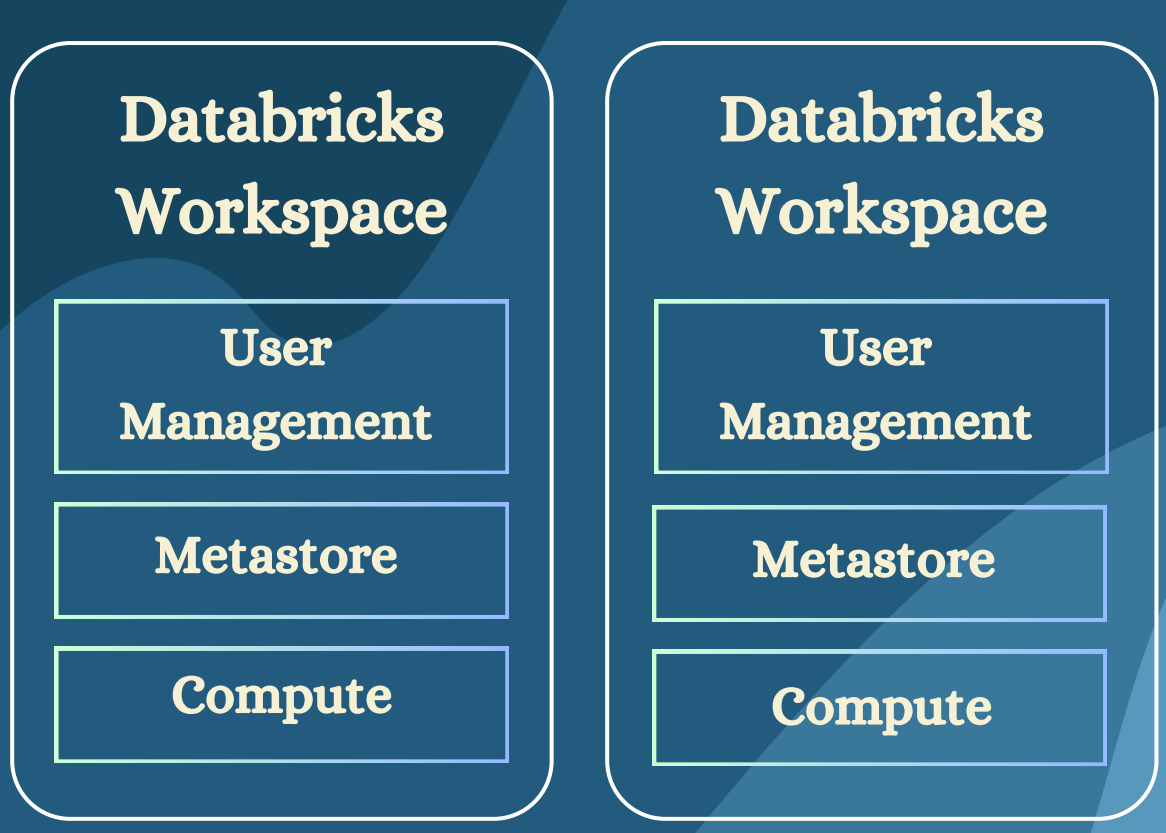
WTD Analytics

UNITY CATALOG IN DATABRICKS

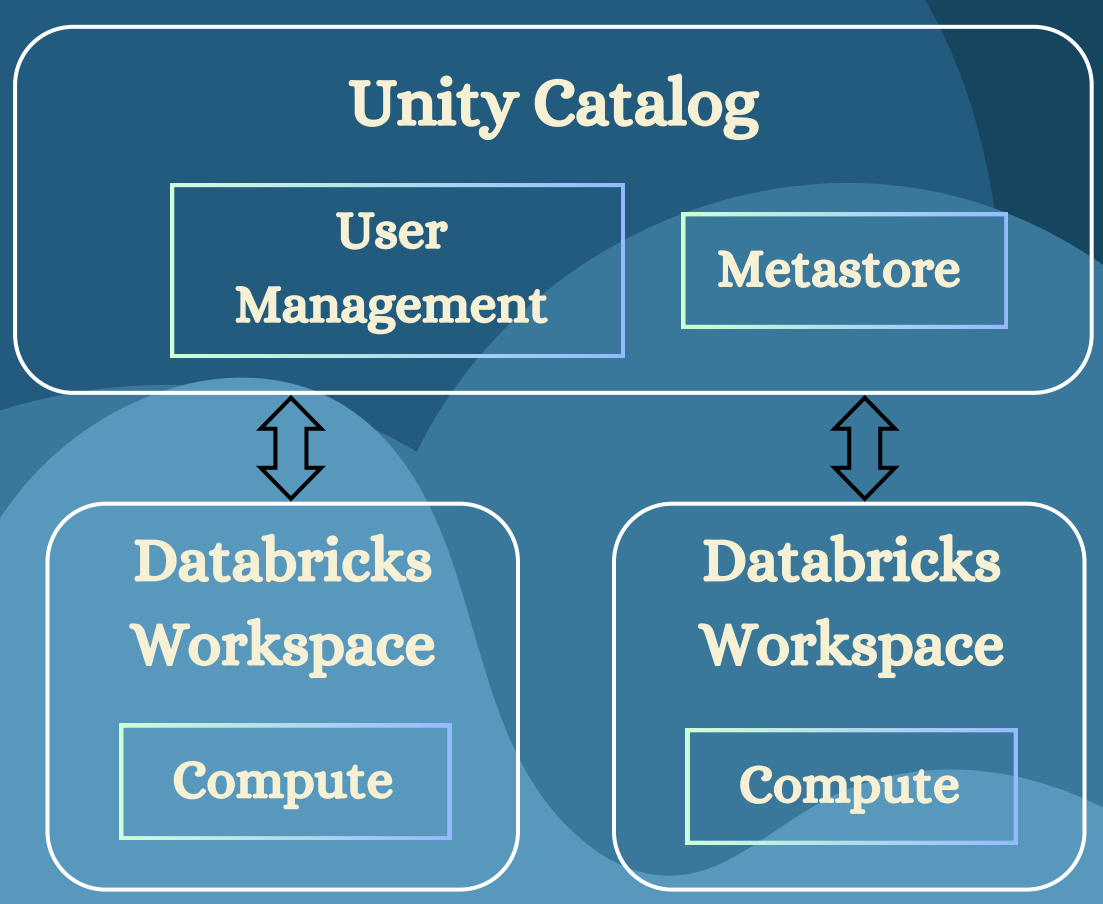
WHAT IS UNITY CATALOG

- Unity Catalog is a unified governance solution for managing data and AI assets across Databricks workspaces.
- It offers a centralized platform for controlling access, auditing, tracking lineage, and discovering data assets.

WITHOUT UNITY CATALOG



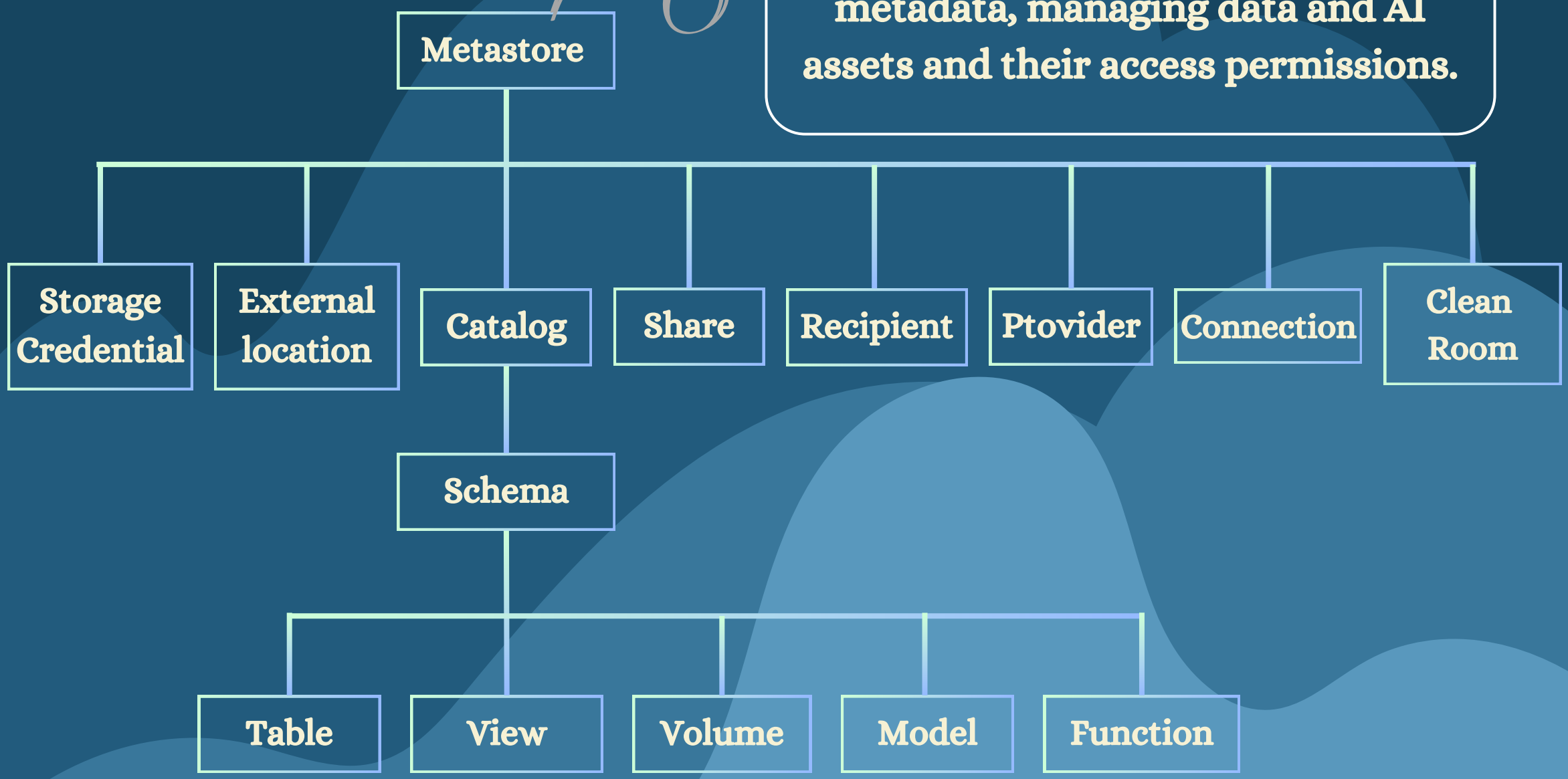
WITH UNITY CATALOG



KEY FEATURES:

- **Centralized Access Control**
  - Define data access policies in one place, enforce them across all workspaces.
- **Standards-Compliant Security Model**
  - Based on ANSI SQL, permissions are managed at catalog, schema, table, and view levels using familiar syntax.
- **Built-In Auditing and Lineage**
  - Automatic logging of user access and tracking of data lineage, showing the creation and usage of data assets.
- **Data Discovery**
  - Tag and document data assets, with a search interface for easy discovery.

UNITY CATALOG OBJECT MODEL AND HIERARCHY



Catalogs

- Organize data assets; often reflect organizational units or development scopes.
- Non-data securable objects like storage credentials and external locations also live at this level.

Schema

- Also known as databases; contain tables, views, volumes, AI models, and functions.
- Organize assets into logical categories, usually representing a specific use case, project, or team.

Data & AI Objects

- **Volumes:** Logical storage for unstructured, non-tabular data
- **Tables:** Collections of data, either managed or external.
- **Views:** Saved queries against tables.
- **Functions:** Saved logic that returns values or sets of rows.
- **Models:** AI models with MLflow.

UNITY CATALOG GOVERNANCE MODELS

Centralized Governance:

- Governance administrators own the metastore.
- Administrators can take ownership of any object and manage permissions.

Distributed Governance:

- Data domains are managed at the catalog level.
- Catalog owners manage all assets and governance within their domain.

Best Practice:

- Set a group as the metastore admin or catalog owner for consistent management.

```sql
-- Assign a group as the metastore admin
GRANT CREATE ON METASTORE TO `data-admins-group`;

-- Assign a group as a catalog owner
GRANT OWNERSHIP ON CATALOG my_catalog TO `catalog-admins-group`;
```

STORAGE SEPARATION AND HIERARCHY

Data Separation:

- Store specific data types in designated cloud accounts or buckets.
- Example: HR production data stored in s3://mycompany-hr-prod/unity-catalog.

Storage Hierarchy:

- Storage locations can be configured at the metastore, catalog, or schema level.
- **Hierarchical Evaluation:**
    a. Schema-level location
    b. Catalog-level location
    c. Metastore-level location

```
-- Set storage location at the metastore level
ALTER METASTORE SET STORAGE ROOT 's3://mycompany-unity-metastore/';

-- Set storage location at the catalog level
ALTER CATALOG my_catalog SET STORAGE ROOT 's3://mycompany-catalog-storage/';

-- Set storage location at the schema level
ALTER SCHEMA my_schema SET STORAGE ROOT 's3://mycompany-schema-storage/';
```

DATA ACCESS CONTROL IN DESIGNATED ENVIRONMENTS

Environment-Specific Access:

- Workspaces are primary data processing environments.
- Catalogs as primary data domains.
- Metastore admins and catalog owners can bind catalogs to specific workspaces.

Use Cases:

- Isolate production data from development environments.
- Ensure data compliance by restricting access to specific environments.

```
-- Bind a catalog to a specific workspace
ALTER CATALOG my_catalog SET WORKSPACE `prod_workspace`;
```

CONFIGURING UNITY CATALOG METASTORE

Metastore Overview:

- Top-level container managing data assets (tables, views, volumes).
- Configure one metastore per region for Databricks workspaces.

Best Practices:

- Use a dedicated bucket for metastore-managed storage.
- Avoid giving direct access to the managed storage location.
- Prefer catalog-level managed storage over metastore-level.

```
-- Create a new metastore
CREATE METASTORE my_metastore LOCATION 's3://mycompany-metastore-bucket/';

-- Assign managed storage to a catalog within the metastore
ALTER CATALOG my_catalog SET STORAGE ROOT 's3://mycompany-catalog-storage/';
```

EXTERNAL LOCATIONS AND STORAGE CREDENTIALS

External Locations:

- Combine storage credentials with a cloud storage path.
- Use to register external tables and volumes.

Best Practices:

- Limit direct access to external locations.
- Avoid using external locations for path-based access outside of registered tables or volumes.
- Use volumes for SQL-based file management and access.

```
-- Create a storage credential for an external location
CREATE CREDENTIAL my_credential WITH STORAGE_ACCOUNT_KEY = 'your-key';

-- Register an external location
CREATE EXTERNAL LOCATION my_location
WITH URL 's3://mycompany-external-storage/'
CREDENTIAL my_credential;

-- Register an external table with the external location
CREATE TABLE my_table
USING DELTA
LOCATION 's3://mycompany-external-storage/my-table/';
```