GPT MODEL SELF HOSTING AND EXPOSING IT AS AN API TO THE WORLD

Step 1: System Requirements

- **Ubuntu 22.04+**
- **Python 3.12**
- **H100 GPU** with CUDA 12.1+ (driver installed)
- Ports open: `8000` (for API access)
- Internet access (for model download)

Step 2: Install Python 3.12

```
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update
sudo apt install python3.12 python3.12-venv python3.12-dev -y
```

Step 3: Create Virtual Environment

```
python3.12 -m venv .venv
source .venv/bin/activate
```

Step 4: Upgrade pip, setuptools, wheel

```
pip install --upgrade pip setuptools wheel
```

Step 5: Install `vLLM` with GPT-OSS Wheels (Follow Below Step Only For Chat Completion Model)

```
pip install --pre vllm==0.10.1+gptoss \
 --extra-index-url https://wheels.vllm.ai/gpt-oss \
 --extra-index-url https://download.pytorch.org/whl/nightly/cu128
```

Step 6: Run the vLLM Server (Follow Below Step Only For Chat Completion Model)

```
vllm serve openai/gpt-oss-20b \
 --port 8000 \
 --gpu-memory-utilization 0.9 \
 --max-model-len 8192
```

You now have a live **OpenAI-compatible chat API** running at:

```
http://<YOUR_PUBLIC_IP>:8000/v1/chat/completions
```

Step 7: Test the API (Python and curl) (Follow Below Step Only For Chat Completion Model)

?? Non-streaming

```
python

import openai

openai.api_base = "http://<YOUR_PUBLIC_IP>:8000/v1"
openai.api_key = "sk-fake" # Not used in vLLM

res = openai.ChatCompletion.create(
 model="openai/gpt-oss-20b",
 messages=[{"role": "user", "content": "Explain quantum mechanics"}] )
print(res.choices[0].message["content"])


curl -X POST "http://185.216.20.149:8000/v1/chat/completions" \
-H "Content-Type: application/json" \
 -H "Authorization: Bearer not-needed" \
 -d '{
```

```
 "model": "openai/gpt-oss-20b",
 "messages": [
 {"role": "user", "content": "Explain quantum mechanics"}  ],
 "max_tokens": 200
 }'
```

?? Streaming (recommended)

python

```
import openai

openai.api_base = "http://<YOUR_PUBLIC_IP>:8000/v1"
openai.api_key = "not-needed"

res = openai.ChatCompletion.create(
 model="openai/gpt-oss-20b",
 messages=[{"role": "user", "content": "Explain quantum mechanics"}],
stream=True
)

for chunk in res:
 print(chunk.choices[0].delta.get("content", ""), end="", flush=True)
```

```
curl -N -X POST "http://185.216.20.149:8000/v1/chat/completions" \
-H "Content-Type: application/json" \
 -H "Authorization: Bearer not-needed" \
 -d '{
 "model": "openai/gpt-oss-20b",
 "messages": [
 {"role": "user", "content": "Explain quantum mechanics"}  ],
 "max_tokens": 2000,
 "stream": true
 }'
```

(Follow Below Step Only For Embediing Model)

```
pip install setuptools==79.0.0

pip install -U vllm
vllm serve BAAI/bge-m3 \
--host 0.0.0.0 --port 8000 \
--gpu-memory-utilization 0.90 --enforce-eager
curl -X POST http://38.128.233.100:8000/v1/embeddings \
-H "Authorization: Bearer dummy" \
```

```
-H "Content-Type: application/json" \
-d '{
  "model": "BAAI/bge-m3",
  "input": "The quick brown fox jumped over the lazy dog"
}'

curl -X POST http://69.19.137.107:8000/v1/embeddings \
-H "Authorization: Bearer dummy" \
-H "Content-Type: application/json" \
-d '{
  "model": "BAAI/bge-m3",
  "input": "The quick brown fox jumped over the lazy dog"
}'
```

(Optional) Step 8: Expose Publicly

Option A: Use ngrok (quick test)

```
ngrok http 8000
```

Option B: Use Nginx + domain + HTTPS

Let me know if you want a full `nginx + SSL` reverse proxy config.

(Optional) Step 9: Run on Boot with `systemd`

Create a unit file to auto-run the API after reboot.

Let me know and I'll generate this if needed.

Bonus: Benchmark

```
ab -n 100 -c 10 http://localhost:8000/v1/chat/completions
```

Or use Locust/JMeter for load testing.

You're Done!

You now have:

A 20B model running locally
Streaming support via OpenAI-compatible API
GPU-efficient inference with vLLM
Ready to serve apps, agents, or frontends

Happy scaling!