# Introduction

* • 2 minutes

AI agents are transforming how applications interact with users and automate tasks. Unlike traditional programs, AI agents use generative AI to interpret data, make decisions, and complete tasks with minimal human intervention. These agents use large language models to streamline complex workflows, making them ideal for automating business processes.

Developers can build AI agents using different tools, including the Semantic Kernel SDK. This open-source SDK simplifies the integration of AI models into applications. The Semantic Kernel Agent Framework supports different types of agents, including ChatCompletionAgent, OpenAIAssistantAgent, and AzureAIAgent. This module focuses on the AzureAIAgent, a type of agent that streamlines the features and functionalities of Azure AI Agent Service.

ChatCompletionAgent

OpenAIAssistantAgent

AzureAIAgent

AzureAIAgent

Azure AI Agent Service is a fully managed service that enables developers to securely build, deploy, and scale high-quality extensible AI agents. Using the Azure AI Agent Service, developers don't need to manage the underlying compute or storage resources. Using the Semantic Kernel Agent Framework allows developers to quickly build agents on the Azure AI Agent Service, with support for natural language processing and access to built-in tools in just a few lines of code.

While Azure AI Agent Service provides a powerful foundation for building AI agents, Semantic Kernel offers additional flexibility and scalability. If you've already started developing agents with Semantic Kernel, you can seamlessly integrate Azure AI Agent capabilities, such as built-in tools and project deployment, without rewriting your code. Also, if your solution requires multiple types of agents, using Semantic Kernel ensures consistency across your implementation. Finally, if you're planning to develop multi-agent solutions, Semantic Kernel's GroupChat feature allows you to orchestrate collaborative agents efficiently—a topic covered in more detail in a later module.

Suppose you need to develop an AI agent that automatically formats and emails expense reports for employees. Your AI agent can extract data from submitted expense reports, format them correctly, and send them to the appropriate recipients. To do this, you can use the Semantic Kernel Agent Framework. The plugins and functions feature allows your AI agent to interact with APIs, retrieve necessary data, and complete tasks.

In this module, you learn about the core features of the Semantic Kernel SDK and the Agent Framework. You also learn how to create your own AI agents and extend their capabilities with plugin functions.

After completing this module, you're now able to:

* • Use Semantic Kernel to connect to an Azure AI Foundry project.
* • Create Azure AI Agent Service agents using the Semantic Kernel SDK.
* • Integrate plugin functions with your AI agent.

## Feedback

Was this page helpful?

# Understand Semantic Kernel AI agents

* • 6 minutes

An AI agent is a program that uses generative AI to interpret data, make decisions, and perform tasks on behalf of users or other applications. AI agents rely on large language models to perform their tasks. Unlike conventional programs, AI agents can function autonomously, handling complex workflows and automating processes without requiring continuous human oversight.

AI Agents can be developed using many different tools and platforms, including the Semantic Kernel SDK. Semantic Kernel is an open-source SDK that enables developers to easily integrate the latest AI models into their applications. Part of that SDK includes the Semantic Kernel Agent Framework, which allows developers to quickly create functional agents that can use natural language processing to complete tasks.

## Semantic Kernel core components

The Semantic Kernel offers different components that can be used individually or combined.

* • AI service connectors - connect the code to AI services from different providers under a common interface. Supported services include Chat Completion, Text Generation, and more.

AI service connectors - connect the code to AI services from different providers under a common interface. Supported services include Chat Completion, Text Generation, and more.

* • Memory connectors - expose vector stores from other providers under a common interface.

Memory connectors - expose vector stores from other providers under a common interface.

* • Functions and plugins - containers for functions that are registered with the kernel. Once registered, functions can be invoked by the AI or through prompt templates.

Functions and plugins - containers for functions that are registered with the kernel. Once registered, functions can be invoked by the AI or through prompt templates.

* • Prompt templates - combine instructions, user input, and function outputs into a reusable format. Prompt templates allow AI models to execute predefined steps dynamically.

Prompt templates - combine instructions, user input, and function outputs into a reusable format. Prompt templates allow AI models to execute predefined steps dynamically.

* • Filters - allow custom actions to be performed before and after a function or prompt is invoked. When registered, function filters act as outer layers and prompt filters as inner layers.

Filters - allow custom actions to be performed before and after a function or prompt is invoked. When registered, function filters act as outer layers and prompt filters as inner layers.

## Agent framework components

The Agent Framework within Semantic Kernel helps streamline the creation of agents and enables multi-agent collaboration in conversations while integrating human input. The framework supports different types of agents, including ChatCompletionAgent, OpenAIAssistantAgent, and AzureAIAgent.

ChatCompletionAgent

OpenAIAssistantAgent

AzureAIAgent

### What is an Azure AI Agent?

The AzureAIAgent class provides a seamless way to build and interact with AI agents using the Azure AI Agent Service. It abstracts the complexity of managing AI agents by offering a more structured and intuitive interface within the Semantic Kernel Agent Framework. Key benefits include:

AzureAIAgent

* • Simplified agent creation – The AzureAIAgent class allows developers to define AI agents with minimal configuration, leveraging the power of Azure AI Agent Service without managing the underlying infrastructure.

Simplified agent creation – The AzureAIAgent class allows developers to define AI agents with minimal configuration, leveraging the power of Azure AI Agent Service without managing the underlying infrastructure.

* • Automatic tool invocation – The agent can automatically call and execute tools, integrating seamlessly with Azure AI Search, Bing, Azure Functions, and more.

Automatic tool invocation – The agent can automatically call and execute tools, integrating seamlessly with Azure AI Search, Bing, Azure Functions, and more.

* • Thread and conversation management – Provides built-in mechanisms for managing conversation states, ensuring smooth multi-agent interactions.

Thread and conversation management – Provides built-in mechanisms for managing conversation states, ensuring smooth multi-agent interactions.

* • Secure enterprise integration – Enables secure and compliant AI agent development with keyless authentication and customizable storage options.

Secure enterprise integration – Enables secure and compliant AI agent development with keyless authentication and customizable storage options.

By using the AzureAIAgent class, developers can take full advantage of Azure AI Agent Service while taking advantage of the features offered by the Semantic Kernel SDK. This allows for robust AI-driven workflows that scale efficiently across enterprise applications.

AzureAIAgent

### Agent framework core concepts

* • Agent - abstraction for AI agents, with specialized subclasses like AzureAIAgent, allowing for task completion and human interaction in conversations.

Agent - abstraction for AI agents, with specialized subclasses like AzureAIAgent, allowing for task completion and human interaction in conversations.

* • Agent threads - manage conversation state and stores conversations.

Agent threads - manage conversation state and stores conversations.

* • Agent chat - the foundation for multi-agent interactions, allows for structured conversations and collaboration.

Agent chat - the foundation for multi-agent interactions, allows for structured conversations and collaboration.

* • Agent channel - used for custom agent development, allows different types of agents to participate in AgentChat.

Agent channel - used for custom agent development, allows different types of agents to participate in AgentChat.

* • Agent messages - a unified structure for agent communication, provides seamless communication and integration with existing AI workflows.

Agent messages - a unified structure for agent communication, provides seamless communication and integration with existing AI workflows.

* • Templating - like Semantic Kernel prompt templates, templates use dynamic prompt configurations to shape agent behavior.

Templating - like Semantic Kernel prompt templates, templates use dynamic prompt configurations to shape agent behavior.

* • Functions and plugins - like Semantic Kernel plugins, agent plugin functions allow developers to extend agent capabilities by incorporating custom functions.

Functions and plugins - like Semantic Kernel plugins, agent plugin functions allow developers to extend agent capabilities by incorporating custom functions.

This framework supports autonomous, multi-agent AI behaviors while maintaining consistency with Semantic Kernel's design, allowing developers to build intelligent, adaptable AI systems.

## Feedback

Was this page helpful?

# Create an Azure AI agent with Semantic Kernel

* • 7 minutes

AzureAIAgent is a specialized agent within the Semantic Kernel framework, designed to provide advanced conversational capabilities with seamless tool integration. It automates tool calling, eliminating the need for manual parsing and invocation. The agent also securely manages conversation history using threads, reducing the overhead of maintaining state. The AzureAIAgent class supports many built-in tools, including file retrieval, code execution, and data interaction via Bing, Azure AI Search, Azure Functions, and OpenAPI.

## Creating an AzureAIAgent

An AzureAIAgent object encapsulates all the core capabilities you typically use the Kernel for, like function execution, planning, and memory access. This object acts as a self-contained agent runtime.

To use an AzureAIAgent:

* • Create an Azure AI Foundry project.
* • Add the project connection string to your Semantic Kernel application code.
* • Create an AzureAIAgentSettings object.
* • Create an AzureAIAgent client.
* • Create an agent definition on the agent service provided by the client.
* • Create an agent based on the definition.

Here's the code that illustrates how to create an AzureAIAgent:

from azure.identity.aio import DefaultAzureCredential  
from semantic\_kernel.agents import AzureAIAgent, AzureAIAgentThread, AzureAIAgentSettings  
  
# Create an AzureAIAgentSettings object  
ai\_agent\_settings = AzureAIAgentSettings()  
  
# Create an AzureAIAgent client  
async with (@  
 DefaultAzureCredential() as creds,  
 AzureAIAgent.create\_client(credential=creds) as client,  
):  
 # Create an agent definition on the agent service provided by the client  
 agent\_definition = await client.agents.create\_agent(  
 model=ai\_agent\_settings.model\_deployment\_name,  
 name="<name>",  
 instructions="<instructions>",  
 )  
  
 # Create the AI agent based on the agent definition  
 agent = AzureAIAgent(  
 client=client,  
 definition=agent\_definition,  
 )

from azure.identity.aio import DefaultAzureCredential  
from semantic\_kernel.agents import AzureAIAgent, AzureAIAgentThread, AzureAIAgentSettings  
  
# Create an AzureAIAgentSettings object  
ai\_agent\_settings = AzureAIAgentSettings()  
  
# Create an AzureAIAgent client  
async with (@  
 DefaultAzureCredential() as creds,  
 AzureAIAgent.create\_client(credential=creds) as client,  
):  
 # Create an agent definition on the agent service provided by the client  
 agent\_definition = await client.agents.create\_agent(  
 model=ai\_agent\_settings.model\_deployment\_name,  
 name="<name>",  
 instructions="<instructions>",  
 )  
  
 # Create the AI agent based on the agent definition  
 agent = AzureAIAgent(  
 client=client,  
 definition=agent\_definition,  
 )

Once your agent is defined, you can create a thread to interact with your agent and invoke responses for inputs. For example:

# Create the agent thread  
thread: AzureAIAgentThread = AzureAIAgentThread(client=client)  
  
try:  
 # Create prompts   
 prompt\_messages = ["What are the largest semiconductor manufacturing companies?"]  
  
 # Invoke a response from the agent  
 response = await agent.get\_response(messages=prompt\_messages, thread\_id=thread.id)  
  
 # View the response  
 print(response)  
finally:  
 # Clean up the thread  
 await thread.delete() if thread else None

# Create the agent thread  
thread: AzureAIAgentThread = AzureAIAgentThread(client=client)  
  
try:  
 # Create prompts   
 prompt\_messages = ["What are the largest semiconductor manufacturing companies?"]  
  
 # Invoke a response from the agent  
 response = await agent.get\_response(messages=prompt\_messages, thread\_id=thread.id)  
  
 # View the response  
 print(response)  
finally:  
 # Clean up the thread  
 await thread.delete() if thread else None

### AzureAIAgent key components

The Semantic Kernel AzureAIAgent object relies on the following components to function:

* • AzureAISAgentSettings - an object that automatically includes the Azure AI Agent settings from the environment configuration. These settings will be used by the AzureAIAgents you create.

AzureAISAgentSettings - an object that automatically includes the Azure AI Agent settings from the environment configuration. These settings will be used by the AzureAIAgents you create.

* • AzureAIAgent client - an object that manages the connection to your Azure AI Foundry project. This object allows you to access the services and models associated with your project.

AzureAIAgent client - an object that manages the connection to your Azure AI Foundry project. This object allows you to access the services and models associated with your project.

* • Agent service - the AzureAIAgent client also contains an agent operations service. This service helps streamline the process of creating, managing, and running the agents for your project.

Agent service - the AzureAIAgent client also contains an agent operations service. This service helps streamline the process of creating, managing, and running the agents for your project.

* • Agent definition - the AzureAI Agent model created via the AzureAI Project client. This definition specifies the AI deployment model that should be used, and the name and instructions for the agent.

Agent definition - the AzureAI Agent model created via the AzureAI Project client. This definition specifies the AI deployment model that should be used, and the name and instructions for the agent.

* • AzureAIAgentThread - automatically maintains the conversation history between agents and users, and the state. You can add messages to a thread and use the agent to invoke a response from the LLM.

AzureAIAgentThread - automatically maintains the conversation history between agents and users, and the state. You can add messages to a thread and use the agent to invoke a response from the LLM.

These components work together to allow you to create an agent with instructions to define its purpose and invoke responses from the AI model.

## Feedback

Was this page helpful?

# Add plugins to Azure AI agent

* • 5 minutes

In Semantic Kernel, plugins allow your AI agent to use existing APIs to perform tasks it couldn't do on its own. Plugins work through function calling, allowing AI to request and use specific functions. Semantic Kernel routes the request to the appropriate function in your codebase and returns the results back to the LLM so the LLM can generate a final response.

To enable automatic orchestration with function calling, plugins also need to provide details that describe how they behave. The function's input, output, and side effects should be described in a way that the AI can understand, otherwise, the AI will not correctly call the function.

## How to use plugins with AzureAIAgent

* • Define your plugin  
  You can create a plugin by defining a class and annotating its methods with the kernel\_function decorator. The decorator lets Semantic Kernel know that this function can be called by the AI or referenced in a prompt. The kernel\_function decorator also supports a description attribute to help the AI understand how to use the function.

Define your plugin

You can create a plugin by defining a class and annotating its methods with the kernel\_function decorator. The decorator lets Semantic Kernel know that this function can be called by the AI or referenced in a prompt. The kernel\_function decorator also supports a description attribute to help the AI understand how to use the function.

kernel\_function

kernel\_function

description

* • Add the plugin to your agent  
  Once you define your plugin, you can add it to your AzureAIAgent by creating a new instance of the plugin and adding it to the agent's plugin collection.

Add the plugin to your agent

Once you define your plugin, you can add it to your AzureAIAgent by creating a new instance of the plugin and adding it to the agent's plugin collection.

* • Invoke the plugin's functions  
  You can invoke your plugin's functions by using prompts on your agent's message thread. For example, if you have a plugin function called get\_tasks, your prompt to the agent might be "What tasks do I have?".

Invoke the plugin's functions

You can invoke your plugin's functions by using prompts on your agent's message thread. For example, if you have a plugin function called get\_tasks, your prompt to the agent might be "What tasks do I have?".

get\_tasks

By following these steps, you can extend your AzureAIAgent with plugins, allowing it to interact with APIs and perform more advanced tasks. This makes your AI more powerful and capable of handling real-world applications efficiently.

## Feedback

Was this page helpful?

# Exercise - Develop an Azure AI agent with the Semantic Kernel SDK

* • 30 minutes

Now you're ready to build an agent with the Semantic Kernel Agent Framework. In this exercise, you use Azure AI Agent Service and Semantic Kernel to create an AI agent that creates an expense claim email.

Note

To complete this exercise, you will need a Microsoft Azure subscription. If you don't already have one, you can sign up for one.

If you need to set up your computer for this exercise, you can use this setup guide and then follow the exercise instructions linked below. Note that the setup guide is designed for multiple development exercises, and may include software that is not required for this specific exercise. Additionally, due to the range of possible operating systems and setup configurations, we can't provide support if you choose to complete the exercise on your own computer.

Launch the exercise and follow the instructions.

Tip

After completing the exercise, if you're finished exploring Azure AI Agents, delete the Azure resources that you created during the exercise.

## Feedback

Was this page helpful?

# Knowledge check

* • 3 minutes

What are the key steps to create an AzureAIAgent?

Initialize the agent by defining an LLM model in the AzureAIAgentThread constructor.

Create an Azure AI agents client, create an agent definition, and instantiate the agent.

Deploy a custom AI model before creating an AzureAIAgentSettings object.

Which component in the Agent Framework manages conversation state and stores messages?

Agent threads

Agent chat

AI service connectors

Which step is necessary to enable an AzureAIAgent to use a plugin?

Modify the AI model's architecture to integrate the plugin

Configure the plugin in Azure portal before using it in Semantic Kernel

Define a class with methods annotated using the kernel\_function decorator

kernel\_function

You must answer all questions before checking your work.

You must answer all questions before checking your work.

## Feedback

Was this page helpful?

# Summary

* • 2 minutes

In this module, you learned how the Semantic Kernel Agent Framework enables developers to build AI agents. You learned about the components of the Semantic Kernel SDK and the core concepts of the agent framework. You also learned how to create plugin functions to extend your agent's capabilities. By applying these concepts and skills, you can leverage the Semantic Kernel Agent Framework to create dynamic, adaptable AI solutions that enhance user interactions and automate complex tasks.

More reading:

* • Learn more about the Semantic Kernel Agent Framework
* • Learn more about Azure AI Agent Service
* • Practice developing generative AI apps with Semantic Kernel

## Feedback

Was this page helpful?