LLM Engine

A software that can load the LLM Models

- Open WebUI
- Kuwa Gen AI OS
- AnythingLLM
- Ollama
- LM Studio
- OpenLLM
- Bechmark
- More
- Llama.Cpp

Open WebUI

A Web UI Tool for Ollama

URLs

- https://openwebui.com/
- GitHub: https://github.com/open-webui/open-webui
- Docs: https://docs.openwebui.com/

Installation

Installing Both Open WebUI and Ollama Together:

```
# With GPU Support
docker run -d -p 3000:8080 --gpus=all \
  -v ollama:/root/.ollama \
  -v open-webui:/app/backend/data \
  --name open-webui \
  --restart always \
  ghcr.io/open-webui/open-webui:ollama
```

```
# For CPU only
docker run -d -p 3000:8080 \
  -v ollama:/root/.ollama \
  -v open-webui:/app/backend/data \
  --name open-webui \
  --restart always \
  ghcr.io/open-webui/open-webui:ollama
```

Kuwa Gen AI OS

??GenAI???

1. ? ?????GenAI???????????????Windows?Linux
2. ? ?????????? Prompt ?????/??/?????????
3. ? ????? Prompt x RAGs x Bot x ?? x ??/GPUs???????
4. ? ????????????????????????????????????
5. ? ????????????????????????????????

URLs

- Kuwa AI | Kuwa AI
- ??? | Kuwa AI
- Kuwa GenAI OS - ?? | Kuwa AI

AnythingLLM

The ultimate AI business intelligence tool. Any LLM, any document, full control, full privacy.

AnythingLLM is a **"single-player"** ??????application you can install on any Mac, Windows, or Linux operating system and get local LLMs, RAG, and Agents with little to zero configuration and full privacy.

> AnythingLLM ??????????????????

You can install AnythingLLM as a Desktop Application, Self Host it locally using Docker and Host it on cloud (aws, google cloud, railway etc..) using Docker

You want AnythingLLM Desktop if...

- You want a one-click installable app to use local LLMs, RAG, and Agents locally
- You do not need multi-user support
- Everything needs to stay only on your device
- You do not need to "publish" anything to the public internet. Eg: Chat widget for website

URLs

- https://useanything.com/
- Doc: https://docs.useanything.com/
- ?????Mintplex-Labs/anything-llm

Ollama

Run Llama 3, Phi 3, Mistral, Gemma, and other models. Customize and create your own.

- https://ollama.com/
- GitHub: https://github.com/ollama/ollama
- Doc: https://github.com/ollama/ollama/tree/main/docs
- Video: ????????????? AI ?? Ollama ???????????????? - YouTube

Installation

ollama + open webui

```
mkdir ollama-data download open-webui-data
```

docker-compose.yml:

```
services:
  ollama:
    image: ollama/ollama:latest
    ports:
      - 11434:11434
    volumes:
      - ./ollama-data:/root/.ollama
      - ./download:/download
    container_name: ollama
    pull_policy: always
    tty: true
    restart: always
    networks:
      - ollama-docker

  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    container_name: open-webui
    volumes:
```

```
      - ./open-webui-data:/app/backend/data
    depends_on:
      - ollama
    ports:
      - 3000:8080
    environment:
      - 'OLLAMA_BASE_URL=http://ollama:11434'
    extra_hosts:
      - host.docker.internal:host-gateway
    restart: unless-stopped
    networks:
      - ollama-docker

networks:
  ollama-docker:
    external: false
```

ollama

```
mkdir ollama-data download


docker run --name ollama -d --rm \
    -v $PWD/ollama-data:/root/.ollama \
    -v $PWD/download:/download \
    -p 11434:11434 \
    ollama/ollama
```

K8s Deployment

- Ollama Kubernetes: Run AI Models Seamlessly on K8s
- Ollama Kubernetes ??????? ????????????? ????????????? - ?????
- ? Kubernetes ??? llama3 | Kubernetes ????
- Enable GPU Support in Kubernetes: Complete Guide

1. ?? *hostpath-storage*

```
microk8s enable hostpath-storage
microk8s status
```

Verify the Storage Class

```
❯ kubectl get storageclass
NAME                          PROVISIONER            RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
microk8s-hostpath (default)   microk8s.io/hostpath   Delete          WaitForFirstConsumer
false                  17m
```

2. `ollama-pvc.yaml` :

- PVC ????????? *Pending*??????????????? *Bound*?
- PersistentVolume ???????????????

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ollama-pvc
  namespace: ollama
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

3. `ollama-deployment.yaml` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ollama
  namespace: ollama
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ollama
  template:
    metadata:
      labels:
        app: ollama
    spec:
```

```
      containers:
        - name: ollama
          image: ollama/ollama:latest
          env:
            - name: OLLAMA_HOST
              value: 0.0.0.0:11434
          ports:
            - name: http
              containerPort: 11434
              protocol: TCP
          volumeMounts:
            - name: ollama-data
              mountPath: /root/.ollama
      volumes:
        - name: ollama-data
          persistentVolumeClaim:
            claimName: ollama-pvc
```

4. `ollama-svc.yaml` :

```
apiVersion: v1
kind: Service
metadata:
  name: ollama-service
  namespace: ollama
spec:
  selector:
    app: ollama
  ports:
  - protocol: TCP
    port: 11434
    targetPort: 11434
  type: ClusterIP
```

Testing with curl

```
curl -s http://<NODE_IP>:<nodeport>/api/generate -d '{
  "model": "llama2",
```

```
  "prompt": "Why is the sky blue?"
}' | jq -r '.response' | tr -d '\n'
```

Verify GPU support

`kubectl logs -n ollama -l name=ollama`

The last line in the example output above shows that Ollama is using a single Tesla V100-SXM2-16GB GPU.

```
2024/09/27 18:51:55 routes.go:1153: INFO server config env="map[CUDA_VISIBLE_DEVICES:
GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVICES: HSA_OVERRIDE_GFX_VERSION: HTTPS_PROXY: HTTP_PROXY:
NO_PROXY: OLLAMA_DEBUG:false OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHEAD:0
OLLAMA_HOST:http://0.0.0.0:11434 OLLAMA_INTEL_GPU:false OLLAMA_KEEP_ALIVE:5m0s
OLLAMA_LLM_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512
OLLAMA_MODELS:/root/.ollama/models OLLAMA_NOHISTORY:false OLLAMA_NOPRUNE:false
OLLAMA_NUM_PARALLEL:0 OLLAMA_ORIGINS:[http://localhost https://localhost http://localhost:*
https://localhost:* http://127.0.0.1 https://127.0.0.1 http://127.0.0.1:* https://127.0.0.1:*
http://0.0.0.0 https://0.0.0.0 http://0.0.0.0:* https://0.0.0.0:* app://* file://* tauri://*]
OLLAMA_SCHED_SPREAD:false OLLAMA_TMPDIR: ROCR_VISIBLE_DEVICES: http_proxy: https_proxy:
no_proxy:]"
time=2024-09-27T18:51:55.719Z level=INFO source=images.go:753 msg="total blobs: 0"
time=2024-09-27T18:51:55.719Z level=INFO source=images.go:760 msg="total unused blobs removed:
0"
time=2024-09-27T18:51:55.719Z level=INFO source=routes.go:1200 msg="Listening on [::]:11434
(version 0.3.12)"
time=2024-09-27T18:51:55.720Z level=INFO source=common.go:49 msg="Dynamic LLM libraries"
runners="[cpu_avx cpu_avx2 cuda_v11 cuda_v12 cpu]"
time=2024-09-27T18:51:55.720Z level=INFO source=gpu.go:199 msg="looking for compatible GPUs"
time=2024-09-27T18:51:55.942Z level=INFO source=types.go:107 msg="inference compute" id=GPU-
d8c505a1-8af4-7ce4-517d-4f57fa576097 library=cuda variant=v12 compute=7.0 driver=12.2
name="Tesla V100-SXM2-16GB" total="15.8 GiB" available="15.5 GiB"
```

Models

List Models Installed

```
ollama list
```

Load a GGUF model manually

```
ollama create <my-model-name> -f <modelfile>
```

Page Assist

Page Assist is an open-source Chrome Extension that provides a Sidebar and Web UI for your Local AI model.

- Video: This Chrome Extension Surprised Me - YouTube

LM Studio

Discover, download, and run local LLMs.

With LM Studio, you can ...

? - Run LLMs on your laptop, entirely offline
? - Use models through the in-app Chat UI or an OpenAI compatible local server
? - Download any compatible model files from HuggingFace
? repositories
? - Discover new & noteworthy LLMs in the app's home page

URLs

- https://lmstudio.ai/
- Doc: https://lmstudio.ai/docs

OpenLLM

OpenLLM helps developers run any open-source LLMs, such as Llama 2 and Mistral, as OpenAI-compatible API endpoints, locally and in the cloud, optimized for serving throughput and production deployment.

- GitHub: https://github.com/bentoml/OpenLLM
- CoLab: https://colab.research.google.com/github/bentoml/OpenLLM/blob/main/examples/llama2.ipynb

Install

> Recommend using a Python Virtual Environment

```
pip install openllm
```

Start a LLM Server

```
openllm start microsoft/Phi-3-mini-4k-instruct --trust-remote-code
```

To interact with the server, you can visit the web UI at http://localhost:3000/ or send a request using curl. You can also use OpenLLM's built-in Python client to interact with the server:

```
import openllm

client = openllm.HTTPClient('http://localhost:3000')
client.generate('Explain to me the difference between "further" and "farther"')
```

OpenAI Compatible Endpoints

```
import openai

client = openai.OpenAI(base_url='http://localhost:3000/v1', api_key='na')  # Here the server is running on 0.0.0.0:3000

completions = client.chat.completions.create(
```

```
  prompt='Write me a tag line for an ice cream shop.', model=model, max_tokens=64,
stream=stream
)
```

LangChain

```
from langchain.llms import OpenLLMAPI

llm = OpenLLMAPI(server_url='http://44.23.123.1:3000')
llm.invoke('What is the difference between a duck and a goose? And why there are so many Goose
in Canada?')

# streaming
for it in llm.stream('What is the difference between a duck and a goose? And why there are so
many Goose in Canada?'):
  print(it, flush=True, end='')

# async context
await llm.ainvoke('What is the difference between a duck and a goose? And why there are so
many Goose in Canada?')

# async streaming
async for it in llm.astream('What is the difference between a duck and a goose? And why there
are so many Goose in Canada?'):
  print(it, flush=True, end='')
```

Bechmark

Benchmark for LLM engines

bench.py

- ollama ??????? vllm ????????????_ollama vllm-CSDN??
- YT: ollama vs vllm - ??????? ollama ? vllm ?????? - YouTube

```
import aiohttp
import asyncio
import time
from tqdm import tqdm

import random

questions = [
    "Why is the sky blue?", "Why do we dream?", "Why is the ocean salty?", "Why do leaves change color?",
    "Why do birds sing?", "Why do we have seasons?", "Why do stars twinkle?", "Why do we yawn?",
    "Why is the sun hot?", "Why do cats purr?", "Why do dogs bark?", "Why do fish swim?",
    "Why do we have fingerprints?", "Why do we sneeze?", "Why do we have eyebrows?", "Why do we have hair?",
    "Why do we have nails?", "Why do we have teeth?", "Why do we have bones?", "Why do we have muscles?",
    "Why do we have blood?", "Why do we have a heart?", "Why do we have lungs?", "Why do we have a brain?",
    "Why do we have skin?", "Why do we have ears?", "Why do we have eyes?", "Why do we have a nose?",
    "Why do we have a mouth?", "Why do we have a tongue?", "Why do we have a stomach?", "Why do we have intestines?",
    "Why do we have a liver?", "Why do we have kidneys?", "Why do we have a bladder?", "Why do we have a pancreas?",
    "Why do we have a spleen?", "Why do we have a gallbladder?", "Why do we have a thyroid?", "Why do we have adrenal glands?",
    "Why do we have a pituitary gland?", "Why do we have a hypothalamus?", "Why do we have a
```

```
thymus?", "Why do we have lymph nodes?",
    "Why do we have a spinal cord?", "Why do we have nerves?", "Why do we have a circulatory system?", "Why do we have a respiratory system?",
    "Why do we have a digestive system?", "Why do we have an immune system?"
]

async def fetch(session, url):
    """
    参数:
        session (aiohttp.ClientSession): 用于请求的会话。
        url (str): 要发送请求的 URL。

    返回:
        tuple: 包含完成 token 数量和请求时间。
    """
    start_time = time.time()

    # 随机选择一个问题
    question = random.choice(questions) # <--- 这两个必须注释一个

    # 固定问题
    # question = questions[0]             # <--- 这两个必须注释一个

    # 请求的内容
    json_payload = {
        "model": "llama3:8b-instruct-fp16",
        "messages": [{"role": "user", "content": question}],
        "stream": False,
        "temperature": 0.7 # 参数使用 0.7 保证每次的结果略有区别
    }
    async with session.post(url, json=json_payload) as response:
        response_json = await response.json()
        end_time = time.time()
        request_time = end_time - start_time
        completion_tokens = response_json['usage']['completion_tokens'] # 从返回的参数里获取生成的 token 的数量
        return completion_tokens, request_time

async def bound_fetch(sem, session, url, pbar):
    # 使用信号量 sem 来限制并发请求的数量，确保不会超过最大并发请求数
```

```
    async with sem:
        result = await fetch(session, url)
        pbar.update(1)
        return result

async def run(load_url, max_concurrent_requests, total_requests):
    """
    通过发送多个并发请求来运行基准测试。

    参数:
        load_url (str): 要发送请求的URL。
        max_concurrent_requests (int): 最大并发请求数。
        total_requests (int): 要发送的总请求数。

    返回:
        tuple: 包含完成 token 总数列表和响应时间列表。
    """
    # 创建 Semaphore 来限制并发请求的数量
    sem = asyncio.Semaphore(max_concurrent_requests)

    # 创建一个异步的HTTP会话
    async with aiohttp.ClientSession() as session:
        tasks = []

        # 创建一个进度条来可视化请求的进度
        with tqdm(total=total_requests) as pbar:
            # 循环创建任务，直到达到总请求数
            for _ in range(total_requests):
                # 为每个请求创建一个任务，确保它遵守信号量的限制
                task = asyncio.ensure_future(bound_fetch(sem, session, load_url, pbar))
                tasks.append(task)  # 将任务添加到任务列表中

            # 等待所有任务完成并收集它们的结果
            results = await asyncio.gather(*tasks)

        # 计算所有结果中的完成token总数
        completion_tokens = sum(result[0] for result in results)

        # 从所有结果中提取响应时间
        response_times = [result[1] for result in results]
```

```
        # 返回完成token的总数和响应时间的列表
        return completion_tokens, response_times

if __name__ == '__main__':
    import sys

    if len(sys.argv) != 3:
        print("Usage: python bench.py <C> <N>")
        sys.exit(1)

    C = int(sys.argv[1])  # 最大并发数
    N = int(sys.argv[2])  # 请求总数

    # vllm 和 ollama 都兼容了 openai 的 api 让测试变得更简单了
    url = 'http://localhost:11434/v1/chat/completions'

    start_time = time.time()
    completion_tokens, response_times = asyncio.run(run(url, C, N))
    end_time = time.time()

    # 计算总时间
    total_time = end_time - start_time
    # 计算每个请求的平均时间
    avg_time_per_request = sum(response_times) / len(response_times)
    # 计算每秒生成的 token 数量
    tokens_per_second = completion_tokens / total_time

    print(f'Performance Results:')
    print(f'  Total requests            : {N}')
    print(f'  Max concurrent requests   : {C}')
    print(f'  Total time                : {total_time:.2f} seconds')
    print(f'  Average time per request  : {avg_time_per_request:.2f} seconds')
    print(f'  Tokens per second         : {tokens_per_second:.2f}')
```

More

LocalAI

LocalAI is the free, Open Source OpenAI alternative. LocalAI act as a drop-in replacement REST API that's compatible with OpenAI API specifications for local inferencing. It allows you to run LLMs, generate images, audio (and not only) locally or on-prem with consumer grade hardware, supporting multiple model families and architectures.

- Overview | LocalAI documentation
- GitHub: https://github.com/mudler/LocalAI

Xinference

Xorbits Inference (Xinference) is an open-source platform to streamline the operation and integration of a wide array of AI models. With Xinference, you're empowered to run inference using any open-source LLMs, embedding models, and multimodal models either in the cloud or on your own premises, and create robust AI-driven applications.

- Welcome to Xinference! — Xinference
- GitHub: https://github.com/xorbitsai/inference

NVIDIA NIM

Explore the latest community-built AI models with an API optimized and accelerated by NVIDIA, then deploy anywhere with NVIDIA NIM inference microservices.

- NVIDIA NIM for Deploying Generative AI | NVIDIA
- Doc: Introduction - NVIDIA Docs
- Models: google / gemma-7b
- YT: Self-Host and Deploy Local LLAMA-3 with NIMs - YouTube

text-generation-webui

A Gradio web UI for Large Language Models.

> ??????????????? API?

????????? AI ??

- Chat
- Fine-Tune Model
- Multiple model backends: Transformers, llama.cpp (through llama-cpp-python), ExLlamaV2, AutoGPTQ, AutoAWQ, GPTQ-for-LLaMa, QuIP#.
- OpenAI-compatible API server with Chat and Completions endpoints

??

- GitHub: https://github.com/oobabooga/text-generation-webui
- GitHub: https://github.com/Atinoda/text-generation-webui-docker
- ??????LLMs???? ???? (?) - HackMD
  - YOUTUBE [?? TextGen]
  - YOUTUBE [????????]
  - YOUTUBE [??AI??]
  - YOUTUBE [????]
  - YOUTUBE [??????]
  - ??? Z01_TextGen_Colab.ipynb
  - ?????????? (account:nchc password:nchc) ?????

koboldcpp

- GitHub: https://github.com/LostRuins/koboldcpp
- ?????/???/??????
- ?? GGUF ??
- ?? OuteTTS (Text-To-Speech), Whisper (Speech-To-Text), ??/????
- ?? KoboldAI Lite UI

Llama.Cpp

GitHub: https://github.com/ggml-org/llama.cpp

Tutorials

- Windows ?? AI ?????llama.cpp ???? CUDA 13 / Vulkan / HIP / SYCL???? GGUF ?????? - ????