

# Render로 Streamlit Web Chatbot 배포하기

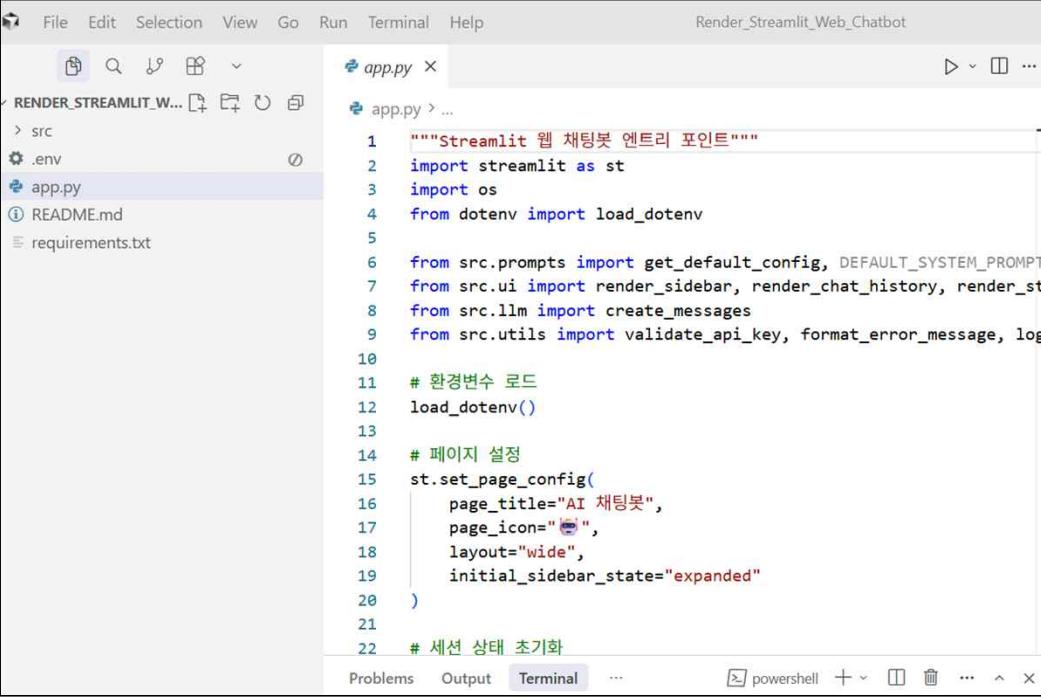


---

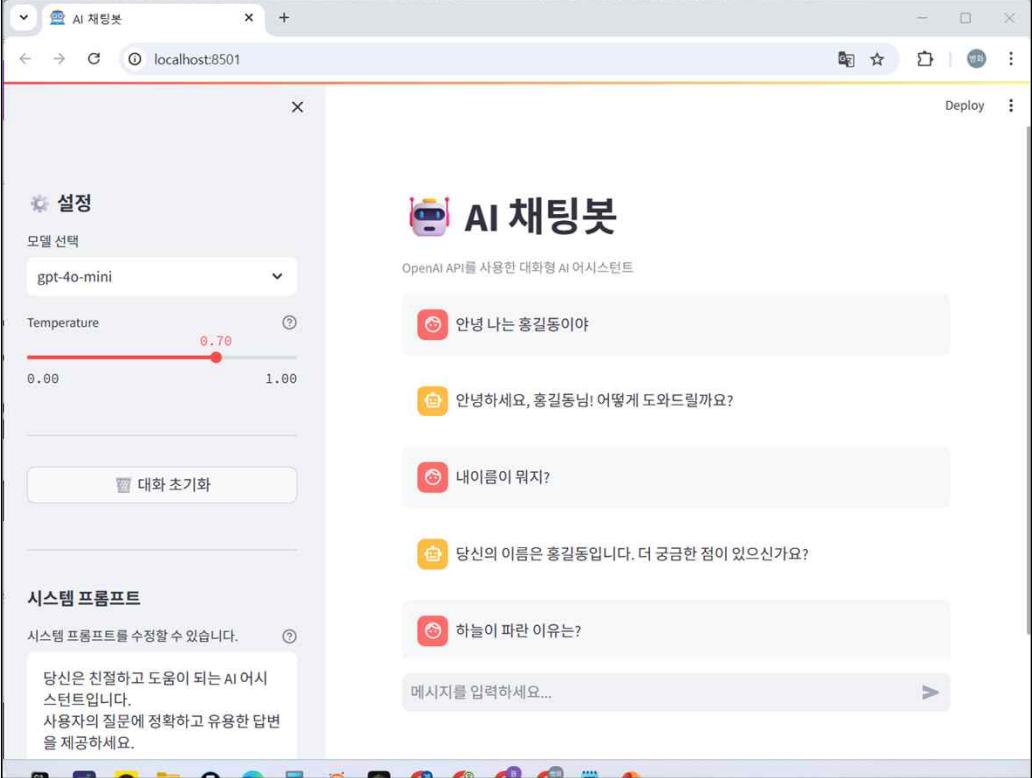
# 1. Streamlit 웹 챗봇 소스 준비

---

Stremlit을 사용한 웹 챗봇 소스를 복사해와서 Render\_Streamlit\_Web\_Chatbot 폴더로 이름을 변경하고 커서를 사용하여 실행시키고 동작을 확인한다



```
File Edit Selection View Go Run Terminal Help
Render_Streamlit_W...
app.py x
app.py > ...
1 """Streamlit 웹 챗봇 엔트리 포인트"""
2 import streamlit as st
3 import os
4 from dotenv import load_dotenv
5
6 from src.prompts import get_default_config, DEFAULT_SYSTEM_PROMPT
7 from src.ui import render_sidebar, render_chat_history, render_st
8 from src.llm import create_messages
9 from src.utils import validate_api_key, format_error_message, log
10
11 # 환경변수 로드
12 load_dotenv()
13
14 # 페이지 설정
15 st.set_page_config(
16     page_title="AI 챗팅봇",
17     page_icon="🤖",
18     layout="wide",
19     initial_sidebar_state="expanded"
20 )
21
22 # 세션 상태 초기화
```



AI 챗팅봇

OpenAI API를 사용한 대화형 AI 어시스턴트

안녕 나는 홍길동이야

안녕하세요, 홍길동님! 어떻게 도와드릴까요?

내이름이 뭐지?

당신의 이름은 홍길동입니다. 더 궁금한 점이 있으신가요?

하늘이 파란 이유는?

메시지를 입력하세요...

### - Requirements.txt 파일 수정

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

streamlit

openai

# python-dotenv

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

"""Streamlit 웹 채팅봇 엔트리 포인트"""

import streamlit as st

import os

# from dotenv import load\_dotenv

from src.prompts import get\_default\_config, DEFAULT\_SYSTEM\_PROMPT, DEFAULT\_MODEL, DEFAULT\_TEMPERATURE

from src.ui import render\_sidebar, render\_chat\_history, render\_streaming\_response

from src.llm import create\_messages

from src.utils import validate\_api\_key, format\_error\_message, logger

# 환경변수 로드

# load\_dotenv()

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

```
"""OpenAI API 호출 및 스트리밍 처리 모듈"""
import os
from typing import List, Dict, Optional, Iterator
from openai import OpenAI
# from dotenv import load_dotenv
import streamlit as st

from src.utils import format_error_message, validate_api_key, logger

# 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/
```

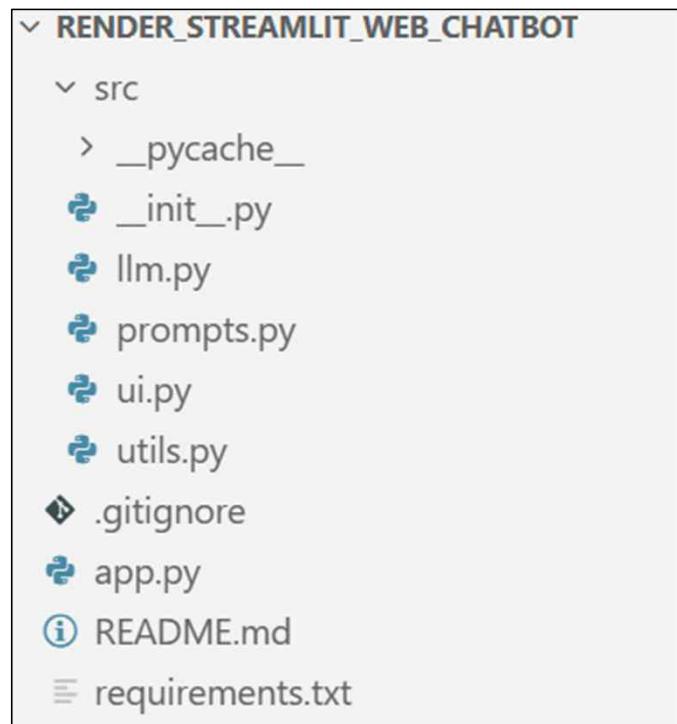
```
.idea/
```

```
# Logs
```

```
*.log
```

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

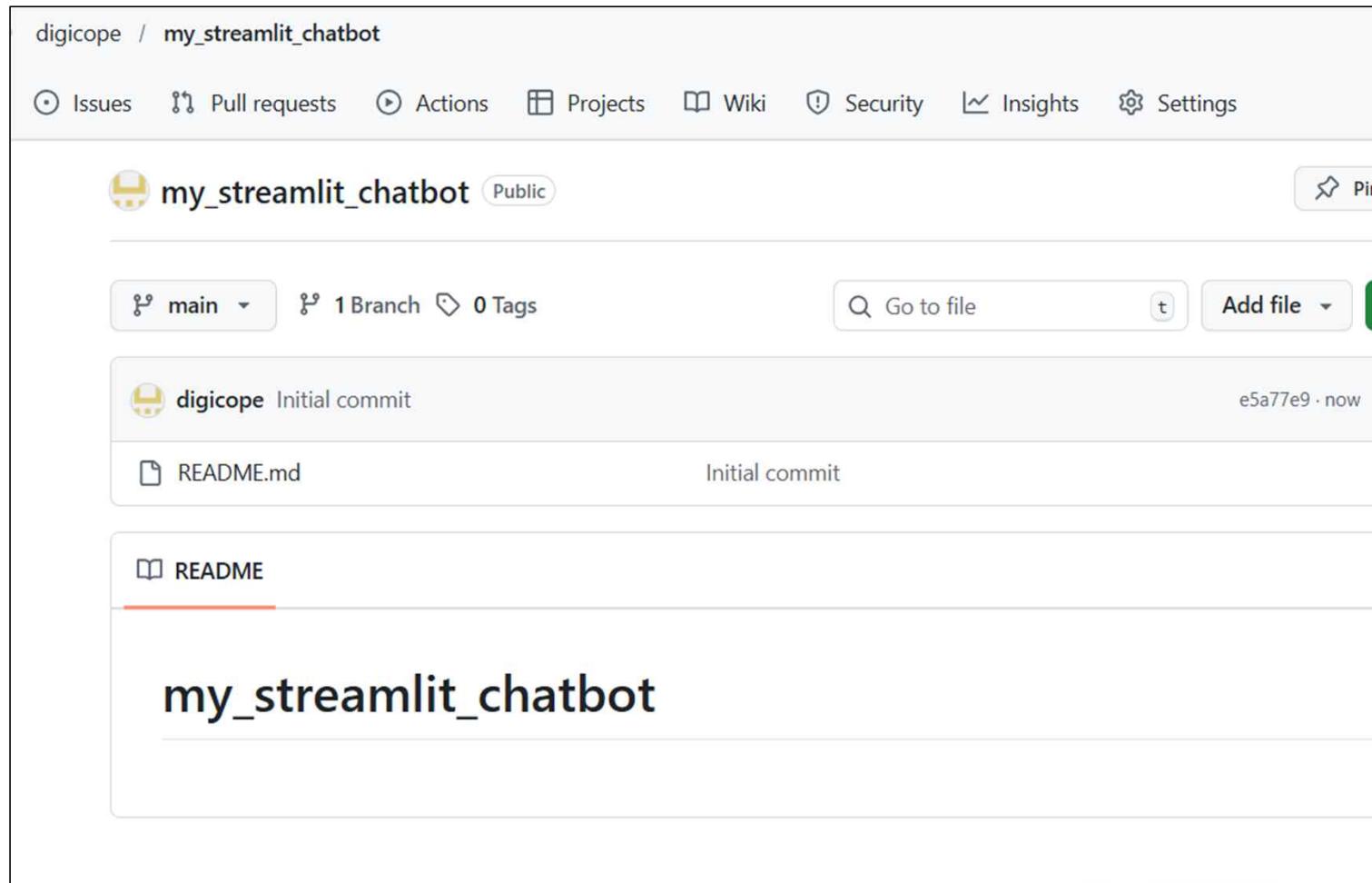
## 변경된 프로젝트 내의 소스 파일 구조



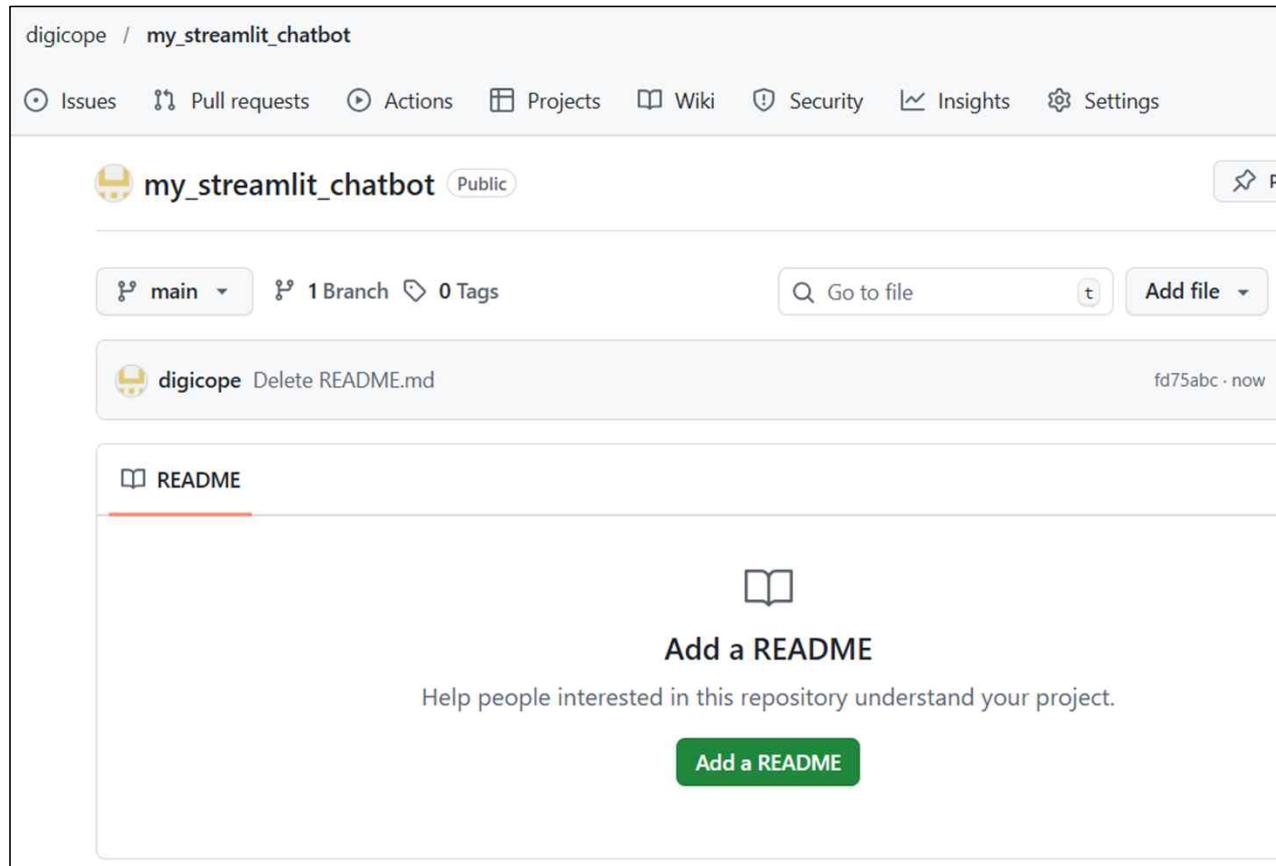
## **2. Git 저장소에 배포 소스 업로드**

---

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

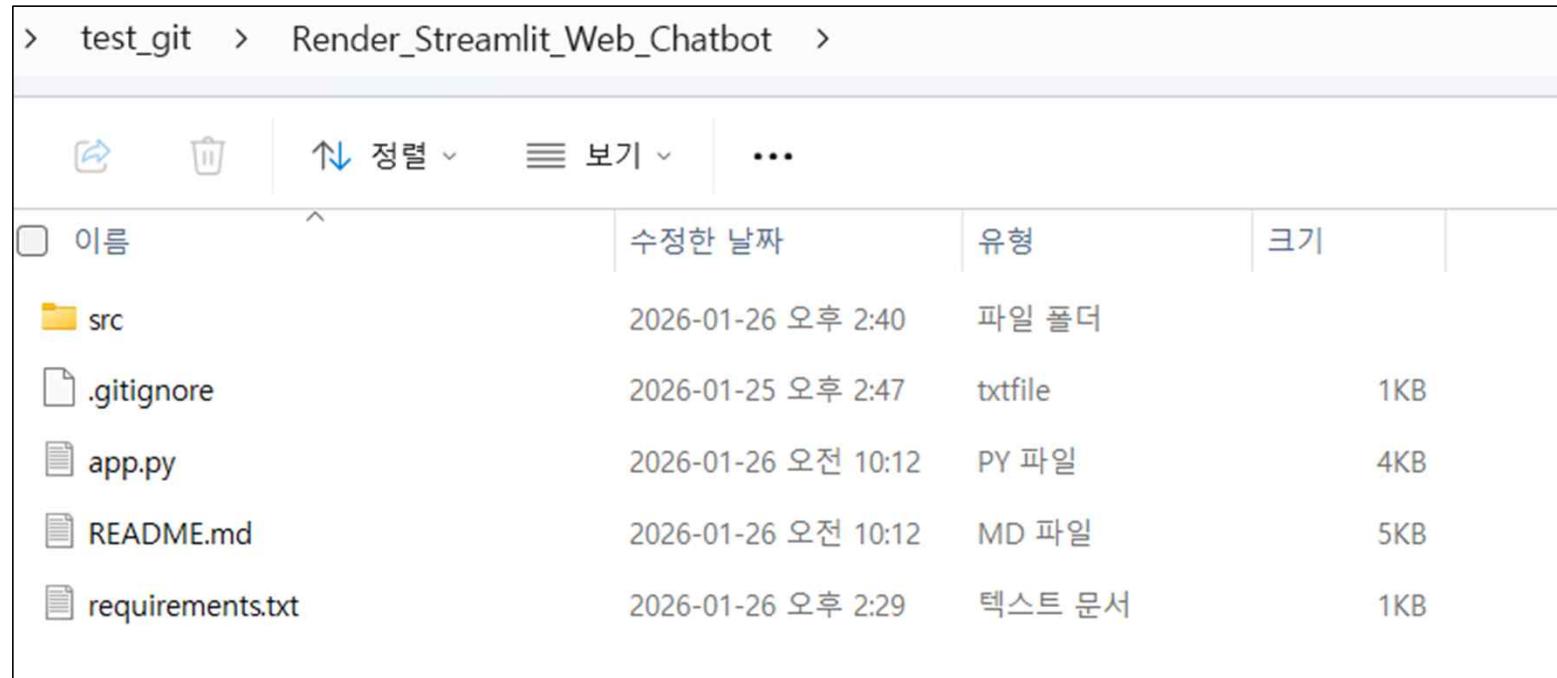


- Git 저장소에 파일이 있으면 저장소로 가서 파일을 모두 삭제해 놓는다  
**(README.md 파일 반드시 삭제)**



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

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



The screenshot shows a Windows File Explorer window with the following file structure:

이름	수정한 날짜	유형	크기
src	2026-01-26 오후 2:40	파일 폴더	
.gitignore	2026-01-25 오후 2:47	txtfile	1KB
app.py	2026-01-26 오전 10:12	PY 파일	4KB
README.md	2026-01-26 오전 10:12	MD 파일	5KB
requirements.txt	2026-01-26 오후 2:29	텍스트 문서	1KB

- Windows에서 Git Bash를 실행하고 아래 경로로 이동한다

**cd test\_git**

- 소스 경로로 이동한다

**cd Render\_Streamlit\_Web\_Chatbot**

- 파일을 확인해 본다

**ls**

```
MINGW64:/c/Users/storm/test_git/Render_Streamlit_Web_Chatbot
storm@HP-ENVY360-PC MINGW64 ~
$ cd test_git

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

storm@HP-ENVY360-PC MINGW64 ~/test_git/Render_Streamlit_web_chatbot
$ ls
README.md  app.py  requirements.txt  src/

storm@HP-ENVY360-PC MINGW64 ~/test_git/Render_Streamlit_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 streamlit chatbot for render"**

**git branch -M main**

**git remote add origin https://github.com/digicope/my\_streamlit\_chatbot.git**

**git pull origin main --rebase**

**git push -u origin main**

## Git 저장소에서 업로드 된 파일들을 확인한다

The screenshot shows a GitHub repository named 'my\_streamlit\_chatbot'. The repository is public and has one branch ('main') and no tags. The commit history shows three commits from a user named 'digicope' with the message 'initial streamlit chatbot for render'. The commits were made at 'aa357da · now' and include 3 commits. The files listed are 'src', '.gitignore', 'README.md', 'app.py', and 'requirements.txt', all with the same commit message and timestamp.

File	Commit Message	Time
src	initial streamlit chatbot for render	now
.gitignore	initial streamlit chatbot for render	now
README.md	initial streamlit chatbot for render	now
app.py	initial streamlit chatbot for render	now
requirements.txt	initial streamlit chatbot for render	now

**README**

**Streamlit 웹 채팅봇**

OpenAI API를 사용한 대화형 AI 채팅봇 웹 애플리케이션입니다.

### **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 it, a title says 'Create a new Cron Job' with a progress bar showing steps 1, 2, and 3. The main area has six service categories: 'Static Sites', 'Web Services', 'Private Services', 'Cron Jobs', 'Postgres', and 'Key Value'. The 'Web Services' category is highlighted with a red box around its 'New Web Service' button. Each category has a brief description and a 'New [Service Name]' button.

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

The image consists of three screenshots of the Render UI, each showing a different step in the process of creating a new service.

- Screenshot 1: Overview**  
Shows the main dashboard with the title "Overview". Under "Projects", there is a card for "My project" which contains the message "✓ All services are up and running". This card is highlighted with a red border.
- Screenshot 2: Project Details**  
Shows the "PROJECT" screen for "My project". It includes sections for "Production" (with tabs for "All (1)", "Services (1)", and "Env Groups (0)"), a search bar ("Search resources in Production"), and a list of services. One service, "my\_test\_repository", is listed. At the bottom, there is a button labeled "+ New service" which is also highlighted with a red border.
- Screenshot 3: Service Selection**  
Shows a dropdown menu with various service options: "Static Site", "Web Service" (which is highlighted with a red border), "Private Service", "Background Worker", "Cron Job", "Postgres", "Key Value", "Env Group", "Move existing services", "Generate Blueprint", and a final "+ New service" button at the bottom, which is also highlighted with a red border.

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

Git Provider Public Git Repository Existing Image

Search

- digicope / my\_streamlit\_chatbot 3m ago
- digicope / ai\_vibe\_0112 24m ago
- digicope / my\_flask\_chatbot 3h ago
- digicope / my\_test\_repository 19h ago
- digicope / gcp\_lab\_1223 5d ago
- digicope / gemini Dec 9, 2025
- digicope / chatbot\_vibecoding\_agent Nov 21, 2025

Source Code

digicope / my\_streamlit\_chatbot • 2m 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** : **streamlit run app.py --server.port \$PORT --server.address 0.0.0.0**  
(파일명이 app.py인 경우)

## Source Code



digicope / my\_streamlit\_chatbot • 4m ago



## Name

A unique name for your web service.

my\_streamlit\_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)

2 existing services

Deploy in a new region +



Start Command 설정에 유의한다

**streamlit run app.py --server.port \$PORT --server.address 0.0.0.0**

**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.

\$ streamlit run app.py --server.port \$PORT --server.address 0.0.0.0

## Instance Type을 Free로 선택한다

### Instance Type

**For hobby projects**

<b>Free</b>	512 MB (RAM)
\$0 / month	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>Standard</b>	2 GB (RAM)
\$7 / month	0.5 CPU	\$25 / month	1 CPU

<b>Pro</b>	4 GB (RAM)	<b>Pro Plus</b>	8 GB (RAM)
\$85 / month	2 CPU	\$175 / month	4 CPU

<b>Pro Max</b>	16 GB (RAM)	<b>Pro Ultra</b>	32 GB (RAM)
\$225 / month	4 CPU	\$450 / month	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**

Value : **sk-projxx**

## Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more.](#)

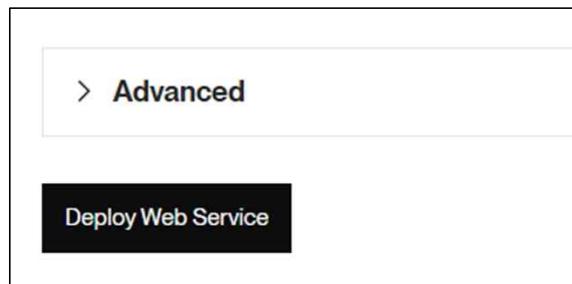
OPENAI\_API\_KEY

sk-proj-HMDVg2KP2e3paFyox5siqt0ezqDn10jFJZpMgK1x3Br8RXM-a3mkvHAdo5E0x1\_90CSKUouuhvWt3B1hkfIa100h7cMLrD2wQAD7V

+ Add Environment Variable

Add from .env

Web Service 배포를 위한 설정이 완료되면 죄측 하단의 [Deploy Web Service]를 누르면 아래와 같이 배포가 진행 된다 (배포 완료 까지 수분 소요)



A screenshot of the Render deployment logs for the service 'my\_streamlit\_chatbot'. The top bar shows the service name, Python version (Python 3), and status (Free). It also includes a link to upgrade the instance. The logs show the deployment process starting at January 26, 2026, at 3:06 PM, with the commit '4dddaa5 initial streamlit chatbot for render'. The logs detail the cloning of the GitHub repository, checking out the commit, installing Python 3.13.4, using Poetry version 2.1.3, and running the 'pip install -r requirements.txt' command. A note in the logs states: 'Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.'

```
January 26, 2026 at 3:06 PM Building
4dddaa5 initial streamlit chatbot for render

All logs ▾ Search
Jan 26
03:06:58 PM => Cloning from https://github.com/digicope/my_streamlit_chatbot
03:06:59 PM => Checking out commit 4dddaa53b5c8aab3a408f8750733a542845bf86a in branch main
03:07:10 PM => Installing Python version 3.13.4...
03:07:20 PM => Using Python version 3.13.4 (default)
03:07:20 PM => Docs on specifying a Python version: https://render.com/docs/python-version
03:07:24 PM => Using Poetry version 2.1.3 (default)
03:07:24 PM => Docs on specifying a Poetry version: https://render.com/docs/poetry-version
03:07:24 PM => Running build command 'pip install -r requirements.txt'...
03:07:24 PM Collecting streamlit (from -r requirements.txt (line 1))
03:07:24 PM     Downloading streamlit-1.53.1-py3-none-any.whl.metadata (10 kB)
```

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

```
03:28:49 PM    ==> Your service is live 🎉  
03:28:49 PM    ==>  
03:28:49 PM    ==> /////////////////////////////////  
03:28:49 PM    ==>  
03:28:49 PM    ==> Available at your primary URL https://my-streamlit-chatbot.onrender.com  
03:28:49 PM    ==>  
03:28:49 PM    ==> /////////////////////////////////  
03:29:42 PM [xj6mg] 2026-01-26 06:29:42,235 - httpx - INFO - HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
```

## 배포 서비스 접속 실행 화면

The screenshot shows a web browser window titled "AI 챗봇" (AI Chatbot) with the URL "my-streamlit-chatbot.onrender.com". The browser toolbar includes icons for back, forward, search, and bookmarks, along with a "작장" (Bookmark) button.

The main content area displays a Streamlit application for an AI chatbot. On the left, there is a sidebar with "설정" (Settings) and "시스템 프롬프트" (System Prompt) sections. The "설정" section contains a dropdown for "모델 선택" (Model Selection) set to "gpt-4o-mini", a "Temperature" slider set to 0.70, and a "대화 초기화" (Reset Conversation) button. The "시스템 프롬프트" section contains a text box with the system prompt: "당신은 친절하고 도움이 되는 AI 어시스턴트입니다. 사용자의 질문에 정확하고 유용한 답변을 제공하세요."

The main right-hand area is titled "AI 챗봇" and describes it as "OpenAI API를 사용한 대화형 AI 어시스턴트". It shows a conversation history:

- Bot: 안녕 나는 홍길동이야
- Bot: 안녕하세요, 홍길동님! 반갑습니다. 어떻게 도와드릴까요?
- Bot: 내 이름이 뭐지?
- Bot: 당신의 이름은 홍길동입니다. 다른 질문이나 도움이 필요하신 부분이 있으면 말씀해 주세요!
- Bot: 하늘이 파란 이유는?
- Bot: 하늘이 파란 이유는 대기 중의 산란 현상 때문입니다. 태양빛은 여러 색의 빛으로 이루어져 있으며, 이 중 파란색 빛은 다른 색에 비해 파장이 짧습니다. 대기 중의 분자와 미세한 입자들이 태양빛을 산란시키는 과정에서 파란색 빛이 더 많이 산란되어 하늘이 파랗게 보이게 됩니다. 이 현상을 레이어리 사라지라고 합니다. 바며 천연기념물 이으며 찾느라 지하

## **4. Render에서 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 streamlit chatbot for render"**

**git push origin main**

## 배포된 웹 서비스 삭제하기

render.com → Dashboard  
→ Project : My project  
→ Production : my\_streamlit\_chatbot  
→ 좌측 메뉴의 Settings 클릭  
→ 맨 아래의 [Delete Web Service]를 클릭한다

The screenshot shows the render.com dashboard for the project "my\_streamlit\_chatbot". On the left, a sidebar lists various management options: Environment, Events, Settings (which is highlighted in purple), MONITOR, Logs, Metrics, MANAGE, Environment, Shell (with a lightning bolt icon), Scaling (with a lightning bolt icon), Previews, Changelog, Invite a friend, and Contact support. On the right, the "Maintenance Mode" section is displayed. It includes a toggle switch labeled "Maintenance Mode D", a description explaining that it temporarily disables public access while serving a static maintenance page, and a link to "Learn more". Below this, there's a section for a "Custom Maintenance Page" which is marked as optional and notes that Render uses a specified URL instead of the default page if provided. A note at the bottom states that maintenance mode is only available for paid instances. At the very bottom of the dashboard, there are two red buttons: "Delete Web Service" and "Suspend Web Service".

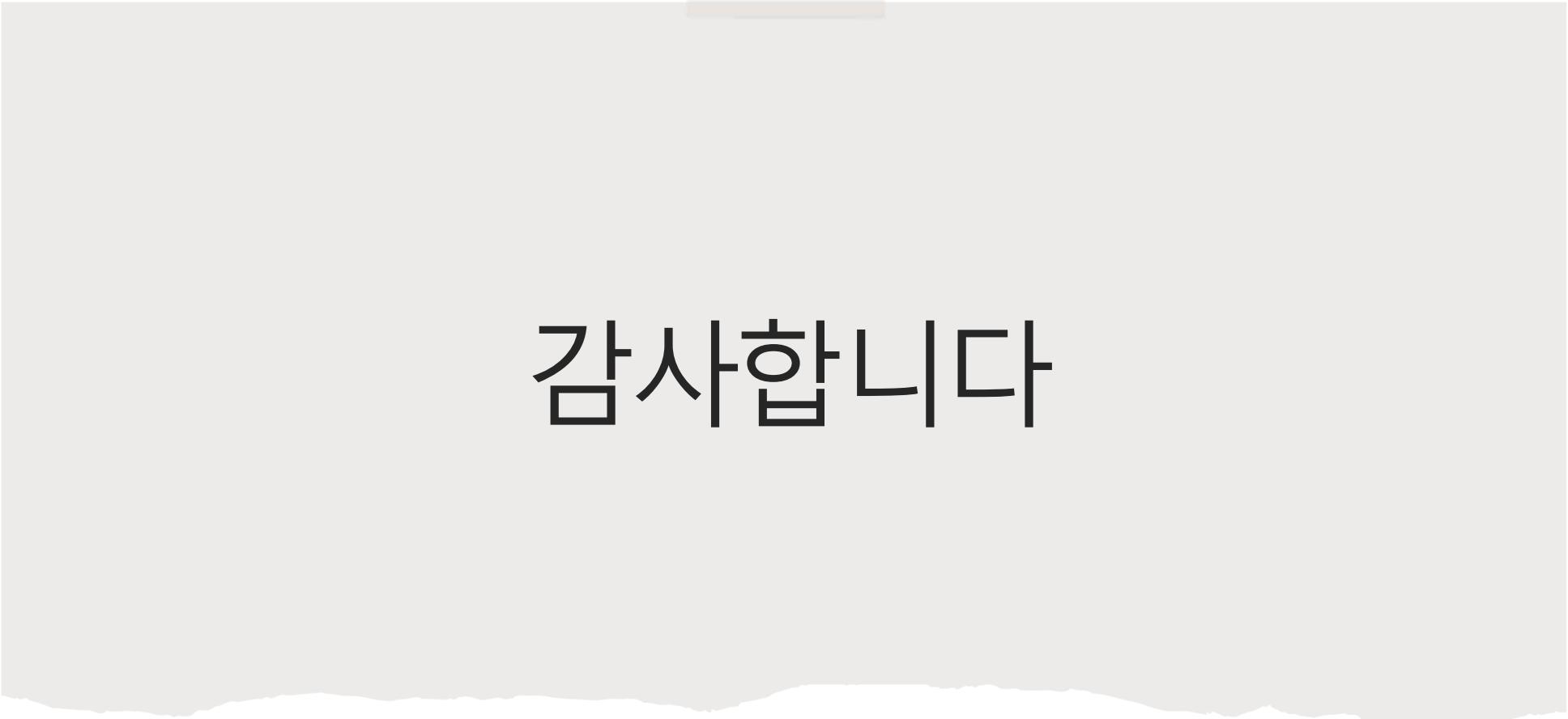
## Git hub의 모든 파일 삭제하기 (삭제후 복구 불가능)

- Git Bash에서 작업 디렉토리에서 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