Introduction to Medallion Architecture

Karthick Selvam

May 10, 2025

Contents

1 Introduction: The Data Dilemma and the Lakehouse Concept

Today's enterprise faces a significant data dilemma: despite generating vast amounts of data, a large percentage remains unused. According to **IDC** in **2023**, **73% of enterprise data goes unused** for decision-making. This paradox stems from the limitations of traditional data architectures, which forced organizations to make difficult trade-offs.

Historically, organizations had to choose between:

- **Data Warehouses**: Designed for performance on structured data, excellent for traditional Business Intelligence (BI), but rigid and lacked support for modern workloads.
- **Data Lakes**: Offering flexibility and cost-efficiency for handling diverse data types, but notorious for poor data governance and reliability.

This forced choice led to fragmented architectures, data silos, and operational complexity. The **Data Lakehouse architecture** emerged as a synthesis, designed to **merge the best capabilities of both data lakes and data warehouses** onto a single, unified platform.

The Lakehouse Definition A new open data architecture that merges the flexibility, cost-efficiency, and scalability of data lakes with the data management and asset transaction capabilities of data warehouses.

While the term "Data Lakehouse" was originally coined by **Databricks**, the architecture it represents is now widely adopted. This fusion enables support for both BI and Machine Learning (ML) workloads on a unified platform, eliminating the need for complex, multi-system architectures.

---

2 Evolution of Data Architectures: Warehouses and Lakes

To understand the Lakehouse, let's briefly review the architectural evolution that led to its necessity.

2.1 Data Warehouses: The Era of Structured Analytics

Data warehouses first emerged in the early **1980s** to provide a centralized repository for organizational data. This allowed businesses to move away from departmental silos and consolidate information for comprehensive decision-making.

Key Characteristics:

- Stored primarily **structured operational data**, with some large warehouses including external data. Data was typically **structured**, though semi-structured formats like JSON or XML were sometimes handled. Crucially, they **couldn't process unstructured data** such as images and videos.
- Data went through a rigorous **ETL (Extract, Transform, Load)** process *before* loading, ensuring data quality and consistency upfront.
- Often included **data marts** focused on specific subject areas or regions, containing cleaned, validated, and aggregated data for KPIs.
- Accessed mainly through **Business Intelligence reports**.

By the early **2000s**, most large companies relied on data warehouses, which were crucial for business decision-making based on available structured data.

2.1.1 Challenges of Data Warehouses

The exponential growth of data volumes and the emergence of new data types like videos and images due to the internet posed significant challenges for traditional data warehouses:

Challenges:

- **Limited Data Type Support:** Unable to process unstructured data, missing out on insights from a vast and growing data source.
- **Long Development Times:** The ETL process required thorough quality checks and transformations before loading. This led to **longer development times** to add new data and significant delays (often **24-48 hours of latency**), resulting in stale insights in fast-moving markets.
- **Proprietary Technology & Vendor Lock-in:** Built on traditional relational databases or MPP engines, they used proprietary file formats and could create **vendor lock-in**.
- **Scalability Issues:** Traditional on-premises warehouses were **hard to scale**, sometimes requiring large migration projects.
- **High and Inflexible Storage Costs:** Storage was expensive (**$23/TB**) and it was **impossible to expand storage independently of computing resources**.
- **Insufficient AI/ML Support:** They **didn't provide enough support** for data science, machine learning, and AI workloads, which require flexible access to raw data and iterative processing.

These issues paved the way for data lakes.

2.2 Data Lakes: The Promise of Flexibility

Data lakes were introduced around the year **2011** to address the challenges with data warehouses, particularly handling data variety and reducing cost.

Key Characteristics:

- Designed to handle **all data types**: structured, semi-structured, and crucially, **unstructured data**, which makes up roughly **90%** of modern data.
- Raw data ingested directly (**schema-on-read**) **without any initial cleansing or transformation**. This allowed for quicker solution development and faster ingestion times.
- Built on **cheap storage solutions** like HDFS and cloud object stores such as **Amazon S3** and **Azure Data Lake Storage Gen2**, which kept the costs low (around **$2.30/TB**).
- Utilized **open source file formats** like **Parquet**, **ORC**, and **Avro**, allowing for a wide range of tools and libraries to be used.
- Supported **data science and machine learning workloads** by providing access to both raw and transformed data.

Data lakes offered the necessary flexibility and cost benefits for the big data era. However, there was one major problem: data lakes were too slow for interactive BI reports and lacked proper data governance. This often led companies to copy data to a separate warehouse for BI, resulting in complex, fragmented architectures.

2.2.1 Challenges of Data Lakes: The "Data Swamp"

While data lakes offered flexibility and cost savings, they often became "data swamps" due to a lack of critical data management capabilities and governance:

Challenges (The "Data Swamp"):

- **No ACID Transactions:** This was a major problem. Data lakes did not have built-in support for **ACID** (Atomicity, Consistency, Isolation, Durability), essential for reliable data management. This led to many issues:

  - **Failed jobs** could leave behind partially loaded files, requiring additional cleanup processes.
  - **No guarantee of consistent reads**, leading to the possibility of users accessing partially written data, compromising reliability.
  - **No direct support for updates or deletes**. Developers had to partition the files and rewrite entire partitions just to correct or update data, which was both time-consuming and error-prone.
  - **No way to roll back changes**, making it difficult to recover from failures.
  - Handling ”right to be forgotten” requests (**GDPR**) was challenging. Since deletes weren’t well supported, entire files sometimes had to be rewritten to remove individual data, which was again time-consuming and expensive.
- **Lack of Version Control:** Made it harder to track changes, perform rollbacks, or ensure data governance and reproducibility.
- **Poor Performance for BI:** Data lakes struggled to provide fast, interactive query performance (**12-15 seconds per query**) and **lacked adequate support for basic needs** such as security and governance.
- **Operational Complexity:** Setting up and managing data lakes was complex and required significant expertise.
- **Complex Lambda Architecture:** Streaming and batch data needed to be processed separately, leading to complex **Lambda architectures**.

In summary, data warehouses excelled at handling BI, but lacked support for streaming, data science, and ML. Data lakes were designed for modern workloads but fell short in supporting BI effectively.

---

3 The Data Lakehouse Architecture: The Unified Solution

The **Data Lakehouse architecture** is the synthesis, designed to effectively support **both Business Intelligence (BI) and Data Science, Machine Learning, and AI workloads** on a single platform, eliminating the complexity and limitations of previous approaches.

A Lakehouse is essentially a data lake (built on cheap cloud object storage with open formats) enhanced with data management and governance capabilities borrowed from data warehouses. This is achieved by adding a transactional layer and a unified governance layer on top of the lake storage.

3.1 Lakehouse Core Concepts

Key Enablers:

- **Transactional Storage Layer (e.g., Delta Lake):** Brings **ACID transactions**, schema enforcement, and time travel to data lakes, providing reliability and data management.
- **Unified Governance Layer (e.g., Unity Catalog):** Provides a single source of truth for data discovery, access control, and data lineage across all data assets.

With transactional support offered by the Delta Lake file format, the Lakehouse can **seamlessly combine streaming and batch data processing** on the same tables, **eliminating the need for the complex Lambda architecture**. Data becomes reliable and consistent whether it arrives in real-time or in batches.

The data within the Lakehouse platform can be directly used for data science and machine learning tasks, leveraging the flexibility and access to raw and processed data. Additionally, the Lakehouse integrates directly with popular BI tools such as Power BI and Tableau, allowing analysts to query data without copying it into a separate data warehouse. This is facilitated by underlying query engines optimized for performance and the centralized governance layer (like Unity Catalog) which provides necessary security (e.g., role-based access control) and schema information.

This unified approach **eliminates data silos**, simplifies the architecture, reduces data duplication, and ensures all users are working with the same, reliable data.

---

4 Medallion Architecture: Data Design Pattern for the Lakehouse

As we saw, the Data Lakehouse integrates capabilities of warehouses and lakes into a single entity. To effectively manage and process data within this Lakehouse structure, a common data design pattern is used, known as the **Medallion architecture**. This is again a term originally coined by Databricks and now widely adopted.

It's important to understand that the medallion architecture is not an architectural pattern for the entire platform, but rather a **data design pattern** or structure for organizing data within the Lakehouse storage layers. While layered data design patterns have existed before, Databricks refined this within the Lakehouse context using a simple, intuitive structure.

The Medallion Architecture Pattern A data design pattern where data is organized into layers (typically Bronze, Silver, and Gold) within a Data Lakehouse. Data flows through these layers, improving in quality and structure at each stage.

This is where the term ”medallion” comes from – representing the progression through distinct, valuable layers. Data flows through these layers, and as it progresses, its quality and structure improve, making it suitable for different downstream use cases.

While the **three-layer structure (Bronze, Silver, Gold)** is most common, the medallion architecture is flexible. Some complex projects may include a fourth layer (e.g., Platinum), or simpler projects might only require two layers. The key is to **clearly identify and define each layer's purpose, characteristics, and expected data quality upfront** for maximum benefit.

4.1 The Layered Structure: Bronze, Silver, Gold

Bronze Layer: Raw Data

- Typically contains **raw data** as received from various data sources (batch or streaming).
- Undergoes **minimal to no transformation**. At most, additional metadata like a load timestamp or source filename may be added for tracking.
- **Crucial for auditing** and identifying data issues.
- Maintains a **historical record** of all data received, making it easy to **replay data** if pipelines fail.
- Supports **fast ingestion** of high volume and high velocity data.

Silver Layer: Filtered, Cleansed, Enriched

- Holds **filtered, cleansed, and enriched data.**
- A **structure is applied**, and schema is either enforced or evolved to maintain consistency.

- Data undergoes **quality checks**: invalid records removed, column values standardized, duplicates eliminated, missing values handled.
- **Context or descriptions** are added.
- Results in **structured, high quality, and reliable data**.
- Data in this layer is suitable for **data science, machine learning, and AI workloads**.

Gold Layer: Business-Level Aggregated Data

- Contains **business-level aggregated data.**
- Data from the Silver layer is **further aggregated and enriched** with additional context.
- **Ready for high-level business reporting and analysis**.
- Also used for **advanced analytics and applications** where needed.

4.2 Benefits of the Medallion Architecture

Designing a data Lakehouse using the medallion architecture provides several key benefits:

Key Benefits:

- **Improved Data Lineage and Traceability:** Moving data through clearly defined layers makes it easier to track data origin, transformations, and usage in applications.
- **Enforced Data Governance and Compliance:** Each layer has defined meaning and granularity, helping to define and enforce policies such as retention policies (e.g., deleting raw data from Bronze after a period but retaining processed data longer). Suitable for regulations like **GDPR, CCPA**, etc.
- **Supports Incremental Processing:** Allows pipelines to process only new and changed data received at each layer, reducing computational cost and improving performance compared to full refreshes.
- **Better Workload Management and Scalability:** The layered approach allows for better workload isolation and management for processing each layer, contributing to creating scalable solutions.
- **Enhanced Security Control:** Provides better control over security by defining role-based access control (**RBAC**) at the layer level. For example, users who only need high-level reports can be granted access *only* to the Gold layer, restricting access to granular transactional data available in Bronze and Silver.

In summary, the medallion architecture provides a flexible data design pattern that can be adapted based on project requirements. By clearly defining each layer's purpose and characteristics, organizations can ensure data consistency, quality, and efficiency throughout their data pipeline within the Lakehouse.

---

5 Introducing the Databricks Platform

Now that we have an understanding about the Data Lakehouse concept and the Medallion architecture pattern used within it, let's introduce **Databricks**, the platform that helps organizations build and leverage data lakehouses effectively. We'll start with a high-level overview and then explore its key components.

5.1 Databricks: Founded by the Creators of Spark

At its core, the Databricks platform is built around the open source distributed compute engine, **Apache Spark**, which is widely used in the industry for building big data and machine learning projects. Spark is a fast, unified analytical engine designed for big data processing and machine learning.

Spark was originally developed at **UC Berkeley** in **2009** and open sourced in **2010**. In **2013**, it became a project under the Apache Software Foundation and has seen significant adoption since. Companies like **Yahoo, eBay, and Netflix** use Spark for large scale data processing, handling petabytes of data on clusters with thousands of nodes.

Spark was designed to address the limitations of **Hadoop**, which was widely used as the big data processing engine at the time. Hadoop was slow and inefficient for interactive and iterative computing tasks. In contrast, Spark provides simpler and faster APIs. It can be **up to 100 times faster** than Hadoop for certain workloads by utilizing in-memory computing and various optimizations. Like most big data engines, Spark runs on a distributed computing platform. It has a **unified engine** that supports both batch and streaming workloads. Spark also includes built-in libraries for **SQL queries**, **machine learning**, and **graph processing**, making it versatile.

While Spark is a powerful engine, **setting up clusters**, **managing security**, and using third party tools to write programs can be quite **challenging**. This is where Databricks comes in.

5.2 Key Components of the Databricks Platform

Databricks was founded by the original creators of Apache Spark to **make working with Spark easier** by providing the essential management layers and a unified platform. It is available on all major cloud platforms including **Microsoft Azure, AWS, and Google Cloud**.

The Databricks platform provides a comprehensive suite of tools and services built on the foundation of Apache Spark and implementing the Lakehouse architecture.

5.2.1 *Delta Lake: The Transactional Foundation*

Delta Lake is an open-source storage layer that brings reliability to the data lake. Its key features include:

- **ACID Transactions:** Ensures data integrity, prevents corruption from failed writes, and provides consistent views for readers.
- **Time Travel (Data Versioning):** Allows access to historical versions of data. This is crucial for audits, rollbacks, and reproducing ML experiments.
- **Schema Enforcement & Evolution:** Enforces schema on writes to prevent bad data entry while allowing schemas to evolve over time without disruptive rewrites. **Upserts/Deletes:** Enables efficient updates and deletes directly on the data lake, simplifying compliance and data correction.
- **Optimization Features:** Includes techniques like Z-ordering to optimize data layout, leading to significantly faster query performance (e.g., 94% faster scans for certain queries).

Delta Lake transforms the unreliable data lake into a reliable, transactional database layer, forming the basis of the Bronze, Silver, and Gold layers in the Medallion architecture.

5.2.2 *Unity Catalog: Unified Governance*

Unity Catalog is a centralized metadata, governance, and security layer for the Lakehouse. It provides:

- **Single Source of Truth:** A unified view and management plane for data, AI, and analytics assets across clouds and data types.
- **Fine-grained Security:** Offers robust Role-Based Access Control (RBAC), including row-level and column-level security, essential for meeting strict data privacy and compliance requirements (e.g., HIPAA). This is critical for implementing layer-specific security in the Medallion architecture.
- **Automated Data Lineage:** Automatically tracks how data flows and is transformed, providing essential audit trails for compliance (e.g., SOX) and making data lineage in the Medallion architecture clear.
- **Global Data Search & Discovery:** Makes it easy for users to find relevant data assets quickly (e.g., $\sim$200ms metadata retrieval across millions of assets).
- **Audit Trails:** Logs access to data for monitoring and compliance.

Unity Catalog addresses the governance failures of raw data lakes, enabling secure and compliant data sharing and access across the different Medallion layers.

Other key components of the Databricks platform that enhance the Lakehouse experience and help implement the Medallion architecture include:

- **Simplified Cluster Management:** Easily spin up and manage Spark clusters with just a few clicks. You can choose from different **runtime options** (general purpose, memory optimized, GPU support for ML) to fit your needs for processing data at different layers (Bronze, Silver, Gold).
- **Integrated Development Environment:** Provides an integrated **Jupyter-style notebook IDE** for creating and running code in multiple languages (Python, SQL, Scala, R), with built-in collaboration and version control integration (**Git**). Used for developing the pipelines that move data between Medallion layers. It also includes **administrative controls** to help manage user access to workspaces and clusters, ensuring secure usage.
- **Optimized Spark Runtime:** Databricks provides a Spark runtime which is highly optimized for the Databricks platform, known to be **up to 5 times faster** than the vanilla Apache Spark. This speeds up data processing at every layer.
- **Photon Engine:** Databricks also comes with a **vectorized query engine** called **Photon**, which provides extremely fast query performance, offering **up to 8 times improvement** on the standard Databricks runtime for SQL and data frame operations. Essential for high-performance processing, especially in the Silver and Gold layers.

- **Delta Live Tables (DLT):** Recently introduced, DLT is a **declarative ETL framework** that helps create reliable data pipelines with ease, automating error handling and data quality checks. DLT is specifically designed to simplify building Medallion architecture layers (Bronze to Silver, Silver to Gold).
- **Databricks Workflows:** A built-in feature that helps you **schedule and orchestrate tasks and pipelines** (often DLT pipelines) as required to automate the data flow between Medallion layers.
- **Databricks SQL:** Provides data analysts a **SQL based analytical environment** to explore data, create dashboards, and schedule regular refreshes of the dashboard. Primarily used for querying data in the higher-quality Silver and Gold layers.
- **Managed MLflow:** Includes a fully managed service for managing the end-to-end **machine learning lifecycle**, including experiment tracking, model training, deployment, and model registry. ML workloads typically consume data from the Silver layer.
- **Databricks IQ:** The latest addition, Databricks IQ serves as an **AI assistant** to help users develop and debug code, add commands, and create dashboards, leveraging generative AI. Can assist developers building pipelines across Medallion layers.

5.3 Cloud Platform Integration

Databricks is available as a service on all three major cloud providers: **Microsoft Azure**, **AWS**, and **Google Cloud**. The integration with the cloud providers is quite similar, except that **Azure hosts Databricks as a first-party service**. This means on Azure, you get a unified billing and direct support from Microsoft for all your services.

Deep Cloud Integration:

- **Availability:** Available on Azure (as **Azure Databricks**), AWS (as **Databricks on AWS**), and Google Cloud (as **Databricks on Google Cloud**).
- **Leveraging Cloud Security and Governance:** Integrates with native cloud security and identity services such as **Azure Active Directory**, **AWS Identity and Access Management (IAM)**, and **Google Cloud IAM**, etc. These are leveraged by Unity Catalog to enforce security policies down to the layer level.
- **Integration with Cloud Storage:** Integrates with storage services such as **Azure Data Lake Storage Gen2**, **AWS S3**, **Google Cloud Storage**, etc., which form the basis of the Lakehouse storage for the Medallion layers.
- **Cloud Compute Resources:** The underlying virtual machines for the Databricks clusters are provisioned from the user's cloud account. These resources are used to power the processing pipelines between the Medallion layers.
- **Cloud Monitoring Integration:** Can use the monitoring services provided by the cloud providers to help track and analyze Databricks workloads processing data through the layers.
- **DevOps Integration:** Integrates with DevOps services offered by the cloud providers such as **Azure DevOps** to help enable **Continuous Integration and Continuous Deployment (CI/CD)** for the data pipelines and code that build the Medallion layers.

In summary, Databricks is a Spark based, unified data analytics platform that's optimized for each of the major cloud providers, making the Data Lakehouse architecture and the Medallion design pattern practical and powerful.

---

6 Benefits and Performance of the Lakehouse (Databricks Implementation)

The Lakehouse architecture, as implemented by the Databricks platform and structured with the Medallion design pattern, delivers significant advantages over traditional fragmented approaches:

6.1 Key Benefits

Key Benefits:

- **Handles All Data Types:** Processes structured, semi-structured, and unstructured data on a single platform.
- **Cost-Effective Storage:** Leverages cheap cloud object storage (around **$2.30/TB**), offering roughly a **90% reduction** in storage costs compared to data warehouses.
- **Unified Workloads:** Supports BI, SQL analytics, Data Science, Machine Learning, AI, and streaming on a single, consistent copy of the data. Data is organized and refined through the Medallion layers for different workload needs.
- **Direct BI Integration:** BI tools connect directly to the Lakehouse via **Databricks SQL**, typically querying the Silver and Gold layers, ensuring analysts have access to the most up-to-date, refined data without duplication.
- **Reliability and Data Management: ACID transactions**, versioning (time travel), schema management, and efficient upserts/deletes prevent data swamps and ensure data quality, enabled by **Delta Lake** at the storage layer for all Medallion stages.
- **Improved Performance:** Query engines like **Photon**, combined with optimized Spark runtime and Delta Lake features (like Z-ordering), provide competitive or superior performance for many workloads across the layers.
- **Simplified Architecture:** Eliminates the need for multiple specialized systems (warehouses, lakes, separate streaming, governance) by implementing the Lakehouse and Medallion pattern on a single platform, reducing operational overhead and complexity.
- **Managed Services:** Databricks provides managed services for clusters, notebooks, MLflow, DLT, etc., reducing operational burden on data teams building and managing the Medallion layers.
- **Enhanced Developer Productivity:** Integrated IDE, AI assistance (**Databricks IQ**), and declarative frameworks (**DLT**) boost productivity for building and maintaining the ETL/ELT pipelines between Medallion layers.
- **Improved Data Lineage and Traceability (via Medallion):** Clear progression through Bronze, Silver, and Gold layers makes tracking data flow straightforward.
- **Enhanced Governance and Compliance (via Medallion & Unity Catalog):** Layered structure combined with Unity Catalog facilitates defining and enforcing data quality, security, and retention policies at each stage.
- **Supports Incremental Processing (via Medallion Delta Lake):** Delta Lake features enable efficient processing of changes between layers.
- **Better Workload Management & Scalability (via Medallion Databricks):** Layers isolate processing requirements, managed by scalable Databricks compute.
- **Layer-Specific Security (via Medallion & Unity Catalog):** RBAC can be applied at the layer granularity for fine-grained access control.

6.2 Performance Benchmarks

Benchmarks highlight the tangible performance and cost benefits achievable with the Databricks Lakehouse platform:

Metric	Improvement	Context
BI Query Speed (TPC-DS)	**Up to 8x faster** (with Photon)	Lakehouse query engines are highly optimized for SQL workloads, typically against Silver/Gold layers.
Spark Runtime Speed	**Up to 5x faster** vs Vanilla Spark	Databricks-optimized Spark engine provides significant acceleration for processing data across layers.
Storage Cost	**90% Reduction**	Lakehouse utilizes inexpensive cloud object storage (e.g., S3, ADLS Gen2, GCS) for all Medallion layers.
Data Pipeline Failures	**83% Reduction**	Achieved by customers due to ACID reliability and schema enforcement (Delta Lake) when building layers.
ML Model Deployment	**6x Faster**	Enabled by unifying data prep (often in Silver layer) and ML on a single, reliable platform with tools like MLflow.

Table 1: Lakehouse Performance Gains and Cost Reductions (Source: Databricks internal benchmarks and customer reports)

These improvements directly translate to faster time-to-insight, reduced operational costs, and increased agility for data teams leveraging the Lakehouse and Medallion architecture.

---

7 Real-World Adoption and Getting Started

The Lakehouse architecture, powered by platforms like Databricks and structured with patterns like Medallion, is being adopted by leading organizations across industries to drive significant business outcomes.

7.1 Customer Success Stories

Examples of Impact:

- **Goldman Sachs**: Processing **45TB/day** of transaction data for **sub-second fraud detection**. Likely involves transforming raw data through structured layers for real-time analytics.
- **Moderna**: Accelerating vaccine research data cycles from **weeks to hours**. Implies rapid processing and refinement of diverse data types through a layered structure for scientific analysis.
- **Starbucks**: Unifying and leveraging real-time store and mobile app data for personalized recommendations. Requires processing high-velocity data into reliable, consumable layers for applications.
- **Unilever**: Consolidating **17PB** of previously siloed data, leading to **83% fewer pipeline failures** and **6x faster ML deployment**. Achieved by implementing a reliable, layered Lakehouse structure.

7.2 Getting Started with Lakehouse on Databricks

Migrating to or implementing a Lakehouse on Databricks typically follows a phased approach that aligns well with the Medallion pattern:

1. **Assess Current Architecture:** Understand existing data sources, workloads, and identify key pain points the Lakehouse can solve. Define the target Medallion layers needed.
2. **Pilot with Databricks & Delta Lake:** Start by landing new data or converting a subset of existing data into the **Bronze layer** using Delta Lake format within the Databricks environment. Build initial pipelines (perhaps using DLT) to create Silver and Gold layers for a specific use case to demonstrate value and build expertise.
3. **Implement Unity Catalog:** Establish centralized governance, security (including layer-specific RBAC), and discovery using Unity Catalog for the pilot data across Bronze, Silver, and Gold layers, and plan for broader implementation across your Lakehouse assets.
4. **Migrate High-Value Workloads:** Move critical BI and AI workloads onto the Databricks Lakehouse, targeting the appropriate Silver and Gold layers. Consolidate your data landscape by migrating other sources into the Bronze layer and building out the Medallion layers systematically. Leverage Databricks SQL for BI and MLflow/Workflows for data science and orchestration.

---

8 Conclusion: The Future is Lakehouse

In summary, data warehouses excelled at traditional BI but were ill-equipped for modern data diversity and AI. Data lakes offered flexibility and cost but suffered from reliability and governance issues. The **Data Lakehouse architecture**, structured with design patterns like the **Medallion architecture**, combines the best of both worlds, providing a unified, reliable, cost-effective, and high-performance platform for all data, analytics, and AI workloads. Enabled by foundational technologies like **Delta Lake** and **Unity Catalog**, and powered by platforms like **Databricks**, the Lakehouse simplifies the data stack, accelerates innovation, and allows organizations to finally leverage the vast majority of their data that currently remains unused. The Lakehouse is not just another architecture; it is becoming the **de facto standard** and the foundation for the next decade of data-driven innovation.

Thank You.