



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

SIG Scheduling Update: Transition From Pod To Workload Scheduling

Kensei Nakada (Independent), Maciej Skoczeń (Google)



SIG Scheduling

SIG Scheduling is responsible for the components that make Pod placement decisions.



kube-scheduler



kueue



descheduler



kwok

Scheduler

The scheduler is the component that decides the **pod placement**, which implements

Resource

PodAffinity/AntiAffinity

NodeAffinity

Image Locality

Ports

Taint/Toleration

etc etc...

Scheduler plugins

The scheduler is composed of **plugins** and each plugin has a dedicated responsibility:

Resource Fit Plugin

Inter-Pod Affinity Plugin

NodeAffinity Plugin

Image Locality Plugin

NodePorts Plugin

TaintToleration Plugin

etc etc...

Extension points - Filter / Score

Filter and **score** are two main points involved in the scheduling decision.

Filter

Each filter plugin filters out nodes that **shouldn't** run the pods.

For example, nodes that don't have enough resources, don't match the node selector, etc.

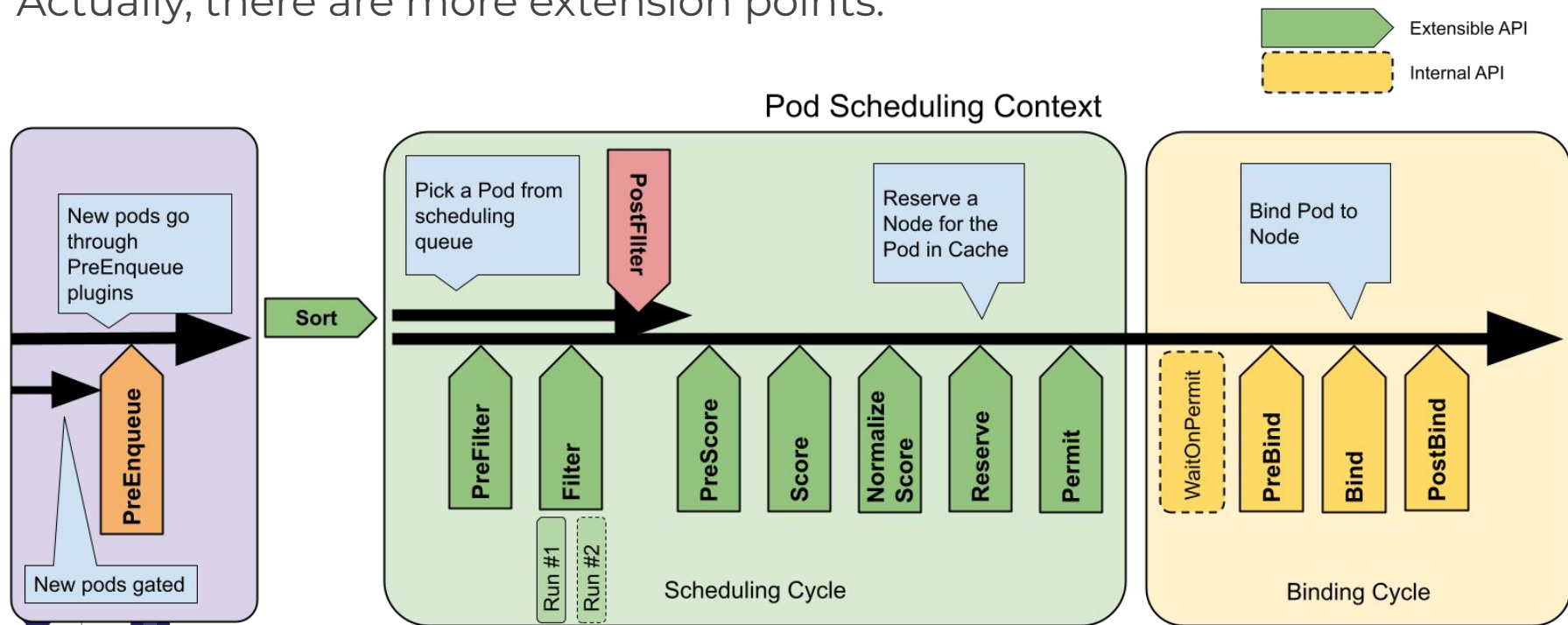
Score

Each score plugin **scores** nodes based on the preference.

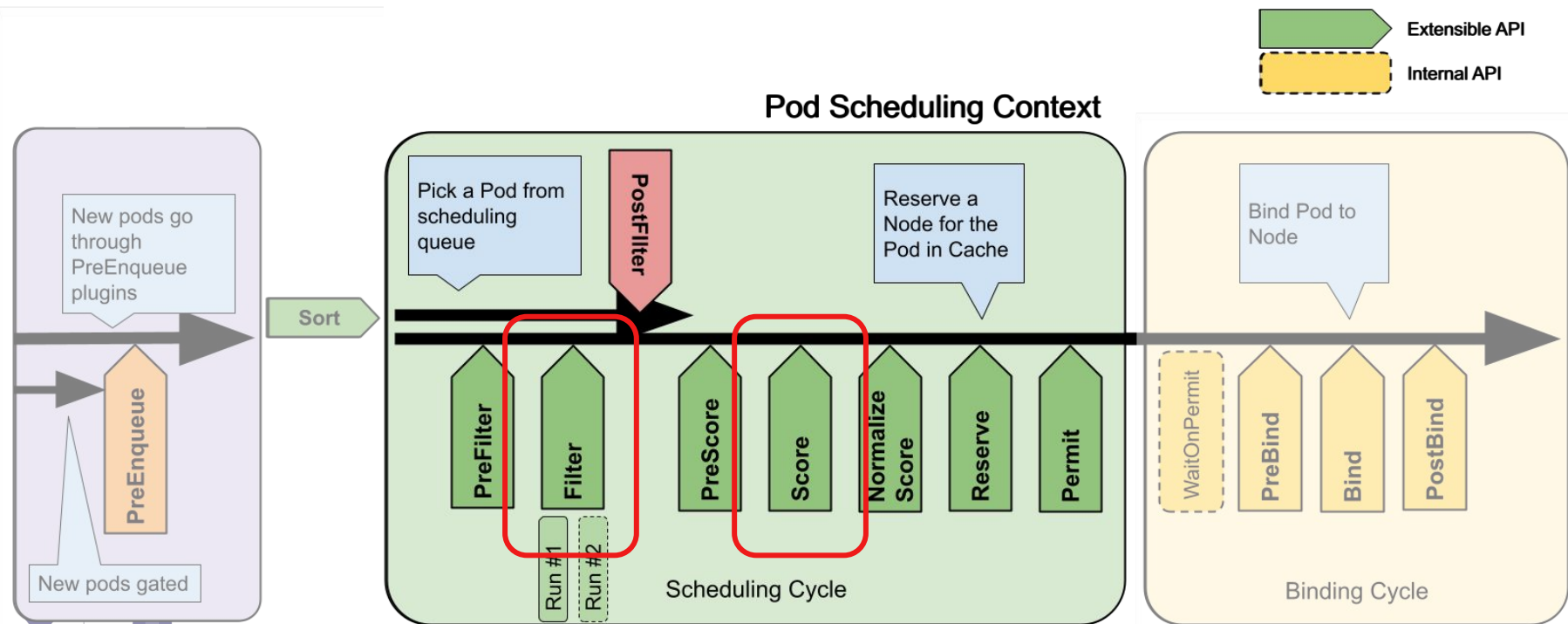
For example, nodes that have the pod's image in the cache get higher score, etc.

Scheduling Framework

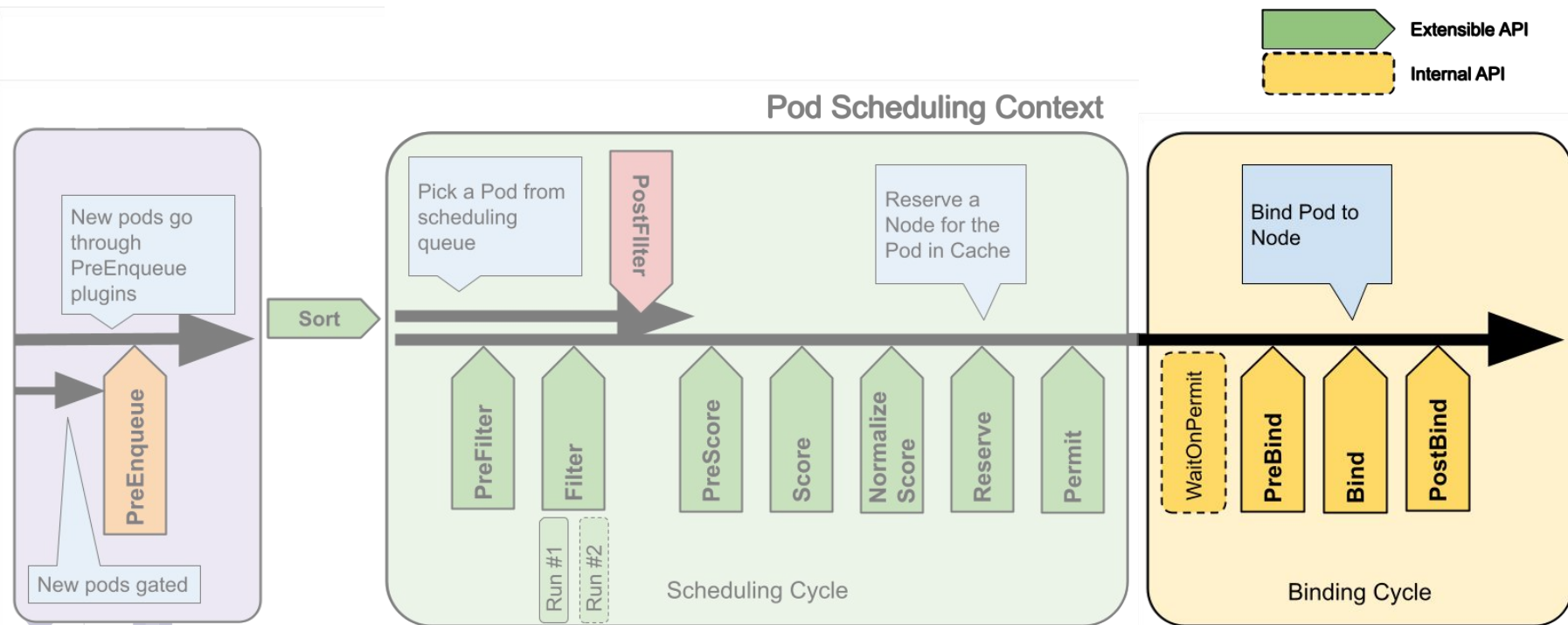
Actually, there are more extension points:



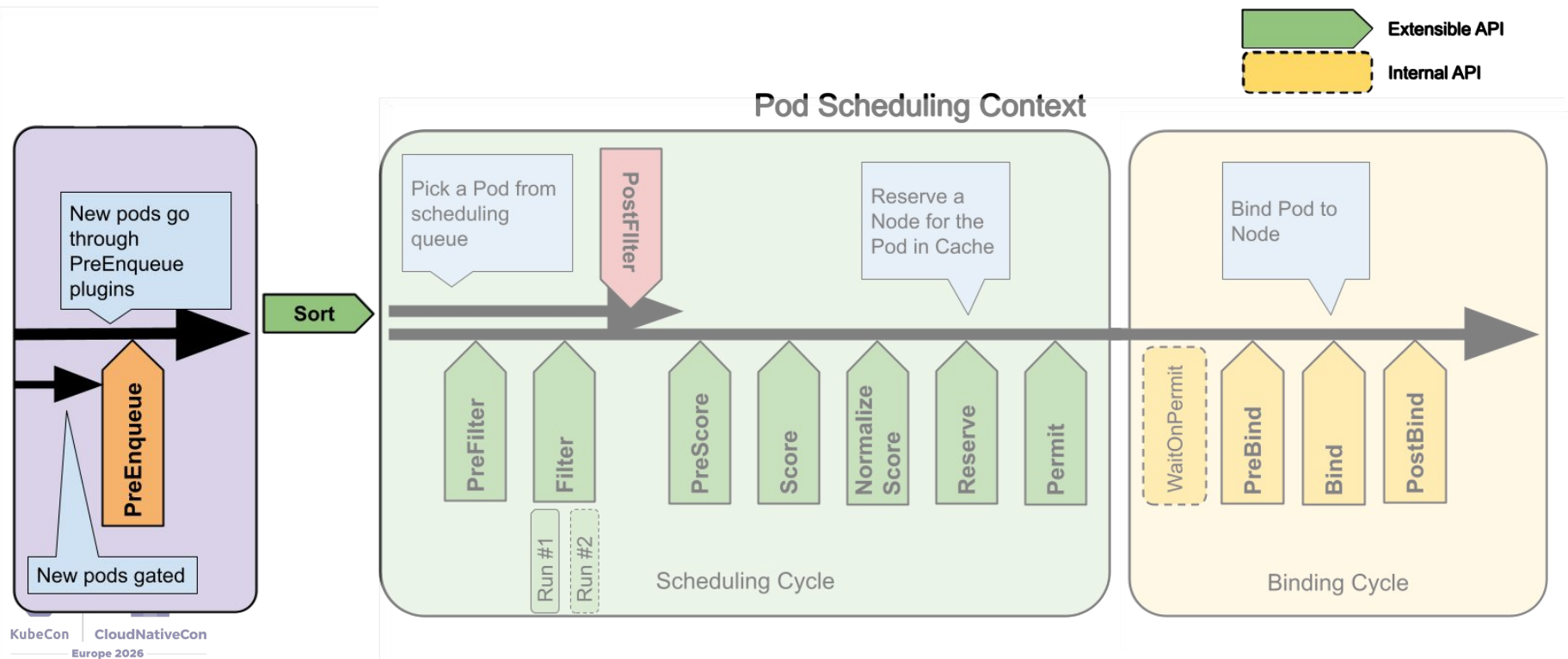
Scheduling Framework



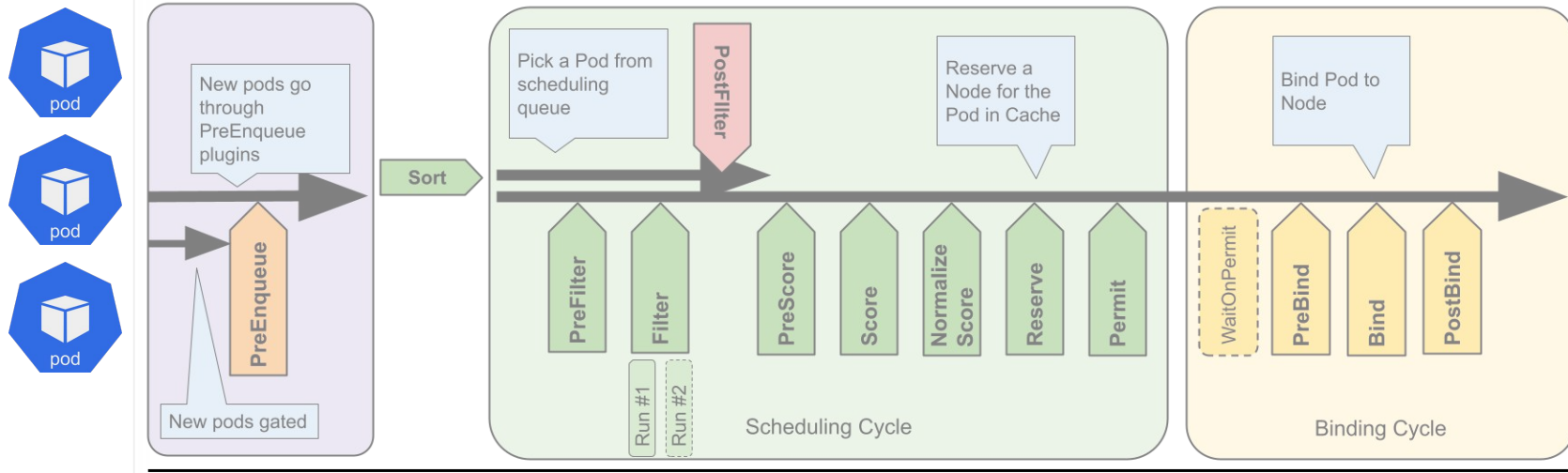
Scheduling Framework



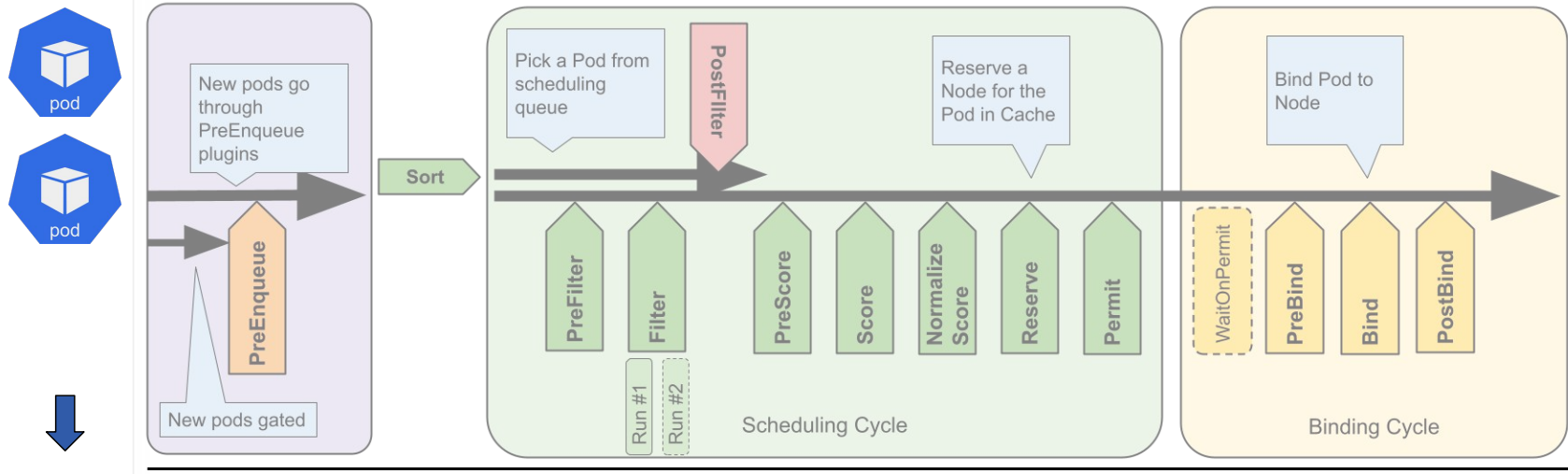
Scheduling Framework



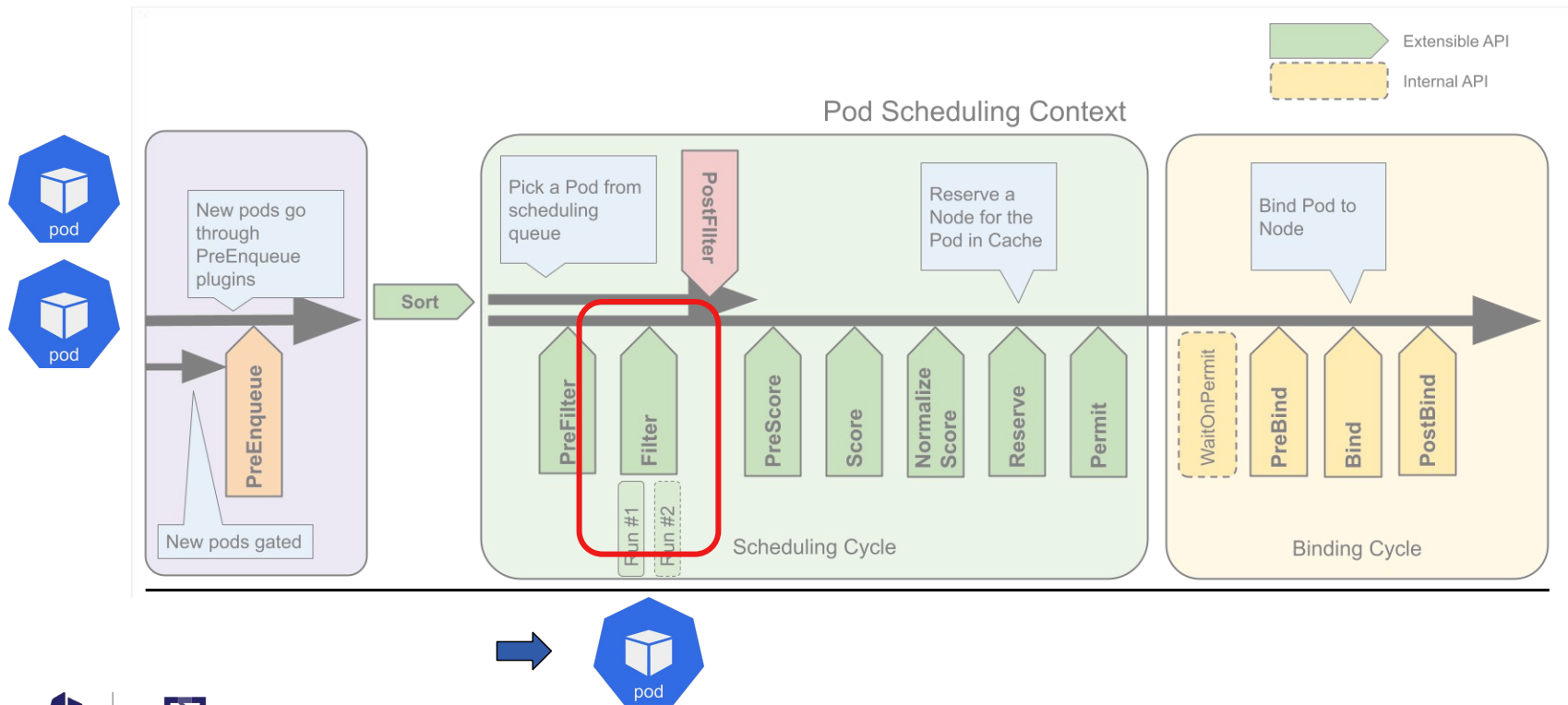
Pod-by-pod scheduling



Pod-by-pod scheduling



Pod-by-pod scheduling



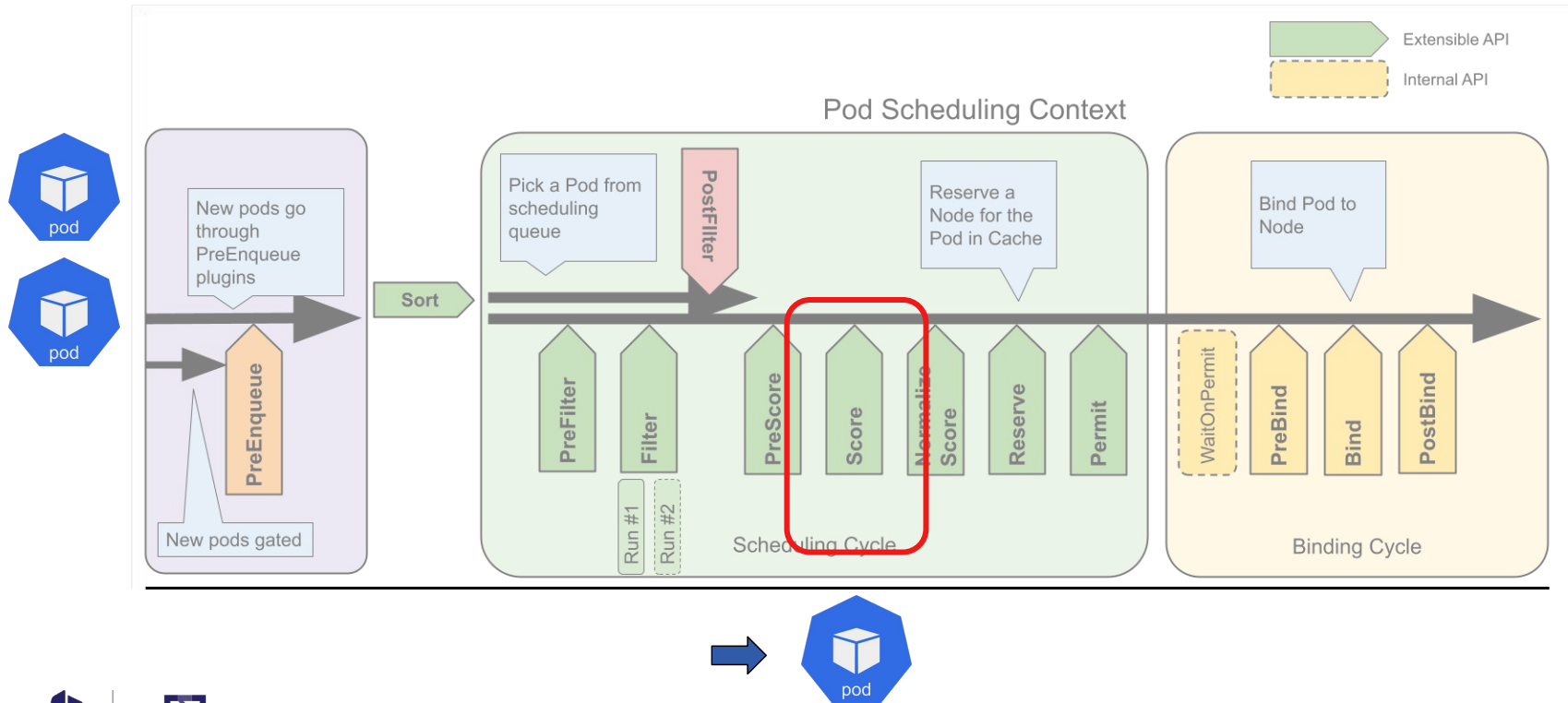
KubeCon



CloudNativeCon

Europe 2026

Pod-by-pod scheduling



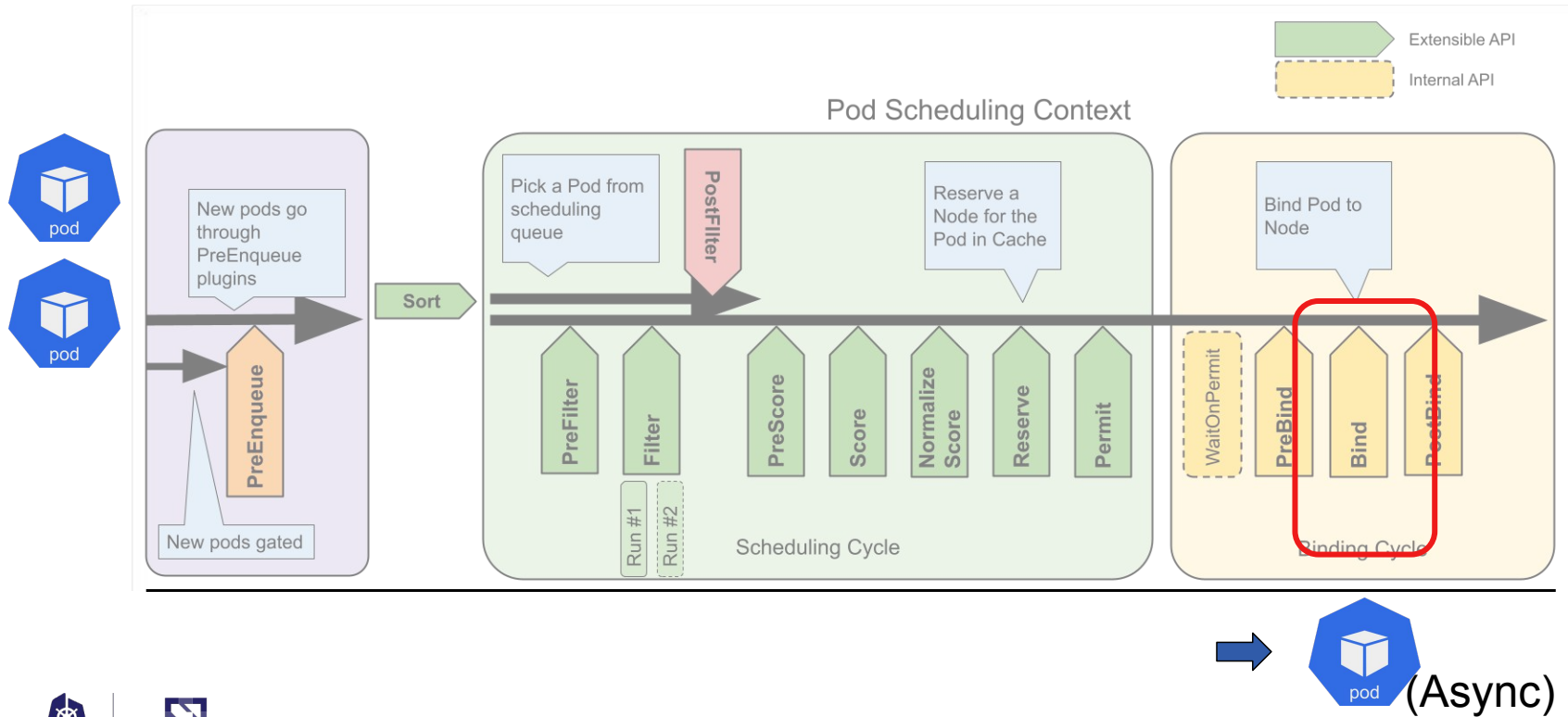
KubeCon



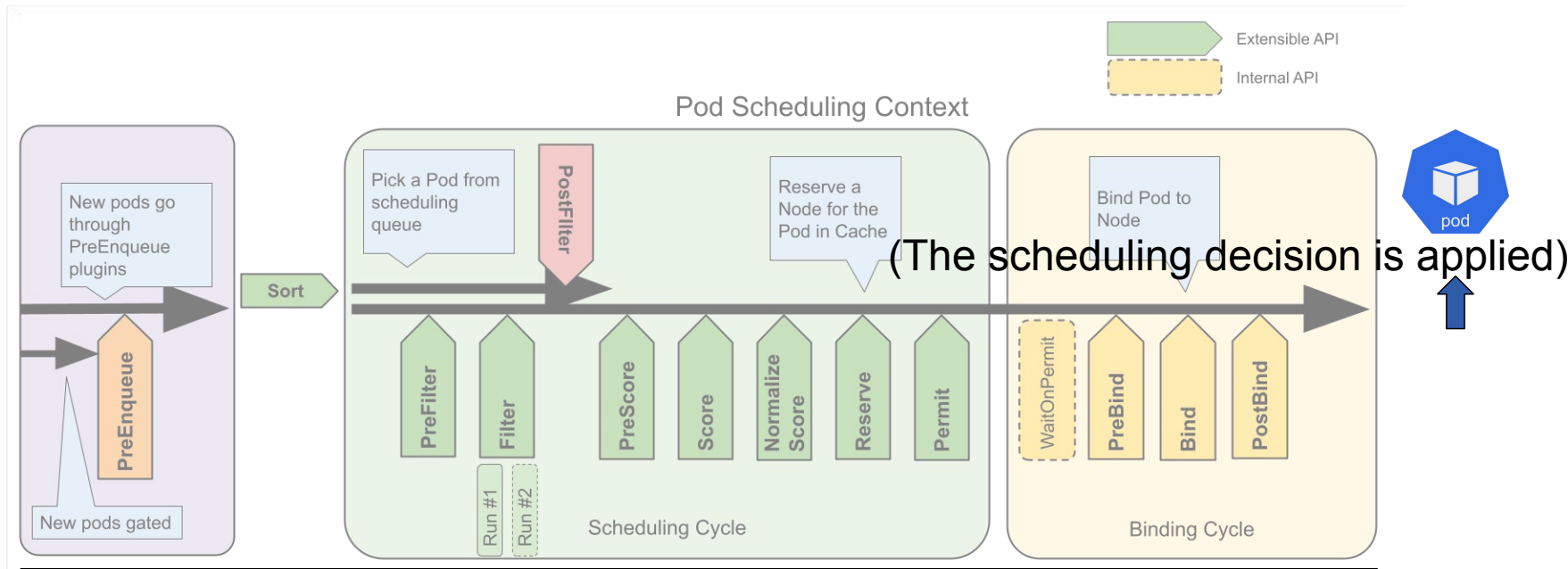
CloudNativeCon

Europe 2026

Pod-by-pod scheduling



Pod-by-pod scheduling



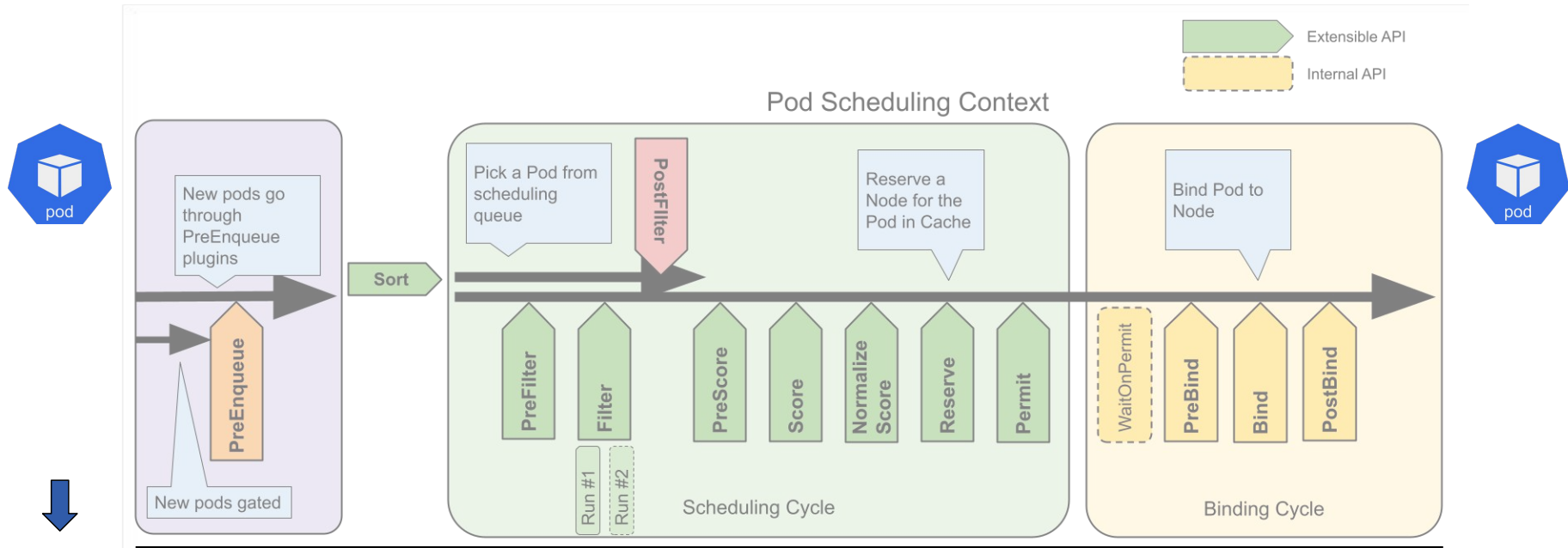
KubeCon



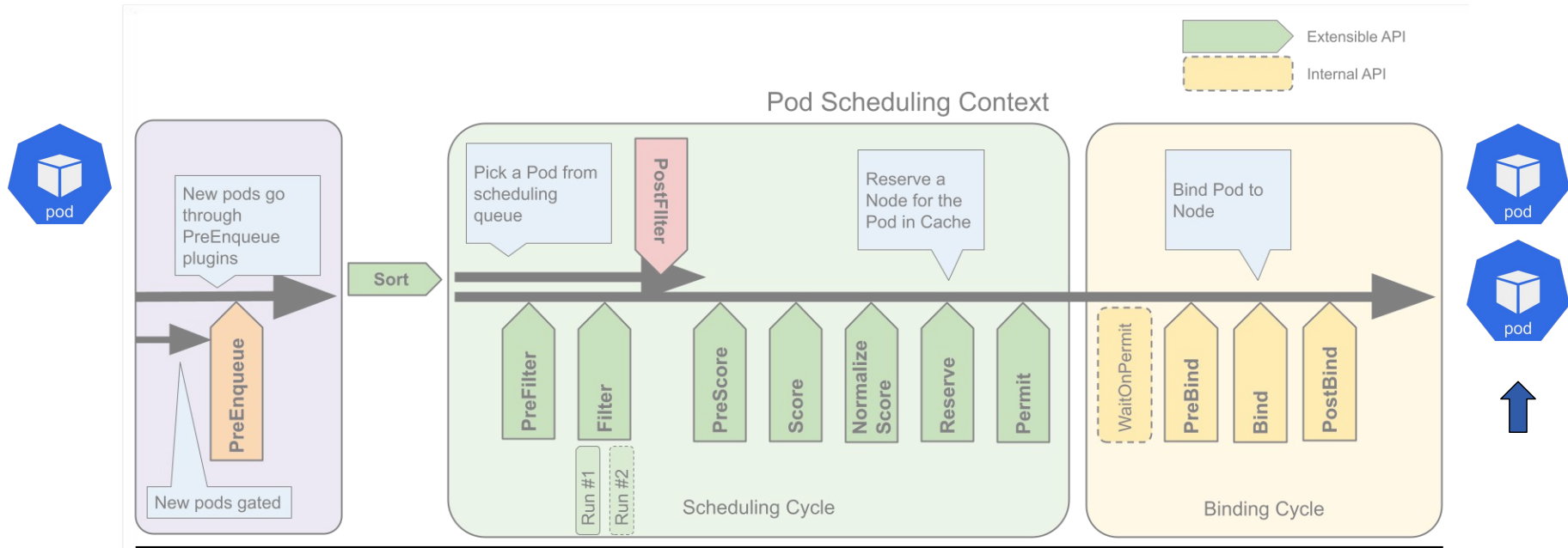
CloudNativeCon

Europe 2026

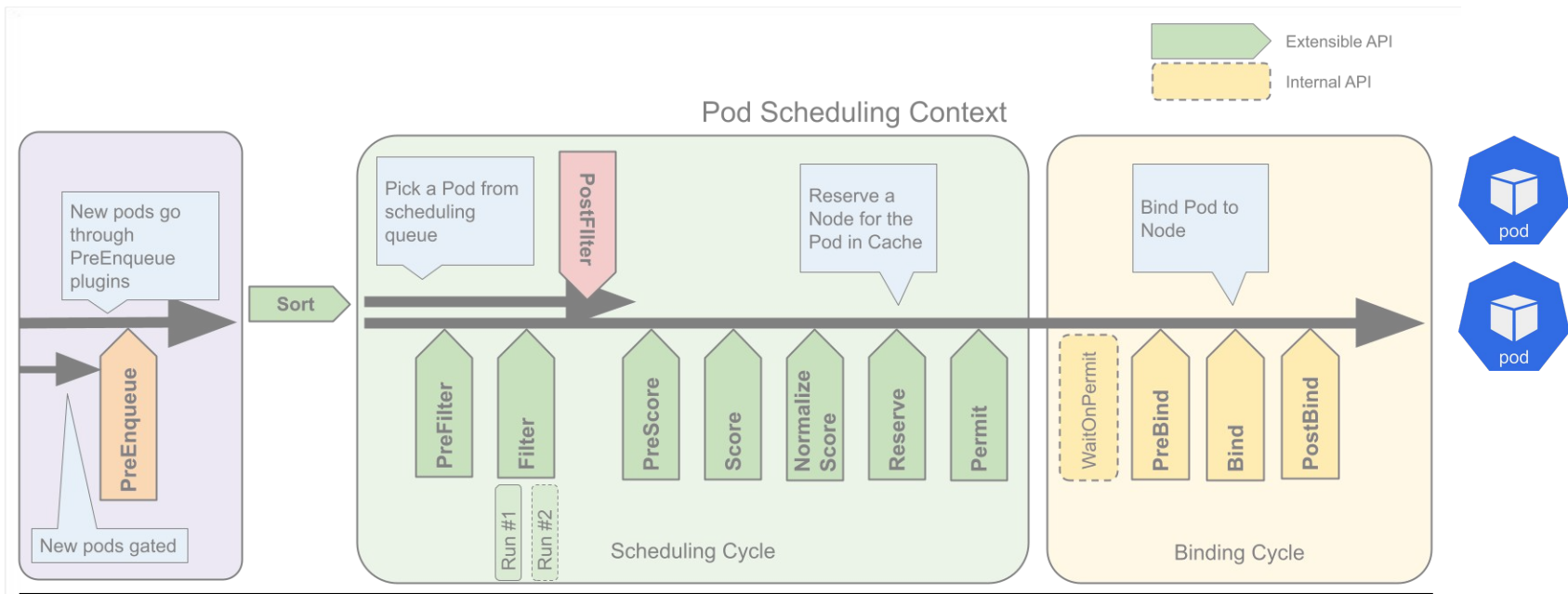
Pod-by-pod scheduling



Pod-by-pod scheduling



Pod-by-pod scheduling

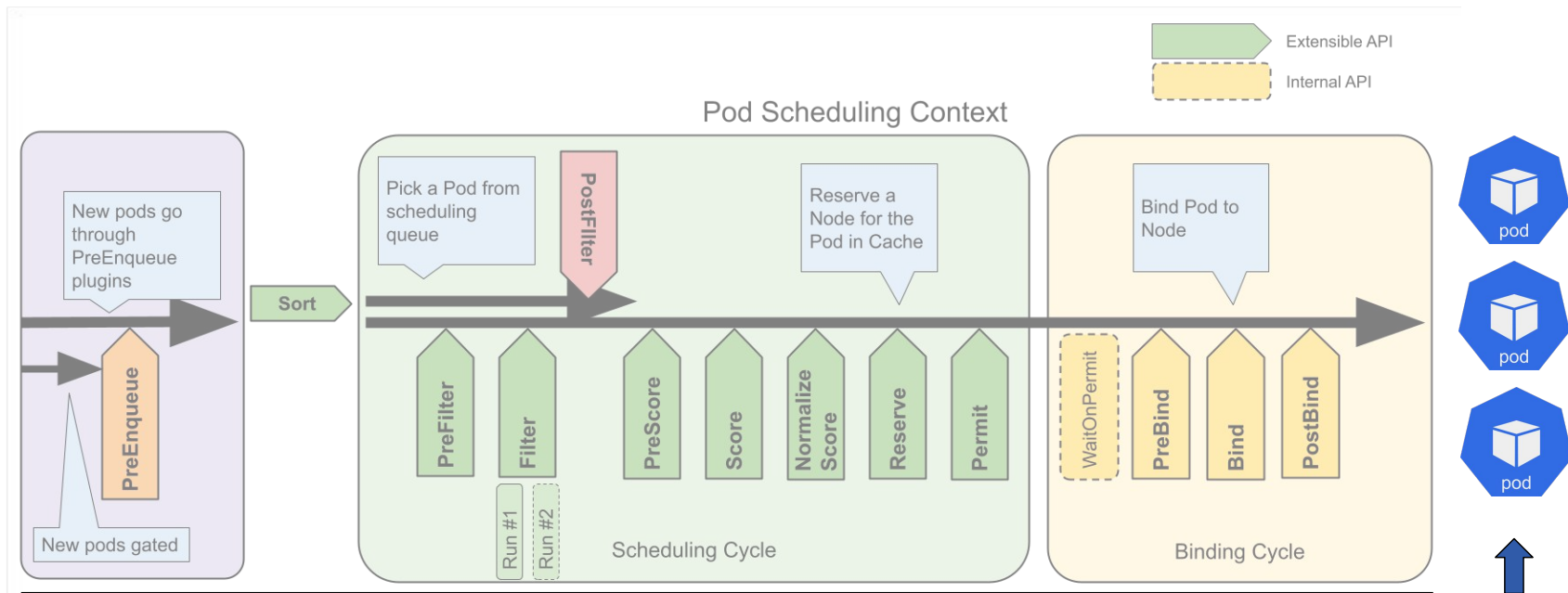


KubeCon

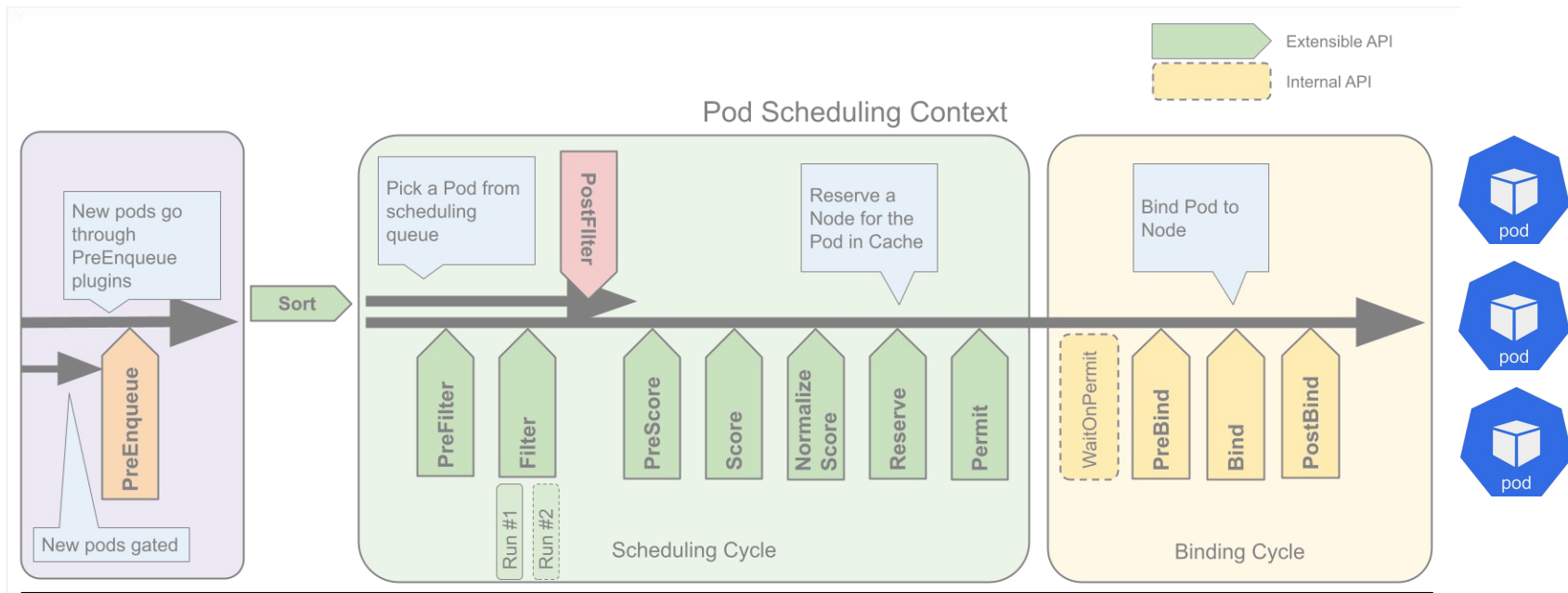
CloudNativeCon

Europe 2026

Pod-by-pod scheduling



Pod-by-pod scheduling





KubeCon



CloudNativeCon

Europe 2026

Workload-Aware Scheduling

Workload-Aware Scheduling

- Kicked off the Kubernetes-wide effort in v1.35
- Initial implementation of Workload API and Gang Scheduling in v1.35 (KEP-4671)
- Implemented Opportunistic Batching to reuse scheduling results for similar pods in v1.35 (KEP-5598)

Workload-Aware Scheduling

More WAS features implemented in v1.36 (all in alpha in v1.36):

- Workload API and Gang Scheduling (KEP-4671)
 - Changed the API structure in Decouple PodGroup API (KEP-5832)
 - Extended scheduler-side implementation
- Topology-Aware Scheduling (KEP-5732)
- Workload-Aware Preemption (KEP-5710)
- ResourceClaim support for PodGroups (KEP-5729)
- Job controller integration (KEP-5547)

Initial Workload API (v1.35) - v1alpha1

Workload API (Single object)

- The pod grouping API that defined the scheduling intent with ~none runtime support (scalability concerns)
- Consisted of PodGroups that defined the scheduling policies
- Pods had to reference the specific PodGroup instance indirectly, through the Workload object

```
apiVersion: scheduling.k8s.io/v1alpha1
kind: Workload
metadata:
  name: training-job-workload
  namespace: some-ns
spec:
  podGroups:
  - name: workers
    policy:
      gang:
        # The gang is schedulable only
        minCount: 4
```

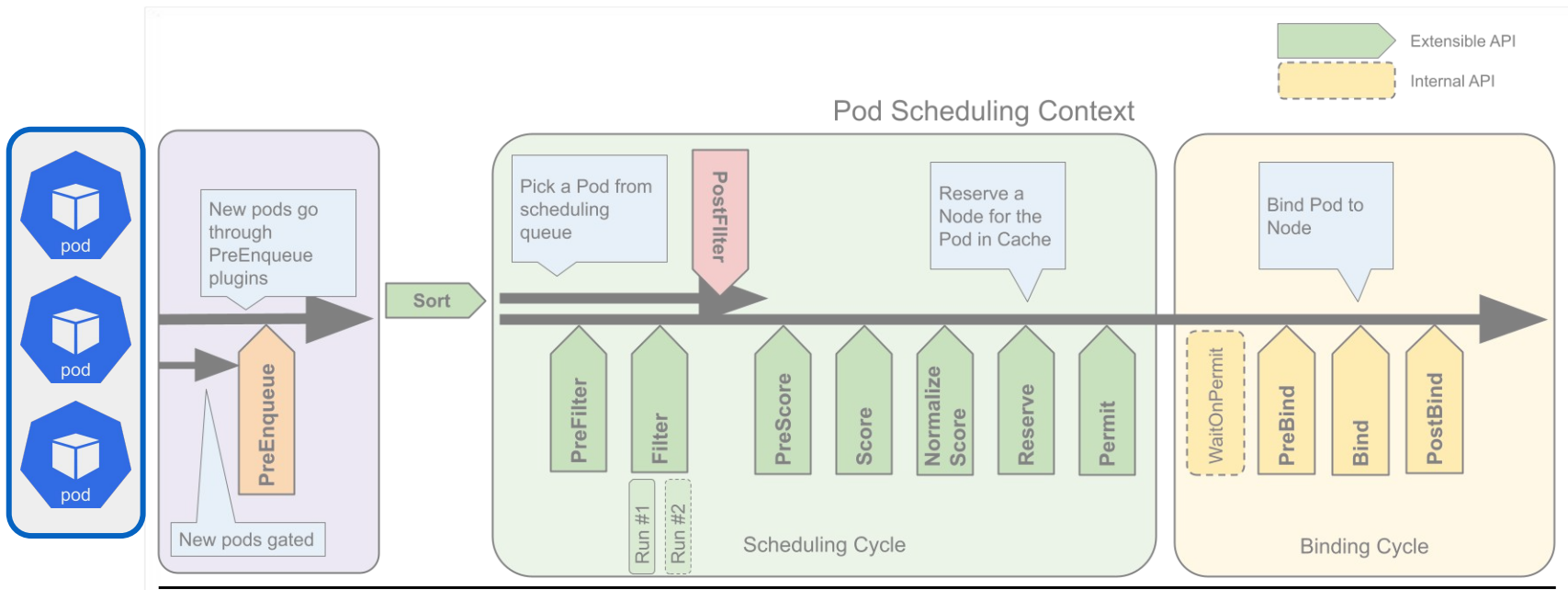
Initial Workload API (v1.35) - v1alpha1

Workload API (Single object)

- The pod grouping API that defined the scheduling intent with ~none runtime support (scalability concerns)
- Consisted of PodGroups that defined the scheduling policies
- Pods had to reference the specific PodGroup instance indirectly, through the Workload object

```
apiVersion: v1
kind: Pod
metadata:
  name: worker-0
  namespace: some-ns
spec:
  workloadRef:
    name: training-job-workload
    podGroup: workers
...
```

Gang Scheduling (v1.35)



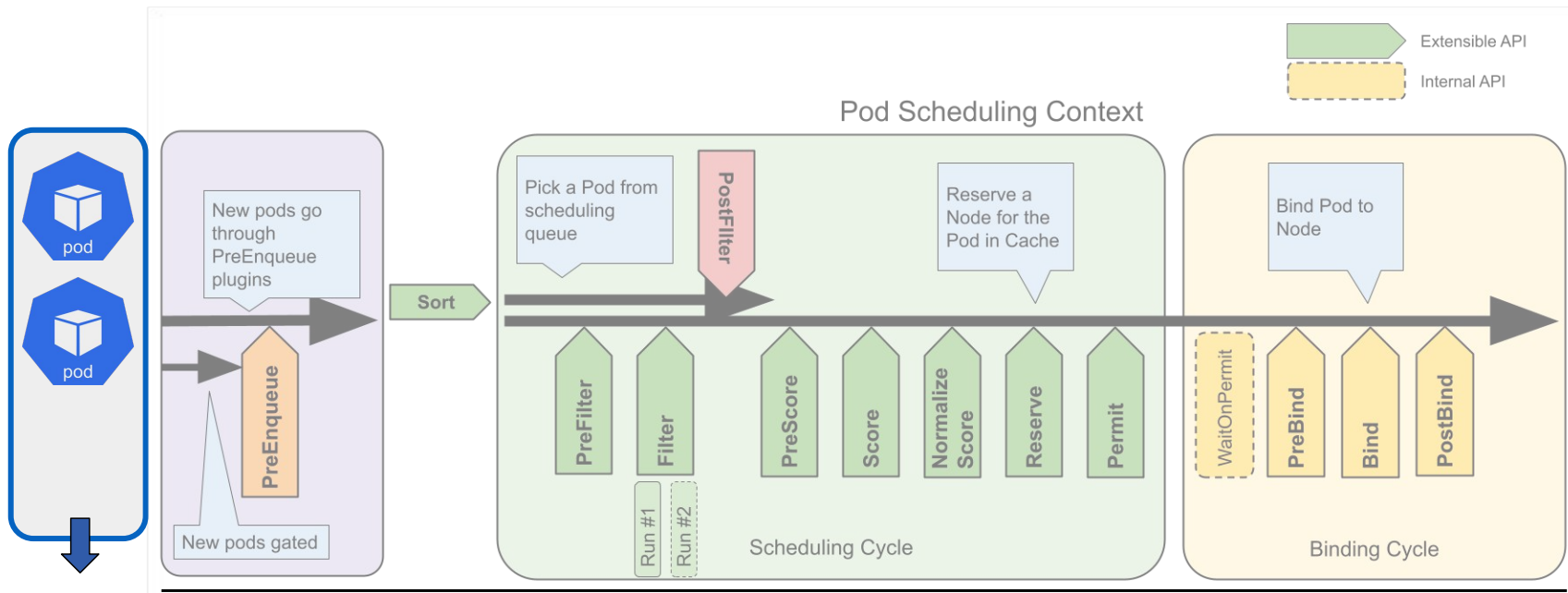
KubeCon



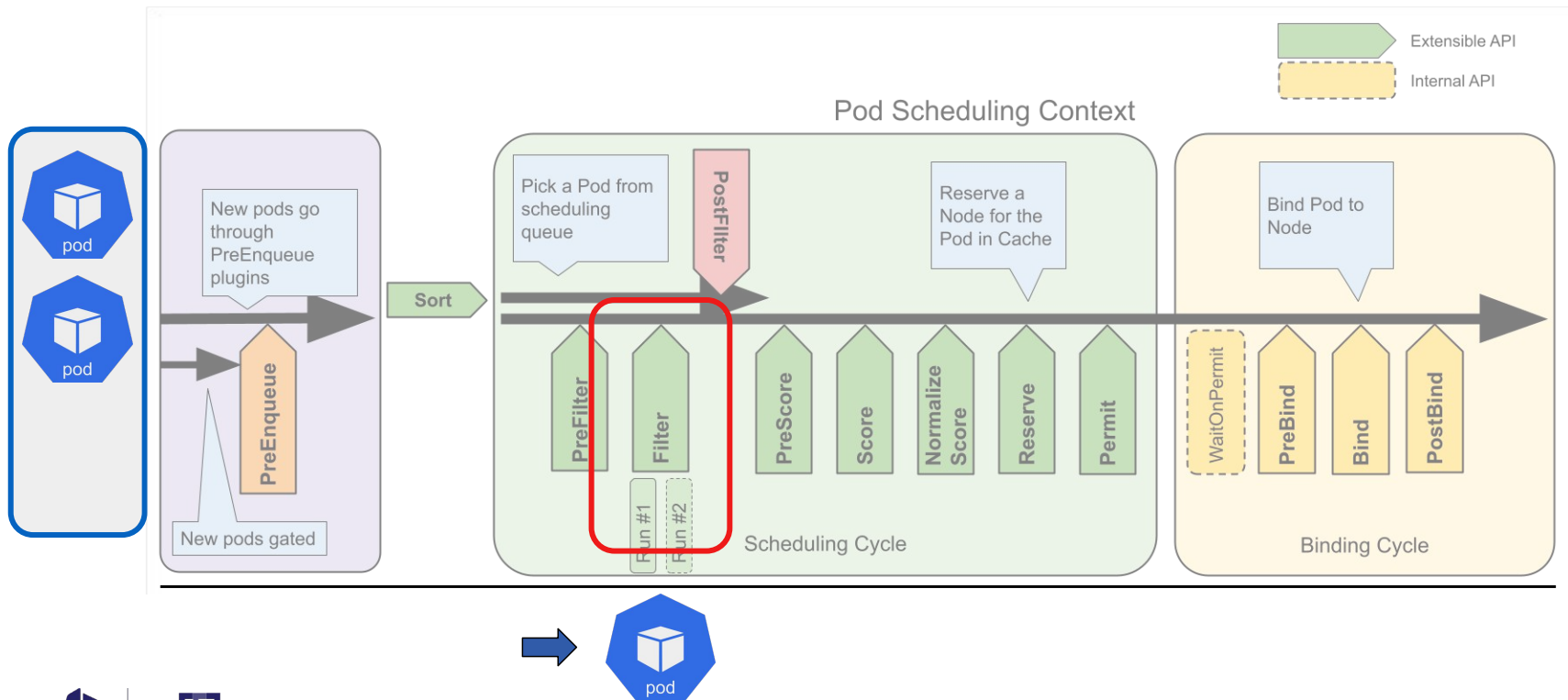
CloudNativeCon

Europe 2026

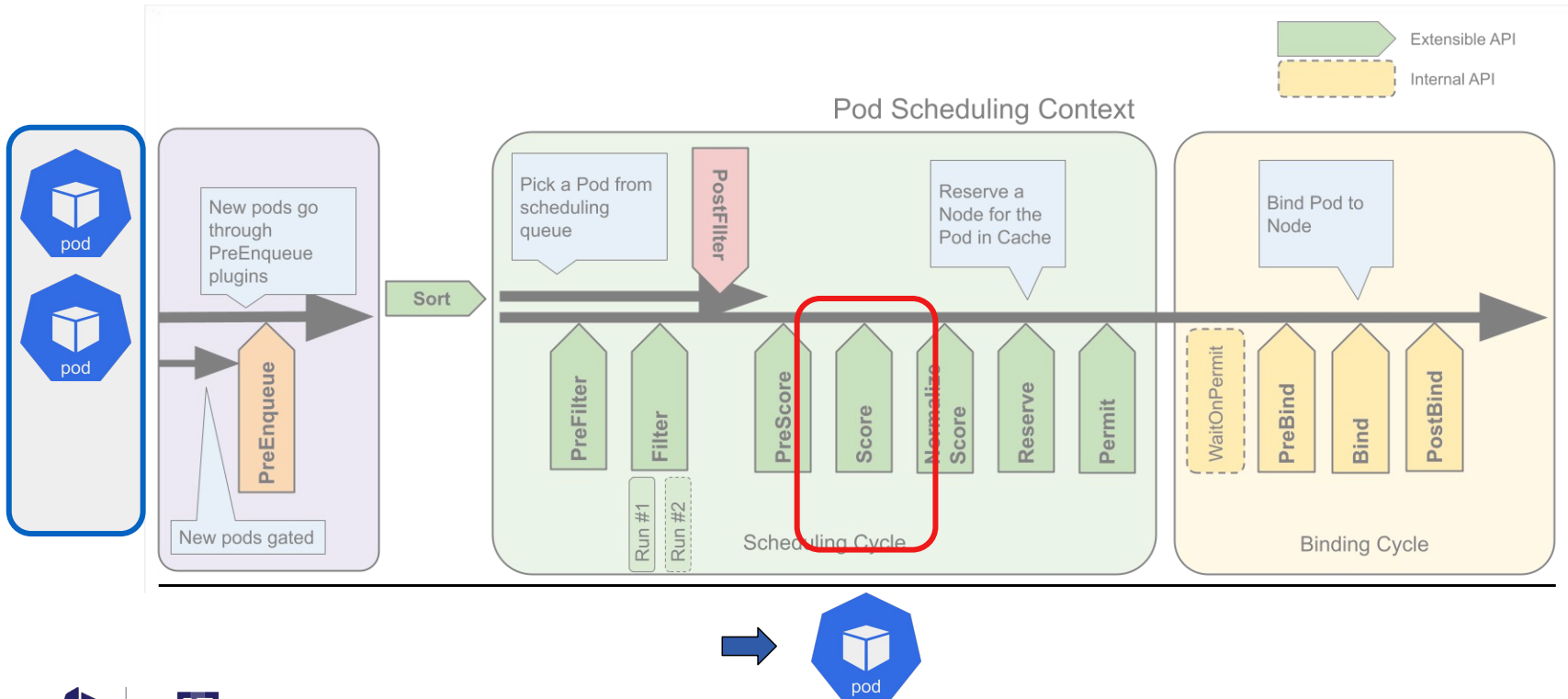
Gang Scheduling (v1.35)



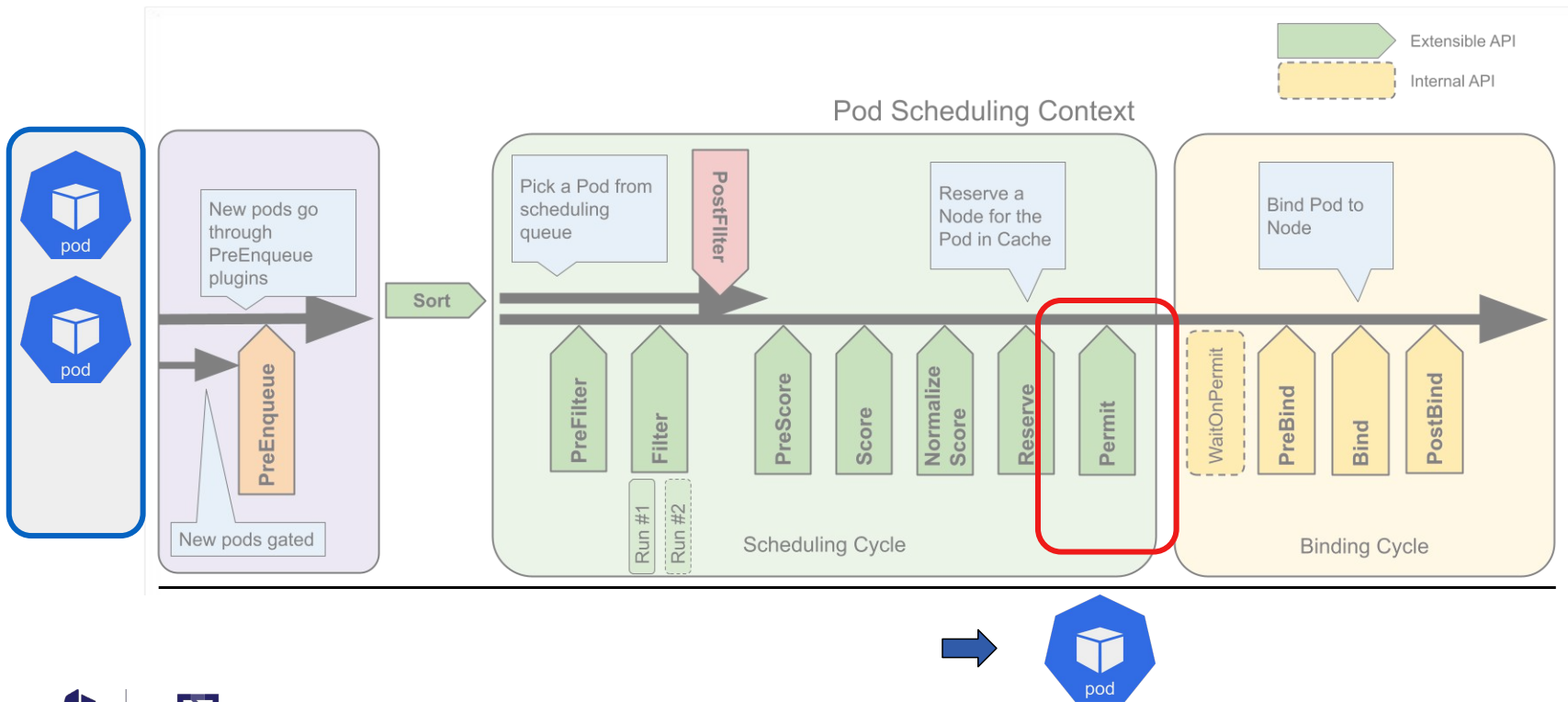
Gang Scheduling (v1.35)



Gang Scheduling (v1.35)



Gang Scheduling (v1.35)



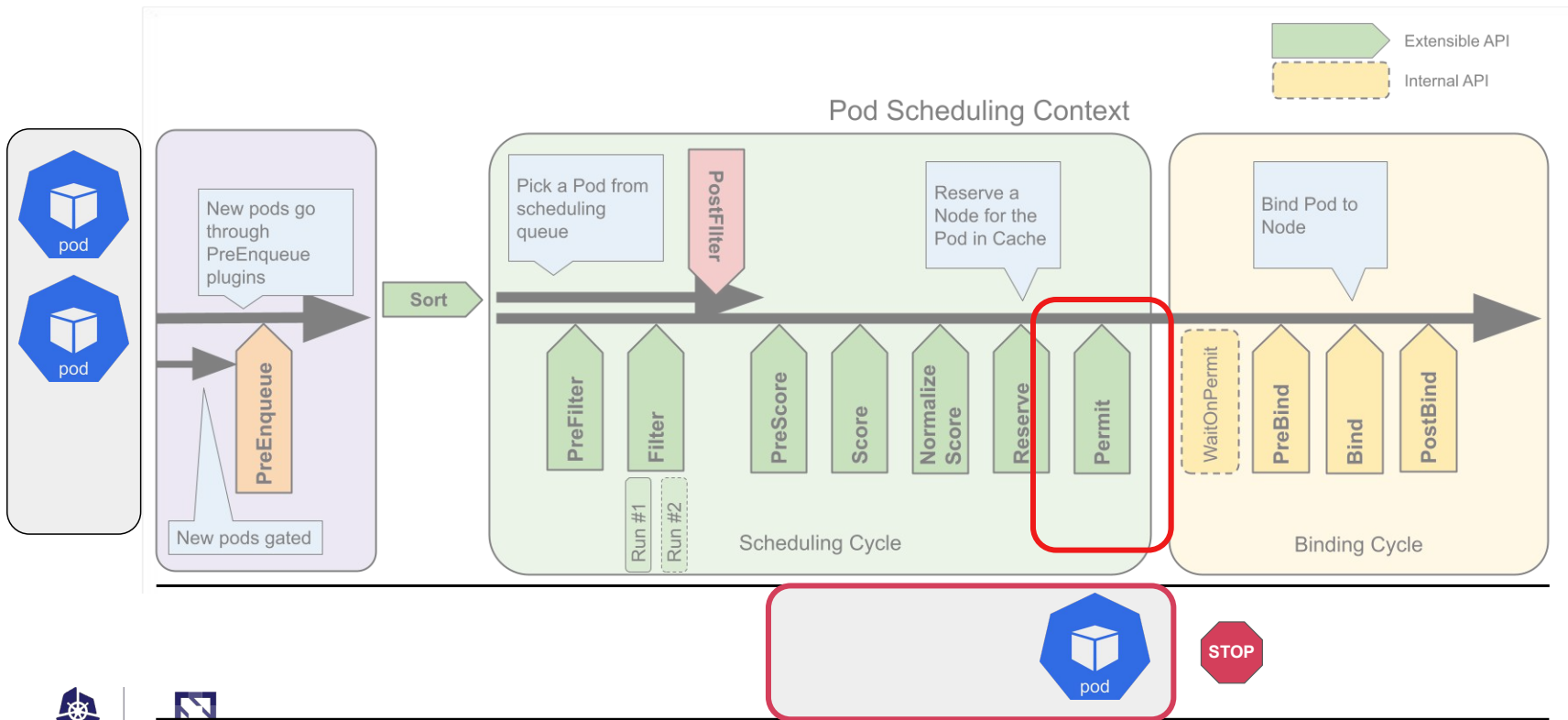
KubeCon



CloudNativeCon

Europe 2026

Gang Scheduling (v1.35)



KubeCon

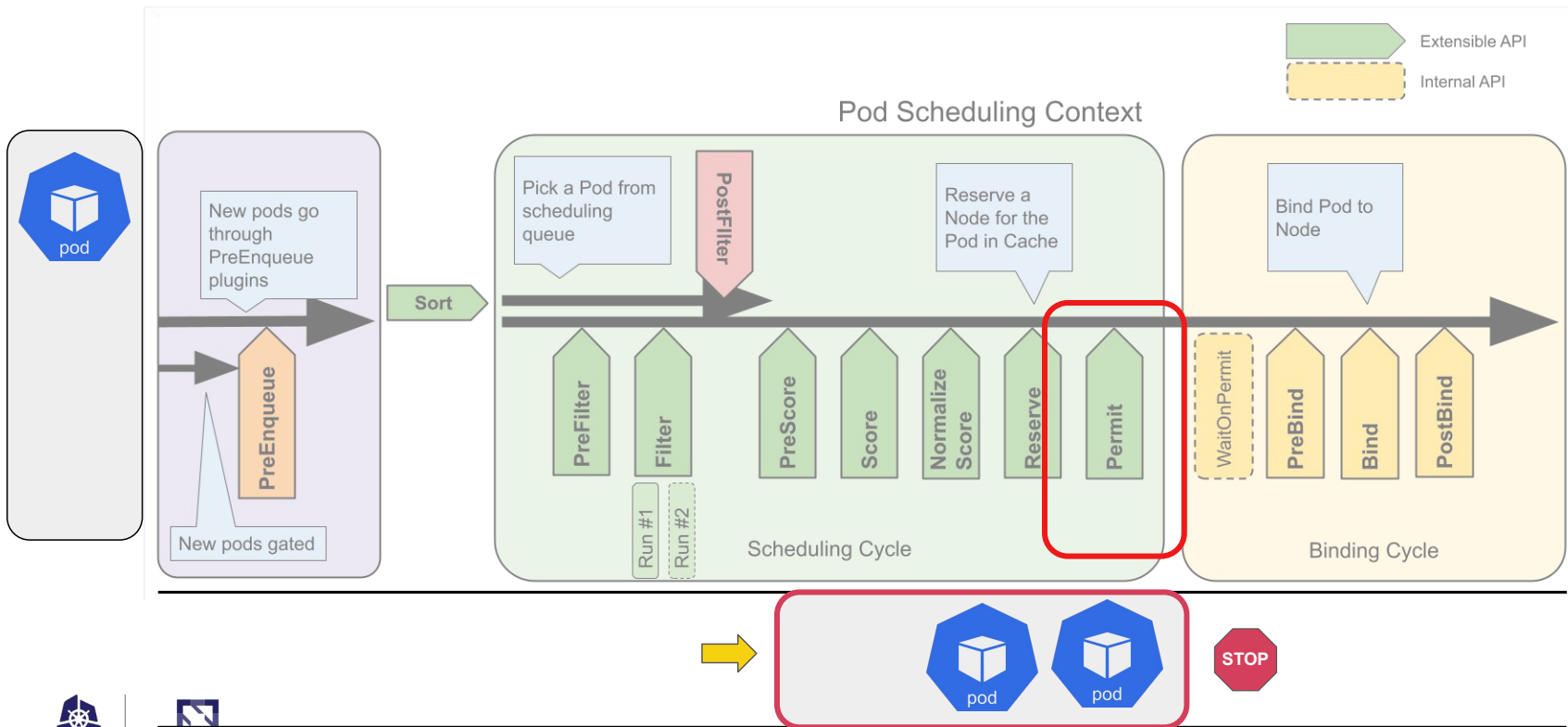


CloudNativeCon

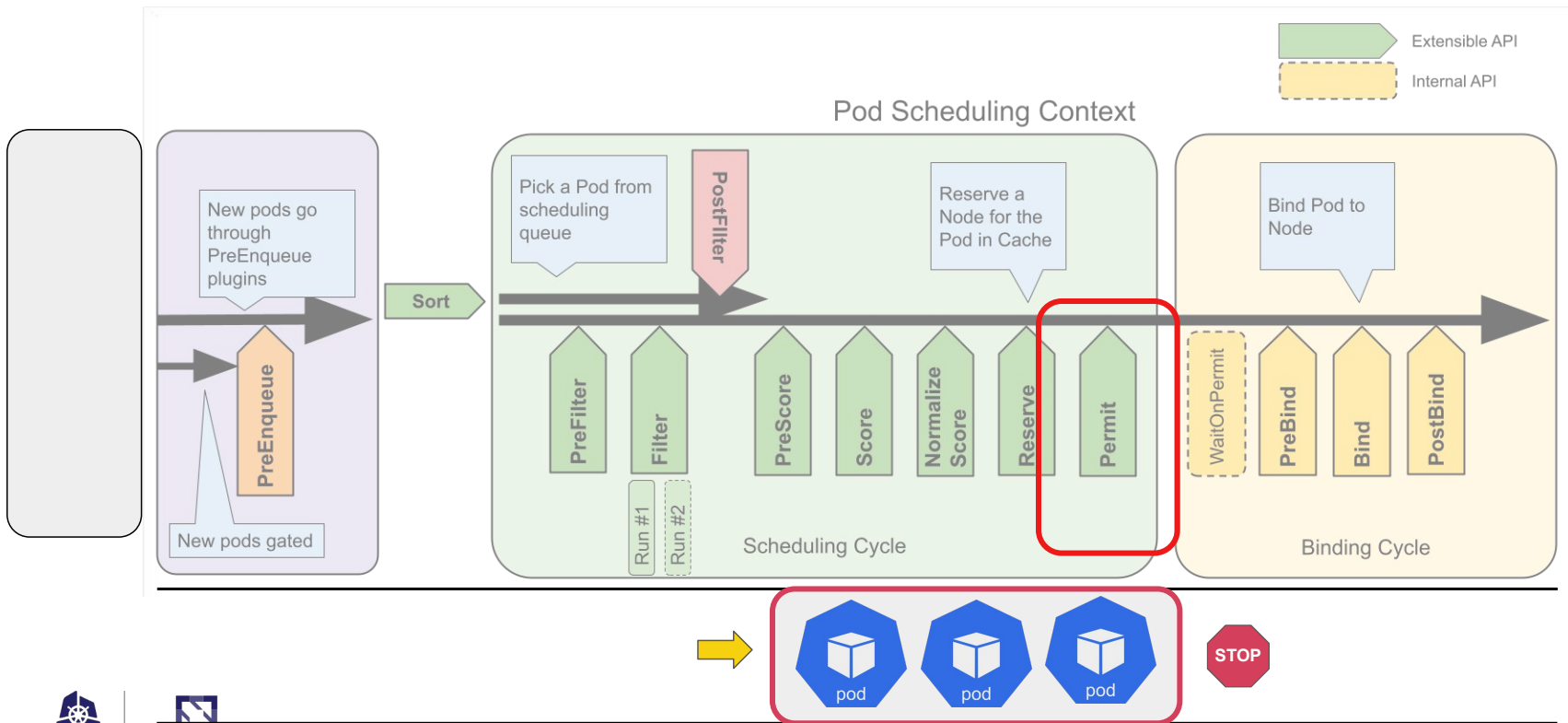
Europe 2026

(Scheduler reserves places for waiting pods)

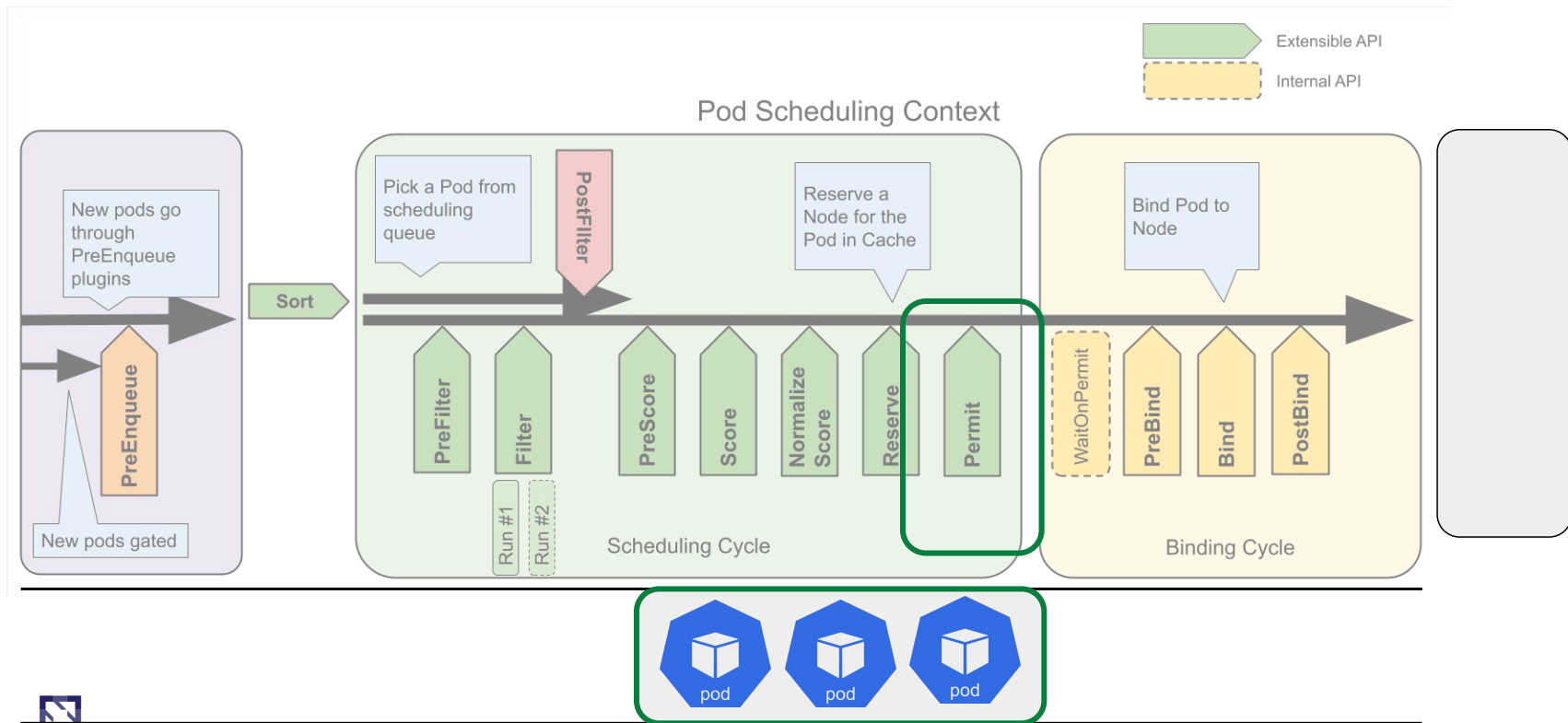
Gang Scheduling (v1.35)



Gang Scheduling (v1.35)

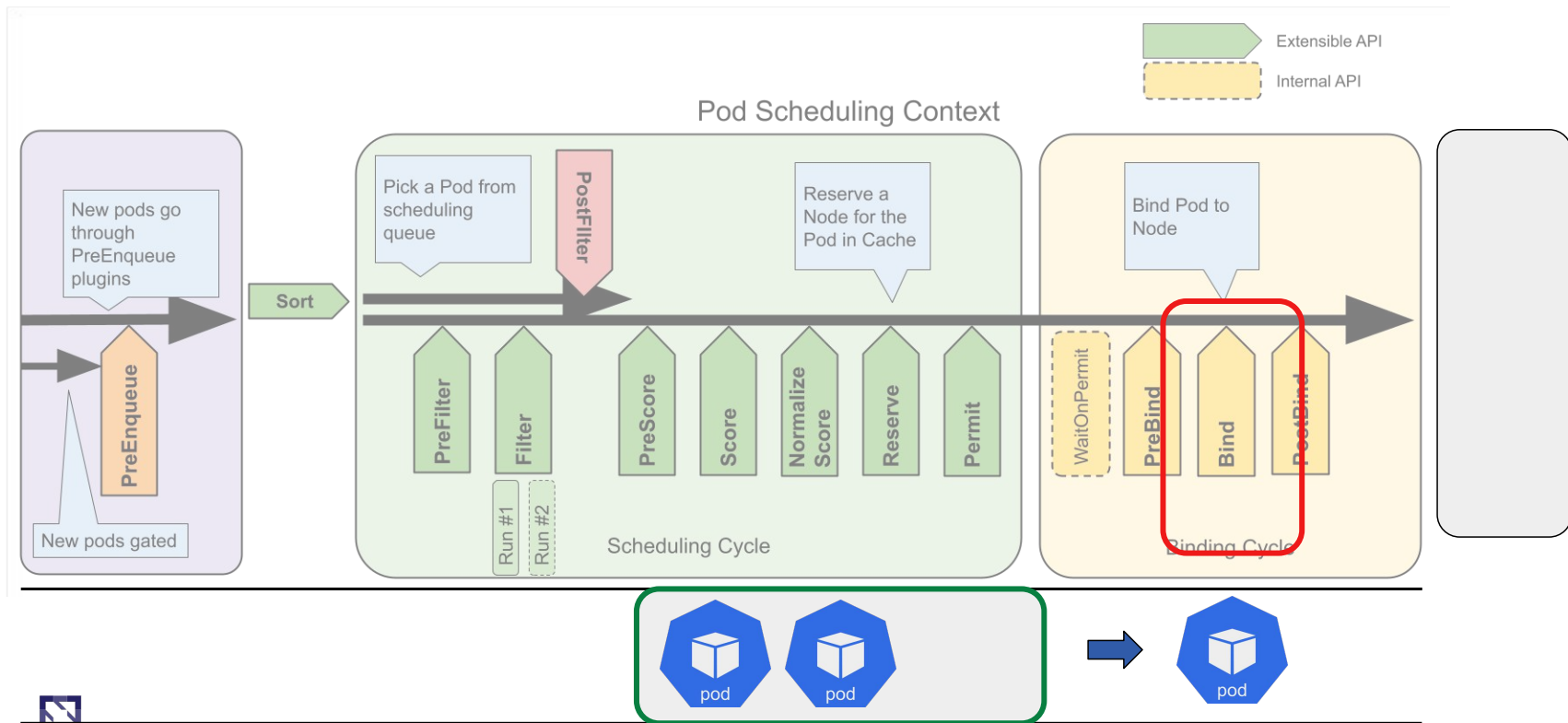


Gang Scheduling (v1.35)



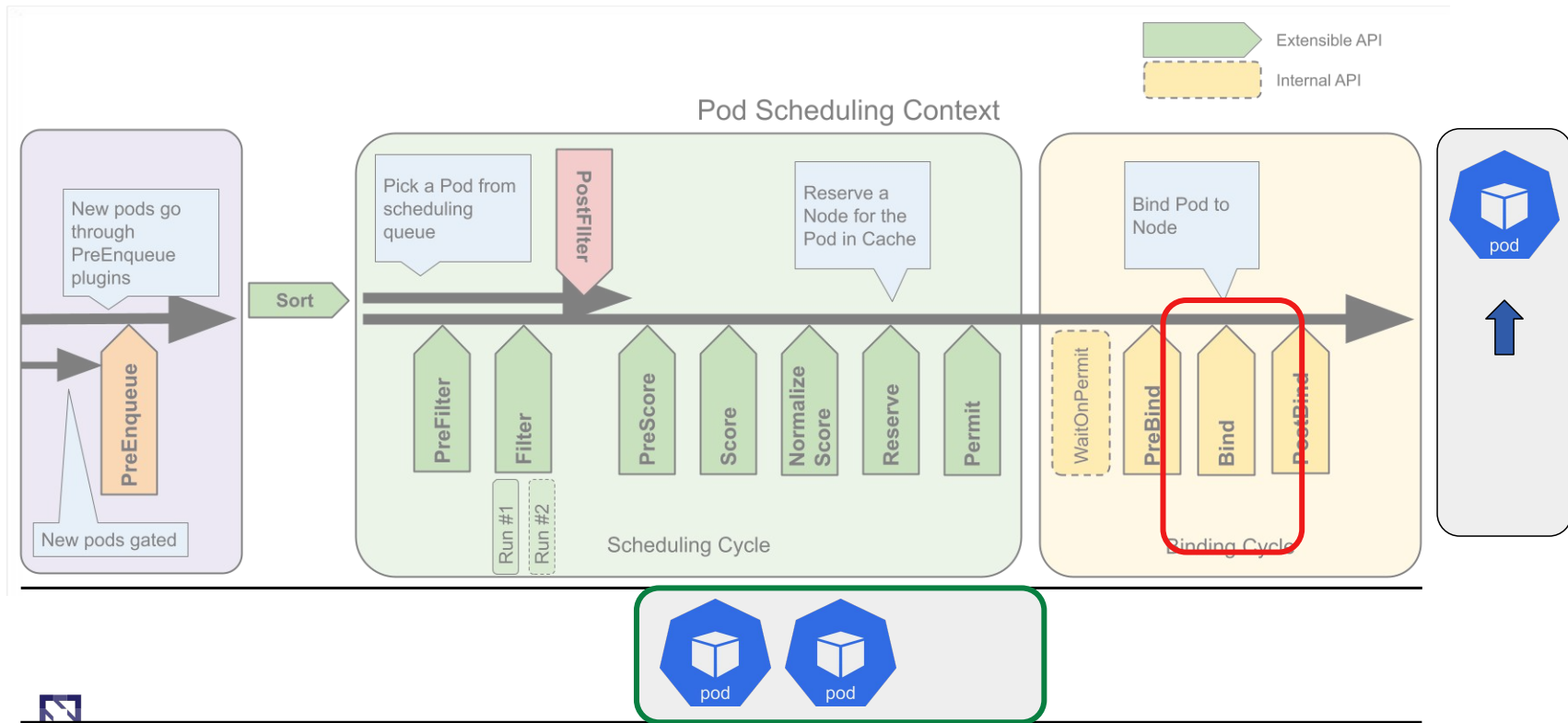
Admit all of them!

Gang Scheduling (v1.35)



Admit all of them!

Gang Scheduling (v1.35)



KubeCon

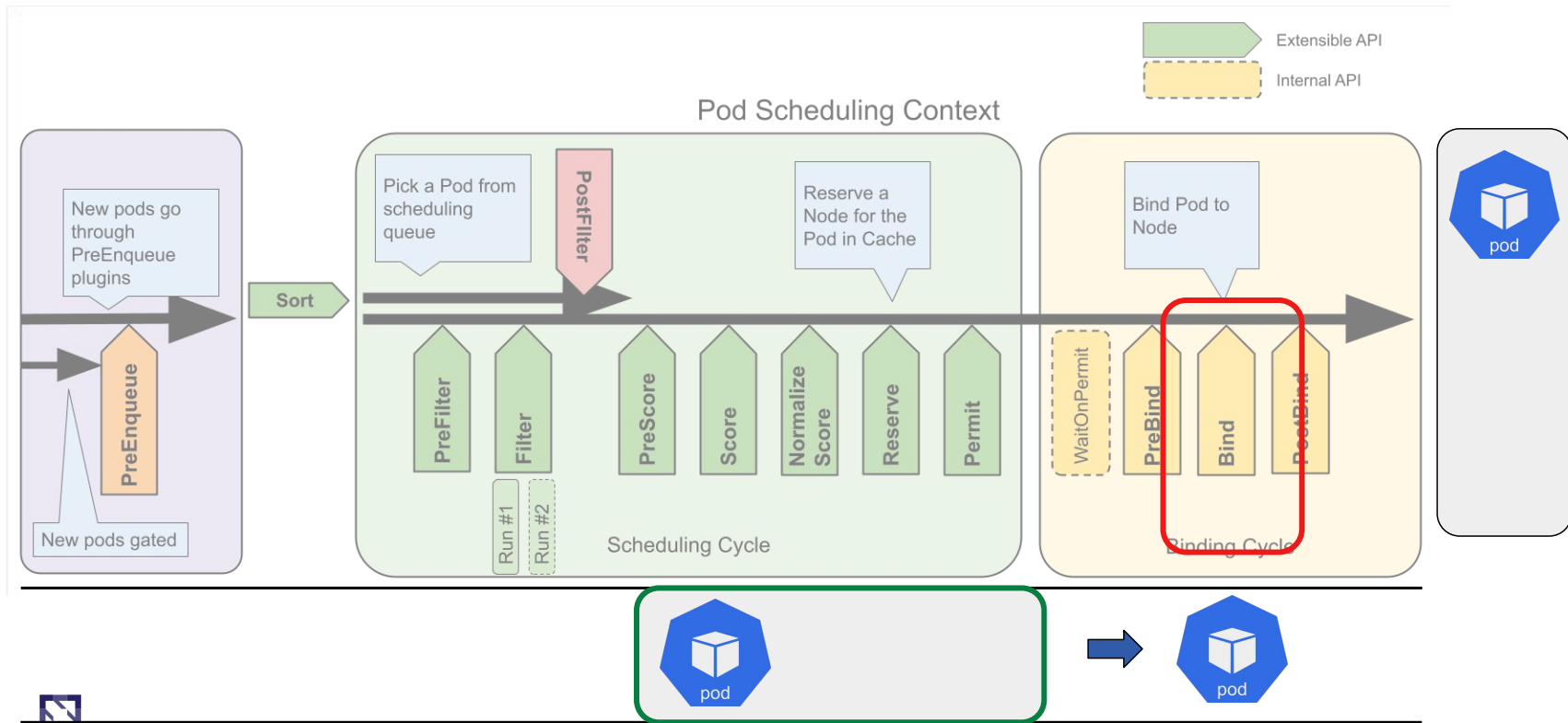


CloudNativeCon

Europe 2026

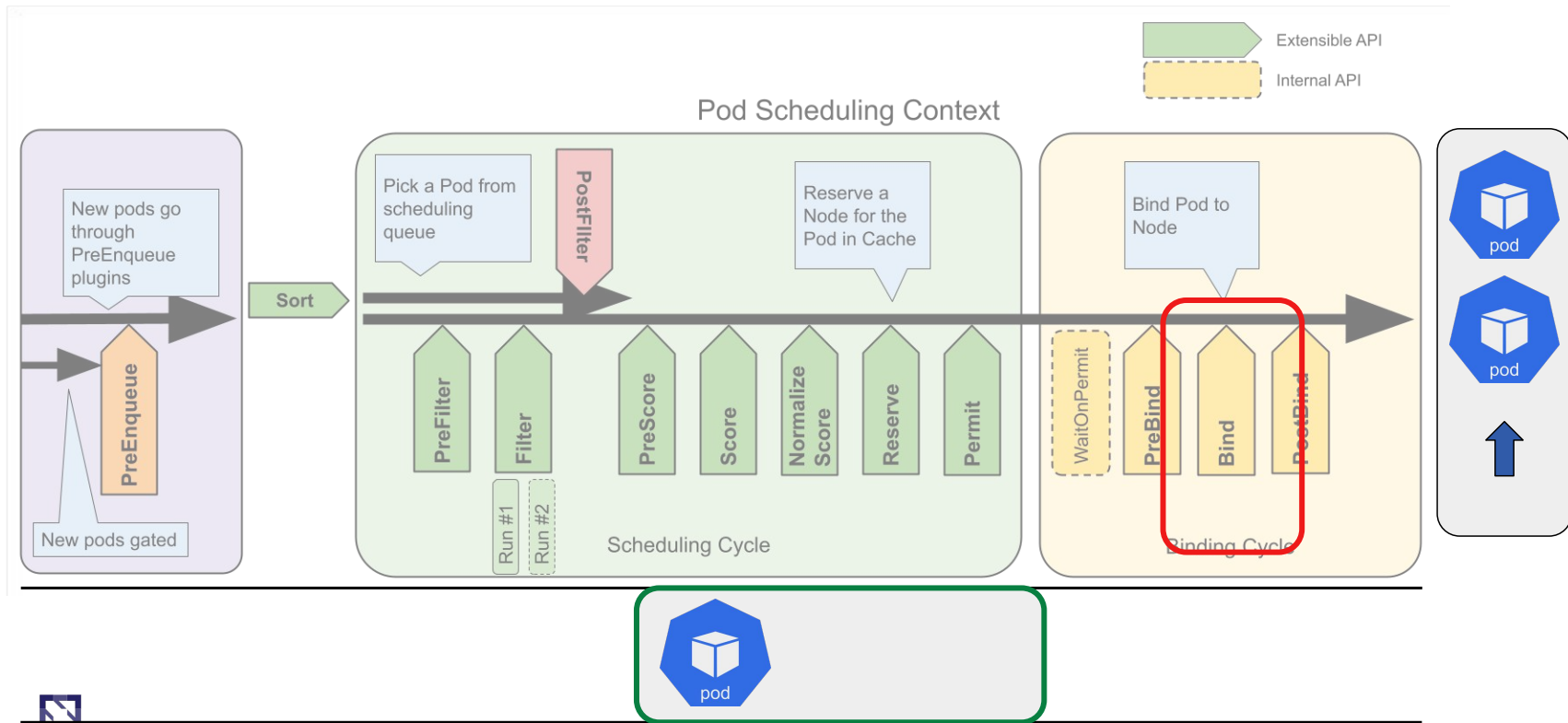
Admit all of them!

Gang Scheduling (v1.35)



Admit all of them!

Gang Scheduling (v1.35)



KubeCon

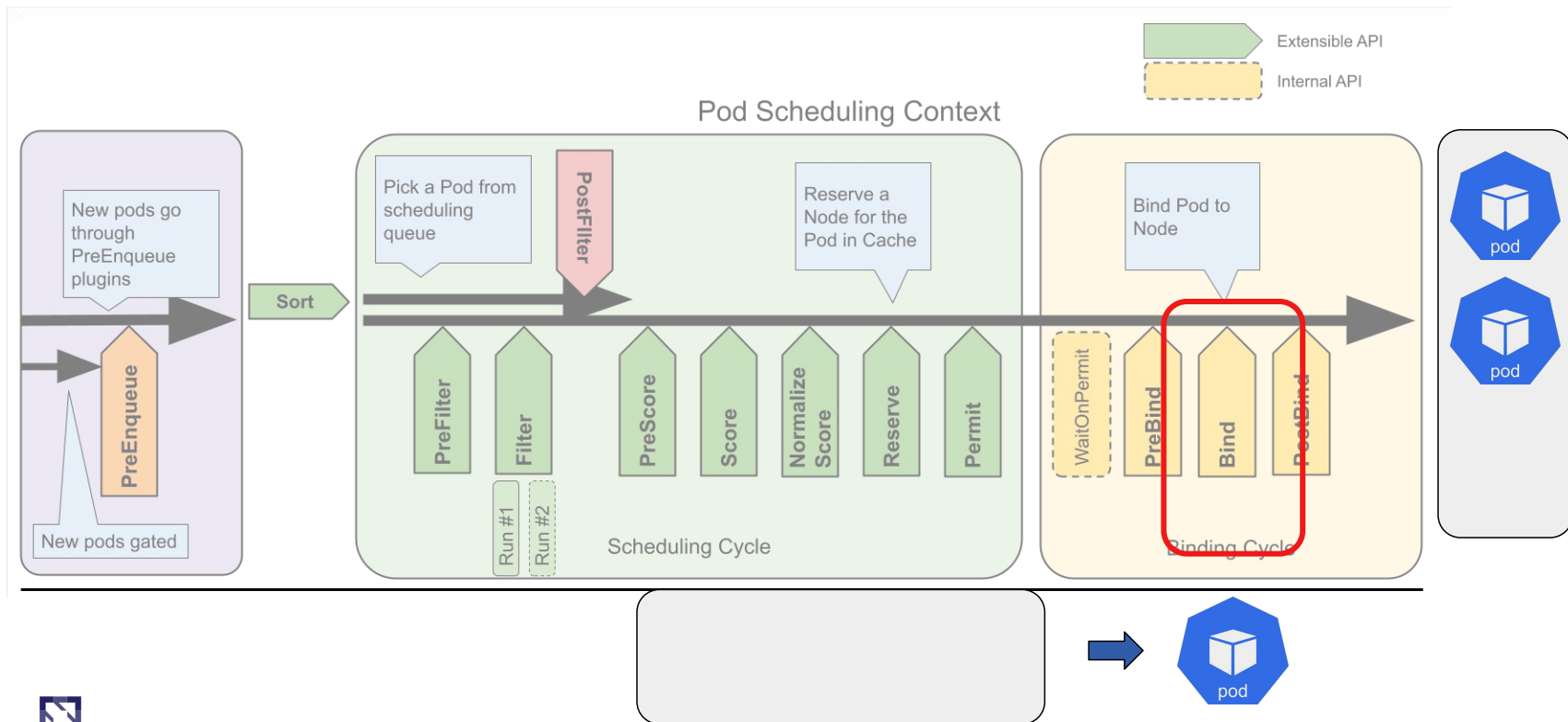


CloudNativeCon

Europe 2026

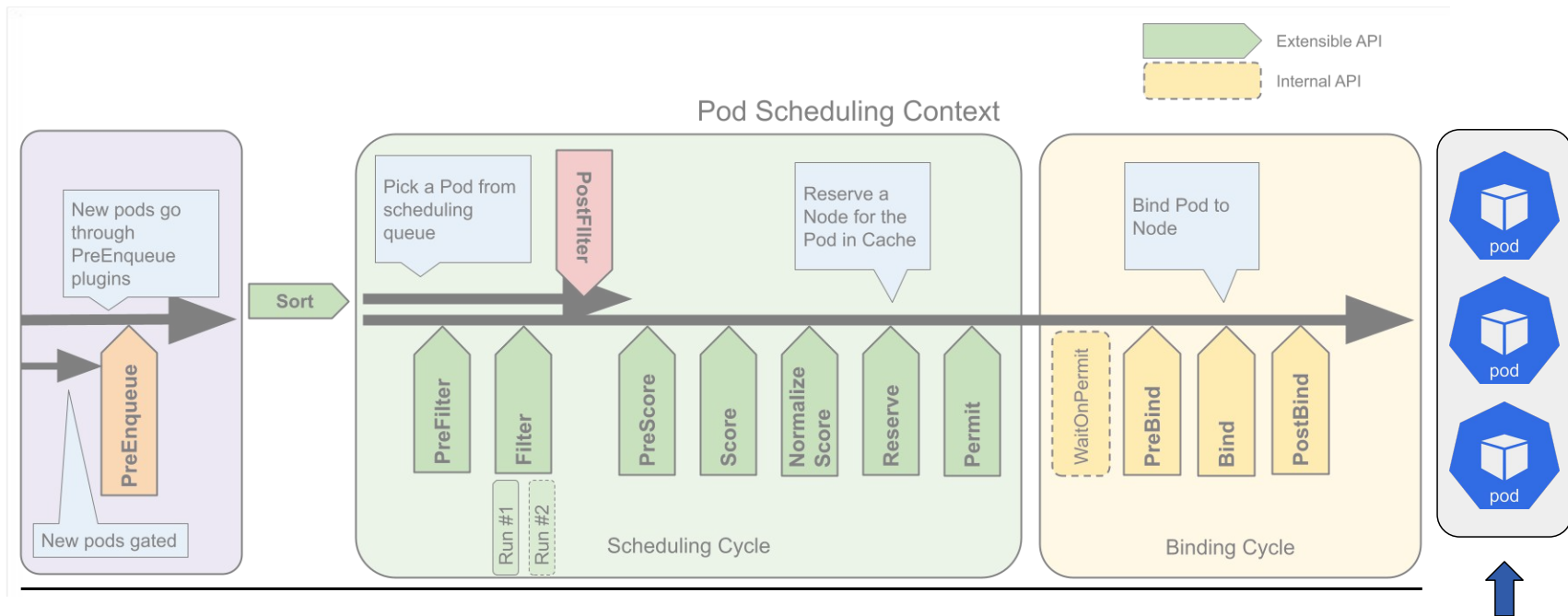
Admit all of them!

Gang Scheduling (v1.35)

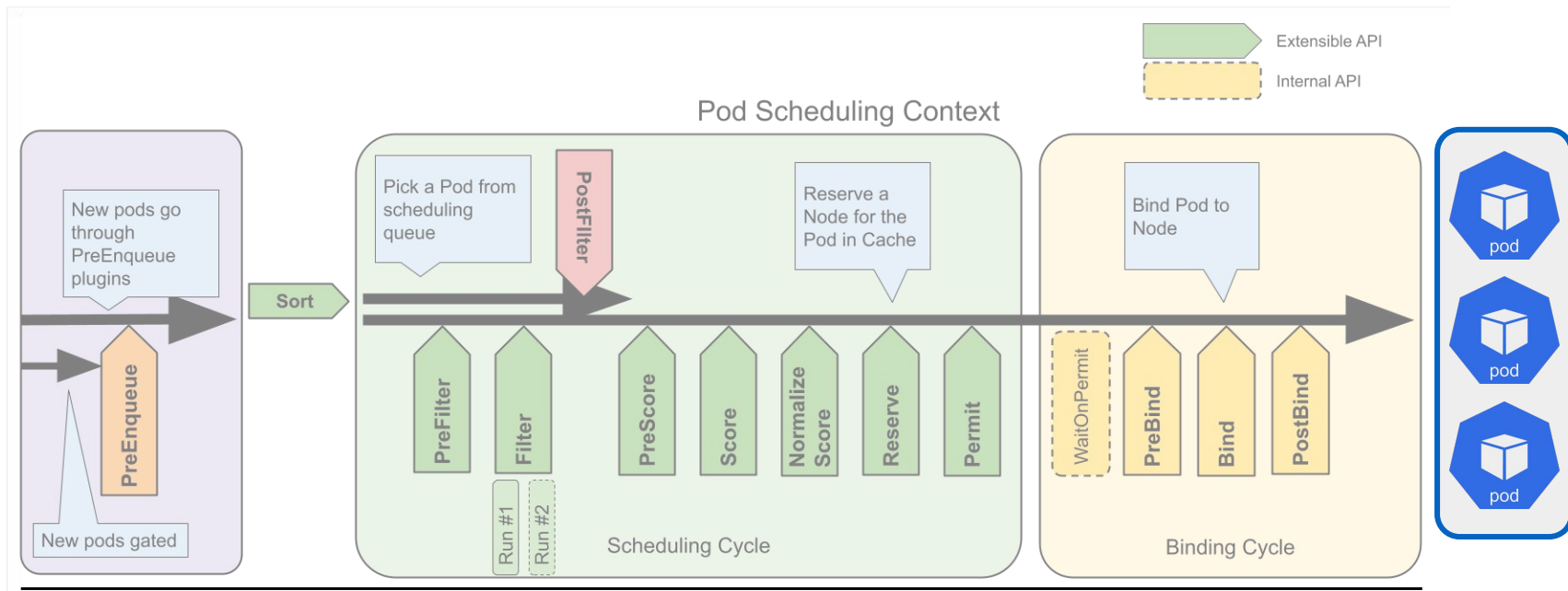


Admit all of them!

Gang Scheduling (v1.35)



Gang Scheduling (v1.35)



KubeCon



CloudNativeCon

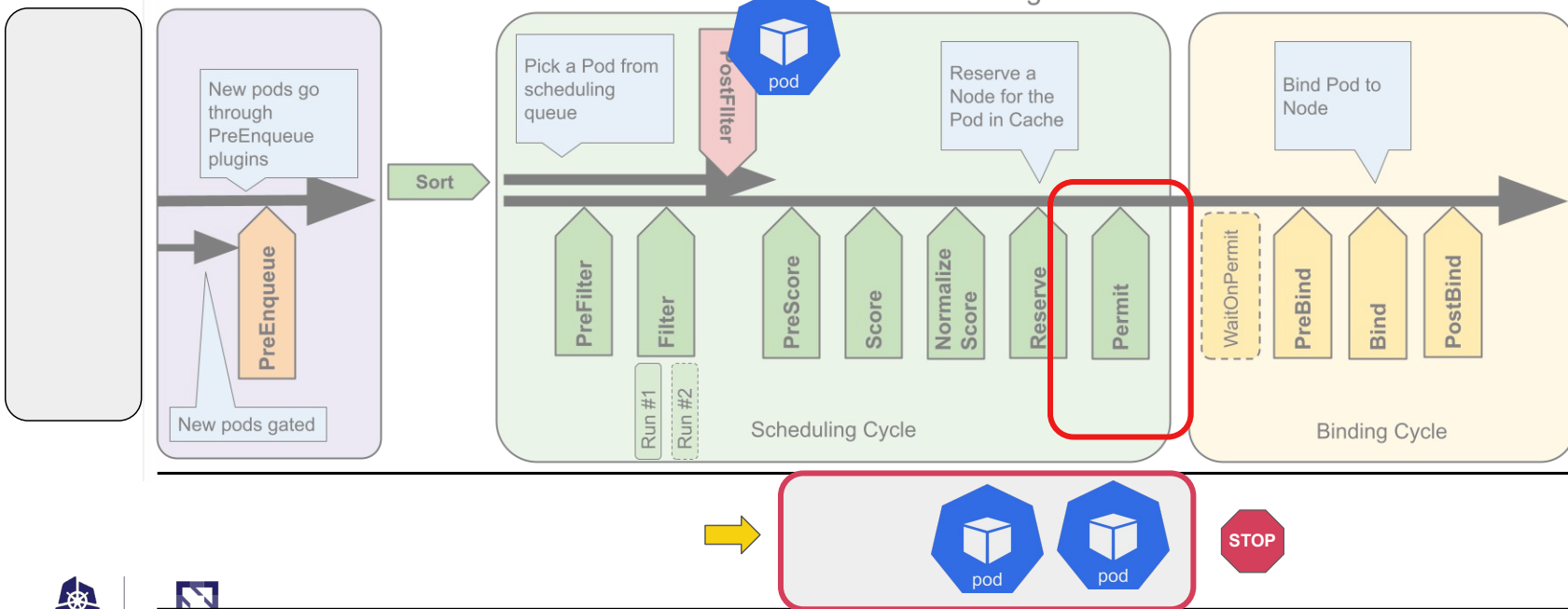
Europe 2026

Gang Scheduling (v1.35) - Problem

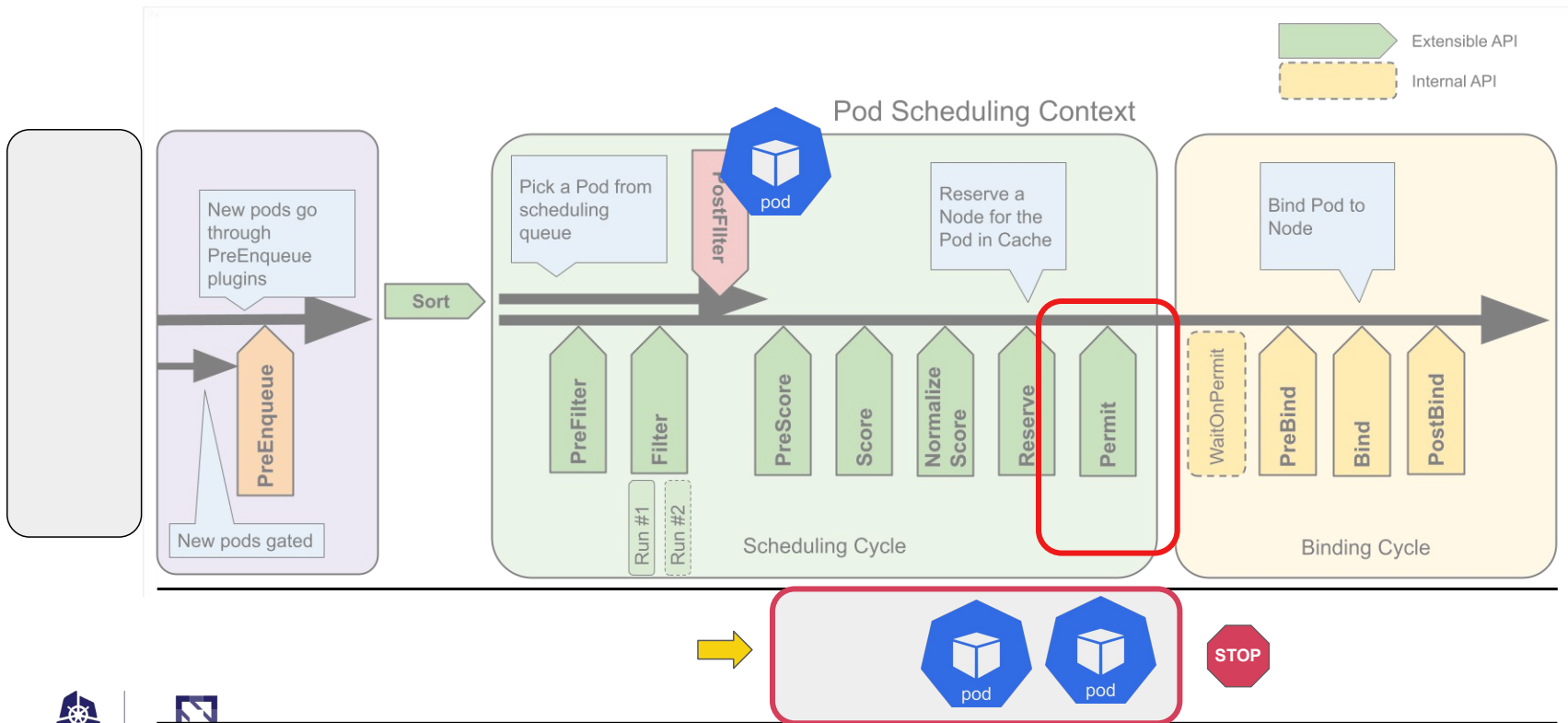
What if some pods are unschedulable..?

Extensible API
Internal API

Pod Scheduling Context

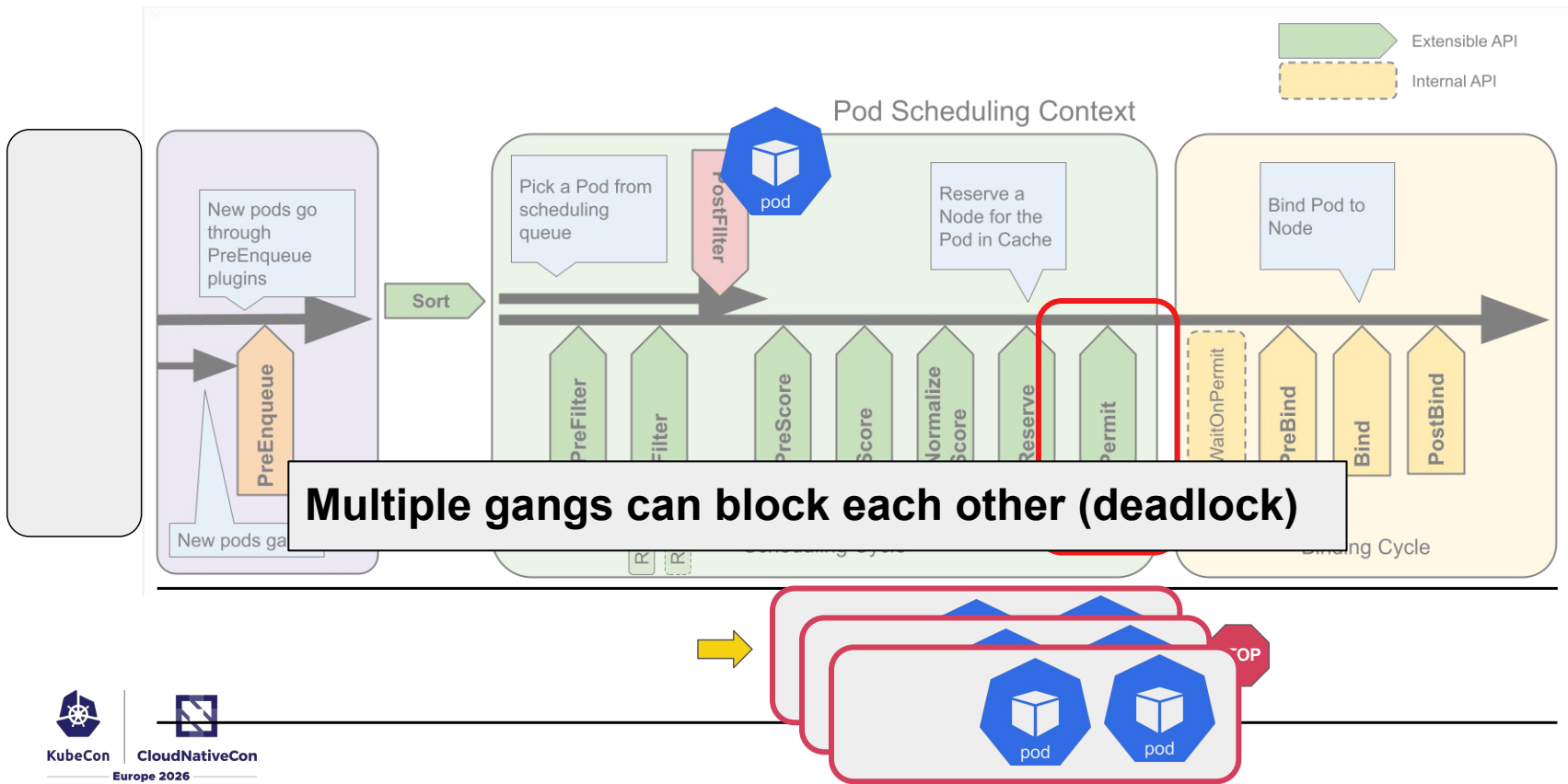


Gang Scheduling (v1.35) - Problem



Scheduler keeps their places

Gang Scheduling (v1.35) - Problem



Workload and PodGroup API (v1.36) - v1alpha2

Workload API (Policy)

- The "**static**" specification that defines the intent and scheduling policies without runtime
- Consists of PodGroupTemplates, renamed from PodGroups to clarify role as a blueprint

Workload and PodGroup API (v1.36) - v1alpha2

Workload API (Policy)

- The "**static**" specification that defines the intent and scheduling policies without runtime
- Consists of PodGroupTemplates, renamed from PodGroups to clarify role as a blueprint

PodGroup API (Runtime)

- The "**dynamic**" instance that represents runtime grouping of pods
- Provides a concrete entity for pods to reference directly
- Contains a status with information about the scheduling state
- Enables granular cleanup of ResourceClaims and other PodGroup-bound state

Workload and PodGroup API (v1.36) - v1alpha2

```
apiVersion: scheduling.k8s.io/v1alpha2
kind: Workload
metadata:
  name: jobset
spec:
  podGroupTemplates:
    - name: "job-1"
      schedulingPolicy:
        gang:
          minCount: 100
```

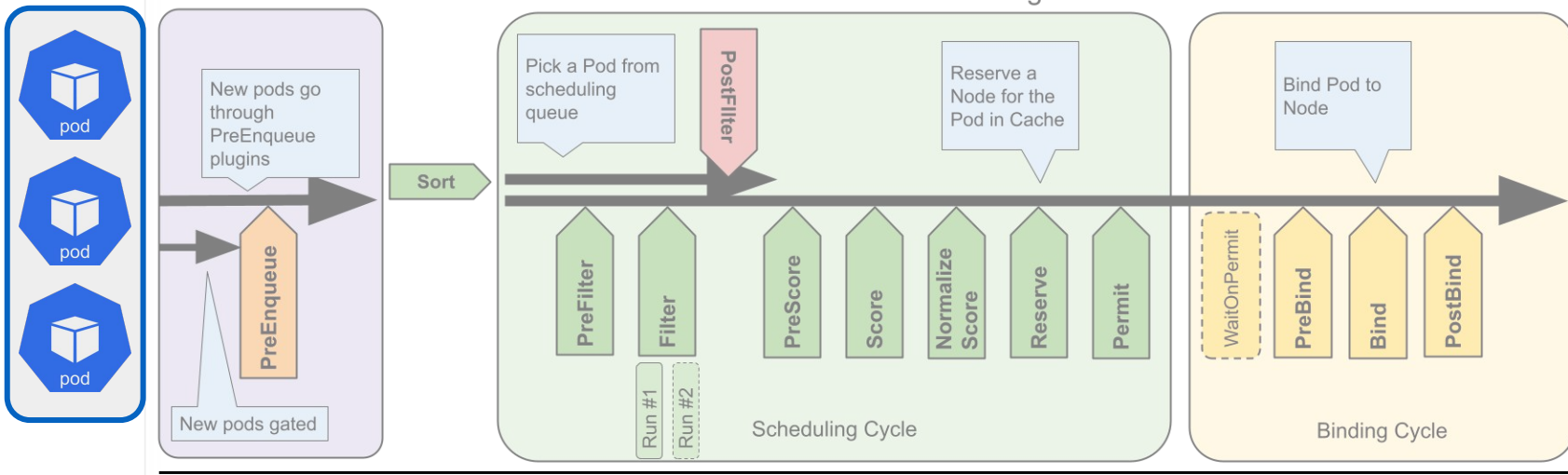
```
apiVersion: v1
kind: Pod
metadata:
  name: jobset-job-1-abc123
spec:
  ...
  schedulingGroup:
    podGroupName: job-instance-worker-0
  ...
```

```
apiVersion: scheduling.k8s.io/v1alpha2
kind: PodGroup
metadata:
  name: job-instance-worker-0
spec:
  podGroupTemplateRef:
    workloadName: job-policy
    podGroupTemplateName: worker-template
  # schedulingPolicy is copied from template on PodGroup creation.
  schedulingPolicy:
    gang:
      minCount: 100
```

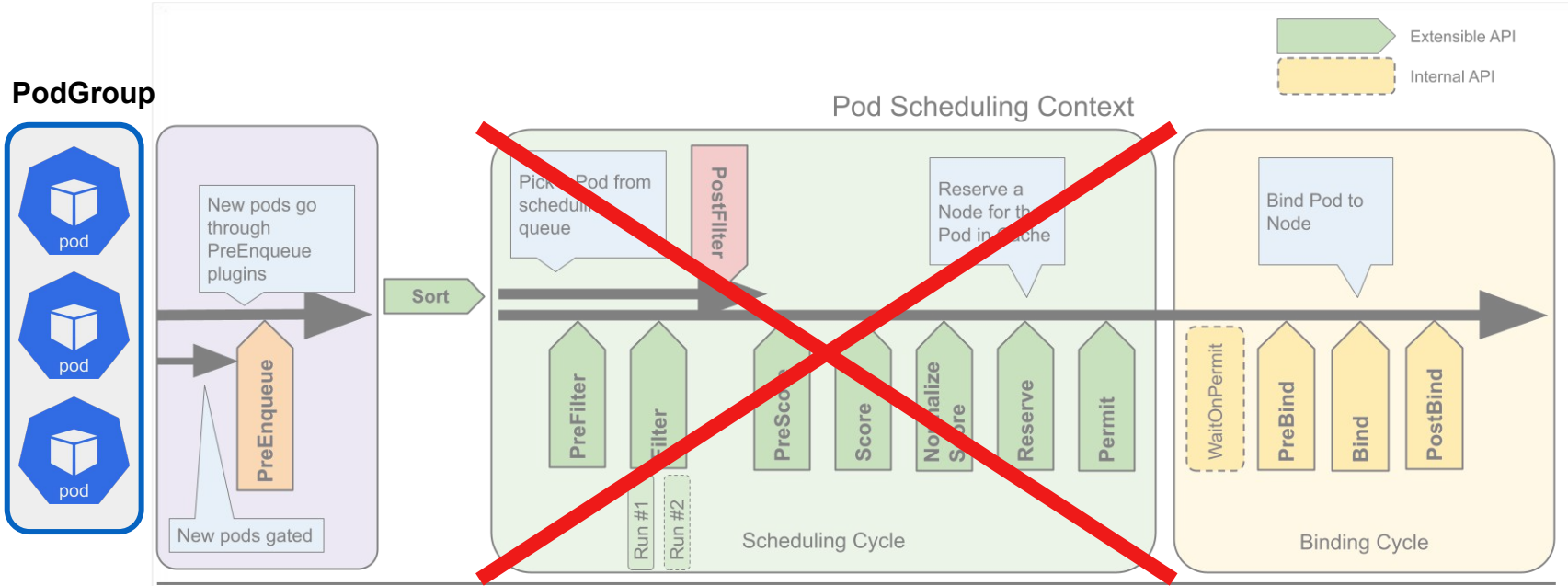


PodGroup Scheduling Cycle

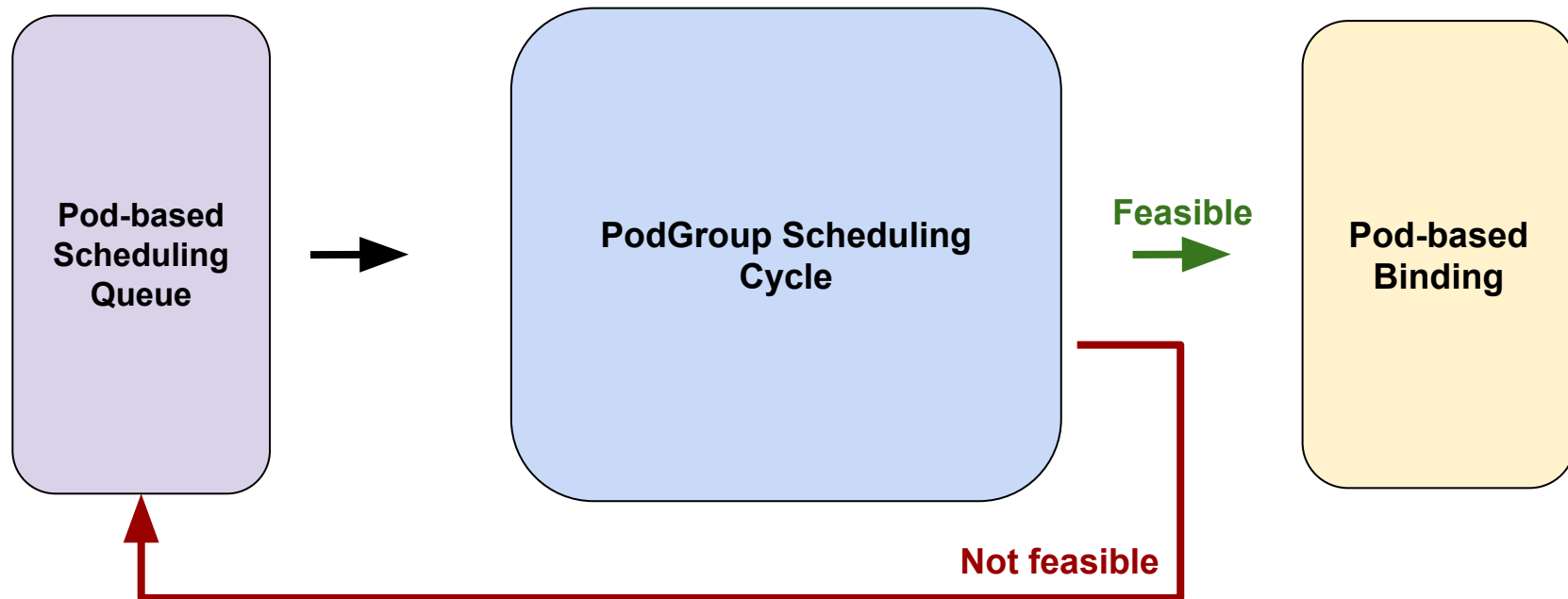
PodGroup



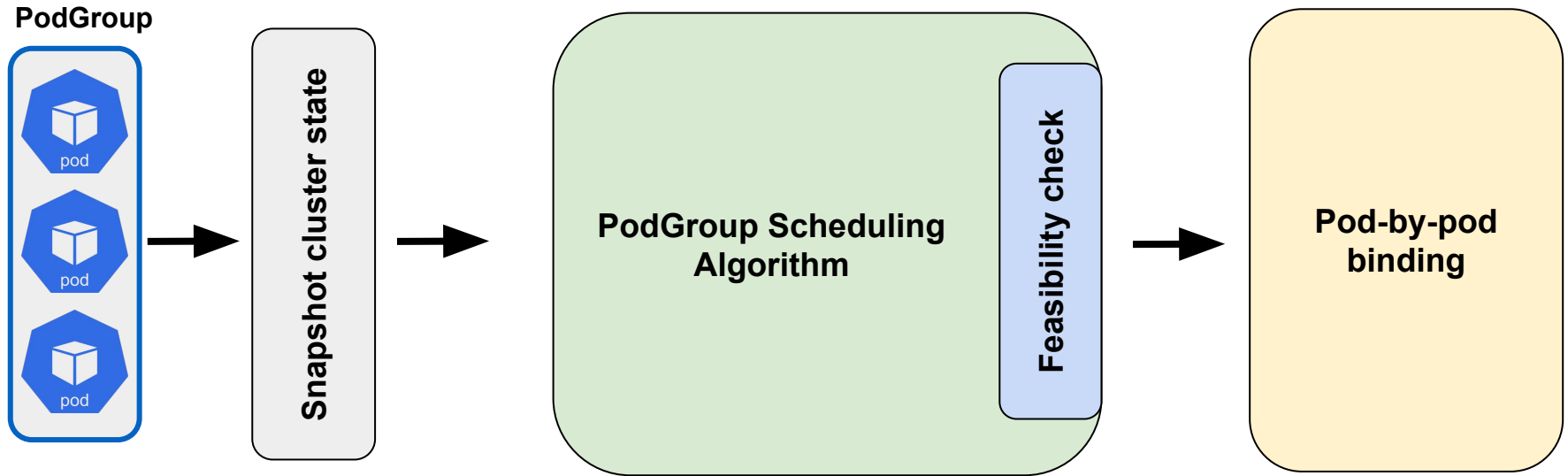
PodGroup Scheduling Cycle



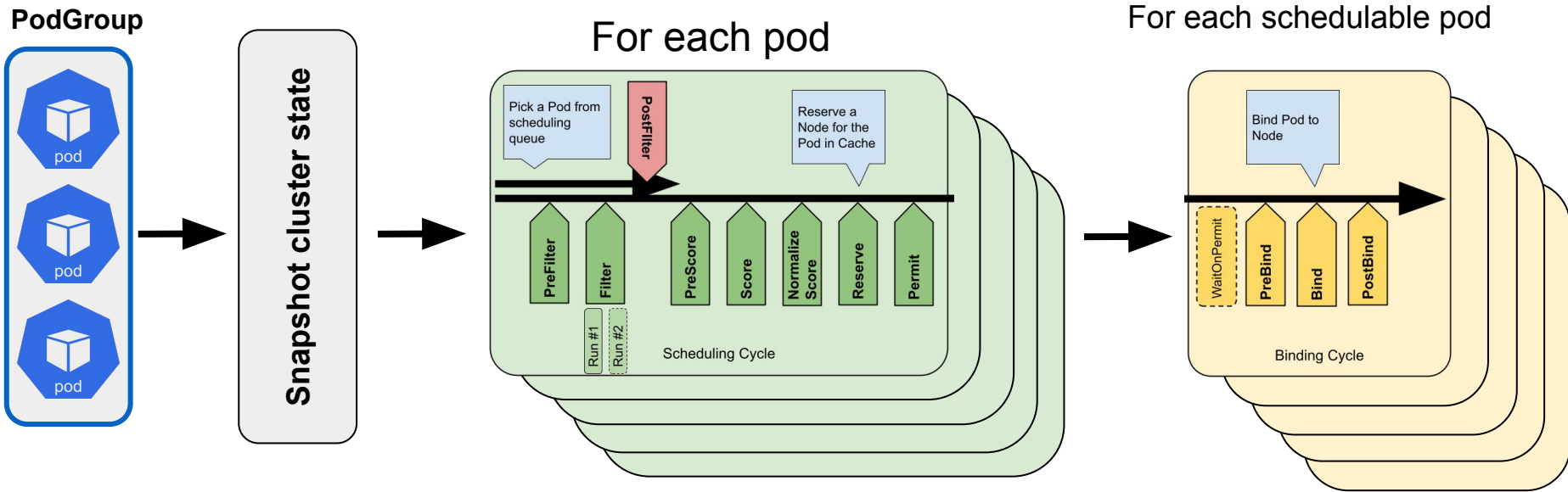
PodGroup Scheduling



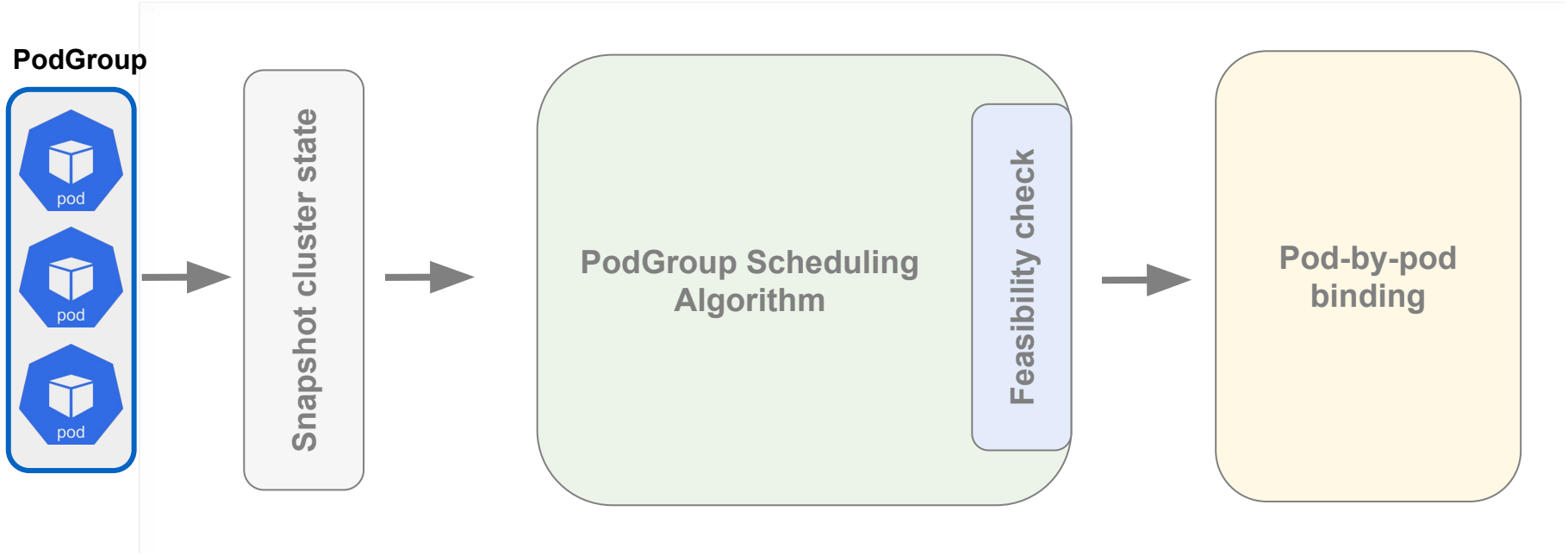
PodGroup Scheduling Cycle



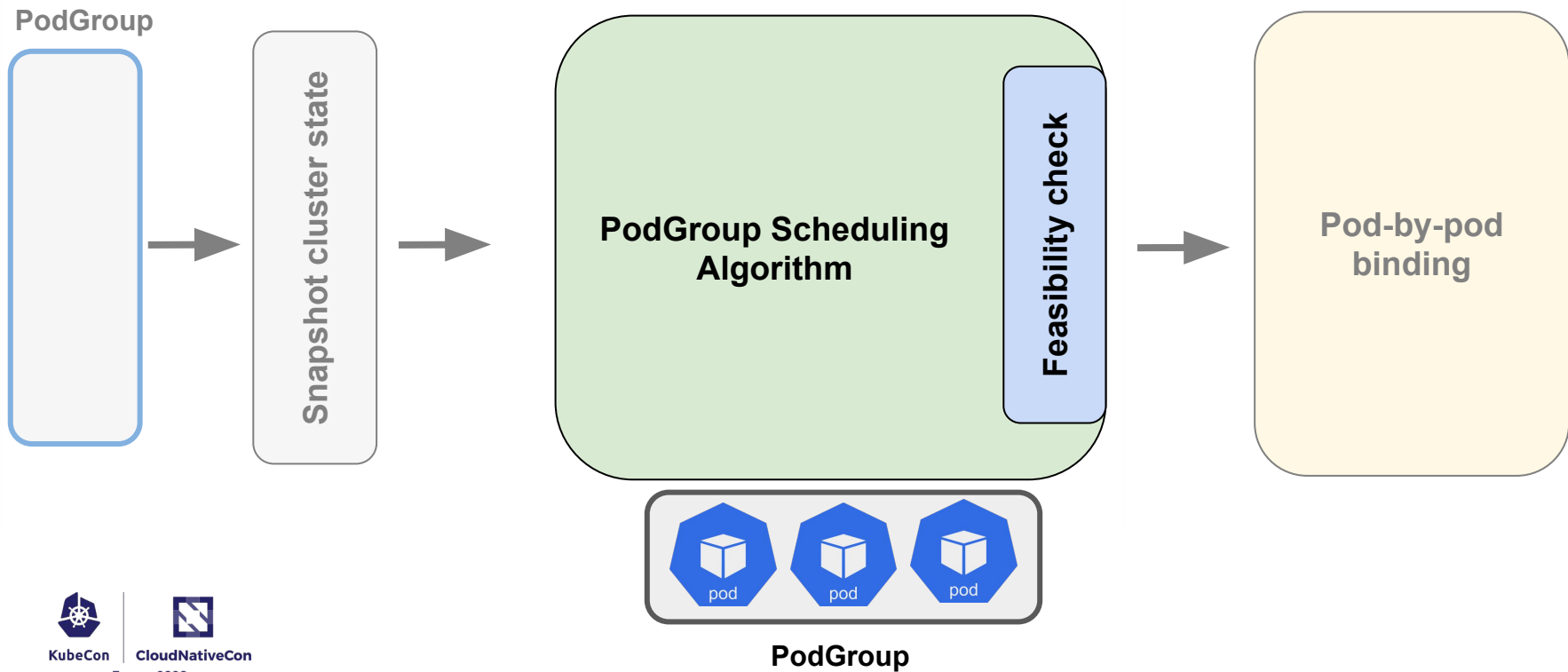
PodGroup Scheduling Cycle



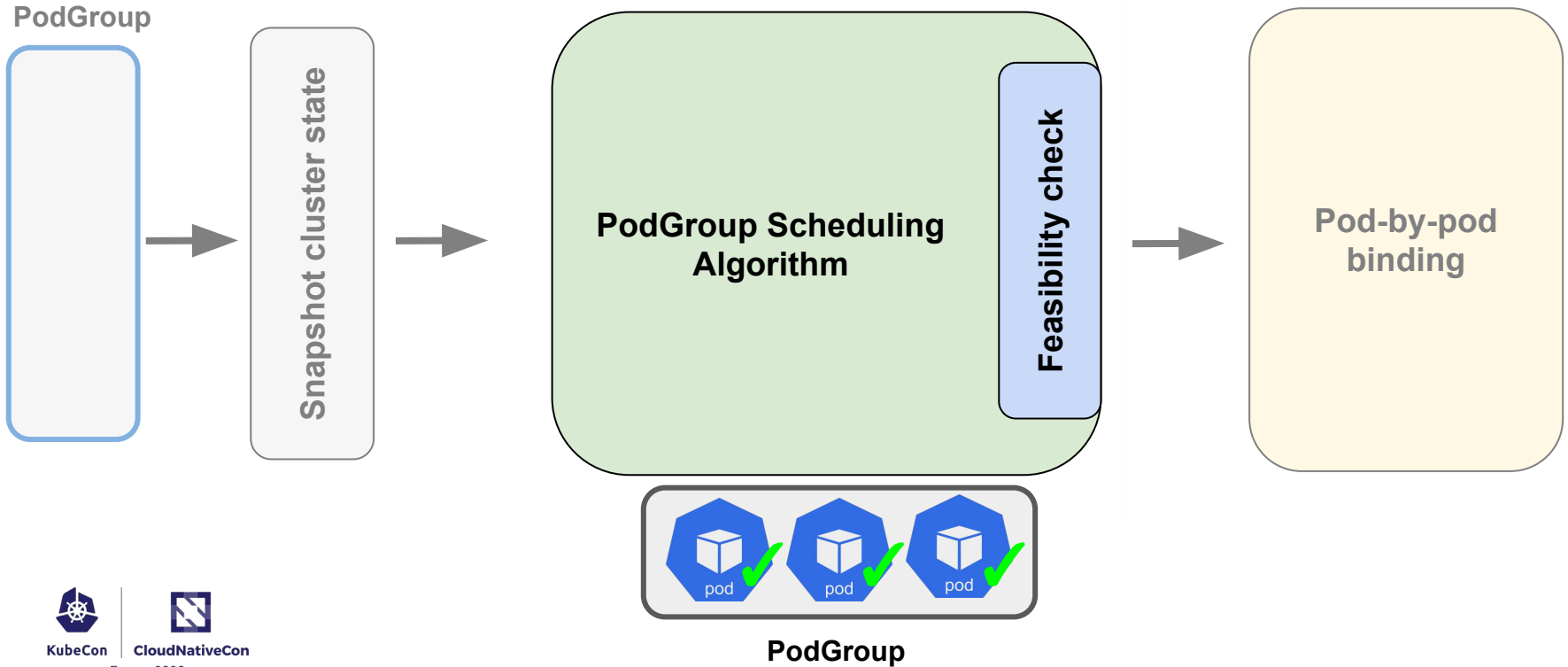
PodGroup Scheduling Cycle



PodGroup Scheduling Cycle



PodGroup Scheduling Cycle



KubeCon

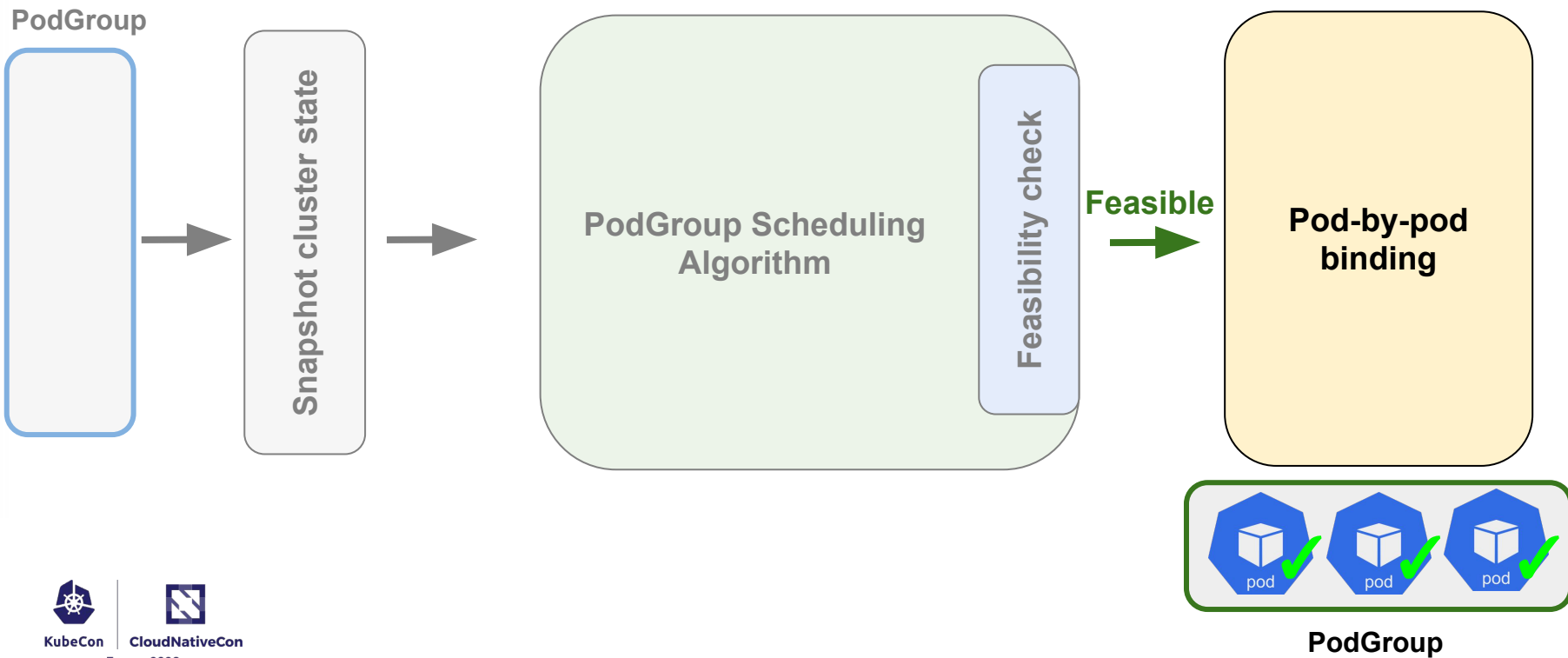


CloudNativeCon

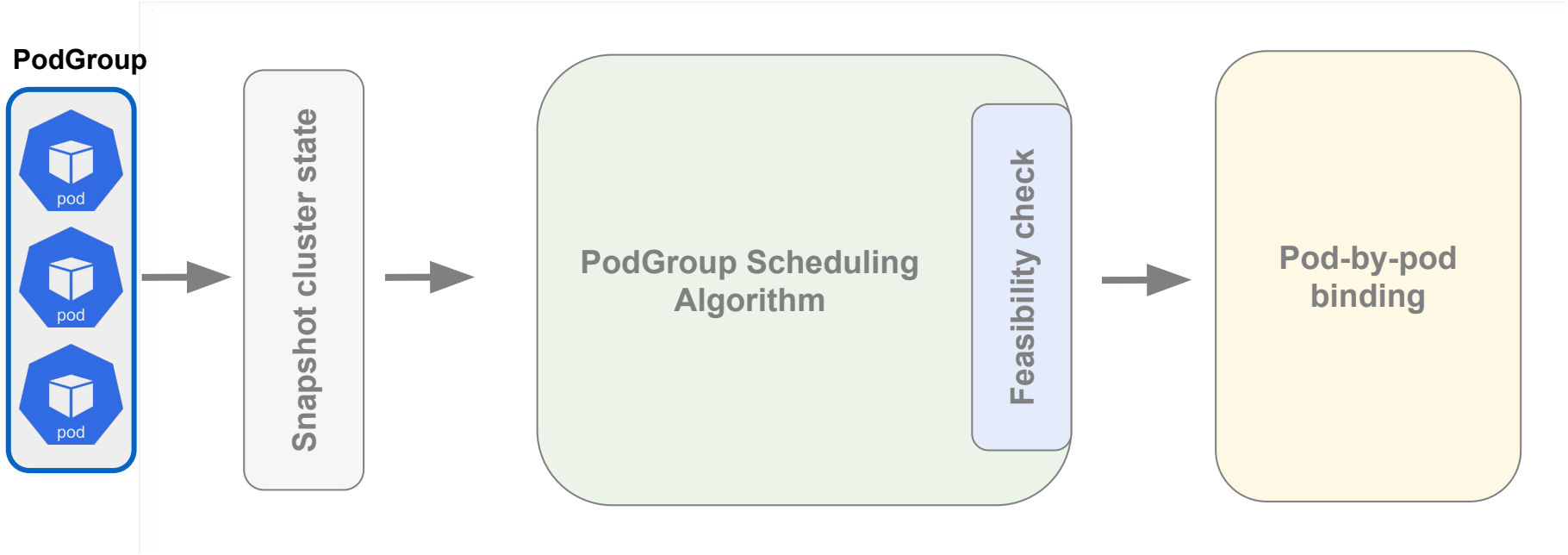
Europe 2026

PodGroup

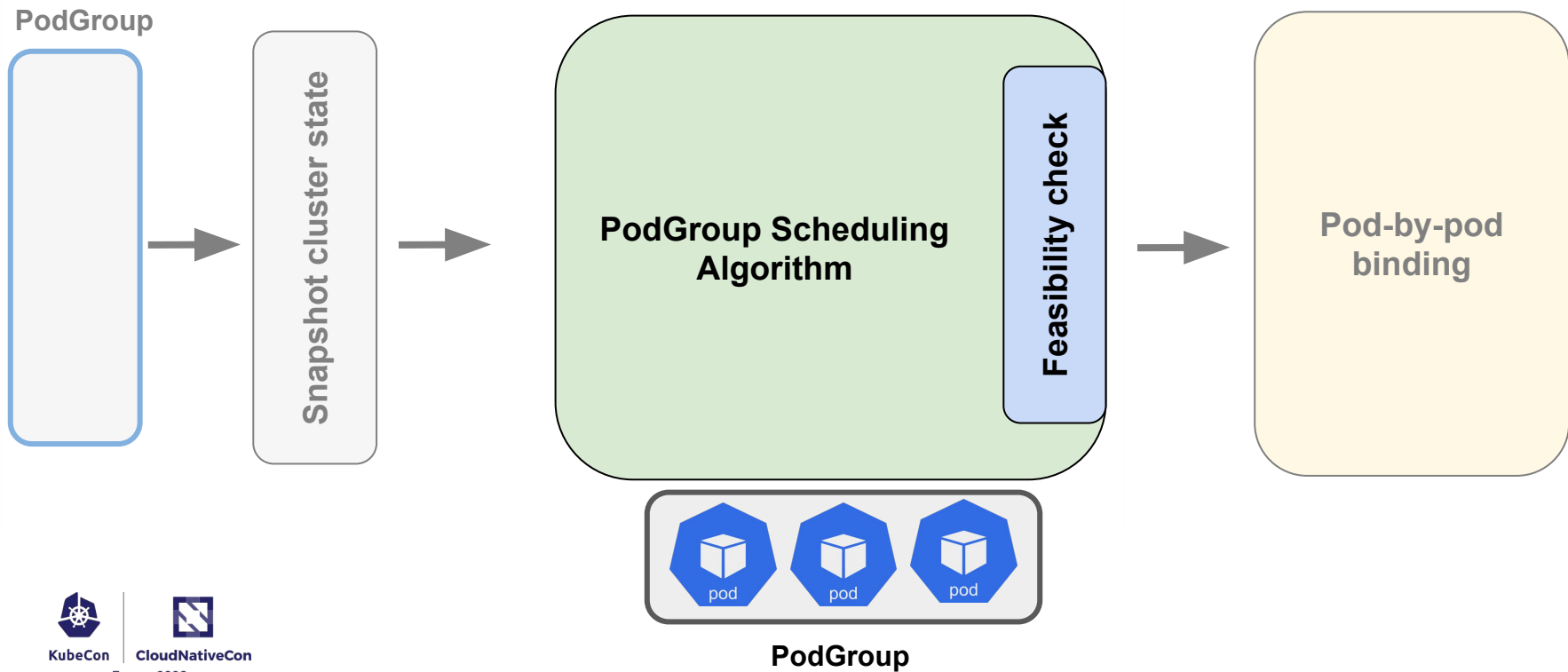
PodGroup Scheduling Cycle



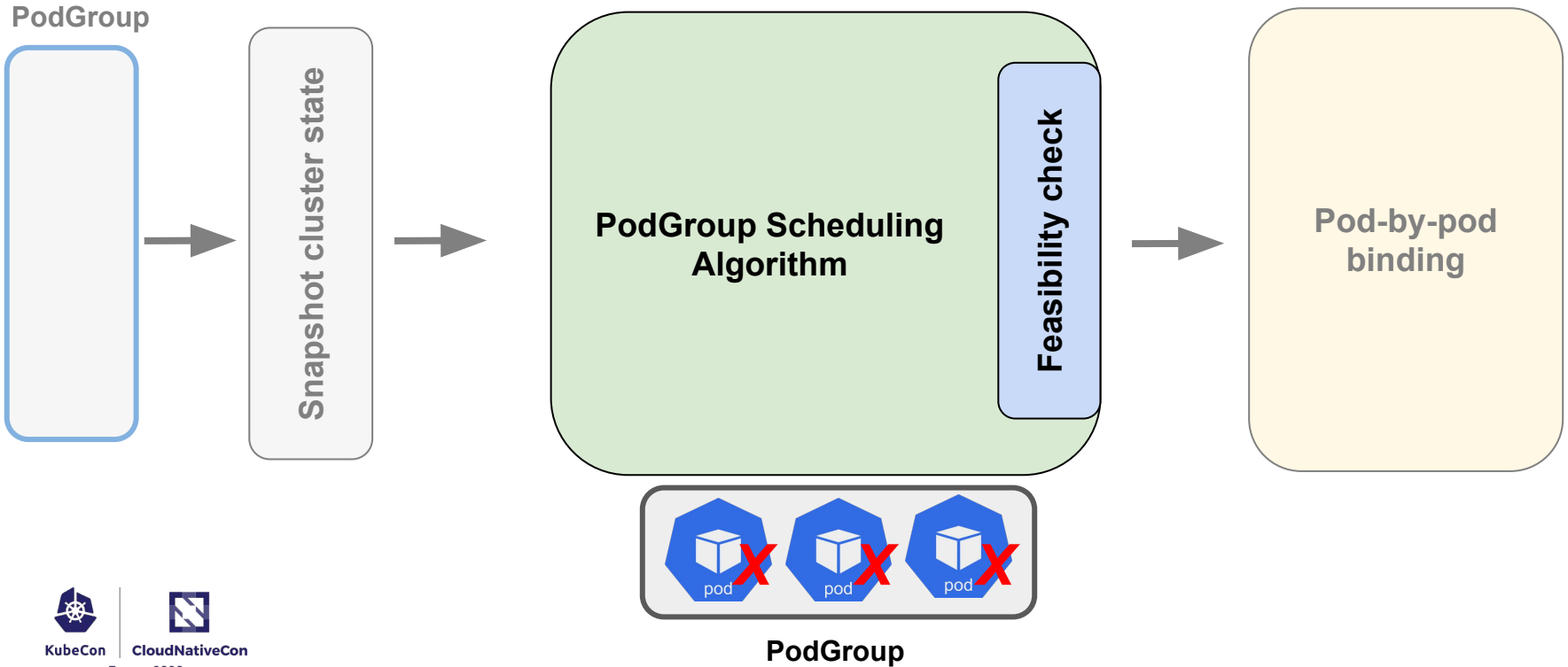
PodGroup Scheduling Cycle



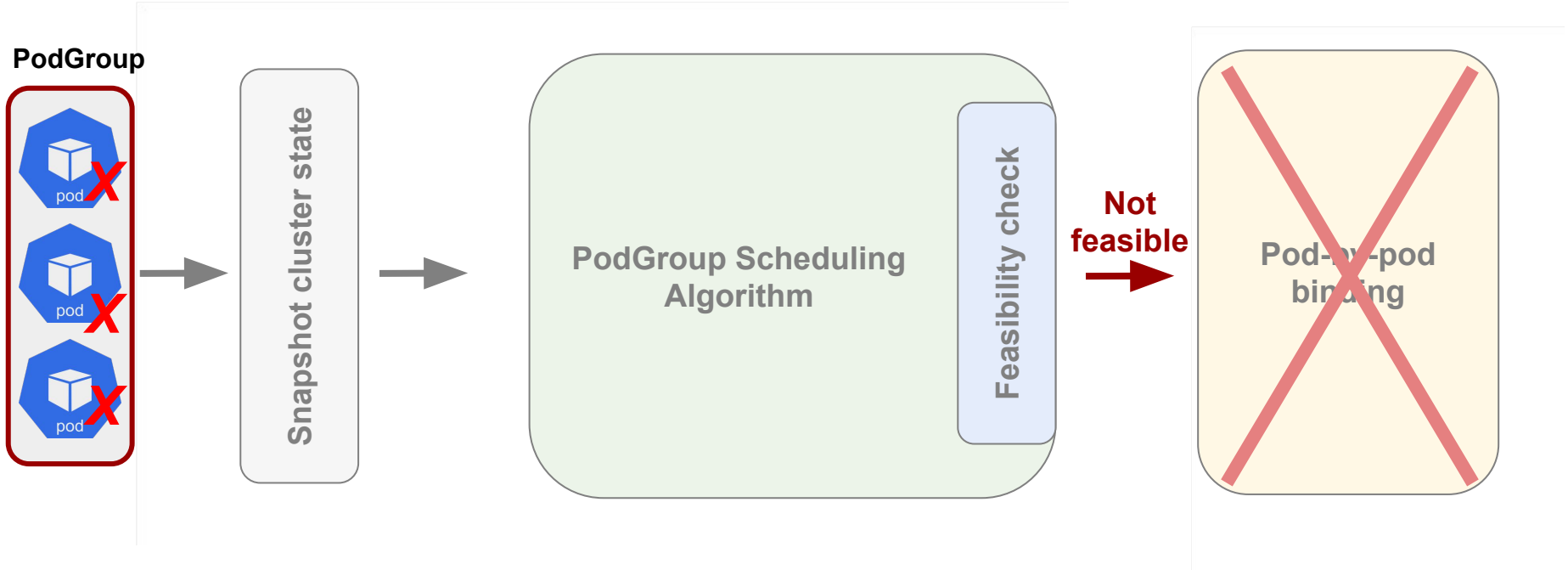
PodGroup Scheduling Cycle



PodGroup Scheduling Cycle



PodGroup Scheduling Cycle



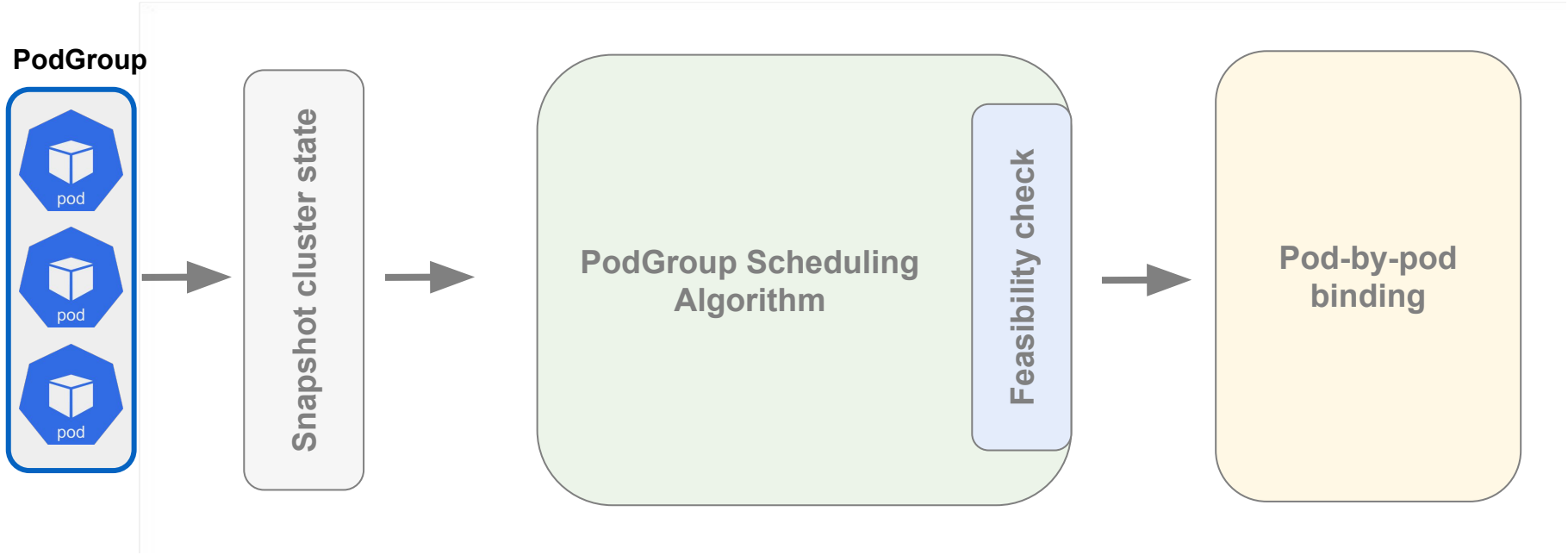
KubeCon



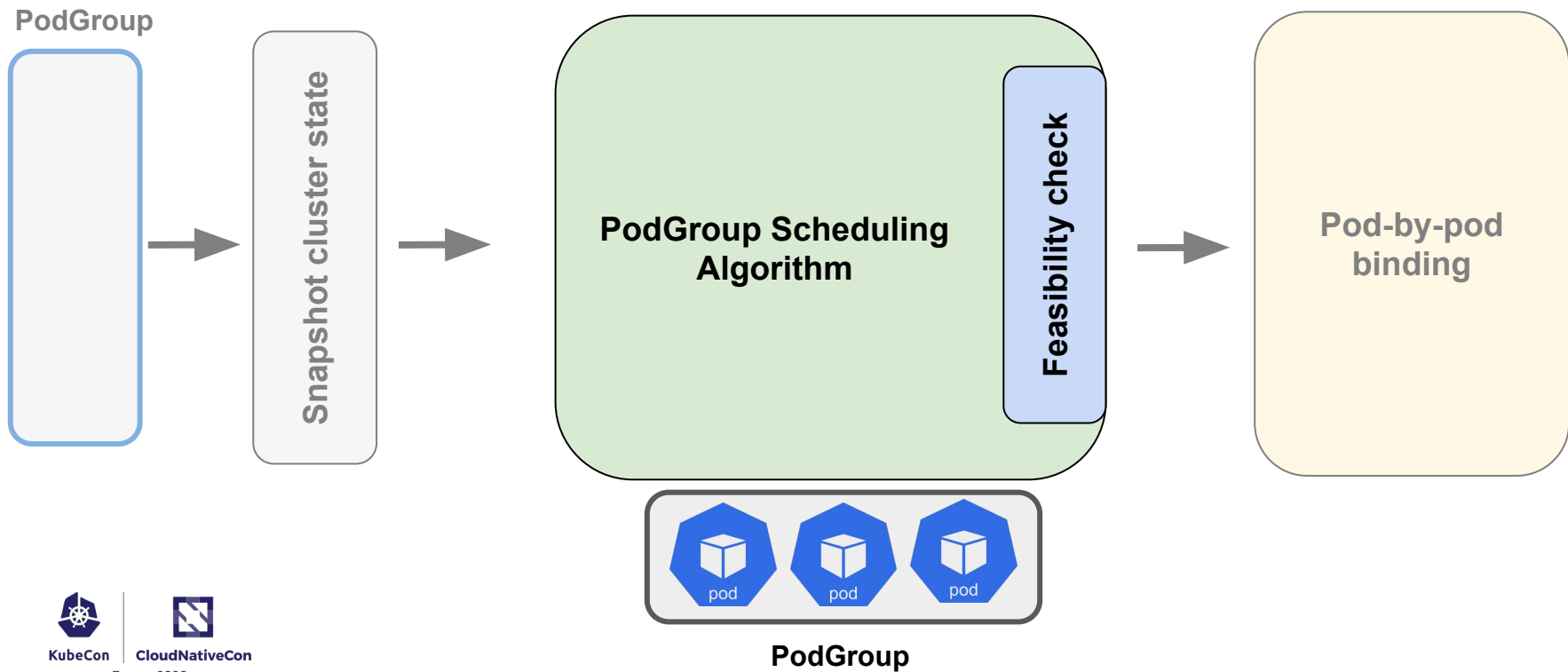
CloudNativeCon

Europe 2026

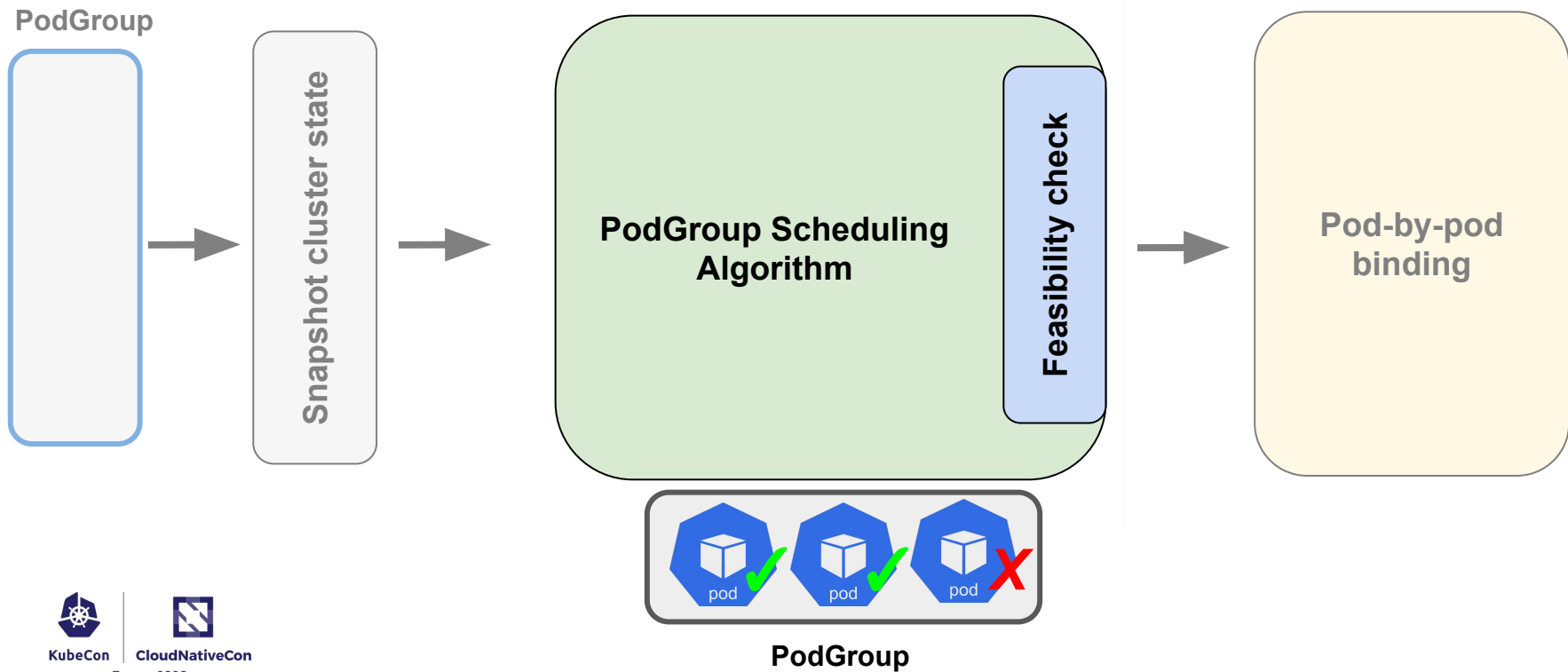
PodGroup Scheduling Cycle



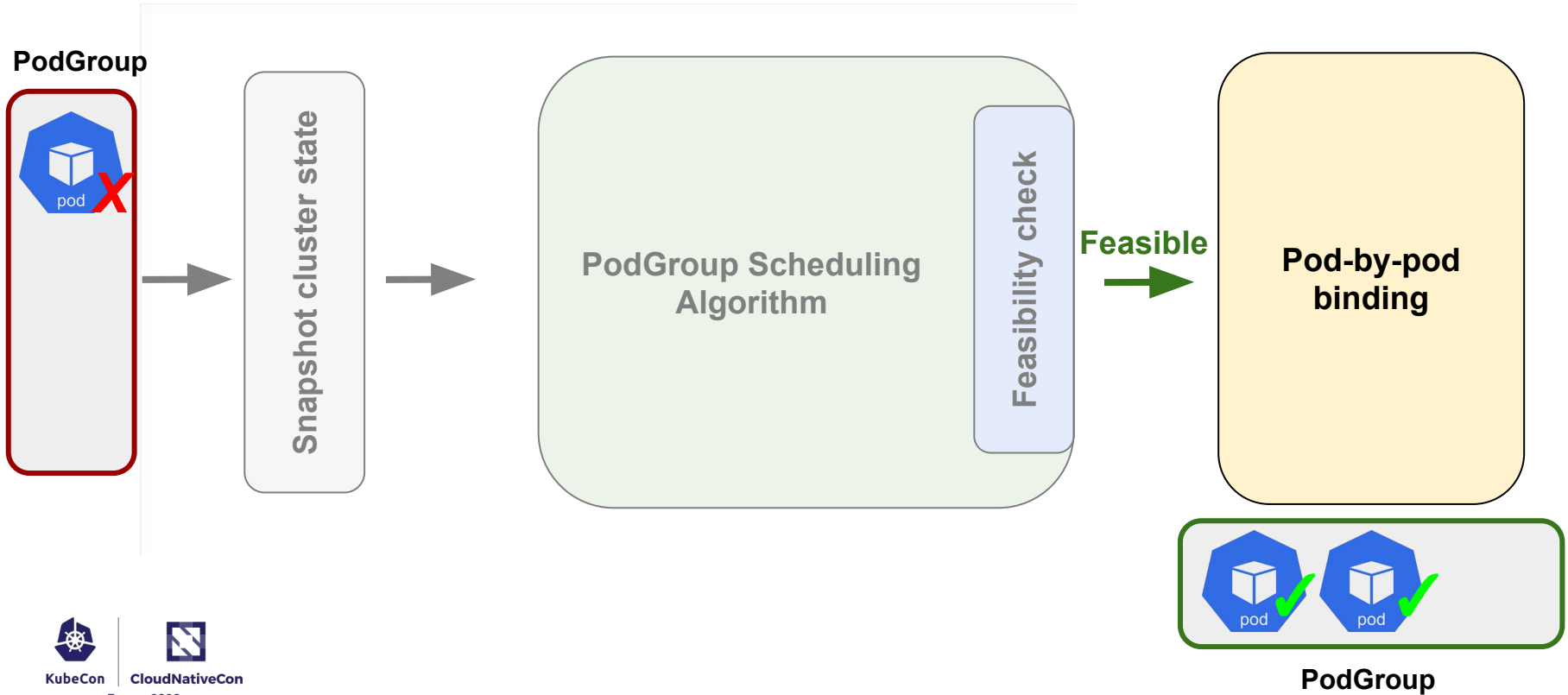
PodGroup Scheduling Cycle



PodGroup Scheduling Cycle



PodGroup Scheduling Cycle



KubeCon



CloudNativeCon

Europe 2026

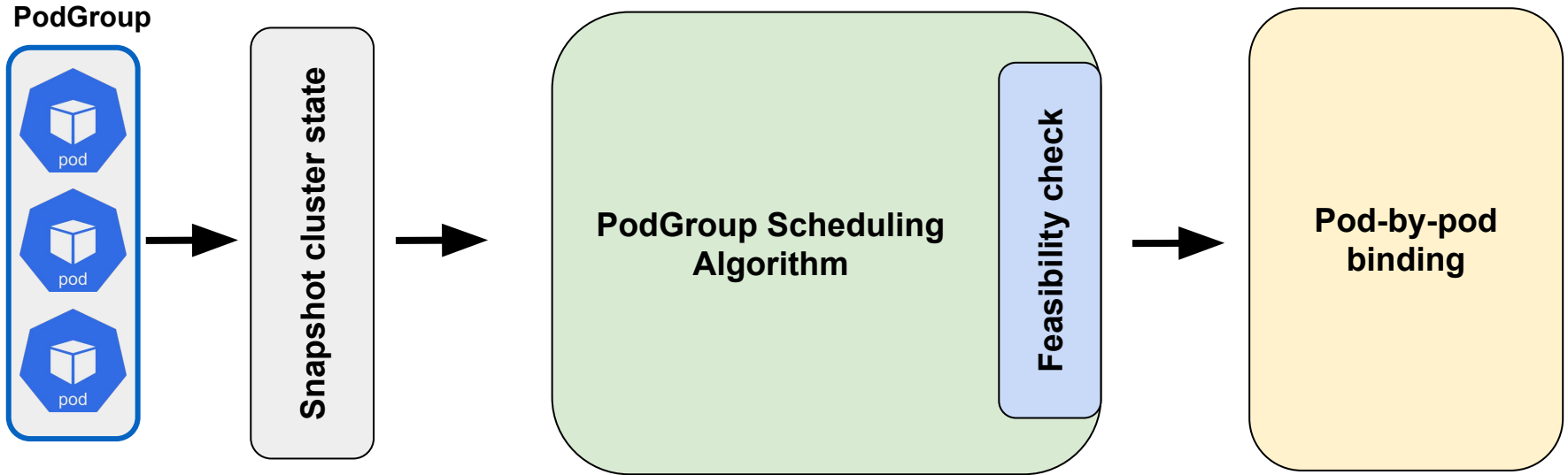
PodGroup Scheduling - Limitations & Future

- No PodGroup-oriented Scheduling Queue implementation yet
- Preemption actuation is not decoupled from per-Pod victim calculation
- The algorithm is prone to permanent unschedulability for heterogeneous PodGroups
- Inter-pod dependencies (e.g., Pod affinity) are partially incompatible

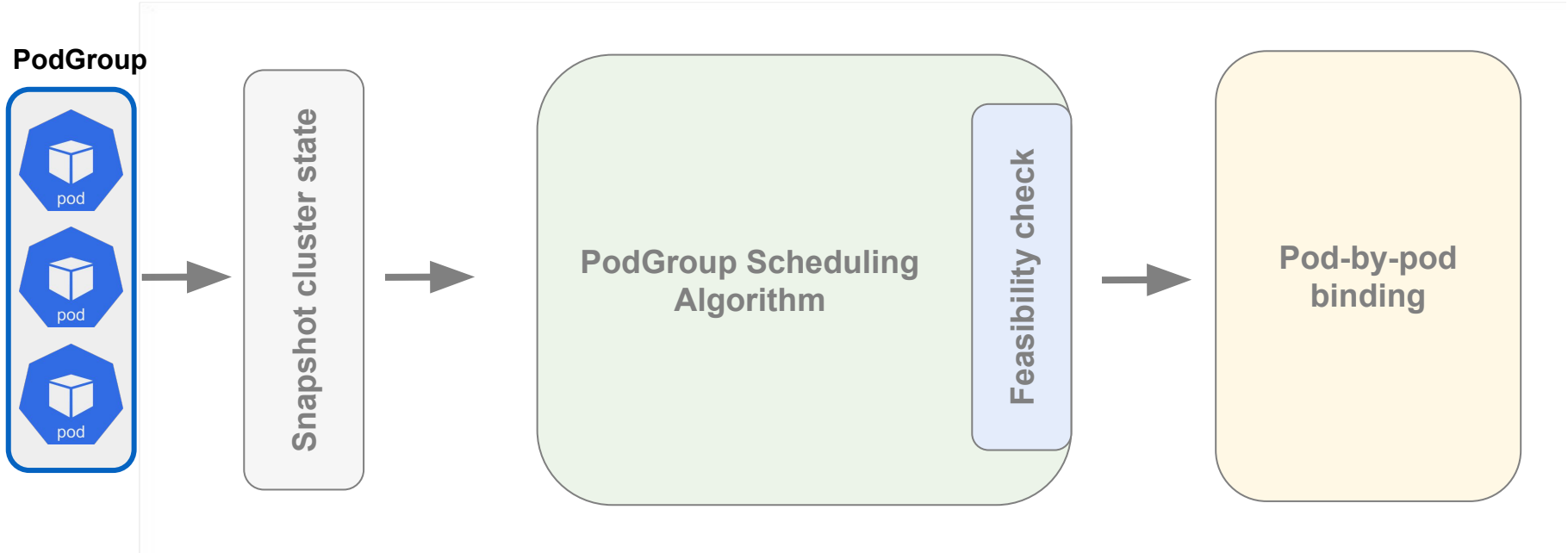
Workload-Aware Preemption

- New preemption algorithm is called *after* the PodGroup Scheduling Algorithm considers the entire PodGroup as not feasible
- Workload-Aware Preemption initially heavily reuses the Pod-based preemption concepts, but with the PodGroup as a preemptor and **the entire cluster as a preemption domain**
- New fields in the PodGroup API
 - *PriorityClassName/Priority*: used to determine PodGroup-level priority during preemption
 - *DisruptionMode*: used to select either a Pod or a PodGroup as a preemption unit

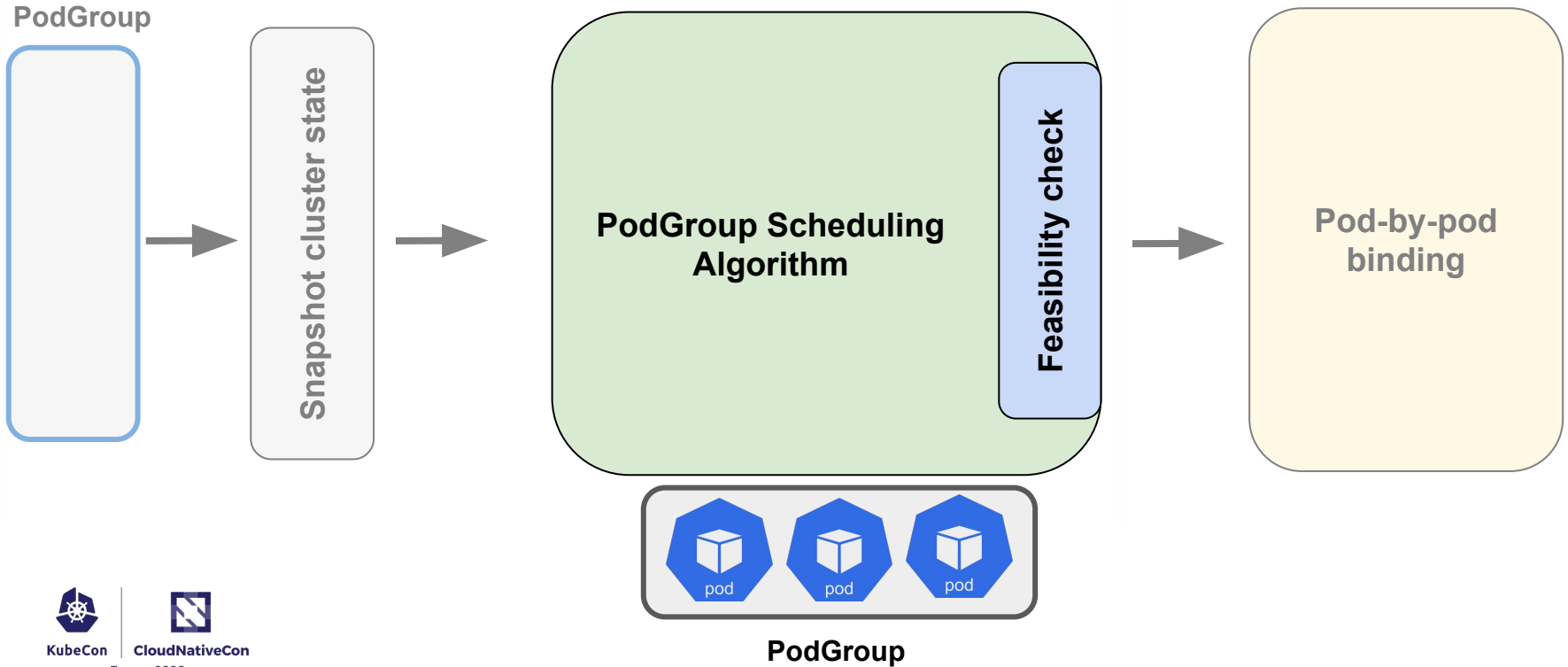
Workload-Aware Preemption



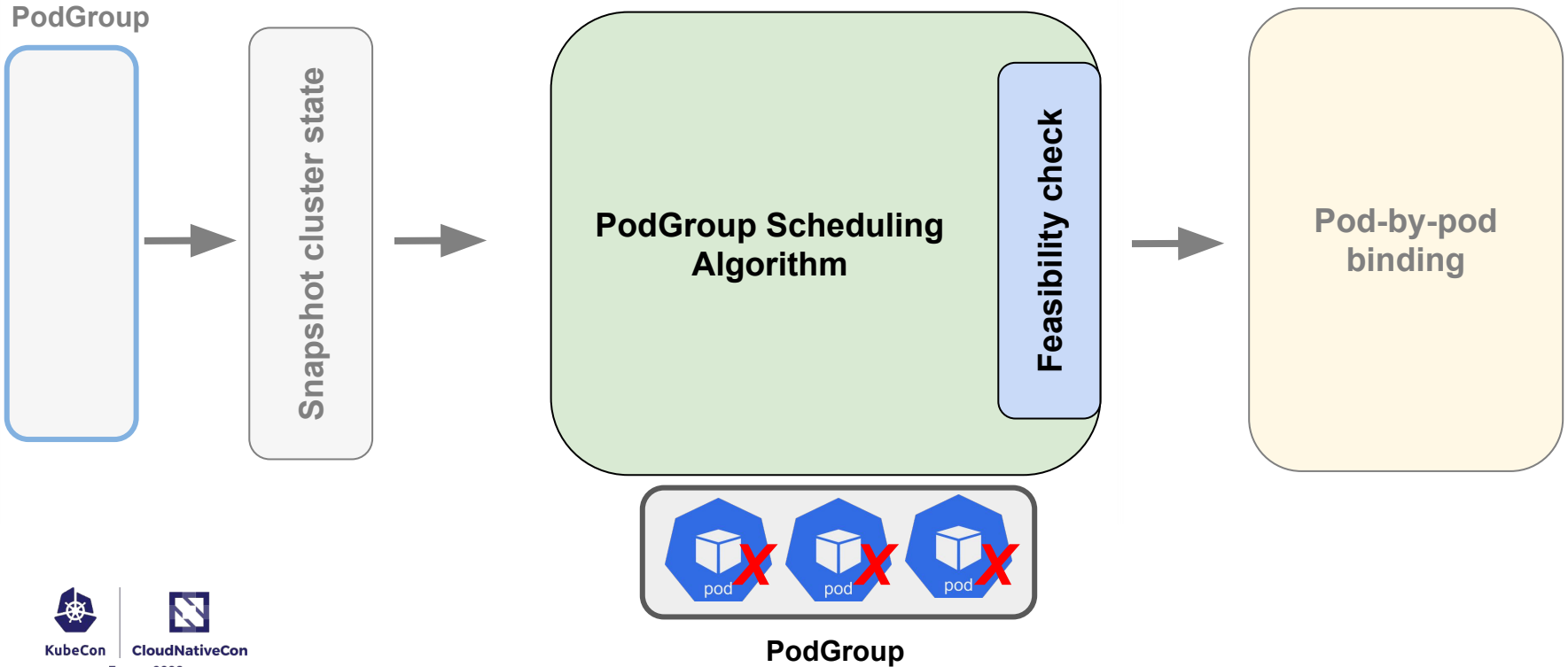
Workload-Aware Preemption



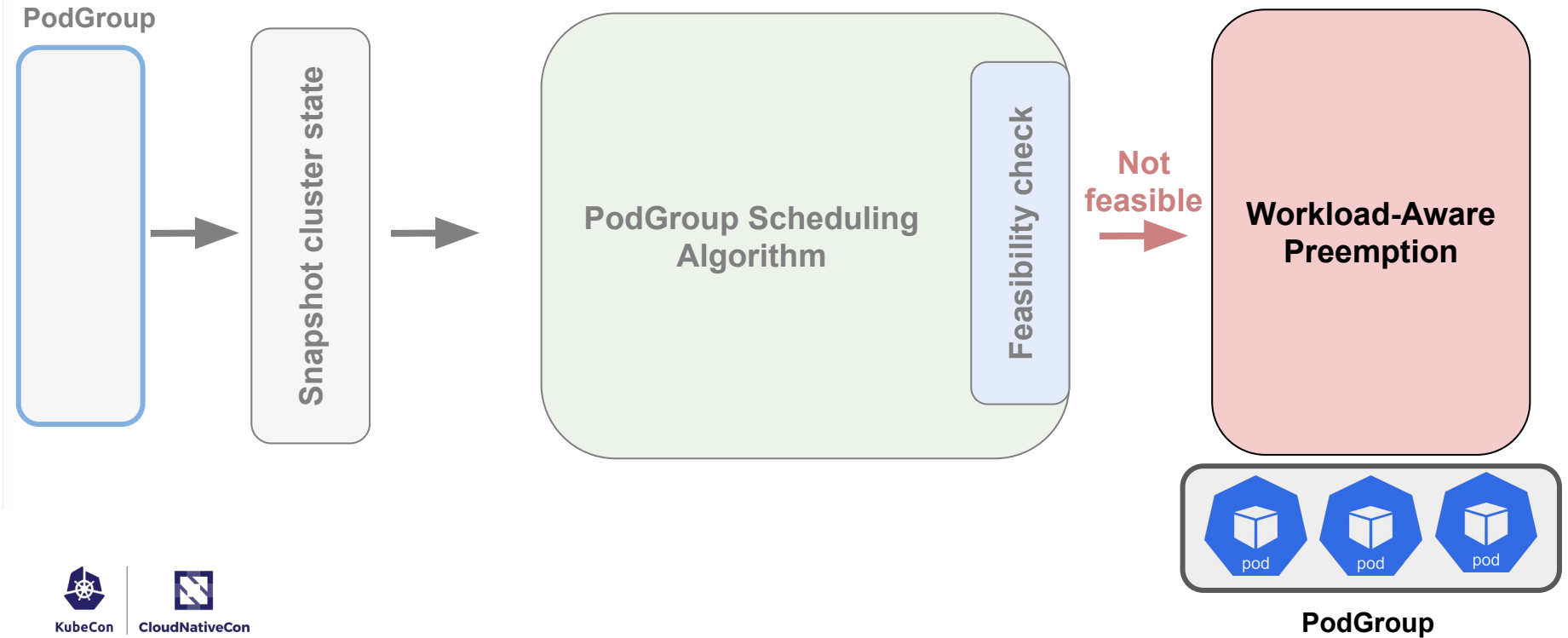
Workload-Aware Preemption



Workload-Aware Preemption

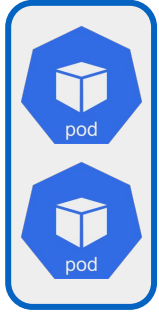


Workload-Aware Preemption



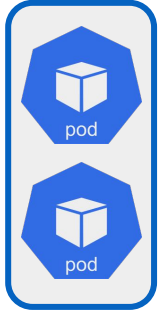
Workload-Aware Preemption

Preemptor
PodGroup



Workload-Aware Preemption

**Preemptor
PodGroup**



Preempts



Victim Pod's PodGroup API



KubeCon

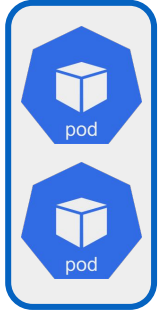


CloudNativeCon

Europe 2026

Workload-Aware Preemption

**Preemptor
PodGroup**



Preempts



Victim Pod's PodGroup API

DisruptionMode: Pod



KubeCon

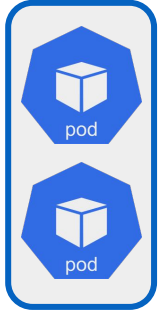


CloudNativeCon

Europe 2026

Workload-Aware Preemption

Preemptor
PodGroup

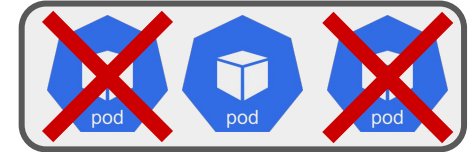


Preempts



Victim Pod's PodGroup API

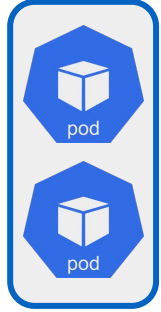
DisruptionMode: Pod



Victim Pod's PodGroup

Workload-Aware Preemption

Preemptor
PodGroup



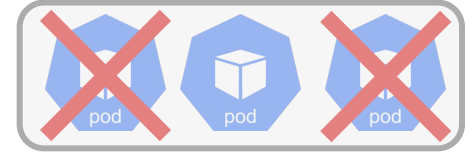
Preempts



Victim Pod's PodGroup API

DisruptionMode: Pod

DisruptionMode: PodGroup



Victim Pod's PodGroup



KubeCon

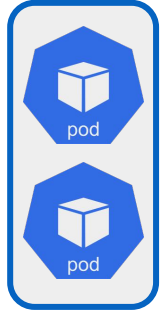


CloudNativeCon

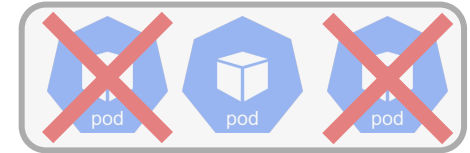
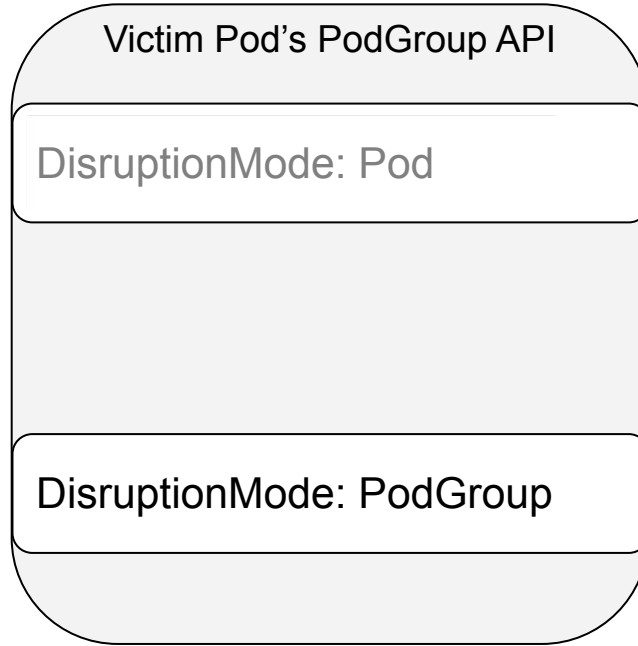
Europe 2026

Workload-Aware Preemption

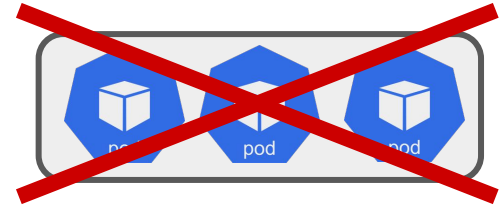
Preemptor
PodGroup



Preempts



Victim Pod's PodGroup



Victim Pod's PodGroup

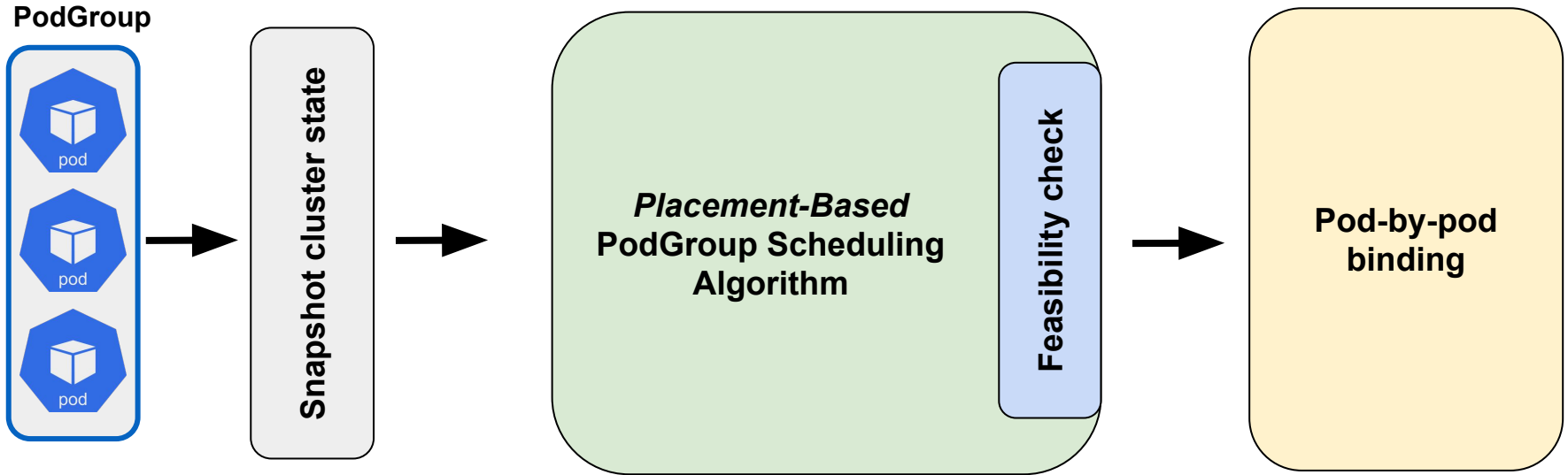
Workload-Aware Preemption - Limitations & Future

- Default Pod-based preemption is ignoring existence of PodGroup and their disruption mode
- Other PostFilter plugins (e.g., DRA) require adjustments for better integration
- New extension point: PodGroupPostFilter is planned in v1.37

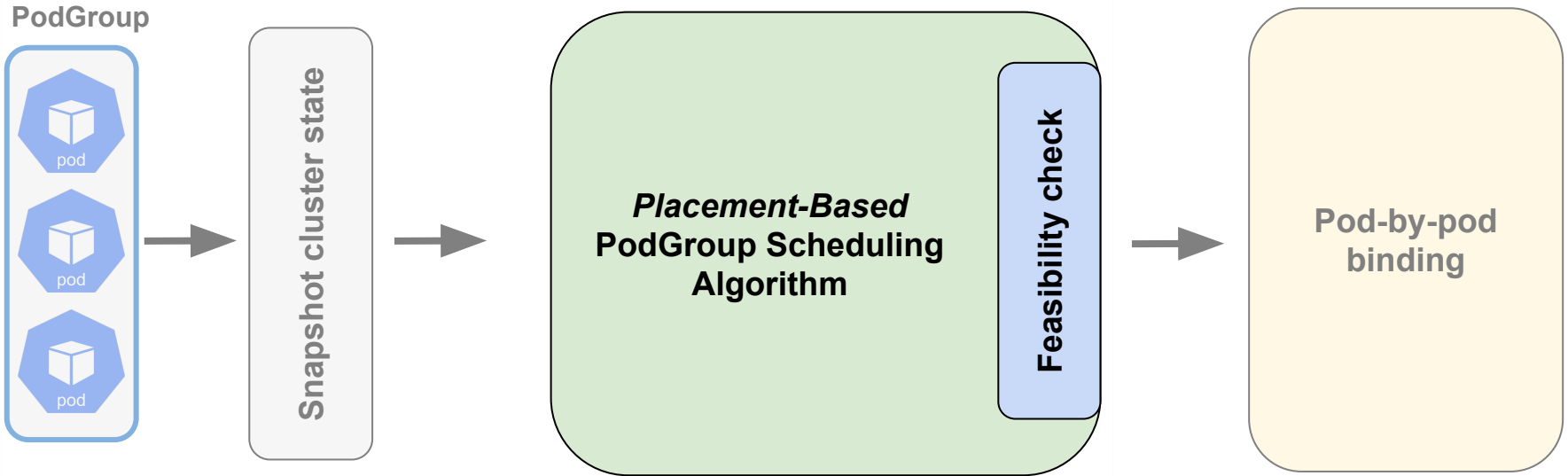
Topology-Aware Scheduling

- New fields in the PodGroup API
 - Topology constraints defining a topology key
- New Placement-based PodGroup Scheduling Algorithm with new extension points
- Topology-Aware Scheduling implemented as a plugin

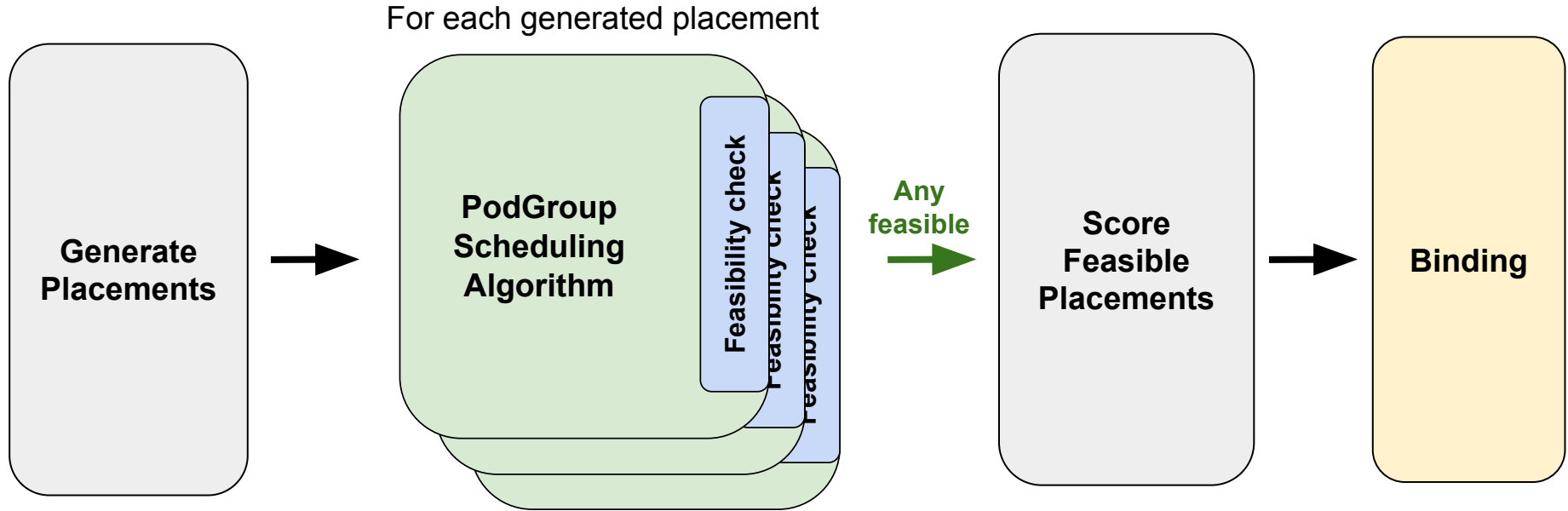
Topology-Aware Scheduling - Algorithm



Topology-Aware Scheduling - Algorithm



Topology-Aware Scheduling - Algorithm



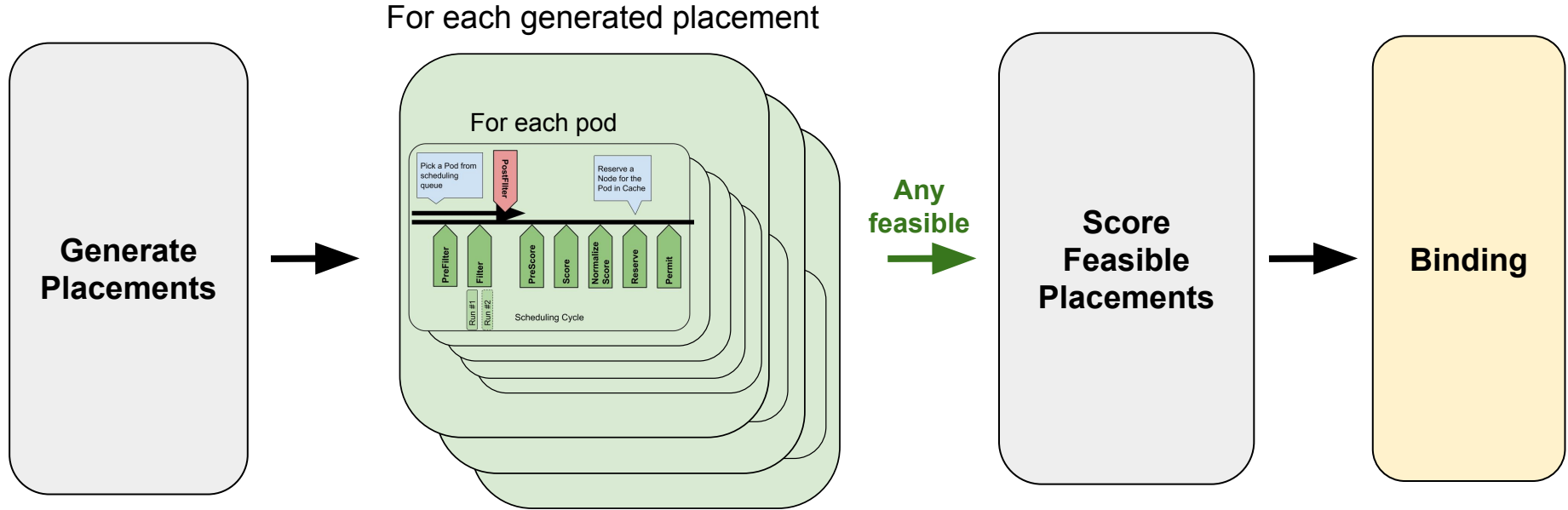
KubeCon



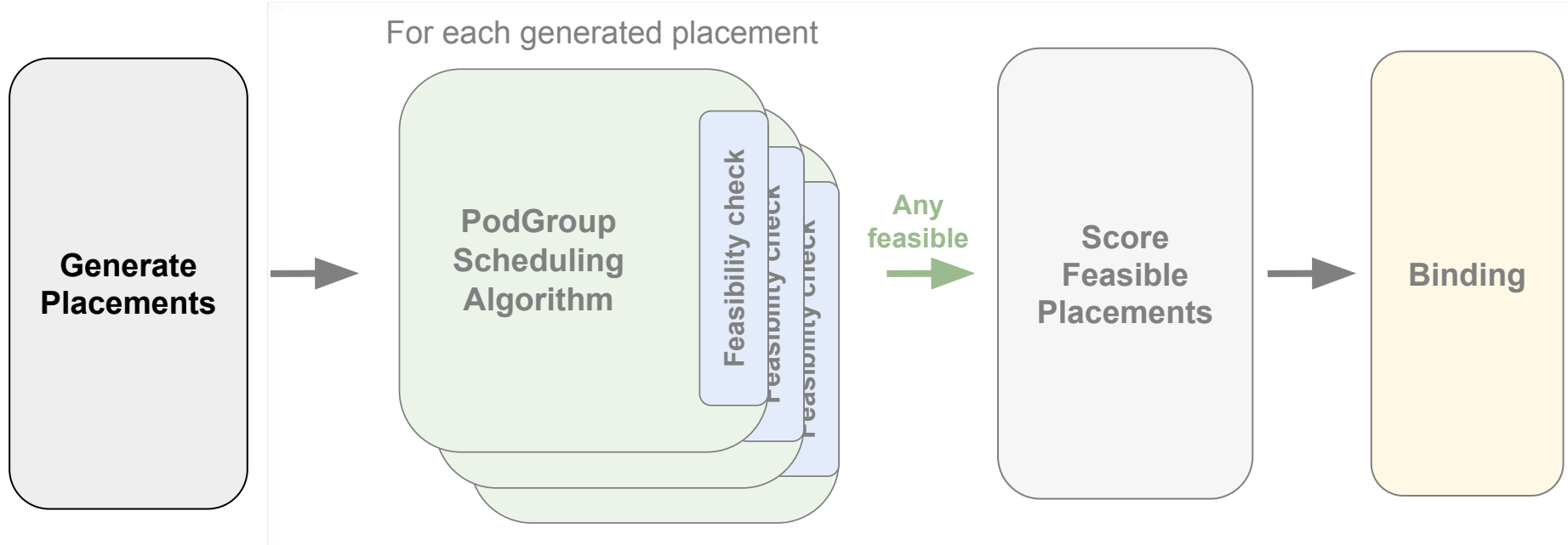
CloudNativeCon

Europe 2026

Topology-Aware Scheduling - Algorithm



Topology-Aware Scheduling - Algorithm



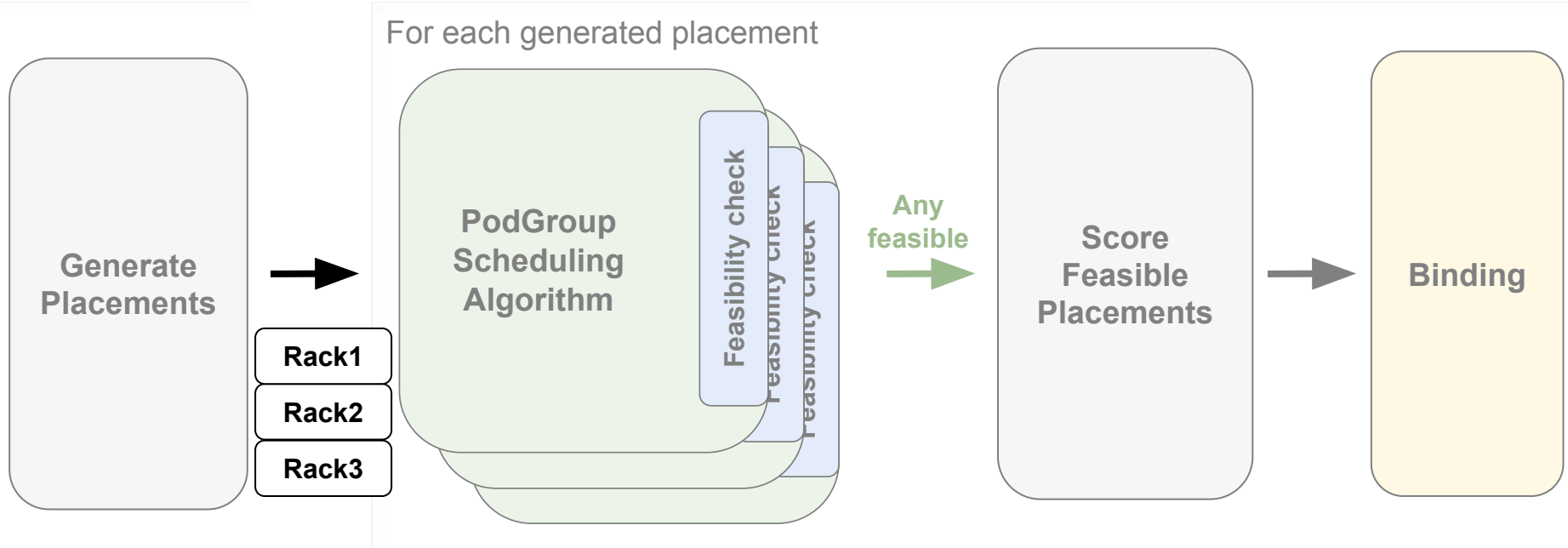
KubeCon



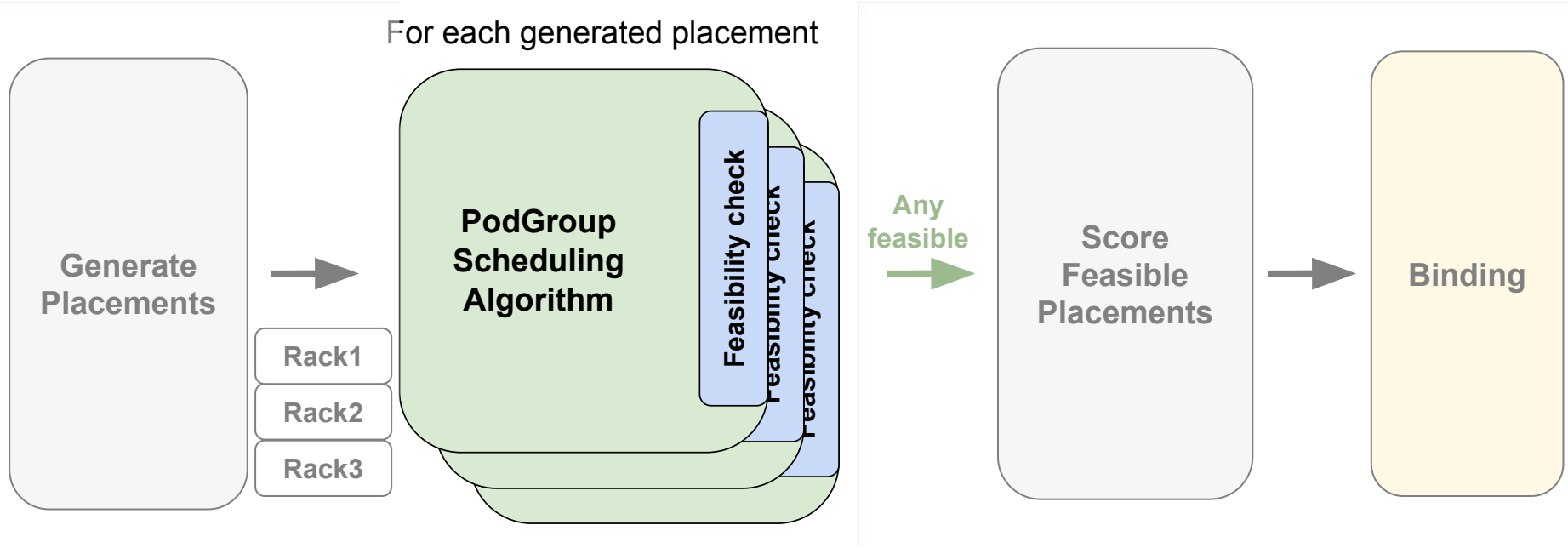
CloudNativeCon

Europe 2026

Topology-Aware Scheduling - Algorithm



Topology-Aware Scheduling - Algorithm



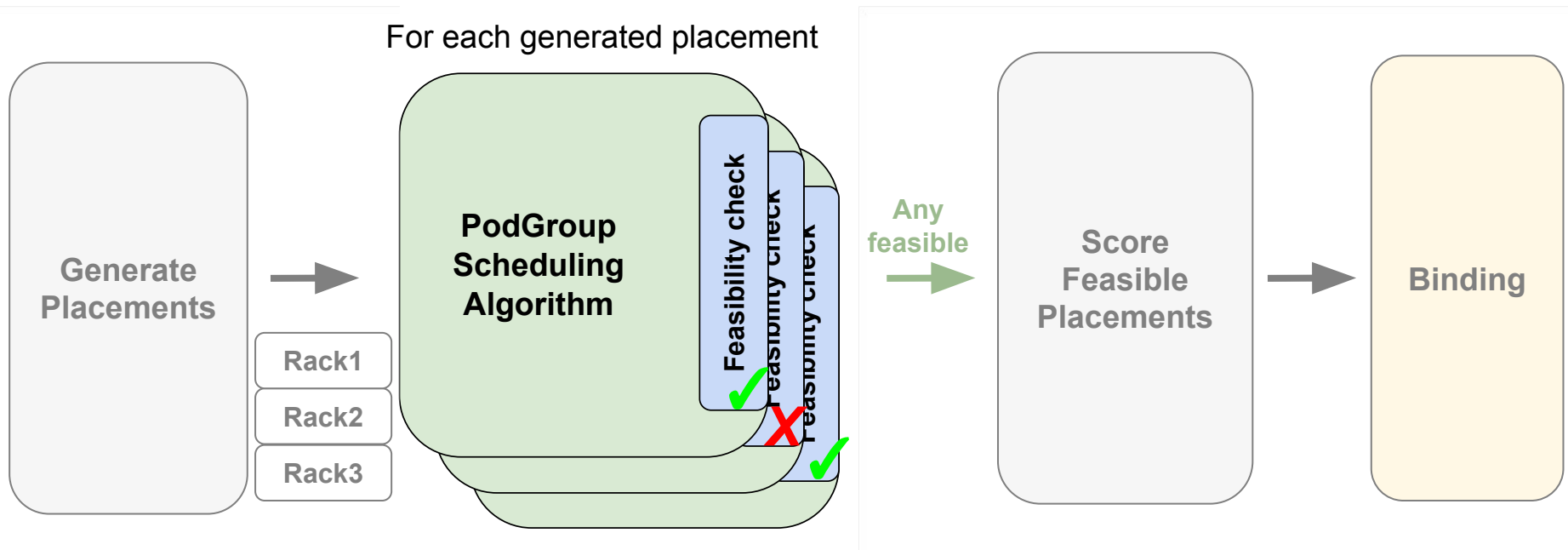
KubeCon



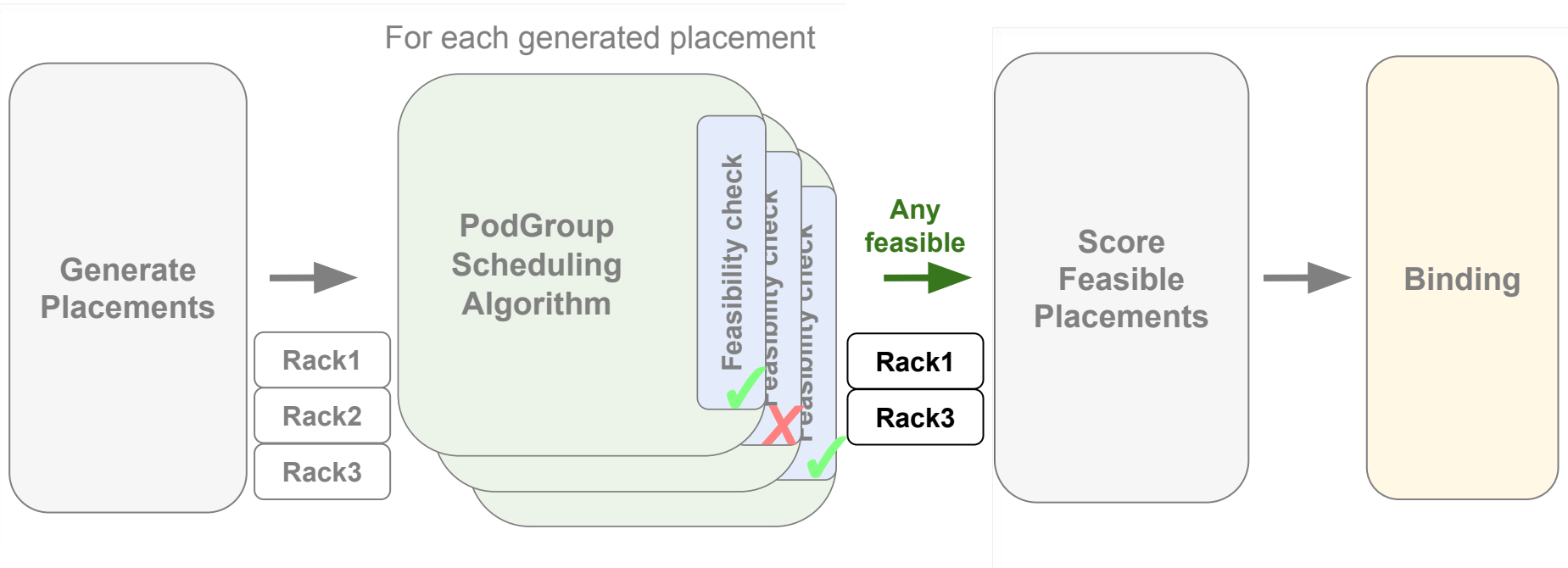
CloudNativeCon

Europe 2026

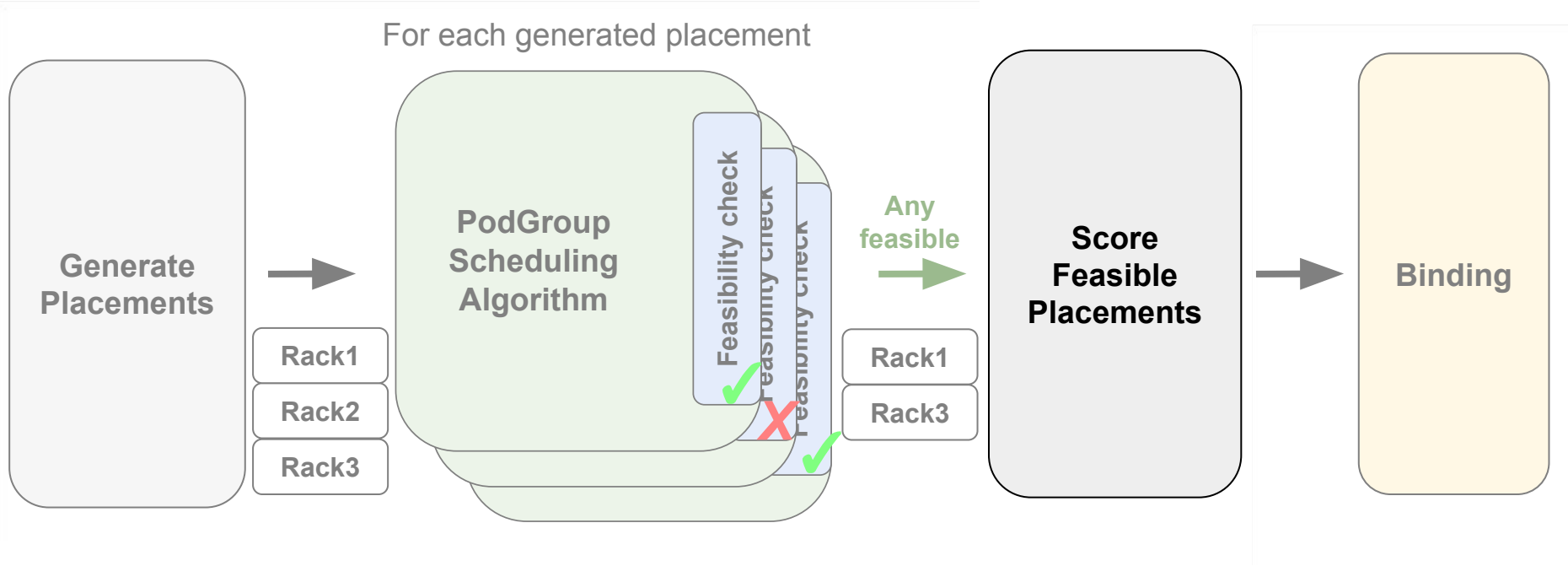
Topology-Aware Scheduling - Algorithm



Topology-Aware Scheduling - Algorithm



Topology-Aware Scheduling - Algorithm



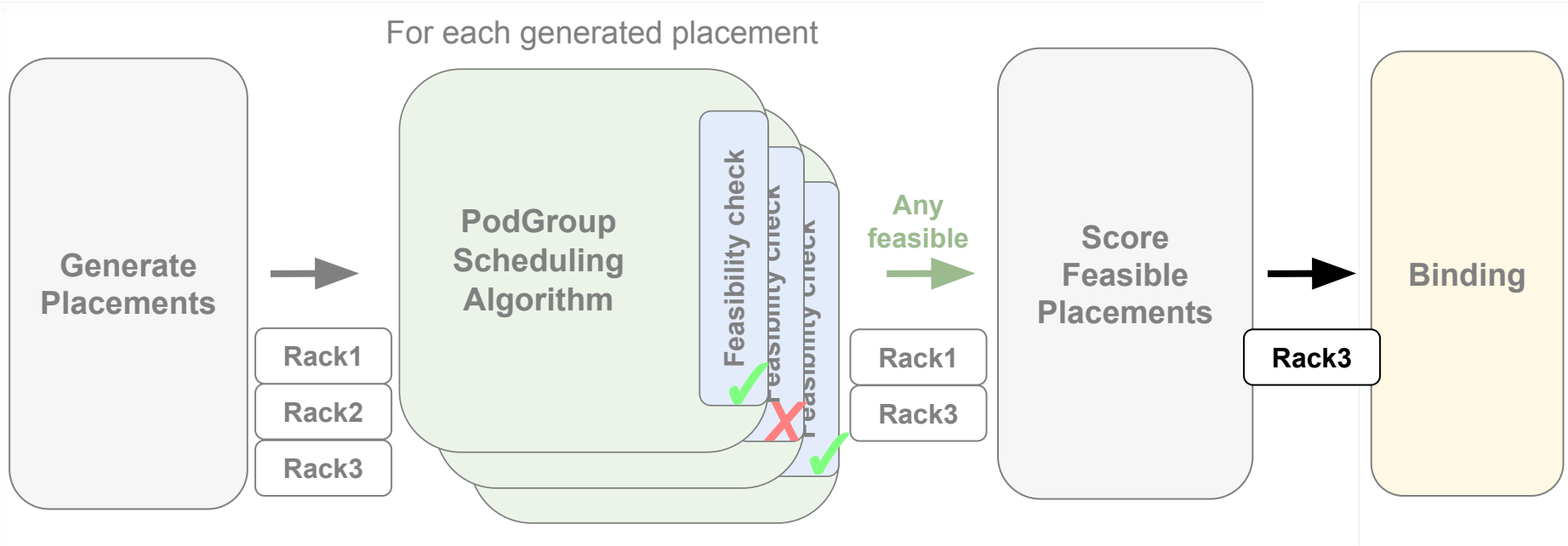
KubeCon



CloudNativeCon

Europe 2026

Topology-Aware Scheduling - Algorithm



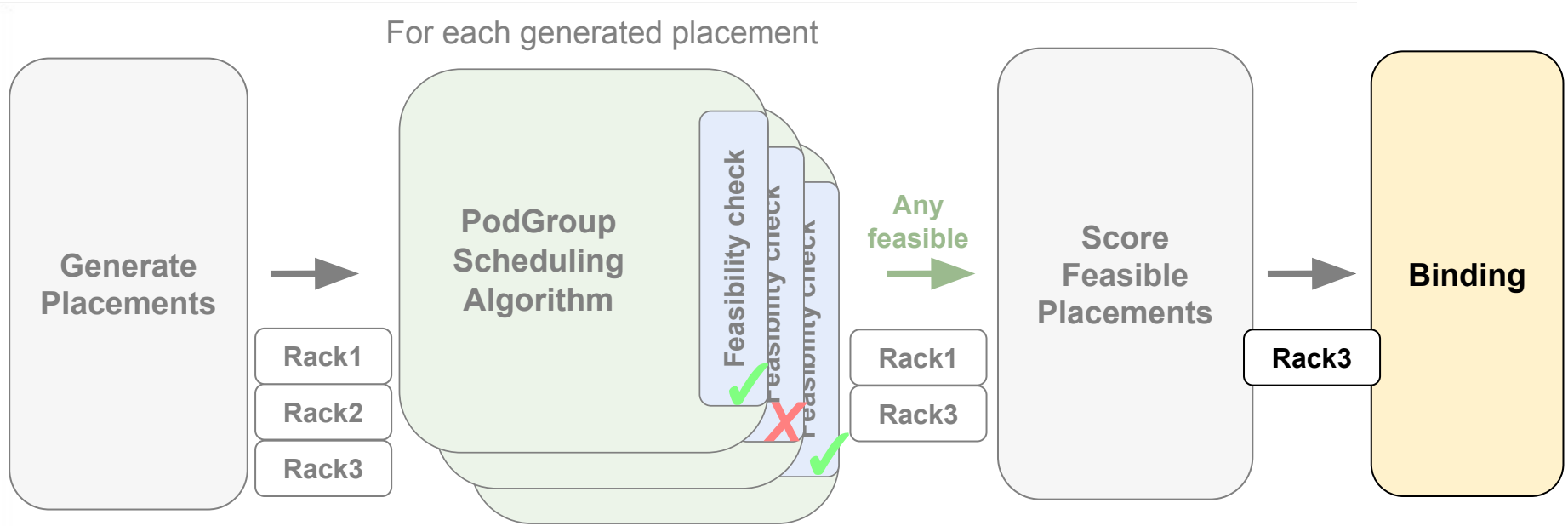
KubeCon



CloudNativeCon

Europe 2026

Topology-Aware Scheduling - Algorithm



KubeCon



CloudNativeCon

Europe 2026

Topology-Aware Scheduling - Limitations & Future

- Dynamic Resource Allocation plugin doesn't yet implement the introduced extension points
- Initial version is limited to single-level topology constraints
- Algorithm is gang optimized with poor support for non-gangs
- Currently lacks integration with default preemption and Workload-Aware Preemption when the PodGroup is using TAS constraints

WAS integration in v1.36

- ResourceClaim support for PodGroups (KEP-5729)
 - Adds a support for PodGroup-level ResourceClaims in PodGroup API
- MVP Job controller integration (KEP-5547)

Future plans

- **Promotion to beta in v1.37**
 - Address limitations and improve current features: PodGroup Cycle, Gang Scheduling, Topology-Aware Scheduling, Workload-Aware Preemption
- Extend PodGroup API with multi-layer structure (PodSubGroup, PodSet)
- Integration with workload controllers (JobSet, LWS, etc.)
- Integration with orchestrators (Kueue, Grove, etc.)
- Integration with Autoscalers (Cluster Autoscaler, Karpenter, etc.)



KubeCon



CloudNativeCon

Europe 2026

Other Recent Updates

Other Recent Updates (DRAs!)

- DRA: Partitionable Devices (stable at v1.36)
- DRA: Prioritized Alternatives in Device Requests (beta at v1.36)
- DRA: Device Binding Conditions (beta at v1.36)
- DRA: Device taints and tolerations (beta at v1.36)
- DRA: Consumable Capacity (beta at v1.36)
- DRA: List Types for Attributes (alpha at v1.36)
- DRA: Native Resource Requests (alpha at v1.36)

Other Recent Updates (DRAs!)

- DRA: Partitioned

The Latest in GPU, TPU, NIC and Other Device Support - WG Device Management - John Belamaric, Google & Patrick Ohly, Intel

📅 Wednesday March 25, 2026 17:30 - 18:00 CET

📍 E106-108

WG Device Management continues to make great progress in enhancing support for GPUs, TPUs, NICs, and other specialized hardware in Kubernetes.

With the 1.34 release, Dynamic Resource Allocation (DRA) has finally reached General Availability, making it easier than ever to configure, allocate, and share advanced hardware resources efficiently. But the 1.34 support is just the basics, and the work is nowhere near done!

Come learn about what the community has built in 1.35 and what's coming in 1.36 to improve how you use specialized devices in Kubernetes, such as features for managing device failures, groups of devices working together across nodes, and controlled sharing of devices.

Today: 17:30 - 18:00 !!



KubeCon



CloudNativeCon

Europe 2026



KubeCon



CloudNativeCon

Europe 2026

Subprojects

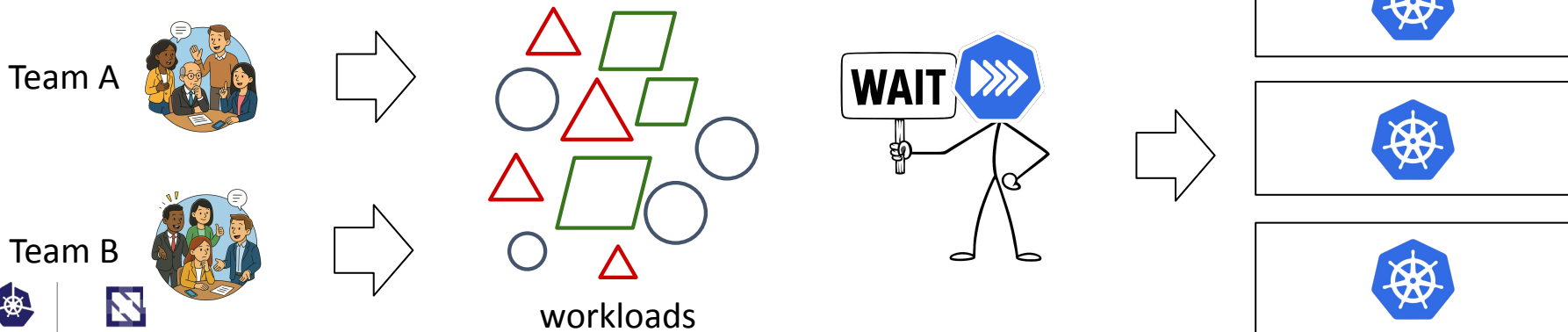
Kueue: what's Kueue?

- **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)



Kueue: Recent Highlights

v0.15.1 - 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
- And more ...

Kueue: Recent Highlights

WG-Batch Updates: What's New and What Is Next? - Yuki Iwai, CyberAgent, Inc. & Kevin Hannon, Red Hat

Thursday March 26, 2026 15:15 - 15:45 CET
E103-105

Join WG Batch maintainers to explore how Kubernetes is evolving to meet the demands of the AI and HPC era. We will present foundational updates to the Kubernetes Job API, such as the managedBy and Mutable Container Resource features. These enhancements enable Job to unlock higher elasticity and efficiency.

Building on this, we will highlight JobSet advancements designed for resilience. We will introduce the InPlaceRestart strategy, which significantly accelerates recovery for large-scale distributed training by preserving execution context and minimizing the costly overhead of Pod recreation.

Additionally, we will demonstrate Kueue as a platform for AI/ML, which showcases how capabilities such as Multi-Cluster dispatching, Fair Sharing, and Topology-Aware Scheduling can maximize hardware utilization across tenants.

Finally, we will discuss our collaboration with SIG Scheduling on "Workload-Aware Scheduling", which aims to integrate batch semantics into the core scheduler deeply.

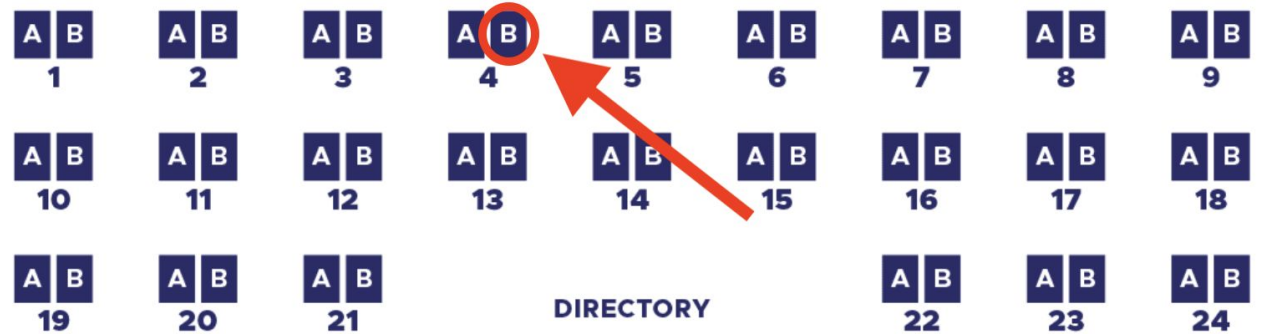
Tomorrow: 15:15 - 15:45 !!

Kueue and WAS booth

Tuesday 10:15 - 19:00

Thursday 10:00 - 12:00

Project Pavilion 4B



CNCF
PROJECTS
TEAM

DIRECTORY



JOB
BOARD

PROJECT
PAVILION
STAGE

STICKER
WALL

← Ad-Hoc Meeting Tables

Capture The Flag Collaboration →



KubeCon



CloudNativeCon

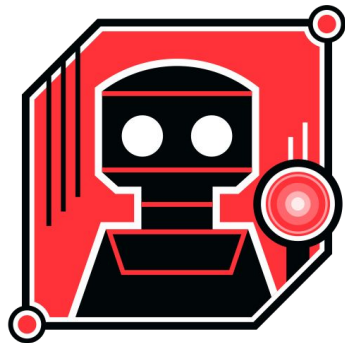
Europe 2026

descheduler

The Kubernetes scheduler only evaluates constraints at startup, meaning it won't move pods if conditions change. The **Descheduler** fixes this by monitoring running pods and evicting those that no longer meet your policies so they can be rescheduled.

What's new:

- **DefaultEvictor** plugin: a new namespace label selector argument to dynamically include namespaces at runtime
- **PodLifeTime** plugin: now also evicts pods based on status transitions, conditions, states, exit codes, and owner kinds.



DESCHEDULER

Getting involved

- **We need reviewers and long-term contributors!**
- [#sig-scheduling](#) Slack channel at Kubernetes Slack
- Periodical SIG Scheduling community meetings
 - Biweekly Asia & Europe meeting on Tuesday at 9:00 CET
 - Biweekly America & Europe meeting on Thursday at 10:00 PST (19:00 CET)
- **New!** Periodical Workload API integration meetings
 - Weekly meeting on Monday at 9:00 PST (18:00 CET)



KubeCon



CloudNativeCon

Europe 2026

