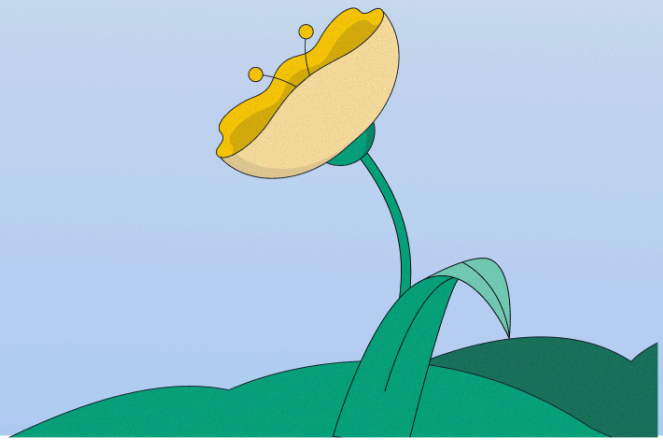
groundcover

Datadog Pricing Explained: Why Your Bill Keeps Growing

White Paper



Datadog is one of the most powerful observability platforms available. It is also one of the most expensive, and one of the hardest to predict costs for. If you have submitted a budget forecast and seen the actual invoice come in 30–40% higher, you are not misconfiguring anything. The pricing model is genuinely structured in a way that makes total cost of ownership difficult to calculate upfront.

This guide breaks down how Datadog charges, which combinations drive the biggest surprises, and why the bill tends to grow faster than your infrastructure does.

The structure: 20+ products, each billed separately

Datadog provides a single pane of glass to help developers get a full view of their entire stack. The breadth of their observability solution offers the promise to automatically catch issues before they escalate, detect critical issues in your pipelines before they even go into production with CI/CD testing, and rapidly identify bottlenecks, errors, heavy traffic issues, slow-running queries, and more.

Datadog is able to provide these capabilities with their full suite of observability including infrastructure monitoring, APM, log management, database monitoring, LLM observability, security, real user monitoring, CI visibility, and AI SRE investigations. These are all offered as separately priced products, and each requires a separate purchase and introduces its own usage-based charges.

Within each product, you pay based on usage volume: per host, per GB ingested, per million log events indexed, per million LLM requests monitored. The bill is the sum of all these dimensions, and they interact with each other in ways that are difficult to model before you have real production data.

According to Datadog's own investor presentations, roughly 56% of customers use fewer than four products. For a platform built around the promise of a single pane of glass, that number raises an obvious question: why aren't customers adopting more of the platform? The answer is less about product quality than pricing. Every additional capability introduces another budget decision, another procurement conversation, and another variable cost to manage. Teams naturally stick with the products they have already paid for.

Roughly 56% of Datadog customers use fewer than four of the available products.

Developers have understood this dynamic for years. Reddit and Hacker News are full of discussions about unexpected bills driven by custom metrics, high-cardinality tags, and log ingestion costs. The advice is rarely "collect more data." Instead, it's to control costs by reducing visibility: limit ingestion, aggressively manage cardinality, rate limit collectors, or

sample away most of the telemetry. One common recommendation is to sample 99% of logs from services that engineers are not actively investigating.

The result is that pricing doesn't just affect the bill. It changes engineering behavior. Instead of asking, "What data do we need to solve this problem?" teams are forced to ask, "What data can we afford to keep?"

> *"When you start telling engineers to log less because it's too expensive, something's gone wrong."*
>
> –Joe Garlick, Lead Architect, FlipCX

> *"Our Datadog bill is a threat, an ever-growing line item that threatens to consume what remains of our cloud spend budget."*
>
> –From a real engineering team's internal job description for a 'Datadog Whisperer': a role created specifically to reduce Datadog spend.

The five billing dimensions that compound

Understanding the bill means understanding how each major product category charges, and how those charges stack on top of each other.

1. Infrastructure: the base that everything else requires

Infrastructure monitoring charges per host per month: $15 on Pro, $23 on Enterprise. For a 500-node Kubernetes cluster on Enterprise, that is $11,500/month before any logs or traces are sent. Every other Datadog product (APM, database monitoring, network monitoring) requires a compatible infrastructure tier on those same hosts, so the infrastructure bill is a multiplier, not just a line item.

Custom metrics are allotted per host (100 on Pro, 200 on Enterprise). Beyond the allotment, the charge is $1 per 100 additional custom metrics per month, averaged across your entire account. A single service emitting high-cardinality metrics (tagged by user ID, request ID, or pod name) can push your whole account over the allotment without any deliberate change to instrumentation.

2. APM: per host, on top of infrastructure, with ingestion limits

APM costs $31–$40 per host per month in addition to infrastructure (or $36–$47 standalone). Each APM host includes an allotment of 150 GB of ingested spans and 1 million indexed spans per month. Overages are charged at $0.10/GB for additional span ingestion and $1.27–$2.50 per million additional indexed spans, depending on the retention period.

In practice, most teams running Datadog APM enable trace sampling at 25–50% to stay within their ingestion allotment. One digital health platform running multi-region Kubernetes workloads sampled production traces to 25% and non-production to 5%, and routed logs and infrastructure metrics to a separate Grafana stack rather than pay Datadog to ingest them. The consequence is that during an incident, when you need complete trace coverage most, you are working from an incomplete picture.

3. Logs: ingestion, indexing, and retention are three separate charges

Log billing is where the largest surprises occur, because it layers three distinct cost dimensions:

- **Ingestion**: $0.10/GB of uncompressed data received by Datadog, regardless of whether those logs are indexed or immediately discarded. If 85% of your logs are filtered out after ingestion, you still pay for 100% of the volume that arrived.
- **Standard Indexing**: $1.70 per million log events, for logs that are searchable and can trigger monitors. Default retention is 15 days. Extending to 30 days requires a plan change; anything longer means a sales conversation.

- **Flex Logs**: A cheaper storage tier at $0.05/million events with retention up to 15 months. But Flex Logs removes support for monitors and Watchdog Insights. Teams that route logs to Flex to save money lose the ability to alert on those logs in real time, creating a forced choice between cost and coverage.

There is also a rehydration cost that catches teams off guard: when archived logs are pulled back into Datadog for analysis, the charge is $0.10 per compressed GB scanned, not per GB retrieved. If you need to find a specific event in a large archive, you pay for the full scan even if you retrieve only a few lines. This cost arrives at exactly the moment of an incident, when you least want a surprise.

4. The on-demand surcharge

Every Datadog product carries an on-demand rate approximately 50% higher than the annual committed price. Any usage above your committed volume (an incident-driven log spike, a traffic surge, a deployment that briefly inflates host count) is billed at the on-demand rate automatically. The annual committed rate is a floor, not a ceiling.

5. Each new product adds a new unpredictable variable

The product catalog keeps expanding. LLM Observability is now priced at $8 per 10,000 monitored LLM requests per month. The newest AI product, Bits AI SRE Investigations, runs $500/month per 20 investigations, a standalone charge with no bundling into existing contracts. Every new capability that teams want to adopt requires a separate evaluation of what it will cost at their scale.

What the log bill looks like in practice

```
None
Setup: 7 TB/day of logs. Only 15% indexed by Datadog.

What Datadog charges:
  Ingestion fee (100% of data, even what's discarded):
    7,000 GB/day × 30 days × $0.10/GB  =  $21,000/month

  Indexing fee (the 15% you actually keep and search):
    charged per million log events at $1.70/million
    15-day default retention; 30+ days requires a sales call

  Result: you pay to ingest logs you never see,
          then pay again to index the ones you keep.

Same data stored in your own S3 (via groundcover BYOC):
  210,000 GB × $0.023/GB × ~10% compression  =  ~$483/month
  100% of logs retained. Always queryable. No rehydration.

The original notes from a real groundcover customer: $17,850/month
was spent on logs that went straight to the trash.
```

Why the total is hard to predict

None of the individual rates are secret. They are listed on the pricing page. The unpredictability comes from the fact that the inputs driving each dimension change independently, and not always in ways that are visible before the invoice arrives.

The variables that make Datadog bills hard to forecast:

- Host count fluctuates daily with Kubernetes auto-scaling
- Log volume spikes during incidents, precisely when you need it most
- Custom metric cardinality grows silently when developers add tags
- APM span volume scales with traffic; overages hit at $0.10/GB
- LLM request volume grows with every new AI feature shipped
- On-demand surcharge (~50%) applies automatically to any excess

The compounding effect is the core problem. Teams size their contracts based on current usage. Six months later, infrastructure has grown, a high-cardinality service has been added, an AI feature has shipped, and the bill reflects the product of all those changes multiplied across every dimension simultaneously.

What teams typically do about it

There are three patterns that emerge when teams try to control a growing Datadog bill, and each one trades observability coverage for cost.

The first is filtering and sampling: drop logs below a severity threshold before they reach Datadog, sample traces to 25–50%, and enforce cardinality limits on metric tags. This works, but it means you may not have the log line or trace that explains the next incident.

The second is selective activation: only use the Datadog products already under contract and resist expanding to others. Route lower-priority logs to Flex to reduce indexing costs, accepting that you can no longer alert on them in real time. This is now the modal approach for mature Datadog customers, and it means deliberately using a fraction of the platform you are paying for.

The third is rearchitecting around the data storage model entirely. The root cause of the bill (and the reason the two approaches above are necessary) is that Datadog stores your data in their cloud and charges you a margin on every GB that flows through. Teams evaluating alternatives are increasingly looking at architectures where observability data stays in their own cloud account, eliminating the per-data-unit charge at the source.

Why the switch gets deferred

Most teams reading this already know their Datadog bill is too high. The reason they stay is rarely a defense of the pricing. It is an estimate of what leaving costs, and that estimate is usually built on an assumption that no longer holds.

The assumption is that migrating means rebuilding. Years of dashboards, hundreds of monitors, alert routing, all recreated by hand in a new query language. That was a fair description of an observability migration a few years ago. It is not what the work looks like now.

groundcover's Datadog migration connects with a Datadog API key and a read-scoped application key, then pulls your monitors, dashboards, and the data sources those assets depend on. The fetch takes about ten seconds. Every asset comes back labeled: fully supported and migratable as-is, partial with warnings to review, or unsupported and needing manual attention. You see that breakdown before anything installs. Dashboards carry over their layout, widget positions, query logic, filters, time ranges, and formatting, and you can preview each one before installing it. Monitors install individually or in bulk. Warnings do not block a migration; they tell you what the query translation changed so

you can verify the behavior afterward. The keys are used for the fetch and discarded rather than stored.

What that leaves is a review exercise rather than a rebuild. Connect any data sources the assets depend on, work through the partial and unsupported list, verify that your handful of genuinely critical dashboards return the data you expect, then bulk migrate the rest. The long tail of dashboards nobody has opened in a year does not need careful attention, and pulling usage data first tells you which ones those are.

Getting data flowing is the shorter half of the job. Because collection runs through an eBPF sensor rather than per-language libraries, there is no instrumentation project on the other side of the migration: no tracing library per service, no code changes, no redeploys, no coverage matrix to maintain.

Three things are still real, and they are worth planning around rather than discovering:

Vendor review has its own clock. A new platform holding production telemetry means a security review and a data processing agreement, and in regulated industries a BAA. Engineering cannot make this go faster by working harder. The data residency portion of that conversation is usually shorter when the telemetry stays inside your own VPC, but the review still has to happen and it still takes calendar time.

Contract timing decides more than it should. Annual and multi-year commitments, products co-termed onto a single renewal date, credits that expire unused. A team that decides in month three of a twelve-month term usually waits, and by the renewal date usage has grown and the replacement commit is larger than it would have been. Start the evaluation 90 to 120 days ahead of renewal so the date is a decision point rather than a deadline you missed.

Familiarity is worth real money during an incident. Nobody has muscle memory for a new tool, and the first ten minutes of an outage is where that shows. This is the argument for running both platforms in parallel through at least one full incident cycle before committing, which is also the only way to compare them on real failures rather than on a demo. Parallel running costs you a month or two of overlap, and it is the cheapest insurance in the whole project.

None of this is trivial. But it is scheduling, paperwork, and a review queue rather than a rebuild, and it is worth re-estimating rather than carrying an old number forward.

groundcover case study

b.well: from 25% of traces to all of them

That digital health platform was b.well Connected Health — the healthcare data backbone behind a leading frontier AI company's flagship product, running FHIR-based multi-region workloads on Kubernetes and AWS.

Under Datadog, cost control set observability policy. Production traces sampled to 25%, non-production to 5%, logs and infrastructure metrics kept out of Datadog entirely. Spend still swung between $15K and $30K a month, so engineers spent their time watching ingestion instead of watching systems.

In the weeks before the AI backbone launch — load-testing at a scale they had never operated at — that was the wrong stack to be holding.

After moving to groundcover they turned sampling off completely and now ingest roughly 10x more application data for under $10K/month, flat. Full-fidelity traces surfaced a 30-second application-level overhead and Kubernetes autoscaling that wasn't responding fast enough to traffic spikes.

"I don't think we ever would have found that issue if we weren't able to visualize the traces the way we were with groundcover." — Brian Kane, Senior Site Reliability Engineer, b.well

The cost that never reaches the invoice

Before turning to a different pricing model, there is one cost worth separating out, because it is the only one in this paper that you pay to your cloud provider rather than to your observability vendor.

Monitoring agents consume resources on the hosts they monitor. Some consume a great deal. In a benchmark run against a Go HTTP server under a constant load of 3,000 requests per second, measuring CPU and memory before and after adding each agent, the results diverged sharply:

Agent	CPU above baseline	Memory above baseline
groundcover (Flora)	+9%	+0%
New Relic Pixie	+32%	+9%
OpenTelemetry	+59%	+27%
Datadog	+249%	+227%

The more consequential result came from constraining the application to 1,000 mCPU, which is closer to how workloads actually run in Kubernetes. Under that limit, the overhead did not simply show up on a graph. It ate throughput. The application served 71% fewer requests with the Datadog agent attached, 19% fewer with OpenTelemetry, and 12% fewer with Pixie.

Read that as a cost, not a performance footnote. Capacity consumed by the agent is capacity you provisioned and paid for, and if serving the same traffic under the same limits requires more replicas or larger nodes, you pay your cloud provider for that difference. Then you pay Datadog per host for the additional hosts. An agent-heavy stack raises the two largest line items at once, and neither increase is visible as an observability cost. The infrastructure bill grows, the per-host bill grows with it, and nothing in either invoice says why.

The same benchmark measured total resource consumption, agent plus induced overhead, and found groundcover used 73% less CPU and 74% less memory than Datadog.

A different pricing model

The structural reason Datadog's bill is hard to control is that the business model is built on data volume. The more telemetry you send, the more Datadog charges. That creates a direct misalignment: the platform is most valuable when you send everything, but the pricing penalizes you for doing so.

groundcover is built on a Bring Your Own Cloud (BYOC) architecture that inverts this. Because observability data is processed and stored inside the customer's own VPC rather than in a SaaS cloud, there is no per-GB, per-event, or per-request charge. groundcover charges a flat per-host rate that covers the full platform: APM, logs, metrics, distributed traces, LLM observability, and every feature released going forward.

The rate is published rather than quoted. Pro is $30 per host per month and Enterprise is $35, both on BYOC, with a fully on-premise deployment at $50 for regulated and air-gapped environments. There is a free tier at 12 hour retention for evaluation. A host is a Kubernetes node or any Linux host of any size or machine type, and billing is calculated on the monthly average of monitored hosts rather than the peak, so an autoscaling event or a deployment that briefly doubles your node count does not produce a bill you have to explain.

That last detail is worth sitting with, because it is the direct inverse of the on-demand surcharge described earlier. On a volume-based model, the moments when your systems generate the most telemetry are the moments the meter runs fastest. On a per-host average, a traffic spike is just a traffic spike.

Why a flat per-host rate is possible

A reasonable reaction to a flat rate is suspicion. If the platform ingests everything and charges by the host, who absorbs the cost of the data, and what happens at renewal when someone works out the margin? The answer is architectural rather than promotional, and it has two parts.

The first is where collection happens. groundcover collects through an eBPF sensor, which runs sandboxed programs inside the Linux kernel. That gives it visibility into every process on the host without a tracing library in your application, without code changes, and without redeployments, across any language runtime. Deployment is a single command, and coverage arrives with it rather than being built up service by service over quarters. Because the sensor observes out of band from the applications it watches, rather than running inside them, it is also the reason the overhead numbers above look the way they do. It also means the sensor can capture full trace payloads, including bodies and headers, rather than the pre-aggregated summary a per-language agent typically forwards.

The second is where processing happens. Rather than shipping raw telemetry to a vendor to be parsed, indexed and billed, the sensor processes and reduces data in place, on the node, in a stream. Cardinality is not a cost input, because nobody is billing per time series, which is why there is no metrics allotment to blow through when a developer adds a tag. Sampling is not needed to control spend, because volume is not the meter. And the data that is retained never leaves your cloud, so groundcover is not paying to store it either.

Those two facts are what make the pricing sustainable rather than promotional. A vendor that stores your data in their cloud has to charge for volume, because volume is their cost. A vendor that processes at the edge and stores in your account does not have that cost to pass on.

What changes when there is no per-byte charge:

→ 100% of customers use APM (vs. ~25% on Datadog)

→ Log retention defaults to 60–365 days, no sales conversation required

→ All logs are always queryable (no Flex trade-offs, no rehydration costs)

→ Trace sampling is off by default, full production coverage

→ LLM observability and AI features are included, no per-request billing

→ Teams send 5–10x more telemetry data than they did on Datadog

Retention deserves a closer look, because this is where Datadog forces the sharpest trade-off. In groundcover, retention is configured per data type in the app: traces, logs, events, APM, and monitor issues each get their own policy. Each policy has a default retention period, an optional point at which data moves to lower-cost cold storage, and up to twenty indexes. An index is a query filter plus its own retention period, so you can keep production error logs for ninety days, everything else in production for thirty, and staging for seven, and change any of it later. Rules evaluate top to bottom and the first match wins.

The distinction from Flex Logs matters. On Datadog, moving logs to the cheap tier costs you the ability to monitor them. In groundcover, choosing what to keep and for how long is a retention policy, not a decision to give up alerting on a class of data.

The pricing model determines how much visibility a team actually has into their systems, not just what they could have in theory. A model that charges per data unit creates a structural incentive to send less data. One that charges per host creates an incentive to instrument everything.

Side-by-side: where the cost structure differs

The table below covers the dimensions most likely to produce a surprise on a Datadog invoice. For a full feature and pricing comparison, see the groundcover vs. Datadog guide.

Dimension	Datadog	groundcover
Pricing basis	Per host, per GB, per million events, per million requests, per product	Flat per host: $30 Pro, $35 Enterprise, $50 on-prem
How volume is counted	Committed volume per dimension, overages billed automatically	Monthly average of monitored hosts, peaks not billed

Instrumentation	Tracing library per service per language, code changes and redeploys	eBPF sensor, one command, no code changes
Agent overhead	Measured at +249% CPU and +227% memory on a benchmarked service	Measured at +9% CPU and +0% memory
Log ingestion	$0.10/GB, charged even for logs you discard	Stored in your S3; no per-GB ingestion charge
Log retention	15 days default; 30+ days = sales call; monitors lost on Flex	60–365 days, fully queryable, no extra cost
Log rehydration	$0.10/compressed GB scanned. Pay for the search, not just results	No rehydration; logs are always live
APM / traces	$31–$40/host/month; overage charged per GB and per million spans	Included. All hosts. No sampling required.
Trace detail	Sampled spans, typically 25–50% in cost-managed accounts	Full payloads including bodies and headers, no sampling
Custom metrics	100–200/host allotted; overages at $1/100; spikes silently	Included; unlimited cardinality, no allotment
On-demand surcharge	~50% premium over annual rate on every product, automatically	No on-demand penalty

Feature access	20+ products, each contracted and billed separately	One SKU, all features included
Where data lives	Datadog's cloud; you pay their margin on every GB retained	Your VPC; you pay cloud rates directly

To put this in concrete terms: one team running ~700 Kubernetes nodes with 5 TB/day of logs and 500K custom metrics was paying $2.54M/year on Datadog. The same setup on groundcover cost $297K/year, an 87% reduction, with full APM enabled and no trace sampling.

Where this comparison has limits

A pricing argument is more useful when it says where it does not apply. Three cases worth being direct about:

Datadog's product breadth is still wider in places. The integration catalog is the largest in the category, and some Datadog product lines have no direct equivalent here: the security suite spanning posture management, workload protection and SIEM, incident management and on-call, and CI visibility. If your contract is load-bearing for any of those, this is not one platform replacing another. It is a question of which parts move and what you keep paying for, and the savings math should be run on the portion that actually moves.

Per-host pricing only helps if you have hosts, and volume behind them. Billing counts Kubernetes nodes and Linux hosts, so an estate running mostly on Lambda and Fargate with little persistent compute does not map cleanly onto the model. The same is true of collection: an eBPF sensor needs a kernel to run in. Neither does a small footprint sending modest volume benefit much: a team on a handful of hosts is not the team paying to ingest logs it never reads, and Datadog's lower tiers may well come out cheaper. The savings described in this paper come from decoupling cost from data volume, so they scale with how much telemetry you send per host. If you are not sending much, there is not much to recover.

BYOC moves your spend rather than removing all of it. This is worth being precise about, because BYOC is often misread as self-hosting. The groundcover control plane deploys and manages the backend inside your own cloud, including patching, health monitoring and scaling, so you are not operating a storage cluster. What does change is where the cost appears: cloud provider fees, block storage and object storage land on your cloud bill rather than on a vendor invoice. groundcover's own cost calculator breaks these out as separate line items alongside the license, which is the honest way to read a total. The combined figure sits far below volume-based pricing, but it arrives in a different budget line, and someone should own that reallocation before finance encounters it as a surprise.

A short self-test. Is most of your bill driven by data volume rather than seats or hosts? Do you run a real host footprint? Are you sampling traces or dropping logs for cost reasons rather than signal reasons? Three yeses and this paper applies to you directly. Two and the savings are real but smaller than the headline numbers above. One or none, and your observability costs have a different cause worth diagnosing before you evaluate anything.

Next steps

Wherever you are in your Datadog migration journey, here are a few resources to guide your next steps:

- If you want to benchmark your current Datadog spend, the groundcover vs. Datadog comparison guide covers a full feature-by-feature and cost breakdown.
- If you are actively evaluating a switch, the migration guide walks through data parity, automated dashboard migration, and running both platforms in parallel before committing. Ready to explore the product, groundcover has a free trial with unlimited seats. All features, full BYOC, so you can run it against production data alongside your existing setup.