**2.1 Get started with AI agent development on Azure**

**What are AI agents?**

AI agents are smart software services that combine generative AI models with contextual data and the ability to automate tasks based on user input and environmental factors that they perceive.

**Options for agent development**

**Azure AI Foundry Agent Service** is a managed service in Azure that is designed to provide a framework for creating, managing, and using AI agents within Azure AI Foundry. The service is based on the OpenAI Assistants API but with increased choice of models, data integration, and enterprise security; enabling you to use both the OpenAI SDK and the Azure Foundry SDK to develop agentic solutions.

**OpenAI Assistants API** provides a subset of the features in Foundry Agent Service, and can only be used with OpenAI models. In Azure, you can use the Assistants API with Azure OpenAI, though in practice the Foundry Agent Service provides greater flexibility and functionality for agent development on Azure.

**Semantic Kernel** is a lightweight, open-source development kit that you can use to build AI agents and orchestrate multi-agent solutions. The core Semantic Kernel SDK is designed for all kinds of generative AI development, while the Semantic Kernel Agent Framework is a platform specifically optimized for creating agents and implementing agentic solution patterns.

**AutoGen** is an open-source framework for developing agents rapidly. It's useful as a research and ideation tool when experimenting with agents.

**Microsoft 365 Agents SDK** Developers can create self-hosted agents for delivery through a wide range of channels by using the Microsoft 365 Agents SDK. Despite the name, agents built using this SDK are not limited to Microsoft 365, but can be delivered through channels like Slack or Messenger.

**Microsoft Copilot Studio** provides a low-code development environment that "citizen developers" can use to quickly build and deploy agents that integrate with a Microsoft 365 ecosystem or commonly used channels like Slack and Messenger. The visual design interface of Copilot Studio makes it a good choice for building agents when you have little or no professional software development experience.

**Choosing an agent development solution**

With such a wide range of available tools and frameworks, it can be challenging to decide which ones to use. Use the following considerations to help you identify the right choices for your scenario:

For business users with little or no software development experience, Copilot Studio agent builder in Microsoft 365 Copilot Chat provides a way to create simple declarative agents that automate everyday tasks. This approach can empower users across an organization to benefit from AI agents with minimal impact on IT.

If business users have sufficient technical skills to build low-code solutions using Microsoft Power Platform technologies, Copilot Studio enables them to combine those skills with their business domain knowledge and build agent solutions that extend the capabilities of Microsoft 365 Copilot or add agentic functionality to common channels like Microsoft Teams, Slack, or Messenger.

When an organization needs more complex extensions to Microsoft 365 Copilot capabilities, professional developers can use the Microsoft 365 Agents SDK to build agents that target the same channels as Copilot Studio.

To develop agentic solutions that use Azure back-end services with a wide choice of models, custom storage and search services, and integration with Azure AI services, professional developers should use Foundry Agent Service.

Start with Foundry Agent Service to develop single, standalone agents. When you need to build multi-agent solutions, use Semantic Kernel to orchestrate the agents in your solution.

**Azure AI Foundry Agent Service**

Azure AI Foundry Agent Service is a service within Azure that you can use to create, test, and manage AI agents. It provides both a visual agent development experience in the Azure AI Foundry portal and a code-first development experience using the Azure AI Foundry SDK.

**Components of an agent**

Agents developed using Foundry Agent Service have the following elements:

**Model**: A deployed generative AI model that enables the agent to reason and generate natural language responses to prompts. You can use common OpenAI models and a selection of models from the Azure AI Foundry model catalog.

**Knowledge**: data sources that enable the agent to ground prompts with contextual data. Potential knowledge sources include Internet search results from Microsoft Bing, an Azure AI Search index, or your own data and documents.

**Tools**: Programmatic functions that enable the agent to automate actions. Built-in tools to access knowledge in Azure AI Search and Bing are provided as well as a code interpreter tool that you can use to generate and run Python code. You can also create custom tools using your own code or Azure Functions.

Conversations between users and agents take place on a thread, which retains a history of the messages exchanged in the conversation as well as any data assets, such as files, that are generated.

**1.2 Develop an AI agent with Azure AI Foundry Agent Service**

Azure AI Foundry Agent Service is a fully managed service designed to empower developers to securely build, deploy, and scale high-quality, extensible AI agents without needing to manage the underlying compute and storage resources.

**Key features of Foundry Agent Service**

Foundry Agent Service offers several key features:

**Automatic tool calling**: The service handles the entire tool-calling lifecycle, including running the model, invoking tools, and returning results.

**Securely managed data**: Conversation states are securely managed using threads, eliminating the need for developers to handle this manually.

**Out-of-the-box tools**: The service includes tools for file retrieval, code interpretation, and interaction with data sources like Bing, Azure AI Search, and Azure Functions.

**Flexible model selection**: Developers can choose from various models, including Azure OpenAI models and others like Llama 3, Mistral, and Cohere.

**Enterprise-grade security**: The service ensures data privacy and compliance with secure data handling and keyless authentication.

**Customizable storage solutions**: Developers can use either platform-managed storage or bring their own Azure Blob storage for full visibility and control.

**Foundry Agent Service resources**

Foundry Agent Service is fully managed and designed to help developers build agents without having to worry about underlying resources. Through Azure, AI Foundry and the Agent Service will provision the necessary cloud resources. If desired, you can choose to connect your own resources when building your agent

You need to create an Azure AI hub with an Azure AI project for your agent. You can add more Azure services as required.

Two common architectures for Foundry Agent Service solutions are:

**Basic agent setup**: A minimal configuration that includes Azure AI hub, Azure AI project, and Azure AI Services resources.

**Standard agent setup**: A more comprehensive configuration that includes the basic agent setup plus Azure Key Vault, Azure AI Search, and Azure Storage.

**Develop agents with the Azure AI Foundry Agent Service**

Developing apps that use agents

Foundry Agent Service provides several SDKs and a REST API for you to integrate agents into your app using your preferred programming language.

**Below are the following high-level steps that you must implement in your code:**

Connect to the AI Foundry project for your agent, using the project endpoint and Entra ID authentication.

Get a reference to an existing agent that you created in the Azure AI Foundry portal, or create a new one specifying:

The model deployment in the project that the agent should use to interpret and respond to prompts.

Instructions that determine the functionality and behavior of agent.

Tools and resources that the agent can use to perform tasks.

Create a thread for a chat session with the agent. All conversations with an agent are conducted on a stateful thread that retains message history and data artifacts generated during the chat.

Add messages to the thread and invoke it with the agent.

Check the thread status, and when ready, retrieve the messages and data artifacts.

Repeat the previous two steps as a chat loop until the conversation can be concluded.

When finished, delete the agent and the thread to clean up the resources and delete data that is no longer required.

Tools available to your agent

Much of the enhanced functionality of an agent comes from the agent's ability to determine when and how to use tools. Tools make additional functionality available to your agent, and if the conversation or task warrants the use of one or more of the tools, the agent calls that tool and handle the response.

You can assign tools when creating an agent in the Azure AI Foundry portal, or when defining an agent in code using the SDK.

Available tools are split into two categories:

**Knowledge tools** enhance the context or knowledge of your agent. Available knowledge tools include:

**Bing Search**: Uses Bing search results to ground prompts with real-time live data from the web.

**File search**: Grounds prompts with data from files in a vector store.

**Azure AI Search**: Grounds prompts with data from Azure AI Search query results.

**Microsoft Fabric**: Uses the Fabric Data Agent to ground prompts with data from your Fabric data stores.

**Action tools** perform an action or run a function. Available tools include:

**Code Interpreter**: A sandbox for model-generated Python code that can access and process uploaded files.

**Custom function**: Call your custom function code – you must provide function definitions and implementations.

**Azure Function**: Call code in serverless Azure Functions.

**OpenAPI Spec**: Call external APIs based on the OpenAPI 3.0 spec.

**By connecting built-in and custom tools, you can allow your agent to perform countless tasks on your behalf.**

**Integrate custom tools into your agent**

Azure AI Foundry Agent Service offers a powerful platform for integrating custom tools to enhance productivity and provide tailored solutions for specific business needs. By using these custom tools, businesses can achieve greater efficiency and effectiveness in their operations.

**Why use custom tools?**

Custom tools in Azure AI services can significantly enhance productivity by automating repetitive tasks and streamlining workflows that are specific to your use case. These tools improve accuracy by providing precise and consistent outputs, reducing the likelihood of human error. Additionally, custom tools offer tailored solutions that address specific business needs, enabling organizations to optimize their processes and achieve better outcomes.

Adding tools makes custom functionality available for the agent to use, depending on how it decides to respond to the user prompt.

**Options for implementing custom tools**

Azure AI Foundry Agent Service offers various custom tools that enhance the capabilities and efficiency of your AI agents. These tools allow for scalable interoperability with various applications, making it easier to integrate with existing infrastructure or web services.

**Custom tool options available in Azure AI Foundry Agent Service**

**Custom function**: Function calling allows you to describe the structure of custom functions to an agent and return the functions that need to be called along with their arguments. The agent can dynamically identify appropriate functions based on their definitions. This feature is useful for integrating custom logic and workflows, in a selection of programming languages, into your AI agents.

**Azure Functions**: Enable you to create intelligent, event-driven applications with minimal overhead. They support triggers and bindings, which simplify how your AI Agents interact with external systems and services. Triggers determine when a function executes, while bindings facilitate streamlined connections to input or output data sources.

**OpenAPI specification tools**: These tools allow you to connect your Azure AI Agent to an external API using an OpenAPI 3.0 specification. This provides standardized, automated, and scalable API integrations that enhance the capabilities of your agent. OpenAPI specifications describe HTTP APIs, enabling people to understand how an API works, generate client code, create tests, and apply design standards.

**Azure Logic Apps**: This action provides low-code/no-code solutions to add workflows and connects apps, data, and services with the low-code Logic App.

**How to integrate custom tools**

Custom tools in an agent can be defined in a handful of ways, depending on what works best for your scenario. You may find that your company already has Azure Functions implemented for your agent to use, or a public OpenAPI specification gives your agent the functionality you're looking for.

**Function Calling**

Function calling allows agents to execute predefined functions dynamically based on user input. This feature is ideal for scenarios where agents need to perform specific tasks, such as retrieving data or processing user queries, and can be done in code from within the agent. Your function may call out to other APIs to get additional information or initiate a program.

**Azure Functions**

Azure Functions provide serverless computing capabilities for real-time processing. This integration is ideal for event-driven workflows, enabling agents to respond to triggers such as HTTP requests or queue messages.

**OpenAPI Specification**

OpenAPI defined tools allow agents to interact with external APIs using standardized specifications. This approach simplifies API integration and ensures compatibility with various services. The Foundry Agent Service uses OpenAPI 3.0 specified tools.

Currently, three authentication types are supported with OpenAPI 3.0 tools: anonymous, API key, and managed identity.