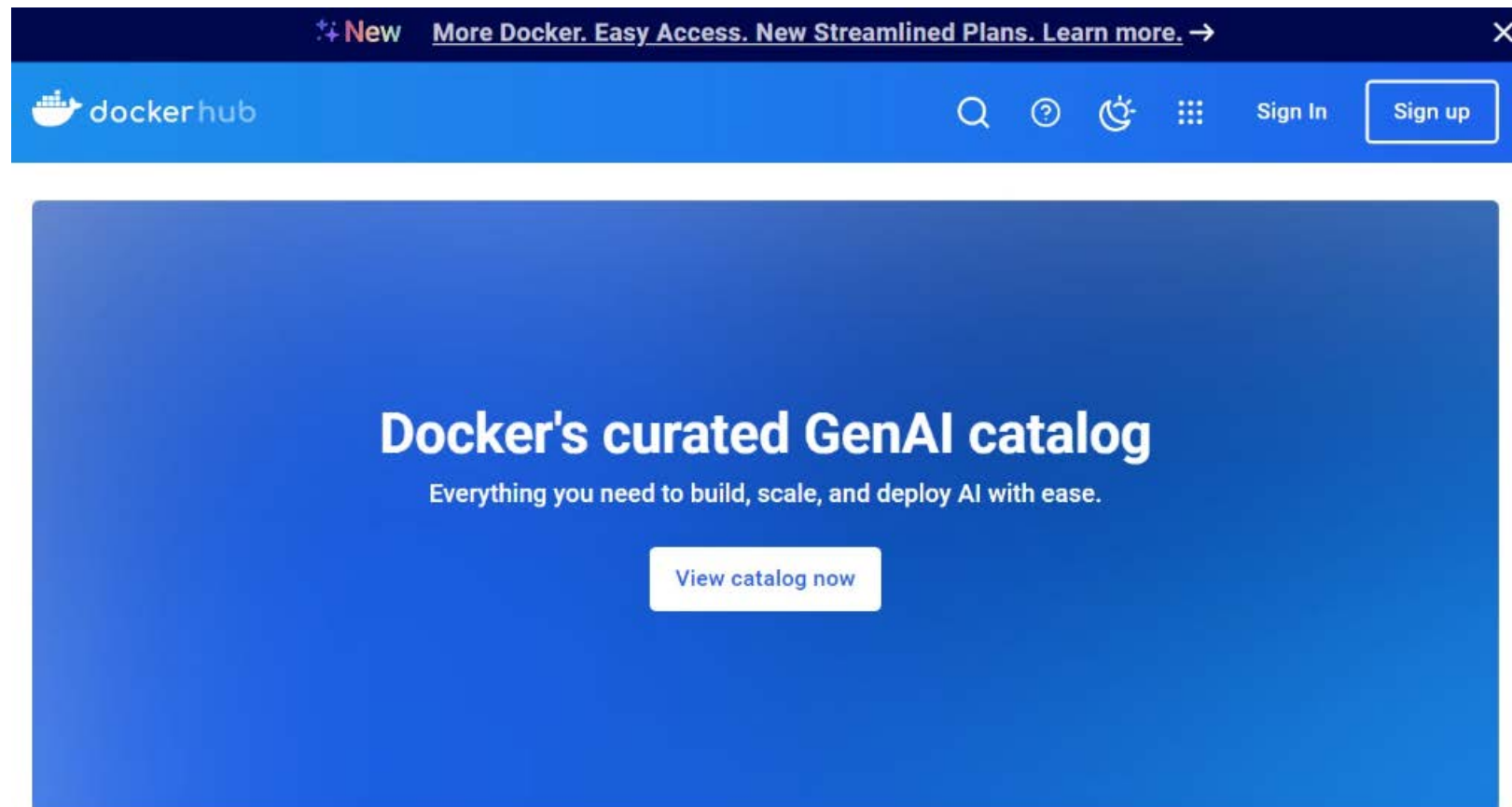
Docker Hub 회원 가입을 하면 개발자들은 자신의 이미지를 더욱 효과적으로 공유할 수 있습니다.

# Docker Hub 회원 가입

Docker Image를 도서관에 등록하기 위해서 DockerHub에 가입합니다.

<https://hub.docker.com/> 접속합니다.

Sign up을 클릭합니다



# Docker Hub 회원 가입

Docker Hub 회원 가입을 합니다



## Create your account

We suggest signing up with your work email address.

☐ Send me occasional product updates and announcements.

OR

[Already have an account? Sign in](#)

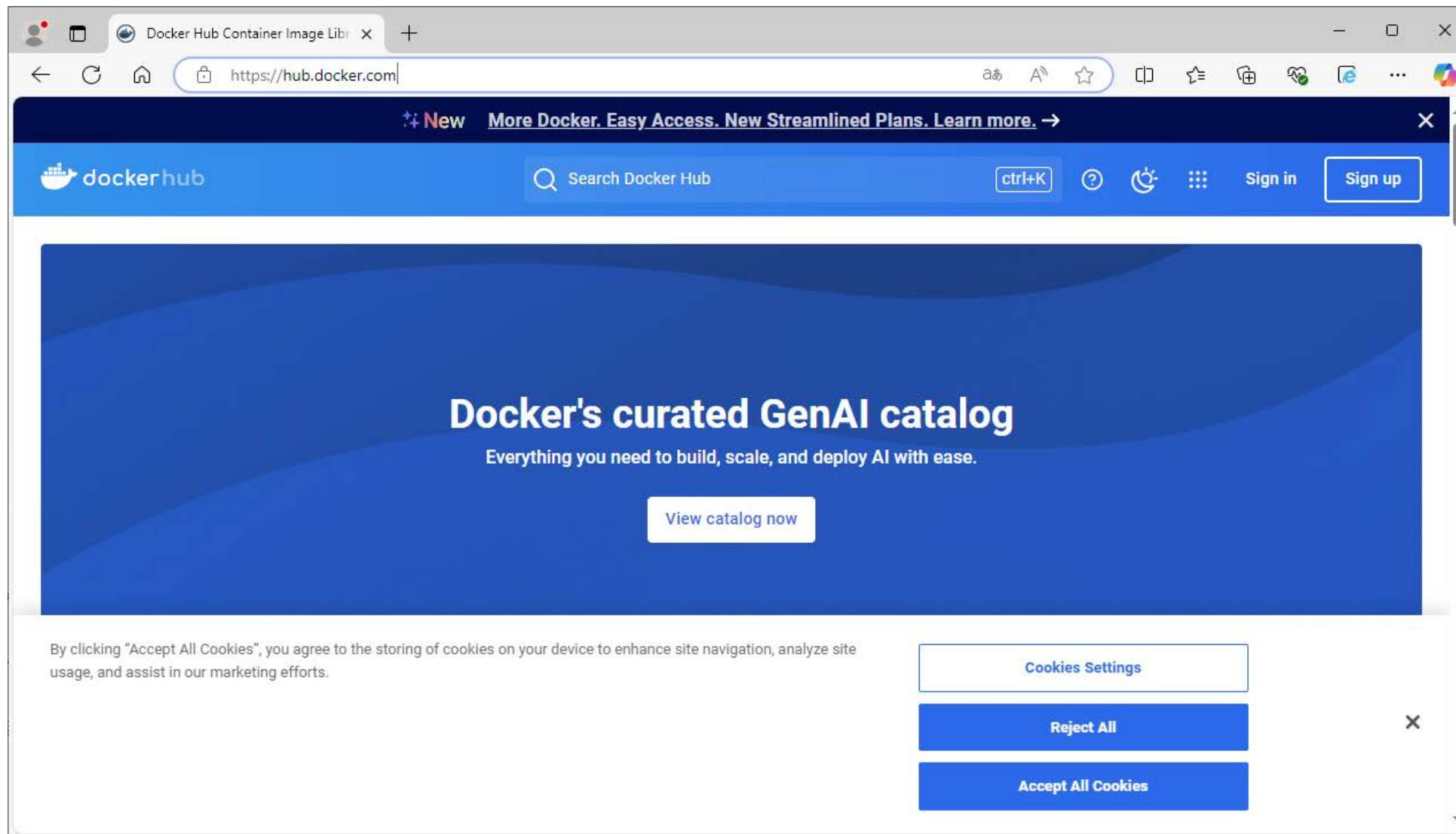
## 3.3 Docker Hub 회원명 확인

Docker Hub 회원명을 확인 합니다



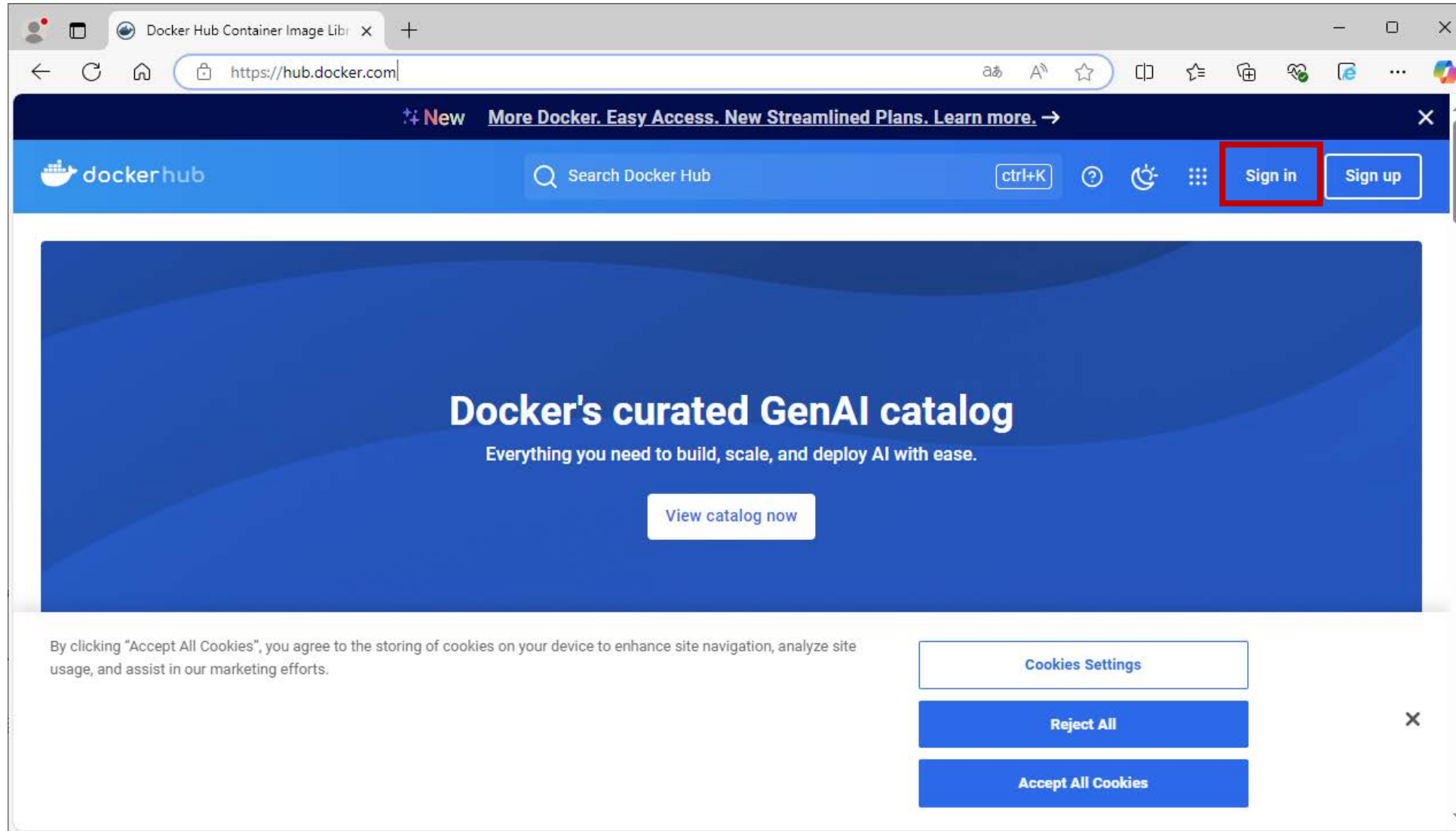
# Docker Hub 회원명 확인

<https://hub.docker.com/> 접속 합니다



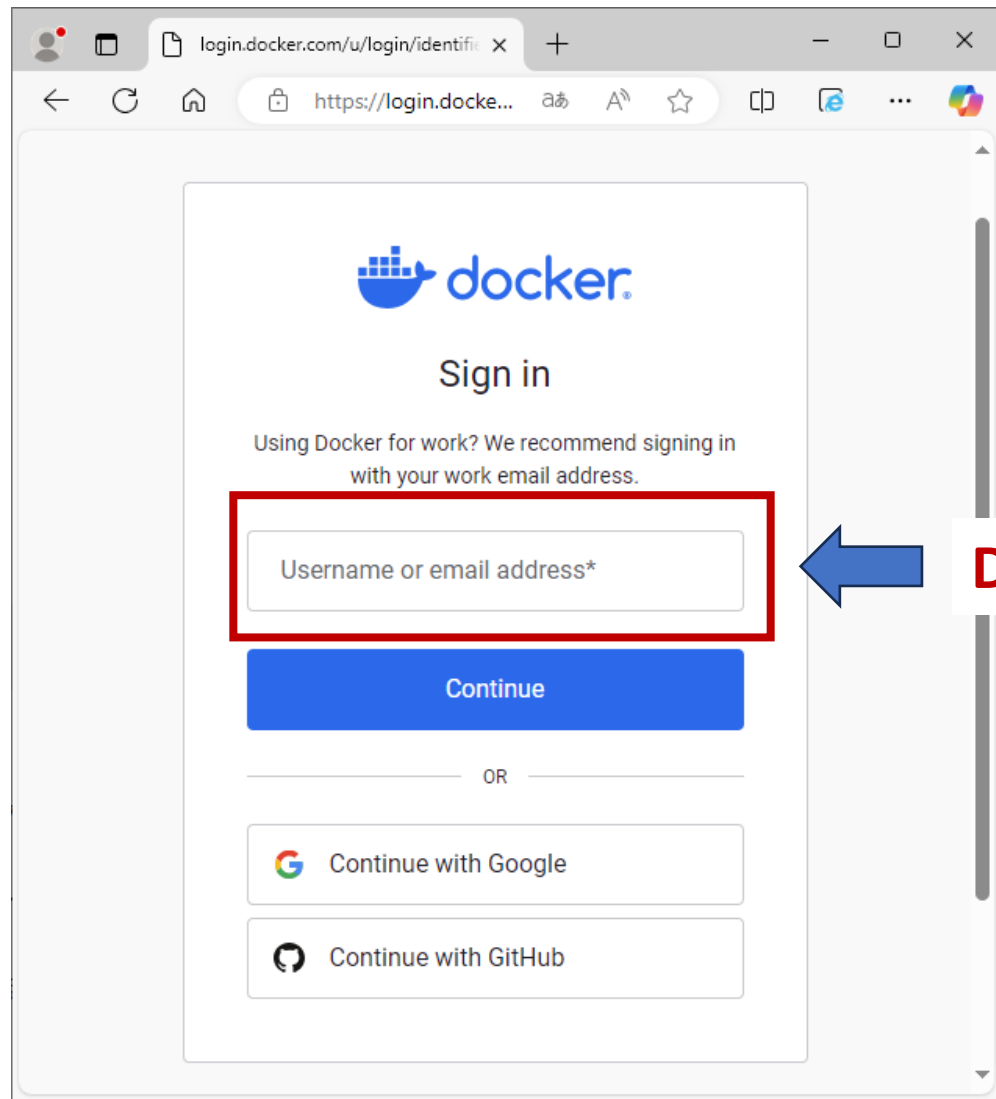
# Docker Hub 회원명 확인

sign in 을 클릭 합니다




# Docker Hub 회원명 확인

Docker Hub에 가입한 이메일을 입력 합니다



login.docker.com/u/login/identifi

https://login.docke...

 docker


Sign in


Using Docker for work? We recommend signing in with your work email address.

Username or email address\*

Continue

OR

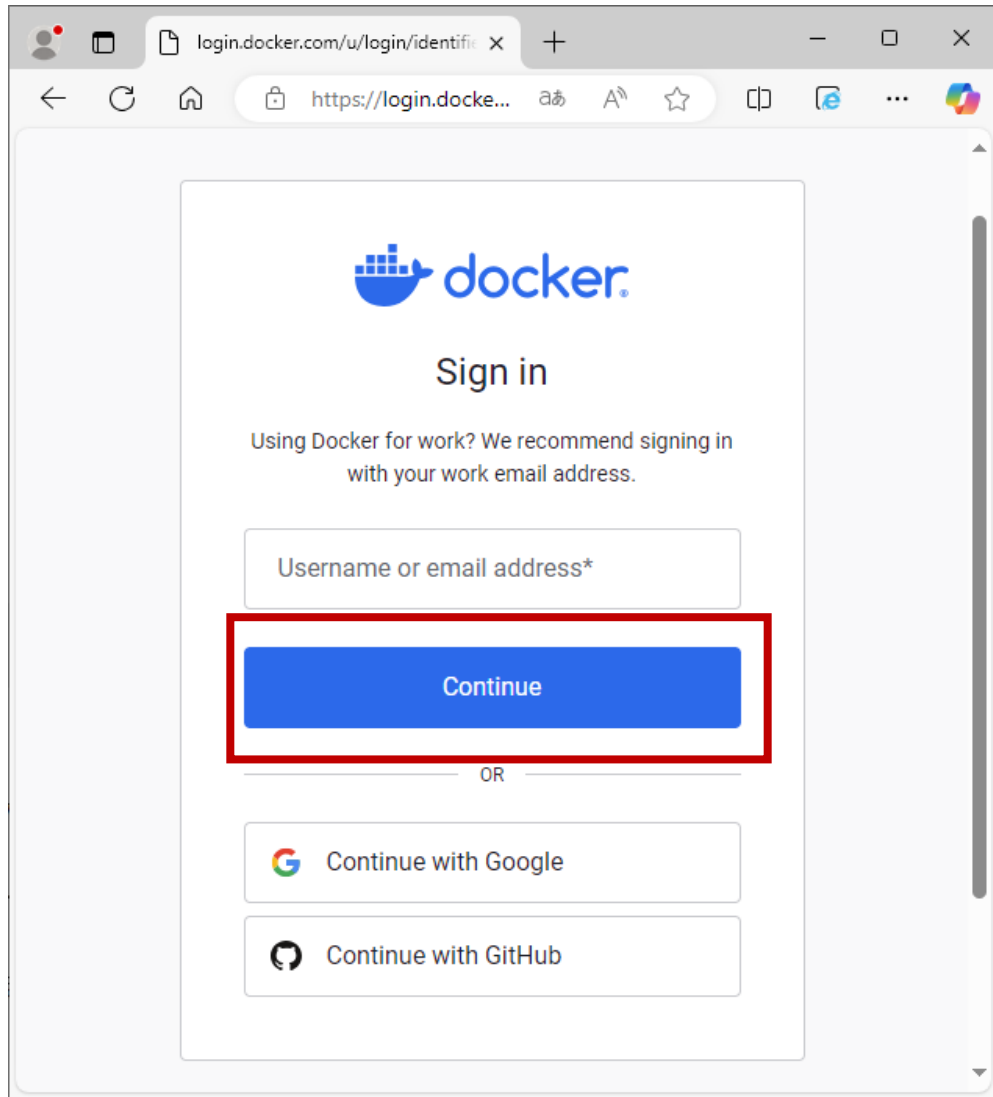
 Continue with Google

 Continue with GitHub

**Docker Hub에 가입한 이메일을 입력합니다**

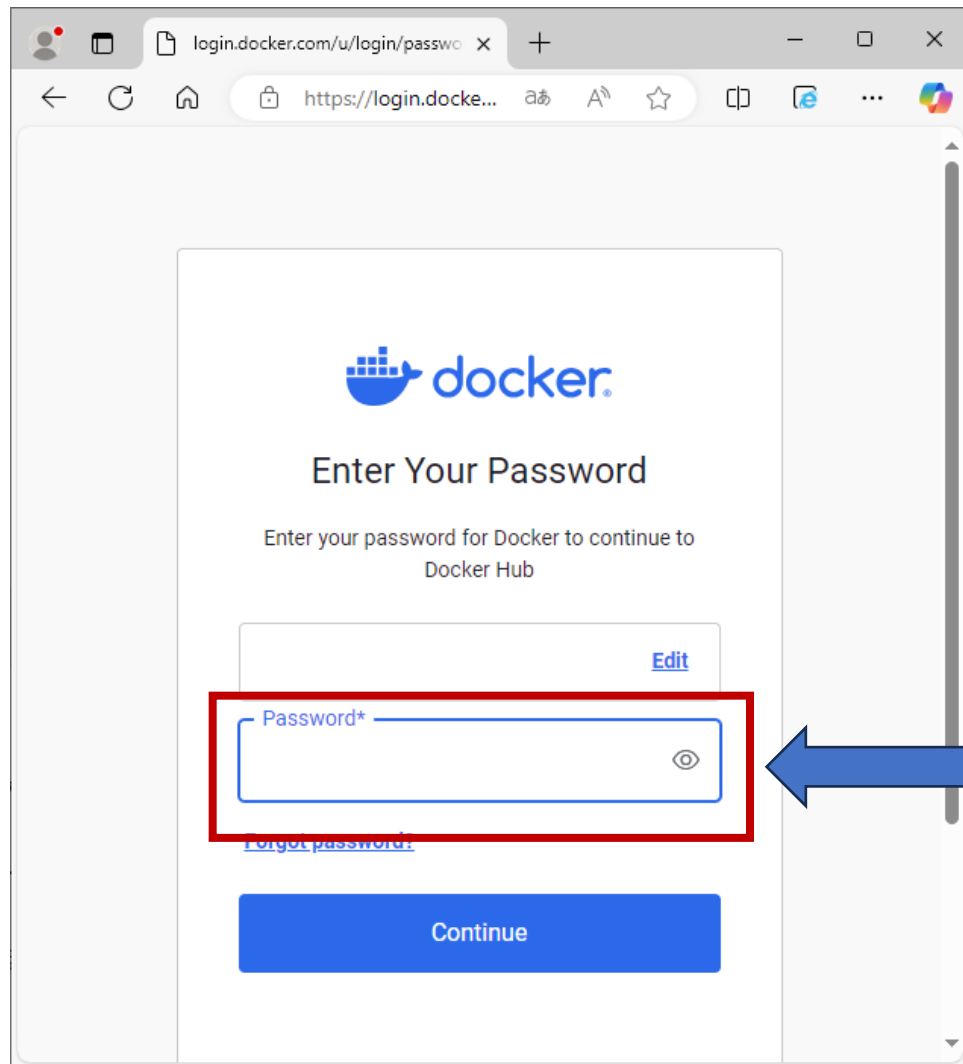
# Docker Hub 회원명 확인

Continue를 클릭 합니다



# Docker Hub 회원명 확인

Docker Hub에 가입한 비밀번호를 입력 합니다

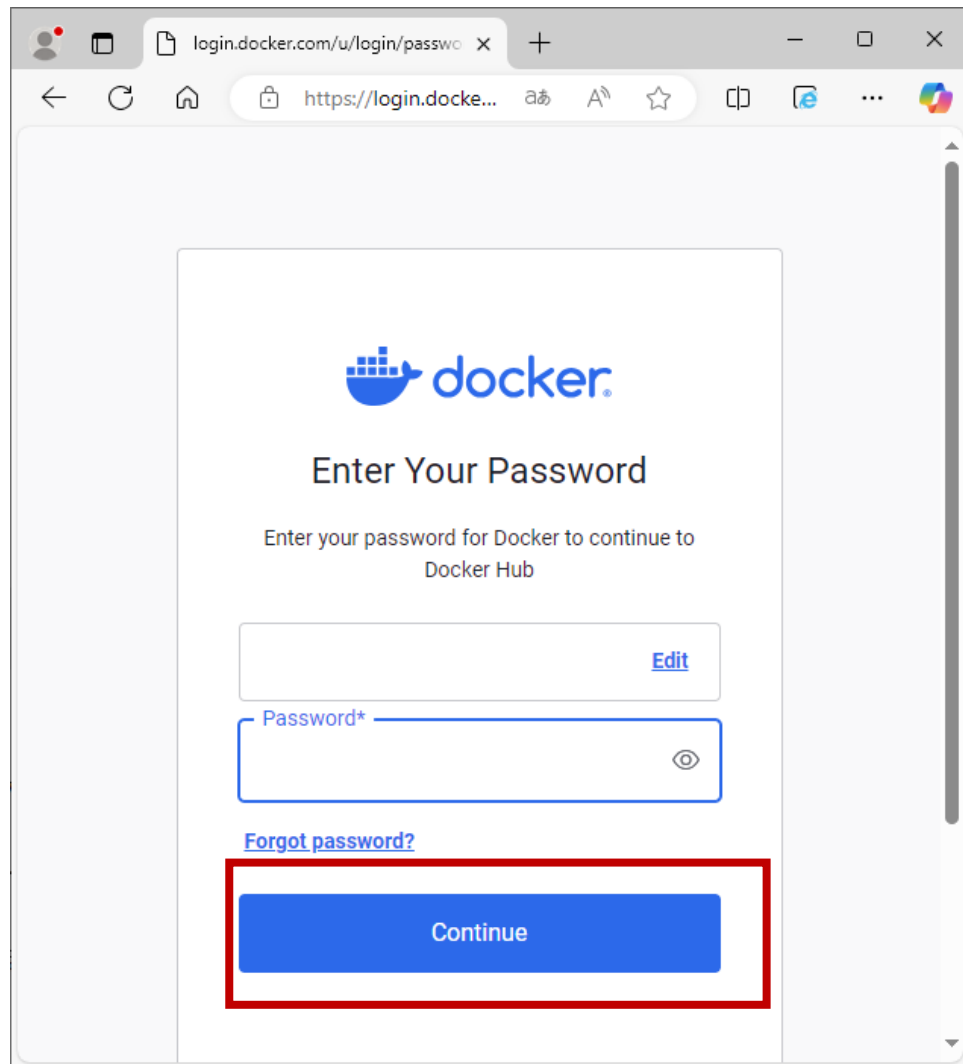


The screenshot shows the Docker Hub login page in a web browser. The page has the Docker logo at the top, followed by the text "Enter Your Password" and "Enter your password for Docker to continue to Docker Hub". Below this is a password input field labeled "Password\*" with a red rectangular box around it. To the right of the input field is a blue arrow pointing left towards the text "Docker Hub에 가입한 비밀번호를 입력합니다". Below the input field is a "Continue" button. There are also links for "Edit" and "Forget password?" near the input field.

**Docker Hub에 가입한 비밀번호를 입력합니다**

# Docker Hub 회원명 확인

Continue 버튼을 클릭 합니다



The screenshot shows a web browser window with the URL `login.docker.com/u/login/passwo`. The page displays the Docker logo and the heading "Enter Your Password". Below this, it says "Enter your password for Docker to continue to Docker Hub". There is a password input field with an "Edit" link to its right. Below the password field is a "Forgot password?" link. At the bottom of the form, there is a blue "Continue" button, which is highlighted with a red rectangular border.

[Edit](#)

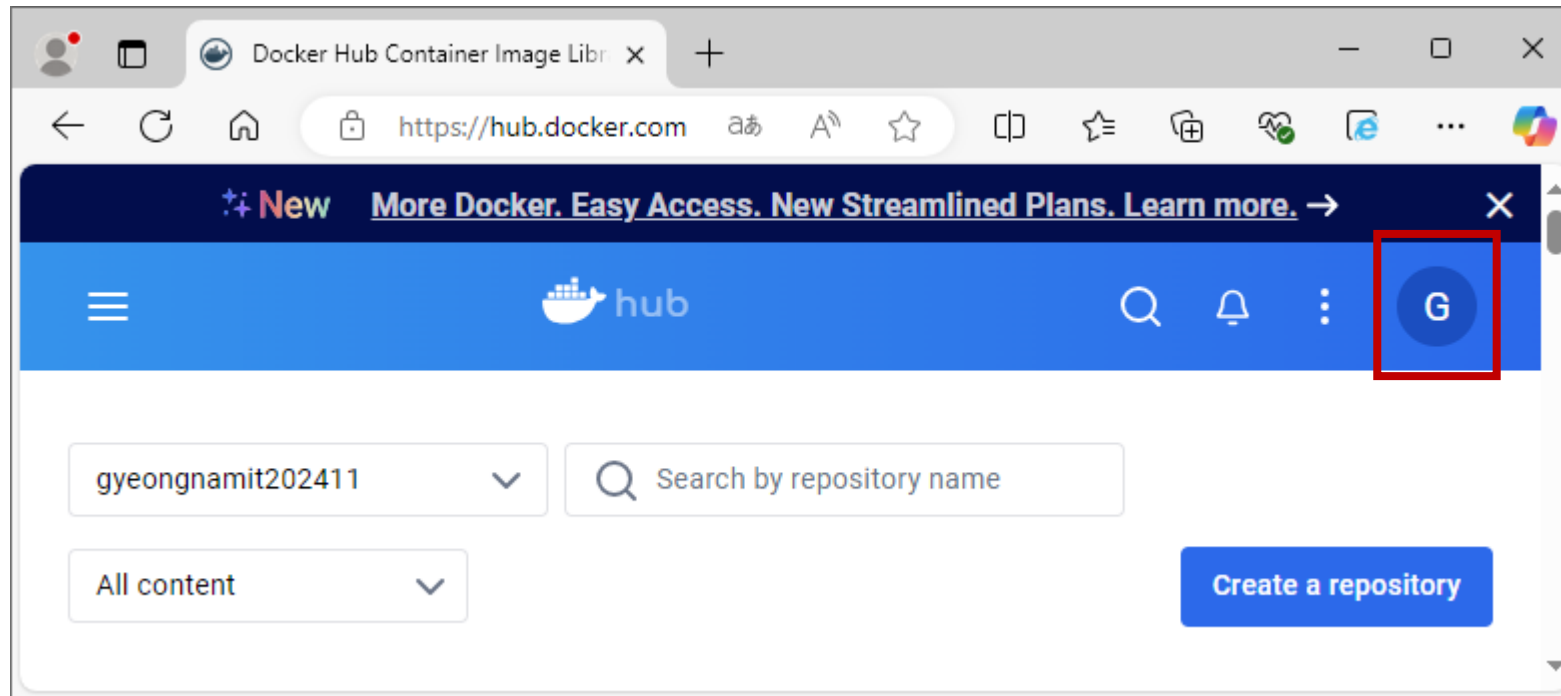
Password\*

[Forgot password?](#)

Continue

# Docker Hub 회원명 확인

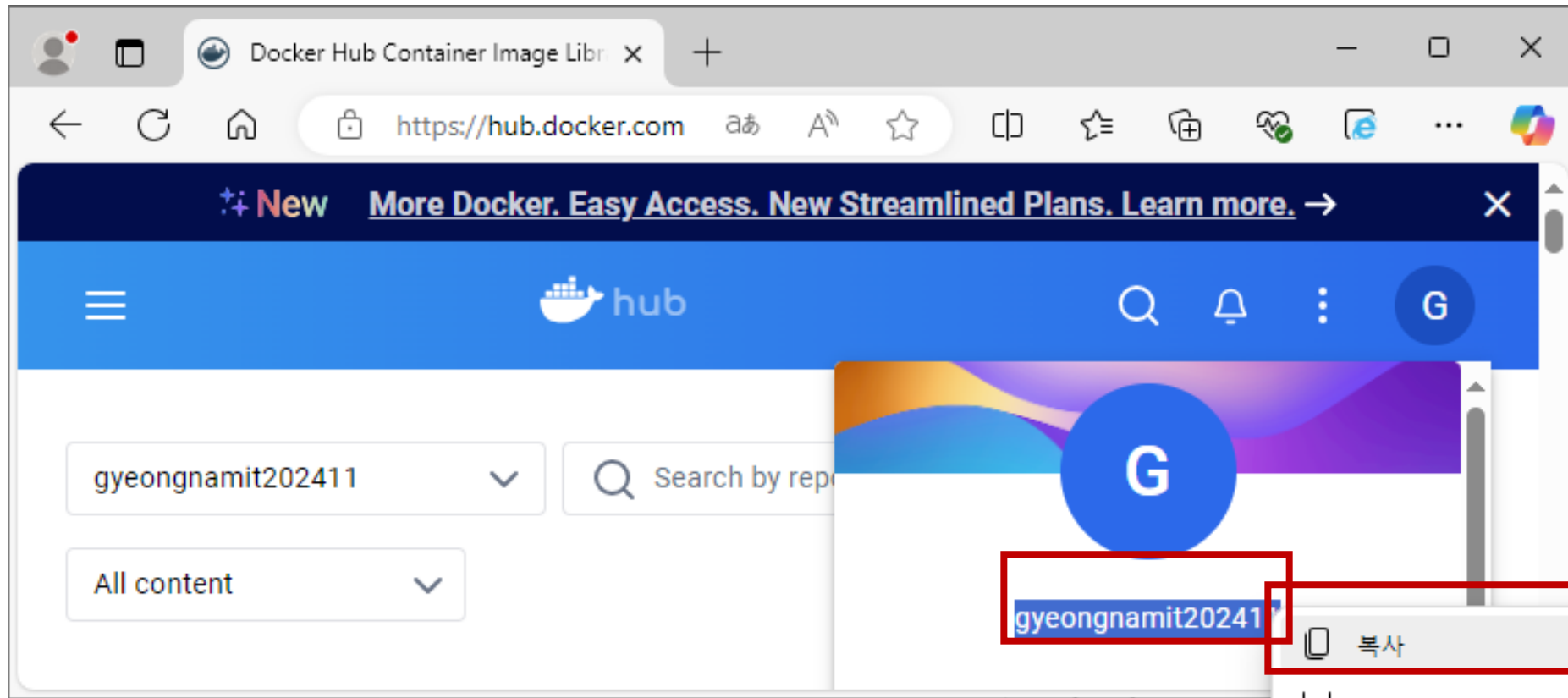
화면 오른쪽의 사용자명 첫글자 아이콘을 클릭 합니다





# Docker Hub 회원명 확인

사용자 이름을 드래그 한 후 마우스 오른쪽 버튼을 클릭해서 복사 합니다. 복사한 사용자 이름을 메모장에 붙여 넣은 후 저장 합니다



## 3.4.도커 이미지 만들기

이제 실제 영화 추천 프론트엔드를 위한 Docker Image를 만들어 보겠습니다.

Dockerfile (식당 설계 스케치)을 사용하여 Docker Image (실제 식당 설계도)를 생성합니다.

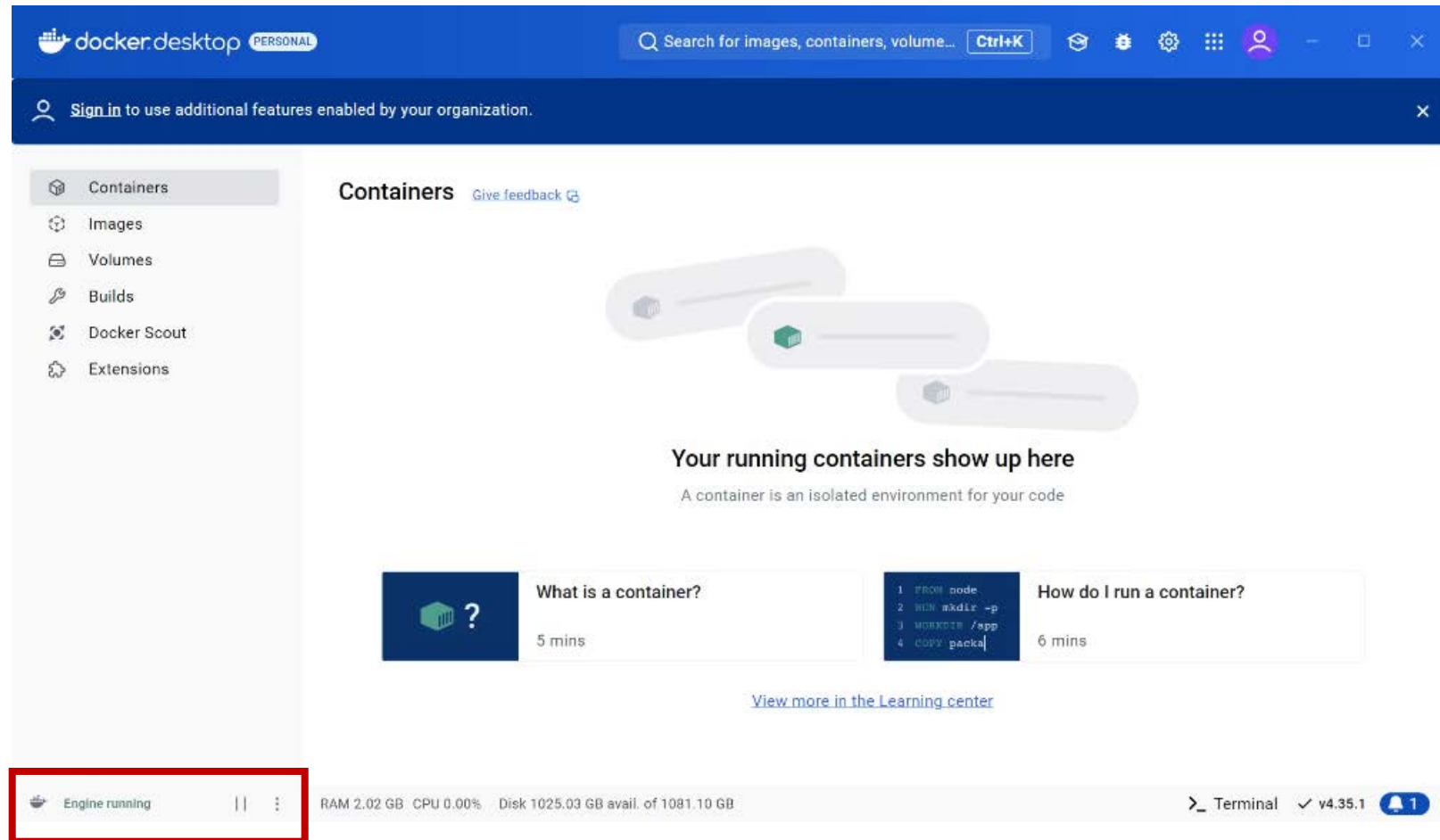
# Docker 실행

바탕화면에서 Docker Desktop 아이콘 더블클릭



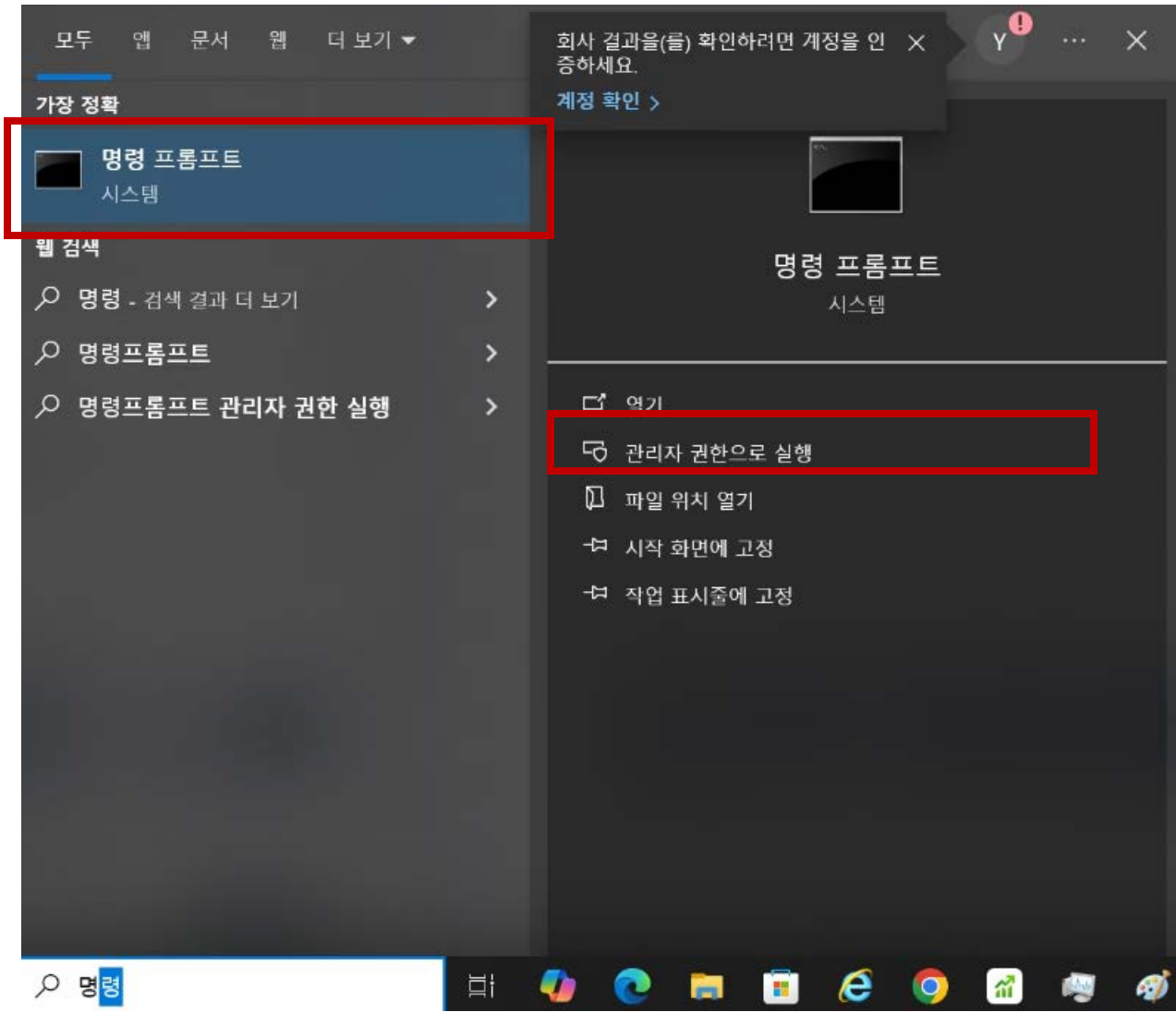
# Docker 실행

Engine running 메시지가 나올때 까지 때까지 기다리세요



# 도커 이미지 만들기

명령 프롬프트를 관리자 권한으로 실행 합니다



# 도커 이미지 만들기

영화 추천 프론트엔드 프로그램을 구현한 C:\movie\_project01\movie\_recommend\_frontend\_step1 로 이동합니다.



```
C:\> 명령 프롬프트
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yujin>cd c:\movie_project01


c:\movie_project01>cd movie_recommend_frontend_step1

c:\movie_project01\movie_recommend_frontend_step1>
```

# 도커 이미지 만들기

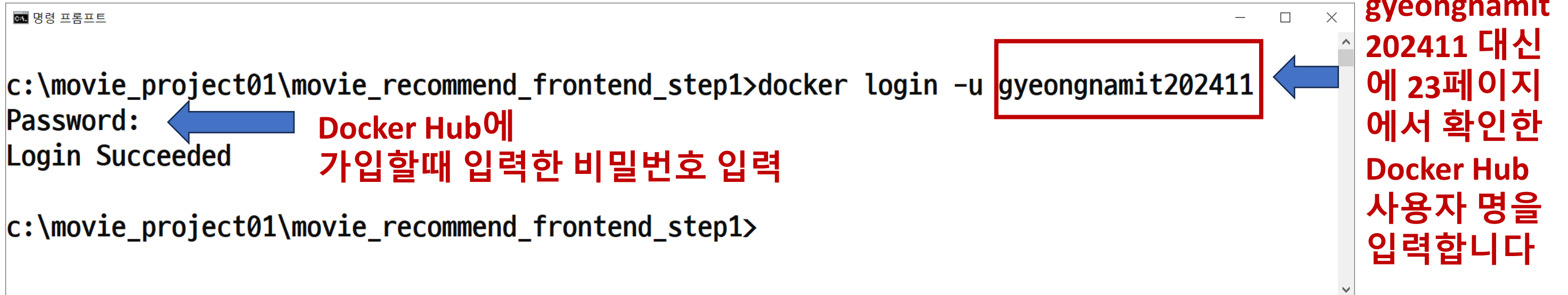
Docker Hub에 로그인 합니다.

```
bash
```

 Copy code

```
docker login -u <사용자 이름>
```

DockerHub에 가입한 사용자 명을 입력합니다.



```
c:\movie_project01\movie_recommend_frontend_step1>docker login -u gyeongnamit202411
Password:
Login Succeeded

c:\movie_project01\movie_recommend_frontend_step1>
```

**gyeongnamit  
202411 대신  
에 23페이지  
에서 확인한  
Docker Hub  
사용자 명을  
입력합니다**


**Docker Hub에  
가입할때 입력한 비밀번호 입력**



# 도커 이미지 만들기

`docker build -t gyeongnamit202411/movie_recommend_frontend_step1 .`


bash

 Copy code

```
docker build -t gyeongnamit202411/movie_recommend_frontend_step1 .
```



**gyeongnamit202411 대신에 23페이지에서 확인한 Docker Hub 사용자 이름을 입력합니다**


 관리자: 명령 프롬프트

```
c:\movie_project01\movie_recommend_frontend_step1>docker build -t gyeongnamit202411/movie_recommend_frontend_step1 .
```

## 도커 이미지 만들기

## 에러 없이 이미지가 생성 되었는지 확인

bash

 Copy code

```
docker build -t gyeongnamit202411/movie_recommend_frontend_step1 .
```

이 명령어는 Docker(도커)라는 도구를 이용해 프로그램이 실행되는 환경(이미지)을 만드는 작업을 수행합니다.

쉽게 말하면, 햄버거를 만들기 위한 레시피(Dockerfile)를 사용해서 포장된 완성품(도커 이미지)을 만드는 과정입니다.

- **docker build**

- 도커에서 새로운 이미지(환경)를 만드는 명령어입니다.
- 예: 요리법(Dockerfile)을 따라 요리를 조리하는 과정입니다.

- **-t (태그 이름 지정)**

- -t 옵션은 새로 만든 이미지에 이름과 태그를 붙이는 역할을 합니다.
- 예: 만든 요리에 이름표를 붙이는 것과 같습니다.
- 여기서 gyeongnamit202411/movie\_recommend\_frontend는 도커 이미지 이름입니다.
  - gyeongnamit202411 → 사용자 이름(요리사 이름)
  - movie\_recommend\_frontend\_step1 → 프로그램 이름(요리 메뉴 이름)

- **.(점)**

- 현재 폴더에 있는 Dockerfile(레시피)를 사용하겠다는 의미입니다.
- 예: 요리를 만들기 위해 현재 주방에서 재료와 도구를 가져옵니다.

# 도커 이미지 만들기 설명

- **요리법(Dockerfile) 확인**
  - 현재 폴더에서 Dockerfile을 찾습니다.이 파일에는 프로그램을 설치하고 실행하는 방법이 적혀 있습니다.
- **재료 준비 및 설치**
  - Dockerfile에 따라 프로그램에 필요한 재료(파일)와 도구를 복사합니다.
  - 예: 재료를 씻고, 손질하고, 요리를 시작합니다.
- **환경 세팅 및 포장(이미지 생성)**
  - 프로그램을 실행할 수 있는 환경을 설정합니다.
  - 설정이 끝나면 포장된 프로그램 파일(이미지)을 생성합니다.
  - 예: 포장된 햄버거를 진열할 준비를 마칩니다.

# 도커 이미지 만들기 설명

- 이 명령어는 햄버거 조리 과정과 비슷합니다.
- **레시피(Dockerfile) 확인**
  - 어떤 재료와 도구가 필요한지 살펴봅니다.
- **재료 준비 및 조리**
  - 필요한 재료(파일)와 도구(패키지)를 조립하고, 요리를 완성합니다.
- **포장 완료(이미지 생성)**
  - 완성된 햄버거(이미지)를 포장해서 판매 준비를 완료합니다.
- **이름표 붙이기**
  - 햄버거에 이름(movie\_recommend\_frontend\_step1)을 붙여서 구분합니다.


## 3.6.도커 이미지 DockerHub 등록

Docker Hub는 개발자들이 자신의 Docker Image를 쉽게 관리하고 다른 사람들과 협업할 수 있는 플랫폼을 제공합니다. 이는 마치 여러 식당의 설계도를 한 곳에 모아둔 거대한 도서관과 같습니다.

# 도커 이미지 DockerHub 등록

Docker Hub에 이미지 업로드


bash

 Copy code

```
docker push gyeongnamit202411/movie_recommend_frontend_step1
```




**gyeongnamit202411 대신에 23페이지에서 확인한 Docker Hub 사용자 이름을 입력합니다**

 관리자: 명령 프롬프트

```
c:\movie_project01\movie_recommend_frontend_step1>docker push gyeongnamit202411/movie_recommend_frontend_step1_
```



# 도커 이미지 DockerHub 등록


 관리자: 명령 프롬프트

```
c:\movie_project01\movie_recommend_frontend_step1>docker push gyeongnamit202411/movie_recommend_frontend_step1
Using default tag: latest
The push refers to repository [docker.io/gyeongnamit202411/movie_recommend_frontend_step1]
4f4fb700ef54: Mounted from gyeongnamit202411/movie_recommend_frontend
2ce963c369bc: Mounted from gyeongnamit202411/movie_recommend_frontend
a20276ab5ec8: Mounted from gyeongnamit202411/movie_recommend_frontend
53c7c67fba89: Pushed
114b86119f66: Mounted from gyeongnamit202411/movie_recommend_frontend
59b9d2200e63: Mounted from gyeongnamit202411/movie_recommend_frontend
c23b4f8cf279: Mounted from gyeongnamit202411/movie_recommend_frontend
56d2a120498c: Mounted from gyeongnamit202411/movie_recommend_frontend
c28dd04bc1fd: Mounted from gyeongnamit202411/movie_recommend_frontend
3e1e579c95fe: Mounted from gyeongnamit202411/movie_recommend_frontend
1f21f983520d: Mounted from gyeongnamit202411/movie_recommend_frontend
385b5e90091a: Mounted from gyeongnamit202411/movie_recommend_frontend
547a97583f72: Mounted from gyeongnamit202411/movie_recommend_frontend
f56be85fc22e: Mounted from gyeongnamit202411/movie_recommend_frontend
latest: digest: sha256:a6dc865c4449c392e78040c0fb7cdf41cc9bed52a6199798d5f6906a8aacc31d size: 856

c:\movie_project01\movie_recommend_frontend_step1>
```

에러 없이  
업로드 되는지 확인

bash

 Copy code


```
docker push gyeongnamit202411/movie_recommend_frontend_step1
```

이 명령어는 내 컴퓨터에 있는 도커 이미지(프로그램 실행 환경)를 인터넷의 저장소(Docker Hub)에 업로드(저장)하는 명령어입니다.

쉽게 말해, 내가 만든 요리(프로그램 환경)를 클라우드 냉장고(인터넷 저장소)에 보관하는 과정입니다.

이렇게 하면 다른 사람이나 다른 컴퓨터에서도 이 요리를 가져다 쓸 수 있습니다.

bash

 Copy code

```
docker push gyeongnamit202411/movie_recommend_frontend_step1
```

## 명령어 구성 살펴보기

- **docker push**

- 도커 이미지(완성된 프로그램)를 인터넷에 업로드하는 명령어입니다.
- 예: 햄버거를 만들어서 인터넷 냉장고(Docker Hub)에 저장합니다.

- **yeongnamit202411/movie\_recommend\_frontend\_step1**

- 업로드할 도커 이미지의 이름입니다.
- yyeongnamit202411 → 도커 허브 아이디(사용자 이름).
- movie\_recommend\_frontend\_step1 → 프로그램 이름.예:

- 'yeongnamit202411'이라는 요리사(사용자)가 만든 'movie\_recommend\_frontend\_step1'(요리 메뉴)를 인터넷에 올립니다.

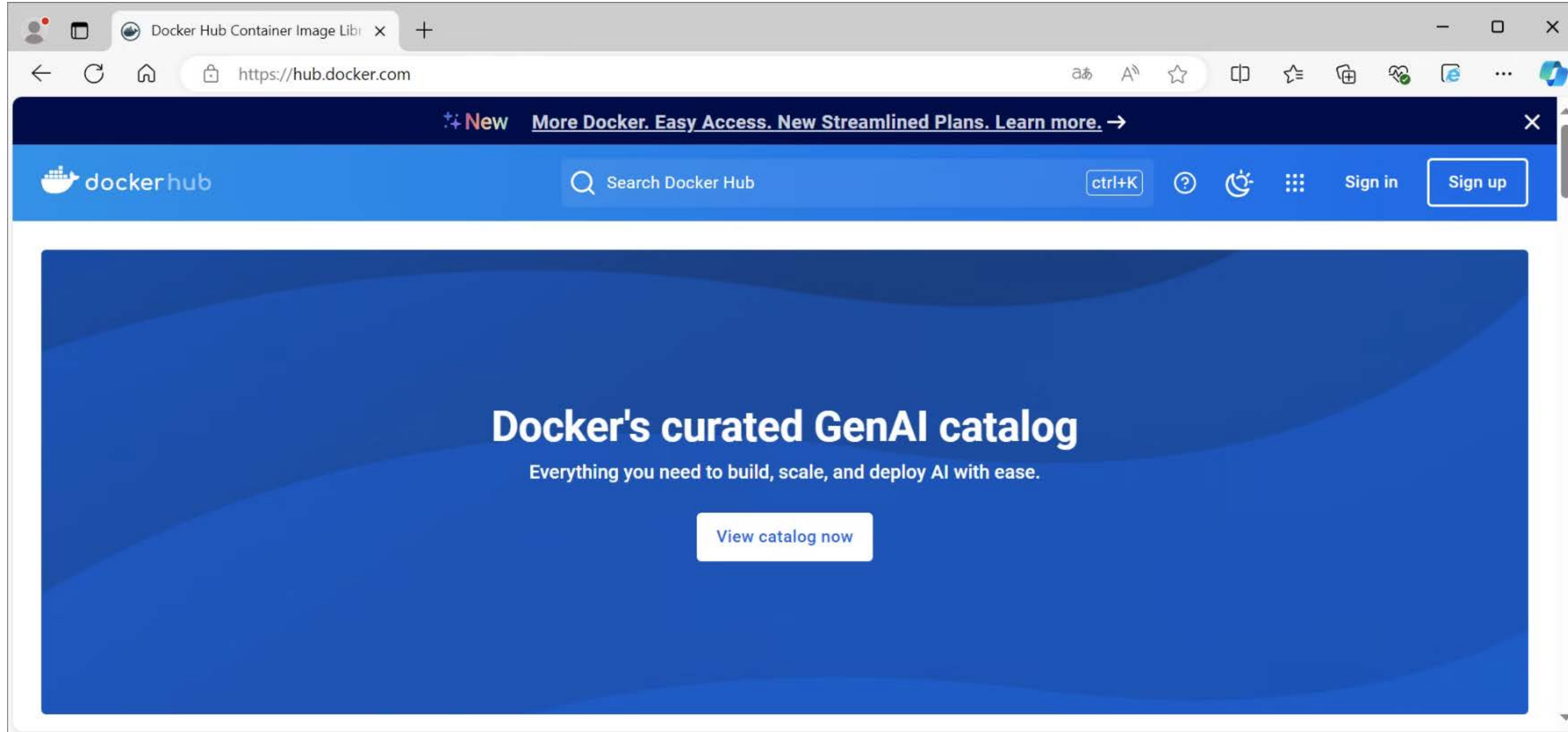
## 5.DockerHub 이미지 등록 확인

Docker 이미지가 생성되어 Docker Hub 저장소에 업로드됩니다. 이는 CI/CD 파이프라인의 중요한 단계입니다.

Docker Hub에 접속하여 새로운 이미지가 성공적으로 업로드되었는지 확인하고, 이미지 태그와 버전이 올바르게 지정되었는지 검증합니다. 이를 통해 애플리케이션 배포를 위한 준비가 완료되었음을 확인할 수 있습니다.

# DockerHub 이미지 등록 확인

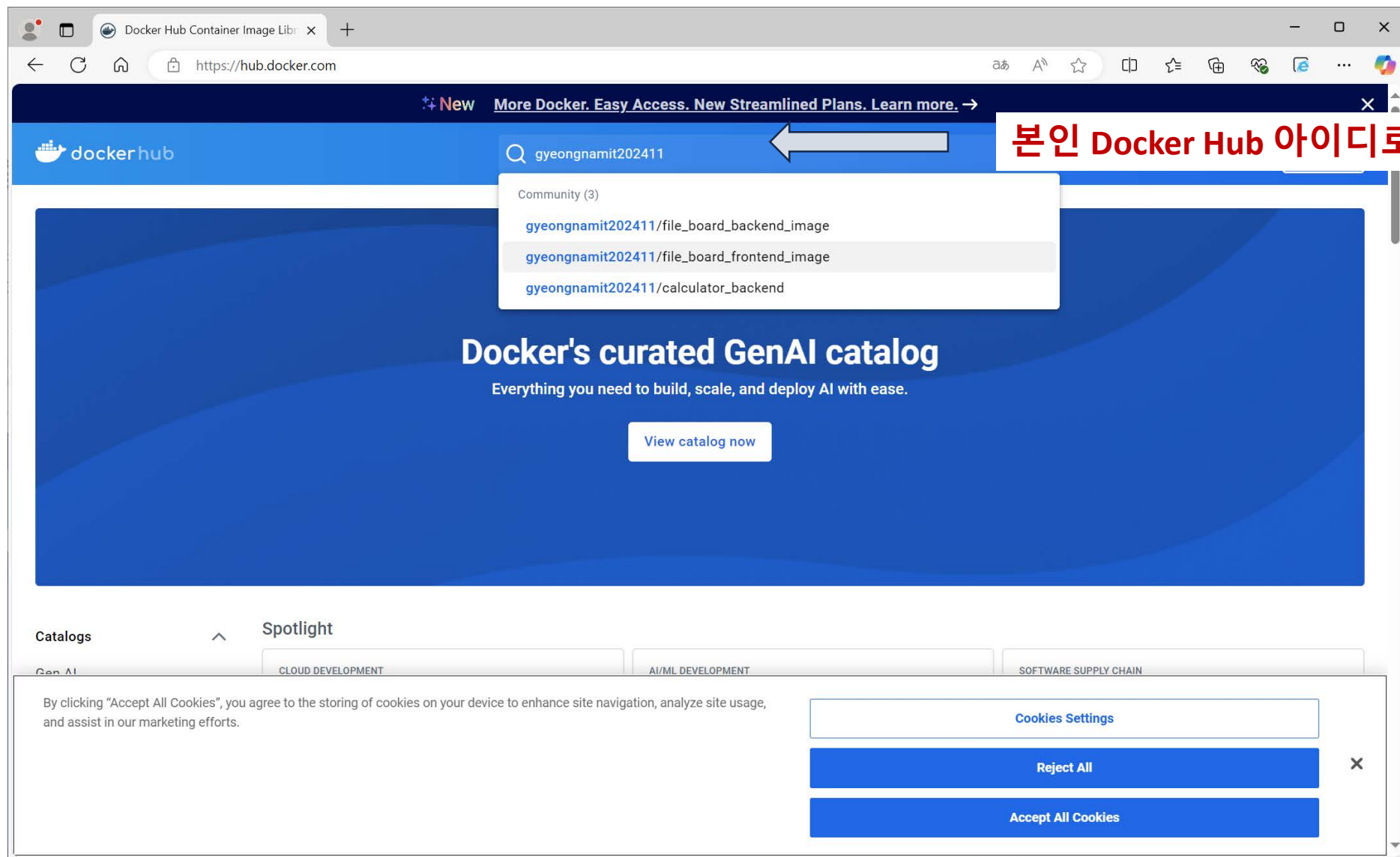
<https://hub.docker.com/> 접속 합니다



# DockerHub 이미지 등록 확인

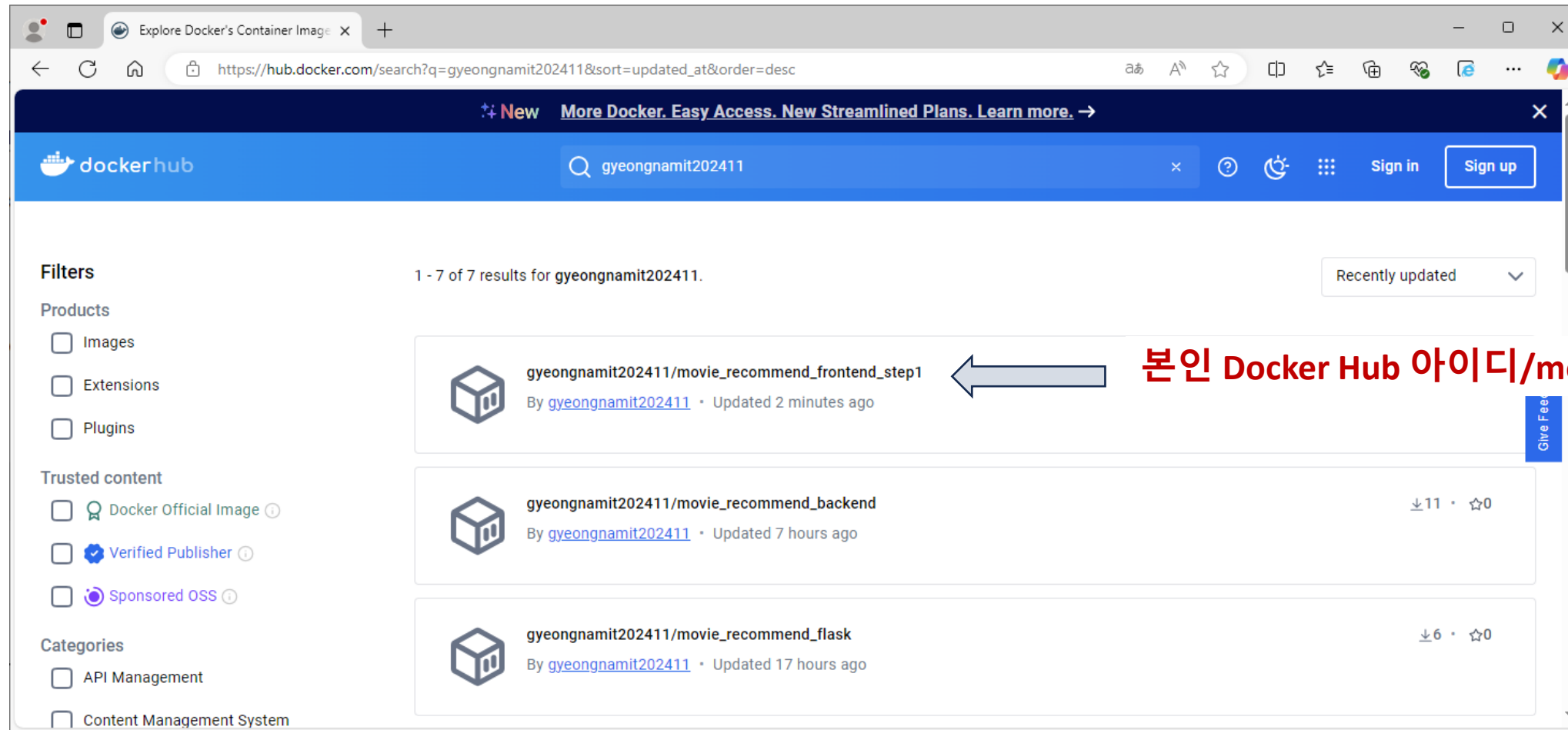
본인 Docker Hub 아이디로 검색 합니다

gyeongnamit202411 가 아이디 라면 gyeongnamit202411 로 검색 합니다



# 이미지 생성 확인

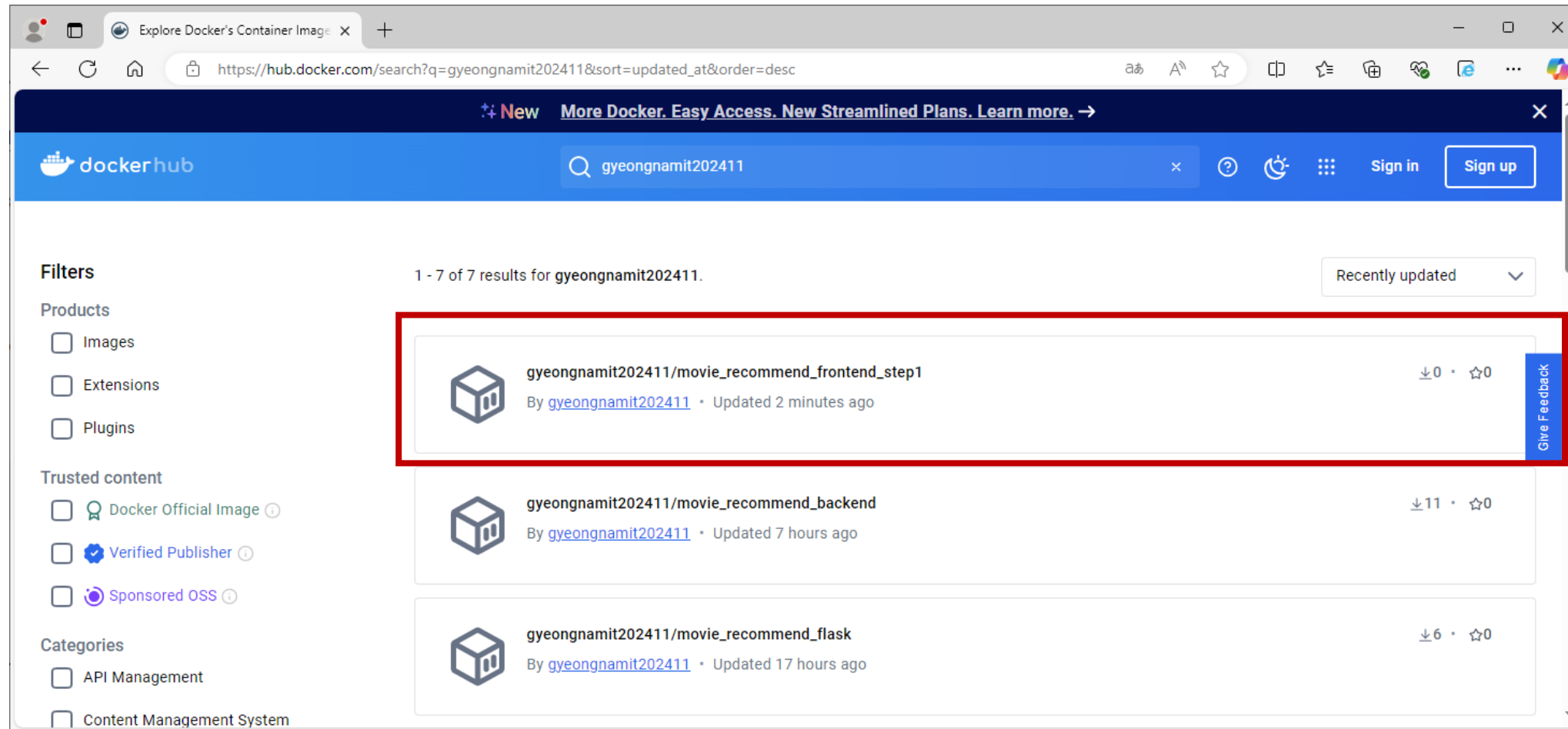
본인 Docker Hub 아이디/movie\_recommend\_frontend\_step1 가 검색 되는지 확인 합니다



본인 Docker Hub 아이디/movie\_recommend\_frontend\_step1 확인

# 이미지 생성 확인

본인 Docker Hub 아이디/movie\_recommend\_frontend\_step1 클릭 합니다





# 이미지 생성 확인

Docker 이미지가 생성 되었는지 확인 합니다

gyeongnamit202411/movie\_recor


+

← ↻ 🏠 🔒 https://hub.docker.com/r/gyeongnamit202411/movie\_recommend\_frontend\_step1 🔍 🗒️ ⭐️ 📄 📌 📧 ⋮ 🌐

New

More Docker. Easy Access. New Streamlined Plans. Learn more. →

×

 dockerhub

×

?


🔄

⋮

Sign in

Sign up

[Explore](#) / gyeongnamit202411/movie\_recommend\_frontend\_step1



gyeongnamit202411/movie\_recommend\_frontend\_step1


By [gyeongnamit202411](#) • Updated 3 minutes ago

IMAGE

☆0 ↓0

Overview

Tags



No overview available

This repository doesn't have an overview

Docker Pull Command

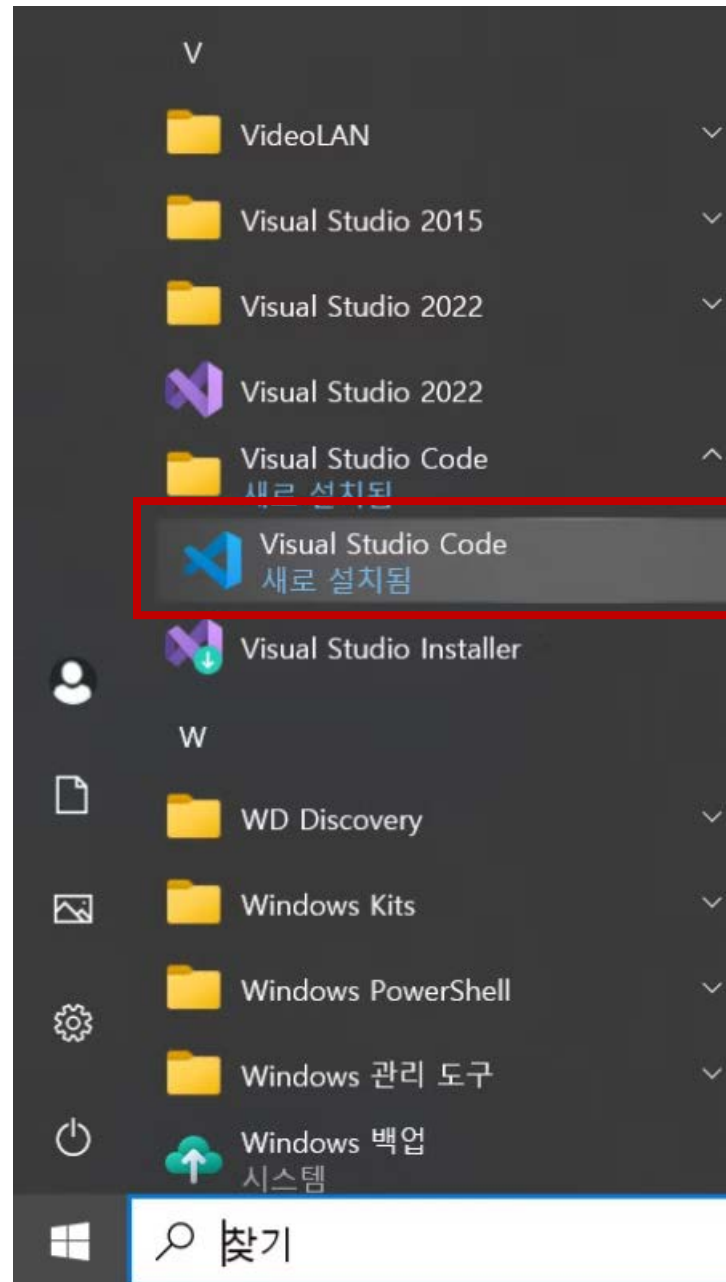
```
docker pull gyeongnamit202411/movie_r  
ecommand_frontend_step1
```

Copy

## 4.영화 추천 백엔드 엔드 실행

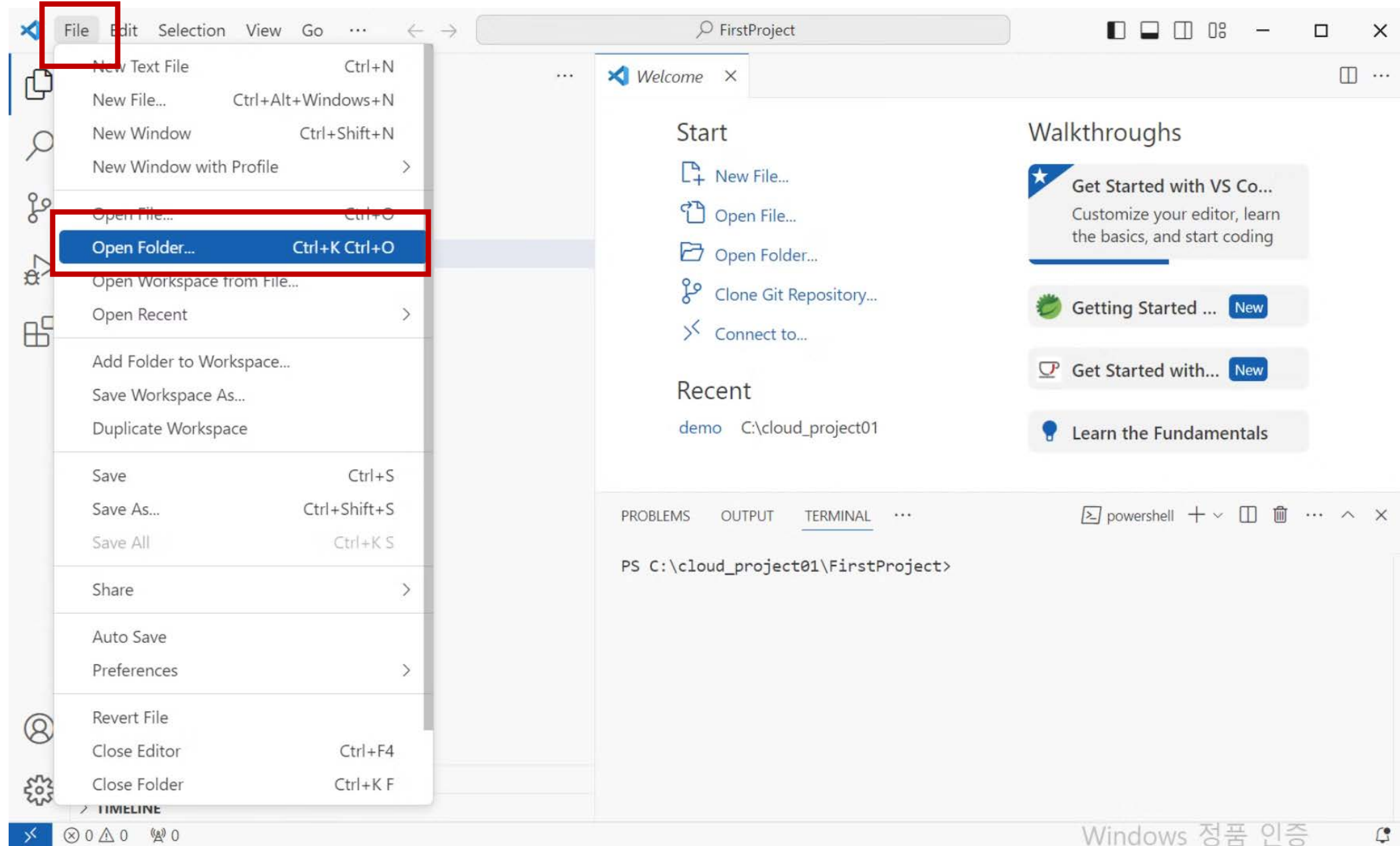
# visual studio code

visual studio code 실행



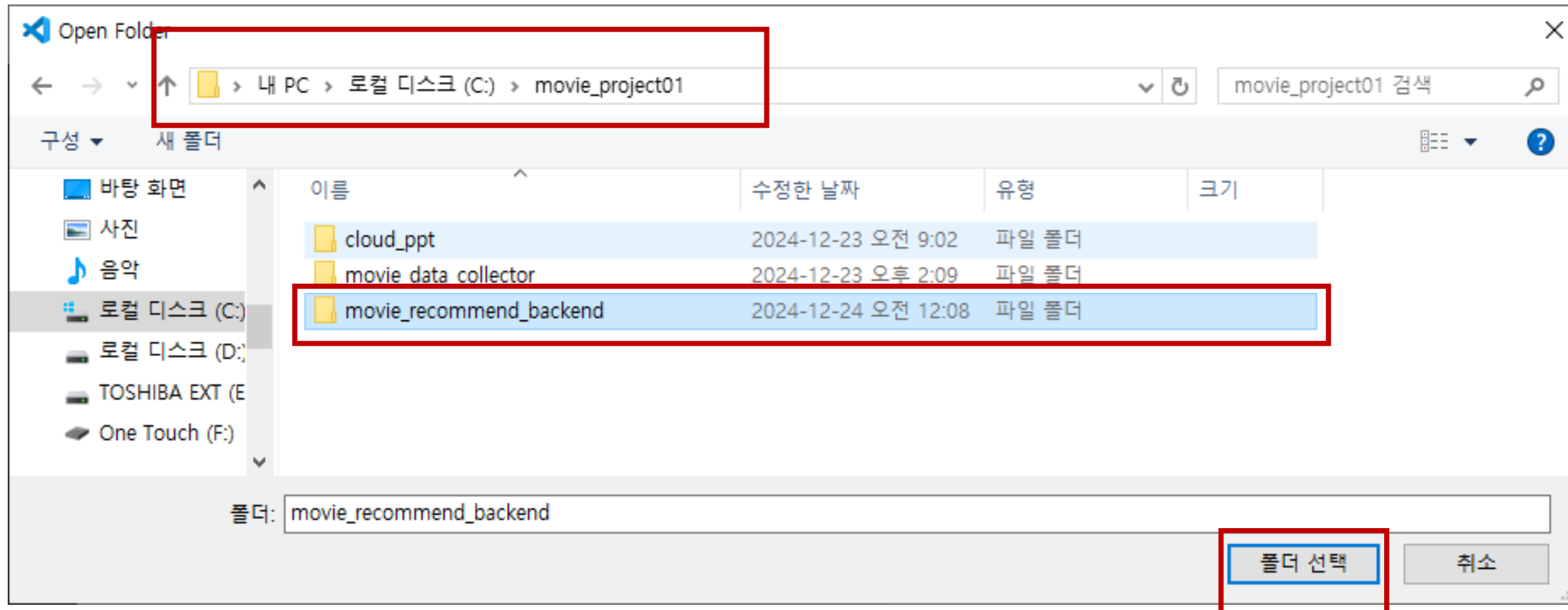
# 프로젝트 열기

File → Open Folder를 클릭합니다



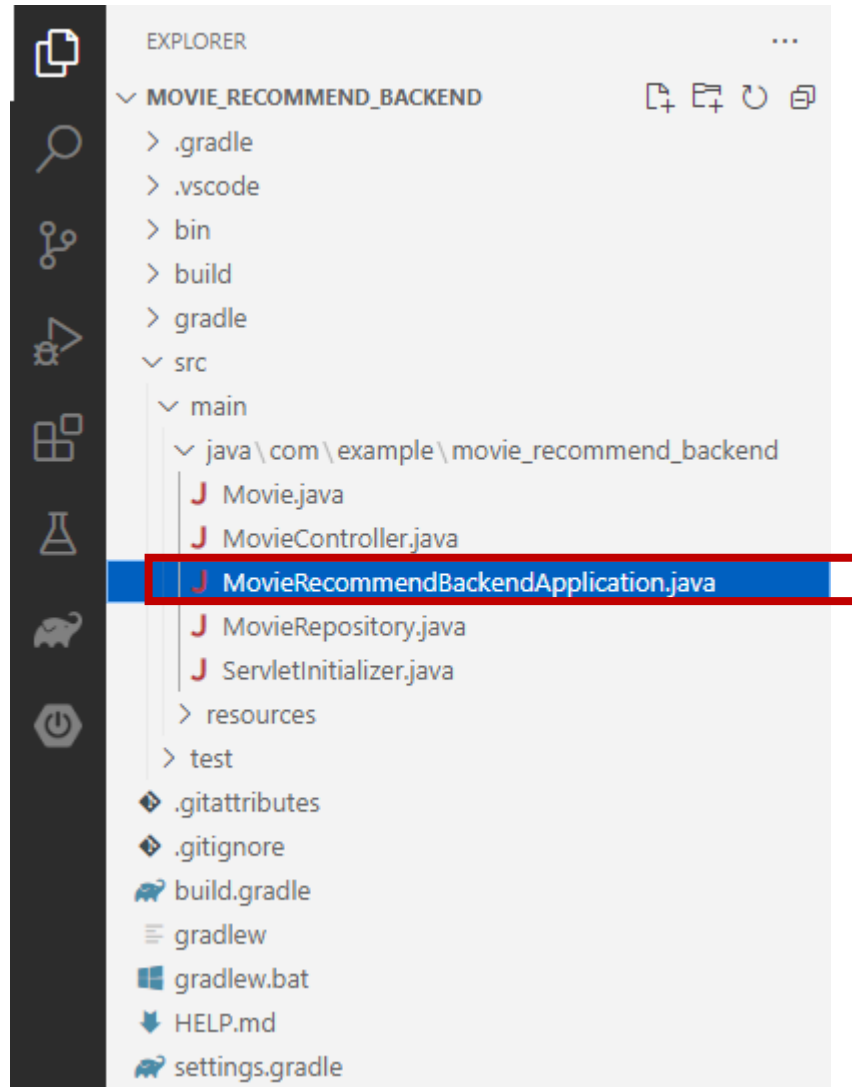
# 프로젝트 열기

c:\movie\_project01\movie\_recommend\_backend를 선택합니다



# 영화 조회 실행

MovieRecommendBackendApplication을 선택합니다



# 영화 조회 실행

MovieRecommendBackendApplication을 실행 합니다

MovieRecommendBackendApplication.java

src > main > java > com > example > movie\_recommend\_backend > MovieRecommendBackendApplication.java > Language Support for Java(TM) by Red Hat > MovieRecommendBackendApplication

```
1 package com.example.movie_recommend_backend;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class MovieRecommendBackendApplication {
8
9     Run | Debug
10     public static void main(String[] args) {
11         SpringApplication.run(MovieRecommendBackendApplication.class, args);
12     }
13 }
14
```

# 영화 조회 실행

에러 없이 실행 되는지 확인 합니다

```
C:\cloud_project01\file_board_backend>
C:\cloud_project01\file_board_backend>
C:\cloud_project01\file_board_backend> c: && cd c:\cloud_project01\file_board_backend && cmd /C ""C:\Program Files\Java\jdk-17\bin\java.exe" @C:\Users\uyujin\AppData\Local\Temp\cp_7sfjbdplsubqyzxzfho2hg1r.argfile com.example.file_board_backend.FileBoardBackendApplication "
```

```
.   _/_--'      (-)    --_/_ \
( )\_---|_|_|_|_|_|V_-||\\|\ |
\V___)|_|_|_|_|_|(|_|))))) 
=====|_|=====|_/=/_/_/_/
```

```
:: Spring Boot ::                (v3.3.6)
```

```
2024-12-03T09:07:08.487+09:00 INFO 38380 --- [file_board_backend] [ restartedMain] c.e.f.FileBoardBackendApplication : Starting FileBoardBackendApplication using Java 17.
0.12 with PID 38380 (C:\cloud_project01\file_board_backend\bin\main started by uyujin in C:\cloud_project01\file_board_backend)
2024-12-03T09:07:08.491+09:00 INFO 38380 --- [file_board_backend] [ restartedMain] c.e.f.FileBoardBackendApplication : No active profile set, falling back to 1 default pr
ofile: "default"
2024-12-03T09:07:08.567+09:00 INFO 38380 --- [file_board_backend] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devt
ools.add-properties' to 'false' to disable
2024-12-03T09:07:08.568+09:00 INFO 38380 --- [file_board_backend] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting
the 'logging.level.web' property to 'DEBUG'
2024-12-03T09:07:09.554+09:00 INFO 38380 --- [file_board_backend] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAU
IT mode.
```



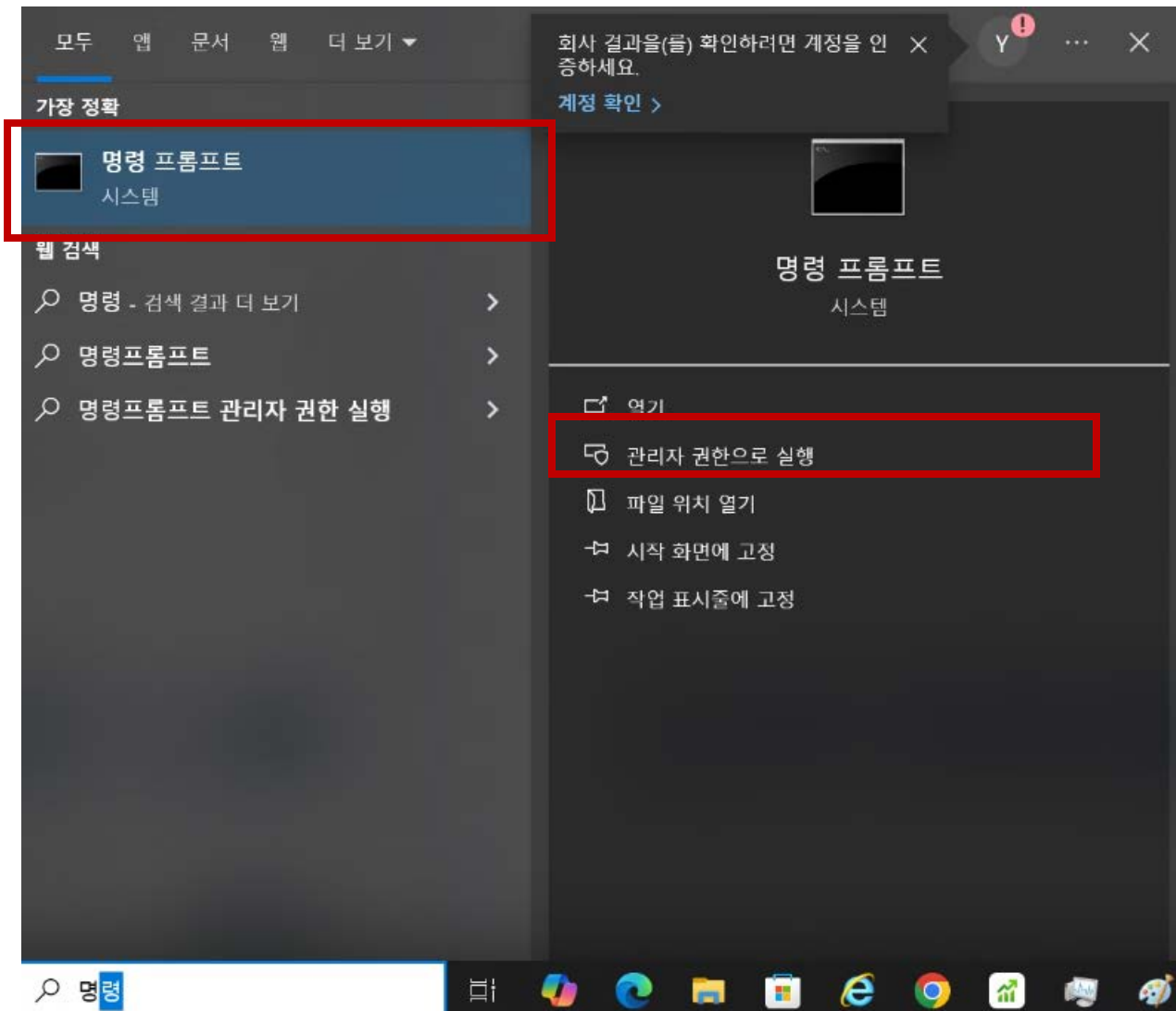
**에러 없이  
실행 되는지  
확인**



## 5.영화 추천 프론트 엔드 실행

# 영화 추천 프론트 엔드 실행

명령 프롬프트를 관리자 권한으로 실행 합니다



# 영화 추천 프론트 엔드 실행

영화 추천 프론트엔드 프로그램을 구현한 C:\movie\_project01\movie\_recommend\_frontend\_step1 로 이동합니다.



```
C:\> 명령 프롬프트
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yujin>cd c:\movie_project01

c:\movie_project01>cd movie_recommend_frontend_step1

c:\movie_project01\movie_recommend_frontend_step1>
```

# 영화 추천 프론트 엔드 실행

```
docker run -p 80:80 -it -e BACKEND_URL=host.docker.internal Github아이디/movie_recommend_frontend_step1
```

```
docker run -p 80:80 -it -e BACKEND_URL=host.docker.internal gyeongnamit202411/movie_recommend_frontend_step1_
```



gyeongnamit202411 대신에 23페이지에서 확인한 Docker Hub 사용자 이름을 입력합니다

# 영화 추천 프론트 엔드 실행

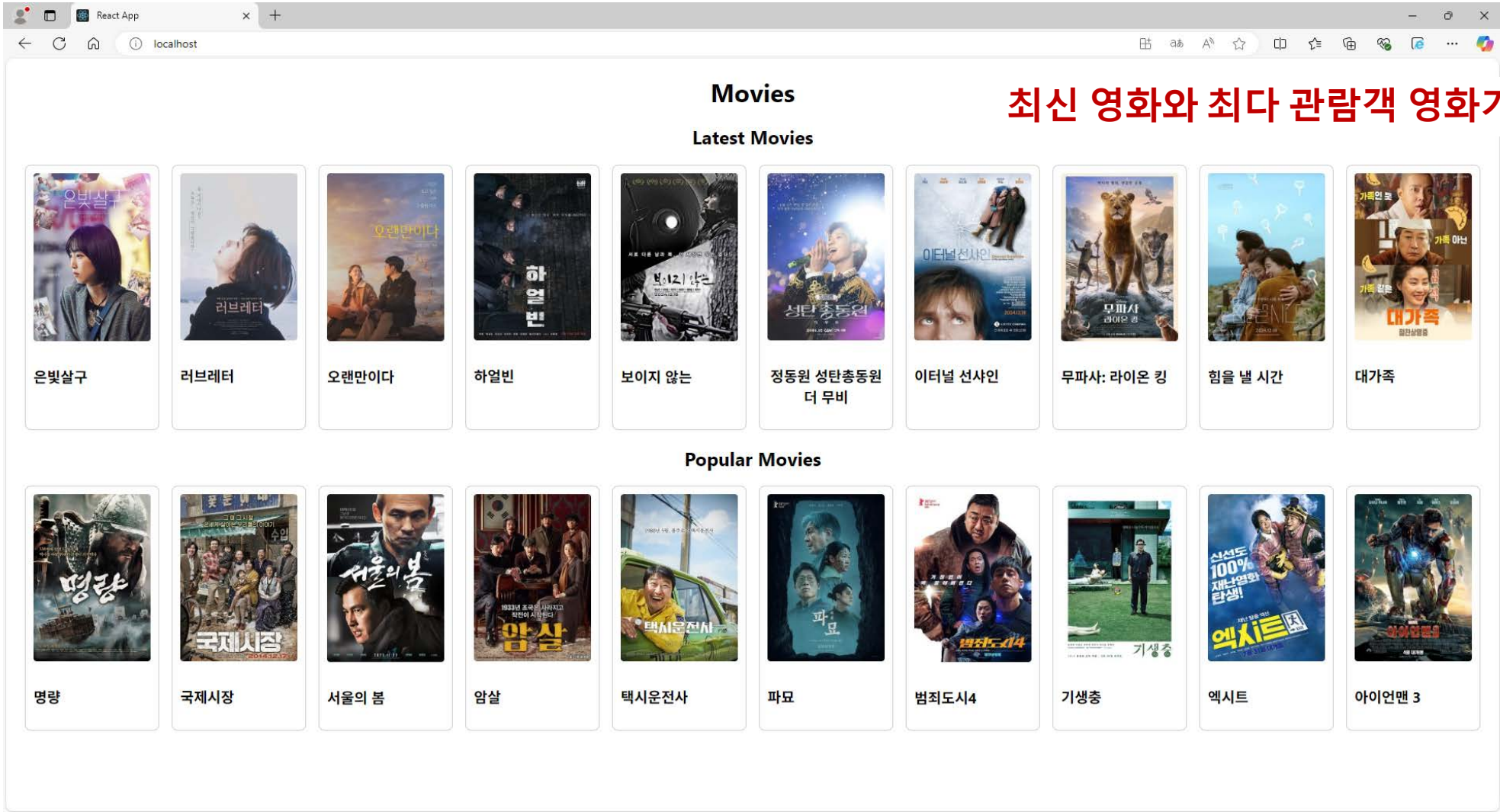
관리자: 명령 프롬프트 - docker run -p 80:80 -it -e BACKEND\_URL=host.docker.internal gyeongnamit202411/movie\_recommend\_frontend\_step1

```
c:\movie_project01\movie_recommend_frontend_step1>docker run -p 80:80 -it -e BACKEND_URL=host.docker.internal gyeongnamit202411/movie_recommend_frontend_step1
2024/12/30 11:27:02 [notice] 8#8: using the "epoll" event method
2024/12/30 11:27:02 [notice] 8#8: nginx/1.23.4
2024/12/30 11:27:02 [notice] 8#8: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r4)
2024/12/30 11:27:02 [notice] 8#8: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2024/12/30 11:27:02 [notice] 8#8: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/12/30 11:27:02 [notice] 8#8: start worker processes
2024/12/30 11:27:02 [notice] 8#8: start worker process 9
2024/12/30 11:27:02 [notice] 8#8: start worker process 10
2024/12/30 11:27:02 [notice] 8#8: start worker process 11
2024/12/30 11:27:02 [notice] 8#8: start worker process 12
2024/12/30 11:27:02 [notice] 8#8: start worker process 13
2024/12/30 11:27:02 [notice] 8#8: start worker process 14
2024/12/30 11:27:02 [notice] 8#8: start worker process 15
2024/12/30 11:27:02 [notice] 8#8: start worker process 16
2024/12/30 11:27:02 [notice] 8#8: start worker process 17
2024/12/30 11:27:02 [notice] 8#8: start worker process 18
2024/12/30 11:27:02 [notice] 8#8: start worker process 19
2024/12/30 11:27:02 [notice] 8#8: start worker process 20
```

**에러 없이  
실행 되는지 확인**

# 영화 추천 프론트 엔드 실행

웹브라우저에 <http://localhost> 입력 합니다



최신 영화와 최다 관람객 영화가 조회 되는지 확인

```
docker run -p 80:80 -it -e BACKEND_URL=host.docker.internal gyeongnamit202411/movie_recommend_frontend_step1
```

- **docker run**

- 도커 컨테이너(작은 가상 서버)를 실행하는 명령어입니다.예: 가게를 열고 손님을 맞이하는 준비를 시작합니다.

- **-p 80:80**

- 포트 연결 설정을 의미합니다.외부 손님(웹 브라우저)이 80번 문(포트)으로 요청하면,내부 컨테이너의 80번 문(포트)에 연결됩니다.
- 예: 식당 입구 문(80번)을 열고, 주방 문(80번)과 연결하는 것과 같습니다.
- 결과적으로, 사용자가 웹 브라우저에 http://localhost를 입력하면웹사이트가 열립니다.

- **-it**

- 터미널과 연결해서 명령어를 입력할 수 있도록 합니다.
- 예: 주방(서버)에 직접 들어가서 조리 상태를 확인할 수 있는 기능입니다.

- **-e BACKEND\_URL=host.docker.internal**

- 환경 변수 설정을 의미합니다.여기서 BACKEND\_URL은 백엔드 서버의 주소를 설정하는 변수입니다.
- host.docker.internal은 도커가 실행되는 컴퓨터의 주소를 의미합니다.
- 예: 주방(백엔드 서버)과 가게(프론트엔드)가 서로 소통할 수 있는 전화번호를 설정합니다.

- **gyeongnamit202411/movie\_recommend\_frontend\_step1**

- 실행할 도커 이미지의 이름입니다.gyeongnamit202411은 Docker Hub 아이디입니다.
- movie\_frontend\_step1는 웹사이트 코드가 담긴 도커 이미지의 이름입니다.:latest는 최신 버전임을 의미합니다.
- 예: 최신 버전의 메뉴판을 사용하는 식당을 연다는 뜻입니다.

## 6.영화 추천 프론트 엔드 종료



# 영화 추천 프론트 엔드 종료

관리자: 명령 프롬프트

```
c:\movie_project01\movie_recommend_frontend_step1>docker run -p 80:80 -it -e BACKEND_URL=host.docker.internal gyeongnamit202411/movie_recommend_frontend_step1
2024/12/30 11:27:02 [notice] 8#8: using the "epoll" event method
2024/12/30 11:27:02 [notice] 8#8: nginx/1.23.4
2024/12/30 11:27:02 [notice] 8#8: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r4)
2024/12/30 11:27:02 [notice] 8#8: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2024/12/30 11:27:02 [notice] 8#8: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/12/30 11:27:02 [notice] 8#8: start worker processes
2024/12/30 11:27:02 [notice] 8#8: start worker process 9
2024/12/30 11:27:02 [notice] 8#8: start worker process 10
2024/12/30 11:27:02 [notice] 8#8: start worker process 11
2024/12/30 11:27:02 [notice] 8#8: start worker process 12
2024/12/30 11:27:02 [notice] 8#8: start worker process 13
2024/12/30 11:27:02 [notice] 8#8: start worker process 14
2024/12/30 11:27:02 [notice] 8#8: start worker process 15
2024/12/30 11:27:02 [notice] 8#8: start worker process 16
2024/12/30 11:27:02 [notice] 8#8: start worker process 17
2024/12/30 11:27:02 [notice] 8#8: start worker process 18
2024/12/30 11:27:02 [notice] 8#8: start worker process 19
2024/12/30 11:27:02 [notice] 8#8: start worker process 20
```

**Ctrl+C 입력해서 종료**

## 7. Visual Studio 종료

# Visual Studio 종료

모든 Visual Studio Code 창을 종료 합니다

(영화 추천 백엔드와 프론트 엔드)

