

HanuAnalytics: Product Analytics and Session Replay on a Free-Tier Virtual Machine with a Bucket as Its Home

Madhu

Kansas City · Independent

ABSTRACT

We describe HanuAnalytics, an open-source product analytics and session replay system designed to run on the smallest virtual machine a public cloud offers at no charge, with a cloud object bucket as the durable home of all data. A single Rust binary serves the browser SDK, the ingest API, the analytics API and the admin application. Events are buffered for up to sixty seconds and written as zstd-compressed Parquet files laid out by project and day, compacted daily to one file per project per day; queries are evaluated by Apache DataFusion over a local mirror of those files and never read from the bucket. Session recordings captured with rrweb are masked in the visitor's browser, shipped as gzip chunks and grouped into objects of about 512 KB. Metadata, person profiles, sessions and the live view are kept in HanuDB, a bucket-resident document database that runs on the same machine. We give the cost model at 2026 list prices: the machine and its disk are free, storage is about \$0.02 per GB-month, a million events occupy about 60 MB, and bucket writes at \$5 per million are the only item billed by the piece, which the buffering, compaction and chunk grouping are designed to minimise; a product with a few hundred thousand pageviews a month and recording on is estimated at under one dollar a month. We set out the privacy rules (no raw IP address is stored, a daily salted visitor hash, an optional cookieless mode, masking in the browser, bot exclusion, per-project retention) and the limits: one machine is the ceiling, and no benchmark record exists yet. The cost figures in this report are arithmetic at list prices, not measurements.

Keywords: web analytics, session replay, object storage, Parquet, DataFusion, privacy, free tier

1. Introduction

A product team needs a small number of answers about its visitors: how many came, from where, which pages they read, where a funnel loses them, which errors they hit, and, for one puzzling case, what a particular visitor actually did on the screen. Two kinds of system answer these questions today. The first is free and is paid for with the data: every visit is sent to an advertising company, and the visitor is asked to consent through a banner. The second is hosted

and billed per event, so the bill rises with the product's success and does not fall in a quiet month; session replay is typically the most expensive item.

HanuAnalytics is a third kind. It applies the design of HanuDB [1], in which the machine that serves the data holds no truth and a cloud bucket does, to the analytics problem. The result is one binary that runs on a free-tier virtual machine, keeps every event as files in a bucket the operator owns, and serves dashboards, funnels, retention, errors, Web Vitals, a live view, person profiles, session replays and a read-only SQL explorer from that machine. The system is open source under the MIT licence and a hosted instance is in public beta.

This report describes the design (section 3), the storage layout and its consequences for cost (sections 3.3 and 4), the privacy rules (section 5) and the limits (section 6). It is a technical report rather than an experimental paper: unlike HanuDB, which was shaped by more than a hundred controlled experiments against a fixed benchmark, HanuAnalytics was built on top of that engine and has a correctness and smoke suite but no benchmark record yet. Every cost figure below is a list price multiplied out, and is labelled as such.

2. Requirements

The requirements were fixed before the code was written.

1. **One free machine.** The whole system, including its database, must run within the always-free e2-micro: two shared vCPUs sustaining a quarter of one core, 1 GB of memory, a 30 GB standard disk.
2. **The bucket is the truth.** The machine may be deleted at any time. A new machine must rebuild from the bucket alone, with no acknowledged event lost.
3. **Cost that tracks gigabytes, not events.** The design must keep the one per-piece charge, bucket writes, in the hundreds per day for a busy site.
4. **Defensible privacy.** No raw IP address stored; identification only when the operator's own site asks for it; masking before data leaves the browser; bots excluded; retention per project.
5. **A complete product on day one.** The dashboards a team expects from a hosted tool, session replay included, with many organizations and projects on one instance.

3. Design

Figure 1 shows the parts. A visitor's browser loads the SDK from the machine and posts batches of events to it. The machine runs Caddy for HTTPS, the `hanu` binary, and HanuDB, under systemd with memory limits suited to 1 GB. The bucket holds three prefixes: Parquet event files, recording chunks, and HanuDB's own log and snapshots.

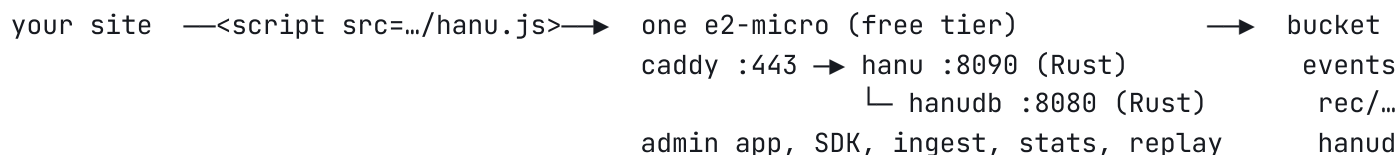


Figure 1. One machine, one bucket. The binary serves the SDK, ingest, the analytics API and the admin; HanuDB holds the small and changing rows; the bucket holds the truth.

3.1 The SDK

The browser SDK is written in TypeScript and built to a single script of 21 KB, 8 KB compressed, served by the binary with a one-hour cache and a day of stale-while-revalidate. It captures pageviews, including single-page-application navigations, clicks, outbound links, downloads, form submits, rage clicks, JavaScript errors and Web Vitals without configuration, and exposes `capture`, `identify`, `reset` and `optOut`. Events are batched and sent to one ingest endpoint with `sendBeacon` on page hide. The recording bundle, which contains `rrweb` [4], is a separate 74 KB file (23 KB compressed) loaded only when the project has recording on and the session is sampled. A native iOS client of one Swift file with no dependencies reports screens as pageviews under an `app://<bundle id>/<path>` address, so the hostname separates an app's traffic from its website's.

Bot detection runs in the browser: known crawlers by user agent, headless browsers and `navigator.webdriver` send nothing. The server applies the same test to the user agent it receives, so a bot that bypasses the script is still excluded from statistics.

3.2 Ingest

A batch carries the project's public key. The server accepts it only when the request's Origin or Referer matches one of the project's listed domains, or, for a request with no Origin, when the `X-Hanu-App` header names a bundle id on the project's list of native apps, or when the project has explicitly allowed origin-less requests. Otherwise the batch is refused with 403. The public key is public; the allow-lists keep honest clients in their lane and stop a leaked key from being used to pollute a project's data from elsewhere.

For each accepted event the server reads the client IP once, in memory, to look up the country, region and city in the DB-IP City Lite database [5] and, in cookieless mode, to compute the visitor hash (section 5). The IP is then dropped. The event is enriched with the parsed user agent (browser, version, operating system, device class, or "iOS app" for the native client), the session id, and the visitor id, and placed in a per-project buffer.

The buffer is flushed when it holds enough events or after sixty seconds, whichever comes first, as one Parquet row group written to the bucket and mirrored to the local disk. This single deci-

sion governs the cost model: a bucket charges by the write, and a site with traffic around the clock makes about 1,500 writes a day regardless of how many events arrive.

3.3 Storage layout

Events are Parquet files [\[3\]](#) compressed with zstd, laid out as `events/<project>/<day>/<file>.parquet`. A daily job compacts a day's files into one, so a year of a project is a few hundred objects and a day's scan touches one file. A million events occupy about 60 MB. The schema is one wide table: timestamp, project, event name, visitor and session ids, the URL and its parts, referrer and UTM fields, geography at city level, device fields, and a JSON column of custom properties.

Recordings are stored as `rec/<project>/<session>/<seq>.json.gz`: rrweb events, already gzip-compressed in the browser, grouped by the server into objects of about 512 KB. A ten-minute recording is roughly 300 KB. The bucket carries a lifecycle rule that deletes any recording object older than 90 days, independent of the project's own retention setting, as a backstop against a misconfiguration.

HanuDB's prefix holds its hash-chained log segments and snapshots, as described in [\[1\]](#).

3.4 Queries

Every panel in the admin is a SQL query evaluated by Apache DataFusion [\[2\]](#), embedded in the binary, over the Parquet files on the machine's disk. The disk is a mirror of the bucket's event prefix for the projects the machine serves: files written by this machine are already present, and a fresh machine fills its mirror on demand, pulling a day's file the first time a query needs it. Queries therefore never read from the bucket and never incur a bucket operation. Query memory is bounded by a setting, 192 MB by default and 160 MB on the deployed free machine, raised on a larger one; a query is cut off after sixty seconds.

The same engine serves the admin's read-only SQL explorer, which lets an operator ask any question the panels do not, restricted to that project's `events` table, with a row limit and the same timeout. Counts in the API are integers; nullable aggregates such as a 75th percentile over no rows are `null`, and the admin renders a dash for them.

3.5 Session replay

Recording is a per-project setting with a sample rate, a minimum duration below which a recording is discarded, and three masking controls: mask all inputs (on by default), a selector for text to mask, and a selector for elements to block entirely. Password fields are always masked. E-mail addresses, card numbers and similar patterns in captured click text are masked by the SDK. Because masking happens in the browser, masked text never leaves it and there is

nothing to protect on the server. The admin replays a recording with rrweb-player beside a timeline of the session's events, and a daily job deletes recordings whose retention has expired.

3.6 Visitors, sessions and people

By default the SDK keeps a random anonymous id in `localStorage`, with a cookie fallback on old browsers. A session is the anonymous id plus thirty minutes of inactivity. When the operator's site calls `identify`, the anonymous id is linked to the given user id and a person profile is created or updated; nothing is identified otherwise. In cookieless mode (section 5) the SDK stores nothing and the server supplies the visitor id from the daily hash.

Organizations, projects, users, invites, access requests, person profiles, the session index and the live view are rows in HanuDB: they change often and are read by key, which suits a document store and not daily files. HanuDB runs on the same machine and keeps its truth in the same bucket.

3.7 Accounts and tenancy

One instance serves any number of organizations, each with its own projects, members and roles. The first account created at `/setup` becomes the instance admin. Members join by invitation link, or anyone may register when the instance admin opens sign-up. Sign-in is e-mail and password, or Google through OpenID Connect when a client is configured. A visitor can request a hosted account from the home page, which ships inside the binary; the instance admin approves it from a Requests page, which creates the organization and an invitation link.

3.8 Deployment

One script creates the bucket in the machine's region with its lifecycle rule, a service account that can write that bucket and read two secrets and nothing else, the secrets (HanuDB's token and the visitor-hash salt), a static address, firewall rules (80 and 443 public, SSH only through Identity-Aware Proxy) and one e2-micro. The machine's startup script installs Caddy for automatic HTTPS, fetches HanuDB from its public release bucket and `hanu` from the operator's bucket, both verified by checksum, and runs them under systemd with memory limits. The same script with a flag uploads a new binary and restarts, for a few seconds of downtime. A Cloudflare Tunnel option runs the machine with no public address at all.

4. Cost model

Table 1 gives the items at 2026 list prices for us-central1. The figures are arithmetic, not a measured bill.

Table 1. The bill, line by line, at 2026 list prices for us-central1.

Item	Price	Note
e2-micro with 30 GB standard disk	free	one per billing account in us-west1, us-central1 or us-east1
Static external address	free	while attached to the free-tier machine; otherwise about \$3.60 a month, or \$0 with the tunnel
Bucket storage	about \$0.02 per GB-month	a million events is about 60 MB; a ten-minute recording about 300 KB
Bucket writes	\$5 per million	the one item billed by the piece; about 1,500 a day for a site with traffic around the clock, about \$0.23 a month
Egress	1 GB free	the 8 KB script served 100,000 times is about 0.8 GB; \$0 behind Cloudflare's free plan
HanuDB	same machine and bucket	its own writes are a few cents a month

Three design decisions determine the writes line. Events are buffered for up to sixty seconds and written once per flush rather than once per event. A day's files are compacted into one, which costs one write per project per day and makes every later scan cheaper. Recording chunks are grouped into objects of about 512 KB, so a ten-minute recording is a handful of writes rather than hundreds. With these in place, a product with a few hundred thousand pageviews a month and recording on is estimated at under one dollar a month, most of it writes.

The shape of the bill matters more than its size. A hosted tool's price is a function of events, so doubling traffic doubles the line. Here the machine is a fixed zero, writes are a function of time rather than of events, and the only line that tracks traffic is storage at \$0.02 per GB-month, which a year of a small product's events barely moves. Retention, 365 days for events and 30 for recordings by default, bounds the storage.

The cost model of HanuDB on the same kind of machine was measured at about one cent a day after a day of heavy benchmarking [\[1\]](#). HanuAnalytics writes to the bucket less often than that workload did. A measured month will be reported when there is one.

5. Privacy

- **No raw IP address is stored.** The IP is used once, in memory, for the geographic lookup and the visitor hash, and dropped.
- **Visitor identity.** By default a random anonymous id in the browser. In cookieless mode the SDK stores nothing between page loads and the server counts visitors with a daily hash of (salt, day, project, IP, user agent). The salt is a secret of the instance and the hash

changes every day, the approach of Plausible and Fathom, with no consent banner needed in most readings of the ePrivacy rules.

- **Identification is the operator's call.** Nothing is identified unless the operator's own site calls `identify`.
- **Masking in the browser.** Every input is masked by default; per-project selectors mask text and block elements; passwords are always masked; e-mail and card patterns in click text are masked. Masked text never leaves the visitor's machine.
- **Do Not Track and opt-out.** DNT can be honoured per project; `opt0ut` stores an opt-out in the browser that a consent banner can set.
- **Bots** send nothing and are excluded from statistics.
- **Retention** is per project, with a daily job that deletes expired events and recordings, and a bucket lifecycle rule that removes any recording object older than 90 days regardless.
- **Allow-lists.** A project's key works only from its listed domains and native apps.
- **Residency.** The data lives in the operator's bucket, project and region. The server's only outbound call is a monthly download of the GeoIP database, and none if the operator uploads it.

6. Limitations

One machine is the ceiling. Queries run in the machine's memory, which on the free machine is a gigabyte shared with HanuDB. Tens of millions of events a month want an e2-small or e2-medium with the query-memory setting raised; the design otherwise does not change, because the truth was never on the machine. There is no clustering of the server, no second region, and no automatic failover, since a second machine is not free. The door is open: `hanu` instances are stateless apart from their buffers, the bucket is the truth, and HanuDB has followers and a leader lease.

No benchmark record. HanuDB was shaped by controlled experiments with hard gates and a written acceptance rule. HanuAnalytics was built on that engine in a day and has a correctness and smoke suite covering every endpoint, but no benchmark and no measured bill. The cost figures in this report are list prices multiplied out. Ingest throughput and query latency on the free machine have not been measured and are not claimed.

Buffered events are at risk for up to a minute. An event acknowledged to the browser sits in the machine's buffer until the next flush. A machine lost in that minute loses that buffer. This is the trade made for the writes line, and it is the one place where the machine briefly holds truth; the sixty-second limit bounds it.

Session replay is web only. The native client reports screens and events and records nothing.

7. Availability

The code is public under the MIT licence at github.com/madhudream/hanuanalytics: the Rust server (about 5,700 lines), the TypeScript SDK (about 1,600 lines), the React admin (about 4,600 lines, eighteen pages), the Swift client, the deployment scripts and six documents, including the API contract every part is built against and the privacy document this section summarises. A hosted instance runs at analytics.carebun.com, free while in beta. The GeoIP database is DB-IP City Lite, CC BY 4.0, with its attribution shown in the admin.

References

1. Madhu. *HanuDB: A Bucket-Resident Document Database for a Free-Tier Virtual Machine, and the 108 Experiments That Shaped It*. Preprint, September 2026. github.com/madhudream/hanudb.
2. Apache Arrow DataFusion. *An extensible query engine written in Rust that uses Apache Arrow as its in-memory format*. datafusion.apache.org.
3. Apache Parquet. *Columnar storage format*. parquet.apache.org.
4. rrweb. *Record and replay the web*. rrweb.io.
5. DB-IP. *IP to City Lite database*, CC BY 4.0. db-ip.com.
6. Google Cloud. *Free Tier: Compute Engine, Cloud Storage*. cloud.google.com/free, 2026 list prices.