

Understand Azure Databricks Integrations

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/1-introduction>

Introduction

Completed

Azure Databricks excels at large-scale data processing and machine learning, but modern data solutions require integration with business intelligence tools, application frameworks, and AI services. Understanding how Azure Databricks connects with the **Microsoft ecosystem** helps you design comprehensive solutions that leverage the strengths of each platform.

Microsoft provides multiple integration points with Azure Databricks. You can connect **Microsoft Fabric** for unified analytics, publish data to **Power BI** for interactive reporting, develop locally with **Visual Studio Code** while executing on remote clusters, build applications with **Power Platform**, create AI agents in **Copilot Studio**, implement governance with **Microsoft Purview**, and orchestrate AI workflows with **Microsoft Foundry**. These integrations maintain your governance policies while extending data access across your organization.

The key to successful integration lies in choosing the right pattern for each scenario. **Security models** vary—some pass through individual user credentials to enforce fine-grained permissions, while others use **service principals** for consistent access. Each integration preserves **Unity Catalog** as the central governance layer, but enforcement mechanisms differ based on the consuming service's architecture.

In this module, you explore seven integrations between Azure Databricks and Microsoft services. You learn how each works, when to apply it, and what security considerations matter while maintaining the governance and quality standards your organization requires.

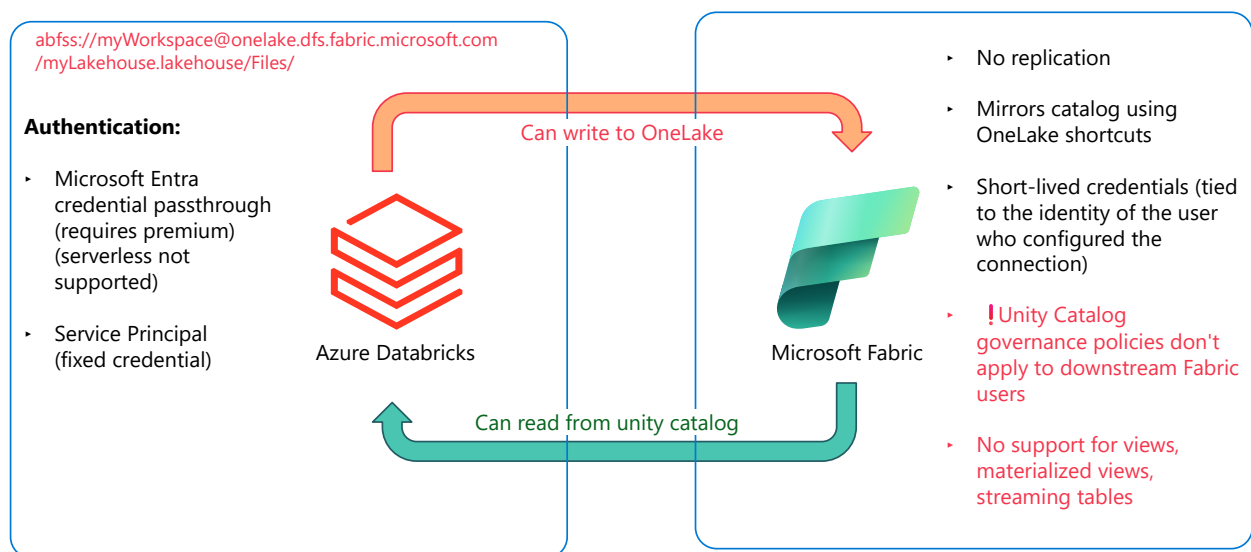
2. Understand integration with Microsoft Fabric

Understand integration with Microsoft Fabric

Completed

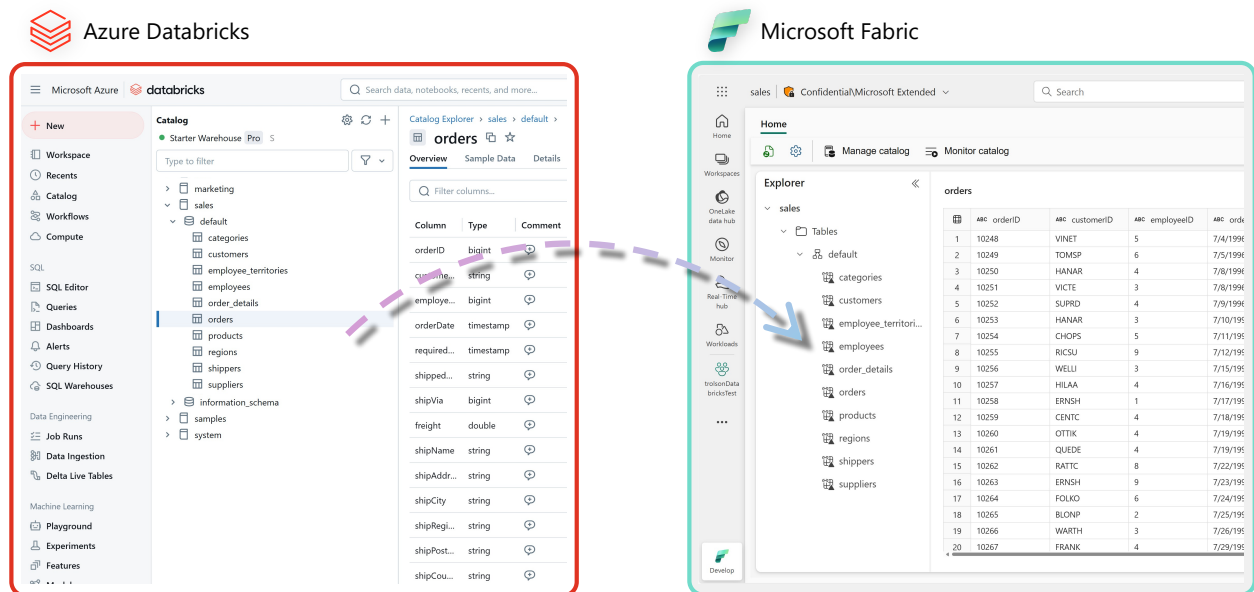
Microsoft Fabric and Azure Databricks complement each other in modern data architectures. Both platforms excel at data engineering and advanced analytics, with Azure Databricks specializing in large-scale data processing and machine learning, while Fabric offers a unified platform with integrated business intelligence and reporting capabilities. Understanding how these platforms integrate helps you design solutions that leverage the strengths of both.

The integration between Azure Databricks and Microsoft Fabric works **bidirectionally**. Fabric can **read** data registered in Unity Catalog, and Azure Databricks can **write** data to OneLake. Each integration pattern serves different scenarios and comes with specific considerations for governance and security.



Access Unity Catalog data from Fabric

Fabric can read tables registered in Unity Catalog **without replicating the data**. When you create a **Mirrored Azure Databricks Catalog** item in Fabric, it mirrors the catalog structure and automatically creates **OneLake shortcuts** for each table. These shortcuts point to the Delta tables stored in Azure Data Lake Storage. The mirroring process doesn't copy your data. Instead, Fabric uses Unity Catalog's open APIs to obtain credentials for accessing the underlying storage paths.



The authentication mechanism relies on **short-lived credentials** that Unity Catalog provides to Fabric. These credentials refresh every hour and can be revoked through Unity Catalog at any time. This approach maintains data in a **single location** while allowing Fabric users to query it.

However, this integration has an **important security implication**. Fabric engines perform authorization using the identity of the user who configured the connection, **not the identity of users who query the data**. **Unity Catalog governance policies don't apply to downstream Fabric users**. Once a table is exposed in Fabric, any Fabric user with access to the connection can query it, regardless of Unity Catalog permissions. Evaluate this against your organization's security requirements.

The user who configures the Fabric connection must have the **EXTERNAL USE SCHEMA privilege** on the schemas in Unity Catalog that contain the tables to share. This privilege allows Unity Catalog to issue credentials for the underlying storage. After the connection is established, downstream Fabric users don't need this privilege.

Several **limitations** apply to this integration pattern. Fabric can't access views, materialized views, streaming tables, Delta Sharing catalogs, or tables with row-level filters or column masks. **Only Delta format tables are supported**. The integration doesn't work with workspaces that use private endpoints or IP access lists, and Unity Catalog lineage doesn't track operations performed in Fabric.

Write data to OneLake from Databricks

Azure Databricks can write data directly to Microsoft Fabric lakehouses in OneLake. This integration pattern supports scenarios where you process and transform data in Databricks, then make it available in Fabric for analytics and reporting.

To connect Azure Databricks to OneLake, you use the **Azure Blob Filesystem (ABFS) driver** with OneLake endpoints. The connection path follows the format:

```
abfss://myWorkspace@onelake.dfs.fabric.microsoft.com/myLakehouse.lakehouse/Files/
```

This path structure identifies your Fabric workspace and lakehouse, allowing Databricks to read from and write to specific locations.

You can authenticate to OneLake through **two methods**. With **Microsoft Entra credential passthrough**, user identities flow through to OneLake, but this requires a **premium Azure Databricks workspace** and proper cluster configuration with **ADLS credential passthrough enabled**. Alternatively, you can authenticate using a **service principal**, which uses a fixed credential rather than individual user identities. Service principals work with both traditional clusters and Azure Databricks serverless compute, providing more flexibility for automated workflows and job execution.

Note

Credential passthrough is not supported with serverless compute

The workflow is straightforward. You load data in Databricks, apply transformations using Spark, and write the results to OneLake using standard Spark write operations. Once data lands in your Fabric lakehouse, it becomes immediately available to Power BI, data warehouses, and other Fabric services without additional data movement.

This integration supports **serverless compute**, allowing you to run workloads without provisioning clusters. When using serverless compute, you must use service principal authentication and ensure your code doesn't modify unsupported Spark configuration properties. The service principal needs appropriate workspace role assignments in Fabric to write data.

Integration scenarios

These integration patterns enable several practical scenarios for organizations using both platforms. You can build **data pipelines** where Databricks handles complex transformations and machine learning workloads, then publishes curated datasets to Fabric for business analytics. Data engineers work in Azure Databricks while business analysts access the results through familiar Fabric tools.

With Fabric reading from Unity Catalog, you can make specific datasets available for reporting **without duplicating storage**. Business users create Power BI reports that query data managed in Unity Catalog, maintaining a **single source of truth**. This approach applies when you have data governance established in Unity Catalog and want to extend access to Fabric users for specific tables.

For scenarios requiring **real-time or near-real-time** analytics, Databricks streaming jobs can write processed data to OneLake continuously. Fabric workloads can then consume this data as it arrives. This pattern separates the responsibilities of data engineering and business intelligence while keeping them connected through shared storage.

When you need to combine data from multiple sources, you can use Azure Databricks to **join and transform** data from various systems, then land the integrated results in OneLake. Fabric serves as the presentation layer, providing self-service analytics on the prepared datasets. This architecture lets data engineers focus on data quality and transformation logic while business users create their own analyses.

Key considerations

Security and governance are critical factors when you implement these integrations. The Fabric-to-Unity Catalog integration **bypasses Unity Catalog's fine-grained access controls** for downstream users. Carefully consider which tables to expose and ensure your organization accepts this security model. For scenarios requiring strict governance, consider using **Power BI Direct Query** instead, which honors Unity Catalog permissions.

Performance and cost also matter. Reading Unity Catalog data from Fabric requires a running Fabric capacity for metadata scans and refreshes, which introduces additional costs. When writing from Azure Databricks to OneLake, network transfer costs apply if data crosses region boundaries. Plan your architecture to minimize unnecessary data movement and optimize for your workload patterns.

Storage architecture remains important. Both integration patterns leverage **Azure Data Lake Storage** as the underlying storage layer. Azure Databricks uses Unity Catalog to manage and govern this storage, while Fabric accesses it through OneLake abstractions. Understanding this **shared foundation** helps you design efficient solutions that avoid data duplication and maintain consistency.

The choice between integration patterns depends on your specific requirements. Use Fabric reading from Unity Catalog when you want to expose existing Databricks-managed data to Fabric users and can accept the security model. Use Azure Databricks writing to OneLake when you need to publish curated datasets specifically for Fabric consumption and want more control over what data is shared. In many architectures, you might use both patterns for different scenarios within the same organization.

3. Understand integration with Power BI

<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/3-integration-databricks-power-bi>

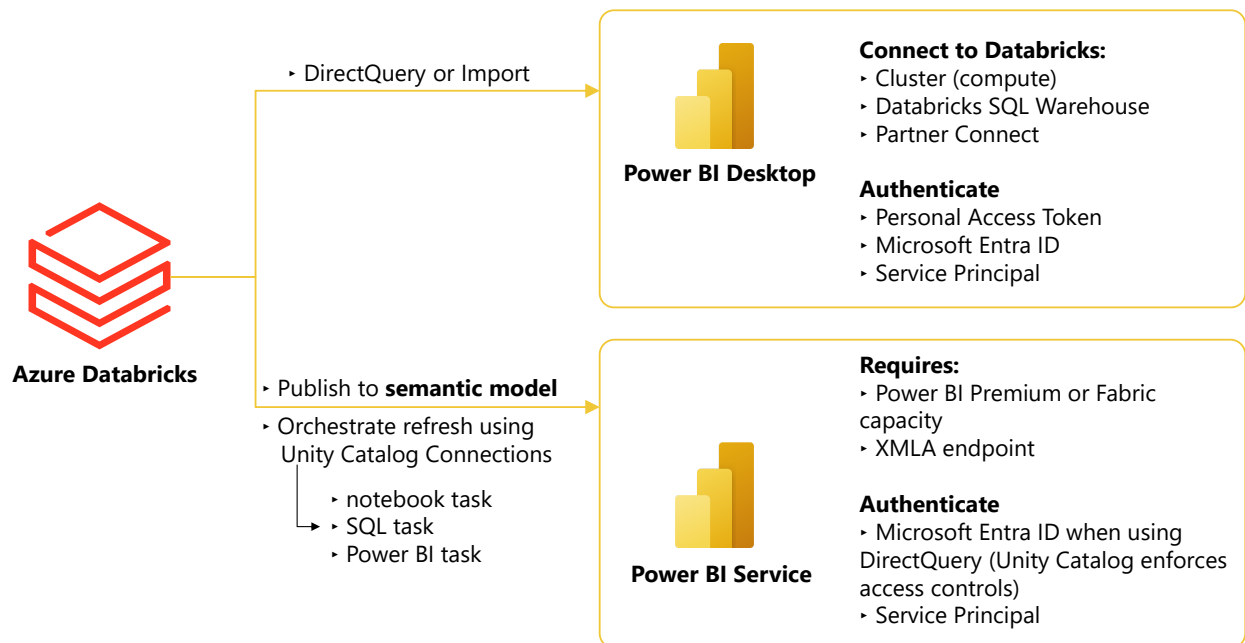
Understand integration with Power BI

Completed

Power BI provides interactive visualizations and business intelligence capabilities. Integrating Power BI with Azure Databricks allows users to create reports and dashboards from Databricks-managed data. This integration supports multiple user roles, from data engineers to business analysts.

The integration between Azure Databricks and Power BI works through **multiple connection methods**. You can connect Power BI Desktop to your compute resources, publish data directly from Azure Databricks to the Power BI service, or orchestrate semantic model refreshes using Unity

Catalog connections. Each method serves different scenarios and supports various authentication options.



Connect Power BI Desktop to Azure Databricks

Power BI Desktop is a Windows-based application that enables you to create interactive reports and dashboards. You can connect Power BI Desktop to both Azure Databricks **clusters** and **Databricks SQL warehouses**. **Use SQL warehouses with DirectQuery mode** for optimized query performance and built-in serverless scaling.

The fastest way to establish a connection is through **Partner Connect**. From the Azure Databricks workspace Marketplace, you select the Power BI tile, choose your compute resource, and download a connection file. This file opens Power BI Desktop with preconfigured connection settings. Alternatively, you can configure the connection manually by entering the **Server Hostname** and **HTTP Path** from your compute resource details.

Connect to partner

 **Microsoft Power BI**

Quickly find meaningful insights within your data and easily build rich, visual analytic reports.

You can use Partner Connect to connect Power BI Desktop to a Azure Databricks cluster or SQL warehouse. Select the target cluster or SQL warehouse, and then download and open the connection file (.pbids) to start Power BI Desktop. You must have Power BI Desktop version 2.99.563.0 or above installed.

[Learn more](#)

Compute ⓘ

Serverless Starter Warehouse ▼

Download connection file

Cancel

After establishing the connection, you select your **data connectivity mode**. With **Import** mode, Power BI loads data into its internal storage engine, enabling fast queries but requiring periodic refreshes to stay current. With **DirectQuery** mode, Power BI queries Azure Databricks directly for each visualization, ensuring real-time data access but requiring an active connection and running compute resource.

Authentication determines how Power BI accesses your data. **Personal access tokens** provide simple authentication but represent a single user identity. **Microsoft Entra ID** enables single sign-on where each user's credentials pass through to Azure Databricks, allowing Unity Catalog to enforce fine-grained permissions. For automated scenarios, **service principals** with machine-to-machine OAuth provide fixed credentials that don't depend on individual user accounts.

Power BI Desktop also supports **native SQL queries** for compute-intensive operations. Instead of using Power BI's visual query builder, you write SQL directly against your Databricks SQL warehouse. This approach gives you full control over query optimization and access to Databricks-specific SQL features.

Note

Power BI Desktop requires Windows. If you use a different operating system, run Power BI Desktop on a Windows-based virtual machine and connect to it remotely.

Publish to the Power BI service from Azure Databricks

The Power BI service is a cloud-based platform where you share reports and collaborate with your organization. Rather than connecting Power BI to Databricks, you can publish data **from** Azure Databricks directly to the Power BI service. This workflow simplifies the publishing process by initiating it from the Azure Databricks UI.

To publish data, you use **Catalog Explorer** in your Azure Databricks workspace. You select a schema or specific tables from Unity Catalog, then choose **Publish to Power BI workspace**. The publishing wizard authenticates you with Microsoft Entra ID and prompts you to select your target Power BI workspace, choose between DirectQuery and Import modes, and configure authentication settings.

Connect to partner



Microsoft Power BI

Quickly find meaningful insights within your data and easily build rich, visual analytic reports.

You can use Partner Connect to directly publish to Power BI. Connect to Microsoft Entra ID to select a Power BI workspace and publish your data.

Power BI Premium is required for publishing data to your Power BI workspace.

[Learn more](#)

SQL warehouse ⓘ

Serverless Starter Warehouse

Power BI workspace* ⓘ

AdventureWorks

Power BI semantic model*

☒ Publish new semantic model

☐ Update an existing semantic model

Select a semantic model

Power BI query mode*

DirectQuery

Tables to publish*

databricks_learn.adventure-works
24 out of 24 tables selected

► Select tables and set query modes

Authentication method in Power BI* ⓘ

OAuth

Publish to Power BI

Cancel

This publishing method requires **Unity Catalog** for data governance and a **Power BI Premium license** (Premium capacity, Premium Per User, or Microsoft Fabric capacity). The Premium license

provides the necessary capacity for enterprise scenarios and enables the XMLA (XML for Analysis) endpoint that Databricks uses to create and update semantic models programmatically.

When you publish a schema containing multiple tables, Databricks creates a **semantic model** in your Power BI workspace. **Column comments** from Unity Catalog tables transfer to Power BI as column descriptions, preserving your documentation. **Foreign key relationships** also transfer, though Power BI supports only one active relationship path between any two tables, so some relationships might become inactive if multiple paths exist.

Use OAuth authentication for published semantic models to enable fine-grained access control and user-level auditing. When users access a report built on DirectQuery mode, Power BI passes their Microsoft Entra ID credentials to Azure Databricks. Unity Catalog evaluates permissions for that specific user, enforcing row-level filters, column masks, and table access controls. This approach maintains governance consistency across platforms.

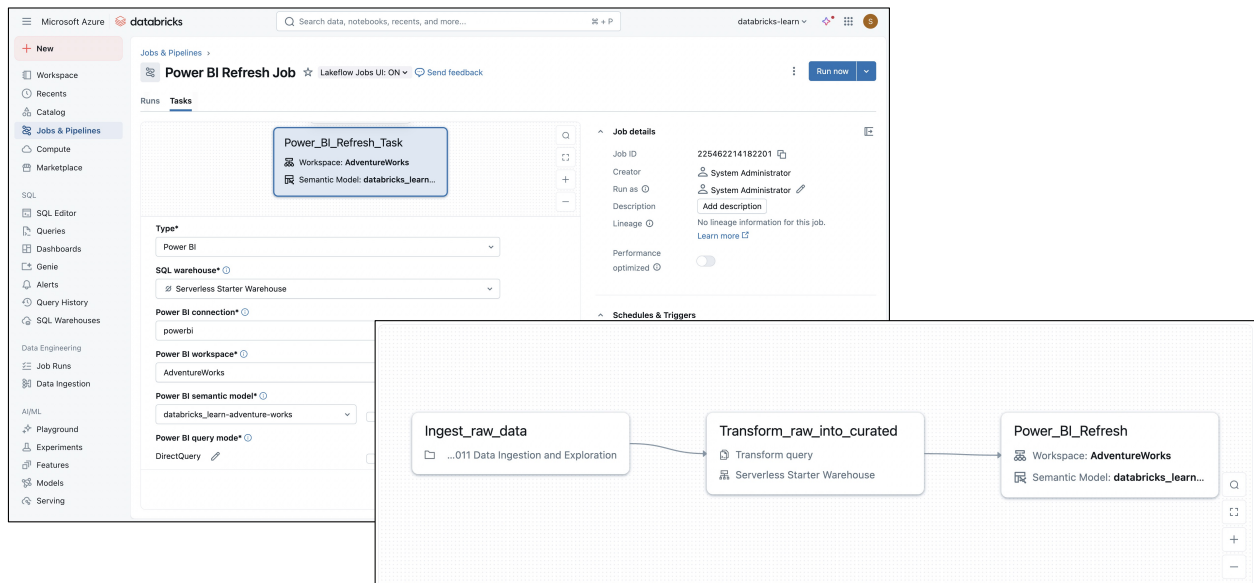
For scenarios requiring automated refreshes or shared credentials, you can configure **machine-to-machine OAuth** using service principals. After publishing, you edit the semantic model's data source credentials in Power BI and provide the service principal's client ID and secret. This method works with both semantic models and Power BI gateways for private network scenarios.

Important

Publishing to Power BI service requires the XMLA endpoint to be enabled with Read and Write capabilities in your Power BI/Fabric capacity settings.

Automate Power BI publishing with Power BI tasks

While you can publish to Power BI manually from Azure Databricks, **Power BI tasks** enable you to orchestrate semantic model publishing and refreshes automatically as part of your Databricks workflows. This automation integrates Power BI updates into your data pipelines, ensuring that reports refresh only after upstream data transformations complete.

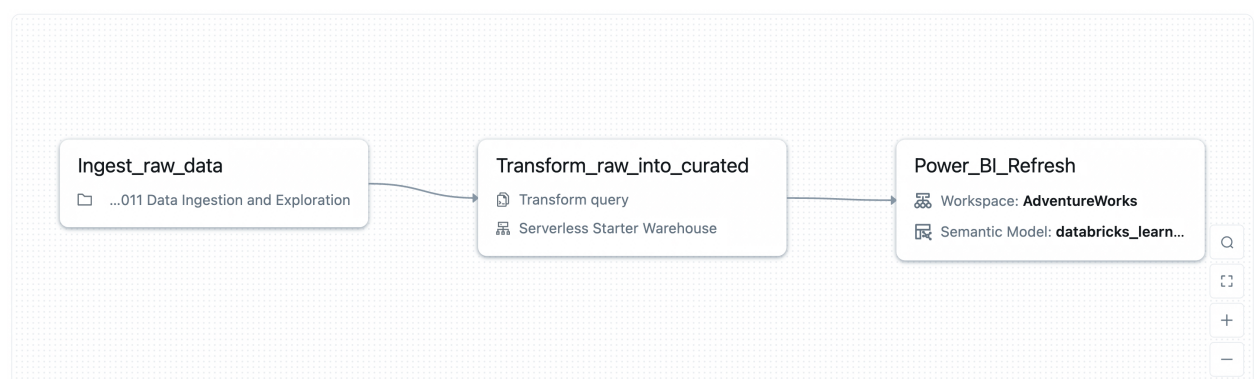


Power BI tasks require a **Unity Catalog connection** to Power BI. This connection stores authentication credentials and workspace information that the task uses to publish or update semantic models. You choose an authentication method (service credentials, OAuth machine-to-machine, or OAuth user-to-machine) based on your security requirements.

Power BI tasks integrate into Databricks workflows alongside data engineering tasks, machine learning model training, and other automation. You create task dependencies to control execution order, ensuring that Power BI semantic models refresh only after source data completes processing.

For example, a workflow might include:

- A **notebook task** that ingests raw data from external sources into Unity Catalog tables
- A **SQL task** that transforms the raw data into curated analytics tables
- A **Power BI task** that publishes the curated tables to a semantic model and triggers a refresh



When upstream data updates complete successfully, the Power BI semantic model automatically refreshes with the latest data.

Choose the right integration method

The choice between integration methods depends on your workflow and requirements. **Use Power BI Desktop connections** when business analysts need to explore data interactively and build reports. Desktop provides a report authoring environment with drag-and-drop visualizations and the flexibility to combine multiple data sources.

Use publishing from Azure Databricks to Power BI service when you want to share curated datasets with specific governance and want to initiate sharing from the data engineering side. This method ensures that data engineers control which tables become available in Power BI and can document them with column descriptions that transfer automatically.

Orchestrate Power BI updates with data pipelines when you need to automate semantic model refreshes as part of larger data pipelines. This approach coordinates Power BI updates with upstream data transformations, ensuring that reports refresh only after source data completes processing.

For organizations with private network requirements, Power BI gateways enable secure connectivity without exposing Azure Databricks endpoints publicly. You configure an on-premises or virtual network gateway with your Azure Databricks connection details and authentication credentials. Power BI service routes queries through the gateway, which connects to your private Azure Databricks workspace.

Security and governance considerations

Security architecture differs between connection methods and authentication types. The following table compares how Unity Catalog enforces permissions and tracks access:

Use case	Authentication method	Unity Catalog permission enforcement	Audit granularity
Interactive exploration with Power BI Desktop or DirectQuery reports requiring user-specific governance.	Microsoft Entra ID (SSO)	Evaluates permissions per user. Row-level filters, column masks, and table access controls apply based on each viewer's identity.	User-level audit logs show which specific users accessed which data.
Scenarios where all users should see identical data, or when implementing row-level security within Power BI.	Import mode or no SSO	Uses fixed credentials. All report viewers see the same data regardless of their Unity Catalog permissions.	Limited to the credential owner's identity. Can't track individual viewer access.
Automated workflows, scheduled refreshes, and scenarios where consistent data access is required.	Service principals	Evaluates permissions based on the service principal's grants, not individual viewers.	Shows only that the service principal accessed data, not which report viewers consumed it.
Development and testing.	Personal access tokens	Represents a single user identity. Permissions apply to that user only. Not	Shows the token owner accessed data,

Use case	Authentication method	Unity Catalog permission enforcement	Audit granularity
		recommended for production automation.	not the actual end user.

4. Understand integration with VS Code

<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/4-integration-databricks-vscode>

Understand integration with VS Code

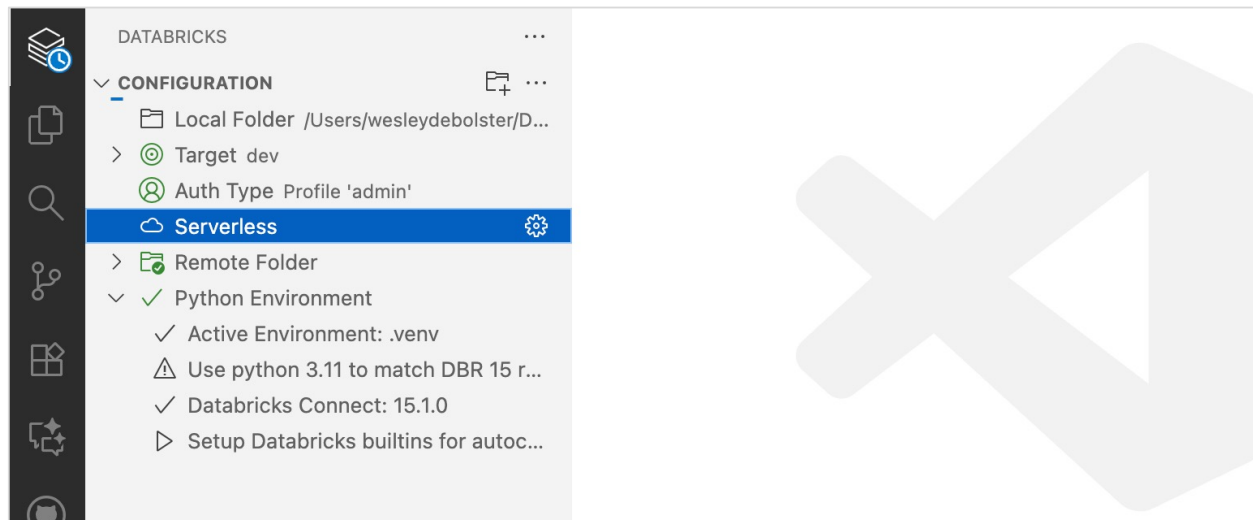
Completed

Data engineers working with Azure Databricks often switch between their local development environment and remote workspaces. You write code locally in your preferred editor, then copy it to Databricks notebooks or upload files to test on clusters. This back-and-forth workflow slows down development and makes debugging difficult. The Databricks extension for Visual Studio Code changes this by connecting your local development environment directly to your remote Azure Databricks workspace.

With this integration, you develop in Visual Studio Code using familiar tools and shortcuts while executing code on Azure Databricks compute resources. You don't need to leave your editor to run notebooks, test Python scripts, or deploy workflows. This approach keeps your development environment consistent while taking advantage of Databricks' distributed computing power.

Develop locally and execute remotely

Visual Studio Code provides a development environment on your local machine with features like **IntelliSense**, **syntax highlighting**, and **Git integration**. The [Databricks extension](#) adds the ability to execute your local code on remote Azure Databricks clusters or serverless compute. You write code in `.py` files or notebooks on your computer, then run them directly on Azure Databricks infrastructure without manually copying files or switching between applications.



You can also run Python, R, Scala, and SQL notebooks as **Lakeflow Jobs**. Instead of executing code interactively on a cluster, you package your notebook as a job that runs on a schedule or trigger. This capability bridges local development and production deployment, letting you test job configurations before committing them to source control.

The extension supports multiple Databricks projects within a single Visual Studio Code workspace. If you work on different Databricks environments or modules, you switch between project configurations without closing and reopening folders. Each project maintains its own workspace connection, authentication settings, and cluster selection.

Debug code with Databricks Connect

The Databricks extension integrates **Databricks Connect** to enable full debugging capabilities within Visual Studio Code.

Databricks Connect creates a direct connection between your local Python environment and a remote Azure Databricks cluster. When you debug code, the Python debugger in Visual Studio Code controls execution on the cluster. You set breakpoints, inspect variables, and step through code just as you would with local Python scripts. The difference is that your code runs on Databricks compute with access to distributed data and Spark APIs.

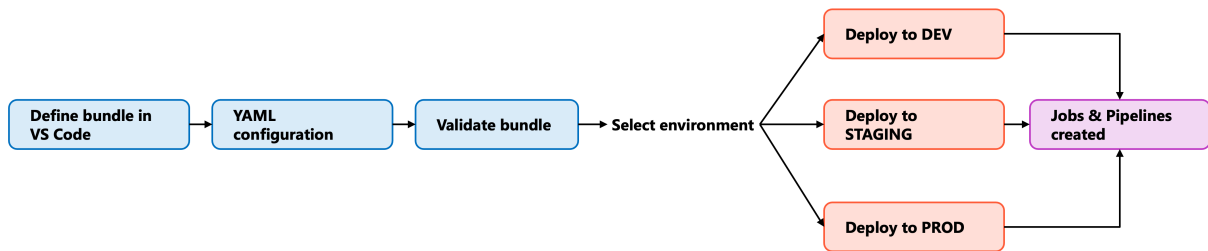
This debugging environment works with both Python files and notebooks. You can debug notebooks cell by cell, examining the state of DataFrames and variables at each step. When you encounter an error, you modify your code locally, set a breakpoint before the problematic line, and rerun the debugger to investigate. This tight feedback loop reduces the time spent troubleshooting data pipelines and transformations.

Beyond interactive debugging, the extension supports running Python tests with `pytest`. You write unit tests for your data processing logic and execute them on Databricks compute. This capability ensures that your tests run against the same environment and dependencies as your production code, catching environment-specific issues early in development.

Deploy workflows with Asset Bundles

The Databricks extension simplifies deployment through **Databricks Asset Bundles**, which package your code, configurations, and dependencies into deployable units.

Asset Bundles define **Lakeflow Jobs**, **Lakeflow Spark Declarative Pipelines (SDP)**, and **MLOps Stacks** using configuration files. You specify compute settings, schedule triggers, and task dependencies in YAML format. The extension provides a UI for creating, validating, and deploying these bundles without leaving Visual Studio Code.



When you're ready to deploy, the extension applies your bundle to a target environment (development, staging, or production). It creates or updates jobs, configures clusters, and sets up permissions according to your specifications. This declarative approach ensures consistent deployments and makes it easier to apply CI/CD patterns to your data engineering workflows.

With bundles, you can version your entire workflow configuration alongside your code in Git. Changes to job schedules, cluster sizes, or task dependencies go through the same review and approval process as code changes. This practice improves collaboration and provides an audit trail for production deployments.

Synchronize code between local and remote workspaces

While Asset Bundles handle deployment, you might also want to keep local code synchronized with workspace folders for quick prototyping or collaboration. The extension supports one-way automatic synchronization from your local Visual Studio Code project to workspace directories. You edit files locally, and changes automatically upload to the workspace. The workspace files are intended to be transient, so you should avoid making changes directly in the workspace, as these changes won't sync back to your local project.

This synchronization capability works well for teams transitioning from workspace-based development to local development. You maintain existing workspace folders while gradually adopting local development practices and version control. By syncing your local files to the workspace, you can test and run them remotely while keeping your local project as the source of truth for version control in Git.

However, for production workflows, Asset Bundles provide better control and consistency. Synchronization suits exploratory work and quick iterations, while bundles suit structured deployment pipelines with environment-specific configurations.

5. Understand integration with Power Platform

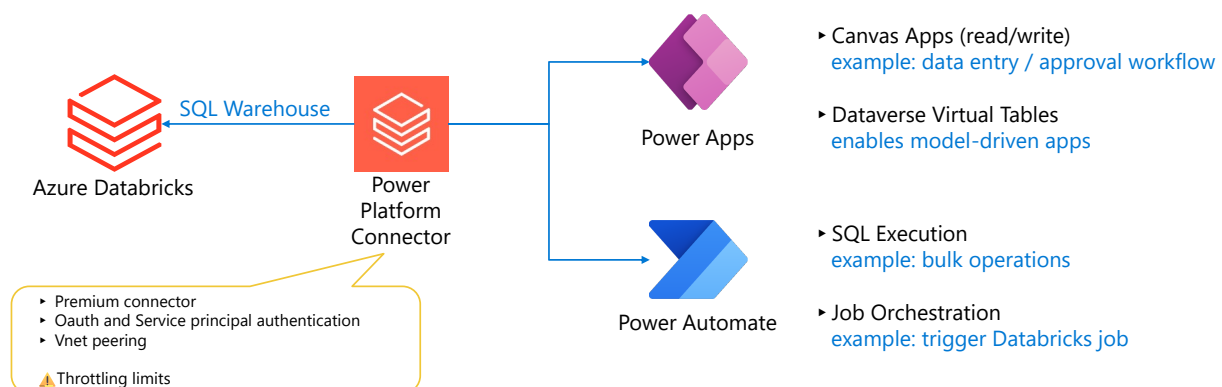
<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/5-integration-databricks-power-platform>

Understand integration with Power Platform

Completed

Power Platform enables you to build custom applications, automate workflows, and analyze data without extensive coding. Integrating Azure Databricks with Power Platform allows you to bring governed, high-quality data from your lakehouse directly into business applications and automation workflows. This integration supports data engineers in delivering data solutions that empower business users across the organization.

The integration between Azure Databricks and Power Platform works through the **Azure Databricks connector**. This connector enables Power Apps and Power Automate to access data stored in Unity Catalog while preserving your governance controls. You can build applications that read and write data, automate job execution, or orchestrate complex data workflows—all while maintaining the security policies and access controls defined in Unity Catalog.



Connect Power Platform to Azure Databricks

Power Platform connects to Azure Databricks through a **premium connector** that requires specific prerequisites. You need a premium Power Apps license, a Microsoft Entra ID account, and access to a SQL warehouse in your Azure Databricks workspace. The connector uses SQL warehouses as the compute resource because they provide optimized query performance and serverless scaling for analytical workloads.

To establish a connection, you create an **Azure Databricks connection** in Power Apps or Power Automate. From the Connections page, you search for the Azure Databricks connector and select

your authentication method. The connector supports two authentication types: **OAuth connection** (which uses Microsoft Entra ID to authenticate individual users) and **service principal connection** (which uses fixed credentials for authentication). Each authentication type serves different scenarios and offers different capabilities.

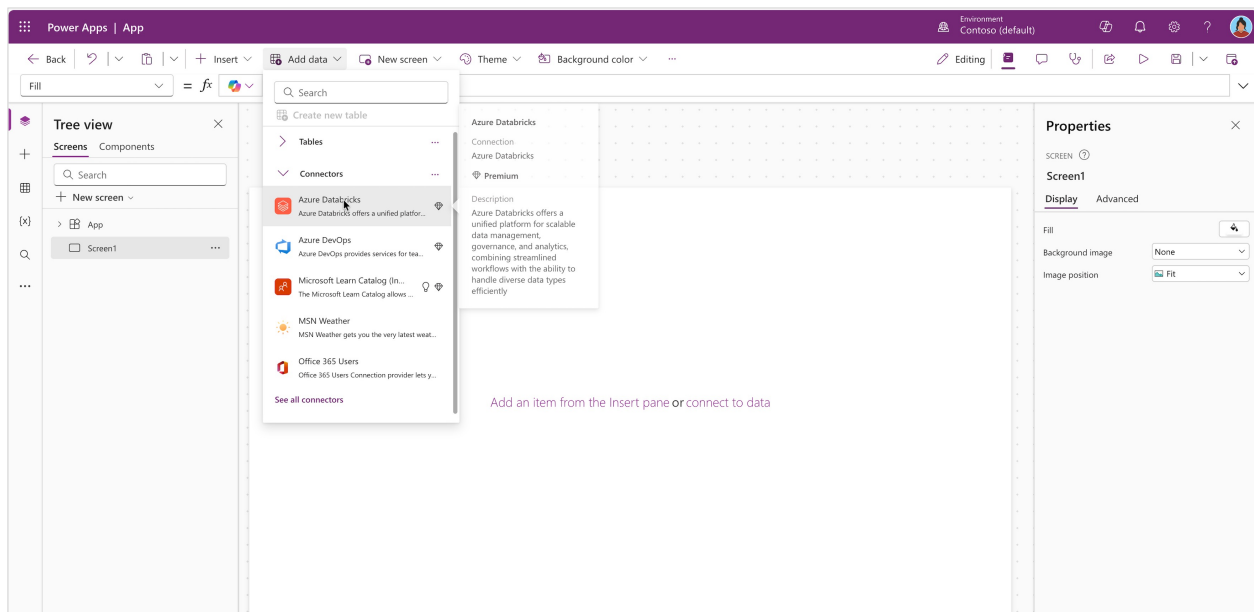
To complete the connection setup, you provide the **Server Hostname** and **HTTP Path** from your SQL warehouse. These connection details identify which compute resource Power Platform uses to query your data. You can find these values in the connection details section of your SQL warehouse in the Azure Databricks workspace.

Note

If your Azure Databricks workspace uses virtual networks, you can integrate Power Platform using VNet peering for private connectivity or by configuring IP access lists to allow AzureConnectors IP ranges.

Build applications with Power Apps

Power Apps enables you to build **canvas apps** that provide custom interfaces for reading and writing data. When you connect a canvas app to Azure Databricks, you select a catalog and choose the tables that your application will access. The app can perform create, update, and delete operations on your data, making it possible to build data entry forms, approval workflows, or dashboards that interact directly with your lakehouse.



With a direct connection to Azure Databricks, you can build canvas apps that interact with data while **preserving Unity Catalog governance**. When you use OAuth authentication, Unity Catalog evaluates each user's permissions in real time, ensuring that the app respects row-level security, column masking, and other access controls. This approach lets business users work with trusted data through an intuitive interface without compromising security.

For applications that require **bulk operations**, Power Automate flows can process multiple changes in a batch. This pattern handles large-scale updates more efficiently than individual operations from the canvas app.

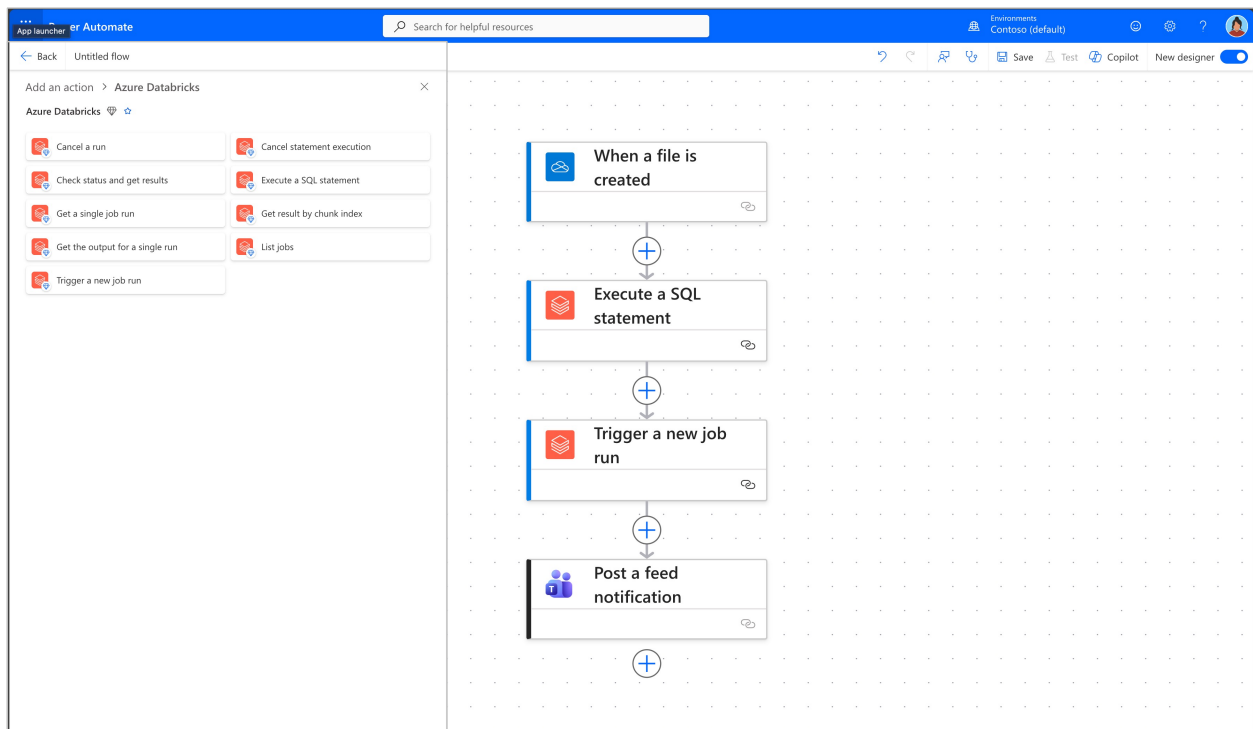
Power Apps also supports **Dataverse virtual tables** backed by Azure Databricks data. Virtual tables integrate data from Azure Databricks with Microsoft Dataverse without physically copying the data. This approach enables you to build **model-driven apps** that provide structured forms, views, and business logic on top of your Databricks data.

Tip

For better performance, use direct connections rather than virtual tables when possible. While virtual tables don't consume Dataverse storage capacity, direct connections typically offer faster query response times.

Automate workflows with Power Automate

Power Automate enables you to build **flows** that automate business processes and data operations. The Azure Databricks connector exposes two primary capabilities within Power Automate: **SQL statement execution** for running queries and **job orchestration** for triggering existing Databricks jobs. These capabilities enable you to create automated workflows that respond to business events, schedule data processing tasks, or coordinate complex data pipelines.



With **SQL statement execution**, you can write and execute SQL queries directly from a flow. You create a flow with any trigger type—such as a schedule, a manual trigger, or an event from another

service—then add an Azure Databricks action to execute SQL. The flow can retrieve query results and use them in subsequent steps, with support for handling large result sets through chunked retrieval.

Beyond SQL execution, **job orchestration** lets you trigger existing Azure Databricks jobs from Power Automate. You can start a job run, track its progress, and retrieve metadata about the run including its status, start and end times, and execution duration. This integration enables you to **coordinate data processing workflows** with business processes—for example, triggering a data refresh when new files arrive or sending notifications when a data quality job completes.

The connector also supports canceling running statements or jobs, listing available jobs, and retrieving output from completed runs. These capabilities enable scenarios like automatic retry logic, conditional workflows based on data quality checks, or multi-step data pipelines that combine Databricks processing with other services.

Important

The Power Platform connector has throttling limits. Design your flows to batch operations appropriately when processing large volumes.

Integration scenarios

The Azure Databricks and Power Platform integration enables several practical scenarios that bridge the gap between data engineering and business operations. Consider a **planning and forecasting application** where business users submit next month's forecasts, adjust revenue targets, or update budget allocations through a Power App that writes directly to a Unity Catalog table. With OAuth authentication and Unity Catalog governance, each user sees only the data they're authorized to access, and all changes are tracked in the table's audit log.

Another scenario involves **automated data quality workflows**. You build a Power Automate flow that runs on a schedule, executes a SQL query to check data quality metrics, and sends notifications to the data team when quality thresholds are exceeded. The flow can also trigger a Databricks job to repair data issues automatically or escalate to a human review process based on the severity of the issue.

For **event-driven data processing**, you create a flow that monitors a business system for events—such as a new customer order or a completed transaction—and triggers a Databricks job to process the associated data. The flow passes relevant parameters to the job, waits for completion, and updates the business system with the results. This pattern enables real-time data integration without building custom integration code.

Dashboard refresh automation represents another valuable use case. You schedule a Power Automate flow that executes SQL to prepare aggregated data, stores the results in a table, and triggers a refresh of a business intelligence dashboard. This approach provides business users with up-to-date insights without manual intervention from the data team.

These scenarios become possible because the integration **preserves governance while enabling access**. Unity Catalog continues to enforce all security policies, data masking, and access controls even when data flows through Power Platform. Business users gain the ability to work with data through familiar interfaces while data engineers maintain centralized control over data quality, security, and compliance.

Considerations and limitations

While the Azure Databricks Power Platform connector provides broad capabilities, some considerations affect how you design your solutions. The connector **doesn't support government clouds**, including US Government and China Cloud environments. Organizations operating in these environments need alternative integration approaches.

Within Power Apps, certain **PowerFx formulas** calculate values using only data that has been retrieved locally to the app, rather than delegating computation to Azure Databricks. For large datasets, this can affect performance and accuracy. Consider filtering data or performing aggregations in SQL within Azure Databricks before bringing data into Power Apps.

Concurrent write operations from multiple Power Apps users or flows benefit from row-level concurrency features in Azure Databricks. These features are available in recent Databricks Runtime versions and help reduce write conflicts when multiple users modify data simultaneously.

When choosing between **virtual tables and direct connections**, consider the trade-offs. Virtual tables enable model-driven apps and integrate with the broader Dataverse ecosystem, but they don't support OAuth credential passthrough. Direct connections offer better performance and support OAuth for governance passthrough, but they only work with canvas apps. Choose the approach that best fits your application architecture and governance requirements.

Finally, **data policies** in Power Platform affect connector usage. You can add the Azure Databricks connector to a Business data policy to control which other connectors can share data with it, helping you maintain compliance with data handling regulations.

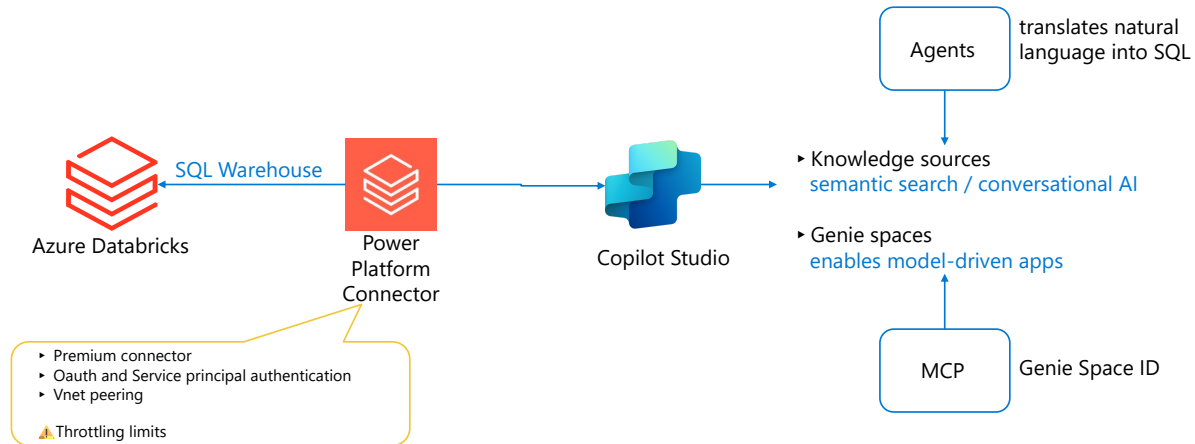
6. Understand integration with Copilot Studio

<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/6-integration-databricks-copilot-studio>

Understand integration with Copilot Studio

Completed

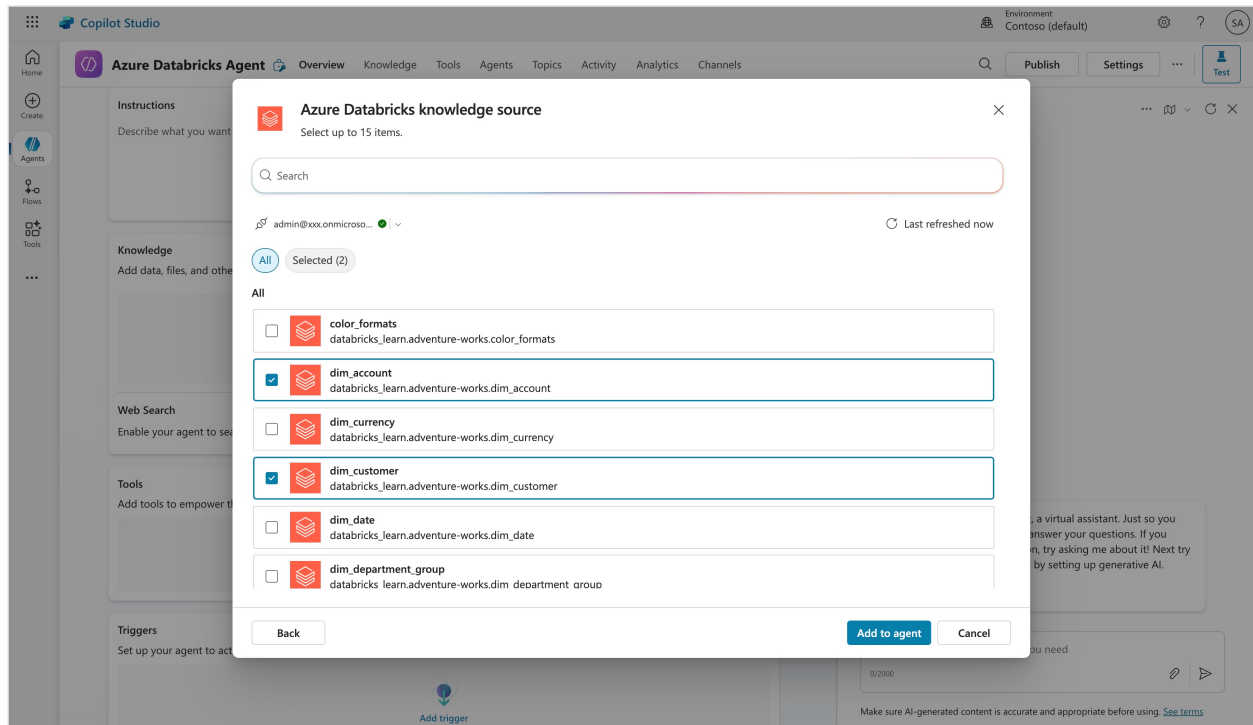
Azure Databricks integrates with Microsoft Copilot Studio to bring governed lakehouse data into conversational AI agents. The **Azure Databricks connector** enables two integration patterns: **knowledge sources** for semantic search and question answering, and **Genie spaces** as intelligent analytical tools. Both approaches preserve Unity Catalog governance, ensuring users access only authorized data.



Use tables as knowledge sources

Copilot Studio agents access Azure Databricks tables as **knowledge sources** to ground responses in your organization's data. Users ask questions in natural language, and the agent retrieves relevant information to provide data-driven answers.

Configure knowledge sources by connecting Copilot Studio to Azure Databricks using OAuth authentication or a service principal. Select a catalog and specific tables for your agent. The agent queries tables directly through the SQL warehouse without copying data, ensuring Unity Catalog enforces all governance controls including row-level security and column masking.



OAuth authentication passes each user's identity to Azure Databricks, where Unity Catalog evaluates permissions in real time. Service principal authentication runs all queries with consistent permissions, useful for uniform data access or cross-tenant scenarios.

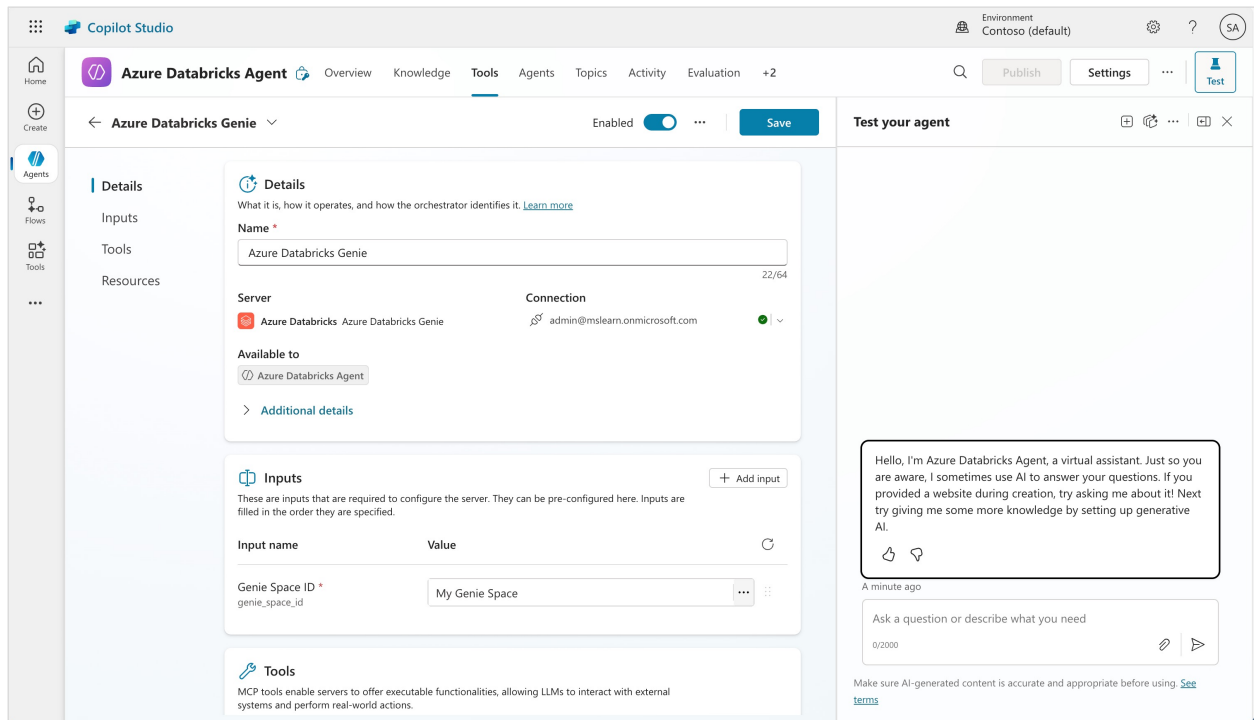
The agent performs **semantic search** by translating natural language into SQL, retrieving relevant rows, and synthesizing conversational responses. This enables business users to explore data without writing SQL or understanding the data model.

Note

Knowledge sources work best with tables that have descriptive column names and well-structured data. Consider creating curated views or aggregated tables specifically for agent consumption.

Connect Genie spaces as tools

Copilot Studio agents can interact with **Genie spaces** as intelligent tools. Genie is Azure Databricks' AI-powered analytics interface that understands natural language questions. The integration uses **Model Context Protocol (MCP)**, enabling your agent to orchestrate between conversation management and analytical queries.



To set up Genie integration, enable **Managed MCP Servers** preview in your Azure Databricks workspace, then connect Genie in Copilot Studio using the Genie Space ID. The agent can then send analytical questions to Genie and return results conversationally.

Genie differs from knowledge sources: instead of semantic search and data retrieval, Genie performs complex analytics, generates visualizations, aggregates data, and explains trends through its specialized analytics engine.

Important

Genie integration is currently in Public Preview and requires enabling the Managed MCP Servers preview in your workspace. Not all Azure Databricks features in preview are recommended for production workloads.

Integration scenarios

Customer support agents answer questions about order history, inventory, or shipping status using Azure Databricks tables as knowledge sources. Unity Catalog ensures each representative sees only authorized orders and customers.

Executive analytics agents connect to Genie spaces for business insights. When asked "What drove the increase in customer churn last quarter?", the agent invokes Genie to analyze trends, identify factors, and present findings conversationally.

Data literacy agents help employees understand datasets by answering questions like "What tables contain customer demographic data?" or "How is revenue calculated in the sales fact table?".

Compliance and audit agents query Unity Catalog metadata to answer questions about data lineage, access patterns, or governance policies, helping teams respond efficiently to audit requests.

These scenarios combine conversational AI with governed data access—users interact through natural language while Unity Catalog enforces security, audit logging, and access controls.

Considerations and limitations

The connector **doesn't support government clouds** (GCC, GCC High, China Cloud).

Agent performance depends on **table structure and data quality**. Use clear column names, consistent data types, and appropriate indexes. Consider creating dedicated views for agent consumption rather than exposing complex schemas with many joins.

The connector requires a **SQL warehouse**—serverless or pro only (classic warehouses aren't supported for Genie). Size appropriately for many small queries and configure auto-scaling and auto-stop for cost optimization.

Genie integration is in Public Preview. Enable Managed MCP Servers in workspace settings and consider stability requirements before production deployment.

Service principal authentication executes all queries with the principal's permissions, providing consistent access but preventing user-level row security. OAuth enables user-specific governance but requires users to have Azure Databricks permissions and both platforms in the same Microsoft Entra tenant.

Power Platform **throttling limits** apply: 100 API calls per 60 seconds. Design agents to batch operations and cache frequently accessed information.

7. Understand integration with Microsoft Purview

<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/7-integration-databricks-microsoft-purview>

Understand integration with Microsoft Purview

Completed

As organizations build data lakehouses in Azure Databricks, they face data governance challenges. Without integration between Databricks and governance tools, data engineers, business users, and

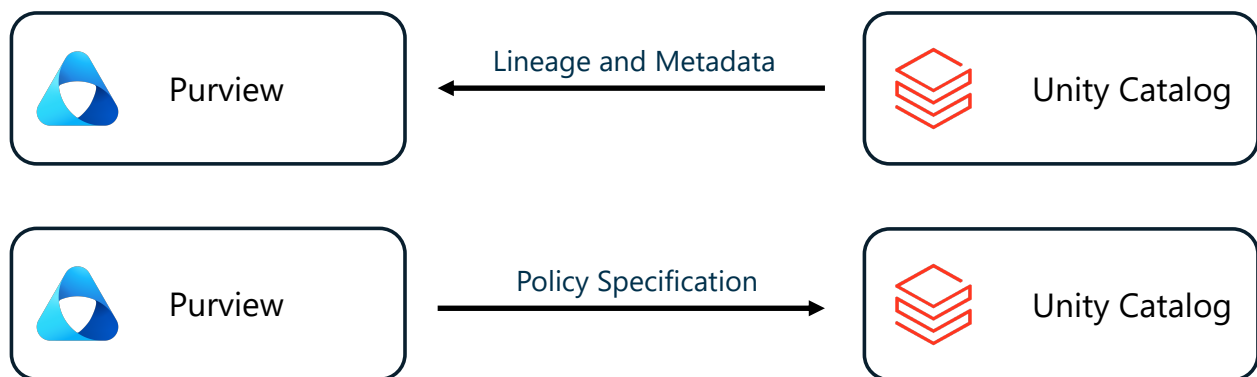
compliance teams work in isolation, creating data silos and governance gaps.

Microsoft Purview provides unified data governance across multiple sources, including Azure Databricks. This integration enables a single catalog of data assets, lineage tracking, and consistent governance policies.

How the integration works

The integration uses **metadata synchronization**. Purview reads metadata about databases, tables, columns, and relationships from Azure Databricks without accessing actual data. This metadata is cataloged alongside other sources.

The process involves **registration** (establishing the connection and authentication) and **scanning** (extracting metadata at scheduled intervals or on demand).



Scanning metadata sources

Purview can scan metadata from both Hive Metastore and Unity Catalog, each offering different capabilities based on the architectural differences between these metadata storage systems.

Scanning Hive Metastore metadata

Purview scans workspace-scoped Hive metastores to discover databases, tables, views, and column definitions. For external tables, it captures **storage relationships** between logical tables and physical storage locations.

Hive metastore scanning captures **static lineage** from view definitions, showing dependencies between views and underlying tables. This requires a self-hosted integration runtime to connect to Azure Databricks clusters.

Note

Hive Metastore scanning does not support incremental scanning. Each scan performs a full extraction, unlike Unity Catalog which supports incremental synchronization.

Scanning Unity Catalog metadata

Unity Catalog scanning extracts the complete hierarchy—metastores, catalogs, schemas, tables, and views. It supports **incremental synchronization**, processing only changes after the initial full scan, which improves efficiency for large environments.

Unlike Hive metastore, Unity Catalog scanning uses cloud-native Azure integration runtime to connect through SQL Warehouses, simplifying deployment.

Understanding runtime lineage

Unity Catalog scanning captures **runtime lineage**—actual data transformations from notebook execution. When notebooks read, transform, and write data, Unity Catalog records these operations in system tables that Purview scans.

Runtime lineage provides:

- **Table lineage:** shows which tables feed into other tables
- **Column lineage:** traces specific columns from source to destination

Limitations: Only transformations logged by Unity Catalog appear in lineage. External orchestration tools like Azure Data Factory don't appear, and column-level lineage may not capture all complex transformations.

Benefits of integration

Unified data discovery

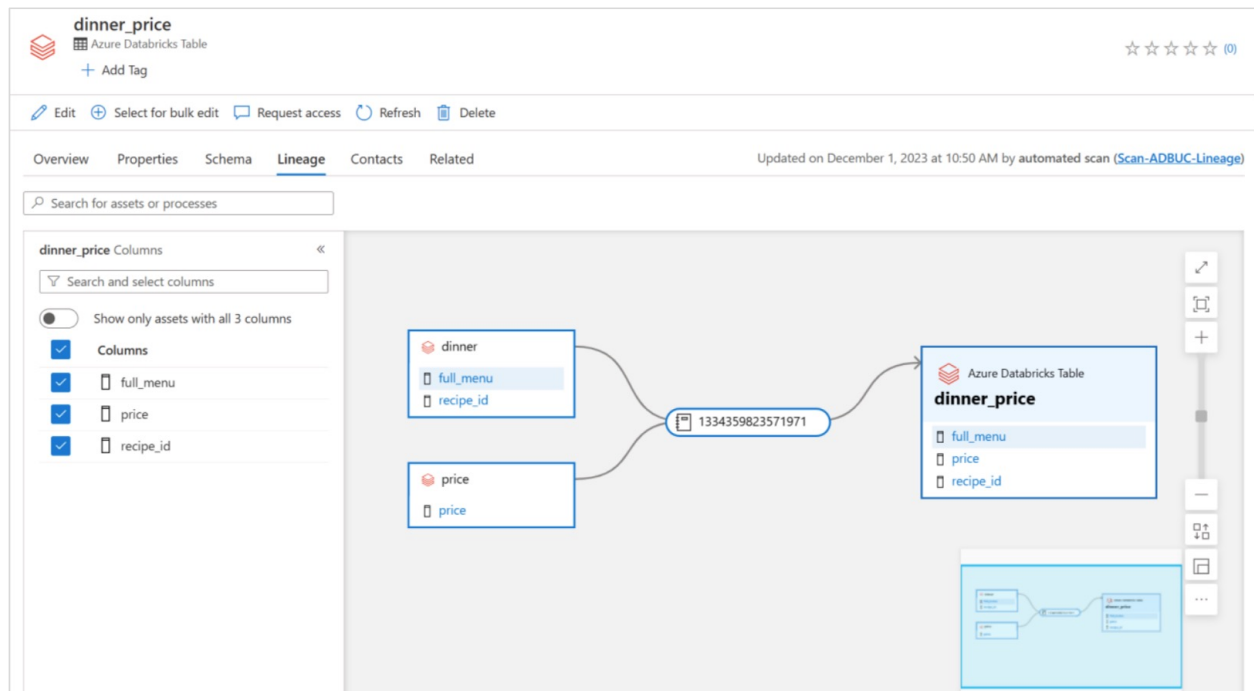
Integration eliminates catalog fragmentation. Azure Databricks tables appear in Purview alongside other data sources, enabling users to find all relevant data regardless of origin. This reduces duplication and improves collaboration.

Consistent governance

Governance teams apply classifications, ownership, and policies consistently across all data sources. Data classification and compliance reporting span the entire data estate using unified tools and processes.

Lineage for impact analysis and compliance

Lineage enables **impact analysis** by showing which downstream assets depend on source data before making changes. For **compliance**, lineage automatically documents data flows required by certain regulations, transforming manual documentation into an automated process.



Scanning approaches

Organizations choose between Hive Metastore and Unity Catalog scanning based on their architecture and governance maturity. During **migration** periods, Purview can scan both simultaneously without duplicating assets.

The approaches differ in infrastructure: Hive Metastore requires self-hosted infrastructure, while Unity Catalog uses cloud-native Azure services through SQL Warehouses. Choose based on network requirements and cloud-native preferences.

8. Understand integration with Microsoft Foundry

<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/8-integration-databricks-microsoft-foundry>

Understand integration with Microsoft Foundry

Completed

Azure Databricks and Microsoft Foundry serve complementary roles in modern AI and data architectures. Azure Databricks provides a platform for large-scale data engineering, transformation,

and machine learning, while Microsoft Foundry offers tools for building and deploying AI agents and generative AI applications. Understanding how these platforms integrate helps you design solutions that combine enterprise data capabilities with conversational AI experiences.

The integration between Azure Databricks and Microsoft Foundry uses the **Model Context Protocol (MCP)** to connect AI agents with data insights from Azure Databricks **Genie spaces**. This connection lets AI agents in Microsoft Foundry query data managed in Azure Databricks using natural language, without requiring direct database access or custom API development.

What is the Azure Databricks connector in Microsoft Foundry

Microsoft Foundry supports adding external tools and services to AI agents through standardized connectors. The **Azure Databricks Genie connector** appears as a tool option within Microsoft Foundry, enabling AI agents to access curated datasets and business intelligence insights from Azure Databricks.

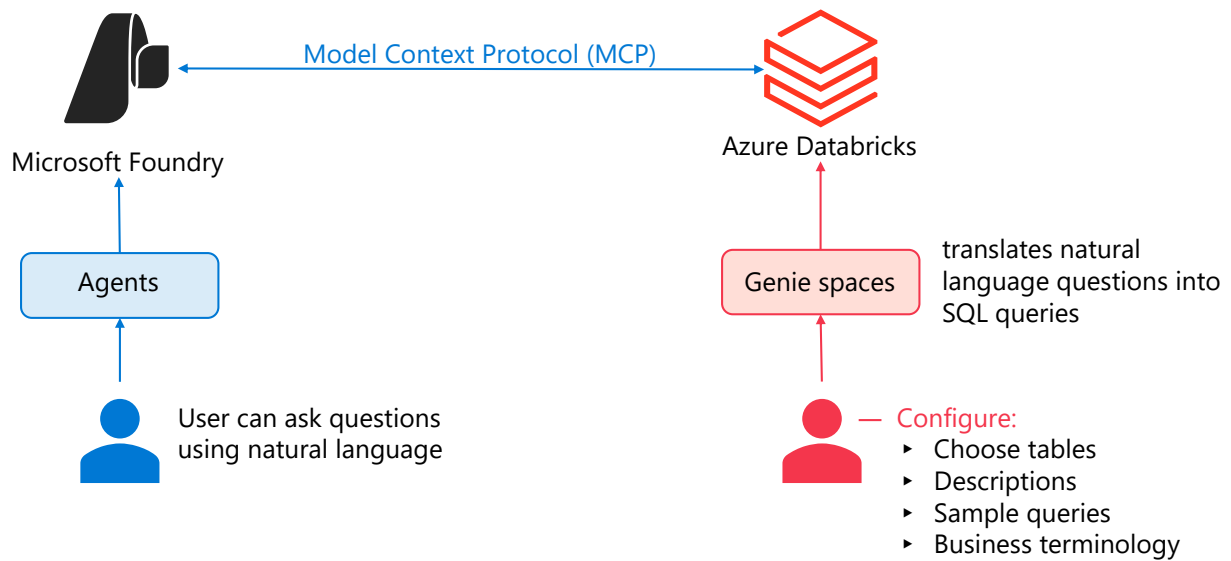
The connector works through Genie spaces, which are conversational interfaces built on top of Azure Databricks data. Domain experts configure these spaces with specific datasets, sample queries, and business terminology. When you add a Genie space as a tool in Microsoft Foundry, your AI agents gain the ability to answer questions about that data using natural language.

With this connector, Microsoft Foundry agents don't query raw tables or write SQL directly. Instead, they interact with the preconfigured Genie space, which translates natural language questions into SQL queries and returns results. This approach provides a layer of abstraction that simplifies integration while maintaining data governance through the Genie space configuration.

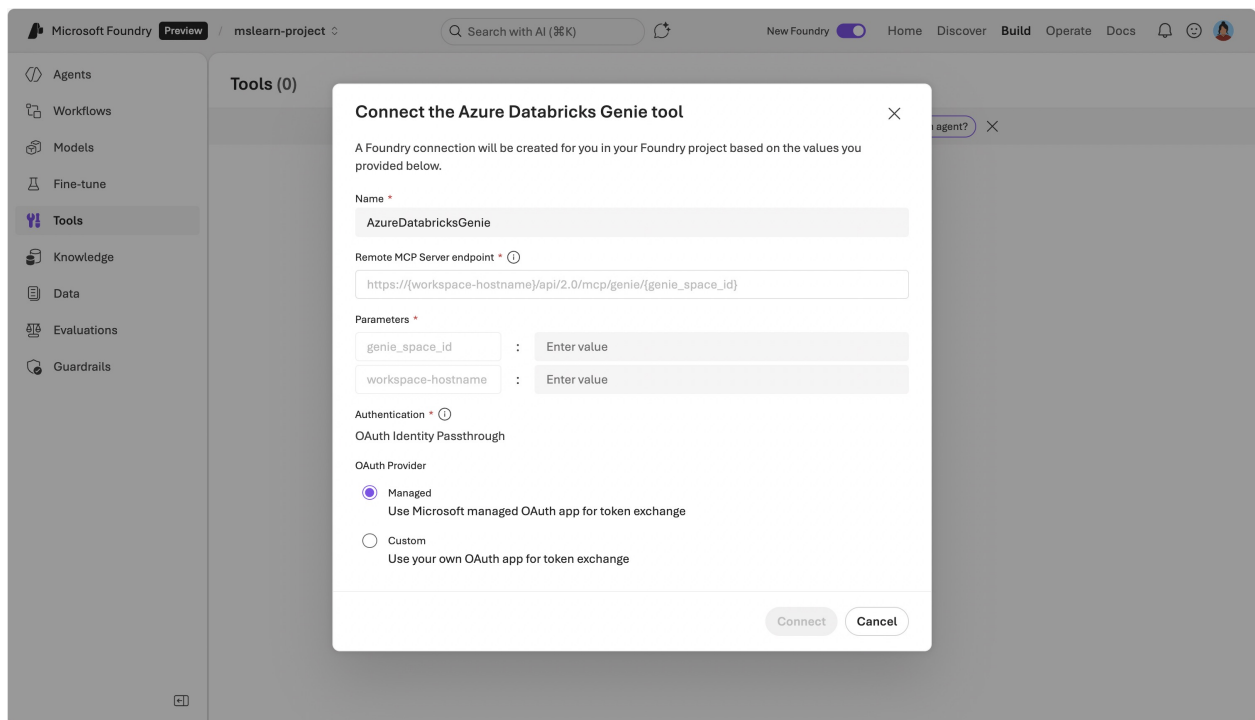
The integration relies on the Model Context Protocol, an open standard that defines how AI agents communicate with external tools and data sources. Azure Databricks implements MCP through managed servers that expose Genie spaces, Unity Catalog functions, and other resources to compatible AI platforms. Microsoft Foundry acts as an MCP client, sending requests to the Azure Databricks MCP server and receiving structured responses.

How the integration works

The architecture involves three main components working together. **Microsoft Foundry** hosts the AI agent and manages user interactions. The **Azure Databricks MCP server** exposes Genie spaces as MCP-compatible tools. **Genie spaces** contain the curated data, metadata, and business logic that enable natural language querying.



When you connect a Genie space to a Microsoft Foundry agent, you specify connection details including your workspace hostname, Genie space ID, and authentication method. Microsoft Foundry uses these details to establish a secure connection to the Azure Databricks MCP server endpoint. The MCP server validates permissions and provides access to the specific Genie space you've configured.



During runtime, when a user asks the AI agent a question, Microsoft Foundry determines whether to route that question to the Genie tool based on the agent's configuration and the question's context. If the Genie tool is invoked, Microsoft Foundry sends the question to the Azure Databricks MCP server. The Genie space processes the question, generates a SQL query, executes it against the underlying data, and returns the results to Microsoft Foundry. The AI agent then incorporates these results into its response to the user.

Authentication occurs through **OAuth Identity Passthrough**, which means user identities flow from Microsoft Foundry to Azure Databricks. You can choose between **Managed** OAuth, which uses Microsoft Entra ID authentication, or **Custom** OAuth for other identity providers. Databricks recommends Managed OAuth for most scenarios. Each time the AI agent accesses the Genie space, Azure Databricks validates that the requesting identity has appropriate permissions to use that space.

When to use this integration

This integration enables several practical scenarios where combining conversational AI with enterprise data creates value. You can build **AI assistants** that answer business questions using live data from your lakehouse. Users interact with these assistants through natural language, and the assistants query Databricks data through Genie spaces to provide accurate, data-driven responses.

For **self-service analytics**, business users can ask questions about data without learning SQL or understanding database schemas. The Genie space handles the complexity of translating questions into queries, while the Microsoft Foundry agent provides a conversational interface. This pattern works well when you want to extend data access beyond technical teams while maintaining governance through Genie space configuration.

You might use this integration to build **domain-specific assistants** for different parts of your organization. For example, a sales team assistant could connect to a Genie space configured with sales pipeline data, while a finance assistant connects to a different Genie space with financial metrics. Each assistant operates within its defined data scope, and you control what data each Genie space exposes.

The integration also supports **multi-tool agents** that combine Databricks data insights with other capabilities. An AI agent in Microsoft Foundry might use the Genie connector to retrieve data, use another tool to perform calculations or predictions, and use additional tools to take actions based on the results. This orchestration happens within Microsoft Foundry's agent framework, while Azure Databricks provides the data foundation.

Key considerations

Security and permissions require attention at multiple levels. In Azure Databricks, you must enable the **Managed MCP Servers** preview feature in your workspace. Users who configure the connection in Microsoft Foundry need permissions to use the Genie space in Azure Databricks. The Genie space itself enforces permissions on the underlying data through Unity Catalog, so users can only access data they're authorized to see through the Genie space configuration.

Genie space setup directly impacts the quality of responses your AI agent can provide. Domain experts should configure Genie spaces with relevant datasets, clear column descriptions, and sample queries that reflect common questions. The better configured your Genie space, the more effectively it can translate natural language questions into accurate SQL queries. Consider this as a one-time setup cost that improves the ongoing experience for all users.

Performance depends on both Microsoft Foundry and Azure Databricks. Questions that require complex queries or access large datasets take longer to process than simple lookups. The underlying SQL warehouse or compute resources in Azure Databricks affect query execution speed. Plan your compute capacity based on expected query patterns and user concurrency.

The integration has **specific limitations** to consider. The Genie MCP server doesn't maintain conversation history when invoked as an MCP tool, so each question is processed independently. If your scenario requires multi-turn conversations with context, you might need to explore alternative integration patterns. Additionally, network access between Microsoft Foundry and your Azure Databricks workspace must be configured appropriately, considering any IP restrictions or private endpoints you've implemented.

Cost considerations include both Microsoft Foundry and Azure Databricks components. Microsoft Foundry charges for agent usage and API calls. Azure Databricks charges for serverless SQL compute when Genie spaces execute queries. Genie uses serverless SQL compute to run, and pricing follows the standard Databricks SQL serverless pricing model. Evaluate your expected query volume and complexity to estimate costs accurately.

Note

The feature is currently in **Public Preview**, which means it's suitable for development and testing but may undergo changes before general availability. Plan for potential updates to configuration requirements or functionality as the feature evolves. Monitor Azure Databricks release notes for updates that affect your implementation.

9. Module assessment

<https://learn.microsoft.com/en-us/training/modules/understand-azure-databricks-integrations/9-knowledge-check>

Module assessment

Completed

10. Summary

Summary

Completed

Azure Databricks becomes more powerful when integrated with the Microsoft ecosystem. Throughout this module, you explored seven integrations that extend your data lakehouse capabilities across analytics, development, applications, and AI services. Each integration serves specific use cases while maintaining Unity Catalog as the central governance layer.

You learned how Microsoft Fabric and Power BI enable business intelligence, letting analysts create reports from Databricks data. The Visual Studio Code integration transforms development by enabling local coding with remote execution and debugging. Power Platform and Copilot Studio bring governed data into applications and AI agents, making lakehouse data accessible through familiar interfaces. Microsoft Purview provides unified governance, while Microsoft Foundry connects AI agents with analytical insights.

Understanding the security models, authentication methods, and limitations of each integration helps you design solutions that balance accessibility with governance. As you implement these patterns, match integration choices to specific business scenarios, evaluate security implications carefully, and test configurations thoroughly. The combination of Azure Databricks with Microsoft services creates comprehensive data and AI solutions that serve diverse needs while maintaining centralized governance and quality standards.