웹 애플리케이션 배포 방법

1. 웹 애플리케이션 배포 서비스 종류

1. 서비스 종류

[1] Render

범용 웹 애플리케이션을 무료로 배포할 수 있는 PaaS 클라우드이다.

Flask, FastAPI, Streamlit, PHP 등 다양한 웹 프레임워크를 하나의 플랫폼에서 배포할 수 있다.

[2] Railway

개발자 친화적인 설정과 자동 배포 기능을 제공하는 PaaS 클라우드이다.

초기 무료 크레딧은 제공되지만, 지속적인 무료 사용에는 제한이 있다 (유료서비스)

[3] Streamlit Cloud

Streamlit 전용 애플리케이션 배포에 특화된 PaaS 클라우드이다.

서버 설정 없이 Streamlit 앱을 빠르게 배포하는 데 적합하다.

[4] Hugging Face Spaces

AI 데모와 머신러닝 모델 시연용 애플리케이션 배포에 특화된 PaaS 클라우드이다.

Streamlit과 Gradio 기반의 AI 데모를 무료로 배포할 수 있다.

[5] 일반 Cloud (GCP / AWS / Azure)

가상 머신과 네트워크를 직접 구성·관리하는 IaaS 클라우드이다.

높은 자유도와 확장성을 제공하지만, 비용과 운영 관리 부담이 크다.

서비스별 비교

구분	Render	Railway	Streamlit Cloud	Hugging Face	일반 Cloud
서비스 유형	PaaS	PaaS	PaaS	PaaS	IaaS
무료 지속 사용	가능	어려움	가능	가능	불가
신용카드 필요	없음	필요	없음	없음	필요
Streamlit	가능	가능	가능	가능	가능
Flask / FastAPI	가능	가능	불가	비권장	가능
PHP	가능	가능	불가	불가	가능
난이도	중	중상	하	하	상
교육·실습 적합	높음	보통	높음	높음	낮음
실서비스 운영	제한적	가능	부적합	부적합	적합

2. Render 서비스 소개 및 가입하기

Render는 **웹 애플리케이션과 백엔드 서비스를 쉽게 배포·운영할 수 있는 클라우드 플랫폼**이다.

Render 개요

Render는 서버 설정이나 인프라 관리 없이도
GitHub에 있는 코드를 바로 실행 환경에 배포할 수 있도록 제공하는
PaaS(Platform as a Service) 서비스이다.

Render의 주요 특징

- GitHub 저장소 연동 기반 배포
- Flask, FastAPI, Django, Node.js 등 지원
- 빌드·실행 자동화 (CI/CD 내장)
- HTTPS 기본 제공
- 환경 변수 관리 기능 제공
- 무료(Hobby) 플랜 제공

Render는 GitHub 코드만 있으면 서버 운영 없이 웹 서비스를 배포할 수 있는 클라우드 플랫폼이다.

Render의 장점

- 서버 생성, 방화벽, SSL 설정이 필요 없다
- 코드 push만으로 자동 재배포가 가능하다
- 학습·실습·프로토타입 제작에 적합하다
- 소규모 상용 서비스까지 확장 가능하다

Render의 단점 및 제약

- 무료 플랜은 일정 시간 무접속 시 Sleep 발생
- 고트래픽 서비스에는 비용이 증가한다
- 인프라 세부 제어는 IaaS보다 제한적이다

Render의 활용 예

- Flask / FastAPI 웹 서버 배포
- REST API 서버 운영
- 개인 포트폴리오 웹사이트
- 강의·실습용 데모 서버

Render 회원 가입 방식

Render는 **비회원 사용이 불가능**하고, 아래 계정 중 하나로 가입해야 합니다.

- GitHub 계정
- GitLab 계정
- 이메일 계정

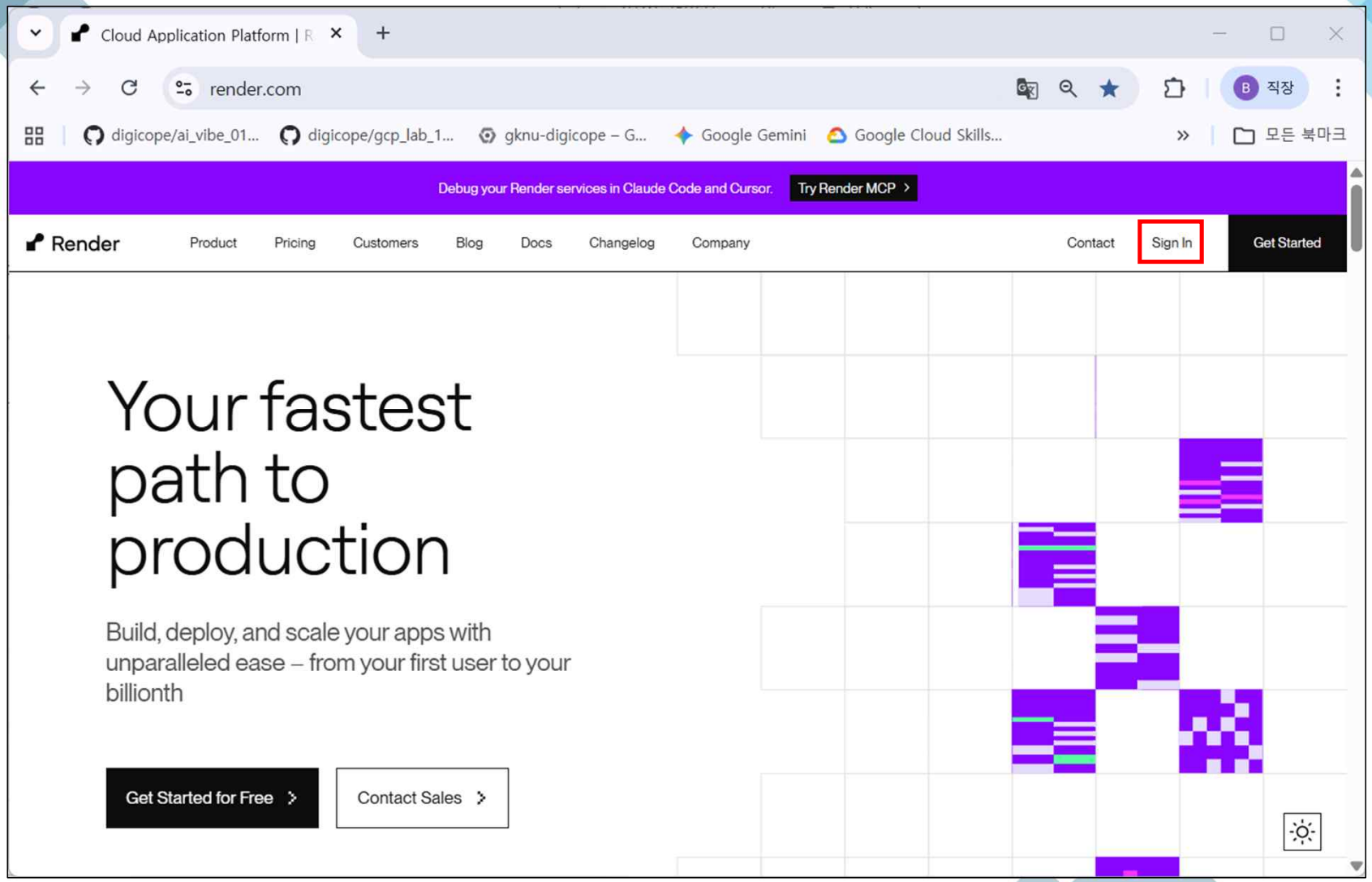
실무·실습에서는 **GitHub 계정으로 가입하는 방식이 가장 일반적**이다.
이유는 GitHub 저장소와 바로 연동해서 배포할 수 있기 때문이다.

왜 회원 가입이 필요한가

Render는 다음 기능을 제공하기 때문에 계정이 필요하다.

- GitHub 레포지터리 연결 및 자동 배포
- 배포 이력 관리
- 환경 변수(SECRET_KEY, API KEY 등) 관리
- 서비스 상태 모니터링 및 로그 확인
- 무료/유료 플랜 관리

https://render.com/



Hobby

For personal projects and small-scale applications.

$0 USD

per user/month plus compute costs*

Start deploying

- Deploy full-stack apps in minutes
- Fully-managed datastores
- Custom domains
- Global CDN & regional hosting
- Get security out of the box
- Email support

Professional

For teams building production applications.

$19 USD

per user/month plus compute costs*

Select plan

Everything in Hobby, plus:

- 500 GB of bandwidth included
- Collaborate with 10 team members
- Unlimited projects & environments
- Horizontal autoscaling
- Test with preview environments
- Private link connections
- Isolated environments
- Chat support

Organization

For teams with higher traffic demands and compliance needs.

$29 USD

per user/month plus compute costs*

Select plan

Everything in Professional, plus:

- 1 TB of bandwidth included
- Unlimited team members
- Audit logs
- SOC 2 Type II certificate
- ISO 27001 certificate

Enterprise

For enterprises with critical security, performance and support needs.

Custom pricing

Get in touch

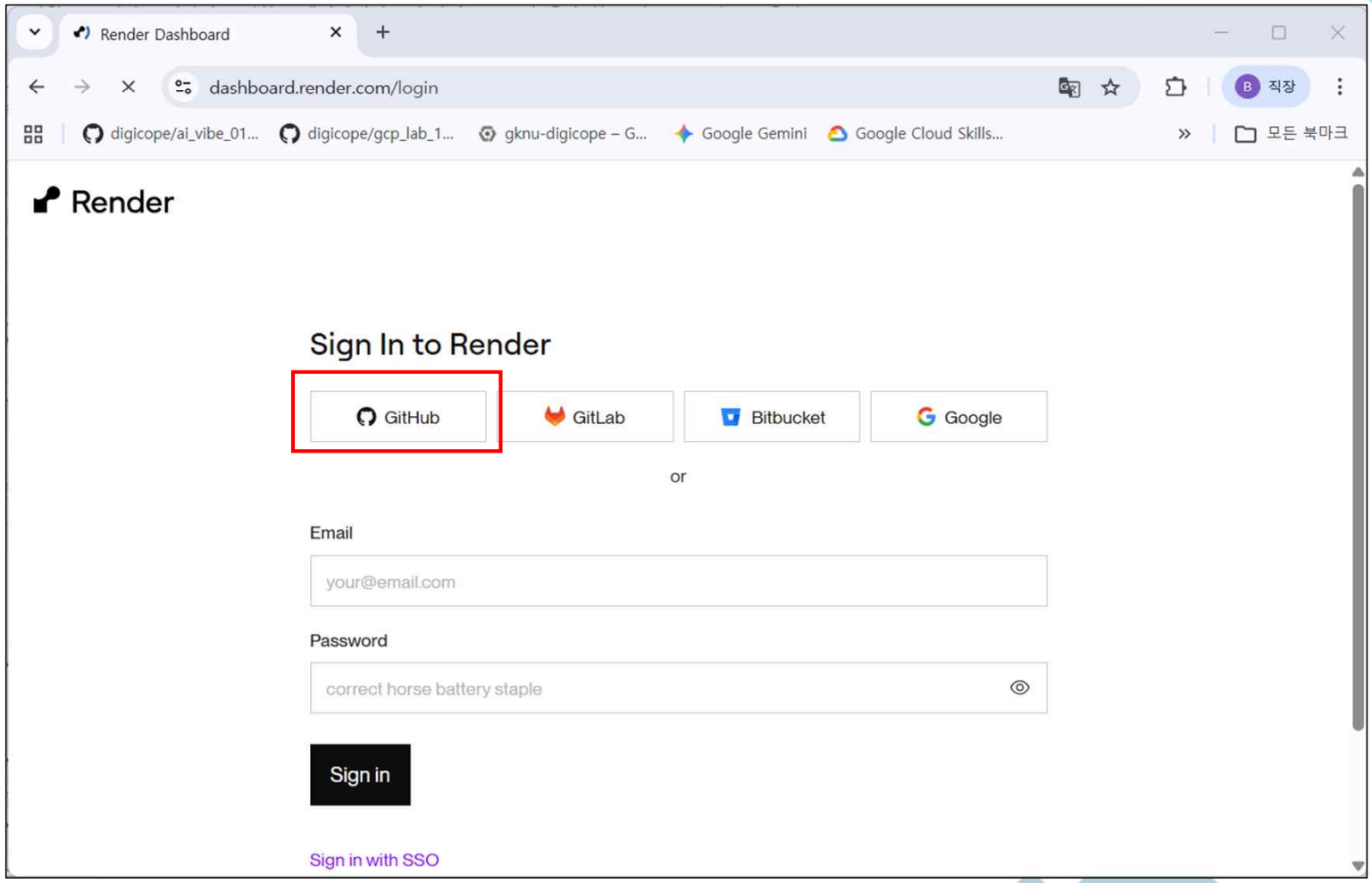
Everything in Organization, plus:

- Centralized team management
- Guest users
- SAML SSO & SCIM
- Guaranteed uptime
- Premium support
- Customer success

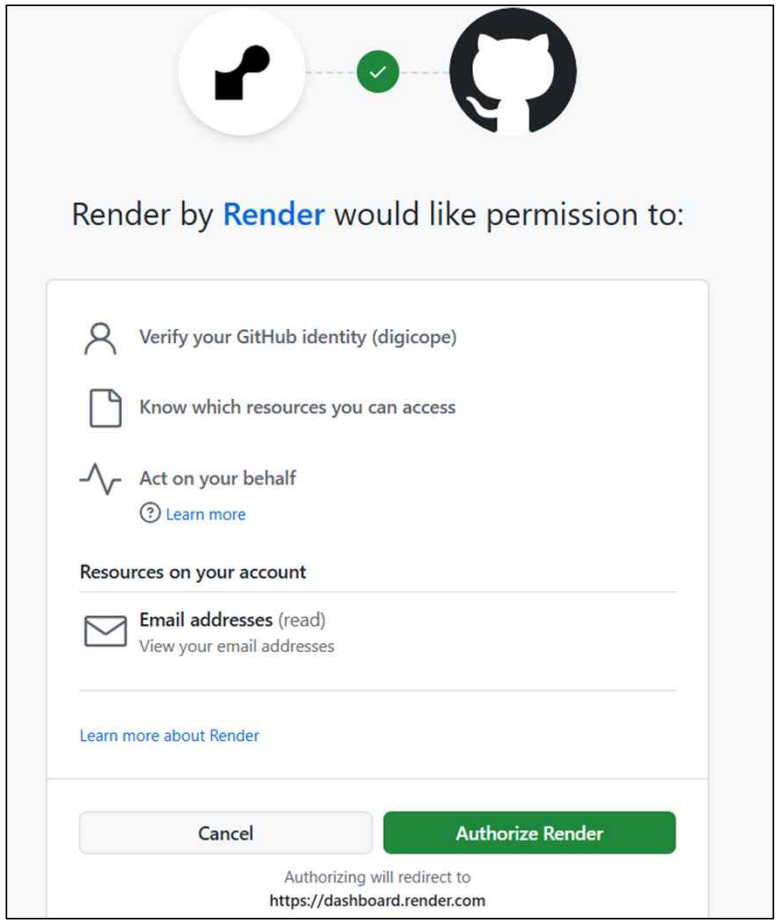
View our compute pricing* and FAQs

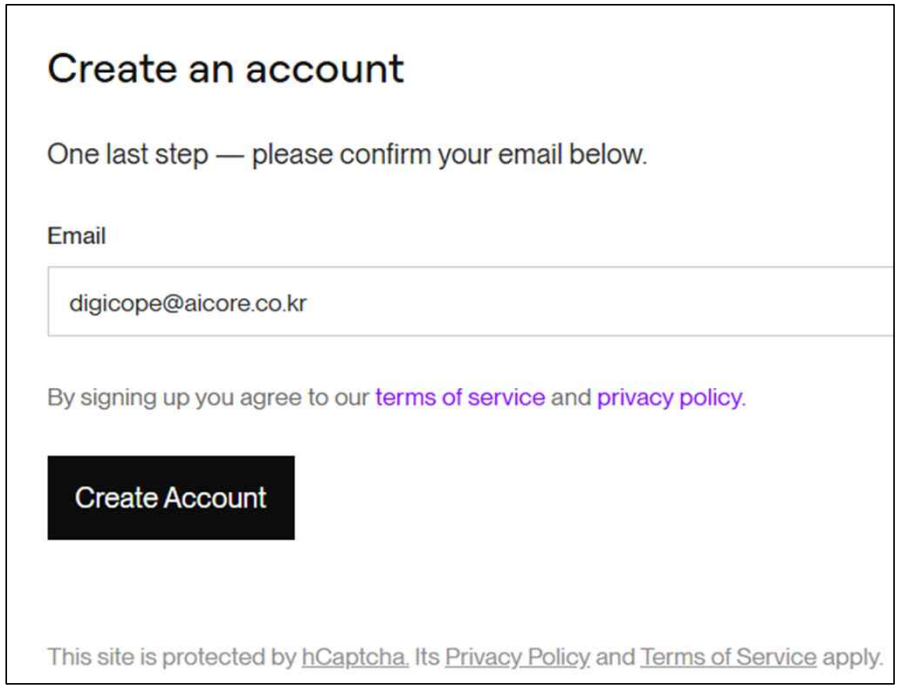
*Compute costs: Pay only for provisioned resources, with transparent pricing based on CPU and service activation prorated to the second.

Render 홈페이지 우측 상단의 [Sign In]을 누르고 아래 화면에서 [GitHub]를 선택하여 회원 가입 한다.(회원 가입 전에 미리 웹 브라우저에서 GitHub 계정에 로그인해 놓는다)

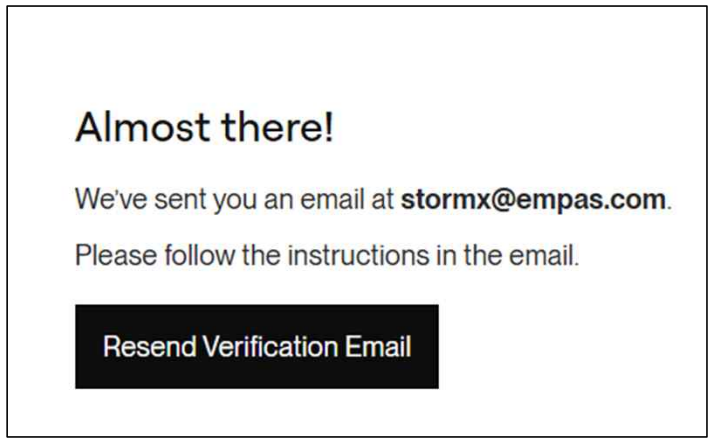


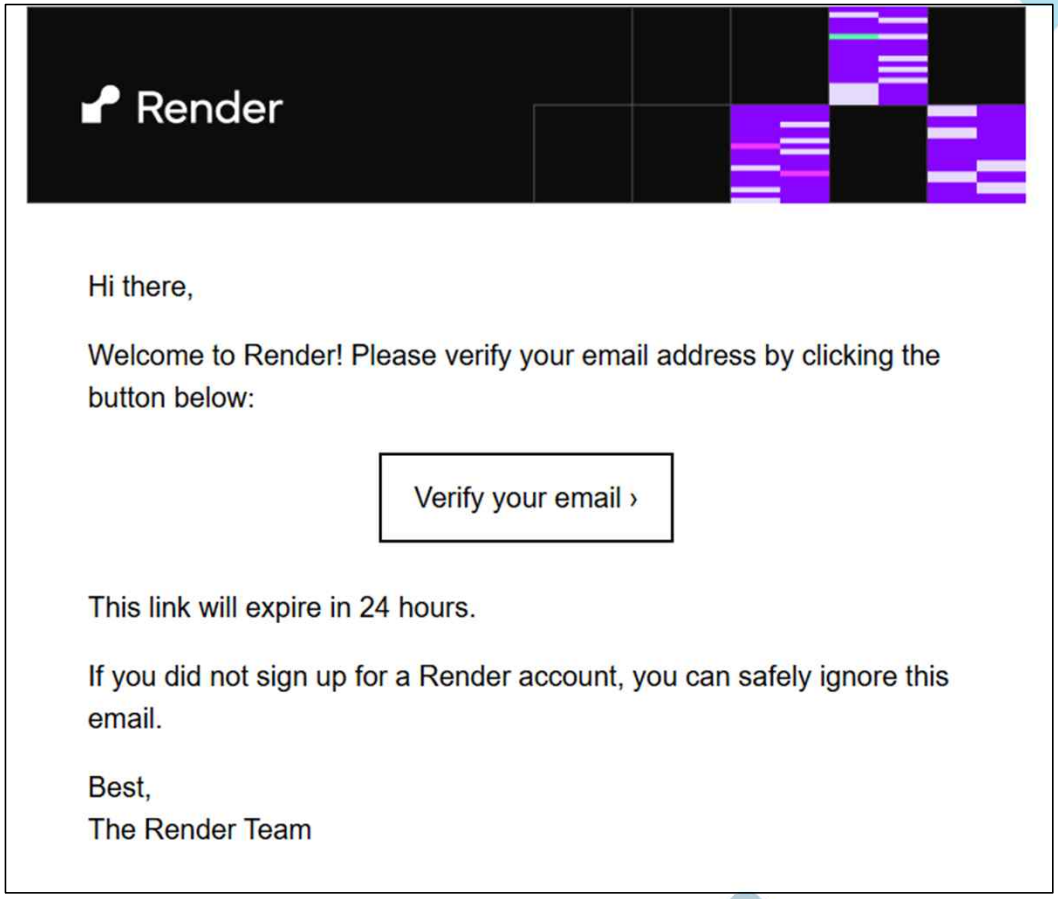
[Authorize Render]을 누르고 Email을 확인하고 [Create Account]를 누른다



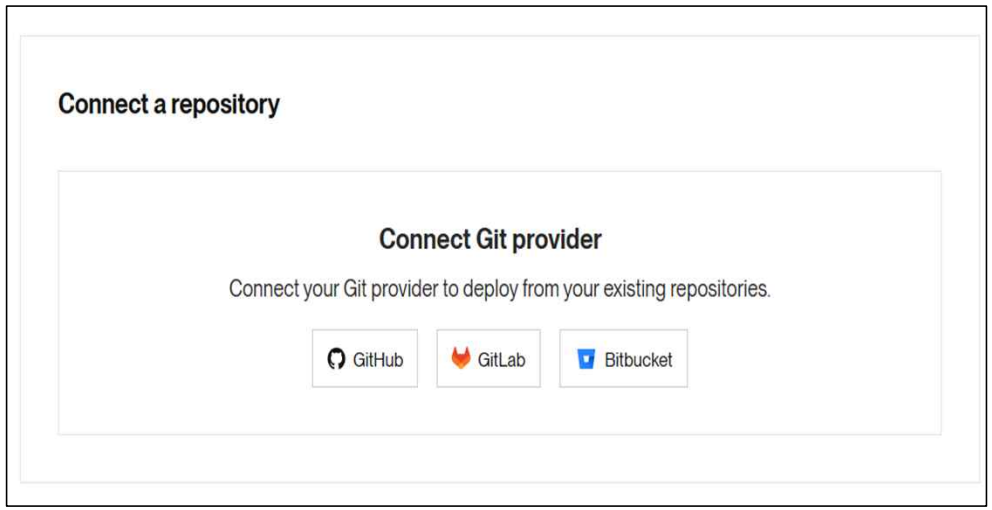


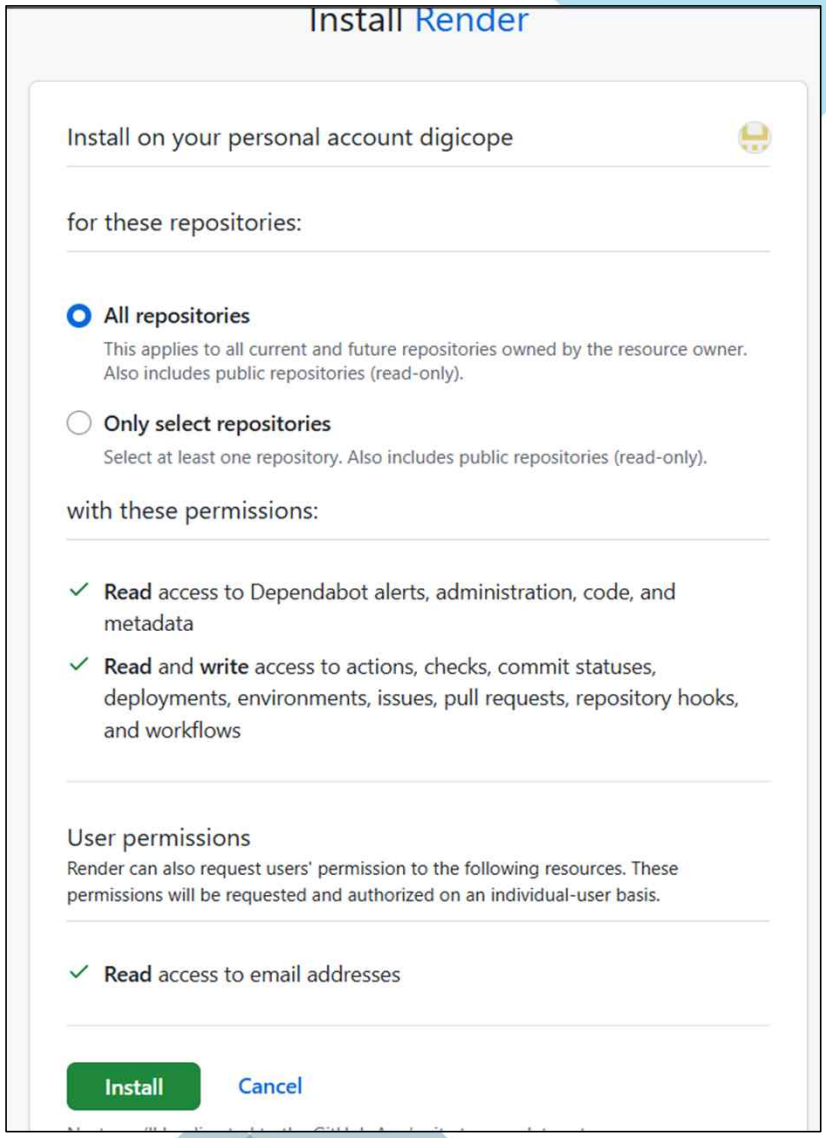
수신된 Email 을 확인하고 아래와 같이 [Verify your email]를 누른다





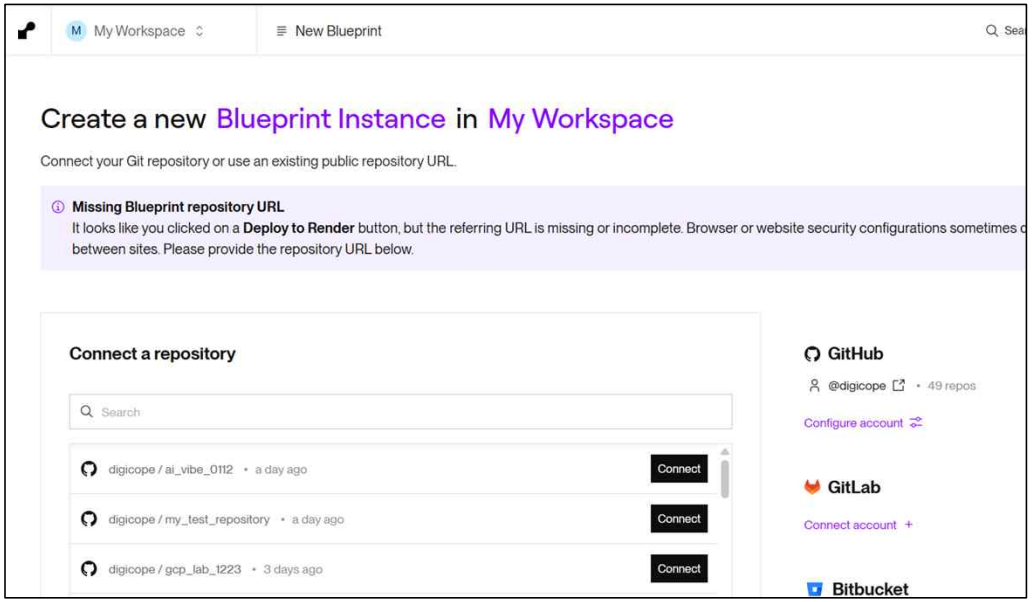
Connect Git provider 에서 [GitHub] 누른 다음
오른쪽 창이 나오면 All repository를 선택한 상태에서
하단의 [Install] 버튼을 클릭한다





Github 계정 비밀번호 입력 창이 나오면 입력하고 계속 진행한다

아래 Create a new Blueprint Instance in My Workspace 화면에서 좌측 상단의 ( ) 부분을 눌러서 Render 대시보드로 이동한다



Web Service

- **하나의 웹 애플리케이션을 직접 배포하는 방식**
- Render 화면에서 설정값을 입력해서 배포
- Flask, FastAPI, Streamlit 실습에 가장 적합

Blueprint

- **render.yaml 파일로 여러 서비스를 한 번에 배포하는 방식**
- 인프라 구성을 코드로 관리(IaC 개념)
- 운영·팀 환경에 적합

Blueprint는 언제 쓰는가?

다음 경우에만 사용한다.

- Web Service + DB를 한 번에 배포
- Infra를 코드로 관리(IaC 개념)
- 팀/운영 환경

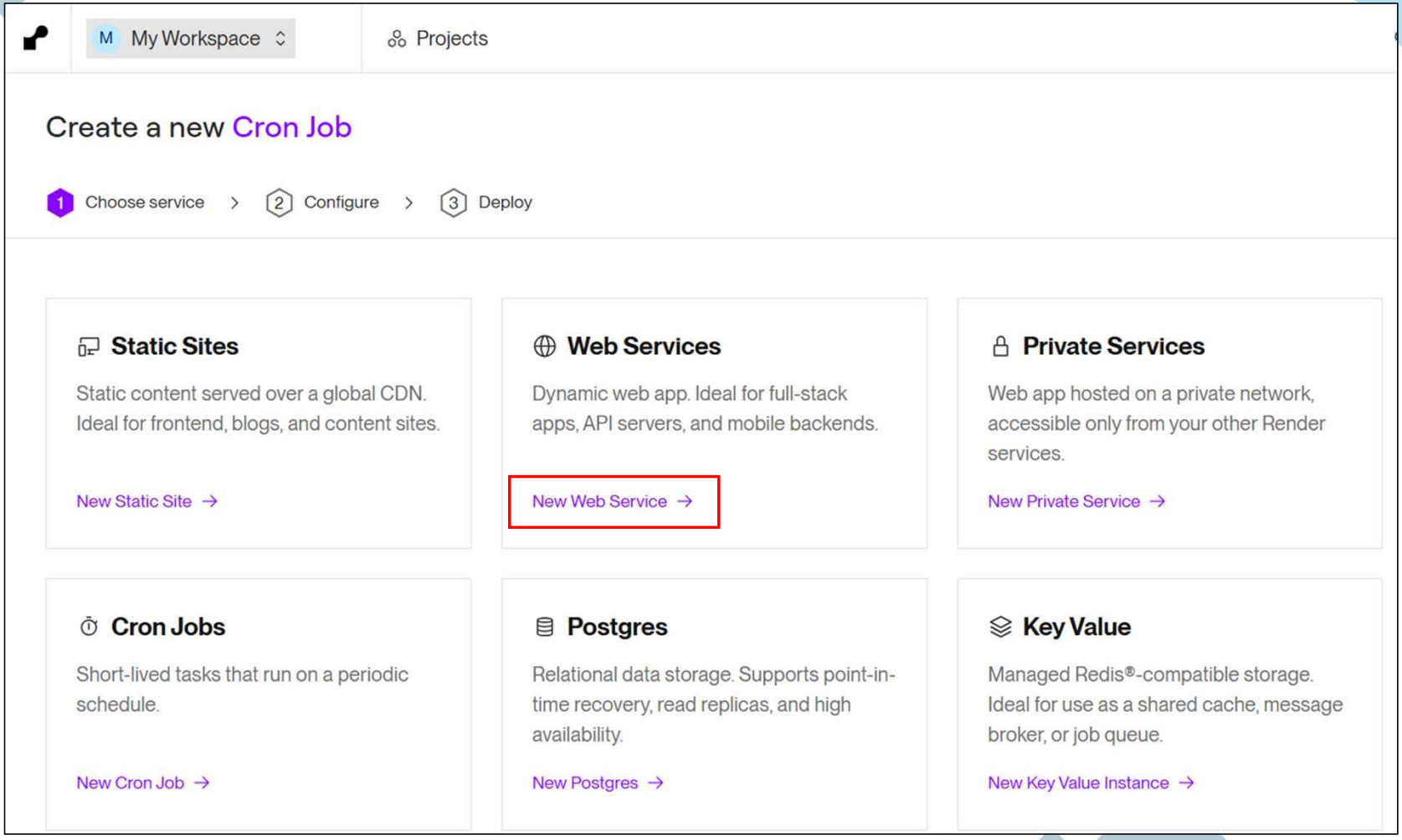
예시 render.yaml이 있을 때

```yaml
services:
  - type: web
    name: flask-app
    env: python
    buildCommand: pip install -r requirements.txt
    startCommand: gunicorn app:app
```

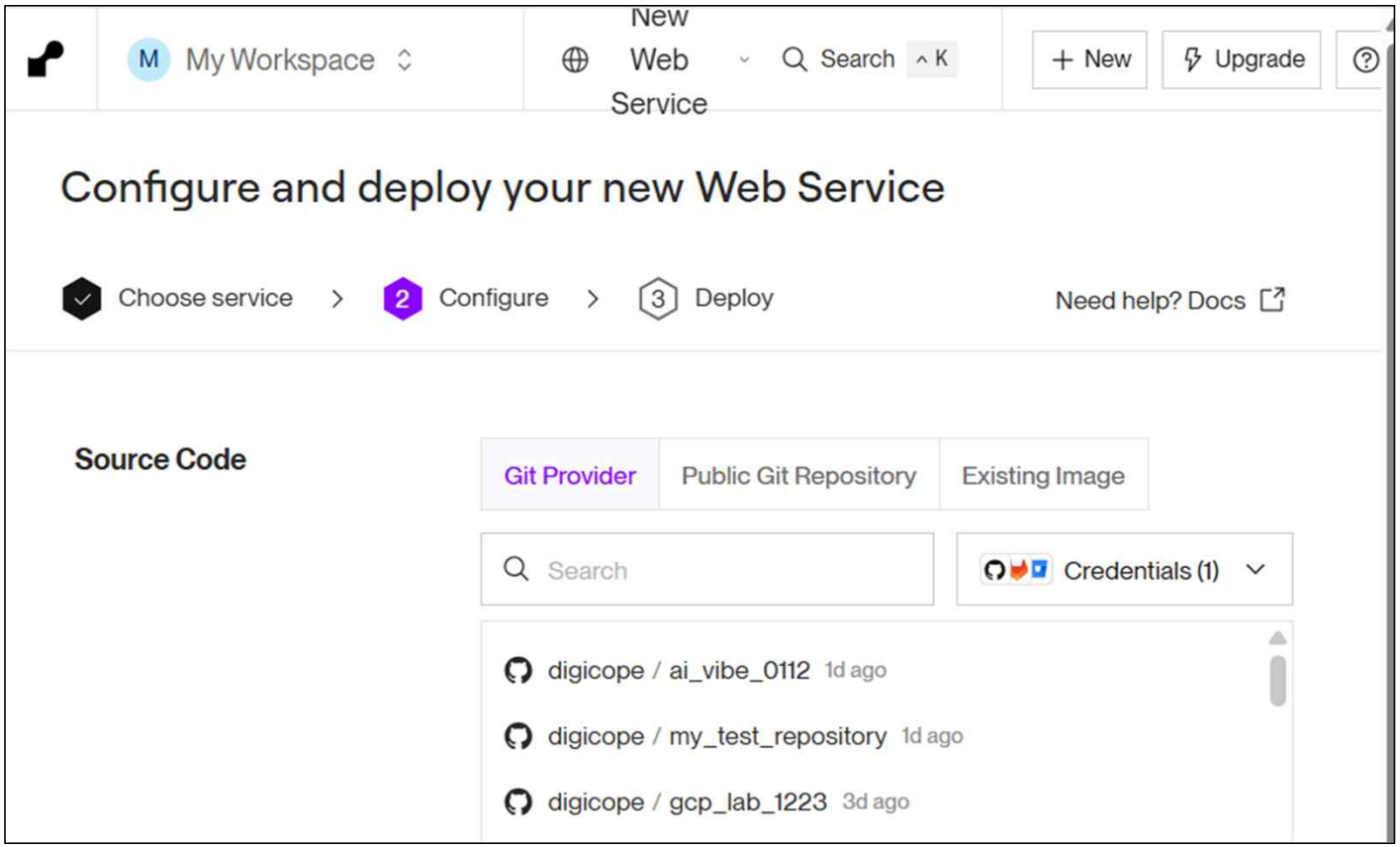
Flask 웹 서버 배포라면

Blueprint가 아니라 **일반 Web Service 방식**을 써야 한다.

아래 대시보드화면에서 **Web Services**의 **New Web Service** 를 누른다



Configure and deploy your new Web Service에서 배포할 서비스에 맞게 설정하고 배포할 수 있다



2. Render 사용 Flask 웹 앱 배포

Flask 사용 웹 서버 소스와 배포에 필요한 파일을 준비하여 Git에 업로드해 놓는다

프로젝트 소스 구조

```
my_flask_render_app/
 ├─ app.py
 ├─ requirements.txt
 ├─ .gitignore
 └─ README.md
```

app.py 소스

```
import os
from flask import Flask, jsonify

app = Flask(__name__)

@app.get("/")
def home():
    return """
    <h2>Hello Render + Flask</h2>
    <p>이 페이지가 보이면 배포가 정상이다.</p>
    <p><a href="/health">/health</a> 로 헬스체크 확인 가능하다.</p>
    """

@app.get("/health")
def health():
    return jsonify(status="ok")

if __name__ == "__main__":
    # Render는 PORT 환경변수로 포트를 전달한다.
    port = int(os.environ.get("PORT", "5000"))
    # 외부 접속을 위해 0.0.0.0 바인딩이 필요하다.
    app.run(host="0.0.0.0", port=port)
```

requirements.txt

```
Flask==3.0.3
gunicorn==22.0.0
```

.gitignore (Git이 추적하지 말아야 할 파일/폴더를 지정하는 설정 파일 이다)

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

# Virtual env
.venv/
venv/

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

# Logs
*.log
```

README.md

Flask on Render 실습용

1) 로컬 실행
가상환경 생성 후 설치한다.

- Windows (PowerShell)
  - python -m venv .venv
  - .₩.venv₩Scripts₩activate
  - pip install -r requirements.txt
  - python app.py

브라우저에서 아래 주소로 확인한다.
- http://127.0.0.1:5000/
- http://127.0.0.1:5000/health

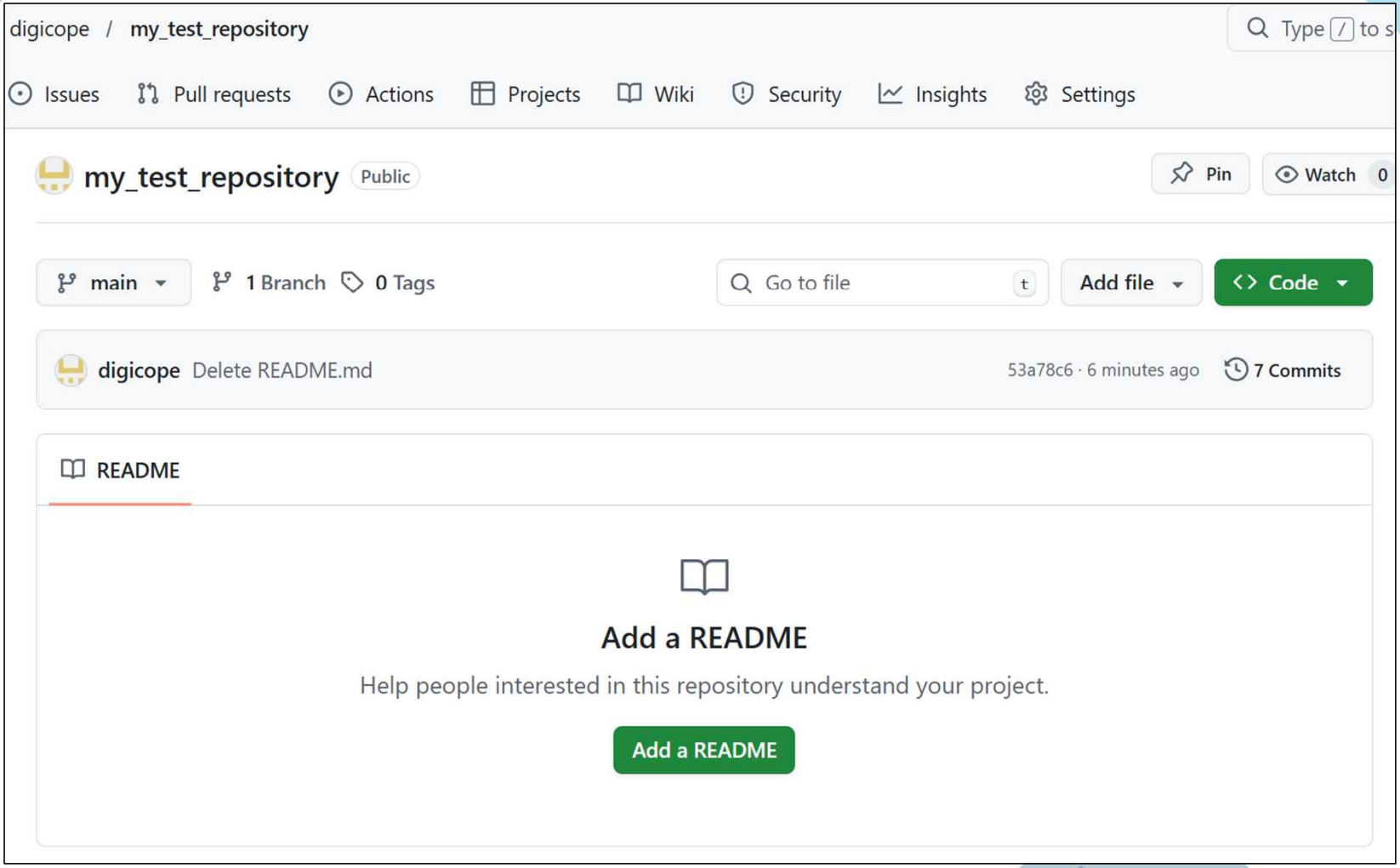
2) Render 배포(Web Service)
Render에서 New -> Web Service -> GitHub 저장소 선택 후 아래처럼 설정한다.

- Build Command
  - pip install -r requirements.txt

- Start Command
  - gunicorn app:app

배포 후 Render가 제공하는 URL로 접속해서 확인한다.

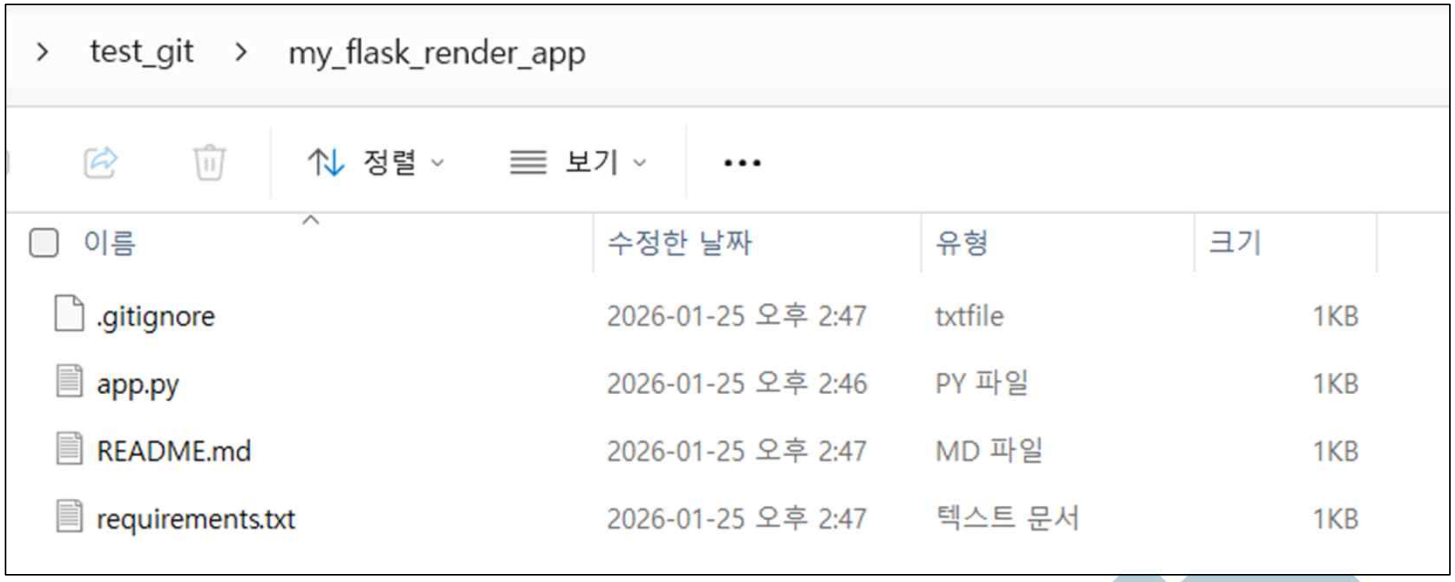
사용할 Github 저장소로 가서 파일을 모두 삭제해 놓는다



Windows에서 Git Bash를 실행하고 아래 경로로 이동한다
cd test_git

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

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

윈도우 탐색기에서 위 test_git 폴더 아래 배포할 앱의 압축 파일을 풀어 놓는다
(강사 배포 파일 : **my_flask_render_app.zip**)



- 소스 디렉토리로 경로를 이동한다

cd my_flask_render_ap/

- 아래 명령을 차례로 수행한다
(user name과 email, 저장소 경로는 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 app for render"

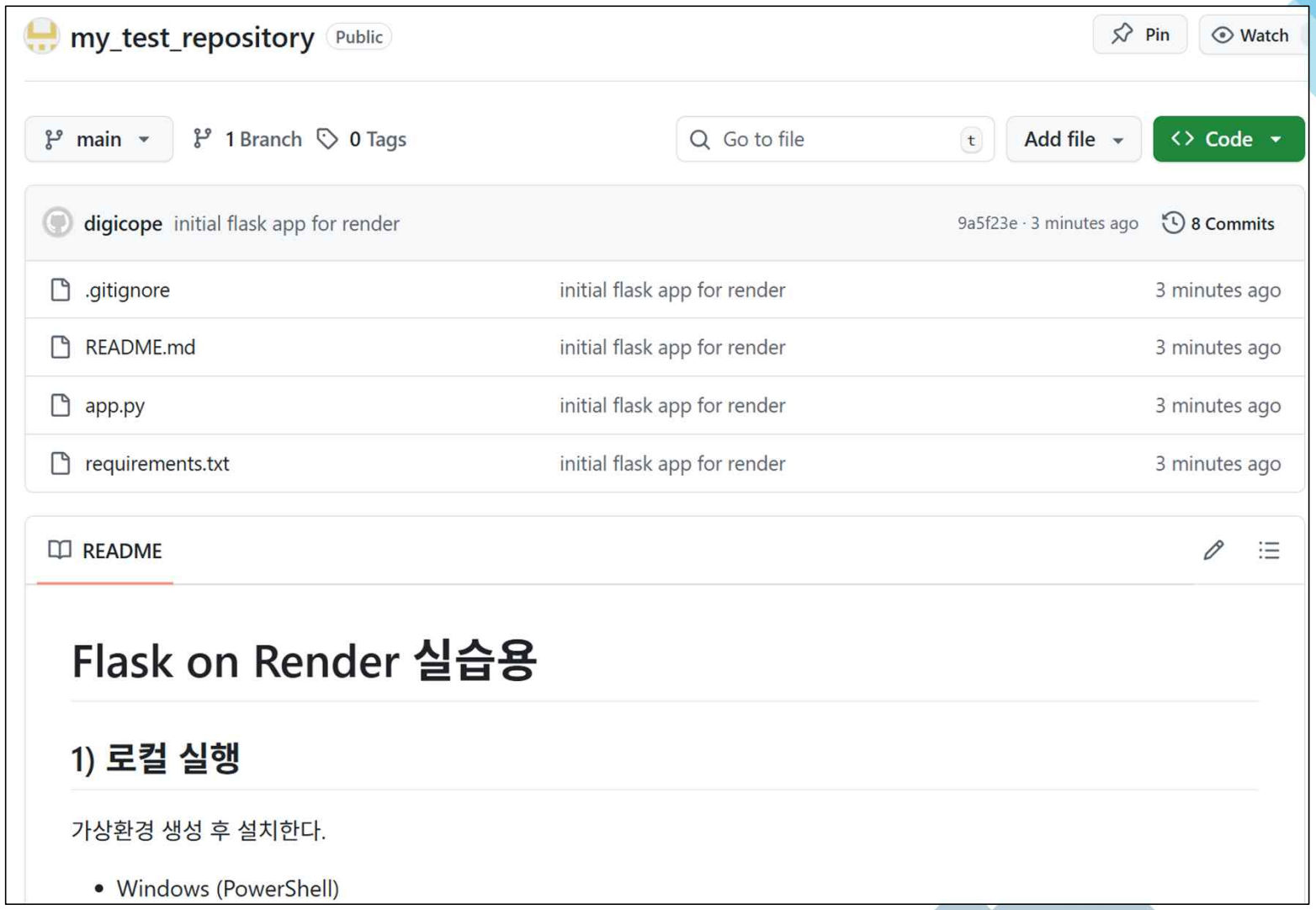
git branch -M main

git remote add origin https://github.com/digicope/my_test_repository.git

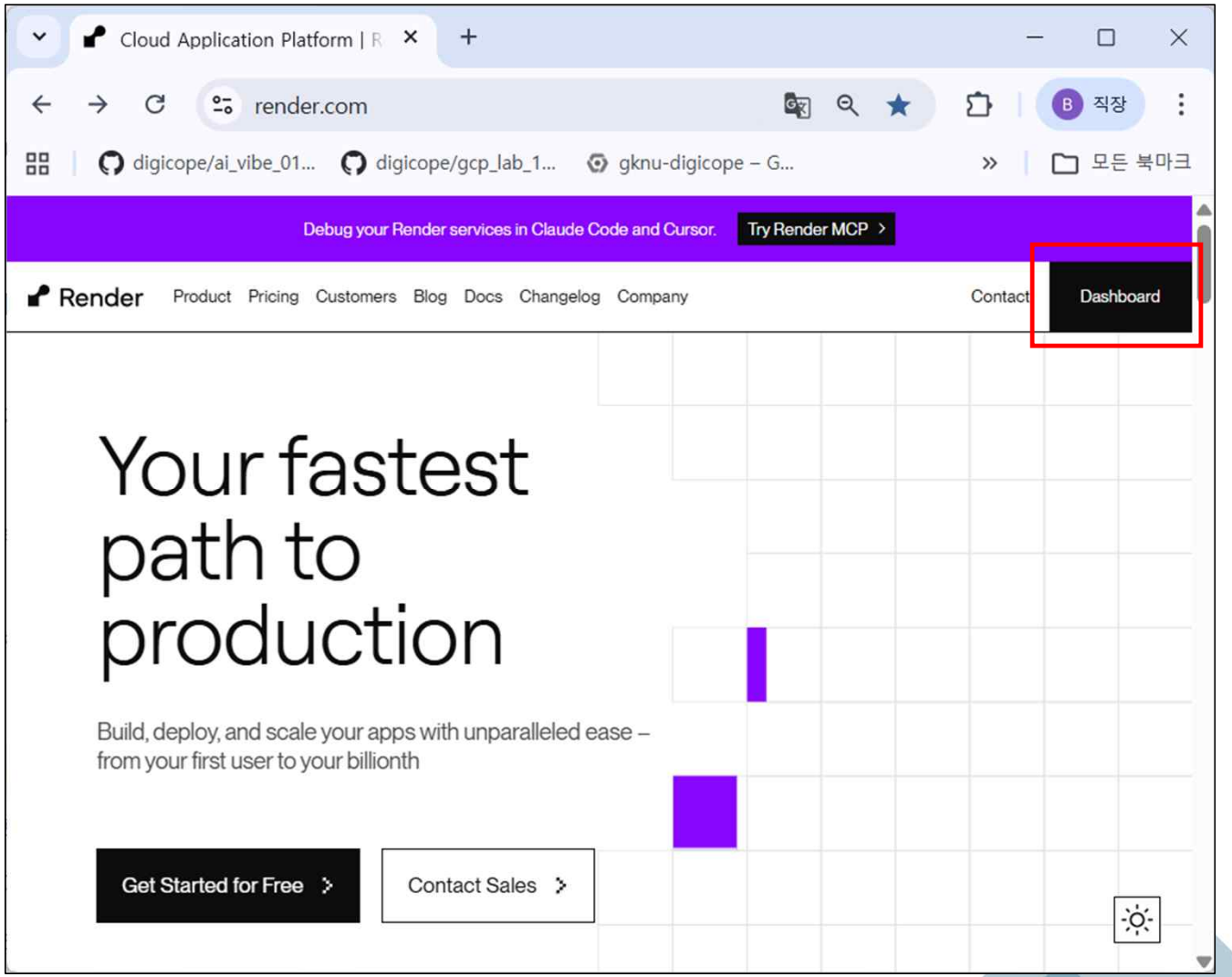
git pull origin main --rebase

git push -u origin main

배포에 사용되는 Git 저장소로 가서 업로드 된 파일들을 확인한다

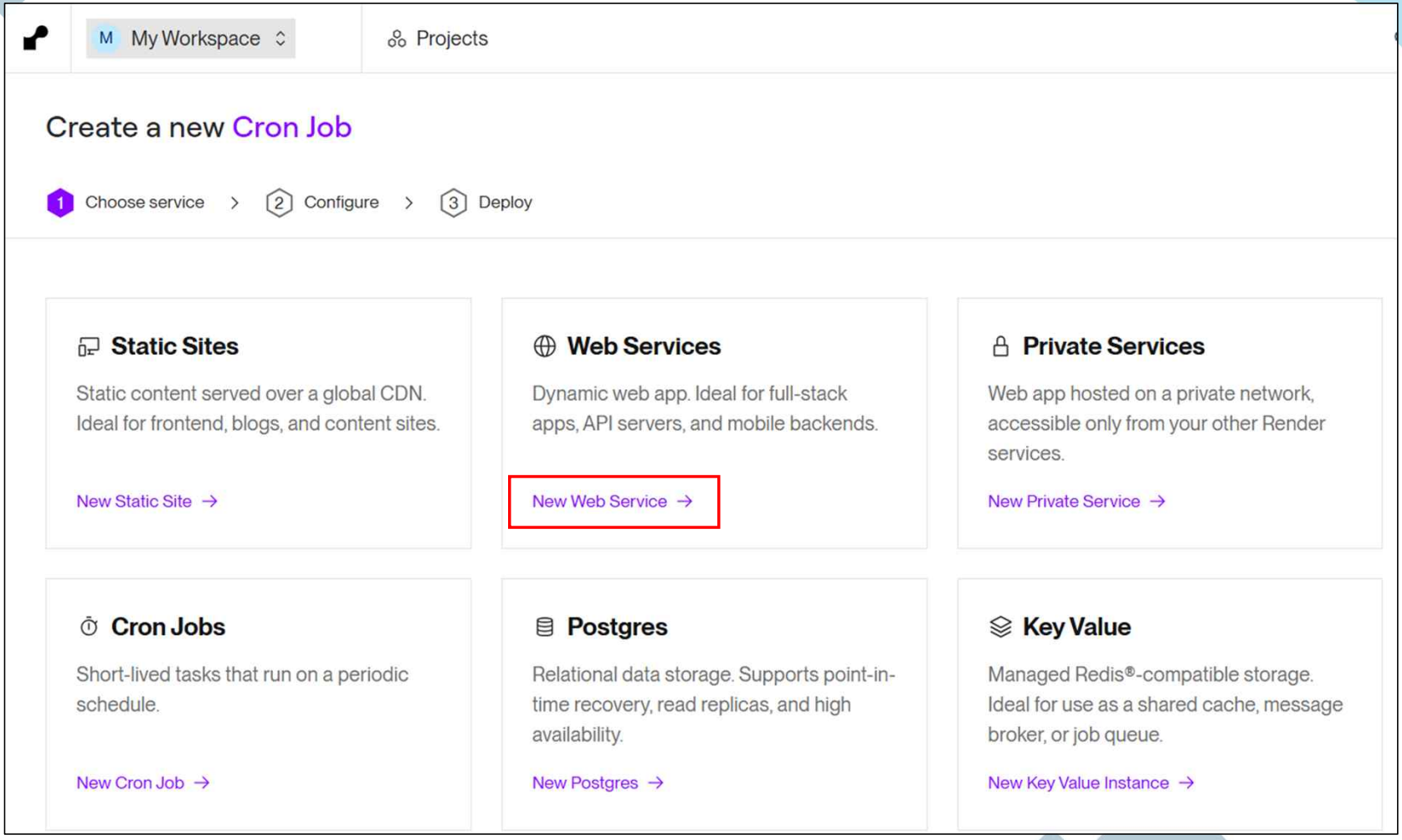


Render 사이트에 계정 가입하고 이메일 인증 완료 후 다시 접속하여 우측 상단의 [Dashboard]를 클릭한다

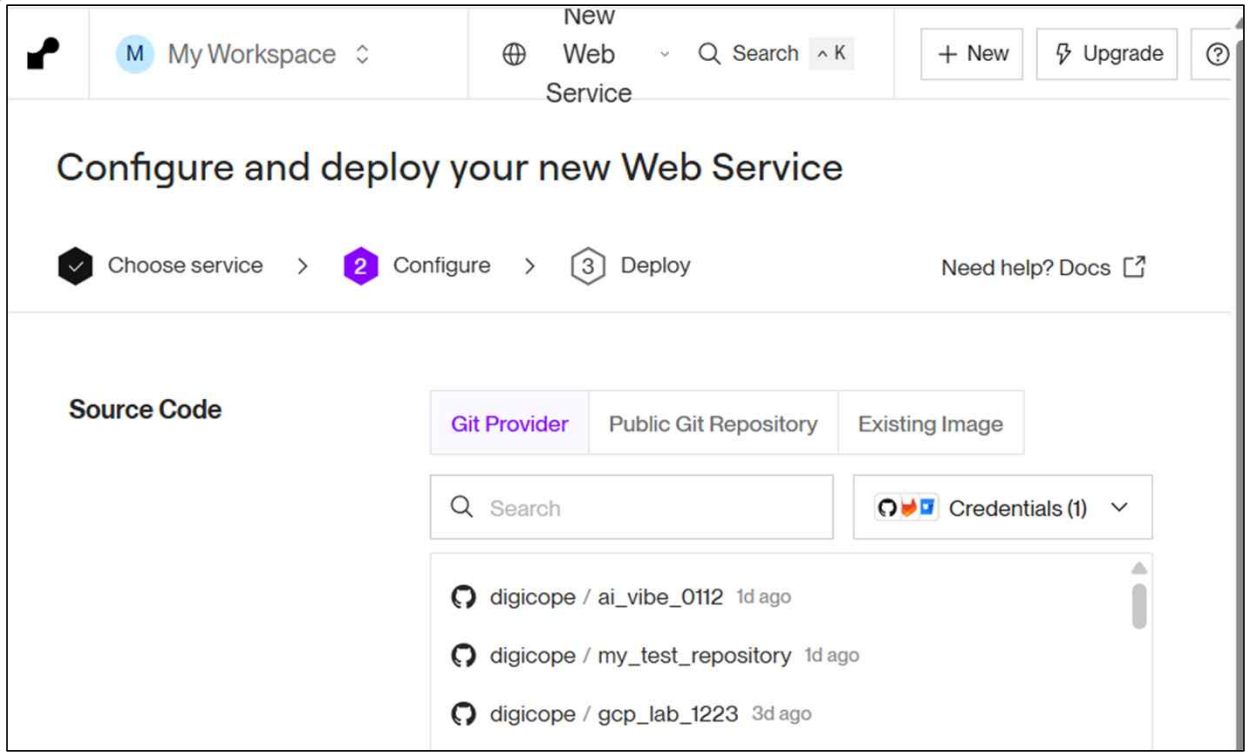


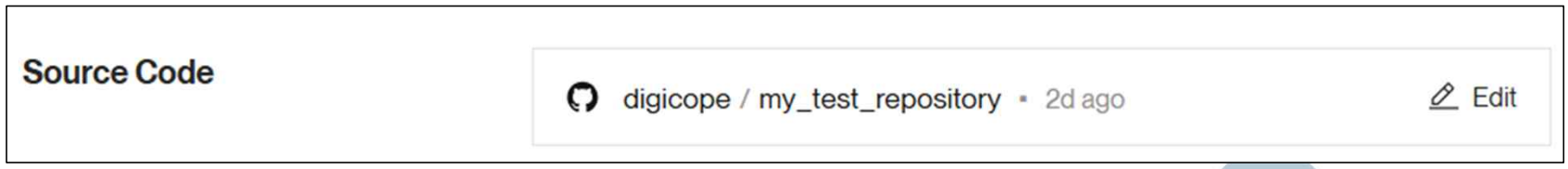
render.com

아래 대시보드화면에서 **Web Services**의 **New Web Service** 를 누른다



Configure and deploy your new Web Service에서 배포에 사용할 저장소를 선택해준다





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_test_repository • 2d ago Edit

Name

A unique name for your web service.

my_test_repository

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.

Oregon (US West)

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.

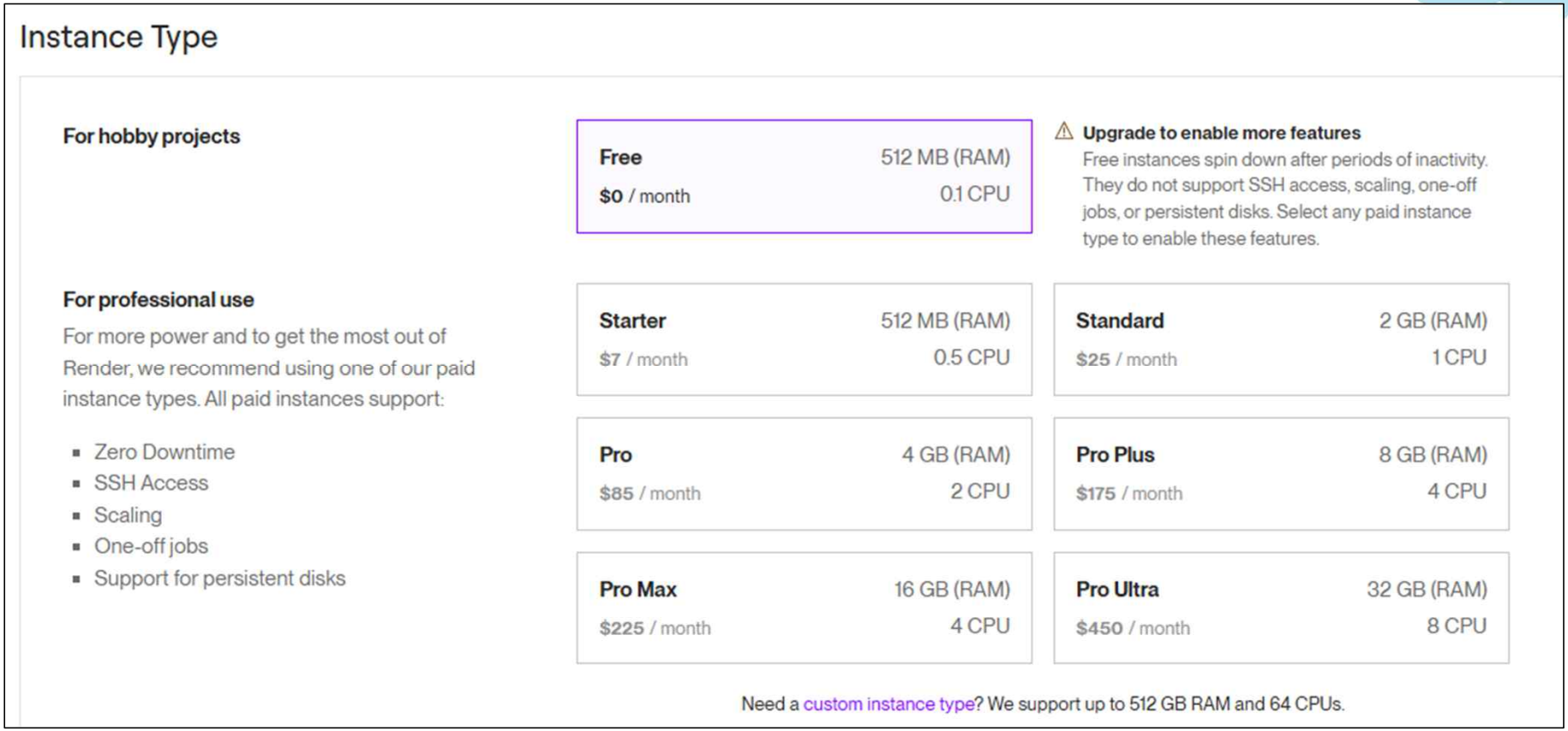
```
$ pip install -r requirements.txt
```

Start Command

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

```
$ gunicorn app:app
```

Instance Type을 Free 로 선택한다

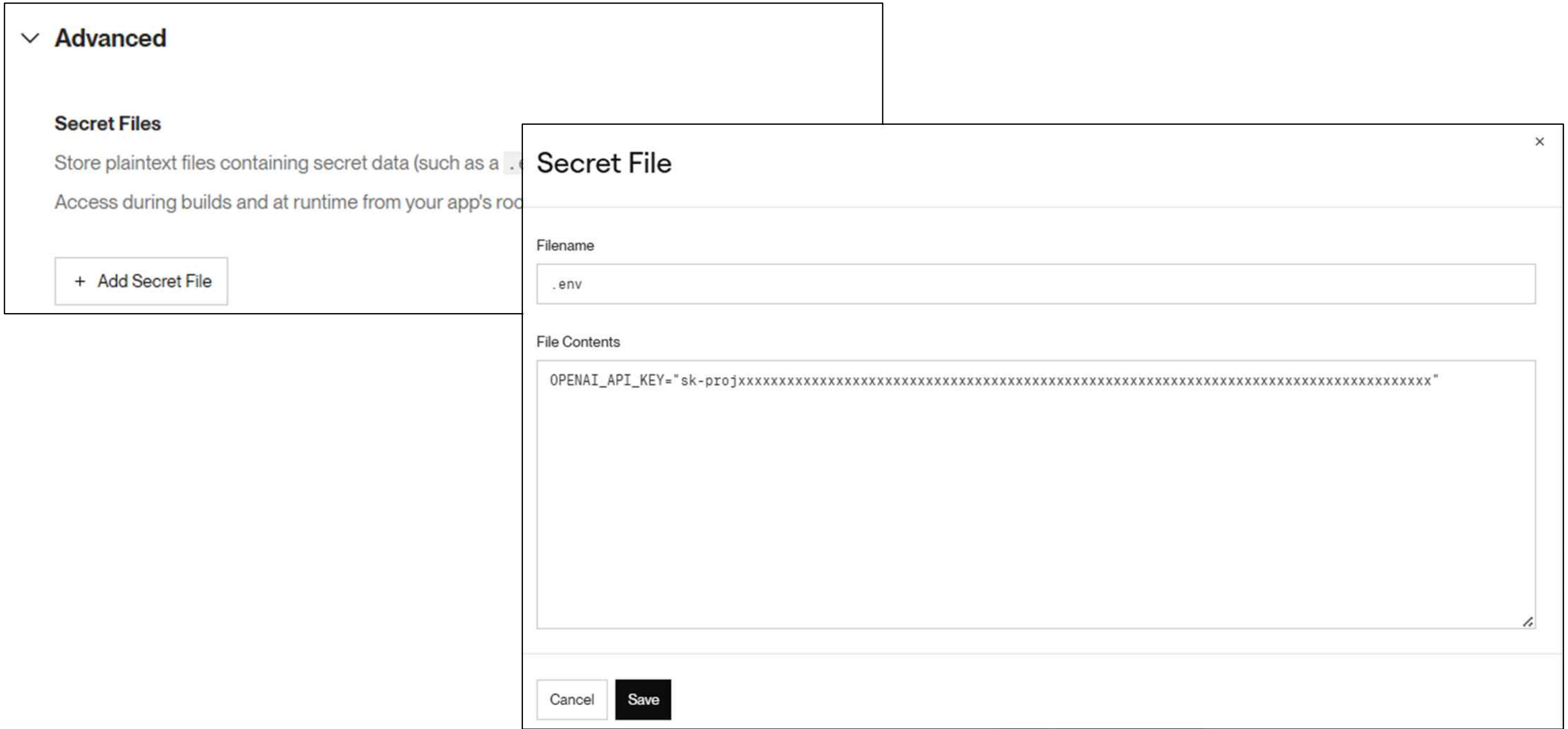


OpenAI API 를 사용하여 환경 변수가 필요한 경우 입력해준다 **(옵션)**

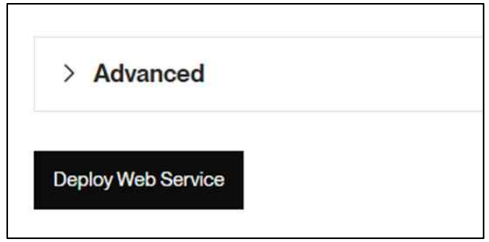
예시)
NAME_OF_VARIABLE : OPENAI_API_KEY
value : sk-projxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (실제 API 키값)

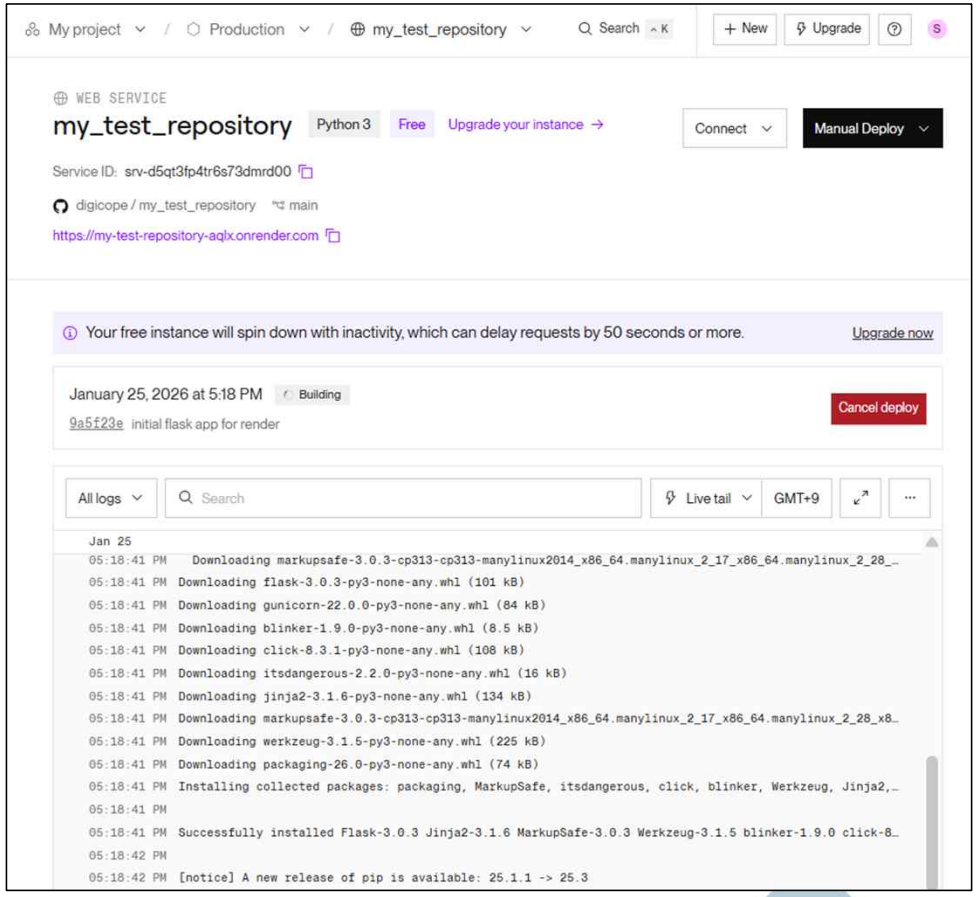
Advanced 에서 [+ Add Secret File]을 클릭하여 Secret File을 작성할 수 있다 **(옵션)**

Filename : .env
File Contents : OPENAI_API_KEY=“sk-projxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”



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





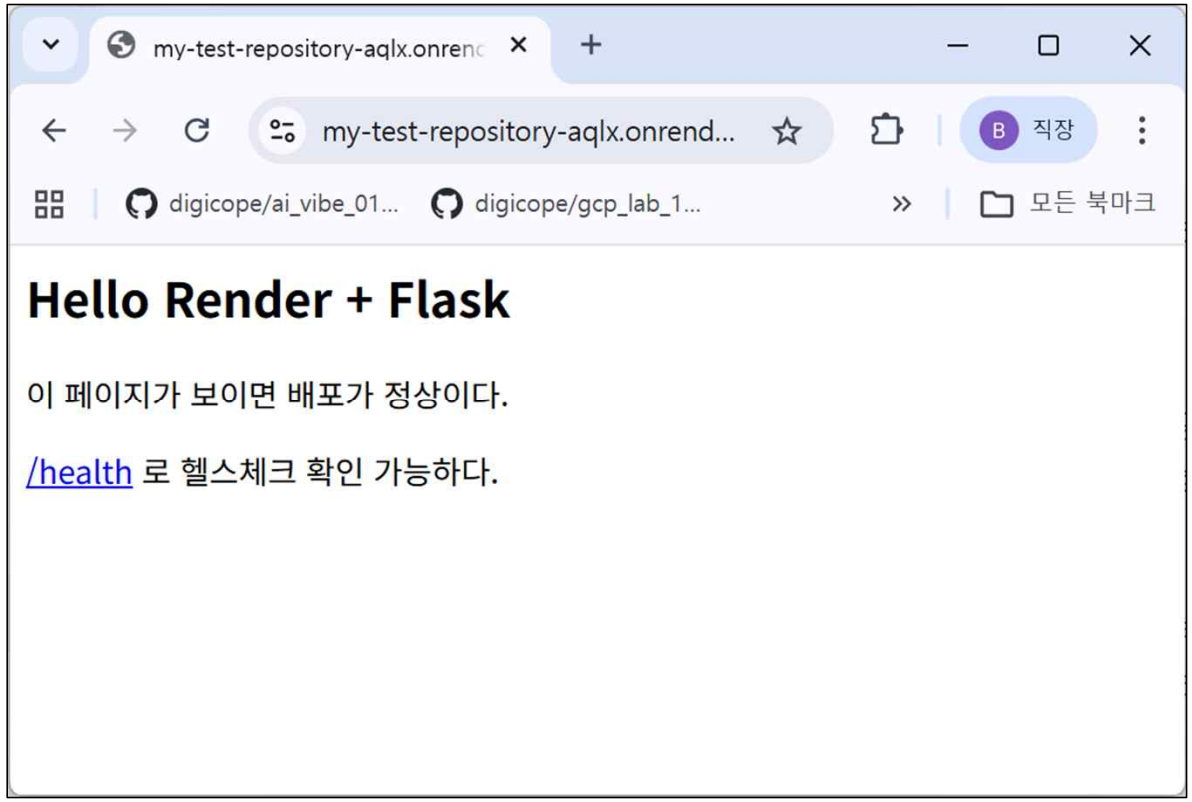
배포 과정 로그 메시지

```
Jan 25
07:03:13 PM  Downloading click-8.3.1-py3-none-any.whl (108 kB)
07:03:13 PM  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
07:03:13 PM  Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
07:03:13 PM  Downloading markupsafe-3.0.3-cp313-cp313-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (22 kB)
07:03:14 PM  Downloading werkzeug-3.1.5-py3-none-any.whl (225 kB)
07:03:14 PM  Installing collected packages: markupsafe, itsdangerous, click, blinker, werkzeug, jinja2, Flask
07:03:16 PM
07:03:16 PM  Successfully installed Flask-3.1.2 blinker-1.9.0 click-8.3.1 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.3 werkzeug-3.1.5
07:03:17 PM
07:03:17 PM  [notice] A new release of pip is available: 25.1.1 -> 25.3
07:03:17 PM  [notice] To update, run: pip install --upgrade pip
07:03:26 PM  ==> Uploading build...
07:03:48 PM  ==> Uploaded in 12.1s. Compression took 10.4s
07:03:48 PM  ==> Build successful 🎉
07:04:02 PM  ==> Setting WEB_CONCURRENCY=1 by default, based on available CPUs in the instance
07:04:02 PM  ==> Deploying...
```

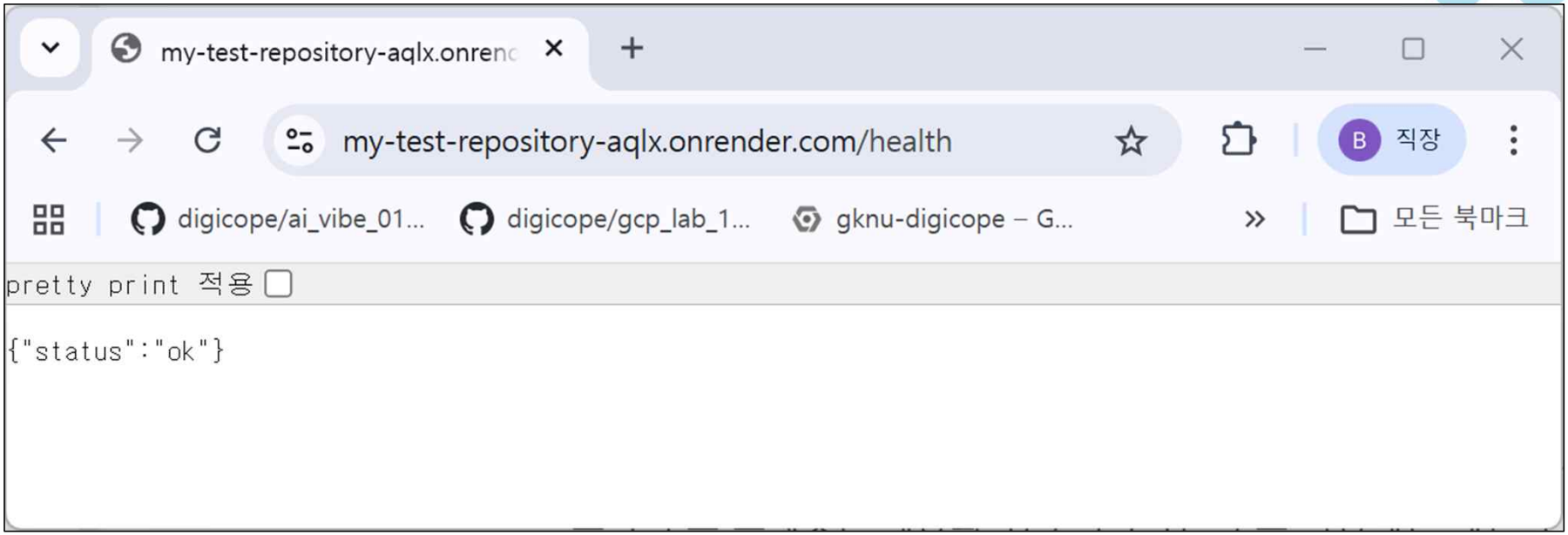
잠시 기다리면 아래와 같이 접속 가능한 URL이 보여진다
URL을 클릭하면 배포된 웹서버로 접속된다

```
05:19:51 PM          ==> ///////////////////////////////////////////////////////////
05:19:51 PM          ==>
05:19:52 PM          ==> Available at your primary URL https://my-test-repository-aqlx.onrender.com
05:19:52 PM          ==>
05:19:52 PM          ==> ///////////////////////////////////////////////////////////
05:19:53 PM  [st47b]  127.0.0.1 - - [25/Jan/2026:08:19:53 +0000] "GET / HTTP/1.1" 200 180 "-" "Go-http-client/2.0"
05:20:53 PM  [st47b]  127.0.0.1 - - [25/Jan/2026:08:20:53 +0000] "GET / HTTP/1.1" 200 180 "-" "Mozilla/5.0 (Window…
05:20:54 PM  [st47b]  127.0.0.1 - - [25/Jan/2026:08:20:54 +0000] "GET /favicon.ico HTTP/1.1" 404 207 "https://my-t…
```

실행된 웹서버 화면


my-test-repository-aqlx.onrenc
my-test-repository-aqlx.onrend...
직장
digicope/ai_vibe_01...
digicope/gcp_lab_1...
모든 북마크
Hello Render + Flask
이 페이지가 보이면 배포가 정상이다.
/health 로 헬스체크 확인 가능하다.

웹 페이지의 하단의 **/health**를 누르면 아래와 같이 OK 출력이 확인 된다



3. Render 사용 Flask 배포 소스 업데이트 하기

배포된 소스 업데이트 하기

- app.py 소스의 home() 함수를 아래와 같이 수정한다(빨간색 부분만 수정)

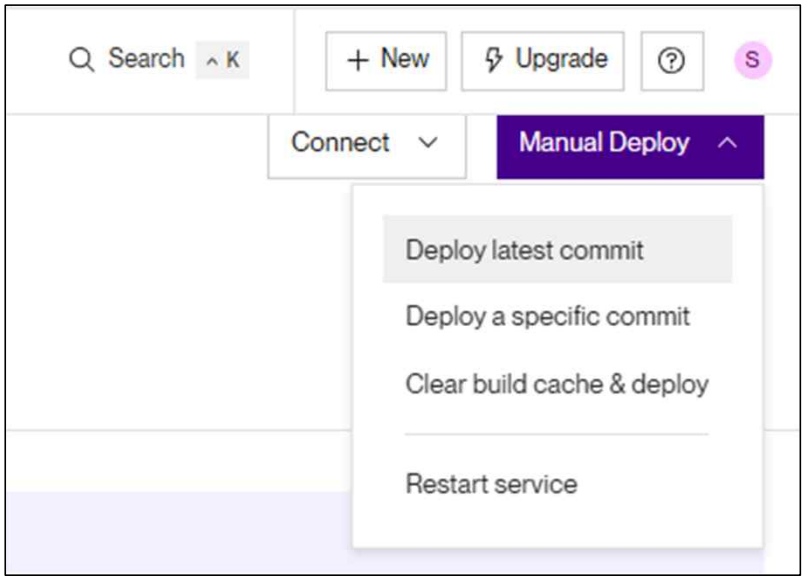
```
@app.get("/")
def home():
    return """
    <h2>Hello Render + Flask</h2>
    <p>이 페이지가 보이면 업데이트 배포가 정상이다.</p>
    <p><a href="/health">/health</a> 로 헬스체크 확인 가능하다.</p>
    """
```

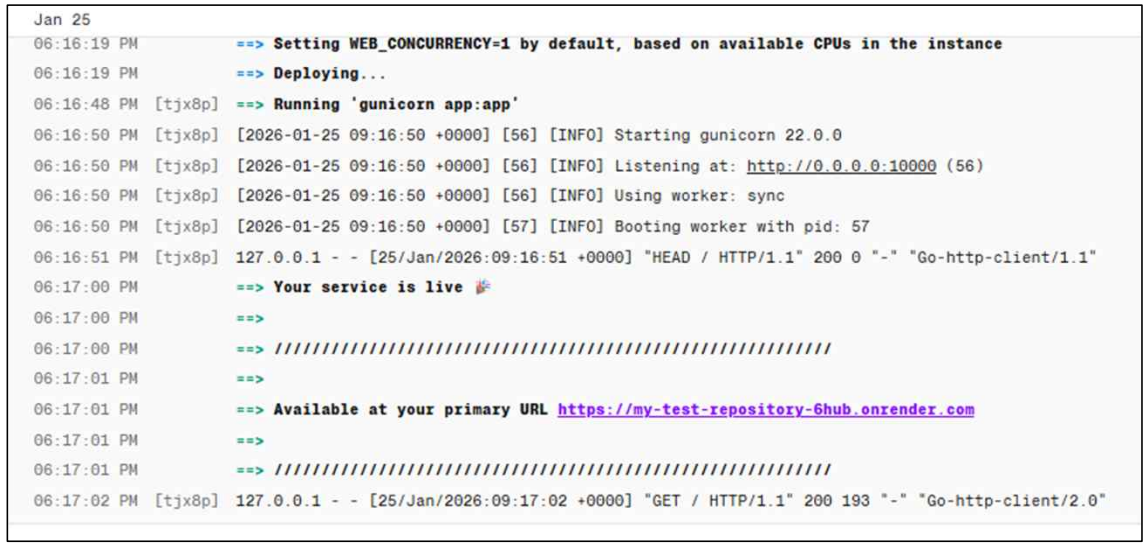
- 소스 수정 후 저장한다음 Git Bash에서 아래 명령을 수행한다

```
git add .
git commit -m "update flask app for render"
git push origin main
```

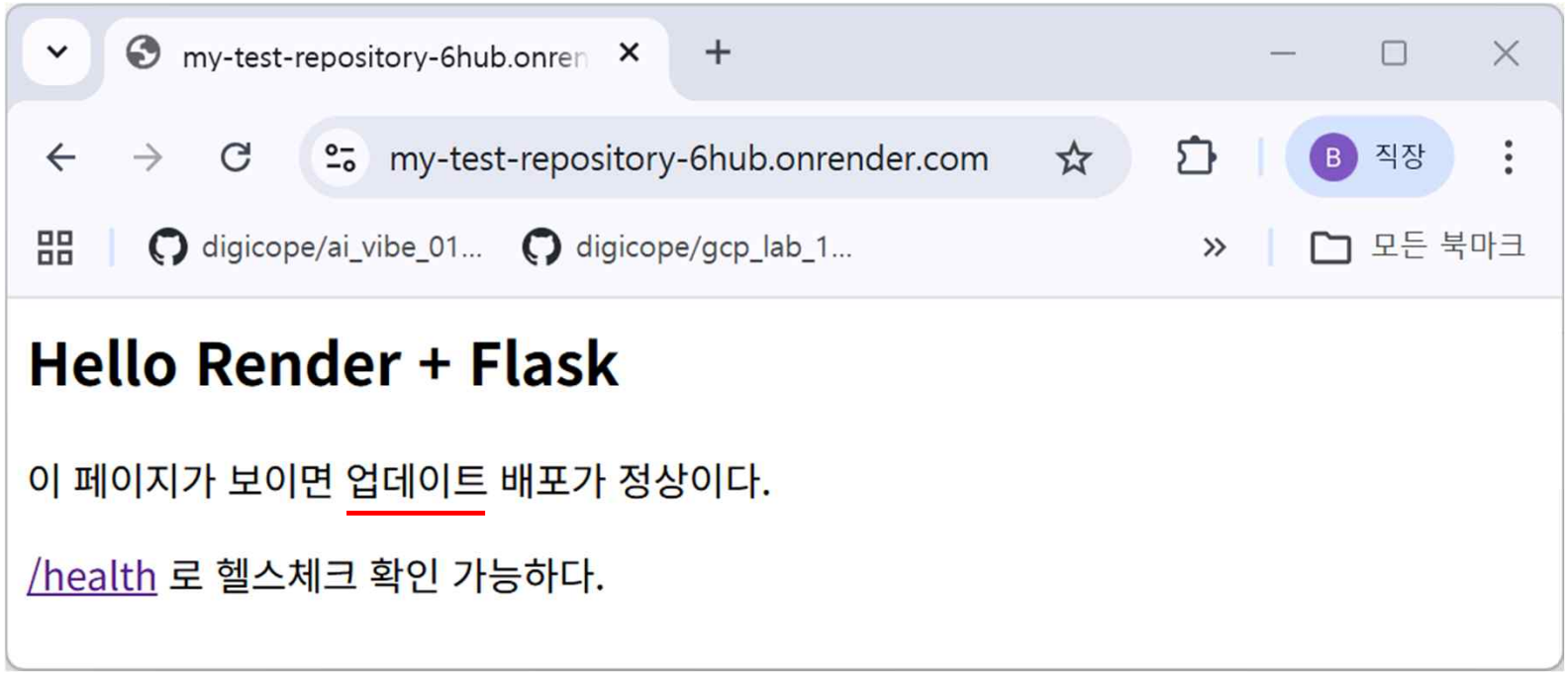
[Manual Deploy] → [Deploy latest commit] 을 누르면 마지막으로 commit 된 소스로 다시 배포 가 수행된다(처음 배포 과정과 동일한 배포 과정이 다시 수행된다)

< Manual Deploy 메뉴로 이동하는 방법 >
render.com → Dashboard → Project : My project
→ Production : my_test_repository → 목록 맨위의 Deploy live for 5d225ce: update flask app for render 클릭





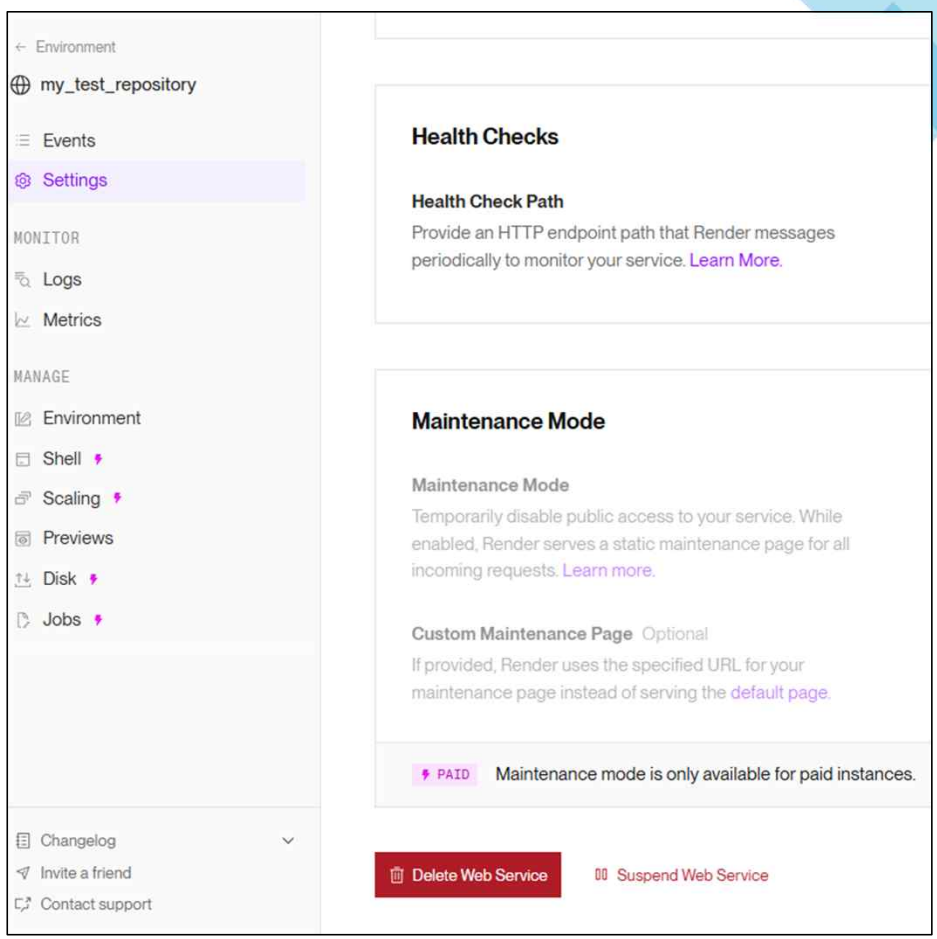
배포 완료 후 생성된 URL을 클릭하면 아래와 같이 변경된 내용을 확인할 수 있다



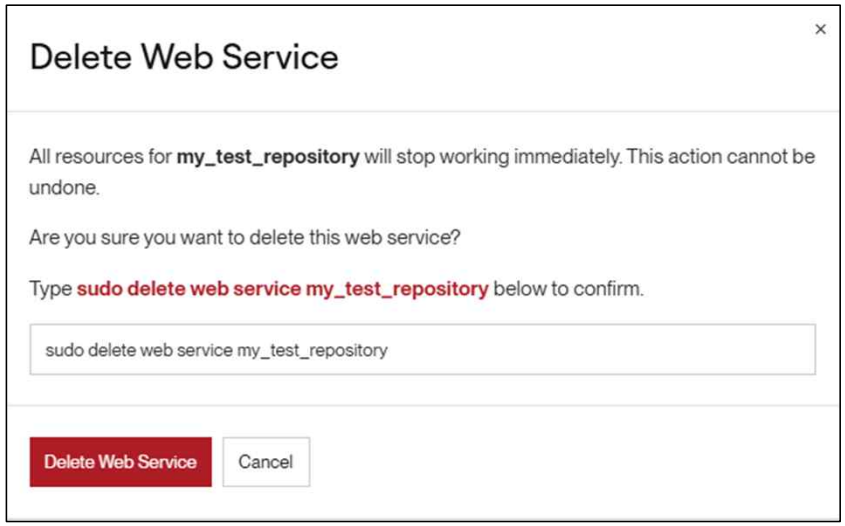
4. Render 사용 Flask 배포 서비스 삭제 하기

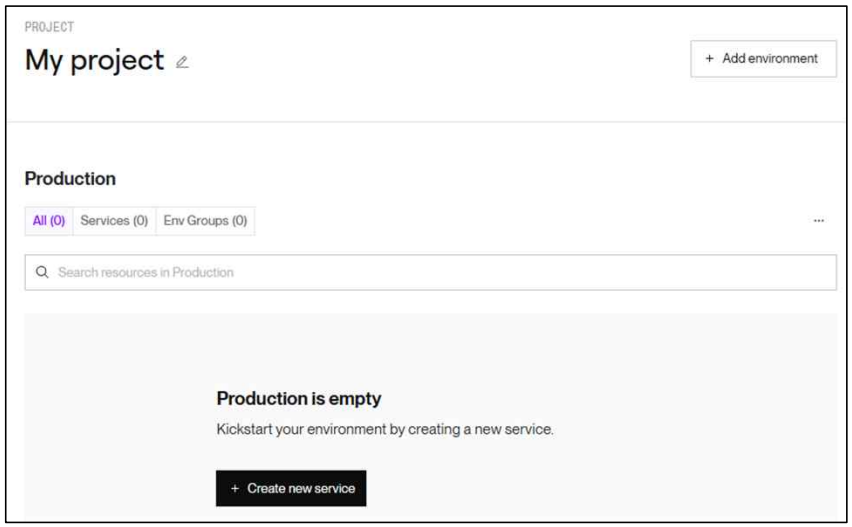
배포된 웹 서비스 삭제하기

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

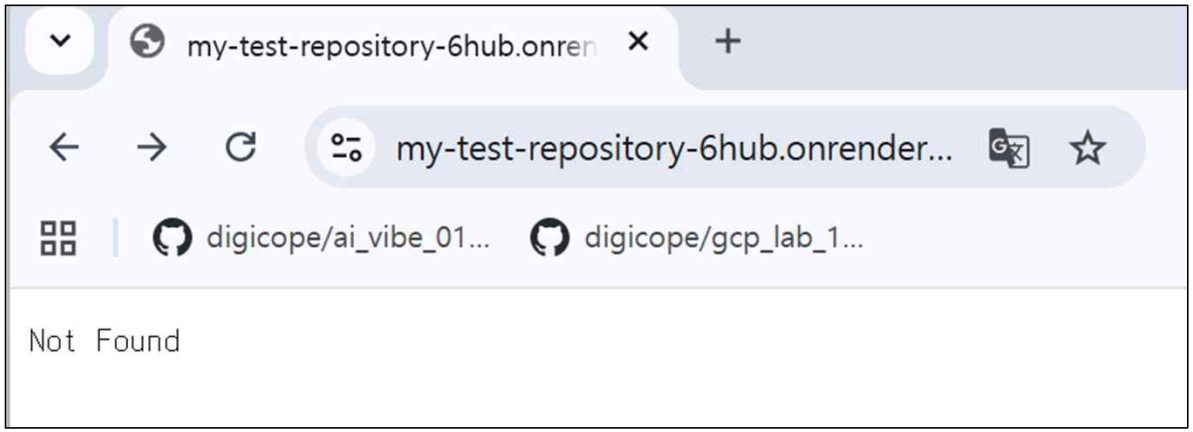


Delete Web Service 에서 sudo delete web service my_test_repository를 입력하고 [Delete Web Service] 버튼을 클릭하면 배포된 웹 서비스가 삭제된다





앞에서 배포된 URL을 클릭하면 Not Found 가 출력된다



Git hub의 모든 파일 삭제하기

(삭제 후에는 다시 되돌리 수 없다. 복구 불가능)

- Git Bash에서 아래 명령 수행한다

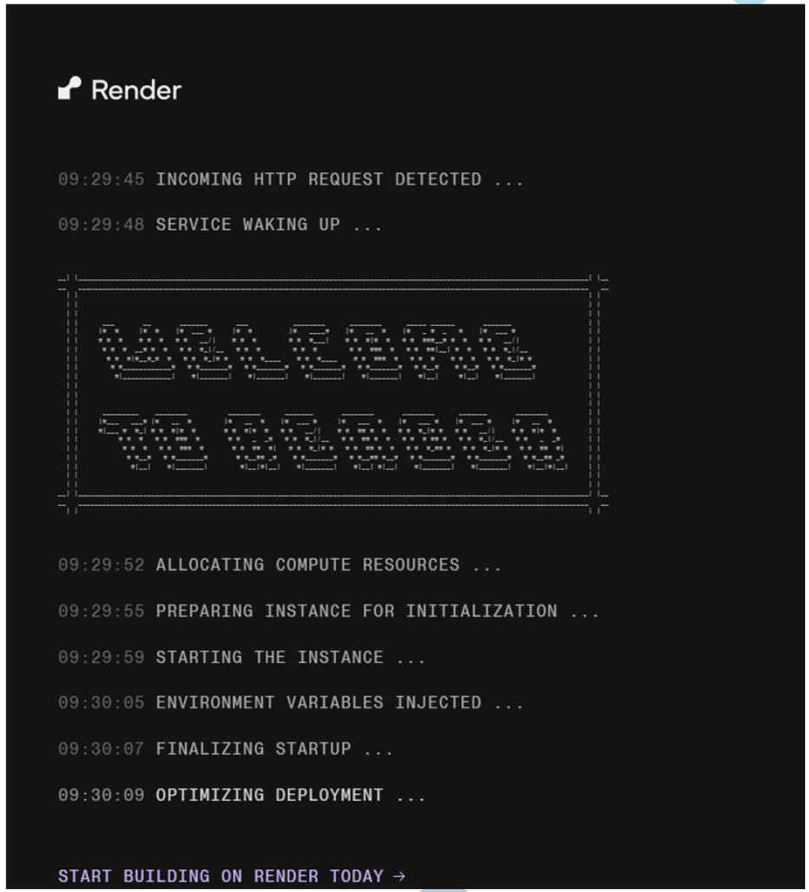
```
git checkout --orphan main
git rm -rf .

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)

Render 무료 버전에서는 배포된 웹 서버에 15분 이상 접속이 없으면 서버가 자동으로 중단(stop) 되고 서버에 다시 접속하면 약 1분 정도 start-up 시간이 걸린다.

우회: 외부에서 10~14분마다 "핑 요청"을 보내 트래픽을 만들어 깨워 둔다
예: UptimeRobot 같은 모니터링, cron-job.org 같은 외부 크론, **GitHub Actions 스케줄러** 등으로 서비스 URL에 GET 요청을 주기적으로 보낸다(15분보다 짧게). 이런 방식이 흔히 사용된다. 다만 이 방식은 한계가 있다.
무료 사용 시간/리소스를 더 소모하게 된다 (결국 비용 대신 제약을 소모하는 구조이다).

서버 중지 후 재 접속 시 시동 화면



감사합니다