# Unity Catalog Design

| | |
|---|---|
| **Document Owner(s)** | @Jay Jiang @Chhavi Kasturia |
| **Work Package** | WP4 |
| **Status** | APPROVED |
| **Consulted** | @David Murphy<br>@Sampath Jagannathan<br>@Vishnu Devarajan (Unlicensed)<br>@Yolandi Jardim<br>@Former user (Deleted)<br>@Sudha Sharma<br>@Davood Shaiek<br>@Ben Sykes (Deactivated)<br>@David Hunter<br>@Neil Belford<br>@Wayne Woodington |
| **List of Approvers** | ☑ @Vishnu Devarajan (Unlicensed) @Sampath Jagannathan<br>☑ @Yolandi Jardim<br>☑ @Former user (Deleted) / @Matt Birch / @Davood Shaiek @Brandon Lay (Unlicensed)<br>☑ @Neil Belford<br>☑ @Sudha Sharma |

## Change Tracker

| | Date | Description of Change | Author |
|---|---|---|---|
| 1 | 21/08/24 | Publish document | @Jay Jiang |

| | | | |
|---|---|---|---|
| 2 | 26/08/2024 | Documented decision by TDA | @Jay Jiang |
| 3 | 16/02/2025 | Add as-built details | @Jay Jiang |
| 4 | 24/07/2025 | Updated Catalog Binding design to include Corporate workspace + catalogs | @Jay Jiang |
| 5 | 12/08/2025 | Update Corporate workspace URLs | @Jay Jiang |

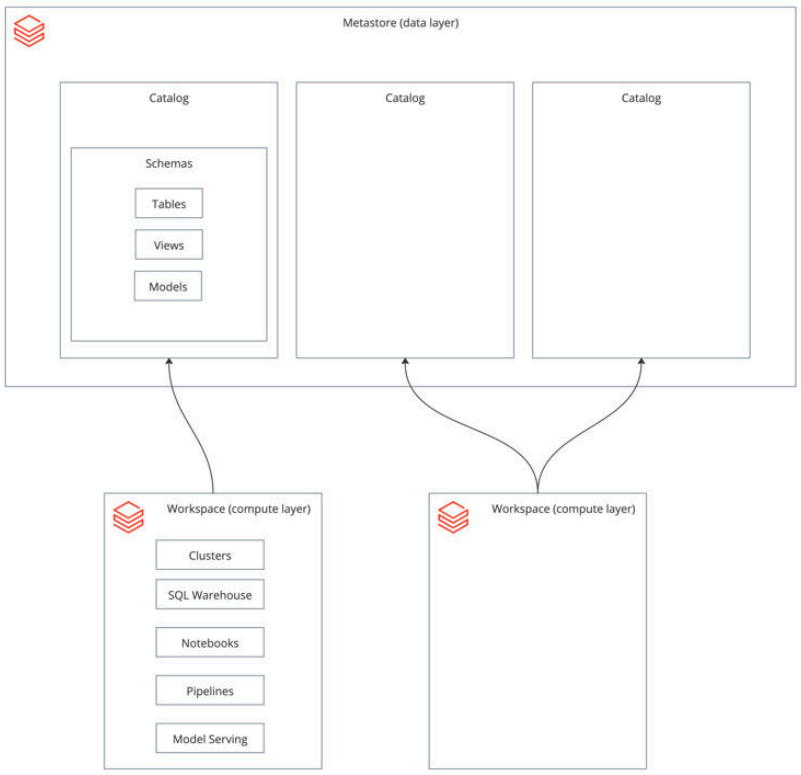

Change Tracker
Overview & Terminology
Number of Workspaces and Catalogs
1. Each Business Unit has its own set of Workspaces
2. Data Product teams within each Business Unit have their own set of Catalogs
3. Data Product teams to maintain their own medallion Schemas within relevant Catalogs
4. S3 Topology
Recommendations for using external locations
Recommendations for using external volumes
Recommendations for using external tables
Workspace Binding


### Overview & Terminology

Before digging into the design, there are some core concepts, terminology and object model unique to Databricks that we need to define.

- `Metastore` - this is the highest level container for all your organisation's **data**
  - Each Databricks **account** can only have 1 metastore per AWS region.
  - Jemena should only have 1 single prod `metastore` for all `data product teams` to hosts all of its data products; data product development stage distinctions (e.g. prod/lab/qa/test) are not made at the metastore level, all data are hosted under `prod` metastore; stage distinction is made at the `Catalog` level (see next item)
  - `data platform team` has a separate nonprod `metastore` for platform level changes under a different databricks account; `data product teams` do not have access to this nonprod `metastore`
- `Catalog` - this is a high level container for data to provide some semantic boundaries.

    - You can think of it like a database in the traditional data warehousing sense.
    - All Catalogs are children under the single `Metastore`.
    - Each `data product team` can have multiple Catalogs corresponding to different stages of the data product development process (prod/lab/qa/test)
- `Workspace` - this is the compute environment `data product teams` operate in.
    - Compute environment distinctions are made at the workspace level; (prod/lab)
    - One `workspace` may access data from multiple `catalogs`
    - Each `data product team` should operate within its own compute environment; i.e. workspace. This is known as LOB (line of business) workspace separation in databricks deployments; see databricks best practice documentation here 5 Best Practices for Data bricks Workspaces




For a more detailed overview, please refer to Unity Catalog best practices | Databricks on AWS

**Number of Workspaces and Catalogs**

As per agreement at `Future Networks DataHub and Analytics - Tactical Design Authority` held on `22 August 2024`

**1. Each Business Unit has its own set of `Workspaces`**

| BU | Description | Schedule | As Built |
|---|---|---|---|
| Elec-Network | Jemena Electricity Network; all teams relevant to Jemena's Electricity Business | WP4 | https://jemena-elec-network-field.cloud.databricks.com<br>https://jemena-elec-network-lab.cloud.databricks.com/ |
| JGN | Jemena Gas Network; all teams revlevant to Jemena's Gas business | FUTURE | TBD |
| Digital | All technology teams shared between Jemena's Electricity and Gas business | WP4 | https://jemena-digital-field.cloud.databricks.com/<br>https://jemena-digital-lab.cloud.databricks.com |
| Corporate | Non technology related corporate teams, e.g. Finance, HR, Marketing etc. | August 2025 | https://jemena-corporate-field.cloud.databricks.com/<br>https://jemena-corporate-lab.cloud.databricks.com/ |

| | | | |
|---|---|---|---|
| | | | Non Production URLs<br>https://dbc-eaec721a-4503.cloud.databricks.com [Lab]<br>https://dbc-2634eda3-a3d5.cloud.databricks.com/ [Field] |
| ... | ... | FUTURE | |

**2. Data Product teams within each Business Unit have their own set of `Catalogs`**

As a part of WP4, 2 data product teams will be onboarded onto databricks platform:

| Team | Name | Function | BU |
|---|---|---|---|
| NRI | Network Reliability &Intelligence | This team's primary focus is on developing network analysis for Jemena's electricity network engineers. Its data product outputs are **consumer aligned**. | Elec-Network |

| | | | |
|---|---|---|---|
| DA | Digital Analytics | This team's primary focus is on ingesting **source aligned** data products from upstream operational systems such as SIQ, SAP HANA, Zepben Workbench. | Digital |

#### 3. Data Product teams to maintain their own medallion `Schemas` within relevant `Catalogs`

Data Product teams are to follow medallion architecture when designing data products and processing pipelines.

In the context of the first 2 data product teams, DA team will primarily be ingesting source aligned data products into DA Catalog's `bronze` schemas and processing into `silver` and `gold` schemas.

DA team's `gold` schemas can then be exposed to NRI team for use. NRI team will then create their own pipelines, populating NRI's `gold` schemas.

#### 4. S3 Topology

Each catalog and each schema will have a corresponding s3 bucket defined. Retention to follow Data Storage Design .

For `landing` layer, ingestion access for clusters would be configured via databricks UC `Volumes` as opposed to `External Locations`. Unity Catalog best practices | Databricks on AWS. Each source system's `landing` layer (if exists) should have its own underlying `s3` buckets to enable varying configuration in bucket lifecycle policy. These buckets can be optionally created in either databricks AWS account or upstream AWS account, creator would need to configure cross account IAM policy to enable access via databricks Unity Catalog.

from Databricks docs

**Recommendations for using external locations**

Recommendations for granting permissions on external locations:

- Grant the ability to create external locations only to an administrator who is tasked with setting up connections between Unity Catalog and cloud storage, or to trusted data engineers.

  External locations provide access from within Unity Catalog to a broadly encompassing location in cloud storage—for example, an entire bucket or container (s3://mybucket) or a broad subpath (s3://mybucket/alotofdata). The intention is that a cloud administrator can be involved in setting up a few external locations and then delegate the responsibility of managing those locations to a Databricks administrator in your organization. The Databricks administrator can then further organize the external location into areas with more granular permissions by registering external volumes or external tables at specific prefixes under the external location.

  Because external locations are so encompassing, Databricks recommends giving the `CREATE EXTERNAL LOCATION` permission only to an administrator who is tasked with setting up connections between Unity Catalog and cloud storage, or to trusted data engineers. To provide other users with more granular access, Databricks recommends registering external tables or volumes on top of external locations and granting users access to data using volumes or tables. Since tables and volumes are children of a catalog and schema, catalog or schema administrators have the ultimate control over access permissions.

  You can also control access to an external location by binding it to specific workspaces. See (Optional) Assign an external location to specific workspaces.

- Don't grant general `READ FILES` or `WRITE FILES` permissions on external locations to end users.

  With the availability of volumes, users shouldn't use external locations for anything but creating tables, volumes, or managed locations. They should not use external locations for path-based access for data science or other non-tabular data use cases.

  Volumes provide support for working with files using SQL commands, dbutils, Spark APIs, REST APIs, Terraform, and a user interface for browsing, uploading, and downloading files. Moreover, volumes offer a FUSE mount that is accessible on the local file system under `/Volumes/<catalog_name>/<schema_name>/<volume_name>/`. The

FUSE mount allows data scientists and ML engineers to access files as if they were in a local filesystem, as required by many machine learning or operating system libraries.

If you must grant direct access to files in an external location (for exploring files in cloud storage before a user creates an external table or volume, for example), you can grant `READ FILES`. Use cases for granting `WRITE FILES` are rare.

You should use external locations to do the following:

- Register external tables and volumes using the `CREATE EXTERNAL VOLUME` or `CREATE TABLE` commands.
- Explore existing files in cloud storage before you create an external table or volume at a specific prefix. The `READ FILES` privilege is a precondition.
- Register a location as managed storage for catalogs and schemas instead of the metastore root bucket. The `CREATE MANAGED STORAGE` privilege is a precondition.

More recommendations for using external locations:

- Avoid path overlap conflicts: never create external volumes or tables at the root of an external location.

  If you do create external volumes or tables at the external location root, you can't create any additional external volumes or tables on the external location. Instead, create external volumes or tables on a sub-directory inside the external location.

#### Recommendations for using external volumes

You should use external volumes to do the following:

- Register landing areas for raw data produced by external systems to support its processing in the early stages of ETL pipelines and other data engineering activities.
- Register staging locations for ingestion, for example, using Auto Loader, `COPY INTO`, or CTAS ( `CREATE TABLE AS` ) statements.
- Provide file storage locations for data scientists, data analysts, and machine learning engineers to use as parts of their exploratory data analysis and other data science tasks, when managed volumes are not an option.
- Give Databricks users access to arbitrary files produced and deposited in cloud storage by other systems, for example, large collections of unstructured data (such as image, audio, video, and PDF files) captured by surveillance systems or IoT devices, or library files (JARs and Python wheel files) exported from local dependency management systems or CI/CD pipelines.

- Store operational data, such as logging or checkpointing files, when managed volumes are not an option.

More recommendations for using external volumes:

- Databricks recommends that you create external volumes from one external location within one schema.

Tip

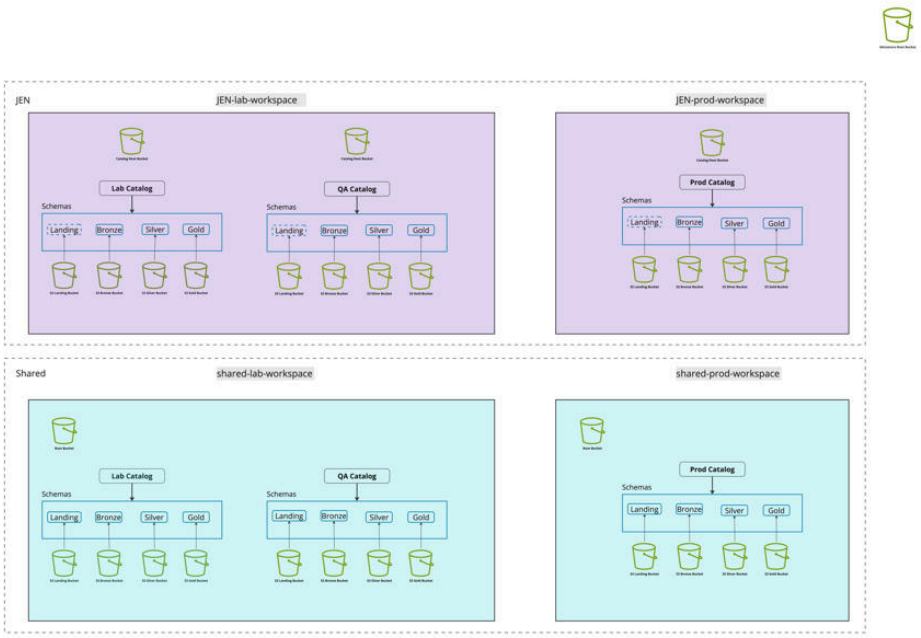
For ingestion use cases in which the data is copied to another location—for example using Auto Loader or `COPY INTO`—use external volumes. Use external tables when you want to query data in place as a table, with no copy involved.

#### Recommendations for using external tables

You should use external tables to support normal querying patterns on top of data stored in cloud storage, when creating managed tables is not an option.

More recommendations for using external tables:

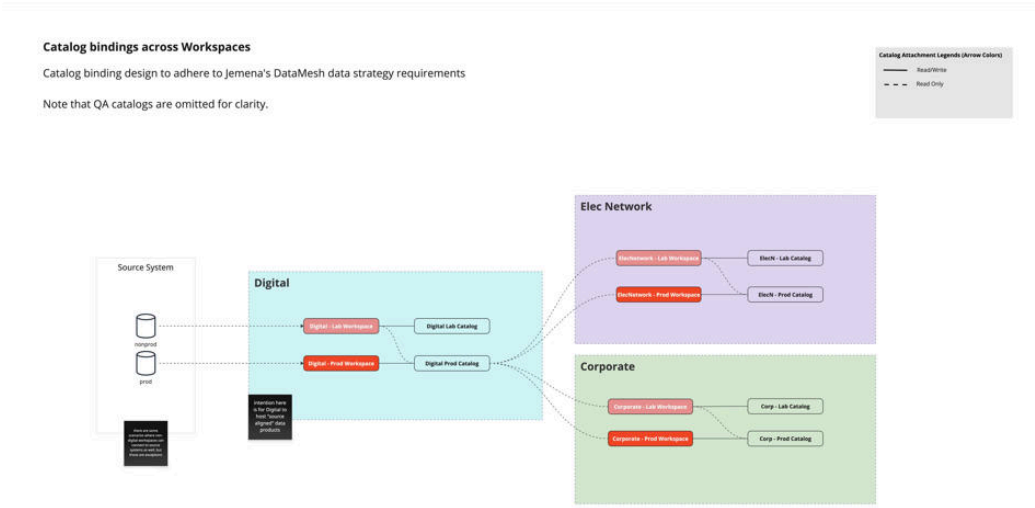
- Databricks recommends that you create external tables using one external location per schema.
- Databricks strongly recommends against registering a table as an external table in more than one metastore due to the risk of consistency issues. For example, a change to the schema in one metastore will not register in the second metastore. Use Delta Sharing for sharing data between metastores. See Share data securely using Delta Sharing.



### Workspace Binding

Data access between NRI and DA teams are enabled via workspace binding as shown in diagram below.

Visibility of `schemas` within each `catalog` can be controlled, refer to RBAC High Level Design for exact design.




Previous Versions

**Catalog bindings across Workspaces**

Catalog binding design to adhere to Jemena's DataMesh data strategy requirements