



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

WG-Batch Updates:

What's new and What's next?

Kevin Hannon (@kannon92), Red Hat
Yuki Iwai (@tenzen-y), CyberAgent, Inc.





KubeCon



CloudNativeCon

Europe 2026

What's WG Batch



WG Batch

Forum to discuss enhancements to **better support batch workloads** in Kubernetes (HPC, AI/ML, data analytics, CI)

The goal is to **reduce fragmentation** in the Kubernetes batch ecosystem

Stakeholders:

- [SIG Scheduling](#)
- [SIG Apps](#)
- [SIG Node](#)
- [SIG Autoscaling](#)



WG Batch: Scope

- Identifying challenges for batch workloads
- Additions to the batch APIs (eg. Job, JobSet)
- Primitives for workload-level queuing
- Improvements to workload-level scheduling for better:
 - cluster utilization
 - workload performance
- Support for specialized hardware





KubeCon



CloudNativeCon

Europe 2026

Job



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

- **PodReplacementPolicy - GA from 1.34**
 - Allows delaying creation of replacement Pods until the previous Pod is terminal
- **MutablePodResourcesForSuspendedJobs - Beta in 1.36**
 - Allows changing Pod Template Spec resources for suspended Jobs
- **MutableSchedulingDirectivesForSuspendedJobs - Beta in 1.36**
 - Allows resetting status.startTime for suspended Jobs
- **EnableWorkloadWithJob - Alpha in 1.36**
 - Allows automatic creation of Workload and PodGroup resources for Job



KubeCon

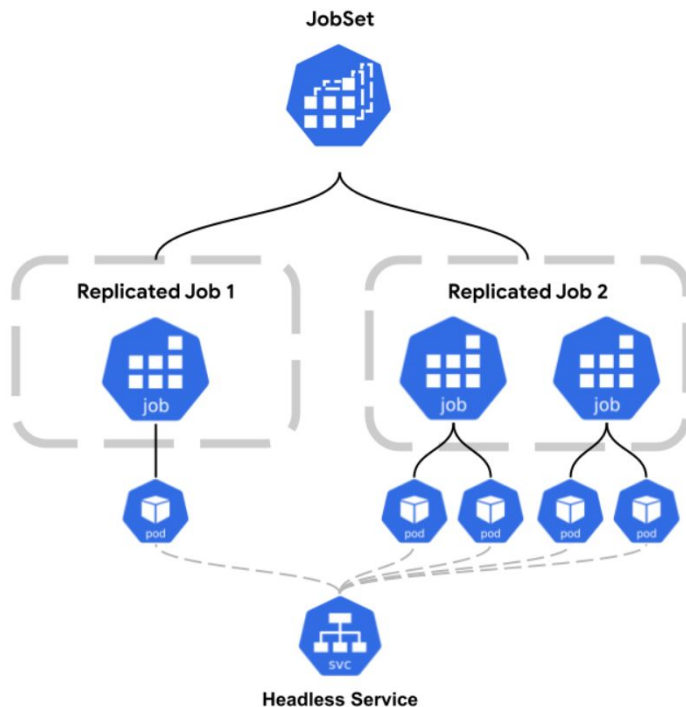


CloudNativeCon

Europe 2026

JobSet





JobSet is a Kubernetes-native API for managing a group of k8s Jobs as a unit. It aims to offer a unified API for deploying HPC (e.g., MPI) and AI/ML training workloads (PyTorch, Jax, TensorFlow etc.) on Kubernetes.

Depends on

```
1  # This example creates three ReplicatedJobs.
2  # Then node ReplicatedJob waits until the dataset-initializer and model-initializer ReplicatedJob are Complete.
3  apiVersion: jobset.x-k8s.io/v1alpha2
4  kind: JobSet
5  metadata:
6    name: multiple-depends-on
7  spec:
8    replicatedJobs:
9      - name: dataset-initializer
10      ...
11      - name: model-initializer
12      ...
13      - name: worker
14      dependsOn:
15        - name: dataset-initializer
16          status: Complete
17        - name: model-initializer
18          status: Complete
```

Volume Claim Policies

- Bringing StatefulSet-like policies to Batch workloads
- RetentionPolicies

```
volumeClaimPolicies:  
  - templates:  
    - metadata:  
      name: initializer  
    spec:  
      accessModes: ["ReadWriteOnce"]  
      resources:  
        requests:  
          storage: 1Gi  
  - templates:  
    - metadata:  
      name: checkpoints  
    spec:  
      accessModes: ["ReadWriteOnce"]  
      resources:  
        requests:  
          storage: 1Gi  
  retentionPolicy:  
    whenDeleted: Retain
```

Failure Policy

- Match on specific Job failures

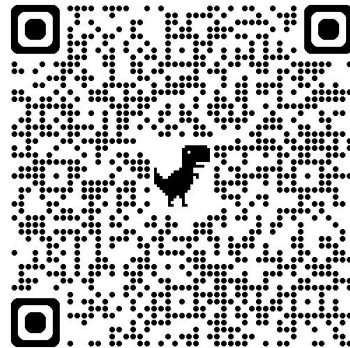
```
apiVersion: jobset.x-k8s.io/v1alpha2
kind: JobSet
metadata:
  name: jobset-example
spec:
  failurePolicy:
    maxRestarts: 5
    rules:
      - action: RestartJobSetAndIgnoreMaxRestarts
        onJobFailureReasons:
          - PodFailurePolicy
        onJobFailureMessagePatterns: # New field
          - ".*DisruptionTarget.*" # Failure message: Pod <pod namespace>/<pod name> has condition DisruptionTarget match
      - action: FailJobSet
        onJobFailureReasons:
          - PodFailurePolicy
        onJobFailureMessagePatterns: # New field
          - ".*exit code 42.*" # Failure message: Container <container name> for pod <pod namespace>/<pod name> failed wi
  replicatedJobs:
    - name: leader
      replicas: 1
      template:
        spec:
          backoffLimit: 0
          completions: 1
          parallelism: 1
          template:
            spec:
              restartPolicy: Never
              containers:
                - name: main
                  ...
          podFailurePolicy:
            rules:
              - action: FailJob
                onPodConditions:
                  - type: DisruptionTarget
              - action: FailJob
                onExitCodes:
                  containerName: main
                  operator: In
                  values: [42]
```

Restarts in Place

```
apiVersion: v1
kind: Pod
metadata:
  name: ml-worker-pod
spec:
  restartPolicy: Never
  initContainers:
    # This init container will re-run on every in-place restart
    - name: setup-environment
      image: my-repo/setup-worker:1.0
    - name: watcher-sidecar
      image: my-repo/watcher:1.0
      restartPolicy: Always
      restartPolicyRules:
        - action: RestartAllContainers
          exitCodes:
            operator: In
            # A specific exit code from the watcher triggers a full pod restart
            values: [88]
  containers:
    - name: main-application
      image: my-repo/training-app:1.0
```

- Rescheduling workloads is expensive

- In place restart of Containers based on exit code





KubeCon



CloudNativeCon

Europe 2026

Kueue



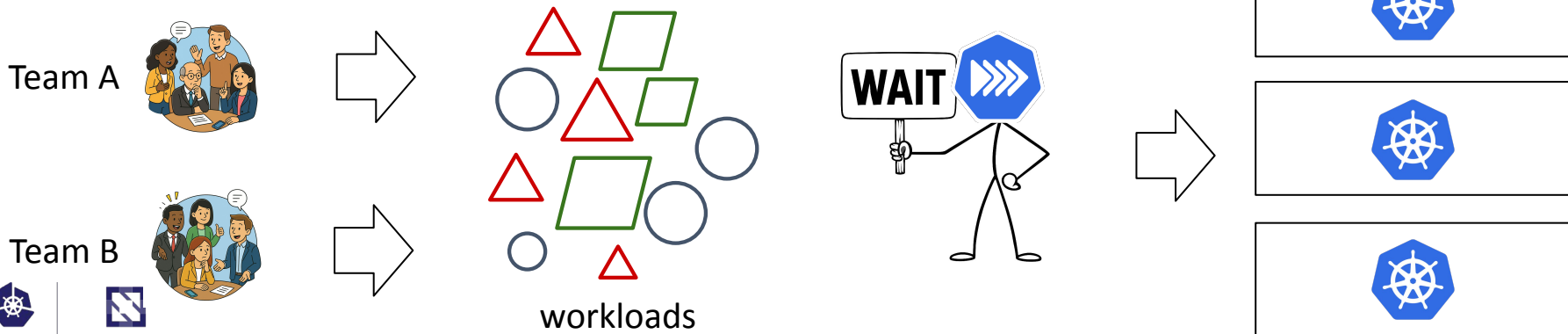
What's Kueue

- **AI Workload-level scheduler**

- all-or-nothing scheduling
- delayed Pod creation for off-loading control-plane components
- multi-cluster workloads dispatching (MultiKueue)
- topology-aware Workload scheduling (TAS)

- **Resource quota manager**

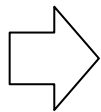
- multi-tenant quota management
- control over resource preferences
- fair resource sharing (FairSharing)



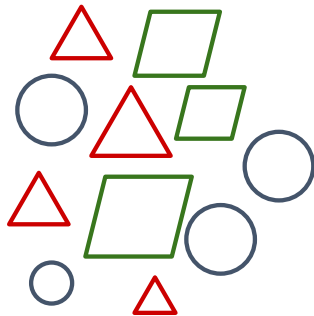
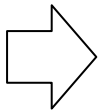
What's Kueue

- **AI Workload-level scheduler**
 - all-or-nothing scheduling
 - delayed Pod creation for off-loading control-plane components
 - multi-cluster workloads dispatching (MultiKueue)
 - **topology-aware Workload scheduling (TAS)**
- **Resource quota manager**
 - multi-tenant quota management
 - control over resource preferences
 - fair resource sharing (FairSharing)

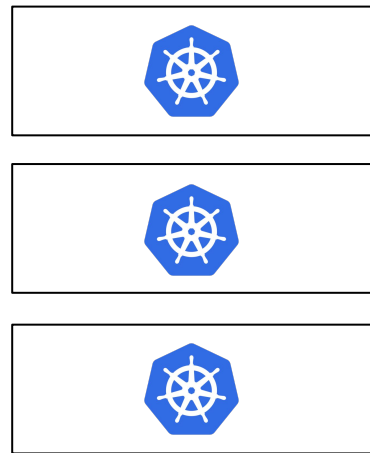
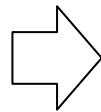
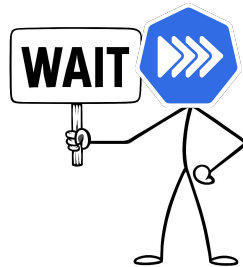
Team A



Team B



workloads



Kueue: Dive to TopologyAwareScheduling

- Find optimal placement for a Workload

Algorithm	Topology Preference Level	Kueue Feature Gate
BestFit	● Required ● Preferred	N/A
BestFit	● Unconstrained	TASProfileMixed=false
LeastFreeCapacity	● Unconstrained	TASProfileMixed=true (0.10.0, beta 0.15.0)
BalancedPlacement	● Preferred	TASBalancedPlacement=true (0.15.0)
MultilayerTopology	● Required ● Preferred	TASMultilayerTopology=true (0.17.0)

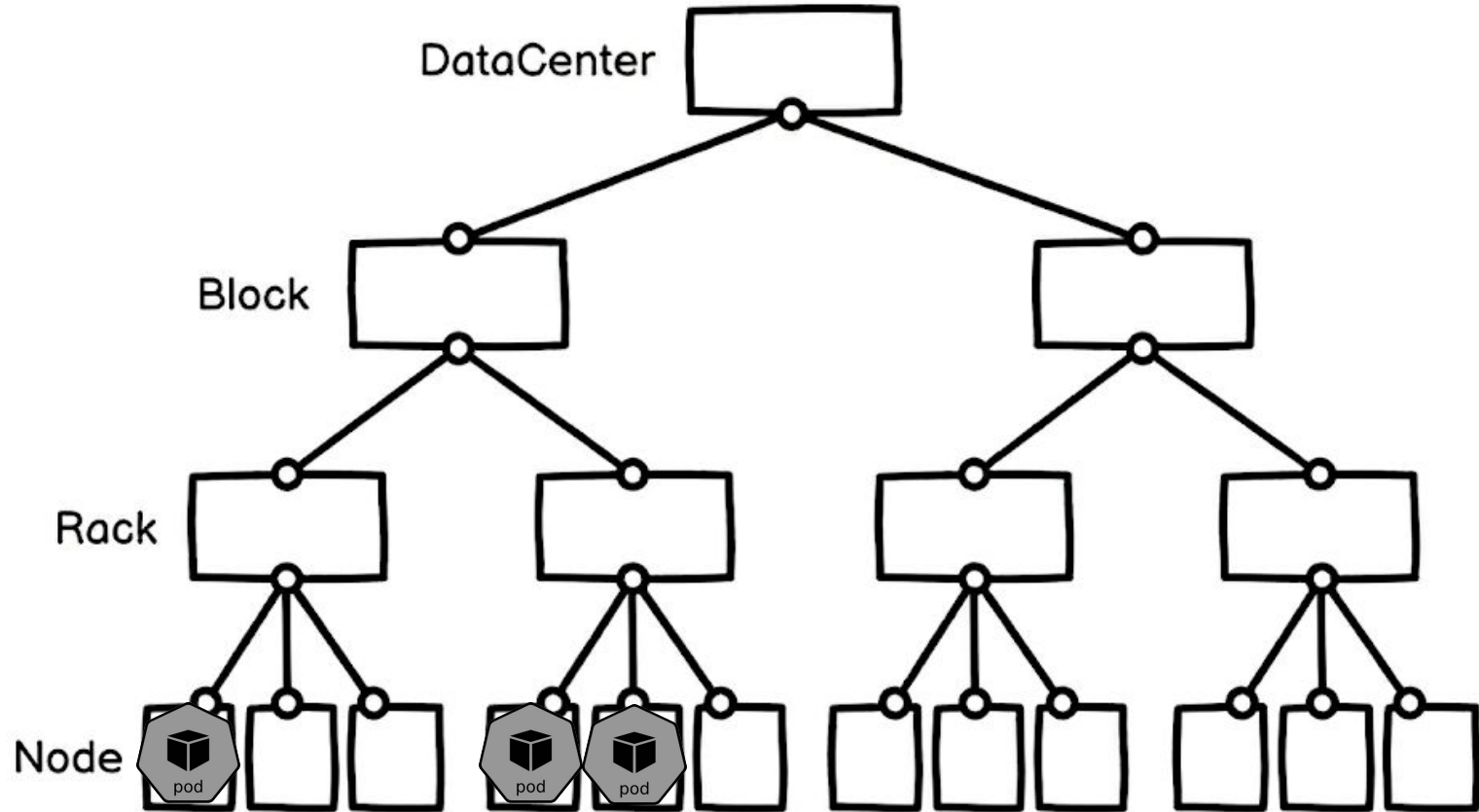
Kueue: Dive to TopologyAwareScheduling

- Find optimal placement for a Workload

Algorithm	Topology Preference Level	Kueue Feature Gate
BestFit	● Required ● Preferred	N/A
BestFit	● Unconstrained	TASProfileMixed=false
LeastFreeCapacity	● Unconstrained	TASProfileMixed=true (0.10.0, beta 0.15.0)
BalancedPlacement	● Preferred	TASBalancedPlacement=true (0.15.0)
MultilayerTopology	● Required ● Preferred	TASMultilayerTopology=true (0.17.0)

Kueue: TopologyAwareScheduling

BestFit



KubeCon

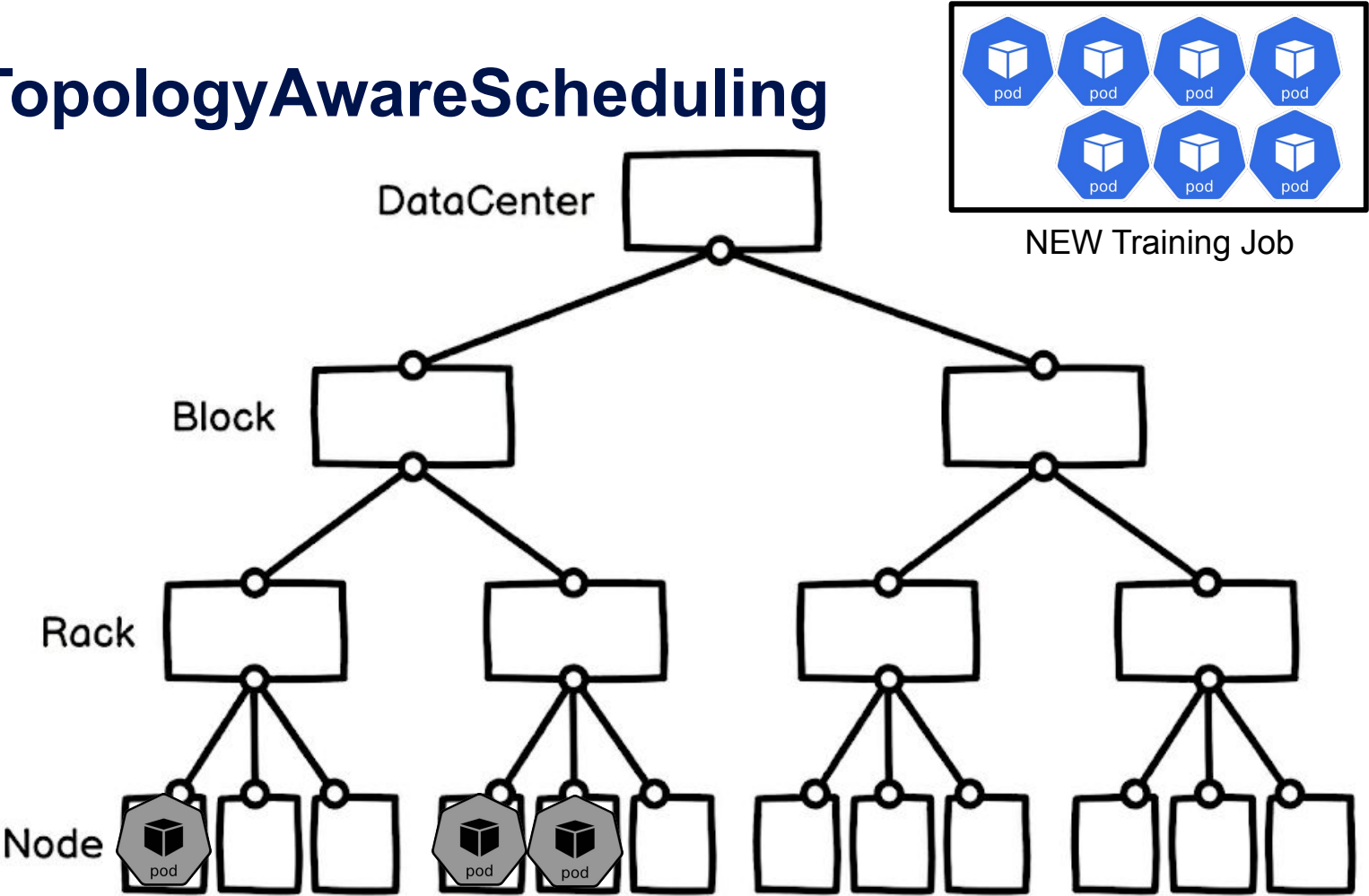


CloudNativeCon

Europe 2026

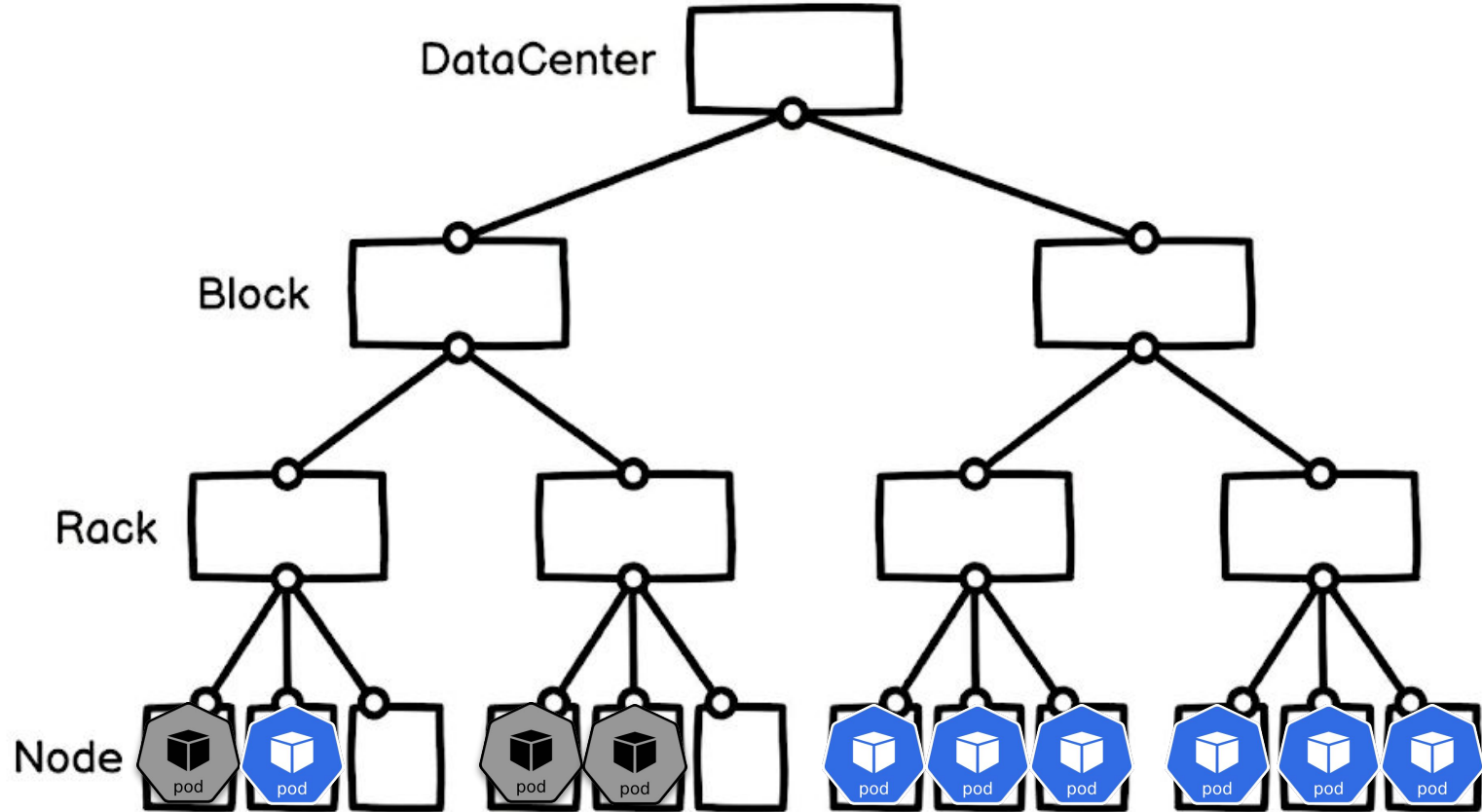
Kueue: TopologyAwareScheduling

BestFit



Kueue: TopologyAwareScheduling

BestFit



KubeCon



CloudNativeCon

Europe 2026

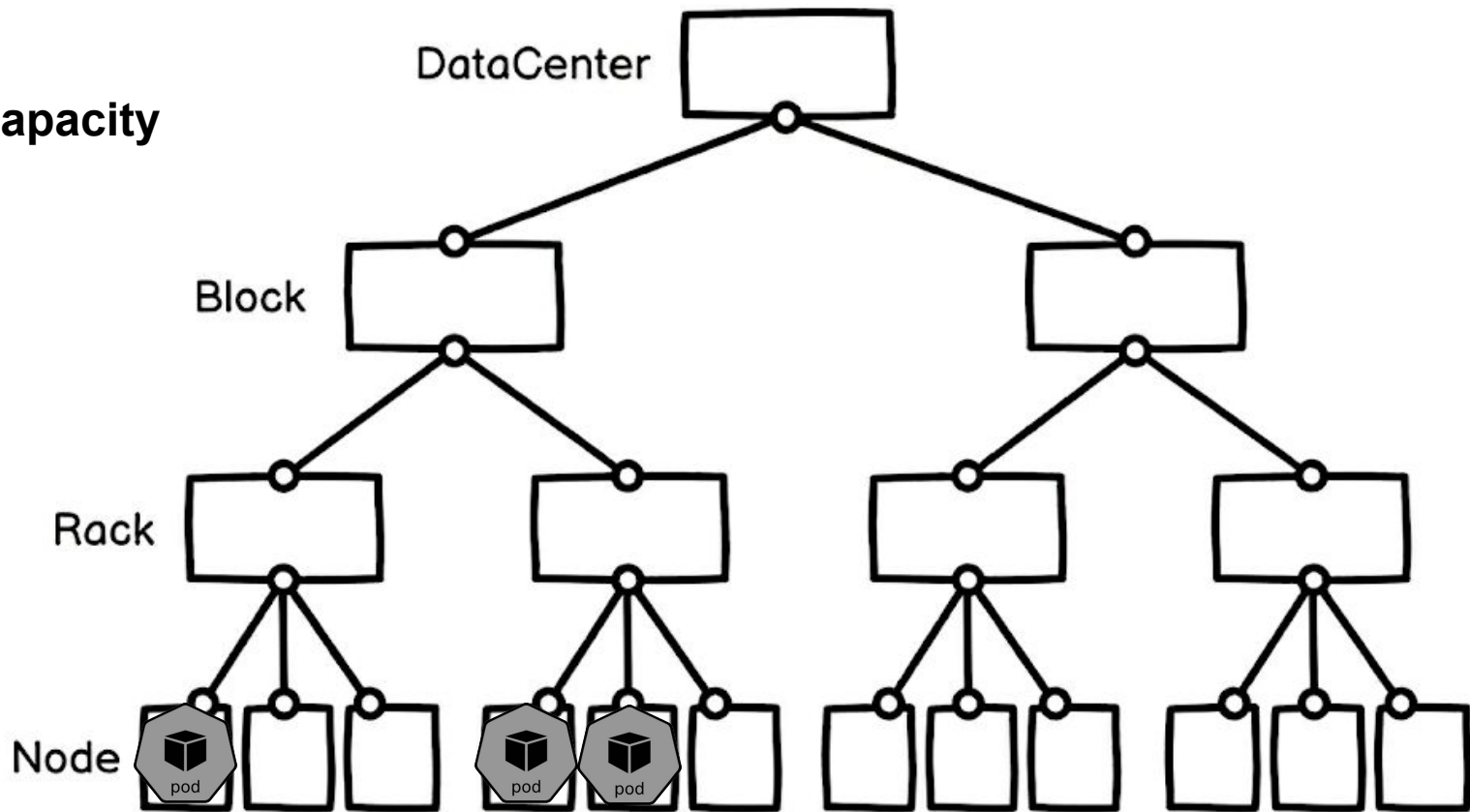
Kueue: Dive to TopologyAwareScheduling

- Find optimal placement for a Workload

Algorithm	Topology Preference Level	Kueue Feature Gate
BestFit	● Required ● Preferred	N/A
BestFit	● Unconstrained	TASProfileMixed=false
LeastFreeCapacity	● Unconstrained	TASProfileMixed=true (0.10.0, beta 0.15.0)
BalancedPlacement	● Preferred	TASBalancedPlacement=true (0.15.0)
MultilayerTopology	● Required ● Preferred	TASMultilayerTopology=true (0.17.0)

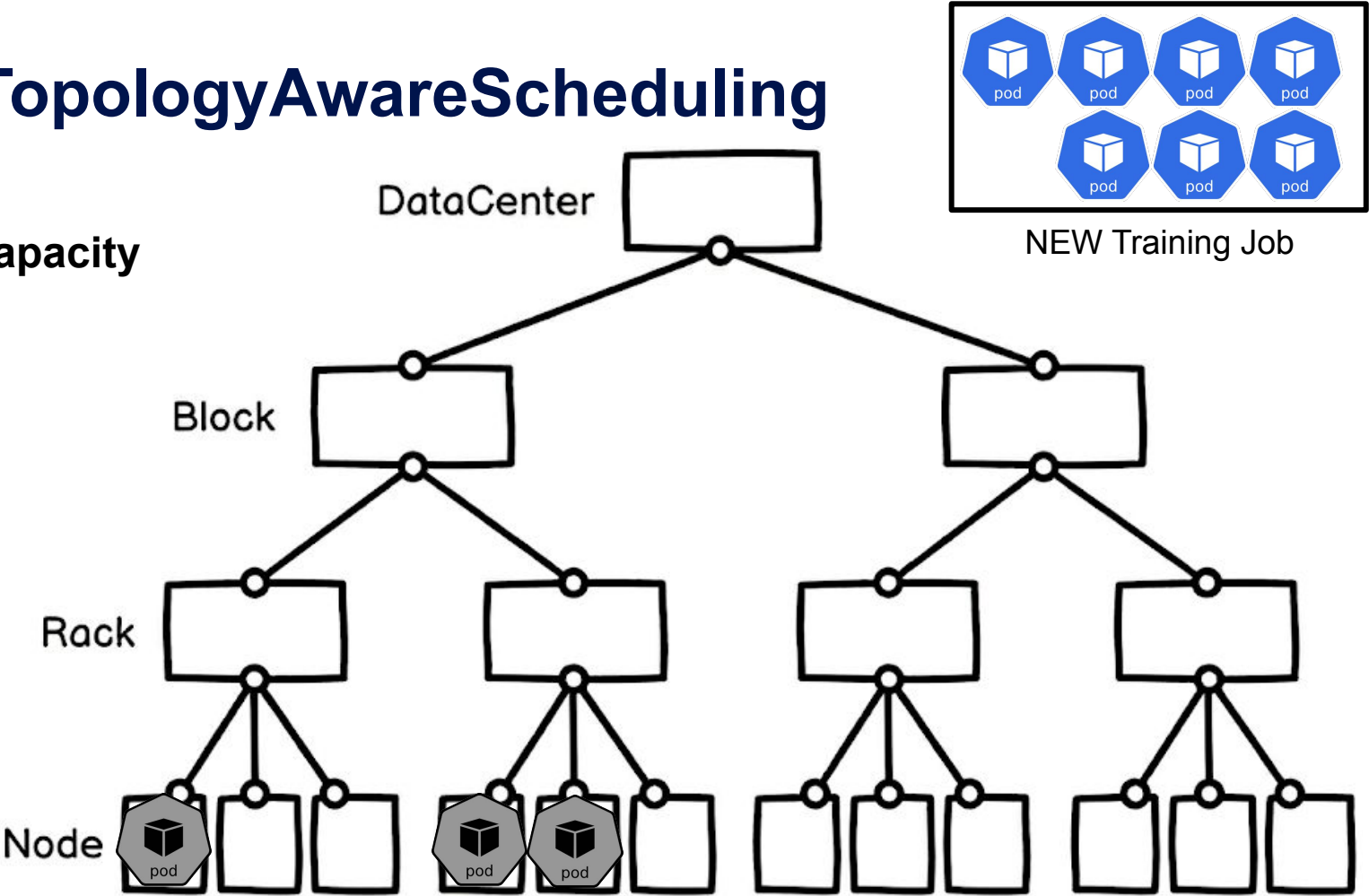
Kueue: TopologyAwareScheduling

LeastFreeCapacity



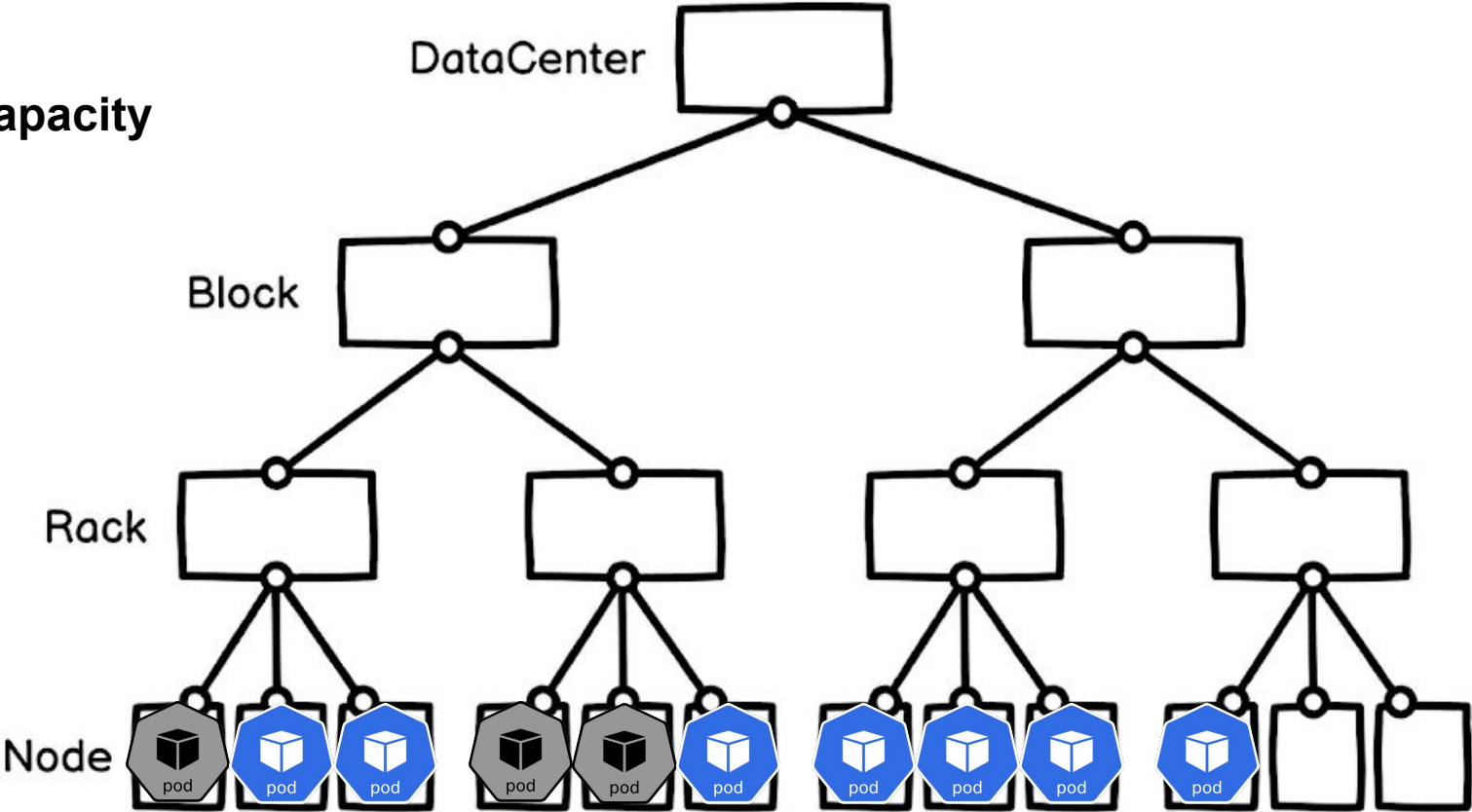
Kueue: TopologyAwareScheduling

LeastFreeCapacity



Kueue: TopologyAwareScheduling

LeastFreeCapacity



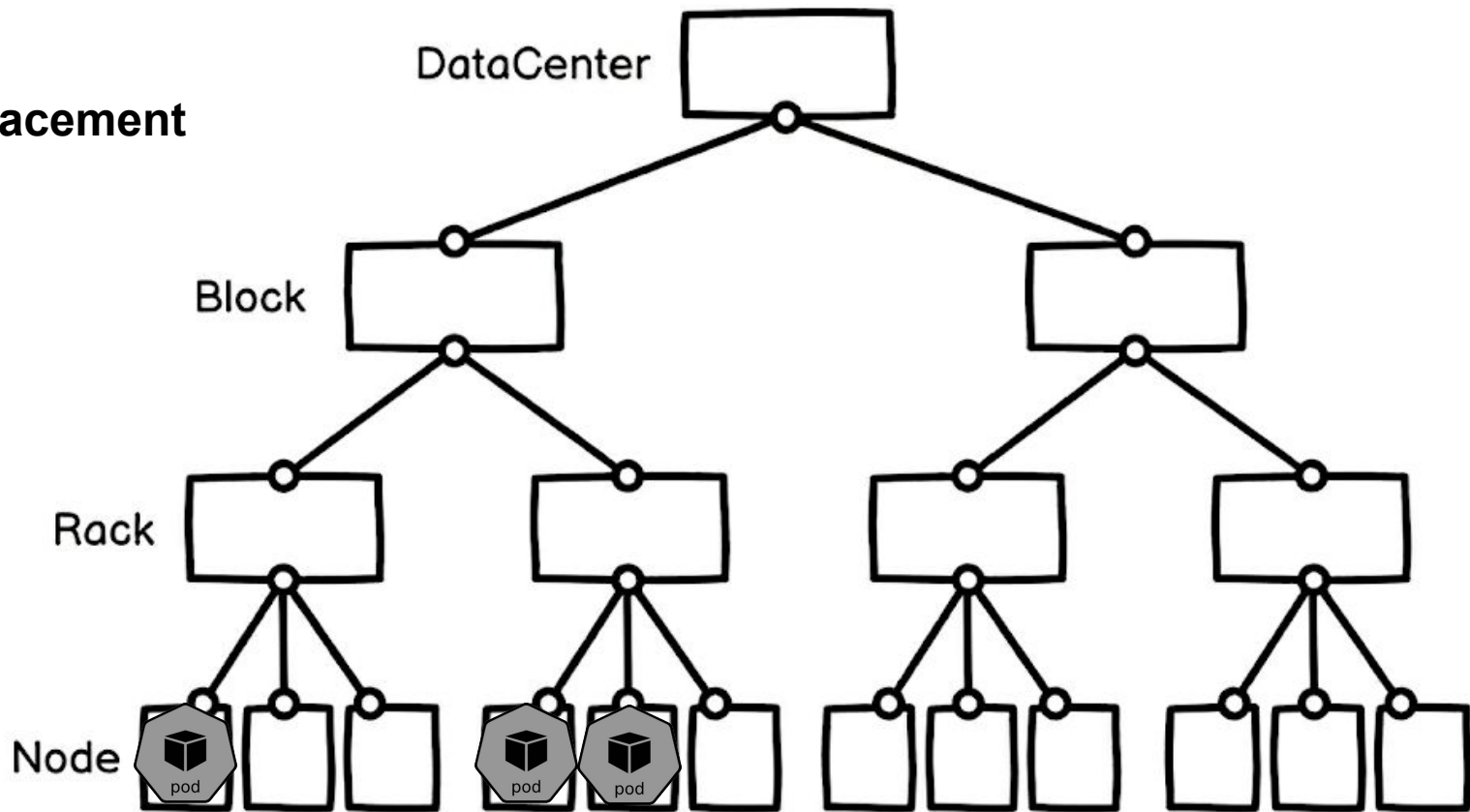
Kueue: Dive to TopologyAwareScheduling

- Find optimal placement for a Workload

Algorithm	Topology Preference Level	Kueue Feature Gate
BestFit	● Required ● Preferred	N/A
BestFit	● Unconstrained	TASProfileMixed=false
LeastFreeCapacity	● Unconstrained	TASProfileMixed=true (0.10.0, beta 0.15.0)
BalancedPlacement	● Preferred	TASBalancedPlacement=true (0.15.0)
MultilayerTopology	● Required ● Preferred	TASMultilayerTopology=true (0.17.0)

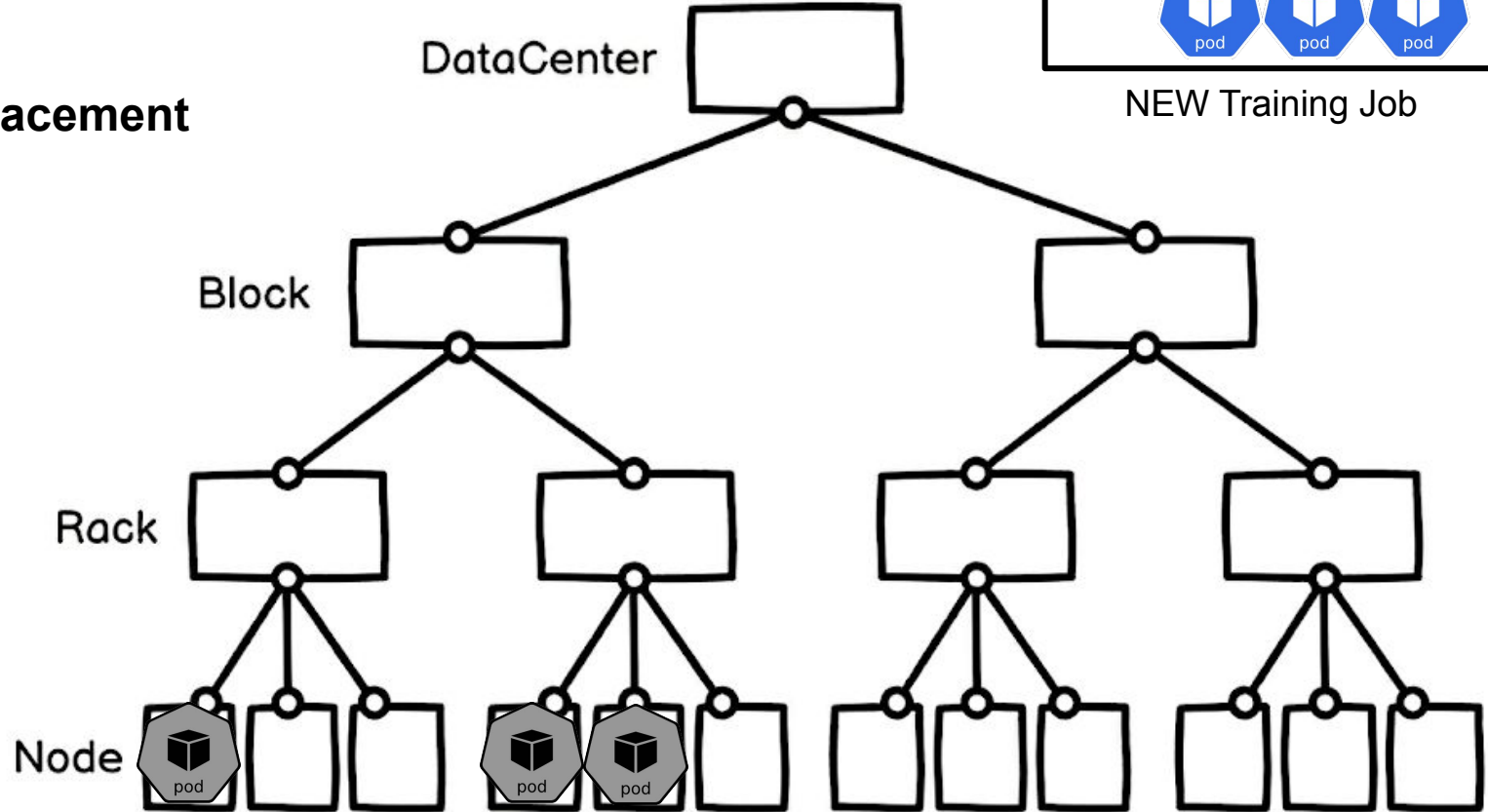
Kueue: TopologyAwareScheduling

BalancedPlacement



Kueue: TopologyAwareScheduling

BalancedPlacement



KubeCon

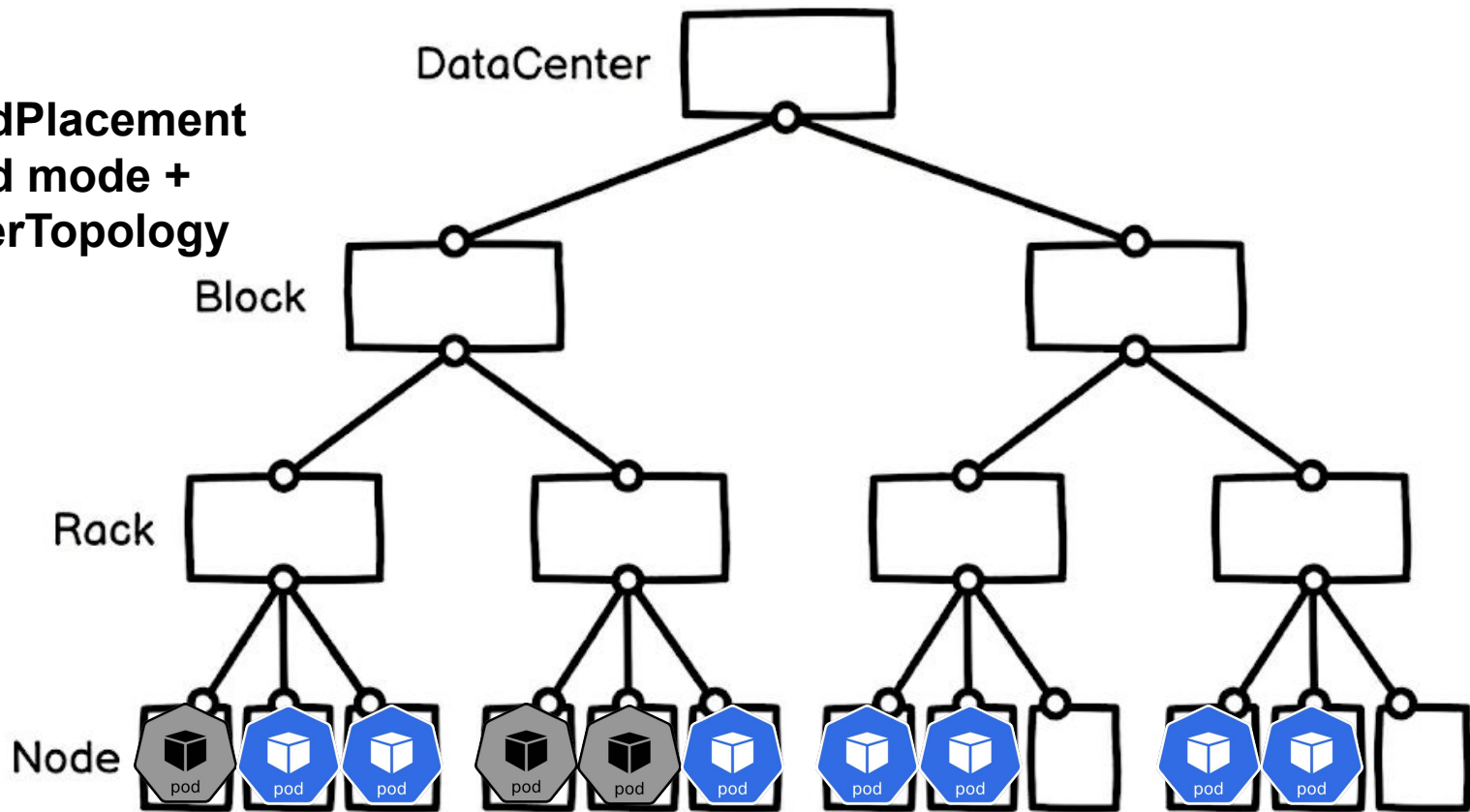


CloudNativeCon

Europe 2026

Kueue: TopologyAwareScheduling

- **BalancedPlacement**
- **Preferred mode + MultilayerTopology**



KubeCon

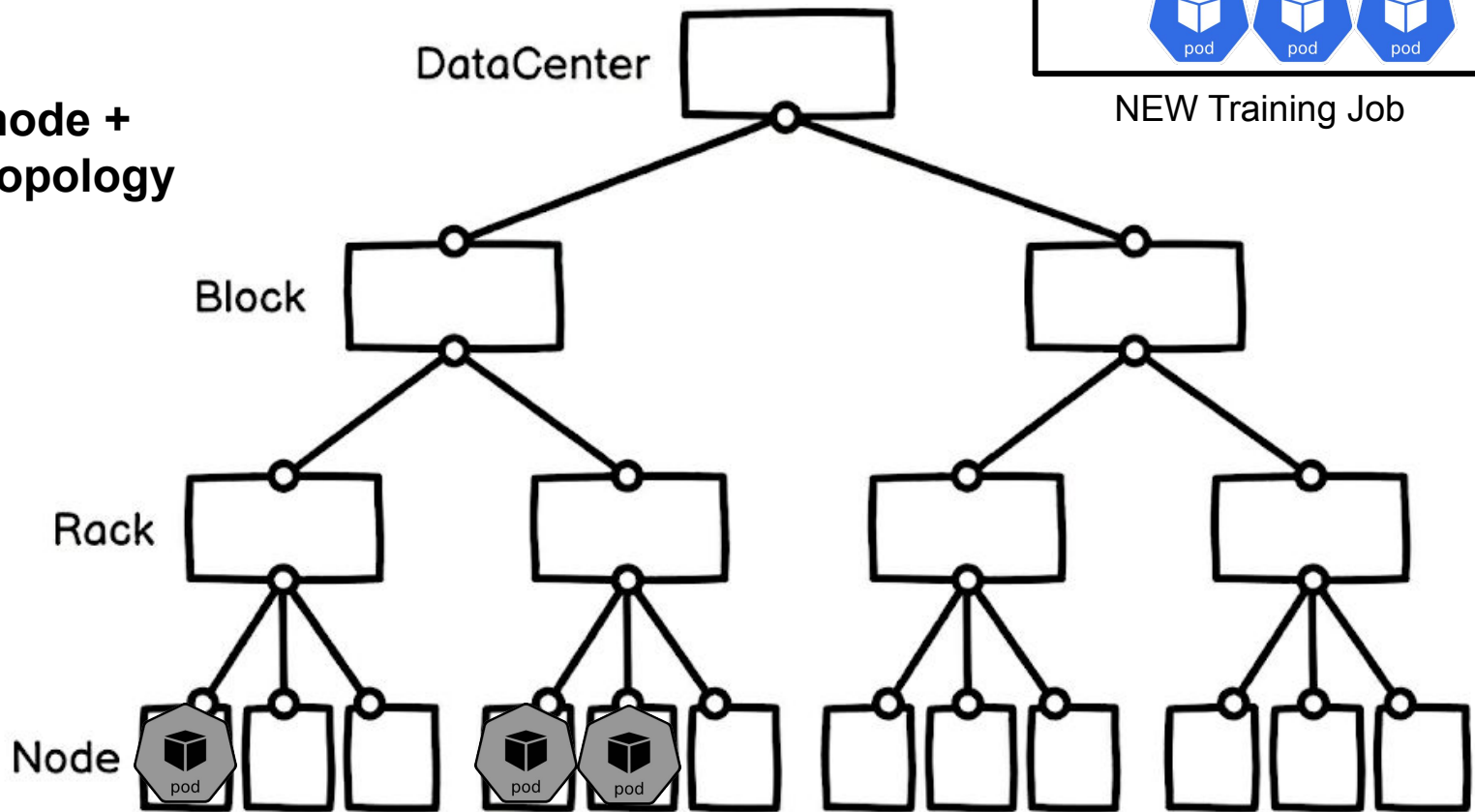


CloudNativeCon

Europe 2026

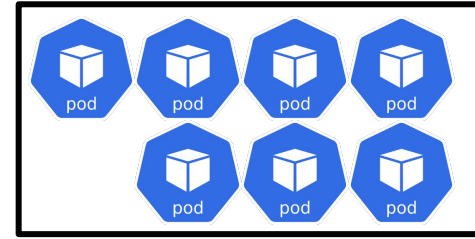
Kueue: TopologyAwareScheduling

Required mode +
MultilayerTopology

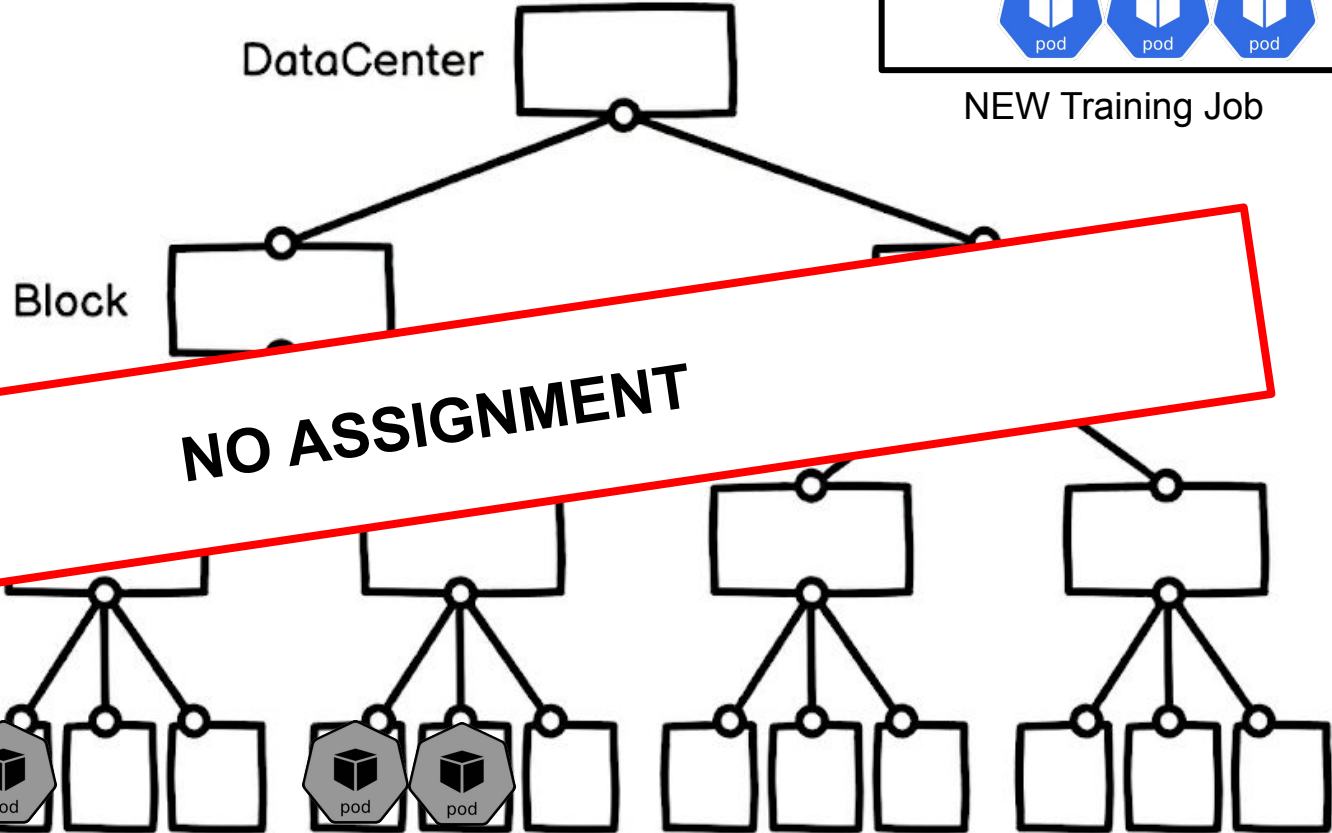


Kueue: TopologyAwareScheduling

Required mode +
MultilayerTopology



NEW Training Job



KubeCon

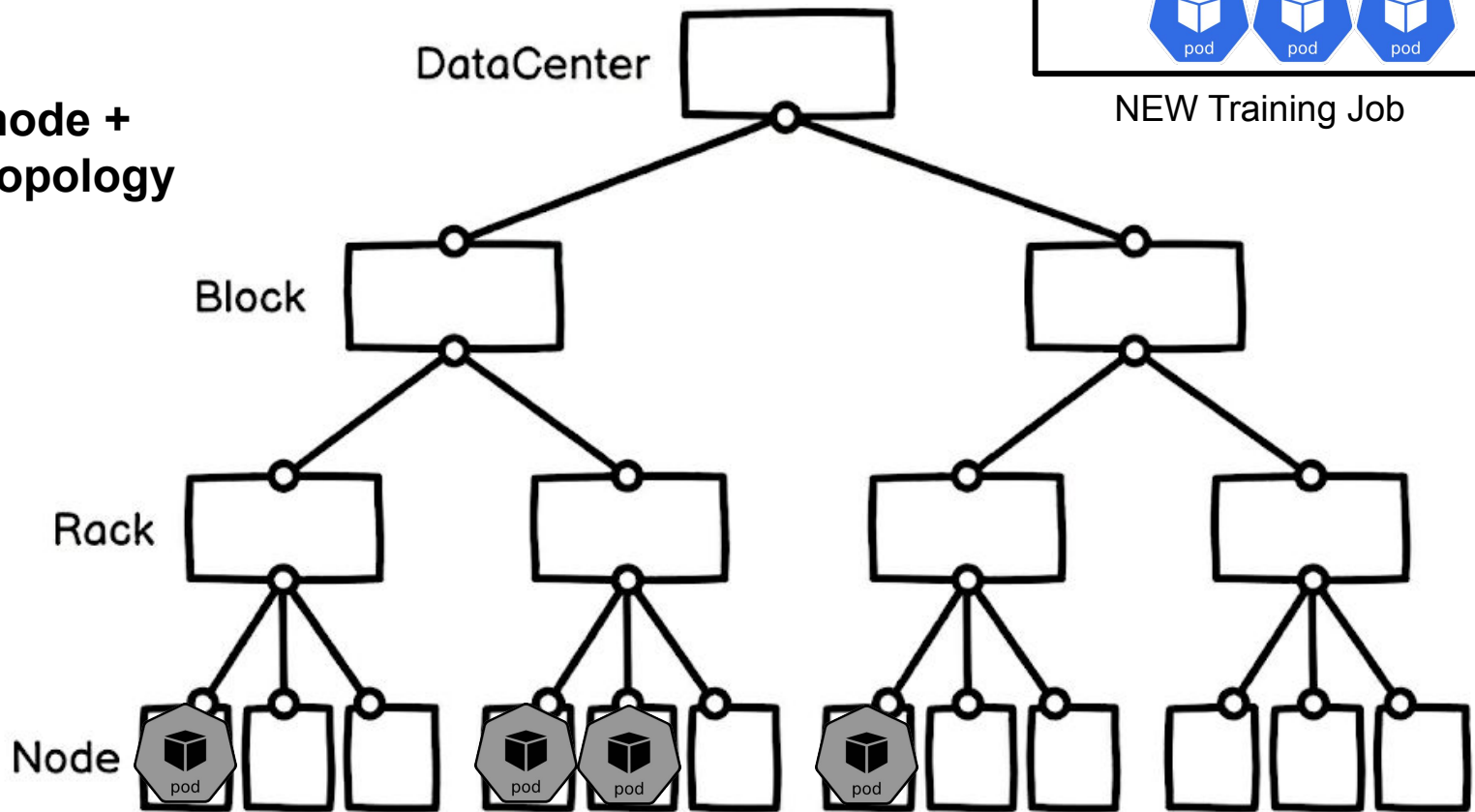


CloudNativeCon

Europe 2026

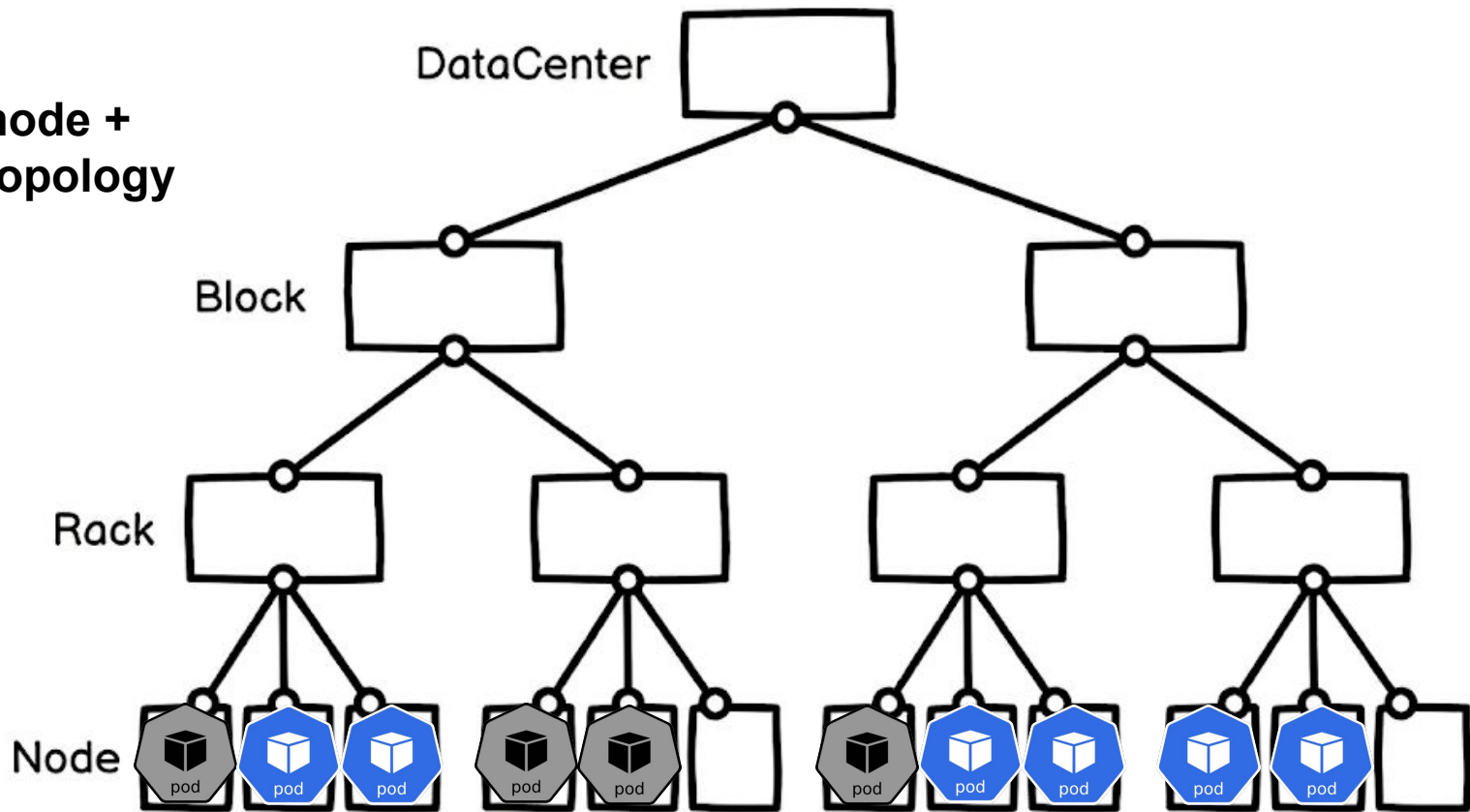
Kueue: TopologyAwareScheduling

Required mode +
MultilayerTopology



Kueue: TopologyAwareScheduling

Required mode +
MultilayerTopology



Kueue: Recent Highlights

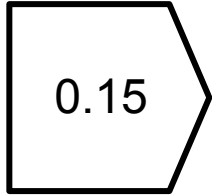
v0.15.0 - v0.15.7 and v0.16.0 - v0.16.4

- API: Graduate Kueue v1beta1 API to v1beta2 API (v0.15)
- DRA: Update DRA API from v1beta2 to v1
- TAS: Support for Kubeflow TrainJob
- TAS: Balanced placement algorithm
- TAS: LeastFreeCapacity algorithm for unconstrained mode workloads
- TAS: Extend TASReplaceNodeOnNodeTaints to handle NoSchedule and NoExecute taints
- MultiKueue: Support TAS and ProvisioningRequest integration
- MultiKueue: Authentication to worker clusters using ClusterProfile API

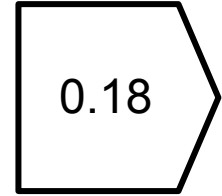
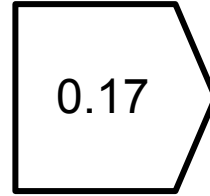
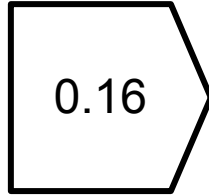
Kueue: Recent Highlights

v0.15.0 - v0.15.7 and v0.16.0 - v0.16.4

- **API: Graduate Kueue v1beta1 API to v1beta2 API (v0.15)**



Introduced v1beta2 v1beta2 as storageVersion



v1beta1 is Discontinued



KubeCon



CloudNativeCon

Europe 2026

Kueue: Recent Highlights

- Scheduling: Support mutating WorkloadPriorityClass label for Jobs with reserved quota
- Scheduling: Improve scheduling performance via SchedulingEquivalenceHashing, SchedulerTimestampPreemptionBuffer, and SchedulerLongRequeueInterval
- FailureRecovery: Introduce mechanism to terminate stuck Pods
- KueuePopulator: New experimental sub-project for default ClusterQueue, ResourceFlavor, Topology, and LocalQueue setup
- ElasticJobs: Support RayJob InTreeAutoscaling (ElasticJobsViaWorkloadSlices)

Kueue: Recent Highlights

- Observability: Counter and gauge metrics for finished workloads (kueue_finished_workloads_total, kueue_local_queue_finished_workloads_total)
- Security: TLS customization (TLSMinVersion, CipherSuites) for webhooks and visibility ondemand server
- Integrations: Pod and Pod-based integrations (Deployment, StatefulSet, LeaderWorkerSet) enabled by default

Kueue: Future Plans

0.17 (planned for 30th March):

- DRA: Alpha support for extended resources (.spec.extendedResourceName)
- TAS: Support for ElasticJobs (ElasticJobsViaWorkloadSlices)
- TAS: Support automatically Workload eviction when Nodes get NoSchedule or NoExecute nodeTaints
- TAS: Multi-Layer topology constraints
- MultiKueue: Support StatefulSet, LeaderWorkerSet, and RayService
- MultiKueue: Support Atomic Preemptions across clusters per Workload Admission

Kueue: Future Plans

- Scheduling: Support Priority Boost mechanism
- Scheduling: Support delayed queueing mechanism (AdmissionGatedBy)
- Integrations: RayService and SparkApplication is natively supported
- GA Graduation FeatureGates:
 - HierarchicalCohorts
 - LendingLimit
 - SanitizePodSets
 - LocalQueueDefaulting
 - ObjectRetentionPolicies
 - PropagateBatchJobLabelsToWorkload
 - MultiKueueBatchJobWithManagedBy



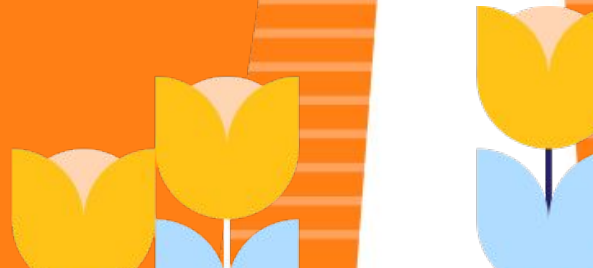
KubeCon



CloudNativeCon

Europe 2026

WAS / TAS



Gang Scheduling



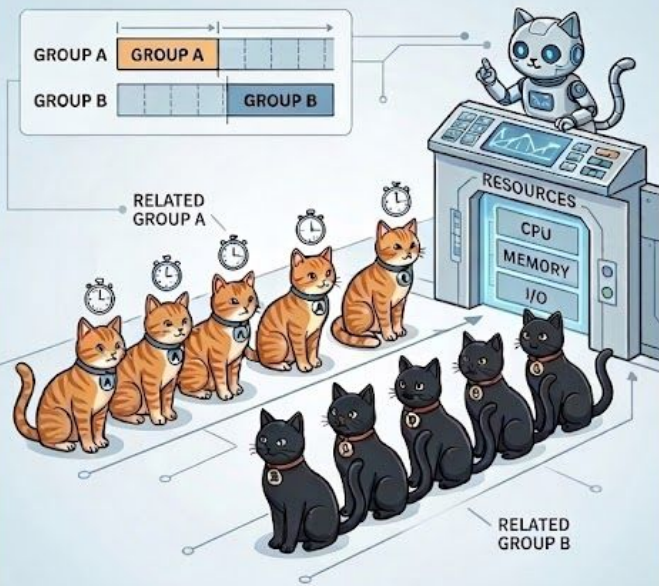
KubeCon



CloudNativeCon

Europe 2026

GANG SCHEDULING SYNC



NO GANG SCHEDULING CHAOS



Gang of Gang scheduling



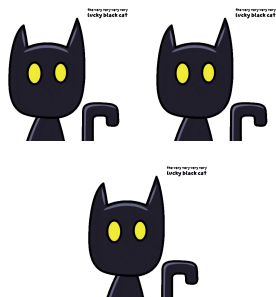
KubeCon



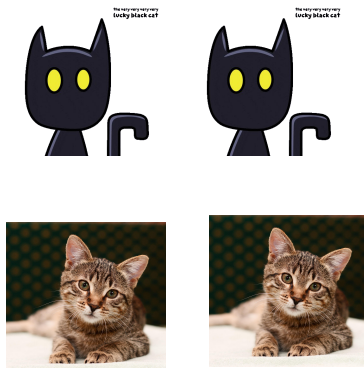
CloudNativeCon

Europe 2026

Job



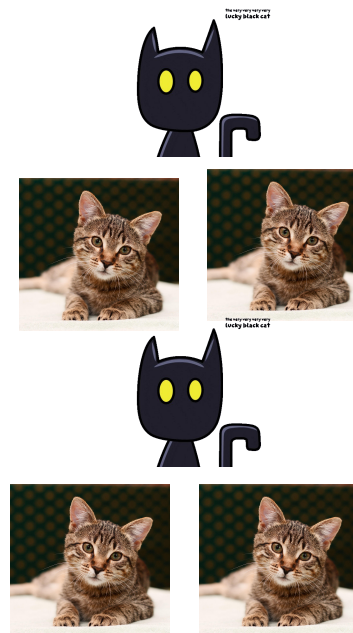
JobSet



leader

worker

Disaggregated
Serving



prefill-leader

prefill-worker

decoder-leader

decoder-worker

Gang Disruption



KubeCon



CloudNativeCon

Europe 2026



Workload Aware Scheduling



KubeCon



CloudNativeCon

Europe 2026

- Make Kubernetes aware of group of pods
- Two APIs: Workload and PodGroup

Workload

- Static definition
- MinCount
- No Status

PodGroup

- Runtime
- MinCount
- Status

Workload Aware Scheduling



KubeCon



CloudNativeCon

Europe 2026

Workload

```
apiVersion: scheduling.k8s.io/v1alpha2
kind: Workload
metadata:
  namespace: default
  name: job-1
spec:
  podGroupTemplates:
    - name: "job-1-template"
      schedulingPolicy:
        gang:
          minCount: 1000
```

PodGroup

```
apiVersion: scheduling.k8s.io/v1alpha2
kind: PodGroup
metadata:
  namespace: default
  name: job-1-template
spec:
  podGroupTemplateRef:
    workload:
      workloadName: job-1
      podGroupTemplateName: job-1-template
  schedulingPolicy:
    gang:
      minCount: 1000
```

Job

```
apiVersion: batch/v1
kind: Job
metadata:
  name: sleep-job
spec:
  parallelism: 1000
  completions: 1000
  spec:
    schedulingGroup:
      podGroupName: job-1-template
```

Workload aware Disruption

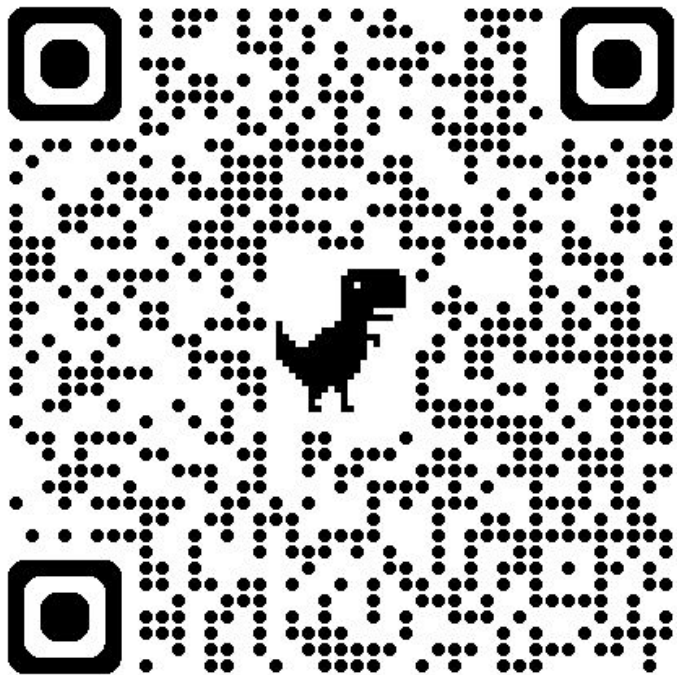


KubeCon



CloudNativeCon

Europe 2026



Scheduler is aware of PodGroups
and can preempt them all
together

```
// DisruptionMode describes the mode in which a PodGroup can be disrupted (e.g. preempted).
// +enum
type DisruptionMode string

const (
    // DisruptionModePod means that individual pods can be disrupted or preempted independently.
    // It doesn't depend on exact set of Pods currently running in this PodGroup.
    DisruptionModePod = "Pod"
    // DisruptionModePodGroup means that the whole PodGroup replica needs to be disrupted or
    // preempted together.
    DisruptionModePodGroup = "PodGroup"
)

type PodGroupSchedulingPolicy struct {
    // Existing field(s).

    // DisruptionMode defines the mode in which a given PodGroup can be disrupted.
    // One of Pod, PodGroup.
    // Defaults to Pod if unset.
    //
    // This field is immutable.
    DisruptionMode "DisruptionMode"
}
```

Workload aware Resource Claims



KubeCon



CloudNativeCon

Europe 2026

What would you like to be added:

I would like to be able to associate a ResourceClaimTemplate with a Job in the JobSet, such that each time the Job is instantiated, a new ResourceClaim is created based on the template, and all Pods in the Job are pointed to that ResourceClaim.

Why is this needed:

ResourceClaimTemplates in DRA are used to create ResourceClaims that follow a Pod's lifecycle. For example, it allows you to use a Deployment, and when the Deployment creates a Pod, a new ResourceClaim is also created for that Pod in order to get it a GPU.

```
type PodGroupTemplate struct {
    ...

    // ResourceClaims defines which ResourceClaims may be shared among Pods in
    // the group. Pods must reference these claims in order to consume the
    // allocated devices.
    //
    // This is an alpha-level field and requires that the
    // WorkloadPodGroupResourceClaimTemplate feature gate is enabled.
    //
    // This field is immutable.
    //
    ResourceClaims []PodGroupResourceClaim `json:"resourceClaims,omitempty"`
}

type PodGroupResourceClaim struct {
    Name string `json:"name"`

    ResourceClaimName *string `json:"resourceClaimName,omitempty"`

    ResourceClaimTemplateName *string `json:"resourceClaimTemplateName,omitempty"`
}
```



Workload Aware Controllers



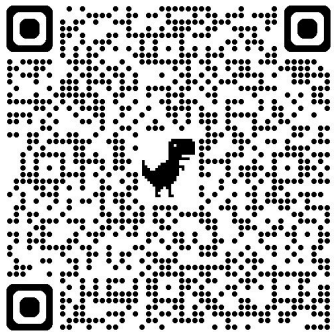
KubeCon



CloudNativeCon

Europe 2026

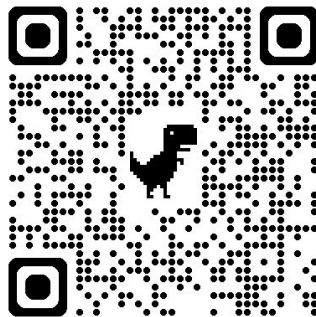
- Step 1: Create a Workload
- Step 2: Create a podgroup from the Workload PodGroupTemplate
- Step 3: link pod specs to podGroupRef
- Scheduler will obey pod group specifications for that pod



Job



JobSet



KubeRay

Roadmap



KubeCon



CloudNativeCon

Europe 2026

In Design

Feature	Alpha	Beta
<u>Gang Scheduling</u>	1.36	1.37
<u>Resource Claim Support for Workloads</u>	1.36	1.37
<u>Topology Aware Scheduling</u>	1.36	1.37
<u>Workload Aware Disruption</u>	1.36	1.37
<u>Gang Jobs</u>	1.36	1.38

Feature
<u>Gang JobSets</u>
<u>Gang KubeRay</u>
Kueue awareness of WAS
Autoscaling and WAS
PodSubGroups

Demo



KubeCon



CloudNativeCon

Europe 2026

Examples



What to do next?



KubeCon



CloudNativeCon

Europe 2026

- Join us at #workload-aware-scheduling in k8s slack
- Try out the features with our controllers on 1.35/1.36
- Kueue + WAS/TAS was at solutions showcase
 - P-4B
- Or join as #wg-batch!



KubeCon



CloudNativeCon

Europe 2026



Back up demo



KubeCon



CloudNativeCon

Europe 2026

