ONEHOUSE

Building a Universal Data Lakehouse

Updated: 11/15/23

Introduction

Businesses are producing and leveraging data at an unprecedented rate. The amount of data produced is growing at a compound annual rate of nearly 20% and is expected to surpass 180 zettabytes created per year by 2025. As data volumes grow, new ways of processing and querying the data have emerged, such as:

- Apache Spark (Spark for brevity) for processing data at massive scale with distributed computing
- Flink for processing data in real-time with event streams
- Data warehouses for querying data fast by using methods like advanced optimizations and caching
- Federated SQL query engines to query data across disparate sources like databases, data lakes, and warehouses.
- Real-time analytics engines for querying data even faster with techniques like indexing and local storage architectures.
- Specialized compute frameworks, optimized for training machine learning models and powering data science.

The separation of storage and compute found in the cloud has encouraged the development of compute engines - platforms for processing data in various ways. All of the above technologies can be seen as compute engines, some special-purpose, others with a wide range of capabilities.

Innovation continues to accelerate, with AI startups more than doubling their share of the total venture funding up to 26% in 2023 – and much of that going to the data infrastructure that powers AI and analytics.

To keep up with the rapid evolution of these technologies, organizations must build a future-proof data architecture that allows them to maintain high-quality data while ensuring that their data is not locked into a single vendor ecosystem or compute engine. Too many organizations have invested in building a data platform, only to find out years later that costs have risen astronomically as the business demands new data use cases.

This whitepaper defines the universal data lakehouse architecture, a next-generation approach for managing data that leverages the best features of the data lake, such as open-source data formats and the use of object storage, while supporting incremental updates like a data warehouse does, with cross-engine cross-cloud interoperability. The data lakehouse optimizes data pipeline costs, simplifies data preparation, and makes data universally accessible for a wide range of compute engines and use cases. Countless industry leaders have built their data infrastructure around the lakehouse concept, including Uber, GE, TikTok, Amazon, Walmart, Disney, Twilio, Robinhood, and Zoom.

With the universal data lakehouse architecture, your organization can achieve the following benefits:

- Unify data around one source of truth
- Bring the best possible compute engine to each of your diverse workloads
- Empower your organization with high data quality
- Reduce costs by combining cost-efficient storage and compute
- Achieve faster performance and fresher data by processing data incrementally
- Keep data private on your own cloud accounts
- Avoid vendor lock-in by storing data in open source formats within your own cloud bucket
- Simplify access control and data sharing

Beyond introducing the universal data lakehouse architecture, this whitepaper also provides strategic guidance on building your data infrastructure. By the end, you will learn:

- The history of the data warehouse, data lake, and data lakehouse – and how this led to the universal data lakehouse architecture
- How to build your data integration and ingestion to optimize costs, meet data freshness goals, ensure data privacy, and enable interoperability
- How to prepare your data in a way that ensures data quality, saves on costs and manual efforts through amortization, and processes data incrementally
- How to choose the right compute engine for any use case across analytics and reporting, data science, real-time analytics, and more

Defining the Universal Data Lakehouse

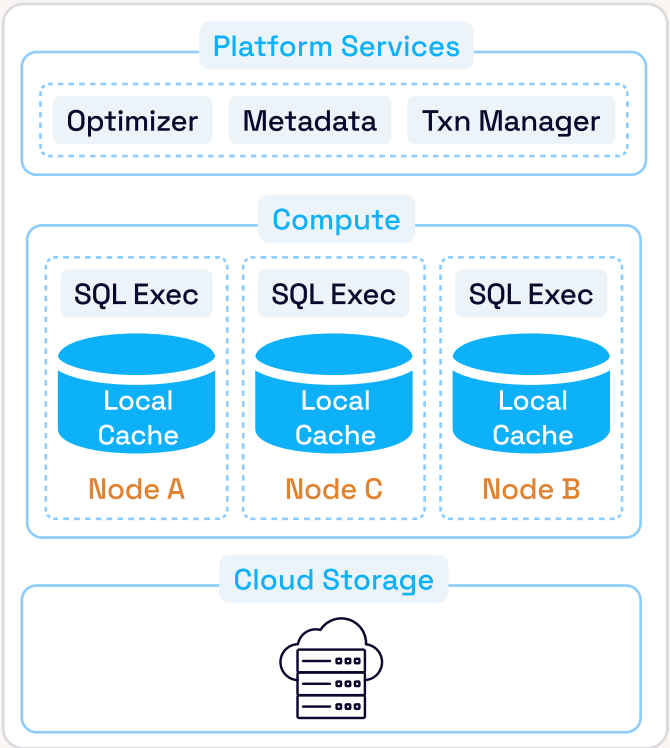
The universal data lakehouse architecture is a new innovation in data infrastructure that is being adopted by organizations on the cutting edge of technology and is now moving toward the mainstream. Every data engineering practitioner and IT manager should understand the data lakehouse and its benefits, while looking for opportunities to deploy the lakehouse productively in their organization.

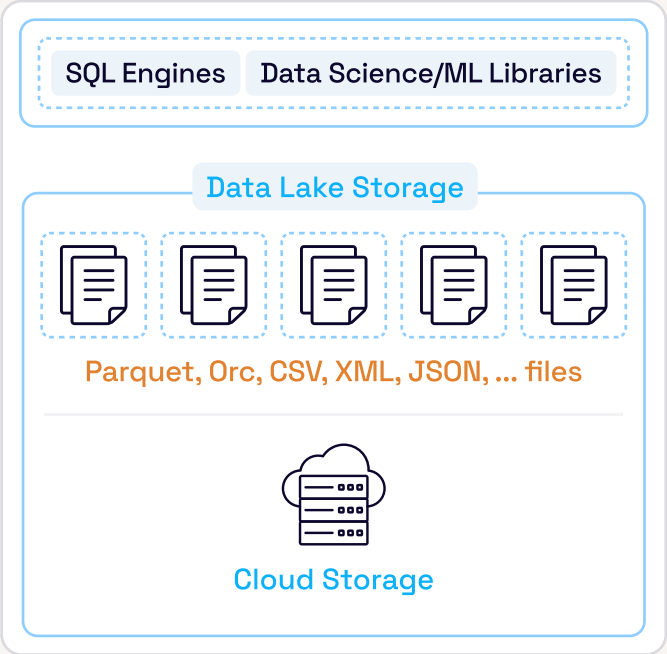
The data warehouse, the data lake, and the data lakehouse

Before diving into the universal data lakehouse architecture, it's essential that you first understand the differences between a data warehouse, a data lake, and a data lakehouse.

The Data Warehouse

For decades, the data warehouse has served as the de facto database for analytics, offering capabilities that include ACID transactions, strict schema enforcement, and SQL support. Unlike transaction-oriented databases such as MySQL or PostgreSQL, which use row-oriented data formats, data warehouses store data in columnar formats to efficiently process common analytics query patterns such as aggregations. Over the last few years, data warehouses have moved from on-premises into the cloud, separating storage and compute to enable flexibility and elastic scaling.



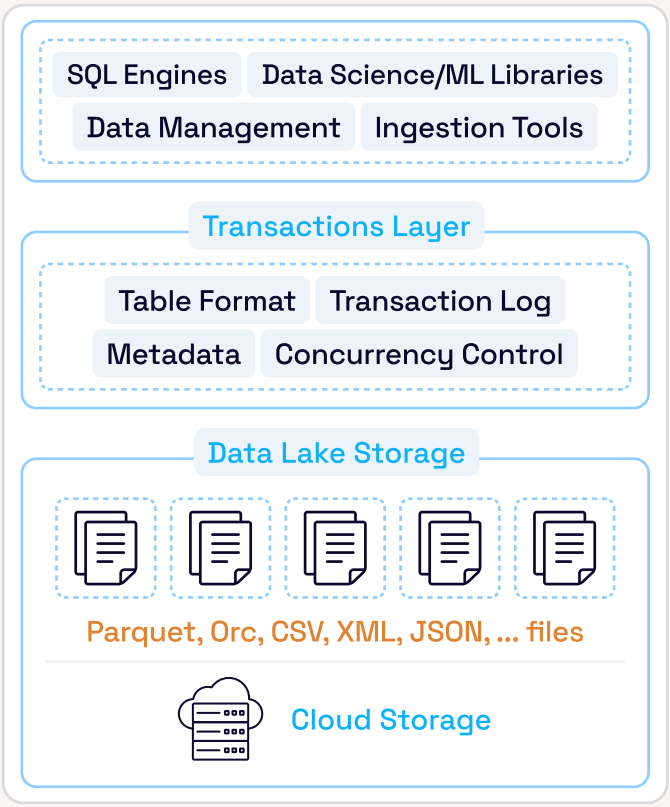


The Data Lake

While data warehouses primarily serve analytics use cases like business intelligence, data lakes emerged to cover the warehouse's weak points, such as handling unstructured data and processing large-scale data workloads with cost-efficient object storage and decoupled compute to process data at scale. However, data lakes come with their own challenges: cloud object storage is append-only with no updates to data in place, it is difficult to manage query performance, and data quality is a nightmare to maintain.

The Data Lakehouse

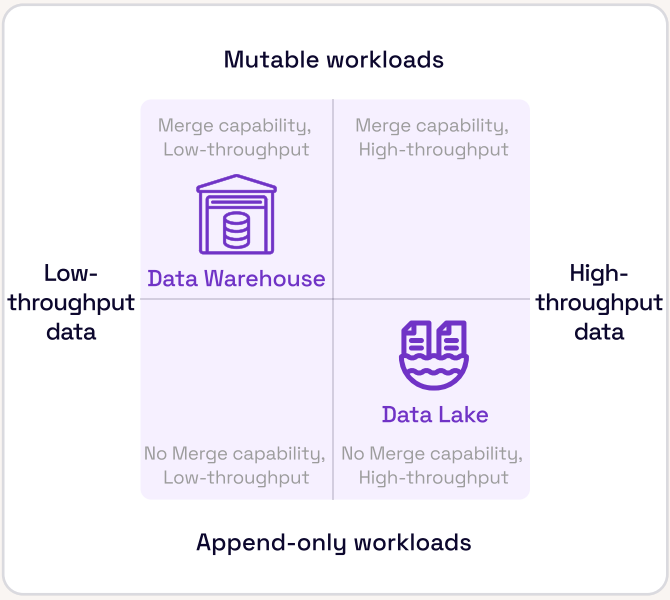
In 2016, the very first data lakehouse was created at Uber to fill the gaps left by the warehouse and the lake. The lakehouse blends the database-like capabilities of the warehouse with the flexibility, cost-efficiency, and scalability of the lake. The lakehouse achieves this by adding a transaction layer over the raw file storage in the lake. The lakehouse is built atop open-source table formats like Apache Hudi, Apache Iceberg, and Delta Lake (Linux Foundation), enabling full data ownership and portability - an advantage over the vast majority of data warehouses, which store data in proprietary formats.



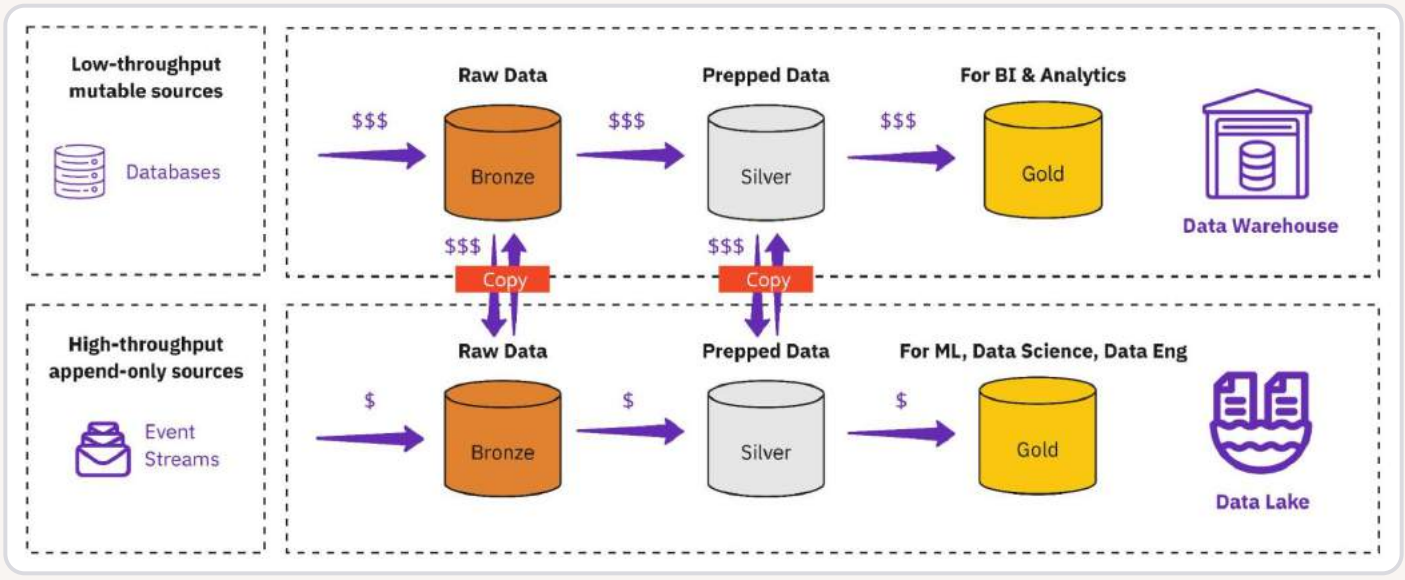
Why do we need the data lakehouse?

Before the creation of the data lakehouse, data warehouses and data lakes fulfilled separate requirements:

- The data warehouse handled low-throughput, mutable workloads like database replication
- The data lake handled high-throughput, append-only workloads like event streams



In most pre-lakehouse data architectures, organizations ended up maintaining both a data warehouse and data lakes. To consolidate data across sources, they would periodically copy data between the data warehouse and data lake, creating complex pipelines and multiple copies of identical or similar data. The data warehouse with its fast queries serves business intelligence (BI) and reporting use cases, while the data lake, with its support for unstructured storage and low-cost compute, serves use cases for data engineering, data science, and machine learning.



In the diagram above, you see an example of the medallion architecture, used across two separate workflows for mutable transactional data and immutable data from event streams. The typical "modern data stack" is born by replicating operational data into a raw "bronze" layer on a cloud data warehouse, using point-to-point data integration tools. This data is then subsequently cleaned, audited for quality, and prepared into a "silver" layer. Then a set of ETL jobs transform this data into facts, dimensions, and other models to ultimately create a "gold" data layer, ready to power analytics and reporting.

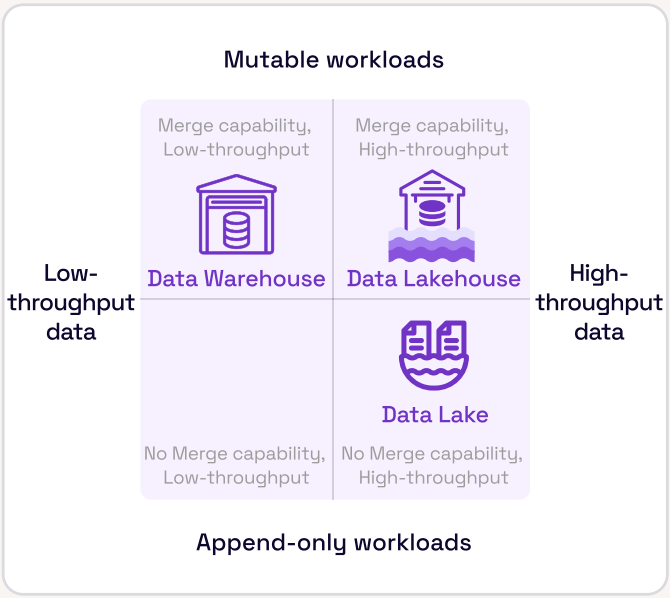
Sustaining the architecture above across both a data warehouse and data lake is challenging, expensive, and error-prone. Periodic data copying between the lake and the warehouse leads to stale and inconsistent data. Governance becomes a headache for everyone involved, as access control is split between systems and data deletion (think GDPR) must be managed on multiple copies of the data - including copies housed in immutable storage. Not to mention, teams are on the hook for each of these complex pipelines, and ownership can quickly become murky.

This introduces many challenges for an organization:

1. **Vendor lock-in**: The source of truth for high-value operational data is often a proprietary data warehouse, which creates lock-in points.
2. **Expensive ingestion and data prep**: While data warehouses offer merge capabilities for mutable data, they lack the ability to quickly and cost-effectively ingest data from upstream databases or data streams. The warehouse's premium compute engine is optimized for high-performance serving of analytics at the gold layer. This expensive compute engine is often also employed for commodity workloads such as data ingestion and data preparation at the bronze and silver layers. This typically results in ballooning costs for workloads that could easily have run on cheaper, purpose-built compute layers.
3. **Wasteful data duplication**: As new use cases emerge, organizations duplicate their work across lakes and warehouses, wasting storage and compute resources. For example, the same data is ingested/copied once for analytics and once for data science, wasting engineering and cloud resources. Considering that organizations also provision multiple environments such as development, staging, and production, the compounded costs across the entire infrastructure can be staggering.
4. **Poor data quality**: Individual teams often reinvent the foundational data infrastructure for ingesting, optimizing, and preparing data in a piecemeal fashion. These efforts have frustratingly slow ROI or fail altogether due to a lack of resources, putting data quality at risk across the organization, as data quality is only as strong as the weakest data pipeline.
5. **Data governance challenges**: The costs associated with deleting data to comply with regulations such as GDPR, and CCPA are incurred multiple times across each copy of the same data flowing in through a separate entry point. Furthermore, the risks of a mistake increase as multiple copies of the same data are persisted.

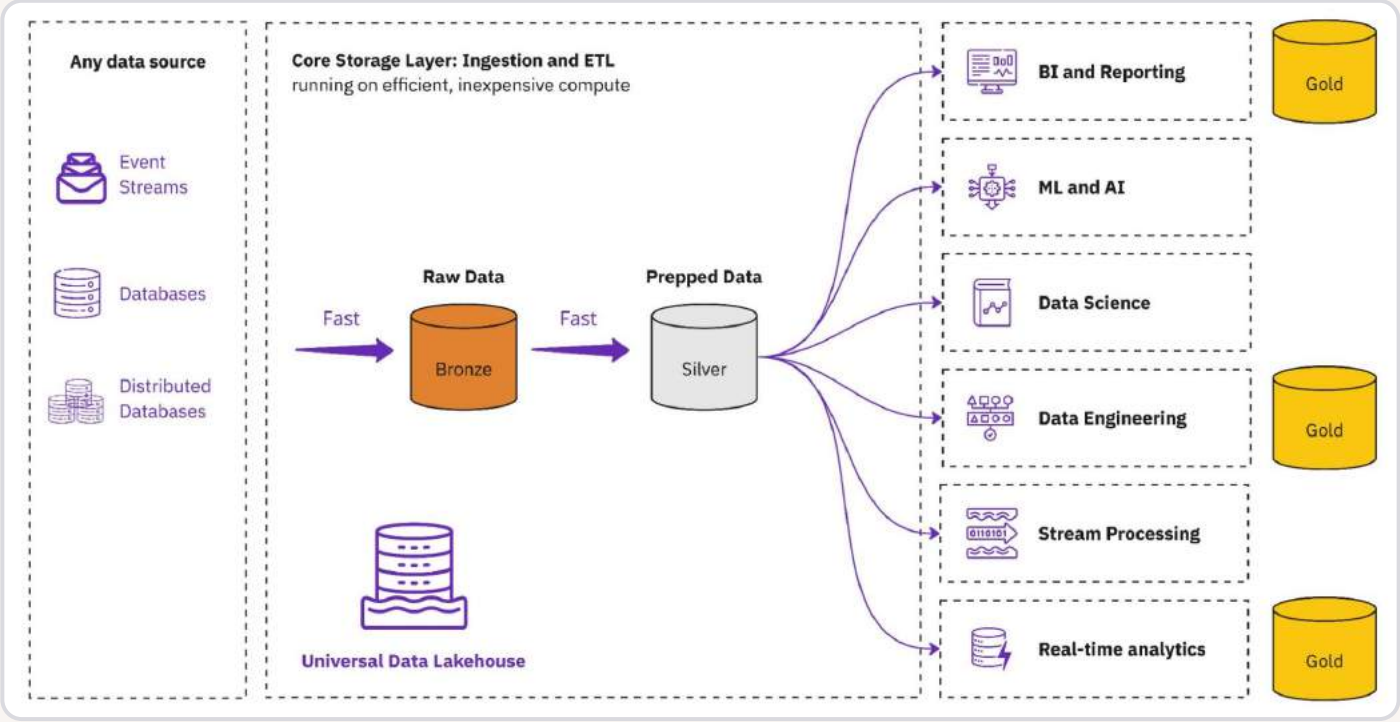
Uber, facing these same challenges, introduced the transactional data lake (now called the data lakehouse) in 2016 to unify their data into a single layer with the strengths of both the data lake and the data warehouse.

This innovation finally made it possible to consolidate high-throughput streaming workloads with mutable workloads (like database replication) in a single storage layer. Now reaching maturity, with thousands of companies using it, the lakehouse offers an opportunity to shed old paradigms and the challenges that came with them, unlocking a superior approach to building data platforms.



Introducing the universal data lakehouse architecture

The universal data lakehouse architecture puts a data lakehouse at the center of your data infrastructure, giving you a fast, open, and easy-to-manage source of truth for all your analytics workloads - business intelligence, data science, and more.



By adopting the universal data lakehouse architecture, organizations can overcome the previously insurmountable challenges of the old, disjoint architecture that continually copies data between the lake and the warehouse. The thousands of organizations already using both data lakes and data warehouses can replace that bifurcated architecture with a data lakehouse for the bronze and silver layers, while continuing to use one or more data warehouse platforms for downstream query serving.

With this architecture, your organization can reap the following benefits:

Unifying Data

The universal data lakehouse architecture uses a data lakehouse as the source of truth inside your organization's low-cost object storage, with data stored in open source formats. Additionally, the lakehouse can handle the ever-increasing scale of upstream databases (such as complex distributed databases like CockroachDB).

Ensuring Data Quality

This universal storage layer provides a convenient entry point to perform data quality checks, schematize semi-structured data, and enforce data contracts between data producers and consumers. Data quality issues can be contained and corrected within the bronze and silver layers, ensuring that downstream tables are always built on fresh, high-quality data. This streamlining of the data flow simplifies the architecture, reduces cost by moving workloads to cost-efficient compute, and eliminates duplicate compliance efforts for requirements such as data deletion under GDPR.

Reducing Costs

Since both operational data from databases and high-scale event data are ingested and processed across a single bronze and silver layer, ingestion and data prep can run just once on low-cost compute. We have seen impressive examples of multi-million dollar savings from cloud data warehouse costs by moving ingestion and ELT workloads to this architecture on a data lakehouse.

Keeping data in open formats enables all data optimizations and management costs to be amortized across all three layers, instead of each compute engine imposing its own redundant storage optimizations, bringing dramatic cost savings to your data platform.

Faster Performance

The universal data lakehouse improves performance in two ways. First, it's designed for mutable data, rapidly absorbing updates from change data capture (CDC), streaming data, and other sources. Second, it opens the door to moving workloads away from big, bloated batch processing to an incremental model for cost-efficiency and fresher data for analytics. Uber saved ~80% in overall compute cost by using Hudi for incremental ETL. They simultaneously improved performance, data quality, and observability.

Bringing Freedom to Choose Compute Engines

Unlike a decade ago, today's data needs don't stop at traditional analytics and reporting. Data science, machine learning, and streaming data are mainstream and ubiquitous across Fortune 500 companies, mid-sized companies, and startups alike. Emerging data use cases such as deep learning and LLMs are bringing a wide variety of new compute engines optimized for each workload independently. **The conventional wisdom of picking one warehouse or lake engine up front throws away many of the advantages that the cloud offers; the universal data lakehouse makes it easy to spin up the right compute engine on demand for each use case.**

The universal data lakehouse architecture makes data accessible across all major data warehouses and data lake engines and integrates with any data catalog – a major shift from the prior approach of coupling data storage with a single compute engine. This architecture enables you to seamlessly build specialized downstream gold layers across BI & reporting, machine learning, data science, and countless additional use cases, using the engine that is the best fit for each unique job. For example, Spark is great for data science workloads, while data warehouses are battle-tested for traditional analytics and reporting. Beyond technical differences, pricing and the move to open source play a crucial role in which compute engines an organization adopts for each use case.

For example, Walmart built their lakehouse on Apache Hudi, ensuring that they could easily leverage new technologies in the future by storing data in an open source format. They use the universal data lakehouse architecture to empower data consumers to query the lakehouse with a wide range of technologies, including Hive, Spark, Databricks, Presto, Trino, BigQuery, and Flink.

Taking Back Ownership of Your Data

All the source-of-truth data are held in open source formats in the bronze and silver layers within your organization's cloud storage buckets. Data stays in your account across the entire lifecycle, so you can avoid using proprietary data ingestion tools that create security risks by moving data out of your account.

Accessibility of data is dictated by you – not by an opaque third-party system with vendor lock-in. This architecture gives you the flexibility to run data services inside the organization's cloud networks (rather than in vendors' accounts), to tighten security, and to support highly regulated environments.

Additionally, you'll be free to either manage data using open data services or to buy managed services, avoiding points of lock-in data services.

Simplifying Access Control

With data consumers operating on a single copy of the bronze and silver data within the lakehouse, access control becomes much easier to manage and enforce. The data lineage is clearly defined, and teams no longer need to manage separate permissions across multiple disjoint systems and copies of the data.

Implementing and Optimizing the Universal Data Lakehouse

The universal data lakehouse architecture offers the opportunity to transformationally improve your data infrastructure and achieve all of the idealistic goals above. However, the architecture is only as good as its implementation.

The rest of this whitepaper offers tactical advice for successfully building your data platform with the universal data lakehouse architecture. We will dive deep into each of the key components:

1. Data ingestion
2. Data preparation
3. Selecting compute engines for your downstream analytics use cases

By understanding the best practices and common pitfalls for each of these components, you will be well-equipped to lead your team through a successful data transformation. Furthermore, these best practices can be adopted incrementally to provide value without completely overhauling your data infrastructure.

Best practices for data integration

Data ingestion is the process of extracting data from source systems such as a database (eg. MongoDB or PostgreSQL), event stream (eg. Kafka or Pulsar), or file storage (eg. Amazon S3 or Google Cloud Storage) and loading it into the data lakehouse.

There are two popular approaches to ingesting data for analytics:

1. Extract, Load, Transform (ELT): Source data is ingested into raw tables (the bronze layer, in the medallion architecture), then transformed into separate cleaned and filtered tables (the silver layer)
2. Extract, Transform, Load (ETL): Source data is ingested directly into cleaned and filtered tables

Choosing the right approach depends on your business needs. ETL reduces upfront costs by storing less data. ELT maintains a copy of the raw data, which can be useful in case you later need to backfill data due to new requirements or issues with the pipeline. This is safer than relying on source systems like Kafka which may not maintain the data indefinitely.

Whether you use ETL, or ELT, or both, ingesting data presents challenges, that you can solve with a combination of the universal data lakehouse architecture and leveraging the right technologies. You should consider the following when building an ingestion pipeline using the universal lakehouse architecture:

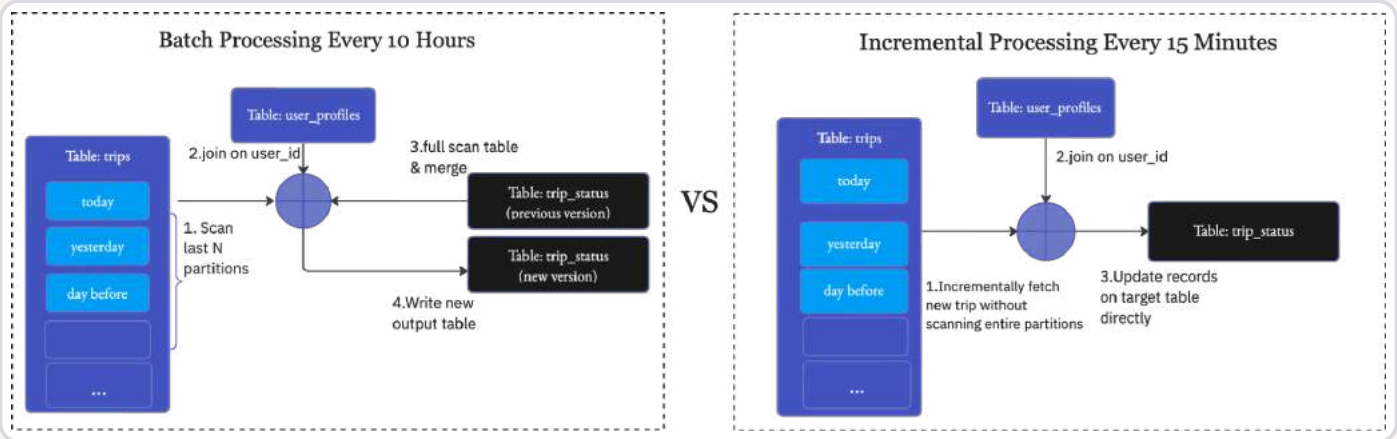
Optimize Costs

Data integration can become extremely expensive as you bring more data into your analytics systems more frequently. Often, costs will become so prohibitive that organizations are forced to reduce the frequency of their data syncs, leading to stale data that is hours behind the source.

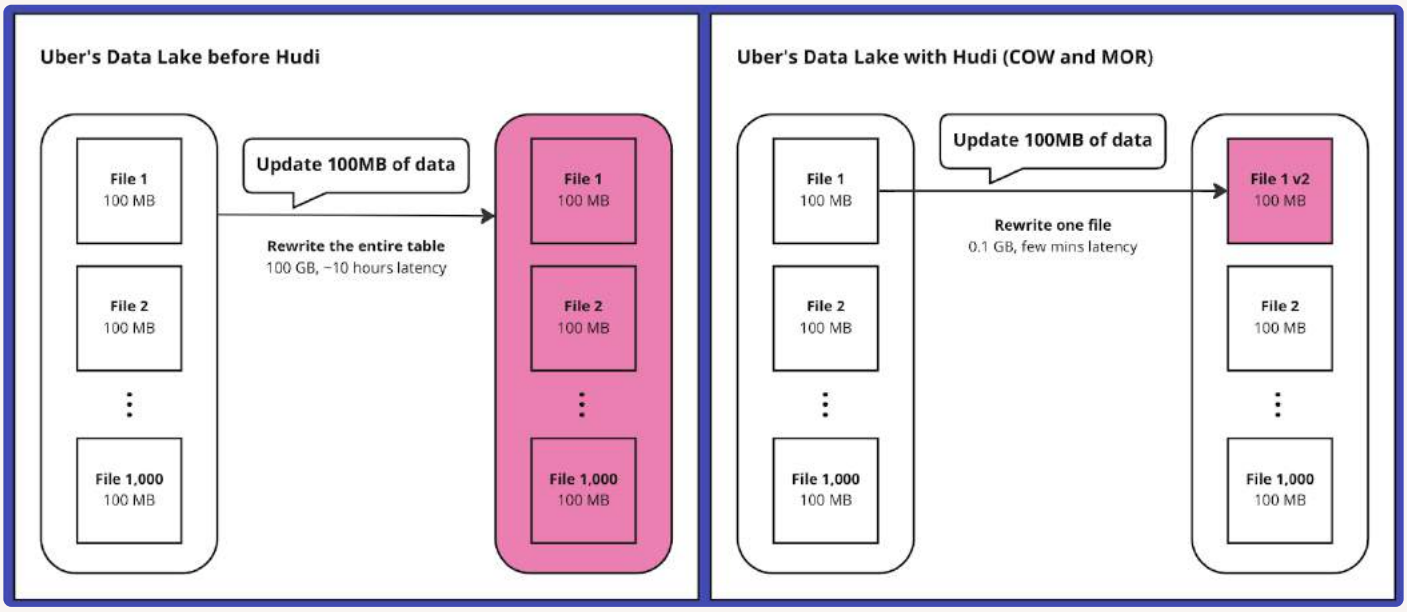
A common pitfall for ingestion is using a tool that charges by data volume (on top of your compute costs) to move your data. For example, Fivetran charges a fee per monthly active row on top of the compute costs that you pay to actually write the data. When you ingest more data with Fivetran, you increase your compute costs in your downstream system (warehouse or lakehouse) while also paying Fivetran for the additional rows. As an alternative, you can use open source data integration tools like Airbyte and Kafka Connect or storage systems with built-in data ingestion to save on the additional costs. We offer this at Onehouse (see example), and you can find other offerings on the market like Redshift's Zero-ETL.

Another driver of cost inflation is using a system that can't process data incrementally. Old-school data lakes on parquet files only support batch processing which requires rewriting entire files, partitions, or tables when new data arrives. This is extremely inefficient, leading to expensive and slow data ingestion. Instead, you should use a system that processes data incrementally, ensuring you only write data that has changed and avoiding costly write amplification (ie. writing more data than necessary).

Uber, for example, saved ~80% in compute costs by processing trip data incrementally every 15 minutes, moving away from their antiquated batch system:



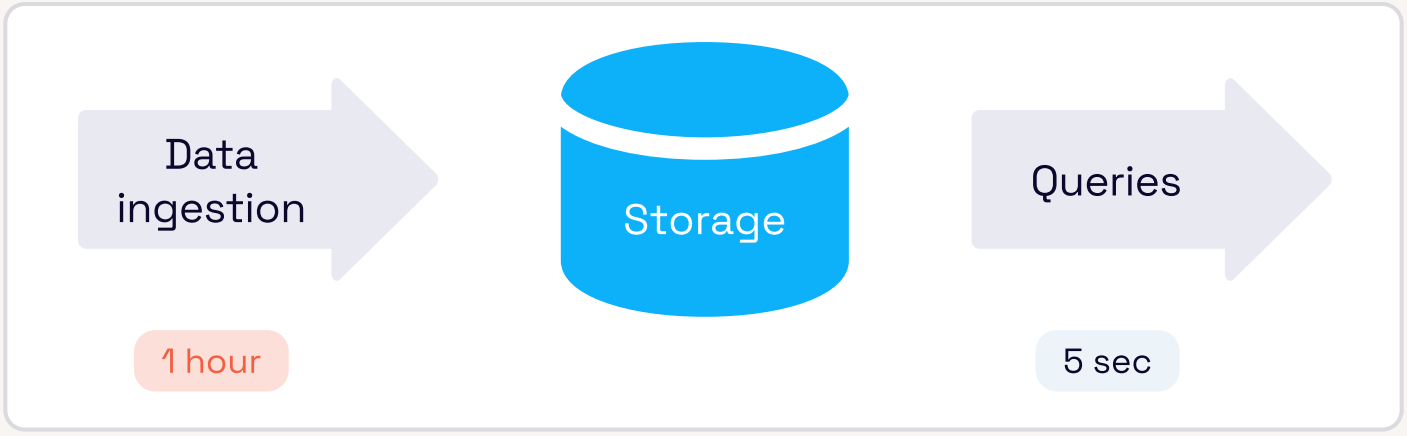
Uber leveraged Apache Hudi's Merge on Read (MoR) tables, which are especially efficient for update-heavy workloads by virtue of writing data incrementally at the file level, rewriting only the files that have changed:



When selecting your storage layer for analytics, ensure that it can write data incrementally in an efficient manner for your current and future workloads. The best way to evaluate these systems is to run benchmarks with realistic workloads and monitor the total costs for data ingestion.

Overcome Data Freshness Challenges

Modern data projects require fresh data to serve use cases that run in near real-time. However, you can only read data as fast as the slowest point in your pipeline. Data ingestion is a critical point in the pipeline, as large volumes of data moving from source systems into analytics systems can cause bottlenecks. If the data in your lakehouse is delayed by one hour due to slow ingestion, your "real-time" machine learning applications will be delayed by at least an hour as well.



Data teams spend significant time and energy (and often money) on optimizing query speed. This effort is important but does not speed up the end-to-end pipeline if data ingestion is a bottleneck.

Similar to the challenge with costs, failing to process data incrementally is a common ingestion latency bottleneck. By processing data incrementally, you can write faster and more efficiently to your storage system. In the same Uber example, the data team was able to decrease the pipeline run time by 50% and also decrease the SLA by 60% with Hudi's incremental writes.

As your data volumes grow, data freshness can be seriously impacted especially for update-heavy workloads such as change data capture (CDC). When incoming records are used to update an existing table, the table has to be scanned for each incoming record to determine if a row needs to be updated or simply appended to the table. Scanning the table can be a costly and time-intensive activity that scales linearly with the growth of your table over time.

Apache Hudi provides a unique approach to breaking this performance bottleneck with a new feature called the record-level index (RLI). RLI keeps and builds an index with unique keys. When a batch of incoming records needs to be processed, the RLI allows efficient point lookups versus full table scans. Comprehensive benchmarks have shown >70% performance improvements when employing RLI vs a traditional global simple index.

Openness and Universal Interoperability for Your Data

It's enticing to use managed platforms for data integration, which are often closed source. For example, many organizations use Fivetran (a black box, closed source integration tool) to ingest data into Snowflake (which uses a proprietary table format). This can be costly and invites challenges down the road as you become locked into these proprietary vendor ecosystems.

In a proprietary ecosystem, interoperability is controlled completely by the vendor – you're only free to use only the tools that play well with their products. Migrations are a challenge as well. When you find, and want to move to, other tools that better suit your organization's growing needs, you must rebuild your ingestion infrastructure from the ground up.

If you were designing a car, you would be wise not to use parts that are dependent on a single supplier, who may go out of business or dramatically raise prices. Instead, you would want to use parts that are standard across the industry and can be seamlessly replaced by other manufacturers as needed. Similarly, organizations should build their end-to-end data infrastructure on open source technologies instead of locking data into proprietary systems.

Here are some examples of open and proprietary systems for data movement and storage:

	Open	Proprietary
Data Movement	• Apache Kafka • Apache Pulsar • Airbyte	• Fivetran • Hevo Data • Matillion
Data Storage	• Apache Hudi • Apache Iceberg • Delta Lake (Linux Foundation) • Apache Parquet	• Snowflake • BigQuery • Redshift • Azure Synapse

The universal data lakehouse architecture involves ingesting your data directly into an open source format such as Apache Hudi, Apache Iceberg, or Delta Lake. Data should stay in open source systems along the way by using event stream technologies like Apache Kafka or Apache Pulsar. This approach ensures you don't have a single point of vendor lock-in for your critical data ingestion pipelines.

Using open source data lakehouse formats also ensures that you maintain interoperability to query the data in downstream compute engines including Spark, Flink, Databricks, Snowflake, Redshift, Trino, Presto, Dask, and many more.

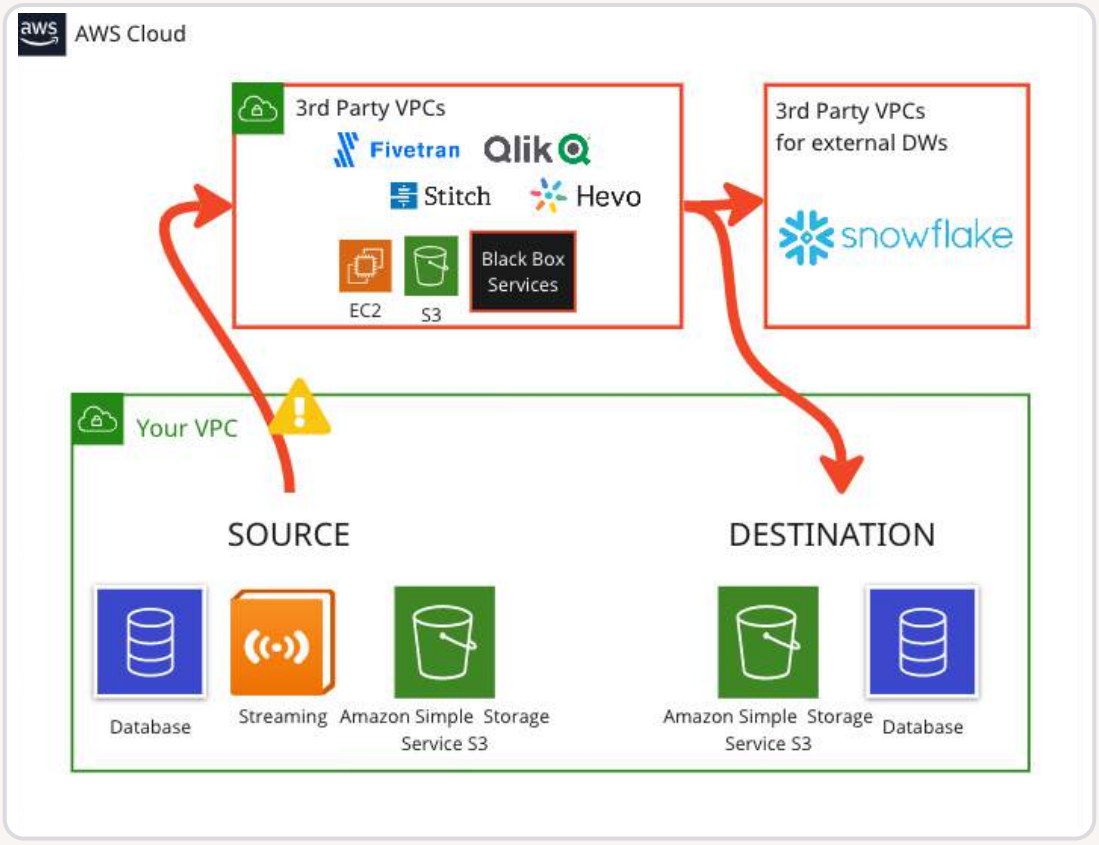
Furthermore, you can reap the interoperability benefits of all three data lakehouse table formats (Hudi, Iceberg, and Delta) by using OneTable, an open source project supported by Onehouse, Microsoft, Google, and many others. OneTable makes it possible to maintain a single copy of your data, along with metadata from all lakehouse table formats, to ensure complete interoperability across compute engines.

Ensure Data Privacy

Security, privacy, and compliance are vital characteristics of any data architecture. Your architecture should be well-designed to prevent unauthorized access, prevent data exfiltration, and ensure compliance with complex and fast-changing privacy regulations. Introducing third-party vendors who take control of your data into their environments can increase the scope of your infosec boundary and require you to put a large degree of trust in the hands of suppliers.

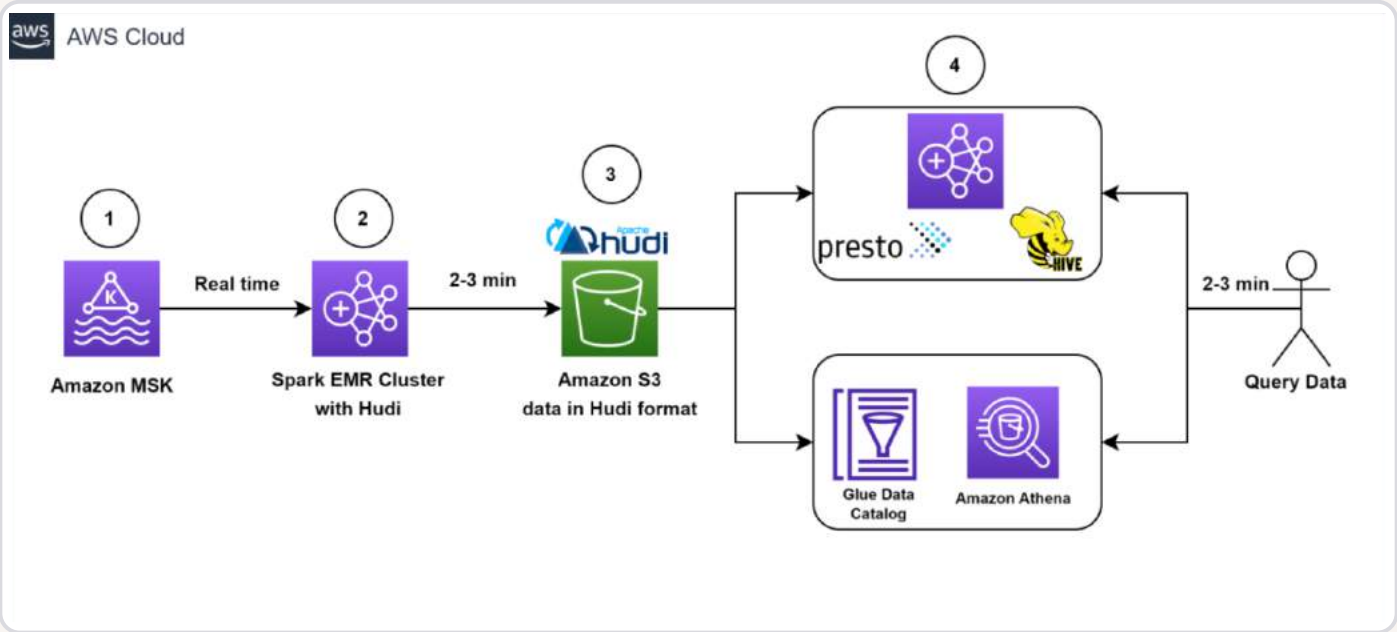
The tools you choose to use for data ingestion are a crucial part of your infrastructure, since ingestion will touch all of your data and set the stage for how the data is stored. In the vendor market today, a majority of data ingestion / integration tools require you to export your data out of your cloud accounts and through their networks.

In such cases, data exported out of your virtual private cloud (VPC) is processed and stored in external third-party systems, becoming susceptible to vulnerabilities. With recent security events that occurred at places like Datadog suffering a breach from third-party tool CircleCI, Atlassian suffering a breach from third-party tool Envoy, and any company becoming vulnerable who was depending on third-party tool LastPass. It is increasingly hard to justify why data should leave your security boundaries.



You can ensure data privacy by keeping data within your own VPC throughout ingestion and storage. You can run open source ingestion tools like Airbyte or Kafka Connect directly in your VPC with compute services like EMR for AWS or DataProc for Google Cloud.

In this example, Zoom leveraged open source tools like Kafka, Spark, and Hudi to ingest data into the data lakehouse without exposing the data outside of their AWS private cloud.



When evaluating services for data integration or storage, it's critical to ask if the data ever leaves your account - and if so, why - so that you can minimize exfiltration risks.

Best practices for data preparation

Data preparation is the process of transforming raw data into cleaned and filtered tables that are ready to be consumed for analytics. It's essentially the T step in ELT or ETL. Data preparation serves as a cornerstone of your data architecture by ensuring that data is high quality and easily accessible to power all of your organization's analytics use cases, from reporting to business intelligence to data science to machine learning.

As an organization grows, the breadth of data becomes more complex and the teams producing that data become more distributed. New data is produced in different formats and schemas evolve, breaking data pipelines that were previously working. Data infrastructure that fails to reliably prepare data can take down business-critical analytics systems and lead to middle-of-the-night, on-call firefighting by data teams.

On the other side, downstream data consumers are constantly pursuing new use cases for the data, which often require new data models. These users - data analysts, scientists, and engineers - work across disjoint systems such as data warehouses, data science notebooks on ML-optimized compute engines, and open source data engineering engines. When the source data is not properly prepared early in the pipeline, these stakeholders use their own transformations to create new copies of the data. This leads to multiple copies of the data that are poorly maintained and duplication of work due to a lack of data discoverability across the organization.

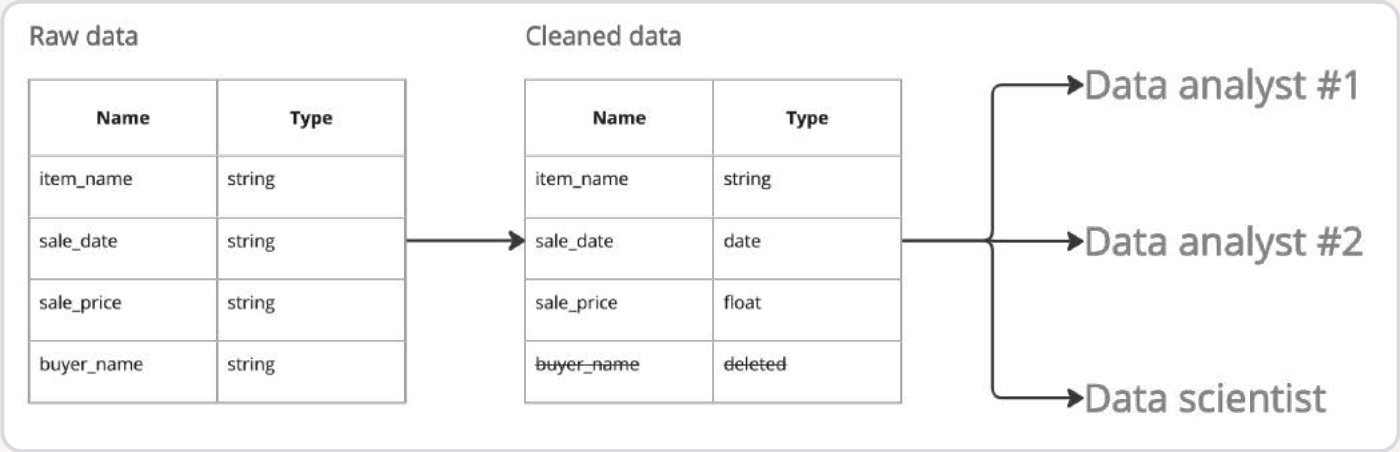
These challenges can be solved by simply preparing data earlier in the lifecycle, using the universal data lakehouse architecture. Let's dive deeper into how to solve these challenges around data preparation:

Ensure Data Quality

Data quality has become an increasingly important and popular topic, evidenced by communities like Data Quality Camp, which racked up nearly 10,000 members in its first year of existence. The rise of machine learning has also made data quality more important, as models are only as good as the data powering them. The interest in data quality is driven by a combination of the importance and the challenging nature of ensuring high-quality data.

Failing to centralize data quality checks and transformations is one of the biggest drivers of data quality issues. In the example below, we see what happens when data consumers each transform their own copy of the raw data:

In this example, analyst #1 casts columns to their proper types and drops the buyer_name column containing sensitive personally identifiable information (PII). The data scientist performs the same casting transformations but fails to drop the buyer_name column, leading to potential privacy issues down the road. Lastly, analyst #2 creates another copy of the table and forgets to cast the columns, which means that analyst #1's queries will not work on analyst #2's table. Instead of relying on each data consumer to apply their own transformations, these transformations should be applied on a shared copy of the data that can serve as the source of truth:



This model ensures that every data consumer is working on high-quality data, free from privacy risks. Furthermore, the data analysts and scientists can now focus on their highest-value work rather than spending time on data quality tasks that could be handled upstream.

The universal data lakehouse architecture makes this data quality model possible by allowing you to use a single copy of data as the source of truth for the organization. Rather than maintaining separate data copies and data quality rules in disjoint systems like a warehouse and data lake, you can consolidate the data in the data lakehouse and use any needed compute engine for reading the data or deriving aggregate tables for specific use cases.

Another best practice for improving data quality is to filter out bad data in flight before it reaches your storage systems. This saves on the overhead of maintaining bad data and ensures that the bad data is not accidentally consumed by others in the organization. You can achieve this by using a dead-letter queue to quarantine data that fails to meet your data quality requirements – for example, data with an invalid schema or record values outside of an expected range.

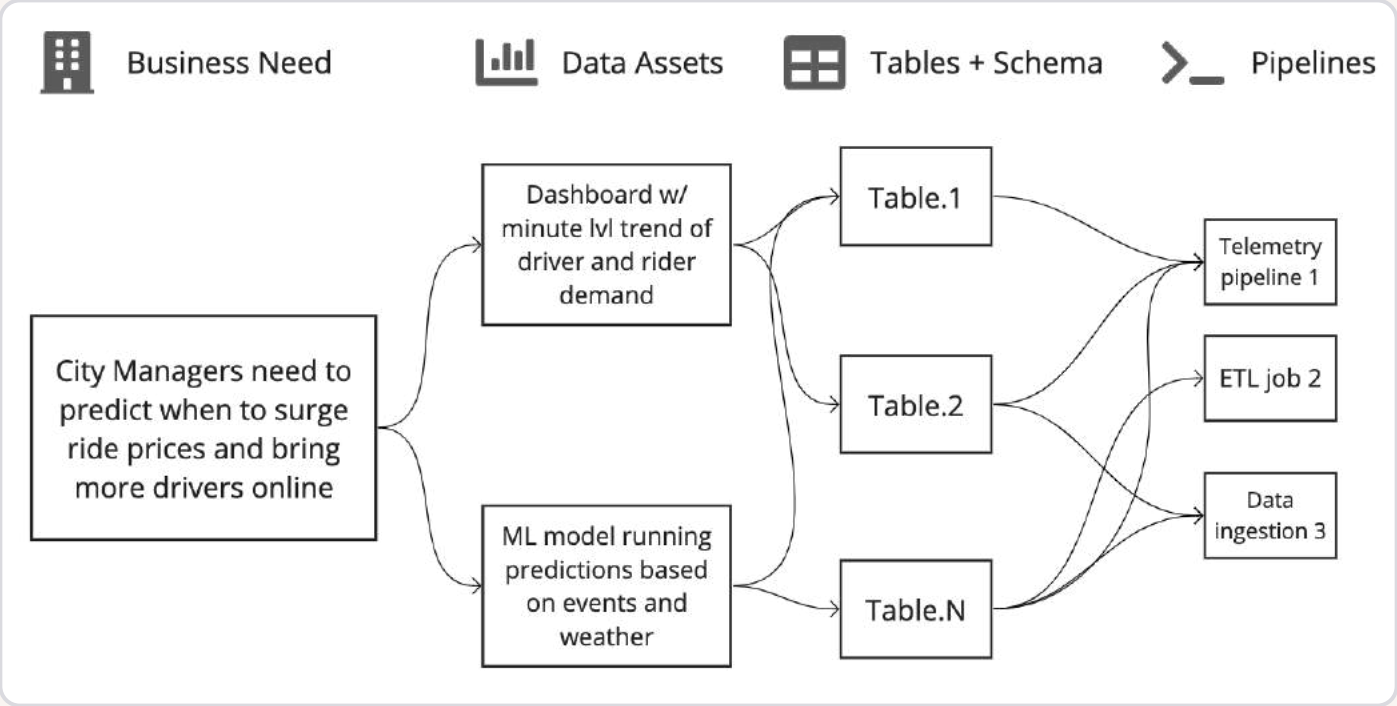
Lastly, you can leverage open source data quality tools like Great Expectations and Deequ to more easily apply data quality validations. Combining this data quality tooling with the universal data lakehouse infrastructure is a powerful way to ensure that your organization never has to operate with bad data.

Model the Data

Data Modeling is one of the most important data preparation activities you need to consider early in the lifecycle of building out your data platform. Some aspects of data modeling will require you to work cross-functionally within your organization, beyond the core data engineering team. Your first objective is to align relevant stakeholders in your business to understand what questions need to be answered with data and which decisions need to be influenced by data. It is important to build a workback plan from there to ensure your physical data model will make an impact on your business. That work back plan to your data model will include:

1. Which data assets you need to create, such as dashboards or machine learning models
2. What specific data is required to generate those assets
3. How you will collect, organize, and serve the data to those assets

For example, if you are running a ride-sharing business, you may have a data modeling exercise that generates the following work back plan:

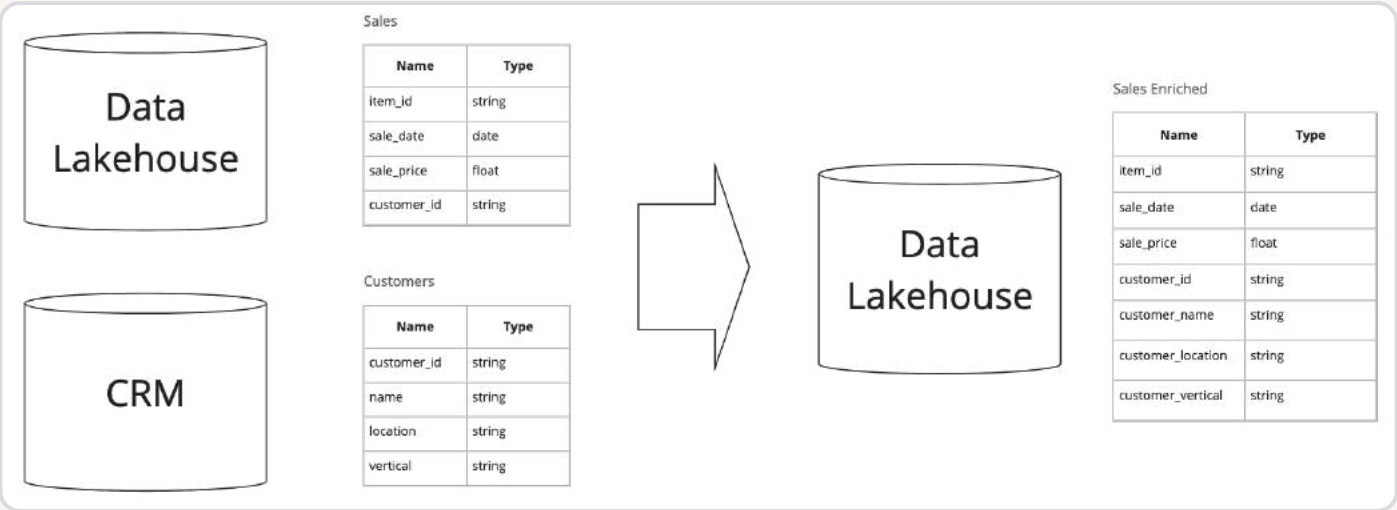


Enrich the Data

Data enrichment is the process of enhancing a dataset by merging and adding either first-party or third-party data. Enriching your data enables data consumers to answer more questions and gain deeper insights, helping them achieve business goals.

Let's take the example of an organization tracking product sales. The organization ingests sales data from an event stream into a table in their analytics system. They maintain information about their customers separately in a CRM accessible by the GTM team. Now, imagine that a data science team wants more demographic info about customers in order to predict the potential return on ad spend (ROAS) for an upcoming marketing campaign. The data science team cannot glean these insights while customer data is siloed away in the CRM.

To empower their data science team, the organization can use their CRM data to enrich their source-of-truth data in the data lakehouse:



In this example, the valuable CRM data is now democratized across the organization, and access can be managed in one central layer. We show a denormalized data model above, but the organization could reap similar benefits by replicating the CRM data as a table in the lakehouse and allowing data consumers to join the data themselves.

As shown in the example above, data enrichment is most effective when done in the early stages of the pipeline. This ensures that all downstream tables are automatically updated with the enriched data. Nerdwallet is a company that follows this same principle for their data infrastructure by enriching data during ingestion into their data lakehouse.

One last consideration when performing data enrichment is to make sure that the data is regularly updated. Data quality will suffer if the enriched data is out-of-date. When enriching with data from outside of your analytics system, be sure to set up jobs that regularly ingest fresh data. Open source data integration tools like Airbyte and Kafka Connect can be useful for connecting to external systems like a CRM.

Optimize Data Layouts

As you ingest data and form the foundation of your data platform, there are several considerations to keep in mind for how to efficiently organize the data layout on storage for performance and cost savings. Without planning ahead, the performance of your pipelines and queries can quickly decay over time. Examples of data layout optimizations include partitioning, indexing, clustering, file-sizing, and cleaning. These layout optimization techniques are usually well supported in databases or data warehouses but are non-existent in cloud object storage systems such as Amazon S3, Azure Data Lake Storage, and Google Cloud Storage.

Apache Hudi bridges this gap and brings critical database-like storage layout optimizations to the data lakehouse. With Hudi it is easy to choose a partitioning strategy. Hudi offers a first-of-its-kind pluggable indexing subsystem for the lake, allowing you to customize depending on your workload. By default, Hudi will automatically manage file-sizing operations, and cleanup of old versions and metadata to reclaim precious storage. Hudi offers advanced sorting techniques that allow you to apply multidimensional clustering algorithms such as Z-Order or Hilbert curves.

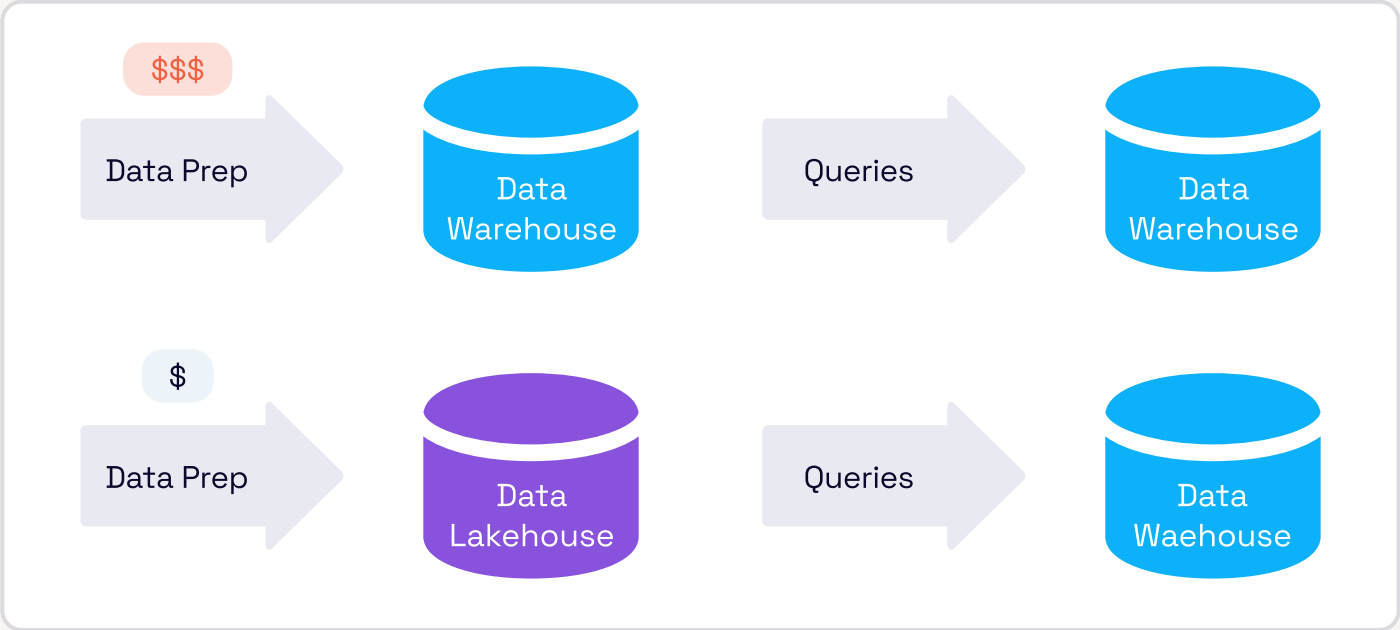
With these data layout optimization tools, you can achieve performance improvements on the order of 100x, which will lead directly to cost savings for your organization.

Amortize the Cost and Effort of Data Preparation

Too often, the data preparation work described in this whitepaper is repeated at multiple points in the pipeline, leading to a higher total cost of ownership (TCO) for the data system.

The universal data lakehouse presents an opportunity to cut down duplicate data engineering efforts and reduce compute costs by amortizing the data preparation work. With this architecture, organizations can centralize their data preparation in the bronze and silver layers of a unified data pipeline. Data engineers should be responsible for transforming data into consumable data products with owners who are responsible for the quality and reliability.

This approach helps your team save on compute costs by processing transformations, validations, etc. on the data just one time, early on in the pipeline. Furthermore, disjoint teams don't need to waste time performing the same transformations on separate copies of the data – the transformed data they need is already available in a centralized data lakehouse. These transformations are performed on the lakehouse, enabling you to use cheap bare-metal compute instead of processing the data on expensive premium compute engines like a data warehouse:



This example shows how you can run data preparation with low-cost commodity compute on the data lakehouse instead of unnecessarily using premium data warehouse compute – all while continuing to serve queries with the data warehouse. Organizations like Apna have already adopted this model to achieve significant cost savings.

When performing data preparation, you can leverage tools like dbt or Airflow that make it easy to manage and reuse transformation code for repetitive tasks like parsing JSON, exploding arrays, and processing change data capture (CDC) events. These UI-based tools also make it accessible for anyone in the data organization to transform and validate data within the data lakehouse without having to learn data engineering technologies like Spark or Flink.

Lastly, you can achieve significant cost savings for data preparation by processing data incrementally. In the data integration section, we covered the benefits of using incremental processing for ingestion. These cost savings and speed enhancements compound when you use incremental processing through all stages in your pipeline, ensuring that all raw, cleaned, and aggregated tables process only the data that has changed.

Choosing the right compute engines for your use cases

Storing your data in a universally accessible format and storage layer unlocks the ability to mix and match various compute engines while operating on a single, high-quality copy of the data. This engine flexibility is a powerful capability, as you can now leverage the unique strengths of multiple purpose-built engines rather than fitting incorrectly and paying additional cost/performance costs. Furthermore, new engines are always emerging; locking yourself into one vendor ecosystem means you will need to either migrate or miss out on future innovations.

At a high level, choosing the right compute engine unlocks the following benefits:

- Cost savings, by choosing the most efficient engine for the workload
- Deliver the desired query performance for certain workloads
- Access to specialized compute engines and libraries (such as machine learning-oriented engines)
- Decoupled compute from ingestion and data prep commodity workloads; no competing for resources or debating the right platform

Mixing and matching compute engines has become standard practice in the industry. For example, data from ETR shows that about 40% of Snowflake customers also use Databricks, and about 46% of Databricks customers also use Snowflake. By moving to the universal data lakehouse architecture for data ingestion and data preparation, these thousands of companies can be saving millions of dollars by not operating on disjoint or copied data between both systems.

When organizations first build their data platform, they don't intend to end up with multiple disparate storage layers. However, as new use cases emerge, such as machine learning, data sharing, or real-time queries, organizations are forced to adopt new storage layers because they initially stored data in a system that is not interoperable with the ecosystem of compute engines. By starting with a universal data lakehouse architecture from the beginning, organizations can avoid the future headaches of migrating data, copying pipelines, or managing data across multiple disjoint systems.

The next sections provide a broad (but non-comprehensive) overview of compute engines and their strengths and weaknesses, alongside examples of how they are used by actual organizations. We suggest that you use this as a starting point for your compute engine research; however, you will also benefit from running your own analysis and benchmarks. The compute engine ecosystem is rapidly evolving, so you may find that functionality and performance change dramatically as these engines improve and new engines emerge. Once again, this only emphasizes the importance of storing data in an architecture that provides flexibility to add or change compute engines over time.

Analytics & Reporting Use Cases

Analytics and reporting involve the collection, processing, and interpretation of data to generate insights for decision-making, often referred to as business intelligence (BI). This function leverages real-time and historical data to produce dashboards, charts, and reports that help stakeholders understand key business metrics. Analytics and reporting functions are typically performed by data analysts and business intelligence professionals.

When choosing a compute engine for analytics and reporting, the following factors are most important:

- **Query speed**: Users should be able to get query results fast, allowing them to interactively explore the data.
- **Concurrency**: Many users in your organization may be performing analytics at the same time. A good engine supports a medium to high level of concurrent queries, and ways to prioritize these queries across users/business functions.
- **Cost**: The engine should be able to execute complex queries quickly, by employing state-of-the-art techniques such as query planning and optimization, to cut down the amount of data scanned.

For analytics and reporting use cases, cloud data warehouses and federated SQL Engines (such as Trino or Presto) on data lakes are often a good fit. Each option has its own benefits and tradeoffs.

Cloud Data Warehouses (Snowflake, Redshift, BigQuery, Azure Synapse)	Federated SQL Engines (Presto, Trino, Dremio)
Benefits	
✓ **Query speed:** Data warehouses serve queries fast, typically due to more advanced query optimizers and planning techniques like statistics maintenance, result-set caching, and materialized views. ✓ **Data management:** Data warehouses are built like complete database systems, with advanced data management functionality such as a built-in catalog and data encryption. ✓ **Ease of use:** Data warehouses often come with user-friendly interfaces and are easy to set up with features like query optimizations, access control, and autoscaling available out-of-the-box.	✓ **Scalability:** High horizontal scalability, especially when crunching very high data volumes by scaling to 100s or even 1000s of machines reading directly from cloud storage. ✓ **Cost:** These engines are typically less expensive than data warehouses on a dollar-per-core basis for similar queries. (Although engines relying on memory as a means to accelerate queries could cost more.) ✓ **Query federation:** These engines allow you to query data from multiple sources such as RDBMS, NoSQL databases, and data lakes, enabling a more flexible and extensible architecture.
Trade-offs	
✗ **Cost:** Data warehouses are typically more expensive than federated SQL Engines. Data warehousing costs grow very quickly with larger datasets, near real-time ELT workloads, and complex ETL jobs. ✗ **Vendor lock-in:** Data warehouses are typically closed-source systems employing proprietary formats, making them unsuitable for source-of-truth data storage due to inherent lock-in. ✗ **Scaling:** Although most cloud warehouses have storage/compute separation, some still have local storage architecture that can limit the amount of data storage available.	✗ **Complexity:** Although highly flexible, these engines may require fine-tuning and expertise in data management for optimal performance. ✗ **Budding ETL support:** While they are excellent query engines for analytics, they are yet to be widely used for creating data pipelines that build reports.
Case studies	
• FedEx uses Azure Synapse data warehouse for everything from customer payment segmentation to freight and shipping anomaly detection. • Faire reported a significant speedup from using Snowflake; Home Depot saw similar experiences with BigQuery, while Redshift's architecture promises to deliver superior performance.	• Twilio serves over 500M queries per day on PrestoDB. • TikTok/ByteDance operates an exabyte-scale data lake on Presto and Hudi. • Trino powers analytics at large companies like Netflix and Linkedin.

Data Engineering Use Cases

Data Engineering is the practice of constructing, managing, and optimizing data pipelines, including ETL (Extract, Transform, Load) processes. Engineers design and implement the architecture needed to collect, clean, transform, and store data so it can be readily used by consumers such as data scientists, analysts, and machine learning models.

When choosing a compute engine for data engineering, the following factors are most important:

- **Cost-Performance**: You should choose an engine that balances your job execution speed and the cost budget as data volumes and complexity of workloads increase.
- **Fault tolerance**: Fault tolerance is critical for ensuring that data is not lost or corrupted during the ETL process failures and ensuring that data pipelines deliver data reliably, without needing manual intervention required when a job fails.
- **Pipeline management**: Data engineers work with many use cases, and the use cases are constantly expanding. It's important to choose a versatile engine that boasts a rich ecosystem of workflow orchestration and tools such as dbt or Airflow.

There are two broad categories of systems employed for authoring data pipelines, and the table below presents a comparison of the approaches in broad strokes.

Cloud Data Warehouses (Snowflake, Redshift, BigQuery, Azure Synapse)	**Distributed Processing Engines** (Spark, Flink, Hive)
Benefits	
✓ **Simplicity:** Running pipelines on the data warehouse itself keeps the data architecture simpler since there are fewer components in use. ✓ **Data management:** The same automatic data management features can be used to manage the tables, reducing time to market.	✓ **Open:** All major distributed processing engines are open source and primarily optimized on open data formats, which makes it easy for pipelines to write data once and read many times from different engines. ✓ **Cost Effective:** These engines directly scan data from infinitely scalable cloud storage from 1000s of nodes to support all shapes and sizes of pipelines in one framework, from analytical reports to complex ML model training. ✓ **Resilient:** All engines offer high levels of resiliency to intermittent failures, handling data skews with graceful degradation and ensuring that your data pipelines are operating round-the-clock, feeding data to your teams. ✓ **Best-in-class lakehouse support:** These engines have tight integrations with radical new data lakehouse technologies to offer new capabilities. For example, Apache Hudi combined with Spark or Flink brings advanced indexing and incremental queries that can speed up pipelines 100x or more.

Cloud Data Warehouses (Snowflake, Redshift, BigQuery, Azure Synapse)	Distributed Processing Engines (Spark, Flink, Hive)
Benefits	
	✓ **Batch and Streaming Models:** Most of these engines make it easy to switch from running scheduled batch jobs to a more streaming/micro-batch architecture with just a few lines of code. ✓ **SQL + Code:** These engines offer powerful DataFrame APIs, in addition to customizable SQL optimizers across popular programming languages.
Trade-offs	
✗ **Cost:** Data warehouses are primarily designed for low-latency query execution by scanning less and concurrency management. ETL workload patterns are very different from analytical workloads and costs can increase dramatically for large ETLs where the problem is more about scanning/crunching data as fast as possible. Problems are exacerbated for mutable workloads with MERGE statements, given that warehouses have been typically built with static data in mind. The premium you pay for the warehouse compute is often more expensive than an engine that is purpose-built for ETL. ✗ **Pipelines as Code:** Data warehouses largely rely on SQL for everything and a good chunk of data engineers author pipelines in code with local test data to ensure data quality and maintainability. ✗ **Interoperability:** Given proprietary data formats are still the primary choice in warehouses, the data produced by these pipelines cannot easily be exposed to other use cases such as data science, so duplicate pipelines are required, doubling the infrastructure and maintenance costs of the pipelines. Data warehouses are starting to improve support for external open table formats like Hudi, Iceberg, and Delta Lake.	✗ **Management Overhead:** While they bring great flexibility, these engines need to be paired with data management functionality either in open source (e.g. Hudi) or a managed service (e.g. Onehouse or Databricks) to optimize and track the output from underlying tables to these pipelines. ✗ **Learning Curve:** You would need to put together a talented team of data engineers who already have familiarity, or can learn these frameworks quickly, to fully realize the benefits above. This is another area where managed data lake products like Onehouse or EMR help.

Cloud Data Warehouses (Snowflake, Redshift, BigQuery, Azure Synapse)	Distributed Processing Engines (Spark, Flink, Hive)
Case studies	
• Instacart spent $56M in total on Snowflake over 2021 and 2022, representing 6.3% of Instacart's total R&D expenses over those years. • Ramp reports advantages from automated maintenance of the underlying tables with Snowflake.	• We have seen a large SaaS startup reduce their ELT costs dramatically by moving their compute workloads from a data warehouse to Spark or Flink with a Hudi data lakehouse. • Walmart operates data pipelines at Fortune 1 scale using Spark and Hudi. • Uber and Zoom have reported impressive savings from adopting incremental data processing to reduce their ETL costs.

Data Science, Machine Learning, and Artificial Intelligence Use Cases

Data science (DS) focuses on drawing insights and making predictions from data. Data scientists use various statistical, mathematical, and programming techniques to analyze data and solve complex problems, usually in a way that can influence strategic or operational decisions.

Machine learning (ML) and artificial intelligence (AI) involve building and training algorithms to learn from data and make decisions or predictions. In a data organization, ML and AI can automate data analysis, improve analytics models, or drive various kinds of automated decision-making processes. Deep learning requires the management and processing of large volumes of images, videos, text, and other unstructured data, in addition to structured data that data warehouses and data lakehouses support. PyTorch and TensorFlow are among the most popular deep learning frameworks.

Examples of data science and AI/ML projects include fraud detection to alert security teams, predictive maintenance in manufacturing to minimize downtime, and recommendation systems to personalize user experiences. Most recently, the AI space has garnered unprecedented amounts of attention due to the viability of LLMs in commercial and everyday applications such as ChatGPT.

Some common compute engines employed for DS/AI/ML are Apache Spark, and distributed compute engines like Dask and Ray that parallelize Python code and libraries.

This ecosystem is still growing and evolving, and has a high chance of short-term disruption with many different types of frameworks emerging that all specialize in their own workloads. With compute engine capabilities rapidly expanding in this area, practitioners typically pick the data formats first and then add their compute engines of choice. This emphasizes the point that storing your data in open and interoperable systems is critical in order to keep up with the DS/ML/AI landscape, which will rapidly evolve in the next few years.

When choosing a compute engine for data science, ML, and AI, the following factors are most important:

- **Framework support:** Choosing from popular DS/ML/AI frameworks like Pandas, Polars, TensorFlow, PyTorch, and Scikit-Learn makes it easier for ML/AI teams to build with their familiar tools.
- **Scalability requirements:** You should understand your scalability requirements when choosing a compute framework for DS/ML/AI. A highly scalable engine may be necessary when working

with large volumes of data, while in other cases it may be overkill. For example, if you are training ML models, the engine's ability to scale horizontally across multiple nodes and support GPU acceleration is vital.

- **Processing Speed**: Real-time prediction use cases such as fraud detection require low-latency query response. If you plan to build real-time DS/ML/AI applications, the engine should process and infer data in real-time. Additionally, faster processing can help data scientists improve their efficiency by delivering quick feedback loops.

Case studies:

- Capital One used Dask to reduce ML model training times by 91% with a few months of development effort
- OpenAI uses Ray to train their largest models (including ChatGPT), enabling them iterate at scale much faster than they could before
- Shopify replaced Spark with Ray in their ML platform to leverage specialized Python libraries and parallelize their ML training workflows
- Instacart uses Ray alongside Snowflake and Databricks to handle diverse requirements for MLOps

Stream Processing Use Cases

Stream Processing often refers to real-time computation on data directly as it is produced or received. It is typically used to filter, aggregate, or enrich incoming data streams before sending them for downstream analysis or initiating some action based on the data. Stream processing offers a great alternative to batch processing, by delivering real-time (a few seconds) data freshness and a completely incremental processing model that avoids recomputing results to achieve real-time performance. Stream processing systems are also great at dealing with unique problems like out-of-order and late-arriving data, treating them as first-class constructs in the processing model.

Stream processing is often used to denormalize data from upstream databases with normalized schema, to aid downstream use cases like warehousing, data science, and real-time analytics, or even updating search indexes. Other key use cases for stream processing are fraud and anomaly detection, where incoming events can be used to detect spurious patterns and fight spam or suspicious user activity. In many cases, stream processing can be employed to simply pre-process system logs, IoT sensor data, and other high-volume data streams before they are stored in a data lake. For example, Pinterest streams about 25 GB/s of data using Apache Kafka.

A stream processing system typically consists of streaming data storage (e.g. Apache Kafka, Apache Pulsar) and a compute engine for stream processing (e.g. Apache Flink, Spark Streaming, Kafka Streams).

When choosing a compute engine for stream processing, the following factors are most important:

- **Query speed**: When dealing with streaming data, speed is paramount. Choose a compute engine with fast end-to-end processing speed to ensure that the engine does not bottleneck your pipelines.
- **State management**: Stream processing systems manage intermediate results in their state stores, which lets them compute results incrementally as new data arrives. The fault tolerance and scalability of the state management play an important role in operating stream processing pipelines at the desired SLAs.

Stream processing routinely needs data from longer-term storage, such as a data lake or data warehouse, to bootstrap the historical state. For example, a job that counts the number of Uber trips per city needs historical data bootstrapped first so it can apply the incoming events correctly on top and produce accurate counts. Thus, using frameworks like Flink or Spark with open data lakehouse systems offering built-in support for streaming data, such as Hudi, is very important to ensure streaming pipelines are easy to build, modify, and operate. Finally, while stream processing unlocks exciting opportunities such as real-time dashboards and applications, it typically comes at a higher cost than batch processing due to always-on containers. Emerging techniques such as incremental data processing offer the ability to bring stream processing benefits in a data lakehouse with on-demand compute.

Case studies:

- Alibaba used Apache Flink to build their real-time search infrastructure due to Flink's agility, consistency, low latency, and cost-efficiency.
- Zalando found that Flink consistently offered lower latency at higher throughputs than Spark for their near real-time business intelligence. Flink's rich programming model also made it easier for them to implement complex semantics and handle out-of-order events.

Real-time Analytics Use Cases

Real-time analytics is the process of analyzing data as it is created, often within milliseconds to seconds after the data is generated. Real-time analytics systems are typically fed by upstream stream processing pipelines. Put together, this allows organizations to respond to emerging trends or issues immediately. Real-time analytics are important for use cases where speed is critical, such as for fraud detection, real-time IoT device monitoring, or live inventory management.

Real-time analytics engines can also serve as a speed layer on top of your data lakehouse. You can incrementally ingest and prepare data in the lakehouse, then use the real-time engine to process data that require sub-second queries for data consumers. Within your organization, real-time analytics might be consumed by data analysts, data scientists, or even by production applications.

Real-time analytics have skyrocketed in popularity in recent years, with many new compute engines emerging. Popular compute engines for real-time analytics include open source engines (eg. Apache Druid, Apache Pinot, ClickHouse, and StarRocks), proprietary engines (eg. Rockset), and specialized engines (eg. InfluxDB for time series data and Elasticsearch for search use cases).

When choosing a compute engine for real-time analytics, the following factors are most important:

- **Query speed**: The engine should be capable of executing queries with sub-second speed for real-time analytics. Most real-time engines prefer a local storage architecture so as to overcome cloud storage/access bottlenecks to support these ultra-low latencies - much like operational databases.
- **Support for high query concurrency**: Real-time analytics may require an unbounded volume of queries per second, so the real-time engine should be able to parallelize many queries.
- **Purpose-specific functionality**: Depending on your real-time use case, you may benefit from a system that offers specialized features, such as a time-series database, search indexes, or specialized handling of high-dimensionality data.

Note that real-time analytics come at a cost – you'll generally pay more for these specialized engines than for the general-purpose analytics engines under the Analytics & Reporting section.

However, for certain low-latency, high-concurrency use cases, you can save on costs by choosing a real-time analytics engine over a general-purpose analytics engine. It's often best to mix and match engines for different use cases, depending on your requirements for data freshness, query speed, cost, and other factors. Since most real-time analytics engines couple compute with storage, the best practice is to copy only the cleaned and processed data that is required for real-time use cases from the data lakehouse to the real-time analytics engine.

Case studies:

- Stripe uses the combination of Kafka and Pinot to process petabyte-scale financial data, handling half a million queries per second
- Airbnb leverages StarRocks to query hundreds of billions of records within seconds
- Robinhood used InfluxDB to quickly and painlessly build a production-ready anomaly detection system

Conclusion

The universal data lakehouse represents a paradigm shift in data architecture, one that synthesizes the strengths of traditional data lakes and warehouses into a unified, highly efficient, and future-proof system. As this whitepaper has demonstrated, the universal data lakehouse architecture not only overcomes the limitations of its predecessors but also opens doors to incorporate new advances in technology as they emerge. By embracing this architecture, organizations can achieve transformational improvements to their data infrastructure, including streamlined data processing, improved data quality, reduced costs, and enhanced flexibility in choosing compute engines tailored to specific needs.

The case studies of leading organizations like Uber, Walmart, and TikTok underscore the practical advantages of the universal data lakehouse in handling diverse, large-scale data workloads with efficiency and agility. Furthermore, you can immediately start leveraging the tactical guidance provided on data integration, preparation, and compute engine selection to earn incremental wins for your data platform.

Adopting the universal data lakehouse architecture is not just a strategic move for current data management needs; it's an investment in an organization's future data capabilities. As we step into an era marked by exponential data growth and rapid technological advancements, the universal data lakehouse stands out as a resilient, adaptable, and robust foundation for harnessing the power of data to drive innovation, efficiency, and competitive advantage.

The team at Onehouse has several decades of combined experience across the various systems discussed in this paper, and we built the Onehouse platform to help organizations more easily adopt the universal data lakehouse architecture. If you are interested in learning how you can transform your organization's data infrastructure, we're happy to chat; reach out at gtm@onehouse.ai.