

Get started with lakehouses in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/1-introduction>

Introduction

Completed

A **lakehouse** in Microsoft Fabric combines the flexible storage of a data lake with the analytical capabilities of a data warehouse. A lakehouse uses Apache Spark and SQL compute engines to process and analyze data at scale, and is built on the **OneLake** storage layer.

A lakehouse is a unified platform that combines:

- The flexible and scalable storage of a data **lake**
- The ability to query and analyze data of a data ware**house**

Imagine your organization relies on a traditional data warehouse for business analytics. The warehouse handles structured transactional data well, but struggles with the growing volume of semi-structured and unstructured data from sources like application logs, IoT devices, and external feeds. Storing and processing these diverse data types requires separate systems, creating data silos and complex integration efforts. Your organization needs a unified solution that handles both structured and unstructured data while maintaining strong analytical capabilities.

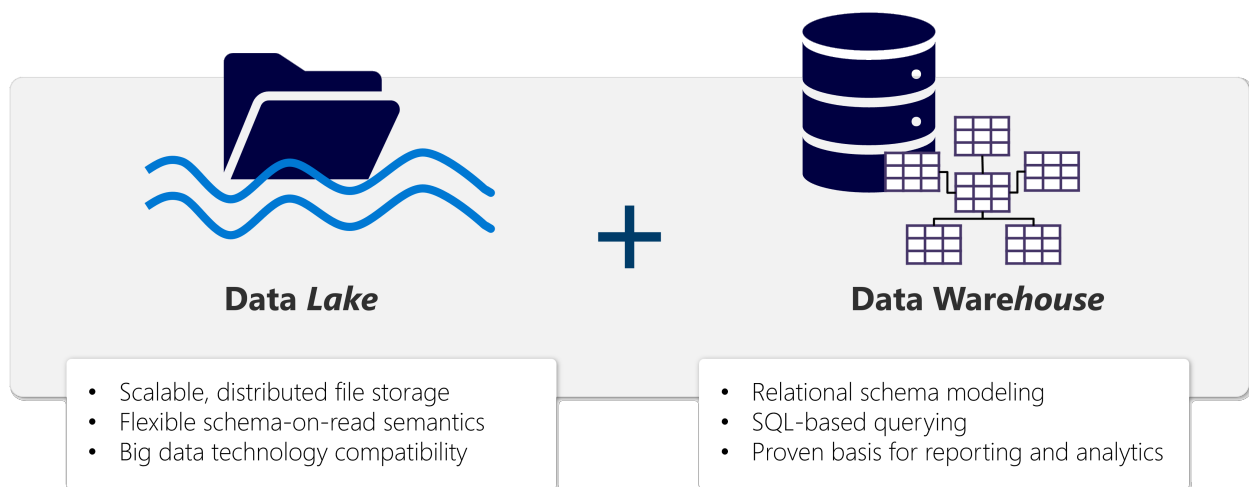
In this module, you learn how lakehouses address these challenges. You explore how to create a lakehouse, ingest and transform data, and query lakehouse data using SQL and Spark. You also learn how well-structured lakehouse data supports downstream analytics and AI-powered experiences across the Microsoft Fabric platform.

2. Describe lakehouse features and capabilities

Describe lakehouse features and capabilities

Completed

Traditional analytics architectures often force you to choose between two approaches. Data lakes offer flexibility and scalability but lack the structure and performance needed for business analytics. Data warehouses provide strong analytical capabilities but struggle with diverse data formats and can be costly to scale. **Lakehouses** bridge this gap by bringing database-like capabilities directly to your data lake, eliminating the need to maintain separate systems for different workloads.



Understand lakehouse design

A lakehouse organizes data into two main areas: **Tables** and **Files**. Understanding this separation helps you design effective data workflows.

Tables folder: This folder contains Delta Lake tables that provide structured, queryable data. Tables in this folder:

- Support SQL queries through the SQL analytics endpoint
- Enforce schemas and support ACID transactions
- Can be accessed in Power BI for reporting
- Benefit from automatic optimization and maintenance

Files folder: This folder stores raw or semi-structured data files in their native format. Files in this folder:

- Support any file format (CSV, JSON, Parquet, images, documents)
- Provide flexibility for data exploration and processing
- Can be staged before transformation into tables

- Don't enforce schema or support direct SQL queries

This separation lets you maintain both raw data (for compliance or reprocessing) and structured tables (for analytics) within the same lakehouse. You can process files using Spark notebooks or Dataflows Gen2, then load the results into tables for querying and reporting.

Understand Delta Lake tables

At the heart of a lakehouse are **Delta Lake tables**. Delta Lake is an open-source storage layer that brings reliability to data lakes. When you create a table in a lakehouse, the data is stored in Delta format in the underlying OneLake storage.

Delta Lake tables provide several key advantages:

- **ACID transactions:** Delta Lake ensures data consistency even when multiple users read and write data simultaneously.
- **Schema enforcement:** Delta Lake validates that the data you write matches the table schema, preventing corrupt data.
- **Time travel:** Delta Lake maintains a transaction log that lets you query previous versions of your data or roll back changes.
- **Efficient updates and deletes:** Unlike traditional data lake files, Delta tables support efficient update and delete operations.

Each Delta table consists of Parquet data files plus a transaction log that tracks all changes. This architecture enables both batch and streaming workloads to work reliably with the same data.

Manage lakehouse access

When you centralize data in your lakehouse, protecting that data becomes critical. Fabric provides layered access controls to secure lakehouse data at multiple levels.

Use **workspace roles** for collaborators who need access to all items in the workspace. Use **item-level sharing** to grant read-only access for specific needs, such as analytics or Power BI report development.

For granular control, the SQL analytics endpoint supports **row-level** and **column-level security**, so you can restrict what specific users see when they query through SQL. If you organize tables into schemas, you can also apply **schema-level permissions** to control access by business domain.

Fabric lakehouses also support data governance features, including sensitivity labels, and can be extended by using Microsoft Purview with your Fabric tenant.

Note

For more information, see the [Security in Microsoft Fabric](#) documentation.

Build a foundation for intelligent analytics

The data you structure in a lakehouse doesn't just serve traditional reports and dashboards. Well-organized lakehouse data becomes the foundation that intelligent experiences across Microsoft Fabric depend on.

When you create tables with clear schemas, consistent naming conventions, and descriptive column names, you make that data accessible to both human analysts and AI-powered tools. Fabric IQ data agents can query your lakehouse tables through the SQL analytics endpoint, translating natural language questions into SQL queries that return accurate answers. The quality of those answers depends directly on how well you structure and document your data.

Copilot capabilities in Fabric also benefit from well-structured lakehouse data. Copilot in Power BI can generate reports and answer business questions when it can reason over clearly defined tables and relationships. The same lakehouse data can feed semantic models that support natural language exploration across Microsoft 365 experiences.

This means the investment you make in organizing, naming, and structuring lakehouse data pays dividends beyond your immediate analytics needs. Good data engineering practices in the lakehouse create a reusable foundation for intelligent experiences across the platform.

3. Ingest and transform data in a lakehouse

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/3-work-lakehouse>

Ingest and transform data in a lakehouse

Completed

Your lakehouse needs data before it can deliver insights. Whether you're loading files from local storage, connecting to data across clouds, or building transformation pipelines, understanding ingestion and transformation techniques is essential. Start by creating a lakehouse and exploring the tools available to get your data in and ready for analysis.

Create and explore a lakehouse

You create lakehouses within a Fabric-enabled workspace. A lakehouse has two main storage areas and a SQL analytics endpoint.

- **Tables:** Stores Delta Lake tables that provide structured, queryable data. These tables support SQL queries, enforce schemas, and provide ACID transaction guarantees. You can query them

through the SQL analytics endpoint and analyze them with Power BI.

- **Files:** Stores raw or semi-structured data files in their native format (CSV, JSON, Parquet, and others). These files don't enforce a schema and provide flexibility for data exploration and processing before transformation into tables.

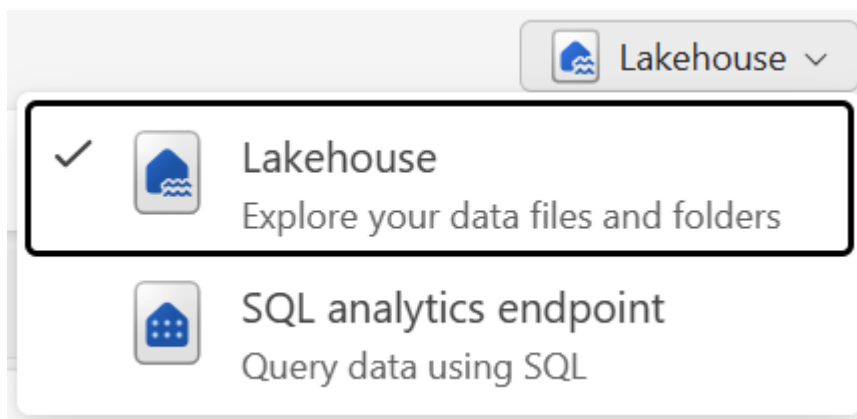
When you create a lakehouse, **schemas are enabled by default**. A schema named **dbo** is created automatically. Schemas let you organize tables into logical groups based on business domains or functions, such as `sales`, `marketing`, or `hr`. You can create other schemas to keep tables organized as your lakehouse grows. Schema-enabled lakehouses also support schema-level permissions and cross-workspace queries using the four-part namespace (`workspace.lakehouse.schema.table`).

Tip

Clear schema organization improves discoverability for everyone working with the lakehouse, including Fabric IQ data agents that translate natural language questions into SQL queries against your tables.

You can work with your lakehouse in two modes:

- **Lakehouse explorer:** Add and interact with tables, files, and folders. This mode allows you to manage data, upload files, create tables, and make changes to your lakehouse. You can also add reference lakehouses to the explorer pane, so you can browse and manage tables across multiple lakehouses side by side.
- **SQL analytics endpoint:** Query Delta tables using T-SQL in *read-only* mode. You can create views, functions, and apply SQL security, but you can't modify the underlying data.



Ingest data into a lakehouse

Ingesting data into your lakehouse is the first step in your ETL (extract, transform, load) process. Use any of the following methods to bring data into your lakehouse.

- **Upload:** Upload local files or folders directly through the lakehouse explorer.

- **Load to Table:** Select a file or folder in the lakehouse explorer and choose **Load to Table** to create a Delta table without writing code. This no-code option supports Parquet and CSV files, and lets you append or overwrite data in new or existing tables.
- **Dataflows Gen2:** Import and transform data using Power Query.
- **Notebooks:** Use Apache Spark to ingest, transform, and load data programmatically.
- **Data Factory pipelines:** Use the Copy data activity to move data from external sources.

Consider your data loading pattern when ingesting data. You can load raw data as files for staging and later processing, or load directly into tables when the data is already in a supported format.

Note

For more information on ingestion options, see the [Options to get data into the Fabric Lakehouse](#) documentation.

Access data using shortcuts

Shortcuts let you integrate data into your lakehouse without copying it. A shortcut references data in external storage and makes it appear as a folder in your lakehouse.

Shortcuts are valuable because they reduce the amount of times you have to copy data. You can create a shortcut to a different storage account, another cloud provider, or other items in Fabric. OneLake manages source data permissions and credentials. When you access data through a shortcut to another OneLake location, OneLake uses your identity to authorize access to the target data. You must have permissions in the target location to read the data.

You can also create **schema shortcuts** that map an entire schema to a folder of Delta tables in another lakehouse or in Azure Data Lake Storage Gen2. All referenced tables appear as local tables within the schema.

Note

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

Transform data in a lakehouse

Most data requires transformation before loading into tables. You might ingest raw data directly into the Files area, then transform and load it into tables. Regardless of your ETL design, you can transform and load data using the same tools available for ingestion.

- **Notebooks** are favored by data engineers familiar with programming languages including PySpark, SQL, and Scala. Copilot in notebooks can generate transformation code from natural language descriptions and explain existing Spark code.

- **Dataflows Gen2** are suited for users familiar with Power BI or Excel, since they use the Power Query interface.
- **Pipelines** provide a visual interface to orchestrate ETL processes. Pipelines can include multiple activities that run in sequence or parallel.

4. Query and analyze lakehouse data

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/4-explore-data-lakehouse>

Query and analyze lakehouse data

Completed

After you ingest and transform data, it's ready for analysis. A lakehouse in Microsoft Fabric provides multiple ways to query and analyze data, so you can choose the right tool for each task. Whether you prefer SQL for familiar querying patterns or Spark for complex analysis, the lakehouse accommodates your workflow.

Query data using the SQL analytics endpoint

The SQL analytics endpoint provides read-only access to lakehouse tables using T-SQL queries. This endpoint is automatically created with every lakehouse. It allows you to query tables without affecting the underlying data files. You can write queries to explore data, filter rows, aggregate values, and join tables, just like you would with a traditional SQL database.

Common use cases for the SQL analytics endpoint include:

- **Ad-hoc queries:** Quickly investigate data to answer business questions.
- **Business intelligence connections:** Connect tools like Power BI, Excel, or Azure Data Studio to retrieve data for reports.
- **Data validation:** Verify transformation results after loading or processing data.

You can also create SQL views to store reusable query logic. Views are useful when you need to apply business rules, simplify complex joins, or provide curated data for downstream consumers. For example, you might create a view that joins sales and product tables and filters for active products only, making it easier for report authors to consume the data.

The SQL analytics endpoint also supports **row-level security** and **column-level security**, so you can control which users see which data when they query through SQL.

Tip

Copilot for SQL queries can help you write T-SQL queries from natural language descriptions. You can describe what you want to analyze, and Copilot suggests query code to accomplish your goal. This approach accelerates query authoring and helps you learn T-SQL patterns.

Query data using Spark notebooks

Notebooks provide a flexible, code-based environment for querying and analyzing lakehouse data. You can use Spark SQL for SQL-like queries or PySpark for programmatic data manipulation. Notebooks are valuable when you need to perform exploratory data analysis, apply statistical methods, or prepare data for machine learning models.

Spark SQL vs PySpark:

- **Spark SQL:** Use SQL syntax within a notebook cell to query lakehouse tables. Write queries like `SELECT * FROM schema.table` to retrieve and analyze data.
- **PySpark:** Use Python code to manipulate data. You can write SQL queries using `spark.sql()` or use the DataFrame API with Python methods like `df.select()` and `df.filter()`.

Choose the approach that fits your preference and the complexity of your analysis. Spark SQL works well for familiar SQL patterns. PySpark provides greater flexibility for complex transformations and integration with Python libraries.

Common use cases for Spark notebooks include:

- **Exploratory data analysis:** Investigate patterns, outliers, and relationships in data.
- **Complex transformations:** Apply business logic that's easier to express in code than SQL.
- **Cross-workspace queries:** Use the four-part namespace (`workspace.lakehouse.schema.table`) to join data across multiple lakehouses in a single query.

Tip

Copilot for notebooks can generate PySpark or Spark SQL code from natural language prompts and explain existing code. This AI-powered assistance accelerates development and helps you learn Spark syntax as you work.

Analyze and visualize with Power BI

Power BI is the business intelligence and reporting layer in Fabric. It serves as the consumption layer where business users access data through interactive reports and dashboards.

Power BI can connect to lakehouse data in two ways:

- **Query the SQL analytics endpoint:** Analysts can connect directly to the SQL analytics endpoint using Power BI or other tools like Excel to run ad-hoc queries and explore data before building reports.
- **Create a semantic model:** You can create a semantic model that references specific lakehouse tables. This model defines relationships, measures, and business logic that Power BI reports consume.

When you build reports on a lakehouse semantic model, Power BI uses **Direct Lake** mode by default. Direct Lake reads data directly from Delta Lake Parquet files without importing or copying data. This approach provides fast query performance while ensuring reports always reflect the current lakehouse data.

Semantic models also support downstream intelligent experiences. When you define clear relationships and business measures in a semantic model, Copilot in Power BI can generate visualizations and answer business questions by reasoning over your lakehouse data.

5. Exercise - Create a Microsoft Fabric lakehouse

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/5-exercise-lakehouse>

Exercise - Create a Microsoft Fabric lakehouse

Completed

A lakehouse in Microsoft Fabric uses OneLake, a highly scalable file store built on Azure Data Lake Storage Gen2. The lakehouse includes a metastore for relational objects like tables and views. These objects use the open-source Delta Lake table format. With Delta Lake, you define table schemas and query them using SQL.

In this exercise, you perform the following tasks:

- Create a lakehouse
- Upload files
- Load file data into tables
- Query lakehouse tables using SQL
- Create a visual query

Note

You need a Microsoft Fabric trial license with the Fabric preview enabled in your tenant. See [Getting started with Fabric](#) to enable your Fabric trial license.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

6. Module assessment

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/6-knowledge-check>

Module assessment

Completed

7. Summary

<https://learn.microsoft.com/en-us/training/modules/get-started-lakehouses/7-summary>

Summary

Completed

In this module, you learned how to create a lakehouse, bring data in through multiple ingestion methods, and query that data using SQL and Spark. You also explored how organizing lakehouse data with clear schemas and naming conventions creates a foundation that supports not only traditional reporting through Power BI, but also intelligent experiences powered by Fabric IQ and Copilot.

Without a lakehouse, organizations often maintain separate systems for flexible file storage and structured analytics, leading to data silos, duplication, and complex integration. A lakehouse in Microsoft Fabric eliminates that divide by combining both capabilities in a single platform, built on OneLake.

For more information, see the [Data Engineering in Microsoft Fabric](#) documentation.