76. Priority classes

Dp 10 2:07

⇒ K8s runs different apps as pods with different priorities. For eg:- the K8s control plane components (api server, scheduler, etcd) itself run as pods within the cluster & no matter what those workloads need to always run

⇒ So, we may also run Databases, Critical apps, running on the cluster as well as lower priority workloads like background jobs

⇒ Now, we need a way to make sure that higher priority workloads always got scheduled without being interrupted by lower priority workloads, this is where priority classes come in.

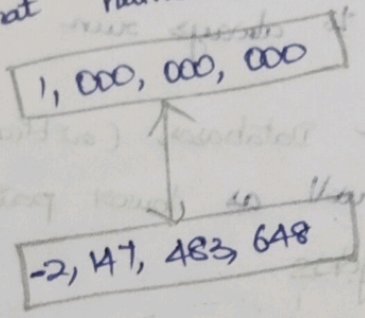
* Priority classes help us define priorities for different workloads, so that higher priority workloads always got a priority over lower priority ones,

⇒ If a higher priority pod cannot be scheduled then, the scheduler will try to eliminate a lower priority workload to make that happen. So priority classes are non-namespace objects, [meaning they're not created within a specific namespace, they're not attached to a specific namespace. So once they're created they're available to be configured on any pod in any namespace

How to define Priorities :-

⇒ We'll define priorities with a range of numbers

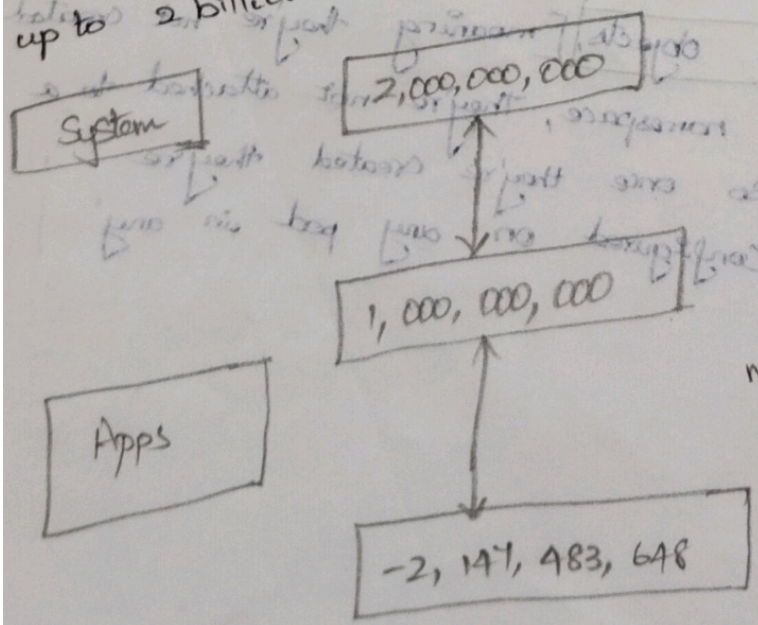
⇒ So a priority can be defined as high as one billion and as low as negative two billion (or) around that numbers



⇒ A larger number indicates higher priorities

⇒ The numbers that are mentioned above are for the apps & workloads that are deployed as apps on the cluster

⇒ But there is a separate range that's dedicated for internal system critical pods, such as the K8s Control plane Components itself, so they always get the highest priority & they're priorities up to 2 billion



List Priority Classes

To list priority class
kubectl get priorityclass

How to create a new priority class

priority-class.yaml

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000000
description: "Priority class for mission critical pods"
```

[The "description" is an "optional field"] Once the "priority-class.yaml" definition file is created, we associate this priority class to a pod by using the "priorityClassName" property, inside the "pod-definition"

Pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 8080
  priorityClassName: high-priority
```

When that pod is created, it is assumed to have this priority that is assigned to the priority class.

What if we dont set a priority class name for a pod?

⇒ By default, it is assumed to have a priority class value of "zero"

⇒ If you would like to modify that, you must create a "new priority class" & assign the global default property to "true", This is an optional field but it can be used to define the default priority of all the pods

⇒ If there're no priority class name explicitly defined in that pod's configuration file

NOTE: Global default property can be defined only through "single priority class" because we cannot have multiple default values

* we cannot have more than one priority class with global default property set to true.

priority-class.yaml

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000000
description: "Priority class for mission critical pods"
globalDefault: true
```

Effect of Pod Priority

⇒ If there are two workloads with two priorities coming into be scheduled

* A critical app with a higher priority of 7
* And a jobs app with a lower priority of 5.

⇒ Since the "critical app" with the "higher priority" of "7" hence it gets placed "first" & if the resource available, so the workload with the lower priority also gets placed next after the higher priority pod gets placed.

⇒ Now, we've a high priority job that comes in with a priority of say 8 & there're no more resources available on the cluster, what happens?

⇒ This behaviour is defined by the "preemption policy" defined in the "priority class" assigned to the new workload.

kill, evict, displace, terminate

priority-class.yaml

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000000
description: "Priority class for mission critical pods"
preemptionPolicy: PreemptLowerPriority  # kills low priority pods
```

⇒ If the preemption policy is not set, its default value is set to "preempt lower priority" This means that it would kill existing lower priority job & take its place

⇒ But if you do not want it to kill (or) evict the existing workload & instead wait for the cluster resources to free up then you must set this "preemption policy to never"

⇒ preemption policy to never makes the pods in that priority class non-preempting meaning that they cannot preempt (will evict) other pods & it will wait to be scheduled in the scheduling queue However, they will get a "higher priority" in scheduling over other lower priority pods that've also waiting to be scheduled

⇒ So the priority still applies only when it is going to be scheduled, but will it preempt / kill / evict the existing workload or not this is defined by "the preemption policy"

you can compare the priority classes on both pods using the following command:-

```
kubectl get pods -o custom-columns="NAME:.metadata.name,
PRIORITY:.spec.priorityClassName
```