VPA in Production

Kubernetes 1.35 fixed the restart tax.

Here are the failure modes it didn't fix.

>_ Modern Infrastructure

What In-Place Resize Actually Fixed

Before 1.35

Resource change → **pod restart**
Restart = disruption for stateful workloads

VPA automation: disabled for safety

After 1.35

Resource change → **cgroup mutation**
Container keeps running. No restart.

VPA InPlaceOrRecreate: viable

That's 1 of 5 failure modes.

Here are the other 4.

Node Capacity Constraints

What happens:

- InPlaceOrRecreate has a silent fallback path
- Node full when resize is attempted
- VPA evicts the pod — no warning
- Pod restarts on a new node
- Looks like normal rescheduling

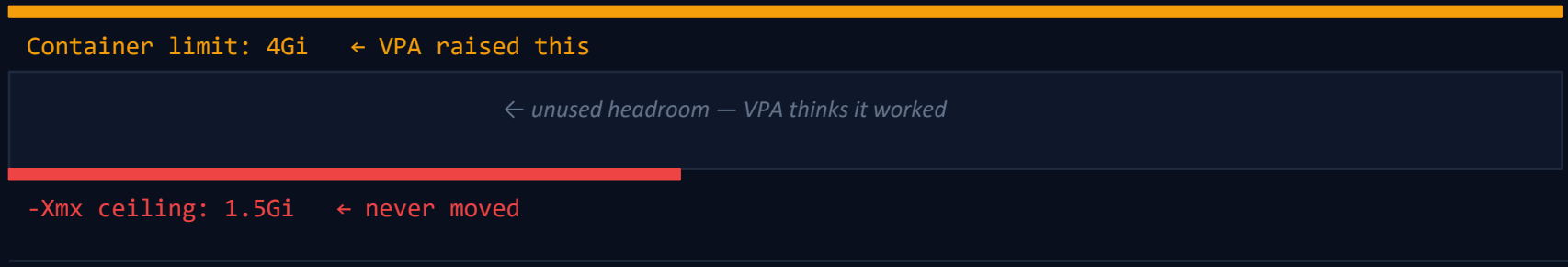
>_ Diagnostic

```
kubectl describe vpa <name> -n <ns>
kubectl get events --field-selector
  involvedObject.name=<pod>
```

Look for EvictedByVPA events
+ pod landing on new node each cycle

Fix: Node headroom ≥ 20% before enabling automation. Tight clusters will trigger eviction fallback.

The JVM Heap Ceiling



-Xmx is a startup parameter. Raising the cgroup limit doesn't raise the heap ceiling.

VPA thinks it fixed the problem. GC pressure continues.

Fix: resizePolicy: RestartContainer for memory on JVM workloads.

Memory Shrink Is Dangerous

What VPA sees:

Observed consumption: 1.5Gi
Lower bound target: 1.8Gi

Recommendation:
reduce memory limit

What the allocator holds:

Peak footprint retained: 2.8Gi
glibc / JVM GC won't release memory proactively

Gap = OOM kill zone

⚠ **OOM kill zone — gap between VPA target and allocator retained footprint**

The kubelet makes a best-effort attempt to prevent OOM kills. Best-effort is not a guarantee.

Fix: Disable downward memory automation on allocator-heavy workloads. Set minAllowed conservatively.

Scheduler Fragmentation

In-place resize keeps the pod on its node — and consumes more of it.

01

Pod resized in-place

Consumes more CPU/mem on current node

02

Node capacity shifts

Other pods no longer schedulable there

03

Pending pods appear

Enough aggregate capacity — no single node fits

Same fragmentation signature as CPU fragmentation failure — pods pending, nodes not full in aggregate.

Fix: VPA + cluster autoscaler coordination. Headroom settings must account for in-place resize growth.

VPA Recommendation Drift

8-day observation window

Low, flat consumption
→ VPA recommends: scale down

High-load event

Outside the window
→ OOM kill or
CPU throttle

Default window: 8 days

Batch jobs, weekly cycles, monthly spikes all fall outside this window.

The recommendation was technically correct — for the wrong load profile.

Fix: Set minAllowed conservatively. Extend observation window for variable workloads.

>_ Pre-Automation Checklist

- → Node headroom ≥ 20% before enabling
- → resizePolicy: RestartContainer for memory on JVM workloads
- → Disable downward memory automation on allocator-heavy workloads
- → Extend observation window for variable load patterns
- → Start with CPU automation only — lower risk than memory
- → Run Off mode for 2+ weeks before enabling automation
- → Test per workload class in non-production first

Architect's Verdict

1.35 solved the restart problem.

The restart was 1 of 5 failure modes.

- Start in Off mode. Observe for 2+ weeks.
- Validate resize behavior per workload class.
- Bound recommendations with minAllowed + maxAllowed.
- Treat memory shrink as higher risk than memory growth.

The teams that get the most out of this feature understood why VPA was disabled — and fixed those reasons deliberately.

Full breakdown on the blog

Diagnostics. YAML configs. Per-workload guidance.

Vertical Pod Autoscaler in Production:

In-Place Resize Works — Until It Doesn't

Link in first comment →

---

rack2cloud.com

`>_ Modern Infrastructure & IaC`