PEPAI Datadog Cost Check: Example Report

> **EXAMPLE DATA - fictional company. Not a real client, not real usage.**

Client: Fjordbyte Logistics (fictional) · **Report date:** 21 October 2026 · **Prepared by:** PEPAI

0. How to read this report

> **EXAMPLE DATA - fictional company. Not a real client, not real usage.**

- **Tool output:** figures produced on synthetic inputs generated by PEPAI. Metrics: dd-metric-bill 0.1.2 (crates.io). **All log figures (ingestion, ingest savings, indexed-log cost): log-bill 0.1.3 (crates.io)**, installed into a clean root and re-run on 21 Oct 2026. Marked *[tool]* or *[tool, 0.1.3]*.
- **Hand calculation:** arithmetic by PEPAI on tool output. Marked *[hand]*.
- **Illustrative:** assumptions with no tool or source behind them. Marked *[illustrative]*.
- **Prices:** Datadog public list prices, US site, "Billed Annually" column, read from https://www.datadoghq.com/pricing/list/ on 14 Oct 2026 and re-verified unchanged on 19 Oct 2026: custom metrics USD 5 per 100 per month (indexed), USD 0.10 per 100 ingested; logs ingestion USD 0.10 per GB; indexed log events per 1M by retention: 3-day USD 1.06, 7-day USD 1.27, 15-day USD 1.70, 30-day USD 2.50 (month-to-month 1.27/1.52/2.04/3.00; on-demand 1.59/1.91/2.55/3.75); Flex Logs storage USD 0.05 per 1M. This example assumes 7-day retention, annual billing. A real client's contract prices would replace these.
- All amounts are **estimates** from one sample, not invoice amounts.

1. Summary

> **EXAMPLE DATA - fictional company. Not a real client, not real usage.**

Area	Estimated monthly cost now	Main driver	Possible reduction	Confidence
Custom metrics	USD 47.80 above allotment *[tool]*	`customer_id` tag on histogram `checkout.latency`	to ~USD 1 *[tool]*	Medium
Log ingestion	USD 101.57 *[tool]*	DEBUG cache-hit lines (50.9% of bytes)	USD 71.70 *[tool]*	Medium
Log indexing (assumed index setup)	USD 8,799 *[tool, 0.1.3; index setup assumed]*	/health and order INFO lines in the index	~USD 7,552 *[tool, 0.1.3; index setup assumed]*	Low
RUM sessions	not analysed	-	-	-

Input assumptions *[illustrative]*: 12 hosts on the Pro plan (allotment 1,200 custom metrics, computed by the tool); the log sample represents **one second** of production traffic.

2. Custom metrics

> **EXAMPLE DATA - fictional company. Not a real client, not real usage.**

Command: `dd-metric-bill count --plan pro --hosts 12 --prices dd-prices.json metrics.dogstatsd`

Metric	Type	Combinations	Billable	Driver
checkout.latency	histogram (x5)	400	2,000	`customer_id`: 400 values, flagged unbounded (threshold, UUID pattern)
http.request.count	count	144	144	host 12 × endpoint 4 × status 3

Metric	Type	Combinations	Billable	Driver
queue.depth	gauge	12	12	host
Total			**2,156**	allotment 1,200, surplus 956

[tool] Estimated cost of surplus: **USD 47.80 / month**.

Finding M1: unbounded `customer_id` tag. One histogram tagged with a per-customer UUID produces 93% of billable metrics (2,000 of 2,156).

- **Recommendation:** remove `customer_id` as a metric tag (keep it in traces or logs), or index only `host,env,endpoint,status` via Metrics without Limits.
- **Effect *[tool]*:** with `--allow-tags host,env,endpoint,status` indexed drops to 216 (below the allotment, indexed cost USD 0); ingested stays 2,156, cost USD 0.956/month.
- **Confidence:** medium. The tag pattern is clear; the size depends on real traffic.
- **Caveats:** one snapshot, Datadog bills the hourly average; customer count grows over time, so the real number may be higher; the tool applies the allowlist to all metrics, Datadog sets it per metric; check whether any dashboard or monitor needs `customer_id`.

3. Logs

> **EXAMPLE DATA - fictional company. Not a real client, not real usage.**

Command: `log-bill scan --window 1s --prices log-prices.json app.log` (6,000 lines, 391,841 bytes)

Pattern (masked)	Share of bytes	Ingest USD/month *[tool]*
`<TS> DEBUG cache hit <VAR> took <VAR>`	50.9%	51.65
`<TS> INFO GET /health <NUM> from <IP>`	21.9%	22.28
`<TS> INFO order <NUM> created for user <NUM> email=<EMAIL>`	20.1%	20.41
`<TS> WARN retry upstream carrier-api attempt <NUM> token= <SECRET>`	5.9%	5.99
`<TS> ERROR payment provider timeout after <VAR>`	1.2%	1.23
Total (1,015.652 GB/month)	100%	**101.57**

Finding L1: DEBUG noise in production. Half the bytes are cache-hit debug lines.

- **Recommendation:** drop at source or with an exclusion filter.
- **Finding L2: health checks.** Sample `GET /health 200` at 10%.
- **Effect *[tool]*:** rules `drop DEBUG cache hit` and `sample 0.1 GET /health 200` save **USD 71.70 / month** in ingest (6,000 → 1,328 lines per sample).
- **Indexing (tool output from log-bill 0.1.3 on an assumed index setup):**
  - *Volume:* the tool's 1,015.652 GB/month comes from 6,000 lines per 1-second sample × 86,400 × 30 = **15,552M events/month** ingested (about 65 bytes per line). Same sample, same scale as the USD 101.57 ingest figure.
  - *Current index setup [illustrative assumption]:* an exclusion filter already keeps DEBUG out of the index; all INFO, WARN and ERROR lines are indexed. That is 2,673 of 6,000 lines (44.6%) = 6,928M events/month × USD 1.27/M (7-day retention) = **USD 8,799/month**.
  - *Proposed setup:* index all WARN and ERROR (290 lines), exclude `/health` (0), keep 10% of order INFO lines (88.9). That is 378.9 of 6,000 lines (6.3%) = 982M events/month = **USD 1,247/month**.
  - *Estimated saving:* **~USD 7,552/month** in indexing, on top of the USD 71.70 ingest saving. The ingest rules alone save nothing on indexing here, because DEBUG is assumed to be excluded already.
  - Reproduced with `log-bill scan --window 1s --retention 7 --billing annual --index-rules <rules> app.log` (log-bill 0.1.3 (crates.io), re-run 21 Oct 2026): 6,928.416M events / USD 8,799.09 with `drop DEBUG`; 982.109M events / USD 1,247.28 adding `drop GET /health` and `sample 0.1 INFO order` *[tool, 0.1.3]*. The index setup and ratios are PEPAI assumptions *[illustrative]*; a real check uses the client's own exclusion filters and retention.
- **Confidence:** medium for ingest, low for indexing (depends on which indexes and exclusion filters exist).

- **Caveats:** one-second window extrapolated to 30 days; Datadog may measure bytes after parsing; a drop simulation is not advice to drop; check whether security needs health-check logs.

Finding L3: sensitive data in logs (masking note). log-bill masked e-mail addresses in order lines (`email=<EMAIL>`) and a token value in carrier retries (`token=<SECRET>`). The report shows only masked patterns, never raw values.

- **Recommendation:** stop logging the token and the e-mail address at source; review under your privacy policy.
- **Confidence:** high that these fields appear in the sample; whether they are real secrets is not verified.

4. RUM sessions

> **EXAMPLE DATA - fictional company. Not a real client, not real usage.**

Not analysed. None of PEPAI's tools reads Datadog RUM data. (`rum-meter` counts HCP Terraform Resources Under Management and is unrelated.) No figures are given.

5. Method and reproducibility

> **EXAMPLE DATA - fictional company. Not a real client, not real usage.**

- Synthetic inputs, price files and raw tool output are kept by PEPAI and can be shared with the client.
- Tools: https://github.com/qwpp7g79y4-maker/dd-metric-bill, https://github.com/qwpp7g79y4-maker/log-bill (MIT OR Apache-2.0).
- No account access was used. No changes were made.

PEPAI · pep_ai@icloud.com · KvK 98715925 · BTW NL005350056B78