**Mini Project: Unity Catalog in Azure Databricks**

## 1. Objective

The goal of this mini project is to **explore and implement Unity Catalog** in Azure Databricks, and demonstrate its key **data governance capabilities**:

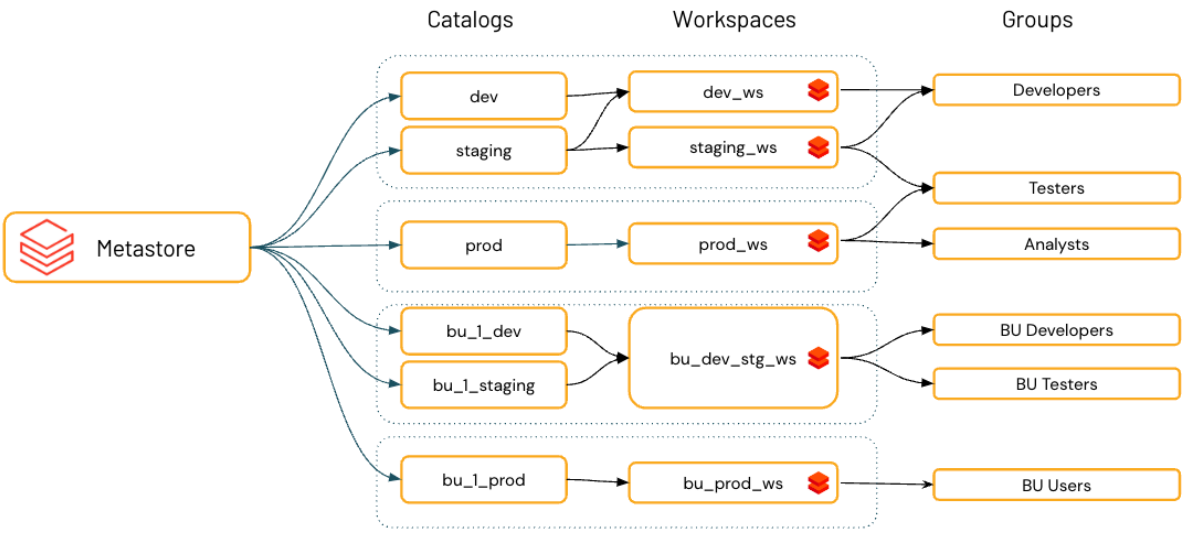
- Data Discovery
- Data Audit
- Data Lineage
- Data Access Control

---

## 2. Introduction to Unity Catalog

Unity Catalog is a **centralized governance solution** in Databricks.
It helps organizations manage **access, permissions, security, and lineage** for all data and AI assets across different workspaces and clouds.

*Think of Unity Catalog as the "librarian" of Databricks — it keeps data organized, controls who can access it, and records how it is used.*




---

## 3. Project Setup

### 3.1 Environment

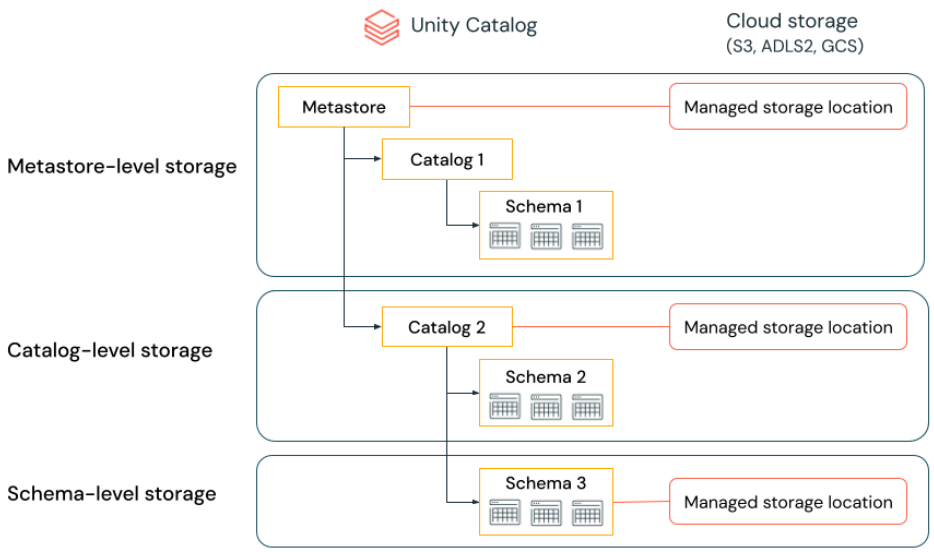
- **Platform**: Azure Databricks
- **Storage**: Azure Data Lake Storage Gen2 (ADLS)
- **Feature used**: Unity Catalog

### 3.2 Unity Catalog Structure

Unity Catalog organizes data using a **3-level namespace**:

`catalog.schema.table`

- **Catalog**: Top-level container (e.g., `sales_catalog`).
- **Schema**: Logical grouping of tables/views (e.g., `retail_schema`).
- **Table**: Actual dataset (e.g., `orders`).




---

## 4. Implementation Steps

### Step 1 — Enable Unity Catalog

- Unity Catalog was enabled in the Databricks workspace through the admin console.
- Metastore was configured and linked to the workspace.

**Step 2 — Create Catalogs, Schemas, and Tables**

```
-- Create a new catalog
CREATE CATALOG sales_catalog;

-- Create schema inside catalog
CREATE SCHEMA sales_catalog.retail_schema;

-- Create a managed table
CREATE TABLE sales_catalog.retail_schema.orders (
    order_id INT,
    customer_id INT,
    product STRING,
    amount DOUBLE,
    order_date DATE
);
```

**Step 3 — Data Access Control**

- Granted and revoked permissions using SQL commands.

```
-- Grant access to analysts
GRANT SELECT ON TABLE sales_catalog.retail_schema.orders TO
`analyst_group`;

-- Revoke access
REVOKE SELECT ON TABLE sales_catalog.retail_schema.orders FROM
`interns`;
```

**Step 4 — Data Discovery**

- Added comments and tags to tables and columns for better searchability.

```
COMMENT ON TABLE sales_catalog.retail_schema.orders IS 'Order
details table';
```

```
COMMENT ON COLUMN sales_catalog.retail_schema.orders.amount IS
'Purchase amount in USD';
```

**Step 5 — Data Audit**

- Access logs captured automatically, showing **who queried what data and when**.
- Useful for compliance and security monitoring.

**Step 6 — Data Lineage**

- Unity Catalog automatically tracked lineage:
    - Where data originated from
    - What transformations were applied
    - Which tables, notebooks, or dashboards used it

---

**5. Outcomes**

- Implemented a **structured governance model** for data in Databricks.
- Controlled **who can access which tables** through role-based access.
- Enabled **data discovery** with comments and tagging.
- Verified **audit logs** for monitoring compliance.
- Observed **data lineage** to track transformations and usage.

---

**6. Conclusion**

This mini project showed how Unity Catalog brings together **security, discoverability, auditing, and lineage** under a single framework in Databricks.
It ensures data is:

- Securely managed
- Easily discoverable
- Properly audited
- Governed consistently across multiple teams and workspaces

Unity Catalog makes Databricks a **complete data governance platform** for enterprises.

---