

Architecture

uv

- uv는 정말 빠른 Python 패키지 및 프로젝트 매니저라고 하며, Rust를 기반으로 만들어졌습니다.
 - Rust(러스트)는 모질라 리서치에서 개발한 시스템 프로그래밍 언어로, 성능과 안전성을 중시합니다.

단계1: uv 설치

- On Windows.

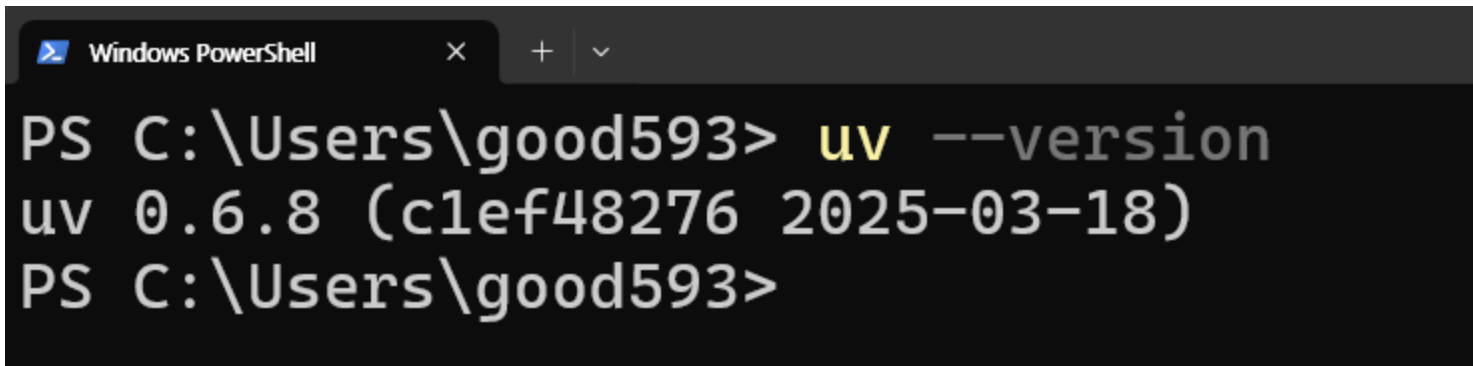
```
# On Windows.  
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

- On macOS and Linux.

```
# On macOS and Linux.  
curl -LsSf https://astral.sh/uv/install.sh | sh
```

단계2: uv 설치 확인

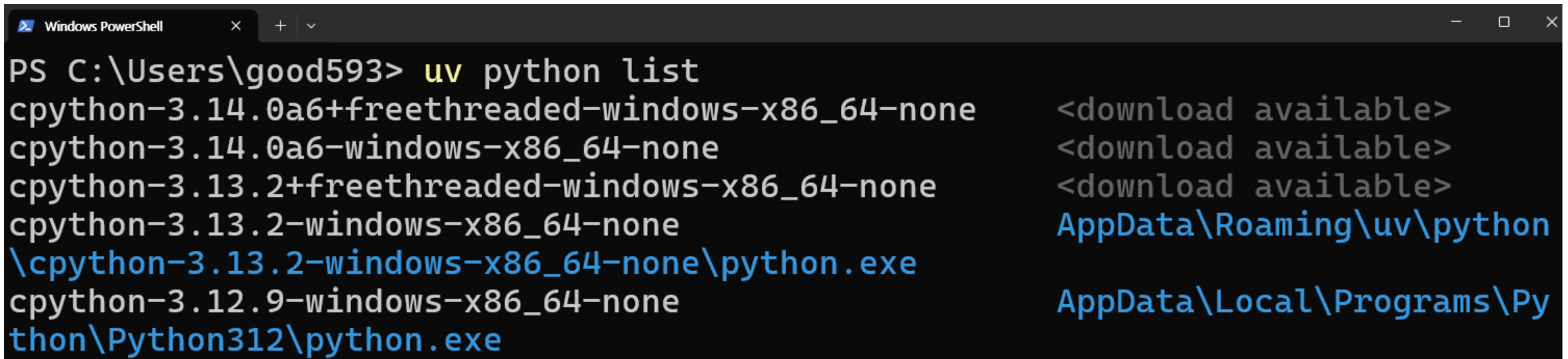
```
uv --version
```



```
Windows PowerShell
PS C:\Users\good593> uv --version
uv 0.6.8 (c1ef48276 2025-03-18)
PS C:\Users\good593>
```

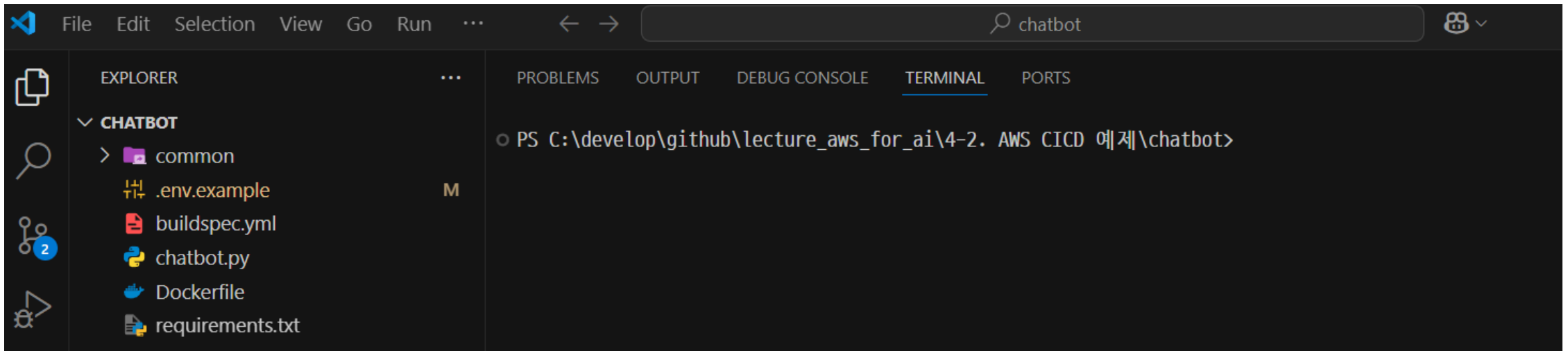
단계3: uv 파이썬 설치 확인

```
uv python list
```



```
Windows PowerShell
PS C:\Users\good593> uv python list
cpython-3.14.0a6+freethreaded-windows-x86_64-none <download available>
cpython-3.14.0a6-windows-x86_64-none <download available>
cpython-3.13.2+freethreaded-windows-x86_64-none <download available>
cpython-3.13.2-windows-x86_64-none AppData\Roaming\uv\python
\cpython-3.13.2-windows-x86_64-none\python.exe
cpython-3.12.9-windows-x86_64-none AppData\Local\Programs\Py
thon\Python312\python.exe
```

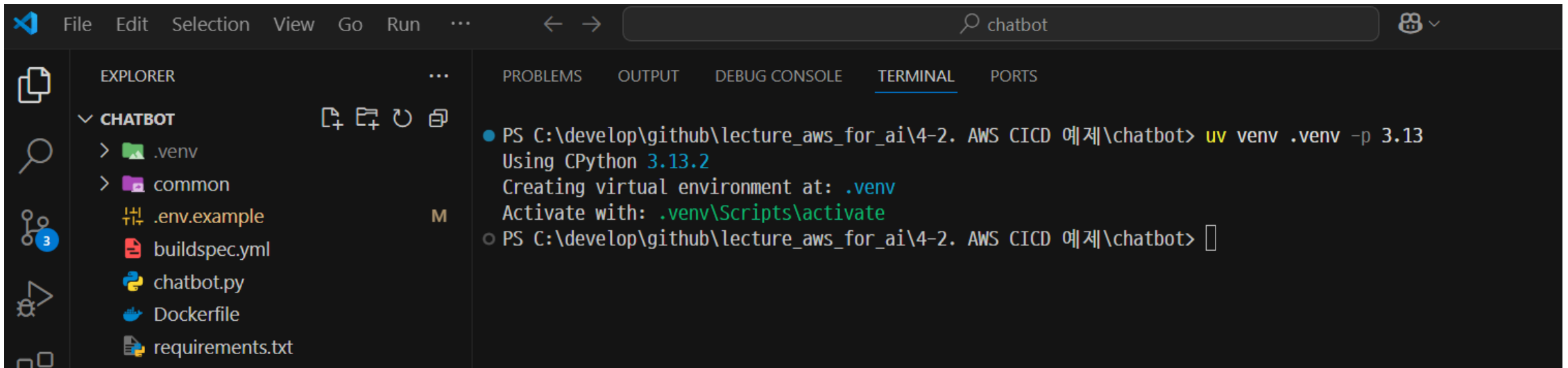
단계4: 프로젝트 폴더로 이동



단계5: 가상환경 생성

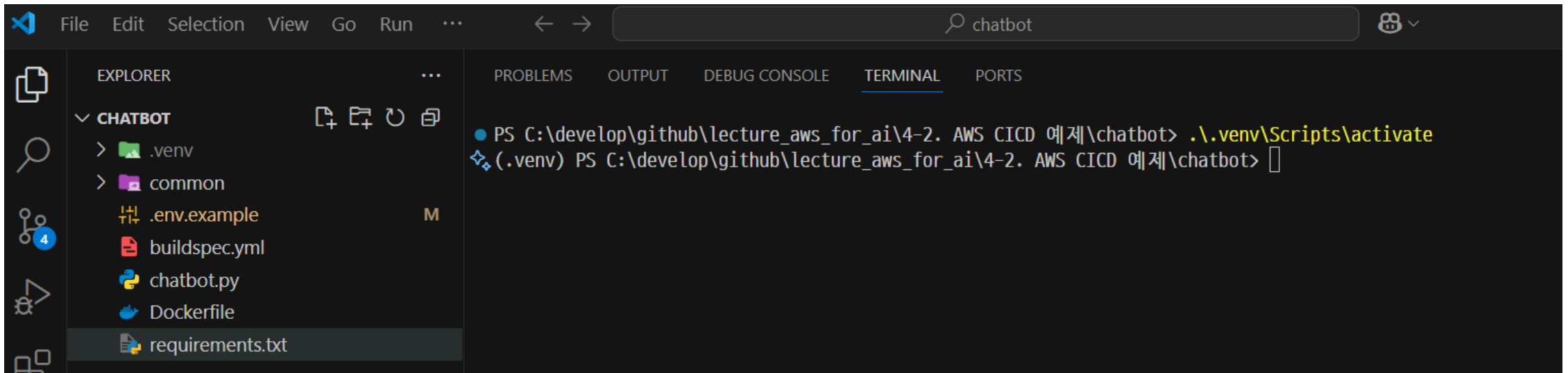
uv venv [가상환경명] -p [파이썬버전]

```
uv venv .venv -p 3.13
```



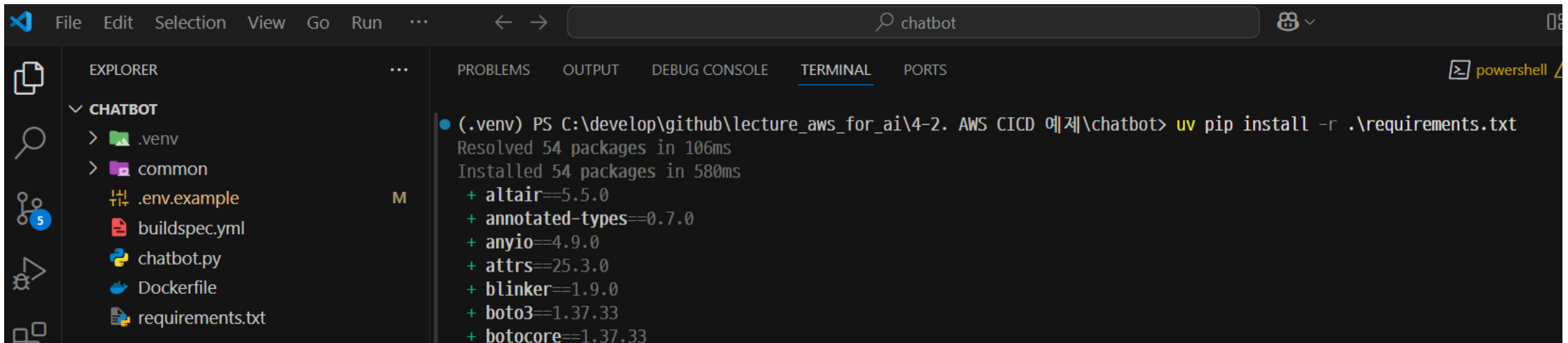
단계6: 가상환경 접속

```
.\.venv\Scripts\activate
```



단계7: 라이브러리 설치

```
uv pip install -r .\requirements.txt
```



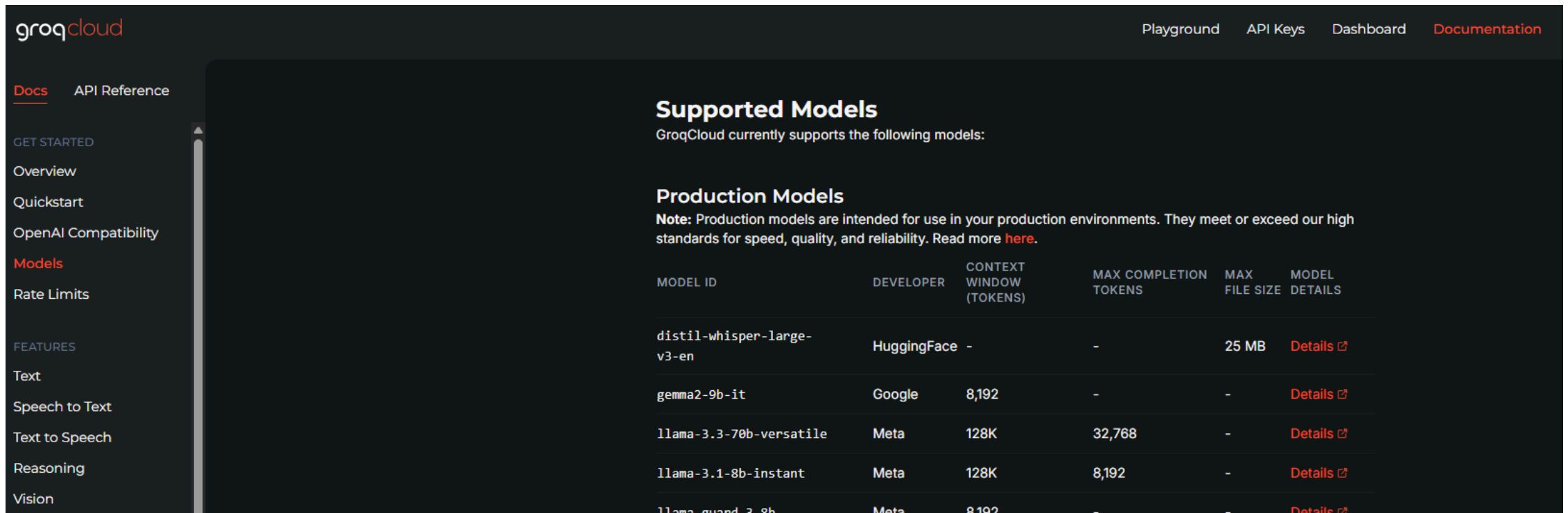
The screenshot shows the Visual Studio Code interface with the Explorer, Problems, Output, Debug Console, and Terminal panels. The Explorer panel on the left shows a project named 'CHATBOT' with files: .env, common, .env.example, buildspec.yml, chatbot.py, Dockerfile, and requirements.txt. The Terminal panel on the right shows the command 'uv pip install -r .\requirements.txt' being executed in a PowerShell prompt. The output indicates that 54 packages were resolved and installed successfully, listing several dependencies with their versions.

```
(.venv) PS C:\develop\github\lecture_aws_for_ai\4-2. AWS CICD 예제\chatbot> uv pip install -r .\requirements.txt
Resolved 54 packages in 106ms
Installed 54 packages in 580ms
+ altair==5.5.0
+ annotated-types==0.7.0
+ anyio==4.9.0
+ attrs==25.3.0
+ blinker==1.9.0
+ boto3==1.37.33
+ botocore==1.37.33
```

Groq Cloud

- Groq API는 AI 모델의 추론 속도가 매우 빠른 API 서비스입니다.
- 구글 Gemma, 메타 Llama 등 오픈소스 LLM을 GPU 없는 환경에서도 빠르게 테스트하고 서비스를 개발할 수 있는 환경을 제공합니다. (현재 시점에서는 무료 사용이 가능합니다.)

Groq Cloud Models



The screenshot shows the Groq Cloud website interface. The top navigation bar includes links for Playground, API Keys, Dashboard, and Documentation. The left sidebar contains a menu with sections like GET STARTED, FEATURES, and Models. The main content area is titled 'Supported Models' and lists production models in a table.

Supported Models
GroqCloud currently supports the following models:

Production Models
Note: Production models are intended for use in your production environments. They meet or exceed our high standards for speed, quality, and reliability. Read more [here](#).

MODEL ID	DEVELOPER	CONTEXT WINDOW (TOKENS)	MAX COMPLETION TOKENS	MAX FILE SIZE	MODEL DETAILS
distil-whisper-large-v3-en	HuggingFace	-	-	25 MB	Details
gemma2-9b-it	Google	8,192	-	-	Details
llama-3.3-70b-versatile	Meta	128K	32,768	-	Details
llama-3.1-8b-instant	Meta	128K	8,192	-	Details
llama-guard-3-8b	Meta	8,192	-	-	Details

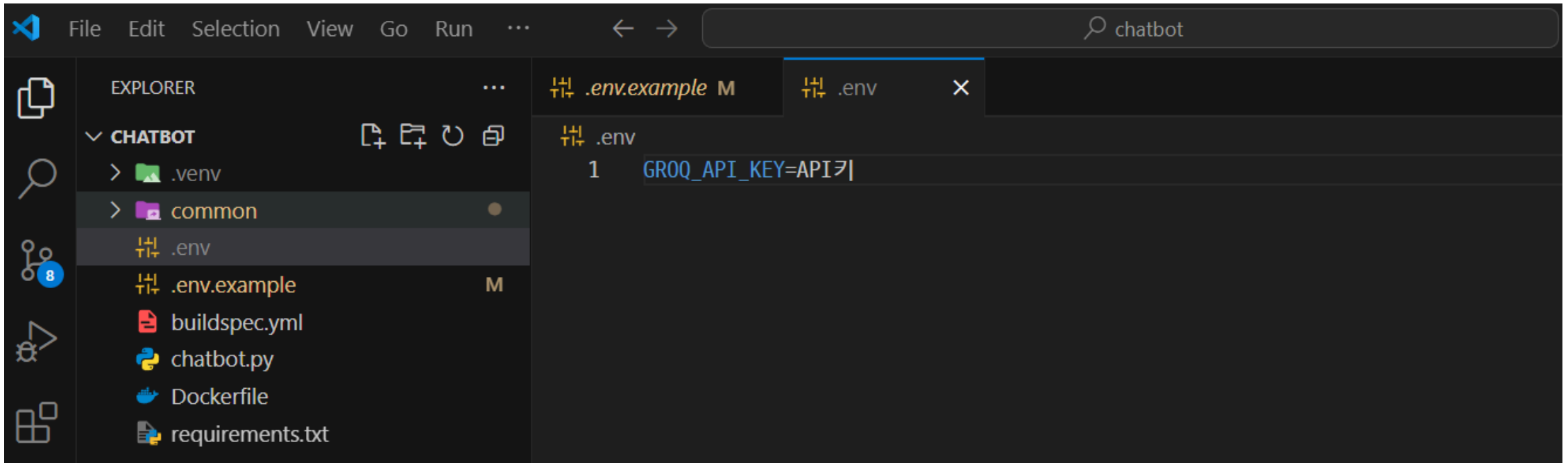
단계1: Groq Model 적용

```
File Edit Selection View ... chatbot
EXPLORER
  CHATBOT
    .venv
    common
      __pycache__
      chat.py M
      constant.py
      message.py
      utils.py
    .env
    .env.example M
    buildspec.yml
    chatbot.py

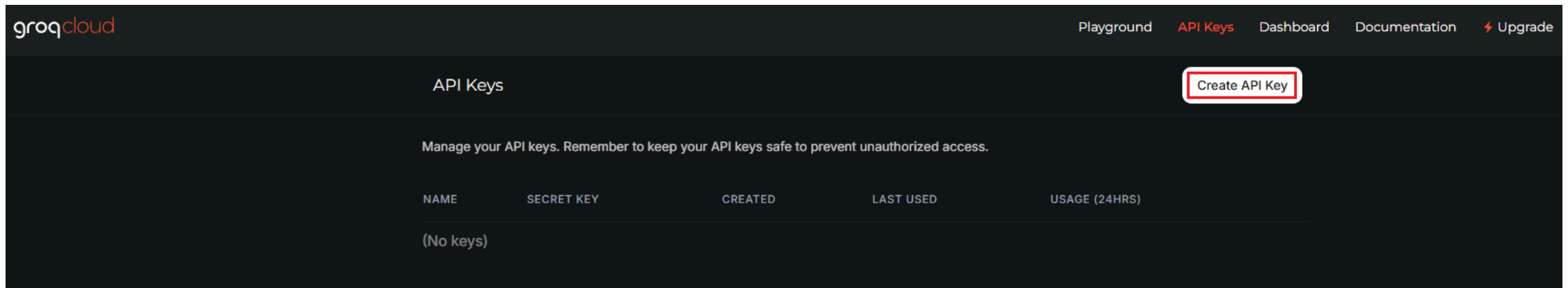
chat.py M X
common > chat.py > response_from_llm
1 import streamlit as st
2 from groq import Groq
3 import time
4
5 # @st.cache_data # 데이터를 caching 처리
6 @st.cache_resource # 객체를 caching 처리
7 def get_client():
8     return Groq()
9
10 def response_from_llm(prompt, message_history=[], model_id:str="llama-3.1-8b-instant"):
11     if len(message_history) == 0:
12         # 최초 질문
13         message_history.append(
```

단계2: .env 파일 생성

.env.example 파일을 이용해서 .env 파일 생성



단계3: Groq key 발급



The screenshot shows the Groq Cloud API Keys management interface. At the top, the Groq Cloud logo is on the left, and navigation links for Playground, API Keys, Dashboard, Documentation, and Upgrade are on the right. The main heading is "API Keys", with a "Create API Key" button to its right. Below this, a message states: "Manage your API keys. Remember to keep your API keys safe to prevent unauthorized access." A table with columns NAME, SECRET KEY, CREATED, LAST USED, and USAGE (24HRS) follows. The table is currently empty, displaying "(No keys)".

groqcloud

Playground API Keys Dashboard Documentation ⚡ Upgrade

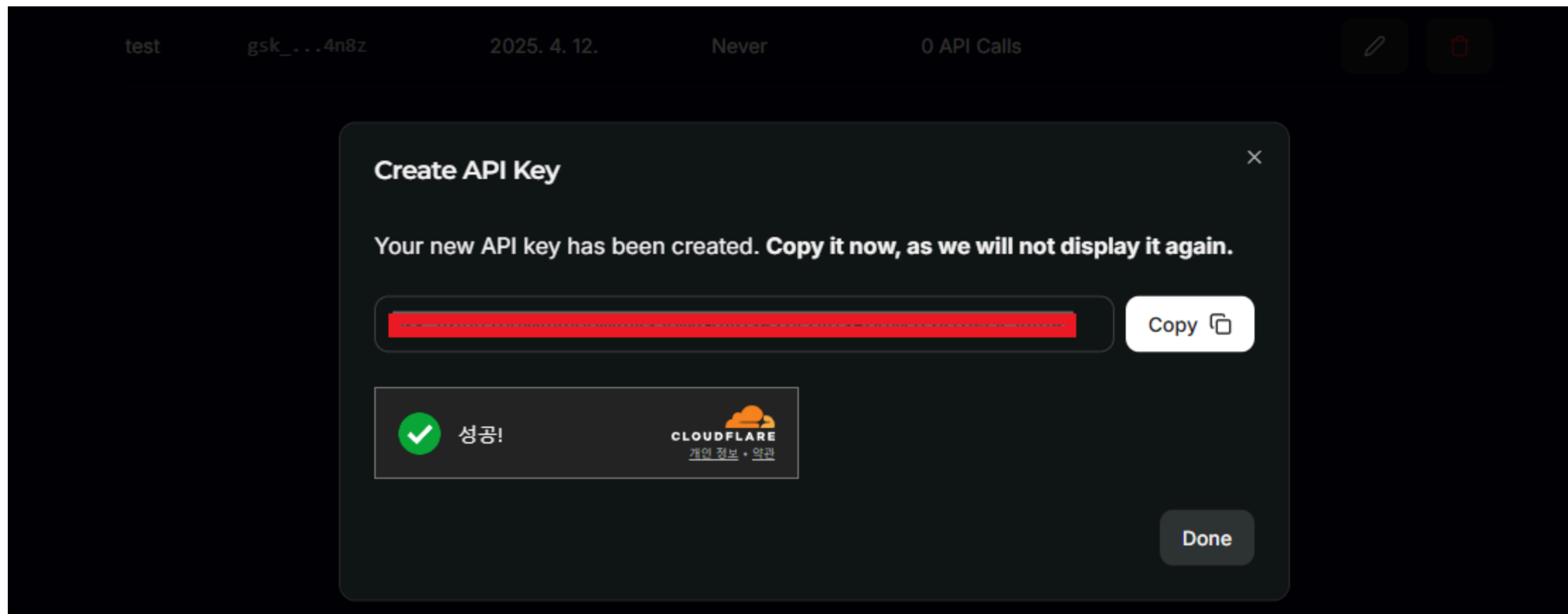
API Keys

Create API Key

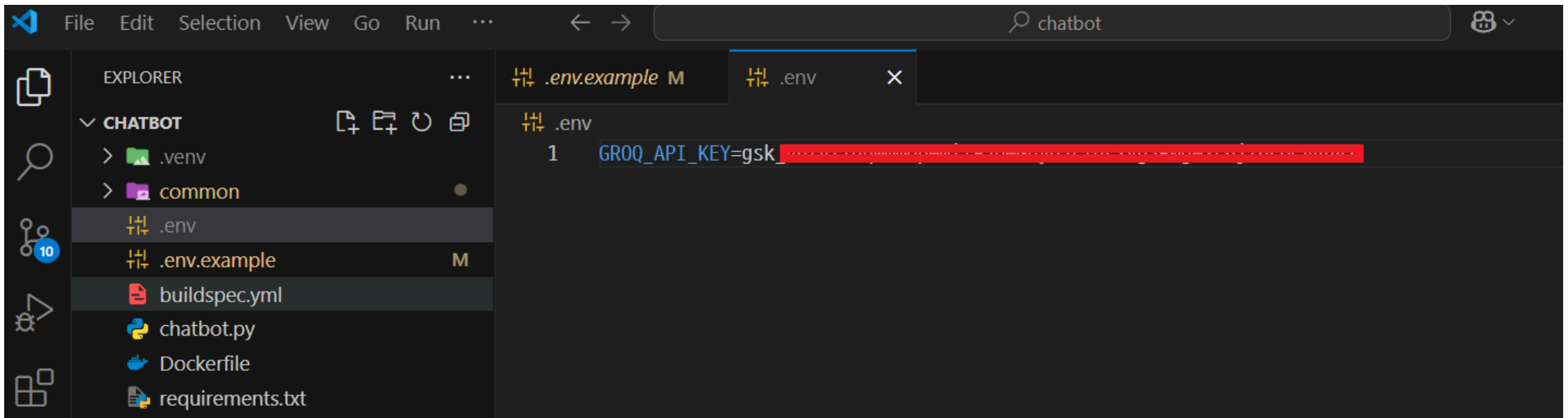
Manage your API keys. Remember to keep your API keys safe to prevent unauthorized access.

NAME	SECRET KEY	CREATED	LAST USED	USAGE (24HRS)
(No keys)				

- Groq key 복사



단계4: .env 파일에 Groq key 적용

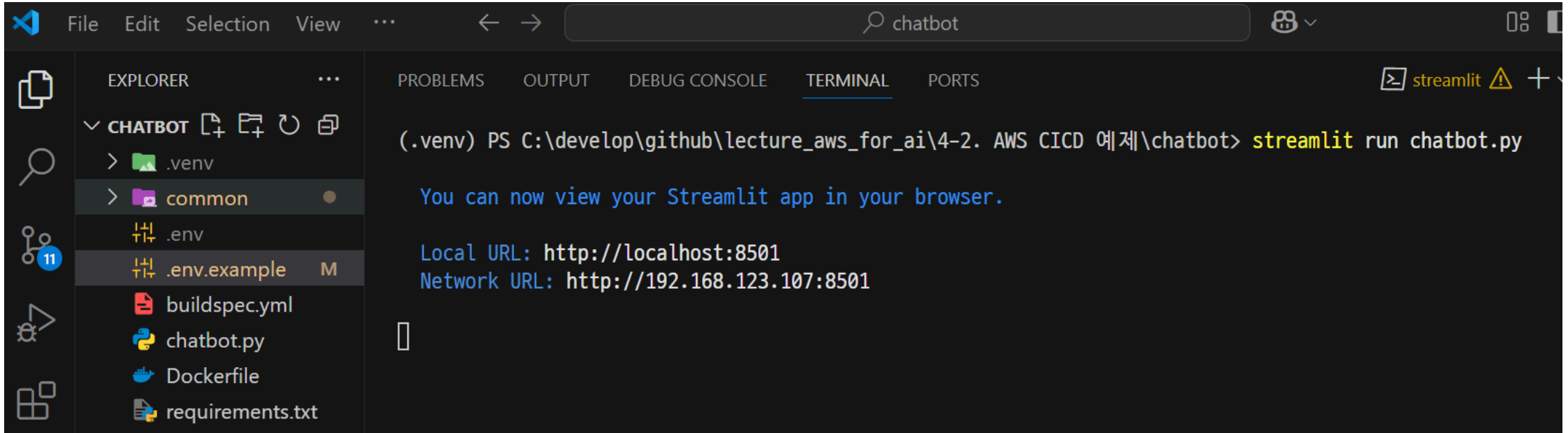


Streamlit

- Streamlit을 설명한 웹 페이지를 가 보면 "A faster way to build and share data apps (데이터 어플리케이션을 만들고 공유하는 빠른 방법)"이라고 설명되어 있습니다.
- 즉 데이터를 보여주는 것에 초점을 맞춘 데모용 웹 프레임워크라고 보면 될 것 같습니다.

단계1: Streamlit 실행

```
streamlit run chatbot.py
```



The screenshot shows the Visual Studio Code interface with the following components:

- Explorer Panel:** Displays the project structure for 'CHATBOT'. It includes a '.venv' folder, a 'common' folder, and files: '.env', '.env.example' (marked with 'M'), 'buildspec.yml', 'chatbot.py', 'Dockerfile', and 'requirements.txt'.
- Terminal Panel:** Shows the command prompt output. The command executed is `streamlit run chatbot.py`. The output indicates that the app is running and provides the following URLs:
 - Local URL: `http://localhost:8501`
 - Network URL: `http://192.168.123.107:8501`
- Streamlit Status:** A small icon in the top right corner of the terminal panel shows a green play button and a yellow warning triangle, with the text 'streamlit' next to it.

단계2: Chatbot 테스트

Local URL: `http://localhost:8501`

