**1. Introduction to Unity Catalog**

**Unity Catalog**  
Unity Catalog is a centralized governance and metadata layer for managing data and AI assets across Databricks workspaces. It offers access control, auditing, lineage tracking, data discovery, and more in one place.

**Why it matters:**

* You define security rules just once and they apply everywhere.
* Built-in audit logs and data lineage help with compliance and understanding data movements.
* It’s easier for teams to find and use data, improving collaboration.

**2. Unity Catalog Metastore & Workspace Setup**

**What is a Metastore?**  
A metastore is the top-level container that holds metadata about tables, views, volumes, and permissions. It's where Unity Catalog stores its information. Each region needs its own metastore, and every workspace must be attached to one to use Unity Catalog.

**Steps to Create & Enable:**

|  |  |
| --- | --- |
| **Step** | **Action** |
| 1 | (Optional) Create a cloud storage location—like an Azure ADLS container or AWS S3 bucket—for storing managed data. |
| 2 | Create a managed identity or service principal (e.g., Azure managed identity or AWS IAM role) that has access to the storage. |
| 3 | In Databricks account console: create the metastore, attach storage, then link workspaces. |
| 4 | Assign Metastore Admin role to a group for better management and security. |

**Optional Storage Hierarchy Diagram:**  
Insert the first image (storage hierarchy) from above here to visually show how storage works across metastore, catalog, and schema levels.

**3. Unity Catalog Object Model & 3-Level Namespace**

**Three-Level Namespace Structure:**  
Unity Catalog organizes data in the catalog.schema.table format—like drawers (catalogs), folders (schemas), and documents (tables/views).

**Hierarchy Details:**

* **Metastore** → Top-level container for metadata.
* **Catalog** → Groups schemas (e.g., by domain or team).
* **Schema (or Database)** → Contains tables, views, and volumes.
* **Tables/Views/Volumes** → Actual data assets. Tables can be managed or external; views can enforce fine-grained row/column security.

**Visual Aid:**  
Insert the object model or namespace image from the carousel here to show the hierarchy clearly.

**4. Creating Catalogs, Schemas & Workspace Binding**

**Creating Catalogs & Schemas:**  
Use simple SQL commands in Databricks:

**CREATE CATALOG IF NOT EXISTS my\_catalog;**

**CREATE SCHEMA IF NOT EXISTS my\_catalog.my\_schema COMMENT 'Description';**

**You need appropriate permissions (metastore admin or CREATE CATALOG) to run these.**

**Workspace–Catalog Binding:**Admins can control which catalogs are visible in which workspaces. This ensures data isolation—e.g., only production workspaces see production catalogs.

**Summary Table for Commands and Permissions:**

|  |  |
| --- | --- |
| **Task** | **Permission Required** |
| Create Metastore | Account Admin |
| Attach Workspace | Account Admin |
| Create Catalog | Metastore Admin |
| Create Schema | Catalog Admin or higher |
| Grant Access | Workspace/Catalog Admin |