

Flask 추천 서버 구현

Flask를 활용한 맞춤형 영화 추천 시스템을 구현해봅니다. Python 기반의 경량 웹 프레임워크인 Flask와 추천 알고리즘을 결합하여 영화 추천 서비스를 만들어보겠습니다.

1. 스프링 부트와 플라스크

스프링 부트와 플라스크는 웹 애플리케이션 개발에 있어 각각 중요한 역할을 담당합니다. 스프링 부트는 강력한 자바 기반의 메인 서버로 동작하며, 플라스크는 파이썬으로 작성된 경량 웹 프레임워크로 추천 시스템을 구현합니다.

1. 스프링부트는 '레스토랑 매장'의 역할을 합니다.

스프링부트(Spring Boot)는 고객의 요청을 처리하고 결과를 제공하는 **레스토랑 매장**에 비유할 수 있습니다.

- 고객(사용자)이 영화 추천을 요청하면,
- 스프링부트(레스토랑)는 이 요청을 처리하여 결과를 전달하는 역할을 수행합니다.

2. 플라스크는 '요리사'의 역할을 담당합니다.

영화 추천 기능은 복잡한 알고리즘과 AI 모델을 활용해야 하므로, 스프링부트만으로는 이를 처리하기가 쉽지 않습니다. 따라서, 영화 추천 기능을 전문적으로 처리하는 플라스크(Flask)라는 파이썬으로 구현된 **요리사**가 추가되었습니다.

- 스프링부트(레스토랑)는 주문을 받고,
- 플라스크(요리사)에게 영화 추천 결과를 생성해 달라고 요청합니다.
- 플라스크(요리사)는 AI 알고리즘을 활용하여 결과를 분석한 후 이를 반환합니다.
- 스프링부트(레스토랑)는 플라스크로부터 받은 결과를 최종적으로 고객에게 전달합니다.

Django와 FastAPI도 같은 역할을 수행할 수 있습니다.

플라스크 외에도 **Django**와 **FastAPI**는 플라스크와 유사하게 AI 기능을 처리할 수 있습니다.

1. Django:

- 보다 구조화된 프레임워크로, 웹 애플리케이션 개발에 적합합니다.
- 대규모 시스템이나 복잡한 기능 구현에 유리합니다.

2. FastAPI:

- 빠른 성능과 간결한 코드 작성이 강점입니다.
- AI 기능과의 연동 시 높은 처리 속도를 자랑합니다.

이처럼 필요에 따라 플라스크 외에도 Django나 FastAPI를 선택하여 유사한 구조로 구현할 수 있습니다.

플라스크를 따로 두는 이유

스프링부트와 플라스크를 분리하여 운영하는 이유는 다음과 같습니다.

1. 전문가 시스템 분리

- 2. 플라스크는 영화 추천 알고리즘과 같은 **AI 전문가 역할**을 수행합니다.
- 3. 스프링부트가 모든 기능을 처리하면 속도가 느려지거나 관리가 어려워질 수 있습니다.
- 4. 따라서 AI 관련 기능을 플라스크에 맡겨 효율성을 높이고 관리가 용이하도록 구성하였습니다.

2. 유연성 확보

- 3. 플라스크는 파이썬(Python)을 기반으로 동작합니다.
- 4. 파이썬은 AI 개발에 강력한 도구를 제공하기 때문에,
- 5. AI 기능을 파이썬으로 개발한 뒤, 스프링부트와 연동하여 활용하는 것이 보다 유연하고 효과적입니다.

시스템 처리 과정 요약

1. 고객이 영화 추천을 요청합니다.
2. 스프링부트(레스토랑)가 요청을 받아 플라스크(요리사)에게 전달합니다.
3. 플라스크(요리사)가 AI 알고리즘을 이용해 추천 결과를 생성합니다.
4. 스프링부트(레스토랑)가 결과를 받아 최종적으로 고객에게 전달합니다.

2.Flask 개요

Flask는 파이썬으로 작성된 경량 웹 애플리케이션 프레임워크입니다. '마이크로 프레임워크'라고도 불리며, 핵심 기능만을 간단하게 제공하여 개발자가 필요한 기능을 자유롭게 확장할 수 있습니다.

Flask는 무엇인가요?

Flask는 웹사이트나 프로그램을 쉽게 만들 수 있도록 도와주는 **도구 상자**입니다.
특히 **작고 가벼운 도구**라서, 빠르게 개발을 시작할 수 있습니다.

Flask를 요리에 비유하면?

웹사이트를 만든다고 생각해 봅시다.

1. Flask는 요리사

- Flask는 간단한 요리를 빠르게 준비할 수 있는 **요리사**입니다.
- 예를 들어, 영화 추천 기능 같은 특별한 요리를 전문적으로 만들 수 있습니다.

2. Flask는 작은 주방 도구 세트

- 필요한 재료와 도구만 준비되어 있어 사용이 쉽습니다.
- 덕분에 복잡한 준비 없이 바로 요리를 시작할 수 있습니다.

3. Flask는 맞춤 요리 전문가

- Flask는 하나의 특정 요리에 집중합니다.
- 예를 들어, 영화 추천 기능처럼 **특별한 기능만을** 처리하도록 설계할 수 있습니다.

왜 Flask를 사용할까요?

- **빠르고 가볍습니다.**
- → 처음 배우거나 간단한 기능을 추가할 때 적합합니다.
- **파이썬(Python)을 사용합니다.**
- → 인공지능(AI)과 데이터 분석에 강한 언어인 파이썬을 활용할 수 있습니다.
- **유연합니다.**
- → 필요에 따라 기능을 쉽게 추가하고 수정할 수 있습니다.

3. Flask 영화 추천 서비스 Git Clone 하기

지금부터 영화 추천 서비스를 위한 Flask 프로젝트를 clone하여 시작하겠습니다.

이 프로젝트는 영화 추천 알고리즘이 구현된 Flask 서버로, Spring Boot 서버와 연동하여 사용자에게 개인화된 영화 추천 서비스를 제공합니다.

Flask 영화 추천 서비스 Git Clone 하기

https://github.com/gyeongnamit/movie_recommend_flask 접속 합니다

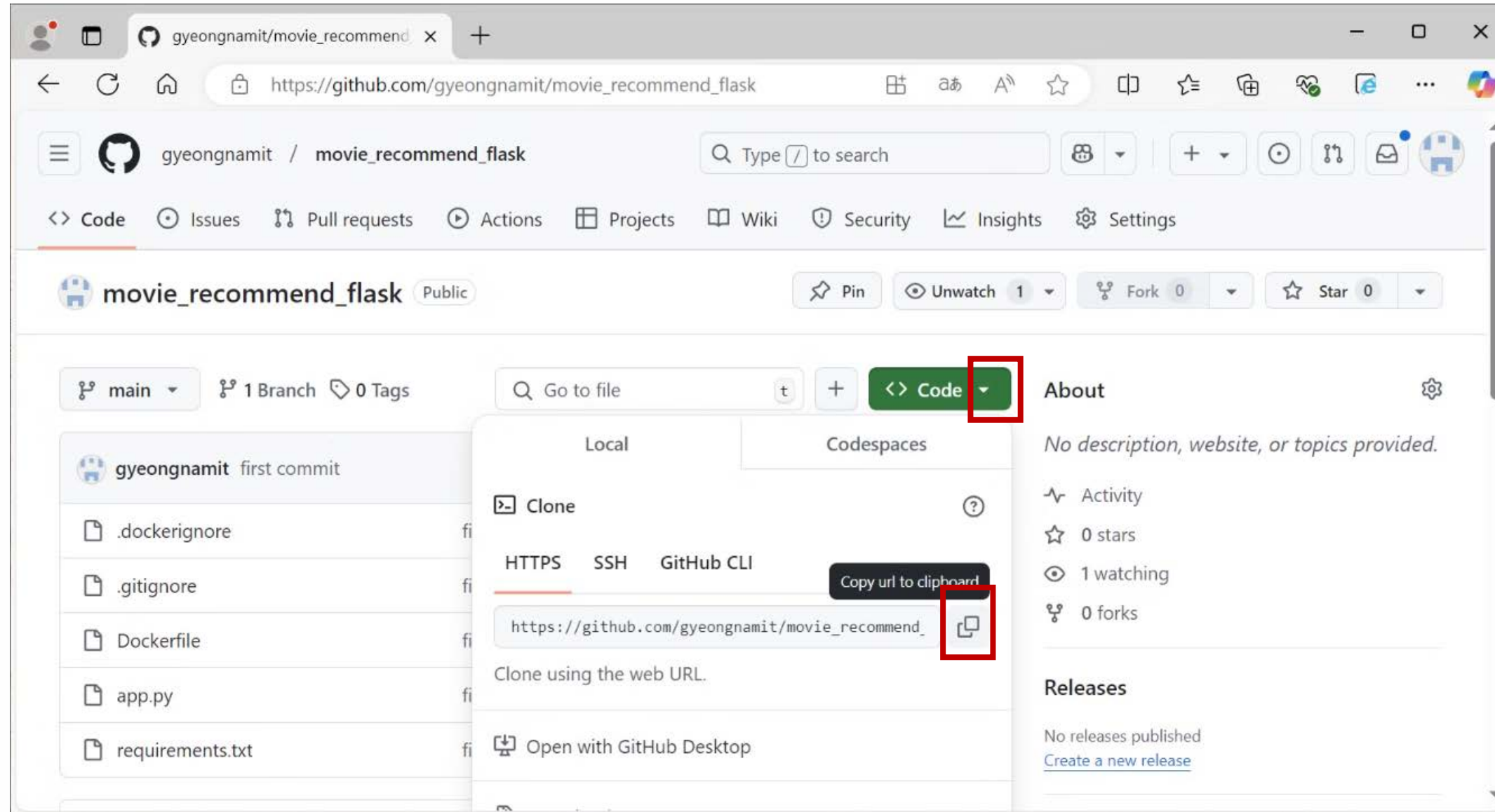
The screenshot shows the GitHub repository page for `gyeongnamit/movie_recommend_flask`. The repository is public and has 1 branch, 0 tags, and 1 commit. The commit history shows a single commit by `gyeongnamit` with the message "first commit" at 6 hours ago. The commit details show the following files:

File	Commit	Time
<code>.dockerignore</code>	first commit	6 hours ago
<code>.gitignore</code>	first commit	6 hours ago
<code>Dockerfile</code>	first commit	6 hours ago
<code>app.py</code>	first commit	6 hours ago
<code>requirements.txt</code>	first commit	6 hours ago

The right sidebar shows the repository's metadata: No description, website, or topics provided. Activity: 0 stars, 1 watching, 0 forks. Releases: No releases published. [Create a new release](#)

Flask 영화 추천 서비스 Git Clone 하기

Git의 주소를 복사 합니다



Flask 영화 추천 서비스 Git Clone 하기

- c:\movie_project01
 - movie_project01 디렉토리로 이동
- `git clone https://github.com/gyeongnamit/movie_recommend_flask.git`
 - **GitHub** 저장소에 있는 파일과 폴더를 현재 위치(c:\movie_project01)로 다운로드(복제)합니다.



```
C:\WINDOWS\system32>cd c:\movie_project01

c:\movie_project01>git clone https://github.com/gyeongnamit/movie_recommend_flask.git
```

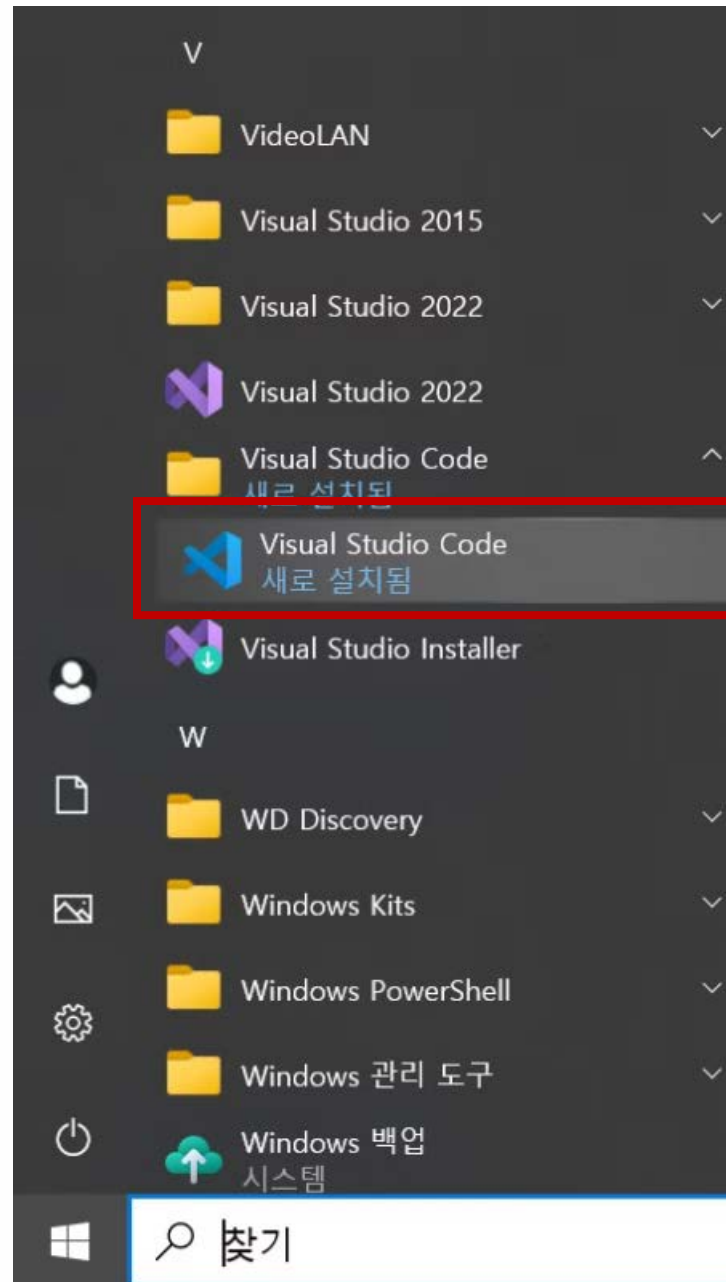
The screenshot shows a Windows Command Prompt window titled "관리자: 명령 프롬프트". The text inside the window displays the directory change to c:\movie_project01 and the successful execution of the git clone command to download the repository from GitHub.

4.Flask 영화 추천 서비스

이 서비스는 Flask 웹 프레임워크를 기반으로 하며, 머신러닝 알고리즘을 통해 정확한 추천 결과를 제공합니다. Spring Boot와의 연동을 통해 안정적이고 확장 가능한 서비스를 구현할 수 있습니다.

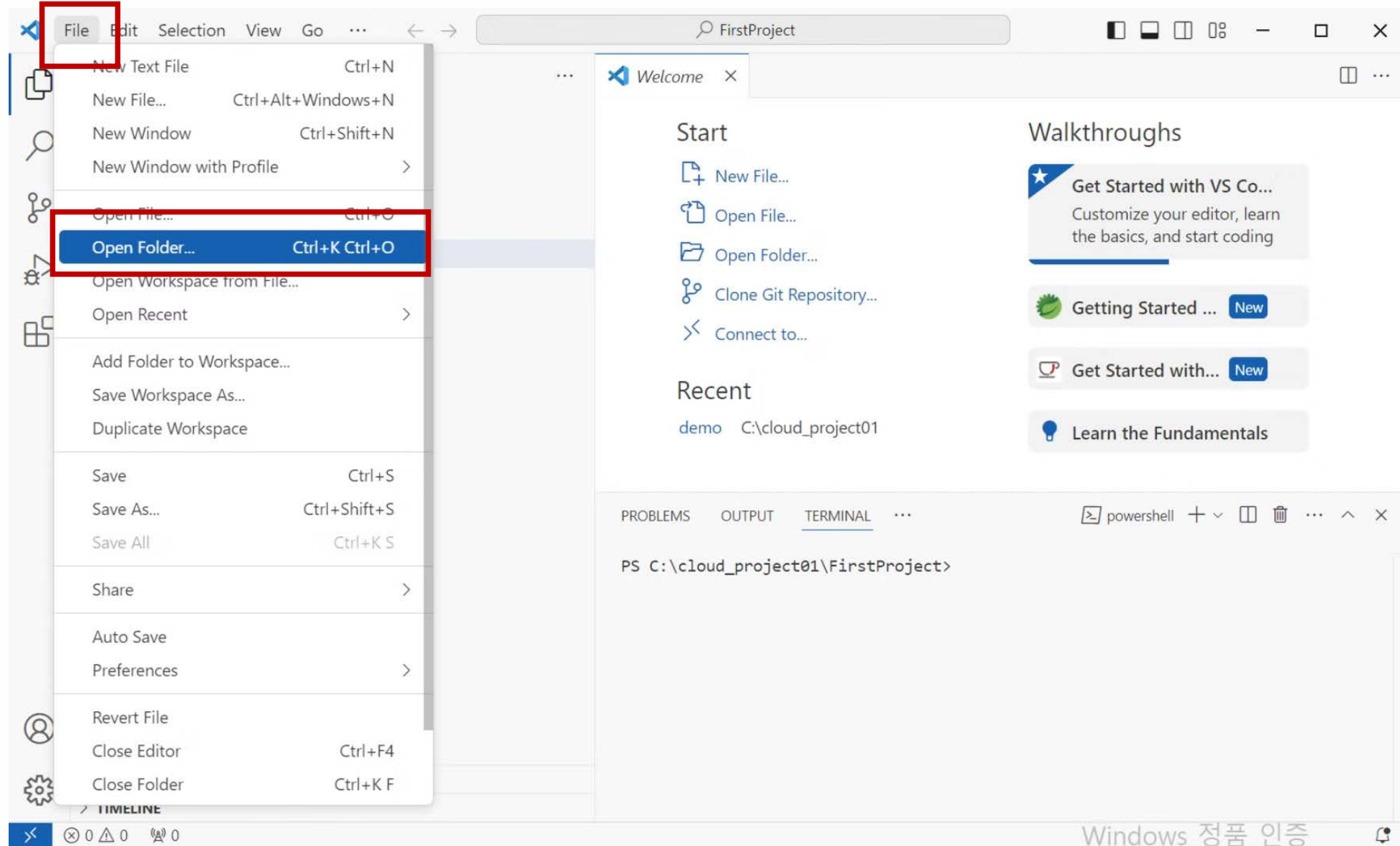
visual studio code

visual studio code 실행



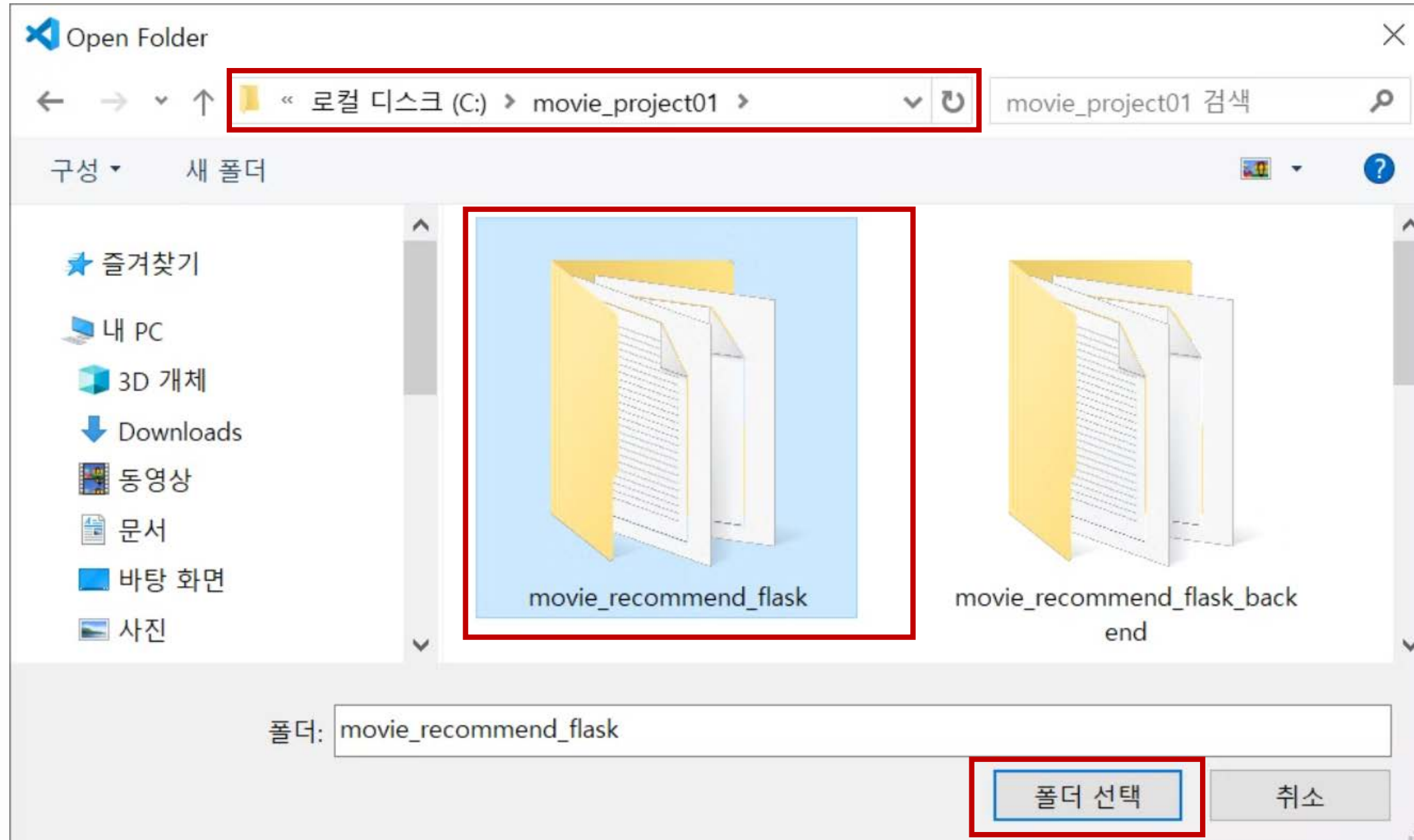
프로젝트 열기

File → Open Folder를 클릭합니다



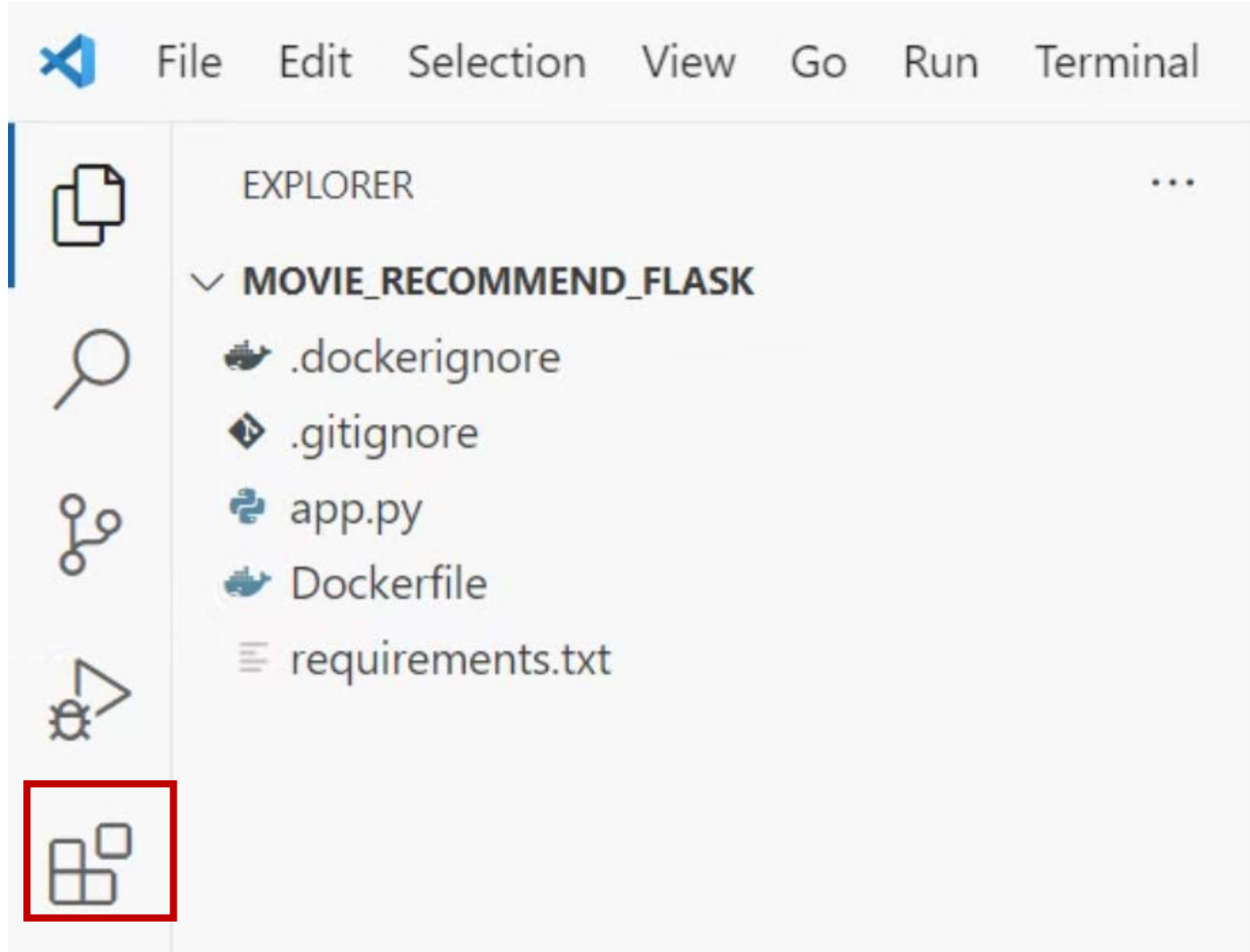
Flask 영화 추천 서비스

c:\movie_project01\movie_recommend_flask 를 선택합니다



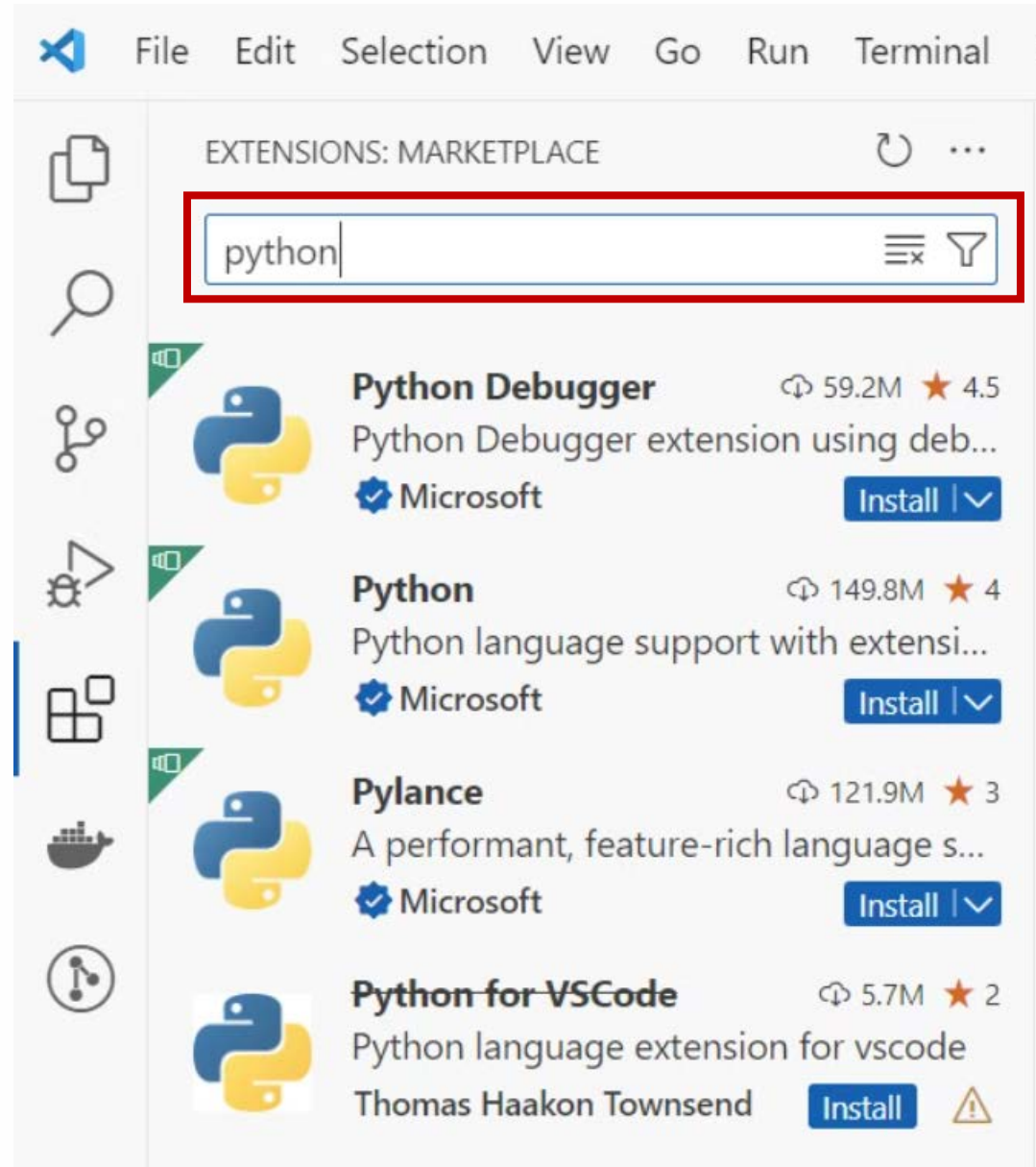
Flask 영화 추천 서비스

Extensions 탭을 선택 합니다



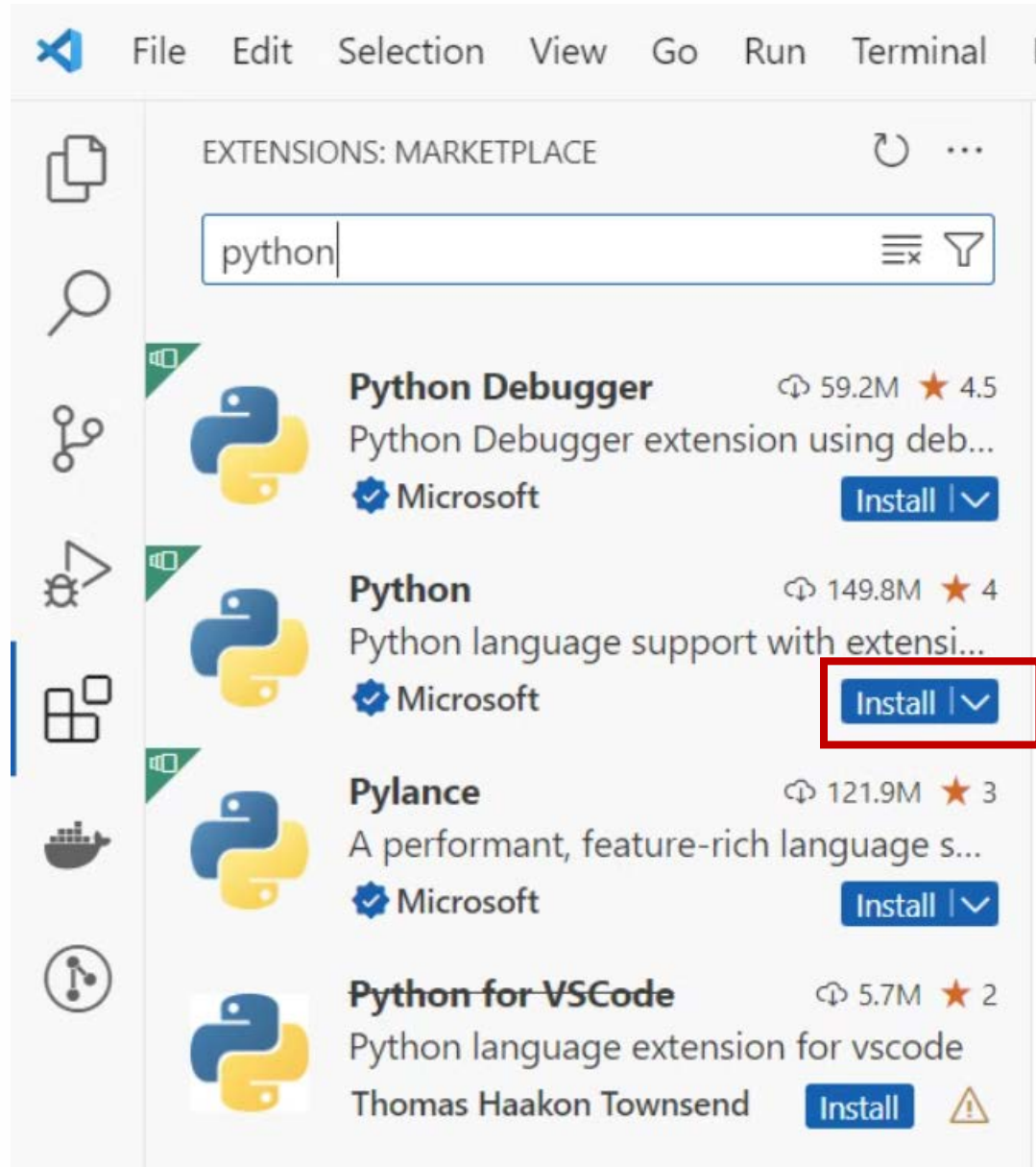
Flask 영화 추천 서비스

python을 입력 합니다



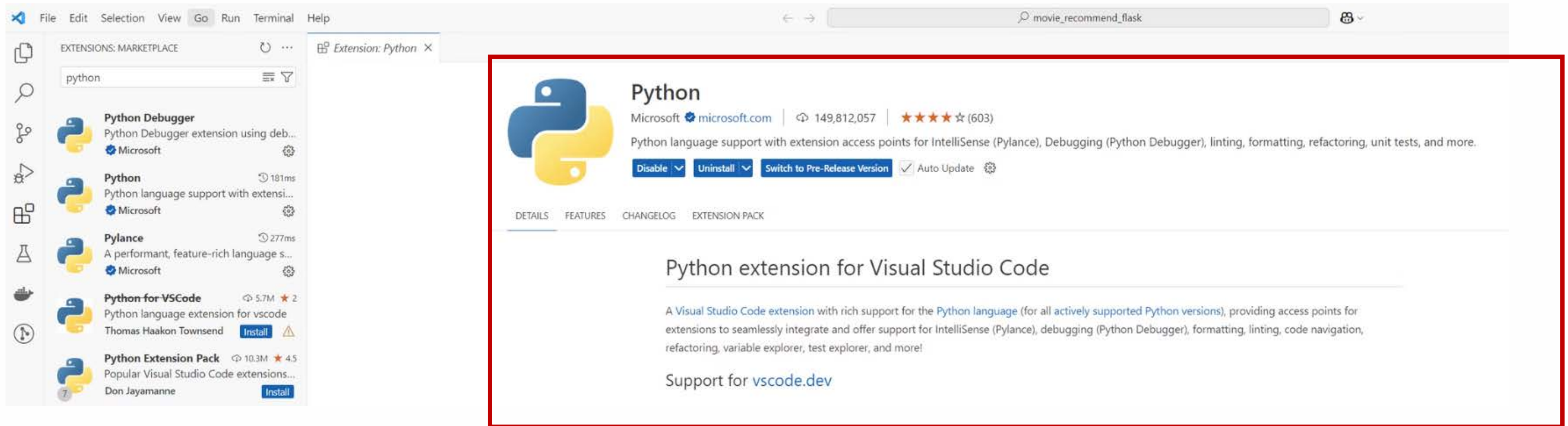
Flask 영화 추천 서비스

install 을 선택합니다



Flask 영화 추천 서비스

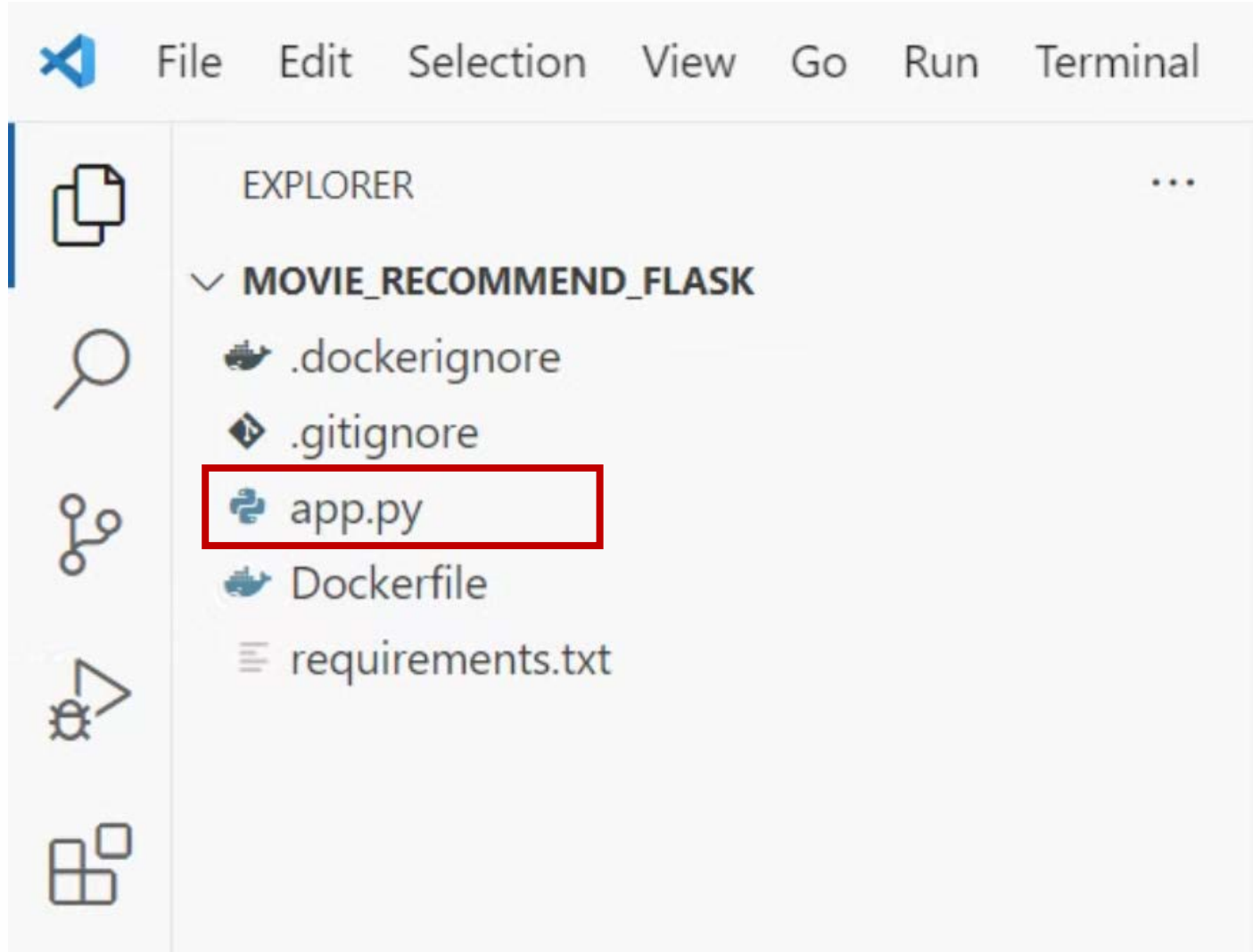
설치가 완료 될때까지 기다립니다



The screenshot displays the Visual Studio Code interface during the installation of the Python extension. On the left, the 'EXTENSIONS: MARKETPLACE' sidebar shows a search for 'python' with several results, including 'Python Debugger', 'Python', 'Pylance', 'Python-for-VSCode', and 'Python Extension Pack'. The main editor area shows the 'Python' extension page, which is highlighted with a red border. The page includes the Python logo, the extension name 'Python', the publisher 'Microsoft', and a description: 'Python language support with extension access points for IntelliSense (Pylance), Debugging (Python Debugger), linting, formatting, refactoring, unit tests, and more.' Below this, there are buttons for 'Disable', 'Uninstall', and 'Switch to Pre-Release Version', along with a checked 'Auto Update' option. The page also features tabs for 'DETAILS', 'FEATURES', 'CHANGELOG', and 'EXTENSION PACK'. The 'Python extension for Visual Studio Code' section describes the extension's capabilities, mentioning support for the Python language, IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, and more. It also mentions support for vscod.dev.

Flask 영화 추천 서비스

app.py 파일 선택



5.app.py 프로그램의 동작

app.py는 Flask 웹 애플리케이션의 핵심 파일로, 영화 추천 서비스의 전체적인 로직을 관리합니다. 이 프로그램은 사용자의 요청을 받아 처리하고, 머신러닝 모델을 통해 개인화된 영화 추천 결과를 제공합니다.

프로그램의 동작

1단계: 영화 목록을 준비합니다.

- 이 프로그램은 영화 제목과 줄거리를 **영화 도서관(데이터베이스)**에 저장해 둡니다.
- 그리고 이 도서관에서 **모든 영화 목록**을 꺼내옵니다.

2단계: 영화 줄거리를 숫자로 바꿉니다.

- 컴퓨터는 글자(문장)를 이해하지 못합니다.
- 그래서 영화 줄거리를 숫자로 바꿔서 비교할 수 있도록 준비합니다.

3단계: 줄거리를 정리합니다.

- 숫자로 바꾼 줄거리의 크기를 일정하게 맞춰 정리합니다.
- (예: 키와 몸무게를 비교할 때, 키는 cm로, 몸무게는 kg으로 맞춰야 비교가 쉬운 것처럼요.)

4단계: 비슷한 영화 찾기

- 이제 프로그램은 영화 줄거리의 **유사도(비슷한 정도)**를 계산합니다.
- 줄거리가 비슷한 영화일수록 더 가까운 친구처럼 판단합니다.

5단계: 추천 영화 10개 찾기

- 사용자가 고른 영화와 **가장 비슷한 영화 10개**를 찾아서 알려줍니다.

6.app.py 코드 구조 살펴보기

app.py는 Flask 애플리케이션의 핵심 파일입니다

app.py 각 부분 설명

(1) 도구 준비

```
from flask import Flask, request, jsonify # Flask는 웹사이트를 만들 수 있는 도구, request는 사용자 요청을 받을 때 사용
import json # 데이터를 주고받기 쉽게 변환하는 도구
import pymysql # MySQL 데이터베이스와 연결할 수 있는 도구

import pandas as pd # 데이터를 표 형태로 다룰 수 있는 도구
import numpy as np # 숫자 데이터 계산을 쉽게 해주는 도구
from sklearn.metrics.pairwise import euclidean_distances # 두 점 사이의 거리를 계산하는 도구
from sklearn.preprocessing import StandardScaler # 데이터를 표준화(정리)하는 도구
import os
```

- **Flask:** 웹사이트를 만드는 도구
- **pymysql:** 영화 도서관(데이터베이스)과 연결하는 도구
- **pandas:** 영화 정보를 표로 정리하는 도구
- **numpy:** 숫자 계산을 쉽게 해주는 도구
- **scikit-learn:** 영화끼리 얼마나 비슷한지 계산하는 도구

app.py 각 부분 설명

(2) 웹사이트 구조 설정

```
app = Flask(__name__)
```

이 부분은 프로그램이 웹사이트처럼 동작할 수 있도록 **기본 틀을 만드는 코드**입니다.
(웹사이트 주소에 입력하면 영화 추천 결과를 보여줄 준비를 합니다.)

app.py 각 부분 설명

(3) 데이터베이스 연결 정보 설정

```
DATABASE_URL = os.getenv('DATABASE_URL', 'default_url')
DATABASE_USER = os.getenv('DATABASE_USER', 'default_user')
DATABASE_PASSWORD = os.getenv('DATABASE_PASSWORD', 'default_password')
```

설명:

- 이 부분은 영화 정보를 보관한 도서관(데이터베이스) 주소와 **열쇠(비밀번호)**를 설정합니다.
- 쉽게 말하면, **데이터 도서관에 들어갈 준비**를 하는 단계입니다.

app.py 각 부분 설명

(4) 추천 기능 실행 준비

```
#post 방식으로 ai_recomend URL일때 실행
@app.route('/ai/ai_recommend', methods=["POST"])
def movie_recommend():
```

- 이 코드는 사용자가 영화 추천을 요청했을 때 실행됩니다.
- 웹사이트에 '영화 추천해줘!'라는 요청이 오면 이 부분이 동작합니다.

app.py 각 부분 설명

(5)도서관(데이터베이스) 연결하기

```
# 데이터베이스와 연결 설정
# 데이터를 저장한 MySQL 데이터베이스에 연결하기 위한 정보 입력
db = pymysql.connect(
    host=DATABASE_URL, # 데이터베이스 주소 (AWS RDS)
    port=3306, # MySQL이 사용하는 기본 포트 번호
    user=DATABASE_USER, # 데이터베이스 사용자 이름
    passwd=DATABASE_PASSWORD, # 데이터베이스 비밀번호
    db='movie_db', # 사용할 데이터베이스 이름
    charset='utf8' # 데이터가 깨지지 않도록 UTF-8 인코딩 설정
)
```

- 영화 정보가 저장된 도서관에 **입장**합니다.
- 예를 들어, 데이터베이스라는 도서관에서 영화 목록을 꺼낼 수 있는 열쇠를 사용합니다.

app.py 각 부분 설명

(6)영화 목록 꺼내오기

```
# MySQL에서 모든 영화 데이터를 가져오는 SQL 명령어  
sql = "select * from movie_tbl;"
```

```
# SQL 명령어 실행 결과를 표 형태의 데이터로 변환해서 movie_df에 저장  
movie_df = pd.read_sql(sql, db)
```

이 부분에서는 영화 도서관에서 **영화 목록 전체**를 꺼내서 표로 정리합니다.

쉽게 말하면, 도서관에서 책 목록을 가져오는 것과 같습니다.

app.py 각 부분 설명

(7)줄거리 데이터를 숫자로 변환

```
# 각 영화의 줄거리 데이터를 숫자로 변환해서 새로운 칸에 저장
movie_df.loc[:, "synopsis_vector_numpy"] = movie_df.loc[:, "synopsis_vector"].apply(
    lambda x: np.fromstring(x, dtype="float32") # 줄거리 데이터를 숫자 배열로 바꾸는 작업
)
```

영화 줄거리(문장)를 컴퓨터가 이해할 수 있도록 숫자로 변환합니다.

이렇게 하면 영화 줄거리를 수치로 비교할 수 있습니다.

app.py 각 부분 설명

(8) 줄거리 정리(표준화)

```
# 데이터를 정리(표준화)하기 위한 도구를 준비
# 표준화: 데이터의 평균을 0으로, 데이터의 크기를 1로 맞추는 작업
scaler = StandardScaler()

# 데이터를 표준화하기 위한 계산 작업 (평균과 표준편차 계산)
scaler.fit(np.array(movie_df["synopsis_vector_numpy"].tolist()))

# 데이터를 표준화한 결과를 새로운 칸에 저장
movie_df["synopsis_vector_numpy_scale"] = scaler.transform(
    np.array(movie_df["synopsis_vector_numpy"].tolist())
).tolist()
```

설명:

숫자 데이터의 크기를 일정하게 맞춥니다.

비교가 더 정확해지도록 데이터를 정리합니다.

app.py 각 부분 설명

(9)영화 간 비슷한 정도 계산

```
# 영화들 간의 유사도를 계산하기 위한 거리(유클리드 거리)를 구함
# 유클리드 거리: 두 영화 줄거리 간의 차이를 계산
sim_score = euclidean_distances(
    movie_df["synopsis_vector_numpy_scale"].tolist(),
    movie_df["synopsis_vector_numpy_scale"].tolist()
)
```

영화들의 줄거리 데이터를 서로 비교하고, **얼마나 비슷한지 거리(차이)**를 계산합니다.

거리가 가까울수록 더 비슷한 영화입니다.

app.py 각 부분 설명

(10)비슷한 영화 10개 찾기

```
# 입력한 영화와 가장 비슷한 영화 10개를 찾음 (자신 제외)  
result = sim_df[title].sort_values()[1:11]
```

사용자가 선택한 영화와 가장 비슷한 영화 10개를 찾아서 리스트로 반환합니다.

예를 들어, '타이타닉'을 입력하면 비슷한 영화 10개가 추천됩니다.

app.py 각 부분 설명

이 프로그램이 실제로 하는 일

1. 영화 정보 가져오기 → 도서관(데이터베이스)에서 영화 정보를 불러옵니다.
2. 줄거리 숫자로 바꾸기 → 줄거리를 숫자로 변환하여 비교합니다.
3. 줄거리 정리하기 → 데이터를 정리해서 비교가 더 쉽도록 만듭니다.
4. 유사 영화 찾기 → 입력한 영화와 비슷한 영화 10개를 찾습니다.
5. 추천 영화 보여주기 → 사용자가 결과를 확인할 수 있도록 보여줍니다.

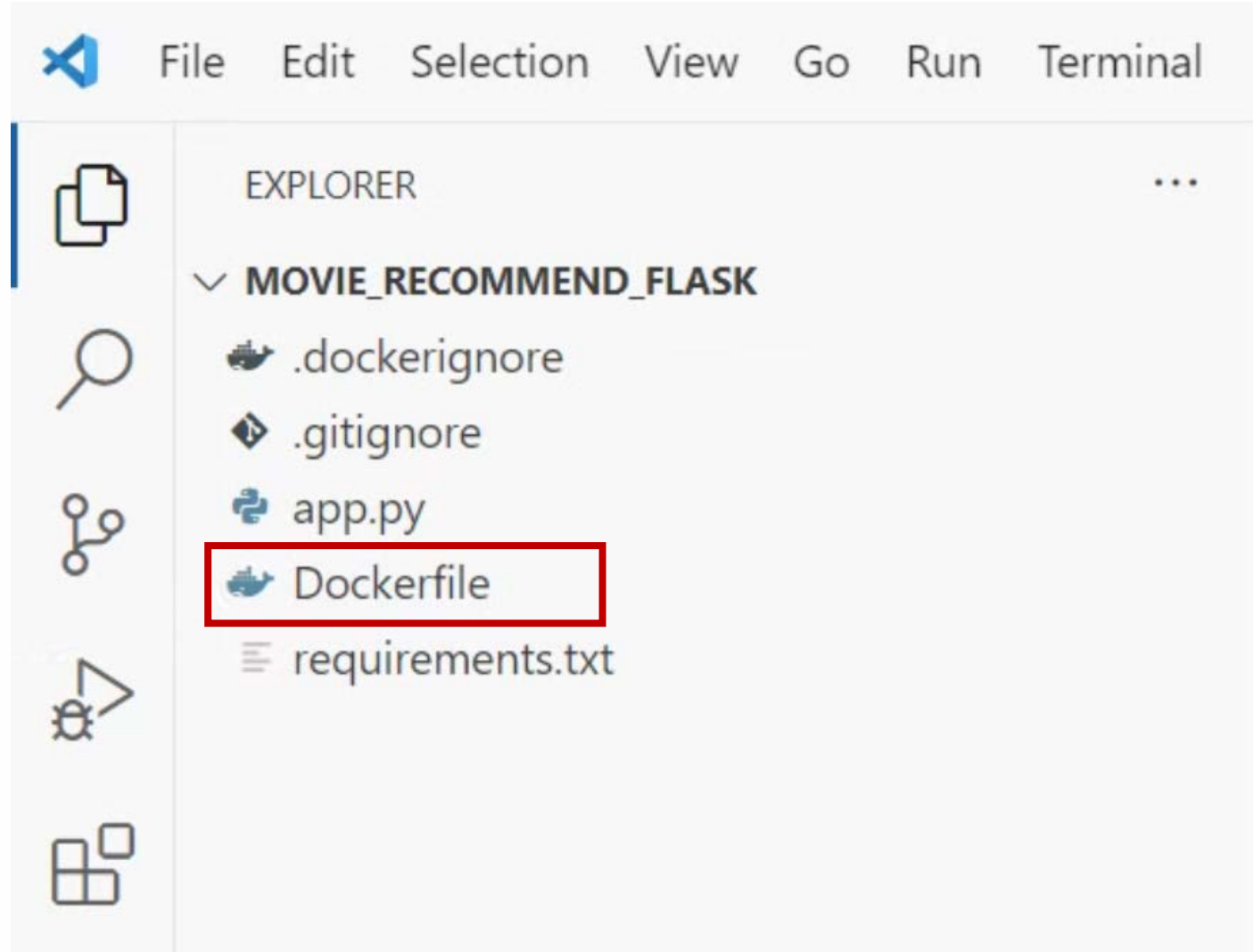
7.Dockerfile

Dockerfile은 도커 컨테이너 이미지를 생성하기 위한 설정 파일입니다. 마치 요리 레시피처럼 컨테이너를 만드는 모든 명령어들을 순서대로 담고 있습니다.

이 파일을 통해 애플리케이션이 실행되는데 필요한 모든 환경을 코드로 정의하고, 어떤 환경에서도 동일하게 실행될 수 있도록 보장합니다.

Flask 영화 추천 서비스

Dockerfile 선택



Dockerfile

이 코드는 도커(Docker)라는 프로그램에서 사용하는 **설정서(레시피)**입니다.

도커는 **가상의 작은 컴퓨터**를 만드는 도구입니다.

이 설정서는 도커에게 "**이 가상 컴퓨터를 이렇게 만들어 주세요**"라고 알려주는 역할을 합니다.

쉽게 말해, 이 코드는 **요리 레시피**처럼 도커가 프로그램을 실행할 수 있는 환경을 준비하는 **설명서**입니다.

Dockerfile

dockerfile

 Copy code

```
FROM python:3.10-slim
```

- "Python(파이썬) 3.10 버전이 설치된 가상 컴퓨터를 만들어 주세요."라는 의미입니다.
- **파이썬(Python)**은 프로그램을 만들 때 사용하는 언어입니다.
- 뒤에 'slim'은 **가벼운 버전(필요한 기능만 포함)**을 의미합니다.

비유:

파이썬이 이미 설치된 **미니멀한 노트북**을 준비하는 것과 비슷합니다.

Dockerfile

```
dockerfile
```

 Copy code

```
WORKDIR /app
```

- 가상 컴퓨터 안에서 **작업할 폴더(서랍)**를 만듭니다.
- 폴더 이름은 **/app**입니다.
- 앞으로 이 폴더에서 작업이 이루어질 거예요.

비유:

새로 산 노트북에 '**작업 폴더**'를 만들어 놓고, 앞으로 모든 작업을 이 폴더에서 하겠다고 선언하는 것과 같습니다.

Dockerfile

```
dockerfile
```

 Copy code

```
COPY . .
```

- 내 컴퓨터에 있는 **모든 파일**을 가상 컴퓨터의 **/app** 폴더로 복사합니다.

비유:

노트북에 있는 파일들을 **USB**에 복사해서 가상 컴퓨터에 옮겨 담는 것과 같습니다.

Dockerfile

dockerfile

 Copy code

```
RUN pip install -r requirements.txt
```

- 프로그램을 실행하기 위해 **필요한 도구들**을 설치합니다.
- 설치할 도구 목록은 **requirements.txt**라는 파일에 적혀 있습니다.

비유:

새 노트북에 문서 작성 프로그램, 인터넷 브라우저, 메신저 등을 **필수 프로그램**으로 설치하는 것과 비슷합니다.

Dockerfile

dockerfile

 Copy code

```
ENV FLASK_ENV=production
```

- 프로그램이 **실제 서비스(운영 환경)**로 작동하도록 설정합니다.
- **FLASK_ENV=production**은 성능을 최적화하고 보안을 강화하는 설정입니다.

비유:

프로그램을 **개발용(연습 모드)**에서 **서비스용(고객에게 제공할 준비 완료)**으로 전환하는 스위치를 켜는 것과 같습니다.

Dockerfile

```
dockerfile
```

 Copy code

```
CMD ["python", "app.py"]
```

- 프로그램을 실행합니다.
- 여기서는 **app.py**라는 파일을 실행합니다.

비유:

노트북에서 앱 아이콘을 클릭해서 프로그램을 실행하는 것과 같습니다.

전체 과정 다시 보기

1. 기본 컴퓨터 준비:

- 파이썬이 설치된 가상 컴퓨터를 준비합니다.

2. 작업 폴더 만들기:

- 가상 컴퓨터 안에 **/app** 폴더를 만듭니다.

3. 파일 복사:

- 내 컴퓨터에서 프로그램 파일들을 가상 컴퓨터로 복사합니다.

4. 필수 도구 설치:

- 프로그램 실행에 필요한 도구들을 설치합니다.

5. 운영 모드 설정:

- 프로그램을 실제 서비스 모드로 실행할 준비를 합니다.

6. 프로그램 실행:

- 프로그램을 실제로 실행합니다.

이 코드가 실제로 하는 일 예시

이 코드를 실행하면 아래와 같은 일이 순서대로 일어납니다.

1. 기본 환경 준비:

- 도커가 파이썬 3.10이 설치된 가상 컴퓨터를 만듭니다.

2. 폴더 만들기:

- 가상 컴퓨터 안에 **작업 폴더(/app)**를 만듭니다.

3. 파일 복사:

- 내 컴퓨터에 있는 파일들을 가상 컴퓨터의 **/app 폴더**에 복사합니다.

4. 필수 도구 설치:

- 프로그램 실행에 필요한 도구들을 자동으로 설치합니다.

5. 서비스 모드 설정:

- 프로그램을 **운영 환경(실제 서비스 준비 완료 상태)**으로 설정합니다.

6. 프로그램 실행:

- 프로그램을 실제로 실행하여 **영화 추천 기능**을 제공합니다.

비유로 마무리하기

이 코드는 마치 **음식 배달 서비스**를 시작하는 과정과 비슷합니다.

1. 새 가게(가상 컴퓨터) 준비:

- 배달 음식 가게를 위한 주방을 설치합니다.

2. 작업 공간(폴더) 준비:

- 주방 안에 요리 도구와 재료를 놓을 공간을 만듭니다.

3. 재료(파일) 옮기기:

- 필요한 재료를 창고에서 주방으로 옮깁니다.

4. 필수 도구 설치:

- 조리 도구와 접시, 포장 용기를 준비합니다.

5. 가게 오픈 설정:

- 손님을 받을 준비가 되었다는 표시를 합니다.

6. 주문 받기 시작:

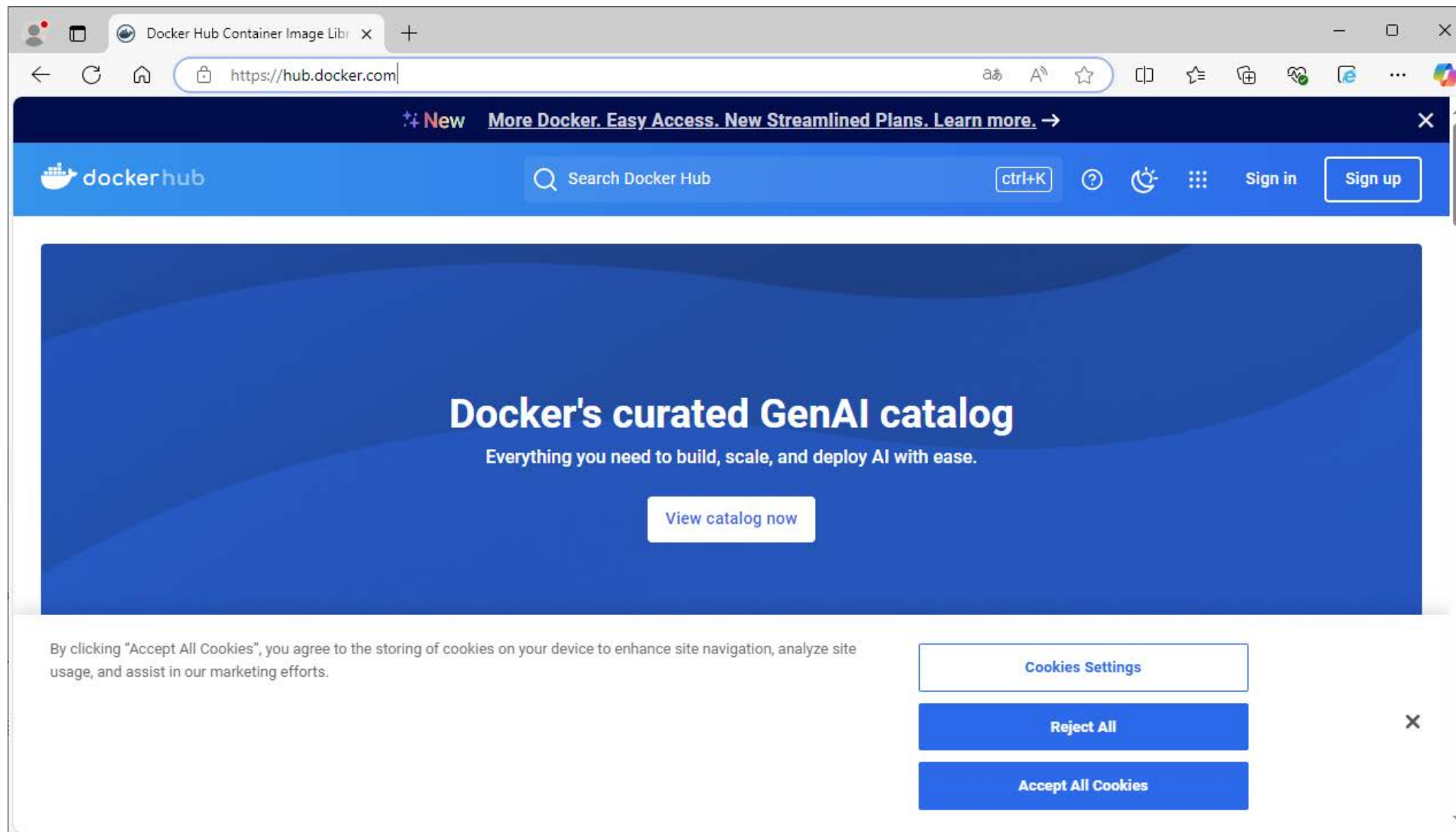
- 첫 번째 주문을 받고 요리를 시작합니다.

8 Docker Hub 회원명 확인

Docker Hub 회원명을 확인 합니다

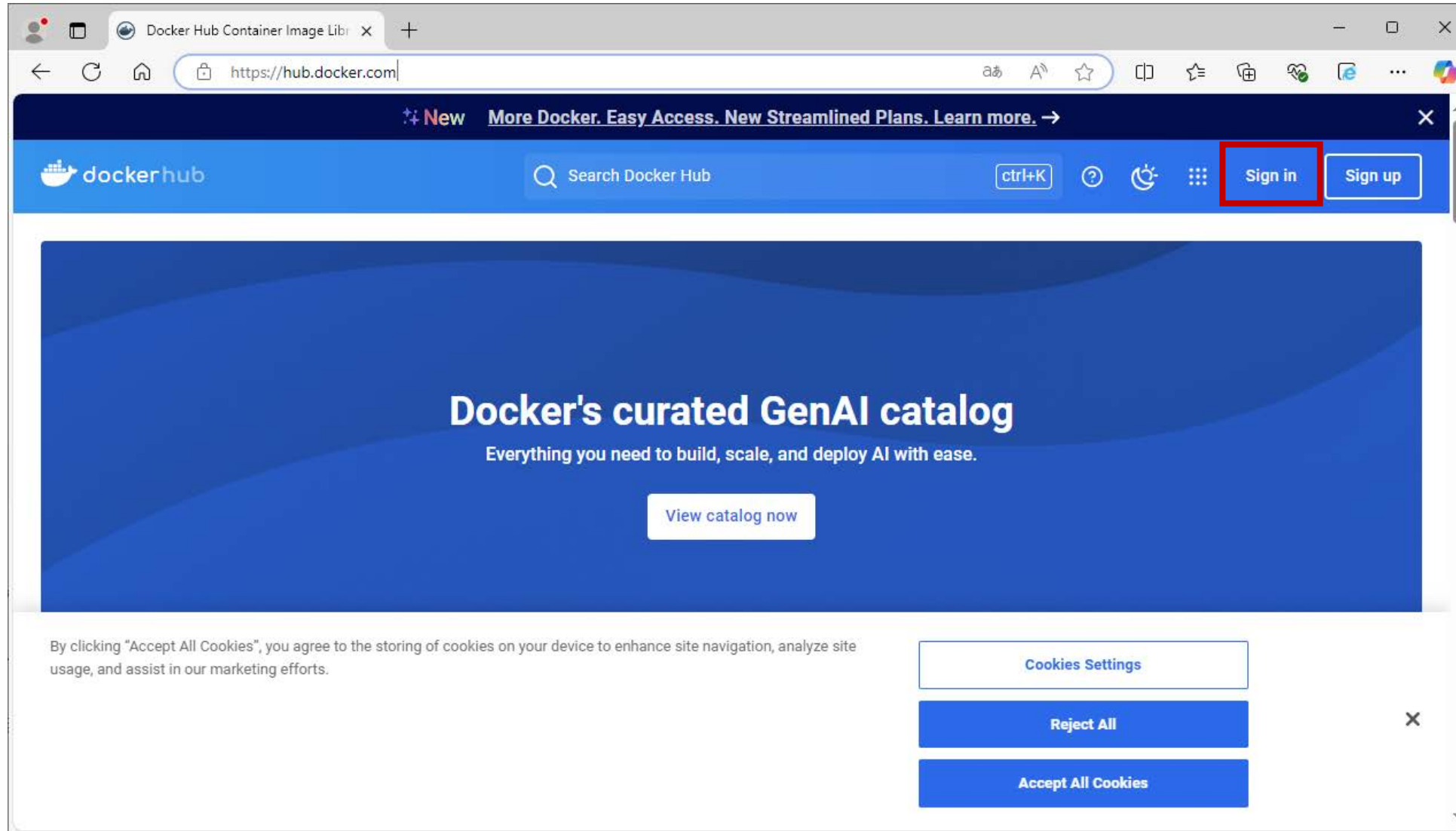
Docker Hub 회원명 확인

<https://hub.docker.com/> 접속 합니다



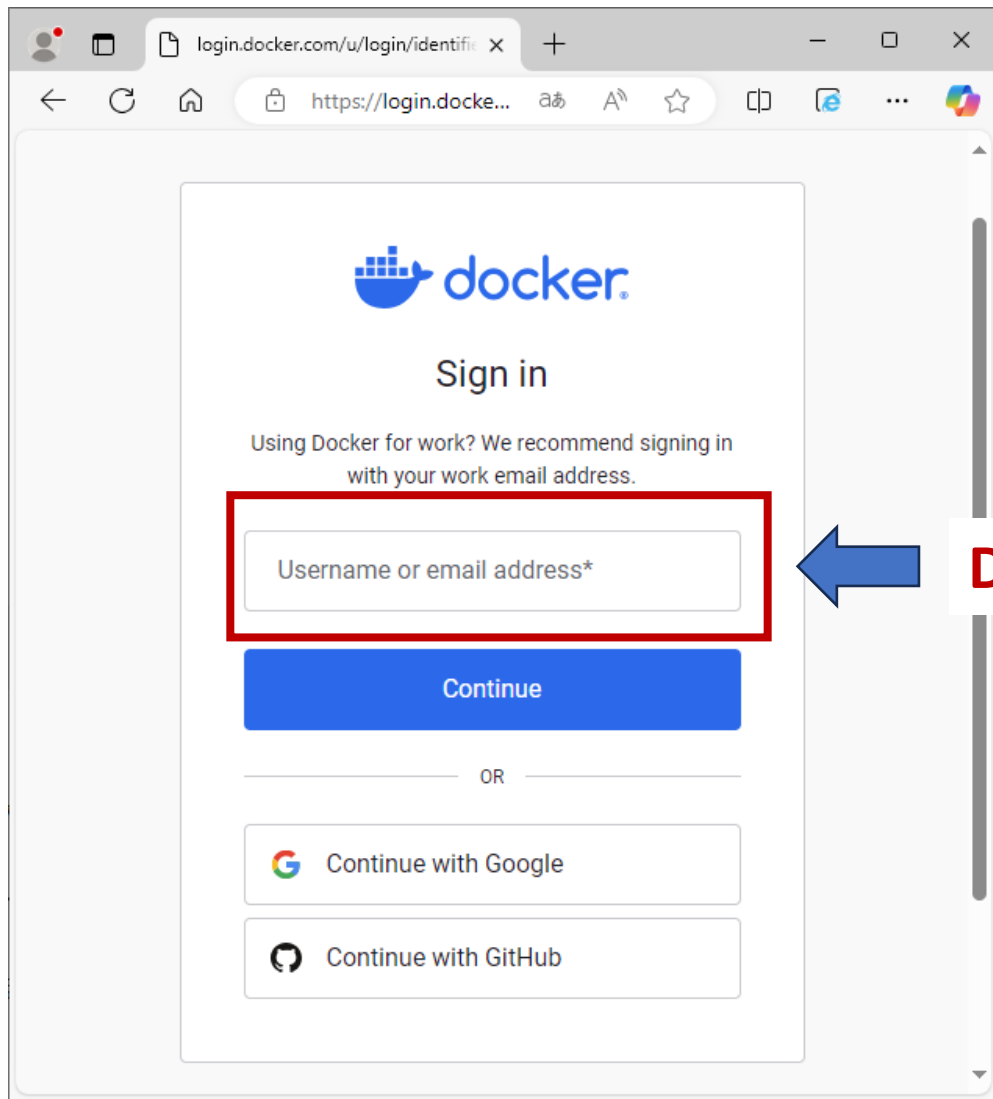
Docker Hub 회원명 확인

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

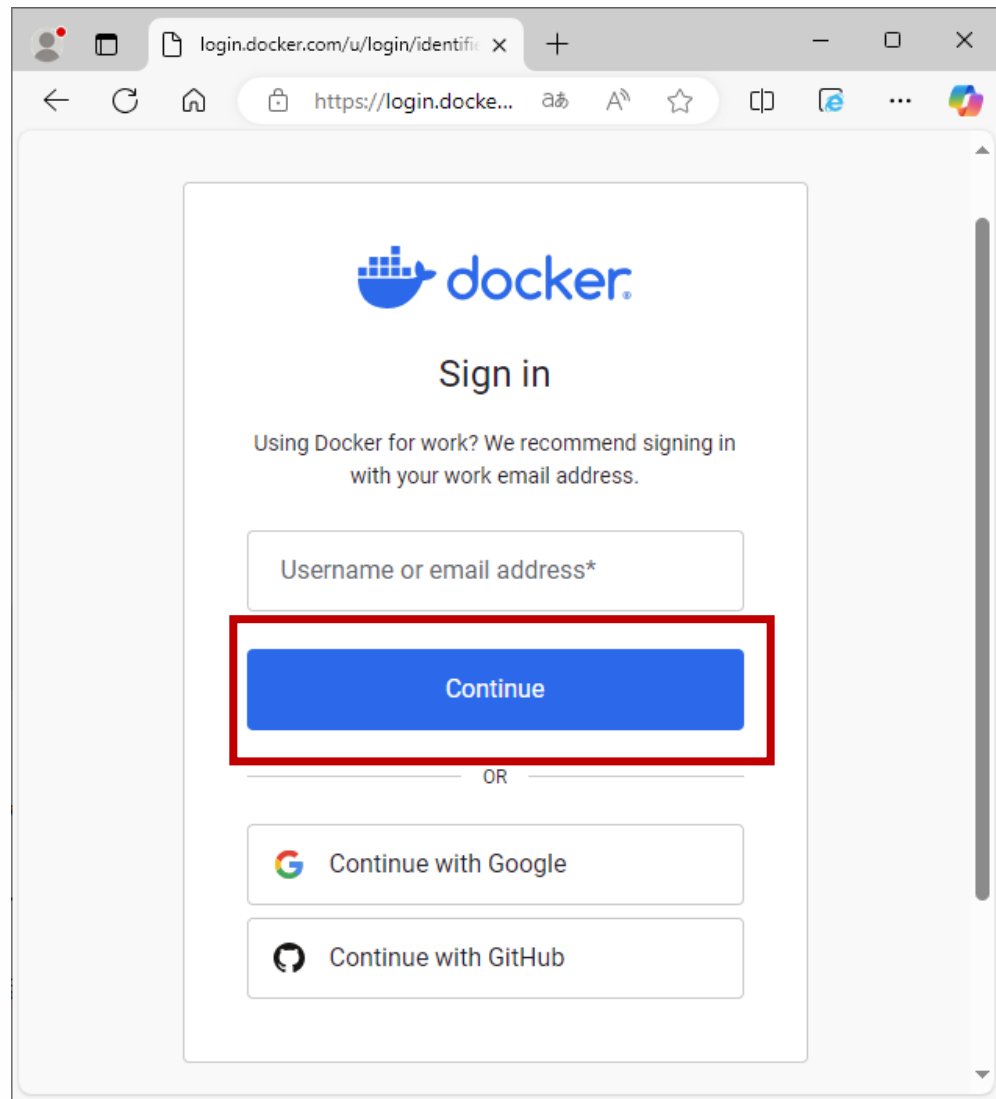
 Continue with Google

 Continue with GitHub

 Docker Hub에 가입한 이메일을 입력합니다

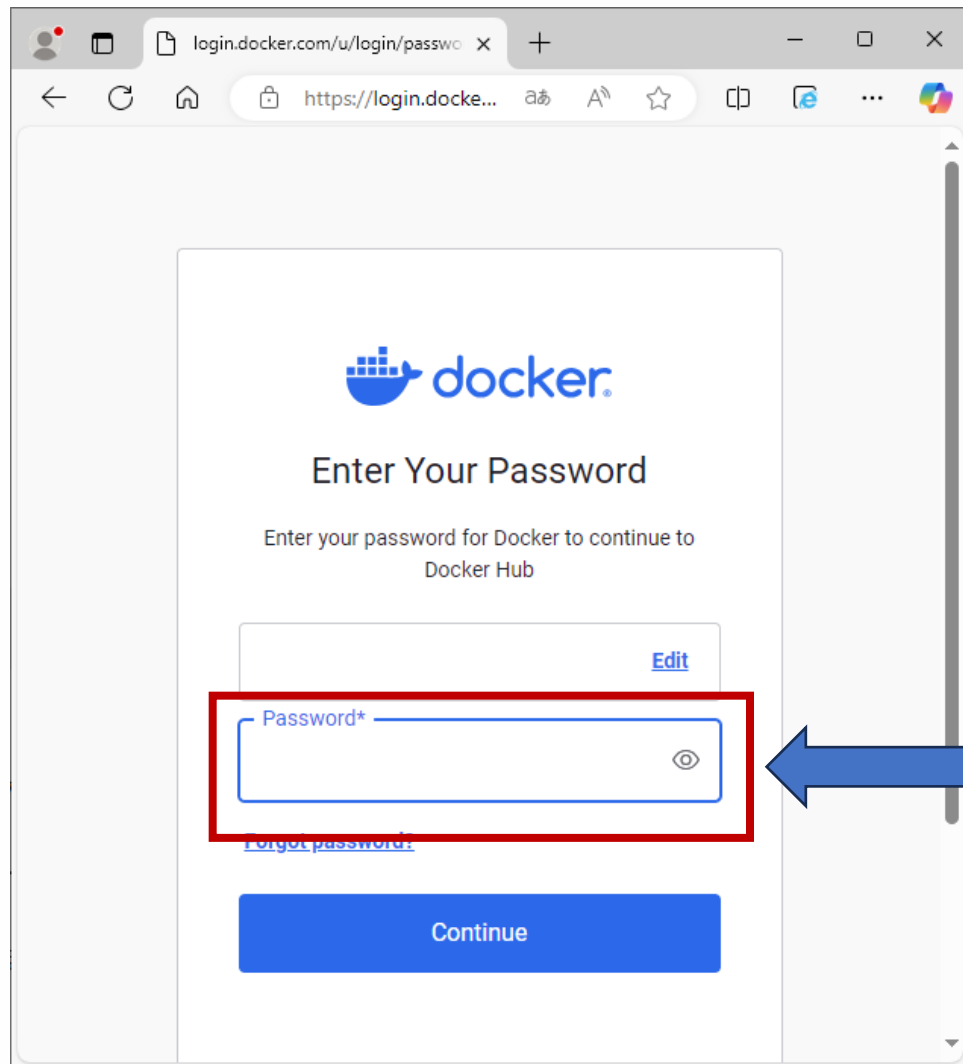
Docker Hub 회원명 확인

Continue를 클릭 합니다



Docker Hub 회원명 확인

Docker Hub에 가입한 비밀번호를 입력 합니다

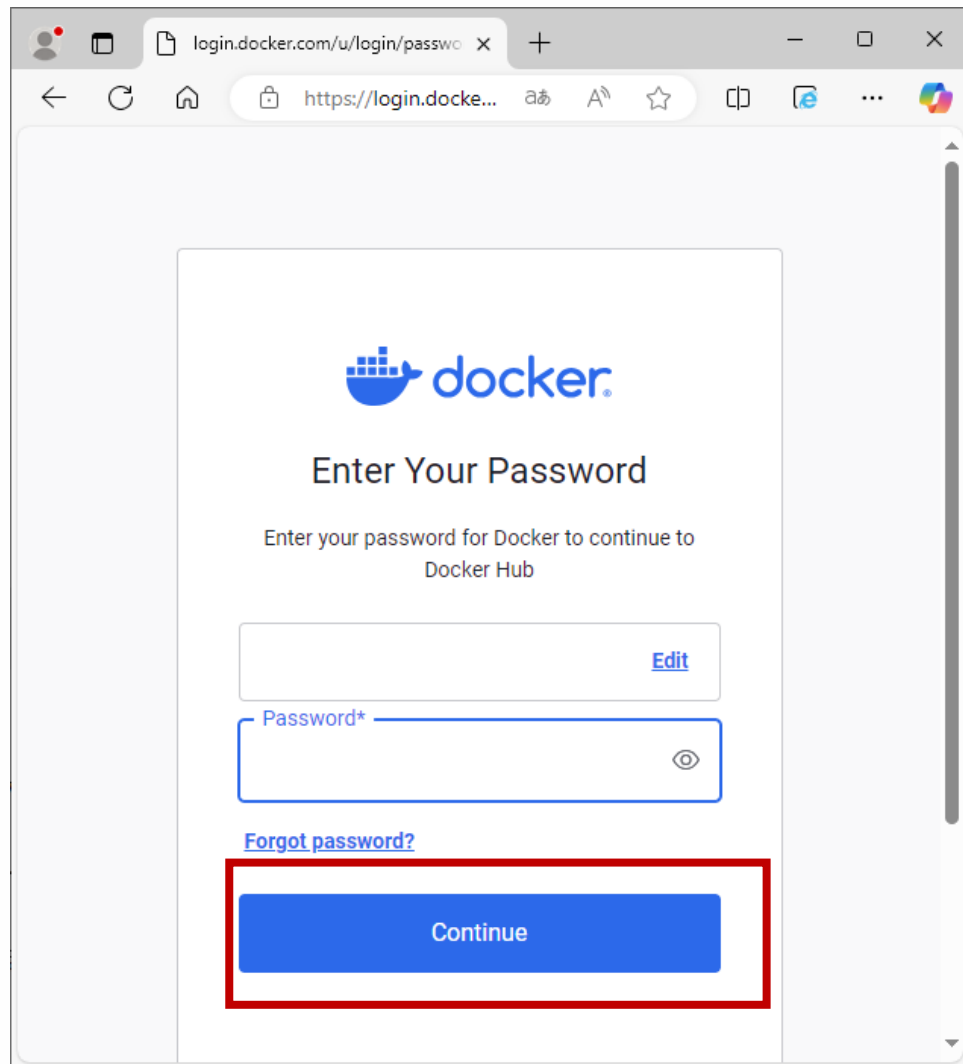


The screenshot shows the Docker Hub login page in a web browser. The page has the Docker logo at the top, followed by the text "Enter Your Password" and "Enter your password for Docker to continue to Docker Hub". Below this is a password input field labeled "Password*" with a red rectangular box around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

[Edit](#)

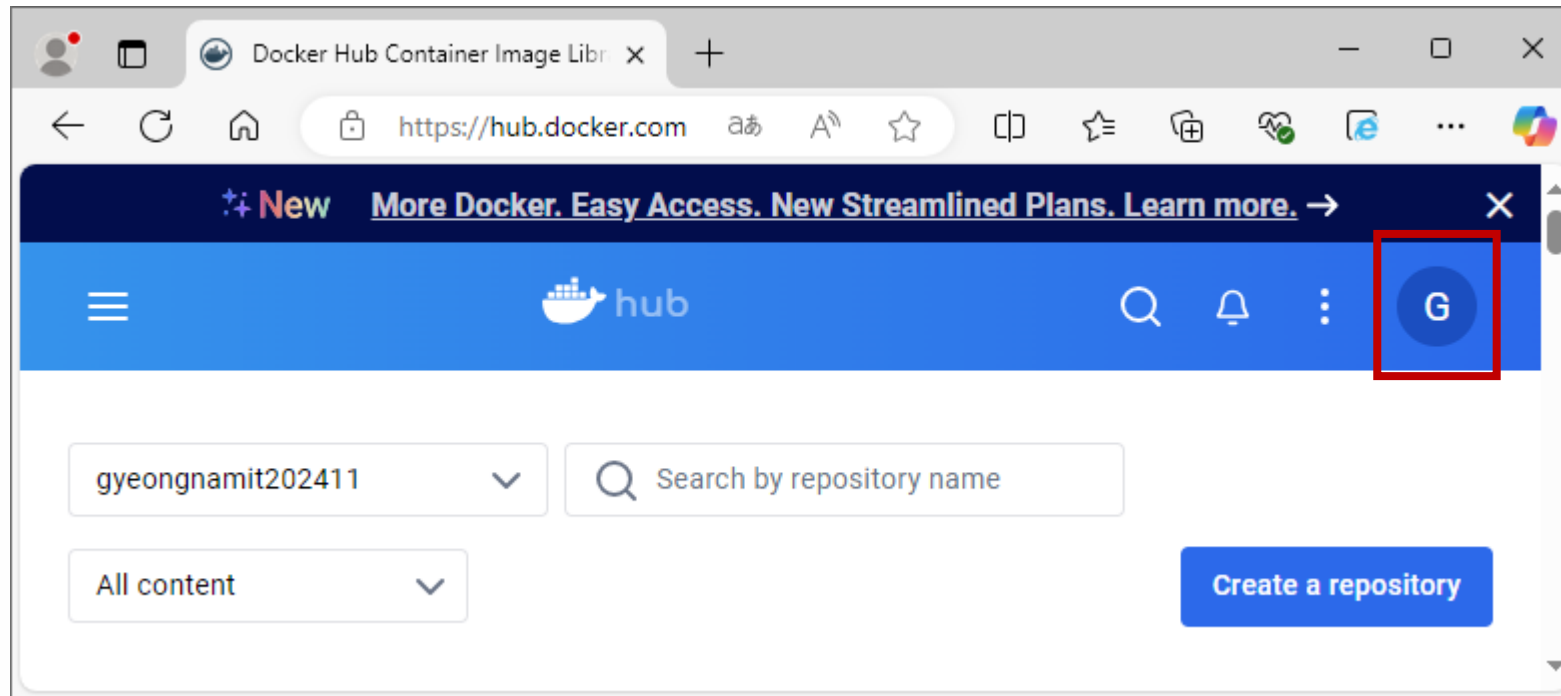
Password*

[Forgot password?](#)

Continue

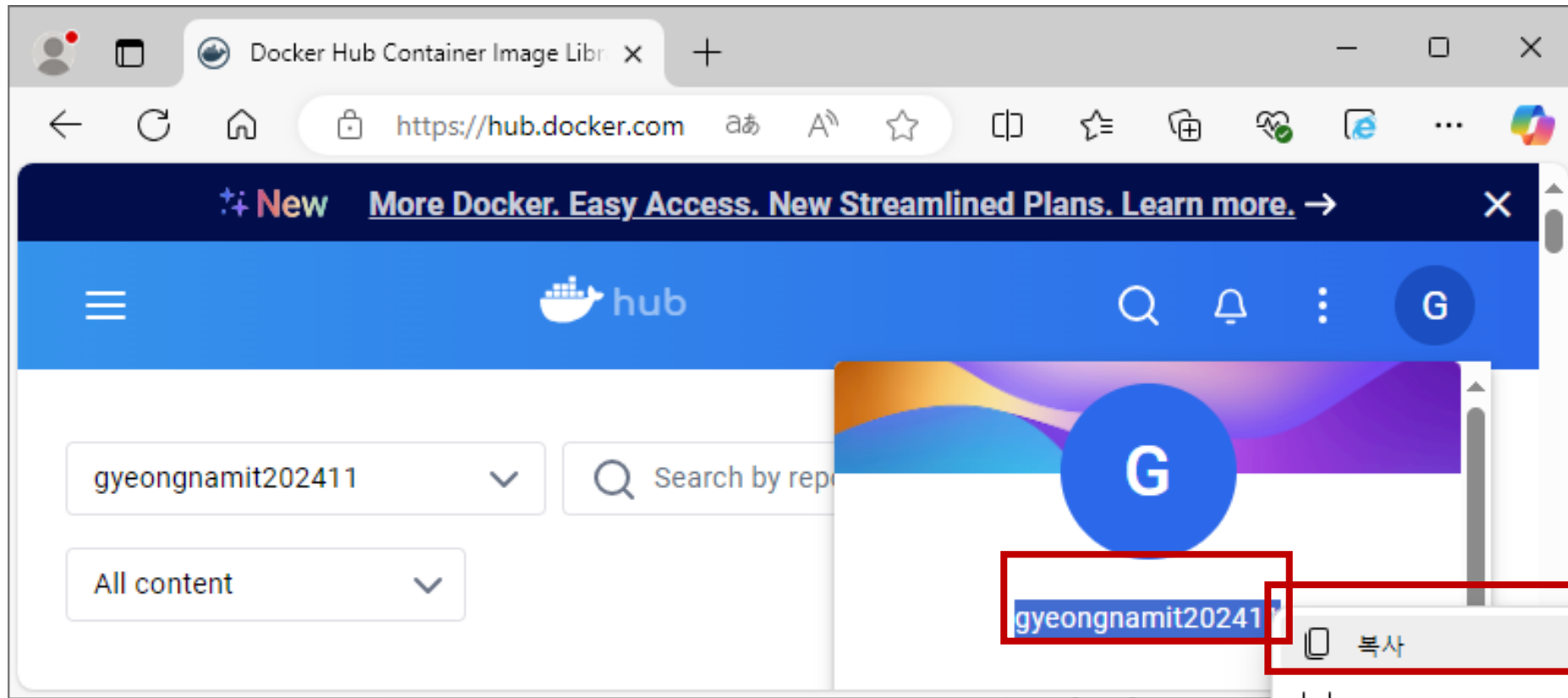
Docker Hub 회원명 확인

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

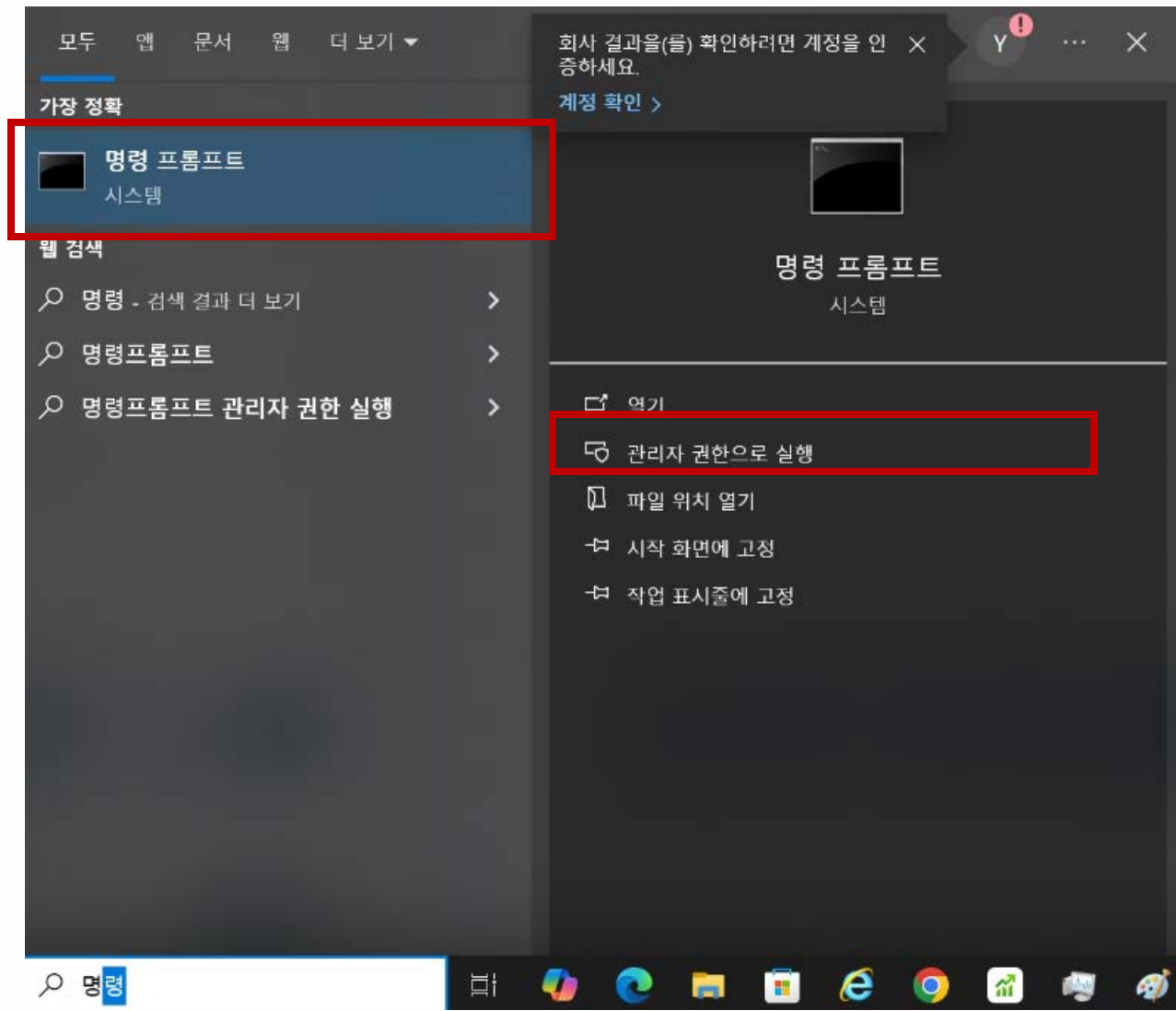


9.도커 이미지 생성

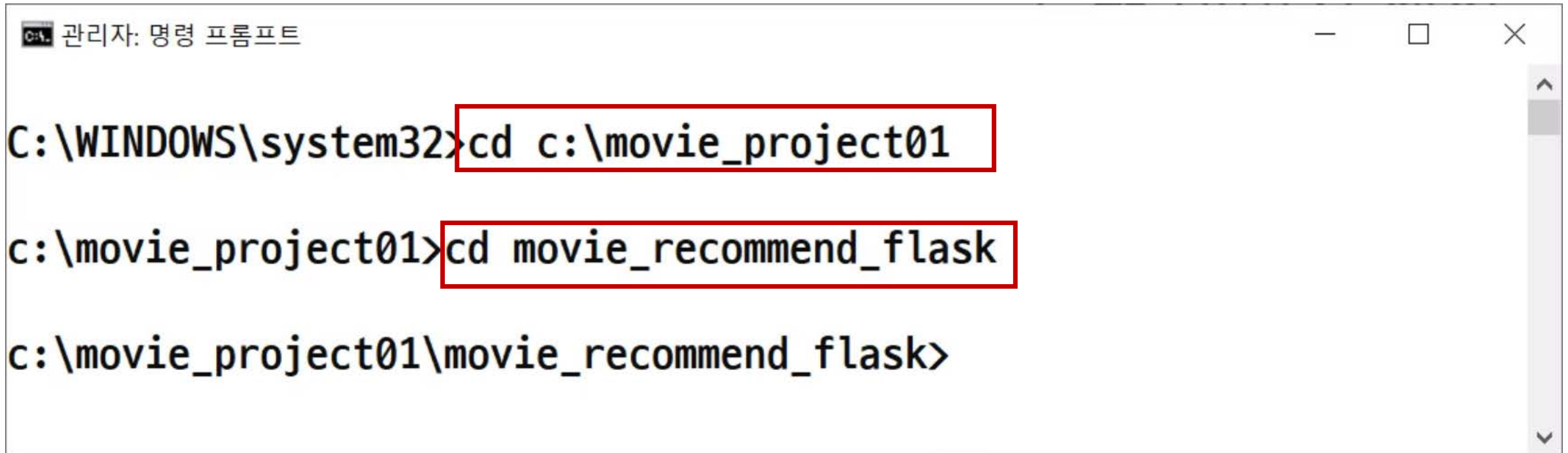
도커 이미지는 애플리케이션 실행에 필요한 모든 것을 포함하는 패키지입니다. 소스 코드, 라이브러리, 환경 변수 등이 모두 포함됩니다.

도커 이미지 생성

명령 프롬프트를 관리자 권한으로 실행 합니다



도커 이미지 생성




A screenshot of a Windows Command Prompt window titled "관리자: 명령 프롬프트" (Administrator: Command Prompt). The window shows three lines of commands being entered. The first line is "C:\WINDOWS\system32>cd c:\movie_project01", with the command "cd c:\movie_project01" highlighted by a red rectangular box. The second line is "c:\movie_project01>cd movie_recommend_flask", with the command "cd movie_recommend_flask" highlighted by a red rectangular box. The third line is "c:\movie_project01\movie_recommend_flask>". The window has standard Windows window controls (minimize, maximize, close) in the top right corner and a vertical scrollbar on the right side.

```
C:\WINDOWS\system32>cd c:\movie_project01  
c:\movie_project01>cd movie_recommend_flask  
c:\movie_project01\movie_recommend_flask>
```

도커 이미지 만들기

Docker Hub에 로그인 합니다.

bash

 Copy code

```
docker login -u <사용자 이름>
```

DockerHub에 가입한 사용자 명을 입력합니다.

```
C:\> 관리자: 명령 프롬프트

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

c:\movie_project01\movie_recommend_flask>
```

**gyeongnam
it202411 대
신에 57페
이지에서
확인한
Docker Hub
사용자 명
을 입력합
니다**

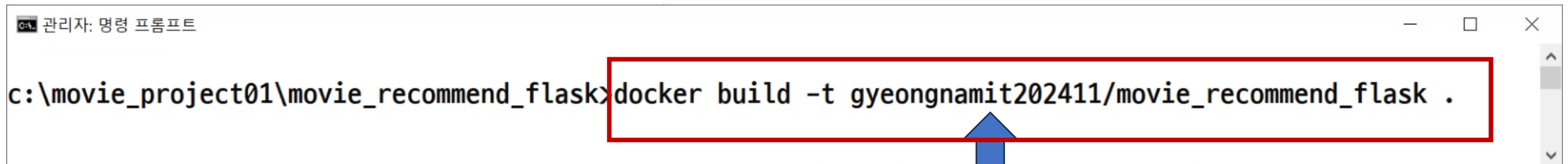
**Docker Hub에
가입할때 입력한 비밀번호 입력**

도커 이미지 생성

`docker build -t 도커허브사용자명/movie_recommend_flask .`

57페이지에서 확인한 도커허브사용자명이 gyeongnamit202411 이라면 아래와 같이 입력 합니다

`docker build -t gyeongnamit202411/movie_recommend_flask .`

A screenshot of a Windows Command Prompt window. The title bar reads '관리자: 명령 프롬프트'. The command prompt shows the current directory as 'c:\movie_project01\movie_recommend_flask' and the command being entered is 'docker build -t gyeongnamit202411/movie_recommend_flask .'. The command and its arguments are enclosed in a red rectangular box. A blue arrow points upwards from the text below towards the username 'gyeongnamit202411' in the command.

```
C:\movie_project01\movie_recommend_flask>docker build -t gyeongnamit202411/movie_recommend_flask .
```

gyeongnamit202411 대신에 57페이지에서 확인한 Docker Hub 사용자 이름을 입력합니다

도커 이미지 생성

관리자: 명령 프롬프트

```
c:\movie_project01\movie_recommend_flask>docker build -t gyeongnamit202411/movie_recommend_flask .  
[+] Building 6.3s (10/10) FINISHED  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 173B  
=> [internal] load metadata for docker.io/library/python:3.10-slim  
=> [auth] library/python:pull token for registry-1.docker.io  
=> [internal] load .dockerignore  
=> => transferring context: 186B  
=> [1/4] FROM docker.io/library/python:3.10-slim@sha256:bdc6c5b8f725df8b009b32da65cbf46bfd24d1c86dce2e6169452c193ad660b4  
=> => resolve docker.io/library/python:3.10-slim@sha256:bdc6c5b8f725df8b009b32da65cbf46bfd24d1c86dce2e6169452c193ad660b4  
=> [internal] load build context  
=> => transferring context: 4.96kB  
=> CACHED [2/4] WORKDIR /app  
=> CACHED [3/4] COPY . .  
=> CACHED [4/4] RUN pip install -r requirements.txt  
=> exporting to image  
=> => exporting layers  
=> => exporting manifest sha256:88ad8e7ab64d04ff9b6d32b005b9927577efffaf7fa3dd1762f9c5675dab80ea  
=> => exporting config sha256:7894e32ac81ee372bb0240528a7e16bd90f6d6351a6e7135d5485bfe9506e80b  
=> => exporting attestation manifest sha256:c37843be7aa80edc5e370121fd3c771caec41774e0de2b1bbbee79d98ba375a2  
=> => exporting manifest list sha256:8719eddfb1a2bb1aea5166a80fa9054c7cea52707703dcb5f1108dce2528d70e  
=> => naming to docker.io/gyeongnamit202411/movie_recommend_flask:latest  
=> => unpacking to docker.io/gyeongnamit202411/movie_recommend_flask:latest
```

c:\movie_project01\movie_recommend_flask>

```
docker:desktop-linux  
0.0s  
0.0s  
2.3s  
0.0s  
0.0s  
0.0s  
0.0s  
0.1s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
3.7s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
0.0s  
3.6s
```

에러 없이
이미지가 생성되는지
확인

도커 이미지 생성 설명

bash

 Copy code

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

- `docker build`: 도커 이미지를 생성하는 명령어입니다.
- `-t gyeongnamit202411/movie_recommend_flask`: 이미지 이름을 도커 허브 사용자명과 함께 설정합니다.
 - `gyeongnamit202411`: 도커 허브 사용자명
 - `movie_recommend_flask`: 이미지 이름
- `.`: 현재 폴더에 있는 **Dockerfile**을 사용합니다.

실행 결과:

이미지 생성이 완료되면 새로운 이미지가 도커에 추가됩니다.


10.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_flask
```




gyeongnamit202411 대신에 57페이지에서 확인한 Docker Hub 사용자 이름을 입력합니다

 관리자: 명령 프롬프트

```
c:\movie_project01\movie_recommend_flask>docker push gyeongnamit202411/movie_recommend_flask_
```

도커 이미지 DockerHub 등록

 관리자: 명령 프롬프트


```
c:\movie_project01\movie_recommend_flask>docker push gyeongnamit202411/movie_recommend_flask
Using default tag: latest
The push refers to repository [docker.io/gyeongnamit202411/movie_recommend_flask]
a1235d039a7d: Layer already exists
f344618db07e: Layer already exists
2b262c3eaf8e: Layer already exists
dfdcdca4cfba: Already exists
e1d5bfac6f75: Layer already exists
03f25e582259: Layer already exists
e17464c8c9fb: Layer already exists
fd674058ff8f: Layer already exists
latest: digest: sha256:8719eddfb1a2bb1aea5166a80fa9054c7cea52707703dcb5f1108dce2528d70e size: 856

c:\movie_project01\movie_recommend_flask>
```

에러 없이
업로드 되는지 확인

도커 이미지 DockerHub 등록

bash

 Copy code

```
docker push gyeongnamit202411/movie_recommend_flask
```

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_flask
```

명령어 구성 살펴보기

- **docker push**

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

- **gyeongnamit202411/movie_recommend_flask**

- 업로드할 도커 이미지의 이름입니다.
- gyeongnamit202411 → 도커 허브 아이디(사용자 이름).
- movie_recommend_flask → 프로그램 이름.예:

- 'gyeongnamit202411'이라는 요리사(사용자)가 만든 'movie_recommend_flask'(요리 메뉴)를 인터넷에 올립니다.

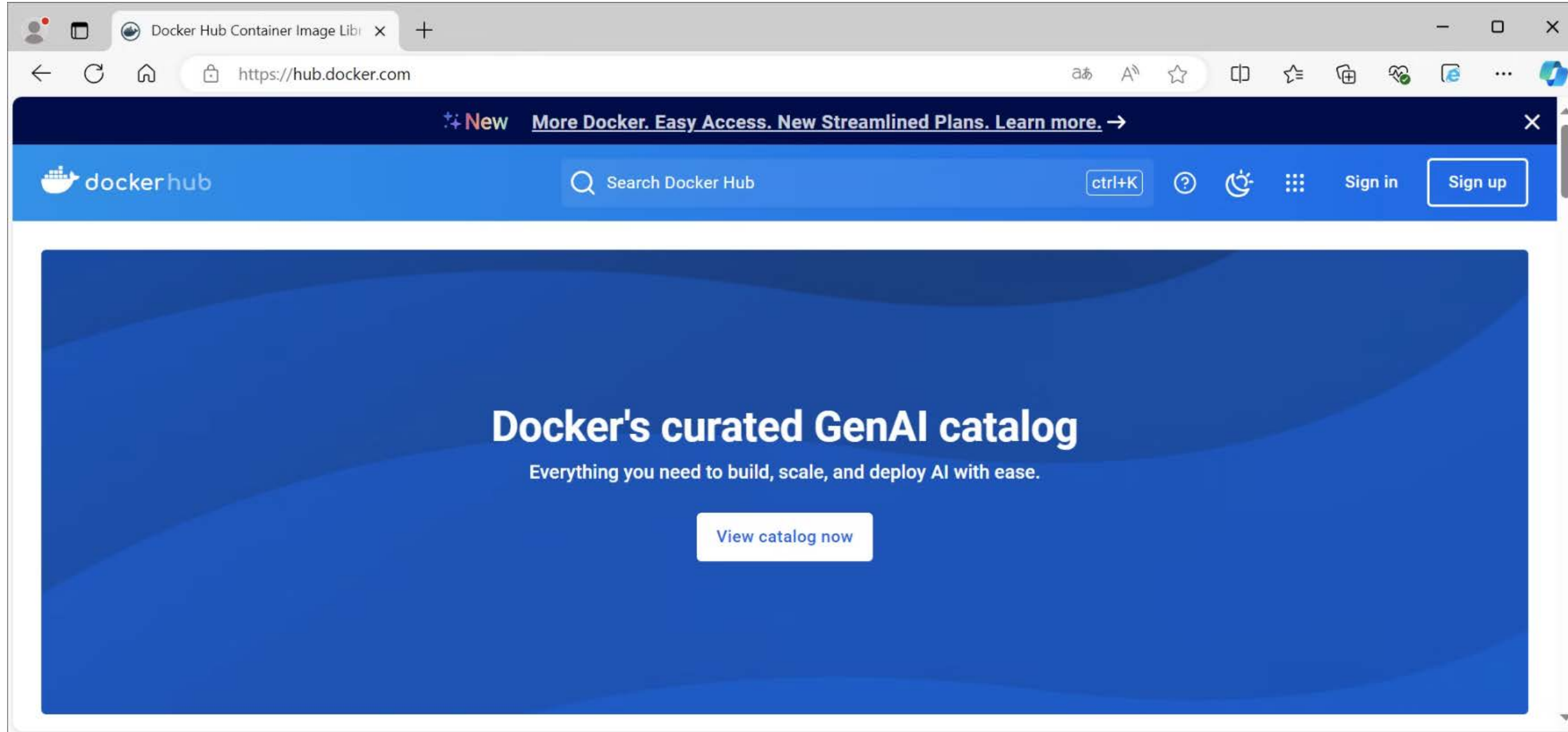
11.DockerHub 이미지 등록 확인

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

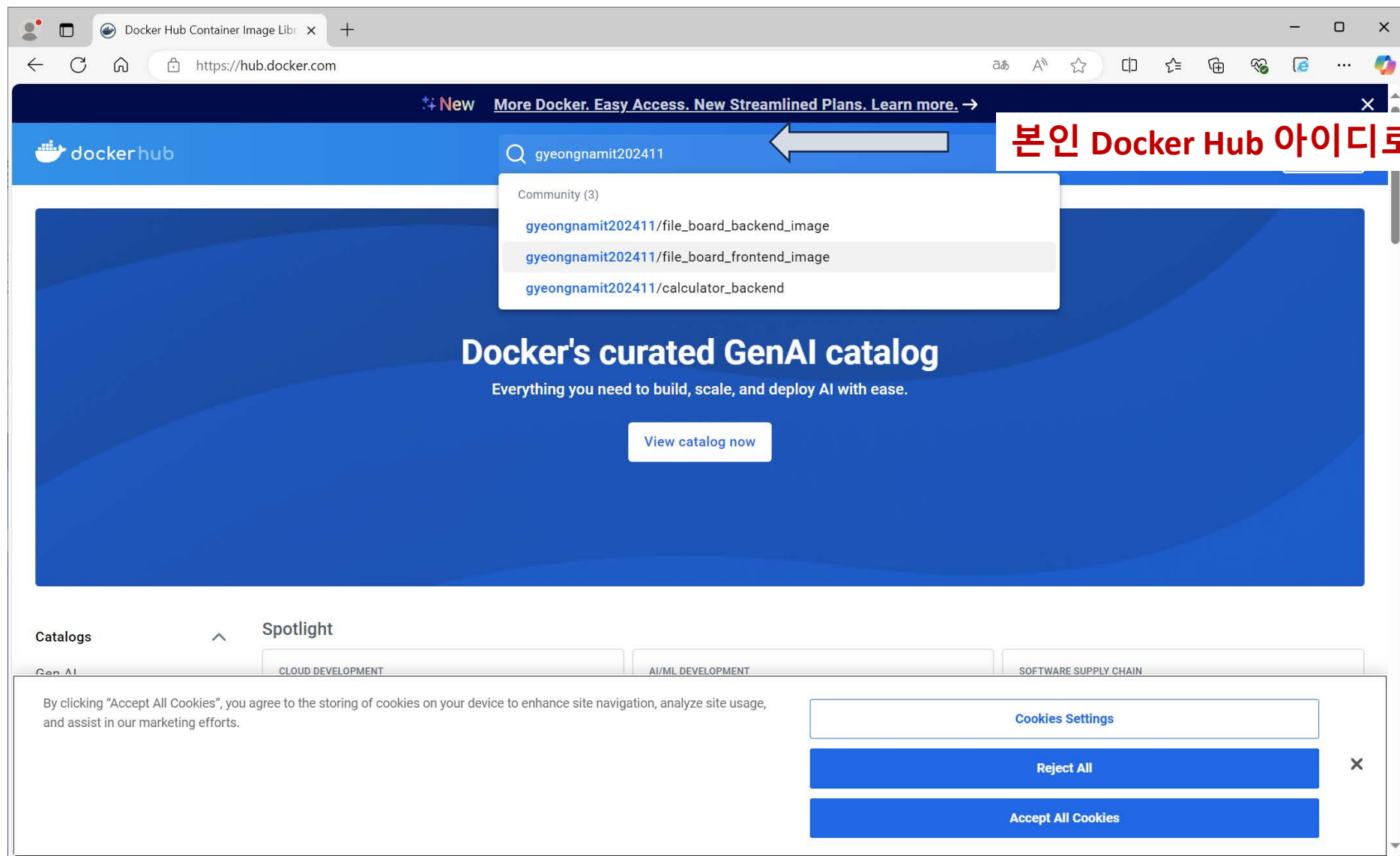
<https://hub.docker.com/> 접속 합니다



DockerHub 이미지 등록 확인

본인 Docker Hub 아이디로 검색 합니다

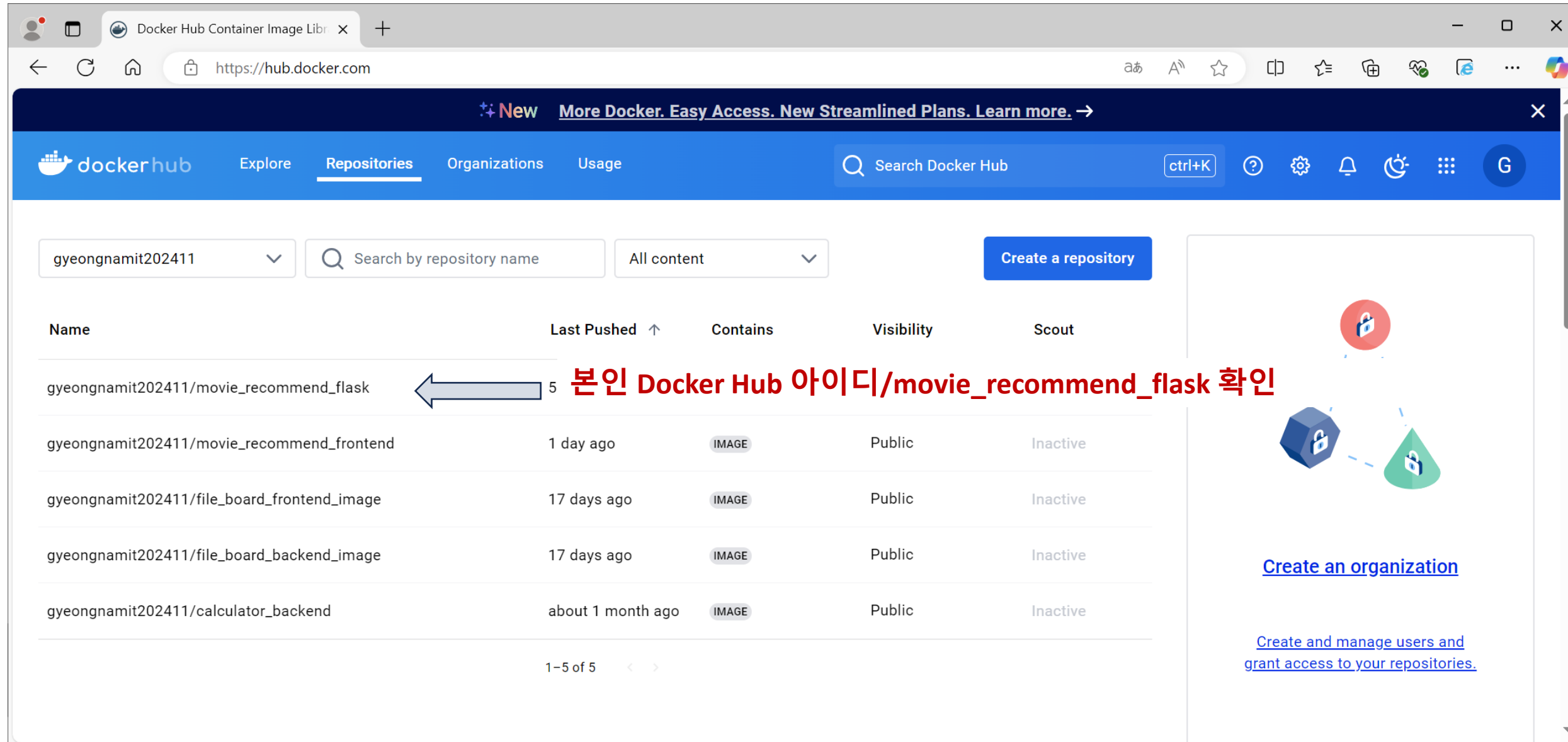
gyeongnamit202411 가 아이디 라면 gyeongnamit202411 로 검색 합니다



본인 Docker Hub 아이디로 검색

이미지 생성 확인

본인 Docker Hub 아이디/movie_recommend_flask 가 검색 되는지 확인 합니다



이미지 생성 확인

본인 Docker Hub 아이디/movie_recommend_flask 클릭 합니다

Docker Hub Container Image Libr

×

+

←

↺

🏠

🔒 https://hub.docker.com

🔍 a

🔊 A

☆

📄

☆

🔖

🔒

🔗

🔗

⋮

🌐

New

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

×

dockerhub

Explore

Repositories

Organizations

Usage

🔍

Search Docker Hub

ctrl+K

?

⚙️

🔔

🔄

⋮

G

gyeongnamit202411

▼

🔍

Search by repository name

All content

▼

Create a repository

Name	Last Pushed ↑	Contains	Visibility	Scout
gyeongnamit202411/movie_recommend_flask	5 minutes ago	IMAGE	Public	Inactive
gyeongnamit202411/movie_recommend_frontend	1 day ago	IMAGE	Public	Inactive
gyeongnamit202411/file_board_frontend_image	17 days ago	IMAGE	Public	Inactive
gyeongnamit202411/file_board_backend_image	17 days ago	IMAGE	Public	Inactive
gyeongnamit202411/calculator_backend	about 1 month ago	IMAGE	Public	Inactive

1-5 of 5 < >

Create an organization

Create and manage users and grant access to your repositories.

이미지 생성 확인

Docker 이미지가 생성 되었는지 확인 합니다

gyeongnamit202411/movie_recor

x

+

←

↺

🏠

🔒

https://hub.docker.com/repository/docker/gyeongnamit202411/movie_recommend_flask/general

あ

A

☆

📄

☆

🔒

🔗

🌐

⋮

🌈

🌟 New

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

🚢 docker hub

Explore

Repositories

Organizations

Usage

🔍 Search Docker Hub

ctrl+K

?

⚙️

🔔

🌙

☰

G

gyeongnamit202411 / Repositories / movie_recommend_flask / General

Using 0 of 1 private repositories.

General

Tags

Builds

Collaborators

Webhooks

Settings

gyeongnamit202411/movie_recommend_flask

🔒

Last pushed 5 minutes ago

Add a description ✎

INCOMPLETE

Add a category ✎

INCOMPLETE

Docker commands

Public view

To push a new tag to this repository:

docker push gyeongnamit202411/movie_recommend_flask:tagname

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
<div>latest</div>	<div>🐧</div>	Image	an hour ago	5 minutes ago

[See all](#)

Automated builds

Manually pushing images to Docker Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#) 🔗.

How likely are you to recommend Docker Hub to another developer?

Not at all likely

0

1

2

3

4

5

6

7

8

9

10

Extremely likely

powered by InMoment

12. Visual Studio 종료

Visual Studio 종료

모든 Visual Studio Code 창을 종료 합니다

(영화 추천 백엔드와 프론트 엔드)

