6.1 - Get started with Azure OpenAI Service

Overview

This module provides engineers with the skills to begin building an Azure OpenAI Service solution.

By the end of this module, you'll be able to:

- Create an Azure OpenAI Service resource and understand types of Azure Ope

---

Introduction

Suppose you want to build a support application that summarizes text and suggest code. To build this app, you want to utilize the capabilities you see in ChatGPT, a chatbot built by the OpenAI research company that takes in natural language input from a user and returns a machine-created, human-like response.

Generative AI models power ChatGPT's ability to produce new content, such as text, code, and images, based on a natural language prompts. Many generative AI models are a subset of deep learning algorithms. These algorithms support various workloads across vision, speech, language, decision, search, and more.

Azure OpenAI Service brings these generative AI models to the Azure platform, enabling you to develop powerful AI solutions that benefit from the security, scalability, and integration of other services provided by the Azure cloud platform. These models are available for building applications through a REST API, various SDKs, and a Foundry interface. This module guides you through the Azure AI Foundry experience, giving you the foundation to further develop solutions with generative AI.

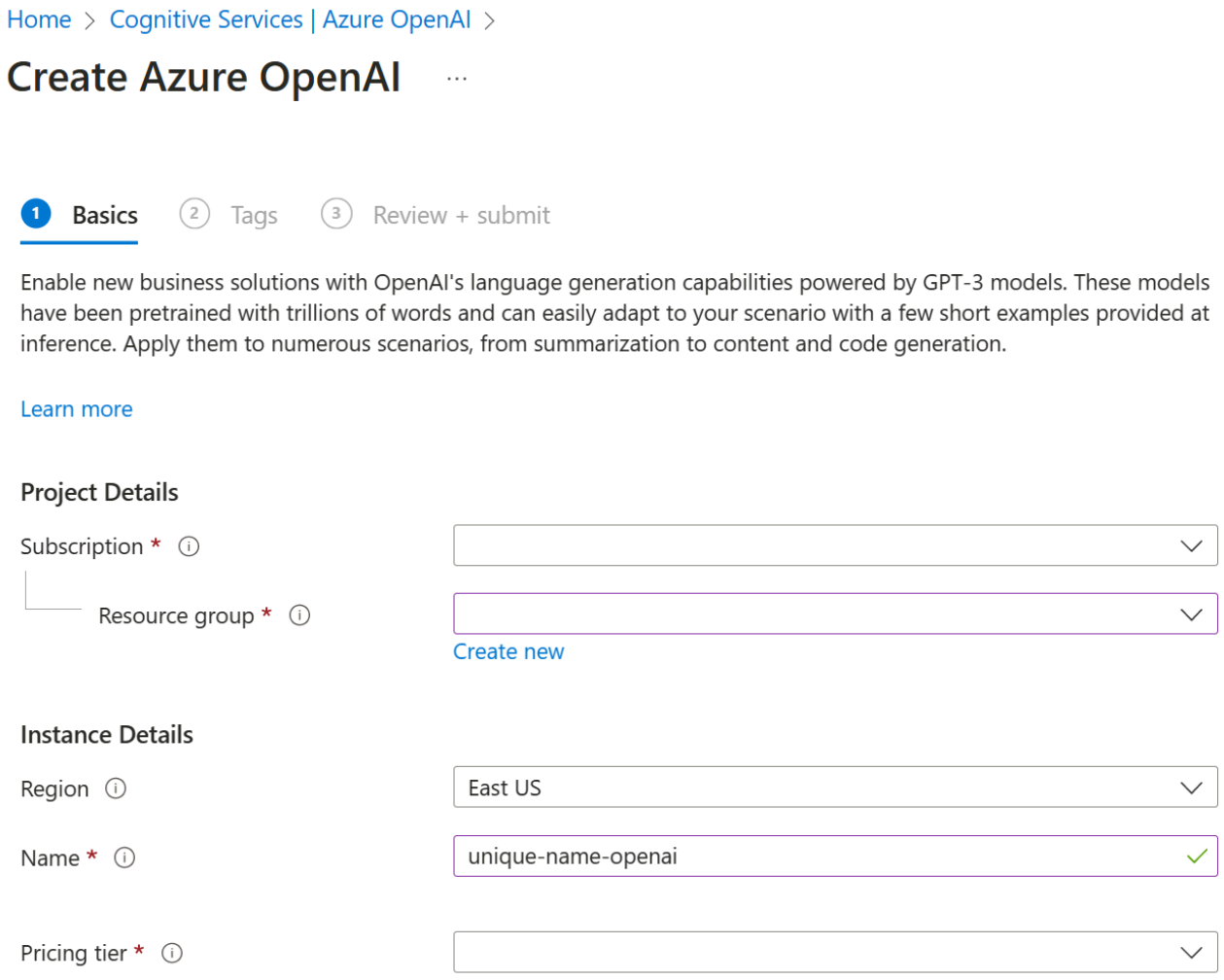
Access Azure OpenAI Service

The first step in building a generative AI solution with Azure OpenAI is to provision an Azure OpenAI resource in your Azure subscription. Azure OpenAI Service is now available to all Azure accounts, with some advanced features (like custom content filters) restricted behind a limited access policy.

Once you have access to Azure OpenAI Service, you can get started by creating a resource in the Azure portal or with the Azure command line interface (CLI).

Create an Azure OpenAI Service resource in the Azure portal

When you create an Azure OpenAI Service resource, you need to provide a subscription name, resource group name, region, unique instance name, and select a pricing tier.



Create an Azure OpenAI Service resource in Azure CLI

To create an Azure OpenAI Service resource from the CLI, refer to this example and replace the following variables with your own:

- MyOpenAIResource: *replace with a unique name for your resource*
- OAIResourceGroup: *replace with your resource group name*
- eastus: *replace with the region to deploy your resource*
- subscriptionID: *replace with your subscription ID*

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

> Note: You can find the regions available for a service through the CLI command `az account list-locations`. To see how to sign into Azure and create an Azure group via the CLI, you can refer to the documentation here.

Regional availability

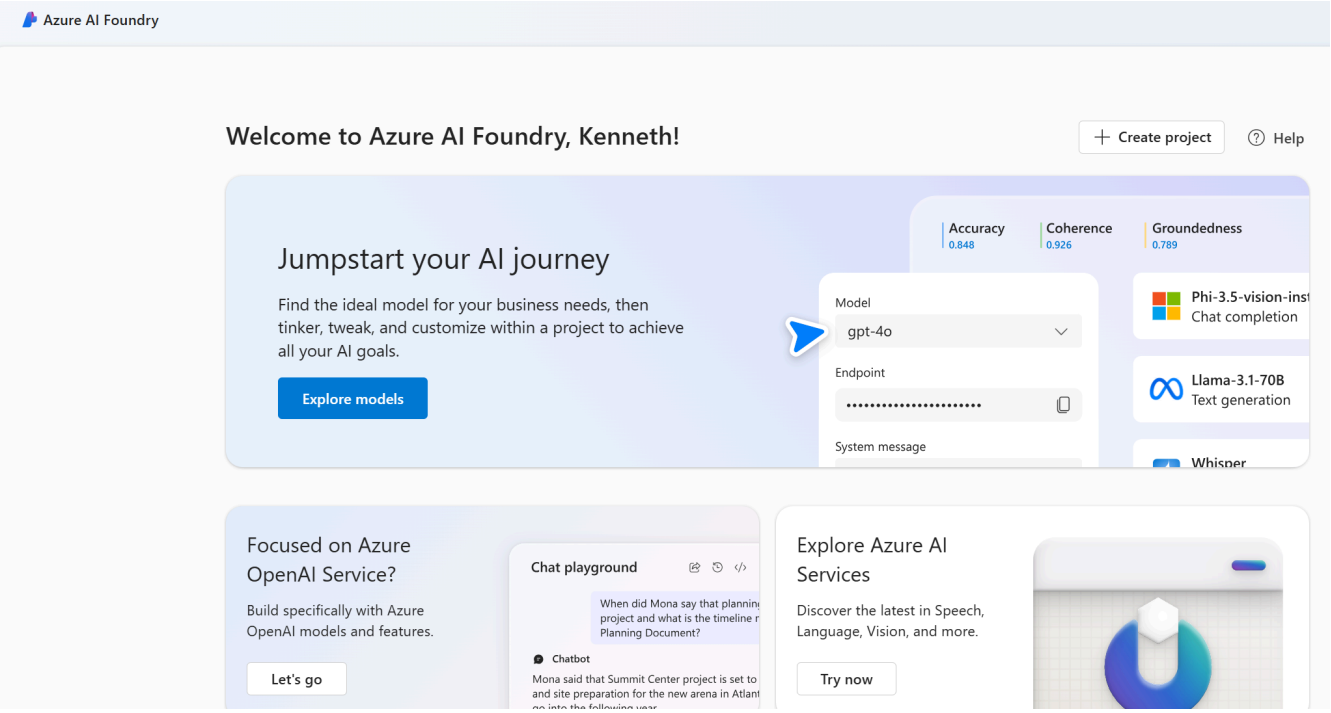
Azure OpenAI Service provides access to many types of models. Certain models are only available in select regions. Consult the Azure OpenAI model availability guide for region availability.

You can create two Azure OpenAI resources per region.

Use Azure AI Foundry

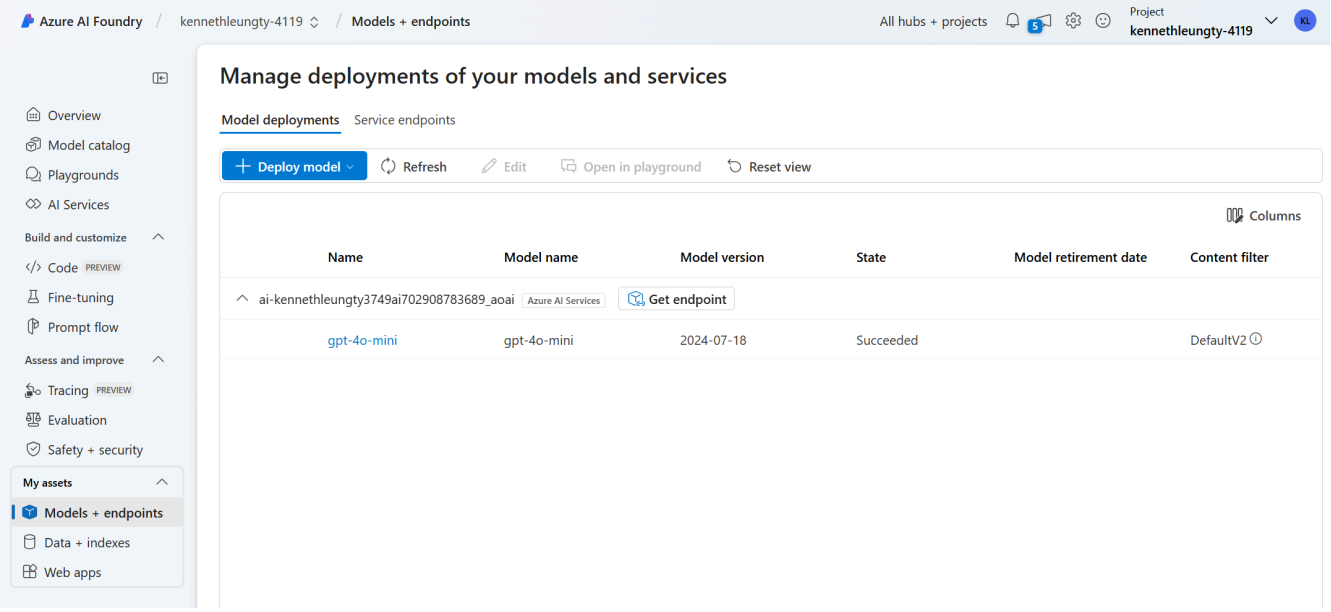
Azure AI Studio (now named as Azure AI Foundry) provides access to model management, deployment, experimentation, customization, and learning resources.

You can access the Azure AI Foundry through the Azure portal after creating a resource, or at https://ai.azure.com/ by signing in to your Azure account. During the sign-in workflow, select the appropriate directory, Azure subscription, and Azure OpenAI resource.



When you first open Azure AI Foundry, you'll want to navigate to the **Azure OpenAI** page, select your resource if you haven't already, and deploy your first model.

To do so, select the **Deployments** page, from where you can deploy a base model and start experimenting with it.



> Note: If you are not the resource owner, you will need the following role-based access controls:
> 1. *Cognitive Services OpenAI User*: This role allows viewing resources and using the chat playground.
> 2. *Cognitive Services OpenAI Contributor*: This role allows the user to create new deployments.

Explore types of generative AI models

To begin building with Azure OpenAI, you need to choose a base model and deploy it. Microsoft provides base models and the option to create customized base models. This module covers the currently available base models.

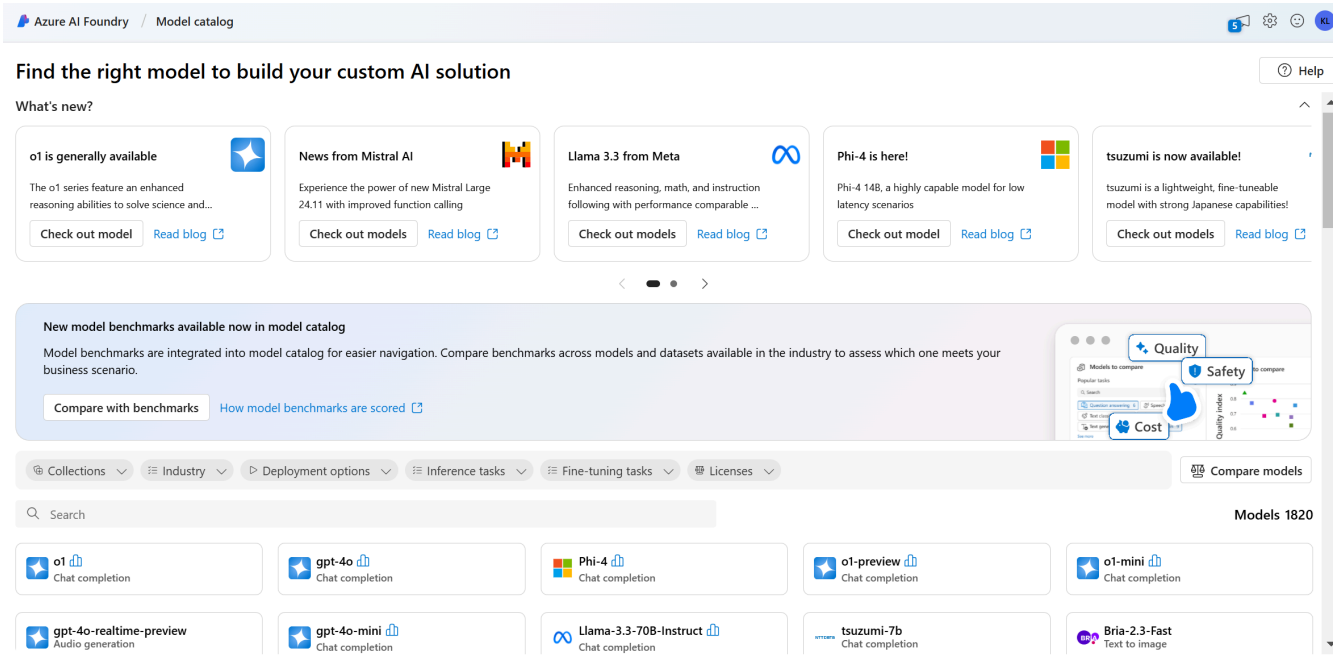
Azure OpenAI includes several types of model:

- **GPT-4 models** are the latest generation of *generative pretrained* (GPT) models that can generate natural language and code completions based on natural language prompts.
- **GPT 3.5 models** can generate natural language and code completions based on natural language prompts. In particular, **GPT-35-turbo** models are optimized for chat-based interactions and work well in most generative AI scenarios.
- **Embeddings models** convert text into numeric vectors, and are useful in language analytics scenarios such as comparing text sources for similarities.
- **DALL-E models** are used to generate images based on natural language prompts. Currently, DALL-E models are in preview. DALL-E models aren't listed in the Azure AI Foundry interface and don't need to be explicitly deployed.

Models differ by speed, cost, and how well they complete specific tasks. You can learn more about the differences and latest models offered in the Azure OpenAI Service documentation.

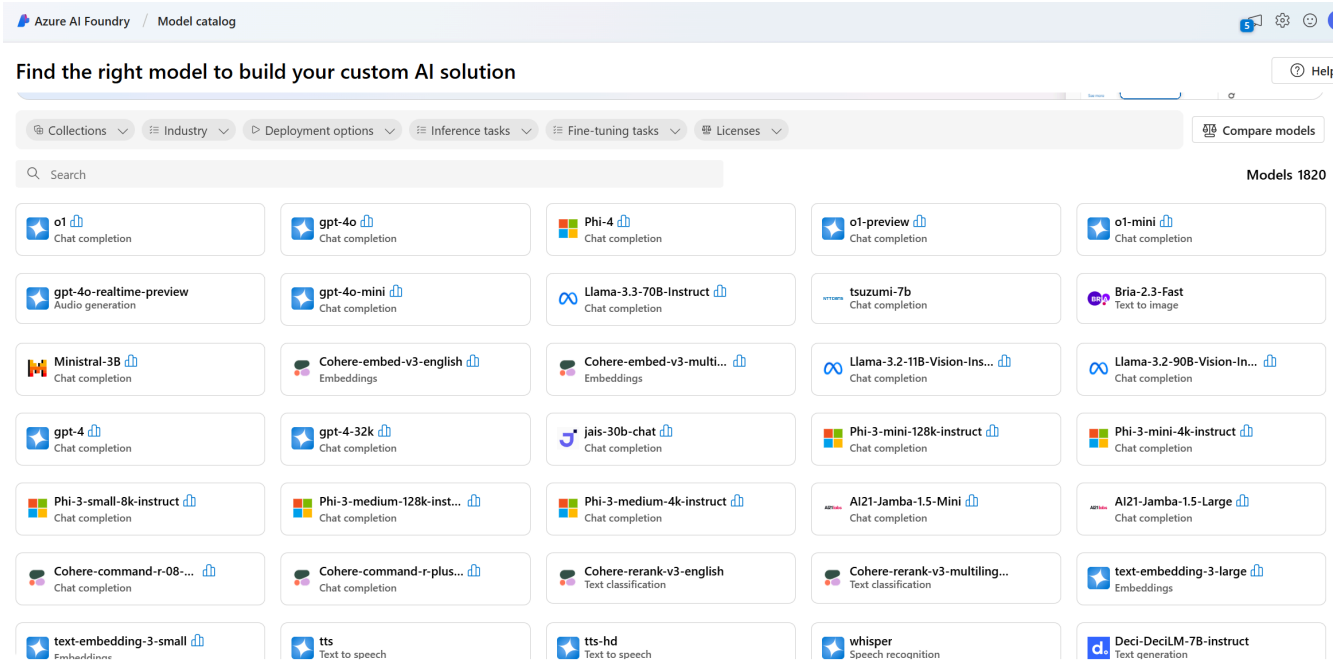
> Note: Pricing is determined by tokens and by model type. Learn more about the latest pricing here.

In the Azure AI Foundry, the **Model Catalog** page lists the available base models and provides an option to create additional customized models by fine-tuning the base models.



The models that have a *Succeeded* status mean they're successfully trained and can be selected for deployment.

You'll notice that there are various models beyond OpenAI available in the Model Catalog, including models from Microsoft, Meta, Mistral, and more. Azure AI Foundry enables you to deploy any of these models for your use case. This module will focus on Azure OpenAI models.

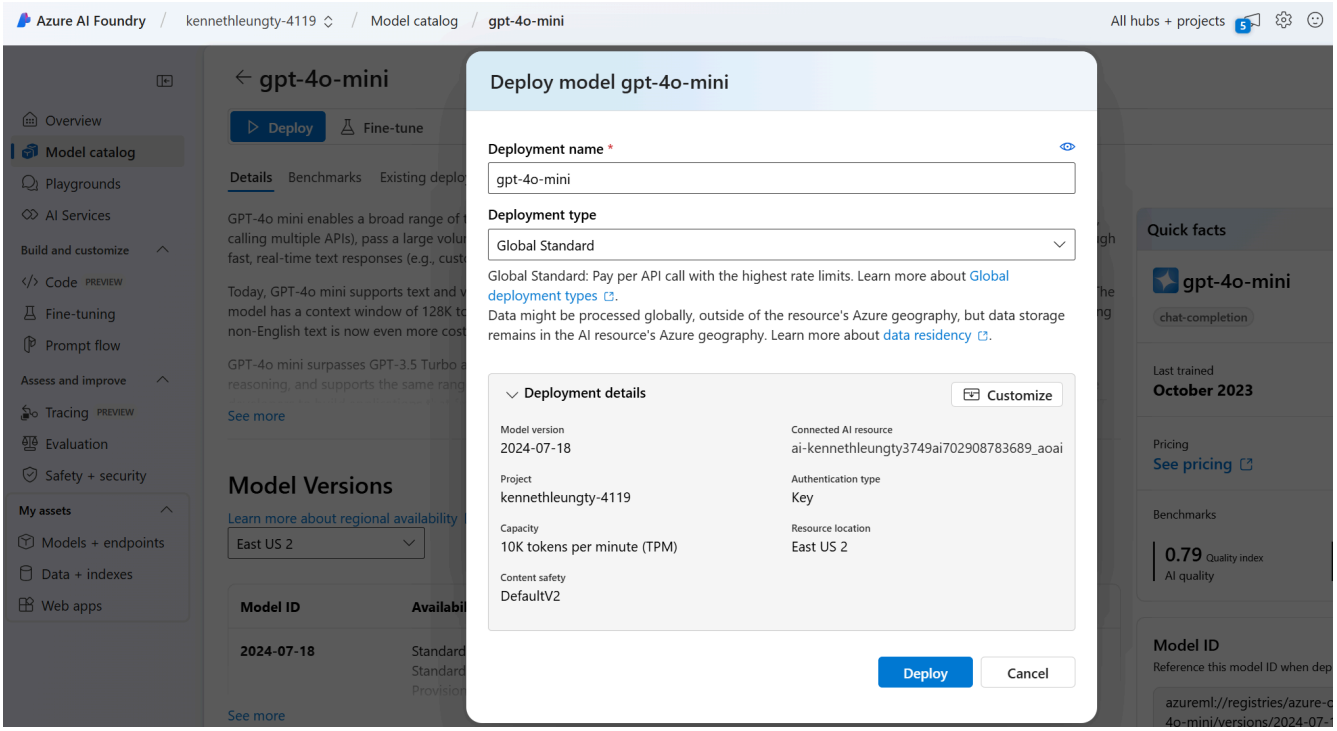


Deploy generative AI models

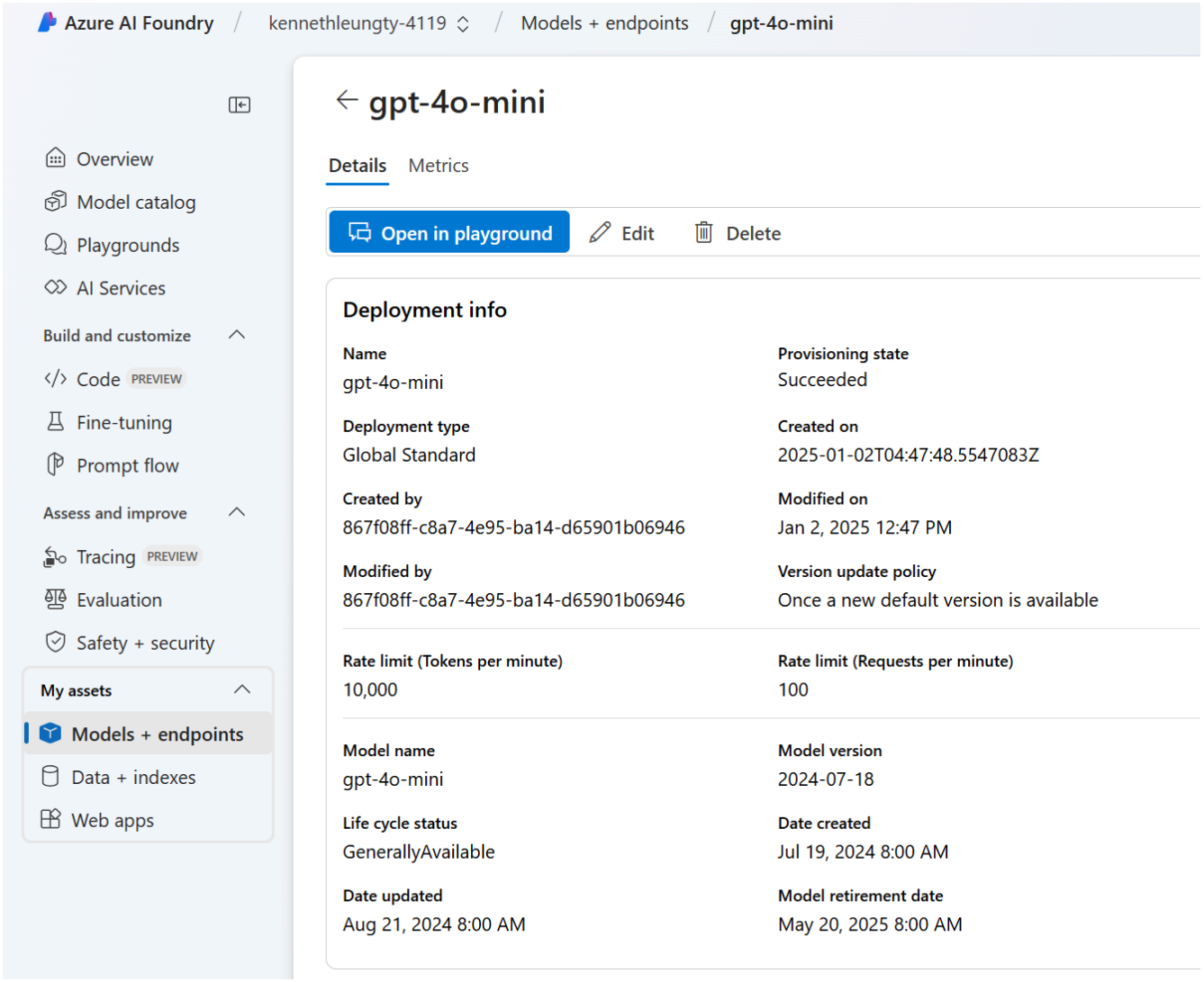
You first need to deploy a model to chat with or make API calls to receive responses to prompts. When you create a new deployment, you need to indicate which base model to deploy. You can deploy any number of deployments in one or multiple Azure OpenAI resources as long as their Tokens Per Minute (TPM) stays within the deployment quota.

Deploy using Azure AI Foundry

In Azure AI Foundry's **Deployments** page, you can create a new deployment by selecting a model name from the menu. The available base models come from the list in the models page.



From the *Deployments* page in the Foundry, you can also view information about all your deployments including deployment name, model name, model version, status, date created, and more.



Deploy using Azure CLI

You can also deploy a model using the console. Using this example, replace the following variables with your own resource values:

- OAIResourceGroup: *replace with your resource group name*
- MyOpenAIResource: *replace with your resource name*
- MyModel: *replace with a unique name for your model*
- gpt-35-turbo: *replace with the base model you wish to deploy*

```
az cognitiveservices account deployment create \
   -g OAIResourceGroup \
   -n MyOpenAIResource \
   --deployment-name MyModel \
   --model-name gpt-35-turbo \
   --model-version "0301"  \
   --model-format OpenAI \
   --sku-name "Standard" \
   --sku-capacity 1
```

Deploy using the REST API

You can deploy a model using the REST API. In the request body, you specify the base model you wish to deploy. See an example in the Azure OpenAI documentation.

Use prompts to get completions from models

Once the model is deployed, you can test how it completes prompts. A prompt is the text portion of a request that is sent to the deployed model's completions endpoint. Responses are referred to as *completions*, which can come in form of text, code, or other formats.

Prompt types

Prompts can be grouped into types of requests based on task.

Task type	Prompt example	Completion example
Classifying content	Tweet: I enjoyed the trip. Sentiment:	Positive
Generating new content	List ways of traveling	1. Bike 2. Car ...
Holding a conversation	A friendly AI assistant	See examples
Transformation (translation and symbol conversion)	English: Hello French:	bonjour
Summarizing content	Provide a summary of the content {text}	The content shares methods of machine learning.
Picking up where you left off	One way to grow tomatoes	is to plant seeds.
Giving factual responses	How many moons does Earth have?	One

Completion quality

Several factors affect the quality of completions you'll get from a generative AI solution.

- The way a prompt is engineered. Learn more about prompt engineering here.
- The model parameters (covered next)
- The data the model is trained on, which can be adapted through model fine-tuning with customization

You have more control over the completions returned by training a custom model than through prompt engineering and parameter adjustment.

Making calls

You can start making calls to your deployed model via the REST API, Python, C#, or from the Studio. If your deployed model has a GPT-3.5 or GPT-4 model base, use the Chat completions documentation, which has different request endpoints and variables required than for other base models.

Test models in Azure AI Studio's playground

Playgrounds are useful interfaces in Azure AI Studio that you can use to experiment with your deployed models without needing to develop your own client application. Azure AI Studio offers multiple playgrounds with different parameter tuning options.

Completions playground

The Completions playground allows you to make calls to your deployed models through a text-in, text-out interface and to adjust parameters.

You need to select the deployment name of your model under Deployments. Optionally, you can use the provided examples to get you started, and then you can enter your own prompts.

Completions Playground parameters

There are many parameters that you can adjust to change the performance of your model:

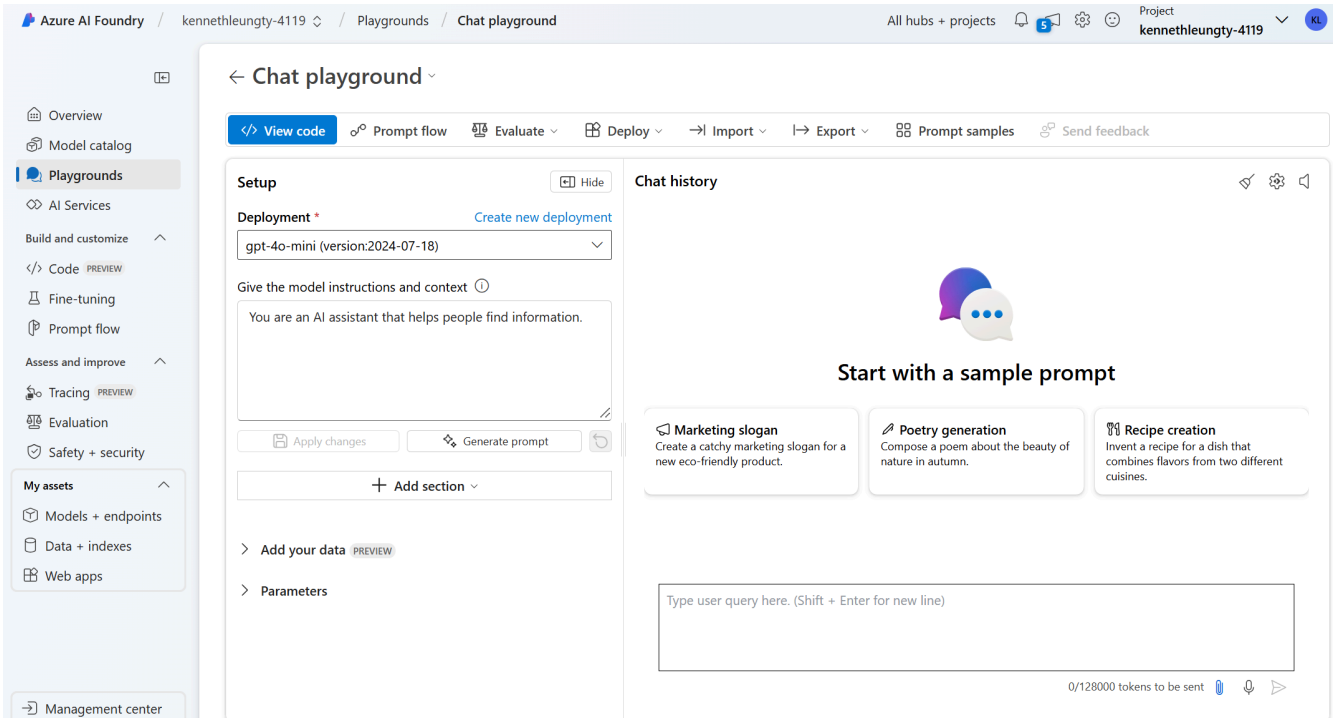
- **Temperature**: Controls randomness. Lowering the temperature means that the model produces more repetitive and deterministic responses. Increasing the temperature results in more unexpected or creative responses. Try adjusting temperature or Top P but not both.
- **Max length (tokens)**: Set a limit on the number of tokens per model response. The API supports a maximum of 4000 tokens shared between the prompt (including system message, examples, message history, and user query) and the model's response. One token is roughly four characters for typical English text.
- **Stop sequences**: Make responses stop at a desired point, such as the end of a sentence or list. Specify up to four sequences where the model will stop generating further tokens in a response. The returned text won't contain the stop sequence.
- **Top probabilities (Top P)**: Similar to temperature, this controls randomness but uses a different method. Lowering Top P narrows the model’s token selection to likelier tokens. Increasing Top P lets the model choose from tokens with both high and low likelihood. **Try adjusting temperature or Top P but not both.**
- **Frequency penalty**: Reduce the chance of repeating a token proportionally based on how often it has appeared in the text so far. This decreases the likelihood of repeating the exact same text in a response.
- **Presence penalty**: Reduce the chance of repeating any token that has appeared in the text at all so far. This increases the likelihood of introducing new topics in a response.
- **Pre-response text**: Insert text after the user’s input and before the model’s response. This can help prepare the model for a response.
- **Post-response text**: Insert text after the model’s generated response to encourage further user input, as when modeling a conversation.

Chat playground

The Chat playground is based on a conversation-in, message-out interface. You can initialize the session with a system message to set up the chat context.

In the Chat playground, you're able to add *few-shot examples*. The term few-shot refers to providing a few of examples to help the model learn what it needs to do. You can think of it in contrast to zero-shot, which refers to providing no examples.

In the *Assistant setup*, you can provide few-shot examples of what the user input may be, and what the assistant response should be. The assistant tries to mimic the responses you include here in tone, rules, and format you've defined in your system message.



Chat playground parameters

The Chat playground, like the Completions playground, also includes parameters to customize the model's behavior. The Chat playground also supports other parameters *not* available in the Completions playground.

These include:

- **Max response**: Set a limit on the number of tokens per model response. The API supports a maximum of 4000 tokens shared between the prompt (including system message, examples, message history, and user query) and the model's response. **One token is roughly four characters for typical English text.**
- **Past messages included**: Select the number of past messages to include in each new API request. Including past messages helps give the model context for new user queries. Setting this number to 10 will include five user queries and five system responses.

The **Current token count** is viewable from the Chat playground. Since the API calls are priced by token and it's possible to set a max response token limit, you'll want to keep an eye out for the current token count to make sure the conversation-in doesn't exceed the max response token count.

Exercise - Get started with Azure OpenAI Service

Azure OpenAI Service brings the generative AI models developed by OpenAI to the Azure platform, enabling you to develop powerful AI solutions that benefit from the security, scalability, and integration of services provided by the Azure cloud platform. In this exercise, you'll learn how to get started with Azure OpenAI by provisioning the service as an Azure resource and using Azure AI Foundry to deploy and explore generative AI models.

In the scenario for this exercise, you will perform the role of a software developer who has been tasked to implement an AI agent that can use generative AI to help a marketing organization improve its effectiveness at reaching customers and advertising new products. The techniques used in the exercise can be applied to any scenario where an organization wants to use generative AI models to help employees be more effective and productive.

This exercise takes approximately **30** minutes.

Provision an Azure OpenAI resource

If you don't already have one, provision an Azure OpenAI resource in your Azure subscription.

1. Sign into the **Azure portal** at `https://portal.azure.com`.
2. Create an **Azure OpenAI** resource with the following settings:
   - **Subscription**: *Select an Azure subscription that has been approved for access to the Azure OpenAI service*
   - **Resource group**: *Choose or create a resource group*
   - **Region**: *Make a **random** choice from any of the following regions**
     - Australia East
     - Canada East
     - East US
     - East US 2
     - France Central
     - Japan East
     - North Central US
     - Sweden Central
     - Switzerland North
     - UK South
   - **Name**: *A unique name of your choice*
   - **Pricing tier**: Standard S0

   > - Azure OpenAI resources are constrained by regional quotas. The listed regions include default quota for the model type(s) used in this exercise. Randomly choosing a region reduces the risk of a single region reaching its quota limit in scenarios where you are sharing a subscription with other users. In the event of a quota limit being reached later in the exercise, there's a possibility you may need to create another resource in a different region.

3. Wait for deployment to complete. Then go to the deployed Azure OpenAI resource in the Azure portal.

Deploy a model

Azure provides a web-based portal named **Azure AI Foundry portal**, that you can use to deploy, manage, and explore models. You'll start your exploration of Azure OpenAI by using Azure AI Foundry portal to deploy a model.

> **Note**: As you use Azure AI Foundry portal, message boxes suggesting tasks for you to perform may be displayed. You can close these and follow the steps in this exercise.

1. In the Azure portal, on the **Overview** page for your Azure OpenAI resource, scroll down to the **Get Started** section and select the button to go to **AI Foundry portal** (previously AI Studio).
2. In Azure AI Foundry portal, in the pane on the left, select the **Deployments** page and view your existing model deployments. If you don't already have one, create a new deployment of the **gpt-35-turbo-16k** model with the following settings:
   - **Deployment name**: *A unique name of your choice*
   - **Model**: gpt-35-turbo-16k *(if the 16k model isn't available, choose gpt-35-turbo)*
   - **Model version**: *Use default version*
   - **Deployment type**: Standard
   - **Tokens per minute rate limit**: 5K*
   - **Content filter**: Default
   - **Enable dynamic quota**: Disabled

> - A rate limit of 5,000 tokens per minute is more than adequate to complete this exercise while leaving capacity for other people using the same subscription.

Use the Chat playground

Now that you've deployed a model, you can use it to generate responses based on natural language prompts. The *Chat* playground in Azure AI Foundry portal provides a chatbot interface for GPT 3.5 and higher models.

> **Note:** The *Chat* playground uses the *ChatCompletions* API rather than the older *Completions* API that is used by the *Completions* playground. The Completions playground is provided for compatibility with older models.

1. In the **Playground** section, select the **Chat** page. The **Chat** playground page consists of a row of buttons and two main panels (which may be arranged right-to-left horizontally, or top-to-bottom vertically depending on your screen resolution):
   - **Configuration** - used to select your deployment, define system message, and set parameters for interacting with your deployment.
   - **Chat session** - used to submit chat messages and view responses.
2. Under **Deployments**, ensure that your gpt-35-turbo-16k model deployment is selected.
3. Review the default **System message**, which should be *You are an AI assistant that helps people find information.* The system message is included in prompts submitted to the model, and provides context for the model's responses; setting expectations about how an AI agent based on the model should interact with the user.
4. In the **Chat session** panel, enter the user query `How can I use generative AI to help me market a new product?`

> **Note**: You may receive a response that the API deployment is not yet ready. If so, wait for a few minutes and try again.

5. Review the response, noting that the model has generated a cohesive natural language answer that is relevant to the query with which it was prompted.
6. Enter the user query `What skills do I need if I want to develop a solution to accomplish this?`.
7. Review the response, noting that the chat session has retained the conversational context (so "this" is interpreted as a generative AI solution for marketing). This contextualization is achieved by including

the recent conversation history in each successive prompt submission, so the prompt sent to the model for the second query included the original query and response as well as the new user input.

8. In the **Chat session** panel toolbar, select **Clear chat** and confirm that you want to restart the chat session.
9. Enter the query `Can you help me find resources to learn those skills?` and review the response, which should be a valid natural language answer, but since the previous chat history has been lost, the answer is likely to be about finding generic skilling resources rather than being related to the specific skills needed to build a generative AI marketing solution.

Experiment with system messages, prompts, and few-shot examples

So far, you've engaged in a chat conversation with your model based on the default system message. You can customize the system setup to have more control over the kinds of responses generated by your model.

1. In the main toolbar, select the **Prompt samples**, and use the **Marketing Writing Assistant** prompt template.
2. Review the new system message, which describes how an AI agent should use the model to respond.
3. In the **Chat session** panel, enter the user query `Create an advertisement for a new scrubbing brush`.
4. Review the response, which should include advertising copy for a scrubbing brush. The copy may be quite extensive and creative. In a real scenario, a marketing professional would likely already know the name of the scrubbing brush product as well as have some ideas about key features that should be highlighted in an advert. To get the most useful results from a generative AI model, users need to design their prompts to include as much pertinent information as possible.
5. Enter the prompt `Revise the advertisement for a scrubbing brush named "Scrubadub 2000", which is made of carbon fiber and reduces cleaning times by half compared to ordinary scrubbing brushes`.
6. Review the response, which should take into account the additional information you provided about the scrubbing brush product. The response should now be more useful, but to have even more control over the output from the model, you can provide one or more *few-shot* examples on which responses should be based.
7. Under the **System message** text box, expand the dropdown for **Add section** and select **Examples**. Then type the following message and response in the designated boxes:

   User:
   `Write an advertisement for the lightweight "Ultramop" mop, which uses patented absorbent materials to clean floors.`

   Assistant:
   `Welcome to the future of cleaning! The Ultramop makes light work of even the dirtiest of floors. Thanks to its patented absorbent materials, it ensures a brilliant shine. Just look at these features: - Lightweight construction, making it easy to use. - High absorbency, enabling you to apply lots of clean soapy water to the floor. - Great low price. Check out this and other products on our website at www.contoso.com.`
8. Use the **Apply changes** button to save the examples and start a new session.
9. In the **Chat session** section, enter the user query `Create an advertisement for the Scrubadub 2000 - a new scrubbing brush made of carbon fiber that reduces cleaning time by half`.

10. Review the response, which should be a new advert for the "Scrubadub 2000" that is modeled on the "Ultramop" example provided in the system setup.

Experiment with parameters

You've explored how the system message, examples, and prompts can help refine the responses returned by the model. You can also use parameters to control model behavior.

1. In the **Configuration** panel, select the **Parameters** tab and set the following parameter values:
   - **Max response**: 1000
   - **Temperature**: 1 (ranges from 0 to 1)
2. In the **Chat session** section, use the **Clear chat** button to reset the chat session. Then enter the user query `Create an advertisement for a cleaning sponge` and review the response. The resulting advertisement copy should include a maximum of 1000 text tokens, and include some creative elements - for example, the model may have invented a product name for the sponge and made some claims about its features.
3. Use the **Clear chat** button to reset the chat session again, and then re-enter the same query as before (`Create an advertisement for a cleaning sponge`) and review the response. The response may be different from the previous response.
4. In the **Configuration** panel, on the **Parameters** tab, change the **Temperature** parameter value to 0.
5. In the **Chat session** section, use the **Clear chat** button to reset the chat session again, and then re-enter the same query as before (`Create an advertisement for a cleaning sponge`) and review the response. This time, the response may not be quite so creative.
6. Use the **Clear chat** button to reset the chat session one more time, and then re-enter the same query as before (`Create an advertisement for a cleaning sponge`) and review the response; which should be very similar (if not identical) to the previous response. The **Temperature** parameter controls the degree to which the model can be creative in its generation of a response. A low value results in a consistent response with little random variation, while a high value encourages the model to add creative elements its output; which may affect the accuracy and realism of the response.

Deploy your model to a web app

Now that you've explored some of the capabilities of a generative AI model in the Azure AI Foundry playground, you can deploy an Azure web app to provide a basic AI agent interface through which users can chat with the model.

> **Note**: For some users, deploying to the web app cannot be deployed due to a bug in the template in the studio. If that's the case, skip this section.

1. At the top right of the **Chat** playground page, in the **Deploy to** menu, select **A new web app**.
2. In the **Deploy to a web app** dialog box, create a new web app with the following settings:
   - **Name**: *A unique name*
   - **Subscription**: *Your Azure subscription*
   - **Resource group**: *The resource group in which you provisioned your Azure OpenAI resource*
   - **Locations**: *The region where you provisioned your Azure OpenAI resource*
   - **Pricing plan**: Free (F1) - *If this is not available, select Basic (B1)*
   - **Enable chat history in the web app**: Unselected
   - **I acknowledge that web apps will incur usage to my account**: Selected
3. Deploy the new web app and wait for deployment to complete (which may take 10 minutes or so)

4. After your web app has deployed successfully, use the button at the top right of the **Chat** playground page to launch the web app. The app may take a few minutes to launch. If prompted, accept the permissions request.
5. In the web app, enter the following chat message:
   `Write an advertisement for the new "WonderWipe" cloth that attracts dust particulates and can be used to clean any household surface.`
6. Review the response.

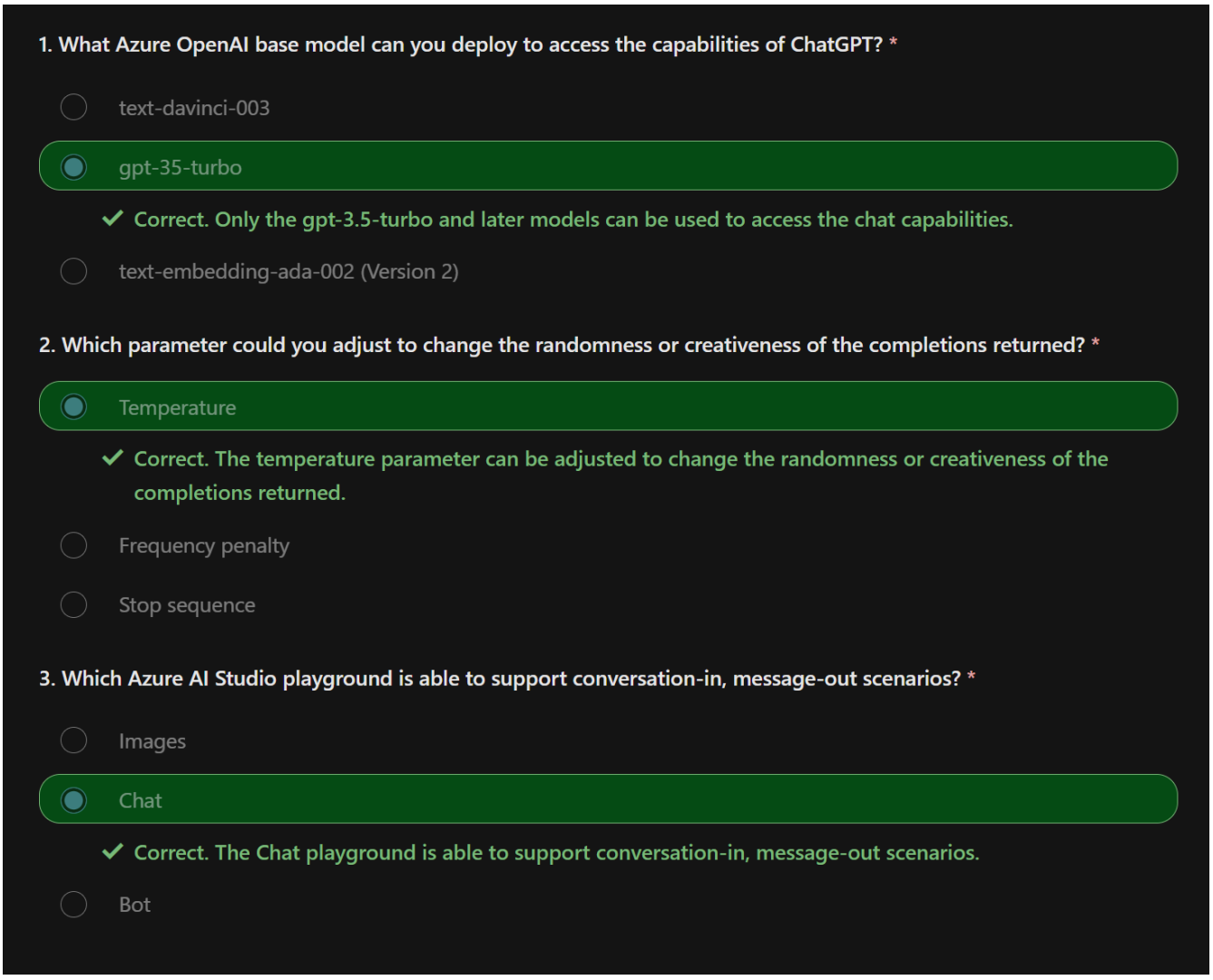
   > **Note**: You deployed the *model* to a web app, but this deployment doesn't include the system settings and parameters you set in the playground; so the response may not reflect the examples you specified in the playground. In a real scenario, you would add logic to your application to modify the prompt so that it includes the appropriate contextual data for the kinds of response you want to generate. This kind of customization is beyond the scope of this introductory-level exercise, but you can learn about prompt engineering techniques and Azure OpenAI APIs in other exercises and product documentation.

7. When you have finished experimenting with your model in the web app, close the web app tab in your browser to return to Azure AI Foundry portal.

Clean up

When you're done with your Azure OpenAI resource, remember to delete the deployment or the entire resource in the **Azure portal** at `https://portal.azure.com`.

Knowledge Check



Summary

This module covered the basics of getting started with Azure OpenAI Service with a focus on using the Azure AI Studio.

You learned how to:

- Create an Azure OpenAI Service resource and understand types of Azure OpenAI base models.
- Use the Azure AI Studio, Azure CLI, and REST API to deploy a base model.
- Generate completions to prompts.
- Test models in the Studio playground and begin to manage model parameters.

One way to learn more is by exploring the Azure OpenAI Service documentation.

✍️ Compiled by Kenneth Leung (2025)