ALLIED BizTech SOLUTIONS

FIELD GUIDE · 2026

Rebuild Datadog yourself.
The plan we follow.

For engineering teams on 200 to 3,000+ hosts who want to own their metrics, logs, traces and alerts, and stop paying per host and per million log lines.

- → Is it worth it, at your size
- → What to rebuild and what to drop
- → The LGTM stack, in three layers
- → Migration and go-live checklists
- → The signal model
- → A week-by-week build plan
- → Retention, cardinality, alerts
- → Where teams get stuck

Datadog grows with your fleet. *LGTM grows with bytes.*

Datadog bills per host for APM and infrastructure, per million log lines, per synthetics test and per RUM session. Each line scales on its own. A rebuild replaces them with an AWS bill for compute and S3 storage, which grows far more gently.

FLEET	DATADOG A YEAR	REBUILT: YEAR ONE / AFTER	SAVED IN YEAR ONE
50 hosts · APM + Infra · 50GB logs a day	~$24,000	~$20,600 / $4,000	~$3,400
200 hosts · APM + Logs + Infra · 300GB a day	~$96,000	~$31,600 / $15,000	~$64,400
500 hosts · full suite · 1TB a day	~$240,000	~$48,600 / $32,000	~$191,400
1,000 hosts · 5TB logs a day · multi-region	~$540,000+	~$124,200 / $88,000	~$415,800
3,000+ hosts · global scale	~$1,800,000+	~$326,200 / $290,000	~$1,473,800

Datadog list prices, May 2026. "Rebuilt" runs Grafana, Loki, Tempo and Mimir on AWS EC2 with S3 for long-term storage, plus our fixed build price: $16,600 up to 500 hosts, $36,200 (the top of the $24,100–$36,200 scale tier) from 1,000 hosts. If your own team builds it, put your engineering cost in that place.

THE RULE OF THUMB

At 50 hosts, year one saves ~$3K and $20K a year comes back after. **At 200 hosts, the rebuild saves ~$64K in year one and ~$81K a year after.** From 500 hosts, year one alone saves more than $190K.

Who you need

- **A senior infrastructure engineer** at home with Kubernetes, Terraform, Prometheus and S3. Production-grade Mimir tuning, multi-tenant config, retention and on-call setup take 4–6 weeks of that person's time.
- **Your platform or SRE lead**, two hours a week: the dashboards and alerts that matter, the parallel-run sign-off.
- **An owner after launch**: upgrades, capacity, retention, the alert review, and on-call for the monitoring itself.

02 · SCOPE

Rebuild the 15–25% you use. *Drop the rest.*

Datadog sells a product for every signal. A typical team opens metrics, logs, traces, dashboards and alerts every day. Those are the project; the specialist products have better specialist tools.

REBUILD ONE FOR ONE

- **APM**: distributed traces, service maps, error rates, p50/p95/p99 latency (Tempo + Mimir)
- **Log search and retention**: structured logs, log-to-trace links, the saved searches your team uses (Loki)
- **Metrics dashboards**: the 5–10 boards your team opens daily, in Grafana
- **Infrastructure metrics**: CPU, memory, disk and network per host and container, plus your service metrics
- **Alerting and on-call routing**: Alertmanager to PagerDuty, OpsGenie and Slack, same escalation shape
- **SLO tracking** with burn-rate alerts
- **Continuous profiling**: CPU and memory, with Pyroscope

REBUILD, SIMPLER

- **RUM**: Grafana Faro, a plainer UI, same data
- **Notebooks**: Grafana Explore and saved queries
- **Watchdog**: threshold and rate-of-change rules, custom anomaly detection where it earns its place

DROP ON PURPOSE

- **Cloud Security Management**: Wazuh self-hosted, or Panther
- **Synthetics and uptime checks**: Checkly, or UptimeRobot
- **Network Performance Monitoring**: ThousandEyes, or keep Datadog NPM for that one feature
- **Cloud Cost Management**: Vantage, AWS Cost Explorer or Infracost

ADD WHAT PER-HOST PRICING DISCOURAGES

- New hosts at no fee; storage scales with bytes
- Retention per workload: hot logs 7 days, cold logs 365 days on S3
- Query access from any tool, with no vendor API rate limits

Three layers. *All open source.*

The LGTM stack from Grafana Labs is the open-source observability standard: Mimir for metrics, Loki for logs, Tempo for traces, Grafana on top. Each backend keeps its data in S3, so storage is cheap and retention is a setting.

LAYER 1 · INGEST

OpenTelemetry collectors

- OTel Collector daemonset, in place of dd-agent
- A Datadog receiver, so existing agents keep sending
- Sampling and filtering at ingest
- Routing per service to the right backend
- Auto-scaling on EKS or ECS

LAYER 2 · STORAGE

The LGTM backends

- **Mimir**: Prometheus-compatible metrics
- **Loki**: log aggregation
- **Tempo**: distributed traces
- **Pyroscope**: continuous profiling
- **S3** behind all four, retention per tenant

LAYER 3 · UI + ALERTING

Grafana + Alertmanager

- Dashboards converted from Datadog
- Alertmanager to PagerDuty, OpsGenie, Slack
- SSO via Okta, Azure AD or Google
- SLOs with burn-rate alerts
- Explore and saved queries

Decisions to make in week one

DECISION	OUR DEFAULT	WHY
Where it runs	Your AWS account, on EKS	Your data, your residency, your bill
Ingest	OTel Collector with its Datadog receiver	Nothing re-instrumented on day one; move services to OTel SDKs as you touch them
Storage	S3 for everything long-term, disk for hot data only	Cost follows bytes kept, not hosts watched
Retention	Per workload: 7 days hot, 365 days cold	Set before the first byte lands, not after the first bill
Labels	A short, fixed list (next page)	Labels decide series count, and series count decides cost
Sign-in	Your SSO from day one	Offboarding happens where it already does

Licenses: Grafana, Loki, Tempo and Mimir are AGPLv3. Running them for your own company on your own AWS carries no restriction, and you can fork the code. Keep every Helm chart and Terraform module in Git from day one.

Five signals. *Labels tie them together.*

An observability stack has no customer tables, but it has a data model: the signals you keep and the labels on them. Agree it before the first collector ships, because every dashboard, alert and cost line reads from it.

SIGNAL · STORE	KEY FIELDS AND LABELS
Resource labels (on everything)	service, env, region, cluster, team, version; the same names on metrics, logs, traces and profiles
Metrics · Mimir	metric name, resource labels, a few bounded labels (status_code, method, route template); samples of timestamp and value
Logs · Loki	stream labels: service, env, level, cluster only; the line itself as JSON; trace_id and request_id inside the line, never as labels
Traces · Tempo	trace_id, span_id, parent_span_id, service.name, span name, duration, status, resource attributes
Profiles · Pyroscope	service, profile type (CPU, memory, goroutines), resource labels
Alert rules · Git	name, query, for, severity, team, runbook_url, datadog_monitor_id
SLOs · Git	service, good-events query, total-events query, objective, window, burn-rate thresholds
Dashboards · Git	Grafana JSON, one folder per team, variables for env and service, datadog_dashboard_id
Retention · per tenant	tenant, signal, hot days, cold days, series limit, ingest rate limit

TWO RULES THAT SAVE A REBUILD

No label with unbounded values: user IDs, request IDs and raw URLs belong in the log line or the span, never in a metric or stream label, or the series count and the bill explode. **Put trace_id in every log line**: it is what turns three tools into one, a log that opens its trace.

Six weeks, *parallel run included.*

This is the order we build in. Each week ends with something your on-call engineers can use, and the last two run both stacks side by side before Datadog goes quiet.

WEEK	BUILD	DONE WHEN
1	**Inventory and foundations.** List every dd-agent, integration, monitor, dashboard and log pipeline. EKS, S3 buckets and Terraform; Grafana behind your SSO; the label list agreed.	The inventory is signed off; engineers sign in to Grafana
2	**Ingest and storage.** The OTel Collector with its Datadog receiver in staging; Mimir, Loki and Tempo on S3; retention and limits per tenant.	Staging metrics, logs and traces show in Grafana
3	**Dashboards and alerts.** Convert the Datadog JSON exports, daily boards first. Translate the monitors that fired in the last 90 days. Alertmanager routes to PagerDuty and Slack.	On-call can work a shift from Grafana
4	**Production ingest.** Collectors on every cluster and host, sampling and filtering at ingest, Pyroscope, SLOs with burn-rate alerts. Load-test the write path.	Peak traffic lands with nothing dropped
5	**Parallel run.** Both stacks get the same data for a week. Compare alerts and dashboards daily and work a fix list.	Every page Datadog sent, the new stack sent too
6	**Cutover.** Agents point only at the collectors, Datadog integrations off, the runbook handed over.	On-call signs off on the new stack

AFTER CUTOVER

Turn Datadog ingestion off and keep the account **read-only until your renewal date**, then cancel. Old history stays one click away while the team settles in, and nobody is in a hurry.

Cost follows series and bytes. *Control both.*

Installing LGTM is the easy part. Keeping it fast, cheap and quiet at 200 hosts and beyond is the real work: how many series you keep, how long you keep the bytes, and which alerts wake someone up.

1. **Cap cardinality at the door.** Every unique label combination is a series in Mimir. Drop or rewrite unbounded labels in the collector, set series and ingest limits per tenant, and review the top series by metric and team every week.
2. **Keep Loki's index small.** Loki indexes labels, not text, which is why it is cheap. Keep stream labels few and bounded; search inside lines with filters. Forensics-heavy teams add ClickHouse for free-text search.
3. **Set retention per workload.** Hot data on disk for 7 days, cold data on S3 for 365, longer where compliance asks. Downsample old metrics, and sample traces at ingest: keep every error and slow trace, a share of the rest.
4. **Move only the alerts that matter.** Translate monitors that fired in the last 90 days; give each an owner, a severity and a runbook link. For user-facing services, alert on SLO burn rate, not CPU.
5. **Replace Watchdog with rules.** Threshold and rate-of-change alerts cover about 80% of what anomaly detection is used for. Add custom detection only where a rule cannot express the problem.

Watch the watcher

Run a small, separate monitor of the stack itself: ingest errors, dropped samples, query latency, S3 errors and disk. When monitoring fails, it fails silently, unless something else is watching it.

Measure the saving

Track bytes ingested a day, active series and alerts a week per team, next to the AWS bill. Those four numbers prove the saving and show which team to talk to first.

Move it, launch it, *run it.*

Migration

- ☐ Inventory agents, integrations, monitors and dashboards
- ☐ Put the OTel Collector's Datadog receiver in front of existing agents
- ☐ Export dashboards as JSON and convert; about 85% convert automatically, rebuild the rest by hand, most-used first
- ☐ Keep the Datadog ID on every dashboard and alert
- ☐ Translate only the monitors that fired in the last 90 days
- ☐ Re-point log shippers (Fluent Bit, Vector, Filebeat) to Loki
- ☐ Agree retention per workload before ingest
- ☐ Run both stacks for a week and compare

Go-live

- ☐ Collectors on every cluster and host
- ☐ Engineers signing in through your SSO
- ☐ Alert routes match today's PagerDuty, OpsGenie and Slack
- ☐ Meta-monitoring on the stack itself
- ☐ A restore from S3 proven
- ☐ Rollback ready: agents one config change from Datadog

Running it

- ☐ **Always on:** meta-monitoring, security patches
- ☐ **Weekly:** top series and log volume by team, noisy alerts
- ☐ **Monthly:** LGTM upgrades through staging first, dashboard and alert review
- ☐ **Quarterly:** cost review, the AWS bill against the Datadog bill you replaced, retention review
- ☐ **Yearly:** a restore drill and an access review

Who owns it

Name one person before launch. Monitoring is how you learn everything else is broken; someone has to answer when the monitoring itself is the thing that broke.

Five ways a rebuild *stalls.*

1. Switching vendors and calling it a saving

New Relic, Honeycomb and Chronosphere are the same hosted-vendor model under another brand. New Relic's ingest overage caps the win; Honeycomb's per-event pricing is great at low volume and expensive at SaaS scale; Chronosphere's hosted pricing is comparable to Datadog's at scale.

2. Grafana Cloud past 500 hosts

Grafana Cloud is a great option below 200 hosts. Above 500, its pricing approaches Datadog's, and self-hosting the same stack is far cheaper.

3. A demo install in production

A quick-start install looks finished. Production-grade Mimir tuning, multi-tenant config, retention and on-call setup are 4–6 weeks of senior infra time. Plan for them.

4. Unbounded labels

One label holding user IDs turns one series into one per user. The bill moves from Datadog to your S3 and compute, and queries slow down with it.

5. Skipping the parallel run

A week of both stacks finds the monitor nobody listed and the dashboard the head of engineering opens every Monday.

And one that is not a mistake: negotiating

Datadog discounts Pro and Enterprise 15–25% on multi-year commits. Do it if the rebuild is a year away; it captures less than a third of what self-hosting does.

RATHER HAVE IT BUILT?

We build this exact plan. *Fixed price.*

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

$16,600 fixed up to 500 hosts and 1TB of logs a day: a senior engineer, a coder and a half-time delivery manager for 5 weeks; $24,100–$36,200 from 1,000 hosts; agreed before we start

---

Live in 4–6 weeks, with a one-week parallel run next to Datadog before cutover

---

Nothing re-instrumented: dd-agents send through the OTel Collector; dashboards convert from your JSON exports

---

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

---

We can run it for $6,000 a year: monitoring, monthly upgrades and SLA support

alliedbiz.tech/rebuild/datadog-alternative

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