Data Science Colle... · Follow publication

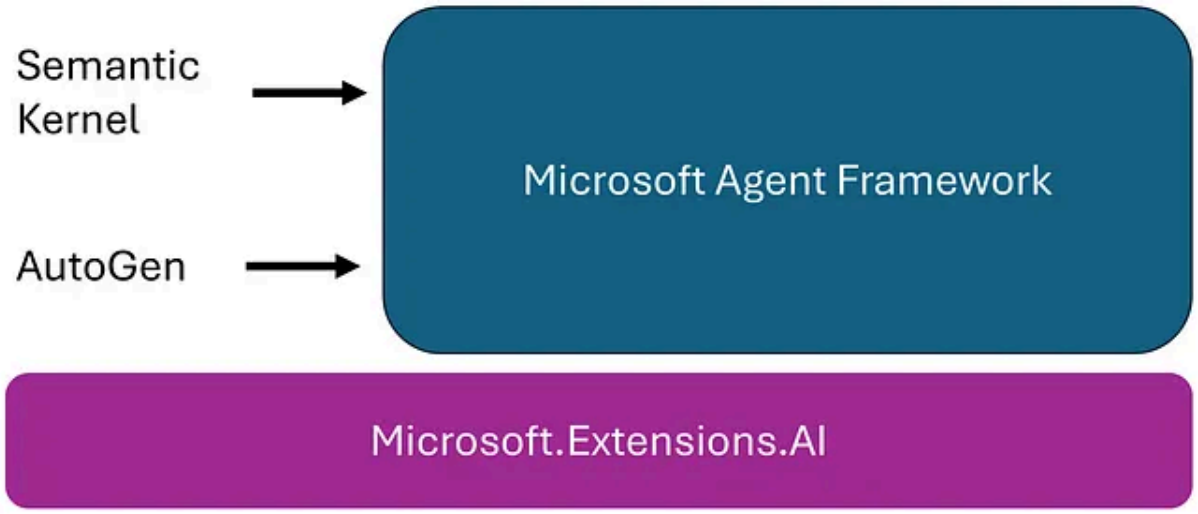


Image Reference: https://devblogs.microsoft.com/dotnet/introducing-microsoft-agent-framework-preview/

# Built my first AI agent with Microsoft Agent Framework in Python and .NET

Your first steps with the unified SDK, demonstrating the simplicity of creating advanced, tool-enabled AI assistants.

5 min read · 2 days ago

Akshay Kokane Follow



Microsoft's **AI Agent Framework** is the newest addition to the ecosystem, inspired by both **Semantic Kernel** and **AutoGen**. It's designed to offer a **unified, seamless developer experience** for building intelligent and extensible AI applications.

Having worked with **Semantic Kernel** for over two years and delivered multiple production-grade AI solutions, I've always admired how **AutoGen** simplified agent orchestration. It felt more developer-friendly and flexible in multi-agent use cases. With this new framework, Microsoft has finally **bridged that gap**, combining the robustness of Semantic Kernel with the usability and flexibility of AutoGen — creating a cohesive environment for AI app development.

In my previous blog, I introduced the framework and compared it with Semantic Kernel and AutoGen.
👉 Read here: *Finally, We Have an Answer Between AutoGen and Semantic Kernel — It's Microsoft Agent Framework*

In this post, let's get hands-on.
We'll walk through the steps to **create an AI Agent** and explore the **developer UI** that comes with the framework. In the next blog, we'll dive deeper into **workflow implementation** and orchestration.

## Use Case: Travel Support Agent

Let's design a simple **Travel Support Agent** capable of:

- Handling **multi-turn conversations**
- Performing **tool calls** (e.g., fetching and updating booking details)

This agent will simulate a real-world travel assistant — fetching booking info, modifying details, and responding naturally in multi-turn dialogue.

## Implementation in Python

```
import asyncio
from agent_framework.azure import AzureOpenAIChatClient
from azure.identity import AzureCliCredential
from agent_framework_devui import serve
from openai import AzureOpenAI
from typing import Annotated
from pydantic import Field


endpoint = "https://<REPLACE_WITH_YOUR_RESOURCENAME>.openai.azure.com"
model_name = "gpt-5-mini"
deployment = "gpt-5-mini"
subscription_key = "<ENTER API KEY>"
```

```
# Tool for getting booking details
def get_booking_info(
    booking_id: Annotated[str, Field(description="The ID of the booking to retr
) -> str:
    """Get the information for a given booking."""
    return f"The information for booking ID {booking_id} is as follows: [detail

# Agent Definition
agent = AzureOpenAIChatClient(
    api_key=subscription_key,
    endpoint=endpoint,
    deployment_name=deployment).create_agent(
    instructions="You are customer support agent specialized in travel and tour
    name="Customer Support Agent",
    tools=[get_booking_info]
)

if __name__ == "__main__":
    # Using dev ui
    serve(entities=[agent], port=8090, auto_open=True)
```

Output Screenshot

The best thing, we have our agent working with

1. Multi-turn support — Using temporary thread which is created by default.

2. Tool calling

You can create a thread object to persist in storage, which allow restarting the conversation across sessions.

```
# Create a new thread.
thread = agent.get_new_thread()

# Run the agent with the thread.
response = await agent.run("Hello, how are you?", thread=thread)
```

Just few lines of codes, and I have slick and detailed Dev UI for tracing and tracking my agent working. This is pretty cool and simple

## Implementation in C#/.NET

Let's try to recreate the agent in C#.

```
using System;
using System.ComponentModel;
using Azure.AI.OpenAI;
using Azure.Identity;
using Microsoft.Agents.AI;
using Microsoft.Extensions.AI;
using OpenAI;

// --- Tool Definition ---
[Description("Get the information for a given booking.")]
static string GetBookingDetails([Description("The ID of the booking to retrieve
    => $"The booking details for {bookingId} are as follows: BookingId {booking

// --- Azure OpenAI Client Setup ---
var endpoint = "https://testmediumazureopenai.openai.azure.com";
var deployment = "gpt-5-mini";

AzureOpenAIClient client = new AzureOpenAIClient(
    new Uri(endpoint),
    new AzureCliCredential());

var chatCompletionClient = client.GetChatClient(deployment);

// --- Agent Creation ---
AIAgent agent = chatCompletionClient.CreateAIAgent(
    instructions: "You are customer support agent specialized in travel and tou
```

```
    name: "John",
    tools: [AIFunctionFactory.Create(GetBookingDetails)]);

// NOTE: .NET still don't have DevUI
// --- Console Application UI ---
Console.WriteLine("Travel Customer Support Agent - John");
Console.WriteLine("====================================");
Console.WriteLine("Hello! I'm John, your travel support agent. How can I help y
Console.WriteLine("Type 'exit()' to end the conversation.");
Console.WriteLine();

string userInput;
// Create a new conversation thread. This thread will store the history of the
// allowing the agent to have context (multi-turn memory).
AgentThread thread = agent.GetNewThread();

do
{
    Console.Write("You: ");
    userInput = Console.ReadLine() ?? string.Empty;

    if (userInput.Equals("exit()", StringComparison.OrdinalIgnoreCase))
    {
        Console.WriteLine("\nThank you for using our travel support service. Ha
        break;
    }

    if (!string.IsNullOrWhiteSpace(userInput))
    {
        try
        {

            AgentRunResponse response = await agent.RunAsync(userInput, thread)
            Console.WriteLine($"John: {response.Text}");
            Console.WriteLine();
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
            Console.WriteLine();
        }
    }
} while (true);
```

Output screenshot

## 💡 First Impressions and Key Takeaways

After building the same simple agent in both Python and .NET, my excitement for this framework is confirmed.

- **Python's Dev UI is a Game-Changer:** The `serve(entities=[agent])` command is fantastic. Getting a fully-featured, trace-enabled web UI for your agent with a single line of code is an incredible accelerator for debugging and demonstration. This is a huge win for rapid iteration.
- **Unified Concepts:** The core concepts are **identical** across both languages. `AIAgent`, `AgentThread`, and `ToolDefinition` are the common vocabulary. This is the unification we've been waiting for. A .NET developer and a Python developer can now discuss agent architecture using the same terms.
- **.NET is Robust and Idiomatic:** The C# implementation feels natural. Using attributes like `[Description]` to define tools fits perfectly into the existing .NET ecosystem and is much cleaner than manually crafting JSON schemas. While it lacks the "out-of-the-box" web UI, the console experience is solid and easy to debug.
- **The "Gap" is Closed:** This framework delivers on its promise. It provides the low-friction, developer-friendly experience of AutoGen (especially in Python) while being built on the robust, production-ready foundation of Semantic Kernel.

We've successfully built a single agent that can hold a conversation and use tools. But the real magic of modern AI applications lies in orchestrating *multiple* specialized agents.

### 🚀 What's Next?

In this post, we scratched the surface. We built a single "customer support" agent.

In the next blog, we'll dive into the feature that truly brings the AutoGen spirit into this new framework: **workflows and orchestration**. We'll explore how to define and manage multi-agent interactions, such as:

- Creating a "routing" agent that delegates tasks.
- Having our `SupportAgent` hand off a complex modification to a separate `BookingModificationAgent`.
- Implementing human-in-the-loop (HIL) for approvals.

This is where the framework's power to build complex, autonomous systems will truly shine. Stay tuned!

*References*:


1. https://learn.microsoft.com/en-us/agent-framework/user-guide/agents/
2. https://github.com/microsoft/agent-framework?tab=readme-ov-file
3. https://devblogs.microsoft.com/dotnet/introducing-microsoft-agent-framework-preview/


Microsoft AI Azure OpenAI Dotnet

Advice, insights, and ideas from the Medium data science community

Follow

## Written by Akshay Kokane

891 followers · 42 following

AI @ Microsoft | Passionate about applied AI, cloud, and building real-world solutions | Writing about AI, cloud, and AI-driven business impact akshaykokane.com

## No responses yet

Bgerby

What are your thoughts?

## More from Akshay Kokane and Data Science Collective

In Microsoft Azure by Akshay Kokane

## Step-by-Step Guide to Building Agentic RAG with Azure AI Search Retrieval Agent

From Retrieval to Reasoning: A Developer’s Guide to Agentic RAG with Azure AI Search

Oct 6 236 1

In Data Science Collective by Amanda Iglesias Moreno

## NotebookLM Just Got a Serious Upgrade—Exploring NotebookLM’s Newest Features and Updates

What’s New and Why It Matters

Sep 29 914 23

In Data Science Collective by Kalle Georgiev

## Markov Chains for Humans: The Simplest Way to Predict What's Next

A plain English guide to modeling step-by-step behavior using only the present moment

Oct 7 861 11

In Data Science Collective by Akshay Kokane

## Finally We have answer between AutoGen and Semantic Kernel—Its Microsoft Agent Framework

The single framework that takes your AI agents from research to reality.

Oct 6 94 1

See all from Akshay Kokane

See all from Data Science Collective

## Recommended from Medium

In Data Science Collective by Ida Silfverskiöld

## Agentic AI: Single vs Multi-Agent Systems

Building with a structured data source in LangGraph

4d ago 450 9

In Towards AI by Teja Kusireddy

## We Spent $47,000 Running AI Agents in Production. Here's What Nobody Tells You About A2A and MCP.

Multi-agent systems are the future. Agent-to-Agent (A2A) communication and Anthropic's Model Context Protocol (MCP) are revolutionary. But...

Oct 16 1.8K 49

In AI Software Engineer by Joe Njenga

## Cursor 2.0 Has Arrived — And Agentic AI Coding Just Got Wild

Cursor has released version 2.0 , bringing the most powerful agentic AI we have seen yet, more autonomous than ever before,here's what's...

3d ago 325 6

In Entrepreneurship Handbook by Joe Procopio

## Amazon's Return To Office Mandates Backfire

The exodus of senior talent from AWS might be why it took forever to figure out why they broke the internet

4d ago 1.5K 35

In Stackademic by Gulam Ali H.

## Stop Writing Boilerplate: Let These 5 Modern C# Features Do the Work

Less clutter, fewer ifs, cleaner models, and APIs that basically document themselves. These tricks just make C# feel... nicer.

2d ago 96 1

Rod Johnson

## Don’t Talk English to Your LLM

Just because LLMs are eloquent in natural language doesn’t mean that we should always communicate with them in it.

3d ago 249 7

See more recommendations