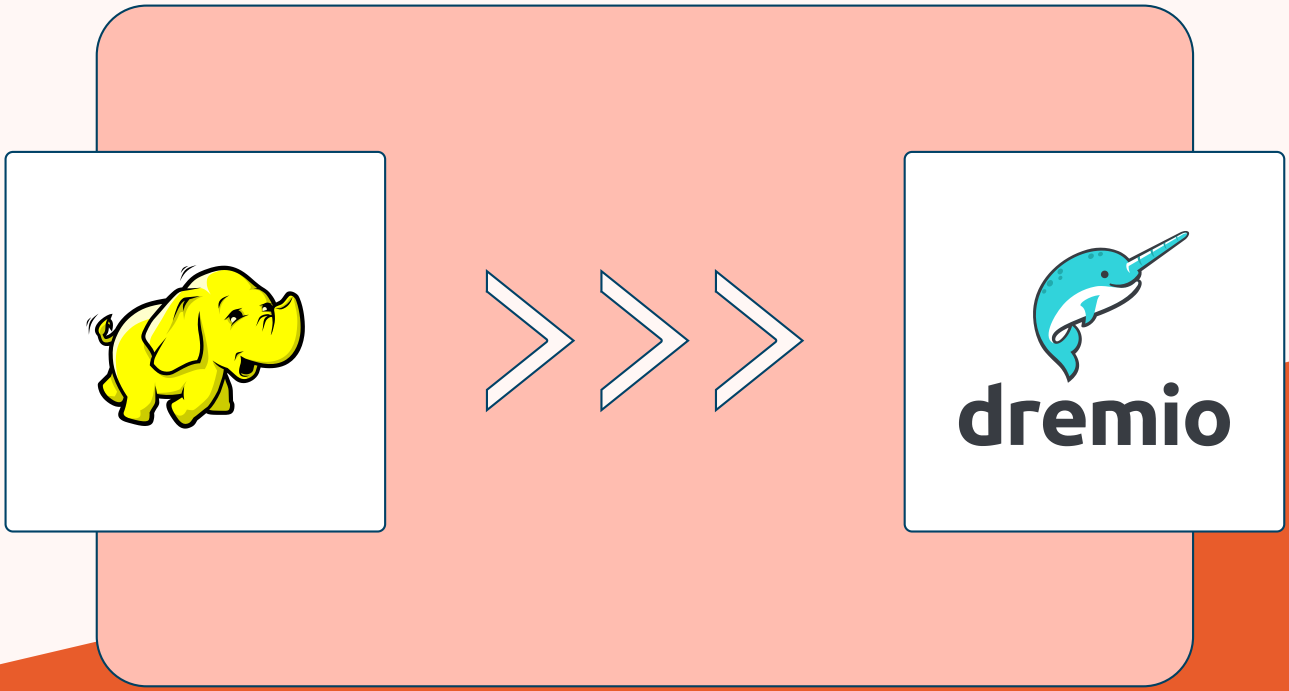
WHITEPAPER

From Hadoop to Data Lakehouse: A Migration Playbook

Three steps to transform Hadoop into an open architecture for self-service analytics

Author: Donald Farmer, Principal at TreeHive Strategy



Introduction

Hadoop, the foundation for big data analytics, was created in 2006 by engineers at Yahoo. Their aim: an open-source system to store and process large amounts of unstructured web data – something other systems weren't reliably or efficiently able to do. Their work was a genuine success; businesses have relied on Hadoop for several years when constructing their analytic infrastructures.

But the landscape of big data processing has changed, and new cloud-based solutions, being more scalable, more performant, and easier to manage, have become more popular for data storage. Additionally, many Hadoop distributions are now near the end-of-life: support and security updates are limited, and it is difficult to hire for the specialized skills required to manage Hadoop. Enterprises have either migrated Hadoop, are in the midst of a migration, or working on a migration project.

In this paper, we will look at more detailed reasons to migrate and we'll explore some of the options that may come to mind. In particular, we'll propose a unique, phased approach to migration with Dremio which de-risks an otherwise complex process while increasing performance and usability.

But first, let's be sure we understand why Hadoop has been so successful, so we don't lose its valuable features in the rush for something new.

Why Hadoop?

In its day, Hadoop was seen as a revolutionary development in data management. Indeed, to be fair and objective, there are some excellent features within the Hadoop framework.

- The distributed file system, HDFS, provides fault-tolerance by replicating data across multiple nodes. This helps to ensure that data is highly available even in the event of hardware failure.
- The framework's MapReduce programming model is designed for parallel processing, making it highly scalable for big data.
- Hadoop was uniquely flexible in its day, supporting structured, semi-structured, and unstructured data. This allowed enterprises to store and process data in its native format, without the need for extensive (and expensive) pre-processing such as ETL. This alone was an advantage over traditional, highly structured data warehouses. With Hadoop, the Data Lake became a practical reality.

In addition to these technical advantages, Hadoop was seen as a low-cost solution. It was open source, with a wide variety of distributions; it could run on commodity hardware, such as standard servers, enabling IT teams to build out large-scale data processing and storage systems at a much lower cost compared to traditional solutions.

Nevertheless, for all these advantages, there have always been some problems with Hadoop, which have gradually led to its decline.

Pain points of Hadoop

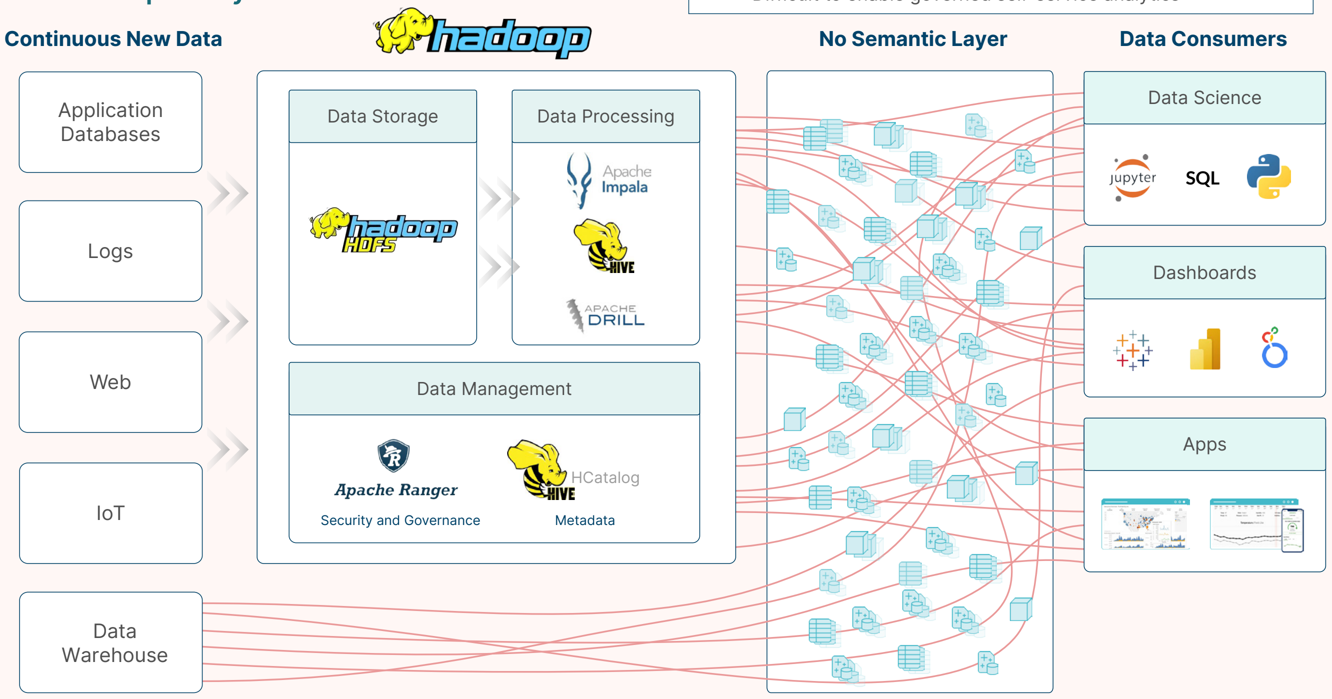
Hadoop, despite its many benefits, had some issues which increased costs and reduced efficiency.

- Hadoop requires a distinct level of expertise to set up and maintain. A Hadoop cluster can be complex and time-consuming, especially as the size of the cluster grows. The hidden costs of this administration can be exacerbated by the difficulty in hiring for the right Hadoop skills.
- Hadoop has some inherent security vulnerabilities, including rather weak authentication protocols & authorization policies. Similarly, the lack of native encryption at rest or in transit requires additional libraries – and the skills to implement those technologies.
- As an open-source framework with many tools and components, Hadoop suffers from a lack of standardization across distributions and implementations.

Challenges

- ✓ Requires deep expertise in the Hadoop ecosystem to maintain
- ✓ High cost of scalability as your data grows
- ✓ Query performance management
- ✓ Difficult to enable governed self-service analytics

The Hadoop Ecosystem



The Hadoop ecosystem as shown in the above diagram comes with a number of challenges associated with various common components including HDFS, Hive, Drill, Impala, Ranger, and Hive metastore.

- At the storage layer, HDFS can become a bottleneck especially as the number of files and directories increases. This can lead to slow read and write operations and ultimately impact the performance of the entire system.
- The Hive query engine is designed for batch processing and is not optimized for interactive queries, which can result in long query execution times.
- Meanwhile, Impala (a SQL query engine, although with a limited SQL implementation) is designed to provide fast interactive queries, it is optimized only for small to medium-sized clusters.
- In contrast, Apache Drill is an open-source distributed SQL query engine, providing high performance over large and complex datasets. It is also highly configurable, but with this comes considerable complexity, making it difficult for users to set up and manage.

“Hadoop is cheap to provision and expensive to run.”

Data management features suffer from similar complexities ...

- Apache Ranger is a security management tool that provides centralized administration. It is designed to provide granular access control and auditing but it’s difficult to configure. Moreover, it can prove to be resource-intensive in practice, which in some cases can impact the performance of the entire Hadoop cluster.
- Hive metastore is a repository describing Hive tables and partitions. It is somewhat difficult to scale, but more importantly it’s metadata support is limited especially for complex data types. Data Lineage is not comprehensively supported and there is limited statistical metadata available without additional analysis.

At one time, the advantages of Hadoop overcame these drawbacks and their hidden costs. But over the years many enterprises have found that Hadoop is cheap to provision and expensive to run. The case for migration is compelling, but the task ahead can appear formidable, especially for businesses that have committed so much to the Hadoop platform. Let’s look at some alternatives.

What are the choices?

For enterprises looking to migrate away from Hadoop, there are several alternatives available, but the migration can be daunting, not least because in Hadoop, compute and storage are coupled: the data nodes in a Hadoop cluster run both MapReduce tasks as well as store blocks of data from HDFS files.

This coupling limits scalability - adding extra computing power requires additional disks to be added at the same time. Maintaining the Hadoop cluster can be complex, as it requires managing both the data storage and processing components. And, as most enterprises discover sooner or later, tight coupling of compute and data in Hadoop can limit the performance of the system, as the processing power of the query engine is limited by the capacity of the data storage.

As a result, decoupling storage and compute is often one of the motivators for migration. Ironically, attempts to do so often increase costs even more, unless carefully phased. In this paper, we will propose just such a phased migration, but let us also consider the other options, in particular, a migration to cloud-based Hadoop, a cloud data warehouse, and a data lakehouse.

Hadoop in the Cloud

Naturally, one of the most tempting approaches to modernizing Hadoop is simply to move from an on-premises solution to the cloud, especially if you think the migration can be a simple "lift and shift" - simply moving all existing data & applications over without making any changes or modifications.

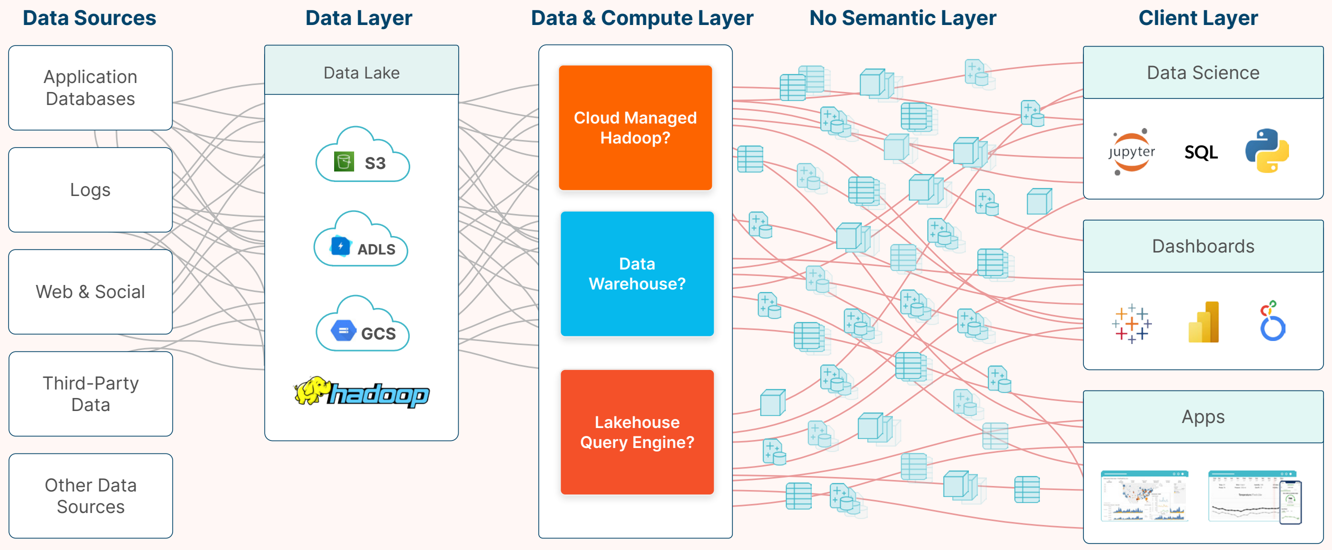
However, this might not always be feasible or certainly not simple. Some of the applications running on-premises may require specific versions or configurations that aren't supported in a cloud environment. This is more common than you may think, simply because there are such a wide variety of Hadoop distributions.

Nevertheless, there are some advantages to cloud-based Hadoop. Scalability is simpler than on-premises, making it easier to meet changing demands for processing and storage. Similarly, maintenance is reduced because the cloud provider typically manages the hardware and software components of the solution.

You may expect reduced costs from the cloud, but ... hold that thought! While the cloud can provide cost savings in some cases, it can be more expensive in others, especially for organizations with large amounts of data that will result in high storage and processing costs. The pay-as-you go contracts soon add up. And there are some other concerns:

- Vendor lock-in is a real concern for many businesses. Running Hadoop in the cloud makes you dependent on the cloud provider for the maintenance, security, and availability of the solution. It may be difficult to move to another cloud vendor if their service is unacceptable or costs increase.

What are your options?



- Customizing Hadoop in the cloud can be more challenging than on-premises, as cloud providers may have limitations on what can be done to the solution. There may not always be flexibility around certain parameters such as memory allocation. These new constraints can be frustrating for teams that have deployed and maintained their own customizations on-premises.
- Data processing and analysis in the cloud can result in higher latency compared to on-premises solutions, especially for organizations with large amounts of data.
- As we described earlier, there are different Hadoop query engines for different purposes, such as Hive for batch processing and Impala for interactive queries. This situation, with its complex deployment and configuration, does not improve in the cloud.

In short, migrating Hadoop to the cloud is neither as simple nor cost-efficient as you may at first think. Enterprises that initially went this route have found themselves migrating again to a cloud data lake that uses object storage and a data lakehouse query engine, such as Dremio Sonar.

Migrate to a cloud data warehouse

There is another, more radical, solution which enterprises often consider: migrating the data in their Hadoop system and the applications built over it to a cloud data warehouse. As the cloud has given the old data warehouse architecture a new lease of life this is no doubt tempting, but it’s a very different proposition compared to just moving your Hadoop data lake to the cloud.

If your data lake use cases are only analytical, you may get lucky with this pattern, but even with structured data only, if you have deployed a columnar database - like HBase or MapR - over Hadoop, you will find the migration to a data warehouse very difficult, especially for the high-performance lookups associated with columnar data.

Perhaps the more important point is simpler - if you are only using structured data for analytic workloads, why would you be using a data lake in the first place? Most data lakes are used for a variety of workloads, often including structured, semi-structured, and unstructured data. In these cases, users often find they still land the data in the lake, before copying it into the warehouse: a waste of time and resources when there are better solutions.

Another complexity is that moving to a cloud data warehouse requires a deep understanding of both architectures. You will also find it involves significant data engineering work to reformat and restructure the data for the data warehouse, particularly for organizations with large and complex data sets. And you’ll discover that data engineering is an ongoing cost.

Large enterprises, especially with international subsidiaries or which have grown by merger and acquisition, find themselves with multiple cloud and data warehouse vendors. These vendors generally

Lakes, warehouses and lakehouses: quick definitions.

Data Lake: A data lake is a large, centralized repository for storing all types of data in its raw, unprocessed form. This flexibility allows you to store all types of data, from structured data, such as spreadsheets, to unstructured data, such as text, images, and video.

Data Warehouse: A data warehouse is a database that is optimized for fast, efficient querying and analysis of structured data. Data Warehouses also apply business rules to the data, for example, to generate complex financial reports at multiple levels of the organization. This makes a data warehouse a great option if you have a large amount of structured data that you need to analyze quickly with well-defined business policies applied.

Data Lakehouse: A data lakehouse combines the best of both data lakes and data warehouses. It provides the flexibility of a data lake by allowing you to store all types of data, but it also provides the organization and efficiency of a data warehouse. One important benefit of this architecture is that it helps reduce data duplication by serving as the single platform for all types of workloads.

Zones and Medallions. Promising patterns and problematic practices.

In data lakes, it has been best practice to organize data into zones. - logical areas or partitions each designed to support a specific type of data and a specific set of data processing activities.

- The Raw Data Zone is where the data is ingested into the data lake in its original form, without any processing or transformation. This zone is primarily used to store the data as it is, without any modification, and to make it available for further processing.
- The Trusted Data Zone is where the data has been cleansed, normalized, and enriched to create a reliable version of the data.It is ready to be used for reporting, and some centrally-driven analytics like corporate dashboards.
- The Refined Data Zone is where the data is processed and transformed into specific data structures often for specific analytical use cases. The refined data can be used to support business intelligence, data mining, machine learning, and other types of data analysis. However, the semantics of the data are still difficult for business users to understand - the data is often further extracted to BI tools for this purpose.

In the cloud data warehouse, we often see a similar design pattern, sometimes called a Medallion architecture, because the “zones” are called Bronze, Silver, and Gold.

In simple terms, the Bronze layer acts as the first point of contact for all the data from the external source systems, this layer is often used for raw data storage. Table structures in this layer are an exact reflection of the table structures of the source system. Data in the Silver layer is cleansed and conformed, much like the Trusted zone of a data lake, and is intended for use by Data Engineers, Data Scientists, and analysts. The Gold layer is highly prepared data, sanctioned for use by non-specialist business users working with self-service tools.

The medallion pattern is an attractive concept, but it creates some significant problems, not least the repetitive engineering to create new copies of data in the different warehouse zones. Maintaining consistency, governance and the authority of data across such a system, where numerous different versions of the same data serve different audiences - that’s a recipe for confusion and unreliability.

We’ll see later how the medallion pattern can be implemented in a more effective way with Dremio.

use proprietary technologies and APIs that can make it difficult to migrate data between different platforms. This can lead to vendor lock-in, where enterprises are effectively tied to a particular vendor and have limited flexibility to switch to a different platform. In some cases, data engineers may even end up keeping the data lake to use as a staging area when moving data between platforms.

For business users, cloud data warehouses offer limited semantics. So, for self-service analytics, we often find there are additional data layers required, such as the data extracts used by PowerBI, Tableau, or Qlik. Although often offering good performance, the complexity and supplementary governance involved is a real drawback of this approach.

With all this, migrating Hadoop to a cloud data warehouse is neither simple nor fast. The migration process not only involves assessing the current Hadoop setup, determining the target cloud data warehouse platform, and moving the data, but you also need to substantially rebuild your applications and workflows to run on the new platform. You may even find that mixed workloads – analytical and transactional – that you have engineered on Hadoop simply don’t work in your cloud data warehouse, or if they do, only with reduced functionality.

The cloud data warehouse is such a different paradigm from Hadoop that migration either does not get the best from either architecture or becomes a lengthy, complex process of redesign and re-architecture.

Migrate to a data lakehouse

The most impactful modernization option is to migrate to a data lakehouse. We have already defined this new architecture briefly as offering the best of both data lakes and data warehouses.

In the data lake, sources are just stored in a raw format - so you will often hear it referred to, rather disparagingly, referred to as a data dump. The data warehouse has more organization, but at the cost of constant data engineering, whether with scripts, pipelines, ETL, ELT, or some other process of moving and transforming data. Many practitioners use a rough rule-of-thumb that 80% of the effort in building and managing a data warehouse project is spent on data movement in order to provide a store for BI and reporting tools to work with.

With the lakehouse, various BI and reporting tools have direct access to the data they need without complex and error-prone ETL processes. Additionally, since the data is stored in open file formats like Parquet, data scientists and ML engineers can build models directly on any data (structured, unstructured) depending on their use cases.

It's not just about performance: a Dremio data lakehouse has some significant organizational advantages over other lakehouse architectures which improve governance and usability too.

Earlier, we described the popular "medallion" approach, where data is incrementally refined through Bronze, Silver, and Gold tiers from raw source copies to business-ready, integrated, highly-prepared versions. But the problem with this model is apparent, even in the description: copies, versions - every new variation generates more artifacts to manage, maintain and govern.

In a Dremio lakehouse, data use cases are tiered, but data can remain in a single, tightly-governed, performant dataset that is already under management by IT. This model affords predictable, excellent performance (the same engine with the same resources drives each scenario), usability for business users, and access to reporting applications, all without copying data.

We can think of this model as having three layers, too:

- The **Staging Layer** is where data preparation takes place, including data cleansing, data-type conversions (such as from strings to dates), and calculated columns. The result, for each physical data set, is an enhanced virtual data set without the need for a new copy of the physical data.
- The **Semantic Layer**, as the name suggests, further enhances virtual data sets with metadata focussed on business usage. This creates a common view for business users which is readily accessible and user-friendly for self-service analytics, visualization, and data storytelling.
- The **Reporting Layer** is where applications can access data prepared for reporting purposes. For example, this is where you would implement grouping for reports and create aggregation tables. Again, the virtual nature of this layer means that you are not imposing a burden of management and governance for each object. Performance, however, remains excellent.

There's something particularly powerful about migrating to a Dremio data lakehouse. The federation of data, rather than endless copying of data, means that this is not much of a migration at all. As you'll see below, the task at hand is not a massive "lift and shift" operation, but rather the swapping out of the query engine, defining the virtualized semantic layer, and only then migrating your HDFS objects to the cloud.

Let's look at this Dremio-specific migration process.

A Phased Approach for Migrating Hadoop to an Open Data Lakehouse with Dremio

Just before we dig into the steps of migrating to a Dremio lakehouse, let's think a little about the end game: what advantages are we looking for?

Multiple Analytical Workload Support

Business users, data scientists, and machine learning engineers should be able to use the same architecture, directly accessing the right data, at the right time, and in the right format *for them*.

Cost

With all the data stored in a cost-effective cloud object storage, organizations don't have to pay hefty costs associated with data warehouses. Data lakes also serve as a central repository for an organization's data, so there is no overhead to storing data in multiple systems and managing them.

Data Copies

With a data lakehouse architecture, engines can access data directly from the data lake storage without copying data. This ultimately ensures reliable data in the downstream applications and helps prevent issues such as data drift, concept drift, etc.

No Lock-In or Lock-Out

The open nature of the data lakehouse architecture allows businesses to use multiple engines on the same data, depending on the use case and helps to avoid vendor lock-in. For new workloads, organizations can add any new tool to their stack.

Independent Scaling of Compute and Storage

A lakehouse architecture separates compute and storage which helps scale these components independently to cater to the needs of an organization.

Infinite Scalability

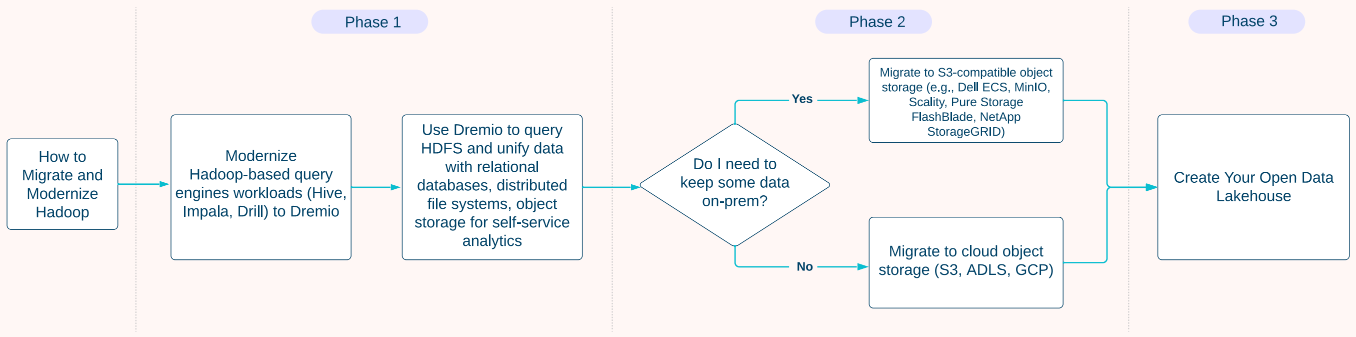
Scaling up resources infinitely based on the type of workload is another important aspect that the lakehouse addresses. If it's a resource-intensive task, organizations can decide to scale up easily.

Those are very tempting goals for an organization. Now let's look at the straightforward, three-step process to get there.

Step 1: Modernize the Query Engine & Provide Self-Service Analytics

The first thing we need to do is to separate out storage and compute – one of the most constraining bottlenecks of the Hadoop architecture. This allows data to be processed faster, as the processor does not need to wait for disk input/output operations. It also reduces contention between IO requests from multiple users or applications that can slow down a system if all resources are in one box. What's more, it's easier to scale out either resource independently of the other when needed. This is especially useful during times of peak demand – a common scenario in data analytics - where additional computing power may be required without having to add extra storage at the same time.

When we decouple storage and compute, we are effectively replacing the query engine in your Hadoop distribution with the Dremio engine.


Phase 1
Phase 2
Phase 3
How to Migrate and Modernize Hadoop
Modernize Hadoop-based query engines workloads (Hive, Impala, Drill) to Dremio
Use Dremio to query HDFS and unify data with relational databases, distributed file systems, object storage for self-service analytics
Do I need to keep some data on-prem?
Yes
Migrate to S3-compatible object storage (e.g., Dell ECS, MinIO, Scality, Pure Storage FlashBlade, NetApp StorageGRID)
No
Migrate to cloud object storage (S3, ADLS, GCP)
Create Your Open Data Lakehouse

Step 1a:

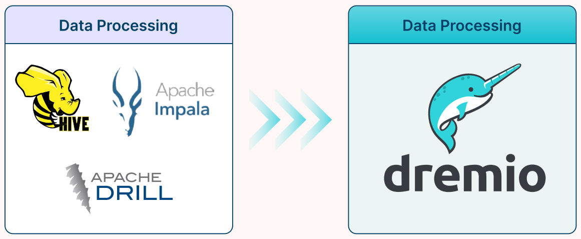
Modernize Hadoop Query Engine

What happens?

- ✓ Connect Dremio to existing Hadoop clusters and simplify the transition to modern cloud object storage.
- ✓ **Minimize** impact to production system.

Results:

- ✓ Immediately improve query performance over Hive, Drill, and Impala



Minimize any impact on your production workloads by deploying Dremio in parallel with your Apache query engine. Dremio provides faster performance for both interactive and ad-hoc queries compared to Hive and Impala, enabling sub-second queries on your dashboard that run directly on the data lake.

Hadoop distributions typically include one of the following query engines:

- Apache Hive: A mature, but slow, data warehousing and SQL-like query language for big data.
- Apache Impala: A faster, interactive distributed SQL query engine for Apache Hadoop.
- Apache Drill: An open-source SQL query engine for big data exploration with important improvements such as schema-free querying and an architecture that enables storage plug-ins for new data sources.

Some of these characteristics may have been decisive in your choice of Hadoop distribution. Or you may already have swapped out different Apache engines in the search for better performance or flexibility.

Whatever your current Hadoop query engine is, you'll find advantages in decoupling it and modernizing it with Dremio.

Overall, Dremio offers significant advantages over the query engines included in Hadoop distributions, including faster performance, and a more intuitive SQL interface, all while minimizing the impact on production systems.

Although performance is one business justification for migrating from Hadoop, it is not the only criteria in choosing a query engine. We must prioritize governed data access for data consumers without bottlenecking IT. With Hadoop, data consumers could access Impala or Hive from any BI tool. But performance and efficiency simply were not there, causing IT to lock down those environments and only provide curated datasets (which require IT tickets for additional datasets and changes). Data engineers had self-service access because they had the skills to write their own (usually more) efficient jobs to do exploratory access and build their own datasets, leaving data consumers out of the picture.

Swapping engines - it's not surgery

To replace a query engine like Apache Hive with Dremio, all you need to do is set up the Dremio ODBC or JDBC driver. After configuring your connection with one of these two options, update any existing application code that accesses data stored in Hive and point it toward your new connection string utilizing either a JDBC or an ODBC URL format depending on which option was used during configuration; this will then allow those applications to start querying against data within Dremio instead of Apache Hive without making any other changes necessary.

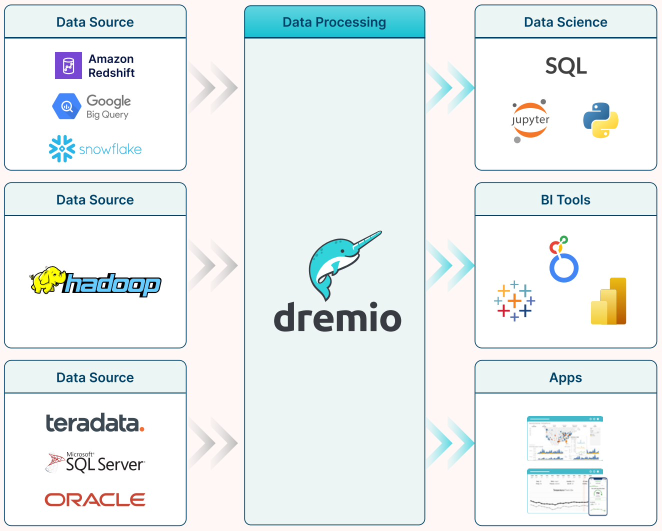
Step 1b:
Provide Self-Service Analytics

What happens?

- ✓ Unify all your data for self-service analytics using Dremio's semantic layer
- ✓ Connect and federate queries across other data sources
- ✓ Minimize impact to production system and **reduce complex ETL footprint**

Results:

- ✓ With Dremio's semantic layer, these sources are given business-friendly names, helping to deliver reliable data products across all your downstream applications.



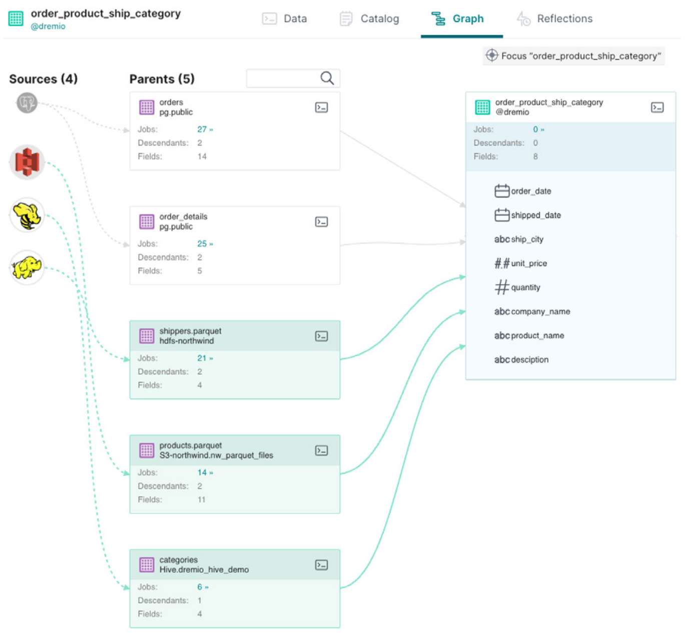
Next, we will cover how Dremio's semantic layer provides self-service analytics across all your data.

With Dremio's query engine, you also get a semantic layer for self-service. This helps create a more centralized approach to self-service analytics and integrates with a wide variety of data sources beyond Hadoop. Federate queries and unify data across various types of databases and file systems, both structured (e.g. relational, Parquet, Iceberg) and unstructured (such as JSON).

Dremio's semantic layer also gives your team a flexible way to organize views of data that makes the most sense to them. Data teams can create a logical view of the data for data consumers, with the physical data residing in multiple locations such as cloud data lakes (S3, ADLS), on-prem data lakes (HDFS), and RDBMS and NoSQL databases. Without this ability, data teams have to copy all the required data into one location and then create logical views against it. Then, additional copies must be created to further aggregate the data for performance.

Having made the decision to modernize and having taken the first step of decoupling storage from compute and swapping in the Dremio query engine, you will already see some significant rewards from day one: performance, the federation of data, improved usability, and the power of a rich business semantics for analysts.

These business users now have the option to connect and use Dremio from their BI tools and SQL clients for their self-service needs. Once users experience the improved ease of use and performance, they quickly switch to only using Dremio for self-service access to data. That's a compelling first step in itself.



Dremio unifying data in HDFS, Hive, Amazon S3, and PostgreSQL database.

The Unified Semantic Layer

Dremio's semantic layer is a powerful tool for unifying different types of information from multiple databases into coherent views without needing to write complex code. The semantic layer serves as a bridge between the raw data and the business user or data analyst, providing a consistent and standardized view of the data that enables self-service analytics across all your data.

A semantic layer also helps to insulate the user from changes to the underlying data structure, such as changes to table names, columns, or relationships, so that the user's view of the data remains consistent even as the underlying database evolves over time.

It does this by enabling a user to query distributed datasets by providing virtual "views" on top of physical storage, allowing for easier exploration without having to write complex code or learn new languages.

Dremio's semantic layer enables some specific advantages:

- **Consistent View of Your Data Products:** A centralized location for defining and managing data definitions, policies, and metadata, which helps to ensure data consistency and accuracy across the organization.
- **Self-Service Data Curation:** Why define isolated definitions of data, calculated fields, and virtual datasets in individual BI tools where they can't be used by your teammates? Create dataset definitions and calculated fields that can be leveraged by any downstream application.
- **Data Lineage:** Tracking the lineage of data and providing a clear understanding of where data comes from, how it is processed, and how it is used.
- **Data Governance:** The Dremio semantic layer can be used to implement data access controls and security policies that can be enforced at the virtual layer, reducing the risk of data breaches and unauthorized access. For example, role-based access controls (RBAC) can be implemented, much like in a database system, using familiar ANSI-SQL commands.

With these capabilities, Dremio mitigates the need for an additional semantic layer architecture, whether from a specialized vendor or built into a business intelligence platform. It also dispenses with additional security frameworks, such as Apache Ranger.

Step 2: Migrate from HDFS to Cloud Object Storage

Once you have created a high-performance and self-service experience with Dremio, you may feel you have already won some victories in your migration. But there is more advantage to be gained if you take another step.

Migrating from the Hadoop File System (HDFS) to object storage is a sound idea for several reasons. Object stores are more cost-efficient than traditional file systems such as HDFS, and they offer better scalability due to their distributed nature, being designed to scale horizontally.

HDFS is optimized for batch processing, but object stores provide direct access to individual files, making it easier and more efficient to retrieve specific data.

Another attraction of this approach, especially for security-critical businesses, is that using an API layer enables access control policies and data protection measures (such as encryption) at the level of an individual object rather than being limited by directory or folder security settings that come with traditional file systems like HDFS. Additionally, these APIs enable built-in versioning capabilities which makes it easier to track changes over time compared with files on a file system like HDFS.

As mentioned above, object storage can be in the cloud or on-premises. In fact, it is not uncommon to build a hybrid system. For example, some customers store some of their data on-premises due to compliance or organizational reasons. Other organizations store data in cloud object storage to take advantage of the cost, maintenance, and flexibility.

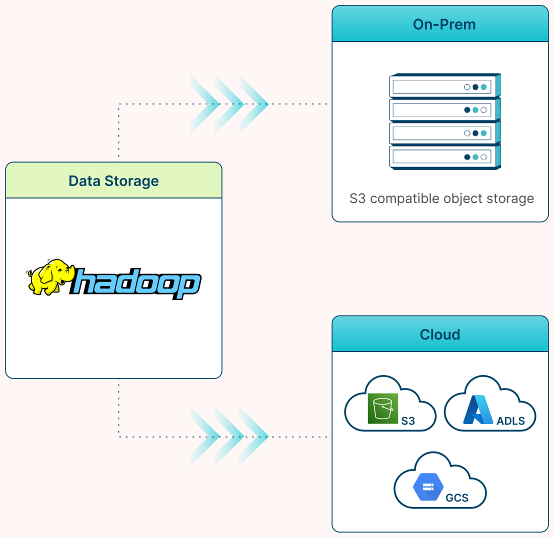
Step 2:
Migrate off HDFS to Object Storage

What happens?

- ✓ Start migrating off HDFS to object storage
    - Data that needs to be on-prem can migrate to S3 compatible object storage
    - Everything else can go to cloud object storage

Results:

- ✓ Eliminate Hadoop costs (Cloudera license and underutilized servers)
- ✓ Minimize impact to business continuity on HDFS with this phased approach
- ✓ Scalability



Customers can use Dremio software on-premises and Dremio Cloud for data migrated to cloud object storage. Use the Dremio connector to bridge the Dremio instances to ensure this hybrid experience is seamless for the business. This provides a flexible deployment model to accommodate your specific object storage requirements, enabling a smooth and secure migration from HDFS.

This second step requires thorough planning and testing to ensure that migration is failsafe. But the advantages are considerable, in cost, security, scalability, and ease of administration. Furthermore, with business user access established through Dremio's semantic layer, the move to object storage can occur without business users even noticing the migration.

Step 3:
Create an Open Cloud Data Lakehouse

Earlier in this paper, we were quite critical of the approach that involves migrating from Hadoop to a cloud data warehouse. In particular, we noted that the warehouse model is a very different concept from Hadoop, so migration is really a process of completely redesigning and re-architecting the system.

Yet it is fair to say that data warehouses have their attractions, such as a highly efficient architecture for querying, albeit at the cost of considerable upfront data preparation.

Luckily, there is a third alternative: the data lakehouse. If this is unfamiliar to you, see the sidebar for a brief definition.

This third step is where you will realize the full benefits of the Dremio data lakehouse.

For administrators the advantages are numerous. As we have described, the security of features of the semantic layer obviates the need for the Apache Ranger framework. As we will see later, lakehouse management with Dremio Arctic replaces the requirement for the Hive metastore. When dealing with varied workloads, Dremio Sonar replaces both Hive and Impala with better performance and manageability.

Most importantly, the order-of-magnitude increase in query performance provided by Dremio eliminates the need for many of the local copies of data made previously to achieve good query performance and mitigates the downsides of maintaining and securing these copies.

For example, batch reports that send users emails, PDFs, Excel files, etc., are often used because interactive reporting is too much for the system – especially with Hadoop. With Dremio efficiently providing interactivity directly on the data lakehouse, the need for these batch reports is reduced; canned reports can be made interactive, giving users of the report the ability to drill down into the report with the latest data.

Step 3:

Create your open data lakehouse

What happens?

- Data is in cloud object storage
- Migrate Hive tables to open table format like Apache Iceberg

Results:

- Future proof your data architecture
- Achieve higher performance, DML, schema evolution, time-travel, and other data warehouse functionality
- Avoid vendor lock-in from proprietary table formats
- Make data accessible to your query engine(s)

HIVE
dremio

Hive Table	Apache Iceberg
Hive Metastore	Dremio Arctic

Similarly, aggregation tables, another form of self-managed data copies, are used even in cloud data warehouses to provide interactivity on lower granularity data, whether for pure performance or cost-efficiency. These are created and managed by ELT scripts which must be written, scheduled, operated, monitored, and maintained as things change upstream and downstream. Dremio removes the need for self-managed aggregation tables with **data reflections.**

This step is also generally when data scientists begin to leverage Dremio for core datasets and standard KPI definitions in their work. They're able to use standard definitions to enrich their machine-learning models, and still access them when their queries have large result sets.

We recommend migrating your Hive tables to Apache Iceberg, an open table format built specifically for large, enterprise datasets. Iceberg is an open-source project with contributors from many technology companies, including Amazon Web Services (AWS), Google, Snowflake, Netflix, and more. It provides a unified table abstraction for data stored in different formats like Parquet or ORC files. Apache Iceberg also enables users to track changes over time by storing metadata about each version of the dataset so they can easily access historical versions as needed.

However, Dremio is more than just a SQL lakehouse query engine. It also includes a powerful *metastore* called Dremio Arctic.

Table Formats in the Lakehouse

A key consideration of a data lakehouse design is the table format: a means for tools to look at files in the data lake and see groups of those files as a single table. You can then query and transform the data performantly directly on the data lake, enabling data warehouse-like workloads.

Open table formats like **Apache Iceberg**, enable advantages such as ACID transactions, time travel, schema evolution, partition evolution, and more. Iceberg is decoupled from the file layout of Apache Hive, so it can deliver higher performance at scale than other table formats. It's also open-source which makes it easier for tools and vendors to support this format.

A metastore handles metadata such as table definitions, column names, and types, or even partitioning information. They also serve other functions like providing access control rights over particular objects within the system.

Hadoop users frequently use the Hive Metastore as a single point of access for metadata about all their data, whatever its structure, format, or location. However, Hive Metastore is little more than a catalog of tables and their access privileges. You need something more for serious metadata management.

Dremio Arctic is a data lakehouse management service built for Apache Iceberg, with some key capabilities:

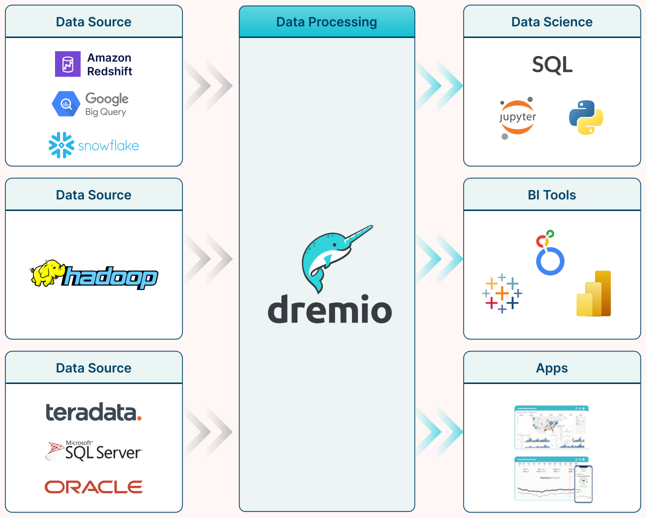
- **A modern lakehouse catalog:** Dremio Arctic features an Iceberg-native catalog that ensures data remains in an open format that is accessible by multiple execution engines, so data teams can choose the best tool for each analytic workload. It enables easy management and governance of multiple isolated domains, and features table, row, and column-based access control and integration with user and group directories, such as Azure Active Directory and Okta.
- **Automatic data optimization:** Dremio Arctic automates tedious data management tasks, including table partitioning, which rewrites several small files into larger files and groups similar rows together, and table cleanup, which removes unused manifest files, manifest lists, and data files. Automatic data optimization ensures high performance queries and optimizes storage resources.
- **Data as code:** Dremio Arctic enables data teams to manage data the same way software developers manage code. With Git-like functionality such as branches, tags, and commits, data teams can make changes, experiment with data, and test for quality in isolation, without impacting production datasets. Data as code enables a consistent and accurate view of the data across data consumers.

In operation, the Dremio data lakehouse offers numerous other efficiencies. For example, Dremio allows for seamless interactivity at scale through data reflections, which combine the best features of both materialized views and indexes, but are managed internally by Dremio. These reflections are transparent to end users yet require no additional administration.

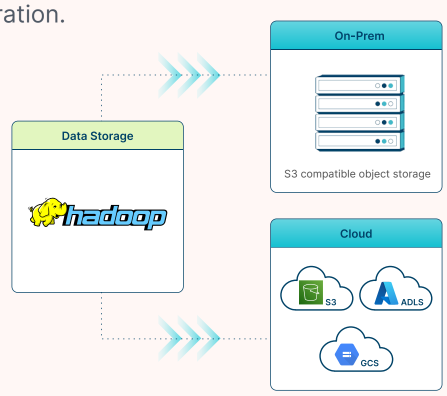
At this final step of the migration, you will have quickly and efficiently moved from an aging and restrictive Hadoop implementation to a more efficient architecture for analytics, machine learning, and data management. But as we have seen, this need not be a disruptive or costly process.

A summary of the Hadoop Migration to the Dremio Data Lakehouse

Step 1: Decouple storage and compute. Replace your Hadoop query engine with Dremio. Result: Faster performance, more intuitive SQL, and data federation. which makes it easier for tools and vendors to support this format.



Step 2: Migrate Hadoop to object storage. Deployment can be on-premises, in the cloud, or hybrid. Result: Lower cost, better security, greater scalability, simpler administration.



Step 3: Build your Open Data Lakehouse. Start migrating your Hive tables to an open table format like Apache Iceberg. Result: A modern, scalable, flexible lakehouse with outstanding performance, manageability, and ease of use.

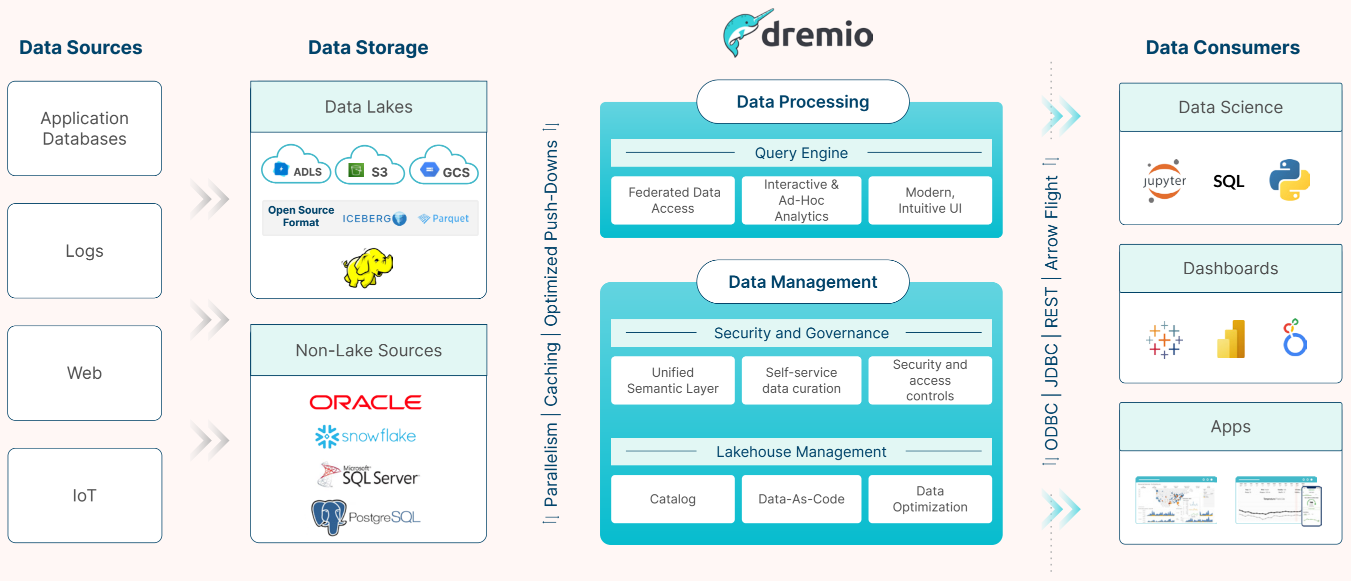
HIVE
dremio

Hive Table	Apache Iceberg
Hive Metastore	Dremio Arctic

Hadoop to Dremio Data Lakehouse

hadoop	dremio
Data Storage • HDFS	**Data Storage** • Cloud object storage (S3, ADLS, Google) • On-Prem S3 compatible object storage
Query Engine • Hive, Impala, Drill	**Query Engine** • Dremio Sonar
Table Format • Hive	**Open Table Format** • Apache Iceberg
Metastore • Hive Metastore	**Metastore** • Dremio Arctic
Security and Governance • Apache Ranger	**Security and Governance** • Dremio Sonar

Mapping Hadoop components to the Data Lakehouse with Dremio



Open Data Lakehouse Architecture with Dremio

Summary

If you are running Hadoop today, you are - or should be - already considering how you migrate to a new generation of technology. There's little doubt that Hadoop was revolutionary in its day, but that day has passed.

What should that new generation of technology look like? Running Hadoop in the cloud is not really an upgrade. It does improve scalability and maintenance, but it's still an older technology that affords few new opportunities.

The cloud data warehouse is promising but it remains a "back office" upgrade with few advantages for business users. In fact, the lack of user-ready semantics in the cloud data warehouse imposes a need for yet more technology: whether a specialized product or applying business definitions and formats by extracting data into the semantic layer of business intelligence tools.

A better option is a data lakehouse. More especially, as we have shown, a cloud data lakehouse built with Dremio is not only high performance, but the integrated semantic layer and advanced metadata management make it a perfect complement to business intelligence and data science, while still meeting the needs of IT.

Our suggested approach to migration - with three technically effective and economically compelling phases - takes the risk out of migration, but enables business users with better data access from the first step.

Learn more about getting started with Hadoop Migration and Modernization with Dremio.

Request a demo

Learn More

Additional Resources: Customers That Have Migrated Hadoop to the Data Lakehouse with Dremio

- OTP Bank
- NCR
- eMAG

dremio

ABOUT DREMIO

Dremio is the easy and open data lakehouse, providing self-service analytics with data warehouse functionality and data lake flexibility across all of your data. Use Dremio's lightning-fast SQL query service and any other processing engine on the same data. Dremio increases agility with a revolutionary data-as-code approach that enables Git-like data experimentation, version control, and governance. In addition, Dremio eliminates data silos by enabling queries across data lakes, databases, and data warehouses, and by simplifying ingestion into the lakehouse. Dremio's fully managed service helps organizations get started with analytics in minutes, and automatically optimizes data for every workload. As the original creator of Apache Arrow and committed to Arrow and Iceberg's community-driven standards, Dremio is on a mission to reinvent SQL for data lakes and meet customers where they are on their lakehouse journey.