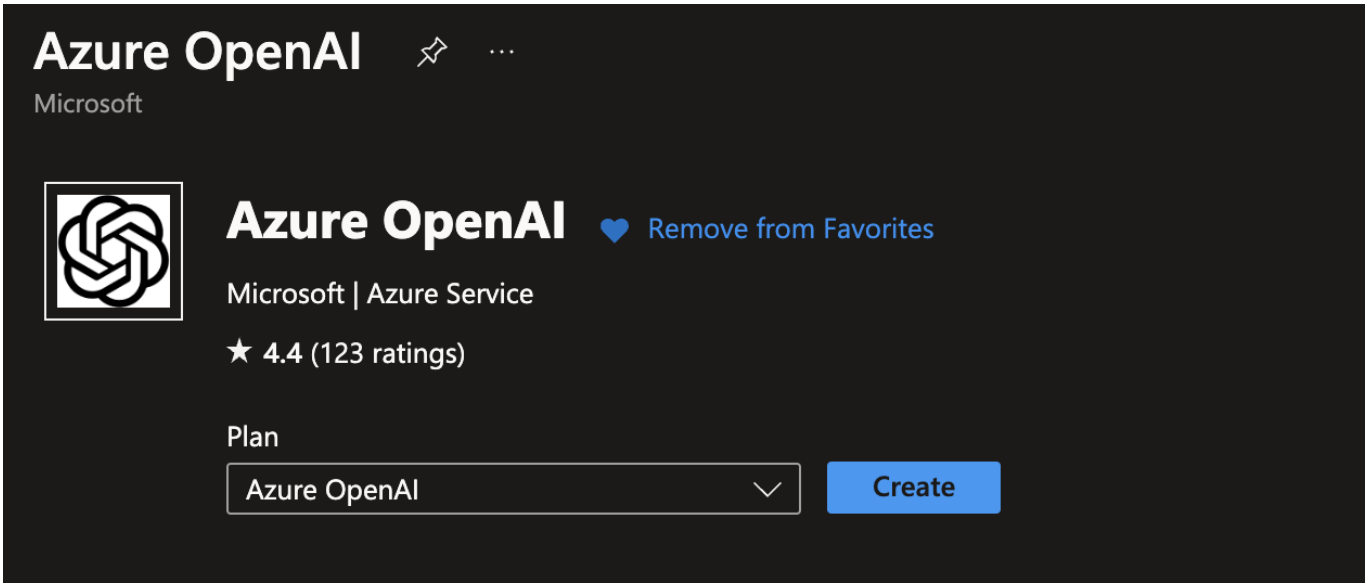
第一章：用 SDK 访问 Azure OpenAI Service

在前言部分，我们了解了大型语言模型的相关知识，下面我想谈谈如何使用大型语义模型，在未进入 Semantic Kernel 之前，我更希望大家看看如何正确通过 SDK 去访问 Azure OpenAI Service 上的 Azure OpenAI 模型。

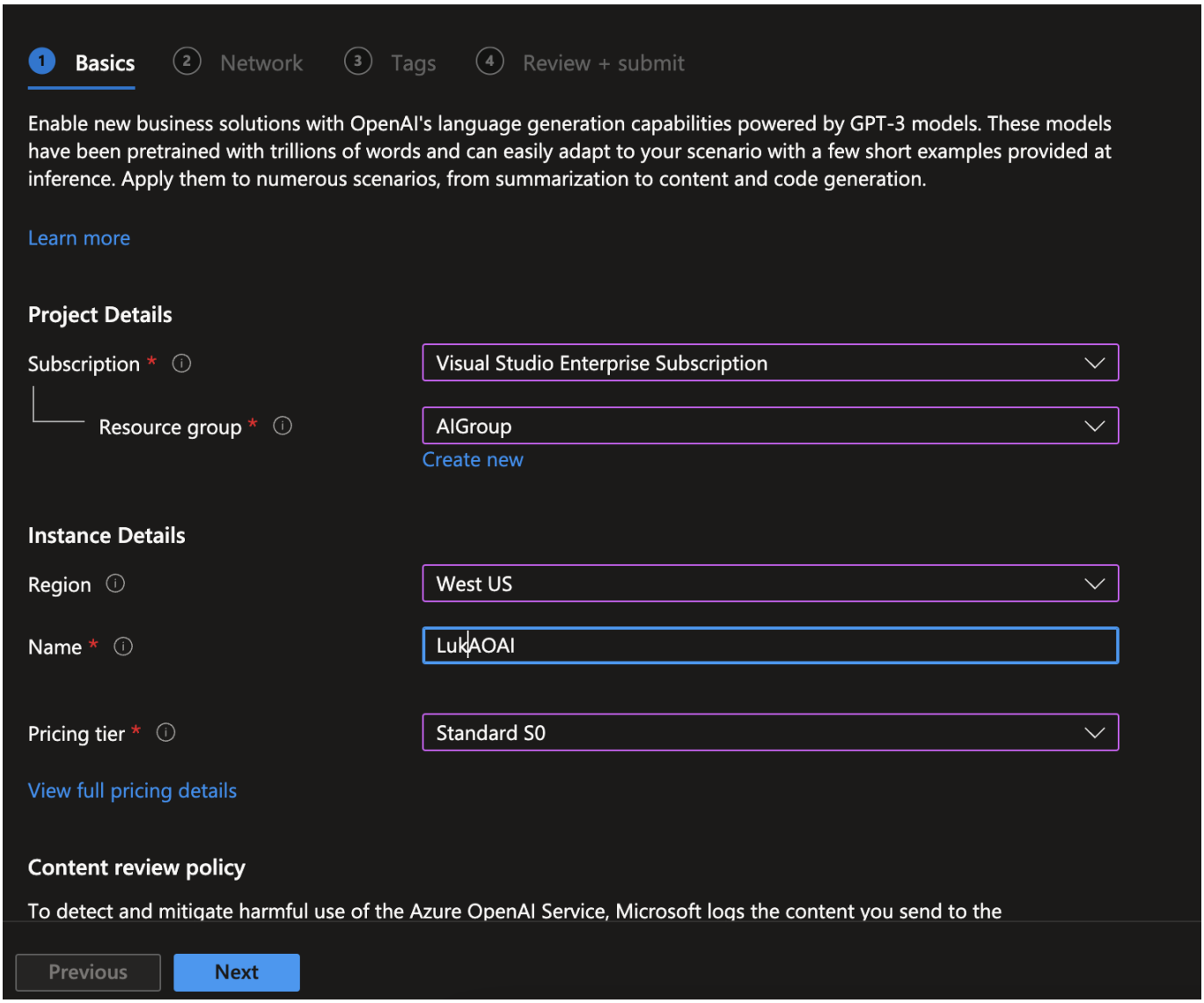
准备工作：在 Azure OpenAI Studio 部署模型

部署 Azure OpenAI 模型很简单，在申请成功 Azure OpenAI Service 后，通过在 Azure Portal 创建资源进行部署。以下是相关步骤：

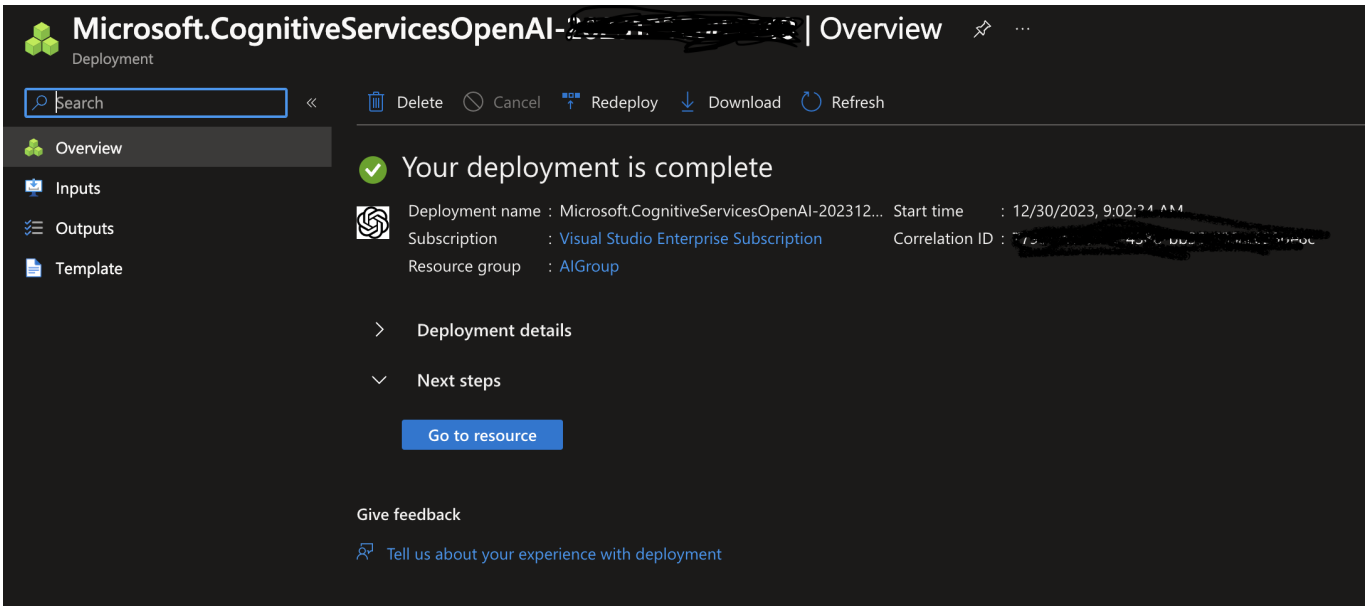
1. 在 Azure Portal 选择 Azure OpenAI 创建资源



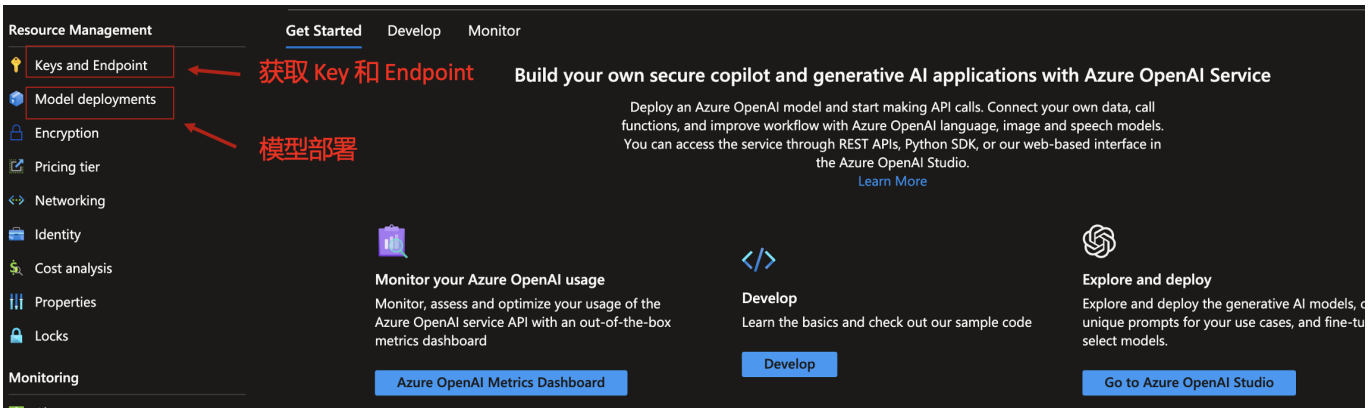
选择 'Create' 后，配置好 Azure OpenAI 所在区域，需要注意：因为资源分布不同，不同区域所拥有的 OpenAI 模型不尽相同，在使用前一定要了解清楚。



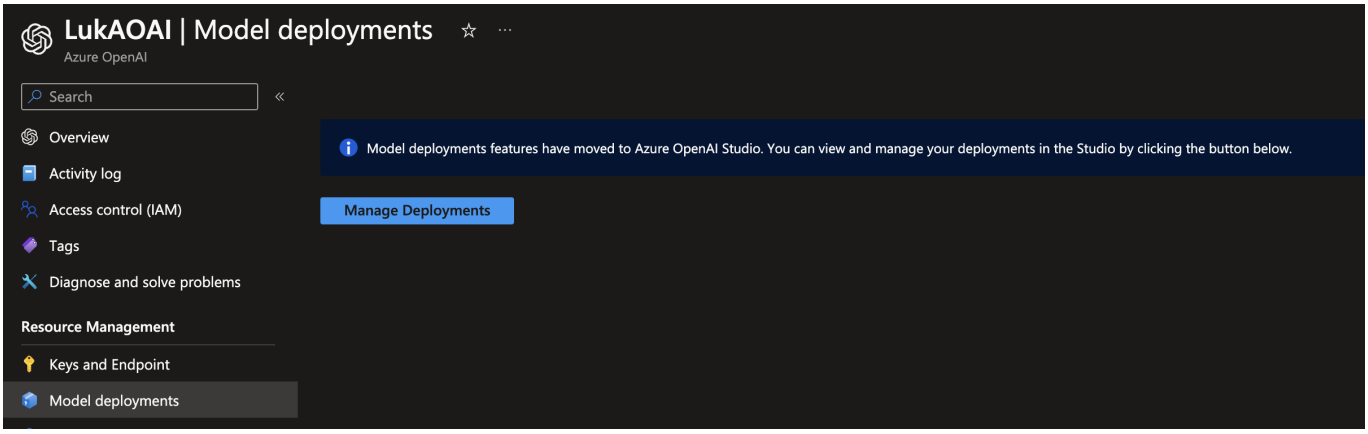
等待片刻，即创建成功



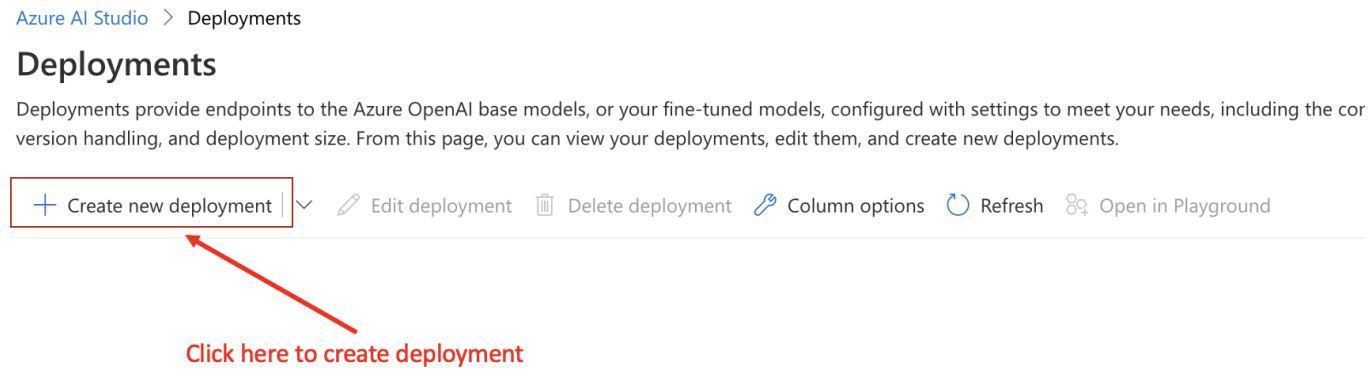
2. 进入创建好的资源，你可以部署模型，以及获取 SDK 调用时需要的 Key，以及 Endpoint



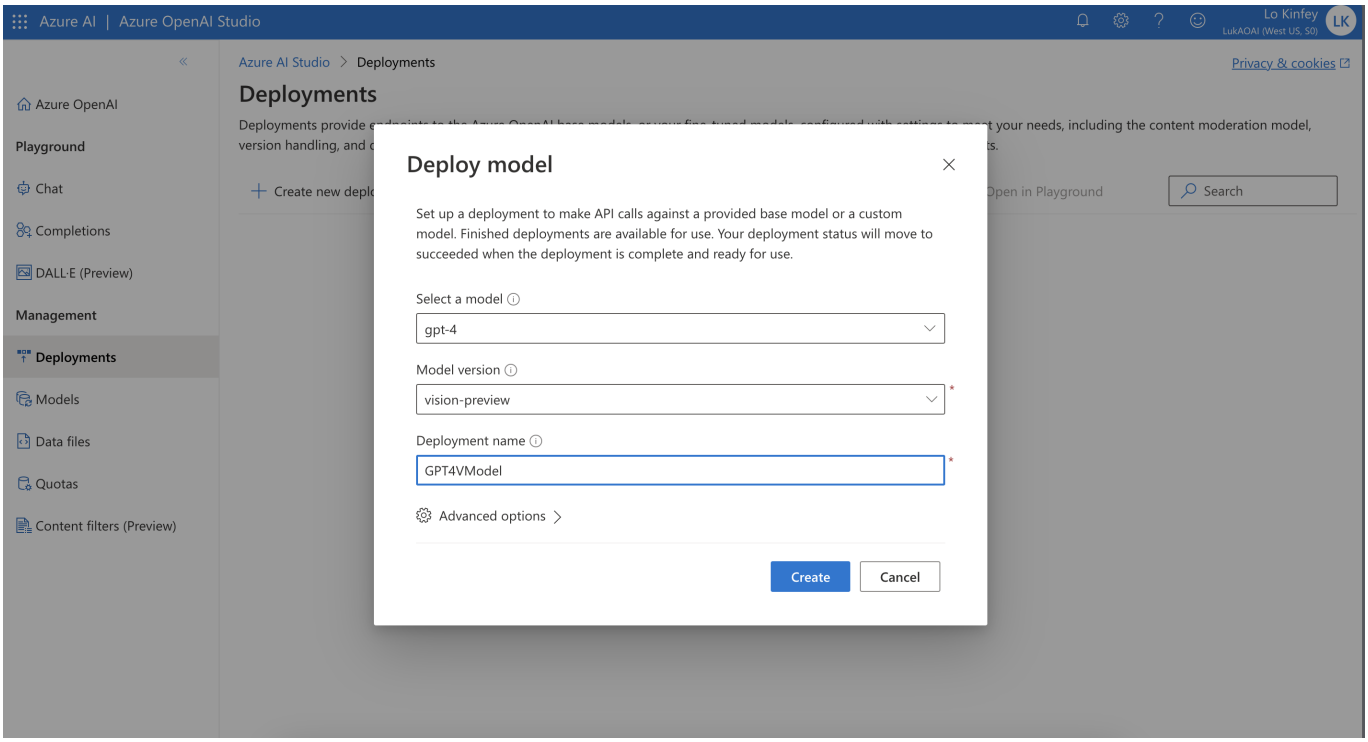
3. 进入 'Model Deployment' 选择 'Management Deployment' 进入 Azure OpenAI Studio



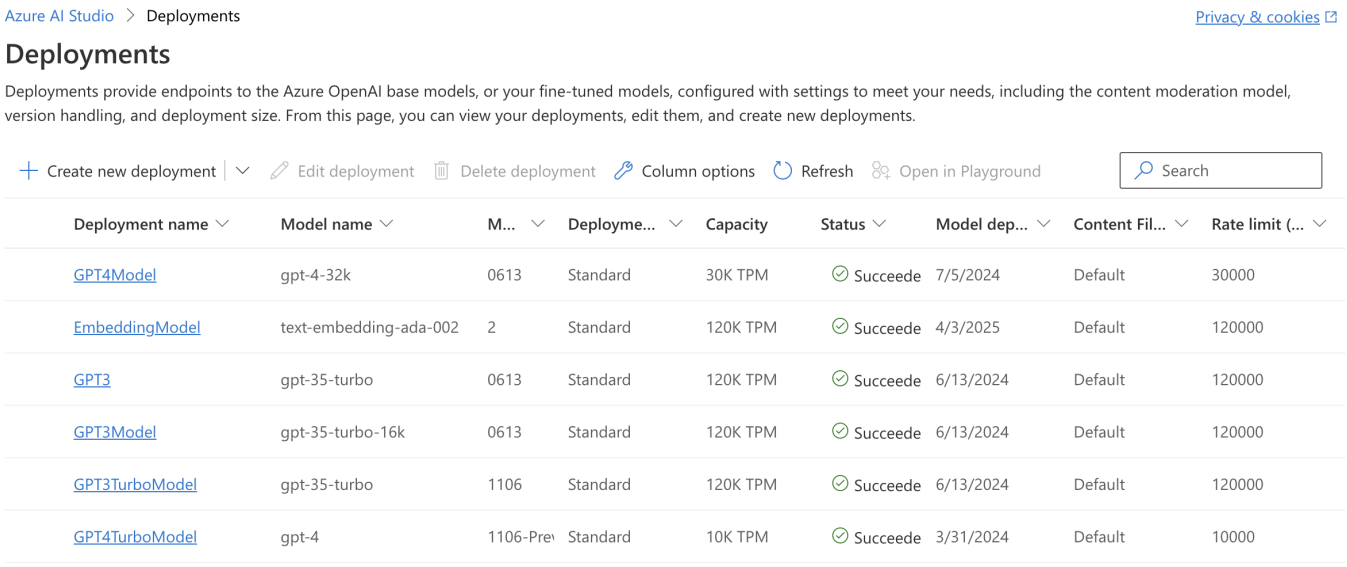
4. 在 Azure OpenAI Studio 部署您的模型



选择你需要的模型



可以看到您的模型列表



恭喜你，成功部署了模型，接下来就可以用 SDK 对接了。

使用 SDK 链接 Azure OpenAI Service

和 Azure OpenAI Service 对接的 SDK，针对 Python 版本有 OpenAI 发布的 SDK， 针对 .NET 也有 Microsoft 发布的 SDK。作为初学者建议都在 Notebook 环境下使用，以便更容易理解执行的关键步骤。

关于 Python SDK

OpenAI 发布的官方 Python SDK, 支持链接 OpenAI 和 Azure OpenAI Service。现在 OpenAI SDK 发布了 1.x 版本，但在市面上很多都在用 0.2x 版本。***本次课程内容都会基于 OpenAI SDK 1.x 版本，并使用 Python 3.10.x***。

安装方式

```
! pip install openai -U
```

关于 .NET SDK

Microsoft 发布基于 Azure OpenAI Service 的 SDK，你可以通过 Nuget 获取最新的包来完成 .NET 生成式 AI 应用。***本次课程内容都会基于 .NET 8 以及最新的 Azure.AI.OpenAI SDK 来展示例子，当然也会使用 Polyglot Notebook 作为环境***

安装方式

```
#r "nuget: Azure.AI.OpenAI, *-*"
```

在上面我们已经配置好基于 .NET / Python 的 SDK 环境，接下来我们需要创建好链接的类以完成相关的初始化工作

针对 .NET 环境的初识化

```
string endpoint = "Your Azure OpenAI Service Endpoint";
string key = "Your Azure OpenAI Service Key";

OpenAIClient client = new(new Uri(endpoint), new AzureKeyCredential(key));
```

针对 Python 环境的初始化

```
client = AzureOpenAI(
  azure_endpoint = 'Your Azure OpenAI Service Endpoint',
  api_key='Your Azure OpenAI Service Key',
  api_version="Your Azure OpenAI API version"
)
```

Azure OpenAI Service 的常用 API 以及 SDK 调用方法

1. 文本补全 API

这是基于 gpt-35-turbo-instruct 模型，对于文本补全是非常重要的接口

.NET 场景下的文本补全

```
CompletionsOptions completionsOptions = new()
{
    DeploymentName = "gpt-35-turbo-instruct",
    Prompts = { "Can you introduce what is generative AI ?" },
};

Response<Completions> completionsResponse =
client.GetCompletions(completionsOptions);

string completion = completionsResponse.Value.Choices[0].Text;
```

Python 场景下的文本补全

```
start_phrase = 'Can you introduce what is generative AI ?'

response = openai.Completion.create(engine=deployment_name,
prompt=start_phrase, max_tokens=1000)

text = response['choices'][0]['text'].replace('\n', '').replace(' .',
'.').strip()
```

2. Chat API

这是基于 gpt-35-turbo 和 gpt-4 模型，基于聊天场景的接口

.NET 场景下的 Chat

```
var chatCompletionsOptions = new ChatCompletionsOptions()
{
    DeploymentName = "gpt-4",
    Messages =
    {
        new ChatRequestSystemMessage("You are my coding assistant."),
        new ChatRequestUserMessage("Can you tell me how to write python
flask application?"),
    },
    MaxTokens = 10000
};

Response<ChatCompletions> response =
```

```
client.GetChatCompletions(chatCompletionsOptions);
```

Python 场景下的 Chat

```
response = client.chat.completions.create(
    model="gpt-35-turbo", # model = "deployment_name".
    messages=[
        {"role": "system", "content": "You are my coding assistant."},
        {"role": "user", "content": "Can you tell me how to write python
flask application?"}
    ]
)

print(response.choices[0].message.content)
```

3. 文生图 API

基于 DallE 3 模型，文生图的场景

.NET 场景下的文生图

```
Response imageGenerations = await client.GetImageGenerationsAsync(
        new ImageGenerationOptions()
        {
            DeploymentName = "Your Azure OpenAI Service Dall-E 3 model
Deployment Name",
            Prompt = "Chinese New Year picture for the Year of the
Dragon",
            Size = ImageSize.Size1024x1024,
        });
```

Python 场景下的文生图

```
result = client.images.generate(
    model="dalle3",
    prompt="Chinese New Year picture for the Year of the Dragon",
    n=1
)
```

```
json_response = json.loads(result.model_dump_json())
```

4. Embeddings API

基于 text-embedding-ada-002 模型，基于向量转换的实现

.NET 场景下的 Embeddings

```
EmbeddingsOptions embeddingOptions = new()
{
    DeploymentName = "text-embedding-ada-002",
    Input = { "Kinfey is Microsoft Cloud Advocate" },
};

var returnValue = openAIClient.GetEmbeddings(embeddingOptions);

foreach (float item in returnValue.Value.Data[0].Embedding.ToArray())
{
    Console.WriteLine(item);
}
```

Python 场景下的 Embeddings

```
client.embeddings.create(input = ['Kinfey is Microsoft Cloud Advocate'],
model='text-embedding-ada-002 model').data[0].embedding
```

例子

以下列出了和上述接口相关的例子，请根据您的语言环境进行学习

Python 例子 请点击访问这里

.NET 例子 请点击访问这里

小结

我们用最原始，也是最基础的 SDK 与 Azure OpenAI Service 打交道，这也是我们面向生成式人工智能编程的第一步，在没有使用框架下可以更快速地理解不同接口，也为我们进入 Semantic Kernel 打下基础。