WHITEPAPER · EVENT HORIZON DATA

From raw events to decisions, without a data team

Closing the gap between the event data organizations already collect and the timely, trustworthy insight they struggle to get out of it.

Audience: technical & non-technical **Reading time**: ~12 min

Subject: self-service analytics

~9,000 ev/s
accepted · 4 workers, one host

~30 ms
time-filtered dashboard query, 20 M rows

22×–370×
columnar compression on dimensions

0 lost
events dropped, verified two ways

01: EXECUTIVE SUMMARY

Everyone collects event data. Few can turn it into answers.

Modern organizations instrument everything. Every product click, pipeline run, transaction, and sensor reading is an event, and storage is cheap enough that most teams keep all of it "for later." Yet the distance between *collecting* that data and *understanding* it has never been wider.

The industry's answer is the **modern data stack**: assemble an ingestion tool, a cloud warehouse, a transformation layer, and a business-intelligence front end, then hire the data engineers and analysts to operate it. That path works, but it is out of reach for exactly the teams that need answers most: agencies and consultancies serving many clients, operations teams, and lean product companies with no dedicated data function. They inherit the tools' complexity without the team to absorb it.

Event Horizon Data collapses that stack into a single self-service platform. Data flows in through drop-in SDKs, a language-agnostic collection agent, change-data-capture, and webhooks. It lands in a **columnar warehouse** engineered for low per-tenant cost and sub-second reads. And it becomes drag-and-drop dashboards, no-SQL exploration, and **data mining computed in the warehouse itself** (anomaly detection, forecasting,

driver analysis, cohorts) that a non-analyst can drive. It is multi-tenant and white-label by design, so an agency can put *its own* brand in front of its clients.

This paper frames the problem, weighs the realistic solutions, and explains why an integrated, self-service, columnar approach is the right answer for organizations that need decisions rather than a data department.

02: INTRODUCTION & BACKGROUND

The instrumentation reflex, and the comprehension debt it creates

Two decades of cheap storage and easy telemetry produced a habit: instrument first, ask questions later. It is a reasonable habit (you cannot analyze what you did not record) but it quietly accumulates a debt. Recording an event is nearly free; making that event *trustworthy, joined, and legible* months later is not.

The tooling that grew up to service this debt (managed connectors, cloud warehouses, transformation frameworks, and BI dashboards) is genuinely excellent. It is also built around an assumption that is invisible until you lack it: **a team.** Someone models the tables. Someone maintains the pipelines. Someone writes the query behind the chart. The tools are "self-service" in the sense that a data professional can serve themselves; they are not self-service for the marketer, the account manager, or the founder who actually has the question.

So the organizations with the sharpest need and the least infrastructure end up in one of two places. Either they buy the powerful stack and watch it sit half-configured because nobody owns it, or they never start, and decisions keep getting made on gut feel while the data they paid to collect goes cold in a bucket. The result is the same: **dashboards that lag reality, numbers nobody quite trusts, and analysis that waits in a queue, or never happens.**

03: DESCRIPTION OF THE PROBLEM

Where "we have the data" stops being true

"Turn raw events into timely, trustworthy insight" sounds like one problem. It is really five, each with its own failure mode, and a team without a data function tends to hit all of them.

1. Ingestion is plumbing, and plumbing leaks

Getting events reliably from many sources into a warehouse (typed, time-normalized, and each event counted *exactly once*) is a distributed-systems problem that teams routinely underestimate. Retries duplicate rows. A clock in the wrong timezone shifts a day's worth of data. A dropped connection loses the tail of a batch nobody notices until a total is wrong. Change-data-capture, webhooks, file drops, and agents each fail differently.

2. The warehouse is both a cost and a commitment

Row-oriented databases buckle under analytical scans; cloud warehouses meter every one of those scans, so the bill scales with curiosity. And getting the *economics per tenant* right (cheap to store, cheap to isolate, cheap to query) is a design problem most teams solve badly on the first try, then live with.

3. The last mile still needs SQL, and therefore an analyst

Most BI tools assume a person writes the query, models the join, and knows which aggregation is correct. That person is the bottleneck. When "self-service analytics" means "self-service for people who already know SQL," the marketer with the actual question is still filing a ticket.

4. Insight is reactive, and by then the window has closed

A human notices an anomaly after it has done its damage, builds a forecast in a spreadsheet that is stale by Monday, and rarely gets to ask *why* a number moved. Anomaly detection, forecasting, and driver analysis usually live in a separate machine-learning effort that most teams never staff.

5. Trust erodes at the seams

Every hand-off, ingest to transform to model to dashboard, is a place for a number to drift. When the stack is five vendors, nobody can answer the most corrosive question in analytics: *why do these two dashboards disagree?* Trust, once lost, is expensive to rebuild.

> For an agency, multiply every one of these by every client, then add white-label branding and hard tenant isolation on top.

04: POSSIBLE SOLUTIONS

Four honest ways to close the gap

There is no shortage of tools. The question is which shape of solution fits an organization *without* a data team. Four approaches are worth weighing plainly.

a. Assemble the modern data stack yourself

Best-in-class connectors, a cloud warehouse, a transformation framework, and a BI tool. Unbounded flexibility and the industry's deepest ecosystem, but it needs a data engineer and an analyst, several recurring vendor bills, and months to stand up and keep healthy.

RIGHT WHEN YOU ALREADY HAVE A DATA TEAM. WRONG WHEN YOU DON'T.

b. Product-analytics suites

Fast for product funnels, retention, and event taxonomies. But they are narrow by construction: bound to an event model, not a general warehouse, thin on custom modeling, and metered so that success, more volume, is what raises the price.

GREAT FOR ONE NARROW JOB; NOT A HOME FOR ALL YOUR DATA.

c. Embedded BI on your own warehouse

Solves the dashboard layer cleanly and embeds well. But it still assumes you bring the warehouse, the pipelines that fill it, and the person who models the data underneath: the hard three-quarters of the problem.

SOLVES THE LAST MILE; LEAVES THE FIRST THREE.

d. An integrated, self-service analytics platform

Ingestion, warehouse, exploration, and data mining as one product: drivable without SQL, multi-tenant, one bill, one coherent system. Fewer degrees of freedom than assembling your own stack, and that is the point: the trade is à-la-carte flexibility for a system a non-specialist can actually run, and whose numbers are defensible end to end because nothing crosses a vendor seam.

THE FIT FOR A TEAM THAT NEEDS DECISIONS, NOT A DATA DEPARTMENT.

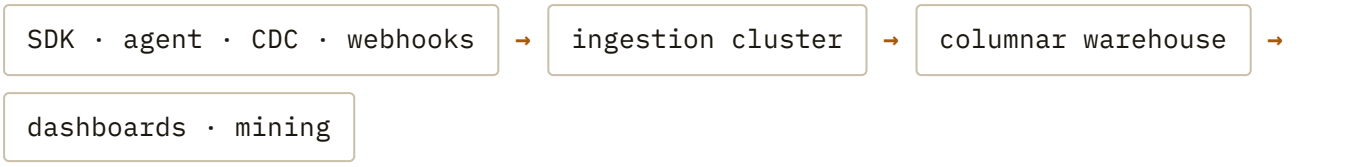
The trade-off is real and worth stating: integration versus flexibility. If you have data engineers, keep your degrees of freedom: approach (a) is built for you. If you do not, every degree of freedom is a degree of maintenance you cannot afford, and integration is not a compromise but the whole value.

What the integrated approach looks like, built honestly

For organizations without a data function, and especially for agencies reselling insight to their own clients, we recommend approach (d): an integrated, self-service platform on a columnar warehouse, with data mining computed in the warehouse rather than bolted on. Event Horizon Data is one embodiment of that recommendation; the architecture below is described as a template for what to look for, not a feature list.

Self-service ingestion the customer never operates

Data enters through whatever the source already speaks: a drop-in JavaScript/TypeScript SDK (the same code in a browser or on a server); a language-agnostic **collection agent** that tails files, watches directories, and receives syslog over TLS with mutual authentication, delivering *exactly once across crashes* and shipping as a single self-contained binary; change-data-capture from Postgres, MySQL, and MongoDB; webhooks; scheduled pulls; and file transfer. Behind them, a distributed ingestion cluster, coordinated for leader-election and fed through a durable log, handles ordering, retries, and deduplication so the customer never has to.



A columnar warehouse with per-tenant economics

Every event lands in a **single shared columnar table**, ordered so that each tenant's rows sit together and a query skips every other tenant's data before it reads a byte. Each event carries its attributes as *typed, semi-structured columns on the row itself* (an immutable point-in-time fact, not a surrogate key pointing at a mutable lookup) so relabeling a category tomorrow can never silently rewrite last year's numbers. Low-cardinality attributes dictionary-compress by factors of tens to hundreds, holding the marginal storage cost per tenant to a few tens of bytes a row. Partitioning is by month and retention is a policy, not a cron job. Queries serve *directly from that table* with time-bucketing applied at read time, so there is no fragile hierarchy of pre-aggregates to drift out of sync; a hot field is promoted to a fully-typed column when it earns it, and onboarding a new tenant with its own schema is a *row, not a migration*. Because one table holds every tenant, the model scales to **thousands of tenants** without the file-and-metadata explosion that sinks a table-per-tenant design, and it is cluster-ready, sharded by a hash of the tenant, so it also scales out horizontally.

The last mile: no SQL required, SQL on rails when you want it

A drag-and-drop dashboard builder and a no-SQL explorer (filters, aggregations, drill-through) let a non-analyst ask and answer questions directly. On top sits **data mining that runs in the warehouse**, not in a separate pipeline: anomaly detection that flags a number the moment it leaves its expected band, forecasting that draws the trend forward, driver analysis that explains *which dimension moved the metric*, and cohort/retention over first-touch behavior. Add target and threshold lines, alerts, scheduled PDF reports, and shareable embeds, and the platform becomes a planning tool, not just a rear-view mirror.

For the minority who do write SQL, that no-SQL default is a floor, not a ceiling. A **governed SQL Explorer** and an **HTTP connector for notebooks** (Python, R, Jupyter) let a power user query their own data directly,

both riding the same rails. You write ordinary SQL against your *buckets as logical tables*, blind to how the warehouse physically stores those rows, so the storage model stays ours to make faster without breaking a saved query. Your text never executes as written: it is parsed, checked against a default-deny allowlist, and reconstructed server-side into a query locked to your tenant, so a SQL surface adds no new path across the isolation boundary. Every query is **cost-checked before it runs**, the way BigQuery's dry-run estimates a query, and one that would scan past your plan's budget is refused up front with the estimate shown, so no single query, and no noisy neighbour, can run a shared warehouse into the ground. The notebook connector authenticates with a **revocable, tenant-scoped token**, and a saved SQL query drops onto a dashboard as an ordinary widget. SQL is an addition here, never a prerequisite.

Multi-tenant and white-label by construction

Tenant isolation is enforced at a single fail-closed checkpoint through which every warehouse read passes: a query that cannot name the tenant it is working for does not run. On the shared table, that checkpoint injects the tenant predicate itself (isolation is a property of every query, not of a table name a caller might forget) and warehouse access uses least-privilege identities. On top of that boundary, an agency's brand (name, logo, colors) carries all the way through dashboards, embeds, transactional emails, and PDF exports, so the agency's clients see the agency, not the platform.

Evidence, measured and scoped

Architecture is a claim; measurement is evidence. On a single modest host, a small cluster of ingestion workers sustains roughly **4,400 events per second with no backlog** at a 44-millisecond p99, and delivers about **7,400 per second with zero data loss**: integrity verified two independent ways, by reconciling the data table against the audit log row for row and by a shutdown test proving no buffered event is lost. Push past the sustained rate and the excess is *draining backlog, not lost data*; with four workers on the host, acceptance climbs to close to **9,000 events per second**. The bottleneck throughout is ingestion compute, not the warehouse. Separately, on a single modest **warehouse** node (about four cores and six gigabytes of memory), over a **20-million-row** event table, time-filtered dashboard queries return in about **30 milliseconds**, distinct counts in 186, and full-history scans in around a second, with no memory exhaustion up to sixteen heavy queries at once.

> HONEST SCOPE
>
> These numbers are measured on modest hardware: a small ingestion cluster of three to four workers on one host, and a single warehouse node. The larger sharded, multi-host cluster the design targets is not yet load-validated at production scale, and published latencies are best-case on shared-core hardware. They are a floor to reproduce, not a promise to quote. Stating the boundary is part of earning the trust the rest of the paper is about.

A checklist for evaluating any such platform

- **Exactly-once ingestion** that survives crashes, not "at least once" with duplicates you clean up later.

- **Columnar economics**: cheap to store and scan per tenant, with compression you can measure.
- **A no-SQL last mile** a non-analyst can actually drive, with an **optional SQL surface on the same rails** for those who want it.
- **A cost gate on every query**: an estimate-and-refuse budget, so one careless query cannot run up the bill.
- **In-warehouse data mining** (anomaly, forecast, drivers): not a separate ML project.
- **Provable tenant isolation** at a single enforced boundary, plus real white-label depth.
- **Honest measurements** with their scope stated. A vendor that hides the boundary is hiding something.

06: REFERENCE

Notes & further reading

1 ClickHouse: open-source column-oriented OLAP database. Documentation and architecture notes, `clickhouse.com/docs`.

2 S. Steinarsson, *Downsampling Time Series for Visual Representation* (Largest-Triangle-Three-Buckets), M.Sc. thesis, University of Iceland, 2013: the shape-preserving downsampling behind readable long-range charts.

3 Debezium: open-source change-data-capture for Postgres, MySQL, and MongoDB, `debezium.io`.

4 Apache Kafka: the durable log underpinning ordered, replayable ingestion, `kafka.apache.org`.

5 On the “modern data stack” and its implicit assumption of a dedicated data team: general industry discourse; this paper’s argument is that the assumption, not the tools, is what excludes lean teams.

6 Columnar storage and compression: the economics of reading only the columns a query needs, and why low-cardinality dimensions compress by orders of magnitude.

Event Horizon Data, self-service analytics: ingest, warehouse, explore, and mine event data without a data team.
This document describes architecture and measured results as of its writing; measurements state their own scope. Figures are single-node unless noted.