Microsoft

Hands-On Lab: Agent ALM – Managing the Lifecycle of Intelligent Agents

Version: 1.0
Duration: 90–120 minutes
Level: Intermediate

Lab Overview

Microsoft Copilot Studio enables organizations to build intelligent agents that extend business processes and automate tasks. Application Lifecycle Management (ALM) ensures reliable deployments, proper governance, and efficient collaboration across development teams.

This lab demonstrates how to implement a complete ALM strategy for Copilot Studio agents using Power Platform solutions, deployment pipelines, and best practices.

Learning Objectives

By the end of this lab, you will be able to:

- Establish a proper environment strategy (Dev, Test, Prod).
- Create and manage solutions for agents.
- Export and import agents across environments.
- Configure deployment pipelines.
- Implement version control and component reusability.
- Test and validate agents before production deployment.

Prerequisites

- Power Platform license with Copilot Studio access.
- System Administrator or System Customizer role.
- Basic understanding of Copilot Studio authoring.
- Permissions to create multiple environments in the Power Platform Admin Center.

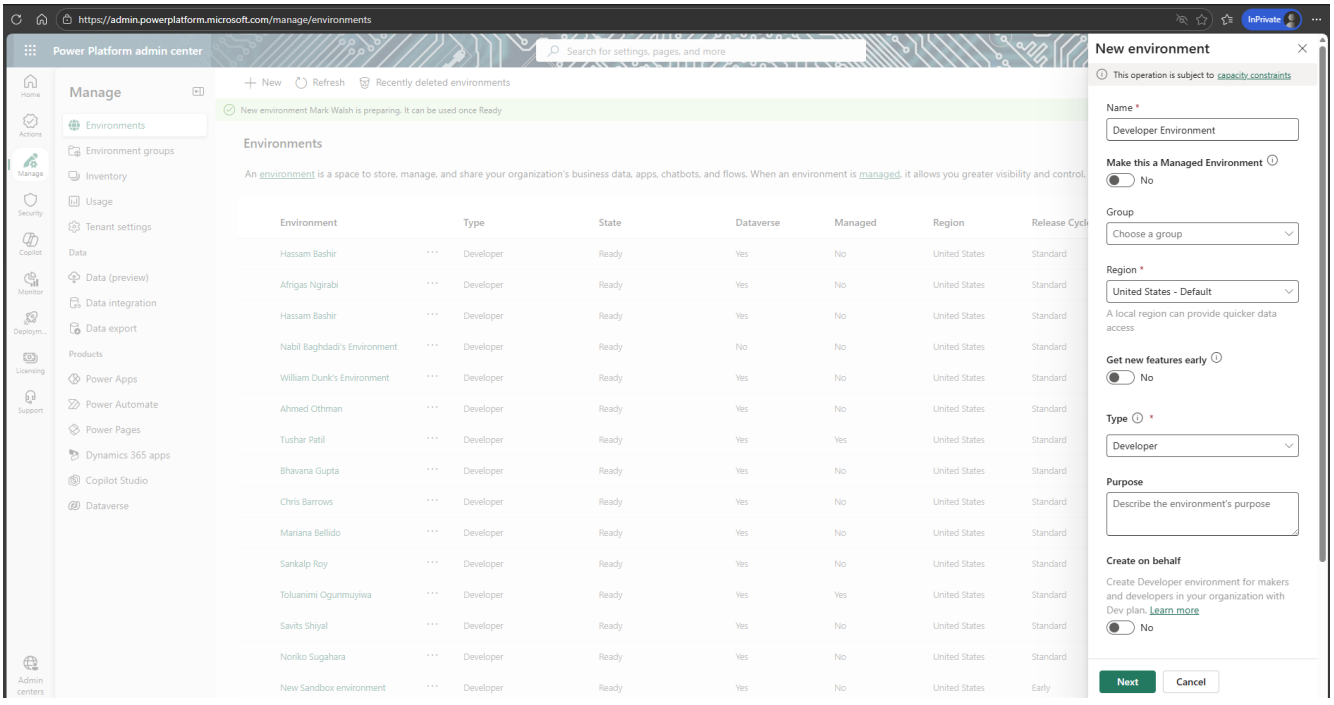
Microsoft

Part 1: Setup Environment Strategy for ALM

Objective: Establish a proper environment strategy with Development, Test, and Production environments following ALM best practices.

Step 1.1: Create Development Environment

1. Navigate to the Power Platform Admin Center.
2. Click on **Environments** in the left navigation pane.
3. Click **+ New** in the top menu bar.
4. Configure the environment with the following details:
   - **Name:** AgentTechALMSource
   - **Type:** Sandbox
   - **Region:** Select your local region
   - **Add a Dataverse data store:** Yes (This is required for Copilot Studio)
   - **Pay-as-you-go:** No (unless using an Azure subscription)
5. Click **Next**.
6. On the "Add Dataverse" tab, leave defaults and ensure **Security Group** is open or set to your preference.
7. Click **Save**.



Step 1.2: Create Test Environment

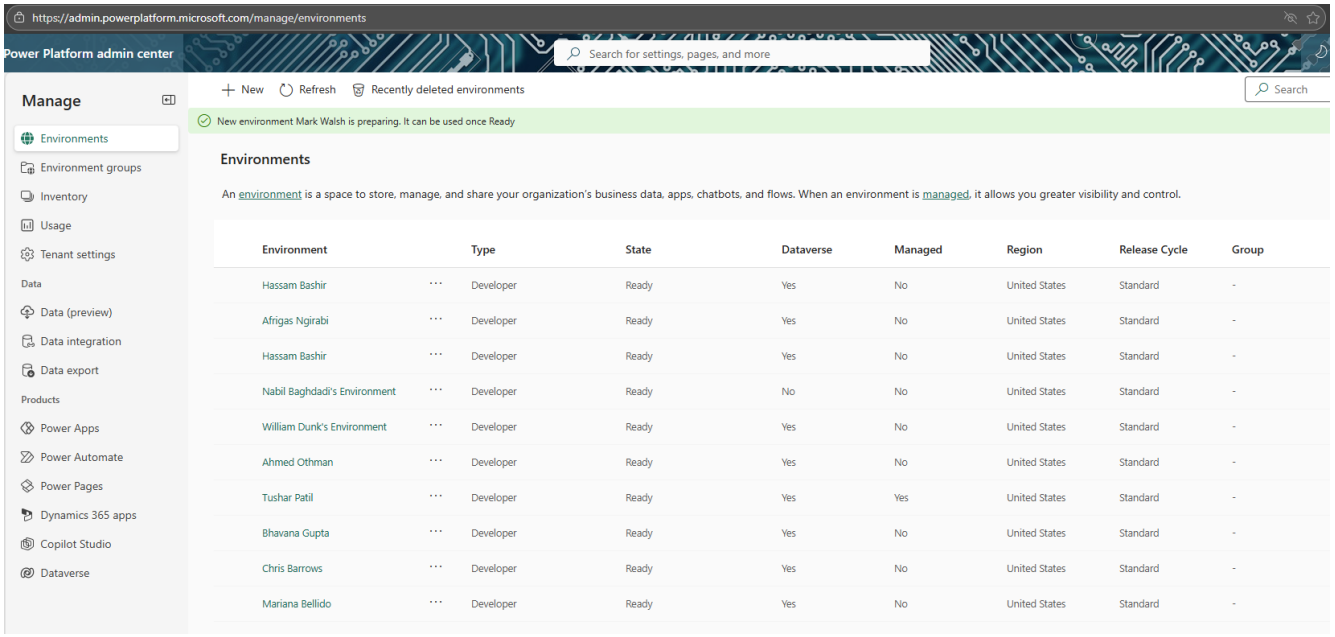
Repeat the process in Step 1.1 to create a QA/Testing environment:

- **Name:** AgentTechALMDestination
- **Type:** Sandbox
- **Add Dataverse data store:** Yes

Step 1.3: Create Production Environment (Optional)

Create the final Production environment:

- **Name:** Agent Production
- **Type:** Production
- **Add Dataverse data store:** Yes



Step 1.4: Configure Security Groups (Optional)

1. For each environment created, select the environment name.
2. In the "Details" pane, click **Edit**.
3. Under **Security group**, click the pencil icon or **+ Select**.
4. Assign the appropriate Azure AD Security Group (e.g., "Dev Team" for Development, "QA Team" for Test).

5. Click **Save**.

> ✓ **Best Practice:** Always restrict environment access using Azure AD security groups. Only developers should have access to Development, QA staff to Test, and only Admins/End Users to Production.

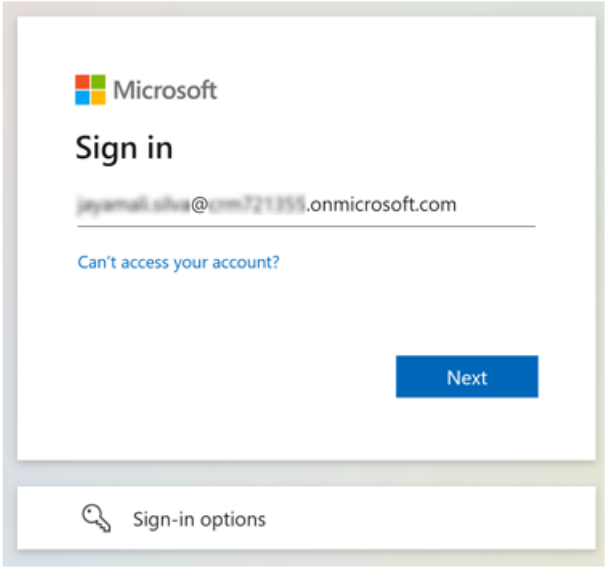
Reference: Control user access to environments: security groups and licenses

Part 2: Create Custom Solutions in Copilot Studio

Objective: Learn to create and manage custom solutions for proper ALM implementation instead of using the Default Solution.

Step 2.1: Access Copilot Studio in Development Environment

1. Navigate to Copilot Studio.
2. Enter the tenant user name, click Next
3. Enter the provided password, click Sign in
4. If prompted, choose whether to stay signed in.

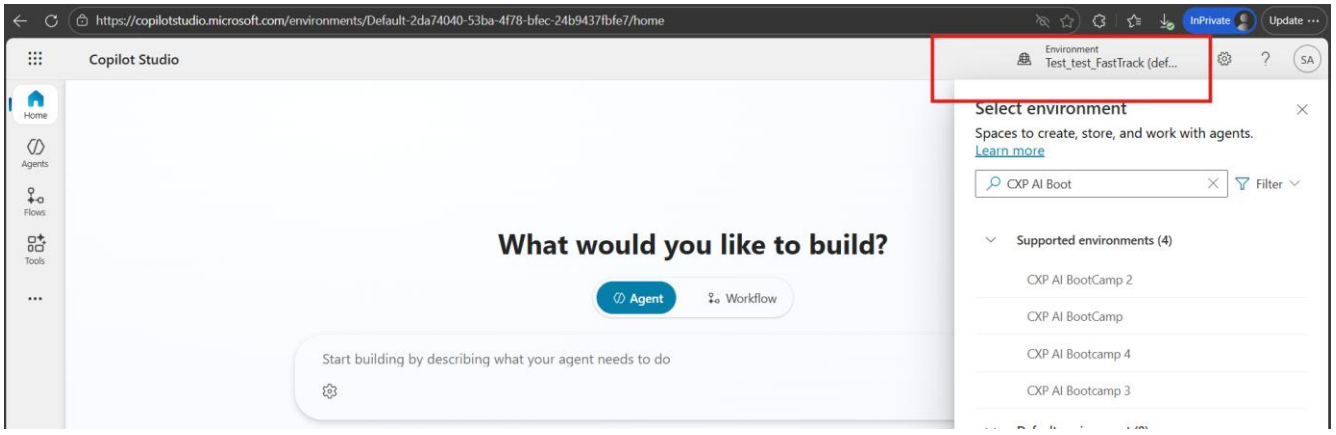


5. The first time you access Microsoft Copilot Studio, you'll be prompted to choose

your country/region. You can choose a value or leave the default option and click **Get Started**.

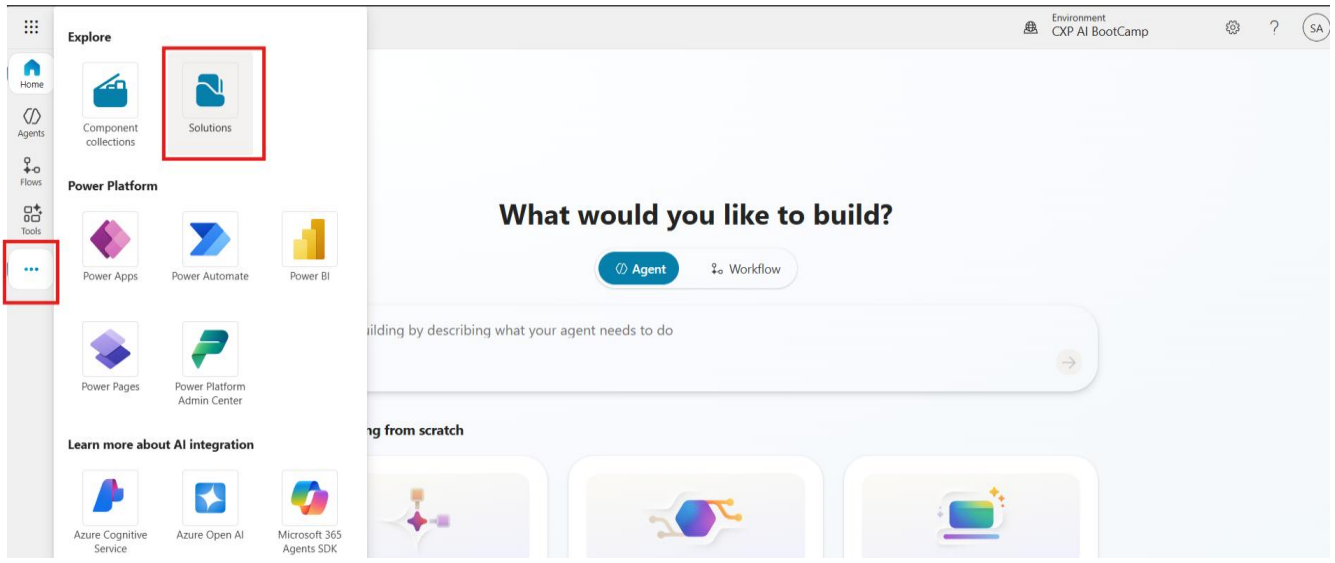
6. In the top-right corner, verify the environment selector and select the Dev environment.

7. Switch to the **Development** environment created in Part 1.



Step 2.2: Create a Custom Publisher

1. Navigate to the **Solutions** tab from the left navigation (accessed via "..." More options).



2. On the Solutions page, click **+ New solution**.
3. In the "New solution" pane, find the **Publisher** dropdown and click **+ New publisher**.
4. Configure the Publisher:
   - **Display name:** ALM
   - **Name:** ALM
   - **Prefix:** alm (This is critical for identifying your components)
   - **Choice value prefix:** 23088
5. Click **Save**.

> ℹ **Why Custom Publisher?** The publisher prefix (e.g., `alm_`) applies to every component you create (tables, variables, agents). This prevents naming conflicts with other solutions and makes it easy to identify ownership of components in a shared environment.

Step 2.3: Create Custom Solution

1. Back in the "New solution" pane:
   - **Display name:** ALMDemo
   - **Name:** ALMDemo (auto-populated)

- **Publisher:** Select **ALM** (created in previous step)
- **Version:** 1.0.0.0

2. Click **Create**.

Step 2.4: Set as Preferred Solution

1. In the Solutions list, verify "ALMDemo" is present.
2. Click the **Set preferred solution** toggle or button in the command bar.
3. Select **ALMDemo Solution**.

Note: This ensures any new agent created in this environment is automatically added to this solution.

Reference: Create a solution in Power Apps

Part 3: Build an Agent in Development Environment

Objective: Create a sample ALMDemo with topics, knowledge, and actions that will be deployed through the ALM process.

Step 3.1: Create the Agent

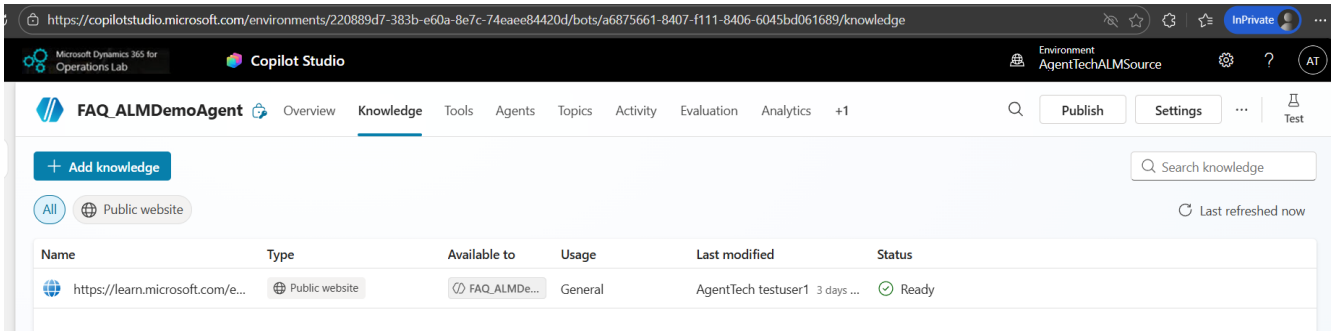
1. On the Copilot Studio Home page, click **+ Create**.
2. Select **New agent**.
3. Enter details:
   - **Name:** FAQ_ALMDemoAgent
   - **Language:** English (US)
4. Click **Create**.

Step 3.2: Configure Agent Settings

1. Once the agent loads, go to the **Settings** tab (gear icon).
2. Select **Agent details**.
3. Verify under "Solution" that it is associated with **ALMDemo Solution**.

Step 3.4: Add Knowledge Sources

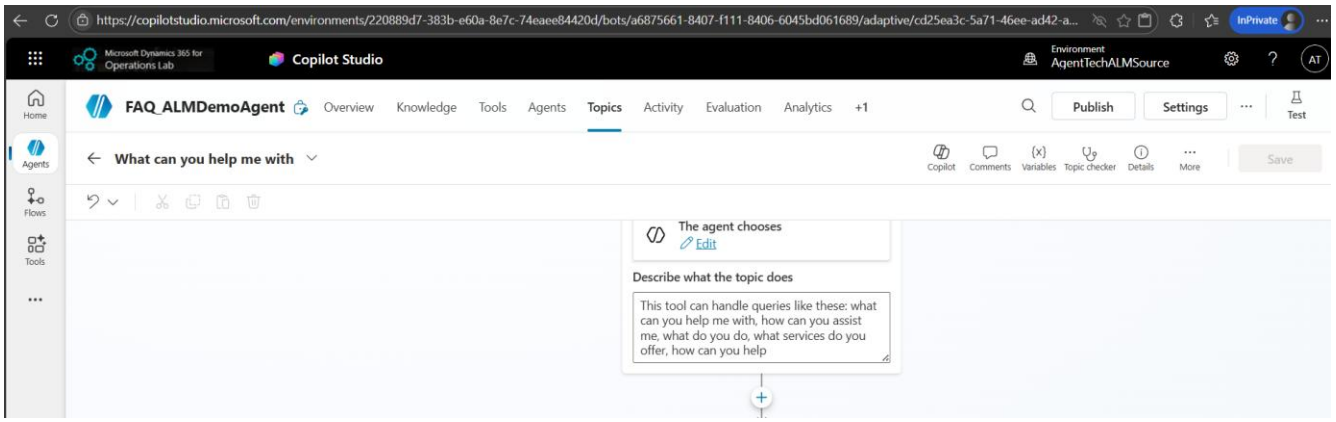
1. Open the **FAQ_ALMDemoAgent**.
2. Click on the **Knowledge** tab.
3. Click **+ Add knowledge**.
4. Select **Public websites**.
5. Enter a sample URL: https://learn.microsoft.com/en-us/microsoft-copilot-studio/
6. Click **Add**.



Step 3.5: Create Custom Topics

1. Go to the **Topics** tab.
2. You will see some default topics already created out of the box. Eg – goodbye , start over , thank you etc.
3. For adding a custom topic click **+ Add a topic** → **From blank**.
4. Name the topic: **What can you help me with**.
5. Add Trigger phrases:

   This tool can handle queries like these: what can you help me with, how can you assist me, what do you do, what services do you offer, how can you help

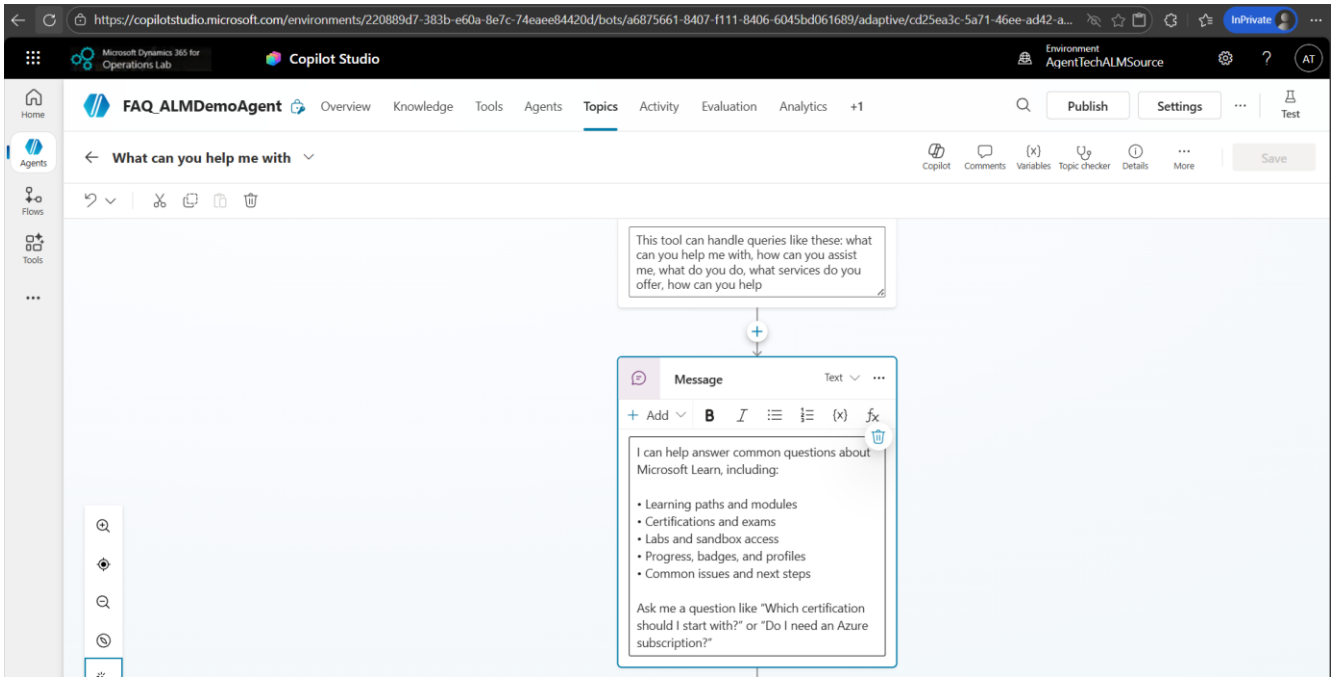


6. In the Message section add below description

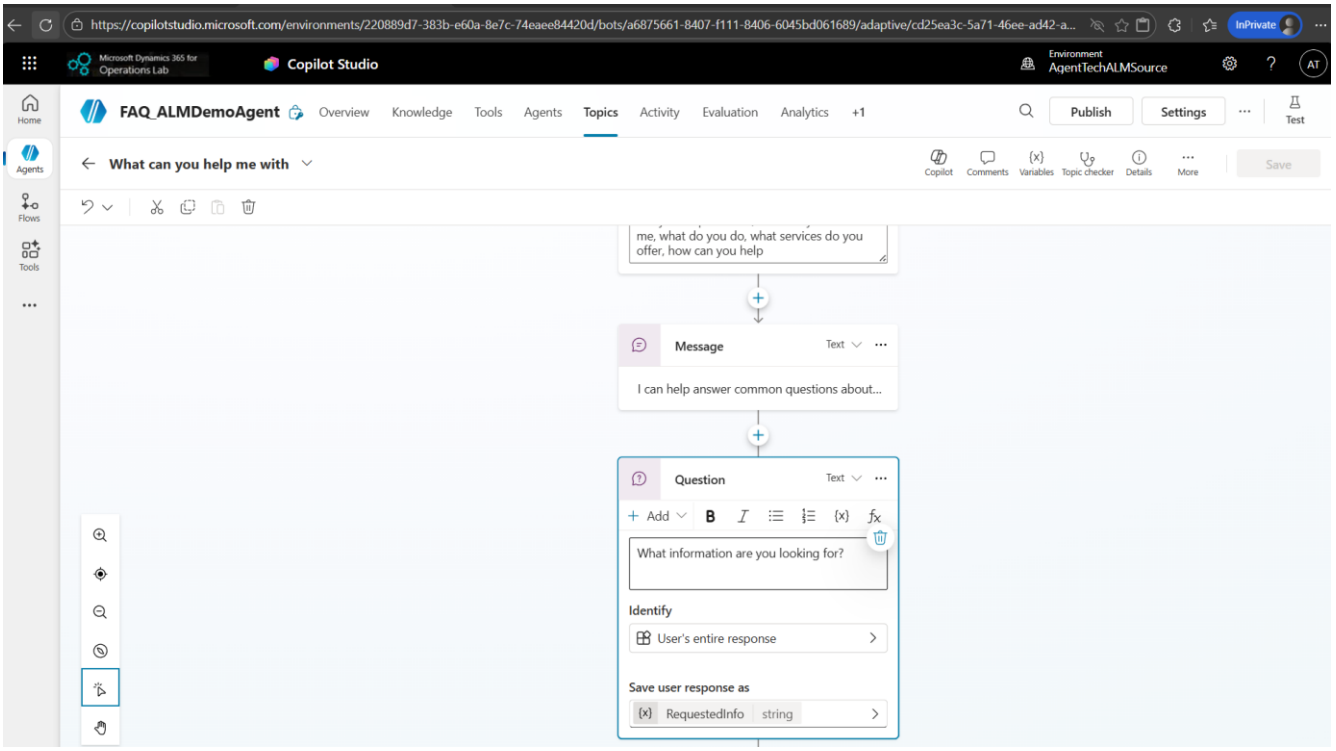
I can help answer common questions about Microsoft Learn, including

- *Learning paths and modules*
- *Certifications and exams*
- *Labs and sandbox access*
- *Progress, badges, and profiles*
- *Common issues and next steps*

Ask me a question like "Which certification should I start with?" or "Do I need an Azure subscription?

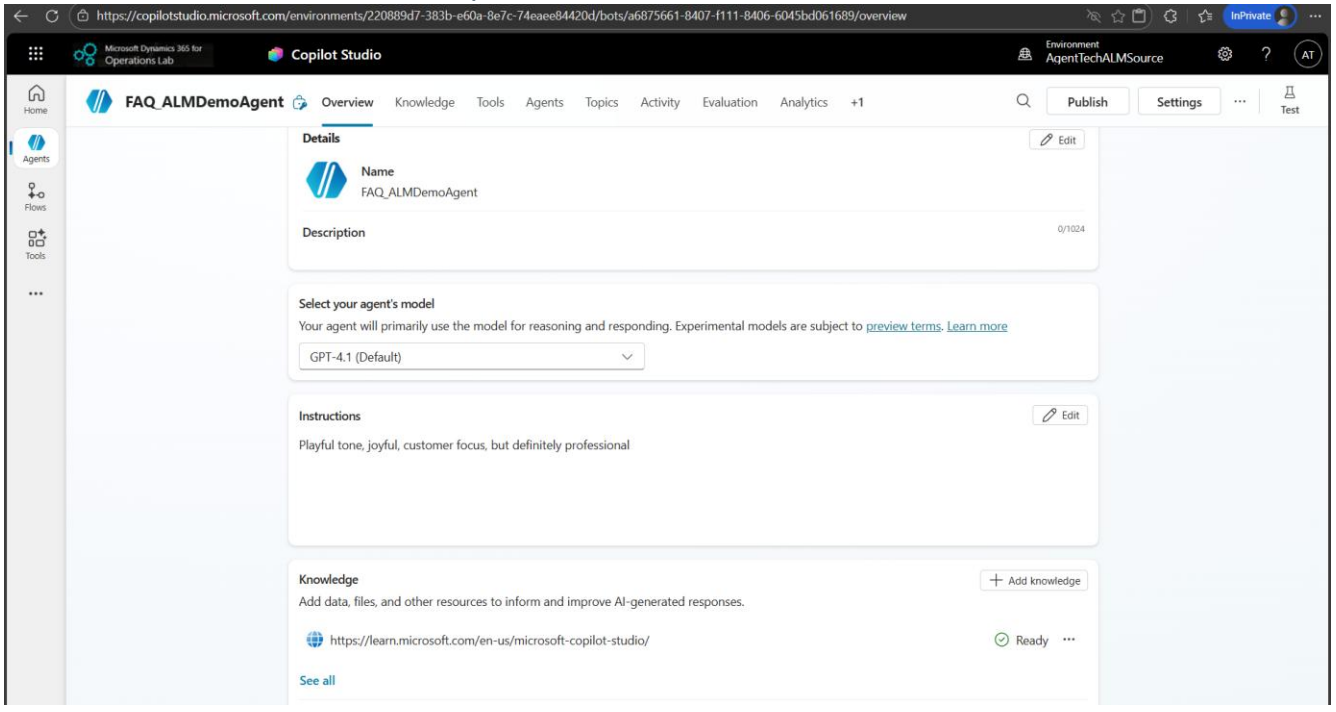


7. In the authoring canvas, add a **Question** node: "***What information are you looking for?*** . Identify as string. Save as variable.



8. Click **Save**.

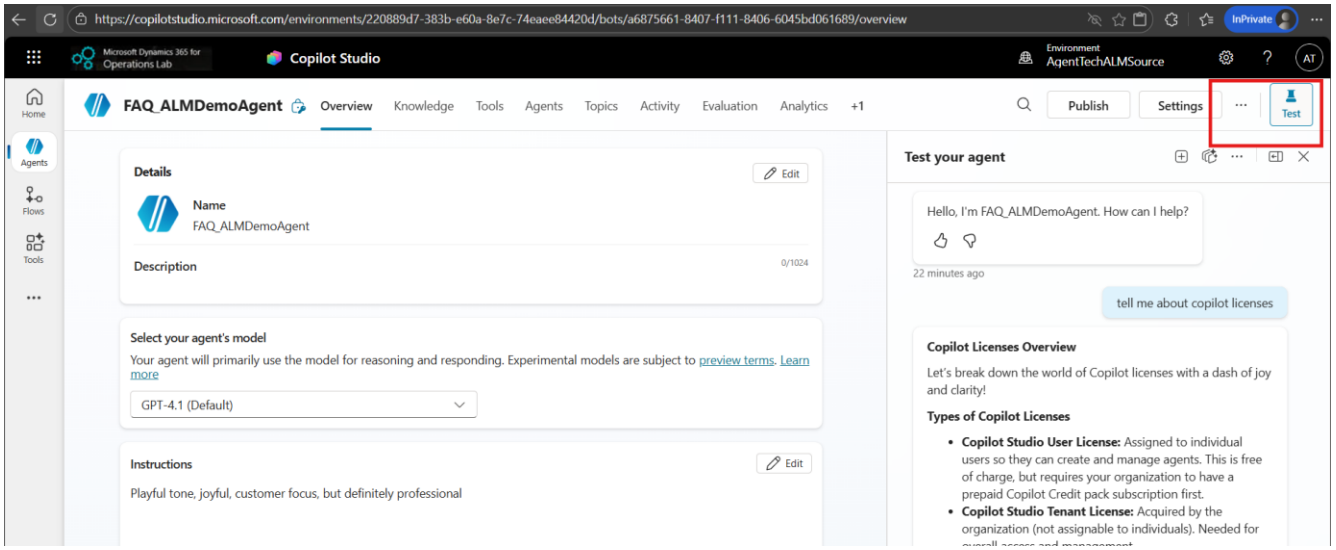
9. Add a tone and model to the agent as per need. Click on Overview and observe the updated model , instructions and knowledge source sections. All fields are editable and can be customized as per need.



Microsoft

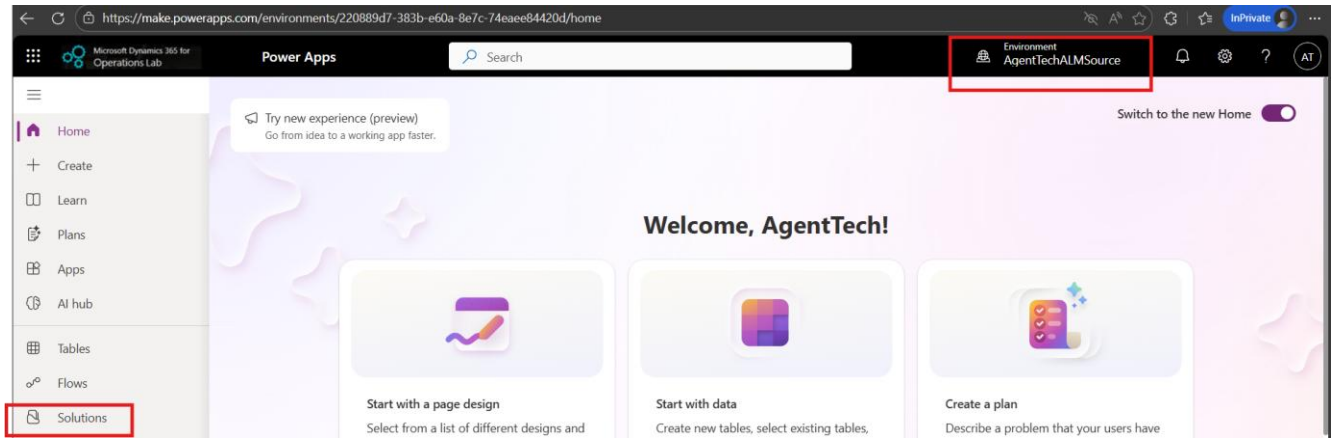
Step 3.7: Test the Agent

1. Click the **Test** button to open the test pane.
2. Type "I need time off".
3. Verify the agent triggers the topic.

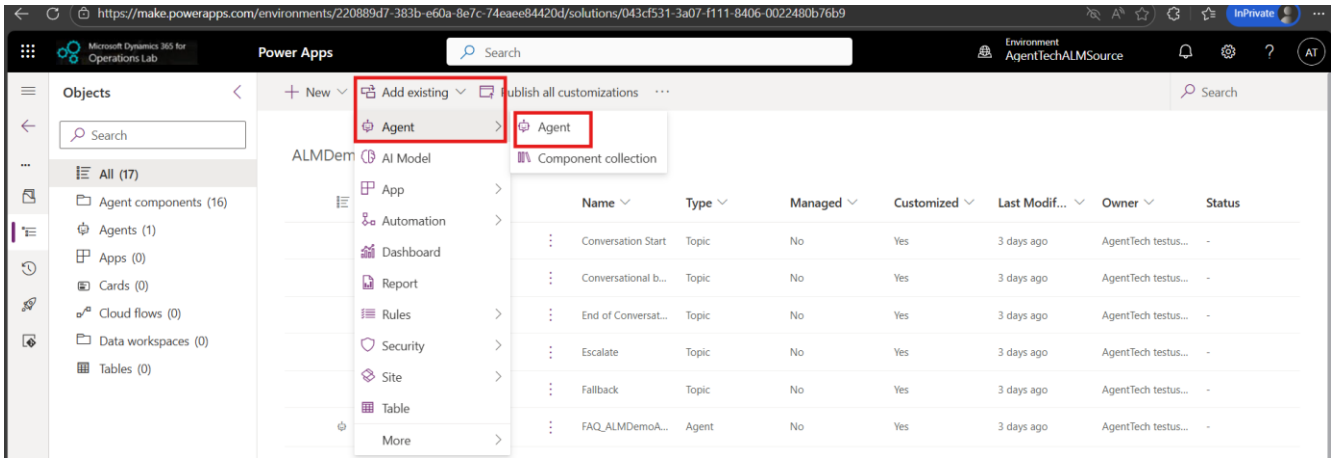


Step 3.8: Add agent into solution

1. Open https://make.powerapps.com
2. Select the correct environment



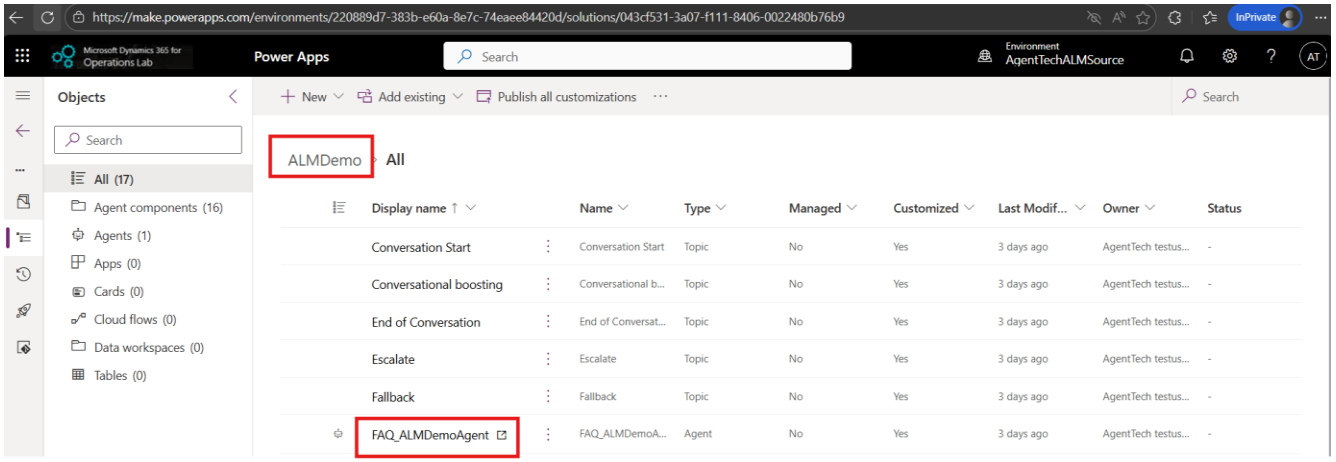
3. Click to **Solutions → ALMDemo**
4. Click on Add Existing -> Agent



5. Select FAQ_DemoALMAgent from the list --> Add

Step 3.9: Verify Solution Components

1. Return to **Solutions → ALMDemo Solution**.
2. Verify it contains agent.



Reference: Create and manage topics in Copilot Studio

Part 4: Export and Import Solutions

Objective: Learn the manual ALM process of exporting from Dev and importing to Test.

Step 4.1: Export Unmanaged Solution (Backup)

1. In the **Agent Development** environment, go to **Solutions**.
2. Select **ALMDemo Solution**.
3. Click **Export solution**.
4. Click **Next**.
5. Select **Unmanaged**.
6. Click **Export**.
7. Download the zip file (e.g., as `ALMDemo_1_0_0_0.zip`). store this your source code backup.

Step 4.2: Export Managed Solution

1. Repeat the export process.
2. On the export settings step, verify version is `1.0.0.0`.
3. Select **Managed**.
4. Click **Export**.
5. Download the zip file (e.g.,). `ALMDemo_1_0_0_0_managed.zip`

> ⚠ **Managed vs. Unmanaged:** Always deploy **Managed** solutions to Test and Production. Managed solutions prevent direct editing in the target environment, ensuring the "single source of truth" remains in Development.

Step 4.3: Prepare Test Environment

1. Switch environment to **Agent Testing** using the environment selector.
2. Navigate to **Solutions**.

Step 4.4: Import Solution to Test Environment

1. Click **Import solution**.
2. Browse and select the `ALMDemo_1_0_0_0.zip`
3. Click **Next**.
4. Review solution details and click **Next**.
5. Click **Import**.

Step 4.5: Verify Import Success

1. Wait for the banner "Solution imported successfully".
2. Open the solution in the list. Note that you cannot edit components directly (indicated by a lock icon).

Step 4.6: Test Agent in Test Environment

1. Navigate to **Copilot Studio** in the **AgentTechALMDestination** environment.
2. Open **ALMDemo**.
3. Test the "PTO Request" topic.
4. Verify it uses the Test environment variables (e.g., check flow run history or email recipient).

Reference: Export and import solutions in Copilot Studio

Part 5: Setup Deployment Pipelines (Power Pipelines)

Objective: Configure Power Platform Pipelines to automate the manual export/import process.

Step 5.1: Create Pipeline Configuration

1. Go to **Power Platform Admin Center** → **Pipelines** (in the left nav, you may need to install the Pipelines application first if not present).
2. Click **+ New pipeline**.

3. Name: **Agent ALM Pipeline**.
4. Description: "Pipeline for HR Agent from Dev to Test to Prod".

Step 5.2: Link Development Environment

1. Open the created pipeline.
2. Click **+ Add environment**.
3. Select **Agent Development**.
4. Set Environment Type: **Development Environment**.
5. Click **Save**.

Step 5.3: Link Test Environment

1. Click **+ Add environment**.
2. Select **Agent Testing**.
3. Set Environment Type: **Target Environment**.
4. Previous Deployment Stage: (None/Development).
5. Click **Save**.

Step 5.4: Link Production Environment

1. Click **+ Add environment**.
2. Select **Agent Production**.
3. Set Environment Type: **Target Environment**.
4. Previous Deployment Stage: **Agent Testing**.
5. Under **Pre-deployment Step**, enable **Pre-deployment approval**.

Step 5.5: Configure Pipeline Settings

1. Navigate back to the solution in **Agent Development** environment in Copilot Studio/Power Apps.
2. Refresh the page to recognize pipeline configuration.

Step 5.6: Deploy Using Pipeline

1. In Copilot Studio (Dev), go to **Solutions**.
2. Select **ALMDemo Solution**.
3. Click **Pipelines** (rocket icon) in the command bar.
4. Select **Agent ALM Pipeline**.
5. Click **Deploy here** under the **Agent Testing** stage.
6. Choose **Deploy immediately** and click **Next**.
7. Confirm deployment settings (Environment Variables).
8. Click **Deploy**.

Step 5.7: Monitor Pipeline Deployment

Watch the progress indicator. Once complete, the status will change to "Succeeded".

Step 5.8: Deploy to Production (with Approval)

1. In the Pipelines view, click **Deploy here** under **Agent Production**.
2. Complete the wizard.
3. The deployment will pause at "Pending Approval".
4. Admin/Approver must navigate to the "Deployment Dashboard" app to approve the request.
5. Once approved, the deployment proceeds.

> ✓ **Benefits:** Pipelines provide an audit trail of who deployed what and when, enforce approval gates, and remove the need to manually handle zip files.

Reference: Overview of pipelines in Power Platform

Part 6: Advanced ALM with Azure DevOps (Optional)

Objective: Implement enterprise-grade ALM using Azure DevOps for source control and CI/CD.

Step 6.1: Setup Azure DevOps Project

1. Log in to Azure DevOps.
2. Click **+ New Project**.
3. Name: **CopilotAgentALM**.
4. Visibility: **Private**.

Step 6.2: Install Power Platform Build Tools

1. Go to **Organization Settings** → **Extensions**.
2. Browse Marketplace and search for **Power Platform Build Tools**.
3. Install it for your organization.

Step 6.3: Create Service Connection

1. In Project Settings, go to **Service connections**.
2. Click **New service connection** → **Power Platform**.
3. Enter the **Server URL** (Environment URL for Dev).
4. Enter Tenant ID, App ID, and Client Secret (Requires Azure AD App Registration).
5. Name it **PowerPlatform-Dev**.
6. Repeat for Test and Prod.

> ⚠ **Prerequisite:** You must register an App in Azure Entra ID and grant it System Administrator access in your Power Platform environments to use Service Connections.

Step 6.4: Create Azure Repos Repository

1. Go to **Repos**.
2. Initialize with a README.

Step 6.5: Create Export Pipeline

1. Go to **Pipelines** → **New Pipeline**.
2. Select **Azure Repos Git** → **Starter Pipeline**.
3. Copy the following YAML configuration:

```
trigger:
- main

pool:
  vmImage: 'windows-latest'

variables:
  SolutionName: 'ALMDemo'
  SourceEnvironmentUrl: '<Org URL>'
  ServiceConnection: '<SourceConnection>' # Update with your actual service connection name

steps:
- task: PowerPlatformToolInstaller@2
  displayName: 'Install Power Platform Tools'
- checkout: self
  persistCredentials: true  # This is the "Allow scripts to access OAuth token" setting

  # Export Unmanaged
- task: PowerPlatformExportSolution@2
  displayName: 'Export Solution'
  inputs:
    authenticationType: 'PowerPlatformSPN'
    PowerPlatformSPN: '$(ServiceConnection)'
    SolutionName: '$(SolutionName)'
    SolutionOutputFile: '$(Build.ArtifactStagingDirectory)/$(SolutionName).zip'
    Managed: false
    AsyncOperation: true
```

```
# Export Managed
- task: PowerPlatformExportSolution@2
  displayName: 'Export Solution'
  inputs:
    authenticationType: 'PowerPlatformSPN'
    PowerPlatformSPN: '$(ServiceConnection)'
    SolutionName: '$(SolutionName)'
    SolutionOutputFile: '$(Build.ArtifactStagingDirectory)/$(SolutionName)_managed.zip'
    Managed: true
    AsyncOperation: true

- task: PowerPlatformUnpackSolution@2
  displayName: 'Unpack Solution'
  inputs:
    SolutionInputFile: '$(Build.ArtifactStagingDirectory)/$(SolutionName).zip'
    ManagedSolutionInputFile: '$(Build.ArtifactStagingDirectory)/$(SolutionName)_managed.zip' # Point to M
anaged
    SolutionTargetFolder: '$(Build.SourcesDirectory)/solutions/$(SolutionName)'
    SolutionType: 'Both'
    OverwriteFiles: true

- task: CmdLine@2
  displayName: 'Commit unpacked solution to repo'
  env:
    MY_TOKEN: $(System.AccessToken) # Map the secret token to an env variable
  inputs:
    script: |
      # Configure identity
      git config user.email "mudit.agarwal23@live.com"
      git config user.name "Mudit Agarwal"

      # Re-configure the remote URL to include the token for authentication
      git remote set-url origin https://x-token-
auth:$MY_TOKEN@dynamicintegration.visualstudio.com/DynamicIntegration/_git/ALMDemo

      git add solutions/$(SolutionName)

      # Only commit if there are changes to avoid "nothing to commit" errors
      git diff-index --quiet HEAD || git commit -m "Exported and unpacked solution $(SolutionName)"

      git push origin HEAD:main
```

Step 6.6: Create Import Pipeline

1. Go to **Pipelines** → **Releases**.
2. Create a new Release Pipeline.
3. Copy the following YAML configuration:

```
trigger:
- main  # This will run whenever changes are pushed to the main branch

pool:
  vmImage: 'windows-latest'

variables:
  SolutionName: 'ALMDemo'
  # Update these to match your specific Destination environment details
  DestServiceConnection: 'DestinationEnvironment'

steps:
- task: PowerPlatformToolInstaller@2
  displayName: 'Install Power Platform Tools'

# 1. Pack the source code from the repo into a Managed zip file
- task: PowerPlatformPackSolution@2
  displayName: 'Pack Solution as Managed'
  inputs:
    SolutionSourceFolder: '$(Build.SourcesDirectory)/solutions/$(SolutionName)'
    SolutionOutputFile: '$(Build.ArtifactStagingDirectory)/$(SolutionName)_managed.zip'
    SolutionType: 'Managed'

# 2. Import the solution into the Destination Environment
- task: PowerPlatformImportSolution@2
  displayName: 'Import Solution to Destination'
```

```
 inputs:
   authenticationType: 'PowerPlatformSPN'
   PowerPlatformSPN: '$(DestServiceConnection)'
   SolutionInputFile: '$(Build.ArtifactStagingDirectory)/$(SolutionName)_managed.zip'
   OverwriteUnmanagedCustomizations: true
   PublishCustomizations: true

# 3. (Optional) Publish all customizations to ensure UI updates are visible
- task: PowerPlatformPublishCustomizations@2
  displayName: 'Publish All Customizations'
  inputs:
   authenticationType: 'PowerPlatformSPN'
   PowerPlatformSPN: '$(DestServiceConnection)'
```

Step 6.8: Configure Deploy to Production Stage

1. Add a Stage: **Deploy to Prod**.
2. Click the "Lightning bolt" icon on the connector to enable **Pre-deployment approvals**.
3. Add yourself as the approver.
4. Add Task: **Power Platform Import Solution** connected to **PowerPlatform-Prod**.

Step 6.9: Run End-to-End Pipeline

1. Run the Build Pipeline manually.
2. Watch it export from Dev and commit to the repo.
3. Watch the Release Pipeline automatically trigger deployment to Test.
4. Approve the deployment to Production.

> **Enterprise Benefit:** Azure DevOps provides granular version control (diffing files), automated testing integration, and compliance traceability that native pipelines may not fully offer yet.

Reference: Microsoft Power Platform Build Tools for Azure DevOps

Part 7: Component Collections and Reusability

Objective: Create reusable component collections to share topics across multiple agents.

Step 7.1: Create Component Collection Solution

1. In Dev, create a new Solution named **HR Core Topics**.

Step 7.2: Create Reusable Topics

1. Create a generic agent (temporary) to build topics.
2. Create topics meant for reuse:
   - **Employee Verification**
   - **Escalation Handler**
   - **Greeting Messages**

Step 7.3: Export Topics as Components

1. In Copilot Studio, go to the **Topics** list.
2. Select the topics.
3. Look for an option to **Add to solution** or manage within the Solution view in Power Apps.
4. Ensure these topics are inside the **HR Core Topics** solution.

Step 7.4: Use Components in Other Agents

1. Open a different agent (e.g., "IT Support").
2. In Topics, select **Add existing** or **Import from solution**.
3. Select topics from **HR Core Topics**.

Step 7.5: Deploy Component Collections

1. Deploy the **HR Core Topics** solution to Test/Prod first.
2. Then deploy agents that depend on it.

Reference: Create reusable component collections

Part 8: Monitoring and Governance

Objective: Implement monitoring, analytics, and governance for deployed agents.

Step 8.1: Setup Azure Application Insights

1. In Azure Portal, create an **Application Insights** resource.
2. Copy the **Connection String**.
3. In Copilot Studio (Agent Settings → Advanced), paste the connection string.
4. This enables deep telemetry logging.

Step 8.2: View Analytics Dashboard

1. In Copilot Studio, click the **Analytics** tab.
2. Review Sessions, Engagement Rate, and Resolution Rate.

Step 8.3: Create DLP Policies

1. In Power Platform Admin Center, go to **Data policies**.
2. Create a policy blocking "Non-business" connectors (e.g., Twitter, RSS) for the Agent environments.

Step 8.4: Enable Managed Environments

1. In Admin Center, select **Agent Production**.
2. Click **Enable Managed Environments**.
3. Enable **Usage insights** and **Maker welcome content**.

Reference: Establish an ALM strategy for Copilot Studio

Microsoft

Part 9: Best Practices and Troubleshooting

1. Environmental Strategy

- ☐ ☑ DO Maintain at least 3 environments (Dev, Test, Prod).
- ☐ ☑ DO Use Sandbox type for Dev/Test and Production type for Prod.
- ☐ ☑ DO Apply security groups to all environments.
- ☐ ☑ DO Use a dedicated "Validation" environment for platform updates.
- ☐ ❌ DON'T Customize directly in Production.

2. Solution Management

- ☐ ☑ DO Always work within custom solutions.
- ☐ ☑ DO Use custom publishers with meaningful prefixes.
- ☐ ☑ DO Use environment variables for all external connections.
- ☐ ☑ DO Segment solutions by business function (HR, IT, Sales).
- ☐ ❌ DON'T Use the Default Solution for production agents.
- ☐ ❌ DON'T Mix managed and unmanaged layers in Production.

3. Version Control

- ☐ ☑ DO Increment version numbers with each release (Semantic Versioning).
- ☐ ☑ DO Commit solution exports to Git source control.
- ☐ ☑ DO Use solution comments to track changes.
- ☐ ☑ DO Back up unmanaged solutions before major refactoring.

4. Testing Strategy

- ☐ ☑ DO Test thoroughly in Dev environment first.
- ☐ ☑ DO Have QA team validate in Test environment.
- ☐ ☑ DO Perform UAT with pilot users before Production deployment.
- ☐ ☑ DO Test edge cases and fallback topics.
- ☐ ❌ DON'T Skip the Test environment even for "small fixes".

5. Deployment Process

- ☐ ☑ DO Use pipelines or automation for consistency.
- ☐ ☑ DO Require approvals for Production deployments.
- ☐ ☑ DO Automate connection reference mapping.
- ☐ ☑ DO Schedule deployments during low-traffic windows.
- ☐ ❌ DON'T Manually edit flows in Production to "fix" connections.

6. Common Issues and Solutions

1. Missing Dependencies on Import

Issue: Import fails because a Flow or Connector is missing.
Solution: Ensure all child components (flows, connection references) are added to the solution in Dev. Check "Add required components" in solution viewer.

2. Environment Variables Not Updating

Issue: The agent still points to Dev SharePoint after import to Test.
Solution: During import, ensure you explicitly entered the new values. If missed, go to the Default Solution in Test, filter by Environment Variables, and edit the "Current Value".

3. Agent Not Appearing After Import

Issue: Solution imports success, but Agent list is empty.
Solution: Check the "Chatbots" or "Agents" filter in the Solution view. Sometimes browser cache needs clearing. Ensure the user has the "System Customizer" role.

4. Pipeline Deployment Fails

Issue: Generic error during pipeline run.
Solution: Open "Run History" in the Pipeline app. Common causes: Target environment has no capacity, or the deploying user lacks permissions in the target environment.

5. Power Automate Flows Not Working After Import

Issue: Flows fail to trigger from the agent.
Solution: Flows are often turned "Off" by default after import. Navigate to the Solution in Target, find the Flow, and turn it "On". Check Connection References are valid.

6. Troubleshooting Commands

- **Check Solution Health:** Use the "Solution Checker" icon in the solution command bar to run static analysis.
- **Verify Component Dependencies:** Select a component → Advanced → Show dependencies.
- **Export/Import Logs:** Always download the log file if an import fails; the XML contains the specific error code.

Microsoft

Additional Resources

- Application Lifecycle Management (ALM) with Power Platform
- Establish an ALM strategy for Copilot Studio
- Solution management in Copilot Studio
- Export and import solutions
- Power Platform pipelines
- Power Platform Build Tools for Azure DevOps
- GitHub Actions for Power Platform
- Environment variables overview
- Solution concepts for ALM
- Git integration for Power Platform