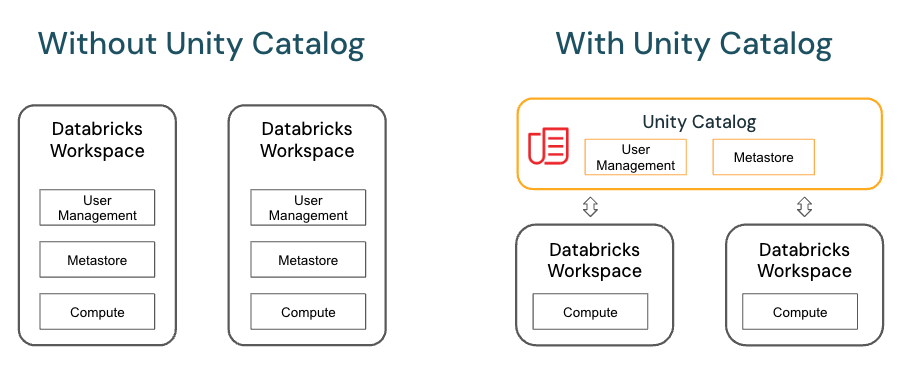
Databricks Unity Catalog

## What is Unity Catalog?

Unity Catalog provides centralized access control, auditing, Lineage and data discovery capabilities acrossDatabricks workspace.

## Before and After Unity Catalog

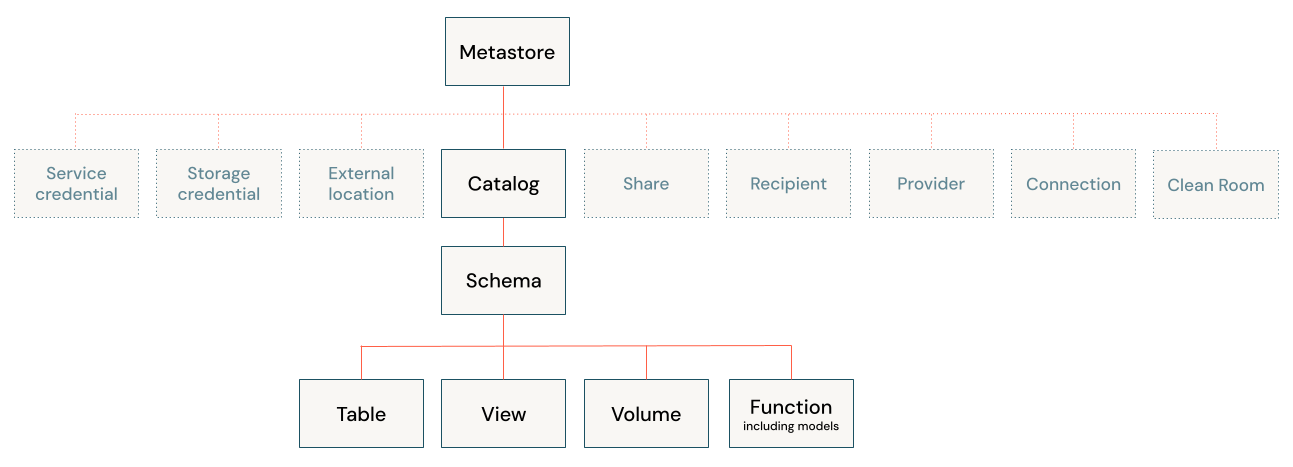


Key features of Unity Catalog include:

* **Define once, secure everywhere**: Unity Catalog offers a single place to administer data access policies that apply across all workspaces.
* **Standards-compliant security model**: Unity Catalog’s security model is based on standard ANSI SQL and allows administrators to grant permissions in their existing data lake using familiar syntax, at the level of catalogs, schemas (also called databases), tables, and views.
* **Built-in auditing and lineage**: Unity Catalog automatically captures user-level audit logs that record access to your data. Unity Catalog also captures lineage data that tracks how data assets are created and used across all languages.
* **Data discovery**: Unity Catalog lets you tag and document data assets, and provides a search interface to help data consumers find data.
* **System tables (Public Preview)**: Unity Catalog lets you easily access and query your account’s operational data, including audit logs, billable usage, and lineage.

## Unity Catalog Object Model:

In Unity Catalog, all metadata is registered in a metastore. The hierarchy of database objects in any Unity Catalog metastore is divided into three levels, represented as a three-level namespace (catalog.schema.table) when you reference tables, views, volumes, models, and functions.



### Metastores

The metastore is the top-level container for metadata in Unity Catalog. It registers metadata about data and AI assets and the permissions that govern access to them. For a workspace to use Unity Catalog, it must have a Unity Catalog metastore attached.

### Object hierarchy in the metastore

In a Unity Catalog metastore, the three-level database object hierarchy consists of catalogs that contain schemas, which in turn contain data and AI objects, like tables and models.

**Level one:**

* **Catalogs** are used to organize your data assets and are typically used as the top level in your data isolation scheme. Catalogs often mirror organizational units or software development lifecycle scopes
* **Non-data securable objects**, such as storage credentials and external locations, are used to manage your data governance model in Unity Catalog. These also live directly under the metastore.

**Level two:**

* **Schemas** (also known as databases) contain tables, views, volumes, AI models, and functions. Schemas organize data and AI assets into logical categories that are more granular than catalogs. Typically a schema represents a single use case, project, or team sandbox. See [What are schemas in Azure Databricks?](https://learn.microsoft.com/en-gb/azure/databricks/schemas/).

**Level three:**

* **Volumes** are logical volumes of unstructured, non-tabular data in cloud object storage. Volumes can be either *managed*, with Unity Catalog managing the full lifecycle and layout of the data in storage, or *external*, with Unity Catalog managing access to the data from within Azure Databricks, but not managing access to the data in cloud storage from other clients. See [What are Unity Catalog volumes?](https://learn.microsoft.com/en-gb/azure/databricks/volumes/) and [Managed versus external tables and volumes](https://learn.microsoft.com/en-gb/azure/databricks/data-governance/unity-catalog/#managed-vs-external).
* **Tables** are collections of data organized by rows and columns. Tables can be either *managed*, with Unity Catalog managing the full lifecycle of the table, or *external*, with Unity Catalog managing access to the data from within Azure Databricks, but not managing access to the data in cloud storage from other clients. See [What are tables and views?](https://learn.microsoft.com/en-gb/azure/databricks/tables/) and [Managed versus external tables and volumes](https://learn.microsoft.com/en-gb/azure/databricks/data-governance/unity-catalog/#managed-vs-external).
* **Views** are saved queries against one or more tables. See [What is a view?](https://learn.microsoft.com/en-gb/azure/databricks/views/).
* **Functions** are units of saved logic that return a scalar value or set of rows. See [User-defined functions (UDFs) in Unity Catalog](https://learn.microsoft.com/en-gb/azure/databricks/udf/unity-catalog).
* **Models** are AI models packaged with MLflow and registered in Unity Catalog as functions. See [Manage model lifecycle in Unity Catalog](https://learn.microsoft.com/en-gb/azure/databricks/machine-learning/manage-model-lifecycle/).

### Other securable objects

In addition to the database objects and AI assets that are contained in schemas, Unity Catalog also governs access to data using the following securable objects:

* **Service credentials**, which encapsulate a long-term cloud credential that provides access to an external service. See [Manage access to external cloud services using service credentials](https://learn.microsoft.com/en-gb/azure/databricks/connect/unity-catalog/cloud-services/service-credentials).
* **Storage credentials** : An Authentication and authorization mechanism for accessing data store on your cloud tenant(Cloud Storage).

**NOTE:** To know how to create them check the Storage Credential and External location section in the document.

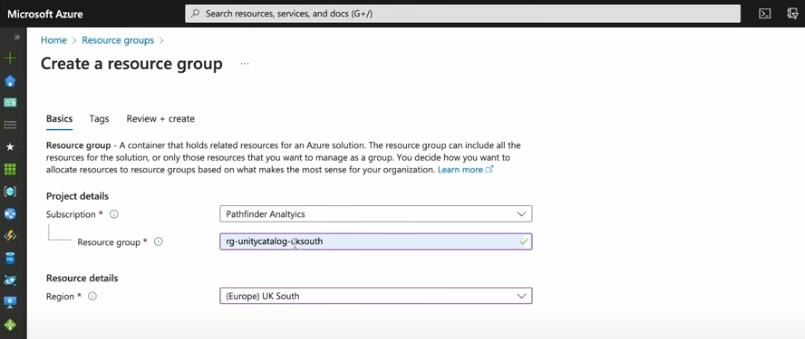
* **External locations**: An object that combines a cloud storage path with a storage credential that authorizes access to the cloud storage path.

**NOTE:** To know how to create them check the Storage Credential and External location section in the document.

* **Connections**, which represent credentials that give read-only access to an external database in a database system like MySQL using Lakehouse Federation. See [Lakehouse Federation and Unity Catalog](https://learn.microsoft.com/en-gb/azure/databricks/data-governance/unity-catalog/#federation) and [What is Lakehouse Federation?](https://learn.microsoft.com/en-gb/azure/databricks/query-federation/).
* **Clean rooms**, which represent a Databricks-managed environment where multiple participants can collaborate on projects without sharing underlying data with each other. See [What is Azure Databricks Clean Rooms?](https://learn.microsoft.com/en-gb/azure/databricks/clean-rooms/).
* **Shares**, which are Delta Sharing objects that represent a read-only collection of data and AI assets that a data provider shares with one or more recipients.
* **Recipients**, which are Delta Sharing objects that represent an entity that receives shares from a data provider.
* **Providers**, which are Delta Sharing objects that represent an entity that shares data with a recipient.

## Set up and manage Unity Catalog:

1. Create **resource group**



1. Create **Premium Tier Databricks Workspace**

A screenshot of a computer

Description automatically generated

1. Create **ADLS Gen 2 Storage Account**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

Create a Container inside Storage Account

A screenshot of a computer

Description automatically generated

1. Create **Access Connectors for Azure Databricks**

A screenshot of a computer

Description automatically generated

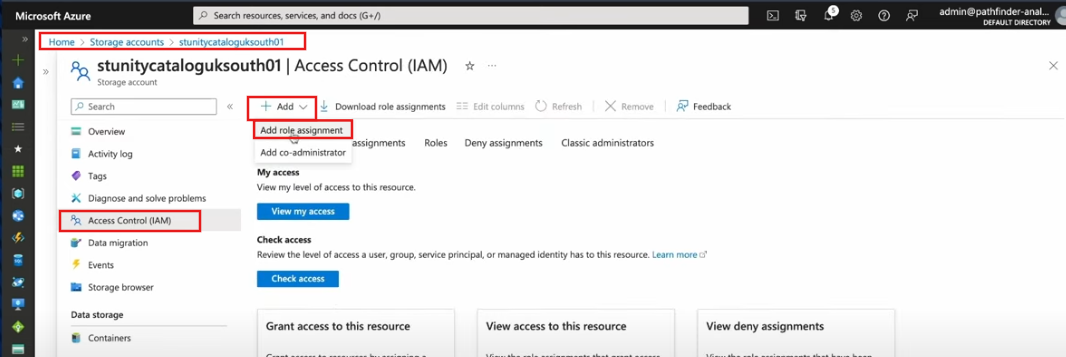
A screenshot of a computer

Description automatically generated

A screenshot of a computer

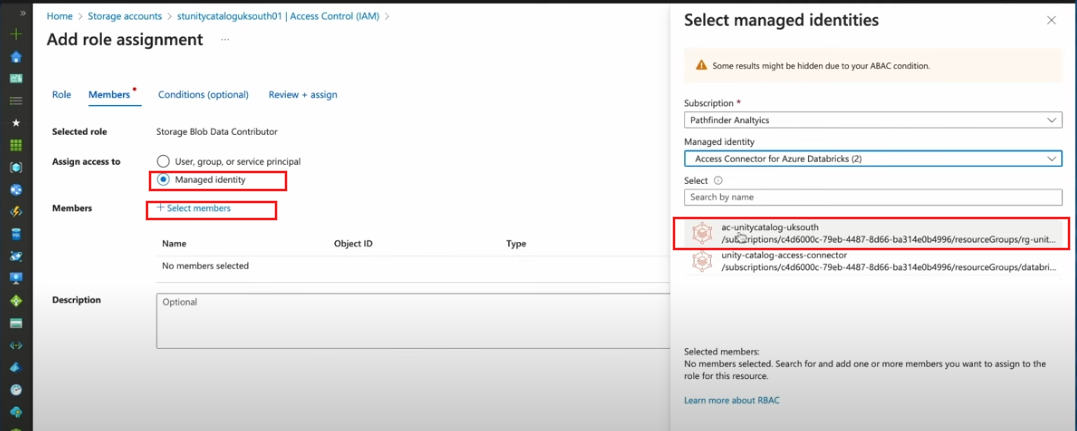
Description automatically generated

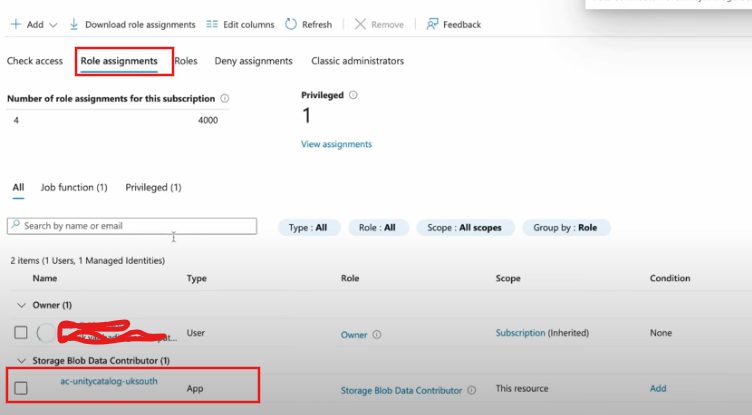
1. Grant **Storage Blob Data Contributor** roleto **Access Connectors for Azure Databricks** on **ADLS Gen2 Storage Account**



A screenshot of a computer

Description automatically generated



Check whether the role has been assigned or not by navigating into **Role assignments**  


1. Enable **Unity Catalog** by creating **Metastore** and assigning to **Workspace**

**NOTE:** Before Enabling the Unity Catalog make sure who have required roles such as Global administrator or Account Admin

Only Global administrator and Account Admin has access to **Manage Account** section

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Verify your role by navigating to User management by the help of Global Administrator/Account Admin

A screenshot of a computer

Description automatically generated

Lets enable Unity Catalog by creating a Metastore

**Note**:

**Region** should be same has Databricks workspace region

**Access Connector Id** is the Resource ID of Access Connector for Azure Databricks service

A screenshot of a computer

Description automatically generated

Once the Metastore has created **Enable Unity Catalog**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Storage Credentials and External Locations:

Navigate to catalog in the Databricks workspace

### Storage Credentials

A screenshot of a computer

Description automatically generated

**NOTE:** Access connector is the Resource ID available in Access Control for Azure Databricks Service

A screenshot of a computer

Description automatically generated

Once the Storage credential is created you grant and revoke permission on that storage

A screenshot of a computer

Description automatically generated

A screenshot of a computer

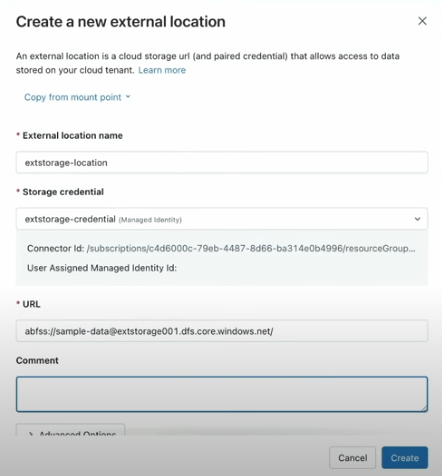
Description automatically generated

### External Locations

Navigate to Catalog and click on External Data

A screenshot of a computer

Description automatically generated



Once it is created test the connection just in case

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now open a notebook and run the code to check whether we can access the files in that storage.

A screenshot of a computer

Description automatically generated

## Create Catalog and Schema

### Create Catalog

Navigate to Catalog and click on Create Catalog

NOTE:

To create a Catalog

* 1. You must be an **Azure Databricks Metastore Admin** or have the **CREATE CATALOG** privilege on the metastore
  2. You must have a unity catalog metastore linked to the workspace where you perform the catalog creation
  3. The cluster that you use to run the notebook to create a catalog muist be Unity Catalog-complaint access mode.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

We are not adding any storage location so that the metadata will store in the default location

A screenshot of a computer

Description automatically generated

The catalog has been created

### Create Schema

**NOTE:** By default there will be two schemas available/created i.e default, information\_schema.

To create a Schema navigate to Catalog and select a catalog in which you want to create a Schema and click on CREATE SCHEMA.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Here I am not specifying location so that the metadata will be stored in the default location where the catalog has stored.

You can specify the storage location if required but make sure that the catalog also has the same storage location.

A screenshot of a computer

Description automatically generated

We can also Create/Delete Catalog and Schema by running commands in Notebooks

NOTE: By default when dropping a schema/catalog RESTRICT is selected

|  |
| --- |
| CREATE CATALOG [ IF NOT EXISTS ] <catalog\_name>  [ USING SHARE provider\_name.share\_name ]  [ MANAGED LOCATION ‘location\_path’ ]  [ COMMENT comment] |
| CREATE SCHEMA [ IF NOT EXISTS ] <schema\_name>  [ COMMENT schema\_comment ]  [ MANAGED LOCATION location\_path ]  [ WITH DBPROPERTIES ( { property\_name = property\_value } [, …..]) ] |
| DROP CATALOG [ IF EXISTS ] catalog\_name [ RESTRICT | CASCADE ] |
| DROP SCHEMA [ IF EXISTS ] schema\_name [ RESTRICT | CASCADE ] |

If you have any objects exists within the catalog or schema **RESTRICT** will prevent it from Deleting it.

So to drop a catalog or schema that contains objects within it then we need to specify **CASCADE**

uc-metastore-eastus@ **dbstoragebzhlf6xnk5pyi**

.dfs.core.windows.net/

**db-metastore-eastus**

uc-metastore-eastus@stunitycatalogeastus001.dfs.core.windows.net/

/subscriptions/2d00bd8e-67d1-43ae-9df2-cf9be20447e0/resourceGroups/databricks-rg-adb-eastus-001-ozivmw7rph34y/providers/Microsoft.Databricks/accessConnectors/unity-catalog-access-connector

select replicate('x',len('0123-0256-8954-5698') - 4) +

SUBSTRing('0123-0256-8954-5698', len('0123-0256-8954-5698') - 3, 4)

select replicate('x',len('0123-0256-8954-5698') - 4) + right('0123-0256-8954-5698',4) --dinesh

select replace('0123-0256-8954-5698', left('0123-0256-8954-5698',len('0123-0256-8954-5698')-4),replicate('\*',len('0123-0256-8954-5698')-4)) --kiran

SELECT REVERSE(STUFF(REVERSE('0123-0256-8954-5698'), 5, len('0123-0256-8954-5698'), REPLICATE('x',len('0123-0256-8954-5698')-4))) --neeraj

SELECT REPLACE(TRANSLATE('01Neeraj01', '0123456789', ' '), ' ', '')--Neeraj

SELECT 'KIRAN@GUNNA'