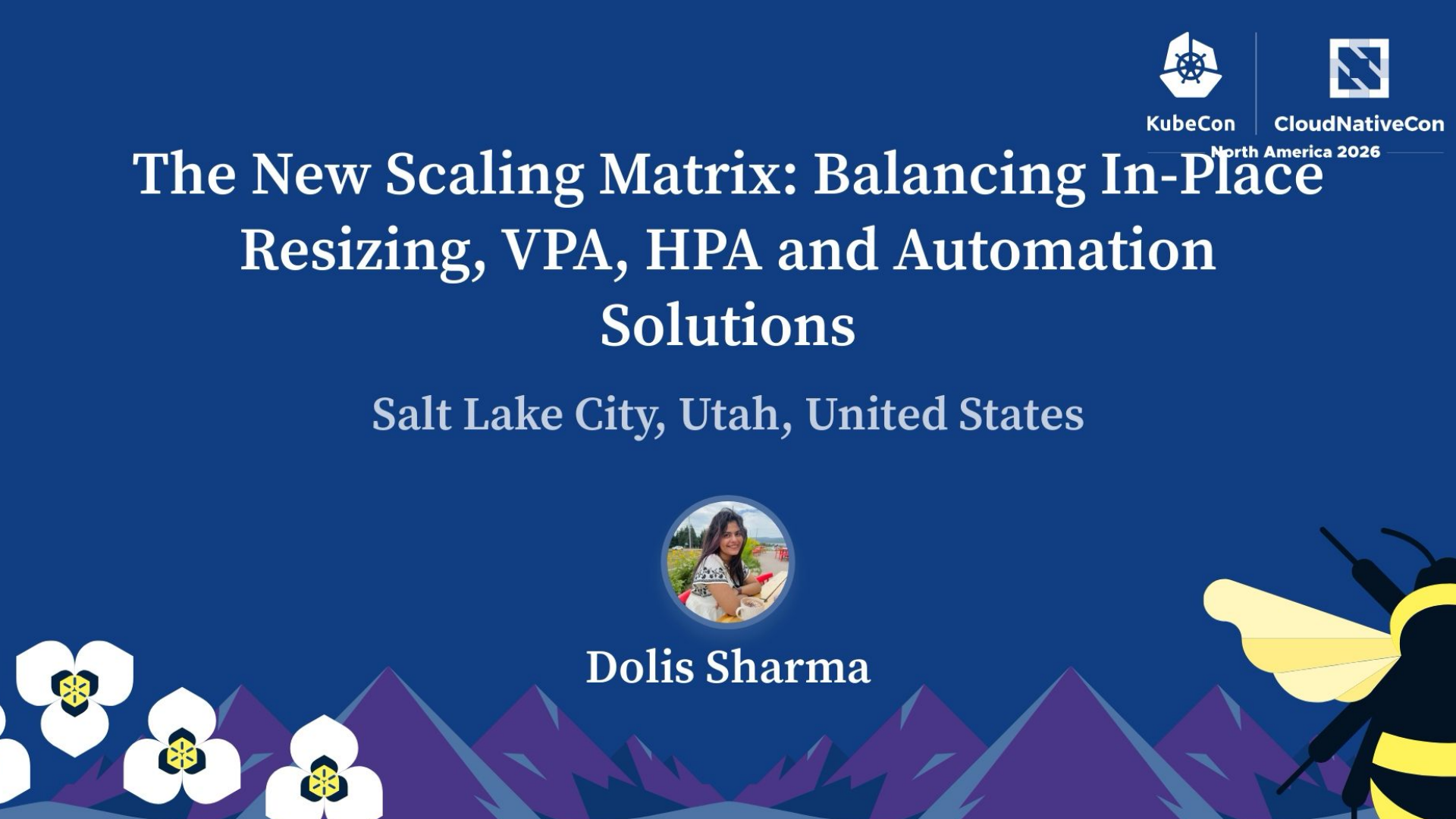


The New Scaling Matrix: Balancing In-Place Resizing, VPA, HPA and Automation Solutions

Salt Lake City, Utah, United States

Dolis Sharma

The classic toolkit still works

- **Label-driven cleanup:** Team label, ttl label and a few cleanup policy can do the job.
- **Off-hours scale-down:** kube-green, kube-downscaler and the KEDA cron scaler all handle this.
- **HPA / KEDA:** adds replicas based on demand. Use event signals (queue depth, RPS) rather than CPU alone where you can.
- **VPA:** right-sizes requests. Start in Off mode to get recommendations only, then move up gradually
- **Karpenter / Cluster Autoscaler:** consolidation and spot instances turn right sized pods into fewer nodes.

Where it breaks, and how it evolved

The HPA/VPA race:

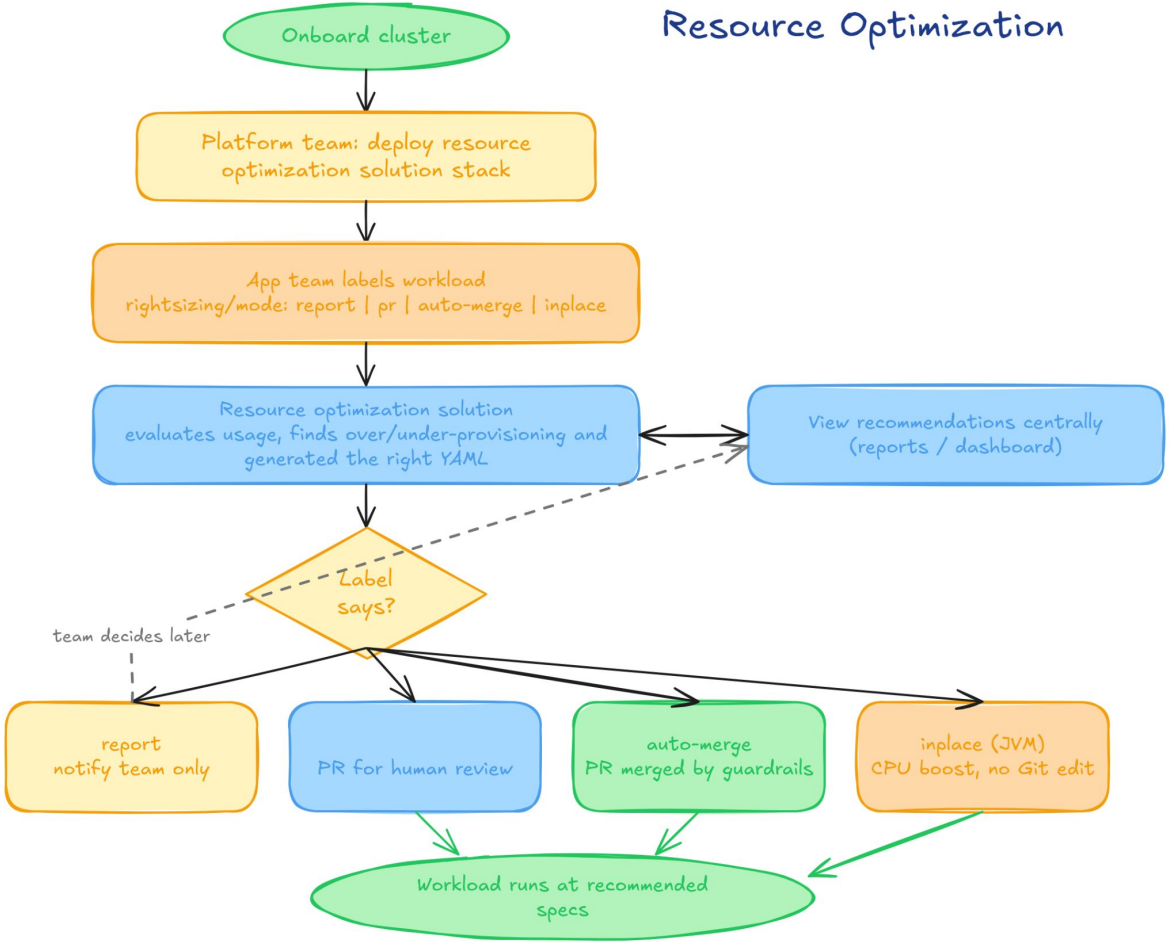
- On the same resource metric, they feed each other into a scaling loop. The safe patterns are:
  - HPA for stateless and VPA for stateful.
  - Do not run HPA and VPA on the same resource metric.
  - Keep VPA in recommender mode for GitOps-safe.

In-place resize is a targeted tool, not a default:

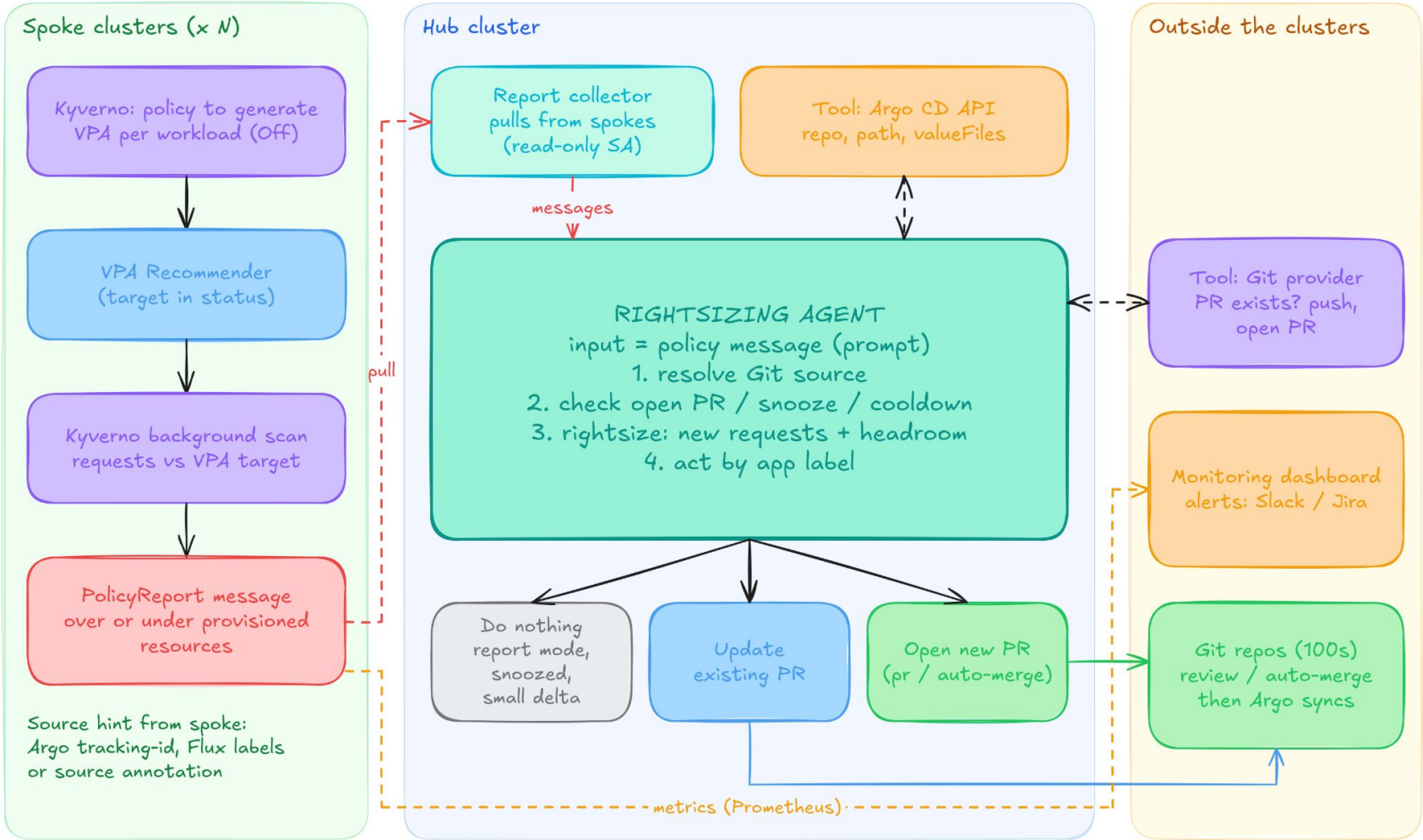
- Best fit: JVM apps. In-place resize went GA in 1.35 Give extra CPU at startup, then scale it back down without a restart.
- Resize CPU in place; restart for memory. The JVM sets its heap at startup, so a memory change needs a restart.
- Don’t enable it cluster-wide. Live resizes cause drift everywhere and do more harm than good.
- Make it GitOps-safe:
  - Opt in per workload with a label.
  - Have Argo CD ignore resource diffs on opted-in apps only.

Rightsizing as an Opt-In Service

- The platform team defines the lanes once, at the org level, and enforces them with policy.
- App teams choose their lane with a single label in their own manifests: *report, pr, auto-merge, or inplace*.
- Every change goes through Git, so GitOps stays the source of truth.
- The platform team provides the automation, and app teams decide how much of it they trust.

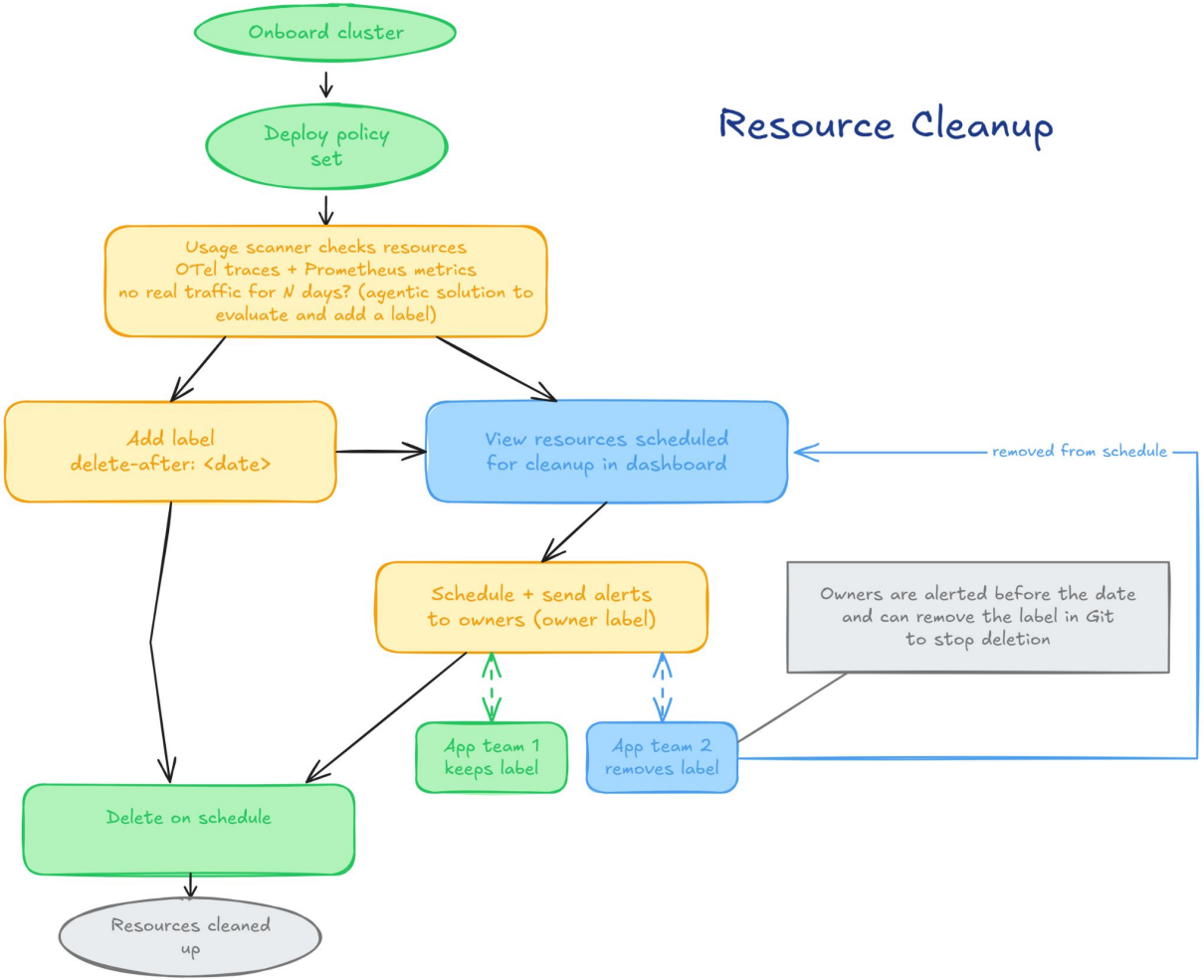


Low-level design: the rightsizing agent lives in the hub



Usage-Aware Cleanup

- **Usage decides what's idle, not age.** OTel traces and Prometheus metrics show which resources have had no real traffic for N days.
- **Label first, delete later.** Idle resources get a `delete-after: <date>` label, and owners are alerted with time to opt out by removing it in Git.
- **Deletion goes through Git.** A delete PR on the scheduled date lets Argo CD prune the resources, so nothing comes back and every step can be audited.



Why none of this may not work for agents?

Agents are not apps

- Agents are long-lived, bursty and idle most of the time: waiting on a person or an LLM.
- One pod per agent turns idle time into the default cost.
- HPA/VPA react in seconds to minutes; interactive agents need milliseconds and must keep their state.

Solution: Agen substrate

Problem	Why Kubernetes struggles	What Agent Substrate does
Idle agents hold pods	Agent-per-pod means you pay for compute while the agent waits on a human or LLM.	Suspends idle Actors and gives the Worker slot to an Actor that is ready to work.
API server in the hot path	The API server is not built for millions of create/update writes for short-lived sessions.	Uses Kubernetes for what it is good at (pools, CRDs) and a separate control layer for suspend/resume.
Pod start takes seconds	Eventually consistent scheduling converges in seconds; interactive agents need milliseconds.	Workers are already running; resume restores a snapshot onto a warm Worker.
State is hard	Attaching/detaching millions of volumes is not a Kubernetes strength.	Snapshots an Actor's state to object storage (S3, GCS) and restores it on any Worker.

Agent Substrate

- **An open-source runtime for AI agents on Kubernetes.** It's framework-agnostic, and kagent uses it as the runtime under every agent.
- **Many agents share a few pods.** Agent sessions ("Actors") run on a small pool of pre-warmed pods ("Workers"), using the fact that agents are idle most of the time. One Worker can host up to about 1,000 Actors.
- **Idle agents are saved, then woken in milliseconds.** An idle agent is snapshotted to object storage (S3/GCS) and its capacity is released. When traffic arrives, it resumes on any free Worker, with no new pod to start and no cold start.
- **Every agent runs in its own sandbox.** It uses gVisor or a micro-VM, and egress goes through a gateway that can inject credentials.



KubeCon CloudNativeCon
North America 2026

Thankyou!!

