Beyond the Cluster Boundary: Sharing GPUs at Scale

Tim Killian · Snap Inc.

KubeCon 2026



© SNAP INC. 2026

Infrastructure at a glance

Thousands

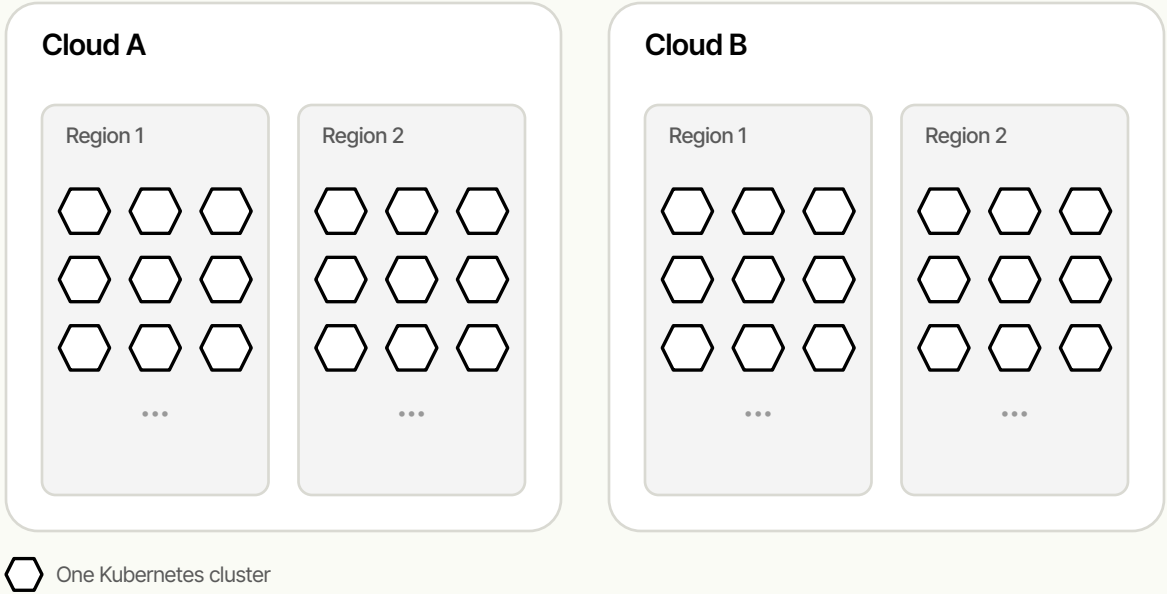
of Kubernetes clusters

100,000+

VMs across the fleet

Multiple

cloud providers and regions

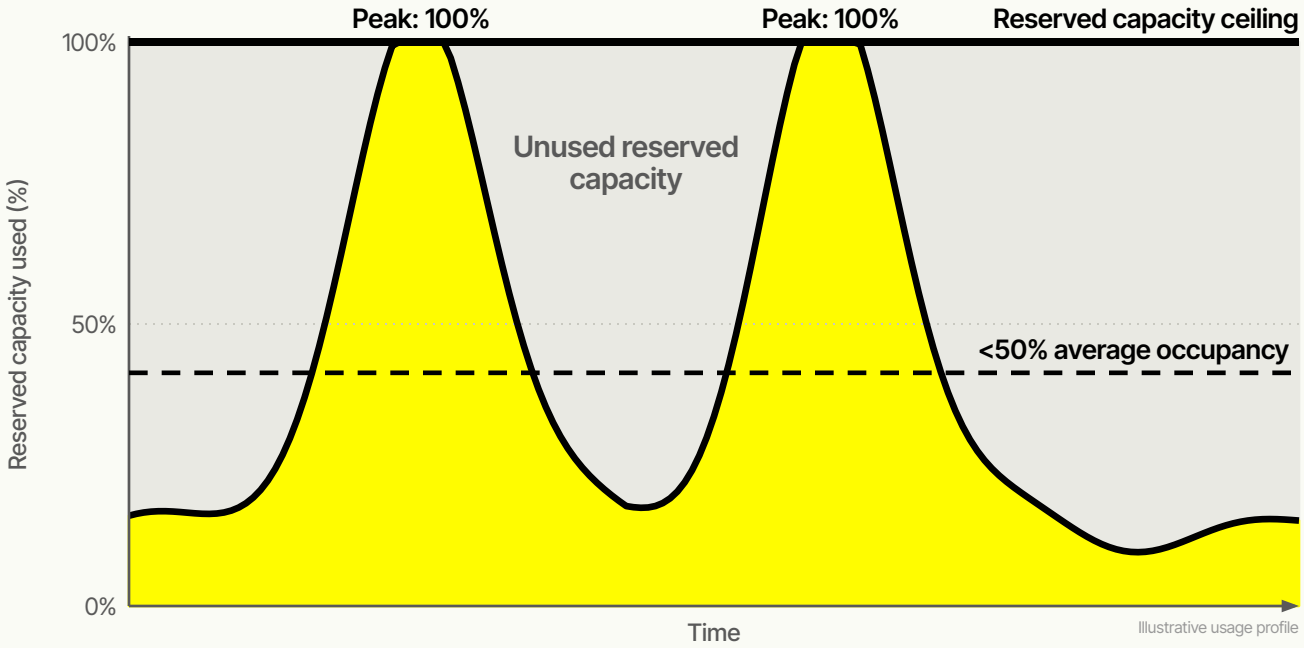


The fleet historically relies on on-demand capacity, but we must reserve for constrained machine types

© SNAP INC. 2026

Inference demand: High peaks, low troughs

Guaranteed capacity. Paid for whether busy or idle.



100%

Peak usage of reserved capacity

<50%

Average occupancy

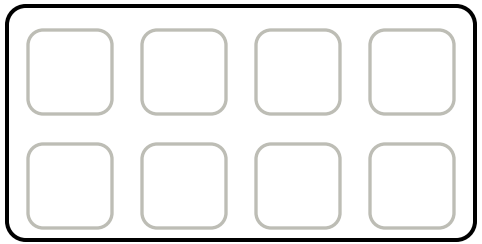
We reserved for peak demand, but paid for many machines that sat idle off-peak.

© SNAP INC. 2026

Cloud reservations let us share, but not reclaim

1 Owner is idle

Owner's cloud reservation

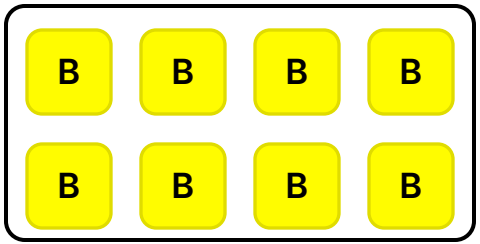


Reserved machines sit idle

Off-peak, owner capacity goes unused.

2 Borrowers use them

Owner's cloud reservation

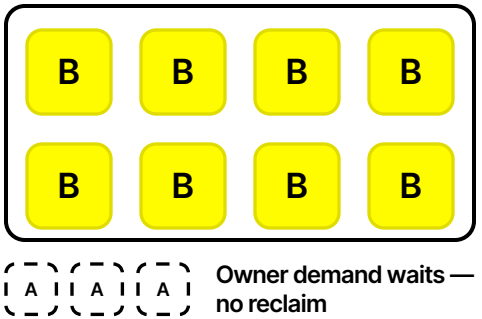


Borrowers run on reserved machines

Let other workloads borrow while the owner is idle.

3 Owner returns

Owner's cloud reservation



Reservation sharing alone does not reclaim the machines when the owner returns.

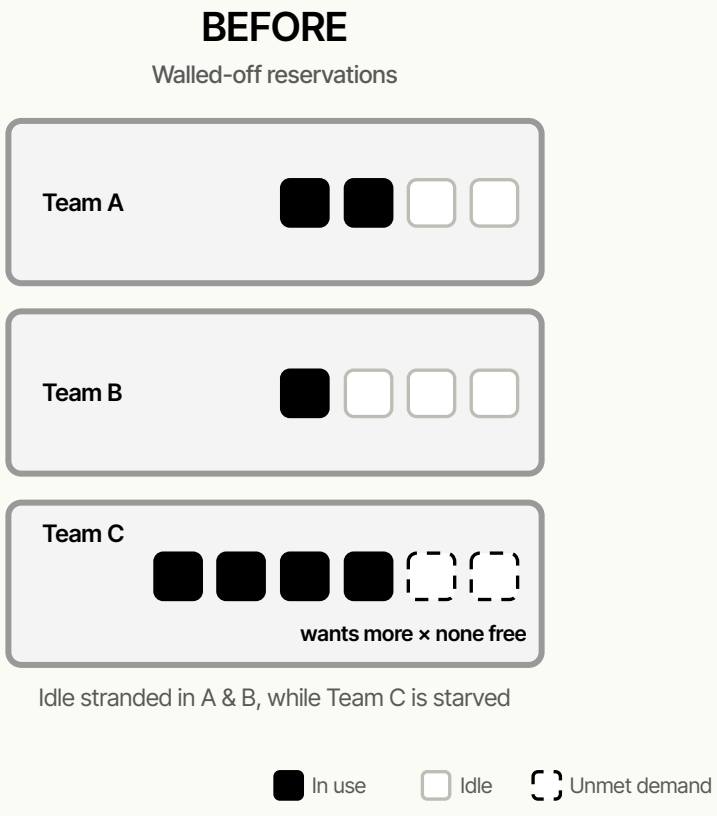
B Borrower workload running

Idle reserved machine

Owner demand waiting

© SNAP INC. 2026

Idle capacity, but nowhere to run



© SNAP INC. 2026

The challenge: safe sharing

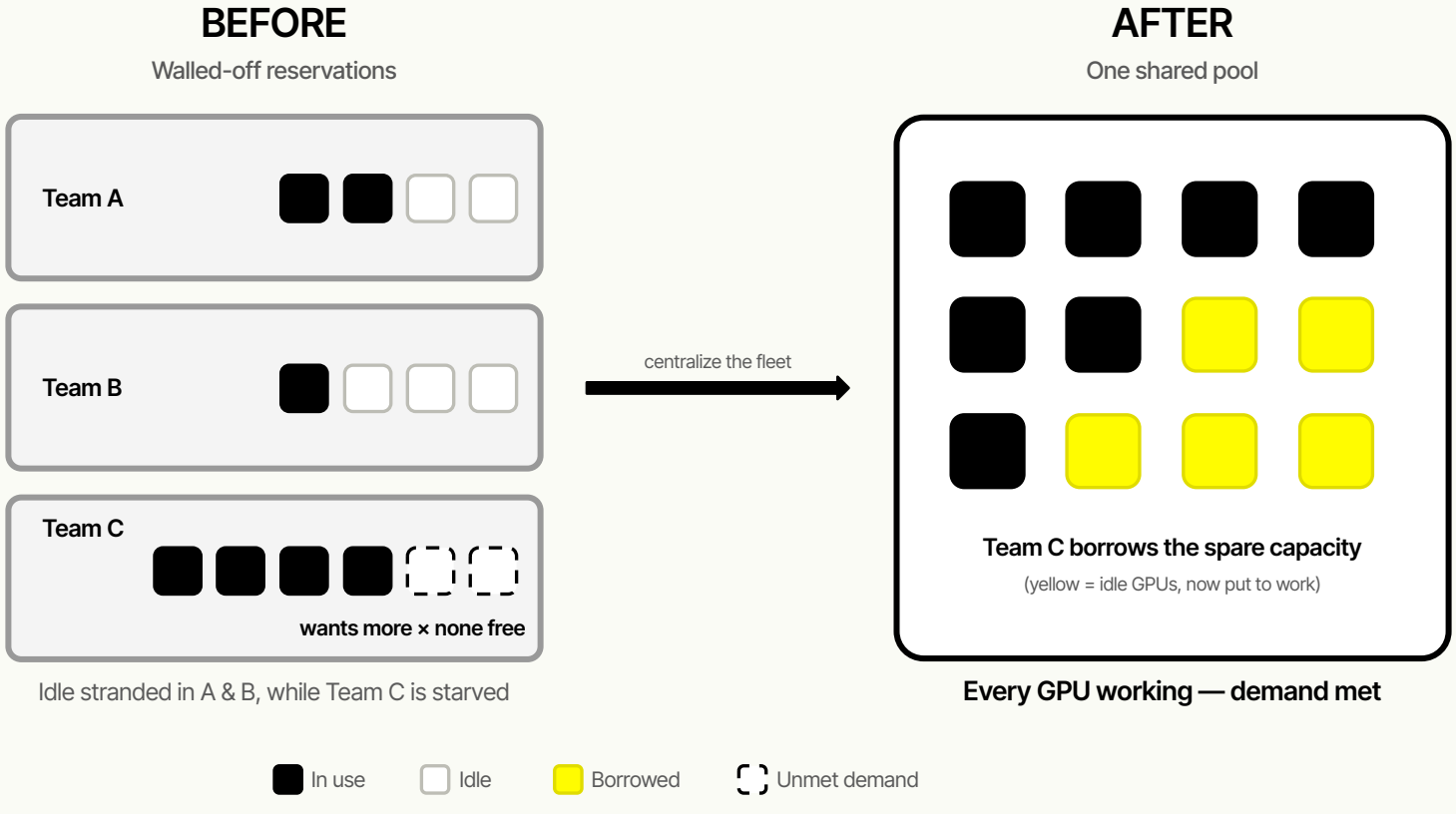
Share idle GPUs.

Guarantee owners can reclaim them.

Even when a team’s workloads span multiple Kubernetes clusters.

© SNAP INC. 2026

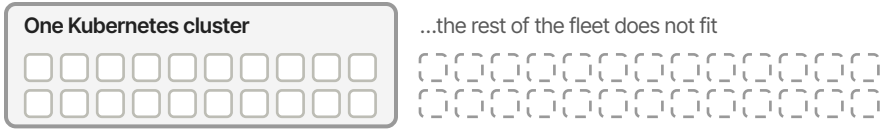
Our answer: centralized capacity management



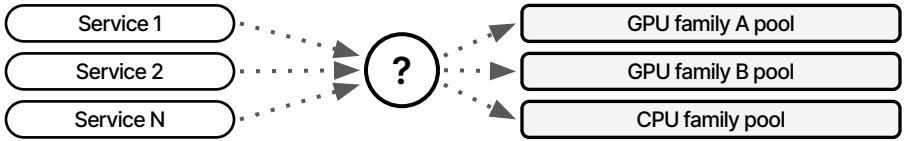
© SNAP INC. 2026

Why not deploy services directly into the shared pools?

1. **One cluster cannot hold the fleet.**



2. **Borrowers change as services autoscale.**



3. **Placement includes machine fallback.**



Keep service deployment separate from capacity placement.

© SNAP INC. 2026

Centralize constrained capacity. Preserve teams' operational models.

Virtualization

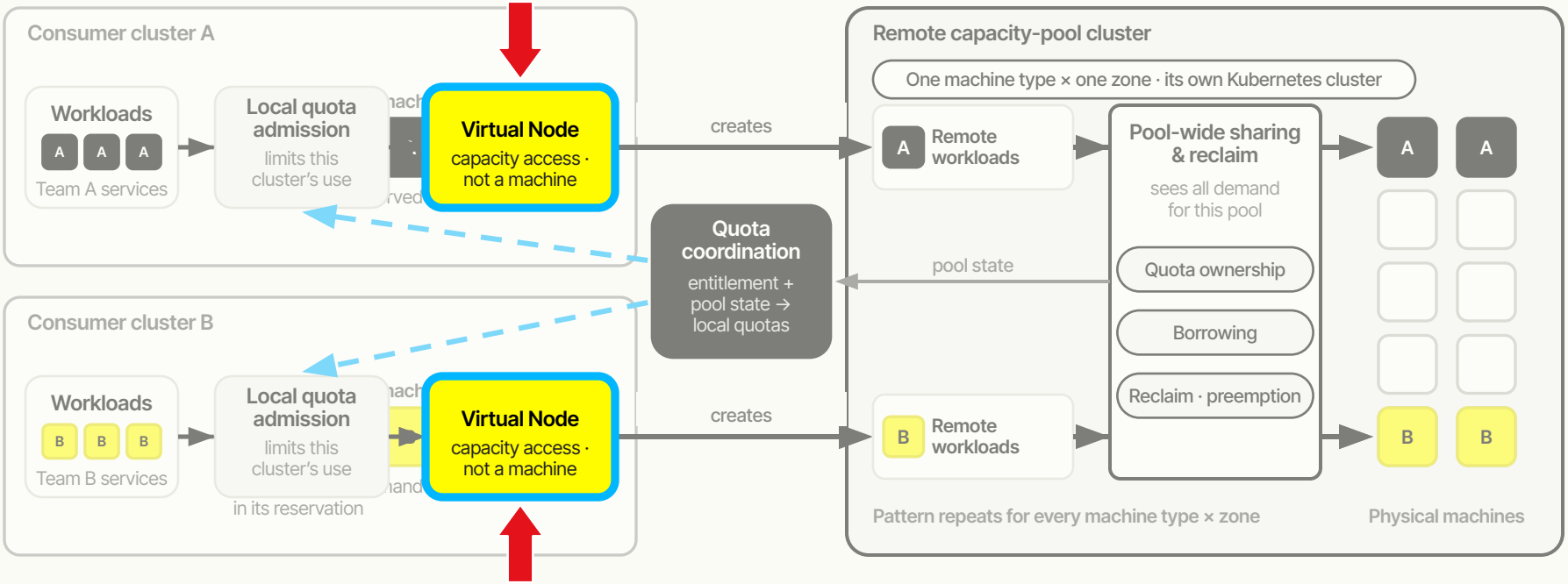
How workloads access pooled capacity.

Workload Quotas

Who can use that capacity, and how much.

© SNAP INC. 2026

From stranded capacity to one shared pool

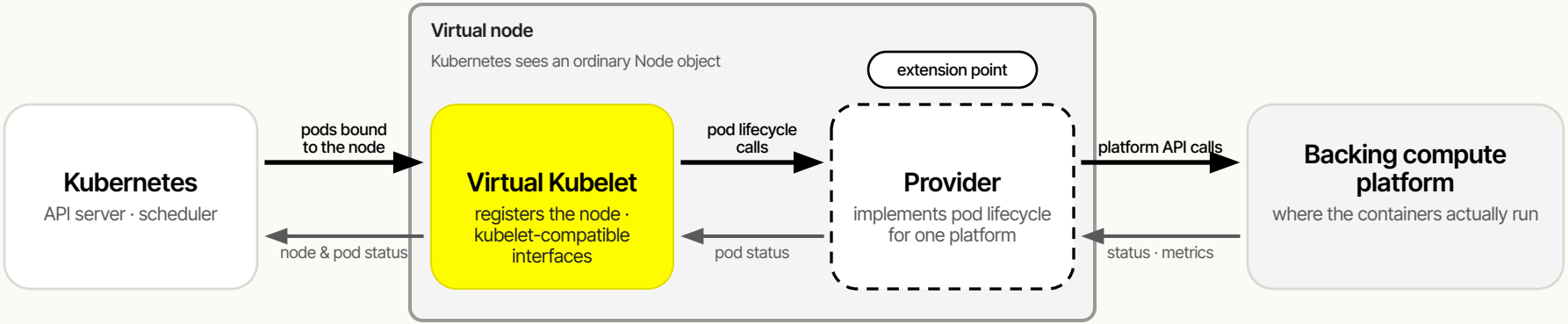


NEXT · VIRTUAL NODES — First, how did we implement the virtual nodes?

A Team A workload · B Team B workload · Idle machine · Execution path · Local quota advertisement

© SNAP INC. 2026

What is Virtual Kubelet?



1 **A framework for a virtual node**

Something other than a physical worker can appear to Kubernetes as a node.

2 **Kubelet-compatible interfaces**

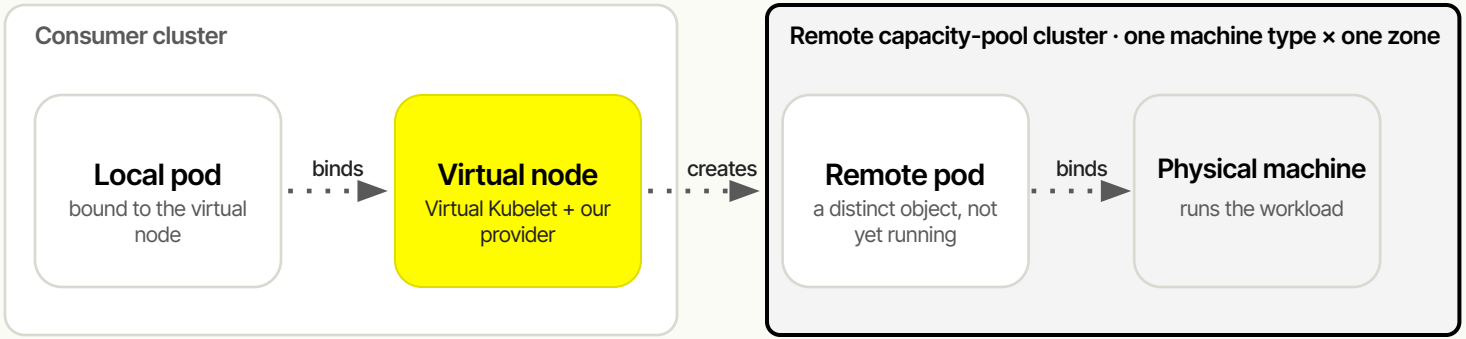
Kubernetes interacts with it the way it interacts with any node.

3 **Providers implement pod lifecycle**

Create, delete and status operations run against the backing platform.

© SNAP INC. 2026

Virtual Kubelet exposes a capacity pool as nodes



1

Register a virtual node in the consumer cluster

2

A local pod binds to that virtual node

3

Our provider creates the corresponding pod in the remote capacity-pool cluster

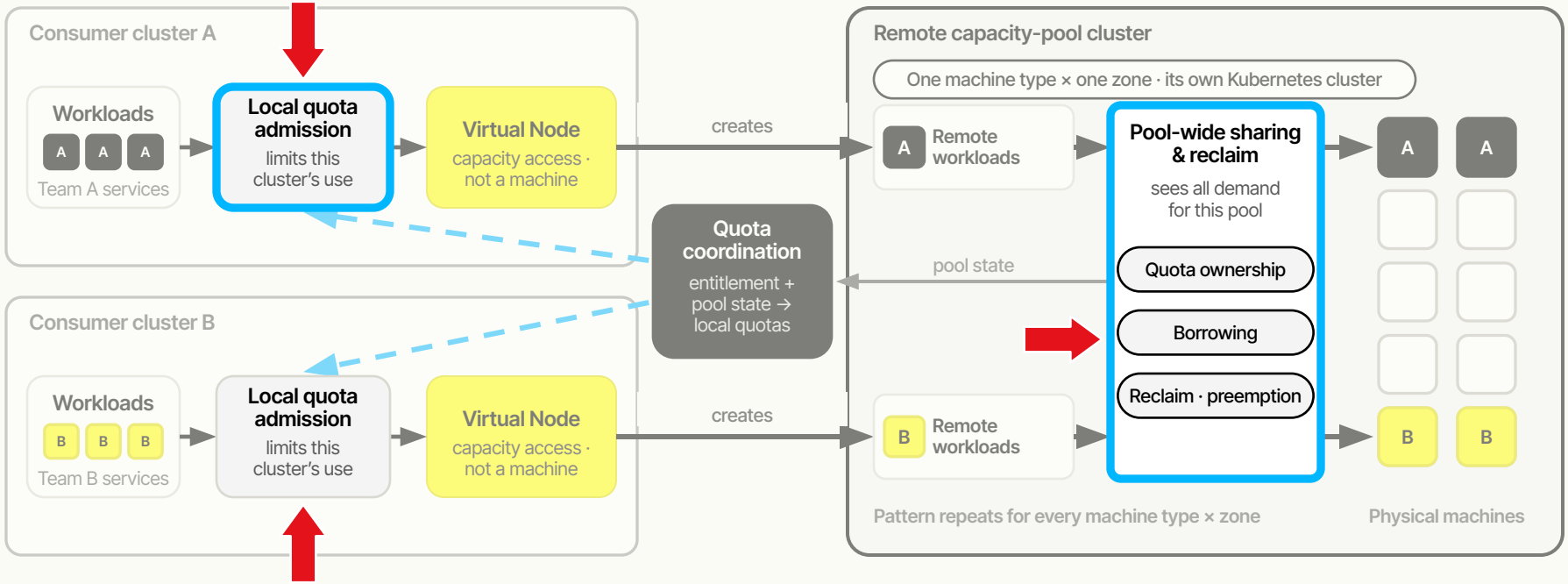
4

The remote pod binds to the physical machine

Virtual nodes give customer clusters access to the central capacity pools, but is “all or nothing”

© SNAP INC. 2026

Next: admission, sharing and reclaim

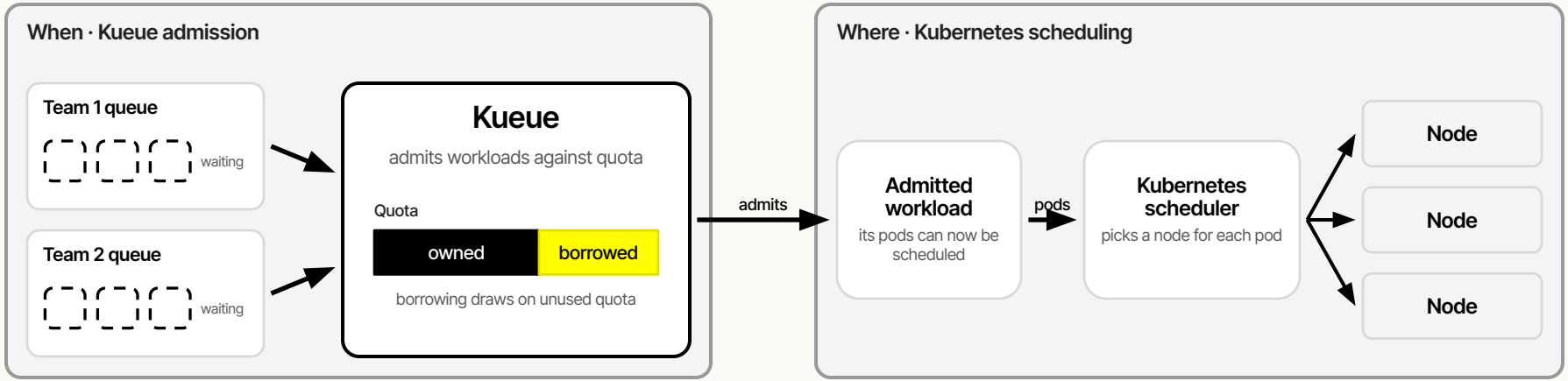


How do we enforce local quotas and arbitrate a shared pool?

A Team A workload · B Team B workload · Idle machine · Execution path · Local quota advertisement

© SNAP INC. 2026

What is Kueue?



1 Admission and quota management

Kubernetes-native: workloads wait until quota lets Kueue admit them.

2 Queues govern when quota is used

Workloads are submitted to queues, which order them for admission.

3 Borrowing and reclaim

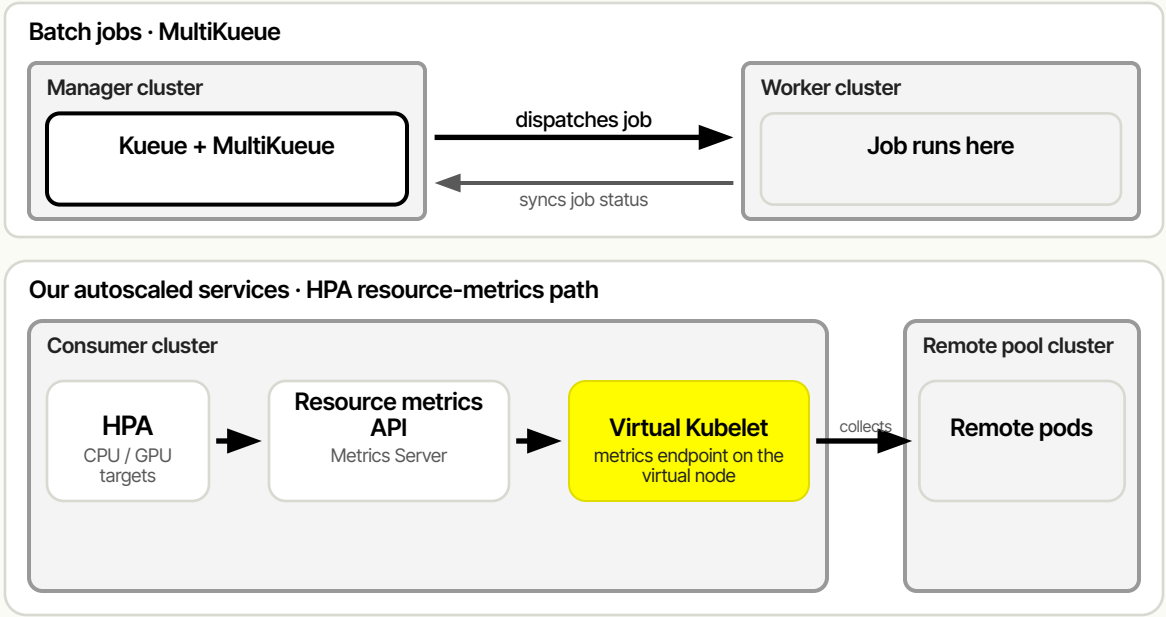
Sharing policies can lend unused quota; preemption policies can reclaim it.

Kueue decides when and where workload may run, but how does this work across cluster boundaries?

© SNAP INC. 2026

Why Not MultiKueue?

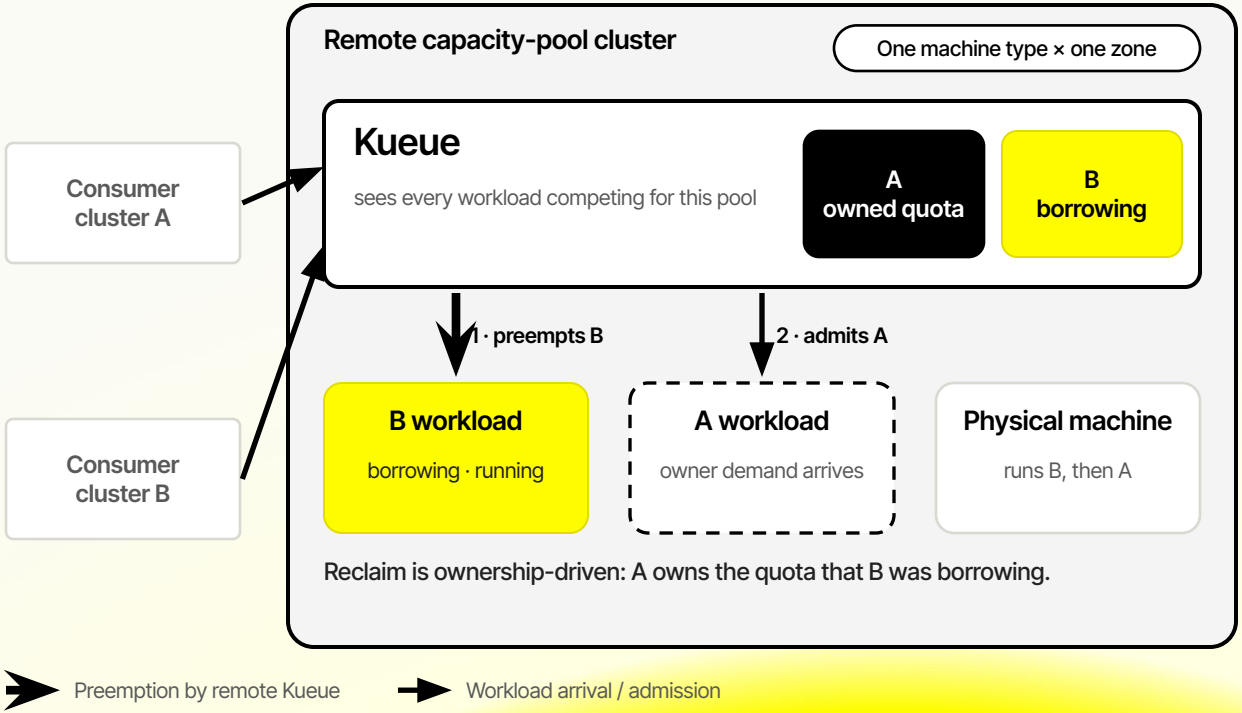
1. MultiKueue dispatches jobs from a manager cluster to worker clusters.
2. Our HPA depends on pod visibility and resource metrics in the consumer cluster.
3. Virtual Kubelet preserves that path.



> An option for batch, but not for our autoscaled serving path.

© SNAP INC. 2026

Kueue on the capacity pool has the whole picture



One cluster per pool

All workloads competing for this machine-type × zone pool arrive here

Owned vs borrowed

Kueue distinguishes A's owned quota from capacity B borrows

Borrow while idle

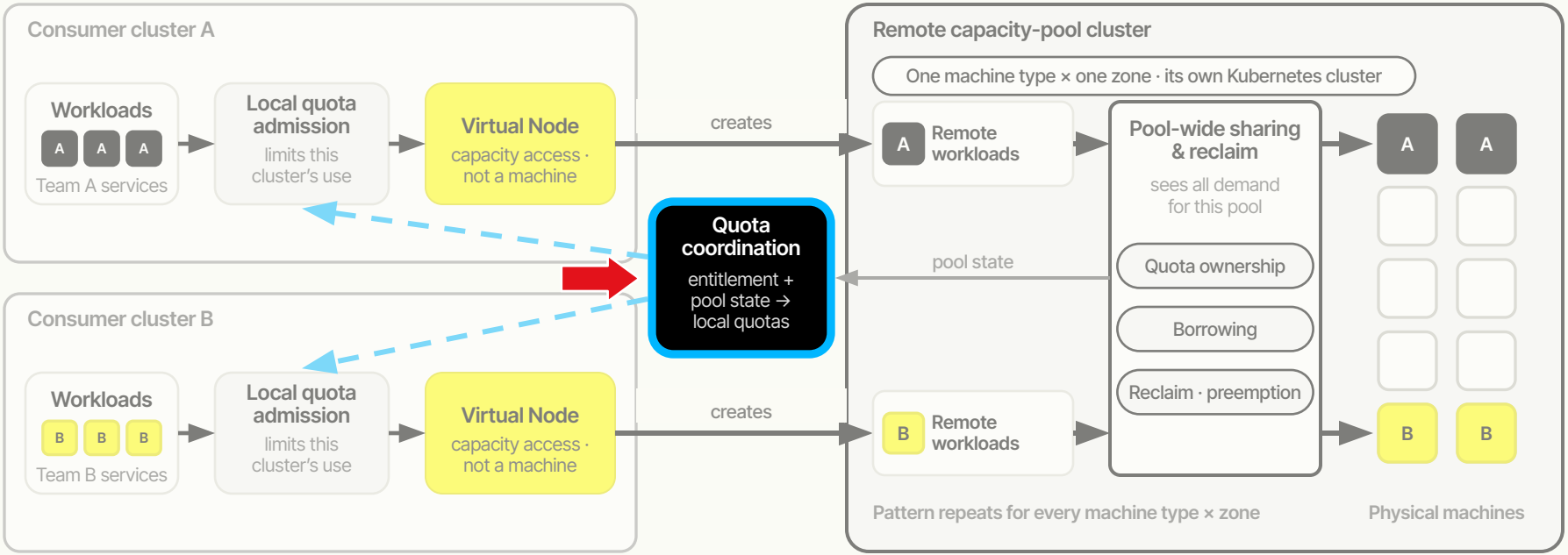
B uses the capacity while owner A is idle

Reclaim for the owner

When A's demand arrives, remote Kueue preempts B under its reclaim policy

© SNAP INC. 2026

Finally: keeping local quotas aligned



How do we decide what quotas to advertise on customer clusters?

A Team A workload · B Team B workload · Idle machine · Execution path · Local quota advertisement

© SNAP INC. 2026

1 + 1 = 1?

Owned quota

Entitlement: who owns the capacity

A	1 machine
B	0 machines

B owns nothing in this pool

Advertised locally

Lets consumer workloads reach the remote pool

A	1 machine
B	1 machine

Idle owner capacity advertised to both A and B

Running remotely

Actual consumption, accounted by remote Kueue

A	0 machines
B	0 machines

Remote pool: 1 physical machine, idle

Advertised quotas need not sum to physical capacity.

© SNAP INC. 2026

Borrowing leaves the owner's local quota available

A owns 1 machine; B owns 0 · all quantities in machines

1 A is idle

Reconciler advertises 1 machine to both A and B

	Owned quota	Advertised locally	Running remotely
A	1 machine	1 machine	0 machines
B	0 machines	1 machine	0 machines

Remote pool · Kueue · nothing running · 1 machine, idle

2 B borrows remotely

B uses its advertised quota and reaches the remote cluster

	Owned quota	Advertised locally	Running remotely
A	1 machine	1 machine	0 machines
B	0 machines	1 machine	1 machine borrowing

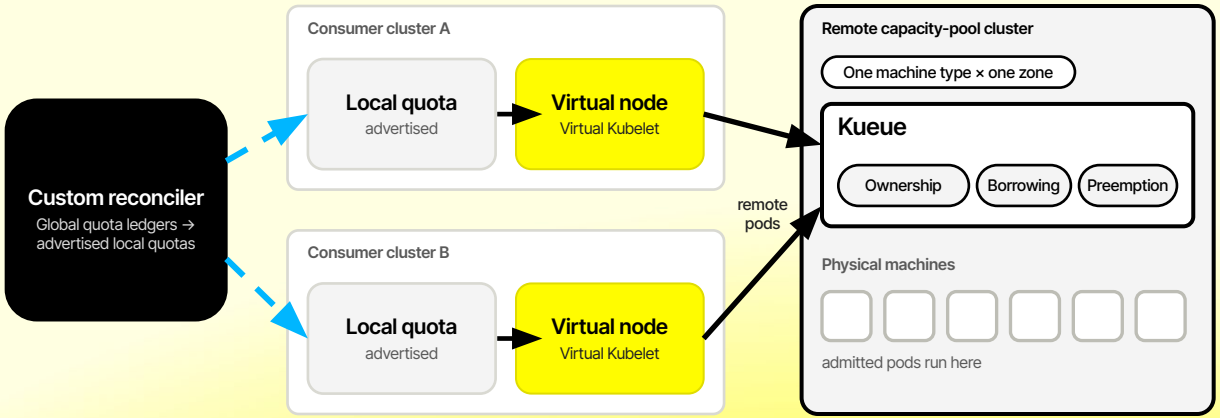
Remote pool · Kueue · records B as borrowing · B · B running

A still has 1 advertised machine — B's borrowing took nothing from the owner's local quota.

A · A · owner · B · B · borrower · Idle physical machine · Changed from state 1

© SNAP INC. 2026

Centralize competition; distribute access



One cluster per pool

Kueue sees the full picture of each pool

Virtual Kubelet access

Reach the pool from many consumer clusters

Advertise, don't conflate

Local quotas are not owned entitlements

Let Kueue reclaim

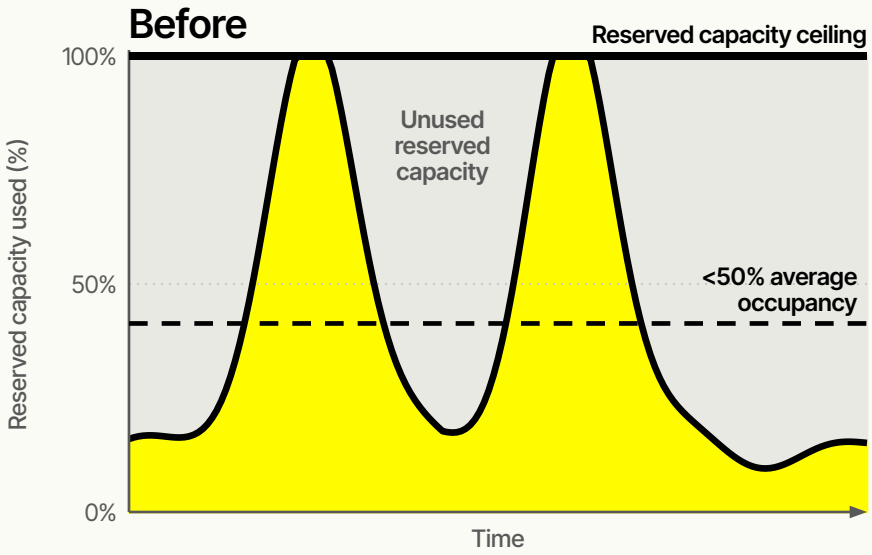
Remote Kueue borrows and preempts

Access across clusters. Borrow and reclaim within the pool.

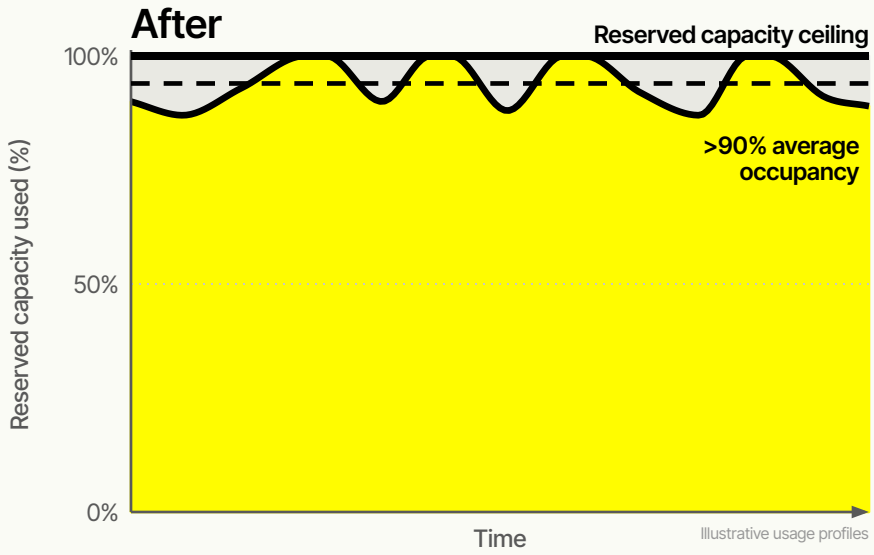
© SNAP INC. 2026

Sharing keeps our reservations busy

Same reserved capacity. Far less of it sitting idle.



Peaks at 100%, then long idle troughs



Reaches 100% frequently, with shallow dips

© SNAP INC. 2026

What's next?

More constrained machine types

Rapidly expand the platform as capacity constraints spread beyond GPUs.

Cross-region capacity failover

Build cross-region failover on Virtual Kubelet when local capacity runs out.

© SNAP INC. 2026

Centralize capacity, make it reachable, and keep it busy.

© SNAP INC. 2026

Thank you!

I’d love to hear your feedback.

Working on similar problems? Let’s share notes!
linkedin.com/in/tim-killian/



Scan to share session feedback

© SNAP INC. 2026