

Towards AI · Following

# GPT-4.5: The Next Evolution in AI

Harnessing GPT-4.5's Power with Azure OpenAI & Foundry: A Hands-On Journey into Next-Generation AI 🚀 🤖



Naveen Krishnan

Published in Towards AI

7 min read · Mar 3, 2025



Last week, I shared my thoughts on phi-4 models and their innovative multimodal approach. Today, I'm thrilled to write about GPT-4.5 — a model that not only pushes the boundaries of conversational AI but also makes it easier for developers to integrate powerful language capabilities into their apps via Azure OpenAI and Foundry. Grab your favorite beverage ☕, settle in, and let's explore how GPT-4.5 is set to transform our interactions with technology!



## GPT-4.5

## From GPT-4 to GPT-4.5: A Quick Evolutionary Recap 🔍

GPT-4 paved the way for richer, more nuanced conversations. With GPT-4.5, OpenAI has fine-tuned the art of understanding context and generating responses that are even more human-like. Improvements in efficiency, contextual awareness, and multimodal integration mean that whether you're building chatbots, content generators, or analytical tools, GPT-4.5 can handle your toughest challenges.

But the real magic happens when you combine GPT-4.5 with the robust enterprise-grade capabilities of Azure OpenAI Service — and then manage everything seamlessly using Azure AI Foundry. The result? A platform that's both flexible and scalable for modern app development. ✨

## Key Features of GPT-4.5 💡

- **Enhanced Conversational Depth:** GPT-4.5 can maintain context over longer conversations, delivering responses that feel more intuitive and relevant.
- **Improved Accuracy & Efficiency:** Faster processing means you get your answers almost in real time without sacrificing quality.
- **Humanized Output:** With its refined tone and style, GPT-4.5's responses feel less mechanical and more like chatting with an insightful friend.
- **Seamless Multimodal Integration:** Whether you're feeding text, images, or data from various sources, GPT-4.5 adapts and responds with finesse.
- **Enterprise-Grade Integration:** Through Azure OpenAI and Foundry, GPT-4.5 becomes a part of a secure, scalable, and fully managed ecosystem ideal for production environments.

## Why Azure OpenAI with Foundry? 🔗

Integrating GPT-4.5 via Azure OpenAI Service offers several advantages:

- **Security & Compliance:** Azure ensures your data is handled in compliance with industry standards (GDPR, HIPAA, etc.).
- **Scalability:** Whether you're a startup or an enterprise, Azure's infrastructure scales with your needs.
- **Unified Management:** Azure AI Foundry simplifies the management of models, data sources, and endpoints.
- **Easy Integration:** With robust SDKs and clear sample code, you can quickly incorporate GPT-4.5 into your applications.

In the sections below, I'll walk you through sample code that demonstrates how to invoke GPT-4.5 using Azure OpenAI and Foundry — across multiple languages so you can pick the one that fits your project best. Let's get coding! 🚀

## Invoking GPT-4.5 via Azure OpenAI Using Foundry

### Setting the Stage: Environment & Authentication

Before diving into the code, ensure you have the following prerequisites:

- An **Azure OpenAI Service** resource with GPT-4.5 available in your subscription.
- Access to **Azure AI Foundry**, which helps manage and connect your models.

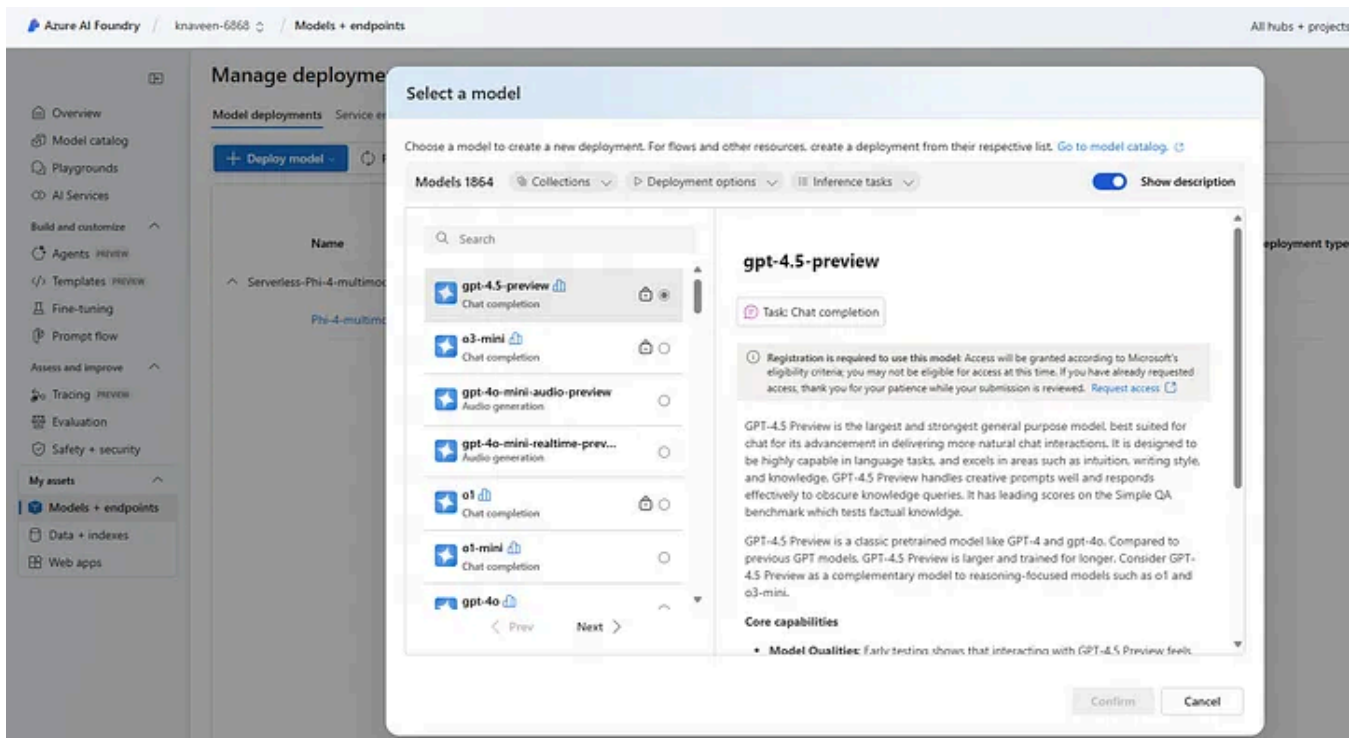


Image Source: Screenshot by User

- Appropriate credentials (API keys or managed identities) stored securely (e.g., in environment variables or Azure Key Vault).

Below, you'll find sample code in C# (.NET) and Python. These examples assume you have set environment variables like `AZURE_OPENAI_ENDPOINT` , `AZURE_OPENAI_API_KEY` , and `AZURE_OPENAI_DEPLOYMENT_NAME` . Adjust these as needed!

### Sample Code in C# (.NET)

Below is a sample console application written in C# that initializes the Azure OpenAI client, configures the Foundry connection, and sends a request to GPT-4.5.

```
using System;
using Azure;
using Azure.AI.OpenAI;
using System.Collections.Generic;

namespace GPT45Demo
{
    class Program
    {
        static void Main(string[] args)
        {
```

```

// Load configuration from environment variables
string endpoint = Environment.GetEnvironmentVariable("AZURE_OPENAI_
string apiKey = Environment.GetEnvironmentVariable("AZURE_OPENAI_AP
string deploymentName = Environment.GetEnvironmentVariable("AZURE_C
// Initialize the Azure OpenAI client using Foundry integration set
OpenAIClient client = new OpenAIClient(new Uri(endpoint), new Azure
// Create a system prompt to guide GPT-4.5's responses
string systemPrompt = "You are a knowledgeable assistant with deep
// Build conversation history - you could extend this to include pr
List<ChatMessage> messages = new List<ChatMessage>
{
    new ChatMessage(ChatRole.System, systemPrompt),
    new ChatMessage(ChatRole.User, "Can you show me how to invoke G
};
// Create chat completion options
ChatCompletionsOptions options = new ChatCompletionsOptions
{
    MaxTokens = 500,
    Temperature = 0.7f,
    // Setting the deployment name from environment variables ensur
    DeploymentName = deploymentName
};
// Add our conversation messages
foreach (var msg in messages)
{
    options.Messages.Add(msg);
}
// Send the request and receive the response
ChatCompletions response = client.GetChatCompletions(options);
// Print the first completion result
Console.WriteLine("Response from GPT-4.5:");
Console.WriteLine(response.Choices[0].Message.Content);
    }
}
}

```

## Explanation:

- We begin by loading our endpoint, API key, and deployment name from environment variables for secure configuration.
- A system prompt is defined to ensure GPT-4.5 understands the tone and style expected.
- The conversation is built as a list of messages (system + user), which is then sent using the Azure OpenAI client.

- Finally, we print out the response — this is the core of our Foundry integration, which helps manage model settings and authentication.

### Sample Code in Python

Here's a Python example using the OpenAI library (configured for Azure OpenAI) to invoke GPT-4.5 with Foundry integration.

```
import os
import openai

# Load environment variables (ensure these are set securely)
azure_endpoint = os.getenv("AZURE_OPENAI_ENDPOINT")
azure_api_key = os.getenv("AZURE_OPENAI_API_KEY")
deployment_name = os.getenv("AZURE_OPENAI_DEPLOYMENT_NAME")
# Configure the OpenAI client to use Azure OpenAI
openai.api_base = azure_endpoint
openai.api_key = azure_api_key
openai.api_version = "2024-10-21" # Adjust API version as needed
# Define a system prompt for context
system_message = (
    "You are a friendly and insightful assistant. Please provide detailed and e"
    "using emojis and human-like language when appropriate. 😊"
)
# Prepare the conversation messages
messages = [
    {"role": "system", "content": system_message},
    {"role": "user", "content": "Show me an example of invoking GPT-4.5 via Azu"
]
# Create a chat completion request
response = openai.ChatCompletion.create(
    model=deployment_name, # This corresponds to the GPT-4.5 deployment in you
    messages=messages,
    max_tokens=500,
    temperature=0.7
)
# Print the generated response
print("Response from GPT-4.5:")
print(response.choices[0].message.content)
```

**Explanation:**

- Environment variables are loaded using `os.getenv()` for secure configuration.
- The `openai` module is configured to point to your Azure endpoint and use your API key.
- We construct a conversation with both a system prompt and a user prompt.
- The `ChatCompletion.create()` method sends our request to the GPT-4.5 model deployed via Azure OpenAI (managed by Foundry).
- Finally, we print the response. This code is ideal for rapid prototyping or integration within larger Python-based applications.

## Integrating with Azure AI Foundry: Best Practices & Tips 🛠️💡

### 1. Secure Your Keys:

Always store your API keys and sensitive configuration data using environment variables or secure vaults (like Azure Key Vault). Avoid hard-coding secrets in your source code. 🔒

### 2. Manage Conversation History:

For a richer dialogue, store past conversation turns (system, user, assistant) and pass them in your request. This context allows GPT-4.5 to generate responses that consider previous interactions. However, be mindful of token limits! 📄

### 3. Customize Your Prompts:


Experiment with the system prompt to adjust tone and response style. GPT-4.5's output can be tailored to different audiences — whether formal, casual, or fun. Emojis, as you've seen, add that extra human touch. 😊

### 4. Monitor & Optimize:

Use telemetry and logging (via Azure Monitor or Application Insights) to track response times, errors, and user interactions. This helps fine-tune both your prompts and integration code. 📊

### 5. Leverage Foundry's Ecosystem:

Azure AI Foundry not only simplifies model deployment and connection management but also allows you to integrate additional data sources (like Azure

Cognitive Search) to augment GPT-4.5's responses. This can be especially powerful for creating context-aware, retrieval-augmented generation (RAG) pipelines. 

### **Deep Dive: Invoking GPT-4.5 in a Production Environment**

When deploying GPT-4.5 in production, consider the following additional points:

- **Token Management:**  
Always monitor token usage to avoid unexpected costs and performance bottlenecks. Limit the conversation history to the most relevant messages.
- **Error Handling:**  
Implement robust error handling for timeouts, API errors, and connectivity issues. Both the .NET and Python examples above include basic structures that you can expand upon for production readiness.
- **Scalability:**  
With Azure's scalable infrastructure, you can handle high volumes of requests. Integrate auto-scaling and load balancing to maintain performance as demand grows.
- **Customization:**  
Use Foundry's configuration capabilities to customize deployment parameters, API versions, and even UI elements if you're building a web app interface on top of GPT-4.5.
- **Feedback & Iteration:**  
Collect user feedback (via built-in UI elements or logging) to iterate on prompts and system settings. This continuous improvement loop is essential for maintaining a high-quality user experience.

### **Conclusion: Embrace the Future with GPT-4.5**

GPT-4.5 represents a significant leap forward in the realm of conversational AI — merging technical excellence with a more natural, humanized interaction style. With seamless integration into the Azure OpenAI ecosystem and the powerful



management capabilities of Azure AI Foundry, developers can now build applications that are not only more intelligent but also easier to manage and scale.

Whether you're working in .NET, Python, or another language, the sample code above should serve as a helpful starting point. Experiment with different prompts, tweak your system messages, and harness the power of GPT-4.5 to transform your applications.

I hope you found this deep dive both informative and inspiring. Drop your thoughts, questions, or feedback in the comments below, and let's continue the conversation!



Happy coding and stay curious! 😊 👍

#### *Additional Resources:*

- [Azure OpenAI Documentation](#)
- [Azure AI Foundry Portal](#)
- [OpenAI API Reference](#)

#### **Thank You!**

Thanks for taking the time to read my story! If you enjoyed it and found it valuable, please consider giving it a clap (or 50!) to show your support. Your claps help others discover this content and motivate me to keep creating more.

Also, don't forget to follow me for more insights and updates on AI. Your support means a lot and helps me continue sharing valuable content with you. Thank you!

Data Science

Data Analysis

Technology

AI

OpenAI



Following

## Published in Towards AI

77K Followers · Last published 1 day ago

The leading AI community and content platform focused on making AI accessible to all. Check out our new course platform: <https://academy.towardsai.net/courses/beginner-to-advanced-llm-dev>



Edit profile

## Written by Naveen Krishnan

158 Followers · 140 Following

AI Architect @ Microsoft. Passionate about leveraging artificial intelligence to solve real-world problems.

### No responses yet



Naveen Krishnan

What are your thoughts?



### More from Naveen Krishnan and Towards AI