streambased.

Frequently Asked Questions

Everything you need to know about Streambased

1. What Streambased is (and isn't)

Is Streambased a Database?

Streambased is not a database. Databases closely couple data storage, indexing and processing in order to provide a rigid query index for use by clients.

By implementing the Iceberg specification, Streambased allows users to bring their own processing engines and process data from either Kafka or Iceberg in a way that fits their requirements. In this way Streambased acts as the central hub, bringing together many of the components of a traditional database, but in a much more flexible way.

Is Streambased a Sink Connector?

Streambased is not a connector. The job of a sink connector is to facilitate the transfer of data from Kafka to a downstream system and to manage any transformation that must be applied during this process (in other words: an ETL).

Streambased takes a completely different approach, it's an abstraction layer that creates a composable view above Kafka and Iceberg. When data is queried through Streambased, it resolves the best system and access pattern required to satisfy the query. In other words: data does not have to be first "sunk" before it is accessible.

Who is Streambased for?

Streambased is for data consumers and processors. Analysts, Data Scientists, AI/ML engineers and BI engineers may take advantage of Streambased composable views in Iceberg to enrich their models/reports/decision making tools. Conversely, Platform engineers and application developers may utilise the same views in Kafka to provide faster startup times, improved resilience and cost savings.

Can I use Streambased to power my applications?

Yes! Streambased surfaces Iceberg format for use in analytical applications, enriches this with indexes for point lookup style applications and surfaces Kafka protocol for operational applications.

I use Databrick/Snowflake, can I use Streambased?

Yes! Both of these systems support external Iceberg tables that allow you to use their powerful processing engines to work directly on Kafka/Iceberg data via Streambased. Users will simply see an expanded set of tables served by Streambased that they will be able to work with like any other.

2. Core concepts: hotset, coldset and merged set

What is the hotset?

The hotset represents the latest data available from Kafka. Streambased guarantees that, at query time, users are working with the very latest data available from Kafka.

What is the coldset?

The coldset represents high volume, historical data available from Iceberg. Streambased guarantees that, at query time, users are working with the maximum scope of data available from Iceberg.

What is the merged set?

The merged set combines hotset and coldset data (see above) to create a continuous view that stretches from the very latest data available in Kafka to the complete scope of historical data in Iceberg.

3. Working with your existing Kafka and Iceberg

What about my existing Kafka?

Streambased is an additional layer that consumes from Kafka. To take advantage of Streambased composable views you are not required to change anything about your Kafka deployment or data ingestion.

What about my existing Iceberg?

Streambased is an additional layer that unifies Kafka and Iceberg. You do not need to change any existing Iceberg data or engines in order to take advantage of Streambased composable views.

What happens if my schema changes?

Streambased guarantees that data layouts presented in Kafka and Iceberg are equivalent. To ensure this guarantee survives changes, Streambased provides schema evolution features to instantly reflect evolutions in the Kafka data layout in Iceberg. For instance, adding a new field in a Kafka topic will result in a new column in the corresponding Iceberg table with no additional work required.

How does Streambased solve Iceberg small write problems?

Iceberg is an analytical data format and not designed to ingest fast moving, frequently written data. When Iceberg encounters data of this type (such as data written to Kafka), ingesting it can create serious performance compromising issues such as:

- **Small files** – Iceberg uses file based data, frequent writes create lots of small files that bloat data sizes, degrade performance when opened and closed for queries and incur high costs in cloud provided file stores.
- **Snapshot expiration** – Every write in Iceberg creates a snapshot of the table's state at the time of writing. Frequent small writes causes the number of these snapshots to increase dramatically, greatly degrading metadata performance.

Streambased is able to address these issues by delaying writes into the physical filesystem until they are efficient (e.g. wait for 256MB of data to accumulate before creating a new file). To ensure data remains fresh, Streambased provides an ephemeral file system above Kafka to serve fresh data. The combination of physical data in Iceberg and ephemeral data in Kafka is then used to satisfy queries, ensuring users can benefit from the freshest data without having to compromise on efficient Iceberg storage.

Check out our Substack article here for more information.

4. Architecture and deployment

How does Streambased deploy in my environment (on-prem, cloud, VPC, hybrid)?

Streambased is deployed as containers (see: https://hub.docker.com/u/streambased). These are deployed into your own Kubernetes or Docker environment.

What are the core components (coordinator, workers, index service, catalog integration, etc.) and how do they scale?

Streambased consists of the following components:

- **Streambased I.S.K.** - The core components that translates Kafka to Iceberg and stitches together continuous Kafka + Iceberg views and serves them as Iceberg.
- **(optional) Streambased K.S.I.** - A component that provides Iceberg -> Kafka translation to serve Kafka + Iceberg views via Kafka protocol.
- **(optional) Slipstream** - A GUI based monitoring and management component for the above services.

All components are stateless and scale horizontally to match the workload requirements.

What are the infrastructure prerequisites (Kafka version, Iceberg catalog types supported, object stores, Kubernetes vs. Docker, network and security requirements)?

Kafka version: 2.5.0+

Catalogs supported:

for Coldset: Any

Presented by Streambased: generic REST

Object Stores: Any S3 compatible

Deployment methods: Kubernetes or Docker (bare metal is possible but must be tailored to specific cases)

Network and Security Requirements: None specific, all Kafka and Iceberg security options are supported.

How does Streambased connect to my existing Kafka clusters (multi-cluster, Confluent Cloud, MSK) and Iceberg catalogs (Hive, REST, Snowflake/ Databricks external Iceberg, etc.)?

Streambased utilises only the public APIs exposed by both Kafka and Iceberg. To all intents and purposes Streambased is just another client to these systems and subject to the same operating parameters and controls as any other application.

How can I get started?

Streambased is provided as a number of docker containers to be deployed using Kubernetes or docker-compose. A great starting point is Breakstream, the Streambased demo environment here: **https://prod.streambased.cloud/**

5. Performance, latency and limits

What query latencies should I expect for: hotset-only queries, merged hot+cold, and large historical scans? Any reference benchmarks?

Query latencies are entirely dependent on data size, layout and the operating parameters of the underlying Kafka and Iceberg, so defendable benchmarks are difficult to provide. What we can say is that query latency for Iceberg clients is dependent on hotset size (the larger the hotset the slower the query) and on the ratio of hotset to coldset. We can say (softly) that, when the hotset is around the size of a single coldset file (~128MB typically for Parquet) and the query addresses enough data so that all data does not fit in memory at the same time, hotset overhead is effectively 0.

How does Streambased achieve query acceleration over raw Kafka (indexing strategy, caching, predicate pushdown into segments, parallelism)?

Streambased splits Kafka data into chunks of offsets e.g.

Chunk 1: partition 0: 0-5000

Chunk 2: partition 0: 5001-10000

Chunk 3: partition 1: 0-5000

By treating Kafka data in this way Streambased can parallelise beyond Kafka's traditional limit (parallelism by partition). On top of this Streambased maintains indexes over these chunks allowing predicate pushdown where chunks are read only if they contain data relevant to the query.

These chunks are also immutable thanks to Kafka semantics. This means Streambased can cache chunk reads. Streambased will only fall back to Kafka to read a chunk if it is not already available from the cache.

Are there limits on topic size, partitions, or table sizes? How does it behave with long-retention Kafka and multi-PB Iceberg tables?

There are no concerns of this type. As an additional layer above Iceberg and Kafka, Streambased acts as any other client and is independent of any limits in the underlying systems. Streambased also scales independently to provide capacity above that of the underlying data providers.

6. Consistency, freshness and failure modes

What consistency guarantees do I get when querying the merged set (e.g. “read-your-writes”, snapshot consistency across Kafka and Iceberg at a given point in time)?

Streambased guarantees that, at the point of query execution, the data available to the query is the latest available in both Iceberg and Kafka. In addition it guarantees that any data that overlaps between Kafka and Iceberg will only be presented once. E.g. if Kafka contains offsets 10000-15000 and Iceberg 2000-12000, a Streambased query would only see offsets 10000-12000 once.

You can think of it as reading all relevant snapshots from Iceberg and then adding an additional snapshot that contains all data available from Kafka.

Both underlying systems are transactional and Streambased mirrors this providing read-your-writes consistency ACID where possible.

How fresh is “fresh”? Can I configure freshness/lag SLAs per view or workload?

Fresh means the freshest data available at the point of query execution. If I execute a query at 10:30:15 I have access to all of the data in Kafka and Iceberg at that time. For this reason the SLA is 0 and not configurable.

However, if the query takes 15 mins to complete Streambased will not continue to ingest data whilst it is running. E.g. if the query finishes at 10:45:15 it represents results as of 10:30:15.

What happens during Kafka or catalog outages? How does query behaviour degrade and recover (failover, retries, partial reads, fallback to coldset only)?

Both Kafka and Iceberg are very resilient systems. However, should there be an outage, the impact on Streambased will be that queries will fail. As soon as the underlying systems have recovered, query execution capabilities will be restored. All of this requires no manual intervention.

7. Integration with existing tools

Which engines and tools are officially supported (Trino/Starburst, Spark, Flink SQL, Snowflake, Databricks, BigQuery, BI tools via JDBC/ODBC)?

Streambased supports querying by any Iceberg compatible engine or Kafka compatible application. This includes Spark, Trino, Databricks, Snowflake, DuckDB etc. etc.

How do I query Streambased from my existing warehouse/lakehouse (e.g. via external Iceberg tables in Snowflake or Databricks) and what does the table naming/namespace look like?

As mentioned, Streambased integrates with existing warehouses as external Iceberg tables. By default it exposes 3 namespaces:

- The hotset - data from Kafka
- The coldset - data from Iceberg
- The mergedset - a combination of hotset and coldset

However individual namespace/table populations are tuneable by the consuming system.

Can I continue to run my existing ETL/ELT jobs and slowly migrate them to Streambased views, or is it an all-or-nothing cutover?

Streambased is an additional stateless layer on top of existing data systems. It requires no “big bang” cutover and all existing ETL jobs can run alongside a deployment.

8. Comparison to alternatives

How is Streambased different from Kafka Connect sink connectors, Confluent Tableflow, Starburst streaming ingest, or home-grown Spark/Flink jobs into Iceberg?

The difference is the freshness guarantee, and where and how complexity is handled.

Traditional ingestion-based approaches (Kafka Connect, Flink, Spark, Tableflow, home-grown pipelines):

Physically materialise Kafka data into files before it is queryable.

Freshness is bounded by pipeline health and lag (if a job is delayed by 30 minutes, queries are unavoidably 30 minutes stale.)

Introduce significant operational overhead, including:

- Small file generation and snapshot churn
- Compaction, expiration, and vacuum jobs
- Backfills, retries, and replay handling
- Connector failures, schema evolution issues, and on-call burden.

Streambased takes a fundamentally different approach:

Streambased does not ingest Kafka data into Iceberg by default.

Kafka data is exposed at query time through an Iceberg-compatible interface, ensuring queries always see the latest available data.

There is no ingestion lag to manage (freshness is determined by Kafka itself, not by downstream pipelines.)

Because data is not continuously materialised into files:

- Small-file problems are avoided for hot data
- Snapshot churn and continuous compaction are eliminated
- There are no always-on ingestion jobs to monitor or recover.

When materialisation into Iceberg is required, it is done as explicit, controlled batch jobs, not as fragile, continuously running pipelines.

What trade-offs do I make versus a "materialised" pattern (zero-copy vs. pre-materialised tables, cost profile, operational risk, failure modes)?

As above.

When should I still use a traditional sink/ETL into Iceberg instead of Streambased, and can they coexist cleanly?

They can co-exist but there isn't really a case where having ETL improves the situation.

The future data stack should not include ETL, instead it should have Stream processing + Unified views.

9. Data modelling, schema and operations

How are Streambased views defined and managed (DDL, config files, API, UI)? How do I version and promote them across environments?

Today this is all automatic, the hotset represents everything in Kafka, the coldset everything in Iceberg and the mergedset everything combined (by naming convention)

How does Streambased handle complex schema evolution scenarios (field deletions, type narrowing/ widening, nested structures, incompatible changes)?

Streambased support the intersection of schema evolution options provided by both Kafka's Schema Registry and Iceberg. This includes:

- Addition of fields
- Deletion of fields
- Expanding of numerical types
- Changing of type definition by union.

What operational tooling is available (metrics, dashboards, health checks, alerts, back-pressure indicators, index maintenance)?

Slipstream (Streambased monitoring and management tool) provides metrics, cluster/user maintenance, index management and alerting.

10. Cost, sizing and licensing

How is Streambased licensed and priced (by cluster, by data volume, by queries, by topics/tables)? Any differences for POC vs. production?

Streambased is licensed as an annual subscription, with pricing based primarily on the number of Kafka clusters your analytical engines need to access through Streambased. This keeps the model tightly aligned to how you operate Kafka in production, rather than how often people query it.

We do not charge based on:

- Data volume in Kafka
- Query counts or concurrency
- Number of consumers or users
- Number of topics or tables
- How your Kafka estate grows over time.

This is a deliberate design choice: Streambased is intended to be used freely for analytics, investigation, and decision-making, without teams worrying that increased adoption or new use cases will unexpectedly inflate costs. POC/POV and pilot commercials are discussed case-by-case based on use case and complexity; further details, including our free-of-charge option, are described elsewhere in this FAQ.

What is the expected infra cost profile vs. “Kafka + connectors + Spark/ Flink jobs + warehouse” (broker load impact, storage, compute)?

Streambased changes the cost profile by reducing the need for always-on ingestion pipelines and duplicate data copies, and by decoupling analytical freshness from continuous data movement.

In practice:

Kafka: No continuous sink consumers are required to make data queryable. Broker load is driven by actual analytical access, with caching to limit repeated reads.

Compute: Reduces reliance on always-running Kafka Connect, Flink, or Spark jobs whose primary role is data ingestion. Movement of data from Kafka to Iceberg, when required, runs as explicit, controlled batch jobs.

Storage: Avoids duplicating “hot” data solely for analytical access. Kafka and Iceberg retain data according to their natural roles.

Warehouses: Less ingestion related compute overhead; warehouse resources are used for analytics rather than freshness maintenance.

Streambased shifts cost from continuous background processing toward on demand access and controlled data movement, improving predictability and reducing operational overhead compared to pipeline heavy architectures.

How do I size a Streambased deployment for my Kafka/Iceberg footprint (topics, partitions, QPS, concurrency)?

There is no one-size-fits-all sizing formula. Streambased sizing depends on your actual workload characteristics, including:

- Kafka topic count, partitions, and retention
- Query patterns (filters, time ranges, joins)
- Concurrency and query frequency
- Cache hit ratios and freshness requirements
- Whether and how often data is materialised into Iceberg.

Because Streambased serves data at query time, sizing is driven by how the data is queried, not just by raw data volume.

For this reason, Streambased deployments are sized collaboratively. During evaluation or onboarding, we review your Kafka footprint and expected query patterns and provide a tailored sizing recommendation for compute, cache, and scaling parameters.

11. Reliability, SLAs and roadmap

What SLAs or SLOs can you commit to (availability, query success rate, freshness) in managed vs. self-hosted deployments?

Streambased's availability and freshness characteristics are inherently dependent on the underlying Kafka and Iceberg infrastructure it integrates with. As a result, Streambased provides the intersection of the guarantees offered by those systems.

Self-hosted deployments (recommended for production):

- Streambased is designed as a stateless, horizontally scalable service with fast failover.
- As long as at least one Streambased instance is running, the service remains available.
- Availability and durability ultimately follow the guarantees of Kafka brokers, object storage, and the Iceberg catalog in use.
- Streambased does not currently provide formal guarantees on query success rate or data freshness; these depend on query shape, Kafka availability, and infrastructure conditions.

Managed Streambased Service:

- The managed offering is currently provided as a demo / evaluation service only.
- No formal SLAs or SLOs are offered for managed deployments at this time.

For any production use cases, Streambased recommends self-hosting in customer controlled infrastructure (Kubernetes / Docker), where reliability, scaling, and operational controls are fully under the customer's control.

What are the main limitations and known "non-goals" today (e.g. multi-region consistency, exactly-once semantics across estates, write-back into Kafka/ Iceberg)?

Streambased is read-only by design.

Streambased does not write data back into Kafka topics or Iceberg tables as part of its core query path.

It does not act as an ingestion, transformation, or ETL system, and does not replace Kafka Connect, Flink, or Spark for data production workflows.

There is no concept of exactly-once semantics across Kafka and Iceberg, as Streambased does not own or control writes to either system.

Streambased's focus is deliberately narrow: to provide a unified, queryable view across hot (Kafka) and cold (Iceberg) data, using standard Iceberg interfaces, without introducing new data copies or ingestion pipelines.

Any data movement (e.g. materialising Kafka data into Iceberg) is handled explicitly via controlled batch jobs, not as part of Streambased's core query semantics.

What is on the near-term roadmap (more catalogs, other table formats, more clouds, additional indexing strategies)?

- ✓ More catalogs
- ✓ More filesystems
- ✓ Better CDC
- ✓ More indexing

12. Evaluation, POC and support

What does a typical proof of value look like (time to deploy, datasets used, success criteria, roles required on the customer side)?

A typical Streambased proof of value focuses on a single, clearly defined use case running on one Kafka/Iceberg cluster, with a maximum duration of two weeks.

Upfront, we agree success criteria together with the client (for example: freshness, query performance, operational simplicity, or cost-reduction goals) and design a small but representative scope to validate them.

During the engagement we deploy Streambased into the target environment, define the composable views required for the use case, and run through agreed test scenarios with your team.

At the end of the two-week period we present measurable outcomes against the success criteria, plus recommended next steps and a productionisation path.

Longer, multi-use-case or multi-cluster engagements are available as paid pilots, tailored to your requirements.

What support, training, and onboarding do you provide (solution architects, best-practice patterns, migration playbooks)?

Streambased provides structured onboarding and training designed to get both platform teams and data consumers productive quickly. We offer solution architecture consultancy and advisory services on request, helping you design views, integrate with existing engines, and align Streambased with your internal platform standards.

Our formal onboarding program typically includes environment readiness checks, guided deployment, best-practice view design, and hands-on training sessions for engineers, analysts, and data scientists. Once live, customers are supported by a dedicated customer success manager who monitors progress, shares best practices, and coordinates any additional enablement you require.

Are there public case studies or reference architectures I can show internally for Kafka+Iceberg with Streambased?

Streambased maintains reference architectures and example deployment patterns that illustrate how to unify Kafka and Iceberg into a single, composable real-time view for typical enterprise use cases. These include architecture diagrams, sample view definitions, and integration patterns with common engines and tools, which you can reuse in internal design documents and stakeholder presentations.

Formal customer case studies are made available subject to client consent and publication timelines; your Streambased representative can share the latest publicly shareable materials and, where appropriate, arrange deeper technical sessions to walk stakeholders through relevant architectures.

For details of our support and service commitments, please refer to our standard terms and conditions.