

Apache Iceberg with Snowflake

Production Readiness Guide

Ownership, Catalogs, Compatibility, Operations, Cost, and Go/No-Go

Vertex Frontier
Production Decision Toolkit
2026 Edition

Contents

1	How to use this guide	1
2	1. The ten-minute architecture decision	1
	2.1 Pick a starting model	2
3	2. Catalog and storage ownership	3
	3.1 Minimum evidence to capture	3
4	3. Verified implementation sequence	3
	4.1 3.1 Create the external volume: AWS S3 example	3
	4.2 3.2 Configure a generic Iceberg REST catalog	4
	4.3 3.3 Verify catalog and storage independently	4
	4.4 3.4 Create a catalog-linked database	4
	4.5 3.5 Create the linked table using the documented context	5
5	4. Compatibility gate: v2, v3, and delete representations	6
	5.1 Test matrix	6
6	5. Maintenance control plane	6
7	6. Benchmark and cost protocol	7
	7.1 Benchmark dimensions	7
	7.2 Distributed cost model	8
8	7. Go / No-Go checklist	8
9	References	9

1 How to use this guide

This guide is the companion decision tool for the Vertex Frontier article on Apache Iceberg with Snowflake. It is designed for a data-platform review, architecture workshop, or production-readiness gate. The objective is not to persuade a team to use Iceberg. The objective is to make the ownership model explicit before the first critical table is created.

> **The table format is only one part of the architecture. The catalog, storage path, credentials, maintenance owner, and external-engine contract determine whether the design remains operable after the demo.**

Use the editable workbook beside this PDF to record the actual decisions. This guide supplies the mental model and the review sequence; the workbook supplies the working evidence tables.

The shortest safe sequence: choose the ownership model, choose the catalog, verify storage and catalog access separately, test every intended reader and writer, measure the real workload, and only then approve production.

2 1. The ten-minute architecture decision

Snowflake can work with Iceberg in more than one way. The decision is not simply whether Snowflake “supports Iceberg.” The decision is which system owns each control-plane layer.

Layer	What it controls	Decision question
Data files	Parquet data and delete files	Which cloud account, bucket, container, or storage service owns the bytes?
Metadata	Schemas, snapshots, manifests, properties, and pointers	Which system can commit a new table state?
Catalog	Table identity and current metadata pointer	Where do all engines discover the current table?
Compute	Planning, scans, transformations, and DML	Which engine pays for and performs the work?
Credentials	Snowflake objects, catalog, storage, and vended access	Which identities are scoped, short-lived, and auditable?
Maintenance	Expiry, compaction, manifests, cleanup, and refresh	Who notices and repairs deterioration?
Governance	Roles, policies, tags, masking, and audit boundaries	Which system enforces each policy?
Billing	Compute, storage, API, optimization, and transfer	Which provider charges for each access path?

2.1 Pick a starting model

Model	Use it when	Main trade-off
Snowflake-managed	Snowflake should remain the catalog, principal writer, and governance surface.	Simpler Snowflake-centered control, but external-engine access still requires compatibility and credential testing.
External REST catalog	A remote catalog already owns the open-table ecosystem.	Multi-engine ownership is preserved, but catalog, refresh, and maintenance remain part of the external operating model.
Catalog-linked database	Snowflake should discover and work with remote namespaces through a supported REST integration.	Convenient discovery and supported writes require careful drop, purge, credential, and ownership rules.
Horizon access	Snowflake-managed tables must be reached by external engines.	Snowflake remains catalog authority, while engine authentication, privileges, and storage credentials widen the security surface.

Decision rule: if Snowflake is the only consumer and a native table or Snowpipe pipeline meets the storage and governance requirements, compare that simpler path first. Iceberg earns its complexity when open storage, shared tables, external engines, sovereignty, or an existing Iceberg catalog are real requirements.

3 2. Catalog and storage ownership

A catalog is not just a directory of table names. It exposes the current table state and coordinates the commit path. An external volume is not a catalog. It is the Snowflake object that connects Snowflake to cloud storage and its access identity.

Object	Owns or exposes	Do not confuse it with
Catalog integration	How Snowflake discovers and communicates with a remote catalog.	The storage location or the Snowflake warehouse.
Catalog-linked database	A Snowflake database object linked to remote namespaces and tables.	A second, independent copy of the table identity.
External volume	Storage base URL, cloud identity, and read/write access.	The Iceberg catalog or metadata commit authority.
Horizon Catalog	A REST access surface for external engines to reach Snowflake-managed tables.	Moving catalog ownership to Spark, Trino, or another engine.

3.1 Minimum evidence to capture

Before creating a production table, record the storage provider, region, base path, encryption, recovery/versioning policy, private-connectivity requirement, IAM role or service principal, Snowflake role privileges, catalog endpoint, namespace, authentication mode, and whether the path is read-only or writable.

The workbook's Ownership Matrix is intentionally repetitive: a responsibility that is not assigned in writing will usually become an incident during the first refresh failure, failed commit, or cleanup event.

4 3. Verified implementation sequence

The following sequence is a deployment template, not a claim that placeholders can be left unchanged. Replace the endpoint, namespace, bucket, role ARN, and secret-management values. Test the provider-side trust policy before enabling writes.

4.1 3.1 Create the external volume: AWS S3 example

```
CREATE OR REPLACE EXTERNAL VOLUME ICEBERG_EXTERNAL_VOLUME
  STORAGE_LOCATIONS = (
    (
      NAME = 'primary_s3'
      STORAGE_PROVIDER = 'S3'
      STORAGE_BASE_URL = 's3://<BUCKET>/<PREFIX>/'
      STORAGE_AWS_ROLE_ARN = '<S3_IAM_ROLE_ARN>'
      ENCRYPTION = ( TYPE = 'AWS_SSE_S3' )
    )
  )
  ALLOW_WRITES = TRUE;
```

For Azure, GCS, or S3-compatible storage, use the provider-specific parameters in the current Snowflake external-volume reference. If the path is read-only, use `ALLOW_WRITES = FALSE` and do not enable catalog-linked writes.

4.2 3.2 Configure a generic Iceberg REST catalog

```
CREATE OR REPLACE CATALOG INTEGRATION EXT_REST_ICEBERG_INT
  CATALOG_SOURCE = ICEBERG_REST
  TABLE_FORMAT = ICEBERG
  CATALOG_NAMESPACE = '<REMOTE_NAMESPACE>'
  REST_CONFIG = (
    CATALOG_URI = '<REST_CATALOG_BASE_URL>'
    ACCESS_DELEGATION_MODE = EXTERNAL_VOLUME_CREDENTIALS
  )
  REST_AUTHENTICATION = (
    TYPE = BEARER
    BEARER_TOKEN = '<BEARER_TOKEN_SECRET>'
  )
  ENABLED = TRUE
  REFRESH_INTERVAL_SECONDS = 60;
```

Keep bearer tokens, OAuth secrets, and signing credentials in the approved secret-management workflow. Do not commit them to source control.

4.3 3.3 Verify catalog and storage independently

```
SELECT SYSTEM$VERIFY_CATALOG_INTEGRATION('EXT_REST_ICEBERG_INT');
SELECT SYSTEM$VERIFY_EXTERNAL_VOLUME('ICEBERG_EXTERNAL_VOLUME');
```

A successful catalog check does not prove that Snowflake can read or write the configured storage location. Treat the two checks as separate gates.

4.4 3.4 Create a catalog-linked database

```
CREATE OR REPLACE DATABASE ICEBERG_REST_DB
  CATALOG = EXT_REST_ICEBERG_INT
  EXTERNAL_VOLUME = ICEBERG_EXTERNAL_VOLUME
  ALLOWED_NAMESPACES = ('<REMOTE_NAMESPACE>')
  ALLOWED_WRITE_OPERATIONS = ALL
  CATALOG_CASE_SENSITIVITY = CASE_INSENSITIVE
  SYNC_INTERVAL_SECONDS = 60;
```

`ALLOWED_WRITE_OPERATIONS = ALL` is consequential. Use a disposable namespace first and test table replacement, DROP, PURGE, and recovery behavior before granting broad write permission.

4.5 3.5 Create the linked table using the documented context

```
USE DATABASE ICEBERG_REST_DB;
USE SCHEMA '<REMOTE_NAMESPACE>';

CREATE OR REPLACE ICEBERG TABLE events (
  event_id  STRING,
  event_ts  TIMESTAMP_NTZ,
  payload   VARIANT
)
PARTITION BY (DAY(event_ts))
STORAGE_SERIALIZATION_POLICY = COMPATIBLE;

SELECT * FROM events LIMIT 10;
```

Use `STORAGE_SERIALIZATION_POLICY = COMPATIBLE` when third-party engines must read Snowflake-written files. The current Snowflake reference distinguishes this interoperability setting from the Snowflake-oriented `OPTIMIZED` path.

Destructive-operation warning: a writable catalog-linked database is not a harmless mirror. Depending on the path and command, a remote drop or purge can affect the catalog and underlying data. Test on disposable data, record the table identifier, and document the recovery owner.

5 4. Compatibility gate: v2, v3, and delete representations

The label “v3 supported” is not a complete compatibility statement. A production decision must name the exact Snowflake behavior, external engine version, Iceberg library, delete representation, credential path, and workload.

Gate	Record	Failure mode if skipped
Format version	Iceberg v2 or v3 for every reader and writer.	One engine can read the table while another fails after a feature is used.
Delete representation	Position deletes, equality deletes, or v3 deletion vectors, as applicable.	Reads silently fail, become slower, or return incompatible results.
Serialization	COMPATIBLE or OPTIMIZED according to the external-engine requirement.	Third-party readers cannot interpret files written under the selected policy.
Nested and timestamp types	Exact data types and timezone behavior per engine.	Schema appears compatible while values change meaning.
Write mode	Copy-on-write or merge-on-read for each writer.	Snowflake’s setting is mistaken for control over Spark or Trino.
Refresh behavior	Automatic or manual refresh interval and operator.	Snowflake reads a stale snapshot after external commits.

5.1 Test matrix

For each intended engine, record: engine version, Iceberg library version, table format version, read capability, write capability, delete representation, authentication method, vended-credential behavior, and a real observed result. “Unknown” is not the same as “Yes.”

Reader / writer	Version	Can read?	Can write?	Evidence to capture
Snowflake SQL	account release	Yes / No	Yes / No	query output, DML result, table mode, delete behavior
Spark	engine + Iceberg lib	Yes / No	Yes / No	catalog client, credentials, v2/v3, files touched
Trino	engine + Iceberg lib	Yes / No	Yes / No	REST client, endpoint, role, delete behavior
Flink	engine + Iceberg lib	Yes / No	Yes / No	streaming commit and recovery test
DuckDB / other	engine + extension	Yes / No	Yes / No	read path, file compatibility, limitations

6 5. Maintenance control plane

A table can be queryable and still be operationally unhealthy. Track snapshot age, metadata growth, manifest counts, file counts, delete files, compaction history, refresh lag, failed commits, and storage growth.

Operation	Snowflake-managed question	External-catalog question	Evidence / owner
Snapshot expiry	Which documented Snowflake feature and retention policy applies?	Which engine or catalog expires snapshots?	owner, schedule, retention, last run
Metadata cleanup	What does Snowflake retain and expose?	Who removes old metadata JSON and manifests?	policy and storage trend
Data-file compaction	Is optimization enabled, scheduled, or billable?	Which engine rewrites files and under what threshold?	file-size distribution
Manifest work	Is manifest compaction supported for this mode?	Who rewrites manifests after external writes?	planning-time trend
Orphan cleanup	Is the operation supported for this table mode?	What retention protects in-flight writers?	dry run and safety window
Refresh	How does Snowflake see the new snapshot?	Who triggers or monitors refresh?	refresh lag and alert
Failed commit	How is the last valid snapshot identified?	Which catalog and storage system owns recovery?	runbook and test result

> **Orphan cleanup is not a generic cron job.** If the retention interval is shorter than the time required for an in-flight writer to commit, cleanup can remove files that a writer still expects to use.

7 6. Benchmark and cost protocol

Do not publish “Iceberg is faster” or “external storage is cheaper” without workload conditions. A Snowflake warehouse reading nearby storage is not the same experiment as Trino reading across a region or an on-premises S3-compatible endpoint.

7.1 Benchmark dimensions

Dimension	Record	Why it changes the result
Data shape	rows, columns, row width, nulls, types	compression, scan volume, and metadata statistics
File layout	count, target size, partitions, clustering, delete files	planning work and pruning quality
Workload	filters, point lookups, joins, aggregates, MERGE, UPDATE, DELETE	read/write behavior differs by format and mode

Dimension	Record	Why it changes the result
Commit pattern	batch size, frequency, concurrency, retries	snapshots, manifests, small files, conflicts
Environment	warehouse, cloud, region, engine/library, network, cache	compute and transfer effects are separated
Metrics	p50/p95, planning, bytes, files, compaction, transfer, failed commits	prevents one favorable query becoming a universal claim

7.2 Distributed cost model

Separate the estimate into Snowflake warehouse compute, Snowflake cloud services or catalog/API charges, cloud storage and retained snapshots, compaction and optimization, external-engine compute, cross-region or cross-cloud transfer, and engineering time for IAM, compatibility, observability, and recovery.

The relevant comparison is not “Snowflake storage versus S3 storage.” It is where bytes live, where compute runs, how often files are rewritten, which catalog APIs are called, and how many teams must understand the table.

8 7. Go / No-Go checklist

Approve production only when the team can answer each question with evidence.

No.	Gate	Evidence expected
1	The authoritative catalog is named.	catalog integration, table identity, commit owner
2	The storage path and recovery policy are verified.	external-volume output, IAM trust, versioning, encryption, recovery test
3	All intended readers and writers are compatible.	engine matrix, format version, delete representation, test result
4	Credentials and governance boundaries are documented.	roles, tokens, vended credentials, network, masking/policy scope
5	Maintenance has named owners and schedules.	expiry, compaction, manifests, orphan cleanup, refresh, failed-commit runbook
6	A representative benchmark is complete.	workload, p50/p95, bytes, files, transfer, maintenance, failed commits
7	The cost model is complete.	rates, units, transfer, external compute, engineering time
8	Rollback and recovery are tested.	last valid snapshot, stop-writes procedure, catalog/storage recovery steps

Proceed means the gates are evidenced under named conditions. It is not a warranty of performance, cost, or future vendor behavior outside those conditions. Re-run the gate when the catalog, table version, engine, storage region, or delete representation changes.

9 References

The workbook and guide are based on the current product and project documentation used in the companion article:

- Snowflake Apache Iceberg tables: <https://docs.snowflake.com/en/user-guide/tables-iceberg>
- Snowflake REST catalog integration: <https://docs.snowflake.com/en/sql-reference/sql/create-catalog-integration-rest>
- Snowflake catalog-linked database: <https://docs.snowflake.com/en/sql-reference/sql/create-database-catalog-linked>
- Snowflake REST CREATE ICEBERG TABLE: <https://docs.snowflake.com/en/sql-reference/sql/create-iceberg-table-rest>
- Snowflake CREATE EXTERNAL VOLUME: <https://docs.snowflake.com/en/sql-reference/sql/create-external-volume>
- Snowflake Horizon Catalog external-engine access: <https://docs.snowflake.com/en/user-guide/tables-iceberg-access-using-external-query-engine-snowflake-horizon>
- Snowflake Iceberg v3 support: <https://docs.snowflake.com/en/user-guide/tables-iceberg-v3-specification-support>
- Snowflake Iceberg table management: <https://docs.snowflake.com/en/user-guide/tables-iceberg-manage>
- Apache Iceberg specification: <https://iceberg.apache.org/spec/>
- Apache Iceberg maintenance: <https://iceberg.apache.org/docs/latest/maintenance/>

Vertex Frontier | Built for architecture reviews, not feature shopping