Brought to you by:

CLOUDERA

Migrating to Apache® Iceberg

for dummies®
A Wiley Brand

The business case for migrating to Iceberg

A step-by-step blueprint to migration

Power AI and analytics with interoperability

Cloudera®
Special Edition

Stephanie Diamond
Akshat Mathur
Foreword by Bill Zhang

About Cloudera

Cloudera is the only data and AI platform company that large organizations trust to bring AI to their data anywhere it lives. Unlike other providers, Cloudera delivers a consistent cloud experience that converges public clouds, on-prem data centers, and the edge, leveraging a proven open-source foundation. As the pioneer in big data, Cloudera empowers businesses to apply AI and assert control over 100% of their data, in all forms, improving security, governance, and real-time and predictive insights. The world's largest brands across all industries rely on Cloudera to transform decision-making and ultimately boost bottom lines, safeguard against threats, and save lives. Learn more at cloudera.com.

Migrating to Apache® Iceberg

Cloudera® Special Edition

by Stephanie Diamond and Akshat Mathur

FOREWORD BY **Bill Zhang**

for dummies®
A Wiley Brand

Migrating to Apache® Iceberg For Dummies®, Cloudera® Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-394-41282-2 (pbk); ISBN 978-1-394-41283-9 (ebk); ISBN 978-1-394-41284-6 (ebk)

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Development Editor/Project Manager: Rebecca Senninger

Acquisition Editor: Traci Martin

Editorial Manager: Rev Mengle

Business Development Representative: Jeremith Coward

Production Editor: Umeshkumar Rajasekhar

Cloudera team: Shaun Ahmadian, Matthew Burgess, Paul de Fusco, David Dichmann, Jessica Espinoza, Valerie Glover, Jeff Healey, Nijjwol Lamsal, Ramesh Mani, Dipankar Mazumdar, Pamela Pan, Ron Pick, Navita Sood, Agnes Tevesz, Stefan Tzanev

Table of Contents

Foreword

Data has always been the backbone of innovation, but how we manage and use it is changing. The last decade focused on collecting and storing data. The next one will emphasize mobilizing it, making it seamless, secure, and smart.

At the core of this change is Apache Iceberg, the open table format that is reshaping how modern organizations organize, govern, and share data. Iceberg brings order and reliability to the chaos of data lakes. It transforms raw files into governed, query-ready tables that can support analytics and AI anywhere. It is doing for data what Kubernetes did for compute by establishing a universal, open standard that allows every system to communicate effectively.

At Cloudera, we view Apache Iceberg as the foundation of the open data lakehouse era. Our goal has always been to provide enterprises the freedom of open source combined with the confidence of enterprise-grade governance. Iceberg represents this balance: openness without fragmentation, performance without lock-in, and governance without compromise.

We believe the future of analytics and AI will rely on open table formats. These formats create the connections that link data producers and consumers across different systems, clouds, and regions. Iceberg's REST-based interoperability, its detailed metadata control, and its rapidly growing ecosystem are speeding up the transition to truly federated, self-service data architectures. In these architectures, every workload, from SQL to LLM training, runs on a uniform data foundation.

Cloudera is proud to be one of the early adopters and key contributors advancing Iceberg. We are helping enterprises move from outdated table formats to this new standard of openness and intelligence. For us, Iceberg is more than just a technology; it is a belief that open innovation prevails and that the next decade of data and AI will be driven by shared standards rather than proprietary limits.

The move from Hive to Iceberg is more than just a step in modernization. It marks the start of a movement toward a world where data is portable, interoperable, and ready for AI. That's the future we are creating at Cloudera, one Iceberg table at a time.

Bill Zhang

Vice President, Product Management

Cloudera

Introduction

Data lakes have come a long way since the days of monolithic, batch-driven pipelines. Apache Hive laid the groundwork for scalable analytics, but its architecture was never designed for today's demands, such as real-time analytics and AI model training. Query planning takes minutes when you need milliseconds. Schema changes in Hive table formats require risky rewrites that disrupt downstream consumers. Concurrent updates risk data corruption. Your teams create duplicate copies of the same data because different engines can't safely share tables.

Apache Iceberg changes this entirely. Iceberg is an Apache-licensed open table format designed for the speed and flexibility that modern businesses require. It eliminates Hive's architectural bottlenecks while delivering capabilities that seemed impossible just a few years ago. You get instant schema evolution without rewrites, time travel to any historical snapshot, multi-engine Atomicity, Consistency, Isolation, Durability (ACID) transactions, and consistent governance across hybrid environments.

What makes Iceberg particularly powerful is its open, vendor-neutral design. Unlike proprietary alternatives that lock you into specific platforms, Iceberg works consistently across on-premises data centers, private clouds, and any cloud object store like AWS S3. Your data remains portable. Your architecture stays flexible. Your teams maintain control. Regardless of your role, Iceberg delivers measurable value.

This book walks you through the why and the how of migrating from Hive table format to Iceberg. You'll see what makes Iceberg different, learn how to plan a smooth migration, and discover how to unlock business benefits, including faster queries, zero-copy data sharing, and reproducible analytics that serve as the foundation for enterprise AI at petabyte, and larger, scale.

About This Book

Welcome to *Apache Iceberg Migration For Dummies*. This book covers the essential steps for modernizing your data lake with Iceberg, from understanding the basics to building a business case and executing a successful migration.

We cover several topics:

- Learning what Iceberg is and how it solves the limitations with the Hive table formats
- Executing migration from Hive to Iceberg table format with a blueprint for readiness, governance, and timelines
- Modernizing data engineering with Apache Spark, streaming with Apache Flink and BI or AI workloads with Trino, all using Iceberg tables
- Enforcing governance with Cloudera Octopai Data Lineage for tracking and Apache Ranger for access policies, while keeping tables healthy with automated compaction and snapshot management
- Designing deployment patterns across data centers, private clouds, and Object Stores like AWS S3 without vendor lock-in
- Reviewing ten clear ways Iceberg migration saves money, reduces risk, and future-proofs your data strategy

Icons Used in This Book

Throughout this book, different icons are used to highlight important information. Here's what they mean:

TIP

The Tip icon highlights information that can make doing things easier or faster.

REMEMBER

The Remember icon highlights items you need to recall when searching your memory bank.

WARNING

The Warning icon alerts you to potential hazards that could harm you or your company.

TECHNICAL STUFF

Sometimes, we provide you with a few additional tidbits of research or facts beyond the basics. If you'd like to know the technical details, watch out for this icon.

Beyond the Book

This book gives you a practical foundation for understanding and planning your Iceberg migration. We have posted all the code examples at `https://github.com/cloudera/iceberg-migration-for-dummies`. If you'd like to dig deeper into Cloudera's hybrid data strategy, AI integration, and more, visit `https://cloudera.com`.

IN THIS CHAPTER

» Migrating from Hive table format to Iceberg table format

» Exploring Iceberg's features

» Building the business case

Chapter 1

Understanding Apache Iceberg and Migration Strategy

Apache Iceberg is an open table format that modernizes data lakes. It supports significantly greater workload flexibility and scalability compared to Apache Hive tables, while introducing advanced capabilities that make your enterprise data AI-ready.

Migrating from Hive to Iceberg table format isn't just a technical upgrade. It's a strategic modernization that boosts your performance, strengthens governance, and unlocks new AI and multi-function analytics use cases.

In this chapter, you learn what Iceberg is, how its architecture works, why it meets today's needs more effectively than Hive table format, and how to build the business case for migration.

Defining Apache Iceberg and Why It Matters

At its simplest form, a table format is the layer of logic that organizes raw files in your data lake into a usable table. Without it, a folder of Parquet or ORC data files (columnar storage formats that organize data by column rather than by row, enabling efficient compression and query performance) is just a collection of unrelated files with no schema or organization. Table formats add metadata and rules that make data sets discoverable and queryable by database or other data processing engines.

Apache Iceberg is the next-generation open table format. It provides a standard, open way to manage large analytic tables in data lakes. Unlike the Hive table format, which was designed for a batch-first era, Iceberg was built for the scale, concurrency, and agility required by AI and streaming analytics. Iceberg gives you:

- **Atomic transactions with snapshot isolation:** Updates happen atomically. Either all changes succeed or none do, preventing data corruption during concurrent operations.
- **Schema and partition evolution without rewrites:** You can modify table schemas, including nested fields, without rewriting existing data or breaking existing queries. This includes adding columns, renaming fields, or changing partition strategies.
- **Version control and time travel:** Every change creates a new snapshot, enabling you to query or restore data from any previous state. This supports model reproducibility (recreating exact training datasets for AI), debugging (investigating what changed and when), and compliance audits (proving data state at specific points in time).
- **Open and engine-neutral interoperability:** Iceberg works seamlessly across Apache Spark, Apache Hive, Apache Flink, Apache Impala, Trino, and Presto, with consistent behavior regardless of which engine you choose. Apache Iceberg enables multiple engines like Spark, Flink, and others to work seamlessly on the same data, unlocking the ability to do AI/ML and multi-function analytics on the same piece of data, at the same time. You can use the right tool for every job without copying data.

Understanding Iceberg fundamentals and architecture

Hive table format stored metadata in the Hive Metastore, but query planning still depended on directory scans (physically listing files in storage to determine which data to read). Iceberg uses a modern architecture for the metadata layer, managing metadata separately from the files themselves, using a layered architecture that scales efficiently.

Iceberg's architecture has three layers:

- **Catalog:** This is the source of truth for table metadata. It maintains the current state of the table and coordinates read and writes across engines.
- **Data files:** These are raw, columnar storage files, typically Parquet, ORC, or Avro formats. Your existing data files can be reused in many migration scenarios. Iceberg works with the file formats you're already using. However, Iceberg requires its own metadata layer to be created, which is why migration is necessary even when keeping the same physical files.
- **Metadata layer:** The metadata layer achieves ACID guarantees, concurrency control mechanisms, and schema definition. It comprises schema info, partition info, transaction logs (snapshots), and statistics. Iceberg tracks which files contain relevant data, eliminating full directory scans. This is not just an index. Understanding this Iceberg architecture is critical for the migration process, the metadata and manifest files determine how data is organized and accessed, helping you make informed migration decisions.

TIP

Instead of scanning an entire directory to find relevant files, Iceberg consults metadata that knows exactly which files contain relevant rows. This makes query planning faster, especially for large tables.

This architecture delivers several critical advantages:

- **Scalability:** Iceberg handles petabyte-scale, or larger, tables with billions of files without performance degradation. The metadata layer scales independently from data volume.

- **Consistency:** Queries always see a stable snapshot, even when concurrent writes are happening. Snapshot isolation prevents the read anomalies that can occur with Hive's directory-based approach. Writes always follow the Optimistic Concurrency Control model (OCC), so only one writer wins.
- **Flexibility:** Multiple engines can safely share the same table without coordination overhead. Each engine interacts through the same metadata layer, ensuring consistency and uniformity across all systems.
- **AI-readiness:** Snapshots provide reproducibility for training datasets and enable comprehensive audits. Data scientists can recreate exact training sets. On top of that, Branching capability enables isolated experimentation environments.

Figure 1-1 shows Apache Iceberg's layered architecture.

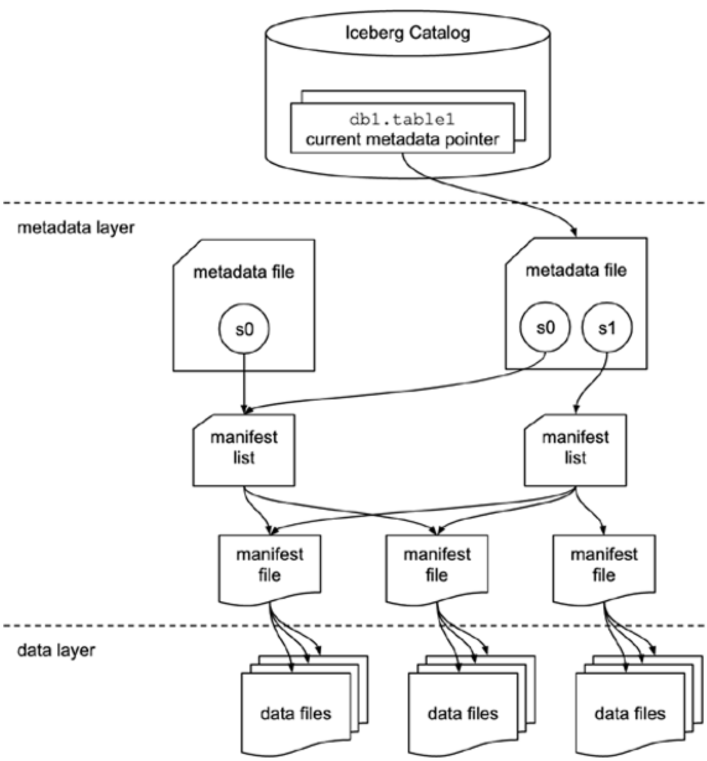

FIGURE 1-1: Apache Iceberg architecture.

Positioning Iceberg for a competitive advantage

Several characteristics distinguish Iceberg from alternative approaches and make it particularly valuable for enterprise deployments:

- **Open, not proprietary:** Some table formats tie enterprises to specific vendors or cloud platforms. Iceberg is Apache-governed with broad industry adoption, ensuring no vendor lock-in and long-term portability. When your strategy changes, your data remains accessible.
- **Hybrid cloud flexibility:** Iceberg is being used in both cloud and on-premises environments. Enterprises can use it competitively with common governance and metadata management across both environments. Cloudera integrates Iceberg for consistent governance and security across environments, enabling several deployment patterns that pure-cloud or pure-on-premises solutions cannot match.

  TIP

  Enterprises can keep sensitive data on-premises for compliance, burst workloads to the cloud without rewriting pipelines, and run a single security and governance model everywhere. This flexibility is particularly valuable for organizations with complex regulatory requirements or existing infrastructure investments.
- **Multi-team empowerment:** With traditional data warehousing architecture, BI teams, data scientists, and data pipleines, developers often create redundant copies of the same dataset to meet their specific needs. Iceberg enables one governed table to serve all users simultaneously, reducing duplication and ensuring everyone works with consistent data. Cloudera comes pre-integrated with best-of-breed engines to operate on Iceberg for all these different users to collaborate with the tools of their choice.

Figure 1-2 illustrates Cloudera's open data lakehouse architecture.

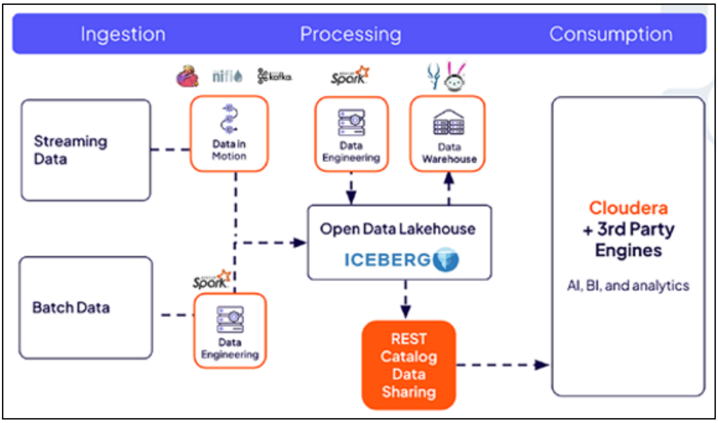

FIGURE 1-2: Cloudera's open data lakehouse architecture.

Comparing Features and Building AI Infrastructure

The most effective way to understand Iceberg's value is through direct comparison with the Hive table format; many are shown in Table 1-1. While the Hive table format has served organizations well for many years, its design choices reflect assumptions that no longer align with modern requirements.

TABLE 1-1 Hive versus Iceberg Feature Comparison

Feature	Hive Tables	Iceberg Tables
Updates	Rewrite partitions	Atomic file-level snapshots; support fine-grained updates
Concurrency	Risk of corruption with concurrent writes	Safe multi-write transactions with ACID guarantees and Optimistic Concurrency Control (OCC)
Partitioning	User-maintained partition columns; users must understand and actively manage the physical table layout, making partition design and management decisions manually	Hidden partitioning Iceberg automatically applies transforms (day, bucket, truncate) without user configuration. Enables flexible partition evolution

Feature	Hive Tables	Iceberg Tables
Schema changes	Risky and disruptive; requires table rewrite in many cases	Safe evolution allows add, rename, drop columns, even nested fields
Time travel and version control	Not supported	Fully supported out-of-the-box
Engine support	Primarily Hive engine (Apache Spark, Trino, Apache Flink, Apache Impala, Apache Hive, Presto; no strict specification between format and engine)	Apache Spark, Trino, Apache Flink, Apache Impala, Apache Hive, Presto, and others; Iceberg has strict specifications providing a clear contract between format and engine

Identifying Hive limitations and impacts

Hive's table format design may create several bottlenecks that become more noticeable as data volumes and concurrency requirements grow especially on object stores where Hive table format is not known to be performant. These include:

- **Slow queries from directory scanning:** Every query must scan directory structures to identify relevant files, resulting in long planning times that grow linearly with table size. For large tables, planning can take longer than actual query execution.
- **Inefficient updates requiring partition rewrites:** A single-row update forces a complete partition rewrite. This amplifies input/output (I/O) costs and makes near-real-time updates impractical.
- **Concurrency risks from uncoordinated writes:** Multiple writers accessing the same dataset can corrupt it because the way Hive doesn't support atomic transaction isolation as well as Iceberg. This forces serialization of writes, limiting throughput.
- **Stale statistics degrade optimization:** Query optimizers make poor execution plans when working with outdated statistics. Keeping statistics current requires manual maintenance and compute overhead.

TIP

These limitations directly increase cost and risk. Failed jobs waste compute resources and delay business outcomes. The productivity loss from waiting for jobs to complete or troubleshooting corrupted data adds up quickly across large organizations.

Discovering advanced features and AI benefits

Iceberg's advanced features directly address Hive table format limitations while enabling new capabilities that modern AI and analytics workloads require.

Schema evolution without rewrites that change table structure instantly:

```
ALTER TABLE patients ADD COLUMN cholesterol_
   level DOUBLE;
ALTER TABLE patients RENAME COLUMN dob
   TO date_of_birth;
```

These operations complete instantly because they only update metadata. Existing data files remain unchanged, with Iceberg handling the schema mapping transparently during reads.

TIP

Schema evolution applies to nested fields too, giving your data engineers the flexibility to adjust structures without requiring rewrites.

Beyond schema changes, Iceberg abstracts another operational complexity: partition management. Hidden partitioning for smarter queries:

```
CREATE TABLE sales_data (
  transaction_id int,
  transaction_time timestamp,
  amount double,
  cust_id string)
USING Iceberg
PARTITIONED BY (day(transaction_time));
```

Users query using the natural column (`transaction_time`) while Iceberg handles the physical partitioning by day automatically. This abstraction makes queries simpler and prevents the partition-awareness requirements that plague Hive queries.

Leveraging time travel for AI training and governance

Snapshot-based time travel enables reproducibility that is increasingly required for AI governance:

```
SELECT * FROM sales_data
FOR SYSTEM_TIME AS OF TIMESTAMP '2025-01-15
  10:00:00';
```

This query returns the exact state of the table at the specified time, regardless of subsequent changes. The capability supports several critical use cases:

- **Model reproducibility:** Data scientists can recreate the exact training dataset used for any model version, essential for debugging performance regressions or satisfying regulatory requirements.
- **Compliance audits:** You can prove what data existed at specific points in time, supporting regulatory investigations or internal audits.
- **Experimentation:** Researchers can compare results across different time periods using consistent snapshots, eliminating variables introduced by data changes.

REMEMBER

Time travel is becoming mandatory for AI governance. As regulations around AI decision-making tighten, the ability to prove which data snapshot trained which model transitions from "nice to have" to "required for compliance." Iceberg provides this capability out of the box.

Building the Migration Business Case and Decision Framework

Migration requires executive sponsorship. Focus your comments on business value in four key areas:

- **Efficiency:** Faster queries consume less compute time, directly reducing infrastructure costs while improving user productivity.

- **Governance:** Consistent access policies and lineage tracking through Apache Ranger and Cloudera Octopai Data Lineage ensure compliance across hybrid environments, ensuring compliance across hybrid environments, reducing audit costs and regulatory risk.
- **Collaboration:** Iceberg delivers flexibility to use any engine and allows concurrent access, which makes it easy for teams to collaborate on the same data.
- **AI readiness:** Iceberg's time travel capabilities enable reproducibility and auditability, addressing emerging requirements for responsible AI. Iceberg also serves as an open, constantly evolving feature store for ML pipelines, with capabilities like incremental changes, faster feature updates with Merge-on-Read (MoR), low-cost storage, and branching for experiments.
- **Ecosystem support:** Broad adoption across engines and vendors ensures future-proof architecture and prevents lock-in. Multi-engine adoption by major vendors guarantees long-term viability. Iceberg delivers flexibility to use any engine and allows concurrent access, which makes it easy for teams to collaborate on the same data.

TIP

Consider Iceberg a strategic investment in AI-ready governance and hybrid flexibility, not just a table format.

Measuring performance gains and cost benefits

Cloudera benchmarks across customer deployments show substantial improvements, including:

- **Query planning acceleration:** Manifest-based metadata reduces planning time by 3-10 times compared to Hive' table format's directory-scanning approach, with larger improvements for bigger tables. This improvement comes from Iceberg's ability to identify relevant files through metadata queries rather than physically listing directories in storage.
- **Storage consolidation:** Eliminating redundant table copies and using efficient file formats produces a 40-50 percent storage reduction on average.

- **Productivity improvements:** Analysts and engineers spend less time waiting for queries and more time analyzing results, producing 40-50 percent productivity gains as measured by tasks completed per week.

Applying decision checklists and AI considerations

Not all tables need immediate migration. Apply these criteria to prioritize your migration roadmap:

- **Reproducibility requirements:** Does this table feed AI models or compliance reports that require point-in-time reconstruction? Tables supporting AI training or regulatory reporting should migrate first.
- **Schema volatility:** Do schema changes frequently disrupt jobs or require complex migration scripts? Tables with evolving schemas benefit immediately from Iceberg's schema evolution.
- **Compliance strictness:** Are there regulatory requirements for audit trails or data lineage? Iceberg's snapshot history and integration with Apache Atlas provide comprehensive lineage.
- **Cost pressure:** Are compute or storage costs rising unsustainably? Tables with high query volumes or multiple copies are ideal candidates for immediate cost reduction through migration.
- **Small file problems:** Does the table have many small files (under 100MB)? Iceberg's compaction features address this performance problem systematically.

Presenting ROI metrics and AI value

ROI from Iceberg migration spans three interconnected areas:

- **Cost savings:** Reduced compute consumption from faster queries and lower storage costs from eliminating duplicates produce immediate, measurable savings. In cloud environments, reading less data from object stores also reduces transfer costs. These savings are easiest to quantify and often justify the migration investment alone.

- **Risk mitigation:** Improved governance, comprehensive audit trails, and reproducibility reduce regulatory risk and potential penalties. While harder to quantify, risk reduction can dwarf operational savings if it prevents a compliance failure.
- **Innovation enablement:** Faster AI development cycles, easier experimentation, and reduced data engineering toil accelerate time-to-value for new initiatives. This strategic benefit compounds over time as your organization becomes more data-driven.

TIP

Evaluate ROI as both short-term savings and long-term AI enablement. The immediate cost reductions justify the migration to finance teams, while the strategic capabilities align with business leaders' concerns about competitive positioning. Both perspectives are valid and mutually reinforcing.

REMEMBER

Migration typically proceeds in phases rather than as a single cutover:

- **Analytical workloads:** Start with BI and reporting tables where improved query performance delivers immediate value.
- **Operational workloads:** Migrate tables supporting operational applications once confidence is established.
- **Data and streaming pipelines:** Transition ingestion pipelines to write directly to Iceberg after downstream consumption patterns are stable.

Reviewing Cloudera's Migration Best Practices

Cloudera's migration framework helps organizations modernize in a controlled, low-risk way. The staged approach shown in Figure 1-3 ensures a measured transition, avoiding downtime while realizing faster analytics, lower cost, and AI-ready governance as Iceberg adoption expands across the enterprise.

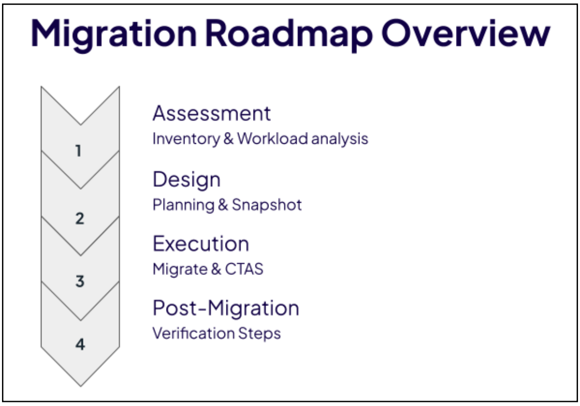

FIGURE 1-3: Iceberg migration journey.

The journey typically follows four phases:

- **Assessment:** Inventory exists in Hive tables, workloads, and dependencies. Identify tables that benefit most from Iceberg's performance, schema evolution, and governance capabilities.
- **Design:** Plan catalog integration (Hive Metastore or Iceberg REST Catalog), policy mapping with Ranger, and hybrid storage architecture under SDX governance.
- **Execution:** Convert Hive tables using `CREATE TABLE AS SELECT` (CTAS) or `ALTER TABLE` commands, test multi-engine access, and validate schemes consistency.
- **Verification:** Benchmark performance, validate lineage and security policies, and retire redundant copies.

IN THIS CHAPTER

» Laying the groundwork

» Picking smarter starters

» Proving success

Chapter 2 Creating the Migration Blueprint

Migrating from Hive table format to Apache Iceberg table format requires careful planning, phased execution, and cooperation across technical, governance, and business teams. This transformation modernizes your data architecture by moving from the limitations of the Hive table format to Apache Iceberg, enabling schema evolution, hidden partitioning, time travel, and native support for both batch and streaming workloads on a single table.

This chapter gives you a structured, step-by-step plan for transitioning to Iceberg with minimal risk and maximum benefit.

Complete migration scripts, validation playbooks, and troubleshooting guides are available in Cloudera's migration repository at `https://github.com/cloudera/iceberg-migration-for-dummies`. Refer to your specific engine's documentation for exact code syntax. Migration procedures vary among different query engines such as Apache Hive, Apache Spark, Apache Impala, and Trino.

Assessing Readiness and Planning the Timeline

Before migrating your first table, you need to take stock of your current environment. Readiness determines success. The better your planning, the smoother your migration will be.

Every Iceberg deployment begins with a catalog. A catalog is the source of truth for your table metadata: schemas, partitions, and snapshots. Without a stable catalog, nothing else works.

Cloudera supports two catalog options:

- **Hive Metastore:** Hive Metastore remains the backbone of Cloudera-managed data lakes and 90 percent of industry workloads. It's reliable and performant for many deployments.
- **REST Catalog:** The modern, flexible option. Cloudera recommends REST Catalog because it scales well, supports hybrid and multi-cloud deployments, and ensures interoperability across multiple Iceberg engines with consistent governance. REST Catalog uses a Metastore backend to store and retrieve metadata, providing consistent access. Users don't choose between HMS and REST. Both work together in Cloudera deployments.

TIP

It's best to pick one catalog strategy and stick with it. Splitting workloads across catalogs creates governance headaches and increases the risk of policy drift and challenges with visibility of data across the different catalogs.

Iceberg's biggest strength is its engine-agnostic nature. But only if the engines are properly configured with enterprise-grade security and compatibility verification. Cloudera ensures compatibility across Apache Spark, Apache Impala, Apache Hive, and Trino in Cloudera; however, you still need to verify versions and connectors are current.

TIP

In Cloudera Iceberg integration spans Cloudera Data Engineering Cloudera Data Warehousing, and Cloudera AI (formerly Machine Learning). This unified approach means that data engineers, BI analysts, and data scientists all work with the same, governed

Iceberg tables, without creating separate copies. One pipeline serves all AI and analytical workloads, dramatically reducing complexity and cost.

TECHNICAL STUFF

In Cloudera, Iceberg is first-class across Apache Spark, Apache Hive, Apache Impala, Apachi Flink, and Apache Nifi. This means you don't need parallel clusters for Hive and Iceberg; one unified environment reduces operational overhead. Migration approach depends on your use case. For tables with small-file problems `CREATE TABLE AS SELECT` (CTAS) with compaction solves data layout issues. For tables without small-file issues, metadata-only snapshot migration reduces cutover time from days to minutes.

Governance is where many migrations stumble. Apache Ranger and Cloudera Octopai Data Lineage are central to Cloudera's unified data fabric and should be validated early. Export Ranger policies if table names change. Ranger policies are format-agnostic and apply to both Hive and Iceberg tables, so if table names remain consistent, no policy changes are needed. Validate lineage in Octopai Data Lineage to ensure Iceberg tables appear in data flows.

WARNING

Policy drift, which occurs when there is a mismatch between old Hive table format rules and new Iceberg tables, is one of the most common migration failures. Always test policies before cutover.

Iceberg changes how your tables are structured, which in turn affects storage behavior. You need to check encryption settings to ensure data in S3, ADLS, HDFS, and Ozone meets compliance needs. Confirm replication and durability settings are correct. CTAS migrations can spike I/O, so verify that network and storage systems can handle the load.

REMEMBER

Cloudera benchmarks show significant savings post migration, in some cases up to 40-50 percent, mainly from eliminating duplicate Hive copies and improving compaction. Plan for this in your ROI projections.

Conducting workload analysis and prioritization

With readiness checked off, the next step is deciding which tables to migrate first. Not every Hive table deserves immediate attention. Prioritize workloads that will demonstrate Iceberg's value quickly.

Create an inventory that captures:

- **Frequency:** How often is the table queried or updated?
- **Size and growth:** Does the table contain billions of rows or grow rapidly?
- **Business criticality:** Is it tied to compliance, BI dashboards, or AI projects?
- **AI adjacency:** Will time travel or schema evolution make a big difference?

Consider a three-step framework:

1. Pick 1–2 pilot workloads that are high-value and high-visibility.
2. Measure and communicate the benefits, including faster queries, lower storage costs, and reproducible AI training sets.
3. Expand migration once the pilots prove successful.

REMEMBER

Don't start with your biggest or most complex table. Begin with medium-sized, high-value workloads to prove success quickly.

Evaluating environment requirements and resources

Even with the right catalog and workload plan, migrations can fail if you don't have the people, processes, and resources in place. Your infrastructure must be Iceberg-ready. You need to:

- Validate orchestration pipelines (Apache Airflow, Apache NiFi) that move data.
- Validate workload. Your ingestion and consumption engines should be Iceberg compatible.

Iceberg introduces new concepts like snapshots, compaction, and partition transforms. That means teams need training. Consider these:

- Data engineers should practice migration commands in test environments.

- Governance teams must rehearse lineage validation.
- Appoint a migration champion to coordinate progress and resolve blocks.

Migrations consume resources. Plan for:

- **Compute:** CTAS jobs can be resource intensive.
- **Storage:** Temporary copies and backups are required.
- **Environments:** You'll need test clusters for validation and rollback rehearsals.

Building an 8–12 week timeline and phase plan

Migration is best approached in phases. Break the process into digestible chunks, with clear milestones for validation and rollback.

- **Weeks 1–2: Inventory & readiness:**
  - Catalog decisions finalized
  - Ranger policies exported
  - Storage encryption and replication validated
- **Weeks 3–4: Governance alignment:**
  - Dry-run Ranger imports into Iceberg tables
  - Atlas lineage validation
  - IAM integration is tested across all engines
- **Weeks 5–6: Pilot migration:**
  - Use Snapshot or CTAS methods to migrate 1-2 workloads
  - Run Hive tables and Iceberg pipelines in parallel for safety
- **Weeks 7–8: Validation:**
  - Benchmark query latencies and scan sizes
  - Test time travel queries to confirm reproducibility
  - Conduct canary testing with BI dashboards and ML pipelines

- **Weeks 9–12: Expansion:**
  - Scale migration to more workloads
  - Automate maintenance (compaction, snapshot expiration)
  - Finalize rollback documentation and drills

Implementing the RACI framework and managing stakeholders

Migration is cross-functional by nature. Without clear roles, tasks slip through the cracks. That's where the RACI framework comes in: Responsible (who does the work), Accountable (who owns the outcome), Consulted (who provides input), Informed (who receives updates). Data engineers, governance staff, platform owners, and business users all have different stakes in migration. RACI ensures that everyone knows their role, reducing confusion and speeding up decision-making.

RACI roles in Iceberg migration are:

- **Responsible:** Data engineers who run migration jobs and validate results
- **Accountable:** Platform owners who ensure migration success
- **Consulted:** Security, governance, and compliance teams, who weigh in on risks
- **Informed:** Business stakeholders who need assurance that their dashboards and AI pipelines won't break

TIP

Hold weekly syncs to review progress and blockers, share before-and-after performance benchmarks with the business team, and keep executives informed about risks and rollback readiness.

Executing Migration Steps and Implementation

Once preparation is complete, execute the migration systematically.

Preparing migration setup and configuration

To set up migration, start with:

- Validate catalog connectivity (Hive Metastore, or Iceberg REST Catalog)
- Export Ranger policies for Hive tables (if needed)
- Backup data files and Hive metadata

REMEMBER

Never skip backups. Even if you don't need them, backups provide confidence during cutover.

Implementing migration operations and CTAS processes

Consider these four methods for Hive tables to Iceberg:

- **Snapshot migration:** Creates an Iceberg table pointing to existing Hive table data files. Fast, minimal data movement. Best for tables with small-file problems.
- **CTAS (`CREATE TABLE AS SELECT`):** Rewrites Hive table data into a new Iceberg table. Best for tables needing data reorganization or fixing small-file issues.
- **In-place migration:** Converts a Hive table directly into Iceberg, with automatic backup for rollback. Best for tables where your engine version supports in-place conversion.
- **Near-zero downtime (Dual writes + `add_files`):** For business-critical tables that cannot tolerate downtime, write to both Hive tables and Iceberg simultaneously during a transition period, then use the `add_files` procedure to import Iceberg-written data. This requires careful coordination but enables truly zero-downtime cutover. Best for mission-critical production tables after migration process is proven.

TIP

Most organizations use CTAS for their first one or two pilot tables because it provides a clean side-by-side migration with easy rollback. Save the near-zero downtime approach for mission-critical production tables after you've proven the migration process.

Managing cutover and rollback points

Cutover is the process of switching workloads to Iceberg. Safeguards include:

- » Recording rollback snapshots before cutover
- » Redirecting downstream jobs to Iceberg tables
- » Keeping Hive tables in read-only mode until Iceberg validation passes

REMEMBER

Cutting over without a rollback plan is the fastest path to a failed migration.

Understanding the Three Core Migration Paths

Before you execute your first migration, you need to understand the technical details of each approach. Each method has specific commands, requirements, and trade-offs.

Using in-place migration for the fastest metadata conversion

In-place migration converts your existing Hive table to Iceberg format without rewriting data files. This is the fastest approach when your data layout is already optimized. The command creates Iceberg metadata that references your existing data:

```
-- Spark SQL exampleCALL prod_catalog.system.
  migrate('table_name');
```

This single command migrates your table in place and automatically generates a backup table for rollback protection. After migration completes, trigger compaction immediately to optimize file layout before resuming production traffic.

Implementing CTAS for clean reorganization

When you need optimized partitioning from day one, CTAS rewrites your data into a new Iceberg table with your desired configuration (`prod.sales_events` is a sample table name):

```
CREATE TABLE prod.sales_events_iceberg (
  event_id BIGINT,
  ts TIMESTAMP,
  cat STRING,
  payload STRING
)
USING iceberg
PARTITIONED BY (bucket(16, event_id), days(ts))
TBLPROPERTIES(
  'write.target-file-size-bytes'='536870912'
)
AS
SELECT event_id, ts, cat, payload
FROM prod.sales_events;
```

This approach gives you control over file size targets (the example uses 512 MB) and partition strategies while keeping your source table intact during validation. The `PARTITIONED BY` clause uses Iceberg's hidden partitioning — users query without specifying partition columns, but Iceberg automatically optimizes scans.

TECHNICAL STUFF

The `bucket(16, event_id)` partition transform distributes data evenly across 16 buckets, preventing partition skew. The `days(ts)` transform creates daily partitions without exposing partition columns to users. This is one of Iceberg's biggest advantages over Hive tables — partition evolution without breaking queries.

Executing near-zero downtime migrations

For mission-critical tables that cannot tolerate downtime, use dual writes with the `add_files` procedure. Start by creating your Iceberg table via snapshot or CTAS, then import new files written during the transition window:

```
-- After initial Iceberg table creation
CALL prod_catalog.system.add_files(
  table => 'prod.sales_events_iceberg',
  source_table => 'prod.sales_events'
);
```

WARNING

Handle rename and move edge cases carefully during near-zero downtime migrations. Clock drift between nodes can cause snapshot misalignment. Always validate time boundaries before switching production workloads to the Iceberg table.

Validating migration success with queries

After migration completes, you need to prove data integrity before cutover. Execute these validation checks systematically.

Comparing row counts

Start with the simplest check — total row counts must match:

```
-- Source Hive table
SELECT COUNT(*) FROM prod.sales_events;
-- Migrated Iceberg table
SELECT COUNT(*) FROM prod.sales_events_iceberg;
```

If counts don't match, investigate before proceeding. Missing rows indicate incomplete migration or filtering errors.

Checking per-partition integrity

Aggregate row counts by partition to catch partial migration failures:

```
SELECT partition_col, COUNT(*)
FROM prod.sales_events_iceberg
GROUP BY partition_col
ORDER BY partition_col;
```

Compare these counts against your source table. Partition-level mismatches reveal problems that aggregate counts might hide.

Validating distinct value counts

NDV (Number of Distinct Values) checks confirm data distribution:

```
-- Source table
SELECT COUNT(DISTINCT event_id) FROM prod.
  sales_events;
```

```
-- Migrated table
SELECT COUNT(DISTINCT event_id) FROM prod.
  sales_events_iceberg;
```

NDV mismatches suggest data corruption or issues with duplicate handling during migration.

Running canary queries

Execute your top five production queries against both tables and compare latency and results. This real-world validation catches edge cases that synthetic tests miss. If a query returns different results or fails against Iceberg, investigate schema mapping or partition pruning before cutover.

Scheduling post-migration maintenance

Migration doesn't end when data moves. Iceberg requires ongoing maintenance to sustain performance benefits.

Compacting small files

After migration, you may end up with suboptimal file layouts. Schedule compaction to consolidate small files:

```
CALL prod_catalog.system.rewrite_data_files(
  table => 'prod.sales_events_iceberg'
);
```

TIP

Run initial compaction immediately after migration, before resuming peak traffic. This prevents small-file bloat from degrading query performance.

Expiring old snapshots

Iceberg creates a new snapshot after each write. Without expiration, snapshots accumulate and slow metadata operations:

```
-- Expire snapshots older than 7 days
CALL prod_catalog.system.expire_snapshots(
  table => 'prod.sales_events_iceberg',
  older_than => TIMESTAMP '2025-08-01 00:00:00'
);
```

Balance retention policies against storage costs and audit requirements. Most organizations retain 7 to 30 days of snapshots for time travel and rollback capability.

REMEMBER

Skipping maintenance leads to metadata growth, which slows query planning. Schedule compaction and snapshot expiration from day one — don't wait for performance to degrade.

Validating Results and Performance Testing

Migration success isn't just about moving data; it's about proving that the new tables meet or exceed the original performance while maintaining data integrity.

Implementing the validation playbook and sampling

Validation starts with a playbook: compare Hive tables versus Iceberg query results, validate row counts and schema mappings, and sample partitions for integrity checks.

Conducting canary testing and time-travel validation

Canary testing runs small but critical workloads against Iceberg:

- **BI dashboards:** Validate that reports generate correctly.
- **AI/ML pipelines:** Confirm training sets remain reproducible with time travel queries.

Measuring performance improvements and SLO compliance

Executives need proof. Benchmark Iceberg tables against Hive tables:

- **Query latency:** Typical improvements of three to ten times in planning speed

- **Scan size:** Manifests reduce wasted I/O by 60-90 percent
- **Storage costs:** Fewer ad hoc data copies
- **SLOs:** Confirm analytics SLAs are met or exceeded

Avoiding Pitfalls and Planning Recovery

As we know, even with preparation, things can go wrong. Planning recovery avoids disaster.

Managing migration risks and contingency plans

Consider these three common risks:

- **Policy drift:** Ranger rules don't align after migration
- **Clock drift:** Unsynced nodes cause snapshot misalignment
- **Small-file storms:** Without compaction, performance degrades

Also, here are some common migration errors and solutions to consider:

- **Schema mismatch:** Validate column types before migration
- **Permission denied:** Ensure IAM roles include Iceberg paths
- **Slow CTAS:** Increase executor memory and parallelize writes
- **Snapshot conflicts:** Check time synchronization across nodes
- **Compaction failures:** Verify sufficient temp space and memory

TIP

Run governance dry runs and schedule compaction jobs immediately after migration.

Implementing rollback strategies and risk communication

Rollback is your safety net. Always plan it to:

- Keep Hive table backups until Iceberg is proven stable.
- Document rollback steps: restore Hive tables, expire Iceberg snapshots.
- Rehearse rollback in lower environments.

IN THIS CHAPTER

» Modernizing batch jobs

» Enabling streaming ingestion

» Maintaining healthy tables

Chapter 3
Building Data Pipelines and Processing AI Data

Apache Iceberg isn't just a table format; it's the foundation for building reliable data and AI pipelines. Migrating from Hive tables isn't the end of your journey; it's the beginning of new ways to process and deliver data.

This chapter explores how to design scalable pipelines, how to modernize batch jobs with Apache Spark, enable streaming pipelines with Apache Flink, Apache Kafka, and Apache NiFi, and keep your schemas and tables performing well.

Foundations of Designing Scalable Data and AI Pipelines

Data pipelines transform raw data into well-curated, cleaned datasets for analytics and AI. In Hive-based systems, pipelines were often brittle: updates required partition rewrites, multiple engines created duplicate tables, and governance rules were hard to maintain.

Iceberg modernizes data pipelines through several key improvements, including:

- **Abstracting physical layout:** Data pipeline jobs no longer break when partitions change. Iceberg handles the physical data organization automatically, freeing your pipeline developers from managing folder structures and partition schemes.
- **Supporting schema evolution:** Fast-changing business and AI requirements demand flexible schemas. Iceberg allows you to add, remove, or modify columns without rewriting entire tables or breaking existing queries.
- **Enabling time travel:** AI training data can be reproduced exactly at any point in time. This capability is essential for model reproducibility, regulatory compliance, and debugging production issues.
- **Working across multiple engines:** A single Iceberg table can serve multiple use cases simultaneously, reducing the need for duplicate pipelines. Instead of maintaining separate Hive tables for BI, AI, and reporting, one Iceberg table serves all three.

REMEMBER

In Cloudera deployments, all these improvements are governed by Cloudera's unified fabric, meaning Apache Ranger security policies and Cloudera Octopai Data Lineage tracking apply consistently across Apache Spark, Apache Hive, Apache Flink, and Trino pipelines.

Modernizing Batch Processing with Apache Spark and Iceberg

Apache Spark has long been the engine of choice for batch data processing, but when coupled with the Hive table format, it inherits certain limitations. Iceberg makes your Spark data processing pipelines faster, safer, and easier to manage.

- **Write-Audit-Publish patterns:** Iceberg enables patterns for pipeline CI/CD, safely stage and validate large batch writes before committing to production tables essential for compliance, data quality, and multi-team workflows.

- **Built-in versioning:** Every write creates a snapshot, allowing easy rollback, auditing, or reproducing data at any point in time-ideal for AI model training and governance.
- **Branching and tagging for reproducible, collaborative workflows:** Iceberg supports lightweight metadata branching and human-readable tags, enabling teams to safely test changes in isolated branches and reference specific snapshots (for example, v1.1-post-cleanup) in downstream workflows.

Transforming batch jobs and integrating Spark

Iceberg gives your batch jobs serious upgrades:

- **Atomic commits:** Batch jobs write atomic commits into Iceberg tables, ensuring that either all changes succeed or none do. This eliminates the partial-write failures that plagued Hive-based pipelines.
- **Safe row-level operations:** Jobs can merge, update, and delete rows safely without the risk of corrupt partitions. Iceberg's copy-on write and merge-on read strategies allow delete file mechanism, upserts, and `MERGE INTO` to handle record-level modifications in specific ways.
- **Automatic partition management:** Partition transforms mean Spark users don't have to manage physical layouts. Iceberg handles partition evolution automatically, adapting as your data patterns change.

Here's a simple example of migrating a Hive batch data pipeline job to Iceberg.

```
-- Old Hive ETL: rewriting partitions
INSERT OVERWRITE TABLE hive.sales PARTITION (dt)
SELECT * FROM staging_sales WHERE dt =
  '2025-01-15';

-- Modern Iceberg ETL
MERGE INTO Iceberg.sales t
USING staging_sales s
```

```
ON t.id = s.id
WHEN MATCHED THEN UPDATE SET *
WHEN NOT MATCHED THEN INSERT *;
```

TECHNICAL STUFF

Iceberg's `MERGE INTO` operation and row-level deletes (via positional/equality delete files) make Spark batch ETL safer and more efficient.

Cloudera Data Engineering, shown in Figure 3-1, empowers enterprise teams to securely build, automate, and scale data pipelines on the foundations of an open lakehouse. It unifies structured and unstructured data with Apache Spark on Iceberg, orchestrated through Airflow — fully open with no vendor lock-ins, enabling teams to power multi-function analytics and AI for data anywhere.

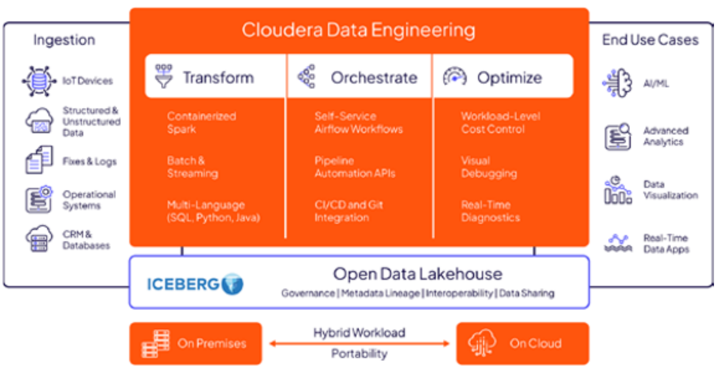

FIGURE 3-1: Cloudera Data Engineering.

Enabling AI pipelines with time travel

Iceberg's time travel capabilities turn your Spark jobs into reproducible AI pipelines. Time travel enables version control for data so you can train a model (say model1) on V1 of the dataset, model2 on V2, and so on. You can audit and trace lineage with snapshots and recreate models on identical data using time-travel queries.

Data scientists can recreate training datasets using (time-travel); modifies queries (as shown in Chapter 1), ensuring experiments are reproducible exactly. Iceberg also supports branching for experimentation, enabling teams to safely test changes in isolated branches without impacting production.

REMEMBER

Reproducibility isn't optional in AI and machine learning. With Iceberg and Apache Spark, AI training sets can always be recreated, satisfying auditors and regulators.

Enabling Streaming with Apache Flink, Apache Kafka, and Apache NiFi

Real-time data has become essential for modern businesses. You can't wait for overnight batch jobs when customer behavior, market conditions, or operational metrics change by the second. Streaming pipelines deliver fresh data continuously into your analytics environment.

Cloudera Data in Motion, shown in Figure 3-2 is an end-to-end platform that accelerates real-time business insights and trusted AI applications. Utilizing Apache NiFi, Kafka, and Flink, it delivers real-time data ingest, transformation, and analysis across the entire data lifecycle, from the edge to AI. The solution provides trusted security and governance and enables immediate insight through easy-to-use visuals and no-code tools.

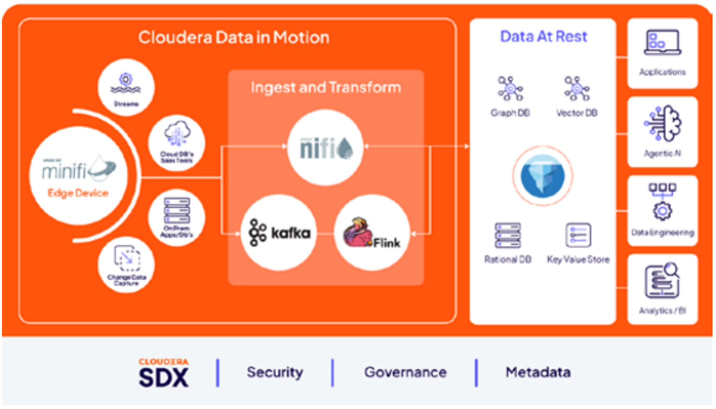

FIGURE 3-2: Cloudera Data in Motion.

Creating real-time pipelines and data flow

Streaming integration with Apache Flink and Apache Kafka provides several advantages, including:

- **Direct stream ingestion:** Stream ingestion writes data continuously from sources like Apache Kafka or AWS Kinesis directly into Iceberg tables without intermediate staging areas. This reduces latency from hours to seconds. Kafka topics flow through Flink transformations and land in Iceberg, making data available for queries almost immediately.
- **Snapshot isolation:** Iceberg's snapshot isolation ensures concurrent reads and writes are safe. Streaming writers can continuously append data while analytical queries read consistent snapshots without locking or conflicts.
- **Near real-time queries:** Queries can consume the latest snapshot in near real-time. As new data is committed, subsequent queries automatically display the updated information without requiring manual refresh operations.

Building streaming architectures

Iceberg supports *upserts* (update or insert operations) is the ability to update existing records or insert new ones. Via Apache Flink or Spark Structured Streaming using equality deletes. This allows you to update existing records in near real-time as new information arrives:

```
-- Flink streaming upsert into Iceberg
-- Primary key defined in table DDL handles
   conflicts automatically
INSERT INTO Iceberg.transactions
SELECT transaction_id, amount, customer_id,
   event_time
FROM kafka_stream;
```

REMEMBER

Iceberg handles upserts automatically when tables have primary keys defined. Flink writes equality delete files for existing keys before inserting updated records, ensuring exactly-once semantics.

Incremental processing is another key pattern enabled by Iceberg's snapshot model:

- **It processes only new data.** Instead of full table scans, your streaming jobs can process only new data by comparing snapshots. This dramatically reduces computational costs and latency.
- **It maintains exactly-once semantics.** Iceberg's atomic commits ensure exactly-once semantics, preventing duplicate records even when streaming jobs restart or fail.

TIP

Consider building incremental pipelines (process only new data via snapshots) instead of full refreshes. This cuts cost and latency dramatically.

Applying streaming solutions and industry examples

Organizations across industries are leveraging streaming with Iceberg to solve real-time data challenges. Here are some common patterns:

- **Finance:** Fraud detection pipelines run on Kafka and Iceberg with millisecond latency. As transactions flow through payment systems, streaming analytics identify suspicious patterns and trigger alerts before fraudulent transactions are complete.
- **Telecom:** Network operations teams stream real-time usage metrics into Iceberg tables. Engineers monitor network performance, capacity utilization, and service quality in near real-time dashboards, enabling rapid response to issues.
- **Healthcare:** Patient monitoring data ingested via NiFi streams into AI dashboards. Continuous vital sign monitoring flows into Iceberg tables, where machine learning models identify deteriorating patient conditions and alert clinical staff.

WARNING

Streaming pipelines can create small-file storms. Always schedule compaction jobs in Iceberg after heavy streaming loads to consolidate these small files. See the earlier section, "Building streaming architectures" for more information.

Applying Schema Evolution and Table Maintenance

Schema changes and table maintenance are essential for keeping pipelines efficient and AI-ready over time. As business requirements evolve and data sources change, your table schemas must adapt without breaking existing queries or pipelines.

Managing schema evolution

Iceberg lets you safely evolve schemas with several operations that would be risky or impossible in Hive tables:

- **Add or drop columns:** You can add new columns to support additional features or remove deprecated columns without rewriting data or breaking existing queries that don't reference the new fields.
- **Rename fields:** Column renames propagate through the metadata layer without breaking queries, provided you use the appropriate compatibility settings.
- **Apply nested schema updates:** For complex data structures, especially with ML features that typically store highly nested data — feature vectors, metadata, and embeddings you can modify nested fields within structs and arrays without impacting the parent structure.

REMEMBER

Schema evolution supports long-lived AI pipelines that frequently change features. As your data science team discovers new predictive features or regulatory requirements demand additional fields, you can adapt your tables without disrupting production pipelines or retraining all models.

Implementing compaction and snapshot policies

Compaction and snapshot expiration keep your Iceberg tables performing well over time:

- **Compaction merges small files into larger ones for faster reads.** This is particularly important after streaming ingestion or frequent incremental updates that create many small files. Larger files reduce metadata overhead and improve query performance by minimizing the number of file opens, typically targeting file sizes of 100 MB to 1 GB.

- **Rewrite manifests reduce metadata overhead by consolidating manifest files.** As tables evolve with many commits, manifest files accumulate. Rewriting them periodically improves query planning performance.
- **Expire snapshots reclaim storage while preserving governance policies.** Each Iceberg commit creates a new snapshot, and without expiration, storage costs grow indefinitely. Snapshot expiration removes old snapshots according to your retention policies while maintaining audit trails.

Cloudera Lakehouse Optimizer takes care of these optimizations automatically. (Refer to Chapter 4 for more details.)

With Iceberg, data pipelines become faster, safer, and future-proof. The platform delivers governed, reproducible datasets for analytics and AI while reducing operational complexity and infrastructure costs.

IN THIS CHAPTER

» Accelerating analytics with Trino

» Enforcing governance with Apache Ranger

» Maintaining table performance

Chapter 4

Boosting Analytics and Managing Security

Now that you've successfully migrated your tables to Iceberg, the next challenge is performance. Traditional query engines often struggle to fully unlock Iceberg's potential. That's where Trino comes in.

This chapter explores how to accelerate your BI queries on Iceberg with Trino (a distributed SQL query engine), enforce governance to ensure unified access policies and controls, implement metadata intelligence and lineage, and keep Iceberg tables performing with smart maintenance.

Enabling Fast Analytics with Trino

Trino has become the engine of choice for interactive SQL analytics on data lakes. Combined with Iceberg, Trino allows enterprises to deliver multi-workload, federated analytics without duplicating data.

Leveraging Trino for multi-workload analytics

Trino's combination of speed, SQL compatibility, and federation capabilities makes it ideal for diverse analytics workloads. When paired with Iceberg, these capabilities extend across your entire data platform:

- **BI analysts run dashboards directly on Iceberg tables.** Instead of extracting data into separate data warehouses, analysts query governed Iceberg tables directly. This eliminates data movement, reduces latency between source updates and analytics availability, and ensures everyone works with the same, current data.
- **Federation crosses systems.** Trino enables querying multiple data sources in a single query. For example, a BI analyst can JOIN customer data from Iceberg lakehouse tables with account details from an operational database, using Trino to federate access across systems without copying data between them.
- **Data scientists query the same data with Python/SQL notebooks.** Trino can be used with Jupyter or other data science tools to query the same Iceberg tables that power the BI dashboards. This unified access eliminates the need for separate "data science" copies of production data.
- **Engineers integrate Trino queries into orchestration workflows.** Apache Airflow, Apache NiFi, and other orchestration tools can execute Trino queries as part of automated workflows, enabling complex data processing pipelines that span multiple systems.

TIP

Think of Trino as the "last mile" for analytics. Instead of copying Hive table data into warehouses, your users query governed Iceberg tables directly, cutting cost and complexity. This eliminates the lag time between source system updates and analytics availability, while reducing the infrastructure needed to maintain separate analytical databases.

Optimizing multi-engine performance and access

Cloudera benchmarks show Iceberg boosts Trino performance across multiple dimensions:

- **3-10 times faster query planning:** Manifest-based pruning eliminates the need to list and inspect thousands of data files during query planning. Trino reads manifest metadata to identify relevant files, dramatically reducing planning overhead for large tables.
- **Reduced input/output (I/O) and compute costs:** Queries scan only relevant files based on partition pruning and min-max statistics stored in manifests. This targeted I/O reduces query execution time and lowers compute costs.

TECHNICAL STUFF

Trino pushes filters into Iceberg manifests:

```
SELECT customer_id, amount
FROM Iceberg.transactions
WHERE transaction_date BETWEEN DATE '2025-01-01'
   AND DATE '2025-01-31';
```

Instead of scanning all January partitions, Trino uses manifest metadata to skip irrelevant files. The manifest contains min-max statistics for each data file, allowing Trino to eliminate files that couldn't possibly contain matching records before any I/O occurs.

Demonstrating analytics performance improvements

Real-world deployments show substantial performance gains across industries when combining Trino with Iceberg:

- **Finance:** Risk dashboards refresh in minutes instead of hours. Regulatory reporting that previously required overnight batch processing now completes interactively, enabling analysts to iterate on reports and respond to changing requirements quickly.

- **Telecom:** Network operations teams run queries eight times faster than with Hive tables. Real-time monitoring dashboards can now track millions of network events without compromising performance, enabling rapid responses to service issues.
- **Healthcare:** Research cohorts are assembled 40 percent faster. Clinical researchers can identify patient populations for studies more quickly, accelerating medical research while maintaining legal compliance through Apache Ranger policies.

Security and Metadata Intelligence for Modern Lakehouses

Analytics requires robust governance and security. Cloudera's unified fabric ensures that consistent Apache Ranger policies and Cloudera Octopai Data Lineage tracks across Iceberg tables.

Migrating Ranger policies and enforcing security

Moving access control from Hive tables to Iceberg tables requires careful planning around your existing security rules. Ranger's access policy model is format-agnostic. It doesn't have separate "Iceberg policies."

The process depends on whether table names remain consistent:

- **If table names are preserved:** No policy changes are needed. Existing Ranger policies automatically apply to Iceberg tables with the same database and table names.
- **If table names changed:** You'll need to update resource paths in your existing policies to reference the new Iceberg table locations.

What matters most is understanding this distinction upfront to avoid unnecessary policy migration work.

REMEMBER

The majority of migrations preserve table names, requiring zero policy changes. Only migrate policies when you've specifically changed table names or namespaces during the Iceberg conversion. Always validate this assumption before starting policy work.

TIP

Before migrating any policies, confirm that your table names are actually changing. Run a quick inventory comparing `database.table` names in Hive versus your planned Iceberg naming convention. Many teams discover they can skip policy migration entirely.

For scenarios where table names change, follow these steps:

1. **Export existing Hive table policies.**

   Begin by exporting your current Hive table and column-level policies. These policies represent years of refinement and business logic about who should access what data.

2. **Update resource paths.**

   Modify the exported policies to reference the new Iceberg table names and locations.

3. **Import updated policies.**

   Ranger allows you to import the modified policies, maintaining your existing security model while pointing to the new table names.

4. **Transition to tag-based policies.**

   Rather than brittle path-based rules that break when data moves, use tag-based policies for flexibility. Tag-based security remains valid regardless of the physical location of data or storage reorganization.

REMEMBER

Policy drift is the most common governance failure. Always test your policies in staging before making a production cutover.

EXPORTING/IMPORTING TABLE NAMES

TECHNICAL STUFF

There are two ways to export and import when table names change within Ranger.

Using the Ranger admin user interface:

1. Navigate to Ranger Admin UI to Service Manager to Your Hive service.
2. Click the Export icon to download policies as JSON.
3. Edit the JSON file to update database/table resource paths.
4. Click the Import icon and upload the modified JSON.

Using the Ranger REST API:

```
# Export policies via REST API
curl -u admin:password \
  "http://ranger-host:6080/service/plugins/
   policies/exportJson?serviceName=your-
   hive-service" \
  -o hive-policies.json

# Edit hive-policies.json to update table paths
# Import updated policies
curl -u admin:password -X POST \
  "http://ranger-host:6080/service/plugins/
   policies/importPoliciesFromFile?serviceName=
   your-hive-service" \
  -F "file=@hive-policies.json"
```

Cloudera's migration utilities may also include automated scripts for bulk policy remapping.

Implementing automated metadata intelligence

Modern lakehouses generate substantial metadata that must be managed systematically. Cloudera Octopai Data Lineage provides automated data lineage and metadata intelligence to help you maintain control by:

- **Automated metadata harvesting:** Multi-vendor connectors build a comprehensive knowledge graph of your data ecosystem quickly, discovering all existing datasets, tables, views, and pipelines across legacy and cloud systems.
- **Column-level lineage:** Validate column semantics after migration and produce targeted tests. This detailed tracking ensures individual columns maintain their meaning and transformation logic when you move from Hive table format to Iceberg.
- **Reporting and governance capabilities:** Generate reports on data owners, consumer lists, and data flows to build migration and data governance runbooks and stakeholder communications. These exports help you create prioritized migration plans and keep teams informed throughout the process.

REMEMBER

Metadata isn't just about compliance. It's a productivity booster. The faster your analysts can find trustworthy data, the more value you get from Iceberg.

Tracking data lineage

Data lineage shows you how data flows through your systems, from source to final report. Cloudera Octopai Data Lineage integration with Iceberg provides comprehensive lineage tracking:

- **Source-to-report lineage:** Track data from initial ingestion through Iceberg tables to BI dashboards and reports. When questions arise about report accuracy, lineage allows you to trace back through every transformation and join to identify issues.
- **Transformation lineage:** Apache Spark and Apache Flink job history is tied to specific snapshots, providing an audit trail of how data was processed. You can see exactly which version of your transformation logic produced which version of your data.
- **AI lineage:** Prove which dataset snapshot trained which model. When model performance changes unexpectedly, lineage reveals whether the issue stems from code changes, data drift, or both. This capability is increasingly required by AI governance regulations and internal audit frameworks.

Making the migration risk-free with Cloudera Octopai Data Lineage

Migrating Hive tables to Iceberg format introduces risks, broken dependencies, untested transformations, and incomplete cutover. Cloudera Octopai Data Lineage turns migration from guesswork into systematic validation:

- **Discover what needs moving.** Cloudera Octopai Data Lineage's automatically harvests metadata across your data ecosystem, creating a complete inventory of tables, views, transformations, and pipelines. Before migration begins, you'll know exactly which assets exist, who owns them, and what depends on them.
- **Prioritize migration by impact.** Not all tables carry equal risk. Using Cloudera Octopai Data Lineage maps, identify low-risk tables with minimal downstream consumers for early migration. Business-critical tables with complex dependencies move later, after validating your approach works.
- **Choose the right migration method.** Iceberg supports multiple migration approaches — in-place metadata conversion, add-files, or full data copy. Cloudera Octopai Data Lineage's runtime usage patterns reveal file sizes, compaction frequency, and access patterns that determine the safest method per table.
- **Validate post-migration flows.** After migrating a table, compare pre- and post-migration lineage at column level. Cloudera Octopai Data Lineage surfaces mismatches in data flows, missing transformations, or broken consumer connections before users discover problems.
- **Maintain governance after cutover.** Keep Cloudera Octopai Data Lineage current as consumers switch to Iceberg tables. Governance teams, auditors, and downstream users see new table provenance, transformation history, and data flow documentation, supporting reproducibility and troubleshooting.

Optimizing Lakehouse Performance with Smart Table Maintenance

Migrating to Iceberg resolves many Hive table format issues, but ongoing maintenance is crucial for maintaining long-term performance. Without regular upkeep, even Iceberg tables will eventually degrade as small files accumulate and metadata grows. Cloudera automates optimizations for peak performance in Iceberg tables with its Lakehouse Optimizer service.

Implementing compaction, clustering, and metadata pruning

Three core maintenance operations keep Iceberg tables performing optimally:

- **Compaction:** Merges small files into larger ones for faster queries. Small files create metadata overhead and reduce query performance because each file requires separate I/O operations. Compaction consolidates these small files into fewer, larger files that scan more efficiently.
- **Clustering:** Groups similar records together to improve data locality. When related records are stored together, queries with filters on clustering keys read fewer files. For example, clustering by `customer_id` ensures that all transactions for a customer are collocated (stored in adjacent files), making customer-level queries significantly faster.
- **Metadata maintenance:** Manifests eliminate 60–90 percent of unnecessary file scans during query planning. Cloudera Lakehouse Optimizer automatically handles critical metadata tasks including manifest rewrites, position delete file compaction, and orphan file removal. Position deletes pile up as small files after row-level operations, slowing query planning as engines must scan them to identify deleted rows. The Optimizer compacts these delete files, reducing metadata bloat and maintaining query performance.

TIP

Schedule compaction after streaming or high-ingest jobs. A little tuning prevents massive slowdowns later.

Automating table maintenance with intelligent optimizers

Rather than running maintenance operations manually, organizations use orchestration tools like Apache Airflow to automate upkeep:

- **Rewrite manifests regularly.** Schedule manifest rewrites every few hours for high-change tables or daily for stable tables. This operation is lightweight but provides substantial improvements in query planning.

- **Compact data files based on workload.** Configure compaction frequency based on write patterns. Streaming tables with continuous small writes need frequent compaction, while batch-loaded tables might need only weekly maintenance.
- **Expire old snapshots to save storage.** Implement snapshot retention policies that balance time-travel requirements against storage costs. For example, keep 30 days of snapshots for operational troubleshooting and compliance, then expire older snapshots to reclaim storage space.

Cloudera Lakehouse Optimizer delivers intelligent, automated optimizations beyond basic maintenance. The service provides adaptive intelligence based on individual customer usage patterns, making it engine-agnostic so optimizations benefit all query engines accessing your data.

TIP

Internal benchmarks show query performance improvements up to 13 times faster, storage cost reductions of 36 percent, and over 95 percent reduction in administrative effort compared to manual maintenance.

The Optimizer creates policies defining when to run activities like snapshot expiration, file compaction, and orphan file removal, then assigns these policies to specific tables or entire catalogs. This automation works across any public cloud and will be available on-premises in future releases.

REMEMBER

Orphan file removal deletes data files that are no longer referenced by any snapshot, reclaiming storage space that accumulates after compaction or failed operations.

IN THIS CHAPTER

» Eliminating data duplication with zero-copy patterns

» Enabling multi-engine access through the REST catalog

» Measuring ROI from reduced data pipelines and storage

Chapter 5
Enabling Zero-Copy Access with the REST Catalog

The promise of Iceberg isn't just faster queries or easier schema changes. It's the ability to maintain a single copy of data that multiple engines and teams can access without duplication. But having a single copy isn't enough; you need a way for different tools to discover and access that data consistently.

Today, multiple catalog implementations exist for Iceberg like Hive Metastore, Nessie, and others. Each speaks its own dialect, requiring custom integrations and creating fragmentation. The REST Catalog specification standardizes how Iceberg clients communicate with catalogs, eliminating the need for custom integrations and enabling true interoperability across different catalog implementations.

This standardization enables third-party engines to work on the same copy of an Iceberg table with bi-directional interoperability where changes made from one engine propagate throughout the lakehouse. Instead of building duplicate pipelines, your enterprise can query the same governed dataset with Apache Spark, Trino, Hive, Flink, or Apache Impala, all through a consistent catalog.

This chapter explores how the REST catalog enables zero-copy data collaboration, what it means for interoperability and security, and how it drives cost savings and ROI.

Reducing Data Pipelines Through Zero-Copy Patterns

Traditional data pipelines were designed to move data into specialized silos such as a warehouse for BI, a data lake for data science, and an operational store for streaming. Each copy introduced cost, governance gaps, and latency.

With the REST catalog, Cloudera enables zero-copy patterns where multiple engines can point to one Iceberg REST catalog to access an Iceberg table, eliminating duplication.

Understanding zero-copy benefits and architecture

Zero-copy architecture fundamentally changes how you think about data access. The REST catalog solves the fragmentation issue with multiple different catalogs that cannot read/write the same Iceberg table because they don't speak the same language. Before REST standardization, if your table was registered in one catalog, you couldn't easily share or manage it from another without building special connectors. The REST catalog lets all engines share the same Iceberg table with consistent governance and immediate access to updates. Key advantages of zero-copy architecture include:

- **A single source of truth:** A single copy of data serves BI, AI, and streaming use cases simultaneously. When data updates, all consumers see the change immediately without waiting for data pipelines to propagate changes across multiple copies (eliminating the data freshness lag that plagues traditional architectures).
- **Consistent governance:** Apache Ranger (part of Cloudera's unified fabric) provides unified security and governance across your data platform. Cloudera's data fabric ensures

consistent policies apply regardless of which engine accesses your data. Ranger enforces policies at the engine level so that engines must be integrated with Ranger to provide table, row, and column-level access controls. You define access controls once, and they apply to all integrating engines accessing the data.

Provides standards-based APIs that support tools like Athena, Databricks, Redshift, and Snowflake — enabling interoperability without vendor lock-in.

- **Reduced operational complexity:** Zero-copy eliminates the overhead of maintaining multiple data pipelines, managing duplicate storage, and reconciling inconsistencies across data copies. Teams spend less time on data movement and more time on analysis and innovation.

REMEMBER

Zero-copy is not just about avoiding copies. It's about governance consistency and reproducibility. When all workloads access the same dataset, you eliminate the risk of different teams working with different versions of the "truth." Reports, ML models, and operational dashboards all work from identical data, ensuring consistency across your organization.

Implementing cross-platform interoperability and security

The REST catalog serves as the universal translator between different compute engines and Iceberg tables. By standardizing metadata access through REST-based APIs ensures that governance remains consistent for Iceberg REST-enabled tools accessing your data.

The REST catalog achieves this standardization through several mechanisms:

- **Standard APIs for all engines:** The REST catalog exposes a consistent REST-based API that all engines use to discover tables, read metadata, and coordinate commits. Whether you're using Spark, Trino, or Flink, the catalog interaction follows the same protocol.
- **Engine-level security enforcement:** Rather than configuring engine-level security separately for each tool, security policies are centrally defined but enforced at each integrated

engine. Apache Ranger policies are applied when engines query data, ensuring consistent security across all access patterns regardless of the query tool.

- **Consistent access across hybrid and multi-cloud deployments:** The REST catalog works identically whether your data resides on-premises, in AWS, Azure, or across multiple clouds. This consistency simplifies operations and ensures uniform governance everywhere.
- **Secure token management:** The REST catalog uses OAuth authentication through Apache Knox to issue time-limited access tokens for third-party engines such as AWS Athena, Databricks, and Snowflake. This token-based approach eliminates sharing long-lived credentials with external systems while Ranger continues enforcing consistent access policies across all engines.

Organizations can have multiple catalogs for different teams, but each Iceberg table should only be tied to one catalog (not multiple) as using multiple catalogs for a single table can violate ACID guarantees and lead to consistency issues.

Interoperating with Multiple Engines

Zero-copy only works if all engines can interoperate through the REST catalog. Cloudera has made Iceberg a first-class citizen across Spark, Hive, Trino, Flink, and Impala, ensuring seamless multi-engine access.

Leveraging REST catalogs for direct access

Direct access means any supported engine can immediately query Iceberg tables without intermediate data pipeline steps. The REST catalog provides a standardized set of APIs to handle metadata operations, schema discovery, and access control transparently, making multi-engine access seamless for users.

TIP

When demonstrating interoperability, show the same table being read by two different engines in the same demo.

Enabling multi-engine access and integration

True interoperability means more than just being able to read the same files; it also encompasses the ability to exchange data seamlessly and efficiently. Different engines must see consistent snapshots, respect the same security policies, and coordinate updates without conflicts. The REST catalog orchestrates this complex interaction behind a simple interface:

- **Consistent snapshot views:** Iceberg's table format ensures all engines see consistent snapshots. When Spark writes new data, a new snapshot is created. Trino queries automatically see newly appended data without manual refresh. (*Note:* Cache invalidation occurs when snapshot files expire or manifests are rewritten, which affects query caching.)
- **Unified governance:** Hive tables read and Impala queries with identical governance policies. Security rules defined in Ranger are applied consistently, regardless of the SQL engine the user prefers to use.
- **Streaming and batch coordination:** Flink streams continuously append data while Spark batch jobs consume without conflicts. Iceberg's snapshot isolation ensures that long-running batch queries see a consistent view even as streaming writes commit new data.

Applying zero-copy data collaboration across industries

Organizations across industries are discovering that zero-copy patterns solve long-standing data silo problems. By centralizing data access through REST catalog and Iceberg, they're breaking down barriers between analytics, AI, and operational workloads.

Here's how real-world deployments demonstrate zero-copy value across sectors:

- **Finance:** Analysts, AI teams, and auditors all query the same Iceberg table through the standardized REST catalog interface, without creating separate copies across different catalog implementations. Risk models train on the same transaction data that powers compliance reports, ensuring consistency and reducing storage costs. When regulatory questions arise, the lineage from raw transactions to risk scores is clear because everyone works with the same source data.

- **Healthcare:** Patient cohorts are accessed by both BI dashboards and ML training pipelines. Clinical researchers can identify patient populations using the same data that powers operational dashboards, without creating compliant copies for each use case. This unified access accelerates research while simplifying compliance.
- **Telecom:** Real-time network metrics ingested via Flink are immediately queryable by Trino dashboards. Network operations teams see the same metrics that feed capacity planning models and billing systems, ensuring consistent reporting across the organization. Zero-copy access eliminates the lag time that previously delayed problem detection.

Demonstrating Cost and Performance Advantages

Zero-copy delivers measurable business value that resonates with both technical teams and executives. The combination of reduced infrastructure costs, faster time-to-insight, and improved governance creates a compelling ROI story that can justify migration investments.

Measuring zero-copy ROI and operational efficiency

Quantifying the benefits of zero-copy requires looking beyond storage savings to include operational efficiency, reduced complexity, and faster innovation cycles. Cloudera's benchmarks provide concrete metrics that you can use to build your business case:

- **Reduce data pipelines.** Once the REST catalog is adopted, many data movement pipelines become unnecessary. Teams can query source data directly rather than creating specialized copies, eliminating the code, infrastructure, and maintenance overhead of those pipelines.

- **Onboard new workloads faster:** When new teams need access to data, they simply query existing Iceberg tables through the REST catalog. No data pipeline development, no data copying, no waiting for batch jobs to complete. What previously took weeks now happens in hours.
- **Improved compliance:** Lineage and policies apply uniformly across all engines and access patterns. Audit requirements are simpler to satisfy because there's only one data lineage to track instead of multiple data pipeline chains with potential inconsistencies.

Managing operational costs and optimizing storage

While zero-copy eliminates duplication, it still requires operational discipline. You need to manage catalog performance, monitor access patterns, and maintain Iceberg tables to make sure you get all the benefits.

Zero-copy patterns reduce several cost categories:

- **Storage sprawl from multiple table copies:** Instead of maintaining separate tables for BI, data science, and operations, one table serves all purposes. As data volumes grow, this consolidation produces exponential savings.
- **Compute waste from redundant data pipelines:** Data pipeline jobs that copy data between systems consume significant resources. Eliminating these jobs frees compute capacity for value-adding workloads, such as analytics and ML training.
- **Governance gaps from siloed security policies:** When each copy has separate access controls, security becomes fragmented and error-prone. Centralized governance through the REST catalog reduces both security risks and administrative overhead.

TECHNICAL STUFF

The REST catalog integrates with Apache Ranger to ensure all engines query with unified security policies and audit logging. A single Ranger policy controls access regardless of which query tool is used, providing consistent governance across all Iceberg REST-enabled engines.

The REST catalog transforms Iceberg into a true open data lakehouse, unifying data access without duplication. By eliminating data pipeline overhead and enabling seamless multi-engine access, zero-copy patterns deliver both immediate cost savings and long-term operational advantages.

IN THIS CHAPTER

» **Planning hybrid deployments with governance**

» **Implementing cloud-to-on-premises patterns**

Chapter 6
Deploying Hybrid with Cloudera and AWS

Most enterprises won't move entirely to the cloud overnight, and there's no reason they should. You likely have significant investments in on-premises infrastructure and face regulatory requirements that mandate data residency. The challenge is finding an approach that lets you migrate gradually while maintaining operational continuity. What you need is a hybrid architecture that seamlessly combines cloud and on-premises resources.

Apache Iceberg, paired with the Cloudera platform becomes the foundation that makes secure, governed, and flexible hybrid deployments actually work. Cloudera's unified data fabric layer ensures your security and metadata management stay consistent everywhere. This unified approach means that Ranger policies and Atlas lineage work exactly the same whether your data sits on-premises or on AWS.

This chapter explores how to plan hybrid architecture, avoid vendor lock-in, manage deployment risks, and effectively integrate AWS with Cloudera's platform.

Understanding Hybrid Deployment Strategy

Cloudera's hybrid strategy lets you put workloads where they make sense based on performance needs, costs, and compliance requirements. The platform provides centralized visibility and management across your on-premises and cloud data lakes through a unified control plane. While these remain physically separate environments with independent storage and compute, Cloudera enables you to monitor and manage them from a single interface, reducing operational complexity for your teams.

Planning hybrid architecture and avoiding vendor lock-in

One of the primary concerns with any platform investment is vendor lock-in. When your data and processes become dependent on proprietary formats or closed ecosystems, you lose strategic flexibility and negotiating power. By using open table formats like Apache Iceberg, Cloudera ensures your data remains portable and accessible by any compatible engine. The REST Catalog provides a vendor-neutral interface for table metadata, preventing lock-in to proprietary catalog implementations. This architecture separates compute from storage, allowing you to move workloads between environments without format conversion or vendor-specific dependencies. Your data remains under your control regardless of where you choose to process it. You can:

- **Keep sensitive data on-premises for compliance.** Cloudera's unified fabric provides unified governance, ensuring that your data governance policies work the same everywhere, so you can meet data residency requirements while keeping unified access control.
- **Use AWS for elasticity and burst workloads.** Cloudera on AWS gives you elastic compute resources that scale automatically based on workload demands, while maintaining integration with on-premises clusters.
- **Apply consistent governance everywhere.** The unified data fabric layer centralizes your security policies through Ranger, data classification through Atlas, and audit logging across all hybrid environments. Policies defined once are enforced everywhere.

TIP

Cloudera architecture separates compute from storage and uses open table formats like Apache Iceberg as the foundation for open data architecture, ensuring data portability. You can move workloads between environments without format conversion or pipeline rewrites. The REST Catalog provides a vendor-neutral interface for table metadata, preventing lock-in to proprietary catalog implementations. This design gives you genuine flexibility your data remains under your control regardless of where you choose to process it.

Managing strategic deployment and risk considerations

While hybrid deployment offers significant flexibility, it also brings complexity that you need to manage carefully. You need to balance the benefits of distributed infrastructure with the operational challenges of maintaining consistency across different environments. Cloudera's governance framework addresses these challenges systematically. Key risks in hybrid deployments include:

- **Policy drift across environments:** Without centralized governance, security policies and access controls can diverge between on-premises and cloud environments over time. Cloudera's unified data fabric prevents this fragmentation by centralizing governance. Ranger's format-agnostic policies work consistently across both Hive and Iceberg tables in all environments. You define governance policies centrally through the fabric, and each environment enforces those policies locally. While policies must be configured for each environment, Cloudera's unified data fabric provides a consistent framework and tooling to maintain policy alignment across your hybrid deployment.
- **Cost overruns from ungoverned cloud workloads:** Cloud resources can scale quickly, and without proper controls, costs can escalate unexpectedly. Cloudera includes workload management capabilities that enforce resource quotas and automatic shutdown policies. For example, you can configure development clusters to automatically shut down after business hours or set budget alerts that trigger when spending approaches defined thresholds.

- **Latency in cross-environment queries:** When queries need to access data across multiple environments, network latency can impact performance. Cloudera optimizes data locality and implements intelligent caching to minimize this overhead. When workloads need to access data, queries execute on the data lake where users initiate them. Cloudera optimizes performance through intelligent caching and data placement strategies.

  The platform implements data locality principles, processing data where it resides whenever possible, and strategically caches frequently accessed data to reduce repeated transfers. Organizations planning cross-environment data access should consider using Cloudera's Replication Manager to copy data to the environment where it will be queried, rather than relying on cross-network data access which introduces latency.

TIP

Cloudera's migration blueprint recommends a phased approach of 8–12 weeks that systematically addresses these risks. The blueprint includes governance alignment in weeks 3–4 and validation phases in weeks 7–8 to make sure that policies work correctly everywhere.

Presenting strategic flexibility benefits

The true value of hybrid deployment shines when organizations need to respond quickly to changing business requirements. Whether facing new compliance regulations, seasonal workload spikes, or budget constraints, hybrid architecture provides options that pure cloud or on-premises deployments cannot match.

Hybrid with Cloudera delivers:

- **Elastic scaling:** You can burst workloads to AWS without re-architecting. Cloudera automatically provisions and deprovisions compute resources based on the needs of your workloads, optimizing resource utilization.
- **Sovereignty compliance:** You can keep restricted data on-premises. You can enforce data locality requirements through the unified fabric while still running unified analytics across hybrid environments.

- **Unified security:** Ranger and Atlas span across all clusters. Security policies and data lineage tracking work consistently across on-premises HDFS and cloud object storage.
- **Metadata-only migration:** Cloudera's approach often lets you convert tables from Hive to Iceberg without moving data, cutting migration time from days to minutes.

Designing Hybrid and Multi-Cloud Patterns

Cloudera supports multiple deployment patterns, allowing you to choose one based on your specific requirements. These patterns leverage Iceberg's open format to enable seamless data access across environments.

Implementing deployment patterns and architecture

Several proven hybrid patterns have emerged from enterprise implementations. Understanding these patterns helps you design an architecture that matches your operational needs. There are several hybrid patterns:

- **Analytics in the cloud, storage on-prem:** Cloudera compute clusters in AWS can query Iceberg tables stored in on-premises HDFS through the unified catalog.
- **Ingest in AWS, process on-prem:** Data collected in Amazon S3 can be processed by on-premises Spark clusters, with Iceberg providing a consistent table format.
- **Bi-directional pipelines between HDFS and Amazon S3:** ClouderaReplication Manager automates Iceberg table replication between environments, preserving snapshots, metadata, and schema evolution history. This enables disaster recovery scenarios and hybrid data locality optimization. Unlike traditional ETL-based replication, Replication Manager understands Iceberg's metadata structure and can perform incremental snapshot-based replication efficiently.

TECHNICAL STUFF

Amazon S3 integrates with Iceberg tables natively. Cloudera provisions secure connectors to read/write Iceberg tables stored in Amazon S3 with governance intact. For scale, enterprises can use S3FileIO and ObjectStoreLocationProvider to manage prefixes and avoid performance bottlenecks.

Managing operations and global performance

Running hybrid environments means coordinating across teams, time zones, and infrastructure boundaries. Success depends on unified monitoring, consistent policy enforcement, and intelligent workload placement. Cloudera provides the operational framework to manage this complexity, allowing your end users to remain unaware of it. Cloudera emphasizes:

- **Centralized monitoring across environments:** Cloudera Manager provides unified monitoring for both on-premises and cloud clusters, consolidating metrics and logs in a single interface for either on-premises clusters or cloud environments.
- **Policies that replicate globally:** Ranger policies automatically synchronize across regions and environments through unified policy replication.
- **Performance optimization with data locality plus caching strategies:** Clouderaimplements intelligent caching and data placement strategies to minimize cross-environment data movement.

TIP

Cloudera's workload management features include automatic resource allocation, query prioritization, and cost-based optimization for hybrid deployments. Workload metrics are collected centrally to enable visibility and performance optimization.

Cloudera's workload management framework focuses on optimizing query and job performance across hybrid environments. It collects execution metrics, analyzes resource usage, and provides predictive insights to improve efficiency. The workload management framework centralizes workload data from multiple clusters — on-premises and in the cloud — so that the administrators can visualize performance trends, uncover bottlenecks, and fine-tune configurations for optimal throughput and cost control.

REMEMBER

Hybrid isn't just a technical architecture. It's about operating seamlessly at a global scale under one governance model. The goal is to provide your users with a unified data platform experience while giving your operations team centralized control and visibility.

Deploying Iceberg on AWS with Cloudera

Getting started with Cloudera on AWS follows a structured deployment process that automates most of the infrastructure complexity. The platform handles the technical cloud configuration details while keeping the governance and security models familiar from on-premises deployments. The deployment process includes these key steps:

- **Provision Cloudera on AWS.** Cloudera's automated provisioning creates Amazon Virtual Private Cloud's (VPCs), subnets, security groups, and AWS Identity and Access Management (IAM) roles according to AWS best practices.
- **Connect to Amazon S3 buckets for Iceberg storage.** Cloude configures Amazon S3 access with the right Amazon IAM policies and encryption settings. The platform supports options including Amazon SSE-S3, Amazon SSE-KMS, and Amazon DSSE-KMS for data at rest protection.
- **Enable Ranger plus Atlas policies in Cloudera.** Your existing policies from on-premises deployments can be imported and applied to cloud resources through the unified fabric layer.
- **Run workloads with Spark, Hive, and Trino directly on Iceberg tables.** Cloudera includes optimized versions of these engines with native Iceberg support. Users query tables using standard SQL syntax.

Cloudera's deployment automation handles network configuration, security group setup, and the creation of Amazon IAM roles. The platform implements AWS security best practices, including Amazon VPC isolation, encryption at rest, and encryption in transit.

IN THIS CHAPTER

» Boosting efficiency and cutting costs

» Strengthening governance and agility

» Future-proofing enterprise data

Chapter 7 Eleven Ways Apache Iceberg Migration Benefits Your Business

Migrating from Hive table format to Apache Iceberg isn't just a technical improvement; it creates a substantial business value.

- **Boosts performance and query efficiency (for Analysts, Data Scientists, and Data Engineers):** Iceberg tables use metadata pruning instead of full directory scans, so queries run much faster and cost less. Analysts and data scientists can get answers in minutes instead of hours. Data engineers benefit from faster pipelining, reduced compute costs, and easier pipeline management.
- **Avoids vendor lock-in with open standards:** Unlike proprietary formats, Iceberg is open and broadly supported. Organizations can choose the engines and platforms that best fit their strategy without fear of lock-in.

- **Reduces storage and infrastructure costs:** By consolidating small files and eliminating redundant data copies, Iceberg reduces storage requirements and associated infrastructure costs compared to traditional data warehouse solutions. Optimized queries also lower compute expenses.
- **Enables reproducible AI and ML pipelines (for Data Scientists and ML Engineers):** Capture the exact dataset used to train a model, ensuring reproducibility for auditing and troubleshooting. Branches let teams safely experiment with new transformations on production data without risk, eliminating the need for separate development datasets.
- **Ensures successful migration with automated data lineage (for Platform Teams and Data Engineers):** Cloudera Octopai Data Lineage provides automated data discovery and lineage tracking to map your data estate before migration begins. By understanding dependencies and downstream impacts, teams can prioritize migration efforts and validate success, ensuring business continuity throughout the transition.
- **Strengthens compliance and governance:** Integration with Cloudera unified data fabric (Ranger and Cloudera Octopai for Data Lineage) enforces unified consistent access policies and lineage tracking. This ensures data is secure, governed, and auditable across the enterprise.
- **Simplifies schema evolution and data agility (for Data Engineers and Platform Teams):** Iceberg allows you to add, drop, or rename columns without rewriting data or interrupting jobs. Pipelines can adapt quickly to changing business requirements.
- **Unifies batch and streaming in one table format (for Data Engineers):** Iceberg supports both batch data pipeline and real-time streaming ingestion. Apache Spark, Apache Flink, Apache NiFi, and other engines can all use the same data eliminating silos.
- **Empowers hybrid and multi-cloud flexibility:** With Cloudera, Iceberg works consistently across data centers, private cloud, and public cloud, with Cloudera's unified data fabric ensuring uniform governance across all environments.
- **Accelerates time-to-insight through zero-copy data collaboration (for Analysts and Business Users):** Cloudera's Iceberg REST Catalog enables interoperability and collaboration across heterogeneous environments including

Snowflake, AWS EMR, and Databricks. Analysts can query data directly in the lakehouse with Iceberg tables instead of waiting for data engineering teams to make it available through time-consuming database jobs from lake to warehouse. This eliminates costly data copies and data pipelines, providing faster access to current data for quicker, better-informed business decisions.

- **Future-proofs your data strategy with Iceberg:** Open standards, hybrid cloud flexibility, and AI readiness make Iceberg a long-term foundation for enterprise data.

Modernize your data lake with Apache Iceberg

Data lakes have come a long way since the days of monolithic, batch-driven pipelines. Apache Hive laid the groundwork for scalable analytics. Iceberg, an open, Apache-licensed table format, goes beyond Hive table formats. It delivers real-time analytics, AI model training, and consistent governance across hybrid environments. Find out the benefits to your business with this practical guide and make a seamless transition to Apache Iceberg.

Inside...

- Why migrate to Apache Iceberg
- Step-by-step guide to migration
- Unify batch and streaming pipelines
- Drive AI with trust and collaboration
- Accelerate analytics across diverse systems
- Centralize security, governance, and metadata for trusted AI

CLOUDERA

Stephanie Diamond is a marketing innovator and author. **Akshat Mathur** is a Product Manager for Cloudera's open data lakehouse and frequent open-source contributor.

ISBN: 978-1-394-41282-2
Not For Resale

9 781394 412822

for dummies®
A Wiley Brand