cloudaware

Independently Successful FinOps

Four problems FinOps practitioners are navigating right now and how to solve them.

CLOUDAWARE FIELD GUIDE

2026

In this guide

> *FinOps practitioners should be independently successful — completing the work without anyone's cooperation, and advising the business rather than reporting to it.*

01

Cost Allocation

Tagging never closes the gap on its own. Every dollar on the estate can be attributed today, from the signals your infrastructure already emits, without asking engineering for anything.

02

Unit Economics

Cost per customer is the wrong metric. Profit per customer is the number that connects cloud spend to business value.

03

Exposure Management

The goal of FinOps is cloud cost control. A team that has eliminated every identifiable waste but cannot answer "are we in control?" has not reached the goal.

04

FinOps for AI

AI cost forecasting isn't broken because AI is new. You just can't forecast non-deterministic cost as a single number.

cloudaware

01

Cost Allocation

Tagging never closes the gap on its own. Every dollar on the estate can be attributed today, from the signals your infrastructure already emits, without asking engineering for anything.

The slide goes up and the room changes. It always changes at the same moment, when the unallocated number appears. Not 30 percent in this case. Not the horror-show figure from an estate that never tried. This team has been running a tagging program for eight months. They have a coverage dashboard. They have Slack alerts for compliance failures. They have a dedicated engineer whose primary job, for two quarters running, has been tag remediation. The slide says 76 percent allocated, and the CFO looks at the remaining 24 and says, very quietly: "That's more than we spend on the payments platform. What's in it?"

She isn't wrong. The unallocated bucket in this case is $4.8 million annually, roughly the entire cloud footprint of one of the company's three main product lines. The FinOps lead knows this. She also knows that she cannot tell the CFO what's in it. Not because the data doesn't exist somewhere (it does, scattered across fourteen accounts and six deployment environments) but because the tagging program never reached it. Legacy workloads that predated the initiative. Kubernetes clusters where cost attribution doesn't survive the billing aggregation. Shared networking infrastructure that technically belongs to everyone and therefore belongs to no one. Resources created by CI/CD pipelines with tag templates from a version the repo deprecated fourteen months ago.

The honest answer is embarrassing. And the meeting pivots, as it always does, from what FinOps knows to what it doesn't.

That 24 percent is not a gap the next tagging cycle will close. The answer to the CFO's question (what does this application cost?) doesn't sit at the end of the program's roadmap. It already exists on the estate, in the signals the infrastructure itself emits. Treating the tagging roadmap as the only route to that answer means building toward a ceiling the program cannot raise.

The 2024 State of Cloud Cost Report surveyed a thousand engineering and finance professionals at organizations with at least $500K in annual cloud spend. The result: 87 percent of organizations use tagging as their primary allocation method. Among those organizations actively running tagging programs, not companies that gave up,

the median allocation rate is 75 percent. One in four cloud dollars, across the full sample of companies doing their best on this problem, lands in the unallocated bucket by default.

The 25 percent gap persists because a portion of cloud infrastructure cannot be tagged at its source, regardless of engineering discipline. Support fees. NAT gateway transfers. Kubernetes shared control plane costs. Cross-region data transfer charges. These billing line items arrive in the cost data without the context needed to attach them to a tag, because the infrastructure layer that generates them is shared, serving multiple workloads simultaneously, and the tags applied to those workloads don't propagate down. The FinOps Foundation, which sets the benchmarks this industry measures itself against, acknowledges the limit in its own guidance: the initial compliance target is greater than 90 percent, not 100 percent, precisely because 100 percent is architecturally unreachable with tags as the primary mechanism. The ceiling exists in the framework documentation but doesn't come up in budget reviews.

> You cannot enforce your way past an incentive problem.

Tag coverage erodes without active enforcement. Industry estimates from practitioner guides put the drift rate at 8 to 12 percent of compliance per quarter. New resources get provisioned by pipelines that inherit stale templates. Engineers spin up environments for testing without completing required fields. Acquisitions arrive with incompatible taxonomies. The organization that reaches 85 percent coverage in January and allows the tagging initiative to settle into the background, because the team has other priorities and the program seemed to be working, is sitting at somewhere between 60 and 70 percent by Q4. The tide runs one direction on its own.

Riley Jenkins, a Principal SRE and FinOps Architect at Domo, described this in the FinOps Foundation's working group documentation on engineering engagement: "After a few rounds of mass tagging we weren't able to sustain a good coverage percentage of our spend by a 'you must tag' sort of policy. This led to large holes in

our data." Domo eventually got to 99 percent coverage, but through rebuilding the organizational incentive structure around why engineers would care about tags, not through better enforcement. That kind of change requires the authority to change how engineering teams are measured, authority that a FinOps function, in most organizations, does not have.

Jan Karstens, then CTO of Blue Yonder, described the structural reason that enforcement doesn't solve the problem. In the same FinOps Foundation documentation: "Product engineering has always been run as a cost center measured on new features. Thus, to product engineering the features requested by [the hosting team] were not attractive as they did not deliver incremental value to end users." The engineers who didn't tag were acting rationally. The organizational structure they work inside gives them no material reason to spend time on cost metadata instead of shipping features. You cannot enforce your way past an incentive problem. You can only make the enforcement increasingly expensive and increasingly resented until it collapses.

Forty percent of FinOps practitioners list "getting engineers to take action" as their top challenge, the single largest category in the State of FinOps 2025 survey, representing 861 practitioners accounting for $69 billion in cloud spend. Tagging programs fail repeatedly not because they are badly designed but because they require a collaboration that the organizational structure doesn't reward and a FinOps team that typically lacks the authority to change that structure. The dependency on that collaboration is the problem. A better tagging program raises the coverage baseline, but it doesn't remove the structural ceiling or the organizational dynamics the number depends on.

Before cloud billing existed, IT infrastructure management solved the same underlying question: what does this infrastructure belong to, who owns it, what does it cost. And the methodology it used did not require the engineers who deployed the infrastructure to annotate it.

The approach is called configuration management, and its core insight is that ownership attribution can be inferred from observed behavior rather than from metadata that humans must apply. An agentless discovery process scans a network, notes what port ranges respond, classifies those responses against known device fingerprints, reads the process metadata and communication patterns of whatever answers, and builds a relationship graph from what it observes. A database that receives queries from a specific cluster of application servers, and from no other source, is classified as belonging to those application servers, with ownership following from the topology rather than from anything a developer documented.

Cloud infrastructure makes this even more tractable than the on-premises case, because providers record service relationships as structural configuration data: the metadata that governs how infrastructure runs, written before anyone thought about cost reporting. An IAM execution role that authorizes one Lambda function to write to one DynamoDB table documents a relationship between those resources at the provider level. A Kubernetes deployment spec selects pods by label, establishing service topology through the selector itself. A VPC configuration places compute and database resources in the same network segment, encoding proximity by construction. A CloudFormation or Terraform template provisions resources together, and the provider records that co-provisioning in the deployment history. None of this was written for cost attribution. All of it carries ownership signal, for anyone reading the provider APIs.

Configuration management discovery was designed for infrastructure that lived for months and changed quarterly. A nightly scan-and-classify cycle is adequate when a server has a three-year lifecycle. Cloud infrastructure operates at a different cadence: a container might run for four minutes, a Fargate task might provision and deprovision fifty times in a single hour. Traditional configuration management tooling, running on overnight cycles, misses the majority of ephemeral cloud resources. A cloud-native implementation does what configuration management has always done: it infers ownership from available signals, reading provider APIs at billing-record speed rather than network scans on overnight cycles.

Cloud engineers encode service identity into every architectural decision they make, whether they intend to or not.

Account structure is the most durable signal. Most organizations provision cloud accounts along business or application lines: the payments account, the analytics account, the production environment for the checkout service. An account is a governance boundary, an organizational decision baked into the structure of the estate itself. Resources deployed into an account inherit that attribution by construction, not by developer discipline. The same logic applies to resource groups in Azure, projects in GCP, and organizational units in a multi-account AWS structure. These boundaries exist everywhere, because they were created for operational and security reasons long before anyone started worrying about cost attribution. They carry ownership information simply because the people who drew them were thinking about ownership.

Below the account level, the signals get more granular. VPC structure, subnet assignment, IAM role names, security group membership, resource identifiers, deployment metadata: every one of these carries information about what a resource is and what it was built to serve. A database whose identifier contains "checkout-prod-rds" was named by an engineer who knew what it was for. A Lambda function with an execution role named "payments-service-processor" was configured by someone who knew which service it belonged to. These names exist in every environment, applied at deployment time for operational reasons, not to satisfy a cost attribution requirement that might not have existed when the infrastructure was built. The service identity is implicit in the infrastructure.

And then there is the relationship layer. Once an anchor resource is established (a resource that can be attributed through any of the signals above), relationship analysis extends attribution to the infrastructure that depends on it. A database whose security group permits connections only from a specific compute fleet's security group belongs to whatever application that fleet serves. A load balancer routing to a specific target group is serving the application behind it. An S3 bucket accessed by only one IAM role belongs to whatever service that role was created to enable. These inferences are the same logical operation that configuration management has applied to on-premises infrastructure for decades, now reading provider configuration APIs instead of scanning for network traffic. None of it requires engineering to tag anything. The provider records the configuration at provisioning time. The signals are readable from cloud APIs regardless of what traffic has since flowed.

When tags are the only attribution mechanism, the practitioner's capability floor is set by engineering's discipline in a given quarter. An active, enforced tagging program might reach 80 or 85 percent of spend. The remaining 15 to 20 percent represents the structural ceiling described earlier plus whatever drift accumulated since the last enforcement cycle. When the tagging initiative is deprioritized, that number drops to whatever percentage of spend happens to be in well-named accounts. The worst case is not a slow quarter but no answer: the practitioner cannot resolve the remaining gaps through any path that doesn't require engineering cooperation they don't currently have.

Signal-based attribution reads tags as one input among several. On an estate where tagging coverage is strong, that coverage carries directly into the attribution output and raises the baseline. The model also reads account structure, naming conventions, VPC assignments, IAM role metadata, and deployment metadata, signals that exist regardless of the tagging program's state. On a chaotic estate these signals are imperfect for the same reason the tags are: the engineers who didn't complete required tag fields also didn't maintain clean naming standards, and both problems trace to an organization that built infrastructure without cost attribution as a design constraint. Messy signals leave more for the next layer to resolve. Where the signals are insufficient, relationship analysis cascades attribution to dependent infrastructure. Where relationship analysis cannot make a confident inference, the practitioner encodes manual assignments for individual resources, tedious work but work they do themselves without needing anyone else's cooperation. On the worst-case estate, the floor is unconditional: the practitioner is slower, not blocked.

The asymmetry is not about which produces more accurate attribution when conditions are good. A strong tagging program raises the signal baseline. The asymmetry is about what happens when engineering discipline slips or the tagging program loses priority, as it does in most enterprise cloud environments at some point. When tags are the only mechanism and they're insufficient, the answer is unavailable until engineering cooperates. When tags are one signal among several, insufficient tags mean more manual work, not a blocked answer. This is what independent success actually requires: not that everything works smoothly, but that the function holds when it doesn't.

Cloudaware is, among other things, a CMDB for the cloud estate, applying the configuration management discipline described above at billing-record speed across the full cloud asset graph. The platform reads account structure, VPC assignments, IAM role metadata, naming conventions, deployment metadata, and tags where present, not as supplementary fallbacks but as the primary attribution layer. Relationship analysis extends attribution from anchor resources to dependent infrastructure automatically. Manual assignment fills the gaps that tags, signals, and relationships cannot resolve, allowing the practitioner to reach 100 percent attribution. Whatever tag coverage the organization has already built feeds into that output directly. The practitioner reaches complete attribution from wherever the estate stands, without waiting on engineering to close the remaining gap.

The signals for cost attribution (account structure, naming conventions, relationship graphs, deployment metadata) have always existed on the cloud estate. Observability tools have been reading those signals to map service topology without developer tags for years. The FinOps Foundation's own documentation lists CMDB data and observability tools as recognized attribution mechanisms. So why did the industry spend a decade building coverage dashboards and enforcement tooling around the one layer that requires perpetual engineering cooperation to maintain? Because asking was the only move the function thought it had.

The deeper issue is not tags but the FinOps team's dependency on a mechanism outside their control. A function that cannot produce its own foundational answer is accountable not for the answer but for asking. When the asking fails, it absorbs the fallout. Allocation is the foundation, and for a decade it was rented from people the function could not direct. A practitioner who can attribute every dollar on the estate without anyone's cooperation is no longer producing a report for finance to receive and question. Instead, they hold a complete account of where the money is. This is the only position from which anyone can say what is exposed and what to do about it. The point was never for FinOps to report the cost; it was to own it.

cloudaware

02

Unit Economics

Cost per customer is the wrong metric. Profit per customer is the number that connects cloud spend to business value.

The cost model has taken three months and two rebuilds. She has pulled apart the shared layer, the RDS clusters serving four products at once, the API gateway, the egress that arrives as one undifferentiated charge, and attributed every slice to the product that drove it. What she has is a complete cost for each product line, built from real usage rather than rough splits. The slide goes up.

The CFO reads to the bottom and asks, "Should I be worried about any of these?" She can show him exactly where every dollar goes, down to the workload behind it, which product carries the shared layer and which one the egress. His question is not what the products cost but which of them to worry about, and the model she spent three months building cannot answer it.

The model has the cost side but not the revenue side. Cost per product tells you what a product costs to run, not whether that cost is worth paying. Cost alone cannot tell you which products earn more than they consume, or whether the efficiency you just won landed on a product worth the effort. Take what an application earns, then subtract what it costs to run. What's left tells you whether the application pays for itself, and by how much. Finance calls that number contribution margin.

The calculation takes two numbers, and FinOps has only ever held one of them. A practitioner can produce the cost of an application down to the dollar and still not say whether the application is worth running, because the second number has always lived in another system, owned by another team. The billing system has already counted the revenue, to the dollar. The cost of an application, split honestly across infrastructure that serves a dozen of them, is the part that takes real work.

Every dish on a restaurant menu has a plate cost, the protein and the produce and the share of the gas it takes to cook. A chef who watches only that number will tell you the steak is the problem. It costs more to put on a plate than anything else on the menu, so cutting it drops the average plate cost and makes the kitchen cheaper to run.

What that chef cannot see is the other side of the ticket. The steak sells for four times what it costs to plate, and the table that orders it orders wine. The pasta that looks so efficient, cheap to fire and quick to send out, clears a few dollars over cost and holds a table for an hour. Restaurants that have studied this even have a name for it, menu engineering, ranking each dish by what it contributes and how often it sells. The expensive dish is often the one carrying the room, and the cheap one is often the one losing money on every cover.

Read the menu by cost alone and you cut the dish that pays for the lights. The figure that would have stopped you, what each dish earns against what it costs, never reaches the kitchen. It lives at the register, on the other side of the house, and the chef cooks without it.

The analogy breaks where the kitchen ends. A chef can read the whole menu in an afternoon, a dozen dishes on a single card. A cloud estate has thousands of dishes coming out of one shared kitchen, the same infrastructure plating every order at once, and the cost of that kitchen does not divide itself cleanly by dish. Splitting it honestly is the work. The mistake, though, is identical in both rooms. Optimize the cost of something without knowing what it earns, and sooner or later you cut the steak.

Contribution margin counts only the costs that move with the product itself, the compute, the storage, the data transfer, the managed services it actually consumes, and leaves out the fixed overhead that would be there whether or not the product shipped. Fold the corporate G&A and the office lease into a product's cost and you have measured something, but not the thing that tells you whether the product earns its keep. Contribution margin isolates the product's own economics and leaves the broader profitability question, the one that folds in labor and overhead, for finance to layer on top.

> Optimize the cost of something without knowing what it earns, and sooner or later you cut the steak.

The work was born from the complexities of the cloud bill, and the FinOps Foundation has already named where it leads. Its guidance on cloud unit economics sets cost against revenue to find where a product stops costing money and starts making it, and to let a practitioner say what that product adds to gross margin.

You can watch a team reach for it in the Foundation's own library. In one case study, a FinOps practitioner working directly with the CTO set out to "calculate the Cost-to-Serve for each customer purchasing product Tier A," because Tier A's gross margin was, in the practitioner's words, "falling short of our expectations." The question was not what Tier A cost to run. It was whether the customers buying it were worth serving. Answering it was a cross-functional effort, finance and data and product and engineering lining up what each customer paid against what each customer cost to serve. One product tier, and a single margin number to show for it.

The two numbers had simply lived apart, so producing a margin meant a project every time. When they share one model, the margin is a query rather than a project, and it can run every month instead of once a year. It sorts the products that earn from the ones that lose, and for many businesses that is answer enough.

A product's margin is an average across the customers it serves, and an average hides its own spread. Two customers on the same product can sit at opposite ends of the economics. One self-serves and barely touches support, while the other runs heavy queries, pulls terabytes of egress, and files a ticket a week. The product-level margin blends the two into a single comfortable number. The same model that produced the product's margin runs one level deeper, attributing both cost and revenue to the customer rather than the product line. A customer who looked fine inside the product's average can turn out, on their own, to cost more than they pay.

A company serving millions of customers on a single product cannot read profit one customer at a time, and would learn little if it could. It reads at the level of cohorts and tiers, by pricing plan, by acquisition channel, by usage band, a profit per customer that is an average across a segment chosen because the customers inside it

behave alike. A company with sixty large accounts has the opposite grain and reads profitability one account at a time, because at sixty accounts the spread between them is the whole story.

Netflix reported its streaming business in two segments through 2016, domestic and international, and published a contribution profit for each, revenue less the cost of content and the marketing spent to win members. For the full year, the domestic segment earned a contribution profit of $1.8 billion on $5.1 billion of revenue. The international segment, on $3.2 billion of revenue, posted a contribution loss of $309 million.

The consolidated company looked healthy. Beneath that figure, the segment carrying the cost of expansion was running at a loss, visible only because Netflix set its two segments side by side rather than averaging them into one. A single consolidated number would have hidden exactly what management needed to see.

The same holds inside a FinOps model. A cost per customer that is trending down looks like progress, but if it is falling because the customers being added cost little and pay even less, the trend is a warning, not a win.

Consider two segments a cost model sees as identical, both served by the same shared infrastructure at the same $2,200 a month per customer. The first holds four hundred mid-market accounts paying an average of $18,000 a year, which is $1,500 a month against that $2,200 cost. Its margin on infrastructure alone is negative. The second holds sixty enterprise accounts paying $180,000 a year, or $15,000 a month, against the same $2,200. Its margin is 85 percent. The cost model records the same $2,200 for both, and it has no field for what separates them.

You would not price them the same way, invest in keeping them equally, or accept the same cost to win more of them. There is a reasonable case for pulling infrastructure investment out of the first segment and letting service degrade rather than keep subsidizing a loss, and an equally reasonable case for over-building reliability for the second and calling it deliberate.

Seven hundred dollars a month across four hundred accounts is more than three million dollars a year. A cost program would attack the only number it can see and try to shave the $2,200. But the mid-market segment is not expensive. It is underpriced, and price is the number the cost model never had. The fix might be to charge more, to sell something different, or to carry the segment on purpose.

Bill Gurley, then a general partner at Benchmark, described a market that had stopped asking whether the businesses underneath its valuations actually worked. In a 2016 essay titled "On the Road to Recap," he pointed to burn rates running "5-10x those of the 1999 timeframe" and most companies "operating far, far away from profitability." Growth was easy to show and easy to fund. Whether the growth was worth anything was a different question, and almost no one was asking it. What Gurley wanted was "a return to an appreciation for sound business execution."

The cheap capital lasted a few more years before rates rose and funding tightened, and the companies that had grown without ever proving their economics were the ones left exposed. "Path to profitability" turned from a phrase founders could wave away into the first question every board asked.

The same blindness operates inside a single company. A product can grow, a segment can expand, the spend can be trimmed, and none of it says whether the business is better off, because none of it is set against what the customer pays. Without that comparison, a company can scale into a hole and call the descent progress.

The architects of Activity-Based Costing noticed that crude overhead allocation consistently undercosts low-volume products and overcosts high-volume ones. The high-volume products, made to look more expensive than they are, get overpriced and lose business. The remaining mix absorbs the overhead they leave behind, which deepens the distortion. Margins compress a little each year, gradually enough that nothing trips an alarm, hidden in the normal noise of the business. They had a name for where it ends: the death spiral.

Any organization that allocates shared infrastructure without a revenue counterpart to calibrate against runs the same risk. Such an organization optimizes the cost of segments whose profitability it cannot see, while the profitable segments quietly carry the rest.

Cloudaware is, among other things, a model of the entire cloud estate, with the cost of every resource allocated down to the workload that drove it. It also holds the subscription, billing, and customer data where revenue lives, so cost and revenue sit together rather than in two systems waiting to be joined. A practitioner reads cost per customer and revenue per customer from the same source. There is no parallel pipeline to build and no schema to keep in sync as either side changes. The margin is visible by product, by segment, or by customer.

For as long as the two numbers lived in separate systems, the function could answer only half the question. It could say what a product cost, never whether the product was worth keeping. What a product earns, the half that settles things, fell to whoever happened to hold it. The CFO asks which products to worry about, and a cost model, however exact, can only point at the spending.

On one model the two numbers stop living apart, and the person who built the cost is the one who can now read the margin. That practitioner is not handing finance a report to receive and question. They can say which plan is underpriced and by how much, whether the next dollar of reliability should go to the segment that pays for it or the one already underwater, which customers are worth keeping and which are being carried at a loss. The conversation stops being what the infrastructure costs and becomes where the business makes money and where it loses it. Cost is a record of what the business spent. Margin is a position on what should be done, and the practitioner who owns the margin is no longer reporting the past but arguing for what comes next.

cloudaware

03

Exposure Management

The goal of FinOps is cloud cost control. A team that has eliminated every identifiable waste but cannot answer “are we in control?” has not reached the goal.

The slide goes up, and it is a strong one. The practitioner walks the room through a year of work. A rightsizing pass across three major accounts. A Reserved Instance renewal that caught a commitment cliff six months before it materialized. Scheduling automation on the dev and staging fleets nobody had touched in two years. Annualized savings of $1.2 million, documented and attributed. The CFO acknowledges it, then leans forward and asks, "What are you going to find next quarter?"

The practitioner has no good answer. Not because the pipeline has run dry or because she missed something (she didn't) but because the question has a structure with no closing condition. The $1.2 million delivered this quarter is now the baseline. Finding another $1.2 million next quarter means performing to standard. Finding $800K means underperforming. There is no number that ends the conversation, because nothing was ever defined as enough.

Waste elimination and cost control are not the same thing. FinOps has become very good at waste elimination but has never had a tolerance, a defined threshold past which the job is done and the question "are we in control?" has a yes-or-no answer backed by evidence. Every other financial discipline in the same organization already has one. Without it, savings will always reset the conversation.

A treasury department manages to a coverage ratio, a declared threshold specifying the minimum level of liquid assets the organization must hold against its obligations under stress. Basel III calls it the Liquidity Coverage Ratio, and a bank must stay above the line the regulator sets. The CFO doesn't ask treasury "how much more liquid could you be?" She asks "are we in compliance?" and treasury answers yes or no.

The order is fixed. Security and liquidity come first, and the treasurer turns to maximizing yield only once those are satisfied. Optimization begins after the threshold is met, not instead of meeting it. A FinOps function operating on savings targets alone is optimizing without a threshold to satisfy first.

A lending institution doesn't try to eliminate defaults from its loan portfolio, since a portfolio that never defaults is too conservative to be profitable. The institution sets a loss threshold, the maximum default rate it can absorb given its capital position and provisioning. When losses track within tolerance, the function is working. When they breach it, there is a specific finding with a specific remediation path. The question "are we done?" has a clear answer in either case.

Enterprise risk frameworks define tolerance as the acceptable variation in outcomes relative to a declared objective. Risk appetite is the strategic posture ("we accept moderate cost risk to support growth"). Tolerance is the operational guardrail, a measurable boundary past which an outcome requires action. Only a declared tolerance makes the function finite. The two standards corporate risk and audit teams actually run on, COSO's Enterprise Risk Management framework and ISO 31000, both describe risk treatment as complete when residual risk falls within declared tolerance, not when it reaches zero or some minimized level. The function runs until the answer is yes, then it shifts to monitoring. FinOps has never had this shift available because tolerance has never been declared.

After the 1987 Black Monday crash, Dennis Weatherstone, then chairman of JP Morgan, demanded a single daily risk report, one number across all the bank's trading books that expressed how much the portfolio could lose by tomorrow's close. The result, the 4:15 Report, was built around a metric called Value at Risk. VaR expressed risk as a threshold, the loss level that would not be exceeded on 99 of every 100 trading days. JP Morgan published the methodology publicly in 1994, and VaR spread across the industry within a decade, putting risk on a common denominator and making portfolio comparison possible across different trading books.

Waste reduction and control are orthogonal.

VaR had the same blind spot a cloud cost estimate has. It set the threshold of the tail and said nothing about the depth of it. A portfolio could show a favorable 99 percent VaR while containing positions that, on the remaining 1 percent of days, produced

losses an order of magnitude worse than the number implied. The measure captured the edge of the distribution and threw away everything beyond it.

In 1998, Long-Term Capital Management required a Federal Reserve-coordinated bailout despite having looked well-contained on conventional risk metrics, its losses landing in exactly the tail those metrics had discarded. The industry built a measure of that tail, Expected Shortfall, the average loss on the days the threshold is breached rather than the threshold itself.

Basel III, the international accord governing bank capital, requires institutions to hold capital against that tail, real capital they cannot deploy elsewhere. The expected outcome is manageable, because they planned for it. The tail is what breaks an institution, so the tail is where it sets a tolerance and reserves against the loss. An organization managed to its expected outcome controls the median and nothing beyond it. Everything past the median is exposure it carries without acknowledging, because a point estimate collapses the whole distribution into one number and discards the tail.

A company that purchased three-year Savings Plan commitments during its 2021 migration has a cliff date when those commitments expire. The billing impact is calculable, the difference between committed and on-demand rates multiplied across the affected fleet. For a large organization, this can represent hundreds of millions in annualized compute shifting pricing tiers overnight. The date has been visible in the billing data for years. The magnitude is calculable today. The fix, a renewal program with a deadline and a business case, is well within the standard playbook.

An ML training platform that has grown to support forty teams running jobs on shared GPU infrastructure has no per-team resource limits, no job-level cost ceilings, and no alerting on GPU spend velocity. Most months, the platform runs close to budget. Three times in the past two years, a team ran an unreviewed training job over a holiday weekend and the bill for those 72 hours exceeded the previous month's total for the entire platform. The architectural conditions that produced those events have not changed.

The overruns are not a mistake to fix. The GPU fleet is doing exactly what it was provisioned to do, and the bill swings with what the teams running on it choose to do. This is an exposure, not waste. It is a condition present in the architecture now, with financial consequences that vary depending on events that have not happened yet. One type of risk is a debt with a known balance, the other an option you've written without knowing the strike price.

Deterministic waste recovery reduces expected cost directly. Architectural exposure management changes the shape of the distribution, specifically by collapsing the tail without necessarily moving the median. An ML platform with resource quotas in place might run at a similar monthly average as the uncontrolled version. Its worst-case outcome, the number you need to absorb when a team has a bad weekend, drops significantly. An architectural control's value was always there in the architecture, but until now it was impossible to put a number on. Expressing cost as a distribution rather than a point estimate changes that. Whether you present it as avoided cost, risk reduction, or exposure management is a choice that belongs to the practitioner.

One team has done every piece of waste remediation correctly. Right-sized instances across every account. Reserved Instance coverage above 80 percent. Idle resources terminated within 24 hours. Spot use maximized wherever workloads can tolerate interruption. A weekly optimization review that clears every actionable finding on the list. The ML training platform, though, has no resource governance. The event-driven streaming platform that serves the core product has no cost ceiling. When the business runs a major launch, the infrastructure auto-scales, and the cost consequence arrives in the bill four weeks later, with no visible connection to the launch that produced it. The architectural conditions that generated the worst billing months in the past two years are still present. This team eliminates waste well but does not have control.

A different team carries deliberate headroom, provisioned above the expected baseline because reliability matters and the business made that call explicitly. A pure optimization pass would flag those resources. The team knows. They have defined tolerance thresholds for every major workload, resource quotas on the training

platform, spend velocity alerts, and a ranked list of the five findings closest to the tolerance boundary, each with a tail cost, a remediation timeline, and an owner. This team has waste inside the bounds they've defined and is in control.

Waste reduction and control are orthogonal. Exhaustive, disciplined waste elimination is valuable on its own terms, but it does not add up to cost control. Deliberate waste is compatible with full control. The field has spent a decade building the first. The second was never available to build, because there is no "are we within tolerance?" until a tolerance has been declared.

Before FAIR (Factor Analysis of Information Risk), the prevailing approach to communicating risk was qualitative labels, the same language most FinOps teams use for cloud cost risk today. High, medium, low. Red, amber, green. Heat maps produced by the judgment of whoever happened to be closest to the system. Jack Jones, then at a major financial institution, was presenting one of these assessments to the board when a director asked how much risk the organization was carrying. The question was structurally unanswerable in the language he was using.

Jones went to the dictionary that afternoon. The definition of "risk" that fit was "exposure to loss." Risk is not a color but a dollar range, the range of financial loss to which an organization is exposed given the conditions in its environment. Colors cannot be added up, because two "high" ratings are not the same magnitude. A heat map cannot tell a board whether total exposure is inside or outside what the organization has agreed to accept.

FAIR replaced the label with a loss exceedance curve, a probability distribution over dollar outcomes. It made prioritization possible and made the question "are we within tolerance?" answerable with evidence rather than with subjective confidence.

Most cloud cost risk is still reported as a point estimate with a severity label. An overprovisioned database cluster is "$200K per year." The ML platform with no governance is "$200K per year typical." Both receive the same expected number. The ML platform has a tail that runs to five times that figure on a bad quarter. The database cluster stays in a tight band around a known cost. Collapsing both into "$200K" throws away the only variable that matters for the tolerance conversation.

Treasury, the trading desk, and the credit function all manage a danger they cannot remove. None of them can stop the market from moving or a borrower from defaulting. Their only lever is how much capital to hold against the loss. The

conditions that produce a cloud cost tail are not external. The ungoverned training platform and the launch with no ceiling live inside the architecture, and the practitioner can change them. Those functions reserve against their tail. A FinOps function can collapse the tail itself.

The old quarterly business review ran like this. "We recovered $1.2 million in savings this quarter through rightsizing, Reserved Instance renewals, and scheduling automation. Here are the line items. We're requesting the same budget for next year." The CFO acknowledges the savings, then asks the question she always asks about next quarter. A specific number invites scrutiny of whether it's large enough. No number signals the pipeline has dried up.

The same meeting, run from a different frame, looks like this. The practitioner opens: "Our total expected cloud cost exposure across the estate is $4.2 million annually. Our p95, the level of spend we need to be able to absorb if conditions are adverse, is $9.8 million. Our declared tolerance is $10 million on the tail. We're within it."

"Here are the five findings closest to the boundary. The ML training platform has no resource governance. Its tail exposure is $620K and we can close it with a two-week engineering sprint. The event-driven streaming platform has no cost ceiling. Its tail is $1.1 million and it requires a deployment gate change currently in the Q3 backlog. The Savings Plan cohort expiring in July is a deterministic $340K annualized step-change. The renewal decision needs to be signed off by April 15th or we miss the window. The remaining two are architectural controls with a three-month implementation timeline each."

"We're recommending we address the top three this quarter. That reduces the tail from $9.8 million to $7.7 million at a cost of roughly $180K in engineering time. We're within tolerance today, but I'd want the streaming platform finding approved before Q2."

The CFO can approve the program, defer the architectural controls, ask about sequencing, push back on the tolerance threshold. What she cannot do is reset the conversation with a question the practitioner cannot answer, because the practitioner has already answered it. The findings are ranked by their distance from the boundary, not by the size of the savings they would produce.

"What will you find next quarter?" is only unanswerable when the conversation is about savings. When the conversation is about tolerance, the answer is already on the table. We're within it.

If the goal is cost control and not only waste elimination, then someone has to declare the tolerance. Not just for individual workloads, but for the estate as a whole. A number that finance signs off on. A boundary that doesn't reset when the baseline changes. A threshold that makes "are we in control?" answerable by evidence rather than intuition.

Every financial risk function that has made this transition (treasury, credit risk, the trading desk) has made it with the involvement of the people who hold financial accountability. The tolerance was not set by the risk function alone. It was negotiated, documented, made durable. The treasury team goes to the board with a coverage ratio and asks for sign-off. The credit function presents its loss threshold to the CFO and the audit committee. The tolerance becomes part of the organization's formal risk posture, not a practitioner's internal benchmark but a declared position the organization can be held to.

FinOps has not had this conversation at most organizations, but none of the pieces are exotic. The exposure can be quantified. What it takes from there is a declared tolerance finance signs off on, a review cadence, and someone who owns the briefing, the same structure every other financial function in the building already runs on.

The FinOps Foundation's State of FinOps 2026, drawn from 1,192 practitioners representing more than $83 billion in annual cloud spend, found that implementing governance and policy at scale had become practitioners' top priority for the year ahead, with workload optimization slipping to second. The field is turning toward guardrails where it used to build dashboards.

Cloudaware is built for this. It holds the cost of every resource in the estate, allocated down to the workload that produced it, and on the same model it quantifies what each workload is exposed to, the shape of its distribution and the size of its tail.

Expected cost and tail exposure come from one source. A practitioner can put the whole briefing on the table, what the estate is expected to cost, what it is exposed to at the tail, and the tolerance it is operating within, and answer "are we in control?"

The one part still unwritten is the conversation itself, where a practitioner brings a CFO a tail exposure number and a proposed tolerance and asks the organization to stand behind it. The teams that have that conversation first will run a materially different practice than the teams that don't, not because they found more waste, but because they defined what done means and made the organization agree to it. A savings number can always be asked to grow. But when a declared tolerance is met, the practitioner can finally say, "We are in control."

cloudaware

04

FinOps for AI

AI cost forecasting isn't broken because AI is new. You just can't forecast non-deterministic cost as a single number.

The slide goes up. It is the year's budget for the new AI coding assistant, a single number the finance team can plan around. The practitioner has done the work, and the assumptions behind the number are sound. The CFO looks up and asks, "How confident are you in that number?"

The honest answer is that it is the most likely number, which is not the same as a number to count on. The practitioner can stand behind the estimate and still not say how high the bill could go in a month when usage runs hot, because a single number was never built to hold that. The number shows the expected case and nothing on either side of it.

December 2025. Uber deploys Claude Code, Anthropic's AI coding tool, across roughly five thousand engineers, all at once. To ensure adoption doesn't stall in the middle layers of the organization, they build a leaderboard, teams ranked by total AI tool usage, visible across the engineering org. The leaderboard works exactly as designed. Teams compete, usage climbs, and the metrics look the way a successful rollout is supposed to look. By April 2026, four months into a twelve-month fiscal year, CTO Praveen Neppalli Naga is looking at the numbers. He tells The Information, "I'm back to the drawing board, because the budget I thought I would need is blown away already."

A month later, President and COO Andrew Macdonald appears on the Rapid Response podcast, still unable to draw the line between what was spent and what it bought. "That link is not there yet. Maybe implicitly there's more that is getting shipped, but it's very hard to draw a line between one of those stats and okay now we're actually producing like 25% more useful consumer features." He adds, "If you're not actually able to draw a direct line to how [many] useful features and functionality you're shipping to your users, that trade becomes harder to justify."

The CTO has the cost problem, what happened to the money. The COO has the value problem, what did the money buy. Leadership backed an intentional rollout, building in a leaderboard as a deliberate adoption mechanism, a reasonable tool for driving organizational behavior change. They still burned the year's budget in four months, not because engineers were irresponsible, and not because the leaderboard was a

mistake, but because nobody had a model that could hold the cost. The model they built it on was a point estimate, and a point estimate cannot forecast a non-deterministic cost.

When someone built Uber's AI budget in the fall of 2025, they did what every budget process does. They took a reasonable assumption about usage, multiplied it by a cost-per-unit, and produced a number that looked defensible in November when it was built. What that model didn't account for is that AI token consumption doesn't scale linearly with headcount or with tool adoption rates. It amplifies with usage patterns, prompt behaviors, and the second- and third-order effects of organizational incentives. A leaderboard that ranks teams by AI tool usage is, from a cost modeling perspective, an uncapped variable in a linear forecast. You cannot forecast a non-deterministic cost as a single number and be right. You can model it as a range and be honest.

In September 2025, Mavvrik and Benchmarkit published the 2025 State of AI Cost Governance Research, a survey of 372 companies on AI cost forecasting accuracy. Only 15 percent of those companies forecasted AI costs within a 10 percent margin. The other 85 percent missed by more than 10 percent, and nearly one in four missed by more than 50 percent. For a $1M AI budget, that means 85 percent of companies are off by more than $100K, and almost a quarter are off by more than $500K.

The standard industry response to data like this has been to call for better forecasting tools. More frequent data refresh. Better dashboards. But the problem was never precision. Better forecasting tools still produce point estimates, and a point estimate cannot represent a cost that lives in a range, however precise it gets.

The objection that a distribution is too wide, that the data isn't granular enough yet, asks for more rigor from the one model that is honest about what it doesn't know. The point estimate it would keep instead misses by more than 10 percent in 85 percent of cases. Granularity sharpens a distribution over time, but it cannot rescue a point estimate that has already failed at scale. The width that reads as a weakness is the first honest measure of the exposure anyone has put in front of the room.

Cloud infrastructure costs include variable components like autoscaling groups, spot instances, and egress charges, but they're anchored to deterministic resource consumption. A virtual machine has a size. It runs for a number of hours. Those hours times that size produce a bill that behaves predictably from utilization patterns, even when utilization fluctuates. The variance exists, but it's bounded by infrastructure decisions that engineers make deliberately, at provisioning time, before the workload runs.

AI token costs don't work that way. The consumption unit, the token, is a function of what a human types into a prompt, how the model responds, how many times they iterate, and which model they choose for the iteration. None of those inputs are set at provisioning time. They're determined at the moment of use, by individual behavior, multiplied across however many engineers are running queries. Scale that across five thousand engineers on a leaderboard and you've created a cost surface that behaves like insurance risk, individual events unpredictable, aggregate distribution characterizable. Insurance risk has been modeled accurately for more than two centuries, and the model has never been a point estimate.

> The AI bill lands and everyone looks to the person who already had the model.

A federal flood insurer pricing a policy on a coastal property is not trying to forecast the storm. They're not predicting in which year the flood occurs or what the rainfall will be. They're building a distribution of outcomes. In any given year, this location carries some probability of a flood that causes some magnitude of damage. That probability, multiplied by that expected loss, produces a premium, the price of bearing the risk without knowing the specific outcome in advance.

The information content lives in the width of the distribution, not in a single point on the curve. The actuary who tells you there's a 1 percent annual probability of a $500K flood has given you a p99 tail figure that a point forecast of "no flood this

year" cannot. The homeowner who responds "but when will it flood exactly?" has confused precision with accuracy. Precision means a tight number. Accuracy means a right one. For a storm you cannot predict, a wide and honest distribution is more accurate than a narrow and confident one.

Expected spend is what consumption looks like when usage patterns are stable and nobody's running anything unusual. The 95th percentile (the p95) is what consumption looks like when a team runs an extended automation job, when prompt behavior spikes, when a leaderboard produces a month where everyone is competing on usage. The gap between expected and p95 is the unmanaged tail: the exposure that lives in the distribution but doesn't appear in any spreadsheet.

The analogy breaks down at the governance layer. A flood insurer cannot prevent the storm. The risk is fully exogenous, and the only variable it controls is the premium. An engineering organization has levers the insurer doesn't, rate limits, dedicated capacity reservations, per-deployment budget caps, gateway logic that routes lower-stakes queries to less expensive models automatically. The tail of an AI cost distribution is an organizational behavior, not an act of nature, and it's governable as soon as there's a distribution to work from and a number that makes the controls worth building.

Total AI spend as a distribution tells you the tail exists. AI spend by application (the business function consuming the tokens, the owner responsible for the behavior) tells you which tail is your problem and which controls close it. Without that attribution layer, the distribution describes exposure you can observe but can't act on. With it, the distribution becomes a map. This application owns this share of the tail, this control closes this gap, this investment has this return. Attribution is what makes the distribution governable rather than merely visible.

The attribution problem for AI costs is structurally identical to the attribution problem FinOps teams solved for cloud infrastructure. Without a reliable mapping from a cost event to the business application that generated it, you can measure aggregate spend but you can't govern it. The difference is that AI token consumption flows through shared API endpoints rather than tagged resources. The relationship model that allocates cloud spend by account, resource group, deployment metadata,

and usage pattern applies directly to AI services. Azure OpenAI, AWS Bedrock, and SageMaker are assets in the same estate. The billing unit shifts from compute-hours to tokens, but the framework doesn't.

"We should invest in AI cost governance" is a request. It competes with every other item in the engineering backlog and usually loses, because it's abstract, with no number behind it and no consequence attached. It's a reasonable ask that dies in triage.

"Our expected monthly AI spend is $200K. Our p95 is $2M. That $1.8M gap is unmanaged tail exposure. Application-level attribution and rate limiting collapse the tail to $400K. The engineering investment is $150K."

That is a decision. The distribution generates the business case for closing the problem, not just describing it. The $1.8M tail gap is the number that didn't exist before, the figure that converts a governance conversation into a capital allocation conversation, $150K of engineering work to close $1.4M of tail exposure. Before the distribution, you're asking for investment in an abstract control. After it, you're presenting a return.

LiteLLM's proxy layer lets operators bind spending limits to individual API keys, capping tokens per minute, requests per minute, and the monthly budget. That's per-team or per-application spend control enforced at the gateway, before a query reaches a model, without requiring engineers to change anything about how they write prompts. Azure OpenAI's provisioned deployments reserve dedicated capacity for a given workload, so a production customer-facing service and an experimental engineering prompt draw on separate capacity and bill independently. Custom gateway logic can route lower-stakes queries to less expensive models automatically. The technical layer for governing AI spend is more mature than most FinOps teams have had time to map.

What it's waiting for is a number that justifies building it.

In February 2026, the FinOps Foundation formally updated its mission statement, from "advancing the people who manage the value of cloud" to "advancing the people who manage the value of technology." J.R. Storment, the Foundation's executive director, said it plainly, "This is not a cloud cost discipline anymore. This is a broad technology value discipline." The same survey behind that mission change, the sixth annual State of FinOps report with 1,192 respondents, found that 98 percent of FinOps practitioners now manage AI spend, up from 31 percent two years prior, a curve tracking almost exactly with the moment AI coding tools became standard developer infrastructure at large organizations.

The practitioners managing AI spend now are, for the most part, the same practitioners who built the cloud cost models. They're carrying those models into a domain where the models don't behave the same way. Their organizations won't know it until the bill arrives.

The newest wave of AI cost tools is framed around "cost per outcome," shifting the question from what was spent to what the spend produced. It's the right question asked too early. Cost per outcome is a ratio, and a ratio with an inaccurate numerator is noise. Modeling cost as a range and attributing it to the application that generated it is upstream of the outcome conversation. It fixes the numerator, and nothing about the ratio means anything until the numerator is right.

The practitioner who has the distribution is in a different meeting than the practitioner who has the forecast. The forecast creates one kind of conversation: "Why were we off? Who was responsible? What's the revised projection?" That's a reactive conversation, and the practitioner in it is explaining what happened. The distribution creates a different one: "Our expected spend is X. Our p95 is Y. We're within tolerance, or we're not, and here's what closing the gap costs." That's a conversation the practitioner is running. The AI bill lands and everyone looks to the person who already had the model.

The Uber story is useful in that meeting, not as a cautionary tale but as cover. This happened to a company that made the right decisions. Strong engineering culture. Leadership alignment. Deliberate rollout. They still burned the year's budget in four

months because the cost structure of AI consumption doesn't respond to the model everyone used. Once that's established, the question shifts from "how did we let this happen" to "what's our number, and what's our tolerance."

Cloudaware produces both the attribution and the distribution from one model. It allocates token spend to the application that generated it the same way it allocates infrastructure cost, and it reads that spend as a distribution rather than a single number, expected cost, p95, and the tail, by application. The practitioner sees which application owns which part of the tail and what closing it is worth, from one source rather than a cost report and a separate estimate stitched together.

Every distribution still needs a line, a level of tail the organization builds against and a level it accepts, and where that line sits is a business decision made with the people who own the competitive plan and the appetite for overrun. The practitioner brings the number and a proposed range; the organization sets the line. What none of them can settle yet is how much of the tail to carry on purpose, because that turns on what the spend actually buys, and that number doesn't exist at most organizations — Macdonald, five months into the rollout, still couldn't draw the line between token spend and shipped features. But it is a ratio, and its numerator is the one the practitioner just built. The cost of AI was the question that looked impossible, and it is the one now answered. What that cost buys is the question still open, and it will be settled on the number the practitioner already holds.

cloudaware

Everything a FinOps platform does and more.

Allocation, unit economics, exposure management, and AI spend on one complete model of your cloud estate. A complete understanding of your technology portfolio and a position on what to do about it.

Schedule a demo

Learn more at cloudaware.com