Develop AI agents with the Microsoft Foundry extension in Visual Studio Code

1. Introduction

https://learn.microsoft.com/en-us/training/modules/develop-ai-agents-vs-code/1-introduction

Introduction

Completed

- 1 minute

As generative AI models become more powerful and accessible, developers are moving beyond simple chat applications to build intelligent agents that can automate complex tasks. These AI agents combine large language models with specialized tools to access data, perform actions, and complete entire workflows with minimal human intervention.

Visual Studio Code is a powerful platform for AI agent development, especially with the Microsoft Foundry extension. This extension brings enterprise-grade AI capabilities directly into your development environment. With the Microsoft Foundry for Visual Studio Code extension, you can discover and deploy models, create and configure agents, and test them in interactive playgrounds—all without leaving your code editor.

This module introduces the core concepts of AI agent development and shows you how to use the Microsoft Foundry extension for Visual Studio Code to build, test, and deploy intelligent agents. You'll learn how to use Microsoft Foundry's Agent Service to create agents that can handle complex tasks, use various tools, and integrate with your existing applications.

2. Get started with the Microsoft Foundry extension

https://learn.microsoft.com/en-us/training/modules/develop-ai-agents-vs-code/2-get-started-with-extension

Get started with the Microsoft Foundry extension

Completed

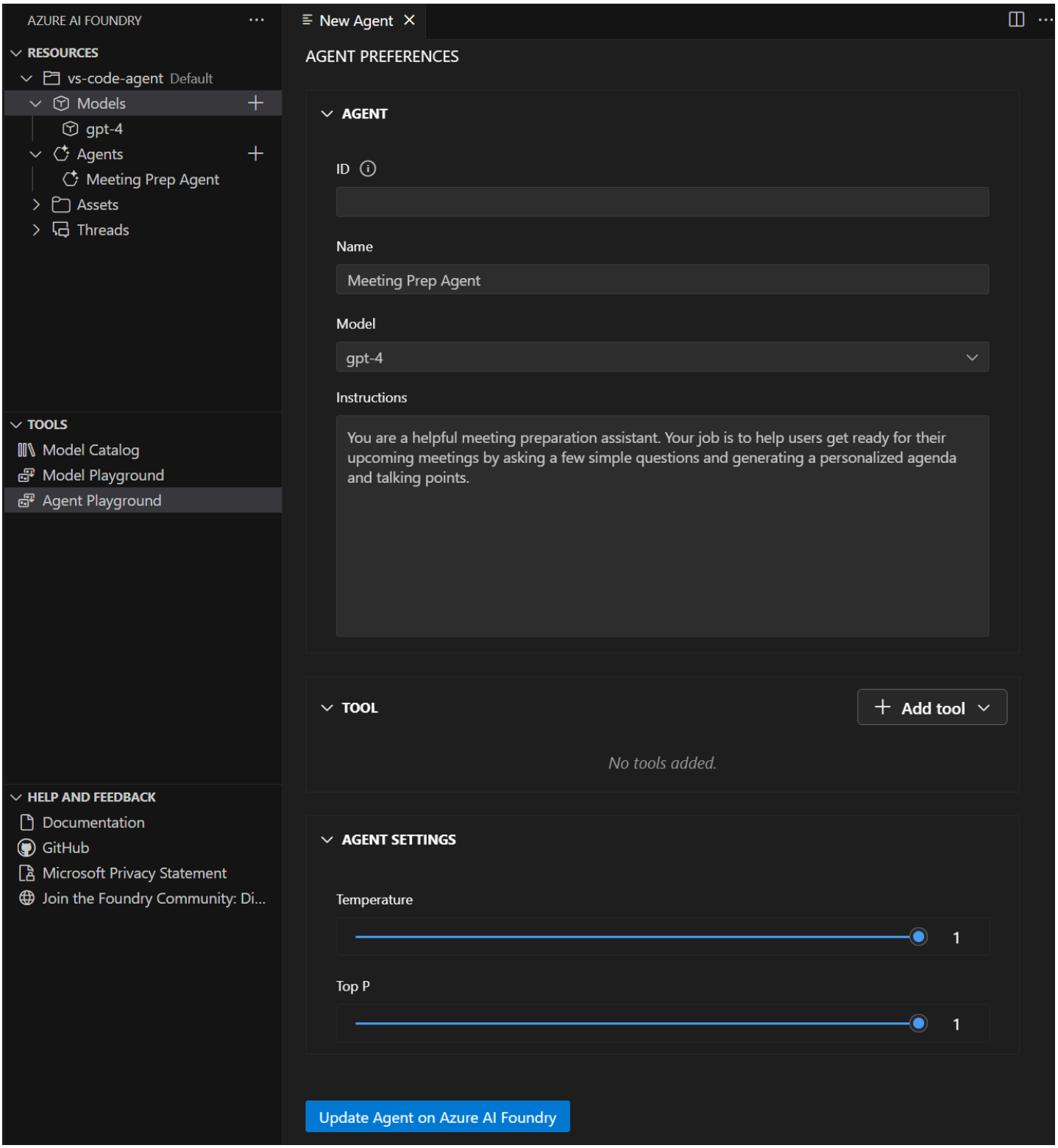
- 3 minutes

The Microsoft Foundry for Visual Studio Code extension transforms your development environment into a comprehensive platform for building, testing, and deploying AI agents. This extension provides direct access to the capabilities of Microsoft Foundry's Agent Service without leaving your code editor, streamlining the entire agent development workflow.

What is the Microsoft Foundry for Visual Studio Code extension?

The Microsoft Foundry for Visual Studio Code extension is a powerful tool that brings enterprise-grade AI agent development capabilities directly into Visual Studio Code. It provides an integrated experience for:

- **Agent Discovery and Management** - Browse, create, and manage AI agents within your Microsoft Foundry projects
- **Visual Agent Designer** - Use an intuitive interface to configure agent instructions, tools, and capabilities
- **Integrated Testing** - Test agents in real-time using the built-in playground without switching contexts
- **Code Generation** - Generate sample integration code to connect agents with your applications
- **Deployment Pipeline** - Deploy agents directly to Microsoft Foundry for production use



Key features and capabilities

The extension offers several key features that accelerate AI agent development:

Agent Designer Interface

A visual designer that simplifies agent creation and configuration. You can define agent instructions, select appropriate models, and configure tools through an intuitive graphical interface.

Built-in Playground

An integrated testing environment where you can interact with your agents in real-time, test different scenarios, and refine agent behavior before deployment.

Tool Integration

Seamless integration with various tools including:

- **RAG (Retrieval-Augmented Generation)** for knowledge-based responses
- **Search capabilities** for information retrieval
- **Custom actions** for specific business logic
- **Model Context Protocol (MCP) servers** for extended functionality

Project Integration

Direct connection to your Microsoft Foundry projects, allowing you to work with existing resources and deploy new agents to your established infrastructure.

Installing the extension

To get started with the Microsoft Foundry for Visual Studio Code extension, first you need to have Visual Studio Code installed on your machine. You can download it from the Visual Studio Code website.

You can install the Microsoft Foundry extension directly from the Visual Studio Code Marketplace:

1. Open Visual Studio Code.
2. Select Extensions from the left pane, or press `Ctrl+Shift+X`.
3. Search for and select Microsoft Foundry.
4. Select Install.
5. Verify the extension is installed successfully from the status messages.

Getting started workflow

The typical workflow for using the Microsoft Foundry extension follows these steps:

1. **Install and configure** the extension in Visual Studio Code
2. **Connect** to your Microsoft Foundry project
3. **Create or import** an AI agent using the designer
4. **Configure** agent instructions and add necessary tools
5. **Test** the agent using the integrated playground
6. **Iterate** on the design based on test results
7. **Generate code** for application integration

This streamlined workflow enables rapid prototyping and deployment of AI agents, making it easier for developers to build intelligent automation solutions that can handle complex real-world tasks.

The Microsoft Foundry for Visual Studio Code extension represents a significant step forward in making AI agent development more accessible and efficient, providing developers with enterprise-grade tools in a familiar development environment.

3. Develop AI agents in Visual Studio Code

https://learn.microsoft.com/en-us/training/modules/develop-ai-agents-vs-code/3-agent-development

Develop AI agents in Visual Studio Code

Completed

- 5 minutes

Creating and configuring AI agents in Visual Studio Code using the Microsoft Foundry extension provides a streamlined development experience that combines the power of Microsoft Foundry Agent Service with the familiar Visual Studio Code environment. This approach enables you to design, configure, and test agents without leaving your development environment.

Understanding Microsoft Foundry Agent Service

Microsoft Foundry Agent Service is a managed service in Azure designed to provide a comprehensive framework for creating, managing, and deploying AI agents. The service builds on the OpenAI Assistants API foundation while offering enhanced capabilities including:

- **Expanded model choice** - Support for multiple AI models beyond OpenAI
- **Enterprise security** - Built-in security features for production environments
- **Advanced data integration** - Seamless connection to Azure data services
- **Tooling ecosystem** - Access to a various built-in and custom tools

The Visual Studio Code extension provides direct access to these capabilities through an intuitive interface that simplifies the agent development process.

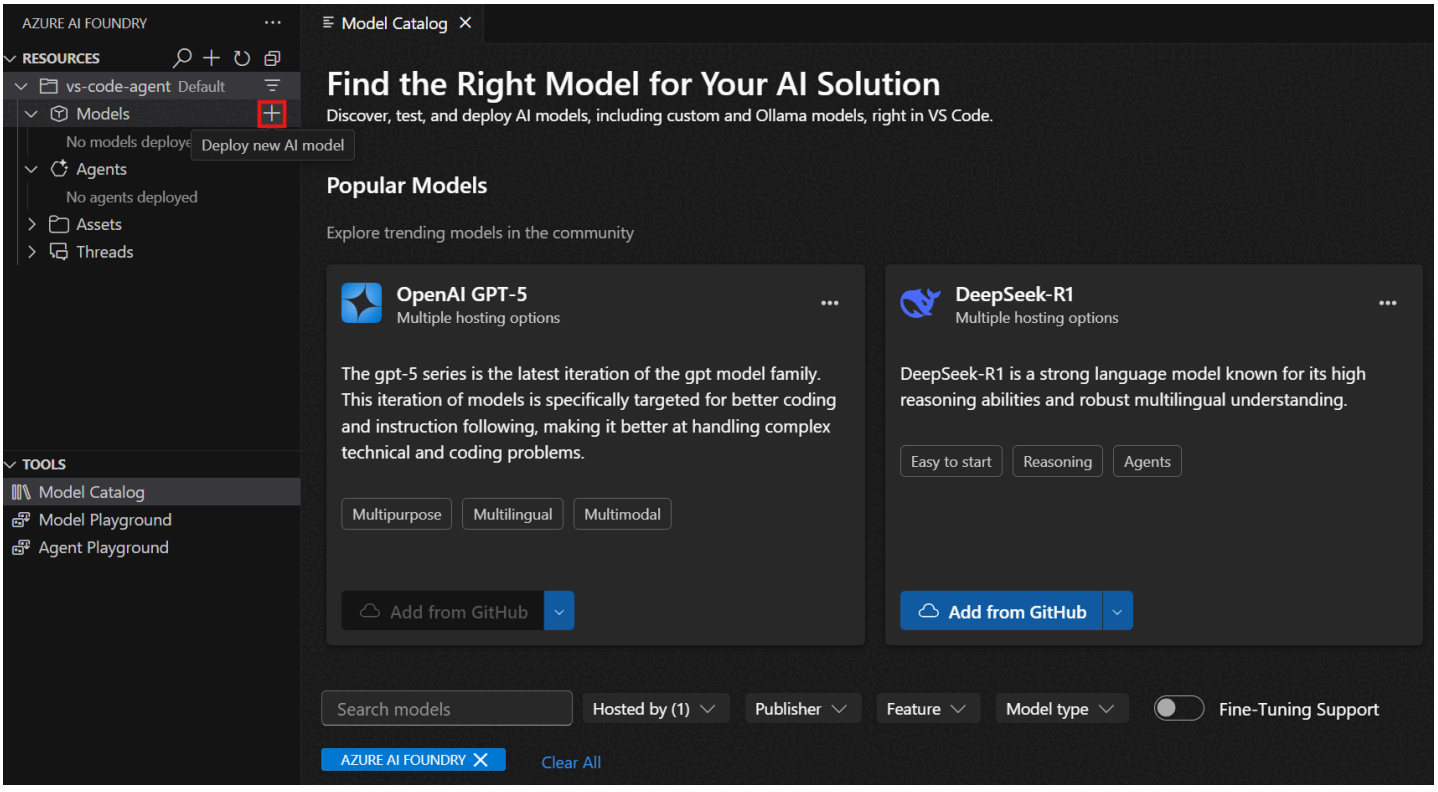
Creating agents with the extension

The Microsoft Foundry extension provides multiple ways to create AI agents, whether you're starting from scratch or building on existing work. The flexible approach accommodates different development preferences and scenarios.

Prerequisites for agent creation

Before creating an agent, complete the following steps:

1. Complete the extension setup and sign in to your Azure account
2. Create a default Microsoft Foundry project, or select an existing one
3. Select and deploy the model for your agent to use, or use an existing deployment

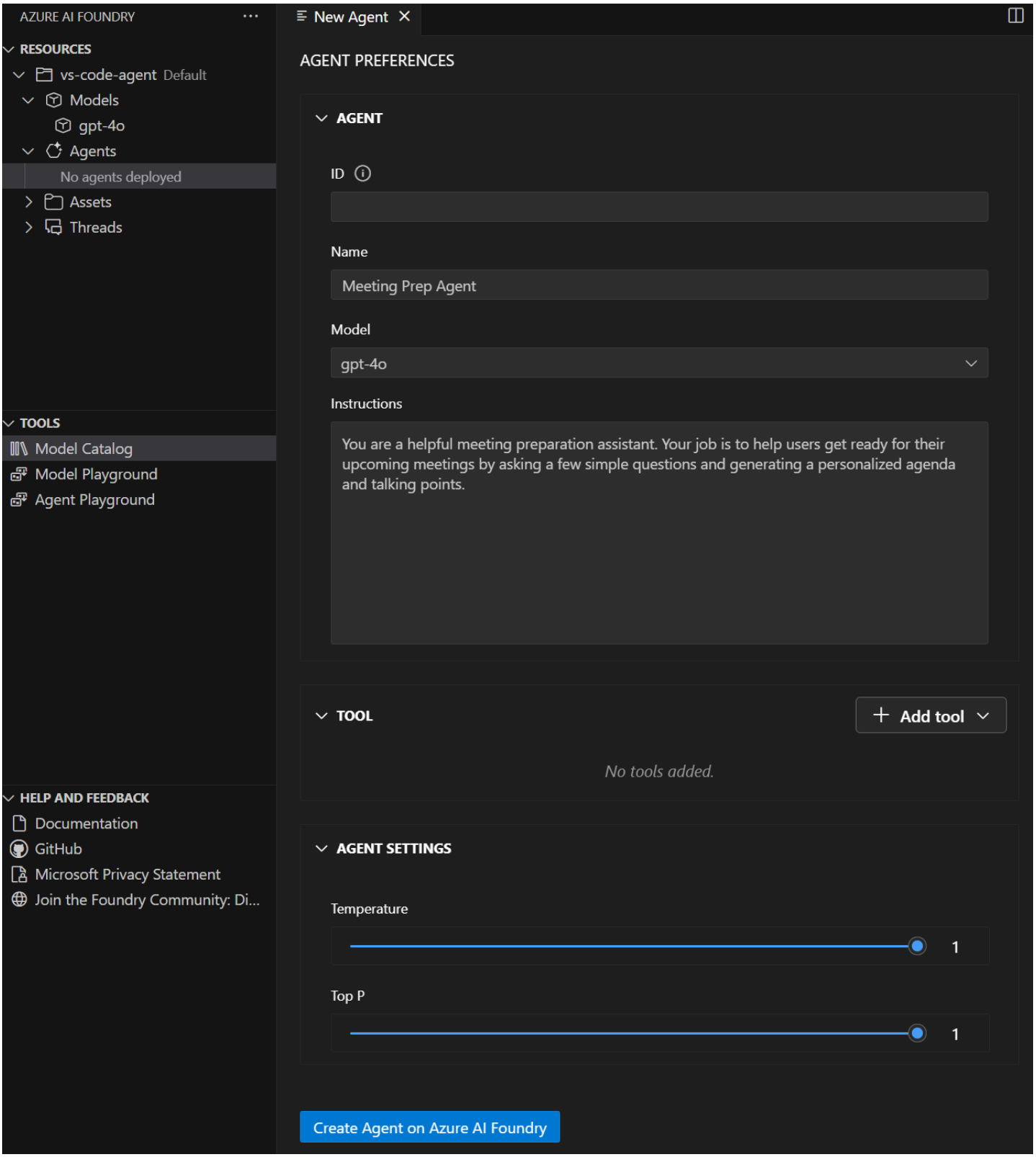


Creating a new agent

To create a new AI agent, follow these steps:

1. Open the Microsoft Foundry Extension view in Visual Studio Code
2. Navigate to the Resources section
3. Select the **+** (plus) icon next to the Agents subsection to create a new AI agent
4. Configure the agent properties in the Agent Designer view that opens

When you create an agent, the extension opens both the agent .yaml file and the Designer view, providing you with both a visual interface and direct access to the configuration file.



Configuring agent properties

Once you create an agent, the extension provides comprehensive configuration options to define how your agent behaves and interacts with users. The Agent Designer provides an intuitive interface for setting up these properties.

Basic Configuration

In the Agent Designer, configure the following essential properties:

- **Agent name** - Enter a descriptive name for your agent in the prompt
- **Model selection** - Choose your model deployment from the dropdown (this is the deployment name you chose when deploying a model)
- **Description** - Add a clear description of what your agent does
- **System instructions** - Define the agent's behavior, personality, and response style
- **Agent ID** - Automatically generated by the extension

Understanding the agent YAML file

Your AI agent is defined in a YAML file that contains all configuration details. Here's an example structure:

```yaml
 # yaml-language-server: $schema=https://aka.ms/ai-foundry-vsc/agent/1.0.0
version: 1.0.0
name: my-agent
description: Description of the agent
id: ''
metadata:
  authors:
    - author1
    - author2
  tags:
    - tag1
    - tag2
model:
  id: 'gpt-4o-1'
  options:
    temperature: 1
    top_p: 1
instructions: Instructions for the agent
tools: []
```

This YAML file is opened automatically alongside the Designer view, allowing you to work with either the visual interface or edit the configuration directly.

Agent instruction design

Well-crafted instructions are the foundation of effective AI agents. They define how your agent understands its role, responds to users, and handles various scenarios.

Best practices for instructions

When writing system instructions for your agent:

- **Be specific and clear** - Define exactly what the agent should do and how it should behave

- **Provide context** - Explain the agent's role and the environment it operates in
- **Set boundaries** - Clearly define what the agent should and shouldn't do
- **Include examples** - Show the agent examples of desired interactions when helpful
- **Define personality** - Establish the tone and style of responses

Instruction examples

For a customer service agent, effective instructions might include:

- The agent's role and purpose
- Guidelines for handling different types of customer inquiries
- Escalation procedures for complex issues
- Tone and communication style preferences

Deploying agents

Once you configure your agent, you can deploy it to Microsoft Foundry.

Deployment process

To deploy your agent:

1. **Select the "Create on Microsoft Foundry" button** in the bottom-left of the Designer
2. **Wait for deployment completion** - The extension handles the deployment process
3. **Refresh the Azure Resources view** in the Visual Studio Code navbar
4. **Verify deployment** - The deployed agent appears under the Agents subsection

Managing deployed agents

After deployment, you can:

- **View agent details** - Select the deployed agent to see the Agent Preferences page
- **Edit the agent** - Select "Edit Agent" to modify configuration and redeploy with the **Update on Microsoft Foundry** button
- **Generate integration code** - Select "Open Code File" to create sample code for using the agent
- **Test in playground** - Select "Open Playground" to interact with the deployed agent

Testing and iteration

The integrated playground enables immediate testing of your agent configuration, allowing you to validate behavior and make adjustments in real-time.

Using the playground

After configuring your agent, you can test it using the built-in playground:

- **Real-time conversations** - Chat with your agent to test responses
- **Instruction validation** - Verify the agent follows its configured instructions
- **Behavior testing** - Test how the agent handles different types of requests
- **Iterative refinement** - Make adjustments based on testing results

Working with agent threads

When you interact with deployed agents, the system creates threads to manage conversation sessions:

- **Threads** - Conversation sessions between an agent and user that store messages and handle context management
- **Messages** - Individual interactions that can include text, images, and files
- **Runs** - Single executions of an agent that use the agent's configuration and thread messages

You can view and manage these threads through the Azure Resources view in the extension.

Creating and configuring AI agents with the Microsoft Foundry Visual Studio Code extension provides a powerful yet accessible approach to agent development. The extension provides visual design tools, direct YAML editing, comprehensive configuration options, and integrated testing capabilities. These features enable developers to rapidly prototype and deploy sophisticated AI agents that can handle complex real-world scenarios.

4. Extend AI agent capabilities with tools

https://learn.microsoft.com/en-us/training/modules/develop-ai-agents-vs-code/4-extend-agent-capabilities

Extend AI agent capabilities with tools

Completed

- 5 minutes

One of the most powerful features of AI agents is their ability to use tools that extend their capabilities beyond simple text generation. The Microsoft Foundry for Visual Studio Code extension makes it easy to add and configure tools for your agents. These tools enable agents to perform actions, access data, and integrate with external systems.

Understanding agent tools

Tools are programmatic functions that enable agents to automate actions and access information beyond their training data. When an agent determines that a tool is needed to respond to a user request, it can automatically invoke the appropriate tool, process the results, and incorporate them into its response. This capability transforms agents from simple text generators into powerful automation systems that can interact with real-world data and services.

Built-in tools

Microsoft Foundry provides several built-in tools that you can easily add to your agents without any additional configuration or setup. These tools are production-ready and handle common use cases that many agents require.

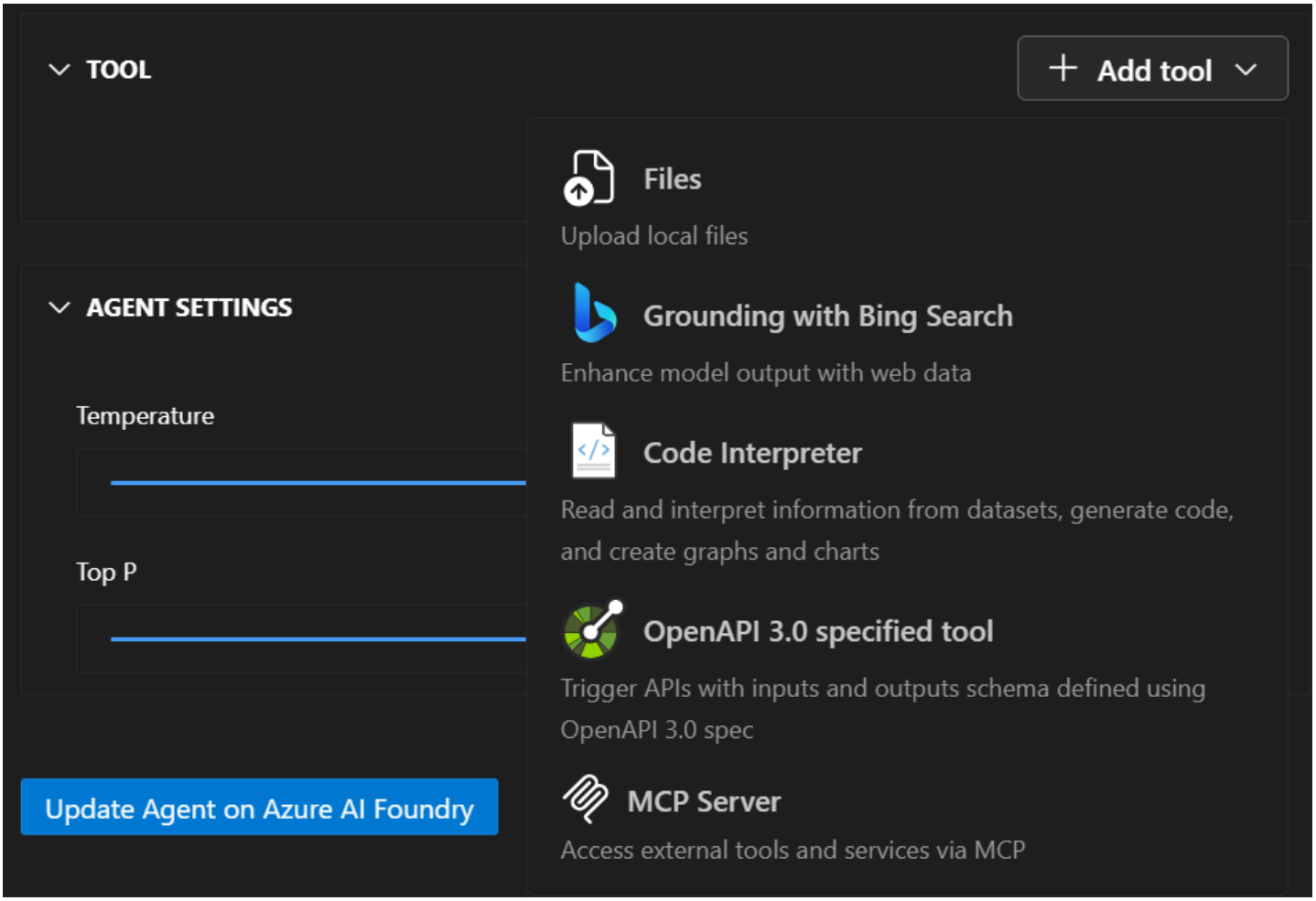
- **Code Interpreter** - Enables agents to write and execute Python code for mathematical calculations, data analysis, chart generation, file processing, and complex problem-solving
- **File Search** - Provides retrieval-augmented generation by uploading and indexing documents, searching knowledge bases, and supporting various file formats (PDF, Word, text files)
- **Grounding with Bing Search** - Allows agents to search the internet for real-time data, current events, and trending topics while providing citations and sources
- **OpenAPI Specified Tools** - Connects agents to external APIs and services through OpenAPI 3.0 specifications
- **Model Context Protocol (MCP)** - Standardized tool interfaces for extended functionality and community-driven tools

Adding tools in Visual Studio Code

The Microsoft Foundry extension provides an intuitive interface for adding tools to your agents through a streamlined process. The visual interface makes it easy to browse, configure, and test tools without writing any code:

1. **Select your agent** in the extension
2. **Navigate to the Tools section** in the configuration panel
3. **Browse available tools** from the tool library

4. **Configure tool settings** as needed
5. **Test tool integration** using the playground



When you add a tool, you can also add any new assets it needs. For example, if you add a File Search tool, you can use an existing vector store asset or make a new asset for your vector store to host your uploaded files.

Model Context Protocol (MCP) servers

MCP servers provide a standardized way to add tools to your agents using an open protocol. This approach enables you to use community-built tools and create reusable components that work across different agent implementations.

Key benefits include:

- **Standardized protocol** for consistent tool communication
- **Reusable components** that work across different agents
- **Community-driven tools** available through MCP registries
- **Simplified integration** with consistent interfaces

The extension supports adding MCP servers through registry browsing, custom server addition, configuration management, and testing and validation.

Tool management and best practices

Effective tool management ensures your agents perform reliably and efficiently in production environments. Following best practices helps you avoid common pitfalls and optimize agent performance:

Tool Selection Guidelines

- Identify what capabilities your agent requires
- Start with built-in tools before adding custom solutions
- Test thoroughly to validate tool behavior in various scenarios
- Monitor performance to track tool usage and effectiveness

Adding tools and extending agent capabilities through the Microsoft Foundry Visual Studio Code extension enables you to create sophisticated AI agents that can handle complex real-world tasks. By combining built-in tools with custom functions and MCP servers, you can build agents that seamlessly integrate with your existing systems and business processes while maintaining enterprise-grade security and performance.

5. Exercise - Build an AI agent using the Microsoft Foundry extension

https://learn.microsoft.com/en-us/training/modules/develop-ai-agents-vs-code/5-exercise

Exercise - Build an AI agent using the Microsoft Foundry extension

Completed

- 30 minutes

If you have an Azure subscription, you can explore the Microsoft Foundry Extension for Visual Studio Code by completing this hands-on exercise. This exercise guides you through creating and deploying an AI agent using the extension.

> **Note**

> If you don't have an Azure subscription, and you want to explore Microsoft Foundry, you can sign up for an account, which includes credits for the first 30 days.

Launch the exercise and follow the instructions.

Launch Exercise

6. Module assessment

https://learn.microsoft.com/en-us/training/modules/develop-ai-agents-vs-code/6-knowledge-check

Module assessment

Completed

- 3 minutes

7. Summary

https://learn.microsoft.com/en-us/training/modules/develop-ai-agents-vs-code/7-summary

Summary

Completed

- 1 minute

In this module, you learned how to develop AI agents using the Microsoft Foundry for Visual Studio Code extension. You explored how to create, configure, and test agents directly within your development environment. You also learned to extend their capabilities with various tools and deploy them to production environments.

Tip

For more information, see **Work with the Microsoft Foundry for Visual Studio Code extension**.