kubernetes Pod Priority and Preemption

Maha Amer Follow
Feb 19 · 5 min read

> *Pod priority and preemption graduated to beta in Kubernetes 1.11 and to Stable in Kubernetes 1.14. They have been enabled by default since 1.11.*

Pod priority indicates the importance of a pod relative to other pods and queues the pods based on that priority.

Pod preemption allows the cluster to evict, or preempt, lower-priority pods so that higher-priority pods can be scheduled if there is no available space on a suitable node Pod priority also affects the scheduling order of pods and out-of-resource eviction ordering on the node.

Why should I set pod priority and premption?

Let's assume we have Two types of pods pending scheduling. By default (1) The Kubernetes scheduler finds an available worker node with room for all three pods, and schedules them in order of creation.But you want to control which pods are more critical to your cluster workload like (2). So

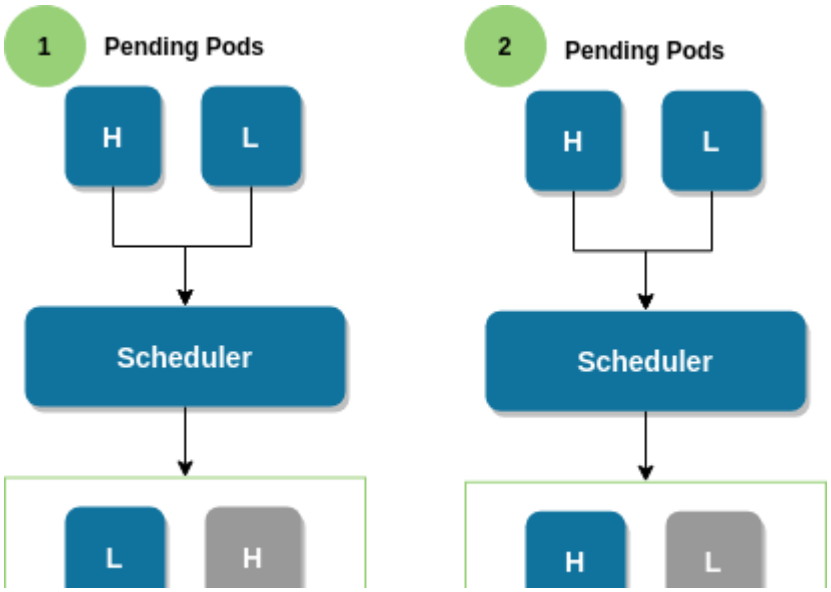


Figure: Pod scheduler scenarios (default vs control)

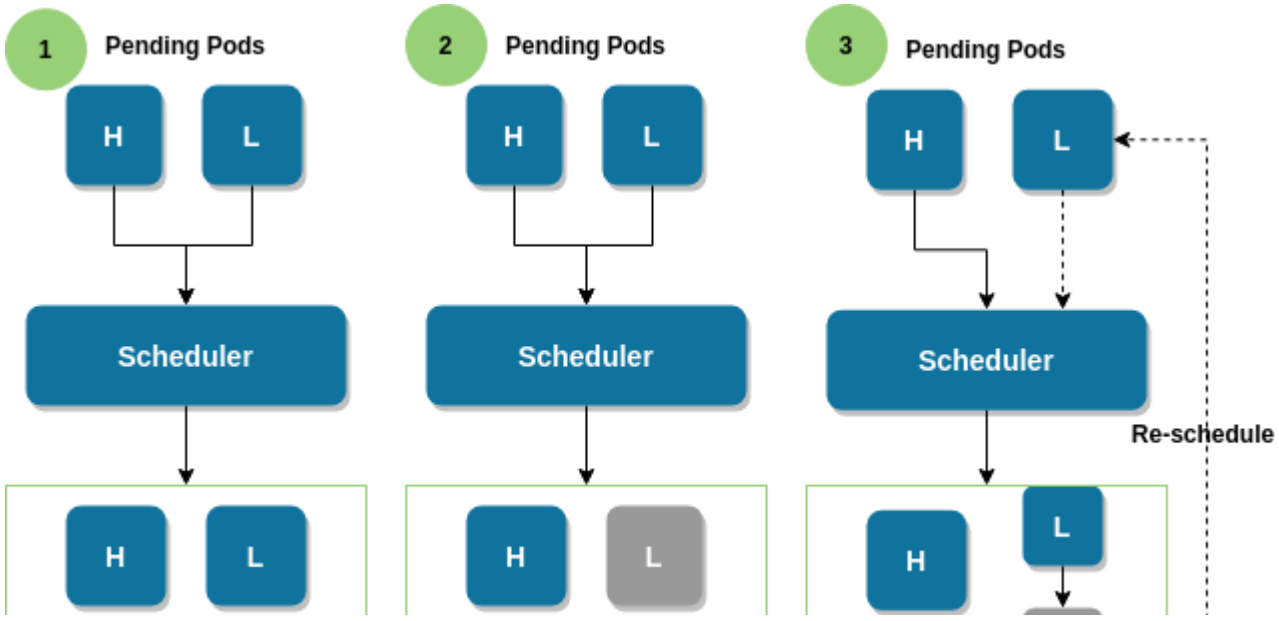
Priority classes can help you control the Kubernetes scheduler decisions to favor higher priority pods over lower priority pods.
The Kubernetes scheduler can even preempt (remove) lower priority pods that are running so that pending higher priority pods can be scheduled.

By setting pod priority, you can help prevent lower priority workloads from impacting critical workloads in your cluster, especially in cases where the cluster starts to reach its resource capacity.

How does priority scheduling and preemption work with Kubernetes scheduler ?

You can assign pods a priority class, which is a non-namespaced object that defines a mapping from a name to the integer value of the priority. The higher the value, the higher the priority.
If you do not specify a priority for your pod deployment, the default is set to the priority class that is set as the `globalDefault` . If you do not have a `globalDefault` priority class, the default priority for all pods is zero (`0`).

Example scenarios :-



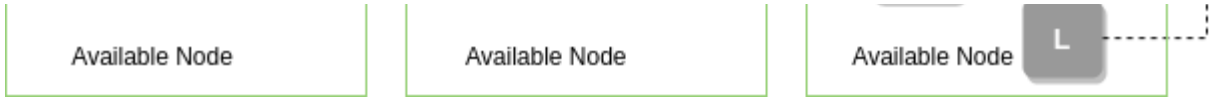


Figure: Pod priority scenarios

1. Two pods with high, and low priority are pending scheduling. The Kubernetes scheduler finds an available worker node with room for all Two pods, and schedules them in order of priority, with the highest priority pod scheduled first.

2. Two pods with high, and low priority are pending scheduling. The Kubernetes scheduler finds an available worker node, but the worker node has only enough resources to support the high and low priority pods. The low-priority pod is not scheduled and it remains in pending.

3. Pods with high is pending scheduling. A anther pod with low priority exists on an available worker node. However, the worker node does not have enough resources to schedule the pending pod. The Kubernetes scheduler preempts, or removes, the low-priority pod, which returns the pod to a pending state. Then, the Kubernetes scheduler tries to schedule the high priority pod.

Default priority classes

A priority class object can take any 32-bit integer value smaller than or equal to 1000000000 (one billion). Reserve numbers larger than one billion for critical pods that should not be preempted or evicted. By default, Kubernetes or OKD has two reserved priority classes for critical system pods to have guaranteed scheduling.

- **System-node-critical** — This priority class has a value of 2000001000 and is used for all pods that should never be evicted from a node. Examples of pods that have this priority class are sdn-ovs, sdn, and so forth.

- **System-cluster-critical** — This priority class has a value of 2000000000 (two billion) and is used with pods that are important for the cluster. Pods with this priority class can be evicted from a node in certain circumstances. For example, pods configured with the `system-node-critical` priority class can take priority. However, this priority class does ensure guaranteed scheduling. Examples of pods that can have this priority class are fluentd, add-on components like descheduler, and so forth.

How to use priority and preemption?

You apply pod priority and preemption by creating a priority class objects and associating pods to the priority using the `priorityClassName` in your pod specifications.

Sample priority class object

```
apiVersion: scheduling.k8s.io/v1beta1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
description: "This priority class should be used for High priority
service pods only."
```

`globalDefault:` This field is `false` by default. Adding a priority class with `globalDefault:true` affects only pods created after the priority class is added and does not change the priorities of existing pods.Optional arbitrary text string that describes which pods developers should use with this priority class.

Sample pod specification with priority class name

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  priorityClassName: high-priority
```

(Notes) if you delete a PriorityClass, existing Pods that use the name of the deleted PriorityClass remain unchanged, but you cannot create more Pods that use the name of the deleted PriorityClass.

How does Pod preemption work with other scheduler settings?

If you enable pod priority and preemption, consider your other scheduler settings:

priority and pod disruption budget

If you specify pod disruption budgets, kubernetes or OKD respects them when preempting pods at a best effort level. The scheduler attempts to preempt pods without violating the pod disruption budget. If no such pods are found, lower-priority pods might be preempted despite their pod disruption budget requirements.

Pod priority and pod affinity

If a pending pod has inter-pod affinity with one or more of the lower-priority pods on a node, the scheduler cannot preempt the lower-priority pods without violating the affinity requirements. In this case, the scheduler looks for another node to schedule the pending pod. However, there is no guarantee that the scheduler can find an appropriate node and pending pod might not be scheduled.

To prevent this situation, carefully configure pod affinity with equal-priority pods.

How to disable preemption?

> *In Kubernetes 1.11 and later, preemption is controlled by a kube-scheduler flag* `disablePreemption`*, which is set to* `false` *by default. If you want to disable preemption despite the above note, you can set* `disablePreemption` *to* `true`*.*
>
> *This option is available in component configs only and is not available in old-style command line options. Below is a sample component config to disable preemption:*

```
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration
algorithmSource:
  provider: DefaultProvider

...

disablePreemption: true
```

References

[1] Pod Priority and Preemption:
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
[2] IBM cloud
https://cloud.ibm.com/docs/containers?topic=containers-pod_priority

Scheduling Kubernetes Cluster Kubernetes Pod Priority Disruption