Unity Catalog – Mini Project Summary

1. Introduction

Data is everywhere and plays a critical role in daily life. With such large volumes of information, ensuring that data is secure, accurate, and well-organized is essential.
Data governance provides the rules, processes, and tools to achieve this by controlling who can access data, tracking where it comes from, and ensuring it is protected.

Databricks Unity Catalog is a unified governance layer for data and AI within the Databricks platform. It allows organizations to centrally manage all assets — structured/unstructured data, machine learning models, notebooks, dashboards, and files — across any cloud.

2. Purpose

- Centralize governance for all data assets.
- Provide fine-grained access control.
- Enable auditing and data lineage.
- Simplify secure data sharing.

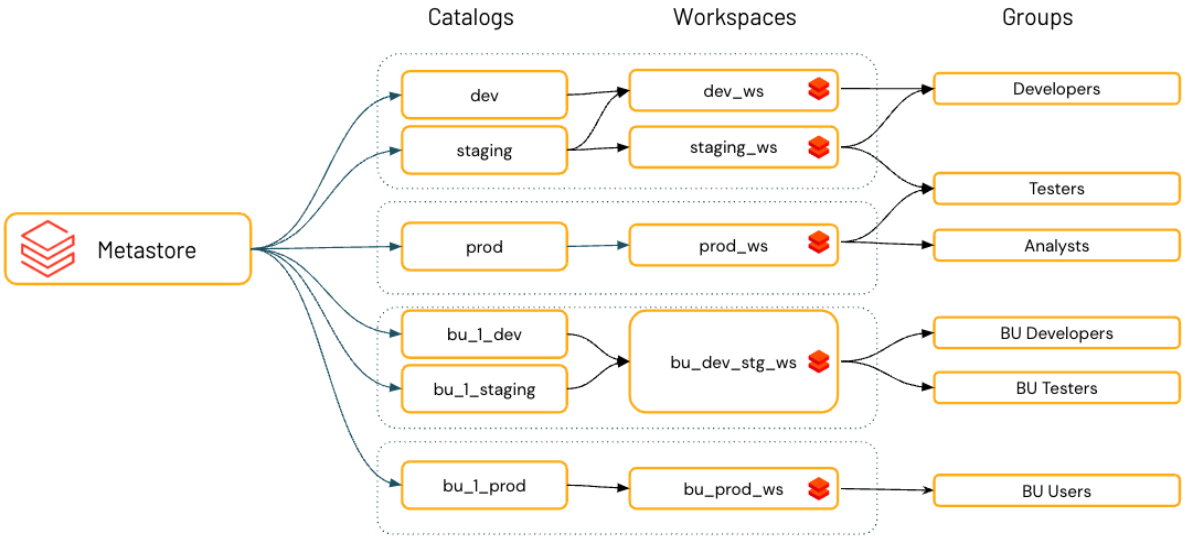
3. Architecture Overview

Unity Catalog uses a three-level namespace:

Metastore → Catalog → Schema → Tables / Views / Models

- Metastore – Top-level container.
- Catalog – Logical grouping of schemas.
- Schema – Collection of tables/views.
- Table/View – Actual data objects.

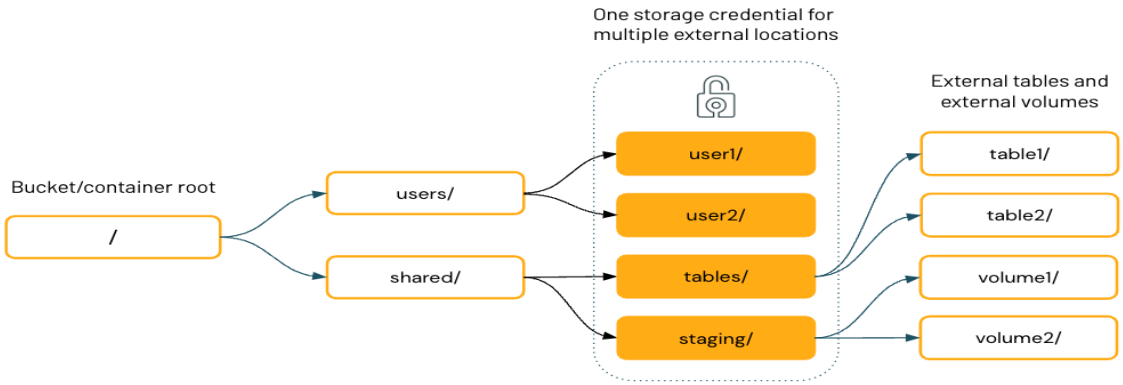
Architecture Diagram



4. Key Features

- **Centralized Policy Management** – One place to manage access for all workspaces.
- **ANSI SQL-Based Security** – Use familiar SQL syntax for granting/revoking permissions.
- **Built-in Auditing & Lineage** – Track data usage and dependencies.
- **Data Discovery** – Tag and search for data.
- **Managed & External Storage** – Store data in Databricks-managed or external sources.
- **Delta Sharing** – Securely share data across clouds/platforms.

Governance Overview

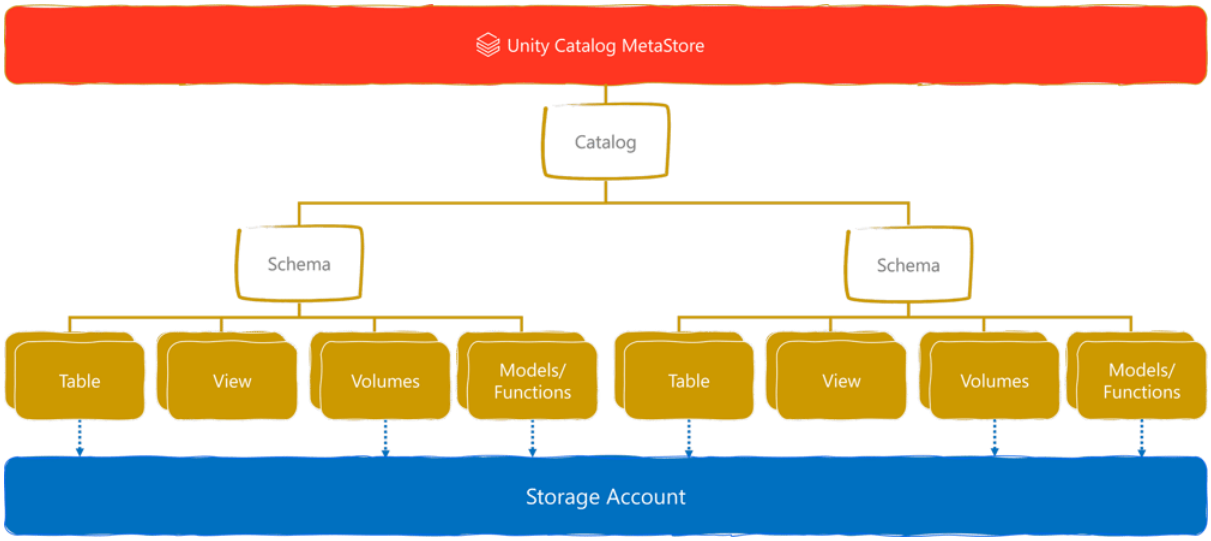


5. Access Control

Unity Catalog uses:

- **Privilege Inheritance** – Permissions set at a higher level flow down to lower levels.
- **Object Ownership Model** – The creator of an object is its owner by default.
- **Dynamic Views** – For row/column-level security.
- **External Locations & Storage Credentials** – To securely access data in external sources.

Catalog Explorer & Permissions

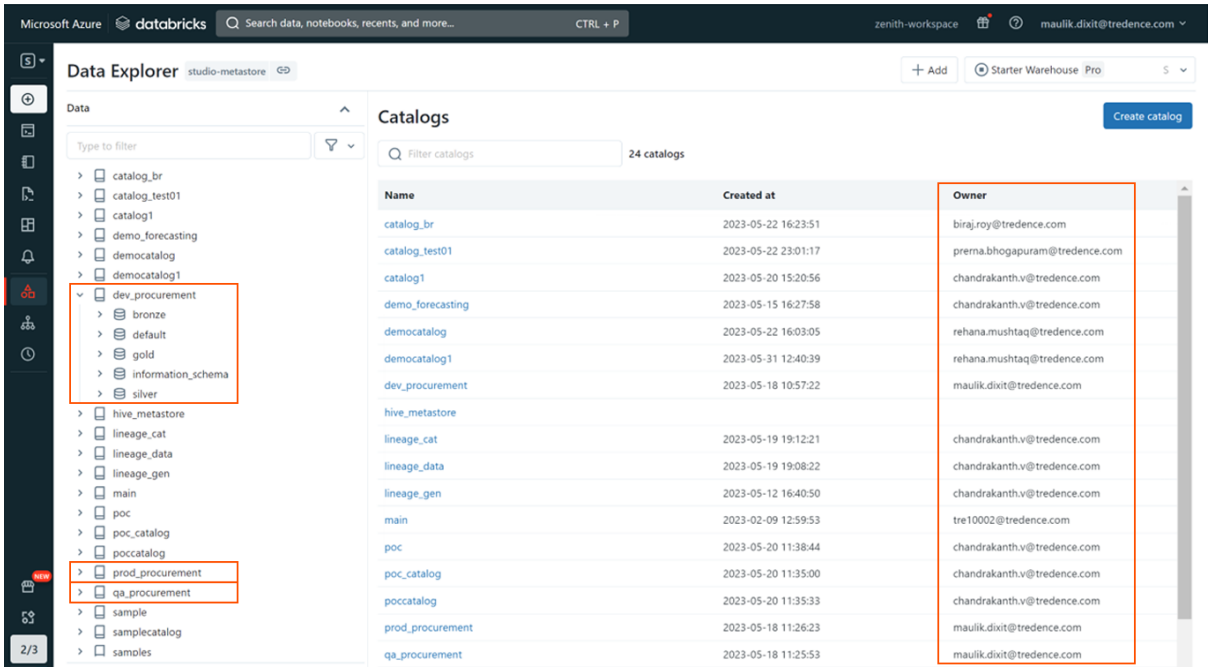


6. Setup Steps

Step 1 – Create Metastore

1. Navigate to Data → Unity Catalog → Metastores in Databricks.
2. Click Create Metastore, enter name, region, and storage location.
3. Assign to workspace.

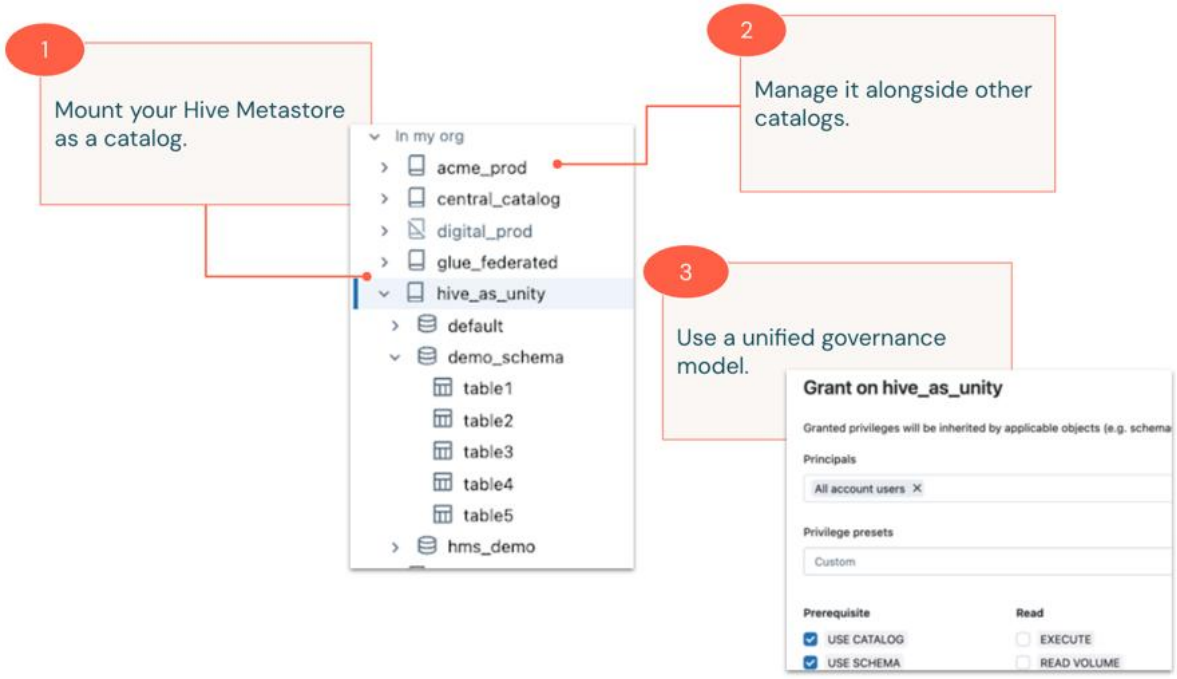
Metastore Creation UI



Step 2 – Configure Storage Credentials

- Create a service principal with storage access.
- Add credentials in Unity Catalog.

Storage Credential Config



Step 3 – Create Catalog & Schema

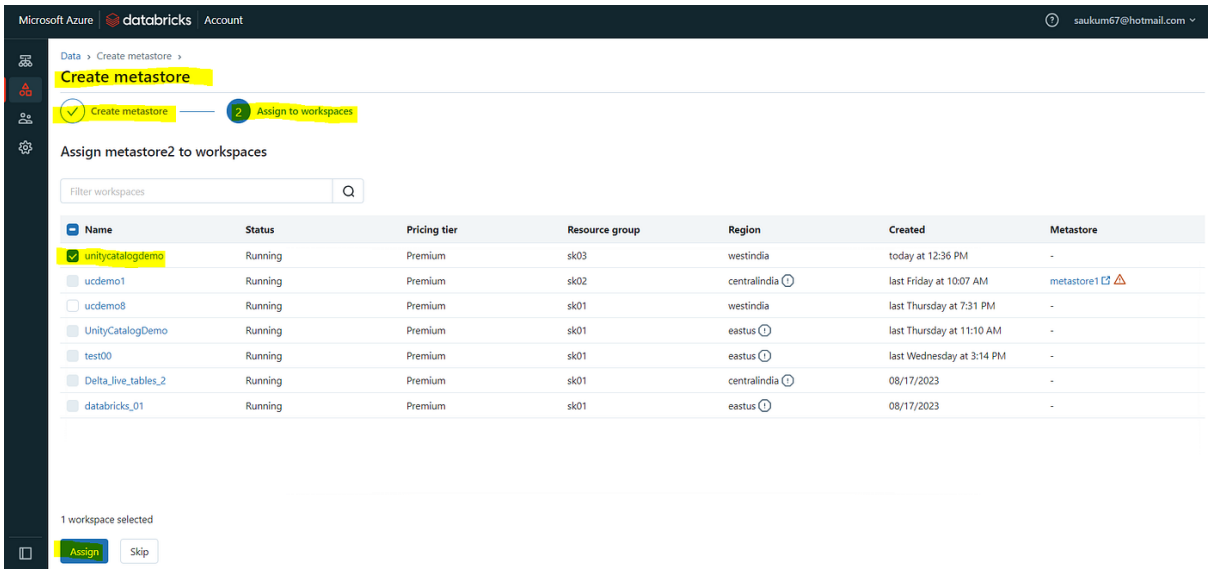
CREATE CATALOG my_catalog;

USE CATALOG my_catalog;

CREATE SCHEMA my_schema;

USE SCHEMA my_schema;

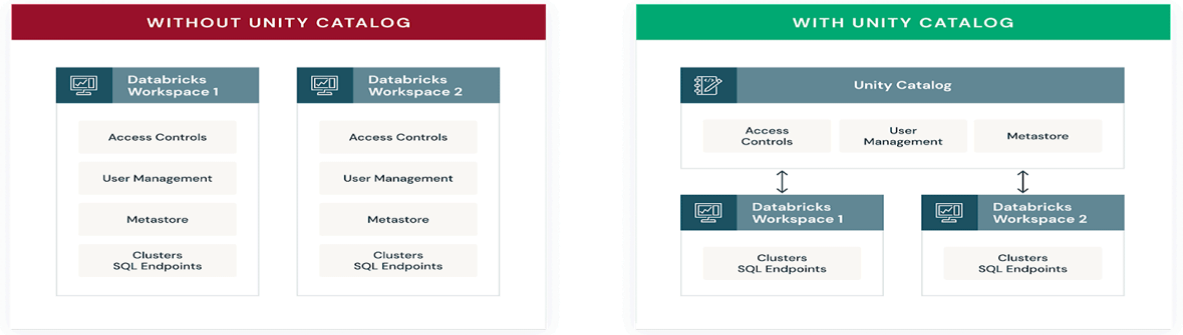
Catalog & Schema Creation UI



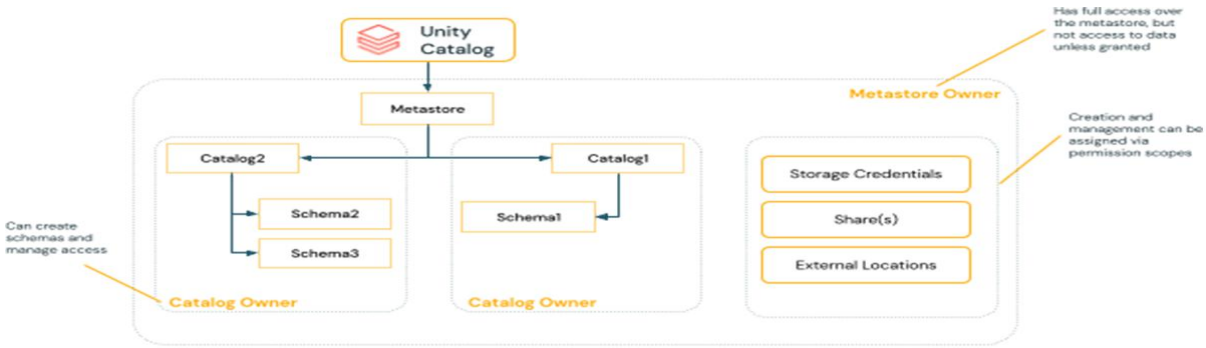
7. Best Practices

1. Use catalogs for isolation (prod/dev, sensitive/non-sensitive).
2. Group-based ownership instead of individual ownership.
3. Enable full audit logging.
4. Implement least privilege access.
5. Use Delta Sharing for external collaboration.

Delta Sharing Interface



Data Lineage View



8. Limitations

- Requires Databricks Runtime 11.3 LTS+.
- No bucketing or custom partition schemes.
- Limited R workload security.
- Naming restrictions on objects & columns.
- Workspace-level groups not supported in GRANT statements.

9. Conclusion

Unity Catalog is a centralized governance solution that unifies data security, discovery, auditing, and sharing across the Databricks platform. By following best practices and understanding its limitations, organizations can ensure secure, compliant, and efficient data management.