eBook

Data Governance Architecture Patterns for Data Architects

Contents

Contents

Introduction

The dynamic interplay of data, analytics and AI is driving a wave of transformative innovations across diverse sectors, reshaping revenue streams and redefining corporate management paradigms.

Governance ensures data and AI products are consistently developed and maintained, adhering to precise guidelines and standards. It's the blueprint for organizations to bring solutions and data vision to life with consistency, guidelines and standards. It enables scale and speed for data engineers through repeatable workflow management. It allows you to collaboratively build and operationalize AI models for data scientists with transparency. It's security for data managers, ensuring data assets are shared far and wide to benefit all, yet are private when needed. It's trust for executives, with transparency of business insights based on their data and AI assets. It also drives operational efficiency for finance, particularly when powered by Databricks Unity Catalog.

This eBook provides practitioners with tools, tips, best practices and methodologies that drive successful data and AI governance implementation using Unity Catalog. Additionally, it uncovers the role of the Databricks Data Intelligence Platform as a catalyst for streamlining these transformative efforts. As organizations navigate an era characterized by AI-centric data strategies, the convergence of innovation and governance becomes the foundation for sustainable growth and responsible technological advancement.

Databricks Unity Catalog is the industry's only unified governance solution for all of a company's data and AI — across clouds and data platforms. Its foundation is the Databricks Data Intelligence Platform, which understands the uniqueness of your data — and drives the most comprehensive and unified governance solution for all of your company's data and AI. And it's built on a lakehouse to be open, scalable, low cost and high performance — the best of all worlds!

So, what are Unity Catalog's main value levers?

- Mitigating data and architectural risks
- Ensuring compliance
- Accelerating innovation
- Reducing platform complexity and cost while improving operational efficiencies
- Enabling collaboration and monetizing the value of data

How does Unity Catalog specifically provide for these positive outcomes?

- It provides a unified view and discovery of the entire data estate for accelerating innovation, which is quite helpful for data and solution architects. Having a unified solution for access management and auditing not only lowers license costs (often by 50% or even 90%), but it also enhances data and AI security
- By offering comprehensive data and AI monitoring and reporting, it improves trustworthiness for nontechnical users and experts alike
- By providing a collaborative environment — platform agnostic for data and model sharing — it democratizes every person within a business, unlocking new business values

Throughout, governance is simplified with intelligence. Data intelligence enables context-aware search using AI-powered knowledge engines. It automatically generates AI-enhanced descriptions, comments and documentation. It finds data using natural language — so that nontechnical people can ask questions themselves — without having to go through their IT staff to create SQL queries. Questions like, "Which marketing campaigns are most successful?" or "What vendors have been least productive across my supply chain?" This is finally the real-life democratization of data.

The democratization is broad — Unity Catalog unifies data and AI-enhanced governance across BI, data warehousing, data engineering, data streaming, data science and ML. It provides views and controls across all structured, semi-structured, unstructured and streaming data, as well as AI models, notebooks, workplaces, files, tables and dashboards. It provides more informative and actionable oversight through AI-enhanced holistic search, discovery and monitoring of usage trends, data lineage, discovery and model transparency. Whether with natural language or with SQL, organizations that harness this transformative AI-enhanced technology successfully unleash all of their data assets and become leaders for the future.

Unity Catalog Fits Across Cloud Provider Abstractions

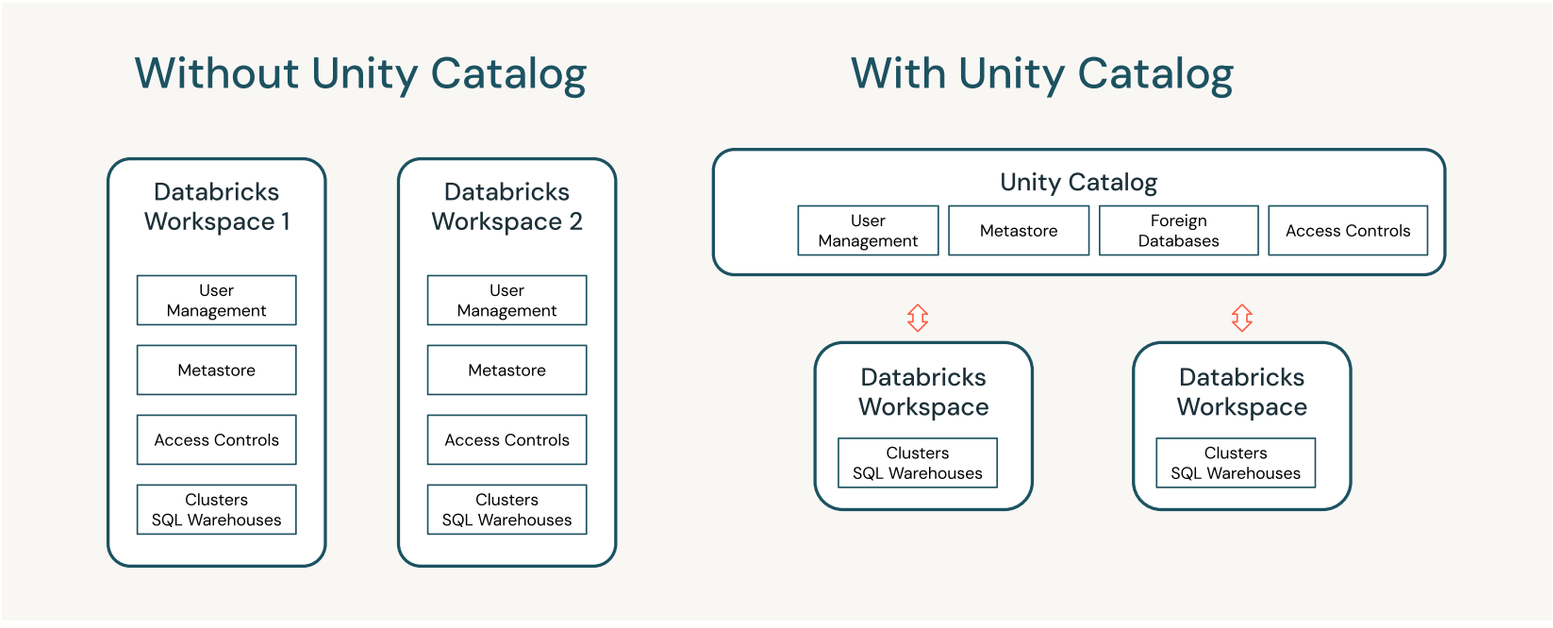
The Databricks Data Intelligence Platform is a cloud-native data platform available on all three major cloud providers: AWS, Azure and GCP. Customers have the flexibility to choose their preferred cloud provider when deploying Databricks, with a consistent experience across each of them. They can also opt for a multicloud strategy, seamlessly executing and integrating workloads across different providers.

In this context, effective data governance is essential for organizations relying on data, analytics and AI, yet challenges persist in implementation due to inadequate organizational processes and resources. These challenges are often exacerbated by the evolving role of chief data officers (CDOs). As a result, governance responsibilities often remain decentralized, leading to policy variations and multiple governing bodies across organizational divisions. This lack of centralized data governance function is termed *distributed governance*. Within the Data Intelligence Platform, we facilitate the implementation of a distributed governance model by leveraging Unity Catalog.

Unity Catalog enables the management of structured and unstructured data, as well as AI models within Databricks and external sources, under a single governance framework. Unity Catalog creates a consistent experience across cloud providers, abstracting many specifics across access, governance and identity. In this initial chapter, we delve into the specific aspects of each major cloud provider — AWS, Azure and GCP — and explore how Unity Catalog aligns with them.

Where Unity Catalog fits within each cloud provider

Before Unity Catalog, each Databricks workspace functioned monolithically, with support for dedicated users, pipelines, clusters, SQL warehouses and the concept of a local-to-workspace metastore. With Unity Catalog, the Databricks estate is unified for data and users, allowing these concepts to be shared across workspaces. Workspaces, in turn, continue to support the organization and isolation of nondata, workloads and compute.

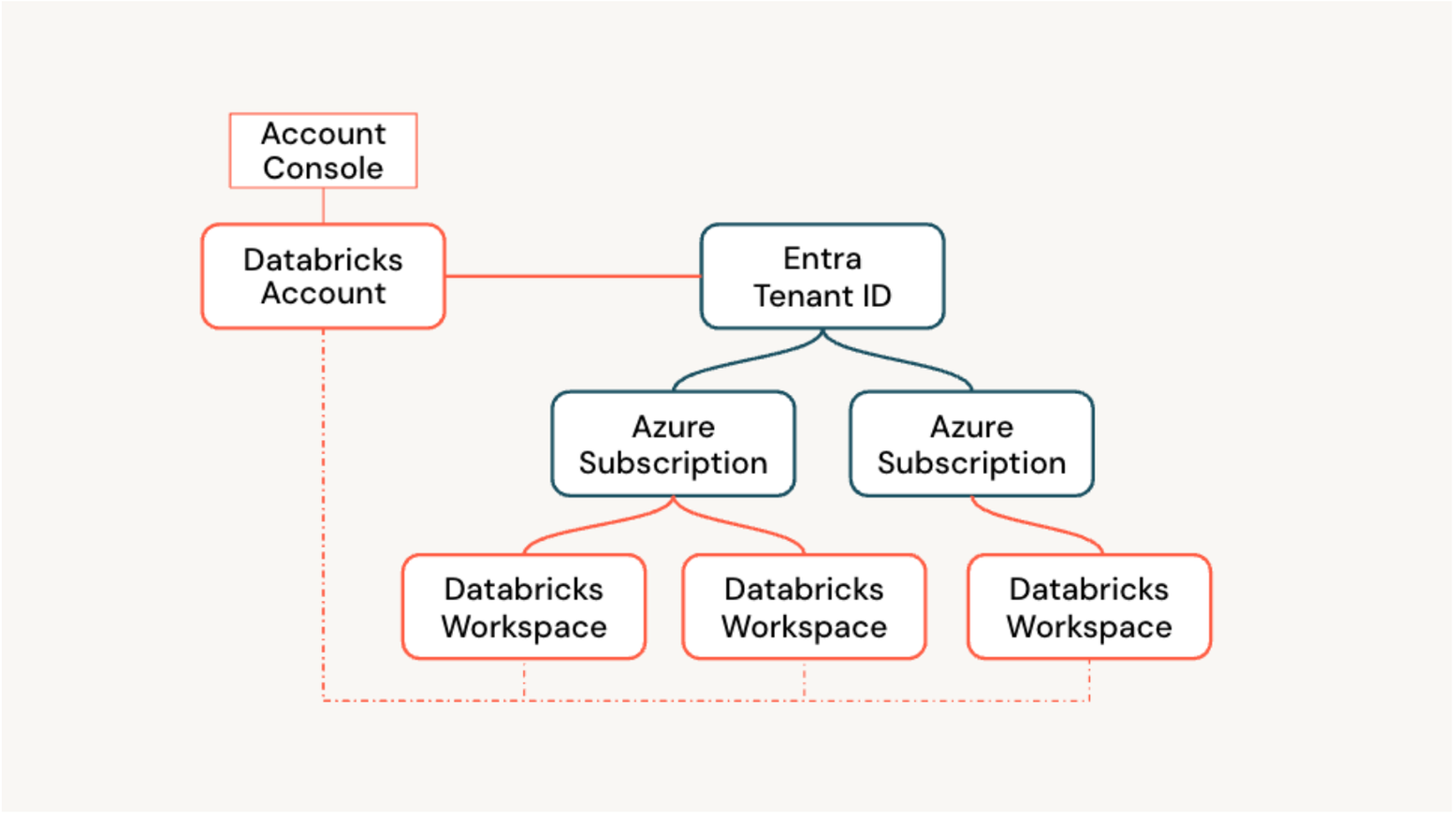


With Unity Catalog, the concept of the Databricks account gains increased significance compared to before. Now, users and metastore reside within the Databricks account, positioned one level above the workspaces.

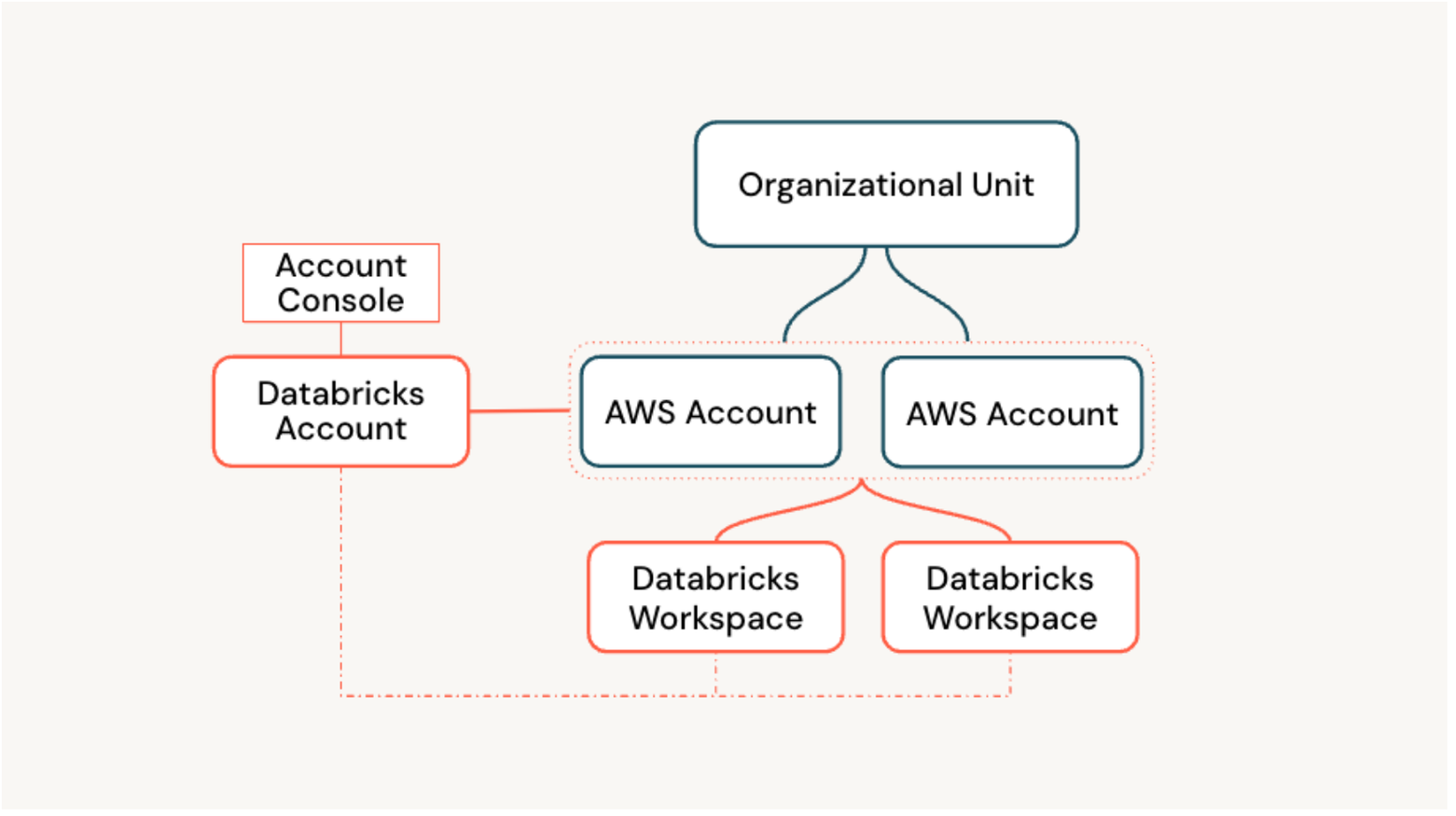
This new architecture also alters the distribution of responsibilities within Databricks. While the *workspace admin* previously held the sole administrative role, the introduction of the *account admin* role adds a new layer. The workspace admins retain control over individual workspaces, including user assignments and compute resource management. In contrast, the account admin oversees permissions related to data within Unity Catalog across all metastores, as well as other broader capabilities such as network connectivity for serverless compute that extend beyond a single workspace. Additionally, each metastore features a *metastore admin* role responsible for managing the data within it.

Depending on the cloud provider, the Databricks account maps differently, as shown in the following figures.

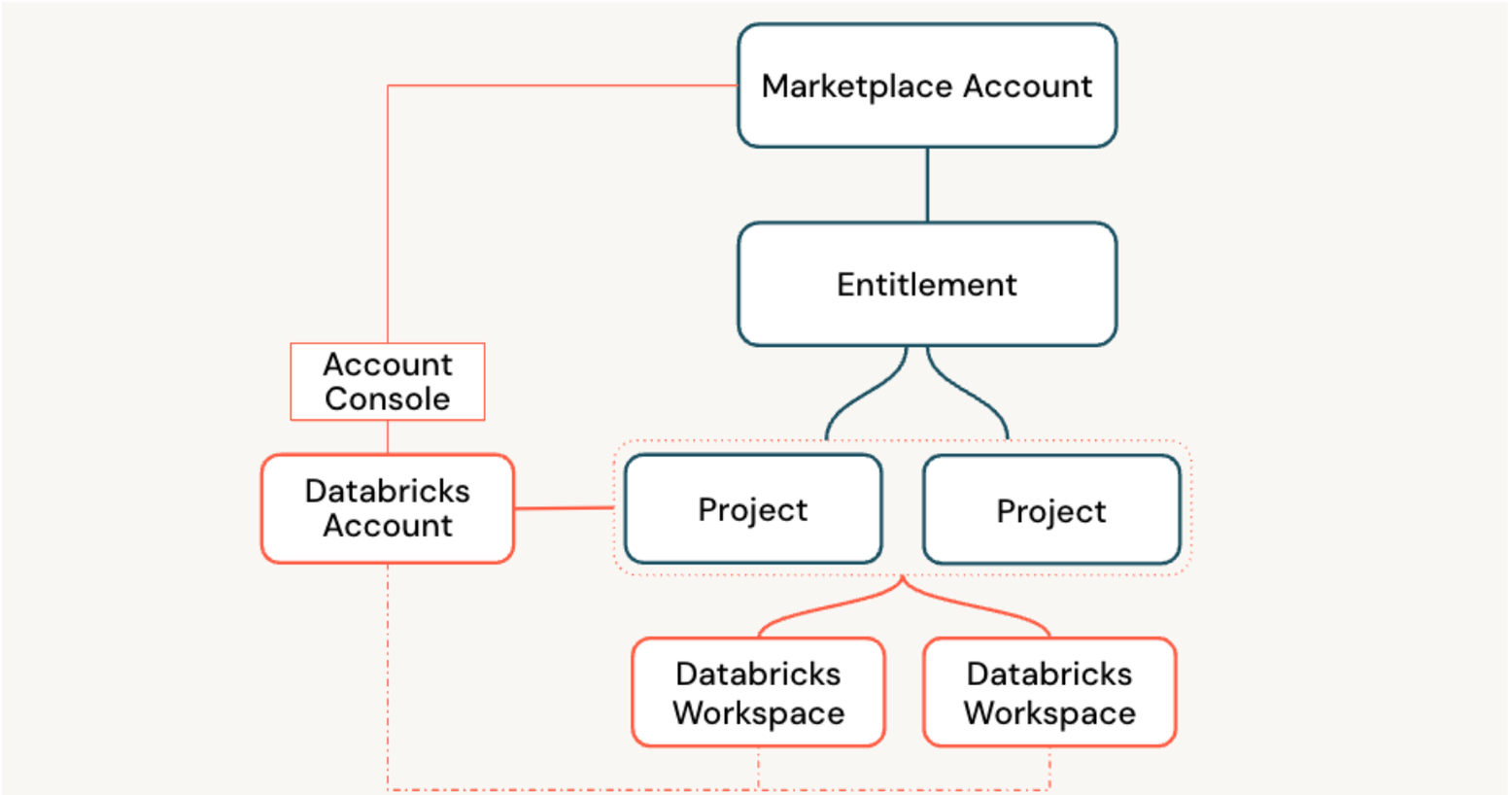
On Azure, the Databricks account is linked to the Microsoft Entra tenant ID. Any subscription inside the tenant is mapped to the same Databricks account. Consequently, any workspace in any subscription and resource group will be able to access the same metastore in the respective regions if admins allow this.



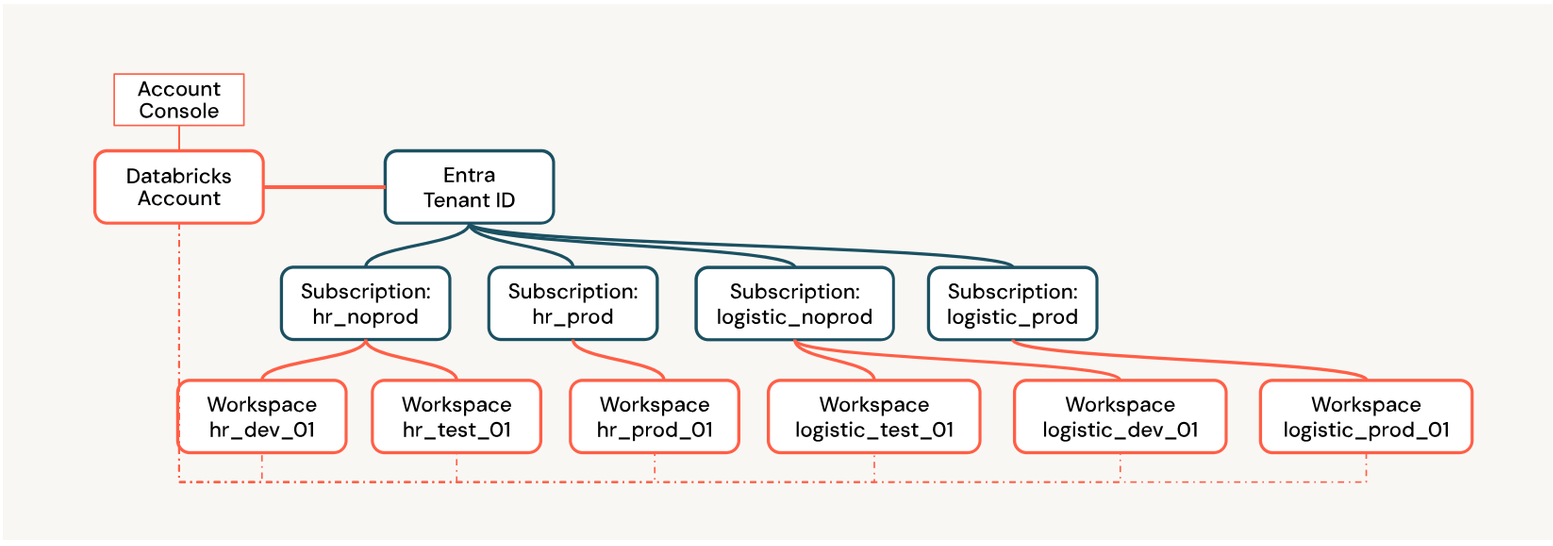
On AWS, the Databricks account maps to the organizational unit. It's common on AWS to organize the cloud environment in different accounts, and you can link one or more of them to your Databricks account so that all workspaces can access the same metastore.



On GCP, a Databricks account maps to a Marketplace account. Similarly to AWS, multiple projects on GCP can be linked to Databricks and all workspaces in them will be aggregated.



In the context provided, various strategies exist for organizing cloud environments within Azure, AWS or GCP, typically involving subscriptions, accounts and projects. Factors influencing these strategies often include business units, divisions and environments (such as dev/test/prod or production/non-production). Consider a scenario where a company organizes Azure subscriptions based on business units and environments. In this example, business units like HR and Logistics utilize Databricks, each mandated to maintain three environments categorized in two subscriptions as *prod* and *non-prod*: development, test and production. The Databricks account alignment for this is illustrated in the figure below.



How Unity Catalog abstracts the specifics of each cloud provider

Each cloud provider employs distinct underlying and proprietary concepts and features that Databricks utilizes, ranging from authentication methods like identity and access management (IAM) roles and service principals to storage accounts such as GCS, S3 and ADLS. While Databricks has always functioned as a multicloud solution, the specifics were previously exposed directly to end users rather than abstracted. For instance, in AWS, users had to utilize IAM roles configured within clusters to enable access to data in an S3 bucket at a specified path (s3://some-bucket). Similarly, on Azure, configuration of a service principal was necessary to read from an ADLS Gen2 storage, while on GCP, a service account was required to read from GCS. In the context of a multicloud data strategy, it becomes evident that such scenarios can quickly become complicated and confusing for end users. For example, if a user creates a notebook to read from S3, the same notebook would not function when attempting to access data from ADLS or GCS if moved to a workspace in another cloud.

With the unified governance facilitated by Unity Catalog, Databricks has also unified the approach to accessing data and services across different cloud providers, utilizing abstractions that are agnostic to the chosen provider. These abstractions include credentials for services and storage and external locations. Administrators only need to engage with the specifics of cloud providers when creating these objects; thereafter, they can operate within Databricks exclusively.

A service credential is used to access cloud-specific services other than storage such as SQS for AWS or Key Vault for Azure, via the respective SDKs. Service credentials give a uniformly managed solution to authenticate via Unity Catalog that is no longer cluster-bound via IAM roles, service principals or service accounts.

In a similar way, a storage credential functions like service credentials but is specific to storage, acting as a secure container that embeds the authentication mechanism for Unity Catalog when accessing it. This authentication is implemented as an IAM role on AWS, a managed identity on Azure and a service account on GCP.

Once authentication means are established, administrators require another secure element, the external location, to map the storage credential to a specific storage account. Administrators assign the storage credential to a particular S3 bucket, ADLS Gen2 or GCS when creating an external location.

With these two secure elements in place, end users can interact with their data independently from the cloud provider. Administrators grant end users permission to utilize external locations, enabling them to create tables and volumes for accessing structured and unstructured data, respectively.

In summary, Unity Catalog aligns with various cloud provider models and further provides mapping concepts from cloud provider storage and access abstractions into its hierarchy. This eases the burden of administration by allowing users to manage access and data in one place in a governed, secure and scalable way without requiring cloud-specific knowledge.

Designing Your Three-Level Namespace

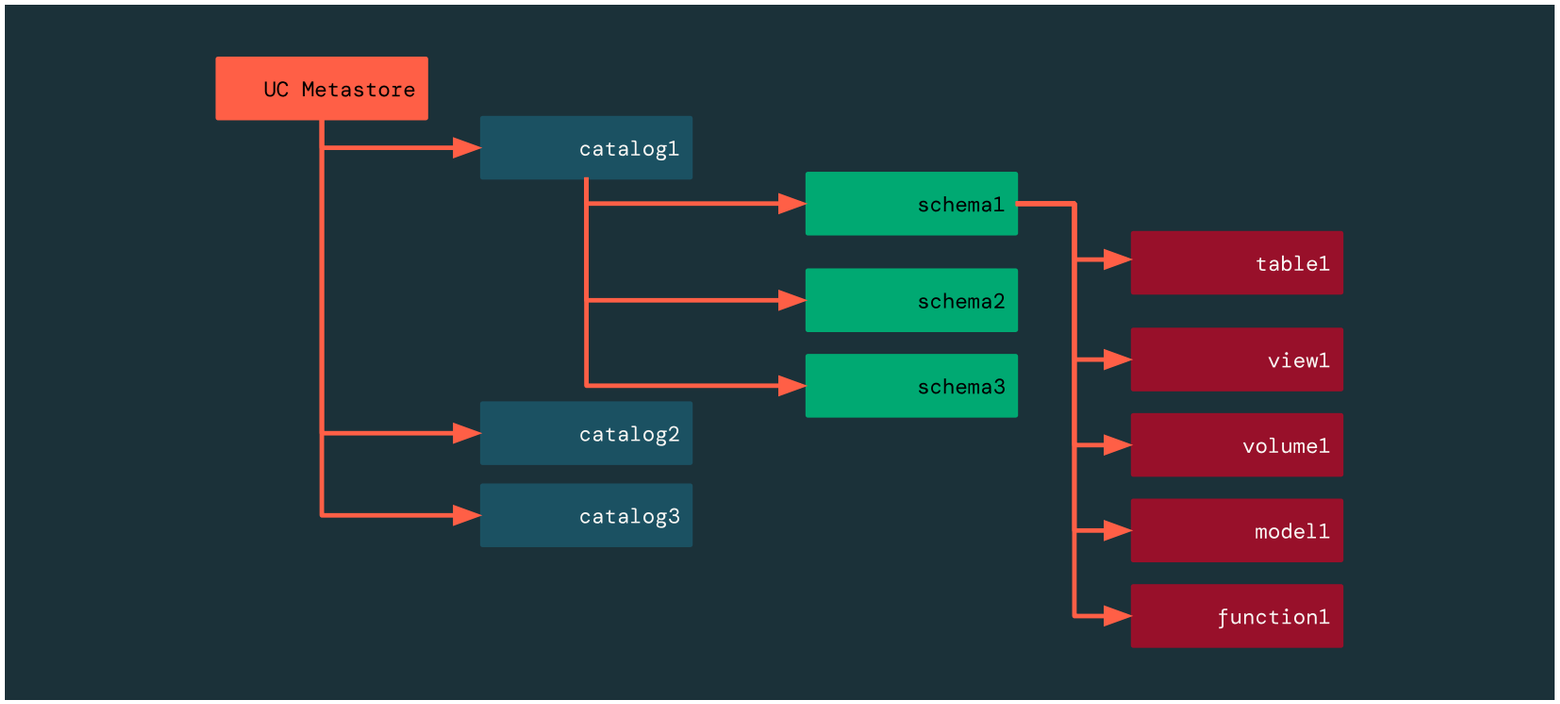
Unity Catalog provides a three-level namespace that can be used to represent a logical layout of how data is accessible and managed in your organization. The three levels are catalogs, schemas and assets.

Catalogs represent the top level of the structure. They can be made available for selective workspaces, which allows different operating units to enforce that their data is only available within their environments. This also allows teams to enforce the availability of data across software development lifecycle (SDLC) scopes (e.g., prod data in prod, dev data in dev).

Schemas are defined inside the catalogs. They can serve as a grouping for per-domain data assets.

Assets are inside schemas. They include the following objects:

- **Data assets:** Tables, views, materialized views, streaming tables
- **Machine learning models**
- **Unity Catalog Volumes:** These are logically assigned to a specific catalog and schema and represent folders with files (e.g., images or documents)
- **Functions:** Reusable procedures that can be written in SQL and Python



Catalogs can also be *internal* and *foreign*, depending on the data source.

Internal catalogs are created, owned and registered within the metastore based on data stored in relevant cloud storage. Foreign catalogs can originate from federated SQL databases or other metastores.

Unity Catalog supports two types of foreign catalogs: *Delta shares* registered as catalogs and foreign catalogs registered via Lakehouse Federation.

Naming conventions

When building the logical structure of the catalogs, consider the following axes of potential separation.

- **Region:** The region in which the data of a catalog is *physically* stored. This indication is quite useful when a multicloud (or multiregional) organization requires registering foreign catalogs from different regions.
- **Provisioner:** The division responsible for *providing* this asset (e.g., is it a *business unit* or a *central* data team of the platform).
- **Environment:** Indicates separate dev, staging and prod environments.
- **Logical data domain:** Identifies inside business units or cross-unit teams.

Using these rules, we can combine catalog names into the following naming convention:

{region}_
{provisioner}_
{environment}.{team_name or domain_name}.{asset_name}

Common scenarios

Following are some scenarios that allow us to conceptualize these naming conventions.

- **Naming by Business Unit:** An internal catalog with data provided from a business unit (BU) named BU1, with data related to marketing domain, inside the prod environment

  Since the table is in the internal catalog, we can omit the region name to avoid creating a long name. Applying our naming convention logic, the name for this catalog would look like this:

  Bu1_prod.marketing.some_table_name

- **Naming by Region:** A foreign catalog, sourced from region R1, distributed by the central team, with data related to the finance topic for dev environment

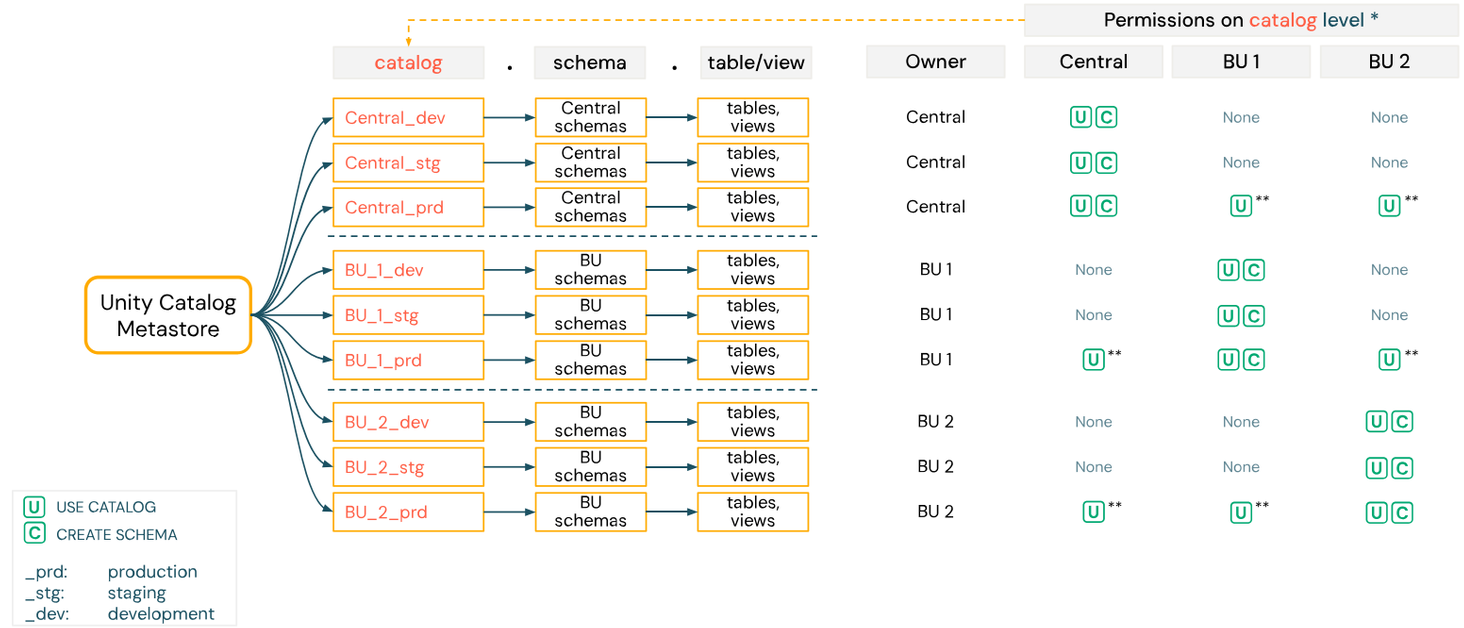
  r1_central_dev.finance.some_table_name

- **Naming by Multicloud or Multiregion:** A multicloud and multiregional organization, which has several separate data platforms, and each part of the platform is represented inside each region

The generic structure of such a scenario would look like this:

{cloud}_{region}_{team}_{env}.{domain}.{asset_name}

It's also important to understand how the logical blocks should be distributed in terms of ownership. This topic is extensively covered in the following chapters, but the overall layout is shown in the figure below.



General recommendations

When designing the catalog/schema structure, there are a few important aspects to keep in mind.

- **Strike a balance:** It's better to have a balance between the amount of catalogs and the number of schemas inside them. For example, choosing between 10 to 20 catalogs with thousands of schemas inside them or 200 to 300 catalogs with 100 to 200 schemas. This will increase the manageability of the setup.
- **Work with catalogs:** Catalogs serve as a very convenient boundary for teams using a Data Mesh architecture. Self-service teams can have their own catalogs and set up their own logical grouping inside these catalogs without having to reach out to account and metastore admins for changes.
- **Use automation tooling:** When it comes to managing the catalog structure at scale, we strongly recommend you use automation tooling. Databricks provides capabilities to create, control and manage catalogs and schemas via Databricks Terraform Provider. For advanced cases and integrations with third-party systems, you should use Databricks Rest API, and in particular, Databricks Python SDK.

Manage Your Data

In this chapter, we'll review various types of data and the relevant functionality provided by Unity Catalog to access and govern them in a secure fashion.

Supported storage types

For data sources that can be accessed and governed via Unity Catalog, we can classify the potential storage types into the following categories:

- Cloud storages with structured data
- Cloud storages with unstructured data
- Relational databases with structured data

Given these types, we can easily map to different technologies provided within Unity Catalog:

1. **Cloud storages with structured data:** Delta tables, as well as structured file formats like Parquet, are usually stored inside the cloud storage, in some kind of path-like layout, for instance:

   /org-root/teamA/events/ contains a Delta table with _delta_log folder and Parquet files

   To access and govern such data, use external locations in combination with storage credentials.
2. **Cloud storages with unstructured data:** Unstructured data (e.g., images or PDF files) is also stored in the cloud storage:

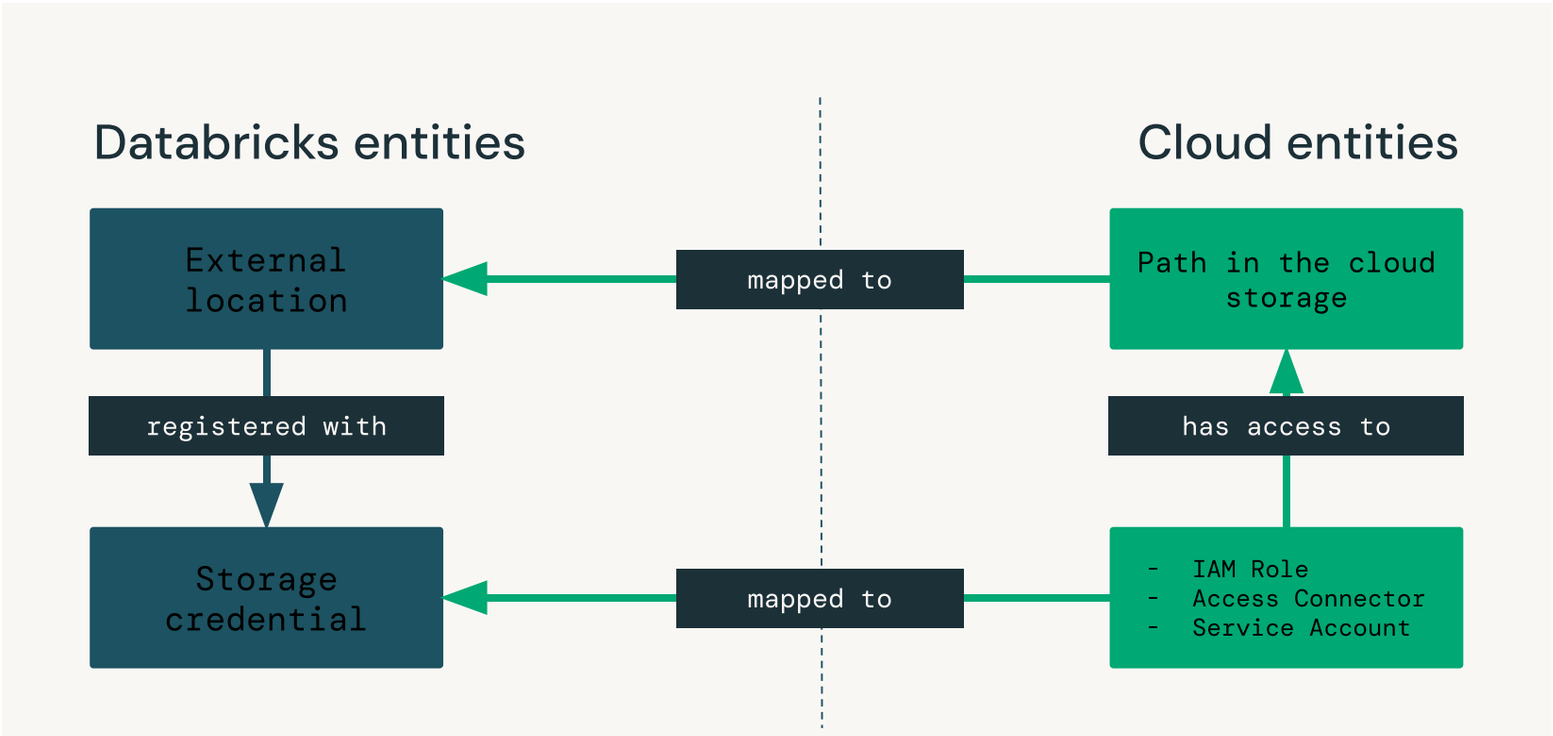
   /org-root/teamB/docs contains hundreds of PDF files

   Unstructured data is a core object in Unity Catalog. Access and governance for such folders (or subpaths) is governed via Unity Catalog Volumes.
3. **Relational databases with structured data:** Relational databases such as MySQL can store schemas with tables inside them. Usually such databases store structured information. To access, govern and control data from relational databases such as MySQL, Postgresql, SQL Server and more, Unity Catalog provides the Lakehouse Federation feature.

EXTERNAL LOCATIONS AND STORAGE CREDENTIALS

To access the data inside the same cloud region as the Unity Catalog metastore, the following is required:

- **Storage credentials:** These represent relevant cloud-level entities that are entitled to read the data on behalf of the users. For example, on AWS this role is dedicated to IAM roles, and on Azure these entities are managed identities (and service principals in rare cases).
- **External locations:** These are cloud storage objects (e.g., buckets, containers or subpaths inside these buckets or containers) that can be accessed by users through Unity Catalog. External locations can't be registered without a storage credential.



A combination of these entities is typically how teams access their existing data on the cloud storage in a secure way. For better manageability, multiple external locations can use the same storage credential to access the data as soon as the storage credential is entitled for the relevant operations on the underlying cloud storage objects.

To provide teams with self-service capabilities, they can be granted permissions to create storage credentials and external locations. Such a setup allows teams to easily plug in their existing cloud storages without being dependent on the central platform team.

MANAGED AND EXTERNAL TABLES

Unity Catalog metastore also comes with the capability to register managed storage. Managed storage is still storage located on the data plane (in the customer object storage), but it doesn't require users to create folder-level structures to separately store the data.

Unity Catalog supports setting managed storage at several levels:

- **Metastore:** Can have a root managed storage, which needs to be registered during the metastore creation process. Though this is a simpler approach, enterprise-grade organizations usually require the data to be stored in logically, per-environment separated cloud object storages.
- **Catalog-level storage:** Can be set when the catalog is created by using the LOCATION clause. This is the most common way to logically separate the managed storage across the environments and teams, which also provides a lot of flexibility for their operations.
- **Schema-level storage:** Is used in cases when there's a strict organizational requirement to store data even at a schema level inside separated cloud containers or buckets. This may be useful for cases when teams perform so many IO operations that they may hit the underlying "per-container/per-bucket" limits of the cloud object storage. To add a managed location to a schema-level storage, the LOCATION clause should be used when creating a schema.

In practice, **catalog-level storage by location** is the most commonly used setting for enforcing enough logical control without introducing too much management overhead.

Since tables can be both managed and external, some logical questions arise: Which tables shall be used in which cases, and what are the benefits of using managed tables?

Managed tables shall be used in most cases, specifically when:

- Performance of accessing data in these tables is critical for business continuity (e.g., BI and analytics). In such case, a managed table with powerful features like Predictive Optimization can deliver lightning-fast analytics at scale.
- Overall management of the cloud storage is already (or is becoming) a burden for the platform team. In this case it would be a question of improving the efficiency of internal processes and decreasing the dependency on the underlying storage layout.

One use case for an external table is when the table is accessed from a system that's outside of the Unity Catalog control and users can't read data via standard open protocols such as Delta Sharing and ODBC or JDBC.

GOVERNANCE FOR UNSTRUCTURED DATA USING UNITY CATALOG VOLUMES

Given that a lot of modern valuable data can't easily fit into a table-like format, Unity Catalog comes with a tool for governing access to unstructured data called a *volume*.

In short, volumes come on top of the external locations, adding a capability to govern access to the unstructured data. They also provide FUSE-like access to the data stored in them when they're accessed via Databricks clusters.

Volumes are perfect for storing unstructured data and data-related assets such as:

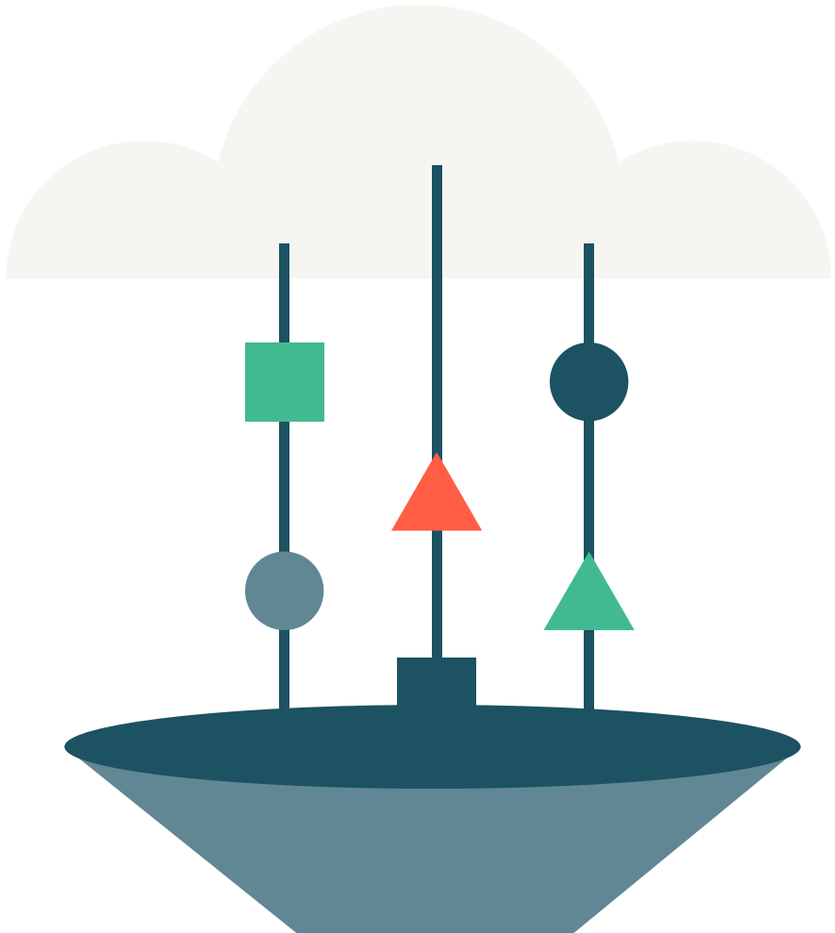
- Spark Streaming Checkpoints
- Dependent libraries (e.g., jars)
- A distribution location for Python packages that are used in the workflows and on the all-purpose clusters

Lakehouse Federation

Lakehouse Federation provides an interface to easily access data inside relational databases. Databricks offers federation into external databases like MySQL, PostgreSQL, SQL Server, Snowflake, BigQuery and more.

Foreign catalogs registered inside the Unity Catalog metastore can be further governed on the Unity Catalog side, which makes access control seamless and integrated with other Unity Catalog tooling.

Lakehouse Federation reduces data fragmentation and access management overhead. Therefore it's crucial to incorporate existing sources that can't be replicated to the data lakehouse into the overall data access pattern enforced by Unity Catalog.



Securing Your Data

In this chapter, we'll look into how Unity Catalog secures and governs data and AI assets, the various access patterns provided by Unity Catalog and some of the key best practices for securing and governing your data with Unity Catalog.

As we've seen in the previous chapters, Unity Catalog provides:

- Fine-grained access control to data and AI assets cataloged under the standard catalog
  - **Tables:** Structured data in various different formats such as Delta and Parquet
  - **Volumes:** Unstructured data formats such as CSV, XLSX, MP3, JSON, etc.
  - **Models:** Machine learning models
- Read-only access to popular database solutions cataloged under the foreign catalog

The data and AI assets under the standard catalog actually live in the cloud provider's object store. Unity Catalog manages access to these assets using the cloud provider–specific mechanisms registered as storage credentials within the Unity Catalog metastore:

- Service principal or managed identity in Azure
- IAM role in AWS
- Service account in GCP

Storage credentials are configured at the Unity Catalog metastore level, usually by the platform team or the workspace admins. The most important thing to note here is that the Unity Catalog service must be the only way to access the object store. If there's a way tvo bypass Unity Catalog and access the object store by any other means, then this would also bypass the entire governance structure of the data platform.

Permissions model

Unity Catalog provides a logical governance layer that abstracts out the file-level access controls. In other words, regardless of the nature of the datasets, Databricks users no longer need to deal with the data in terms of files and folders but only as Unity Catalog objects. Moreover, for all the assets cataloged under Unity Catalog, a single standard governance model applies.

Elaborating on the Unity Catalog governance model, let's look at the important aspects of the Unity Catalog permissions model:

- **Unity Catalog permissions are at the metastore level:** Given that Unity Catalog objects are defined at the metastore level, it's obvious that the permissions associated with these objects are also defined at the metastore level. This means that a user will be able to access the objects from any workspace (associated with that particular metastore) as long as they have permission.
- **Unity Catalog permissions apply to account-level identities:** Permissions on the Unity Catalog objects can be granted only to the account-level identities. So the first step for granting permission is to define or provision the identities at the Databricks account level and assign them workspaces through which users can access the objects.
- **Privilege inheritance:** Privileges are inherited down the Unity Catalog object hierarchy. For instance, if you provide ALL PRIVILEGES for a user at the catalog level, then the user will have ALL PRIVILEGES for all the objects within that catalog.
- **Permission assignment:** Granting permissions on an object can only be granted by the identities with the following permissions.
  - Owner of the object
  - Owner of the catalog/schema containing the object
  - Metastore admin
- **Dropping an object:** Only the object's owner or the owner of the catalog/schema that contains it can drop an object.

Advanced access controls

In addition to granting permissions on objects to individuals and groups, Unity Catalog provides some advanced controls for managing access.

- **Row filtering and column masking:** Unity Catalog allows for fine-grained access controls in the form of dynamic row-level security (RLS) and column-level masking (CLM). These can be applied directly to the tables, preventing the need for data duplication and creating and maintaining views.
- **Attribute-based access control (ABAC):** ABAC allows access control to be conditional based on broader properties of the user, resource and request. It builds on top of and coexists with the Unity Catalog core security model, including all privileges. The core concepts of ABAC are *attributes*, *rules* and *inheritance*. Attributes are derived from the context of the object such as identity, tag and time. Rules use these attributes to decide access. Rules are inherited down the object hierarchy and coexist with existing object grants. ABAC provides enhanced capabilities that enable organizations to fulfill complex governance requirements at scale.
- **Workspace-catalog binding:** A catalog object can be bound to a subset of workspaces in order to restrict access (read-write/read-only) to that catalog from only those workspaces. This provides an additional layer of access control. For example, this feature can help prevent users from accidentally accessing production data from the development workspace — even if they have permission to do so — by binding production catalogs to production workspaces. Unity Catalog now allows binding storage credentials and external locations to specific workspaces for enhanced security.

Access patterns

Unity Catalog object hierarchy, permissions model and advanced access control mechanisms pave the way to various access patterns that simplify the overall governance of data and AI assets. Let's look at some of the most common access patterns.

1. **Isolating assets at the business unit (BU) level:** In this pattern, each BU is assigned a set of catalogs that are independently owned and operated by the business unit. This means the BUs are allowed to create schemas, tables and other Unity Catalog objects within the assigned catalogs and manage access to everything under these catalogs. This pattern implements a federated governance approach, keeping certain aspects of it still centralized. The catalogs can be associated with the object store that belongs to the BUs by setting the managed locations. Furthermore, the catalogs can be bound to the corresponding workspaces of the BUs. These measures facilitate strict isolation of assets at the storage level.

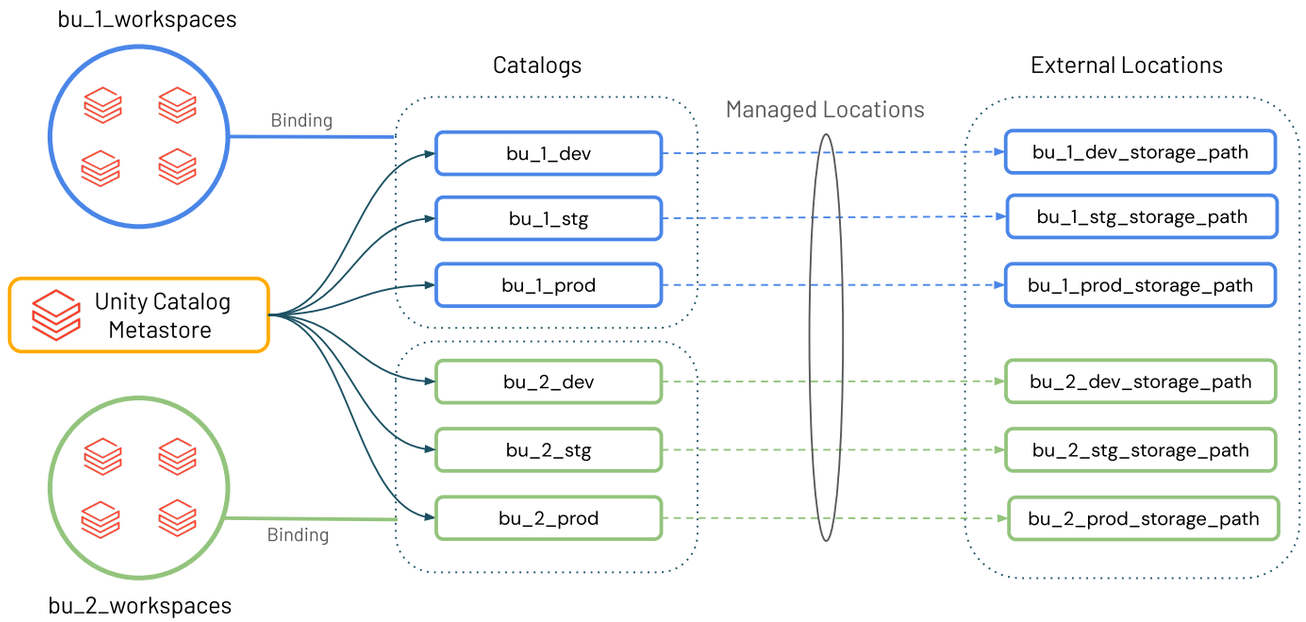


Figure: Business unit–specific catalogs with managed locations and workspace-catalog binding

2. **Cross-functional access:** The power of data lies in how best it can be leveraged. The value of data increases as more and more use cases are powered by it. This means it's crucial to provide data to the relevant teams that can make use of the data, even if they're not a part of the project or the business unit that owns the data. Unity Catalog makes it much easier to share data across teams and business units. For instance, consider a situation where data science or machine learning teams might need access to the production datasets from another business unit. With Unity Catalog, access to the datasets can be granted with simple GRANT statements, and the production catalogs can be bound to the recipient workspaces as read-only catalogs. This creates two layers of protection against accidental writes while providing secure and easy access to the datasets.

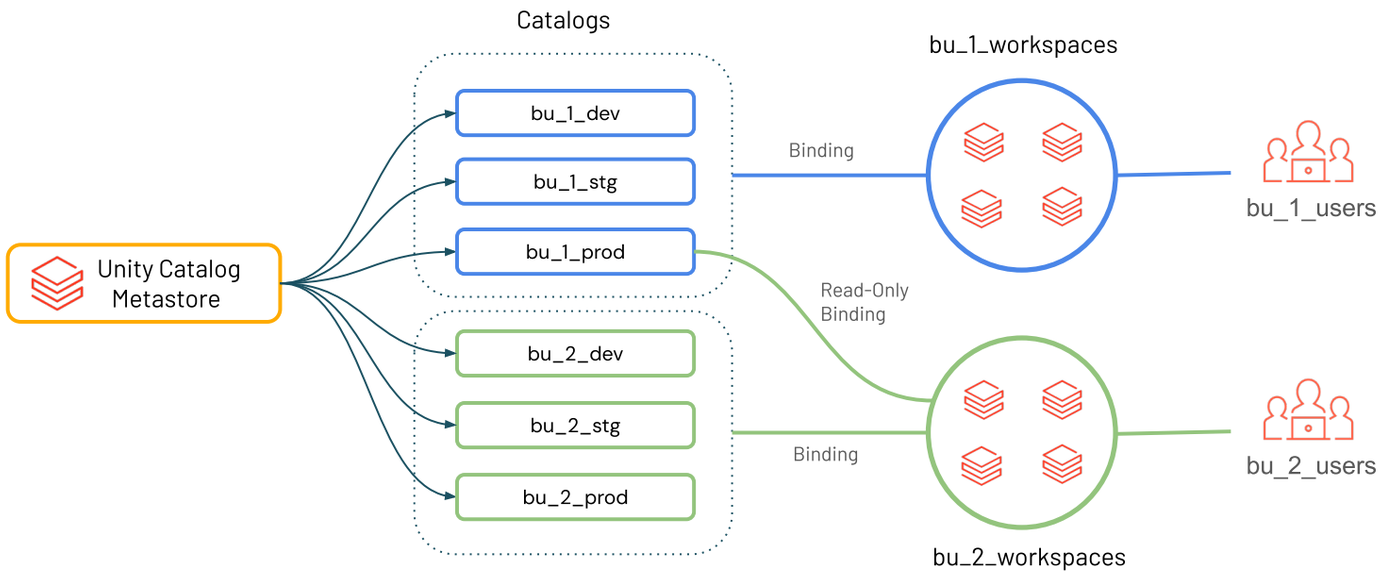


Figure: BU 2 users with access to BU 1 prod catalog with read-only workspace-catalog binding

3. **Data as products:** Data and AI artifacts have emerged as valuable assets that have value beyond departmental or even organizational boundaries. As a result, the concept of data as a product (DaaP) is becoming more and more popular, and many organizations are trying to develop data products and make them available to others. This poses several challenges, including data product cataloging, discovery, access mechanisms, governance and observability. Unity Catalog can help address some of these challenges. For details about implementing data products with Databricks, please refer to this blog post.

These are just a few examples of how the Unity Catalog features and permissions model can be leveraged to set up different access patterns for easy and secure access to all your data and AI assets. With Unity Catalog, it's easy to develop other patterns to address the unique requirements of a given organization.

Best practices

Following are some of the best practices for keeping your data and AI assets secure with Unity Catalog.

- Lock down the object store and prevent external data access. The data in the object store associated with Databricks must be accessible only through Unity Catalog.
- Grant access to groups instead of individuals. This practice simplifies permissions management, especially when onboarding and offboarding users.
- Powerful roles such as *metastore admin* should be granted to an empty group. As needed, a person or service principal can temporarily be elevated to a metastore admin role. Since metastore admin is a role with very high privileges, this practice will avoid any accidental incidents and enforce the least privilege principle. Privileges in Unity Catalog are inherited, so it's better to provide specific access based on necessity rather than providing additional privileges to those who don't need it. This is known as *the least privilege principle*. When granting access to Unity Catalog objects, follow the least privilege principle.

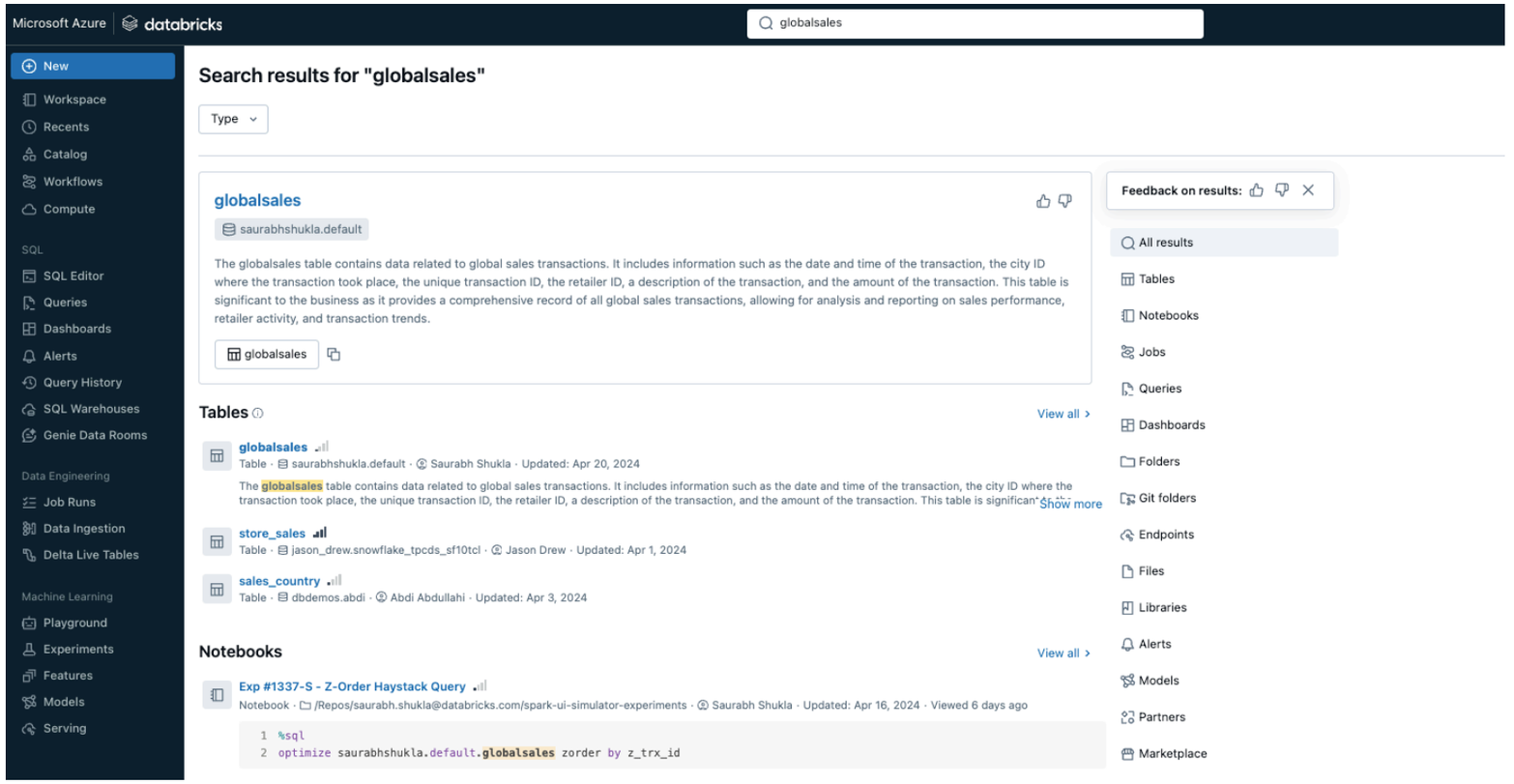
Searching for Data

Intelligent search is critical for working effectively with the data and AI assets in the Databricks Data Intelligence Platform. When considering search capabilities, it's important to address two critical use cases:

1. **Navigation:** Helping users find what they're looking for quickly and efficiently. This usually implies the user already knows precisely what they want to find.
2. **Discovery:** Helping users who might have a general notion of what they want but don't know what specific things to search for

Databricks search

Databricks search allows users to search for tables registered in Unity Catalog, notebooks, queries, dashboards, alerts, files, folders, libraries, jobs, repositories, endpoints, models, partners and Databricks Marketplace listings in your Databricks workspace.



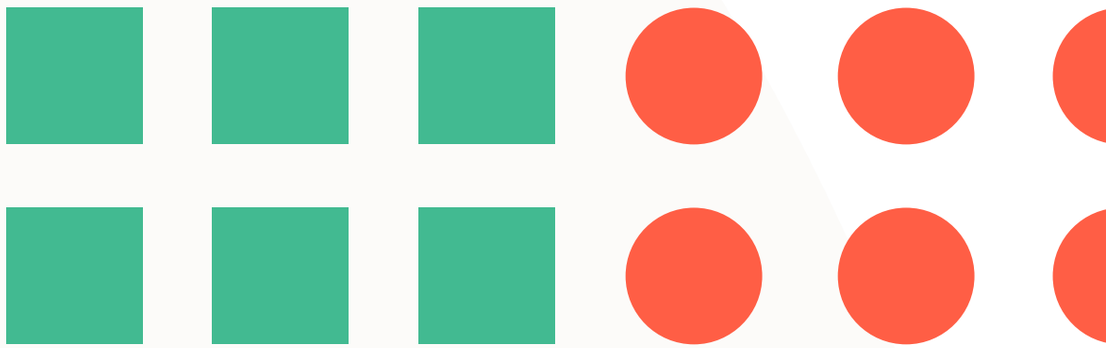
Databricks search leverages DatabricksIQ — the Data Intelligence Engine for Databricks — to provide a more intelligent AI-powered search experience. AI-generated comments use large language models (LLMs) learned from multiple sources to automatically add descriptions to tables and columns managed by Unity Catalog. These comments equip the search engine with an understanding of unique company jargon, metrics and semantics, providing the context required to make search results more relevant, accurate and actionable.

Databricks search offers *semantic search*, which interprets the meaning of words and allows natural language queries. It retrieves semantically similar assets and combines them with keyword searches. For instance, if a user searched for "What should I use for geographies," the search engine would focus on "geographies" and find related terms containing geographic attributes such as cities, countries, territories, geolocations, etc.

Databricks search separates search terms from filters, enhancing natural language queries and *search query understanding*. For example, the query "Show me tables about inspections" would yield "inspections" (key term) and "table" (object type).

Using popularity signals based on user interactions to rank objects, Databricks search delivers improved *relevance with popularity*. This method ranks the most popular assets higher, reducing trial and error.

Databricks search also incorporates *knowledge cards*. When search identifies what you're looking for with high confidence, the top search result turns into a knowledge card. These cards provide additional asset metadata for easier identification of relevant resources.

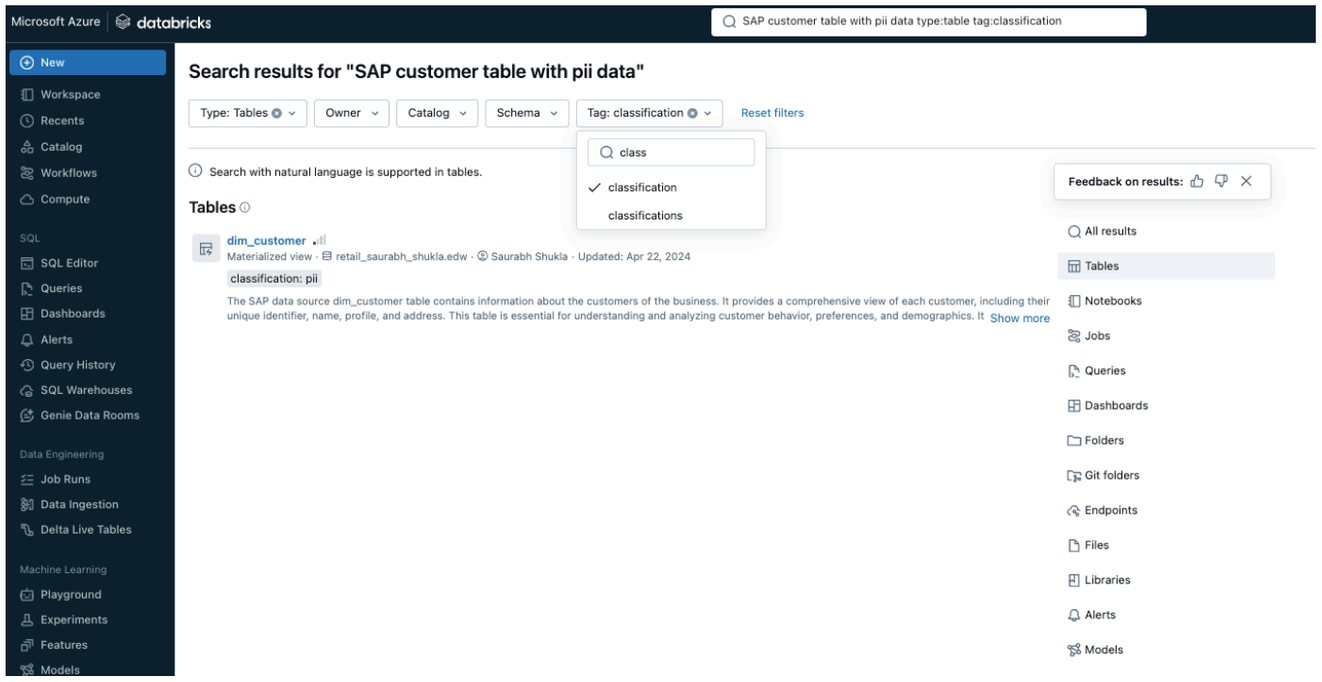


Tags as business taxonomy

Datasets can be challenging to discover and categorize. Tags are additional metadata that can be attached to securable Unity Catalog entities to improve various use cases, including governance, search and discovery of datasets.

To establish a business taxonomy using discovery tags in Databricks, follow these guidelines:

- **Define tagging standards:** Collaborate with stakeholders to define a standardized set of tags that align with business objectives and regulatory requirements
- **Apply tags:** Use the Databricks tagging feature to label data assets manually or automatically based on data ingestion workflows
- **Train users:** Educate data users about how to use tags for searching and managing data, ensuring that everyone understands the tagging system and its benefits
- **Monitor and refine:** Regularly review the tagging system to ensure it meets the evolving needs of the business. Adjust and add new tags to keep the taxonomy relevant and valuable.



Data lineage as a search enabler

Data lineage is essential to a pragmatic data management and governance strategy. Following are important key points for leveraging lineage as a data search enabler.

- **Tracking data evolution** — Lineage allows users to visualize the journey of data from its origin through various transformations to its final form. This visibility into data evolution helps users search and identify the datasets relevant to their analysis or business needs.
- **Supporting data compliance and audit** — By providing a clear map of where data comes from and how it's processed, lineage helps organizations comply with regulatory requirements. Users can search for specific datasets and understand their provenance, which is crucial for audit trails and compliance reporting.
- **Enabling observability for the data stack** — Lineage contributes to the observability of the data stack by showing how data is used and where it flows. This makes it easier for users to search for data assets and assess their impact on different parts of the system, such as ML models, dashboards and reports.
- **Facilitating impact analysis** — Before making changes to data assets, users can search and review the lineage to understand the downstream effects of those changes. This helps manage risks associated with data modifications and ensures that dependent processes aren't adversely affected.
- **Improving data quality assurance** — Understanding the lineage of data helps in identifying the sources of data quality issues. Users can search for and trace errors back to their origins, which is essential for maintaining the integrity of data pipelines and ensuring the accuracy of analytics.
- **Enhancing collaboration and knowledge sharing** — Lineage provides a shared understanding of data flows, which is valuable for collaboration across data teams. Users can search for data assets and share insights about their lineage, promoting knowledge transfer and alignment on data practices.
- **Integration with other data governance tools** — Lineage information can be exported via REST API and integrated with other data catalogs and governance tools. This allows for a unified view of data lineage across the entire data ecosystem, enhancing searchability and governance across multiple platforms.

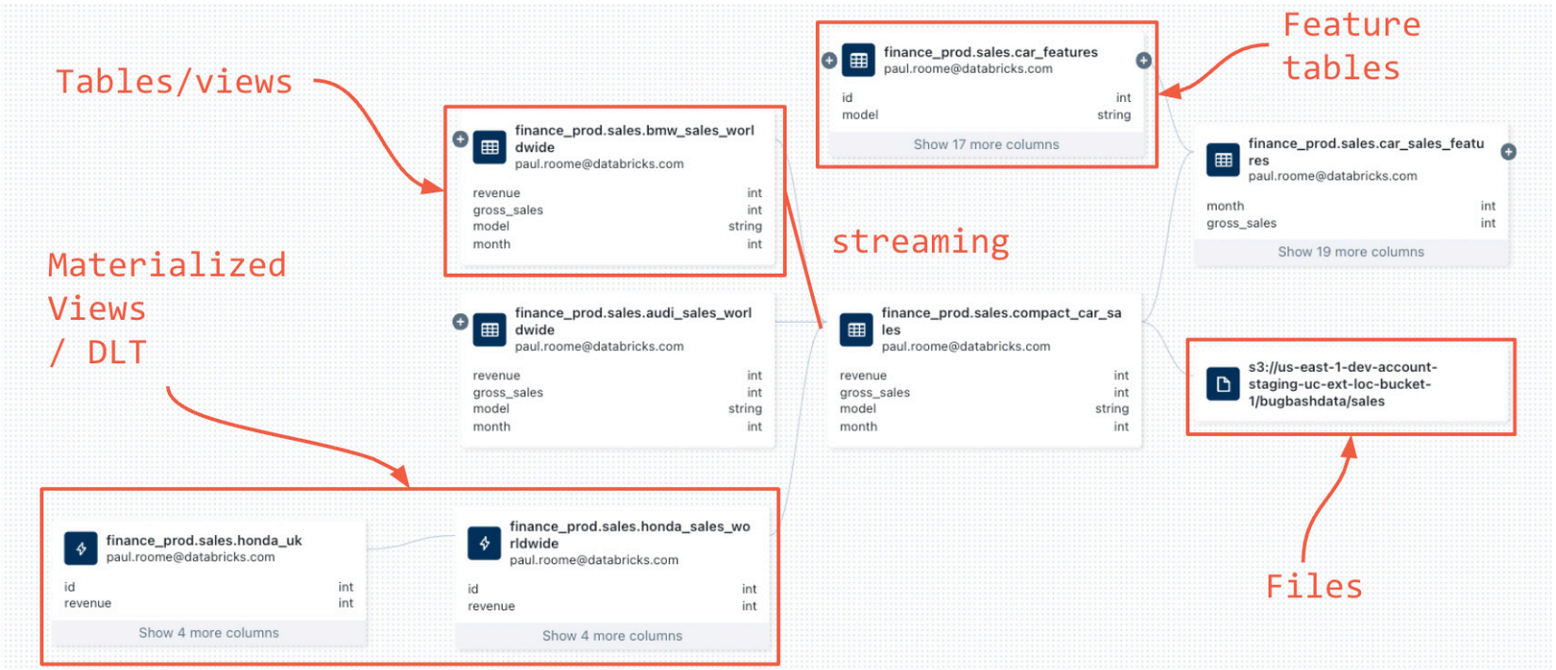


Figure: Data lineage for all data and AI assets

GRANTING BROWSE PERMISSION ACROSS THE DATA ESTATE

With browse permission, users can discover data without having read access to the data. A user can view an object's metadata using Catalog Explorer, the schema browser, search results, the lineage graph, information_schema and the REST API.

Tracking and Promoting Data Quality

Tracking and monitoring data quality is essential for all kinds of data pipelines. It's an integral part of a modern and mature data and AI architecture.

Data quality as a practice isn't a one-time check but a continuum across your data and AI pipelines. It's also crucial for machine learning and generative AI in addition to more traditional ETL motions. There's a known maxim in the AI world that goes "Garbage in, garbage out," which refers to the idea that the quality of the data used to develop a model has a direct impact on the quality of the model output. The same principle applies to reporting and other downstream decision support systems.

Databricks Unity Catalog provides a host of features dedicated to ensuring the quality of your pipelines can be accurately measured, monitored, tracked and maintained for any use, from data to features to models.

Databricks Lakehouse Monitoring

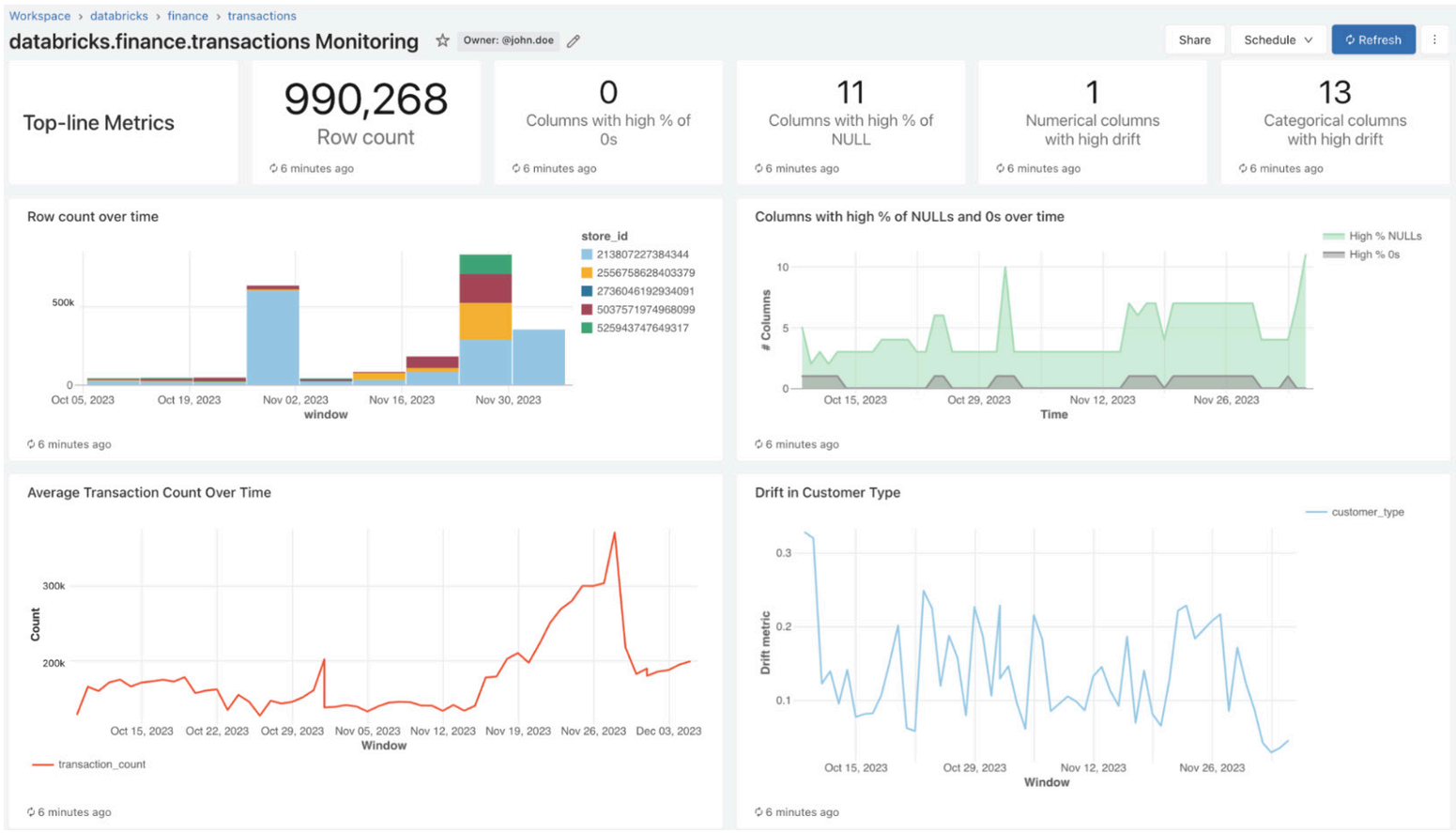
Unity Catalog facilitates data quality tracking and monitoring with Databricks Lakehouse Monitoring. This feature allows you to specify a monitoring endpoint on any given Unity Catalog table or Model Serving endpoint in your environment.

For example, once a monitoring endpoint is configured on a table, Lakehouse Monitoring calculates a set of out-of-the box metrics such as min, max and other distributional metrics, including statistical distribution tests like the Kolmogorov-Smirnov (K-S) test. For a complete list of metrics, you can refer to our Databricks product documentation. Custom metrics of your choice can also be added. All the information is aggregated in a dashboard that's automatically created. This dashboard can be used not only to visualize and track the evolution of these metrics but also to set alerts via the use of Databricks SQL Alerts. For instance, you could set an alert, coupled with a variety of webhook options from email, for Slack notifications to let you know when a certain column in your data is out of range.

Lakehouse Monitoring is a unified data monitoring solution that spans data engineering and ML/AI use cases. This can be particularly useful for tracking the performance of machine learning models and Model Serving endpoints by monitoring inference tables that contain model inputs and predictions.

When changes in your table's data distribution or the performance of the corresponding model are detected, the tables created by Databricks Lakehouse Monitoring can alert you to the change and help you identify the cause. This can be crucial in identifying and addressing *concept drift*, which arises when the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways.

Lakehouse Monitoring also allows you to control the time granularity of observations and set up custom metrics, providing flexibility in how you track and analyze your data and models. For instance, you can use it for tables that contain the request log for a model, with each row being a request, and columns for the timestamp, the model inputs, the corresponding prediction and an optional ground-truth label. Lakehouse Monitoring then compares model performance and data quality metrics across time-based windows of the request log.



Lakehouse Monitoring provides the following benefits for organizations looking to better track, enforce and monitor data quality best practices:

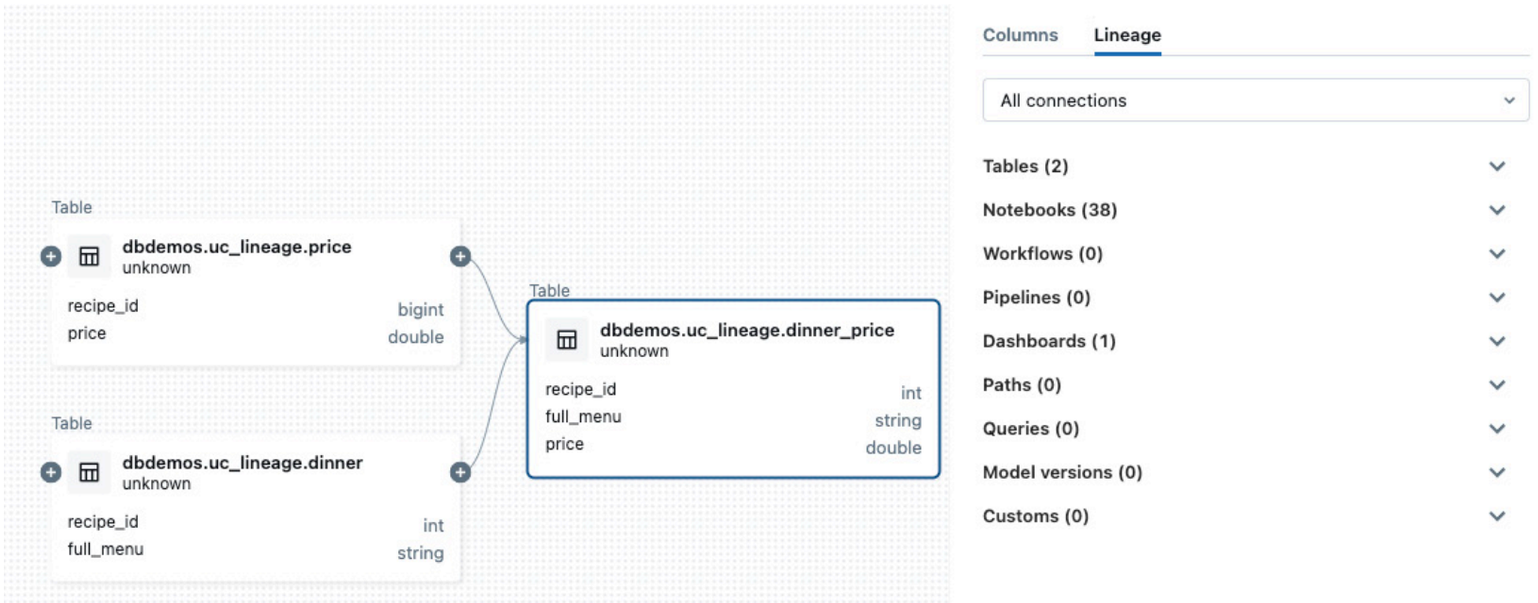
- **Comprehensive monitoring** — Allows you to monitor the statistical properties and quality of all the tables in your account. This includes tracking the performance of machine learning models and Model Serving endpoints by monitoring inference tables that contain model inputs and predictions.
- **Data confidence** — To draw useful insights from your data, you must have confidence in the quality of your data. Lakehouse Monitoring provides quantitative measures that help you track and confirm the quality and consistency of your data over time.
- **Alerts on data changes** — When changes in your table's data distribution or the performance of the corresponding models are detected, the tables created by Databricks Lakehouse Monitoring can alert you to the change and help you identify the cause.
- **Customization** — Databricks Lakehouse Monitoring lets you control the time granularity of observations and set up custom metrics. You can also set slicing expressions to monitor feature subsets of the table in addition to the table as a whole.
- **Quality visualization** — An automatically generated dashboard visualizes data quality for any Delta table in Unity Catalog. All metrics are stored in Delta tables to enable ad hoc analyses, custom visualizations and alerts.
- **Proactive issue detection** — You can proactively discover quality issues before downstream processes are impacted, ensuring that pipelines run smoothly and machine learning models remain effective over time.
- **Serverless** — Lakehouse Monitoring is fully serverless, so you never have to worry about infrastructure or tuning compute configuration.

Lineage

Another crucial feature that Unity Catalog provides to track data quality and facilitate its promotion is lineage. Any operation in Databricks that's performed on a Unity Catalog registered table (managed or external) using a Unity Catalog–enabled cluster is logged. This includes the lineage information.

Lineage within Unity Catalog includes information about upstream and downstream tables. For example, if a table is the product of a merge between two other tables, this information is captured and queryable, as is the information about additional artifacts where tables may be used like notebooks, workflows and dashboards. Lineage extends beyond the table level. Unity Catalog is actually able to provide lineage information up to the column level, which makes it extremely powerful in terms of identifying data quality issues with high granularity.

The lineage information captured by Unity Catalog is available via the Databricks UI, but it's also directly available through Databricks system tables, namely through the table and column lineage tables. For more information, you can refer to the system tables documentation. The information is also programmatically available via the API.



Lineage plays a crucial role in AI development, providing a comprehensive view of the data journey from its original source through all its transformations to the point it's used in a machine learning model. This visibility is essential for understanding the impact of data quality on model performance, as well as for troubleshooting issues that may arise during the model development process.

Lineage information within Databricks can help track different versions of a model, allowing for comparisons and evaluations of their performance over time. This can be particularly useful in identifying when and why a model's performance may have changed, which can inform future model development and refinement.

A powerful tool for auditing purposes, lineage provides a transparent record of the data's journey, which can be crucial for regulatory compliance, especially in industries where explainability and accountability are paramount.

Maintaining data quality is a continuous process, and lineage can support this by highlighting areas where data quality may be compromised, such as in the data ingestion or transformation stages. By identifying these areas, teams can take proactive steps to improve data quality, thereby enhancing the overall performance and reliability of their AI systems.

Lineage captured through Unity Catalog affords organizations the following benefits with regards to tracking and promoting data quality:

- **End-to-end lineage tracking** — Unity Catalog captures lineage data across all lakehouse entities, including notebooks, workflows, dashboards, queries, tables and machine learning models. This end-to-end visibility helps you understand how data assets are created and used across all languages.
- **Data governance and discovery** — Lineage data is crucial for managing change, ensuring data quality and implementing data governance in an organization. It aids in data discovery by providing a clear picture of the data's origin and its transformation journey.
- **Problem identification and change management** — By tracking the lineage of data, you can identify issues at their source and manage changes effectively. This allows for impact analysis, helping you understand the potential consequences of changes in your data.
- **Integration with other features** — Lineage data can be used to support intelligent actions in other product features, such as data insights or data quality alerting. It can also be integrated with partner catalog vendors for seamless lineage tracking.
- **Access and permissions** — Lineage data is aggregated across all workspaces attached to a Unity Catalog metastore, meaning that lineage captured in one workspace is visible in any other workspace sharing that metastore. However, users must still have the correct permissions to view the lineage data.
- **Data quality platform integration** — Catalog Explorer can display health/quality status and violations along the lineage chain to assist in root cause analysis. This requires all tables in the lineage chain to be monitored through Lakehouse Monitoring.

Auditing Your Data and Its Usage

By now it should be evident how Unity Catalog can assist organizations in securing data and AI in a unified manner at scale. However, implementing security measures does not guarantee immunity from unexpected events or security incidents. From a security standpoint, many organizations have embraced the "Never trust, always verify" principle by adopting a zero trust architecture (ZTA) model. The core tenet of the ZTA model emphasizes that even with robust identity verification, compliance validation before granting access and least privilege access controls, default trust should be avoided and continuous monitoring of system activities — including the Databricks Data Intelligence Platform — is essential. In this chapter, we'll explain how this can be achieved using Unity Catalog system tables.

System tables

System tables function as a centralized operational data repository, supported by Delta Lake and supervised by Unity Catalog. They facilitate querying in multiple languages, enabling diverse applications across BI, AI and even generative AI. Increasingly, customers are utilizing system tables for various purposes, including usage analytics, consumption/cost forecasting, efficiency analysis, security and compliance audits, service level objective analytics and reporting, actionable DataOps, and data quality monitoring and reporting. This consolidation of information for auditing purposes significantly streamlines the tasks of security and compliance teams. Prior to the introduction of system tables in Unity Catalog, this data was fragmented across different systems, requiring an ETL process to prepare and render the data usable.

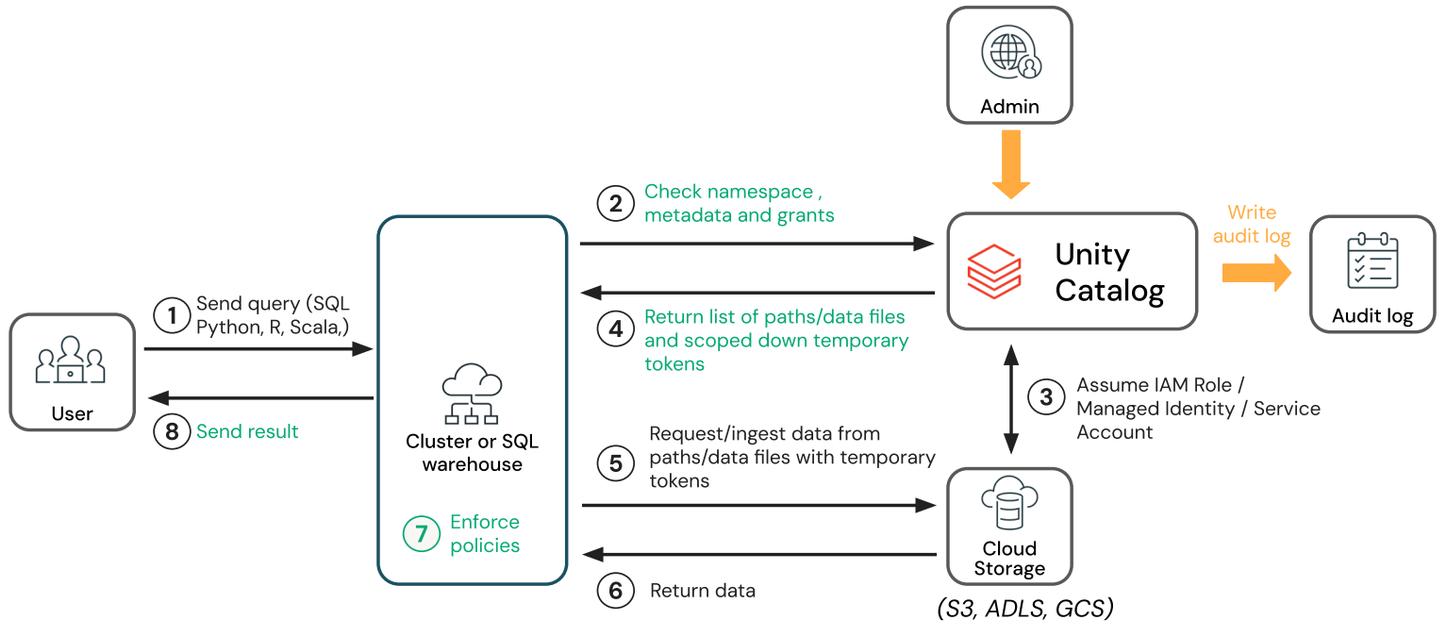
While multiple schemas are available, this chapter predominantly concentrates on the *audit* and *information schema* tables and their role in monitoring and auditing usage of operations conducted on data stored in Unity Catalog.

The audit and information schema system tables

When auditing data and its usage, the two most critical system tables to consider are *system.access.audit, system.information_schema* and all tables under *system.information_schema*.

The audit system table encompasses more than just data access–related information. It aggregates all audit logs at both the Databricks workspaces and account levels. Within the account-level logs, records of all significant data accesses facilitated by Unity Catalog are included. For further details, see the official documentation.

The following figure illustrates the lifecycle of a query in the Databricks Data Intelligence Platform. It showcases how Unity Catalog mediates access to underlying data in storage accounts from any compute resource in Databricks, ensuring permissions are validated against each query. This process enables data availability to the Unity Catalog audit system table.



Examining the Unity Catalog–related information within the audit log reveals over 100 events, detailing access to various asset types, including structured (tables/views) and unstructured (volumes) data, as well as AI models.

On the other hand, the information schema table furnishes details about securables in Unity Catalog across all catalogs within the metastore. It offers insights into data at rest, with a particular emphasis on schemas, grants and lineage of various securables within this context.

Questions for audit logs

System administrators, data governance teams or data owners seek to comprehend how the data they oversee is accessed. They want to monitor access patterns and receive alerts if any unusual activity occurs, aligning with the ZTA model introduced earlier in the chapter. This approach enhances security by providing an additional layer of protection, thereby mitigating the risk of security incidents. Proactive monitoring can take various forms, such as real-time monitoring to respond promptly to potentially harmful events or periodic assessments to detect any anomalies.

The Databricks Data Intelligence Platform offers comprehensive tools for implementing an alerting system using data from system tables. Users can employ ETL processes to aggregate data, query it in multiple languages and configure alerts using internal alerting capabilities. The system is also equipped to take action, such as revoking permissions, if instructed, when a user accesses a resource they shouldn't have access to.

The questions that can be addressed using audit logs found in system tables are diverse and contingent upon the nature of the data within the data platform. The following list categorizes such questions into a nonexhaustive list of categories:

- Proactively identifying flaws in the permissions model
- Monitoring access and operations
- Tracking alterations to tables and schemas
- Detecting new and unexpected access patterns
- Monitoring data downloads and uploads

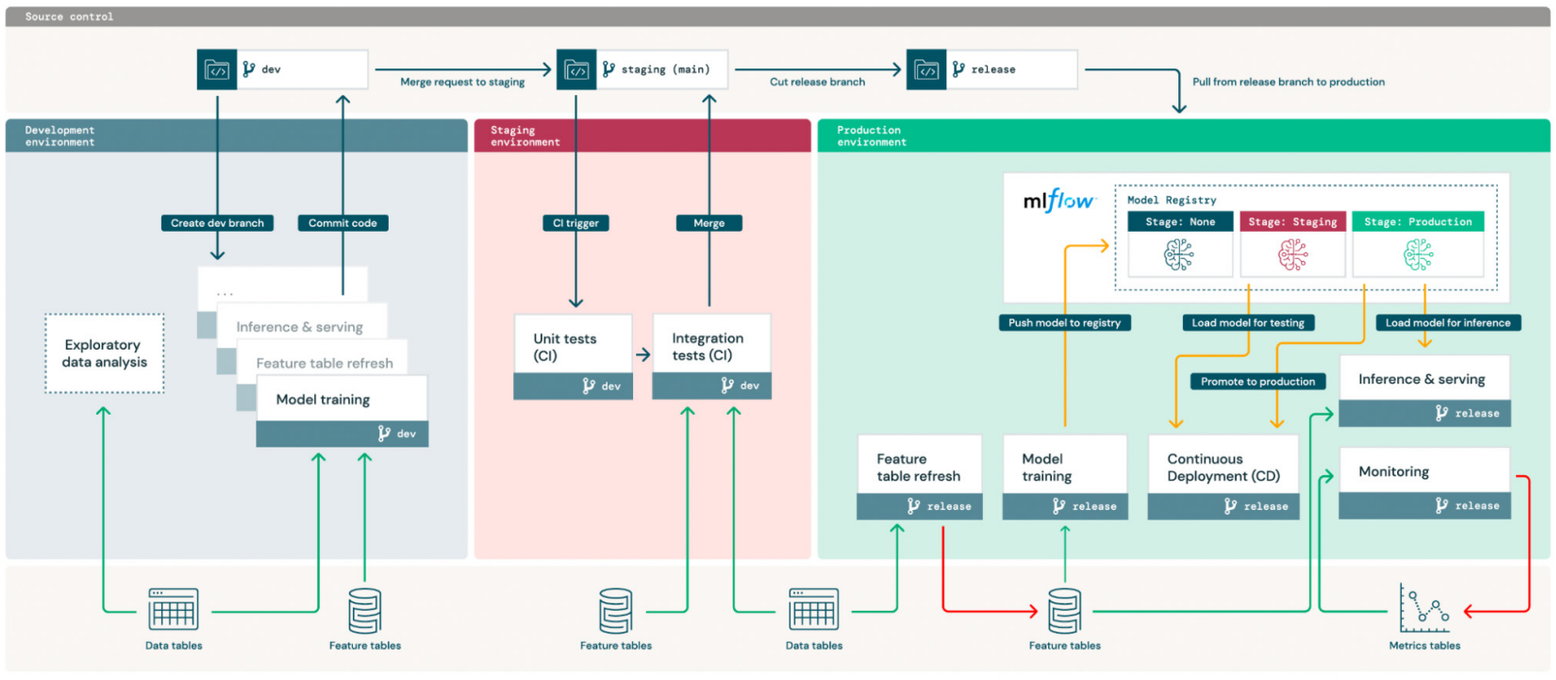
In summary, Unity Catalog system tables provide a powerful tool for organizations to enhance their data security and governance practices within the Databricks Data Intelligence Platform. By leveraging these tables, security teams can centralize and streamline their auditing processes, enabling them to gain comprehensive insights into data access and usage patterns. The integration of a zero trust architecture (ZTA) model further fortifies this approach, ensuring continuous monitoring and verification of all activities. As organizations continue to navigate the complexities of data security, the capabilities offered by Unity Catalog system tables will be instrumental in proactively identifying potential vulnerabilities, monitoring compliance and maintaining robust security standards at scale.

Governing Models and AI

Unity Catalog offers a suite of features that support generative AI models, enhancing their governance, deployment and overall lifecycle management. One of the key features is the Model Registry in Unity Catalog, which extends the benefits of Unity Catalog to ML models. This includes centralized access control, auditing, lineage and model discovery across workspaces. It also provides namespacing and governance for models, allowing you to group and govern models at the environment, project or team level.

As organizations look to scale their use of AI, a typical challenge has been bringing decision support systems based on AI to production. This is where MLOps and extensions of it like LLMOps play a critical role.

These frameworks, when properly implemented, meaningfully integrate data and AI operations, enhance collaboration and facilitate an efficient iteration loop that prioritizes *reproducibility*, *explainability* and *auditability*.



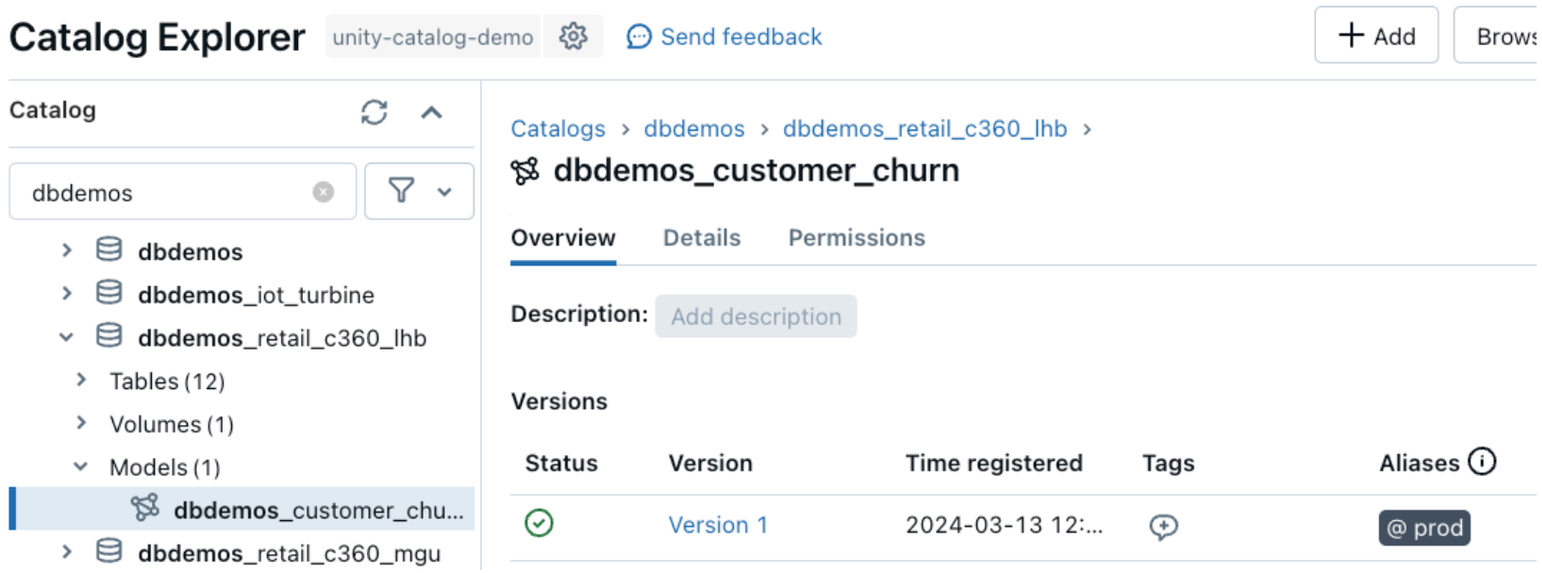
Model development, training and evaluation

In the "Tracking and Promoting Data Quality" chapter, we discussed the benefits of Unity Catalog lineage for data and model quality monitoring and tracking. This functionality is further enhanced by the use of MLflow, which allows you to keep track of the experiment and run that produced a model at a given time. You can use this information to understand the evolution of the model and its performance over time. Even more practically, this allows you to keep track of model versions and link them to specific data and code versions.

With model versioning, you can more easily compare how changes in parameters, data and code impact the overall performance of your models.

Unity Catalog unifies the management of modeling resources by giving development teams the option of having data, embeddings and models under a single schema. This extends the cataloging capabilities of data that is both structured (tables) and unstructured (volumes) to AI models.

This also aids in scalability and discoverability because as an organization's data needs grow, so too does the number of AI assets. A well-architected Unity Catalog structure helps manage this growth, ensuring that the number of use cases can scale smoothly without becoming unmanageable.



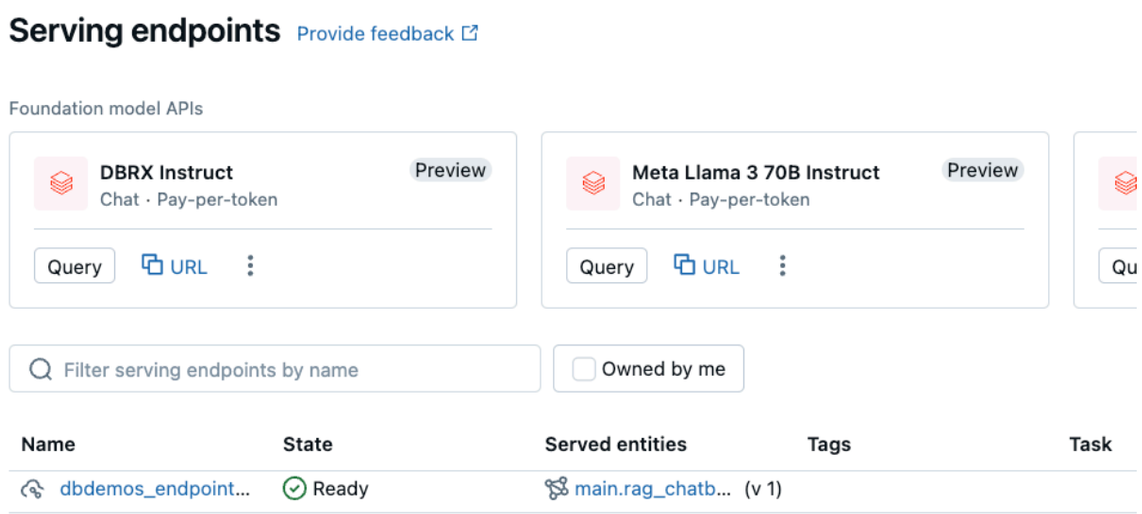
Model Serving

Model Serving in Unity Catalog offers a multitude of benefits that streamline the deployment, governance and querying of AI models. It provides a unified interface that allows you to manage all models in one location and query them with a single API, regardless of whether they're hosted on Databricks or externally. This simplifies the process of experimenting with, customizing and deploying models in production across various clouds and providers.

Through native integration with the Databricks feature store and Databricks Vector Search, Model Serving also simplifies the integration of features and embeddings into models. This allows for improved accuracy and contextual understanding, as models can be enhanced with additional context — for example when leveraging a RAG–based approach or fine-tuned with proprietary data — and deployed effortlessly.

Furthermore, the Model Serving UI allows you to centrally manage all model endpoints, including those that are externally hosted. You can manage permissions, track and set usage limits and monitor the quality of all types of models. This democratizes access to SaaS and open LLMs within your organization while ensuring appropriate guardrails are in place.

Model deployment via aliases is another feature that enhances the flexibility of model management. If the default catalog for your workspace is configured to a catalog in Unity Catalog, models registered using MLflow APIs are registered to Unity Catalog by default.



Security, safety and permissions

Securing models, AI assets and overall AI safety is an emerging area of focus. Unity Catalog allows for setting permissions on the use and access to models so that only authorized teams can modify and deploy selected models.

Moreover, model access to data can also be enforced to follow data access permissions. For example, consider a model that uses RAG to answer questions about a given subject by using Confluence pages. With Unity Catalog, data access permissions are respected and enforced throughout the whole chain. So if a user queries a model, the model crafts its answer using only Confluence pages the user has access to.

Elements of Unity Catalog discussed earlier in this chapter, like lineage and auditability, further enhance AI safety and security by providing full transparency on who accessed the model, what was done and when it was done.

This transparency, together with a robust data monitoring, evaluation and alerting system, can be used to capture issues with model performance and implement the appropriate corrective actions.

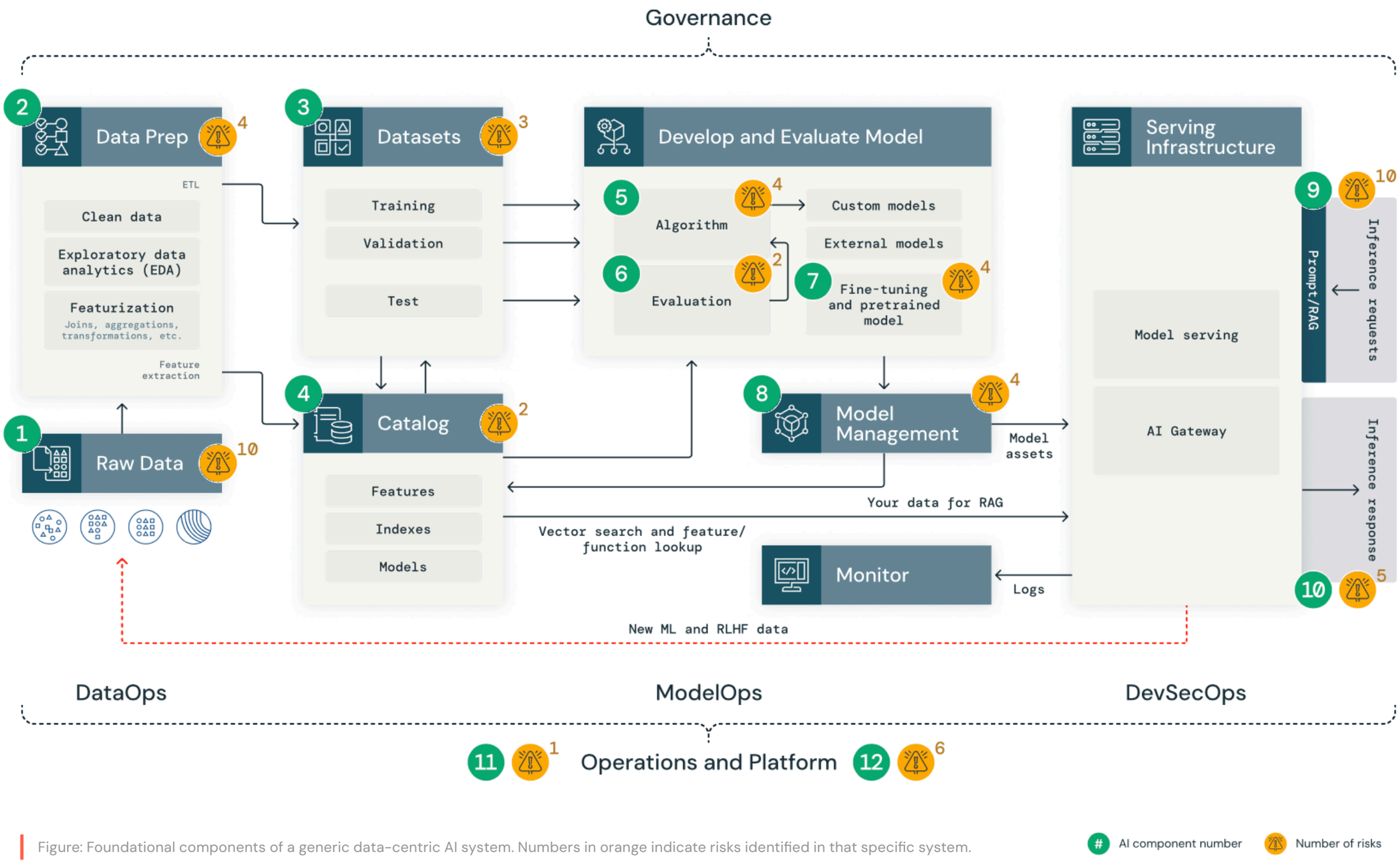


Figure: Foundational components of a generic data-centric AI system. Numbers in orange indicate risks identified in that specific system.

In summary, managing your models with the help of Unity Catalog provides the following benefits:

- **Centralized access control, auditing, lineage and model discovery** — Unity Catalog extends the benefits of centralized access control, auditing, lineage and model discovery across workspaces to ML models. It's compatible with the open source MLflow Python client.
- **Namespacing and governance for models** — You can group and govern models at the environment, project or team level. This allows for better control and organization of your models.
- **Chronological model lineage** — Unity Catalog tracks which MLflow experiment and run produced the model at a given time. This helps in understanding the evolution of the model over time.
- **Model Serving, versioning and deployment via aliases** — Unity Catalog supports Model Serving, versioning and deployment via aliases. For example, you can mark the "Champion" version of a model within your prod catalog. Models registered using MLflow APIs are registered to Unity Catalog by default.
- **Sharing models across workspaces** — To collaborate with other users on a registered model you created, you must grant ownership of the model to a group containing yourself and the users you'd like to collaborate with. Collaborators must also have the USE CATALOG and USE SCHEMA privileges on the catalog and schema containing the model.
- **Annotating models** — You can provide information about a model or model version by annotating it. This could include an overview of the problem or information about the methodology and algorithm used.

Sharing Your Data

In today's digital economy, seamlessly and securely sharing data and AI assets has become crucial for businesses that want to unlock the full potential of their data. The Databricks Data Intelligence Platform is designed for open collaboration with partners, customers and vendors. Unlike the restrictive systems of our competitors, Databricks offers an open platform that facilitates secure data sharing across various regions, clouds and platforms.

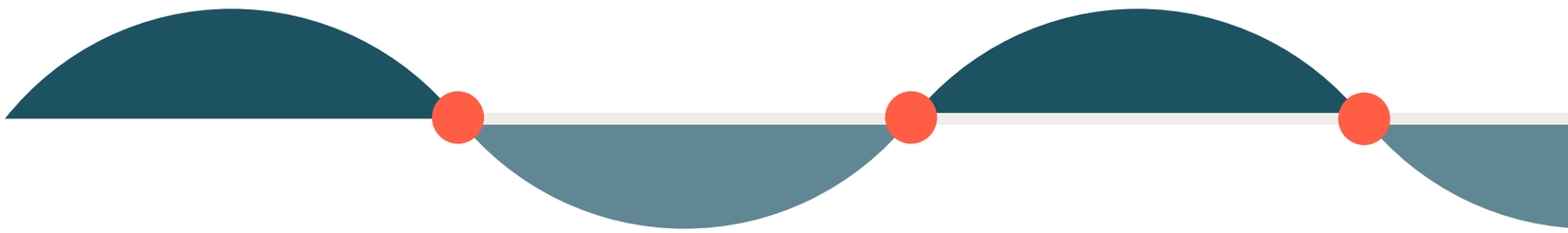
The backbone of this collaboration is Delta Sharing, which provides a flexible, secure method for sharing live data with any recipient, regardless of their infrastructure. Developed by Databricks and the Linux Foundation, Delta Sharing is the first open source approach to data sharing across data, analytics and AI. Customers can share live data and AI assets across platforms, clouds and regions with robust security and governance. Whether self-hosting the open source project or using the fully managed Delta Sharing on Databricks, both options provide a platform-agnostic, flexible and cost-effective solution for global data delivery. Databricks customers benefit from a managed environment that minimizes administrative overhead and integrates natively with Databricks Unity Catalog, offering a secure data-sharing experience.

Databricks commitment to innovation and collaboration has yielded significant results in the past year, with the ecosystem seeing impressive growth, including 16,000+ data recipients from a wide range of organizations that have adopted Delta Sharing to collaborate with partners and customers. Since we announced the open source Delta Sharing project three years ago, Delta Sharing has also continued to innovate and make it easy for customers to share live data and AI across platforms, clouds and regions — with no need for replication. We've continued developing new sharing capabilities for both data providers and data recipients, including:

- Volume sharing for large amounts of unstructured data (e.g., images, audio, videos or PDF files)
- AI Model Sharing for Delta Sharing to easily share models with partners and customers and deploy them in their Databricks environment
- Cross-platform view sharing to securely share views to any recipient across platforms
- Cloudflare R2 support to help joint customers take advantage of zero egress fees

Databricks Marketplace is powered by Delta Sharing and offers an open marketplace for discovering, evaluating and installing data and AI assets. Over the past year, Databricks Marketplace has introduced several new features such as foundation and proprietary AI models on Databricks Marketplace, volume sharing on Databricks Marketplace (see Shutterstock Uses Volume Sharing for Seamless Collaboration), Databricks to open sharing, private exchanges and Solution Accelerators to help data consumers discover and evaluate data products faster and accelerate their analytics and AI initiatives.

Databricks Clean Rooms provides a privacy-safe environment for collaboration for all your data and AI assets without direct access to sensitive data. Organizations are looking for ways to securely exchange their data and collaborate with external partners to foster data-driven innovations. In the past, organizations had limited data sharing solutions, relinquishing control over how their sensitive data was shared with partners, and little to no visibility into how their data was consumed. This created a risk for potential data misuse and data privacy breaches. Customers who tried using other clean room solutions have told us these solutions are limited and don't meet their needs because they often require all parties to copy their data into the same platform, don't allow sophisticated analysis beyond basic SQL queries and offer customers limited visibility or control over their data.



Organizations need an open, flexible and privacy-safe way to collaborate on data. Databricks Clean Rooms meets these critical needs. We recently announced the Public Preview of Databricks Clean Rooms for AWS and Azure.

- **Any cloud, any platform:** Secure, open, flexible collaboration is powered by Delta Sharing. Databricks Clean Rooms allows you to collaborate across clouds, regions and even across platforms using the new Sharing for Lakehouse Federation (see more about Lakehouse Federation).
- **Any language and workload of your choice:** Unlike other data clean rooms on the market, Databricks Clean Rooms supports any language or workload, including native support for ML and AI with Python. Databricks Clean Rooms is a flexible, interoperable solution, enabling organizations to collaborate with anyone, regardless of cloud or platform without the need for replication.
- **Any scale:** Databricks Clean Rooms also supports collaboration and operational capabilities at scale. With support for APIs, SQL commands and built-in Databricks Workflows orchestration, you can easily automate Databricks Clean Rooms workloads. Collaborators also get approved output data directly in their Unity Catalog that can be conveniently used for subsequent use cases. Coming soon, multiple collaborators can work together in Databricks Clean Rooms.

In this chapter, we'll explore the security architecture of Delta Sharing through three distinct scenarios:

- Databricks customer to Databricks customer
- Databricks customer to open sharing
- Cross-cloud data sharing

We'll highlight the advantages of incorporating Delta Sharing into a modern data collaboration strategy, such as improved operational efficiency through streamlined and secure data exchanges across various platforms and clouds while minimizing complexity and risk. This robust framework accelerates the time to insight, facilitating quicker decision-making while ensuring strong privacy protections that build stakeholder trust. Moreover, the versatility of Delta Sharing supports a wide array of data formats and applications, making it adaptable to evolving business requirements in a secure manner. Each scenario is accompanied by a customer testimonial that provides firsthand insight into the transformative impact of the solution.

Data sharing with Databricks Delta Sharing

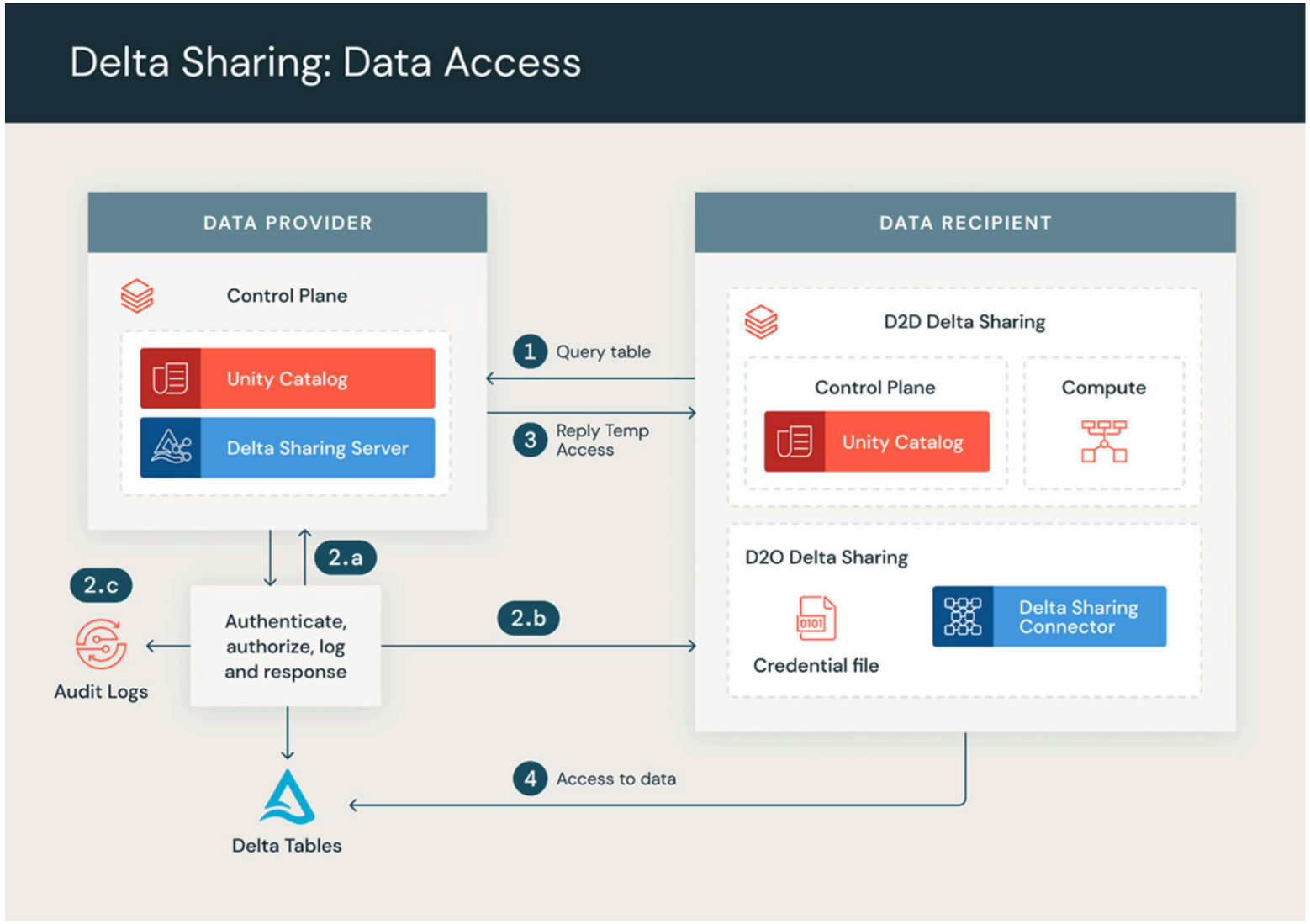
The following sections look at Databricks Delta Sharing scenarios, specifically when the data provider is utilizing the managed version of the Databricks Platform.

DATABRICKS TO DATABRICKS (D2D) DATA SHARING

The Databricks to Databricks (D2D) data sharing scenario illustrates secure, efficient data exchange between two Databricks customers within the Databricks ecosystem. It features managed connections and a no-token exchange system, ensuring simplicity and security. Customers benefit from Delta Sharing's native integration with Unity Catalog, which offers unified governance and security for sharing operations. Sharing isn't limited to data alone — Unity Catalog extends to volumes, notebooks and AI models, showcasing a broad range of functions. Intra-account sharing is enabled by default, while external sharing requires admin-level access for activation. Setting up Databricks Delta Sharing requires a Databricks workspace enabled for Unity Catalog and metastore, along with admin role privileges. Unity Catalog provides a unified governance layer throughout the sharing process — from creating a recipient and establishing shares to granting access. The Delta Sharing service processes API requests, performs thorough authorization checks and maintains detailed activity logs, ensuring transparent and secure operations.

Data access

Unity Catalog plays a crucial role in postauthorization data access. It determines the access method — either cloud tokens or presigned URLs — based on asset type and sharing arrangement. For cloud tokens, a read-only, scoped-down SAS token is created by the provider's Unity Catalog and forwarded to the recipient's compute plane, providing secure, limited-time storage access to the table root directory. Similarly, presigned URLs are generated and sent to the recipient's compute plane for secure, temporary access to storage files. This methodology ensures data sharing is both flexible and secure, meeting a wide array of business needs.


Delta Sharing: Data Access
DATA PROVIDER
Control Plane
Unity Catalog
Delta Sharing Server
1 Query table
3 Reply Temp Access
2.a
2.c
Authenticate, authorize, log and response
2.b
Audit Logs
Delta Tables
4 Access to data
DATA RECIPIENT
D2D Delta Sharing
Control Plane
Unity Catalog
Compute
D2O Delta Sharing
Credential file
Delta Sharing Connector

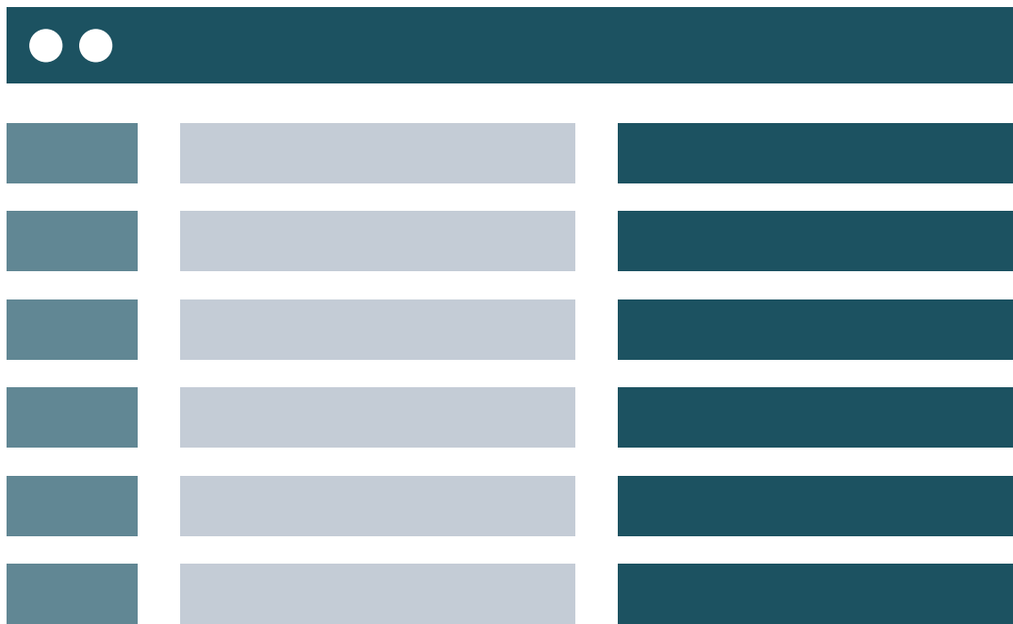
DATABRICKS TO OPEN (D2O) DATA SHARING

In the Databricks to Open (D2O) data sharing scenario, strict security protocols are upheld for Databricks customers sharing data with external third-party users not on the Databricks Platform. Recipients can directly connect to shared data using Delta Sharing connectors that support various systems like pandas, Tableau, Apache Spark™, Rust and more without needing a specific compute platform.

Upon creating an open recipient in Databricks, a secure, one-time activation URL is generated, allowing the recipient to download a credential file containing a Delta Sharing endpoint address and a token. Providers can take immediate action in case of a security breach, such as changing recipient credentials or revoking read permissions.

Data access workflow

When a recipient queries a shared table using the supported connectors, Delta Sharing verifies the recipient using tokens from the credential file and provides presigned URLs for accessing the data. This ensures compatibility with various open source connectors, safeguarding the integrity and security of the shared assets.



CROSS-CLOUD DATA SHARING

Many enterprises adopt cross-cloud strategies to support diverse functionalities across different cloud platforms, facilitate partnerships or integrate data after acquisition. Delta Sharing enables seamless and secure sharing within and across multiple cloud platforms, ensuring operational continuity, fostering innovation and driving growth. Delta Sharing cross-platform sharing capabilities support multicloud environments, making it a clear advantage. It facilitates secure and efficient data exchanges whether within a single cloud or across multiple cloud platforms. Many customers have reported that Delta Sharing promotes interoperability and enhances security across their cloud ecosystems.

Network and storage configuration

Delta Sharing integrates seamlessly with the cloud's native storage security architecture without significant modifications. Designed for Databricks on Azure, AWS and GCP, it aligns with Unity Catalog's requirements and supports data sharing through cloud storage solutions (ADLS Gen2, S3, GCS) using private communication channels or IP address allowlisting for enhanced security. The network and storage configuration works across both intracloud and cross-cloud scenarios, ensuring secure data exchange within the same cloud ecosystem using private endpoints, storage firewalls and network gateways, and leveraging existing cross-cloud private connections for secure data access across different platforms.

Data filtering

Delta Sharing employs two primary methods for data filtering:

- **Partition filtering:** Shares specific table partitions based on recipient properties, allowing controlled access to needed data portions
- **Dynamic views:** Enables sharing subsets of data with recipients using dynamic functions like `current_recipient`, offering fine-grained control over data access and improved manageability

Security, flexibility and seamless integration with Databricks Delta Sharing

Data sharing and collaboration across organizations and platforms are vital for modern business operations. Delta Sharing, a cutting-edge open data sharing protocol, empowers organizations to securely share and access data across diverse platforms, emphasizing security and scalability without vendor or data format constraints.

It's important to focus on exploring data replication options within Delta Sharing, providing architectural guidance for specific data sharing scenarios. Drawing from our extensive experience with Delta Sharing clients, our objective is to reduce egress costs and improve performance by offering targeted data replication alternatives. While live sharing is suitable for many cross-region data sharing scenarios, there are instances where replicating the entire dataset and establishing a data refresh process for local regional replicas prove to be more cost efficient. Delta Sharing facilitates this through Cloudflare R2 storage, change data feed (CDF) Delta Sharing and Delta deep cloning functionalities. These capabilities make Delta Sharing highly valued for its exceptional flexibility in meeting diverse data sharing needs.

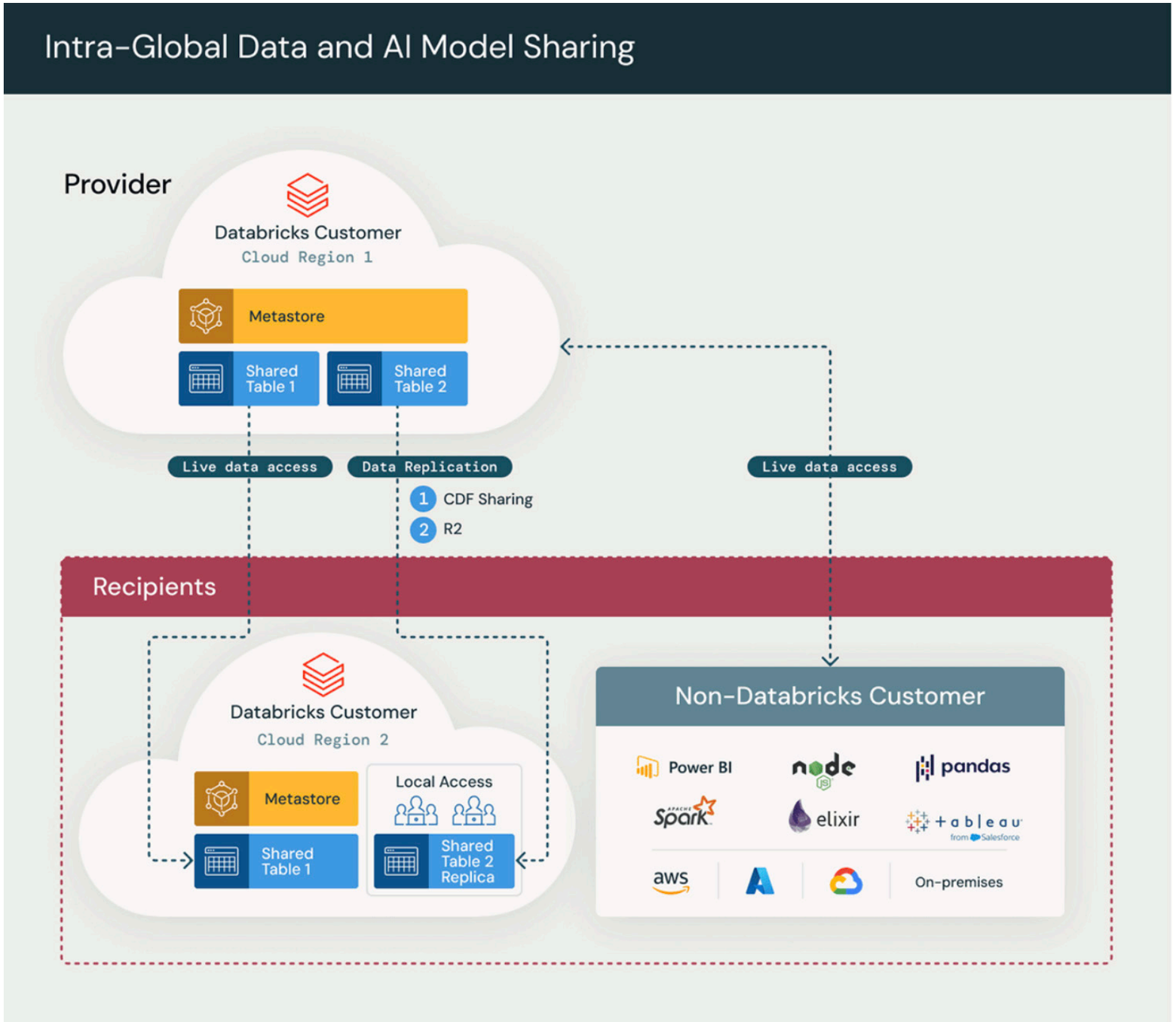
Since its general availability in August 2022, Delta Sharing on Databricks has seen widespread adoption across various collaboration scenarios. Let's explore two common architectural patterns where Delta Sharing has played a pivotal role in enabling and enhancing critical business scenarios:

- Intra-enterprise cross-regional data sharing
- Data aggregator (hub and spoke) model

INTRA-ENTERPRISE CROSS-REGIONAL DATA SHARING

In this scenario, Delta Sharing enables the sharing of data across regions, such as when a QA team is in different regions or a reporting team needs global business activity data. Typically, this involves:

- **Sharing large tables:** Sharing large tables in real time, where recipients execute diverse queries with different predicates. An example is clickstream and user activity data.
- **Local replication:** To enhance performance and manage egress costs, data is replicated to create a local copy, especially when a significant number of users in the recipient's region frequently access these tables.



In this scenario, the business units for both the data provider and the data recipient share the same Unity Catalog account but have different metastores on Databricks.

Intraglobal data and AI model sharing

The high-level architecture of the Delta Sharing solution highlights key steps in the process:

1. **Creation of a share:** Live tables are shared with the recipient for immediate data access
2. **On-demand data replication:** Generating a regional duplicate of the data improves performance, reduces the need for cross-region network access and minimizes egress fees

Data replication can be achieved through:

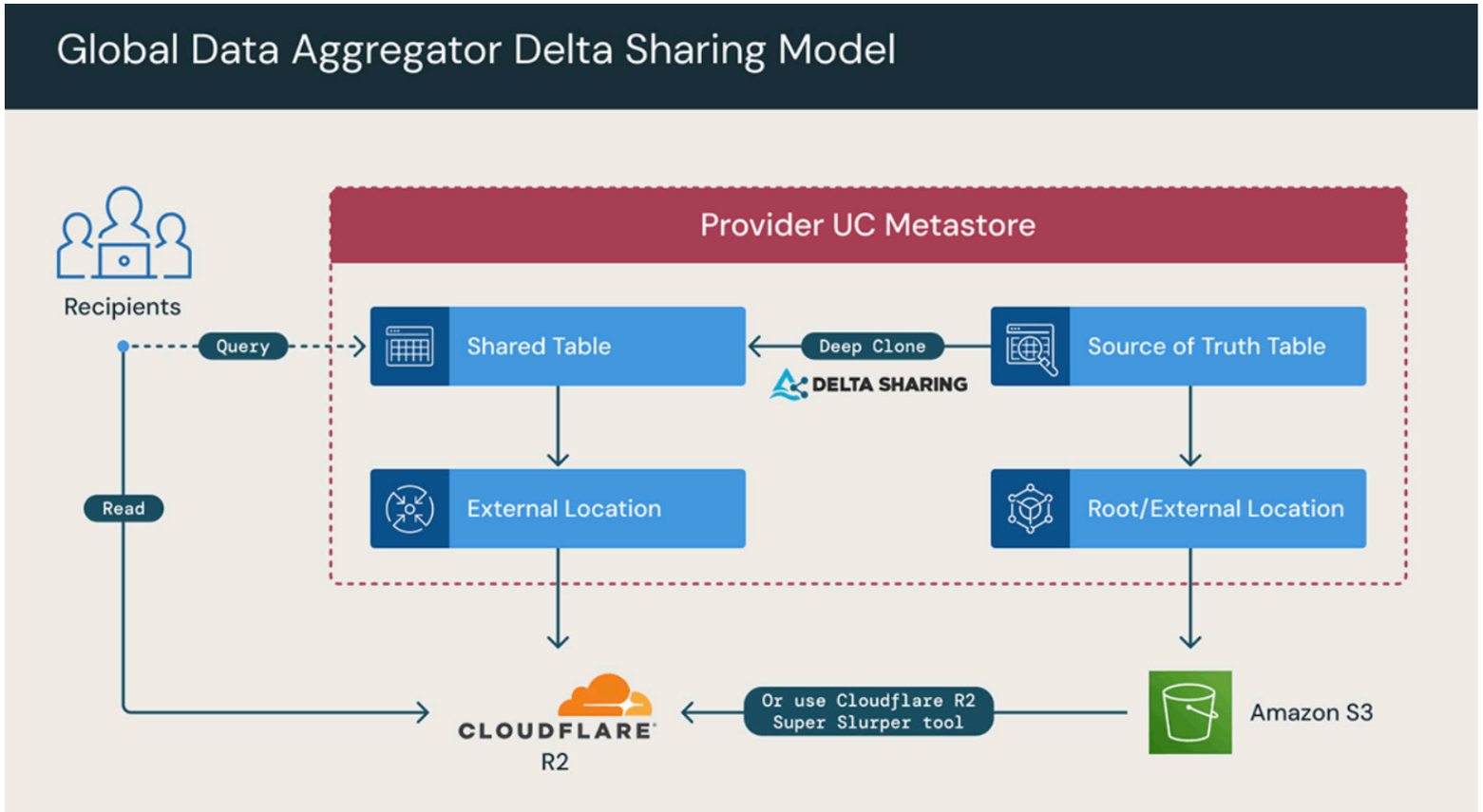
- **Change data feed (CDF) on a shared table:** Sharing table history and enabling CDF for incremental data updates
- **Cloudflare R2 with Databricks:** Using R2 for large-scale data sharing without egress charges
- **Delta deep clone:** Copying source table data and metadata to the clone target within the same Databricks cloud account for efficient data refresh

DATA AGGREGATOR (HUB AND SPOKE) MODEL

This scenario involves sharing data with clients, particularly for data aggregator enterprises collecting and merging data from various sources. This model supports:

- **Connecting recipients across clouds:** AWS, Azure and GCP
- **Supporting a diverse platform:** From Python code to Excel spreadsheets
- **Scalability:** For the number of recipients, shares and data volumes

Using Cloudflare R2 with Databricks simplifies and secures data sharing, eliminating egress fees and complex data transfers. This integration is currently in Private Preview.



In conclusion, Delta Sharing is a cornerstone of the Databricks Data Intelligence Platform, offering secure, flexible and cross-platform data sharing capabilities. Supporting both structured and unstructured data, as well as AI models, Delta Sharing stands out among data exchange platforms. It's widely trusted across industries, as reflected in customer testimonials highlighting its significant impact on operational efficiency and innovation. As the data sharing landscape evolves, Delta Sharing remains future-proof, prioritizing security, flexibility and seamless integration across diverse ecosystems and making it an invaluable asset for enterprises worldwide.

Upgrading to Unity Catalog

With Databricks Unity Catalog, governance capabilities have undergone a significant transformation, overcoming the limitations of the traditional Hive metastore which used to be the default catalog for Databricks workspaces. Unity Catalog represents a paradigm shift towards a centralized governance solution compared to the former approach of distributed governance at the workplace level, and it requires careful planning, attention to detail and collaboration among multiple stakeholders to ensure successful implementation.

The process for upgrading to Unity Catalog

Upgrading to Unity Catalog is a seamless process enabled by inbuilt capabilities for table migration and by using utilities like UCX that are purpose-built for the upgrade process. The UCX tool streamlines the process of assessing existing workspaces to identify assets requiring migration and automates this task. It also aids in planning the upgrade process and determining its complexity.

Once the assessment is completed, for clarity and ease of understanding, the process can be broken down into different stages, which are outlined in the following sections.

GLOBAL SETUP

Global setup involves tasks that are performed once for the entire Databricks Unity Catalog upgrade process.

- **Account console setup:** The Databricks account console is the centralized hub to manage all the workspaces within an enterprise Databricks subscription. An account admin role allows access to the account console to manage workspaces, users and metastores centrally. Identity federation including the automated user sync using SCIM is also set up at the account console.
- **Architecture design:** The upgrade of Unity Catalog warrants a well-designed architecture that aligns with the upgraded governance patterns enforced by Unity Catalog. By adhering to these best practices, the user experience is optimized, ensuring a smooth transition without impacting existing downstream consumers of the system.
- **Set up the metastore:** The metastore acts as the top-level container for data governed by Unity Catalog for a specified cloud region. Unity Catalog metastores register metadata about securable objects (such as tables, volumes, external locations and shares) and the permissions that govern access to them.

PER-APPLICATION TASKS

These tasks are carried out for each individual application during the upgrade process and may vary based on the deployment architecture, specifically whether one application is part of one workspace or multiple applications are deployed in the same workspace.

- **Set up the catalogs:** The first layer of the object hierarchy in Unity Catalog is the *catalog*. It serves as the top-level container for organizing and managing all data assets, including tables, views, functions, volumes and machine learning models. By setting up meaningful catalogs, teams can efficiently structure and govern their data assets, enhancing data discoverability and productivity.
- **Code changes:** The upgrade to Unity Catalog also introduces features such as governed file systems like volumes, which replace DBFS, and a three-level namespace. This allows for more efficient and organized data management. However, it's essential to note that any unsupported code patterns must be migrated to Unity Catalog–compatible ones. For instance, any RDD operations should be updated to use DataFrames in the Spark API.

PER-WORKSPACE TASKS

These tasks are completed for each workspace as part of the Databricks Unity Catalog upgrade.

- **Enable the workspace:** A workspace is enabled for Unity Catalog — and in turn identity federation — when a metastore is attached to the workspace from the account console.
- **Upgrade tables and workflows:** Existing tables can be upgraded, and the process depends on whether it's a managed or an external table. Databricks Workflows needs to be updated to use the Unity Catalog–enabled compute.

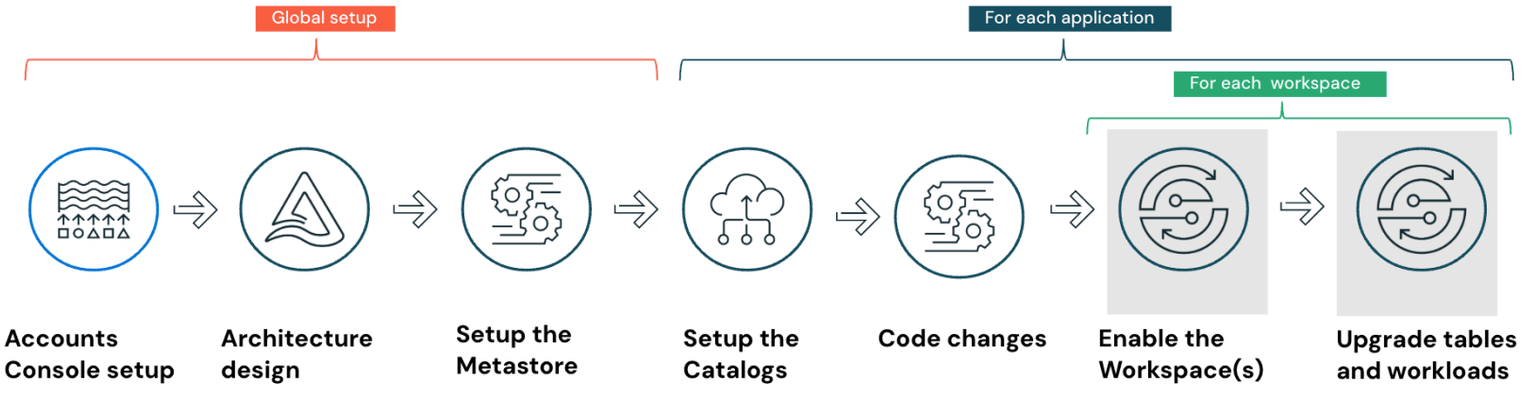


Figure: Unity Catalog upgrade process

IDENTITY FEDERATION

When Unity Catalog is enabled for a workspace, it also enables identity federation by default. With identity federation, you configure Databricks users, service principals and groups once in the account console, rather than repeating configuration separately in each workspace. This both reduces friction in onboarding a new team to Databricks and enables you to maintain one SCIM provisioning application with your identity provider to the Databricks account, instead of a separate SCIM provisioning application for each workspace. Once users, service principals and groups are added to the account, you can assign them permissions on workspaces. You can only assign account-level identities access to workspaces that are enabled for identity federation.

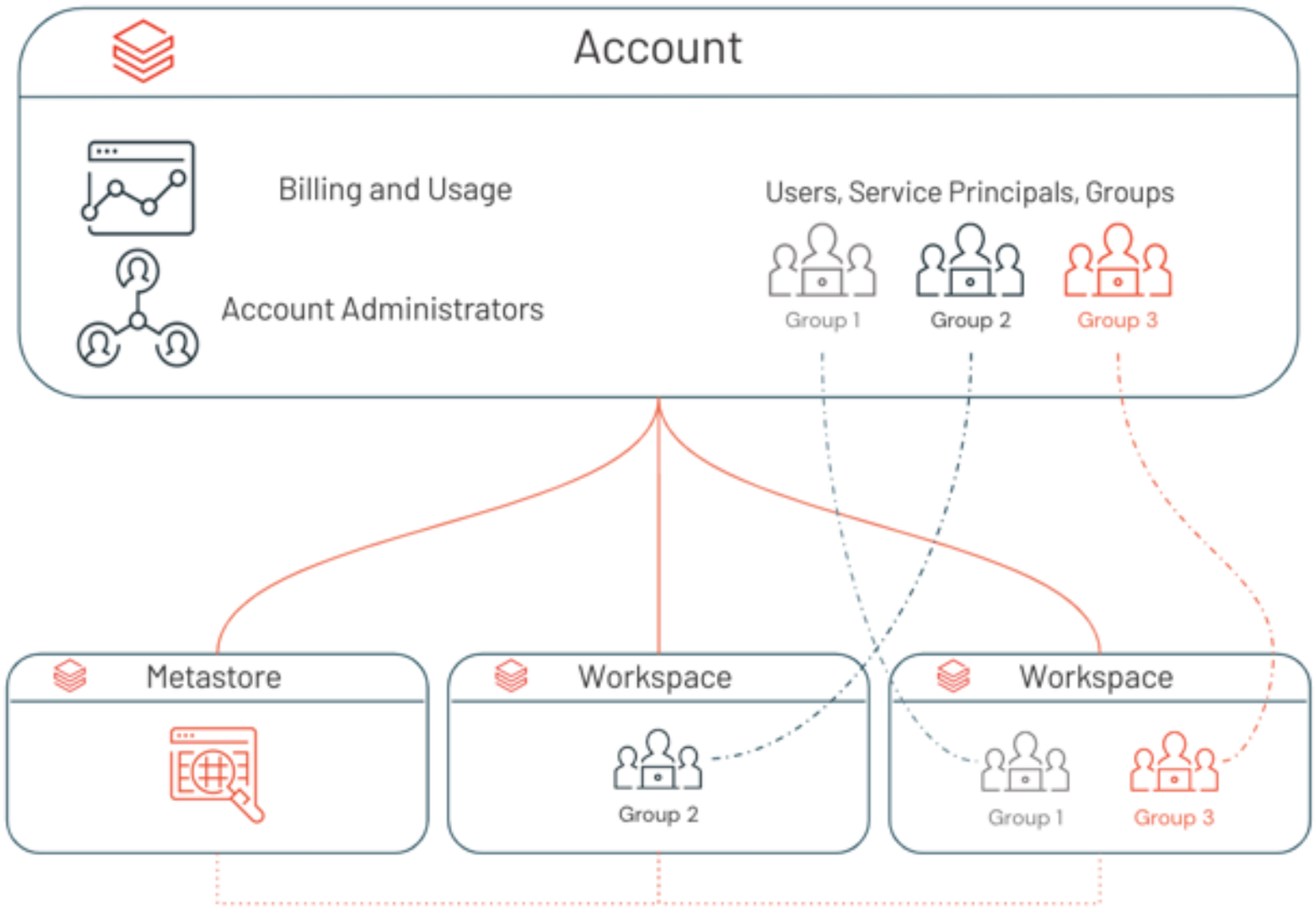


Figure: Identity federation with principals at the account level

Migrate workspace-level SCIM provisioning to the account level

If you have workspace-level SCIM provisioning set up in your workspace, you should set up account-level SCIM provisioning and turn off the workspace-level SCIM provisioner. Workspace-level SCIM will continue to create and update workspace-local groups. Databricks recommends using account groups instead of workspace-local groups to take advantage of centralized workspace assignment and data access management using Unity Catalog. Workspace-level SCIM doesn't recognize account groups that are assigned to your identity federated workspace, so workspace-level SCIM API calls will fail if they involve account groups.

> **Note:** Microsoft Entra ID doesn't support the automatic provisioning of nested groups to Azure Databricks. Microsoft Entra ID can only read and provision users that are immediate members of the explicitly assigned group.

ADMIN ROLES IN DATABRICKS

As part of the upgrade process, a few key users need to be identified for the admin roles within the platform.

The **account admin** is responsible for:

- Creating workspaces
- Creating and configuring metastores
- Creating users, groups and service principals
- Granting users access to workspaces
- Setting billing budget threshold alerts
- Enabling system tables and delegating access to them

The **metastore admin** is responsible for:

- Creating CATALOG, CONNECTION, STORAGE CREDENTIAL, EXTERNAL LOCATION, FOREIGN CATALOG
- Creating SHARE, RECIPIENT, PROVIDER, ALLOWLIST, MATERIALIZED VIEW
- Changing OWNER of any securable object

> **Note:** Through their ability to transfer ownership of all objects, metastore admins have the ability to grant themselves read and write access to any data in the metastore. There's no direct access by default. Granting of permissions is audit-logged. A metastore admin role is an optional role, and the responsibilities can be delegated to another group/user or the workspace admin.

The **workspace admin** is responsible for:

- Adding users and groups to workspace
- Creating clusters and cluster policies
- Changing OWNER of clusters, workflows, notebooks, queries, dashboards

If your workspace was enabled for Unity Catalog automatically, workspace admins have the following privileges on the attached metastore by default.

- Creating CATALOG, EXTERNAL LOCATION, STORAGE CREDENTIAL
- Creating CONNECTION, SHARE, RECIPIENT, PROVIDER, MATERIALIZED VIEW

Apart from the admin roles, the **data owner** role is also available and is responsible for:

- Changing ownership of securable objects (CATALOG, SCHEMA, TABLE, VIEW, etc.)
- Granting any privilege to any principal

> **Note:** The ownership of the data can be defined at the catalog, schema and table levels. Each securable object in Unity Catalog has an owner. The owner can be any user, service principal or account group, known as a *principal*. The principal that creates an object becomes its initial owner.

Using the Databricks Labs UCX tool for accelerating migration

UCX is a Databricks Labs project that provides tools to help you assess your environment and help you upgrade your non-Unity Catalog workspace to Unity Catalog.

Assessing the existing workspace (assessment workflow)

One of the key benefits of the UCX tool is its ability to save time and resources by automatically assessing and identifying assets that need migration. This allows businesses to focus on other critical aspects of the upgrade process and make informed decisions about their workspace upgrades. UCX accelerates the Unity Catalog upgrade process by deploying an assessment job which generates a detailed report on the assets that need to be migrated.

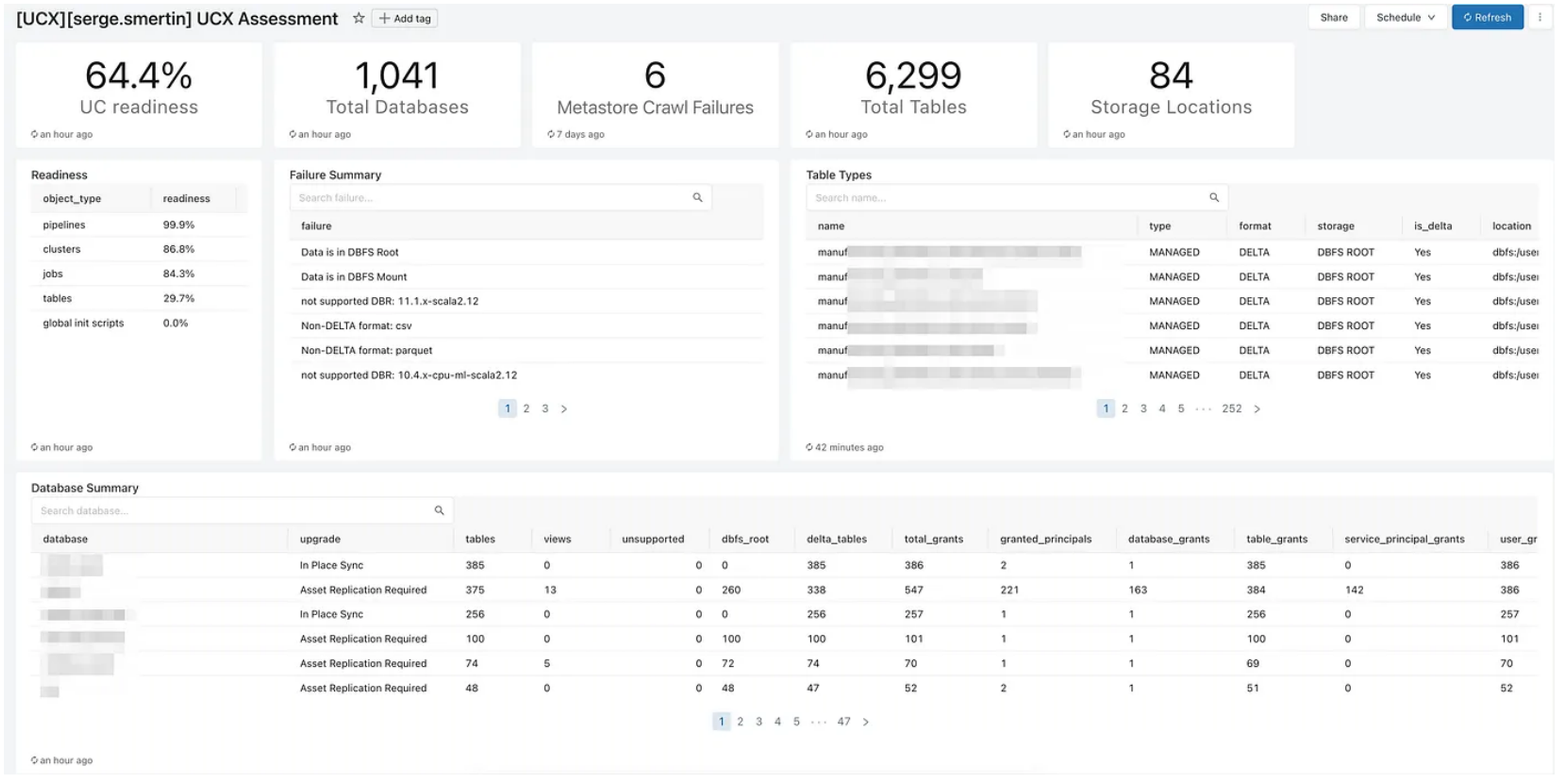


Figure: Sample UCX assessment report

After the assessment report becomes available, the upgrade planning process can commence by examining the report for complexity and the number of assets, including tables, groups, storage locations and more. The Unity Catalog upgrade can be executed on a per-application or per-workspace basis, and it's recommended that you carry out the upgrade in a phased manner to reduce downtime during the process.

Group migration from workspace to the account (group migration workflow)

Unity Catalog introduces identities at the identity federation and account levels, enhancing security and streamlining access management. Identity federation enables you to configure users, service principals and groups in the account console and then assign those identities access to specific workspaces. This simplifies Databricks administration and data governance. However, groups created or synced to existing workspaces using the SCIM API won't be supported in Unity Catalog and must be upgraded to the account level to benefit from these new features.

The UCX tool automates the migration of groups from workspace to the account level and helps you to upgrade all Databricks workspace assets: legacy table ACLs, entitlements, AWS instance profiles, clusters, cluster policies, instance pools, Databricks SQL warehouses, Delta Live Tables, jobs, MLflow experiments, MLflow Model Registry, SQL dashboards and queries, SQL alerts, token and password usage permissions that are set on the workspace level, secret scopes, notebooks, directories, repositories and files.

Table migration from Hive metastore to Unity Catalog (table migration workflow)

To enable asset governance, existing tables in the Hive metastore must be migrated to Unity Catalog. The migration process varies depending on whether the table is an external table or a managed table in Hive. External tables in the Hive metastore are upgraded as external tables in Unity Catalog, using SYNC. Managed tables in the Hive metastore that are stored in workspace storage (also known as DBFS root) are upgraded as managed tables in Unity Catalog using DEEP CLONE.

Hive-managed tables must be in Delta or Parquet format to be upgraded. External Hive tables must be in one of the following data formats:

- DELTA
- CSV
- JSON
- AVRO
- PARQUET
- ORC
- TEXT

Note: UCX, like all projects in the Databricks Labs GitHub account, is provided for your exploration only and isn't formally supported by Databricks with service-level agreements (SLAs). It's provided as is. We make no guarantees of any kind. Don't submit a Databricks support ticket relating to issues that arise from the use of this project. Instead, file a GitHub issue. Issues will be reviewed as time permits, but there are no formal SLAs for support.

Upgrading assets

The following sections document the different assets involved in upgrading to Unity Catalog.

UNITY CATALOG ARTIFACT CREATION

There are numerous artifacts that should be created before starting your upgrade process.

Metastore

A metastore is the top-level container that stores metadata about all of your data assets (such as tables, volumes, external locations and shares) and the access permissions associated with them. You should have one metastore for each region in which your organization operates. To enable a workspace with Unity Catalog, you must attach the workspace to a metastore. You can find the steps to create a metastore and attach it to the workspace in the official documentation.

Storage credentials

Storage credentials authenticate and authorize Databricks to access your data stored in cloud storage. Principals can be granted access to it. Storage credentials are primarily used to create external locations.

External locations

An external location is a reference to a location in cloud storage. When you create an external location, you specify the storage credentials to use and the path to the location in the cloud storage.

Catalogs

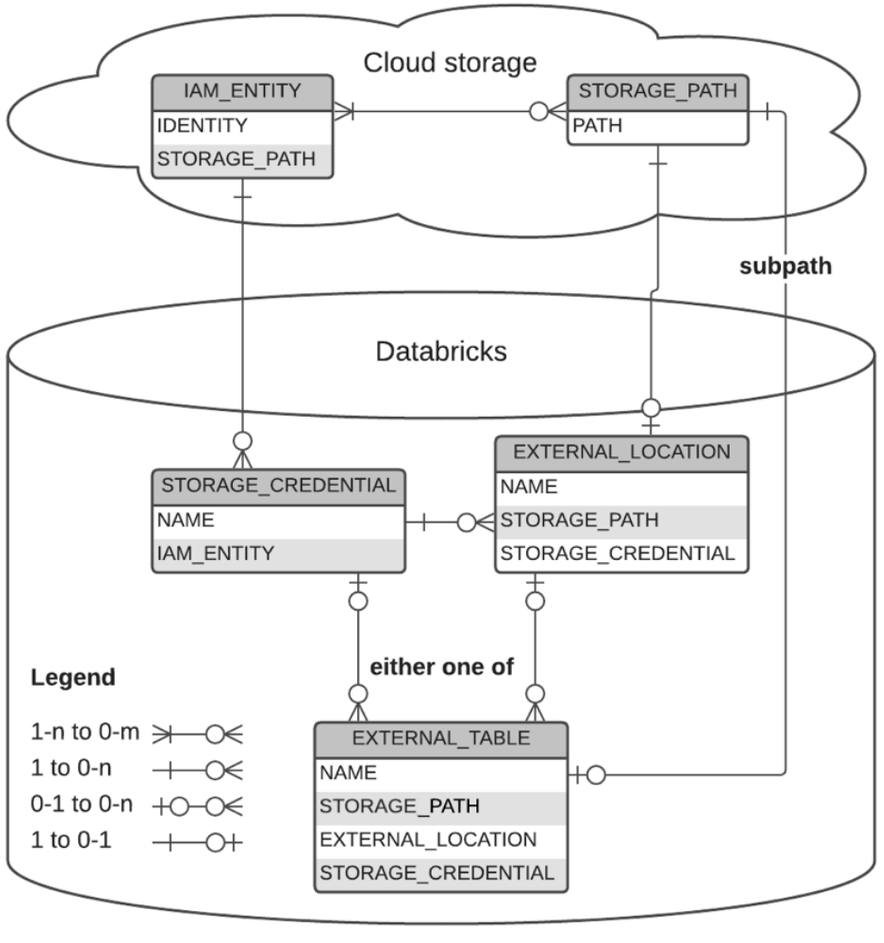
A catalog in Databricks is a container for a collection of schemas. It provides a way to organize your databases in a hierarchical manner, making it easier to manage and access them.

Schemas

A schema in Databricks is a logical grouping of data and AI assets like tables, views, volumes, functions and models. It provides a way to organize your data and AI assets, making it easier to manage and access them.

Volumes

Volumes are Unity Catalog objects representing a logical volume of storage in a cloud object storage location. Volumes provide capabilities for accessing, storing, governing and organizing files. Volumes can be either managed or external.



UPGRADING TABLES

When it comes to table upgrade, the main goal is to avoid or minimize data movement as much as possible.

Remember, a table can either be managed or external. The location of the data can either be on DBFS root storage location, DBFS mounted cloud object storage or directly specified cloud storage (such as S3://, abfss:// or gcs://). The table file format and interface can be a file format such as Delta, Parquet, Avro or an interface such as Hive SerDe.

For managed tables, there are multiple scenarios:

- **Tables on DBFS root location:** The data files for the managed tables reside within DBFS root, which is the default location for the Databricks-managed HMS database
- **Tables on DBFS mounted location:** This is when the parent database has its location set to external paths, e.g., a mounted path from the object store
- **Tables on cloud storage location:** This is when the parent database has its location set to external paths, e.g., a cloud object store

For external tables, you can have the following scenarios:

- **Tables on DBFS root location:** The data files for the external tables reside within DBFS root, which is the default location for the Databricks-managed HMS database. The table definition has the "location" clause which makes the table external.
- **Tables on DBFS mounted location:** This is when the table is created with a location clause with a path specifying a mounted path from the object store
- **Tables on cloud storage location:** This is when the parent database has its location set to external paths, e.g., a cloud object store
- **Hive SerDe tables:** These are the tables which were created using the Hive SerDe interface
- **Azure tables:** Tables on Gen 1 blob storage require managed tables to be moved to a Gen 2 blob storage prior to upgrade in order to avoid a schema error

MIGRATION MATRIX

The following matrices identify the different scenarios and the methodology for upgrading them.

HMS storage format using DBFS root storage

HMS Table Type	Target Unity Catalog Table Type	Target Unity Catalog Data File Format	Methodology
Managed	External or Managed	1. Delta for managed (Preferred) 2. As that of the HMS source data file format or Delta (Preferred) in case of external Note: Preferably change it to Delta while you're doing the migration using CTAS	CTAS
External	External or Managed	1. Delta for managed (Preferred) 2. As that of the HMS source data file format or Delta (Preferred) in case of external Note: Preferably change it to Delta while you're doing the migration using CTAS	CTAS

HMS Hive SerDe table

HMS Table Type	Target Unity Catalog Table Type	Target Unity Catalog Data File Format	Methodology
Hive SerDe external or managed Note: Regardless of the underlying storage format, Hive SerDe follows the same migration path	External or Managed	1. Delta for managed (Preferred) 2. As that of the HMS source data file format or Delta (Preferred) in case of external Note: Preferably change it to Delta while you're doing the migration using CTAS	CTAS

HMS storage format using DBFS mounted storage

HMS Table Type	Target Unity Catalog Table Type	Target Unity Catalog Data File Format	Methodology
Managed	External	As that of the HMS source data file format	1. Run Sync to create Unity Catalog external table 2. Convert HMS managed to HMS external (code provided below) 3. Drop HMS table after all dependencies are resolved so that there's no way to access the data using HMS table 4. Unmount the mount point after all dependencies are resolved so that there's no way to access the data using mount points Note: Ensure that the HMS table is dropped individually after conversion to an external table. If the HMS database/schema was defined with a location and if the database is dropped with the cascade option, the underlying data will be lost and the migrated Unity Catalog tables will lose the data.
Managed	Managed	Delta	CTAS
External	External	As that of the HMS source data file format	1. Run Sync to create Unity Catalog external table 2. Drop HMS table after all dependencies are resolved so that there's no way to access the data using HMS table 3. Unmount the mount point after all dependencies are resolved so that there's no way to access the data using mount points
External	Managed	Delta	CTAS

HMS storage format using cloud object storage

HMS Table Type	Target Unity Catalog Table Type	Target Unity Catalog Data File Format	Methodology
Managed	External	As of the source data file format	1. Run Sync to create Unity Catalog external table 2. Convert HMS managed to HMS external (code provided below) 3. Drop HMS table after all dependencies are resolved so that there's no way to access the data using HMS table Note: Ensure that the HMS table is dropped individually after conversion to an external table. If the HMS database/schema was defined with a location and if the database is dropped with the cascade option, the underlying data will lose the data.
Managed	Managed	Delta	CTAS
External	External	As of the source data file format	1. Run Sync to create Unity Catalog external table 2. Drop HMS table after all dependencies are resolved so that there is no way to access the data using HMS table
External	Managed	Delta	CTAS

Use this code to upgrade HMS managed to HMS external:

```
import org.apache.spark.sql.catalyst.catalog.{CatalogTable,
CatalogTableType}
import org.apache.spark.sql.catalyst.TableIdentifier
val tableName = "table"
val dbName = "dbname"
val oldTable: CatalogTable =
spark.sessionState.catalog.getTableMetadata(TableIdentifier(tableName,
Some(dbName)))
val alteredTable: CatalogTable = oldTable.copy(tableType =
CatalogTableType.EXTERNAL)
spark.sessionState.catalog.alterTable(alteredTable)
```

Before upgrading the views, make sure that you've upgraded all of the referenced tables and views to Unity Catalog. After you upgrade all of a view's references to the same Unity Catalog metastore, you can re-create a new view that references the new tables.

If dynamic views are present, they should be modified to use the `is_account_group_member` function. Additionally, use the row filters and column masking to enable fine-grained access at the row and column levels. Functions also need to be re-created in the corresponding catalog - schema.

We recommend you use Databricks Labs UCX to manage the table, view and function upgrades seamlessly, instead of developing custom scripts.

UPGRADING CLUSTERS TO UNITY CATALOG

Prerequisites

Before initiating the upgrade, ensure that Unity Catalog is enabled for your account. A metastore should be created and assigned to the workspace, and catalogs/databases/tables should be created with proper access control lists (ACLs). The cluster should operate in either shared access mode or single user access mode to leverage Unity Catalog. Shared access mode is recommended for most scenarios unless your workload requires certain functionalities supported by this mode. For production workloads, it's recommended to restrict the access modes using cluster policies to ensure consistent usage of Unity Catalog–enabled clusters.

Upgrade scenarios

Consider the following scenarios when upgrading:

- **Clusters running Databricks Runtime (DBR) earlier than 11.1:** In this case, you must dismantle the existing cluster and rebuild it. Upgrading clusters to use Unity Catalog isn't possible in this scenario as Unity Catalog requires DBR 11.1 or later. Upgrade the DBR to version 11.1 or later, with the latest long-term support (LTS) version being highly recommended.
- **Clusters running DBR 11.1 and later:** To upgrade from any of the legacy cluster modes to the Unity Catalog–leveraged security mode, edit your cluster and change the access mode to either single user or shared from the access mode drop-down menu in the cluster UI.

Interactive and automated clusters

For an interactive cluster, switch to Unity Catalog mode by clicking the Edit button on the cluster page and changing the access mode to either single user or shared.

For job clusters, the prerequisites are the same as for interactive clusters. Navigate to the job details screen and click Configure under Compute.

To minimize effort, change and fix/default the access mode in the compute policy and roll out the changes to all existing clusters. Also, ensure that credential pass-through isn't enabled in a cluster (either in Unity Catalog or fixed in compute policies), as this would disable Unity Catalog access. To add a default catalog, use the following configuration in the Spark settings of the cluster: `spark.databricks.sql.initial.catalog.name name_of_catalog`.

Limitations

Note that Databricks Runtime for Machine Learning and Spark machine learning library (MLlib) aren't supported in shared access mode. Spark-submit jobs are also not supported in shared access mode. Single-node cluster isn't supported in the SHARED mode.

The use of dynamic views for row-level or column-level security isn't supported in single user mode. Also, to read from a view, you must have SELECT on all referenced tables and views in this mode. Refer to this documentation to see the full list of limitations in both access modes.

UPGRADING MOUNT POINTS AND DBFS

Mount points and file storage in DBFS aren't supported by Unity Catalog, so all data stored in mount points or DBFS needs to be moved to external tables or volumes.

Prior to Unity Catalog, mount points were used to read tabular data from paths to the object storage and unstructured data. In Unity Catalog, these patterns are covered by external tables and volumes respectively.

Premigration assessment

Before migrating, collect information about the existing mount points, the underlying storage location and the impacted code assets. This can be accomplished via `dbutils.fs.mounts()` or by running the UCX assessment.

Data migration

Data stored in the DBFS root location should be copied to an external location for Unity Catalog to access it. For DBFS-mounted cloud paths, the data doesn't need to be moved. Depending on the type of data, there are two upgrade scenarios:

- **Tabular data:** To read tabular data from paths to the object storage, use external tables. Follow the steps mentioned in the "Upgrading tables" section of this chapter.
- **Unstructured data:** To read unstructured data in the object storage, define external volumes over the cloud path that contains the data. Unity Catalog volumes come under a schema, just like a table, hence Unity Catalog volume location is a subfolder of the external location pointed to by its parent schema or catalog. You can reuse the same external location to create multiple volumes as subfolders. You can manage permissions for each volume separately, even though they share the same parent location.

Migration execution

During the migration, once you've defined volumes, you can access the same cloud storage location in cloud storage via mount point and Unity Catalog (external volume). If you upload a file via Unity Catalog Volumes, the file will be available via the mount point and vice versa. This allows you to migrate the mount points with zero downtime for your application.

Code adjustment and cleanup

Once your volume is in place, update your code to use the volume instead of the mount point. If there are multiple notebooks and files in your workspace, consider scanning all of them and replacing the respective paths. As you eliminate the need for each mount point, clean up permissions and storage credentials.

Best practices

Remember, mount points have workspace-level scope, while Unity Catalog volumes have metastore-level scope. It's always good practice to use parameters in your code. This is even more critical for Unity Catalog volume paths.

UPGRADING BATCH AND STREAMING WORKLOADS

To upgrade both batch and streaming workloads, make sure all of the upgrades mentioned in the previous section that are relevant to the job are already in place, including table upgrades, cluster upgrades, mount points and DBFS upgrades. The code should be adjusted to refer to the volumes created and it should read and write to Unity Catalog tables. To avoid too many changes in the code, set the default catalog at the cluster level, so that you can continue using a two-level namespace instead of the three-level namespace. If this isn't a viable option, you can also consider adding "USE <catalog-name>" at the top of the job, so that rest of the queries in the job don't change.

When upgrading a streaming workload, stop the streaming job to assure a seamless transition. For streaming workloads, the checkpoint locations in the code should be modified to refer to the corresponding external location or volume path. If the checkpoints are stored in a DBFS root location, they should be copied over to an external location/volume. Since the checkpoint location points to the same location or contains the previous checkpoints, the job will resume from the previous point upon restart.

Any use of global init scripts should also be replaced with cluster-scoped init scripts.

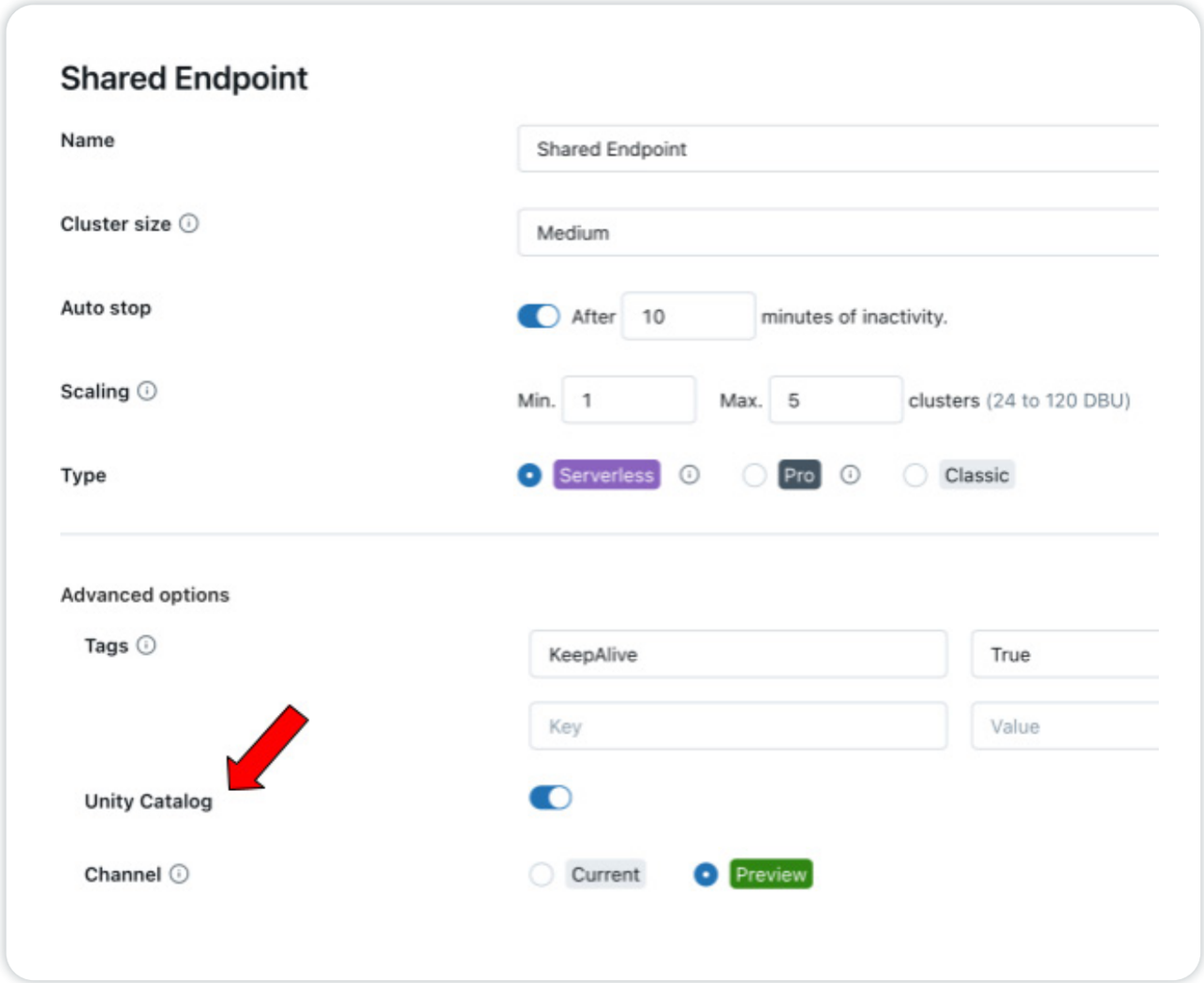
UPGRADING DLT PIPELINES

To begin the migration process, ensure that your DLT job cluster runs on Databricks Runtime 13.1 or later and follow these steps.

1. Temporarily halt the DLT job you intend to upgrade to prevent new data ingestion during the migration.
2. Create a new Unity Catalog table to store data previously handled by your DLT pipeline as mentioned in the table migration section.
3. When migrating the table, ensure that there is schema compatibility with the existing DLT table.
4. Once the new table is in place, modify the DLT job definition to use it as the data sink, replacing references to the old DLT table.
5. Restart the DLT job, which will now write data to the Unity Catalog table.

UPGRADING WAREHOUSES

Once your workspace is attached to the metastore, any new warehouse created will be enabled for Unity Catalog by default. For existing warehouses, edit the configuration, and enable the Unity Catalog toggle under "Advanced options" as shown below:



If the majority of your queries in a workspace point to the same catalog, then you can make that catalog the default catalog for the workspace, and hence for the warehouses. You can change the default catalog using the admin console as described here.

UPGRADING BI AND VISUALIZATION TOOLS

The majority of BI and visualization tools interact with Databricks by connecting to SQL warehouses, so once the warehouses are Unity Catalog–enabled, they'll be able to communicate with Unity Catalog. Connection strings used to connect to the SQL warehouses by the external tools should be modified to include the catalog name if the workspace default catalog doesn't suffice.

Upgrading machine learning models

Models that are registered in the workspace model registry can be upgraded to the Databricks-hosted version of MLflow Model Registry in Unity Catalog.

To create new registered models, you need the `CREATE_MODEL` privilege on a schema, in addition to the `USE SCHEMA` and `USE CATALOG` privileges on the schema and its enclosing catalog. `CREATE_MODEL` is a new schema-level privilege that you can grant using the Catalog Explorer UI or the `SQL GRANT` command, as shown below.

```
GRANT CREATE_MODEL ON SCHEMA <schema-name> TO <principal>
```

To upgrade existing ML workflows to Unity Catalog, you can register models under a schema in Unity Catalog. By default, the MLflow Python client creates models in the Databricks workspace model registry. To use Unity Catalog, configure the MLflow client to set the model registry as **`databricks-uc`**, which points to the Unity Catalog registry, and use the three-level namespace when registering the model.

```
import mlflow
mlflow.set_registry_uri("databricks-uc")
...
mlflow.register_model(model_uri, "catalog.schema.model_name")
```

Individual models can be upgraded to Unity Catalog by using the sample Python notebook, which uses the model registry APIs to upgrade existing models in the workspace Model Registry to Unity Catalog.

UPGRADING FEATURE TABLES

Feature tables upgrade from workspace to Unity Catalog requires the underlying table to be upgraded to Unity Catalog as a requirement. Follow instructions in the "Upgrading tables" section of this chapter to make sure that the table is available prior to migrating the feature tables. Once the underlying tables are available in Unity Catalog, use `upgrade_workspace_table` to upgrade the workspace feature table metadata to Unity Catalog, as illustrated in the following code.

```
%pip install databricks-feature-engineering --upgrade

dbutils.library.restartPython()

from databricks.feature_engineering import UpgradeClient
upgrade_client = UpgradeClient()
upgrade_client.upgrade_workspace_table(
  source_workspace_table='recommender_system.customer_features',
  target_uc_table='ml.recommender_system.customer_features'
)
```

The following metadata is upgraded to Unity Catalog:

- Primary keys
- Time series columns
- Table and column comments (descriptions)
- Table and column tags
- Notebook and job lineage

Upgrade challenges and limitations

Even though Unity Catalog is a significant deviation from the way the Hive metastore is governed, the upgrade process is designed to be streamlined, with minimal impact on end users and their day-to-day operations interacting with the data. The primary challenges of upgrading to Unity Catalog include the time investment required for the upgrade process and the need to maintain both the Hive metastore and Unity Catalog during the upgrade to minimize downtime and impact on end users and downstream data consumers.

Certain limitations exist in Unity Catalog–enabled compute that didn't exist in non–Unity Catalog compute due to the strong isolation for processes running in the Unity Catalog compute enabled by Lakeguard. User and process isolation is enforced by Lakeguard on shared and single-user clusters supporting fine-grained access controls including row-level and column-level filters.

Some of these limitations are described in the following sections.

SINGLE USER ACCESS MODE LIMITATIONS ON UNITY CATALOG

Single user access mode on Unity Catalog has the following limitations.

Fine-grained access control limitations for Unity Catalog single user access mode on DBR 15.3 and earlier

- Dynamic views aren't supported
- To read from a view, you must have SELECT on all referenced tables and views
- You can't access a table that has a row filter or column mask

Compute with single user access mode on Databricks Runtime 15.4 LTS or later (Public Preview)
Databricks Runtime 15.4 LTS and later support fine-grained access control on single-user compute. To take advantage of the data filtering provided in Databricks Runtime 15.4 LTS and later, you must also verify that your workspace is enabled for serverless compute, because the data filtering functionality that supports fine-grained access controls runs on serverless compute. You might therefore be charged for serverless compute resources when you use single-user compute to run data filtering operations. See Fine-grained access control on single-user compute.

Streaming limitations for Unity Catalog single user access mode

- Asynchronous checkpointing isn't supported in Databricks Runtime 11.3 LTS and earlier
- StreamingQueryListener requires Databricks Runtime 15.1 or later to use credentials or interact with objects managed by Unity Catalog on single user compute

SHARED ACCESS MODE LIMITATIONS ON UNITY CATALOG

Shared access mode on Unity Catalog has the following limitations.

- Databricks Runtime ML and Spark Machine Learning Library (MLlib) aren't supported
- Spark-submit jobs aren't supported
- In Databricks Runtime 13.3 and later, individual rows must not exceed 128MB
- PySpark UDFs can't access Git folders, workspace files or volumes to import modules in Databricks Runtime 14.2 and earlier
- DBFS root and mounts don't support FUSE
- When you use shared access mode with credential pass-through, Unity Catalog features are disabled
- Custom containers aren't supported

Language support for Unity Catalog shared access mode

- R isn't supported
- Scala is supported in Databricks Runtime 13.3 and later
  - In Databricks Runtime 15.4 LTS and later, all Java or Scala libraries (JAR files) bundled with Databricks Runtime are available on compute in Unity Catalog access modes
  - For Databricks Runtime 15.3 or earlier on compute that uses shared access mode, set the Spark config `spark.databricks.scala.kernel.fullClasspath.enabled` to `true`

Spark API limitations for Unity Catalog shared access mode

- RDD APIs aren't supported
- DBUtils and other clients that directly read the data from cloud storage are only supported when you use an external location to access the storage location. See Create an external location to connect cloud storage to Databricks.
- Spark Context (sc), spark.sparkContext and sqlContext aren't supported for Scala in any Databricks Runtime and aren't supported for Python in Databricks Runtime 14.0 and later
  - Databricks recommends using the spark variable to interact with the SparkSession instance
  - The following sc functions are also not supported: emptyRDD, range, init_batched_serializer, parallelize, pickleFile, textFile, wholeTextFiles, binaryFiles, binaryRecords, sequenceFile, newAPIHadoopFile, newAPIHadoopRDD, hadoopFile, hadoopRDD, union, runJob, setSystemProperty, uiWebUrl, stop, setJobGroup, setLocalProperty, getConf
  - The following Scala Dataset API operations require Databricks Runtime 15.4 LTS or later: map, mapPartitions, foreachPartition, flatMap, reduce and filter

UDF limitations for Unity Catalog shared access mode

User-defined functions (UDFs) have the following limitations with shared access mode:

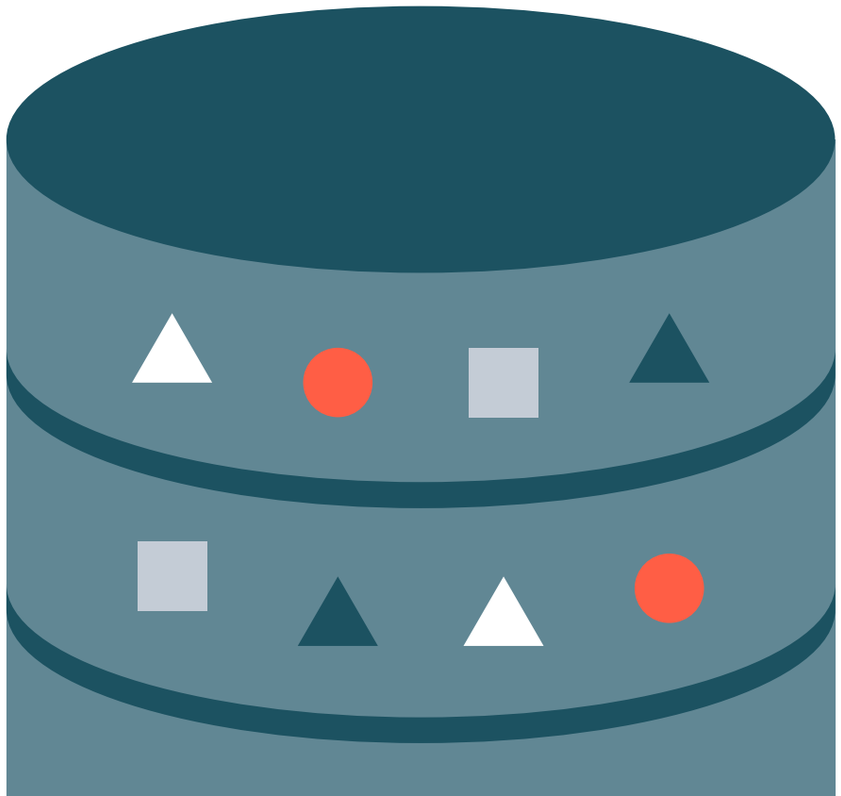
- Hive UDFs aren't supported
- applyInPandas and mapInPandas require Databricks Runtime 14.3 or later
- Scala scalar UDFs require Databricks Runtime 14.2 or later. Other Scala UDFs and UDAFs aren't supported.
- In Databricks Runtime 14.2 and earlier, using a custom version of grpc, pyarrow or protobuf in a PySpark UDF through notebook-scoped or cluster-scoped libraries isn't supported because the installed version is always preferred. To find the version of installed libraries, see the System Environment section of the specific Databricks Runtime version release notes.
- Python scalar UDFs and Pandas UDFs require Databricks Runtime 13.3 LTS or later. Other Python UDFs, including UDAFs, UDTFs and Pandas on Spark aren't supported.

For a complete list of limitations, refer to the documentation.

Rollback strategy

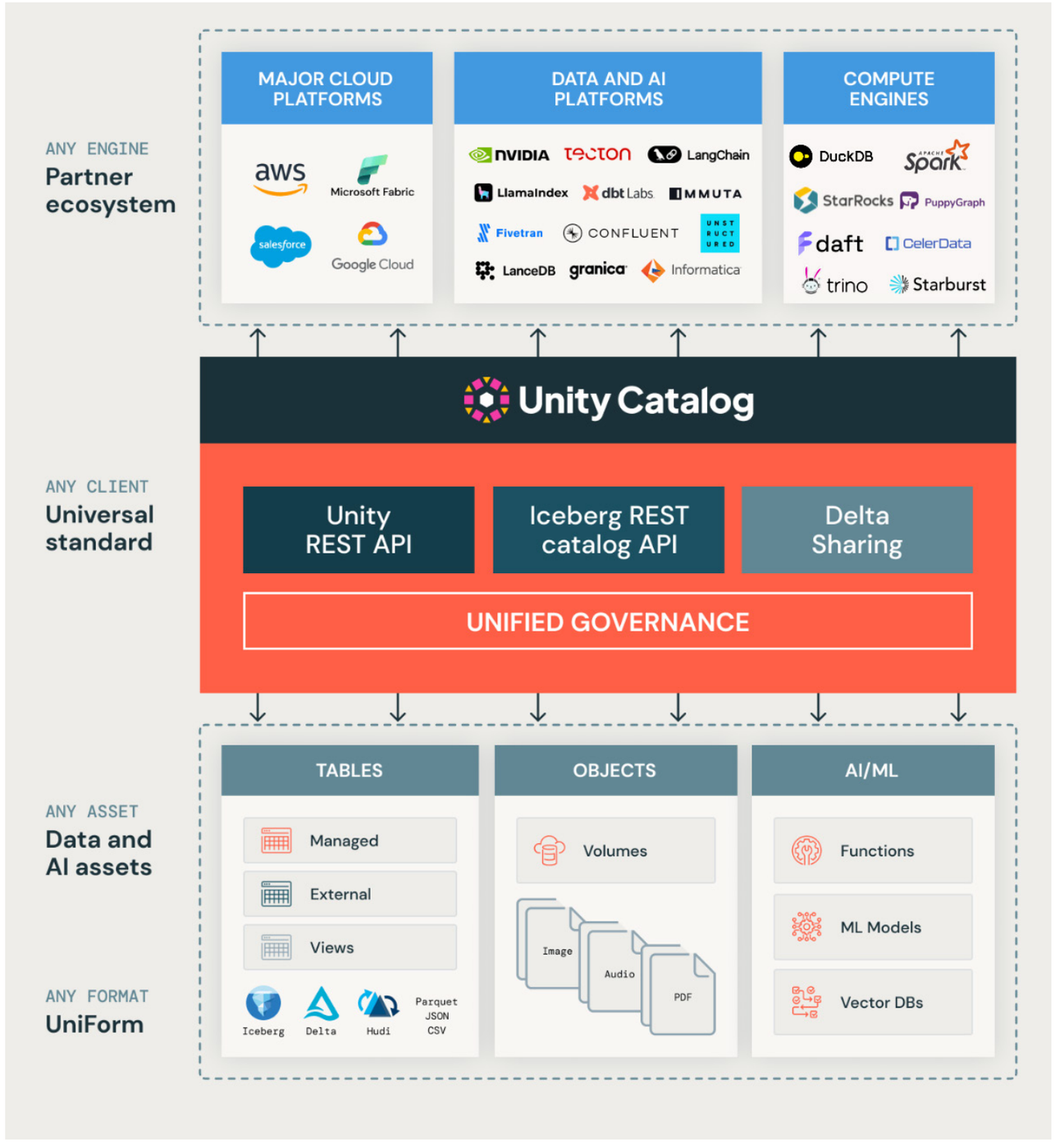
The Unity Catalog upgrade is a significant change to the Databricks Platform, requiring modifications to code, operations and infrastructure. It's important to have a rollback strategy in case an issue arises that prevents the upgrade from proceeding.

- We recommend that you continue hydrating the Hive metastore during the Unity Catalog upgrade to ensure a fallback option is available in case of any issues
- A phased approach is recommended for the Unity Catalog upgrade instead of a big bang approach. This gradual implementation can help minimize risks and disruptions.
- All the code changes should only come via a code repository and CI/CD process
- Make use of Databricks Terraform Provider to automate the Unity Catalog deployment process



Open Data Accessibility

Here at Databricks, we're excited to announce the open sourcing of Unity Catalog, the industry's first open source catalog for data and AI governance across clouds, data formats and data platforms. The project is now available on GitHub, marking the first step in our journey to bring the Unity Catalog vision into open source. Unity Catalog is hosted by LF AI & Data, a part of the Linux Foundation that fosters open source innovation in AI and data.



Why open source?

In the past two years since Unity Catalog's General Availability we've seen widespread adoption. Organizations have consistently expressed the need for an open foundation for their data and AI applications, not just for today but for the innovations of the coming decades.

Unfortunately, most data platforms today function as walled gardens. Many cloud data warehouses use "native tables" that aren't in open formats, while other platforms require customers to pay for always-on compute even when accessing data from external engines, de facto doubling the computation costs. Additionally, many platforms limit the data formats and clients they support.

This leads to siloed data and fragmented governance across assets. Without a multimodal interface for tabular data and AI assets, organizations are forced to piece together multiple disjointed solutions. Databricks has already taken a strong stance in the industry by being the only major platform where all tables are in open formats by default and by opening up Delta tables to Apache Iceberg™ clients with Delta Lake UniForm last year. By open sourcing Unity Catalog, we're providing organizations with an open foundation for their current and future workloads.

INTEROPERABLE, OPEN, UNIFIED

Among all the characteristics of Unity Catalog, the three most important ones you need to remember are: *interoperability*, *openness* and *unified governance*.

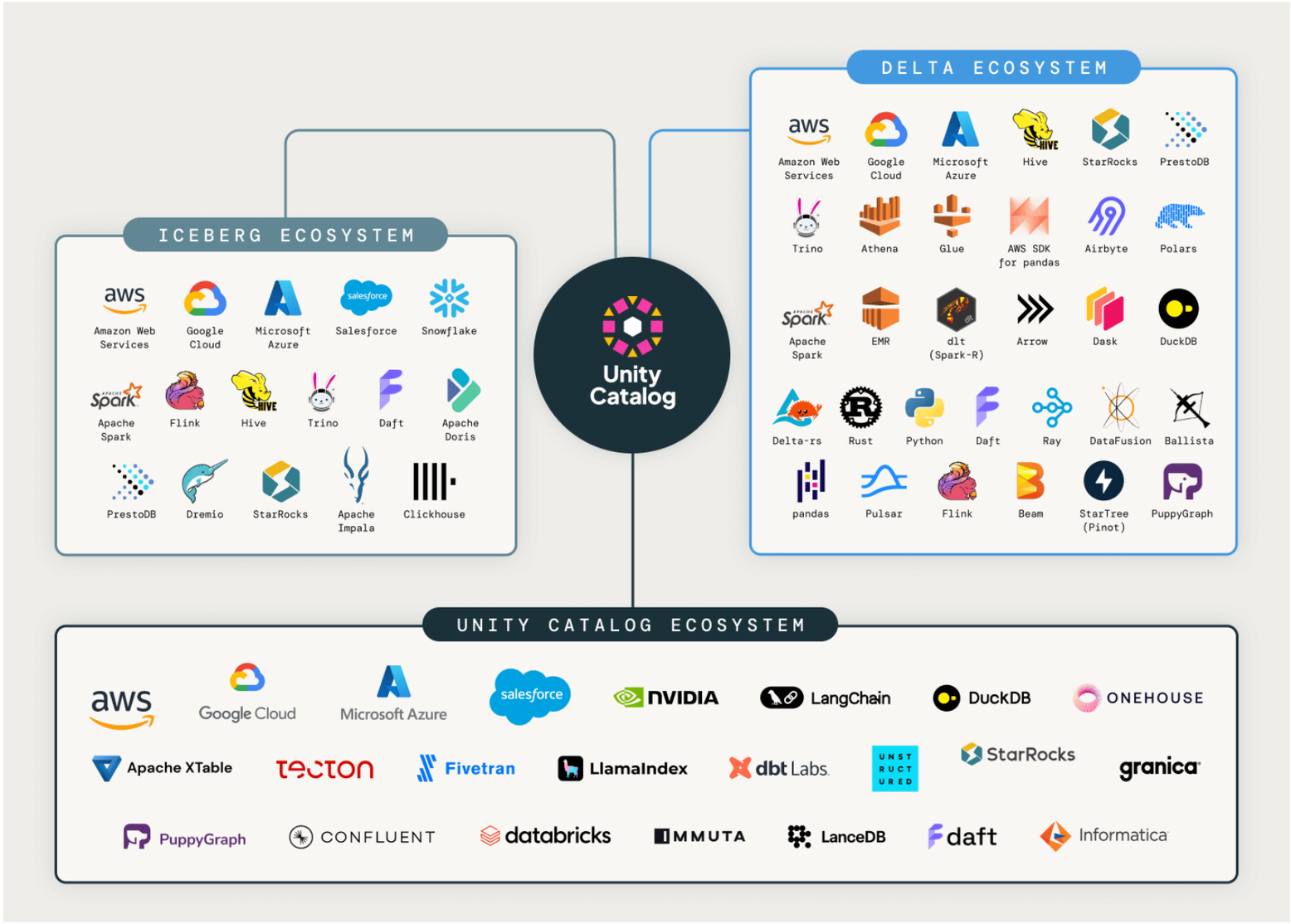
Unity Catalog is interoperable across any format and any engine. It supports Delta Lake, Apache Iceberg via UniForm, Parquet, CSV, JSON and many other formats. It also implements the Iceberg REST Catalog API to interoperate with a broad ecosystem of engines.

Unity Catalog is now also open source under the Apache 2.0 license, including an OpenAPI specification, with both server and clients. Customers need to be flexible and must be able to choose which engine to use, and open standards maximize this by ensuring extensive interoperability across engines, tools and platforms.

Unity Catalog is a universal catalog that's able to govern and secure any type of data, be it structured Delta tables, unstructured folders of images and documents or GenAI models and tools with a unified solution. It's able to secure access with strong authentication, secure credential vending and asset-level access control.

ECOSYSTEM

Unity Catalog open source is supported by a vibrant community comprising all three main cloud providers, data and AI platforms and many software vendors already. Choosing Unity Catalog today makes your company future-ready.



JOIN THE COMMUNITY TODAY

Join the Unity Catalog open source community on LinkedIn, GitHub or Slack.

Summary

The Databricks Data Intelligence Platform is a versatile, cloud-native solution that operates seamlessly across AWS, Azure and Google Cloud Platform (GCP). This platform empowers organizations with the flexibility to choose their preferred cloud provider or adopt a multicloud strategy, enabling efficient execution and integration of workloads across different environments. However, as organizations scale their use of data, effective governance becomes increasingly critical, particularly in the face of challenges like decentralized responsibilities and evolving roles such as the chief data officer (CDO).

To address these governance challenges, Databricks introduces **Unity Catalog**, a unified governance framework that centralizes the management of structured and unstructured data, AI models and data sharing across all cloud environments. Unity Catalog simplifies governance by providing a consistent experience across clouds, making it easier for organizations to implement and enforce data governance policies at scale.

Unity Catalog's cross-cloud data sharing capabilities are further enhanced by **Delta Sharing**, an open protocol for secure data sharing across platforms, clouds and regions. Delta Sharing allows organizations to share live data with external partners, customers and vendors, regardless of their infrastructure. This open source protocol supports various data formats and is designed to integrate seamlessly with Unity Catalog, enabling secure, scalable and governed data sharing.

Delta Sharing, a key feature of Unity Catalog, revolutionizes how data is shared across organizations and platforms. It provides a secure, flexible and open method for sharing live data with any recipient, regardless of their infrastructure or cloud provider. Delta Sharing is the first open source data sharing protocol designed to facilitate collaboration across different platforms and ecosystems.

In addition to its robust governance and data sharing capabilities, Unity Catalog is now available as an open source project, hosted by the Linux Foundation. By open sourcing Unity Catalog, Databricks provides organizations with an open foundation for data and AI governance that's interoperable across clouds, data formats and platforms.

The Databricks Data Intelligence Platform, with the integration of Unity Catalog, provides a powerful and unified solution for managing data governance, security and sharing across multicloud environments. Unity Catalog's ability to centralize governance, abstract cloud-specific complexities and facilitate secure data sharing through Delta Sharing makes it an indispensable tool for modern organizations. By open sourcing Unity Catalog, Databricks extends its commitment to openness and interoperability, empowering organizations to build scalable, secure and innovative data and AI solutions.

As businesses continue to navigate the complexities of data governance and multicloud strategies, Unity Catalog stands out as a future-proof solution that not only simplifies data management but also enhances security, compliance and collaboration across the enterprise.

Resources

Demos

- Watch the Unity Catalog demos

Documentation

- AWS documentation
- Azure documentation
- GCP documentation

Featured talks

- [Keynote] Evolving Data Governance With Unity Catalog Presented by Matei Zaharia at Data + AI Summit 2024
- What's New in Unity Catalog — With Live Demos
- Technical Deep Dive for Practitioners: Databricks Unity Catalog from A-Z

About the Authors

Pearl Ubaru is a Senior Technical Product Marketing Engineer at Databricks with skills in data sharing, data warehousing, data analytics and BI, and data governance. You can reach out to Pearl on LinkedIn.

Fabian Lanz is a Senior Solutions Architect at Databricks with a primary focus on AI and governance. He works with customers to develop secure and scalable end-to-end data and AI solutions. You can reach out to Fabian on LinkedIn.

Karthik Subbarao is a Specialist Solutions Architect at Databricks, focusing on platform, security and governance topics. He helps organizations design and implement secure, scalable and well-governed data platforms. You can reach Karthik on LinkedIn.

Kiran Sreekumar is a Specialist Solutions Architect at Databricks, focusing on data governance. He works with customers to implement their enterprise data governance strategies. You can reach Kiran on LinkedIn.

Mattia Zeni is a Senior Solutions Architect at Databricks, where he collaborates with strategic customers to maximize the value of their data using open data platforms. His primary areas of focus are data governance and managed serverless compute. Connect with Mattia on LinkedIn.

Jince James is a Senior Resident Solutions Architect at Databricks, focusing on data engineering, governance and data strategy. He helps organizations navigate and solve complex data and AI challenges. You can connect with Jince on LinkedIn.

Ivan Trusov is a Senior Specialist Solutions Architect on the Databricks EMEA team. His focus areas are governance, platform automation and security. He enables data teams to build efficient and secure data platforms. You can reach out to Ivan on LinkedIn.

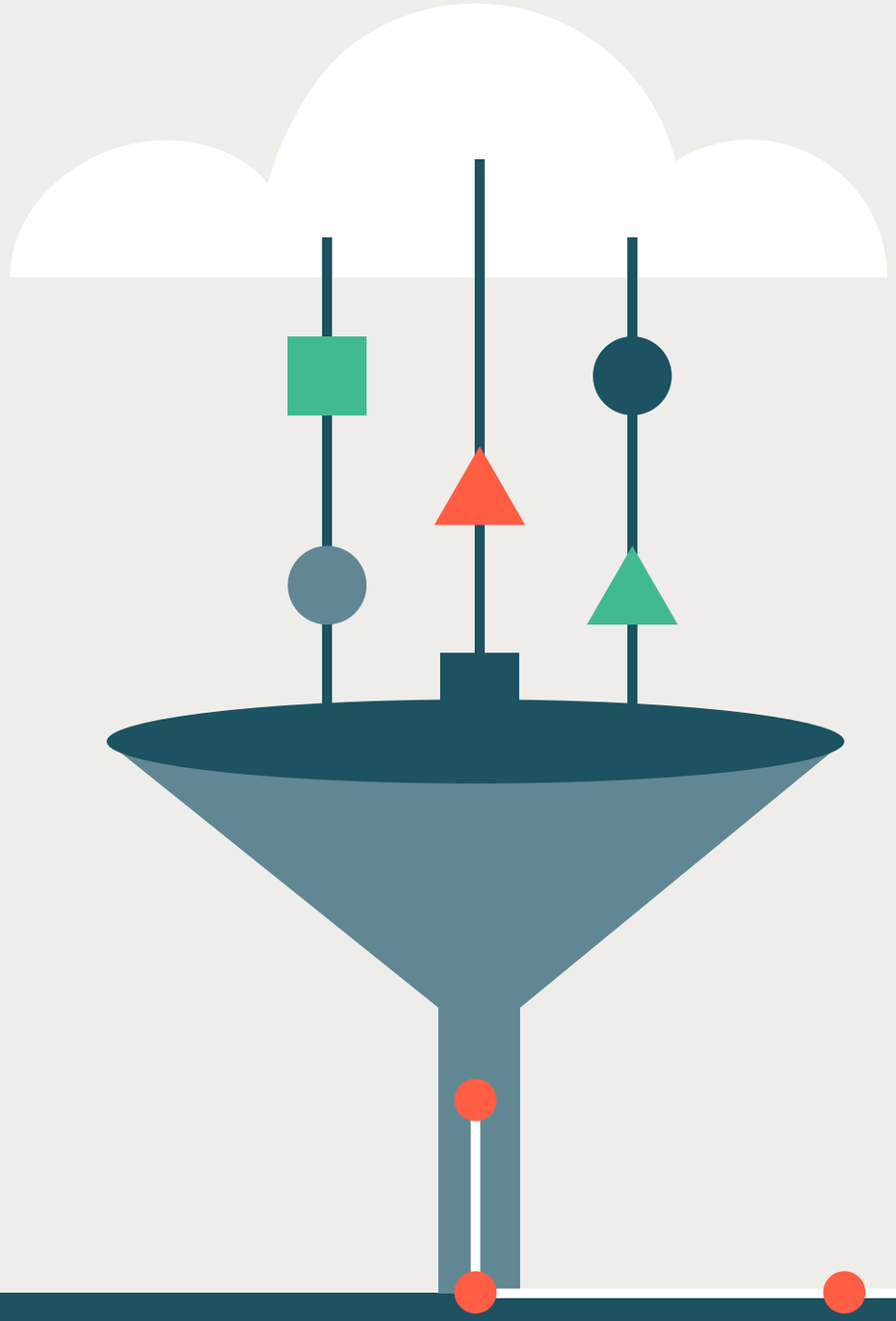
Saurabh Shukla is a Senior Specialist Solutions Architect at Databricks with deep expertise in big data engineering, real-time analytics, cloud computing, data architecture and advanced software development. Saurabh crafts scalable data solutions and leverages advanced technology to solve complex challenges. You can reach out to Saurabh on LinkedIn.

Amit Kara, Director of Technical Marketing at Databricks, is a seasoned expert in data management, combining technical expertise with strategic vision to help organizations realize the full potential of their data. Connect with Amit on LinkedIn.

About Databricks

Databricks is the data and AI company. More than 10,000 organizations worldwide — including Block, Comcast, Condé Nast, Rivian, Shell and over 60% of the Fortune 500 — rely on the Databricks Data Intelligence Platform to take control of their data and put it to work with AI. Databricks is headquartered in San Francisco, with offices around the globe, and was founded by the original creators of Lakehouse, Apache Spark™, Delta Lake and MLflow. To learn more, follow Databricks on LinkedIn, X and Facebook.

TRY DATABRICKS FREE



databricks