KubeCon CloudNativeCon
North America 2026

WG-Batch Updates: What's New and What Is Next?

Amy Chen (@amy), CoreWeave
Marcin Wielgus (@mwielgus), Google
Yuki Iwai (@tenzen-y), Preferred Networks, Inc.



KubeCon CloudNativeCon
North America 2026

Kueue



Kueue: Batch & AI Queueing in Kubernetes

KubeCon CloudNativeCon North America 2026

Core Architectural Role

- **Quota management and sharing:** defining the rules for sharing computational resources between users.
- **Workload-Level Admission:** Users workload are either admitted in full or stay pending.
- **Native Ecosystem:** First-class support for Jobs, JobSets, Ray, Kubeflow, LWS, Deployments and Pod groups.

Topology Aware Scheduling (TAS)

- **Hardware Affinity:** Aligns workloads with physical CPU/TPU blocks, racks, and superblocks.
- **Communication Throughput:** Minimizes network latency for distributed LLM and AI training.
- **Optimized Packing:** Dense bin-packing maximizes accelerator bandwidth utilization.

Marcin

Kueue Growth: Metrics Since Nov 2025

KubeCon CloudNativeCon North America 2026

RELEASES

~54

6× Frequency

- 108 total over 2 years
- 7 minor versions (v0.15–v0.21)
- Active N-1 & N-2 patch backports

MERGED PRS

3,063

+88% YoY Growth

- Merged into main branch
- 7,779 total PR actions processed
- 3,876 human merges, 3,367 backports

ISSUE DYNAMICS

2,040

+123% Intake

- 2,040 new / 1,799 closed (+148%)
- 542 net open issues
- 57% submitted in last 3 months

CONTRIBUTORS

242

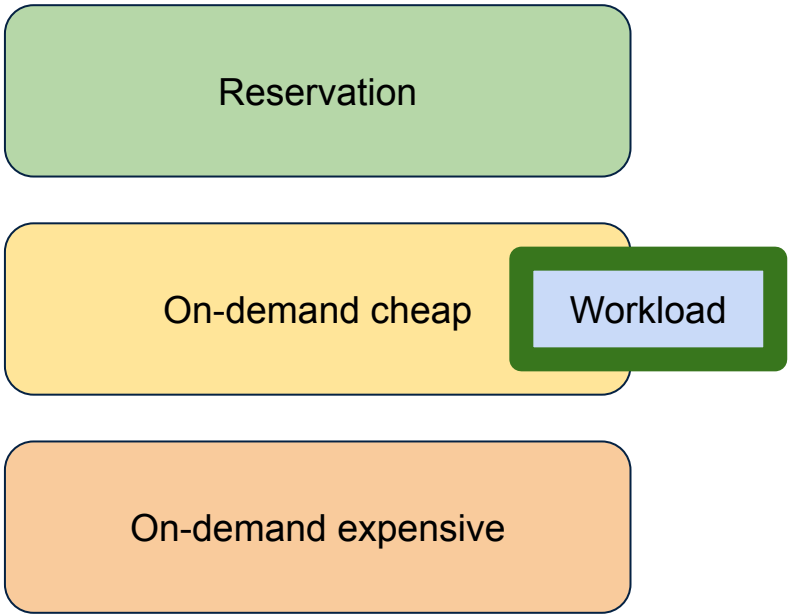
3.1× Contributor Growth

- Active developers (up from 79)
- 199 first-time contributors (82%)
- 70% of merges from non-OWNERs

Concurrent Admission: Problem & Motivation

The Problem: Sequential Resource Pursuit

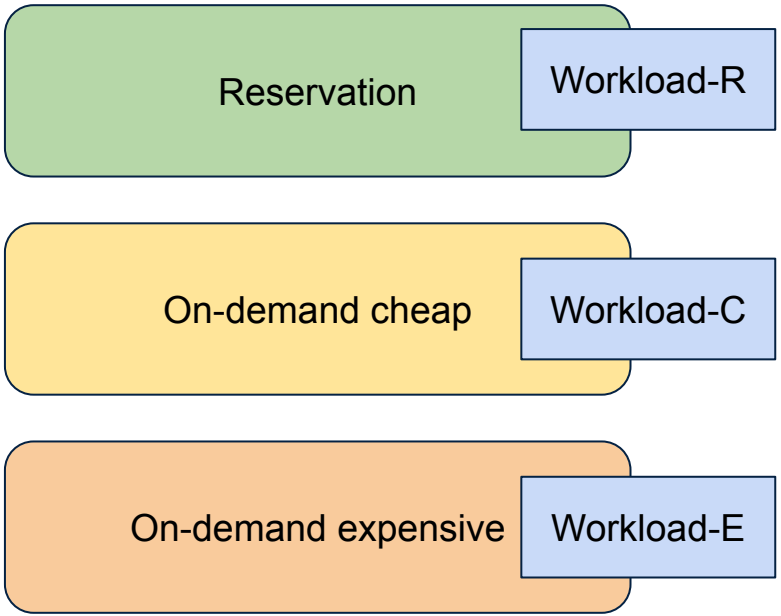
- **Resource Flavor Lock:** After a workload is admitted, it remains statically assigned to the selected machine or accelerator type.
- **Suboptimal Placement:** Jobs sequentially queue on initial flavors; if those are busy, workloads wait or get placed on expensive, inefficient machine types.
- **Static Assignment & Missed Upgrades:** Even when ideal reserved capacity or cheaper spots open up, the running or waiting workload cannot dynamically adapt.



Concurrent Admission: Architecture & Policies

Concurrent Admission Architecture & Mechanisms

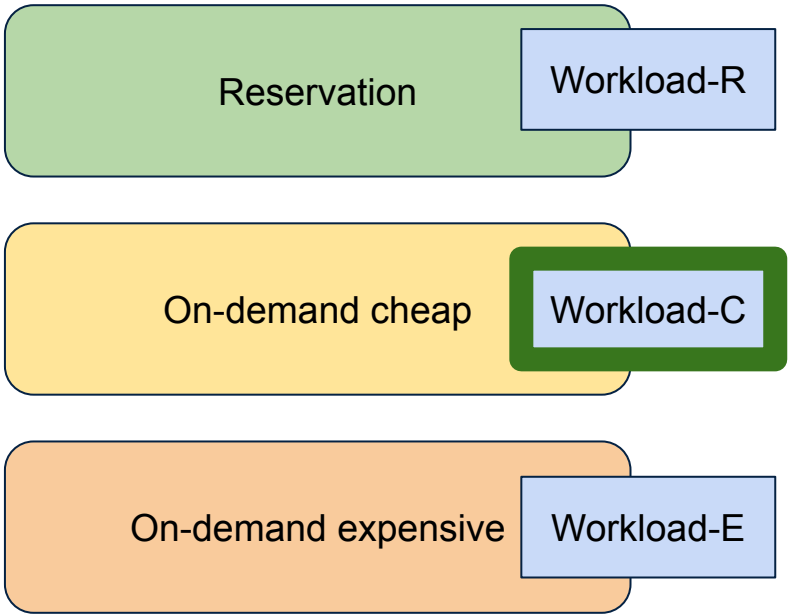
- **Virtual Option Workloads:** Creates internal virtual workloads evaluating candidate resource flavors concurrently in parallel, each targeting a specific Resource Flavor.



Concurrent Admission: Architecture & Policies

Concurrent Admission Architecture & Mechanisms

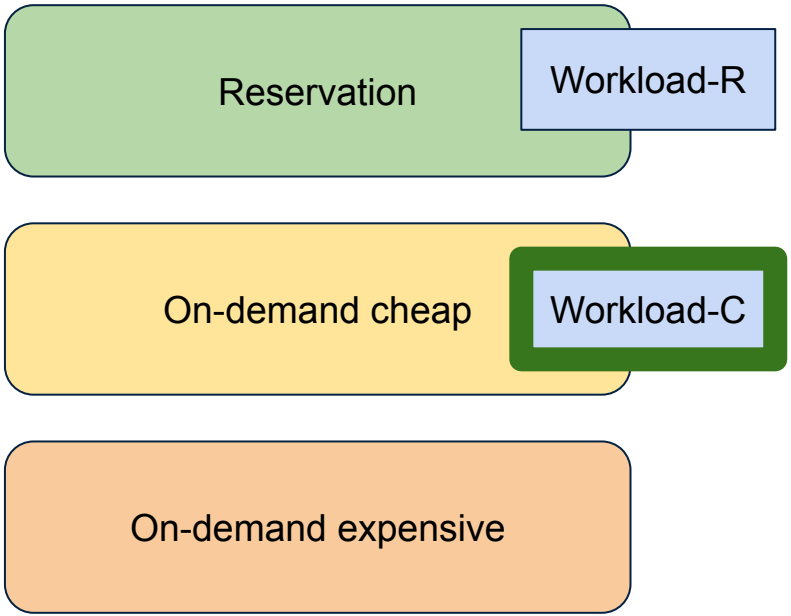
- **Eventual admission:** Eventually one of the virtual workloads will be admitted.



Concurrent Admission: Architecture & Policies

Concurrent Admission Architecture & Mechanisms

- **Removal of less preferred options:** The more preferred options are still queued and waiting.

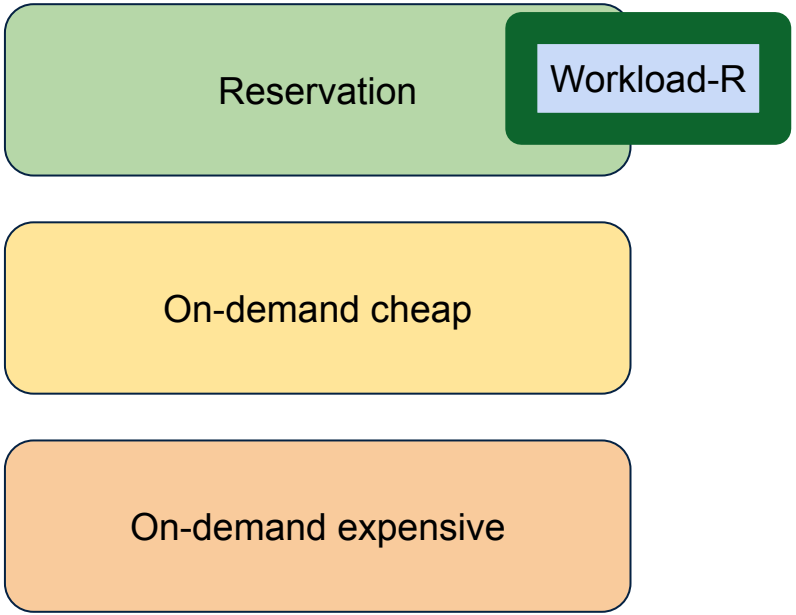


Concurrent Admission: Architecture & Policies

KubeCon CloudNativeCon North America 2026

Concurrent Admission Architecture & Mechanisms

• **Upgrade:** When/if the most preferred option gets admitted, other, already running options are stopped and the workload continues on the desired hardware.



PreemptionConfig: The Problem

Legacy API

- **Static Preemption Policy:** Preemptions in Kueue are based on priority, borrowing/reclamation and fair sharing.
- **Limited configurability:** Only handful of enums to set up the behavior. Not possible to fine tune the behavior.

Operational Bottlenecks in AI Clusters

- **Topology Fragmentation:** Distributed training is blocked by fragmented racks. Big workloads cannot preempt small, low priority workloads from other queues if they in the guaranteed quota or FS share.
- **Mixing preemptable and non-preemptable workloads:** Protecting vulnerable workloads requires config acrobatics.
- **Hero workloads:** Big and important workloads with elevated privileges are hard to configure properly.

PreemptionConfig: Declarative API

KubeCon CloudNativeCon North America 2026

PreemptionConfig API Architecture

- **CRD PreemptionConfig:** Cluster-scoped resource (shortName: preempcfg) in apiVersion kueue.x-k8s.io/v1alpha1.
- **Activation Triggers:** Triggers in activationPolicy: Always, InsufficientQuota, or QuotaFeasibleAndInsufficientTopology.
- **Preemption Scopes:** Candidate queue boundary: WithinLocalQueue, WithinClusterQueue, WithinParentCohort, WithinCohortTree, AnyClusterQueue.
- **Candidate Selectors:** Filter candidates via labelSelector, clusterQueueSelector, priority, priorityClassSelector, and numericLabels.
- **Priority Constraints:** Compares Base or Boosted priority using LessThan, LessThanOrEqual, GreaterThan, or GreaterThanOrEqual.
- **ClusterQueue Link:** Referenced in Alpha via annotation kueue.x-k8s.io/preemption-config-name.

PreemptionConfig Spec Example

```
apiVersion: kueue.x-k8s.io/v1alpha1
kind: PreemptionConfig
metadata:
  name: topology-preemption
spec:
  rules:
    - name: defrag-small-slices
      activationPolicy:
        trigger: QuotaFeasibleAndInsufficientTopology
      candidateSelectors:
        - scope: AnyClusterQueue
          priority:
            mode: Base
            comparison: LessThan
          numericLabels:
            - key: "number-of-accelerators"
              comparison: LessThan
          labelSelector:
            matchLabels: { preemptable: "true" }
```

PreemptionConfig: Future Work

KubeCon CloudNativeCon North America 2026

Upcoming Roadmap & Enhancements

- **Candidate Ordering:** Order and rank candidates by runtime, size, or resource cost to minimize disruption.
- **Delayed Preemption Triggers:** Trigger preemption only after high-priority workloads wait past a defined threshold.
- **Workload Inactivity Checks:** Prioritize preempting idle or paused workloads before active ones.
- **Cohort Hierarchy Policies:** Refine cross-cohort borrowing and preemption policies in hierarchical cohort trees.

Advanced Safeguards & Governance

- **Preemption Rate Limiting:** Bound preemption frequency via sliding time windows to eliminate cluster churn.
- **Graceful Termination:** Coordinate with checkpointing mechanisms and drain periods for safe workload eviction.
- **Fair-Sharing Integration:** Harmonize fine-grained preemption rules with dynamic Fair Sharing and borrowing.
- **Simulation & Telemetry:** Dry-run validation of preemption rules and rich metrics for preemption reasons.

Topology Spreading: Motivation

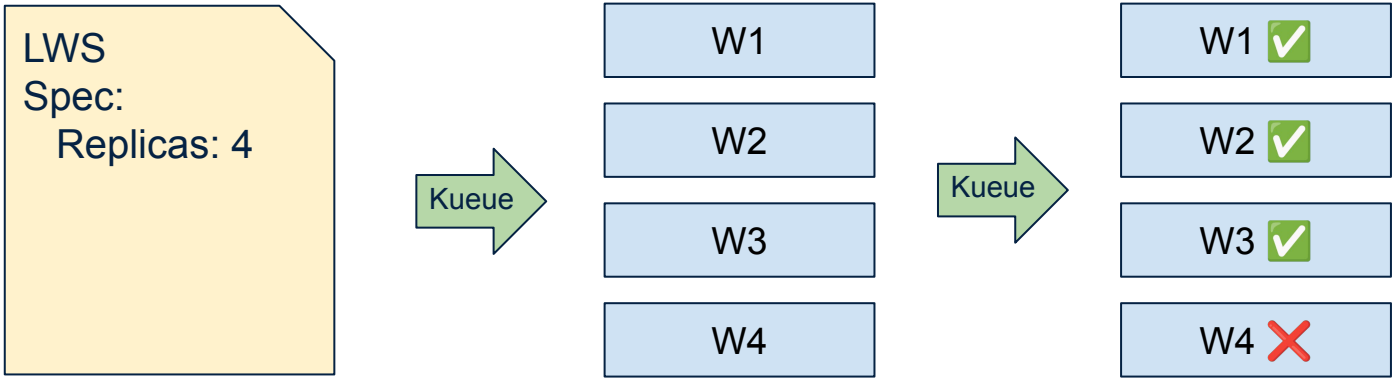
Density vs Availability Trade-off

- **TAS Packing:** TAS packs workloads tightly to optimize communication throughput.
- **Single Failure Domain:** Packing all replicas into one rack risks complete outage from hardware faults.
- **Serving Resilience:** Inference services need replicas spread across domains to maintain availability.

LWS & Accelerator Topologies

- **Worker Packing:** Packs workers tightly within each group to maximize inter-accelerator bandwidth.
- **Topology Alignment:** Aligns groups to TPU slices or GPU blocks without spreading individual workers.
- **Multi-Tier Spreading:** Spreads LWS groups across failure domains while keeping workers tightly co-located.

Topology Spreading: Serving APIs in Kueue



- **Replicas:** Each serving replica is represented as a separate Kueue workload and is admitted (and scheduled) separately. It is OK that some of them may not get in, and having fractional capacity is better than having no running service.

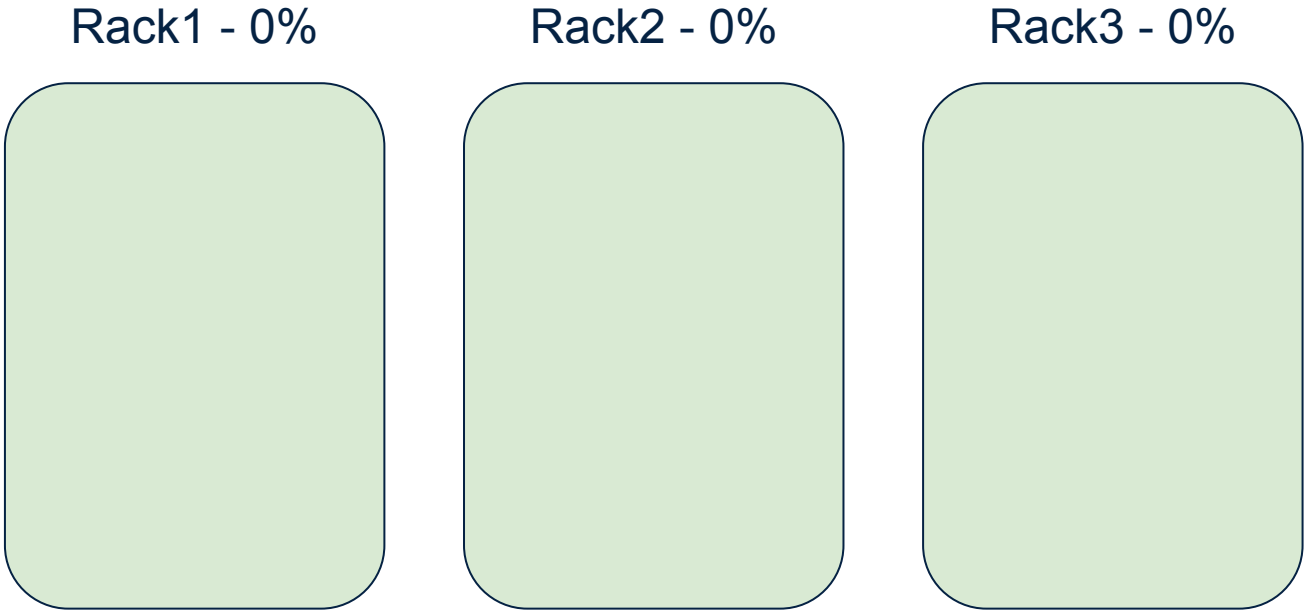
Topology Spreading: API & Workload Spec

Topology Spreading API Architecture

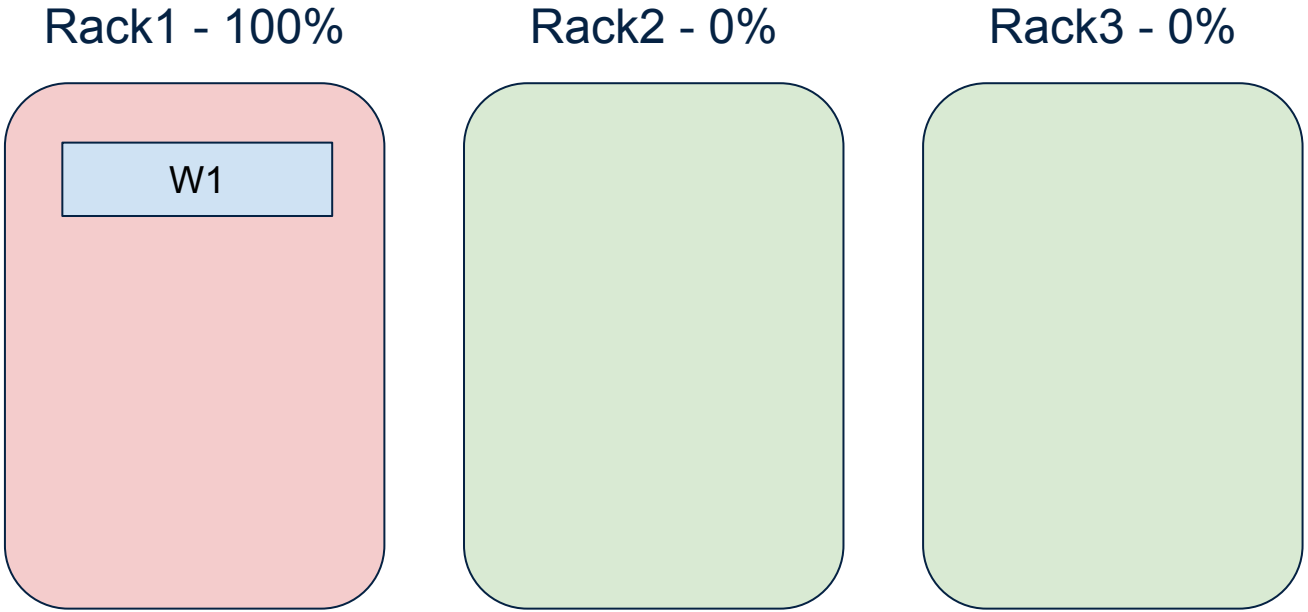
- **Workload Annotation:** Configured on pods/workloads via kueue.x-k8s.io/topology-spreading as a JSON array.
- **Topology Level:** topologyLevel matches a level from ResourceFlavor (e.g. rack, block, zone).
- **Max Domain Percentage:** maxShareAllowingPlacement specifies the maximum allowable capacity fraction (e.g. 34% or 50%) in any single domain.
- **Peer Workload Selector:** selector (labelSelector) matches peer service replicas (e.g. app: vllm-llama) that share the spreading budget. On LWS/Deployment workload selector is set up automatically.
- **Scheduling Semantics:** Spreads model replicas across failure domains while preserving tight TAS packing within each replica.

```
kueue.x-k8s.io/podset-required-topology:
cloud.provider.com/rack
kueue.x-k8s.io/podset-topology-spreading: |
{
"rules": [
  {"topologyKey": "cloud.provider.com/rack",
   "maxShareAllowingPlacement": "0.45"}
]
}
```

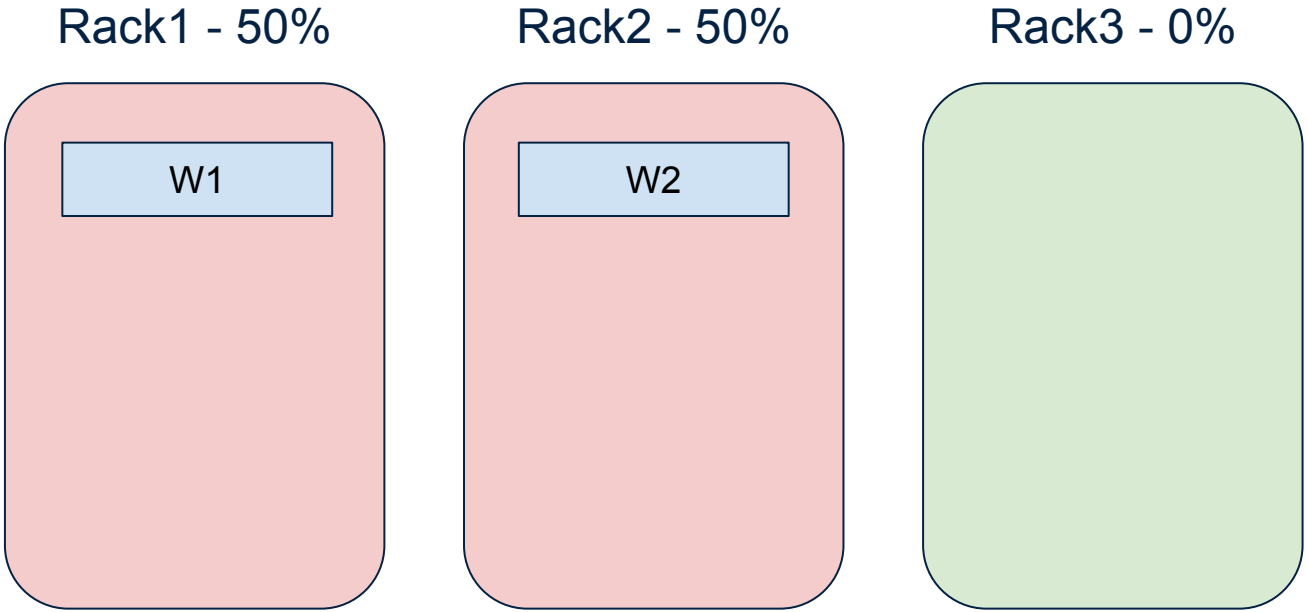
Topology Spreading: maxShareAllowingPlacement 0.45



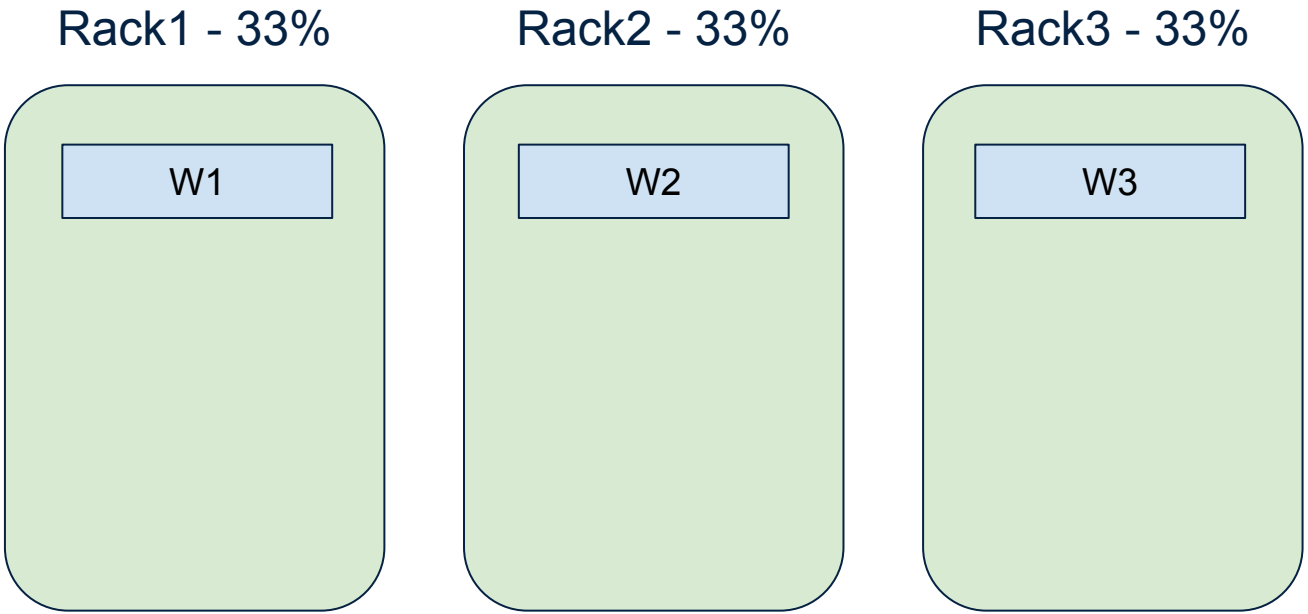
Topology Spreading: maxShareAllowingPlacement 0.45



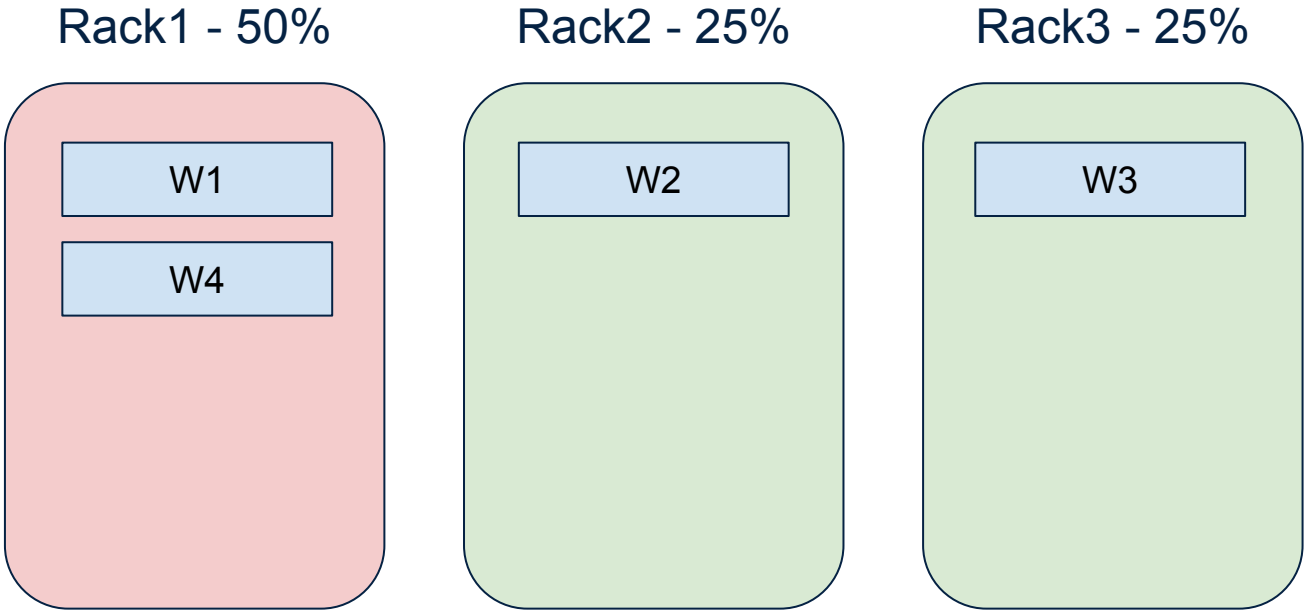
Topology Spreading: maxShareAllowingPlacement 0.45



Topology Spreading: maxShareAllowingPlacement 0.45



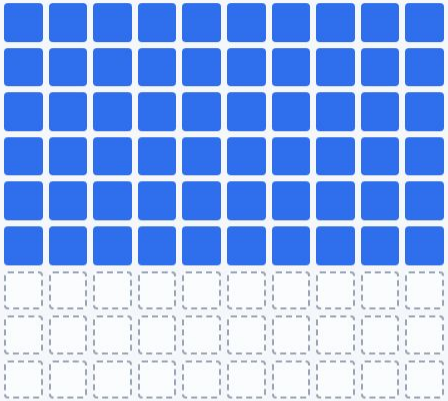
Topology Spreading: maxShareAllowingPlacement 0.45



Today you keep quota honest

KubeCon CloudNativeCon North America 2026

Nodepool grows 60 → 90 GPUs · GitOps quota vending



60 covered by quota

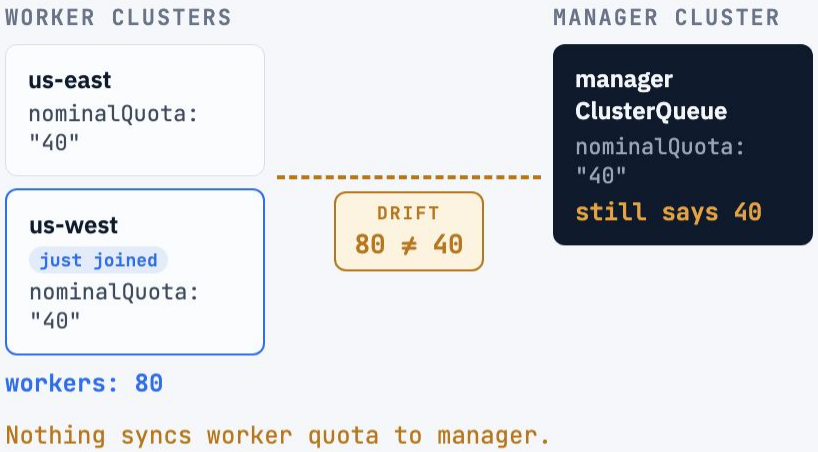
30 invisible to Kueue

```
kind: ClusterQueue
nominalQuota: "60"
```

PR open, day 2

Notice. PR. Review. Sync. Every resize.

Workers changed. Manager did not notice.



workers: 80

Nothing syncs worker quota to manager.

Kueue had no built-in way for vended nominalQuota to follow actual capacity.

DQO: Stop hand editing nominalQuota

KubeCon CloudNativeCon North America 2026

An admin writes **nominalQuota** once and nothing updates it when nodes come or go. Too high, and quota promises capacity the cluster does not have. Too low, and GPUs sit idle while jobs wait.

STATUS	USE CASE	CAPACITY CHANGES	WHAT QUOTA SHOULD DO
v0.21	**Node pools** #10270	A new nodepool joins the cluster, or an existing one is resized.	Grow as nodes go Ready, shrink as they leave. Nobody should have to edit YAML.
v0.21	**MultiKueue manager** #9988	Worker clusters join, leave, or get their own quotas changed.	Equal the sum of worker quotas.
potential	**DRA devices** #14977	ResourceSlices publish devices; partitions and MIG profiles change the count.	Follow the device inventory the drivers advertise.
potential	**Time windows** #8654	A reservation or a one-off team grant is active from 2pm to 4pm.	Rise for the window, fall back when it closes.

Dynamic Quota Orchestrator Mechanics

KubeCon CloudNativeCon North America 2026

PHASE 1 · CAPACITY DISCOVERY

PHASE 2 · DISTRIBUTION

SOURCE	CAPACITYPROVIDER	ORCHESTRATOR	EFFECTIVE QUOTA
Nodepool resize			
Nodes in flavor h100 60 GPUs Ready → 90	**local-capacity** status.capacity: 90	**Sum the providers** effectiveCapacity: 90	**Split by spec ratio 1:2** team-a 20 → 30 · team-b 40 → 60
MultiKueue manager			
Worker ClusterQueues us-east 40 + us-west 40	**multikueue-capacity** status.capacity: 80	**Sum the providers** effectiveCapacity: 80	**Manager ClusterQueue** effective 80

As nodes or worker clusters come and go, DQO raises and lowers effective quota to match. No one edits a ClusterQueue by hand.

Where DQO Stands

KubeCon CloudNativeCon North America 2026

SHIPPED · v0.20 ALPHA

- CapacityProvider and DynamicQuotaOrchestrator CRDs
- Discovery, proportional distribution, effectiveQuotas
- Scheduler, preemption, metrics read effective quota
- Bring your own provider controller

`gate: DynamicQuotaOrchestration`

LANDING · v0.21

- Nodes provider: sums allocatable of Ready nodes matching a flavor's nodeLabels
- MultiKueue provider: manager quota equals sum of workers, without writing spec
- Concept and setup docs on kueue.sigs.k8s.io

`gates: DQOLocalCapacityProvider, DQOMultiKueueCapacityProvider`

POTENTIAL FUTURE

DRA provider from ResourceSlices (#14977)

Time-windowed quota (#8654)

Elastic jobs: Partial Scale Up

A Ray autoscaler asks to go from 1000 to 5000 workers. The ClusterQueue only has room for 2000 more.

Atomic scale-up (default)

Blocked. 2000 workers' worth of quota idle.



Partial scale-up

3000 run now. A probe slice waits for the last 2000.



Multiple worker groups: shrink the last-defined group first. Put spot workers last, reserved workers first.

Opt in per job: kueue.x-k8s.io/elastic-job-scale-up-strategy: partial

Highlights

KubeCon CloudNativeCon North America 2026

v0.18 · May 29 → v0.19 · Jul 22 → v0.20 · Sep 30 → v0.21 · code freeze Nov 19

API · v0.20

v1beta1 is gone

Only v1beta2 is served. Run the migration script before upgrading.

SCHEDULING · v0.18 alpha

Concurrent Admission

Admitted workloads migrate between flavors to chase better capacity.

PREEMPTION · v0.20 alpha

Configurable Preemptions

A PreemptionConfig picks extra victims when classic or fair-sharing candidates fall short.

READINESS · v0.19 to v0.20

WaitForPodsReady on by default

Plus an unscheduled-pods timeout and per-workload timeouts via annotation.

MULTIKUEUE · v0.20

Orchestrated preemption, Ray autoscaling

Worker preemptions run one at a time (beta). Elastic Ray clusters autoscale on the worker (alpha).

MULTIKUEUE · v0.19

Dispatch you can steer

Incremental dispatcher follows config order with a stepSize: try on-prem first, cloud next.

Highlights

KubeCon CloudNativeCon North America 2026

TAS · v0.20

Topology spreading

Spread LeaderWorkerSet groups across racks or zones while keeping each group's Pods together.

TAS · v0.20

Survive more failures

Replace several failed nodes incrementally instead of evicting. Grouped leaders with sliced workers.

TAS · v0.19 to v0.20

Scale and correctness

Built for 100k nodes: compact assignments, cached trees. Per-node feasibility for taints and affinity, no more admit-then-Pending.

TAS · v0.19 beta

Multi-layer slices

One annotation, up to 3 nested constraints: groups of 32 per block, 16 per rack. On by default.

DRA · v0.18 to v0.20

DRA grows up

Integration beta. Partitionable devices and extended resources beta. Device feasibility and taints alpha.

OBSERVABILITY · v0.20 on by default

Why is my job not running?

QuotaReserved carries a real reason: WaitingForQuota, ExceedsMaxQuota, TopologyPlacementFailed. Plus new metrics: execution time, recovery wait, custom labels from your Workloads.

The AI/ML landscape is not slowing down. Neither is Kueue.

KubeCon CloudNativeCon North America 2026



WORKLOAD TYPES

Batch | HPC | Pre-training | Post-training | Inference

SCHEDULING NEEDS

Quota and fair share | Gang admission | Topology | Elastic capacity | Priority and preemption | Multi-cluster

RUNS ON KUEUE TODAY

TRAINING
Job, JobSet, TrainJob, MPIJob, Ray, Spark, AppWrapper

SERVING
LeaderWorkerSet, StatefulSet, Deployment, plain Pods

GET INVOLVED

Docs
kueue.sigs.k8s.io

Code and KEPs
github.com/kubernetes-sigs/kueue

Community
#wg-batch on Kubernetes Slack


KubeCon
CloudNativeCon
North America 2026
Job

Job - Feature Graduations



Mature Core K8S API, most features already in. Graduating:

1.37

WorkloadWithJob - Alpha2

Extend Job API with ".spec.scheduling" for WAS / TAS

1.38

MutablePodResourcesForSuspendedJobs - GA

Allows changing Pod Template Spec resources for suspended Jobs

WorkloadWithJob - Beta

Extend Job API with ".spec.scheduling" for WAS / TAS

Workload Aware Scheduling Controller APIs - Beta

Reusable API building blocks (scheduling.k8s.io) for integration controllers

JobPodFailurePolicyName - Alpha

Fine-grained JobPodFailurePolicy reason

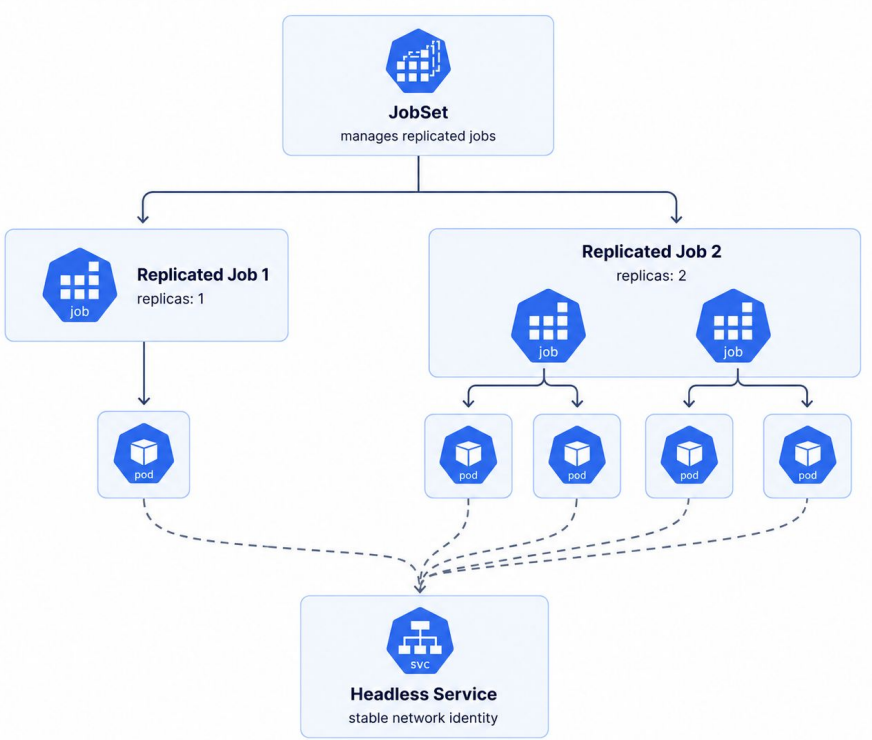
1.39 (plan)

Workload Aware Scheduling Controller APIs - GA

API versions have already been v1 in 1.38


KubeCon
CloudNativeCon
North America 2026
JobSet

JobSet



JobSet is a Kubernetes-native API:

- Managing a group of k8s Jobs as a unit.
- A unified API to deploy the below workloads on Kubernetes
  - HPC (e.g., MPI) workloads
  - AI / ML training workloads (PyTorch, Jax, TensorFlow etc.)

JobSet - Recent Notable Features

KubeCon CloudNativeCon North America 2026

0.12

In-Place Restart SideCar mode - Alpha

Allowing to restart container without Pod re-creation. This is a new sidecar container mode.

Elastic JobSet - Alpha

Allowing to elastically scale parallelisms and completions after Job started

RestartJob - Alpha

Allowing to recreate only failed ReplicatedJob instead of a whole of JobSet

0.13

ExecutionAttemptsTracking - Alpha

Allowing to track the number of Job restarting, suspend, and resume in JobSet status

JobSetWorkloadAwareSchedulingAPI - Alpha

Allowing to support Workload-Aware Scheduling which automatically create Workload, PodGroup, and CompositePodGroup resources

JobSetActiveDeadlineSeconds - Alpha

Allowing to specify maximum execution time for the JobSet level

0.14 (plan)

Graduate JobSet API version to v1

v1alpha1 API will still be supported by conversion webhooks for a while.

KubeCon | CloudNativeCon
North America 2026

WAS / TAS Integrations



What's Workload-Aware Scheduling (WAS) / Topology-Aware Scheduling (TAS)

Distributed jobs make no progress with a partial set of Pods.

Pod by Pod



5 bound, holding GPUs

3 pending

- No progress until all 8 run
- Held capacity blocks the rest: near self-deadlock
- Scattered placement, Pod-level preemption

Group (gang)



5 of 8 fit: nothing binds, capacity stays free



8 of 8 fit: group admitted, then each Pod binds

The group is also the unit for

Admission
All-or-nothing up to minCount

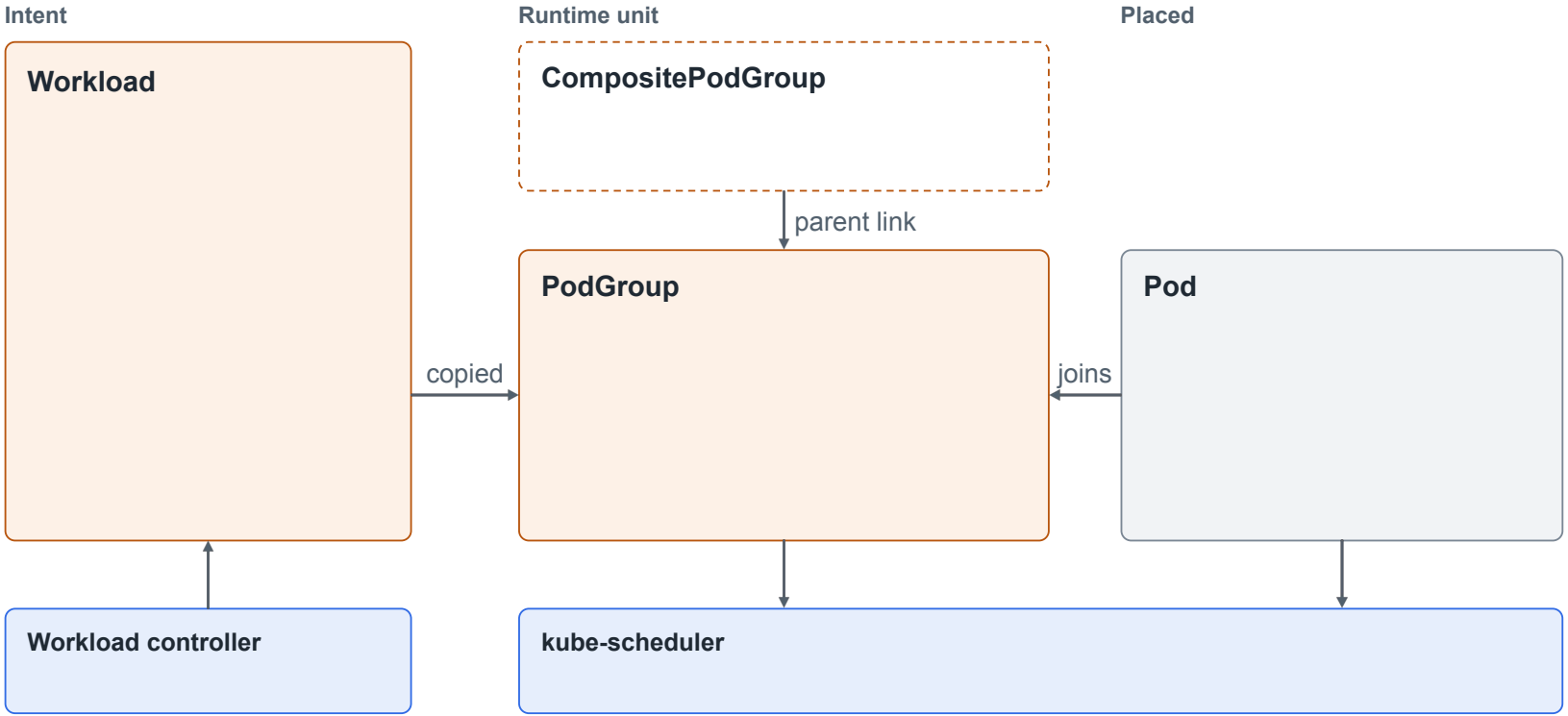
Topology
One topology domain for the whole group

Preemption
Groups as preemptors and as victims

Device claims
One ResourceClaim shared by all members

What's Workload-Aware Scheduling (WAS) / Topology-Aware Scheduling (TAS)

Declared intent, runtime scheduling unit, and the Pods that get placed are separate objects.



What's Workload-Aware Scheduling (WAS) / Topology-Aware Scheduling (TAS)

KubeCon CloudNativeCon North America 2026

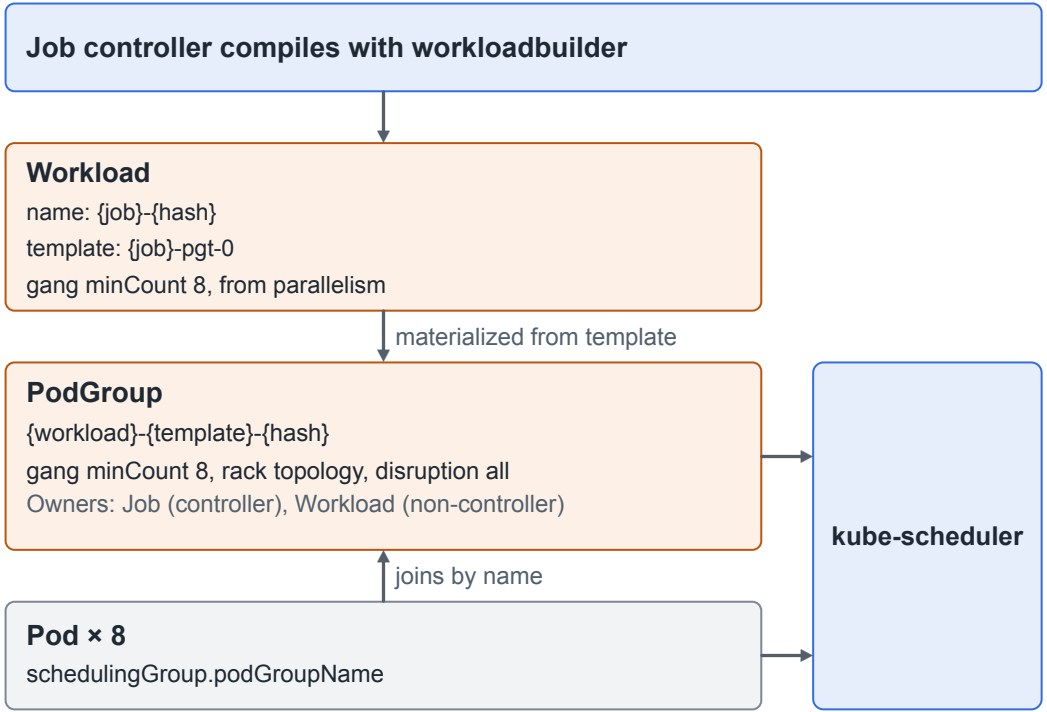
SIG Scheduling Update: Multi-Level and Topology-Aware Workload Scheduling
Thursday at 11:50-12:20; Salt Palace | Level 2 | 250 A-C

Heba Elayoty, Software Engineer, Microsoft
Dominik Marcinski, Software Engineer, Google

Job

spec.scheduling compiles to one Workload and one PodGroup.

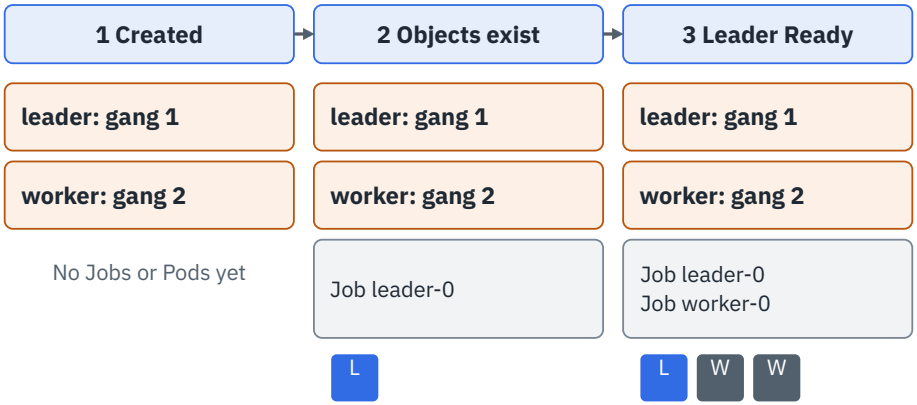
```
apiVersion: batch/v1
kind: Job
metadata:
  name: finetune
spec:
  completionMode: Indexed
  parallelism: 8
  completions: 8
  scheduling:
    schedulingPolicy:
      gang: {}
    schedulingConstraints:
      topology:
      - key: topology.kubernetes.io/rack
    disruptionMode:
      all: {}
  template: ...
```



JobSet

Sequenced stages: One gang per stage. The worker Job appears only after the leader is Ready.

```yaml
apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  name: sequenced-gang
spec:
  scheduling:
    replicatedJobs:
    - targetReplicatedJobs: [leader]
      schedulingPolicy: {gang: {}}
    - targetReplicatedJobs: [worker]
      schedulingPolicy: {gang: {}}
  replicatedJobs:
  - name: leader
    replicas: 1
    template: ...
  - name: worker
    replicas: 1
    dependsOn:
    - name: leader
      status: Ready
    template:
      spec:
        parallelism: 2
        completions: 2
        template: ...
```



Job & JobSet

WG-Workload Aware Scheduling: An Introduction
Thursday at 11:00-11:30; Salt Palace | Level 2 | 250 A-C

Kevin Hannon, Principal Software Engineer, Red Hat
Andrey Velichkevich, AI Engineer, Apple
Heba Elayoty, Software Engineer, Microsoft
Matt Matejczyk, Software Engineer, Google

Kueue: Why WAS / TAS integrations

KubeCon CloudNativeCon North America 2026

Kueue Native TAS

Kueue | hand-written copy | kube-scheduler

BASELINE

Queue	Quota · fairness	Victim selection	Topology capacity	Node feasibility hand-written copy	Assignment	Bind

Feasibility is Kueue's own re-implementation: taint / toleration + required affinity only. **Every other predicate assumed schedulable** .

AFTER BIND

TopologyAssignment stamped
quota is held

→

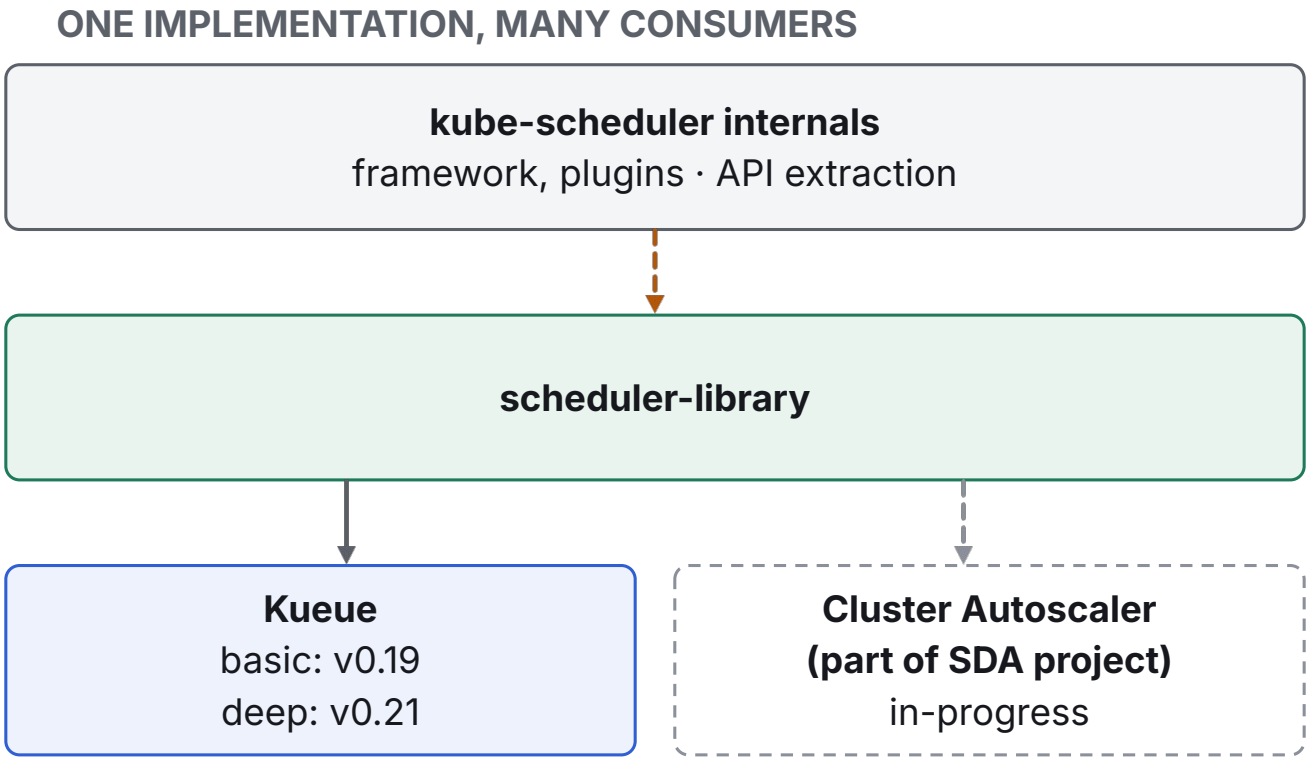
kube-scheduler — full plugin set
TopologySpread | InterPod | Ports | Volumes | DRA ...

→

✓ agrees Pods run

× **refuses: Pods Pending** and quota still held

Kueue: High-level architectures



WAS / TAS Integrations - Kueue

Kueue · scheduler-library · WAS / kube-scheduler

BASIC

Queue	Quota · fairness	Victim order	Victim search	Topology capacity	Node feasibility	Assignment	Bind

DEEP: FIRST STAGE

Queue	Quota · fairness	Victim order	Victim search	Topology capacity	Node feasibility	Assignment	Bind

END STATE: TRACK 4

Queue	Quota · fairness	WAS / kube-scheduler scheduling loop victims · topology · feasibility · assignment · bind

Kueue: simple Job scheduling scenario

KubeCon CloudNativeCon North America 2026

One Job, End to End

RUNS IN **Kueue** scheduler-library kube-scheduler

STEP 1 / 5

Job train: 4 Pods × (2 CPU, 4Gi), required rack. Six nodes with 4 CPU each.

PENDING none yet — Kueue has only the Workload

block b1				block b2	
rack r1		rack r2		rack r3	
node-r1-a	node-r1-b	node-r2-a	node-r2-b	node-r3-a	node-r3-b

Kueue · Workload
job-train-8c4f2 · count 4
mode: Fit

scheduler-library · snapshot
Nodes and bound Pods only

Result
—

Kueue **AssignFlavors**: quota for 8 of 24 CPU fits, so the mode is Fit. No Pod exists yet, so the placement must be simulated.

Kueue: simple Job scheduling scenario

KubeCon CloudNativeCon North America 2026

One Job, End to End

RUNS IN Kueue scheduler-library kube-scheduler

STEP 2 / 5

Job train: 4 Pods × (2 CPU, 4Gi), required rack. Six nodes with 4 CPU each.

PENDING: main-0 main-1 main-2 main-3

block b1				block b2	
rack r1		rack r2		rack r3	
node-r1-a	node-r1-b	node-r2-a	node-r2-b	node-r3-a	node-r3-b

Kueue · Workload
job-train-8c4f2 · count 4
mode: Fit

scheduler-library · snapshot
4 virtual Pending Pods
PodGroup p.main · minCount 4 · rack

Result
—

Kueue + scheduler-library

Virtualize: wasPlanner writes 4 virtual Pending Pods and one PodGroup, minCount 4 on rack, into the cycle snapshot only.

Kueue: simple Job scheduling scenario

KubeCon CloudNativeCon North America 2026

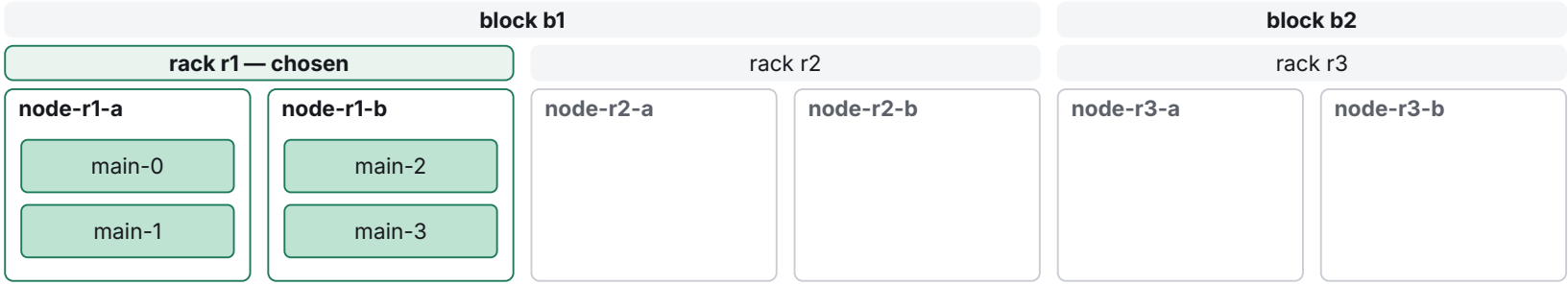
One Job, End to End

RUNS IN Kueue **scheduler-library** kube-scheduler

STEP 3 / 5

Job train: 4 Pods × (2 CPU, 4Gi), required rack. Six nodes with 4 CPU each.

PENDING



Kueue · Workload
job-train-8c4f2 · count 4
mode: Fit

scheduler-library · snapshot
4 virtual Pods, placed
PodGroup p.main

Result
Placement: rack r1
2 Pods per node

scheduler-library **ScheduleWorkload**: the library generates one placement per rack. r1 fits all 4 Pods, 2 per node; r2 and r3 would too.

Kueue: simple Job scheduling scenario

KubeCon CloudNativeCon North America 2026

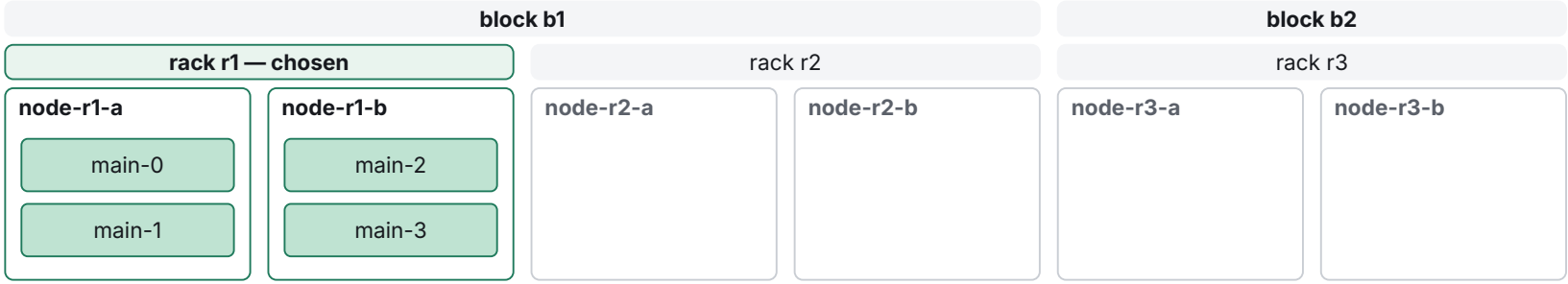
One Job, End to End

RUNS IN **Kueue** scheduler-library kube-scheduler

STEP 4 / 5

Job train: 4 Pods × (2 CPU, 4Gi), required rack. Six nodes with 4 CPU each.

PENDING



Kueue · Workload
QuotaReserved=True
Admitted=True

scheduler-library · snapshot
Virtual objects removed

Result
TopologyAssignment · hostname
node-r1-a: 2 · node-r1-b: 2

Kueue **Fold the result**: Pod-to-node results become a hostname-level TopologyAssignment. Virtual objects are removed. The Workload is admitted.

Kueue: simple Job scheduling scenario

KubeCon CloudNativeCon North America 2026

One Job, End to End

RUNS IN Kueue | scheduler-library | kube-scheduler

STEP 5 / 5

Job train: 4 Pods × (2 CPU, 4Gi), required rack. Six nodes with 4 CPU each.

PENDING

block b1

rack r1 — chosen

node-r1-a
- main-0 · Running
- main-1 · Running

node-r1-b
- main-2 · Running
- main-3 · Running

rack r2

node-r2-a

node-r2-b

block b2

rack r3

node-r3-a

node-r3-b

Kueue · Workload
Admitted
real Pods get nodeSelector

scheduler-library · snapshot
4 virtual Running Pods
on node-r1-a, node-r1-b

Result
kube-scheduler binds
the real Pods

Kueue + kube-scheduler

Next cycle: the admission replays as virtual Running Pods on r1. TopologyUngater sets nodeSelector. kube-scheduler binds.

Try it yourself



Try yourself with enabling SchedulerLibraryDeepIntegration feature gate in Kueue!

Conclusion

- Kueue
  - Concurrent Admission
  - TAS TopologySpread
  - Configurable Preemptions
  - Dynamic Quota Orchestrator (DQO)
- Job
  - Feature Maturity for v0.37 - v0.39
- JobSet
  - Feature Maturities for v0.12 - 0.14
- WAS TAS Integrations
  - Job
  - JobSet
  - Kueue Basic Integration / Deep Integration

WG Batch

KubeCon CloudNativeCon North America 2026

How to find us?

SLACK CHANNEL
slack.k8s.io #wg-batch

WEEKLY MEETINGS
Thursdays @ 4pm CET

COMMUNITY
git.k8s.io/community/wg-batch



Questions?



Amy Chen
- GitHub: amy
- Slack: Amy Chen



Marcin Wielgus
- GitHub: mwielgus
- Slack: mwielgus



Yuki Iwai
- GitHub: tenzen-y
- Slack: tenzen-y

WG-Batch Updates: What's New and What Is Next?

Kueue: Why WAS / TAS integrations

Where the Boundary Sits Today

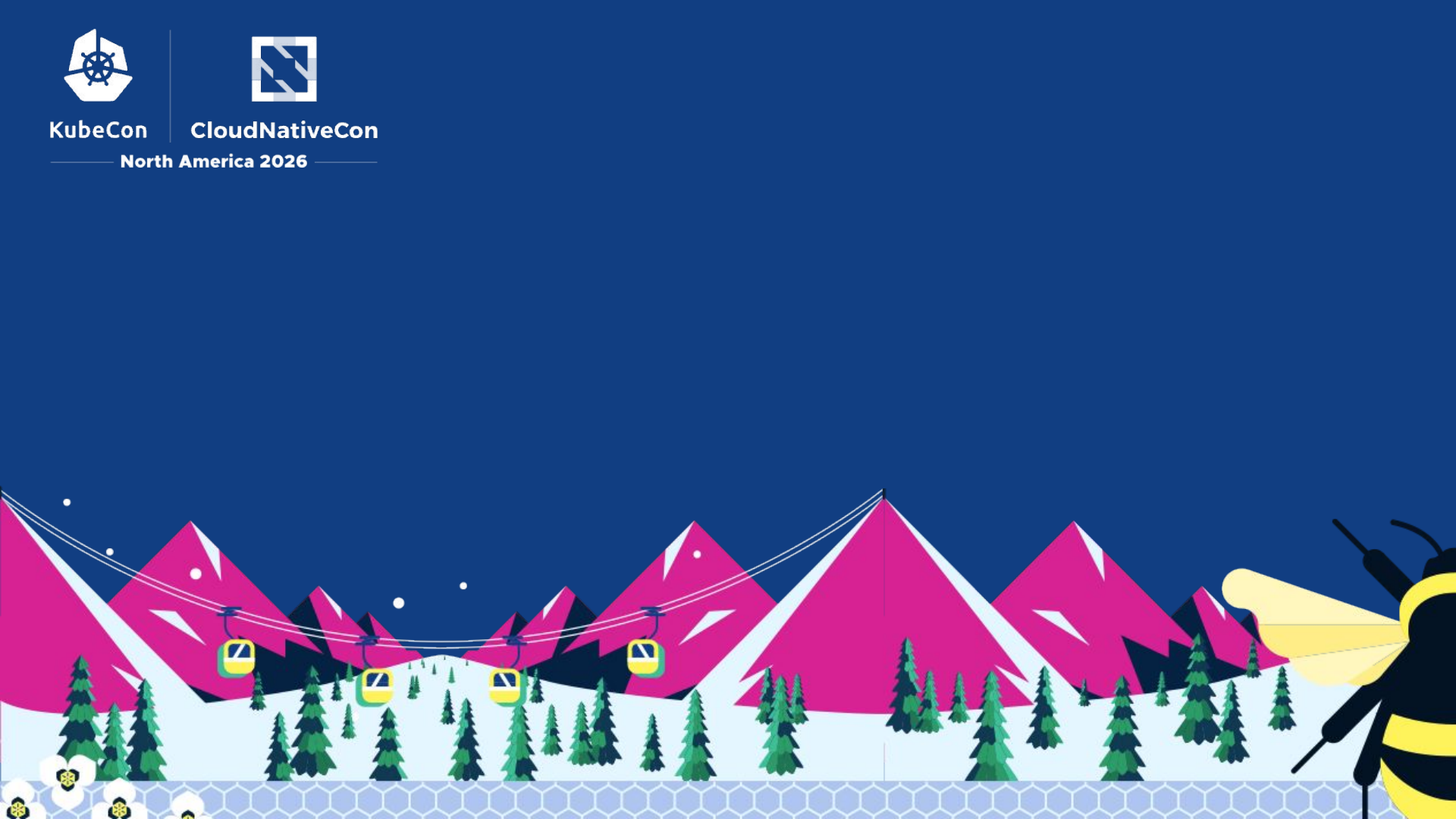
Kueue scheduler-library kube-scheduler

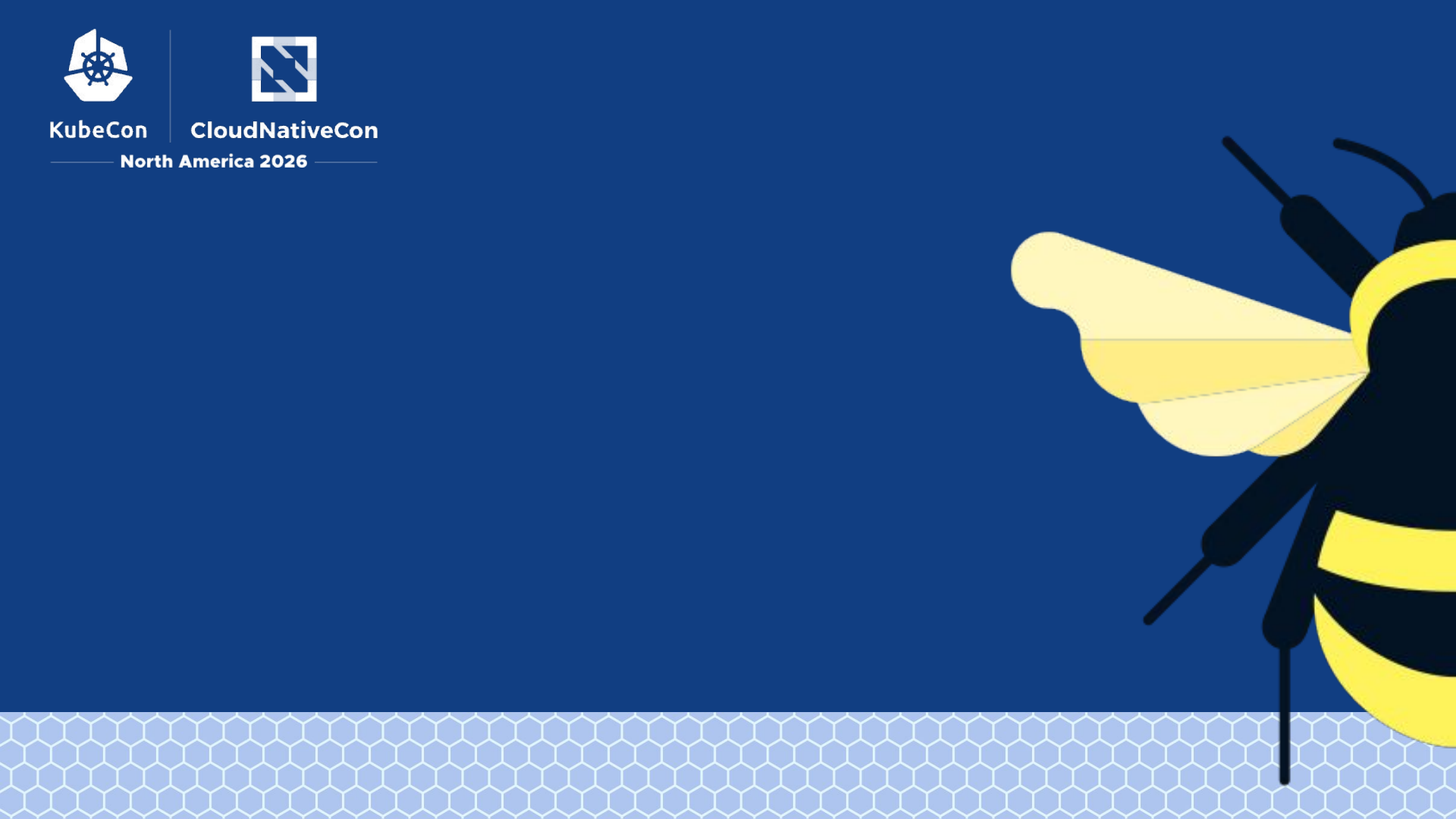
Native	Queue fairness	Quota flavor	Victim order	Topology search	Node feasibility	Node capacity	Preempt node check	Bind
Shallow	Queue fairness	Quota flavor	Victim order	Topology search	**Node feasibility**	Node capacity	Preempt node check	Bind
Deep	Queue fairness	Quota flavor	Victim order	**Topology search**	**Node feasibility**	**Node capacity**	**Preempt node check**	Bind


KubeCon
CloudNativeCon
North America 2026


KubeCon
CloudNativeCon
North America 2026


KubeCon
CloudNativeCon
North America 2026


KubeCon
CloudNativeCon
North America 2026


KubeCon
CloudNativeCon
North America 2026

PROPOSAL: AGENDA (25 min + 5 min buffer)

- Job + JobSet excluding WAS/TAS (3 min)
  - Graduations
  - Highlights
- Kueue (15 min)
  - Intro (30 sec - 1 min)
  - 3 choice from [DRA integrations, ElasticJobs w/ Partial Scaling, Configurable Preemptions, DQO, TASTopologySpread, ConcurrentAdmission]
    - Choice A (4min)
    - Choice B (4min)
    - Choice C (4min)
  - Highlights + Future Plan (2 min)
- WAS / TAS integrations (5 min)
- Summary (2 min)

AGENDA (25 min + 5 min buffer)

- Kueue (16.5 min)
  - Marcin (8.5 min)
    - Kueue Intro (30 sec)
    - Configurable Preemptions (4min)
    - Topology Spread (2min)
    - Concurrent Admission (2min)
  - Amy (8 min)
    - DQO (4 min)
    - Highlights (2 min)
    - Future Plan (2 min)
- Yuki (8.5 min)
  - Job + JobSet excluding WAS/TAS (3 min)
    - Graduations
    - Highlights
  - WAS / TAS integrations (4 min)
    - Job WAS / TAS
    - JobSet WAS / TAS
    - Kueue WAS / TAS
  - Summary of this talk (1.5 min)

Marcin: 8.5 min
Amy: 8 min
Yuki: 8.5 min