APTO SOLUTIONS

The Cardinality Standoff

Why Observability and Pipeline Teams Disagree, and What Actually Settles It

aptosolutions.co.uk
September 2026

Assessed. Architected. Operated.

Introduction

Two teams look at the same telemetry estate and reach opposite conclusions. Both of them are right.

The observability team wants high cardinality. Cardinality — the number of distinct label combinations attached to a signal — is what makes a problem diagnosable. It is the difference between knowing that checkout latency has risen and knowing that it has risen on one pod, in one region, for one customer tier, since one release. Strip it out and you are left with dashboards that report that something is wrong and nothing about where. That is not a preference. It is the difference between a fifteen-minute diagnosis and a four-hour one.

The data pipeline team wants ingest streamlined. They are measured on multiple platform cost, throughput and stability, and they can see precisely what the observability team's instincts cost. Their position is equally sound: a great deal of what arrives is never queried by anyone, and paying premium rates to keep it is difficult to defend to a CFO.

Neither team is wrong. Neither team can settle it. That is what separates this from the ordinary data sprawl conversation, and it is why the problem persists for years inside organisations that are otherwise extremely well run.

This Is Not the Data Sprawl Problem

In The Telemetry Pipeline Playbook we argue that a large share of ingested telemetry is low-value noise — health checks, debug output from production systems, duplicate events from overlapping agents — and that filtering it is the fastest cost win available to most organisations. We stand by that entirely. It describes waste: data nobody chose deliberately, nobody would defend if asked, and nobody misses once it is gone.

This paper is about the opposite case, and the distinction matters more than it first appears.

> **High-cardinality telemetry is not waste. Somebody chose it deliberately, can explain why, and will notice when it goes.**
>
> The volume is the residue of a genuine disagreement about value, not an absence of housekeeping. This is why the tactics that clear up waste do not resolve it. You cannot filter your way out of a decision that nobody has made.

The practical consequence is that the two problems need different interventions. Waste is a data engineering exercise: find it, prove nothing depends on it, remove it. The standoff is a governance exercise: make the trade-off visible, give it an owner, and decide it deliberately rather than by default. Organisations that treat the second as though it were the first end up cutting the wrong things — usually retention, usually at budget time, usually over the objections of the team that will be paged when it matters.

The Hypothesis

Everything that follows rests on a claim that can be tested in almost any estate inside a fortnight.

> **In most estates, the bulk of cardinality cost sits in dimensions nobody queries, and in environments where diagnostic value decays within days rather than months.**

If that holds, the standoff is largely false. Most of the available saving can be taken without losing any insight that anyone actually uses, and two capable teams have spent a year arguing over a trade that barely exists.

If it does not hold — if the expensive dimensions turn out to be the ones carrying real diagnostic load — then this is a genuine budget-versus-insight decision. It cannot be engineered away at the pipeline layer, and it

should not be settled by whichever team argues more persuasively in the meeting. It belongs to someone who owns both the incident risk and the budget.

Either answer is useful. What is not useful is continuing without knowing which of the two you are in, which is where most estates sit.

The Two Mandates

The standoff is structural rather than personal. Each team is behaving rationally against the objectives it has been given, and the objectives conflict.

	Observability / SRE team	Data pipeline team
Measured on	MTTD, MTTR, incident volume, SLO achievement	Ingest cost, throughput, platform stability, licence headroom
Rewarded for	Diagnosing faster and catching problems earlier	Delivering the same coverage for less
Penalised for	An incident nobody could explain from the available data	A bill that grows faster than the estate
Rational response	Instrument richly, keep the labels, resist reduction	Reduce dimensions, sample aggressively, question every source
What they cannot see	What the labels they added cost	Which labels were load-bearing during the last incident

Read the last row carefully, because it is the whole problem. Each team is missing exactly the information that would let it concede gracefully. The observability team is not ignoring cost; it has never been shown a per-label figure. The pipeline team is not indifferent to diagnosis; it has never been shown which dimensions were actually used at three in the morning. Neither is being unreasonable, and no amount of goodwill in a meeting fixes an information gap.

Cardinality Is a Different Axis From Volume

Most cost conversations, including much of our own published material, are denominated in gigabytes. Cardinality is a separate dimension, and on a significant number of platforms it is the one that drives the bill.

A time series is created for every unique combination of metric name and label values. The count is multiplicative, which is why it surprises people:

Metric	Labels	Distinct values	Running total
http_request_duration_seconds	pod	500	500
	endpoint	40	20,000
	method	5	100,000
	status	12	1,200,000
	region	3	3,600,000
Total from one metric name			up to 3,600,000

Remove the pod label alone and the same metric falls to 7,200 series — a reduction of more than 99% — while the underlying request volume, measured in bytes, does not change at all. The reverse also holds: an estate can halve its ingest in gigabytes and increase its series count at the same time. Bytes and series are correlated, but not tightly enough to manage one by watching the other.

Which axis governs cost depends on the platform:

Platform	Primary cost driver	Cardinality exposure
Splunk	Ingest volume, or workload-based pricing	Indirect — high-cardinality fields affect index and search performance
Microsoft Sentinel	Volume ingested and retained	Indirect — similar to above
Grafana Cloud / Mimir / Prometheus	Active series for metrics	Direct — series count is the unit
Datadog	Hosts, plus custom metrics counted as unique tag combinations	Direct for custom metrics
Elastic	Resource-based (compute and storage)	Indirect — mapping explosion and shard pressure

The Aggregation Trap

Our own Playbook contains an example worth revisiting in this light. It describes collapsing a million individual 200 OK web server log lines into one metric: a million successful requests to a named endpoint in five minutes, with a p99 latency figure. As written, that is a single series and an unambiguous win.

The technique becomes hazardous at exactly the point it becomes useful. A summary is only actionable if it carries the dimensions people query, so the natural next step is to break it down per route, per pod, per status code, per region. Each dimension multiplies the series count while the byte reduction stays real. On a volume-priced platform this remains a straight win. On a series-priced platform it can be a loss dressed as a win — and the reporting will show a satisfying fall in gigabytes while the invoice climbs.

> **Aggregate on the dimensions people actually query, and drop the ones they do not.**
>
> If nobody has grouped by pod on that metric in ninety days, pod does not belong in the summary. This is an empirical question with an available answer, not a matter of judgement — but somebody has to look.

Why Non-Production Is Where It Bites

Cardinality pressure concentrates in development and test environments, for reasons that have nothing to do with carelessness.

- **Identifiers churn constantly.** Ephemeral namespaces, per-pull-request environments and short-lived test clusters generate fresh pod, container and instance identifiers continuously. Where billing counts a series as active for a window after its last write, that churn accumulates into a series count that can rival or exceed production.
- **Label discipline is loosest where debugging is heaviest.** Engineers add labels while chasing a problem, which is exactly what labels are for. Removing them afterwards is nobody's ticket.
- **Synthetic load produces genuine series.** Load tests and generated traffic create the same label combinations as real users, at volumes chosen for stress rather than realism.

The obvious response is to cut non-production monitoring back, and it is the wrong one. Non-production is where a cardinality problem should be caught. If development and test are instrumented more thinly than production, the first time anyone sees a label explosion is when it reaches production — which is the most expensive possible place to discover it, and the least convenient.

> **The answer is not less non-production observability. It is differently governed non-production observability.**
>
> Shorter retention, coarser resolution, a tighter label allow-list and an explicit series budget — while keeping full fidelity for a short window, because that window is where problems are cheapest to find.

Why Neither Team Can Settle It

Escalation is the obvious route, and it usually fails for a specific reason.

The observability team cannot simply accept the cost mandate. Doing so means shipping something they know to be less diagnosable, and they are the team that carries the incident when it proves so. The pipeline team cannot simply accept the cardinality mandate, because doing so means missing a target they are measured against, in service of a benefit that never appears in their own numbers.

So it goes upward — to someone who owns both budgets and has no basis on which to arbitrate, because the one artefact that would settle it does not exist: a list of series with an owner, a cost, and a record of when each was last queried. In its absence the discussion becomes a contest of assertions, and the person chairing it reasonably declines to overrule either specialist.

The decision is therefore not made. The default holds, which is to keep everything. Cost accretes quietly for another few quarters until it becomes conspicuous enough to force action, at which point the lever available is a blunt one — usually retention, usually applied at budget time, usually landing on the observability team, and usually removing the oldest evidence rather than the least useful.

What Actually Resolves It

Five controls, in the order we would introduce them. None requires either team to concede the principle.

1. A series budget per environment

Allocate the series allowance by environment rather than by platform: production, staging and development each get a number. This converts an argument about principle into a decision about allocation, made locally by the people who understand the trade-off, and it stops non-production growth being invisible until the invoice arrives.

2. Label allow-lists enforced at the pipeline

Governing labels in application code depends on the discipline of every team that ships, indefinitely. It drifts, and it drifts silently. A pipeline rule does not. Enforcement in Tier 2 also means the policy is one place to read and one place to change, rather than distributed across every repository.

3. Different rules for non-production

Shorter retention, coarser scrape resolution and a tighter allow-list, with full fidelity preserved for a short window. Most non-production diagnosis happens within hours of the event; almost none of it happens after a fortnight.

4. Shared series-by-team visibility

A single view showing series count, owning team, and when each series was last queried, visible to both sides. Most of the argument dissolves once both teams are looking at the same number — the observability team will readily drop dimensions nobody has touched in three months, and the pipeline team stops proposing cuts to the ones that carried the last major incident.

5. A named arbiter and a standing review

Somebody has to own the trade-off, and it should not be either team. Monthly is sufficient. The role is not to referee a dispute but to make a decision that neither party is structurally able to make alone.

A Worked Model

> **This is a model, not a customer.**
>
> The figures are illustrative and deliberately round, so the arithmetic can be followed and your own numbers substituted. Its purpose is to show what a visible trade-off looks like — because the argument of this paper is that invisibility, rather than cost, is the actual problem.

Take an estate carrying four million active series on a series-priced platform:

Environment	Active series	Share of total
Production	1.6m	40%
Staging	0.9m	22%
Development	1.5m	38%
Total	4.0m	100%

Non-production accounts for 60% of series here, and 60% of the cost that follows from them. Whether that is true of your estate is precisely the question worth measuring first.

Four options. Cost is expressed as a change against current metrics spend rather than in pounds, so the model survives a change in vendor pricing:

Option	Change	Series removed	Cost	Insight given up
A	Do nothing	—	0%	Nothing now, and the series count keeps growing. The decision simply passes to next year's budget round.
B	Remove the pod label from non-production metrics	1.05m	−26%	Cannot attribute a non-production anomaly to a specific pod without re-running the test.
C	Remove the pod label everywhere	2.20m	−55%	Cannot isolate a single failing pod during a production incident.
D	Non-production allow-list, 7-day retention, 60-second resolution. Production untouched.	1.55m	−39%	Non-production comparison beyond a week, and sub-minute non-production resolution.

Option C is the one that looks best on a spreadsheet and the one to refuse. It buys the largest saving by removing exactly the dimension that matters most on the worst day of the year, and the cost of that choice does not surface until an incident — by which point almost nobody connects the two events.

Option D delivers around two-thirds of C's saving and gives up nothing that anyone queries after the first week of a test cycle. It is also the option that cannot be chosen without evidence. Absent the knowledge that non-production carries 60% of series and that nobody queries it after seven days, D is indistinguishable from B and C as a matter of opinion — which is why the argument never resolves.

The Trade Nobody Should Make by Accident

Strip away the architecture and this reduces to one sentence: you are buying insight with budget, at an exchange rate nobody has calculated.

Every organisation running a monitored estate makes this trade. Most make it twice, and implicitly on both occasions — first by default, when everything is kept because keeping it requires no decision, and then abruptly, when a retention cut arrives in a budget round. Neither is a decision. The first is an accumulation and the second is a reaction.

The alternative is not to spend less. It is to know the rate. An organisation that can say it deliberately gave up per-pod attribution in staging in order to fund full-fidelity tracing in production has made a strategic choice, and can defend it in either direction — to the CFO and to the engineer on call. An organisation whose staging cardinality grew quietly until somebody cut retention across the board has made no choice at all, and will discover which insight it needed during the next incident rather than before it.

That is the whole argument. Both teams are right, the trade is real, and the only genuinely wrong answer is to make it by accident.

The Operating Model

Everything above describes a decision. This section describes the machinery that makes the decision possible, because a series budget is a wish until something meters it, and an allow-list is a policy until something enforces it.

The obstacle is that cost accumulates in two places, in two different units, usually owned by two different teams.

Layer	Unit of cost	Attributable to	Answers	Cannot answer
Pipeline — Cribl Stream	Credits consumed	Worker group, pipeline, route	What it costs to collect, process, transform and move this data	Whether anyone ever reads it
Platform — Grafana and equivalents	Active series	Tenant, team, label set	What it costs to keep this data queryable	What was spent getting it there
Both, operated together	Cost per owned outcome	Team, service, environment	The exchange rate between budget and insight	—

Neither layer answers the question alone. The pipeline knows what processing cost and nothing about value; the platform knows what retention cost and nothing about the journey. The trade-off this paper describes lives across both, which is why it is invisible to anyone holding only one of them.

Attribution is the mechanism

Operated together, the two layers produce a chain that can be followed end to end: worker group to pipeline to destination to series to owning team. Once that chain exists, the five controls stop being aspirations. A series budget per environment becomes an allocation someone can be held to, because the same operator who sets it also holds the valve that enforces it. A label allow-list becomes a route configuration rather than a request to twenty engineering teams. The monthly review has a number to review.

Credit allocation to worker groups is the practical expression of this. Align worker groups to ownership boundaries — team, environment, or business unit — and consumption becomes attributable rather than aggregate. Most organisations start with showback, because visibility alone changes behaviour and costs nothing politically. Chargeback is available afterwards for those who want it, but it is rarely necessary; teams that can see their own number generally act on it.

What this fixes

The gap identified at the start of this paper was informational. The observability team had never been shown what its labels cost; the pipeline team had never been shown which dimensions carried diagnostic load. Attribution across both layers closes that gap directly, and collapses a feedback loop that currently runs in months into one that runs in weeks. An instrumentation decision taken in one sprint shows up as a number the team owns in the next, rather than as a line in a budget conversation two quarters later involving neither of them.

> **Two honest caveats.**
>
> Attribution is only as good as the boundaries it follows. Where worker groups have been organised by geography, environment or historical accident rather than ownership, the resulting numbers attribute cost to nobody in particular. Reorganising them around ownership is frequently the first real piece of work, and it is not trivial.
>
> Showback creates its own failure mode. A team measured on its consumption can reduce the number by moving data rather than by needing less of it. This is why the never-queried measure in the next section belongs alongside the cost figures rather than after them.

Measuring the Right Thing

Cost per gigabyte remains a valid measure and should be retained. It is simply insufficient on its own, and an estate managed exclusively by that metric can report progress while the bill increases.

Metric	What it measures	Why it matters
Active series by environment	Cardinality load, split prod / staging / dev	The number that governs cost on series-priced platforms
Series growth rate	Month-on-month change	Distinguishes estate growth from label sprawl
Series churn rate	Proportion of series short-lived	High churn points at ephemeral identifiers as labels
Series never queried in 90 days	Proportion carrying no analytical load	The empirical basis for reduction, in place of assertion
Cost per active series	Efficiency on the cardinality axis	Complements cost per GB; neither is sufficient alone
Cost per GB ingested	Efficiency on the volume axis	Retained from the existing framework

Where Apto Sits

We are able to arbitrate this because we run the layer and have no stake in either team's targets. We are not measured on your MTTR and we are not measured on your ingest bill, which means we can put both numbers on the same page and say what the trade-off actually is.

This work sits in the Operate phase of our Assess, Build, Operate lifecycle, and it is the reason we operate the pipeline and the platform as one engagement rather than two. The attribution described in the previous section only exists if somebody is running both layers continuously and reconciling them. It is not a project with a completion date: series counts drift, teams ship new labels, environments multiply, and a policy set once will be wrong within two quarters.

Apto's Operate model provides continuous pipeline operations, platform management and cost governance across both layers, so the trade-off is metered and reviewed by someone whose job it is, rather than deferred indefinitely by two teams who each have something more urgent to do.

Next Steps

Apto Solutions offers a range of entry-point engagements designed to give you clarity before commitment:

- **Free Assessment:** A no-obligation conversation with one of our platform specialists to understand your current state and identify quick wins.
- **Observability Maturity Assessment:** A framework-driven evaluation of your monitoring and observability capabilities against industry best practice.
- **Data Mapping and Discovery:** An analysis of your telemetry data flows, identifying redundancy, gaps, and optimisation opportunities.
- **SIEM Health Check:** A structured review of your existing SIEM deployment covering architecture, detection coverage, data quality, and operational efficiency.

Book your free assessment at aptosolutions.co.uk/contact or email enquiries@aptosolutions.co.uk to start the conversation.

Assessed. Architected. Operated.