snowflake®

MODERNIZING TO A DATA LAKEHOUSE ON SNOWFLAKE

How architects and data engineers can manage data lakes better

TABLE OF CONTENTS

INTRODUCTION

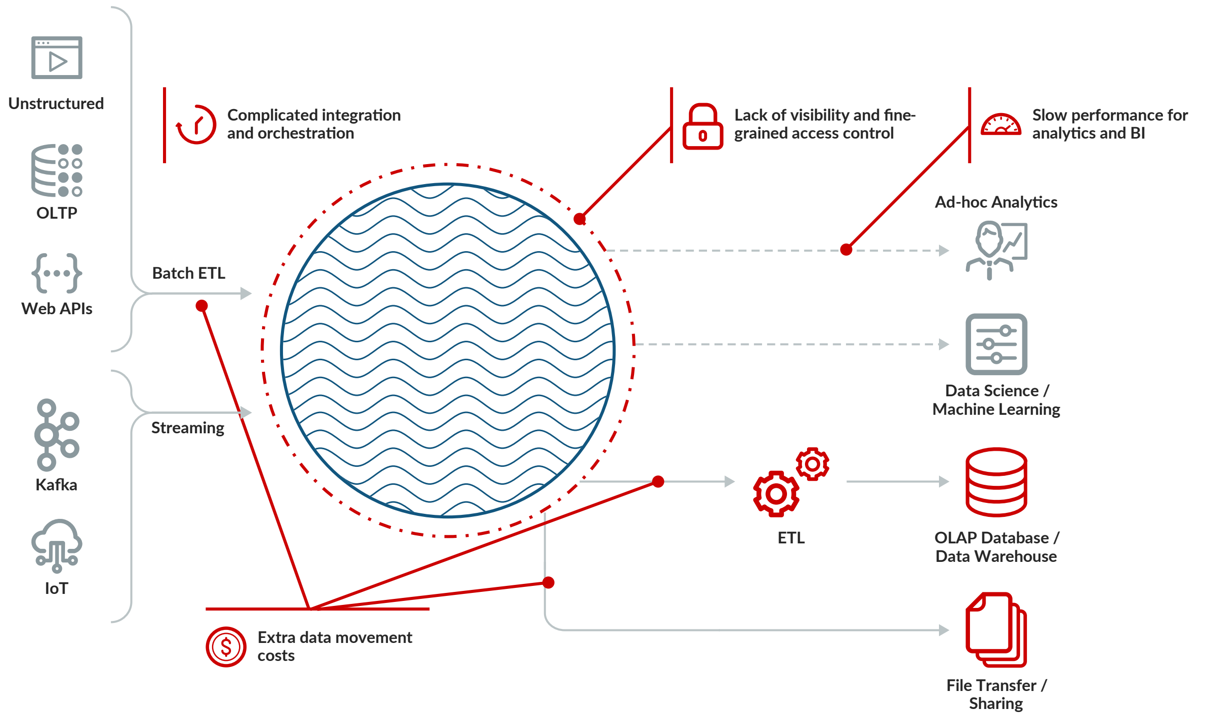
In today's data and AI landscape, efficiently managing vast amounts of data to power analytics and AI applications can create new business opportunities and a competitive edge. Traditional on-premises data lakes are often limited in scalability and agility, prompting organizations to explore cloud-based solutions. But even so, many cloud data lakes lack the reliability, performance and governance controls required by organizations.

This ebook serves as a comprehensive guide for data architects seeking to transform their cumbersome data lakes into a data lakehouse architecture with the Snowflake platform. From understanding core concepts to mastering migration strategies, each chapter provides guidance from in-house experts to help you create the necessary data foundation to unlock the full potential of your data with AI.

CHALLENGES WITH DATA LAKES

Data lakes, with their flexibility and scalability, are often the foundation for organizations' strategic initiatives to make decisions and innovate. But organizations should carefully consider several factors before deciding to build one or modernize from current tooling to a new platform. Cloud data lakes have usually leveraged cloud object storage rather than the Hadoop Distributed File System (HDFS), and they involve working with data in the form of a large number of files. Whether organizations are running Apache Hadoop® on premises or lift-and-shifted data and workloads to the cloud, they likely still face many challenges in terms of administration complexity, performance, governance, compliance, security and usability.

While there are many open source tools for data lakes in the form of storage formats and query engines, integrating many of these tools into a complete solution is complex and difficult to scale, often requiring a team of highly skilled engineers to keep everything running smoothly.


Unstructured
OLTP
Web APIs
Kafka
IoT
Batch ETL
Streaming
Complicated integration and orchestration
Lack of visibility and fine-grained access control
Slow performance for analytics and BI
Ad-hoc Analytics
Data Science / Machine Learning
ETL
OLAP Database / Data Warehouse
File Transfer / Sharing
Extra data movement costs

COMPLEXITY

Managing and maintaining data platforms can be complicated, especially for data lake solutions that integrate many separate tools. For example, data teams are responsible for integrating lots of incoming data from many sources, while ensuring data quality, accuracy, security and privacy practices are compliant with evolving regulations. As data volumes grow, concurrency requirements increase and latency requirements speed up, it becomes even more complex to manage data lakes — emphasizing the need for an integrated, scalable and easy-to-use platform.

SILOED

One of the challenges organizations face with data lakes is the siloed nature of their architecture. Each data lake is often isolated from others, meaning data sharing across organizations or departments becomes a complex, cumbersome process. This lack of direct connectivity may increase security risk, as organizations may resort to manually transferring or duplicating data, which introduces the potential for breaches and inconsistencies. Moreover, the repeated copying of data adds unnecessary costs, both in terms of storage and processing. As a result, the value of the data is diminished, and the potential for cross-organizational collaboration is hindered, creating significant inefficiencies.

PERFORMANCE

Both on-premises Hadoop data lakes and cloud data lakes often require lots of effort to optimize storage for performance. Meanwhile, stale (or even nonexistent) statistics offer little help in query planning, ultimately hampering efficiency in execution. Columnar file formats such as Apache Parquet can provide statistics to aid performance, but choosing to forgo writing these statistics in ingestion and transformation pipelines can negatively impact downstream read performance for analytics and business intelligence. Poor performance limits the value of data — extending the time it takes to derive insights, take action and make an impact.

GOVERNANCE, COMPLIANCE AND SECURITY RISKS

Data lakes often store vast amounts of data from various sources with limited structure, which can make it challenging to establish proper governance controls and audit usage. Manual effort and additional tooling are often required to implement access controls and monitor data access, quality, lineage, PII detection, disaster recovery and more.

DESIGNING A MODERN ARCHITECTURE WITH SNOWFLAKE

WHY LAKEHOUSE ARCHITECTURE

An open data lakehouse architecture uses open standards to provide a table abstraction over files, providing the flexibility to run a variety of workloads with engine interoperability on a shared table abstraction. Apache Iceberg™ is an open table format specification, providing an abstraction layer on data files stored in open file formats and improving data lakes with:

- **Reliability of processing and queries with ACID (atomicity, consistency, isolation, durability) transactions**
- **Ability to simplify data engineering tasks with time travel and more efficient schema and partition evolution**
- **A more intuitive user experience and better query performance with hidden partitioning and more metadata for efficient pruning**

Originally developed at Netflix and open sourced to the Apache Software Foundation, Apache Iceberg has created an open ecosystem that optimizes engine and catalog interoperability. The diverse project management committee and contributors have kept the project from being bound to any single vendor's whims, resulting in significant adoption and wide read-and-write support from query and processing engines.

WHY SNOWFLAKE

Snowflake is a single, global platform that connects businesses and their data of any type, scale or workload. While initially designed as a data warehouse built for the cloud, Snowflake has expanded as a platform powering many use cases — such as AI and machine learning (ML), data engineering, applications and transactional data — and many architecture patterns, including data lake, data warehouse, data lakehouse and data mesh.

Snowflake's platform architecture is a hybrid of traditional shared-disk and shared-nothing database architectures. Similar to shared-disk and data lake architectures, Snowflake uses a central data repository for persisted data that is accessible from all compute nodes in the platform. But, similar to shared-nothing database or data warehouse architectures, Snowflake processes queries using compute clusters, where each node in the cluster stores a portion of the entire data set locally. This approach offers the data management simplicity of a shared-disk architecture but with the performance and scale-out benefits of a shared-nothing architecture.

What are the benefits of modernizing a data lake with a lakehouse architecture on Snowflake?

- **Easy:** Snowflake abstracts away complexity to allow you to get the job done quickly. It's offered as a service, so you don't need to worry about hardware, software, installation, patches, updates or table optimization — Snowflake automatically handles this. Spinning up compute to handle demand peaks, and shutting down during lulls, can be done quickly.
- **Connected:** You can tap into Snowflake's global ecosystem of thousands of providers of apps, models and data sets to enrich your own data lakehouse. This is done with live, secure sharing, so you don't have to worry about costs involved with additional data movement or ETL.
- **Trusted:** Centralize security controls on top of your data in one place, where it is accessible and usable to many commercial and open source tools. Snowflake Open Catalog provides this interoperability, while Snowflake Horizon Catalog can satisfy more advanced governance such as PII detection and tracking.

ORGANIZATIONS MODERNIZING DATA LAKES WITH APACHE ICEBERG ON SNOWFLAKE

BMW GROUP

"Everything has to feel seamless for our users. Snowflake's ability to bring compute to us has made that a reality. It's what made us realize Snowflake would be a great partner, as we can store data on AWS, Iceberg or anywhere else and still get the compute benefits of Snowflake."

Ruben Simon
Head of Product Management,
Cloud Data Hub, BMW Group

Learn more

Goldman Sachs

"Across 900+ data sources, we've been able to consistently reduce the entire end-to-end process from 15 days to one day. Thanks to our lakehouse built on Snowflake and Iceberg, it's all canonicalized and centralized in one place."

Abhishek Narang
Managing Director, Goldman Sachs

Learn more

Booking.com

"Apache Iceberg's large and diverse ecosystem of contributors and products made it a clear choice for us to provide an open and common data layer across our internal and external ecosystem. With Iceberg, we can broaden our use cases for Snowflake as our open data lakehouse for machine learning, AI, business intelligence and geospatial analysis, even for data stored externally."

Thomas Davey
Chief Data Officer, Booking.com

Learn more

WHOOP®

"The flexibility and vendor-neutral control of both Apache Iceberg and Apache Polaris open source projects are very appealing to ensure interoperability and no lock-in."

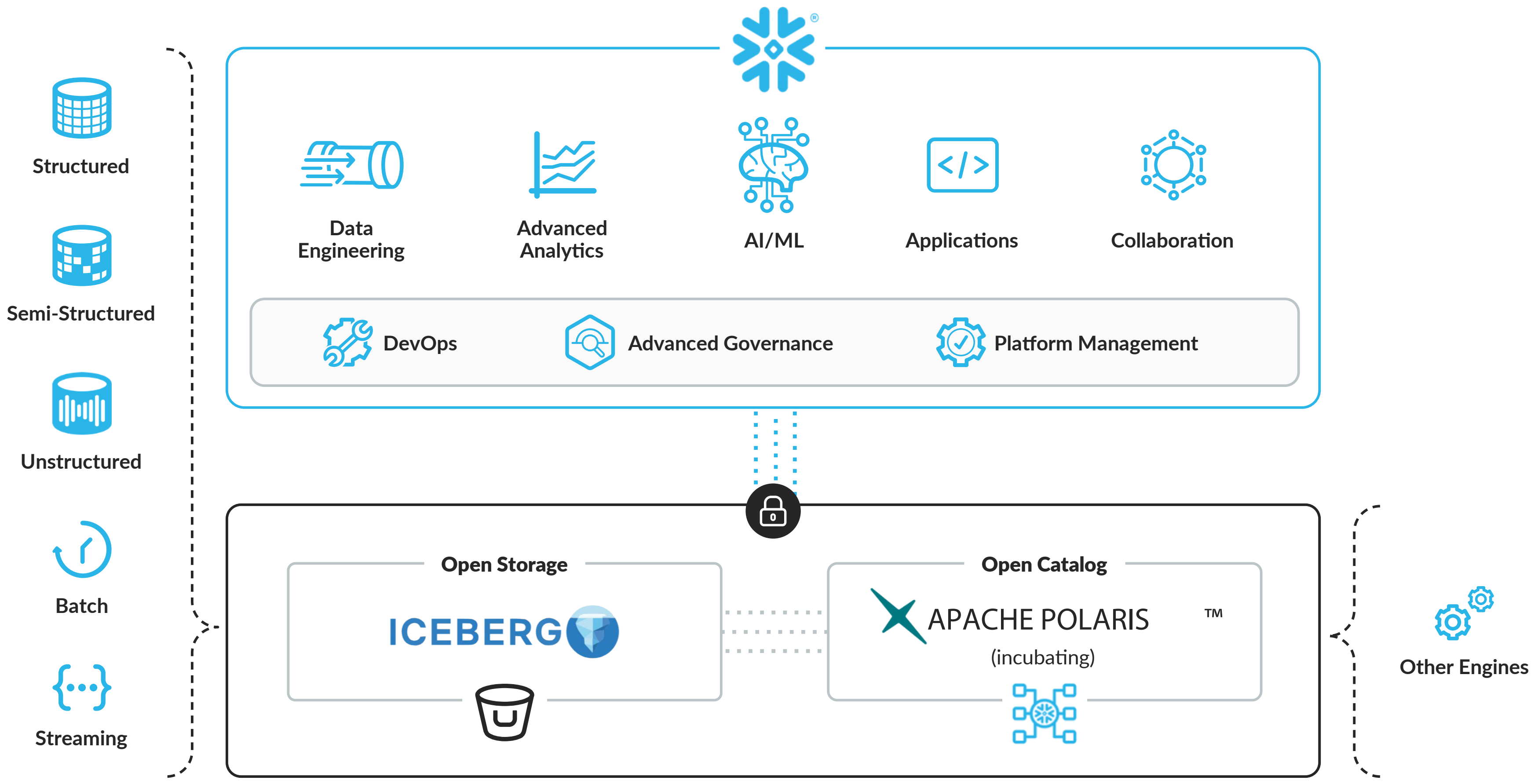
"Having our data instantly available through Snowflake saves us tens of thousands of dollars per month, which is essential for a company of our size and scale."

Matt Luizzi
Director of Business Analytics, WHOOP

Learn more

DESIGNING AN ARCHITECTURE WITH SNOWFLAKE

Interoperability is one of the great benefits of the open source standards at the foundation of a data lakehouse. As the industry support for Iceberg rapidly grows, so too does the flexibility for what tools can be integrated. Whether your organization needs many tools in your architecture, or you prefer consolidating to fewer tools, you can integrate Snowflake to satisfy a variety of use cases. In the pages that follow, we highlight some examples of lakehouse architecture patterns you can build with Snowflake.



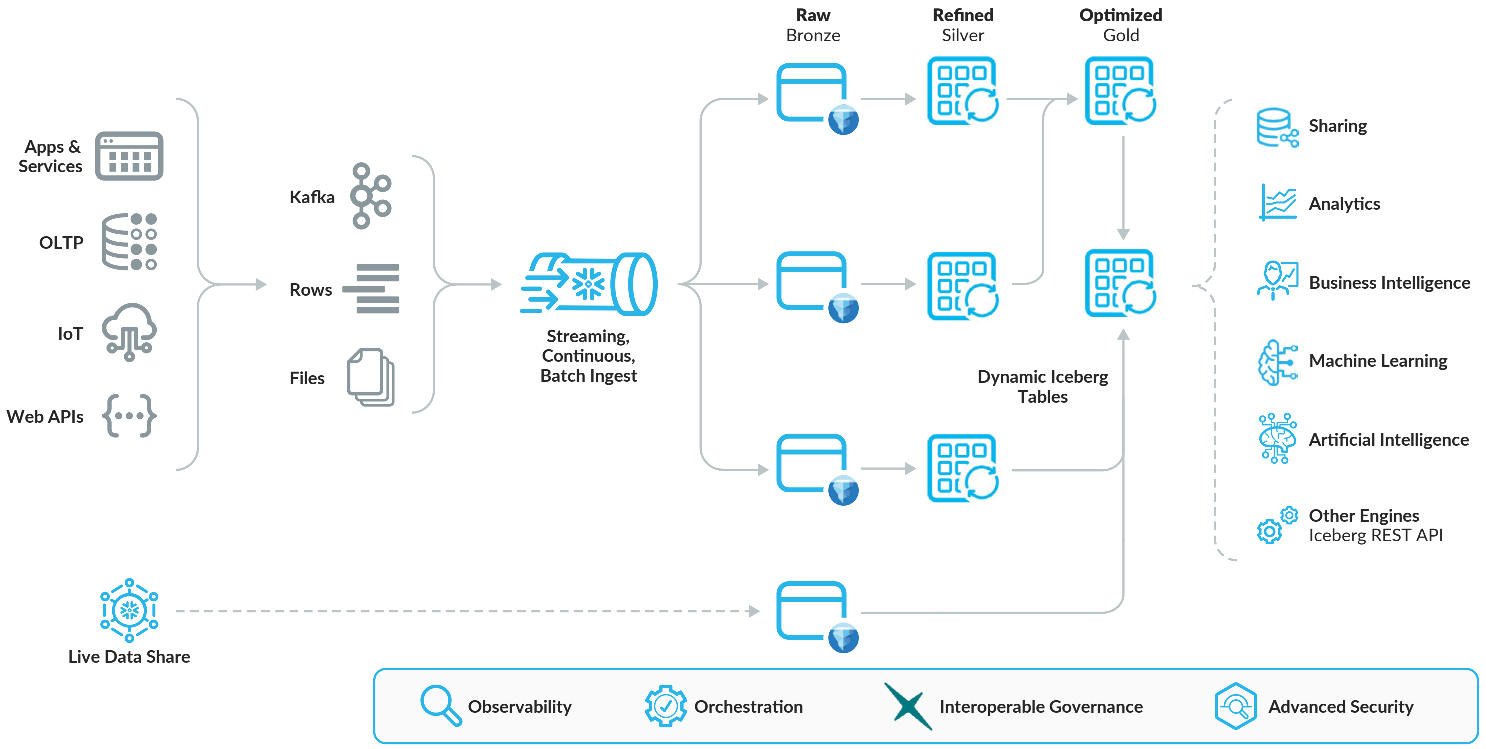
STREAMING AND REAL-TIME DATA PIPELINES

While most pipelines are sufficient using batch processing, streaming is a common solution for low-latency use cases such as real-time monitoring and personalization. While batch processing is typically completed in a matter of hours, streaming can make data available in minutes or even seconds. However, speed and cost are typically at odds when building a solution.

Snowflake provides multiple options for making real-time data available with less complexity and at lower costs. Rather than managing separate sets of streaming and batch infrastructure, Snowflake makes it possible to handle both, with support for loading to Iceberg tables. Snowflake's connector for Apache Kafka® enables streaming from Kafka topics directly to Iceberg tables. It's common in streaming use cases that updates to tables are made frequently, which can lead to the accumulation of many small files and result in extra metadata overhead, inefficient queries and extra storage costs. However, Snowflake-managed Apache Iceberg tables automatically handle **table optimization** in the background, including compaction, snapshot expiration and orphan file deletion.

For real-time analytics, streaming ingestion and automated transformations, data engineering teams often need to build incremental data pipelines. Traditional approaches using batch ETL jobs can be complex, resource-intensive and difficult to maintain, especially when dealing with frequent updates and large data sets. Snowflake Dynamic Tables simplify this process by allowing engineers to define declarative, auto-refreshing transformations using SQL, eliminating the need for manual ETL orchestration. Dynamic Tables support incremental updates, ensuring that transformed data remains fresh while optimizing compute costs. Additionally, they can write data in Apache Iceberg format, enabling access with tools such as Apache Flink®, Apache Spark™, Trino and more.

In addition, some of the extraneous costs of moving data in and out of your data lake can be avoided with Snowflake's secure, live data sharing.



BATCH AND CONTINUOUS INTEGRATION FROM DATA LAKES

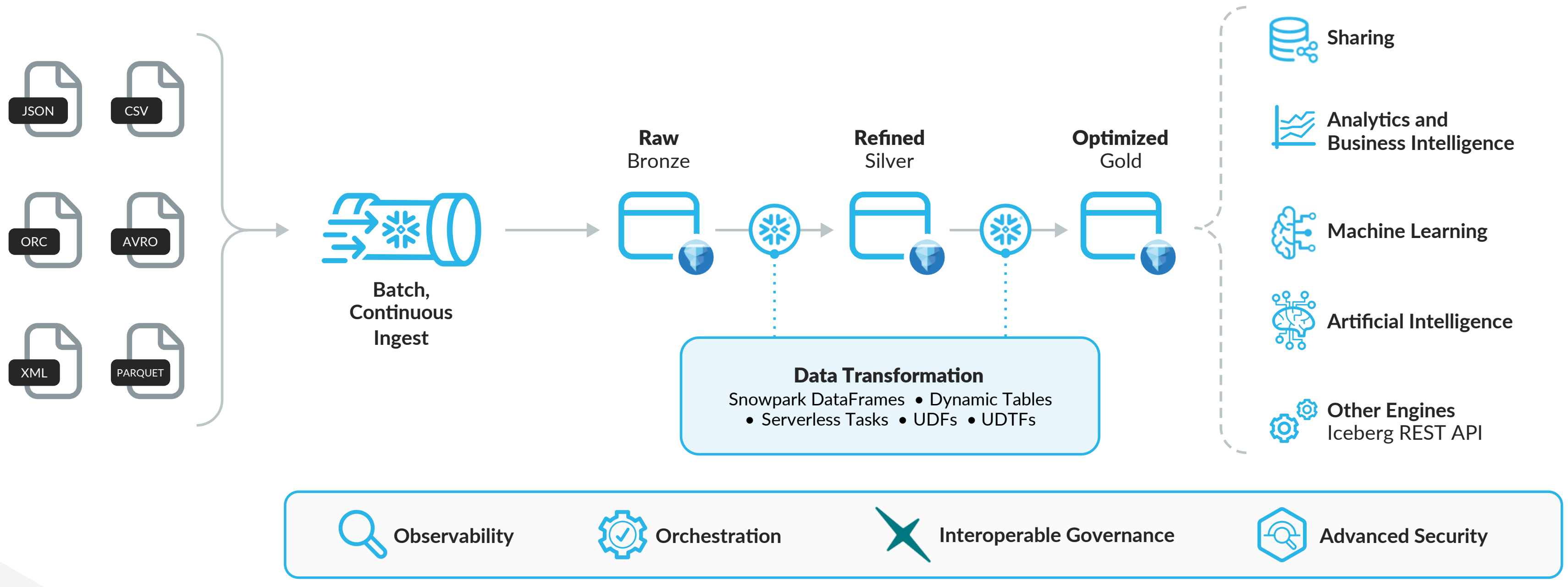
Perhaps your pipelines are already writing files to a data lake, and you're seeking a way to upgrade to a lakehouse architecture. There are three primary use cases for integrating data lake files with your Iceberg tables:

- **Loading files in a variety of formats**
- **Integrating Parquet without rewrite**
- **Continuous conversion from Delta Lake**

Loading files in a variety of formats

An appealing characteristic of data lakes is the ability to store data in a variety of formats, such as CSV, JSON, Parquet, XML, ORC and Avro. A data lake may even contain custom or industry-specific file formats, such as FHIR (Fast Healthcare Interoperability Resources), DICOM (Digital Imaging and Communications in Medicine) and HL7 (Health Level 7).

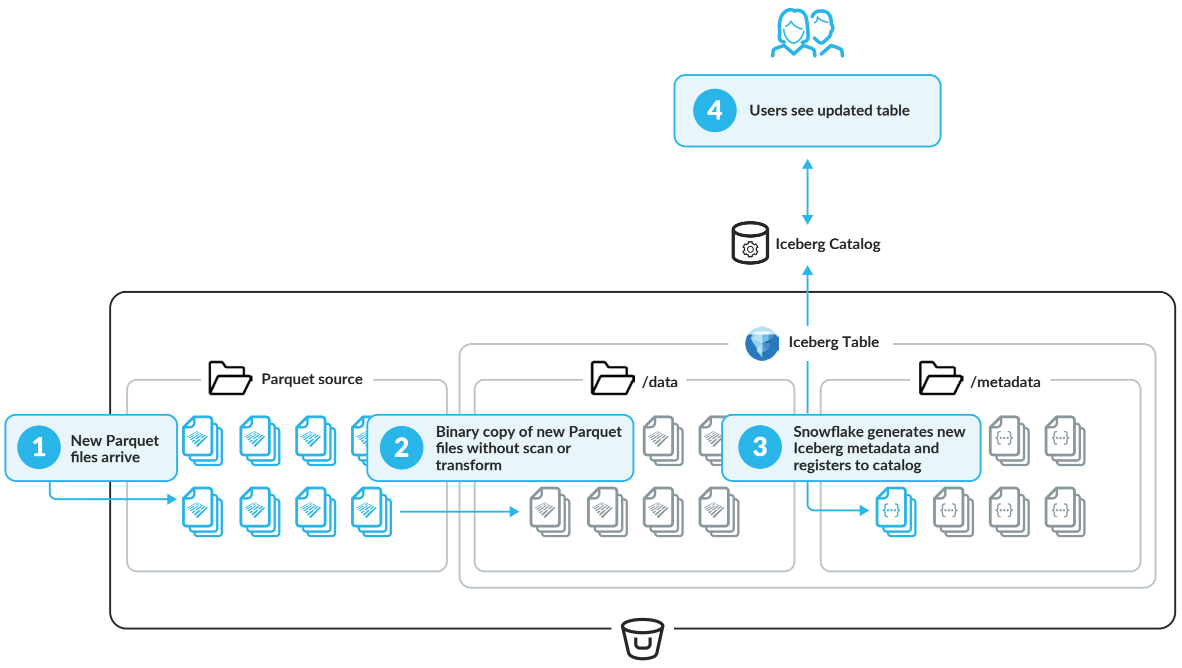
Snowflake can simplify the pipelines for loading these files into Iceberg tables — whether in batch or in a continuous, serverless manner. The platform renders file format conversions and file splitting with third-party tools as unnecessary: Simply run the data load with your source files, and Snowflake will automatically convert your source files into Parquet files optimized for your Iceberg table. Snowflake provides options for automatically detecting schema of the source files and evolving the Iceberg table schema, as well.



Integrating Parquet without rewrite

A very common columnar file format in data lakes is Parquet. Because Iceberg provides table metadata on top of data files, including Parquet, you may be able to simply register Parquet files to Iceberg tables rather than rewrite them. This is especially appealing for data lakes with vast quantities of Parquet files. In order for this approach to work, however, the Parquet files must use data types that are **compatible with Iceberg.**

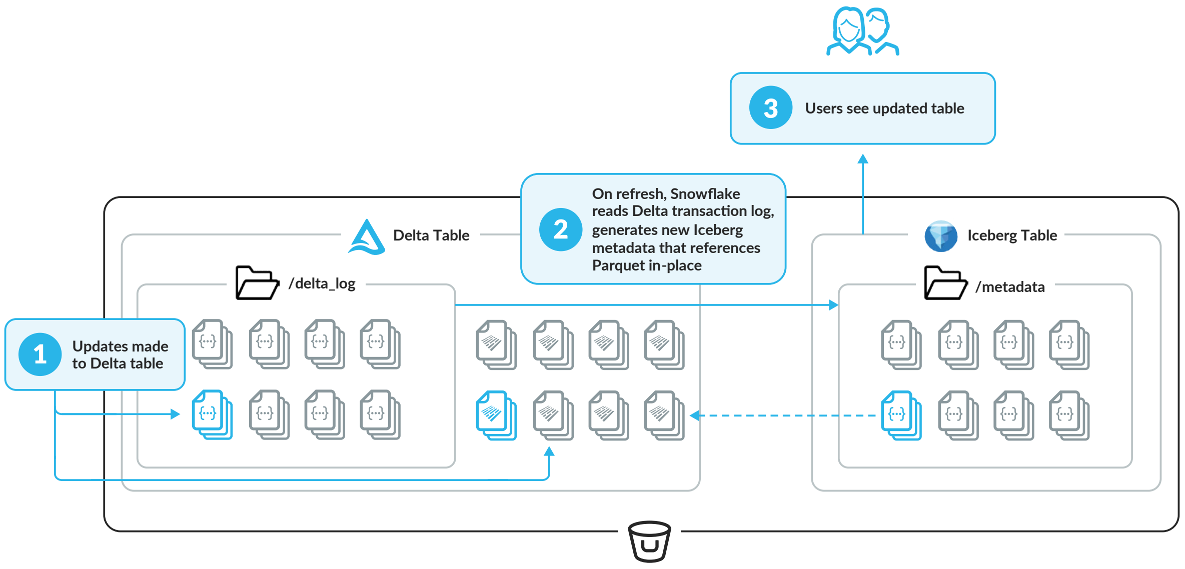
Snowflake provides an option to **load Parquet files into Iceberg tables** in this manner for scheduled batches, or continuously with Snowpipe, as new Parquet files land in the data lake. Instead of opening and reading the Parquet files, Snowflake simply binary copies them, exactly as is, from the existing data lake stage location to the Iceberg table's external volume location, where table data and metadata reside. Once all the files are in the Iceberg table's data folder, Snowflake will generate Iceberg metadata into the metadata folder and register the updates to the Iceberg catalog. This method is much faster and cheaper at loading, as files are not read; however, it is important to note that no transformations, filters or schema validations can occur.


4 Users see updated table
Iceberg Catalog
Iceberg Table
Parquet source
/data
/metadata
1 New Parquet files arrive
2 Binary copy of new Parquet files without scan or transform
3 Snowflake generates new Iceberg metadata and registers to catalog

Continuous conversion from Delta Lake

Though Iceberg, with its vast ecosystem and widespread interoperability, has gained popularity over time, there are still many legacy data lakehouses utilizing the Delta Lake table format. Snowflake's feature **Delta Lake Direct** bridges the two by converting Delta Lake tables into Iceberg format. Provide a location for the Delta transaction log, and Snowflake will interpret it to generate Iceberg metadata — meaning Snowflake and all of your tools supporting Iceberg can read these tables. Instead of rewriting or copying the Delta table's Parquet files, the Iceberg metadata created by Snowflake references those Parquet files in place. Delta Lake Direct from Snowflake provides the following benefits, as compared to other format conversion technologies:

- Source Delta Lake tables do not have to be registered to a proprietary Delta catalog
- Column mapping is not required for source Delta Lake tables
- There are no minimum Delta reader or writer versions for Delta Lake clients
- Any supported Delta Lake runtimes

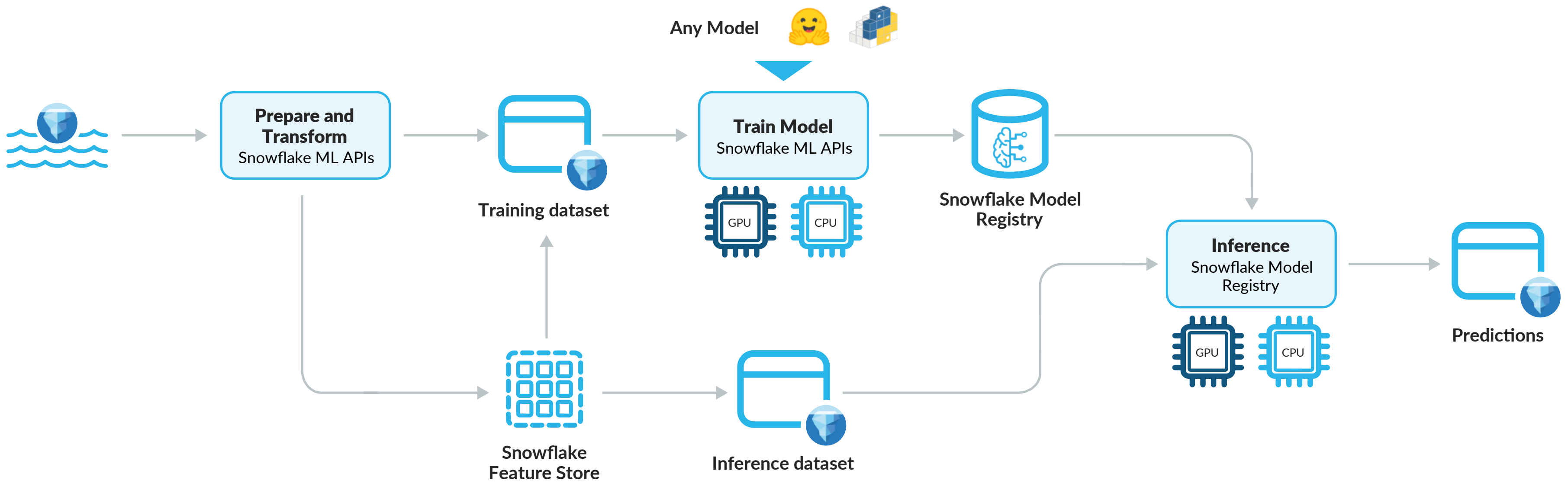

3 Users see updated table
2 On refresh, Snowflake reads Delta transaction log, generates new Iceberg metadata that references Parquet in-place
Delta Table
Iceberg Table
/delta_log
/metadata
1 Updates made to Delta table

MACHINE LEARNING

Machine learning teams need a platform to support their workflows, such as automating data preparation, to help ensure data quality and manage large data sets for model training and inference. Strong governance is crucial for tracking data and model usage and helping you ensure compliance with regulations. You also need tools that support data and model versioning, as well as scalable, real-time inference for production. Collaboration across teams is essential, as data engineers, data scientists and analysts all need access to the same data. The right platform streamlines the ML lifecycle, making projects more efficient and impactful.

Snowflake ML allows you to run large-scale training and inference jobs on top of your data lakehouse — no need to move data, create silos or lose auditability. Snowflake ML allows ML teams to primarily focus on model performance by abstracting away infrastructure configuration and management. Snowflake Horizon Catalog's governance controls are extended to Iceberg tables and models, making it easy to discover, manage and share models and features. Example use cases for Snowflake ML include:

- **Recommendation systems:** Product recommendation, customer targeting, next best action
- **Embedding models:** Product recommendation, RAG, similarity search
- **Computer vision:** Anomaly detection for defects, image classification for ecommerce, OCR analysis
- **Large-scale tree models:** Demand forecasting, churn prediction, lifetime value models



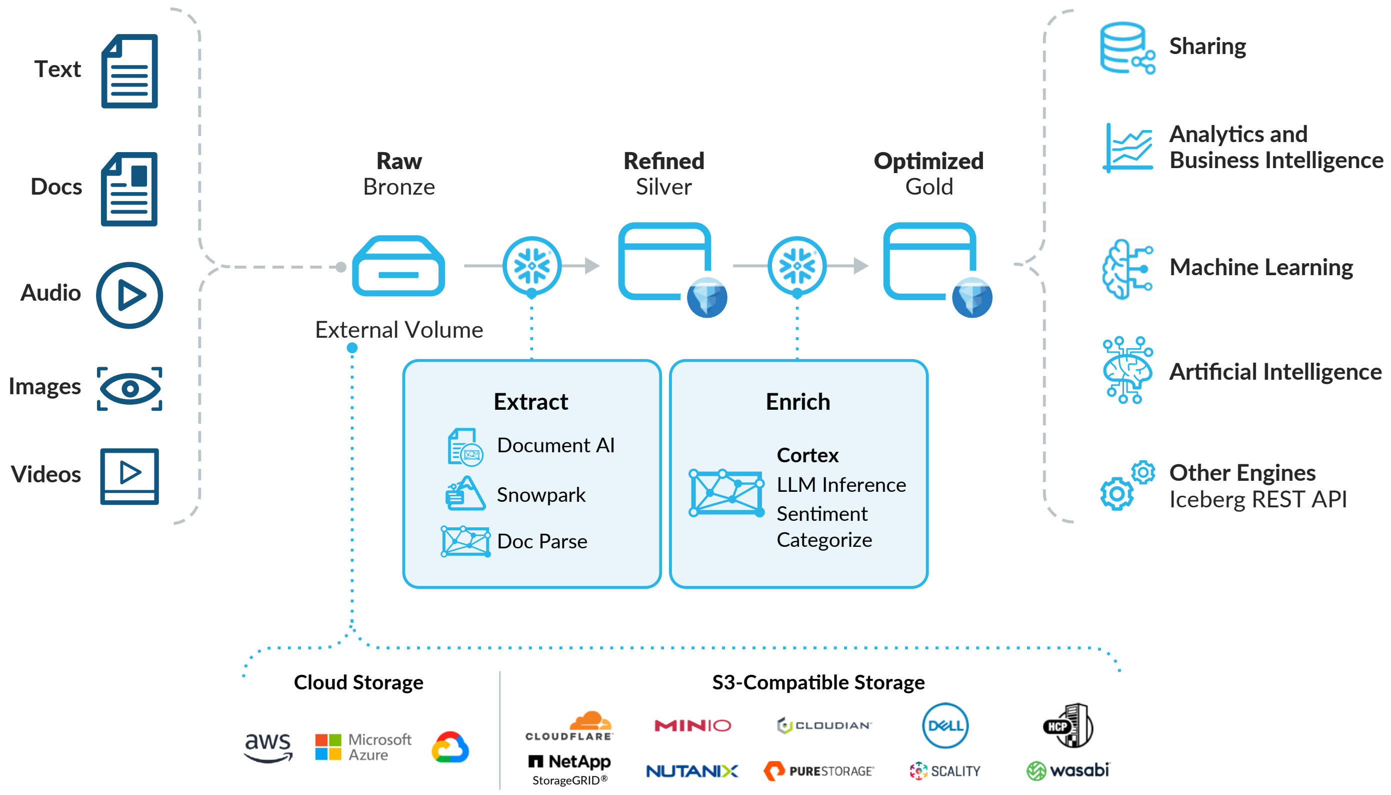
ARTIFICIAL INTELLIGENCE

Modern language models and AI products can aid the generation of insights across both structured and unstructured data. While data lake storage may be ideal for storing unstructured data, batch inference across thousands or millions of files is likely much more efficient when that data is transformed into table structure. In addition, the AI landscape is rapidly evolving, so flexibility to quickly and cost-effectively switch tools can help enable you to use the latest and best technology. A lakehouse can provide both performance and interoperability benefits in comparison to traditional data lakes.

A broad range of teams can focus on delivering value instead of managing compute with Snowflake's fully managed AI infrastructure for LLMs, RAG and text-to-SQL. Snowflake provides easy access to top-tier, cost-efficient AI models, frameworks and applications, which help you streamline the development-to-deployment lifecycle.

Unstructured data insights

Snowflake provides features such as Snowflake Cortex AI and Document AI that transform unstructured data into tables and enrich these tables with inference — storing all results in Iceberg format. Processing can be automated and performed incrementally to reduce costs.

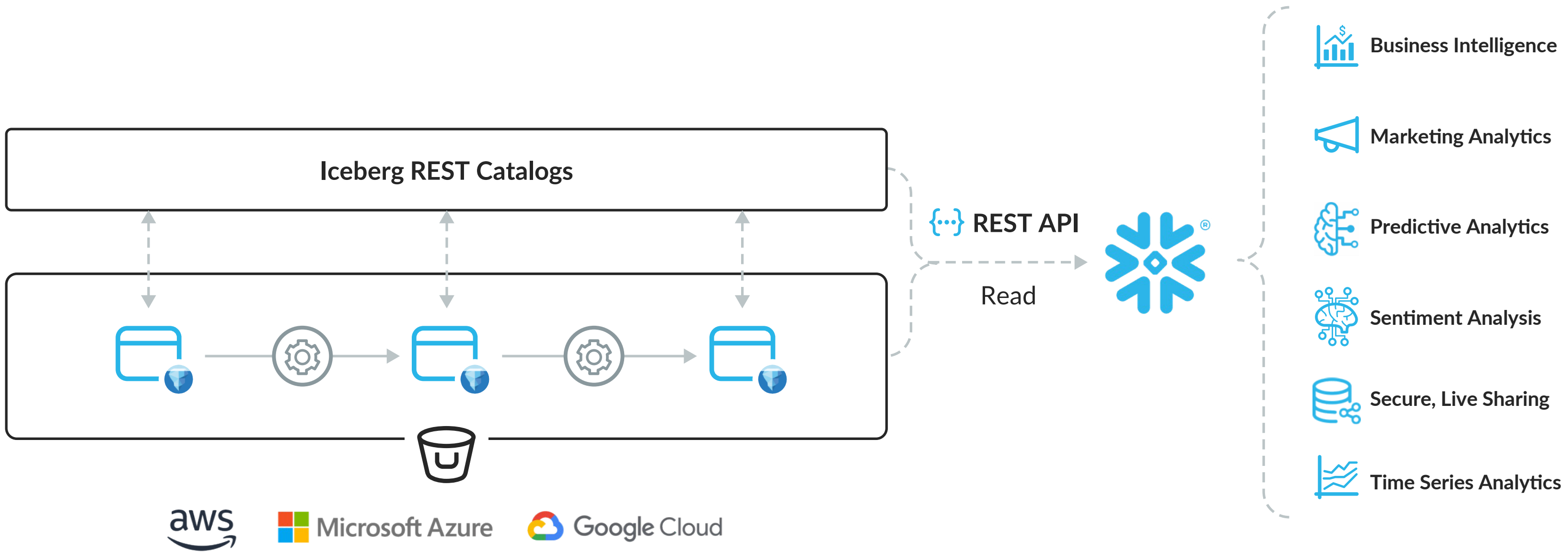


ANALYTICS

Improved performance for analytics is one of the primary benefits of Iceberg for data lakes. With added metadata for more efficient pruning, and no table locking to bottleneck readers, Iceberg enables much faster analytical performance on data lake files lacking a table format. Additionally, Snowflake provides critical benefits for analytics on Iceberg:

- **Performance with intelligent caching, elastic scaling and a query engine that's optimized for Iceberg and Parquet**
- **Support for a spectrum of analytical use cases with user-friendly SQL and out-of-the-box functions for time series, LLMs and more**
- **Flexibility to integrate with a variety of object storage services and Iceberg catalogs**

Apache Iceberg includes a standardized REST API for engines to interact with catalogs and tables, and catalog providers such as AWS Glue Data Catalog can support this API. Snowflake can integrate with a variety of these Iceberg REST catalogs, including AWS Glue.



A CATALOG FOR GOVERNED INTEROPERABILITY

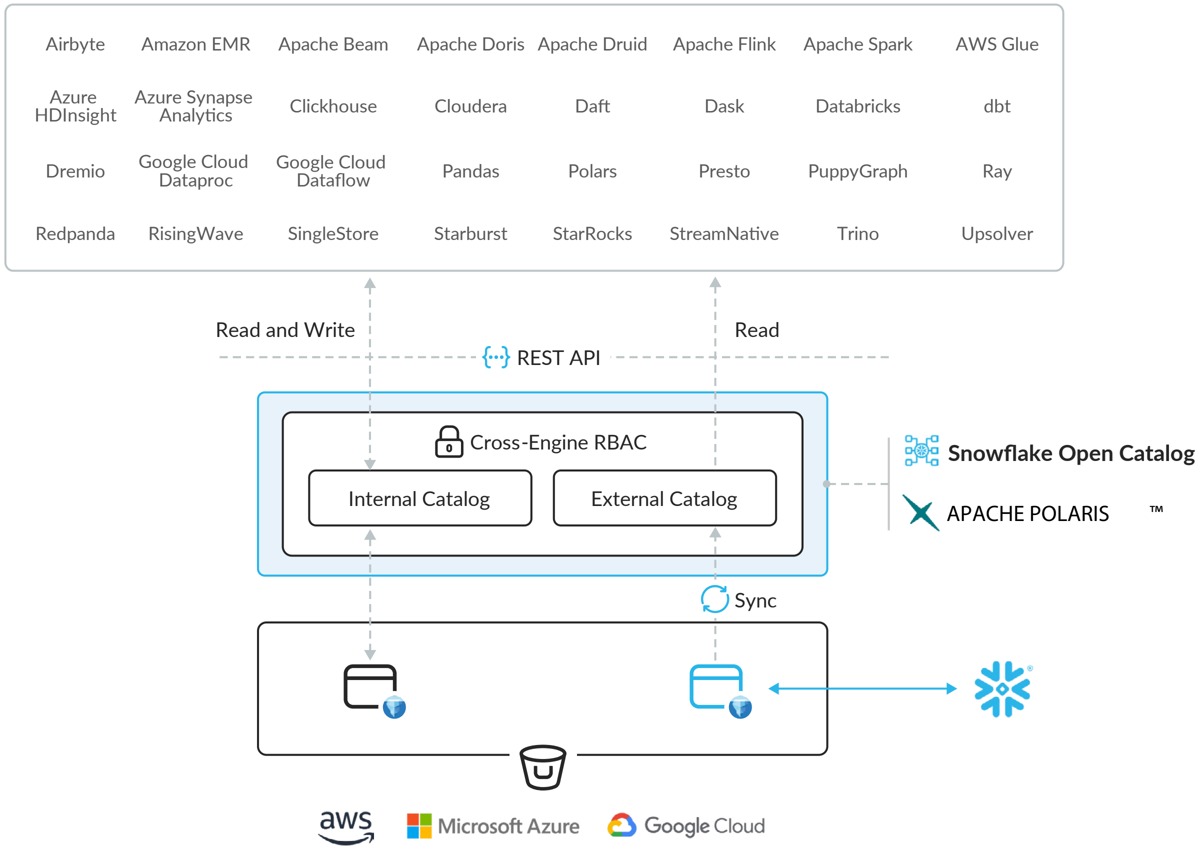
In a lakehouse architecture with Iceberg, a catalog serves as a central repository for managing metadata about data sets, tables and schemas. It tracks table versions atomically to help ensure consistency, and it may manage access controls, enabling data governance and security. An Iceberg catalog, which may also be referred to as a metastore, is different from other data catalogs you may be familiar with. Other data catalogs don't facilitate table transactions for write operations; instead they simply surface objects from external metastores and provide an access control plane.

The lakehouse architecture is built on open source standards at the storage level with open table and file formats. Because catalogs are the critical piece in a lakehouse architecture to unlock cross-engine operations and compatibility, open source standards are also necessary for both the catalog API and implementation for two key reasons:

- **An open standard API allows engines and catalogs to build support for the API once and unlock integration with many other tools.**
- **An open standard catalog implementation helps ensure compatibility of objects and access controls between different catalog service providers.**

Apache Polaris (incubating) is an incubating project with the Apache Software Foundation that supports Iceberg's standardized catalog REST API and provides a vendor-neutral, open source implementation. Because Polaris supports the Iceberg REST API, many different engines can be integrated for both reads and writes. Polaris unifies access controls for any engine that's connected, and follows security best practices by providing temporary scoped credentials.

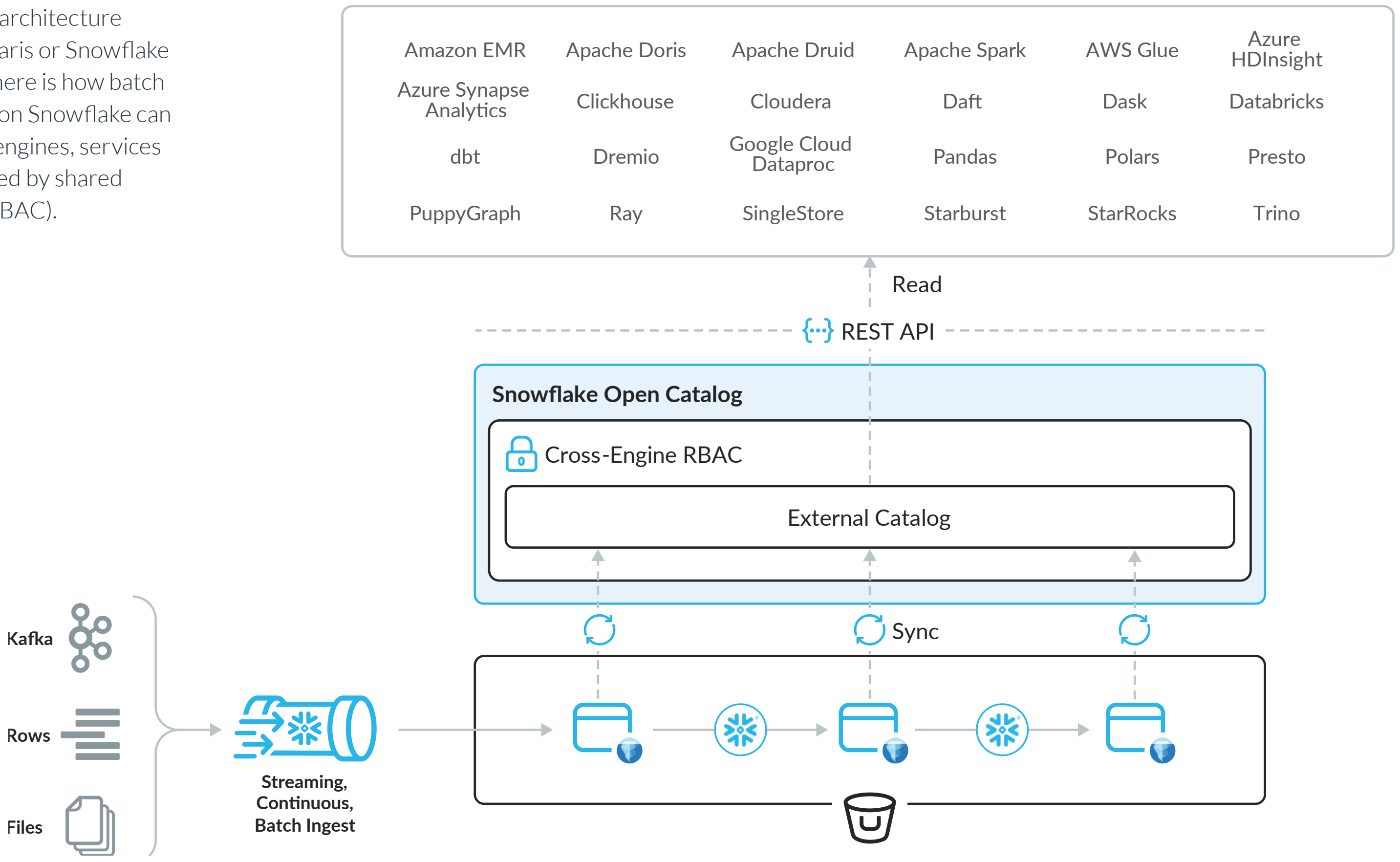
Snowflake Open Catalog, a managed service for Polaris, provides capabilities such as multi-factor authentication (MFA), availability in multiple clouds and regions, and a web interface to make it easier to use. That said, Polaris's open source implementation under Apache 2.0 license enables portability between a service and self-hosted options.



“Snowflake Open Catalog gives our global teams the flexibility to integrate all of our tools in one place [...] while maintaining the unified governance we depend on to effectively manage our open data lakehouse. Snowflake’s commitment to true open source without vendor lock-in gives us the confidence to innovate faster, without having to worry about the complexities of setup, maintenance and updates for our lakehouse strategy.”

— VINEET GORHE,
Chief Technology Officer, DemandHelm

Many of the aforementioned architecture patterns can be built with Polaris or Snowflake Open Catalog. For example, here is how batch and streaming pipelines built on Snowflake can be read by a variety of other engines, services and frameworks — all governed by shared role-based access controls (RBAC).



MIGRATION PLANNING

When you are planning to migrate your data lake storage from either on-premises Hadoop or a cloud-based object store with open file formats such as Apache Parquet, a well-defined strategy and comprehensive plan are essential for success. Snowflake's migration framework stands as a tried-and-tested methodology, recommended by Snowflake Professional Services to help customers create a resilient blueprint. This framework is designed to support the success and cost-effectiveness of the migration process for data lake workloads to the Snowflake platform.

SNOWFLAKE MIGRATION FRAMEWORK

Snowflake's migration framework consists of four phases, with clear outcomes defined for each phase.

1. **Discovery:** Collect high-level inventory and define a business case for migration.
2. **Assessment and planning:** Collect a complete set of impacted components, define future state architecture and plan migration strategy.
3. **Delivery:** Migrate, validate and deploy data and code.
4. **Optimization and modernization:** Optimize and re-engineer workloads to unlock additional benefits.

	DISCOVERY	ASSESSMENT AND PLANNING	DELIVERY	OPTIMIZATION AND MODERNIZATION
DESCRIPTION	Initial discovery guides migration recommendations	Deeper assessment will determine full requirements and scope, including use case qualification, migration plan and implementation approach	Execution of the overall migration includes code conversion, data migration, orchestration, deployment, validation and consumption	Complete the migration with optimization as needed, as well as modernization and implementation of new use cases on Snowflake
ACTIVITIES	• Discovery • Collection inventory • Strategy workshops	• Migration Readiness Assessment (MRA) • Detailed discovery and scoping • Definition of future-state architecture • Migration approach	• Data migration/ conversion • Data validation • Application code conversion • Data consumption conversion • Testing and go-live	• Optimization • Modernization and re-engineering • Development of new use cases
OUTCOME	• Well-defined business case • High-level inventory	• Full inventory of object/ code in scope • Migration plan and timelines	• Data validated and performance benchmarked • Workload deployed to production	• Implementation of new use cases

DISCOVERY

First, collect information about existing data lake architecture: the technologies used and the types of data stored and used. Information collected in this phase helps identify high-level dependencies and migration scale. The information is also used to define the business case for the migration.

ASSESSMENT AND PLANNING

Dive deeper into the existing data lake architecture to collect information on the details in the table to understand the current landscape and define the future-state architecture on Snowflake.

DETAIL	EXAMPLES
File Format	Parquet, Avro, ORC, JSON
Object storage location	On premises, cloud
Storage system	Amazon S3, Azure Data Lake Storage Gen 2, Cloudflare R2, HDFS
Data Volume	Number of tables, file size
Workload types	Batch processing, stream processing, machine learning, business intelligence
Query and processing engines	Apache Spark, Presto, Trino, Amazon Athena, Hive
Catalogs and metastores	Hive Metastore, AWS Glue Data Catalog
Security and governance requirements	AES 256-bit encryption, row- and column-level security, PCI-DSS

These requirements influence the ideal storage patterns used in the future-state architecture: Iceberg tables or Snowflake native tables. For example, if you require many query engines or data to be in storage that you manage, then Iceberg is likely the recommended table format. The details gathered on code, data inventory and approach should be used as inputs to define a migration plan with timelines.

DELIVERY

In this phase, the output plan from the previous stage is executed, which may require both conversion of storage formats and refactoring of pipelines. In the planning phase, you will have decided on an architecture, including a catalog or metastore. Using the same architecture patterns described earlier as examples, let's explore what steps are involved in delivery.

Storage migration

Moving files to cloud storage
If you are migrating the data from on-premises Hadoop, the first step is to move the underlying data files from HDFS to cloud object storage using either direct copy or a migration service like AWS Snowball or Azure Data Box. This step can be skipped if you are migrating from a cloud data lake.

Full conversion to native Snowflake tables or Snowflake-managed Iceberg tables (data file rewrite)
Once the files are on cloud or S3-compatible object storage, you can create external tables pointing to the files from storage. To help accelerate the migration, Snowflake's Professional Services team can help export the table definitions from a current catalog, such as Hive Metastore, and convert to Snowflake-compatible table definitions automatically. Documentation provides instructions for creating an external volume, which is needed prior to creating an Iceberg table in Snowflake. Once the tables are defined and created, you can use existing Snowflake ingestion methods such as COPY INTO, Snowpipe and Create Table as Select (CTAS) from external tables to load data into native or Snowflake-managed Iceberg tables. Once tables are fully loaded and validated for completeness, the files in the original storage location can be removed.

In-place conversion from Parquet to Iceberg (no data file rewrite)
When creating an Iceberg table for Snowflake, the underlying data file structures and partitions are reused and only the Iceberg metadata is written. This approach is generally faster and cheaper up front, and can be leveraged if the current partition structure of files is efficient and does not need to be restructured. You can simply point Snowflake to a repository of Parquet files, the location for where the Iceberg table should be written, and Snowflake will generate Iceberg metadata. As part of its Apache Spark runtime package, Iceberg also provides a few procedures that can be used to migrate files using a similar approach. While it is a best practice to not run any process that can modify the underlying files while the conversion is in progress, any new files not captured as part of the in-place conversion can be added using the add_files procedure, which is also available as part of the Iceberg Spark runtime package.

Convert existing Iceberg tables to Snowflake-managed Iceberg table
If you have data that is already in the Iceberg table format, managed by an external catalog, and now want full read-and-write support on Snowflake, you can use a simple, one-line SQL command to convert the table's catalog without rewriting any underlying files. This approach is useful if you have been using Snowflake as a read-only consumption layer and now want to fully manage the table and use Snowflake for data loading and transformation workloads. As a best practice, hold off on changes to data when the table is being converted to avoid data-corruption issues.

	LEGACY DATA LAKE	EXTERNALLY MANAGED OPEN DATA LAKEHOUSE	SNOWFLAKE-MANAGED OPEN DATA LAKEHOUSE
Storage	• HDFS • Amazon S3 • ADLS • GCS	• Amazon S3 • ADLS • GCS • S3-compatible storage	• Amazon S3 • ADLS • GCS
File Format	• Parquet • Avro • ORC • JSON • XML • CSV	• Parquet	• Parquet
Table Format	• Hive • Delta Lake • None	• Iceberg	• Iceberg
Metastore / Catalog	• Hive Metastore • AWS Glue Data Catalog	Iceberg REST catalogs: • AWS Glue Data Catalog • Apache Polaris • Nessie • Snowflake Open Catalog • Unity Catalog	• Snowflake Horizon Catalog • Snowflake Open Catalog
Data Processing	• Spark • Presto • HiveQL • Impala	Snowflake (read-only): • SQL • Snowpark (Python, Java, Scala) • Cortex LLM Functions Iceberg REST compatible engines: read and write	Snowflake (read and write): • SQL • Dynamic Iceberg Tables • Snowpark (Python, Java, Scala) • Cortex LLM Functions • Snowpipe • Snowpipe Streaming Iceberg REST compatible engines: read-only
Table Maintenance	Self-managed	Self-managed	Snowflake performs automatically

Application migration

Application migration is the process of modifying the processes that read and write data in order to be compatible with your chosen storage formats. This includes ingestion, transformation and consumption pipelines.

Migrating Spark to Snowpark
While Apache Spark can be used to operate on externally managed Iceberg tables and read Snowflake-managed Iceberg tables, Spark pipelines can be easily migrated to Snowpark for performance improvements and cost efficiencies. Snowpark is a set of libraries and runtimes in Snowflake that enable developers to securely process non-SQL code — including Python, Java and Scala — in Snowflake's elastic processing engine without having to move data. Snowpark DataFrames have high compatibility with Spark DataFrames, allowing easy migration of Spark code to Snowpark with minimal effort.

Tools such as the Snowpark Migration Accelerator (SMA) will speed the migration planning and code conversion processes. The tool parses the source Spark code to identify all Spark APIs and map them to equivalent Snowpark APIs. The tool also introduces helper functions through automation for known gaps between Spark and Snowpark.

You can find resources — including hands-on labs, guides and customer stories on migrating Spark to Snowpark — here.

Migrating to Snowflake SQL
If you are managing the data processing pipelines through Hive, Presto, Trino, Impala or other SQL engines, they can be migrated to be Snowflake-compatible SQL that writes data to Snowflake-native or Snowflake-managed Iceberg tables. For client-based applications, such as business intelligence tools, you can use Snowflake's JDBC and ODBC drivers to connect to Snowflake and query tables.

Data validation
As it is perhaps the most important phase in a migration project, we recommend planning migration testing in two parts:

1. **Unit tests:** These tests can be performed with a small subset of data. The idea is that the workload can be run from start to finish to validate that the general logic is equivalent and confirm basic verification of the applied changes.
2. **User acceptance tests:** These tests are more complex because they require running the migrated workload end to end with an environment close to production. This is so it can be benchmarked to determine critical success factors, such as whether target data row counts, data consistency checks or workload total execution times are within tolerances.

Leveraging automation tools to assist with these types of tests will help ensure the migration is accomplished within planned timelines and with high quality.

OPTIMIZATION AND MODERNIZATION

The primary objective of this phase is to remove any inefficiencies that may have been carried over as part of the migration from legacy architecture and to unlock new use cases and capabilities that Snowflake's platform can support. Potential areas of modernization include these:

- **Ingestion and data transformations (ETL/ELT pipelines):** If an external processing engine such as Spark is used for ingestion or data transformation workloads (e.g., batch, streaming), they can be re-engineered to run within Snowflake using Snowpark. With Snowpark, you can use Python, Java or Scala to write more efficient data pipelines and avoid the overhead of managing additional infrastructure and security requirements.
- **AI and ML workloads:** With the data stored as Iceberg tables, you can leverage Snowflake to build end-to-end ML pipelines. With Snowpark ML, you can quickly build features, train models and deploy them into production — all using familiar Python syntax and without having to move or copy data outside Snowflake's security boundary.

SUMMARY AND NEXT STEPS

Modernizing your data lake with Snowflake could help you:

- **Increase productivity and collaboration on data**
- **Reduce security and compliance risks by leveraging enhanced governance features**
- **Reduce total cost of ownership**

Along with our service partners, Snowflake is available to assist with your modernization initiative by helping you:

- **Understand your existing data lake landscape**
- **Quantify the business case of migration**
- **Develop a future-state reference architecture**
- **Create an implementation plan**
- **Execute migration**
- **Optimize and onboard new workloads**

Please **contact us** if you are interested in exploring how Snowflake can help modernize your data lake.

More resources available at **Snowflake for Data Lakehouse**.

ABOUT SNOWFLAKE

Snowflake makes enterprise AI easy, efficient and trusted. Thousands of companies around the globe, including hundreds of the world's largest, use Snowflake's AI Data Cloud to share data, build applications, and power their business with AI. The era of enterprise AI is here.

Learn more at **snowflake.com** (NYSE: SNOW)

ebook-modernizing-to-a-data-lakehouse-with-snowflake-04