





**Metastores**

A metastore is the top-level container of objects in Unity Catalog. It registers metadata about data and AI assets and the permissions that govern access to them. Azure Databricks account admins should create one metastore for each region in which they operate and assign them to Azure Databricks workspaces in the same region. For a workspace to use Unity Catalog, it must have a Unity Catalog metastore attached.

A metastore can optionally be configured with a managed storage location in an Azure Data Lake Storage Gen2 container in your Azure account. See Managed storage.

**Note**

This metastore is distinct from the Hive metastore included in Azure Databricks workspaces that have not been enabled for Unity Catalog. If your workspace includes a legacy Hive metastore, the data in that metastore will still be available alongside data defined in Unity Catalog, in a catalog named hive\_metastore. Note that the hive\_metastore catalog is not managed by Unity Catalog and does not benefit from the same feature set as catalogs defined in Unity Catalog.

See Create a Unity Catalog metastore.

**Catalogs**

A catalog is the first layer of Unity Catalog’s three-level namespace. It’s used to organize your data assets. Users can see all catalogs on which they have been assigned the USE CATALOG data permission.

Depending on how your workspace was created and enabled for Unity Catalog, your users may have default permissions on automatically provisioned catalogs, including either the main catalog or the *workspace catalog* (<workspace-name>). For more information, see Default user privileges.

See Create and manage catalogs.

**Schemas**

A schema (also called a database) is the second layer of Unity Catalog’s three-level namespace. A schema organizes tables and views. Users can see all schemas on which they have been assigned the USE SCHEMA permission, along with the USE CATALOG permission on the schema’s parent catalog. To access or list a table or view in a schema, users must also have SELECT permission on the table or view.

If your workspace was enabled for Unity Catalog manually, it includes a default schema named default in the main catalog that is accessible to all users in your workspace. If your workspace was enabled for Unity Catalog automatically and includes a <workspace-name> catalog, that catalog contains a schema named default that is accessible to all users in your workspace.

See Create and manage schemas (databases).

**Tables**

A table resides in the third layer of Unity Catalog’s three-level namespace. It contains rows of data. To create a table, users must have CREATE and USE SCHEMA permissions on the schema, and they must have the USE CATALOG permission on its parent catalog. To query a table, users must have the SELECT permission on the table, the USE SCHEMA permission on its parent schema, and the USE CATALOG permission on its parent catalog.

A table can be *managed* or *external*.

**Managed tables**

Managed tables are the default way to create tables in Unity Catalog. Unity Catalog manages the lifecycle and file layout for these tables. You should not use tools outside of Azure Databricks to manipulate files in these tables directly.

By default, managed tables are stored in the root storage location that you configure when you create a metastore. You can optionally specify managed table storage locations at the catalog or schema levels, overriding the root storage location. Managed tables always use the Delta table format.

When a managed table is dropped, its underlying data is deleted from your cloud tenant within 30 days.

See Managed tables.

**External tables**

External tables are tables whose data lifecycle and file layout are not managed by Unity Catalog. Use external tables to register large amounts of existing data in Unity Catalog, or if you require direct access to the data using tools outside of Azure Databricks clusters or Databricks SQL warehouses.

When you drop an external table, Unity Catalog does not delete the underlying data. You can manage privileges on external tables and use them in queries in the same way as managed tables.

External tables can use the following file formats:

* DELTA
* CSV
* JSON
* AVRO
* PARQUET
* ORC
* TEXT

See External tables.

**Views**

A view is a read-only object created from one or more tables and views in a metastore. It resides in the third layer of Unity Catalog’s three-level namespace. A view can be created from tables and other views in multiple schemas and catalogs. You can create dynamic views to enable row- and column-level permissions.

See Create a dynamic view.

**Volumes**

**Important**

This feature is in **Public Preview**.

A volume resides in the third layer of Unity Catalog’s three-level namespace. Volumes are siblings to tables, views, and other objects organized under a schema in Unity Catalog.

Volumes contain directories and files for data stored in any format. Volumes provide non-tabular access to data, meaning that files in volumes cannot be registered as tables.

* To create a volume, users must have CREATE VOLUME and USE SCHEMA permissions on the schema, and they must have the USE CATALOG permission on its parent catalog.
* To read files and directories stored inside a volume, users must have the READ VOLUME permission, the USE SCHEMA permission on its parent schema, and the USE CATALOG permission on its parent catalog.
* To add, remove, or modify files and directories stored inside a volume, users must have WRITE VOLUME permission, the USE SCHEMA permission on its parent schema, and the USE CATALOG permission on its parent catalog.

A volume can be *managed* or *external*.

**Note**

When you define a volume, you can no longer access any paths that overlap the volume location using external locations in the Catalog Explorer or cloud URIs.

**Managed volumes**

Managed volumes store files in the Unity Catalog default storage location for the schema in which they’re contained. Managed volumes are a convenient solution when you want to provision a governed location for working with files without the overhead of creating and managing external locations and storage credentials.

The following precedence governs which location is used for a managed volume:

* Schema location
* Catalog location
* Unity Catalog root storage location

When you delete a managed volume, the files stored in this volume are also deleted from your cloud tenant within 30 days.

See What is a managed volume?.

**External volumes**

An external volume is registered to a Unity Catalog external location and provides access to existing files in cloud storage without requiring data migration. Users must have the CREATE EXTERNAL VOLUME permission on the external location to create an external volume.

External volumes support scenarios where files are produced by other systems and staged for access from within Azure Databricks using object storage or where tools outside Azure Databricks require direct file access.

Unity Catalog does not manage the lifecycle and layout of the files in external volumes. When you drop an external volume, Unity Catalog does not delete the underlying data.

See What is an external volume?.

**Models**

A model resides in the third layer of Unity Catalog’s three-level namespace. In this context, “model” refers to a machine learning model that is registered in the MLflow Model Registry. To create a model in Unity Catalog, users must have the CREATE MODEL privilege for the catalog or schema. The user must also have the USE CATALOG privilege on the parent catalog and USE SCHEMA on the parent schema.

**Managed storage**

You can store managed tables and managed volumes at any of these levels in the Unity Catalog object hierarchy: metastore, catalog, or schema. Storage at lower levels in the hierarchy overrides storage defined at higher levels.

When an account admin creates a metastore manually, they have the option to assign a storage location in an Azure Data Lake Storage Gen2 container in your Azure account to use as metastore-level storage for managed tables and volumes. If a metastore-level managed storage location has been assigned, then managed storage locations at the catalog and schema levels are optional. That said, metastore-level storage is optional, and Databricks recommends assigning managed storage at the catalog level for logical data isolation. See Data governance and data isolation building blocks.

**Important**

If your workspace was enabled for Unity Catalog automatically, the Unity Catalog metastore was created without metastore-level managed storage. You can opt to add metastore-level storage, but Databricks recommends assigning managed storage at the catalog and schema levels. For help deciding whether you need metastore-level storage, see **(Optional) Create metastore-level storage** and **Data is physically separated in storage**.