**LAB-**02 **Set yourself up for success & discover ALM best practices**

**Build with confidence. Deploy with control. Master the lifecycle of your Copilot Studio agents with best practices.**

# Lab Details

|  |  |  |  |
| --- | --- | --- | --- |
| Level | Persona | Purpose | Estimated time to complete |
| 200 | Maker | After completing this lab, attendees will be able to apply Application Lifecycle Management (ALM) best practices to their Copilot Studio solutions. They will know how to structure their work using solutions and publishers, configure environment variables and connection references for deployment readiness, and set up Git-based source control using Azure DevOps—all without writing code. | 20 minutes |

## Prerequisites

* Developer environment.
* Access to an Azure DevOps organization, project and branch.

## Summary of targets

In this lab, you'll configure your ALM foundation for working with Copilot Studio like a pro. By the end of the lab, you will:

* Create and configure a structured solution for your customizations
* Set up a custom publisher to track ownership and maintain ALM hygiene
* Add environment variables and connection references for better portability
* Learn how to source control your Copilot Studio project in a Git repo
* Understand which Copilot Studio settings require manual post-deployment steps

These practices will prepare you for sustainable, secure, and automated delivery of your Copilot agents across environments.

| Use case/topic | Tagline | Page |
| --- | --- | --- |
| Create a solution and custom publisher [Use a structured container to group all agents components for better lifecycle management.] | **Structure your success** – Group, manage, and deploy all your agent components with clarity and control. | 3 |
| Create environment variables and connection references [Add reusable variables for URLs, keys, or other settings that differ across environments along with connection references to other systems] | **Adapt with flexibility** – Environment variables future-proof your agents for seamless multi-environment deployments. Manage credentials and services cleanly across dev, test, and prod. | 5 |
| Set up Git source control [Connect your solution to Azure DevOps Git to track changes and prepare for CI/CD—no code required.] | **Track and evolve** – Use Git to version, review, and automate deployment of your agent assets. | 7 |
| Summary of learnings | Mastery is not a destination but a journey—a joyful path where every step brings growth, discovery, and endless possibilities. | 6 |
| Glossary | Speak the language, bridge the world—unlock hearts, opportunities, and the true essence of every land. | 7 |

## Documentation and additional training links

* [Application lifecycle management (ALM) with Microsoft Power Platform](https://learn.microsoft.com/en-us/power-platform/alm/)
* [Key concepts - Publish and deploy your agent - Microsoft Copilot Studio](https://learn.microsoft.com/en-us/microsoft-copilot-studio/publication-fundamentals-publish-channels?tabs=web)
* [Power CAT Webinar: Copilot Studio ALM](https://aka.ms/MCS/ALM)
* [Use environment variables in solutions](https://learn.microsoft.com/en-us/power-apps/maker/data-platform/environmentvariables)
* [Use a connection reference in a solution](https://learn.microsoft.com/en-us/power-apps/maker/data-platform/create-connection-reference)
* [Overview of Dataverse Git integration - Power Platform](https://learn.microsoft.com/en-us/power-platform/alm/git-integration/overview)

# Use Case #1: Create a solution and custom publisher

*Use a structured container to group all agents components for better lifecycle management.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Create a solution and custom publisher | Structure your success – Group, manage, and deploy all your agent components with clarity and control. | 5 minutes |

## Summary of tasks

In this section, you’ll learn how to access the Solutions area of Copilot Studio, create a new solution, new publisher, and set the solution as default.

**Scenario**: Properly setup your development environment so that you can later easily package and deploy your agents to other environments.

## Step-by-step instructions

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

<https://aka.ms/MCSStart> https://copilotstudio.microsoft.com/

Make sure you land in the expected environment by checking the top-right corner, where the environment name is displayed.

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**.

1. If this is your first time creating a solution in this environment, create a new **Publisher**—this can be your organization’s name. Choose a prefix that Copilot Studio will use for all your customizations' technical names.
2. To make sure any new components are added to this new solution by default, select **Set as your preferred solution**.
3. Select **Create** when ready.

## Test your understanding

Now that you’ve created a solution in Copilot Studio, 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 #2: Create environment variables and connection references

*Use a structured container to group all agents components for better lifecycle management.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Create environment variables and connection references | Adapt with flexibility – Environment variables future-proof your agents for seamless multi-environment deployments. Manage credentials and services cleanly across dev, test, and prod. | 5 minutes |

## Summary of tasks

In this section, you’ll learn how to create environment variables and connection references in your solution.

**Scenario**: Configure your environment by anticipating future elements that will need to be updated as your solution moves from development to production.

## Step-by-step instructions

1. Open the **solution** you created in the previous use case.

1. Select **New**, then go to **More**, and select **Environment variable**.
2. In **Name**, use:

|  |
| --- |
| Custom Knowledge Endpoint |

1. In **Data Type**, select **Text**.
2. Define a **current value** based on [aka.ms/MCSWorkshopLabAssets](https://aka.ms/MCSWorkshopLabAssets)

💡 **PRO TIPS:**

* **Environment variables can be of various types, including Secret if you wish to retrieve secrets at runtime from an Azure Key Vault.**

1. In your **solution**, select **New**, then go to **More**, and select **Connection reference**.
2. Use the name of the connector as the **name**, e.g., **MSN Weather**
3. Select the connector: **MSN Weather**
4. In the **connection** dropdown, select **New connection** if no value is suggested.
5. **Create** the connection in Power Apps by logging in, and **come back** to the previous tab.
6. **Refresh** the connection and **select** the newly created one.
7. **Create**
8. Repeat these steps for the below connectors:

**Microsoft Dataverse**

**SharePoint**

**ServiceNow** (based on the values in [aka.ms/MCSWorkshopLabAssets](https://aka.ms/MCSWorkshopLabAssets))  
For ServiceNow, use Basic Authentication

# Use Case #3: Set up Git source control

*Connect your solution to Azure DevOps Git to track changes and prepare for CI/CD—no code required.*

|  |  |  |
| --- | --- | --- |
| Use case | Value added | Estimated effort |
| Set up Git source control | **Track and evolve** – Use Git to version, review, and automate deployment of your agent assets. | 10 minutes |

## Summary of tasks

In this section, you’ll learn how to create a new project in Azure DevOps, how to instantiate the main branch, and how to connect it to you developer environment.

## Step-by-step instructions

1. **Navigate** and log into **Azure DevOps**.

[aka.ms/MCSWorkshopADO](https://aka.ms/MCSWorkshopADO) <https://copilotstudio.microsoft.com/>

The first time you log in, you may need to **confirm** your user’s fictious name and country.

1. **Create** a project with the **name for your fictitious user**.
2. Select **Create project**
3. Once the project is created, navigate to **Repos** and go to **Branches**.

1. Choose **Initialize** under **Initialize main branch with a README or gitignore**

That’s it! Now let’s return in Copilot Studio.

1. Return to **Copilot Studio** ([aka.ms/MCSStart](https://aka.ms/MCSStart)), and return to the **Solutions** home page.
2. In the menu, select **Connect to Git**
3. For **Connection type**, select the **solution**.
4. Choose the **CopilotStudioTraining** organization, and select the **project** you just created, your repository. For **Root git folder**, set **Solutions**
5. Select **Next**, and select the **Solution** you have created in Use Case #1.
6. For Branch, select **Create new branch**, and call it **dev.**
7. **Save**
8. **Connect**

# Summary of learnings

Une image contenant jaune, conception

Le contenu généré par l’IA peut être incorrect.*True learning comes from doing, questioning, and reflecting—so let’s put your skills to the test.*

To maximize the impact of your ALM setup in Copilot Studio:

* **Use solutions as your foundation** – Keep all your components within a solution to simplify lifecycle management and ensure clean deployment.
* **Name wisely** – Adopt a consistent naming convention and always use a custom publisher to avoid default clutter.
* **Plan for portability** – Use environment variables and connection references to ensure your agent configurations adapt across dev, test, and production.
* **Document post-deployment steps** – Track settings that aren’t part of the solution (e.g., authentication, channels, sharing) so nothing is missed.
* **Leverage source control** – Use Git integration to track, audit, and collaborate—setting the stage for CI/CD without complex tooling.
* **Automate where it counts** – Consider using pipelines with Azure DevOps or GitHub for streamlined, repeatable deployments.

**Conclusions and recommendations**

ALM golden rules:

* Work in the context of solutions.
* Create separate solutions only if you need to deploy components independently.
* Use a custom publisher and prefix to maintain clarity and traceability.
* Use environment variables for settings and secrets that change across environments.
* Export and deploy solutions as managed, unless you're setting up a dev environment.
* Avoid customizing outside of dev.
* Consider automating ALM for source control and CI/CD pipelines..

By following these principles, you’ll establish a robust, scalable foundation for managing Copilot agents and Power Platform assets across their full lifecycle.

**We want your feedback!**

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

# Glossary

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

**Solution**  
A container in Power Platform that groups related components (like agents, flows, environment variables) for better organization, deployment, and lifecycle management.

**Publisher**A label associated with your solution components. Using a custom publisher helps track ownership and apply consistent naming prefixes.

**Environment**

A workspace in Power Platform where apps, agents, and data reside. ALM best practices often involve multiple environments (e.g., dev, test, prod) for structured deployments.

**Environment variable**

A reusable setting (like a URL or key) that can vary between environments without modifying individual components.

**Connection reference**

An abstraction that links connectors (e.g., SharePoint, Dataverse) to credentials and environment-specific settings—allowing reuse and cleaner ALM processes.

**Managed solution**

A read-only version of a solution used for deployment to downstream environments (test, prod).

Managed solutions support clean, controlled updates.

**Unmanaged solution**

Editable solutions used in development environments. Changes in unmanaged solutions can be exported, versioned, and eventually deployed as managed.

**Solution-aware**

A component or setting that is included in a solution and can be moved across environments as part of it. Not all Copilot Studio settings are solution-aware.

**Source control**

The practice of tracking and managing changes to your assets over time. In Power Platform, Git integration with Azure DevOps or GitHub helps enable collaboration, auditing, and automation.

**CI/CD (Continuous Integration / Continuous Delivery)**A practice of automating the build, testing, and deployment of solutions using tools like Azure DevOps pipelines or GitHub Actions.