Unlocking Just-in-Time Resource Optimization with In-Place Pod Resize

CruiseKube

Speaker Introduction

Shubham Rai

Engineering @ TrueFoundry

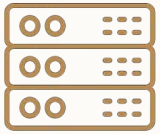
- B.Tech from IIT Guwahati.
- 9+ years total work experience.
- Core contributor @ KubeElasti.

Raman Tehlan

Cloud Native Consultant @ Zurich Lab

- Creator of Vxplain (visualization tool for coding agents).
- 5+ Years in Distributed Computing
- Organizer of PyData Delhi.

Why Kubernetes Clusters Stay Expensive



Defensive Padding

Fear of **CPU throttling and OOMKills** leads teams to over-request resources, creating significant waste.

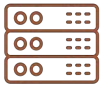
Operational Lag

Manually **editing YAML files** is too slow for dynamic workloads

Inefficient Scaling

Inflated requests result in **underutilized nodes**, forcing cluster autoscalers into inefficient and costly decisions.

Cluster Autoscaling Alone Does Not Solve Pod-Level Waste



Node Layer: Cluster Autoscaler / Karpenter

- **Responsibility:** Adds or removes nodes based on **aggregated pod requests**.
- **Limitation:** Operates on the demand signal it is given; cannot fix **inaccurate or inflated pod requests**.

Pod Layer: CruiseKube

- **Responsibility:** Fixes the demand signal itself by **right-sizing individual pod CPU and memory requests**.
- **Benefit:** Provides **high-quality, accurate input** to the node autoscaler, improving overall cluster efficiency.

CruiseKube vs VPA

Comparison: CruiseKube vs. VPA

Dimension	CruiseKube	Kubernetes VPA
Optimization Model	Continuous, feedback-driven control loop	Periodic analysis and recommendation
Unit of Decision	Individual pod, in node context	Workload template
View of the Cluster	Node-local, with shared capacity	Pod isolated from node context
Time Horizon	Short-term, adaptive	Long-term, historical

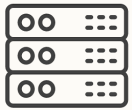
Key Advantages

Dynamic Resource Sharing
Allows pods on the same node to share headroom instead of duplicating it.

Correction over Prediction
Relies on frequent real-time correction rather than long-horizon prediction.

Efficiency & Safety
Preserves reliability through conservative memory handling and priority-based protection.

Why Closed-Loop Right-Sizing Is Feasible Now



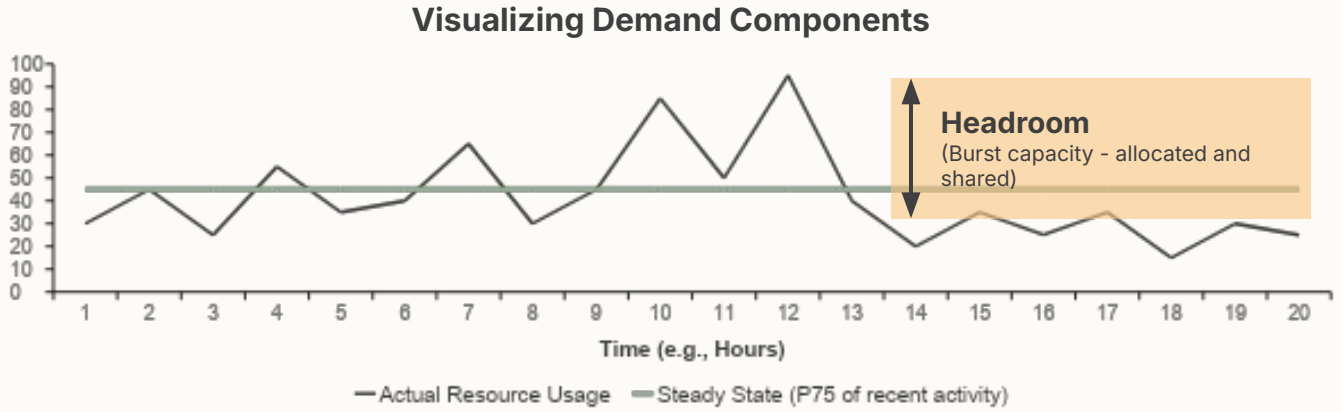
In-Place Pod Resource Updates (GA in Kubernetes v1.35)

Allows resizing of CPU and memory for running pods **without a restart**. This is critical for **minimizing disruption** to stateful services and long-running jobs.

Pressure Stall Information (PSI) Metrics (Beta in Kubernetes v1.34)

Provides **feedback** on CPU contention when no limits are specified. Used to continuously improve recommendations

Optimizing with Node Level Context



Steady State

Typical usage, calculated from the P75 of recent activity.

Peak Usage

Maximum observed usage from recurring historical patterns.

Headroom

Difference between peak and steady-state; burst capacity without constant allocation.

How CruiseKube works

1. Observe Collect real-time and historical metrics **via Prometheus.**

2. Learn Analyze usage patterns to understand **workload behavior.**

3. Recommend Compute **optimized CPU and memory requests.**

4. Apply Safely implement changes at **pod admission and runtime.**

5. Re-observe Repeat the cycle to adapt to **ongoing environment changes.**

Two phases of optimization

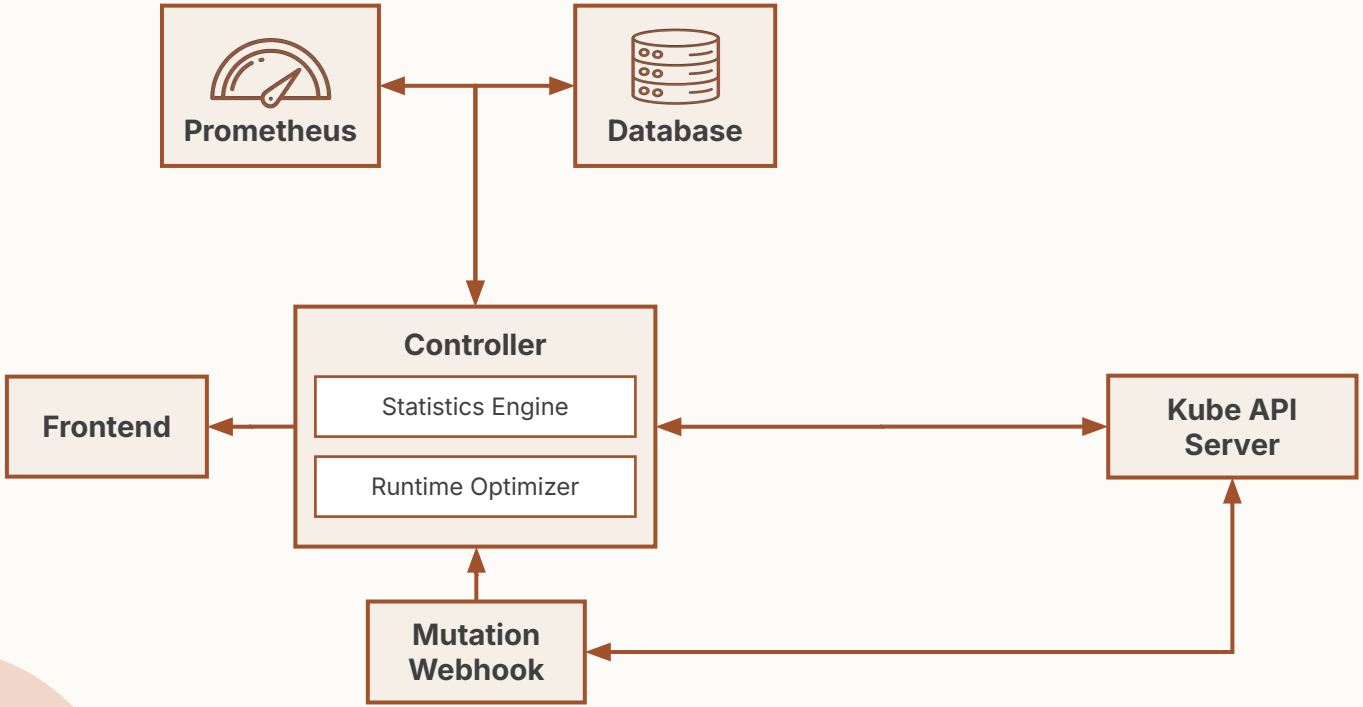
Phase 1: Admission Webhook
(Goal: Safety and Placement)

- Intercepts pod create requests.
- Applies safe, **historically-informed resource requests**.
- Ensures reliable pod scheduling.

Phase 2: Runtime Optimizer
(Goal: Efficiency and Optimization)

- Continuously observes running pods on specific nodes.
- Uses node-aware context (like available headroom) for **in-place updates**.

Components of CruiseKube



Optimization With Guardrails

Recommend Only Mode

CruiseKube only suggests changes, requiring explicit 'Cruise Mode' activation for applications.

Priority-Aware Eviction

Define workload criticality (Low to High). Less critical services are evicted first if needed

OOM & Overload Feedback

Continuous monitoring for Out-of-Memory (OOM) and node pressure. Automatically halts optimization on affected nodes to maintain system stability.

Dashboard

Automation Control

Seamlessly toggle workloads between 'Recommend' and 'Cruise' (auto-pilot) modes.

Resource Monitoring

Compare current vs. recommended resources for every workload.

Policy Management

Define granular workload priorities and exclusion policies to fit business needs.

Savings Tracking

Gain clear insights into estimated monthly cost savings across the cluster.

Live Demo: From Recommendation to Policy Control

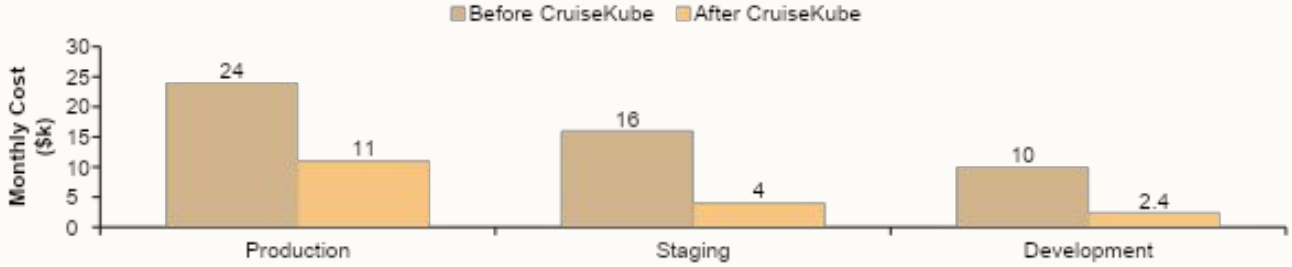
Achieves Average 65% Reduction in CPU Costs by Eliminating Waste

By safely matching resource requests to actual usage in real-time, CruiseKube significantly reduces over-provisioning and lowers cloud spend.

Sample Cost Savings Breakdown (Quarterly)

Environment	Metric	Before CruiseKube	After CruiseKube	Savings
Production	Avg. CPU Requests	1,200 cores	550 cores	**-54%**
	Monthly Cost	$24,000	$11,000	**$13,000**
Staging	Avg. CPU Requests	800 cores	200 cores	**-75%**
	Monthly Cost	$16,000	$4,000	**$12,000**
Development	Avg. CPU Requests	500 cores	120 cores	**-76%**
	Monthly Cost	$10,000	$2,400	**$7,600**

Monthly Costs Before vs. After CruiseKube



Key Takeaway: The largest savings are often found in non-production environments where utilization is spiky and over-provisioning is common.

Limitations

Use Cases that Fit Well

- **Stateless Services:** Best for replicated services that don't maintain state.

- **Batch Processing:** Ideal for short-lived, intensive computing jobs.

- **Dev / Staging:** Perfect for environments where cost-saving is a priority.

Use Cases that Need Caution

- **Latency Sensitive:** Critical services where even millisecond delays matter.

- **Stateful Apps:** Databases and apps sensitive to any disruption.

- **Complex Scaling:** Workloads with intricate HPA interactions needing careful testing.

Contribute & Connect with the Community

CruiseKube is a **community-driven project**. We welcome **contributions of all forms**, from bug reports to new features.

GitHub Repository

- **Star our repo:**
  github.com/truefoundry/CruiseKube

Join the Conversation

- **Discord:**
  https://discord.gg/Dqek4xJa3N