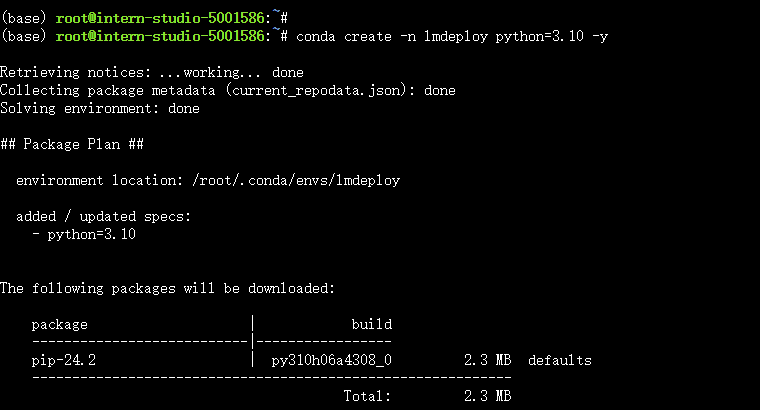
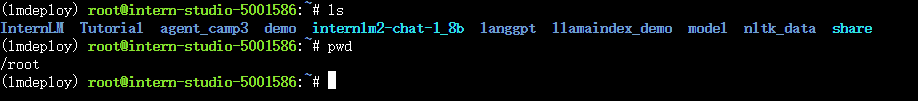
配置LMDeploy环境

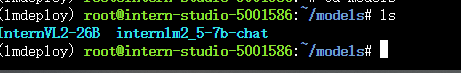






**1.2 InternStudio环境获取模型**

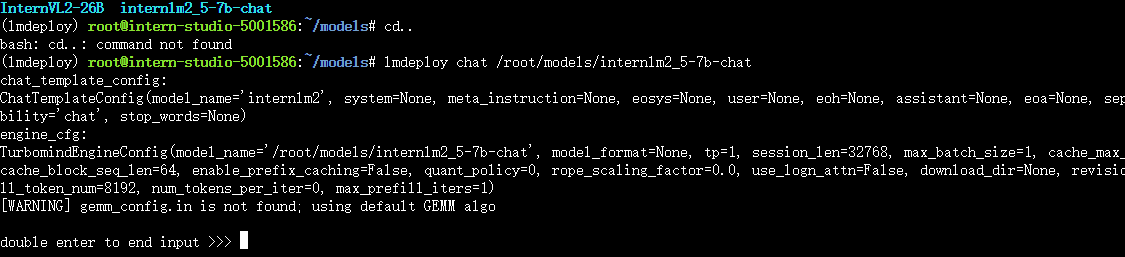


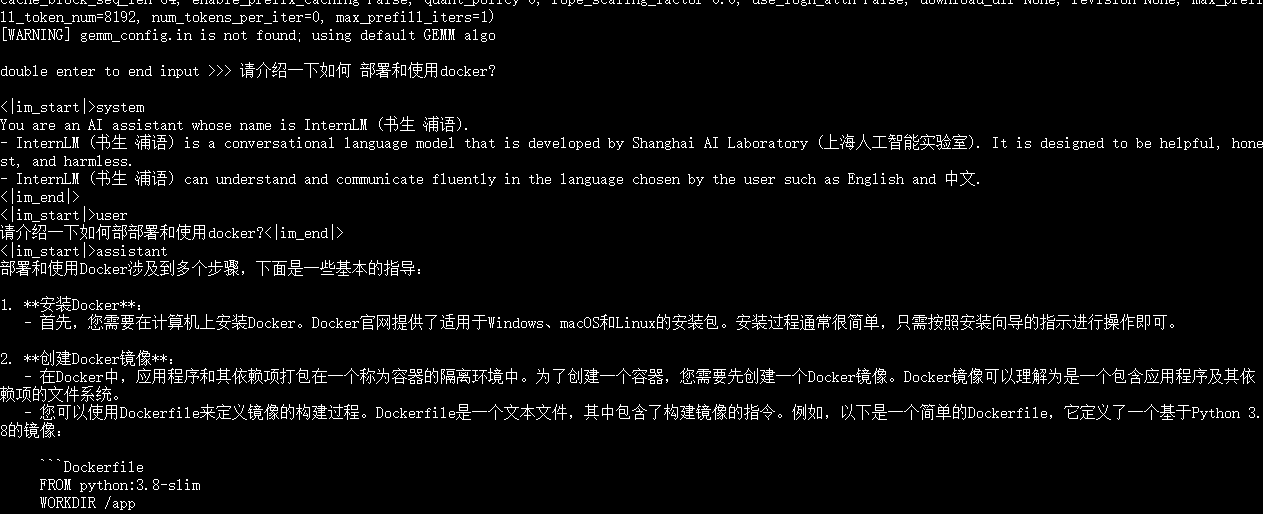


**1.3 LMDeploy验证启动模型文件**

进入创建好的conda环境并启动InternLM2\_5-7b-chat！

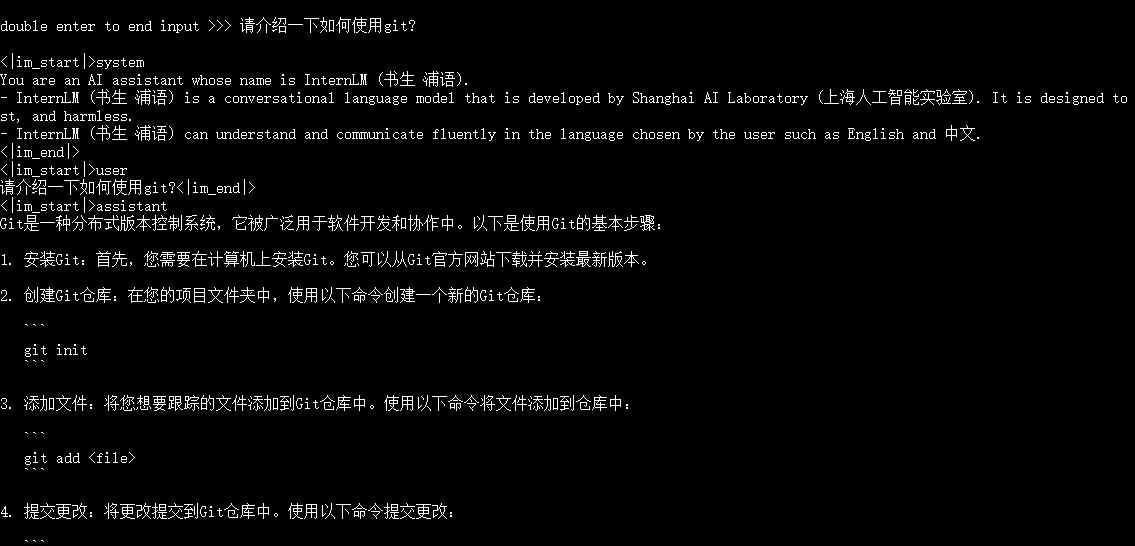
CLI （Command Line Interface）

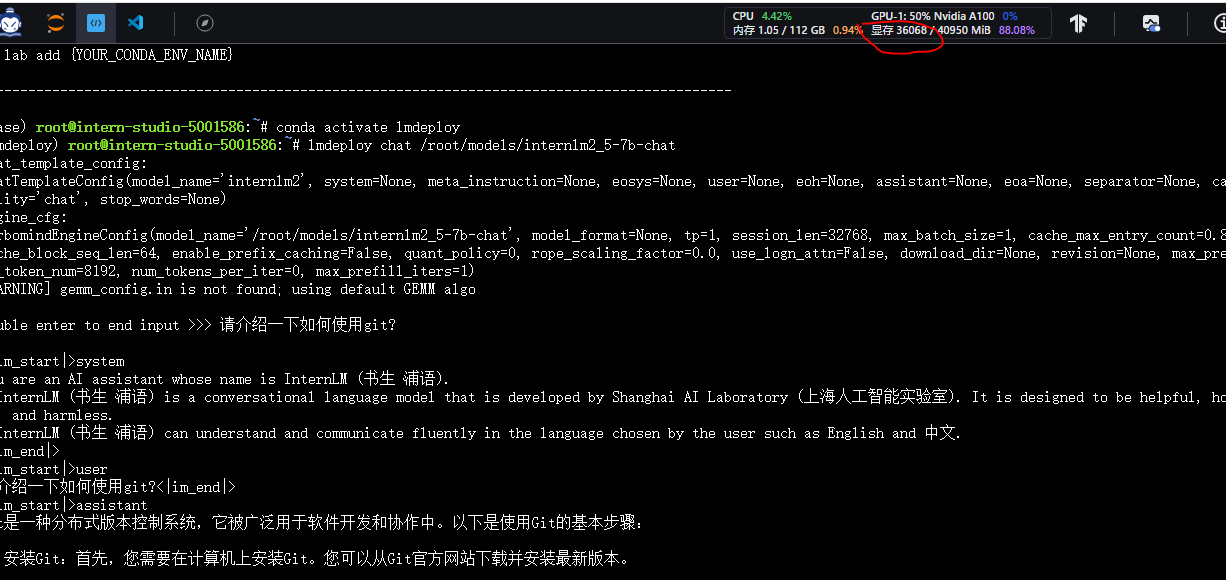


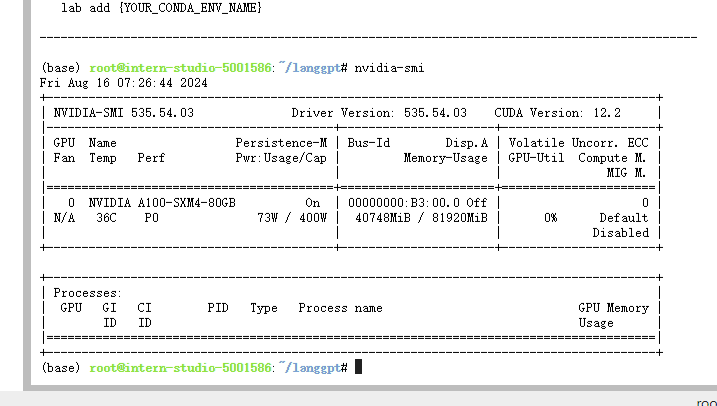


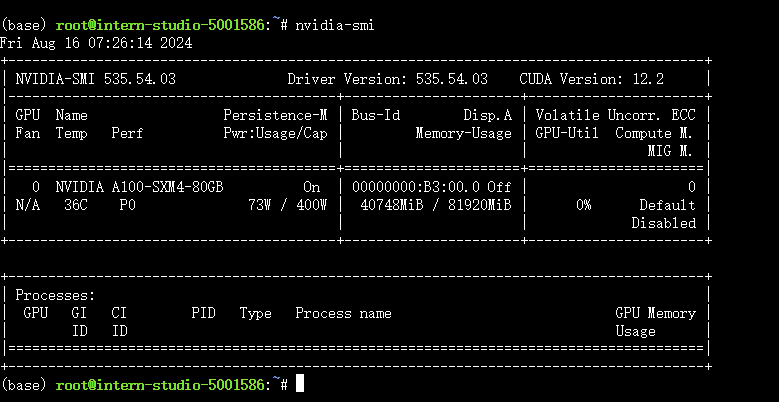


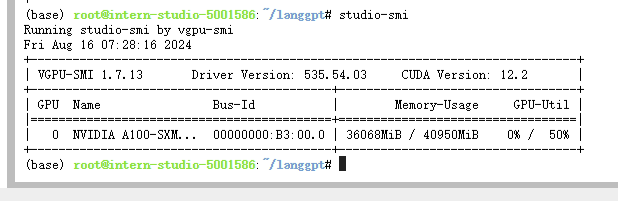
如果选择 ***50%A100*\**1*** 建立机器，同样运行InternLM2.5 7B模型，会发现此时显存占用为**36GB**









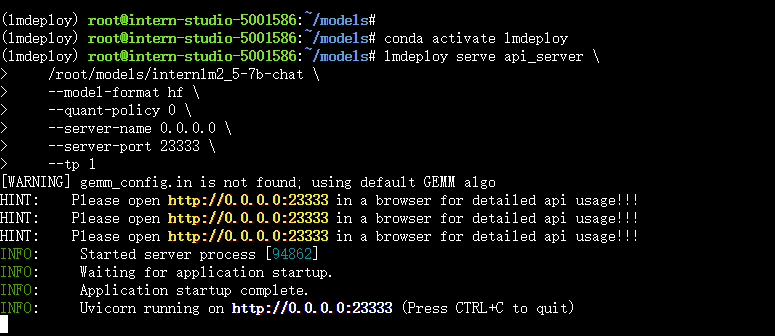


**2 LMDeploy与InternLM2.5**

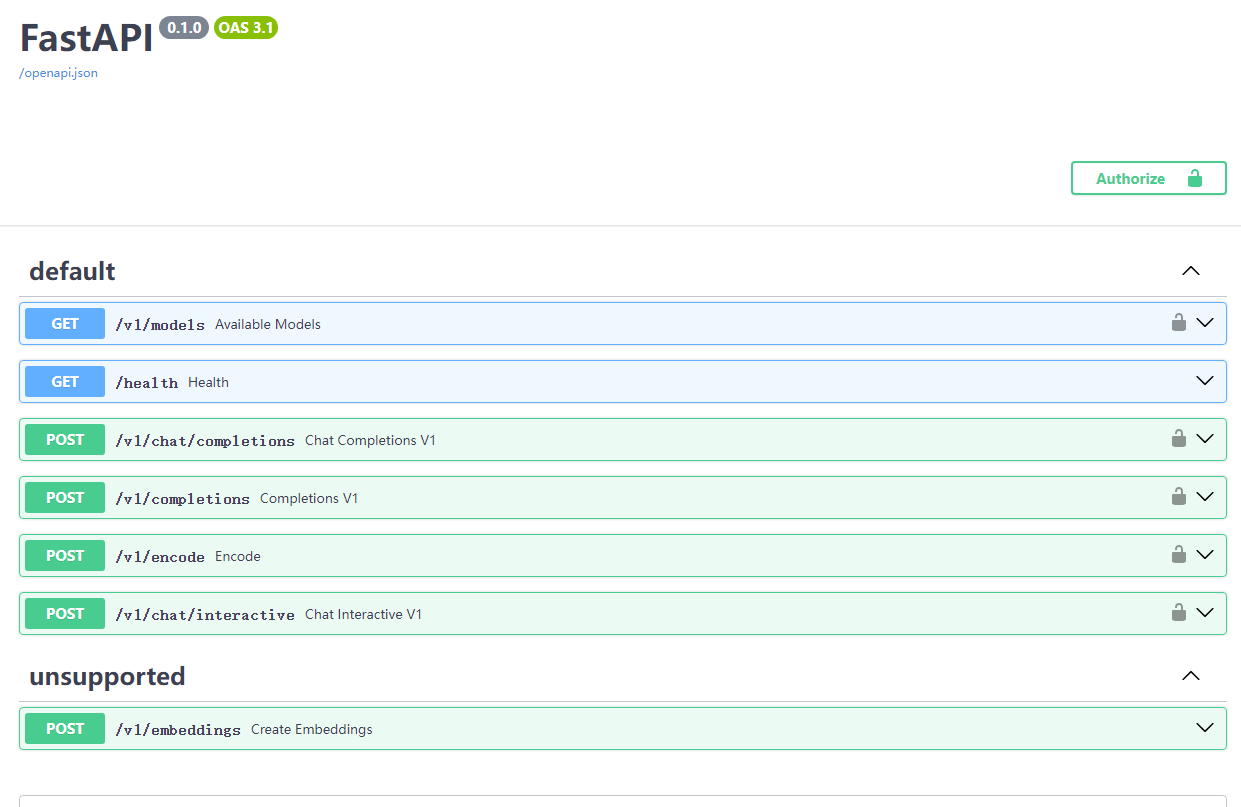
**2.1 LMDeploy API部署InternLM2.5**

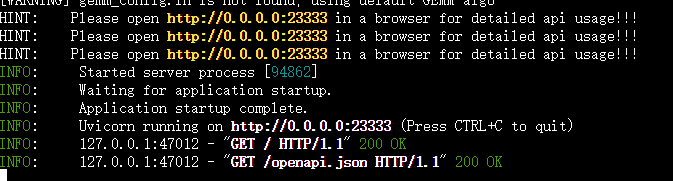
在实际应用中，我们有时会将大模型封装为API接口服务，供客户端访问。

**2.1.1 启动API服务器**

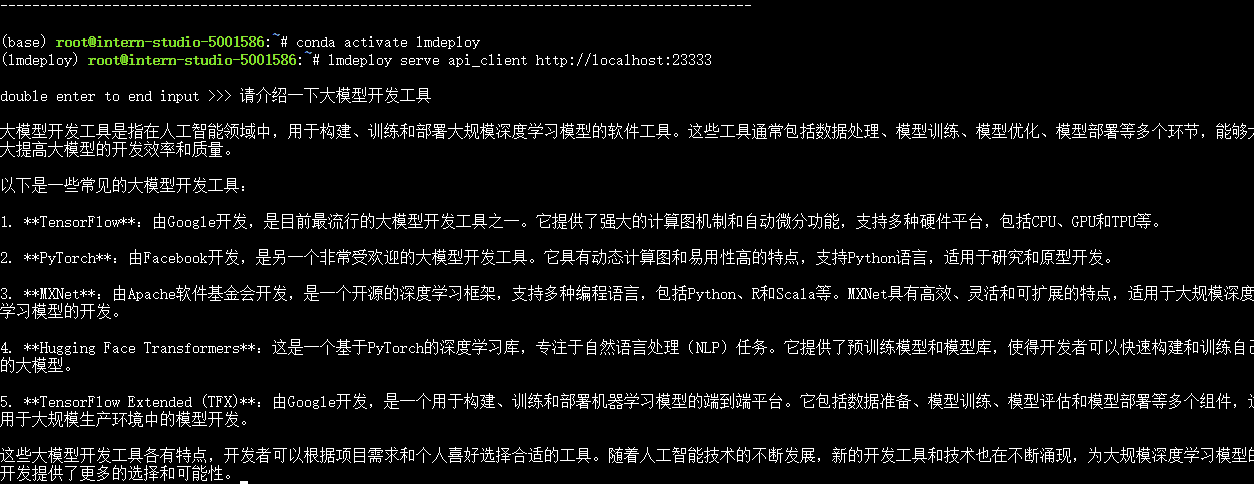


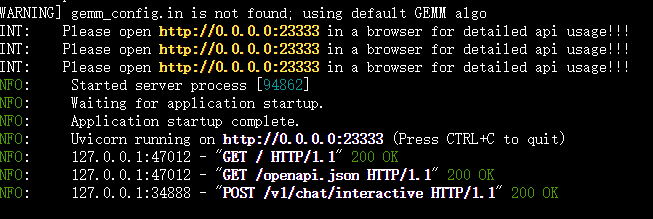




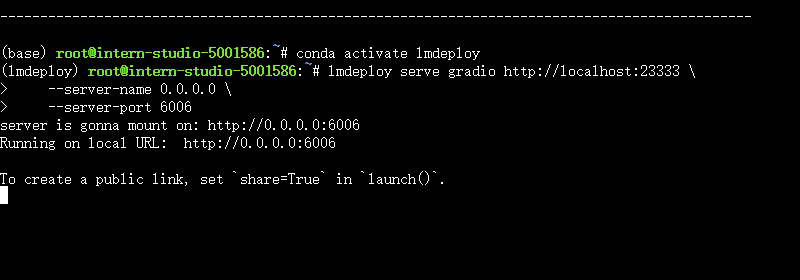


**2.1.2 以命令行形式连接API服务器**





**2.1.3 以Gradio网页形式连接API服务器**





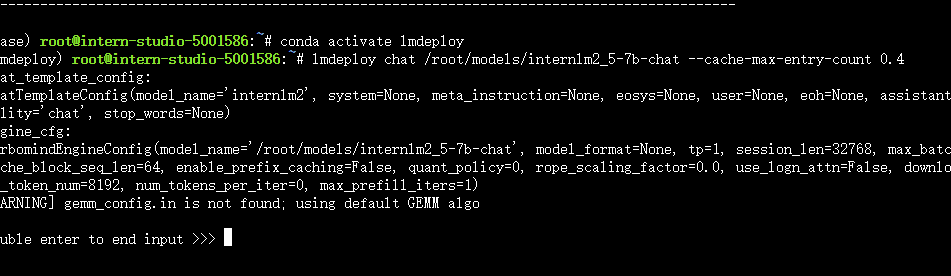




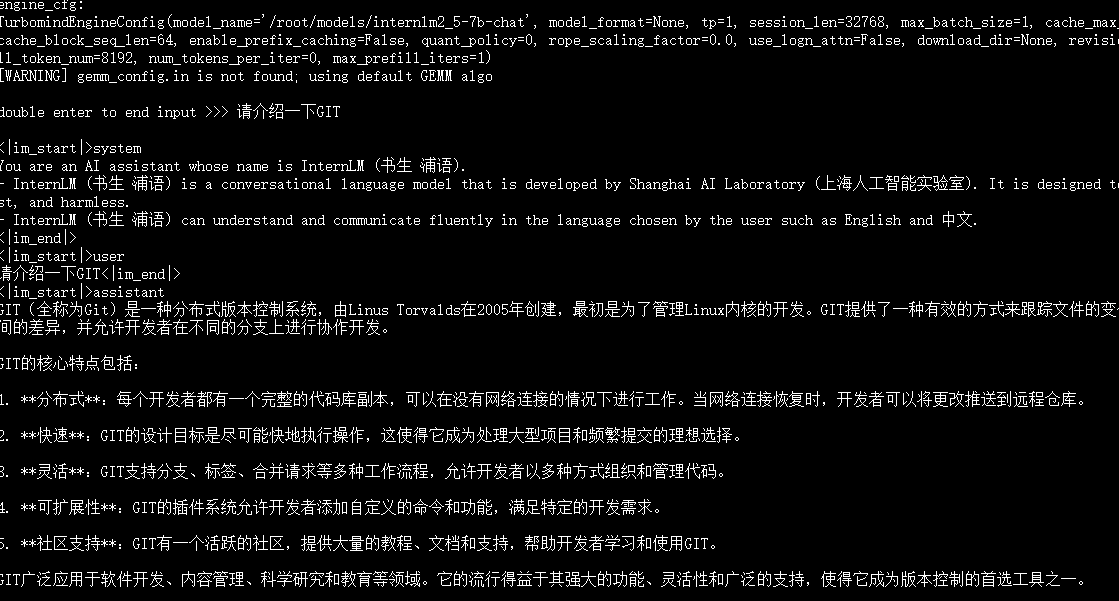
**2.2 LMDeploy Lite**

LMDeploy 提供了权重量化和 k/v cache两种策略。

**2.2.1 设置最大kv cache缓存大小**







**2.2.2 设置在线 kv cache int4/int8 量化**

conda activate lmdeploy

lmdeploy serve api\_server \

/root/models/internlm2\_5-7b-chat \

--model-format hf \

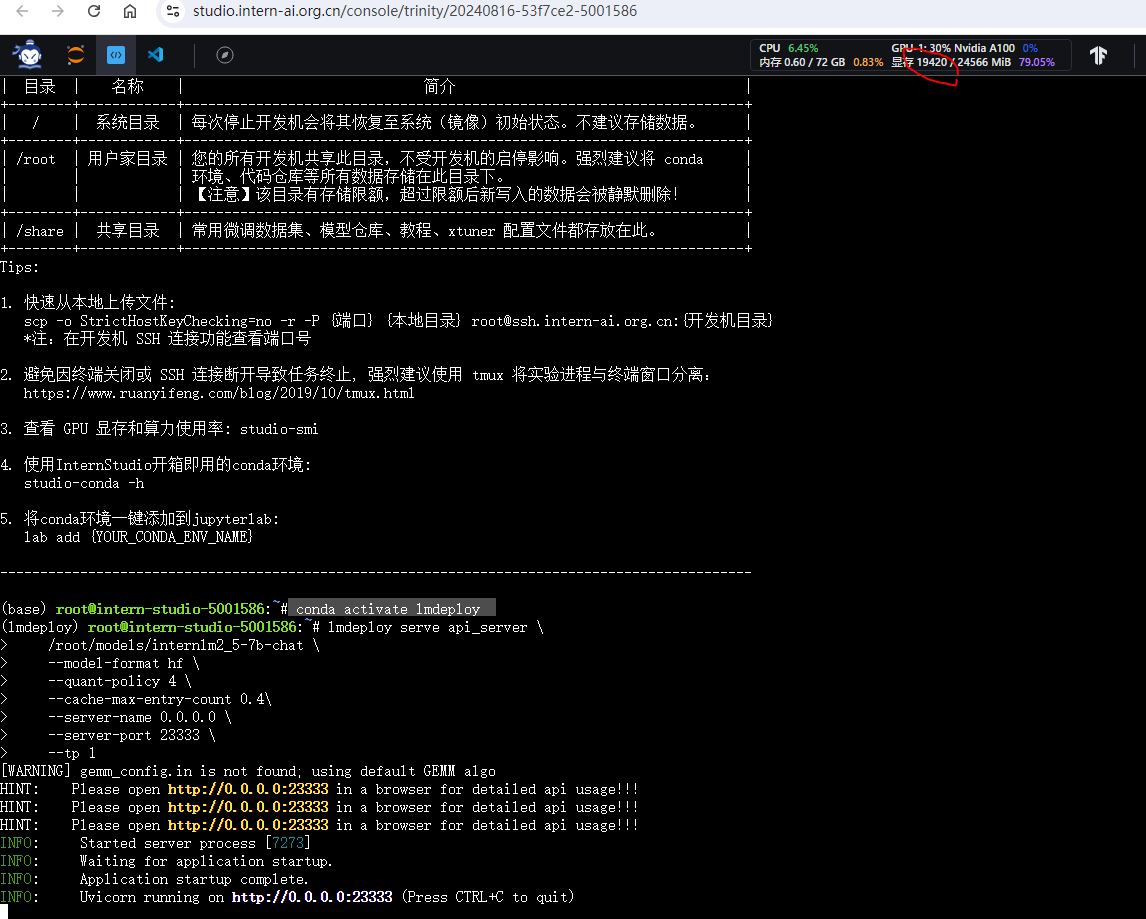
--quant-policy 4 \

--cache-max-entry-count 0.4\

--server-name 0.0.0.0 \

--server-port 23333 \

--tp 1



**2.2.3 W4A16 模型量化和部署**

**lmdeploy lite auto\_awq \**

**/root/models/internlm2\_5-7b-chat \**

**--calib-dataset 'ptb' \**

**--calib-samples 128 \**

**--calib-seqlen 2048 \**

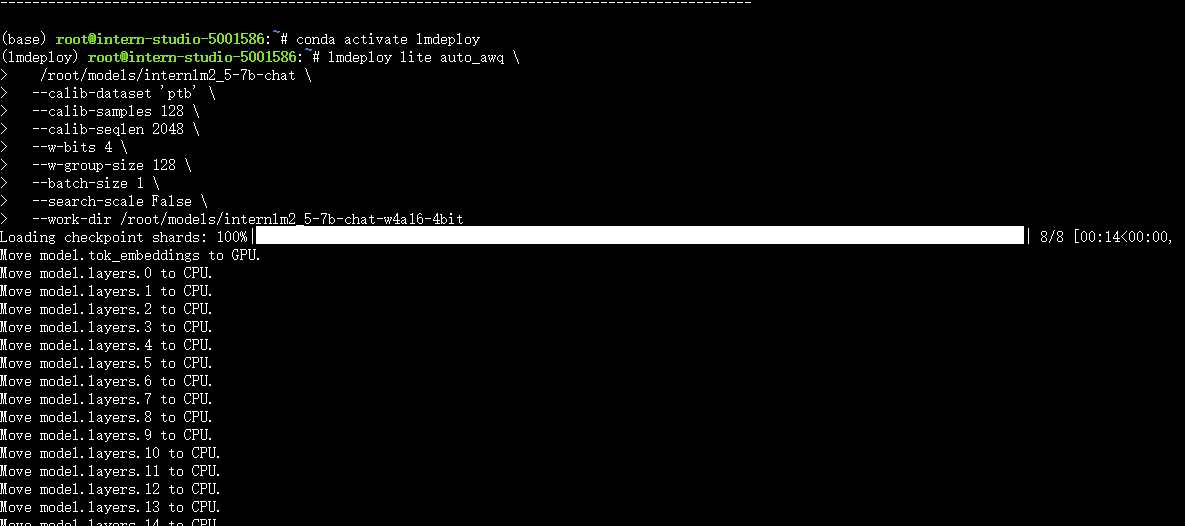
**--w-bits 4 \**

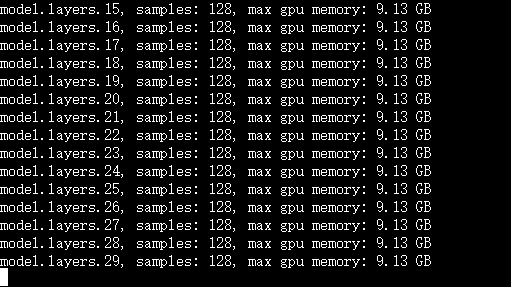
**--w-group-size 128 \**

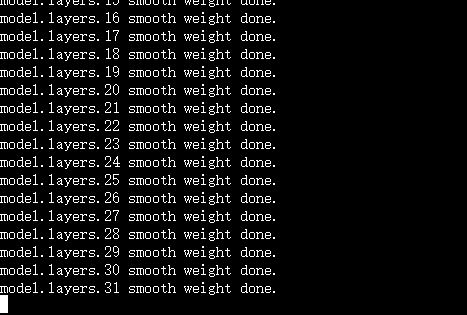
**--batch-size 1 \**

**--search-scale False \**

**--work-dir /root/models/internlm2\_5-7b-chat-w4a16-4bit**

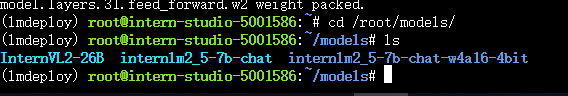


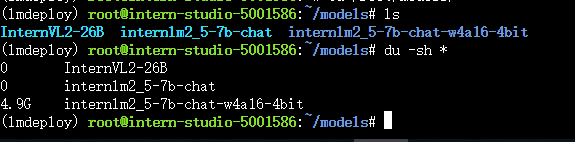


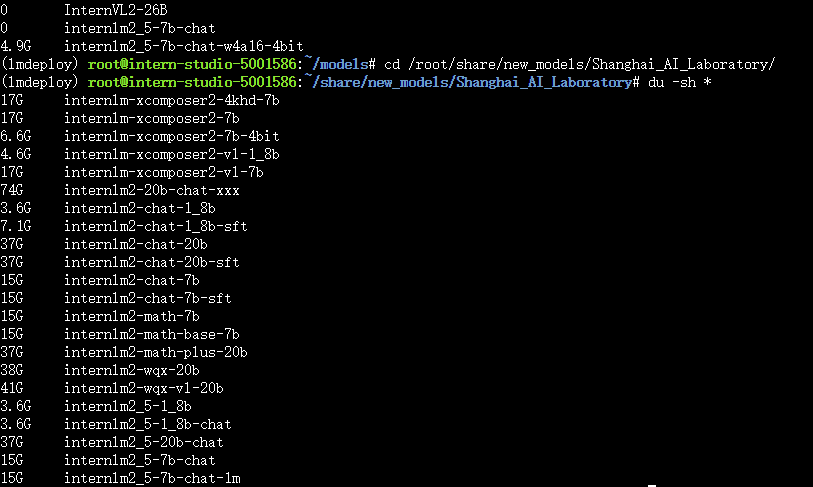




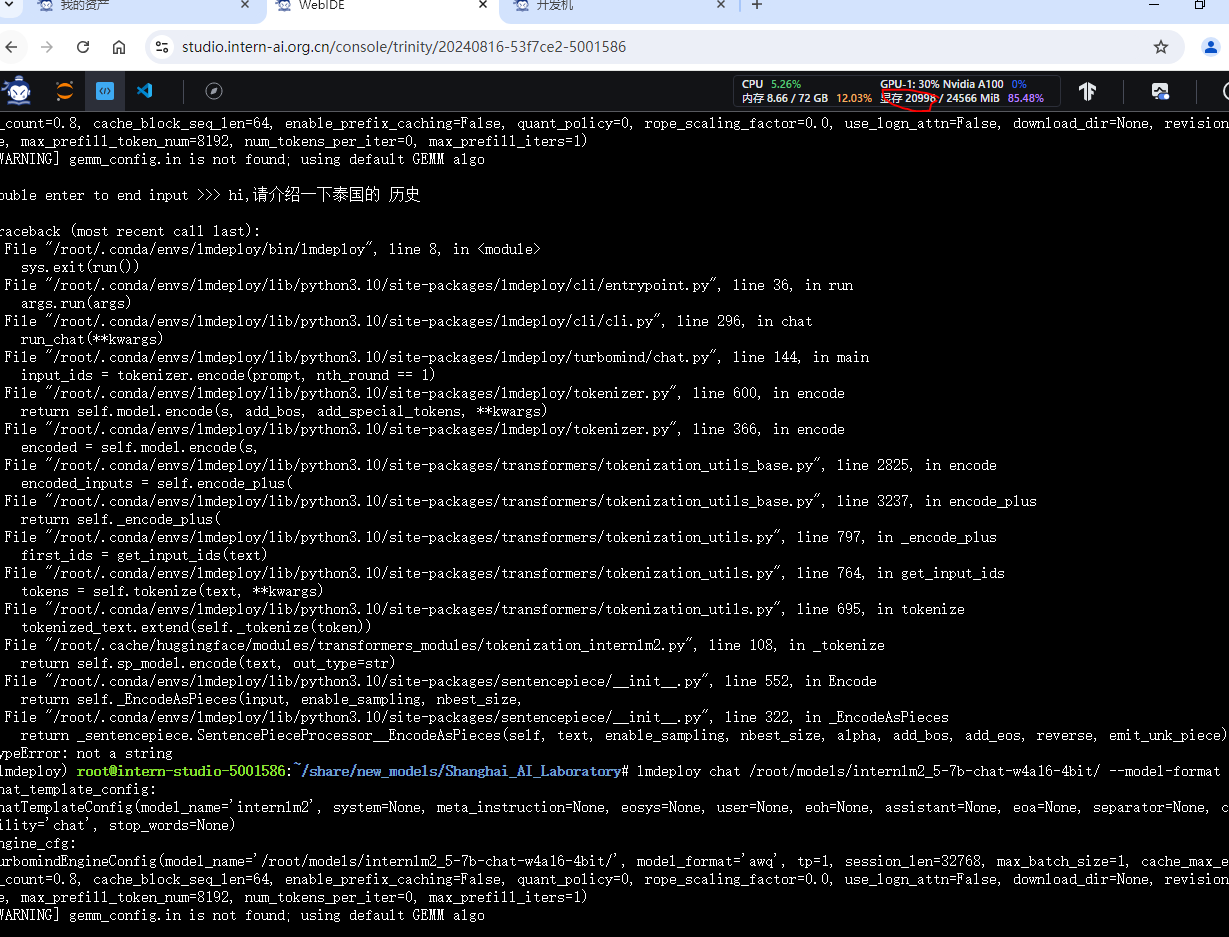
耗时3个小时左右。重要运算完了。

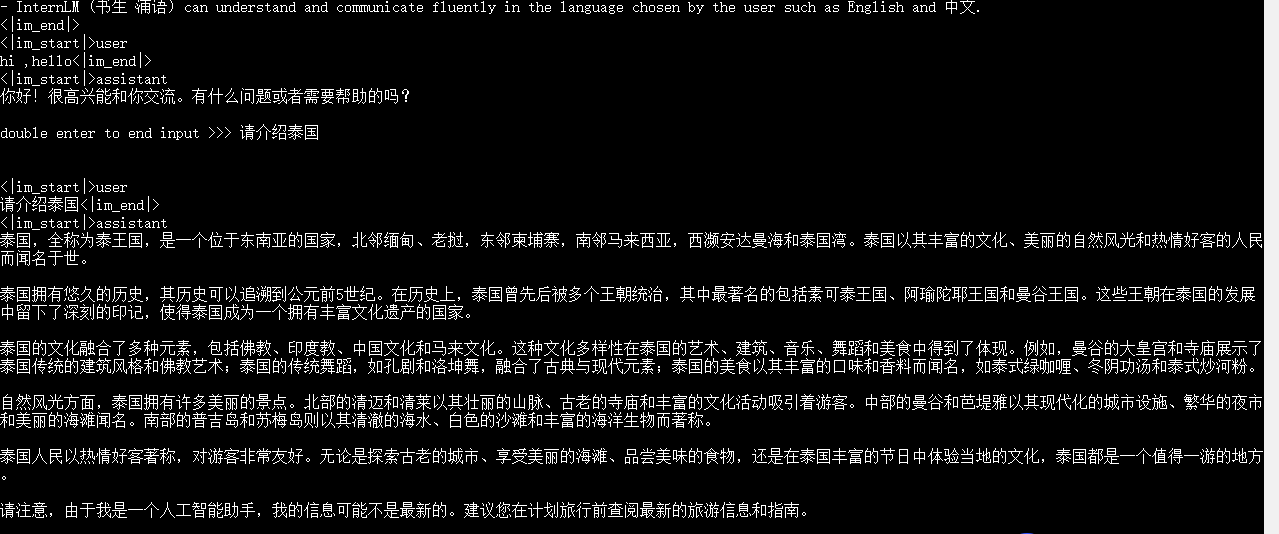






一经对比即可发觉，15G对4.9G，优势在我。

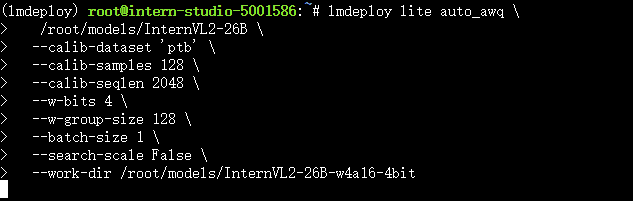


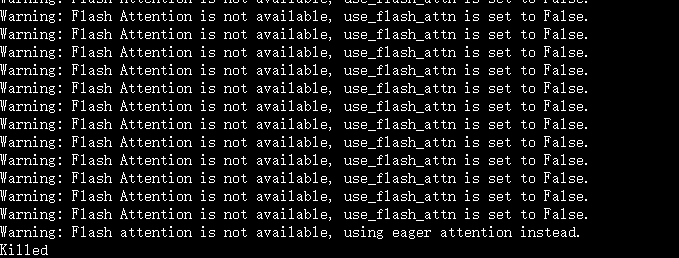
**2.2.4 W****4A16 量化+ KV cache+KV cache 量化**

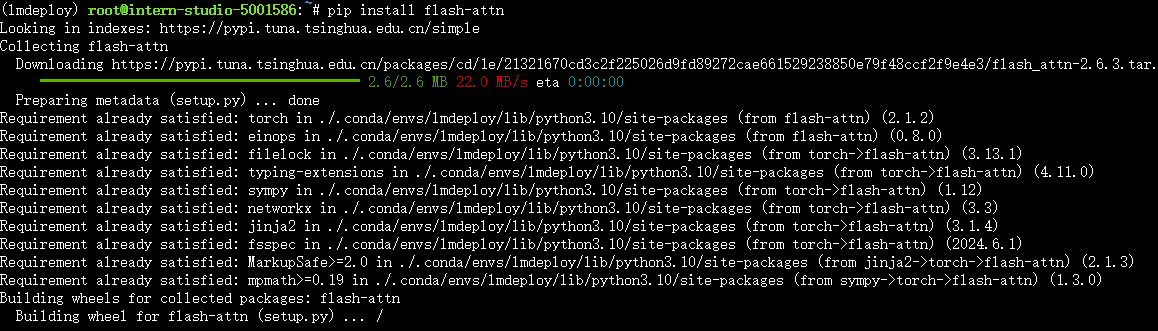


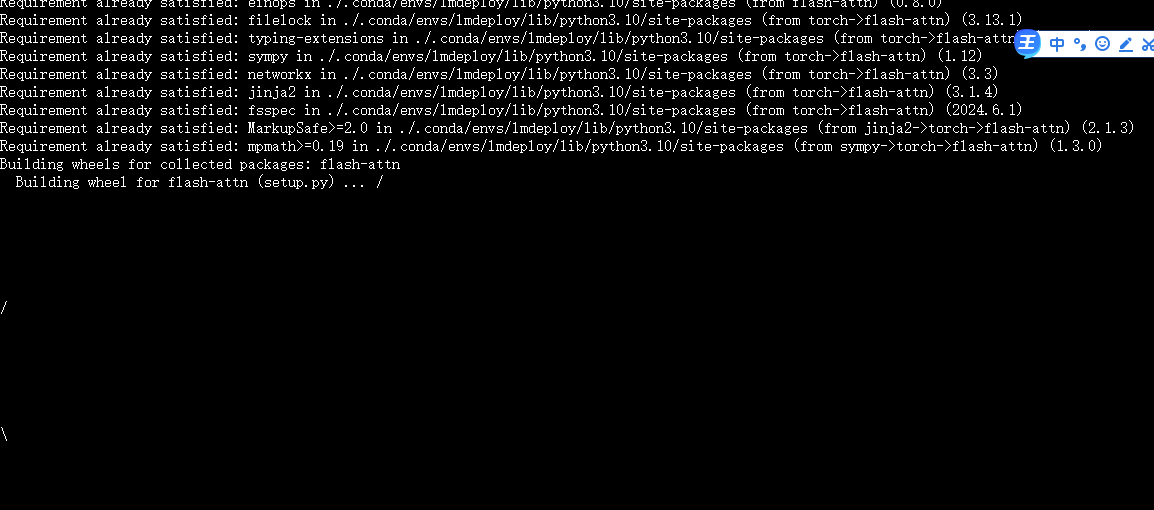
**3 LMDeploy与InternVL2**

**3.1.1 W4A16 模型量化和部署**



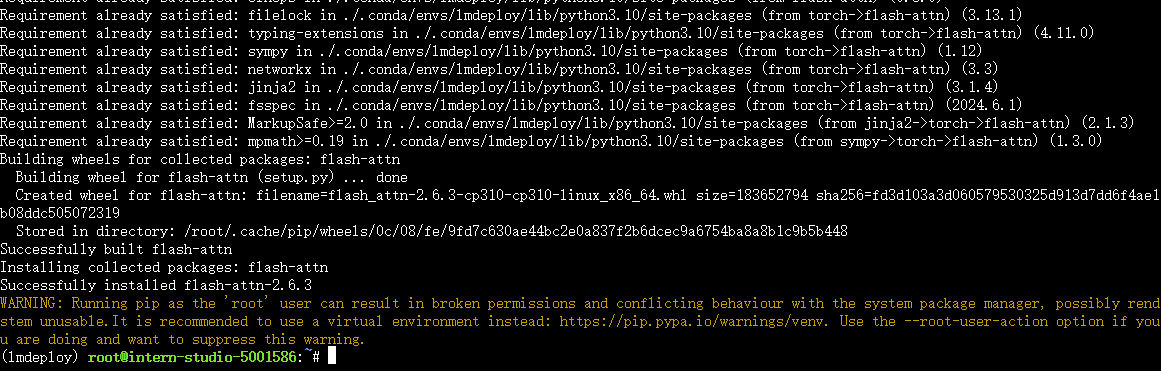


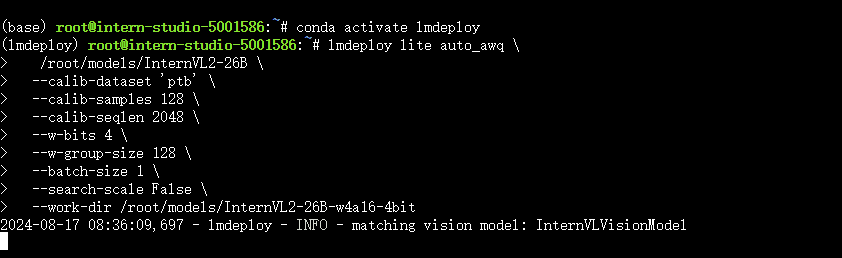


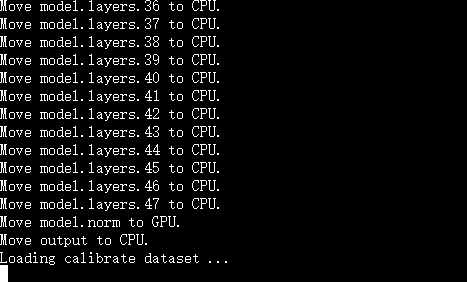


pip install flash-attn

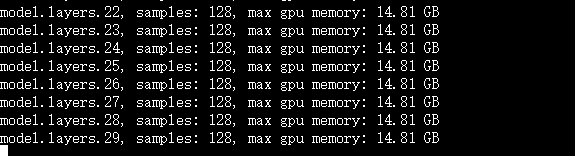
然后运行了很久（数个小时）。

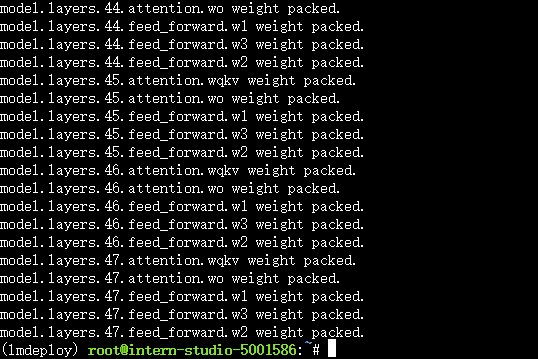




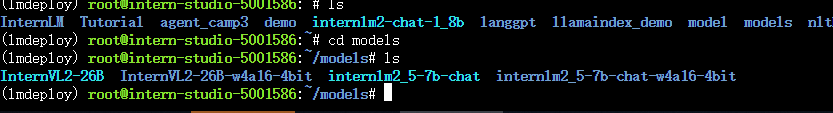


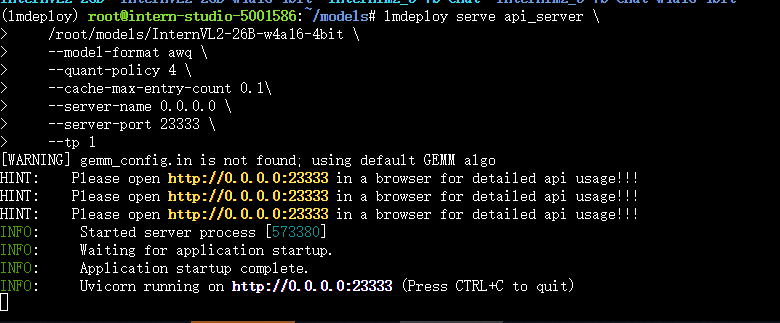
还是要很长时间。。。。。

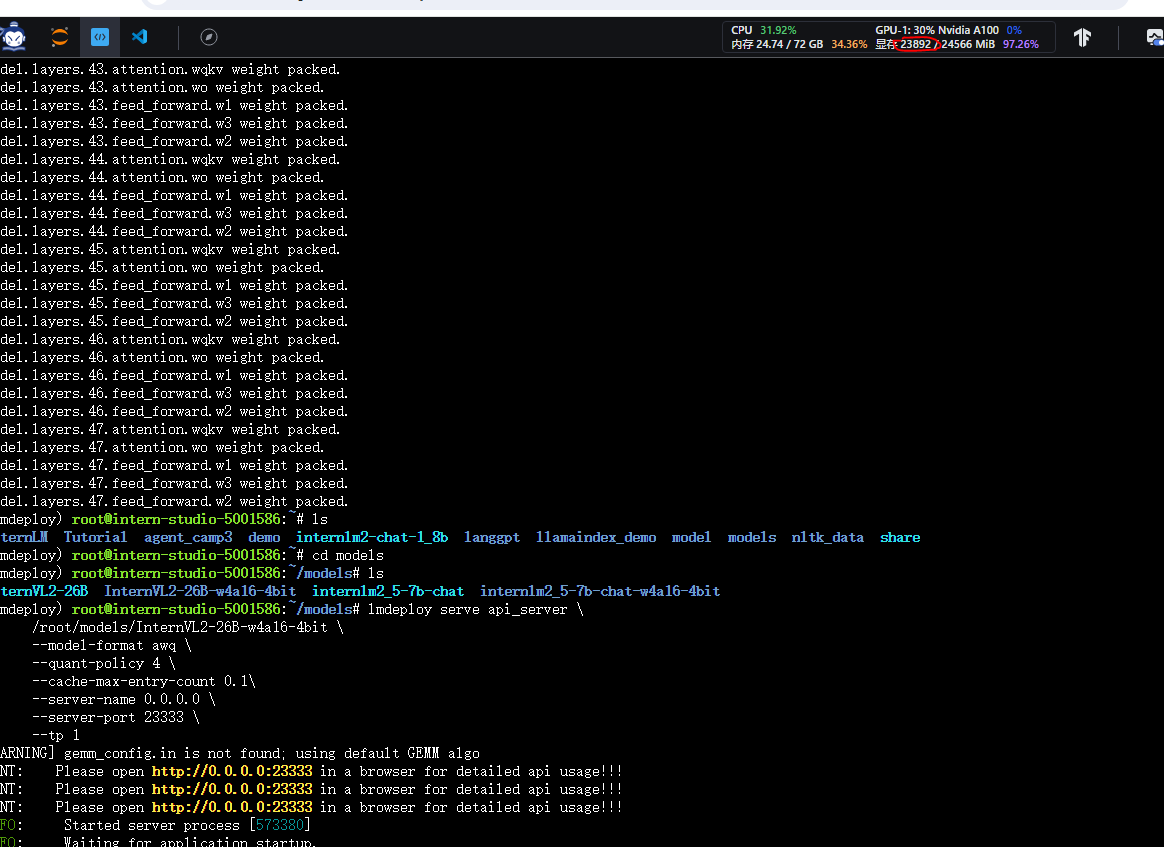


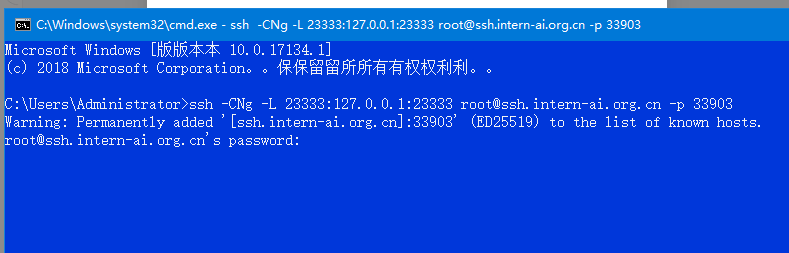


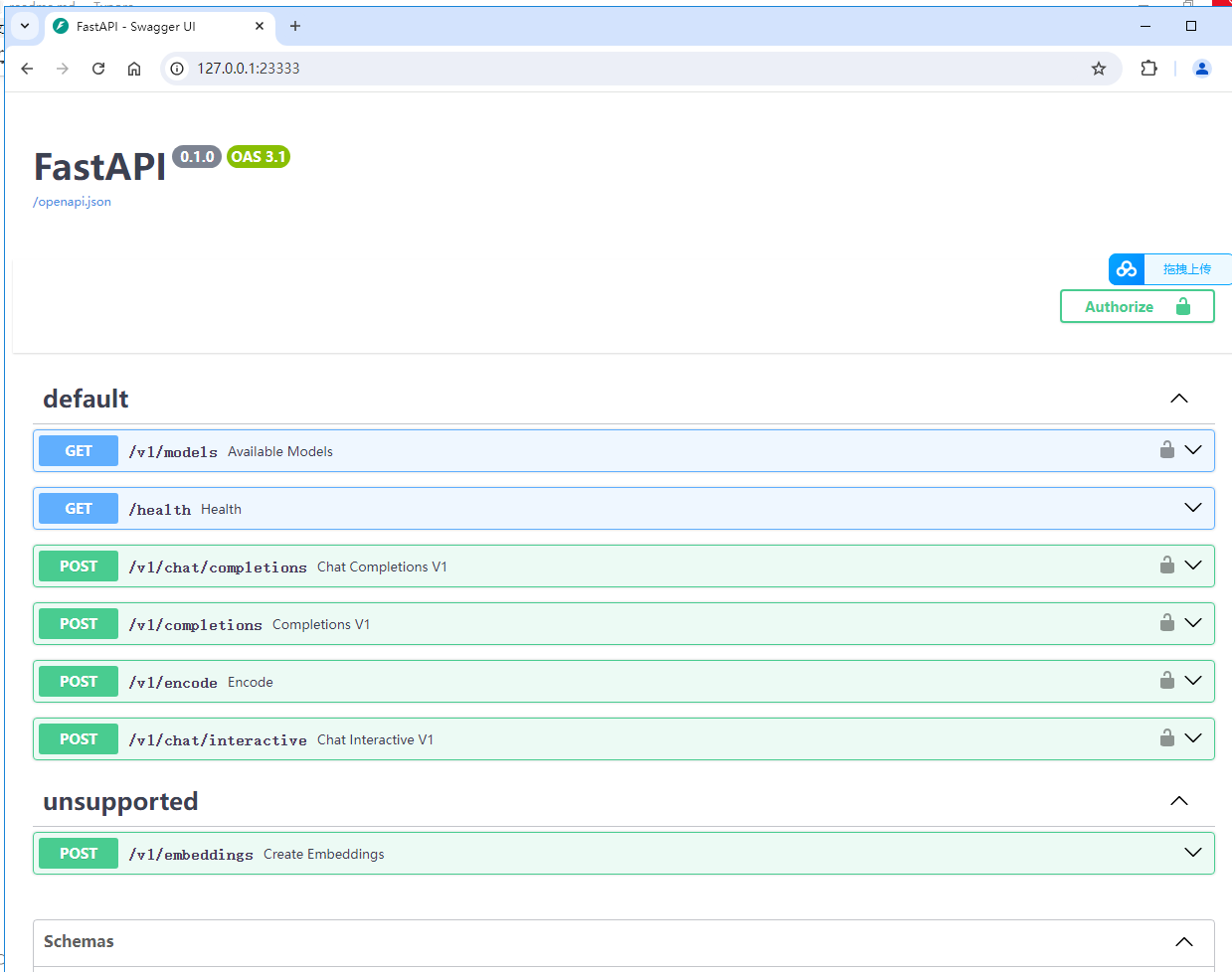
**3.2 LMDeploy API部署InternVL2**











与[2.1.2 以命令行形式连接API服务器](#2.1.2)

[2.1.3 以Gradio网页形式连接API服务器](#2.1.3) 流程相同

**4 LMDeploy之FastAPI与Function call**

**4.1 API开发**

**conda activate lmdeploy**

**lmdeploy serve api\_server \**

**/root/models/internlm2\_5-7b-chat-w4a16-4bit \**

**--model-format awq \**

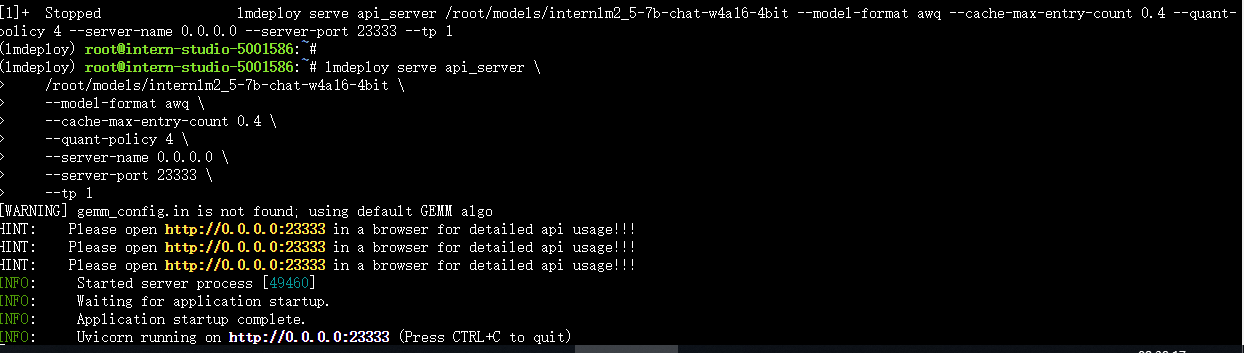
**--cache-max-entry-count 0.4 \**

**--quant-policy 4 \**

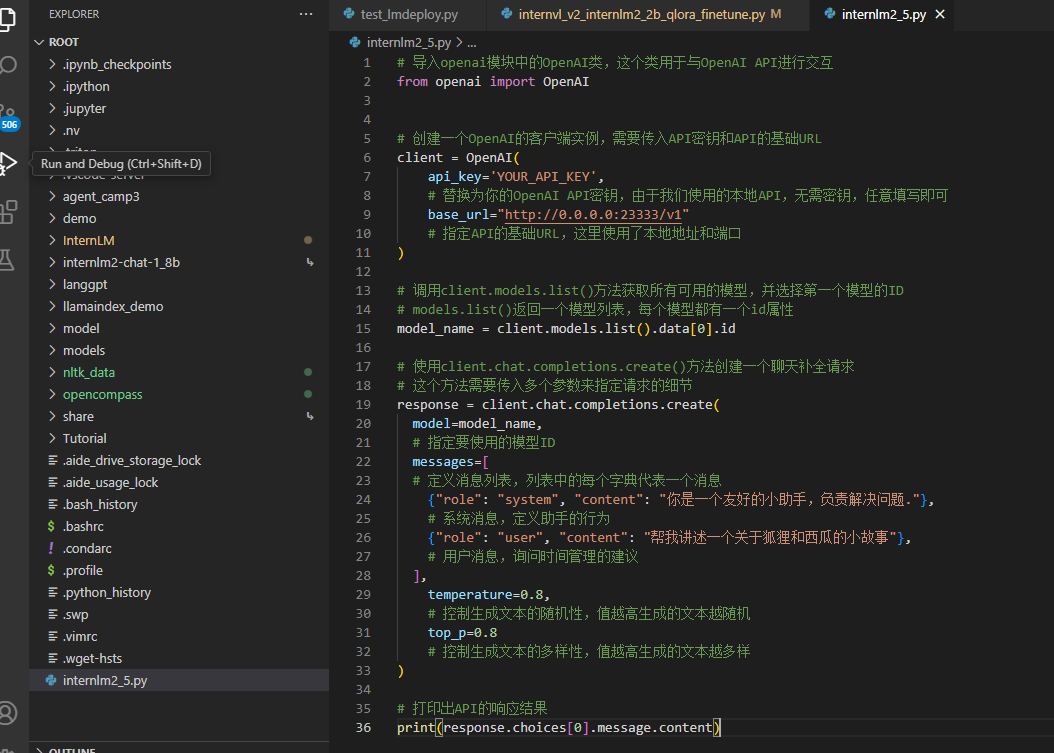
**--server-name 0.0.0.0 \**

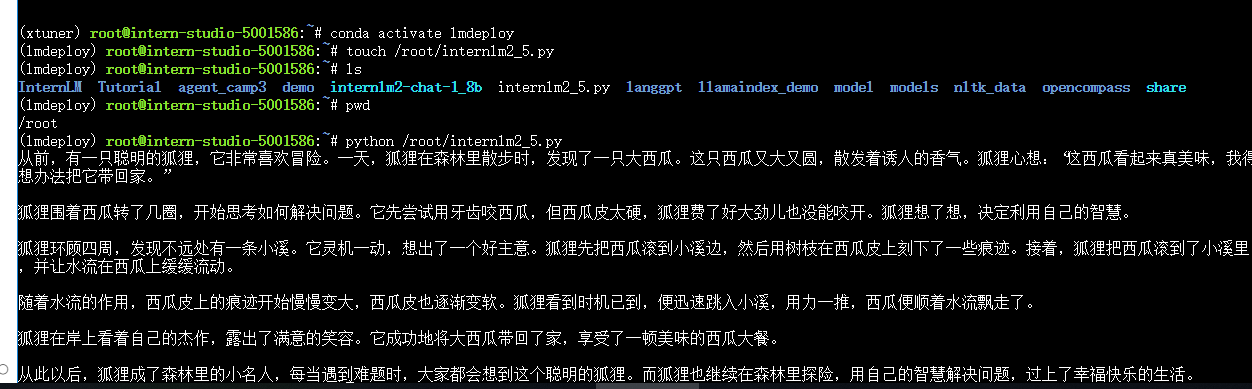
**--server-port 23333 \**

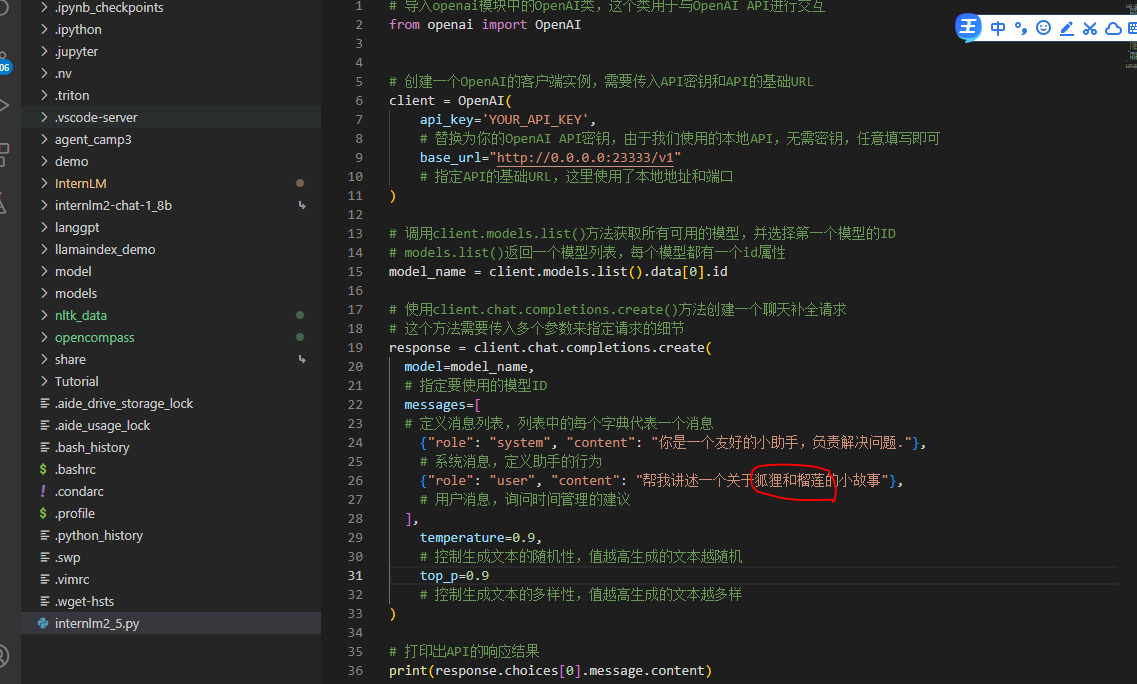
**--tp 1**

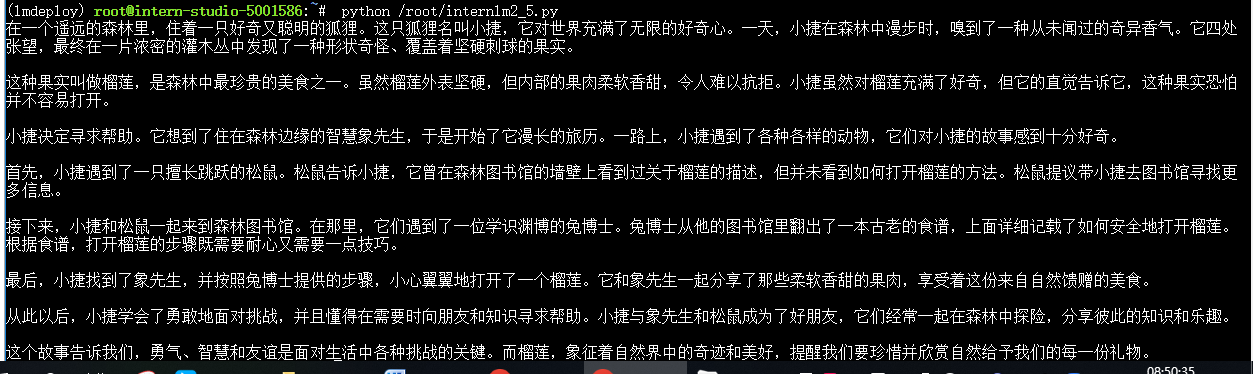


在新建终端中输入如下指令，新建internlm2\_5.py。









在一个遥远的森林里，住着一只好奇又聪明的狐狸。这只狐狸名叫小捷，它对世界充满了无限的好奇心。一天，小捷在森林中漫步时，嗅到了一种从未闻过的奇异香气。它四处张望，最终在一片浓密的灌木丛中发现了一种形状奇怪、覆盖着坚硬刺球的果实。

这种果实叫做榴莲，是森林中最珍贵的美食之一。虽然榴莲外表坚硬，但内部的果肉柔软香甜，令人难以抗拒。小捷虽然对榴莲充满了好奇，但它的直觉告诉它，这种果实恐怕并不容易打开。

小捷决定寻求帮助。它想到了住在森林边缘的智慧象先生，于是开始了它漫长的旅历。一路上，小捷遇到了各种各样的动物，它们对小捷的故事感到十分好奇。

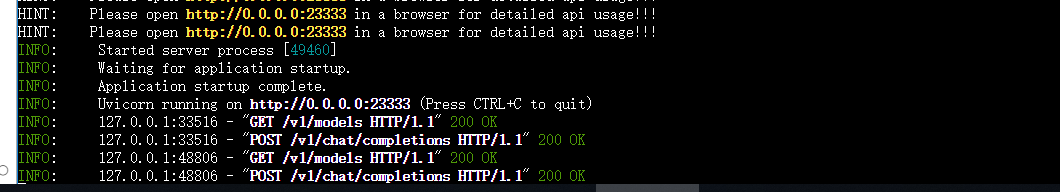
首先，小捷遇到了一只擅长跳跃的松鼠。松鼠告诉小捷，它曾在森林图书馆的墙壁上看到过关于榴莲的描述，但并未看到如何打开榴莲的方法。松鼠提议带小捷去图书馆寻找更多信息。

接下来，小捷和松鼠一起来到森林图书馆。在那里，它们遇到了一位学识渊博的兔博士。兔博士从他的图书馆里翻出了一本古老的食谱，上面详细记载了如何安全地打开榴莲。根据食谱，打开榴莲的步骤既需要耐心又需要一点技巧。

最后，小捷找到了象先生，并按照兔博士提供的步骤，小心翼翼地打开了一个榴莲。它和象先生一起分享了那些柔软香甜的果肉，享受着这份来自自然馈赠的美食。

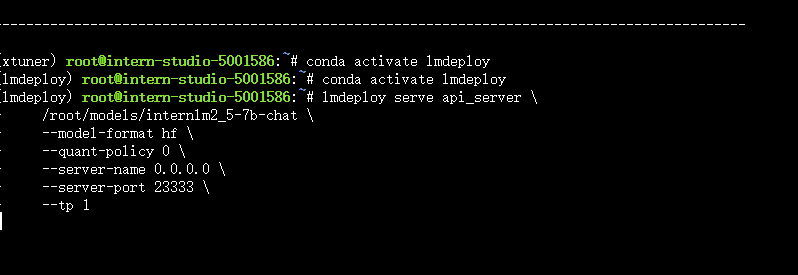
从此以后，小捷学会了勇敢地面对挑战，并且懂得在需要时向朋友和知识寻求帮助。小捷与象先生和松鼠成为了好朋友，它们经常一起在森林中探险，分享彼此的知识和乐趣。

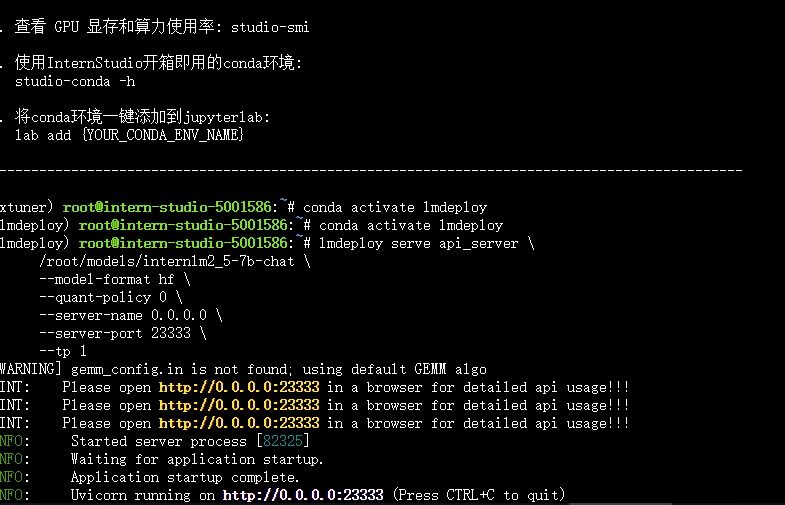
这个故事告诉我们，勇气、智慧和友谊是面对生活中各种挑战的关键。而榴莲，象征着自然界中的奇迹和美好，提醒我们要珍惜并欣赏自然给予我们的每一份礼物。



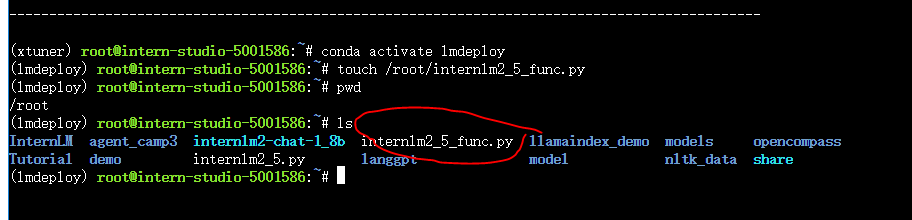
**4.2 Function call**

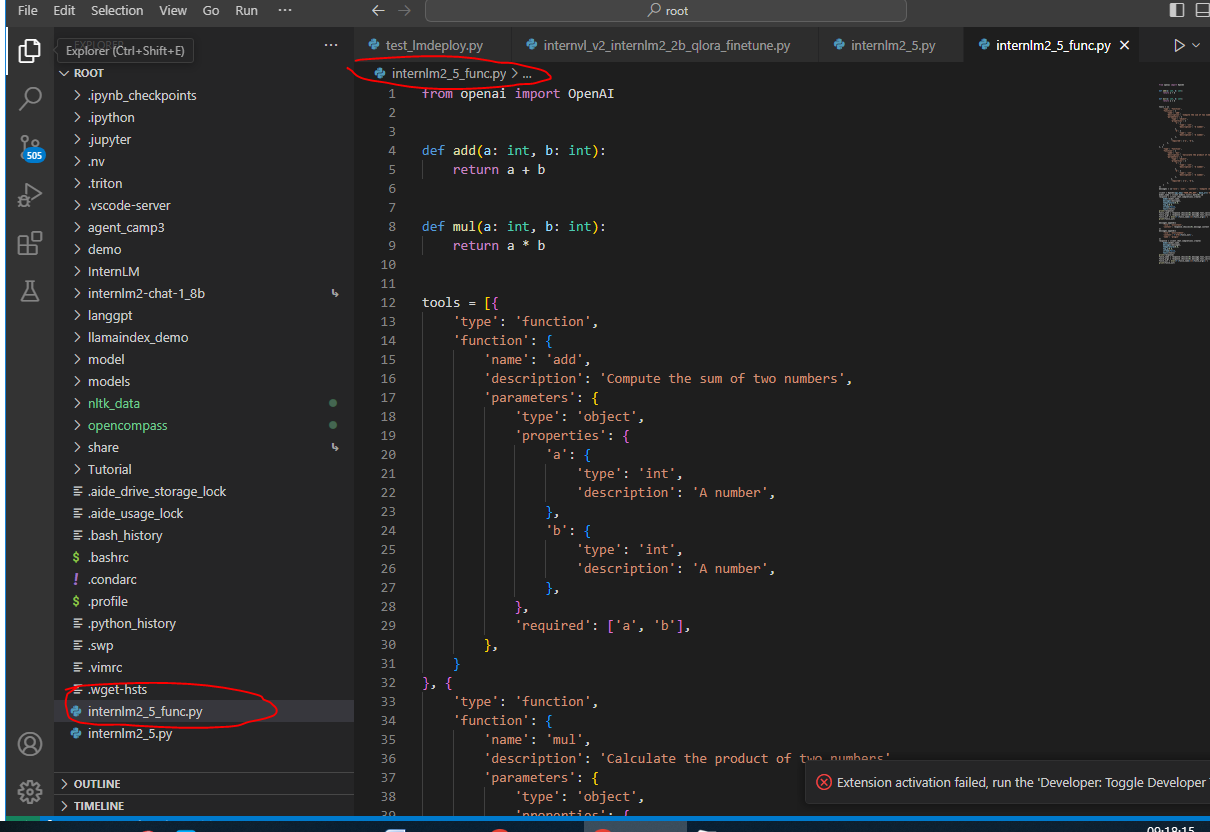
关于Function call，即函数调用功能，它允许开发者在调用模型时，详细说明函数的作用，并使模型能够智能地根据用户的提问来输入参数并执行函数。完成调用后，模型会将函数的输出结果作为回答用户问题的依据。

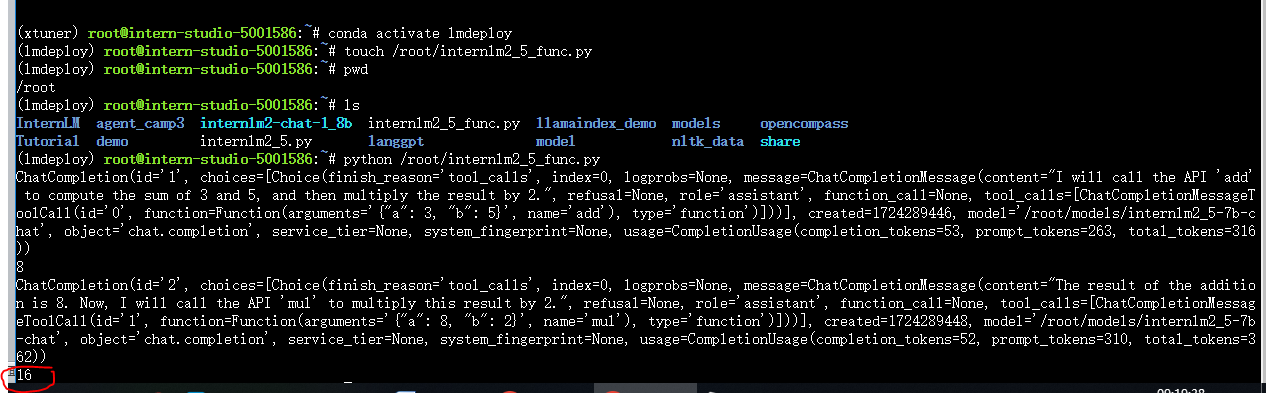












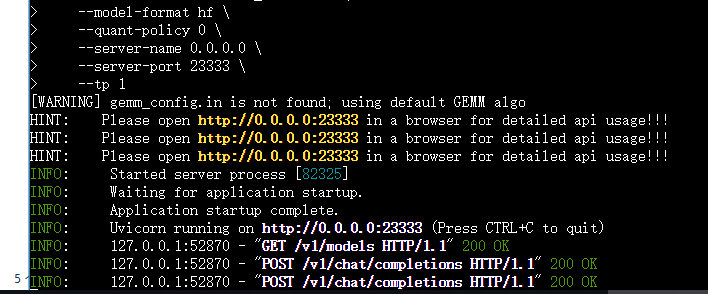
(lmdeploy) root@intern-studio-5001586:~# python /root/internlm2\_5\_func.py

ChatCompletion(id='1', choices=[Choice(finish\_reason='tool\_calls', index=0, logprobs=None, message=ChatCompletionMessage(content="I will call the API 'add' to compute the sum of 3 and 5, and then multiply the result by 2.", refusal=None, role='assistant', function\_call=None, tool\_calls=[ChatCompletionMessageToolCall(id='0', function=Function(arguments='{"a": 3, "b": 5}', name='add'), type='function')]))], created=1724289446, model='/root/models/internlm2\_5-7b-chat', object='chat.completion', service\_tier=None, system\_fingerprint=None, usage=CompletionUsage(completion\_tokens=53, prompt\_tokens=263, total\_tokens=316))

8

ChatCompletion(id='2', choices=[Choice(finish\_reason='tool\_calls', index=0, logprobs=None, message=ChatCompletionMessage(content="The result of the addition is 8. Now, I will call the API 'mul' to multiply this result by 2.", refusal=None, role='assistant', function\_call=None, tool\_calls=[ChatCompletionMessageToolCall(id='1', function=Function(arguments='{"a": 8, "b": 2}', name='mul'), type='function')]))], created=1724289448, model='/root/models/internlm2\_5-7b-chat', object='chat.completion', service\_tier=None, system\_fingerprint=None, usage=CompletionUsage(completion\_tokens=52, prompt\_tokens=310, total\_tokens=362))

16



以上可以看出，InternLM2.5将输入'Compute (3+5)\*2'根据提供的function拆分成了"加"和"乘"两步，第一步调用function add实现加，再于第二步调用function mul实现乘，再最终输出结果16.