Deploying a Flask Web App on Hugging Face
Step-by-Step Guide for Beginners

NIELIT Chandigarh/Ropar

HUGGING FACE

Spaces

NIELIT

What is Hugging Face Spaces?

- A platform to host ML demos and apps.

- Supports **Gradio**, **Streamlit**, and **custom HTML/Flask apps**.

- Free hosting with GitHub-like repo interface.

- Ideal for ML/AI project showcases.

What is Flask?

- Flask is a Python micro web framework.

- Lightweight and easy to use for building web apps.

- Ideal for REST APIs and small frontends.

Requirements for Deployment

- A Hugging Face account

- A Flask project (with app.py)

- requirements.txt file
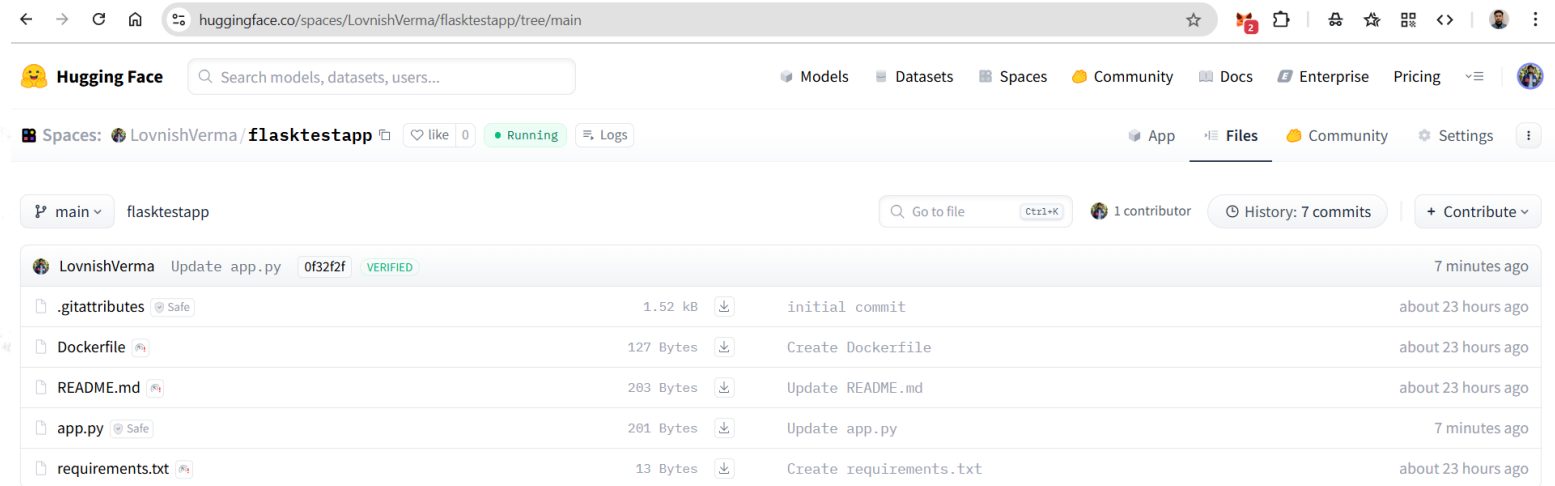
- Dockerfile for deployment

Project Directory Structure

my-flask-app/

├── app.py

├── requirements.txt

└── Dockerfile

Add more files like templates/static if needed.

Sample app.py

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")

def home():
    return "Hello from Flask on Hugging Face!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=7860, debug=True)
```
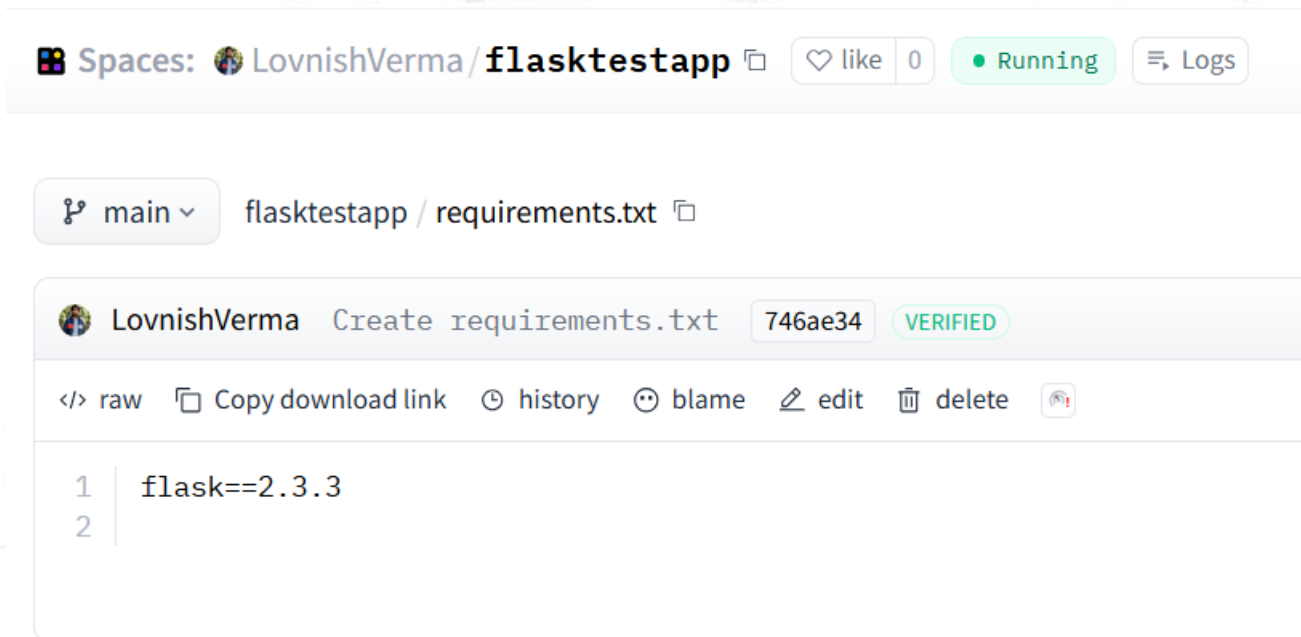
Spaces: 👤 LovnishVerma / **flasktestapp** 📋   ❤ like 1   ● Running   ≡ Logs

⌐ main ˅   flasktestapp / app.py 📋

👤 LovnishVerma   Update app.py   0f32f2f   VERIFIED

</> raw   📋 Copy download link   🕐 history   😐 blame   ✎ edit   🗑 delete   ⊘ Safe

```python
1    from flask import Flask
2    app = Flask(__name__)
3
4    @app.route('/')
5    def home():
6        return "Hello from Flask on Hugging Face!"
7
8    if __name__ == '__main__':
9        app.run(host="0.0.0.0", port=7860, debug=True)
10
```

HUGGING FACE

Create requirements.txt

Include all dependencies:

flask==2.3.3

📌 *Always pin exact versions to avoid errors.*

Dockerfile

FROM python:3.9

WORKDIR /app

COPY requirements.txt ./

RUN pip install -r requirements.txt

COPY . .

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

Spaces: 👤 LovnishVerma / **flasktestapp** 🗐   ♡ like 0   ● Running   ☰ Logs

⑂ main ∨   flasktestapp / Dockerfile 🗐

👤 LovnishVerma   Create Dockerfile   9a84d4d   VERIFIED

</> raw   🗐 Copy download link   🕐 history   ☺ blame   ✏ edit   🗑 delete   📋

```
 1    FROM python:3.9
 2
 3    WORKDIR /app
 4
 5    COPY requirements.txt ./
 6    RUN pip install -r requirements.txt
 7
 8    COPY . .
 9
10    CMD ["python", "app.py"]
```
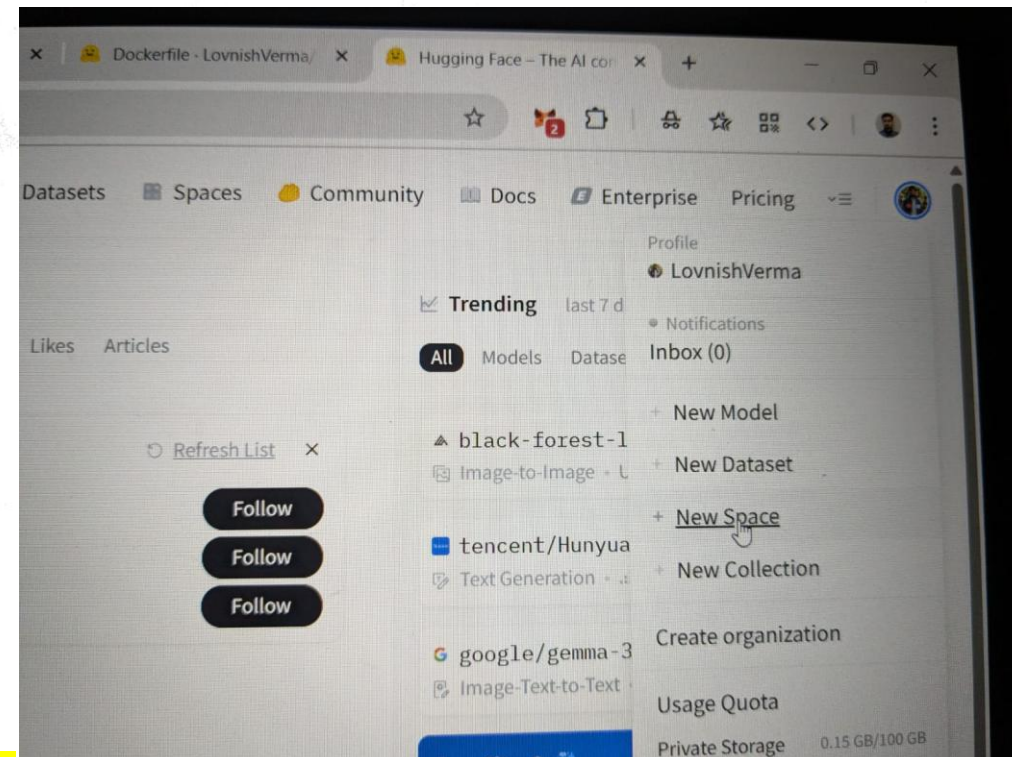
Create a Hugging Face Space

Steps:

1. Go to: https://huggingface.co/spaces

2. Click "Create New Space"

3. Fill details:
   1. Space name
   2. License
   3. SDK: **Docker**
   4. Visibility: Public / Private

4. Link your GitHub repo OR upload files directly.

Auto Build & Deploy

Deployment Process
Content:

- Hugging Face reads Dockerfile and builds the container.

- Flask app runs on port 7860 inside the container.

- Space will show live preview when deployed.

Final Output

Deployed App Preview
Content:

- Share the URL: https://huggingface.co/spaces/username/space-name

- Demo:

Troubleshooting Tips

Common Errors & Fixes

Content:

- App not loading? Check port=7860 and host="0.0.0.0"

- Missing module? Add to requirements.txt

- Docker build failed? Ensure all files are included and no syntax errors.

Resources

Helpful Links
Content:

- Flask Official Docs

- Hugging Face Spaces

- Dockerfile Reference

- Demo project with gunicorn

Want to Use Gunicorn (Optional)

Gunicorn (short for **Green Unicorn**) is a **Python WSGI HTTP server** that is commonly used to run Python web applications in **production environments** — especially apps built with **Flask**, **Django**, **FastAPI**, and other WSGI-compatible frameworks.

🧠 **What is WSGI?**

WSGI stands for **Web Server Gateway Interface** — it's a standard interface between **Python web applications** and **web servers**.

- Flask, Django, etc. are **WSGI applications**.

- Gunicorn is a **WSGI server** that knows how to run those apps efficiently.

Why Use Gunicorn?

While developing, you use app.run() in Flask, which is fine for testing but **not suitable for production** because:

- It handles only one request at a time.

- It's not scalable or secure.

Gunicorn solves this by:

- Running your app on multiple **worker processes**.

- Efficiently handling **many simultaneous requests**.

- Being compatible with production web servers like **Nginx**.

🔍 Gunicorn Features

- Supports **multiple worker types** (sync, async, threaded).

- Can be used with **load balancers**.

- Runs in the background with **process management**.

- Supports **pre-fork model** like Unicorn (Ruby), hence the name.

Gunicorn is a production-ready Python WSGI server that serves your Flask/Django app efficiently, usually behind a web server like Nginx.

How to Use Gunicorn with Flask

- You have to add gunicorn in your Dockerfile and also in requirements.txt

- Modify your Dockerfile

Spaces: LovnishVerma/**flasktestapp** ♥ like 1 ● Running ≡ Logs

⑂ main ∨   flasktestapp / Dockerfile

LovnishVerma   Update Dockerfile   bbebcbd   VERIFIED

</> raw   Copy download link   ⏱ history   ☺ blame   ✎ edit   🗑 delete

```
1   FROM python:3.9
2
3   WORKDIR /app
4
5   COPY requirements.txt ./
6   RUN pip install -r requirements.txt
7
8   COPY . .
9
10  CMD ["gunicorn", "app:app", "-b", "0.0.0.0:7860"]
```

Add gunicorn in requirements.txt

- gunicorn

🤗 **Hugging Face**   🔍 Search models, datasets, users...

🔳 Spaces: 🧑 LovnishVerma / **flasktestapp** 🗐   ❤️ like   1   • Running   ≡. Logs

ᛘ main ˅   flasktestapp / requirements.txt 🗐

🧑 LovnishVerma   Update requirements.txt   019df54   VERIFIED

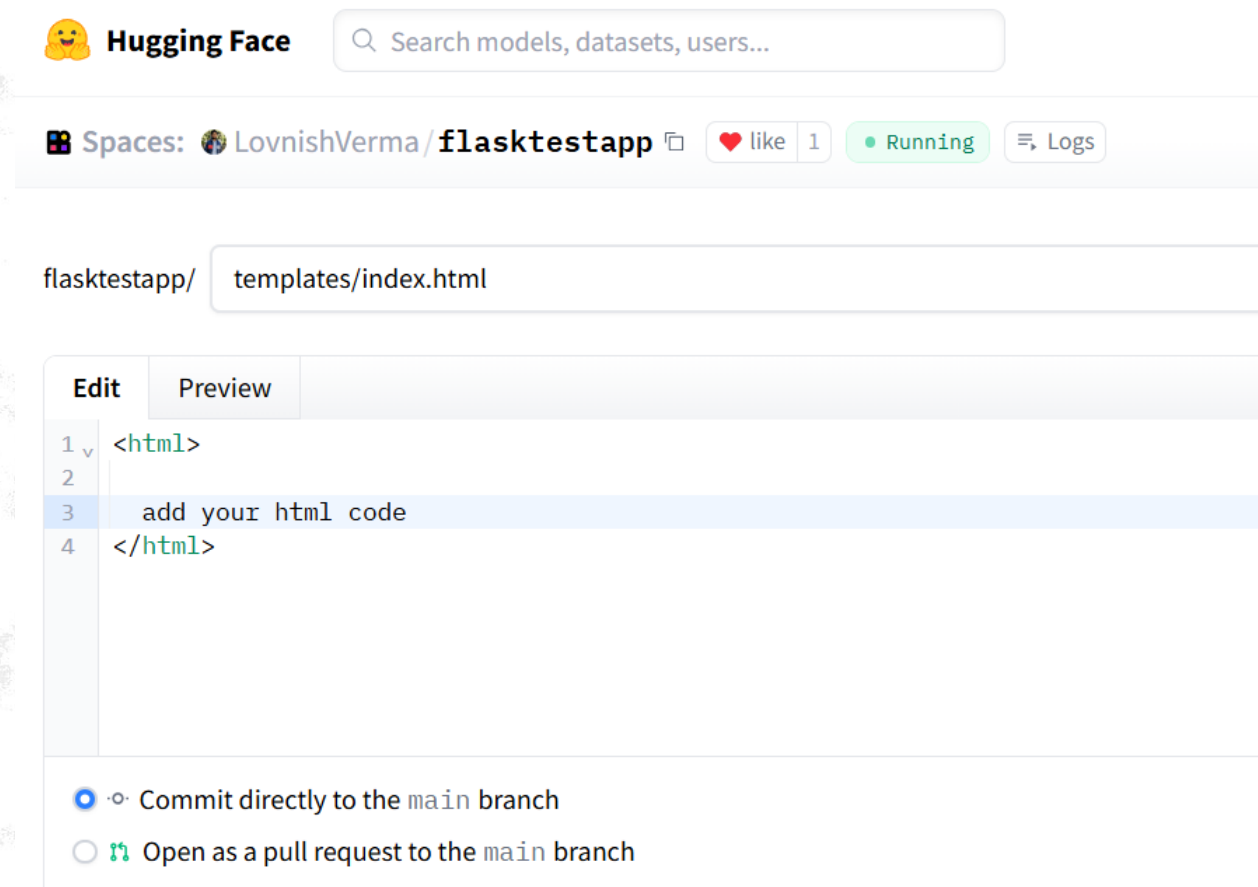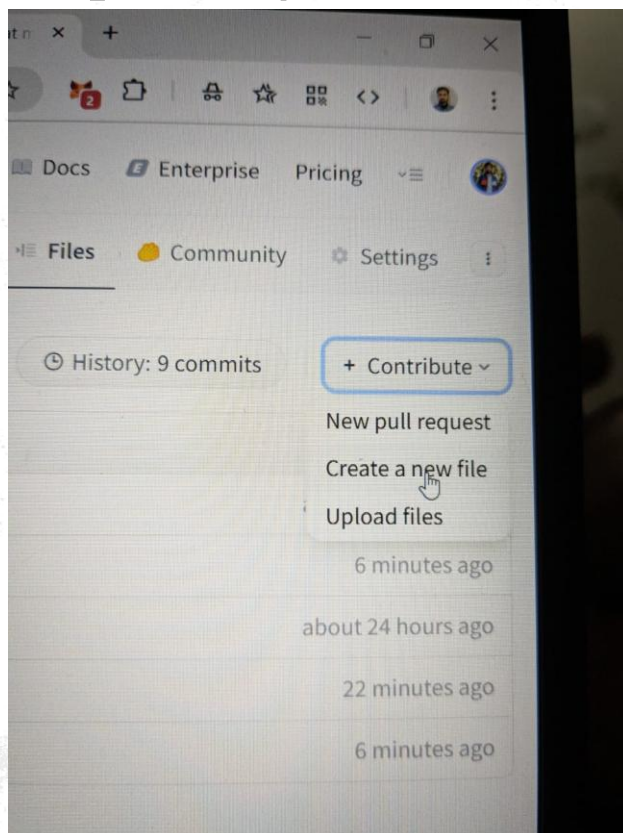</> raw   🗐 Copy download link   ⏱ history   ☺ blame   ✎ edit   🗑 delete   🛡 Safe

```
1    flask==2.3.3
2    gunicorn
```

How to create templates/index.html

- Click on contribute then create a new file

- templates/index.html

THANKS