Pushing Intelligence to the Edge

How making smart filtering decisions early in the pipeline decluttered our dashboards and decreased costs

Hello O11y Summit!
I'm Alex Degitz

- VP of Product at ElastiFlow
- Based in Denver, Colorado
- Not a LinkedIn guy
- Find me in CNCF Slack

ElastiFlow and OTel

- Mermin = Rust-based open-source eBPF agent for network flow traces via OTLP
- Bringing deep network insights into OTel
- Contributing to network-related semantic conventions
- Our SRE team is running several context enrichment services
  - Dive into our OTel story

The Open Telemetry High and the Cost Hangover

The High

eBPF auto-instrumentation (OBI) makes signal collection absurdly easy.

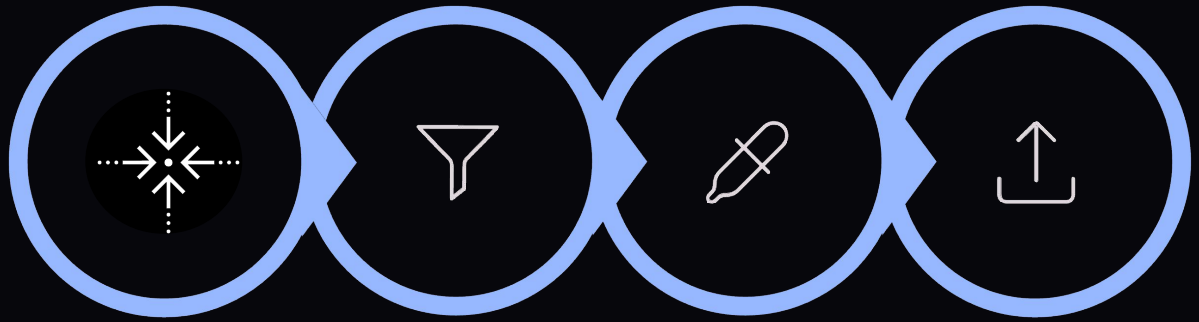
- Vendor-agnostic bliss.
- No proprietary agent.

The Hangover

OTel gives freedom, but zero hand-holding on cost optimization.
You're on your own.
Many O11y platforms charge a premium for OTel data.

OTel Collector features and issues we ran into

First attempt at taming the firehose — filter, sample, aggregate, and deduplicate *before* signals hit the observability platform.



These mechanisms are necessary — but often not sufficient. Good first step. Not the final answer.

Filter rules: Between a Rock and a Hard Place

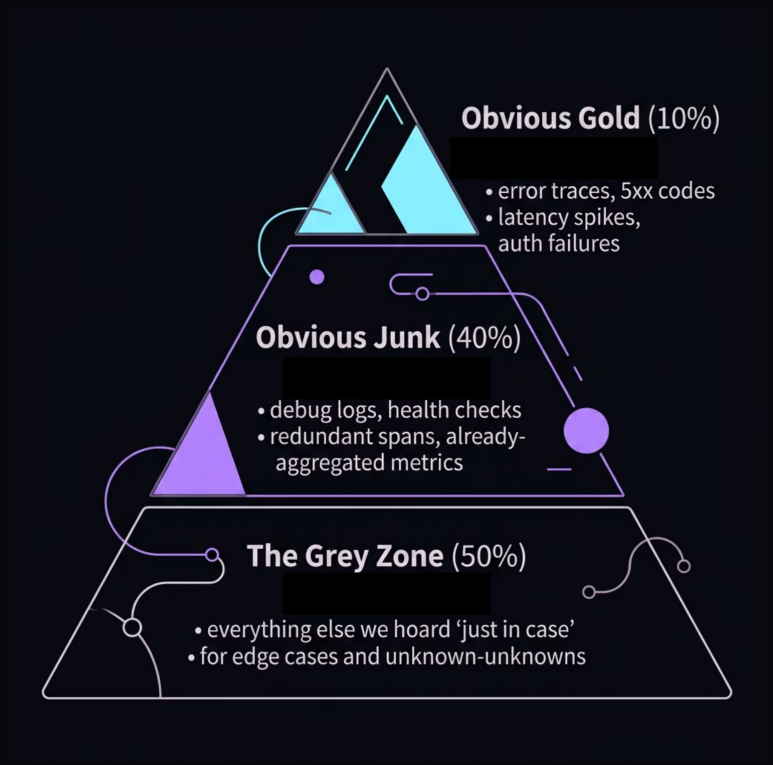
Drop more data and stay within budget OR

keep the data your team believes is needed.

If an outage hits with missing traces,
...*you* take the blame.

⚠ The fear of dropping data is not irrational.
It's career-preserving instinct.

The Grey Area of Collected Data



We store the grey zone **just in case.**
Unknown edge cases. Future RCA we can't predict.
That 50% is where the savings potential is located.

The Solution: Edge Intelligence

We already created a Rust-based correlation engine for network flow and app traces.
Use AI to make smarter calls as close to the source as possible.

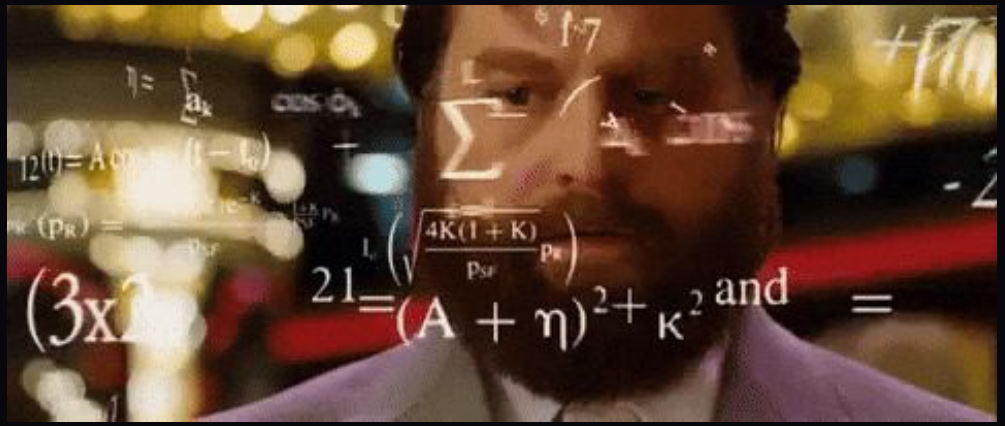
Detect Anomalies
ML models flag unusual patterns in real-time

Decide at the Edge
Keep/drop logic runs on the collector, not the backend

Call a Lifeline
Request additional data if required for troubleshooting

Step 1

Intelligent Sampling



Move past basic head/tail sampling.

Don't just flag latency or 500s.

Use anomaly detection to decide if a trace should be kept based on traffic exceeding a learned threshold.

- Baseline normal traffic patterns per service
- Sample aggressively when behavior is normal
- Escalate sample rate when anomaly score spikes
- Protect from cost and scale spikes

Step 2

The What If: Route & rehydrate or store locally

You need to store less than you think. An MCP server gives your AI SRE a chance to pull data back when more info is required.

$150

SSD Cost

Buy 1TB once.
Own it for years.

$21,600

DataDog / year

Rent 1TB of indexed log storage.
Every month.

~140x

Cost Difference

Local storage is
basically free.

Step 3

Let the Platform Teach the Pipeline

The observability backend knows what data actually gets used for RCA.

Feed that signal back into the pipeline as suggested sampling/filtering rules.

> **Example rule:** Increase sample rate for this trace type when X condition is met — this data hasn't been used for RCA in 90 days.

The platform becomes self-tuning.
Less guesswork.
More evidence-based dropping.

Let’s see it in action

Drop Aggressively. Sleep Soundly.

OTel gives flexibility, ownership, and power.
With great power comes great responsibility.
Responsibility requires Intelligence.

Smart Edge

Anomaly-driven sampling decisions near the source

Leverage your agents

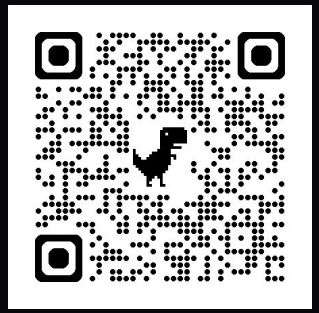
SSD-backed safety net for on-demand RCA retrieval

Feedback Loop

Platform teaches the pipeline what to keep

Thanks for listening!

- Check out Mermin on GitHub
- Join us in the CNCF Slack
  - send me a DM
  - join the #mermin channel



Mermin