

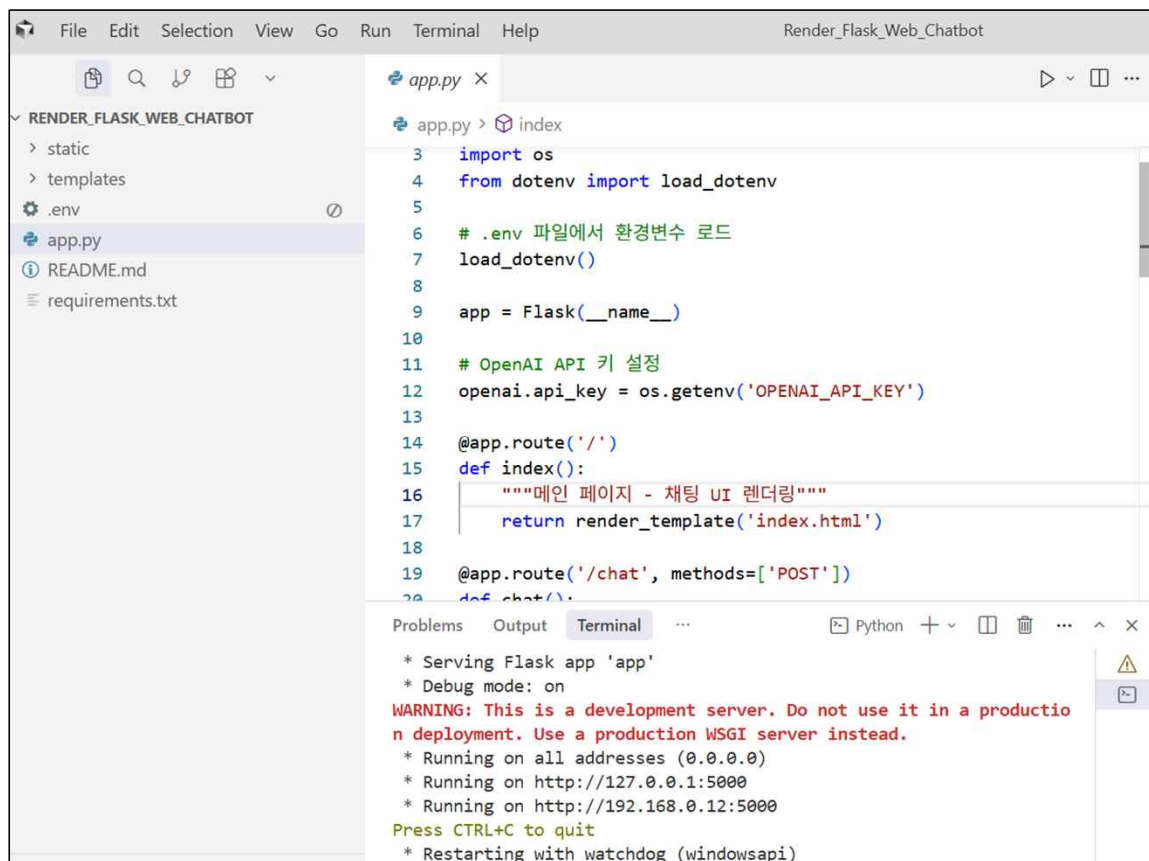
# Render로 Flask Web Chatbot 배포하기



# 1. Flask 웹 챗봇 소스 준비

---

Flask 를 사용한 웹 챗봇 소스를 복사해와서  
Render\_Flask\_Web\_Chatbot 폴더로 이름을  
변경하고 커서를 사용하여 동작을 확인한다

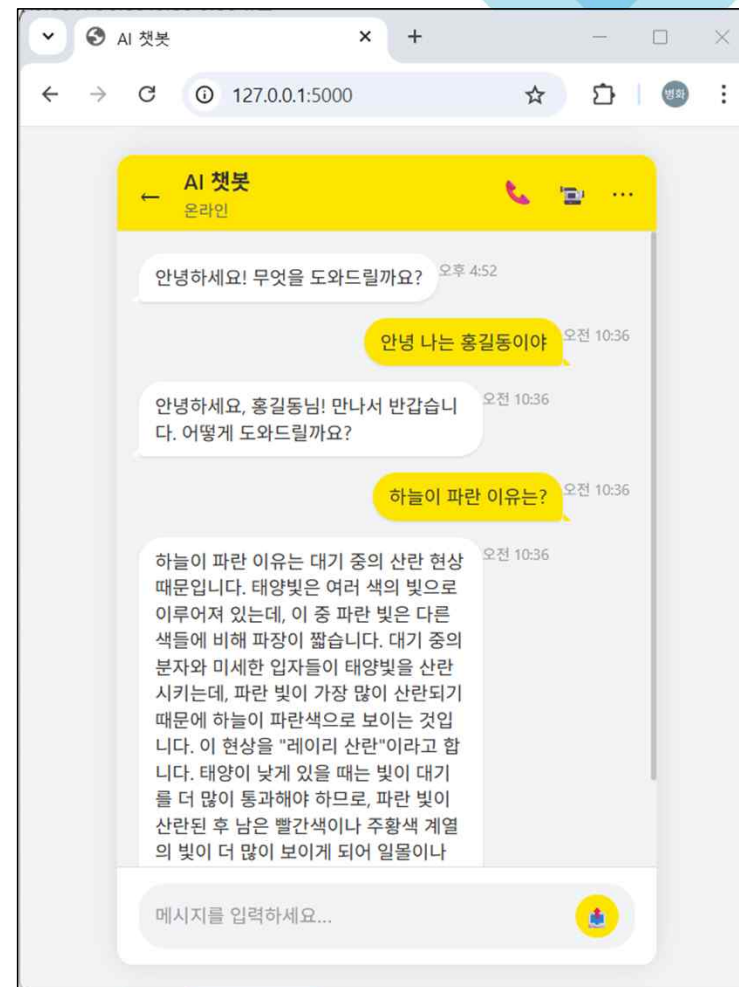


The screenshot shows a code editor with the file 'app.py' open. The code defines a Flask application with a single route '/' that renders the 'index.html' template. The terminal output shows the application running successfully on http://127.0.0.1:5000.

```
app.py
3 import os
4 from dotenv import load_dotenv
5
6 # .env 파일에서 환경변수 로드
7 load_dotenv()
8
9 app = Flask(__name__)
10
11 # OpenAI API 키 설정
12 openai.api_key = os.getenv('OPENAI_API_KEY')
13
14 @app.route('/')
15 def index():
16     """메인 페이지 - 채팅 UI 렌더링"""
17     return render_template('index.html')
18
19 @app.route('/chat', methods=['POST'])
20 def chat():
```

Terminal Output:

```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production
n deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.0.12:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
```



- **Requirements.txt** 파일에 **gunicorn**을 추가한다

python-dotenv는 .env를 사용하지 않으므로 필요 없으므로 삭제한다(주석 처리)

**Flask==3.0.3**

**Openai>=1.5.0**

*# python-dotenv==1.0.0*

**Gunicorn==22.0.0**

- **app.py** 소스에 가서 **dotenv** 사용 부분(파란색 2줄)을 주석 처리하고 저장한다

```
from flask import Flask, render_template, request, jsonify
```

```
import openai
```

```
import os
```

```
# from dotenv import load_dotenv
```

```
# .env 파일에서 환경변수 로드
```

```
# load_dotenv()
```

.gitignore 파일을 만들고 아래 내용을 넣어 커밋 시 무시해야 할 파일을 지정한다.

```
# Python
__pycache__/
*.pyc
*.pyo
*.pyd

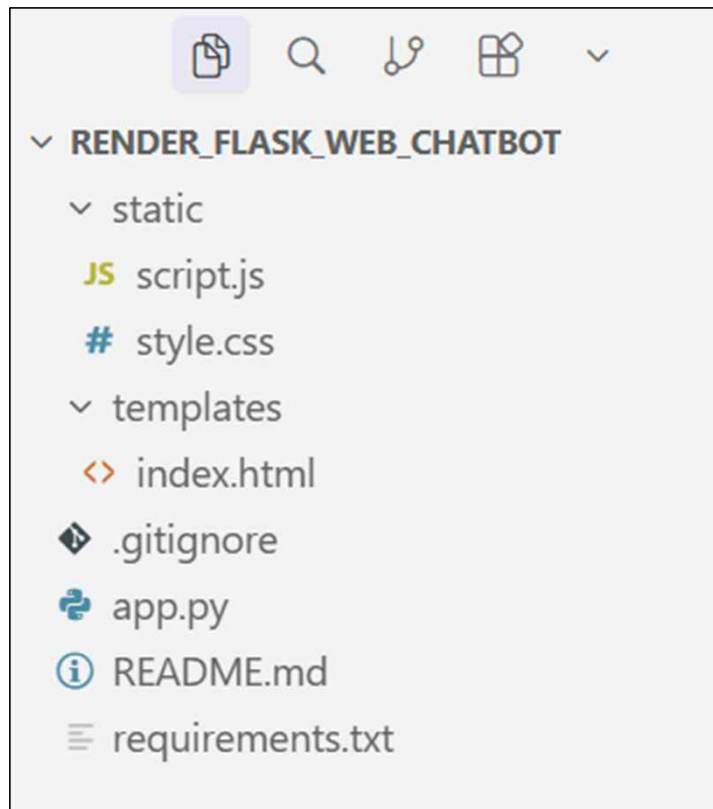
# Virtual env
.venv/
venv/

# OS / IDE
.DS_Store
.vscode/
.idea/

# Logs
*.log
```

- .env 파일은 Git 저장소에 올리면 보안상 위험 하므로 삭제한다  
Render에서 수동으로 환경변수에 OpenAI API 키 값을 지정해야 하므로 다른 곳에 사본  
을 복사해놓고 프로젝트내에서 삭제한다

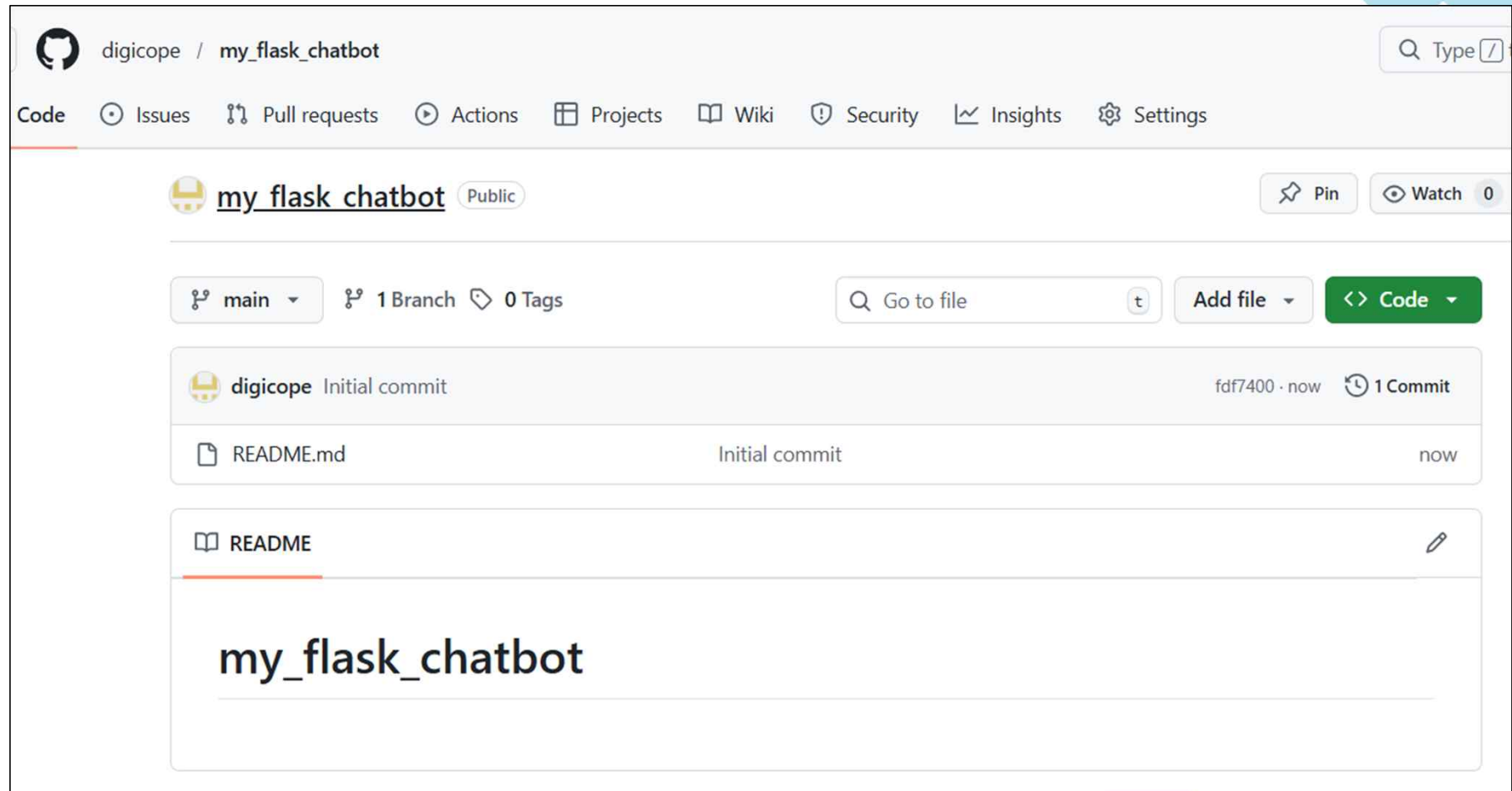
변경된 프로젝트 내의 소스 파일 구조



## 2. Git 저장소에 배포 소스 업로드

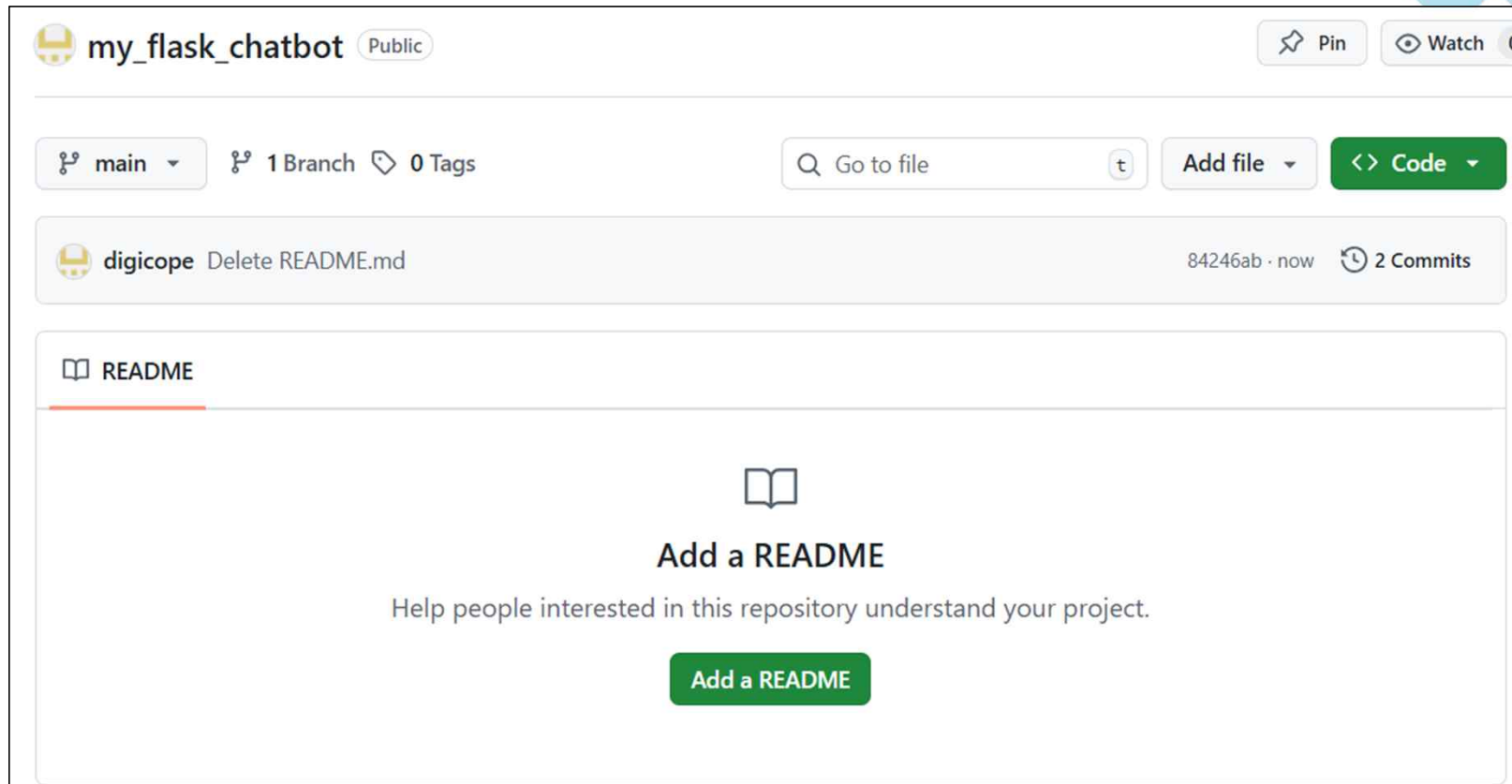
---

- Git에 새로운 저장소를 **my\_flask\_chatbot** 이름으로 생성해 놓는다



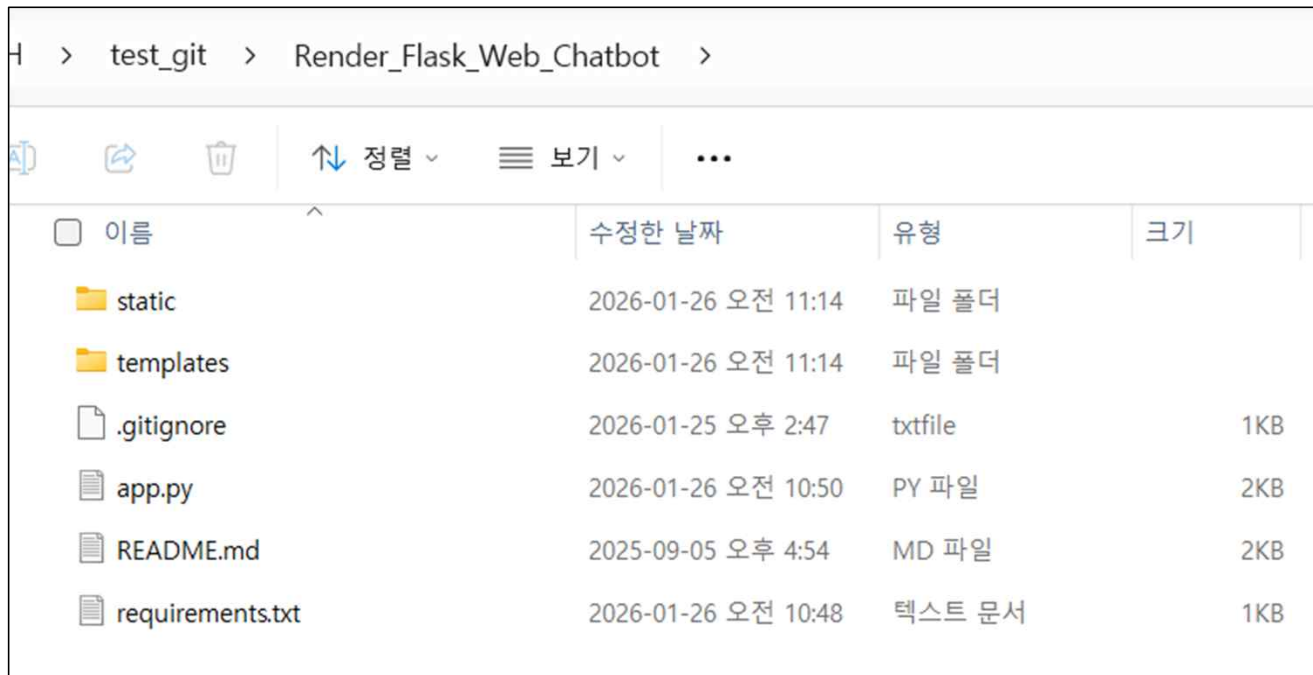


- 사용할 Github 저장소로 가서 파일을 모두 삭제해 놓는다 (README.md 파일 반드시 삭제)



- 윈도우 탐색기에서 test\_git 폴더 아래 배포할 앱을 복사해 놓는다  
(강사 배포 파일 사용시 : Render\_Flask\_Web\_Chatbot.zip )

실제 경로: C:\Users\storm\test\_git\Render\_Flask\_Web\_Chatbot



<input type="checkbox"/> 이름	수정한 날짜	유형	크기
static	2026-01-26 오전 11:14	파일 폴더	
templates	2026-01-26 오전 11:14	파일 폴더	
.gitignore	2026-01-25 오후 2:47	txtfile	1KB
app.py	2026-01-26 오전 10:50	PY 파일	2KB
README.md	2025-09-05 오후 4:54	MD 파일	2KB
requirements.txt	2026-01-26 오전 10:48	텍스트 문서	1KB

- Windows에서 Git Bash를 실행하고 아래 경로로 이동한다

**cd test\_git**

- 소스 경로로 이동한다

**cd Render\_Flask\_Web\_Chatbot**

- 파일을 확인해 본다

**ls**

```
MINGW64:/c/Users/storm/test_git/Render_Flask_Web_Chatbot

storm@HP-ENVY360-PC MINGW64 ~
$ cd test_git/

storm@HP-ENVY360-PC MINGW64 ~/test_git
$ cd Render_Flask_Web_Chatbot/

storm@HP-ENVY360-PC MINGW64 ~/test_git/Render_Flask_web_Chatbot
$ ls
README.md  app.py  requirements.txt  static/  templates/

storm@HP-ENVY360-PC MINGW64 ~/test_git/Render_Flask_web_Chatbot
$
```

## Git 저장소에 소스 파일 업로드

아래 명령을 차례로 수행한다

(**user name**과 **email**, **저장소 경로**는 본인의 이름으로 모두 수정한 다음 실행한다)

```
git init
```

```
git config --global user.name digicope
```

```
git config --global user.email digicope@aicore.co.kr
```

```
git add .
```

```
git commit -m "initial flask chatbot for render"
```

```
git branch -M main
```

```
git remote add origin
```

```
https://github.com/digicope/my\_flask\_chatbot.git
```

```
git pull origin main --rebase
```

```
git push -u origin main
```

## Git 저장소에서 업로드 된 파일들을 확인한다

The screenshot shows the GitHub interface for the repository 'my\_flask\_chatbot' by user 'digicope'. The repository is public and has 1 branch (main) and 0 tags. The file list shows the following files and folders, all committed by 'digicope' at '66fd881' (now) with 3 commits:

File/Folder	Commit Message	Commit Hash	Time
static	initial flask chatbot for render	66fd881	now
templates	initial flask chatbot for render	66fd881	now
.gitignore	initial flask chatbot for render	66fd881	now
README.md	initial flask chatbot for render	66fd881	now
app.py	initial flask chatbot for render	66fd881	now
requirements.txt	initial flask chatbot for render	66fd881	now

Below the file list, the README content is visible, starting with the title 'Flask 웹 챗봇'.

### 3. Render 에서 서비스 배포하기



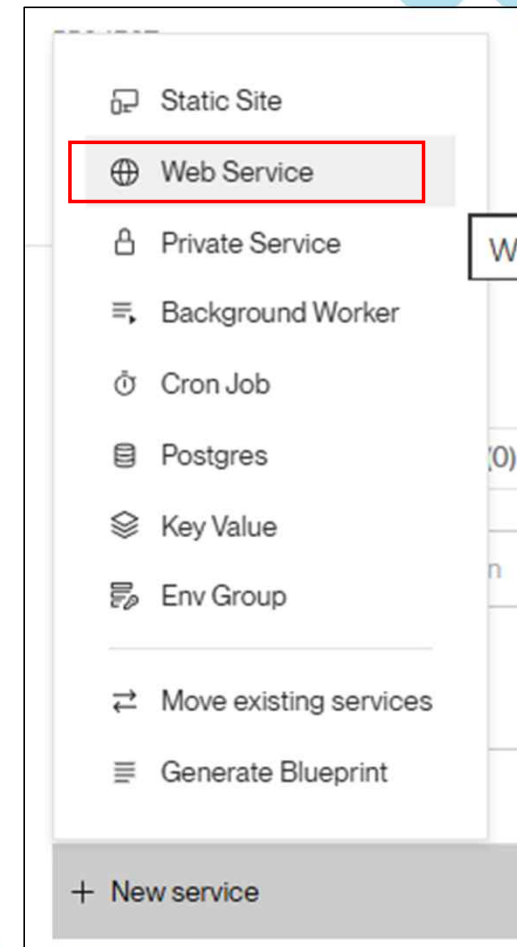
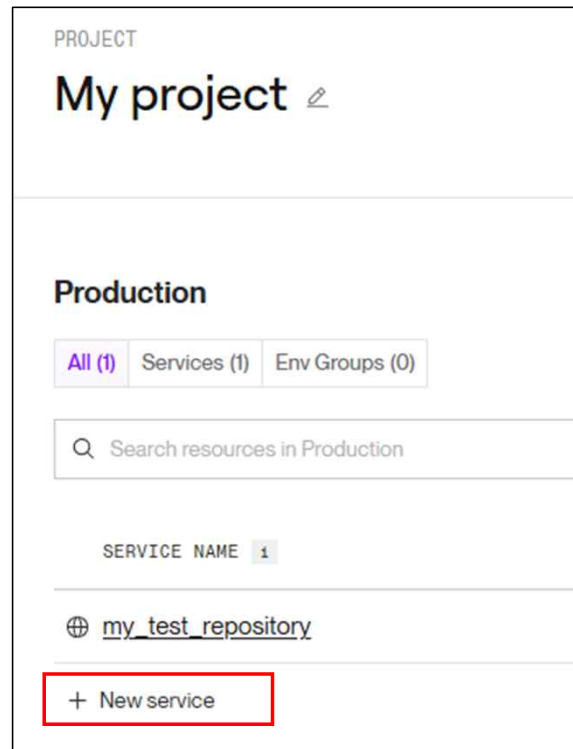
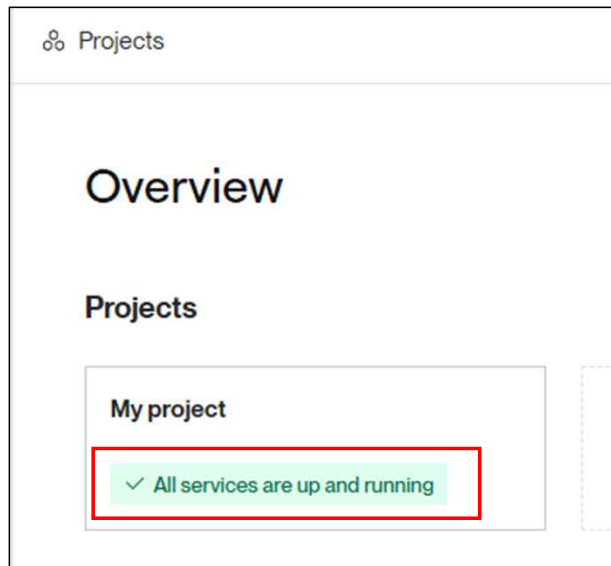
Render 홈페이지 대시보드로 가서 **Web Services**의 **New Web Service** 를 누른다

The screenshot shows the Render dashboard interface. At the top, there's a navigation bar with 'My Workspace' and 'Projects'. Below this, a section titled 'Create a new Cron Job' is visible, with a progress indicator showing '1 Choose service', '2 Configure', and '3 Deploy'. The main content area displays six service options in a grid:

- Static Sites**: Static content served over a global CDN. Ideal for frontend, blogs, and content sites. Link: [New Static Site →](#)
- Web Services**: Dynamic web app. Ideal for full-stack apps, API servers, and mobile backends. Link: [New Web Service →](#) (highlighted with a red box)
- Private Services**: Web app hosted on a private network, accessible only from your other Render services. Link: [New Private Service →](#)
- Cron Jobs**: Short-lived tasks that run on a periodic schedule. Link: [New Cron Job →](#)
- Postgres**: Relational data storage. Supports point-in-time recovery, read replicas, and high availability. Link: [New Postgres →](#)
- Key Value**: Managed Redis®-compatible storage. Ideal for use as a shared cache, message broker, or job queue. Link: [New Key Value Instance →](#)

<https://dashboard.render.com/>

Render에서 이미 배포를 위한 프로젝트가 생성 되어 있을 경우에는 **Projects**에 있는 **My project**를 아래를 클릭하고 하단의 **New service**를 클릭한다음 **[Web Service]**를 클릭한다






New Web Service에서 배포에 사용할 저장소 my\_flask\_chatbot을 선택해준다


Git Provider


Public Git Repository


Existing Image



Q Search



 digicope / my\_flask\_chatbot 2h ago



 digicope / ai\_vibe\_0112 3h ago

 digicope / my\_test\_repository 17h ago

 digicope / gcp\_lab\_1223 5d ago

 digicope /  gemini Dec 9, 2025

 digicope /  chatgpt-vibecoding-agent Nov 21, 2025

 digicope /  ai-agent-lab-0022 Nov 11, 2025

## New Web Service

It looks like you're using **Flask**, so we've autofilled some fields accordingly.

### Source Code


 digicope / my\_flask\_chatbot • 2h ago


 Edit

저장소를 선택 후 다음 항목을 설정한다.  
대부분 기본 값으로 사용하면 된다

- **Name** : 서비스 이름 (기본값인 **저장소 이름**을 사용한다)
- **Environment** : **Python 3**
- **Region** : 기본값 사용 (**Oregon (US West)**)
- **Branch** : **main** (또는 master)
- **Build Command** : **pip install -r requirements.txt**
- **Start Command** : **gunicorn app:app** (파일명이 app.py이고 Flask 객체가 app인 경우)

Source Code

 digicope / my\_flask\_chatbot • 2h ago

 Edit

Name

A unique name for your web service.

my\_flask\_chatbot

Project Optional

Add this web service to a [project](#) once it's created.

 My project

 / 

 Production

Language

Choose the [runtime environment](#) for this service.

Python 3

Branch

The Git branch to build and deploy.

main

Region

Your services in the same [region](#) can communicate over a [private network](#). You currently have services running in **Oregon**.

 Oregon (US West)

1 existing service

Deploy in a new region +

### Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a [monorepo](#).

```
e.g. src
```

### Build Command

Render runs this command to build your app before each deploy.

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

### Start Command

Render runs this command to start your app with each deploy.

```
$ gunicorn app:app
```

## Instance Type을 Free 로 선택한다

### Instance Type

#### For hobby projects

<b>Free</b>	512 MB (RAM)
<b>\$0 / month</b>	0.1 CPU

#### ⚠ Upgrade to enable more features

Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.

#### For professional use

For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:

- Zero Downtime
- SSH Access
- Scaling
- One-off jobs
- Support for persistent disks

<b>Starter</b>	512 MB (RAM)
<b>\$7 / month</b>	0.5 CPU

<b>Standard</b>	2 GB (RAM)
<b>\$25 / month</b>	1 CPU

<b>Pro</b>	4 GB (RAM)
<b>\$85 / month</b>	2 CPU

<b>Pro Plus</b>	8 GB (RAM)
<b>\$175 / month</b>	4 CPU

<b>Pro Max</b>	16 GB (RAM)
<b>\$225 / month</b>	4 CPU

<b>Pro Ultra</b>	32 GB (RAM)
<b>\$450 / month</b>	8 CPU

Need a [custom instance type](#)? We support up to 512 GB RAM and 64 CPUs.

**Environment Variables** 에 .env에 있는 환경변수를 입력한다

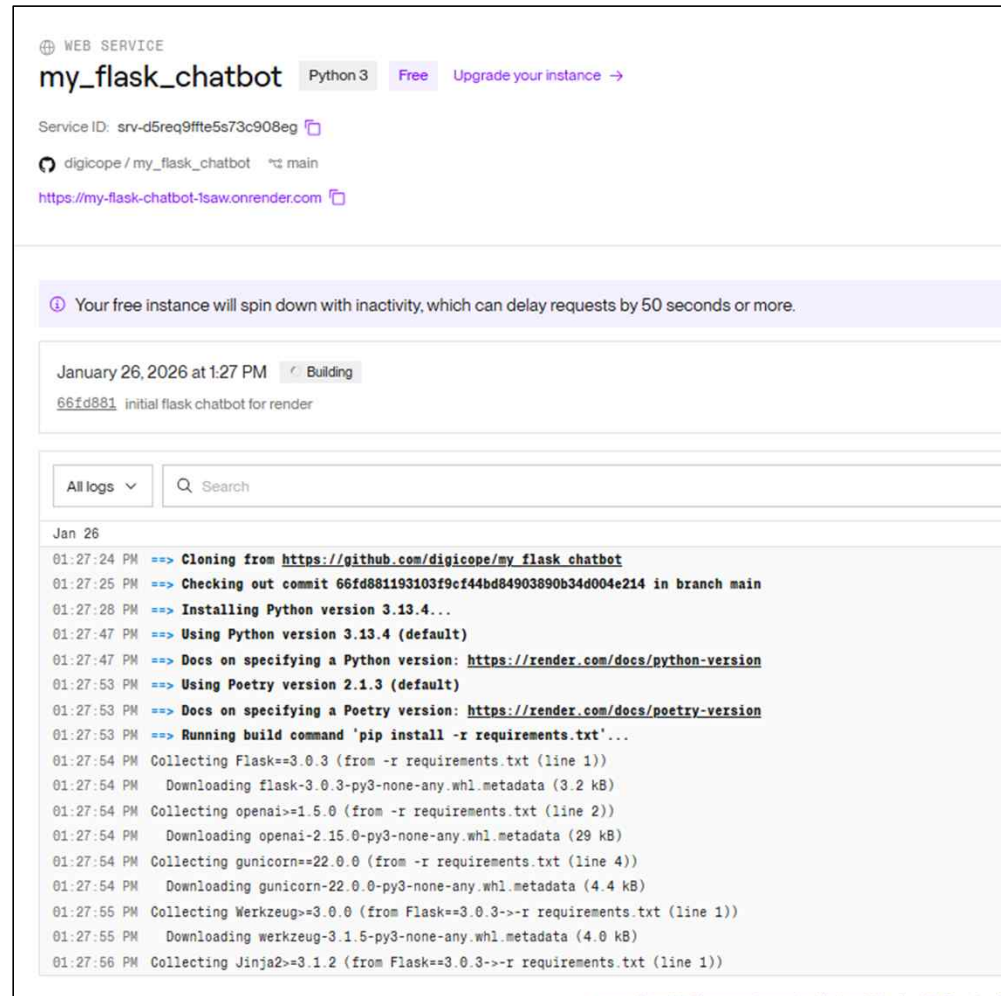
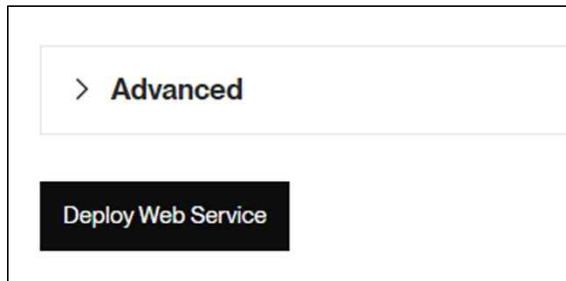
NAME\_OF\_VARIABLE : **OPENAI\_API\_KEY**

[illegible]

 Add from .env



Web Service 배포를 위한 설정이 완료되면 좌측 하단의 [Deploy Web Service]를 누르면 아래와 같이 배포가 진행 된다

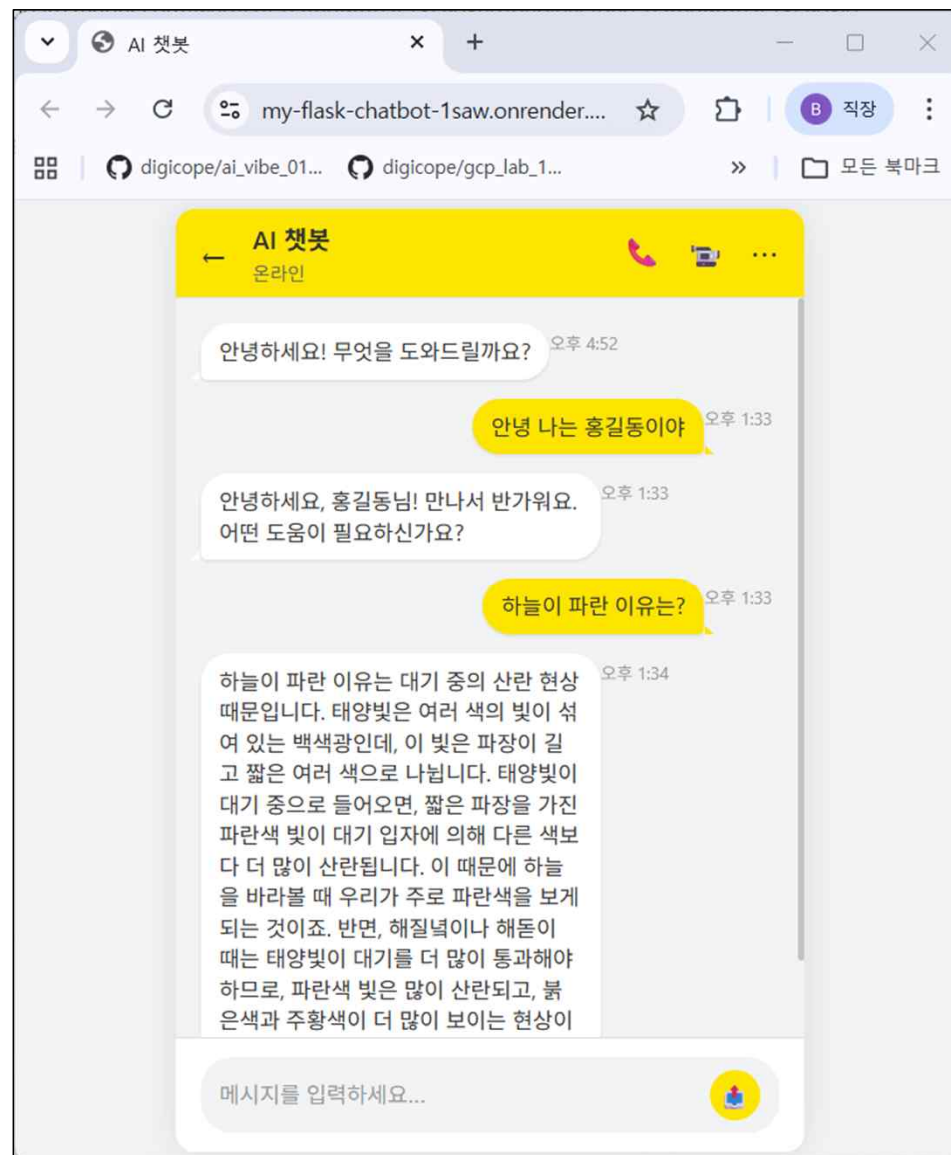


잠시 기다리면 아래와 같이 접속 가능한 URL이 보여진다  
URL을 클릭하면 배포된 챗봇 웹 서버로 접속된다

```
01:33:19 PM      ==> Your service is live 🎉
01:33:19 PM      ==>
01:33:19 PM      ==> //////////////////////////////////////
01:33:20 PM      ==>
01:33:20 PM      ==> Available at your primary URL https://my-flask-chatbot-1saw.onrender.com
01:33:20 PM      ==>
01:33:20 PM      ==> //////////////////////////////////////
01:33:20 PM [wtvmd] 127.0.0.1 - - [26/Jan/2026:04:33:20 +0000] "GET / HTTP/1.1" 200 1778 "-" "Go-http-client/2.0"
```



## 배포 서비스 접속 실행 화면



## 4. Render 에서 배포 서비스 업데이트 및 삭제

---

## 배포된 소스 업데이트 하기

- app.py 소스를 수정하고 Git Bash 아래 명령을 수행하면 자동 배포 업데이트 된다

**git add .**

**git commit -m "update flask chatbot for render"**

**git push origin main**

## 배포된 웹 서비스 삭제하기

render.com → Dashboard

→ Project : My project

→ Production : my\_flask\_chatbot

→ 좌측 메뉴의 Settings 클릭

→ 맨 아래의 [Delete Web Service]  
를 클릭한다

The screenshot displays the Render.com dashboard interface. On the left, a sidebar menu is visible with the following items: Environment, my\_flask\_chatbot, Events, Settings (highlighted in purple), MONITOR (Logs, Metrics), MANAGE (Environment, Shell, Scaling, Previous), Changelog, Invite a friend, and Contact support. The main content area shows the 'Maintenance Mode' configuration page. It includes a toggle switch for 'Maintenance Mode Disabled' (currently off), a section for 'Custom Maintenance Page' with an input field, and a 'PAID' badge indicating that maintenance mode is only available for paid plans. At the bottom, there are two buttons: 'Delete Web Service' (in a red box) and 'Suspend Web Service'.

← Environment

🌐 my\_flask\_chatbot

☰ Events

⚙️ Settings

MONITOR

🔍 Logs

📊 Metrics

MANAGE

📁 Environment

📁 Shell ⚡

📁 Scaling ⚡

📁 Previous

📄 Changelog

👤 Invite a friend

🔗 Contact support

### Maintenance Mode

**Maintenance Mode**

Temporarily disable public access to your service and show a custom maintenance page for all incoming requests. [Learn more](#)

☐ Maintenance Mode Disabled

**Custom Maintenance Page** Optional

If provided, Render uses the specified URL for your custom maintenance page.

⚡ PAID Maintenance mode is only available for paid plans

🗑️ Delete Web Service

⏸️ Suspend Web Service

Git hub의 모든 파일 삭제하기 (삭제후 복구 불가능)

- Git Bash에서 작업 디렉토리에 아래 명령을 수행한다

```
git checkout --orphan main  
git rm -rf .
```

```
echo "# clean repo" > README.md  
git add README.md  
git commit -m "initial commit"  
git push -f origin main
```

(업로드 된 파일이 복잡해서 삭제가 잘 안될 때는 저장소 자체를 삭제하고 다시 만든다)



감사합니다