Managing your data efficiently using Databricks Unity Catalog

Wilson Mok

Jan 29, 2025

msdevmtl

Bio

Wilson Mok

Sr. Data architect & Consultant

/wilson-mok

/wilson-mok

@the-analytics-lab

My experiences includes:
- Avanade
- CAE
- Air Canada

Expected Audience and Presentation Structure

- This session, we will cover:
  - What is Unity Catalog and Why it is important
  - Key concepts of Unity Catalog
  - Designing our Unity Catalog
  - Demo: Implementing Unity Catalog and apply the access control.

- Fundamental knowledge: Databricks and SQL would be beneficial.

Before Unity Catalog, multiple workspaces create Fragmented Governance for organizations to manage…

Create Data Silos

- External tables are only accessible within the workspace

- Any Data created in the Hive tables are not accessible to other workspaces

- Cross-functional collaboration becomes cumbersome

No Traceability

- Metadata is stored within the workspace. Leading to inconsistencies.

- Lack ability to track data lineage across workspaces to maintain single source of truth.

- Organizations struggled to audit data usage and ensure compliance.

Access Control Complexity

- Access control is applied at the workspace level.

- Administrators have to manually configure each workspace and apply policy. Leads to delays.

- To simplify processes, teams often over-provision access, increasing security risks and data compliance challenges.

Fragmented Governance

Unity Catalog centralizes access management and data connectivity

1 Define once, secure everywhere

2 Standards-compliant security model

3 Built-in auditing and lineage
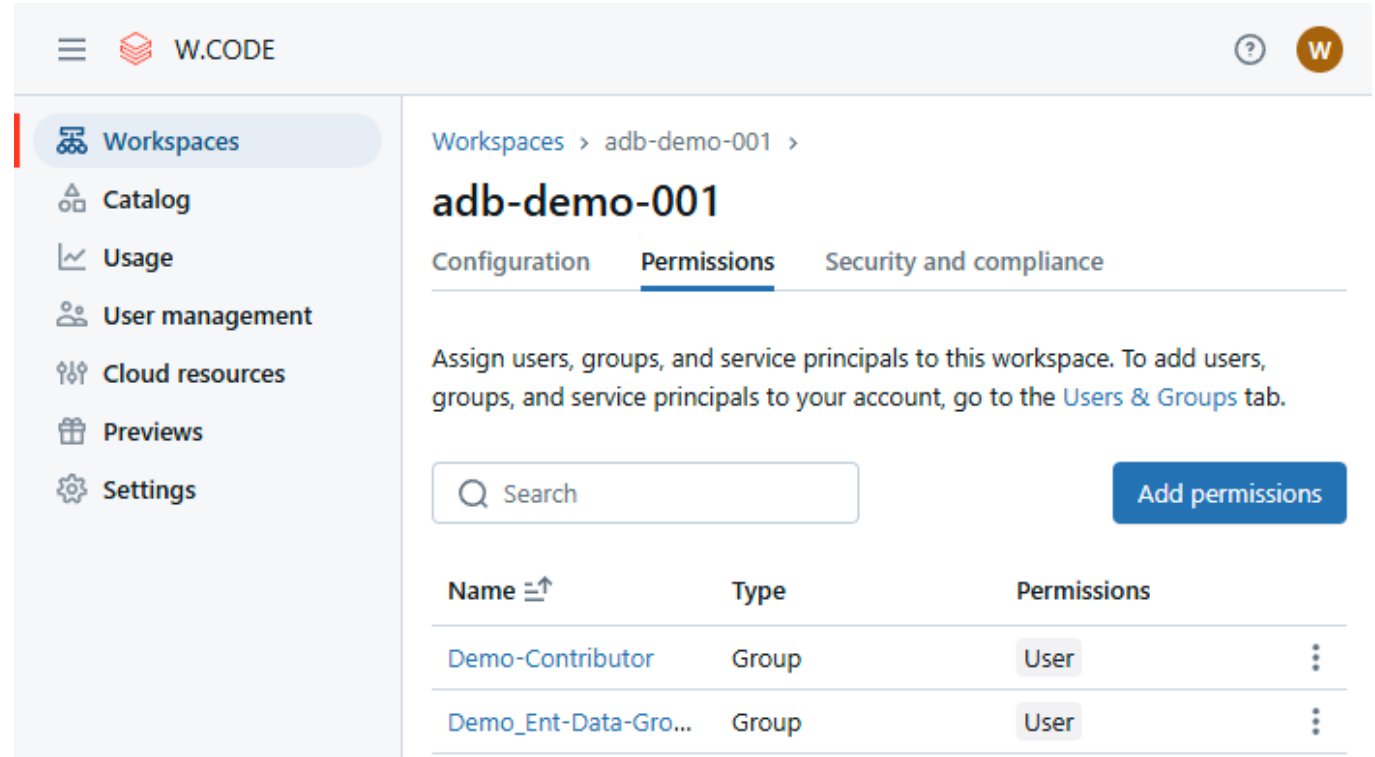
4 Data Discovery
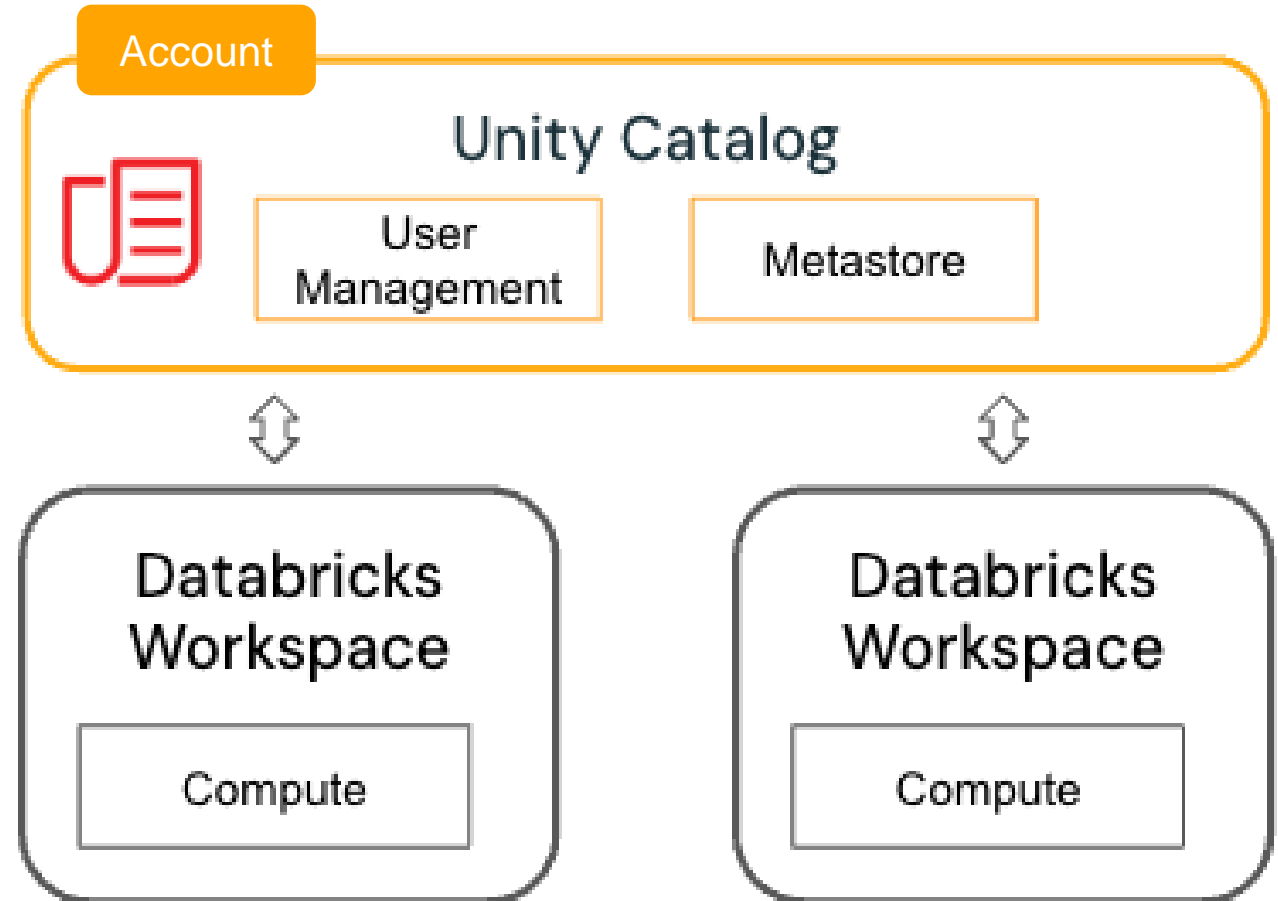
Unity Catalog centralizes access management and data connectivity

1   Define once, secure everywhere

- Centralized User Management by provisioning access to the Databricks Account first.
- Then assign the identity to the applicable workspace.

<u>Best Practice</u>: Configure SCIM to sync identity with Databricks Account.

Unity Catalog centralizes access management and data connectivity

1 Define once, secure everywhere

2 Standards-compliant security model

- Standard based ANSI SQL security model
- Fine-Grained access control: Grant/Revoke access at various level (Catalog, Schemas, Tables/Views)
- Access permissions are stored in the Metastore (region specific).

Best Practice: Grant access to Groups, instead of individual users. Least Privilege.

Grant on test.testschema.testsales ✕

Principals

Demo_Ent-Data-Group_Prod ✕ Demo_BU-Sales-Group ✕

Privileges

☐ APPLY TAG gives ability to apply tags to an object

☐ MODIFY gives ability to add, delete, and modify data to or from an object

☑ SELECT gives read access to an object

☐ ALL PRIVILEGES gives all privileges ⓘ

☐ MANAGE gives ownership-like ability for the object, such as managing permissions, dropping, or renaming

⚠ Additional privileges required for access
In order to perform actions on this table, the selected principal(s) must be granted USE CATALOG and USE SCHEMA on its parent resources. Learn more

☑ Also grant USE CATALOG on **test** and USE SCHEMA on **test.testschema**

Cancel   Grant

Unity Catalog centralizes access management and data connectivity
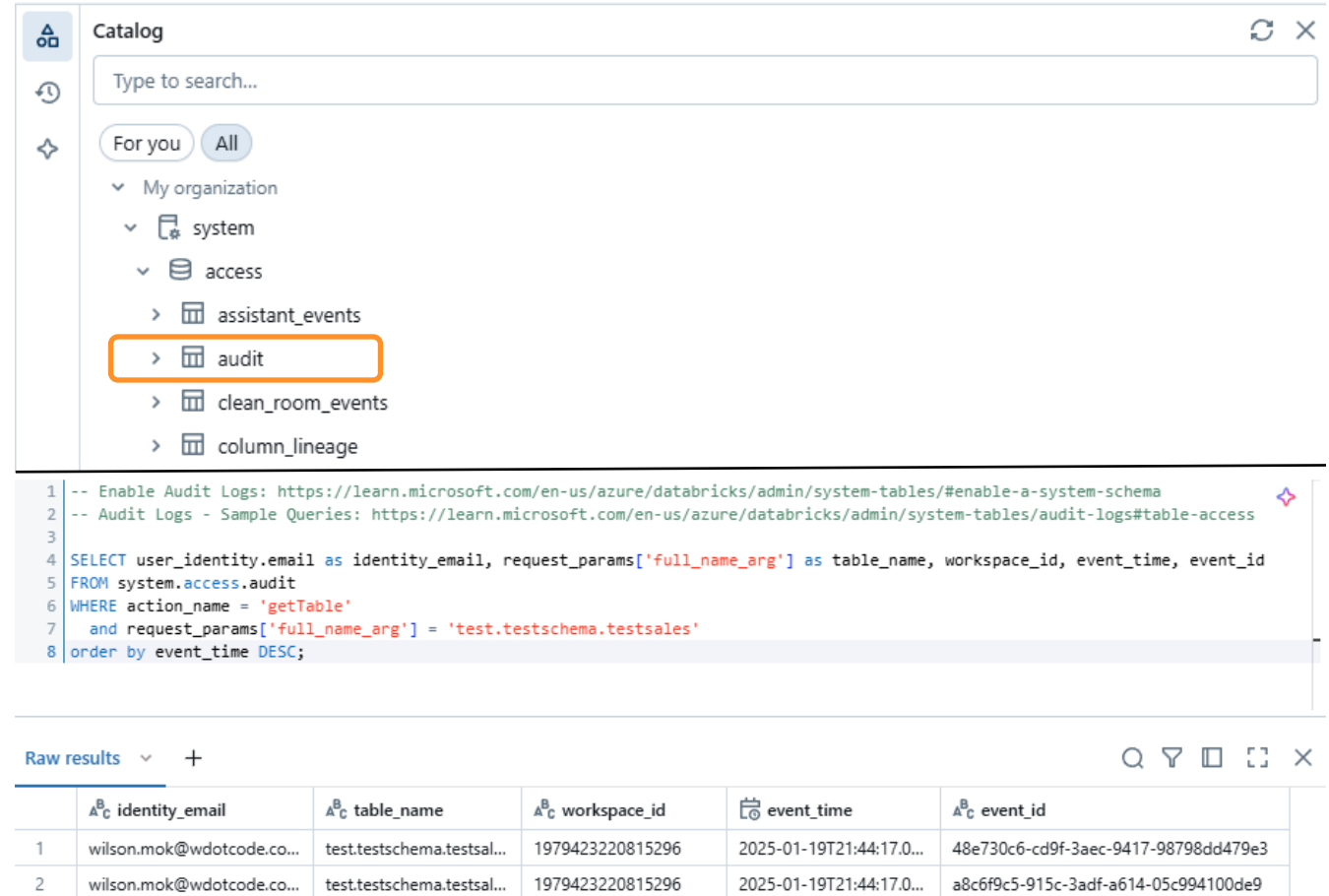
1 Define once, secure everywhere

2 Standards-compliant security model

3 Built-in auditing and lineage   `Public Preview`

- Built-in system tables to provide operational data for audit logs, billable usage, lineage, etc.
- This data is stored in the Metastore (Region specific) as well.

Best Practice: Create reports and alerts based on access data to ensure compliance and privacy.


```
1  -- Enable Audit Logs: https://learn.microsoft.com/en-us/azure/databricks/admin/system-tables/#enable-a-system-schema
2  -- Audit Logs - Sample Queries: https://learn.microsoft.com/en-us/azure/databricks/admin/system-tables/audit-logs#table-access
3
4  SELECT user_identity.email as identity_email, request_params['full_name_arg'] as table_name, workspace_id, event_time, event_id
5  FROM system.access.audit
6  WHERE action_name = 'getTable'
7    and request_params['full_name_arg'] = 'test.testschema.testsales'
8  order by event_time DESC;
```
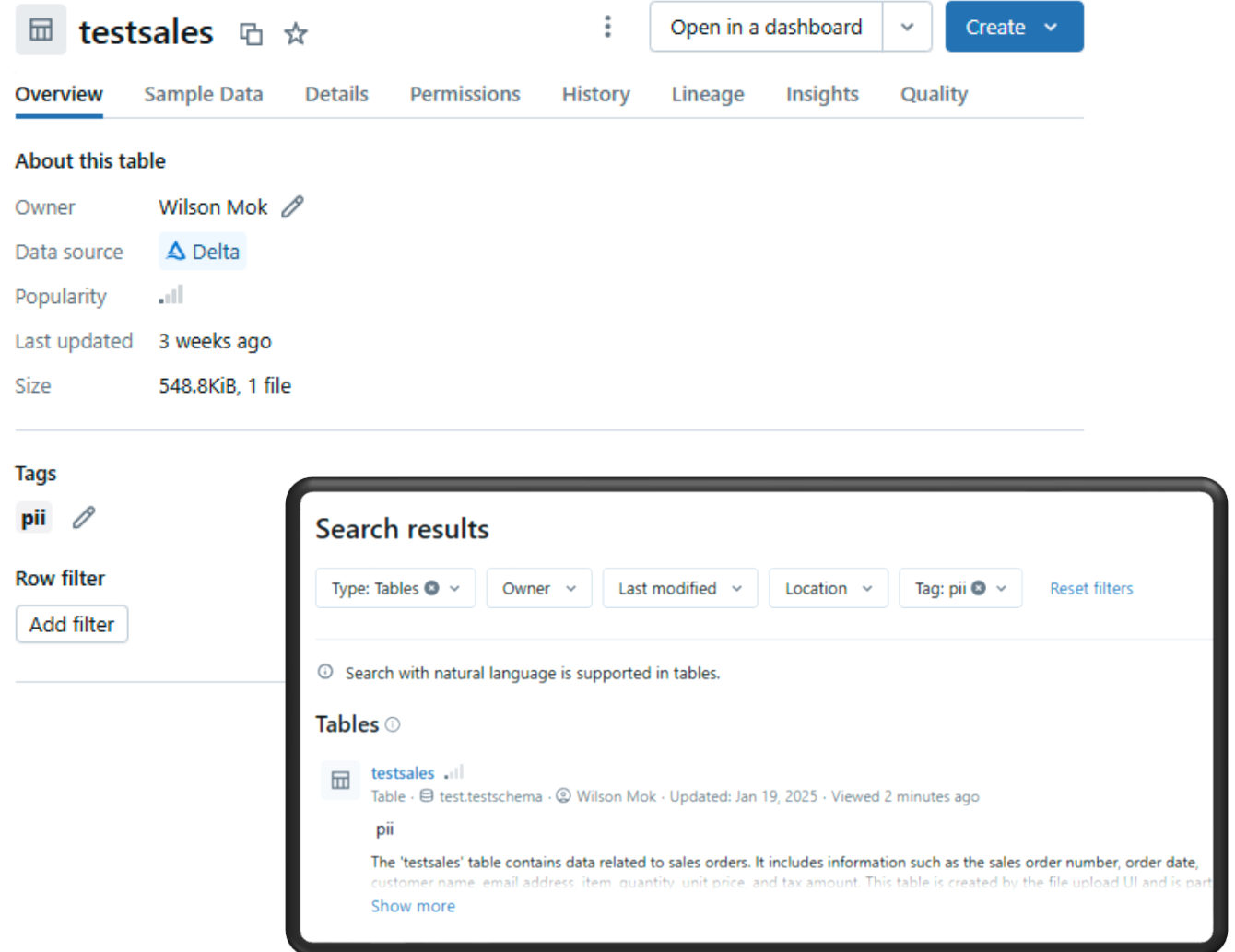
	identity_email	table_name	workspace_id	event_time	event_id
1	wilson.mok@wdotcode.co...	test.testschema.testsal...	1979423220815296	2025-01-19T21:44:17.0...	48e730c6-cd9f-3aec-9417-98798dd479e3
2	wilson.mok@wdotcode.co...	test.testschema.testsal...	1979423220815296	2025-01-19T21:44:17.0...	a8c6f9c5-915c-3adf-a614-05c994100de9

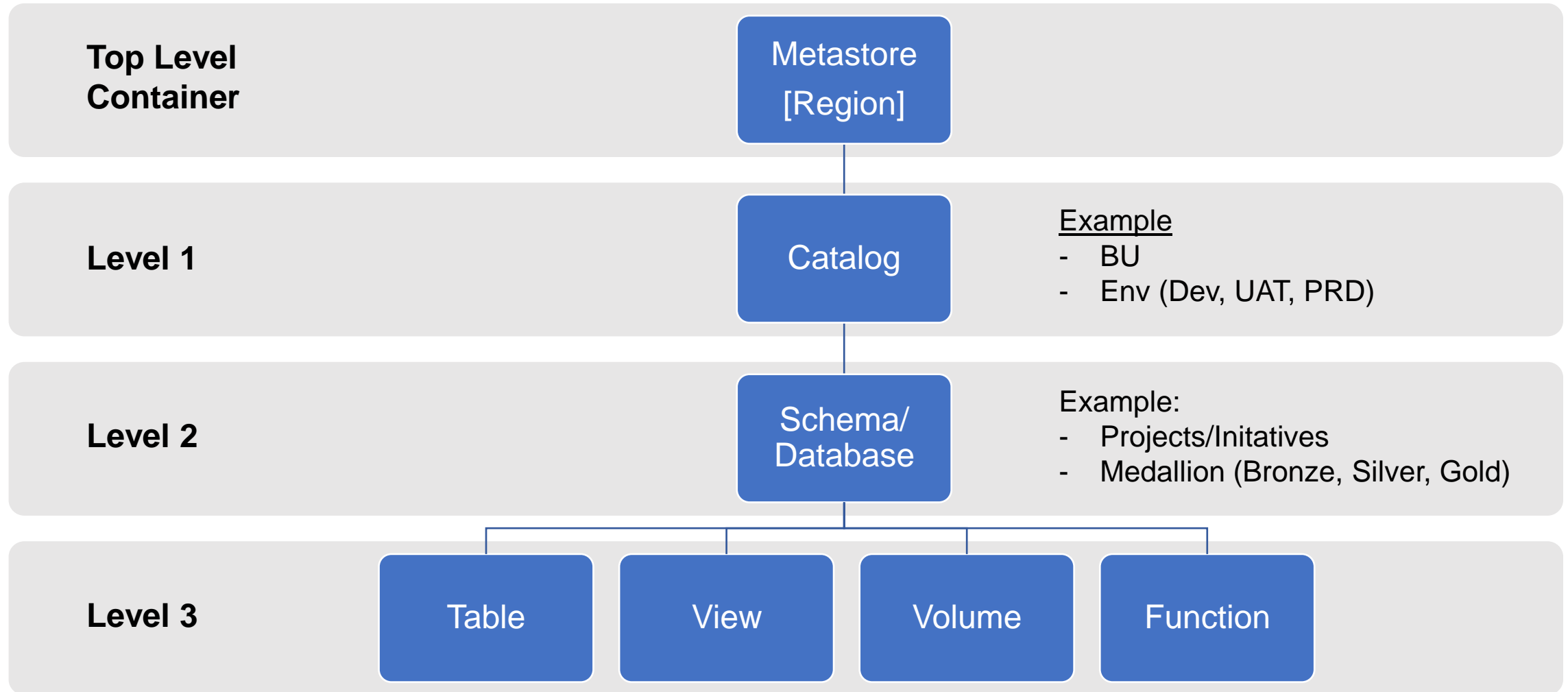# Unity Catalog centralizes access management and data connectivity

1 Define once, secure everywhere

2 Standards-compliant security model

3 Built-in auditing and lineage

4 Data Discovery

- Tag and document data assets
- Using Workspace Search to find tables with tags.

<u>Best Practice</u>: Add tags for data sensitivity (PII, GDPR, etc)

Credit: <u>What is Unity Catalog? - Azure Databricks | Microsoft Learn</u>

Key Concepts of Unity Catalog's Object Model

Top Level Container	Metastore [Region]	
Level 1	Catalog	Example - BU - Env (Dev, UAT, PRD)
Level 2	Schema/ Database	Example: - Projects/Initatives - Medallion (Bronze, Silver, Gold)
Level 3	Table View Volume Function	

Important to understand the difference between external vs managed objects

	Managed	External
Data Location	ADLS Gen 2 – One per Region	ADLS Gen 2 – Multiple
Ownership	Managed by Databricks	Managed by you
Data Access	Fully governed by Databricks	Accessible via cloud tools (Azure Storage Explorer) and Databricks
Security model	Fully controlled by Unity Catalog's access policies	Manage access policies in Unity Catalog and Cloud storage.
Metastore	Metadata & data are stored in the Metastore.	Metadata stores in Metastore, but data remains in external storage.
Supported Formats	Table: Delta Volume: Any files	Table: Delta, Parquet, CSV, JSON, etc Volume: Any files
Performance	Table: Auto optimize and compaction	Table: Manually optimized
Dropping a Table	Deletes both data & metadata	Deletes metadata. Data persists in storage.
Backup & Recovery	Table: 7 days retention period (UNDROP Table) Note: Parent schema & catalog must exist	Managed by cloud provider
Best for	Simplicity – Databricks-native workflows.	Flexibility – BYOD

Common Privileges used in Unity Catalog…

Privilege	Description	Applies To
USE CATALOG, SCHEMA	Grant access to see the objects within the Catalog or Schema. User cannot read or modify the data objects.	Catalog, Schema
SELECT	Read access to the table or view.	Table, View
MODIFY	Allow to add, update, delete data to or from a table	Table
CREATE TABLE, VIEW, VOLUME, FUNCTION	Allow to create objects.	Table, View, Volume, Function
READ FILES	Provide read access to files in the external location.	External Volume, Table (checkpoint)
WRITE FILES	Allow create, modify or delete files in the external location.	External Volume, Table (checkpoint)
All PRIVILEGES	Allows full access to the objects. Does not allow to permission management.	All
MANAGE	**Admin access** - Allows full access to the objects, permission management, transfer ownership.	All

Q & A 1

Design a Unity Catalog for our Organization

Enterprise Data Team

Goal: Ingest data, transform data for reporting to support business operations.

Requirements:
- Dedicated workspace for each environment (Dev, Uat, Prd).
- Ingest raw data (CSV, Parquet, JSON)
- Ensure data quality, lineage and security
- Adopted Medallion architecture using Delta Lakehouse.

Business Teams

Goal: Explore data & AI use cases and create business cases for funding.

Requirements:
- Secure sandbox environment.
- Ability to controlled access to specific datasets.
- Run experiments and POCs without impacting production workloads.
- Ability to access Enterprise's Uat and Prd data to support analysis.

Design a Unity Catalog for our Organization: Enterprise Data Team
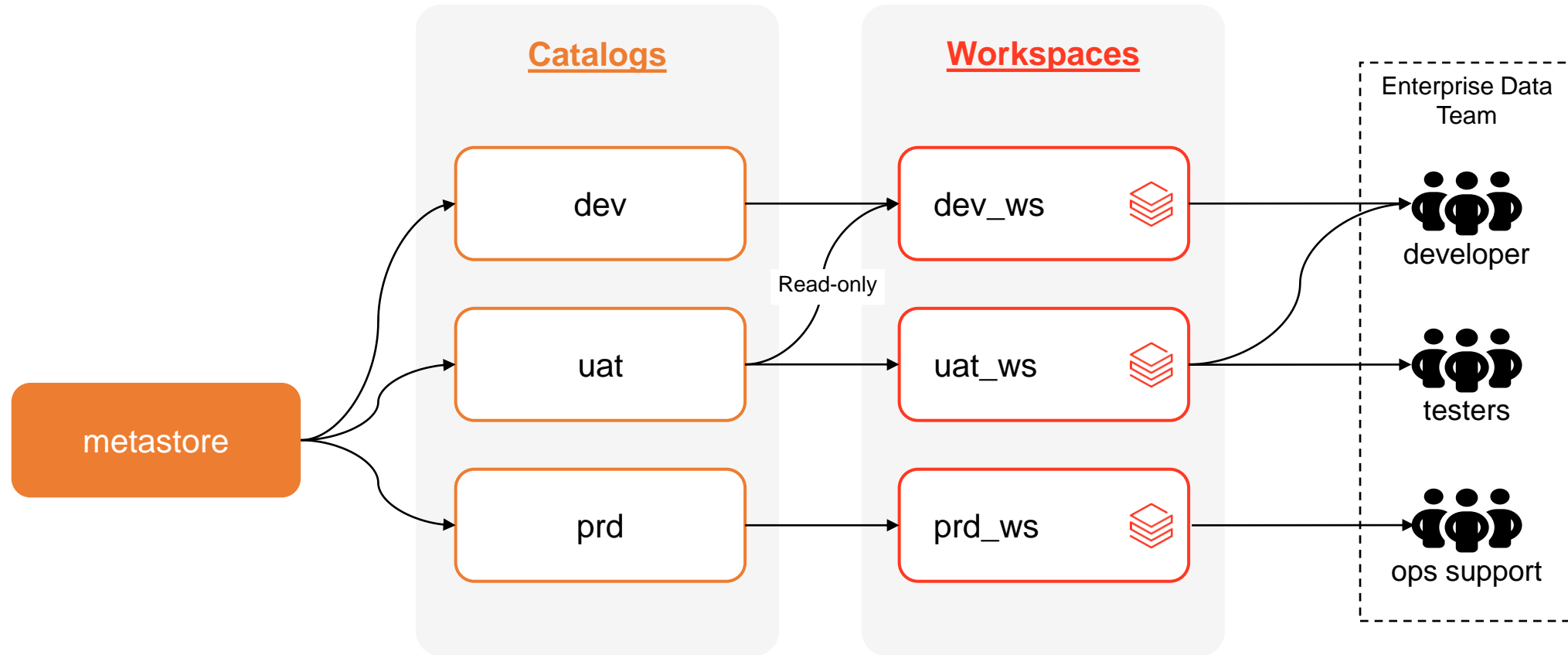
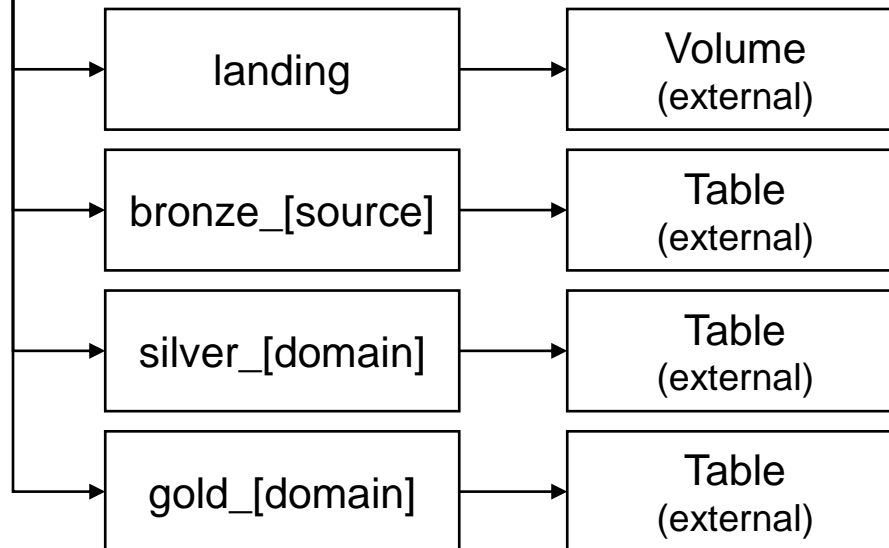# Design a Unity Catalog for our Organization: Ops team
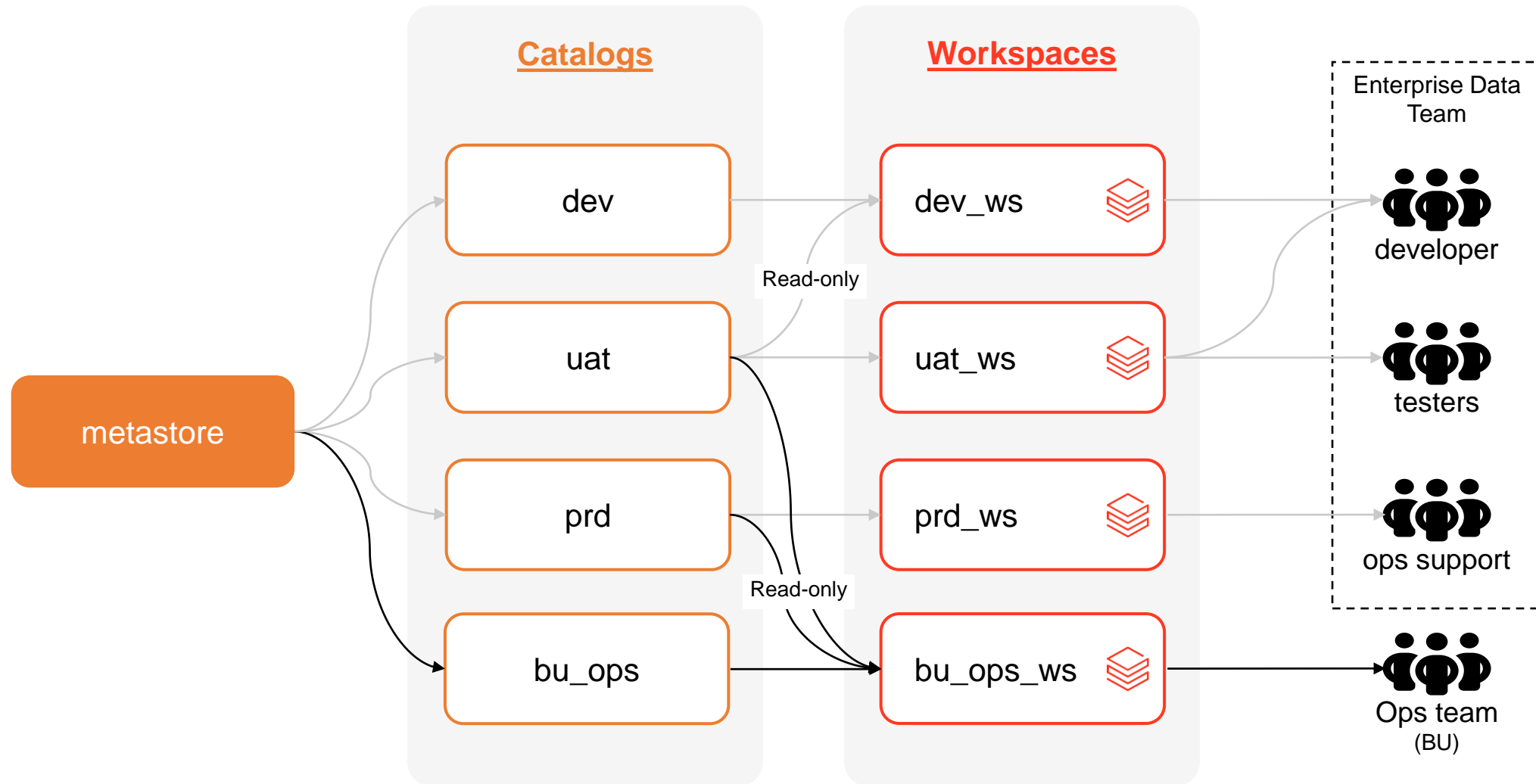
Design a Unity Catalog for our Organization: Schemas design
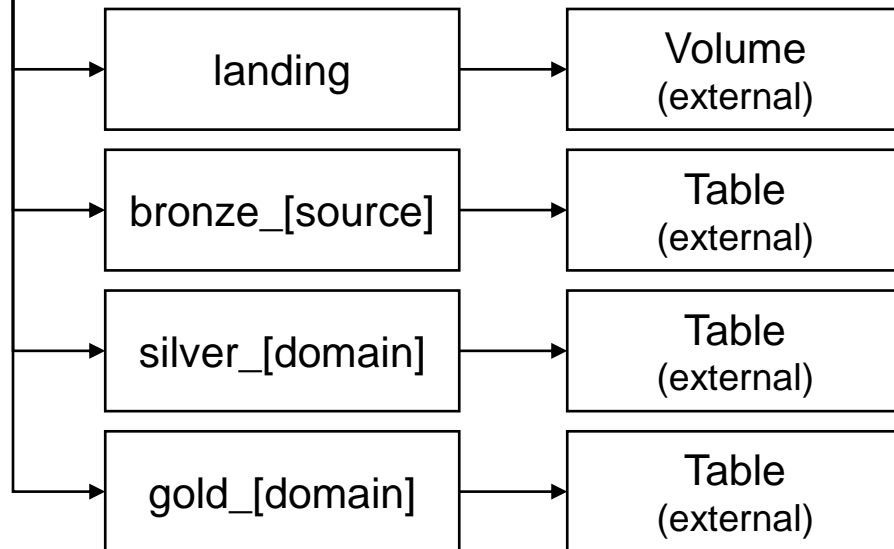
Enterprise Data Team

dev/uat/prd

landing → Volume (external)

bronze_[source] → Table (external)

silver_[domain] → Table (external)

gold_[domain] → Table (external)

* Can create view in all schemas

17

Design a Unity Catalog for our Organization: Schemas design

Demo: Building a Unity Catalog
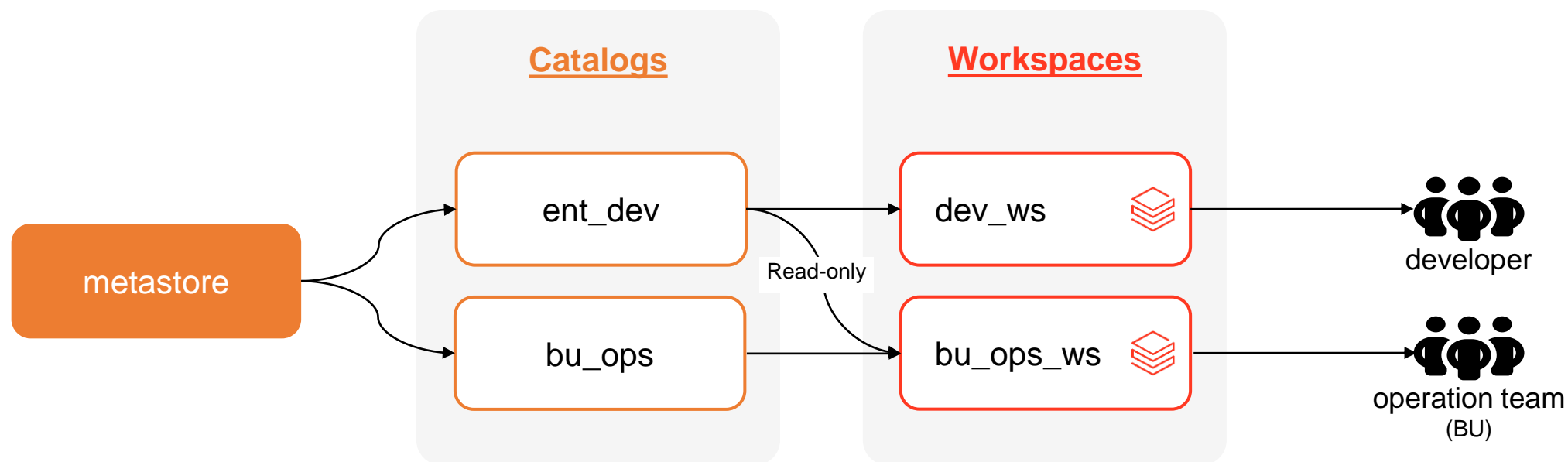
Bug: Assign specific workspace to an external location

PATCH ⌄ https://adb-1979423220815296.16.azuredatabricks.net/api/2.1/unity-catalog/bindings/external_location/ext-loc-ent-data-dev-bronze

Params   Authorization ●   Headers (9)   **Body** ●   Pre-request Script   Tests   Settings

○ none   ○ form-data   ○ x-www-form-urlencoded   ● raw   ○ binary   **JSON** ⌄

```
1  {
2      "add": [
3          {
4              "workspace_id": 1979423220815296,
5              "binding_type": "BINDING_TYPE_READ_WRITE"
6          }
7      ]
8  }
```

Enabling audit log

https://adb-1979423220815296.16.azuredatabricks.net/api/2.0/unity-catalog/metastores/e486b2cb-0c58-4212-8b6d-227cfef74394/systemschemas/access

| PUT ⌄ | https://adb-1979423220815296.16.azuredatabricks.net/api/2.0/unity-catalog/metastores/93a4de60-c24a-4788-9fbd-c8bb1dfc6ef6/systemschema... | Se |

Params   Authorization ●   Headers (8)   Body   Pre-request Script   Tests   Settings

Query Params

Key	Value	
Key	Value	

After all this, we have run through…

1. We learned about the key components of Unity Catalog.

   - Centralizes the governance across multiple Databricks workspaces through the Admin portal.

2. The importance of the Metastore and the 3-levels object model.

   - Only one Metastore per region.

   - Object model: [catalog].[schema].[table]

3. Managed vs External Objects

   - Managed: Fully governed by Databricks, optimized for performance and stored within the Metastore.

   - External: Stored in a user managed cloud storage (ADLS), offers more flexibility but requires additional access management.

4. Granular Access Control & Privileges

   - Permissions inherit down from Catalog > Schema > Table/View/Volume/Function.

5. Designed our own catalogs

   - Enabling the business teams to access the enterprise data safe and securely.

   - The code is in my Github repo.

     - https://github.com/wilson-mok/demo/tree/main/2025/MSDEVMTL/1-Databricks-Unity-Catalog-Intro

Q & A 2

Thank you