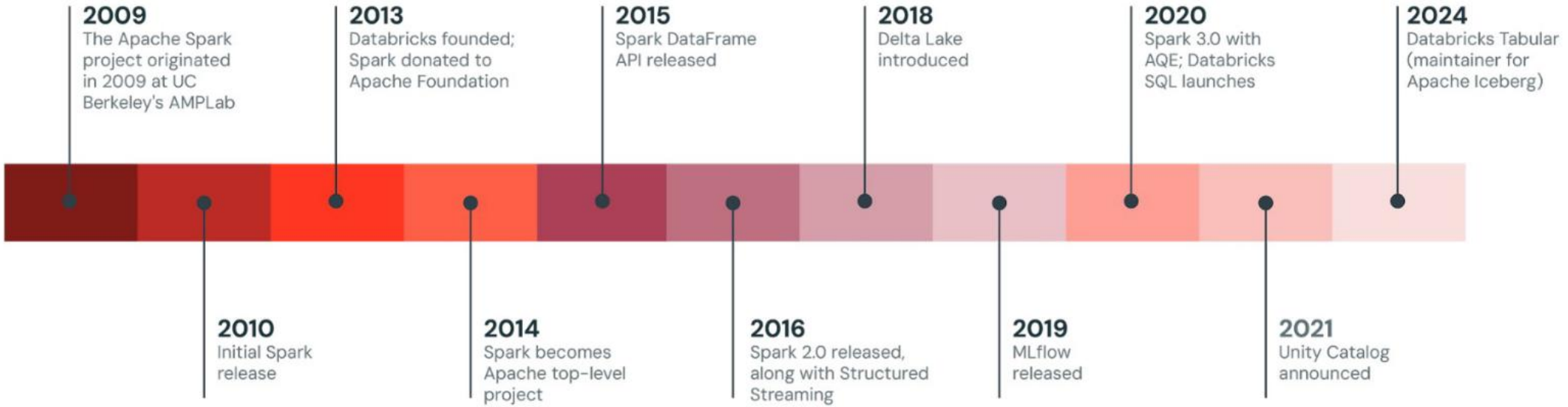
307401
Big Data and Data Warehouses

Introduction to Databricks

Introduction to Databricks

- Databricks is a cloud-based data and AI platform designed to unify data engineering, data science, and machine learning workflows.
- It was founded in 2013 by the original creators of Apache Spark and has since become one of the leading platforms for large-scale data processing and analytics.
- Built on top of Apache Spark, Databricks simplifies the process of working with big data by providing a collaborative workspace that integrates with popular cloud providers such as AWS, Azure, and Google Cloud.
- Its goal is to remove the operational complexity of big data and AI systems so that organizations can focus on generating insights and building innovative applications.
- The platform supports data ingestion, transformation, and advanced analytics through features like Delta Lake for reliable data lakes, MLflow for machine learning lifecycle management, and the Unity Catalog for data governance. Databricks provides a serverless and scalable environment where teams can run notebooks, SQL queries, streaming jobs, and machine learning models seamlessly.
- It is widely used by enterprises for building modern data architectures, enabling real-time analytics, and accelerating AI development.

Spark Timeline



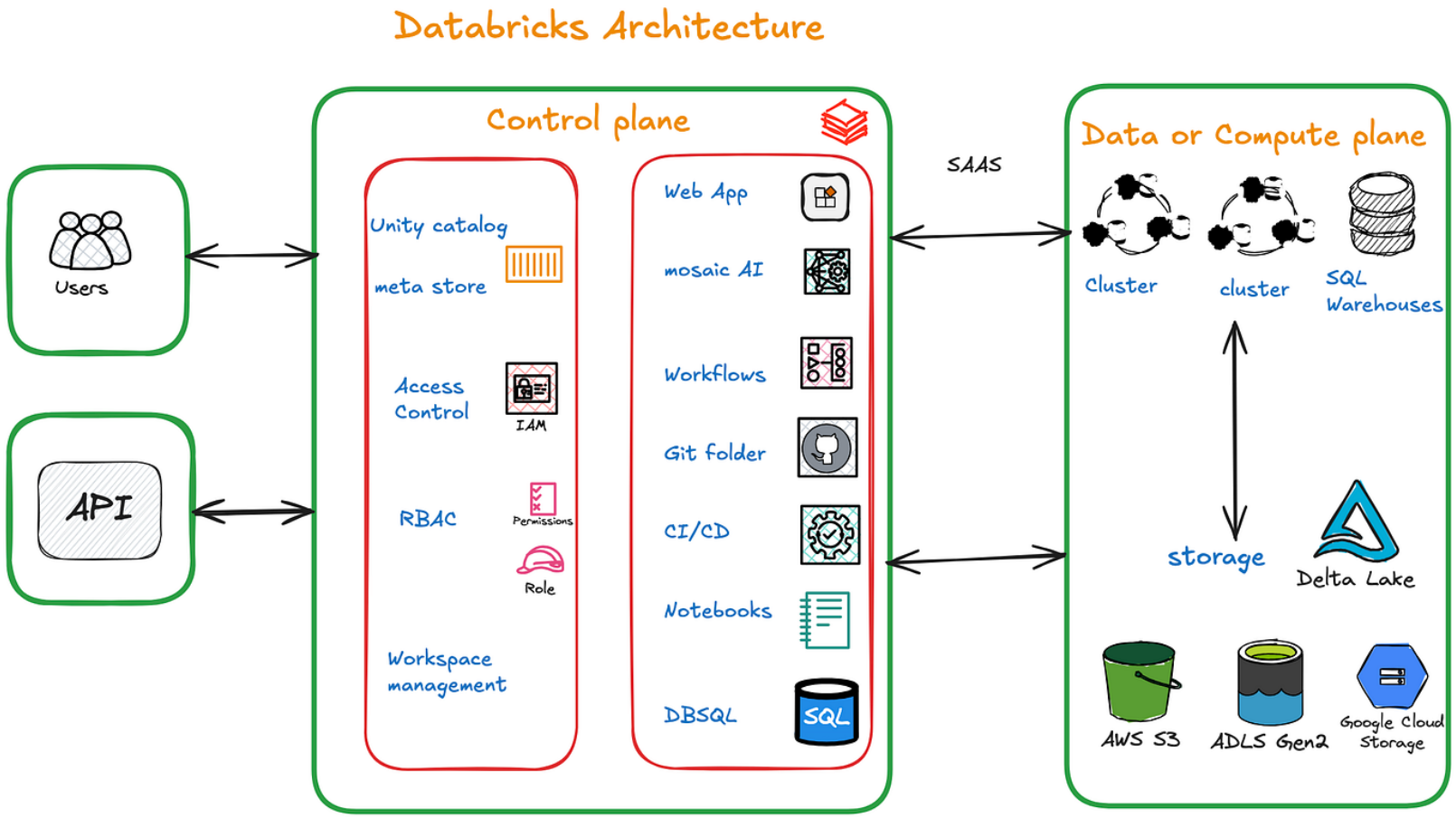
Databricks Architecture

Databricks architecture is designed to run at cloud scale and is structured around three main layers, Storage, Compute, and Management.

1. **Storage Layer**: Databricks does not store data itself but integrates with cloud-native storage services like AWS S3, Azure Data Lake Storage (ADLS), and GCP GCS. Databricks File System (DBFS) is a virtual abstraction on top of these storages for seamless interaction. Delta Lake adds ACID transactions and schema enforcement, making the storage layer reliable.
2. **Compute Layer** – This layer consists of clusters that run Spark jobs. A cluster has a Driver node (coordinates execution) and Worker nodes (perform distributed tasks). Clusters can be on-demand, autoscaled, and optimized with Databricks Runtime. The Photon execution engine improves SQL performance.
3. **Management Layer** – This is where Databricks shines. It provides a web-based workspace for collaboration, security features like role-based access control, integration with Identity providers (AD, Okta), and tools like Jobs, MLflow, and Unity Catalog. It also provides notebooks for collaborative development and Jobs for production scheduling.

Overall, the architecture separates data storage from compute, ensuring scalability and flexibility. Users interact through the workspace, submit jobs to clusters, and store results in the storage layer. This modular design makes Databricks powerful for both batch and streaming analytics, machine learning pipelines, and ad-hoc exploration.

Databricks Architecture: Control Plane & Data Plane



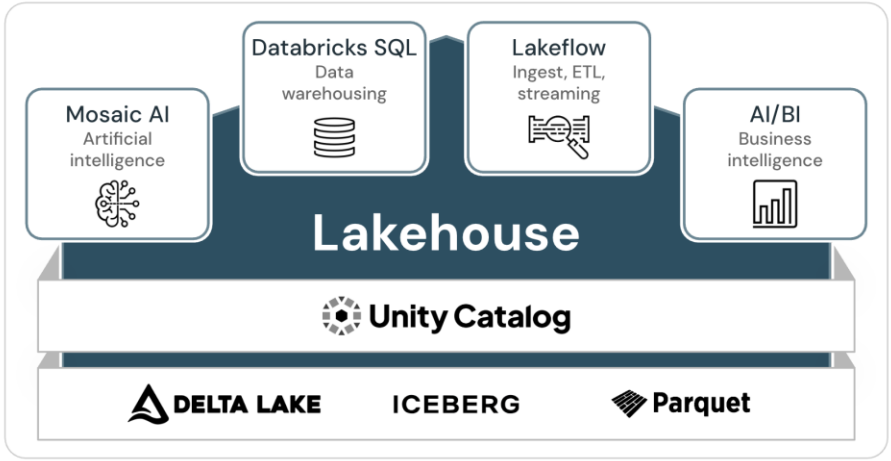
Databricks Architecture: Control Plane & Data Plane

- Databricks is built on a two-plane architecture that separates platform management from data processing to ensure security, scalability, and governance.
- The **control plane** is responsible for orchestration and decision-making:
- it hosts the Databricks workspace, manages user authentication and authorization, schedules jobs, coordinates notebook execution, maintains metadata, and enforces governance through services such as Unity Catalog and MLflow.
- The control plane does not directly process or store customer data; instead, it sends control signals and execution instructions.
- The **data plane** runs inside the customer's own cloud account and is where actual computation and data access occur.
- It consists of Spark clusters with driver and worker nodes, running Databricks Runtime and execution engines such as Photon.
- The data plane reads from and writes to cloud-native storage systems like S3, ADLS, or GCS, where Delta Lake tables provide transactional guarantees and reliability.
- Communication between the control plane and data plane is limited to secure metadata and coordination messages, ensuring that customer data remains fully isolated and under customer control.

Lakehouse Architecture

Why the Lakehouse Exists

- Traditional data platforms forced organizations to choose between data lakes and data warehouses.
- Data lakes are inexpensive and scalable but lack reliability, governance, and strong performance guarantees.
- Data warehouses provide structured analytics and governance but are costly, rigid, and difficult to scale for machine learning.
- The Lakehouse architecture was introduced to unify these two worlds into a single, coherent data platform.



Lakehouse Architecture

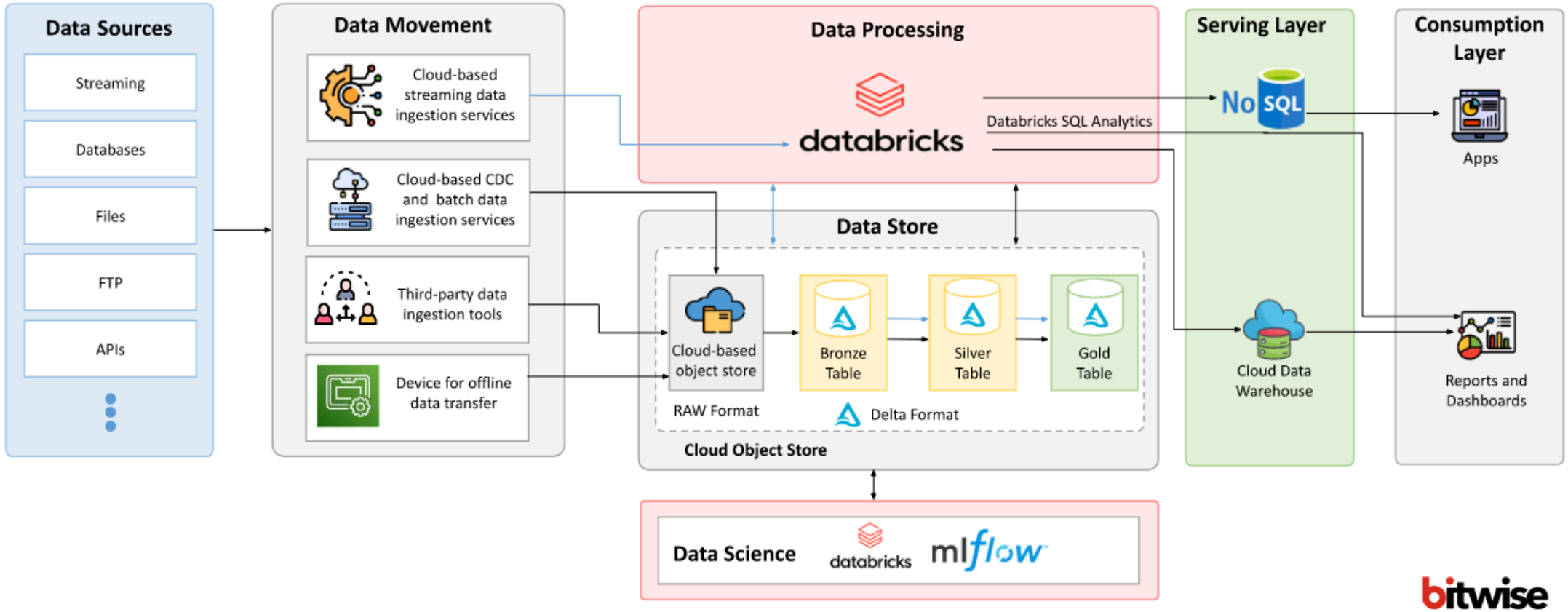
Core Lakehouse Principles

- The Lakehouse is built on open data formats, such as Parquet and Delta, ensuring long-term interoperability and avoiding vendor lock-in.
- Storage and compute are separated so that data can be stored cheaply while compute resources scale independently.
- A single copy of data supports multiple workloads, including analytics, streaming, and machine learning, eliminating redundant pipelines.

Key Architectural Components

- Cloud object storage acts as the system of record where all data is persisted.
- A transactional layer adds reliability and consistency on top of this storage.
- Multiple compute engines operate on the same data, optimized for different workloads such as SQL analytics or machine learning. Governance and catalog services provide visibility, security, and control.

Lakehouse Architecture

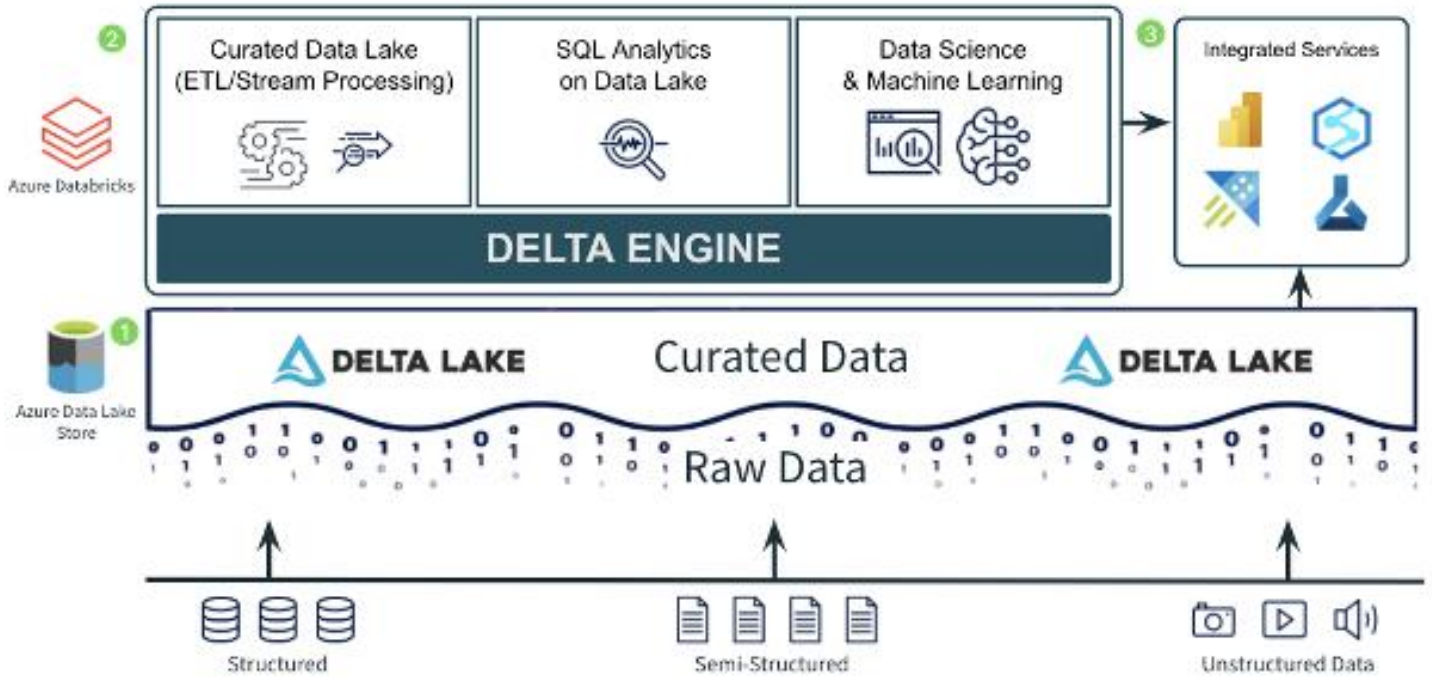


bitwise

Delta Lake Internals

Why Delta Exists

Object storage was never designed for concurrent writes or transactions. Delta Lake adds a transactional layer without replacing the storage system, enabling data lakes to behave like databases.



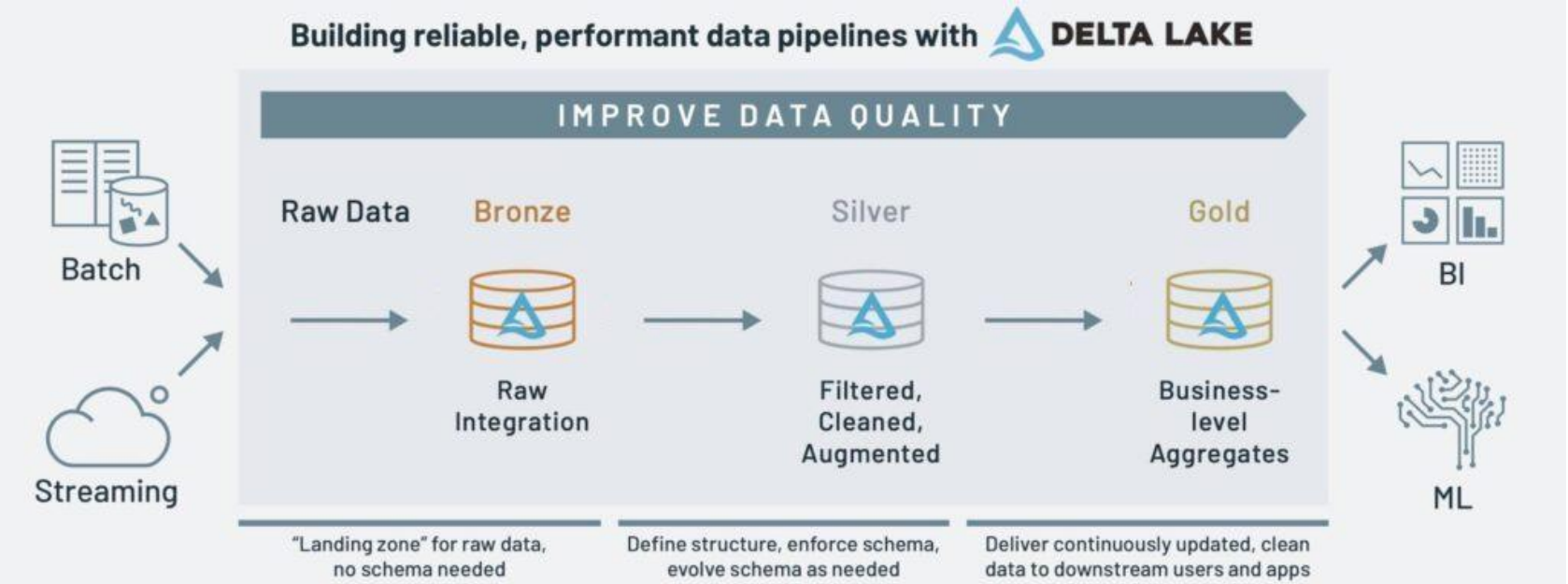
Delta Lake Internals

- **Transaction Log**
  The _delta_log is the heart of Delta Lake. Every change—insert, update, delete—is recorded. Readers reconstruct table state by replaying the log, enabling snapshot isolation and time travel.
- **ACID Guarantees**
  Explain that ACID is achieved without locking the entire dataset. Optimistic concurrency control allows multiple writers while preventing corruption.
- **Performance Mechanics**
  Delta stores statistics that allow engines to skip files. OPTIMIZE compacts small files, while Z-ORDER improves data locality. VACUUM removes obsolete data but must be used carefully due to time travel implications.

The Medalion Architecture

Concept and Purpose

- The Medallion Architecture is a structured data design pattern that organizes data into layers representing increasing levels of quality and refinement.
- It introduces discipline into data lakes by making data transformations explicit and incremental. Each layer serves a clear purpose and supports both scalability and governance.



The Medalion Architecture

Bronze Layer – Raw Data

- The Bronze layer contains raw data ingested from source systems with minimal transformation.
- Data is typically appended rather than updated, preserving the original structure and content.
- This layer is designed for traceability, auditability, and the ability to reprocess data if downstream logic changes.

Silver Layer – Cleaned Data

- The Silver layer refines Bronze data by applying cleansing, standardization, and business rules.
- Duplicates are removed, schemas are enforced, and data from multiple sources may be joined.
- This layer represents trustworthy, reusable data suitable for multiple downstream consumers.

Gold Layer – Business-Ready Data

- The Gold layer contains curated, high-quality datasets optimized for analytics and reporting.
- Data is often aggregated, denormalized, and aligned with business concepts such as KPIs or metrics.
- Performance and usability take precedence over flexibility at this stage.

Delta Lake Internals

What Delta Lake Adds

- Delta Lake is a storage layer that brings database-like reliability to data lakes. It sits on top of cloud object storage and extends Parquet files with transaction management.
- This enables data lakes to support concurrent reads and writes without corruption or inconsistency.

Transaction Log (_delta_log)

- All changes to a Delta table are recorded in a transaction log stored alongside the data.
- This log maintains a complete history of table operations, allowing the system to reconstruct any version of the table.
- The transaction log is the foundation for consistency, time travel, and concurrency control.

ACID Guarantees

- Delta Lake provides atomicity, consistency, isolation, and durability using optimistic concurrency control.
- Multiple writers can safely operate on the same table, and readers always see a consistent snapshot.
- Failed writes do not leave partial or corrupted data behind.

Performance Capabilities

- Delta Lake collects metadata and statistics that enable efficient query execution.
- File compaction reduces the small file problem, while data ordering improves locality for analytical queries. Old data versions can be safely removed to manage storage costs.

Performance Optimization & Cost Control

Compute Optimization

- Databricks workloads run on clusters that must be sized appropriately. Over-sized clusters waste money, while under-sized clusters cause slow and unstable jobs.
- Autoscaling allows compute to grow and shrink based on demand, and separating interactive and job clusters improves cost efficiency.

Data Layout and Storage

- Poor data layout is a common cause of slow queries. Excessive small files increase overhead, while improper partitioning prevents effective pruning.
- Periodic optimization reorganizes data to improve read performance and reduce compute usage.

Query Execution Behavior

- Spark optimizes queries through techniques such as predicate pushdown and adaptive execution.
- Understanding how queries are planned and executed helps diagnose performance issues.
- Caching can improve repeated access but must be used selectively to avoid memory pressure.

Cost Governance

- Technical optimization must be combined with financial controls.
- Usage tracking, cost attribution, and scheduling help organizations understand where resources are consumed. Governance practices enable cost transparency and accountability.

Governance, Security & Unity Catalog

Governance Challenges

- As data platforms scale, uncontrolled data access and duplication become major risks. Governance ensures that data is discoverable, secure, and compliant with regulations. Without centralized governance, organizations struggle with trust and accountability.

Unity Catalog Overview

- Unity Catalog provides a centralized metadata and governance layer across workspaces. It standardizes how data assets are defined, discovered, and accessed. This enables consistent security policies and simplifies cross-team collaboration.

Security and Compliance

- Fine-grained permissions allow access control at table, column, or row level. Data lineage tracks how data moves and transforms across pipelines. Audit logs provide visibility into data access and support compliance requirements.

Machine Learning with Databricks

End-to-End ML Workflow

- Machine learning workflows include data preparation, feature engineering, training, evaluation, and deployment. Databricks integrates these steps into a single platform, reducing friction between experimentation and production.

Scaling Machine Learning

- Large datasets require distributed processing for feature engineering and training. Databricks leverages Spark to scale data preparation and integrates with popular ML libraries. This allows models to be trained on full datasets rather than samples.

Feature Engineering

- Features must be consistent between training and inference to avoid model degradation. Centralizing feature logic improves reuse and reduces errors. Both batch and streaming features can be supported within the same architecture.

MLOps & Model Lifecycle Management

- **Experiment Tracking**
- Experimentation produces many model versions with different parameters and results. Tracking experiments enables reproducibility and collaboration. Metrics, artifacts, and configurations must be recorded systematically.
- **Model Registry**
- A model registry acts as a centralized system for managing trained models. It supports versioning, documentation, and controlled promotion through lifecycle stages. This introduces governance and accountability into ML deployment.
- **Production Models**
- Deployed models must be monitored for performance and drift. Data and business conditions change over time, affecting model accuracy. Effective MLOps includes retraining strategies and rollback mechanisms.

SQL Analytics & BI Integration

- **Role of SQL in Analytics**
- SQL remains the primary interface for analytics and reporting. Databricks enables SQL users to query Lakehouse data directly without duplication. This reduces latency and simplifies data architectures.
- **BI Workloads**
- Business intelligence workloads prioritize concurrency, predictable performance, and low latency. SQL warehouses are optimized to support dashboards and ad hoc queries. Workload isolation prevents interference from data engineering or ML jobs.
- **Analytics Consumption**
- Analytical datasets must be understandable and reliable. Semantic consistency and performance optimization are essential for adoption. The Lakehouse enables governed data sharing across teams.

Deployment, CI/CD & Productionization

Environment Management

- Separating development, testing, and production environments reduces risk. Configuration and data isolation prevent accidental impact on production systems. Environment consistency is critical for reliability.

CI/CD for Data Systems

- Data pipelines and ML workflows should follow software engineering practices. Source control, automated testing, and repeatable deployments improve quality and speed. Manual changes do not scale in complex systems.

Operating in Production

- Production systems require monitoring, alerting, and clear ownership. Failures are inevitable, but impact can be minimized through observability and automation. Continuous improvement is an ongoing operational responsibility.

Databricks Platform Components

Databricks.com

Databricks Platform

- **Workspace** – The Workspace is the central collaborative environment in Databricks where users organize notebooks, files, libraries, and folders. It supports team-based development by enabling shared access, versioning, and structured project organization across data engineering, analytics, and machine learning workflows.
- **Recents** – Recents provides quick access to notebooks, queries, dashboards, and other assets that were recently opened or modified. It improves productivity by reducing navigation time and helping users resume work efficiently.
- **Catalog** – Catalog (Unity Catalog) is the centralized governance layer for managing data and AI assets, including tables, views, volumes, models, and functions. It enforces fine-grained access control, lineage tracking, and auditing across workspaces.
- **Jobs & Pipelines** – Jobs & Pipelines is used to orchestrate production workloads such as scheduled jobs, batch processing, and continuous data pipelines. It enables reliable execution, monitoring, retries, and dependency management for end-to-end workflows.
- **Compute** – Compute manages the clusters and SQL warehouses that execute Databricks workloads. It allows users to configure performance, scalability, cost controls, and security for interactive and automated workloads.

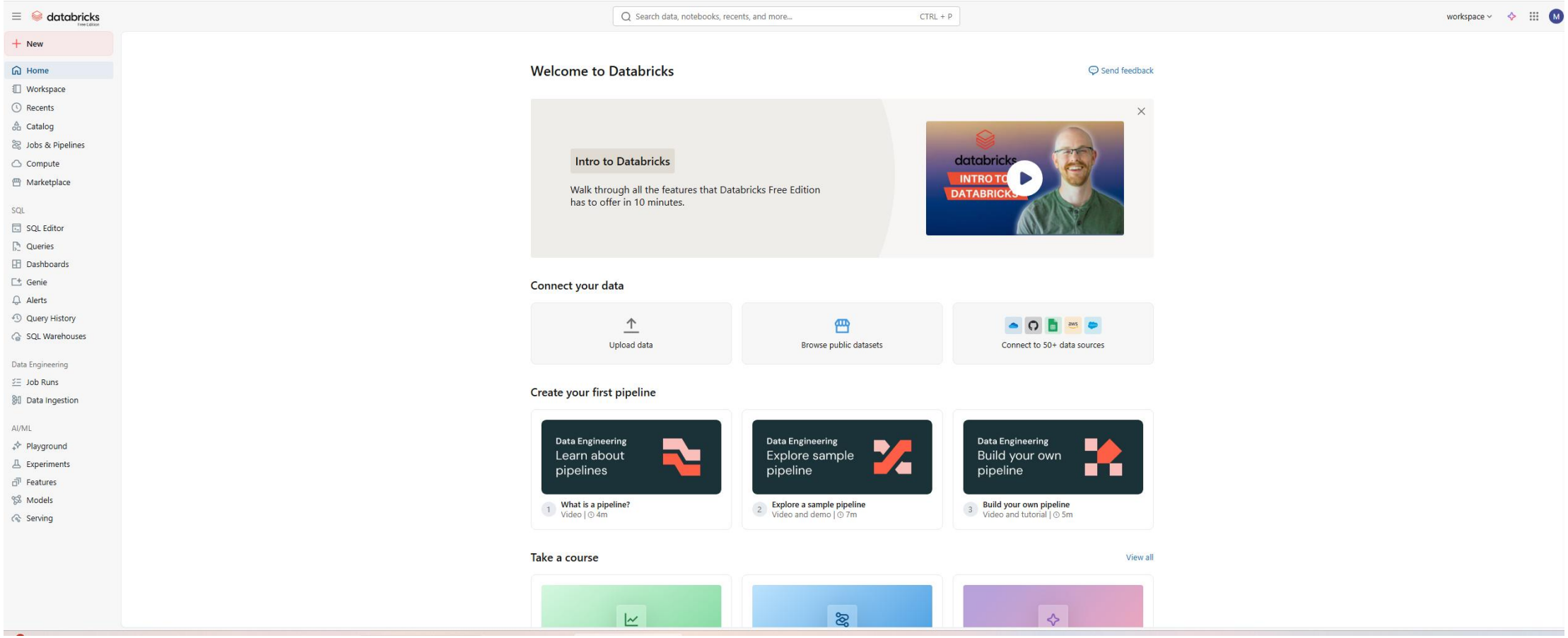
Databricks Platform

- **Marketplace** – Marketplace provides access to third-party and partner data products, machine learning models, and solution accelerators. It enables organizations to discover, evaluate, and integrate external assets securely.
- **SQL Editor** – SQL Editor is an interactive environment for writing and executing SQL queries against data stored in Databricks. It is optimized for analytics, exploration, and collaboration with built-in visualization support.
- **Queries** – Queries stores and organizes saved SQL queries for reuse and sharing. It enables teams to standardize analytics logic and maintain consistency across reports and dashboards.
- **Dashboards** – Dashboards allow users to create interactive, shareable visualizations based on SQL queries. They are designed for business intelligence use cases and can be shared with stakeholders securely.
- **Genie** – Genie is an AI-powered assistant that enables natural-language interaction with data. It helps users ask questions, generate queries, and explore insights without requiring deep SQL expertise.
- **Alerts** – Alerts monitor query results and data conditions, automatically notifying users when thresholds or rules are met. They support proactive data quality checks and operational monitoring.

Databricks Platform

- **Query History** – Query History provides a searchable log of executed SQL queries, including performance metrics and execution details. It is useful for debugging, optimization, auditing, and cost analysis.
- **SQL Warehouses** – SQL Warehouses are optimized compute resources dedicated to SQL analytics and BI workloads. They provide fast, scalable, and cost-efficient query execution with concurrency controls.
- **Job Runs** – Job Runs displays execution history and status for scheduled and triggered jobs. It provides detailed logs, metrics, and error information for troubleshooting and operational oversight.
- **Data Ingestion** – Data Ingestion tools support loading data from streaming and batch sources into Databricks. They enable scalable, reliable ingestion with built-in support for incremental processing and schema evolution.
- **Playground** – Playground is an experimental environment for testing and interacting with AI capabilities, including generative AI and large language models. It allows rapid prototyping without impacting production assets.
- **Experiments** – Experiments (MLflow Experiments) track machine learning runs, parameters, metrics, and artifacts. They enable reproducibility, comparison, and collaboration across ML development cycles.
- **Features** – Features (Feature Store) manages curated, reusable machine learning features. It ensures consistency between training and inference while supporting governance and lineage.
- **Models** – Models (MLflow Model Registry) provides centralized management of machine learning models, including versioning, staging, approvals, and lifecycle tracking. It supports collaboration and controlled deployment.
- **Serving** – Serving enables real-time and batch deployment of machine learning and foundation models as scalable endpoints. It provides low-latency inference, monitoring, and integration with applications.

Databricks Free Account



Databricks Free Edition

Databricks Free Edition (replacing Community Edition) — good for students, hobbyists, learning / prototyping.)

Databricks Free Trial — gives you temporary credits and access to more features. Valid for 14 days.

Steps to create a free Databricks account / workspace

Here are the general steps. Depending on your region or cloud provider, some details might differ.

1. Go to the Databricks website and find **Databricks Free Edition** or “Try Databricks for Free”.

2. Click **Sign up** for Free Edition, or start the Free Trial.

3. Choose your signup method:

You can often sign up just with an email (“Express signup”).

Or use your existing cloud provider account (e.g. AWS) if you want to link storage / compute resources.

4. Fill in registration details (name, email, password, etc.). If needed, verify your email.

5. (If Free Trial) You might need to enter payment method or link a cloud account so that after the trial expires, there is a way to transition if you decide to continue.

6. A workspace is created for you. In Free Edition, it’s serverless and quota-limited.

7. Once you’re inside the workspace, you can:

- Create notebooks
- Run code
- Explore tools like SQL editor, dashboards
- Work with data stored or uploaded into the workspace.

Limitations / things to watch out for

- Free Edition has **quotas** and **limitations** (e.g. on compute, storage, features).
- Free Trial gives you more access, but only for \~14 days. After that, if you don’t upgrade, some services might become unavailable.
- In some signup flows, especially via cloud providers, you may incur charges for cloud resources (storage, VM use) if those are outside what the credit allows. Always check what is free vs what might cost.

Important Databricks Resources

Databricks Training

- Databricks Fundamentals Course
- Getting Started with Lakehouse Architecture
- Get Started With Data Engineering on Databricks
- Redefine what’s possible with generative AI

Databricks Architectures Center

- https://www.databricks.com/resources/architectures