DATABAHN

WHITEPAPER

The Real Challenge of AI Observability

Why Watching AI Isn't the Same as Governing It.

AUTHOR

Preston Wood

Chief Security and Strategy Officer

Table of Contents

1 Introduction

Every organization now has an AI usage problem, and most of them don't know the shape of it. Not because the AI tooling is difficult to implement — a browser tab and a corporate credit card are enough to start — but because AI adoption happened faster than the ability to watch it. Employees are pasting source code into chat assistants, analysts are wiring provider APIs into pipelines, and a new class of autonomous agents is making dozens of model calls per task with no human in the loop. The activity is real, the spend is real, and the risk is real. The visibility, for most teams, is not.

The instinct is to treat this as a monitoring gap — bolt on a dashboard and move on. But AI observability is harder than traditional observability, because the thing you're trying to see is distributed across surfaces that were never designed to report to a single place. Getting it right means being honest about what each surface can and can't tell you, and about the difference between watching AI and being able to do something about what you see.

2

What "The Right Telemetry" Means

When leaders say they want visibility into AI, they usually mean five distinct questions, each answered by different data:

- **Token Usage** - how much is actually being consumed, by whom, against which models. This is the raw unit of AI work and the basis for almost everything downstream.
- **Cost** - what that consumption translates to in dollars, attributable to a team, workflow, or individual, not just a single blended invoice at month's end.
- **User Activity** - who is using AI, for what, and whether sensitive data is moving into prompts. This is where governance and insider-risk questions live.
- **API Usage** - programmatic calls made by services and integrations, which often dwarf interactive chat usage and are the easiest to forget about.
- **Agent Usage** - the fastest-growing and least understood category. Agent runtimes (Cowork, OpenClaw, coding agents, and autonomous orchestrations) don't make one call and stop; they plan, call tools, retry, and chain steps. A single "task" can be a whole tree of model invocations, and the interesting failures happen in the middle of that tree.

The trouble is that no single data source answers all five cleanly. The signals are scattered, and the completeness of your answer depends entirely on where you stand to collect them.

3

Where AI Usage Signals Live

Usage telemetry surfaces in more places than most teams expect:

- **Provider and Vendor Analytics** - the usage, seat, and cost data an AI vendor exposes through its own APIs. And most provider APIs are limited, and difficult to pull and understand.
- **Chat Activity** - interactive sessions, including the prompts and content that carry the real governance risk.
- **Agent Execution** - multi-step runs that need to be reconstructed to be understood at all.
- **Endpoint Logs** - what's happening on the machines where AI clients and agents actually run.
- **Network and DNS Logs** - queries resolving to AI-related domains, which reveal usage no one told you about.
- **Web Proxy Logs** - outbound traffic to AI services, including the tools that never went through procurement.

Each of these is a partial view. Endpoint and network logs are great at telling you that AI is being used and by which host, but say almost nothing about tokens, cost, or what was in the prompt. Vendor analytics tell you spend and seat utilization but only for the vendors you've formally onboarded — they're blind to everything else. Chat and agent telemetry are the richest source of content and behavior, but only if something is positioned to capture them as they happen. Which brings us to the distinction that matters most.

4

Passive vs. Active Observability

There are two fundamentally different ways to observe AI usage, and conflating them is where most observability strategies go wrong.

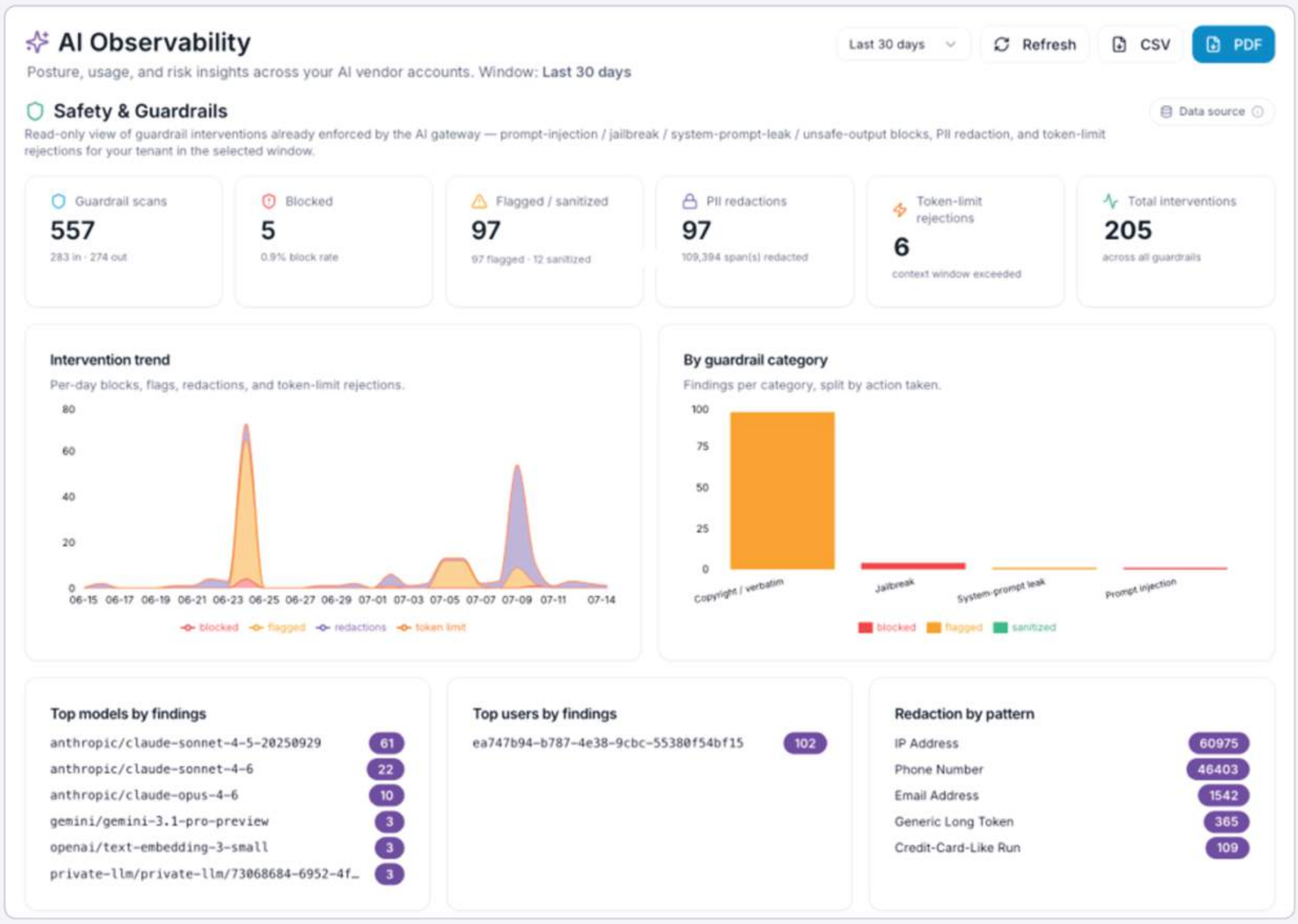
Passive observability is collecting data after the event has occurred. You pull a provider's usage API on a schedule, you parse yesterday's proxy logs, you reconcile last week's spend. It's essential, but it will never let you intervene — by the time you see the event, it's already happened. Passive observability is a rear-view mirror: excellent for accounting, auditing, trend analysis, and discovering shadow usage, but structurally incapable of prevention.

Active observability means sitting in the request path with the ability to apply logic and take action before the call completes. When AI traffic flows through a control point, you can do things that are simply impossible after the fact: block a request, redact sensitive data out of a prompt before it reaches a provider, enforce a quota, deny an unapproved model, cap the cost of a single runaway agent, or catch a prompt-injection attempt in flight. Active observability is a gate, not a mirror — and crucially, it also produces telemetry that no passive source can reconstruct, because the interesting data (queue wait, retry counts, why a request was blocked, which guardrail fired) only exists at the moment of the call.

This is the key insight: **some telemetry cannot be captured without running the calls through an LLM gateway, and any real management or blocking requires one.** You cannot redact a prompt from a log file. You cannot block a request you're reading about tomorrow.

5

Gateway Visibility and Enforcement



A read-only safety view: 205 guardrail interventions over the window — 5 blocked, 97 flagged, and 109,394 PII spans redacted — split by guardrail category, top models, and redaction pattern.

An LLM gateway is the control plane that every AI call passes through on its way to a provider. Because it's in the path, it becomes both the richest telemetry source and the only enforcement point.

On the visibility side, a gateway records the things that don't exist anywhere else: pre-flight token estimates reconciled against actual usage, per-call cost attributed to a specific workflow, node, and user, the split between production and sandbox traffic, and which credential tier paid for the call. It captures not just successes but outcomes — whether a request succeeded, errored, was blocked, hit a rate limit, exhausted a quota, or timed out — along with retry counts, cache hits, and time spent waiting in the queue. It records provider health transitions and circuit-breaker events so a degrading provider is visible as it happens rather than inferred from a spike in failures.

On the enforcement side, being in the path is what makes governance real rather than advisory. A gateway can enforce per-tenant monthly token quotas, apply sliding-window rate limits it discovers from provider headers, and fair-share a priority queue so one team's batch job doesn't starve another's interactive work. It can redact PII and secrets — emails, API keys, tokens — out of outbound prompts before they ever leave the building so to speak. It can run guardrails for prompt injection and jailbreak attempts, restrict which providers and models a tenant is even allowed to call, and defend against "denial-of-wallet" attacks with per-request cost ceilings and a cost-spike breaker.

For agents specifically, it can cap iterations and tool fan-out so an autonomous run can't spiral. Every one of these actions is also a telemetry event — enforcement and observability are the same act.

Because every intervention is logged, a downstream reporting surface can present them read-only — as in the safety view below, where blocks, flags, PII redactions, and token-limit rejections are broken out by guardrail category, model, and user.

6

Consolidating the Passive Layer

Everything the gateway can't see — the shadow AI, the vendor-side record, the infrastructure signals — is passive by nature. But the hard part of passive observability isn't collecting any one source; it's that the sources are scattered, differently shaped, and never correlated. DNS logs live in one system, proxy logs in another, EDR on the endpoints, and each vendor's usage data behind its own API in its own format. Individually they can look like noise. The value only appears when they're pulled together, normalized, and analyzed as one — and that consolidation is exactly what a data pipeline exists to do.

This is where the DataBahn Agentic Data Control Plane becomes the backbone of the passive layer. Rather than standing up a separate integration for every log source and every vendor API, you point connectors at each source and let the pipeline do the collecting, normalizing, enriching, and routing on a schedule. Two things flow through it:

- **Infrastructure and log telemetry** — DNS queries resolving to AI domains, web proxy records of outbound AI traffic, and EDR logs from the endpoints where AI clients and agents actually run. This is the discovery layer that surfaces the shadow AI no gateway ever saw and no vendor ever billed. A significant amount of AI activity is already sitting in logs organizations collect but never correlate against AI usage; the pipeline is what turns that dormant data into a usage signal.
- **Provider and vendor API data** — usage, seat utilization, cost, and (for vendors that expose it) the chat and audit timeline needed for insider-risk and data-exfiltration analysis. Piping this through the same pipeline means vendor telemetry lands in the same normalized store as everything else, ready for the same reporting and analysis instead of trapped in a per-vendor console.

Once both streams run through one pipeline, the payoff is correlation. A spike in proxy traffic to an AI domain can be lined up against vendor-reported spend and gateway telemetry to tell a single story instead of three partial ones.

PII can be redacted on the way through, findings can be ranked by severity, and the whole thing can run unattended on a schedule with cost controls — so consolidation is continuous rather than a quarterly fire drill. The pipeline is also where the passive layer gains an enforcement edge: the same network and proxy signals it consolidates can feed controls that block traffic to unsanctioned AI services, turning after-the-fact discovery into a policy that acts.

7

Token Usage and Cost in One View

Of the five questions, cost is the one leaders ask first and the one most often answered wrong — because "cost" is really several different numbers, and the relationship between them is not intuitive.

Start with the unit. Tokens are the raw measure of AI work, but tokens are not dollars, and raw tokens are not even billable tokens. Modern providers cache large, repeated chunks of context, and a cache read is billed at a fraction of a fresh token — on the order of a tenth of the input rate. So the headline token count can be dominated by cheap cache reads while the actual bill is driven by the much smaller volume of fresh input, output, and cache-creation tokens. Read cost off raw token volume and you will be wrong, often by an order of magnitude.

This is why a usage view has to separate raw activity from billable activity. In the trend below, billable tokens are defined explicitly as input + output + cache-create, with cache reads excluded from the billable total precisely because they bill at roughly ten percent of input — the composition, not just the count, is what maps to spend.

Billable spend and billable tokens over the window. Billable tokens count input, output, and cache-create; cache reads are excluded because they bill at roughly a tenth of the input rate — which is how a huge raw-token figure can still map to a modest bill.

Caching also behaves differently depending on how the AI is used, which matters for who has to do anything about it. For interactive chat and console usage, several providers can turn caching on automatically — it's a setting in the provider console, and repeated system prompts or context get cached without anyone writing code. For direct API usage, caching is generally deliberate: developers have to mark cache breakpoints in their calls, and if they don't, every repeated block of context is billed again at full price. The same workload can cost dramatically more or less depending on whether that one implementation detail was handled.

On top of the caching mechanics, cost splits by where the usage originates — and each origin is billed and measured differently:

- **Chat and agent** use (Chat, Cowork, and the like) — interactive, human- or agent-paced, bursty, and the hardest to attribute to a budget line.

- Direct API use billed straight against provider accounts — often the largest share of spend, the easiest to overlook, and the place where a missing cache breakpoint quietly doubles a bill.

- Gateway-metered calls — the slice routed through a control point, where per-call cost, cache hits, latency, and errors are captured precisely.

Pull these together per provider and the token-versus-cost divergence becomes concrete. In the breakdown below, one Anthropic source shows 10.28B tokens for about $8,064 while a second, direct-API Anthropic source shows only 1.08B tokens for a nearly identical $7,269 — roughly a tenth of the tokens at almost the same cost, because the large figure is inflated by cheap cache reads while the direct-API traffic is closer to full price. A per-provider view with gateway-versus-direct badges is the only way to see that two lines with wildly different token counts can carry the same bill.

Where spend comes from

Each provider's calls, tokens, cost, and share of total spend — gateway + direct combined, with a badge showing which side each row came from.

Provider	Source	Calls	Tokens	Cost	Spend %
Anthropic	GW Direct	882	10.28B	$8063.88	52.6%
Anthropic_api	Direct	—	1.08B	$7269.06	47.4%
Gemini	GW	8	1.74M	$2.71	0.0%
Openai	Direct	—	90.24M	$0.0000	0.0%
Private-Llm	GW	33	2.43M	$0.0000	0.0%

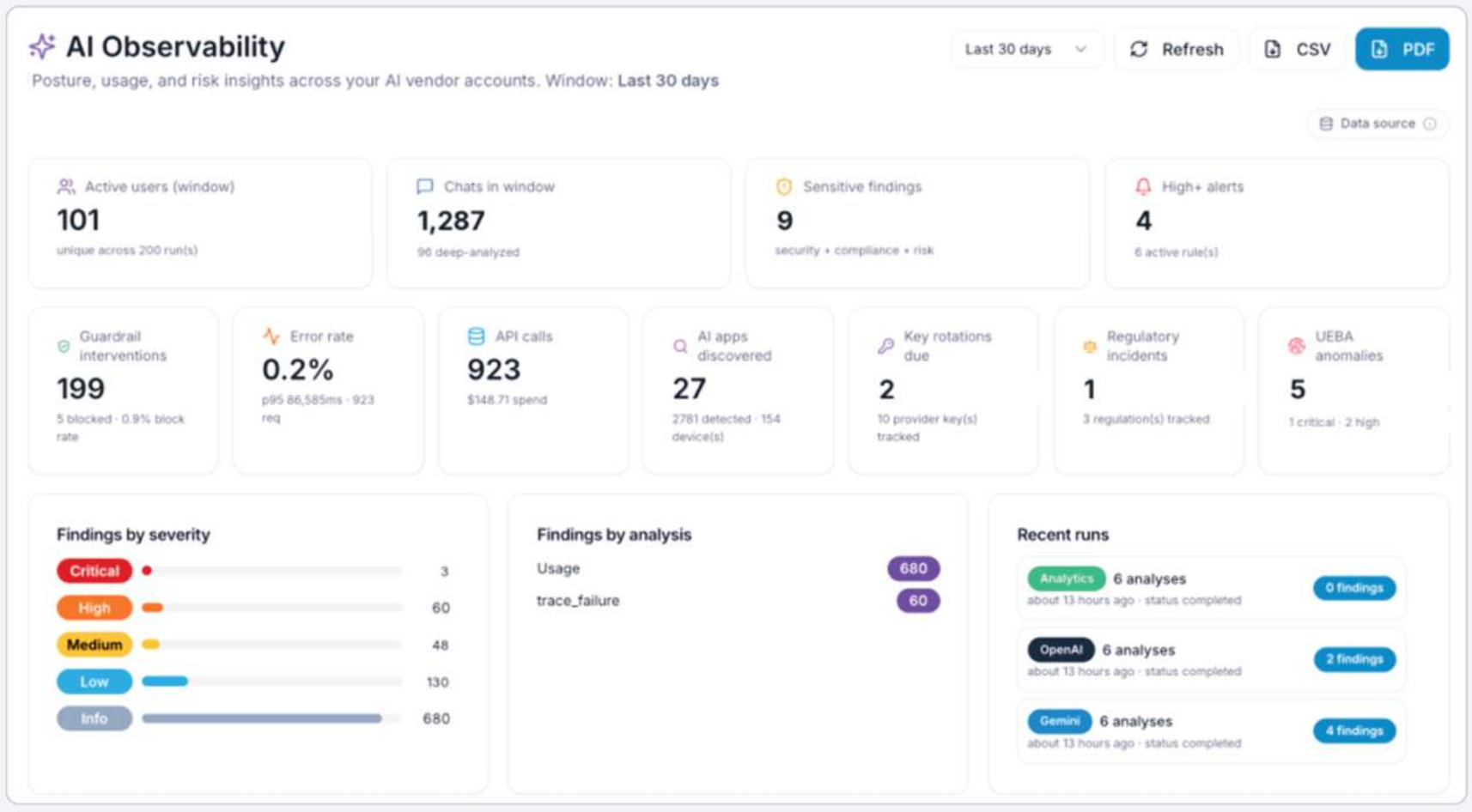
Where spend comes from, per provider, gateway and direct combined: near-identical cost against a 10x difference in tokens is the caching story made visible — raw volume and real spend are not the same axis.

There's a second point hiding in that same table: most of the tokens and spend sit on direct rows, not gateway-routed ones. Whatever you route through a control point, a large share of real usage and cost may be flowing around it — which is exactly why the billable picture has to be consolidated from both the metered path and the direct provider accounts before any total means anything.

8

Bringing the Two Layers Together

Full visibility is two complementary layers, not a single tool. Brought together, the passive layer can present a single posture view across every connected vendor account — active users, findings by severity, discovered AI apps, and per-provider analysis runs in one place.



Where spend comes from, per provider, gateway and direct combined: near-identical cost against a 10x difference in tokens is the caching story made visible — raw volume and real spend are not the same axis.

An LLM gateway is the active layer. For every AI call you route through it, it answers token, cost, user, API, and agent-usage questions with precision, and it's the only place you can actually intervene in flight. Its limitation is scope — it only sees what flows through it.

The DataBahn Agentic Data Control Plane is the passive layer. It consolidates everything the gateway doesn't see — infrastructure logs and vendor API data alike — into one normalized, analyzable, continuously refreshed store, and feeds the reporting and analysis that turn scattered signals into governable findings. Its limitation is timing — it tells you what already happened.

Together they close the loop: the gateway governs and instruments the sanctioned path, and the pipeline discovers, correlates, and reports on everything else.

9 The Takeaway

AI observability isn't a dashboard you buy; it's a posture you build from two complementary layers. An LLM gateway gives you active control and the deepest telemetry for the sanctioned path. The DataBahn Agentic Data Control Plane consolidates the passive world — the vendor APIs and the infrastructure logs — into one normalized store you can actually report and analyze against, and surfaces the shadow AI that never came near your gateway. Lean only on passive collection and you'll have partial accounting with no ability to change anything. Lean only on the gateway and you'll be blind to everything that routed around it. Put a control point in the path and a pipeline behind the logs, and the blind spots close.

The organizations getting this right aren't the ones with the most data. They're the ones who understand which layer answers which question — and who put a control point in the path before the questions become incidents.

DATABAHN

DataBahn provides an AI-powered data pipeline and fabric platform that helps organizations securely collect, enrich, orchestrate and optimize enterprise data—including security, application, observability and IoT/OT data—for analytics and AI. The platform streamlines data workflows, improves data quality and enables real-time insights, helping businesses make faster, smarter decisions. By replacing fragmented toolchains with a unified, intuitive solution that requires no specialist training, DataBahn reduces complexity and cost while accelerating time to value for numerous Fortune 500 enterprises.

Learn more at www.databahn.ai