


# TECHNICAL DEEP DIVE INTO UNITY CATALOG: PRACTITIONER'S PLAYBOOK



---

Ifi Derekli  
Pamela Pettit

# Product Safe Harbor Statement

This information is provided to outline Databricks' general product direction and is for informational purposes only. Customers who purchase Databricks services should make their purchase decisions relying solely upon services, features, and functions that are currently available. Unreleased features or functionality described in forward-looking statements are subject to change at Databricks discretion and may not be delivered as planned or at all.



# Ifi Derekli



- 14+ years of experience with big data platforms
- 4 years of experience on Databricks
- 7 years of security & governance focus

Field Engineering Sr Manager  
& Unity Catalog Specialist

# Pamela Pettit



- 12+ years experience as engineer and architect of production systems
- 2.5 years at Databricks

Senior Solution Architect &  
Unity Catalog Specialist

# Databricks Unity Catalog

## Unified governance for data and AI

*And now Open Source!!*

Unified visibility into data and AI

Single permission model for data and AI

AI-powered monitoring and observability

Open data sharing



### Databricks Unity Catalog

Access  
Controls

Lineage

Discovery

Monitoring

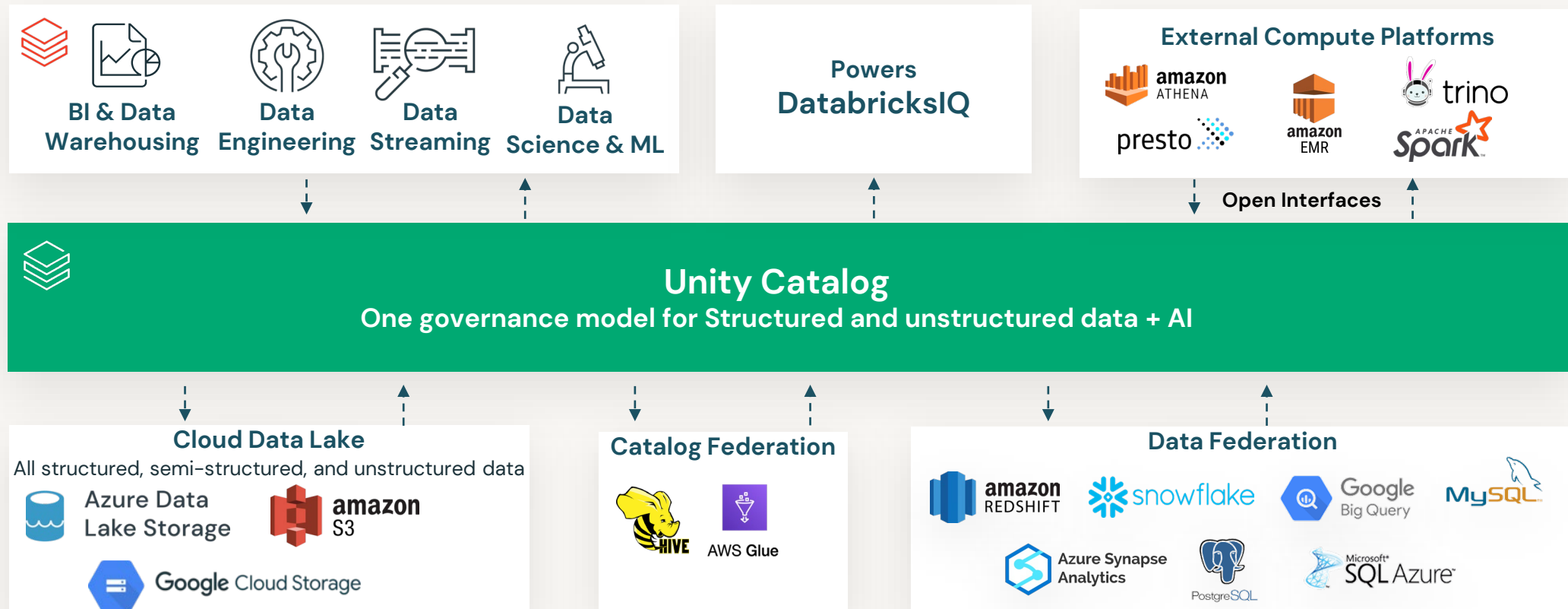
Auditing

Sharing

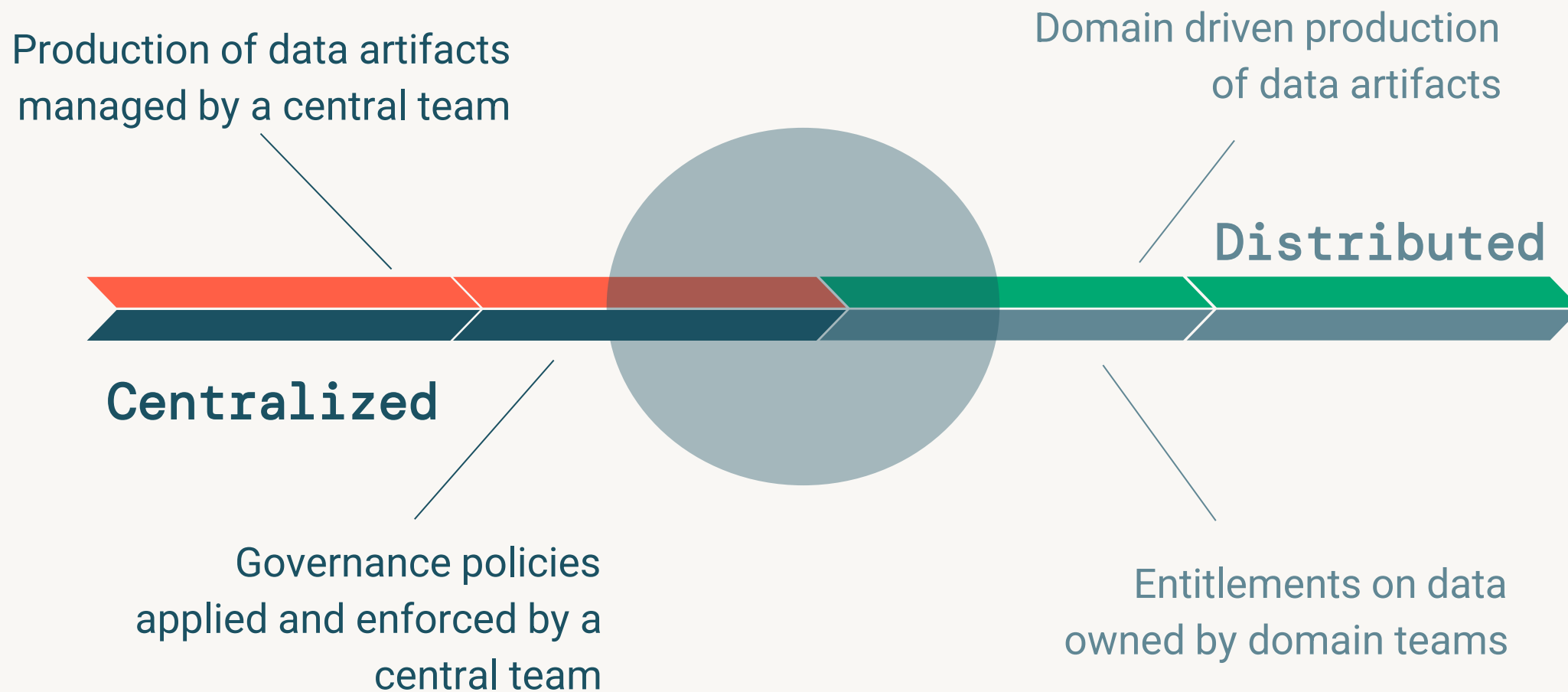
Metadata Management  
(Files | Tables | ML Models | Notebooks | Dashboards)



# Databricks Lakehouse unifies data and AI governance



# Spectrum of Organizational Governance



# Agenda

- Unity Catalog and Cloud Providers
- Getting started with the Metastore
- Register your data
- Discover your data with Search and Lineage
- Secure your data
- Open data sharing
- Audit your data
- Upgrade to Unity Catalog
- Architecture Patterns
- Q&A



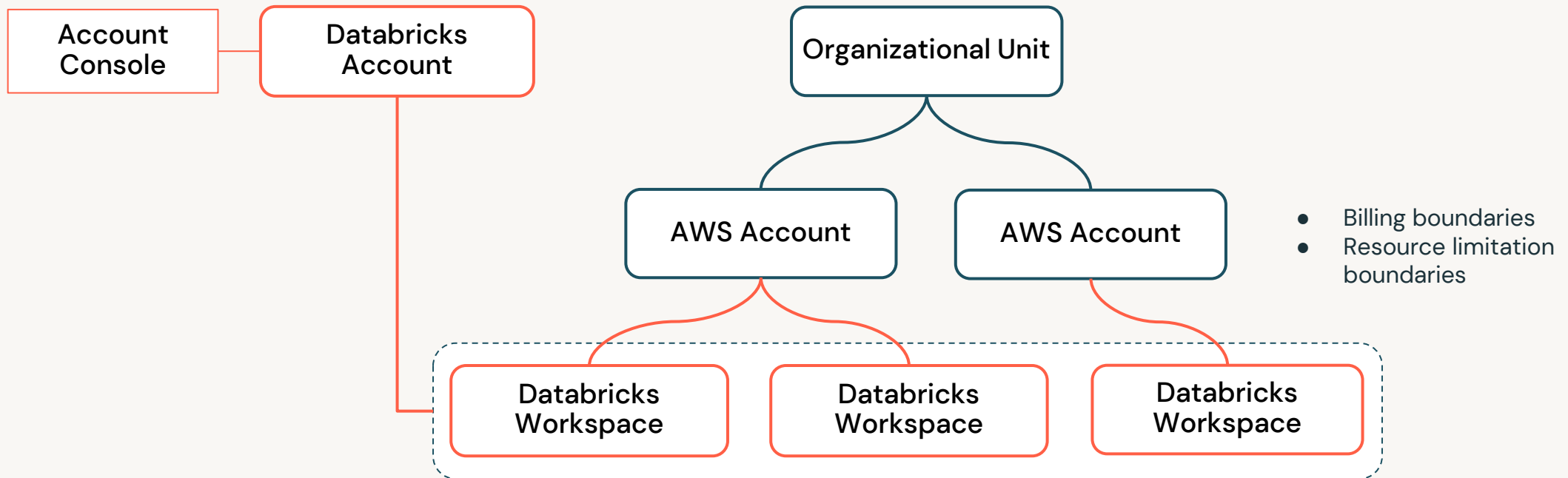
# Unity Catalog and Cloud Providers

# Top Questions: Deployment

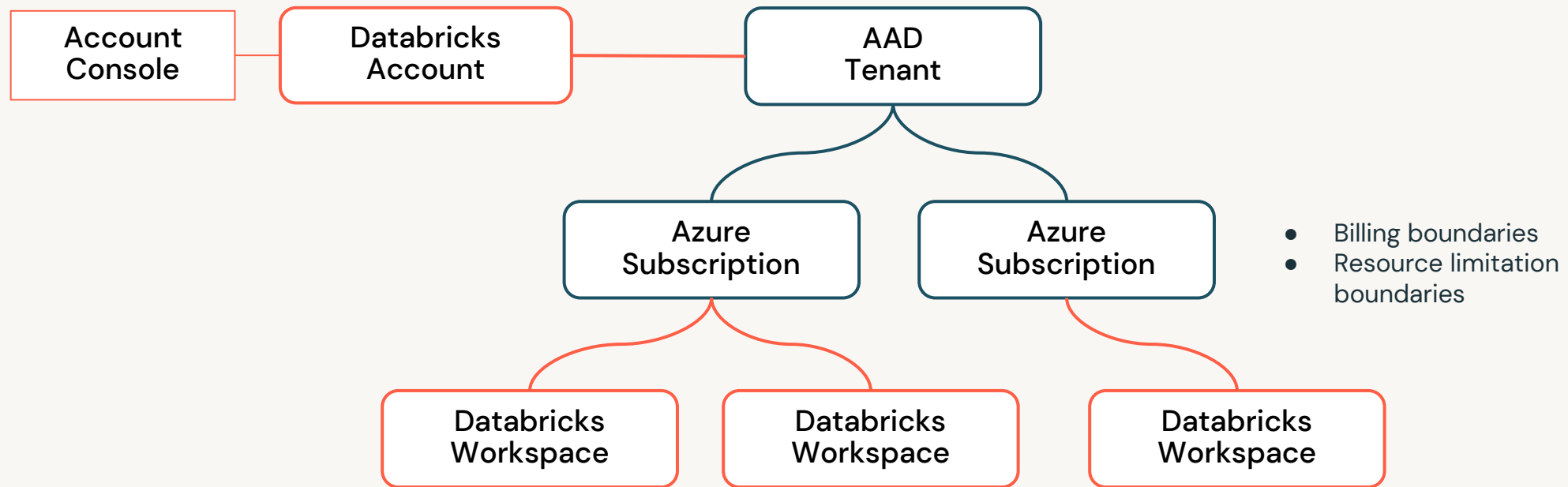
- Where does Unity Catalog fit in my Cloud Deployment?
- How should I think about the user roles that interact with Unity Catalog?



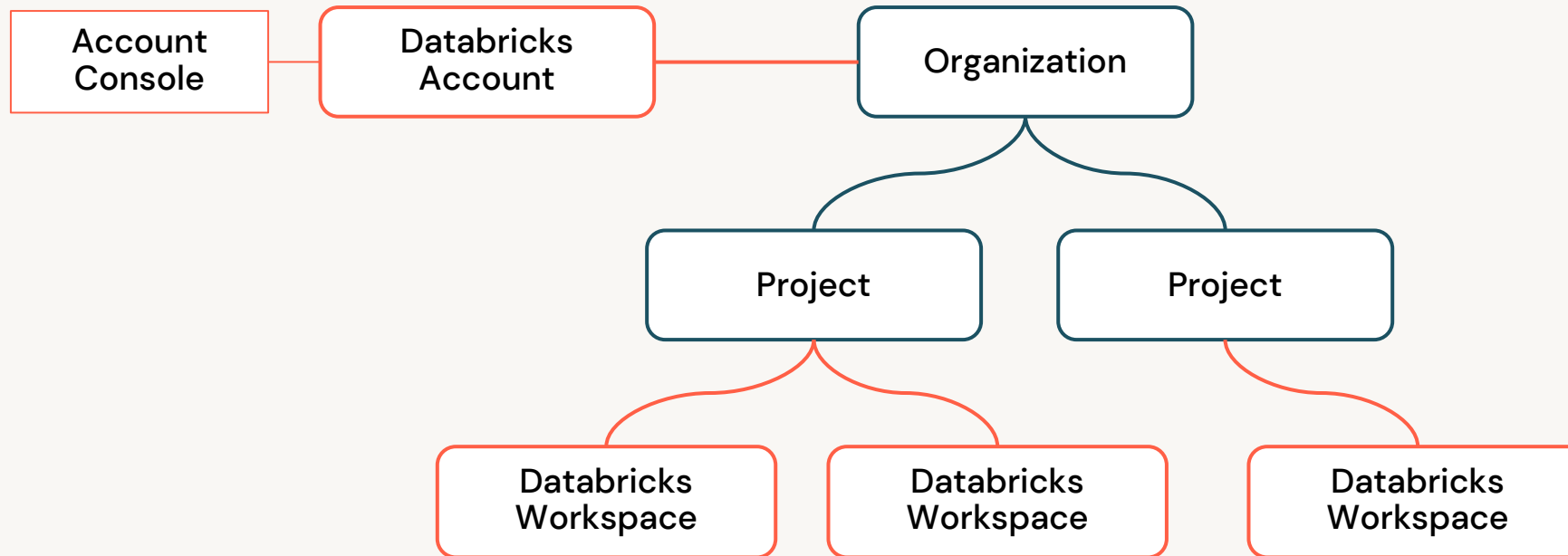
# Databricks Account: AWS



# Databricks Account: Azure



# Databricks Account: GCP

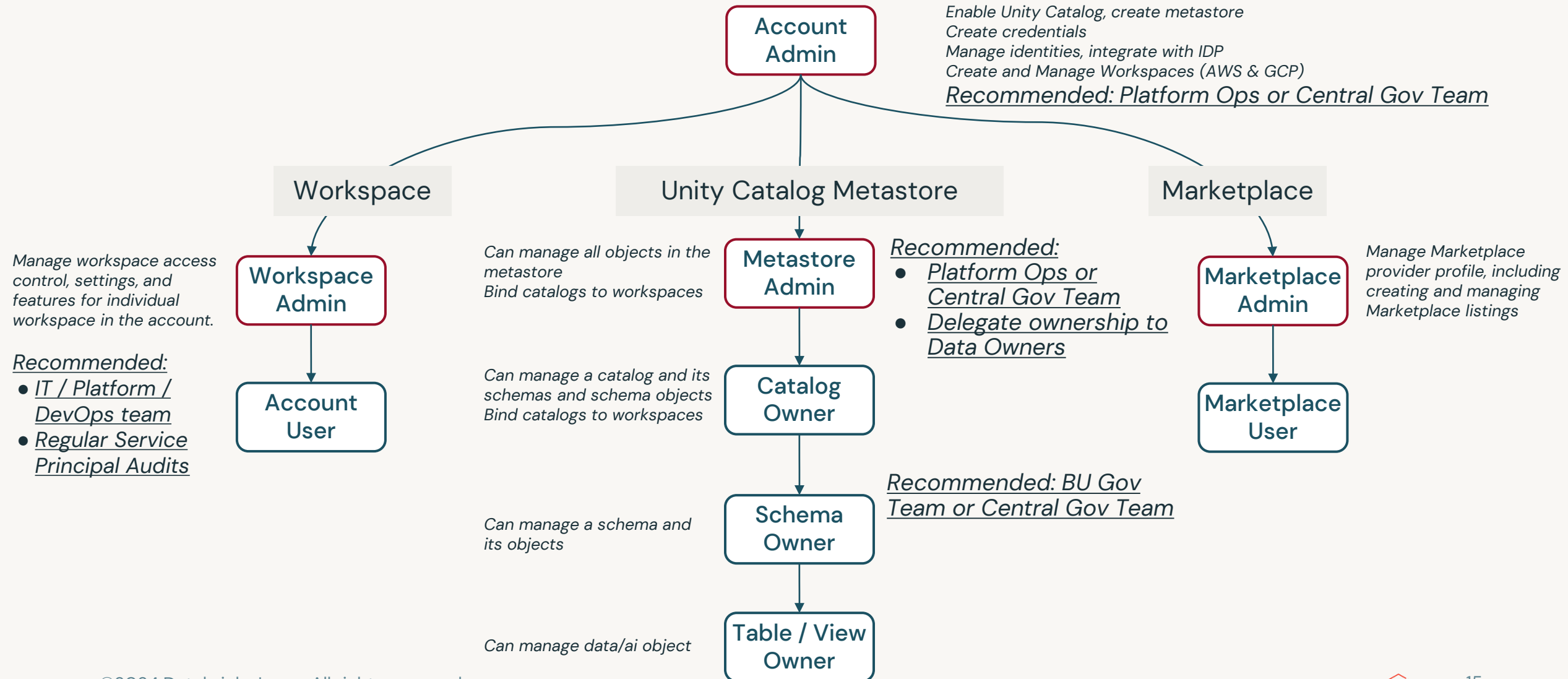


# Unity Catalog and Cloud Constructs

|                           | <b>AWS</b>      | <b>Azure</b>         | <b>GCP</b>      |
|---------------------------|-----------------|----------------------|-----------------|
| <b>Databricks Account</b> | Accounts        | Tenant               | Organization    |
| <b>Metastore</b>          | Region          | Region               | Region          |
| <b>Catalog</b>            | <i>Account*</i> | <i>Subscription*</i> | <i>Project*</i> |
| <b>Storage Location</b>   | S3 Bucket       | ADLS Account         | GCS Bucket      |
| <b>Credential</b>         | IAM Role        | Managed Identity     | Service Account |

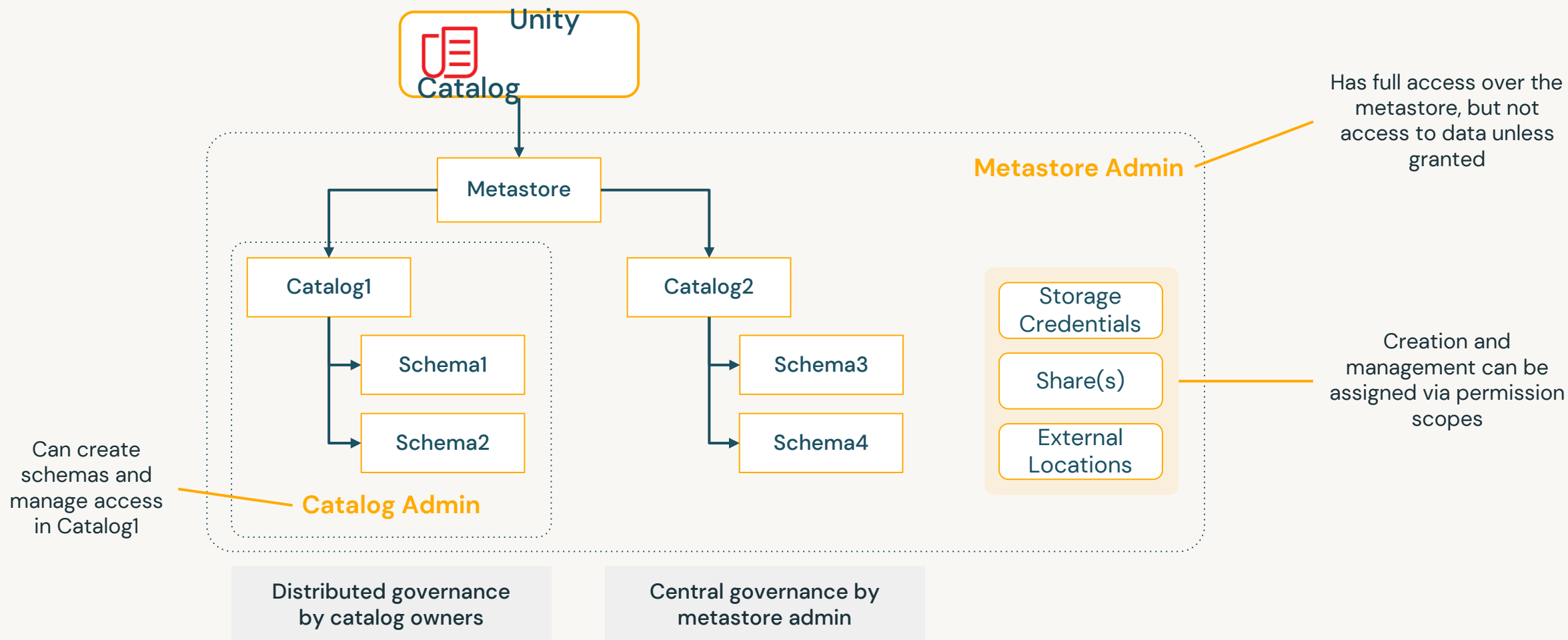
\* *Minimum one, more are optional*

# Databricks Account Roles Hierarchy



# Flexible Organizational Governance

Use centralized or distributed governance approaches



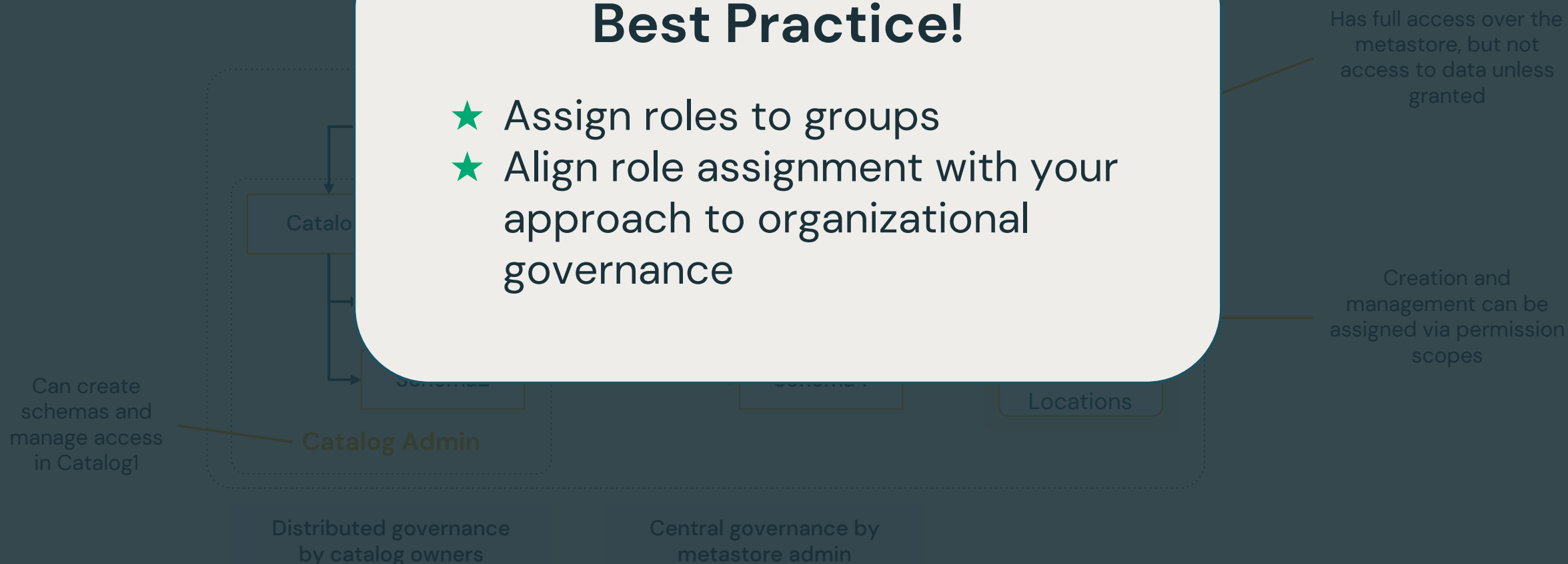


# Flexible Organizational Governance

Use centralized or distributed governance approaches

## Best Practice!

- ★ Assign roles to groups
- ★ Align role assignment with your approach to organizational governance



# Getting started with the Metastore

# Top Questions: Metastore Setup

- How do I set up a Metastore?
- What storage configurations do I need when setting it up?
- Are there best practices for configuration?



# Default / Automated Metastore Setup

1

UC is auto-enabled for the workspace

2

Dedicated catalog for the workspace

3

Catalog owned by workspace admins

4

No change to Hive support

Catalog Explorer

Catalog

Type to filter

> acme\_prod

> hive\_metastore

> samples

> system

Catalogs >

acme\_prod

Owner: \_\_workspace\_admins\_acme\_prod\_4819238477391440

Tags: Add tags

Comment: Add comment

Schemas Details Permissions

Filter schemas 2 schemas

| Name               | Created at      |
|--------------------|-----------------|
| default            | 2023-08-30 09:0 |
| information_schema | 2023-08-30 09:0 |



# Manual Metastore Setup

1  
Account Admin  
creates metastore in  
**Account Console**

Account Admin becomes **Metastore Admin** by default, and can update Admin to User or Group

Metastore can be **assigned to Workspaces** after Metastore Creation

2  
Region must be defined at creation and **only** workspaces in the same region can be attached

3  
Defining Storage Credential and Location for the Metastore is optional. If not set here, must be **set individually when creating each Catalog**

databricks Account

Catalog > Create metastore >

## Create metastore

1 Create metastore — 2 Assign to workspaces

\* Name

Region

Select the region for your metastore. You will only be able to assign workspaces in this region to this metastore.

S3 bucket path (optional) ?

Optional location for storing managed tables data across all catalogs in the metastore. Once configured this path can't be removed. [Learn more](#)

IAM role ARN (optional) ?

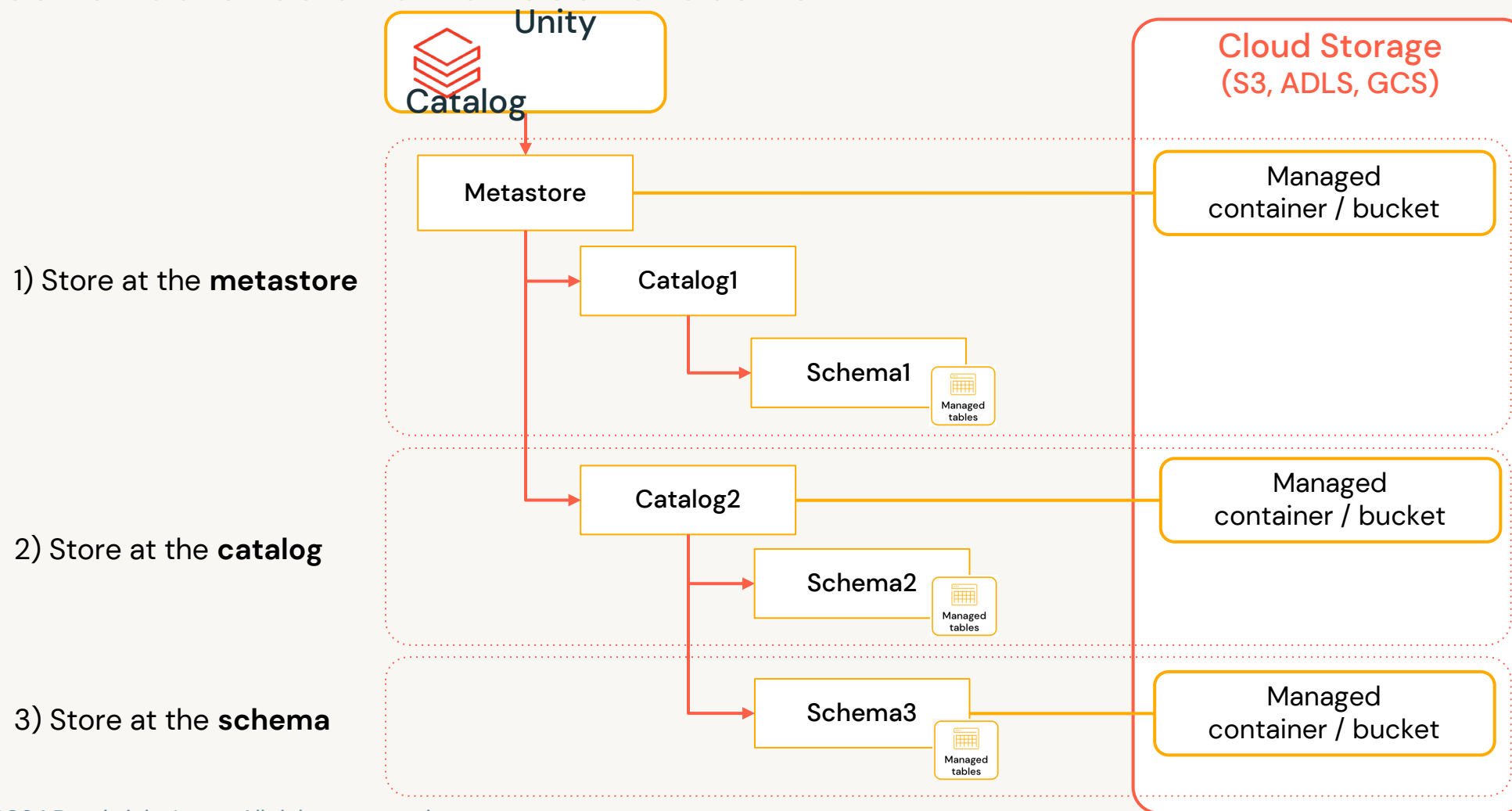
Enter the IAM role that Databricks will use to access the S3 bucket. [Learn more](#)

Create Cancel



# How to think of Managed Storage

Use for data isolation or cost allocation



**Note:**  
The lower location supersedes the higher: If, for example, both catalog and schema of a table have a managed location, then the table data is stored in the schema location

# How to think of Managed Storage

Use for data isolation or cost allocation

## Best Practice!

### ★ Use Catalog-Level Storage

*Gives flexibility for physical isolation between catalogs associated with different business units and/or development environments*

1) Store at the **metastore**

2) Store at the **catalog**

3) Store at the **schema**



**Note:**  
The lower location supersedes the higher: If, for example, both catalog and schema of a table have a managed location, then the table data is stored in the schema location

# Register data with Unity Catalog



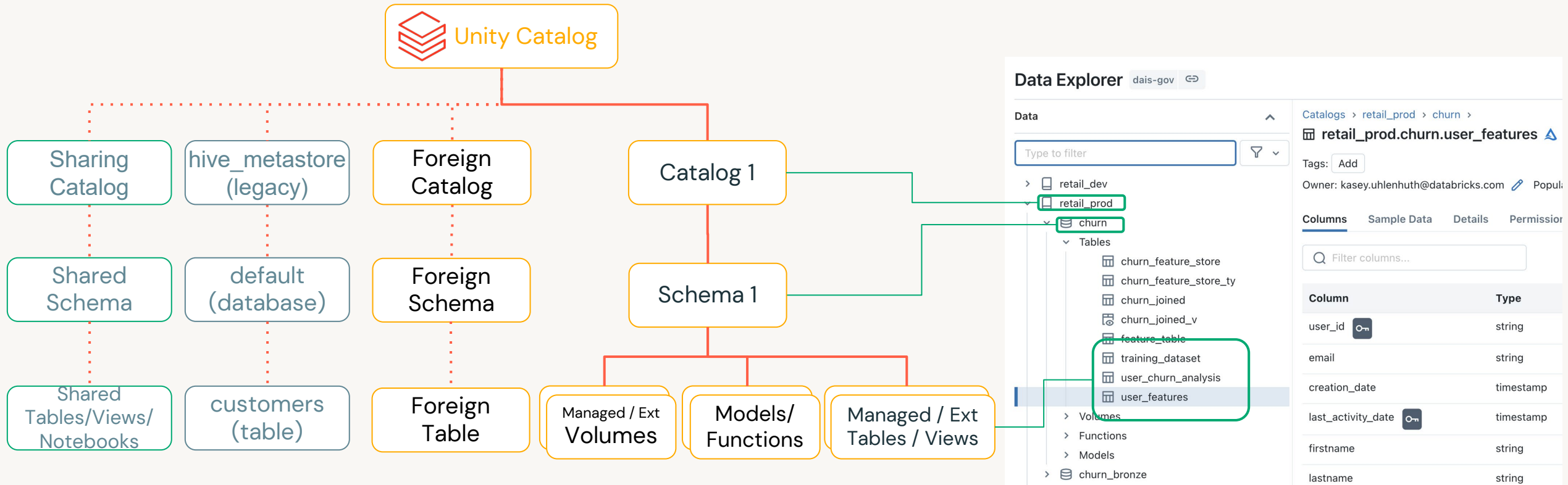
# Top Questions: Data Organization + Creation

- What objects can Unity Catalog secure?
- What permissions do I need to configure securable objects?
- How do I best organize data + ai assets (tables but also external data, files, models, etc)?



# Governed namespace across file and database sources

Access legacy metastore and foreign databases powered by Query Federation `



```
SELECT * FROM main.paul.red_wine; -- <catalog>.<database>.<table>
```

```
SELECT * FROM hive_metastore.default.customers;
```

```
SELECT * FROM snowflake_warehouse.some_schema.some_table;
```

# Data Assets

# Storage Credentials

## Cloud Credentials used by Unity Catalog

Storage credentials governed by access-control policies

CREATE STORAGE CREDENTIAL privilege on the Unity Catalog metastore attached to the workspace required. Account admins and metastore admins have this privilege by default.

Permission to create storage credentials should only be granted to users who need to define external locations

Create a new storage credential

×

A storage credential represents an authentication and authorization mechanism for accessing data stored on your cloud tenant. [Learn more](#)

\* Credential Type

Azure Managed Identity

\* Storage credential name

\* Access connector ID [Learn more](#)

User assigned managed identity

Comment

> Advanced Options

Create a new storage credential

×

A storage credential represents an authentication and authorization mechanism for accessing data stored on your cloud tenant. [Learn more](#)

\* Credential Type

AWS IAM Role

Copy from instance profile

\* Storage credential name

\* IAM role (ARN) [Learn more](#)

arn:aws:iam::account:role/role-name-with-path

Enter the ARN of the IAM role that has access to the S3 bucket

Comment

# External Locations

## Securable Object that combines a Storage Path with a Storage Credential

```
CREATE EXTERNAL LOCATION `s3-remote`  
URL 's3://us-east-1/location'  
WITH (STORAGE CREDENTIAL `s3-remote-cred`)  
COMMENT 'Default source for AWS external data'
```

CREATE EXTERNAL LOCATION **privilege on the Storage Credential required.**

A user or group with permission to use an external location can access any storage path within the location's path without direct access to the storage credential.

External Locations >

Create a new external location

An external location is a cloud storage url (and paired credential) that allows access to data stored on your cloud tenant. [Learn more](#)

Copy from mount point ▾

\* External location name

\* Storage credential [Learn more](#)

Select storage credential ▾

\* URL [Learn more](#)

Enter the bucket path that you want to use as the external location

Comment

> Advanced Options

# Volumes

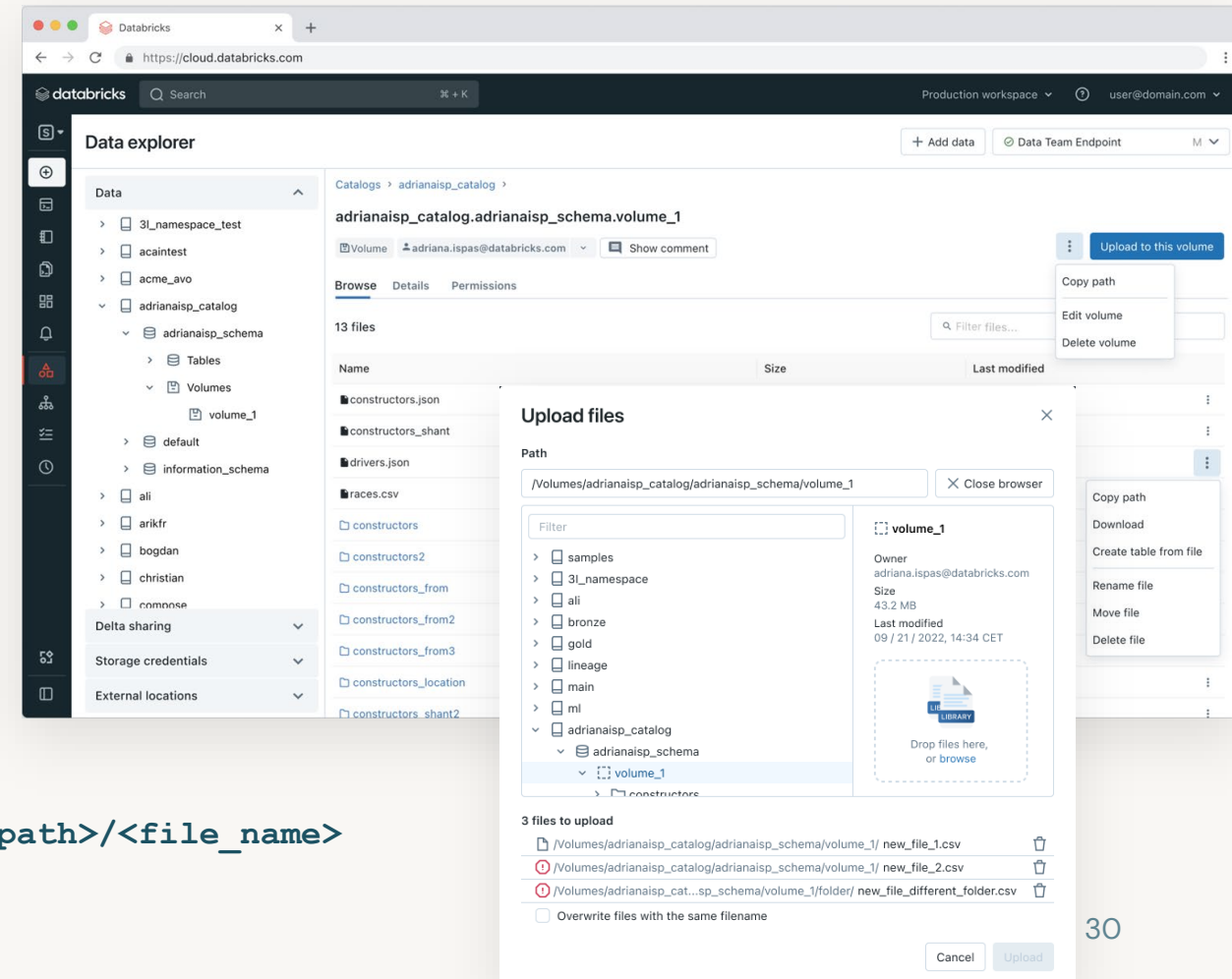
## Catalog collections of files in Unity Catalog

Access, store, organize and process **any file format** with Unity Catalog governance

~~CREATE VOLUME~~ privilege on the Catalog required.

Volumes can be managed or external, for external volumes, users also need the ~~CREATE EXTERNAL VOLUME~~ permission on the External Location

The path to access files in volumes uses the following format:  
`/Volumes/<catalog_name>/<schema_name>/<volume_name>/<path>/<file_name>`



# Volumes

## Catalog collections of files in Unity Catalog

```
CREATE EXTERNAL VOLUME IF NOT EXISTS
myCatalog.mySchema.myExternalVolume
COMMENT 'This is my example external volume'
LOCATION 's3://my-bucket/my-location/my-path'
```

~or~

```
CREATE VOLUME myManagedVolume
COMMENT 'This is my example managed volume'
```

Create a new volume

Volume name

Volume type

☒ Managed volume

☐ External volume

Comment (optional)

Create a new volume

Volume name

Volume type

☐ Managed volume

☒ External volume

External location

Select external location ▾

This external location will act as a source for your volume

Path

abfss://<container\_name>@<storage\_account\_name>.dfs.core.windows.net/<

The prefix where the volume will be created

Comment (optional)

# Comparison of Managed and Unmanaged UC Tables

## Consider the benefits of Managed tables

| Characteristic                     | Managed  | Unmanaged (a.k.a. “External”)   |
|------------------------------------|--|---|
| Table’s <b>Type</b> Property Value | “MANAGED”  | “EXTERNAL”  |
| DROP Table Behavior                | Discards metadata and <b>deletes</b> the associated data <ul style="list-style-type: none"><li>• UNDROP exists</li></ul> | Discards metadata only. Does not delete the data <ul style="list-style-type: none"><li>• If data needs to be deleted, must be done separately</li></ul> |
| Create Table Syntax                | <code>CREATE TABLE [&lt;catalog&gt;.] [&lt;schema&gt;.]&lt;table&gt; ...</code>  | <code>CREATE TABLE [&lt;catalog&gt;.] [&lt;schema&gt;.]&lt;table&gt; ...<br/>LOCATION 'abfss://cont@stacct.dfs.core.windows.net';</code>                |
| Data File Location                 | Whichever managed location has been specified and is found first: schema, catalog, metastore.                            | The path specified by the LOCATION keyword  |
| Performance Optimizations          | <b>Auto Tune</b> (Predictive Optimization, more coming)  | Manually managed  |
| Data Format Support                | DELTA  | DELTA, CSV, JSON, AVRO, PARQUET, ORC, TEXT  |
| Pros                               | New features, performance, simplicity, stricter access   | Non-delta tables, enforce storage naming, better compatibility with external readers/writers, simpler migration   |



# Comparison of Managed and Unmanaged UC Tables

Consider the benefits of Managed tables

| Characteristic                     | Managed (a.k.a. "Internal")                            | Unmanaged (a.k.a. "External")   |
|------------------------------------|--|---|
| Table's <b>Type</b> Property Value | MANAGED  | EXTERNAL  |
| DROP Table Behavior                | Deletes the data                                       | Does not delete the data; deleted, must be done separately  |
| Create Table Syntax                | <code>CREATE TABLE &lt;table&gt; ...</code>            | <code>CREATE TABLE &lt;table&gt; ... LOCATION &lt;location&gt;</code>   |
| Data File Location                 | Default location                                       | LOCATION keyword  |
| Performance Optimizations          | Delta Lake optimizations                               | None  |
| Data Format Support                | DELTA  | DELTA, CSV, JSON, AVRO, PARQUET, ORC, TEXT  |
| Pros                               | New features, performance, simplicity, stricter access | Non-delta tables, enforce storage naming, better compatibility with external readers/writers, simpler migration |

## Best Practice!

★ Lead with Managed Tables

*Use Managed tables when possible for best performance and simplicity. Open APIs and ongoing interoperability work will continue reducing need for external tables.*



# Lakehouse Federation

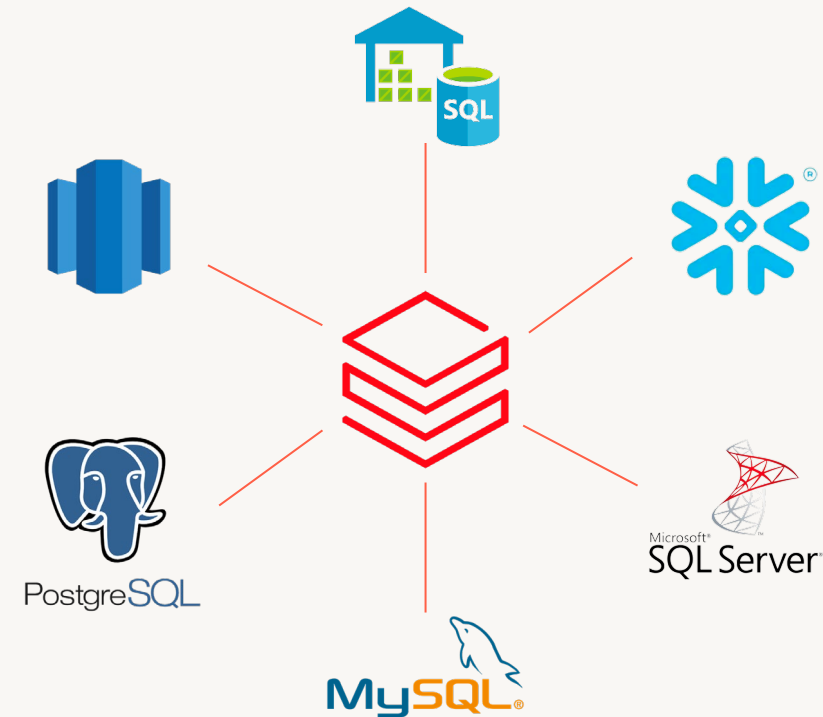
Discover, query, and govern all your data – no matter where it lives

Lakehouse Federation provides one **single point of secure access** to all your data with unified permission controls and intelligent **pushdown optimizations**.

**Supported sources GA:** MySQL, PostgreSQL, Redshift, Snowflake, SQL Server, Synapse, Databricks

**Supported sources Preview:** BigQuery, Hive, AWS Glue

**In roadmap:** Teradata, Oracle, SFDC



# Connections

## Securable Object Defining Path & Credentials for an External Database

```
CREATE CONNECTION your_connection_name
TYPE POSTGRESQL
OPTIONS (
  host 'qf-postgresql-demo...com',
  port '1234',
  user secret('secrets.r.us', 'your_username'),
  password secret('secrets.r.us', 'your_password'))
```

CREATE CONNECTION **privilege on the Unity Catalog metastore attached to the workspace required. Account admins and metastore admins have this privilege by default.**

Users with the `USE CONNECTION` privilege on the metastore see all connections. Otherwise, you can view only the connections for which you are the connection object owner or have some privilege

on. ©2024 Databricks Inc. — All rights reserved

### Create Connection

\* Connection name

\* Connection type

 PostgreSQL

\* Host

Port

\* User

\* Password

Connections >

### Create Connection

\* Connection name

\* Connection type

 Snowflake

\* Auth type

OAuth

Username and password

OAuth Access Token

✓ OAuth

\* User

\* Client ID

\* Client secret

# Foreign Catalog

## Read-only Mirror of Database in an External Data System

```
CREATE FOREIGN CATALOG IF NOT EXISTS my_foreign_catalog
USING CONNECTION your_connection_name
OPTIONS (database 'external_database_name')
```

CREATE CATALOG permission on the metastore required, and CREATE FOREIGN CATALOG privilege on the connection or must be the owner of the connection

You can query and manage access to data in a Foreign Catalog using Unity Catalog. Foreign catalog metadata is synced into Unity Catalog on each interaction with the catalog.

©2024 Databricks Inc. — All rights reserved

Create a new catalog

A catalog is the first layer of Unity Catalog's three-level namespace and is used to organize your data assets. [Learn more](#)

\* Catalog name

\* Type

Foreign

\* Connection

snowflake\_connection

\* Database

Comment

Test connection

Cancel

Create



# Tagging + AI generated Documentation

- Auto-generate concise and informative **table and column comments** for Unity Catalog
- Document your backlog of data assets with **missing documentation** in minutes
- **Tag** your data for easier discovery and to facilitate tag-based policies

The screenshot displays the Databricks Unity Catalog interface for the 'retention\_prod.sandbox.features' table. The left sidebar shows the catalog structure, including 'sandbox' and its sub-items 'features' and 'users'. The main panel shows the 'Overview' tab for the table, with a search bar for columns and a 'Hide AI suggestions' button. The table columns are listed with their types, AI-generated comments, and tags. The 'About this table' section on the right provides additional details about the table's ownership, format, and popularity.

| Column   | Type   | Comment  | Tags              | Mask |
|----------|--------|--|-------------------|------|
| user_id  | string | Unique identifier for the user, allowing tracking of user behavior and preferences. ✓                | saptype: Customer |      |
| event_id | string | Identifier for the event, allowing tracking of different events and their associated user actions. ✓ |                   |      |
| platform | string | Represents the platform or application where the user interacted with the event. ✓                   |                   |      |
| date     | string | Date when the user interacted ✓  |                   |      |

**About this table**

Owner: paul.roome@databricks.com

Data source format: [Delta](#)

Last Updated: 21 hours ago

Popularity: [View](#)

Size: 69.8KiB, 1 file

Tags: [Add tags](#)

Row filter: [Add filter](#)

**Comment**

The 'features' table captures user engagement data across different platforms. It includes information about the user's actions and interactions, such as the date of the event, the platform used, and the specific action taken. The table also includes details about the user's session, such as the URL they were on and any rescued data they may have encountered. This data can be used to understand user behavior patterns, identify popular features or areas of interest, and track user engagement over time. It can also be useful in predicting user churn or rescuing data for users in distress.



# Tagging + AI generated Documentation

- Auto-generate concise and informative **table comments** for Un
- Document your b with **missing doc** minutes
- **Tag** your data for to facilitate tag-ba

## Best Practice!

- ★ Add Tags During Object Creation
- ★ Generate / Add Comments Early

*Metadata information such as tags and comments are critical for data discovery and comprehension by consumers. This data is also available for auditing in system tables, and powers Databricks IQ.*

The screenshot shows the Databricks web interface. At the top, there's a search bar and user information. The main area displays a table's details. On the right, the 'About this table' section shows metadata: Owner (paul.roome@databricks.com), Data source format (Delta), Last Updated (21 hours ago), Popularity, and Size (69.8KIB, 1 file). Below this is a 'Tags' section with an 'Add tags' button. A 'Comment' section contains a detailed description of the 'features' table, explaining it captures user engagement data across platforms and can be used for understanding user behavior, identifying popular features, and predicting user churn. The bottom of the interface shows a table with columns like 'date', 'string', and 'Date when the user interacted'.

# AI Assets

# Feature Store

## Leveraging Data Foundation for Accelerated ML Development

You can use **any Delta table** in Unity Catalog **with a primary key** as a feature table for model training or inference.

~~CREATE TABLE~~ privilege on the Schema required.

Unity Catalog provides feature discovery, governance, lineage, and cross-workspace access.

You can use tags, which are simple key-value pairs, to categorize and manage your feature tables and features.

Microsoft Azure | databricks | Search data, notebooks, recents, and more... | dais-uc101-eastus

### Features

Any Delta table with a primary key can be used as a feature table. [Learn more](#)

iot | Unity Catalog | Hive Metastore | Catalogs: dev | Tag: department | Owned by me | Reset filters | Sort by: Relevance

| Table name                                      | Owner           | Online stores | Last written  | Tags            | Comment  |
|---|-----------------|---------------|---------------|-----------------|--|
| <a href="#">dev.iot.turbine_hourly_features</a> | @databricks.com |               | 3 minutes ago | department: MFG | These features are derived from the turbine_t... |



# Models

## MLFlow Model Registry in Unity Catalog

Discover, manage, organize and execute **models across workspaces** with Unity Catalog governance

`CREATE MODEL` privilege on the Schema required to register a new model.

Owners of the model are able to create new versions of registered models.

ML model versions in UC must have a model signature.

The screenshot displays the Unity Catalog interface. On the left, a sidebar shows the 'Catalog' structure with a tree view containing 'In my org', 'mfg\_sandbox', 'psp\_turbine\_iot', and 'Models (1)'. The 'Models (1)' section is expanded, showing 'dbdemos\_turbine\_maintenance'. The main panel shows the details for this model. At the top, the breadcrumb is 'Catalogs > mfg\_sandbox > psp\_turbine\_iot > dbdemos\_turbine\_maintenance'. Below this, there are tabs for 'Overview', 'Details', and 'Permissions'. The 'Overview' tab is active, showing a 'Description' field with a placeholder 'Add description'. Below the description, there is a 'Versions' section with a table. The table has columns: Status, Version, Time registered, Tags, Aliases, Registered by, and Comment. There is one row for 'Version 1' with a green checkmark status, registered on '2024-04-09 0...', and an alias '@ prod'.

| Status | Version   | Time registered | Tags | Aliases | Registered by | Comment |
|--------|-----------|-----------------|------|---------|---------------|---------|
| ✓      | Version 1 | 2024-04-09 0... |      | @ prod  | @...          |         |

# Functions

## Custom Functions Governed by Unity Catalog

Use **SQL-native syntax** to register custom functions to schemas

CREATE FUNCTION privilege on the Schema required.

Python UDFs are designed to provide the full expressiveness of Python directly within SQL functions, allowing for customized operations such as advanced transformations, data masking, and hashing.

The screenshot shows the Databricks Catalog Explorer interface. The left sidebar displays a tree view of the catalog structure: 'Catalog' > 'In my org' > 'mfg\_sandbox' > 'psp\_turbine\_iot' > 'Functions (3)'. The 'roll\_dice' function is selected. The main panel shows the function's details for 'roll\_dice' in the 'psp\_turbine\_iot' schema of the 'mfg\_sandbox' catalog. The function definition is `(rand() * 6)::INT + 1`. The function metadata table lists parameters, type, return type, language, and determinism. The details table lists the name, catalog name, schema name, SQL data access, and specific name.

| Function metadata |        |
|-------------------|--------|
| Parameters        |        |
| Type              | SCALAR |
| Return type       | INT    |
| Language          | SQL    |
| Deterministic     | False  |

| Details         |                 |
|-----------------|-----------------|
| Name            | roll_dice       |
| Catalog Name    | mfg_sandbox     |
| Schema Name     | psp_turbine_iot |
| Sql Data Access | CONTAINS_SQL    |
| Specific Name   | roll_dice       |

# Vector Search

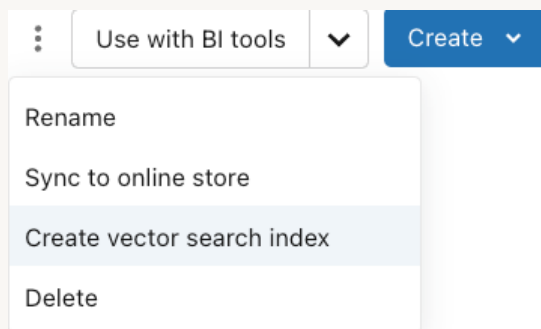
Create auto-updating vector indexes, managed by Unity Catalog

Choose your source table



| id | text                          | col1 | col2 |
|----|-------------------------------|------|------|
| 1  | The quick brown fox jumps ... |      |      |
| 2  | How quickly daft...           |      |      |
| 3  | The five boxing wizards...    |      |      |

Create semantic search index  
via Unity Catalog UI or via API



Choose any embedding model

## Model Serving

- Foundation Model API
- Custom model
- External model

Call endpoint for  
real-time retrieval

```
result = index.similarity_search(  
    query_text="What is Spark Connect?",  
    columns=["id", "text", "link"],  
    filters={"doctype": "wiki"})
```

- Ingestion pipelines managed for you
- Indexes managed by Unity Catalog
- Also, APIs for
  - Self-managed embeddings
  - CRUD API upsert/delete

- Integrate with LangChain, LlamaIndex, etc.
- Scale out endpoints as needed

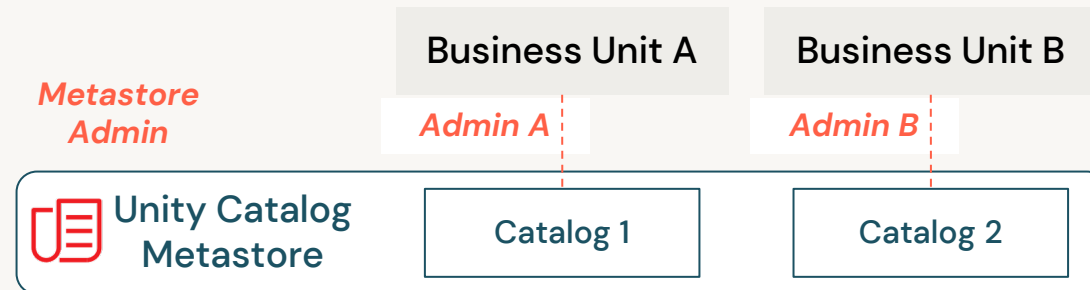


# Organizing Assets

# Unity Catalog Isolation features

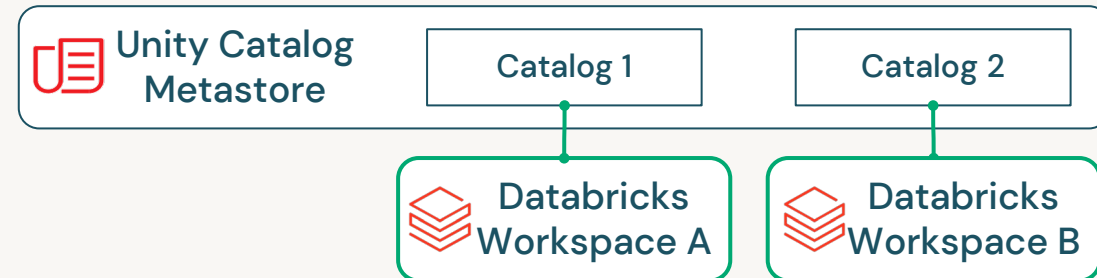
## Delegation of Management (admin isolation)

*Data should be managed by designated people/teams, based on the purpose/ownership of that data*



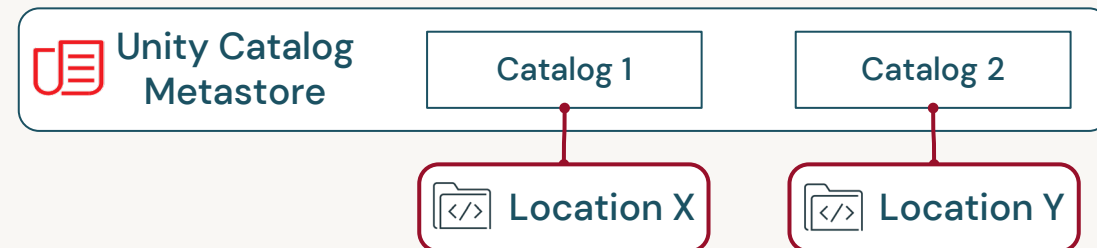
## Workspace to catalog binding

*Data should only be accessed in designated environments, based on the purpose of that data*



## Storage isolation

*Data should be physically separated in storage*



# Unity Catalog Isolation combined

## Delegation of Management (admin isolation)

Data should be managed by designated people / teams, based on the purpose/ownership of that data

## Workspace to catalog binding

Data should only be accessed in designated environments, based on the purpose of that data

- One catalog can be bound to multiple workspaces
- One Workspaces can be bound to multiple catalogs

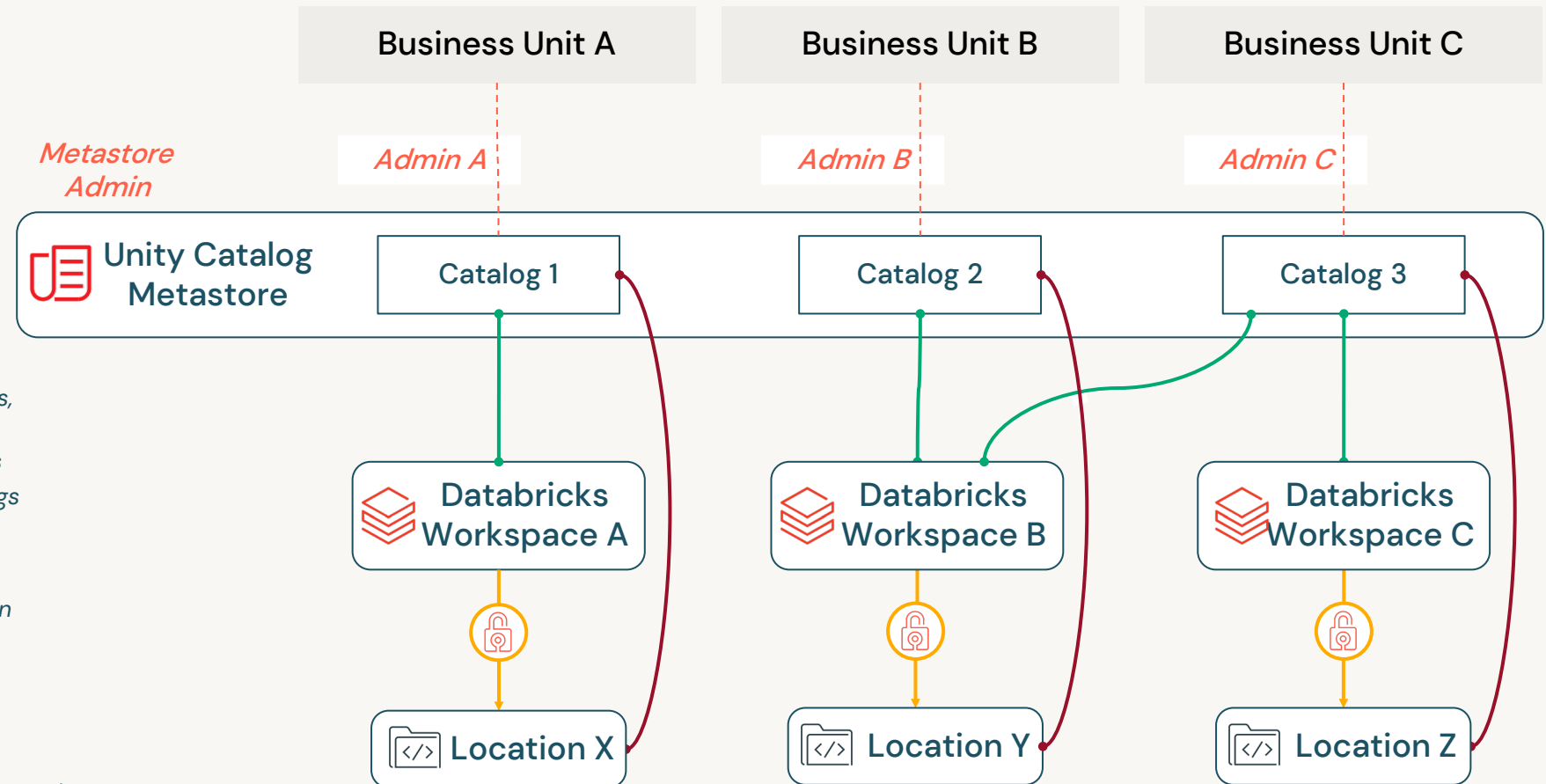
## UC Access Control

Users should only gain access to data/ metadata based on agreed access rules

## Storage isolation


Data should be physically separated in storage

"Location" is the unique combination of container/bucket + path



# Demo





# Discover your data with search and lineage



# Top Questions: Discovery

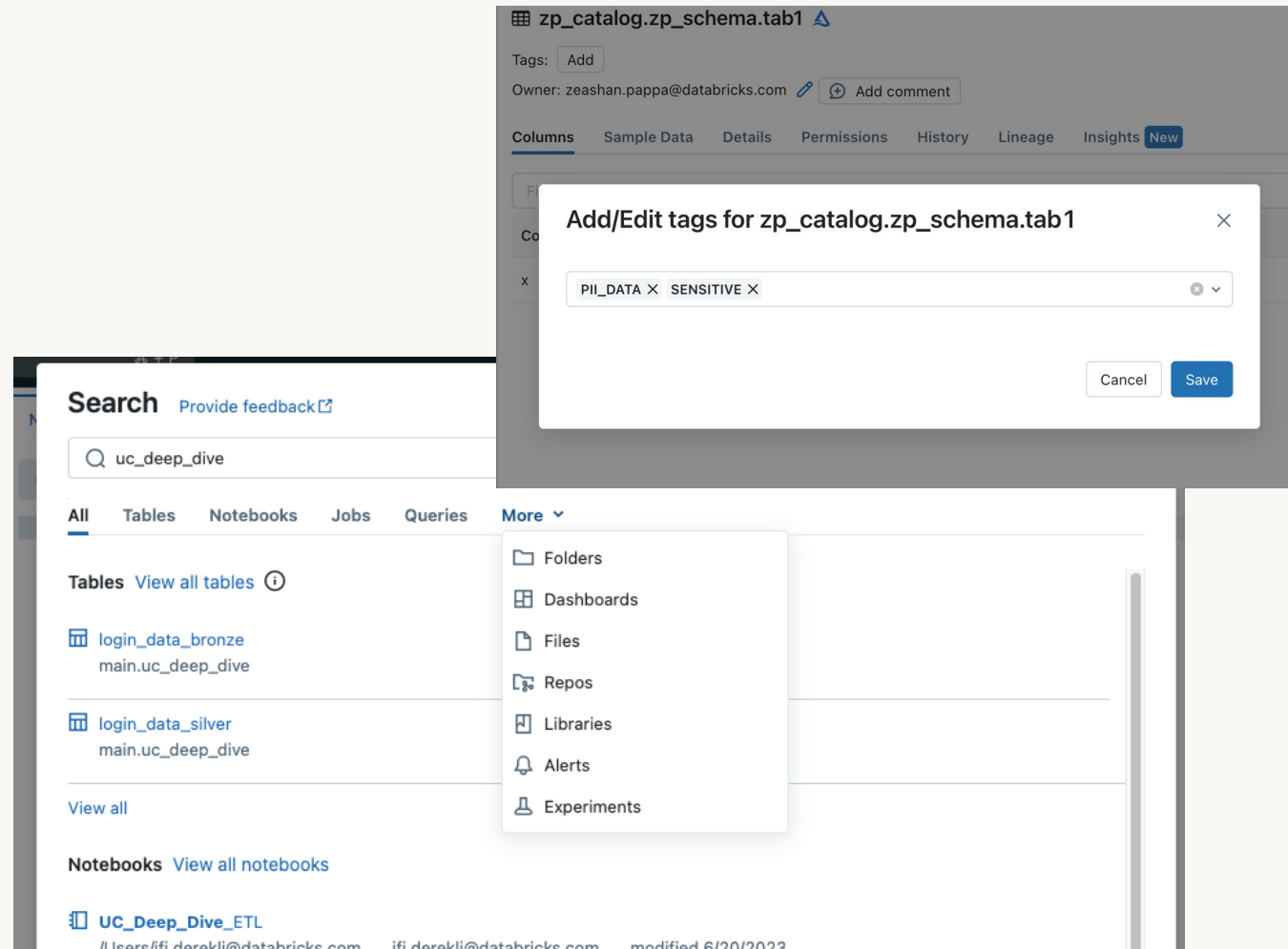
- How does UC help with data discovery and productivity?
- What should I be doing to enable my users to discover relevant data assets more effectively?
- What about existing catalogs or business glossaries that exist in my business?



# Built-in search and discovery

Accelerate time to value with low latency data discovery

- Unified UI to search for data assets stored in Unity Catalog
- Leverage common permission model from Unity Catalog
- Tag Column, Table, Schema, Catalog objects in UC
- Search for objects on tags
- ***Coming soon: request-for-access workflow integrated with Jira/ServiceNow!***



# Why is data lineage important?

## Compliance

- **Regulatory** requirements to verify data lineage
- Track the **spread of sensitive data** across datasets

## Discovery

- Understand **context** and **trustworthiness** of data before using it in analytics
- **Prevent duplicative** work and data

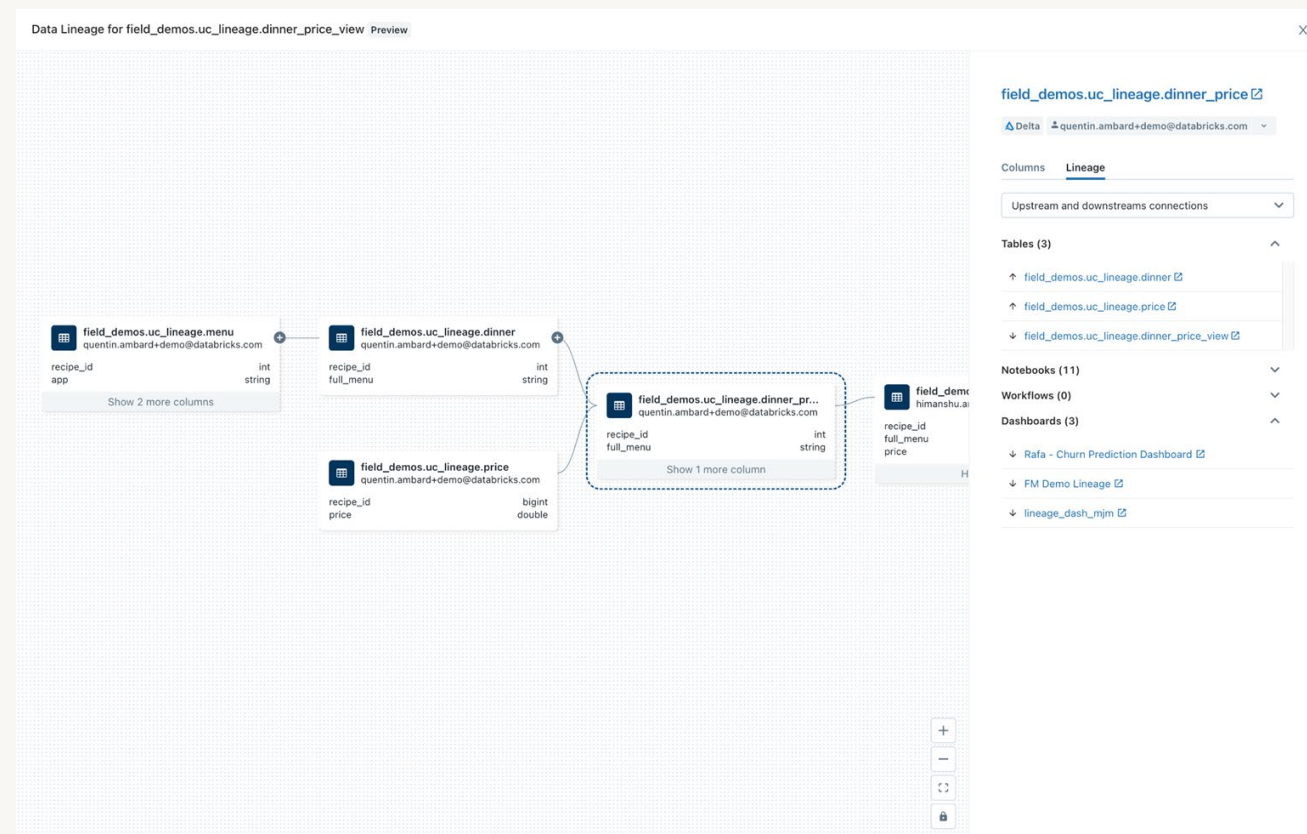
## Observability

- Track down **issues / discrepancies** in reports by tracing back the data
- Analyze **impact of proposed changes** to downstream reports e.g. column deprecation

# Automated lineage for all workloads

End-to-end visibility into how data flows and consumed in your organization

- Auto-capture runtime data lineage on a Databricks cluster or SQL warehouse
- Leverage common permission model from Unity Catalog
- Lineage across tables, columns, dashboards, workflows, notebooks, files, external sources, and models

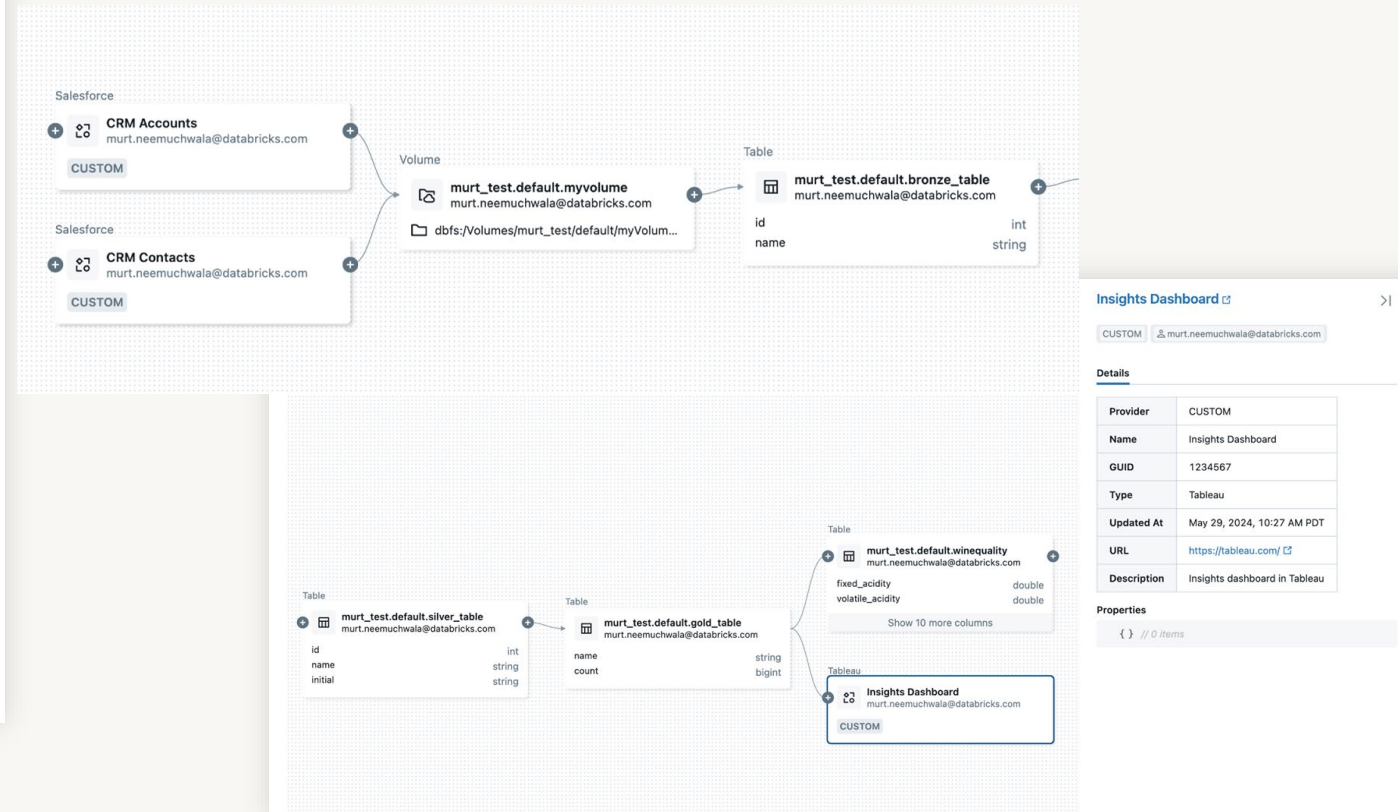


# Bring-your-own lineage

## Add custom lineage metadata to Unity Catalog today!

### Augment data lineage

- **First-mile lineage**
  - Specify upstream sources such as Kafka topics, SFDC objects, and their relationships with UC tables/paths
- **Last-mile lineage**
  - Specify downstream sources such as Tableau dashboards, PBI reports and their relationships with tables they query
- **Enrich existing lineage**
  - Add custom entities representing objects for which lineage was lost/not captured to connect broken lineage links

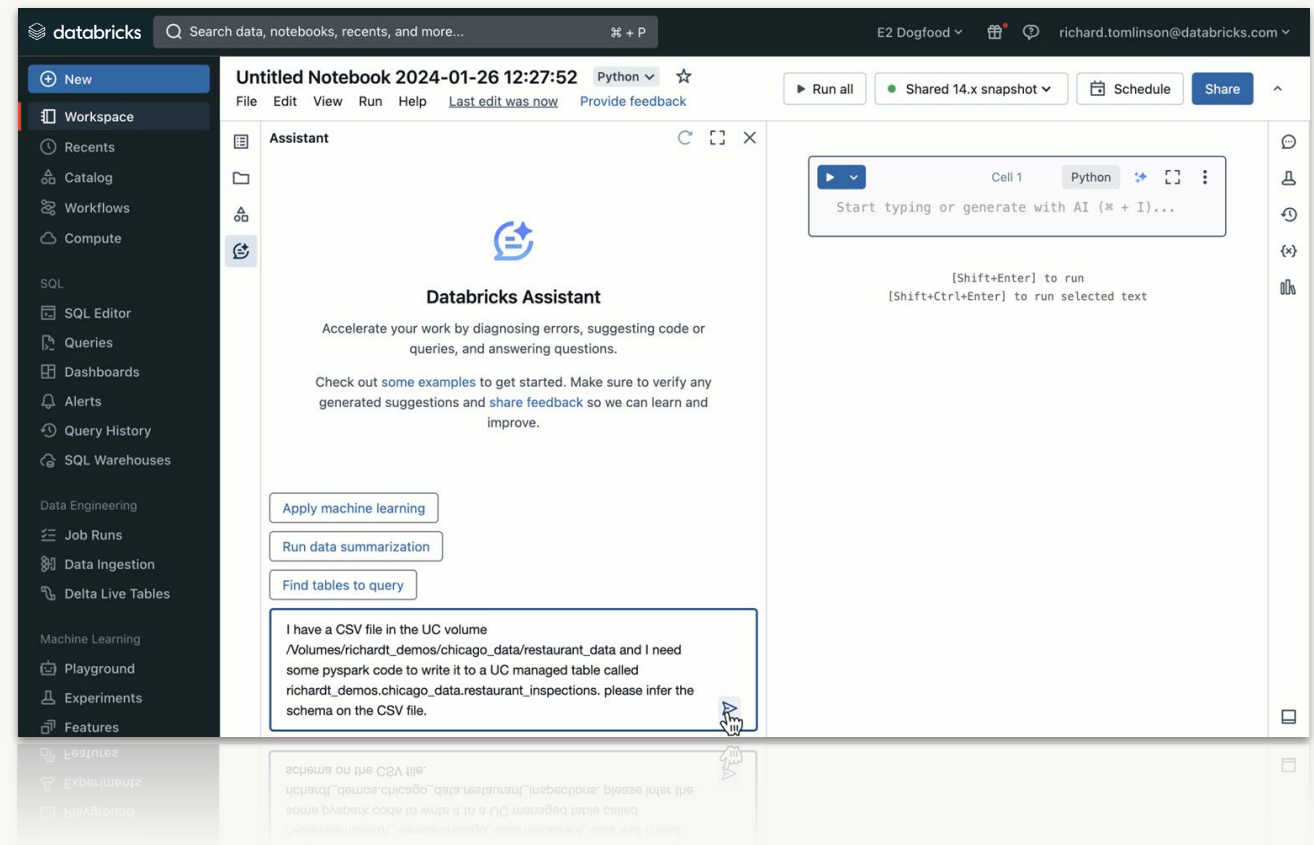


# Discovery + DatabricksIQ

## Example: Databricks Assistant, AI/BI Genie

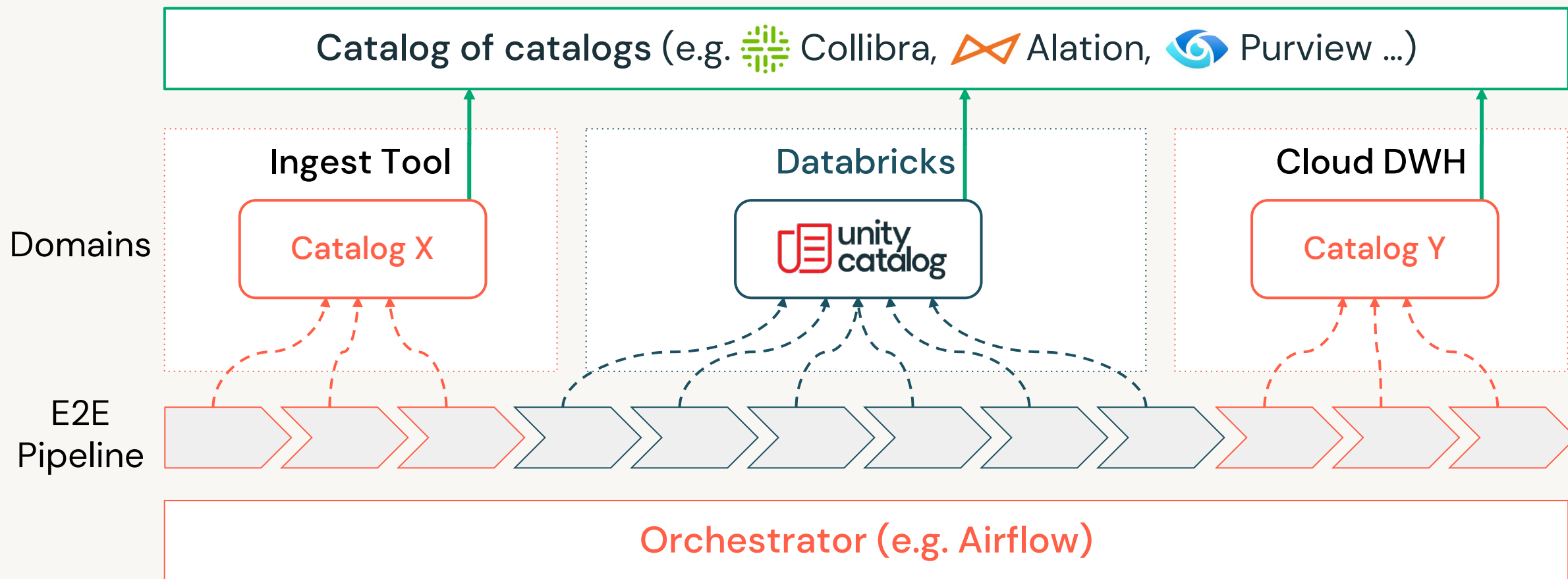
Increased productivity with contextual results relevant to your data assets using Unity Catalog and GenAI.

- Databricks Assistant
  - generates and auto-completes code and queries, native to Notebook, SQL Editor, File Editor
  - Explains and fixes issues
- AI/BI Genie
  - generates and executes SQL based on a Natural Language prompt
  - Enables *any* user to interact with the lakehouse data using Natural Language







# Enterprise Level Catalog integration



Lineage information flow:

-  Pipeline step sending lineage to domain's catalog (e.g. UC)
-  Domain's catalog to global catalog of catalogs

# Secure your Data



# Top Questions: Permissions

- How do I manage permissions on my data assets?
- How can I apply fine-grained permissions, like row-level filtering, column level masking, or attributed based controls?



# Centralized Access Controls

Centrally grant and manage access permissions across workloads and foreign databases

## Using ANSI SQL DCL

```
GRANT <privilege> ON <securable_type>  
<securable_name> TO <principal>
```

```
GRANT SELECT ON iot.events TO engineers
```

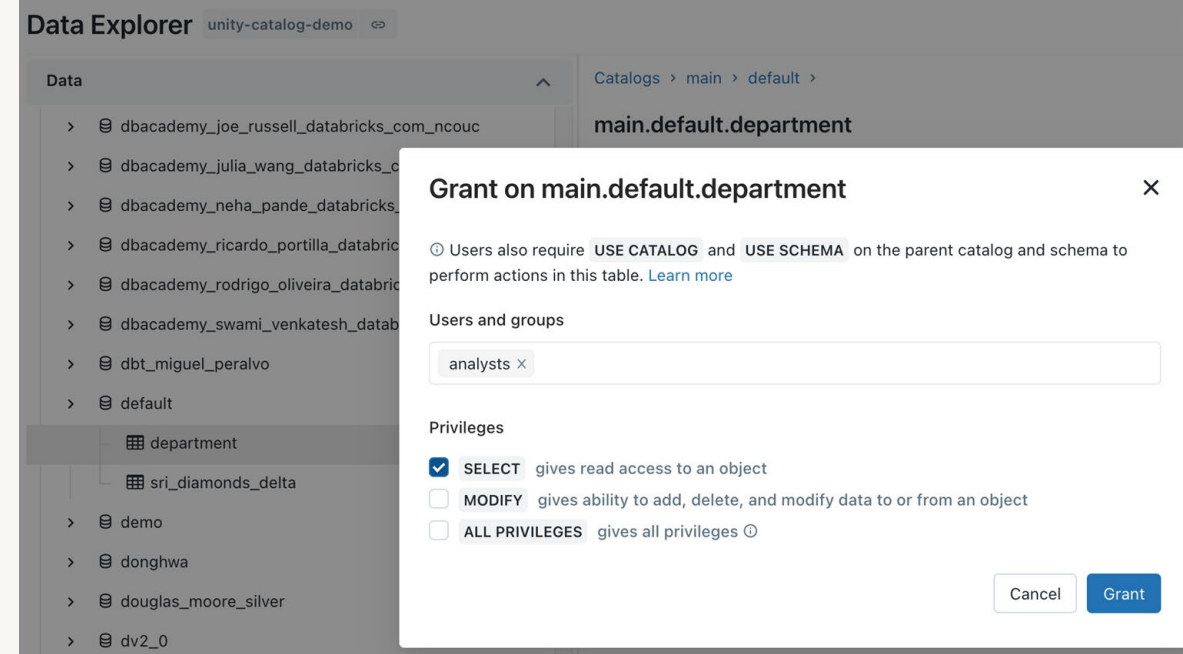
Choose  
permission level

'Table' = collection of  
files in S3/ADLS

Sync groups from  
your identity  
provider

*securable = catalog, schema, table, view,  
function, share, volume, model, etc*

## Using UI



# Row Level Security and Column Level Masking

Provide differential fine grained access to file based datasets and foreign tables

## Only show specific rows

```
CREATE FUNCTION <name> ( <parameter_name >  
<parameter_type> .. )  
RETURN {filter clause whose output must be a boolean}
```

```
CREATE FUNCTION us_filter(region STRING)  
RETURN IF(IS_MEMBER('admin'), true, region="US");
```

```
ALTER TABLE sales SET ROW FILTER us_filter ON region;
```

Test for group  
membership

Assign reusable  
filter to table

Specify filter  
predicates

## Mask or redact sensitive columns

```
CREATE FUNCTION <name> (<parameter_name>,  
<parameter_type>, [, <column>...])  
RETURN {expression with the same type as the first  
parameter}
```

```
CREATE FUNCTION ssn_mask(ssn STRING)  
RETURN IF(IS_MEMBER('admin'), ssn, "****");
```

```
ALTER TABLE users ALTER COLUMN table_ssn SET MASK  
ssn_mask;
```

Test for group  
membership

Assign reusable  
mask to column

Specify mask or  
function to mask

# Attribute based access controls

Scale access management

Scalable policies across all data + AI assets

Policies using tags, location, identity, & time attributes

SQL and UI based authoring

**New policy**

Policies allow you to manage access and data masking for your Unity Catalog assets. [Learn more](#)

< > View as code

---

**1. What should this policy apply to?**  
Choose the scope of the policy by choosing objects or tags.

Scope

pii\_us\_ssn X pii\_credit\_card X

Continue Cancel

**2. What should this policy do?**  
Policies can manage access to data or apply a data filter, such as a column mask.

**3. Who should this policy apply to?**  
Choose principals that will be affected by this policy.

**4. (Optional) Conditions**  
Only enforce this policy under certain conditions, such as region or a time limit.

**Summary**

**New policy 1** [Edit](#)

Add a description...

Details of your policy will appear here as

---

**New policy**

Policies allow you to manage access and data masking for your Unity Catalog assets. [Learn more](#)

< > View as code

---

**1. What should this policy do?**  
Policies can manage access to data or apply a data filter, such as a column mask.

**2. Who should this policy apply to?**  
Choose principals that will be affected by this policy.

**3. What should this policy apply to?**  
Choose the scope of the policy by choosing objects or tags.

**4. (Optional) Conditions**  
Only enforce this policy under certain conditions, such as region or a time limit.

**Enforce when...**

1 Region is US West X

AND OR

2 Object name starts with restricted\_ X +

Save & enforce Back



# Attribute based access controls

## Scale access management

### DAIS Session Alert!

- ★ Attribute-Based Access Controls in Unity Catalog – Building a Scalable Access Management Framework

**Attend today's session at 2:50 pm** by Zeashan Pappa (Staff Product Manager, Databricks), Kristen Wilder (Product Manager, Databricks) for a deep dive!



Access and data

apply to?  
by choosing

credit\_card

data or appl

y to?  
restricted by

action control

2 Object name starts with restricted\_ X

Save &amp; enforce

Back

# High Leverage Governance with Terraform & APIs

Use data-sec-ops, policies as code patterns to scale your efforts

- Privileges for UC objects can be managed programmatically using our Terraform provider, especially for teams already using Terraform
- This will pair naturally with the management of the UC objects (Metastore, Catalog, Assignments etc.) themselves.

(If not already using Terraform, maybe now is a good time!)

[Documentation](#) > [Data governance guide](#) > [What is Unity Catalog?](#) >  
[Automate Unity Catalog setup using Terraform](#)

## Automate Unity Catalog setup using Terraform

March 10, 2023

You can automate Unity Catalog setup by using the [Databricks Terraform provider](#). This article shows one approach to deploying an end-to-end Unity Catalog implementation. If you already have some Unity Catalog infrastructure components in place, you can also use this article to deploy additional Unity Catalog infrastructure components as needed.

For more information, see [Deploying pre-requisite resources and enabling Unity Catalog](#) in the Databricks Terraform provider documentation.

```
resource "databricks_grants" "sandbox" {  
  provider = databricks.workspace  
  catalog = databricks_catalog.sandbox.name  
  grant {  
    principal = "Data Scientists"  
    privileges = ["USAGE", "CREATE"]  
  }  
  grant {  
    principal = "Data Engineers"  
    privileges = ["USAGE"]  
  }  
}
```

# Demo



# Open data sharing powered by Unity Catalog



# Top Questions: Data Sharing

- How does Unity Catalog help me share data within and outside my organization simply, quickly, and securely?



# Data sharing & collaboration

Accelerate innovation and open new business practices



## Delta Sharing

Open sharing  
between  
organizations



## Databricks Marketplace

Open Marketplace  
for all your data, AI,  
and applications

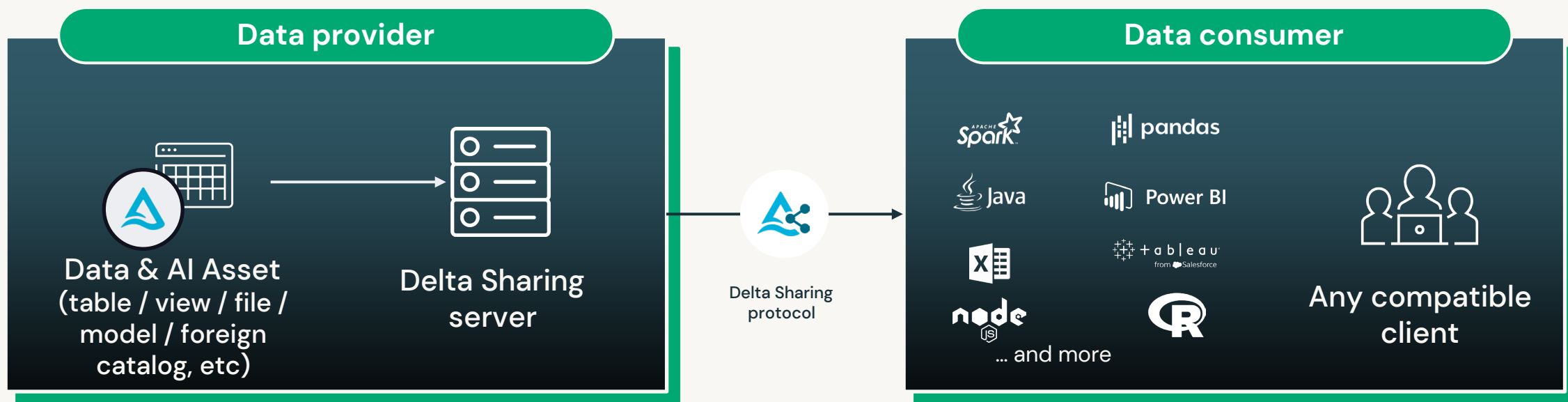


## Databricks Clean Room

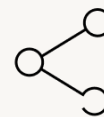
Privacy-safe  
computing and  
collaboration

# Delta Sharing

An open standard for secure sharing of tables, views, files, models, and more



Share cross-platform w/ open protocol



Share data with no replication

# Databricks Marketplace

An open marketplace for all data assets



# Databricks Clean Rooms

Privacy-safe collaboration for data and AI

- Flexible, Native Support for ML with Python
- Cross-Platform, Cross-Cloud Support
- Manage Complex Workloads at Scale

The screenshot displays the Databricks Clean Rooms configuration interface. It includes a form for setting up a clean room with the following fields:

- \* Clean Room name**: A text input field.
- Select a cloud and region**: A section with the instruction "This will be the cloud and region for your secure central clean room station." It contains three radio buttons for AWS, GCP, and Azure, and a dropdown menu for the region, currently set to "us-west-2".

Below the configuration form, the interface shows a breadcrumb trail: "Catalog Explorer > Clean Rooms > currency-14 > media\_measurement\_rf\_advanced". It also indicates "Managed by: wayne-enterprise" and has tabs for "Preview" and "My recent runs".

The main content area shows a step indicator "0/ Import Measurement Co's private library" and a code snippet: `python_wheel_name = 'currency_identity_package-0.0.1-py3-none-any.whl'`.

Below this, another breadcrumb trail is shown: "Catalog Explorer > Clean Rooms > brickready > Add assets". The "Add assets" section has a sub-header "Select assets across catalogs and schemas" and a search bar containing "bigquery".

On the left, a tree view shows the asset selection process:

- For you | All
- bigquery\_demo
  - default
  - schema
    - diamonds (selected with a checkmark)

On the right, a table lists the selected assets:

| Type  | Name  |
|-------|---|
| Table | bigquery_demo.schema.diamonds (via BigQuery)  |
| Table | snowflake_demo.myschema.zones (via Snowflake) |

At the bottom, a step indicator shows "2/ Load tables into local dataframes".

# Audit your data

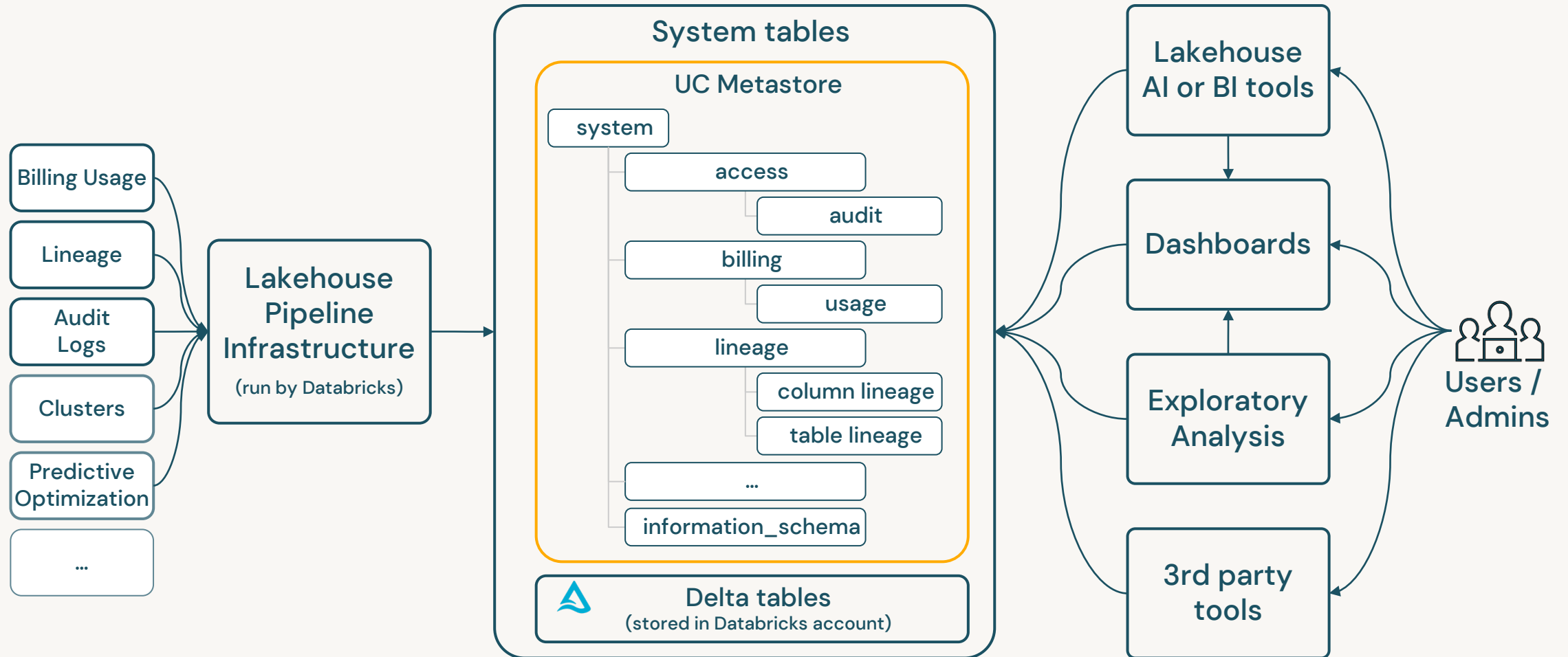
# Top Questions: Audit

- How do I answer questions about usage patterns, configuration changes, and so on, regarding my lakehouse?
- How do I monitor usage and costs, and ensure I'm staying on budget?
- How do I monitor data and model quality over time?



# Lakehouse Observability

Powered by Unity Catalog's System Tables





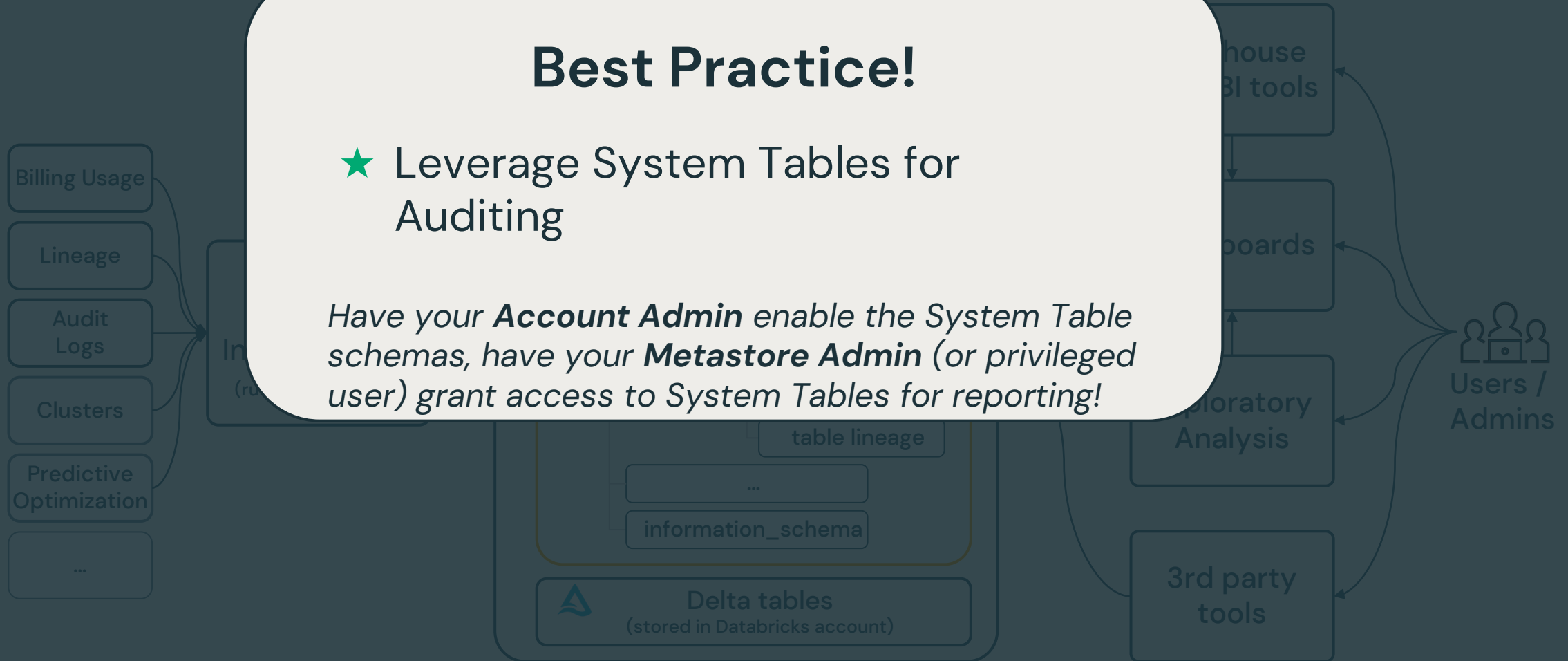
# Lakehouse Observability

Powered by Unity Catalog's System Tables

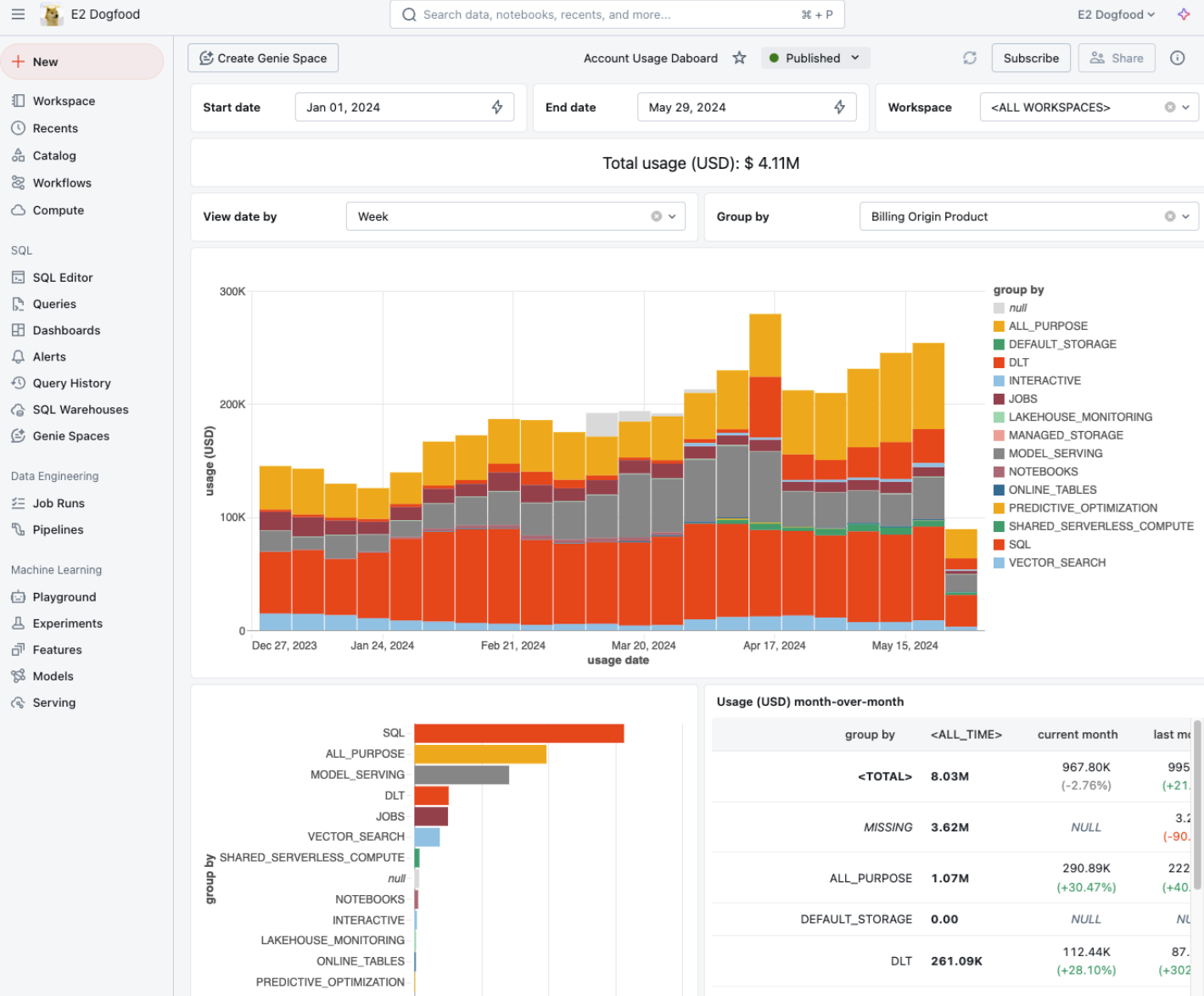
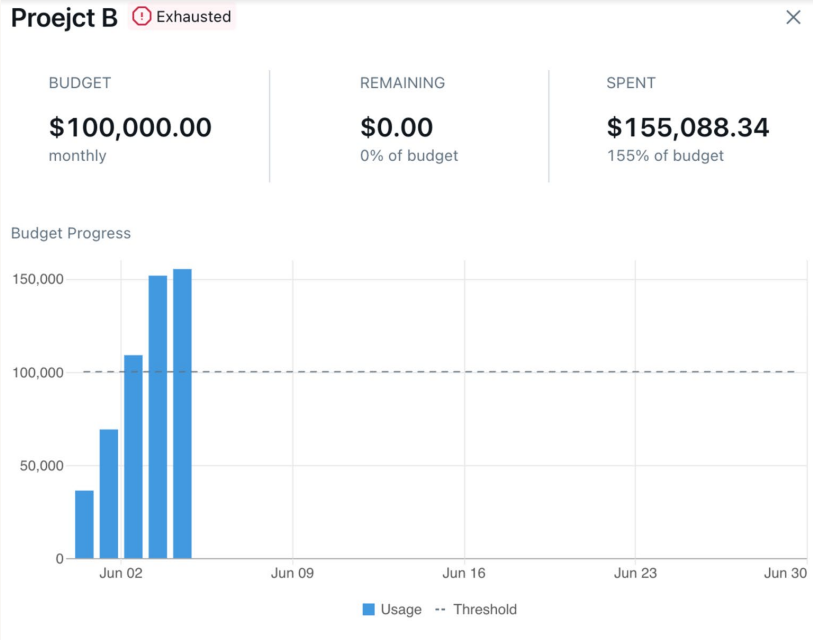
## Best Practice!

- ★ Leverage System Tables for Auditing

*Have your **Account Admin** enable the System Table schemas, have your **Metastore Admin** (or privileged user) grant access to System Tables for reporting!*



# Observability Dashboards and Budget Alerts



# Observability Dashboards and Budget Alerts

## DAIS Session Alert!

★ Lower TCO and Increased ROI:  
Managing Your Databricks Costs

*Attend today's session at 4:00 pm by Greg Kroleski (Product Manager, Billing, Databricks) for a deep dive into getting started with system table backed observability dashboards and budget alerts!*



# Lakehouse Monitoring

Unified monitoring for reliable, insightful, and simple data-to-AI pipelines



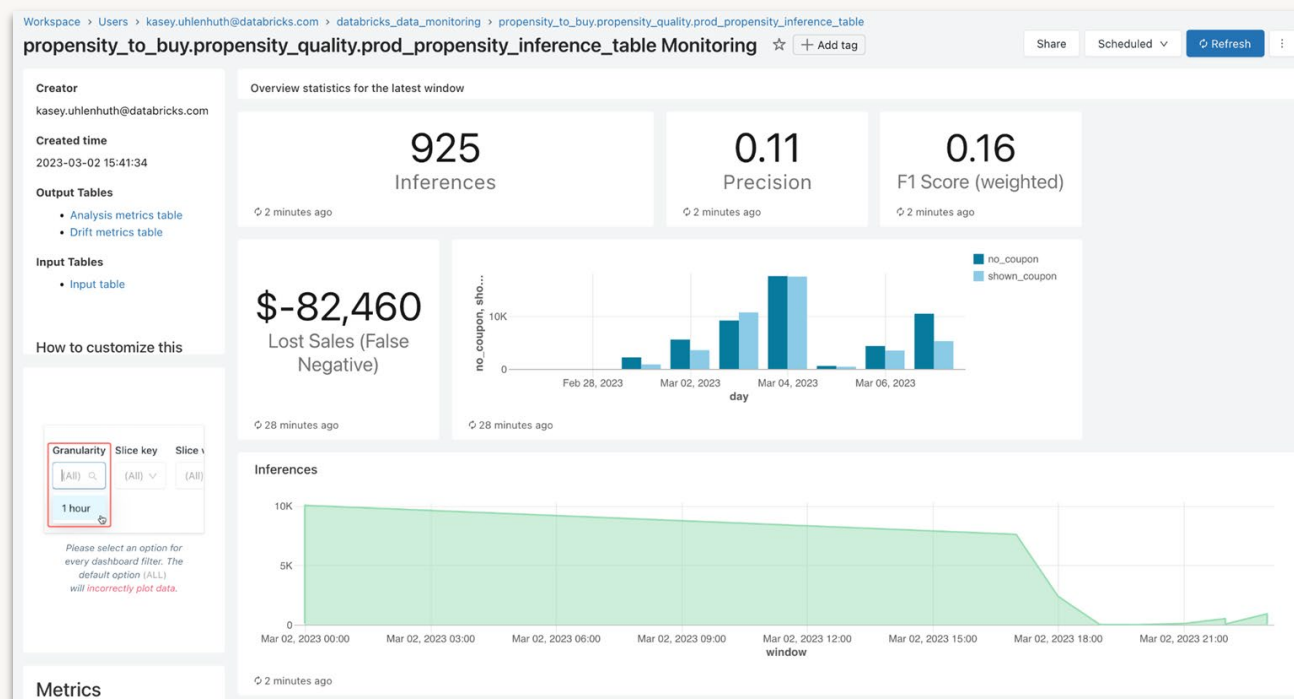
**Simple:** Log inference tables automatically, and generate metric tables and SQL dashboards.



**Proactive:** Automate alerts on table quality and custom metrics, and diagnose data or model issues.

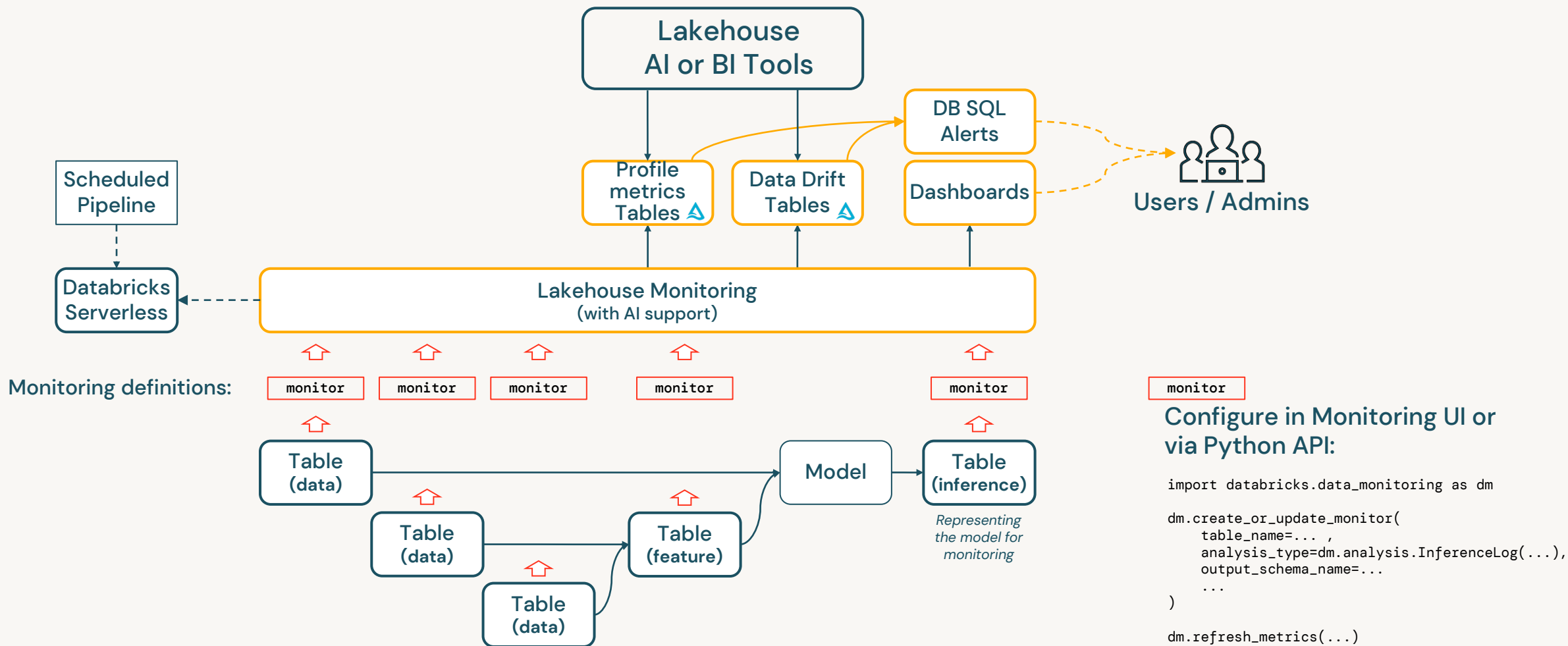


**Integrated:** Track end-to-end lineage in the Unity Catalog for training data, feature tables, models, and inference logs, for simpler governance.



# Lakehouse Monitoring

## Open monitoring for tables and models



# Demo



# Upgrade to Unity Catalog

# Top Questions: Upgrade

- If I've been using Databricks already without Unity, how do I upgrade to Unity Catalog quickly?





# UCX


## Unity Catalog Upgrade Framework

- Databricks Labs Project
- >500 Active Customers
- [github.com/databrickslabs/ucx](https://github.com/databrickslabs/ucx)
- Functionality:
  - Assessment
  - Group Migration
  - Cloud Infrastructure
  - Table Migration
  - Code Migration

[README](#) [License](#)

### Databricks Labs UCX

# UCX



## Automated upgrade to Unity Catalog

The companion for upgrading to Unity Catalog.




After [installation](#), ensure to [trigger](#) the [assessment workflow](#), so that you'll be able to [scope the migration](#) and execute the [group migration workflow](#).

The [README notebook](#), which can be found in the installation folder contains further instructions and explanations of the different ucx workflows & dashboards. Once the migration is scoped, you can start executing the [table migration workflow](#).

More workflows, like notebook code migration are coming in future releases.

UCX also provides a number of command line utilities accessible via `databricks labs ucx`.

For questions, troubleshooting or bug fixes, please see our [troubleshooting guide](#) or submit [an issue](#). See [contributing instructions](#) to help improve this project.

 build  codecov 89%  lines of code 51.5K



# UC Upgrade Best Practices

1. Assess/Inventorize (UCX)
  - a. If help needed – Account Team, select Partners well versed to assist
2. Migrate Groups (UCX)
3. Attach Metastore (UCX)
4. Migrate your external tables (UCX)
5. Migrate SQL Warehouses (UCX)
6. Migrate Jobs (UCX)
7. Migrate Managed Tables (UCX)
8. Migrate Code/Jobs/Notebooks (partially UCX)



# UC Upgrade Best Practices

## DAIS Session Alert!

- ★ Upgrading to Unity Catalog With Ease, Using UCX

*Re-watch Tuesday's session by Liran Bareket, Unity Catalog Product Specialist for a deep dive into the UCX tools and the process for upgrading!*



8. Migrate Code/Jobs/Notebooks (partially UCX)



# Architectural Patterns

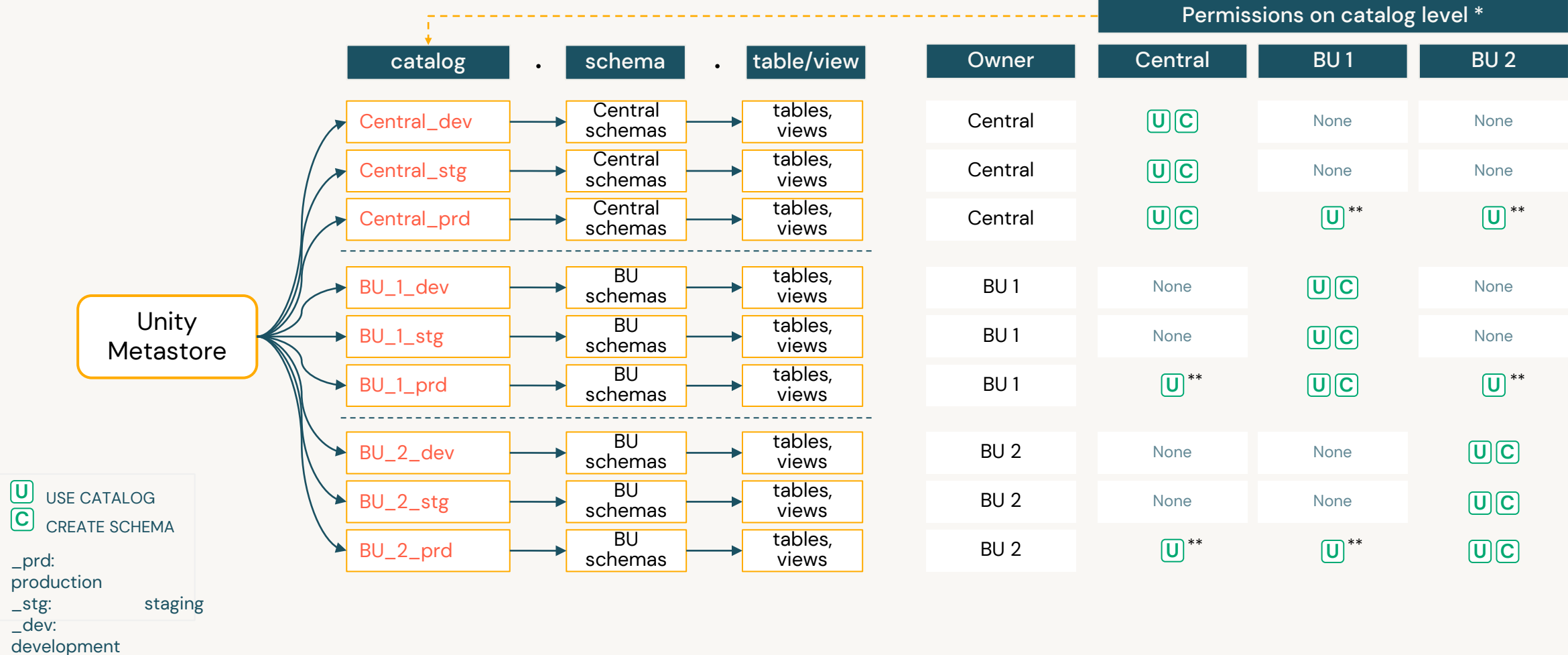
# Top Questions: Architecture

- How many catalogs should I configure?
- What are common access patterns to catalogs?
- What data isolation controls do I have and should I use?
- How should I design my multi-cloud or multi-region deployment?



# Practical Permissioning Example

## Catalog Organization and Ownership

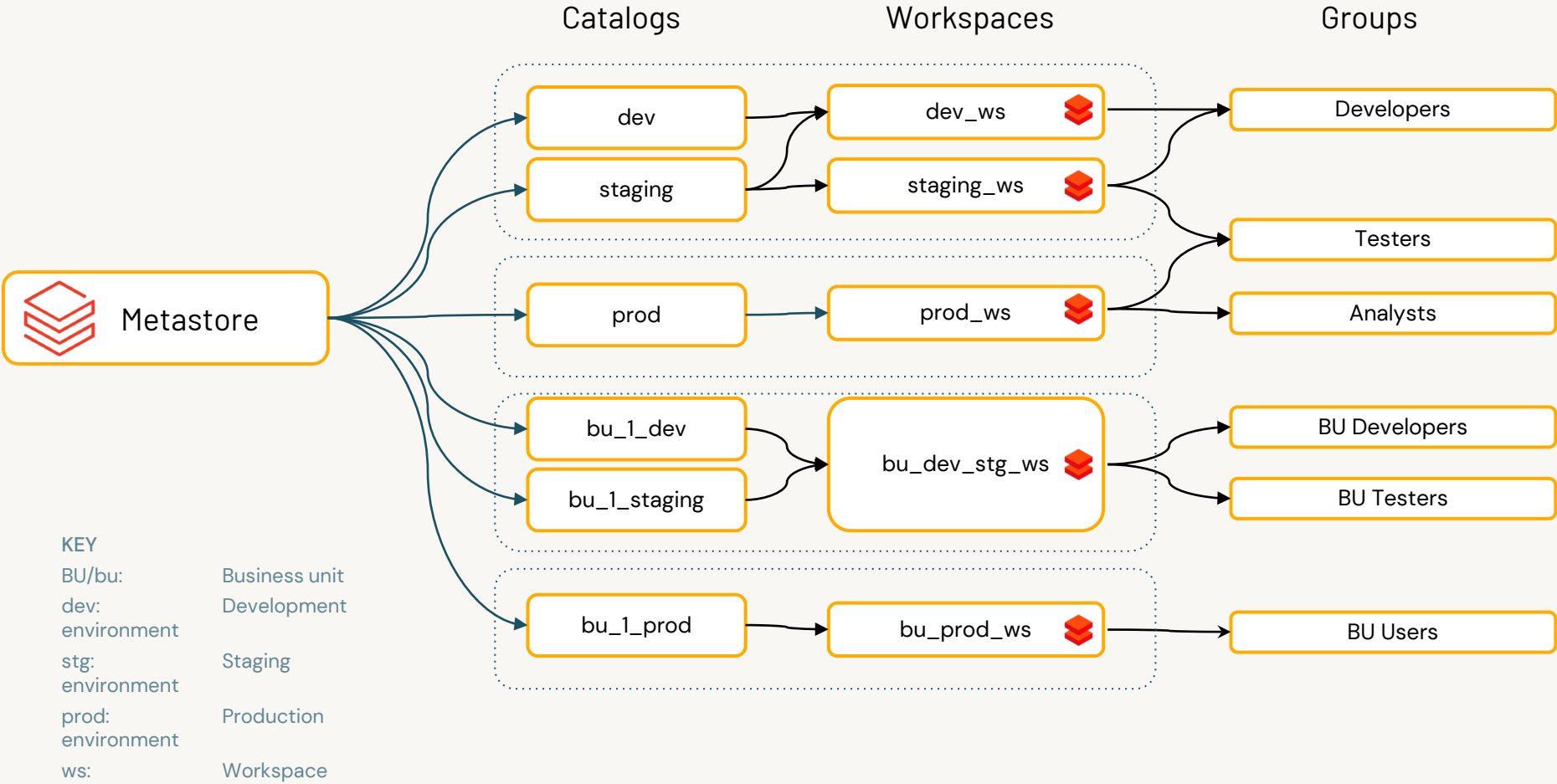


\* To access tables, USE SCHEMA and SELECT also need to be provided on schema and table level  
\*\* Catalogs where other BUs can be given access to schemas and tables/views by Central team



# Access data from specified environments only

Restrict catalog access by environment or purpose



Access to data and availability of data can be isolated across workspaces and groups



# Access data from specified environments only

Restrict access to data

## Best Practice!

### ★ Separate Catalogs by Responsibility

- Development Environment (dev / qa / prod)
- Business Unit (IT, Finance, Sales, etc.)
- Sharing / Foreign Databases

### ★ Use Workspace-Catalog Binding where required

*Create the minimum catalogs required to meet your team's isolation and ownership requirements.*

Access to data  
and availability of  
data can be  
isolated across  
workspaces and  
groups

KEY

BU/bu:

dev:

environment

stg:

environment

prod:

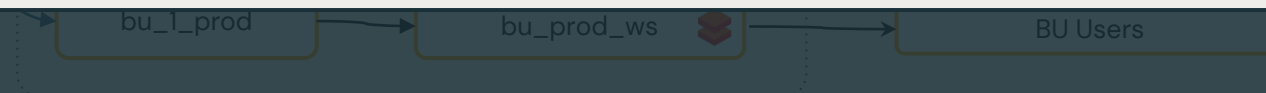
environment

ws:

Staging

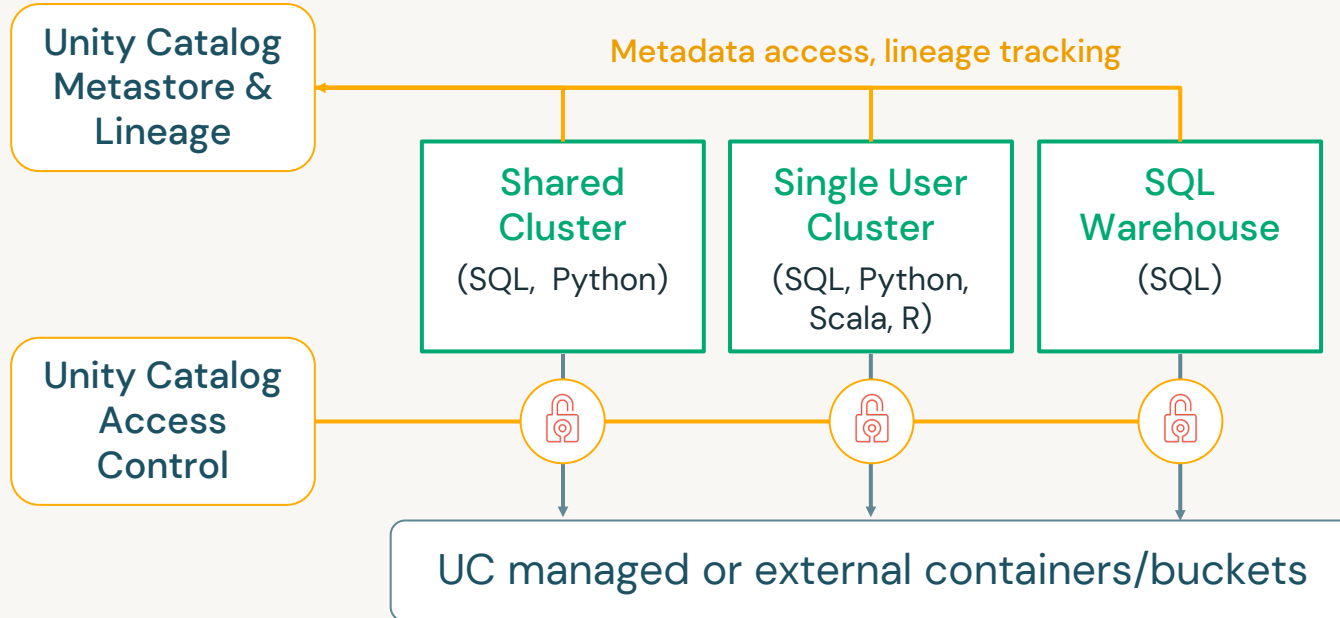
Production

Workspace





# Clusters/warehouses with Unity Catalog



## Shared clusters (Access mode: "Shared")

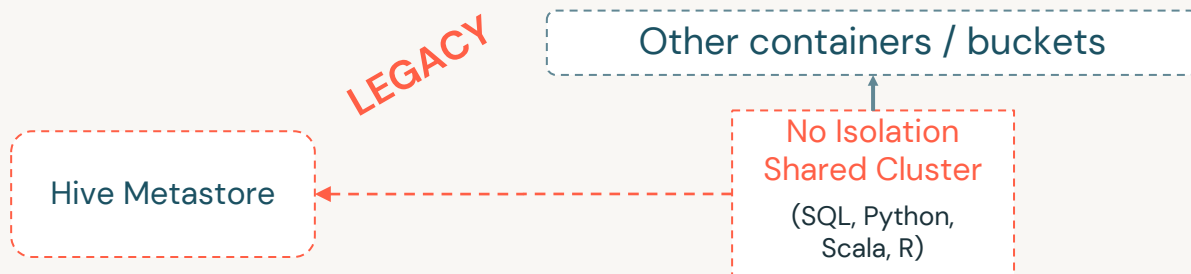
- Multiple users can work on a cluster in parallel
- Access to data is secured by Unity Catalog
- Python, SQL and Scala
- For workloads such as ETL or data exploration

## Single user clusters (Access mode: "Single User")

- A single user cluster is dedicated to a single user at creation time
- Access to data is secured by Unity Catalog
- Additionally supports R + ML Runtimes, MLflow, Spark submit jobs
- No dynamic views or row-level and column-level security

## SQL Warehouse (Classic, Pro and Serverless)

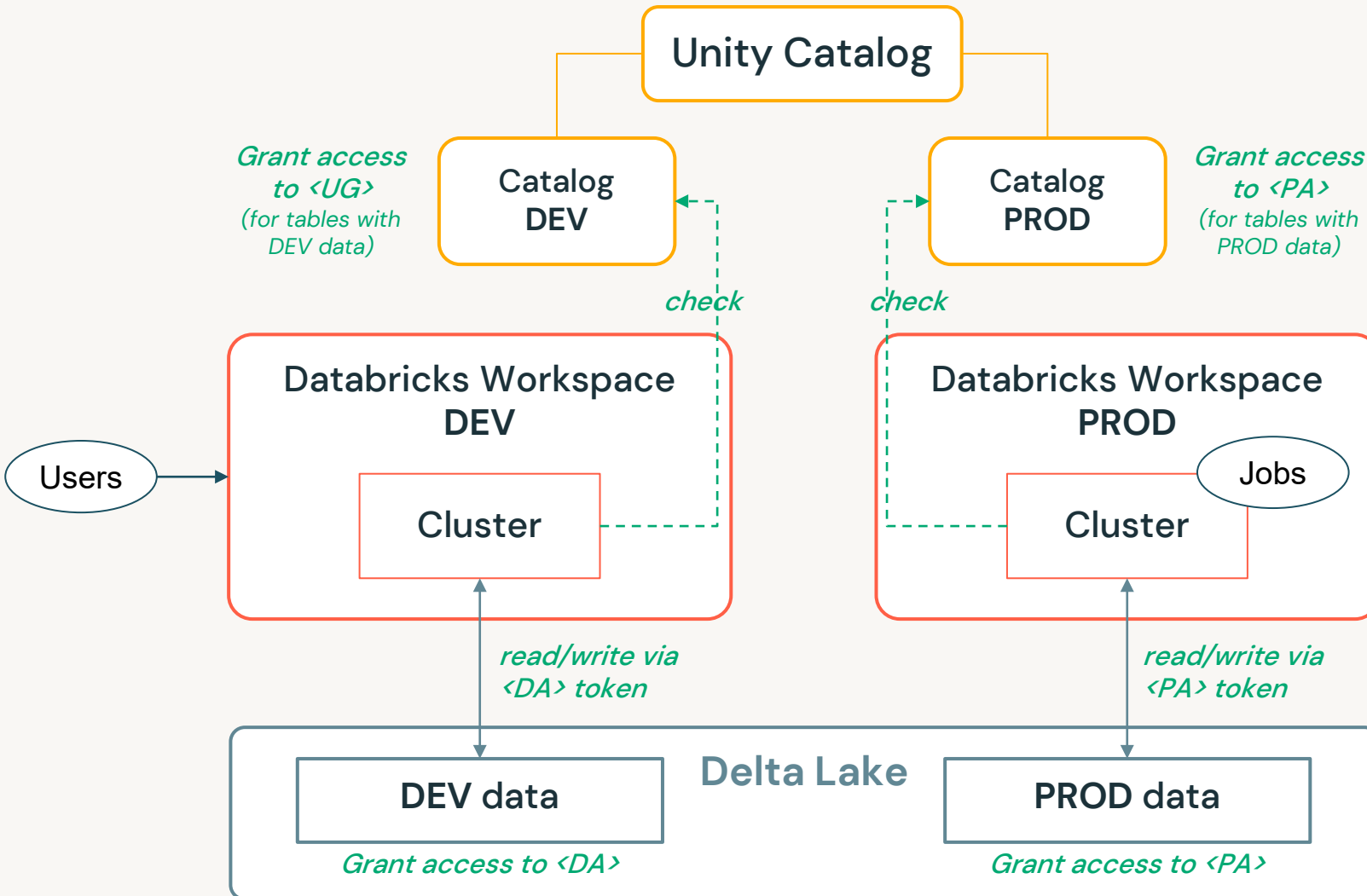
- Multiple users can work on a shared cluster in parallel
- Access to data is secured by Unity Catalog
- SQL only
- For Business Analysts using Databricks SQL Editor or external BI tools like Power BI, Tableau, ...



## No isolation shared clusters (legacy)

- Multiple users can work on a cluster in parallel
- Any language, any workload
- No governance by Unity Catalog (access control, lineage, ...) and no access to UC tables

# Software Development Lifecycle setup w/ UC



## Note:

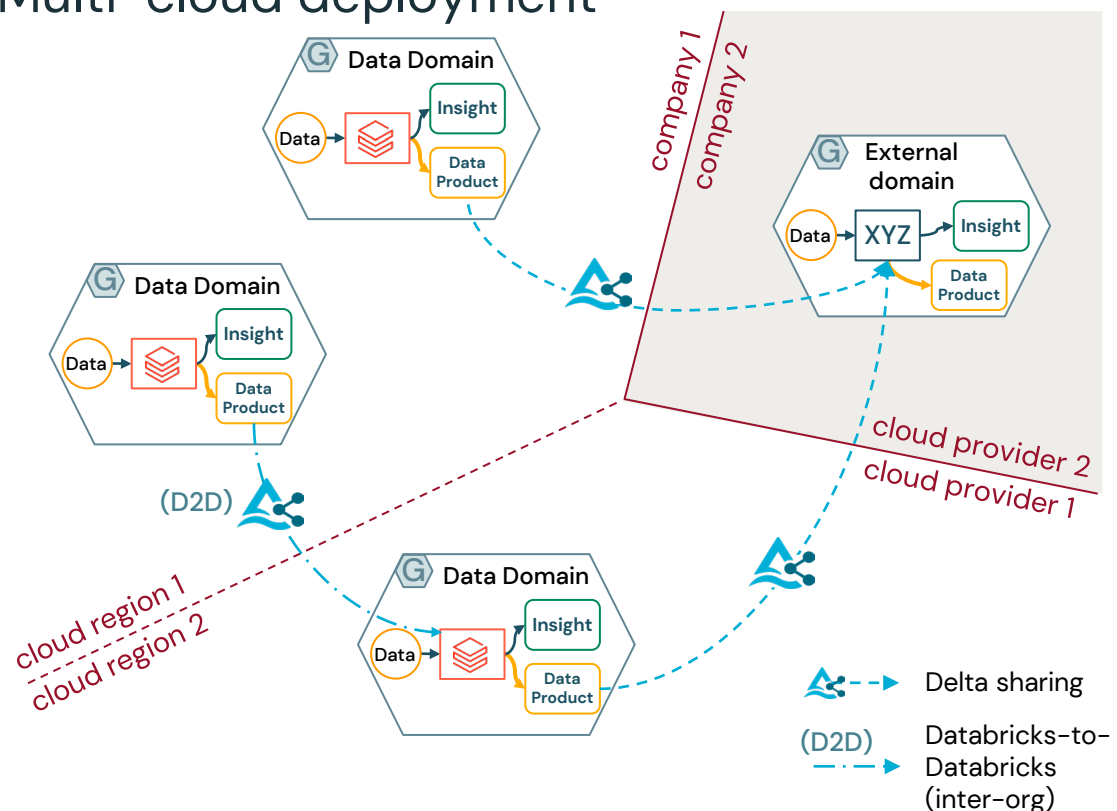
- One of the reasons to have different Workspaces for DEV and PROD is that they could reside in different VNets/VPCs. This is independent of UC, but leads to a setup as it is shown here.

- <DA>** DEV System Account (Service Principal, Instance Profile, Service Account)
- <PA>** PROD System Account (Service Principal, Instance Profile, Service Account)
- <UG>** User Group (Developers, Data Engineers, Data Scientists)

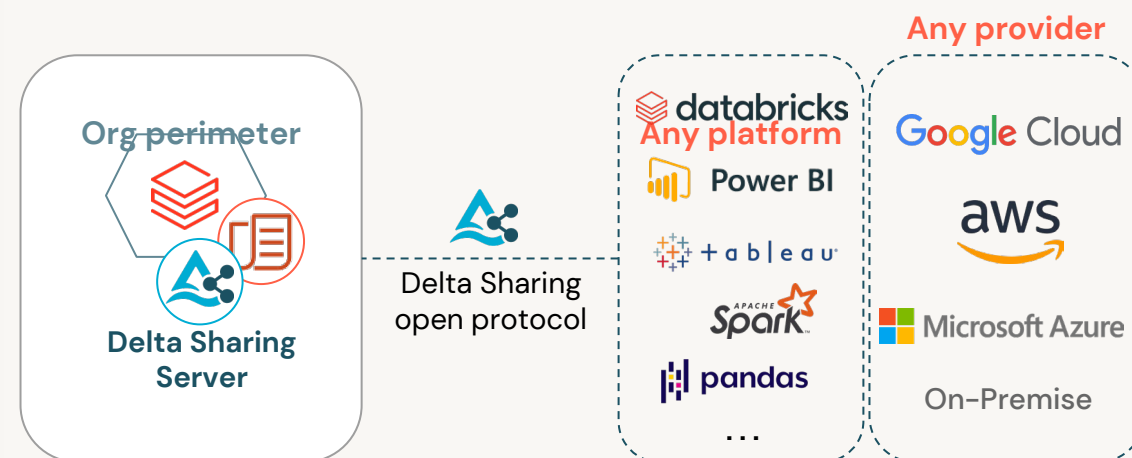
# Data Sharing

Secure access across organisational and regional boundaries

## Multi-cloud deployment



## Delta Sharing protocol



Unity Catalog manages all internal governance and access controls, and processes Delta Sharing requests

Delta Sharing Server receives requests via Delta Sharing protocol for secure access to data by external parties

Delta Sharing is an open and vendor-agnostic protocol

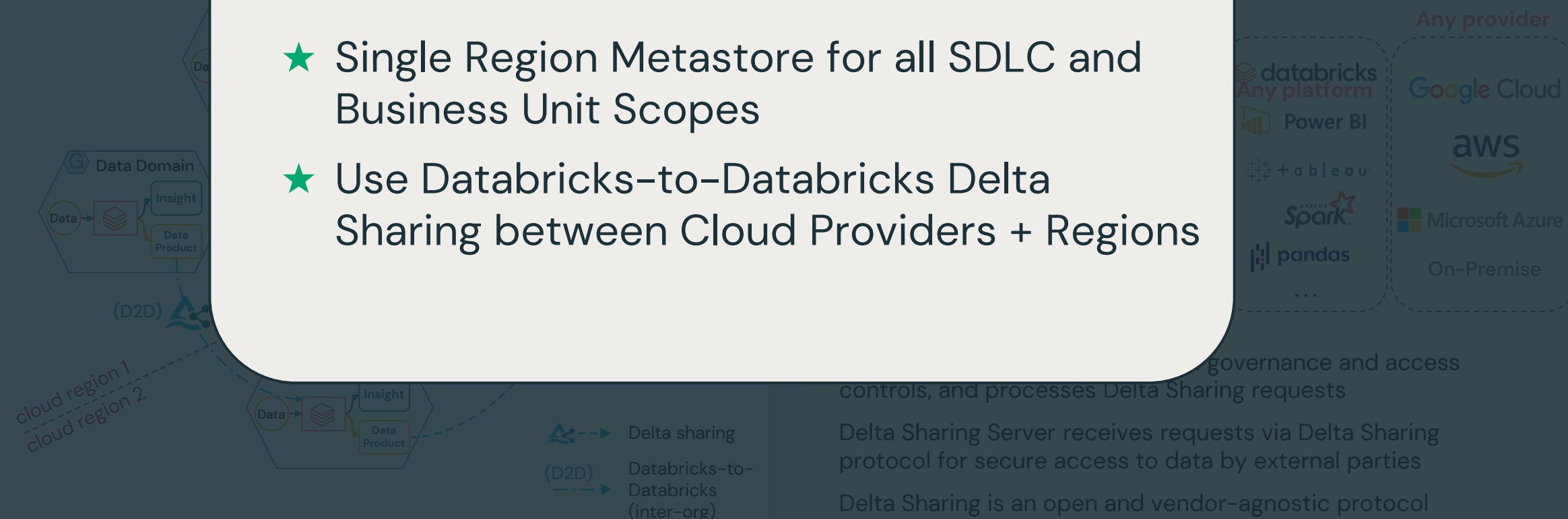
# Data Sharing

Secure access across organisational and regional boundaries

## Best Practice!

- ★ Single Region Metastore for all SDLC and Business Unit Scopes
- ★ Use Databricks-to-Databricks Delta Sharing between Cloud Providers + Regions

Multi-cloud



# Next Steps

# Learn more at the summit!



Databricks  
Events App



## Tells us what you think

- We kindly request your valuable **feedback**. Please take a moment to rate and share your thoughts about the session through the **Mobile App**.



## What to do next?

- Discover more related sessions in the mobile app!
- Visit the Demo Booth: Experience innovation firsthand!
- More Activities: Engage and connect further at the Databricks Zone!



## Get trained and certified

- Visit the Learning Hub Experience at **Moscone West, 2nd Floor!**
- Take complimentary certification at the event; come by the Certified Lounge
- Visit our Databricks Learning website for more training, courses and workshops! [databricks.com/learn](https://databricks.com/learn)



# Questions?



# Thank you!