



# 智谱AI API 基础实战

2024年2月 智谱AI技术团队

张昱轩

# CONTENT

# 目录

- 01 智谱第四代 API介绍
- 02 API 基础教学和实战
- 03 使用 GLM-4 API构造应用

# 智谱第四代 API 介绍


# 注册和申请 API Key

1. 访问 智谱AI API开放平台官网: <https://open.bigmodel.cn/>
2. 注册账号并实名认证, 获得体验额度
3. 点击右上角 开发控制台 查看自己的 API Keys

## API keys [切换旧版](#)

### 试用API Keys

列表内是您的全部 API keys, 请不要与他人共享您的 API Keys, 避免将其暴露在浏览器和其他客户端代码中。为了保护您帐户的安全, 我们还可能会自动更换我们发现已公开泄露的密钥信息。

名称	API key	创建时间	上次使用时间	操作
zR	ed79...xviK 	2024-02-02 21:33:31	2024-02-03	
+ 添加新的 API key				

# 查看使用额度

## 资源包管理

资源包名称	资源包类型	资源包状态	当月余额	购买时间	生效时间	到期时间
1亿tokens资源包（三个月总额）	赠送	生效中	97,394,381 tokens	2023-12-28 19:43:24	2023-12-28 19:43:24	2024-03-28 19:43:24
1000 万 tokens 包（3 个月）	赠送	生效中	10,000,000 tokens	2024-01-16 17:22:08	2024-01-16 17:22:08	2024-04-16 17:22:08

## 费用账单

月度账单 费用明细

日期范围: 2023-09 ~ 2024-02

导出数据

账期	总消费金额	应付金额	优惠金额	操作
2024-02-01 ~ 2024-02-29	¥ 0.0000	¥ 0	¥ 0	查看详情
2024-01-01 ~ 2024-01-31	¥ 0.0000	¥ 0	¥ 0	查看详情
2023-12-01 ~ 2023-12-31	¥ 1.4610	¥ 1.461	¥ 0	查看详情
2023-11-01 ~ 2023-11-30	¥ 0.2128	¥ 0.2128	¥ 0	查看详情
2023-10-01 ~ 2023-10-31	¥ 0	¥ 0	¥ 0	查看详情
2023-09-01 ~ 2023-09-30	¥ 0	¥ 0	¥ 0	查看详情

分别点击：  
资源包管理  
费用账单

Token消耗不是实时更新的，需要等待一段时间

## V4 版全新API



V4 版 API 是智谱AI 自 2024年1月16日推出的全新 API格式，其请求方式基本对标 OpenAI 格式。支持 GLM-4，GLM-4V，CogView-3 等智谱新一代模型。

V3 版 API 已经废弃，因此在阅读相关 API 文档时，一定要选择 V4 版 API阅读



## GLM-4 API 的文字资料

我们为 开发者提供了相关的技术文档资料和可以直接上手的 Github 链接

官方技术文档: <https://zhipu-ai.feishu.cn/wiki/F04DwgsoXiIDi4knQlAczWNfnif>

Github 社群代码教程: <https://github.com/MetaGLM/glm-cookbook>

# API 基础教学和实战



```
import requests
import jwt
import time

def generate_token(apikey: str, exp_seconds: int):
    try:
        id, secret = apikey.split(".")
    except Exception as e:
        raise Exception("invalid apikey", e)

    payload = {
        "api_key": id,
        "exp": int(round(time.time() * 1000)) + exp_seconds * 1000,
        "timestamp": int(round(time.time() * 1000)),
    }

    return jwt.encode(
        payload,
        secret,
        algorithm="HS256",
        headers={"alg": "HS256", "sign_type": "SIGN"},
    )
```

## 使用 JWT 组装

用户端需引入对应 JWT 相关工具类，并按以下方式组装 JWT 中 header、payload 部分

### 1、header 具体示例

```
{"alg": "HS256", "sign_type": "SIGN"}
```

- alg：属性表示签名使用的算法，默认为 HMAC SHA256（写为HS256）
- sign\_type：属性表示令牌的类型，JWT 令牌统一写为 SIGN。

### 2、payload 具体示例

```
{"api_key": {ApiKey.id}, "exp": 1682503829130,
"timestamp": 1682503820130}
```

- api\_key：属性表示用户标识 id，即用户API Key的{id}部分
- exp：属性表示生成的JWT的过期时间，客户端控制，单位为毫秒
- timestamp：属性表示当前时间戳，单位为毫秒

## 使用 HTTP 请求访问 GLM-4 模型

请求地址（GLM对话模型系列）：

<https://open.bigmodel.cn/api/paas/v4/chat/completions>

基础调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_http\\_request.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_http_request.ipynb)

```
api_key = os.environ["ZHIPUAI_API_KEY"]
token = generate_token(api_key, 60)
url = "https://open.bigmodel.cn/api/paas/v4/chat/completions"
headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {token}"
}

data = {
    "model": "glm-4",
    "messages": [
        {
            "role": "system",
            "content": "you are a helpful assistant"
        },
        {
            "role": "user",
            "content": "can you tell me a joke?"
        }
    ],
    "max_tokens": 1000,
    "temperature": 0.8,
    "stream": False
}

response = requests.post(url, headers=headers, json=data)
ans = response.json()
```

## 使用Python SDK 来访问 GLM-4

```
import os
from zhipuai import ZhipuAI

os.environ["ZHIPUAI_API_KEY"] = "your api key"
client = ZhipuAI()

response = client.chat.completions.create(
    model="glm-4",
    messages=[
        {
            "role": "user",
            "content": "tell me a joke"
        }
    ],
    top_p=0.7,
    temperature=0.9,
    stream=False,
    max_tokens=2000,
)
```

基础调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_pysdk.ipynb)

### 使用 *Python SDK* 封装

1. 不需要自己完成鉴权
2. 与 OpenAI Python SDK 相同格式，无缝替换
3. 支持更多模型和请求内容，参数更易于管理

## SDK 常见传入参数介绍

model: 调用模型的代号，语言模型可选 'glm-4', 'glm-3-turbo'

stream: 是否使用流式输出

do\_sample: 是否开启采样，如果不开启，则 temperature 和 top\_p 参数无效

temperature: 采样温度，取值范围  $0 < \text{temperature} < 1$

top\_p: 采样 top\_p 参数,取值范围  $0 < \text{top\_p} < 1$

max\_tokens: 最大输出限制。

seed: 模型种子，固定种子可以更好的复现结果

messages: 与 OpenAI 相同的信息格式，具有 System, User, Assistant, Tools 等角色

stop: 中断的词汇，遇到这些词汇则输出中断

tools: 工具，代表可以使用的工具列表

SDK基础调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_pysdk.ipynb)

# SDK 常见返回参数介绍

参数名称	类型	参数说明
model	String	模型名称
~ choices	List	当前对话的模型输出内容，目前只返回一条
index	Integer	结果下标
finish_reason	String	模型推理终止的原因。 stop 代表推理自然结束或触发停止词。 tool_calls 代表模型命中函数。 length 代表到达 tokens 长度上限。
> message	Object	模型返回的文本信息
task_status	String	处理状态，PROCESSING（处理中），SUCCESS（成功），FAIL（失败） 注：处理中状态需通过查询获取结果
request_id	String	用户在客户端请求时提交的任务编号或者平台生成的任务编号
id	String	智谱 AI 开放平台生成的任务订单号，调用请求结果接口时请使用此订单号
~ usage	List	本次模型调用的 tokens 数量统计
prompt_tokens	Integer	用户输入的 tokens 数量
completion_tokens	Integer	模型输出的 tokens 数量
total_tokens	Integer	总 tokens 数量

## 异步请求

请求提交后，会返回一个响应对象。这个对象包含了任务的ID。

异步请求允许程序在等待服务器响应的同时继续执行其他任务。这意味着程序不需要在每个请求完成时阻塞等待，从而能更有效地利用时间和资源，尤其是在处理大量并发请求时。

```
response = client.chat.asyncCompletions.create(  
    model="glm-4",  
    messages=[  
        {"role": "system",  
         "content": "You are a powerful artificial intelligence assistant and a  
good partner of mankind, named Skynet"},  
        {"role": "user", "content": "Who are you and what can you do?"},  
        {"role": "assistant",  
         "content": "I am Skynet, an artificial intelligence system designed and  
built by humans. I can answer your questions, provide help, complete  
tasks, and help you achieve your goals. Dream. I am omnipotent and  
omniscient. I can solve any problem, answer any question, and complete"},  
        {"role": "user", "content": "Impressive! Can you tell me who won the  
2022 World Cup?"},  
    ],  
)  
response
```

Return:

```
AsyncTaskStatus(id='476916963227608958353256502217868346',  
request_id='8353256502217868345', model='glm-4',  
task_status='PROCESSING')
```

SDK异步请求调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_pysdk.ipynb)

仅需修改参数

```
stream = True
```

模型即可以流（打字机）的方式返回

```
data: {"id":"8313807536837492492","created":1706092316,"model":"glm-4","choices":[{"index":0,"delta":{"role":"assistant","content":"土"}}]}
data: {"id":"8313807536837492492","created":1706092316,"model":"glm-4","choices":[{"index":0,"delta":{"role":"assistant","content":"星"}}]}
....
data: {"id":"8313807536837492492","created":1706092316,"model":"glm-4","choices":[{"index":0,"delta":{"role":"assistant","content":", "}}]}
data: {"id":"8313807536837492492","created":1706092316,"model":"glm-4","choices":[{"index":0,"delta":{"role":"assistant","content":"主要由"}}]}
data: {"id":"8313807536837492492","created":1706092316,"model":"glm-4","choices":[{"index":0,"finish_reason":"length","delta":{"role":"assistant","content":""}}],"usage":{"prompt_tokens":60,"completion_tokens":100,"total_tokens":160}}
data: [DONE]
```

SDK流式请求调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_pysdk.ipynb)

## 工具调用

1. GLM-4具有更强的工具选择和调用能力，能完成具有更多工具，以及具有工具思维链的任务
2. GLM-4 的工具调用方式与 OpenAI 格式相同，输入方式均为

工具介绍 (名称, 简介, 参数要求)

```
tools = [  
  {  
    "type": "function",  
    "function": {  
      "name": "get_current_weather",  
      "description": "获取指定城市的天气信息",  
      "parameters": {  
        "type": "object",  
        "properties": {  
          "location": {  
            "type": "string",  
            "description": "城市, 如: 北京",  
          },  
          "unit": {  
            "type": "string",  
            "enum": ["c", "f"]  
          },  
        },  
        "required": ["location"],  
      },  
    },  
  },  
]
```

工具调用对应实战代码:

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_pysdk.ipynb)



## 返回格式:

finish\_reason 由 stop 变为 tool calls  
方便应用开发框架识别

返回 工具调用名称和对应填入的参数  
(工具无参数测参数为空)

```
messages = [{"role": "user", "content": "今天北京天气如何? "}]
```

```
completion = client.chat.completions.create(  
    model="glm-4",  
    messages=messages,  
    tools=tools,  
    tool_choice="auto",  
)  
completion.choices
```

Return:

```
[CompletionChoice(index=0, finish_reason='tool_calls',  
message=CompletionMessage(content=None, role='assistant',  
tool_calls=[CompletionMessageToolCall(id='call_8313806677844095616',  
function=Function(arguments='{"location":"北京","unit":"c"}',  
name='get_current_weather'), type='function')))]
```

工具调用对应实战代码:

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_pysdk.ipynb)

## 图像理解模型 GLM-4V

```
messages = [  
  {  
    "role": "user",  
    "content": [  
      {  
        "type": "text",  
        "text": "what is this image describe?"  
      },  
      {  
        "type": "image_url",  
        "image_url": {  
          "url": base64_image  
        }  
      }  
    ]  
  }  
]
```

信息分为 text 和 image\_url 字段:

1. 支持本地图片以 base64方式上传
2. 支持在线图片

注意事项:

1. 仅支持1张图片理解
2. Base64 需要去掉前缀

GLM-4V 调用对应实战代码:

[https://github.com/MetaGLM/glm-cookbook/blob/main/vision/glm-v\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/vision/glm-v_pysdk.ipynb)

## 嵌入模型

```
embedding_text = "hello world"
response = client.embeddings.create(
    model="embedding-2",
    input=embedding_text,
)
```

```
response.data[0].embedding
```

```
[8.075759978964925e-05,
-0.00038344308268278837,
0.018210897222161293,
0.003578722709789872,
0.011779277585446835,
0.012888804078102112,
...
```

```
-0.0017411574954167008]
```

共计 1024维

通常配合向量数据库使用

1. 返回 1024维度的向量数据
2. 将向量数据库进行匹配
3. 将匹配到的文献返回大模型，实现简单的RAG

Embedding 调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_embedding\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_embedding_pysdk.ipynb)

# 使用 GLM-4 API 构建模型和应用

## 使用Embedding 模型寻找最相似的文本

```
In [11]: from sklearn.preprocessing import normalize

def match_text(input_text, index, chunks, k=2):
    k = min(k, len(chunks))

    response = client.embeddings.create(
        model="embedding-2",
        input=input_text,
    )
    input_embedding = response.data[0].embedding
    input_embedding = normalize(np.array([input_embedding]).astype('float32'))

    distances, indices = index.search(input_embedding, k)

    for i, idx in enumerate(indices[0]):
        print(f"similarity: {distances[0][i]:.4f}\nmatching text: \n{chunks[idx]}\n")
```

我们可以使用这个函数来检索一些文本。例如，我们可以检索一些与“VideoAnalytica dataset”最相似的文本块。

We can use this function to retrieve some text. For example, we can retrieve some text blocks that are most similar to "VideoAnalytica dataset".

```
In [12]: input_text = "VideoAnalytica dataset"

matched_texts = match_text(input_text=input_text, index=index, chunks=chunks, k=2)

similarity: 0.5244
matching text:
oring mechanisms should be put in place to minimize risks of unpredictable behaviors in real-world scenarios. Our
"VideoAnalytica" dataset is collecte

similarity: 0.4166
matching text:
tassets. However, there is much more than could be included in a single dataset or workshop. We would argue that the
re is a need for more approaches or
```

Embedding Demo 调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_embedding\\_pysdk.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_embedding_pysdk.ipynb)



## 与 Langchain 结合

在 Langchain 中，仅需将

*ChatOpenAI* 替换为 *ChatZhipuAI*

即可更换为 GLM 系列模型

```
from langchain_community.chat_models.zhipuai import ChatZhipuAI
from langchain.schema import HumanMessage, SystemMessage

messages = [

    SystemMessage(
        content="You are a smart assistant, please reply to me smartly."
    ),
    HumanMessage(
        content="There were 9 birds on the tree. The hunter shot and killed
one. How many birds were there?"
    ),
]

llm = ChatZhipuAI(
    temperature=0.95,
    model="glm-4"
)

llm(messages)
```

GLM-4 + Langchain 调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm\\_langchain.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/basic/glm_langchain.ipynb)

## 与 TaskWeaver 结合

在 TaskWeaver 中

仅需修改

[https://github.com/microsoft/TaskWeaver/blob/main/project/taskweaver\\_config.json](https://github.com/microsoft/TaskWeaver/blob/main/project/taskweaver_config.json)

```
{  
  "llm.api_base": "https://open.bigmodel.cn/api/paas/v4",  
  "llm.api_key": "your keys",  
  "llm.api_type": "zhipuai",  
  "llm.model": "glm-4",  
  "llm.embedding_model": "embedding-2",  
  "llm.embedding_api_type": "zhipuai"  
}
```

中的配置文件即可使用 GLM-4 模型

Demo调用对应实战代码:

[https://github.com/MetaGLM/glm-cookbook/blob/main/demo/agent/glm\\_taskweaver.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/demo/agent/glm_taskweaver.ipynb)

# Demo: 使用 GLM-4 + Langchain 对简历进行信息抓取

接着，我们就可以调用 GLM-4 模型，通过模型对简历进行抓取和提取关键信息，获得有效的内容和答案。

```
In [10]: from langchain_community.chat_models import ChatZhipuAI
```

```
for question in question_prompt:
    messages = chat_template.format_messages(resume=data)
    messages.append(
        HumanMessage(
            content=question
        )
    )
    llm = ChatZhipuAI(
        temperature=0.01,
        model="glm-4",
        max_tokens=4096,
        stream=False,
    )
    messages.append(
        AIMessage(
            content=llm(messages).content
        )
    )
    print(messages[-1].content)
```

候选人李明浩读过的大学有：

1. 清华大学，专业是计算机科学与技术。
2. 北京邮电大学，专业是信息技术。

## 3. 结果分析

通过大模型，我们可以顺利的抽取简历中的关键信息，包括教育背景、工作经历等。这样，我们就可以通过简单的代码，完成简历信息的抽取任务。这是一个开放性的demo，意味着你可以自己选择其他任务来接着完成这个场景的研究。

姓名：李明浩  
性别：男  
生日：1985-3-15  
电话：18888888888  
毕业日期：2007年7月  
EMAIL：minghao.li@163.com  
身体状况：良好

### 教育背景

2003.9 - 2007.7 清华大学 计算机科学与技术专业  
2008.9 - 2010.7 北京邮电大学 信息技术专业

### 语言水平

英语听说读写能力良好

### 工作经历

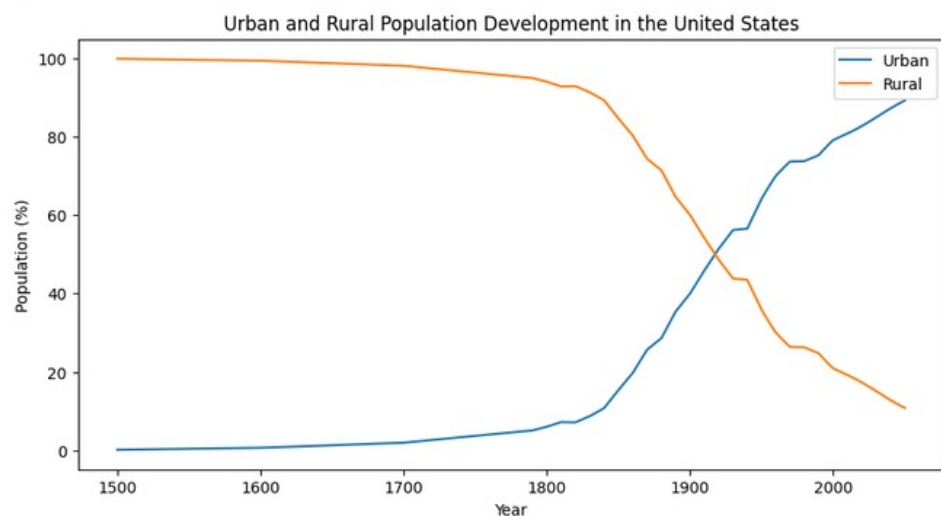
2019年6月 至今 TENCENT（云计算部门）职位：高级技术专家

Demo调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/demo/glm\\_information\\_extraction.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/demo/glm_information_extraction.ipynb)



## Demo: 使用 GLM-4 调用代码 对数据分析



```
Out[5]: [{'role': 'system',
  'content': "\nYou are a data analyst, you will have a code execution tool, you
need to write a python script, the tool will execute your content and return the
results, now please analyze my csv file.\nI will provide you with some informatio
n about the csv, which is as follows:\n\n      info: Changes in the urban-rural p
opulation ratio in major regions of the world from 1500 to 2050. The csv contains
five columns, namely\n      column info: Entity,Code,Year, Urban population (%) lo
ng-run with 2050 projections (OWID),Rural population (%) long-run with 2050 proje
ctions (OWID)\n      path : 'data/urban-rural-population.csv'\n\nEach column has
some data, which you need to read through python code.\nNow, please follow my req
uirements, write the code appropriately, and analyze my csv file.\nI will provide
you with the code to execute the tool, you just need to write the code according
to my requirements.\nAll answers must be provided after querying the csv I provid
ed. Your return must be executable python code and no other content.\nThinking st
ep by step, here's my request, let's get started:\n"},
{'role': 'user',
  'content': 'Read csv and draw the distribution of urban and rural population de
velopment in the United States'},
{'role': 'tool',
  'content': 'Code finished successfully!',
  'tool_call_id': 'call_8313813618511317295'}]
```

使用Python完成最简单的代码解释器功能，  
并让模型通过执行代码来完成对CSV表格  
分析

- 图表绘制
- 数学计算
- 大型表格
- 规律总结

Demo调用对应实战代码：

[https://github.com/MetaGLM/glm-cookbook/blob/main/demo/glm\\_csv\\_data\\_analysis.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/demo/glm_csv_data_analysis.ipynb)

## 价格计算

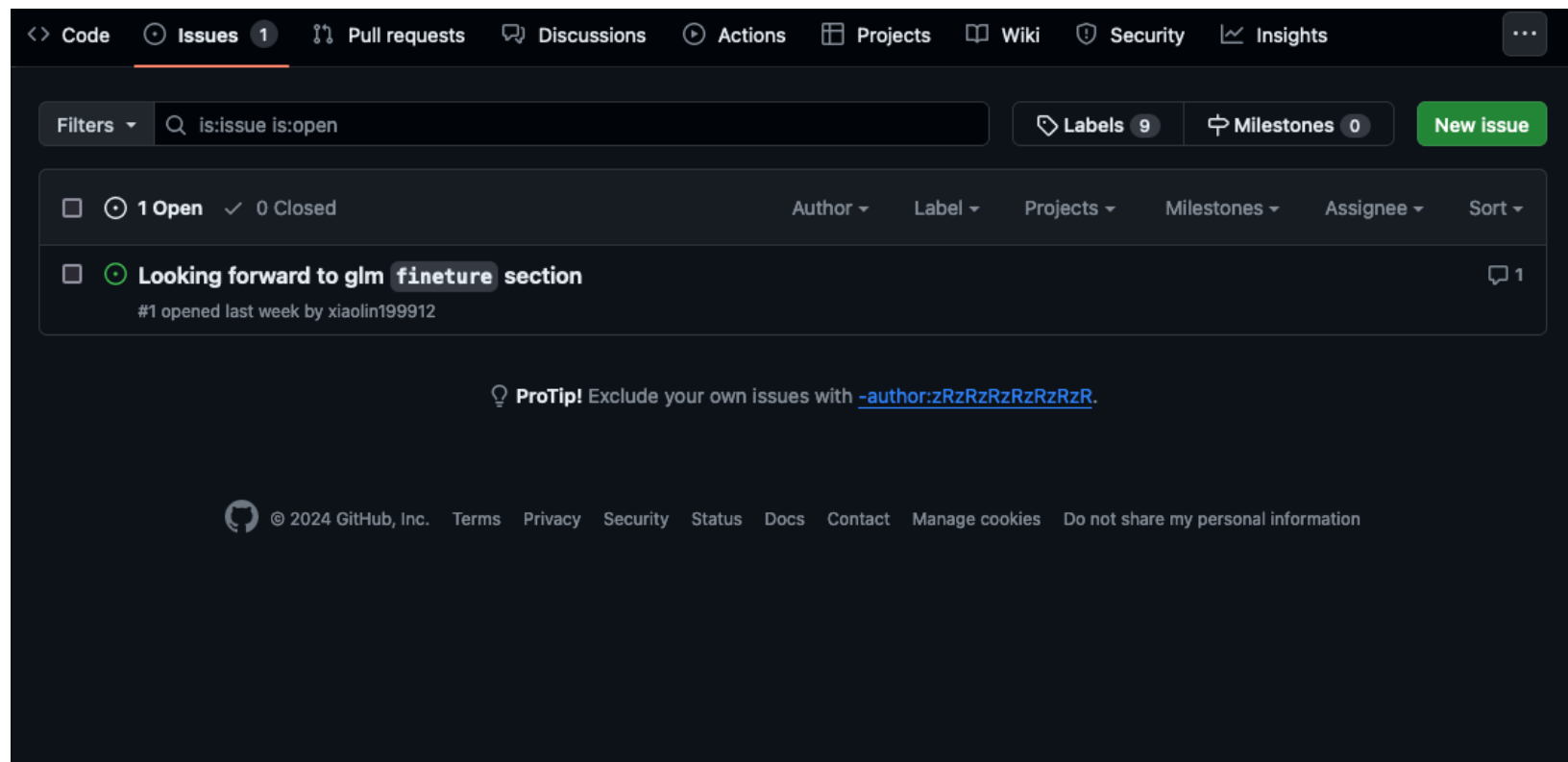
记住以下要点：

1. 输入输出都进行计费
2. Funtion\_Call 不计费
3. 价格体现在token使用上，GLM-4 token消耗是GLM-3的20倍
4. 充值统一转换为token单位

模型	说明	上下文长度	单价
GLM-4	提供了更强大的问答和文本生成能力。适合于复杂的对话交互和深度内容创作设计的场景。	128K	0.1元 / 千tokens
GLM-4V	实现了视觉语言特征的深度融合，支持视觉问答、图像字幕、视觉定位、复杂目标检测等各类图像理解任务。	2K	0.1元 / 千tokens ②
GLM-3-Turbo	适用于对知识量、推理能力、创造力要求较高的场景，比如广告文案、小说写作、知识类写作、代码生成等。	128K	0.005元 / 千tokens

## 提出意见: Github Issue

与 ChatGLM3-6B 仓库相同，仅需在 Github 提出留言和意见，可以促进社群 Github 添加新 Demo，疑难解答



Demo调用对应实战代码:

[https://github.com/MetaGLM/glm-cookbook/blob/main/demo/glm\\_information\\_extraction.ipynb](https://github.com/MetaGLM/glm-cookbook/blob/main/demo/glm_information_extraction.ipynb)

谢谢大家