

OneLake in Microsoft Fabric documentation

OneLake is a single, unified, logical data lake for the whole organization. OneLake comes automatically with every Microsoft Fabric tenant with no infrastructure to manage.

About OneLake

OVERVIEW

[What is OneLake?](#)

[OneLake security](#)

[OneLake catalog](#)

[OneLake access and APIs](#)

DEPLOY

[Implement medallion lakehouse architecture](#)

GET STARTED

[Create a lakehouse with OneLake](#)

[OneLake file explorer](#)

[Find data in the OneLake catalog](#)

[Use Iceberg tables in OneLake](#)

OneLake shortcuts

CONCEPT

[What are shortcuts?](#)

GET STARTED

[Create a shortcut](#)

HOW-TO GUIDE

[Access shortcuts](#)

OneLake and Azure integration

HOW-TO GUIDE

[Integrate OneLake with Azure Databricks](#)

[Integrate OneLake with Azure HDInsight](#)

[Integrate OneLake with Azure Storage Explorer](#)

[Integrate OneLake with Azure Synapse Analytics](#)

OneLake, the OneDrive for data

Article • 07/25/2024

OneLake is a single, unified, logical data lake for your whole organization. A data Lake processes large volumes of data from various sources. Like OneDrive, OneLake comes automatically with every Microsoft Fabric tenant and is designed to be the single place for all your analytics data. OneLake brings customers:

- **One data lake for the entire organization**
- **One copy of data for use with multiple analytical engines**

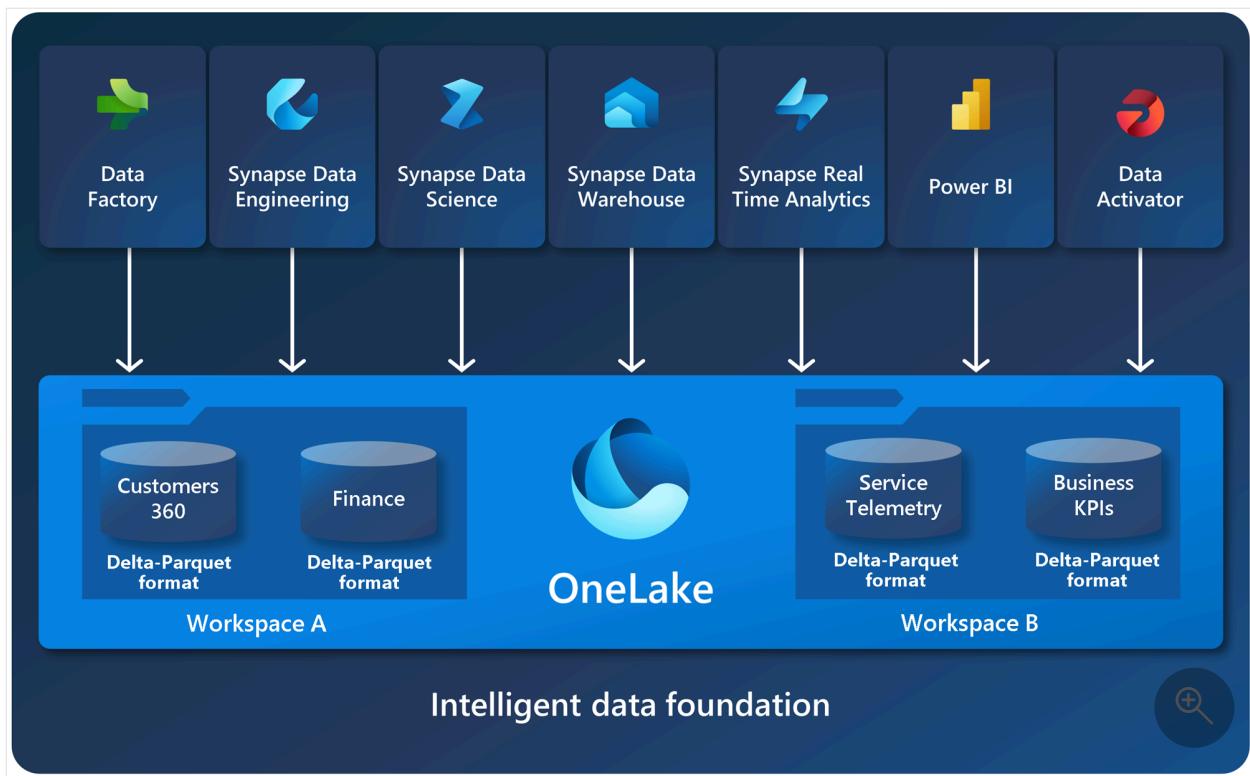
One data lake for the entire organization

Before OneLake, it was easier for customers to create multiple lakes for different business groups rather than collaborating on a single lake, even with the extra overhead of managing multiple resources. OneLake focuses on removing these challenges by improving collaboration. Every customer tenant has exactly one OneLake. There can never be more than one and if you have Fabric, there can never be zero. Every Fabric tenant automatically provisions OneLake, with no extra resources to set up or manage.

Governed by default with distributed ownership for collaboration

The concept of a tenant is a unique benefit of a SaaS service. Knowing where a customer's organization begins and ends provides a natural governance and compliance boundary, which is under the control of a tenant admin. Any data that lands in OneLake is governed by default. While all data is within the boundaries set by the tenant admin, it's important that this admin doesn't become a central gatekeeper preventing other parts of the organization from contributing to OneLake.

Within a tenant, you can create any number of workspaces. Workspaces enable different parts of the organization to distribute ownership and access policies. Each workspace is part of a capacity that is tied to a specific region and is billed separately.



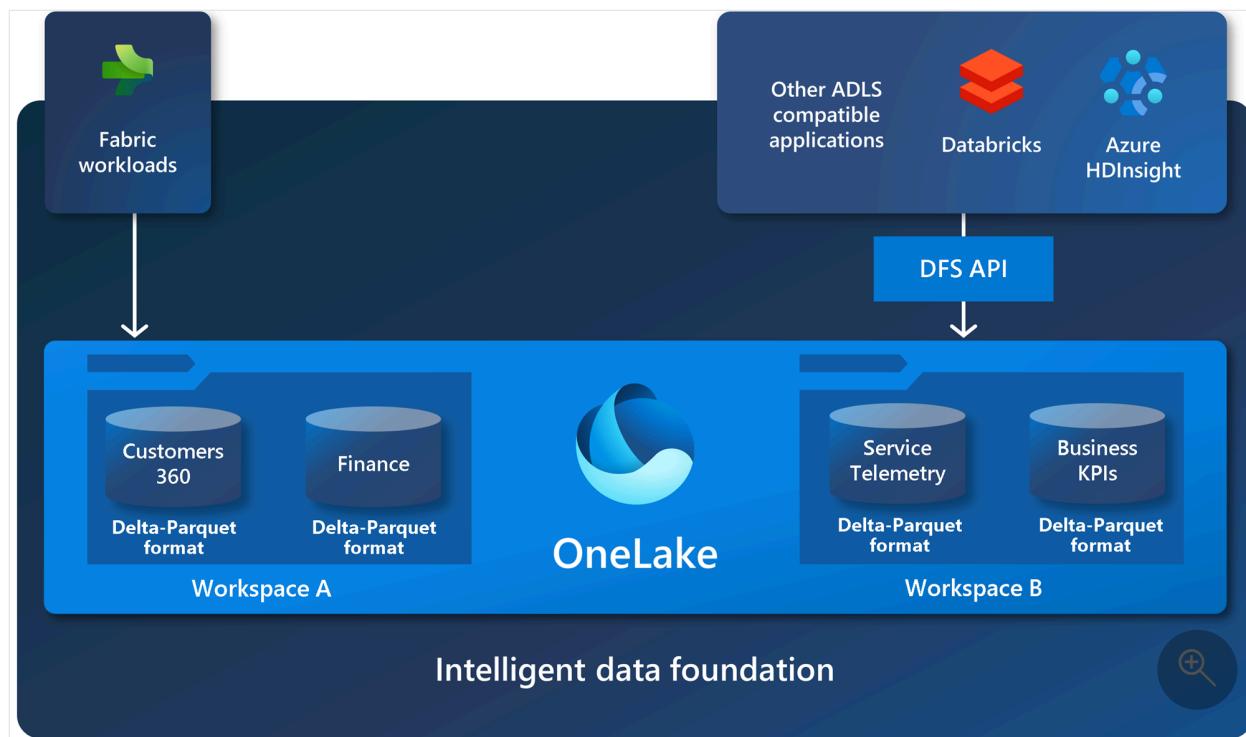
Within a workspace, you can create data items and you access all data in OneLake through data items. Similar to how Office stores Word, Excel, and PowerPoint files in OneDrive, Fabric stores lakehouses, warehouses, and other items in OneLake. Items can give tailored experiences for each persona, such the Apache Spark developer experience in a lakehouse.

For more information on how to get started using OneLake, see [Creating a lakehouse with OneLake](#).

Open at every level

OneLake is open at every level. OneLake is built on top of Azure Data Lake Storage (ADLS) Gen2 and can support any type of file, structured or unstructured. All Fabric data items like data warehouses and lakehouses store their data automatically in OneLake in Delta Parquet format. If a data engineer loads data into a lakehouse using Apache Spark, and then a SQL developer uses T-SQL to load data in a fully transactional data warehouse, both are contributing to the same data lake. OneLake stores all tabular data in Delta Parquet format.

OneLake supports the same ADLS Gen2 APIs and SDKs to be compatible with existing ADLS Gen2 applications, including Azure Databricks. You can address data in OneLake as if it's one big ADLS storage account for the entire organization. Every workspace appears as a container within that storage account, and different data items appear as folders within those containers.



For more information on APIs and endpoints, see [OneLake access and APIs](#). For examples of OneLake integrations with Azure, see [Azure Synapse Analytics](#), [Azure storage explorer](#), [Azure Databricks](#), and [Azure HDInsight](#) articles.

OneLake file explorer for Windows

OneLake is the OneDrive for data. Just like OneDrive, you can easily explore OneLake data from Windows using the [OneLake file explorer](#) for Windows. You can navigate all your workspaces and data items, easily uploading, downloading, or modifying files just like you do in Office. The OneLake file explorer simplifies working with data lakes, allowing even nontechnical business users to use them.

For more information, see [OneLake file explorer](#).

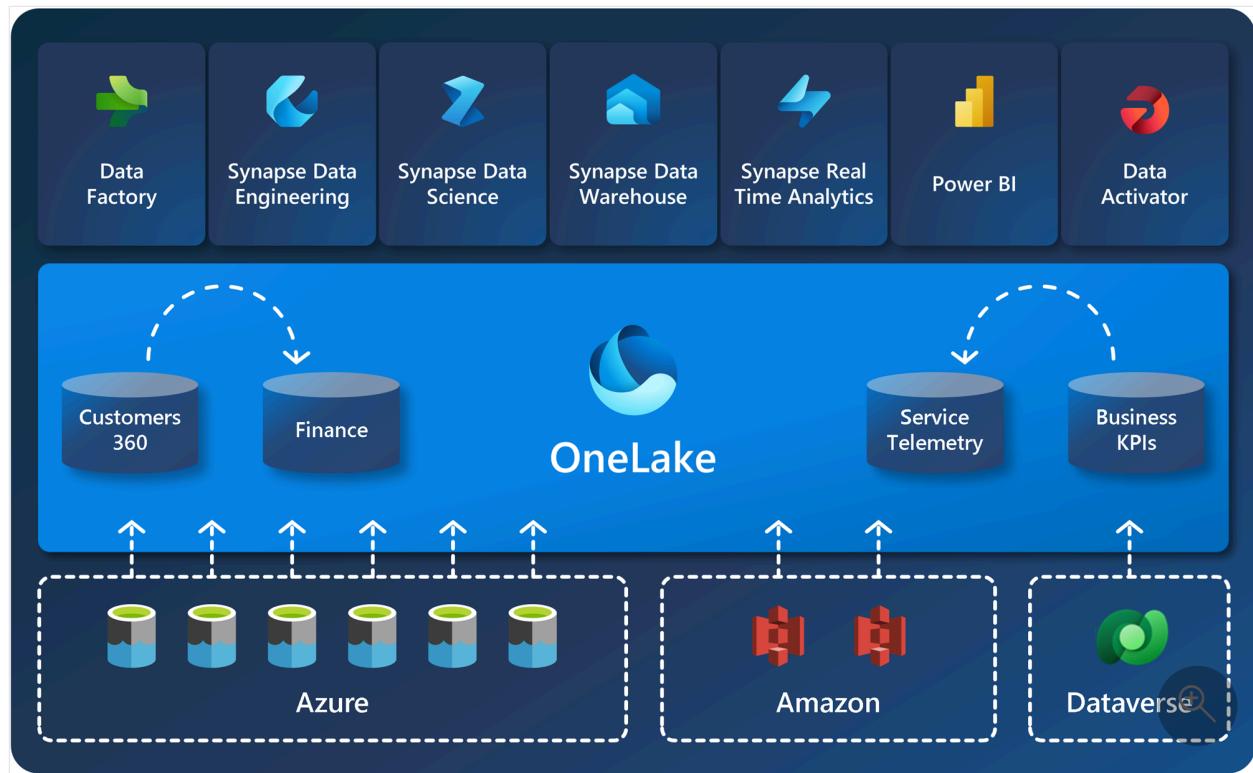
One copy of data

OneLake aims to give you the most value possible out of a single copy of data without data movement or duplication. You no longer need to copy data just to use it with another engine or to break down silos so you can analyze the data with data from other sources.

Shortcuts connect data across domains without data movement

Shortcuts allow your organization to easily share data between users and applications without having to move and duplicate information unnecessarily. When teams work independently in separate workspaces, shortcuts enable you to combine data across different business groups and domains into a virtual data product to fit a user's specific needs.

A shortcut is a reference to data stored in other file locations. These file locations can be within the same workspace or across different workspaces, within OneLake or external to OneLake in ADLS, S3, or Dataverse — with more target locations coming soon. No matter the location, shortcuts make files and folders look like you have them stored locally.



For more information on how to use shortcuts, see [OneLake shortcuts](#).

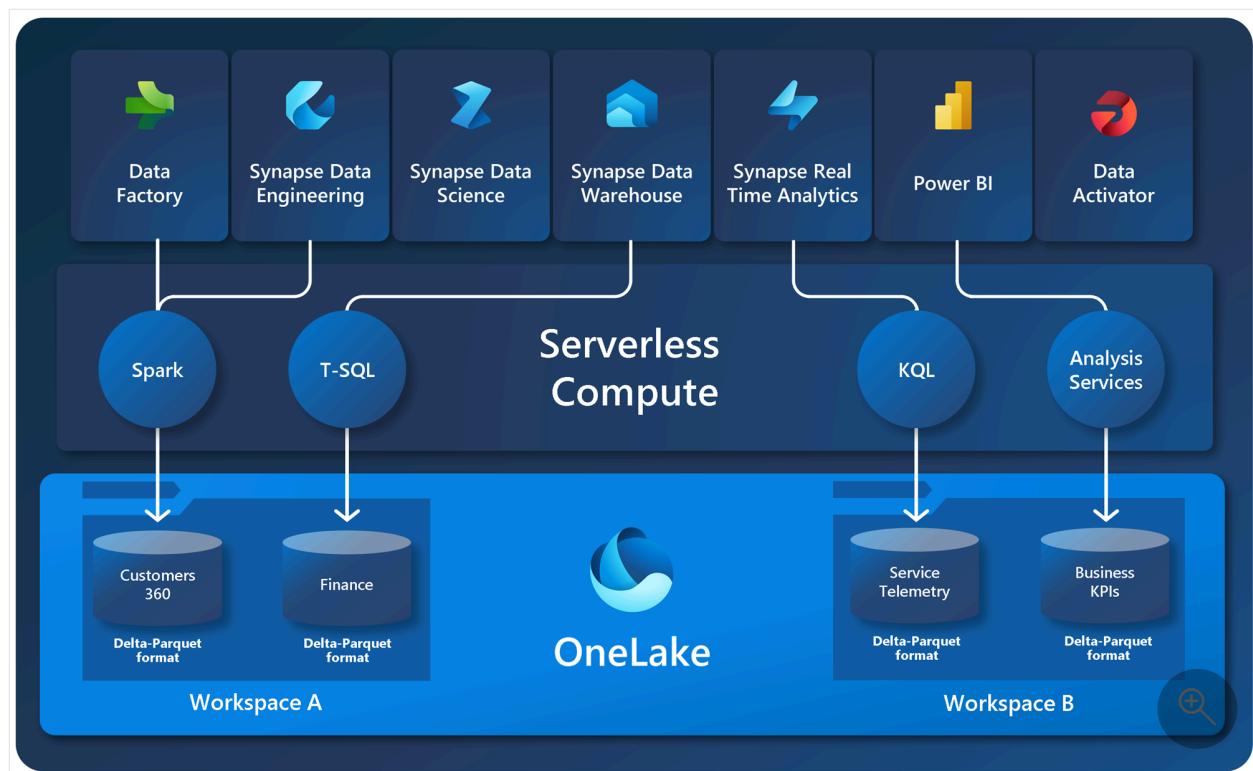
One copy of data with multiple analytical engines

While applications might have separation of storage and computing, the data is often optimized for a single engine, which makes it difficult to reuse the same data for multiple applications. With Fabric, the different analytical engines (T-SQL, Apache Spark, Analysis Services, etc.) store data in the open Delta Parquet format to allow you to use the same data across multiple engines.

There's no longer a need to copy data just to use it with another engine. You're always able to choose the best engine for the job that you're trying to do. For example, imagine you have a team of SQL engineers building a fully transactional data warehouse. They can use the T-SQL engine and all the power of T-SQL to create tables, transform data,

and load the data to tables. If a data scientist wants to make use of this data, they no longer need to go through a special Spark/SQL driver. OneLake stores all data in Delta Parquet format. Data scientists can use the full power of the Spark engine and its open-source libraries directly over the data.

Business users can build Power BI reports directly on top of OneLake using the new Direct Lake mode in the Analysis Services engine. The Analysis Services engine is what powers Power BI semantic models, and it has always offered two modes of accessing data: import and direct query. Direct Lake mode gives users all the speed of import without needing to copy the data, combining the best of import and direct query. For more information, see [Direct Lake](#).



Example diagram showing loading data using Spark, querying using T-SQL, and viewing the data in a Power BI report.

Related content

- [Creating a lakehouse with OneLake](#)

Feedback

Was this page helpful?

Yes

No

[Provide product feedback](#) | [Ask the community](#)

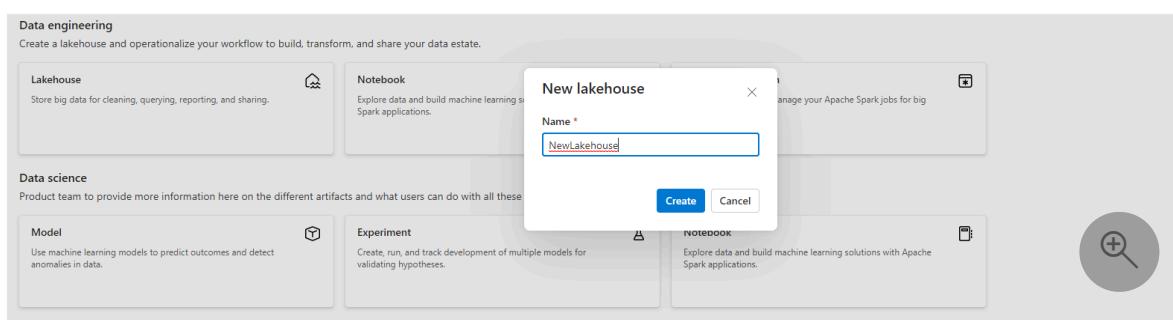
Bring your data to OneLake with Lakehouse

Article • 03/13/2025

This tutorial is a quick guide to creating a lakehouse and getting started with the basic methods of interacting with it. After completing this tutorial, you'll have a lakehouse provisioned inside of Microsoft Fabric working on top of OneLake.

Create a lakehouse

1. Sign in to [Microsoft Fabric](#).
2. Select **Workspaces** from the left-hand menu.
3. To open your workspace, enter its name in the search textbox located at the top and select it from the search results.
4. In the upper left corner of the workspace home page, select **New item** and then choose **Lakehouse** from the **Store data** section.
5. Give your lakehouse a name and select **Create**.

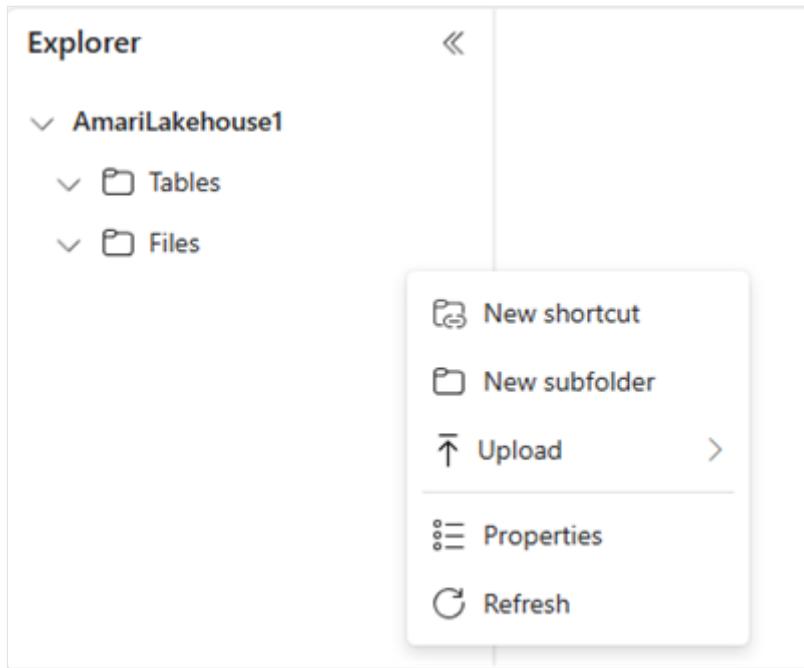


6. A new lakehouse is created and, if this lakehouse is your first OneLake item, OneLake is provisioned behind the scenes.

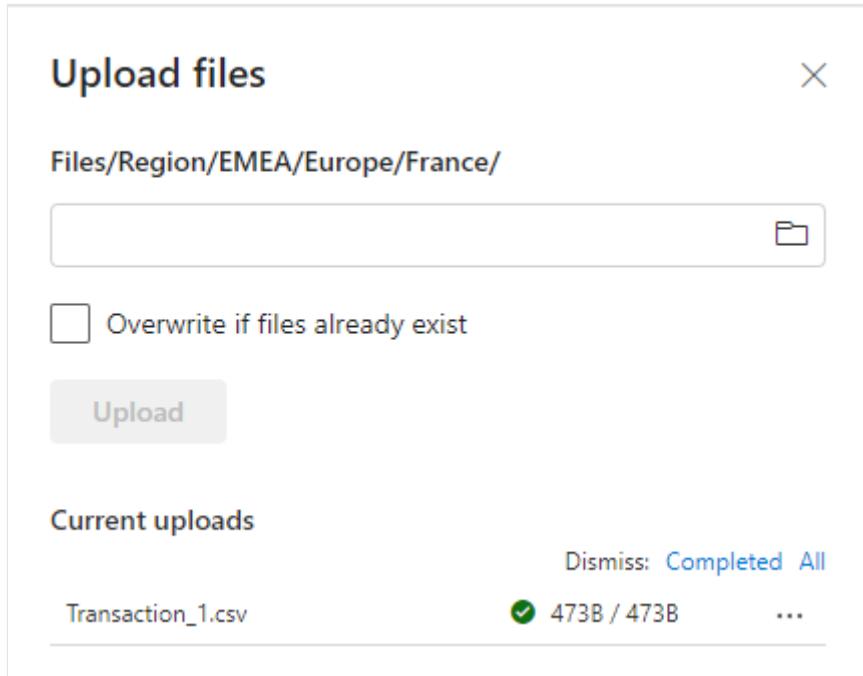
At this point, you have a lakehouse running on top of OneLake. Next, add some data and start organizing your lake.

Load data into a lakehouse

1. In the file browser on the left, select more options (...) next to **Files** and then select **New subfolder**. Name your subfolder and select **Create**.



2. You can repeat this step to add more subfolders as needed.
3. Select more options (...) next to your folder, and then select **Upload > Upload files** from the menu.
4. Choose the file you want from your local machine and then select **Upload**.



5. You now have data in OneLake. To add data in bulk or schedule data loads into OneLake, use the **Get data** button to create pipelines. Find more details about options for getting data in [Microsoft Fabric decision guide: copy activity, dataflow, or Spark](#).
6. Select more options (...) for the file you uploaded and select **Properties** from the menu.

The **Properties** screen shows the various details for the file, including the URL and Azure Blob File System (ABFS) path for use with Notebooks. You can copy the ABFS into a Fabric Notebook to query the data using Apache Spark. To learn more about notebooks in Fabric, see [Explore the data in your lakehouse with a notebook](#).

Now you have your first lakehouse with data stored in OneLake.

Related content

Learn how to connect to existing data sources with [OneLake shortcuts](#).

Feedback

Was this page helpful?



Yes



No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Transform data with Apache Spark and query with SQL

Article • 03/13/2025

In this guide, you will:

- Upload data to OneLake with the OneLake file explorer.
- Use a Fabric notebook to read data on OneLake and write back as a Delta table.
- Analyze and transform data with Spark using a Fabric notebook.
- Query one copy of data on OneLake with SQL.

Prerequisites

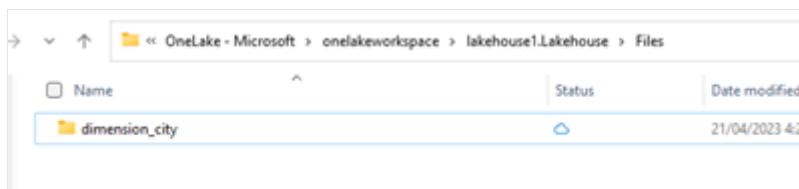
Before you begin, you must:

- Download and install [OneLake file explorer](#).
- Create a workspace with a Lakehouse item.
- Download the WideWorldImportersDW dataset. You can use [Azure Storage Explorer](#) to connect to
`https://fabRICTutorialData.blob.core.windows.net/sampledData/WideWorldImportersDW/csv/full/dimension_city` and download the set of csv files. Or you can use your own csv data and update the details as required.

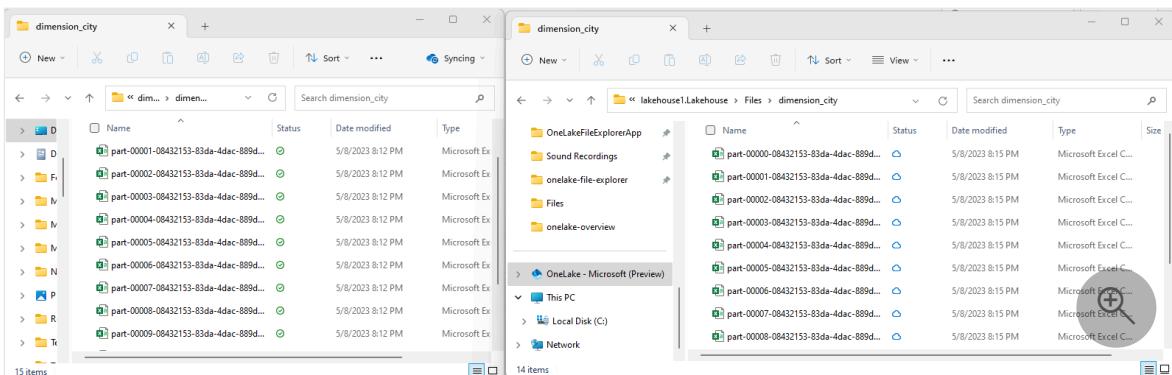
Upload data

In this section, you upload test data into your lakehouse using OneLake file explorer.

1. In OneLake file explorer, navigate to your lakehouse and create a subdirectory named `dimension_city` under the `/Files` directory.



2. Copy your sample csv files to the OneLake directory `/Files/dimension_city` using OneLake file explorer.



3. Navigate to your lakehouse in the Power BI or Fabric service and view your files.

| Name |
|--|
| part-00001-08432153-83da-4dac-889d-ca08c4a341e5-c000.csv |
| part-00002-08432153-83da-4dac-889d-ca08c4a341e5-c000.csv |
| part-00003-08432153-83da-4dac-889d-ca08c4a341e5-c000.csv |
| part-00004-08432153-83da-4dac-889d-ca08c4a341e5-c000.csv |
| part-00005-08432153-83da-4dac-889d-ca08c4a341e5-c000.csv |
| part-00006-08432153-83da-4dac-889d-ca08c4a341e5-c000.csv |

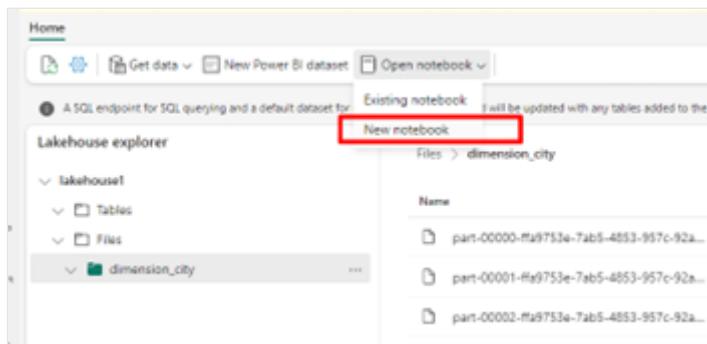
Create a Delta table

In this section, you convert the unmanaged CSV files into a managed table using Delta format.

Note

Always create, load, or create a shortcut to Delta-Parquet data *directly* under the **Tables** section of the lakehouse. Don't nest your tables in subfolders under the **Tables** section. The lakehouse doesn't recognize subfolders as tables and labels them as **Unidentified**.

1. In your lakehouse, select **Open notebook**, then **New notebook** to create a notebook.



2. Using the Fabric notebook, convert the CSV files to Delta format. The following code snippet reads data from user created directory `/Files/dimension_city` and converts it to a Delta table `dim_city`.

Copy the code snippet into the notebook cell editor. Replace the placeholders with your own workspace details, then select **Run cell** or **Run all**.

Python

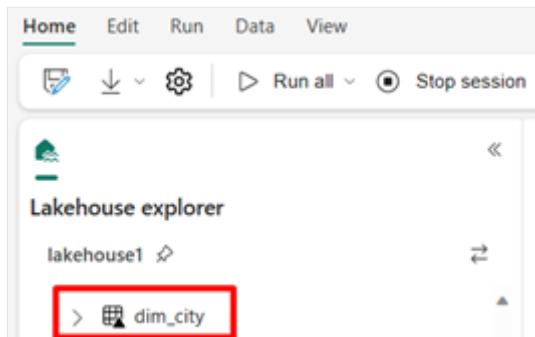
```
import os
from pyspark.sql.types import *
for filename in os.listdir("/lakehouse/default/Files/dimension_city"):

    df=spark.read.format('csv').options(header="true",inferSchema="true").load("abfss://<YOUR_WORKSPACE_NAME>@onelake.dfs.fabric.microsoft.com/<YOUR_LAKEHOUSE_NAME>.Lakehouse/Files/dimension_city/" +filename, on_bad_lines="skip")
    df.write.mode("overwrite").format("delta").save("Tables/dim_city")
```

💡 Tip

You can retrieve the full ABFS path to your directory by right-clicking on the directory name and selecting **Copy ABFS path**.

3. To see your new table, refresh your view of the `/Tables` directory. Select more options (...) next to the Tables directory, then select **Refresh**.



Query and modify data

In this section, you use a Fabric notebook to interact with the data in your table.

1. Query your table with SparkSQL in the same Fabric notebook.

```
Python

%%sql
SELECT * from <LAKEHOUSE_NAME>.dim_city LIMIT 10;
```

2. Modify the Delta table by adding a new column named **newColumn** with data type integer. Set the value of 9 for all the records for this newly added column.

```
Python

%%sql1

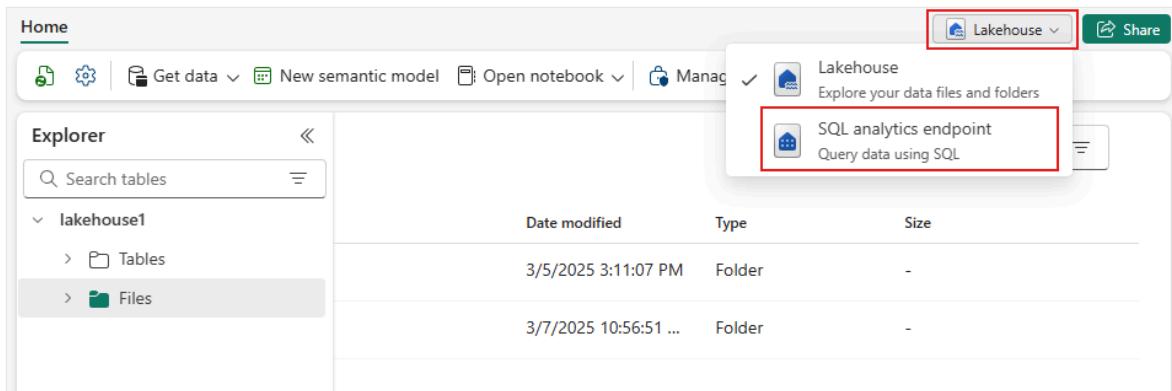
ALTER TABLE <LAKEHOUSE_NAME>.dim_city ADD COLUMN newColumn int;

UPDATE <LAKEHOUSE_NAME>.dim_city SET newColumn = 9;

SELECT City,newColumn FROM <LAKEHOUSE_NAME>.dim_city LIMIT 10;
```

You can also access any Delta table on OneLake via a SQL analytics endpoint. A SQL analytics endpoint references the same physical copy of Delta table on OneLake and offers the T-SQL experience.

1. Navigate to your lakehouse, then select **Lakehouse > SQL analytics endpoint** from the drop-down menu.



2. Select **New SQL Query** to query the table using T-SQL.
3. Copy and paste the following code into the query editor, then select **Run**.

```
SQL
```

```
SELECT TOP (100) * FROM [<LAKEHOUSE_NAME>].[dbo].[dim_city];
```

Related content

- Connect to ADLS using a OneLake shortcut
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Connect to ADLS and transform the data with Azure Databricks

Article • 11/29/2023

In this guide, you will:

- Create a Delta table in your Azure Data Lake Storage (ADLS) Gen2 account using Azure Databricks.
- Create a OneLake shortcut to a Delta table in ADLS.
- Use Power BI to analyze data via the ADLS shortcut.

Prerequisites

Before you start, you must have:

- A workspace with a Lakehouse item
- An Azure Databricks workspace
- An ADLS Gen2 account to store Delta tables

Create a Delta table, create a shortcut, and analyze the data

1. Using an Azure Databricks notebook, create a Delta table in your ADLS Gen2 account.

Python

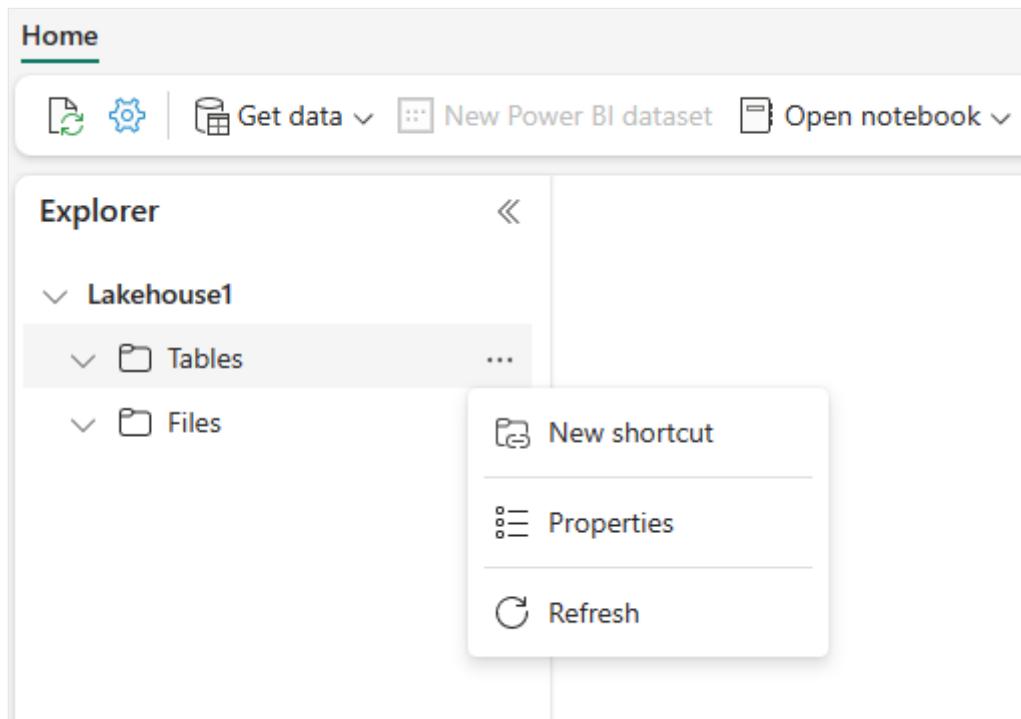
```
# Replace the path below to refer to your sample parquet data with
# this syntax "abfss://<storage name>@<container
# name>.dfs.core.windows.net/<filepath>"

# Read Parquet files from an ADLS account
df =
spark.read.format('Parquet').load("abfss://datasetsv1@olsdemo.dfs.core.
windows.net/demo/full/dimension_city/")

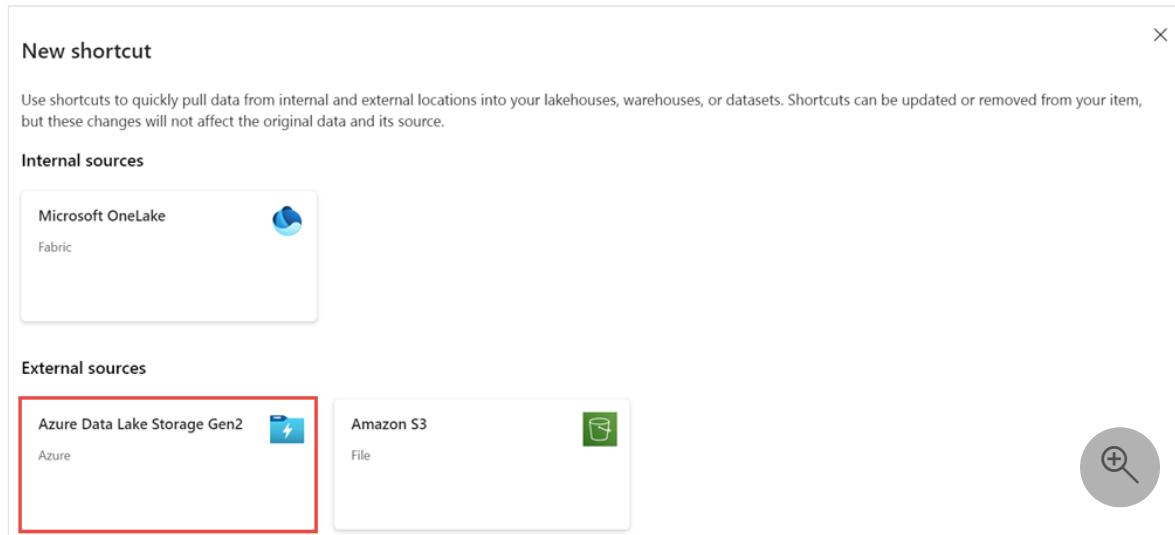
# Write Delta tables to ADLS account
```

```
df.write.mode("overwrite").format("delta").save("abfss://datasetsv1@olsdemo.dfs.core.windows.net/demo/adb_dim_city_delta/")
```

2. In your lakehouse, select the ellipses (...) next to **Tables** and then select **New shortcut**.



3. In the **New shortcut** screen, select the **Azure Data Lake Storage Gen2** tile.



4. Specify the connection details for the shortcut and select **Next**.

New shortcut

(i) Geography_Lakehouse is located in the region **West Central US**. Any data sourced through this shortcut will be processed in the same region.

 Azure Data Lake Storage
Gen2
Azure

Connection settings

URL * ⓘ
Example: https://contosoadlscdm.dfs.core.windows.net/file...

Connection credentials

Connection
Create new connection ⚡

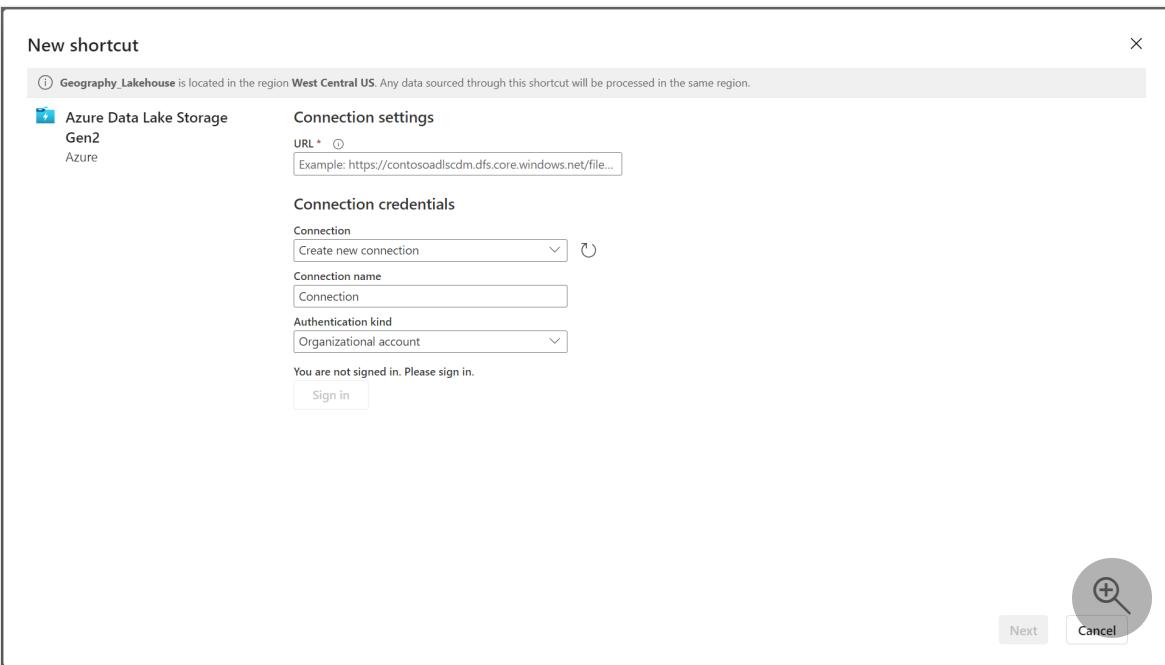
Connection name
Connection

Authentication kind
Organizational account

You are not signed in. Please sign in.
[Sign in](#)



Next Cancel



5. Specify the shortcut details. Provide a **Shortcut Name** and **Sub path** details and then select **Create**. The sub path should point to the directory where the Delta table resides.

New shortcut

(i) Geography_Lakehouse is located in the region **West Central US**. Any data sourced through this shortcut will be processed in the same region.

 Azure Data Lake Storage
Gen2
Azure

Shortcut settings

Shortcut Name *

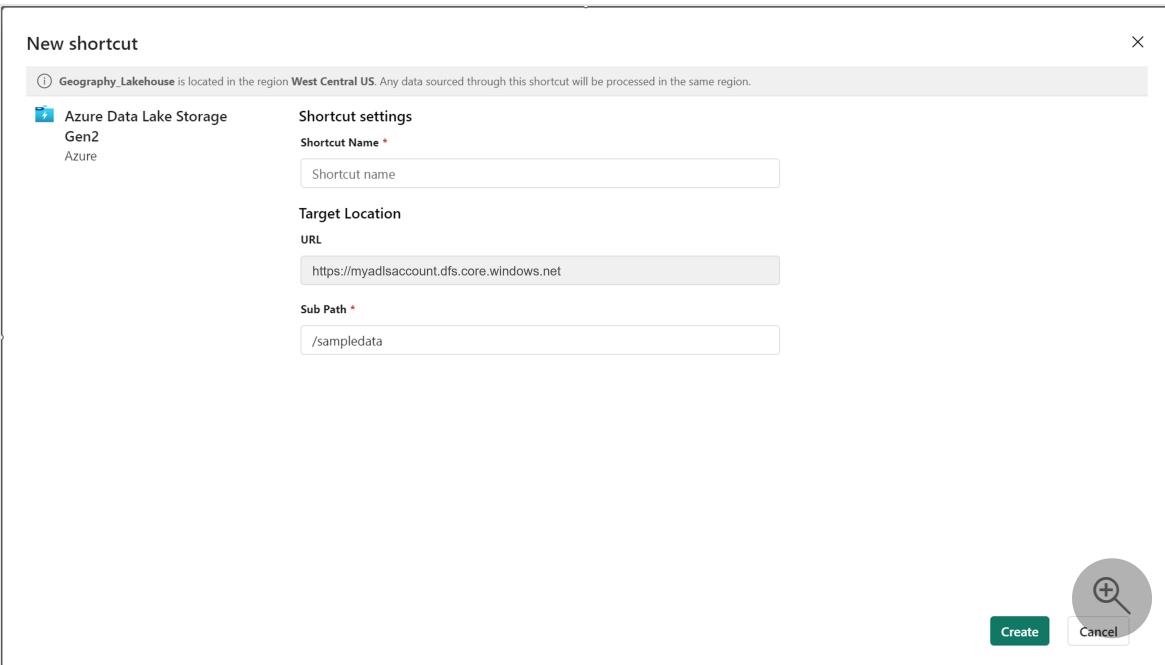
Target Location

URL

Sub Path *



Create Cancel



6. The shortcut appears as a Delta table under **Tables**.

The screenshot shows the Azure Data Lake Storage Explorer interface. In the left pane, under 'lakehouse1/Tables', the 'adls_shortcut_adb_dim_city_delta' table is selected and highlighted with a red box. Below it, other tables are listed: 'ABC CityKey', 'ABC WWICityID', 'ABC City', 'ABC StateProvince', 'ABC Country', and 'ABC Continent'. The right pane contains a vertical scroll bar.

7. You can now query this data directly from a notebook.

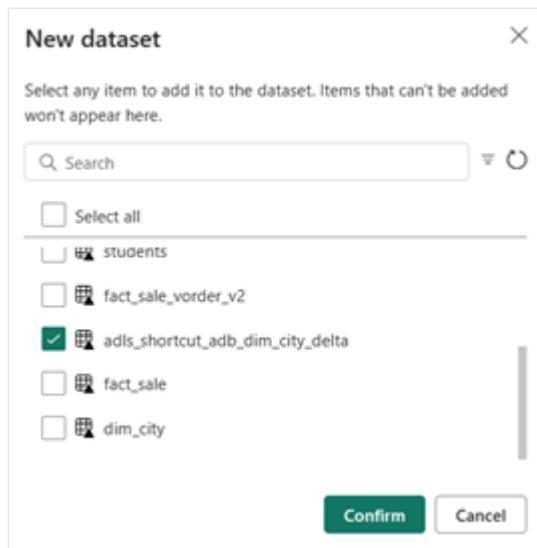
Python

```
df = spark.sql("SELECT * FROM  
lakehouse1.adls_shortcut_adb_dim_city_delta LIMIT 1000")  
display(df)
```

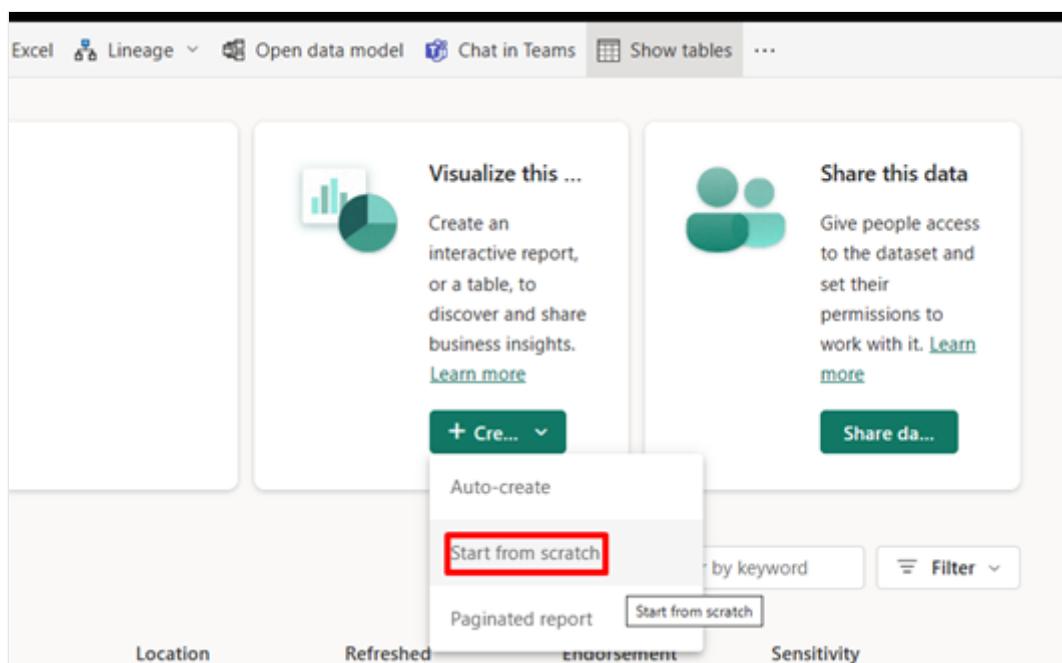
8. To access and analyze this Delta table via Power BI, select **New Power BI semantic model**.

The screenshot shows the Azure Data Lake Storage Explorer interface. At the top, the 'New Power BI dataset' button is highlighted with a red box. Below it, a message says: 'A SQL endpoint for SQL querying and a default dataset for reporting were created and will be updated automatically.' The 'Explorer' pane on the left shows the 'lakehouse1/Tables' section with the 'adls_shortcut_adb_dim_city_delta' table selected.

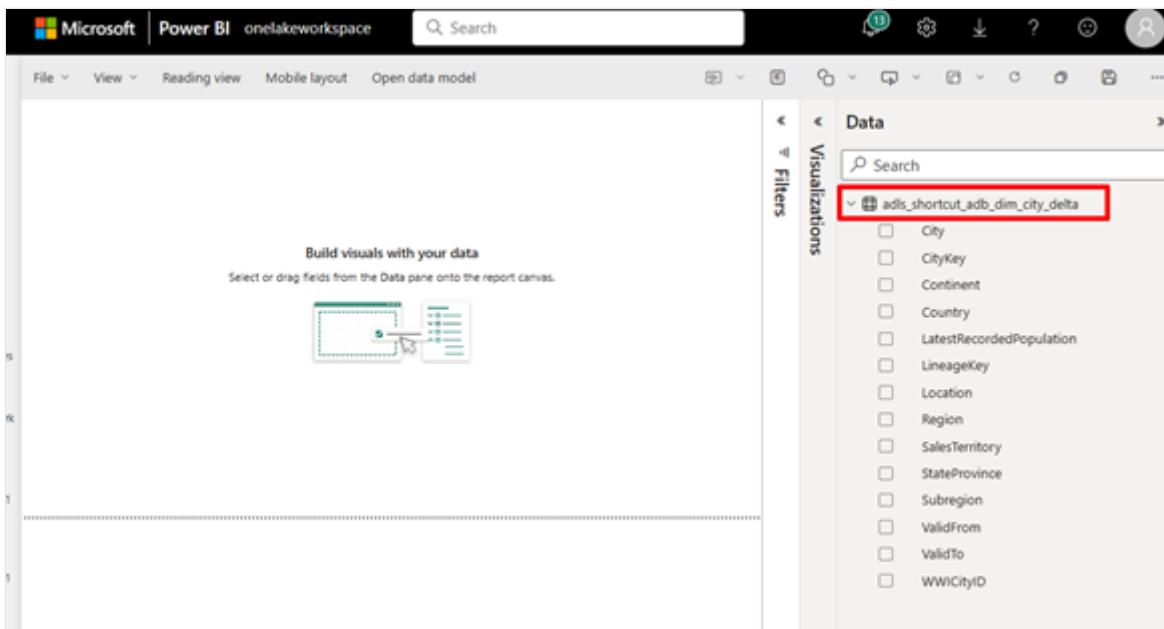
9. Select the shortcut and then select **Confirm**.



10. When the data is published, select **Start from scratch**.



11. In the report authoring experience, the shortcut data appears as a table along with all its attributes.



12. To build a Power BI report, drag the attributes to the pane on the left-hand side.

A screenshot of the Microsoft Power BI workspace. On the left, there is a table visual showing data for various cities across different countries and regions. The columns include City, Country, Location, Region, SalesTerritory, StateProvince, and Subregion. The data is represented as a grid of rows and columns. On the right, the 'Data' pane is open, showing the same dataset 'adls_shortcut_adb_dim_city_delta'. The attributes are listed with checkboxes next to them, indicating which ones are selected. The selected attributes are: City, Country, Location, Region, SalesTerritory, StateProvince, and Subregion. The 'Visualizations' pane is also visible on the far left.

Related content

- Ingest data into OneLake and analyze with Azure Databricks

Feedback

Was this page helpful?

Yes

No

Provide product feedback | Ask the community

Ingest data into OneLake and analyze with Azure Databricks

Article • 02/25/2025

In this guide, you will:

- Create a pipeline in a workspace and ingest data into your OneLake in Delta format.
- Read and modify a Delta table in OneLake with Azure Databricks.

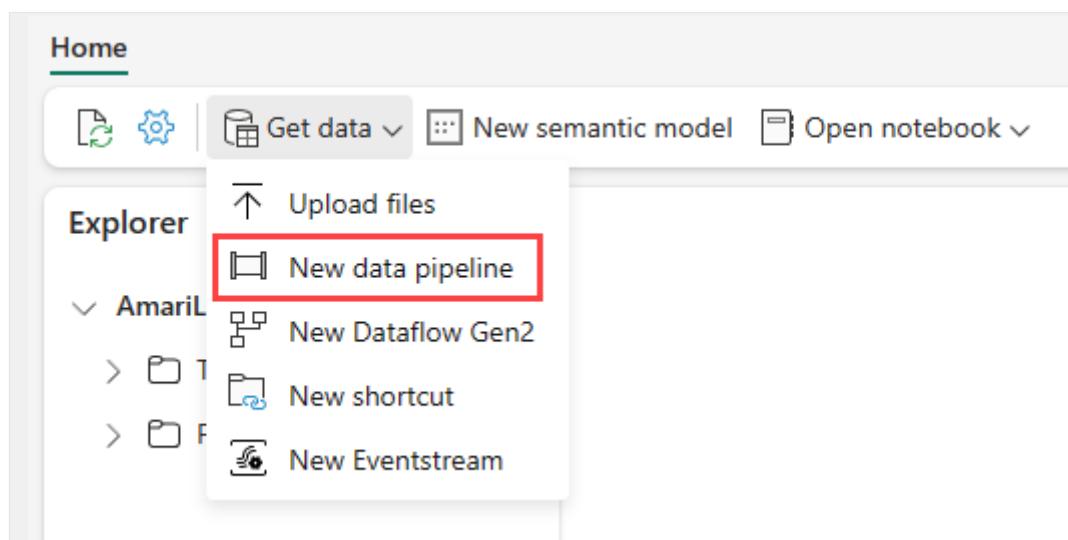
Prerequisites

Before you start, you must have:

- A workspace with a Lakehouse item.
- A premium Azure Databricks workspace. Only premium Azure Databricks workspaces support Microsoft Entra credential passthrough. When creating your cluster, enable Azure Data Lake Storage credential passthrough in the **Advanced Options**.
- A sample dataset.

Ingest data and modify the Delta table

1. Navigate to your lakehouse in the Power BI service and select **Get data** and then select **New data pipeline**.



2. In the New Pipeline prompt, enter a name for the new pipeline and then select **Create**.

3. For this exercise, select the **NYC Taxi - Green** sample data as the data source.

The screenshot shows the OneLake Sample data interface. On the left, a vertical navigation bar lists steps: Choose data source, Connect to data source, Choose data destination, Connect to data destination, and Review + save. The 'Choose data source' step is highlighted with a green circle. The main area displays a list of datasets. The 'NYC Taxi - Green' dataset is selected and highlighted with a red box. It is described as containing 2 GB of Parquet data. Other datasets shown include 'Diabetes' (14 KB Parquet), 'Public Holidays' (500 KB Parquet), and 'Retail Data Model from Wide World Importers' (352MB Parquet).

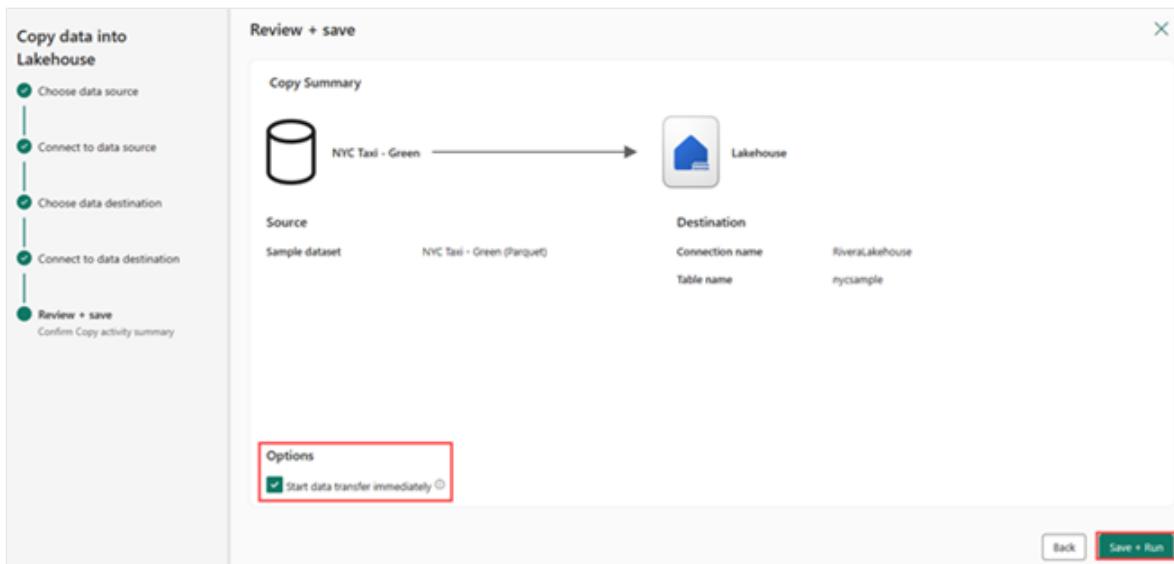
4. On the preview screen, select **Next**.

5. For data destination, select the name of the lakehouse you want to use to store the OneLake Delta table data. You can choose an existing lakehouse or create a new one.

The screenshot shows the 'Copy data into Lakehouse' preview screen. The left sidebar shows the progress: 'Choose data source' (checked), 'Connect to data source' (checked), 'Choose data destination' (checked), 'Connect to data destination' (unchecked), and 'Review + save' (unchecked). The central area shows a 'Lakehouse' icon and a 'Learn more' link. On the right, there are two radio button options: 'Existing Lakehouse' (selected) and 'Create new Lakehouse'. A dropdown menu under 'Lakehouse' shows 'lakehouse1' as the selected option. A 'Refresh' button is also present.

6. Select where you want to store the output. Choose **Tables** as the Root folder. Enter "nycsample" as the table name and select **Next**.

7. On the **Review + Save** screen, select **Start data transfer immediately** and then select **Save + Run**.



8. When the job is complete, navigate to your lakehouse and view the delta table listed under /Tables folder.
9. Right-click on the created table name, select **Properties**, and copy the Azure Blob Filesystem (ABFS) path.
10. Open your Azure Databricks notebook. Read the Delta table on OneLake.

Python

```
olsPath = "abfss://<replace with workspace
name>@onelake.dfs.fabric.microsoft.com/<replace with item
name>.Lakehouse/Tables/nycsample"
df=spark.read.format('delta').option("inferSchema","true").load(olsPath
)
df.show(5)
```

11. Update the Delta table data by changing a field value.

Python

```
%sql
update delta.`abfss://<replace with workspace
name>@onelake.dfs.fabric.microsoft.com/<replace with item
name>.Lakehouse/Tables/nycsample` set vendorID = 99999 where vendorID =
1;
```

Related content

- Transform data with Apache Spark and query with SQL

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Understand medallion lakehouse architecture for Microsoft Fabric with OneLake

The medallion lakehouse architecture, commonly known as *medallion architecture*, is a design pattern that's used to organize data in a lakehouse. It's the recommended design approach for Fabric. Since OneLake is the data lake for Fabric, medallion architecture is implemented by creating lakehouses in OneLake.

Medallion architecture comprises three distinct layers. The three medallion layers are: bronze (raw data), silver (enriched data), and gold (curated data). Each layer indicates the quality of data stored in the lakehouse, with higher levels representing higher quality.

Medallion architecture helps your data stay accurate and reliable according to the principles of atomicity, consistency, isolation, and durability (ACID). Your data starts in its raw form, and the original copies are preserved as a source of truth while your pipelines of validations and transformations prepares the data for analytics.

For more information, see [What is the medallion lakehouse architecture?](#).

Audience

This article introduces medallion lake architecture and describes how you can implement the design pattern in Microsoft Fabric. It's targeted at multiple audiences:

- **Data engineers:** Technical staff who design, build, and maintain infrastructures and systems that enable their organization to collect, store, process, and analyze large volumes of data.
- **Center of Excellence, IT, and BI teams:** The teams that are responsible for overseeing analytics throughout the organization.
- **Fabric administrators:** The administrators who are responsible for overseeing Fabric in the organization.

What is medallion architecture?

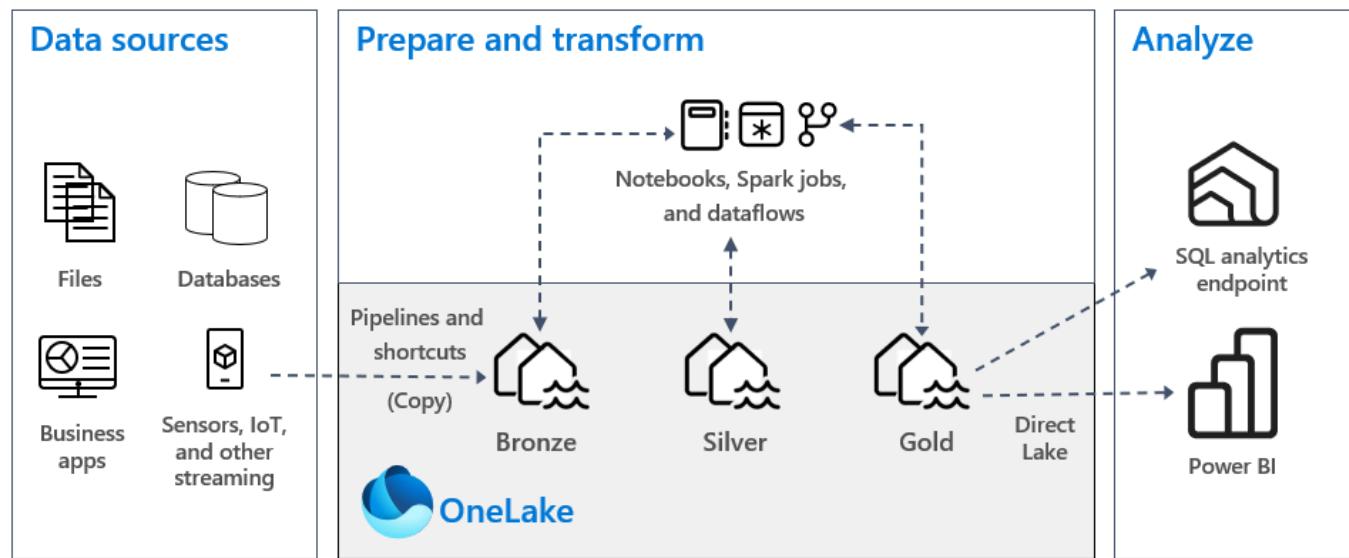
The goal of medallion architecture is to incrementally improve the structure and quality of data. Think of medallion architecture as a three-stage cleaning and organizing process for your data. Each layer makes your data more reliable and easier to use.

1. **Bronze (Raw):** Store everything exactly as it arrives. No changes are allowed.

2. **Silver (Enriched)**: Fix errors, standardize formats, and remove duplicates.

3. **Gold (Curated)**: Organize for reports and dashboards.

Keep each layer separated in its own lakehouse or data warehouse in OneLake, with data moving between the layers as it's transformed and refined.



In a typical medallion architecture implementation in Fabric, the bronze layer stores the data in the same format as the data source. When the data source is a relational database, Delta tables are a good choice. The silver and gold layers should contain Delta tables.

Tip

To learn how to create a lakehouse, work through the [Lakehouse end-to-end scenario](#) tutorial.

Real-world example

Consider the following example of an e-commerce company that applies medallion architecture to its data:

Bronze Layer:

- Store raw sales data from website (JSON)
- Store raw inventory data from warehouse (CSV)
- Store raw customer data from CRM (SQL export)

Silver Layer:

- Standardize date formats across all sources
- Convert all currency to USD

- Remove test transactions
- Match customer records across systems

Gold Layer:

- Create daily sales dashboard table
- Build customer lifetime value table
- Generate inventory forecasting table

Medallion architecture in OneLake

The basis of a modern data warehouse is a data lake. Microsoft OneLake is a single, unified, logical data lake for your entire organization. It comes automatically provisioned with every Fabric tenant, and it's the single location for all your analytics data.

To store data in OneLake, you create a *lakehouse* in Fabric. A lakehouse is a data architecture platform for storing, managing, and analyzing structured and unstructured data in a single location. It can scale to large data volumes of all file types and sizes, and because the data is stored in a single location, it can be shared and reused across the organization.

For more information, see [What is a lakehouse in Microsoft Fabric?](#).

Tables and files

When you create a lakehouse in OneLake, two physical storage locations are provisioned automatically:

- **Tables** stores tables of all formats in Apache Spark (CSV, Parquet, or Delta).
- **Files** stores data in any file format. If you want to create a table based on data in the files area, you can create a [shortcut](#) that points to the folder that contains the table files.

In the bronze layer, you store data in its original format, which might be either tables or files. If the source data is from OneLake, Azure Data Lake Store Gen2 (ADLS Gen2), Amazon S3, or Google, create a shortcut in the bronze layer instead of copying the data across.

In the silver and gold layers, you typically store data in Delta tables. However, you can also store data in Parquet or CSV files. If you do that, you must explicitly create a shortcut or an external table with a location that points to the unmanaged folder that contains the Delta Lake files in Apache Spark.

In Microsoft Fabric, the [Lakehouse explorer](#) provides a unified graphical representation of the whole Lakehouse for users to navigate, access, and update their data.

Delta Lake storage

Delta Lake is an optimized storage layer that provides the foundation for storing data and tables. It supports ACID transactions for big data workloads, and for this reason it's the default storage format in a Fabric lakehouse.

Delta Lake delivers reliability, security, and performance in the lakehouse for both streaming and batch operations. Internally, it stores data in Parquet file format, however, it also maintains transaction logs and statistics that provide features and performance improvement over the standard Parquet format.

Delta Lake format delivers the following benefits compared to generic file formats:

- Support for ACID properties, especially durability to prevent data corruption.
- Faster read queries.
- Increased data freshness.
- Support for both batch and streaming workloads.
- Support for data rollback by using [Delta Lake time travel](#).
- Enhanced regulatory compliance and audit by using [Delta Lake table history](#).

Fabric standardizes storage file format with Delta Lake. By default, every workload engine in Fabric creates Delta tables when you write data to a new table. For more information, see [Lakehouse and Delta Lake tables](#).

Deployment model

To implement medallion architecture in Fabric, you can either use lakehouses (one for each layer), a data warehouse, or combination of both. Your decision should be based on your preference and the expertise of your team. With Fabric, you can use different analytic engines that work on the one copy of your data in OneLake.

Here are two patterns to consider:

- **Pattern 1:** Create each layer as a lakehouse. In this case, business users access data by using the SQL analytics endpoint.
- **Pattern 2:** Create the bronze and silver layers as lakehouses, and the gold layer as a data warehouse. In this case, business users access data by using the data warehouse endpoint.

While you can create all lakehouses in a single [Fabric workspace](#), we recommend that you create each lakehouse in its own, separate workspace. This approach provides you with more control and better governance at the layer level.

For the bronze layer, we recommend that you store the data in its original format, or use Parquet or Delta Lake. Whenever possible, keep the data in its original format. If the source data is from OneLake, Azure Data Lake Store Gen2 (ADLS Gen2), Amazon S3, or Google, create a [shortcut](#) in the bronze layer instead of copying the data across.

For the silver and gold layers, we recommend that you use Delta tables because of the extra capabilities and performance enhancements they provide. Fabric standardizes on Delta Lake format, and by default every engine in Fabric writes data in this format. Further, these engines use V-Order write-time optimization to the Parquet file format. That optimization enables fast reads by Fabric compute engines, such as Power BI, SQL, Apache Spark, and others. For more information, see [Delta Lake table optimization and V-Order](#).

Lastly, today many organizations face massive growth in data volumes, together with an increasing need to organize and manage that data in a logical way while facilitating more targeted and efficient use and governance. That can lead you to establish and manage a decentralized or federated data organization with governance. To meet this objective, consider implementing a *data mesh architecture*. [Data mesh](#) is an architectural pattern that focuses on creating data domains that offer data as a product.

You can create a data mesh architecture for your data estate in Fabric by creating data domains. You might create domains that map to your business domains, for example, marketing, sales, inventory, human resources, and others. You can then implement medallion architecture by setting up data layers within each of your domains. For more information about domains, see [Domains](#).

Use materialized lake views for medallion architecture

[Materialized lake views](#) in Microsoft Fabric help you to implement medallion architecture in your lakehouse. Rather than building complex pipelines to transform data between bronze, silver, and gold layers, you can define materialized lake views that automatically manage the transformations.

Key benefits of using materialized lake views for medallion architecture include:

- **Declarative pipelines:** Define data transformations using SQL statements rather than building manual pipelines between layers.
- **Automatic dependency management:** Fabric automatically determines the correct execution order based on view dependencies.
- **Data quality rules:** Built-in support for defining and enforcing data quality constraints as data moves through layers.
- **Optimal refresh:** The system automatically determines whether to perform incremental, full, or no refresh for each view.

- **Visualization and monitoring:** View lineage across all layers and track execution progress.

For example, you can create a silver layer view that cleanses and joins data from bronze tables, and then create gold layer views that aggregate the silver layer data for reporting. The system handles the refresh orchestration automatically.

For more information, see [Implement medallion architecture with materialized lake views](#).

Understand Delta table data storage

This section describes other guidance related to implementing a medallion lakehouse architecture in Fabric.

File size

Generally, a big data platform performs better when it has a few large files rather than many small files. Performance degradation occurs when the compute engine has many metadata and file operations to manage. For better query performance, we recommend that you aim for data files that are approximately 1 GB in size.

Delta Lake has a feature called *predictive optimization*. Predictive optimization automates maintenance operations for Delta tables. When this feature is enabled, Delta Lake identifies tables that would benefit from maintenance operations and then optimizes their storage. While this feature should form part of your operational excellence and your data preparation work, Fabric can optimize data files during data write, too. For more information, see [Predictive optimization for Delta Lake](#).

Historical retention

By default, Delta Lake maintains a history of all changes made, so the size of historical metadata grows over time. Based on your business requirements, keep historical data only for a certain period of time to reduce your storage costs. Consider retaining historical data for only the last month, or other appropriate period of time.

You can remove older historical data from a Delta table by using the [VACUUM command](#). However, by default you can't delete historical data within the last seven days. That restriction maintains the consistency in data. Configure the default number of days with the table property `delta.deletedFileRetentionDuration = "interval <interval>"`. That property determines the period of time that a file must be deleted before it can be considered a candidate for a vacuum operation.

Table partitions

When you store data in each layer, we recommended that you use a partitioned folder structure wherever applicable. This technique improves data manageability and query performance. Generally, partitioned data in a folder structure results in faster search for specific data entries because of partition pruning/elimination.

Typically, you append data to your target table as new data arrives. However, in some cases you might merge data because you need to update existing data at the same time. In that case, you can perform an *upsert* operation by using the [MERGE command](#). When your target table is partitioned, be sure to use a partition filter to speed up the operation. That way, the engine can eliminate partitions that don't require updating.

Data access

You should plan and control who needs access to specific data in the lakehouse. You should also understand the various transaction patterns they're going to use while accessing this data. You can then define the right table partitioning scheme, and data collocation with Delta Lake [Z-order indexes](#).

Related content

For more information about implementing medallion lakehouse architecture, see the following resources.

- [Tutorial: Lakehouse end-to-end scenario](#)
- [Tutorial: Implement medallion architecture with materialized lake views](#)
- [Lakehouse and Delta Lake tables](#)
- [Microsoft Fabric decision guide: choose a data store](#)
- [The need for optimize write on Apache Spark](#)
- Questions? Try asking the [Fabric community](#).
- Suggestions? [Contribute ideas to improve Fabric](#).

ⓘ Note: The author created this article with assistance from AI. [Learn more](#)

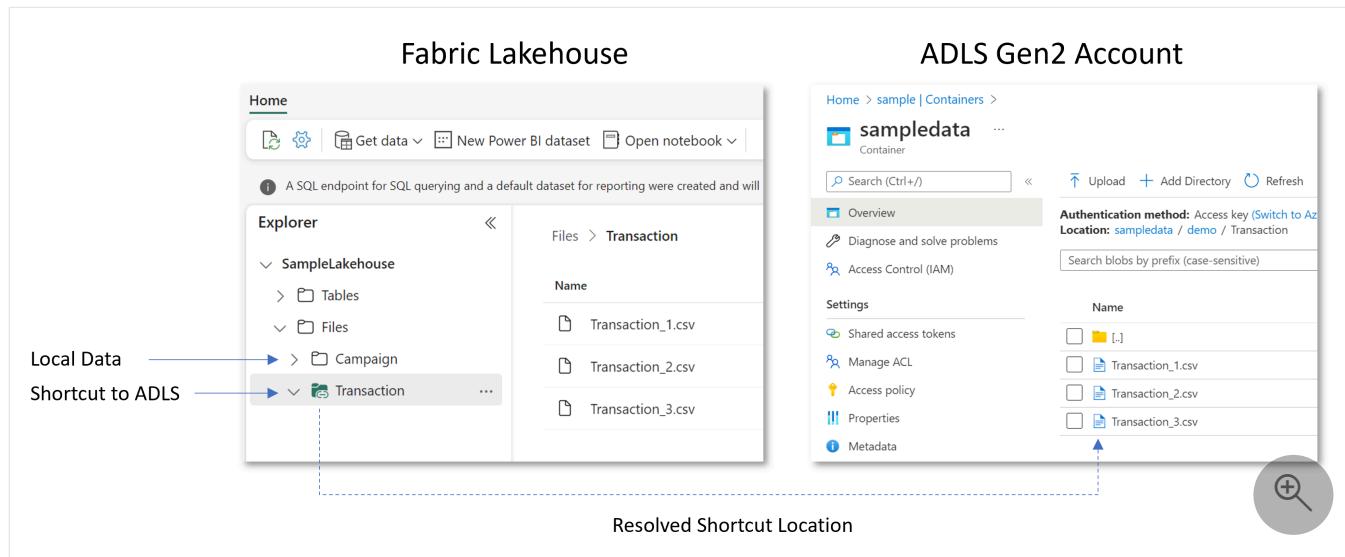
OneLake shortcuts

Article • 05/19/2025

Shortcuts in Microsoft OneLake allow you to unify your data across domains, clouds, and accounts by creating a single virtual data lake for your entire enterprise. All Fabric experiences and analytical engines can directly connect to your existing data sources such as Azure, Amazon Web Services (AWS), and OneLake through a unified namespace. OneLake manages all permissions and credentials, so you don't need to separately configure each Fabric workload to connect to each data source. Additionally, you can use shortcuts to eliminate edge copies of data and reduce process latency associated with data copies and staging.

What are shortcuts?

Shortcuts are objects in OneLake that point to other storage locations. The location can be internal or external to OneLake. The location that a shortcut points to is known as the target path of the shortcut. The location where the shortcut appears is known as the shortcut path. Shortcuts appear as folders in OneLake and any workload or service that has access to OneLake can use them. Shortcuts behave like symbolic links. They're an independent object from the target. If you delete a shortcut, the target remains unaffected. If you move, rename, or delete a target path, the shortcut can break.



Where can I create shortcuts?

You can create shortcuts in lakehouses and Kusto Query Language (KQL) databases. Furthermore, the shortcuts you create within these items can point to other OneLake locations, Azure Data Lake Storage (ADLS) Gen2, Amazon S3 storage accounts, or Dataverse. You can even [create shortcuts to on-premises or network-restricted locations](#) with the use of the Fabric on-premises data gateway (OPDG).

You can use the Fabric UI to create shortcuts interactively, and you can use the [REST API](#) to create shortcuts programmatically.

Lakehouse

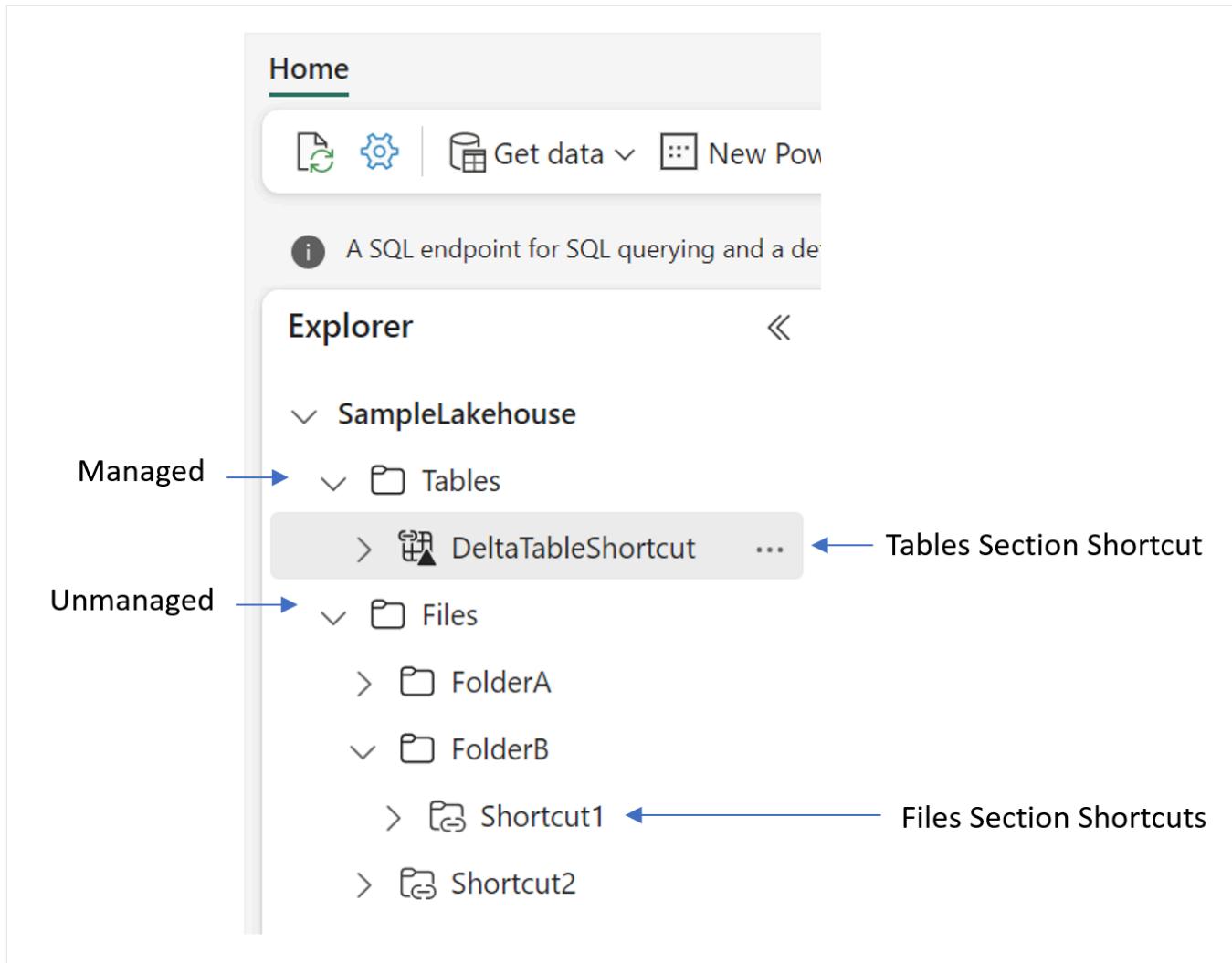
When creating shortcuts in a lakehouse, you must understand the folder structure of the item. Lakehouses are composed of two top-level folders: the **Tables** folder and the **Files** folder. The **Tables** folder represents the managed portion of the lakehouse for structured datasets. While the **Files** folder is the unmanaged portion of the lakehouse for unstructured or semi-structured data.

In the **Tables** folder, you can only create shortcuts at the top level. Shortcuts aren't supported in subdirectories of the **Tables** folder. Shortcuts in the tables section typically point to internal sources within OneLake or link to other data assets that conform to the Delta table format. If the target of the shortcut contains data in the Delta\Parquet format, the lakehouse automatically synchronizes the metadata and recognizes the folder as a table. Shortcuts in the tables section can link to either a single table or a schema, which is a parent folder for multiple tables.

 **Note**

The Delta format doesn't support tables with space characters in the name. Any shortcut containing a space in the name won't be discovered as a Delta table in the lakehouse.

In the **Files** folder, there are no restrictions on where you can create shortcuts. You can create them at any level of the folder hierarchy. Table discovery doesn't happen in the **Files** folder. Shortcuts here can point to both internal (OneLake) and external storage systems with data in any format.

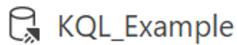


KQL database

When you create a shortcut in a KQL database, it appears in the **Shortcuts** folder of the database. The KQL database treats shortcuts like external tables. To query the shortcut, use the `external_table` function of the Kusto Query Language.

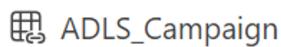
Data tree

Database

 Search

> Tables

Shortcuts



> Materialized views

> Functions

> Data streams

Database / Shortcuts

Name

S3_Campaign

ADLS_Campaign

Where can I access shortcuts?

Any Fabric or non-Fabric service that can access data in OneLake can use shortcuts. Shortcuts are transparent to any service accessing data through the OneLake API. Shortcuts just appear as another folder in the lake. Apache Spark, SQL, Real-Time Intelligence, and Analysis Services can all use shortcuts when querying data.

Apache Spark

Apache Spark notebooks and Apache Spark jobs can use shortcuts that you create in OneLake. Relative file paths can be used to directly read data from shortcuts. Additionally, if you create a shortcut in the **Tables** section of the lakehouse and it is in the Delta format, you can read it as a managed table using Apache Spark SQL syntax.

Python

```
df = spark.read.format("delta").load("Tables/MyShortcut")
display(df)
```

Python

```
df = spark.sql("SELECT * FROM MyLakehouse.MyShortcut LIMIT 1000")
display(df)
```

SQL

You can read shortcuts in the **Tables** section of a lakehouse through the SQL analytics endpoint for the lakehouse. You can access the SQL analytics endpoint through the mode selector of the lakehouse or through SQL Server Management Studio (SSMS).

SQL

```
SELECT TOP (100) *
FROM [MyLakehouse].[dbo].[MyShortcut]
```

Real-Time Intelligence

Shortcuts in KQL databases are recognized as external tables. To query the shortcut, use the `external_table` function of the Kusto Query Language.

Kusto

```
external_table('MyShortcut')
| take 100
```

Analysis Services

You can create semantic models for lakehouses containing shortcuts in the **Tables** section of the lakehouse. When the semantic model runs in Direct Lake mode, Analysis Services can read data directly from the shortcut.

Non-Fabric

Applications and services outside of Fabric can also access shortcuts through the OneLake API. OneLake supports a subset of the ADLS Gen2 and Blob storage APIs. To learn more about the OneLake API, see [OneLake access with APIs](#).

HTTP

```
https://onelake.dfs.fabric.microsoft.com/MyWorkspace/MyLakehouse/Tables/MyShortcut/
```

Types of shortcuts

OneLake shortcuts support multiple filesystem data sources. These include internal OneLake locations, Azure Data Lake Storage (ADLS) Gen2, Amazon S3, S3 Compatible, Google Cloud Storage(GCS) and Dataverse.

Internal OneLake shortcuts

Internal OneLake shortcuts allow you to reference data within existing Fabric items, including:

- KQL databases
- Lakehouses
- Mirrored Azure Databricks Catalogs
- Mirrored Databases
- Semantic models
- SQL databases
- Warehouses
- Warehouse snapshots

The shortcut can point to a folder location within the same item, across items within the same workspace, or even across items in different workspaces. When you create a shortcut across items, the item types don't need to match. For instance, you can create a shortcut in a lakehouse that points to data in a data warehouse.

When a user accesses data through a shortcut to another OneLake location, OneLake uses the identity of the calling user to authorize access to the data in the target path of the shortcut. This user must have permissions in the target location to read the data.

 **Important**

When users access shortcuts through Power BI semantic models or T-SQL, the calling user's identity is not passed through to the shortcut target. The calling item owner's identity is passed instead, delegating access to the calling user.

Azure Data Lake Storage shortcuts

When you create shortcuts to Azure Data Lake Storage (ADLS) Gen2 storage accounts, the target path can point to any folder within the hierarchical namespace. At a minimum, the target

path must include a container name.

(!) Note

You must have hierarchical namespaces enabled on your ADLS Gen 2 storage account.

Access

ADLS shortcuts must point to the DFS endpoint for the storage account.

Example: `https://accountname.dfs.core.windows.net/`

If your storage account is protected by a storage firewall, you can configure trusted service access. For more information, see [Trusted workspace access](#)

Authorization

ADLS shortcuts use a delegated authorization model. In this model, the shortcut creator specifies a credential for the ADLS shortcut and all access to that shortcut is authorized using that credential. ADLS shortcuts support the following delegated authorization types:

- **Organizational account** - must have Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner role on the storage account; or Delegator role on the storage account plus file or directory access granted within the storage account.
- **Service principal** - must have Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner role on the storage account; or Delegator role on the storage account plus file or directory access granted within the storage account.
- **Workspace identity** - must have Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner role on the storage account; or Delegator role on the storage account plus file or directory access granted within the storage account.
- **Shared Access Signature (SAS)** - must include at least the following permissions: Read, List, and Execute.

Microsoft Entra ID delegated authorization types (organizational account, service principal, or workspace identity) require the **Generate a user delegation key** action at the storage account level. This action is included as part of the Storage Blob Data Reader, Storage Blob Data Contributor, Storage Blob Data Owner, and Delegator roles. If you don't want to give a user reader, contributor, or owner permissions for the whole storage account, assign them the Delegator role instead. Then, define detailed data access rights using [Access control lists \(ACLs\) in Azure Data Lake Storage](#).

Important

Currently, when workspace identity is used as the delegated authorization type for an ADLS shortcut, users can authenticate directly to the storage account without needing to create a delegation key. However, this behavior will be restricted in the future. We recommend making sure that all users have the **Generate a user delegation key** action to ensure that your users' access isn't affected when this behavior changes.

Azure Blob Storage shortcuts

Access

Azure Blob Storage shortcut can point to the account name or URL for the Storage account.

Example: `accountname` or `https://accountname.blob.core.windows.net/`

Authorization

Blob storage shortcuts use a delegated authorization model. In this model, the shortcut creator specifies a credential for the shortcut and all access to that shortcut is authorized using that credential. Blob shortcuts support the following delegated authorization types:

- **Organizational account** - must have Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner role on the storage account; or Delegator role on the storage account plus file or directory access granted within the storage account.
- **Service principal** - must have Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner role on the storage account; or Delegator role on the storage account plus file or directory access granted within the storage account.
- **Workspace identity** - must have Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner role on the storage account; or Delegator role on the storage account plus file or directory access granted within the storage account.
- **Shared Access Signature (SAS)** - must include at least the following permissions: Read, List, and Execute.

S3 shortcuts

When you create shortcuts to Amazon S3 accounts, the target path must contain a bucket name at a minimum. S3 doesn't natively support hierarchical namespaces but you can use prefixes to mimic a directory structure. You can include prefixes in the shortcut path to further

narrow the scope of data accessible through the shortcut. When you access data through an S3 shortcut, prefixes are represented as folders.

! Note

S3 shortcuts are read-only. They don't support write operations regardless of the user's permissions.

Access

S3 shortcuts must point to the https endpoint for the S3 bucket.

Example: `https://bucketname.s3.region.amazonaws.com/`

! Note

You don't need to disable the S3 Block Public Access setting for your S3 account for the S3 shortcut to function.

Access to the S3 endpoint must not be blocked by a storage firewall or Virtual Private Cloud.

Authorization

S3 shortcuts use a delegated authorization model. In this model, the shortcut creator specifies a credential for the S3 shortcut and all access to that shortcut is authorized using that credential. The supported delegated credential is a key and secret for an IAM user.

The IAM user must have the following permissions on the bucket that the shortcut is pointing to:

- `S3:GetObject`
- `S3:GetBucketLocation`
- `S3>ListBucket`

S3 shortcuts support S3 buckets that use S3 Bucket Keys for SSE-KMS encryption. To access data encrypted with SSE-KMS encryption, the user must have encrypt/decrypt permissions for the bucket key, otherwise they receive a "Forbidden" error (403). For more information, see [Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects](#).

Google Cloud Storage shortcuts

Shortcuts can be created to Google Cloud Storage(GCS) using the XML API for GCS. When you create shortcuts to Google Cloud Storage, the target path must contain a bucket name at a minimum. You can also restrict the scope of the shortcut by further specifying the prefix/folder you want to point to within the storage hierarchy.

! Note

GCS shortcuts are read-only. They don't support write operations regardless of the user's permissions.

Access

When configuring the connection for a GCS shortcut, you can either specify the global endpoint for the storage service or use a bucket-specific endpoint.

- Global endpoint example: `https://storage.googleapis.com`
- Bucket-specific endpoint example: `https://<BucketName>.storage.googleapis.com`

Authorization

GCS shortcuts use a delegated authorization model. In this model, the shortcut creator specifies a credential for the GCS shortcut and all access to that shortcut is authorized using that credential. The supported delegated credential is an HMAC key and secret for a Service account or User account.

The account must have permission to access the data within the GCS bucket. If the bucket-specific endpoint was used in the connection for the shortcut, the account must have the following permissions:

- `storage.objects.get`
- `storage.objects.list`

If the global endpoint was used in the connection for the shortcut, the account must also have the following permission:

- `storage.buckets.list`

Dataverse shortcuts

Dataverse direct integration with Microsoft Fabric enables organizations to extend their Dynamics 365 enterprise applications and business processes into Fabric. This integration is accomplished through shortcuts, which can be created in two ways: through the PowerApps maker portal, or through Fabric directly.

 **Note**

Dataverse shortcuts are read-only. They don't support write operations regardless of the user's permissions.

Creating shortcuts through PowerApps maker portal

Authorized PowerApps users can access the PowerApps maker portal and use the [Link to Microsoft Fabric](#) feature. From this single action, a lakehouse is created in Fabric and shortcuts are automatically generated for each table in the Dataverse environment.

For more information, see [Dataverse direct integration with Microsoft Fabric](#).

Creating shortcuts through Fabric

Fabric users can also create shortcuts to Dataverse. When users create shortcuts, they can select **Dataverse**, supply their environment URL, and browse the available tables. This experience allows users to choose which tables to bring into Fabric rather than bringing in all tables.

 **Note**

Dataverse tables must first be available in the Dataverse Managed Lake before they're visible in the Fabric create shortcuts UX. If your tables aren't visible from Fabric, use the [Link to Microsoft Fabric](#) feature from the PowerApps maker portal.

Authorization

Dataverse shortcuts use a delegated authorization model. In this model, the shortcut creator specifies a credential for the Dataverse shortcut, and all access to that shortcut is authorized using that credential. The supported delegated credential type is organizational account (OAuth2). The organizational account must have the system administrator permission to access data in Dataverse Managed Lake.

 **Note**

Service principals added to the fabric workspace must have the admin role to authorize the Dataverse shortcut.

Caching

Shortcut caching can reduce egress costs associated with cross-cloud data access. As files are read through an external shortcut, the files are stored in a cache for the Fabric workspace. Subsequent read requests are served from cache rather than the remote storage provider. The retention period for cached files can be set from 1-28 days. Each time the file is accessed, the retention period is reset. If the file in remote storage provider is more recent than the file in the cache, the request is served from remote storage provider and the updated file will then be stored in cache. If a file hasn't been accessed for more than the selected retention period, it's purged from the cache. Individual files greater than 1 GB in size aren't cached.

 **Note**

Shortcut caching is currently supported for GCS, S3, S3 compatible, and on-premises data gateway shortcuts.

To enable caching for shortcuts, open the **Workspace settings** panel. Choose the **OneLake** tab. Toggle the cache setting to **On** and select the **Retention Period**.

The cache can also be cleared at any time. From the same settings page, select the **Reset cache** button. This action removes all files from the shortcut cache in this workspace.

Workspace settings

Search

OneLake Settings

Configure and manage settings for OneLake in this workspace
[Learn more about OneLake](#)

- General
- License info
- Azure connections
- System storage
- Git integration
- OneLake**
- Workspace identity
- Network security
- Power BI
- Delegated Settings
- Extension API playground
- Data Engineering/Science
- Data Factory

OneLake File Explorer

The OneLake file explorer application seamlessly integrates OneLake with Windows File Explorer
[Download OneLake app](#)

- Shortcut Settings

Enable cache for shortcuts

Data accessed through shortcuts in this workspace will be cached in OneLake. This data will remain in cache up to the defined retention period.
[Learn more about shortcuts cache settings](#)

Retention Period

Select the maximum time that data from a shortcut can be retained in cache within the workspace. You can set retention period from 1 - 28 days.

28 days

Reset cache

This action will remove all cached data for shortcuts in this workspace from OneLake.

Reset cache

How shortcuts utilize cloud connections

ADLS and S3 shortcut authorization is delegated by using cloud connections. When you create a new ADLS or S3 shortcut, you either create a new connection or select an existing connection for the data source. Setting a connection for a shortcut is a bind operation. Only users with permission on the connection can perform the bind operation. If you don't have permissions on the connection, you can't create new shortcuts using that connection.

Shortcut security

Shortcuts require certain permissions to manage and use. [OneLake shortcut security](#) looks at the permissions required to create shortcuts and access data using them.

How do shortcuts handle deletions?

Shortcuts don't perform cascading deletes. When you delete a shortcut, you only delete the shortcut object. The data in the shortcut target remains unchanged. However, if you delete a file or folder within a shortcut, and you have permissions in the shortcut target to perform the delete operation, the files or folders are deleted in the target.

For example, consider a lakehouse with the following path in it:

`MyLakehouse\Files\MyShortcut\Foo\Bar`. **MyShortcut** is a shortcut that points to an ADLS Gen2 account that contains the *Foo\Bar* directories.

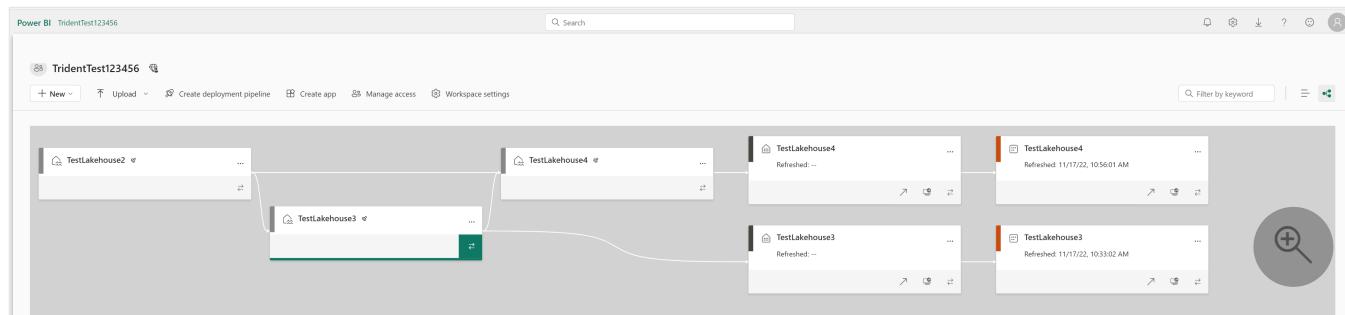
You can perform a delete operation on the following path: `MyLakehouse\Files\MyShortcut`. In this case, the **MyShortcut** shortcut is deleted from the lakehouse but the files and directories in the ADLS Gen2 account *Foo\Bar* remain unaffected.

You can also perform a delete operation on the following path:

`MyLakehouse\Files\MyShortcut\Foo\Bar`. In this case, if you write permissions in the ADLS Gen2 account, the **Bar** directory is deleted from the ADLS Gen2 account.

Workspace lineage view

When creating shortcuts between multiple Fabric items within a workspace, you can visualize the shortcut relationships through the workspace lineage view. Select the **Lineage view** button () in the upper right corner of the Workspace explorer.



! Note

The lineage view is scoped to a single workspace. Shortcuts to locations outside the selected workspace don't appear.

Limitations and considerations

- The maximum number of shortcuts per Fabric item is 100,000. In this context, the term item refers to: apps, lakehouses, warehouses, reports, and more.

- The maximum number of shortcuts in a single OneLake path is 10.
- The maximum number of direct shortcuts to shortcut links is 5.
- ADLS and S3 shortcut target paths can't contain any reserved characters from [RFC 3986 section 2.2](#). For allowed characters, see [RFC 3968 section 2.3](#).
- OneLake shortcut names, parent paths, and target paths can't contain "%" or "+" characters.
- Shortcuts don't support non-Latin characters.
- Copy Blob API isn't supported for ADLS or S3 shortcuts.
- Copy function doesn't work on shortcuts that directly point to ADLS containers. It's recommended to create ADLS shortcuts to a directory that is at least one level below a container.
- More shortcuts can't be created inside ADLS or S3 shortcuts.
- Lineage for shortcuts to Data Warehouses and Semantic Models isn't currently available.
- A Fabric shortcut syncs with the source almost instantly, but propagation time might vary due to data source performance, cached views, or network connectivity issues.
- It might take up to a minute for the Table API to recognize new shortcuts.
- OneLake shortcuts don't support connections to ADLS Gen2 storage accounts that use managed private endpoints. For more information, see [managed private endpoints for Fabric](#).

Related content

- [Create a OneLake shortcut](#)
- [Use OneLake shortcuts REST APIs](#)

Shortcuts file transformations

Shortcut transformations convert raw files (CSV, Parquet, and JSON) into **Delta tables** that stay *always in sync* with the source data. The transformation is executed by **Fabric Spark compute**, which copies the data referenced by a OneLake shortcut into a managed Delta table so you don't have to build and orchestrate traditional extract, transform, load (ETL) pipelines yourself. With automatic schema handling, deep flattening capabilities, and support for multiple compression formats, shortcut transformations eliminate the complexity of building and maintaining ETL pipelines.

! Note

Shortcut transformations are currently in **public preview** and are subject to change.

Why use shortcut transformations?

- **No manual pipelines** – Fabric automatically copies and converts the source files to Delta format; you don't have to orchestrate incremental loads.
- **Frequent refresh** – Fabric checks the shortcut every **2 minutes** and synchronizes any changes almost immediately.
- **Open & analytics-ready** – Output is a Delta Lake table that any Apache Spark-compatible engine can query.
- **Unified governance** – The shortcut inherits OneLake lineage, permissions, and Microsoft Purview policies.
- **Spark based** – Transforms built for scale.

Prerequisites

 Expand table

| Requirement | Details |
|----------------------|---|
| Microsoft Fabric SKU | Capacity or Trial that supports Lakehouse workloads. |
| Source data | A folder that contains homogeneous CSV, Parquet, or JSON files. |
| Workspace role | Contributor or higher. |

Supported sources, formats and destinations

All data sources supported in OneLake are supported.

 Expand table

| Source file format | Destination | Supported Extensions | Supported Compression types | Notes |
|---------------------|---|--|--|--|
| CSV (UTF-8, UTF-16) | Delta Lake table in the / Tables folder | .csv,.txt(delimiter),.tsv(tab-separated),.psv(pipe-separated), | .csv.gz,.csv.bz2 | .csv.zip,.csv.snappy aren't supported as of date |
| Parquet | Delta Lake table in the / Tables folder | .parquet | .parquet.snappy,.parquet.gzip,.parquet.lz4,.parquet.brotli,.parquet.zstd | |

| Source | Destination | Supported Extensions | Supported Compression types | Notes |
|-------------|---|----------------------|--|--|
| file format | | | | |
| JSON | Delta Lake table in the Lakehouse / Tables folder | .json,jsonl,.ndjson | .json.gz,.json.bz2,.jsonl.gz,.ndjson.gz,.jsonl.bz2,.ndjson.bz2 | .json.zip,.json.snappy aren't supported as of date |

- Excel file support is part of roadmap
- AI Transformations available to support unstructured file formats (.txt, .doc, .docx) with Text Analytics use case live with more enhancements upcoming

Set up a shortcut transformation

1. In your lakehouse, select **New Table Shortcut** in **Tables** section which is **Shortcut transformation (preview)** and choose your source (for example, Azure Data Lake, Azure Blob Storage, Dataverse, Amazon S3, GCP, SharePoint, OneDrive etc.).

| city | condition | date | humidity_percent | precipitation_inches | state | temperature_F | filepath |
|---------------|---------------|------------|------------------|----------------------|----------|---------------|------------------|
| Chicago | Windy | 2025-10-07 | 92 | 1.83 | Illinois | 51.8 | usa_weather_data |
| Orlando | Thunderstorms | 2025-10-08 | 100 | 0.74 | Florida | 70.5 | usa_weather_data |
| | Sunny | 2025-10-07 | 29 | 1.03 | Texas | 72.8 | usa_weather_data |
| | Thunderstorms | 2025-10-07 | 21 | 1.39 | New York | 55.9 | usa_weather_data |
| | Sunny | 2025-10-13 | 95 | 0.58 | Florida | 37 | usa_weather_data |
| | Cloudy | 2025-10-11 | 24 | 1.13 | New York | 93.9 | usa_weather_data |
| | Cloudy | 2025-10-13 | 44 | 0.75 | New York | 71.6 | usa_weather_data |
| | Sunny | 2025-10-12 | 21 | 0.23 | New York | 42.6 | usa_weather_data |
| | Thunderstorms | 2025-10-08 | 44 | 0.76 | Illinois | 78.5 | usa_weather_data |
| | Thunderstorms | 2025-10-06 | 54 | 1.98 | Illinois | 64.3 | usa_weather_data |
| | Sunny | 2025-10-12 | 30 | 0.07 | New York | 81.9 | usa_weather_data |
| New York City | Thunderstorms | 2025-10-07 | 89 | 1.2 | Illinois | 68.8 | usa_weather_data |
| Chicago | Thunderstorms | 2025-10-14 | 80 | 0.10 | Texas | 83.7 | usa_weather_data |

2. **Choose file, Configure transformation & create shortcut** – Browse to an existing OneLake shortcut that points to the folder with your CSV files, configure parameters, and initiate creation.

- *Delimiter* in CSV files – Select the character used to separate columns (comma, semicolon, pipe, tab, ampersand, space).
- *First row as headers* – Indicate whether the first row contains column names.
- *Table Shortcut name* – Provide a friendly name; Fabric creates it under **/Tables**.

3. Track refreshes and view logs for transparency in **Manage Shortcut monitoring hub**.

Fabric Spark compute copies the data into a Delta table and shows progress in the **Manage shortcut** pane. Shortcut transformations are available in Lakehouse items. They create Delta Lake tables in the **Lakehouse / Tables** folder.

How synchronization works

After the initial load, Fabric Spark compute:

- Polls the shortcut target **every 2 minutes**.
- Detects **new or modified files** and appends or overwrites rows accordingly.
- Detects **deleted files** and removes corresponding rows.

Monitor and troubleshoot

Shortcut transformations include monitoring and error handling to help you track ingestion status and diagnose issues.

1. Open the lakehouse and right-click the shortcut that feeds your transformation.

2. Select **Manage shortcut**.

3. In the details pane, you can view:

- **Status** – Last scan result and current sync state.
- **Refresh history** – Chronological list of sync operations with row counts and any error details.

Manage shortcut

Manage the status of your transforms and edit properties of your shortcut.

Properties

| | |
|--------------------|--------------------------------------|
| Name | Data type |
| WeatherUSAJson | Delta |
| Target type | Target workspace |
| OneLake | cb54ebc5-0f91-4e50-bec9-91e1c722b44a |
| Target location | Target subpath |
| perftesting | Files/Json - Weather USA |
| Shortcut location | Last modified |
| Tables/dbo | Oct 13, 6:11:49 PM |
| Transform | Transform ID |
| Target file format | 5d5b38f3-fef4-4fa0-8a99-bea3f9bfd336 |
| JSON | Source file format |
| | Delta table |

Monitor

Refresh

Search

Filter

| Job Id | Status | Start time | Duration | Last updated |
|-------------------------------|-----------|--------------------|----------|--------------------|
| b37c799f-8727-460c-a67f-3b... | Succeeded | 10:40 PM, 10/13/25 | 0:00 | 10:40 PM, 10/13/25 |
| 113a7974-36ba-4ba2-b6a4-... | Succeeded | 6:11 PM, 10/13/25 | 0:00 | 6:12 PM, 10/13/25 |

4. View more details in logs to troubleshoot

| Job Id | Status | Start time |
|-------------------------------|--------------------------|----------------------|
| b37c799f-8727-460c-a67f-3b... | ✓ Succeeded | 10:40 PM, 10/13/2025 |
| 113a7974-36ba-4ba2-b6a4-... | ✓ Succeeded | 6:11 PM, 10/13/2025 |

Details

```
{  
  "OneLakeTransformId":  
    "5d5b38f3-fef4-4fa0-8a99-  
    bea3f9bfd336",  
  "OneLakeTransformJobId":  
    "1dc78a1-2dfe-4379-8aab-  
    0a84ad0a9d07",  
  "Status": "SUCCESS",  
  "JobDetails": {  
    "StartTime": "2025-10-  
    13T12:42:22.630668+00:00",  
    "ErrorDetails": [],  
    "EndTime": "2025-10-  
    13T12:42:37.367697+00:00"  
  },  
  "FileDetails": {  
    "Errors": [],  
    "Warnings": [],  
    "Success": [  
      {  
        "FileName":  
          "usa_weather_data.json",  
        "RequestType":  
          "update",  
        "RowsAffected": 1000  
      }  
    ]  
  }  
}
```

! Note

Pause or Delete the transformation from this tab is an upcoming feature part of roadmap

Limitations

Current limitations of shortcut transformations:

- Only CSV, Parquet, JSON file formats are supported.
- Files must share an identical schema; schema drift isn't yet supported.
- Transformations are *read-optimized*; MERGE INTO or DELETE statements directly on the table are blocked.
- Available only in Lakehouse items (not Warehouses or KQL databases).
- **Unsupported datatypes for CSV:** Mixed data type columns, Timestamp_Nanos, Complex logical types - MAP/LIST/STRUCT, Raw binary
- **Unsupported datatype for Parquet:** Timestamp_nanos, Decimal with INT32/INT64, INT96, Unassigned integer types - UINT_8/UINT_16/UINT_64, Complex logical types - MAP/LIST/STRUCT)
- **Unsupported datatypes for JSON:** Mixed data types in an array, Raw binary blobs inside JSON, Timestamp_Nanos
- **Flattening of Array data type in JSON:** Array data type shall be retained in delta table and data accessible with Spark SQL & Pyspark where for further transformations Fabric Materialized Lake Views could be used for silver layer
- **Source format:** Only CSV, JSON, and Parquet files are supported as of date.
- **Flattening depth in JSON:** Nested structures are flattened up to five levels deep. Deeper nesting requires preprocessing.
- **Write operations:** Transformations are *read-optimized*; direct MERGE INTO or DELETE statements on the transformation target table aren't supported.
- **Workspace availability:** Available only in Lakehouse items (not Data Warehouses or KQL databases).
- **File schema consistency:** Files must share an identical schema.

 **Note**

Adding support for some of the above and reducing limitations is part of our roadmap. Track our release communications for further updates.

Clean up

To stop synchronization, delete the shortcut transformation from the lakehouse UI.

Deleting the transformation doesn't remove the underlying files.

 **Note:** The author created this article with assistance from AI. [Learn more](#)

Last updated on 12/01/2025

AI-powered transforms in OneLake shortcut transformations

07/17/2025

Important

AI transforms for OneLake shortcuts are currently **Public Preview**. Features and behavior may change before general availability.

Modern data lakes are brimming with raw, unstructured text, product reviews, support emails, IoT device logs, and more. Turning that text into actionable insights typically requires custom code, orchestration pipelines, and constant maintenance. **OneLake Shortcut Transformations** remove that overhead: you point to your files once, choose an AI transform, and Fabric does the rest.

Why use AI-powered transforms?

 Expand table

| Benefit | What it means for you |
|--------------------------------|--|
| Accelerate time-to-insight | Go from raw text to a queryable Delta table in minutes, no ETL required. |
| Lower maintenance | The transformation engine watches the source folder on a 2-minute schedule, so outputs stay up to date automatically. |
| Enterprise-grade security | PII detection helps you comply with GDPR, HIPAA, and other regulations by redacting sensitive data before it lands in analytics. |
| Consistent, repeatable results | Built-in AI models provide standardized sentiment scores, entity tags, and translations, eliminating manual data-prep drift. |

OneLake Shortcut Transformations in **Microsoft Fabric** include a set of built-in, AI-powered transforms that you can apply directly to `.txt` files referenced through shortcuts, without writing code or building pipelines. The engine automatically keeps the output **Delta table** in sync, so your data is query-ready for **Power BI**, notebooks, pipelines, and other Fabric experiences.

Supported AI transforms

| Transform | Purpose |
|--------------------|---|
| Summarization | Generates concise summaries from long-form text. |
| Translation | Translates text between supported languages. |
| Sentiment analysis | Labels text sentiment as <i>positive</i> , <i>negative</i> , or <i>neutral</i> . |
| PII detection | Finds and redacts personally identifiable information (names, phone numbers, emails). |
| Name recognition | Extracts named entities such as people, organizations, or locations. |

(!) Note

AI transforms currently support `.txt` files only as input.

Example — PII detection in customer feedback

Customer feedback stored in a data lake may contain sensitive details (names, emails, phone numbers). Apply the **PII detection** transform to automatically scan and redact this content and produce a privacy-compliant Delta table for analysis.

How it works

1. Create a shortcut

Reference a folder of `.txt` files in Azure Data Lake, Amazon S3, or another OneLake shortcuts source.

2. Select an AI transform

Pick one of the supported transforms during shortcut creation.

3. Automatic sync

The engine checks the source folder every **2 minutes**. New, modified, or deleted files are reflected in the Delta table.

4. Query-ready output

Use the resulting table immediately in reports, notebooks, or downstream pipelines.

Regional availability

AI-powered transforms are currently available in these regions: [Azure AI Language regional support](#)

Create a OneLake shortcut

Article • 03/31/2025

In this article, you learn how to create a OneLake shortcut inside a Fabric lakehouse. You can use a lakehouse, a data warehouse, or a Kusto Query Language (KQL) database as the source for your shortcut.

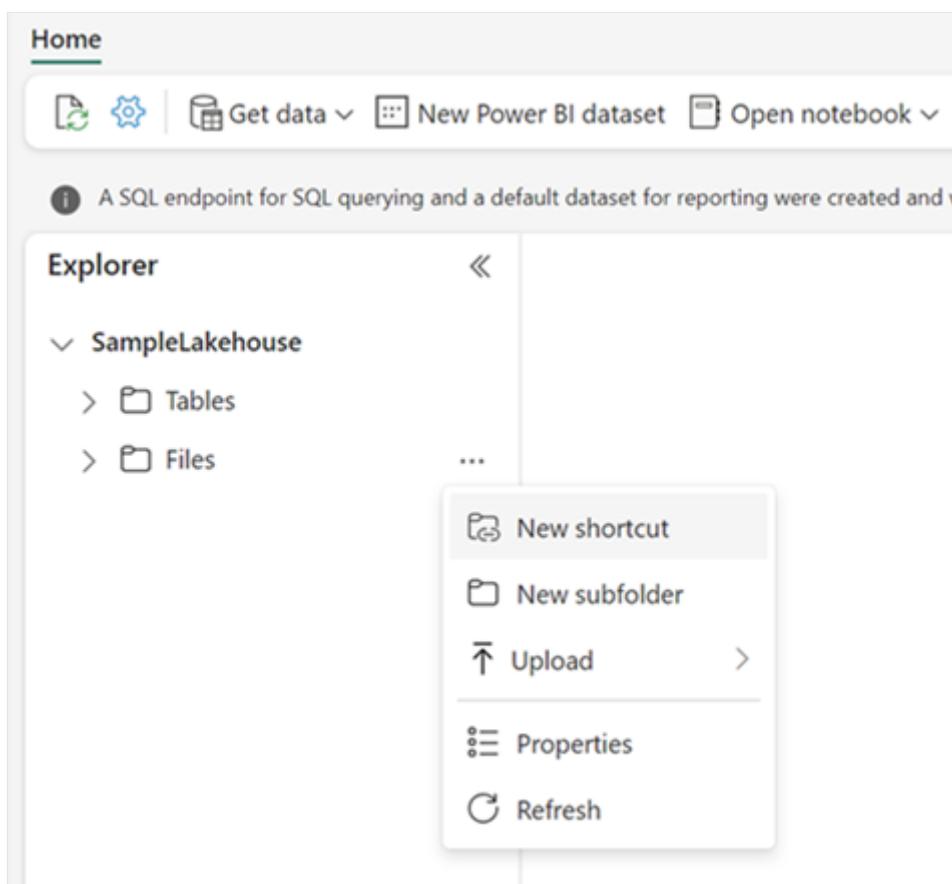
For an overview of shortcuts, see [OneLake shortcuts](#). To create shortcuts programmatically, see [OneLake shortcuts REST APIs](#).

Prerequisite

A lakehouse in OneLake. If you don't have a lakehouse, create one by following these steps: [Create a lakehouse with OneLake](#).

Create a shortcut

1. Open a lakehouse.
2. Right-click on a directory within the **Explorer** pane of the lakehouse.
3. Select **New shortcut**.



Select a source

- Under Internal sources, select Microsoft OneLake.

New shortcut

Use shortcuts to quickly pull data from internal and external locations into your lakehouses, warehouses, or datasets. Shortcuts can be updated or removed from your item, but these changes will not affect the original data and its source.

Internal sources

Microsoft OneLake 

Fabric

External sources

Amazon S3 

AWS

Amazon S3 Compatible 

Generic Connector

Azure Data Lake Storage Gen2 

Azure

Dataverse 

Power Platform

Google Cloud Storage 

GCP

- Select the data source that you want to connect to, and then select **Next**.

Select a data source type

Find and connect to the data you want to use with your shortcut.

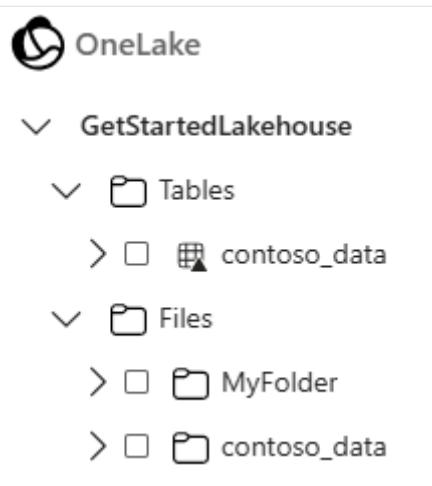
All Endorsed in your org My data Favorites

Filter by keyword Filter All domains

| » | Name | Type | Capacity region | Owner |
|---|---------------------|-----------|-----------------|--------------|
|  | GetStartedLakehouse | Lakehouse | West Central US | Avery Howard |
|  | SampleWarehouse | Warehouse | West Central US | Avery Howard |

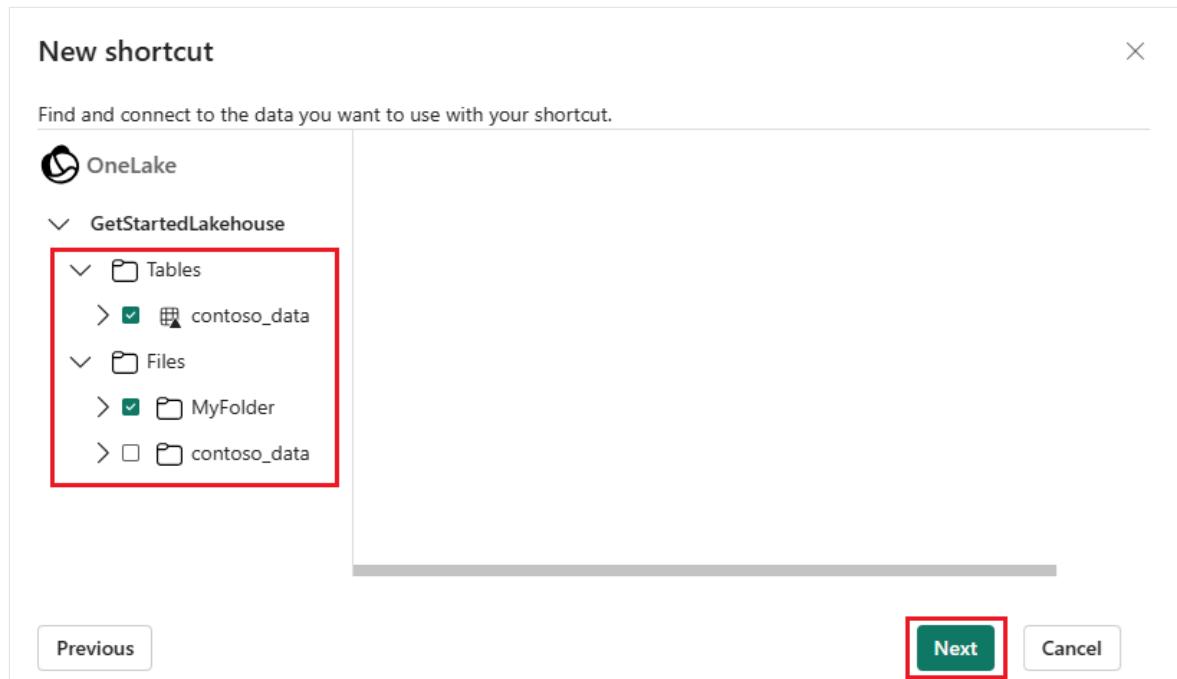
Previous Next Cancel 

- Expand **Files** or **Tables** to view the available subfolders. Subfolders in the tables directory that contain valid Delta or Iceberg tables are indicated with a table icon. Files or unidentified folders in the tables section are indicated with a folder icon.

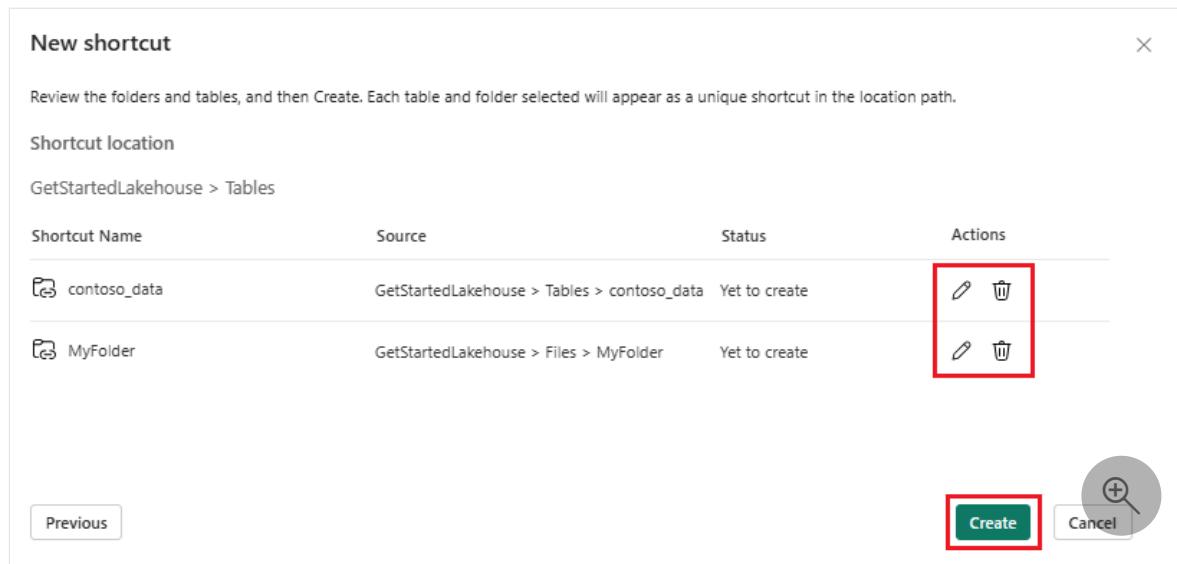


4. Select one or more subfolders to connect to, then select **Next**.

You can select up to 50 subfolders when creating OneLake shortcuts.



5. Review your selected shortcut locations. Use the edit action to change the default shortcut name. Use the delete action to remove any undesired selections. Select **Create** to generate shortcuts.



The lakehouse automatically refreshes. The shortcut appears under the selected directory in the **Explorer** pane. You can differentiate a regular file or table from the shortcut from its properties. The properties have a **Shortcut Type** parameter that indicates the item is a shortcut.

Home

Get data | New semantic model | Open

A SQL analytics endpoint for SQL querying and a default Power BI sema

Explorer

- SampleLH
 - Tables
 - dim_customer
 - dim_product
 - Files
 - File1.xlsx

Edit a shortcut

Editing shortcuts requires write permission on the item being edited. The admin, member, and contributor roles grant write permissions.

1. To edit a shortcut, right-click on the shortcut and select **Manage shortcut**.

2. In the **Manage shortcut** view, you can edit the following fields:

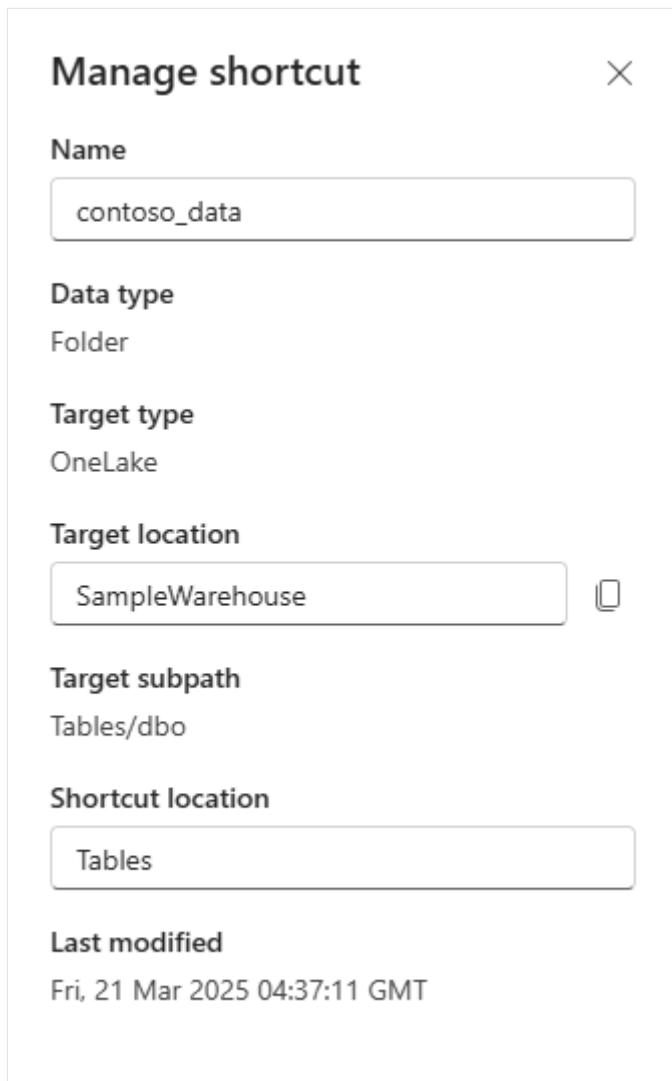
- **Name**
- **Target connection**

Not all shortcut types use the target connection feature.

- **Target location** and **Target subpath**

Both of these fields are editable by selecting the **Target location**.

- **Shortcut location**



You can also edit shortcuts by using the [OneLake shortcuts REST APIs](#).

Remove a shortcut

To delete a shortcut, select the ... icon next to the shortcut file or table and select **Delete**. To delete shortcuts programmatically, see [OneLake shortcuts REST APIs](#).

Related content

- [Create an Azure Data Lake Storage Gen2 shortcut](#)
 - [Create an Amazon S3 shortcut](#)
 - [Use OneLake shortcuts REST APIs](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Ask the community](#)

Create an Azure Data Lake Storage Gen2 shortcut

Article • 07/25/2024

In this article, you learn how to create an Azure Data Lake Storage (ADLS) Gen2 shortcut inside a Microsoft Fabric lakehouse.

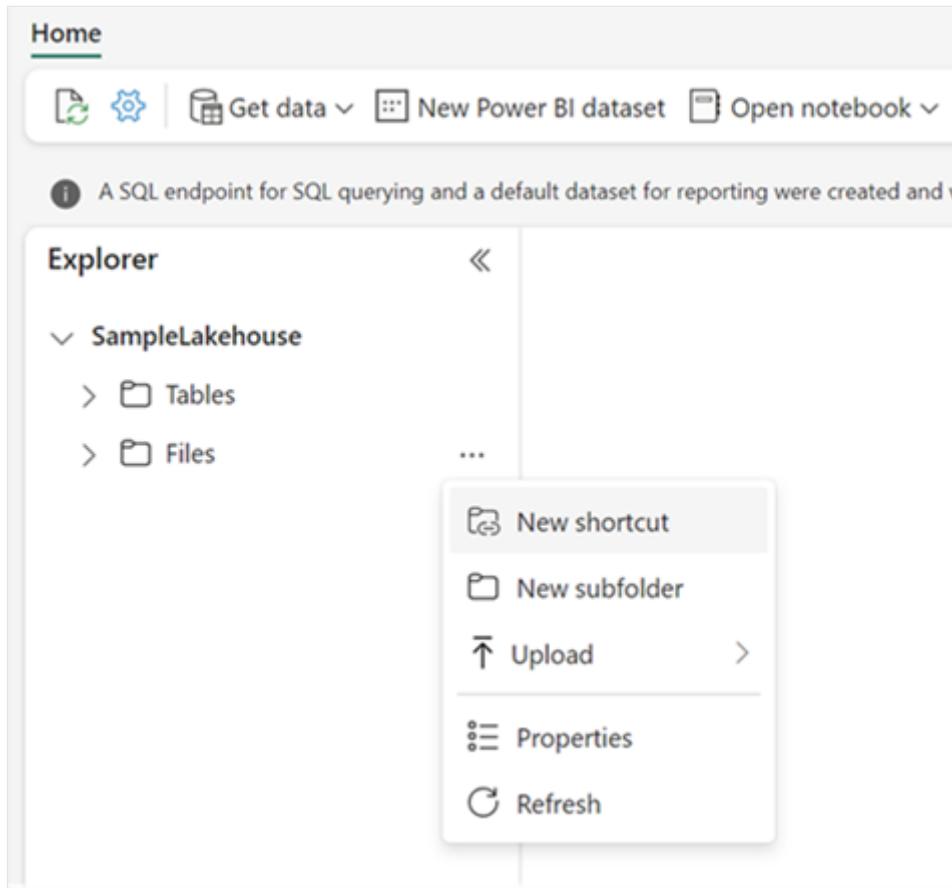
For an overview of shortcuts, see [OneLake shortcuts](#). To create shortcuts programmatically, see [OneLake shortcuts REST APIs](#).

Prerequisites

- If you don't have a lakehouse, create one by following these steps: [Create a lakehouse with OneLake](#).
- You must have Hierarchical Namespaces enabled on your ADLS Gen 2 storage account.

Create a shortcut

1. Open a lakehouse.
2. Right-click on a directory within the **Lake view** of the lakehouse.
3. Select **New shortcut**.



Select a source

1. Under External sources, select Azure Data Lake Storage Gen2.

New shortcut

Use shortcuts to quickly pull data from internal and external locations into your lakehouses, warehouses, or datasets. Shortcuts can be updated or removed from your item, but these changes will not affect the original data and its source.

Internal sources

Microsoft OneLake

Fabric

External sources

Azure Data Lake Storage Gen2

Azure

Amazon S3

File

2. Enter the **Connection settings** according to the following table:

New shortcut

123abc12-3abc-123a-bc12-3abc123abc12 is located in the region **Central US**. Any data sourced through this shortcut will be processed in the same region.

| | |
|--|--|
|  Azure Data Lake Storage Gen2 Azure Learn more | Connection settings URL * ⓘ <input type="text" value="https://123abc.dfs.core.windows.net/"/> |
| Connection credentials | |
| Connection <input type="button" value="Create new connection"/>  | |
| Connection name <input type="text" value="TestingADLSGen2"/> | |
| Authentication kind <input type="button" value="Shared Access Signature (SAS)"/> | |
| SAS token <input type="text" value="*****"/> | |

+
Next
Cancel

 [Expand table](#)

| Field | Description | Value |
|---------------------|---|---|
| URL | The connection string for your delta container. | <code>https://StorageAccountName.dfs.core.windows.net</code> |
| Connection | Previously defined connections for the specified storage location appear in the drop-down. If no connections exist, create a new connection. | <i>Create new connection.</i> |
| Connection name | The Azure Data Lake Storage Gen2 connection name. | A name for your connection. |
| Authentication kind | The authorization model. The supported models are: Organizational account, Account key, Shared Access Signature (SAS), and Service principal. For | Dependent on the authorization model. Once you select an authentication kind, fill in the required credentials. |

| Field | Description | Value |
|-------|--|-------|
| | more information, see ADLS shortcuts . | |

3. Select Next.

4. Browse to the target location for the shortcut.

| Name | Field type | Last modified |
|---------------------|------------|-------------------------------|
| Customers | Folder | Sat, 09 Mar 2024 20:10:21 GMT |
| EmployeeTerritories | Folder | Sat, 09 Mar 2024 20:10:22 GMT |
| Employees | Folder | Sat, 09 Mar 2024 20:10:21 GMT |
| FACT_Order | Folder | Wed, 25 Jan 2023 20:57:16 GM* |
| OrderDetails | Folder | Sat, 09 Mar 2024 20:10:21 GMT |
| Orders | Folder | Sat, 09 Mar 2024 20:10:21 GMT |
| Products | Folder | Sat, 09 Mar 2024 20:10:21 GMT |
| Regions | Folder | Sat, 09 Mar 2024 20:10:22 GMT |
| Shippers | Folder | Sat, 09 Mar 2024 20:10:22 GMT |

If you just used the storage account in the connection URL, all of your available containers appear in the left navigation view. If you specified a container in connection URL, only the specified container and its contents appear in the navigation view.

Navigate the storage account by selecting a folder or clicking on the expansion arrow next to a folder.

In this view, you can select one or more shortcut target locations. Choose target locations by clicking the checkbox next a folder in the left navigation view.

5. Select Next

New shortcut

(i) Browse_Example is located in the region **East US 2 EUAP**. Any data sourced through this shortcut will be processed in the same region.

Review the folders and tables, and then Create. Each table and folder selected will appear as a unique shortcut in the location path.

Shortcut Location
Browse_Example > Tables

| Shortcut Name | Source | Status | Actions |
|---------------|---------------------------------------|---------------|---------|
| dim_customer | northwindsales > delta > dim_customer | Yet to create | |
| dim_product | northwindsales > delta > dim_product | Yet to create | |

Previous **Create** Cancel

The review page allows you to verify all of your selections. Here you can see each shortcut that will be created. In the action column, you can click the pencil icon to edit the shortcut name. You can click the trash can icon to delete shortcut.

6. Select **Create**.

7. The lakehouse automatically refreshes. The shortcut appears in the left **Explorer** pane.

Home

Get data New semantic model Open

(i) A SQL analytics endpoint for SQL querying and a default Power BI semantic model has been created.

Explorer

- SampleLH
 - Tables
 - > dim_customer
 - > dim_product
 - Files

Related content

- [Create a OneLake shortcut](#)
 - [Create an Amazon S3 shortcut](#)
 - [Use OneLake shortcuts REST APIs](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Ask the community](#)

Create an Azure Blob Storage shortcut (preview)

Article • 05/19/2025

In this article, you learn how to create an Azure Blob Storage shortcut inside a Microsoft Fabric lakehouse.

For an overview of shortcuts, see [OneLake shortcuts](#). To create shortcuts programmatically, see [OneLake shortcuts REST APIs](#).

 Note

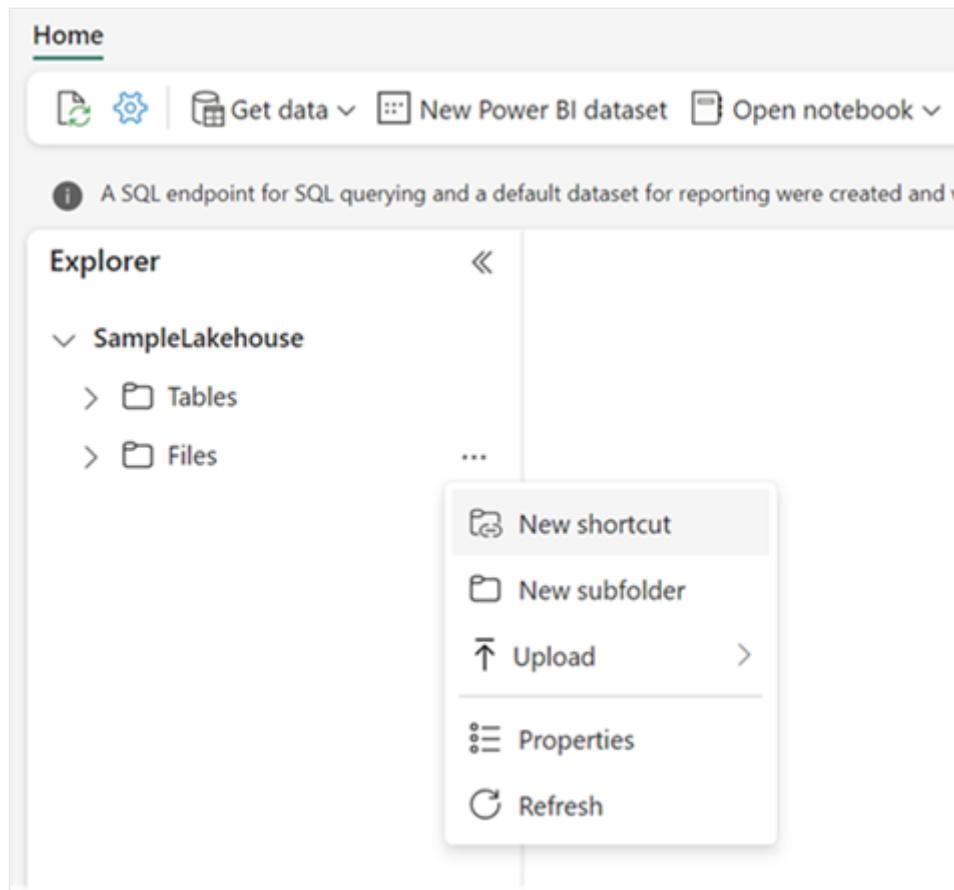
Azure Blob Storage shortcuts are currently in public preview.

Prerequisites

- A lakehouse in Microsoft Fabric. If you don't have a lakehouse, create one by following these steps: [Create a lakehouse with OneLake](#).
- An Azure Storage account with data in a container.

Create a shortcut

1. Open a lakehouse in Fabric.
2. Right-click on a directory in the **Explorer** pane of the lakehouse.
3. Select **New shortcut**.



Select a source

When you create a shortcut in a lakehouse, the **New shortcut** window opens to walk you through the configuration details.

1. On the **New shortcut** window, under **External sources**, select **Azure Blob Storage (preview)**.
2. Select **Existing connection** or **Create new connection**, depending on whether this Storage account is already connected in your OneLake.
 - For an **Existing connection**, select the connection from the drop-down menu.
 - To **Create new connection**, provide the following connection settings.

New shortcut X

ContosoLakehouse is located in the region East US 2 EUAP. Any data sourced through this shortcut will be processed in the same region.

Azure Blob Storage (Preview) Existing connection Create new connection

Connection settings

Account name or URL *
contosostorage

Connection credentials

Connection

Connection name

Authentication kind

SAS token

Previous Next Cancel

Expand table

| Field | Description |
|---------------------|---|
| Account name or URL | The name of your blob storage account. |
| Connection | The default value, Create new connection . |
| Connection name | A name for your Azure Blob Storage connection. The service generates a suggested connection name based on the storage account name, but you can overwrite with a preferred name. |
| Authentication kind | Select the authorization model from the drop-down menu that you want to use to connect to the Storage account. The supported models are: account key, organizational account, Shared Access Signature (SAS), service principal, and workspace identity. Once you select a model, fill in the required credentials. For more information, see Azure Blob Storage shortcuts authorization . |

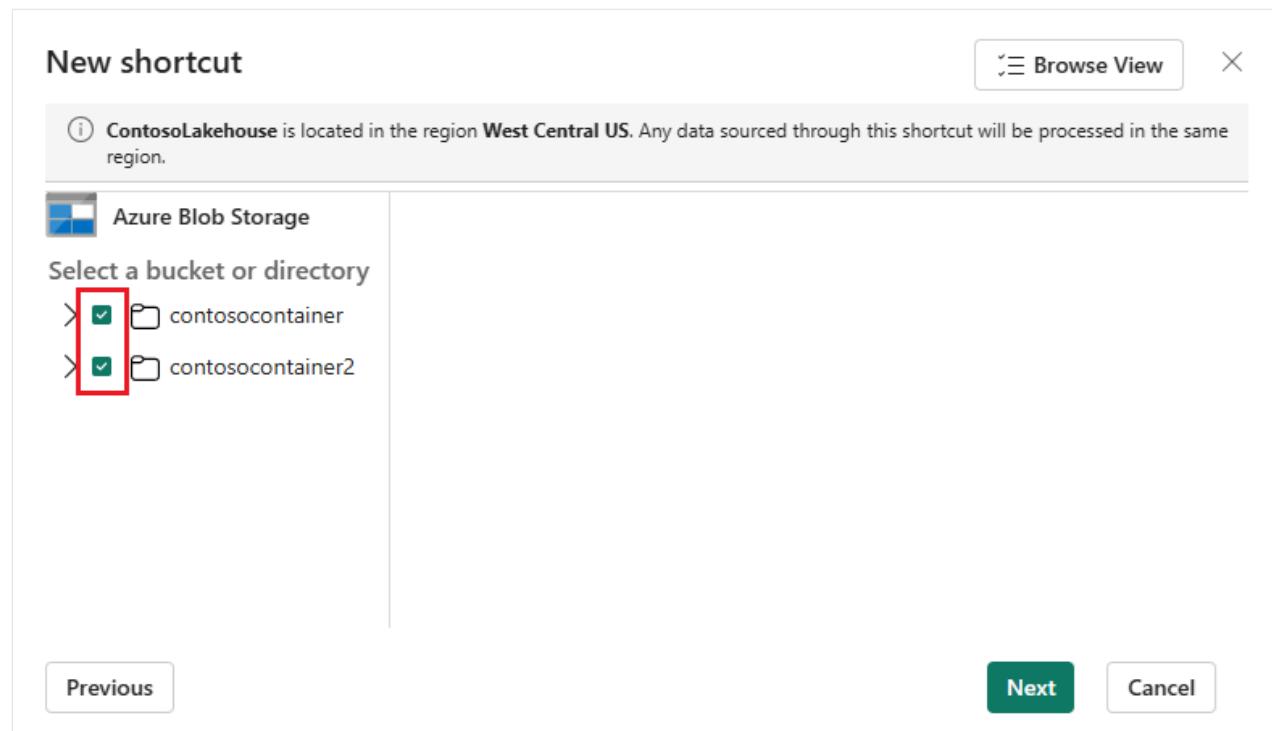
3. Select **Next**.

4. Browse to the target location for the shortcut.

If you provided the storage account name in the connection details, all of your available containers appear in the navigation view. If you specified a container in connection URL, only the specified container and its contents appear in the navigation view.

Navigate the storage account by selecting a folder or expanding a folder to view its child items.

Choose one or more target locations by selecting the checkbox next a folder in the navigation view. Then, select **Next**.



5. On the review page, verify your selections. Here you can see each shortcut to be created. In the **Actions** column, you can select the pencil icon to edit the shortcut name. You can select the trash can icon to delete the shortcut.

6. Select **Create**.

7. The lakehouse automatically refreshes. The shortcut or shortcuts appear in the **Explorer** pane.

The screenshot shows the Power BI 'Explorer' pane. On the left, under 'ContosoLakehouse', the 'Tables' and 'Files' sections are expanded. The 'Files' section contains three items: 'contoso_data', 'contosocontainer', and 'contosocontainer2', all of which are highlighted with a red box. To the right, the 'Files' list shows three items: 'contoso_data' (Date modified: 3/7/2025, Type: Folder), 'contosocontainer' (Date modified: 5/15/2025, Type: Folder), and 'contosocontainer2' (Date modified: 5/15/2025, Type: Folder). A search bar at the top right says 'Showing 3 items'.

Related content

- Create a OneLake shortcut
- Create an Amazon S3 shortcut
- Use OneLake shortcuts REST APIs

Create a OneDrive or SharePoint shortcut (preview)

In this article, you learn how to create a OneDrive or SharePoint shortcut inside a Microsoft Fabric lakehouse.

For an overview of shortcuts, see [OneLake shortcuts](#). To create shortcuts programmatically, see [OneLake shortcuts REST APIs](#).

 **Note**

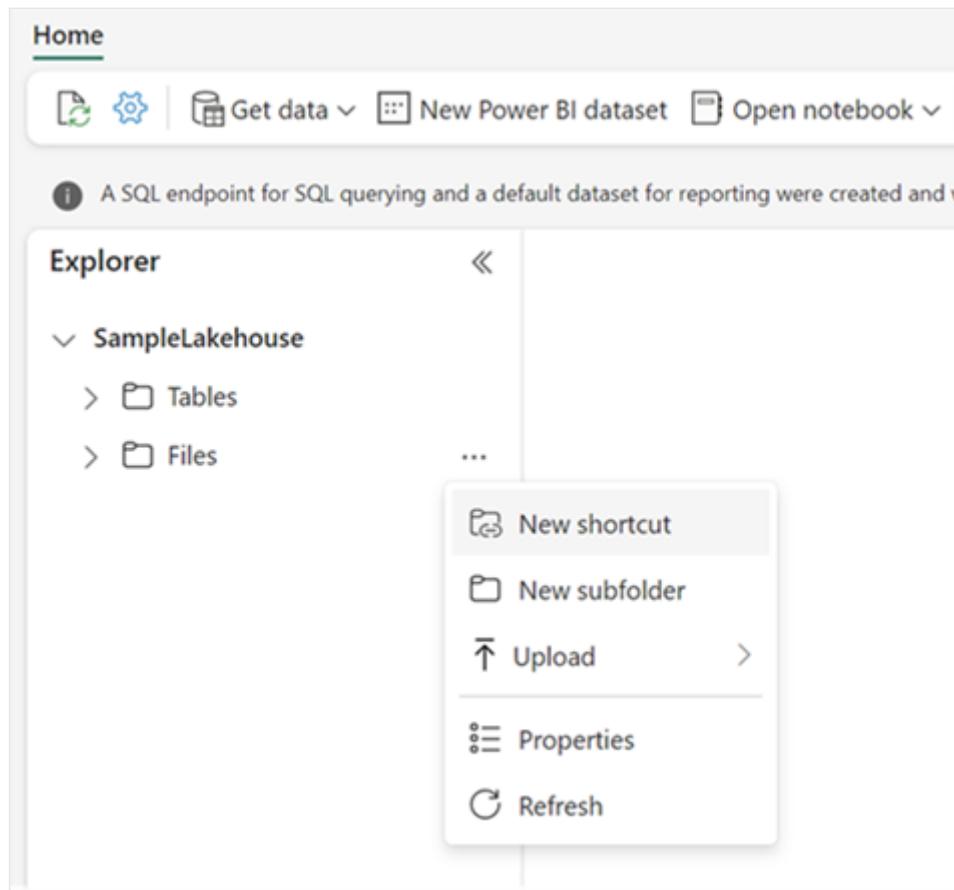
OneDrive and SharePoint shortcuts are currently in public preview.

Prerequisites

- A lakehouse in Microsoft Fabric. If you don't have a lakehouse, create one by following these steps: [Create a lakehouse with OneLake](#).
- Data in a OneDrive or SharePoint folder.

Create a shortcut

1. Open a lakehouse in Fabric.
2. Right-click on a directory in the **Explorer** pane of the lakehouse.
3. Select **New shortcut**.



Select a source

When you create a shortcut in a lakehouse, the **New shortcut** window opens to walk you through the configuration details.

1. On the **New shortcut** window, under **External sources**, select **OneDrive (preview)** or **SharePoint Folder (preview)**.
2. Select **Existing connection** or **New connection**, depending on whether this account is already connected in your OneLake.
 - For an **Existing connection**, select the connection from the drop-down menu.
 - To create a **New connection**, provide the following connection settings:

[] Expand table

| Field | Description |
|----------|---|
| Site URL | The root URL of your SharePoint account. To retrieve your URL, sign in to OneDrive. Select the settings gear icon, then OneDrive settings > More settings . Copy the OneDrive web URL from the more |

| Field | Description |
|---------------------|--|
| | settings page and remove anything after <code>_onmicrosoft_com</code> . For example, https://mytenant-my.sharepoint.com/personal/user01_mytenant_onmicrosoft_com . |
| Connection | The default value, Create new connection . |
| Connection name | A name for your connection. The service generates a suggested connection name based on the storage account name, but you can overwrite with a preferred name. |
| Authentication kind | The supported authentication type for this shortcut is Organizational account . |

3. Select **Next**.

4. Browse to the target location for the shortcut.

Navigate by selecting a folder or expanding a folder to view its child items.

Choose one or more target locations by selecting the checkbox next a folder in the navigation view. Then, select **Next**.

New shortcut Browse View X

Sample_Lakehouse is located in the region **West Central US**. Any data sourced through this shortcut will be processed in the same region.

OneDrive (Preview)

Select a directory

- > Copilot
- > Photos
- > Shared Documents
 - Atlanta
 - Downloads
 - ECP
 - Sweden
 - > Images
 - > Atlanta Video
 - > BayArea
 - > Media

Shared Documents

| Name | Field type | Last modified |
|-----------|------------|-------------------------------|
| Atlanta | Folder | Thu, 04 Feb 2021 19:13:32 GMT |
| Downloads | Folder | Mon, 08 Mar 2021 23:59:01 GMT |
| ECP | Folder | Wed, 01 Nov 2023 18:38:19 GMT |
| Sweden | Folder | Wed, 29 Apr 2020 17:25:48 GMT |

Previous
Next
Cancel

5. On the **Transform** page, select a transformation option if you want to transform the data in your shortcut or select **Skip**.

For more information, see [AI-powered transforms](#).

6. On the review page, verify your selections. Here you can see each shortcut to be created. In the **Actions** column, you can select the pencil icon to edit the shortcut name. You can select the trash can icon to delete the shortcut.

7. Select **Create**.

8. The lakehouse automatically refreshes. The shortcut or shortcuts appear in the **Explorer** pane.

The screenshot shows the OneLake Explorer interface. On the left, the 'Explorer' sidebar lists 'Sample_Lakehouse' with its subfolders 'Tables' and 'Files'. The 'Files' folder is currently selected, highlighted with a grey background. Inside 'Files', there are two sub-folders: 'Atlanta' and 'Sweden'. On the right, the main workspace area displays the file structure under 'My workspace > Sample_Lakehouse > Files'. A search bar at the top right says 'Search files'. The table lists the following items:

| Name | Date modified | Type | Size |
|---------|-----------------|--------|------|
| Atlanta | 11/12/2025, ... | Folder | - |
| Sweden | 11/12/2025, ... | Folder | - |

Related content

- [Create a OneLake shortcut](#)
- [Use OneLake shortcuts REST APIs](#)

Last updated on 11/18/2025

Create an Amazon S3 shortcut

Article • 03/31/2025

In this article, you learn how to create an Amazon S3 shortcut inside a Fabric lakehouse.

For an overview of shortcuts, see [OneLake shortcuts](#). To create shortcuts programmatically, see [OneLake shortcuts REST APIs](#).

You can secure your S3 buckets using customer-managed KMS keys. As long as the IAM user has encrypt/decrypt permissions for the bucket key, OneLake can access the encrypted data in the S3 bucket. For more information, see [Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects](#).

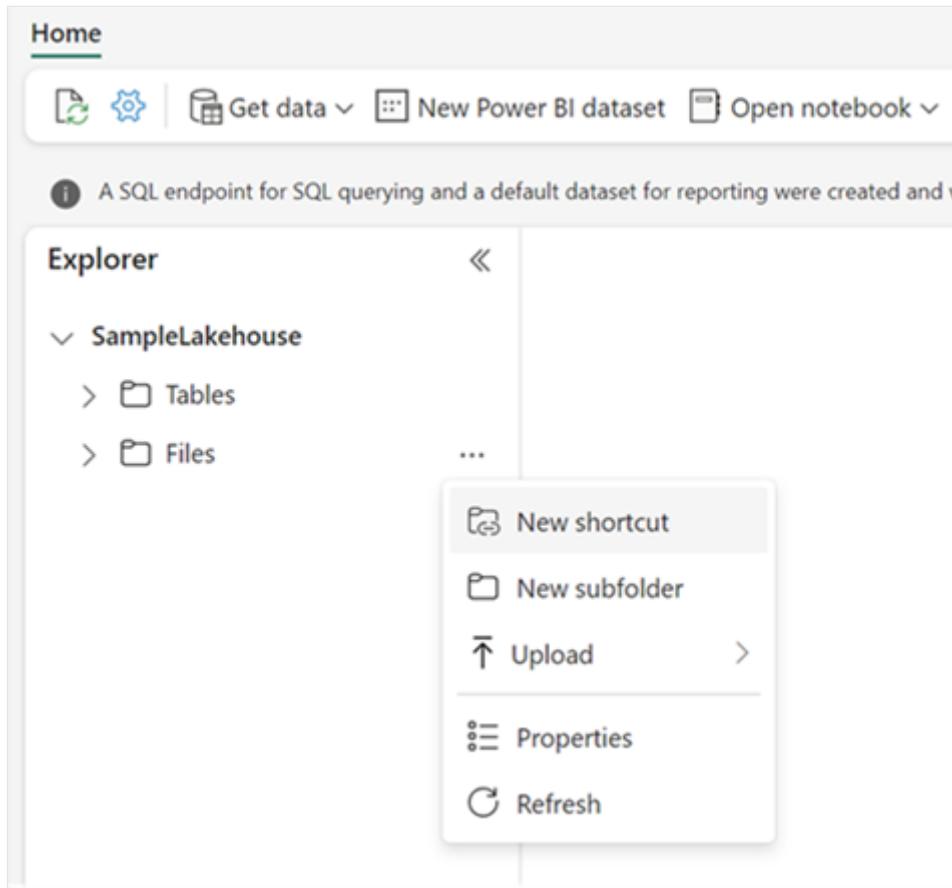
S3 shortcuts can take advantage of file caching to reduce egress costs associated with cross-cloud data access. For more information, see [OneLake shortcuts > Caching](#).

Prerequisites

- If you don't have a lakehouse, create one by following these steps: [Create a lakehouse with OneLake](#).
- Ensure your chosen S3 bucket and IAM user meet the [access and authorization requirements for S3 shortcuts](#).

Create a shortcut

1. Open a lakehouse.
2. Right-click on the **Tables** directory within the lakehouse.
3. Select **New shortcut**.



Select a source

1. Under External sources, select Amazon S3.

The screenshot shows the 'New shortcut' dialog box. It has sections for 'Internal sources' (Microsoft OneLake) and 'External sources' (Azure Data Lake Storage Gen2, Amazon S3). The 'Amazon S3' item is highlighted with a red box. A search icon is located in the bottom right corner of the dialog.

2. Enter the **Connection settings** according to the following table:

New shortcut X

ⓘ abc123abc-abc1-abcd-1234-abcd123abcd1 is located in the region **Central US**. Any data sourced through this shortcut will be processed in the same region.

| | |
|--|---|
| ■ Amazon S3 File Learn more ↗ | Connection settings URL <input type="text" value="https://abc-abc.s3.us-east.amazonaws.com"/> |
| Connection credentials | |
| Connection <input style="width: 150px; height: 20px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px;" type="button" value="Create new connection"/> ↻ Connection name <input type="text" value="https://abc-abc.s3.us-east.amazonaws.com"/> Authentication kind <input style="width: 150px; height: 20px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px;" type="button" value="Access Key"/> Access Key Id <input type="text" value="ABC123ABC123ABC123ABC"/> Secret Access Key <input type="password" value="*****"/> | |
| + 🔍 | |
| <input style="width: 60px; background-color: #0070C0; color: white; border: 1px solid #0070C0; border-radius: 5px; padding: 5px 10px; margin-right: 10px;" type="button" value="Next"/> <input style="width: 60px; border: 1px solid #ccc; border-radius: 5px; padding: 5px 10px;" type="button" value="Cancel"/> | |

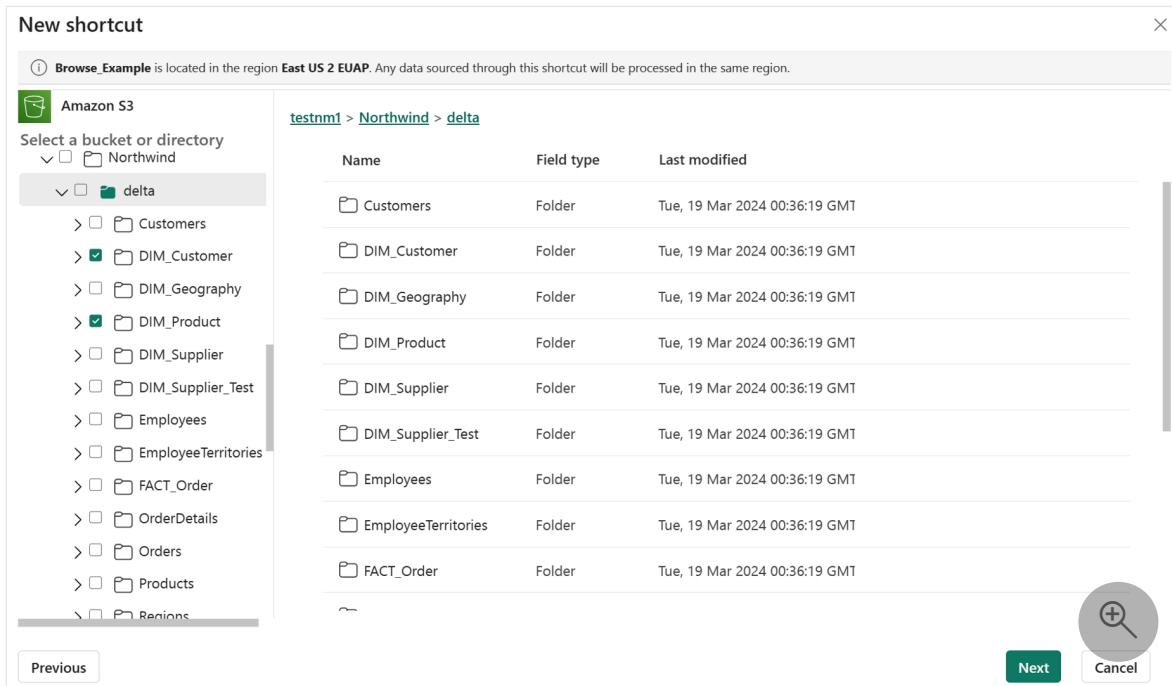
[+] [Expand table](#)

| Field | Description | Value |
|---------------------|---|---|
| URL | The connection string for your Amazon S3 bucket. | <code>https://BucketName.s3.RegionCode.amazonaws.com</code> |
| Connection | Previously defined connections for the specified storage location appear in the drop-down. If no connections exist, create a new connection. | <i>Create new connection</i> |
| Connection name | The Amazon S3 connection name. | A name for your connection. |
| Authentication kind | The <i>Identity and Access Management (IAM)</i> policy. The policy must have read and list permissions. For more information, see IAM users . | Dependent on the bucket policy. |

| Field | Description | Value |
|-------------------|--|------------------|
| Access Key ID | The <i>Identity and Access Management (IAM)</i> user key. For more information, see Manage access keys for IAM users . | Your access key. |
| Secret Access Key | The <i>Identity and Access Management (IAM)</i> secret key. | Your secret key. |

3. Select **Next**.

4. Browse to the target location for the shortcut.



The screenshot shows the 'New shortcut' dialog with the following details:

- Title:** New shortcut
- Note:** `Browse_Example` is located in the region **East US 2 EUAP**. Any data sourced through this shortcut will be processed in the same region.
- Left Panel (Navigation):**
 - Amazon S3
 - Select a bucket or directory
 - Northwind
 - delta** (selected)
 - Customers
 - DIM_Customer
 - DIM_Geography
 - DIM_Product
 - DIM_Supplier
 - DIM_Supplier_Test
 - Employees
 - EmployeeTerritories
 - FACT_Order
 - OrderDetails
 - Orders
 - Products
 - Regions
- Right Panel (List View):**
 - Path:** `testnm1 > Northwind > delta`
 - Table Headers:** Name, Field type, Last modified
 - Table Data:**

| Name | Field type | Last modified |
|---------------------|------------|-------------------------------|
| Customers | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| DIM_Customer | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| DIM_Geography | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| DIM_Product | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| DIM_Supplier | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| DIM_Supplier_Test | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| Employees | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| EmployeeTerritories | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
| FACT_Order | Folder | Tue, 19 Mar 2024 00:36:19 GMT |
- Buttons:**
 - Previous
 - Next
 - Cancel

If you used the global endpoint in the connection URL, all of your available buckets appear in the left navigation view. If you used a bucket specific endpoint in the connection URL, only the specified bucket and its contents appear in the navigation view.

Navigate the storage account by selecting a folder or clicking on the expansion arrow next to a folder.

In this view, you can select one or more shortcut target locations. Choose target locations by clicking the checkbox next a folder in the left navigation view.

5. Select Next

The screenshot shows a 'New shortcut' dialog box. At the top, a message says 'Browse_Example is located in the region East US 2 EUAP. Any data sourced through this shortcut will be processed in the same region.' Below this, it says 'Review the folders and tables, and then Create. Each table and folder selected will appear as a unique shortcut in the location path.' Under 'Shortcut Location', it shows 'Browse_Example > Tables'. A table lists the selected items:

| Shortcut Name | Source | Status | Actions |
|---------------|--|---------------|---------|
| DIM_Customer | testnm1 > Northwind > delta > DIM_Customer | Yet to create | |
| DIM_Product | testnm1 > Northwind > delta > DIM_Product | Yet to create | |

At the bottom left are 'Previous' and 'Create' buttons. On the right are 'Create' and 'Cancel' buttons.

The review page allows you to verify all of your selections. Here you can see each shortcut that will be created. In the action column, you can click the pencil icon to edit the shortcut name. You can click the trash can icon to delete shortcut.

6. Select Create.

The lakehouse automatically refreshes. The shortcut appears in the left **Explorer** pane under the **Tables** section.

The screenshot shows the Power BI Home screen. The top navigation bar includes 'Home', 'Get data', 'New semantic model', and 'Open'. Below the navigation bar, there is a tooltip: 'A SQL analytics endpoint for SQL querying and a default Power BI semantic model'. The main area is titled 'Explorer' with a 'SampleLH' folder expanded. Inside 'SampleLH', there is a 'Tables' folder expanded, showing two tables: 'dim_customer' and 'dim_product', which are highlighted with a red rectangular box. There is also a 'Files' folder expanded.

Related content

- [Create a OneLake shortcut](#)
 - [Create an Azure Data Lake Storage Gen2 shortcut](#)
 - [Use OneLake shortcuts REST APIs](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Ask the community](#)

Create an Amazon S3 compatible shortcut

07/28/2025

In this article, you learn how to create an S3 compatible shortcut inside a Fabric lakehouse. For an overview of shortcuts, see [OneLake shortcuts](#).

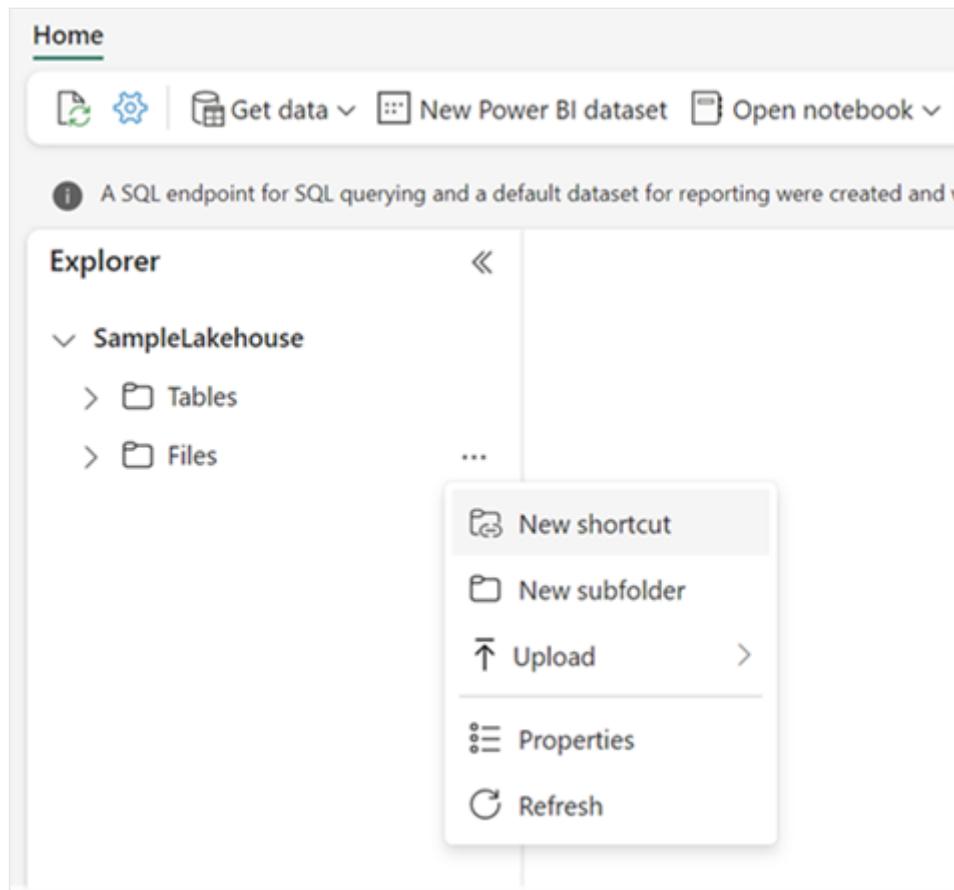
S3 compatible shortcuts can take advantage of file caching to reduce egress costs associated with cross-cloud data access. For more information, see [OneLake shortcuts Caching](#). Currently only key or secret authentication is supported for S3-compatible sources. Entra-based OAuth, Service Principal, and RoleArn are not yet supported.

Prerequisites

- If you don't have a lakehouse, create one by following these steps: [Create a lakehouse with OneLake](#).
- Ensure your chosen S3 compatible bucket and secret key credentials meet the [access and authorization requirements for S3 shortcuts](#).

Create a shortcut

1. Open a lakehouse.
2. Right-click on a directory within the **Lake view** of the lakehouse.
3. Select **New shortcut**.



Select a source

1. Under **External sources**, select **Amazon S3 compatible**.

New shortcut

Use shortcuts to quickly pull data from internal and external locations into your lakehouses, warehouses, or datasets. Shortcuts can be updated or removed from your item, but these changes will not affect the original data and its source.

Internal sources

- Microsoft OneLake

External sources

- Amazon S3
- Amazon S3 Compatible (Preview) **(highlighted)**
- Azure Data Lake Storage Gen2
- Google Cloud Storage (Preview)
- Dataverse

2. Enter the **Connection settings** according to the following table:

New shortcut

 Amazon S3 Compatible (Preview)
Generic connector [Learn more](#)

Connection settings
A^BC Url

Connection credentials

Connection [Create new connection](#) 

Connection name

Authentication kind [Access Key](#) 

Access Key Id

Secret Access Key 

 Expand table

| Field | Description | Value |
|-------------------|---|---|
| URL | The connection string for your S3 compatible endpoint. For this shortcut type, you must provide a non-bucket specific URL. This URL must allow path style bucket addressing, not just virtual hosted style. | https://s3.contoso.com |
| Connection | Previously defined connections for the specified storage location appear in the drop-down. If no connections exist, create a new connection. | Create new connection |
| Connection name | The S3 compatible connection name. | A name for your connection. |
| Access Key ID | The access key ID to be used when accessing the S3 compatible endpoint. | Your access key. |
| Secret Access Key | The secret key associated with the access key ID. | Your secret key. |

3. Select **Next**.

4. Enter a name for your shortcut.

Optionally, you can enter a sub path to select a specific folder in your S3 bucket.

 Note

Shortcut paths are case sensitive.

5. Select **Create**.

The lakehouse automatically refreshes. The shortcut appears under **Files** in the **Explorer** pane.

Related content

- [Create a OneLake shortcut](#)
- [Create an Azure Data Lake Storage Gen2 shortcut](#)

Integrate Microsoft Entra with AWS S3 shortcuts using service principal authentication

07/10/2025

You can integrate Microsoft Entra with AWS S3 using the Service Principal Name (SPN) approach. This integration enables seamless, secure access to S3 buckets using Microsoft Entra credentials, simplifying identity management and enhancing security.

Key benefits

- **Unified identity management:** Use Microsoft Entra credentials to access Amazon S3. No need to manage AWS IAM users.
- **OIDC-based authentication:** Uses OpenID Connect for secure authentication with AWS IAM roles.
- **Auditing support:** Full traceability through AWS CloudTrail to monitor role assumptions.
- **Seamless integration:** Designed to integrate with existing AWS deployments with minimal configuration changes.

Architecture

The Entra-AWS integration is built on a federated identity model that uses OpenID Connect (OIDC) to enable secure, temporary access to AWS resources. The architecture consists of the following three main components that work together to establish trust, authenticate users, and authorize access to Amazon S3 from Microsoft Fabric:

1. A **Service Principal (SPN)** registered in Microsoft Entra.
2. An **OIDC trust relationship** between AWS and Microsoft Entra.
3. A **Fabric connection** that uses temporary credentials from AWS Security Token Service (STS).

In the following sections you'll configure Microsoft Entra ID, AWS IAM, and Microsoft Fabric for secure access to Amazon S3 using the service principal-based integration. This setup establishes the necessary trust relationships and connection details required for the integration to work.

(!) Note

Only key or secret authentication is supported for S3-compatible sources; Entra-based OAuth, Service Principal, and RoleArn are not supported.

Configure Microsoft Entra ID

Step1: Register a Microsoft Entra application

- Sign in to [Azure portal](#) and navigate to Microsoft Entra ID.
- From the left-hand menu, expand **Manage > App registrations > New registration**. Fill out the following details:
 - **Name:** Enter a name for your application such as *S3AccessServicePrincipal*.
 - **Redirect URL:** Leave it blank or set to `https://localhost` if necessary.
 - Select **Register** to register your application.

(!) Note

It's recommended to use a unique Service Principal per AWS role for enhanced security

Step2: Create a client secret

- Open the Microsoft Entra application you created above.
- From the left-hand menu, expand **Manage > Certificates and secrets > New client secret** to add a new secret
- Note down the generated secret and its expiration date

Step3: Get the application details

- **Client Secret:** Get this value from the previous step
- ***Tenant ID:** From the Azure portal, navigate to **Microsoft Entra ID** and open the **Overview** tab and get the Tenant ID value.

- From the Azure portal, navigate to **Microsoft Entra ID**. From the left-hand navigation, expand the **Manage** tab and open **Enterprise applications**. Search for the application you created in the previous step. Copy the following values
 - Application ID (also known as **Client ID**)
 - Object ID

! Note

These values are from **Microsoft Entra ID > Enterprise applications tab** and NOT from **Microsoft Entra ID > App registrations tab**.

The following screenshot shows you how to get the application/client ID and object ID.

The screenshot shows the Microsoft Entra ID interface for managing enterprise applications. The top navigation bar includes 'Home', 'Microsoft Customer Led | Enterprise applications', 'Enterprise applications', and 'All applications'. The main page title is 'S3AccessServicePrincipal | Overview' under the 'Enterprise Application' category. On the left, there's a sidebar with 'Overview', 'Deployment Plan', 'Diagnose and solve problems', and a 'Manage' section containing 'Properties', 'Owners', 'Roles and administrators', 'Users and groups', 'Single sign-on', 'Provisioning', 'Application proxy', 'Self-service', and 'Custom security attributes'. The 'Properties' section is highlighted with a red box. It displays the application's name ('S3AccessServicePrincipal'), its 'Application ID' (which is currently empty), and its 'Object ID' (also empty). Below the properties, there's a 'Getting Started' section with three steps: 1. Assign users and groups (with a link to 'Assign users and groups'), 2. Provision User Accounts (with a link to 'Learn more'), and 3. Self service (with a link to 'Get started' and a magnifying glass icon).

AWS IAM configuration

Step 1: Create OIDC identity provider

- Sign into the [AWS IAM portal](#).
- Navigate to **AWS IAM** → **Identity providers** → **Add provider**
- Select *Provider type* as **Open ID Connect**
- Provider URL: `https://sts.windows.net/<your-tenant-id>`
- Audience: `https://analysis.windows.net/powerbi/connector/AmazonS3`

Add Identity provider Info

Provider details

Provider type Info

SAML
Establish trust between your AWS account and a SAML 2.0 compatible Identity Provider such as Shibboleth or Active Directory Federation Services.

OpenID Connect
Establish trust between your AWS account and Identity Provider services, such as Google or Salesforce.

Provider URL
Specify the secure OpenID Connect URL for authentication requests.
`https://`

Audience Info
Specify the client ID issued by the identity provider for your app.
`|`

Tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.
No tags associated with the resource.

[Add new tag](#)
You can add up to 50 more tags.

[Cancel](#) [Add provider](#)

Step 2: Create IAM roles

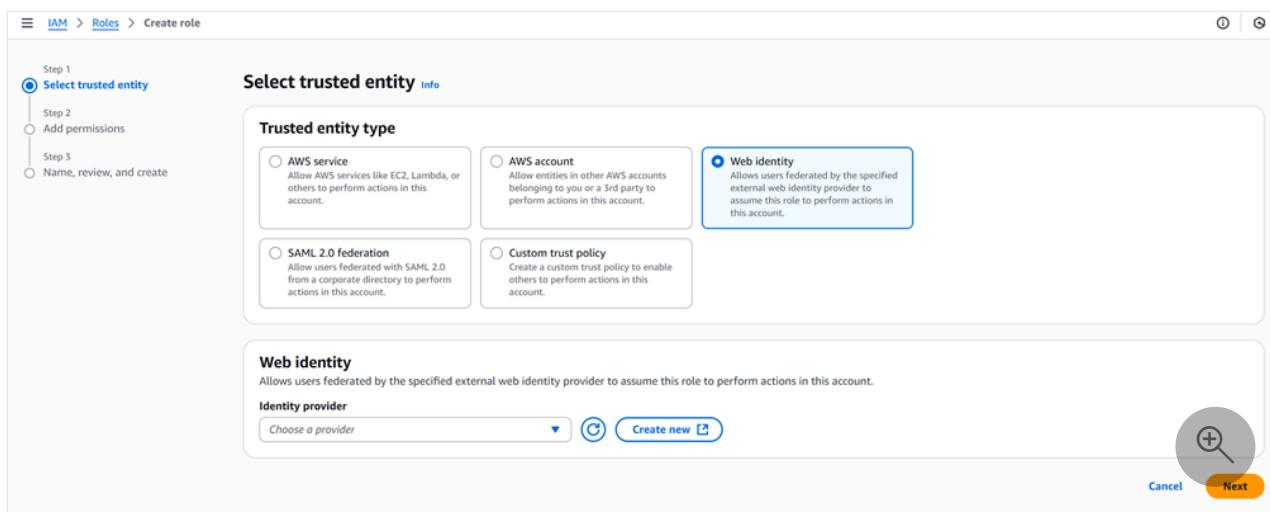
- Navigate to **AWS IAM** → **Roles** → **Create Role**
- Trusted entity type: **Web identity**
- Identity provider: Select the Open ID Connect provider created in Step 1
- Audience: `https://analysis.windows.net/powerbi/connector/AmazonS3`
- Assign appropriate S3 access policies to the role
- Ensure that the Trust policy has the service principal as one of the conditions

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<aws-account>:oidc-provider/sts.windows.net/<tenant-id>/" // (1)
      },
      "Action": "sts:AssumeRoleWithWebIdentity", // (2)
      "Condition": {
        "StringEquals": {
          "sts.windows.net/<tenant-id>/:sub": "<Object ID of the SPN that will assume this role>", // (3)
          "sts.windows.net/<tenant-id>/:aud": "https://analysis.windows.net/powerbi/connector/AmazonS3" // (4)
        }
      }
    }
  ]
}
```

Description of key fields:

1. **Principal.Federated** – Specifies the external identity provider(OIDC from Microsoft Entra ID).
2. **Action** – Grants permission to assume the role using a web identity token.
3. **Condition > :sub** – Limits which Microsoft Entra ID service can assume the role. This is the Object ID that you noted in [Step3: Get the application details](#)
4. **Condition > :aud** – Ensures the request is from Power BI's S3 connector.



Create an S3 connection in Fabric

Use Microsoft Fabric OneLake's shortcut creation interface to create the shortcut as described in the [create an S3 shortcut](#) article. Follow the same steps, but set **RoleARN** to the Amazon Resource Name (ARN) for the IAM role, and set the *Authentication Kind* to **Service Principal** and fill in the following details:

- **Tenant ID:** Tenant ID of the Microsoft Entra application
- **Service principal client ID:** The Application ID you got in the previous step.
- **Service principal key:** The client secret of the Microsoft Entra application

Security recommendations

- Use a separate service principal per AWS role for better isolation and auditability

- Rotate secrets periodically and store them securely
- Monitor AWS CloudTrail for STS-related activity

Current limitations

- This feature currently supports only the service principal-based approach; OAuth and Workspace Identity aren't yet supported.
- Access to S3 buckets behind a firewall via on-premises data gateway isn't currently supported with service principal or OAuth.

Related content

- [Create an S3 shortcut](#)
- [Create an Amazon S3 compatible shortcut](#)

Create a Google Cloud Storage (GCS) shortcut

Article • 03/31/2025

In this article, you learn how to create a Google Cloud Storage (GCS) shortcut inside a Fabric lakehouse.

For an overview of shortcuts, see [OneLake shortcuts](#). To create shortcuts programmatically, see [OneLake shortcuts REST APIs](#).

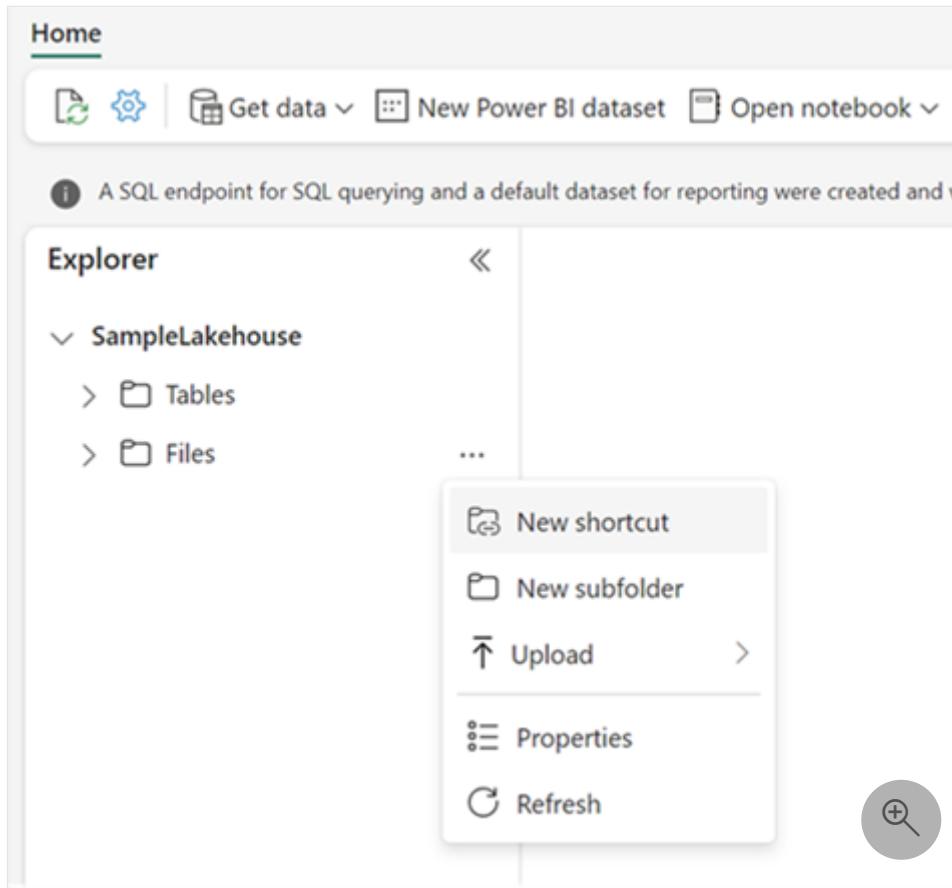
GCS shortcuts can take advantage of file caching to reduce egress costs associated with cross-cloud data access. For more information, see [OneLake shortcuts > Caching](#).

Prerequisites

- If you don't have a lakehouse, create one by following these steps: [Creating a lakehouse with OneLake](#).
- Ensure your chosen GCS bucket and user meet the [access and authorization requirements for GCS shortcuts](#).

Create a shortcut

1. Open a lakehouse.
2. Right-click on a directory within the **Lake view** of the lakehouse.
3. Select **New shortcut**.



Select a source

1. Under **External sources**, select **Google Cloud Storage**.

New shortcut

Use shortcuts to quickly pull data from internal and external locations into your lakehouses, warehouses, or datasets. Shortcuts can be updated or removed from your item, but these changes will not affect the original data and its source.

Internal sources

- Microsoft OneLake

External sources

- Amazon S3
- Amazon S3 Compatible (Preview)
- Azure Data Lake Storage Gen2
- Dataverse
- Google Cloud Storage (Preview)

2. Enter the **Connection settings** according to the following table:

New shortcut

(1) Browse Test is located in the region East US 2 EUAP. Any data sourced through this shortcut will be processed in the same region.

Google Cloud Storage
(Preview)
GCP
[Learn more](#)

Connection settings

Url
Example: `https://storage.googleapis.com`

Connection credentials

Connection
Create new connection

Connection name
`https://storage.googleapis.com`

Authentication kind
HMAC Key

Access ID
[empty field]

Secret
[empty field]

[Previous](#) [Next](#) [Cancel](#) 

[Expand table](#)

| Field | Description | Value |
|---------------------|--|---|
| URL | The connection string for your GCS bucket. The bucket name is optional. | <code>https://BucketName.storage.googleapis.com</code> <code>https://storage.googleapis.com</code> |
| Connection | Previously defined connections for the specified storage location appear in the drop-down. If no connections exist, create a new connection. | <i>Create new connection</i> |
| Connection name | The user defined name for the connection. | A name for your connection. |
| Authentication kind | Fabric uses Hash-based Message Authentication Code (HMAC) keys to access Google Cloud storage. These keys are associated with a user or service account. The account must have permission to access the data within the GCS bucket. If the bucket specific endpoint was used in the connection | HMAC Key |

| Field | Description | Value |
|-----------|--|------------------|
| | URL, the account must have the <code>storage.objects.get</code> and <code>storage.objects.list</code> permissions. If the global endpoint was used in the connection URL, the account must also have the <code>storage.buckets.list</code> permission. | |
| Access ID | The access key associated with a user or service account. For more on creating HMAC keys, see Manage HMAC Keys . | Your access key. |
| Secret | The secret for the access key. | Your secret key. |

3. Select **Next**.

4. Browse to the target location for the shortcut.

New shortcut

(1) `Browse_Test` is located in the region **East US 2 EUAP**. Any data sourced through this shortcut will be processed in the same region.

Google Cloud Storage (Preview)

Select a bucket or directory

- > bulk-1000
- < contoso-outdoors
 - > _data
 - > Customers
 - > Employees
 - > EmployeeTerritories
 - > OrderDetails
 - > Orders
 - > Products
 - > Regions
 - > Shippers
 - > Suppliers
 - > Territories
- > contosooutdoors

`contoso-outdoors`

| Name | Field type | Last modified |
|---------------------|------------|-------------------------------|
| _data | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| Customers | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| Employees | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| EmployeeTerritories | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| OrderDetails | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| Orders | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| Products | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| Regions | Folder | Sat, 16 Mar 2024 16:29:11 GMT |
| Shippers | Folder | Sat, 16 Mar 2024 16:29:11 GMT |

Previous Next Cancel

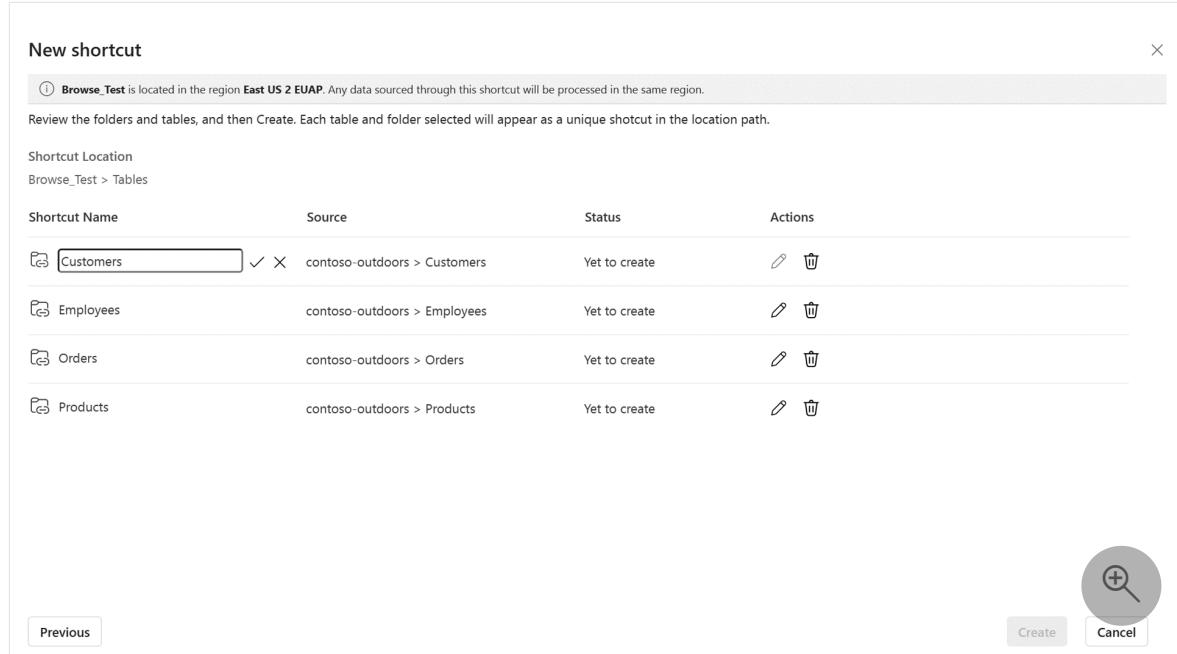
If you used the global endpoint in the connection URL, all of your available buckets appear in the left navigation view. If you used a bucket specific endpoint in the

connection URL, only the specified bucket and its contents appear in the navigation view.

Navigate the storage account by selecting a folder or clicking on the expansion arrow next to a folder.

In this view, you can select one or more shortcut target locations. Choose target locations by clicking the checkbox next a folder in the left navigation view.

5. Select Next



The screenshot shows a 'New shortcut' dialog box. At the top, it says 'New shortcut' and has a note: 'Browse_Test is located in the region East US 2 EUAP. Any data sourced through this shortcut will be processed in the same region.' Below this, it says 'Review the folders and tables, and then Create. Each table and folder selected will appear as a unique shortcut in the location path.' Under 'Shortcut Location', it shows 'Browse_Test > Tables'. A table lists four items:

| Shortcut Name | Source | Status | Actions |
|---------------|------------------------------|---------------|---------|
| Customers | contoso-outdoors > Customers | Yet to create | |
| Employees | contoso-outdoors > Employees | Yet to create | |
| Orders | contoso-outdoors > Orders | Yet to create | |
| Products | contoso-outdoors > Products | Yet to create | |

At the bottom right are 'Create' and 'Cancel' buttons, and a magnifying glass icon with a plus sign inside a circle.

The review page allows you to verify all of your selections. Here you can see each shortcut that will be created. In the action column, you can click the pencil icon to edit the shortcut name. You can click the trash can icon to delete shortcut.

6. Select Create.

The lakehouse automatically refreshes. The shortcut appears in the left **Explorer** pane.

Home



Get data ▾



New semantic model



A SQL analytics endpoint for SQL querying and a default Power

Explorer



▼ GCS_Browse_Example

▼ Tables

- Customers
- Employees
- Orders
- Products

▼ Files



Related content

- [Create a OneLake shortcut](#)
- [Create an Azure Data Lake Storage Gen2 shortcut](#)
- [Use OneLake shortcuts REST APIs](#)

Feedback

Was this page helpful?



[Provide product feedback](#) | [Ask the community](#)

Create shortcuts to on-premises data

Article • 03/31/2025

With OneLake Shortcuts, you can create virtual references to bring together data from a variety sources across clouds, regions, systems, and domains – all with no data movement or duplication. By using a Fabric on-premises data gateway (OPDG), you can also create shortcuts to on-premises data sources, such as S3 compatible storage hosted on-premises. With this feature, you can also create shortcuts to other network-restricted data sources, such as Amazon S3 or Google Cloud Storage buckets configured behind a firewall or Virtual Private Cloud (VPC).

On-premises data gateways are software agents that you install on a Windows machine and configure to connect to your data endpoints. By selecting an OPDG when creating a shortcut, you can establish network connectivity between OneLake and your data source.

This feature is available for the following shortcut types:

- Amazon S3
- S3 compatible
- Google Cloud Storage

You can use this feature in any Fabric-enabled workspace.

On-premises data gateway shortcuts can take advantage of file caching to reduce egress costs associated with cross-cloud data access. For more information, see [OneLake shortcuts > Caching](#).

In this document, we show you how to install and use these on-premises data gateways to create shortcuts to on-premises or network-restricted data.

Important

This feature is in [preview](#).

Prerequisites

- Create or identify a Fabric lakehouse that will contain your shortcut(s).
- Identify the endpoint URL associated with your Amazon S3, Google Cloud Storage, or S3 compatible location.

- For S3 compatible, the endpoint is the URL for the service, not a specific bucket.
For example:
 - `https://mys3api.contoso.com`
 - `http://10.0.1.4:9000`
- For Amazon S3, the endpoint is the URL for a specific bucket. For example:
 - `https://BucketName.s3.us-east.amazonaws.com`
- For Google Cloud Storage, the endpoint is either the URL for the bucket or the service. For example:
 - `https://storage.googleapis.com`
 - `https://bucketname.storage.googleapis.com`
- Identify the user or identity credentials that meet the [necessary access and authorization requirements](#) for your data source. Your credentials generally need to be able to list buckets, list objects, and read data.
- Identify a physical or virtual machine that:
 - Has network connectivity to your storage endpoint. This article explains how you can confirm this connectivity before creating your shortcut.
 - Allows you to install software.
- Follow [the instructions to install a standard on-premises data gateway](#) on the machine you identified. Be sure to install the latest version.
 - If your storage endpoint uses a self-signed certificate for HTTPS connections, be sure to trust this certificate on the machine hosting your gateway.

Check connectivity from gateway host

Before setting up your shortcut, follow these steps to confirm that your gateway can connect to your storage endpoint.

1. Log in to the machine hosting the gateway.
2. Install a client application that can query S3 compatible data sources, such as the Amazon Web Services Command Line Interface, WinSCP, or another tool of choice.
3. Connect to your endpoint URL and provide the credentials you identified in the prerequisite steps.
4. Ensure you can explore and read data from your storage location.

Create a shortcut

Review the instructions for creating an Amazon S3, Google Cloud Storage, or S3 compatible shortcut.

During shortcut creation, select your on-premises data gateway (OPDG) in the **Data gateway** dropdown field.



Note

If you do not see your OPDG in the **Data gateway** dropdown field and someone else created the gateway, ask them to share the gateway with you from the **Manage connections and gateways** interface.

Troubleshooting

If you encounter any connectivity issues during shortcut creation, try the following troubleshooting steps.

- As needed, ensure the machine hosting your gateway can connect to your storage endpoint. [Follow the steps to check connectivity](#).
- If you're using HTTPS and need to use a self-signed certificate, ensure the machine hosting your gateway trusts the certificate. You may need to install the self-signed certificate on the machine.

Related content

- [Create an Amazon S3 shortcut](#)
- [Create a Google Cloud Storage shortcut](#)
- [Create an S3 compatible shortcut](#)

Feedback

Was this page helpful?

Yes

No

[Provide product feedback](#) | [Ask the community](#)

Use Iceberg tables with OneLake

07/01/2025

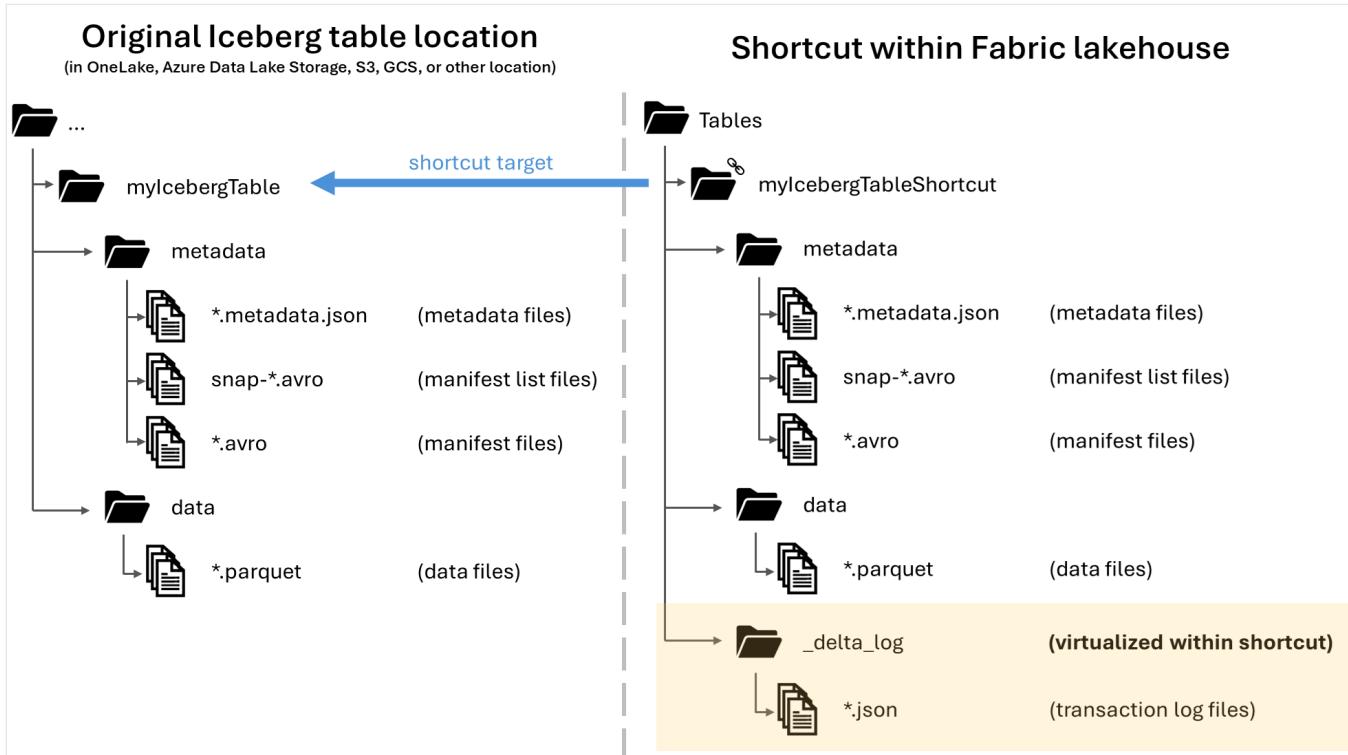
In Microsoft OneLake, you can seamlessly work with tables in both Delta Lake and Apache Iceberg formats.

This flexibility is enabled through **metadata virtualization**, a feature that allows Iceberg tables to be interpreted as Delta Lake tables, and vice versa. You can directly write Iceberg tables or create shortcuts to them, making these tables accessible across various Fabric workloads. Similarly, Fabric tables written in the Delta Lake format can be read using Iceberg readers.

When you write or create a shortcut to an Iceberg table folder, OneLake automatically generates virtual Delta Lake metadata (Delta log) for the table, enabling its use with Fabric workloads. Conversely, Delta Lake tables now include virtual Iceberg metadata, allowing compatibility with Iceberg readers.

Important

This feature is in [preview](#).



While this article includes guidance for using Iceberg tables with Snowflake, this feature is intended to work with any Iceberg tables with Parquet-formatted data files in storage.

Virtualize Delta Lake tables as Iceberg

To set up the automatic conversion and virtualization of tables from Delta Lake format to Iceberg format, follow these steps.

1. Enable automatic table virtualization of Delta Lake tables to the Iceberg format by turning on the delegated OneLake setting named **Enable Delta Lake to Apache Iceberg table format virtualization** in your workspace settings.

 **Note**

This setting controls a feature that is currently in preview. This setting will be removed in a future update when the feature is enabled for all users and is no longer in preview.

2. Make sure your Delta Lake table, or a shortcut to it, is located in the `Tables` section of your data item. The data item may be a lakehouse or another Fabric data item.

 **Tip**

If your lakehouse is schema-enabled, then your table directory will be located directly within a schema such as `dbo`. If your lakehouse is not schema-enabled, then your table directory will be directly within the `Tables` directory.

3. Confirm that your Delta Lake table has converted successfully to the virtual Iceberg format. You can do this by examining the directory behind the table.

To view the directory if your table is in a lakehouse, you can right-click the table in the Fabric UI and select **View files**.

If your table is in another data item type, such as a warehouse, a database, or a mirrored database, you will need to use a client like Azure Storage Explorer or OneLake File Explorer, rather than the Fabric UI, to view the files behind the table.

4. You should see a directory named `metadata` inside the table folder, and it should contain multiple files, including the conversion log file. Open the conversion log file to see more info about the Delta Lake to Iceberg conversion, including the timestamp of the most recent conversion and any error details.

5. If the conversion log file shows that the table was successfully converted, read the Iceberg table using your service, app, or library of choice.

Depending on what Iceberg reader you use, you will need to know either the path to the table directory or to the most recent `.metadata.json` file shown in the `metadata`

directory.

You can see the HTTP path to the latest metadata file of your table by opening the **Properties** view for the `*.metadata.json` file with the highest version number. Take note of this path.

The path to your data item's `Tables` folder may look like this:

```
https://onelake.dfs.fabric.microsoft.com/83896315-c5ba-4777-8d1c-e4ab3a7016bc/a95f62fa-2826-49f8-b561-a163ba537828/Tables/
```

Within that folder, the relative path to the latest metadata file may look like `dbo/MyTable/metadata/321.metadata.json`.

To read your virtual Iceberg table using Snowflake, [follow the steps in this guide](#).

Create a table shortcut to an Iceberg table

If you already have an Iceberg table in a storage location supported by [OneLake shortcuts](#), follow these steps to create a shortcut and have your Iceberg table appear with the Delta Lake format.

1. **Locate your Iceberg table.** Find where your Iceberg table is stored, which could be in Azure Data Lake Storage, OneLake, Amazon S3, Google Cloud Storage, or an S3 compatible storage service.

Note

If you're using Snowflake and aren't sure where your Iceberg table is stored, you can run the following statement to see the storage location of your Iceberg table.

```
SELECT SYSTEM$GET_ICEBERG_TABLE_INFORMATION('<table_name>');
```

Running this statement returns a path to the metadata file for the Iceberg table. This path tells you which storage account contains the Iceberg table. For example, here's the relevant info to find the path of an Iceberg table stored in Azure Data Lake Storage:

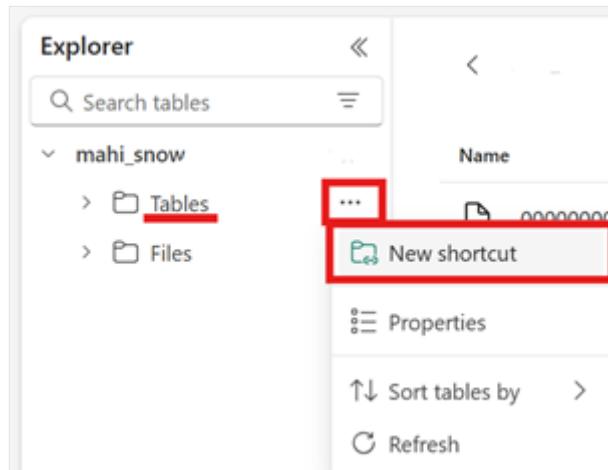
```
{"metadataLocation": "azure://<storage_account_path>/<path_within_storage>/<table_name>/metadata/00001-389700a2-977f-47a2-9f5f-7fd80a0d41b2.metadata.json", "status": "success"}
```

Your Iceberg table folder needs to contain a `metadata` folder, which itself contains at least one file ending in `.metadata.json`.

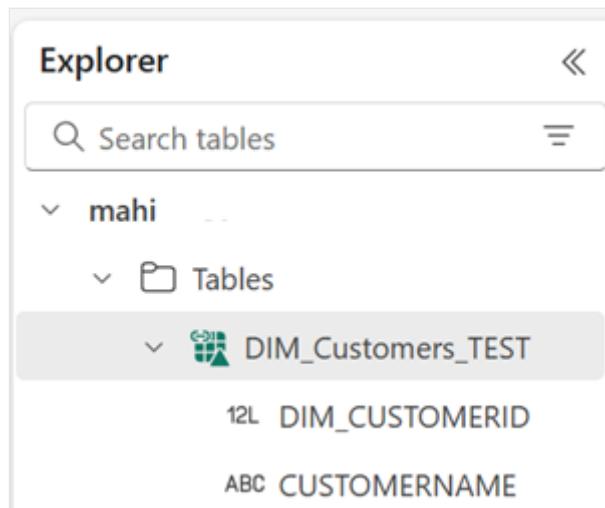
2. In your Fabric lakehouse, create a new table shortcut in the Tables area of a lakehouse.

💡 Tip

If you see schemas such as dbo under the Tables folder of your lakehouse, then the lakehouse is schema-enabled. In this case, right-click on the schema and create a table shortcut under the schema.



3. For the target path of your shortcut, select the Iceberg table folder. The Iceberg table folder contains the `metadata` and `data` folders.
4. Once your shortcut is created, you should automatically see this table reflected as a Delta Lake table in your lakehouse, ready for you to use throughout Fabric.



If your new Iceberg table shortcut doesn't appear as a usable table, check the [Troubleshooting](#) section.

Troubleshooting

The following tips can help make sure your Iceberg tables are compatible with this feature:

Check the folder structure of your Iceberg table

Open your Iceberg folder in your preferred storage explorer tool, and check the directory listing of your Iceberg folder in its original location. You should see a folder structure like the following example.

```
../
|-- MyIcebergTable123/
|   |-- data/
|   |   |-- A5WYPKG0_2o_APgwTeNOAxg_0_1_002.parquet
|   |   |-- A5WYPKG0_2o_AAIBON_h9Rc_0_1_003.parquet
|   |-- metadata/
|       |-- 00000-1bdf7d4c-dc90-488e-9dd9-2e44de30a465.metadata.json
|       |-- 00001-08bf3227-b5d2-40e2-a8c7-2934ea97e6da.metadata.json
|       |-- 00002-0f6303de-382e-4ebc-b9ed-6195bd0fb0e7.metadata.json
|       |-- 1730313479898000000-Kws8nlgCX2QxoDHYHm4uMQ.avro
|       |-- 1730313479898000000-0dsKRrRogW_PVK9njHIqAA.avro
|       |-- snap-1730313479898000000-9029d7a2-b3cc-46af-96c1-ac92356e93e9.avro
|       |-- snap-1730313479898000000-913546ba-bb04-4c8e-81be-342b0cbc5b50.avro
```

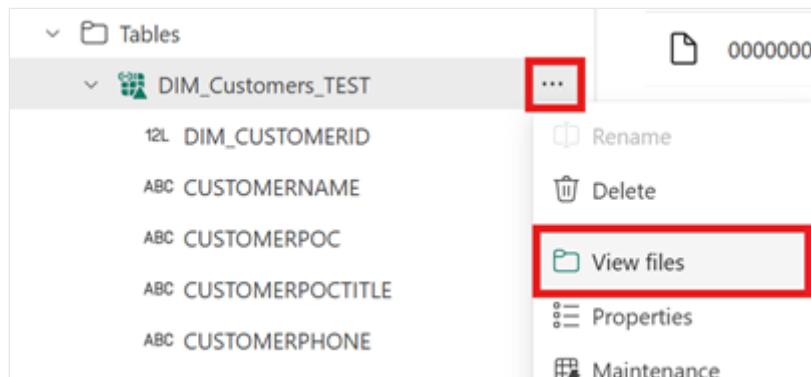
If you don't see the metadata folder, or if you don't see files with the extensions shown in this example, then you might not have a properly generated Iceberg table.

Check the conversion log

When an Iceberg table is virtualized as a Delta Lake table, a folder named `_delta_log/` can be found inside the shortcut folder. This folder contains the Delta Lake format's metadata (the Delta log) after successful conversion.

This folder also includes the `latest_conversion_log.txt` file, which contains the latest attempted conversion's success or failure details.

To see the contents of this file after creating your shortcut, open the menu for the Iceberg table shortcut under Tables area of your lakehouse and select **View files**.



You should see a structure like the following example:

```
Tables/
|-- MyIcebergTable123/
  |-- data/
    |-- <data files>
  |-- metadata/
    |-- <metadata files>
  |-- _delta_log/  <-- Virtual folder. This folder doesn't exist in the
original location.
    |-- 000000000000000000000000.json
    |-- latest_conversion_log.txt  <-- Conversion log with latest
success/failure details.
```

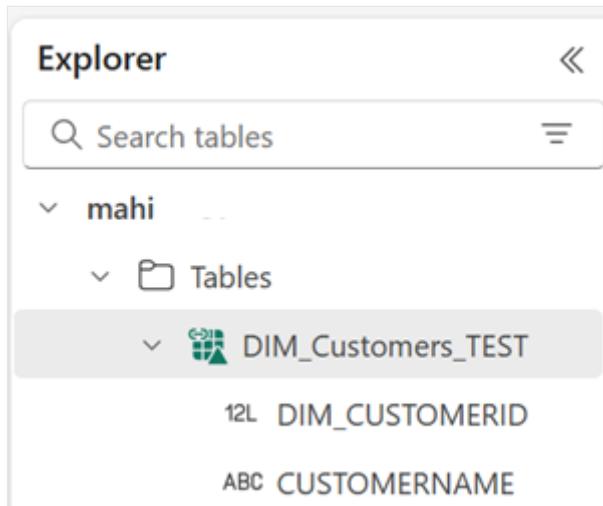
Open the conversion log file to see the latest conversion time or failure details. If you don't see a conversion log file, [conversion wasn't attempted](#).

If conversion wasn't attempted

If you don't see a conversion log file, then the conversion wasn't attempted. Here are two common reasons why conversion isn't attempted:

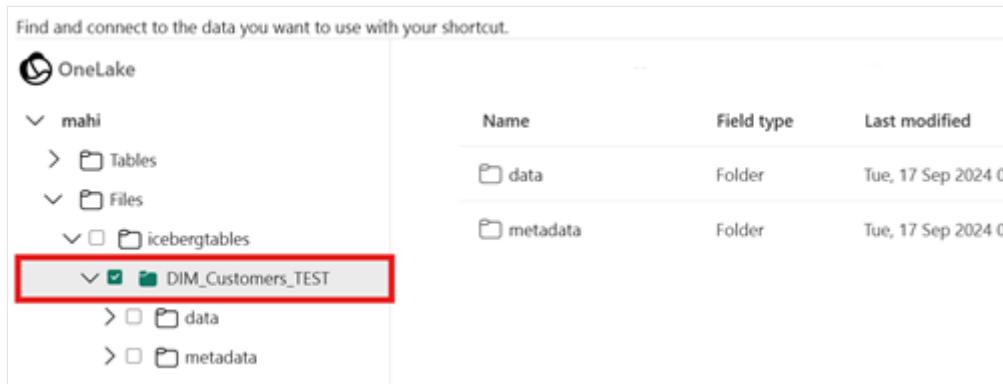
- The shortcut wasn't created in the right place.

In order for a shortcut to an Iceberg table to get converted to the Delta Lake format, the shortcut must be placed directly under the Tables folder of a non-schema-enabled lakehouse. You shouldn't place the shortcut in the Files section or under another folder if you want the table to be automatically virtualized as a Delta Lake table.



- The shortcut's target path is not the Iceberg folder path.

When you create the shortcut, the folder path you select in the target storage location must only be the Iceberg table folder. This folder *contains* the `metadata` and `data` folders.



"Fabric capacity region cannot be validated" error message in Snowflake

If you are using Snowflake to write a new Iceberg table to OneLake, you might see the following error message:

Fabric capacity region cannot be validated. Reason: 'Invalid access token. This may be due to authentication and scoping. Please verify delegated scopes.'

If you see this error, have your Fabric tenant admin double-check that you've enabled both tenant settings mentioned in the [Write an Iceberg table to OneLake using Snowflake](#) section:

1. In the upper-right corner of the Fabric UI, open **Settings**, and select **Admin portal**.
2. Under **Tenant settings**, in the **Developer settings** section, enable the setting labeled **Service principals can use Fabric APIs**.
3. In the same area, in the **OneLake settings** section, enable the setting labeled **Users can access data stored in OneLake with apps external to Fabric**.

Limitations and considerations

Keep in mind the following temporary limitations when you use this feature:

- **Supported data types**

The following Iceberg column data types map to their corresponding Delta Lake types using this feature.

 Expand table

| Iceberg column type | Delta Lake column type | Comments |
|------------------------|---------------------------|--|
| int | integer | |
| long | long | See Type width issue . |
| float | float | |
| double | double | See Type width issue . |
| decimal(P, S) | decimal(P, S) | See Type width issue . |
| boolean | boolean | |
| date | date | |
| timestamp | timestamp_ntz | The <code>timestamp</code> Iceberg data type doesn't contain time zone information. The <code>timestamp_ntz</code> Delta Lake type isn't fully supported across Fabric workloads. We recommend the use of timestamps with time zones included. |
| timestamptz | timestamp | In Snowflake, to use this type, specify <code>timestamp_ltz</code> as the column type during Iceberg table creation. More info on Iceberg data types supported in Snowflake can be found here . |
| string | string | |
| binary | binary | |
| time | N/A | Not supported |

- **Type width issue**

If you use Snowflake to write your Iceberg table and the table contains column types `INT64`, `double`, or `Decimal` with precision ≥ 10 , then the resulting virtual Delta Lake table may not be consumable by all Fabric engines. You may see errors such as:

```
Parquet column cannot be converted in file ... Column: [ColumnA], Expected: decimal(18,4), Found: INT32.
```

We're working on a fix for this issue.

Workaround: If you're using the Lakehouse table preview UI and see this issue, you can resolve this error by switching to the SQL Endpoint view (top right corner, select Lakehouse view, switch to SQL Endpoint) and previewing the table from there. If you then switch back to the Lakehouse view, the table preview should display properly.

If you're running a Spark notebook or job and encounter this issue, you can resolve this error by setting the `spark.sql.parquet.enableVectorizedReader` Spark configuration to `false`. Here's an example PySpark command to run in a Spark notebook:

```
spark.conf.set("spark.sql.parquet.enableVectorizedReader","false")
```

- **Iceberg table metadata storage isn't portable**

The metadata files of an Iceberg table refer to each other using absolute path references. If you copy or move an Iceberg table's folder contents to another location without rewriting the Iceberg metadata files, the table becomes unreadable by Iceberg readers, including this OneLake feature.

Workaround:

If you need to move your Iceberg table to another location to use this feature, use the tool that originally wrote the Iceberg table to write a new Iceberg table in the desired location.

- **Iceberg table folders must contain only one set of metadata files**

If you drop and recreate an Iceberg table in Snowflake, the metadata files aren't cleaned up. This behavior is by design, in support of the `UNDROP` feature in Snowflake. However, because your shortcut points directly to a folder and that folder now has multiple sets of

metadata files within it, we can't convert the table until you remove the old table's metadata files.

Conversion will fail if more than one set of metadata files are found in the Iceberg table's metadata folder.

Workaround:

To ensure the converted table reflects the correct version of the table:

- Ensure you aren't storing more than one Iceberg table in the same folder.
- Clean up any contents of an Iceberg table folder after dropping it, before recreating the table.

- **Metadata changes not immediately reflected**

If you make metadata changes to your Iceberg table, such as adding a column, deleting a column, renaming a column, or changing a column type, the table may not be reconverted until a data change is made, such as adding a row of data.

We're working on a fix that picks up the correct latest metadata file that includes the latest metadata change.

Workaround:

After making the schema change to your Iceberg table, add a row of data or make any other change to the data. After that change, you should be able to refresh and see the latest view of your table in Fabric.

- **Region availability limitation**

The feature isn't yet available in the following regions:

- Qatar Central
- Norway West

Workaround:

Workspaces attached to Fabric capacities in other regions can use this feature. [See the full list of regions where Microsoft Fabric is available.](#)

- **Private links not supported**

This feature isn't currently supported for tenants or workspaces that have private links enabled.

We're working on an improvement to remove this limitation.

- **OneLake shortcuts must be same-region**

We have a temporary limitation on the use of this feature with shortcuts that point to OneLake locations: the target location of the shortcut must be in the same region as the shortcut itself.

We're working on an improvement to remove this requirement.

Workaround:

If you have a OneLake shortcut to an Iceberg table in another lakehouse, be sure that the other lakehouse is associated with a capacity in the same region.

- **Certain Iceberg partition transform types are not supported**

Currently, the [Iceberg partition types](#) `bucket[N]`, `truncate[W]`, and `void` are not supported.

If the Iceberg table being converted contains these partition transform types, virtualization to the Delta Lake format will not succeed.

We're working on an improvement to remove this limitation.

Related content

- [Use Snowflake to write or read Iceberg tables in OneLake.](#)
- [Learn more about Fabric and OneLake security.](#)
- [Learn more about OneLake shortcuts.](#)

Use Snowflake with Iceberg tables in OneLake

Microsoft OneLake can be used with Snowflake for storage and access of Apache Iceberg tables.

Follow this guide to use Snowflake on Azure to:

- write Iceberg tables directly to OneLake
- read virtual Iceberg tables converted from the Delta Lake format

Important

This feature is in [preview](#).

Before getting started, follow the pre-requisite steps shown below.

Prerequisite

To use Snowflake on Azure to write or read Iceberg tables with OneLake, your Snowflake account's identity in Entra ID needs to be able to communicate with Fabric. Enable the Fabric tenant-level settings that allow service principals to [call Fabric APIs](#) and to [call OneLake APIs](#).

Write an Iceberg table to OneLake using Snowflake on Azure

If you use Snowflake on Azure, you can write Iceberg tables to OneLake by following these steps:

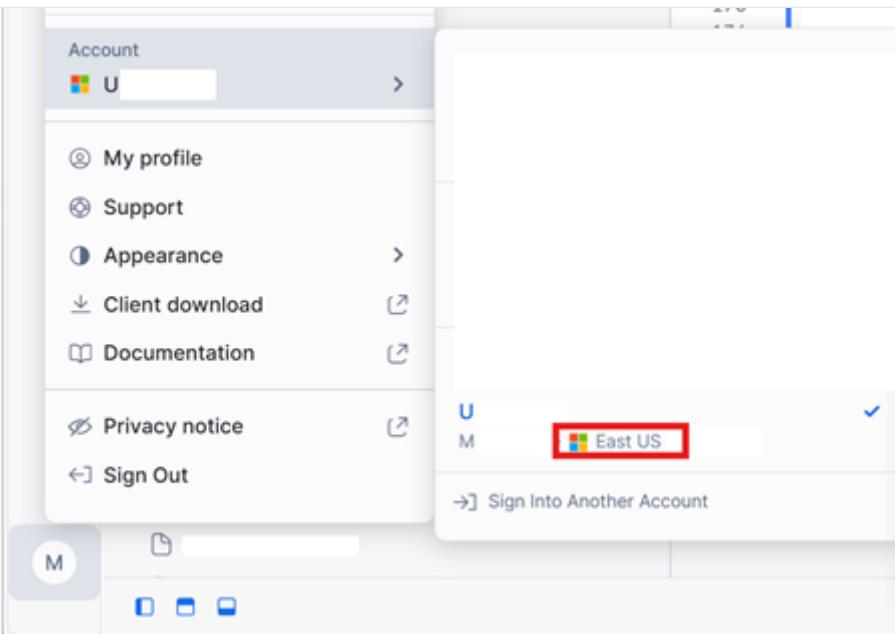
1. Make sure your Fabric capacity is in the same Azure location as your Snowflake instance.

Identify the location of the Fabric capacity associated with your Fabric lakehouse. Open the settings of the Fabric workspace that contains your lakehouse.

Workspace settings

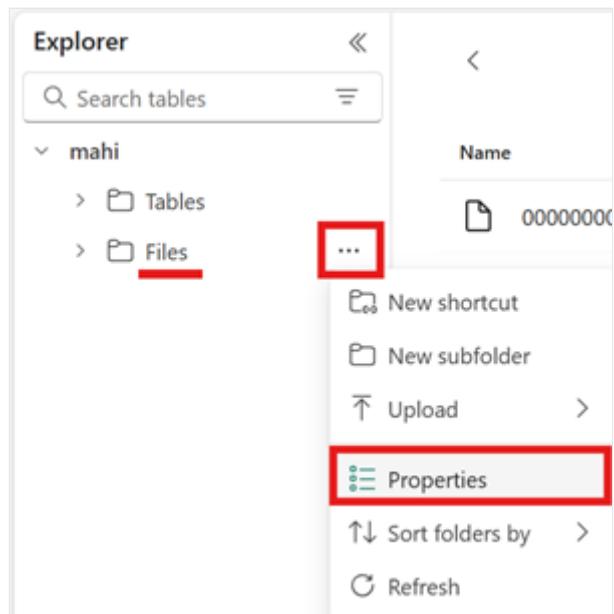
The screenshot shows the 'License info' section of the workspace settings. On the left, there's a sidebar with icons for General, License info (which is selected and highlighted in blue), Azure connections, System storage, Git integration, and OneLake. The main area has a title 'License info' and a subtitle 'Choose a license for this workspace.' Below this is a 'License Configuration' section with 'Current license' and 'License capacity'. A red box highlights the 'Region: West Central US' field. At the bottom, it says 'Semantic model storage format' and 'Small semantic model storage format'.

In the bottom-left corner of your Snowflake on Azure account interface, check the Azure region of the Snowflake account.

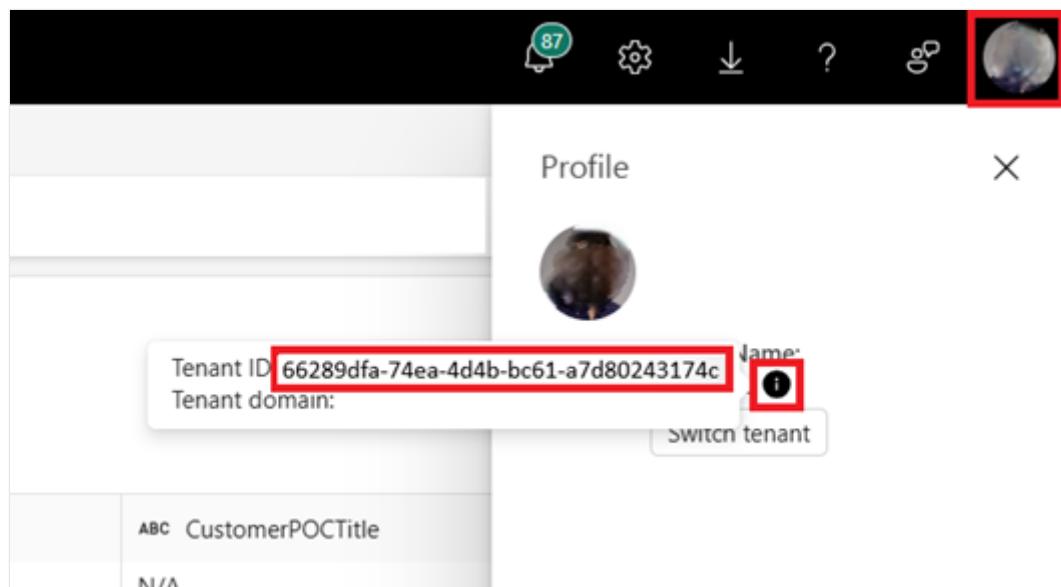


If these regions are different, you need to use a different Fabric capacity in the same region as your Snowflake account.

2. Open the menu for the Files area of your lakehouse, select Properties, and copy the URL (the HTTPS path) of that folder.



3. Identify your Fabric tenant ID. Select your user profile in the top-right corner of the Fabric UI, and hover over the info bubble next to your Tenant Name. Copy the Tenant ID.



4. In Snowflake, set up your EXTERNAL VOLUME using the path to the Files folder in your lakehouse. [More info on setting up Snowflake external volumes can be found here.](#)

Note

Snowflake requires the URL scheme to be `azure://`, so be sure to change the path from `https://` to `azure://`.

SQL

```
CREATE OR REPLACE EXTERNAL VOLUME onelake_write_exvol
STORAGE_LOCATIONS =
(
```

```
(  
    NAME = 'onelake_write_exvol'  
    STORAGE_PROVIDER = 'AZURE'  
    STORAGE_BASE_URL = 'azure://<path_to_lakehouse>/Files/icebergtables'  
    AZURE_TENANT_ID = '<Tenant_ID>'  
)  
);
```

In this sample, any tables created using this external volume are stored in the Fabric lakehouse, within the `Files/icebergtables` folder.

- Now that your external volume is created, run the following command to retrieve the consent URL and name of the application that Snowflake uses to write to OneLake. This application is used by any other external volume in your Snowflake account.

SQL

```
DESC EXTERNAL VOLUME onelake_write_exvol;
```

The output of this command returns the `AZURE_CONSENT_URL` and `AZURE_MULTI_TENANT_APP_NAME` properties. Take note of both values. The Azure multitenant app name looks like `<name>_<number>`, but you only need to capture the `<name>` portion.

- Open the consent URL from the previous step in a new browser tab, if you haven't done this previously. If you would like to proceed, consent to the required application permissions, if prompted. You may be redirected to the main Snowflake website.
- Back in Fabric, open your workspace and select **Manage access**, then **Add people or groups**. Grant the application used by your Snowflake external volume the permissions needed to write data to lakehouses in your workspace. We recommend granting the **Contributor** role.
- Back in Snowflake, use your new external volume to create an Iceberg table.

SQL

```
CREATE OR REPLACE ICEBERG TABLE MYDATABASE.PUBLIC.Inventory (  
    InventoryId int,  
    ItemName STRING  
)  
EXTERNAL_VOLUME = 'onelake_write_exvol'  
CATALOG = 'SNOWFLAKE'  
BASE_LOCATION = 'Inventory/' ;
```

After running this statement, a new Iceberg table folder named `Inventory` has been created within the folder path defined in the external volume.

9. Add some data to your Iceberg table.

SQL

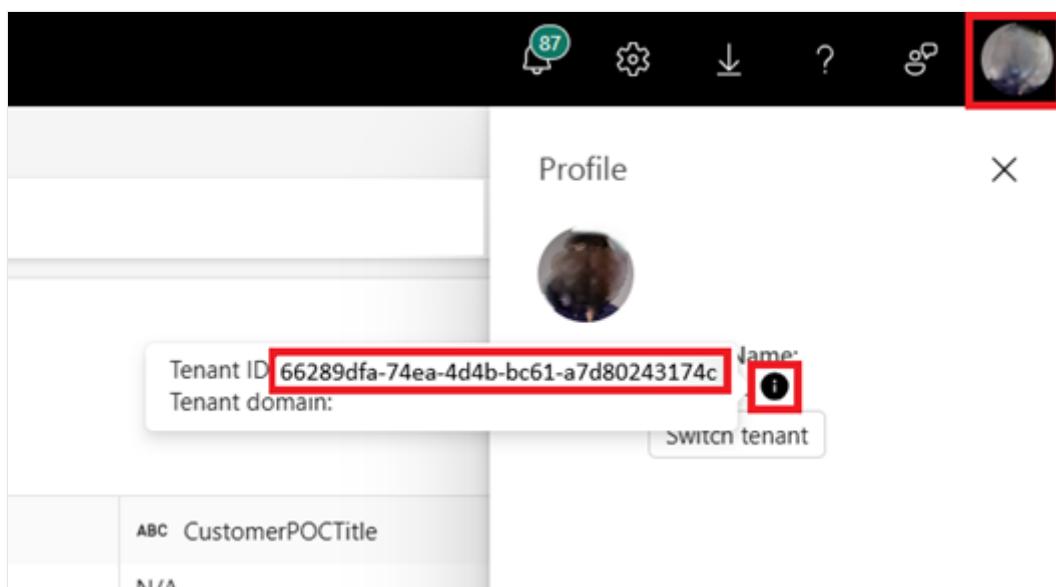
```
INSERT INTO MYDATABASE.PUBLIC.Inventory  
VALUES  
(123456, 'Amatriciana');
```

10. Finally, in the Tables area of the same lakehouse, [you can create a OneLake shortcut to your Iceberg table](#). Through that shortcut, your Iceberg table appears as a Delta Lake table for consumption across Fabric workloads.

Read a virtual Iceberg table from OneLake using Snowflake on Azure

To use Snowflake on Azure to read a virtual Iceberg table based on a Delta Lake table in Fabric, follow these steps.

1. Follow [the guide to confirm your Delta Lake table has converted successfully to Iceberg](#), and take note of the path to the data item containing your table, as well as your table's most recent `*.metadata.json` file.
2. Identify your Fabric tenant ID. Select your user profile in the top-right corner of the Fabric UI, and hover over the info bubble next to your **Tenant Name**. Copy the Tenant ID.



3. In Snowflake, set up your `EXTERNAL VOLUME` using the path to the `Tables` folder of the data item that contains your table. [More info on setting up Snowflake external volumes can be found here.](#)

SQL

```
CREATE OR REPLACE EXTERNAL VOLUME onelake_read_exvol
STORAGE_LOCATIONS =
(
  (
    NAME = 'onelake_read_exvol'
    STORAGE_PROVIDER = 'AZURE'
    STORAGE_BASE_URL = 'azure://<path_to_data_item>/Tables/'
    AZURE_TENANT_ID = '<Tenant_ID>'
  )
)
ALLOW_WRITES = false;
```

ⓘ Note

Snowflake requires the URL scheme to be `azure://`, so be sure to change `https://` to `azure://`.

Replace `<path_to_data_item>` with the path to your data item, such as

`https://onelake.dfs.fabric.microsoft.com/83896315-c5ba-4777-8d1c-e4ab3a7016bc/a95f62fa-2826-49f8-b561-a163ba537828`.

- Now that your external volume is created, run the following command to retrieve the consent URL and name of the application that Snowflake uses to write to OneLake. This application is used by any other external volume in your Snowflake account.

SQL

```
DESC EXTERNAL VOLUME onelake_read_exvol;
```

The output of this command returns the `AZURE_CONSENT_URL` and `AZURE_MULTI_TENANT_APP_NAME` properties. Take note of both values. The Azure multitenant app name looks like `<name>_<number>`, but you only need to capture the `<name>` portion.

- Open the consent URL from the previous step in a new browser tab, if you haven't done this previously. If you would like to proceed, consent to the required application permissions, if prompted. You may be redirected to the main Snowflake website.
- Back in Fabric, open your workspace and select **Manage access**, then **Add people or groups**. Grant the application used by your Snowflake external volume the permissions needed to read data from data items in your workspace.

💡 Tip

You may instead choose to grant permissions at the data item level, if you wish.

[Learn more about OneLake data access.](#)

7. Create the `CATALOG INTEGRATION` object in Snowflake, if you haven't done this previously. This is required by Snowflake to reference existing Iceberg tables in storage.

SQL

```
CREATE CATALOG INTEGRATION onelake_catalog_integration
CATALOG_SOURCE = OBJECT_STORE
TABLE_FORMAT = ICEBERG
ENABLED = TRUE;
```

8. Back in Snowflake, create an Iceberg table referencing the latest metadata file for the virtualized Iceberg table in OneLake.

SQL

```
CREATE OR REPLACE ICEBERG TABLE MYDATABASE.PUBLIC.<TABLE_NAME>
EXTERNAL_VOLUME = 'onelake_read_exvol'
CATALOG = onelake_catalog_integration
METADATA_FILE_PATH = '<metadata_file_path>';
```

ⓘ Note

Replace `<TABLE_NAME>` with your table name, and `<metadata_file_path>` with your Iceberg table's metadata file path, such as `dbo/MyTable/metadata/321.metadata.json`.

After running this statement, you now have a reference to your virtualized Iceberg table that you can now query using Snowflake.

9. Query your virtualized Iceberg table by running the following statement.

SQL

```
SELECT TOP 10 * FROM MYDATABASE.PUBLIC.<TABLE_NAME>;
```

Troubleshooting

See the [troubleshooting](#) and [limitations and considerations](#) sections of our documentation of OneLake table format virtualization and conversion between Delta Lake and Apache Iceberg table formats.

Last updated on 07/01/2025

OneLake shortcut security

OneLake shortcuts serve as pointers to data residing in various storage accounts, whether within OneLake itself or in external systems like Azure Data Lake Storage (ADLS). This article looks at the permissions required to create shortcuts and access data using them.

To ensure clarity around the components of a shortcut this document uses the following terms:

- Target path: The location that a shortcut points to.
- Shortcut path: The location where the shortcut appears.

Create and delete shortcuts

To create a shortcut a user needs to have Write permission on the Fabric Item where the shortcut is being created. In addition, the user needs Read access to the data the shortcut is pointing to. Shortcuts to external sources might require certain permissions in the external system. The [What are shortcuts?](#) article has the full list of shortcut types and required permissions.

 Expand table

| Capability | Permission on shortcut path | Permission on target path |
|-------------------|-----------------------------|---------------------------|
| Create a shortcut | Write ² | ReadAll ¹ |
| Delete a shortcut | Write ² | N/A |

¹ If [OneLake security](#) is enabled, the user needs to be in a role that grants access to the target path. ² If [OneLake data access roles](#) is enabled, the user needs to be in a role that grants access to the target path.

Accessing shortcuts

A combination of the permissions in the shortcut path and the target path governs the permissions for shortcuts. When a user accesses a shortcut, the most restrictive permission of the two locations is applied. Therefore, a user that has read/write permissions in the lakehouse but only read permissions in the target path can't write to the target path. Likewise, a user that only has read permissions in the lakehouse but read/write in the target path also can't write to the target path.

This table shows the permissions needed for each shortcut action.

| Capability | Permission on shortcut path | Permission on target path |
|---|-----------------------------|---------------------------|
| Read file/folder content of shortcut | ReadAll ¹ | ReadAll ¹ |
| Write to shortcut target location | Write ² | Write ² |
| Read data from shortcuts in table section of the lakehouse via TDS endpoint | Read | ReadAll ³ |

¹ If [OneLake security](#) is enabled the user needs to be in a role that grants access to the target path.

² Alternatively, OneLake security with ReadWrite permission on the shortcut path.

i Important

³ **Exception to identity passthrough:** While OneLake security typically passes through the calling user's identity to enforce permissions, certain query engines operate differently. When accessing shortcut data through **Power BI semantic models using DirectLake over SQL or T-SQL engines configured for Delegated identity mode**, these engines don't pass through the calling user's identity to the shortcut target. Instead, they use the **item owner's identity** to access the data, and then apply OneLake security roles to filter what the calling user can see.

This means:

- The shortcut target is accessed using the item owner's permissions (not the end user's)
- OneLake security roles still determine what data the end user can read
- Any permissions configured directly at the shortcut target path for the end user are bypassed

OneLake security

[OneLake security \(preview\)](#) is a feature that enables you to apply role-based access control (RBAC) to your data stored in OneLake. You can define security roles that grant read access to specific tables and folders within a Fabric item, and assign them to users or groups. The access permissions determine what users will have across all engines in Fabric, ensuring consistent access control.

Users in the Admin, Member, and Contributor roles have full access to read data from a shortcut regardless of the OneLake data access roles defined. However they still need access on both the shortcut path and target path as mentioned in [Workspace roles](#).

Users in the Viewer role or that had a lakehouse shared with them directly have access restricted based on if the user has access through a OneLake data access role. For more information on the access control model with shortcuts, see [Data Access Control Model in OneLake](#).

Users in Viewer roles can create shortcuts if they have ReadWrite permissions on the path where the shortcut is created.

The following table illustrates the necessary permissions to perform shortcut operations.

[] [Expand table](#)

| Shortcut operation | Permission on shortcut path | Permission on target path |
|---------------------------|---|---|
| Create | Fabric Read <i>and</i> OneLake security ReadWrite | OneLake security Read |
| Read (GET/LIST shortcuts) | Fabric Read <i>and</i> OneLake security Read | N/A |
| Update | Fabric Read <i>and</i> OneLake security ReadWritten | OneLake security Read (on the new target) |
| Delete | Fabric Read <i>and</i> OneLake security ReadWrite | N/A |

Shortcut auth models

Shortcuts use two authentication models with OneLake security: passthrough and delegated.

In the passthrough model, the shortcut accesses data in the target location by 'passing' the user's identity to the target system. This ensures that any user accessing the shortcut is only able to see whatever they have access to in the target.

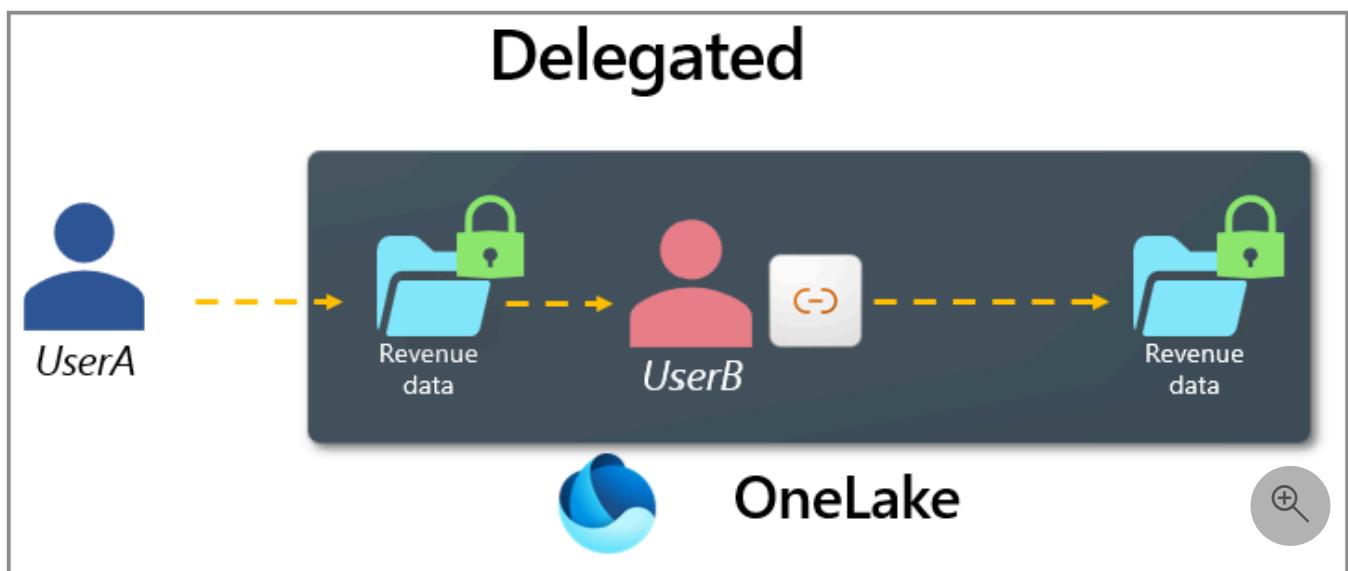
With OneLake to OneLake shortcuts, only passthrough mode is supported. This design ensures that the source system retains full control over its data. Organizations benefit from enhanced security because there's no need to replicate or redefine access controls for the shortcut. However, it's important to understand that security for OneLake shortcuts can't be modified directly from the downstream item. Any changes to access permissions must be made at the source location.

Passthrough



Delegated shortcuts access data by using some intermediate credential, such as another user or an account key. These shortcuts allow for permission management to be separated or 'delegated' to another team or downstream user to manage. Delegated shortcuts always break the flow of security from one system to another. All delegated shortcuts in OneLake can have OneLake security roles defined for them.

All shortcuts from OneLake to external systems (multicloud shortcuts) like AWS S3 or Google Cloud Storage are delegated. This allows users to connect to the external system without being given direct access. OneLake security can then be configured on the shortcut to limit what data in the external system can be accessed



OneLake security limitations

- In addition to OneLake security access to the target path, accessing external shortcuts via Spark or direct API calls also require read permissions on the item containing the external shortcut path.

Related content

- [What are shortcuts?](#)
 - [Create a OneLake shortcut](#)
 - [Use OneLake shortcuts REST APIs](#)
 - [Data Access Control Model in OneLake.](#)
-

Last updated on 11/18/2025

Manage connections for shortcuts

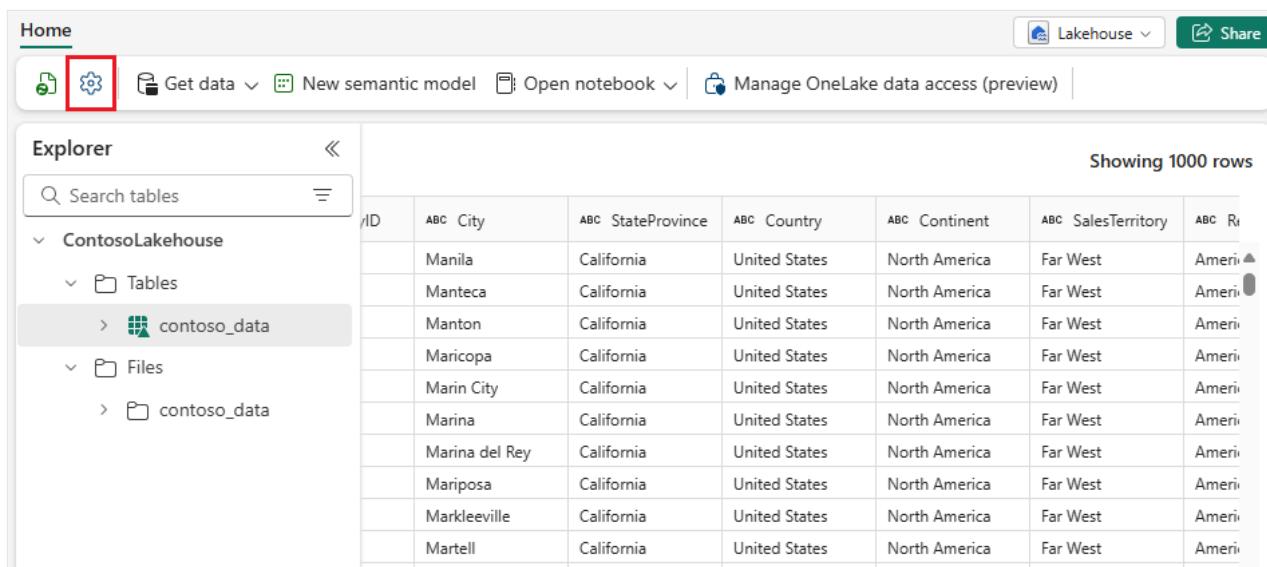
Article • 04/25/2025

Shortcuts in OneLake use shared cloud connections to access the cloud resources where your data is stored. These connections can be managed on a per-shortcut basis, but you can also view and update connections in bulk to keep all of your shortcuts working efficiently.

View shortcut connections

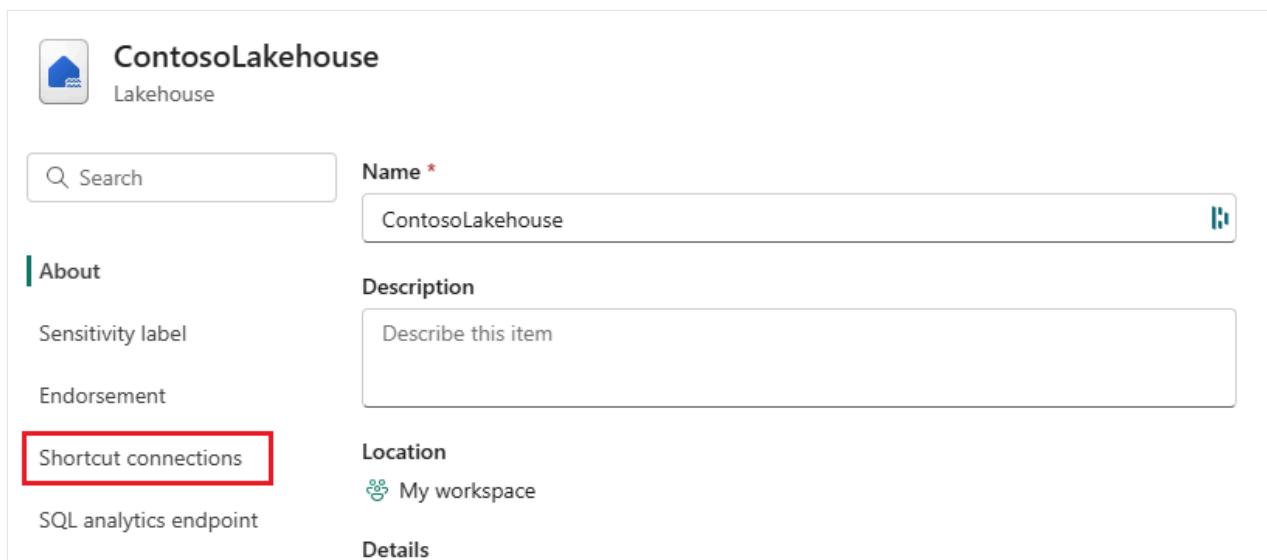
You can view and manage all existing cloud connections for shortcuts in a single lakehouse.

1. In the [Microsoft Fabric portal](#), navigate to your lakehouse.
2. Select **Settings**.



The screenshot shows the Microsoft Fabric portal interface. At the top, there's a navigation bar with links for 'Home', 'Lakehouse', and 'Share'. Below the navigation bar is the 'Settings' gear icon, which is highlighted with a red box. The main area is titled 'Explorer' and shows a tree view of 'ContosoLakehouse' containing 'Tables' and 'Files'. Under 'Tables', 'contoso_data' is selected. To the right of the tree view is a large table titled 'Showing 1000 rows' with columns for ID, City, StateProvince, Country, Continent, SalesTerritory, and Region. The table lists various California cities like Manila, Manteca, Manton, Maricopa, Marin City, Marina, Marina del Rey, Mariposa, Markleeville, and Martell, all categorized under North America, Far West, and America.

3. Select **Shortcut connections**.



The screenshot shows the 'ContosoLakehouse' settings page. On the left, there's a sidebar with sections for 'About', 'Sensitivity label', 'Endorsement', and 'Shortcut connections'. The 'Shortcut connections' section is highlighted with a red box. On the right, there are input fields for 'Name' (set to 'ContosoLakehouse') and 'Description' (with placeholder text 'Describe this item'). Below these are sections for 'Location' (set to 'My workspace') and 'Details'. There's also a small 'Edit' icon next to the name field.

4. On the **Manage OneLake shortcut connections** page, you can view all connections. The **Action required** section highlights any broken connections that need attention. You can also see how many shortcuts share each connection.

Replace shortcut connections

There are many reasons that you might have to replace a cloud connection. Maybe the connection is broken, maybe the user that created that connection left your organization and you can't access the connection anymore, or maybe you want to switch to a different connection that uses a different authentication method.

1. On the **Manage OneLake shortcut connections** page, select **Replace** for the connection that you want to update.

Manage OneLake shortcut connections

Track existing connections and replace broken connections in this Lakehouse. [Learn more](#)

Action required

The following connections are broken. You can replace the connection to resolve the issue. Replacing a connection can impact multiple shortcuts.

| | |
|--|--|
|  Azure Data Lake Storage Gen2 |  Replace |
| Shortcuts 1 | URL https://contoso-adls.dfs.core.windows.net |

All connections

| | |
|---|---|
|  Amazon S3 |  Replace |
| Shortcuts 5 | URL s3://contoso-s3 |

2. Select either **Existing connection** or **Create new connection**.

3. Provide the new connection information.

- For an existing connection, use the drop-down menu to select the connection, then select **Save**.
- For a new connection, provide the connection settings and credentials, then select **Save**.

Once the new cloud connection is established, all of the shortcuts that used the old connection are updated to use the new connection.

Access Fabric OneLake shortcuts in an Apache Spark notebook

Article • 06/05/2024

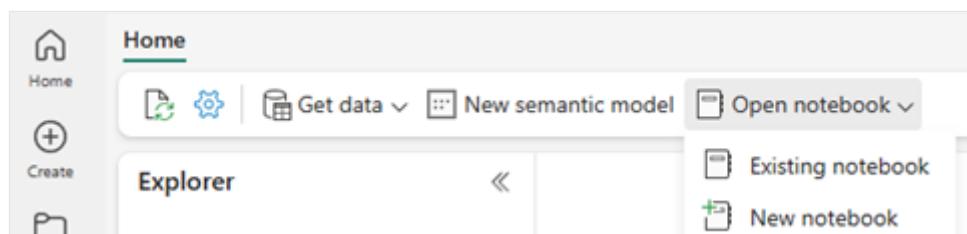
For an overview of shortcuts, see [OneLake shortcuts](#).

Access shortcuts as folders in an Apache Spark notebook

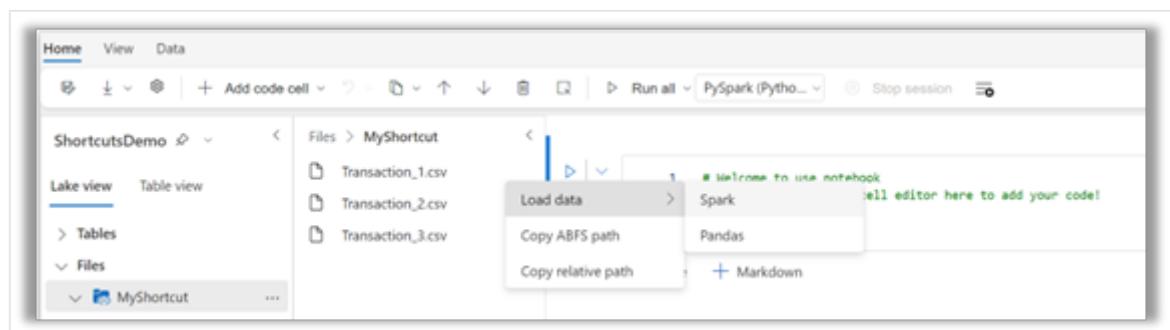
Shortcuts appear as folders in OneLake, and Apache Spark can read from them just like any other folder in OneLake.

To access a shortcut as a folder:

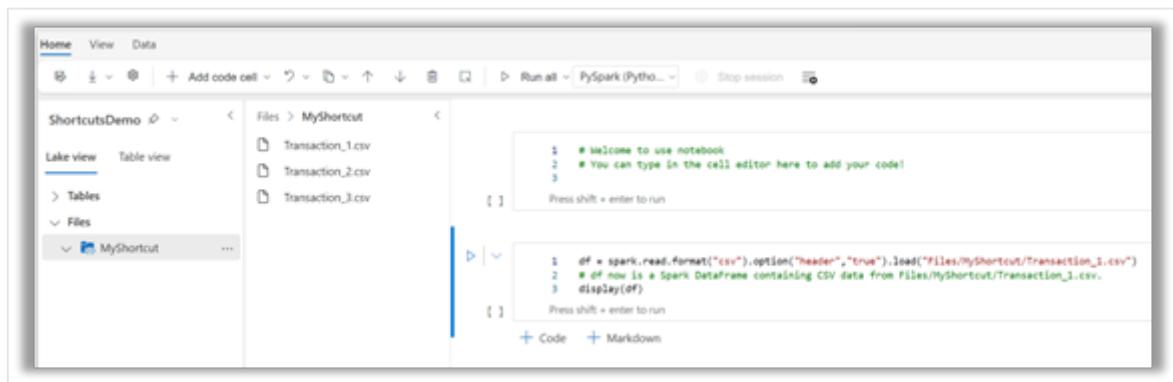
1. From a lakehouse containing shortcuts, select **Open notebook** and then select **New notebook**.



2. Select a shortcut and right-click on a file from the shortcut.
3. In the right-click menu, select **Load data** and then select **Spark**.



4. Run the automatically generated code cell.

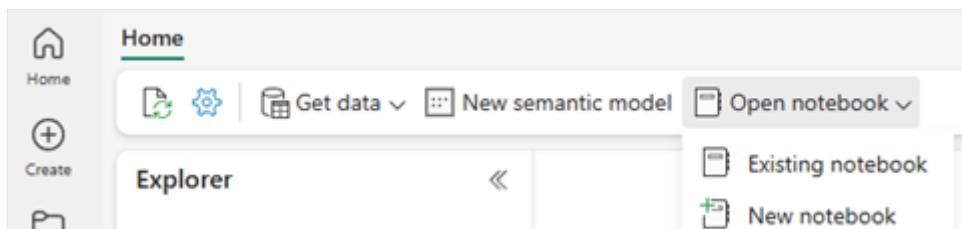


Access shortcuts as tables in a Spark notebook

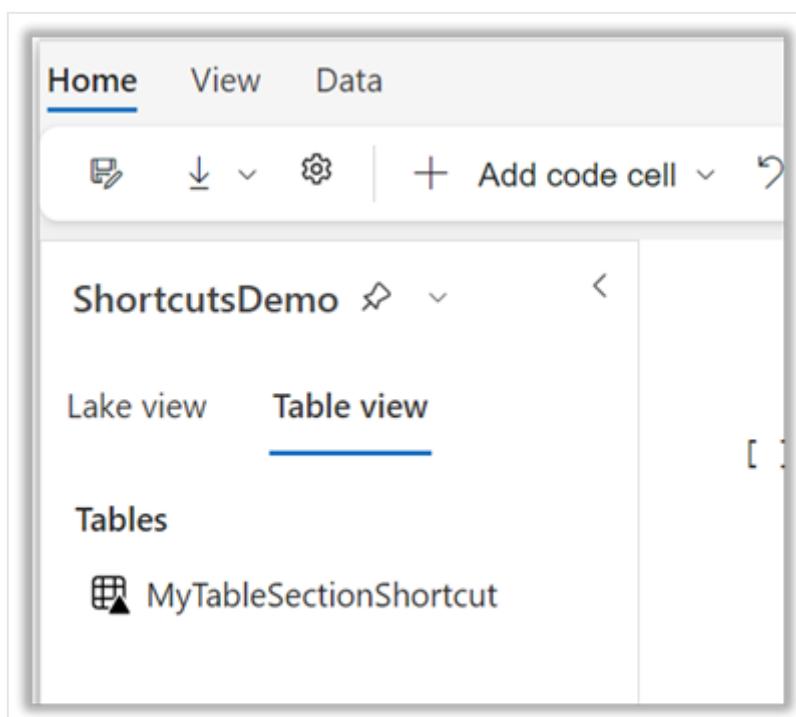
Microsoft Fabric automatically recognizes shortcuts in the **Tables** section of the lakehouse that have data in the Delta\Parquet format as tables. You can reference these tables directly from a Spark notebook.

To access a shortcut as a table:

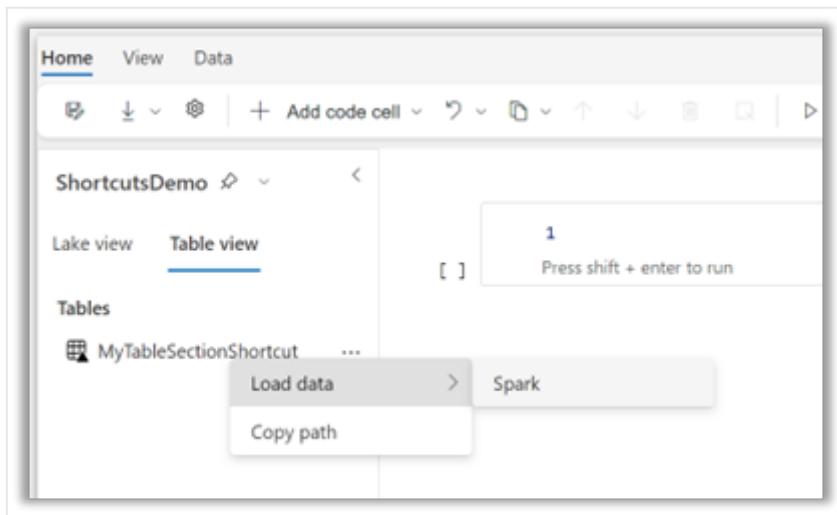
1. From a Lakehouse containing shortcuts, select **Open notebook** and then select **New notebook**.



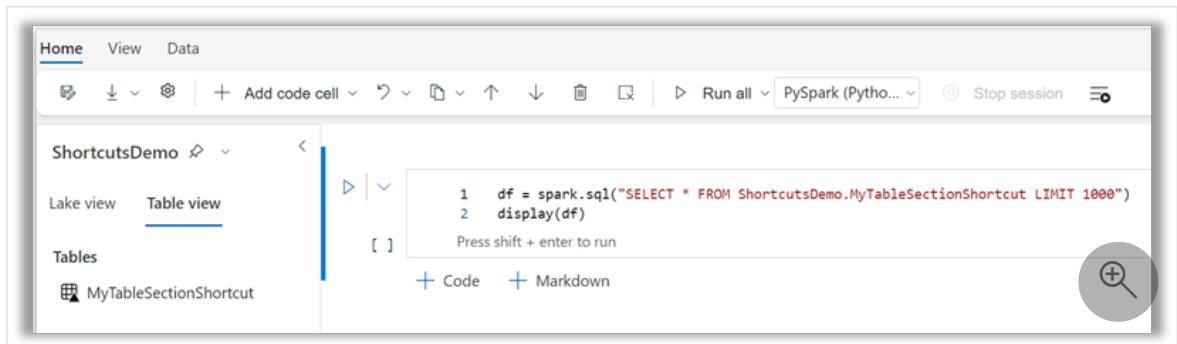
2. Select the **Table view** in the notebook.



3. Right-click on the table, then select Load data and Spark.



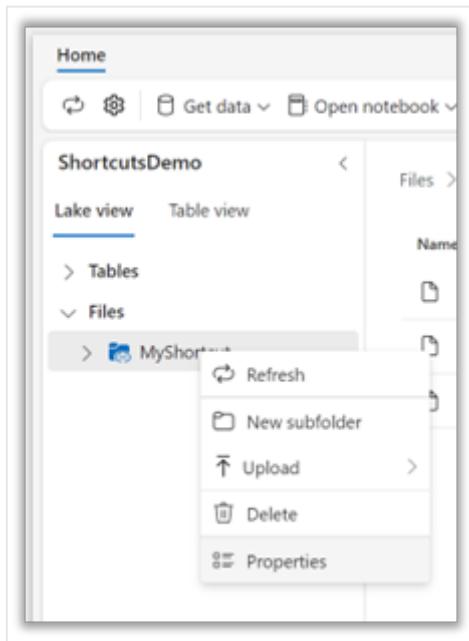
4. Run the automatically generated code cell.



Access the HTTPS and ABFS paths of a shortcut

You can also access shortcuts through the Azure Blob Filesystem (ABFS) driver or REST endpoint directly. Copy these paths from the lakehouse.

1. Open a Lakehouse containing shortcuts.
2. Right-click on a shortcut and select Properties.



3. Select the copy icon next to the ABFS path or URL in the Properties screen.

Related content

- [OneLake access and APIs](#)

Feedback

Was this page helpful?

Yes

No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Assign variables to shortcuts (preview)

06/13/2025

Fabric lifecycle management tools allow for the simple collaboration and continuous development of analytical solutions across multiple environments like testing and production. To learn more about these tools and processes see: [Introduction to CI/CD in Microsoft Fabric](#)

When deploying solutions across environments, you may want to configure properties that are unique to each environment so your testing environment points to test data and your production environment points to production data. Workspace variables make this possible.

You can use workspace variables within individual shortcut properties. This allows you to have unique values for properties like connection ID or target location for each environment.

Workspace variables and variable values sets can be defined within a variable library. See: [Learn how to use Variable libraries](#).

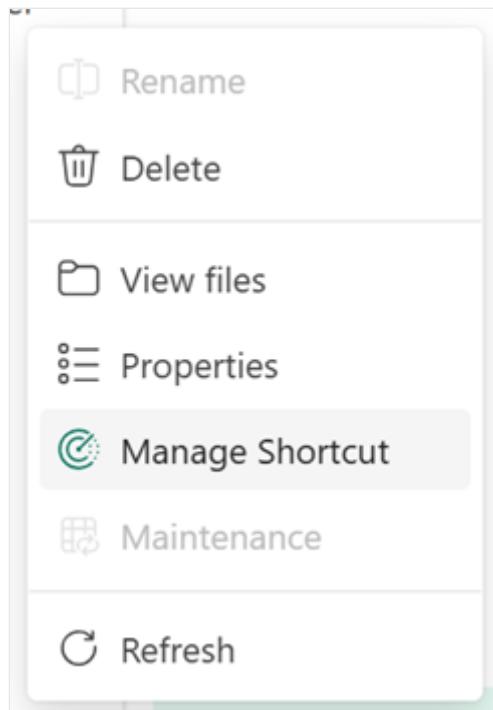
Once a variable is defined within a variable library, it can be assigned to a shortcut property using the manage shortcuts UX.

 **Important**

This feature is in [preview](#).

Assign a variable through the UX

1. Open a lakehouse and select an existing shortcut
2. Right click on the shortcut and choose **Manage Shortcut**



3. Select **Edit variables** and choose the desired property to assign the variable to.

The screenshot shows the "Manage shortcut" dialog box. At the top, there are two buttons: "Edit properties" and "Edit variables". The "Edit variables" button is highlighted with a green border. Below these buttons, there are two sections: "Name" (containing "DIM_Customer") and "Data type" (containing "Delta"). To the right of these sections, a dropdown menu is open, listing three options: "Target connection", "Target location", and "Target subpath". The "Target connection" option is highlighted with a green border.

4. Assign a variable from the variable library

5. Once the variable is assigned, the variable name and variable value appear below the shortcut property



Target connection

S3_testnm1



Variable ^

Name

\$(/**/Variables_2/MyConnectionID)

Value

788c042b-c2a3-4cbb-8855-bfdadd51c7

ⓘ Note

Only variables of type string are supported. Selecting a variable of any other type results in an error.

ⓘ Note

Assignment of workspace variables through the shortcuts REST API is not currently supported.

OneLake Shortcuts

Service: Core

API Version: v1

Operations

 [Expand table](#)

| | |
|---|--|
| Create Shortcut | Creates a new shortcut or updates an existing shortcut. |
| Creates Shortcuts In Bulk | Creates bulk shortcuts. |
| Delete Shortcut | Deletes the shortcut but does not delete the destination storage folder. |
| Get Shortcut | Returns shortcut properties. |
| List Shortcuts | Returns a list of shortcuts for the item, including all the subfolders exhaustively. |
| Reset Shortcut Cache | Deletes any cached files that were stored while reading from shortcuts. |

Connecting to Microsoft OneLake

09/19/2025

Microsoft OneLake provides open access to all of your Fabric items through existing Azure Data Lake Storage (ADLS) and Blob APIs and SDKs. You can access your data in OneLake through any API, SDK, or tool compatible with ADLS or Azure Blob Storage just by using a OneLake URI instead. You can upload data to a lakehouse through Azure Storage Explorer, or read a delta table through a shortcut from Azure Databricks.

As OneLake is software as a service (SaaS), some operations, such as managing permissions or updating items, must be done through Fabric experiences, and can't be done via ADLS APIs. For a full list of changes to these APIs, see [OneLake API parity](#).

URI syntax

Because OneLake exists across your entire Microsoft Fabric tenant, you can refer to anything in your tenant by its workspace, item, and path:

HTTP

```
https://onelake.dfs.fabric.microsoft.com/<workspace>/<item>.<itemtype>/<path>/<fileName>
```

⚠ Note

Because you can reuse item names across multiple item types, you must specify the item type in the extension. For example, `.lakehouse` for a lakehouse and `.warehouse` for a warehouse.

OneLake also supports referencing workspaces and items with globally unique identifiers (GUIDs). OneLake assigns GUIDs and GUIDs don't change, even if the workspace or item name changes. You can find the associated GUID for your workspace or item in the URL on the Fabric portal. You must use GUIDs for both the workspace and the item, and don't need the item type.

HTTP

```
https://onelake.dfs.fabric.microsoft.com/<workspaceGUID>/<itemGUID>/<path>/<fileName>
```

When adopting a tool for use over OneLake instead of ADLS, use the following mapping:

- The account name is always `onelake`.
- The container name is your workspace name.
- The data path starts at the item. For example: `/mylakehouse.lakehouse/Files/`.

OneLake also supports the [Azure Blob Filesystem driver](#) (ABFS) for more compatibility with ADLS and Azure Blob Storage. The ABFS driver uses its own scheme identifier `abfs` and a different URI format to address files and directories in ADLS accounts. To use this URI format over OneLake, swap workspace for filesystem and include the item and item type.

HTTP

```
abfs[s]://<workspace>@onelake.dfs.fabric.microsoft.com/<item>.  
<itemtype>/<path>/<fileName>
```

The abfs driver URI doesn't allow special characters, such as spaces, in the workspace name. In these cases, you can reference workspaces and items with the globally unique identifiers (GUIDs) as described earlier in this section.

Authorization

You can authenticate OneLake APIs using Microsoft Entra ID by passing through an authorization header. If a tool supports logging into your Azure account to enable token passthrough, you can select any subscription. OneLake only requires your user token and doesn't care about your Azure subscription.

When calling OneLake via DFS APIs directly, you can authenticate with a bearer token for your Microsoft Entra account. To learn more about requesting and managing bearer tokens for your organization, check out the [Microsoft Authentication Library](#).

For quick, ad-hoc testing of OneLake using direct API calls, here's a simple example using PowerShell to sign in to your Azure account, retrieve a storage-scoped token, and copy it to your clipboard for easy use elsewhere. For more information about retrieving access tokens using PowerShell, see [Get-AzAccessToken](#).

(!) Note

OneLake only supports tokens in the `Storage` audience. In the following example, we set the audience through the `ResourceTypeName` parameter.

PowerShell

```
Connect-AzAccount  
$testToken = Get-AzAccessToken -AsSecureString -ResourceType Name Storage  
# Retrieved token is of string type which you can validate with the  
"$testToken.Token.GetTypeCode()" command.  
$testToken.Token | Set-Clipboard
```

Data residency

If you use the global endpoint ('<https://onelake.dfs.fabric.microsoft.com>') to query data in a region different than your workspace's region, there's a possibility that data could leave your region during the endpoint resolution process. If you're concerned about data residency, using the correct regional endpoint for your workspace ensures your data stays within its current region and doesn't cross any regional boundaries. You can discover the correct regional endpoint by checking the region of the capacity that the workspace is attached to.

OneLake regional endpoints all follow the same format: <https://<region>-onelake.dfs.fabric.microsoft.com>. For example, a workspace attached to a capacity in the West US region would be accessible through the regional endpoint <https://westus-onelake.dfs.fabric.microsoft.com>.

Common issues

If a tool or package compatible with ADLS isn't working over OneLake, the most common issue is URL validation. As OneLake uses a different endpoint (dfs.fabric.microsoft.com) than ADLS (dfs.core.windows.net), some tools don't recognize the OneLake endpoint and block it. Some tools allow you to use custom endpoints (such as PowerShell). Otherwise, it's often a simple fix to add OneLake's endpoint as a supported endpoint. If you find a URL validation issue or have any other issues connecting to OneLake, [let us know ↗](#).

Resources

OneLake is accessible through the same APIs and SDKs as ADLS. To learn more about using ADLS APIs, please see the following pages:

- [ADLS Gen2 API Reference](#)
- ADLS Gen2 Filesystem SDKs
 - [.NET](#)
 - [Python](#)
 - [Java](#)

Samples

Create file

 Expand table

Request PUT `https://onelake.dfs.fabric.microsoft.com/{workspace}/{item}.`
`{itemtype}/Files/sample?resource=file`

Headers Authorization: Bearer <userAADToken>

Response ResponseCode: 201 Created

Headers:

`x-ms-version : 2021-06-08`
`x-ms-request-id : 272526c7-0995-4cc4-b04a-8ea3477bc67b`
`x-ms-content-crc64 : OAJ6r0dQWP0=`
`x-ms-request-server-encrypted : true`
`ETag : 0x8DA58EE365`

Body:

Related content

- [OneLake parity and integration](#)
- [Connect to OneLake with Python](#)
- [OneLake integration with Azure Synapse Analytics](#)

OneLake and Azure Data Lake Storage (ADLS) API parity

Article • 02/14/2025

OneLake supports the same APIs as Azure Data Lake Storage (ADLS) and Azure Blob Storage, enabling users to read, write, and manage their data in OneLake with the tools they already use today. Because OneLake is a managed, logical data lake, some features are managed differently than in Azure Storage, and not all behaviors are supported over OneLake. This page details these differences, including OneLake managed folders, API differences, and open source compatibility.

Managed OneLake folders

The workspaces and data items in your Fabric tenant define the structure of OneLake. Managing workspaces and items is done through Fabric experiences - OneLake doesn't support creating, updating, or deleting workspaces or items through the ADLS APIs. OneLake only allows HEAD calls at the workspace (container) level and tenant (account) level, as you must make changes to the tenant and workspaces in the Fabric administration portal.

OneLake also enforces a folder structure for Fabric items, protecting items and their managed subfolders from creation, deletion, or renaming through ADLS and Blob APIs. Fabric-managed folders include the top-level folder in an item (for example, `/MyLakehouse.lakehouse`) and the first level of folders within it (for example, `/MyLakehouse.lakehouse/Files` and `/MyLakehouse.lakehouse/Tables`).

You can perform CRUD operations on any folder or file created within these managed folders, and perform read-only operations on workspace and item folders.

Unsupported request headers and parameters

Even in user-created files and folders, OneLake restricts some Fabric management operations through ADLS APIs. You must use Fabric experiences to update permissions or edit items and workspaces, and Fabric manages other options such as access tiers.

OneLake accepts almost all of the same headers as Storage, ignoring only some headers that relate to unpermitted actions on OneLake. Since these headers don't alter the behavior of the entire call, OneLake ignores the banned headers, returns them in a new 'x-ms-rejected-headers' response header, and permits the rest of the call. For example,

OneLake ignores the 'x-ms-owner' parameter in a PUT call since Fabric and OneLake don't have the same concept of owning users as Azure Storage.

OneLake rejects requests containing unallowed query parameters since query parameters change the behavior of the entire call. For example, UPDATE calls with the 'setAccessControl' parameter are blocked since OneLake never supports setting access control via Azure Storage APIs.

OneLake doesn't allow the following behaviors and their associated request headers and URI parameters:

- Set access control
 - URI Parameter:
 - action: setAccessControl (Request rejected)
 - action: setAccessControlRecursive (Request rejected)
 - Request headers:
 - x-ms-owner (Header ignored)
 - x-ms-group (Header ignored)
 - x-ms-permissions (Header ignored)
 - x-ms-group (Header ignored)
 - x-ms-acls (Header ignored)
- Set encryption scope
 - Request headers:
 - x-ms-encryption-key (Header ignored)
 - x-ms-encryption-key (Header ignored)
 - x-ms-encryption-algorithm:AES256 (Header ignored)
- Set access tier
 - Request headers:
 - x-ms-access-tier (Header ignored)

Response header differences

Since OneLake uses a different permission model than ADLS, response headers related to permissions are handled differently:

- 'x-ms-owner' and 'x-ms-group' always returns '\$superuser' as OneLake doesn't have owning users or groups
- 'x-ms-permissions' always returns '-----' as OneLake doesn't have owning users, groups, or public access permissions
- 'x-ms-acl' returns the Fabric permissions for the calling user converted to a POSIX access control list (ACL), in the form 'rwx'

Open Source Integration

Since OneLake supports the same APIs as ADLS and Blob Storage, many open source libraries and packages compatible with ADLS and Blob Storage work seamlessly with OneLake (for example, [Azure Storage Explorer](#)). Other libraries may require small updates to accommodate OneLake endpoints or other compatibility issues. The following libraries are confirmed to be compatible with OneLake due to recent changes. This list isn't exhaustive:

- [Delta-RS](#)
- [Rust Object Store](#)

Examples

List items within a workspace (ADLS)

HTTP

```
GET https://onelake.dfs.fabric.microsoft.com/myWorkspace?  
resource=filesystem&recursive=false
```

List items within a workspace (Blob)

HTTP

```
GET https://onelake.blob.fabric.microsoft.com/myWorkspace?  
restype=container&comp=list&delimiter=%2F
```

Create a folder within a lakehouse (ADLS)

HTTP

```
PUT  
https://onelake.dfs.fabric.microsoft.com/myWorkspace/myLakehouse.Lakehouse/F  
iles/newFolder/?resource=directory
```

Get blob properties (Blob)

HTTP

HEAD

<https://onelake.blob.fabric.microsoft.com/myWorkspace/myLakehouse.Lakehouse/Files/file.txt>

Related content

- [Connect to OneLake using Python](#)
- [Use Azure Storage Explorer to manage OneLake](#)

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Use Python to manage files and folders in Microsoft OneLake

Article • 11/21/2023

This article shows how you can use the Azure Storage Python SDK to manage files and directories in OneLake. This walkthrough covers the same content as [Use Python to manage directories and files in ADLS Gen2](#) and highlights the differences when connecting to OneLake.

Prerequisites

Before starting your project, make sure you have the following prerequisites:

- A workspace in your Fabric tenant with Contributor permissions.
- A lakehouse in the workspace. Optionally, have data preloaded to read using Python.

Set up your project

From your project directory, install packages for the Azure Data Lake Storage and Azure Identity client libraries. OneLake supports the same SDKs as Azure Data Lake Storage (ADLS) Gen2 and supports Microsoft Entra ID authentication, which is provided by the `azure-identity` package.

Console

```
pip install azure-storage-file-datalake azure-identity
```

Next, add the necessary import statements to your code file:

Python

```
import os
from azure.storage.filedatalake import (
    DataLakeServiceClient,
    DataLakeDirectoryClient,
    FileSystemClient
)
from azure.identity import DefaultAzureCredential
```

Authorize access to OneLake

The following example creates a service client connected to OneLake that you can use to create filesystem clients for other operations. To authenticate to OneLake, this example uses the DefaultAzureCredential to automatically detect credentials and obtain the correct authentication token. Common methods of providing credentials for the Azure SDK include using the 'az login' command in the Azure Command Line Interface or the 'Connect-AzAccount' cmdlet from Azure PowerShell.

Python

```
def get_service_client_token_credential(self, account_name) ->
    DataLakeServiceClient:
    account_url = f"https://{{account_name}}.dfs.fabric.microsoft.com"
    token_credential = DefaultAzureCredential()

    service_client = DataLakeServiceClient(account_url,
    credential=token_credential)

    return service_client
```

To learn more about using DefaultAzureCredential to authorize access to data, see [Overview: Authenticate Python apps to Azure using the Azure SDK](#).

Working with directories

To work with a directory in OneLake, create a filesystem client and directory client. You can use this directory client to perform various operations, including renaming, moving, or listing paths (as seen in the following example). You can also create a directory client when creating a directory, using the [FileSystemClient.create_directory](#) method.

Python

```
def create_file_system_client(self, service_client, file_system_name: str) :
    DataLakeServiceClient) -> FileSystemClient:
    file_system_client = service_client.get_file_system_client(file_system =
    file_system_name)
    return file_system_client

def create_directory_client(self, file_system_client : FileSystemClient,
path: str) -> DataLakeDirectoryClient: directory_client
    directory_client = file_system_client.GetDirectoryClient(path)
    return directory_client

def list_directory_contents(self, file_system_client: FileSystemClient,
directory_name: str):
```

```
paths = file_system_client.get_paths(path=directory_name)

for path in paths:
    print(path.name + '\n')
```

Upload a file

You can upload content to a new or existing file by using the [DataLakeFileClient.upload_data](#) method.

Python

```
def upload_file_to_directory(self, directory_client:
    DataLakeDirectoryClient, local_path: str, file_name: str):
    file_client = directory_client.get_file_client(file_name)

    with open(file=os.path.join(local_path, file_name), mode="rb") as data:
        file_client.upload_data(data, overwrite=True)
```

Sample

The following code sample lists the directory contents of any folder in OneLake.

Python

```
#Install the correct packages first in the same folder as this file.
#pip install azure-storage-file-datalake azure-identity

from azure.storage.filedatalake import (
    DataLakeServiceClient,
    DataLakeDirectoryClient,
    FileSystemClient
)
from azure.identity import DefaultAzureCredential

# Set your account, workspace, and item path here
ACCOUNT_NAME = "onelake"
WORKSPACE_NAME = "<myWorkspace>"
DATA_PATH = "<myLakehouse>.Lakehouse/Files/<path>"

def main():
    #Create a service client using the default Azure credential

    account_url = f"https://{{ACCOUNT_NAME}}.dfs.fabric.microsoft.com"
    token_credential = DefaultAzureCredential()
    service_client = DataLakeServiceClient(account_url,
    credential=token_credential)
```

```
#Create a file system client for the workspace
file_system_client =
service_client.get_file_system_client(WORKSPACE_NAME)

#List a directory within the filesystem
paths = file_system_client.get_paths(path=DATA_PATH)

for path in paths:
    print(path.name + '\n')

if __name__ == "__main__":
    main()
```

To run this sample, save the preceding code into a file `listOneLakeDirectory.py` and run the following command in the same directory. Remember to replace the workspace and path with your own values in the example.

terminal

```
python listOneLakeDirectory.py
```

Learn more

- [Use Python to manage ADLS Gen2](#)
- [OneLake parity and integration](#)
- [Sync OneLake with your Windows File Explorer](#)

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Use Blob and ADLS APIs to mirror data into OneLake

10/10/2025

If your application uses [Azure Data Lake Storage \(ADLS\)](#) or [Blob Storage APIs](#) and needs to connect to OneLake, you can continue using the existing APIs.

We demonstrate how Blob and ADLS APIs are used with OneLake through a real-world mirroring example and share developer insights from OneLake. We explore when and why you might choose one API over another, and how to get the most out of each. All the patterns we cover apply to Azure Storage storage as well.

In this scenario, we cover:

- What is [open mirroring](#)
- How to use the .NET [Azure Blob Storage](#) and [Distributed File System \(DFS\)](#) clients to write data into the open mirror landing zone.
- How to combine the Blob Storage and DFS clients for uploading data and managing folders in OneLake, especially when performance matters
- How to handle scenarios that crop up with block blobs when writing parquet data to blob storage from .NET
- How to test everything locally using the [Azurite emulator](#). (Yes—code you write against OneLake works with the storage emulator too!)

Streaming parquet into OneLake with Blob APIs

In this section, we demonstrate how to efficiently stream parquet data into OneLake, particularly the open mirroring landing zone. Open mirroring is a powerful way to bring data from proprietary systems, where shortcuts [shortcuts](#) can't be used, into Microsoft Fabric. It handles the heavy lifting, converting raw data into [Delta Lake format](#), managing [upserts](#), [delete vectors](#), [optimize](#), [vacuum](#), and more. All you need to do is upload your data into the landing zone, include a row marker, and mirroring takes it from there.

It's common that teams write custom code to extract data from proprietary systems and output it in an open format. While open mirroring ingests both CSV and Parquet into Delta tables, if you're already writing code, you might as well go with Parquet, it's more efficient to upload and process.

[Parquet](#) is a storage [file format](#) designed for analytics. Delta, on the other hand, is a table protocol built on top of Parquet. It adds transactional guarantees, schema enforcement, and support for updates and deletes. When you upload Parquet files to the open mirroring landing

zone, those files are ingested into Delta tables—bringing ACID semantics and query performance optimizations without requiring you to manage those complexities yourself. The row marker indicates how each record should be merged into the table, which enables the mirroring process to know when to insert, update, or delete rows.

Let's walk through a concrete (but fictional) example.

Imagine you've got a .NET application that pulls UK house price data from an on-premises Inland Revenue system. (Just to be clear, this is a made-up scenario. The UK Inland Revenue isn't doing this to my knowledge, but the dataset is [publicly available](#) and makes for a good example.) This app runs monthly as an Azure Function, and to keep costs low, it needs to be fast and avoid using local disk. So, the goal is to stream the data directly from the source into the open mirroring landing zone in Parquet format. The Price Paid dataset from the UK Inland Revenue includes a [RecordStatus](#) field, which must be mapped to the row marker required by the open mirroring format.

This scenario aligns well with the Blob Storage API. It supports streaming writes, allowing you to push data directly into OneLake without staging it locally. That makes it simple, efficient, and cost-effective, especially for serverless workloads like Azure Functions. It's also fast: the Blob API supports parallel block uploads, which can significantly boost throughput when writing large files—and isn't supported when using the DFS endpoint.

For this, we're using the open source [Parquet.NET library](#), a fully managed .NET assembly that makes it easy to write Parquet data on the fly. Also, working with an object store like Blob Storage introduces a few nuances, especially around streaming and buffering, which gives us an opportunity to explore some nuances when working with blob storage.

This approach keeps your Azure Function lightweight, fast, and cost-efficient, no local disk, no staging, just stream, serialize, and upload.

Open mirroring landing zone summary

The [open mirroring landing zone](#) acts like an inbox for your mirrored tables, add files then Fabric takes care of the ingestion. But behind that simplicity is a clear protocol your application needs to follow to ensure data is correctly discovered and processed.

Folder structure

Each mirrored table has a dedicated path in OneLake:

```
<workspace>/mirrored-database/Files/LandingZone/<table-name>/
```

This is where you'll write both metadata and data files. You don't need to explicitly create folders, just write blobs with the appropriate prefix, and the structure is inferred.

Step 1: Declare the table keys

Before writing any data, you must create a `_metadata.json` file in the table's landing zone folder. This file defines the key columns used for upserts and deletes:

C#

```
public async Task CreateTableAsync(OpenMirroredTableId table, params string[] keyColumns)
{
    await using var metadataFile = await OpenWriteAsync(table, "_metadata.json");
    var json = new { keyColumns };
    await JsonSerializer.SerializeAsync(metadataFile, json);
}
```

This metadata file tells Fabric how to uniquely identify rows. Without it, Fabric won't ingest your data.

Step 2: Create data files with the correct name

Once the metadata is in place, you can start writing data. Files must be named sequentially using zero-padded numbers like `00000000000000000001.parquet`, `00000000000000000002.parquet`, etc. This ensures deterministic ordering and avoids collisions.

List APIs return blobs alphabetically, so our logic can quickly find the next sequence number by processing the landing zone folder with a flat listing. Open mirroring moves processed files to folders prefixed with `_` which are sorted below numeric values. Exiting the loop after seeing all parquet files improves performance when you're using the [Azure Storage List Blob API](#) – which would enumerate blobs in sub folders as it matches the prefix. If you're using the [ADLS Path List API](#), you can choose to perform a recursive list, which allows control over whether or not to list the contents of sub folders – this is a clear advantage of the hierarchical namespace.

The logic to determine the next file name looks like this:

C#

```
public async Task<MirrorDataFile> CreateNextTableDataFileAsync(OpenMirroredTableId table)
{
    var (containerClient, path) = GetTableLocation(table);
    var listBlobs = containerClient.GetBlobsAsync(prefix: path);
    var tableFound = false;
```

```

        BlobItem? lastDataFile = null;
        // Parquet files will be first in the folder because other files and folders
        all start with an underscore.
        // So we can just take the last Parquet file to get our sequence number.
        await foreach (var blob in listBlobs)
        {
            tableFound = true;
            if (blob.Name.EndsWith(".parquet") && blob.Properties.ContentLength > 0)
            {
                lastDataFile = blob;
            }
            else
            {
                break;
            }
        }
        if (!tableFound)
        {
            throw new ArgumentException($"Table not found.", nameof(table));
        }

        long lastFileNumber = 0;

        if (lastDataFile is not null)
        {
            var dataFileName = Path.GetFileName(lastDataFile.Name);
            lastFileNumber = long.Parse(dataFileName.Split('.')[0]);
        }

        var stream = await OpenWriteAsync(table, $"{++lastFileNumber:D20}.parquet");
        return new MirrorDataFile(stream)
        {
            FileSequenceNumber = lastFileNumber
        };
    }
}

```

You might be wondering what the `MirrorDataFile` class is for, we'll come back to that shortly when we cover how to work reliably with block blobs.

Step 3: Upload the contents of the file

The actual write is handled by opening a stream to the blob:

C#

```

private async Task<Stream> OpenWriteAsync(OpenMirroredTableId table, string
fileName)
{
    var (containerClient, path) = GetTableLocation(table);
    path += fileName;
    var blobClient = containerClient.GetBlobClient(path);

```

```
    var stream = await blobClient.OpenWriteAsync(true);
    return stream;
}
```

➊ Note

Mirrored databases support schemas, so the landing zone can contain an [optional schema folder](#) to denote that the table is within a schema. Also, in OneLake, Fabric [workspaces](#) are mapped to storage [Containers](#) and [ADLS Filesystems](#).

C#

```
public record OpenMirroredTableId(string WorkspaceName, string
MirroredDatabaseName, string TableName)
{
    public string? Schema { get; init; } = null;

    public string GetTableName() => Schema == null
        ? $"{MirroredDatabaseName}/Files/LandingZone/{TableName}/"
        : $""
{MirroredDatabaseName}/Files/LandingZone/{Schema}.schema/{TableName}/";

private (BlobContainerClient ContainerClient, string TablePath)
GetTableLocation(OpenMirroredTableId table)
{
    var containerClient =
blobServiceClient.GetBlobContainerClient(table.WorkspaceName);
    var path = table.GetTableName();

    return (containerClient, path);
}
```

The business logic that reads Price Paid data from the Inland Revenue system converts each record into Parquet format as it is streamed from the source. Each row is written directly into a Parquet file, which is simultaneously streamed byte-by-byte into Azure Storage using the stream returned by `CreateNextTableDataFileAsync`. This approach avoids local staging and supports efficient, serverless ingestion.

Here's how the code works:

C#

```
public async Task SeedMirrorAsync(OpenMirroredTableId tableId, CancellationToken
cancellationToken = default)
{
    await openMirroringWriter.CreateTableAsync(tableId,
```

```

PricePaidMirroredDataFormat.KeyColumns);
    var data = pricePaidDataReader.ReadCompleteData(cancellationToken);
    await using var mirrorDataFile = await
openMirror.CreateNextTableDataFileAsync(tableId);
    await mirrorDataFile.WriteData(async stream => await data.WriteAsync(stream,
Settings.RowsPerRowGroup, cancellationToken));
}

```

The `WriteAsync` method serializes the data into Parquet format, row group by row group:

C#

```

public static async Task WriteAsync(this IAsyncEnumerable<PricePaid> data, Stream
resultStream, int rowsPerRowGroup = 10000, CancellationToken cancellationToken =
default)
{
    await using var parquetWriter = await ParquetWriter.CreateAsync(
        PricePaidMirroredDataFormat.CreateSchema(),
        resultStream,
        cancellationToken: cancellationToken);

    await foreach (var chunk in data
        .Select(PricePaidMirroredDataFormat.Create)
        .ChunkAsync(rowsPerRowGroup, cancellationToken))
    {
        await ParquetSerializer.SerializeRowGroupAsync(parquetWriter, chunk,
        cancellationToken);
    }
}

```

Each `PricePaid` record is transformed into a `PricePaidMirroredDataFormat` object, which includes the required `_rowMarker_` field:

C#

```

public static PricePaidMirroredDataFormat Create(PricePaid pricePaid)
{
    var recordMarker = pricePaid.RecordStatus.Value switch
    {
        RecordStatus.Added => 0,
        RecordStatus.Changed => 1,
        RecordStatus.Deleted => 2,
        _ => throw new InvalidEnumArgumentException("Unexpected RecordStatus
value")
    };

    return new PricePaidMirroredDataFormat
    {
        TransactionId = pricePaid.TransactionId,
        Price = pricePaid.Price,
        ...
    };
}

```

```
    __rowMarker__ = recordMarker
};

}
```

This protocol is simple and deterministic. It avoids unnecessary API calls, works seamlessly with both batch and streaming pipelines, and integrates smoothly with serverless environments like Azure Functions. Uploading a blob to a prefix automatically creates the parent folders and remains compatible with the storage emulator—more on that in the testing section.

Step 4: Clean up after yourself

Once open mirroring has successfully processed your data, it moves the original files into special folders, `_ProcessedFiles` and `_FilesReadyToDelete`, and adds a `_FilesReadyToDelete.json` file. While Fabric will automatically delete these files after seven days, that retention window can lead to significant storage costs if you're mirroring large volumes of data.

To reduce costs, you can proactively delete these folders once you're confident the data has been ingested. This is a great use case for the ADLS API, which supports atomic directory deletion—far more efficient than enumerating and deleting individual blobs and updating the `_FilesReadyToDelete.json` file.

Here's how to do it:

```
C#  
  
public async Task CleanUpTableAsync(OpenMirroredTableId tableId)  
{  
    var (fileSystemClient, filePath) = GetTableLocation(tableId);  
    var foldersToDelete = new []  
    {  
        "_ProcessedFiles",  
        "_FilesReadyToDelete"  
    };  
    await Parallel.ForEachAsync(foldersToDelete, async (path, _) =>  
    {  
        var fullPath = $"{filePath}{path}/";  
        var directoryClient = fileSystemClient.GetDirectoryClient(fullPath);  
        if (await directoryClient.ExistsAsync())  
        {  
            await directoryClient.DeleteAsync();  
        }  
    });  
  
    private (DataLakeFileSystemClient FileSystemClient, string TablePath)  
    GetTableLocation(OpenMirroredTableId table)  
    {
```

```

    var containerClient =
dataLakeServiceClient.GetFileSystemClient(table.WorkspaceName);
    var path = $""
{table.MirroredDatabaseName}/Files/LandingZone/{table.TableName}/";
    return (containerClient, path);
}

```

Reliably working with block blobs

I chose this example because it opens the door to talk about some nuances with block blobs. These aren't OneLake specific, but because OneLake is built on Azure Storage, OneLake exposes these nuances too.

One such case: the .NET Storage SDK exposes an `OpenWrite` API that returns a Stream. Super handy. As shown in the example above, that stream fits nicely with the Parquet.NET APIs. It also makes testing a breeze—you can easily substitute the stream in unit tests without needing to build extra abstractions just for testability.

C#

```

[TestMethod]
public async Task When_Writing_Price_Paid_Data_to_Parquet()
{
    var row = new PricePaid
    {
        TransactionId = "{34222872-B554-4D2B-E063-4704A8C07853}",
        Price = 375000,
        DateOfTransfer = new DateTime(2004, 4, 27),
        Postcode = "SW13 0NP",
        PropertyType = PropertyType.Detached,
        IsNew = true,
        DurationType = DurationType.Freehold,
        PrimaryAddressableObjectName = "10A",
        SecondaryAddressableObjectName = string.Empty,
        Street = "THE TERRACE",
        Locality = string.Empty,
        TownCity = "LONDON",
        District = "RICHMOND UPON THAMES",
        County = "GREATER LONDON",
        CategoryType = CategoryType.AdditionalPricePaid,
        RecordStatus = RecordStatus.Added
    };

    using var memoryStream = new MemoryStream();
    await new[] { row }.ToAsyncEnumerable().WriteAsync(memoryStream);

    var readData = await
PricePaidMirroredDataFormat.Read(memoryStream).SingleAsync();
    Assert.Multiple(() =>

```

```

    {
        Assert.That(readData.TransactionId, Is.EqualTo(row.TransactionId));
        Assert.That(readData.Price, Is.EqualTo(row.Price));
        Assert.That(readData.DateOfTransfer, Is.EqualTo(row.DateOfTransfer));
        // more assertions
        Assert.That(readData.__rowMarker__, Is.EqualTo(0)); // RecordStatus.Added
    });
}

```

But here's the catch: Blob Storage isn't the same as a local file system.

Calling flush – commits the blocks

To understand why the `Flush()` call in Parquet.NET matters, we need to take a quick detour into how Parquet files are structured—and how that interacts with the Blob Storage block blob API.

Parquet file basics

Parquet is a columnar storage format designed for efficient analytics. A Parquet file is made up of:

- Row groups: These are the core building blocks. Each row group contains a chunk of rows, organized by column. Row groups are written sequentially and independently.
- Column chunks: Within each row group, data is stored column-by-column.
- Metadata: At the end of the file, Parquet writes a footer that includes schema and offset information for fast reads. In Parquet.NET, each time a row group is completed, the library calls `Flush()` on the output stream. This is where things get interesting when you're writing to Blob Storage.

Block blob model

Blob Storage, and therefore OneLake, uses a block blob model, which works like this:

- You upload blocks: Each block can be up to 4,000 MiB in size (default is 4 MiB). You can upload blocks in parallel to maximize throughput—and the .NET Blob client does this for you automatically, which is great. This is one reason to use the Blob client (not the DFS client) for uploads. The DFS client doesn't support parallel block uploads, which can be a performance bottleneck. That said, the DFS client can still read the resulting files just fine.
- You commit the blocks: Once all blocks are uploaded, you call `CommitBlockList()` to finalize the blob. You can upload up to 50,000 blocks per blob, which means—if you're using 4,000-MiB blocks—you could theoretically write a single 190.7-TiB file. (Not that I'd

recommend it.) If you'd like to learn more, read this article [Understanding block blobs, append blobs, and page blobs](#).

Here's the catch

When Parquet.NET calls `Flush()` after each row group, and you're writing to a stream backed by Blob Storage, that flush triggers a BlockList commit. As the file grows, each new row group causes the library to recommit all previously uploaded blocks.

Let's say you're writing a 1-GB file with 100MB row groups, using 4MB blocks. That gives you 250 blocks in total. On the first flush, you commit 25 blocks. On the second, 50. By the final flush, you're committing all 250 blocks. Add that up across all 10 row groups, and you've committed a total of 1,375 blocks—even though the final file only needs one commit.

This is inefficient, and it introduces two key problems:

- **Timeouts and retries.** Large BlockLists can lead to timeouts. Remember, Blob Storage is built on HTTP. It's reliable—but not perfect. A timeout might succeed on the server, but your client doesn't know that, so it retries. That retry can result in a 409 Conflict. Doing something expensive 250 times instead of once increases the chance of this happening. Fewer commits = fewer retries = fewer headaches.
- **Premature blob visibility.** One of the nice things about the Blob API is that the blocks don't become visible until you explicitly commit the BlockList. This aligns well for scenarios like Parquet, where the file isn't valid until the footer metadata is written. It means downstream processes won't accidentally pick up a half-written file. But here's the twist: because Parquet.NET calls `Flush()` after each row group, and that flush triggers a commit, the blob becomes visible before the file is complete. So even though the Blob API is designed to help you avoid this problem, the way Parquet.NET works with a Stream introduces an issue—unless you take steps to prevent it.

To bridge the gap between Parquet.NET and the nuances of Blob Storage, don't implement the `Flush()` call in the `BlobFile.BlobStream` implementation. That way, even though Parquet.NET calls `Flush` after each row group, the underlying stream doesn't flush to storage.

The storage clients (DFS and Blob) will create an empty file when calling `OpenWrite`. The open mirroring processor logs an error when encountering a 0-byte file, which is possible if the replicator process interleaves with file creation. To avoid this write a file to another location and move it to the correct path, which the ADLS APIs supports through a [rename operation](#), like so:

C#

```
public async Task<BlobFile> CreateFileAsync(string filePath)
{
    if (filePath is null)
```

```
{  
    throw new ArgumentNullException(nameof(filePath), "File path cannot be  
null.");  
}  
var containerClient =  
client.blobServiceClient.GetBlobContainerClient(containerName);  
var temporaryPath = $"{filePath}.temp";  
var blobClient = containerClient.GetBlobClient(Combine(temporaryPath));  
var blobStream = await blobClient.OpenWriteAsync(overwrite: true);  
return new BlobFile(blobStream, GetChildPath(temporaryPath),  
Combine(filePath)!);  
}
```

Calling dispose – commits the blocks and move the file

Here's another subtle but important behavior to be aware of: when you use the Azure Blob stream, calling `.Dispose()` (or letting it be disposed implicitly) will commit all uploaded blocks.

That's usually what you want—but not always.

Let's say your source system is streaming data using an `IAsyncEnumerable`, as it does in the example to illustrate the bug. If that source fails partway through, for example, the database connection times out or the network drops, you might only have a partially written Parquet file. But if the stream gets disposed (which it will, due to await using or a using block), those partial blocks get committed.

To avoid this, the example code explicitly commits once only when the entire operation completes successfully. That way, if the producer fails mid-stream, you're not left with a half-written file.

This is why the example returns a `BlobFile` object from `CreateNextTableDataFileAsync`, to encapsulate the stream to prevent the zero-byte file, and issues described with `Flush` and `Dispose` committing partial parquet files.

C#

```
public class BlobFile(Stream stream, IStoragePath temporaryFilePath, string  
finalFilePath) : IAsyncDisposable  
{  
    private readonly BlobStream stream = new(stream, temporaryFilePath,  
finalFilePath);  
  
    public async Task WriteData(Func<Stream, Task> writeOperation)  
    {  
        try  
        {  
            await writeOperation(stream);  
        }  
    }  
}
```

```
        catch (Exception)
    {
        stream.Failed();
        throw;
    }
}

public async ValueTask DisposeAsync()
{
    await stream.DisposeAsync();
}

private class BlobStream(Stream innerStream, IStoragePath temporaryFilePath,
string finalFilePath) : Stream
{
    private bool disposed = false;
    private bool success = true;

    public override bool CanRead => innerStream.CanRead;
    public override bool CanSeek => innerStream.CanSeek;
    public override bool CanWrite => innerStream.CanWrite;

    public override long Length => innerStream.Length;

    public override long Position
    {
        get => innerStream.Position;
        set => innerStream.Position = value;
    }

    public override void Flush()
    {
        // no-op
    }

    public override int Read(byte[] buffer, int offset, int count) =>
innerStream.Read(buffer, offset, count);

    public override long Seek(long offset, SeekOrigin origin) =>
innerStream.Seek(offset, origin);

    public override void SetLength(long value) =>
innerStream.SetLength(value);

    public override void Write(byte[] buffer, int offset, int count) =>
innerStream.Write(buffer, offset, count);

    public void Failed()
    {
        success = false;
    }

    public override async ValueTask DisposeAsync()
    {
        if (disposed)
```

```
        {
            return;
        }
        if (success)
        {
            await innerStream.DisposeAsync();
            await temporaryFilePath.RenameAsync(finalFilePath);
        }
        disposed = true;
    }
}
```

Writing tests with OneLake using the Azurite emulator

One of the great things about OneLake is that it's built on Azure Storage—which means you can test your integration code locally using the Azurite emulator. This makes it easy to write reliable tests, that emulate the behavior of OneLake / Azure Storage, without needing a live Fabric environment or cloud resources.

Azurite emulates the Blob Storage API, which is exactly what OneLake exposes. That means the same code you use in production can run unchanged in your test suite. You can spin up Azurite as a local process or container, point your `BlobServiceClient` at it, and go.

This is especially useful for unit and integration tests. You can:

- Validate that your `_metadata.json` is written correctly.
- Check that your file naming logic produces the expected sequence.
- Simulate partial writes resulting from calling `Flush` and `Dispose` on failure. This allows testing of the nuances described above and that the implementation handles them appropriately.
- Assert that your Parquet serialization round-trips cleanly.

Azurite doesn't support the ADFS APIs. This is why `BlobFile` above uses an `IStoragePath` interface to implement ADFS functionality using the blob APIs so they can work with the emulator under testing. Here's an example:

```
C#  
  
public async Task RenameAsync(string newPath)  
{  
    async Task RenameDirectoryAsync(DataLakeServiceClient dataLakeServiceClient)  
    {  
        var fileSystemClient =
```

```

dataLakeServiceClient.GetFileSystemClient(containerName);
    var directoryClient = fileSystemClient.GetDirectoryClient(path);
    await directoryClient.RenameAsync(newPath);
}

async Task CopyThenDeleteAsync(BlobServiceClient blobServiceClient)
{
    var sourceBlob =
blobServiceClient.GetBlobContainerClient(containerName).GetBlobClient(path);
    var destinationBlob =
blobServiceClient.GetBlobContainerClient(containerName).GetBlobClient(newPath);

    await destinationBlob.StartCopyFromUriAsync(sourceBlob.Uri);
    await sourceBlob.DeleteIfExistsAsync();
}

StorageOperation operation = new()
{
    WithFlatNamespace = CopyThenDeleteAsync,
    WithHierarchicalNamespace = RenameDirectoryAsync
};

await operation.Execute(client);
}

```

Testing corner cases: flush and dispose

Earlier, we talked about how Parquet.NET's use of `Flush()` and `Dispose()` can lead to premature or partial blob commits when writing to OneLake. These behaviors are subtle—but testable.

Here are a few tests to validate that the mirroring logic handles these scenarios correctly:

C#

```

[Test]
public async Task it_should_write_data_to_table()
{
    await setup.FabricPricePaidMirror.SeedMirrorAsync(setup.TableId);
    var mirroredData = await GetMirroredBlobItem();

    var mirroredDataClient =
setup.WorkspaceContainer.GetBlobClient(mirroredData!.Name);
    var mirroredDataContents = await mirroredDataClient.DownloadContentAsync();

    var readData = await
PricePaidMirroredDataFormat.Read(mirroredDataContents.Value.Content.ToStream()).Si
ngleAsync();
    Assert.Multiple(() =>
    {
        Assert.That(mirroredData, Is.Not.Null);
    });
}

```

```

        Assert.That(mirroredData!.Properties.ContentLength, Is.GreaterThan(0));
        Assert.That(readData.TransactionId,
Is.EqualTo(setup.PricePaidReader.TransactionId));
        // ... more assertions ...
    });
}

```

This test confirms that a successful write results in a valid, nonempty Parquet file that round-trips correctly.

Now for the failure cases:

C#

```

[Test]
public async Task it_should_not_commit_partially_written_data()
{
    long? lengthDuringWrite = null;
    setup.PricePaidReader.ActionBetweenRowGroups = async () =>
    {
        var mirroredData = await GetMirroredBlobItem();
        lengthDuringWrite = mirroredData!.Properties.ContentLength;
    };
    await setup.FabricPricePaidMirror.SeedMirrorAsync(setup.TableId);
    Assert.That(lengthDuringWrite, Is.EqualTo(0));
}

```

This test simulates a mid-stream exception and verifies that no partial data is visible while the write is in progress because of the no-op `Flush()` override.

And finally, the failure path:

C#

```

[Test]
public async Task
after_a_row_group_is_written_it_should_not_leave_a_partially_complete_blob()
{
    setup.PricePaidReader.ThrowsAfterFirstRowGroup = true;

    var previouslyMirroredFile = await GetMirroredBlobItem();

    var threw = true;
    try
    {
        await setup.FabricPricePaidMirror.SeedMirrorAsync(setup.TableId);
    }
    catch (Exception)
    {
        threw = true;
    }
}

```

```

    }

    var mirroredData = await GetMirroredBlobItem();
    var mirroredTemporaryData = await GetMirroredBlobTemporaryItem();

    Assert.Multiple(() =>
    {
        Assert.That(threw, Is.True);
        if (previouslyMirroredFile == null)
        {
            Assert.That(mirroredData, Is.Null);
        }
        else
        {
            Assert.That(mirroredData!.Name,
Is.EqualTo(previouslyMirroredFile.Name));
        }
        Assert.That(mirroredTemporaryData, Is.Not.Null);
        Assert.That(mirroredTemporaryData!.Properties.ContentLength,
Is.EqualTo(0));
    });
}

```

This test confirms that even if the stream is disposed due to an exception a new mirrored file isn't partially written.

These tests provide confidence that the mirroring logic is robust, even under failure conditions. They demonstrate how the emulator can be used to simulate real-world behavior without needing a full Fabric (or Azure) environment.

And just to prove compatibility, tweaking the test setup to point at a Fabric workspace, here's a table full of UK house price data.

```

C#

public class when_using_fabric
{
    public class in_success_cases : when_copying_to_mirror_successfully
    {
        [SetUp]
        public void UseFabric() => setup = TestSetup.UsingFabric();
    }

    public class in_failure_cases : when_copying_to_mirror_fails
    {
        [SetUp]
        public void UseFabric() => setup = TestSetup.UsingFabric();
    }
}

public static TestSetup UsingFabric()
{

```

```

        var blobServiceClient = new BlobServiceClient(new
Uri("https://onelake.blob.fabric.microsoft.com/"), new DefaultAzureCredential());
        var pricePaidReader = new TestPricePaidReader();
        var tableId = new OpenMirroredTableId($"TestWorkspace",
"HousePriceOpenMirror.MountedRelationalDatabase", "PricePaid");
        var workspaceContainer =
blobServiceClient.GetBlobContainerClient(tableId.WorkspaceName);
        var fabricPricePaidMirror = new FabricPricePaidMirror(new
FabricOpenMirroringWriter(blobServiceClient), pricePaidReader)
{
    Settings = new FabricPricePaidMirrorSettings { RowsPerRowGroup = 1 }
};

return new TestSetup
{
    BlobServiceClient = blobServiceClient,
    PricePaidReader = pricePaidReader,
    TableId = tableId,
    WorkspaceContainer = workspaceContainer,
    FabricPricePaidMirror = fabricPricePaidMirror
};
}

```

The screenshot shows the OneLake interface. On the left, the 'Explorer' sidebar displays 'Replication status' under 'HousePriceOpenMirror'. It lists 'Tables in OneLake' (dbo.PricePaid) and 'Uploaded files' (PricePaid). In the center, a table titled 'Showing 1000 of 1000 rows' is displayed with the following data:

| | Transactionid | Price | DateOfTransfer | Postcode | PropertyType | OldNew | Duration | PrimaryAddress |
|---|------------------|--------|----------------|----------|--------------|--------|----------|----------------|
| 1 | 842dbd41-225... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 2 | de97af0d-7aa2... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 3 | b26240a7-8f46... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 4 | b4b31c2c-f971... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 5 | b8e82770-3ca8... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 6 | f6c4547b-a60c... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 7 | 295e05dd-5f06... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 8 | bcd4eb78-29cf... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |
| 9 | 4d614c9e-8f88... | 100000 | 2023-01-01 | AB12 3CD | Detached | 1 | Freehold | 123 Main St |

Wrapping up

If you're building new pipelines on OneLake, especially for streaming or serverless workloads, the ADLS and Blob storage APIs work as expected. They're fast, flexible, and work seamlessly with open mirroring. By following the landing zone protocol and handling block blob and file system differences, you can build robust, testable integrations that work just as well in production as they do in your local emulator. And best of all—you don't need to rewrite your app or fight the filesystem. Just stream, serialize, and upload to OneLake.

Integrate OneLake with Azure Synapse Analytics

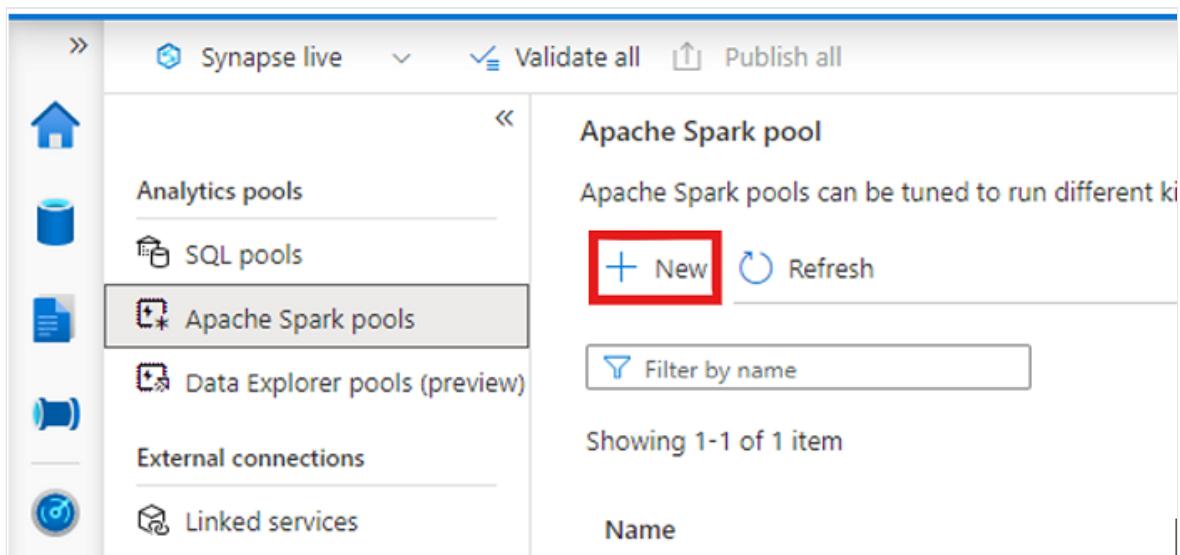
Article • 11/29/2023

Azure Synapse is a limitless analytics service that brings together enterprise data warehousing and Big Data analytics. This tutorial shows how to connect to OneLake using [Azure Synapse Analytics](#).

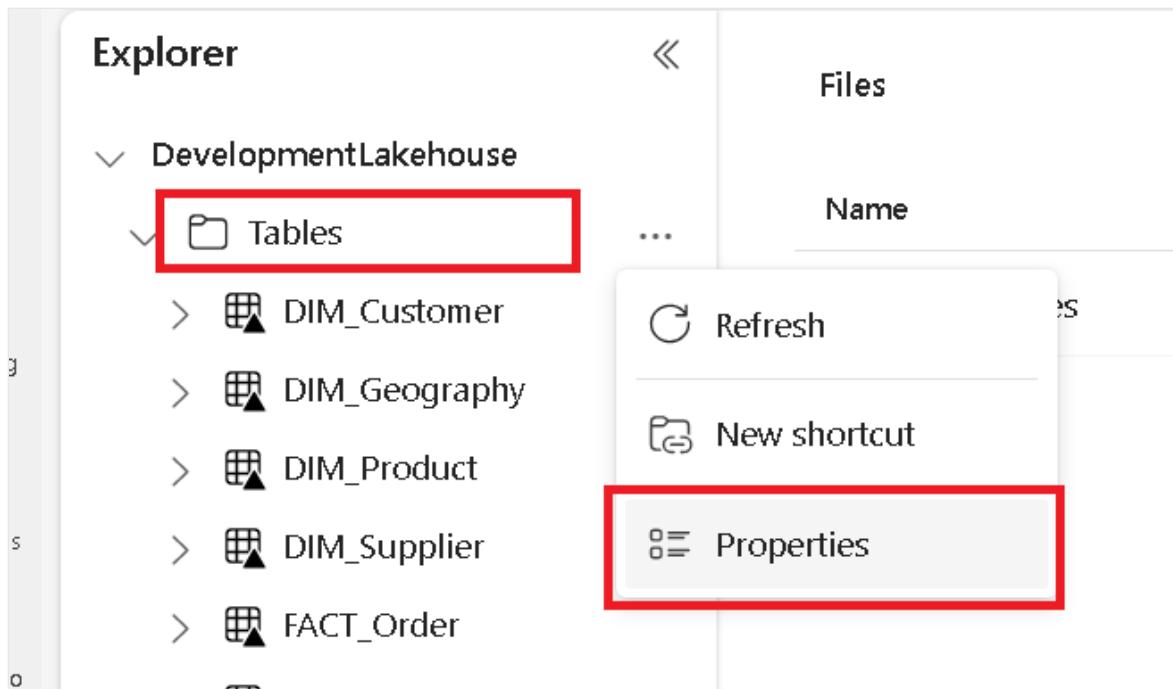
Write data from Synapse using Apache Spark

Follow these steps to use Apache Spark to write sample data to OneLake from Azure Synapse Analytics.

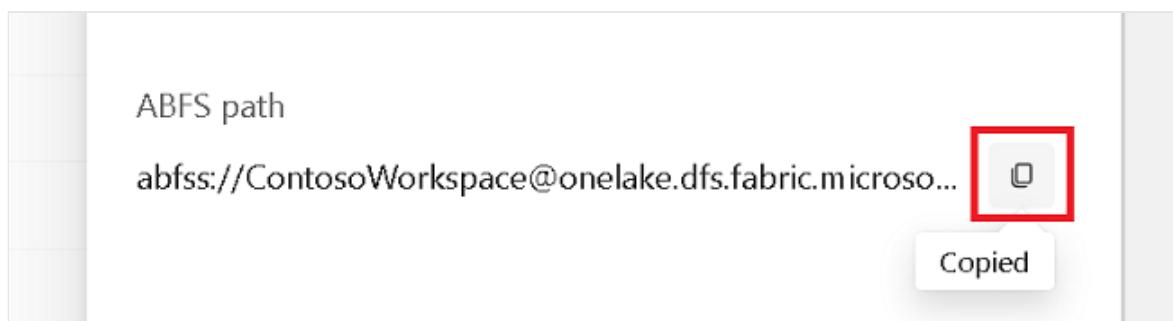
1. Open your Synapse workspace and [create an Apache Spark pool](#) with your preferred parameters.



2. Create a new Apache Spark notebook.
3. Open the notebook, set the language to **PySpark (Python)**, and connect it to your newly created Spark pool.
4. In a separate tab, navigate to your Microsoft Fabric lakehouse and find the top-level **Tables** folder.
5. Right-click on the **Tables** folder and select **Properties**.



6. Copy the ABFS path from the properties pane.



7. Back in the Azure Synapse notebook, in the first new code cell, provide the lakehouse path. This lakehouse is where your data is written later. Run the cell.

```
Python

# Replace the path below with the ABFS path to your lakehouse Tables
# folder.
oneLakePath =
'abfss://WorkSpaceName@onelake.dfs.fabric.microsoft.com/LakehouseName.lakehouse/Tables'
```

8. In a new code cell, load data from an Azure open dataset into a dataframe. This dataset is the one you load into your lakehouse. Run the cell.

```
Python

yellowTaxiDF =
spark.read.parquet('wasbs://nyctlc@azureopendatastorage.blob.core.windows.net/yellow/puYear=2018/puMonth=2/*.parquet')
display(yellowTaxiDF.limit(10))
```

9. In a new code cell, filter, transform, or prep your data. For this scenario, you can trim down your dataset for faster loading, join with other datasets, or filter down to specific results. Run the cell.

```
Python
```

```
filteredTaxiDf =  
yellowTaxiDf.where(yellowTaxiDf.tripDistance>2).where(yellowTaxiDf.pass  
engerCount==1)  
display(filteredTaxiDf.limit(10))
```

10. In a new code cell, using your OneLake path, write your filtered dataframe to a new Delta-Parquet table in your Fabric lakehouse. Run the cell.

```
Python
```

```
filteredTaxiDf.write.format("delta").mode("overwrite").save(oneLakePath  
+ '/Taxi/')
```

11. Finally, in a new code cell, test that your data was successfully written by reading your newly loaded file from OneLake. Run the cell.

```
Python
```

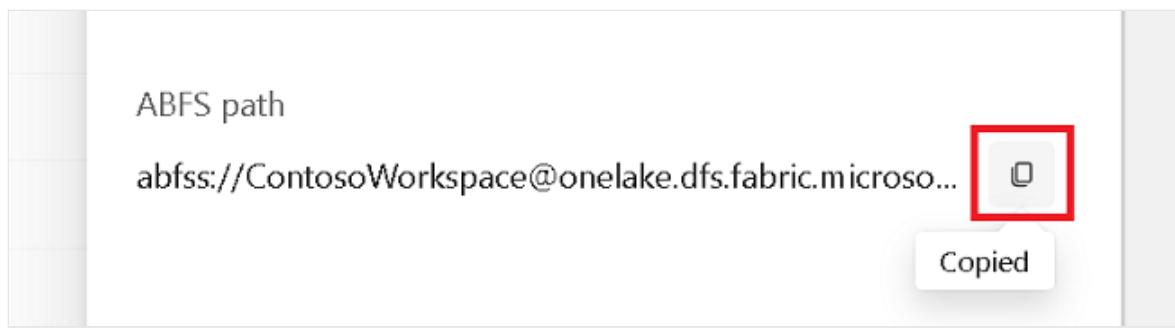
```
lakehouseRead = spark.read.format('delta').load(oneLakePath + '/Taxi/')  
display(lakehouseRead.limit(10))
```

Congratulations. You can now read and write data in OneLake using Apache Spark in Azure Synapse Analytics.

Read data from Synapse using SQL

Follow these steps to use SQL serverless to read data from OneLake from Azure Synapse Analytics.

1. Open a Fabric lakehouse and identify a table that you'd like to query from Synapse.
2. Right-click on the table and select **Properties**.
3. Copy the **ABFS path** for the table.



4. Open your Synapse workspace in [Synapse Studio](#).
5. Create a new SQL script.
6. In the SQL query editor, enter the following query, replacing `ABFS_PATH_HERE` with the path you copied earlier.

SQL

```
SELECT TOP 10 *
FROM OPENROWSET(
BULK 'ABFS_PATH_HERE',
FORMAT = 'delta') as rows;
```

7. Run the query to view the top 10 rows of your table.

Congratulations. You can now read data from OneLake using SQL serverless in Azure Synapse Analytics.

Related content

- [Integrate OneLake with Azure Storage Explorer](#)

Feedback

Was this page helpful?

| | |
|-----|----|
| Yes | No |
|-----|----|

[Provide product feedback](#) | [Ask the community](#)

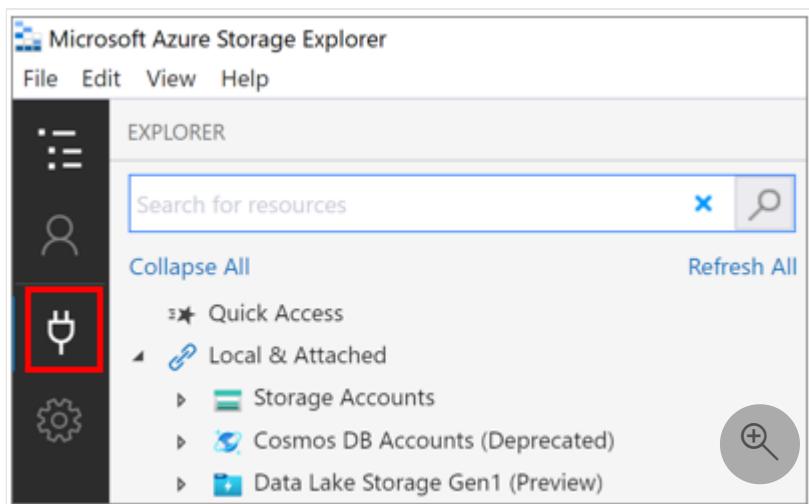
Integrate OneLake with Azure Storage Explorer

Article • 11/29/2023

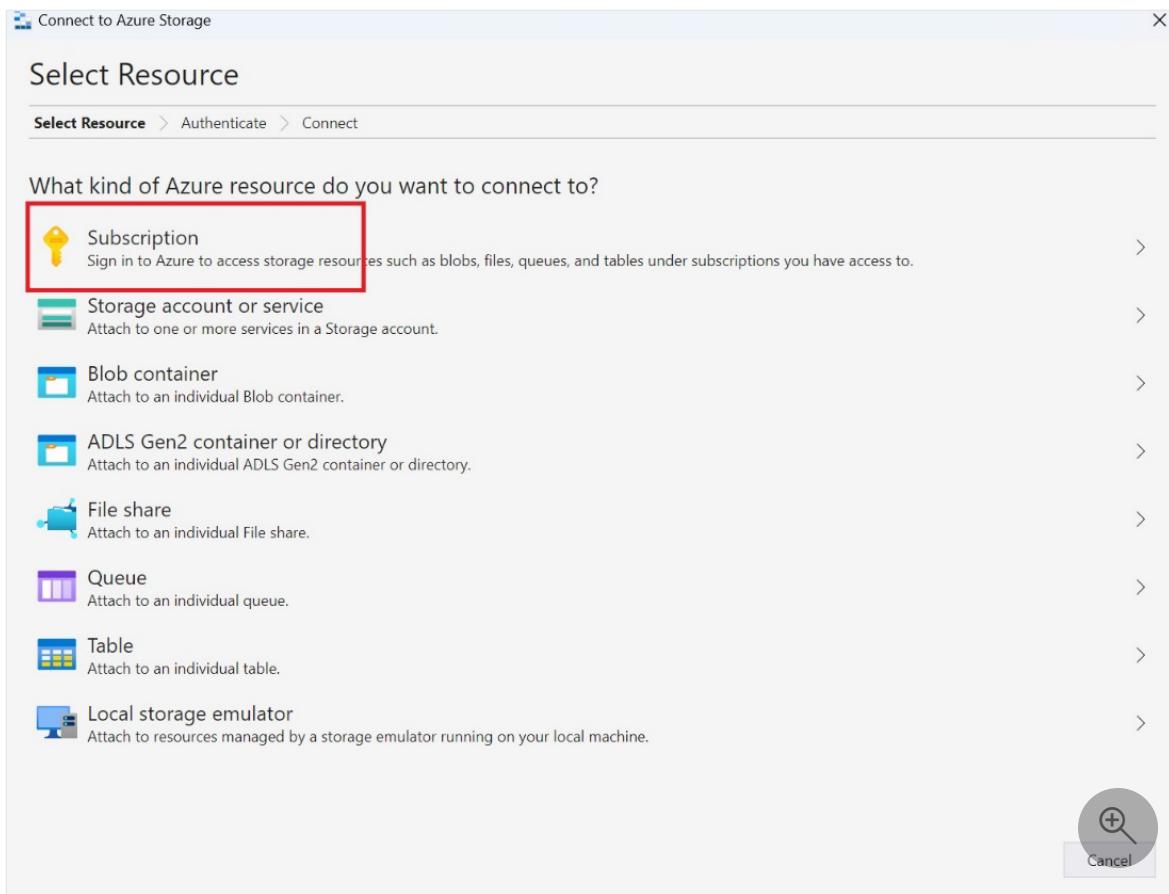
This article demonstrates OneLake integration with Azure Storage Explorer. Azure Storage Explorer allows you to view and manage your cloud storage account's contents. You can upload, download, or move files from one location to another.

Connect and use Azure Storage Explorer

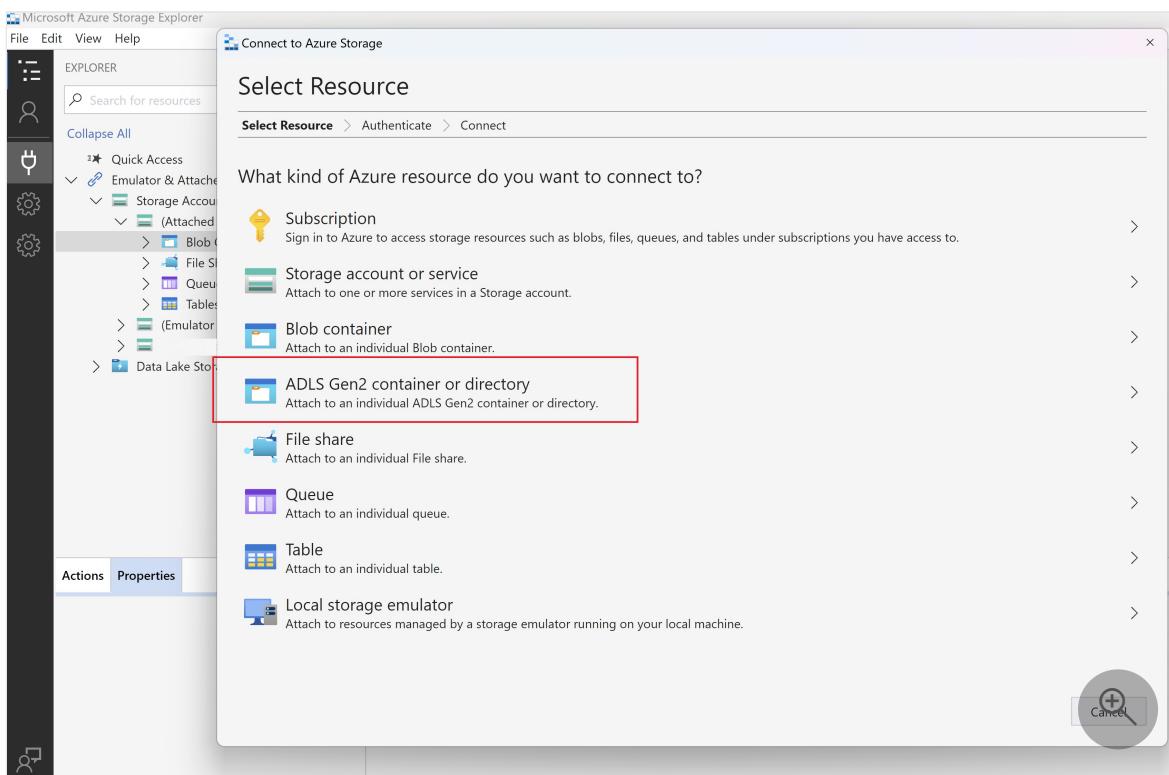
1. Install the latest version of Azure Storage Explorer from the [product webpage](#).
2. Check to ensure the version installed is 1.29.0 or higher. (Check the version by selecting **Help > About**.)
3. Select the **Open connect dialog** icon.



4. Azure Storage Explorer requires you to sign in to connect to Azure resources. Select **Subscription** and follow the instructions to sign in.

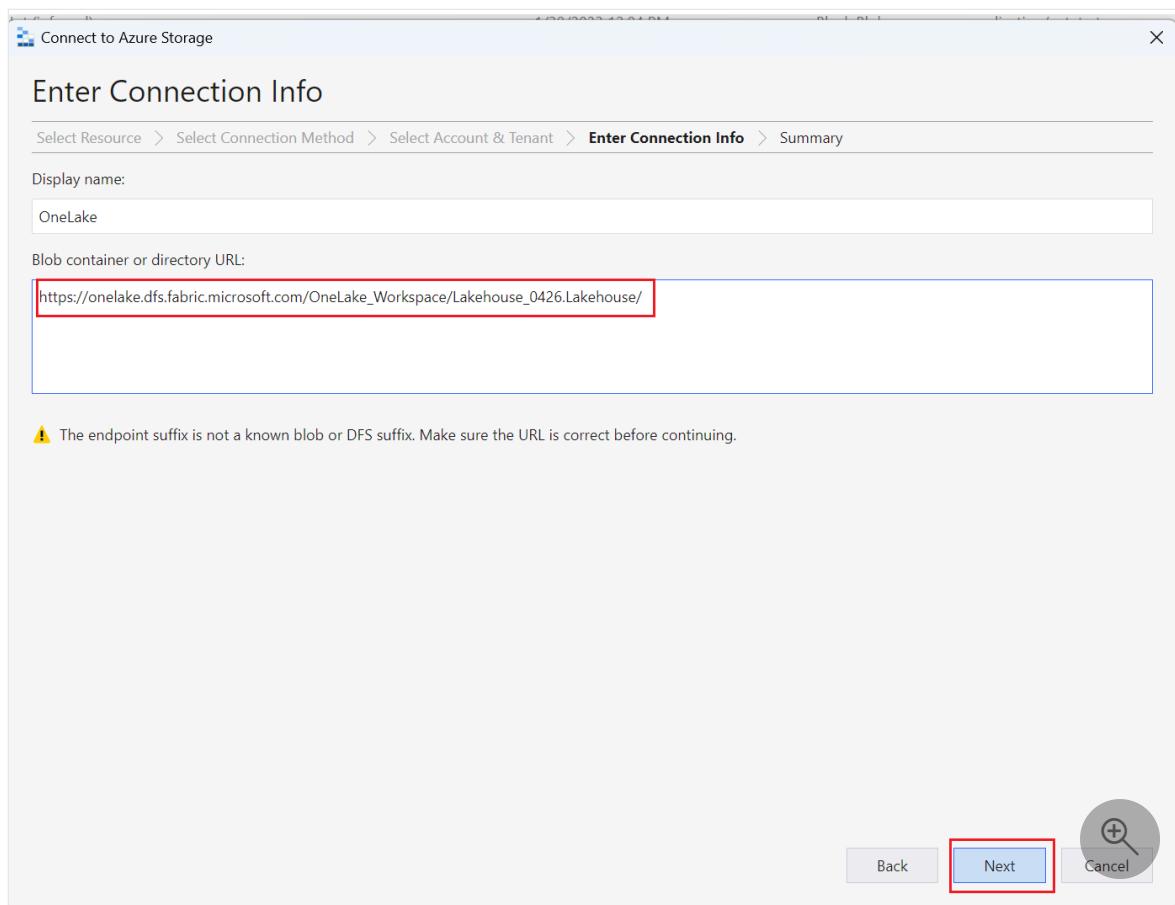


5. Connect to OneLake by selecting the **Open connect dialog** icon again and select **ADLS Gen2 container or directory**.

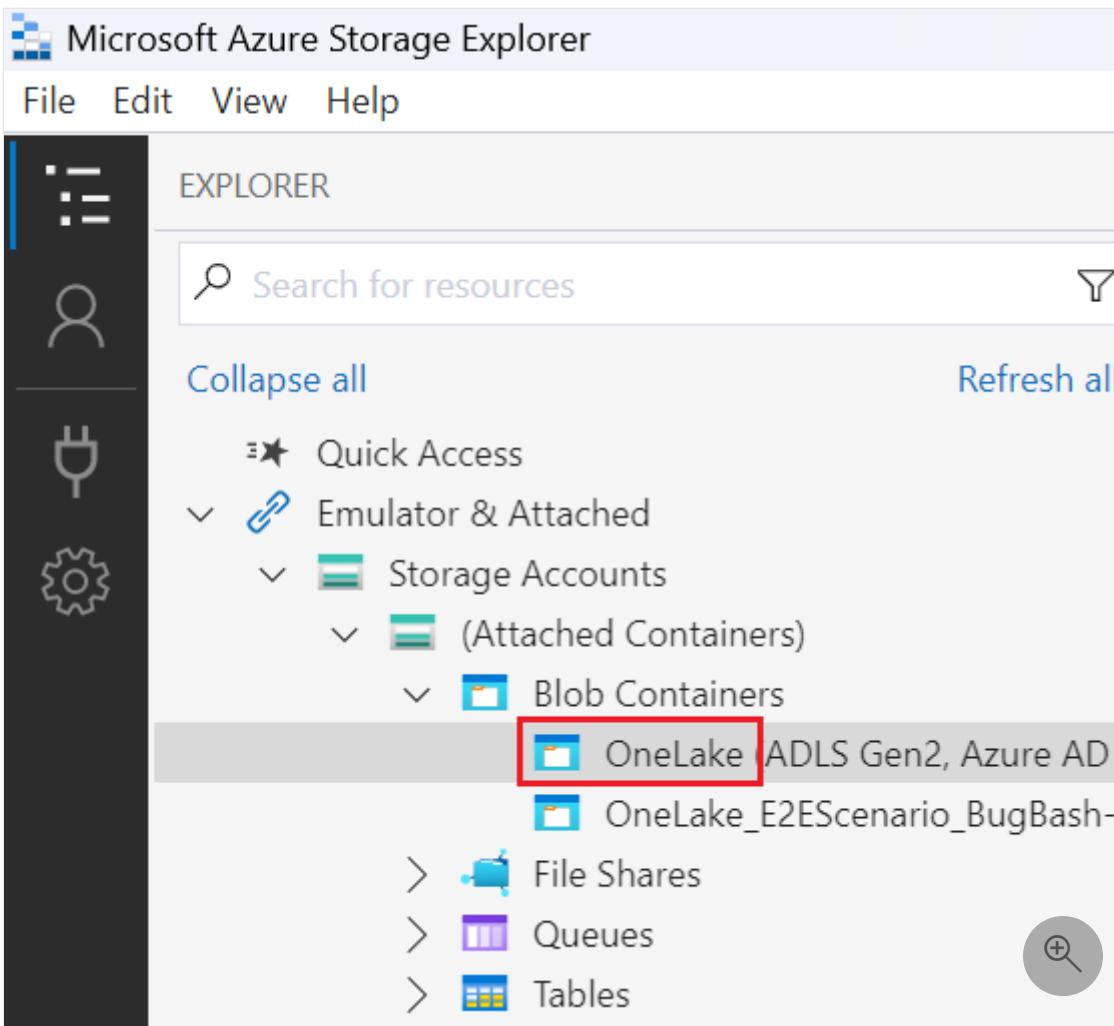


6. Enter URL details of the workspace or item you would like to connect to, in this format: `https://onelake.dfs.fabric.microsoft.com/{workspaceName}/{itemName.itemType}/`. You can find the workspace name and item name in the **Properties** pane of a file in the Microsoft Fabric portal.

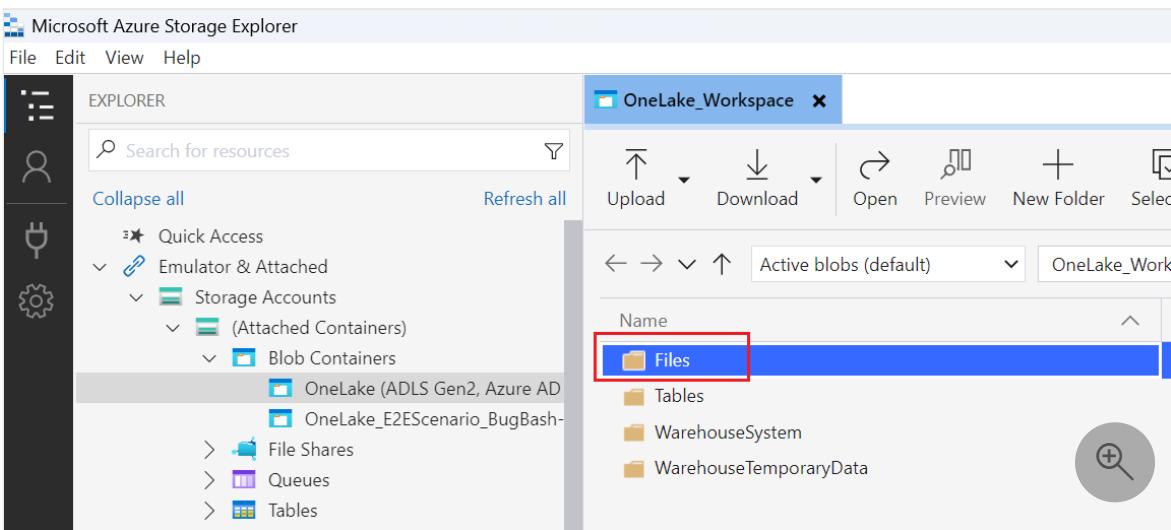
You can choose a **Display name** for convenience, then select **Next**.



7. Storage Explorer browses to the location of the OneLake you entered.



8. To view the contents, select the OneLake folder you connected.



9. Select **Upload**. In the **Select files to upload** dialog box, select the files that you want to upload.

The screenshot shows the OneLake_Workspace interface. At the top, there's a toolbar with various icons: Upload, Download, Open, Preview, New Folder, Select All, Copy, Paste, and Clone. The 'Upload' and 'Download' buttons are highlighted with a red box. Below the toolbar is a breadcrumb navigation bar showing 'OneLake_Workspace > Lakehouse_0426.Lakehouse'. The main area is a table listing files and folders:

| Name | Access Tier | Access |
|---------------------|-------------|----------------|
| Fold1 | | |
| TestFold1 | | |
| Campaign.csv | | Hot (inferred) |
| PhotoFact1.parquet | | Hot (inferred) |
| PhotoFact11.parquet | | Hot (inferred) |
| test11.csv | | Hot (inferred) |

A magnifying glass icon is located in the bottom right corner of the list area.

10. To download, select the folders or files that you want to download and then select **Download**.

11. To copy data across locations, select the folders you want to copy and select **Copy**, then navigate to the destination location and select **Paste**.

This screenshot shows the same workspace interface as the previous one, but with a different focus. The 'Copy' and 'Paste' buttons in the toolbar are highlighted with a red box. The rest of the interface and data list are identical to the first screenshot.

Limitations

If a workspace name has capital letters, deletion of files or folders fails due to a restriction from the storage service. We recommend using your workspace name in lowercase letters.

Related content

- [Integrate OneLake with Azure Databricks](#)
-

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Integrate OneLake with Azure Databricks

10/10/2025

This scenario shows how to connect to OneLake via Azure Databricks. After completing this tutorial, you'll be able to read and write to a Microsoft Fabric lakehouse from your Azure Databricks workspace.

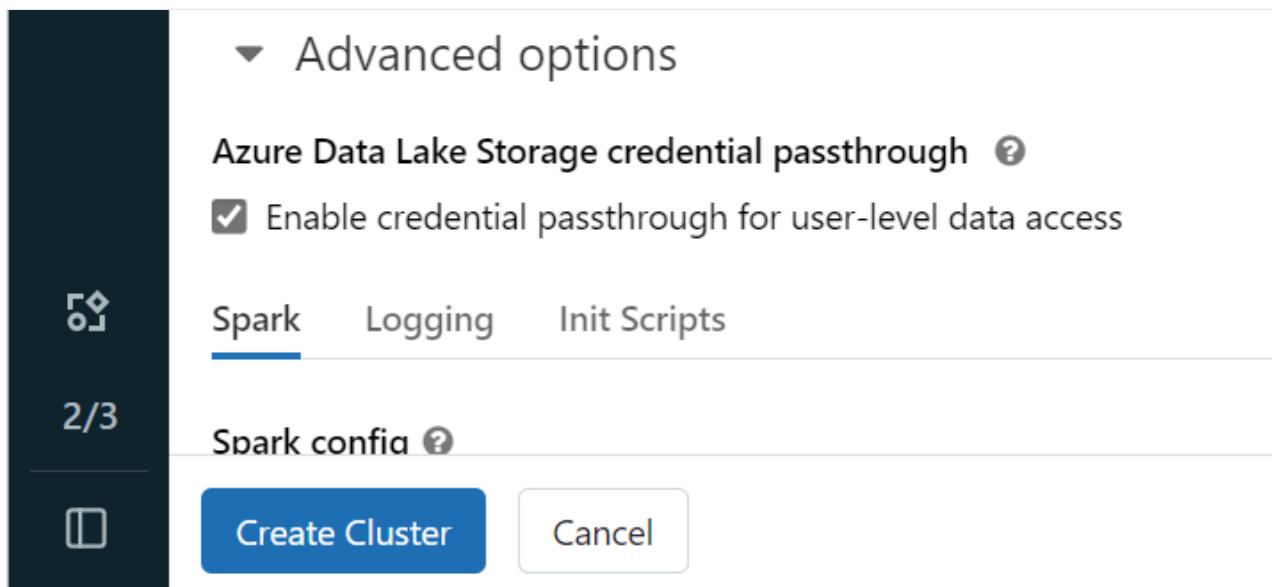
Prerequisites

Before you connect, you must have:

- A Fabric workspace and lakehouse.
- A premium Azure Databricks workspace. Only premium Azure Databricks workspaces support Microsoft Entra credential passthrough, which you need for this scenario.

Set up your Databricks workspace

1. Open your Azure Databricks workspace and select **Create > Cluster**.
2. To authenticate to OneLake with your Microsoft Entra identity, you must enable Azure Data Lake Storage (ADLS) credential passthrough on your cluster in the Advanced Options.



! Note

You can also connect Databricks to OneLake using a service principal. For more information about authenticating Azure Databricks using a service principal, see [Manage service principals](#).

3. Create the cluster with your preferred parameters. For more information on creating a Databricks cluster, see [Configure clusters - Azure Databricks](#).
4. Open a notebook and connect it to your newly created cluster.

Author your notebook

1. Navigate to your Fabric lakehouse and copy the Azure Blob Filesystem (ABFS) path to your lakehouse. You can find it in the **Properties** pane.

 **Note**

Azure Databricks only supports the Azure Blob Filesystem (ABFS) driver when reading and writing to ADLS Gen2 and OneLake:

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

2. Save the path to your lakehouse in your Databricks notebook. This lakehouse is where you write your processed data later:

Python

```
oneLakePath =  
'abfss://myWorkspace@onelake.dfs.fabric.microsoft.com/myLakehouse.lakehouse/F  
iles/'
```

3. Load data from a Databricks public dataset into a dataframe. You can also read a file from elsewhere in Fabric or choose a file from another ADLS Gen2 account you already own.

Python

```
yellowTaxiDF = spark.read.format("csv").option("header",  
"true").option("inferSchema", "true").load("/databricks-  
datasets/nyctaxi/tripdata/yellow/yellow_tripdata_2019-12.csv.gz")
```

4. Filter, transform, or prep your data. For this scenario, you can trim down your dataset for faster loading, join with other datasets, or filter down to specific results.

Python

```
filteredTaxiDF =  
yellowTaxiDF.where(yellowTaxiDF.fare_amount<4).where(yellowTaxiDF.passenger_c  
ount==4)  
display(filteredTaxiDF)
```

5. Write your filtered dataframe to your Fabric lakehouse using your OneLake path.

Python

```
filteredTaxiDF.write.format("csv").option("header",  
"true").mode("overwrite").csv(oneLakePath)
```

6. Test that your data was successfully written by reading your newly loaded file.

Python

```
lakehouseRead = spark.read.format('csv').option("header",  
"true").load(oneLakePath)  
display(lakehouseRead.limit(10))
```

This completes the setup and now you can now read and write data in Fabric using Azure Databricks.

Connecting to OneLake using Databricks serverless compute

[Databricks serverless compute]/azure/databricks/compute/serverless/) allows you to run workloads without provisioning a cluster. As per Databricks serverless limitations, to automate the configuration of Spark on serverless compute, Databricks doesn't allow configuring [Spark properties](#) outside supported properties that are listed [here](#).

ⓘ Note

This limitation isn't unique to Azure Databricks. Databricks Serverless implementations on [Amazon Web Services \(AWS\)](#) and [Google Cloud](#) exhibit the same behavior.

If you attempt to modify or set an unsupported Spark configuration in a notebook linked to Databricks serverless compute, the system returns a CONFIG_NOT_AVAILABLE error.

A screenshot of a Databricks notebook interface. At the top, there's a toolbar with a play button, a status message 'Last execution failed', a page number '9', and language selection 'Python'. Below the toolbar, two lines of Python code are shown: `spark.conf.set("fs.azure.account.auth.type", "OAuth")` and `spark.conf.set("fs.azure.account.oauth.provider.type", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")`. A red box highlights an error message: '[CONFIG_NOT_AVAILABLE] Configuration fs.azure.account.auth.type is not available. SQLSTATE: 42K01'. At the bottom of the code editor are 'Diagnose error' and 'Debug' buttons, and a 'Assistant Quick Fix' dropdown.

OneLake supports inbound connectivity from Databricks serverless compute. You can connect to OneLake as provided you have successfully authenticated and there's network path between Databricks serverless compute and OneLake. With Databricks serverless, you must ensure that your code doesn't modify any unsupported Spark properties.

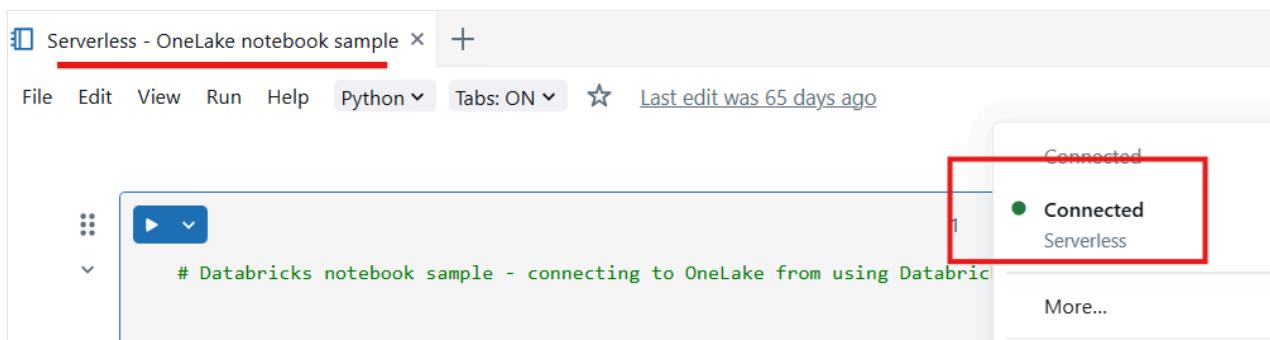
Prerequisites

Before you connect, you must have:

- A Fabric workspace and lakehouse.
- A premium Azure Databricks workspace.
- A service principal with a minimum of **Contributor** workspace role assignment.
- Database secrets or Azure Key Vault (AKV) to store and retrieve secrets. This example uses Databricks secrets.

Author your notebook

1. Create a notebook in Databricks workspace and attach it to serverless compute.



2. Import Python modules - in this sample, you're using three modules:

- **msal** is Microsoft Authentication Library (MSAL) and it is designed to help developers integrate Microsoft identity platform authentication into their applications.
- **requests** module is used to make HTTP requests using Python.
- **delta lake** is used to read and write Delta Lake tables using Python.

Python

```
from msal import ConfidentialClientApplication
import requests
from deltalake import DeltaTable
```

3. Declare variables for Microsoft Entra tenant including application ID. Use the tenant ID of the tenant where Microsoft Fabric is deployed.

Python

```
# Fetch from Databricks secrets.
tenant_id = dbutils.secrets.get(scope="",key="")
client_id = dbutils.secrets.get(scope="",key="")
client_secret = dbutils.secrets.get(scope="",key="")
```

4. Declare Fabric workspace variables.

Python

```
workspace_id = "<replace with workspace name>"
lakehouse_id = "<replace with lakehouse name>"
table_to_read = "<name of lakehouse table to read>"
storage_account_name = workspace_id
onelake_uri =
f"abfss://{{workspace_id}}@onelake.dfs.fabric.microsoft.com/{{lakehouse_id}}.lake
house/Tables/{{table_to_read}}"
```

5. Initialize client to acquire token.

Python

```
authority = f"https://login.microsoftonline.com/{tenant_id}"

app = ConfidentialClientApplication(
    client_id,
    authority=authority,
    client_credential=client_secret
)

result = app.acquire_token_for_client(scopes=
["https://onelake.fabric.microsoft.com/.default"])

if "access_token" in result:
    access_token = result["access_token"]
    print("Access token acquired.")
    token_val = result['access_token']
```

6. Read a delta table from OneLake

Python

```
dt = DeltaTable(onelake_uri, storage_options={"bearer_token": f"{token_val}"},  
"use_fabric_endpoint": "true"})  
df = dt.to_pandas()  
print(df.head())
```

ⓘ Note

The service principal has **Contributor** workspace role assignment and you can use it to write data back to OneLake.

This completes the setup and you can now read data from OneLake using Databricks a notebook attached to serverless compute.

Related content

- [Integrate OneLake with Azure HDInsight](#)

Integrate Databricks Unity Catalog with OneLake

Article • 04/09/2024

This scenario shows how to integrate Unity Catalog external Delta tables to OneLake using shortcuts. After completing this tutorial, you'll be able to automatically sync your Unity Catalog external Delta tables to a Microsoft Fabric lakehouse.

Prerequisites

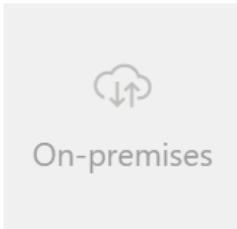
Before you connect, you must have:

- A [Fabric workspace](#).
- A [Fabric lakehouse](#) in your workspace.
- [External Unity Catalog Delta tables](#) created within your Azure Databricks workspace.

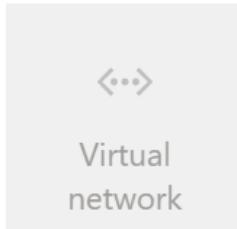
Set up your Cloud storage connection

First, examine which storage locations in Azure Data Lake Storage Gen2 (ADLS Gen2) your Unity Catalog tables are using. This Cloud storage connection is used by OneLake shortcuts. To create a Cloud connection to the appropriate Unity Catalog storage location:

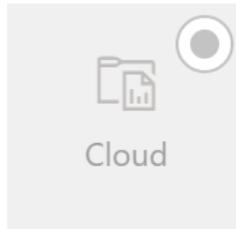
1. Create a Cloud storage connection used by your Unity Catalog tables. See how to set up a [ADLS Gen2 connection](#).
2. Once you create the connection, obtain the connection ID by selecting **Settings**  > **Manage connections and gateways** > **Connections** > **Settings**.



On-premises



Virtual network



Cloud

Connection name *

my_uc_adlsgen2_connection

Connection ID

adff3fc4-c51

Connection type

Azure Data Lake Storage Gen2

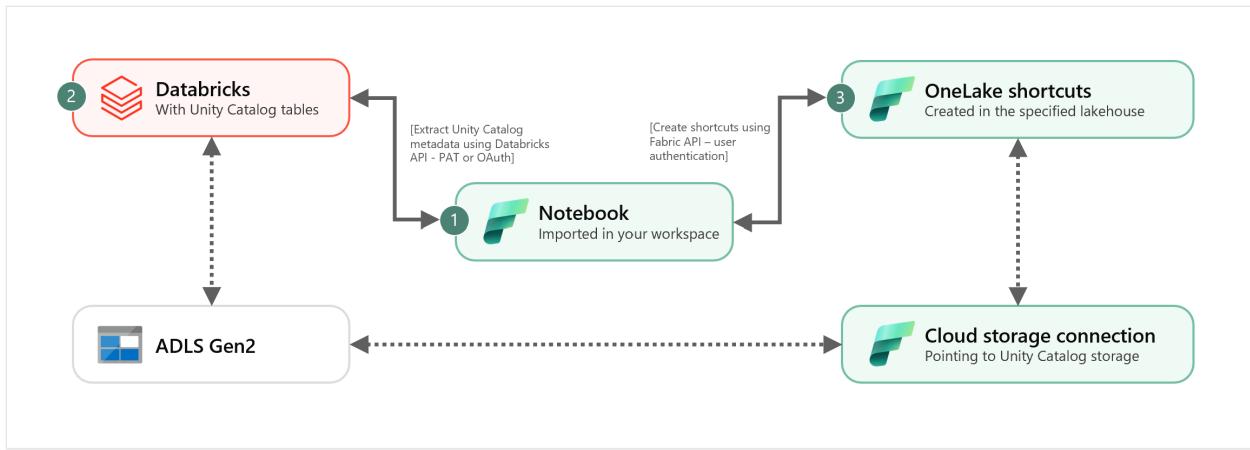


ⓘ Note

Granting users direct storage level access to external location storage in ADLS Gen2 does not honor any permissions granted or audits maintained by Unity Catalog. Direct access will bypass auditing, lineage, and other security/monitoring features of Unity Catalog including access control and permissions. You are responsible for managing direct storage access through ADLS Gen2 and ensuring that users have the appropriate permissions granted via Fabric. Avoid all scenarios granting direct storage level write access for buckets storing Databricks managed tables. Modifying, deleting, or evolving any objects directly through storage which were originally managed by Unity Catalog can result in data corruption.

Run the notebook

Once the Cloud connection ID is obtained, integrate Unity Catalog tables to Fabric lakehouse as follows:



- 1. Import sync notebook** to your Fabric workspace. [This notebook](#) exports all Unity Catalog tables metadata from a given catalog and schemas in your metastore.
- 2. Configure the parameters** in the first cell of the notebook to integrate Unity Catalog tables. The Databricks API, authenticated through PAT token, is utilized for exporting Unity Catalog tables. The following snippet is used to configure the source (Unity Catalog) and destination (OneLake) parameters. Ensure to replace them with your own values.

Python

```

# Databricks workspace
dbx_workspace = "<databricks_workspace_url>"
dbx_token = "<pat_token>"
# Unity Catalog
dbx_uc_catalog = "catalog1"
dbx_uc_schemas = '[ "schema1", "schema2" ]'

# Fabric
fab_workspace_id = "<workspace_id>"
fab_lakehouse_id = "<lakehouse_id>"
fab_shortcut_connection_id = "<connection_id>"
# If True, UC table renames and deletes will be considered
fab_consider_dbx_uc_table_changes = True

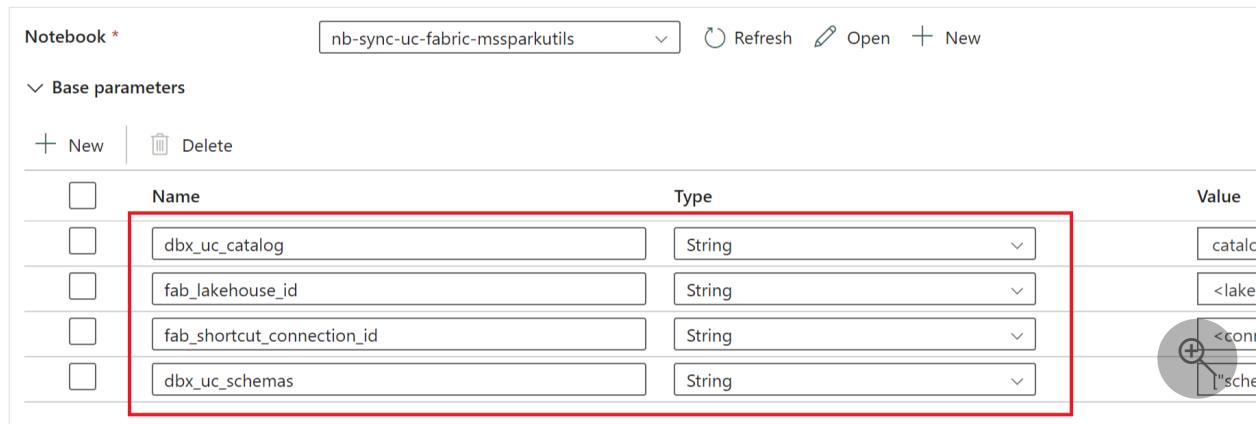
```

- 3. Run all cells** of the notebook to start synchronizing Unity Catalog Delta tables to OneLake using shortcuts. Once notebook is completed, shortcuts to Unity Catalog Delta tables are available in the lakehouse, SQL endpoint, and semantic model.

Schedule the notebook

If you want to execute the notebook at regular intervals to integrate Unity Catalog Delta tables into OneLake without manual resync / rerun, you can either [schedule the notebook](#) or utilize a [notebook activity](#) in a data pipeline within Fabric Data Factory.

In the latter scenario, if you intend to pass parameters from the data pipeline, designate the first cell of the notebook as a [toggle parameter cell](#) and provide the appropriate parameters in the pipeline.



The screenshot shows the 'Base parameters' section of a Databricks notebook named 'nb-sync-uc-fabric-mssparkutils'. It contains four parameters:

| Name | Type | Value |
|----------------------------|--------|---------|
| dbx_uc_catalog | String | catalog |
| fab_lakehouse_id | String | <lake |
| fab_shortcut_connection_id | String | <conn |
| dbx_uc_schemas | String | ["sche |

Other considerations

- For production scenarios, we recommend using [Databricks OAuth](#) for authentication and Azure Key Vault to manage secrets. For instance, you can use the [MSSparkUtils](#) credentials utilities to access Key Vault secrets.
- The notebook works with Unity Catalog external Delta tables. If you're using multiple Cloud storage locations for your Unity Catalog tables, i.e. more than one ADLS Gen2, the recommendation is to run the notebook separately by each Cloud connection.
- Unity Catalog managed Delta tables, views, materialized views, streaming tables and non-Delta tables are not supported.
- Changes to Unity Catalog table schemas like add / delete columns are reflected automatically in the shortcuts. However, some updates like Unity Catalog table rename and deletion require a notebook resync / rerun. This is considered by `fab_consider_dbx_uc_table_changes` parameter.
- For writing scenarios, using the same storage layer across different compute engines can result in unintended consequences. Be sure to grasp the implications when using different Spark compute engines and runtime versions.

Related content

- [Integrate OneLake with Azure Databricks](#)
- [OneLake shortcuts](#)

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Write Iceberg tables from Snowflake to OneLake (Preview)

i Important

This feature is currently in **Public Preview**. Behavior may change before General Availability.

Overview

You can configure a new or existing Snowflake database to automatically store Iceberg tables in Microsoft OneLake. This feature creates a new data item in Microsoft Fabric, and Iceberg tables that you create in Snowflake are stored there by default. With this capability, both Microsoft Fabric and Snowflake can work with a single copy of Iceberg data without duplication or movement.

This article shows you how to:

- Allow connectivity between Snowflake and Fabric
- Configure a new Snowflake database to write Iceberg tables to OneLake by default

Prerequisites

1. Because this feature is in a Preview state, you first need to enable this setting at the **tenant or capacity** level.
 - Your tenant admin can enable the setting tenant-wide using the "[Enable Snowflake database item \(preview\)](#)" setting seen in the [Admin portal](#).
 - Alternatively, your capacity admin can enable this delegated tenant setting in the [capacity settings area](#).
2. Select (or create) a Fabric workspace for the Snowflake database item.
 - To keep things simple, use alphanumeric characters only for your workspace name.
 - If workspace name has special characters, copy the workspace ID from the browser URL seen when the workspace is open.
 - You must have the Admin or Member role to proceed with this feature.
3. Find your Fabric tenant ID (a GUID) and your Snowflake account identifier.

- You can find your Fabric tenant ID by selecting your profile in the top-right corner of the Fabric UI and hovering over the tenant information.
- You can find your Snowflake account identifier in the lower-left corner of the Snowflake UI and selecting **Account details**.

4. Identify the Snowflake warehouse that should be used when Fabric communicates with Snowflake (for example, `COMPUTE_WH`).

5. Pick a strong password to assign to the new Snowflake user for Fabric to use when communicating with Snowflake.

- This article assumes the connection to Snowflake will use a user account with a password. You may use the KeyPair authentication method alternatively.

Create least-privileged user and role in Snowflake

In your Snowflake account, log in with a user that has an administrative role.

1. Run the following SQL statements in Snowflake:

SQL

```
-- Use a privileged role
USE ROLE ACCOUNTADMIN;

-- Create least-privilege role (if not exists)
CREATE ROLE IF NOT EXISTS R_ICEBERG_METADATA;

-- Create service user (adjust LOGIN_NAME / PASSWORD as needed)
CREATE USER IF NOT EXISTS SVC_FABRIC_ICEBERG_METADATA
    TYPE = LEGACY_SERVICE
    LOGIN_NAME = 'SVC_FABRIC_ICEBERG_METADATA'
    DISPLAY_NAME = 'Service - Fabric Iceberg Metadata'
    PASSWORD = '<STRONG_PASSWORD_HERE>'
    MUST_CHANGE_PASSWORD = FALSE
    DEFAULT_ROLE = R_ICEBERG_METADATA;

-- Grant role to user
GRANT ROLE R_ICEBERG_METADATA TO USER SVC_FABRIC_ICEBERG_METADATA;

-- Allow role to use an existing warehouse (adjust COMPUTE_WH as needed)
GRANT USAGE ON WAREHOUSE COMPUTE_WH TO ROLE R_ICEBERG_METADATA;
```

2. Go to **Ingestion > Add data**.

3. Select **Microsoft OneLake**.

4. Enter your Fabric tenant ID (a GUID).

5. If prompted, click **Provide consent**.

- If prompted for permissions, review and accept (may require Entra admin). You may close the popup once complete.

6. Copy the multitenant app name displayed (used to write Iceberg tables to OneLake).

7. Keep this browser tab open.

Prepare connection in Fabric

In a new browser tab:

1. Open [the Fabric web UI](#).

2. Go to **Settings** (top right) > **Manage connections and gateways**.

3. Select + New.

4. Choose **Cloud connection**. Enter the following details:

- **Connection type:** Snowflake
- **Server:** `https://<accountIDpart1>-<accountIDpart2>.snowflakecomputing.com`
- **Warehouse:** `COMPUTE_WH` (or your choice)
- **Authentication method:** Snowflake
- **Username:** `SVC_FABRIC_ICEBERG_METADATA` (unless customized)
- **Password:** *Your chosen password*

5. Create the connection. If it fails, recheck the information from the previous section.

6. Copy the connection ID (GUID) from the connection page.

Connection name *

`SVC_FABRIC_ICEBERG_METADATA`

Connection ID

`00000000-0000-0000-0000-000000000000`

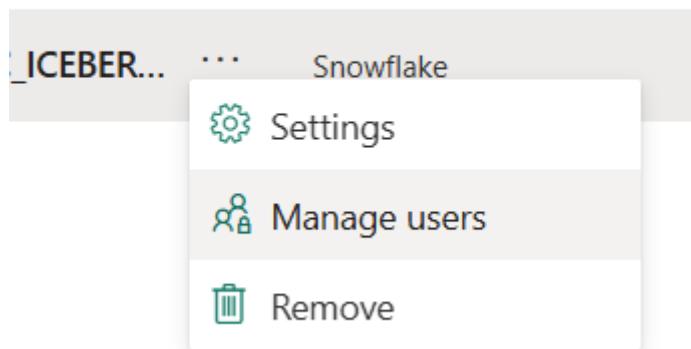
Connection type

`Snowflake`

7. Filter the list of connections to see your new connection.

8. On that connection, select **Manage users**.

Connection type



9. Grant access to the Snowflake multitenant app, then select **Share**.

ⓘ Note

If you granted consent in the previous section, it may take a few minutes for the multitenant app to appear in your search.

10. Navigate to your Fabric workspace.

11. Select **Manage access**, **Add people or groups**.

12. Paste the multitenant app name. Grant at least **Contributor** role to ensure the app can be used by Snowflake to create the data item and write data.

The screenshot shows a 'Add people' dialog. At the top left is a back arrow, the title 'Add people', and a close button. Below the title is the text 'test-icebergdata'. A note below the app name explains roles: 'Admins, members, and contributors have edit and view access. Viewers only have view access.' with a 'Learn more' link. Below the note is a list box containing a single entry: 'uu6bu6snowflakepacint'. At the bottom are buttons for 'Contributor' (with a dropdown arrow) and 'Add'.

13. Keep this browser tab open.

Establish connectivity in Snowflake

Continue the setup flow in Snowflake:

1. Enter the **Fabric workspace name** (if alphanumeric) or workspace ID.
2. Enter the **Fabric connection ID** that you copied in the previous section.
3. Provide a new **Snowflake database name** (for example, `SnowflakeFabricIcebergDB`).
 - If you choose to have the item name in Fabric differ, keep track of that item name.
4. Confirm that you see the message "**Fabric item and database successfully created.**"
5. Continue, and acknowledge the configuration when prompted.
 - If you enter a custom external volume name, keep track of that name.
6. Select **Create volume**, then **View database**.
 - Confirm you see the new database.
7. Run the following SQL statements:
 - Replace `SnowflakeFabricIcebergDB` with your database name.

SQL

```
-- Grant Iceberg metadata role permissions on the new database
BEGIN
    LET db STRING := 'SnowflakeFabricIcebergDB';

    EXECUTE IMMEDIATE 'GRANT USAGE ON DATABASE ' || db || ' TO ROLE
R_ICEBERG_METADATA';
    EXECUTE IMMEDIATE 'GRANT MONITOR ON DATABASE ' || db || ' TO ROLE
R_ICEBERG_METADATA';
    EXECUTE IMMEDIATE 'GRANT USAGE ON ALL SCHEMAS IN DATABASE ' || db || ' TO
ROLE R_ICEBERG_METADATA';
    EXECUTE IMMEDIATE 'GRANT USAGE ON FUTURE SCHEMAS IN DATABASE ' || db || ' TO
ROLE R_ICEBERG_METADATA';
    EXECUTE IMMEDIATE 'GRANT SELECT ON ALL ICEBERG TABLES IN DATABASE ' || db ||
' TO ROLE R_ICEBERG_METADATA';
    EXECUTE IMMEDIATE 'GRANT SELECT ON FUTURE ICEBERG TABLES IN DATABASE ' || db ||
' TO ROLE R_ICEBERG_METADATA';
END;

GRANT USAGE ON EXTERNAL VOLUME SnowflakeFabricIcebergDB TO ROLE
R_ICEBERG_METADATA;
GRANT OWNERSHIP ON EXTERNAL VOLUME SnowflakeFabricIcebergDB TO ROLE
R_ICEBERG_METADATA COPY CURRENT GRANTS;
```

You're done with setup! Proceed to create an Iceberg table and read it in Fabric.

Create an Iceberg table in Snowflake

In Snowflake:

1. Run the following SQL statements to create an Iceberg table and populate it with data:

SQL

```
-- Create a sample Iceberg table
CREATE ICEBERG TABLE SnowflakeFabricIcebergDB.PUBLIC.SampleIcebergTable (
    id INT,
    name STRING
)
CATALOG = 'SNOWFLAKE';

-- Insert sample rows
INSERT INTO SnowflakeFabricIcebergDB.PUBLIC.SampleIcebergTable VALUES
(1, 'Alpha'),
(2, 'Beta'),
(3, 'Gamma');
```

2. Run the following SQL statement to read the Iceberg table in Snowflake:

SQL

```
-- Display all rows in the Iceberg table
SELECT * FROM SnowflakeFabricIcebergDB.PUBLIC.SampleIcebergTable;
```

Read the Iceberg table in Fabric

In [the Fabric web UI](#):

1. Navigate to your workspace.
2. Locate the Snowflake database item. Refresh if needed.
3. Open it to see the Iceberg table from Part 4.
4. Select **SQL analytics endpoint** (top right) to query this table using SQL:

SQL

```
-- Display all rows in the Iceberg table
SELECT * FROM [SnowflakeFabricIcebergDB].[PUBLIC].[SampleIcebergTable];
```

You should see the same data displayed in both Snowflake and Fabric.

This data resides in OneLake, and both Fabric and Snowflake can work with the same copy of data, no data movement or duplication required!

Last updated on 11/18/2025

Integrate OneLake with Azure HDInsight

Article • 06/05/2024

Azure HDInsight is a managed cloud-based service for big data analytics that helps organizations process large amounts of data. This tutorial shows how to connect to OneLake with a Jupyter notebook from an Azure HDInsight cluster.

Using Azure HDInsight

To connect to OneLake with a Jupyter notebook from an HDInsight cluster:

1. Create an HDInsight (HDI) Apache Spark cluster. Follow these instructions: [Set up clusters in HDInsight](#).
 - a. While providing cluster information, remember your Cluster login Username and Password, as you need them to access the cluster later.
 - b. Create a user assigned managed identity (UAMI): [Create for Azure HDInsight - UAMI](#) and choose it as the identity in the **Storage** screen.

The screenshot shows the 'Create HDInsight cluster' wizard on the 'Storage' tab. Under 'Primary storage', the 'Primary storage type' is set to 'Azure Data Lake Storage Gen2'. In the 'Primary storage account' section, a dropdown shows 'Select an existing storage account' and lists 'test111111111-2023-02-07t20-15-10-425z' with a green checkmark. The 'Identity' section at the bottom has a red box around the 'User-assigned managed identity' dropdown, which is currently empty. A search icon is visible next to the dropdown.

2. Give this UAMI access to the Fabric workspace that contains your items. For help deciding what role is best, see [Workspace roles](#).

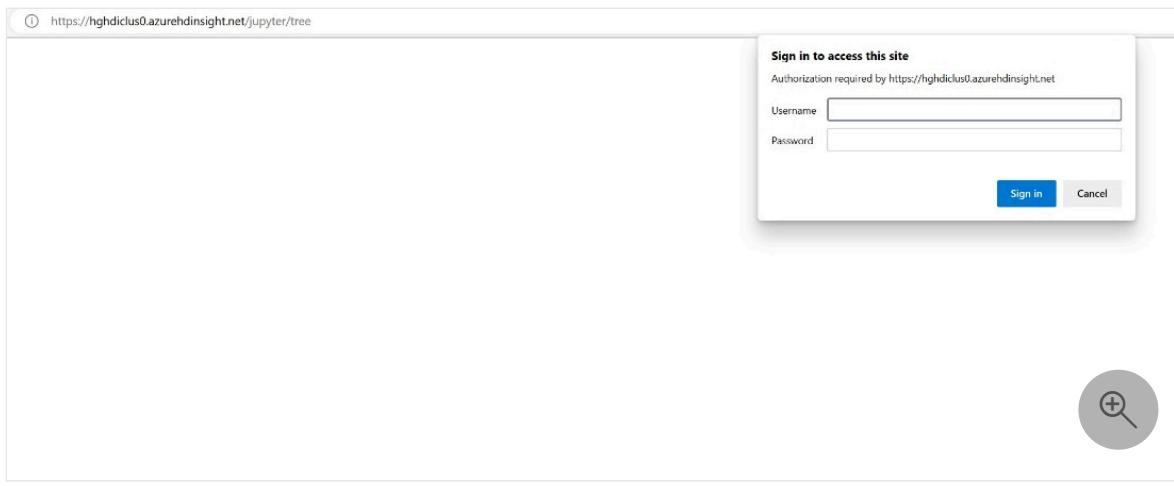
The screenshot shows the Azure Data Lake Storage Gen2 (Lakehouse) interface. On the left, the 'Lakehouse explorer' pane displays a tree view of workspaces and their contents. A workspace named 'Lakehouse_0426' is expanded, showing 'Tables' (with 'triptable' and 'userdata') and 'Files' (with 'TestFold1' and 'adls_sto_shortcut'). To the right, a central area says 'Load data in your lakehouse' with three buttons: 'New Dataflow Gen2', 'New data pipeline', and 'Open notebook'. On the far right, the 'Manage access' pane shows a search bar and a list of users. A user named 'hghdiwest2MI' is highlighted with a red box, showing they are an 'Admin' member.

3. Navigate to your lakehouse and find the name for your workspace and lakehouse.
You can find them in the URL of your lakehouse or the **Properties** pane for a file.

4. In the Azure portal, look for your cluster and select the notebook.

The screenshot shows the Azure HDInsight cluster overview page for 'hghdiclus0'. The left sidebar has sections like Overview, Activity log, Tags, and Settings. Under Settings, there's a 'Jupyter notebook' link which is highlighted with a red box. The main content area shows the cluster status as 'Running' in 'West US 2', and various management options like Auto scale, Applications, Script actions, and Monitor integration.

5. Enter the credential information you provided while creating the cluster.



6. Create a new Apache Spark notebook.
7. Copy the workspace and lakehouse names into your notebook and build the OneLake URL for your lakehouse. Now you can read any file from this file path.

Python

```
fp = 'abfss://' + 'Workspace Name' +
 '@onelake.dfs.fabric.microsoft.com/' + 'Lakehouse Name' + '/Files/'
 df = spark.read.format("csv").option("header", "true").load(fp +
 "test1.csv")
 df.show()
```

8. Try writing some data into the lakehouse.

Python

```
writecsvdf = df.write.format("csv").save(fp + "out.csv")
```

9. Test that your data was successfully written by checking your lakehouse or by reading your newly loaded file.

You can now read and write data in OneLake using your Jupyter notebook in an HDI Spark cluster.

Related content

- [OneLake security](#)

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Ask the community ↗](#)

Use OneLake files in Microsoft Foundry

Use Microsoft OneLake as a knowledge source for Microsoft Foundry. You can connect directly and securely to OneLake from Foundry, index unstructured and semi structured files stored in OneLake (including files that arrive through shortcuts), and then use that indexed content as a knowledge source inside agents in Foundry.

With this integration, you can ground your agents on the same enterprise data that already lives in OneLake, instead of creating new copies of files in separate AI specific stores.

Permissions and governance are enforced through the same OneLake and Fabric controls that you use for analytics workloads.

Prerequisites

- A lakehouse in Fabric. If you don't have a lakehouse, follow the steps in [Create a lakehouse with OneLake](#).
 - Files in the **Files** folder of the lakehouse.
- A Foundry project. If you don't have one, follow the steps in [Create a project](#).
- An Azure AI Search service at the Basic tier or higher. If you don't have one, follow the steps in [Create an Azure AI Search service](#).
 - The search service must be in the same tenant as your Fabric workspace.
 - In this article, you create and assign a managed identity for the search service. To create a managed identity, you must be an Owner or User Access Administrator roles. To assign roles, you must be an Owner, User Access Administrator, Role-based Access Control Administrator, or a member of a custom role with Microsoft.Authorization/roleAssignments/write permissions.

Index data from OneLake files

Use Azure AI Search to configure a OneLake files indexer to make your lakehouse data searchable as a knowledge source.

Review the prerequisites in [Index data from OneLake files and shortcuts > Prerequisites](#).

Then, follow the steps for system managed identity in [Index data from OneLake files and shortcuts > Grant permissions](#).

Create a OneLake connection in Foundry

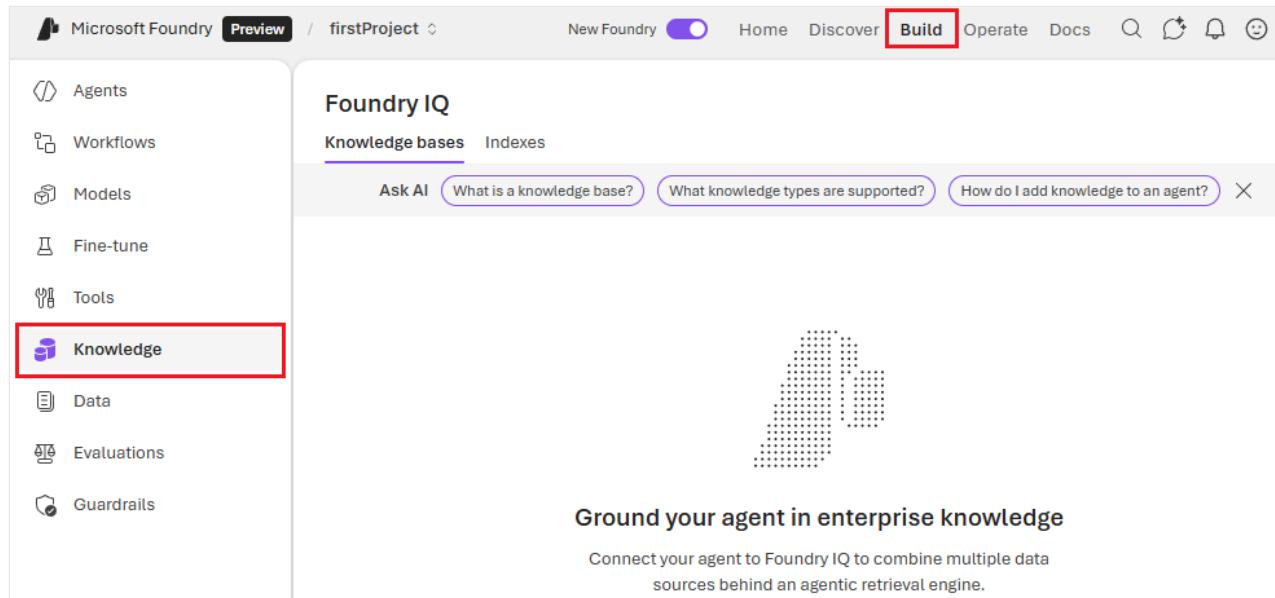
1. Sign in to Microsoft Foundry .

Make sure the **New Foundry** toggle is **On**. The steps in this article refer to [Microsoft Foundry \(new\)](#).



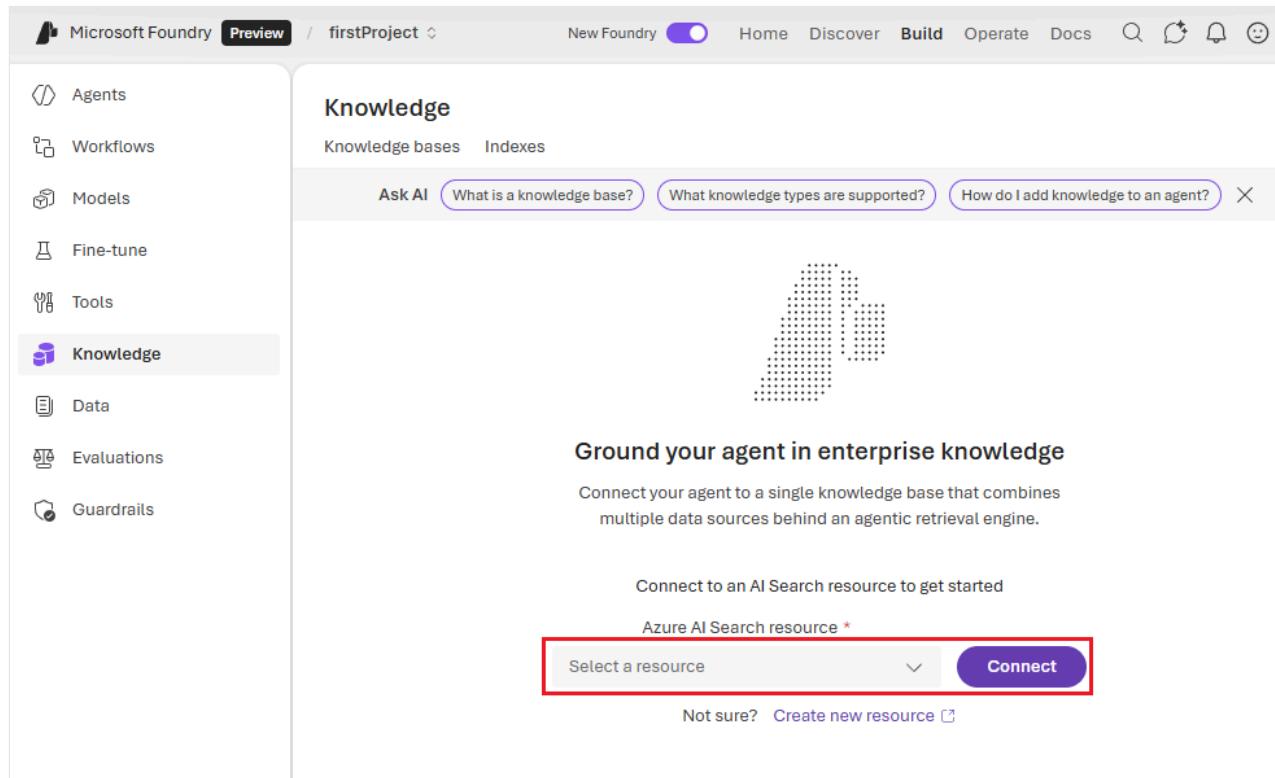
2. Open the project that you want to work in.

3. Select **Build** from the navigation menu, then select **Knowledge** from the left pane.



The screenshot shows the Microsoft Foundry web interface. At the top, there's a navigation bar with tabs for Home, Discover, **Build** (which is highlighted with a red box), Operate, Docs, and several icons for search, refresh, and notifications. Below the navigation is a sidebar on the left containing links for Agents, Workflows, Models, Fine-tune, Tools, and **Knowledge**, which is also highlighted with a red box. The main content area is titled "Foundry IQ" and features sections for "Knowledge bases" and "Indexes". It includes a search bar with three buttons: "Ask AI", "What is a knowledge base?", and "What knowledge types are supported?". Below the search is a large graphic of a brain made of dots and the text "Ground your agent in enterprise knowledge" followed by a subtext: "Connect your agent to Foundry IQ to combine multiple data sources behind an agentic retrieval engine."

4. Select your AI Search resource.



This screenshot shows the "Knowledge" page within the Microsoft Foundry interface. The left sidebar remains the same as the previous screenshot. The main content area is titled "Knowledge" and contains sections for "Knowledge bases" and "Indexes". It features the same search bar with "Ask AI" and three informational buttons. A large graphic of a brain made of dots is present, along with the text "Ground your agent in enterprise knowledge" and its description. Below this, there's a section titled "Connect to an AI Search resource to get started" with a red box around a dropdown menu labeled "Select a resource". To the right of the dropdown is a purple "Connect" button. At the bottom of the page, there's a link "Not sure? Create new resource" with a small icon.

5. Select **Create a knowledge base**.

6. Select **Microsoft OneLake** as the knowledge type. Select **Connect**.

7. Provide your Fabric workspace ID and lakehouse ID.

You can retrieve both of these IDs from your lakehouse URL:

https://app.powerbi.com/groups/<WORKSPACE_ID>/lakehouses/<LAKEHOUSE_ID>.

Create a knowledge source

 Microsoft OneLake
Retrieve from Microsoft OneLake unstructured data

Name *

Description ⓘ
e.g., "October 2025 retail metrics: daily sales, customer patterns, staffing data" (helps your agent know when to use this source)

Fabric workspace ID * ⓘ
Enter Fabric workspace GUID

Lakehouse ID * ⓘ
Enter lakehouse GUID

Content extraction mode ⓘ
minimal

Include embedding model (recommended)

Embedding model * ⓘ
Select model

User-assigned managed identity ⓘ
No identities available on search service

> Show advanced options

Create **Cancel**

8. Select **Create**.

9. Select **Save knowledge base**.

Last updated on 11/18/2025

Manage OneLake with PowerShell

Microsoft OneLake integrates with the Azure PowerShell module for data reading, writing, and management.

Connect to OneLake with Azure PowerShell

Connect to OneLake from PowerShell by following these steps:

1. Install the Azure Storage PowerShell module.

```
PowerShell
```

```
Install-Module Az.Storage -Repository PSGallery -Force
```

2. Sign in to your Azure account.

```
PowerShell
```

```
Connect-AzAccount
```

3. Create the storage account context.

- Storage account name is **onelake**.
- Set `-UseConnectedAccount` to passthrough your Azure credentials.
- Set `-endpoint` as `fabric.microsoft.com`.

4. Run the same commands used for Azure Data Lake Storage (ADLS) Gen2. For more information about ADLS Gen2 and the Azure Storage PowerShell module, see [Use PowerShell to manage ADLS Gen2](#).

Example: Get the size of an item or directory

```
PowerShell
```

```
Install-Module Az.Storage -Repository PSGallery -Force
Connect-AzAccount
$ctx = New-AzStorageContext -StorageAccountName 'onelake' -UseConnectedAccount -
endpoint 'fabric.microsoft.com'

# This example uses the workspace and item name. If the workspace name does not meet
# Azure Storage naming criteria (no special characters), you can use GUIDs instead.
$workspaceName = 'myworkspace'
$itemPath = 'mylakehouse.lakehouse/Files'
```

```
# Recursively get the length of all files within your lakehouse, sum, and convert to GB.  
$colitems = Get-AzDataLakeGen2ChildItem -Context $ctx -FileSystem $workspaceName -  
Path $itemPath -Recurse -FetchProperty | Measure-Object -property Length -sum  
"Total file size: " + ($colitems.sum / 1GB) + " GB"
```

Related content

- [Integrate OneLake with Azure Synapse Analytics](#)
-

Last updated on 11/18/2025

AzCopy

08/28/2025

AzCopy is a powerful command-line utility designed to facilitate the transfer of data between Azure Storage accounts. Because Microsoft OneLake supports the same APIs, SDKs, and tools as Azure Storage, you can also use AzCopy to load data to and from OneLake. This article helps you use AzCopy with OneLake, from copying data between artifacts to uploading or downloading data.

Why use AzCopy and OneLake?

AzCopy is optimized for data plane operations at scale and large scale data movement. When you copy data between storage accounts (including OneLake), data moves directly from storage server to storage server, minimizing performance bottlenecks. AzCopy is also easy-to-use and reliable, with built-in mechanisms to handle network interruptions and retries. With AzCopy, it's easy to upload data to OneLake, or load data from existing sources directly into your items in Fabric!

Trusted workspace access and AzCopy

Trusted workspace access lets you access firewall-enabled Azure Storage accounts securely by configuring a resource instance rule on an Azure Storage account. This rule lets your specific Fabric workspace access the storage account's firewall from select Fabric experiences, like shortcuts, pipelines, and AzCopy. By configuring trusted workspace access, AzCopy can copy data from a firewall-enabled Azure Storage account into OneLake without affecting the firewall protections. Learn more at [trusted workspace access](#).

Getting Started

If you're new to AzCopy, you can learn how to download and get started with AzCopy at [Get started with AzCopy](#).

When you use AzCopy with OneLake, there's a few key points to remember:

1. Add "fabric.microsoft.com" as a trusted domain using the--trusted-microsoft-suffixes parameter.
2. Select the subscription of your source Azure Storage account when logging in with your Microsoft Entra ID, as OneLake only cares about the tenant.
3. Use double quotes when using AzCopy in the command prompt, and single quotes when in PowerShell.

The samples in this article also assume that your Microsoft Entra ID has appropriate permissions to access both the source and destinations.

Finally, you need at least one source and destination for your data movement - the samples in this page use two Fabric lakehouses and one ADLS account.

Sample: Copying data between Fabric workspaces

Use this sample to copy a file from a lakehouse in one workspace to a different workspace by using the [azcopy copy](#) command. Remember to authenticate first by running `azcopy login` first.

Syntax

AzCopy

```
azcopy copy "https://onelake.dfs.fabric.microsoft.com/<source-workspace-name>/<source-item-name>/Files/<source-file-path>"  
"https://onelake.dfs.fabric.microsoft.com/<destination-workspace-name>/<destination-item-name>/Files/<destination-file-path>" --trusted-microsoft-suffixes "fabric.microsoft.com"
```

The copy operation is synchronous so when the command returns, all files are copied.

Sample: Copying data from ADLS to OneLake with a shared access signatures (SAS)

A shared access signature (SAS) provides short-term, delegated access to Azure Storage and OneLake, and is a great option to provide tools or users temporary access to storage for one-time upload or downloads. A SAS is also a great option if the Azure Storage account is in a different tenant than your OneLake, as Entra authorization will not work if the tenants are different.

This sample uses a unique SAS token to authenticate to both Azure Storage and OneLake. To learn more about generating and using SAS tokens with Azure Storage and OneLake, check out the following pages:

- [How to create a OneLake shared access signature \(SAS\)](#)
- [Grant limited access to Azure storage resources using shared access signatures \(SAS\)](#)

Note

When using a SAS token to authenticate to OneLake in AzCopy, you must set the ``-s2s-preserve-access-tier' parameter to false.

AzCopy

```
azcopy copy "https://<account-name>.blob.core.windows.net/<source-container-name>/<source-file-path>?<blob-sas-token>"  
"https://onelake.dfs.fabric.microsoft.com/<destination-workspace-name>/<destination-item-name>/Files/<destination-file-path>?<onelake-sas-token>" -  
-trusted-microsoft-suffixes "fabric.microsoft.com" --s2s-preserve-access-  
tier=false
```

Limitations

Since OneLake is a managed data lake, some operations aren't supported with AzCopy. For example, you can't use AzCopy to move or copy entire items or workspaces. Instead, create the new item in your destination location using a Fabric experience (like the portal), and then use AzCopy to move the contents of the existing item into the new item.

Cross-tenant operations

When attempting to perform operations directly between two Fabric tenants, you must use [external data sharing](#). This means you cannot currently use AzCopy to directly load data between two Fabric tenants, as that results in a direct cross-tenant operation. Other methods to load data, such as downloading the data locally or to a Spark cluster and then re-uploading the data to the new tenant, will function.

Related content

- [Copy blobs between Azure Storage accounts by using AzCopy](#)

Overview of OneLake table APIs

10/27/2025

OneLake offers a REST API endpoint for interacting with tables in Microsoft Fabric. This endpoint can be used with clients and libraries that are compatible with [the Iceberg REST Catalog \(IRC\) API open standard](#) ↗ or the [Unity Catalog API open standard](#) ↗.

 **Important**

This feature is in [preview](#).

Prerequisites

Using these APIs is straightforward once you identify a few pieces of information and select your preferred Microsoft Entra ID authentication flow.

Gathering basic information

To use these APIs, you first need to gather the following pieces of information:

- Your Fabric tenant ID.

The tenant ID is a GUID, and it can be found in the **Profile** card or the **Help, About Fabric** menu in Fabric.

- The workspace and data item ID of the data item (such as a lakehouse) with a top-level Tables directory.

These IDs are GUIDs. They can be found within the OneLake URL of any table in OneLake. They can alternatively be found within the URL seen in your browser when you have a data item open in Fabric.

- The user or service principal identity in Microsoft Entra ID that has permissions to read tables in your chosen data item.

Preparing for authentication

1. Decide how you would like to authenticate with Microsoft Entra ID to obtain an access token for your chosen Microsoft Entra identity.

You can [check this guide](#) to learn about the different ways to obtain an access token with Microsoft Entra ID. Microsoft offers [convenient authentication libraries](#) in several languages.

2. If you are developing a new application that will either allow users to sign in or sign in as a standalone application, [register your application with Microsoft Entra ID](#).
3. [Grant API permission](#) for the Azure Storage (<https://storage.azure.com/>) token audience, to your Microsoft Entra ID application. Granting this permission ensures that your application can obtain tokens for use with the OneLake table endpoint.

 **Note**

The OneLake table API endpoint accepts the same token audience as the OneLake filesystem endpoints.

If you are developing an application, you might already know how to authenticate with Microsoft Entra ID to interact with OneLake filesystem REST APIs. If so, you can use the same approach to authenticate with the new OneLake table endpoint.

Iceberg REST Catalog (IRC) API operations on OneLake

Learn [how to get started with the OneLake table API endpoint](#) to interact with Iceberg tables in [OneLake](#). Initially, read-only metadata table operations are supported, and we plan to add more operations soon.

 **Note**

Before using the Iceberg APIs, be sure you have Delta Lake to Iceberg metadata conversion enabled for your tenant or workspace. Review the [instructions to learn how to enable automatic Delta Lake to Iceberg table format conversion](#).

Delta Lake REST API operations on OneLake

Learn [how to get started with the OneLake table API endpoint](#) to interact with Delta tables in [OneLake](#).

Related content

- Learn more about OneLake table APIs for Iceberg.
- Learn more about OneLake table APIs for Delta.
- Set up [automatic Delta Lake to Iceberg format conversion](#).

OneLake table APIs for Iceberg

10/27/2025

OneLake offers a REST API endpoint for interacting with tables in Microsoft Fabric. This article describes how to get started using this endpoint to interact with Apache Iceberg REST Catalog (IRC) APIs available at this endpoint for metadata read operations.

For overall OneLake table API guidance and prerequisite guidance, see the [OneLake table API overview](#).

For detailed API documentation, see the [Getting started guide](#).

 **Important**

This feature is in [preview](#).

Iceberg table API endpoint

The OneLake table API endpoint is:

```
https://onelake.table.fabric.microsoft.com
```

At the OneLake table API endpoint, the Iceberg REST Catalog (IRC) APIs are available under the following `<BaseUrl>`. You can generally provide this path when initializing existing IRC clients or libraries.

```
https://onelake.table.fabric.microsoft.com/iceberg
```

Examples of IRC client configuration with the OneLake table endpoint are covered in the [Getting started guide](#).

 **Note**

Before using the Iceberg APIs, be sure you have Delta Lake to Iceberg metadata conversion enabled for your tenant or workspace. [See the instructions to learn how to enable automatic Delta Lake to Iceberg metadata conversion](#).

Iceberg table API operations

The following API operations are currently supported at this endpoint. Detailed guidance for these operations is available in the [Getting started guide](#).

- **Get configuration**

```
GET <BaseUrl>/v1/config?warehouse=<Warehouse>
```

This operation accepts the workspace ID and data item ID (or their equivalent friendly names if they don't contain any special characters). `<Warehouse>` is typically `<WorkspaceID>/<dataItemID>`.

This operation returns the `Prefix` string that is used in subsequent requests.

- **List namespaces**

```
GET <BaseUrl>/v1/<Prefix>/namespaces
```

This operation returns the list of schemas within a data item. If the data item doesn't support schemas, a fixed schema named `dbo` is returned.

- **Get namespace**

```
GET <BaseUrl>/v1/<Prefix>/namespaces/<SchemaName>
```

This operation returns information about a schema within a data item, if the schema is found. If the data item doesn't support schemas, a fixed schema named `dbo` is supported here.

- **List tables**

```
GET <BaseUrl>/v1/<Prefix>/namespaces/<SchemaName>/tables
```

This operation returns the list of tables found within a given schema.

- **Get table**

```
GET <BaseUrl>/v1/<Prefix>/namespaces/<SchemaName>/tables/<TableName>
```

This operation returns metadata details for a table within a schema, if the table is found.

Current limitations, considerations

The use of the OneLake table APIs for Iceberg is subject to the following limitations and considerations:

- **Certain data items may not support schemas**

Depending on the type of data item you use, such as non-schema-enabled Fabric lakehouses, there may not be schemas within the Tables directory. In such cases, for compatibility with API clients, the OneLake table APIs provide a default, fixed `dbo` schema (or namespace) to contain all tables within a data item.

- **Current namespace scope**

In Fabric, data items contain a flat list of schemas, which each contains a flat list of tables. Today, the top-level namespaces listed by the Iceberg APIs are schemas, so although the Iceberg REST Catalog (IRC) standard supports multi-level namespaces, the OneLake implementation offers one level, mapping to schemas.

Because of this limitation, we don't yet support the `parent` query parameter for the `list namespaces` operation.

- **Metadata write operations, other operations**

Only the operations listed in [Iceberg table API operations](#) are supported today.

Operations that handle metadata write operations aren't yet supported by the OneLake table API endpoint. We plan to add support for more operations at a later time.

Related content

- Learn more about [OneLake table APIs](#).
- See [detailed guidance and API details](#).
- Set up [automatic Delta Lake to Iceberg format conversion](#).

Getting started with OneLake table APIs for Iceberg

10/27/2025

OneLake offers a REST API endpoint for interacting with tables in Microsoft Fabric. This endpoint supports read-only metadata operations for Apache Iceberg tables in Fabric. These operations are compatible with the [Iceberg REST Catalog \(IRC\) API open standard](#).

 **Important**

This feature is in [preview](#).

Prerequisites

Learn more about [OneLake table APIs for Iceberg](#) and make sure to review the [prerequisite information](#).

Client quickstart examples

Review these samples to learn how to set up existing Iceberg REST Catalog (IRC) clients or libraries for use with the new OneLake table endpoint.

Pylceberg

Use the following sample Python code to configure [Pylceberg](#) to use the OneLake table API endpoint. Then, list schemas and tables within a data item.

This code assumes there is a default AzureCredential available for a currently signed-in user. Alternatively, you can use the [MSAL Python library](#) to obtain a token.

Python

```
from pyiceberg.catalog import load_catalog
from azure.identity import DefaultAzureCredential

# Iceberg base URL at the OneLake table API endpoint
table_api_url = "https://onelake.table.fabric.microsoft.com/iceberg"

# Entra ID token
credential = DefaultAzureCredential()
token = credential.get_token("https://storage.azure.com/.default").token
```

```

# Client configuration options
fabric_workspace_id = "12345678-abcd-4fbd-9e50-3937d8eb1915"
fabric_data_item_id = "98765432-dcba-4209-8ac2-0821c7f8bd91"
warehouse = f"{fabric_workspace_id}/{fabric_data_item_id}"
account_name = "onelake"
account_host = f"{account_name}.blob.fabric.microsoft.com"

# Configure the catalog object for a specific data item
catalog = load_catalog("onelake_catalog", **{
    "uri": table_api_url,
    "token": token,
    "warehouse": warehouse,
    "adls.account-name": account_name,
    "adls.account-host": account_host,
    "adls.credential": credential,
})
# List schemas and tables within a data item
schemas = catalog.list_namespaces()
print(schemas)
for schema in schemas:
    tables = catalog.list_tables(schema)
    print(tables)

```

Snowflake

Use the following sample code to create a new **catalog-linked database** in Snowflake. This database will automatically include any schemas and tables found within the connected Fabric data item. This involves the creation of a [catalog integration](#), an [external volume](#), and a [database](#).

SQL

```

-- Create catalog integration object
CREATE OR REPLACE CATALOG INTEGRATION IRC_CATINT
    CATALOG_SOURCE = ICEBERG_REST
    TABLE_FORMAT = ICEBERG
    REST_CONFIG = (
        CATALOG_URI = 'https://onelake.table.fabric.microsoft.com/iceberg' --
Iceberg base URL at the OneLake table endpoint
        CATALOG_NAME = '12345678-abcd-4fbd-9e50-3937d8eb1915/98765432-dcba-4209-
8ac2-0821c7f8bd91' -- Fabric data item scope, in the form `workspaceID/dataItemID`-
    )
    REST_AUTHENTICATION = (
        TYPE = OAUTH -- Entra auth
        OAUTH_TOKEN_URI = 'https://login.microsoftonline.com/11122233-1122-4138-
8485-a47dc5d60435/oauth2/v2.0/token' -- Entra tenant ID
        OAUTH_CLIENT_ID = '44332211-aabb-4d12-aef5-de09732c24b1' -- Entra
application client ID
        OAUTH_CLIENT_SECRET = '[secret]' -- Entra application client secret value

```

```

        OAUTH_ALLOWED_SCOPES = ('https://storage.azure.com/.default') -- Storage
token audience
)
ENABLED = TRUE
;

-- Create external volume object
CREATE OR REPLACE EXTERNAL VOLUME IRC_EXVOL
    STORAGE_LOCATIONS =
    (
        (
            NAME = 'IRC_EXVOL'
            STORAGE_PROVIDER = 'AZURE'
            STORAGE_BASE_URL = 'azure://onelake.dfs.fabric.microsoft.com/12345678-
abcd-4fbcd-9e50-3937d8eb1915/98765432-dcba-4209-8ac2-0821c7f8bd91'
            AZURE_TENANT_ID='81ba0d80-2361-40bb-9c1f-bf1c84027025' --tenant id
        )
    )
    ALLOW_WRITES = FALSE;
;

-- Describe the external volume
DESC EXTERNAL VOLUME IRC_EXVOL;

```

The response of `DESC EXTERNAL VOLUME` will return metadata about the external volume, including:

- `AZURE_CONSENT_URL`, which is the permissions request page that needs to be followed if it hasn't yet been done for your tenant.
- `AZURE_MULTI_TENANT_APP_NAME`, which is the name of the Snowflake client application that needs access to the data item. Make sure to grant it access to the data item in order for Snowflake to be able to read table contents.

SQL

```

-- Create a Snowflake catalog linked database
CREATE OR REPLACE DATABASE IRC_CATALOG_LINKED
    LINKED_CATALOG =
        (
            CATALOG = 'IRC_CATINT'
        )
    EXTERNAL_VOLUME = 'IRC_EXVOL'
;

SELECT SYSTEM$CATALOG_LINK_STATUS('IRC_CATALOG_LINKED');

SELECT * FROM IRC_CATALOG_LINKED."dbo"."sentiment";

```

DuckDB

Use the following sample Python code to configure DuckDB  to list schemas and tables within a data item.

This code assumes there is a default `AzureCredential` available for a currently signed-in user.

Alternatively, you can use the [MSAL Python library](#) to obtain a token.

Python

```
import duckdb
from azure.identity import DefaultAzureCredential

# Iceberg API base URL at the OneLake table API endpoint
table_api_url = "https://onelake.table.fabric.microsoft.com/iceberg"

# Entra ID token
credential = DefaultAzureCredential()
token = credential.get_token("https://storage.azure.com/.default").token

# Client configuration options
fabric_workspace_id = "12345678-abcd-4fbd-9e50-3937d8eb1915"
fabric_data_item_id = "98765432-dcba-4209-8ac2-0821c7f8bd91"
warehouse = f"{fabric_workspace_id}/{fabric_data_item_id}"

# Connect to DuckDB
con = duckdb.connect()

# Install & load extensions
con.execute("INSTALL iceberg; LOAD iceberg;")
con.execute("INSTALL azure; LOAD azure;")
con.execute("INSTALL httpfs; LOAD httpfs;")

# --- Auth & Catalog ---
# 1) Secret for the Iceberg REST Catalog (use existing bearer token)
con.execute("""
CREATE OR REPLACE SECRET onelake_catalog (
TYPE ICEBERG,
TOKEN ?
);
""", [token])

# 2) Secret for ADLS Gen2 / OneLake filesystem access via Azure extension
#     (access token audience must be https://storage.azure.com; account name is
#     'onelake')
con.execute("""
CREATE OR REPLACE SECRET onelake_storage (
TYPE AZURE,
PROVIDER ACCESS_TOKEN,
ACCESS_TOKEN ?,
ACCOUNT_NAME 'onelake'
);
""", [token])

# 3) Attach the Iceberg REST catalog
```

```
con.execute(f"""
ATTACH '{warehouse}' AS onelake (
TYPE ICEBERG,
SECRET onelake_catalog,
ENDPOINT '{table_api_url}'
);
""")  
  
# --- Explore & Query ---
display(con.execute("SHOW ALL TABLES").fetchhdf())
```

Example requests and responses

These example requests and responses illustrate the use of the Iceberg REST Catalog (IRC) operations currently supported at the OneLake table API endpoint. For more information about IRC, see [the open standard specification ↗](#).

For each of these operations:

- <BaseUrl> is <https://onelake.table.fabric.microsoft.com/iceberg>
- <Warehouse> is <Workspace>/<DataItem>, which can be:
 - <WorkspaceID>/<DataItemID>, such as 12345678-abcd-4fbd-9e50-3937d8eb1915/98765432-dcba-4209-8ac2-0821c7f8bd91
 - <WorkspaceName>/<DataItemName>. <DataItemType>, such as MyWorkspace/MyItem.Lakehouse, as long as both names do not contain special characters.
- <Prefix> is returned by the Get configuration call, and its value is usually the same as <Warehouse>.
- <Token> is the access token value returned by Entra ID upon successful authentication.

Get configuration

List Iceberg catalog configuration settings.

- Request

```
GET <BaseUrl>/v1/config?warehouse=<Warehouse>
Authorization: Bearer <Token>
```

- Response

JSON

```
200 OK
{
  "defaults": {},
  "endpoints": [
    "GET /v1/{prefix}/namespaces",
    "GET /v1/{prefix}/namespaces/{namespace}",
    "HEAD /v1/{prefix}/namespaces/{namespace}",
    "GET /v1/{prefix}/namespaces/{namespace}/tables",
    "GET /v1/{prefix}/namespaces/{namespace}/tables/{table}",
    "HEAD /v1/{prefix}/namespaces/{namespace}/tables/{table}"
  ],
  "overrides": {
    "prefix": "<Prefix>"
  }
}
```

List schemas

List schemas within a Fabric data item.

- Request

```
GET <BaseUrl>/v1/<Prefix>/namespaces
Authorization: Bearer <Token>
```

- Response

JSON

```
200 OK
{
  "namespaces": [
    [
      "dbo"
    ]
  ],
  "next-page-token": null
}
```

Get schema

Get schema details for a given schema.

- o Request

```
GET <BaseUrl>/v1/<Prefix>/namespaces/<SchemaName>
Authorization: Bearer <Token>
```

- o Response

JSON

```
200 OK
{
  "namespace": [
    "dbo"
  ],
  "properties": {
    "location": "d892007b-3216-424a-a339-f3dca61335aa/40ef140a-8542-
4f4c-baf2-0f8127fd59c8/Tables dbo"
  }
}
```

List tables

List tables within a given schema.

- o Request

```
GET <BaseUrl>/v1/<Prefix>/namespaces/<SchemaName>/tables
Authorization: Bearer <Token>
```

- o Response

JSON

```
200 OK
{
  "identifiers": [
    {
      "namespace": [
        "dbo"
      ],
      "name": "DIM_TestTime"
    },
    {
      "namespace": [
```

```
        "dbo"
    ],
    "name": "DIM_TestTable"
}
],
"next-page-token": null
}
```

Get table

Get table details for a given table.

- Request

```
GET <BaseUrl>/v1/<Prefix>/namespaces/<SchemaName>/tables/<TableName>
Authorization: Bearer <Token>
```

- Response

JSON

```
200 OK
{
    "metadata-location":
"abfss://...@onelake.dfs.fabric.microsoft.com/.../Tables/DIM_TestTime/metadata/v3/metadata.json",
    "metadata": {
        "format-version": 2,
        "table-uuid": "...",
        "location":
"abfss://...@onelake.dfs.fabric.microsoft.com/.../Tables/DIM_TestTime",
        "last-sequence-number": 2,
        "last-updated-ms": ...,
        "last-column-id": 4,
        "current-schema-id": 0,
        "schemas": [
            {
                "type": "struct",
                "schema-id": 0,
                "fields": [
                    {
                        "id": 1,
                        "name": "id",
                        "required": false,
                        "type": "int"
                    },
                    {
                        "id": 2,
                        "name": "value",
                        "required": true,
                        "type": "double"
                    }
                ]
            }
        ]
    }
}
```

```
        "name": "name",
        "required": false,
        "type": "string"
    },
    {
        "id": 3,
        "name": "age",
        "required": false,
        "type": "int"
    },
    {
        "id": 4,
        "name": "i",
        "required": false,
        "type": "boolean"
    }
]
},
],
"default-spec-id": 0,
"partition-specs": [
{
    "spec-id": 0,
    "fields": []
},
{
    "last-partition-id": 999,
    "default-sort-order-id": 0,
    "sort-orders": [
{
    "order-id": 0,
    "fields": []
}
],
"properties": {
        "schema.name-mapping.default": "[ \n    \"field-id\" : 1,\n    \"names\" : [ \"id\" ]\n], {\n    \"field-id\" : 2,\n    \"names\" : [ \"name\" ]\n}, {\n    \"field-id\" : 3,\n    \"names\" : [ \"age\" ]\n}, {\n    \"field-id\" : 4,\n    \"names\" : [ \"i\" ]\n} ]",
        "write.metadata.delete-after-commit.enabled": "true",
        "write.data.path": "abfs://...@onelake.dfs.fabric.microsoft.com/.../Tables/DIM_TestTime",
        "XTABLE_METADATA": "{\"lastInstantSynced\":\"...\\\",\\\"instantsToConsiderForNextSync\\\":[],\\\"version\\\":0,\\\"sourceTableFormat\\\":\\\"DELTA\\\",\\\"sourceIdentifier\\\":\\\"3\\\"}",
        "write.parquet.compression-codec": "zstd"
},
"current-snapshot-id": ...,
"refs": {
        "main": {
            "snapshot-id": ...,
            "type": "branch"
        }
},
{
        "name": "name",
        "required": false,
        "type": "string"
    },
    {
        "id": 3,
        "name": "age",
        "required": false,
        "type": "int"
    },
    {
        "id": 4,
        "name": "i",
        "required": false,
        "type": "boolean"
    }
]
},
"last-partition-id": 999,
"partition-count": 1,
"sort-orders": [
{
    "order-id": 0,
    "fields": []
}
],
"table-format": "DELTA"
}
```

```
"snapshots": [
    {
        "sequence-number": 2,
        "snapshot-id": "...",
        "parent-snapshot-id": "...",
        "timestamp-ms": "...",
        "summary": {
            "operation": "overwrite",
            "XTABLE_METADATA": "
{\\"lastInstantSynced\\":\"...\",\\"instantsToConsiderForNextSync\\":
[],\\"version\\":0,\\"sourceTableFormat\\":\"DELTA\\\",\\"sourceIdentifier\\\":\"3\\"
},
            "added-data-files": "1",
            "deleted-data-files": "1",
            "added-records": "1",
            "deleted-records": "1",
            "added-files-size": "2073",
            "removed-files-size": "2046",
            "changed-partition-count": "1",
            "total-records": "6",
            "total-files-size": "4187",
            "total-data-files": "2",
            "total-delete-files": "0",
            "total-position-deletes": "0",
            "total-equality-deletes": "0"
        },
        "manifest-list": [
            "abfss://...@onelake.dfs.fabric.microsoft.com/.../Tables/DIM_TestTime/metadata/snap-....avro",
                "schema-id": 0
            ]
        ],
        "statistics": [],
        "snapshot-log": [
            {
                "timestamp-ms": ...,
                "snapshot-id": ...
            }
        ],
        "metadata-log": [
            {
                "timestamp-ms": ...,
                "metadata-file": "
abfss://...@onelake.dfs.fabric.microsoft.com/.../Tables/DIM_TestTime/metadata/v1.metadata.json"
            },
            {
                "timestamp-ms": ...,
                "metadata-file": "
abfss://...@onelake.dfs.fabric.microsoft.com/.../Tables/DIM_TestTime/metadata/v2.metadata.json"
            }
        ]
    }
}
```

Related content

- Learn more about [OneLake table APIs](#).
- Learn more about [OneLake table APIs for Iceberg](#).
- Set up [automatic Delta Lake to Iceberg format conversion](#).

OneLake table APIs for Delta

10/27/2025

OneLake offers a REST API endpoint for interacting with tables in Microsoft Fabric. This article describes how to get started using this endpoint to interact with Delta APIs available at this endpoint for metadata read operations. These operations are compatible with [Unity Catalog API open standard](#).

For overall OneLake table API guidance and prerequisite guidance, see the [OneLake table API overview](#).

For detailed API documentation, see the [Getting started guide](#).

 **Important**

This feature is in [preview](#).

Delta table API endpoint

The OneLake table API endpoint is:

```
https://onelake.table.fabric.microsoft.com
```

At the OneLake table API endpoint, the Delta APIs are available under the following <BaseUrl>.

```
https://onelake.table.fabric.microsoft.com/delta
```

Delta table API operations

The following Delta API operations are currently supported at this endpoint. Detailed guidance for these operations is available in the [Getting started guide](#).

- List schemas

```
GET <BaseUrl>/<WorkspaceName or WorkspaceID>/<ItemName or ItemID>/api/2.1/unity-catalog/schemas?catalog_name=<ItemName or ItemID>
```

This operation accepts the workspace ID and data item ID (or their equivalent friendly names if they don't contain any special characters).

This operation returns the list of schemas within a data item. If the data item does not support schemas, a fixed schema named `dbo` is returned.

- **List tables**

```
GET <BaseUrl>/<WorkspaceName or WorkspaceID>/<ItemName or ItemID>/api/2.1/unity-catalog/tables?catalog_name=<ItemName or ItemID>&schema_name=<SchemaName>
```

This operation returns the list of tables found within a given schema.

- **Get table**

```
GET <BaseUrl>/<WorkspaceName or WorkspaceID>/<ItemName or ItemID>/api/2.1/unity-catalog/tables/<TableName>
```

This operation returns metadata details for a table within a schema, if the table is found.

- **Schema exists**

```
HEAD <BaseUrl>/<WorkspaceName or WorkspaceID>/<ItemName or ItemID>/api/2.1/unity-catalog/schemas/<SchemaName>
```

This operation checks for the existence of a schema within a data item and returns success if the schema is found.

- **Table exists**

```
HEAD <BaseUrl>/<WorkspaceName or WorkspaceID>/<ItemName or ItemID>/api/2.1/unity-catalog/tables/<TableName>
```

This operation checks for the existence of a table within a schema and returns success if the schema is found.

Current limitations, considerations

The use of the OneLake table APIs for Delta is subject to the following limitations and considerations:

- **Certain data items may not support schemas**

Depending on the type of data item you use, such as non-schema-enabled Fabric lakehouses, there may not be schemas within the Tables directory. In such cases, for

compatibility with API clients, the OneLake table APIs provide a default, fixed `dbo` schema (or namespace) to contain all tables within a data item.

- **Additional query string parameters required if your schema name or table name contains dots**

If your schema or table name contains dots (.) and is included in the URL, you must also provide additional query parameters. For example, when the schema name includes dots, include the `catalog_name` as an additional query parameter in the API call to check whether the schema exists.

- **Metadata write operations, other operations**

Only the operations listed in [Delta table API operations](#) are supported today. Operations that handle metadata write operations are not yet supported by the OneLake table Delta API endpoint.

Related content

- Learn more about [OneLake table APIs](#).
- See [detailed guidance and API details](#).

Getting started with OneLake table APIs for Delta

10/27/2025

OneLake offers a REST API endpoint for interacting with tables in Microsoft Fabric. This endpoint supports read-only metadata operations for Delta tables in Fabric. These operations are compatible with [Unity Catalog API open standard](#).

 **Important**

This feature is in [preview](#).

Example requests and responses

These example requests and responses illustrate the use of the Delta API operations currently supported at the OneLake table API endpoint.

For each of these operations:

- <BaseUrl> is `https://onelake.table.fabric.microsoft.com/delta`
- <Workspace>/DataItem > can be:
 - <WorkspaceID>/<DataItemID>, such as `12345678-abcd-4fbd-9e50-3937d8eb1915/98765432-dcba-4209-8ac2-0821c7f8bd91`
 - <WorkspaceName>/<DataItemName>. <DataItemType>, such as `MyWorkspace/MyItem.Lakehouse`, as long as both names do not contain special characters.
- <Token> is the access token value returned by Microsoft Entra ID upon successful authentication.

List schemas

List schemas within a Fabric data item.

- Request

Bash

```
curl -X GET \
"<BaseUrl>/<Workspace>/testlh.Lakehouse/api/2.1/unity-catalog/schemas?
catalog_name=testlh.Lakehouse" \
```

```
-H "Authorization: Bearer <Token>" \
-H "Content-Type: application/json"
```

- Response

JSON

```
200 OK
{
  "schemas": [
    {
      "name": "dbo",
      "catalog_name": "testlh.Lakehouse",
      "full_name": "testlh.Lakehouse.dbo",
      "created_at": 1759768029062,
      "updated_at": 1759768029062,
      "comment": null,
      "properties": null,
      "owner": null,
      "created_by": null,
      "updated_by": null,
      "schema_id": null
    }
  ],
  "next_page_token": null
}
```

List tables

List tables within a given schema.

- Request

Bash

```
curl -X GET \
  "<BaseUrl>/<Workspace>/testlh.Lakehouse/api/2.1/unity-catalog/tables?
catalog_name=testlh.Lakehouse&schema_name=dbo" \
-H "Authorization: Bearer <Token>" \
-H "Content-Type: application/json"
```

- Response

JSON

```
200 OK
{
  "tables": [
    {
      "
```

```
        "name": "product_table",
        "catalog_name": "testlh.Lakehouse",
        "schema_name": "dbo",
        "table_type": null,
        "data_source_format": "DELTA",
        "columns": null,
        "storage_location":
    "https://onelake.dfs.fabric.microsoft.com/.../.../Tables/product_table",
        "comment": null,
        "properties": null,
        "owner": null,
        "created_at": null,
        "created_by": null,
        "updated_at": null,
        "updated_by": null,
        "table_id": null
    }
],
"next_page_token": null
}
```

Get table

Get table details for a given table.

Request

Bash

```
curl -X GET \
  "<BaseUrl>/<Workspace>/testlh.Lakehouse/api/2.1/unity-
catalog/tables/testlh.Lakehouse.dbo.product_table" \
-H "Authorization: Bearer <Token>" \
-H "Content-Type: application/json"
```

Response

JSON

```
200 OK
{
  "name": "product_table",
  "catalog_name": "testlh.Lakehouse",
  "schema_name": "dbo",
  "table_type": null,
  "data_source_format": "DELTA",
  "columns": [
    {
      "name": "product_id",
      "type_text": null,
```

```
"type_json": null,
"type_name": "string",
"type_precision": 0,
"type_scale": 0,
"type_interval_type": null,
"comment": null,
"partition_index": 0,
"position": 0,
"nullable": true
},
{
  "name": "product_name",
  "type_text": null,
  "type_json": null,
  "type_name": "string",
  "type_precision": 0,
  "type_scale": 0,
  "type_interval_type": null,
  "comment": null,
  "partition_index": 0,
  "position": 1,
  "nullable": true
},
{
  "name": "category",
  "type_text": null,
  "type_json": null,
  "type_name": "string",
  "type_precision": 0,
  "type_scale": 0,
  "type_interval_type": null,
  "comment": null,
  "partition_index": 0,
  "position": 2,
  "nullable": true
},
{
  "name": "brand",
  "type_text": null,
  "type_json": null,
  "type_name": "string",
  "type_precision": 0,
  "type_scale": 0,
  "type_interval_type": null,
  "comment": null,
  "partition_index": 0,
  "position": 3,
  "nullable": true
},
{
  "name": "price",
  "type_text": null,
  "type_json": null,
  "type_name": "double",
  "type_precision": 0,
```

```
        "type_scale": 0,
        "type_interval_type": null,
        "comment": null,
        "partition_index": 0,
        "position": 4,
        "nullable": true
    },
    {
        "name": "launch_date",
        "type_text": null,
        "type_json": null,
        "type_name": "date",
        "type_precision": 0,
        "type_scale": 0,
        "type_interval_type": null,
        "comment": null,
        "partition_index": 0,
        "position": 5,
        "nullable": true
    }
],
"storage_location":
"https://onelake.dfs.fabric.microsoft.com/.../.../Tables/product_table",
"comment": null,
"properties": null,
"owner": null,
"created_at": 1759703452000,
"created_by": null,
"updated_at": 1759703452000,
"updated_by": null,
"table_id": "df2b3038-c21a-429d-90b8-f3bbf2d3db5d"
}
```

Related content

- Learn more about [OneLake table APIs](#).
- Learn more about [OneLake table APIs for Delta](#).

OneLake security overview

09/08/2025

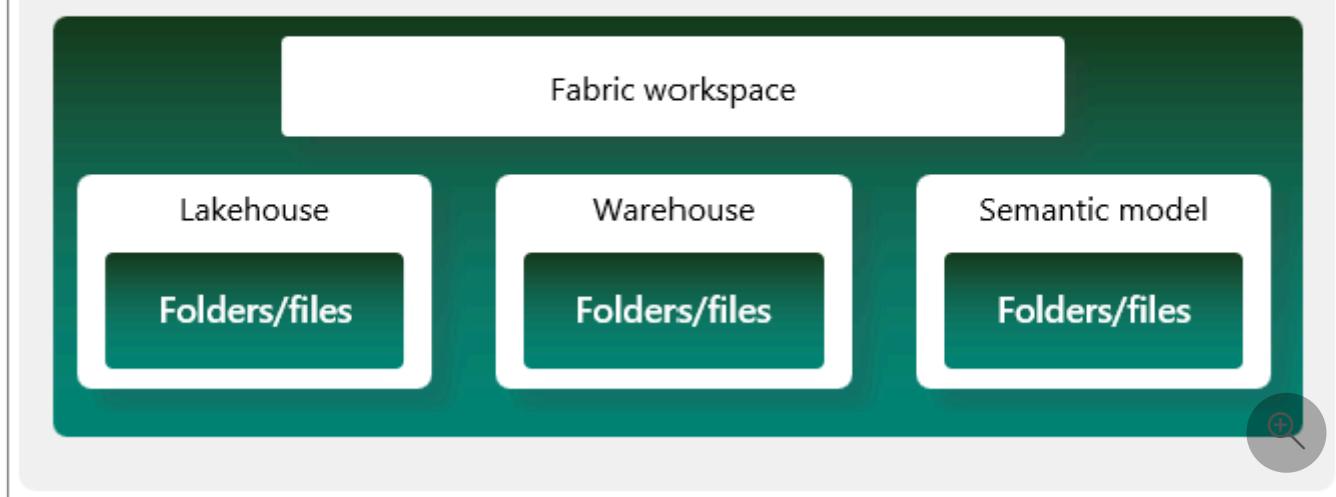
OneLake is a hierarchical data lake, like Azure Data Lake Storage (ADLS) Gen2 or the Windows file system. Security in OneLake is enforced at multiple levels, each corresponding to different aspects of access and control. Understanding the distinction between control plane and data plane permissions is key to effectively securing your data:

- **Control plane permissions:** Govern what actions users can perform within the environment (e.g., creating, managing, or sharing items). Control plane permissions often provide data plane permissions by default.
- **Data plane permissions:** Govern what data users can access or view, regardless of their ability to manage resources.

You can set security at each level within the data lake. However, some levels in the hierarchy are given special treatment because they correlate with Fabric concepts. OneLake security controls all access to OneLake data with different permissions inherited from the parent item or workspace permissions. You can set permissions at the following levels:

- **Workspace:** A collaborative environment for creating and managing items. Security is managed through workspace roles at this level.
- **Item:** A set of capabilities bundled together into a single component. A data item is a subtype of item that allows data to be stored within it using OneLake. Items inherit permissions from the workspace roles, but can have additional permissions as well.
- **Folders:** Folders within an item that are used for storing and managing data, such as Tables/ or Files/.

Items always live within workspaces and workspaces always live directly under the OneLake namespace. You can visualize this structure as follows:



Security in OneLake

This section describes the security model based on generally available OneLake features.

Workspace permissions

Workspace permissions define what actions users can take within a workspace and its items. These permissions are managed at the workspace level and are primarily control plane permissions; they determine administrative and item management capabilities, not direct data access. However, workspace permissions will generally inherit down to the item and folder level to grant data access by default. Workspace permissions allow for defining access to all items within that workspace. There are four different workspace roles, each of which grants different types of access. Below are the default behaviors of each workspace role.

Expand table

| Role | Can add admins? | Can add members? | Can edit OneLake security? | Can write data and create items? | Can read data in OneLake? |
|-------------|-----------------|------------------|----------------------------|----------------------------------|---------------------------|
| Admin | Yes | Yes | Yes | Yes | Yes |
| Member | No | Yes | Yes | Yes | Yes |
| Contributor | No | No | No | Yes | Yes |
| Viewer | No | No | No | No | No* |

(!) Note

*Viewers can be given access to data through OneLake security roles.

You can simplify the management of Fabric workspace roles by assigning them to security groups. This method lets you control access by adding or removing members from the security group.

Item permissions

With the [sharing](#) feature, you can give a user direct access to an item. The user can only see that item in the workspace and isn't a member of any workspace roles. Item permissions grant access to connect to that item and its endpoints the user is able to access.

[+] [Expand table](#)

| Permission | See the item metadata? | See data in SQL? | See data in OneLake? |
|------------|------------------------|------------------|----------------------|
| Read | Yes | No | No |
| ReadData | No | Yes | No |
| ReadAll | No | No | Yes* |

*Not applicable to items with [OneLake security](#) or data access roles enabled. If the preview is enabled, ReadAll only grants access if the DefaultReader role is in use. If the DefaultReader role is edited or deleted, access is instead granted based on what data access roles the user is part of.

Another way to configure permissions is via an item's [Manage permissions](#) page. Using this page, you can add or remove individual item permission for users or groups. The item type determines which permissions are available.

OneLake security (preview)

OneLake security allows users to define granular role-based security to data stored in OneLake, and enforce that security consistently across all compute engines in Fabric. OneLake security is the [data plane](#) security model for data in OneLake.

(!) Note

OneLake security replaces the existing OneLake data access roles (preview) feature that was released in April 2024. All data access roles users are seamlessly and automatically upgraded to OneLake security roles when the feature moved to public preview.

Fabric users in the Admin or Member roles can create OneLake security roles to grant users access to data within an item. Each role has four components:

- **Data**: The tables or folders that users can access.
- **Permission**: The permissions that users have on the data.
- **Members**: The users that are members of the role.
- **Constraints**: The components of the data, if any, that are excluded from role access, such as specific rows or columns.

OneLake security roles grant access to data for users in the **Viewer** workspace role or with **Read** permission on the item. Admins, Members, and Contributors are not affected by OneLake security roles and can read and write all data in an item regardless of their role membership. A DefaultReader role exists in all lakehouses that gives any user with the ReadAll permission access to data in the lakehouse. The DefaultReader role can be deleted or edited to remove that access.

Learn more about creating OneLake security roles for [Tables and folders](#), [Columns](#), and [Rows](#).

Learn more about the [access control model for OneLake security..](#)

Compute permissions

Compute permissions are a type of data plane permission that applies to a specific query engine in Microsoft Fabric. The access granted applies only to queries run against that specific engine, such as the SQL endpoint or a Power BI semantic model. However, users might see different results when they access data through a compute engine compared to when they access data directly in OneLake, depending on the compute permissions applied. **OneLake security is the recommended approach to secure data in OneLake to ensure consistent results across all engines that a user might interact with.**

Compute engines may have more advanced security features that are not yet available in OneLake security, and in that case using the compute permissions may be required to solve some scenarios. When using compute permissions to secure access to data, make sure that end users are given access only to the compute engine where the security is set. This prevents data from being accessed through a different engine without the necessary security features.

Shortcut security

Shortcuts in Microsoft Fabric allow for simplified data management. OneLake folder security applies to OneLake shortcuts based on roles defined in the lakehouse where the data is stored.

For more information on shortcut security considerations, see [OneLake security access control model](#).

For information on access and authentication details for specific shortcuts, see [types of OneLake shortcuts](#).

Authentication

OneLake uses Microsoft Entra ID for authentication; you can use it to give permissions to user identities and service principals. OneLake automatically extracts the user identity from tools, which use Microsoft Entra authentication and maps it to the permissions you set in the Fabric portal.

 **Note**

To use service principals in a Fabric tenant, a tenant administrator must enable Service Principal Names (SPNs) for the entire tenant or specific security groups. Learn more about enabling Service Principals in [Developer settings of the tenant admin portal](#).

Audit Logs

To view your OneLake audit logs, follow the instructions in [Track user activities in Microsoft Fabric](#). OneLake operation names correspond to [ADLS APIs](#) such as CreateFile or DeleteFile. OneLake audit logs don't include read requests or requests made to OneLake via Fabric workloads.

Encryption and networking

Data at Rest

Data stored in OneLake is encrypted at rest by default using Microsoft-managed keys. Microsoft-managed keys are rotated appropriately. Data in OneLake is encrypted and decrypted transparently and is FIPS 140-2 compliant.

Encryption at rest using customer-managed keys currently isn't supported. You can submit a request for this feature on [Microsoft Fabric ideas](#).

Data in transit

Data in transit across the public internet between Microsoft services is always encrypted with at least TLS 1.2. Fabric negotiates to TLS 1.3 whenever possible. Traffic between Microsoft services always routes over the Microsoft global network.

Inbound OneLake communication also enforces TLS 1.2 and negotiates to TLS 1.3 whenever possible. Outbound Fabric communication to customer-owned infrastructure prefers secure protocols but might fall back to older, insecure protocols (including TLS 1.0) when newer protocols aren't supported.

Private links

To configure private links in Fabric, see [Set up and use private links](#).

Allow apps running outside of Fabric to access data via OneLake

You can allow or restrict access to OneLake data from applications that are outside of the Fabric environment. Admins can find this setting in the [OneLake section of the admin portal tenant settings](#).

When you turn on this setting, users can access data from all sources. For example, turn this setting on if you have custom applications that use Azure Data Lake Storage (ADLS) APIs or OneLake file explorer. When you turn off this setting, users can still access data from internal apps like Spark, Data Engineering, and Data Warehouse, but can't access data from applications running outside of Fabric environments.

Related content

- [Fabric and OneLake security overview](#)
- [OneLake data access roles \(preview\)](#)
- [Workspace roles](#)
- [OneLake file explorer](#)
- [Share items](#)

Get started with OneLake security (preview)

OneLake security enables you to apply role-based access control (RBAC) to your data stored in OneLake. You can define security roles that grant access to specific folders within a Fabric item, then assign these roles to users or groups. Roles can also contain row or column level security to further limit access. The OneLake security permissions determine what data that user can see across all experiences in Fabric.

Fabric users with Write and Reshare permissions (generally Admin and Member workspace users) can get started by creating OneLake security roles to grant access to only specific folders or tables in a Fabric data item. To grant access to data in an item, add users to a data access role. Users that aren't part of a data access role see no data in that item.

Prerequisites

To configure OneLake security, you must be an Admin or Member in the workspace, or have Write and Reshare permissions. Role creation and membership assignment take effect as soon as the role is saved, so make sure you want to grant access before adding someone to a role.

The following table outlines which data items support OneLake security:

 Expand table

| Fabric item | Status | Supported permissions |
|-----------------------------------|---------|-----------------------|
| Lakehouse | Preview | Read, ReadWrite |
| Azure Databricks Mirrored Catalog | Preview | Read |

How to opt in

OneLake security is currently in preview and as a result is disabled by default. The preview feature is configured on a per-item basis. The opt-in control allows for a single item to try the preview without enabling it on any other Fabric items.

The preview feature can't be turned off once enabled.

1. Navigate to a lakehouse and select **Manage OneLake security (preview)**.
2. Review the confirmation dialog. The data access roles preview isn't compatible with the External data sharing preview. If you're ok with the change, select **Continue**.

To ensure a smooth opt-in experience, all users with read permission to data in the item continue to have read access through a default data access role called **DefaultReader**. With

[virtualized role memberships](#), all users that had the necessary permissions to view data in the lakehouse (the ReadAll permission) are included as members of this default role. To start restricting access to those users, delete the DefaultReader role or remove the ReadAll permission from the accessing users.

 **Important**

Make sure that any users that are included in a data access role are removed from the DefaultReader role. Otherwise they maintain full access to the data.

What types of data can be secured?

Use OneLake security roles to manage OneLake read access to any tables or folders in an item. Access to tables can be further restricted using row and/or column level security. Any security set applies to access from all engines in Fabric. For more information, see the [data access control model](#).

For specific item types, ReadWrite access can also be configured. This permission gives users the ability to edit data in a lakehouse on specified tables or folders without giving them access to create or manage Fabric items. ReadWrite access enables users to perform write operations through Spark notebooks, the OneLake file explorer, or OneLake APIs. Write operations through the Lakehouse UX for viewers is not supported.

Create a role

Use the following steps to create a OneLake security role.

1. Open the Fabric item where you want to define security.
2. Select **Manage OneLake security (preview)** from the item menu.
3. On the **OneLake security (preview)** pane, select **New**.
4. Provide a name for the new role that meets the following guidelines:
 - The role name can only contain alphanumeric characters.
 - The role name must start with a letter.
 - Names are case insensitive and must be unique.
 - The maximum name length is 128 characters.
5. Select **Grant** as the type of role.

6. Choose the permissions you want to grant. **Read** is selected at a minimum, and you can optionally choose **ReadWrite**.
7. If you want this role to apply to all of the tables and files in this lakehouse, select the **All data** toggle.

This selection also provides access to any folders that are added in the future.
8. If you want this role to apply only to a selected group of tables and folders, select the **Selected data** toggle. Then, use the following steps to define the approved data for this role.
 - a. Select **Browse Lakehouse** or the equivalent for the item that you're working with.

All Lakehouse roles > **New role** X

New role

Create roles to grant specific groups of users access to selected data in your Lakehouse. [Learn more](#) ↗

Role name *

 H

Add data to your role *

All data

Selected data

Browse Lakehouse →

Add members to your role

 ✓ X

[Advanced Configuration](#)

- b. Expand the **Tables** and **Files** directories to view data in your lakehouse.
 - c. Check the boxes next to the tables and files that you want the role to apply to.
 - d. Select **Add data** to add the selected items to your role.
9. Use the **Add members to your role** textbox to manually enter the names or email addresses of users that you want to include in the role. Or, select **Advanced configuration** and follow the guidance in [Assign virtual members](#).

To add members manually:

- a. Enter the name or email address of a user.
 - b. Select the correct name from the suggested list.
 - c. Select the check icon to confirm your selection, or the X icon to clear the selection.
10. Review the **Preview role** summaries.
- a. To edit the data preview, select **Browse Lakehouse** and update the selected tables and folders.
 - b. To remove a user from the members preview, select more options (...) next to their name, then **Remove from role**.
11. Select **Create role** and wait for the notification that the role was successfully published.

Edit a role

Use the following steps to edit an existing OneLake security role.

1. Open the item where you want to define security.
2. Select **Manage OneLake security (preview)** from the item menu.
3. On the **OneLake security (preview)** pane, select the role that you want to edit.

This action opens the role details page, which includes two tabs: **Data in role** and **Members in role**.

4. Review the information in the **Data in role** tab:

This tab shows all of the data that the members of the role can access.

The role name tells you which role you are looking at. To edit the role name, select the **Edit** dropdown in the upper right corner, select **Update role name**, enter a new name, and then confirm with the check mark. You can discard your changes by selecting the X.

The **Permissions** item at the top, tells you what permissions the role currently grants. To change the role permissions, select the **Edit** dropdown in the upper right corner, select **Edit role permissions**, edit the selected permissions with the dropdown, and then confirm with the check mark. You can discard your changes by selecting the X.

The **Data** column shows the name of the tables or folders that are part of the role access. You can expand and collapse schemas to view the items underneath. Hover over an entry to view the full path of the table or folder. Hover over the ... to see options to configure **Row-level security** or **Column-level security**. The [row level security](#) and [column level security](#) guides provide more information on how that works.

The **Type** column tells you the type of item that was selected. The values are either: **Schema**, **Table**, or **Folder**.

The **Data access** column indicates whether any row or column level restrictions are applied to the item. An icon with a lock and horizontal lines indicates row level security is applied, while an icon with a lock and vertical lines indicates column level security is applied.

5. To edit the data included in the role, select **Add data**.

This action opens the table and folder selection dialog.

6. Check and uncheck tables or folders to add or remove them from the role.

7. Select **Add data** to confirm your selections.

8. Select the Members in role tab to view the members of the role.

The **Members** column shows the profile picture and name of the member.

The **Type** column indicates whether the member is a User or Group.

The **Added using** column denotes whether a user was added via their Email as a member of the role, or included as part of a lakehouse permissions group. For more information about adding users using item permissions, see [Assign virtual members](#).

9. To edit the members of the role, select **Add members**.

10. To add members manually, enter a name or email in the **Add members to your role** textbox. Select the correct name from the suggested list. Then, select the check icon to confirm your selection, or select the X icon to clear the selection.

11. To remove users from the role, select more options (...) next to their name and select **Remove from role**.

Making any changes to role membership updates the role immediately. A notification notes the success or failure of any changes.

Delete a role

Use the following steps to delete a OneLake data access role.

1. Open the lakehouse where you want to define security.
2. Select **Manage OneLake security (preview)** from the Lakehouse menu.

3. On the **OneLake security (preview)** pane, check the box next to the roles you want to delete.
4. Select **Delete** and wait for the notification that the roles are successfully deleted.

Assign a member or group

OneLake security role supports two methods of adding users to a role. The main method is by adding users or groups directly to a role using the **Add people or groups** box on the **Assign role** page. The second is by creating virtual memberships with permission groups using the **Advanced configuration** control.

Adding users directly to a role adds the users as explicit members of the role. These users show up with their name and picture shown in the **Members** list.

The virtual members allow for the membership of the role to be dynamically adjusted based on the [Fabric item permissions](#) of the users. By selecting **Advanced configuration** and selecting a permission, you add any user in the Fabric workspace who has all of the selected permissions as an implicit member of the role. For example, if you chose **ReadAll, Write** then any user of the Fabric workspace that has ReadAll *and* Write permissions to the item would be included as a member of the role. You can see which users are being added by a permission group by looking at the **Added using** column in the **Members in role** tab. These members can't be manually removed directly. To remove a member that was added through a permission group, remove the permission group from the role.

Regardless of which membership type you use, OneLake security roles support adding individual users, Microsoft Entra groups, and security principals.

Assign virtual members

The permissions that can be used for virtual members are:

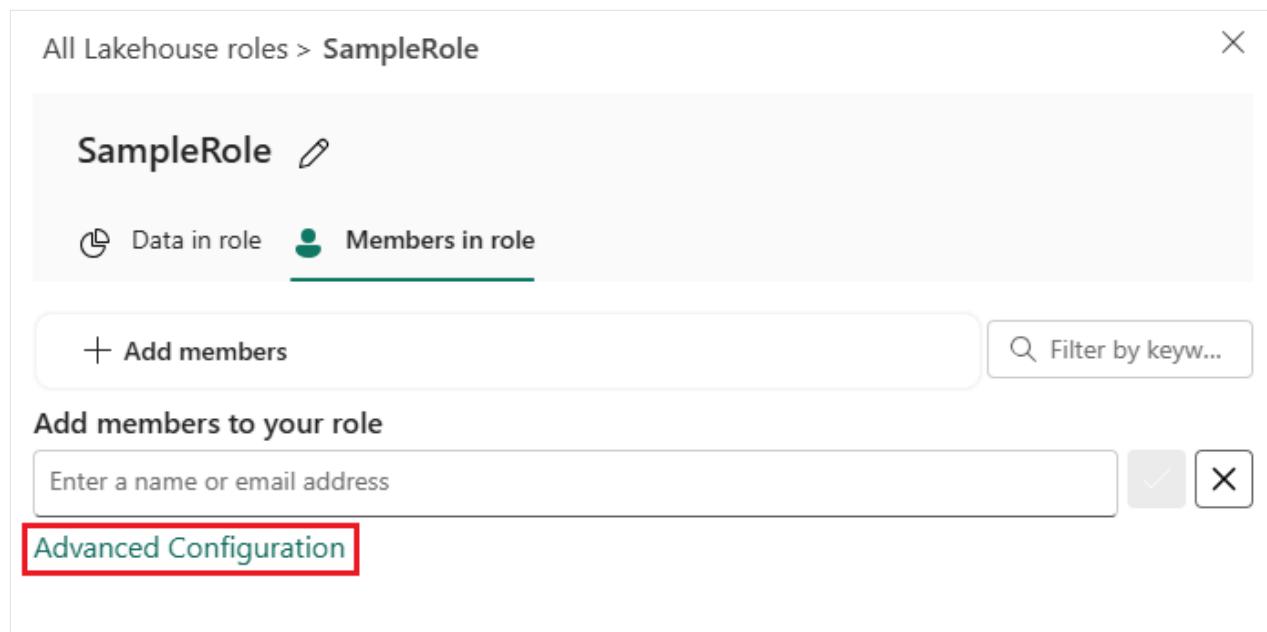
- Read
- Write
- Reshare
- Execute
- ReadAll

To assign users with permission groups, use the following steps:

1. Select the name of the role you want to assign members to.
2. On the role details page, select the **Members in role** tab.

3. Select **Add members**.

4. Select **Advanced configuration**.



5. In the **Permission groups** box, select the checkbox next to each permission that you want to include users for.

Each permission group shows a count of how many users are included in that group.

Selecting multiple permission groups includes users with all of the selected required permissions.

6. Select **Add** to include the groups and save the role.

Related content

- [Fabric Security overview](#)
- [Fabric and OneLake security overview](#)
- [Data access control model](#)

Last updated on 11/18/2025

Best practices for OneLake security

09/08/2025

In this article, we'll look at best practices around securing data in OneLake. For more information on how to implement security for specific use cases, see the how-to guides.

Least privilege

Least privilege access is a fundamental security principle in computer science that advocates for restricting users' permissions and access rights to only those permissions necessary to perform their tasks. For OneLake, this means assigning permissions at the appropriate level to ensure that users aren't over-provisioned and reduce risk.

- If users only need access to a single lakehouse or data item, use the share feature to grant them access to only that item. Assigning a user to a workspace role should only be used if that user needs to see ALL items in that workspace.
- Use [OneLake security](#) to restrict access to folders and tables within a lakehouse. For sensitive data, OneLake security [row](#) or [column](#) level security ensures that protected row and columns remain hidden.

Secure by use case

Different users need the ability to perform different actions in Fabric in order to do their jobs. Some common use cases are identified in this section along with the necessary permissions setup in Fabric and OneLake.

Manage workspace access The admin or member workspace roles are required. These roles can also manage OneLake security roles on an item.

Create new items in Fabric Either Admin, Member, or Contributor roles can create or delete new items.

Write data to OneLake Either Admin, Member, or Contributor roles can write data to OneLake through Spark or through uploads. They can also write data to a warehouse. Users with only read access on a warehouse can be given permissions to write data through [SQL permissions](#).

Read data from OneLake A user needs to be a workspace Viewer, or have the Read permission and the ReadAll permission to read data from OneLake. For lakehouses with the OneLake security (preview) feature enabled, access to data is controlled by the user's OneLake security role permissions.

Subscribe to OneLake events A user needs `SubscribeOneLakeEvents` to be able to subscribe to events from a Fabric item. Admin, Member, and Contributor roles have this permission by default. You can add this permission for a user with Viewer role.

Related content

- [Fabric Security overview](#)
- [Fabric and OneLake security overview](#)
- [Data Access Control Model](#)

OneLake security access control model (preview)

This document provides a detailed guide to how the OneLake security access control model works. It contains details on how the roles are structured, how they apply to data, and what the integration is with other structures within Microsoft Fabric.

OneLake security roles

OneLake security uses a role based access control (RBAC) model for managing access to data in OneLake. Each role is made up of several key components.

- **Type:** Whether the role gives access (GRANT) or removes access (DENY). Only GRANT type roles are supported.
- **Permission:** The specific action or actions that are being granted or denied.
- **Scope:** The OneLake objects that have the permission. Objects are tables, folders, or schemas.
- **Members:** Any Microsoft Entra identity that is assigned to the role, such as users, groups, or nonuser identities. The role is granted to all members of a Microsoft Entra group.

By assigning a member to a role, that user is then subject to the associated permissions on the scope of that role. Because OneLake security uses a deny-by-default model, all users start with no access to data unless explicitly granted by a OneLake security role.

Permissions and supported items

OneLake security roles support the following permission:

- **Read:** Grants the user the ability to read data from a table and view the associated table and column metadata. In SQL terms, this permission is equivalent to both VIEW_DEFINITION and SELECT. For more information, see the [Metadata security](#).
- **ReadWrite:** Grants the user the ability to read and write data from a table or folder and view the associated table and column metadata. In SQL terms, this permission is equivalent to ALTER, DROP, UPDATE, and INSERT. For more information see [ReadWrite permission](#).

OneLake security enables users to define data access roles for the following Fabric items only.

| Fabric item | Status | Supported permissions |
|-----------------------------------|----------------|-----------------------|
| Lakehouse | Public Preview | Read, ReadWrite |
| Azure Databricks Mirrored Catalog | Public Preview | Read |

OneLake security and workspace permissions

Workspace permissions are the first security boundary for data within OneLake. Each workspace represents a single domain or project area where teams can collaborate on data. You manage security in the workspace through Fabric workspace roles. Learn more about Fabric role-based access control (RBAC): [Workspace roles](#)

Fabric workspace roles give permissions that apply to all items in the workspace. The following table outlines the basic permissions allowed by workspace roles.

[\[+\] Expand table](#)

| Permission | Admin | Member | Contributor | Viewer |
|---------------------------------|----------------|----------------|-------------|--|
| View files in OneLake | Always* Yes | Always* Yes | Always* Yes | No by default. Use OneLake security to grant the access. |
| Write files in OneLake | Always* Yes | Always* Yes | Always* Yes | No |
| Can edit OneLake security roles | Always* Yes | Always* Yes | No | No |

*Since Workspace Admin, Member and Contributor roles automatically grant Write permissions to OneLake, they override any OneLake security Read permissions.

Workspace roles manage the control plane data access, meaning interactions with creating and managing Fabric artifacts and permissions. In addition, workspace roles also provide default access levels to data items by using OneLake security default roles. (Note that default roles only apply to Viewers, since Admin, Member, and Contributor have elevated access through the Write permission) A default role is a normal OneLake security role that is created automatically with every new item. It gives users with certain workspace or item permissions a default level of access to data in that item. For example, Lakehouse items have a DefaultReader role that lets users with the ReadAll permission see data in the Lakehouse. This ensures that users accessing a newly created item have a basic level of access. All default roles use a member virtualization feature, so that the members of the role are any user in that workspace with the required permission. For example, all users with ReadAll permission on the Lakehouse.

The following table shows what the standard default roles are. Items may have specialized default roles that apply only to that item type.

[] Expand table

| Fabric item | Role name | Permission | Folders included | Assigned members |
|-------------|------------------|------------|--|-----------------------------------|
| Lakehouse | DefaultReader | Read | All folders under <code>Tables/</code> and <code>Files/</code> | All users with ReadAll permission |
| Lakehouse | DefaultReadWrite | Read | All folders | All users with Write permission |

(!) Note

To restrict the access to specific users or specific folders, either modify the default role or remove it and create a new custom role.

OneLake security and item permissions

Within a workspace, Fabric items can have permissions configured separately from the workspace roles. You can configure permissions either through sharing an item or by managing the permissions of an item. The following permissions determine a user's ability to perform actions on data in OneLake. For more information on item sharing, see [How Lakehouse sharing works](#)

[] Expand table

| Permission | Can view files in OneLake? | Can write files in OneLake? | Can read data through SQL analytics endpoint? |
|-------------------------------|--|-------------------------------|---|
| Read | No by default. Use OneLake security to grant access. | No | No |
| ReadAll | Yes through the DefaultReader role. Use OneLake security to restrict access. | No | No* |
| Write | Yes | Yes | Yes |
| Execute, Reshare, ViewOutput, | N/A - can't be granted on its own | N/A - can't be granted on its | N/A - can't be granted on its own |

| Permission | Can view files in OneLake? | Can write files in OneLake? | Can read data through SQL analytics endpoint? |
|------------|----------------------------|-----------------------------|---|
| ViewLogs | | own | |

*Depends on the [SQL analytics endpoint mode](#).

Create roles

You can define and manage OneLake security roles through your lakehouse data access settings.

Learn more in [Get started with data access roles](#).

Engine and user access to data

Data access to OneLake occurs in one of two ways:

- Through a Fabric query engine or
- Through user access (Queries from non-Fabric engines are considered user access)

OneLake security ensures that data is always kept secure. Because certain OneLake security features like row and column level security aren't supported by storage level operations, not all types of access to row or column level secured data can be permitted. This guarantees that users can't see rows or columns they aren't permitted to. Microsoft Fabric engines are enabled to apply row and column level security filtering to data queries. This means when a user queries data in a lakehouse or other item with OneLake security RLS or CLS on it, the results the user sees have the hidden rows and columns removed. For user access to data in OneLake with RLS or CLS on it, the query is blocked if the user requesting access isn't permitted to see all the rows or columns in that table.

The table below outlines which Microsoft Fabric engines support RLS and CLS filtering.

[] [Expand table](#)

| Engine | RLS/CLS filtering | Status |
|--|-------------------|----------------|
| Lakehouse | Yes | Public preview |
| Spark notebooks | Yes | Public preview |
| SQL Analytics Endpoint in "user's identity mode" | Yes | Public preview |

| Engine | RLS/CLS filtering | Status |
|--|-------------------|----------------|
| Semantic models using DirectLake on OneLake mode | Yes | Public preview |
| Eventhouse | No | Planned |
| Data warehouse external tables | No | Planned |

OneLake security access control model details

This section provides details on how OneLake security roles grant access to specific scopes, how that access operates, and how access is resolved across multiple roles and access types.

Table level security

All OneLake tables are represented by folders in the lake, but not all folders in the lake are tables from the perspective of OneLake security and query engines in Fabric. To be considered a valid table, the following conditions must be met:

- The folder exists in the Tables/ directory of an item.
- The folder contains a _delta_log folder with corresponding JSON files for the table metadata.
- The folder does not contain any child shortcuts.

Any tables that do not meet those criteria will have access denied if table level security is configured on them.

Metadata security

OneLake security's Read access to data grants full access to the data and metadata in a table. For users with no access to a table, the data is never exposed and generally the metadata isn't visible. This also applies to column level security and a user's ability to see or not see a column in that table. However, OneLake security doesn't guarantee that the **metadata** for a table won't be accessible, specifically in the following cases:

- SQL Endpoint queries: SQL Analytics Endpoint uses the same metadata security behavior as SQL Server. This means that if a user doesn't have access to a table or column, the error message for that query will explicitly state the table or column names the user doesn't have access to.
- Semantic models: Giving a user Build permission on a semantic model allows them access to see the table names included in the model, regardless of whether the user has access

to them or not. In addition, report visuals that contain hidden columns show the column name in the error message.

Permission inheritance

For any given folder, OneLake security permissions always inherit to the entire hierarchy of the folder's files and subfolders.

For example, consider the following hierarchy of a lakehouse in OneLake:

Bash

```
Tables/  
      — (empty folder)  
Files/  
      — folder1  
          |   file11.txt  
          |  
          |   — subfolder11  
          |       |   file1111.txt  
          |       |  
          |       |   — subfolder111  
          |           |       file11111.txt  
          |  
          |  
          |   — folder2  
          |       |   file21.txt
```

You create two roles for this lakehouse. `Role1` grants read permission to `folder1`, and `Role2` grants read permission to `folder2`.

For the given hierarchy, OneLake security permissions for `Role1` and `Role2` inherit in the following way:

- `Role1`: Read `folder1`

Bash

```
      |   file11.txt  
      |  
      |   — subfolder11  
      |       |   file1111.txt  
      |       |  
      |       |   — subfolder111  
      |           |       file11111.txt
```

- `Role2`: Read `folder2`

Bash

```
|   file21.txt
```

Traversal and listing in OneLake security

OneLake security provides automatic traversal of parent items to ensure that data is easy to discover. Granting a user Read permissions to subfolder11 grants the user the ability to list and traverse the parent directory folder1. This functionality is similar to Windows folder permissions where giving access to a subfolder provides discovery and traversal for the parent directories. The list and traversal granted to the parent doesn't extend to other items outside of the direct parents, ensuring other folders are kept secure.

For example, consider the following hierarchy of a lakehouse in OneLake.

Bash

```
Tables/
—— (empty folder)
Files/
—— folder1
|   |   file11.txt
|   |
|   —— subfolder11
|   |   file111.txt
|   |
|   —— subfolder111
|       |   file1111.txt
|
—— folder2
|   |   file21.txt
```

For the given hierarchy, OneLake security permissions for 'Role1' provides the following access. Access to file11.txt isn't visible as it isn't a parent of subfolder11. Likewise for Role2, file111.txt isn't visible either.

- Role1: Read subfolder11

Bash

```
Files/
—— folder1
|   |
|   —— subfolder11
|       |   file111.txt
|   |
|   —— subfolder111
|       |   file1111.txt
```

- Role2: Read subfolder111

Bash

```
Files/
  └── folder1
      └── subfolder11
          └── subfolder111
              └── file1111.txt
```

For shortcuts, the listing behavior is slightly different. Shortcuts to external data sources behave the same as folders do, however shortcuts to other OneLake locations have specialized behavior. The target permissions of the shortcut determine access to a OneLake shortcut. When listing shortcuts, no call is made to check the target access. As a result, when listing a directory all internal shortcuts are returned regardless of a user's access to the target. When a user tries to open the shortcut, the access check evaluates and a user only sees data that they have the required permissions to see. For more information on shortcuts, see the [shortcuts security section](#).

Consider the following folder hierarchy that contains shortcuts.

Bash

```
Files/
  └── folder1
      └── shortcut2
      └── shortcut3
```

- Role1: Read folder1

Bash

```
Files/
  └── folder1
      └── shortcut2
      └── shortcut3
```

- Role2: No permissions defined

Bash

```
Files/
  |
  └── shortcut2
  |
  └── shortcut3
```

Row level security

OneLake security allows users to specify row level security by writing SQL predicates to limit what data is shown to a user. RLS operates by showing rows where the predicate evaluates to true. For more information, see the [row level security](#).

Row level security evaluates string data as case insensitive, using the following collation for sorting and comparisons: *Latin1_General_100_CI_AS_KS_WS_SC_UTF8*

When using row level security, ensure that the RLS statements are clean and easy to understand. Use integer columns for sorting and greater than or less than operations. Avoid string equivalencies if you don't know the format of the input data, especially in relation to unicode characters or accent sensitivity.

Column level security

OneLake security supports limiting access to columns by removing (hiding) a user's access to a column. A hidden column is treated as having no permissions assigned to it, resulting in the default policy of no access. Hidden columns won't be visible to users, and queries on data containing hidden columns return no data for that column. As noted in [metadata security](#) there are certain case where the metadata of a column might still be visible in some error messages.

Column level security also follows a more strict behavior in SQL Endpoint by operating through a deny semantic. Deny on a column in SQL Endpoint ensures that all access to the column is blocked, even if multiple roles would combine to give access to it. As a result, CLS in SQL Endpoint operates using an intersection between all roles a user is part of instead of the union behavior in place for all other permission types. See the Evaluating multiple OneLake security roles section for more information on how roles combine.

ReadWrite permission

The ReadWrite permission gives read-only users the ability to perform write operations to specific items. ReadWrite permission is only applicable for Viewers or users with the Read permission on an item. Assigning ReadWrite access to an Admin, Member, or Contributor has no effect as those roles already have that permission implicitly.

ReadWrite access enables users to perform write operations through Spark notebooks, the OneLake file explorer, or OneLake APIs. Write operations through the Lakehouse UX for viewers is not supported.

The ReadWrite permission operates in the following ways:

- The ReadWrite permission includes all privileges granted by the Read permission.
- Users with ReadWrite permissions on an object can perform write operations on that object, inclusive. That is, any operations can also be performed on the object itself.
- ReadWrite allows the following actions:
 - Create a new folder or table
 - Delete a folder or table
 - Rename a folder or table
 - Upload or edit a file
 - Create a shortcut
 - Delete a shortcut
 - Rename a shortcut
- OneLake security roles with ReadWrite access cannot contain RLS or CLS constraints.
- Because Fabric only supports single engine writes to data, users with ReadWrite permission on an object can only Write to that data through OneLake. However, the Read operations will be enforced consistently through all querying engines.

Shortcuts

Shortcuts overview

OneLake security integrates with shortcuts in OneLake to ensure data inside and outside of OneLake can be easily secured. There are two main authentication modes for shortcuts:

- Passthrough shortcuts (SSO): The credential of the querying user is evaluated against the shortcut target to determine what data is allowed to be seen.
- Delegated shortcuts: The shortcut uses a fixed credential to access the target and the querying user is evaluated against OneLake security prior to checking the delegated credential's access to the source.

In addition, OneLake security permissions are evaluated when creating any shortcuts in OneLake. Read about shortcut permissions in the [shortcut security document](#).

OneLake security in passthrough shortcuts

Security set on a OneLake folder always flows across any [internal shortcuts](#) to restrict access to the shortcut source path. When a user accesses data through a shortcut to another OneLake location, the identity of the calling user is used to authorize access to the data in the target path of the shortcut. As a result, this user must have OneLake security permissions in the target location to read the data.

ⓘ Important

When accessing shortcuts through **Power BI semantic models using DirectLake over SQL or T-SQL engines in Delegated identity mode**, the calling user's identity isn't passed through to the shortcut target. The calling item owner's identity is passed instead, delegating access to the calling user. To resolve this, use **Power BI semantic models in DirectLake over OneLake mode or T-SQL in User's identity mode**.

Defining OneLake security permissions for the internal shortcut isn't allowed and must be defined on the target folder located in the target item. The target item must be an item type that supports OneLake security roles. If the target item doesn't support OneLake security, the user's access is evaluated based on whether they have the Fabric ReadAll permission on the target item. Users don't need Fabric Read permission on an item in order to access it through a shortcut.

OneLake security in delegated shortcuts

OneLake supports defining permissions for shortcuts such as [ADLS, S3, and Dataverse shortcuts](#). In this case, the permissions are applied on top of the delegated authorization model enabled for this type of shortcut.

Suppose user1 creates an S3 shortcut in a lakehouse pointing to a folder in an AWS S3 bucket. Then user2 is attempting to access data in this shortcut.

[] [Expand table](#)

| Does S3 connection authorize access for the delegated user1? | Does OneLake security authorize access for the requesting user2? | Result: Can user2 access data in S3 Shortcut? |
|--|--|---|
| Yes | Yes | Yes |
| No | No | No |
| No | Yes | No |
| Yes | No | No |

The OneLake security permissions can be defined either for the entire scope of the shortcut or for selected subfolders. Permissions set on a folder inherit recursively to all subfolders, even if the subfolder is within the shortcut. Security set on an external shortcut can be scoped to grant access either to the entire shortcut, or any subpath inside the shortcut. Another internal shortcut pointing to an external shortcut still requires the user to have access to the original external shortcut.

Unlike other types of access in OneLake security, a user accessing an external shortcut requires Fabric Read permission on the data item where the external shortcut resides. This is necessary for securely resolving the connection to the external system.

Learn more about S3, ADLS, and Dataverse shortcuts in [OneLake shortcuts](#).

Evaluating multiple OneLake security roles

Users can be members of multiple different OneLake security roles, each one providing its own access to data. The combination of these roles together is called the "effective role" and is what a user will see when accessing data in OneLake. Roles combine in OneLake security using a UNION or least-restrictive model. This means if Role1 gives access to TableA, and Role2 gives access to TableB, then the user will be able to see both TableA and TableB.

OneLake security roles also contain row and column level security, which limits access to the rows and columns of a table. Each RLS and CLS policy exists within a role and limits access to data for all users within that single role. For example, if Role1 gives access to Table1, but has RLS on Table1 and only shows some columns of Table1 then the effective role for Role1 is going to be the RLS and CLS subsets of Table1. This can be expressed as $(R1ols \cap R1cls \cap R1rls)$ where \cap is the INTERSECTION of each component in the role.

When dealing with multiple roles, RLS and CLS combine with a UNION semantic on the respective tables. CLS is a direct set UNION of the tables visible in each role. RLS is combined across predicates using an OR operator. For example, WHERE city = 'Redmond' OR city = 'New York'.

To evaluate multiple roles each with RLS or CLS, each role is first resolved based on the access given by the role itself. This means evaluating the INTERSECTION of all object, row, and column level security. Each evaluated role is then combined with all other roles a user is a member of via the UNION operation. The output is the effective role for that user. This can be expressed as:

```
( (R1ols \cap R1cls \cap R1rls) \cup (R2ols \cap R2cls \cap R2rls) )
```

Lastly, each shortcut in a lakehouse generates a set of inferred roles that are used to propagate the shortcut target's permissions to the item being queried. Inferred roles operate in a similar

way to noninferred roles except they're resolved first in place on the shortcut target before being combined with roles in the shortcut lakehouse. This ensures that any inheritance of permissions on the shortcut lakehouse is broken and the inferred roles are evaluated correctly. The full combination logic can then be expressed as:

```
( (R1ols n R1cls n R1rls) u (R2ols n R2cls n R2rls) ) n ( (R1'ols n R1'cls n R1'rls) u  
(R2'ols n R2'cls n R2'rls)) )
```

Where R1' and R2' are the inferred roles and R1 and R2 are the shortcut lakehouse roles.

 **Important**

If two roles combine such that the columns and rows aren't aligned across the queries, access is blocked to ensure that no data is leaked to the end user.

OneLake security limitations

- If you assign a OneLake security role to a B2B guest user, you must [configure your external collaboration settings for B2B in Microsoft Entra External ID](#). The **Guest user access** setting must be set to **Guest users have the same access as members (most inclusive)**.
- OneLake security doesn't support cross-region shortcuts. Any attempts to access shortcut to data across different capacity regions result in 404 errors.
- If you add a distribution list to a role in OneLake security, the SQL endpoint can't resolve the members of the list to enforce access. The result is that users appear not to be members of the role when they access the SQL endpoint. DirectLake on SQL semantic models are subject to this limitation too.
- To query data from a Spark notebook using Spark SQL, the user must have at least Viewer access in the workspace they're querying.
- Mixed-mode queries are not supported. Single queries that access both OneLake security enabled and non-OneLake security enabled data will fail with query errors.
- Spark notebooks require that the environment be 3.5 or higher and using Fabric runtime 1.3.
- OneLake security doesn't work with [private link protection](#).
- The [external data sharing preview](#) feature isn't compatible with the data access roles preview. When you enable the data access roles preview on a lakehouse, any existing

external data shares might stop working.

- Azure Mirrored Databricks Catalog does not support Manage Catalog functionality if OneLake security is enabled on that item. This functionality is coming in November, 2025.
- The following table provides the limitations of OneLake data access roles.

 Expand table

| Scenario | Limit |
|--|-----------------------------------|
| Maximum number of OneLake security roles per Fabric Item | 250 roles per lakehouse |
| Maximum number of members per OneLake security role | 500 users or user groups per role |
| Maximum number of permissions per OneLake security role | 500 permissions per role |

Latencies in OneLake security

- Changes to role definitions take about 5 minutes to apply.
- Changes to a user group in a OneLake security role take about an hour for OneLake to apply the role's permissions on the updated user group.
 - Some Fabric engines have their own caching layer, so might require an extra hour to update access in all systems.

Last updated on 11/18/2025

Table and folder security in OneLake (preview)

10/16/2025

Table-level and folder-level security, or object level security (OLS), is a feature of OneLake security (preview) that lets you grant access to specific tables or folders in a data item. Using OLS you create permissions for both structured and unstructured data at the folder level.

Prerequisites

- An item in Fabric with OneLake security turned on. For more information, see [Get started with OneLake security](#).
- Switch the SQL analytics endpoint on the lakehouse to **User's identity** mode through the **Security** tab.
- For creating semantic models, use the steps to create a [DirectLake model](#).
- For a full list of limitations, see the [known limitations section](#).

Define security rules

You can define object-level security on any folder within a data item. Because delta-parquet tables in OneLake are represented as folders, security can also be configured on tables. Likewise, schemas are also folders and can be secured similarly.

Use the following steps to define security roles for tables or folders.

1. Navigate to your Lakehouse and select **Manage OneLake security (preview)**.
2. Select an existing role that you want to define table or folder security for, or select **New** to create a new role.
3. On the role details page, select **Add data**. This action opens the data browsing experience.

SampleRole 

 Data in role  Members in role

+ Add data  Filter by keyw...

| Data | Type | Permissions | Data access |
|--|--------|-------------|-------------|
| Tables | | | |
|  contoso_data | Table | Read | |
| Files | | | |
|  MyFolder | Folder | Read | |
|  contoso_data | Folder | Read | |

4. Expand the **Tables** or **Files** directories to browse to the items you want to include in the role.
 - For tables, you can expand schemas to choose individual tables.
 - For files, you can expand any number of folders to identify the right items.
5. Select the checkbox next to the items you want to grant access to. You can select up to 500 items per role.
6. Once you have made your selection, select **Add data** to save your changes and return to the data in role page



Select data to add to your role

X



- ▼ ContosoLakehouse
 - ▼ Tables
 - contoso_data
 - ▼ Files
 - > MyFolder
 - > contoso_data

Add data

Cancel

Your changes to the role are saved automatically.

Column-level security in OneLake (preview)

09/08/2025

Column-level security (CLS) is a feature of [OneLake security \(preview\)](#) that allows you to have access to selected columns in a table instead of full access to the table. CLS lets you specify a subset of tables that users can access. Any columns that are removed from the list aren't visible to users.

Prerequisites

- An item in OneLake with OneLake security turned on. For more information, see [Get started with OneLake data access roles](#).
- Switch the SQL analytics endpoint on the lakehouse to **User's identity** mode through the **Security** tab.
- For creating semantic models, use the steps to create a [DirectLake model](#).
- For a full list of limitations, see the [known limitations section](#).

Enforce column-level security

OneLake security CLS gets enforced in one of the following two ways:

- **Filtered tables in Fabric engines:** Queries to the Fabric engines, like Spark notebooks, result in the user seeing only the columns they're allowed to see per the CLS rules.
- **Blocked access to tables:** Tables with CLS rules applied to them can't be read outside of supported Fabric engines.

For filtered tables, the following behaviors apply:

- CLS rules don't restrict access to users with the Admin, Member, and Contributor roles.
- If the CLS rule has a mismatch with the table it's defined on, the query fails and no columns are returned. For example, if CLS is defined for a column that isn't part of the table.
- Queries of CLS tables fail with an error if a user is part of two different roles and one of the roles has row-level security (RLS).
- CLS rules can only be enforced for Delta parquet table objects.
 - CLS rules that are applied to non-Delta table objects block access to the entire table for members of the role.
- If a user runs a `select *` query on a table where they only have access to some of the columns, CLS rules behave differently depending on the Fabric engine.
 - Spark notebooks: The query succeeds and only shows the allowed columns.

- SQL analytics Endpoint: Column access is blocked for the columns the user can't access.
- Semantic models: Column access is blocked for the columns the user can't access.

Define column-level security rules

You can define column-level security as part of a OneLake security role for any Delta-parquet table in the **Tables** section of an item. CLS is always specified for a table and is either enabled or disabled. By default, CLS is disabled and users have access to all columns. Users can enable CLS and remove columns from the list to revoke access.

 **Important**

Removing access to a column doesn't deny access to that column if another role grants access to it.

Use the following steps to define column-level security:

1. Navigate to your data item and select **Manage OneLake security (preview)**.
2. Select an existing role that you want to define table or folder security for, or select **New** to create a new role.
3. On the role details page, select more options (...) next to the table you want to define CLS for, then select **Column security (preview)**.

The screenshot shows the 'SampleRole' configuration page in the Azure portal. The 'Data in role' tab is selected. A context menu is open over a table named 'contoso_data', showing options: 'Table', 'Row security (Preview)', and 'Column security (Preview)'. The 'Column security (Preview)' option is highlighted with a red box.

| Data | Type | Permissions | Data access |
|--------------|--------|-------------|-------------|
| Tables | | | |
| contoso_data | Table | Read | |
| Files | | | |
| MyFolder | Folder | Read | |
| contoso_data | Folder | Read | |

4. By default, CLS for a table is disabled. Select **Enable CLS** or create a **New rule** to enable it.

The UI populates with a list of columns for that table that the users are allowed to see. By default, it shows all of the columns.

5. To restrict access to a column, select the checkbox next to the column name, then select **Remove**. At least one column must remain in the list of allowed columns.

6. Select **Save** to update the role.

7. If you want to add a removed column, select **New rule**. This action adds a new CLS rule entry to the end of the list. Then, use the dropdown to choose the column you want to include in the access.

8. Once you complete your changes, select **Save**.

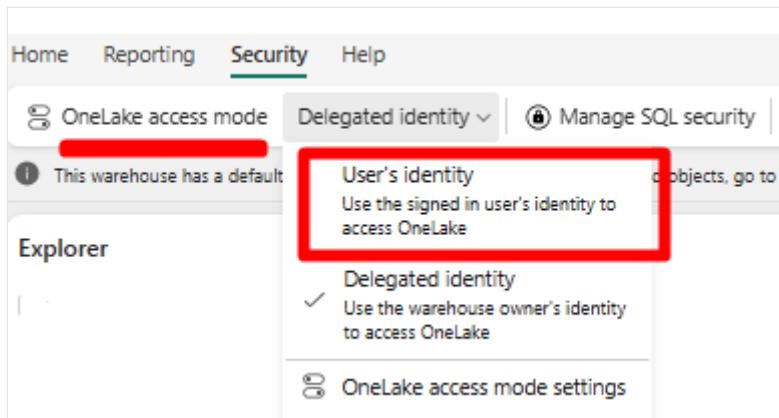
Enable OneLake security for SQL analytics endpoint

Before you can use OneLake security with SQL analytics endpoint, you must enable its **User's identity mode**. Newly created SQL analytics endpoints will default to user's identity mode, so these steps must be followed for existing SQL analytics endpoints.

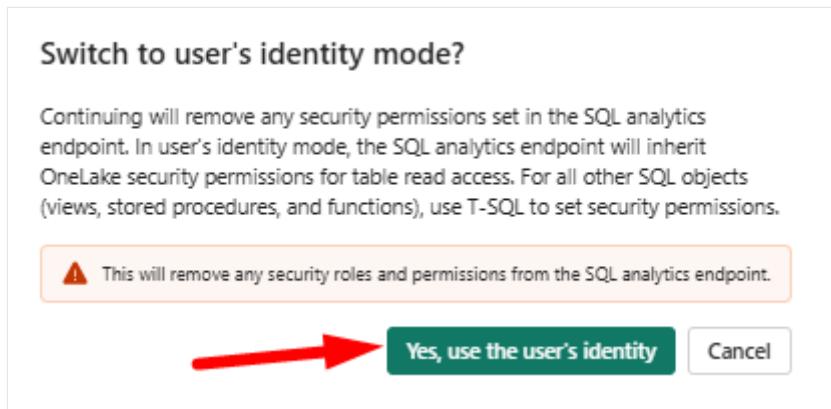
! Note

Switching to **User's identity** mode only needs to be done once per SQL analytics endpoint. Endpoints that are not switched to user's identity mode will continue to use a delegated identity to evaluate permissions.

1. Navigate to SQL analytics endpoint.
2. In the SQL analytics endpoint experience, select the **Security** tab in the top ribbon.
3. Select **User's identity** under **OneLake access mode**.



4. In the prompt, select **Yes, use the user's identity**.



Now the SQL analytics endpoint is ready to use with OneLake security.

Combine row-level and column-level security

Row-level and column-level security can be used together to restrict user access to a table. However, the two policies have to be applied using a single OneLake security role. In this scenario, access to data is restricted according to the rules that are set in the one role.

OneLake security doesn't support the combination of two or more roles where one contains RLS rules and another contains CLS rules. Users that try to access tables that are part of an

unsupported role combination receive query errors.

Row-level security in OneLake (preview)

09/08/2025

Row-level security (RLS) is a feature of OneLake security (preview) that allows for defining row-level data restrictions for tabular data stored in OneLake. Users can define roles in OneLake that contain rules for filtering rows of data for members of that role. When a member of an RLS role goes to query that data, the RLS rules are evaluated and only allowed rows are returned.

Prerequisites

- An item in OneLake with OneLake security turned on. For more information, see [Get started with OneLake data access roles](#).
- Switch the SQL Analytics Endpoint on the lakehouse to "User's identity" mode through the **Security** tab.
- For creating semantic models, use the steps to create a [DirectLake model](#).
- For a full list of limitations, see the [known limitations section](#).

Enforce row-level security

OneLake security RLS gets enforced in one of two ways:

- **Filtered tables in Fabric engines:** Queries to the list of supported Fabric engines, like Spark notebooks, result in the user seeing only the rows they're allowed to see per the RLS rules.
- **Blocked access to tables:** Tables with RLS rules applied to them can't be read outside of supported Fabric engines.

For filtered tables, the following behaviors apply:

- RLS rules don't restrict access for users in the Admin, Member, and Contributor roles.
- If the RLS rule has a mismatch with the table it's defined on, the query fails and no rows are returned. For example, if the RLS rule references a column that isn't part of the table.
- Queries of RLS tables fail with an error if a user is part of two different roles and one of the roles has column-level security (CLS).
- RLS rules can only be enforced for objects that are Delta parquet tables.
 - RLS rules that are applied to non-Delta table objects instead block access to the entire table for members of the role.
- Access to a table might be blocked if the RLS statement contains syntax errors that prevent it from being evaluated.

Define row-level security rules

You can define row-level security rules as part of any OneLake security role that grants access to table data in Delta parquet format. Rows are a concept only relevant to tabular data, so RLS definitions aren't allowed for non-table folders or unstructured data.

RLS rules use SQL syntax to specify the rows that a user can see. This syntax takes the form of a SQL `SELECT` statement with the RLS rules defined in the `WHERE` clause. RLS rules only support a subset of the SQL language as defined in [Syntax rules](#). Queries with invalid RLS syntax or RLS syntax that doesn't match the underlying table result in no rows being shown to users, or query errors in the SQL analytics endpoint.

As a best practice, avoid using vague or overly complex RLS expressions. Strongly-typed expressions with integer or string lookups with "`=`" will be the most secure and easy to understand.

Use the following steps to define RLS rules:

1. Navigate to your Lakehouse and select **Manage OneLake security (preview)**.
2. Select an existing role that you want to define table or folder security for, or select **New** to create a new role.
3. On the role details page, select more options (...) next to the table you want to define RLS for, then select **Row security (preview)**.

The screenshot shows the 'Data in role' tab for the 'SampleRole'. It lists two entries under 'Tables': 'contoso_data' (Table, Read) and 'MyFolder' (Folder, Read). Under 'Files', there is another 'contoso_data' entry (Folder, Read). A context menu is open over the first 'contoso_data' entry, with 'Row security (Preview)' highlighted.

| | Type | Permissions | Data access |
|--------------|--------|-------------|-------------|
| contoso_data | Table | Read | |
| MyFolder | Folder | Read | |
| contoso_data | Folder | Read | |

4. Type the SQL statement for defining which rows you want users to see in the code editor.
Use the [Syntax rules](#) section for guidance.
5. Select **Save** to confirm the row security rules.

Enable OneLake security for SQL analytics endpoint

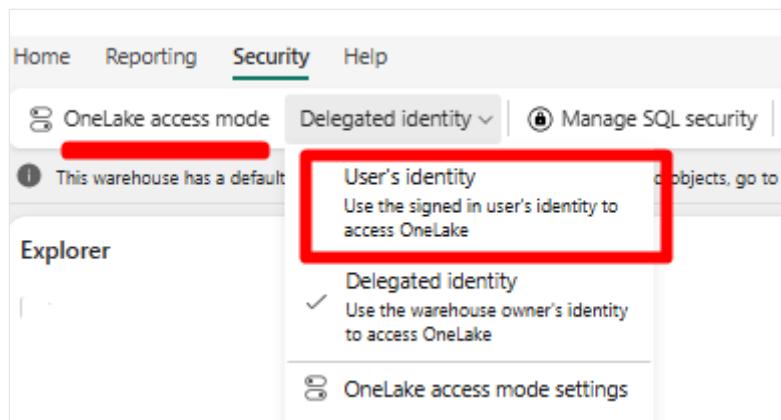
Before you can use OneLake security with SQL analytics endpoint, you must enable its **User's identity mode**. Newly created SQL analytics endpoints will default to user's identity mode, so these steps must be followed for existing SQL analytics endpoints.

Note

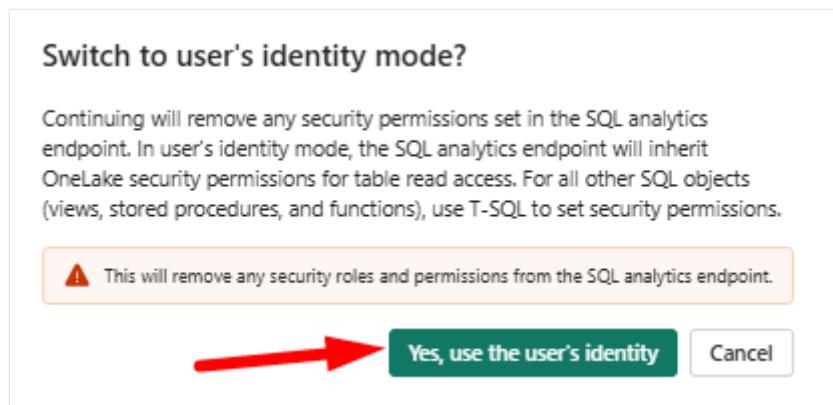
Switching to **User's identity mode** only needs to be done once per SQL analytics endpoint. Endpoints that are not switched to user's identity mode will continue to use a delegated identity to evaluate permissions.

1. Navigate to SQL analytics endpoint.
2. In the SQL analytics endpoint experience, select the **Security** tab in the top ribbon.

3. Select User's identity under OneLake access mode.



4. In the prompt, select Yes, use the user's identity.



Now the SQL analytics endpoint is ready to use with OneLake security.

Syntax rules

All row-level security rules take the following form:

```
SELECT * FROM {schema_name}.{table_name} WHERE {column_level_boolean_1}  
{column_level_boolean_2}...{column_level_boolean_N}
```

For example:

```
SELECT * FROM Sales WHERE Amount>'50000' AND State='CA'
```

The maximum number of characters in a row-level security rule is 1000.

[] Expand table

| Placeholder | Description |
|---------------|---|
| {schema_name} | The name of the schema where {table_name} is located. If the artifact supports schemas, then {schema_name} is required. |

| Placeholder | Description |
|------------------------|--|
| {table_name} | The name of the table that the RLS predicate gets applied to. This value must be an exact match with the name of the table, or the RLS results in no rows being shown. |
| {column_level_boolean} | A Boolean statement containing the following components: <ul style="list-style-type: none"> * Column name: The name of a column in {table_name} as specified in the Delta log schema. Column names can be formatted either as {column_name} or {table_name}.{column_name}. * Operator: One of the Supported operators that evaluates the column name and value to a Boolean value. * Value: A static value or set of values to be evaluated against. <p>You can have one or more Boolean statements separated by AND or OR.</p> |

Supported operators

Row-level security rules support the following list of operators and keywords:

 Expand table

| Operator | Description |
|-------------------------------|--|
| = (equals) | Evaluates to true if the two values are the same data type and exact matches. |
| <> (not equals) | Evaluates to true if the two values aren't the same data type and not exact matches. |
| > (greater than) | Evaluates to true if the column value is greater than the evaluation value. For string values, this operator uses bitwise comparison to determine if one string is greater than the other. |
| >= (greater than or equal to) | Evaluates to true if the column value is greater than or equal to the evaluation value. For string values, this operator uses bitwise comparison to determine if one string is greater than or equal to the other. |
| < (less than) | Evaluates to true if the column value is less than the evaluation value. For string values, this operator uses bitwise comparison to determine if one string is less than the other. |
| <= (less than or equal to) | Evaluates to true if the column value is less than or equal to the evaluation value. For string values, this operator uses bitwise comparison to determine if one string is less than or equal to the other. |
| IN | Evaluates to true if any of the evaluation values are the same data type and exactly match the column value. |

| Operator | Description |
|----------|--|
| NOT | Evaluates to true if any of the evaluation values aren't the same data type or not an exact match of the column value. |
| AND | Combines the previous statement and the subsequent statement using a Boolean AND operation. Both statements must be true for the entire predicate to be true. |
| OR | Combines the previous statement and the subsequent statement using a Boolean OR operation. One of the statements must be true for the entire predicate to be true. |
| TRUE | The Boolean expression for true. |
| FALSE | The Boolean expression for false. |
| BLANK | The blank data type, which can be used with the IS operator. For example, <code>row IS BLANK</code> . |
| NULL | The null data type, which can be used with the IS operator. For example, <code>row IS NULL</code> . |

Combine row-level and column-level security

Row-level and column-level security can be used together to restrict user access to a table. However, the two policies have to be applied using a single OneLake security role. In this scenario, access to data is restricted according to the rules that are set in the one role.

OneLake security doesn't support the combination of two or more roles where one contains RLS rules and another contains CLS rules. Users that try to access tables that are part of an unsupported role combination receive query errors.

Customer-managed keys for Fabric workspaces

Microsoft Fabric encrypts all data-at-rest using Microsoft managed keys. With customer-managed keys for Fabric workspaces, you can use your [Azure Key Vault](#) keys to add another layer of protection to the data in your Microsoft Fabric workspaces - including all data in OneLake. A customer-managed key provides greater flexibility, allowing you to manage its rotation, control access, and usage auditing. It also helps organizations meet data governance needs and comply with data protection and encryption standards.

How customer-managed keys work

All Fabric data stores are encrypted at rest with Microsoft-managed keys. Customer-managed keys use envelope encryption, where a Key Encryption Key (KEK) encrypts a Data Encryption Key (DEK). When using customer-managed keys, the Microsoft managed DEK encrypts your data, and then the DEK is encrypted using your customer-managed KEK. Use of a KEK that never leaves Key Vault allows the data encryption keys themselves to be encrypted and controlled. This ensures that all customer content in a CMK enabled workspace is encrypted using your customer-managed keys.

Enable encryption with customer-managed keys for your workspace

Workspace admins can set up encryption using CMK at the workspace level. Once the workspace admin enables the setting in the portal, all customer content stored in that workspace is encrypted using the specified CMK. CMK integrates with AKV's access policies and role-based access control (RBAC), allowing you flexibility to define granular permissions based on your organization's security model. If you choose to disable CMK encryption later, the workspace will revert to using Microsoft-managed keys. You can also revoke the key at any time and access to the encrypted data will be blocked within an hour of revocation. With workspace level granularity and control, you elevate the security of your data in Fabric.

Supported items

Customer-managed keys are currently supported for the following Fabric items:

- Lakehouse
- Warehouse
- Notebook

- Environment
- Spark Job Definition
- API for GraphQL
- ML model
- Experiment
- Pipeline
- Dataflow
- Industry solutions
- SQL Database (preview)

This feature can't be enabled for a workspace that contains unsupported items. When customer-managed key encryption for a Fabric workspace is enabled, only supported items can be created in that workspace. To use unsupported items, create them in a different workspace that does not have this feature enabled.

Configure encryption with customer-managed keys for your workspace

Customer-managed key for Fabric workspaces requires an initial setup. This setup includes enabling the Fabric encryption tenant setting, configuring Azure Key Vault, and granting the Fabric Platform CMK app access to Azure Key Vault. Once the setup is complete, a user with an [admin workspace role](#) can enable the feature on the workspace.

Step 1: Enable the Fabric tenant setting

A [Fabric administrator](#) needs to make sure that the [Apply customer-managed keys](#) setting is enabled. For more information, see [Encryption tenant setting](#) article.

Step 2: Create a Service Principal for the Fabric Platform CMK app

Fabric uses the *Fabric Platform CMK* app to access your Azure Key Vault. For the app to work, a [service principal](#) must be created for the tenant. This process is performed by a user that has Microsoft Entra ID privileges, such as a [Cloud Application Administrator](#).

Follow the instructions in [Create an enterprise application from a multitenant application in Microsoft Entra ID](#) to create a service principal for an application called *Fabric Platform CMK* with app ID *61d6811f-7544-4e75-a1e6-1c59c0383311* in your Microsoft Entra ID tenant.

Step 3: Configure Azure Key Vault

You need to configure your Key Vault so that Fabric can access it. This step is performed by a user that has Key Vault privileges, such as a [Key Vault Administrator](#). For more information, see Azure [Security roles](#).

1. Open the Azure portal and navigate to your Key Vault. If you don't have Key Vault, follow the instructions in [Create a key vault using the Azure portal](#).
2. In your Key Vault, configure the following settings:
 - [Soft delete](#) - Enabled
 - [Purge protection](#) - Enabled
3. In your Key Vault, open **Access control (IAM)**.
4. From the **Add** dropdown, select **Add Role assignment**.
5. Select the **Members** tab and then click on **Select members**.
6. In the *Select members pane*, search for **Fabric Platform CMK**
7. Select the *Fabric Platform CMK* app and then **Select**.
8. Select the **Role** tab and search for [Key Vault Crypto Service Encryption User](#) or a role that enables *get, wrapkey, and unwrap key* permissions.
9. Select **Key Vault Crypto Service Encryption User**.
10. Select **Review + assign** and then select **Review + assign** to confirm your choice.

Step 4: Create an Azure Key Vault key

To create an Azure Key Vault key, follow the instructions in [Create a key vault using the Azure portal](#).

Key Vault requirements

Fabric only supports [versionless customer-managed keys](#), which are keys in the format

`https://{{vault-name}}.vault.azure.net/{{key-type}}/{{key-name}}` for Vaults and `https://{{hsm-name}}.managedhsm.azure.net/{{key-type}}/{{key-name}}` for Managed HSM. Fabric checks the key vault daily for a new version, and uses the latest version available. To avoid having a period where you can't access data in the workspace after a new key is created, wait 24 hours before disabling the older version.

Key Vault and Managed HSM must have both soft-delete and purge protection enabled and the key must be of RSA or RSA-HSM type. The supported key sizes are:

- 2,048 bit
- 3,072 bit
- 4,096 bit

For more information, see [About keys](#).

 **Note**

4,096 bit keys are not supported for SQL database in Microsoft Fabric.

You can also use Azure Key Vaults for which the [firewall setting is enabled](#). When you disable public access to the Key Vault, you can choose the option to 'Allow Trusted Microsoft Services to bypass this firewall.'

Step 5: Enable encryption using customer-managed keys

After completing the prerequisites, follow the steps in this section to enable customer-managed keys in your Fabric workspace.

1. From your Fabric workspace, select **Workspace settings**.
2. From the *Workspace settings* pane, select **Encryption**.
3. Enable **Apply customer-managed keys**.
4. In the **Key identifier** field, enter your customer-managed key identifier.
5. Select **Apply**.

Once you complete these steps, your workspace is encrypted with a customer-managed key. This means that all data in Onelake is encrypted and that existing and future items in the workspace will be encrypted by the customer-managed key you used for the setup. You can review the encryption status *Active*, *In progress* or *Failed* in the **Encryption** tab in workspace settings. Items for which encryption is in progress or failed are listed categorically too. The key needs to remain active in the Key Vault while encryption is in progress (*Status: In progress*). Refresh the page to view the latest encryption status. If encryption has failed for some items in the workspace, you can retry using a different key.

Revoke access

To revoke access to data in a workspace that's encrypted using a customer-managed key, revoke the key in the Azure Key Vault. Within 60 minutes from the time the key is revoked, read and write calls to the workspace fail.

You can revoke a customer-managed encryption key by changing the access policy, by changing the permissions on the key vault, or by deleting the key.

To reinstate access, restore access to the customer-managed key in the Key Vault.

 **Note**

The workspace does not automatically revalidate the key for SQL Database in Microsoft Fabric. Instead, the user must [manually revalidate](#) the CMK to restore access.

Disable the encryption

To disable encrypting the workspace using a customer-managed key, go to *Workspace settings* disable **Apply customer-managed keys**. The workspace remains encrypted using Microsoft Managed keys.

 **Note**

You can't disable customer-managed keys while encryption for any of the Fabric items in your workspace is in progress.

Monitoring

You can track encryption configuration requests for your Fabric workspaces by audit log entries. The following operation names are used in audit logs:

- ApplyWorkspaceEncryption
- DisableWorkspaceEncryption
- GetWorkspaceEncryption

Considerations and limitations

Before you configure your Fabric workspace with a customer-managed key, consider the following limitations:

- The data listed below isn't protected with customer-managed keys:
 - Lakehouse column names, table format, table compression.
 - All data stored in the Spark Clusters (data stored in temp discs as part of shuffle or data spills or RDD caches in a spark application) are not protected. This includes all the

Spark Jobs from Notebooks, Lakehouses, Spark Job Definitions, Lakehouse Table Load and Maintenance jobs, Shortcut Transforms, Fabric Materialized View Refresh.

- The job logs stored in the history server
 - Libraries attached as part of environments or added as part of the Spark session customization using magic commands are not protected
 - Metadata generated when creating a Pipeline and Copy job, such as DB name, table, schema
 - Metadata of ML model and experiment, like the model name, version, metrics
 - Warehouse queries on Object Explored and backend cache, which is evicted after each use
- CMK is supported on all [F SKUs](#). Trial capacities cannot be used for encryption using CMK. CMK cannot be enabled for workspaces that have BYOK enabled and CMK workspaces cannot be moved to capacities for which BYOK is enabled either.
 - CMK can be enabled using the Fabric portal and does not have API support.
 - CMK can be enabled and disabled for the workspace while the tenant level encryption setting is on. Once the tenant setting is turned off, you can no longer enable CMK for workspaces in that tenant or disable CMK for workspaces that already have CMK turned on in that tenant. Data in workspaces that enabled CMK before the tenant setting was turned off will remain encrypted with the customer managed key. Keep the associated key active to be able to unwrap data in that workspace.

Related content

- [Security fundamentals](#)
- [Microsoft Fabric licenses](#)

Last updated on 11/18/2025

What is a OneLake shared access signature (SAS)?

Article • 04/11/2025

A OneLake shared access signature (SAS) provides secure, short-term, and delegated access to resources in your OneLake. With a OneLake SAS, you have granular control over how a client can access your data. For example:

- What resources the client can access.
- What permissions they have to the resources.
- How long the SAS is valid.

Every OneLake SAS (and user delegation key) is always backed by a Microsoft Entra identity, has a maximum lifetime of 1 hour, and can only grant access to folders and files within a data item, like a lakehouse.

How a shared access signature works

A shared access signature is a token appended to the URI for a OneLake resource. The token contains a special set of query parameters that indicate how the client can access the resource. One of the query parameters is the signature. It's constructed from the SAS parameters and signed with the key that was used to create the SAS. OneLake uses this signature to authorize access to the folder or file in OneLake. OneLake SAS uses the same format and properties as [Azure Storage user-delegated SAS](#), but with more security restrictions on the lifetime and scope.

A OneLake SAS is signed with a user delegation key (UDK), which is backed by a Microsoft Entra credential. You can request a user delegation key with the [Get User Delegation Key](#) operation. Then, you use this key (while it's still valid) to build the OneLake SAS. The permissions of that Microsoft Entra credential, along with the permissions explicitly granted to the SAS, determine the client's access to the resource.

Authorizing a OneLake SAS

When a client or application accesses OneLake with a OneLake SAS, the request is authorized using the Microsoft Entra credentials that requested the UDK used to create the SAS. Therefore, all OneLake permissions granted to that Microsoft Entra identity apply to the SAS, meaning a SAS can never exceed the permissions of the user creating it. Furthermore, when creating a SAS you explicitly grant permissions, letting you provide even more scoped-down permissions to the SAS. Between the Microsoft Entra identity, the explicitly granted

permissions, and the short-lifetime, OneLake follows security best practices for providing delegated access to your data.

When to use a OneLake SAS

OneLake SAS delegates secure and temporary access to OneLake, backed by a Microsoft Entra identity. Applications without native Microsoft Entra support can use a OneLake SAS to gain temporary access to load data without complicated set-up and integration work.

OneLake SAS also supports applications serving as proxies between users and their data. For example, some independent software vendors (ISVs) run between users and their Fabric workspace, providing extra functionality and possibly a different authentication model. By delegating access with a OneLake SAS, these ISVs can manage access to the underlying data and provide direct access to data, even if their users don't have Microsoft Entra identities.

Managing OneLake SAS

Two settings in your Fabric tenant manage the use of OneLake SAS.

The first setting is a tenant-level setting, **Use short-lived user-delegated SAS tokens**, which manages the generation of user delegation keys. Because user delegation keys are generated at the tenant-level, they're controlled by a tenant setting. This setting is turned on by default, since these user delegation keys have equivalent permissions to the Microsoft Entra identity requesting them and are always short-lived.

 **Note**

Turning off this feature prevents all workspaces from using OneLake SAS, as all users will be unable to generate user delegation keys.

The second setting is a delegated workspace setting, **Authenticate with OneLake user-delegated SAS tokens**, which controls whether a workspace accepts OneLake SAS. This setting is turned off by default. A workspace admin can turn on this setting to allow authentication with OneLake SAS in their workspace. A tenant admin can turn this setting on for all workspaces via the tenant setting, or leave it to workspace admins to turn on.

You can also monitor the creation of user delegation keys in the Microsoft Purview portal. To view all keys generated in your tenant, search for the operation name **generateonelakeudk**. Because creating a SAS is a client-side operation, you can't monitor or limit the creation of a OneLake SAS, only the generation of a UDK.

Best practices with OneLake SAS

- Always use HTTPS to create or distribute a SAS to protect against man-in-the-middle attacks seeking to intercept the SAS.
- Track your, key, and SAS token expiry times. OneLake user delegation keys and SAS tokens have a maximum lifetime of 1 hour. Attempting to request a UDK or build a SAS with a lifetime longer than 1 hour causes the request to fail. To prevent SAS being used to extend the lifetime of expiring OAuth tokens, the lifetime of the token must also be longer than the expiry time of the user delegation key and the SAS.
- Be careful with a SAS token's start time. Setting the start time for a SAS as the current time might cause failures for the first few minutes, due to differing start times between machines (clock skew). Setting the start time to be a few minutes in the past helps protect against these errors.
- Grant the least possible privileges to the SAS. Providing the minimum required privileges to the fewest possible resources is a security best-practice and lessens the impact if a SAS is compromised.
- Monitor the generation of user delegation keys. You can audit the creation of user delegation keys in the Microsoft Purview portal. Search for the operation name `generateonelakeudk` to view keys generated in your tenant.
- Understand the limitations of OneLake SAS. Because OneLake SAS tokens can't have workspace-level permissions, they aren't compatible with some Azure Storage tools which expect container-level permissions to traverse data, like Azure Storage Explorer.

Related content

- [How to create a OneLake SAS](#)
- [Generate a user delegation key](#)
- [Fabric and OneLake data security](#)
- [Create a user delegation SAS for a blob with Python](#)

Create a OneLake shared access signature

Article • 05/20/2025

You can create a OneLake shared access signature (SAS) to provide short-term, delegated access to a folder or file in OneLake backed by your Microsoft Entra credentials. A OneLake SAS can provide temporary access to applications that don't support Microsoft Entra. These applications can then load data or serve as proxies between other customer applications or software development companies.

To create a OneLake SAS, request a user delegation key by calling the [Get User Delegation Key](#) operation. Then, use the key to sign the SAS.

A OneLake SAS can grant access to files and folders within data items only. You can't use it for management operations such as creating or deleting workspaces or items.

Creating a OneLake SAS is similar to creating an [Azure Storage user-delegated SAS](#). You use the same parameters for compatibility with tools and applications that work with Azure Storage.

Required permissions

Requesting a user delegation key is a tenant-level operation in Microsoft Fabric. The user or security principal who requests a user delegation key must have at least read permissions in one workspace in the Fabric tenant. The requesting user's identity is used to authenticate the SAS, so the user must have permission to the data that they grant the SAS access to.

Acquire an OAuth 2.0 token

To get the user delegation key, first request an OAuth 2.0 token from Microsoft Entra ID. Authorize the call to the `Get User Delegation Key` operation by providing the [bearer token](#). For more information about requesting an OAuth token from Microsoft Entra ID, see the [article about authentication flows and application scenarios](#).

Request the user delegation key

Calling the `Get User Delegation Key` operation returns the key as a set of values that are used as parameters on the user delegation SAS token. These parameters are described in the [Get User Delegation Key](#) reference and in the next section.

Note

Calling the `Get User Delegation Key` operation by using a Fabric workload, such as a Python notebook, requires the [regional endpoint](#) for OneLake. The capacity region determines this endpoint. Otherwise, the received response is `200 Healthy` instead of the delegation key.

When a client requests a user delegation key by using an OAuth 2.0 token, OneLake returns the key on behalf of the client. A SAS created with this user delegation key is granted, at most, the permissions granted to the client. But they're scoped down to the permissions explicitly granted in the SAS.

You can create any number of OneLake SAS tokens for the lifetime of the user delegation key. However, a OneLake SAS and user delegation keys can be valid for no more than one hour. They can't exceed the lifetime of the token that requested them. These lifetime restrictions are shorter than the maximum lifetime of an Azure Storage user delegation SAS.

Construct a user delegation SAS

The following table summarizes the fields that are supported for a OneLake SAS token. Subsequent sections provide more details about these parameters and how they differ from Azure Storage SAS tokens. OneLake doesn't support every optional parameter that Azure Storage supports. A OneLake SAS constructed with an unsupported parameter will be rejected.

[] [Expand table](#)

| SAS field name | SAS token parameter | Status | Description |
|-----------------------------|---------------------|----------|---|
| <code>signedVersion</code> | <code>sv</code> | Required | This field indicates the version of the storage service that's used to construct the signature field. OneLake supports version <code>2020-02-10</code> and earlier, or version <code>2020-12-06</code> and later. |
| <code>signedResource</code> | <code>sr</code> | Required | This field specifies which resources are accessible via the shared access signature. Only blob (<code>b</code>) and directory (<code>d</code>) are applicable to OneLake. |
| <code>signedStart</code> | <code>st</code> | Optional | This field specifies the time when the shared access signature becomes valid. It's in ISO 8601 UTC format. |
| <code>signedExpiry</code> | <code>se</code> | Required | This field specifies the time when the shared access signature expires. |

| SAS field name | SAS token parameter | Status | Description |
|---------------------------------------|---------------------|-------------|--|
| <code>signedPermissions</code> | <code>sp</code> | Required | This field indicates which operations the SAS can perform on the resource. For more information, see the Specify permissions section. |
| <code>signedObjectId</code> | <code>skoid</code> | Required | This field identifies a Microsoft Entra security principal. |
| <code>signedtenantId</code> | <code>sktid</code> | Required | This field specifies the Microsoft Entra tenant in which a security principal is defined. |
| <code>signedKeyStartTime</code> | <code>skt</code> | Optional | This field specifies the time in UTC when the signing key starts. The Get User Delegation Key operation returns it. |
| <code>signedKeyExpiryTime</code> | <code>ske</code> | Required | This field specifies the time in UTC when the signing key ends. The Get User Delegation Key operation returns it. |
| <code>signedKeyVersion</code> | <code>skv</code> | Required | This field specifies the storage service version that's used to get the user delegation key. The Get User Delegation Key operation returns it. OneLake supports version <code>2020-02-10</code> and earlier, or version <code>2020-12-06</code> and later. |
| <code>signedKeyService</code> | <code>skss</code> | Required | This field indicates the valid service for the user delegation key. OneLake supports only Azure Blob Storage (<code>skss=b</code>). |
| <code>signature</code> | <code>sig</code> | Required | The signature is a hash-based message authentication code (HMAC) computed over the string-to-sign and key by using the SHA256 algorithm, and then encoded by using Base64 encoding. |
| <code>signedDirectoryDepth</code> | <code>sdd</code> | Optional | This field indicates the number of directories within the root folder of the directory specified in the <code>canonicalizedResource</code> field of the string-to-sign. It's supported only when <code>sr=d</code> . |
| <code>signedProtocol</code> | <code>spr</code> | Optional | OneLake supports only HTTPS requests. |
| <code>signedAuthorizedObjectId</code> | <code>saoid</code> | Unsupported | A OneLake SAS doesn't support this feature. |

| SAS field name | SAS token parameter | Status | Description |
|--|---------------------|-------------|---|
| <code>signedUnauthorizedObjectId</code> | <code>suid</code> | Unsupported | A OneLake SAS doesn't support this feature. |
| <code>signedCorrelationId</code> | <code>suid</code> | Unsupported | A OneLake SAS doesn't support this parameter. |
| <code>signedEncryptionScope</code> | <code>ses</code> | Unsupported | A OneLake SAS doesn't currently support custom encryption scopes. |
| <code>signedIP</code> | <code>sip</code> | Unsupported | A OneLake SAS doesn't currently support IP filtering. |
| <code>Cache-Control</code> response header | <code>rscC</code> | Unsupported | A OneLake SAS doesn't support this parameter. |
| <code>Content-Disposition</code> response header | <code>rscD</code> | Unsupported | A OneLake SAS doesn't support this parameter. |
| <code>Content-Encoding</code> response header | <code>rscE</code> | Unsupported | A OneLake SAS doesn't support this parameter. |
| <code>Content-Language</code> response header | <code>rscL</code> | Unsupported | A OneLake SAS doesn't support this parameter. |
| <code>Content Type</code> response header | <code>rscT</code> | Unsupported | A OneLake SAS doesn't support this parameter. |

Specify permissions

The permissions specified in the `signedPermissions` (`sp`) field on the SAS token indicate which operations a client that possesses the SAS can perform on the resource.

Permissions can be combined to permit a client to perform multiple operations with the same SAS. When you construct the SAS, you must include permissions in the following order:
`racwdxyltmeopi`.

Examples of valid permission settings include `rw`, `rd`, `r1`, `wd`, `wl`, and `r1`. You can't specify a permission more than once.

To ensure parity with existing Azure Storage tools, OneLake uses the same permission format as Azure Storage. OneLake evaluates the permissions granted to a SAS in `signedPermissions`, the permissions of the signing identity in Fabric, and any [OneLake data access roles](#), if applicable.

Remember that some operations, such as setting permissions or deleting workspaces, generally aren't permitted on OneLake via Azure Storage APIs. Granting that permission (`sp=op`) doesn't allow a OneLake SAS to perform those operations.

[Expand table](#)

| Permission | URI symbol | Resource | Allowed operations |
|-------------------------|------------|-----------------|---|
| Read | r | Directory, blob | Read the content, blocklist, properties, and metadata of any blob in the container or directory. Use a blob as the source of a copy operation. |
| Add | a | Directory, blob | Add a block to an append blob. |
| Create | c | Directory, blob | Write a new blob, snapshot a blob, or copy a blob to a new blob. |
| Write | w | Directory, blob | Create or write content, properties, metadata, or a blocklist. Snapshot or lease the blob. Use the blob as the destination of a copy operation. |
| Delete | d | Directory, blob | Delete a blob. |
| Delete version | x | Blob | Delete a blob version. |
| Permanent delete | y | Blob | Permanently delete a blob snapshot or version. |
| List | l | Directory | List blobs nonrecursively. |
| Tags | t | Blob | Read or write the tags on a blob. |
| Move | m | Directory, blob | Move a blob or a directory and its contents to a new location. |
| Execute | e | Directory, blob | Get the system properties. If the hierarchical namespace is enabled for the storage account, get the POSIX access control list of a blob. |
| Ownership | o | Directory, blob | Set the owner or owning group. This operation is unsupported in OneLake. |
| Permissions | p | Directory, blob | Set the permissions. This operation is unsupported in OneLake. |
| Set immutability policy | i | Blob | Set or delete the immutability policy or legal hold on a blob. |

Specify the signature

The `signature (sig)` field is used to authorize a request that a client made with the shared access signature. The string-to-sign is a unique string that's constructed from the fields that must be verified to authorize the request. The signature is an HMAC that's computed over the string-to-sign and key by using the SHA256 algorithm, and then encoded by using Base64 encoding.

To construct the signature string of a user delegation SAS:

1. Create the string-to-sign from the fields that the request made.
2. Encode the string as UTF-8.
3. Compute the signature by using the HMAC SHA256 algorithm.

The fields that are included in the string-to-sign must be URL decoded. The required fields depend on the service version that's used for the authorization (`sv`) field. The following sections describe the string-to-sign configurations for versions that support OneLake SAs.

Version 2020-12-06 and later

HTTP

```
StringToSign = signedPermissions + "\n" +
               signedStart + "\n" +
               signedExpiry + "\n" +
               canonicalizedResource + "\n" +
               signedKeyObjectId + "\n" +
               signedKeyTenantId + "\n" +
               signedKeyStart + "\n" +
               signedKeyExpiry + "\n" +
               signedKeyService + "\n" +
               signedKeyVersion + "\n" +
               signedAuthorizedUserObjectId + "\n" +
               signedUnauthorizedUserObjectId + "\n" +
               signedCorrelationId + "\n" +
               signedIP + "\n" +
               signedProtocol + "\n" +
               signedVersion + "\n" +
               signedResource + "\n" +
               signedSnapshotTime + "\n" +
               signedEncryptionScope + "\n" +
               rscc + "\n" +
               rscd + "\n" +
               rsce + "\n" +
               rscl + "\n" +
               rsct
```

Version 2020-02-10 and earlier

This configuration applies to version 2020-02-10 and earlier, except for version 2020-01-10 (which the next section describes).

HTTP

```
StringToSign = signedPermissions + "\n" +
               signedStart + "\n" +
               signedExpiry + "\n" +
               canonicalizedResource + "\n" +
               signedKeyObjectId + "\n" +
               signedKeyTenantId + "\n" +
               signedKeyStart + "\n" +
               signedKeyExpiry + "\n" +
               signedKeyService + "\n" +
               signedKeyVersion + "\n" +
               signedAuthorizedUserObjectId + "\n" +
               signedUnauthorizedUserObjectId + "\n" +
               signedCorrelationId + "\n" +
               signedIP + "\n" +
               signedProtocol + "\n" +
               signedVersion + "\n" +
               signedResource + "\n" +
               rscc + "\n" +
               rscd + "\n" +
               rsce + "\n" +
               rscl + "\n" +
               rsct
```

Version 2020-01-10

HTTP

```
StringToSign = signedPermissions + "\n" +
               signedStart + "\n" +
               signedExpiry + "\n" +
               canonicalizedResource + "\n" +
               signedKeyObjectId + "\n" +
               signedKeyTenantId + "\n" +
               signedKeyStart + "\n" +
               signedKeyExpiry + "\n" +
               signedKeyService + "\n" +
               signedKeyVersion + "\n" +
               signedAuthorizedUserObjectId + "\n" +
               signedUnauthorizedUserObjectId + "\n" +
               signedCorrelationId + "\n" +
               signedIP + "\n" +
               signedProtocol + "\n" +
               signedVersion + "\n" +
```

```
signedResource + "\n" +
signedSnapshotTime + "\n" +
rscc + "\n" +
rscd + "\n" +
rsce + "\n" +
rscl + "\n" +
rsct
```

Canonicalized resource

The `canonicalizedResource` portion of the string is a canonical path to the resource. It must include the OneLake endpoint and the resource name, and it must be URL decoded. A OneLake path must include its workspace. A directory path must include the number of subdirectories that correspond to the `sdd` parameter.

The following examples show how to convert your OneLake URL to the corresponding canonicalized resource. Remember that OneLake supports both Distributed File System (DFS) and blob operations and endpoints. The account name for OneLake is always `onelake`.

Blob file

HTTP

```
URL =
https://onelake.blob.fabric.microsoft.com/myWorkspace/myLakehouse.Lakehouse/Files/
sales.csv
canonicalizedResource =
"/blob/onelake/myWorkspace/myLakehouse.Lakehouse/Files/sales.csv"
```

DFS directory

HTTP

```
URL =
https://onelake.dfs.fabric.microsoft.com/myWorkspace/myLakehouse.Lakehouse/Files/
canonicalizedResource = "/blob/onelake/myWorkspace/myLakehouse.Lakehouse/Files/"
```

OneLake SAS example

The following example shows a OneLake SAS URI with a OneLake SAS token appended to it. The SAS token provides read and write permissions to the `Files` folder in the lakehouse.

HTTP

```
https://onelake.blob.fabric.microsoft.com/myWorkspace/myLakehouse.Lakehouse/Files/  
?sp=rw&st=2023-05-24T01:13:55Z&se=2023-05-24T09:13:55Z&skoid=<object-id>&sktid=  
<tenant-id>&skt=2023-05-24T01:13:55Z&ske=2023-05-24T09:13:55Z&skb=b&skv=2022-11-  
02&sv=2022-11-02&sr=d&sig=<signature>
```

Related content

- [Create a user delegation SAS](#)
- [Request a user delegation key](#)
- [Get started with OneLake data access roles](#)

OneLake diagnostics

OneLake diagnostics provides end-to-end visibility into how data is accessed and used across your Microsoft Fabric environment. It enables organizations to answer critical questions like "who accessed what, when, and how," supporting data governance, operational insight, and compliance reporting.

When enabled at the workspace level, OneLake diagnostics streams data access events as JSON logs into a Lakehouse of your choice within the same capacity. These logs can be easily transformed into analytics-ready Delta tables, allowing teams to build dashboards and reports that track usage patterns, top-accessed items, and trends over time.

As all data in Fabric is unified in OneLake, diagnostics at the workspace level provide a consistent, trustworthy record of data activity—regardless of how or where the data is consumed. This includes:

- User actions in the Fabric web experience
- Programmatic access via APIs, pipelines, and analytics engines
- Cross-workspace shortcuts, with events captured from the source workspace

This unified logging approach ensures that even when data is accessed through shortcuts or across workspaces, visibility is preserved.

Diagnostic events are captured for both Fabric and non-Fabric sources. For access through the Fabric UI and the Blob or Azure Data Lake Storage (ADLS) APIs, every operation is logged. For Fabric workloads access, it records that temporary access was granted, so you can look further in the engine specific logs. This ensures efficient logging while maintaining visibility into how data is consumed across your organization.

Example scenarios supported by OneLake diagnostics

- Security investigation: Track which users accessed sensitive datasets, when, and from where. Helps identify unauthorized access attempts or unusual patterns.
- Performance troubleshooting: Diagnose latency or failure issues by correlating diagnostic events with user actions or system interactions.
- Usage analytics and optimization: Understand which datasets are most frequently accessed, by whom, and how often. Supports data governance and resource optimization.
- Integration monitoring: Monitor external systems interacting with OneLake (via APIs or connectors), ensuring integrations are functioning as expected and diagnosing issues when they arise.

Configuring OneLake diagnostics

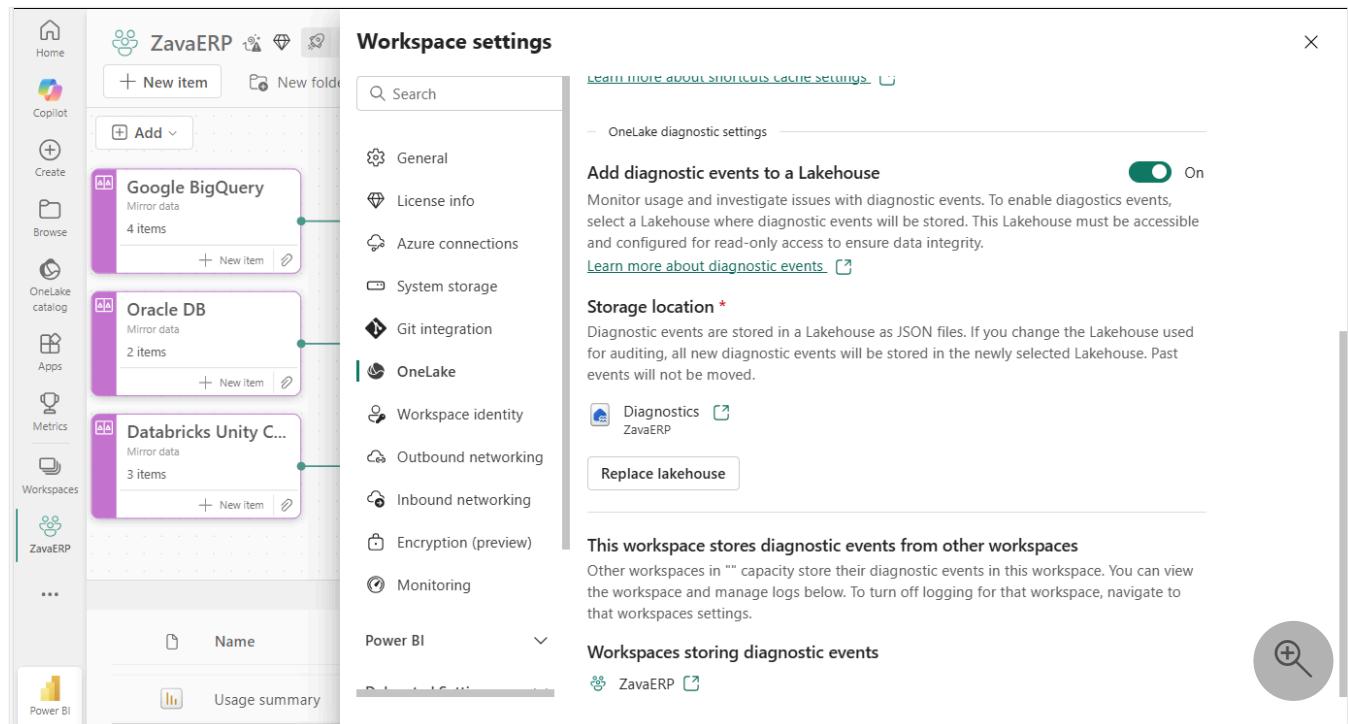
Best practice recommendations

To simplify management and improve access control, consider using a dedicated workspace to store diagnostic events. If you're enabling diagnostics across multiple workspaces in the same capacity, consider centralizing logs in a single Lakehouse to make analysis easier.

Prerequisites

- Create a Lakehouse to store OneLake diagnostic events.
- The Lakehouse must reside in the same capacity as the workspaces you want to enable diagnostics for.
- If the workspace uses private links for inbound network protection, it must be within the same virtual network as the Lakehouse.
- You must be a workspace admin for the workspace where you're enabling OneLake diagnostics, and a contributor to the destination Lakehouse.

Enabling OneLake diagnostics



Use the following steps to enable OneLake diagnostics:

1. Open the workspace settings.
2. Navigate to the OneLake settings tab.
3. Toggle "Add diagnostic events to a Lakehouse" to On.

4. Select the Lakehouse where you want to store the diagnostic events.

 Note

It takes up to 1 hour for diagnostic events to begin flowing into the Lakehouse.

Enabling immutable diagnostic logs

Protect diagnostic events

The diagnostic events stored in this workspace can be protected from tampering by enabling immutability. Once enabled, events cannot be modified or deleted during the immutability period.

Immutability period

You can increase the immutability period after it is set, but you cannot decrease it

365  days

Apply



OneLake diagnostic events can be made immutable, this means that the JSON files that contain diagnostic events can't be tampered with, or deleted, during the immutability retention period. OneLake diagnostics immutability is built on the Immutable storage for Azure Blob Storage capability. For more information, please read [Store business-critical blob data with immutable storage in a write once, read many \(WORM\) state](#)

The immutability period is configured on the workspace that contains diagnostic events. To configure the immutability period, you must have previously configured a workspace to store diagnostic events in this workspace. The immutability period applies to all events stored in this workspace.

1. Enter the required immutability period
2. Press apply

 Note

Once the immutability policy is applied, the files can't be modified or deleted until the immutability retention period passes. Use caution while applying the policy as it can't be changed once set.

Changing the OneLake diagnostic Lakehouse

1. Open the workspace settings.
2. Go to the OneLake settings tab.
3. Select Replace Lakehouse.
4. Choose a new Lakehouse.

! Note

Previously captured diagnostic events remain in the original Lakehouse. New events are stored in the newly selected Lakehouse.

Disabling OneLake diagnostics

1. Open the workspace settings.
2. Navigate to the OneLake settings tab.
3. Toggle "Add diagnostic events to a Lakehouse" to Off.

! Note

The previously selected Lakehouse is retained. If you re-enable diagnostics, it uses the same Lakehouse as before.

OneLake diagnostic events

The screenshot shows the ZavaERP application interface. On the left, the Explorer pane displays a tree structure of diagnostic logs, with the 'DiagnosticLogs' folder expanded. Under 'DiagnosticLogs', there is a 'OneLake' folder, which contains a 'Workspaces' folder. Inside 'Workspaces', there is a folder with a blacked-out name, which contains a 'y=2025' folder. 'y=2025' contains 'm=08', 'm=09', 'm=10', 'd=01', 'd=02', 'd=03', and 'h=00'. 'h=00' contains 'm=00', 'h=01', 'h=02', 'h=03', 'h=04', 'h=05', 'h=06', and 'h=07'. The 'h=00' folder is currently selected. On the right, the main pane shows the contents of the 'PT1H.json' file (preview). The JSON code is as follows:

```
32     "time": "2025-10-03T00:00:58.467903Z",
33     "id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
34     "resourceId": "/SUBSCRIPTIONS/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/RESOURCEGROUPS/",
35     "category": "OneLakeData",
36     "type": "Microsoft.Fabric.Audit.OneLake.Data",
37     "schemaVersion": "1.0",
38     "data": {
39       "workspaceId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
40       "itemId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
41       "itemType": "MountedRelationalDatabase",
42       "tenantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
43       "executingPrincipalId": "Unknown",
44       "correlationId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
45       "operationName": "PutBlobFromURL",
46       "executingUpn": "Unknown",
47       "executingPrincipalType": "ServicePrincipal",
48       "authType": "OAuth",
49       "accessStartTime": "2025-10-02T23:34:38.0000000Z",
50       "accessEndTime": "2025-10-03T00:44:38.0000000Z",
51       "operationCategory": "Write",
52       "originatingApp": "MirroredDatabase",
53       "serviceEndpoint": "Blob",
54       "resource": "/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx/Files/TmpStage/tpch_CUSTOMER",
55       "capacityId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
56       "httpStatusCode": 201,
57       "isShortcut": false,
58       "accessedViaResource": "/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx/Files/TmpStage/tp",
59       "callerIpAddress": "127.0.0.1"
60     }
61   }
62 }
```

OneLake diagnostic events are stored in the DiagnosticLogs folder within the Files section of a Lakehouse. JSON files are written to a folder with the following path:

Files/DiagnosticLogs/OneLake/Workspaces/WorkspaceId/y=YYYY/m=MM/d=DD/h=HH/m=00/PT1H.json

The JSON event contains the following attributes:

 Expand table

| Property | Description |
|------------------------|---|
| workspaceId | The GUID of the workspace with diagnostics enabled. |
| itemId | The GUID of the fabric item, for example the Lakehouse, which is performing the OneLake operation |
| itemType | The kind of item that performed the OneLake operation |
| tenantId | The tenant identifier that performed the OneLake operation |
| executingPrincipalId | The GUID of the Microsoft Entra principle performing the OneLake operation |
| correlationId | A GUID correlation identifier for the OneLake operation |
| operationName | The OneLake operation being performed (not provided for internal Fabric operations). See Operations below for more details. |
| operationCategory | The broad category of the OneLake operation (for example, Read) |
| executingUPN | The Microsoft Entra unique principal name performing the operation (not provided for internal Fabric operations) |
| executingPrincipalType | The type of principal being used, for example User or Service Principal |
| accessStartTime | The time the operation was performed. When temporary access is provided, the time temporary access started |
| accessEndTime | The time the operation was completed. When temporary access is provided, the time temporary access completed |
| originatingApp | The workload performing the operation. For external access, then originatingApp is the user agent string |
| serviceEndpoint | The OneLake service endpoint being used (DFS, Blob or Other) |
| Resource | The resources being accessed (relative to the workspace) |
| capacityId | The identifier of the capacity performing the OneLake operation |
| httpStatusCode | The status code returned to the user |
| isShortcut | Indicates if access was performed via a shortcut |

| Property | Description |
|---------------------|---|
| accessedViaResource | The resource the data was accessed via. When a shortcut is used, this is the location of the shortcut |
| callerIPAddress | The IP address of the caller |

Personal Data

OneLake diagnostic events include `executingUPN` and `callerIpAddress`. To redact this data, tenant admins can disable the setting "Include end-user identifiers in OneLake diagnostic logs" in the Fabric Admin Portal. When disabled, these fields are excluded from new diagnostic events.

Frequently Asked Questions (FAQ)

What happens if the destination Lakehouse is deleted?

If the Lakehouse selected for diagnostics is deleted:

- **Diagnostics will be automatically disabled** for all workspaces that were pointing to it.
- **Previously captured diagnostic data is not deleted**—it remains in the deleted Lakehouse's storage until the workspace itself is deleted. To **resume diagnostics**, select a new Lakehouse in the same workspace. OneLake will enable diagnostics, and all previously captured logs remain accessible.

What happens if the workspace is deleted?

- If a workspace is deleted, **OneLake diagnostics for that workspace are also deleted**.
- If the workspace is **restored**, the diagnostic data is restored.
- Once the workspace is **permanently deleted**, the associated diagnostic events are also permanently removed.

What happens when you change capacities?

- When a workspace is moved to a different capacity, **diagnostic logging is disabled**.
- You must **select a new Lakehouse within the new capacity** to re-enable diagnostics.

What happens when BCDR is enabled for the workspace?

- When Business Continuity and Disaster Recovery (BCDR) is enabled, OneLake diagnostics data is replicated to the secondary region, and is accessible via the OneLake APIs if a failover occurs.

Can you audit OneLake diagnostics?

- Yes. When workspace monitoring is enabled, disabled, or the Lakehouse is updated, a `ModifyOneLakeDiagnosticSettings` event is captured in the [Microsoft 365 security logs](#), allowing you to audit changes to diagnostic settings.

How much consumption does OneLake diagnostics generate?

- OneLake diagnostics is comparable in cost to Azure Storage diagnostics when emitting to a storage account. For the latest details, see the official pricing page: [OneLake consumption – Microsoft Fabric | Microsoft Learn](#).

Limitations

OneLake diagnostics isn't currently compatible with [Workspace outbound access protection \(OAP\)](#) across workspaces. If you require OneLake diagnostics and OAP to work together, you must select a Lakehouse in the same Workspace.

When OneLake diagnostics is configured, the selection of the workspace honors workspace private link configuration by limiting your selection to workspaces within the same private network. However, OneLake diagnostics doesn't automatically respond to networking changes.

Operations

Global operations

[] [Expand table](#)

| Operation | Category |
|-------------------------------|----------|
| ReadFileOrGetBlob | Read |
| GetFileOrBlobProperties | Read |
| GetActionFileOrBlobProperties | Read |
| CheckAccessFileOrBlob | Read |

| Operation | Category |
|------------------|----------|
| DeleteFileOrBlob | Delete |

Blob operations

[Expand table](#)

| Operation | Category |
|--------------------|----------|
| GetBlockList | Read |
| ListBlob | Read |
| GetBlob | Read |
| DeleteBlob | Delete |
| UndeleteBlob | Write |
| GetBlobMetadata | Read |
| SetBlobExpiry | Write |
| SetBlobMetadata | Write |
| SetBlobProperties | Write |
| SetBlobTier | Write |
| LeaseBlob | Write |
| AbortCopyBlob | Write |
| PutBlockFromURL | Write |
| PutBlock | Write |
| PutBlockList | Write |
| AppendBlockFromURL | Write |
| AppendBlock | Write |
| AppendBlobSeal | Write |
| PutBlobFromURL | Write |
| CopyBlob | Write |
| PutBlob | Write |

| Operation | Category |
|----------------------------|----------|
| QueryBlobContents | Read |
| GetBlobProperties | Read |
| CreateContainer | Write |
| DeleteContainer | Delete |
| GetContainerMetadata | Read |
| GetContainerProperties | Read |
| SetContainerMetadata | Write |
| SetContainerAcl | Write |
| LeaseContainer | Write |
| RestoreContainer | Write |
| SnapshotBlob | Write |
| CreateFastPathReadSession | Read |
| CreateFastPathWriteSession | Write |

DFS operations

[Expand table](#)

| Operation | Category |
|-------------------------|----------|
| CreateFileSystem | Write |
| PatchFileSystem | Write |
| DeleteFileSystem | Delete |
| GetFileSystemProperties | Read |
| CreateDirectory | Write |
| CreateFile | Write |
| DeleteDirectory | Delete |
| DeleteFile | Delete |
| RenameFileOrDirectory | Write |

| Operation | Category |
|------------------------------|-----------------|
| ListFilePath | Read |
| AppendDataToFile | Write |
| FlushDataToFile | Write |
| SetFileProperties | Write |
| SetAccessControlForFile | Write |
| SetAccessControlForDirectory | Write |
| LeasePath | Write |
| GetPathStatus | Read |
| GetAccessControlListForFile | Read |

Fabric operations

 [Expand table](#)

| Operation | Category |
|----------------------|-----------------|
| FabricWorkloadAccess | Read |

Last updated on 12/09/2025

Limit inbound requests with inbound access protection

08/21/2025

Inbound access protection secures connections between your virtual network and Microsoft Fabric. By setting up private links, you can prevent access to your data in OneLake from the public internet.

What is inbound access protection?

Turning on inbound access protection restricts public access to your Fabric tenant or workspace. All inbound calls must use an approved private endpoint, either from your own virtual network or from an approved service such as another Fabric workspace. These private endpoints ensure that connections come only from trusted sources and not the public internet. To learn more about how to set up private endpoints and block public access to your workspace, see [Fabric workspace private links](#).



OneLake and private links

To connect to your workspace over a private endpoint, you need to use the workspace fully qualified domain name (FQDN). This workspace FQDN ensures the call goes directly to the private endpoint, and is specific to each workspace. OneLake supports both Blob and DFS (Data File System) versions of the workspace FQDN:

- `https://{{workspaceid}}.z{{xy}}.dfs.fabric.microsoft.com`
- `https://{{workspaceid}}.z{{xy}}.blob.fabric.microsoft.com\`

Where:

- `{{workspaceid}}` is the workspace GUID without dashes.
- `{{xy}}` is the first two characters of the workspace GUID.

To connect to your tenant private endpoint, you can continue to use the OneLake global FQDN:

- `https://onelake.dfs.fabric.microsoft.com`

- <https://onelake.blob.fabric.microsoft.com>

If your environment doesn't have a workspace private link set up, the workspace FQDN connects over the public internet. If only a tenant private link is set up, the workspace FQDN connects to the tenant private link. If both a tenant and workspace private link are set up, the workspace FQDN connects to the workspace private link.

Limit outbound requests with outbound access protection

Outbound access protection protects data by limiting OneLake's outbound requests made through shortcuts and copy operations.

What is outbound access protection?

Outbound access protection helps ensure that data is shared securely within your network security perimeter. For example, data exfiltration protection solutions use outbound access protection controls to limit a malicious actor's ability to move large amounts of data to an untrusted external location. Outbound protections only limit requests that originate in the workspace and communicate with different workspace or location. A comprehensive network security solution also involves [inbound network protection](#) through private links, combined with [data access controls](#) to limit access to your data.

To learn more about managing outbound access protection, see [Workspace outbound access protection](#).

When does OneLake make outbound requests?

There are two scenarios where OneLake makes an outbound request: shortcuts and copy operations. An outbound request is defined as any request made from within the workspace towards a location outside the workspace. Only the directionality of the call matters - both reads and writes to external locations can exfiltrate sensitive information to untrusted locations.

Shortcuts

Shortcuts are objects in OneLake that point to other storage locations, which can be internal or external to OneLake. The location that a shortcut points to is known as the target path, and the location where the shortcut appears is the shortcut path. If the shortcut target is a different workspace or external storage location than the shortcut path, it's an outbound shortcut and subject to outbound access protection.

Outbound access protection doesn't restrict shortcuts with a source and target within the same workspace, because all OneLake calls remain within the boundary of the workspace.



Copying data within OneLake

When you copy data between two OneLake workspaces using Azure Storage copy APIs, OneLake makes an outbound call from the source workspace to the target workspace. If outbound access protection is enabled on the source workspace, that outbound call is blocked, and the copy operation fails. To allow data movement, you must create a managed private endpoint from the source workspace to the target workspace.

The following copy operation from Workspace A to Workspace B is blocked when outbound access protection is enabled, unless there's an approved managed private endpoint from Workspace A to Workspace B. As a reminder, AzCopy operations always follow the format

```
azcopy copy <source> <destination>.
```

Syntax

AzCopy

```
azcopy copy
"https://onelake.dfs.fabric.microsoft.com/WorkspaceA/LakehouseA.Lakehouse/Files/sales.csv"
"https://onelake.dfs.fabric.microsoft.com/WorkspaceB/LakehouseB.Lakehouse/Files/sales.csv" --trusted-microsoft-suffixes "fabric.microsoft.com"
```

Outbound access protection doesn't block copy operations that move data within a workspace.

Copying data between Azure Storage and OneLake

When you copy data between Azure Storage and OneLake, the direction of the outbound requests is **reversed**. The destination account makes an **outbound call to the source account**. This behavior applies to copy operations made directly with Azure Storage [Copy Blob from URL](#) and [Put Block from URL](#) APIs. It also applies to managed copy experiences with [AzCopy](#) and Azure Storage Explorer. Outbound access protection restricts this outbound call from destination to source. **However, this means outbound access protection does not restrict your workspace from being the source of a copy operation, as no outbound call is made from the source workspace.**

For example, the following AzCopy sample moves data from the source Azure Storage account "source" to the destination lakehouse in OneLake. If Workspace A has outbound protection

turned on, then the outbound call from Workspace A to the external Azure Storage account is blocked, and the data isn't loaded.

Syntax

AzCopy

```
azcopy copy "https://source.blob.core.windows.net/myContainer/sales.csv"  
"https://onelake.dfs.fabric.microsoft.com/WorkspaceA/LakehouseA.Lakehouse/Files/sale  
s.csv" --trusted-microsoft-suffixes "fabric.microsoft.com"
```

However, in the following scenario, Workspace A is now the source of the copy operation, with the external Azure Data Lake Storage (ADLS) account as the destination. In this scenario, **outbound access protection does not block this call**, as only inbound calls are made to Workspace A. To restrict these types of operations, see [Protect inbound traffic](#).

Syntax

AzCopy

```
azcopy copy  
"https://onelake.dfs.fabric.microsoft.com/WorkspaceA/LakehouseA.Lakehouse/Files/sale  
s.csv" "https://source.blob.core.windows.net/myContainer/sales.csv" --trusted-  
microsoft-suffixes "fabric.microsoft.com"
```

Cross-workspace operations

To ensure you can continue to read and write data across workspaces, you can create a [managed private endpoint](#) between workspaces. When a valid managed private endpoint exists from one workspace to another, outbound requests are permitted from the source workspace to the target workspace even when outbound access is restricted.

For example, creating a managed private endpoint from Workspace A to Workspace B lets users read data in Workspace B through a shortcut, or copy data from Workspace B to Workspace A using AzCopy.

Related Content

- [Fabric outbound access protection](#).
- [Fabric inbound access protection](#).
- [Manage inbound access to OneLake with workspace private links](#).

Last updated on 11/26/2025

Disaster recovery and data protection for OneLake

Article • 05/20/2025

All data in OneLake is accessed through data items. These data items can reside in different regions depending on their workspace, because a workspace is created under a capacity that's tied to a specific region.

OneLake uses zone-redundant storage (ZRS) where that storage type is available. (See [Azure regions with availability zones](#).) Elsewhere, OneLake uses locally redundant storage (LRS). With both LRS and ZRS, your data is resilient to transient hardware failures within a datacenter.

Just like for [Azure Storage](#), LRS replicates data within a single datacenter in the primary region. LRS provides at least 99.99999999% (11 nines) durability of objects over a year. This durability helps protect against server rack and drive failures, but not against datacenter disasters.

Meanwhile, ZRS provides fault tolerance to datacenter failures by copying data synchronously across three Azure availability zones in the primary region. ZRS offers a durability of at least 99.999999999% (12 nines) over a year.

This article provides guidance on how to further protect your data from rare region-wide outages.

Disaster recovery

You can enable or disable business continuity and disaster recovery (BCDR) for a specific capacity through the capacity admin portal. If your capacity has BCDR activated, your data is duplicated and stored in two geographic regions so that it's geo-redundant. The standard region pairings in Azure determine the choice of the secondary region. You can't modify the secondary region.

If a disaster makes the primary region unrecoverable, OneLake might initiate a regional failover. After the failover finishes, you can use the OneLake APIs through the [global endpoint](#) to access your data in the secondary region. Data replication to the secondary region is asynchronous, so any data not copied during the disaster is lost. After a failover, the new primary datacenter has local redundancy only.

For a comprehensive understanding of the end-to-end experience, see [Reliability in Microsoft Fabric](#).

Soft deletion for OneLake files

In OneLake, soft deletion prevents accidental file loss by retaining deleted files for seven days before permanent removal. Soft-deleted data is billed at the same rate as active data.

You can restore files and folders by using Azure Blob Storage REST APIs, Azure Storage SDKs, and the Azure PowerShell Az.Storage module. Learn how to list and restore files by using [these PowerShell instructions](#) and how to connect to [OneLake with PowerShell](#).

Restore soft-deleted files via Azure Storage Explorer

You can restore deleted Lakehouse files by using Azure Storage Explorer. First, [connect to your workspace from Storage Explorer](#) by using the workspace ID in the URL. For example, use `https://onelake.dfs.fabric.microsoft.com/aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb`. You can find the workspace ID from the Microsoft Fabric portal's browser URL (`/groups/{workspaceID}`). Ensure that you use the GUID-based OneLake path to restore data.

After you connect to your workspace, follow these steps to restore soft-deleted data:

1. Select the dropdown button next to the path bar, and then select **Active and soft deleted blobs** instead of the default **Active blobs**.
2. Go to the folder that contains the soft-deleted file.
3. Right-click the file, and then select **Undelete**.

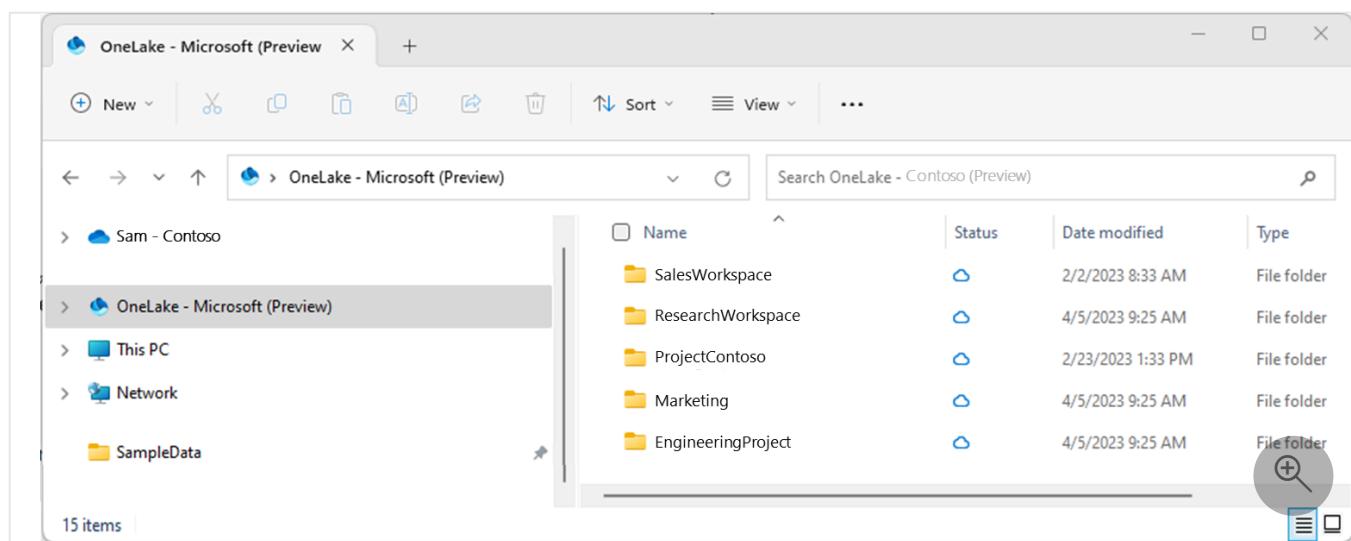
Related content

- [OneLake compute and storage consumption](#)

Use OneLake file explorer (preview) to access Fabric data

07/26/2025

The OneLake file explorer application seamlessly integrates OneLake with Windows File Explorer. This application automatically syncs all OneLake items that you have access to in Windows File Explorer. "Sync" refers to pulling up-to-date metadata on files and folders, and sending changes made locally to the OneLake service. Syncing doesn't download the data, it creates placeholders. You must double-click on a file to download the data locally.



Important

This feature is in [preview](#).

When you create, update, or delete a file via Windows File Explorer, it automatically syncs the changes to OneLake service. Updates to your item made outside of your File Explorer aren't automatically synced. To pull these updates, you need to right-click on the item or subfolder in Windows File Explorer and select **OneLake > Sync from OneLake**.

Installation instructions

OneLake file explorer currently supports Windows and is validated on Windows 10 and 11.

To install:

1. Download the [OneLake file explorer](#).
2. Double-click the file to start installing.

The storage location on your PC for the placeholders and any downloaded content is
`\USERPROFILE\OneLake - Microsoft\`.

Once the application is installed and running, you can see your OneLake data in Windows File Explorer.

Starting in version 1.0.13, the OneLake file explorer app notifies you when a new update is available. When a new version becomes available, you receive a Windows notification and the OneLake icon changes. Right-click on the OneLake icon in the Windows notification area. Select **Update Available** and follow steps to update.

Limitations and considerations

- Workspace names with the "/" character, encoded escape characters such as `%23`, and names that look like GUIDs fail to sync.
- Files or folders containing Windows reserved characters fail to sync. For more information, see [Naming files, paths, and namespaces](#).
- If Windows search is disabled, OneLake file explorer fails to start.
- Windows File Explorer is case insensitive, while OneLake is case sensitive. You can create files with the same name but different cases in the OneLake service using other tools, but Windows File Explorer only shows the oldest of these files.
- If a file fails to sync due to a network issue, you have to trigger the sync to OneLake. To prompt the sync process, open the file and save it. Alternatively, you can trigger a modify event [using PowerShell](#) by executing this command: `(Get-Item -Path "<file_path>").LastWriteTimeUtc = Get-Date`
- OneLake File Explorer does not support environments that require network proxy configurations. Attempts to launch or authenticate the application behind a proxy may result in connection failures or sign-in issues.
- OneLake File Explorer does not support syncing files marked as read-only. This limitation applies specifically to files marked as read-only by the user on their local machine. This behavior is by design to prevent conflicts with local file system permissions and to avoid unintended edits to protected content

Scenarios

The following scenarios provide details for working with the OneLake file explorer.

Start and exit OneLake file explorer

OneLake file explorer starts automatically at startup of Windows. You can disable the application from starting automatically by selecting **Startup apps** in Windows Task Manager and then right-clicking **OneLake**, and selecting **Disable**.

- To manually start the application, search for OneLake using Windows search (Windows+S) and select the OneLake application. The views for any folders that were previously synced refresh automatically.
- To exit, right-click on the OneLake icon in the Windows notification area, located at the far right of the taskbar, and select **Exit**. The sync pauses, and you can't access placeholder files and folders. You continue to see the blue cloud icon for placeholders that were previously synced but not downloaded.

Sync updates from OneLake

To optimize performance during the initial sync, OneLake file explorer syncs the placeholder files for the top-level workspaces and item names. When you open an item, OneLake file explorer syncs the files directly in that folder. Then, opening a folder within the item syncs the files directly in that folder. This functionality allows you to navigate your OneLake content seamlessly, without having to wait for all files to sync before starting to work.

When you create, update, or delete a file via OneLake file explorer, it automatically syncs the changes to OneLake service. Updates to your item made outside of your OneLake file explorer aren't automatically synced. To pull these updates, right-click on the workspace name, item name, folder name, or file in OneLake file explorer and select **OneLake > Sync from OneLake**. This action refreshes the view for any folders that were previously synced. To pull updates for all workspaces, right-click on the OneLake root folder and select **OneLake > Sync from OneLake**.

Sign in to different accounts

Starting in version 1.0.9.0, when you install OneLake file explorer, you can choose which account to sign in with. To switch accounts, right-click the OneLake icon in the Windows notification area, select **Account**, and then **Sign Out**. Signing out exits OneLake file explorer and pauses the sync. To sign in with another account, start OneLake file explorer again and choose the desired account.

When you sign in with another account, you see the list of workspaces and items refresh in OneLake file explorer. If you navigate to workspaces associated with the previous account, you

can manually refresh the view by right-clicking the workspace, then selecting **OneLake > Sync from OneLake**. Those workspaces are inaccessible while you're signed in to a different account.

Option to open workspaces and items on the web portal

Starting in version 1.0.10.0, you can transition between using OneLake file explorer and the Fabric web portal. From OneLake file explorer, right-click on a workspace and select **OneLake > View workspace online**. This action opens the workspace browser on the Fabric web portal.

You can right-click on an item, subfolder, or file and select **OneLake > View item online**. This action opens the item browser on the Fabric web portal. If you select a subfolder or file, the Fabric web portal always opens the root folder of the item.

Offline support

The OneLake file explorer only syncs updates when you're online and the application is running. When the application starts, the views for any folders that were previously synced refresh automatically. Any files that you added or updated while offline show as sync pending until you save them again. Any files that you deleted while offline are recreated during the refresh if they still exist on the service.

Create files or folders in OneLake file explorer

1. Navigate to the **OneLake** section in Windows File Explorer.
2. Navigate to the appropriate folder in your item.
3. Right-click and select **New folder** or **New file type**.

(!) Note

OneLake sync fails if you write data to locations where you don't have write permission, such as the root of the item or workspace. Clean up files or folders that fail to sync by moving them to the correct location or deleting them.

Delete files or folders in OneLake file explorer

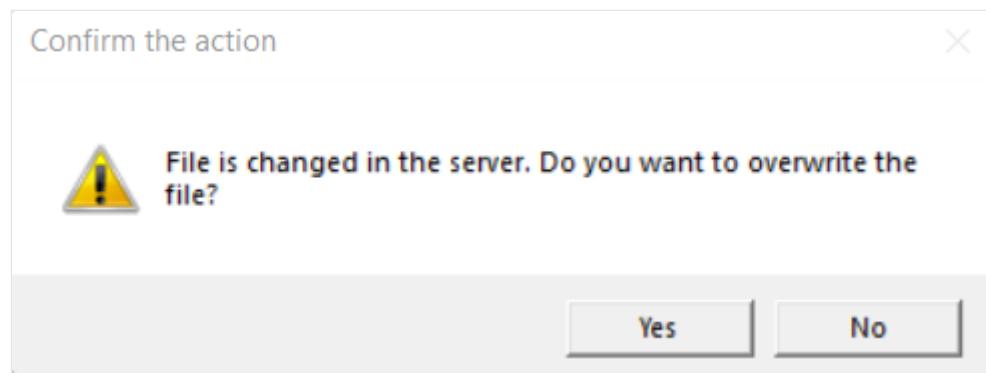
1. Navigate to the **OneLake** section in Windows File Explorer.
2. Navigate to the **Files** or **Tables** folder in your item.

3. Select a file or folder and delete.

Edit files

You can open files using your favorite apps and make edits. Select **Save** to sync the file to OneLake. Starting in version 1.0.11, you can also make updates with Excel to your files. **Close** the file after the update in Excel and it initiates the sync to OneLake.

If you edit a file locally and select **Save**, the OneLake file explorer app detects if that file was updated elsewhere (by someone else) since you last selected **Sync from OneLake**. A **Confirm the action** dialog box appears:



If you select **Yes**, your local changes overwrite any other changes made to the file since the last time you selected **Sync from OneLake**.

If you select **No**, the local changes aren't sent to the OneLake service. You can then select **Sync from OneLake** to revert your local changes and pull the file from the service. Or you can copy the file with a new name to avoid conflicts.

Copy or move files

You can copy files to, from, and within your items using standard keyboard shortcuts like **Ctrl+C** and **Ctrl+V**. You can also move files by dragging and dropping them.

Support for large files and a large number of files

When you upload or download files using the OneLake file explorer, the performance should be similar to using OneLake APIs. In general, the time it takes to sync changes from OneLake is proportional to the number of files.

OneLake shortcut support

All folders in your items including [OneLake shortcuts](#) are visible. You can view, update, and delete the files and folders in those shortcuts.

Client-side logs

Starting in version 1.0.10, you can find your client-side logs by right-clicking on the OneLake icon in the Windows notification area, located at the far right of the taskbar. Select **Diagnostic operations > Open logs folder**. This action opens your logs directory in a new Windows file explorer window.

Client-side logs are stored on your local machine under `%temp%\OneLake\Diagnostics\`.

You can enable more client-side logging by selecting **Diagnostic operations > Enable tracing**.

Release Notes

Starting in version 1.0.11, you can find information about each release of the OneLake file explorer by right-clicking on the OneLake icon in the Windows notification area, located at the far right of the taskbar. Select **About > Release Notes**. This action opens the OneLake file explorer release notes page in your browser window.

Uninstall instructions

To uninstall the app, search for OneLake in Windows. Select **Uninstall** in the list of options under **OneLake**.

Tenant setting enables access to OneLake file explorer

Tenant admins can restrict access to OneLake file explorer for their organization in the [Microsoft Fabric admin portal](#). When the setting is disabled, no one in your organization can start the OneLake file explorer app. If the application is already running and the tenant admin disables the setting, the application exits. Placeholders and any downloaded content remain on local machines, but users can't sync data to or from OneLake.

OneLake file explorer icons

These OneLake file explorer icons appear in Windows File Explorer to indicate the sync state of the file or folder.

| Icon | Icon description | Meaning |
|---|---------------------|--|
|  | Blue cloud icon | The file is only available online. Online-only files don't take up space on your computer. |
|  | Green tick | The file is downloaded to your local computer. |
|  | Sync pending arrows | Sync is in progress. This icon might appear when you're uploading files. If the sync pending arrows are persistent, then your file or folder might have an error syncing. You can find more information in the client-side logs on your local machine under %temp%\OneLake\Diagnostics\. |

Related content

- Learn more about [Fabric and OneLake security](#).
- [What's new in the latest OneLake file explorer?](#)

What's new in the latest OneLake file explorer?

09/02/2025

Continue reading for information on major updates to OneLake file explorer.

Important

This feature is in [preview](#).

September 2025 Update (v 1.0.14.0)

.NET 8 Migration

OneLake File Explorer has been migrated to .NET 8, ensuring the application remains aligned with Microsoft's latest supported frameworks. This upgrade enhances security and ensures continued eligibility for long-term support.

Smarter sync on temporary files

Temporary files (.tmp) created during file edits— often created during edits in applications like Microsoft Excel —will no longer get stuck in the sync pending state. This issue typically occurred when there were network connectivity problems, preventing .tmp files from uploading and causing conflicts with the server. These files are now cleaned up automatically once the network connection is restored, eliminating persistent sync indicators on temporary files.

Stability improvements

We've resolved a memory access violation that previously caused crashes when syncing files that had not been opened before. This fix improves reliability and prevents unexpected shutdowns during background sync operations.

April 2024 Update

v 1.0.13.0 - Update Notifications

We believe that staying informed about app updates is crucial. Whether it's a bug fix, performance improvement, or exciting new features. Starting with this version, the OneLake file explorer app will now notify you when a new update is available. You'll receive a Windows notification when a new version is available and the OneLake icon in the Windows notification area will change. Simply right-click the icon to see if an update is available.

v 1.0.12.0 - Internal update

This release includes minor internal fixes to enhance functionality.

December 2023 Update (v 1.0.11.0)

Ability to update files using Excel

With this release users can make edits and updates with Excel to OneLake files, similar to the experience with OneDrive. Start by opening a csv or xlsx file using Excel, make updates and close the file. Closing the file will initiate the sync to OneLake. You can then view the updated file online in the Fabric web portal. This enhancement aims to streamline your workflow and provide a more intuitive approach to managing and editing your files with Excel.

Menu option to view Release Notes

With this menu option, you can easily find details about what's new in the latest OneLake File Explorer version. Right-click on the OneLake icon in the Windows notification area, located at the far right of the taskbar, and select **About > Release Notes**. This action opens the OneLake file explorer release notes page in your browser window.

TLS 1.3 support

OneLake file explorer will default to the latest TLS version supported by Windows, currently TLS 1.3. Support for TLS 1.3 is recommended for maintaining the security and privacy of data exchanged over the internet.

September 2023 Update (v 1.0.10.0)

Option to open workspaces and items on the web portal

Now you can seamlessly transition between using OneLake file explorer and the Fabric web portal. Browse your OneLake data using OneLake file explorer, right-click on a workspace, and

select **OneLake > View Workspace Online**. This action opens the workspace browser on the Fabric web portal.

In addition, you can right-click on an item, subfolder, or file and select **OneLake > View Item Online**. This action opens the item browser on the Fabric web portal. If you select a subfolder or file, the Fabric web portal always opens the root folder of the item.

Menu option to open logs directory

With this menu option, you can now easily find your client-side logs, which can help you troubleshoot issues. Right-click on the OneLake icon in the Windows notification area, located at the far right of the taskbar, and select **Select Diagnostic Operations > Open logs directory**. This action opens your logs directory in a new Windows file explorer window.

July 2023 Update (v 1.0.9.0)

Option to sign in to different accounts

When you install OneLake file explorer, you can now choose which account to sign-in with. To switch accounts, right-click the OneLake icon in the Windows notification area, select **Account**, and then select **Sign Out**. Signing out exits OneLake file explorer and pauses the sync. To sign in with another account, start OneLake file explorer again by searching for "OneLake" using Windows search (Windows + S) and select the OneLake application. Previously, when you started OneLake file explorer, it automatically used the Microsoft Entra ID currently logged into Windows to sync Fabric workspaces and items.

When you sign in with another account, you see the list of workspaces and items refresh in OneLake file explorer. If you continue to workspaces associated with the previous account, you can manually refresh the view by selecting **Sync from OneLake**. Those workspaces are inaccessible while you're signed into a different account.

Fix known issue during folder moves from outside of OneLake to OneLake

Now when you move a folder (cut and paste or drag and drop) from a location outside of OneLake to OneLake, the contents sync to OneLake successfully. Previously, you had to trigger a sync by either opening the files and saving them or moving them back out of OneLake and then copying and pasting (versus moving).

Support lifecycle

Only the most recent version of OneLake file explorer is supported. If you contact support for OneLake file explorer, they ask you to upgrade to the most recent version. Download the latest [OneLake file explorer ↗](#).

Related content

- [Use OneLake file explorer to access Fabric data](#)

Get the size of OneLake items

Article • 05/09/2025

Learn how to get the size of your OneLake data to manage and plan storage costs. Capacity admins can use the [Microsoft Fabric Capacity Metrics app](#) to find the total size of OneLake data stored in a given capacity or workspace. But you might want a way to measure size at a more granular level.

This article describes Azure Storage PowerShell commands that you can use to understand the size of data in a specific item or folder. Because OneLake is compatible with Azure Data Lake Storage (ADLS) tools, many of the commands work by just replacing the ADLS Gen2 URL with a OneLake URL.

To automate the steps in this article, use REST API commands to get the workspace and item information instead of providing them manually. For more information, see [List workspaces](#) and [List items](#).

Prerequisites

- Azure PowerShell. For more information, see [How to install Azure PowerShell](#)
- The Azure Storage PowerShell module.

```
PowerShell  
  
Install-Module Az.Storage -Repository PSGallery -Force
```

- Sign in to PowerShell with your Azure account.

```
PowerShell  
  
Connect-AzAccount
```

Create a context object for OneLake

Each time you run an Azure Storage command against OneLake, you need to include the `-Context` parameter with an Azure Storage context object. To create a context object that points to OneLake, run the [New-AzStorageContext](#) command with the following values:

 Expand table

| Parameter | Value |
|----------------------|---|
| -StorageAccountName | 'onelake' |
| -UseConnectedAccount | None; instructs the cmdlet to use your Azure account. |
| -Endpoint | 'fabric.microsoft.com' |

For ease of reuse, create this context as a local variable:

PowerShell

```
$ctx = New-AzStorageContext -StorageAccountName 'onelake' -UseConnectedAccount -  
endpoint 'fabric.microsoft.com'
```

Get the size of an item or folder

To get an item size, use the [Get-AzDataLakeGen2ChildItem](#) command with the following values:

[+] [Expand table](#)

| Parameter | Value |
|----------------|--|
| -Context | An Azure Storage context object. For more information, see Create a context object for OneLake . |
| -FileSystem | Fabric workspace name or GUID. Azure Storage naming criteria for containers only supports lowercase letters, numbers, and hyphens. If you have any other characters in your workspace name, use its GUID instead. |
| -Path | Local path to the item or folder inside the workspace. Azure Storage naming criteria for containers only supports lowercase letters, numbers, and hyphens. If you have any other characters in any resources in your item path, use the equivalent GUID instead. |
| -Recurse | None; instructs the cmdlet to recursively get the child item. |
| -FetchProperty | None; instructs the cmdlet to fetch the item properties. |

Use a pipeline to pass the output of the [Get-AzDataLakeGen2ChildItem](#) command to the [Measure-object](#) command with the following values:

[+] [Expand table](#)

| Parameter | Value |
|-----------|---|
| -Property | Length |
| -Sum | None; indicates that the cmdlet displays the sum of the values of the specified property. |

Combined, the full command looks like the following example:

PowerShell

```
Get-AzDataLakeGen2ChildItem -Context <CONTEXT_OBJECT> -FileSystem <WORKSPACE_NAME>
-Path <ITEM_PATH> -Recurse -FetchProperty | Measure-Object -property Length -sum
```

Example: Get the size of an item

PowerShell

```
$ctx = New-AzStorageContext -StorageAccountName 'onelake' -UseConnectedAccount -
endpoint 'fabric.microsoft.com'
$workspaceName = 'myworkspace'
$itemPath = 'mylakehouse.lakehouse'
$colitems = Get-AzDataLakeGen2ChildItem -Context $ctx -FileSystem $workspaceName -
Path $itemPath -Recurse -FetchProperty | Measure-Object -property Length -sum
"Total file size: " + ($colitems.sum / 1GB) + " GB"
```

Example: Get the size of a folder

PowerShell

```
$ctx = New-AzStorageContext -StorageAccountName 'onelake' -UseConnectedAccount -
endpoint 'fabric.microsoft.com'
$workspaceName = 'myworkspace'
$itemPath = 'mylakehouse.lakehouse/Files/folder1'
$colitems = Get-AzDataLakeGen2ChildItem -Context $ctx -FileSystem $workspaceName -
Path $itemPath -Recurse -FetchProperty | Measure-Object -property Length -sum
"Total file size: " + ($colitems.sum / 1GB) + " GB"
```

Example: Get the size of a table with GUIDs

PowerShell

```
$ctx = New-AzStorageContext -StorageAccountName 'onelake' -UseConnectedAccount -
endpoint 'fabric.microsoft.com'
$workspaceName = 'aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb'
$itemPath = 'bbbbbbbb-1111-2222-3333-cccccccccc/Tables/table1'
```

```
$colitems = Get-AzDataLakeGen2ChildItem -Context $ctx -FileSystem $workspaceName -  
Path $itemPath -Recurse -FetchProperty | Measure-Object -property Length -sum  
"Total file size: " + ($colitems.sum / 1GB) + " GB"
```

Limitations

These PowerShell commands don't work on shortcuts that point to ADLS containers directly. Instead, we recommended that you create ADLS shortcuts to directories that are at least one level below a container.

OneLake catalog overview

10/23/2025

OneLake catalog is a centralized place that helps you find, explore, and use the Fabric items you need, and govern the data you own. It features two tabs:

- **Explore tab:** The explore tab has an items list with an in-context item details view that makes it possible to browse through and explore items without losing your list context. It also provides selectors and filters to narrow down and focus the list, making it easier to find what you need. By default, the OneLake catalog opens on the Explore tab. Along with the OneLake catalog, you can open and work across multiple workspaces side by side using the [object explorer](#).
- **Govern tab:** The govern tab provides insights that help you understand the governance posture of all the data you own in Fabric, and presents recommended actions you can take to improve the governance status of your data.
- **Secure tab:** The OneLake catalog Secure tab centralizes security management in Microsoft Fabric by providing a unified view of workspace roles and OneLake security roles across items. It enables admins to audit permissions, view user access, and create, edit, or delete security roles from a single location, ensuring streamlined governance and consistent enforcement of data access policies.

Open the OneLake catalog

To open the OneLake catalog, select the OneLake icon in the Fabric navigation pane. Select the tab you're interested if it's not displayed by default.

The screenshot shows the OneLake catalog interface. On the left, there's a sidebar with a 'OneLake' icon (highlighted with a red box) and sections for 'All items', 'My items', 'Endorsed items', and 'Favorites'. Below that is a 'Workspaces' section with 'All workspaces' (highlighted with a green box), 'My workspace', 'Sales Org', 'Finance Q2', 'Operations 1', 'Revenue FY24', and 'More workspaces...'. The main area displays a semantic model named 'Contoso FY21 goals'. The top navigation bar includes 'File', 'Refresh', 'Share', 'Explore this data', 'Analyze in Excel', and a search bar. The semantic model details show it was last updated on 3/20/22, 3:46:15 PM by Debra Berger. A table below lists tables: 'Goals' (Table), 'Scorecard' (Table), 'Values' (Table), 'Notes' (Table), and 'Statuses' (Table). A magnifying glass icon is in the bottom right corner.

Related content

- [Discover and explore Fabric items in the OneLake catalog](#)
- [View item details](#)
- [Govern your data in Fabric](#)
- [Endorsement](#)
- [Fabric domains](#)
- [Lineage in Fabric](#)
- [Monitor hub](#)

OneLake compute and storage consumption

10/07/2025

OneLake usage is defined by data stored and the number of transactions. For OneLake security, capacity usage is based on the number of rows in the table being secured. This page contains information on how all of OneLake usage is billed and reported.

Storage

OneLake storage is billed at a pay-as-you-go rate per GB of data used and doesn't consume Fabric Capacity Units (CUs). Fabric items like lakehouses and warehouses consume OneLake storage. For Mirroring storage, data up to a certain limit is free based on the purchased compute capacity SKU you provision. For more information about pricing, see [Fabric pricing ↗](#). For native mirrored storage, OneLake storage isn't billed as it's included in the cost of items like [Power BI import semantic models](#) and [Fabric SQL database](#).

You can visualize your OneLake storage usage in the Fabric Capacity Metrics app in the Storage tab. Also note that [soft-deleted data](#) is billed at the same rate as active data. For more information about monitoring usage, see the [Metrics app Storage page](#). To understand OneLake consumption more, see the [OneLake Capacity Consumption page](#)

Transactions

Requests to OneLake, such as reading or writing data, consume Fabric Capacity Units. The rates in this page define how much capacity units are consumed for a given type of operation.

Operation types

OneLake uses the same [mappings](#) as Azure Data Lake Storage (ADLS) to classify the operation to the category.

OneLake supports two access paths: redirect and proxy. Applications will use one of these paths based on specific circumstances, some determined by the workload and others outside its control. Requests via proxy and redirect share the same consumption rate.

This table defines CU consumption when OneLake data is accessed using applications that redirect certain requests.

[Expand table](#)

| Operation in Metrics App | Description | Operation Unit of Measure | Consumption rate |
|---------------------------------------|---------------------------------------|---------------------------|------------------|
| OneLake Read via Redirect | OneLake Read via Redirect | Every 4 MB, per 10,000* | 104 CU seconds |
| OneLake Write via Redirect | OneLake Write via Redirect | Every 4 MB, per 10,000* | 1626 CU seconds |
| OneLake Other Operations via Redirect | OneLake Other Operations via Redirect | Per 10,000 | 104 CU seconds |
| OneLake Iterative Read via Redirect | OneLake Iterative Read via Redirect | Per 10,000 | 1626 CU seconds |
| OneLake Iterative Write via Redirect | OneLake Iterative Write via Redirect | Per 100 | 1300 CU seconds |

This table defines CU consumption when OneLake data is accessed using applications that proxy requests, which match the rate for transactions via redirect.

[Expand table](#)

| Operation in Metrics App | Description | Operation Unit of Measure | Consumption rate |
|-----------------------------------|-----------------------------------|---------------------------|------------------|
| OneLake Read via Proxy | OneLake Read via Proxy | Every 4 MB, per 10,000* | 104 CU seconds |
| OneLake Write via Proxy | OneLake Write via Proxy | Every 4 MB, per 10,000* | 1626 CU seconds |
| OneLake Other Operations | OneLake Other Operations | Per 10,000 | 104 CU seconds |
| OneLake Iterative Read via Proxy | OneLake Iterative Read via Proxy | Per 10,000 | 1626 CU seconds |
| OneLake Iterative Write via Proxy | OneLake Iterative Write via Proxy | Per 100 | 1300 CU seconds |

*For files > 4 MB in size, OneLake counts a transaction for every 4 MB block of data read or written. For files < 4 MB, a full transaction is counted. For example, if you do 10,000 read operations via Redirect and each file read is 16 MB in size, your capacity consumption is 40,000 transactions or 416 CU seconds.

Shortcuts

When you access data via OneLake shortcuts, the transaction usage counts against the capacity tied to the workspace where the shortcut is created. The capacity where the data is ultimately stored (that the shortcut points to) is billed for the data stored.

When you access data via a shortcut to a source external to OneLake, such as to ADLS Gen2, OneLake does not count the CU usage for that external request. The transactions would be charged directly to you by the external service such as ADLS Gen2.

Paused Capacity

When a capacity is paused, the data stored continues to be billed using the pay-as-you-go rate per GB. All transactions to that capacity are rejected when it is paused, so no Fabric CUs are consumed due to OneLake transactions. To access your data or delete a Fabric item, the capacity needs to be resumed. You can delete the workspace while a capacity is paused.

The consumption of the data via shortcuts is always counted against the consumer's capacity, so the capacity where the data is stored can be paused without disrupting downstream consumers in other capacities. See an example on the [OneLake Capacity Consumption page](#)

Disaster recovery

OneLake usage when disaster recovery is enabled is also defined by the amount of data stored and the number of transactions.

Disaster recovery storage

When disaster recovery is enabled, the data in OneLake gets geo-replicated. Thus, the storage is billed as Business Continuity and Disaster Recovery (BCDR) Storage. For more information about pricing, see [Fabric pricing](#).

Disaster recovery transactions

When disaster recovery is enabled for a given capacity, write operations consume higher capacity units.

Disaster recovery operation types

This table defines CU consumption when disaster recovery is enabled and OneLake data is accessed using applications that redirect certain requests. Redirection is an implementation that reduces consumption of OneLake compute.

[Expand table](#)

| Operation | Description | Operation Unit of Measure | Capacity Units |
|--|--|---------------------------|-----------------|
| OneLake BCDR Read via Redirect | OneLake BCDR Read via Redirect | Every 4 MB, per 10,000 | 104 CU seconds |
| OneLake BCDR Write via Redirect | OneLake BCDR Write via Redirect | Every 4 MB, per 10,000 | 3056 CU seconds |
| OneLake BCDR Other Operations Via Redirect | OneLake BCDR Other Operations Via Redirect | Per 10,000 | 104 CU seconds |
| OneLake BCDR Iterative Read via Redirect | OneLake BCDR Iterative Read via Redirect | Per 10,000 | 1626 CU seconds |
| OneLake BCDR Iterative Write via Redirect | OneLake BCDR Iterative Write via Redirect | Per 100 | 2730 CU seconds |

This table defines CU consumption when disaster recovery is enabled and OneLake data is accessed using applications that proxy requests, which match the rate for transactions via redirect.

[Expand table](#)

| Operation | Description | Operation Unit of Measure | Capacity Units |
|--|--|---------------------------|-----------------|
| OneLake BCDR Read via Proxy | OneLake BCDR Read via Proxy | Every 4 MB, per 10,000 | 104 CU seconds |
| OneLake BCDR Write via Proxy | OneLake BCDR Write via Proxy | Every 4 MB, per 10,000 | 3056 CU seconds |
| OneLake BCDR Other Operations | OneLake BCDR Other Operations | Per 10,000 | 104 CU seconds |
| OneLake BCDR Iterative Read via Proxy | OneLake BCDR Iterative Read via Proxy | Per 10,000 | 1626 CU seconds |
| OneLake BCDR Iterative Write via Proxy | OneLake BCDR Iterative Write via Proxy | Per 100 | 2730 CU seconds |

OneLake security

OneLake security consumes capacity for row level security (RLS) transactions based on the number of rows in the table secured by RLS. When you access a table secured with RLS, the

capacity consumption applies to the Fabric item used to execute the query according to the table below.

[+] Expand table

| Operation | Description | Operation Unit of Measure | Capacity Units |
|----------------------|----------------------|---------------------------|----------------|
| OneLake security RLS | OneLake security RLS | Million rows in the table | 0.1 CU seconds |

OneLake diagnostics

OneLake diagnostics consumes capacity when diagnostic events are captured and written to a destination Lakehouse according to the table below.

[+] Expand table

| Operation | Description | Operation Unit of Measure | Capacity Units |
|--|--|---------------------------|-----------------|
| OneLake Diagnostics Event Operation | OneLake diagnostic write operations | Every 4 MB, per 10,000 | 1626 CU seconds |
| OneLake BCDR Diagnostics Event Operation | OneLake diagnostic write operations when BCDR is enabled | Every 4 MB, per 10,000 | 3056 CU seconds |
| OneLake Diagnostics Data Transfer | OneLake diagnostic data transfer | Per GB | 1.389 CU Hours |

Changes to Microsoft Fabric workload consumption rate

Consumption rates are subject to change at any time. Microsoft will use reasonable efforts to provide notice via email or through in-product notification. Changes shall be effective on the date stated in Microsoft's Release Notes or Microsoft Fabric Blog. If any change to a Microsoft Fabric Workload Consumption Rate materially increases the Capacity Units (CU) required to use a particular workload, customers may use the cancellation options available for the chosen payment method.

Related content

- [Disaster recovery guidance for OneLake](#)

Fabric capacity and OneLake consumption

Article • 12/26/2024

You only need one capacity to drive all your Microsoft Fabric experiences, including Microsoft OneLake. Keep reading if you want a detailed example of how OneLake consumes storage and compute.

Overview

[OneLake](#) comes automatically with every Fabric tenant and is designed to be the single place for all your analytics data. All the Fabric data items are prewired to store data in OneLake. For example, when you store data in a lakehouse or warehouse, your data is natively stored in OneLake.

With OneLake, you pay for the data stored, similar to services like Azure Data Lake Storage (ADLS) Gen2 or Amazon S3. However, unlike other services, OneLake doesn't include a separate charge for transactions (for example, reads, writes) to your data. Instead, transactions consume from existing [Fabric capacity](#) that is also used to run your other Fabric experiences. For information about pricing, which is comparable to ADLS Gen2, see [Fabric pricing](#).

To illustrate, let's walk through an example.

- Let's say you purchase an F2 SKU with 2 Capacity Units (CU) every second. Let's name this *Capacity1*.
- You then create *Workspace1* and upload a 450 MB file to a lakehouse using the Fabric portal. This action consumes both OneLake storage and OneLake transactions.

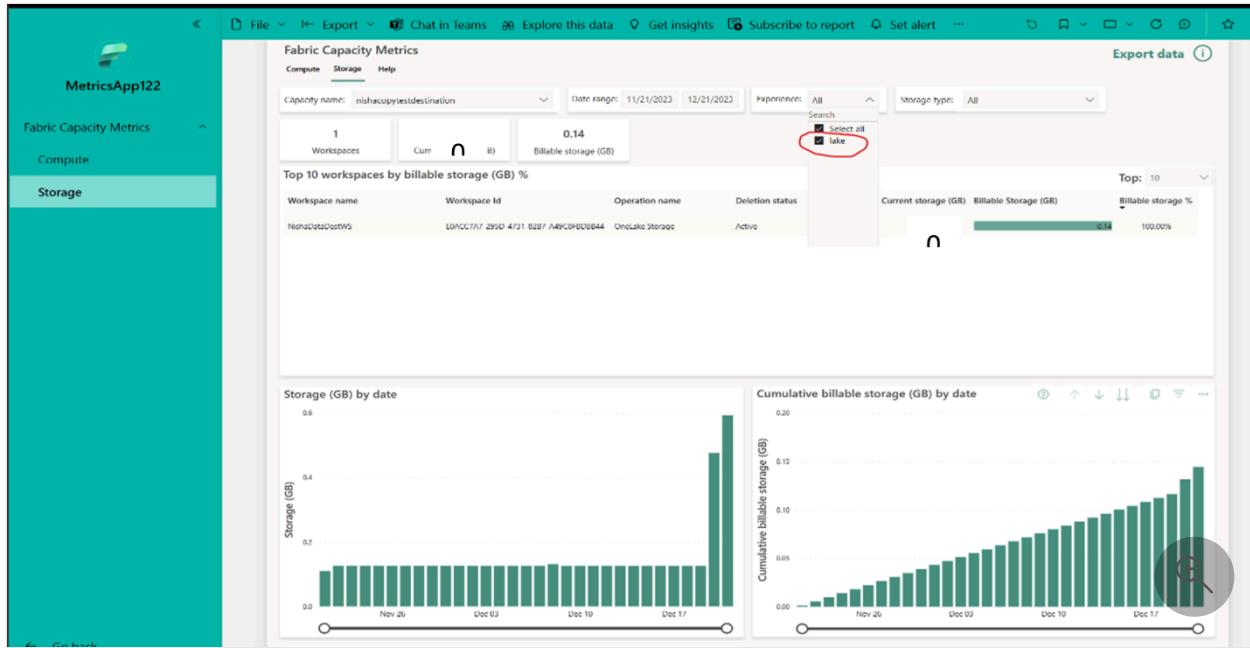
Now, let's dive into each of these dimensions.

OneLake Storage

Since OneLake storage operates on a pay-as-you-go model, a separate charge for "OneLake Storage" appears in your bill corresponding to the 450 MB of data stored.

If you're a capacity admin, you can view your storage consumption in the [Fabric Capacity Metrics app](#). Open the **Storage** tab and choose **Experience as lake** to see the

cost of OneLake storage. If you have multiple workspaces in the capacity, you can see the storage per workspace.



The following image, shows two columns: **Billable storage** and **Current Storage**. Billable storage reflects cumulative data usage over the month. Because the total charge for data stored isn't taken on one day of the month, but on a pro-rated basis throughout the month. You can estimate the monthly price as the billable storage (GB) multiplied by the price per GB per month.

For example, storing 1 TB of data on day 1, adds to 33 GB daily billable storage. On day one it's $1\text{ TB} / 30\text{ days} = 33\text{ GB}$ and every day adds 33 GB until the month ends.

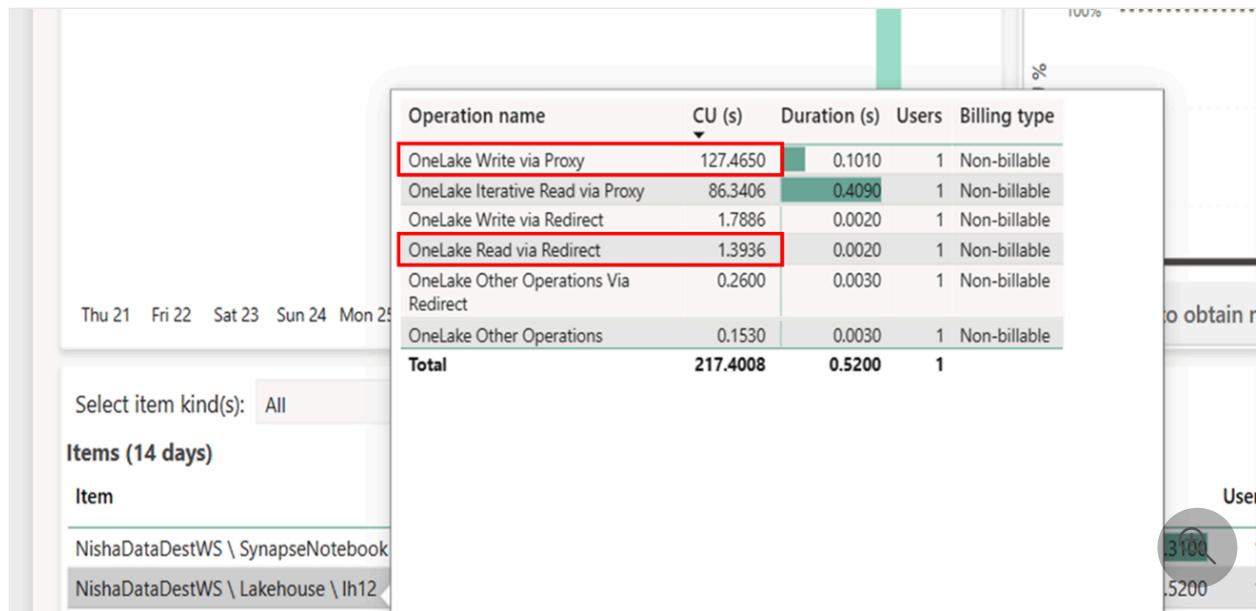
[OneLake soft delete](#) protects individual files from accidental deletion by retaining files for a default retention period before it's permanently deleted. Soft-deleted data is billed at the same rate as active data.



OneLake Compute

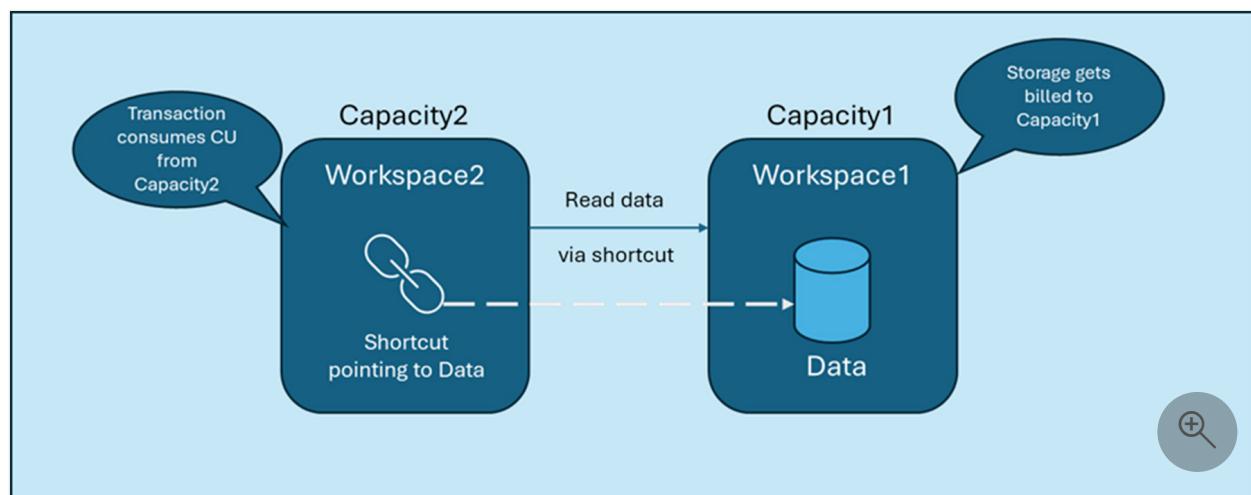
Requests to OneLake (e.g., read, write, or list) consume Fabric capacity. OneLake [maps APIs](#) to operations like ADLS. Capacity usage for each operation is visible in the Capacity Metrics app. In the above example, the file upload resulted in a write transaction consuming 127.46 CU seconds. This consumption is reported as [OneLake Write via Proxy](#) under the operation name column in the capacity metrics App.

Now if you read this data using a notebook. You consume 1.39 CU seconds of read transactions. This consumption is reported as **OneLake Read via Redirect** in the metrics app. See [OneLake consumption page](#) to learn how each type of operation consumes capacity units.



To understand more about the various terminologies on the metrics app, see [Understand the metrics app compute page - Microsoft Fabric](#).

You may be wondering, how do shortcuts affect my OneLake usage? In the above example, both storage and compute are billed to Capacity1. Now, let's say you have a second capacity *Capacity2*, that contains *Workspace2*. You create a lakehouse and create a shortcut to the parquet file you uploaded in *Workspace1*. You create a notebook to query the parquet file. As *Capacity2* accesses the data, the compute or transaction cost for this read operation consumes CU from *Capacity2*. The storage continues to be billed to *Capacity1*.



- If Capacity2 is paused but Capacity1 is active, you can't read the data via the shortcut in Workspace2 (Capacity2) but can access the data directly in Workspace1

(Capacity1).

- If Capacity1 is paused and Capacity2 is active, you can't read the data in Workspace1 (Capacity1) but you can still use the data using the shortcut in Workspace2. In both cases, as the data is still stored in Capacity1, storage costs remain billed to Capacity1

If your CU consumption exceeds the capacity limit, [throttling](#) may occur, causing transactions to be delayed or rejected temporarily.

Start Fabric's 60-day free trial to explore OneLake and other features, and visit the [Fabric forum](#) ↗ for questions.

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) ↗ | [Ask the community](#) ↗

Explore OneLake events in Fabric Real-Time hub

07/22/2025

OneLake events inform you about changes in your data lake, such as the creation, modification, or deletion of files and folders.

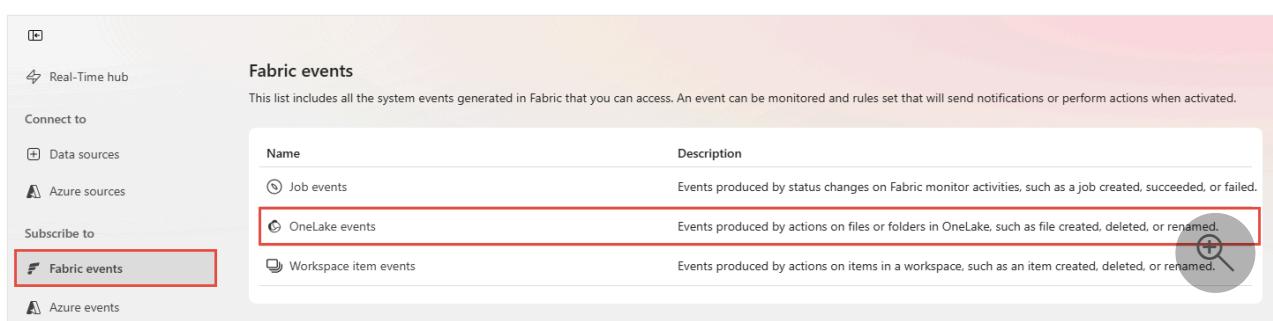
Real-Time Hub enables you to discover and subscribe to these changes within OneLake, allowing you to react instantly. For instance, you can monitor changes in Lakehouse files and folders and utilize Activator's alerting capabilities to set up alerts based on specific conditions and define actions to take when those conditions are met. This article guides you on how to explore OneLake events using the Real-Time Hub

ⓘ Note

Consuming Fabric and Azure events via Eventstream or Fabric Activator isn't supported if the capacity region of the Eventstream or Activator is in the following regions: West India, Israel Central, Korea Central, Qatar Central, Singapore, UAE Central, Spain Central, Brazil Southeast, Central US, South Central US, West US 2, West US 3.

View OneLake events detail page

1. In Real-Time hub, select **Fabric events**.
2. Select **OneLake events** from the list.



The screenshot shows the 'Fabric events' section of the Real-Time hub. On the left, there's a sidebar with 'Real-Time hub' and 'Connect to' sections. Under 'Subscribe to', the 'Fabric events' option is highlighted with a red box. The main area displays a table of events:

| Name | Description |
|-----------------------|--|
| Job events | Events produced by status changes on Fabric monitor activities, such as a job created, succeeded, or failed. |
| OneLake events | Events produced by actions on files or folders in OneLake, such as file created, deleted, or renamed. (This row is also highlighted with a red box.) |
| Workspace item events | Events produced by actions on items in a workspace, such as an item created, deleted, or renamed. |

3. You should see the detail view for OneLake events.

The screenshot shows the 'OneLake events' detail page. At the top, there are three actions: 'Return to Real-Time hub', '+ Create Eventstream', and 'Set alert'. Below these are two main sections: 'OneLake events' and 'Stay on top of your OneLake events'. The 'OneLake events' section contains a table with one row for 'sp-onelake-events'. The 'Stay on top of your OneLake events' section includes a 'Set alert' button. To the right, there's a 'OneLake events profile' sidebar with a 'Schemas' list and a search icon.

Actions

At the top of the detail page, you see the following two actions.

- **Create eventstream**, which lets you create an eventstream based on events from the selected OneLake item.
- **Set alert**, which lets you set an alert when an operation is done for a OneLake item, such as a new file is created.

This screenshot is identical to the first one, but the '+ Create Eventstream' button is highlighted with a red box to draw attention to it.

These actions are also available in the Fabric events list view.

See what's using this category

This section shows the artifacts using OneLake events. Here are the columns and their descriptions:

Expand table

| Column | Description |
|-----------|---|
| Name | Name of the artifact that's using OneLake events. |
| Type | Artifact type – Activator or Eventstream |
| Workspace | Workspace where the artifact lives. |
| Source | Name of the workspace that is source of the events. |

OneLake events profile

| Fabric OneLake events profile |
|--|
| Schemas |
| > Microsoft.Fabric.OneLake.FileCreated |
| > Microsoft.Fabric.OneLake.FileDeleted |
| > Microsoft.Fabric.OneLake.FileRenamed |
| > Microsoft.Fabric.OneLake.FolderCreated |
| > Microsoft.Fabric.OneLake.FolderDeleted |
| > Microsoft.Fabric.OneLake.FolderRenamed |

Event types

Here are the supported OneLake events:

[Expand table](#)

| Event type name | Description |
|--|---|
| Microsoft.Fabric.OneLake.FileCreated | Raised when a file is created or replaced in OneLake. |
| Microsoft.Fabric.OneLake.FileDeleted | Raised when a file is deleted in OneLake. |
| Microsoft.Fabric.OneLake.FileRenamed | Raised when a file is renamed in OneLake. |
| Microsoft.Fabric.OneLake.FolderCreated | Raised when a folder is created in OneLake. |
| Microsoft.Fabric.OneLake.FolderDeleted | Raised when a folder is deleted in OneLake. |

| Event type name | Description |
|--|---|
| Microsoft.Fabric.OneLake.FolderRenamed | Raised when a folder is renamed in OneLake. |

Schemas

An event has the following top-level data:

[Expand table](#)

| Property | Type | Description | Example |
|-------------------|-----------|---|--|
| source | string | Identifies the context in which an event happened. | /aaaaaaaa-0000-1111-2222-bbbbbbbbbbb/worksaces/bbbbbbbb-1111-2222-3333-cccccccccc/items/ccccccc-2222-3333-4444-ddddddddddd |
| subject | string | Identifies the subject of the event in the context of the event producer. | /Files/FolderA/FileName.txt |
| type | string | One of the registered event types for this event source. | Microsoft.Fabric.OneLake.FileCreated |
| time | timestamp | The time the event is generated based on the provider's UTC time. | 2017-06-26T18:41:00.9584103Z |
| id | string | Unique identifier for the event. | bbbbbbbb-1111-2222-3333-cccccccccc |
| data | object | Event data. | See the next table for details. |
| dataschemaversion | string | The version of the data schema. | 1.0 |
| specversion | string | The version of the Cloud Event spec. | 1.0 |

The `data` object has the following properties:

[Expand table](#)

| Property | Type | Description | Example |
|-----------------|--------|---|---|
| eTag | string | The value that you can use to run operations conditionally. | "\"0x8D4BCC2E4835CD0\" |
| contentLength | string | Size of the file in bytes. | 0 |
| contentType | string | Content type specified for the file. | text/plain |
| blobUrl | string | Blob URL to the path of the file. | <a href="https://onelake.blob.fabric.microsoft.com/55556666-ffff-7777-aaaa-8888bbbb9999<66667777-aaaa-8888-bbbb-9999cccc0000/Files/FolderA/File1.txt">https://onelake.blob.fabric.microsoft.com/55556666-ffff-7777-aaaa-8888bbbb9999 < 66667777-aaaa-8888-bbbb-9999cccc0000/Files/FolderA/File1.txt |
| url | string | OneLake URL to the path of the file. | <a href="https://onelake.dfs.fabric.microsoft.com/eeeeeeee-4444-5555-6666-ffffffff<aaaaaaaa-6666-7777-8888-bbbbbbbbbb/Files/FolderA/File1.txt">https://onelake.dfs.fabric.microsoft.com/eeeeeeee-4444-5555-6666-ffffffff < aaaaaaaaa-6666-7777-8888-bbbbbbbbbb/Files/FolderA/File1.txt |
| api | string | The operation that triggered the event. | CreateFile |
| clientRequestId | string | A client-provided request ID for the storage API operation. | aaaabbbb-0000-cccc-1111-dddd2222eeee |
| requestId | string | Service-generated request ID for the storage API operation. | aaaabbbb-0000-cccc-1111-dddd2222eeee |
| contentOffset | number | The offset in bytes of a write operation taken at the point where the event-triggering application completed writing to the file. | 0 |

| Property | Type | Description | Example |
|-----------|--------|---|-------------------------------|
| sequencer | string | An opaque string value representing the logical sequence of events. | 00000000000044200000000028963 |

Subscribe permission

For more information, see [subscribe permission for Fabric events](#).

Related content

- [Explore Azure blob storage events](#)

Get OneLake events in Fabric Real-Time hub

07/22/2025

This article describes how to get OneLake events as an eventstream in Fabric Real-Time hub.

Real-Time hub allows you to discover and subscribe to changes in files and folders in OneLake, and then react to those changes in real-time. For example, you can react to changes in files and folders in Lakehouse and use Activator alerting capabilities to set up alerts based on conditions and specify actions to take when the conditions are met. This article explains how to explore OneLake events in Real-Time hub.

With Fabric event streams, you can capture these OneLake events, transform them, and route them to various destinations in Fabric for further analysis. This seamless integration of OneLake events within Fabric event streams gives you greater flexibility for monitoring and analyzing activities in your OneLake.

Event types

Here are the supported OneLake events:

[] Expand table

| Event type name | Description |
|---|---|
| Microsoft.Fabric.OneLake.FileCreated | Raised when a file is created or replaced in OneLake. |
| Microsoft. Fabric.OneLake.FileDeleted | Raised when a file is deleted in OneLake. |
| Microsoft. Fabric.OneLake.FileRenamed | Raised when a file is renamed in OneLake. |
| Microsoft.Fabric.OneLake.FolderCreated | Raised created when a folder is created in OneLake. |
| Microsoft. Fabric.OneLake.FolderDeleted | Raised when a folder is deleted in OneLake. |
| Microsoft. Fabric.OneLake.FolderRenamed | Raised when a folder is renamed in OneLake. |

For more information, see [Explore OneLake events](#).

! Note

Consuming Fabric and Azure events via Eventstream or Fabric Activator isn't supported if the capacity region of the Eventstream or Activator is in the following regions: West India,

Israel Central, Korea Central, Qatar Central, Singapore, UAE Central, Spain Central, Brazil Southeast, Central US, South Central US, West US 2, West US 3.

Prerequisites

- Access to a workspace in the Fabric capacity license mode (or) the Trial license mode with Contributor or higher permissions.
- SusbcribeOneLakeEvent permission on the data sources.

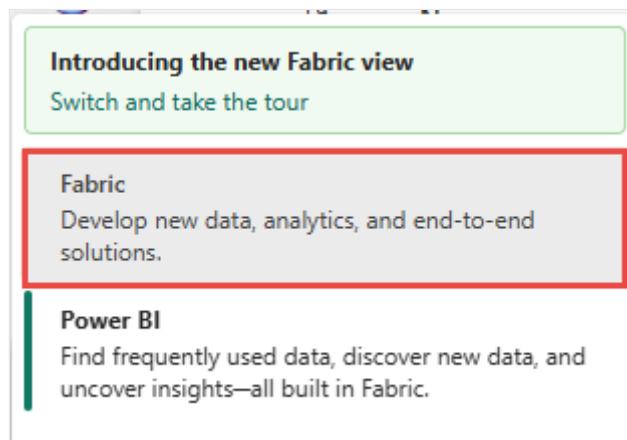
Create streams for OneLake events

You can create streams for OneLake events in Real-Time hub using one of the ways:

- [Using the Data sources page](#)
- [Using the Fabric events page](#)

Data sources page

1. Sign in to [Microsoft Fabric](#).
2. If you see **Power BI** at the bottom-left of the page, switch to the **Fabric** workload by selecting **Power BI** and then by selecting **Fabric**.



3. Select **Real-Time** on the left navigation bar.

Discover, analyze, and act on data-in-motion

Recent streaming data

| Data | Source Item | Item Owner | Workspace | Endorsement | Sensitivity |
|----------------------------------|--------------------------|------------|--------------|-------------|-------------|
| spsbusns0716-event-stream | spsbusns0716-event | Testuser1 | My workspace | — | — |
| spegridns0716-stream | spegridns0716 | Testuser1 | My workspace | — | — |
| spehubns0716_event_stream-stream | spehubns0716_event_strea | Testuser1 | My workspace | — | — |
| stock | stockseventhouse | Testuser1 | My workspace | — | — |
| stock-data-stream | stock-data | Testuser1 | My workspace | — | — |

4. On the **Real-Time hub** page, select **+ Data sources** under **Connect to** on the left navigation menu.

Data sources

Select the source needed to bring in new data and follow the short wizard to ingest the data.

+ Data sources

All Microsoft sources Database CDC Azure events Fabric events

+ Add data

Recommended

| Icon | Name | Description | Connect |
|------------|------------------------------|------------------------------|---------|
| Gift icon | Bicycle rentals | Sample scenarios (streaming) | Connect |
| Gift icon | Stock market | Sample scenarios (streaming) | Connect |
| Cloud icon | OneLake events | Fabric events | Connect |
| Cloud icon | Fabric Workspace Item events | Fabric events | Connect |

You can also get to the **Data sources** page from the **Real-Time hub** page by selecting the **+ Add data** button in the top-right corner.

Discover, analyze, and act on data-in-motion

Real-Time hub

Connect to

- Data sources
- Azure sources
- Subscribe to

 - Fabric events
 - Azure events

Recent streaming data

Filter by Data type Item Owner Item Workspace Search for streaming data Add data

| Data | Source Item | Item Owner | Workspace | Endorsement | Sensitivity |
|----------------------------------|------------------------|------------|--------------|-------------|-------------|
| spsbusns0716-event-stream | spsbusns0716-event | Testuser1 | My workspace | — | — |
| spegridns0716-stream | spegridns0716 | Testuser1 | My workspace | — | — |
| spehubns0716_event_stream-stream | spehubns0716_event_str | Testuser1 | My workspace | — | — |
| stock | stockseventhouse | Testuser1 | My workspace | — | — |
| stock-data-stream | stock-data | Testuser1 | My workspace | — | — |

- On the **Data sources** page, select **OneLake events** category at the top, and then select **Connect** on the **OneLake events** tile. You can also use the search bar to search for OneLake events.

Real-Time hub

Data sources

Select the source needed to bring in new data and follow the short wizard to ingest the data.

Search

All Microsoft sources Database CDC Azure events Fabric events Sample scenarios

F

Fabric Workspace Item events Fabric events Connect

Route your Fabric Workspace Item events, such as creation or deletion, to Fabric.

J

Job events Fabric events Connect

Capture, transform, and route your Job events to various destination in Fabric.

O

OneLake events Fabric events Connect

Route your Fabric OneLake events, such as creation or deletion, to Fabric.

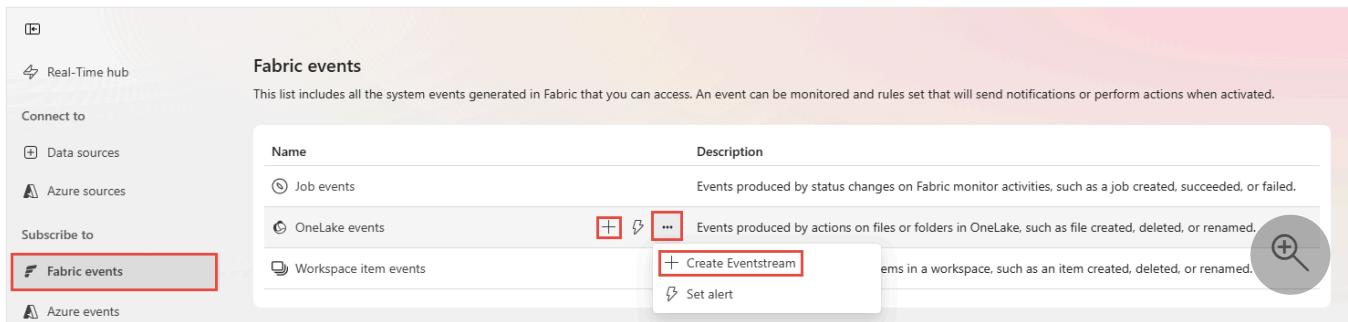
Now, use instructions from the [Configure and create an eventstream](#) section.

Fabric events page

In Real-Time hub, select **Fabric events** on the left menu. You can use either the list view of Fabric events or the detail view of OneLake events to create an eventstream for OneLake events.

Using the list view

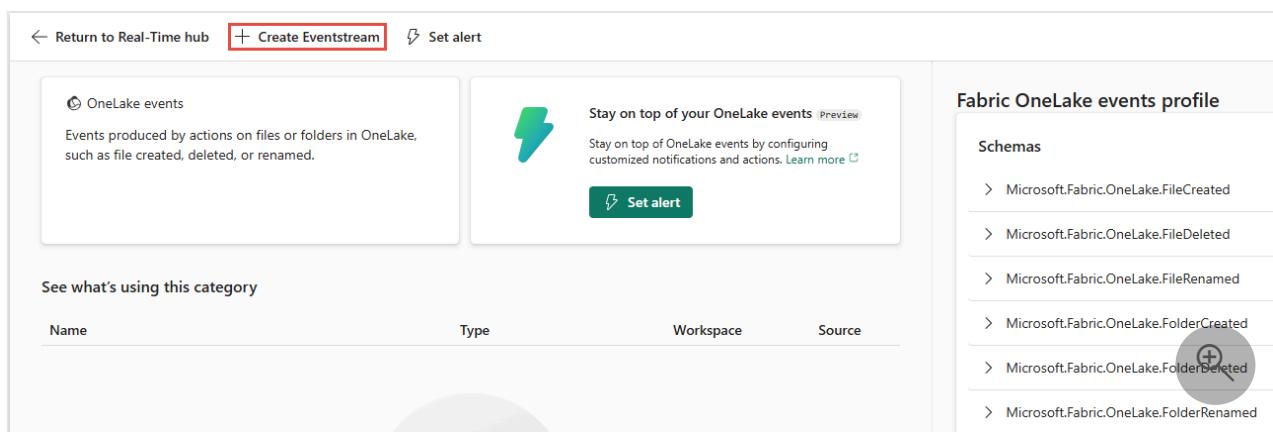
Move the mouse over **OneLake events**, and select the **Create Eventstream** link or select ... (Ellipsis) and then select **Create Eventstream**.



The screenshot shows the 'Fabric events' list view. On the left, there are sections for 'Real-Time hub', 'Connect to' (Data sources, Azure sources), and 'Subscribe to' (Fabric events, Azure events). The 'Fabric events' section is highlighted with a red box. In the main list, there are three rows: 'Job events', 'OneLake events', and 'Workspace item events'. The 'OneLake events' row has a red box around it. To its right is a small card with a plus sign and a lightning bolt icon, followed by a red box. Below this card is another red box containing the text '+ Create Eventstream'. To the right of the list is a search bar with a magnifying glass icon.

Using the detail view

1. On the **Fabric events** page, select **OneLake events** from the list of Fabric events supported.
2. On the **Detail** page, select **+ Create eventstream** from the menu.



The screenshot shows the 'OneLake events' detail page. At the top, there are buttons for 'Return to Real-Time hub', '+ Create Eventstream' (highlighted with a red box), and 'Set alert'. Below this is a summary card for 'OneLake events' with a lightning bolt icon, the text 'Stay on top of your OneLake events', and a 'Set alert' button. To the right is a sidebar titled 'Fabric OneLake events profile' with a 'Schemas' section listing several Microsoft.Fabric.OneLake.* schema names. A circular icon with a plus sign and a cursor arrow is overlaid on the sidebar.

Now, use instructions from the [Configure and create an eventstream](#) section, but skip the first step of using the [Add source](#) page.

Configure and create an eventstream

1. On the **Connect** page, for **Event types**, select the event types that you want to monitor.

Configure connection settings

Select event type(s)

Event type(s) *

Select event type(s)

- Microsoft.Fabric.OnLake.FileCreated
- Microsoft.Fabric.OnLake.FileDeleted
- Microsoft.Fabric.OnLake.FileRenamed
- Microsoft.Fabric.OnLake.FolderCreated
- Microsoft.Fabric.OnLake.FolderDeleted
- Microsoft.Fabric.OnLake.FolderRenamed

Clear all

Stream details

Workspace: My workspace

Eventstream name: onelake_event_stream

Stream name: onelake_event_stream-stream

What is this?

Next

2. This step is optional. To see the schemas for event types, select **View selected event type schemas**. If you select it, browse through schemas for the events, and then navigate back to previous page by selecting the backward arrow button at the top.

3. Select **Add a OneLake source** under **Select data source for events**.

Configure connection settings

Select event type(s)

Event type(s) *

6 selected

View selected event type schemas

Select data source for events

Add a OneLake source

Set filters

Set the filter conditions by selecting the field(s) to watch and the alert value.

+ Filter

Stream details

Workspace: My workspace

Eventstream name: onelake_event_stream

Stream name: onelake_event_stream-stream

What is this?

Next

4. On the **Choose the data you want to connect** page:

a. View all available data sources or only your data sources (My data) or your favorite data sources by using the category buttons at the top. You can use the **Filter by keyword** text box to search for a specific source. You can also use the **Filter** button to

filter based on the type of the resource (KQL Database, Lakehouse, SQL Database, Warehouse). The following example uses the **My data** option.

- b. Select the data source from the list.
- c. Select **Next** at the bottom of the page.

OneLake catalog

Choose the data you want to connect

All Endorsed in your org My data Favorites

Filter by keyword Filter

| Name | Type | Refreshed | Next refresh | Location | Endorsement | Sensitivity |
|-------------------|--------------|-----------|--------------|--------------|-------------|--------------------|
| spkqldatabase | KQL Database | — | — | My Workspace | Promoted | Confidential\Mi... |
| spkqldb2 | KQL Database | — | — | My Workspace | — | Confidential\Mi... |
| spsamplewarehouse | Warehouse | — | — | My Workspace | — | Confidential\Mi... |
| splakehouse040824 | Lakehouse | — | — | My Workspace | — | Confidential\Mi... |
| spfabricwarehouse | Warehouse | — | — | My Workspace | — | Confidential\Mi... |

Next Cancel

5. Select all tables or a specific table that you're interested in, and then select **Add**.

Choose the data you want to connect



spsamplewarehouse

Tables

dbo

> Date

> Geography

> HackneyLicense

> Medallion

> Time

> Trip

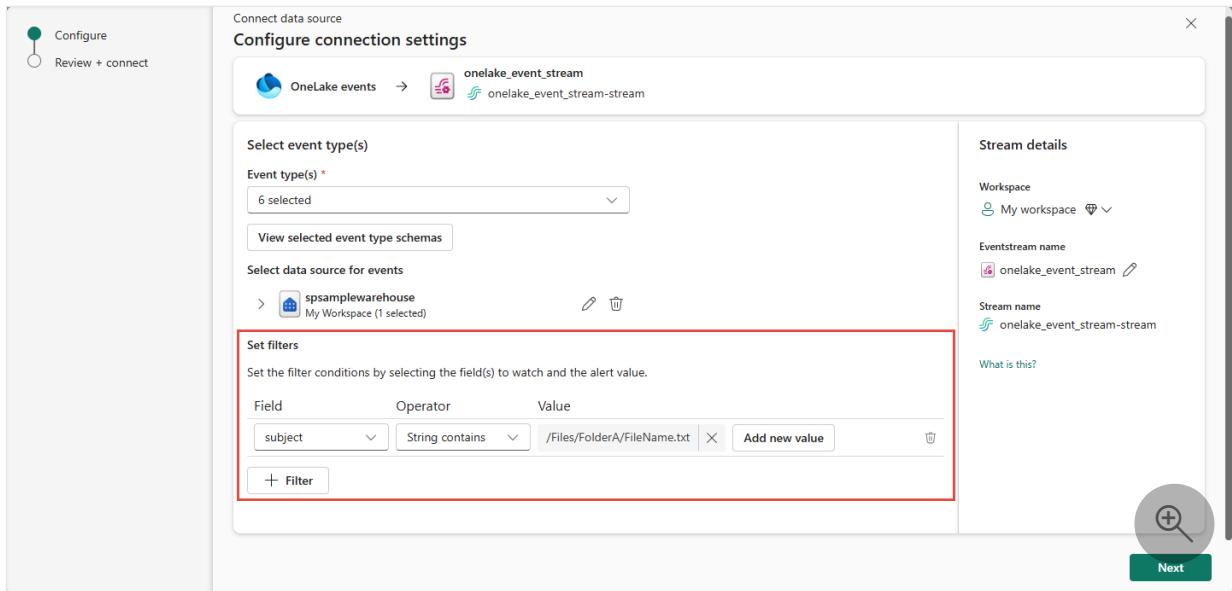
> Weather

[Previous](#)[Add](#)[Cancel](#)

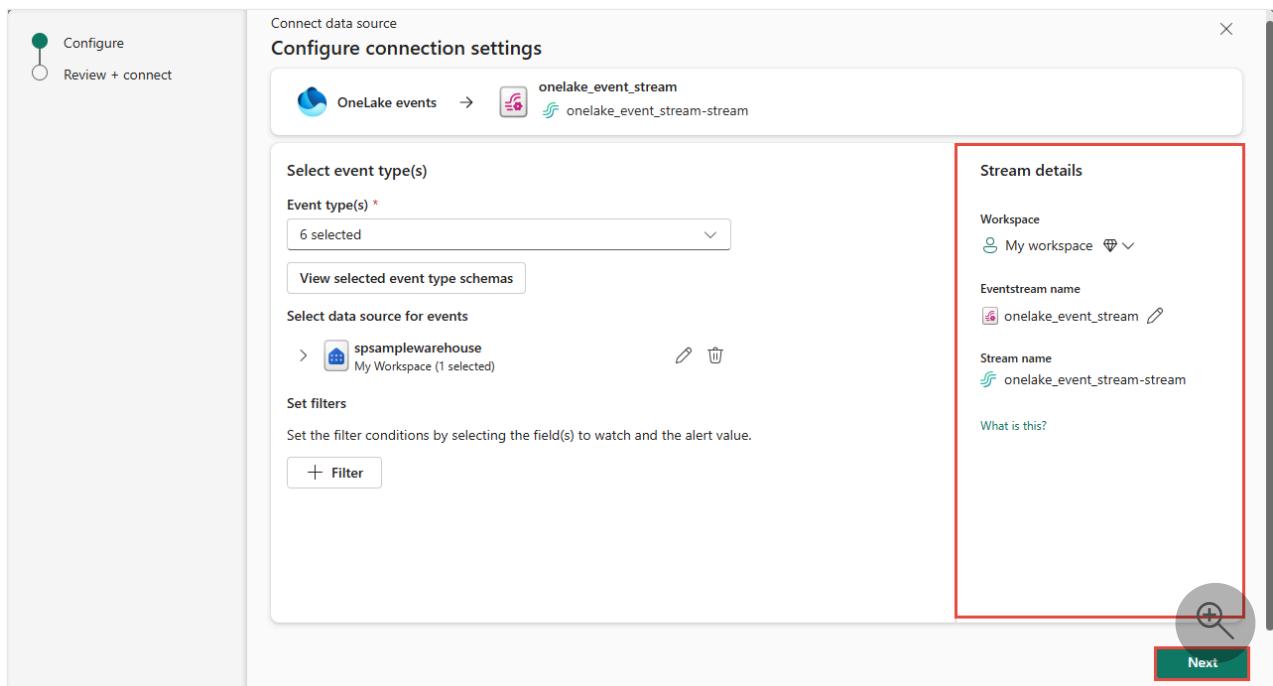
ⓘ Note

OneLake events are supported for data in OneLake. However, events for data in OneLake via shortcuts are not yet available.

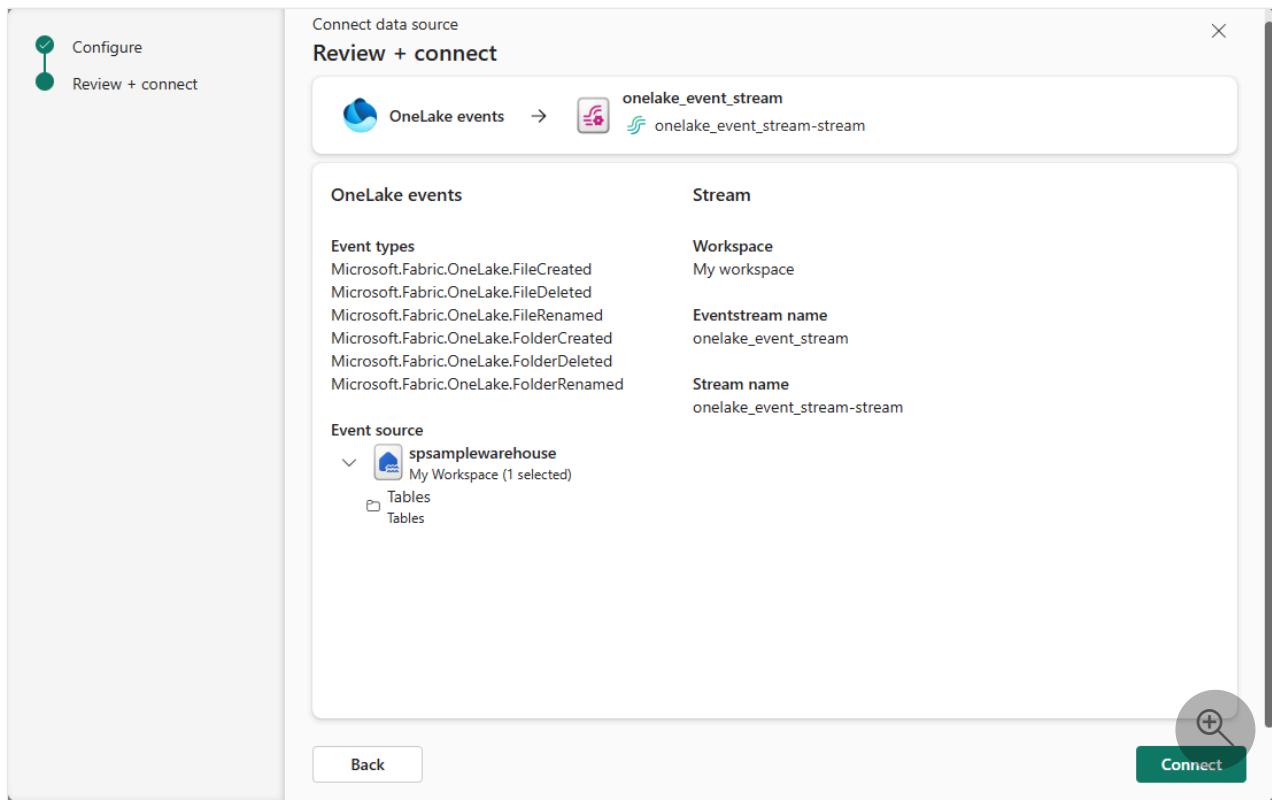
6. Now, on the **Configure connection settings** page, you can add filters to set the filter conditions by selecting fields to watch and the alert value. To add a filter:
 - a. Select **+ Filter**.
 - b. Select a field.
 - c. Select an operator.
 - d. Select one or more values to match.



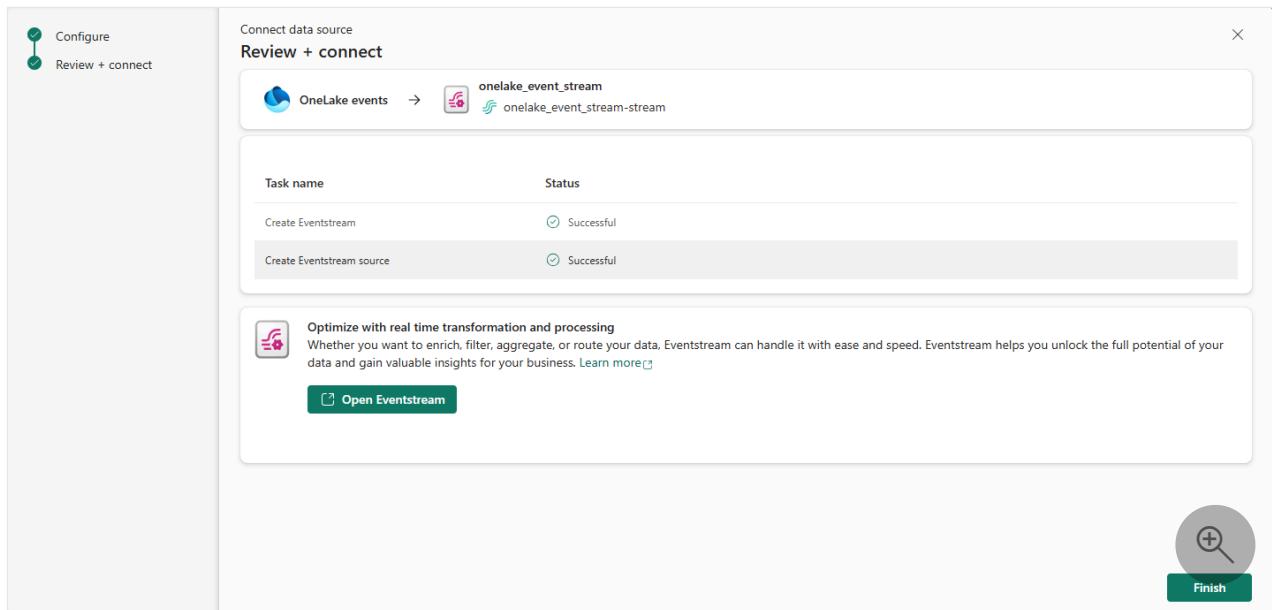
7. In the **Stream details** section to the right, follow these steps.
 - a. Select the **workspace** where you want to save the eventstream.
 - b. Enter a **name for the eventstream**. The **Stream name** is automatically generated for you.
8. Then, select **Next** at the bottom of the page.



9. On the **Review + connect** page, review settings, and select **Connect**.



10. When the wizard succeeds in creating a stream, use **Open eventstream** link to open the eventstream that was created for you. Select **Finish** to close the wizard.



View stream from the Real-Time hub page

Select **Real-Time hub** on the left navigation menu, and confirm that you see the stream you created. Refresh the page if you don't see it.

The screenshot shows the Microsoft Fabric Real-Time hub interface. On the left, there's a sidebar with various navigation options like Home, Workspaces, Copilot, Create, OneLake, Monitor, Real-Time, Workloads, and My workspace. The 'Real-Time' option is highlighted. The main area has a header 'Discover, analyze, and act on data-in-motion'. Below it, there are three cards: 'Subscribe to OneLake events' (Convert OneLake events into streams), 'Act on Job events' (Automate responses to status changes), and 'Visualize data' (Observe data in a real-time dashboard). Underneath these is a section titled 'Recent streaming data' with a table. The table has columns: Data, Source Item, Item Owner, Workspace, Endorsement, and Sensitivity. A row for 'sp-onelake-events-stream' is selected and highlighted with a red border. The table also includes rows for 'sp-fabric-workspace-events-stream', 'spsbusns0716-event-stream', 'spegridns0716-stream', and 'spehubns0716_event_stream-stream'. There are filters at the top of the table: Filter by (Data type, Item Owner, Item, Workspace), a search bar 'Search for streaming data', and a 'Add data' button. A magnifying glass icon is on the right side of the table.

| Data | Source Item | Item Owner | Workspace | Endorsement | Sensitivity |
|-----------------------------------|----------------------------|------------|--------------|-------------|-------------|
| sp-onelake-events-stream | sp-onelake-events | Testuser1 | My workspace | — | — |
| sp-fabric-workspace-events-stream | sp-fabric-workspace-events | Testuser1 | My workspace | — | — |
| spsbusns0716-event-stream | spsbusns0716-event | Testuser1 | My workspace | — | — |
| spegridns0716-stream | spegridns0716 | Testuser1 | My workspace | — | — |
| spehubns0716_event_stream-stream | spehubns0716_event_stream | Testuser1 | My workspace | — | — |

For detailed steps, see [View details of data streams in Fabric Real-Time hub](#).

Related content

To learn about consuming data streams, see the following articles:

- [Process data streams](#)
- [Analyze data streams](#)
- [Set alerts on data streams](#)

Set alerts on OneLake events in Real-Time hub

10/15/2025

This article describes how to set alerts on OneLake events in Real-Time hub.

! Note

Consuming Fabric and Azure events via Eventstream or Fabric Activator isn't supported if the capacity region of the Eventstream or Activator is in the following regions: West India, Israel Central, Korea Central, Qatar Central, Singapore, UAE Central, Spain Central, Brazil Southeast, Central US, South Central US, West US 2, West US 3.

Navigate to Real-Time hub

1. Sign in to [Microsoft Fabric](#).
2. If you see **Power BI** at the bottom-left of the page, switch to the **Fabric** workload by selecting **Power BI** and then by selecting **Fabric**.



3. Select **Real-Time** on the left navigation bar.

Launch the Set alert page

Do steps from one of the following sections, which opens a side panel where you can configure the following options:

- Events you want to monitor.
- Conditions you want to look for in the events.
- Action you want Activator to take.

Using the events list

1. In Real-Time hub, select **Fabric events**.
2. Move the mouse over **OneLake events**, and do one of the following steps:
 - Select the **Alert** button.
 - Select **ellipsis (...)**, and select **Set alert**.

| Name | Description |
|------------------------------------|--|
| Anomaly detection events (preview) | Events produced by anomaly detection models that indicate detected anomalies. |
| Job events | Events produced by status changes on Fabric monitor activities, such as a job created, succeeded, or failed. |
| OneLake events | Events produced by actions on files or folders in OneLake, such as file created, deleted, or renamed. |
| Workspace item events | Events in a workspace, such as an item created, deleted, or renamed. |

Using the event detail page

1. Select **OneLake events** from the list see the detail page.
2. On the detail page, select **Set alert** button at the top of page.

The screenshot shows the 'OneLake events' detail page. At the top right, there is a red box around the 'Set alert' button. Below it, there are two sections: 'OneLake events' (describing file operations) and 'Stay on top of your OneLake events' (describing notifications). A large central area says 'See what's using this category' with a placeholder icon and text 'Items in use will appear here'. To the right, there is a sidebar titled 'Fabric OneLake events profile' listing schemas like Microsoft.Fabric.OneLake.FileCreated, etc., and a search icon.

Details section

On the **Add rule** page, in the **Details** section, for **Rule name**, enter a name for the rule.

The screenshot shows the 'Add rule' dialog. In the 'Details' section, the 'Rule name' field contains 'spstorageaccount1013rule', which is highlighted with a red box.

Monitor section

1. In the **Monitor** section, for **Source**, choose **Select source events**.

Add rule

Details

Rule name: sponelakerule

Monitor

Source: Select source events

Condition

Check: On each event

2. In the **Connect data source** wizard, do these steps:

a. For **Event types**, select event types that you want to monitor.

Configure

Review + connect

Connect data source

Configure connection settings

OneLake events → Set alert

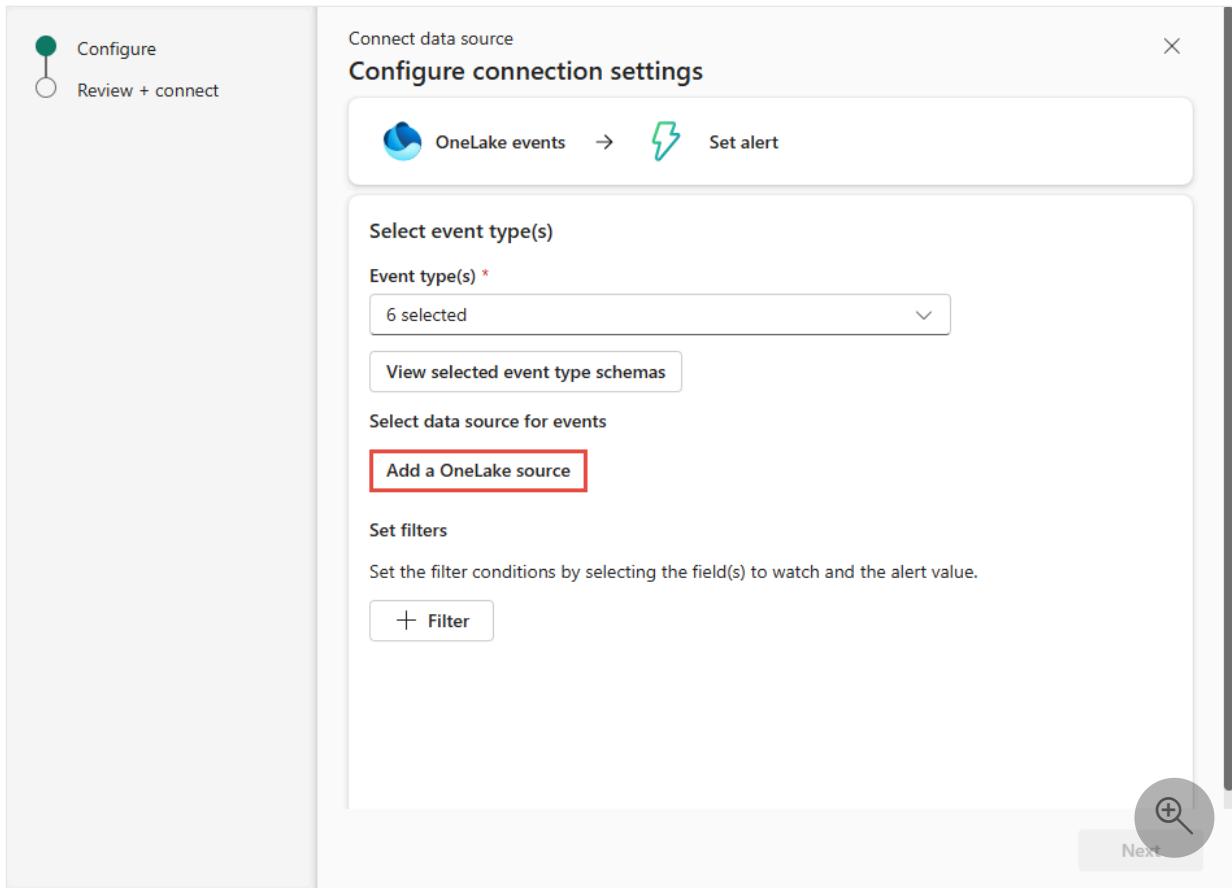
Select event type(s)

Event type(s) * Please fill out this field.

- Microsoft.Fabric.OneLake.FileCreated
- Microsoft.Fabric.OneLake.FileDeleted
- Microsoft.Fabric.OneLake.FileRenamed
- Microsoft.Fabric.OneLake.FolderCreated
- Microsoft.Fabric.OneLake.FolderDeleted
- Microsoft.Fabric.OneLake.FolderRenamed

Next

b. Select **Add a OneLake source** button in the **Select data source for events** section.



c. On the **Choose the data you want to connect** page:

- View all available data sources or only your data sources (My data) or your favorite data sources by using the category buttons at the top. You can use the **Filter by keyword** text box to search for a specific source. You can also use the **Filter** button to filter based on the type of the resource (KQL Database, Lakehouse, SQL Database, Warehouse). The following example uses the **My data** option.
- Select the data source from the list.
- Select **Next** at the bottom of the page.

| Explorer | Name | Type | Refreshed | Next refresh | Location | Endorsement | Sensitivity |
|----------|-------------------|--------------|-----------|--------------|--------------|-------------|--------------------|
| | spkqldatabase | KQL Database | — | — | My Workspace | Promoted | Confidential\Mi... |
| | spkqldb2 | KQL Database | — | — | My Workspace | — | Confidential\Mi... |
| | spsamplewarehouse | Warehouse | — | — | My Workspace | — | Confidential\Mi... |
| | splakehouse040824 | Lakehouse | — | — | My Workspace | — | Confidential\Mi... |
| | spfabricwarehouse | Warehouse | — | — | My Workspace | — | Confidential\Mi... |

iv. Select all tables or a specific table that you're interested in, and then select **Add**.

The screenshot shows the 'OneLake catalog' interface. At the top, it says 'Choose the data you want to connect'. Below that, there's a tree view of database objects:

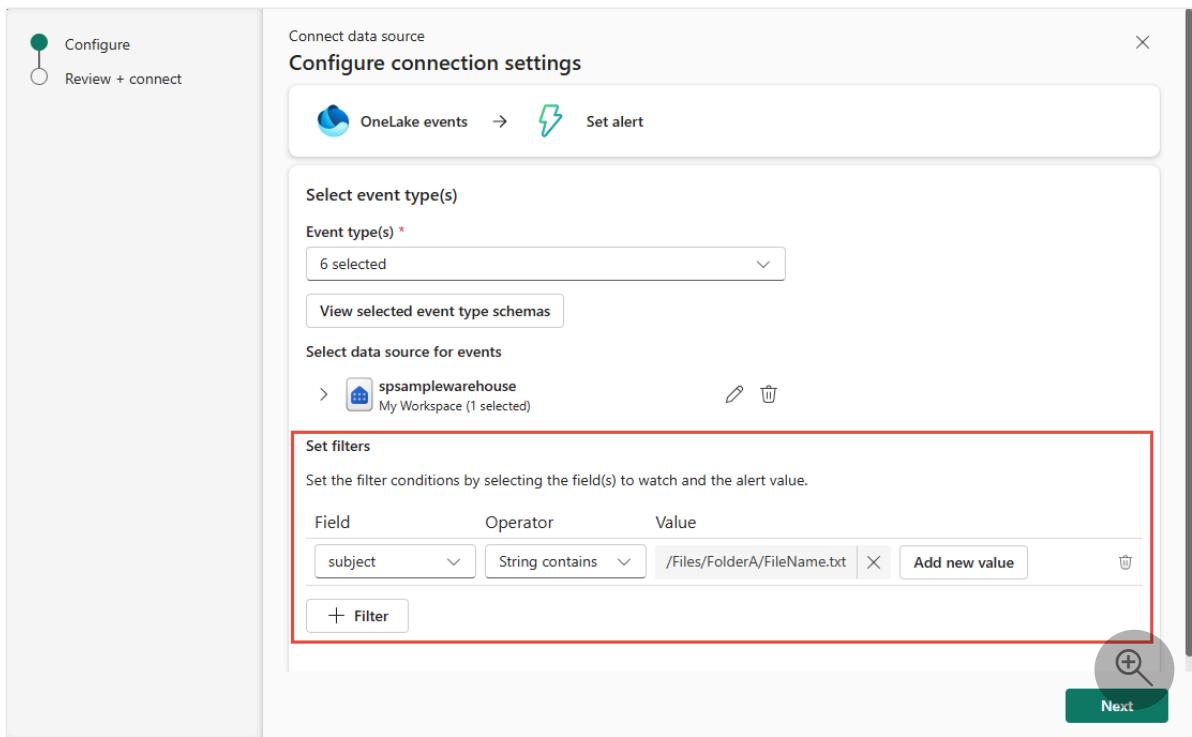
- OneLake
- spsamplewarehouse
 - Tables
 - dbo
 - Date (selected)
 - Geography
 - HackneyLicense
 - Medallion
 - Time (selected)
 - Trip
 - Weather (selected)

At the bottom right are three buttons: 'Previous', 'Add' (in a green box), and 'Cancel'.

! Note

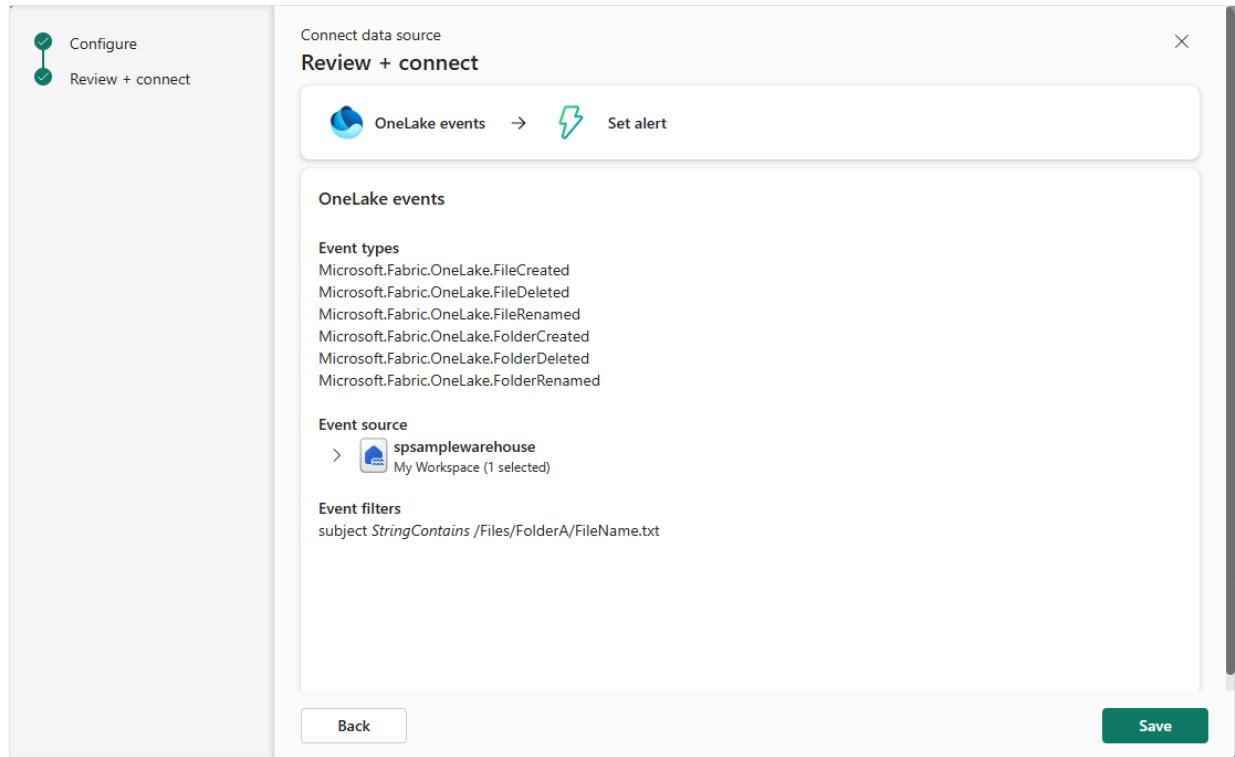
OneLake events are supported for data in OneLake. However, events for data in OneLake via shortcuts aren't yet available.

- d. Now, on the **Configure connection settings** page, you can add filters to set the filter conditions by selecting fields to watch and the alert value. To add a filter:
- Select **+ Filter**.
 - Select a field.
 - Select an operator.
 - Select one or more values to match.



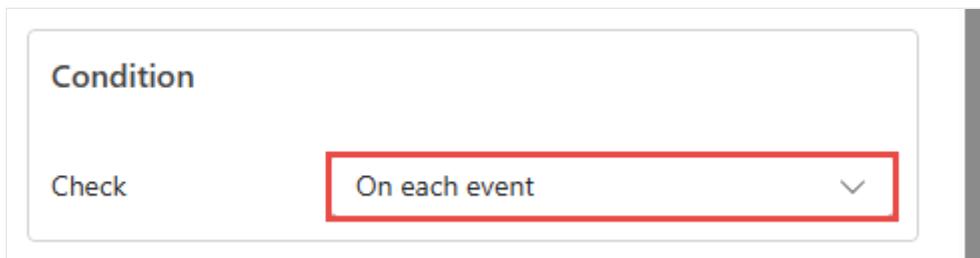
e. Select **Next** at the bottom of the page.

f. On the **Review + connect** page, review the settings, and select **Save**.



Condition section

In the **Condition** section, for **Check**, select **On each event**.



Action section

In the **Action** section, select one of the following actions:

Email

To configure the alert to send an email when the condition is met, follow these steps:

1. For **Select action**, select **Send email**.
2. For **To**, enter the **email address** of the receiver or use the drop-down list to select a property whose value is an email address.
3. For **Subject**, enter a subject for the email.
4. For **Headline**, enter a headline for the email.
5. For **Notes**, enter notes for the emails.

ⓘ Note

When entering subject, headline, or notes, you can refer to properties in the data by typing @ or by selecting the button next to the text boxes. For example, @BikepointID.

6. For **Context**, select values from the drop-down list that you want to include in the context.

Action

| | |
|---------------|--|
| Select action | Send email |
| To * | AdminUser01 X X ▾ |
| Subject * | Low on bikes alert 🔗 |
| Headline * | BikepointID X is low on bikes 🔗 |
| Notes | BikepointID X has fewer than 5 bikes. 🔗 |
| Context ⓘ | Street, Neighbourhood ▼ |

Teams message

To configure the alert to send a Teams message to an individual or a group chat or a channel when the condition is met, follow these steps:

1. For **Select action**, select **Teams** -> **Message to individuals** or **Group chat message**, or **Channel post**.
2. Follow one of these steps depending on the option you selected in the previous step:
 - If you selected the **Message to individuals** option, enter **email addresses** of receivers or use the drop-down list to select a property whose value is an email address. When the condition is met, an email is sent to specified individuals.
 - If you selected the **Group chat message** option, select a **group chat** from the drop-down list. When the condition is met, a message is posted to the group chat.
 - If you selected the **Channel post** option, select a **team** and a **channel**. When the condition is met, a message is posted in the channel.
3. For **Headline**, enter a headline for the email.
4. For **Notes**, enter notes for the emails.

Note

When entering headline, or notes, you can refer to properties in the data by typing `@` or by selecting the button next to the text boxes. For example, `@BikepointID`.

5. For **Context**, select values from the drop-down list that you want to include in the context.

Action

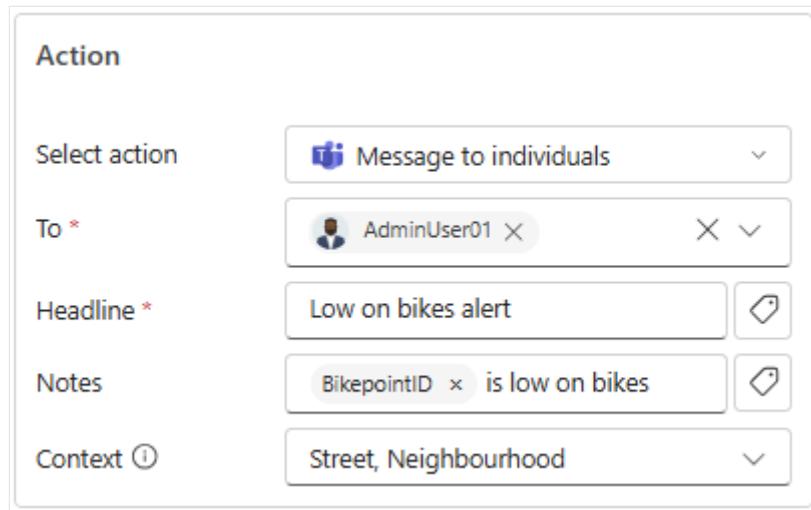
Select action **Message to individuals**

To * **AdminUser01**

Headline * **Low on bikes alert**

Notes **BikepointID is low on bikes**

Context **Street, Neighbourhood**



Run a Fabric item

To configure the alert to launch a Fabric item (pipeline, notebook, Spark job, etc.) when the condition is met, follow these steps:

1. For **Selection action**, select **Run a Fabric item**.

Action

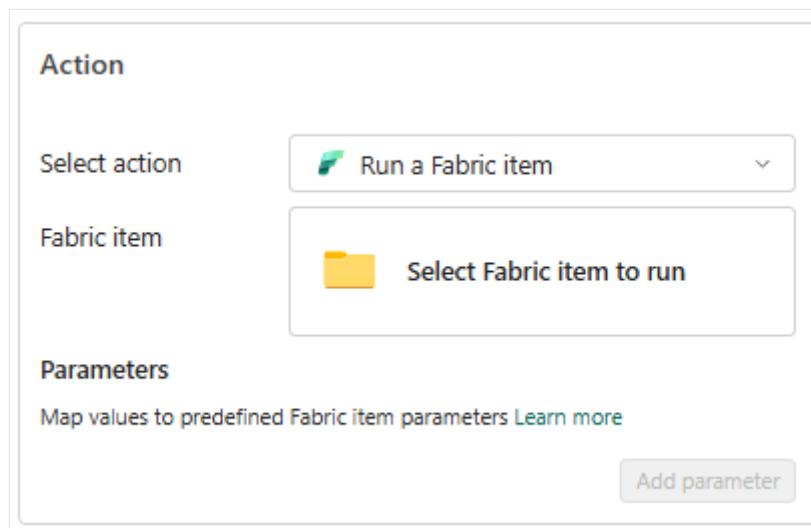
Select action **Run a Fabric item**

Fabric item **Select Fabric item to run**

Parameters

Map values to predefined Fabric item parameters [Learn more](#)

Add parameter



2. Choose **Select Fabric item to run**, and then select the Fabric item from the list.
3. Select **Add parameter** and specify the name of the parameter for the Fabric item and a value for it. You can add more than one parameter.

Action

Select action Run a Fabric item

Fabric item pipeline1
notebooksprtest

Job type * Run pipeline

Parameters

Map values to predefined Fabric item parameters [Learn more](#)

ID String BikepointID

Name * ID

Type * String

Value * BikepointID

NumberOfBikes Number No_Bikes

Name * NumberOfBikes

Type * Number

Value * No_Bikes

Add parameter

The screenshot shows the 'Run a Fabric item' configuration interface. At the top, there are three dropdown menus: 'Select action' (set to 'Run a Fabric item'), 'Fabric item' (set to 'pipeline1' under 'notebooksprtest'), and 'Job type' (set to 'Run pipeline'). Below these is a section titled 'Parameters' with a red border. Inside this section, two parameters are listed: 'BikepointID' (String type) and 'NumberOfBikes' (Number type). Each parameter has a 'Name', 'Type', and 'Value' field. The 'Value' field for 'BikepointID' contains 'BikepointID' with a delete icon. The 'Value' field for 'NumberOfBikes' contains 'No_Bikes' with a delete icon. At the bottom right of the parameter list, there is a red-bordered 'Add parameter' button.

Save location section

In the **Save location** section, for **Workspace**, select the workspace where you want to Fabric activator item to be created or that already exists. If you're creating a new activator item, enter a **name** for the activator item.

Save location

Workspace  My workspace 

Item  Create a new item

New item name

Create

Create alert

1. Select **Create** at the bottom of the page to create the alert.

Add rule

?

X

Details

Rule name

sponelakerule

Monitor

Source

OneLake events

Show event types



Show applied filters



Condition

Check

On each event



Action

Select action

- Send me an email
- Message me in Teams
- Run a Fabric item

Save location

Workspace

My workspace



Item

Create a new item



New item name

sponelakerule

Create

2. You see the **Alert created** page with a link to **open** the rule in the Fabric activator user interface in a separate tab. Select **Done** to close the **Alert created** page.

Alert created

(?) X

The alert was successfully created in sponelakerule. The alert will take action when the condition you set is met. You can open the activator to view the events.

Save location

sponelakerule

Source

My workspace event

Event types

Microsoft.Fabric.OneLake.FileCreated,
Microsoft.Fabric.OneLake.FileDeleted,
Microsoft.Fabric.OneLake.FileRenamed,
Microsoft.Fabric.OneLake.FolderCreated,
Microsoft.Fabric.OneLake.FolderDeleted,
Microsoft.Fabric.OneLake.FolderRenamed

Condition

On each event

Action

Message me in Teams

 Open

Done



3. You see a page with the activator item created by the **Add rule** wizard. If you are on the **Fabric events** page, select **Job events** to see this page.

The screenshot shows the Microsoft Fabric Real-Time hub interface. On the left, there's a sidebar with various navigation options like Home, Workspaces, Create, OneLake catalog, Monitor, Real-Time, Workloads, and My workspace. The main area has a header with 'Fabric' and 'sp-workspace-activator'. Below the header, there are sections for 'OneLake events' and 'See what's using this category'. The 'See what's using this category' section lists a single rule: 'sponelakerule' (Status: Active, Type: Activator, Workspace: My workspace, Source: Lake). A green box highlights this row. To the right, there's a 'OneLake events profile' panel with a 'schemas' section containing a list of Microsoft.Fabric.* schema names. A magnifying glass icon is in the bottom right corner of this panel.

4. Move the mouse over the Activator item, and select Open.

This screenshot shows the same interface as above, but with a context menu open over the 'sponelakerule' row in the 'See what's using this category' table. The menu items are '...', '...', and 'Open', with 'Open' highlighted by a red box. The rest of the interface remains the same, including the 'OneLake events profile' panel on the right.

5. You see the Activator item in the Fabric Activator editor user interface. Select the rule if it's not already selected. You can update the rule in this user interface. For example, update the subject, headline, or change the action from email to Teams message.

The screenshot shows the Microsoft Fabric Rules interface. At the top, there's a navigation bar with 'Fabric' and 'sp-workspace-activator' tabs, a search bar, and a user profile icon. Below the navigation is a toolbar with 'Delete', 'Edit details', 'Start', 'Stop', and 'Send me a test action' buttons.

The main area has tabs for 'Explorer', 'Live feed', 'Definition' (which is selected), 'Analytics', and 'History'. In the 'Definition' tab, there's a 'Monitor' section for 'My workspace event' and a 'Condition' section for 'On every value'. The 'Action' section is set to 'Teams message' and is triggered by 'AdminUser01'. A red box highlights the 'sponelakerule' item in the Explorer sidebar.

On the right side, there's a 'Definition' panel with sections for 'Monitor', 'Condition', and 'Action'. The 'Action' section is expanded, showing the recipient 'AdminUser01' and the message content 'The condition has been met'. Buttons for 'Edit action', 'Send me a test action', 'Reset', 'Save', and 'Save and update' are at the bottom of the panel.

Related content

- [Set alerts on Azure blob storage events](#)
- [Set alerts on Fabric workspace item events](#)

OneLake Shortcuts

Service: Core

API Version: v1

Operations

 [Expand table](#)

| | |
|---|--|
| Create Shortcut | Creates a new shortcut or updates an existing shortcut. |
| Creates Shortcuts In Bulk | Creates bulk shortcuts. |
| Delete Shortcut | Deletes the shortcut but does not delete the destination storage folder. |
| Get Shortcut | Returns shortcut properties. |
| List Shortcuts | Returns a list of shortcuts for the item, including all the subfolders exhaustively. |
| Reset Shortcut Cache | Deletes any cached files that were stored while reading from shortcuts. |

OneLake Data Access Security

Service: Core

API Version: v1

Operations

 [Expand table](#)

| | |
|--|--|
| Create Or Update Data Access Roles | Creates or updates data access roles in OneLake. |
| List Data Access Roles | Returns a list of OneLake roles. |

OneLake security for SQL analytics endpoints (Preview)

With OneLake security, Microsoft Fabric is expanding how organizations can manage and enforce data access across workloads. This new security framework gives administrators greater flexibility to configure permissions. Administrators can choose between **centralized governance through OneLake** or **granular SQL-based control** within the SQL analytics endpoint.

Access modes in SQL analytics endpoint

When using the **SQL analytics endpoint**, the selected access mode determines how data security is enforced. Fabric supports two distinct access models, each offering different benefits depending on your operational and compliance needs:

- **User identity mode:** Enforces security using OneLake roles and policies. In this mode, the SQL analytics endpoint passes the signed-in user's identity to OneLake, and **read access is governed entirely by the security rules defined within OneLake**. SQL-level permissions on tables are supported, ensuring consistent governance across tools like Power BI, notebooks, and lakehouse.
- **Delegated identity mode:** Provides full control through SQL. In this mode, the SQL analytics endpoint connects to OneLake using the identity of the **workspace or item owner**, and **security is governed exclusively by SQL permissions** defined inside the database. This model supports traditional security approaches including GRANT, REVOKE, custom roles, Row-Level Security, and Dynamic Data Masking.

Each mode supports different governance models. Understanding their implications is essential for choosing the right approach in your Fabric environment.

Comparison between access modes

Here's a clear and concise comparison table focused on how and where you set security in user identity mode versus delegated identity mode—broken down by object type and data access policies:

 Expand table

| Security target | User identity mode | Delegated identity mode |
|-----------------------------|---|---|
| Tables | Access is controlled by OneLake security roles. SQL <code>GRANT / REVOKE</code> isn't allowed. | Full control using SQL <code>GRANT / REVOKE</code> . |
| Views | Use SQL GRANT/REVOKE to assign permissions. | Use SQL GRANT/REVOKE to assign permissions. |
| Stored procedures | Use SQL GRANT EXECUTE to assign permissions. | Use SQL GRANT EXECUTE to assign permissions. |
| Functions | Use SQL GRANT EXECUTE to assign permissions. | Use SQL GRANT EXECUTE to assign permissions. |
| Row-Level Security (RLS) | Defined in OneLake UI as part of OneLake security roles. | Defined using SQL <code>CREATE SECURITY POLICY</code> . |
| Column-Level Security (CLS) | Defined in OneLake UI as part of OneLake security roles. | Defined using SQL <code>GRANT SELECT</code> with column list. |
| Dynamic Data Masking (DDM) | Not supported in OneLake security. | Defined using SQL <code>ALTER TABLE</code> with <code>MASKED</code> option. |

User identity mode in OneLake security

In user identity mode, the SQL analytics endpoint uses a **passthrough authentication mechanism** to enforce data access. When a user connects to the SQL analytics endpoint, their Entra ID identity is passed through to OneLake, which performs the permission check. All read operations against tables are evaluated using the security rules defined within the OneLake Lakehouse, not by any SQL-level GRANT or REVOKE statements.

This mode lets you manage security centrally, ensuring consistent enforcement across all Fabric experiences, including Power BI, notebooks, lakehouse, and SQL analytics endpoint. It's designed for governance models where access should be defined once in OneLake and automatically respected everywhere.

In user identity mode:

- Table access is governed entirely by OneLake security. SQL GRANT/REVOKE statements on tables are ignored.
- RLS (Row-Level Security), CLS (Column-Level Security), and Object-Level Security are all defined in the OneLake experience.
- SQL permissions are allowed for nondata objects like views, stored procedures, and functions, enabling flexibility for defining custom logic or user-facing entry points to data.

- Write operations aren't supported at the SQL analytics endpoint. All writes must occur through the Lakehouse UI and are governed by workspace roles (Admin, Member, Contributor).

 **Important**

The SQL Analytics Endpoint requires a one-to-one mapping between item permissions and members in a OneLake security role to sync correctly. If you grant an identity access to a OneLake security role, that same identity needs to have Fabric Read permission to the lakehouse as well. For example, if a user assigns "user123@microsoft.com" to a OneLake security role then "user123@microsoft.com" must also be assigned to that lakehouse.

Workspace role behavior

Users with the **Admin**, **Member**, or **Contributor** role at the workspace level aren't subject to OneLake security enforcement. These roles have elevated privileges and will bypass RLS, CLS, and OLS policies entirely. Follow these requirements to ensure OneLake security is respected:

- Assign users the **Viewer** role in the workspace, or
- Share the Lakehouse or SQL analytics endpoint with users using **read-only** permissions. Only users with read-only access have their queries filtered according to OneLake security roles.

Role precedence: Most permissive access wins

If a user belongs to **multiple OneLake roles**, the most permissive role defines their effective access. For example:

- If one role grants full access to a table and another applies RLS to restrict rows, the **RLS will not be enforced**.
- The broader access role takes precedence. This behavior ensures users aren't unintentionally blocked, but it requires careful role design to avoid conflicts. It's recommended to keep restrictive and permissive roles **mutually exclusive** when enforcing row- or column-level access controls.

For more information, see the [data access control model](#) for OneLake security.

Security sync between OneLake and SQL analytics endpoint

A critical component of user identity mode is the **security sync service**. This background service monitors changes made to security roles in OneLake and ensures those changes are reflected in the SQL analytics endpoint.

The security sync service is responsible for the following:

- Detecting changes to OneLake roles, including new roles, updates, user assignments, and changes to tables.
- Translating OneLake-defined policies (RLS, CLS, OLS) into equivalent SQL-compatible database role structures.
- Ensuring **shortcut objects** (tables sourced from other lakehouses) are properly validated so that the original OneLake security settings are honored, even when accessed remotely.

This synchronization ensures that OneLake security definitions stay authoritative, eliminating the need for manual SQL-level intervention to replicate security behavior. Because security is centrally enforced:

- You can't define RLS, CLS, or OLS directly using T-SQL in this mode.
- You can still apply SQL permissions to views, functions, and stored procedures using GRANT or EXECUTE statements.

Security sync errors & resolution

 Expand table

| Scenario | Behavior in user identity mode | Behavior in delegated mode | Corrective action | Notes |
|---|---|---|---|--|
| RLS policy references a deleted or renamed column | Error: *Row-level security policy references a column that no longer exists.*Database enters error state until policy is fixed. | Error: <i>Invalid column name <column name></i> | Update or remove one or more affected roles, or restore the missing column. | The update will need to be made in the lakehouse where the role was created. |
| CLS policy references a | Error: *Column-level security policy | Error: <i>Invalid column name</i> | Update or remove one or | The update will need to be made in the lakehouse |

| Scenario | Behavior in user identity mode | Behavior in delegated mode | Corrective action | Notes |
|---|---|---|---|---|
| deleted or renamed column | references a column that no longer exists.*Database enters error state until policy is fixed. | <column name> | more affected roles, or restore the missing column. | where the role was created. |
| RLS/CLS policy references a deleted or renamed table | Error: <i>Security policy references a table that no longer exists.</i> | No error surfaced; query fails silently if table is missing. | Update or remove one or more affected roles, or restore the missing table. | The update will need to be made in the lakehouse where the role was created. |
| DDM (Dynamic Data Masking) policy references a deleted or renamed column | DDM not supported from OneLake Security, must be implemented through SQL. | Error: <i>Invalid column name <column name></i> | Update or remove one or more affected DDM rules, or restore the missing column. | Update the DDM policy in the SQL Analytics Endpoint. |
| System error (unexpected failure) | Error: <i>An unexpected system error occurred. Try again or contact support.</i> | Error: <i>An internal error has occurred while applying table changes to SQL.</i> | Retry operation; if issue persists, contact Microsoft Support. | N/A |
| User doesn't have permission on the artifact | Error: <i>User doesn't have permission on the artifact</i> | Error: <i>User doesn't have permission on the artifact</i> | Provide user with objectID {objectID} permission to the artifact. | The object ID must be an exact match between the OneLake security role member and the Fabric item permissions. If a group is added to the role membership, then that same group must be given the Fabric Read permission. Adding a member from that group to the item does not count as a direct match. |

| Scenario | Behavior in user identity mode | Behavior in delegated mode | Corrective action | Notes |
|----------------------------------|--|--|---|---|
| User principal is not supported. | Error: <i>User principal is not supported.</i> | Error: <i>User principal is not supported.</i> | Please remove user {username} from role DefaultReader | This error occurs if the user is no longer a valid Entra ID, such as if the user has left your organization or been deleted. Remove them from the role to resolve this error. |

Shortcuts behavior with security sync

OneLake security is enforced at the source of truth, so security sync disables ownership chaining for tables and views involving shortcuts. This ensures that source system permissions are always evaluated and honored, even for queries from another database.

As a result:

- Users must have valid access on **both** the shortcut **source** (current Lakehouse or SQL analytics endpoint) **and the destination** where the data physically resides.
- If the user lacks permission on either side, **queries will fail** with an access error.
- When designing your applications or views that reference shortcuts, ensure that role assignments are properly configured on **both ends** of the shortcut relationship.

This design preserves security integrity across Lakehouse boundaries, but it introduces scenarios where access failures might occur if cross-Lakehouse roles aren't aligned.

Delegated mode in OneLake security

In **Delegated Identity Mode**, the SQL analytics endpoint uses the same **security model** that exists today in Microsoft Fabric. Security and permissions are managed entirely at the **SQL layer**, and **OneLake roles or access policies aren't enforced** for table-level access. When a user connects to the SQL analytics endpoint and issues a query:

- SQL validates access based on **SQL permissions** (GRANT, REVOKE, RLS, CLS, DDM, roles, etc.).
- If the query is authorized, the system proceeds to access data stored in **OneLake**.
- This data access is performed using the **identity of the Lakehouse or SQL analytics endpoint owner**—also known as the **item account**.

In this model:

- The signed-in user isn't passed through to OneLake.
- All enforcement of access is assumed to be handled at the SQL layer.
- The **item owner** is responsible for having sufficient permissions in OneLake to read the underlying files on behalf of the workload.

Because this is a delegated pattern, any misalignment between SQL permissions and OneLake access for the owner results in query failures. This mode provides full compatibility with:

- SQL GRANT/REVOKE at all object levels
- SQL-defined **Row-Level Security**, **Column-Level Security**, and **Dynamic Data Masking**
- Existing T-SQL tooling and practices used by DBAs or applications

How to change the OneLake access mode

The access mode determines how data access is authenticated and enforced when querying OneLake through SQL analytics endpoint. You can switch between user identity mode and delegated identity mode using the following steps:

1. Navigate to your Fabric workspace and open your lakehouse. From top right hand corner, switch from lakehouse to **SQL analytics endpoint**.
2. From the top navigation, go to **Security** tab and select the one of the following OneLake access modes:
 - **User identity** – Uses the signed-in user's identity. It enforces OneLake roles.
 - **Delegated identity** – Uses the item owner's identity; enforces only SQL permissions.
3. A pop-up launches to confirm your selection. Select **Yes** to confirm the change.

Considerations when switching between modes

Switching to user identity mode

- SQL RLS, CLS, and table-level permissions are ignored.
- OneLake roles must be configured for users to maintain access.

- Only users with Viewer permissions or shared read-only access will be governed by OneLake security.
- Existing SQL Roles are deleted and can't be recovered.

Switching to delegated identity mode

- OneLake roles and security policies are no longer applied.
- SQL roles and security policies become active.
- The item owner must have valid OneLake access, or all queries may fail.

Limitations

- **Applies only to readers:** OneLake Security governs users accessing data as *Viewers*. Users in other workspace roles (Admin Member, or Contributor) bypass OneLake Security and retain full access.
- **SQL objects do not inherit ownership:** Shortcuts are surfaced in SQL analytics endpoint as tables. When accessing these tables, directly or through views, stored procedures, and other derived SQL objects don't carry object-level ownership; all permissions are checked at runtime to prevent the security bypass.
- **Shortcut changes trigger validation downtime:** When a shortcut target changes (for example, rename, URL update), the database enters *single-user mode* briefly while the system validates the new target. During this period queries are blocked, these operations a fairly quick, but sometimes depending on different internal process can take up to 5 minutes to synchronize.
 - Creating schema shortcuts might cause a known error that affects validation and delays metadata sync.
- **Delayed permission propagation:** Permission changes aren't instantaneous. Switching between security modes (User Identity vs. Delegated) may require time to propagate before taking effect, but should take less than 1 minute.
- **Control-plane dependency:** Permissions can't be applied to users or groups that don't already exist in the workspace control plane. You either need to share the source item, or the user must be member of Viewer workspace role. Note that the exact same object ID must be in both places. A group and a member of that group do not count as a match.
- **Most-permissive access prevails:** When users belong to multiple groups or roles, the most permissive effective permission is honored *Example:* If a user has both DENY through one role and GRANT through another, the GRANT takes precedence.

- **Delegated mode limitations:** In Delegated mode, metadata sync on shortcut tables can fail if the source item has OneLake Security policies that don't grant full table access to the item owner.
- **DENY behavior:** When multiple roles apply to a single shortcut table, the intersection of permissions follows SQL Server semantics: DENY overrides GRANT. This can produce unexpected access results.
- **Expected error conditions:** Users may encounter errors in scenarios such as:
 - Shortcut target renamed or invalid
 - *Example:* If the source of table was deleted.
 - RLS (Row-Level Security) misconfiguration
 - Some expressions for RLS filtering aren't supported in OneLake and it might allow unauthorized data access.
 - Dropping the column used on the filter expression invalidates the RLS and Metadata Sync will be stale until the RLS is fixed on OneLake Security Panel.
 - For Public Preview, we only support single expression tables. Dynamic RLS and Multi-Table RLS aren't supported at the moment.
 - Column-Level Security (CLS) limitations
 - CLS works by maintaining an allowlist of columns. If an allowed column is removed or renamed, the CLS policy becomes invalid.
 - When CLS is invalid, metadata sync is blocked until the CLS rule is fixed in the OneLake Security panel.
 - Metadata or permission sync failure
 - If there are changes to the table, like renaming a column, security isn't replicated on the new object, and you receive UI errors showing that the column doesn't exist.
- **Table renames do not preserve security policies:** If OneLake Security (OLS) roles are defined on Schema level, those roles remain in effect only as long as the table name is unchanged. Renaming the table breaks the association, and security policies won't be migrated automatically. This can result in unintended data exposure until policies are reapplied.
- OneLake security roles can't have names longer than 124 characters; otherwise, security sync can't synchronize the roles.

- OneLake security roles are propagated on the SQL analytics endpoint with the OLS_ prefix.
- User changes on the OLS_ roles are not supported, and can cause unexpected behaviors.
- Mail enabled security groups and distribution lists are not supported.
- The owner of the lakehouse must be a member of the admin, member, or contributor workspace roles; otherwise, security isn't applied to the SQL analytics endpoint.
- The owner of the lakehouse cannot be a service principal for security sync to work.

Related content

- [Best practices to secure data in OneLake](#)
- [OneLake security access control model](#)

Last updated on 10/30/2025