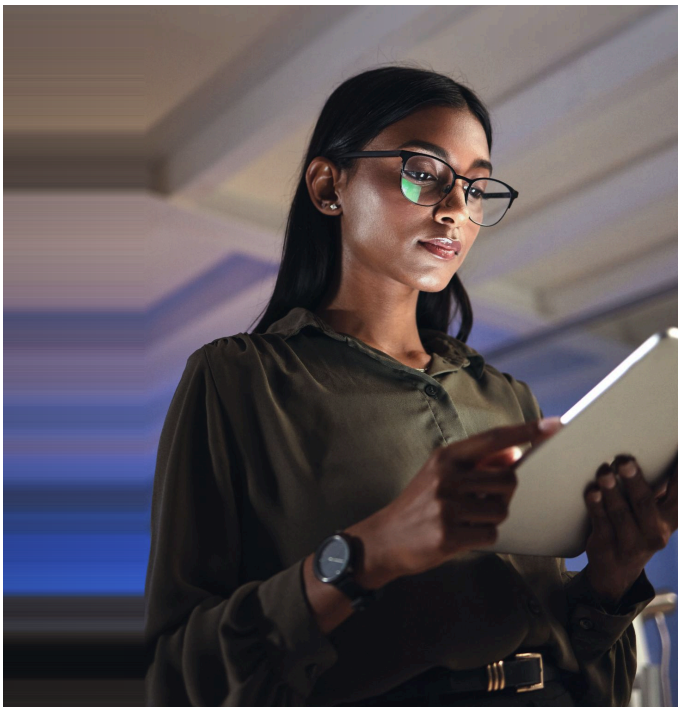
SingleStore + CONFLUENT an IBM Company

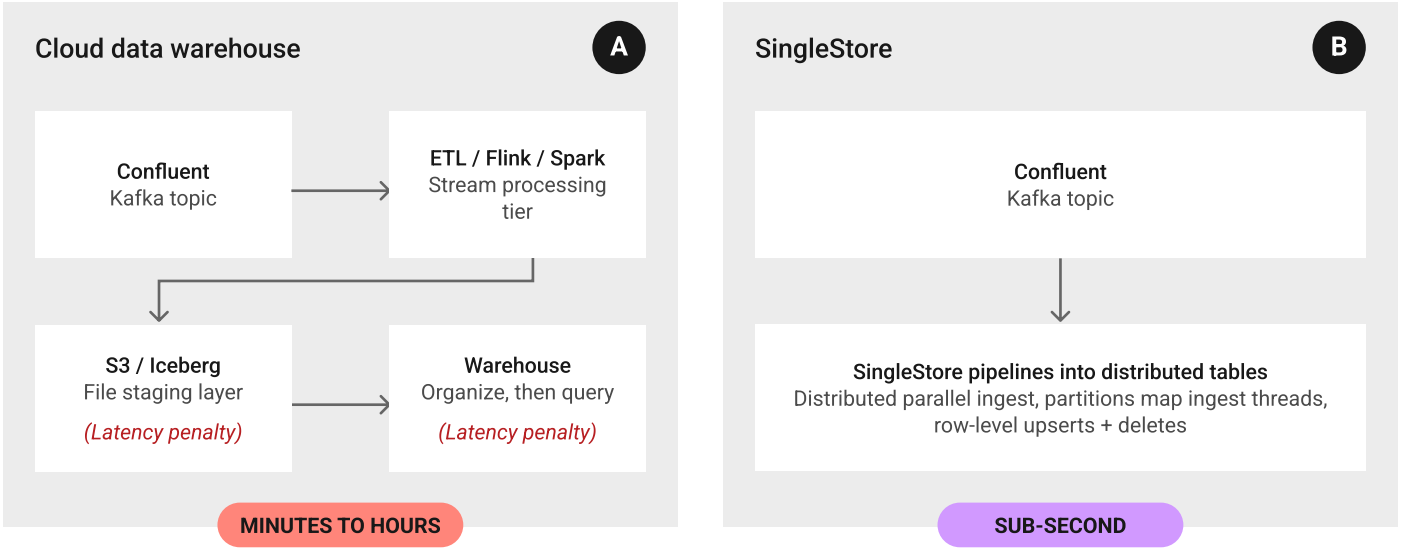
From Stream to Outcomes, in an Instant.

The real-time stack for live applications, analytics and AI

Confluent moves data the moment it is created. Most analytical destinations slow it down before use. In many cloud warehouse architectures, Kafka data passes through connectors, staging layers, object storage, and batch processing before it becomes queryable. It may be reshaped into Iceberg and processed with Flink, Spark, or dbt around the warehouse. Each step introduces latency. By the time the data is queryable, it is no longer real-time.



The question is simple: how do you preserve the freshness Confluent was built to deliver?

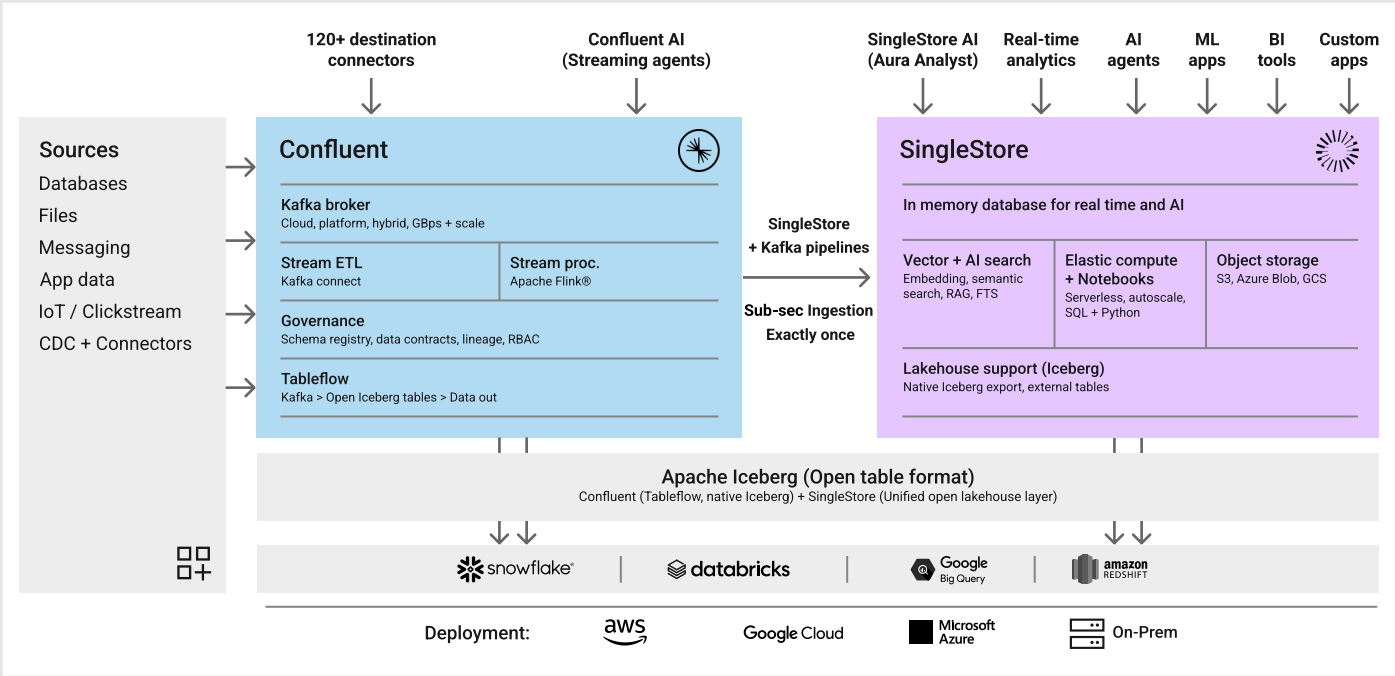


B) SingleStore is the answer. Paired with Confluent, it provides a live serving layer that ingests Kafka directly into distributed tables, making data queryable in sub-second time and available for operational, analytical, and AI workloads in one engine. This architecture is especially relevant when:

- **Kafka is already in production**, but ETL between Kafka and your database is brittle or difficult to maintain.
- **Your dashboards are fast, but the data is hours old,** or your AI applications need live data, not stale snapshots.
- **You want to augment** mainframe, Oracle, or SQL Server environments with a real-time analytical layer.
- **Your fraud, risk, or personalization workloads need to act** before the next event arrives.

What is SingleStore?

SingleStore is a distributed SQL database that runs transactional, analytical, vector, and full-text workloads side by side in a single engine, on the same live data, without moving data between systems. Built for sub-second decisioning at scale, it is available fully managed on AWS, Azure, and GCP, or self-managed. In this architecture, SingleStore serves as the live serving layer, while Iceberg connects that layer to the broader data estate, as shown in the diagram below.



SingleStore exports natively to Apache Iceberg, sending governed, enriched data downstream to lakes, warehouses, and data fabrics. The serving layer stays fast while the wider estate stays connected and consistent.

How SingleStore ingests Kafka

SingleStore ingests Kafka directly into distributed tables, in parallel across the cluster, so data becomes queryable without middleware, staging files, or batch reprocessing. Kafka partitions map to ingest threads, allowing throughput to scale horizontally as partitions and cluster capacity grow. Because pipelines are built into the database, SingleStore manages offsets, schema handling, and writes as part of the core system rather than through a separate ETL layer.

- Direct-to-table ingest eliminates staging delays
- Exactly-once delivery keeps data consistent under retries and failures
- Schema Registry support handles evolution cleanly
- Row-level upserts and deletes preserve operational state in place

Two ways to integrate

SingleStore Kafka Pipelines 1

Kafka flows directly into SingleStore tables for the lowest-latency path from stream to query.

- Direct ingest into SingleStore
- Sub-second freshness
- Exactly-once, schema-aware updates

Best when: The priority is live applications, dashboards, or real-time decisioning.

Confluent Tableflow + Iceberg 2

Tableflow materializes Kafka topics as Iceberg tables that SingleStore can read natively.

- Open lakehouse architecture
- External tables or scheduled ingest
- Lower long-term data engineering overhead

Best when: The long-term plan centers on Iceberg, with SingleStore serving fast workloads.

One engine for every workload

Most architectures split operational serving, analytics, vector retrieval, and text search across multiple systems. SingleStore consolidates those workloads into one engine, running against the same tables and the same live copy of data.

HTAP
Transactional and analytical workloads together run writes and analytical scans on the same live data, without ETL pipelines or replica sprawl.

Vector
AI retrieval built in run embeddings, semantic search, ANN, and RAG as first-class SQL on live data, without syncing a separate vector store.

Full-text
Text search in the engine search text alongside structured data in the same engine, and combine full-text and SQL filters in a single query.

JSON
Flexible schemas for real-world streams query JSON alongside relational data, so structured and semi-structured events can live together in one system.

How SingleStore compares

The architectural difference is most visible in the ingest path.

Capability	SingleStore	Snowflake	ClickHouse	Apache Pinot
Built for	HTAP: Operational + analytical workloads	Cloud data warehouse	OLAP analytics	Real-time OLAP
Kafka ingest pattern	Native distributed pipelines into tables	Kafka Connector / Snowpipe Streaming	Kafka engine + materialized views	Streaming into segments
Stream-to-query freshness	Sub-second	Streaming ingest, but warehouse / service-oriented	Append ingestion	Append ingestion
Row-level upserts on stream	In-place INSERT ... ON DUPLICATE KEY UPDATE / pipeline-driven	Typically downstream MERGE / task logic after ingest	Eventual reconciliation via MergeTree-family engines*	Supported with partitioning and upsert constraints**
Row-level deletes on stream	In-place deletes, DML-capable engine	Typically downstream DELETE / MERGE logic	Mutations / lightweight deletes; not ideal for high-frequency	Tombstone / delete semantics with constraints
Operational point lookups	Yes	Not designed for	Not designed for	Not designed for
Vector + FTS + JSON unified	In-engine	Available across warehouse + Cortex / Search services, less operationally unified	Limited	Limited

** **ClickHouse:** uses background-merge engines (ReplacingMergeTree, CollapsingMergeTree) reconciled with FINAL queries; eventually consistent rather than real-time. Per ClickHouse documentation. ** **Pinot:** upsert tables require partitioning by primary key, hold the PK map in JVM heap, cannot use the star-tree index, and skip out-of-order events. Per Apache Pinot documentation.*

Where this lands in production

Analytics
Live operational analytics dashboards reflect what happened a second ago, not four hours ago. Thousands of concurrent users can query the same live data without read-replica sprawl.

Decisioning
Fraud and risk in flight decisions can happen in the time it takes a transaction to clear, rather than in the next batch cycle. CDC streams update state in place, and queries return an answer immediately.

AI
Retrieval on live context vector search and RAG operate on data that includes the user's most recent actions. Confluent streams the context; SingleStore retrieves it; the application responds with current state, not yesterday's snapshot.

Modernization
A real-time layer for legacy environments events can be streamed from mainframe, Oracle, and SQL Server systems through Kafka into SingleStore, creating a real-time analytical and AI-ready layer without replacing the legacy system underneath.

Proven in production

SingleStore supports production deployments behind streaming infrastructure across banking, industrial, and B2B SaaS environments, including:

Banking + Financial services	Banking + Financial services	Industrial operations	Revenue intelligence
Goldman Sachs	Millennium bcp	SIEMENS	cognism.

> "Our dashboard is fast, but the numbers in it are four hours old."

> "We have Kafka in production, but downstream depends on custom Python and brittle ETL."

> "Our vector store and operational database keep drifting out of sync."

> "We need a real-time analytical layer alongside our mainframe, Oracle, or SQL Server systems."

> "Different teams keep producing different numbers from the same data."

> "Snowflake handles reporting, but we need something faster for live workloads."

If any of these quotes sound familiar...

The Confluent and SingleStore architecture is designed to preserve streaming freshness from ingest to action. **Talk to a SingleStore architect** to review your real-time stack, or **visit singlestore.com to start a free trial.**