一、 DeepSeek 简介

(一) 什么是 DeepSeek

DeepSeek（中文名“深度求索”）是一家专注通用人工智能（AGI）的中国科技公司，主攻大模型研发与应用，由量化资管巨头“幻方量化”于 2023 年 7 月创立。DeepSeek 也指由 DeepSeek 公司研发的、类似于 ChatGPT、文心一言的智能助手。外界也习惯将该公司开发的一系列大模型产品笼统称为“DeepSeek”。

目前 DeepSeek 主要有两条产品线：

- V 系列：通用对话与内容生成（如 V2、V3），面向文本生成、客服对话等场景；
- R 系列：深度推理与逻辑思维（如 R1），适用于复杂问题求解与科学计算。

(二) DeepSeek 发展历程

DeepSeek 目前已发布 13 个大模型，并全部开源，全球开发者均可利用这些大模型技术开发自己的模型、应用和产品。

表 1 DeepSeek 截至 2025 年 2 月发布的大模型

模型名称	模型类型	发布时间
DeepSeek Coder	代码生成模型	2023 年 11 月 02 日
DeepSeek LLM	通用大语言模型	2023 年 11 月 29 日
DreamCraft3D	文生 3D 模型	2023 年 12 月 18 日
DeepSeek MoE	混合专家模型	2024 年 01 月 11 日
DeepSeek Math	数学推理模型	2024 年 02 月 05 日
DeepSeek-VL	多模态模型	2024 年 03 月 11 日
DeepSeek V2	混合专家模型	2024 年 05 月 07 日

DeepSeek Coder V2	代码生成模型	2024 年 06 月 17 日
DeepSeek-V2.5	融合通用与代码能力模型	2024 年 09 月 06 日
DeepSeek-VL2	多模态混合专家模型	2024 年 12 月 13 日
DeepSeek V3	混合专家模型	2024 年 12 月 26 日
DeepSeek-R1	推理模型	2025 年 01 月 20 日
DeepSeek Janus-Pro	多模态模型	2025 年 01 月 28 日

2024 年 12 月，DeepSeek 最新 V 系列模型--DeepSeek-V3 首个版本上线，并同步开源。DeepSeek-V3 是一个 MoE（Mixture-of-Experts）模型，共有 671B 参数，每个 token 激活的参数量为 37B。为实现高效训练与推理，DeepSeek-V3 延续了 DeepSeek-V2 的 MLA（Multi-head Latent Attention）及 DeepSeek MoE 架构。此外，DeepSeek-V3 首创了无需辅助损失的负载均衡策略，还使用了多 Token 预测训练目标以增强性能。

DeepSeek-V3 多项评测成绩超越 Qwen2.5-72B 和 Llama-3.1-405B 等其他开源模型，并在性能上和世界顶尖的闭源模型 GPT-4o 及 Claude-3.5-Sonnet 不分伯仲。

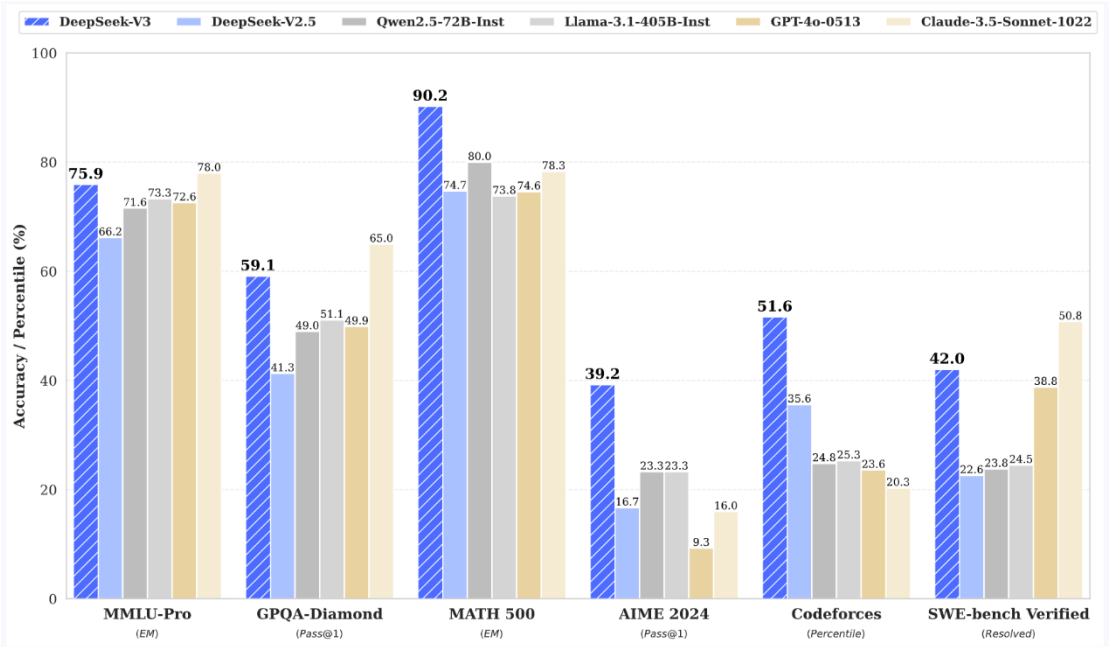


图 1 DeepSeek V3 与其他 Chat 模型开源基准评测效果

2025 年 1 月，DeepSeek 发布了最新 R 系模型--DeepSeek-R1（性能对标 OpenAI-O1 正式版），在数学、编程和逻辑推理方面表现优异。在 AIME（美国数学竞赛）等硬核基准测试中，DeepSeek-R1 超越了 OpenAI-O1 模型，受到了全世界的广泛关注。

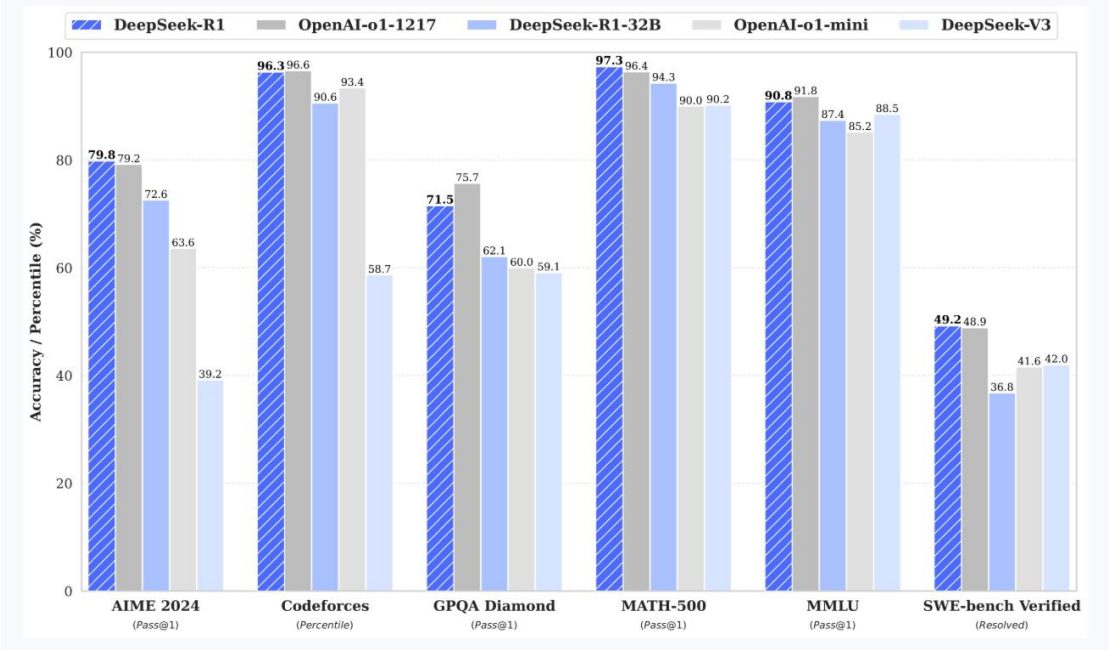


图 2 DeepSeek-R1 与 OpenAI-O1 性能对比

DeepSeek 开发团队在开源 DeepSeek-R1-Zero 和 DeepSeek-R1 两个模型的同时，基于 DeepSeek-R1 的输出，蒸馏了 6 个开源小模型，其中 32B 和 70B 模型在多项能力上实现了对标 OpenAI-o1-mini 的效果。

	AIME 2024 pass@1	AIME 2024 cons@64	MATH-500 pass@1	GPQA Diamond pass@1	LiveCodeBench pass@1	CodeForces rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759.0
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717.0
o1-mini	63.6	80.0	90.0	60.0	53.8	**1820.0**
QwQ-32B	44.0	60.0	90.6	54.5	41.9	1316.0
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954.0
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189.0
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481.0
DeepSeek-R1-Distill-Qwen-32B	**72.6**	83.3	94.3	62.1	57.2	1691.0
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205.0
DeepSeek-R1-Distill-Llama-70B	70.0	**86.7**	**94.5**	**65.2**	**57.5**	1633.0

图 3 DeepSeek 蒸馏小模型等执行不同推理任务性能对比

(三) DeepSeek 影响

DeepSeek 不仅引发了全球新一轮的 AI 应用热潮，而且对全球的算力资本市场产生重大冲击。究其原因，DeepSeek 在训练成本及使用成本、模型训练及优化方式方面均实现了大量工程创新。

DeepSeek 发表的原始报告中指出，DeepSeek-V3 仅使用了 2048 块英伟达 H800 GPU，耗费了 557.6 万美元就完成了训练，相比同等规模的模型（如 GPT-4o、Llama3.1 等），训练成本大幅降低。

DeepSeek 的成功展示了“有限算力+算法创新”的发展模式，为中国 AI 发展提供了宝贵的经验。它证明了在有限算力条件下，通过一系列算法创新同样能够突破算力瓶颈的限制，使开发的大模型获得媲美国际顶尖水平大模型（GPT-4、Claude 等）的性能。

DeepSeek 出圈，很好地证明了我们的竞争优势：通过有限资源的极致高效利用，实现以少胜多。中国与美国在 AI 领域的差距正在缩小。

(四) DeepSeek 项目

在线体验：https://www.deepseek.com

下载地址：https://modelscope.cn/models/deepseek-ai/DeepSeek-R1

二、 DeepSeek-R1 本地部署

(一) 硬件配置要求

DeepSeek-R1 671B 满血版具有 6710 亿参数，这种规模的模型对显存的需求非常高，通常需要多 GPU 分布式推理或高性能计算集群来支持。

DeepSeek-R1 671B 满血版模型文件大小约 671GB（视量化版本而定），需预留充足存储空间。

表 2 DeepSeek-R1 满血版显存需求

模型参数	DeepSeek-R1 671B		
模型显存	671GB	335GB	136GB~226GB

数据精度	FP8	4-bit 量化	1.58-2.51-bit 量化
运行显存最小值（1.3*模型显存）	872.3GB	435.5GB	176GB~239GB

蒸馏后的版本，如 DeepSeek-R1 1.5B/7B/8B 版本，可使用 Ollama 工具在 CPU 上部署运行，但一般仅限于单用户，也可以在消费级 GPU（如 Nvidia RTX 3090、4090）上部署运行，可支持少量并发用户。

对于 DeepSeek-R1 70B 模型，则至少需要 2 张 24G 显存的显卡。如果计算机显存资源不足但内存足够，也可以尝试运行，不过 Ollama 会使用 CPU+GPU 混合推理的模式，运行速度相比单纯的 GPU 模式会下降很多。

表 3 DeepSeek-R1 蒸馏版本硬件配置

设备级别	模型版本	最低配置要求
入门级设备	DeepSeek-R1 1.5B	4GB 内存 + 核显
进阶推荐	DeepSeek-R1 7B	8GB 内存 + 4GB 显存
高性能版本	DeepSeek-R1 32B	32GB 内存 + 12GB 显存
超性能版本	DeepSeek-R1 70B	64GB 内存 + 40GB 显存

(二) Ollama 部署 DeepSeek-R1 蒸馏版

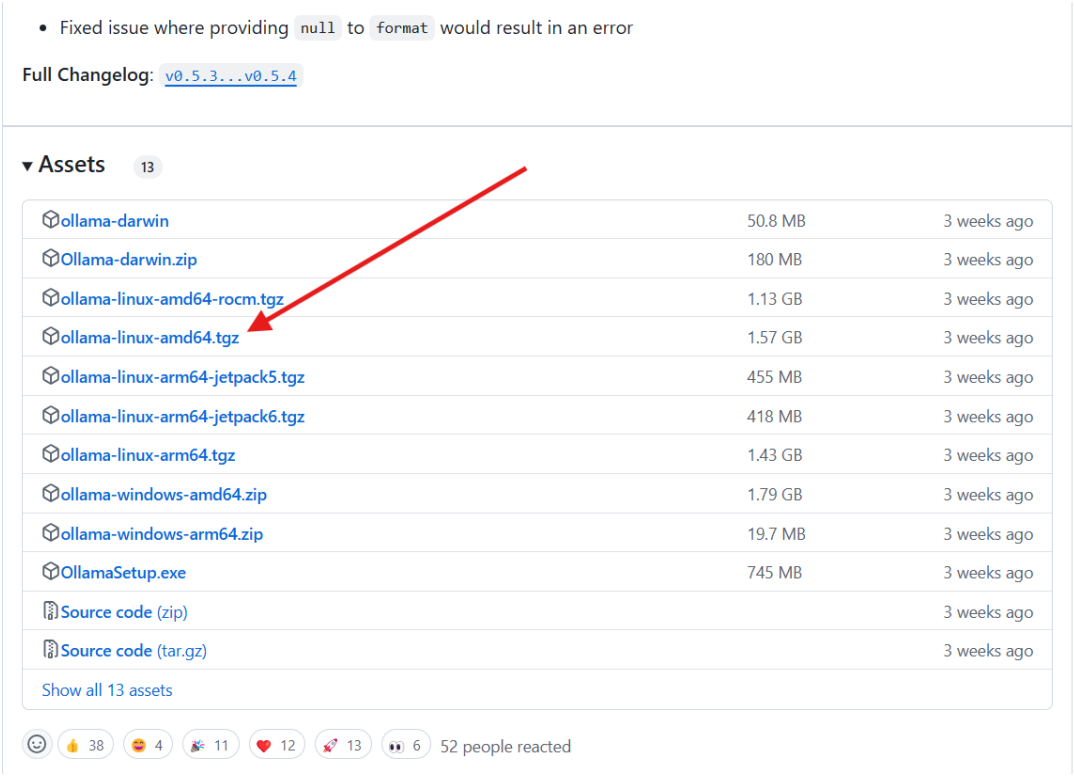
1. 本地计算机硬件配置

CPU：Intel i5-14400F、GPU：Nvidia RTX4060 8G 显存、内存：16GB*2、存储：1TB SSD

2. 安装 Ollama 工具

1) 访问官方网站：https://ollama.com/

下载 Linux 版本 Ollama 工具，打开终端命令窗口，执行：curl -fsSL https://ollama.com/install.sh | sh，如遇无法下载，可通过 GitHub（https://github.com/ollama/ollama/releases）下载最新版本压缩包。



2) 解压安装 Ollama 工具

执行命令：sudo tar -zxf ollama-linux-amd64.tgz -C /usr/local，其中/usr/local 为解压安装目录（若安装至其它目录下，注意环境变量设置）。

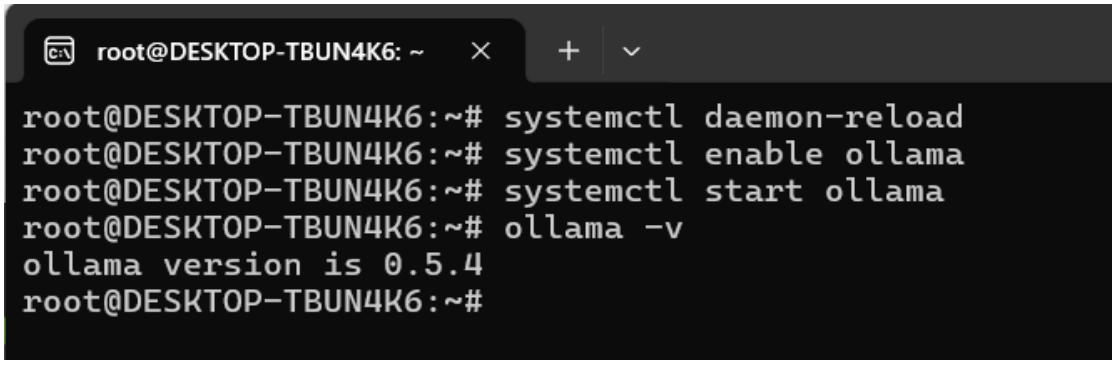
创建服务配置文件：/etc/systemd/system/ollama.service，直接在终端命令窗口中执行命令：sudo vi /etc/systemd/system/ollama.service，输入以下内容。

```
[Unit]
Description=Ollama Service
After=network-online.target

[Service]
Environment="OLLAMA_HOST=0.0.0.0:11434"
ExecStart=/usr/local/bin/ollama serve
User=root
Group=root
Restart=always
RestartSec=3
# 指定模型保存路径
Environment="OLLAMA_MODELS=/home/ghang/ollama/models"
# 多卡环境可以指定运行的GPU
# Environment="CUDA_VISIBLE_DEVICES=0,1"

[Install]
WantedBy=default.target
```

保存 ollama.service 成功后，执行以下命令，使 Ollama 服务生效。



3. 运行 DeepSeek-R1 7B

Ollama 下载并运行 DeepSeek-R1 7B 模型。Ollama 默认下载 Q4_K_M 量化版本，模型文件大约 4.7GB。

(三) vLLM 部署 DeepSeek-R1 满血版

1. 服务器硬件配置

- 服务器 1：CPU：英特尔至强 Max 9468*2、GPU：HGX H20(96GB)*8、内存：64GB*32、存储：3.84T Nvme*4
- 服务器 2：CPU：英特尔至强 Max 9468*2、GPU：HGX H20(96GB)*8、内存：64GB*32、存储：3.84T Nvme*4
- 网络：25Gb 以太组网

2. vLLM 安装

安装 docker 镜像，在 Linux 终端中执行命令：docker pull vllm/vllm-openai:latest，拉取镜像。如遇下载失败，可转至国内镜像源，例如：sudo docker pull docker.1ms.run/vllm/vllm-openai:latest

```
GongHang@server01:~$ sudo docker pull docker.1ms.run/vllm/vllm-openai:latest
latest: Pulling from vllm/vllm-openai
aece8493d397: Downloading  14.01MB
45f7ea5367fe: Waiting
3d97a47c3c73: Waiting
12cd4d19752f: Waiting
da5a484f9d74: Waiting
5e5846364eee: Waiting
fd355de1d1f2: Waiting
3480bb79c638: Waiting
e7016935dd60: Waiting
99541166a133: Waiting
8999112df5b0: Pulling fs layer
8ec811e9882b: Waiting
30a93cde00b2: Waiting
8d8eac6a445f: Waiting
2d64baf0cbf8: Waiting
6988853188d9: Waiting
4f4fb700ef54: Waiting
8131de6930aa: Waiting
c58b91a22c96: Pulling fs layer
b2df723b7287: Waiting
81937065d286: Waiting
ad261cc98b94: Waiting
d3fb3448da6e: Waiting
```

3. 创建 Ray 集群

1) 创建 vllm-openai 容器，然后进入容器 bash 窗口，通过 Ray 指令创建集群。

如有必要，请指定 GLOO 与 NCCL 通信网卡。

```
GongHang@server03:~$ sudo docker run -itd --gpus all --network host --ipc host --name vllm-openai --rm -v /
home/GongHang/models:/workspace/models --entrypoint /bin/bash docker.1ms.run/vllm/vllm-openai:latest
8d4f4bc3b21fcae90bae5b3593183662509c16951cc34583d7ea6c6591ee103a
GongHang@server03:~$ sudo docker ps
CONTAINER ID   IMAGE                                  COMMAND       CREATED         STATUS         PORTS
    NAMES
8d4f4bc3b21f   docker.1ms.run/vllm/vllm-openai:latest   "/bin/bash"   7 seconds ago   Up 6 seconds
    vllm-openai
GongHang@server03:~$ sudo docker exec -it 8d4f4bc3b21f bash
root@server03:/vllm-workspace# ray start --head --node-ip-address='10.0.10.13'
Enable usage stats collection? This prompt will auto-proceed in 10 seconds to avoid blocking cluster startu
p. Confirm [Y/n]: y
Usage stats collection is enabled. To disable this, add `--disable-usage-stats` to the command that starts
the cluster, or run the following command: `ray disable-usage-stats` before starting the cluster. See https
://docs.ray.io/en/master/cluster/usage-stats.html for more details.

Local node IP: 10.0.10.13

--------------------
Ray runtime started.
--------------------
```

- 在服务器 1 中创建 head 节点：ray start --head --node-ip-address=’本机 IP’
- 在服务器 2 中创建 worker 节点：ray start --address='head 节点 IP:6379' --node-ip-address=’本机 IP’

2) 创建 run_cluster.sh 脚本文件，使用脚本文件初始化 Ray 节点（建议使用）。

- 在服务器 1 中创建 head 节点：sudo bash run_cluster.sh 本机 IP –head

- 在服务器 2 中创建 worker 节点：sudo bash run_cluster.sh 主节点 IP --worker

```
#!/bin/bash

if [ $# -lt 2 ]; then
    echo "Usage: $0 head_node_address --head|--worker [additional_args...]"
    exit 1
fi

HEAD_NODE_ADDRESS="$1"
NODE_TYPE="$2"
shift 2

ADDITIONAL_ARGS=("$@")

if [ "${NODE_TYPE}" != "--head" ] && [ "${NODE_TYPE}" != "--worker" ]; then
    echo "Error: Node type must be --head or --worker"
    exit 1
fi

cleanup() {
    if [ -e /var/lib/docker/containers/vllm ]; then
        docker stop vllm
        docker rm vllm
    fi
}
trap cleanup EXIT

RAY_START_CMD="ray start --block"
if [ "${NODE_TYPE}" == "--head" ]; then
    RAY_START_CMD+=" --head --port=6379"
else
    RAY_START_CMD+=" --address=${HEAD_NODE_ADDRESS}:6379"
fi
# 执行docker命令
docker run -itd \
    --privileged \
    --entrypoint /bin/bash \
    --ipc host \
    --network host \
    --name vllm \
    --gpus all \
    --rm \
    -v /data/models:/mnt \
    "${ADDITIONAL_ARGS[@]}" \
    -e GLOO_SOCKET_IFNAME=bond0 \
    -e NCCL_SOCKET_IFNAME=bond0 \
    docker.1ms.run/vllm/vllm-openai:latest \
    -c "${RAY_START_CMD}"
```

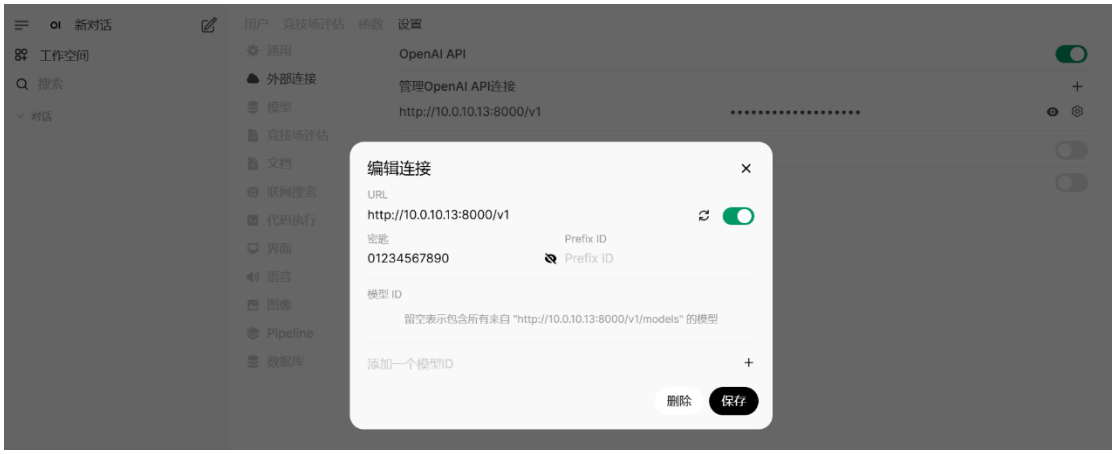
4. 启动 vLLM 服务

```
root@server03:/vllm-workspace# vllm serve /mnt/DeepSeek-R1-671B --host 0.0.0.0 --port 8000 --api-key 012345678
90 --gpu_memory-utilization 0.9 --tensor-parallel-size 8 --pipeline-parallel-size 2 --trust-remote-code --serv
ed-model-name DeepSeek-R1-671B --disable-log-requests
```

```
INFO 04-06 21:04:35 [api_server.py:1028] Starting vLLM API server on http://0.0.0.0:8000
INFO 04-06 21:04:35 [launcher.py:26] Available routes are:
INFO 04-06 21:04:35 [launcher.py:34] Route: /openapi.json, Methods: GET, HEAD
INFO 04-06 21:04:35 [launcher.py:34] Route: /docs, Methods: GET, HEAD
INFO 04-06 21:04:35 [launcher.py:34] Route: /docs/oauth2-redirect, Methods: GET, HEAD
INFO 04-06 21:04:35 [launcher.py:34] Route: /redoc, Methods: GET, HEAD
INFO 04-06 21:04:35 [launcher.py:34] Route: /health, Methods: GET
INFO 04-06 21:04:35 [launcher.py:34] Route: /load, Methods: GET
INFO 04-06 21:04:35 [launcher.py:34] Route: /ping, Methods: GET, POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /tokenize, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /detokenize, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /v1/models, Methods: GET
INFO 04-06 21:04:35 [launcher.py:34] Route: /version, Methods: GET
INFO 04-06 21:04:35 [launcher.py:34] Route: /v1/chat/completions, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /v1/completions, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /v1/embeddings, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /pooling, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /score, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /v1/score, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /v1/audio/transcriptions, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /rerank, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /v1/rerank, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /v2/rerank, Methods: POST
INFO 04-06 21:04:35 [launcher.py:34] Route: /invocations, Methods: POST
INFO:     Started server process [519]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

5. 验证 vLLM 服务

1) 在 Open-WebUI（WebUI 安装过程，请参考官方教程）管理员界面中，设置 OpenAI API 外部连接。



2) 创建聊天窗口，选择 DeepSeek-R1-671B 模型。


新对话
工作空间
搜索
对话
今天
新对话
yjy
DeepSeek-R1-671B
你是谁?
DeepSeek-R1-671B
我是DeepSeek-R1，一个由深度求索公司开发的智能助手，我擅长通过思考来帮您解答复杂的数学，代码和逻辑推理等理工类问题。
</think>
我是DeepSeek-R1，一个由深度求索公司开发的智能助手，我擅长通过思考来帮您解答复杂的数学，代码和逻辑推理等理工类问题。
输入消息
代码解释器