databricks

Manage Data Access for Analytics

databricks

Introduction to Unity Catalog

Module Agenda

Manage Data Access for Analytics with Unity Catalog

Introduction to Unity Catalog

DE 6.1a - Managing principals in Unity Catalog (demo)

DE 6.1b - Managing Unity Catalog metastores (demo)

DE 6.1c - Creating compute resources for Unity Catalog access (demo)

DE 6.4 - Creating and governing data objects with Unity Catalog

DE 6.5 - Create and Share Tables in Unity Catalog

DE 6.6 - Create external tables in Unity Catalog (demo)

DE 6.7 - Upgrade a table to Unity Catalog

DE 6.8 - Create views and limit table access (demo)

Course Objectives

By the end of this course, you will be able to:

1. Describe Unity Catalog key concepts and how it integrates with the Databricks platform
2. Manage groups, users and service principals
3. Create and manage a Unity Catalog metastore

Course Objectives

By the end of this course, you will be able to:

1. Describe Unity Catalog key concepts and how it integrates with the Databricks platform
2. Manage groups, users and service principals
3. Create and manage a Unity Catalog metastore

Course Objectives

By the end of this course, you will be able to:

1. Describe Unity Catalog key concepts and how it integrates with the Databricks platform
2. Manage groups, users and service principals
3. Create and manage a Unity Catalog metastore

Overview of Data Governance

Data Governance Overview

DAALAC

Four key functional areas

Data Access Control

AC

Control who has access to which data

Access should be granted to only those who need the data.

Lock down data and data generating artefacts. (such as Files, Tables, ML Models)

Data Access Audit

AA

Capture and record all access to data

How data has been accessed, when, and by whom.

Data Lineage

L

Capture upstream sources and downstream consumers

Helps to check and verify whether data comes from verifiable trusted sources.

Data Discovery

D

Ability to search for and discover authorized assets

We should be able to readily search, existing data assets.

Data Governance Overview

Four key functional areas

Data Access Control

Control who has access to which data

Data Access Audit

Capture and record all access to data

Data Lineage

Capture upstream sources and downstream consumers

Data Discovery

Ability to search for and discover authorized assets

Data Governance Overview

Four key functional areas

Data Access Control

Control who has access to which data

Data Access Audit

Capture and record all access to data

Data Lineage

Capture upstream sources and downstream consumers

Data Discovery

Ability to search for and discover authorized assets

Data Governance Overview

Four key functional areas

Data Access Control

Control who has access to which data

Data Access Audit

Capture and record all access to data

Data Lineage

Capture upstream sources and downstream consumers

Data Discovery

Ability to search for and discover authorized assets

Data Governance Overview

Four key functional areas

Data Access Control

Control who has access to which data

Data Access Audit

Capture and record all access to data

Data Lineage

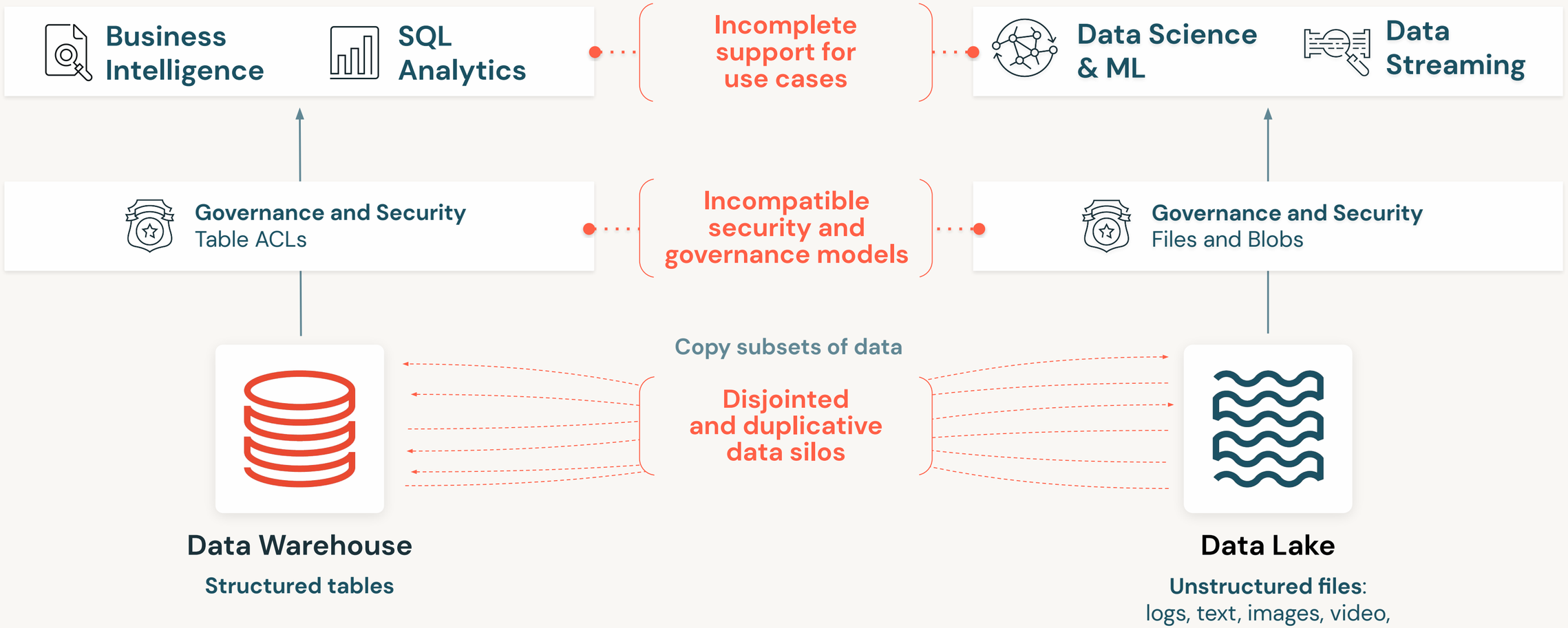
Capture upstream sources and downstream consumers

Data Discovery

Ability to search for and discover authorized assets

Data Governance Overview

Challenges in the Data Lake



Data Governance Overview

Challenges in the Data Lake, related to access control

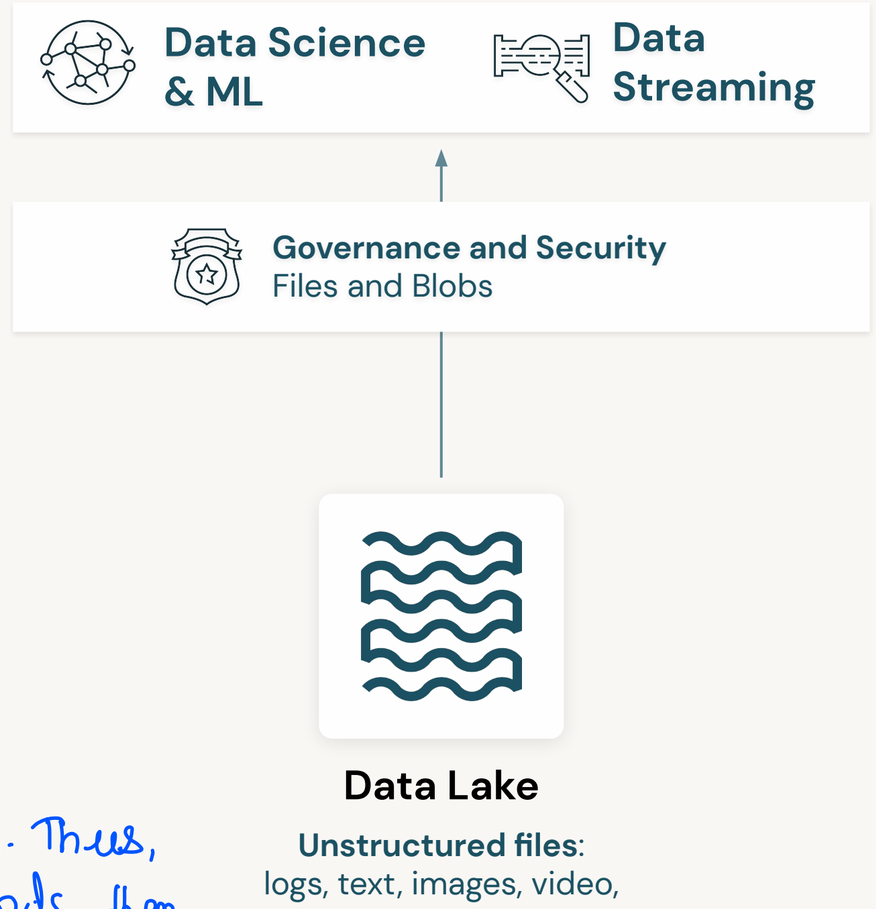
No fine-grained access controls

No common metadata layer

Non-standard cloud-specific governance model

Hard to audit

No common governance model for different data asset types



Cloud provides only provide access-controls to the file-level. Thus, if we want to provide access to, say, only a subset of records, then that is not readily possible.

Databricks Unity Catalog

Overview

Unify governance across clouds

Fine-grained governance for data lakes across clouds - based on open standard ANSI SQL.

Unify data and AI assets

Centrally share, audit, secure and manage all data types with one simple interface.

Unify existing catalogs

Works in concert with existing data, storage, and catalogs - no hard migration required.

Unity catalog solves the above problems, by integrating a centralized Hub, within the Databricks lakehouse Platform, for administering, securing, auditing data.

Databricks Unity Catalog

Overview

Unify governance across clouds

Fine-grained governance for data lakes across clouds - based on open standard ANSI SQL.

Unify data and AI assets

Centrally share, audit, secure and manage all data types with one simple interface.

Unify existing catalogs

Works in concert with existing data, storage, and catalogs - no hard migration required.

1

allows to define our data access rules only once, where they can be applied to multiple workspaces, languages, clouds, use cases.

Databricks Unity Catalog

Overview

Unify governance across clouds

Fine-grained governance for data lakes across clouds - based on open standard ANSI SQL.

Unify data and AI assets

Centrally share, audit, secure and manage all data types with one simple interface.

Unify existing catalogs

Works in concert with existing data, storage, and catalogs - no hard migration required.

Databricks Unity Catalog

Overview

Unify governance across clouds

Fine-grained governance for data lakes across clouds - based on open standard ANSI SQL.

Unify data and AI assets

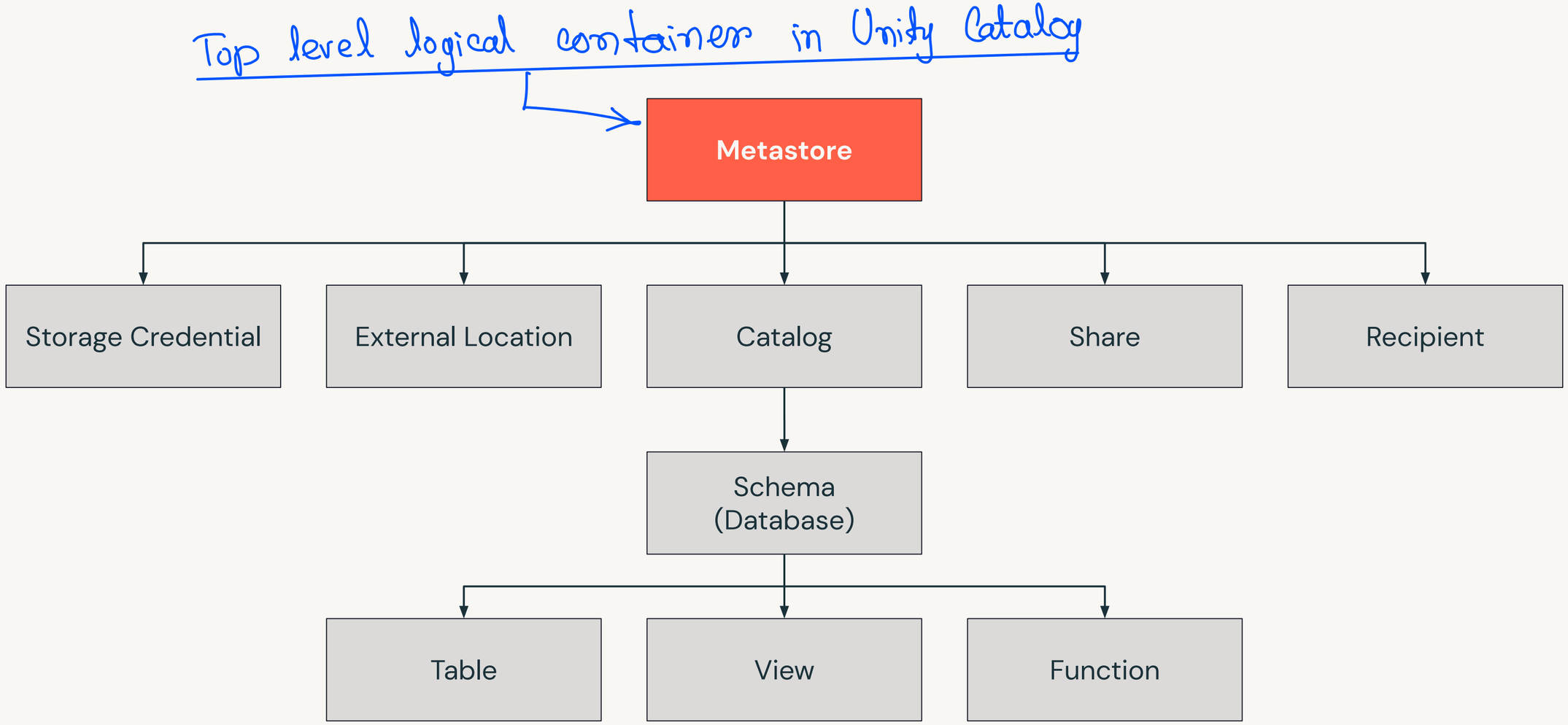
Centrally share, audit, secure and manage all data types with one simple interface.

Unify existing catalogs

Works in concert with existing data, storage, and catalogs - no hard migration required.

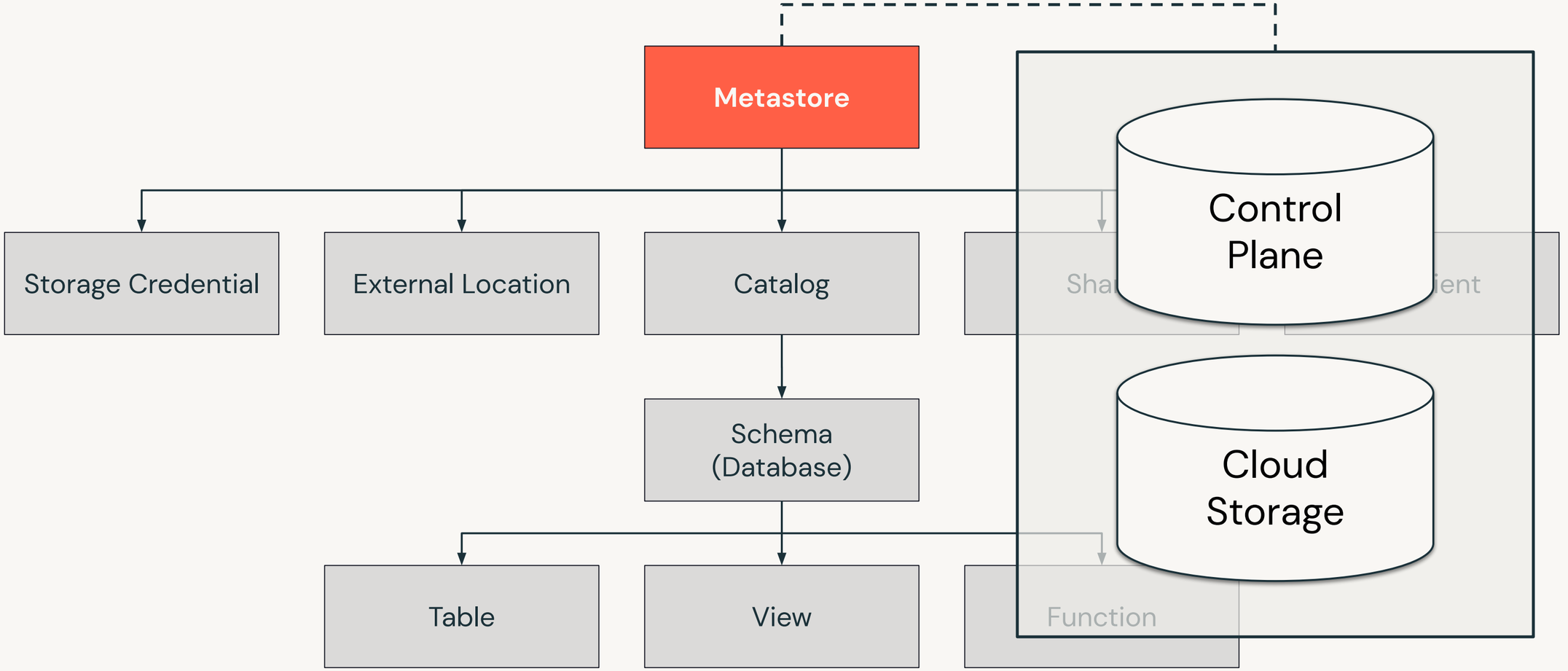
Unity Catalog Key Concepts

Key Concepts

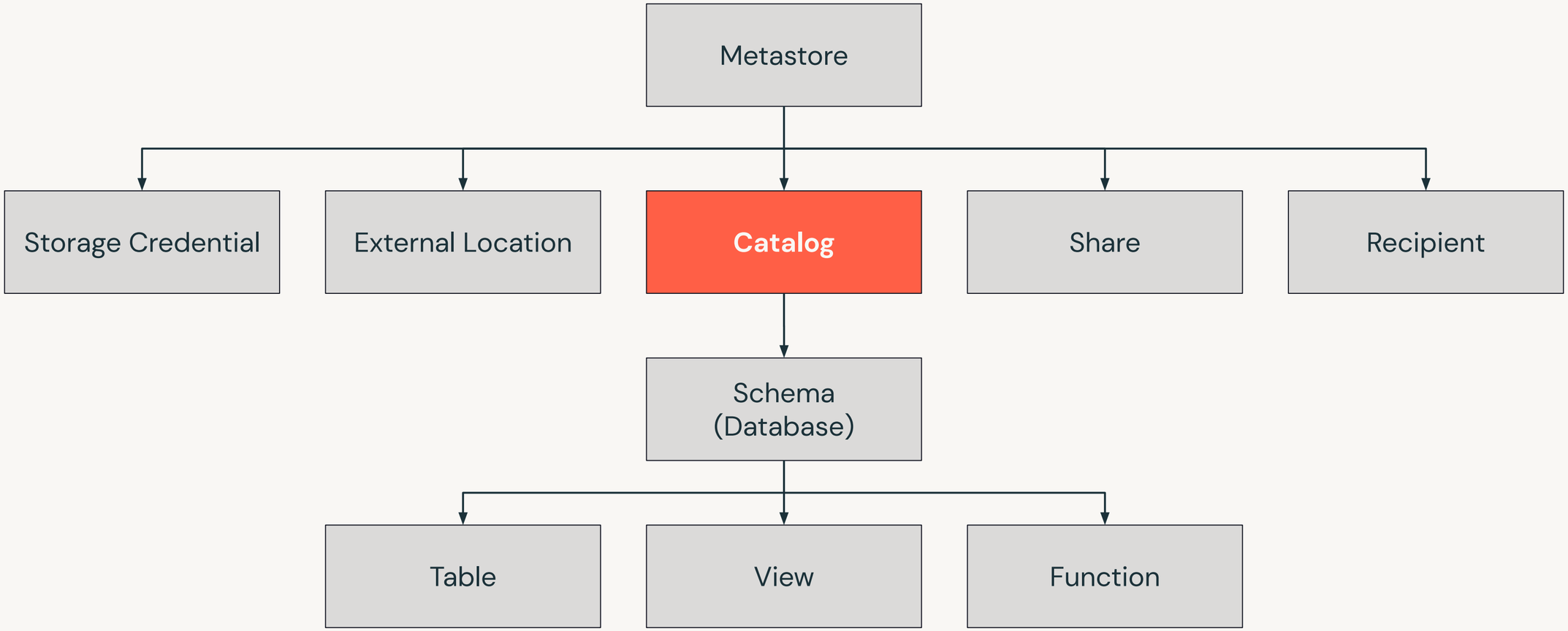


Key Concepts

Metastore elements



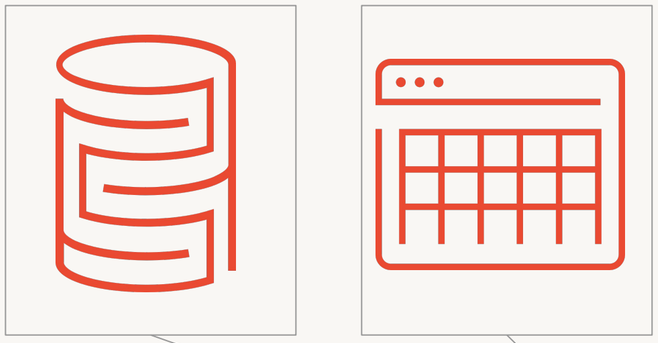
Key Concepts



Key Concepts

Three-level Namespace

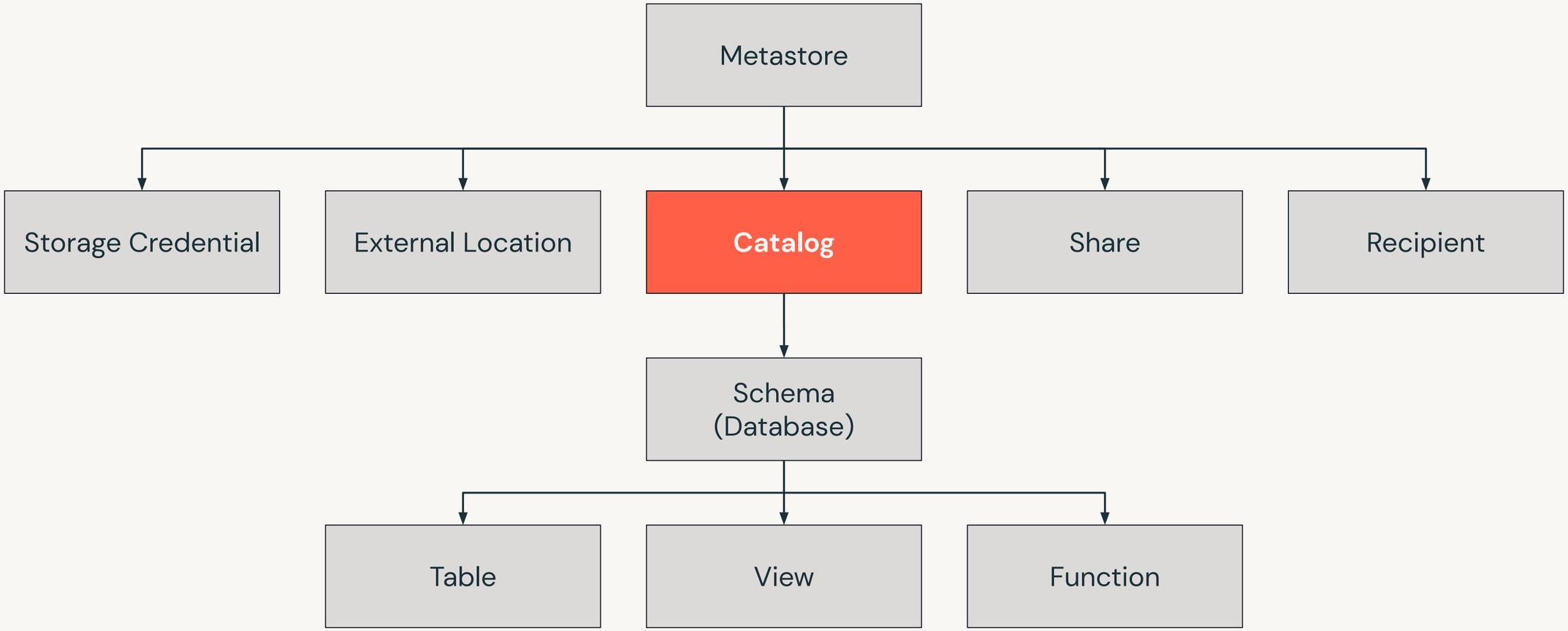
Traditional SQL two-level namespace



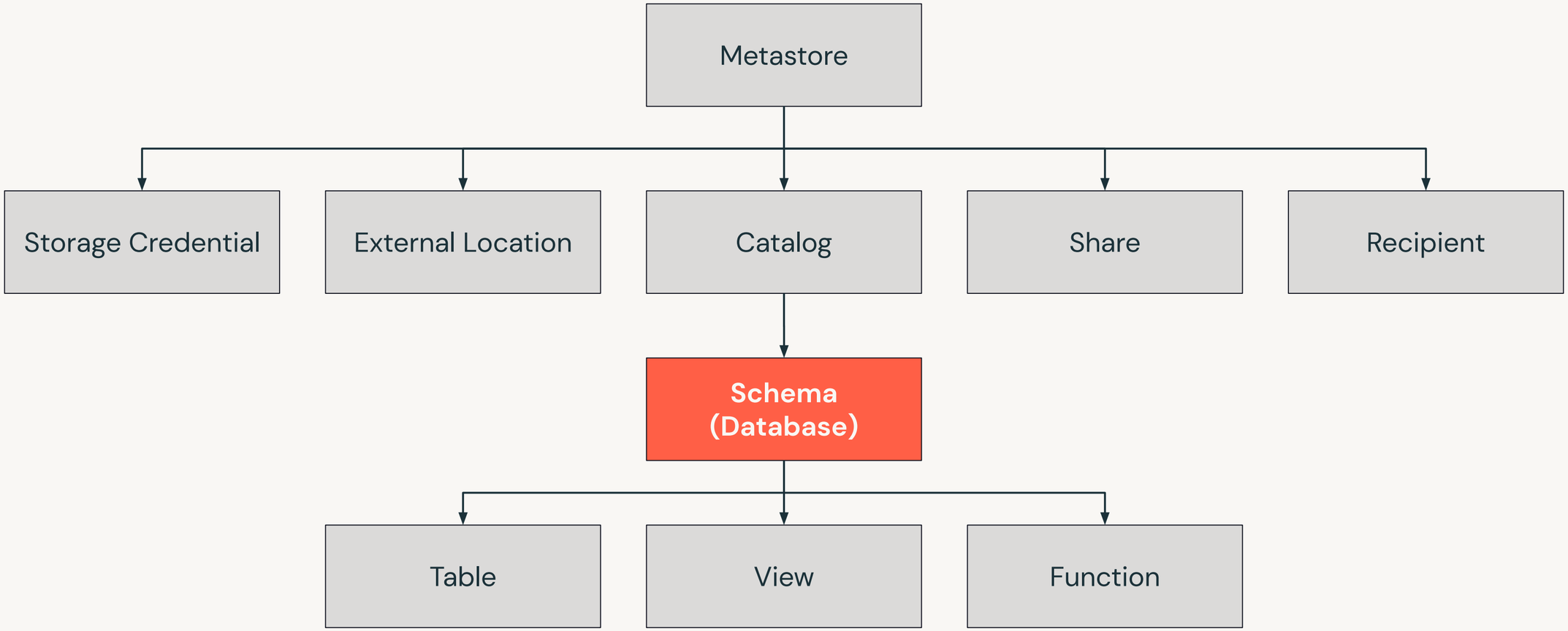
Unity Catalog three-level namespace

Complete SQL references in the Unity Catalog uses a 3-level namespaces.

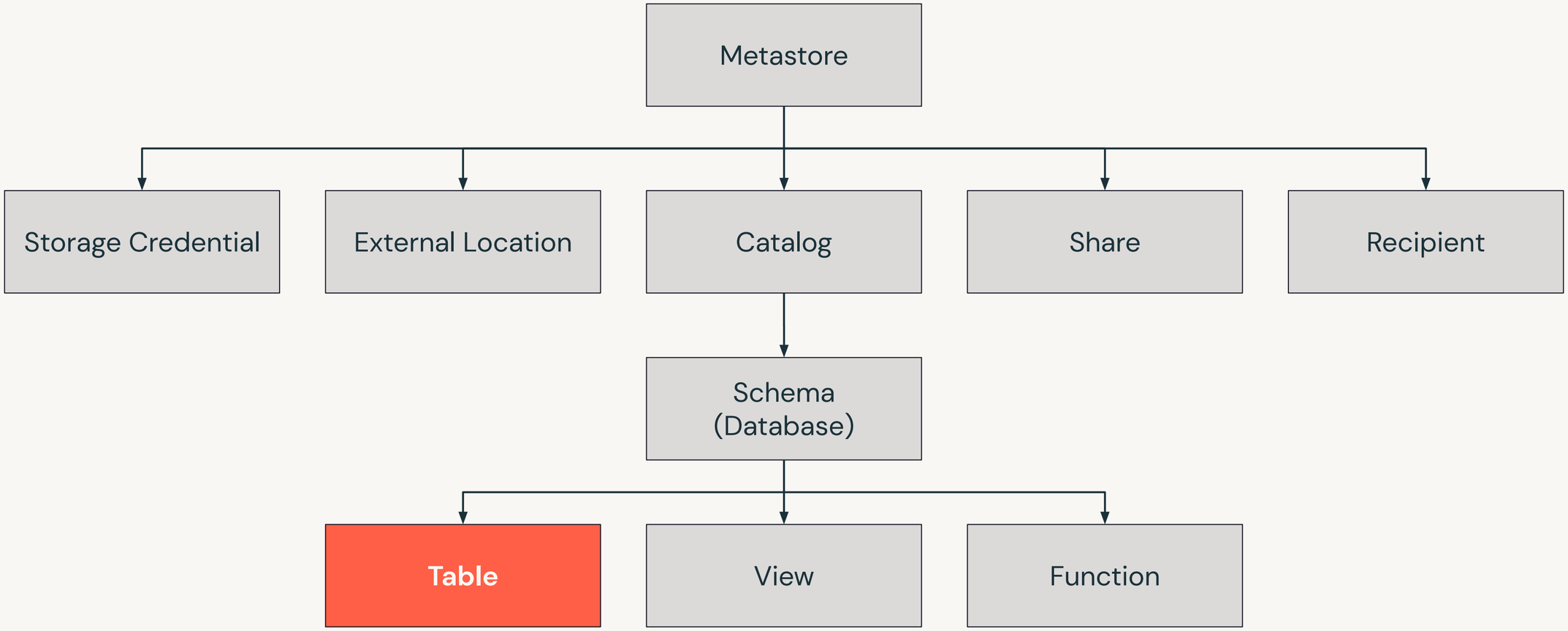
Key Concepts



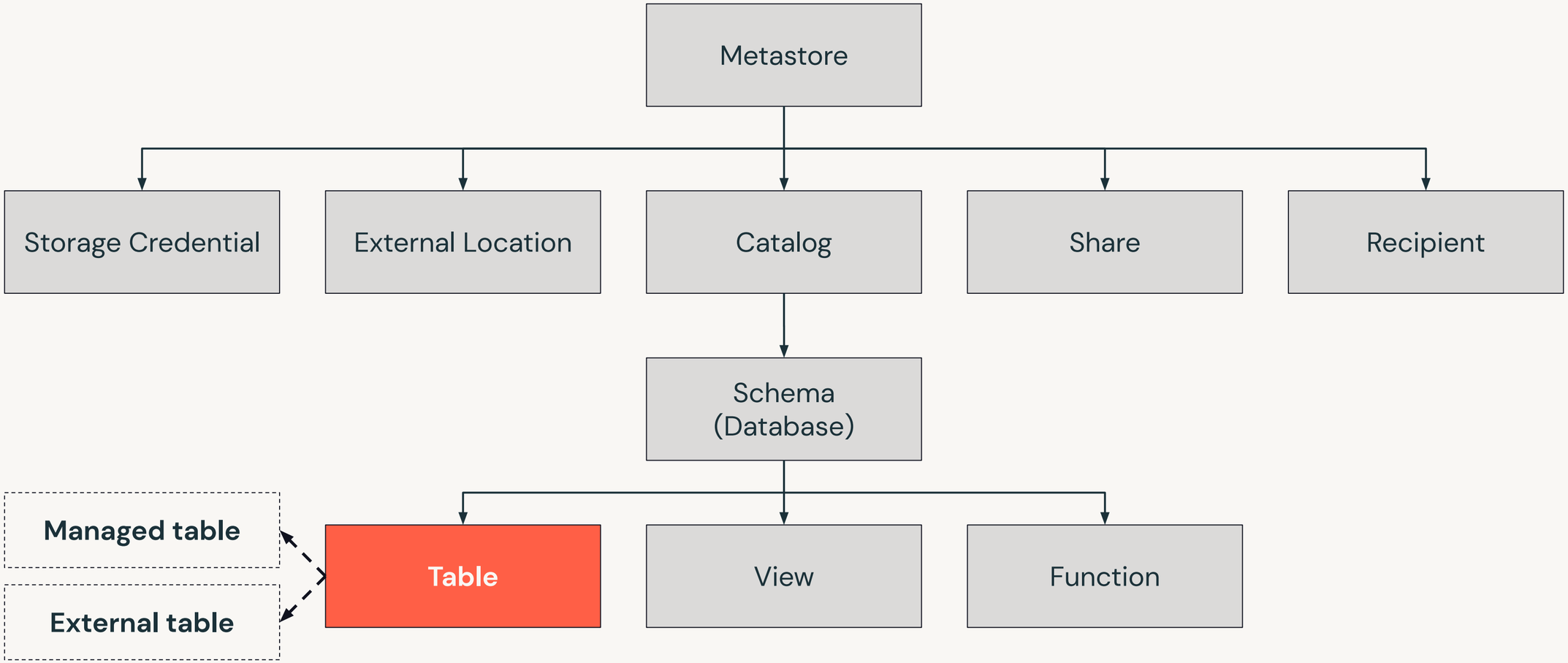
Key Concepts



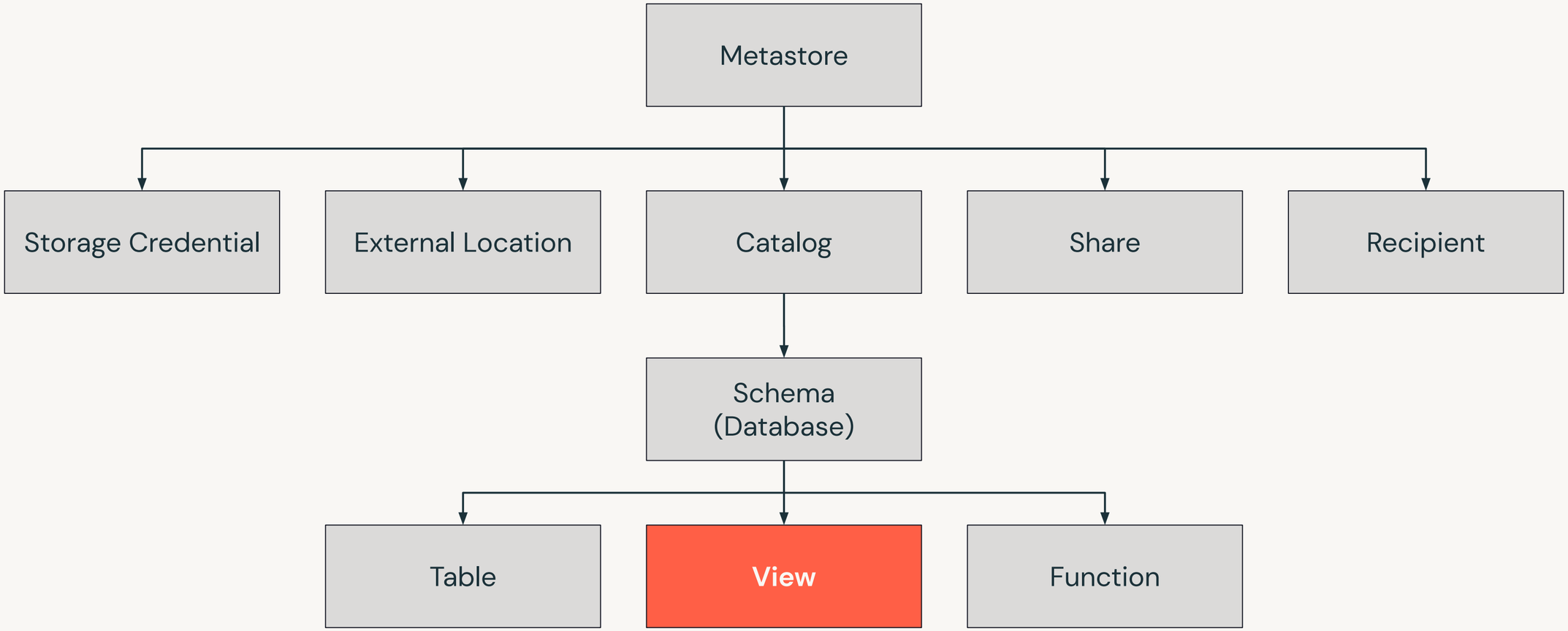
Key Concepts



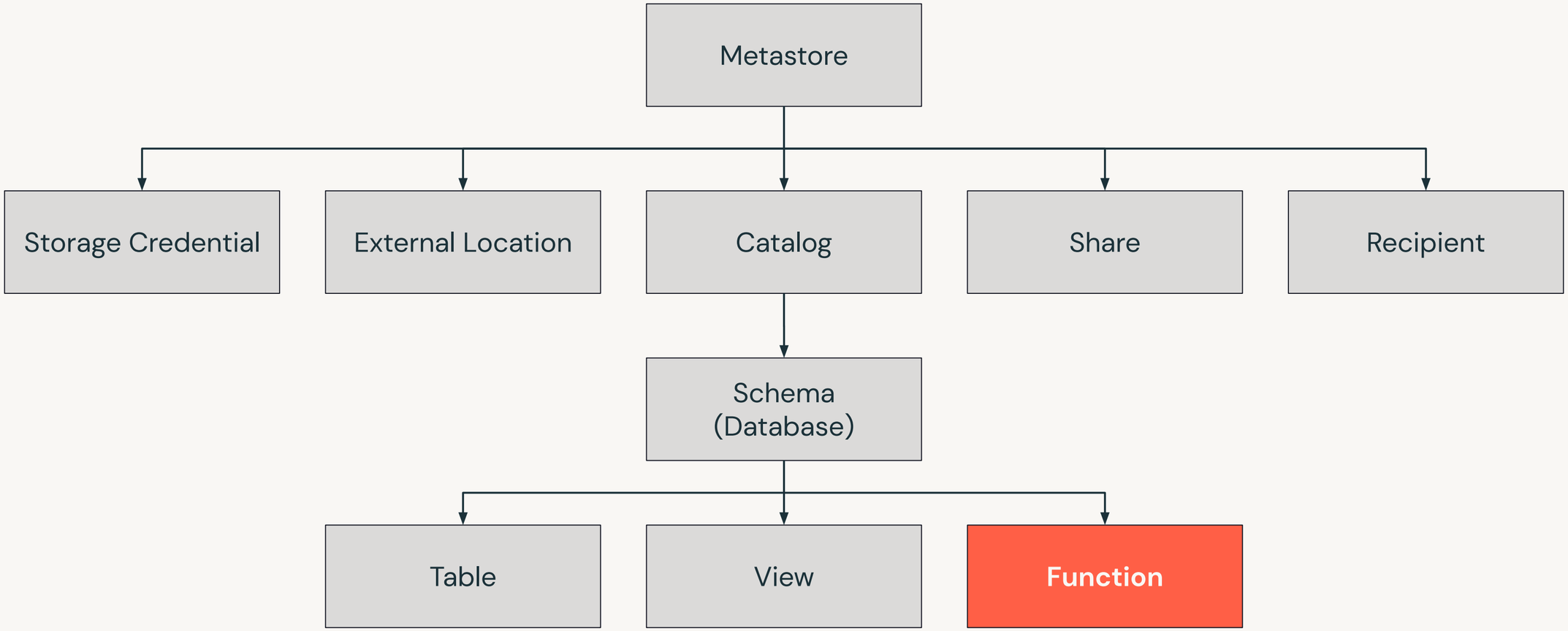
Key Concepts



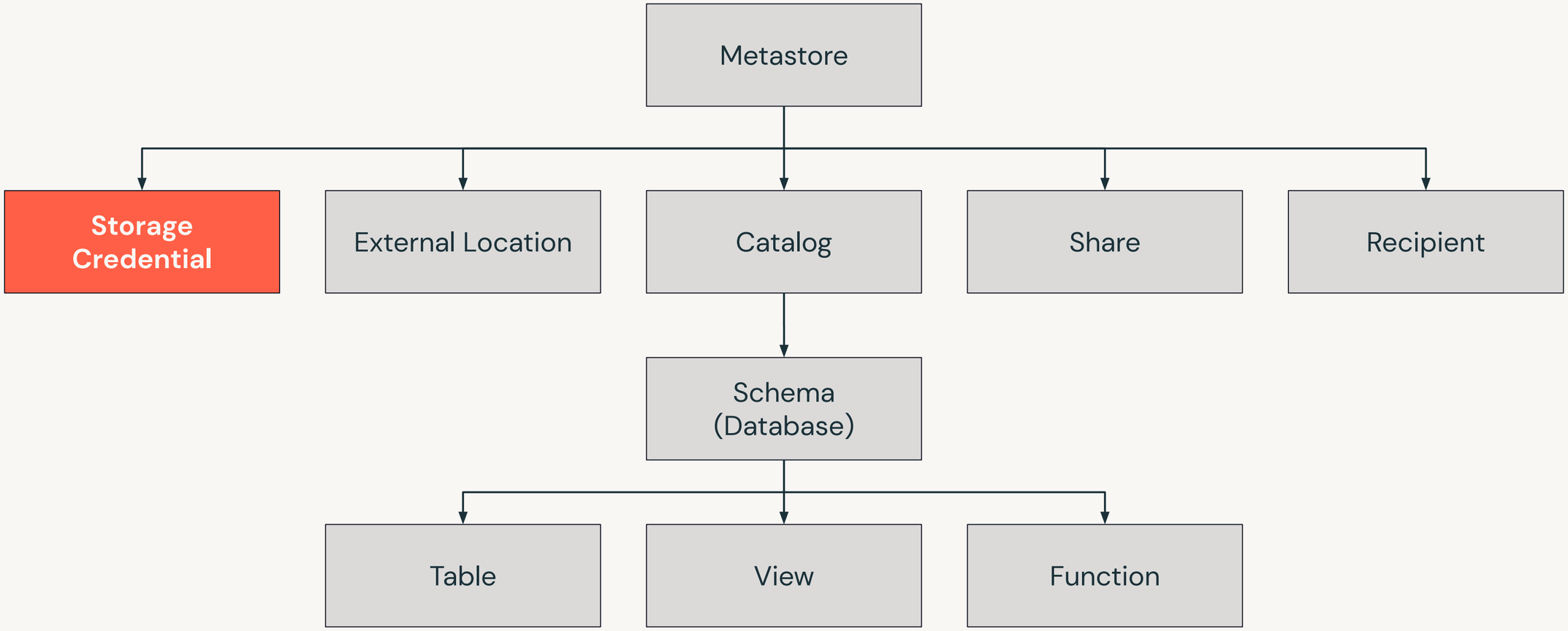
Key Concepts



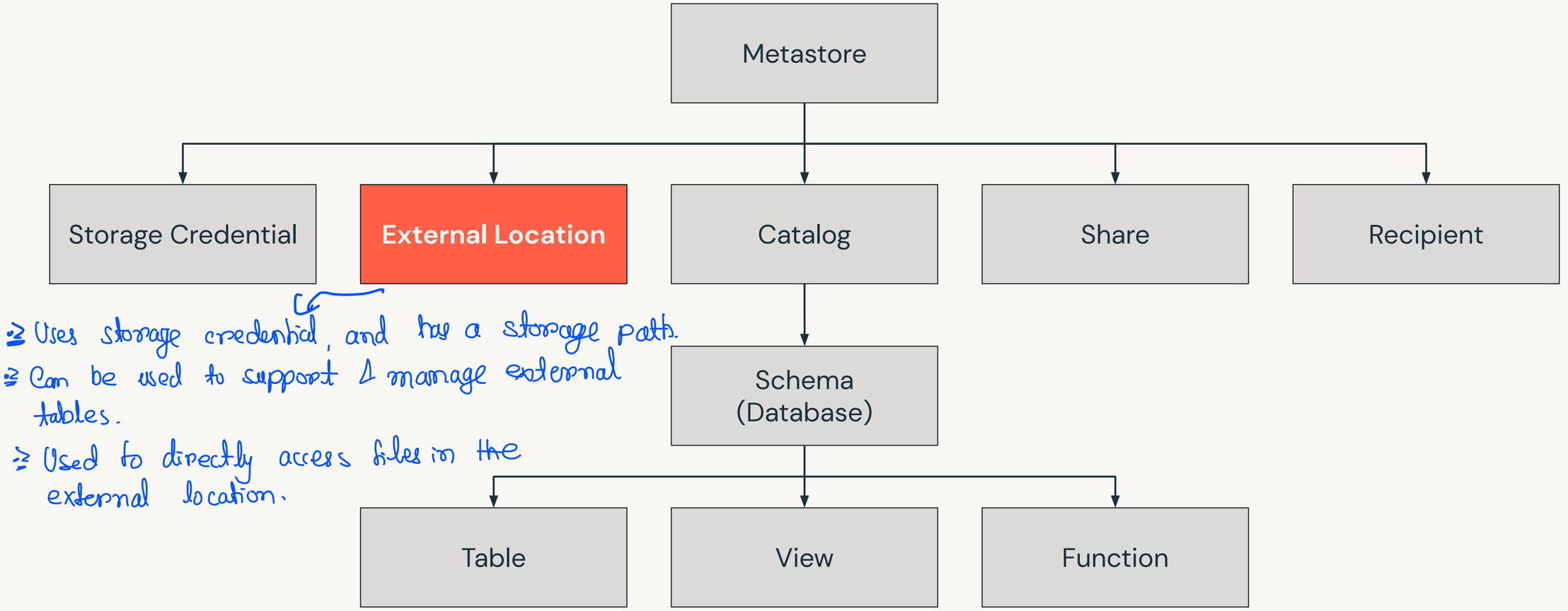
Key Concepts



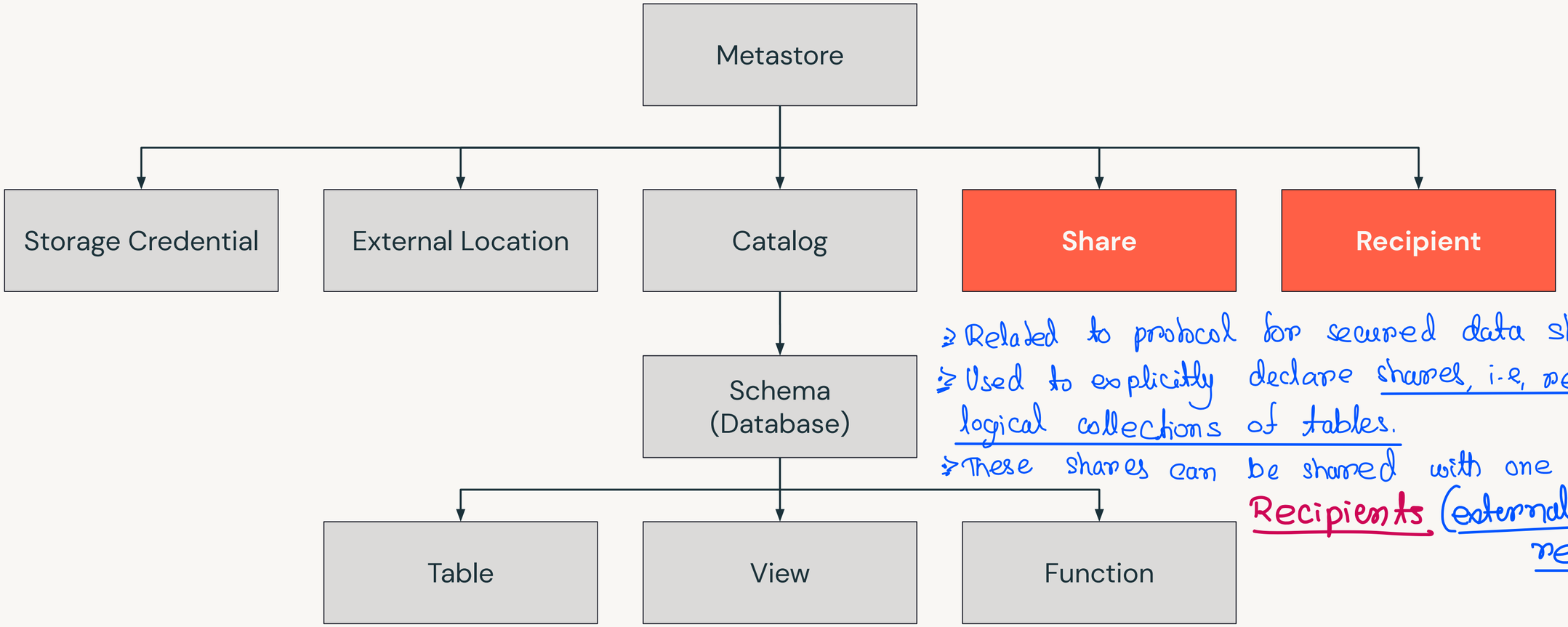
Key Concepts



Key Concepts



Key Concepts

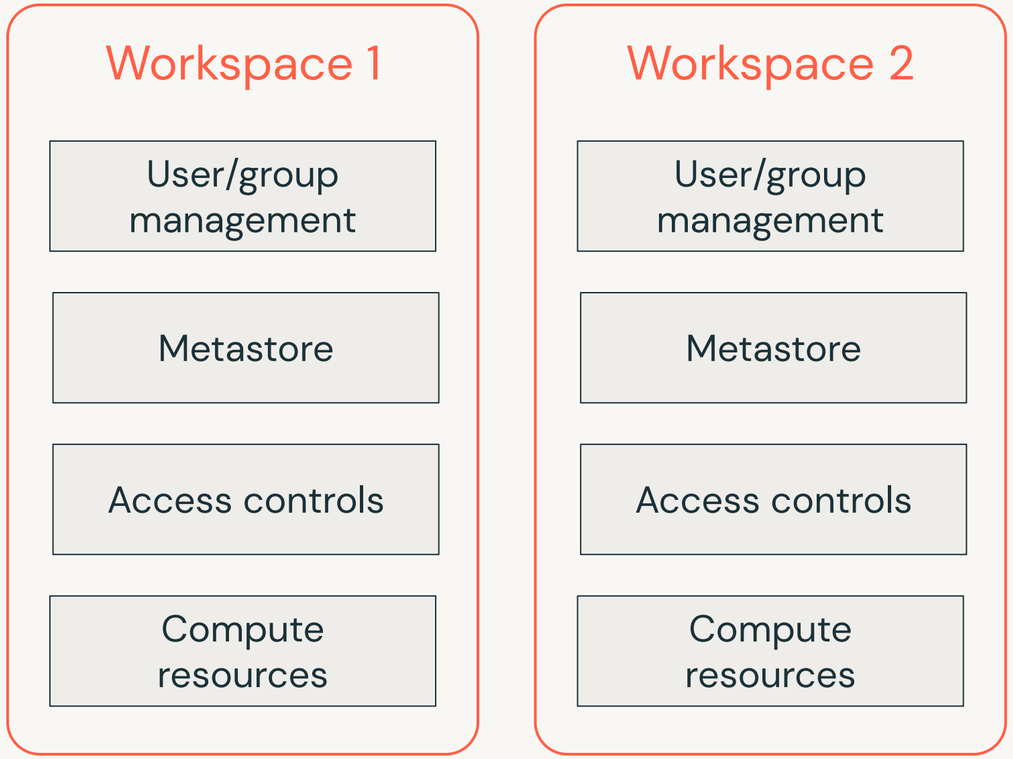


- Related to protocol for secured data sharing.
- Used to explicitly declare shares, i.e, read-only logical collections of tables.
- These shares can be shared with one or more Recipients (external data readers)

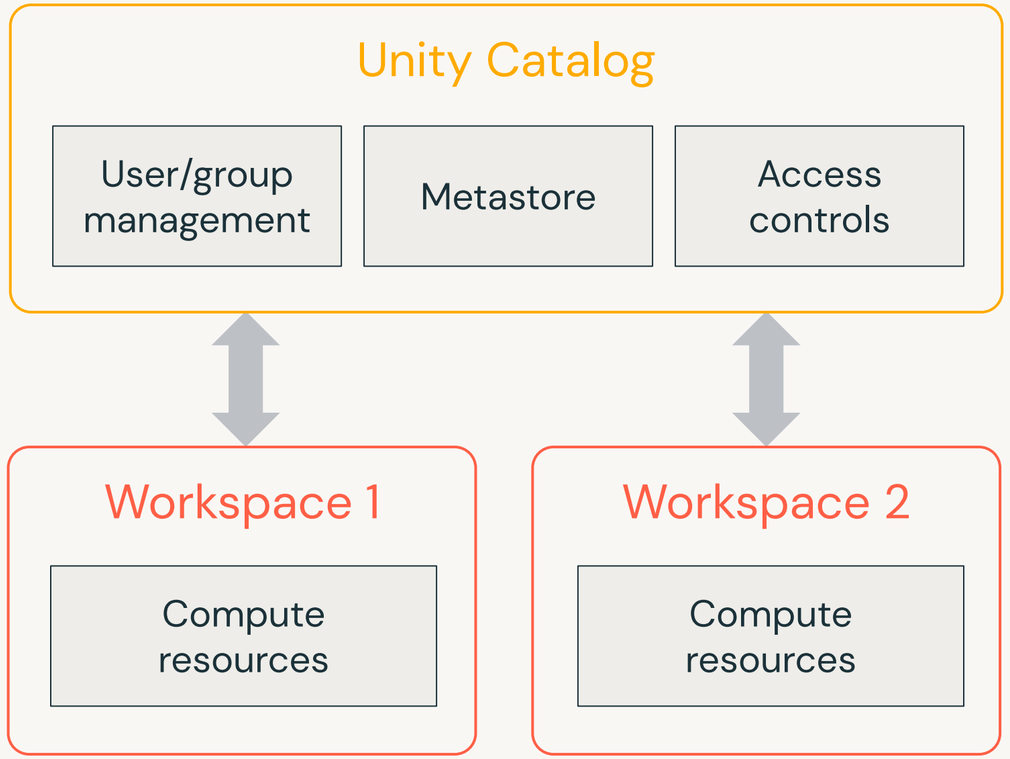
Unity Catalog Architecture

Architecture

Before Unity Catalog

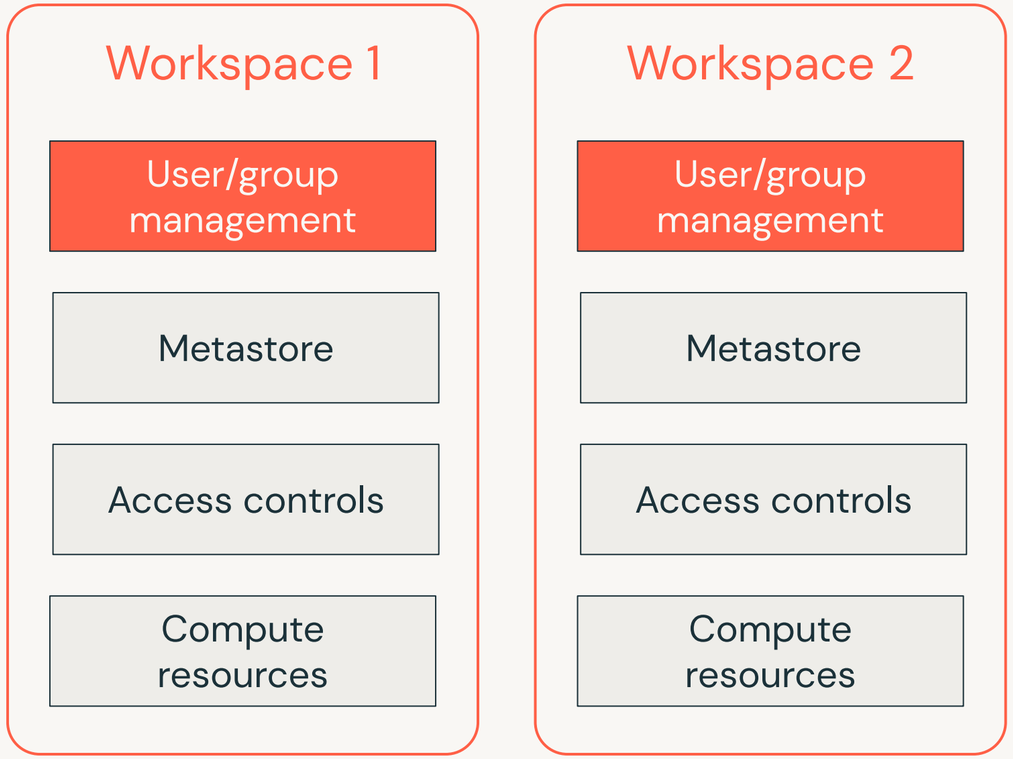


With Unity Catalog

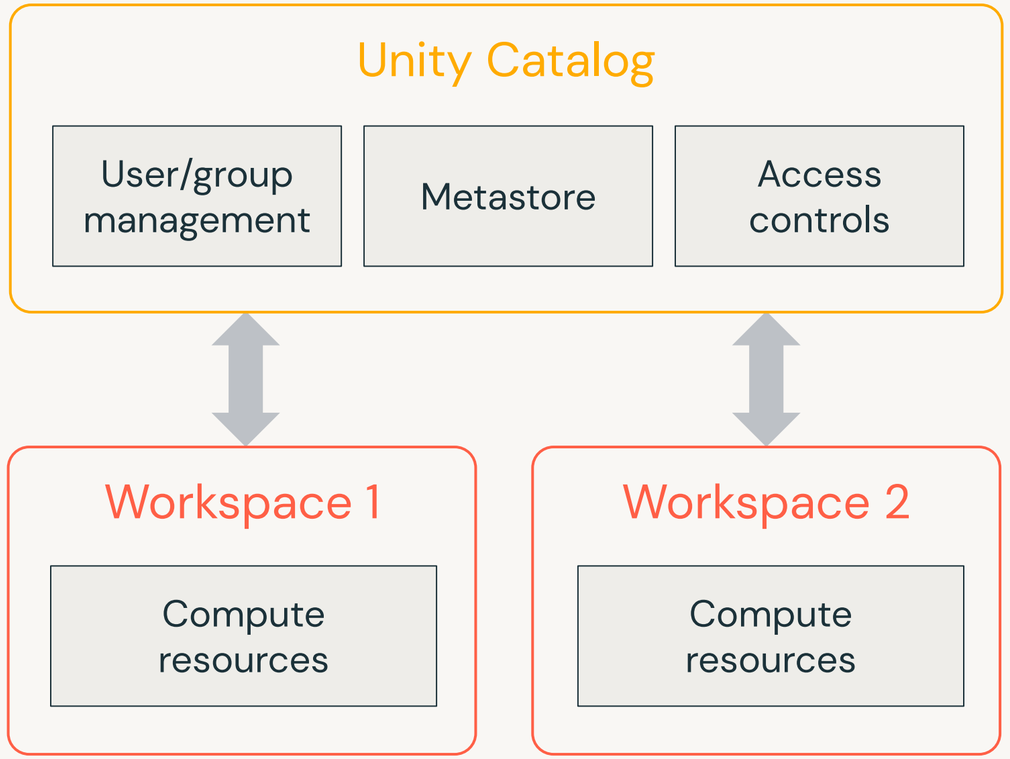


Architecture

Before Unity Catalog

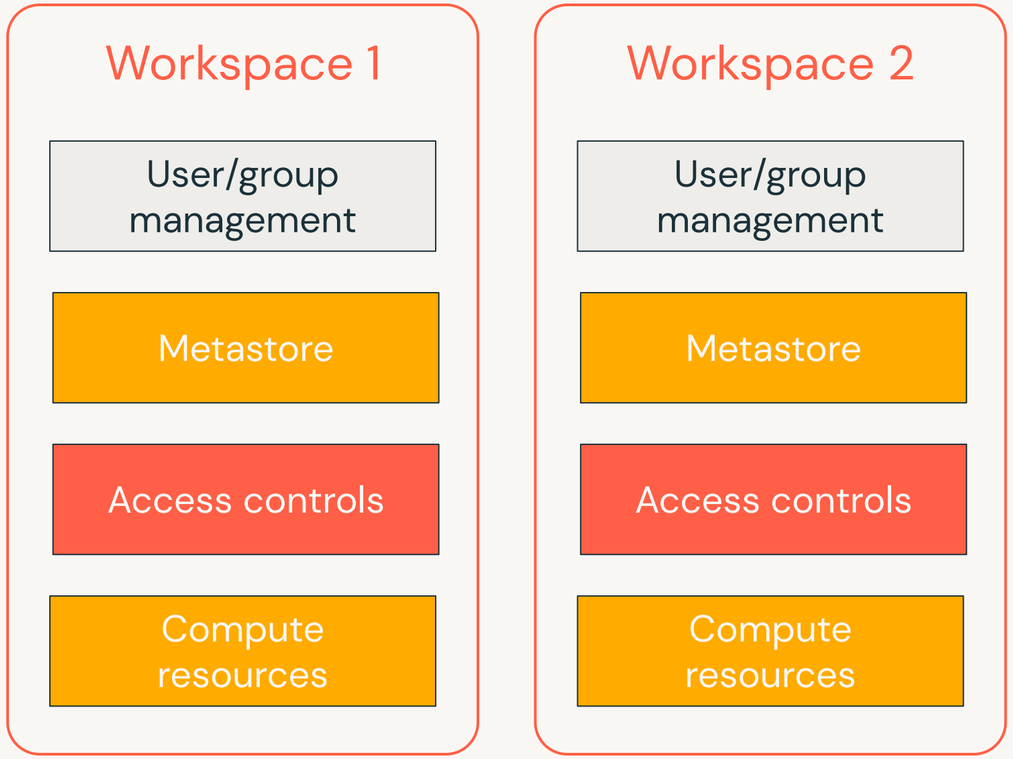


With Unity Catalog

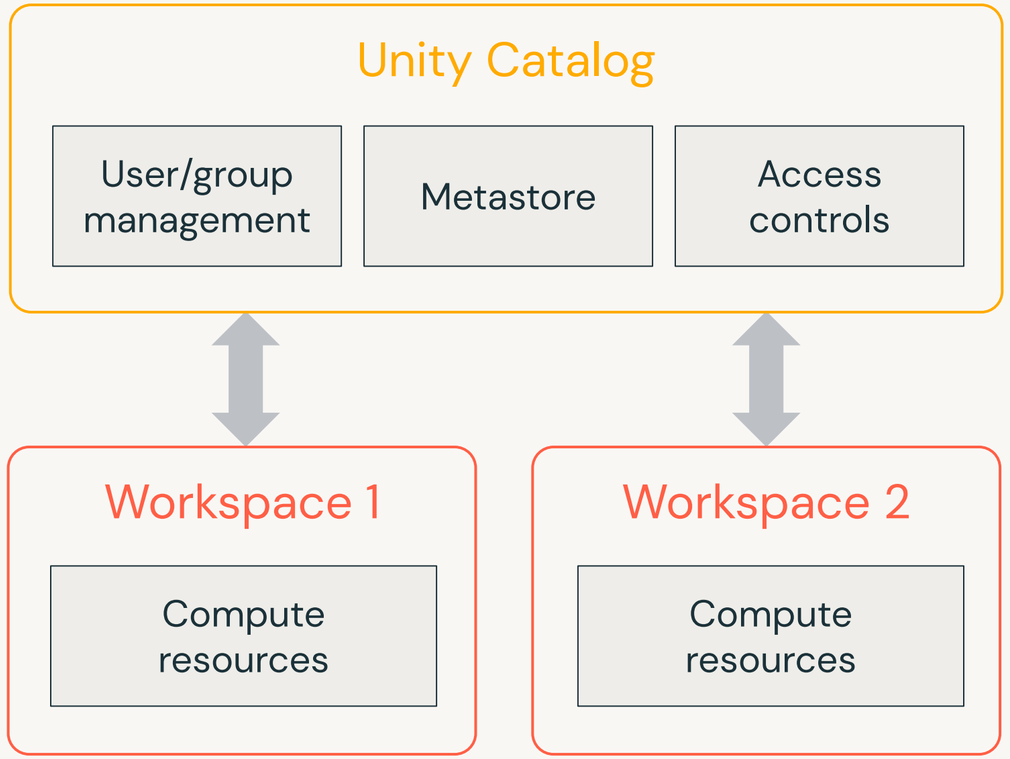


Architecture

Before Unity Catalog

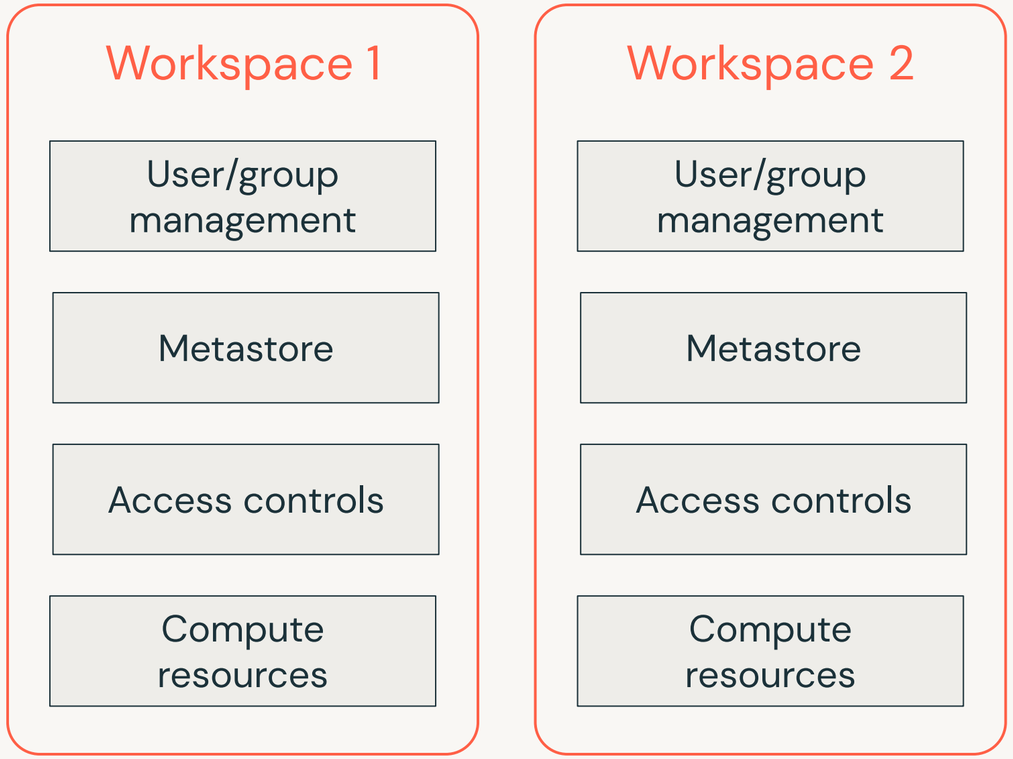


With Unity Catalog

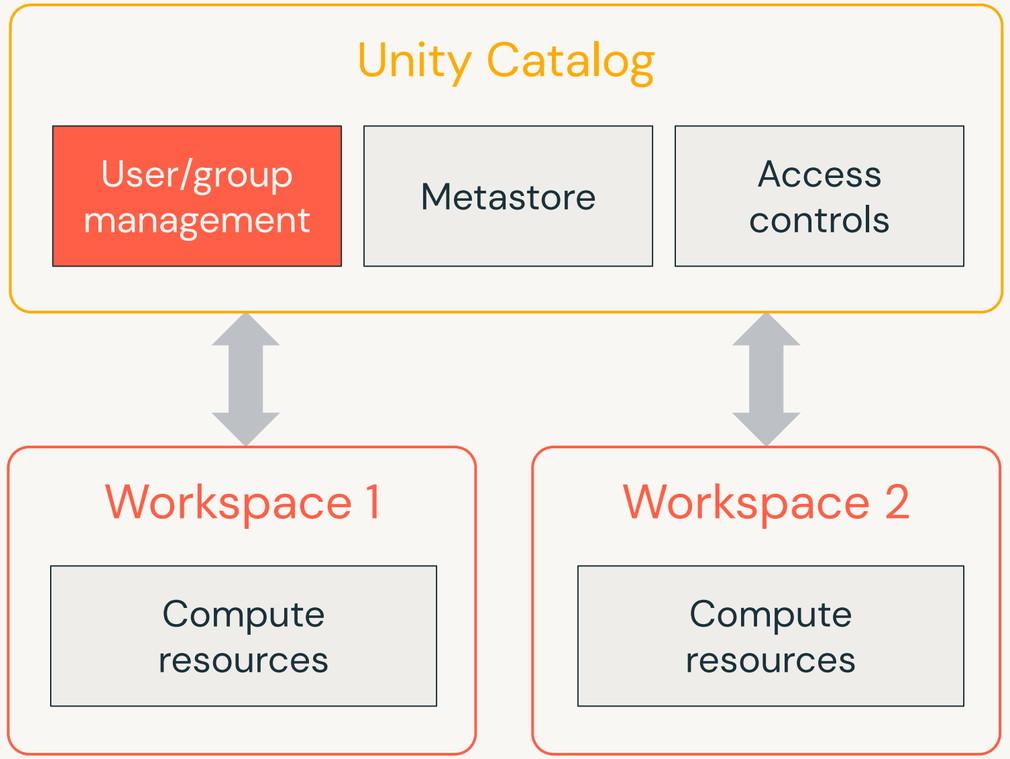


Architecture

Before Unity Catalog

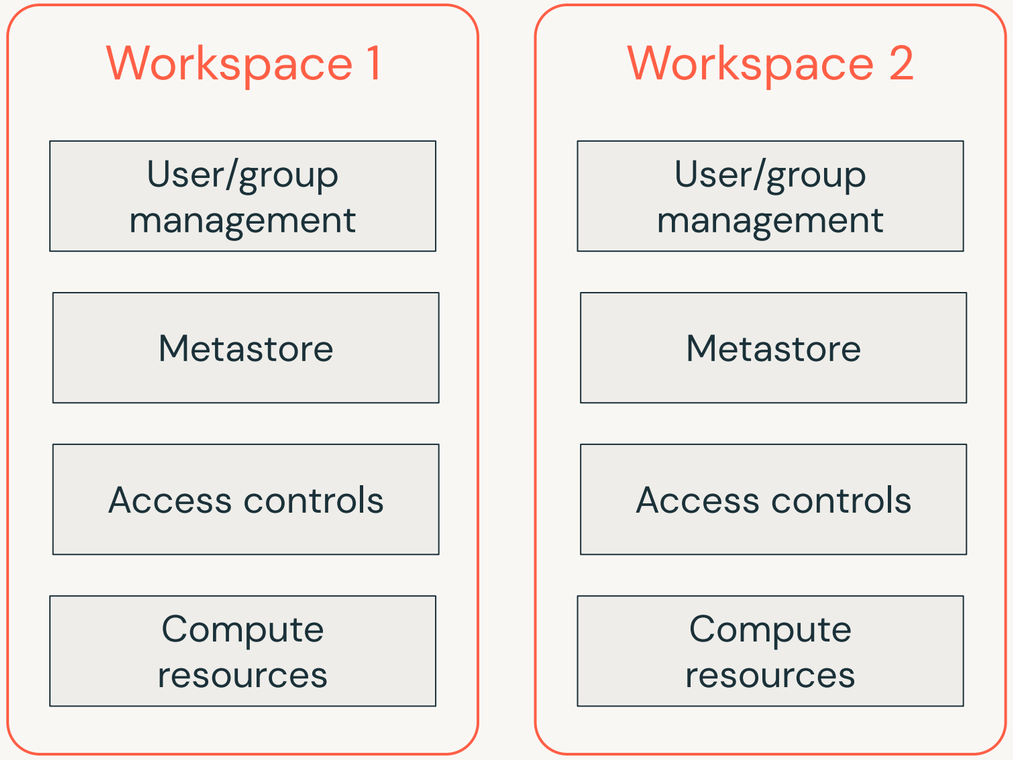


With Unity Catalog

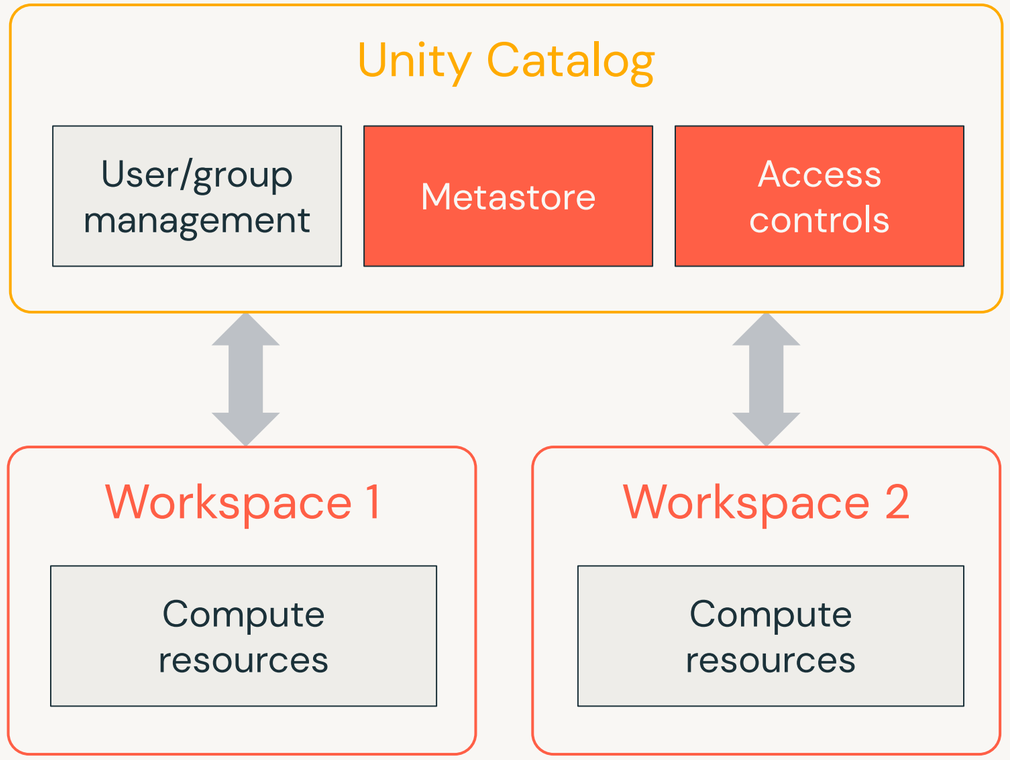


Architecture

Before Unity Catalog



With Unity Catalog



Unity Catalog Roles

Roles

Cloud Administrator

Identity Administrator

Account Administrator

Metastore Admin

Data Owner

Workspace Administrator

Administers underlying cloud resources

- Storage accounts/buckets
- IAM role/service principals/Managed Identities

Roles

Cloud Administrator

Identity Administrator

Account Administrator

Metastore Administrator

Data Owner

Workspace Administrator

Administers underlying identity provider service (when in use)

- Identity provider provisions users and groups into the account
- Avoids need to manually create and manage identities

Roles

- Cloud Administrator
- Identity Administrator
- **Account Administrator**
- Metastore Administrator
- Data Owner
- Workspace Administrator

Administers the account

- Creates, deletes, and assigns metastores to workspaces
- Creates, deletes, and assigns users and groups to workspaces
- Integrates account with an identity provider
- Full access to all data objects

Roles

- Cloud Administrator
- Identity Administrator
- Account Administrator
- **Metastore Administrator**
- Data Owner
- Workspace Administrator

Administers a metastore

- Creates and drops catalogs and other data objects
- Grants privileges on data objects
- Changes ownerships of data objects
- Designated by an account administrator

Roles

- Cloud Administrator
- Identity Administrator
- Account Administrator
- Metastore Administrator
- **Data Owner**
- Workspace Administrator

Owns data objects they created

- Creates nested objects
- Grants privileges to others on owned objects
- Changes ownership of owned objects

Roles

- Cloud Administrator
- Identity Administrator
- Account Administrator
- Metastore Administrator
- Data Owner
- **Workspace Administrator**

Administers a workspace

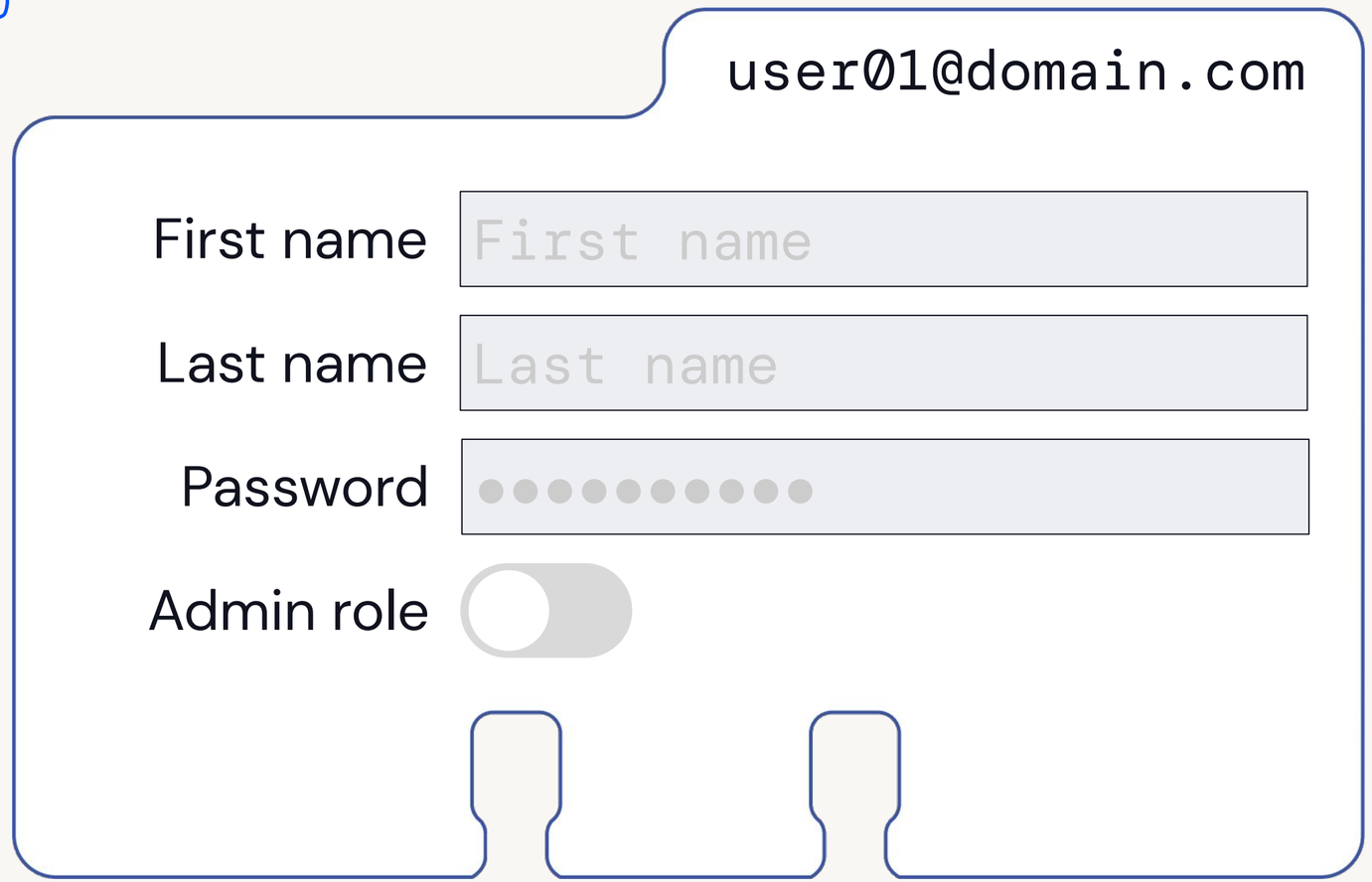
- Manages permissions on workspace assets
- Restricts access to cluster creation
- Adds or removes users
- Elevates users permissions
- Grant privileges to others
- Change job ownership

Unity Catalog Identities

Identities

User

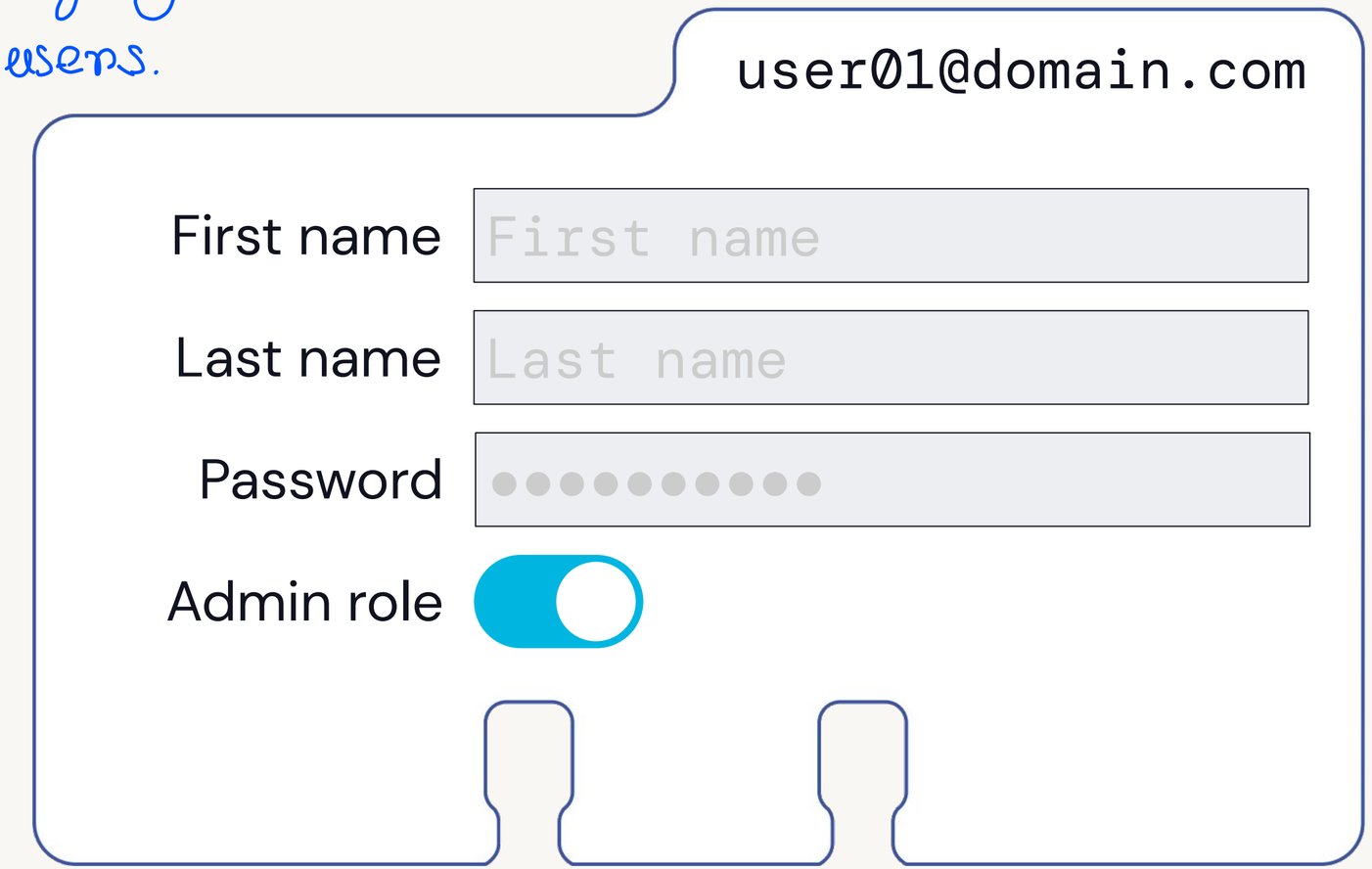
can access functionality through command line tools & REST APIs.



Identities

Account administrator

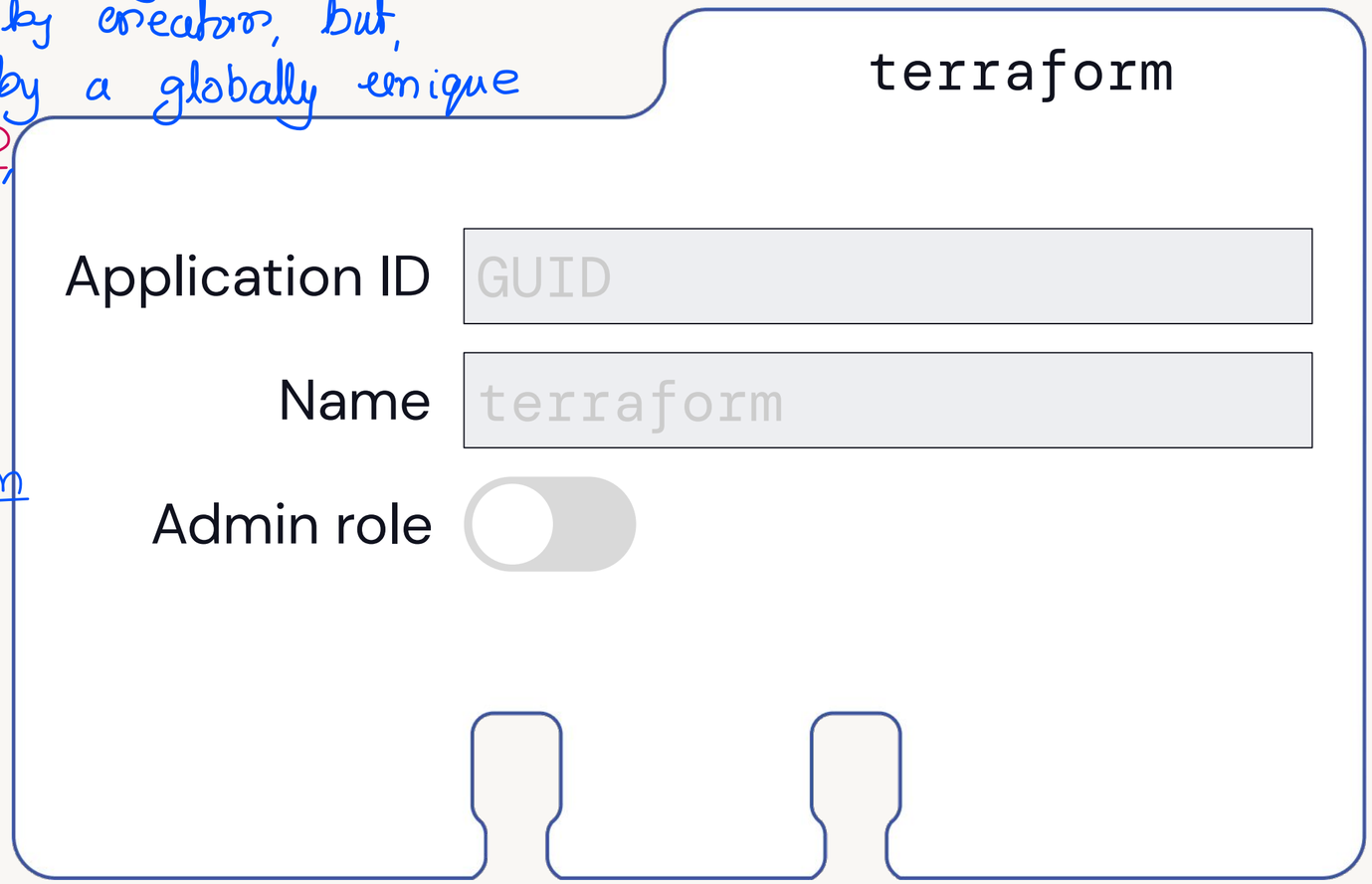
- Managing and assigning metastores to workspaces.
- Managing other users.



Identities

Service Principal

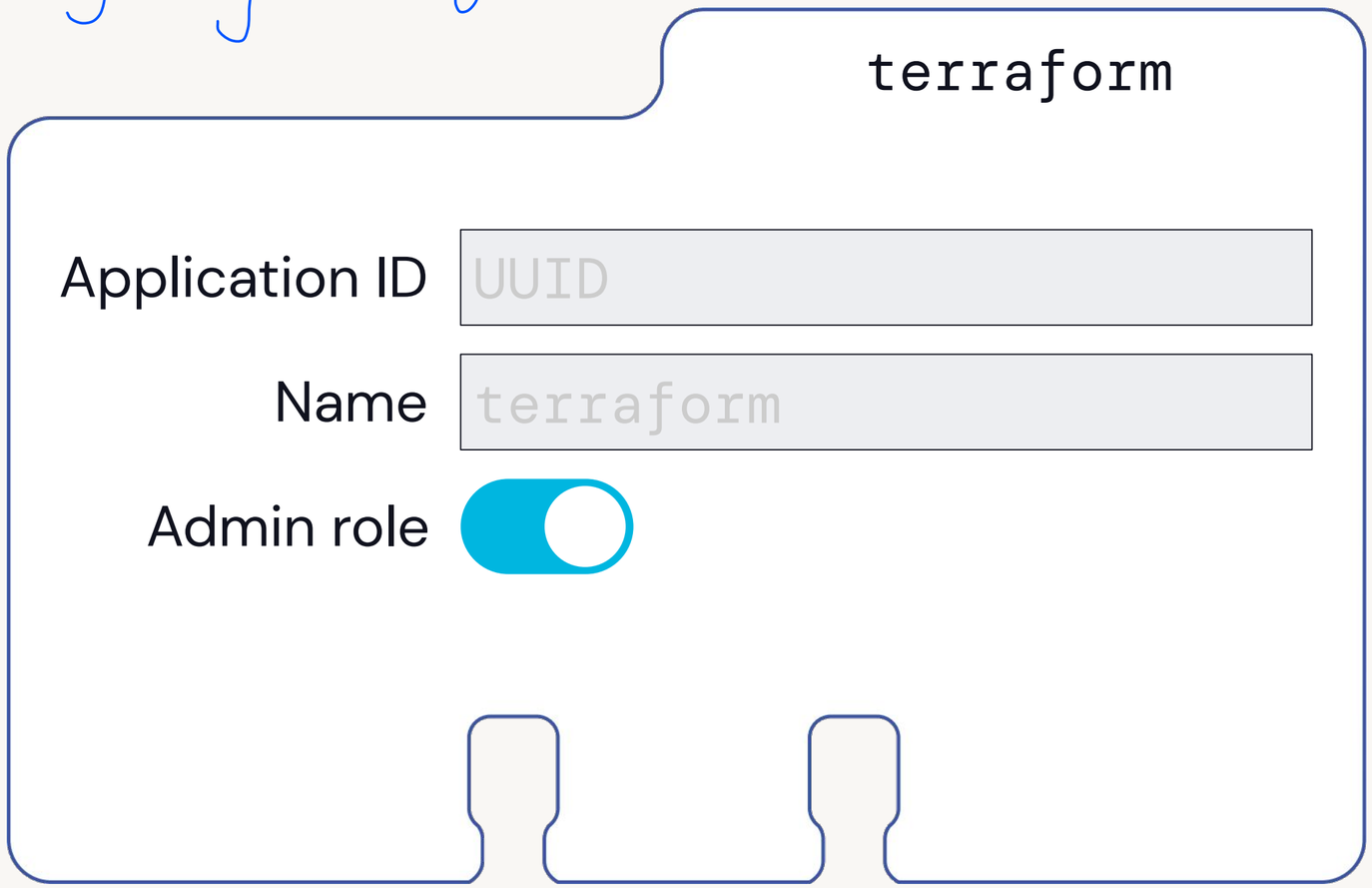
- it is an individual identity for use with automated tools running jobs, and applications
- Assigned a name by creator, but,
- Uniquely identified by a globally unique identifier, called GUID, assigned by the platform when the service principal is created.
- Authenticate with the platform using Access Token
- They access functionality through APIs, or it can run workloads using Databricks.



Identities

Service Principal with administrative privileges

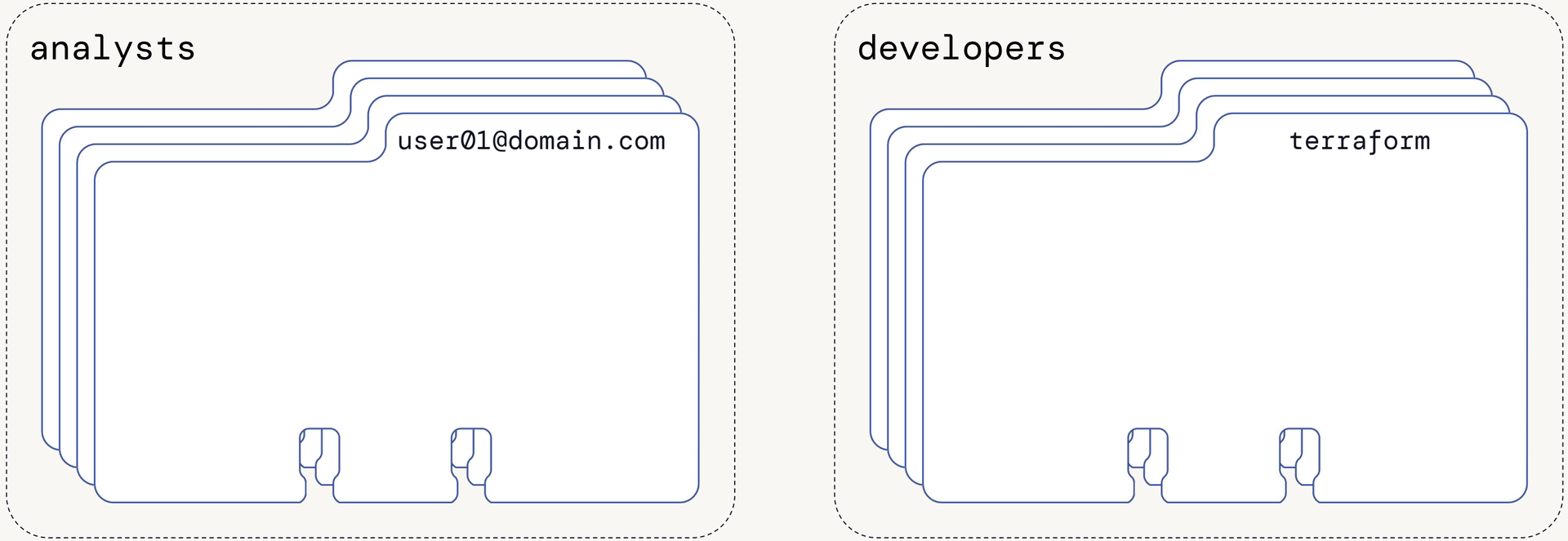
⇒ Allow them to programatically carry out any of the account management tasks that users with administrative privilege can perform.



Identities

Groups

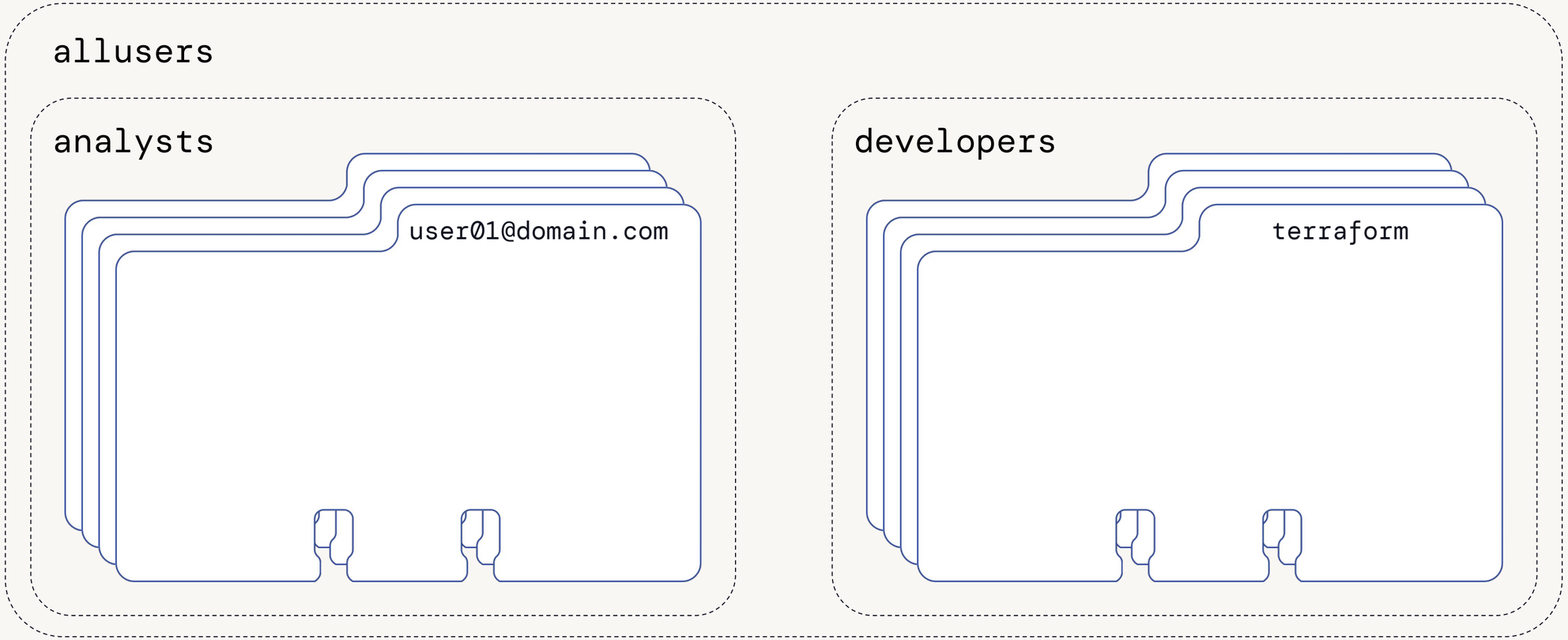
=> Gathers individual users & service principals into units, to simplify management.



Identities

Nesting groups

- Child groups inherits all the grants provided to the parent group.

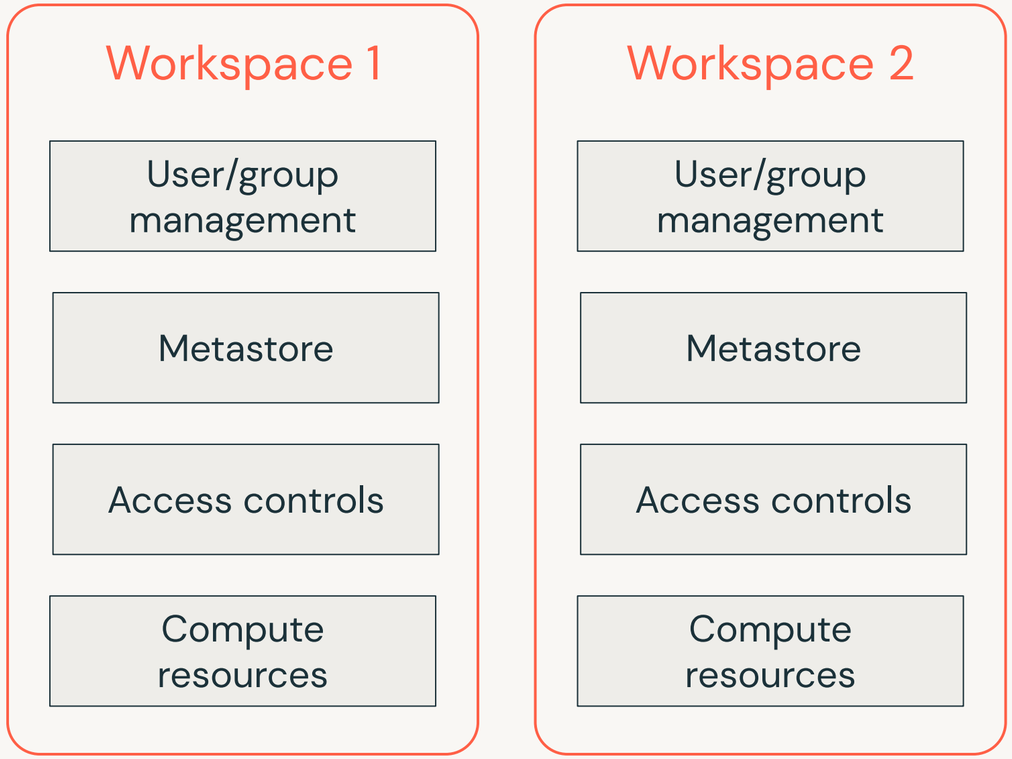


Identities

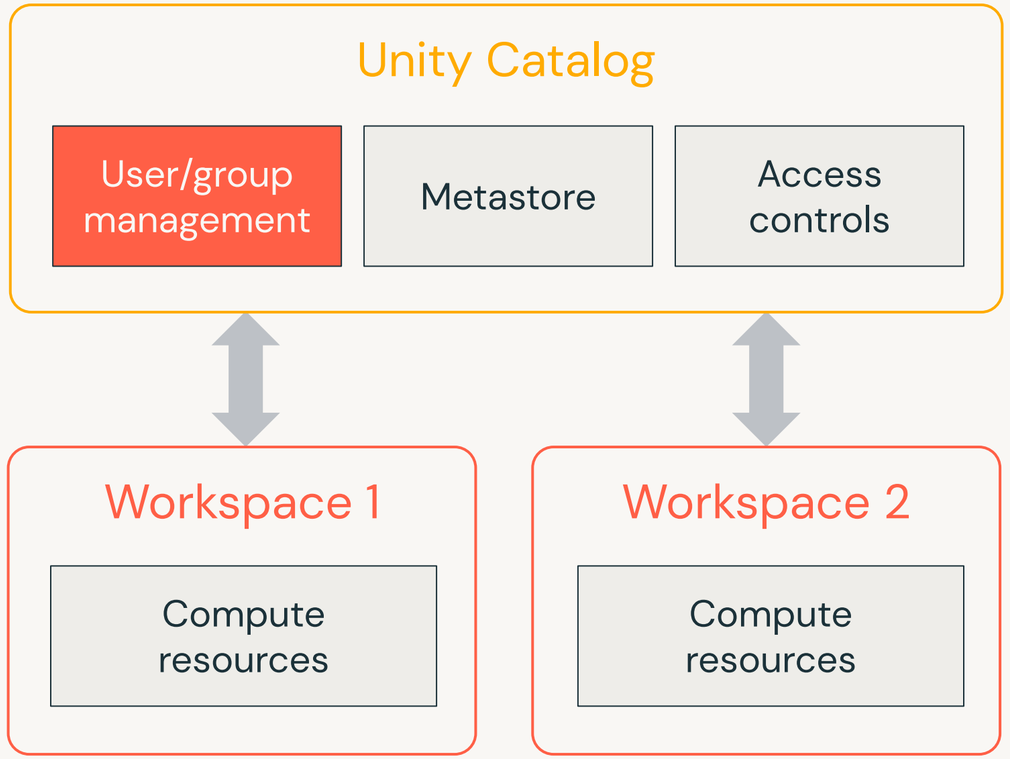
Recap

=> Identities still exist distinctly in the workspaces.

Before Unity Catalog



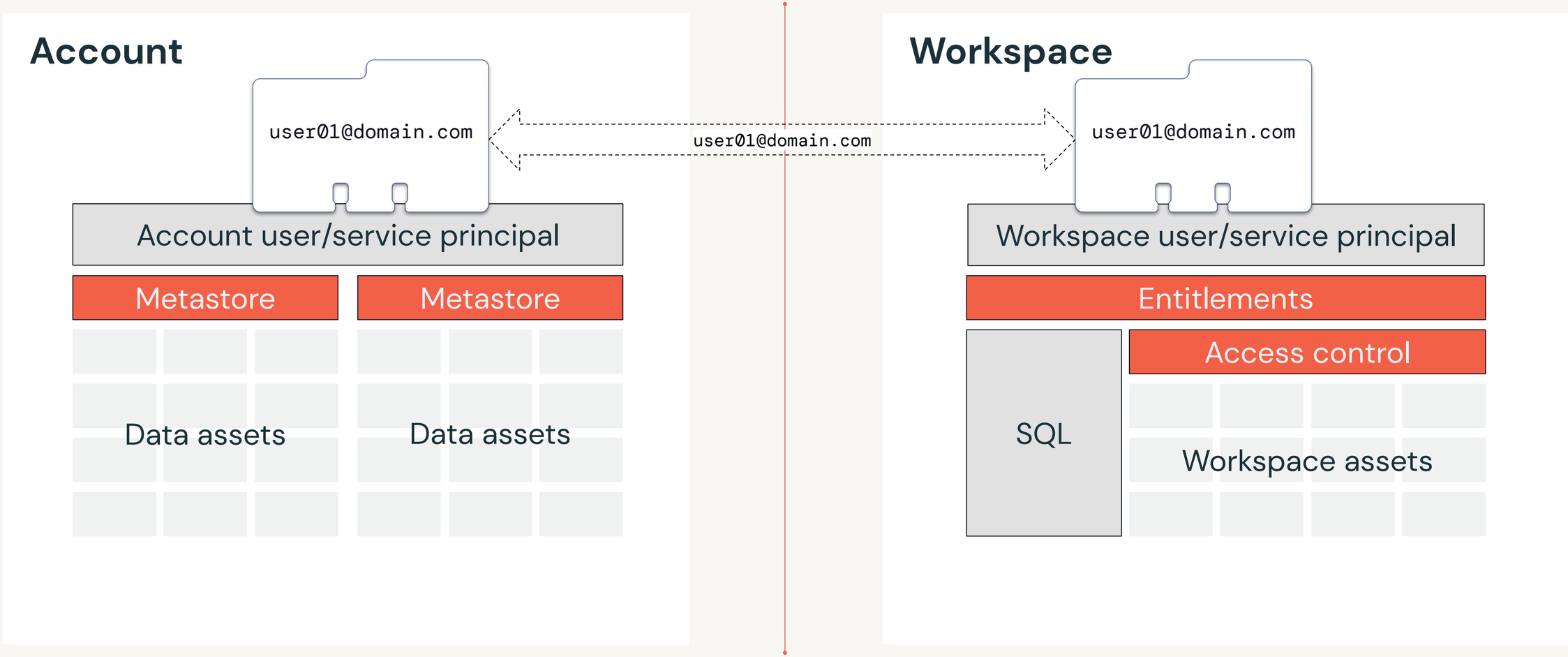
With Unity Catalog



Identities

⇒ Though account console & workspace identities are distinct, they are linked by the identifiable info. common to the two. [email / GUID]
⇒ For this reason, emails of users should match for both the accounts.

Relating account and workspace identities

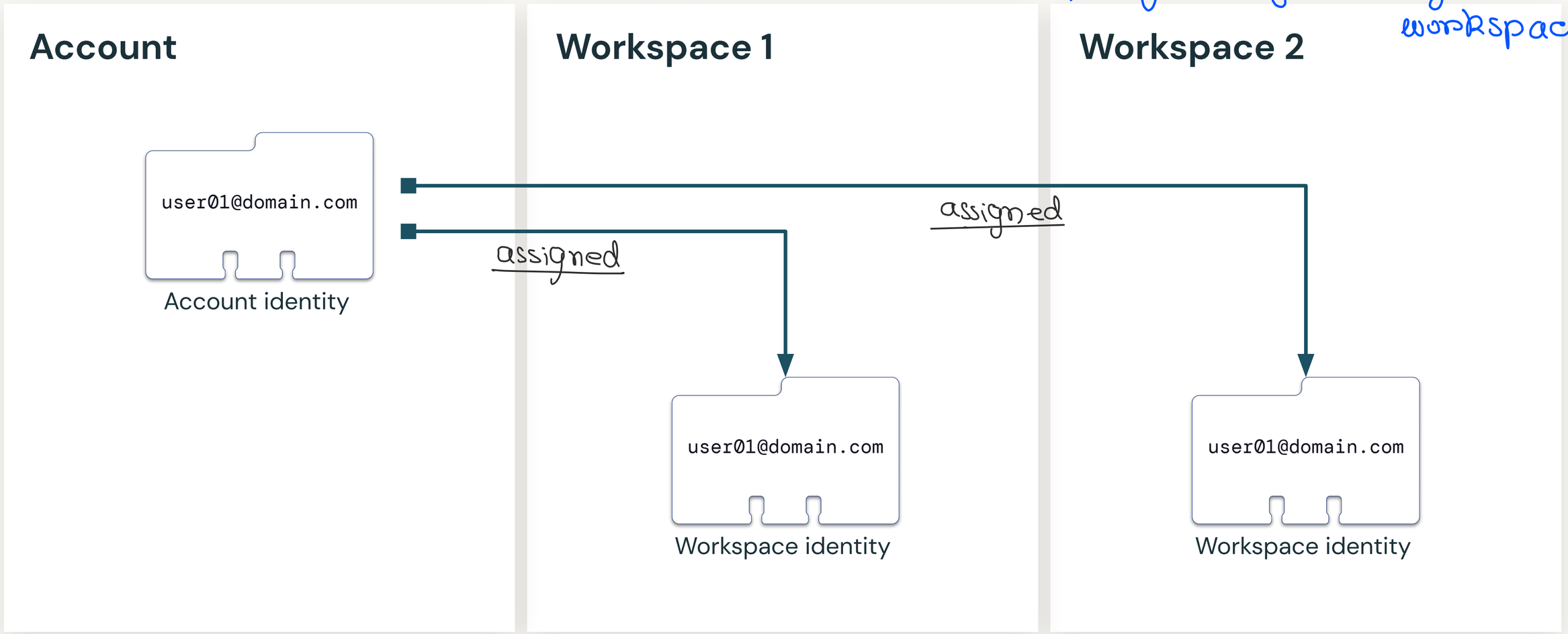


Identities

Identity federation

⇒ Eliminates the needs to manually create and maintain copies of identities at the workspace-level.

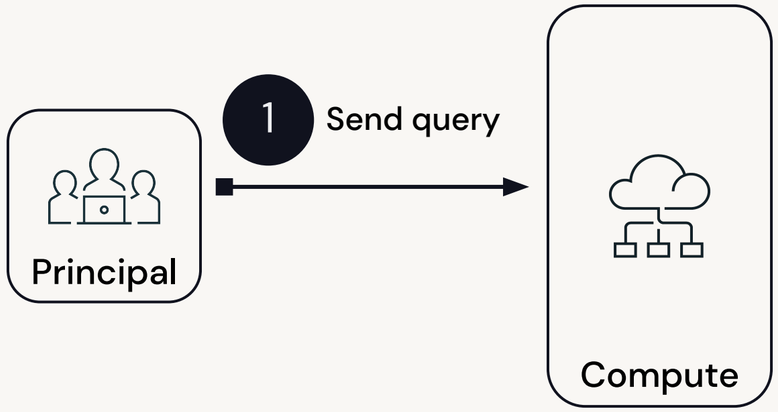
⇒ Once created in account console, they are just assigned to workspaces.



Unity Catalog Security Model

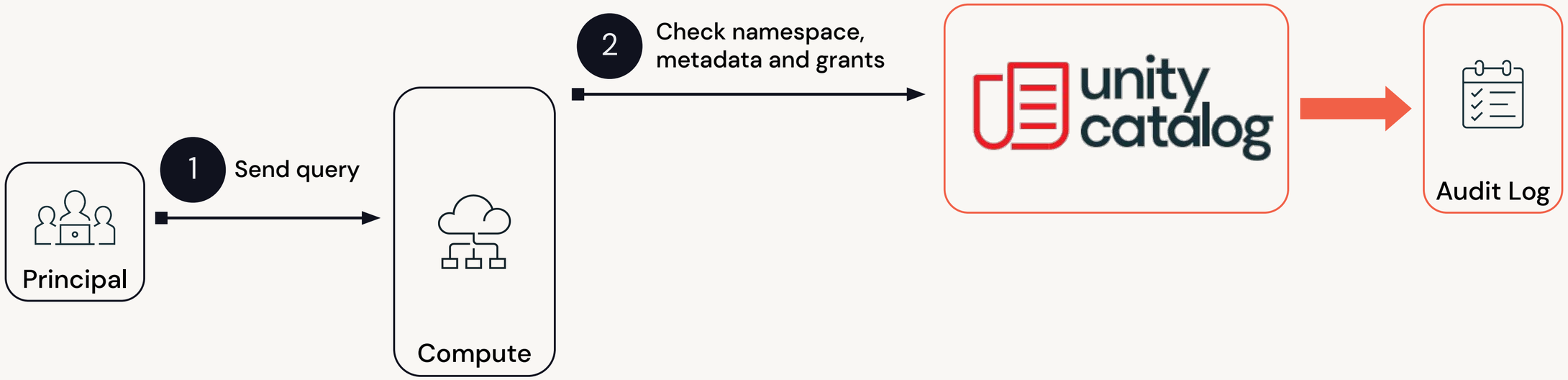
Security Model

Query Lifecycle



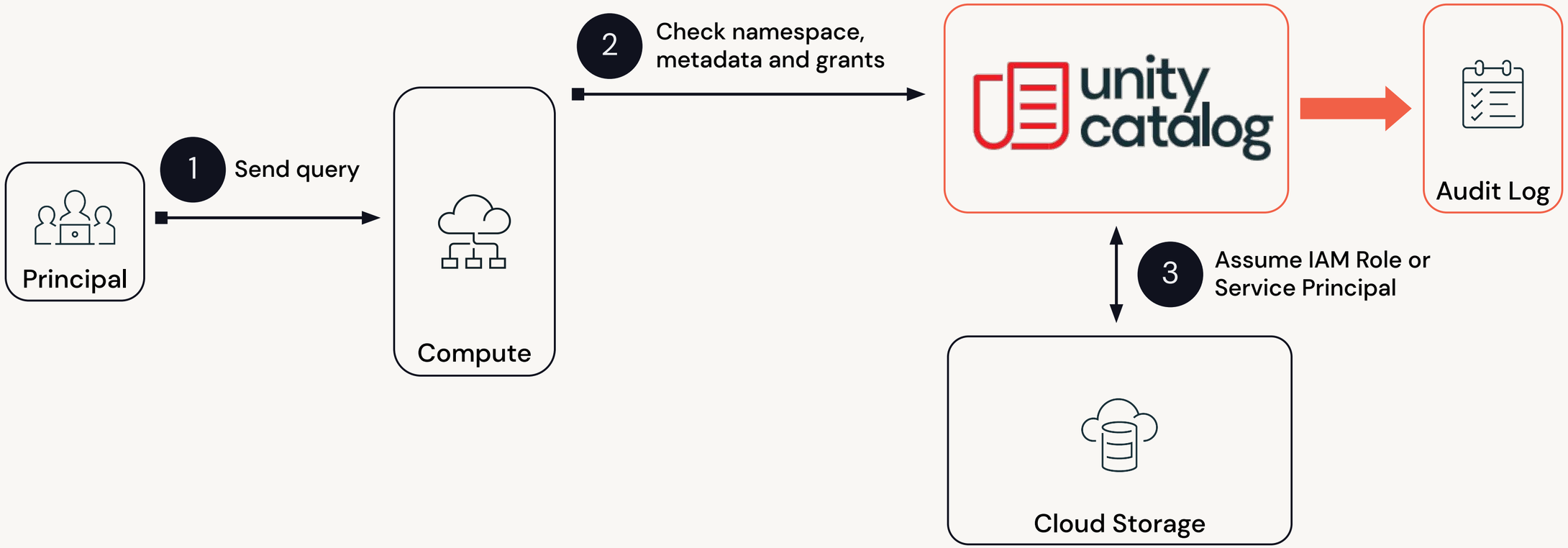
Security Model

Query Lifecycle



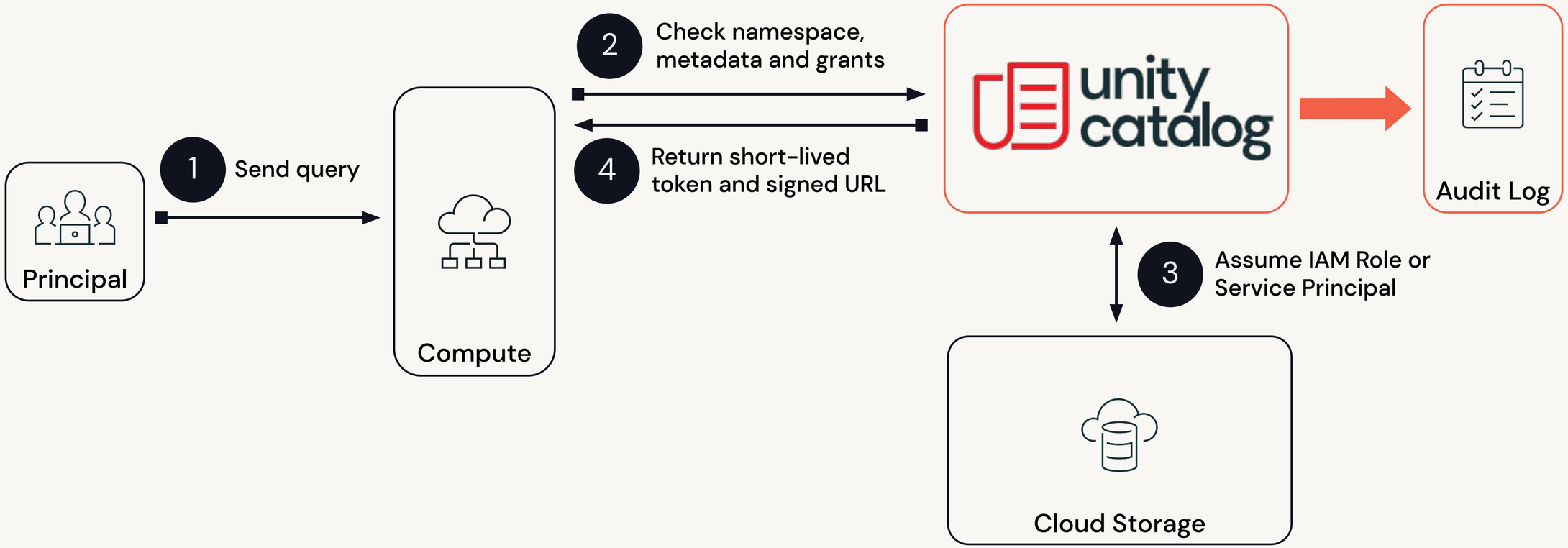
Security Model

Query Lifecycle



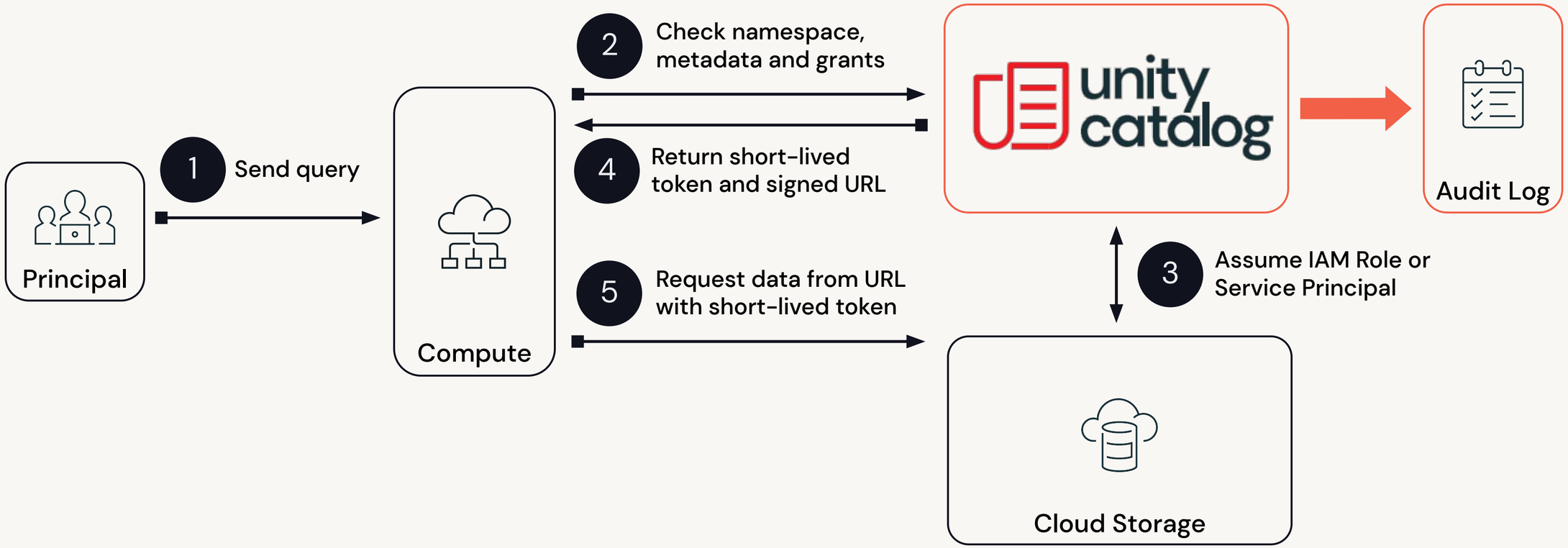
Security Model

Query Lifecycle



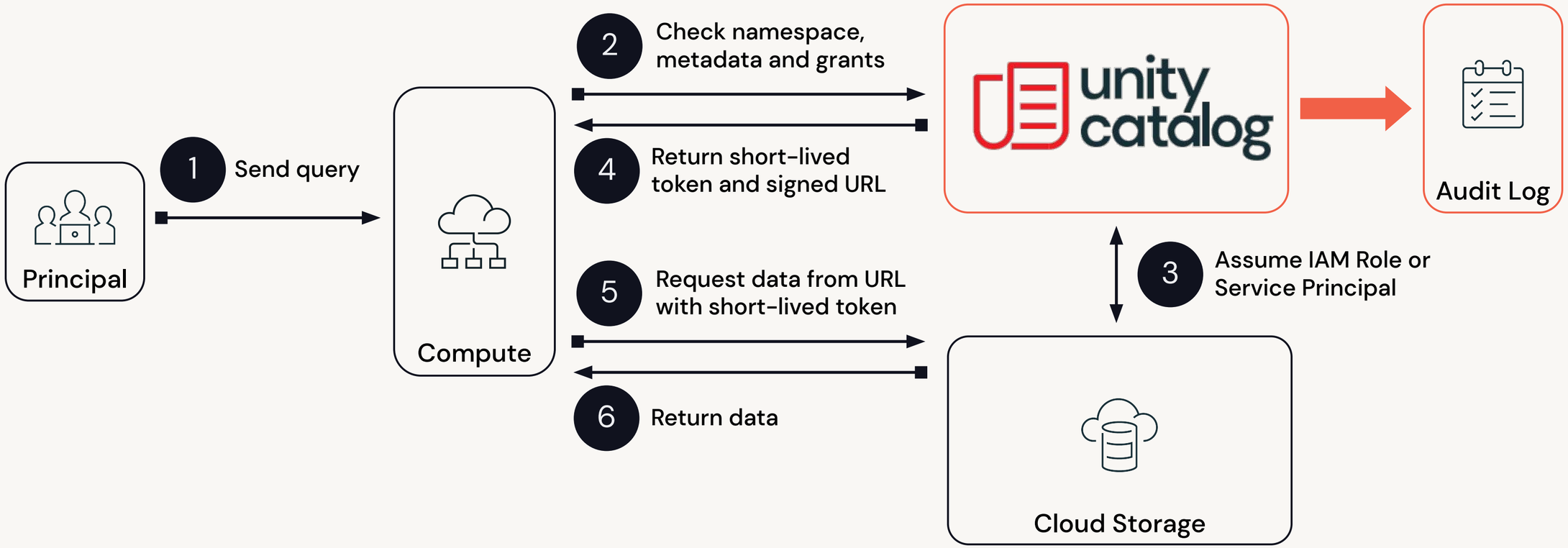
Security Model

Query Lifecycle



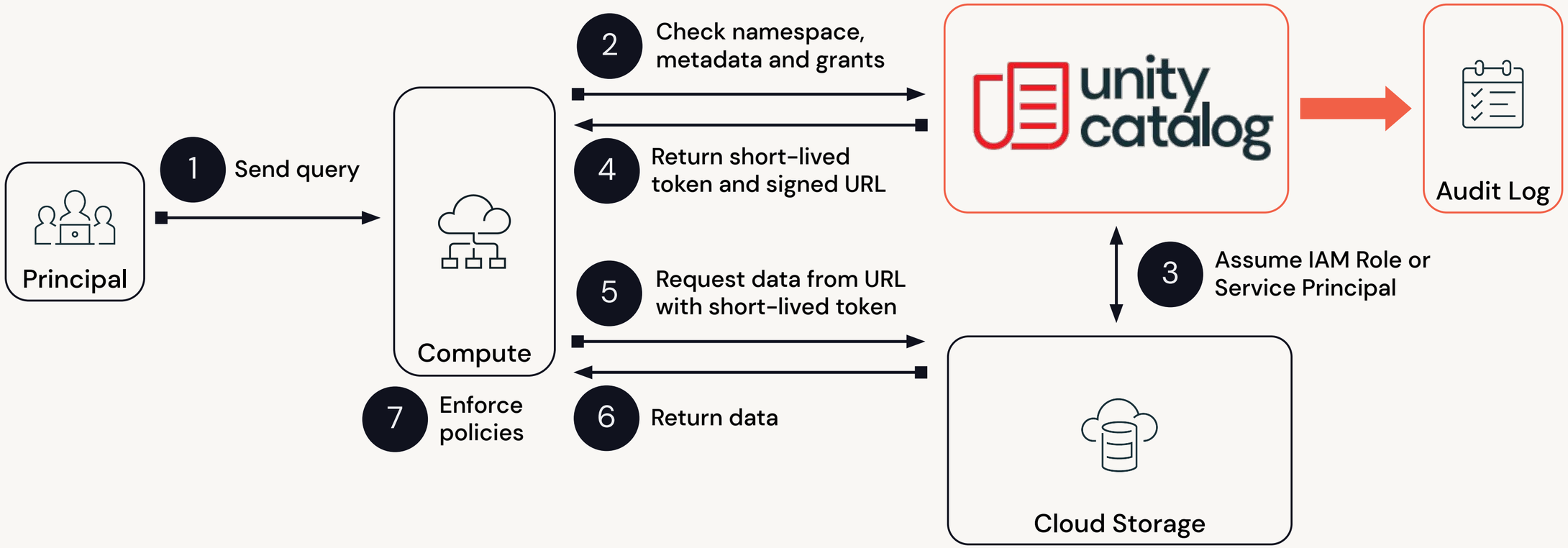
Security Model

Query Lifecycle



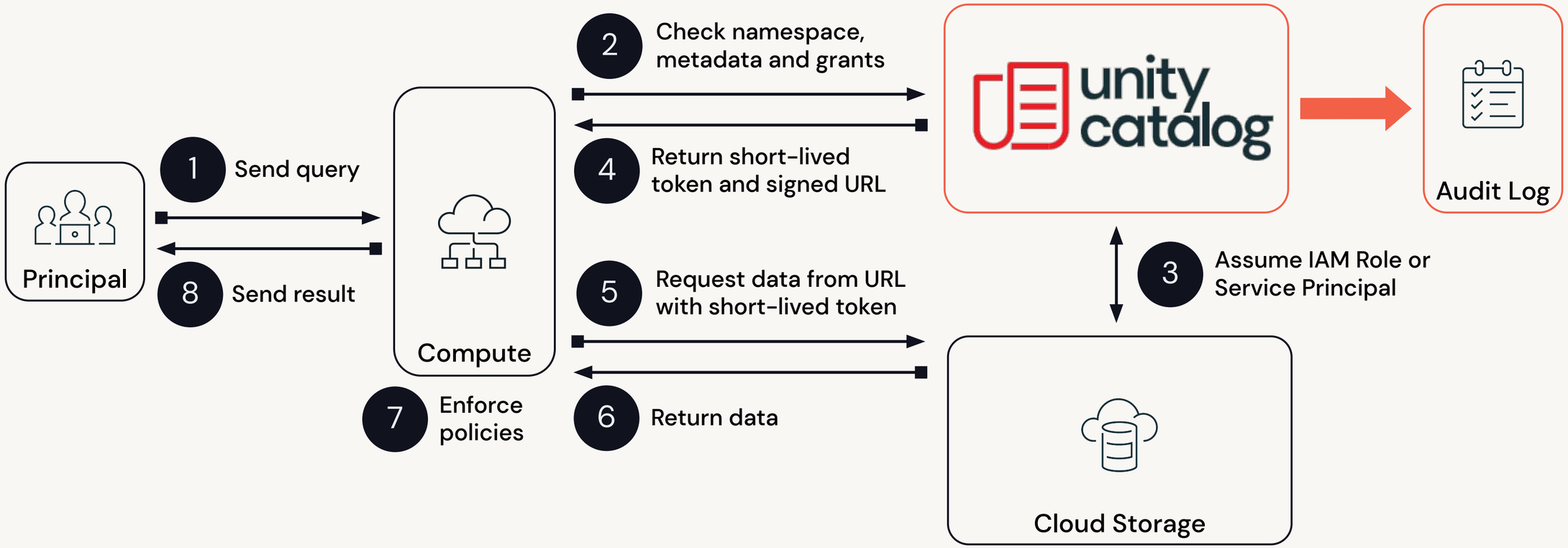
Security Model

Query Lifecycle



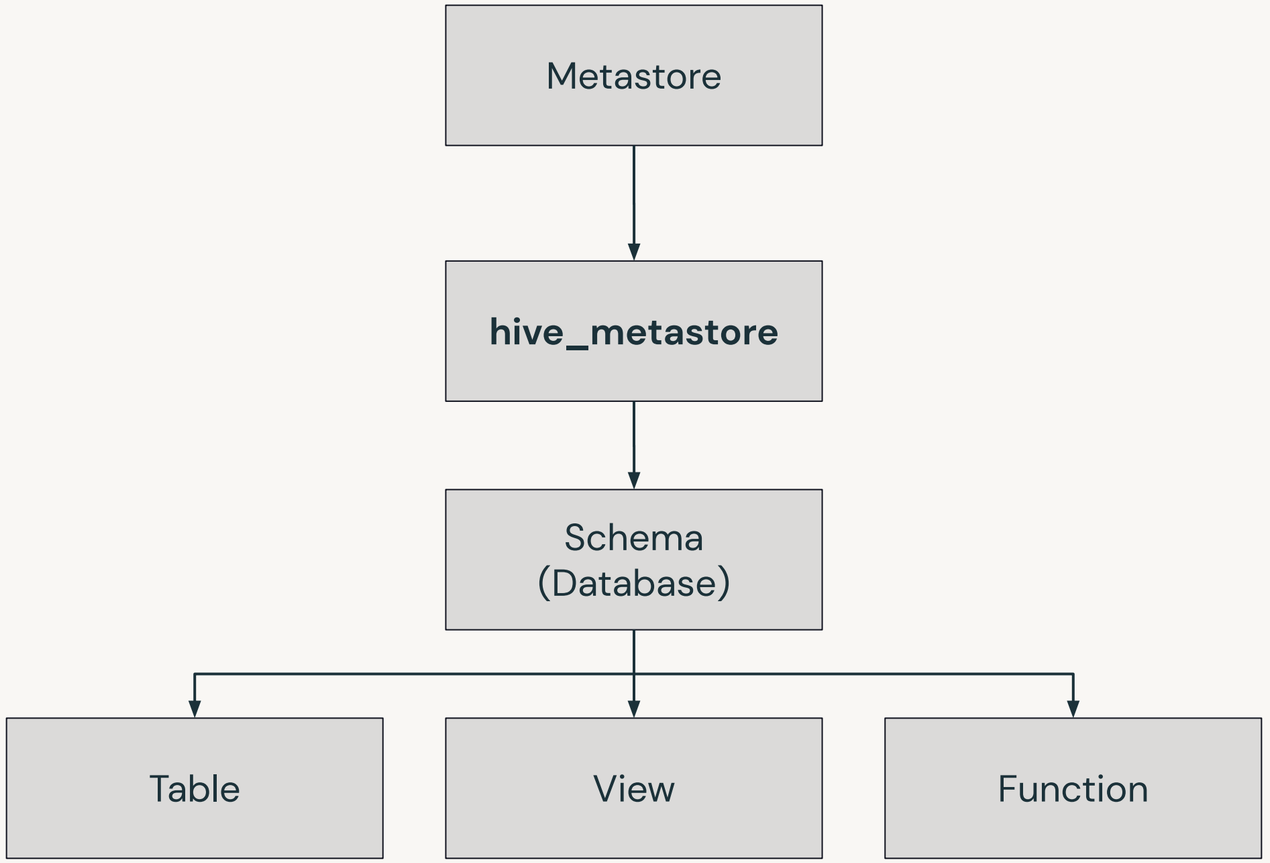
Security Model

Query Lifecycle



Security Model

Accessing legacy metastore



databricks

Compute Resources and Unity Catalog

Course Objectives

By the end of this course, you will be able to:

1. Describe how to access Unity Catalog through Databricks compute resources
2. Create a Unity Catalog enabled cluster
3. Access Unity Catalog through Databricks SQL

Course Objectives

By the end of this course, you will be able to:

1. Describe how to access Unity Catalog through Databricks compute resources
2. Create a Unity Catalog enabled cluster
3. Access Unity Catalog through Databricks SQL

Course Objectives

By the end of this course, you will be able to:

1. Describe how to access Unity Catalog through Databricks compute resources
2. Create a Unity Catalog enabled cluster
3. Access Unity Catalog through Databricks SQL

Cluster Security Mode

Clusters

Security modes

Modes supporting Unity Catalog

Single user
Multiple language support, not shareable

User isolation
Shareable, Python and SQL, legacy table ACLs

Modes not supporting Unity Catalog

None
No security

Table ACL only
Legacy table ACLs, multiple languages, shareable

Passthrough only
Credential passthrough, multiple languages, shareable

Clusters

Security modes

Modes supporting Unity Catalog

Single user
Multiple language support, not shareable

User isolation
Shareable, Python and SQL, legacy table ACLs

Modes not supporting Unity Catalog

None
No security

Table ACL only
Legacy table ACLs, multiple languages, shareable

Passthrough only
Credential passthrough, multiple languages, shareable

Clusters

Security modes

Modes supporting Unity Catalog

Single user
Multiple language support, not shareable

User isolation
Shareable, Python and SQL, legacy table ACLs

Modes not supporting Unity Catalog

None
No security

Table ACL only
Legacy table ACLs, multiple languages, shareable

Passthrough only
Credential passthrough, multiple languages, shareable

Clusters

Security modes

Modes supporting Unity Catalog

Single user
Multiple language support, not shareable

User isolation
Shareable, Python and SQL, legacy table ACLs

Modes not supporting Unity Catalog

None
No security

Table ACL only
Legacy table ACLs, multiple languages, shareable

Passthrough only
Credential passthrough, multiple languages, shareable

Clusters

Security mode feature matrix

Security mode	Supported languages	Legacy table ACL	Credential passthrough	Shareable	RDD API	DBFS Fuse mounts	Init scripts and libraries	Dynamic views	Machine learning
None	All			●	●	●	●		●
Single user	**All**				●	●	●		●
User isolation	**SQL Python**	●		●				●	
Legacy table ACL	SQL Python	●		●			●		
Legacy Passthrough	SQL Python		●	●	●	●	●		

Clusters

Security mode feature matrix

Security mode	Supported languages	Legacy table ACL	Credential passthrough	Shareable	RDD API	DBFS Fuse mounts	Init scripts and libraries	Dynamic views	Machine learning
None	All			●	●	●	●		●
Single user	**All**				●	●	●		●
User isolation	**SQL Python**	●		●				●	
Legacy table ACL	SQL Python	●		●			●		
Legacy Passthrough	SQL Python		●	●	●	●	●		

Clusters

Security mode feature matrix

Security mode	Supported languages	Legacy table ACL	Credential passthrough	Shareable	RDD API	DBFS Fuse mounts	Init scripts and libraries	Dynamic views	Machine learning
None	All			●	●	●	●		●
Single user	**All**				●	●	●		●
User isolation	**SQL Python**	●		●				●	
Legacy table ACL	SQL Python	●		●			●		
Legacy Passthrough	SQL Python		●	●	●	●	●		

databricks

Data Access Control in Unity Catalog

Course Objectives

By the end of this course, you will be able to:

1. Describe the security model for governing data objects in Unity Catalog
2. Define data access rules and manage data ownership
3. Secure access to external storage

Course Objectives

By the end of this course, you will be able to:

1. Describe the security model for governing data objects in Unity Catalog
2. Define data access rules and manage data ownership
3. Secure access to external storage

Course Objectives

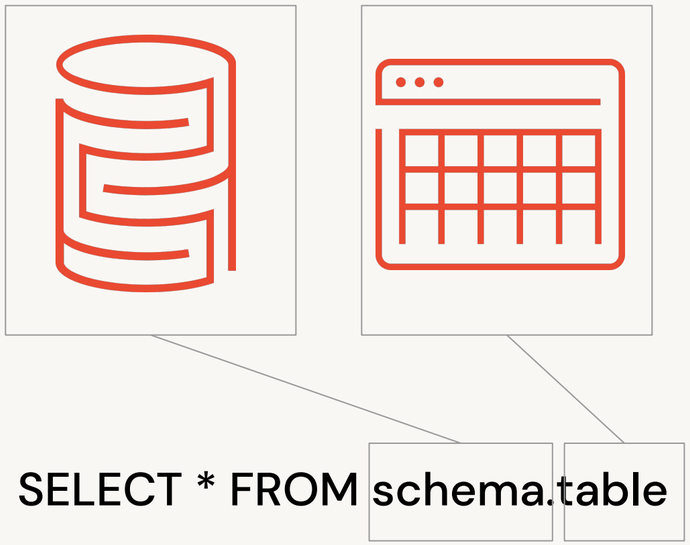
By the end of this course, you will be able to:

1. Describe the security model for governing data objects in Unity Catalog
2. Define data access rules and manage data ownership
3. Secure access to external storage

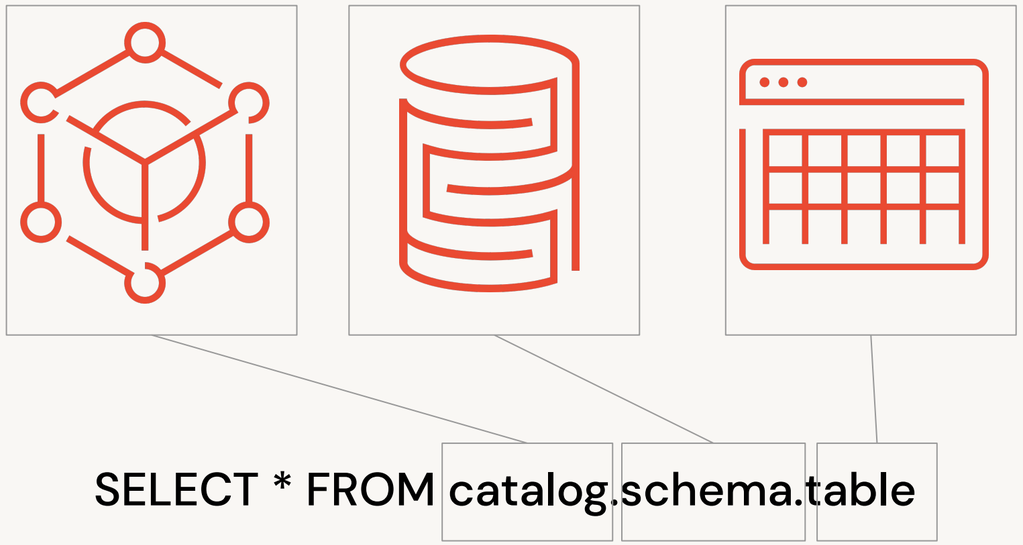
Unity Catalog Three-level Namespace

Three-level Namespace

Traditional SQL two-level namespace

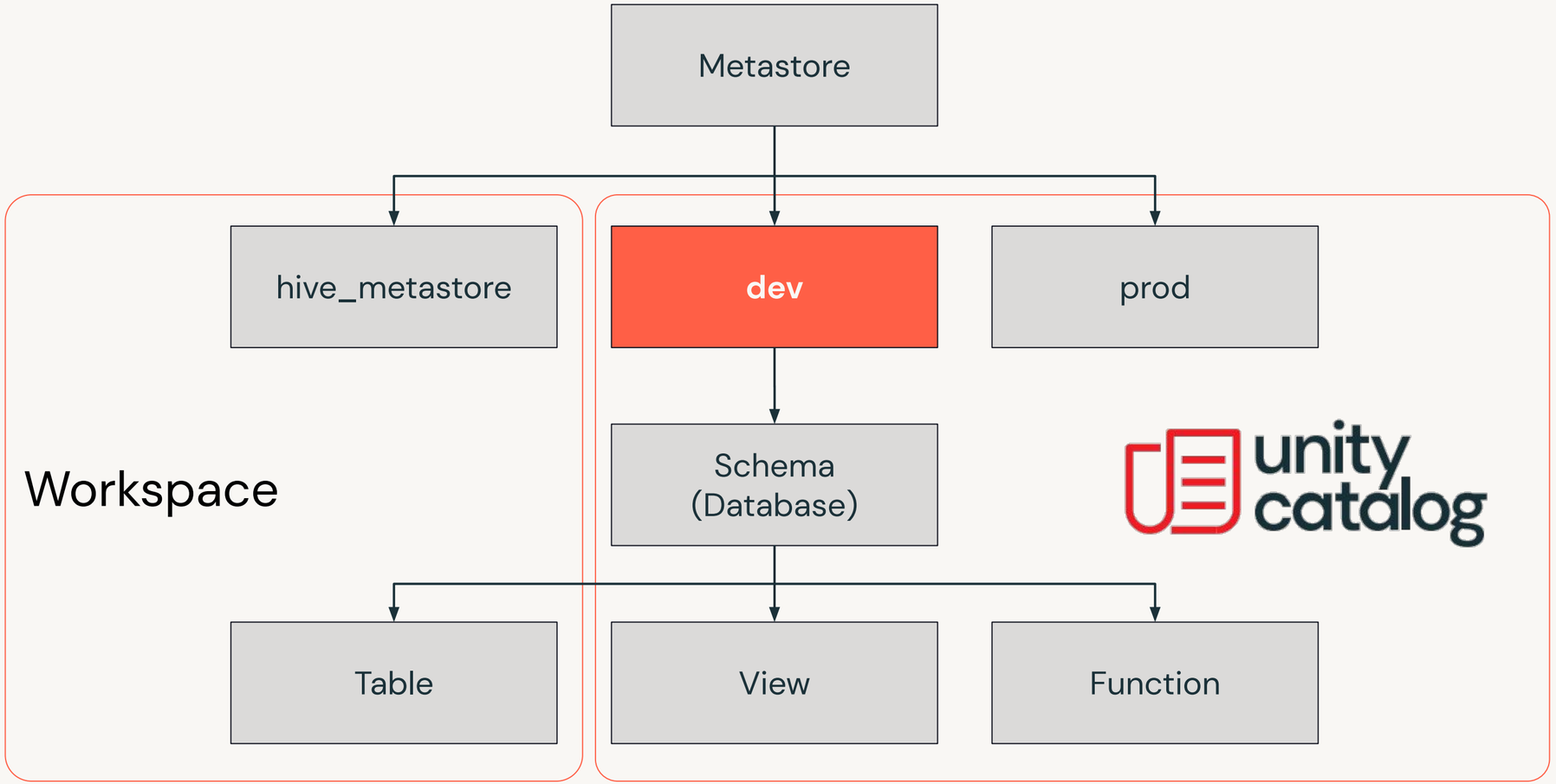


Three-level namespace with Unity Catalog



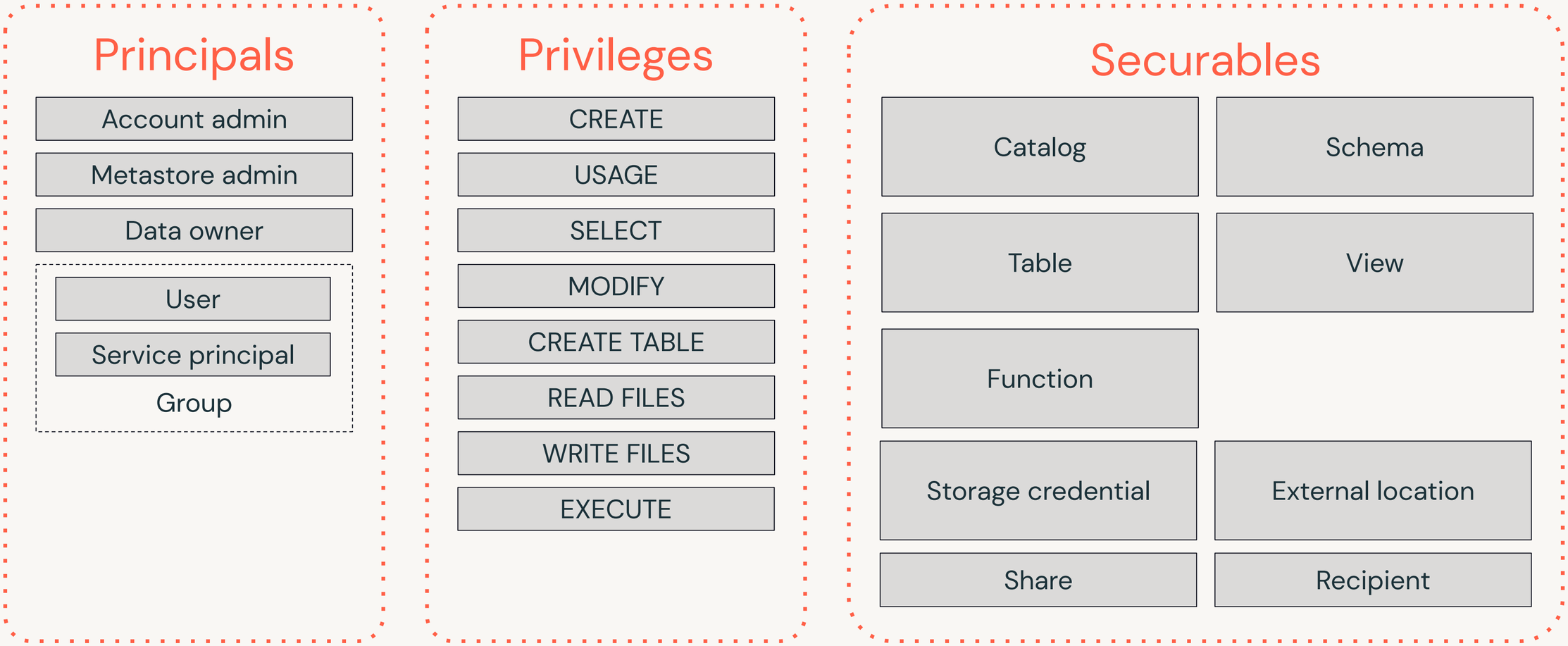
Three-level Namespace

Accessing legacy Hive metastore

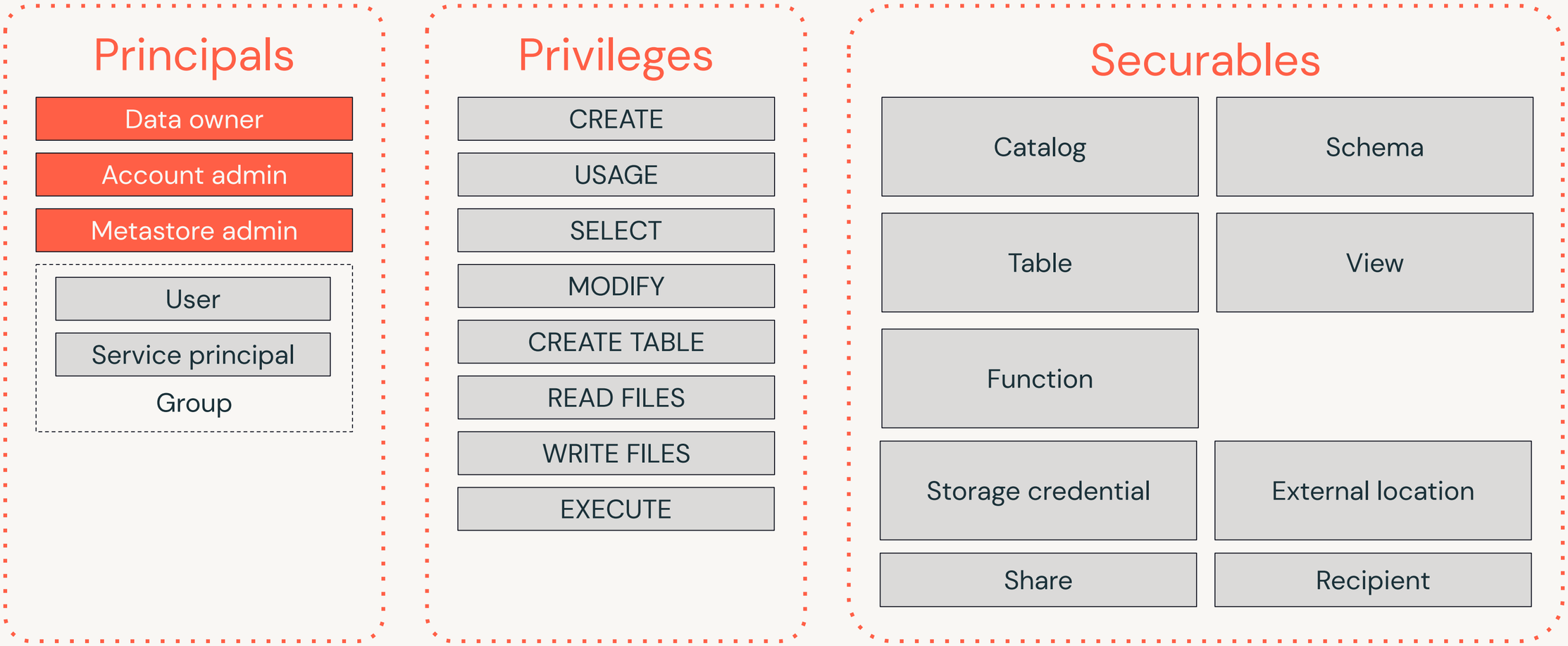


Unity Catalog Security Model

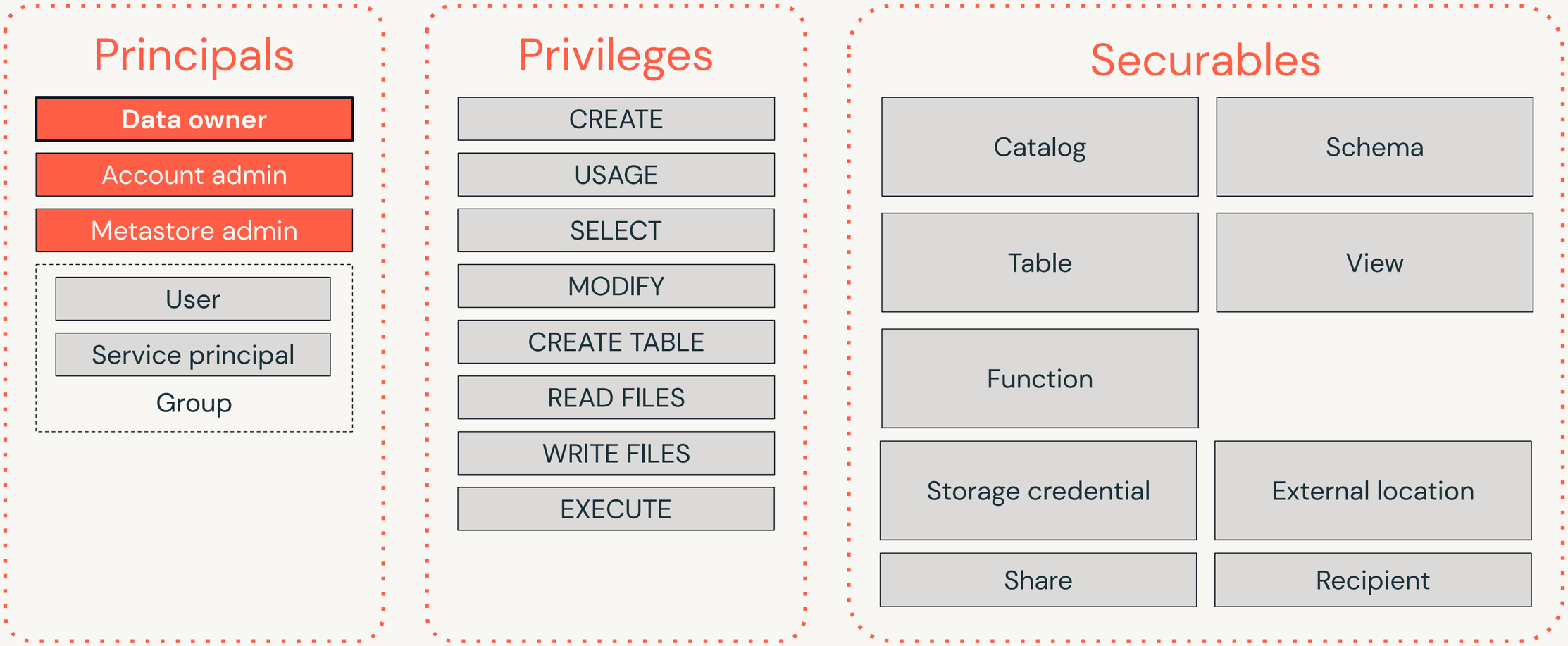
Security model



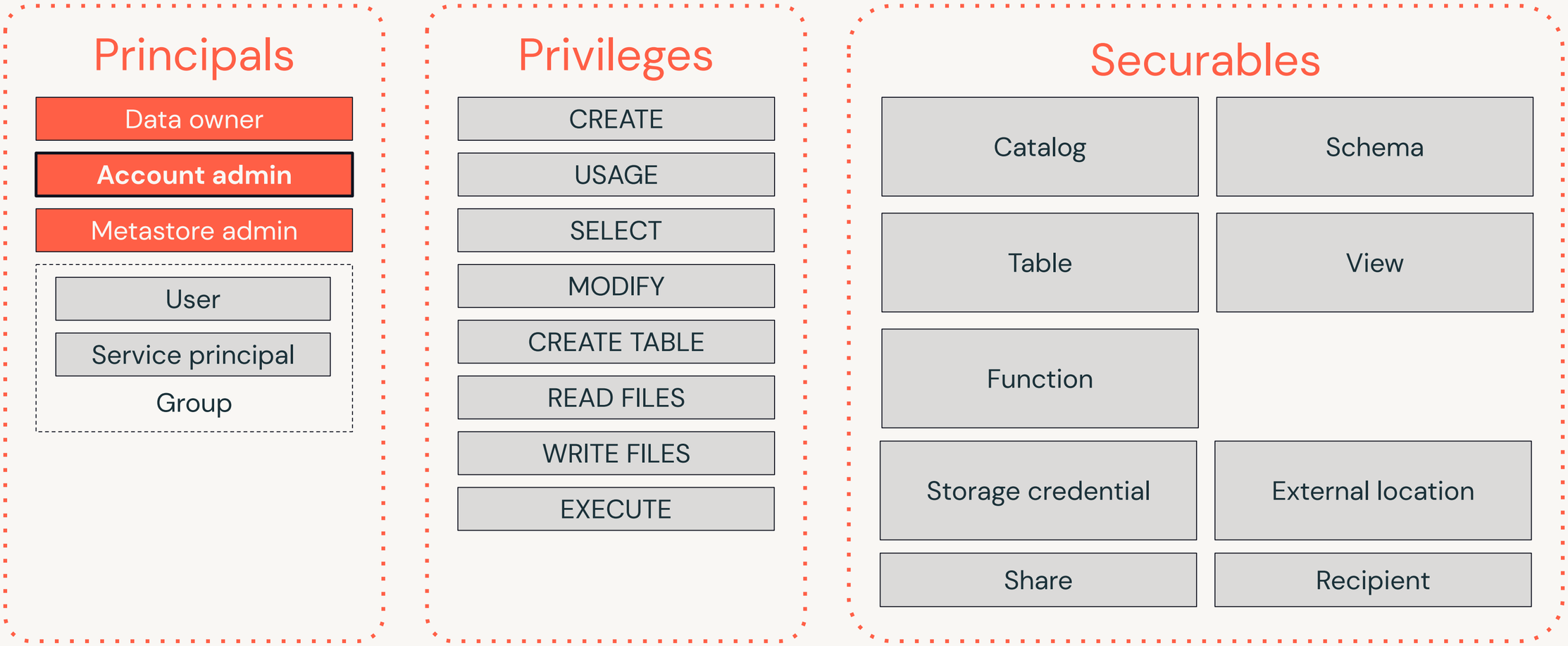
Security model



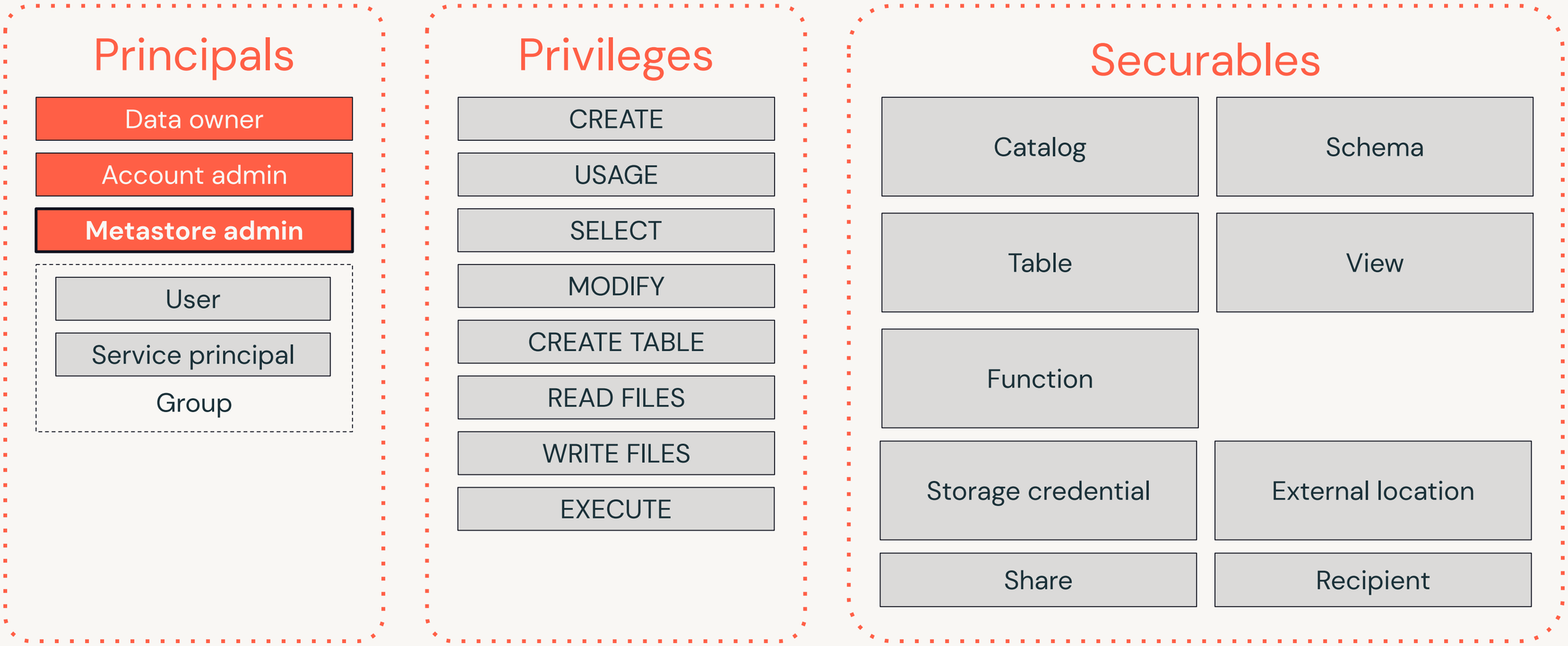
Security model



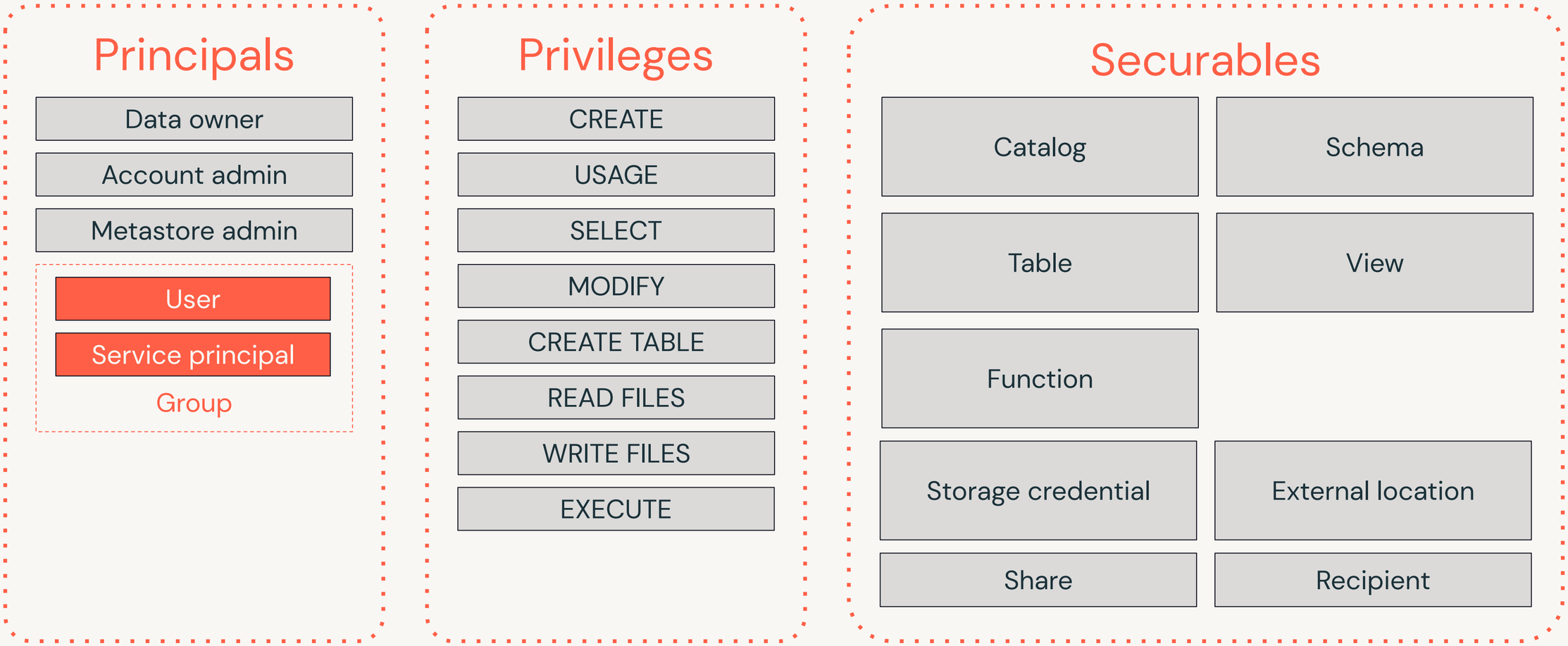
Security model



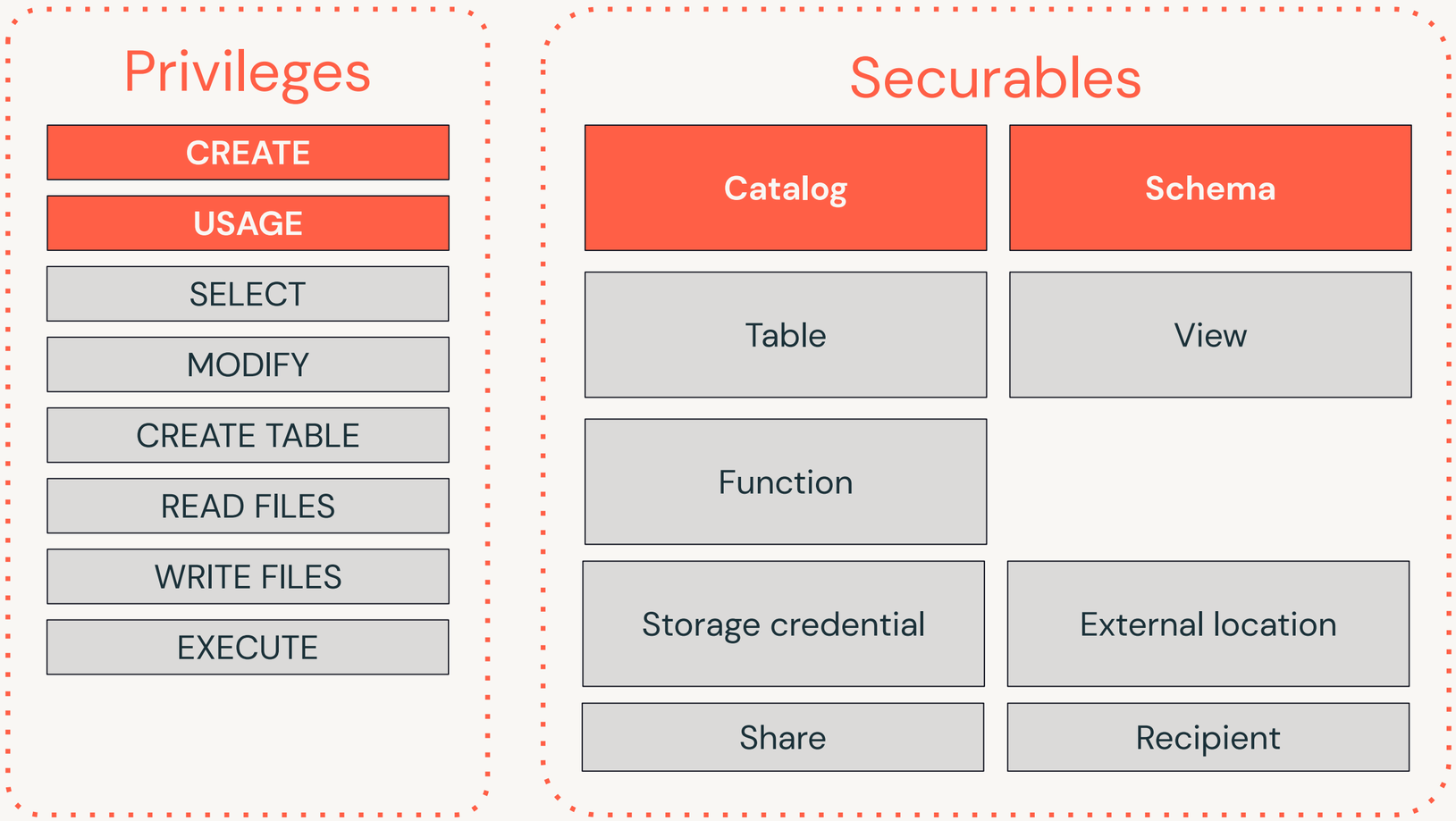
Security model



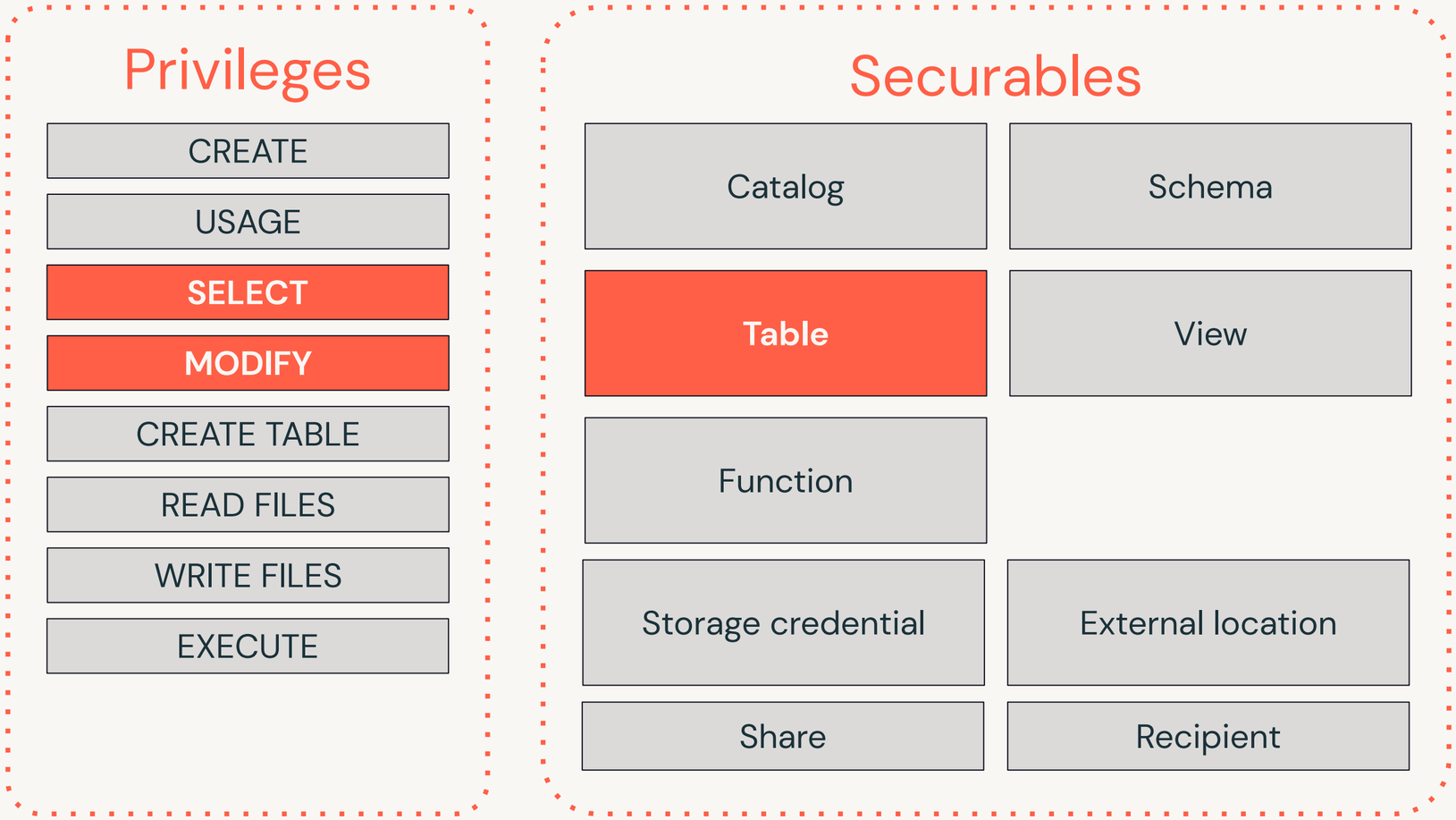
Security model



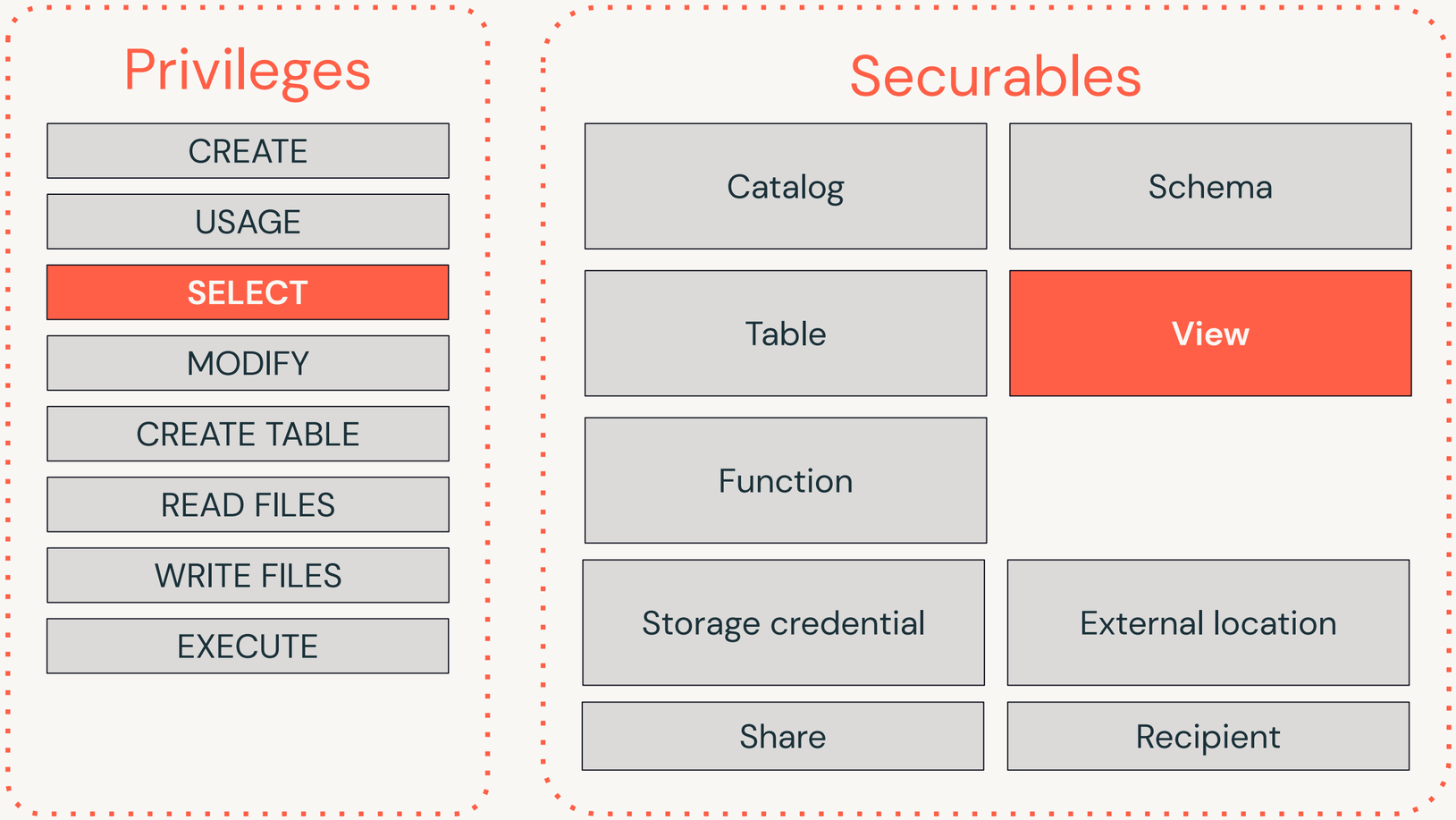
Security model



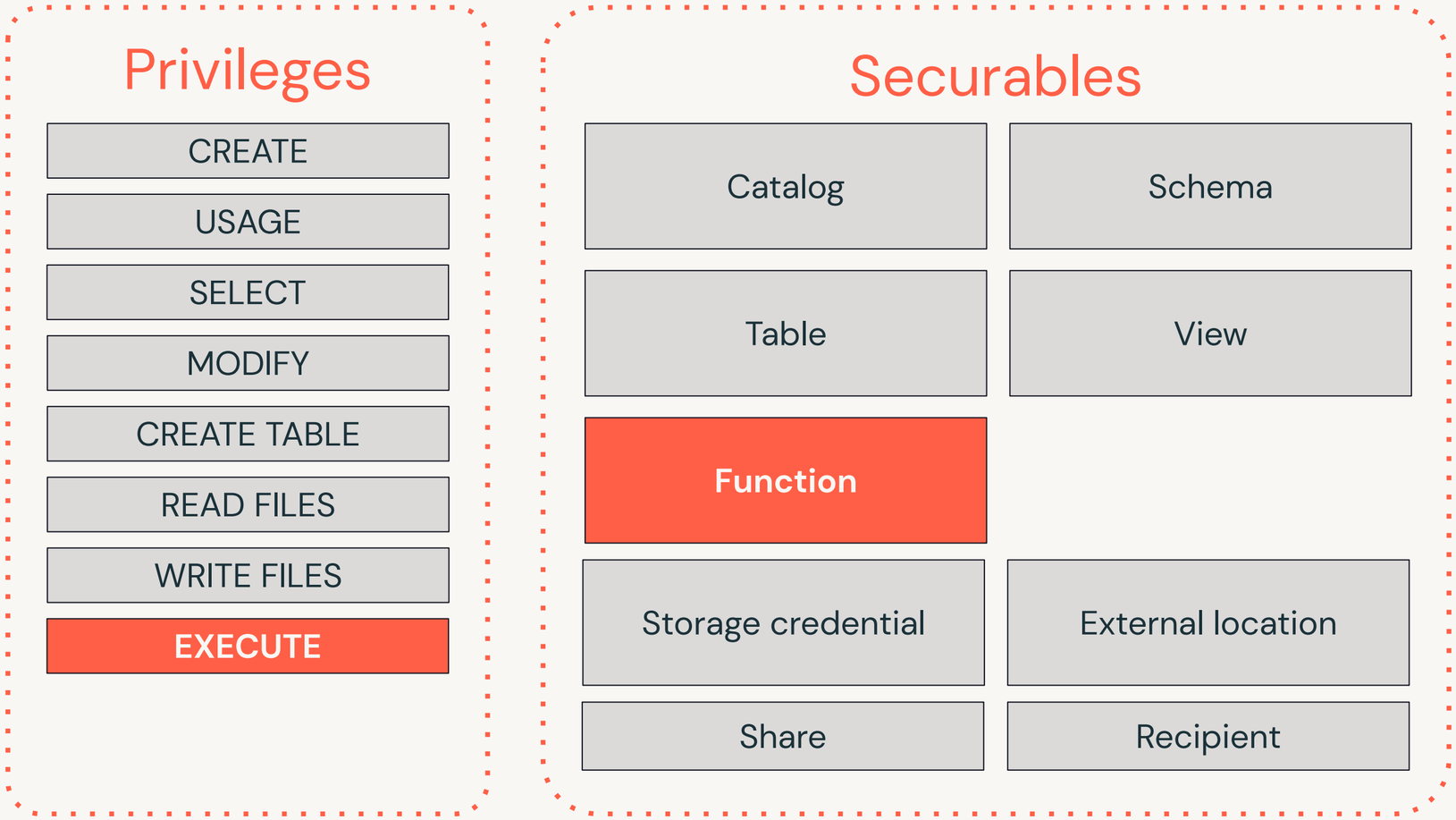
Security model



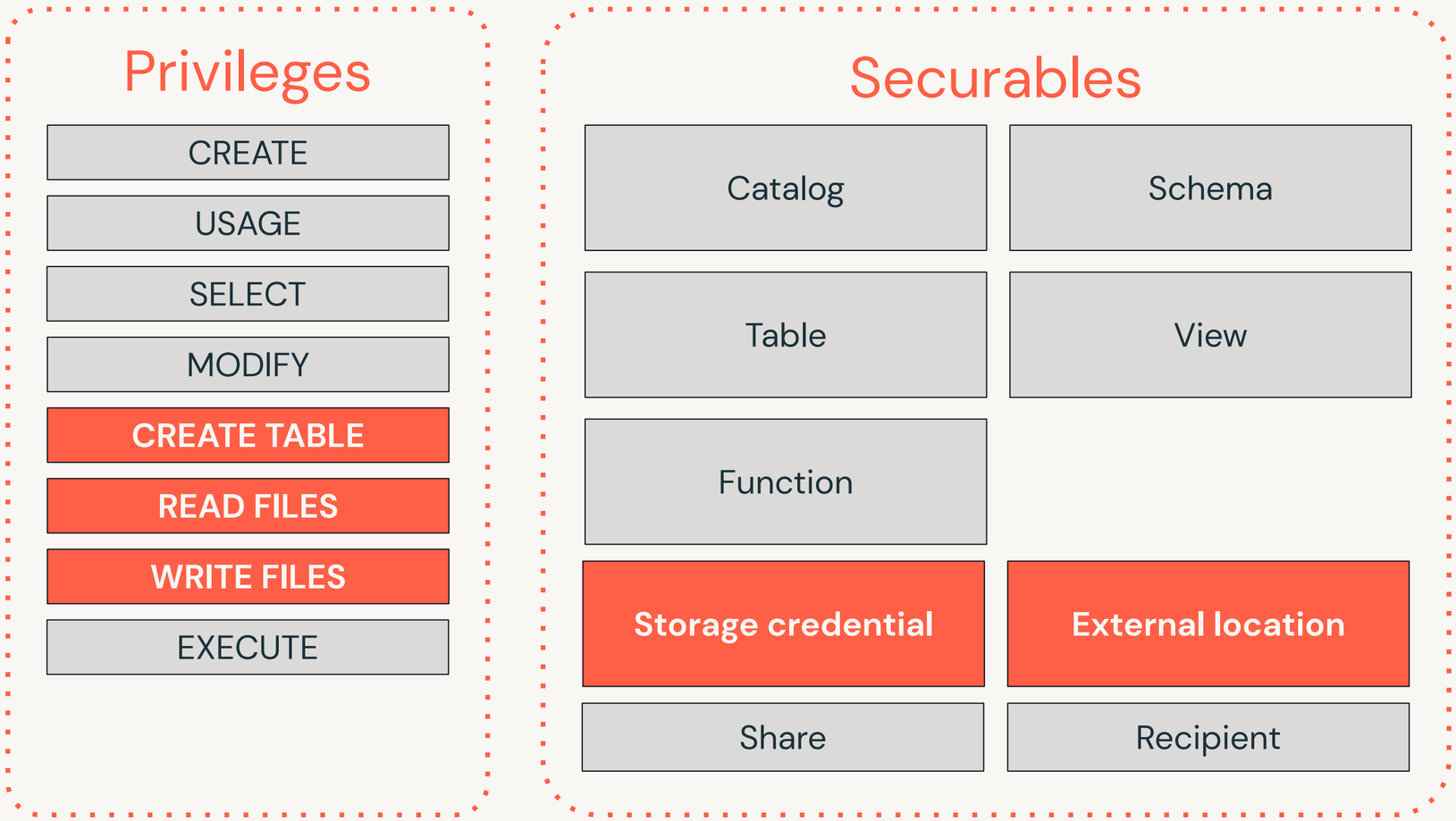
Security model



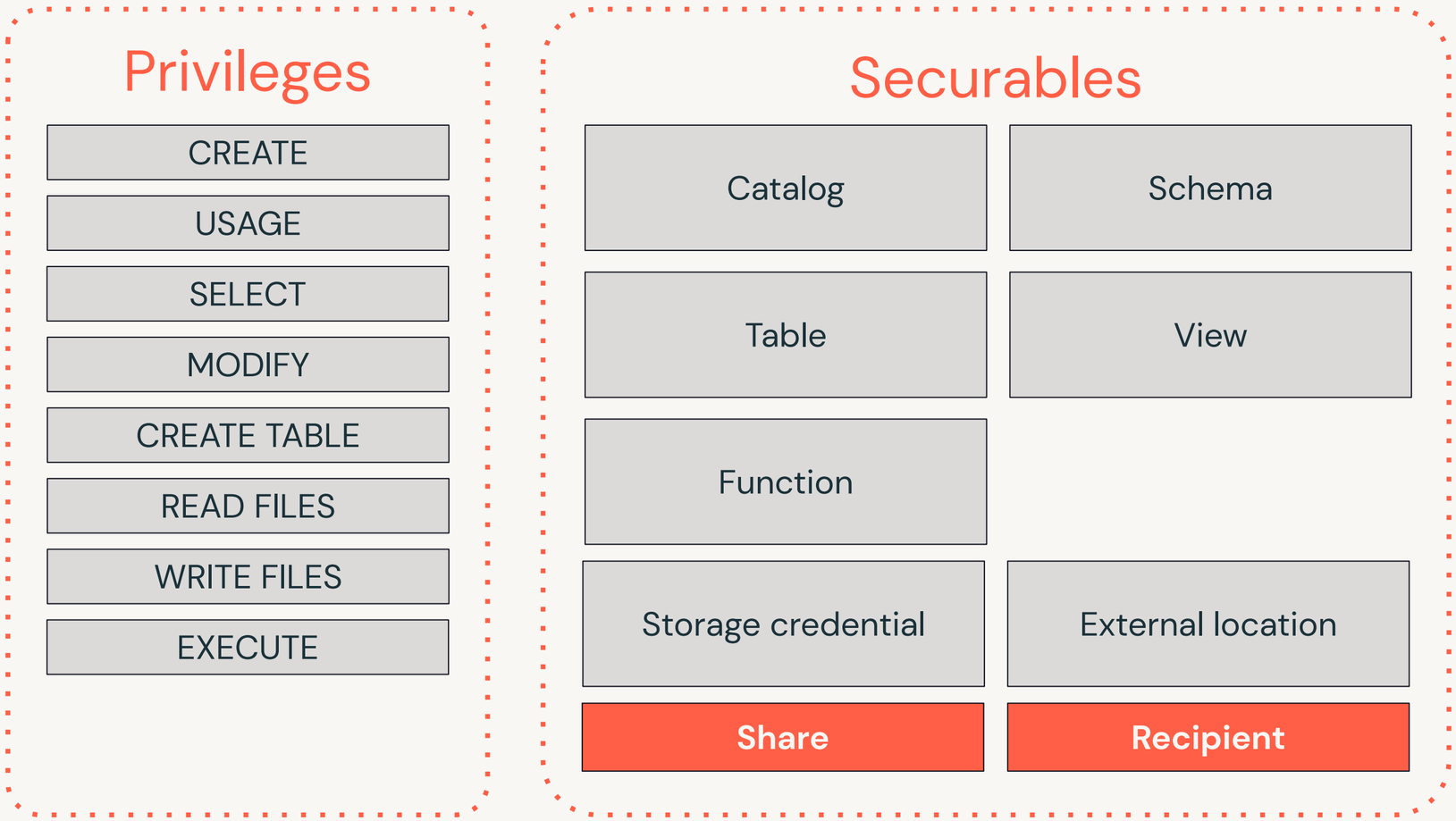
Security model



Security model

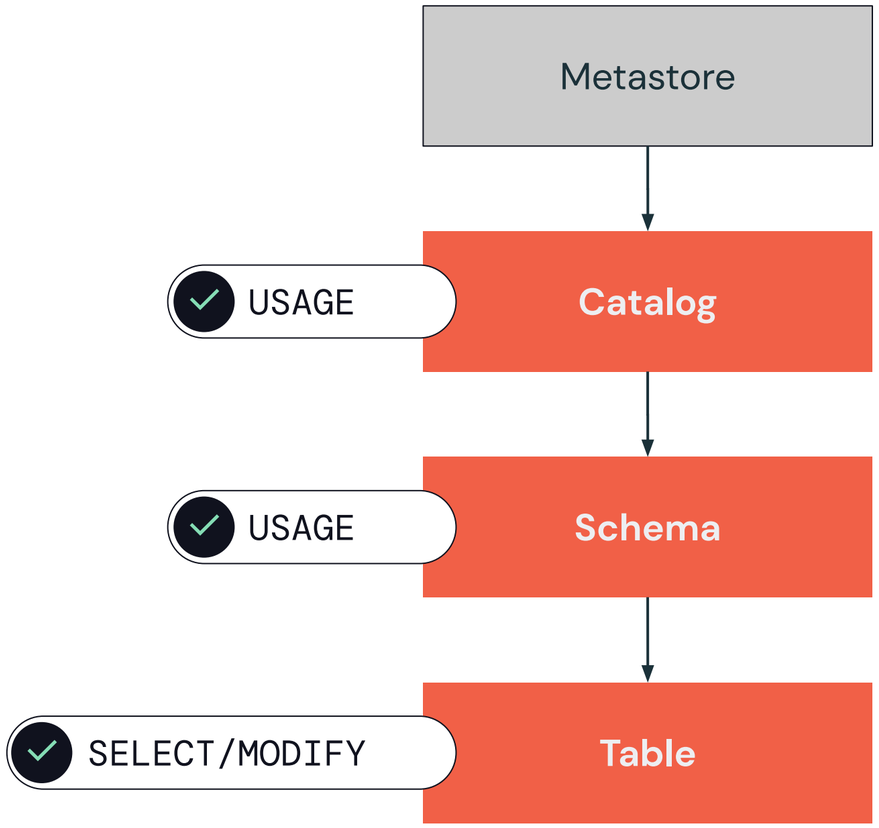


Security model



Privilege Recap

Tables



Querying tables (**SELECT**)

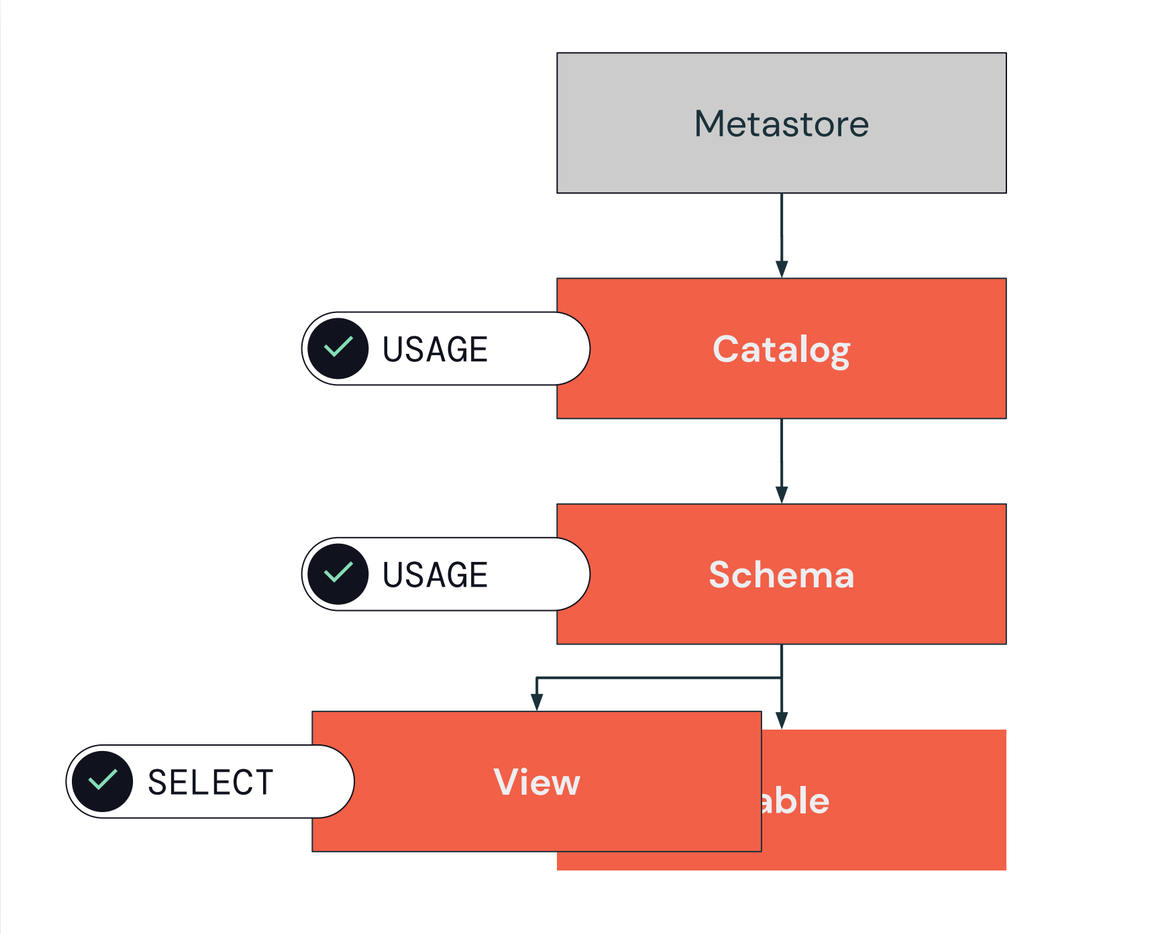
Modifying tables (**MODIFY**)

- Data (INSERT, DELETE)
- Metadata (ALTER)

Traverse container (**USAGE**)

Privilege Recap

Views



Abstract complex queries

- Aggregations
- Transformations
- Joins
- Filters

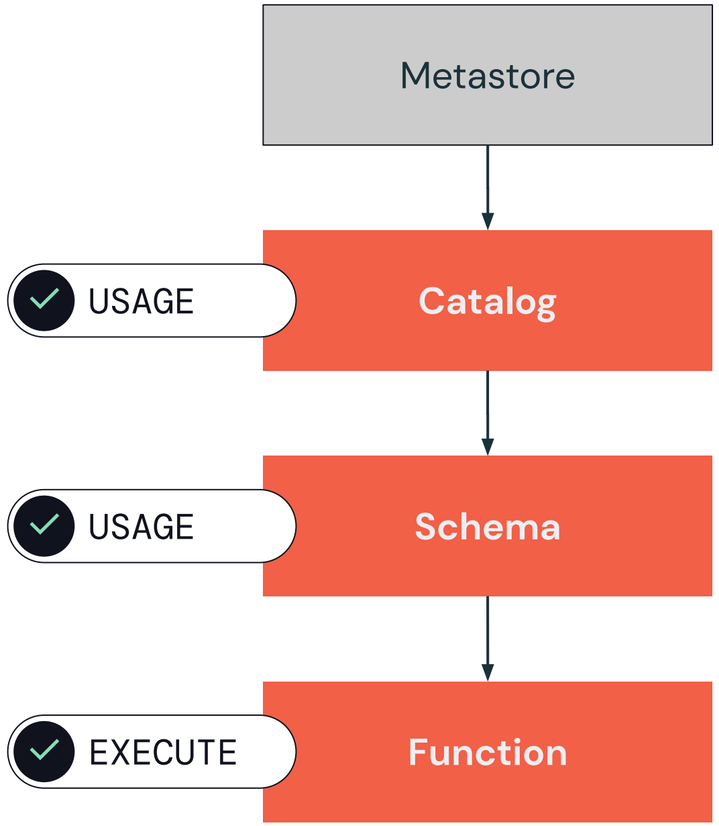
Enhanced table access control

Querying views (**SELECT**)

Traverse container (**USAGE**)

Privilege Recap

Functions



Provide custom code via user-defined functions

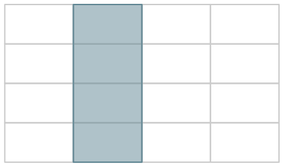
Using functions (**EXECUTE**)

Traverse container (**USAGE**)

Dynamic Views

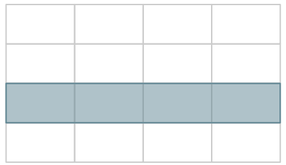
Limit access to columns

Omit column values from output



Limit access to rows

Omit rows from output



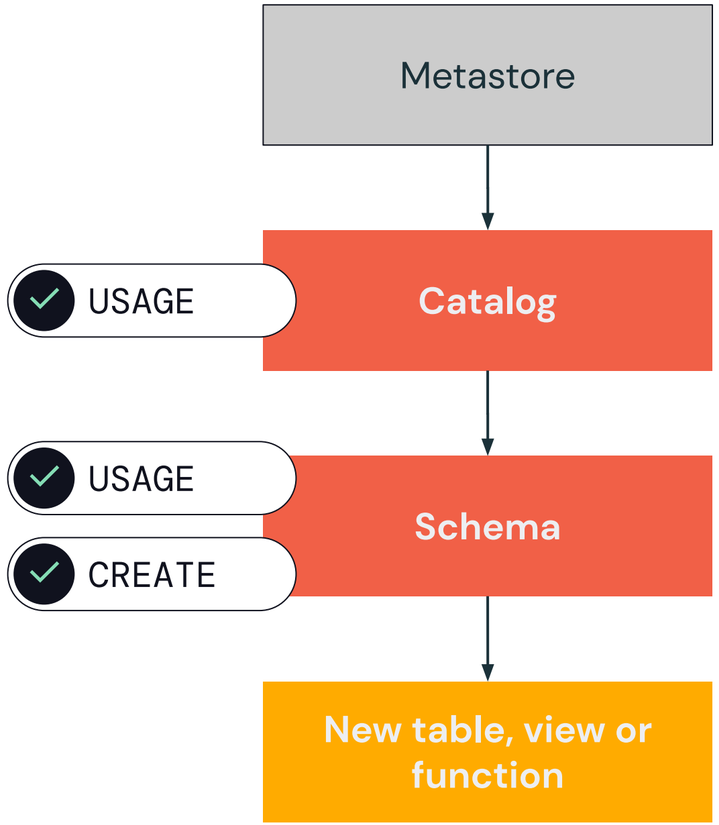
Data Masking

Obscure data

●●●●●●@databricks.com

Can be conditional on a specific user/service principal or group membership through Databricks-provided functions

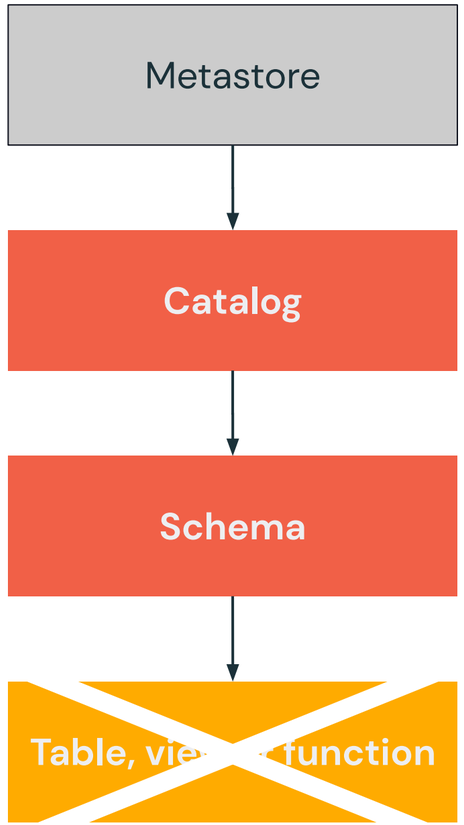
Creating New Objects



Create new objects (**CREATE**)

Traverse container (**USAGE**)

Deleting Objects



DROP objects

Unity Catalog External Storage

Storage Credentials and External Locations

Storage Credential

Enables Unity Catalog to connect to an external cloud store

Examples include:

- IAM role for AWS S3
- Service principal for Azure Storage

External Location

Cloud storage path + storage credential

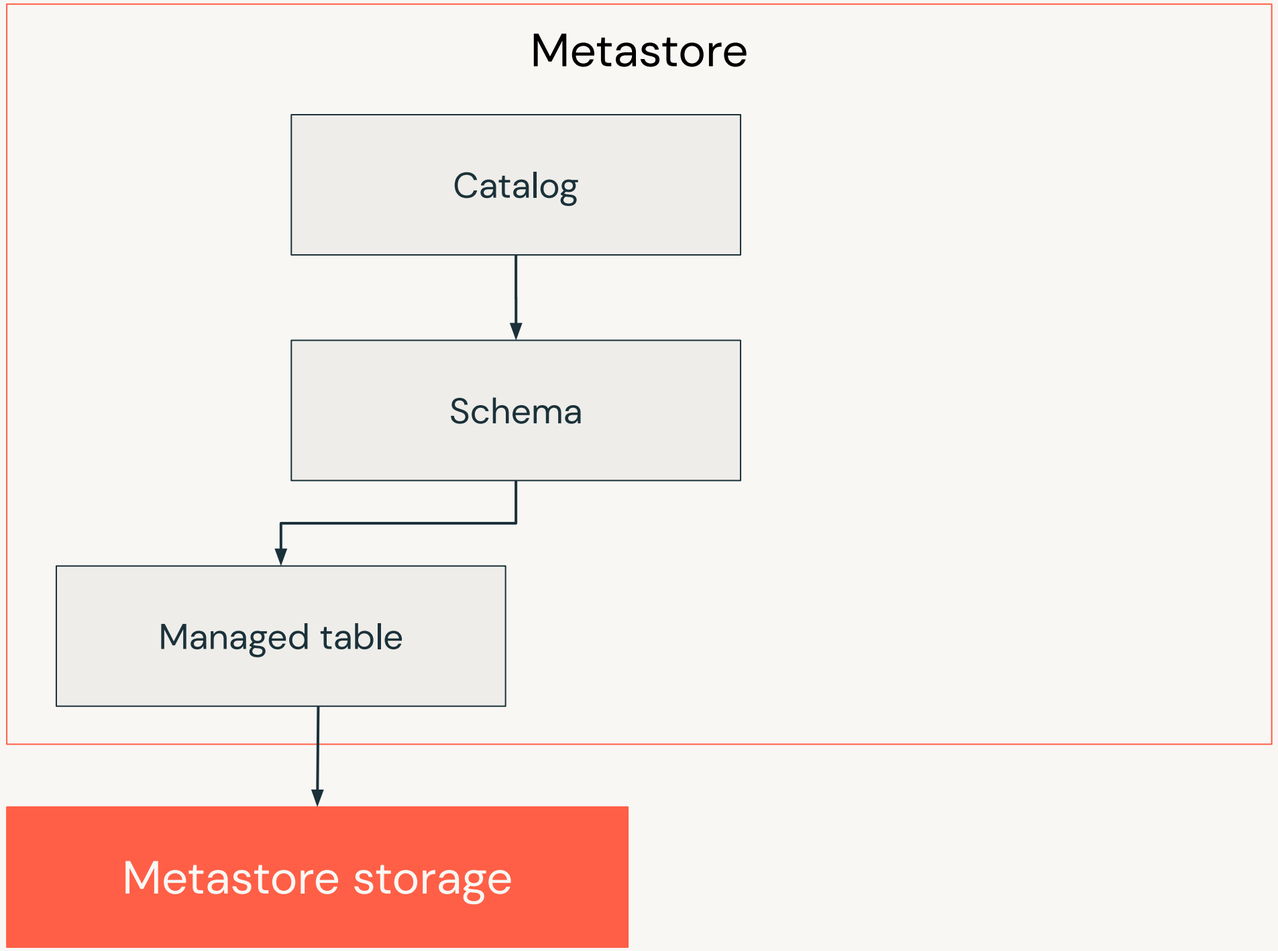
- Self-contained object for accessing specific locations in cloud stores
- Fine-grained control over external storage

Storage Credentials and External Locations

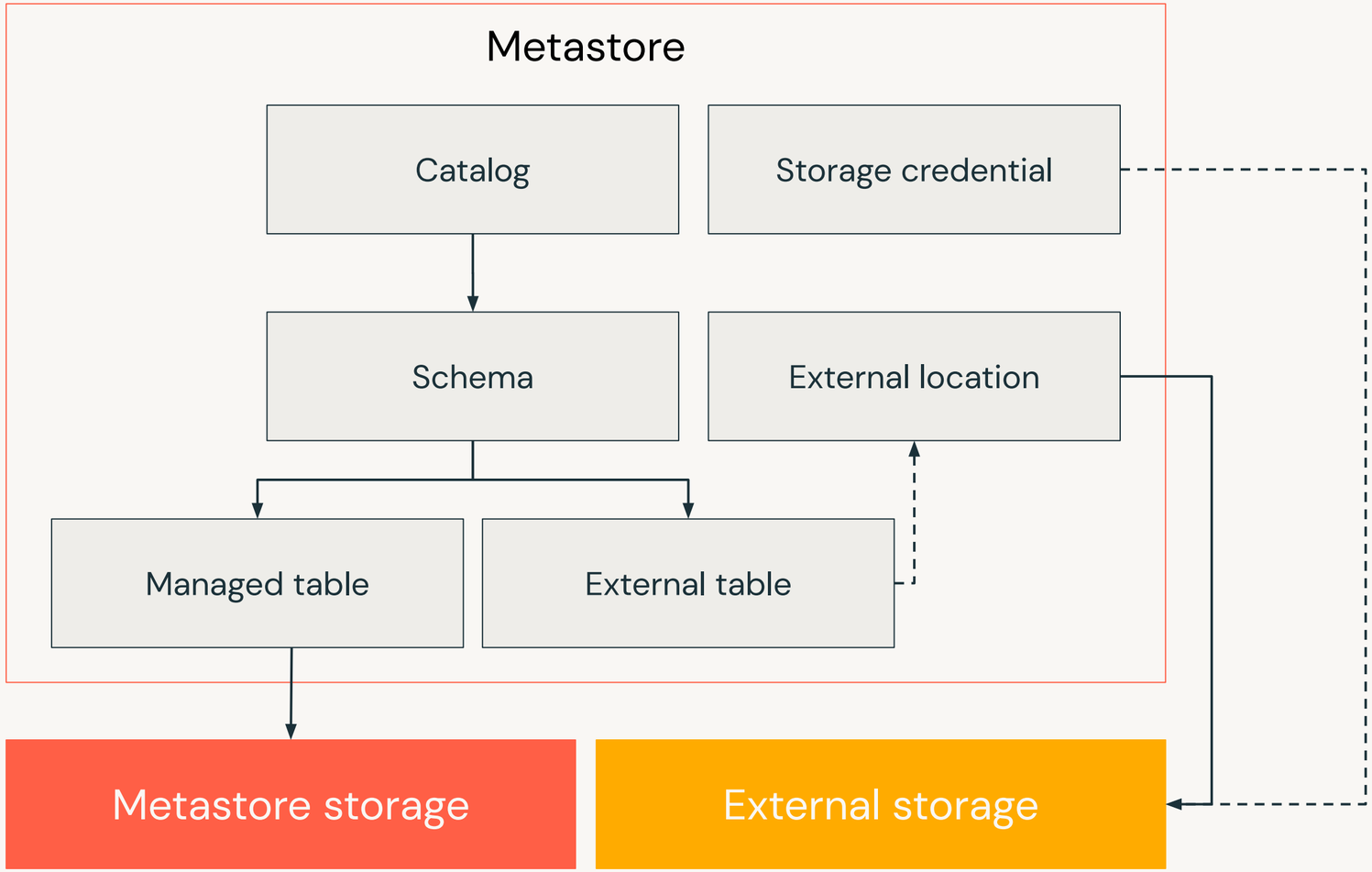
Access Control

CREATE TABLE	READ FILES	WRITE FILES
Create an External Table directly using this Storage Credential	Read files directly using this Storage Credential	Write files directly using this Storage Credential
Storage Credential		
External Location		
Create an External Table from files governed by this External Location	Read files governed by this External Location	Write files governed by this External Location

Managed Tables



External Tables



databricks

Unity Catalog Patterns and Best Practices

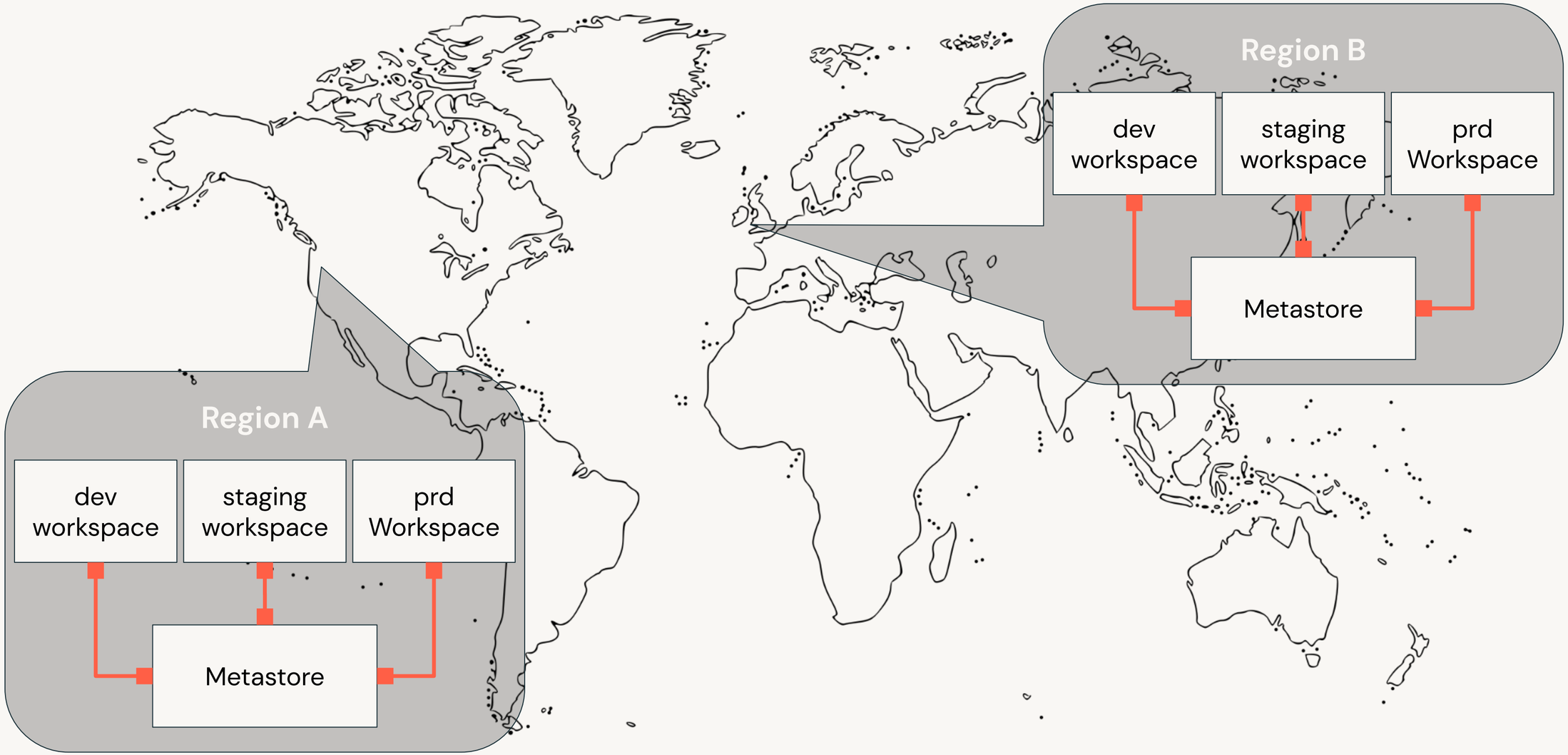
Course Objectives

By the end of this course, you will be able to:

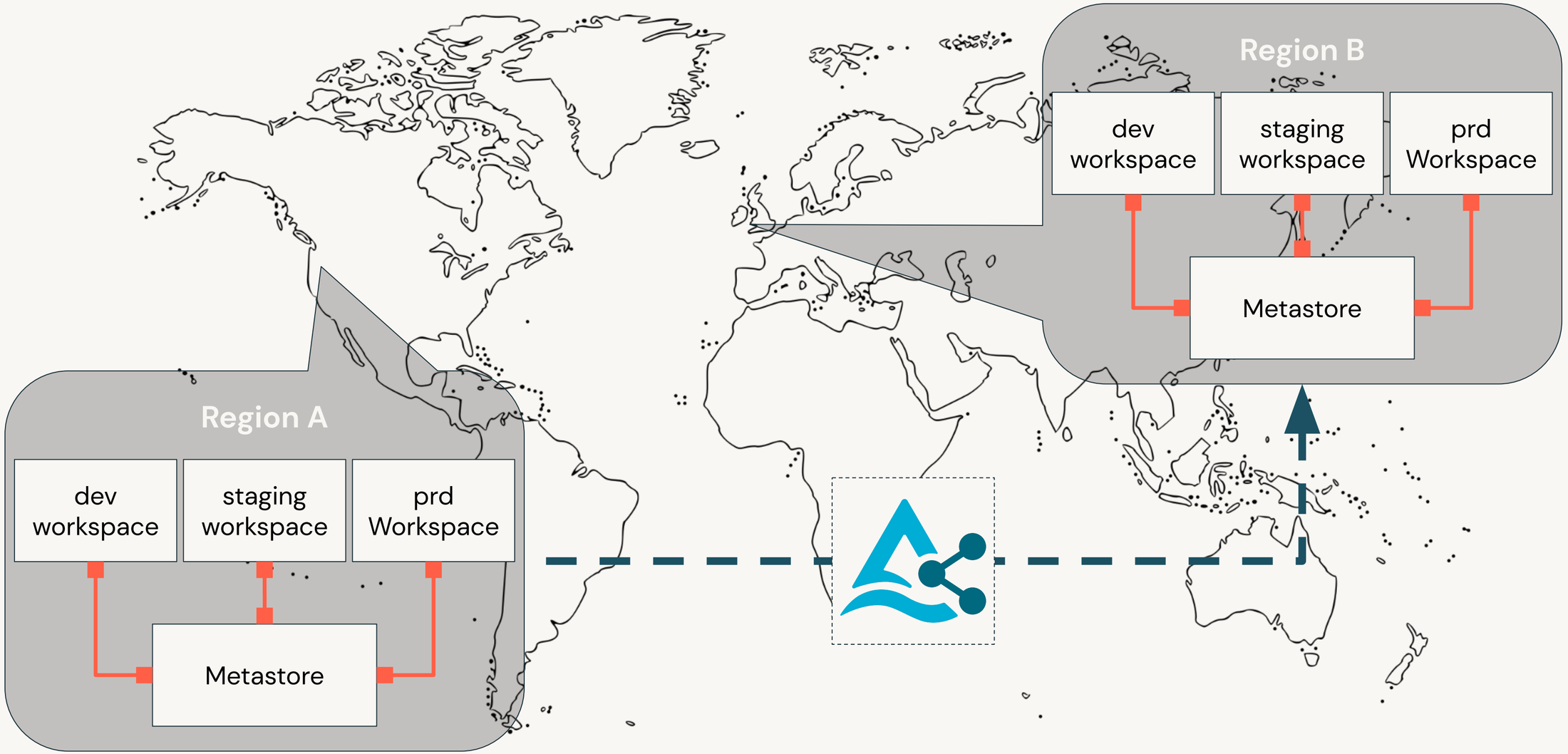
1. Adopt Databricks recommendations into your organization's Unity Catalog based solutions

Metastores

Metastores



Metastores



Data Segregation

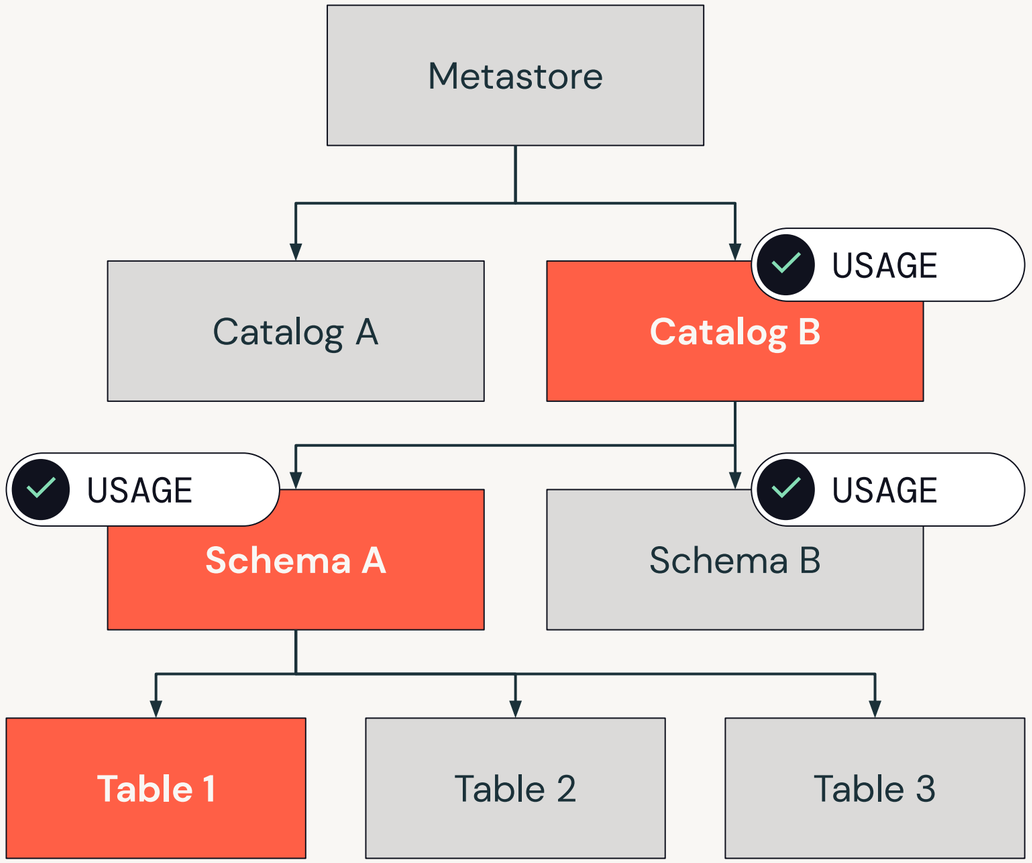
Data Segregation

Use catalogs to segregate data

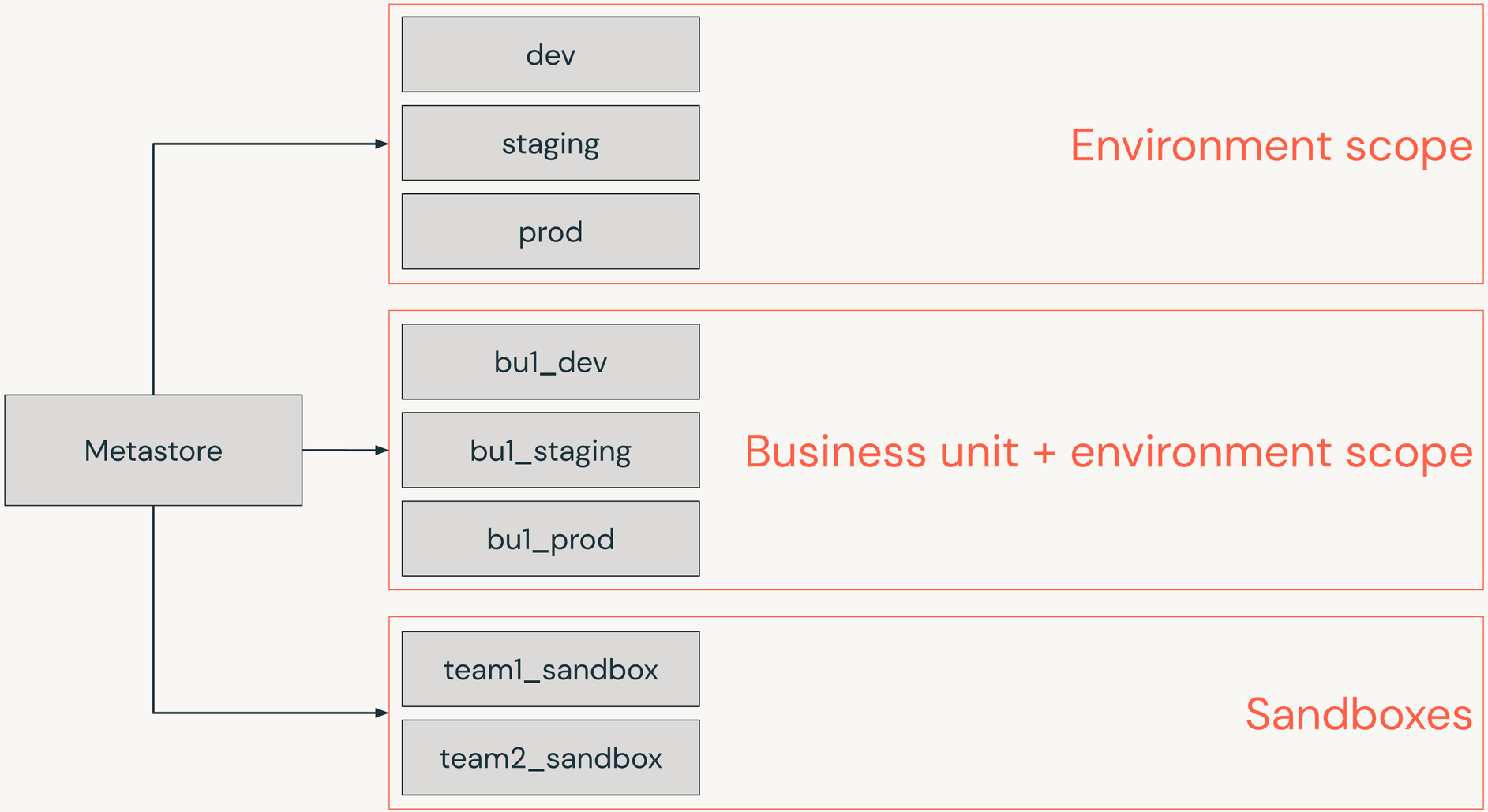
Apply permissions appropriately

For example group B:

- **USAGE** on catalog B
- **USAGE** on all applicable schemas in catalog B
- **SELECT/MODIFY** on applicable tables



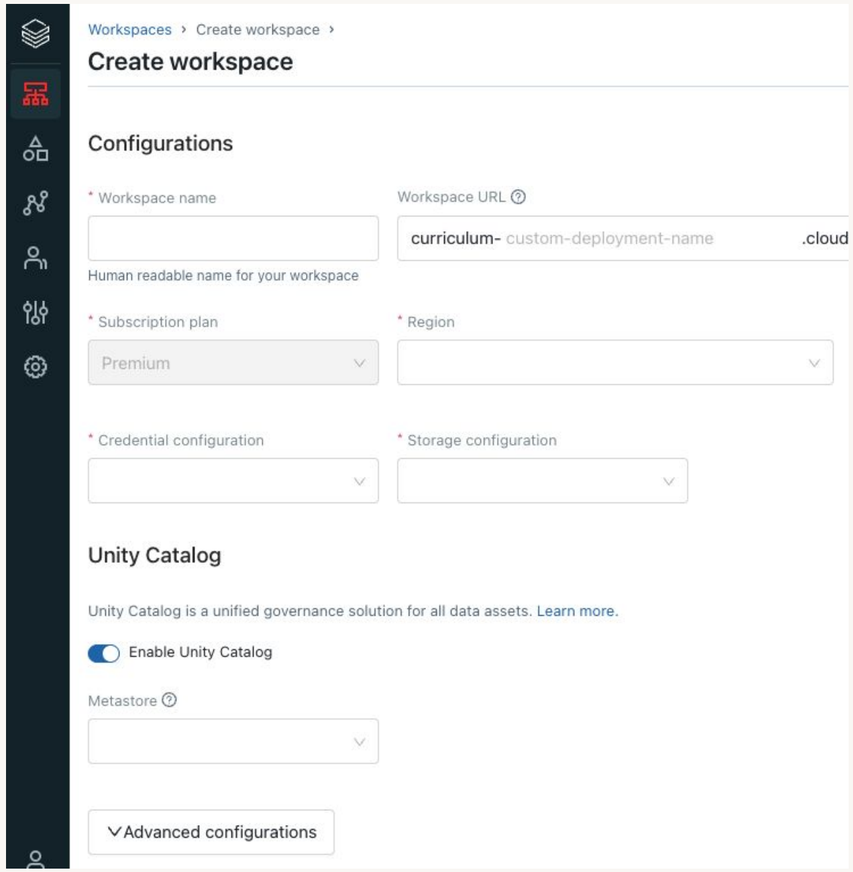
Data Segregation



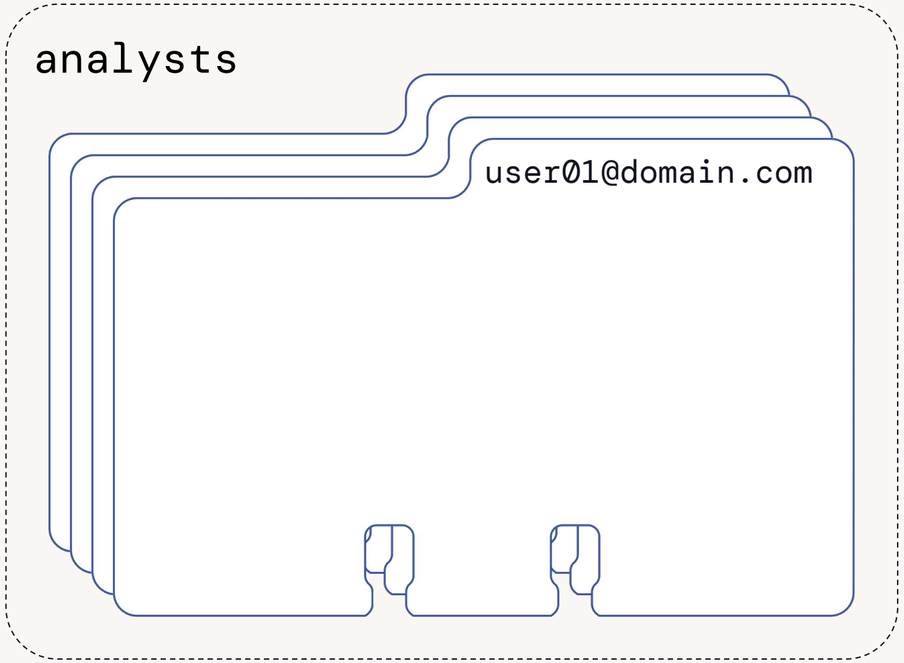
Identity Management

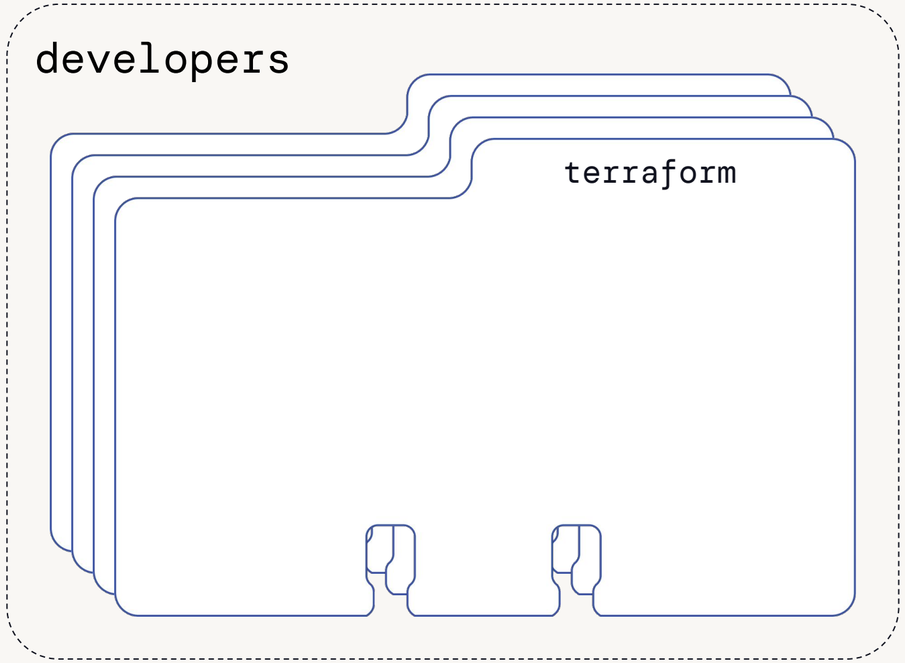
Account-level Identities

- Principals must live within accounts
- Databricks has automatically promoted Workspace-level principals to their respective accounts
- All identities should be managed at the account level moving forward
- Enable Unity Catalog for workspaces to enable identity federation

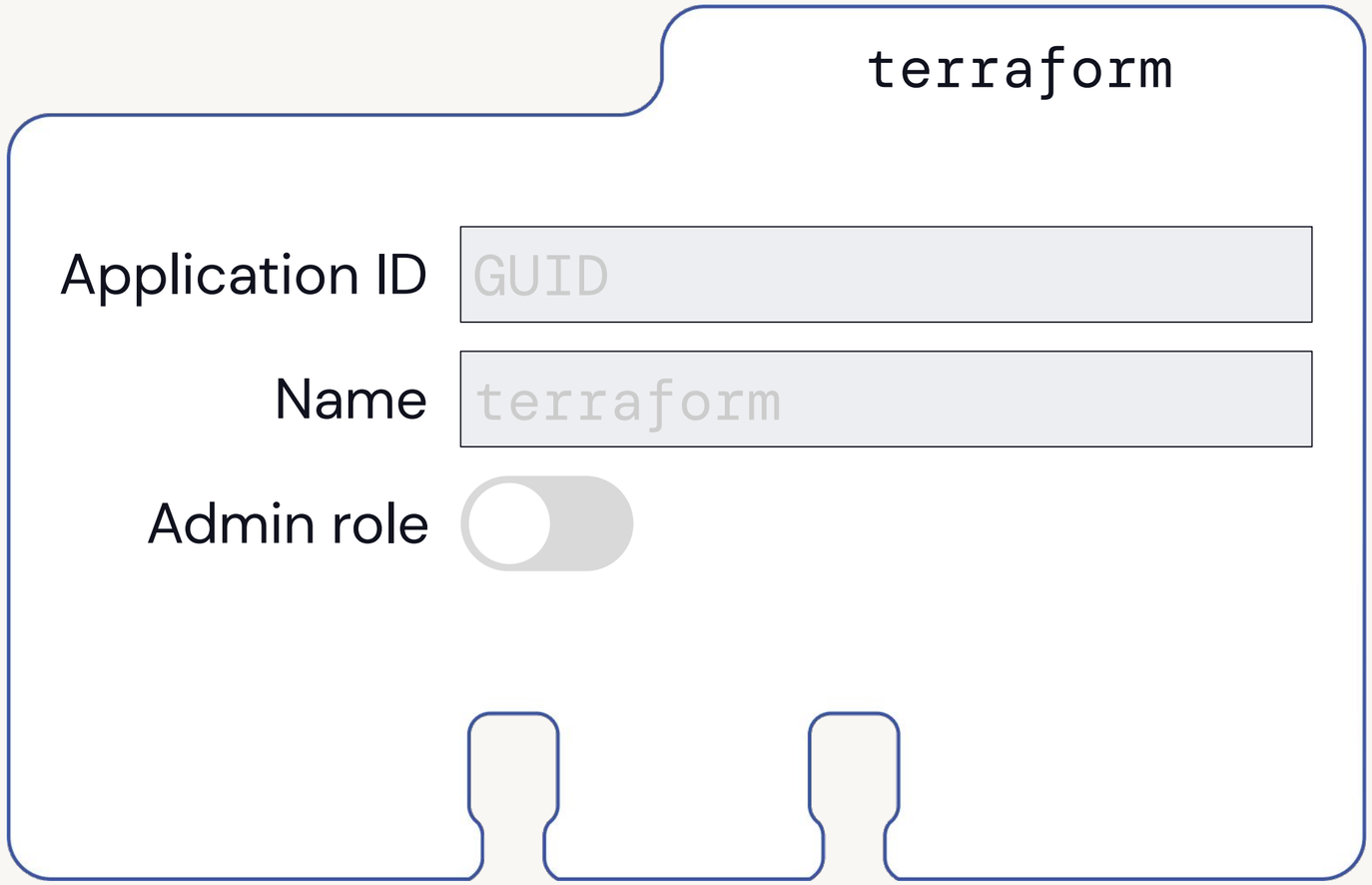


Groups





Service Principals



External Storage

Storage Credentials and External Locations

Storage Credential

Enables Unity Catalog to connect to an external cloud store

Examples include:

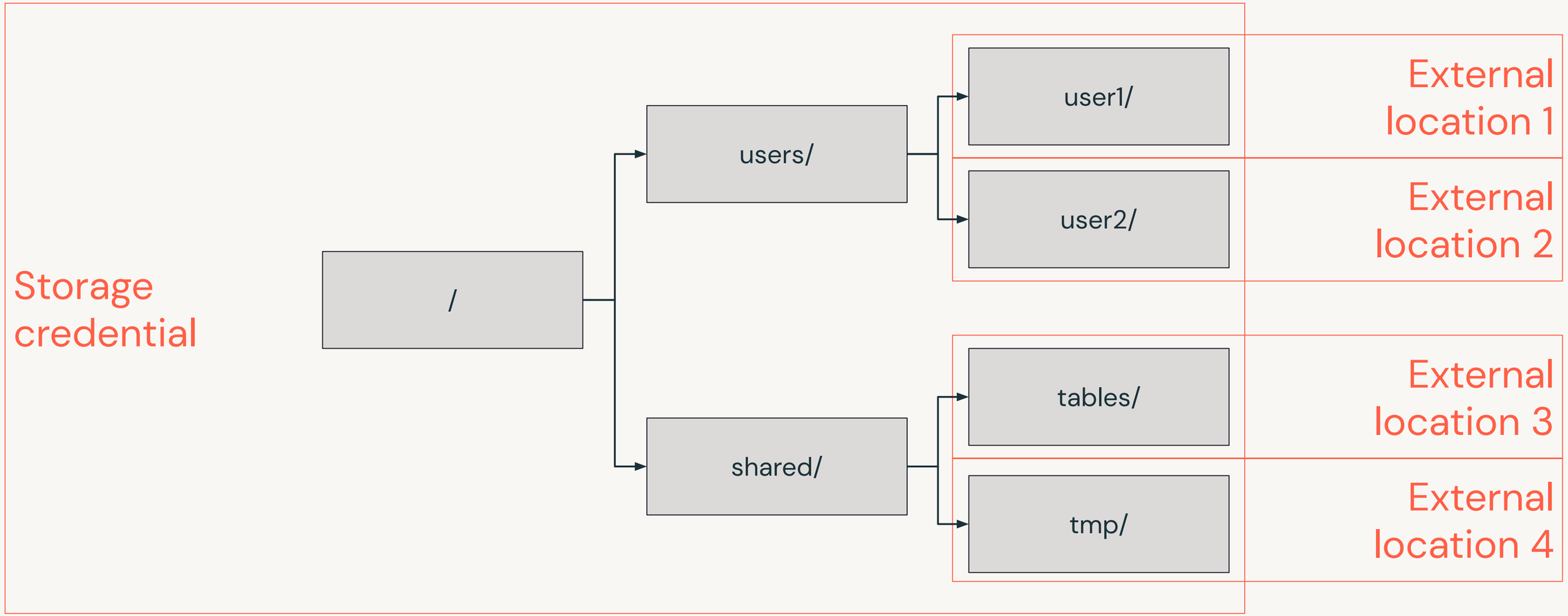
- IAM role for AWS S3
- Service principal for Azure Storage

External Location

Cloud storage path + storage credential

- Self-contained object for accessing specific locations in cloud stores
- Fine-grained control over external storage

Storage Credentials and External Locations



Managed versus External Tables

Managed Tables

Metadata lives in control plane

Data lives in metastore managed storage location

DROP discards data

Delta format only

External Tables

Metadata lives in control plane

Data lives in user-provided storage location

DROP leaves data intact

Several formats supported

External Tables

When to use?

Quick and easy upgrade from external table in Hive metastore

External readers or writers

Requirement for specific storage naming or hierarchy

Infrastructure-level isolation requirements

Non-Delta support requirement

External Tables

When to use?

Quick and easy upgrade from external table in Hive metastore

External readers or writers

Requirement for specific storage naming or hierarchy

Infrastructure-level isolation requirements

Non-Delta support requirement

External Tables

When to use?

Quick and easy upgrade from external table in Hive metastore

External readers or writers

Requirement for specific storage naming or hierarchy

Infrastructure-level isolation requirements

Non-Delta support requirement

External Tables

When to use?

Quick and easy upgrade from external table in Hive metastore

External readers or writers

Requirement for specific storage naming or hierarchy

Infrastructure-level isolation requirements

Non-Delta support requirement

External Tables

When to use?

Quick and easy upgrade from external table in Hive metastore

External readers or writers

Requirement for specific storage naming or hierarchy

Infrastructure-level isolation requirements

Non-Delta support requirement