Develop a multi-agent solution

In this exercise, you'll practice using the sequential orchestration pattern in the Microsoft Agent Framework SDK. You'll create a simple pipeline of three agents that work together to process customer feedback and suggest next steps. You'll create the following agents:

- The Summarizer agent will condense raw feedback into a short, neutral sentence.
- The Classifier agent will categorize the feedback as Positive, Negative, or a Feature request.
- Finally, the Recommended Action agent will recommend an appropriate follow-up step.

You'll learn how to use the Microsoft Agent Framework SDK to break down a problem, route it through the right agents, and produce actionable results. Let's get started!

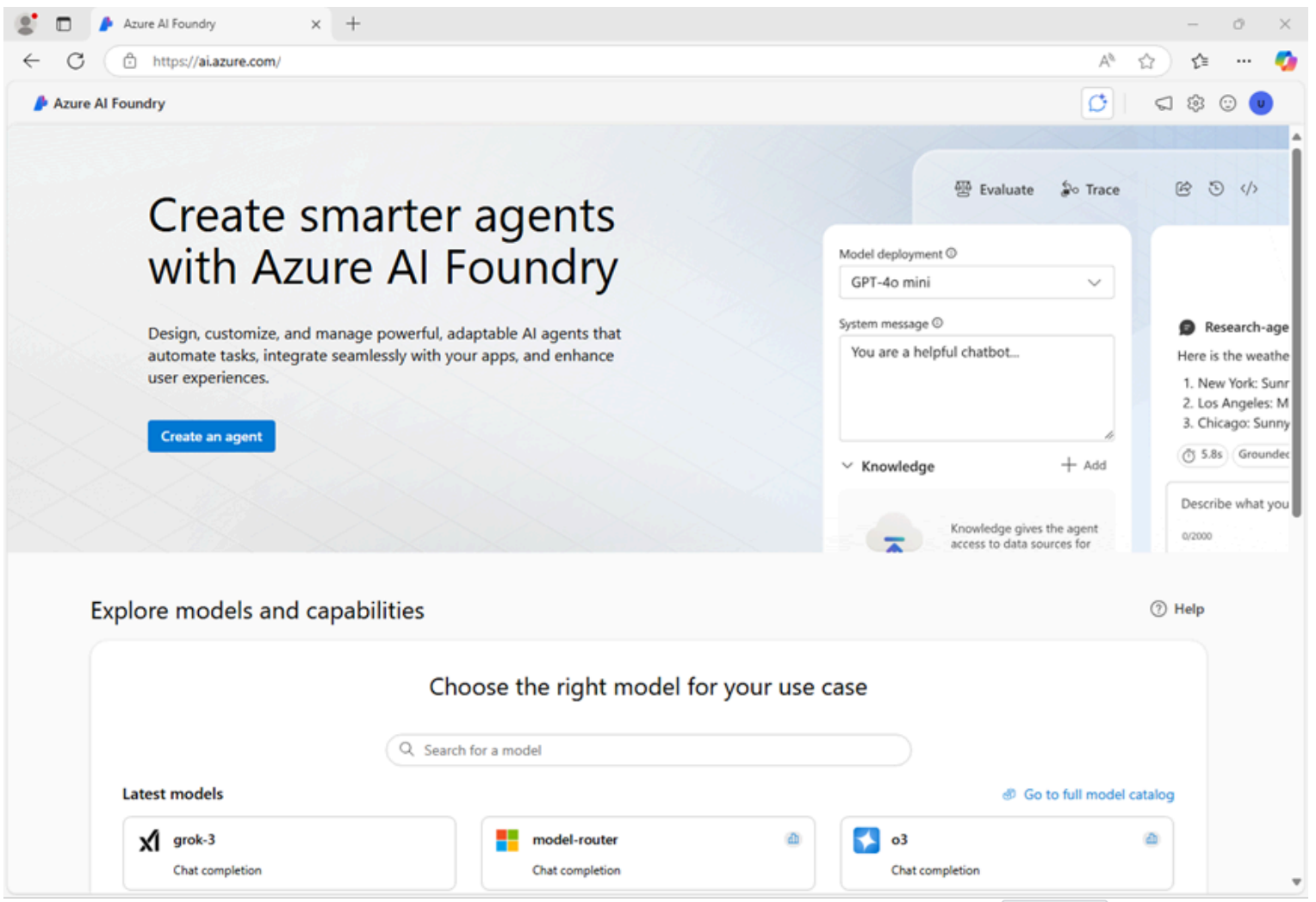
This exercise should take approximately **30** minutes to complete.

> **Note**: Some of the technologies used in this exercise are in preview or in active development. You may experience some unexpected behavior, warnings, or errors.

Deploy a model in an Azure AI Foundry project

Let's start by deploying a model in an Azure AI Foundry project.

1. In a web browser, open the Azure AI Foundry portal at `https://ai.azure.com` and sign in using your Azure credentials. Close any tips or quick start panes that are opened the first time you sign in, and if necessary use the **Azure AI Foundry** logo at the top left to navigate to the home page, which looks similar to the following image (close the **Help** pane if it's open):

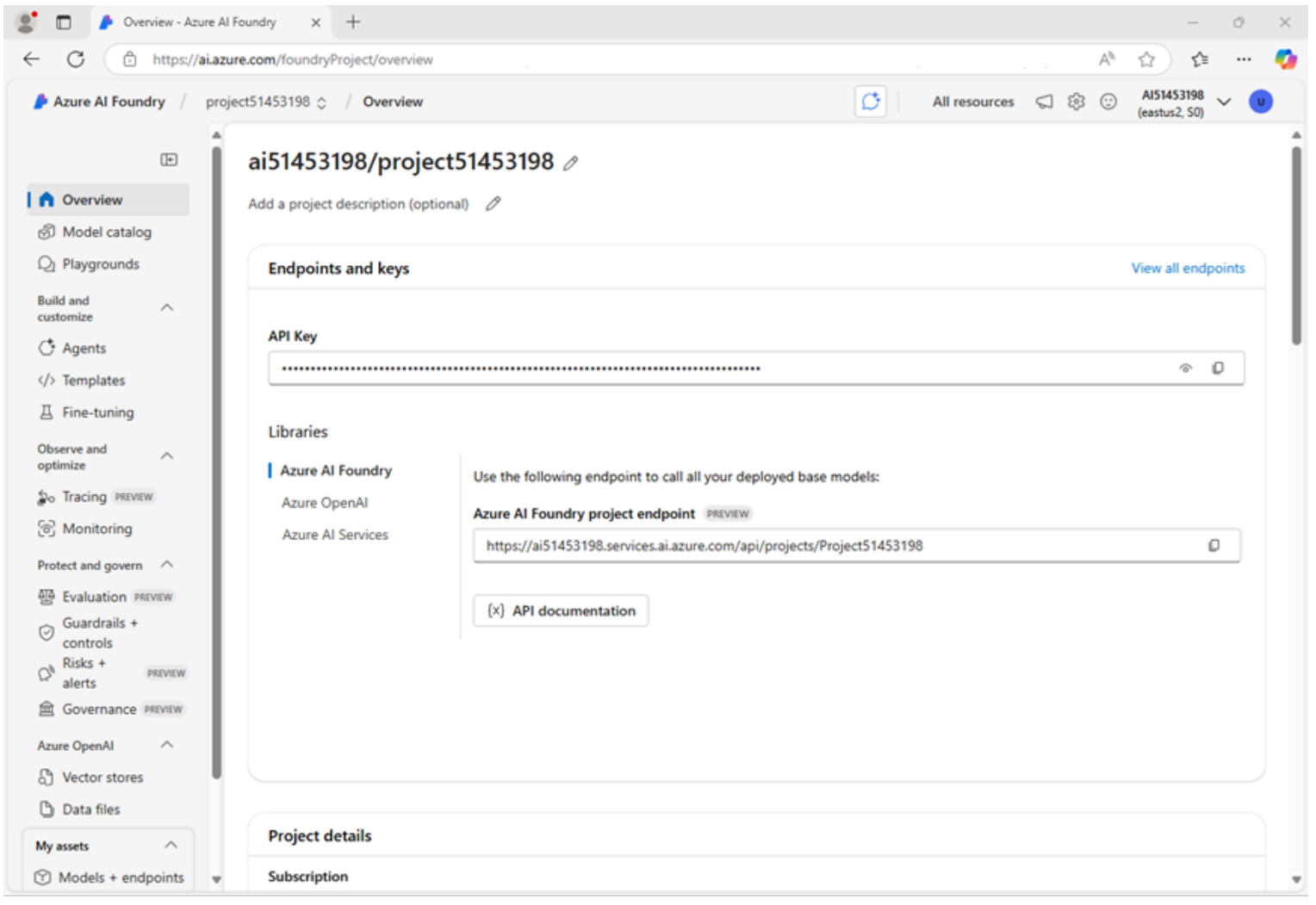


2. In the home page, in the **Explore models and capabilities** section, search for the `gpt-4o` model; which we'll use in our project.
3. In the search results, select the **gpt-4o** model to see its details, and then at the top of the page for the model, select **Use this model**.
4. When prompted to create a project, enter a valid name for your project and expand **Advanced options**.
5. Confirm the following settings for your project:
    - **Azure AI Foundry resource**: *A valid name for your Azure AI Foundry resource*
    - **Subscription**: *Your Azure subscription*
    - **Resource group**: *Create or select a resource group*

- **Region**: *Select any **AI Foundry recommended****

> ! * Some Azure AI resources are constrained by regional model quotas. In the event of a quota limit being exceeded later in the exercise, there's a possibility you may need to create another resource in a different region.

6. Select **Create** and wait for your project, including the gpt-4 model deployment you selected, to be created.
7. When your project is created, the chat playground will be opened automatically.
8. In the navigation pane on the left, select **Models and endpoints** and select your **gpt-4o** deployment.
9. In the **Setup** pane, note the name of your model deployment; which should be **gpt-4o**. You can confirm this by viewing the deployment in the **Models and endpoints** page (just open that page in the navigation pane on the left).
10. In the navigation pane on the left, select **Overview** to see the main page for your project; which looks like this:



Create an AI Agent client app

Now you're ready to create a client app that defines an agent and a custom function. Some code is provided for you in a GitHub repository.

Prepare the environment

1. Open a new browser tab (keeping the Azure AI Foundry portal open in the existing tab). Then in the new tab, browse to the Azure portal at `https://portal.azure.com`; signing in with your Azure credentials if prompted.

   Close any welcome notifications to see the Azure portal home page.

2. Use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment with no storage in your subscription.

   The cloud shell provides a command-line interface in a pane at the bottom of the Azure portal. You can resize or maximize this pane to make it easier to work in.

> ! **Note**: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.

3. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

   Ensure you've switched to the classic version of the cloud shell before continuing.

4. In the cloud shell pane, enter the following commands to clone the GitHub repo containing the code files for this exercise (type the command, or copy it to the clipboard and then right-click in the command line and paste as plain text):

   Code Copy

   ```
   rm -r ai-agents -f
   git clone https://github.com/MicrosoftLearning/mslearn-ai-agents ai-agents
   ```

   > **Tip**: As you enter commands into the cloud shell, the output may take up a large amount of the screen buffer and the cursor on the current line may be obscured. You can clear the screen by entering the `cls` command to make it easier to focus on each task.

5. When the repo has been cloned, enter the following command to change the working directory to the folder containing the code files and list them all.

   Code Copy

   ```
   cd ai-agents/Labfiles/05-agent-orchestration/Python
   ls -a -l
   ```

   The provided files include application code and a file for configuration settings.

Configure the application settings

1. In the cloud shell command-line pane, enter the following command to install the libraries you'll use:

   Code Copy

   ```
   python -m venv labenv
   ./labenv/bin/Activate.ps1
   pip install azure-identity agent-framework
   ```

2. Enter the following command to edit the configuration file that is provided:

   Code Copy

   ```
   code .env
   ```

   The file is opened in a code editor.

3. In the code file, replace the **your_openai_endpoint** placeholder with the endpoint for your project (copied from the project **Overview** page in the Azure AI Foundry portal). Replace the **your_model_deployment** placeholder with the name you assigned to your gpt-4o model deployment.

4. After you've replaced the placeholders, use the **CTRL+S** command to save your changes and then use the **CTRL+Q** command to close the code editor while keeping the cloud shell command line open.

Create AI agents

Now you're ready to create the agents for your multi-agent solution! Let's get started!

1. Enter the following command to edit the **agents.py** file:

Code Copy

```
code agents.py
```

2. At the top of the file under the comment **Add references**, and add the following code to reference the namespaces in the libraries you'll need to implement your agent:

Code Copy

```
# Add references
import asyncio
from typing import cast
from agent_framework import ChatMessage, Role, SequentialBuilder, WorkflowOutputEvent
from agent_framework.azure import AzureAIAgentClient
from azure.identity import AzureCliCredential
```

3. In the **main** function, take a moment to review the agent instructions. These instructions define the behavior of each agent in the orchestration.

4. Add the following code under the comment **Create the chat client**:

Code Copy

```
# Create the chat client
credential = AzureCliCredential()
async with (
    AzureAIAgentClient(async_credential=credential) as chat_client,
):
```

Note that the **AzureCliCredential** object will allow your code to authenticate to your Azure account. The **AzureAIAgentClient** object will automatically include the Azure AI Foundry project settings from the .env configuration.

5. Add the following code under the comment **Create agents**:

(Be sure to maintain the indentation level)

Code Copy

```
# Create agents
summarizer = chat_client.create_agent(
    instructions=summarizer_instructions,
    name="summarizer",
)

classifier = chat_client.create_agent(
    instructions=classifier_instructions,
    name="classifier",
)

action = chat_client.create_agent(
    instructions=action_instructions,
    name="action",
)
```

Create a sequential orchestration

1. In the **main** function, find the comment **Initialize the current feedback** and add the following code:

   (Be sure to maintain the indentation level)

   Code Copy

   ```
   # Initialize the current feedback
   feedback="""
   I use the dashboard every day to monitor metrics, and it works well overall.
   But when I'm working late at night, the bright screen is really harsh on my eyes.
   If you added a dark mode option, it would make the experience much more comfortable.
   """
   ```

2. Under the comment **Build a sequential orchestration**, add the following code to define a sequential orchestration with the agents you defined:

   Code Copy

   ```
   # Build sequential orchestration
   workflow = SequentialBuilder().participants([summarizer, classifier, action]).build()
   ```

   The agents will process the feedback in the order they are added to the orchestration.

3. Add the following code under the comment **Run and collect outputs**:

   Code Copy

   ```
   # Run and collect outputs
   outputs: list[list[ChatMessage]] = []
   async for event in workflow.run_stream(f"Customer feedback: {feedback}"):
       if isinstance(event, WorkflowOutputEvent):
           outputs.append(cast(list[ChatMessage], event.data))
   ```

   This code runs the orchestration and collects the output from each of the participating agents.

4. Add the following code under the comment **Display outputs**:

   Code Copy

   ```
   # Display outputs
   if outputs:
       for i, msg in enumerate(outputs[-1], start=1):
           name = msg.author_name or ("assistant" if msg.role == Role.ASSISTANT else "user")
           print(f"{'-' * 60}\n{i:02d} [{name}]\n{msg.text}")
   ```

   This code formats and displays the messages from the workflow outputs you collected from the orchestration.

5. Use the **CTRL+S** command to save your changes to the code file. You can keep it open (in case you need to edit the code to fix any errors) or use the **CTRL+Q** command to close the code editor while keeping the cloud shell command line open.

Sign into Azure and run the app

Now you're ready to run your code and watch your AI agents collaborate.

1. In the cloud shell command-line pane, enter the following command to sign into Azure.

Code Copy

```
az login
```

You must sign into Azure - even though the cloud shell session is already authenticated.

> **Note**: In most scenarios, just using *az login* will be sufficient. However, if you have subscriptions in multiple tenants, you may need to specify the tenant by using the *–tenant* parameter. See Sign into Azure interactively using the Azure CLI for details.

2. When prompted, follow the instructions to open the sign-in page in a new tab and enter the authentication code provided and your Azure credentials. Then complete the sign in process in the command line, selecting the subscription containing your Azure AI Foundry hub if prompted.

3. After you have signed in, enter the following command to run the application:

Code Copy

```
python agents.py
```

You should see some output similar to the following:

Code Copy

```
---------------------------------------------------------
01 [user]
Customer feedback:
    I use the dashboard every day to monitor metrics, and it works well overall.
    But when I'm working late at night, the bright screen is really harsh on my eyes.
    If you added a dark mode option, it would make the experience much more comfortable.

---------------------------------------------------------
02 [summarizer]
User requests a dark mode for better nighttime usability.
---------------------------------------------------------
03 [classifier]
Feature request
---------------------------------------------------------
04 [action]
  Log as enhancement request for product backlog.
```

4. Optionally, you can try running the code using different feedback inputs, such as:

Code Copy

```
 I use the dashboard every day to monitor metrics, and it works well overall. But when I'm working late at night, the bright screen is really harsh on my eyes. If you added a dark mode option, it would make the experience much more comfortable.
```

Code Copy

```
 I reached out to your customer support yesterday because I couldn't access my account. The representative responded almost immediately, was polite and professional, and fixed the issue within minutes. Honestly, it was one of the best support experiences I've ever had.
```

Summary

In this exercise, you practiced sequential orchestration with the Microsoft Agent Framework SDK, combining multiple agents into a single, streamlined workflow. Great work!

Clean up

If you've finished exploring Azure AI Agent Service, you should delete the resources you have created in this exercise to avoid incurring unnecessary Azure costs.

1. Return to the browser tab containing the Azure portal (or re-open the Azure portal at `https://portal.azure.com` in a new browser tab) and view the contents of the resource group where you deployed the resources used in this exercise.
2. On the toolbar, select **Delete resource group**.
3. Enter the resource group name and confirm that you want to delete it.