ALLIED BizTech SOLUTIONS

FIELD GUIDE · 2026

Rebuild Sentry yourself.
The plan we follow.

For engineering teams on 15M to 1B+ events a month who want to own their error tracking, performance traces and session replay, and stop paying per event.

- → Is it worth it, at your volume
- → What to keep and what to drop
- → The architecture, in three layers
- → Migration and go-live checklists
- → The data model
- → A week-by-week build plan
- → Symbolication, grouping, alerts
- → Where teams get stuck

sentry.io charges per event. *ClickHouse charges per byte.*

sentry.io bills for errors, transactions, replays and spans, and each grows with your traffic. The server behind it is free to self-host. On your AWS, the bill follows storage and compute instead.

VOLUME	SENTRY.IO A YEAR	SELF-HOST: YEAR ONE / AFTER	SAVED IN YEAR ONE
5M events/mo · Team tier	~$1,068	~$12,800 / $1,200	Stay on sentry.io
15M events/mo · Business	~$15,000	~$15,000 / $3,400	Marginal; $11K a year after
50M events/mo · Business + Replay	~$60,000	~$21,200 / $9,600	~$38,800
200M events/mo · Enterprise	~$180,000	~$60,600 / $34,000	~$119,400
1B+ events/mo · Enterprise Plus	~$540,000+	~$124,600 / $98,000	~$415,400

sentry.io list prices, May 2026. "Self-host" runs Sentry on AWS EC2 (multi-AZ), ClickHouse for events and Postgres for metadata, managed for $3,600 a year, plus our fixed build price: $11,600 up to 100M events a month, $26,600 (the top of the $18,300–$26,600 scale tier) above that. If your own team builds it, put your engineering cost in that place.

THE RULE OF THUMB

Below about 15M events a month, stay on sentry.io. At 15M, year one is marginal and $11K a year comes back after. **At 50M events a month, self-hosting saves ~$38K in year one and ~$50K a year after.** From 200M, year one alone saves more than $110K.

Who you need

- **A senior infrastructure engineer** at home with AWS, Terraform, ClickHouse and Postgres. Sentry's docker-compose runs in 30 minutes; production-grade ops takes ~3 weeks of that person's time.
- **Your engineering lead**, two hours a week: the projects and alerts that matter, the parallel-run sign-off.
- **An owner after launch**: upgrades, ClickHouse capacity, backups, and on-call for Sentry itself.

Keep the Sentry you use. *Drop the side products.*

Self-hosted Sentry is the same code base as sentry.io: same UI, same SDKs, same workflow. The differences are operational. The scope is which features you keep, and which belong to a better specialist tool.

KEEP ONE FOR ONE

- **Error tracking and grouping**: stack traces, breadcrumbs, user context, releases
- **Performance monitoring**: transactions, spans, traces, web vitals, p50/p95/p99 latency
- **Session Replay**: DOM playback, network, console, click trail, payloads on S3
- **Releases and suspect commits**: git integration, deploy tracking, regression rates
- **Issue workflow**: ownership rules, assignment, status, snoozing, comments
- **Source maps and symbolication**: JS, Python and mobile, automated in CI/CD
- **SDKs**: 100+ official and community SDKs, unchanged
- **SSO, SCIM and audit log**, with role permissions

KEEP, RUN DIFFERENTLY

- **Uptime**: 99.9% on a properly deployed, monitored self-host, against 99.95% hosted
- **New features**: reach self-host ~2–4 weeks after cloud
- **Upgrades**: Sentry releases monthly; upgrade quarterly, staging first

DROP ON PURPOSE

- **Cron monitoring**: Healthchecks.io
- **Codecov integration**: Codecov direct, or Coveralls
- **Metric alerts**: Grafana and Alertmanager
- **Profiling**: py-spy and pprof, or keep that slice on sentry.io
- **Reselling Sentry as a service**: the FSL rules it out; internal use is unrestricted

ADD WHAT PER-EVENT PRICING DISCOURAGES

- No per-event fee; storage scales with bytes, not error count
- Retention per project: noisy ones 7 days, critical ones 365+
- Air-gap or on-prem, for banking, defence and regulated teams

Three layers. *The same Sentry, your tenancy.*

The stack the team at sentry.io runs: Relay for ingest, Snuba over ClickHouse for events, Postgres for metadata, Redis for queues. Deployed in your AWS account with retention policies, alerting and SSO configured.

LAYER 1 · INGEST

Sentry Relay

- Drop-in compatible with every sentry-sdk
- Rate limits and sampling per project
- PII scrubbing rules
- Multi-region ingest endpoints
- Auto-scaling on ECS or EKS

LAYER 2 · STORAGE

ClickHouse + Postgres

- **ClickHouse**: events, spans, replays
- **Postgres**: projects, users, alerts
- **Redis**: queues and buffers
- **S3**: replay payloads and uploaded source maps
- Retention per project: 7, 30 or 90 days

LAYER 3 · UI + OPS

The Sentry you know

- Sentry at sentry.yourcompany.com
- SSO via Okta, Azure AD or Google
- Slack, PagerDuty, OpsGenie webhooks
- Daily snapshots, restore tested
- A quarterly upgrade cadence

Decisions to make in week one

DECISION	OUR DEFAULT	WHY
Where it runs	Your AWS account, on ECS or EKS, multi-AZ	Your data, your residency, your bill
Ingest	Relay, with rate limits and sampling per project	A crash loop in one app cannot flood the cluster
Storage	ClickHouse for events, Postgres for metadata, S3 for payloads	Cost follows bytes kept, not events counted
Retention	Per project: 7, 30 or 90 days	Set before the first event lands, not after the first disk alarm
Symbol files	Uploaded from CI with every release	Readable stack traces from day one (page 7)
Sign-in	Your SSO and SCIM from day one	Offboarding happens where it already does

License: Sentry is FSL (the Functional Source License). Self-hosting it for your own company is free and unrestricted; the license only restricts offering Sentry as a service to third parties. Keep every Terraform module and Helm chart in Git from day one.

Events roll up into issues. *Releases tie them to code.*

Sentry ships its own schema, so you do not design tables. You do decide what goes into them: the projects, the releases, the tags on every event. Agree it before the first DSN swap, because grouping, alerts and suspect commits read from it.

RECORD · STORE	KEY FIELDS
Teams and members · Postgres	team, member, role, SSO identity, SCIM id
Projects and keys · Postgres	project slug, platform, owning team, DSN, rate limit, retention days
Releases · Postgres	version, dist, commits, deploys per environment, first and last event
Issues · Postgres	fingerprint hash, project, first seen, last seen, status, assignee, times seen
Events · ClickHouse	event_id, project, issue, timestamp, release, environment, platform, tags, user, stack trace
Transactions and spans · ClickHouse	trace_id, span_id, parent_span_id, op, description, duration, status
Replays · ClickHouse + S3	replay_id, linked trace and error ids in ClickHouse; recording segments in S3
Source maps and debug files · S3	release, dist, debug_id, file name, platform (JS bundles, dSYMs, ProGuard mappings)
Alert and ownership rules · Git	conditions, filters, actions, frequency; path and module patterns to a team

TWO RULES THAT SAVE A REBUILD

Every event carries a release and an environment, set in SDK init from CI. Without them there are no regressions, no suspect commits and no source map to match. **Keep tags bounded**: user IDs and request IDs belong in the event's user and context fields, never as tags, or ClickHouse and every search slow down.

Four weeks, *parallel run included.*

This is the order we build in. Each week ends with something your engineers can use, and the third runs both instances side by side before sentry.io goes quiet.

WEEK	BUILD	DONE WHEN
1	**Inventory and foundations.** List every project, DSN, team, alert rule, ownership rule and integration. Terraform for ECS or EKS, ClickHouse, Postgres, Redis and S3; Sentry behind your SSO.	The inventory is signed off; engineers sign in
2	**Ingest, rules and symbols.** Relay with rate limits, sampling and PII scrubbing. Projects, teams and ownership rules recreated; Slack, PagerDuty and OpsGenie wired; source map and debug file upload in every CI pipeline; retention per project; daily snapshots and a restore test.	A staging error arrives symbolicated and pages the right team
3	**Parallel run.** Both instances receive production events for a week. Compare issue counts, grouping and alerts daily, and work a fix list. Load-test Relay and ClickHouse at peak.	Every alert sentry.io sent, your Sentry sent too
4	**Cutover.** One new DSN per project, swapped through an environment variable in CI. GitHub connected for suspect commits; the runbook handed over.	On-call signs off on the new Sentry

AFTER CUTOVER

Stop sending to sentry.io and keep the account **until your renewal date**, then cancel. Old issues stay one click away while the team settles in, and nobody is in a hurry.

The DSN swap

Sentry SDKs accept any DSN, hosted or self-hosted. Keep each DSN in an environment variable, never in code, and the swap is a CI change most teams finish in an afternoon. Mobile apps pick up the new DSN with their next release, so ship it early and run both for that window.

Readable traces, one issue per bug. *Quiet pagers.*

Installing Sentry is the easy part. Keeping it useful at 50M events a month is the real work: stack traces that point at your source, issues that match real bugs, and alerts that wake someone only when it matters.

1. **Symbolicate every build.** Minified JavaScript needs source maps, iOS needs dSYMs, Android needs ProGuard or R8 mappings, native code needs debug files. Upload them with sentry-cli in the same CI job that builds the release, keyed by release and debug ID, and fail the build when the upload fails.
2. **Upload before you deploy.** An event processed before its source map arrives stays minified. Keep source maps off your public CDN: uploaded to Sentry, they stay private and symbolication never waits on a fetch.
3. **Mark your own frames in-app.** Grouping hashes the stack trace. Frames from your code should decide the issue, not framework or vendor frames. Set in-app rules in each SDK and stack trace rules per project.
4. **Fingerprint what the default gets wrong.** Timeouts, network errors and messages with IDs inside them split one bug into hundreds of issues, or fold many bugs into one. Write fingerprint rules for those, review the top issues weekly and merge duplicates.
5. **Alert on issues, not events.** Page on a new issue, a regression or a spike in a known one, routed by ownership rules to the team that owns the code. Never page on every event. Relay's rate limits keep one crash loop from flooding ClickHouse and the pager.

Watch the watcher

Run a small, separate monitor of Sentry itself: Relay accepts and drops, Snuba consumer lag, ClickHouse disk and query time, Postgres size and the nightly snapshot. When the error tracker fails, it fails silently, unless something else is watching it.

Measure the saving

Track events accepted a month per project, new issues a week and alerts a week per team, next to the AWS bill. Those numbers prove the saving and show which team to talk to first.

Move it, launch it, *run it.*

Migration

- ☐ Inventory projects, DSNs, teams, alert rules and integrations
- ☐ Recreate projects and teams with the same slugs
- ☐ Move ownership rules and the alert rules your team acts on
- ☐ Set release and environment in every SDK init
- ☐ Source map and debug file upload in every CI pipeline
- ☐ Connect GitHub for releases and suspect commits
- ☐ Agree retention per project before ingest
- ☐ Run both instances for a week and compare

Go-live

- ☐ Every project sending to its new DSN
- ☐ Engineers signing in through your SSO
- ☐ Alert routes match today's Slack, PagerDuty and OpsGenie
- ☐ Stack traces symbolicated on every platform
- ☐ Meta-monitoring on Sentry itself
- ☐ A restore from the daily snapshot proven
- ☐ Rollback ready: the old DSNs one variable away

Running it

- ☐ **Always on:** meta-monitoring, security patches
- ☐ **Weekly:** top issues, noisy alerts, events by project against rate limits
- ☐ **Monthly:** read Sentry's release notes, flag anything material
- ☐ **Quarterly:** upgrade through staging first; cost review, the AWS bill against the sentry.io bill you replaced; retention review
- ☐ **Yearly:** a restore drill and an access review

Who owns it

Name one person before launch. Sentry is how you learn the product is broken; someone has to answer when Sentry itself is the thing that broke.

Five ways a move off sentry.io *stalls.*

1. Switching vendors and calling it a saving

Bugsnag and Rollbar are cheaper than sentry.io Business, with the same hosted-vendor lock-in. Your team re-learns a new UX, loses Sentry-specific integrations, and you are back at this analysis in 18 months.

2. Moving errors into Datadog

Datadog Errors is solid if you already pay for Datadog. Its per-host pricing scales aggressively, and it often costs more than sentry.io Business at the same engineer count.

3. A smaller clone

GlitchTip is good and accepts Sentry SDKs. Its community is smaller, it has fewer SDKs, and it lacks Replay and Profiling. For a team on the full Sentry feature set, it is a step back.

4. A docker-compose in production

Sentry's docker-compose runs in 30 minutes and looks finished. Production-grade ops is ~3 weeks of senior engineering: ClickHouse tuning, multi-tenancy, alerting, retention and an upgrade cadence. A $5K freelancer delivers the docker-compose and leaves the rest with you.

5. Skipping the parallel run

A week of both instances finds the project whose DSN nobody listed and the alert the mobile team depends on.

And one that is not a mistake: negotiating

Sentry discounts Business and Enterprise 15–25% on multi-year commits. Do it if the move is a year away; self-hosting keeps 80%+ of the bill at 50M+ events a month, and a discount captures less than a quarter of that.

ALLIED BizTech SOLUTIONS

RATHER HAVE IT BUILT?

We build this exact plan. *Fixed price.*

The same scope, stack and four weeks, delivered by a senior team that has shipped 250+ products.

$11,600 fixed up to 100M events a month; $18,300–$26,600 for 500M+ events, multi-region; agreed before we start

Live in 3–4 weeks, with a one-week parallel run next to sentry.io before cutover

A DSN swap, not a migration: one new DSN per project, source maps automated in CI

90-day warranty, and the Terraform and Helm config is in your GitHub

We can run it for $3,600 a year: monitoring, upgrades and SLA support

alliedbiz.tech/rebuild/sentry-alternative

Allied BizTech · 15+ years · ISO 9001:2015 · 250+ products shipped. Sentry is a trademark of its owner, named here for identification only; this guide is independent and not endorsed by Sentry. Prices as of May 2026.