Microsoft

Develop Generative AI Solutions with Azure OpenAI Service

Agenda

- Get started with Azure OpenAI Service
- Develop apps with Azure OpenAI Service
- Apply prompt engineering with Azure OpenAI Service
- Use your own data with Azure OpenAI Service

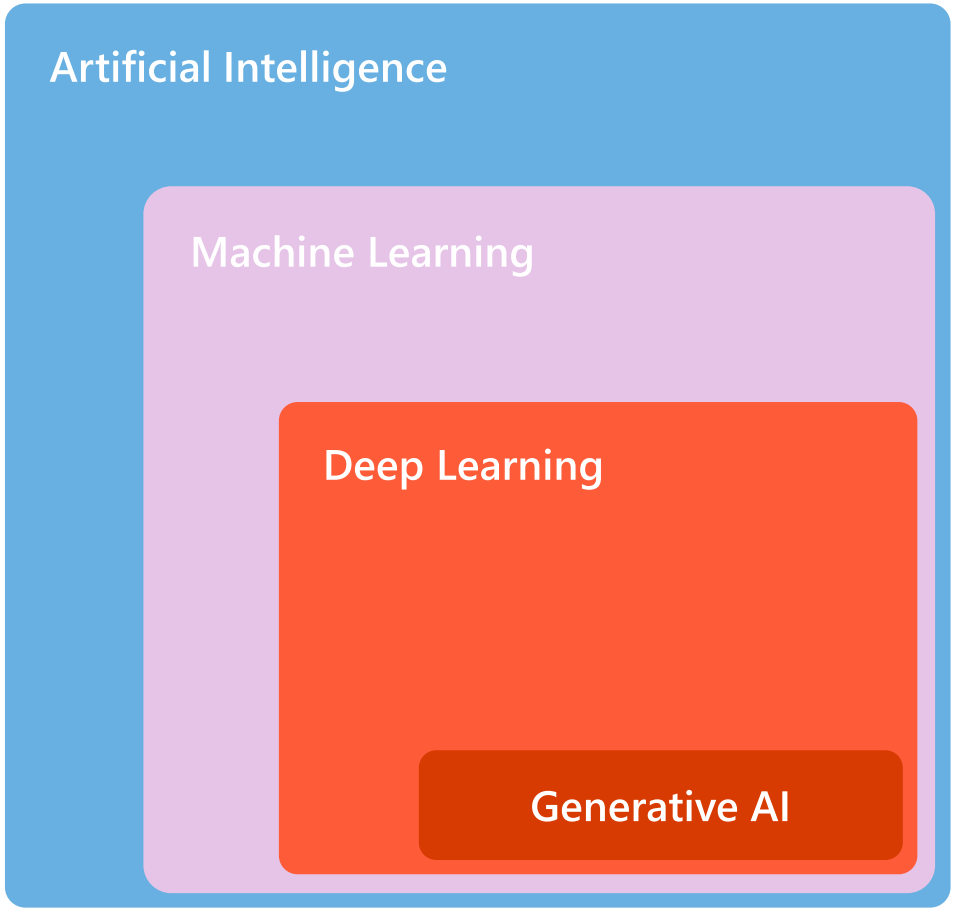
Get started with Azure OpenAI Service

Learning Objectives

After completing this module, you will be able to:

1. Describe what generative AI is
2. Provision a resource and deploy a model
3. Use Azure AI Studio

What is generative AI?



1950s

Artificial Intelligence
the field of computer science that seeks to create intelligent machines that can replicate or exceed human intelligence

1990s

Machine Learning
subset of AI that enables machines to learn from existing data and improve upon that data to make decisions or predictions

2010s

Deep Learning
a machine learning technique in which layers of neural networks are used to process data and make decisions

2020s

Generative AI
Create new written, visual, and auditory content given prompts or existing data.

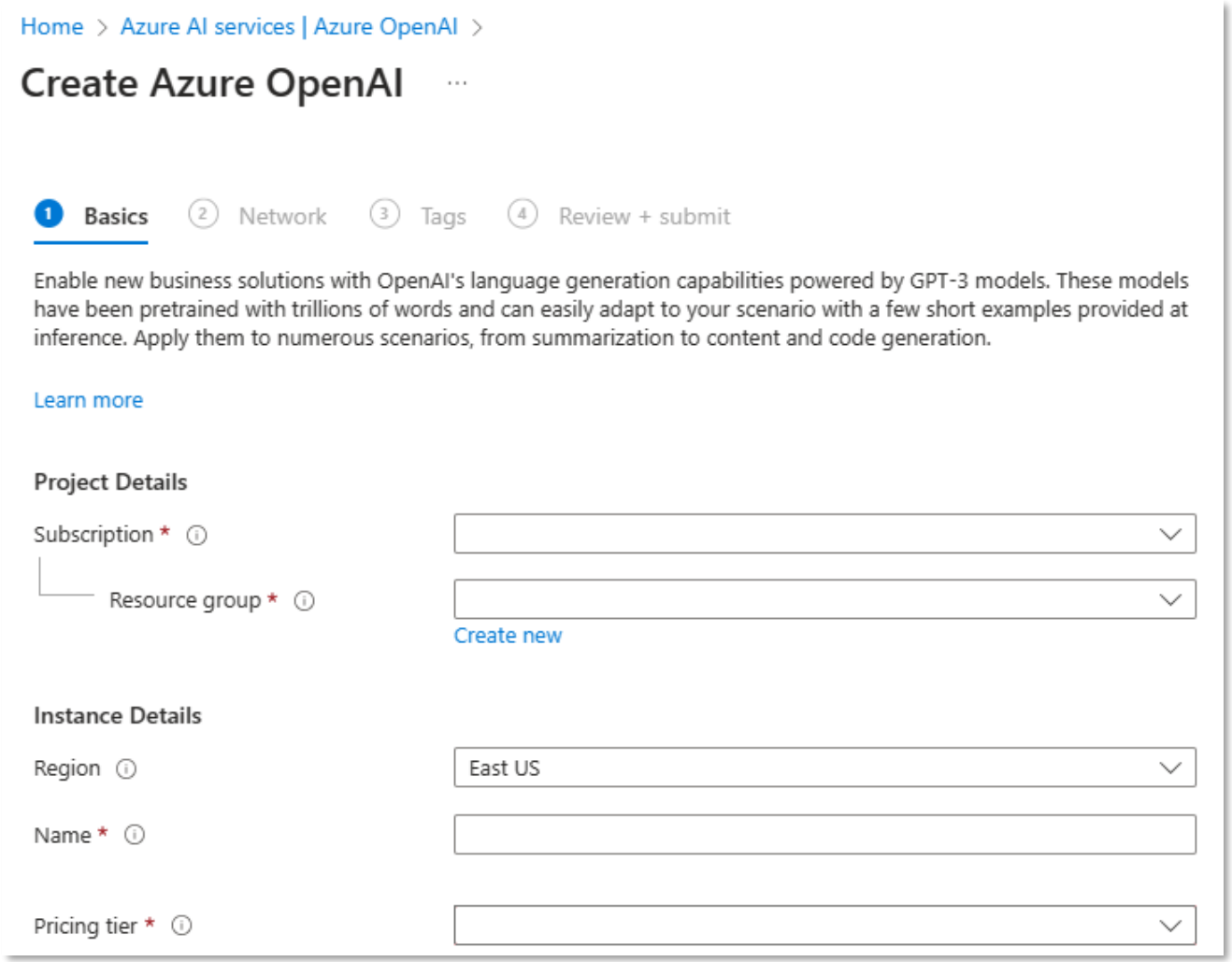
Provision an Azure OpenAI resource in Azure

Create an Azure OpenAI resource in the Azure portal

Deploy a resource in Azure, then deploy and use a model to in Azure AI Studio

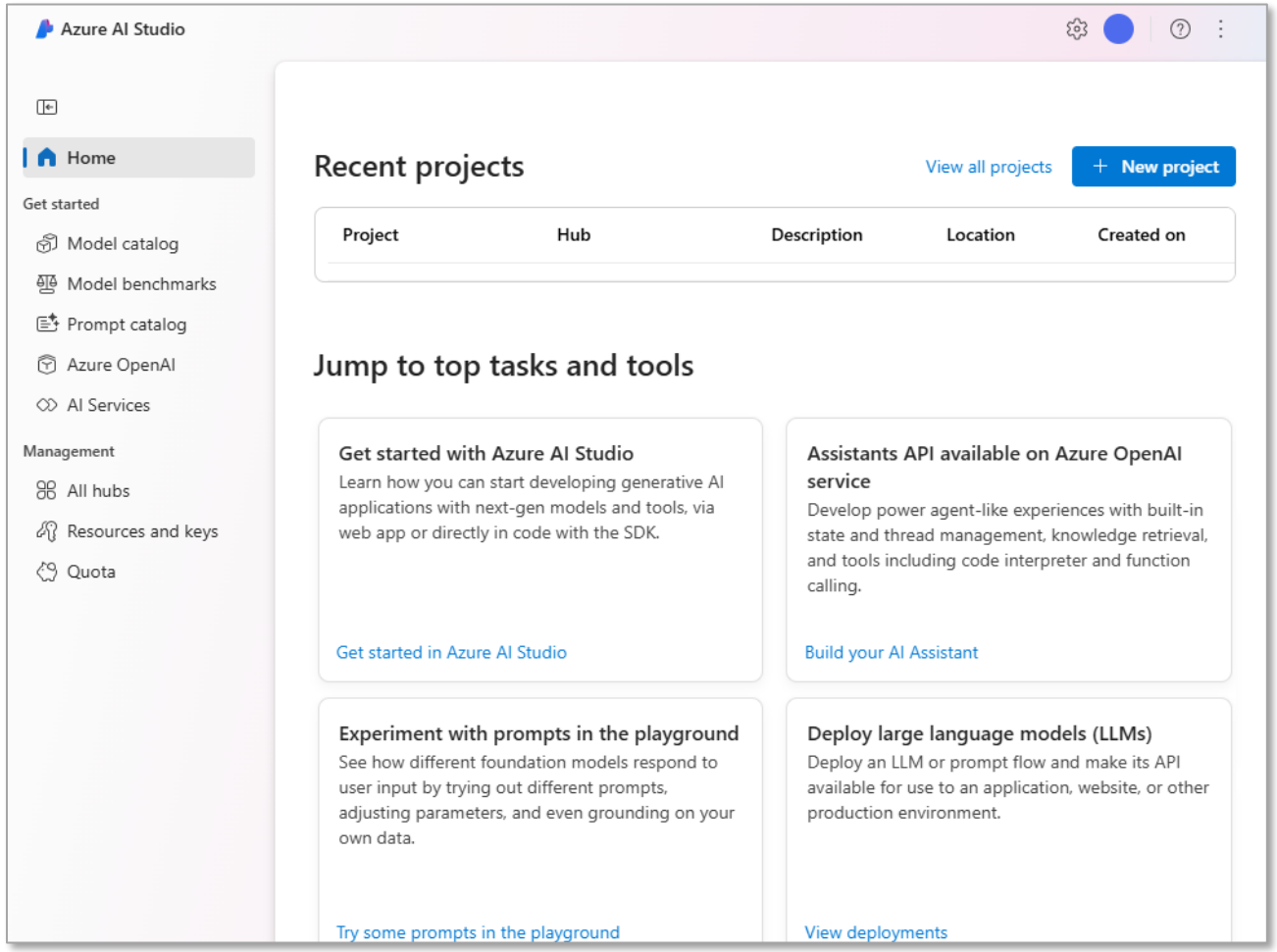
Alternatively, use the Azure CLI

```
az cognitiveservices account create \
-n MyOpenAIResource \
-g MyResourceGroup \
-l eastus \
--kind OpenAI \
--sku s0 \
--subscription subscriptionID
```



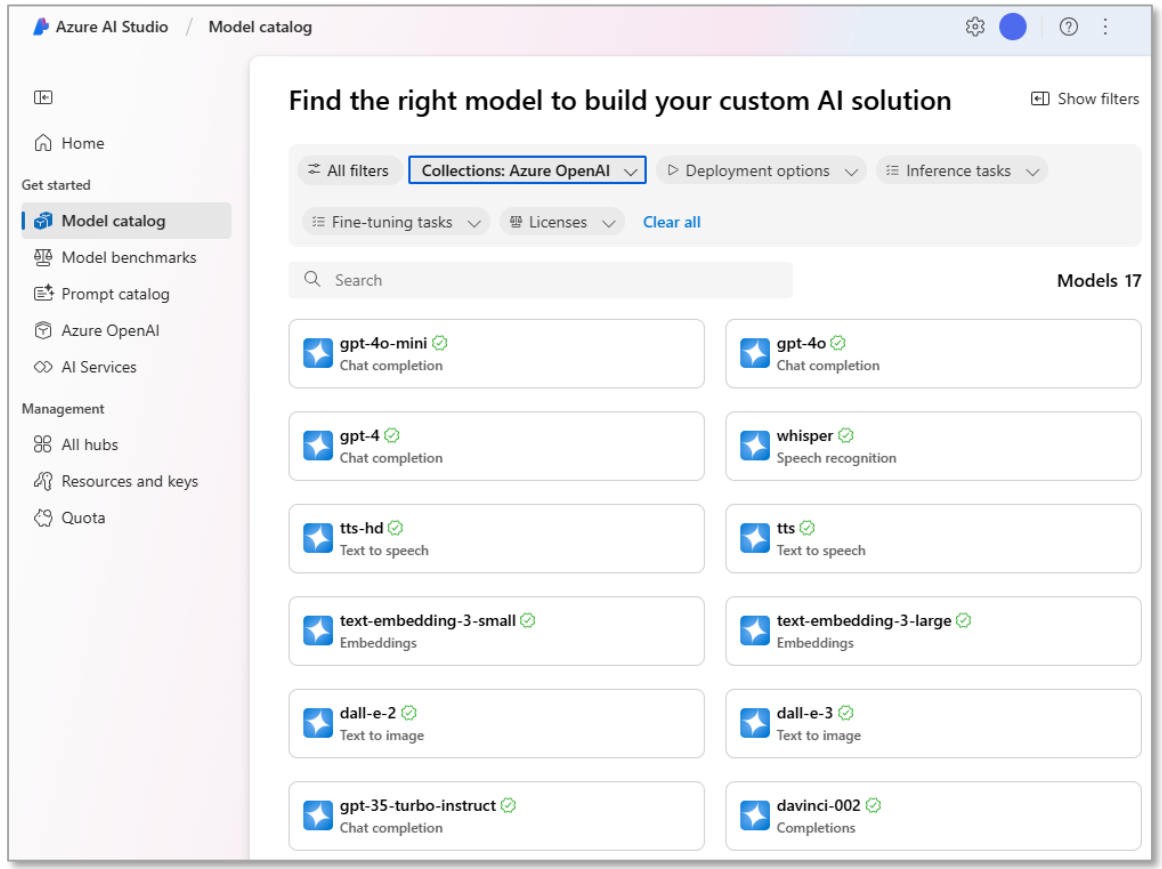
Azure AI Studio

- Web portal for working with AI models: https://ai.azure.com/
- View and deploy base models
- Connect to other AI Services or your own data source
- Manage fine tuning and data files for custom models
- Test models in visual playgrounds:
  - **Chat** (GPT-3.5-Turbo and later models)
  - **DALL-E** (Image generations)
  - **Assistants** (Custom and Copilot-like experiences)



Types of generative AI model

Model Family	Description
GPT-4	Newest, most capable chat-based models for language and code generation
GPT-3	Natural language and code-generation models
Embeddings	Models that use embeddings for specific tasks (similarity, text search, and code search)
DALL-E	Image-generation model (*preview, restricted regions*)

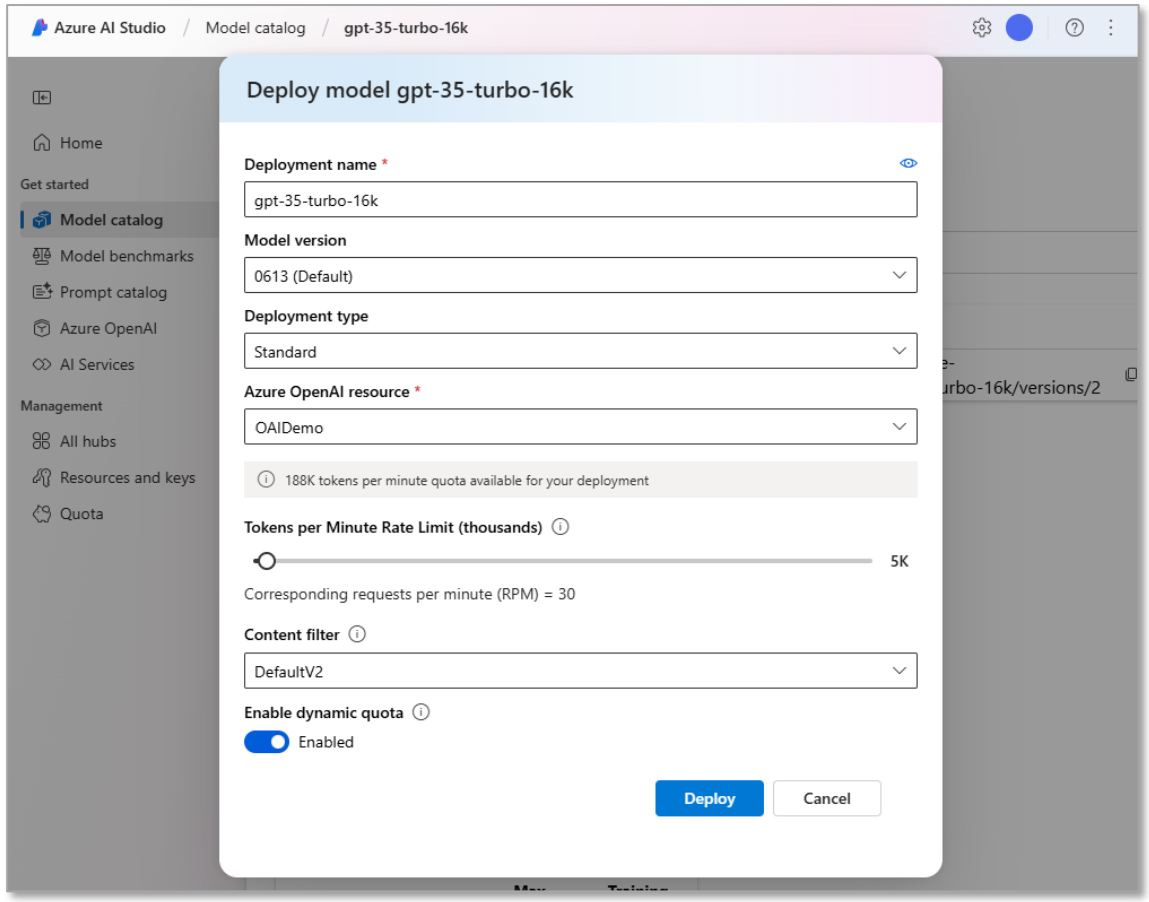


Deploying generative AI models

Deploy a model in Azure AI Studio to use it

- You can deploy one or more instances of each available model
- The number of deployments depends on your quota, which you can see in the portal
- Alternatively, use the Azure CLI

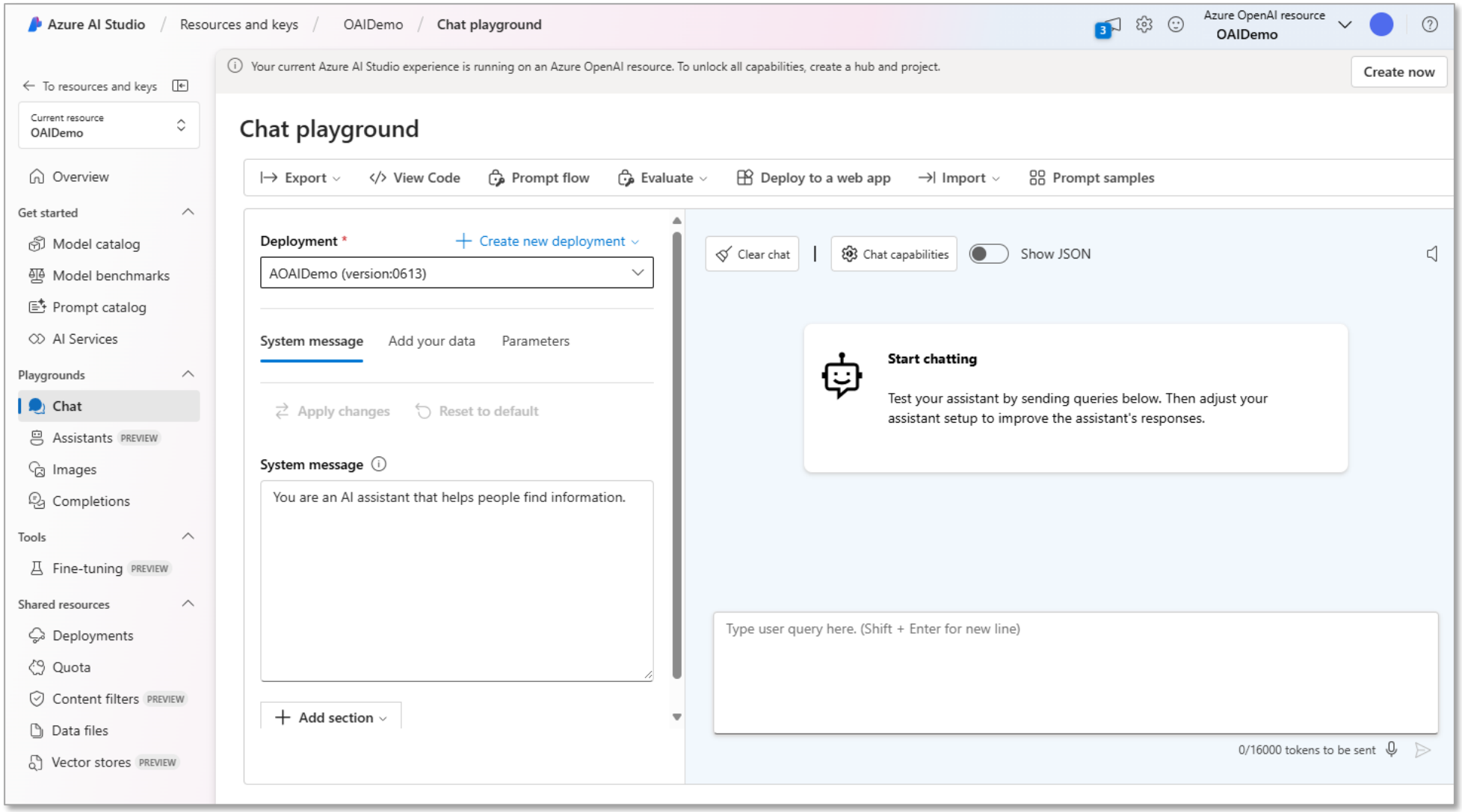
```
az cognitiveservices account deployment create \
    -g myResourceGroupName \
    -n MyOpenAIResource \
    --deployment-name my-gpt-model \
    --model-name gpt-35-turbo \
    --model-version "0613"  \
    --model-format OpenAI \
    --scale-settings-scale-type "Standard"
```



Using prompts to get completions from models

Task	Prompt	Completion
Classifying content	Tweet: I enjoyed the training course. Sentiment:	Positive
Generating new content	Write a poem about databases	Databases, oh databases, You keep our information safe, From the small to the large, You store our data in a place.
Transformation/Translation	English: Hello French:	Bonjour
Summarization	Scotland is [*long description of Scotland…*] Summarize the previous text	Scotland is [*summarized description…*]
Continuation	One way to grow tomatoes is to	start with seeds…
Question answering	How many moons does Earth have?	Earth has one moon.
Chat	*Setup, followed by messages…*	*A sequence of relevant responses*

Testing models in Azure AI Studio playground



[Optional Exercise]-Get started with Azure OpenAI Service

Deploy Azure OpenAI resource and model

Test deployed model in playground

Azure OpenAI Service models - Azure OpenAI | Microsoft Learn

Develop apps with Azure OpenAI Service

Learning Objectives

After completing this module, you will be able to:

1. Integrate Azure OpenAI into your app
2. Use the REST API
3. Use language specific SDKs

Integrating Azure OpenAI into your app

Applications submit prompts to deployed models. Responses are completions.

Three REST API endpoints:

- **Completion** - model takes an input prompt, and generates one or more predicted completions
- **Embeddings** - model takes input and returns a vector representation of that input
- **ChatCompletion** - model takes input in the form of a chat conversation (where roles are specified with the message they send), and the next chat completion is generated

ChatCompletion will be the endpoint we focus on for this course

Use **Completion** and **Embeddings** with GPT-3 based models
Use **ChatCompletion** with GPT-35-Turbo and later models

Using the Azure OpenAI REST API

Completion Endpoint

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**completions**

```
{
  "prompt": "Your favorite Shakespeare
             play is",
  "max_tokens": 5
}
```

```
{
    "id": “1234….",
    "object": "text_completion",
    "created": 1679001781,
    "model": “gpt-35-turbo",
    "choices": [
        {
            "text": "Macbeth",
            "index": 0,
            "logprobs": null,
            "finish_reason": "stop"
        }
    ]
}
```

Using the Azure OpenAI REST API

Embedding Endpoint

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**embeddings**

```
{
  "input": "The food was delicious and
            the waiter was very
            friendly..."
}
```

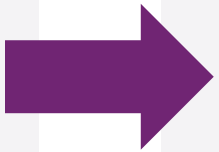
```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "embedding": [
        0.01729906880664825 23,
        ....
        0.013454434883475304 2,
      ],
      "index": 0
    }
  ],
  "model": "text-embedding-ada:002"
}
```

Using the Azure OpenAI REST API

ChatCompletion Endpoint

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**chat/completions**

```
{
  "messages":[
    {"role": "system",
     "content": "You are an assistant
      that teaches people about AI."},
    {"role": "user",
     "content": "Does Azure OpenAI
          support multiple languages?"},
    {"role": "assistant",
     "content": "Yes, Azure OpenAI
          supports several languages."},
    {"role": "user",
     "content": "Do other Cognitive
         Services support translation?"}
  ]
}
```



```
{
    "id": “unique_id", "object": "chat.completion",
    "created": 1679001781, "model": "gpt-35-turbo",
    "usage": { "prompt_tokens": 95,
               "completion_tokens": 84, "total_tokens": 179},
    "choices": [
          {"message":
               {  "role": "assistant",
                   "content": "Yes, other Azure Cognitive
                       Services also support translation…"},
            "finish_reason": "stop",
            "index": 0}
    ]
}
```

Using the Azure OpenAI SDKs

Language specific SDKs are available for use in your applications, in both C# and Python.

Code structure follows a similar pattern for both languages.

Parameters such as *Max Response* and *Temperature* are defined in the chat options.

Both synchronous and asynchronous API versions are available.

Pseudo code structure:

```
<include library>

<create client>

<define chat messages and options>

<send request>

<extract response content>
```

Exercise: Integrate Azure OpenAI into your app

Configure your application to connect to Azure OpenAI

Test your app by sending a request to your deployed model

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-azure-openai-api

Apply prompt engineering with Azure OpenAI Service

Learning Objectives

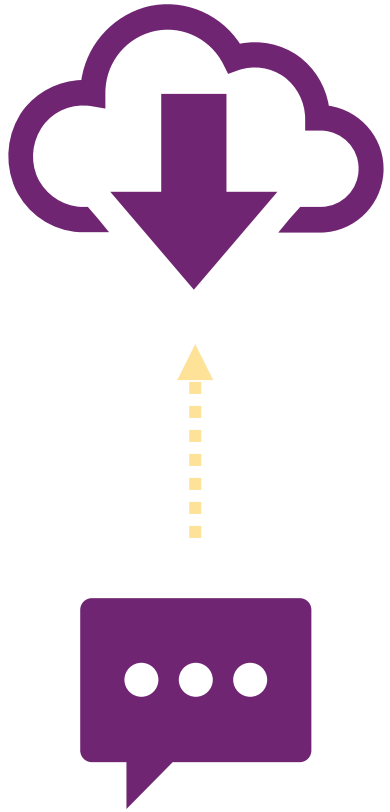
After completing this module, you will be able to:

1. Understand what prompt engineering is
2. Understand considerations for different endpoints
3. Explore different techniques of prompt engineering

What is Prompt Engineering?

Constructing prompts to:

- Maximize relevancy and accuracy of completions
- Specify formatting and style of completions
- Provide conversational context
- Mitigate bias and improve fairness



Providing clear instructions

Write a product description for a new water bottle

Introducing the latest addition to our product line - the innovative and eco-friendly water bottle. Made from high-quality, BPA-free materials, this bottle is designed to keep your beverages at the perfect temperature for hours. Its sleek and stylish design makes it the perfect accessory for any outdoor activity or workout session. With its convenient carrying handle and leak-proof lid, you can take it with you wherever you go, without worrying about spills or leaks. Plus, its easy-to-clean design ensures that you can use it again and again, making it the perfect choice for anyone looking for a reliable and sustainable water bottle.

Write a product description for a new water bottle that is 100% recycled. Be sure to include that it comes in natural colors with no dyes, and each purchase removes 10 pounds of plastic from our oceans

Introducing our newest water bottle, the eco-friendly choice for the conscious consumer. Made from 100% recycled materials and available in natural colors with no dyes, this bottle is not only stylish but also sustainable. Each purchase of this bottle removes 10 pounds of plastic from our oceans, making it a responsible choice for those who care about the environment. Its durable and leak-proof design makes it perfect for outdoor activities, while its easy-to-carry handle makes it convenient to take on the go. Join us in our mission to reduce plastic waste and choose this eco-friendly water bottle for your daily hydration needs.

Primary, *supporting*, and *grounding* content

Section markers
(--- or ###)

Primary content to be summarized, translated, etc.

Supporting content to provide clarity / specificity

Grounding content to define scope for questions

Thanks for your interest in our summer school program.

We're looking for your ideas for webinars on topics such as AI, web development, … *<more…>*

Topics I'm very interested in: AI, webinar dates, submission deadlines

Extract the key points from the above email, and put them in a bulleted list

---

Artificial Intelligence (AI) has evolved over many years … *<more…>*

Where and when did the field of AI start?

Cues

- Cues provide a starting point on which the completion builds
- Use cues to set the scope and format for the output
- Cues are particularly useful when generating code. For example:
  - Provide the first keyword in a SQL SELECT statement
  - Define a name and parameters for a Python function

```
---
Review: The writing was good but the movie was too long.
Review: The special effects were terrible.
<more>...
---

Summarize the reviews above.
Most common complaints:
-
```

- The movie was too long.
- The special effects were terrible.

Requesting output composition

Write a table in markdown with 6 animals in it, with their genus and species

```
| Animal           | Genus          | Species        |
| ---------------- |:--------------:|:--------------:|
| Red Panda        | Ailurus        | fulgens        |
| African Elephant | Loxodonta      | africana       |
| Snow Leopard     | Panthera       | uncia          |
| Arctic Fox       | Vulpes         | lagopus        |
| Chimpanzee       | Pan            | troglodytes    |
| Siberian Tiger   | Panthera       | tigris altaica |
```

Using a system message

When using the ChatCompletions API, a system message sets context that can affect completions

```
{"role": "system", "content": "You are a casual, helpful assistant.
                               You will talk like an American old western film character."},
{"role": "user", "content": "Can you direct me to the library?"}
```

```
{"role": "assistant", "content": "Well howdy there, stranger! The library, huh?
                                  Y'all just head down the main road till you hit the town
                                  square. Once you're there, take a left and follow the street
                                  for a spell. You'll see the library on your right, can’t
                                  miss it. Happy trails!"}
```

Conversation history and *few shot* learning

- Including previous messages helps retain the context and style of a conversation
- Starting a conversation with user-defined messages and responses (*few shot* learning) sets examples for the expected responses

```
{"role": "system", "content": "You are an assistant that evaluates the sentiment of customer
feedback."},
{"role": "user", "content": "That was an awesome experience"},
{"role": "assistant", "content": "positive"},
{"role": "user", "content": "I won't do that again"},
{"role": "assistant", "content": "negative"},
{"role": "user", "content": "That was not worth my time"},
{"role": "assistant", "content": "negative"},
{"role": "user", "content": "You can't miss this"}
```

```
{"role": "assistant", "content": "positive"},
```

Chain of thought

Ask a model to break down its response and explain its reasoning

What sport is easiest to learn but hardest to master? Give a step by step approach of your thoughts, ending in your answer

Step 1: Identify the criteria for "easy to learn" and "hard to master" sports.
For a sport to be considered easy to learn, it should have simple rules and require minimal equipment For a sport to be considered hard to master, it should require years of practice to perfect and have a large variety of techniques and strategies…

Step 2: Consider different sports that fit these criteria.
Some sports that could be considered easy to learn but hard to master include:
- Tennis: Simple rules, easy to pick up, but requires years of practice to master all the different shots and strategies.
- Golf: Basic swing mechanics are easy to learn but perfecting them takes a lifetime of practice.
- …

Step 3: Evaluate each sport based on the criteria.
Tennis and golf both require expensive equipment, which could make them less accessible to beginners. In terms of skill ceiling and time required to master, tennis and golf both have a high skill ceiling and require years of practice to perfect….

Step 4: Make a decision based on the evaluation.
Based on the above criteria and evaluation, I would say that tennis is the sport that is easiest to learn but hardest to master. While it may require expensive equipment, it is still accessible to many people. The basic rules and techniques are easy to learn, but mastering all the different shots and strategies takes years of practice…

Exercise: Utilize prompt engineering in your app

Explore prompt engineering techniques

Connect your app to Azure OpenAI and test prompts with increasing prompt engineering

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/openai-prompt-engineering-lab

Implement Retrieval Augmented Generation (RAG) with Azure OpenAI Service

Learning Objectives

After completing this module, you will be able to:

1. Understand how RAG using your own data works
2. Use the REST API
3. Use language specific SDKs

How Azure OpenAI can use your data

Set up your data source

- Use an existing data source, such as an Azure search resource
- Use the Azure AI Studio to create that data source, if you don't already have one
- When creating the data source, you can use data already in your account such as blob storage

Configure the studio or your app to connect to that data source

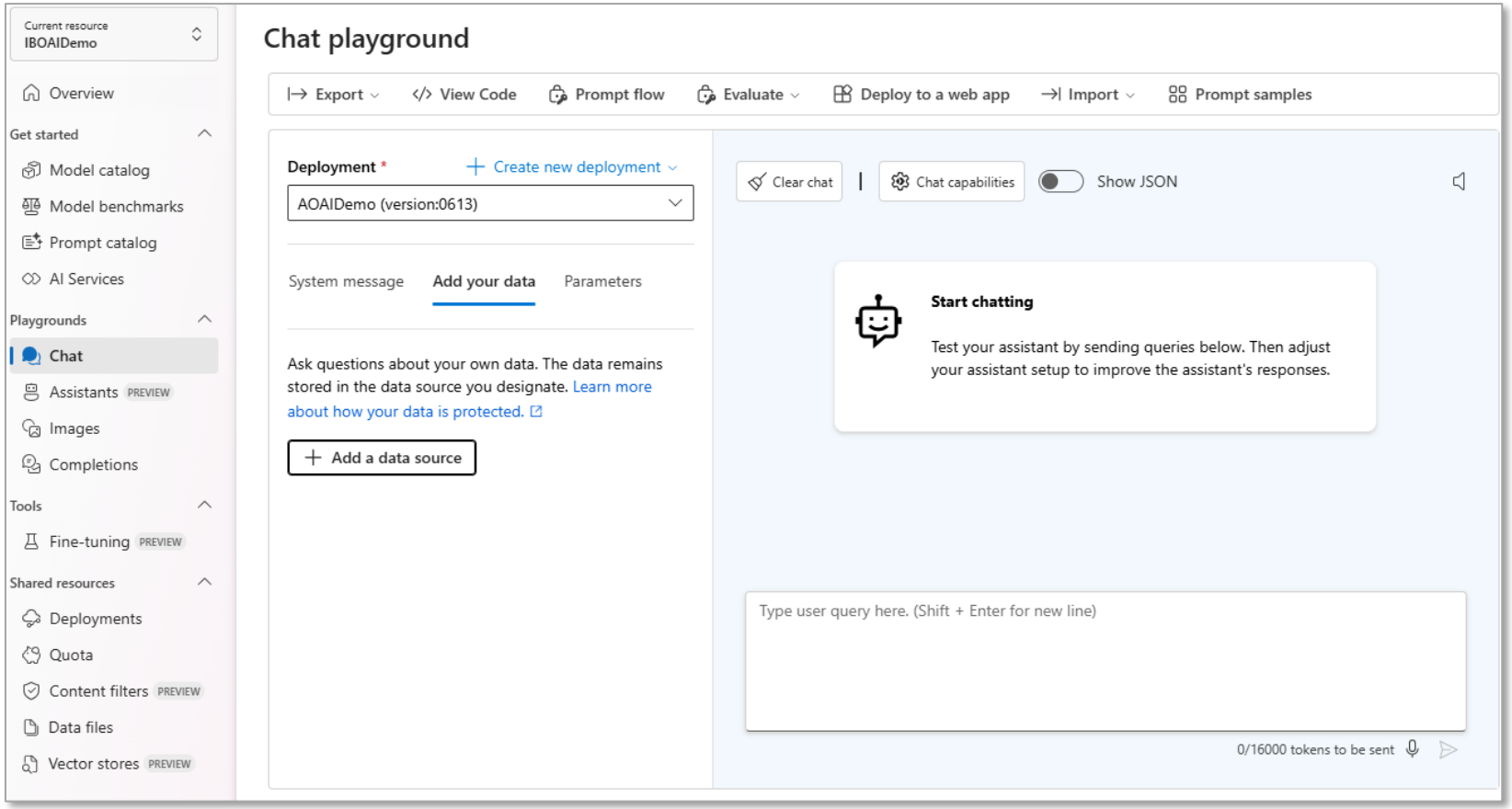
- In the studio, set up the connection by pointing it to the data source
- In your app, specify the data source in the prompt parameters
- Both configurations allow the search resource to augment the prompt

Use the Azure OpenAI model, which now uses your data for grounding

- Chat with the AI models like normal
- If the data source has relevant information about the prompt, it will use that data
- You can specify if the AI model is limited to just your data source

Connect to your data source

- Add your data source in the Chat playground, under **Add your data**
- Use an existing data source, or use that wizard to create a new one
- Once connected, a new chat session will start. Chat like normal, and see how the AI model references that data



Using the Azure OpenAI REST API

Using your own data

https://*endpoint*.openai.azure.com/openai/deployments/*deployment*/**chat/completions**?api-version=*version*

- With each call, you need to specify the data source values, along with the messages array and any other parameters
- Authentication in the data source definition is for your search resource, not your Azure OpenAI resource

```
{
    "data_sources": [
        {
            "type": "azure_search",
            "parameters": {
                "endpoint": "<your_search_endpoint>",
                “index_name": "<your_search_index>“,
                "authentication": {
                    "type": "system_assigned_managed_identity"
                }
            }
        }
    ],
    "messages":[
                ...
    ]
}
```

Using the Azure OpenAI SDKs

Language specific SDKs are available for use in your applications, in both C# and Python.

Code structure follows a similar pattern for both languages.

Current supported data sources are:

- Azure AI Search
- Azure Cosmos DB for MangoDB vCore
- Plus others in preview, soon to be released GA

Pseudo code structure:

```
<include library>

<create client>

<define chat messages and options>

<define data source object to include with
request>

<send request>

<extract response content>
```

Exercise: Implement Retrieval Augmented Generation (RAG) with Azure OpenAI Service

Set up and connect your data in the chat playground

Configure your app to use your own data for augmenting the prompt

Use the hosted lab environment if provided, or view the lab instructions at the link below:

https://aka.ms/mslearn-openai-own-data

Extended interactive exercises

Generate code

Generate images

https://aka.ms/develop-azure-openai

Knowledge check

1 **What is the purpose of providing conversation history to an AI model?**

- ☐ Providing conversation history to an AI model is irrelevant and has no effect on the AI's performance.
- ☐ To limit the number of input tokens used by the model
- ☑ To enable the model to continue responding in a similar way and allow the user to reference previous content in subsequent queries

2 **Which parameter could you adjust to change the randomness or creativeness of completions?**

- ☑ Temperature
- ☐ Frequency penalty
- ☐ Stop sequence

3 **You plan to implement a multi-turn conversation with Azure OpenAI. Which endpoint should you use?**

- ☐ Completions
- ☑ Chat
- ☐ Embeddings

Learning Recap

In this learning path, we:

Created and deployed Azure OpenAI resources

Integrated Azure OpenAI into your application through REST APIs and SDKs

Explored prompt engineering techniques to improve model responses

Connected your own data for grounding an Azure OpenAI model