Deploy with confidence: Power Platform pipelines & ALM

Master the deployment of your Microsoft Copilot Studio agents across environments using Power Platform pipelines. Learn the complete lifecycle from development to production.

Lab Details

Level	Persona	Duration	Purpose
300	Maker/Admin	30 minutes	After completing this lab, participants will be able to deploy Microsoft Copilot Studio solutions across environments using Power Platform pipelines, understand post-deployment configuration requirements, and manage source control integration. They will learn to identify solution-aware vs non-solution-aware settings and establish robust deployment practices.

Table of Contents

Why Master ALM Deployment?

Ready to move beyond development? You've built amazing agents in your dev environment, but now you need to deploy them safely and consistently to test and production environments.

Think of deployment pipelines as your quality assurance assembly line: - **Without pipelines**: Manual exports, forgotten configurations, "it worked in dev" syndrome - **With pipelines**: Automated, consistent, auditable deployments with proper validation

Common deployment challenges solved by proper ALM: - "My agent works perfectly in dev but breaks in production" - "I forgot to configure the authentication settings after deployment" - "Someone deployed directly to production and bypassed testing" - "I can't track what changed between versions"

This lab transforms deployment from a risky manual process into a confident, repeatable workflow!

Introduction

Application Lifecycle Management (ALM) deployment ensures your solutions move safely and consistently from development through testing to production. This lab builds on ALM foundations to implement automated deployment pipelines and understand the complete deployment lifecycle.

Real-world example: Your customer service agent is ready for production, but deployment involves: 1. Exporting the solution as managed 2. Importing to test environment for validation 3. Configuring environment-specific settings 4. Testing functionality end-to-end 5. Promoting to production with proper approvals 6. Configuring production-specific security and channels

With Power Platform pipelines, this becomes an automated, governed process with built-in approvals and validation!

Core Concepts Overview

Concept	Why it matters
Power Platform pipelines	Democratized ALM automation that brings CI/CD capabilities into the service in a manner that's approachable for all makers, admins, and developers. Significantly reduces effort and domain knowledge previously required for healthy ALM

	processes.
Pipeline stages	Sequential deployment environments (Development → Test → Production) that solutions must pass through in order, preventing bypass of QA processes and ensuring proper validation at each stage.
Service principal deployments	Option to deploy using service principal identity instead of the user maker's identity for consistent ownership and automation scenarios.
Solution artifacts	Exported solutions that remain unchanged throughout the pipeline, ensuring the same tested artifact moves through all stages without tampering or modification.
Managed vs Unmanaged Solutions	Managed solutions are read-only deployments for downstream environments (test/prod). Unmanaged solutions should be used only in development.
Solution-aware vs Non-solution-aware	Some Copilot Studio settings travel with solutions, others require manual post-deployment configuration. Understanding this distinction is critical for successful deployments.
Managed Environments	Governance feature that enforces managed solution deployments and prevents unauthorized customizations in target environments.

Documentation and Additional Training Links

- Overview of pipelines in Power Platform
- Export and import agents using solutions
- Extend pipelines in Power Platform
- Managed environments and governance
- Overview of Git integration in Power Platform

Prerequisites

- Completion of the Set yourself up for success lab with a solution containing environment variables and connection references.
- Access to multiple Power Platform environments (DEV, PROD). These are provided in the lab setup.
- Azure DevOps project with Git integration already established.
- PROD environments must be enabled as **Managed Environments** (enforced in lab setup).

Summary of Targets

In this lab, you'll implement a complete ALM deployment process for Microsoft Copilot Studio using Power Platform pipelines. By the end of the lab, you will:

- Create and configure Power Platform pipelines for automated deployment across environments.
- Deploy solutions from DEV to PROD environments using managed solutions.
- Identify and complete post-deployment configuration steps for non-solution-aware settings.
- Experience the benefits of Managed Environment governance in preventing unauthorized customizations.
- Commit changes to source control and understand the structure of unpacked solutions.

Use Cases Covered

Step	Use Case	Value added	Effort
1	Create Power Platform pipelines for deployment	Automate with confidence – Set up governed, repeatable deployment workflows that democratize ALM for all makers while maintaining security and control through platform governance.	10 min
2	Commit changes and understand source control structure	Track and structure – Use Git integration to maintain deployment history and understand how solution components are organized in source control.	5 min

---

Instructions by Use Case

Use Case #1: Create Power Platform pipelines for deployment

Set up automated deployment pipelines that democratize ALM while maintaining proper governance and security through platform controls.

Use case	Value added	Estimated effort
Create Power Platform	Automate with confidence – Set up governed, repeatable deployment	

pipelines for deployment	workflows that democratize ALM for all makers while maintaining security and control.	10 minutes

---

Summary of tasks

In this section, you'll learn how to create Power Platform pipelines, configure deployment stages, and set up automated deployments for controlled releases.

Scenario: You have a solution ready in DEV and need to establish an automated process to deploy it to a PROD environment with minimal effort and maximum consistency.

Objective

Create a deployment pipeline that automates solution deployment across environments with proper validation and governance controls.

Step-by-step instructions

Get a PROD environment

> [!TIP] If you haven't done so already, you need to request a PROD environment to be created for your user. This is a one-time setup step that will allow you to create pipelines for deployment.

1. **Start** by requesting a PROD envrionment to be created for your user. Use the **Workshop Agent** to request this environment, which will be automatically created for you. This will then take a couple of minutes to provision and to show up.

> [!IMPORTANT] Access the workshop agent in the same location as when you created your training user account. You will need the workshop code and your training user's email address if you previously closed the agent. Tell the agent to "Provision a PROD environment". You are limited to a single PROD environment for the duration workshop.

Access Power Platform pipelines

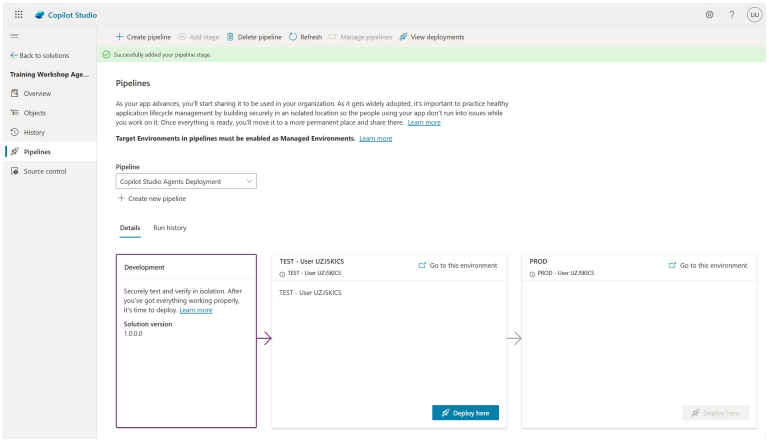
1. Navigate to the Copilot Studio home page at https://copilotstudio.microsoft.com/
2. Go to the **Solutions** menu (located in the left-hand menu under the ellipsis . . .) of your DEV environment
3. Select the solution you had created previously for your labs
4. In the left navigation, select **Pipelines**.
5. Select **+ Create new pipeline**.

Configure pipeline basics

6. Enter a **Name** for your pipeline, e.g., `<your user name> Pipeline`.
7. Set a **Description** to explain the pipeline's purpose (e.g., `Automated deployment of agents from DEV to PROD`).

Set up deployment stage

8. **Select** the **PROD** environment as the **Target environment**.
9. **Save** the pipeline configuration.



alt text

Test your pipeline

13. In the PROD card, select **Deploy here**.

> [!TIP] - The wizard then makes sure that each environment variable has a value set in the target environment, and that all connection references are valid. If any of these checks fail, you will be prompted to fix them before proceeding.

14. **Repeat** the operation for the PROD stage.
15. In Copilot Studio, **switch** to the PROD environment.
16. **See** what the agents look like in the PROD environment. When entering a topic, see how customizations are locked because the solution is managed.

Congratulations! You've created your deployment pipeline!

Test your understanding

Key takeaways:

- **Democratized ALM** – Pipelines make sophisticated deployment processes accessible to all makers without requiring deep ALM knowledge.
- **Automatic governance** – Solutions are automatically exported as managed for target environments, preventing unauthorized

changes.
- **Sequential validation** – Solutions must pass through pipeline stages in order, ensuring proper testing and approval workflows.
- **Built-in safeguards** – Pipeline artifacts can't be tampered with, ensuring the same tested solution moves through all stages.

Lessons learned & troubleshooting tips:

- Target environments must be Managed Environments for governance enforcement.
- Pipelines are only visible from development environments, not target environments.

Use Case #2: Commit changes and understand source control structure

Use Git integration to track deployment changes and understand how Power Platform solution components are organized in source control.

Use case	Value added	Estimated effort
Commit changes and understand source control structure	Track and structure – Use Git integration to maintain deployment history and understand how solution components are organized in source control.	5 minutes

Summary of tasks

In this section, you'll commit your deployment artifacts to source control and explore how Power Platform solutions are structured in Git repositories.

Scenario: Document your deployment process and understand how solution components are versioned and tracked in source control for future pipeline extensions.

Objective

Commit deployment artifacts to Git and understand the structure of unpacked Power Platform solutions in source control.

Step-by-step instructions

Access source control in Copilot Studio

1. Return to your DEV environment in Copilot Studio.
2. Open your solution and navigate to **Source control**.

3. Review any uncommitted changes from pipeline configuration or deployment preparation.

Commit pipeline deployment changes

4. Select any modified components that should be committed.
5. Add a meaningful commit message: “Pipeline deployment: Production-ready configuration with automated ALM”
6. Select **Commit** to save changes to your branch.

Explore solution structure in Azure DevOps

7. Navigate to your Azure DevOps project and browse to **Repos**.
8. Explore the **Solutions** folder structure:

```
Solutions/
├── [SolutionName]/
│   ├── botcomponents/
│   ├── bots/
│   ├── connectionreferences/
│   ├── environmentvariabledefinitions/
│   ├── publishers/
│   └── solutions/
```

Understand component organization

9. Examine key folders:
    - **ConnectionReferences/**: Contains connection reference definitions used by connectors, flows, and tools.
    - **EnvironmentVariables/**: Contains environment variable definitions and values
    - **Workflows/**: Contains Power Automate flows (if any)
    - **Other/Copilot/**: Contains Copilot Studio agents and components
    - **SolutionPackage/**: Contains the overall solution metadata
10. Open a component file to see the XML, YAML, or JSON structure that defines these components.
11. Notice how this structure enables:
    - Granular tracking of changes to individual components
    - Easy integration with other CI/CD tools and processes
    - Professional development workflows for larger teams

Review deployment history

12. Go to **Repos > Commits** to see your deployment history.
13. Compare commits to understand what changes between deployments.
14. Use the diff view to see exactly what components were modified.

Congratulations! You've mastered ALM deployment with pipelines and source control!

Test your understanding

Key takeaways:

- **Professional ALM made accessible** – Pipelines democratize sophisticated deployment processes while maintaining professional standards.
- **Source control enables extension** – The Git integration provides the foundation for advanced CI/CD scenarios when needed.
- **Structured component organization** – Understanding the folder structure helps with troubleshooting and enables pipeline extensions.

Challenge: Apply this to your own use case

- Plan how you might extend pipelines with Power Platform CLI for advanced scenarios.
- Design branching strategies that work with pipeline deployment workflows.
- Consider how to integrate pipelines with existing organizational CI/CD processes.

Summary of learnings

ALM golden rules reinforced:

Work in the context of solutions – Pipelines require properly structured solutions for deployment automation.

Create separate solutions only if you need to deploy components independently – Pipelines work best with cohesive solution packaging.

Use a custom publisher and prefix – Maintains clear ownership in automated deployment scenarios.

Use environment variables for settings and secrets that change across environments – Pipelines validate and update these automatically during deployment.

Export and deploy solutions as managed, unless setting up a dev environment – Pipelines automatically handle this, ensuring governance and preventing unauthorized changes.

Don't do customizations outside of dev – Managed Environments enforce this rule, blocking unmanaged customizations in target environments.

Consider automating ALM for source control and automated deployments – Pipelines provide this automation while remaining extensible for advanced scenarios.

> [!IMPORTANT] **Critical reminder about non-solution-aware settings:**

> These Copilot Studio settings require manual post-deployment configuration: * **Azure Application Insights settings** * **Manual authentication settings** * **Direct Line / Web channel security settings** * **Deployed channels** * **Sharing (with other makers, or with end-users)**
>
> Always include these in your post-deployment checklist!

> [!NOTE] **Managed Environment governance in your lab:** Your PROD environment uses Managed Environment governance to enforce that solutions are managed and unmanaged customizations are blocked. This ensures deployment integrity and prevents unauthorized changes outside of the pipeline process.

Conclusions and recommendations

Pipeline deployment excellence principles:

- **Democratize with governance** – Use pipelines to make ALM accessible to all makers while maintaining enterprise-grade security and control.
- **Validate early and often** – Leverage pipeline pre-deployment validation to catch issues before they impact target environments.
- **Automate the repeatable, manage the exceptions** – Let pipelines handle standard deployment tasks while maintaining checklists for non-solution-aware settings.
- **Extend when needed** – Start with pipelines for core deployment, then extend with CLI and CI/CD tools for advanced scenarios.
- **Trust but verify** – Use Managed Environments to enforce governance while providing makers the flexibility to deploy through approved channels.

By mastering Power Platform pipelines, you've established a deployment process that combines the accessibility makers need with the governance and automation enterprises require. Your deployments are now predictable, auditable, and scalable—setting the foundation for confident production releases at any scale.
