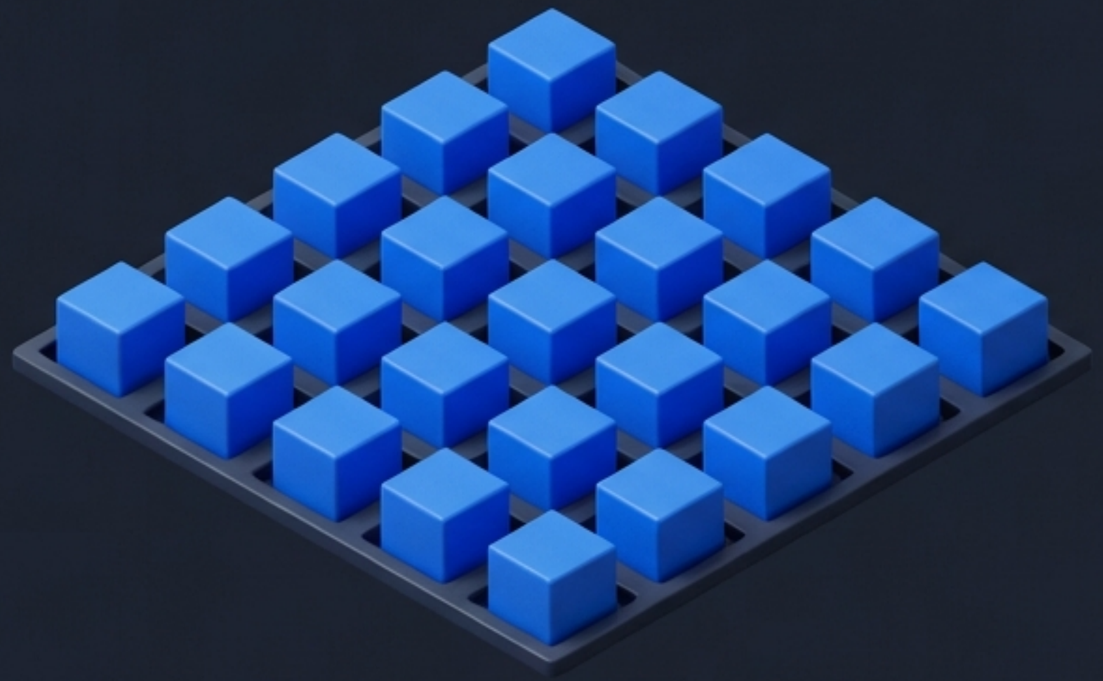
vcloudlabs

Mastering Kubernetes Pod Priority & Preemption

Ensuring the Right Workloads Run First.

When the Cluster is Full

The Perfect World



Infinite resources. No contention.

The Real World

Traffic spikes. Overlapping jobs.
Critical pods pending.

Not All Workloads Are Equal

By default, Kubernetes treats all pods equally. We must teach the scheduler to discriminate based on importance:
Live API User > Background Batch Job

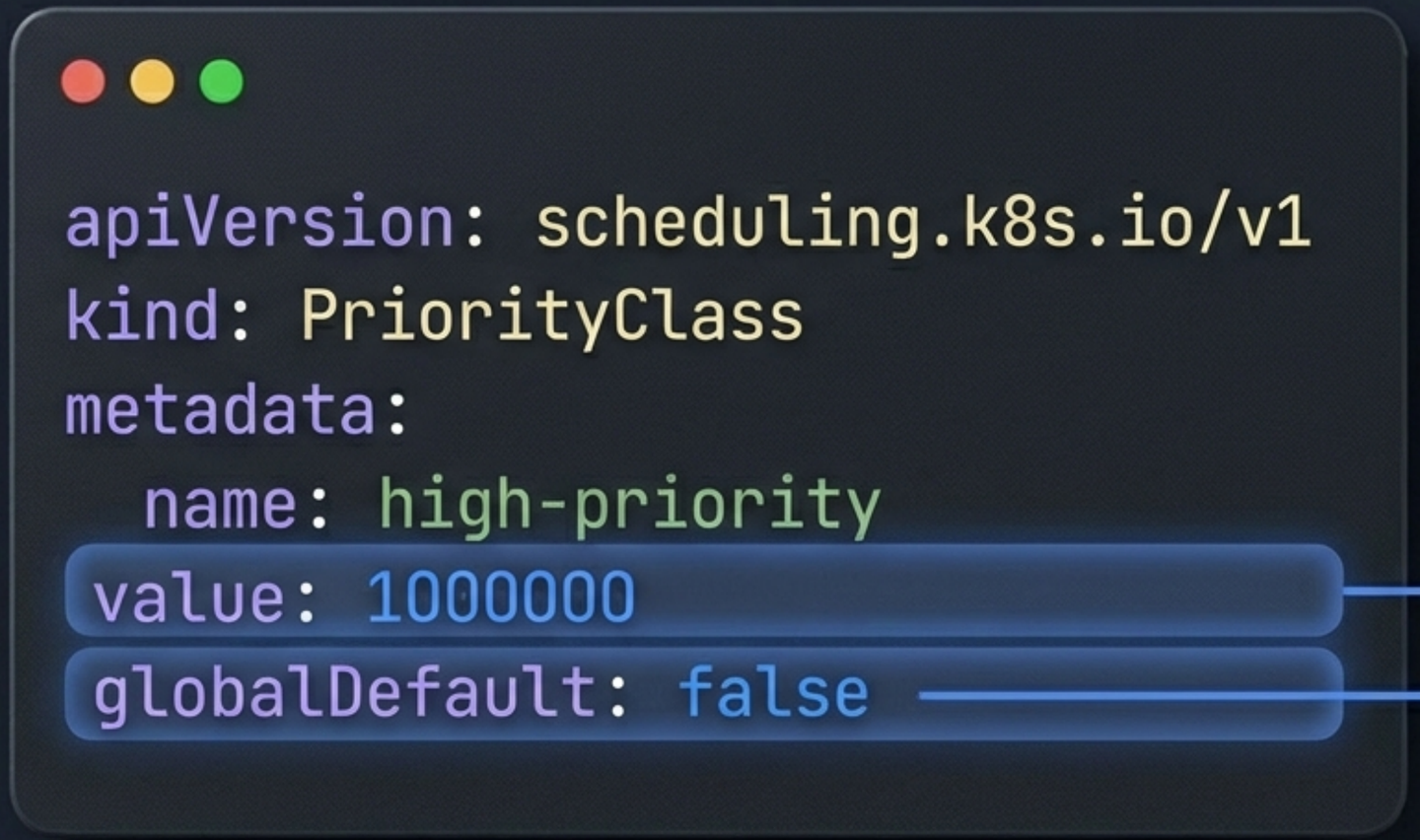
The PriorityClass Object

The Mechanism:
A cluster-level object mapping a name to a number.

The Rule:
Higher number = Higher priority.

Note: System components reserve the highest numbers.

Defining the Hierarchy



The Opt-In Model

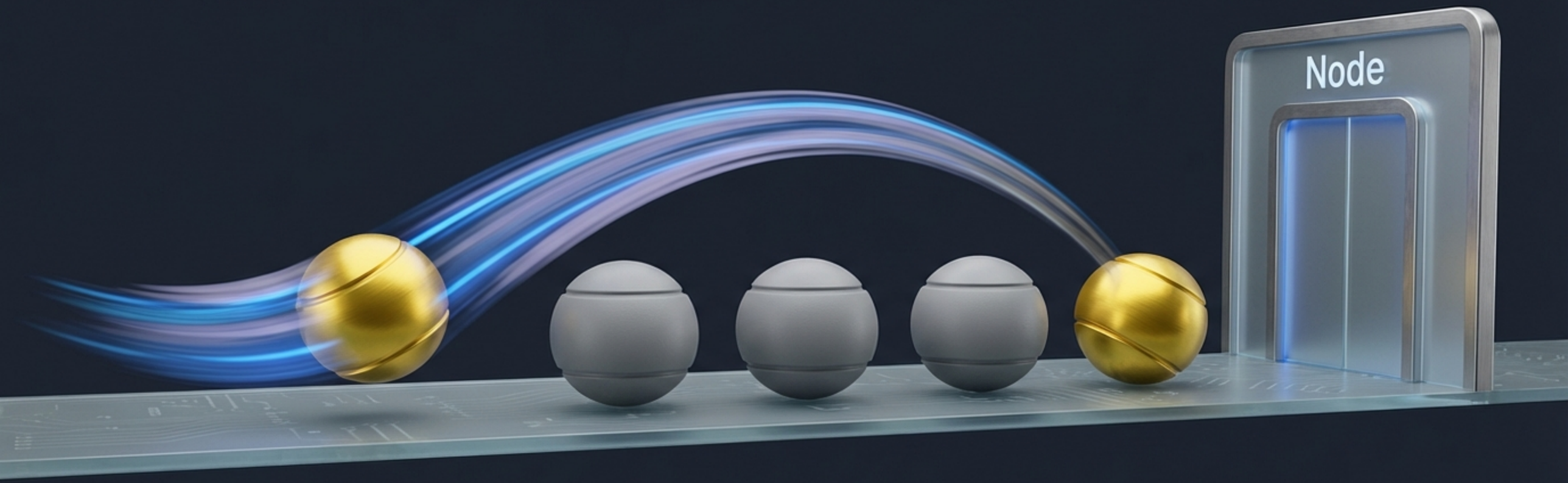
```yaml
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
  value: 1000000
globalDefault: false
```

Pod Configuration

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-critical
spec:
  priorityClassName: high-priority
```

At admission, the name 'high-priority' resolves to the numeric value 1,000,000.

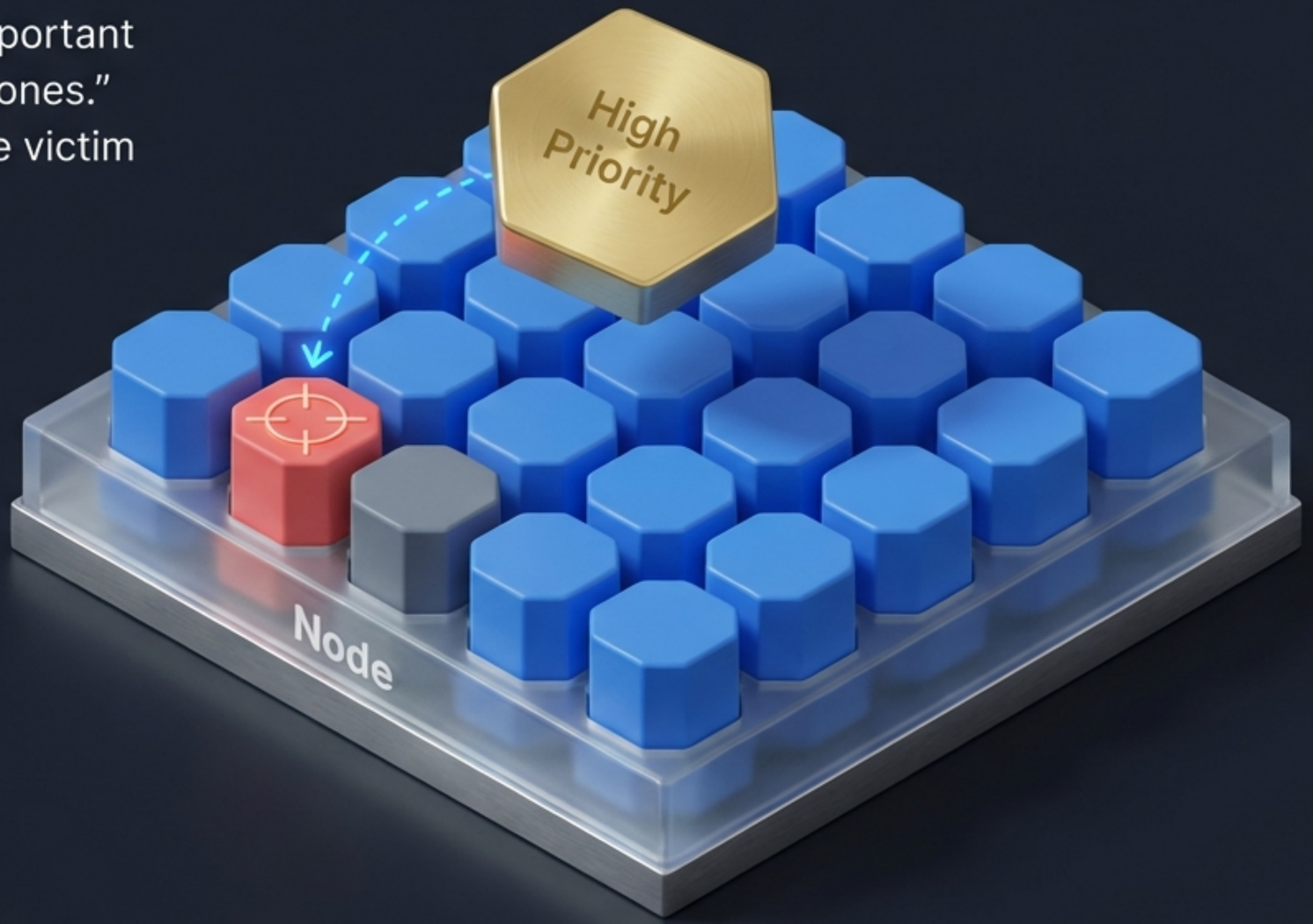
Jumping the Queue



The scheduler orders pending pods by priority. High-priority workloads cut the line. Low-priority workloads wait, regardless of arrival time.

Preemption: Making Space

When the cluster is full: “To run this important pod, I'm willing to evict less important ones.” Preemption only occurs if removing the victim creates enough space.



The Preemption Workflow

Scheduler finds no free resources on any node.

Identifies a node where evicting low-priority pods creates sufficient space.

Victims are removed. The VIP pod is scheduled.

The Fate of the Evicted

1. Evicted pods receive `SIGTERM`.
2. Graceful termination period begins.
3. Workloads must save state and exit.

CRITICAL WARNING

If they don't exit in time, they are forcefully killed (`SIGKILL`). Graceful shutdown logic is essential for low-priority jobs.

Important, But Polite

Configuration:

```
preemptionPolicy: Never
```

- **Use Case:** Expensive batch jobs or data science models.
- **Behavior:** They go to the front of the line (Priority) but will NOT kill running work (No Preemption). They wait patiently for resources to free up.

Priority vs. Quality of Service (QoS)

PRIORITY

- Governs: Scheduling & Preemption
- The Question: Who gets into the cluster?
- Mechanism: `PriorityClass`

QoS

- Governs: Node Pressure Eviction
- The Question: Who dies when the node runs out of RAM?
- Mechanism: `BestEffort`, `Burstable`, `Guaranteed`

Critical Limitations

PDBs are Best-Effort

`PodDisruptionBudgets` are respected when possible, but will be violated if no other eviction options exist.

No Cross-Node Preemption

Kubernetes won't kill a pod on `Node A` just to satisfy affinity rules for a pod on `Node B`.

Abuse Risk

If everyone is High Priority, no one is. Use `ResourceQuotas` to gatekeep access to `PriorityClasses`.

When to Use It

USE FOR

- API Gateways & Ingress
- Core Backend Services
- Control Plane Add-ons

AVOID FOR

- Dev/Test Workloads
- Random Batch Jobs
- Tasks that can easily retry later

Control Your Chaos

Use Priority to decide what matters most, what can wait, and what can be sacrificed.
Turn a potential outage into a graceful degradation.

vcloudlabs