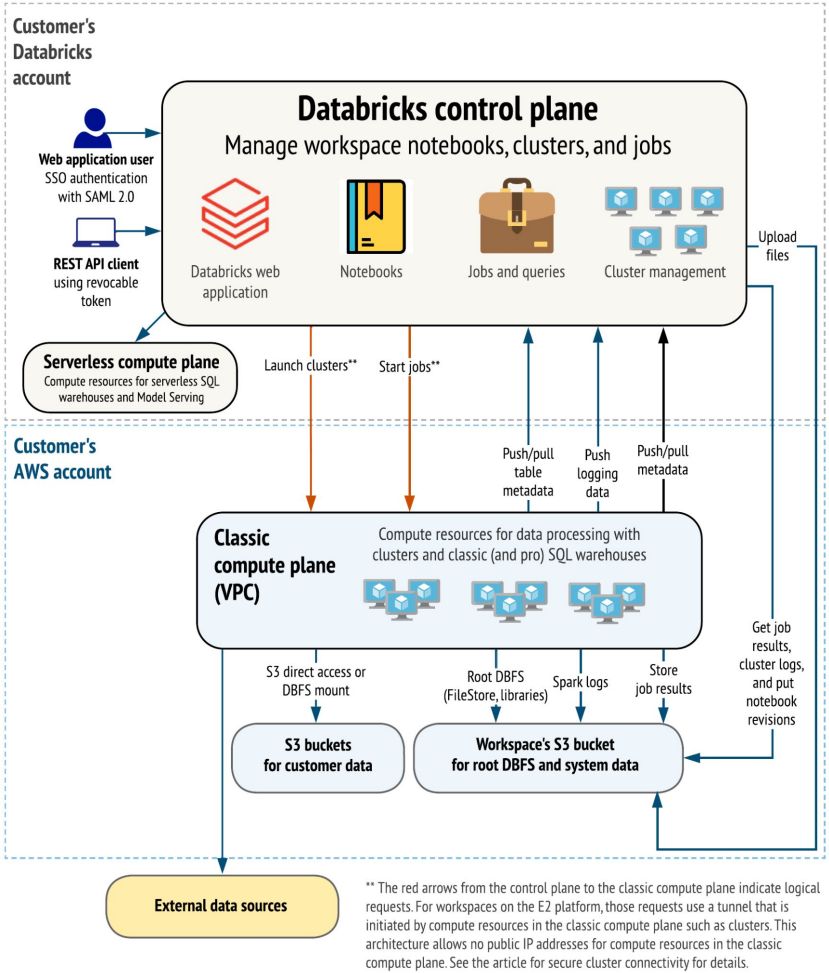
databricks

What is Databricks?

Grow Data Skills

Databricks is a **cloud-based platform** built on Apache Spark that provides a **collaborative environment** for big data processing and analytics. It offers an integrated workspace where data engineers, data scientists, and analysts can work together to leverage the power of Spark for various use cases.

Databricks is important because it makes it easier to use a Apache Spark. Instead of having to worry about all the technical stuff behind the scenes, Databricks gives you a simple and friendly way to use Spark. It takes care of all the **complicated setup and management stuff** so that you can focus on working with your data and doing cool analytics tasks. It’s like having a magic helper that takes care of the boring stuff, so you can have more fun exploring and analyzing your data.



** The red arrows from the control plane to the classic compute plane indicate logical requests. For workspaces on the E2 platform, those requests use a tunnel that is initiated by compute resources in the classic compute plane such as clusters. This architecture allows no public IP addresses for compute resources in the classic compute plane. See the article for secure cluster connectivity for details.

Databricks Components

Grow Data Skills

Databricks is composed of several main components that work together to provide a comprehensive data analytics and AI platform:

- **Apache Spark:** At its core, Databricks is built on Apache Spark, a powerful open-source, distributed computing system that provides fast data processing and analytics capabilities. Databricks enhances Spark with optimized performance and additional features.

- **Databricks Workspace:** This is a collaborative environment where data scientists, data engineers, and analysts can work together. It includes interactive notebooks (supporting languages like Python, Scala, SQL, and R), dashboards, and APIs for collaborative development and data exploration.

- **Databricks Runtime:** A performance-optimized version of Apache Spark with enhancements for reliability and performance, including optimizations for cloud environments and additional data sources.

- **Delta Lake:** An open-source storage layer that brings reliability to Data Lakes. Delta Lake provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing.

- **Workflow**: Databricks Workflows simplify job orchestration, allowing you to create, schedule, and manage data pipelines using a no-code or low-code interface. They support tasks like ETL, machine learning, and batch or streaming workflows, ensuring seamless integration with Databricks Jobs and other tools.

Databricks Components

Grow Data Skills

- **Databricks SQL:** A feature for running SQL queries on your data lakes. It provides a simple way for analysts and data scientists to query big data using SQL, visualize results, and share insights.

- **SQL Warehouses:** Databricks SQL Warehouse is a scalable, cloud-native data warehouse that supports high-performance SQL queries on your data lake. It enables analytics and BI reporting with integrated tools like Power BI and Tableau, ensuring fast, cost-efficient query execution with fully managed infrastructure.

- **Catalog:** Databricks Catalog provides centralized governance, organizing your data and metadata.

- **Data Integration Tools:** These allow for easy integration with various data sources, enabling users to import data from different storage systems.

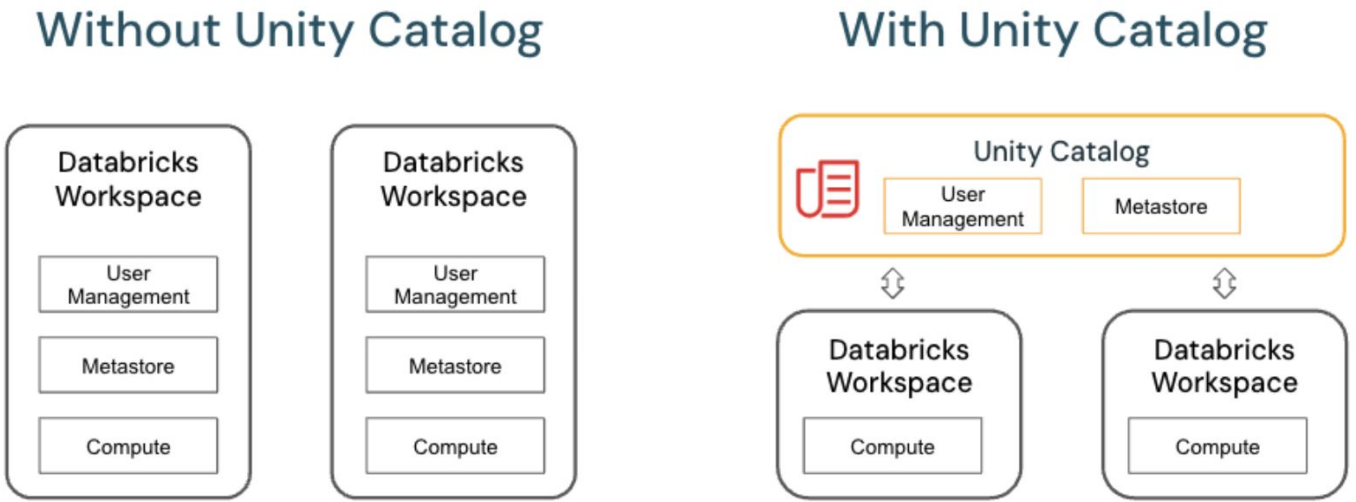
- **Cluster Management:** Databricks allows for easy management of Apache Spark clusters, automating their deployment, scaling, and maintenance. Users can quickly start clusters and adjust their size according to the workload requirements.

- **Delta Live Tables:** They simplify ETL with declarative pipelines, ensuring data quality with built-in validation and real-time monitoring. DLT leverages Delta Lake for incremental processing, making pipelines efficient and reliable.

Unity Catalog

Grow Data Skills

Unity Catalog is a unified governance solution for managing and securing your data assets in Databricks.



Key features of Unity Catalog include:

- **Define once, secure everywhere:** Databricks Catalog provides centralized governance, organizing your data and metadata.

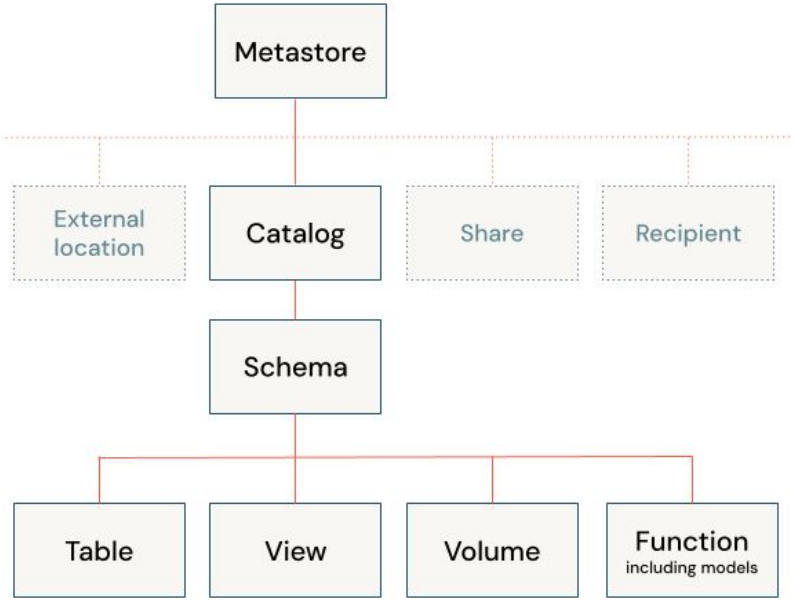
- **Standards-compliant security model:** Unity Catalog's security model is based on standard ANSI SQL and allows administrators to grant permissions in their existing data lake using familiar syntax, at the level of catalogs, schemas (also called databases), tables, and views.

Unity Catalog

Grow Data Skills

- **Built-in auditing and lineage:** Unity Catalog automatically captures user-level audit logs that record access to your data. Unity Catalog also captures lineage data that tracks how data assets are created and used across all languages.

- **System tables (Public Preview):** Unity Catalog lets you easily access and query your account's operational data, including audit logs, billable usage, and lineage.



The Unity Catalog object model

Grow Data Skills

In Unity Catalog, all metadata is registered in a metastore. The hierarchy of database objects in any Unity Catalog metastore is divided into three levels, represented as a three-level namespace ***(catalog.schema.table-etc)*** when you reference tables, views, volumes, models, and functions.

Metastores: The metastore is the top-level container for metadata in Unity Catalog. It registers metadata about data and AI assets and the permissions that govern access to them. For a workspace to use Unity Catalog, it must have a Unity Catalog metastore attached.

Object hierarchy in the metastore: In a Unity Catalog metastore, the three-level database object hierarchy consists of catalogs that contain schemas, which in turn contain data and AI objects, like tables and models.

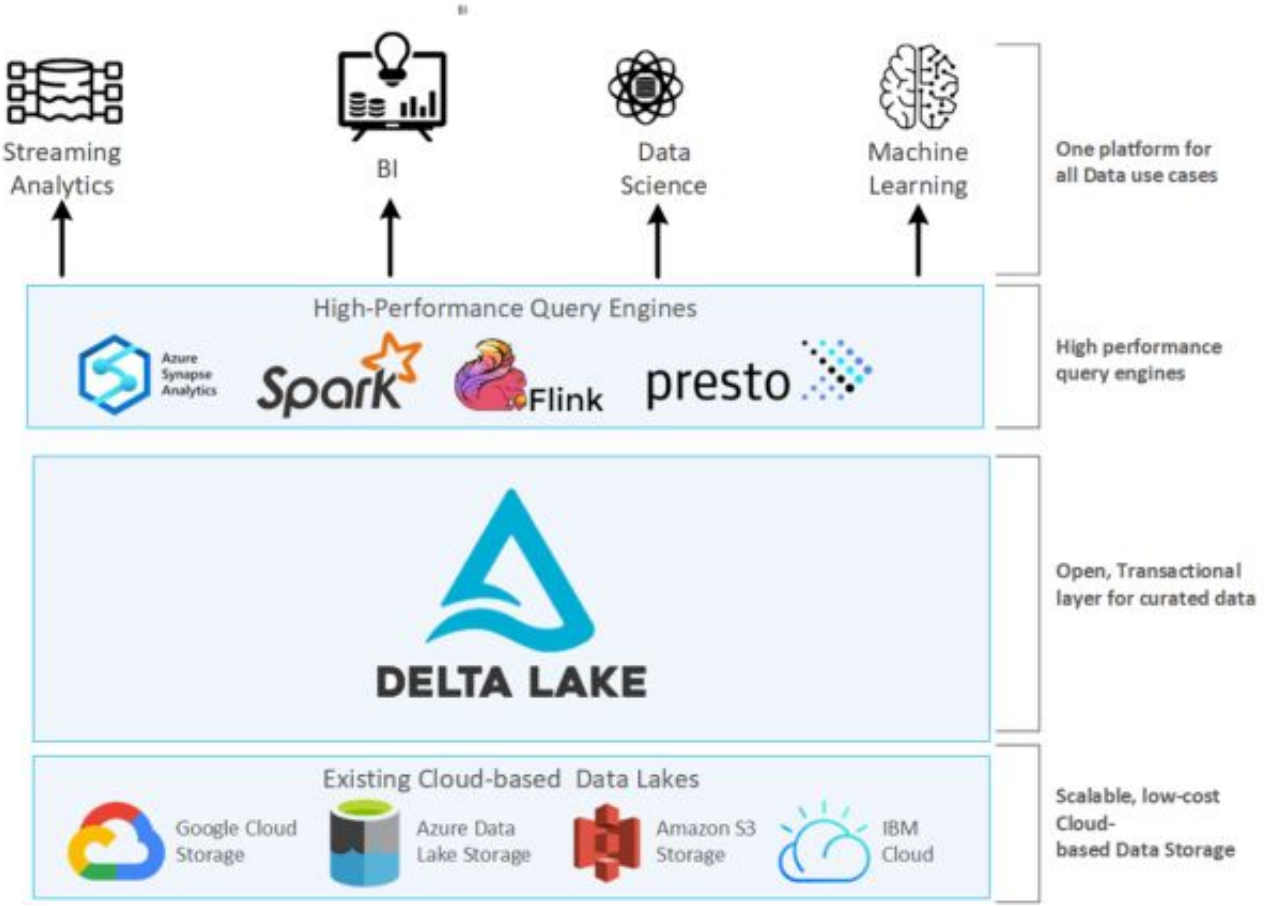
- **Level one:**
    - **Catalogs** are used to organize your data assets and are typically used as the top level in your data isolation scheme. Catalogs often mirror organizational units or software development lifecycle scopes.
    - **Non-data securable objects**, such as storage credentials and external locations, are used to manage your data governance model in Unity Catalog. These also live directly under the metastore. They are described in more detail in Other securable objects.

The Unity Catalog object model

Grow Data Skills

- **Level two:**
  - **Schemas** (also known as databases) contain tables, views, volumes, AI models, and functions. Schemas organize data and AI assets into logical categories that are more granular than catalogs. Typically a schema represents a single use case, project, or team sandbox.
- **Level three:**
  - **Volumes** is essentially a mapping between a cloud storage directory and Databricks' Unity Catalog. It allows you to securely manage and access data stored in cloud object storage directly within Databricks.
  - **Tables** are collections of data organized by rows and columns.
  - **Views** are saved queries against one or more tables.
  - **Functions** are units of saved logic that return a scalar value or set of rows.
  - **Models** are AI models packaged with MLflow and registered in Unity Catalog as functions.

Delta Lake


Streaming Analytics
BI
Data Science
Machine Learning
One platform for all Data use cases
High-Performance Query Engines
Azure Synapse Analytics
Spark
Flink
presto
High performance query engines
DELTA LAKE
Open, Transactional layer for curated data
Existing Cloud-based Data Lakes
Google Cloud Storage
Azure Data Lake Storage
Amazon S3 Storage
IBM Cloud
Scalable, low-cost Cloud-based Data Storage

Delta Lake

Grow Data Skills

Delta Lake is an open-source **storage layer** that brings **ACID** (Atomicity, Consistency, Isolation, Durability) transactions, scalable metadata handling, and other robust features to data lakes. It integrates seamlessly with Databricks and enhances the capabilities of the underlying cloud storage systems, such as Amazon S3, Azure Data Lake Storage, or Google Cloud Storage.

Here's how Delta Lake gets integrated with Databricks:

- **Storage Layer Enhancement:** Delta Lake sits on top of the existing cloud storage (like S3, ADLS, GCS) used by Databricks. It enhances these storage systems by adding a transactional layer that allows for more reliable and performant data operations.

- **ACID Transactions:** Delta Lake provides ACID transactions on Spark, ensuring data integrity and consistency. This is particularly important for handling large-scale data and complex data processing tasks that involve multiple steps and require consistent states.

- **Schema Enforcement and Evolution:** Delta Lake enforces schema on write, ensuring data quality and consistency. It also allows for schema evolution, making it easier to adapt to changing data requirements over time.

- **Time Travel (Data Versioning):** Delta Lake maintains versions of data, allowing users to access and revert to earlier versions of the data for audits, rollbacks, or reproducing experiments.

Delta Lake

Grow Data Skills

- **Unified Batch and Streaming Data Processing:** With Delta Lake, you can treat streaming data and batch data as the same data type, allowing for a simplified and unified data processing approach.

- **Optimized Layout and Indexing:** Delta Lake optimizes data layout automatically (like data compaction and indexing), which can significantly improve query performance.

- **Integration with Databricks Notebooks and Jobs:** Delta Lake tables can be easily read and written from Databricks notebooks and jobs, just like any other data source, but with the added benefits of Delta Lake's features.

- **Compatibility:** Delta Lake is fully compatible with Apache Spark APIs, making it easy to adopt and use within the Databricks environment without significant changes to existing Spark jobs.

- **Ease of Use:** Users can interact with Delta Lake using standard Spark SQL queries or Spark's DataFrame APIs, making it accessible for users with different skill sets.

- **Open Format:** Delta Lake stores data in Parquet format, an open-source, columnar storage format, which makes it accessible for processing with various data processing tools.