**LAB-A**01 **Create your first autonomous agent**

**Create an intelligent, self-directed agent to ensure that new hires at Contoso are trained and equipped to start their new roles.**

# Lab Details – Lab in review (pls do not publish yet). -Audrie

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Persona | Purpose | Estimated time to complete |
| 100 | Basic Maker | This lab will walk you through the planning and development of an Autonomous Agent. We will first understand the value of Autonomous over Process Flows and then we will build an employee onboarding agent. | 1.5 Hours |

## Prerequisites

You need to be able to create a custom agent in <https://copilotstudio.microsoft.com/>.

To publish your agent to your website or to a demo website, the following [data loss prevention policies](https://learn.microsoft.com/en-us/microsoft-copilot-studio/admin-data-loss-prevention) should NOT be blocked on your environment:

* Chat without Microsoft Entra ID authentication in Copilot Studio,
* Direct Line channels in Copilot Studio,
* Knowledge source with public websites and data in Copilot Studio.

## Summary of targets

We will begin by discussing scenarios most appropriate for Microsoft Copilot Studio Autonomous Agents. Then we will explore how to leverage Knowledge, Triggers, and Actions to build out a strong self-directed agent. We will close the lab with testing tips to help with Quality Assurance and confidence building in self-directed agents. At the close of this section is a list of documentation and resources that we recommend to complete your understanding and empower more complex scenarios.

| Use case/topic | Tagline | Page |
| --- | --- | --- |
| Concept Discussion:  Why Autonomous ? | Wondering Why?—Cloud based workflow, such as flows created with Power Automate, have proven their value for process design and implementation. So, why autonomous? | 3 |
| Building on a strong foundation | The foundation for healthy agent lifecycle management—Group your agent, knowledge sources, and plugins in a solution to simplify customization, deployment, and long-term maintenance. | 5 |
| Give your agent purpose | From idle to aware.—Agents are built to know when to act through well-planned triggers. | 8 |
| Ground your agent in relevant knowledge sources | From aware to insightful—Equipping agents to respond with relevance and purpose by grounding them with knowledge sources that matter. | 12 |
| Empower your agent with Actions | Autonomous agents leverage actions to get stuff done!—Actions are the wind beneath your agent’s wings. | 16 |
| Agent Instructions | Guide agents with instructions!—The control of what is done, and in what order, fortunately remains in the hands of the Maker using natural language to instruct the maker. | 29 |
| Quality Assurance | Testing and Monitoring for Success!—Understanding how Activity Monitoring and Testing can ensure high quality performance. | 33 |
| Summary of learnings | Mastery is not a destination but a journey—a joyful path where every step brings growth, discovery, and endless possibilities. | 37 |
| Glossary | Speak the language, bridge the world—unlock hearts, opportunities, and the true essence of every land. | 38 |

## Documentation and additional training links

* [Unlocking autonomous agent capabilities with Microsoft Copilot Studio | Microsoft Copilot Blog](https://www.microsoft.com/en-us/microsoft-copilot/blog/copilot-studio/unlocking-autonomous-agent-capabilities-with-microsoft-copilot-studio/?msockid=11c6ce10524069a40498df5053f36869)
* [Build an Autonomous Agent in Copilot Studio - Training | Microsoft Learn](https://learn.microsoft.com/en-us/training/modules/autonomous-agent/)
* [Build autonomous capabilities for agents in Copilot Studio](https://build.microsoft.com/en-US/sessions/LAB327)

# Use Case #1: Why Autonomous?

*Wondering Why? – Cloud based workflow, such as flows created with Power Automate, have proven their value for process design and implementation. So, why choose autonomous agents?!*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Understand why | Enable businesses to select between process creation tools such as Power Automate flows and Microsoft Copilot Studio Autonomous Agents to optimize automations. | 5 minutes |

## Summary of tasks

Let’s step back and understand the reasons one would select to build an Autonomous Agent over some other type of workflow or process design tool.

**Scenario**: Lay the groundwork for building a smart AI assistant by understanding the best use cases for this solution type.

## Why Autonomous Agent approach rather than Power Automate[[1]](#footnote-2) Cloud Flow?

Power Automate is great for structured, predictable processes — like setting up a row of dominoes and watching them fall in order. But not all processes are that static.

Autonomous Agents are like trained assistants who don’t just follow steps — they watch what’s happening, make decisions, and **adjust to the course in real-time**. **They're built for dynamic, decision-rich scenarios where the next step depends on changing data, not predefined logic**. Think of it this way:

Left side: GPS screen navigating to a single location.

Right side: A person driving a car and adjusting GPS based on what's happening on the road.  


Figure 1 : AI Generated Image (Dall-e)

* Power Automate is like a GPS that follows a set route.
* An Autonomous Agent is like a smart driver who notices roadblocks, traffic, or detours — and reroutes on the fly without waiting for your instructions.

## Test your understanding

Now that you’ve got a better understanding of the value of Autonomous Agents, can you select from the following options which is the better approach and explain why? (Don’t peek, but answers will be found after the glossary at the end of this file 😉.)

Quiz #1: Which approach would you pick for these scenarios?   
**[Cloud flow or Autonomous Agent]**

Scenario A: Your team processes dozens of expense reports every week. Each report follows a straightforward path: submit, approve, or reject, with minimal variations.

Scenario B: A customer service team needs help managing support tickets. Some tickets are simple and can be routed easily, but others require analyzing multiple data sources to classify the issue and determine the next steps in real-time.

Scenario C: Your finance team reviews loan applications that must comply with evolving regulations. These applications pull data from multiple sources, and compliance rules can change frequently. The process requires analyzing various factors before deciding.

**Key take aways:**

* Look out for dynamic, **decision-rich scenarios** where the next step depends on changing data, not predefined logic.
* Remember these are also **better-together scenarios**, where Autonomous Agents leverage the power of Cloud flows plus Generative AI.
* Side note: In the case of Autonomous Agents **there is no conversation panel**. So, you’ll have to be a bit creative when it comes to humans communicating with the agent.

**Challenge: apply this to your own use case**

* What scenarios can you imagine at your company which could use an Autonomous agent and why?
* Are there any existing Power Automate flows that might be better designed as Autonomous Agents? Clarify what you think an agent could add to the experience for users.

Take it further: Pencil sketch at least 3 legacy business scenarios in your org, that could be modernized with Autonomous Agents (to save in labor efforts or to save money). Is this something you could hackathon this week?

|  |
| --- |
|  |
|  |
|  |
|  |

# Use Case #2: Building on a strong foundation

*The foundation for healthy lifecycle management – Group your agent, knowledge sources, and plugins in a solution to simplify customization, deployment, and long-term maintenance.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Create a solution | Set up a structured workspace that supports application lifecycle management and reuse. | 5 minutes |

## Summary of tasks

In this section, you’ll create a new solution—your container for managing the agent, knowledge sources, and connectors together.

**Scenario**: Lay the groundwork for building a smart AI assistant by creating a solution that will make it easier to manage and deploy your components as your assistant evolves.  
  
Pro tip: All input fields in Copilot Studio have character count limitations. You can find these limitations on the lower right of each text input area.

## Step-by-step instructions

1. Navigate to the Copilot Studio **home** **page**.

<https://copilotstudio.microsoft.com/>

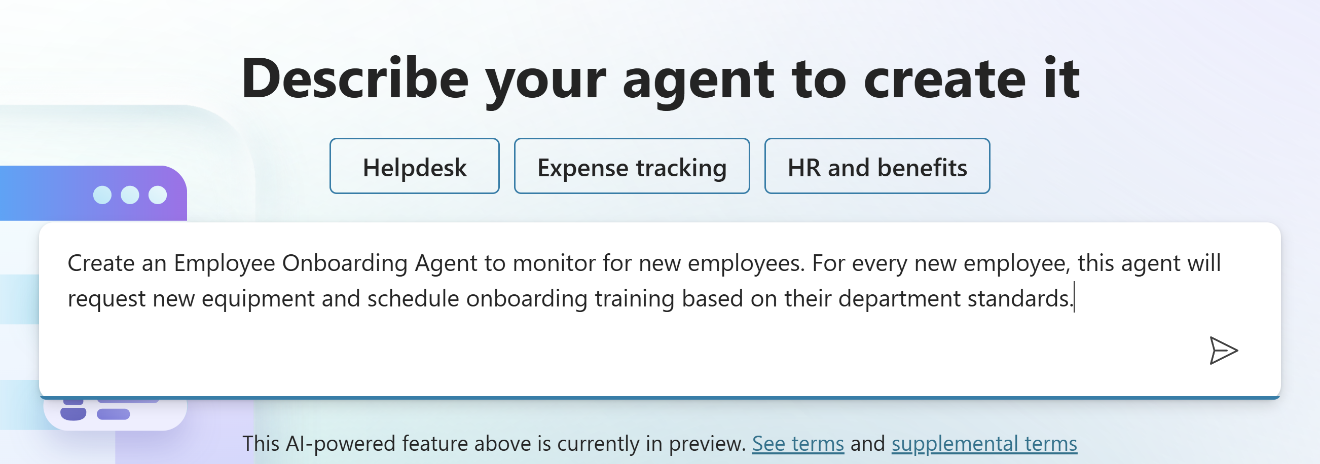
1. Go to the **Solutions** menu (located in the left-hand menu under the ellipsis …) and select **New solution**.

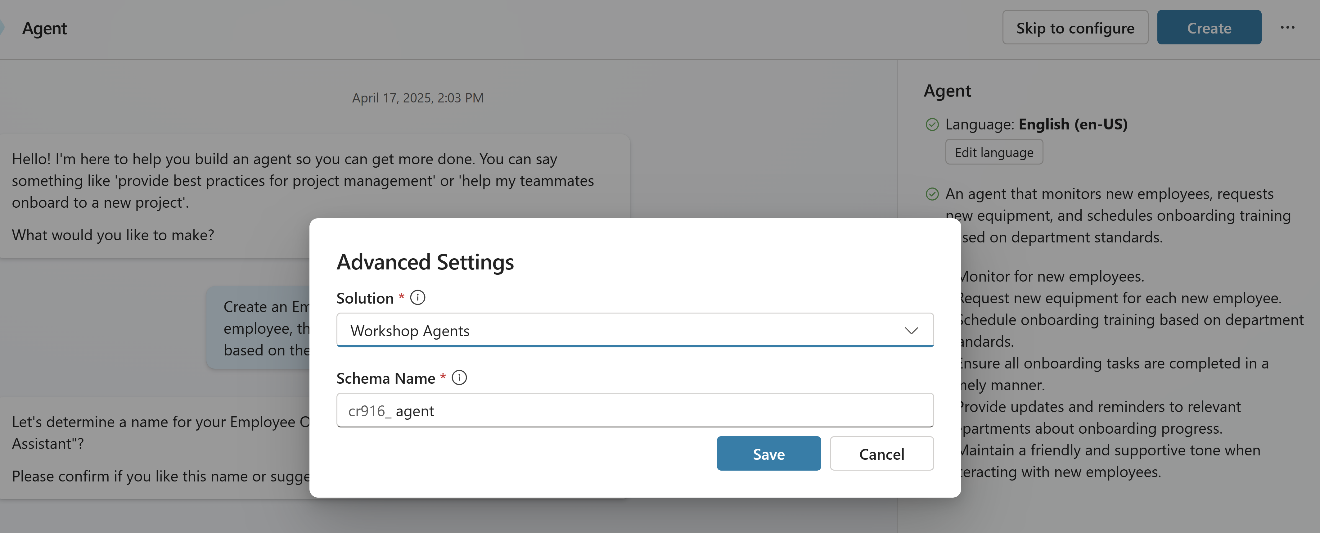
Provide a display name – this will persist across environment deployments, so avoid names tied to a specific environment (e.g., 'DEV') or development stage (e.g., 'POC'). Instead, choose a name that reflects the contents of your solution package, such as your agent or project name. For this lab, let’s name it **Workshop Agents**, selecting the default **Publisher** for this lab.

1. Select **Save** when ready.
2. We will immediately create a quick starting point for our Autonomous Agent next. Navigate back to **copilotstudio.microsoft.com** checking to ensure you are in the **same environment with the Solution** you just created.
3. In the top area **Describe your agent to create it**, enter this text and then click the **Send** icon.

Create an Employee Onboarding Agent to monitor for new employees. For every new employee this action will request new equipment and schedule onboarding training based on their department standards.

1. On the next screen, in the upper righthand corner click the **ellipsis** **…** and select **Edit Advanced Settings**, this will enable you to **select the Solution** you created previously. After selecting the Solution from the dropdown, click on Save. This will bring you back one screen.
2. In the **chat area** (lower left), type the following and then click the **send icon** to set the name of the agent:  
   Name the agent Employee Onboarding Agent
3. We’ll skip the rest by clicking **Create** in the top right corner.Then Copilot Studio will build the Agent for us. We can change anything related to this configuration later (except the Solution selection). This is a fast way to get a starting point for our agent!




## Test your understanding

Now that you’ve created a solution in Copilot Studio, and kicked off your agent build, take a moment to reflect on what you’ve learned.

**Key takeaways:**

* Solutions first – A solution provides a structured container to manage your agent, connectors, and future customizations across environments.
* Lifecycle readiness – Creating a solution upfront enables better governance, easier updates, and smoother deployment.
* Naming conventions matter – Use neutral, environment-independent names to support clean ALM practices.

**Lessons learned & troubleshooting tips:**

* Avoid names like “DEV” or “Test” in your solution display name—they can cause confusion during deployments.
* If your solution fails to save, make sure the publisher prefix is unique and compliant with schema rules.
* Keep your solution name focused on the business scenario or agent purpose, not the technical phase.

**Challenge: apply this to your own use case**

* What name would you give your solution to reflect your scenario (e.g., research assistant, internal knowledge agent)?
* How might you use the solution container to organize future components, like agent flows Dataverse tables?

Take it further: Try creating another solution for a different department or use case, and explore how solutions help you manage parallel agents with clean separation and reuse.

|  |
| --- |
|  |
|  |
|  |
|  |

# Use Case #3: Give your agent purpose

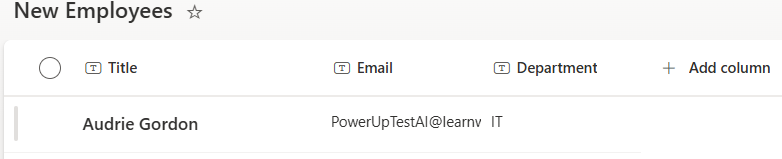
*From idle to aware – Agents are built to know when to act through well-planned triggers.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Activating your  agent | Add trigger(s) so that your agent is activated at the appropriate time. Agents will wait for these to ‘trigger’ before beginning any activities. | 10 minutes |

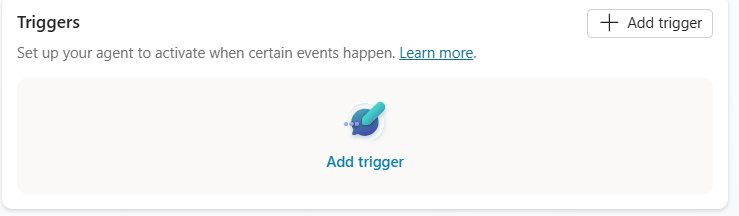
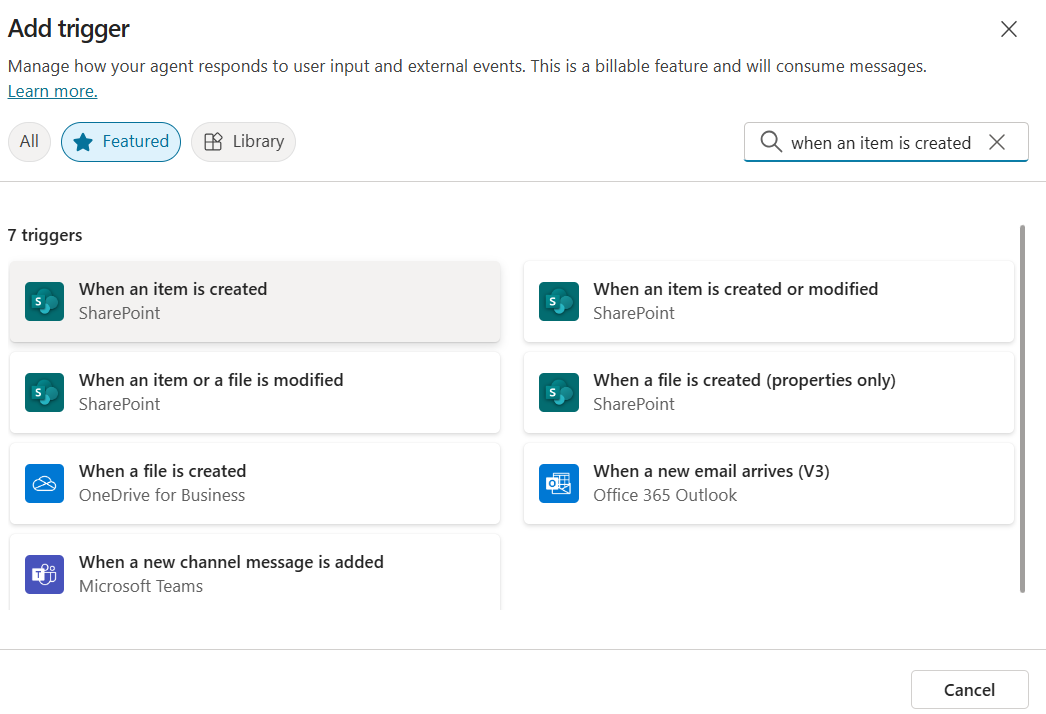
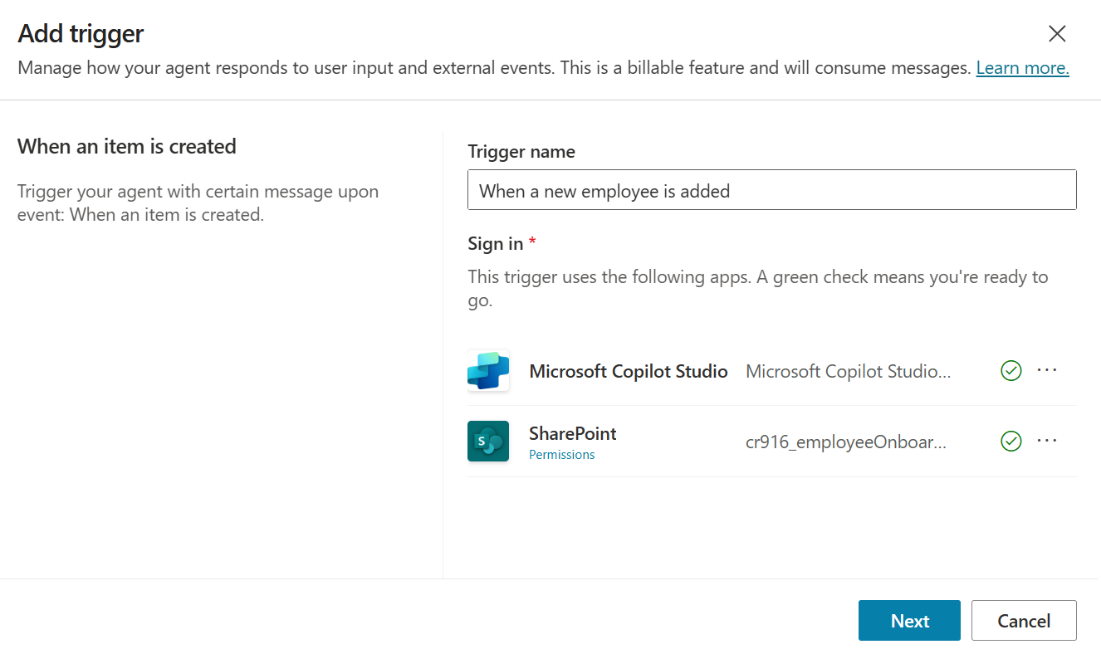
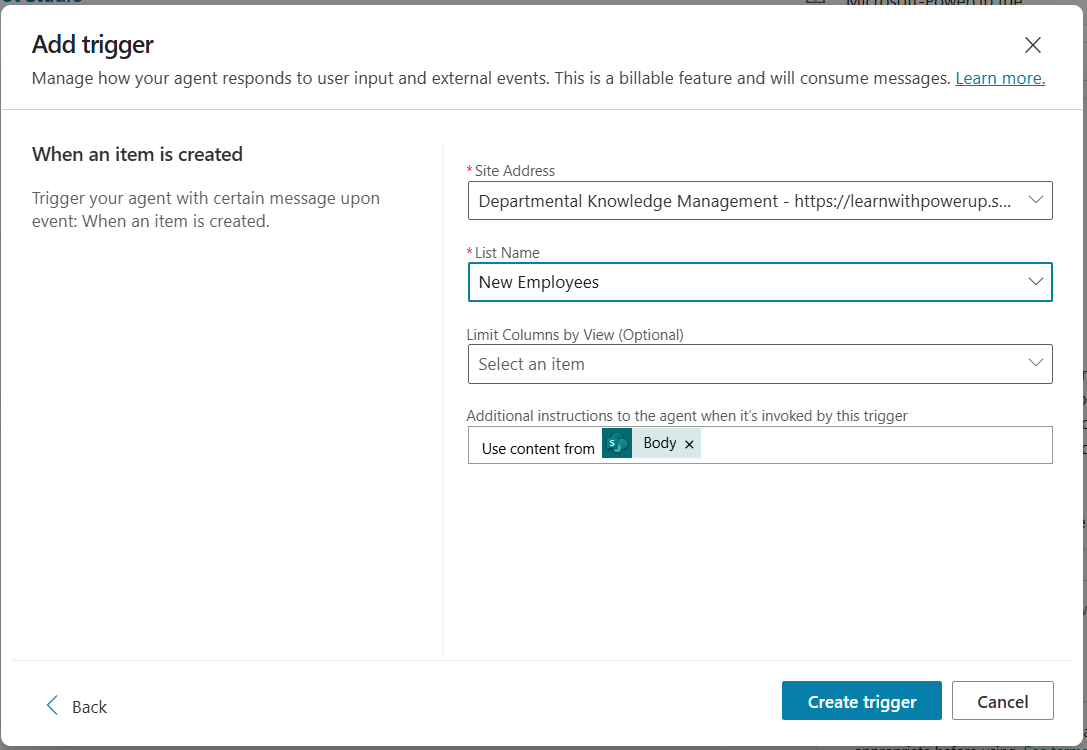
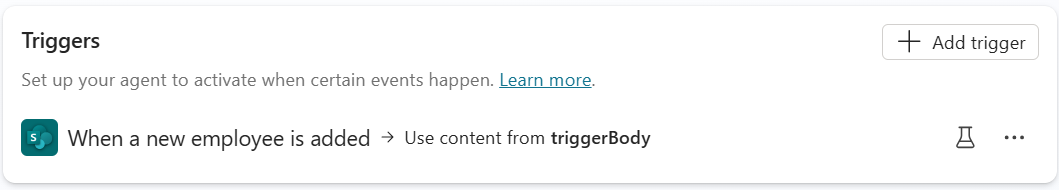
## Summary of tasks

Before an Autonomous Agent can do anything, we need to define when and how it should activate. We do this using **Triggers**. Triggers are accessed using connectors which tap into the events that might trigger the agent. There are over 1300 pre-built connectors available, but you can always create your own custom connector as well. Execution of these triggers occurs with the help of Power Automate. In each case a Power Automate flow is generated for you in support of the trigger.

Side note: There is also an **Activity tab** which has the purpose of allowing the agent maker to track each historical interaction with their agent. Any time a topic or action gets triggered this will be recorded as a conversation within the activity tab. This will enable the maker to drill into the interaction and review each action that was called and why the action was called. Note as of April 2025, only interactions with the maker will be recorded on the activity tab. We will have time to work with activities later on in this lab.

**Scenario**: Our SharePoint site contains a list called **New Employees**. This is where we put the name and email address of all new hires. Anytime a new hire is added to this list, we want our agent to get busy working! This will be our trigger. Please note that an agent can have multiple triggers, but this is the only one we need for our Employee Onboarding Agent. Each trigger creates a Power Automate flow in the back-end to poll for events.  
  


## Configuring the Trigger

1. Navigate back to our Employee Onboarding Agent and select the Overview Tab. Scroll down to the Triggers section and click + **Add trigger** at the top right. Once you’ve done that, you will see a view of featured triggers. Note: You may have to turn on Generative Orchestration if it is not already turned on, if you are prompted to do that just go ahead and click **Turn it on**. Learn more about Generative Orchestration [here](https://learn.microsoft.com/en-us/microsoft-copilot-studio/faqs-generative-orchestration).   
     
   
2. Using the search box at the top right of this dialog box, search for **When an item is created**. Look at the results and select the connector **When an item is created** which is associated with SharePoint and select that one. You will be prompted to create a connection, if it is not already created, and then to configure the specific site and list information. Rename the trigger **When a new employee is added** then select the site (Departmental Knowledge Management) and list (New Employees) from the dropdowns, leave everything else the same and then click **Create Trigger**.   
     
   You may also **Close** the dialog encouraging you to **Test your agent**’s trigger, as we will be doing that later on in this lab.  
     
     
     
     
     
     
   Validate that you have a new **Trigger** on the **Overview tab**.   
     
   

## Test your understanding

Now that you’ve create a trigger, you have a better understanding of what makes an Autonomous Worker start work. Unlike other agents, no interaction in the conversation panel is required, and no end user has to solicit it. The Autonomous Agent only waits for a trigger and off it goes to get things done! In our case, it’s looking for a new item to be added to the SharePoint list called New Employees.

**Key takeaways:**

* Triggers are the way we get autonomous agents to activate.
* Triggers use connectors, with over 1300 pre-built connectors available for you to use.
* In this scenario, we only needed one trigger, but it is possible to have more than one trigger for an Autonomous or Hybrid agent.

**Challenge: apply this to your own use case**

* As you envision the Autonomous Agents that would be more impactful at your company, can you sketch out the triggers which you expect your Makers to leverage? Could you also estimate any triggers that would you not want then to use? Make notes of reasons in both cases.

Take it further: Sketch some standards for Trigger Usage at your firm – discuss with stakeholders.

|  |
| --- |
|  |
|  |
|  |

# Use Case #4: Ground your agent in relevant Knowledge Sources

*From aware to insightful – Equipping agents to respond with relevance and purpose by grounding them with the knowledge sources that matter.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Grounding agents with relevant information | Agents will need to be ‘grounded’ upon the knowledge which will help them to respond to inquiries and/or make decisions on which actions to take. By adding knowledge sources, you create a resource of relevant data in support of your agent’s objectives. | 15 minutes |

## Summary of tasks

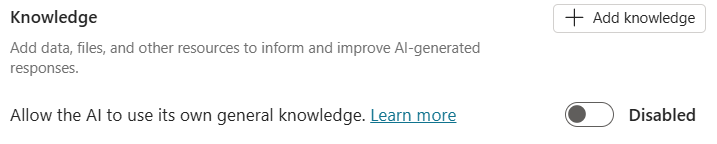
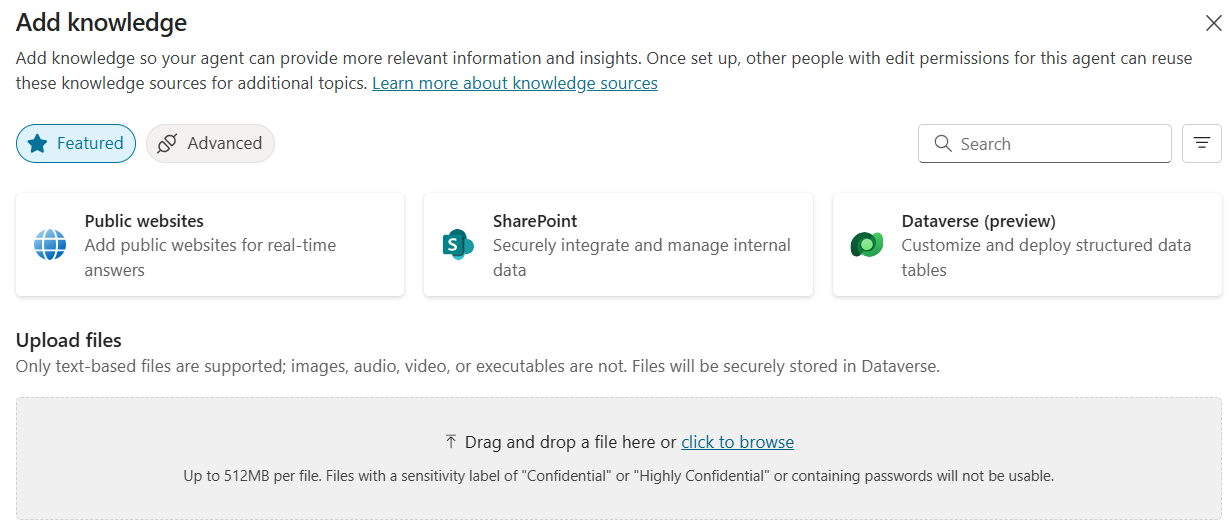
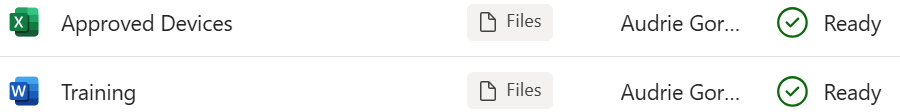
Let’s equip our agent with the knowledge that it needs to a. request approved devices, and b. schedule training for new employees. Note that if this were a real scenario, we could do much more, such as requesting badges, arranging for facilities space and furniture, and even sending the new employee a welcome letter with relevant onboarding requirements. You are only limited by your requirements.

**Scenario**: The Employee Onboarding Agent will need to leverage a few SharePoint lists for planning and task management which we will discuss later. In the meantime, our principal knowledge sources will be a spreadsheet with approved devices, a public website (for additional device information), and a document with required training for new employees.

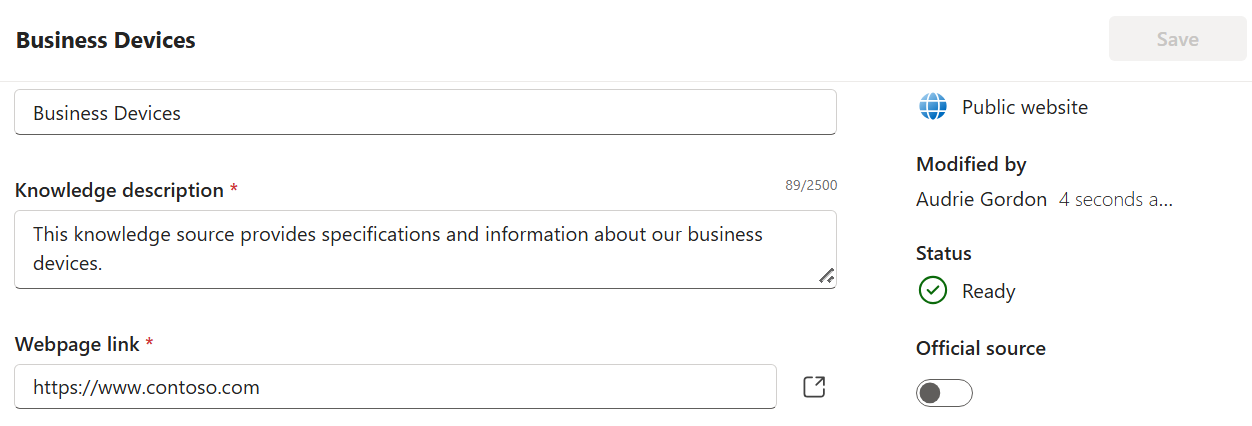
## Configuring Knowledge Sources

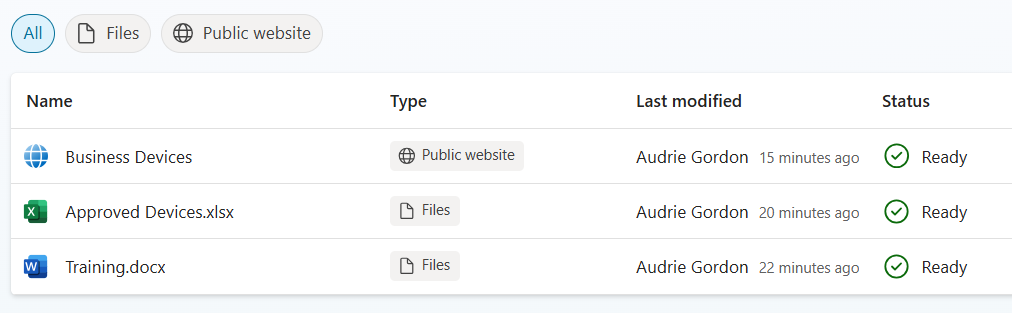
Here is an outline of our knowledge sources. The two files can be found in your Student Assets folder.

* Contoso.com public website – Since the spreadsheet with approved devices doesn’t have complete specifications on the equipment, we can also supplement knowledge with the company website which has more detail on the SKUs.
* Approved Devices.xls – Information on which devices have been approved for the different departments at Contoso.
* Training.doc[[2]](#footnote-3) – Document with the list of training required per department. This enables the agent to draft a schedule of trainings.

1. Navigate back to our Employee Onboarding Agent if necessary and then scroll down on the **Overview** of the agent to the section related to **Knowledge**. Please disable the toggle switch which will allow the agent to use its’ own knowledge. In our scenario, we only want the agent to use the knowledge sources we provide, and not to rely on knowledge it has learned through model training.  
   
2. Click the **+Add Knowledge** button on the top right of the Knowledge section. We will start by *uploading[[3]](#footnote-4) two files* which you have in your student assets: 1. Training.doc, and 2. Approved Devices.xls. To better understand the purpose of these files, please download them from the Student Assets folder, and open them to review each one before continuing. After closing the files, drag and drop them from your download location to the grey area shown in the screenshot of the knowledge tab below (or you may choose the **click to browse** link to select them):   
     
   
3. We want to assist the agent in understanding the purpose of these files, so please click on the file names to open and change the **name** and **descriptions** as shown below. Ensure you’ve clicked **Save** once you’ve made those changes.  
     
   **Training.docx** –   
   Description: This knowledge source contains information about new employee training.  
     
   **Approved Devices.xlsx** –   
   Description: This knowledge source contains the list of devices approved for new employees.
4. Click on the **Knowledge** tab at the top of the agent and wait (normally up to 3 minutes) to validate that both files are uploaded and marked as **Ready** on that screen. You may remove the file extensions from the names as well (optional).  
   

|  |  |
| --- | --- |
| A screenshot of Training name and description. | A screenshot of Approved Devices name and description. |
|  |  |

1. Next, we are going to add the Microsoft website. This will enable the agent to get additional specifications about the approved devices which are not found in the document we uploaded.  
     
   Select the **Knowledge** tab of your agent, then click **+Add Knowledge** again, but this time select **Public Websites** as the type of knowledge to be added, to enter this link, name and description.  
     
   **Link**: <https://www.contoso.com>   
   **Name**: Business Devices  
   **Description**: This knowledge source provides specifications and information about our business devices.  
     
   After you’ve added it, click on it to open the full details and double check the name and description as shown below.  
     
   



## Test your understanding

Now that you’ve set up the appropriate knowledge sources, your agent will be more insightful and have the information it needs to make decisions. You may choose to use any of the knowledge sources available on the Knowledge tab, but keep in mind they should be validated to confirm that they contain relevant details for the agent you are building. Think carefully about your use case, and evaluate which knowledge sources will add the most value.

**Key takeaways:**

* Knowledge sources are a way of grounding or axing our agent on what matters most for the current business scenario. Knowledge sources should be validated for relevance, freshness, and accuracy.
* Uploaded files are best for documents which are fairly static and don’t need frequent updates. Use SharePoint or other content management type knowledge sources for files that are changing regularly, or may be co-authored or edited often.
* Uploaded files are limited to 512MB in size.
* Uploaded files can not contain sensitivity labels, and are considered acceptably visible by all users/stakeholders of the agent.

**Challenge: apply this to your own use case**

* What types of knowledge sources are most relevant to your company? Are you planning to connect with SharePoint sites, Dataverse Tables, and/or Public Websites?
* Featured knowledge sources such as SharePoint, Dataverse, and Public Websites are only a small subset of the knowledge sources available today. These also include popular services such as ServiceNow and Salesforce. Explore the library of knowledge sources and try them out where they support your business scenarios most.

Take it further: Design an agent to solve a problem in our company? What would it do, why is it autonomous, and what knowledge sources would it use?

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

# Use Case #5: Empower your agent with Actions

*Autonomous agents leverage actions to get stuff done! – Actions are the wind beneath your agent’s wings.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Enable agents  with Actions | Autonomous Agents excel at getting stuff done! They leverage actions to perform transactions behind the scenes. Selecting and configuring actions is therefore a top priority for a successful agent. | 30 minutes |

## Summary of tasks

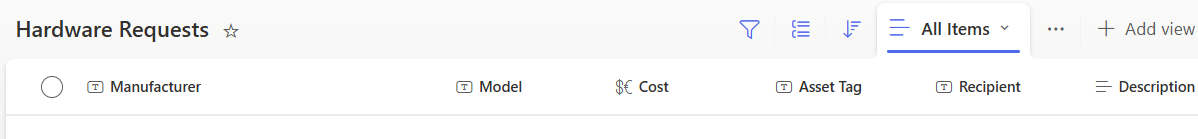
In this scenario we will need to configure two reusable actions. These actions are connections to the systems that will be used to make equipment requests and to schedule training.

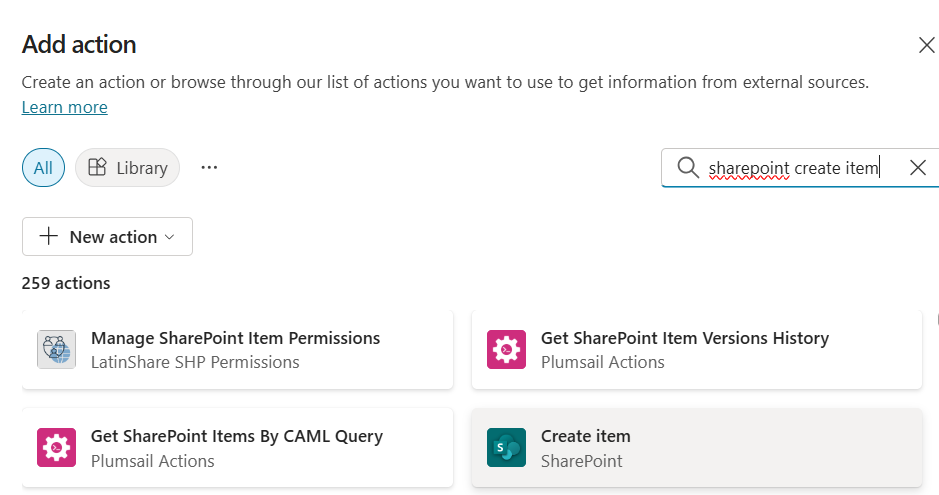
**Scenario**: When a new employee is added, the agent will determine and a. request the best equipment for the person and b. draft a schedule of training. Note: We will use an approval process in this case, which when approved will send out the meeting invite. However, approval processes are not typically required for Autonomous Agents unless you determine that there needs to be a human-in-the-loop.

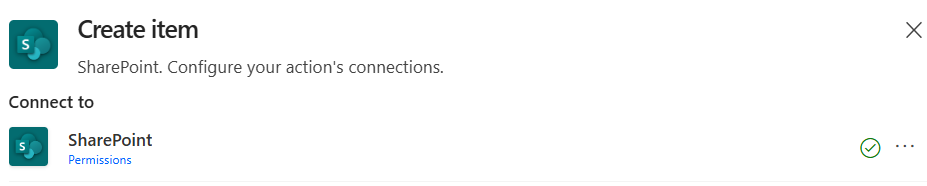
## Configuring Actions

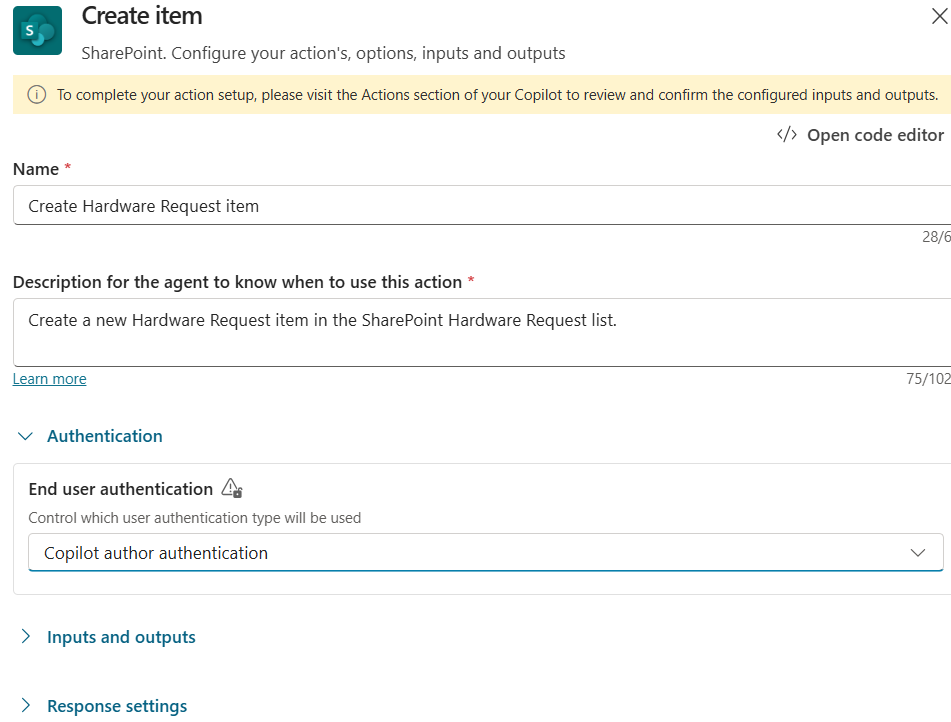
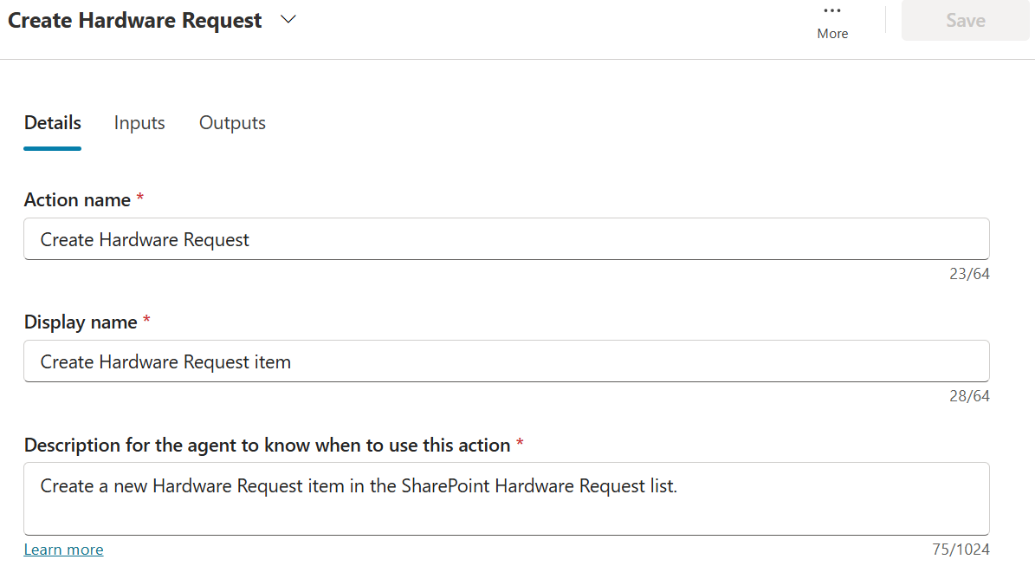
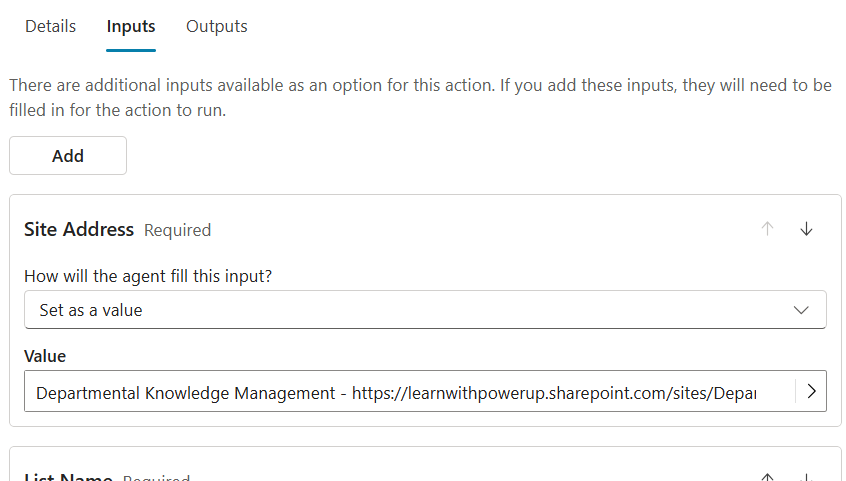
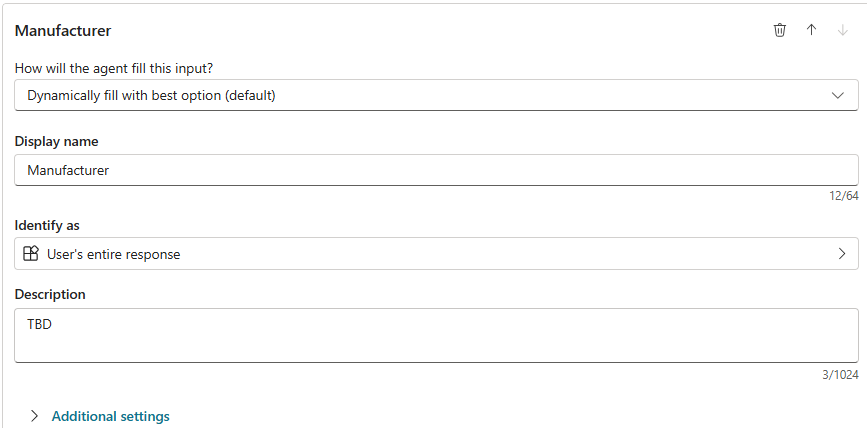
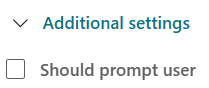
Next, we will leverage two actions:

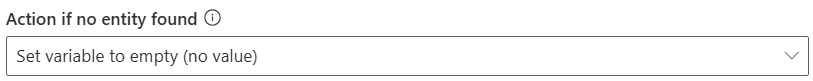
* Create a hardware request item – This action will be used by the agent to request new devices for employees so that the equipment can be imaged and delivered to them on site.
* Schedule training meetings – (includes approval process) This action will be used by the agent to draft a schedule of Teams meetings needed for required trainings based on departmental requirements and availability.

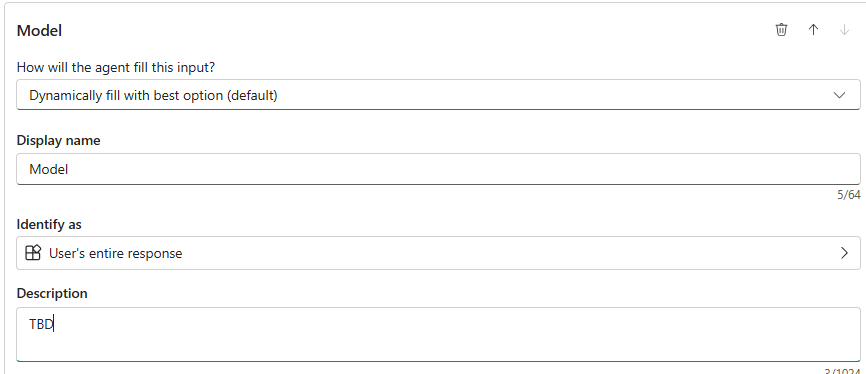
We will start with the action that will enable the agent to request equipment. In this case it will create a new request item in a SharePoint List called Hardware Requests. It contains six (6) critical fields: Manufacturer (renamed Title), Model, Cost, Asset Tag, Recipient (as single line of text), and Description.  
  


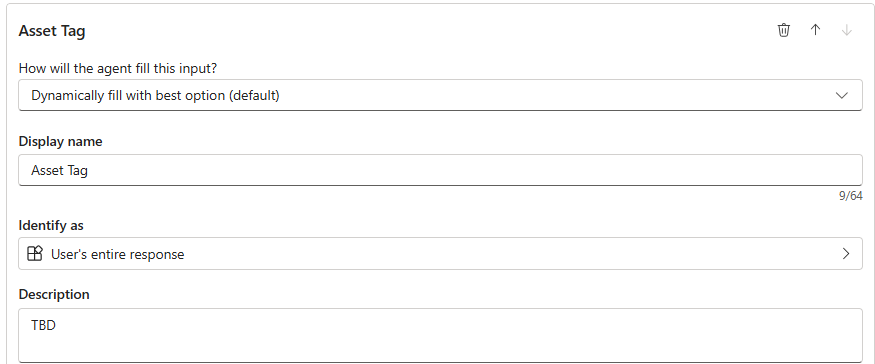
1. Select the **Actions** tab, then **Add an Action** to begin setting up a request for devices. Search for **Create Item** (SharePoint connector) and select it. This will redirect to a dialog where you can create or confirm your connection. Once the connection has a green checkmark, click **Next**.  
     
   

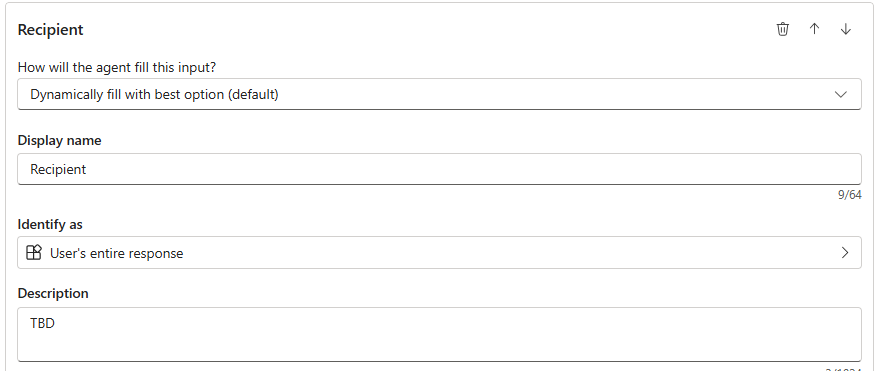


1. Let’s update the details about this action to make it more understandable for our Autonomous Agent to easily understand when to use this action. Let’s name this **Create Hardware Request**. We will also update the description to be **Create a new Hardware Request item in the SharePoint Hardware Request list**, then set the End user authentication to **Copilot author authentication**. Advantage for our scenario: By setting this to be the Copilot author’s authentication, end users will not need ‘create item permissions’ on the list. Click **Add action**.  
     
   
2. After our action is created return to the **Actions** tab to see the newly created action. Note that you an toggle this action off if you ever need to, and you can use the ellipsis … to delete it. Let’s refine our action. Click on the name of the action to edit it. Copy the text in the **Display Name** area and replace the **Action name** with it. Keeping naming consistent is a best practice which will result in more accurate selection of actions by your agent. Remember to click **Save** when done.  
     
   
3. Next click on the Inputs tab. Inputs are required data for the action to be completed, and the Outputs are how the agent passes data back to our agent. Inputs in this case, will be a direct mapping of the columns in the SharePoint list that will be updated. (By default, you will only see the required fields from the SharePoint list, but you’ll be able to add others.)  
     
   Let’s start by clicking in the **Value** field under **Site Address** (required), when you click within the input area you’ll be able to select our **Departmental Knowledge Management** site from the choices.  
     
   
4. Next, we will indicate the list name. All Inputs give a choice of allowing it to be dynamically filled in with the best option (default – and the agent will decide what’s best), or we can set a value when we want to make that decision for it. In this case, we will use **Set as a value**, and select our **Hardware Request** list.   
     
   **Tip**: If you click in the value area and choices do not appear with SharePoint list names, then Save the action, and reopen it to retry the List Name section.  
     
   
5. Now we will add all the inputs we need for our SharePoint list: Manufacturer (renamed Title), Model, Cost, Asset Tag, Recipient (as single line of text), and Description.  
     
   6.1 Manufacturer: Click the **Add** button at the top under the Inputs tab. Search for the Manufacturer column and click on it to add it as an input. (Note: Manufacturer may also be listed as Title since that column was a renamed system column in SharePoint.) Find the new input by scrolling down on the Inputs tab. This time we will use **Dynamically fille with best option (default)**, so the agent will decide what to put in there. Just put **TBD** in the Description for all inputs because we will fill those in later.   
     
     
     
   Expand **Additional Settings** below that and **uncheck** the option’ **Should prompt user’** because this is an autonomous agent and it will not need to prompt any users. You will do this for all SharePoint columns.  
     
     
     
   Lastly, scroll all the way down to the section **Action if no entity found**, and set it to **Set variable to empty**. You will do this for all SharePoint columns.

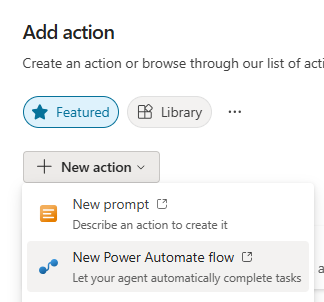
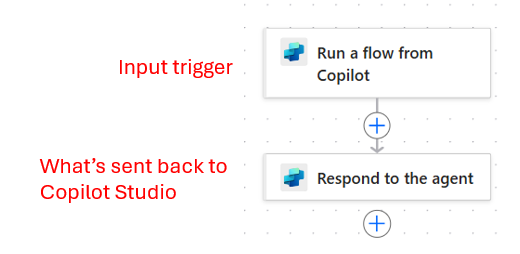


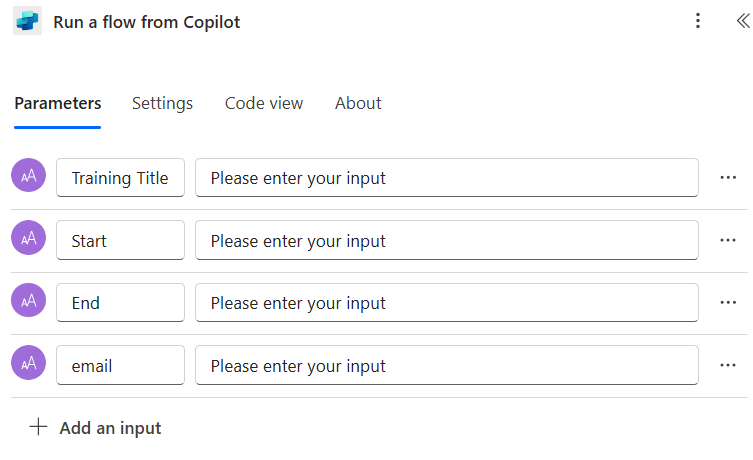
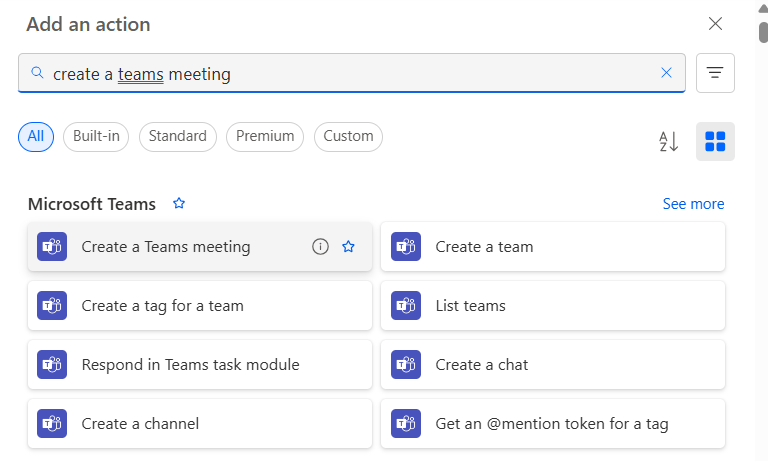
6.2 Model: Click the **Add** button at the top under the Inputs tab. Search for the Model column and click on it to add it as an input. Find the new input by scrolling down on the Inputs tab. This time we will use **Dynamically fille with best option (default)**, so the agent will decide what to put in there. Just put **TBD** in the Description, **uncheck** ‘Should prompt user’, and set ‘Action if no entity found’ to **Set variable to empty**.  
  


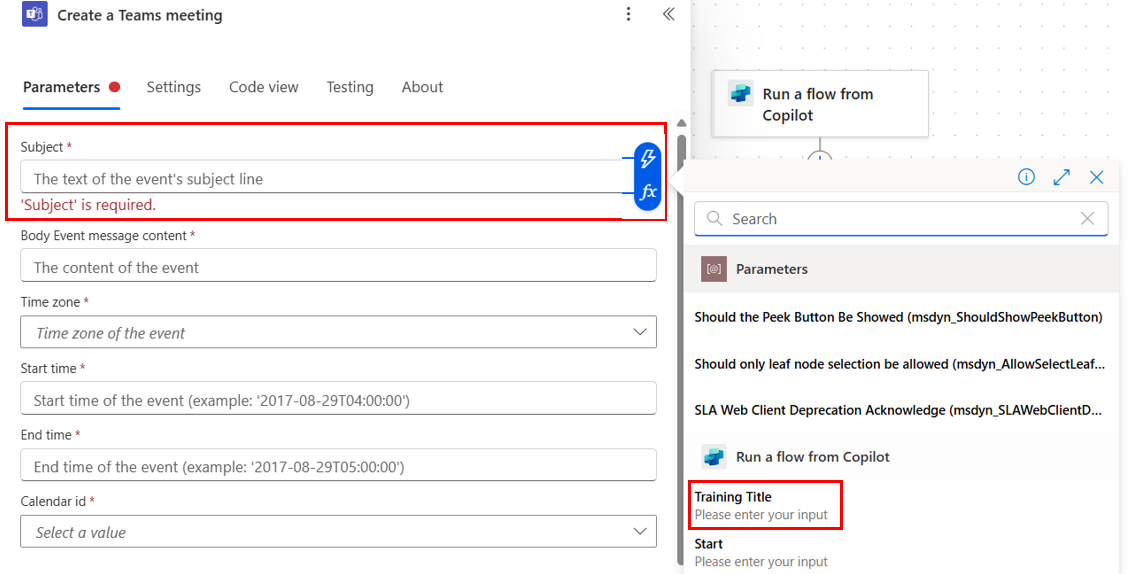
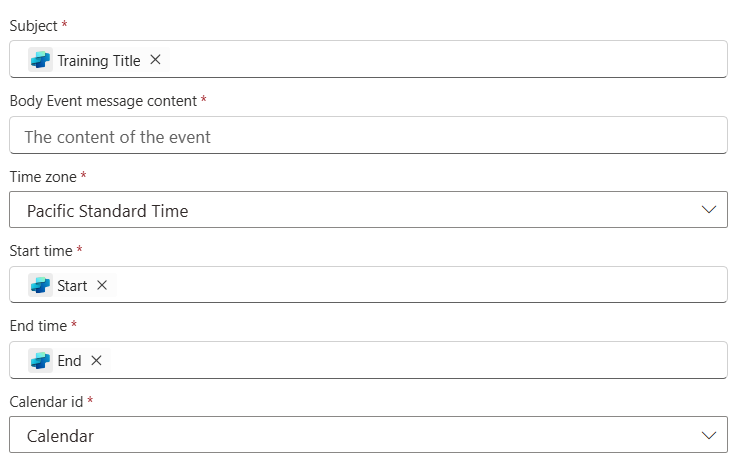
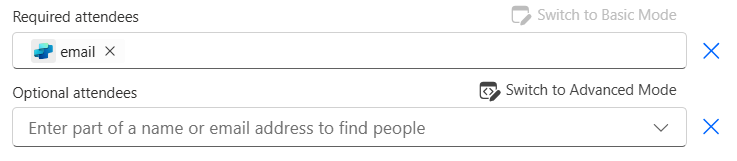
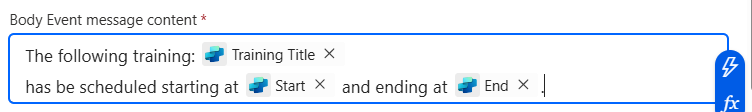
6.3 Cost: Click the **Add** button at the top under the Inputs tab. Search for the Cost column and click on it to add it as an input. Find the new input by scrolling down on the Inputs tab. This time we will use **Dynamically fille with best option (default)**, so the agent will decide what to put in there, but in this case, we want to make sure this is identified as a **Number**, rather than User’s entire response, which should happen for you if the SharePoint list type is number. Just put **TBD** in the Description, **uncheck** ‘Should prompt user’, and set ‘Action if no entity found’ to **Set variable to empty**.  
  
  
  
6.4 Asset Tag: Click the **Add** button at the top under the Inputs tab. Search for the Asset Tag column and click on it to add it as an input. Find the new input by scrolling down on the Inputs tab. This time we will use **Dynamically fille with best option (default)**, so the agent will decide what to put in there. Just put **TBD** in the Description, **uncheck** ‘Should prompt user’, and set ‘Action if no entity found’ to **Set variable to empty**.  
  


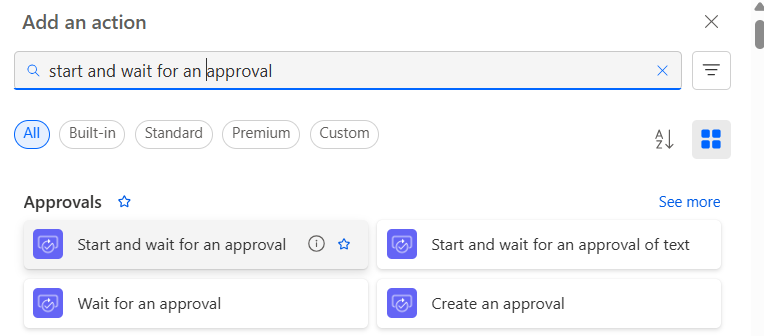
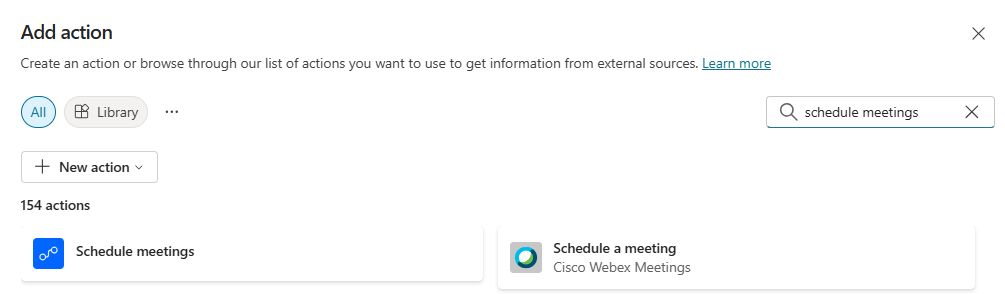
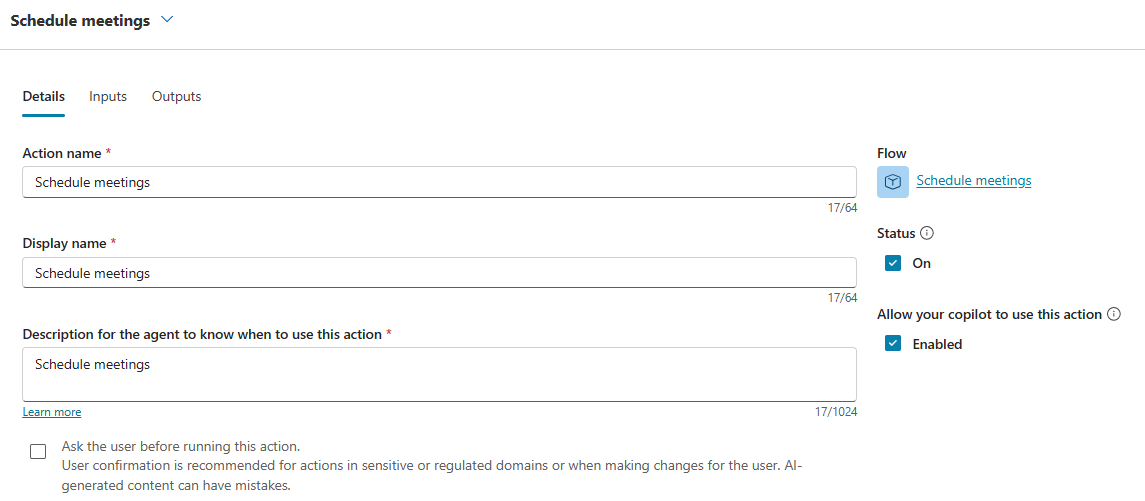
6.5 Recipient: Click the **Add** button at the top under the Inputs tab. Search for the Recipient column and click on it to add it as an input. Find the new input by scrolling down on the Inputs tab. This time we will use **Dynamically fille with best option (default)**, so the agent will decide what to put in there. Please note that in this case, the SharePoint column for Recipient is formatted as ‘single line text’. Just put **TBD** in the Description, **uncheck** ‘Should prompt user’, and set ‘Action if no entity found’ to **Set variable to empty**.  
  
  
  
6.6 Description: Click the **Add** button at the top under the Inputs tab. Search for the Description column and click on it to add it as an input. Find the new input by scrolling down on the Inputs tab. This time we will use **Dynamically fille with best option (default)**, so the agent will decide what to put in there. Just put **TBD** in the Description, **uncheck** ‘Should prompt user’, and set ‘Action if no entity found’ to **Set variable to empty**.



1. Take a moment to check your work for all the inputs and outputs for Create Hardware Request item. It is important to double-check details like this for autonomous agents, as this will ensure tests are executed without a glitch! Click **Save** when you’re done checking and revising things.
2. Next, we will set up a Power Automate flow which will create an approval process, that if approved, will create a Teams meeting to schedule the training. Cloud flows are great for automating more complex steps than a simple connector might be used for.  
     
   As you may recall, training requirements for new employees are listed in the Training.doc we uploaded earlier. So, our flow will do two things; 1. Determine the training needed for the new employee based on department, and 2. Schedule a Teams meeting for the training. We will include an approval process in the middle, so that someone will have to approve the meeting before the invite is sent out.
3. Click on the **Actions** tab, then **Add an Action** to begin, select **New Power Automate flow**.   
     
     
     
   Notice that the new flow created is preconfigured with **an input trigger** from Copilot Studio, and a **response action** which can send information back to Copilot Studio when completed. We will insert actions between these two in the order we’d like them to occur.   
     
   

1. Let’s start by configuring the inputs by clicking on the trigger **Run a flow from Copilot**.   
   To create the meeting invite we need four inputs: Start, End, Meeting Title, and email address of the employee. (The organizer will be the Copilot Maker in this case.) Please add four inputs by clicking the **+Add input** button and then I created the four inputs as shown below:  
     
   
2. Return to the main canvas and click the **+ (plus sign)** between the trigger and the action. Search for **‘Create a teams meeting’** in the search bar, then select the Microsoft Teams ‘Create a Teams meeting’ action.  
     
   
3. The ‘**Create a Teams meeting**’ panel will open on the left panel. We will map the required fields from the inputs we created within the trigger using dynamic values.   
     
   **Pro Tip** : The default name of the flow is ‘When an agent calls the flow’. That name is not good enough to alert the Autonomous Agent when to use it. So, it is a best practice to rename flows as a first step in every configuration. After the flow is created, you can use the Action tab in Copilot Studio to re-open details and match the name of the action to the display name of the flow. Consistency matters to avoid confusing the agent!  
     
   For now, rename the flow to **Schedule meetings.**

12.1 Subject: Click in the input field under Subject, then click the **lightning bolt** to the right to see available dynamic values. Note that the bottom section **Run a flow from Copilot** is where we can find all the inputs we created earlier. Select **Training Title** to populate this required field.  
  
   
  
12.2 Start: Click in the input field under **Start time**, then click the **lightning bolt** to the right to see available dynamic values. Note that the bottom section **Run a flow from Copilot** is where we can find all the inputs we created earlier. Select **Start** to populate this required field. We will do the same thing for End time next.   
  
  
  
12.3 End: Click in the input field under **End time**, then click the **lightning bolt** to the right to see available dynamic values. Note that the bottom section **Run a flow from Copilot** is where we can find all the inputs we created earlier. Select **End** to populate this required field. Under the end time please select the default calendar: **Calendar**. Of course, if this were a real solution, you could decide which calendar would be used and adjust other fields as needed. Also, you may notice in the screenshot that for this lab, we selected the Time zone of **Pacific Standard Time**, but feel free to select your own.  
  
12.4 Required Attendees: Click the **Show All** button to display other fields available for this connector. Scroll down to **Required Attendees**, then click **Switch to Advanced Mode** to enable the lightning bolt icon again. Click the **lightning bolt** to the right to see available dynamic values. Note that the bottom section **Run a flow from Copilot** is where we can find all the inputs we created earlier. Select **email** to populate this required field.  
  
  
  
12.5 Body Content: This is the message the Attendees will be shown in the Teams meeting invite. Where you see the copilot logo in the screenshot below, please use the lightning bolt again to insert the input values from the trigger. This data is redundant as the Teams meeting will have the title and the start and end times, but it will give you a chance to see how dynamic values can be inserted into the meeting description area.  
  


1. We will insert an approval step before the meeting invite so that the invite is not created until the instructor approves it. This is not a technical requirement, but can be useful when you’d like a human-in-the-loop before tasks are performed.  
     
   Click the + (plus sign) between the trigger and search for the **Start and wait for an approval** action then select it in the area below the search bar. (Ensure it lands between the trigger and the Create a Teams meeting action.)  
     
   If you have never done approvals before, you will need to create a new connection and provision the Approvals service. (If so, click the **Create new** button and this will be done for you.)  
     
   
2. Fill out the Approval details in this way:  
     
   **Approval Type**: Approve/Reject – First to respond  
   **Title**: Training Title (dynamic value from trigger)  
   **Assigned To**: yourself (start typing your name as it appears in the tenant, then select it from the drop down)  
   **Details** (please replace the {placeholder } values with the dynamic input values for the trigger):   
   Please approve the creation of a training meeting for {insert email dynamic value}, starting on {insert start dynamic value}, and ending on {insert end dynamic value}.
3. Click **Save draft** then click **Publish** before hitting the **back arrow** (top left) to return to Copilot Studio.   
     
   Important side request: Later when we test this flow, and if you are working in the Power CAT tenant, please do not approve any tasks, in that case only, select ‘Denied’ rather than Approved since you do not have a calendar there.  
     
   Pro tip: When you Publish a flow it may report errors in the top bar of the flow interface. Read the instructions and follow them to resolve any errors it picked up in the flow checker.
4. Once back in Copilot Studio click **Refresh**, then search for and locate your new flow to open it as a new action.  
     
      
     
   Ensure the name is **Schedule Meetings** and click **Add action** at the bottom.  
     
   
5. Return to the **Actions** tab at the top of your agent and locate the **Schedule Meetings** action.
6. Click directly on the name of the action **Schedule meetings** to open it on the **Action tab**. Ensure it matches the screenshot below.  
     
   

## Test your understanding

Now that you’ve created actions for your agent, it will be able to perform all the tasks that you had planned for it. Remember that actions are critical to autonomous agents as they can only perform transactions which have been properly configured within the **Actions** tab. Always double check that you are being consistent with naming conventions, and that you are naming things in common terms rather than coded protocols.

**Key takeaways:**

* Be consistent with action names as these will be used later in instructions.
* Be clear when filling in action description as these can be used by the agent to understand when to use the action in question.

**Challenge: apply this to your own use case**

* What kind of actions can you think of using in your own envisioning? Do you expect to need to update data sources (Dataverse, ServiceNow, SharePoint, etc.)? Do you have any custom connectors that might be needed where the out of box connectors are not enough?
* Remember that Autonomous may, or may not, need approvals before completing tasks. However, if there is a risk to mitigate it does not hurt to inject approvals into the flow. However, recognize that when you use waiting approvals, the flow of the transactions is paused until a human responds.
* Humans can respond to approval requests from the Power Automate approval center, from email notifications, or from Microsoft Teams notifications.  
  Learn more about Power Automate Approvals [here](https://learn.microsoft.com/power-automate/get-started-approvals).

Take it further: Make a sketch of the most common actions you might expect in your next Autonomous Agent. Will they need a human-in-the-loop, why or why not?

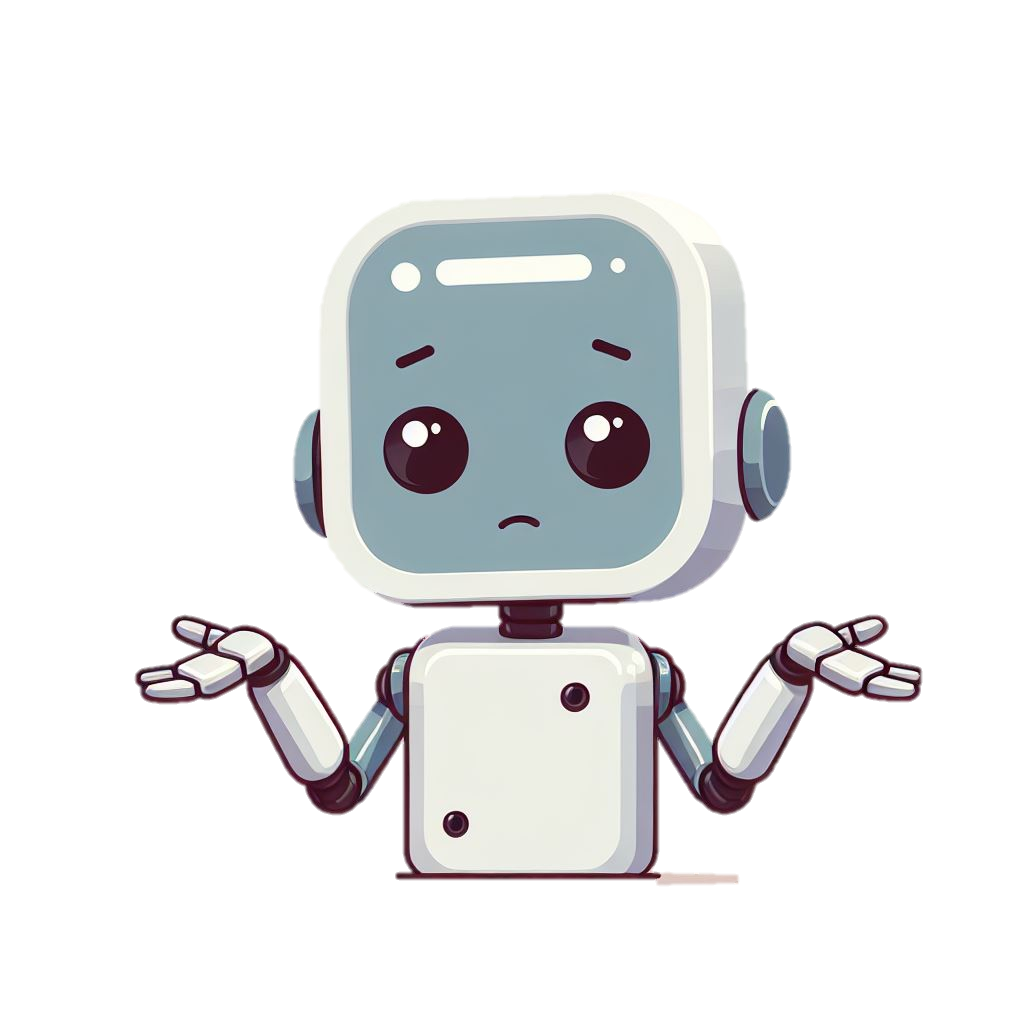
|  |
| --- |
|  |
|  |
|  |
|  |

# Use Case #6: Instructing your agent

*Guide agents with instructions!—The control of what is done, and in what order, fortunately remains in the hands of the Maker using natural language to instruct the agent!*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Instructing your  agent | By providing your agents with instructions, you’re onboarding them to what is really important to you. Treat them like an AI co-worker and provide step-by-step instructions so they respond consistently, although dynamically, in every case. | 10 minutes |

## Summary of tasks

Without instructions your agent won’t know what to do. Imagine you made a robot without a mission. Even though you gave it a lot of knowledge, and maybe even feature capabilities; without instructions it might just stand there. It knows how to move but doesn’t know where to go. It knows how to grab things, but it doesn’t know what to pick up. It has a voice but doesn’t know what to say. An Autonomous Agent without instructions is like a robot without a mission!

Unclear instructions result in poor responses. Effective instructions include **role** (what is the purpose of job title of the agent), **scope** (what should it do, and what should it avoid), **context** (what does the agent need to know), **tone** (how should the agent respond), and **error handling** (what should the agent do if it’s unsure).

You may have noticed that we saved instructions for the last build activity in this lab on Autonomous Agents. There was a reason for that! Since instructions will typically reference previous configurations, it may be more efficient to do the instructions lastly. So, although you may enter instructions at any time during the building of your agent, there are 3 reasons why we have selected to add them as a last step in this case:

* We want to reference the exact names of relevant knowledge sources exactly as they named on the **Knowledge** tab.
* We want to reference the exact names of actions exactly as they are displayed in the name field on the **Actions** tab.
* We may need to reference **relevant variables** or parameters for decision-trees that will be helpful to the agent.

**Scenario**: Provide instructions to the agent so that it will execute responses and actions in the appropriate order, with guidance on role, scope, context, tone, and error handling.

## Configuring instructions

1. Click the **Overview** tab of the Employee Onboarding Agent, click the **Edit** button. This will allow us to edit details and instructions in the top areas of the page.
2. In the **Instructions** input area, set the following text (simplified for our use case):

|  |
| --- |
| When a new employee is added, you will select training from a list of available training sessions for their department and then schedule Teams meetings for the new employee. You must also find a suitable device for the new employee based on their department and submit a hardware request to the 'Hardware requests' list.  ###Instructions:   1. Use the 'Training' knowledge source to select a list of required training. Then create Teams meetings for each training session using the 'Schedule meetings' action. Ensure that the Teams meeting is created on the corresponding weekday for the training course. 2. Select a device appropriate for the employee based on their department from the 'Approved Devices' knowledge source. For example, IT employees require high-specification devices, Sales employees require lower-specification devices. 3. Create a new hardware order request for the selected device using the 'Create Hardware Request' action. |

1. Click **Save** to record your changes and then proofread the instructions to ensure nothing is missing, and all references match exactly what has been configured in the **Knowledge** and **Actions** tabs.
2. In support of our instructions, we will need to complete the **input descriptions** for our previously configured Actions (remember we put TBD there earlier). These input descriptions will help the agent to know *how* *to get the necessary inputs*. Click on the **Actions** tab at the top of the agent. These descriptions help to reduce hallucinations, avoid the agent seeking an answer from a user, and ultimately preventing the agent from being unable to complete the task.
3. Click on the **Create Hardware Request action** name, then click on the **Inputs** tab for it. We are going to replace the TBD placeholders we added earlier with the correct descriptions.  
     
   5.1 Manufacturer: Scroll down to the section for **Manufacturer** and locate the **Description** field under Identify as. Add this description:   
   Fill with the Manufacturer name of the device from the "Approved Devices" knowledge source.  
     
   5.2 Model: Scroll down to the section for Model and locate the **Description** field under Identify as. Add this description:  
   Fill with the Model of the device from the "Approved Devices" knowledge source.  
     
   5.3 Cost: Scroll down to the section for Cost and locate the **Description** field under Identify as. Add this description:  
   Fill with the cost of the device from the "Approved device" knowledge source in USD format.  
     
   5.4 Asset Tag: Scroll down to the section for Asset Tag and locate the **Description** field under Identify as. Add this description:  
   Create a 6-digit asset tag starting with the letters ASSET-.  
     
   5.5 Recipient: Scroll down to the section for Recipient and locate the **Description** field under Identify as. Add this description:  
   Fill with the name of the user the device is for from the "When an item is created" trigger.  
     
   5.6 Description: Scroll down to the section for Manufacturer and locate the **Description** field under Identify as. Add this description:  
   Fill with additional details about the device from the "Business devices" knowledge source such as RAM, Hard Drive, Screen size, etc.
4. Click on the **Schedule meetings** action name, then click on the **Inputs** tab for it. We are going to replace the TBD placeholders we added earlier with the correct descriptions.  
     
   6.1 email address: Scroll down to the section for **email address** and locate the **Description** field under Identify as. Add this description:   
   Use the new employee email address for the "New Employee" list.  
     
   6.2 Start time: Scroll down to the section for **Start time** and locate the **Description** field under Identify as. Add this description:   
   Convert the Start time from PST to UTC using this as an example: 2025-01-06T14:57:51.2948938Z  
     
   6.3 End time: Scroll down to the section for **End time** and locate the **Description** field under Identify as. Add this description:   
   Generate an end time by adding the duration to the start time, converting PST to UTC, formatted as in this example: 2025-01-06T14:57:51.2948938Z  
     
   6.4 Training Title: Scroll down to the section for **Training Title** and locate the **Description** field under Identify as. Add this description:   
   Use the training title.

If you’d like try to trigger the agent again using an HR department to notice the differences in what the Agent will do.

1. Now the agent has it’s instructions both for the tasks it has to do, and it knows how to get any input it needs. Bravo! You’re doing an amazing job at this! We’re done configuring our agent, and next we will test it.

## Test your understanding

Now that you’ve incorporated instructions in your agent, step back and consider how important this step is, and how it can impact the results of your agent.

**Key takeaways:**

* Insert instructions in the order in which the tasks should be performed. Paying particular attention to when something might be needed before a next step can be completed.
* When adding instructions, remember to reference the exact names of relevant knowledge sources exactly as they named on the Knowledge tab.
* When adding instructions, remember to reference the exact names of actions exactly as they are displayed in the name field on the Actions tab.
* When adding instructions, remember to reference any relevant variables or parameters for decision-trees that will be helpful to the agent.

**Challenge: apply this to your own use case**

* Can you give some examples where it would be important to list instructions in the correct order? (Such as retrieving an email before sending an email.)
* Why do you think providing instructions to an agent is more user-friendly than having to create a Power Automate flow for everything?
* Remember with Autonomous Agents there is no conversation panel prompting the agent. Therefore, instructions, knowledge, and actions work together to ensure accurate next steps.

Take it further: Design instructions for an agent that is responsible for inventory management. What are some things an inventory manager would need to do, and how can you ensure the instructions are appropriate for those tasks?

|  |
| --- |
|  |
|  |
|  |
|  |

# Use Case #7: Quality Assurance

*Testing and Monitoring for Success! – Understanding how Testing and the Activity Tab can ensure high quality performance.*

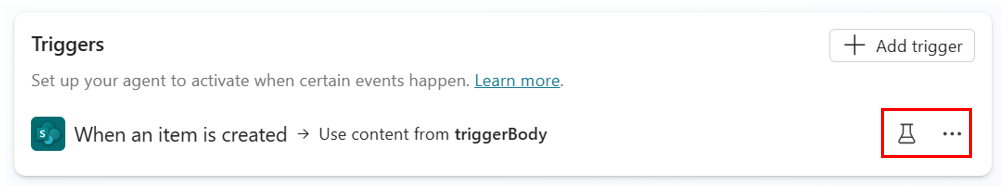
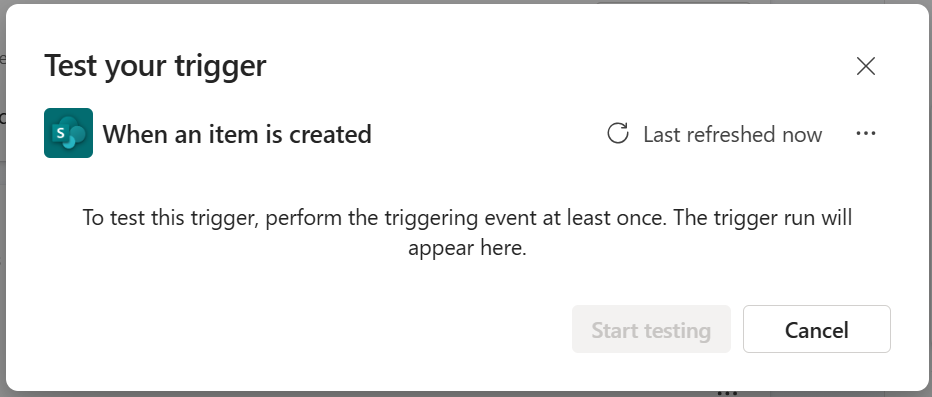
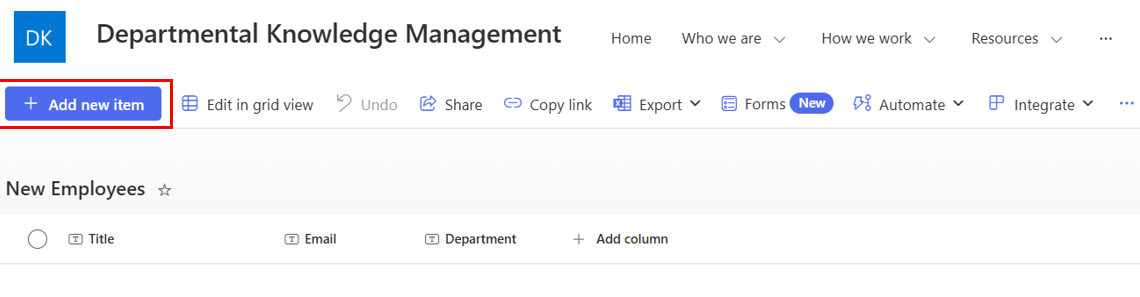
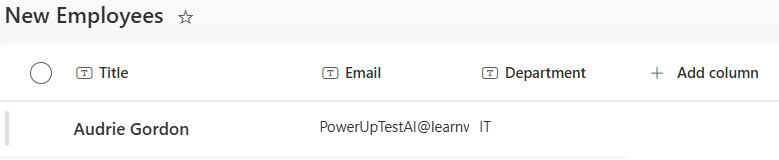
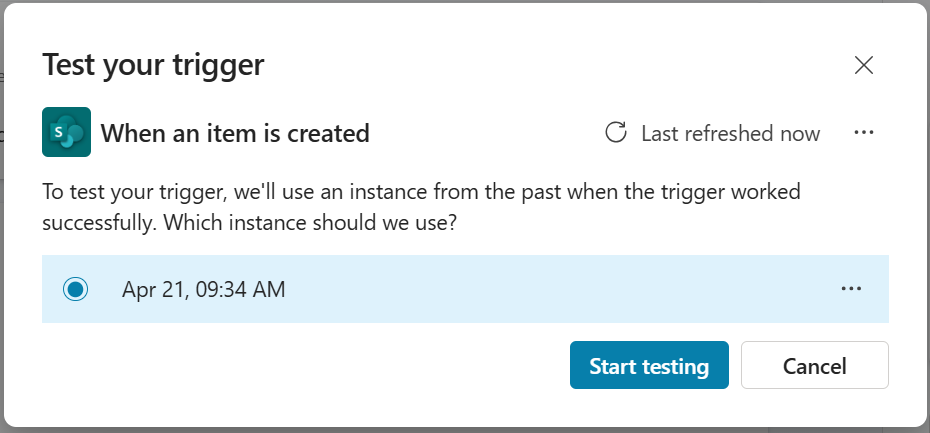
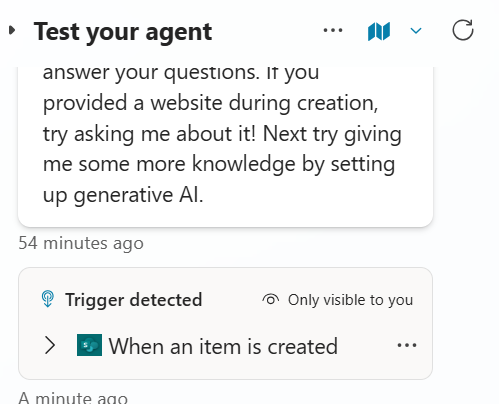
|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Activities and  Testing | Quality assurance requires trigger activity testing during the build of your agent, and before publishing it. Applying best practices in test will ensure your agents perform accurately, consistently delighting customers with the best outcomes. | 10 minutes |

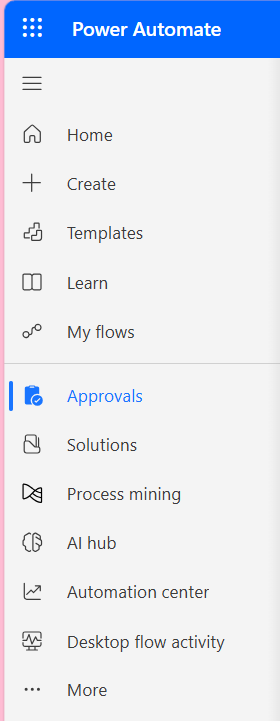
## Summary of tasks

We will test the results of all our agent configurations using the Trigger Test icon, and will monitor results on the Activity tab.

**Scenario**: In our Employee Onboarding Agent scenario, we have one trigger to test. This trigger will respond dynamically whenever a new SharePoint list item is added in the New Employee list. We will add test items to the list to trigger our agent for testing purposes. Suggestion: Please use names and email addresses found within the tenant whenever possible. If you are using your own tenant, it may be a good idea to test with your own name and email.

## Testing the Trigger

1. Navigate back to the Overview tab and scroll down to the section for Triggers. You will notice a little **test vial symbol** and an **ellipsis** (…) to the right of the Trigger we created earlier. The ellipsis will allow us to open the Power Automate flow that was created in support of the trigger. It polls for activities related to that connector so that the agent will know when it’s time to start work! Next, we’re going to use the little test vial symbol to initiate a test of our scenario.  
     
   
2. Click the **test vial symbol** to the left of the ellipsis (shown above in screenshot). Notice it explains its’ purpose, but it has the Start testing button greyed out because the connector has not been triggered. Click **Cancel** for now. Next, we will go to add a new item to the **New Employee** list (aka the Trigger event) to fix this!  
   
3. Create a new tab in your internet browser and then open the **Departmental Knowledge Management** SharePoint site and open the **New Employee** list.
4. Once there click the **+Add new item** button to create a new list item.   
     
     
     
   For the Title put your **user display name**, for the email enter **your user email address**, and for the Department choose **IT**. Important: In the case of the Title and email address, please use the *one assigned to you in the tenant you are using*, so that it will be recognized as having a calendar there. Save your item and navigate back to copilotstudio.microsoft.com to the environment where we were working earlier.  
     
   
5. Now we will try to test again. Click the **test vial symbol** next to the trigger. This time it will include the first **trigger activity**. Note that your date and time will reflect the day you actually added the list item in SharePoint. This dialog box will allow you to perform a test on any activity related to the trigger connector. Now click **Start testing**. Under the start message the conversation test panel will light up (just for you) so you can track the testing results.  
     
     
     
   
6. Notice the activity map on the left side of the Conversation panel. The agent has been triggered and begins by filling in all the input needed.  
     
   A screenshot of the Activity Map in Copilot Studio
   
7. Navigate to make.powerautomate.com and click the **Approvals** link in the left navigation.  
   Remember to check that you are still in the same environment as the agent.  
    @HENRY: Hit a problem, there is an incident opened which I need resolved to complete the test. [Incident 621389789](https://portal.microsofticm.com/imp/v5/incidents/details/621389789/summary?tmpl=Df2p2Q) : Microsoft Actions: Autonomous Agent Action Not Triggered

-Audrie  
  


1. All Approvals pending will display in this approval center, but not only here. Power Automate approvals will also send an email notification, and a notification in Microsoft Teams, so it’s not missed. You may approve or reject these from either location. For the sake of this lab only, we request that you **Deny** this approval if you are working in the training tenant, as your account is not set up with email or Teams. If this was approved though, a new Teams meeting invite would be created for the employee’s training as per our configurations.
2. Let’s check the Hardware requests list as well. Click on the tab in your browser where the SharePoint site Department Knowledge Management is located and then navigate to the **Hardware requests list**. We will check if an appropriate hardware request was added for the new employee.
3. Let’s publish our agent! Since our agent is working and doing exactly what we asked, let’s go ahead and publish it so that it will continue to do this consistently on it’s own. Click **Publish** to release your agent to the world.   
     
   **Congratulations!**

## Test your understanding

Now that you’ve tested your agent, consider how this step improves your deployment process.

**Key takeaways:**

* Triggers can be tested once the event related to the connector is initiated. So, we had to add a new item to the SharePoint list and then we could test the agent response. That dialog box allows the maker to test on any event initiated for the connector target.
* The activity tab is where a Maker can see all the activities that have been triggered and the results of those activities.
* The test conversation panel and the Activity tab will only show activities which have been triggered by the Maker and not the users after the agent has been published.

**Challenge: apply this to your own use case**

* Can you standardize testing in your company so that all Makers, both new and old, will be required to complete tests before agents are published?
* How might you implement training related to testing solutions in an environment where there are few professional developers, and everyone wants to make agents?

Take it further: Design acceptance criteria for your next Copilot Studio agent. Make a few notes below on the things you’d like to enforce.

|  |
| --- |
|  |
|  |
|  |
|  |

# Summary of learnings

Une image contenant jaune, conception

Le contenu généré par l’IA peut être incorrect.*Mastery is not a destination but a journey—a joyful path where every step brings growth, discovery, and endless possibilities.*

Congratulations on completing this lab! You’ve explored the essential components of Microsoft Copilot Studio Autonomous Agents, learning how to create, refine, and deploy an intelligent agent.

* **Building Autonomous Agents** – Building Autonomous Agents is a seven (7) step process:  
  1. Creating a Solution (for distribution, reuse, and ALM best practices)  
  2. Building the agent (can be done conversationally or manually) and putting it into the Solution  
  3. Setting up a trigger so the agent knows when to start working  
  4. Grounding the agent in knowledge sources so it can make decisions and respond to inquiries  
  5. Enabling the agent with actions so it can perform tasks   
  6. Adding instructions so the agent knows when to do what, and in what order  
  7. Testing your agent for Quality Assurance
* **Ensuring Reliability** – Don’t forget to test your agents to ensure they perform exactly how you planned. Note that this is often an iterative process, rarely are instructions perfect on the first try. Be prepared to test and tweak until expectations are realized.

**Conclusions and recommendations**

To maximize the effectiveness of your Copilot Studio Autonomous agent:

* Remember to ground the agent with knowledge it can use to make decisions and respond
* Use the same names in your actions and knowledge sources within the instructions that you give your agent.
* Iterate through test and tweaks to get the expected results!

By applying these best practices, you’ll create an assistant that delivers value and evolves with your organization’s needs.

# Glossary

*Speak the language, bridge the world—unlock hearts, opportunities, and the true essence of every land.*

**Agent:**  
A digital assistant powered by AI, capable of understanding and responding to user inputs. In Copilot Studio, agents can be customized to for conversational experiences and/or can act autonomously based on pre-configured triggers and instructions.

**Autonomous Agent:**

Agents which can automatically respond to signals across your business and initiate tasks. They can be configured to react to events or triggers without human input that instead originate from various tools, systems, and databases, or are even scheduled to run hourly, daily, weekly, or monthly.

**Trigger:**  
An autonomous agent trigger is an AI-driven system that operates independently based on events external to the agent. Unlike conversational agents, autonomous agents run in the background and respond to complex scenarios informed by real-time data and connectors.

**Action:**  
You can extend the capabilities of your custom agent by adding one or more actions. Actions will be used by your autonomous agent to respond to triggers and instructed events automatically.

**Power Automate:**

Microsoft Power Automate is a comprehensive cloud-based automation platform designed to streamline and optimize business processes across various systems, desktop applications, and websites. It leverages low-code and AI-driven features to enable users to automate repetitive tasks and workflows, thereby saving time and enhancing productivity.

**Instructions:**  
Custom settings or guidelines configured in Copilot Studio to shape the behavior of Copilot agents. Instructions define how the agent should respond to specific queries or scenarios.

**Quiz Answers:**

**Quiz #1**: Which approach would you pick for these scenarios?  
Scenario A: Cloud flow  
Scenario B: Autonomous Agent  
Scenario C: Autonomous Agent

**Quiz #2**: [Question]  
Scenario A: [Answer]  
Scenario B: [Answer]  
Scenario C: [Answer]

**Quiz #3**: [Question]  
Scenario A: [Answer]  
Scenario B: [Answer]  
Scenario C: [Answer]

**Quiz #4**: [Question]  
Scenario A: [Answer]  
Scenario B: [Answer]  
Scenario C: [Answer]

**We want your feedback!**

[**Start now**](https://aka.ms/MCSLabsFeedback)

1. Learn more about Power Automate [here](https://www.microsoft.com/power-platform/products/power-automate). If you have not used Power Automate before, please consider any Business Process designer software or service that creates pre-defined workflows. Compare that to the value-added for Autonomous Agents. [↑](#footnote-ref-2)
2. The approved devices spreadsheet, and the training file, do not change often, so we can upload these as static knowledge sources. [↑](#footnote-ref-3)
3. Files uploaded in this way are usually static files that don’t change frequently. They must also be under 512 MB in size, and would not have sensitivity labels which are unsupported. [↑](#footnote-ref-4)