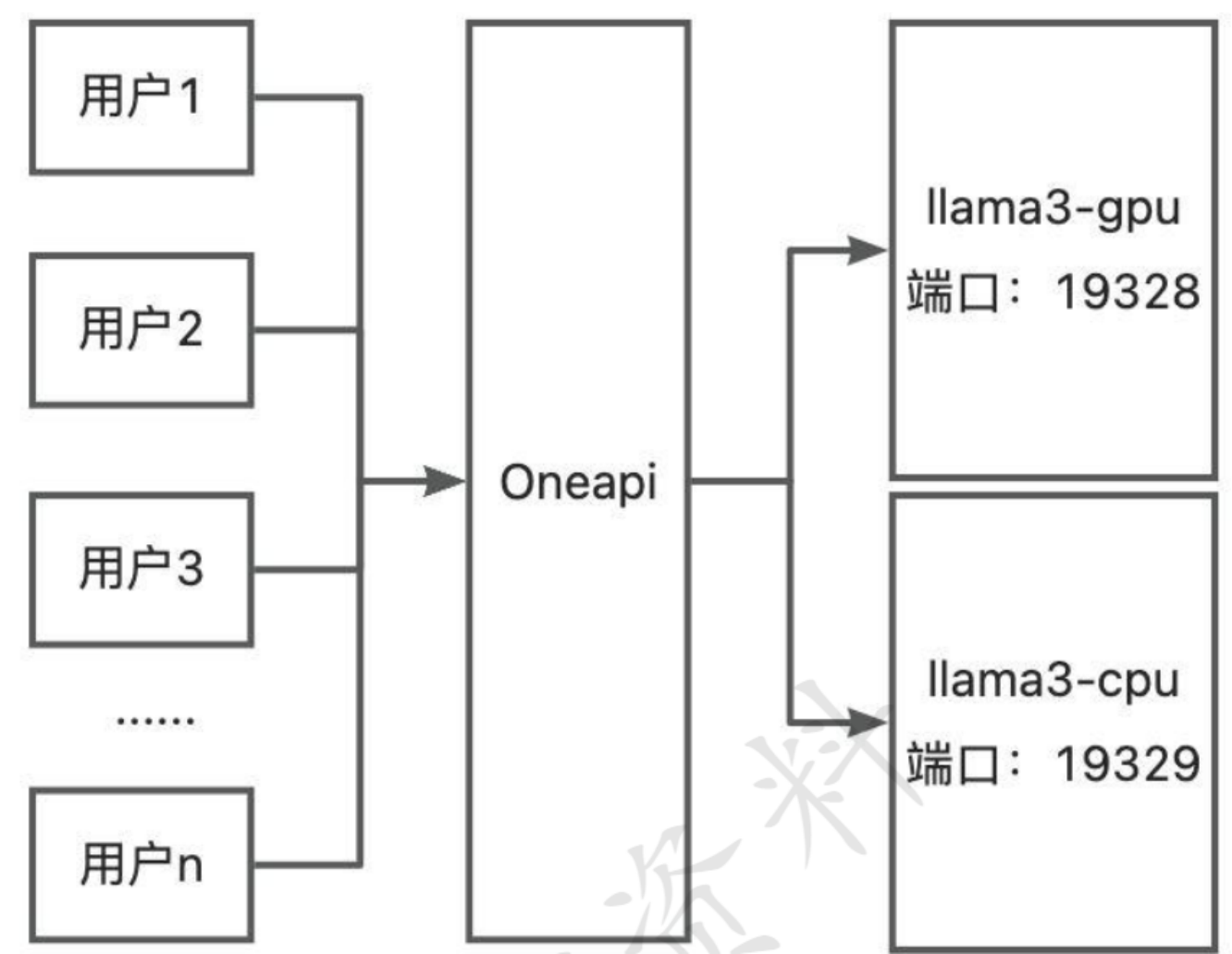


算力平台部署私有化LLM仿OpenAI API接口的高可用工程实践报告

本次课基于Chinese-LLaMA-Alpaca-3(<https://github.com/ymcui/Chinese-LLaMA-Alpaca-3>)项目在本本地Windows/Linux上封装一个大模型接口。



大模型聊天客户端

我们使用ChatGPTNextWeb工具测试我们的接口，如果您没有下载客户端，可以通过下面地址下载：https://github.com/ChatGPTNextWeb/ChatGPT-Next-Web/releases/download/v2.14.2/NextChat_2.14.2_x64-setup.exe

租用实例

在AutoDL上租用一台24G显存以上的服务器

内蒙B区 / 075机
7e24f8c67-ebdb2e4b
设置名称

运行中

无卡模式

RTX 4090 * 1卡

查看详情

系统盘 79.05%
数据盘 0.01%

正常

按量计费

关机15天后释放
设置定时关机

登录指令
ssh*****
密码

JupyterLab
AutoPanel
实例监控
自定义服务

关机

更多

镜像版本

The screenshot shows the 'Image Version' selection interface in the Alibaba Cloud console. A dropdown menu is open, displaying a list of frameworks and their corresponding Python and CUDA versions. The selected version is 'PyTorch / 2.1.2 / 3.10(ubuntu22.04) / 11.8'.

框架名称	Python版本	Cuda版本
PyTorch	3.10(ubuntu22.04)	11.8
TensorFlow		
Miniconda		
JAX		
PaddlePaddle		
TensorRT		
Gromacs		
Jittor		

创建完成后仍然可以更换其他镜像

优惠券: 请选择

日常费用: ¥0.00/日 配置费用: ¥2.08/时 费用明细

账户余额: [余额图标]

取消 立即创建

进入JupyterLab，打开终端

The screenshot shows the JupyterLab interface. The left sidebar displays a file explorer with a list of files and folders. The main area shows the '启动页' (Start Page) with various icons for creating new notebooks, controlling the environment, and other tools. The '终端' (Terminal) icon is highlighted with a red box.

文件 编辑 查看 运行 内核 标签页 设置 帮助

按文件名过滤

名称 已修改

- autodl-fs 19小时前
- autodl-pub 19小时前
- autodl-tmp 19小时前
- Chinese-LLaMA-Alpaca-3-3.0 19小时前
- miniconda3 6个月前
- tf-logs 19小时前
- Chinese-LLaMA-Alpaca-3-3... 4个月前

启动页

笔记本

Python 3 (ipykernel)

控制台

Python 3 (ipykernel)

其他

终端

文本文件

Markdown 文件

Python 文件

显示上下文帮助

简易界面 2 0 0

启动页 1

部署大模型

下载源码压缩包并解压

```
cd ~/ && wget https://file.huishiwei.top/Chinese-LLaMA-Alpaca-3-3.0.tar.gz
tar -xvf Chinese-LLaMA-Alpaca-3-3.0.tar.gz
```

autodl-fs	19小时前
autodl-pub	19小时前
autodl-tmp	19小时前
Chinese-LLaMA-Alpaca-3-3.0	19小时前
miniconda3	6个月前
tf-logs	19小时前
Chinese-LLaMA-Alpaca-3-3....	4个月前

创建相应的虚拟环境

```
create conda -n LLM python=3.8.17
```

激活虚拟环境并下载模型权重文件

```
pip install modelscope -i https://mirrors.aliyun.com/pypi/simple  
modelscope download --model ChineseAlpacaGroup/llama-3-chinese-8b-instruct-v3
```

模型存储位置查看

```
ls -alhrt ~/.cache/modelscope/hub/ChineseAlpacaGroup/llama-3-chinese-8b-instruct-v3
```

推理脚本BUG修复

/ ... / scripts / oai_api_demo /

名称

已修改

openai_api_protocol.py

7个月前

openai_api_server.py

19小时前

README.md

7个月前

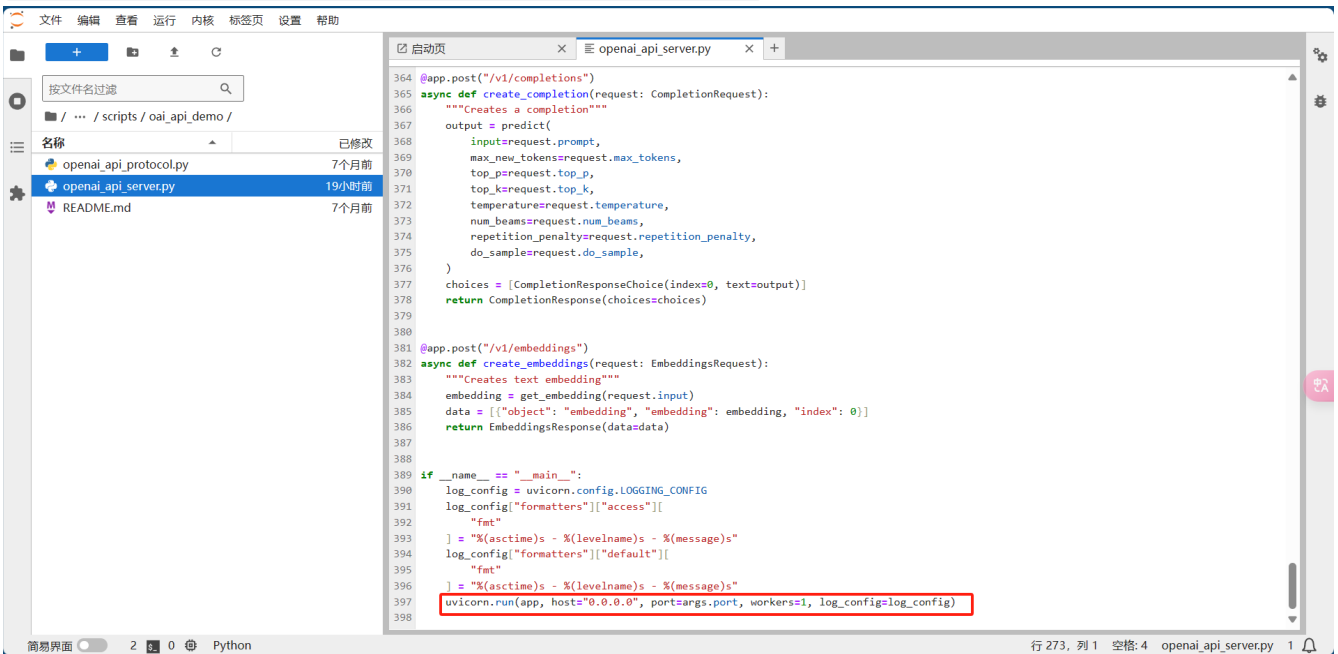
```
259 num_beams=num_beams,
260 do_sample=do_sample,
261 **kwargs,
262 )
263
264 streamer = TextIteratorStreamer(tokenizer, skip_prompt=True, skip_special_tokens=True)
265 generation_kwargs = dict(
266     streamer=streamer,
267     input_ids=input_ids,
268     generation_config=generation_config,
269     return_dict_in_generate=True,
270     output_scores=False,
271     max_new_tokens=max_new_tokens,
272     repetition_penalty=float(repetition_penalty),
273     pad_token_id=tokenizer.eos_token_id,
274     eos_token_id=[tokenizer.eos_token_id, tokenizer.convert_tokens_to_ids("<|eot_id|>")]
275 )
276 Thread(target=model.generate, kwargs=generation_kwargs).start()
277 for new_text in streamer:
278     choice_data = ChatCompletionResponseStreamChoice(
279         index=0, delta=DeltaMessage(content=new_text), finish_reason=None
280     )
281     chunk = ChatCompletionResponse(
282         model=model_id, choices=[choice_data], object="chat.completion.chunk"
283     )
284     yield "{}".format(chunk.json(exclude_unset=True, ensure_ascii=False))
285 choice_data = ChatCompletionResponseStreamChoice(
286     index=0, delta=DeltaMessage(), finish_reason="stop"
287 )
288 chunk = ChatCompletionResponse(
289     model=model_id, choices=[choice_data], object="chat.completion.chunk"
290 )
291 yield "{}".format(chunk.json(exclude_unset=True, ensure_ascii=False))
292 yield "[DONE]"
293
294
```

```
pad_token_id=tokenizer.eos_token_id,
eos_token_id=[tokenizer.eos_token_id, tokenizer.convert_tokens_to_ids("<|eot_id|>")]
```

添加端口参数

```
1 import argparse
2 import os
3 from fastapi import FastAPI
4 from fastapi.middleware.cors import CORSMiddleware
5 import uvicorn
6 from threading import Thread
7 from sse_starlette.sse import EventSourceResponse
8
9 parser = argparse.ArgumentParser()
10 parser.add_argument("--port", default=19328, type=int) # 新添加一行
11 parser.add_argument("--base_model", default=None, type=str, required=True)
12 parser.add_argument("--lora_model", default=None, type=str, help="If None, perform inference on the base model")
13 parser.add_argument("--tokenizer_path", default=None, type=str)
14 parser.add_argument("--gpu", default="0", type=str)
15 parser.add_argument("--load_in_8bit", action="store_true", help="Load the model in 8bit mode")
16 parser.add_argument("--load_in_4bit", action="store_true", help="Load the model in 4bit mode")
17 parser.add_argument("--only_cpu", action="store_true", help="Only use CPU for inference")
18 parser.add_argument("--use_flash_attention_2", action="store_true", help="Use flash-attention2 to accelerate inference")
19 args = parser.parse_args()
20 if args.only_cpu is True:
21     args.gpu = ""
22     if args.load_in_8bit or args.load_in_4bit:
23         raise ValueError("Quantization is unavailable on CPU.")
24 if args.load_in_8bit and args.load_in_4bit:
25     raise ValueError("Only one quantization method can be chosen for inference. Please check your arguments")
26 os.environ["CUDA_VISIBLE_DEVICES"] = args.gpu
27
28 import torch
29 import torch.nn.functional as F
30 from transformers import (
31     AutoModelForCausalLM,
32     AutoTokenizer,
33     GenerationConfig,
34     TextIteratorStreamer,
35     BitsAndBytesConfig
36 )
```

```
parser.add_argument('--port', default=19328, type=int)
```



```
uvicorn.run(app, host="0.0.0.0", port=args.port, workers=1, log_config=log_config)
```

安装依赖

```
accelerate==0.30.0
aiohttp==3.9.5
aiosignal==1.3.1
annotated-types==0.6.0
anyio==4.3.0
async-timeout==4.0.3
attrs==23.2.0
bitsandbytes==0.43.1
certifi==2024.2.2
charset-normalizer==3.3.2
click==8.1.7
datasets==2.20.0
deepspeed==0.13.1
dill==0.3.7
dnspython==2.6.1
einops==0.8.0
email_validator==2.1.1
exceptiongroup==1.2.1
fastapi==0.109.2
fastapi-cli==0.0.3
filelock==3.14.0
frozenlist==1.4.1
fsspec==2023.10.0
h11==0.14.0
hjson==3.1.0
httpcore==1.0.5
httptools==0.6.1
httpx==0.27.0
huggingface-hub==0.23.3
idna==3.7
```

Jinja2==3.1.4
joblib==1.4.2
markdown-it-py==3.0.0
MarkupSafe==2.1.5
mdurl==0.1.2
modelscope==1.17.1
mpmath==1.3.0
multidict==6.0.5
multiprocess==0.70.15
networkx==3.1
ninja==1.11.1.1
numpy==1.24.4
nvidia-cublas-cu12==12.1.3.1
nvidia-cuda-cupti-cu12==12.1.105
nvidia-cuda-nvrtc-cu12==12.1.105
nvidia-cuda-runtime-cu12==12.1.105
nvidia-cudnn-cu12==8.9.2.26
nvidia-cufft-cu12==11.0.2.54
nvidia-curand-cu12==10.3.2.106
nvidia-cusolver-cu12==11.4.5.107
nvidia-cuspars-cu12==12.1.0.106
nvidia-nccl-cu12==2.18.1
nvidia-nvjitlink-cu12==12.4.127
nvidia-nvtx-cu12==12.1.105
orjson==3.10.3
packaging==24.0
pandas==2.0.3
peft==0.7.1
psutil==5.9.8
py-cpuinfo==9.0.0
pyarrow==16.0.0
pyarrow-hotfix==0.6
pydantic==1.10.11
pydantic_core==2.18.2
Pygments==2.18.0
pynvml==11.5.0
python-dateutil==2.9.0.post0
python-decouple==3.8
python-dotenv==1.0.1
python-multipart==0.0.9
pytz==2024.1
PyYAML==6.0.1
regex==2024.4.28
requests==2.32.3
rich==13.7.1
safetensors==0.4.3
scikit-learn==1.3.2
scipy==1.10.1
shellingham==1.5.4
shortuuid==1.0.13
six==1.16.0
sniffio==1.3.1
sse-starlette==2.1.0

```
starlette==0.36.3
sympy==1.12
threadpoolctl==3.5.0
tokenizers==0.19.1
torch==2.1.2
tqdm==4.66.4
transformers==4.41.2
triton==2.1.0
typer==0.12.3
typing_extensions==4.11.0
tzdata==2024.1
ujson==5.9.0
urllib3==2.2.1
uvicorn==0.29.0
uvloop==0.19.0
watchfiles==0.21.0
websockets==12.0
xxhash==3.4.1
yarl==1.9.4
```

启动模型服务 GPU版本

```
cd ~/Chinese-LLaMA-Alpaca-3-3.0/scripts/oai_api_demo/

python openai_api_server.py --gpus 0 --base_model
/home/llm_course/.cache/modelscope/hub/ChineseAlpacaGroup/llama-3-chinese-8b-instruct-v3
```

启动模型服务 CPU版本

```
python openai_api_server.py --only_cpu --port 19329 --base_model
/home/llm_course/.cache/modelscope/hub/ChineseAlpacaGroup/llama-3-chinese-8b-instruct-v3
```

安装Docker

Ubuntu版本

```
# 移除现有的 Docker 相关组件
apt remove docker-ce docker-ce-cli containerd.io docker-compose-plugin docker docker-engine
docker.io containerd runc

# 替换 Ubuntu 的官方源为阿里云源
sed -i -r 's#http://(archive|security).ubuntu.com#https://mirrors.aliyun.com#g'
/etc/apt/sources.list && apt update -y

# 安装 Docker 所需的依赖包
apt install ca-certificates curl gnupg lsb-release -y

# 下载 Docker 的 GPG 密钥, 并将其添加到 apt-key 中
curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | apt-key add -
```

```
# 添加 Docker 的阿里云软件源
add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu
$(lsb_release -cs) stable"

# 更新系统的软件包
apt -y update

# 安装 Docker
apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin -y
```

CentOS版本

```
# 移除现有的 Docker 相关组件
sudo yum remove docker-ce docker-ce-cli containerd.io docker-compose-plugin docker docker-
engine docker.io containerd runc

# 修改 CentOS 的包管理器源为阿里云镜像
sudo sed -i 's|^mirrorlist=|#mirrorlist=|g' /etc/yum.repos.d/CentOS-Base.repo
sudo sed -i 's|^#baseurl=http://mirror.centos.org|baseurl=https://mirrors.aliyun.com|g'
/etc/yum.repos.d/CentOS-Base.repo
sudo yum makecache

# 安装 Docker 所需的依赖包
sudo yum install -y yum-utils device-mapper-persistent-data lvm2

# 添加 Docker 的阿里云软件源
sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-
ce.repo

# 更新系统的软件包
sudo yum update -y

# 安装 Docker
sudo yum install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

启动Docker

```
sudo systemctl start docker
sudo systemctl enable docker
```

部署oneapi

创建oneapi目录并创建docker-compose.yaml文件

```
mkdir -p ~/oneapi-compose
cd ~/oneapi-compose
touch docker-compose.yaml
```

在docker-compose.yaml文件中添加如下内容

```
services:
  oneapi:
    image: m.daocloud.io/ghcr.io/songquanpeng/one-api:v0.6.7
    container_name: oneapi
    restart: always
    ports:
      - 3030:3000
    networks:
      - oneapi_llm_course
    environment:
      - TZ=Asia/Shanghai
    volumes:
      - ./data:/data
networks:
  oneapi_llm_course:
```

启动服务

使用如下命令启动容器：docker compose up -d

(第一次运行时会从镜像仓库拉取)

访问oneapi服务

http://<ip地址>:3030

登录并修改密码

默认用户名：root

默认密码：123456

添加渠道



更新渠道信息

类型 *

自定义渠道

1. 选择自定义渠道

Base URL

http://192.168.9.96:19327

2. 输入渠道URL

名称 *

llama3-gpu

3. 输入渠道名称

分组 *

default

模型 *

llama-3-chinese

4. 输入自定义模型名称：llama-3-chinese，然后点击“填入”

填入相关模型

填入所有模型

清除所有模型

输入自定义模型名称

填入

模型重定向

此项可选，用于修改请求体中的模型名称，为一个 JSON 字符串，键为请求中模型名称，值为要替换的模型名称，例如：

```
{
  "gpt-3.5-turbo-0301": "gpt-3.5-turbo",
  "gpt-4-0314": "gpt-4",
  "gpt-4-32k-0314": "gpt-4-32k"
}
```

密钥 *

请输入渠道对应的密钥

5. 密钥可以随便填入，比如x，填入后点击提交即可

取消

提交

One API 由 JustSong 构建，源代码遵循 MIT 协议

添加令牌

我的令牌

1. 点击“令牌”

搜索令牌名称...

名称	状态	已用额度	剩余额度	创建时间	过期时间	操作
----	----	------	------	------	------	----

llama3-gpu	启用	0%	100%	2023-10-27 10:00	2023-11-27 10:00	复制 聊天 删除 禁用 编辑
------------	----	----	------	------------------	------------------	----------------

添加新的令牌

刷新

添加方式

2. 点击“添加新令牌”

< (1) >

One API 由 JustSong 构建，源代码遵循 MIT 协议

创建新的令牌

名称*

NextChatToken

1. 输入令牌名称

模型范围

llama-3-chinese X

2. 输入模型范围llama-3-chinese

IP 限制

请输入允许访问的网段，例如：192.168.0.0/24，请使用英文逗号分隔多个网段

过期时间

年/月/日 --:--

3. 点击“永不过期”

永不过期

一个月后过期

一天后过期

一小时后过期

一分钟后过期

注意，令牌的额度仅用于限制令牌本身的最大额度使用量，实际的使用受到账户的剩余额度限制。

额度 (等价金额: \$1.00)

500000

取消无额度

4. 点击“设为无限额度”，然后点击“提交即可”

取消

提交

One API 由 JustSong 构建，源代码遵循 MIT 协议

创建隧道

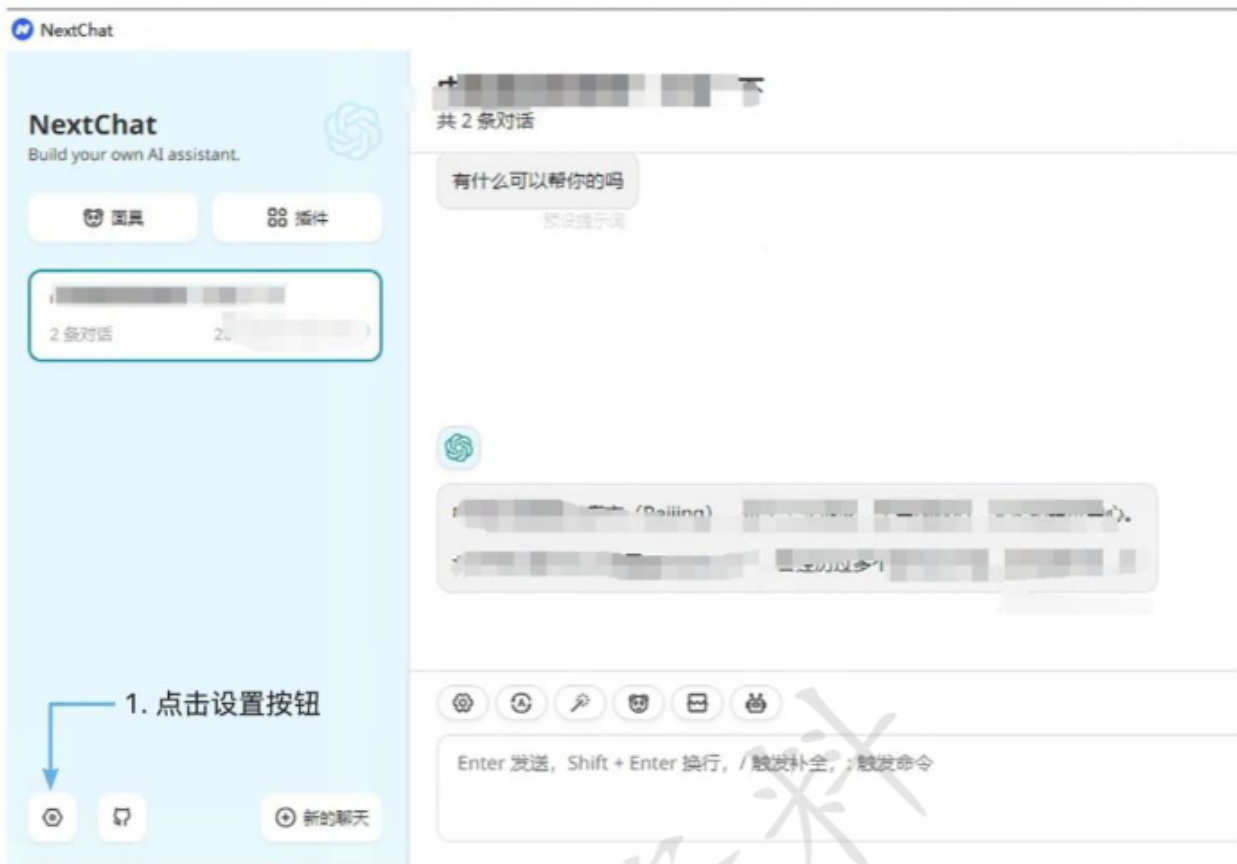
使用虚拟机与算力平台(私有服务器)建立SSH隧道

```
ssh -CNg -L 19328:127.0.0.1:19328 用户名@服务器域名或IP地址 -p 28930
```

-p后面是端口号，默认为22

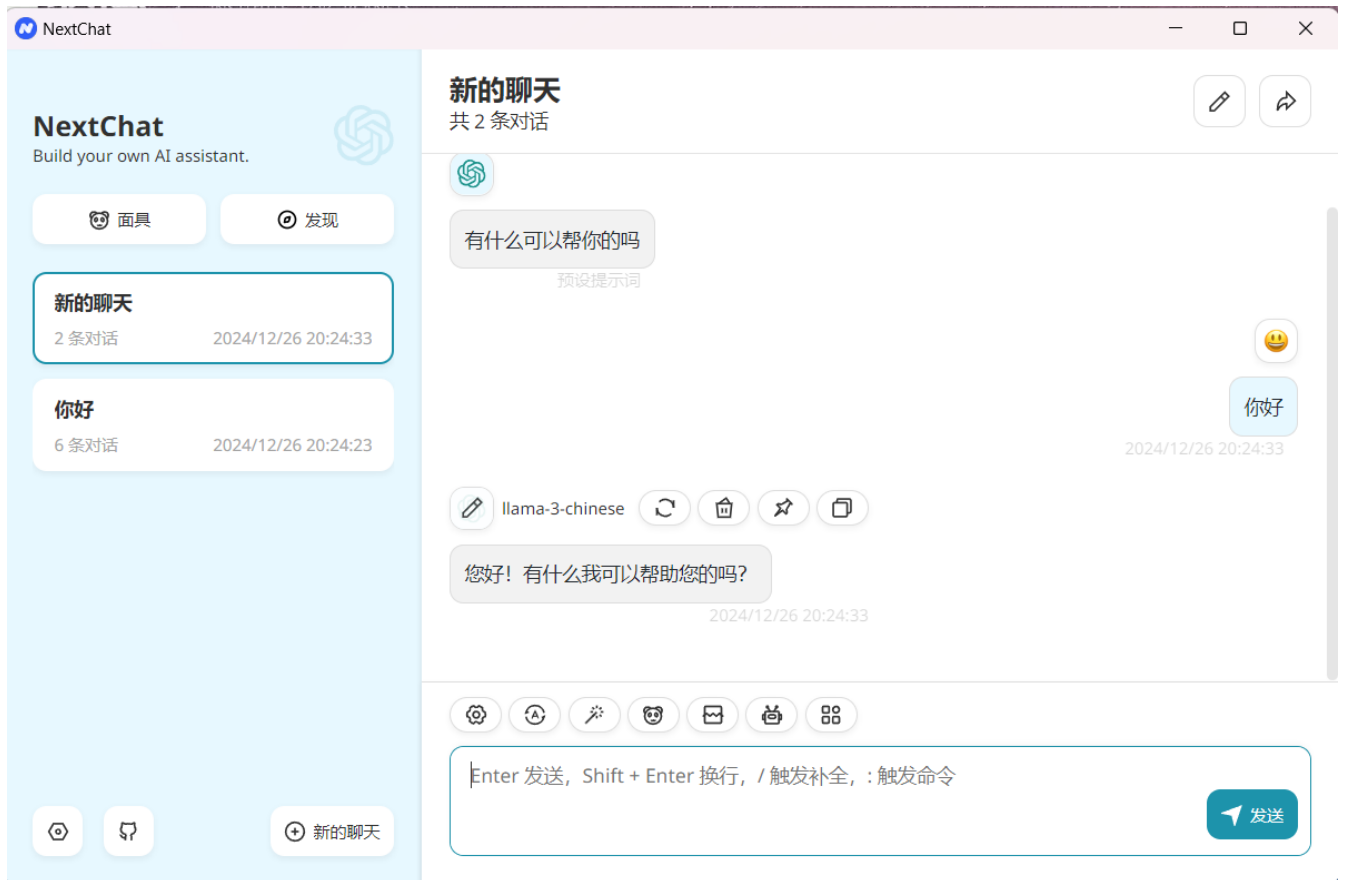
测试大模型

配置ChatGPTNextWeb



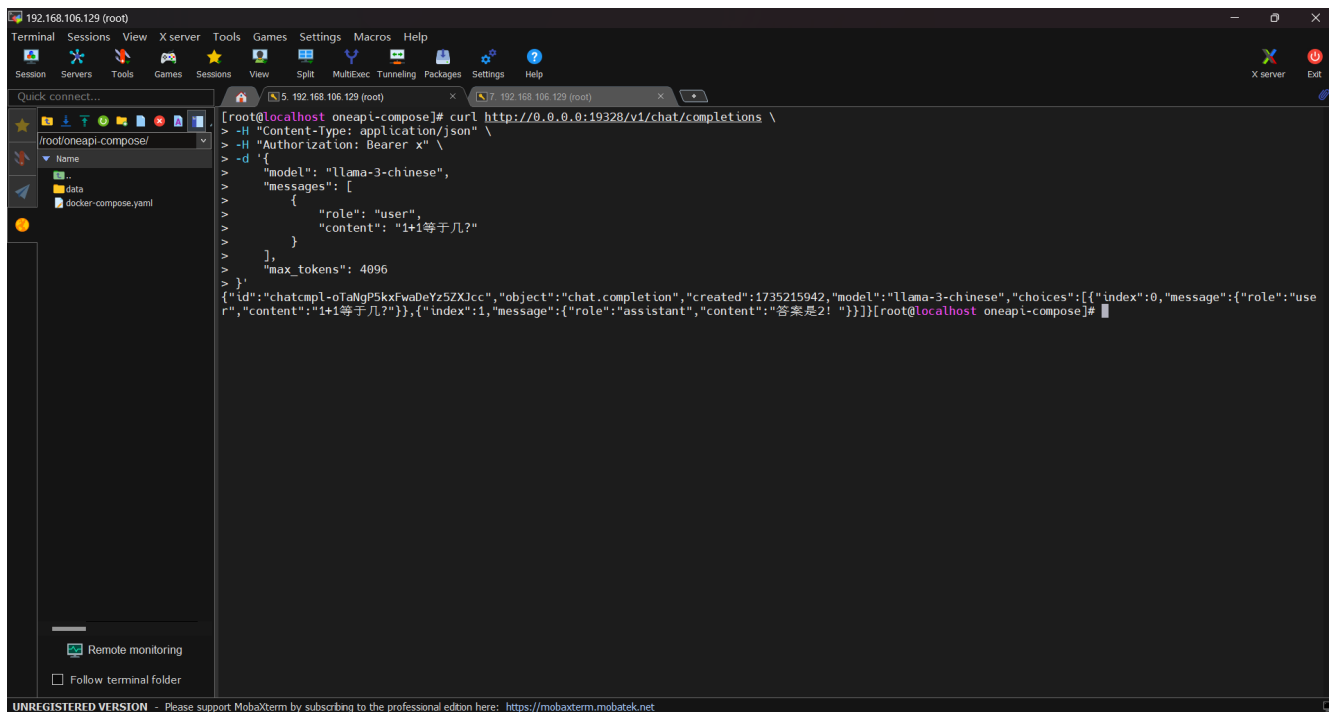
模型服务商 切换不同的服务商	注意地址，要写入oneapi地址	OpenAI
接口地址 除默认地址外，必须包含 http(s)://	1. 填入oneapi地址	http://192.168.9.96:3030
API Key 使用自定义 OpenAI Key 绕过密码访问限制	2. 填入在上一章节获取的token	
自定义模型名 增加自定义模型可选项，使用英文逗号隔开	3. 填入自定义模型名：llama-3-chinese	llama-3-chinese
模型 (model)	4. 选择刚才填入的自定义模型 llama-3-chinese(llama-3-chinese)	llama-3-chinese(llama-3-chinese)

可视化测试



命令行测试

```
curl http://0.0.0.0:19328/v1/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer x" \
-d '{
  "model": "llama-3-chinese",
  "messages": [
    {
      "role": "user",
      "content": "1+1等于几?"
    }
  ],
  "max_tokens": 4096
}'
```



常见问题

oneapi 怎么做到高可用

以火山引擎为例，我们可以在多台 ECS(云服务器)中部署 oneapi，然后设置一个域名映射到多个 ip 即可：

基本信息

- * 域名 .ai ?
- * 记录类型 v
- * TTL v
- * 线路 v
- * 负载均衡 ? ☒ 关闭 ☐ 开启

记录值 (2)

填入多台服务器ip即可

☐	记录值	是否启用	备注	操作
☐	请输入记录值	☒	请输入备注 0/16	删除
☐	请输入记录值	☒	请输入备注 0/16	删除

添加记录值