OPEN SOURCE SUMMIT · PRAGUE · 9 OCTOBER 2026

The Bill That Fixes Itself

Trustworthy Agentic FinOps "with Legs"

Closing the loop on multi-cloud cost with Crossplane, Argo Workflows, OpenTelemetry and CloudEvents, and the trust protocol that earned it autonomy.

Pratik Mohapatra

Confluentis Consulting

Cloud cost dashboards are everywhere. Closed loops are not.

Most FinOps practice ends at a recommendation that a human may or may not apply. This talk is about what it takes to let software apply it instead, and what has to be true before anyone sensibly allows that.

THE PROBLEM, PRECISELY

It was never a visibility problem.

The estate in question runs entirely on PaaS across two public clouds. No virtual machines, which is where most cost tooling does its best work. The team had dashboards and knew their spend profile. Three things still defeated them.

1

Volume beyond review

Hundreds of PaaS resources across prod and non-prod cannot be meaningfully reviewed faster than monthly. By the time a monthly review finds over-provisioning, the waste has already been paid for. The review cycle is itself the constraint.

2

Unauditable remediation

Different teams, different thresholds, different approval chains, changes made directly in consoles. Six months on, nobody can say why a resource sits at its current tier, or who moved it last.

3

Impermanence

The damaging one. A rightsized resource drifts back within days. Someone chases a transient spike, upgrades the tier, the spike passes, the upgrade stays. The saving disappears silently until the next review.

Context: a global consumer goods enterprise. Estate anonymised throughout this deck.

THE SHAPE OF IT

An always-on loop, every six hours.

1 Collect

Metrics from both clouds through one OpenTelemetry collector with dual exporters. Cost exports ingested alongside.

2 Analyse

A state machine hands a structured payload to a pluggable LLM and gets back proposals, each with a projected saving, a confidence score and an action class.

3 Check

Three independent policy layers evaluate every proposal in sequence before anything is allowed to happen.

4 Act

The agent never calls a cloud API. It patches a YAML field, and the control plane reconciles reality to match it.

Separate schedules handle off-hours database pausing and nightly SKU catalogue refresh, so time-critical work is never held up behind a long analysis run.

The bill that fixes itself

An always-on agentic FinOps loop. Open source end to end.

Confluentis Consulting

YOUR MULTI-CLOUD ESTATE

Microsoft Azure | Google Cloud | AWS

1 COLLECT

Argo CronWorkflow, every 6 hours

One collector per cloud, fanned out in parallel

2 ANALYSE

One LLM port, five adapters

A state machine, not a prompt chain

3 DECIDE

CloudEvents over Message brokers

Policy decides what runs unattended

4 ENFORCE

Crossplane applies the change

Declared state, not a script that ran once

drift re-enters the loop

CROSSPLANE RECONCILES EVERY 30 SECONDS

Someone bumps a SKU at 2am. The saving is back within half a minute.

THREE-LAYER POLICY

What it may touch. What it may change it to. How far it may go in one step. Declared outside the model.

TRUST GATES

Recommendation-only until FinOps validated 80% of proposals over 30 days. Production databases are never automated.

WHAT IT RETURNED

40-70% storage

60-70% dev databases

OPEN SOURCE, ALL THE WAY DOWN

Argo Workflows | Crossplane | CloudEvents | OpenTelemetry | Prometheus | Kubernetes

BUILT ON

Everything load-bearing is open source.

Crossplane

Enforcement. Continuous reconciliation across both clouds, every thirty seconds.

Argo Workflows

Orchestration. The six-hour DAG, with step-level retry, entirely in-cluster.

OpenTelemetry

Observation. One collector, dual exporters, no self-managed aggregation tier.

CloudEvents

Approvals. One envelope across Azure Service Bus and GCP Pub/Sub.

CSI Secrets Store

Runtime secret injection. No long-lived credentials anywhere in the system.

Prometheus + Grafana

Local metrics store and the single cost view across both clouds.

CNCF graduated projects throughout. Incubating components were replaced with cloud-managed equivalents in production to meet enterprise procurement standards, which is itself one of the talk's smaller arguments.

HOW THE AGENT GETS ITS CONTEXT

Prose for prose. Arithmetic for numbers.

Almost every agent tutorial assumes a pile of documents, so it reaches for retrieval. A cloud estate is not a pile of documents. It is resource names, tiers, timestamps, CPU percentages and prices per hour. It is a spreadsheet.

LOOKED UP, NEVER SEARCHED

- Metrics, cost rows, utilisation history
- The SKU catalogue and its permitted ranges
- Thresholds, tags, ownership, policy values

Filters, joins and sums. The machinery any database has had for fifty years.

RETRIEVED, WHERE IT EARNS ITS PLACE

- Pricing and provider documentation
- Runbooks and notes from past incidents
- Previous decisions, and why one was rejected

Genuinely wordy material, where the question is what something is about.

To a system that matches on meaning, "14%" and "19%" are practically the same word. To your bill, one of them is fine and the other one is money.

THE ONE ARCHITECTURAL CHOICE THAT MATTERS MOST

Terraform provisions. Crossplane enforces.

For a cost system, provisioning once is not enough. The value is not that a resource was made smaller. It is that it stays smaller.

Triggered

Terraform acts when you tell it to act. Between runs, reality is free to drift, and drift is precisely how a manual console change quietly undoes a saving.

Continuous

Crossplane acts when reality diverges from declared state, and it checks every thirty seconds. A manually upgraded tier is detected and corrected before anyone files a ticket about it.

The saving is structurally protected, not merely executed. That distinction is the whole talk.

KEEPING THE AGENT INSIDE ITS MANDATE

Three policy layers, evaluated in sequence.

Application

Git-hosted JSON rules: per-action-class thresholds, minimum confidence scores, approval requirements, and an annotation any team can put on any resource to opt out permanently.

Control plane

Composite Resource Definitions declare exactly which fields the agent may patch and the permitted value ranges, drawn from a SKU catalogue maintained by the FinOps team in Git.

Cloud provider

Azure Policy and GCP Organisation Policy enforce guardrails at the provider level, entirely independently of the agent and of anything the agent believes.

The agent never writes a cloud API call. It patches a YAML field, and three independent layers have to agree before that patch is allowed to exist.

HOW IT EARNED AUTONOMY

Trust was the deliverable, not the feature set.

The most important decision in the system was not a technology choice. It was refusing to let the agent act for the first five months.

PHASE 1

Observe

Collection only. The agent produces no actions.

PHASE 2

Analyse

Recommendation only. Weekly digests to the FinOps team.

PHASE 3

Act

Auto-remediation, one action class at a time.

PHASE 4

Optimise

Seasonal patterns, cross-cloud placement, commitment proposals.

WHAT OPENS EACH GATE

14 days stable, under 0.1% loss	80% validated across 30 days	Zero false positives, 14 to 30 days	Six months of metrics history

Production databases are permanently recommendation-only. There is no phase in which that changes.

WHAT WE ARE TARGETING

Numbers, with their caveats attached.

30 to 40%

Storage

Automated hot to cool to cold to archive lifecycle, no manual rules

40 to 50%

Dev databases

Pause at 22:00, resume at 07:00, extended across weekends and never applied to production

25 to 45%

App tiers

p95 downgrade, only after CPU stays under 20% for 72 hours

30 to 55%

Commitments

Agent-surfaced proposals, always reviewed by a human before purchase

Read these as targets, not as banked savings.

They are per-category ranges from an implementation currently at Phase 2 of 4, and they will be presented that way in the session. The part of this worth forty minutes of your time is the trust protocol and the failure modes, not the percentages.

THE PART MOST TALKS LEAVE OUT

What went wrong on the way.

A session that only describes the working system is an advert. These are covered honestly, with what each one cost and what it changed.

- Transient spikes read as sustained load, and what a 72 hour window fixed that a 24 hour window did not.
- Confidence scores that were confidently wrong, and why a score alone is never a sufficient gate.
- Reconciliation fighting a human operator, which is exactly as unpleasant as it sounds.
- Approval fatigue: the fastest way to make a FinOps team rubber-stamp an agent is to ask them too often.
- The action classes we tried to automate and then permanently moved back to recommendation-only.

Each of these changed the architecture. Several of them are the reason a gate exists where one did not before.

IF YOU COME TO THIS SESSION

What you should be able to do afterwards.

A pattern you can copy

The collect, analyse, check, act loop with the specific open source components at each stage, and why each was chosen over the obvious alternative.

A reason to prefer reconciliation

Why a continuously reconciling control plane and a triggered provisioner are not interchangeable once software is allowed to act on its own conclusions.

A trust protocol, not a trust feeling

Four phases with hard gates, and the argument for why autonomy has to be earned in measurable increments rather than granted at go-live.

Where to stop

The classes of change that should never be automated at any confidence level, and how to decide which yours are.

Useful if you run a FinOps practice, operate a multi-cloud platform, or are deciding how much of your infrastructure an agent should be allowed to touch.

THE SESSION

The Bill That Fixes Itself

Trustworthy Agentic FinOps "with Legs"

Friday 9 October · 11:55 to 12:35 CEST (GMT+2)

Pratik Mohapatra

Confluentis Consulting

Confluentis is a cloud-native engineering consultancy working on CNCF-aligned platform tooling and AI-assisted infrastructure automation. The system in this talk is the third in a line that began with percentile-based Kubernetes rightsizing and a cost anomaly pipeline, and each one exists because the previous one stopped short of acting.

confluentis.co.in