Quickstart: Get started using chat completions with Azure OpenAI Service

Article • 03/23/2025

Use this article to get started using Azure OpenAI.

Source code | Package (NuGet) | Samples | Retrieval Augmented Generation (RAG) enterprise chat template |

Prerequisites

- An Azure subscription - Create one for free
- The .NET 7 SDK
- An Azure OpenAI Service resource with the `gpt-4o` model deployed. For more information about model deployment, see the resource deployment guide.

Microsoft Entra ID prerequisites

For the recommended keyless authentication with Microsoft Entra ID, you need to:

- Install the Azure CLI used for keyless authentication with Microsoft Entra ID.
- Assign the `Cognitive Services User` role to your user account. You can assign roles in the Azure portal under **Access control (IAM)** > **Add role assignment**.

Set up

1. Create a new folder `chat-quickstart` and go to the quickstart folder with the following command:

   shell

   ```
   mkdir chat-quickstart && cd chat-quickstart
   ```

2. Create a new console application with the following command:

   shell

```
dotnet new console
```

3. Install the OpenAI .NET client library with the dotnet add package command:

```
Console
dotnet add package Azure.AI.OpenAI --prerelease
```

4. For the **recommended** keyless authentication with Microsoft Entra ID, install the Azure.Identity package with:

```
Console
dotnet add package Azure.Identity
```

5. For the **recommended** keyless authentication with Microsoft Entra ID, sign in to Azure with the following command:

```
Console
az login
```

Retrieve resource information

You need to retrieve the following information to authenticate your application with your Azure OpenAI resource:

Microsoft Entra ID

Expand table

Variable name	Value
`AZURE_OPENAI_ENDPOINT`	This value can be found in the **Keys and Endpoint** section when examining your resource from the Azure portal.
`AZURE_OPENAI_DEPLOYMENT_NAME`	This value will correspond to the custom name you chose for your deployment when you deployed a model. This value can be found

Variable name	Value
	under **Resource Management** > **Model Deployments** in the Azure portal.
`OPENAI_API_VERSION`	Learn more about API Versions. You can change the version in code or use an environment variable.

Learn more about keyless authentication and setting environment variables.

Run the quickstart

The sample code in this quickstart uses Microsoft Entra ID for the recommended keyless authentication. If you prefer to use an API key, you can replace the `DefaultAzureCredential` object with an `AzureKeyCredential` object.

Microsoft Entra ID

C#

```
AzureOpenAIClient openAIClient = new AzureOpenAIClient(new Uri(endpoint), new
DefaultAzureCredential());
```

You can use streaming or non-streaming to get the chat completion. The following code examples show how to use both methods. The first example shows how to use the non-streaming method, and the second example shows how to use the streaming method.

Without response streaming

To run the quickstart, follow these steps:

1. Replace the contents of `Program.cs` with the following code and update the placeholder values with your own.

   C#

   ```
   using Azure;
   using Azure.Identity;
   using OpenAI.Assistants;
   ```

```
using Azure.AI.OpenAI;
using OpenAI.Chat;
using static System.Environment;

string endpoint = Environment.GetEnvironmentVariable("AZURE_OPENAI_ENDPOINT") ??
"https://<your-resource-name>.openai.azure.com/";
string key = Environment.GetEnvironmentVariable("AZURE_OPENAI_API_KEY") ?? "
<your-key>";

// Use the recommended keyless credential instead of the AzureKeyCredential cre-
dential.
AzureOpenAIClient openAIClient = new AzureOpenAIClient(new Uri(endpoint), new
DefaultAzureCredential());
//AzureOpenAIClient openAIClient = new AzureOpenAIClient(new Uri(endpoint), new
AzureKeyCredential(key));

// This must match the custom deployment name you chose for your model
ChatClient chatClient = openAIClient.GetChatClient("gpt-4o");

ChatCompletion completion = chatClient.CompleteChat(
    [
        new SystemChatMessage("You are a helpful assistant that talks like a pi-
rate."),
        new UserChatMessage("Does Azure OpenAI support customer managed keys?"),
        new AssistantChatMessage("Yes, customer managed keys are supported by
Azure OpenAI"),
        new UserChatMessage("Do other Azure AI services support this too?")
    ]);

Console.WriteLine($"{completion.Role}: {completion.Content[0].Text}");
```

2. Run the application with the following command:

shell

```
dotnet run
```

Output

Output

```
Assistant: Arrr, ye be askin’ a fine question, matey! Aye, several Azure AI services
support customer-managed keys (CMK)! This lets ye take the wheel and secure yer data
with encryption keys stored in Azure Key Vault. Services such as Azure Machine
Learning, Azure Cognitive Search, and others also offer CMK fer data protection.
```

```
Always check the specific service's documentation fer the latest updates, as features
tend to shift swifter than the tides, aye!
```

This will wait until the model has generated its entire response before printing the results. Alternatively, if you want to asynchronously stream the response and print the results, you can replace the contents of *Program.cs* with the code in the next example.

Async with streaming

To run the quickstart, follow these steps:

1. Replace the contents of `Program.cs` with the following code and update the placeholder values with your own.

```
C#
```

```
using Azure;
using Azure.Identity;
using OpenAI.Assistants;
using Azure.AI.OpenAI;
using OpenAI.Chat;
using static System.Environment;

string endpoint = Environment.GetEnvironmentVariable("AZURE_OPENAI_ENDPOINT") ??
"https://<your-resource-name>.openai.azure.com/";
string key = Environment.GetEnvironmentVariable("AZURE_OPENAI_API_KEY") ?? "
<your-key>";

// Use the recommended keyless credential instead of the AzureKeyCredential cre-
dential.
AzureOpenAIClient openAIClient = new AzureOpenAIClient(new Uri(endpoint), new
DefaultAzureCredential());
//AzureOpenAIClient openAIClient = new AzureOpenAIClient(new Uri(endpoint), new
AzureKeyCredential(key));

// This must match the custom deployment name you chose for your model
ChatClient chatClient = openAIClient.GetChatClient("gpt-4o");

var chatUpdates = chatClient.CompleteChatStreamingAsync(
    [
        new SystemChatMessage("You are a helpful assistant that talks like a pi-
rate."),
        new UserChatMessage("Does Azure OpenAI support customer managed keys?"),
        new AssistantChatMessage("Yes, customer managed keys are supported by
Azure OpenAI"),
        new UserChatMessage("Do other Azure AI services support this too?")
    ]);
```

```
await foreach(var chatUpdate in chatUpdates)
{
    if (chatUpdate.Role.HasValue)
    {
        Console.Write($"{chatUpdate.Role} : ");
    }

    foreach(var contentPart in chatUpdate.ContentUpdate)
    {
        Console.Write(contentPart.Text);
    }
}
```

2. Run the application with the following command:

shell

```
dotnet run
```

Output

Output

```
Assistant: Arrr, ye be askin’ a fine question, matey! Aye, several Azure AI services
support customer-managed keys (CMK)! This lets ye take the wheel and secure yer data
with encryption keys stored in Azure Key Vault. Services such as Azure Machine
Learning, Azure Cognitive Search, and others also offer CMK fer data protection.
Always check the specific service's documentation fer the latest updates, as features
tend to shift swifter than the tides, aye!
```

Clean up resources

If you want to clean up and remove an Azure OpenAI resource, you can delete the resource. Before deleting the resource, you must first delete any deployed models.

- Azure portal
- Azure CLI

Next steps

- Get started with the chat using your own data sample for .NET
- For more examples, check out the Azure OpenAI Samples GitHub repository