81. Configuring profiles

How kubernetes scheduler works

Consider scheduling a pod to one of the 4 nodes which are part of K8S cluster

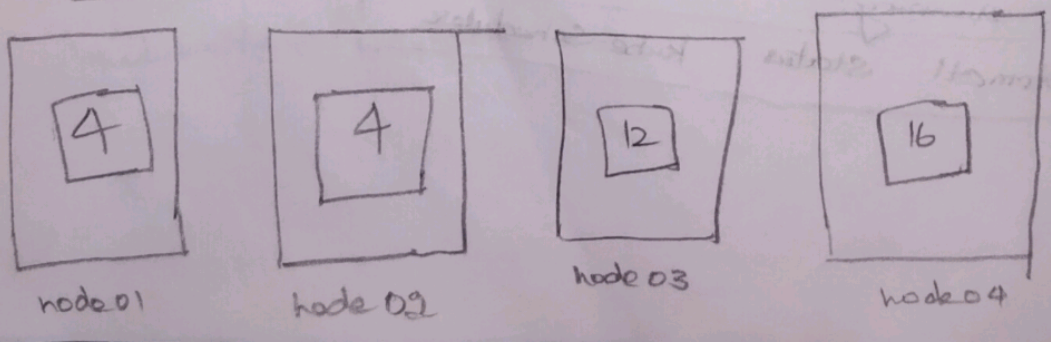
pod-definition.yaml

```
apiVersion : v1
kind: Pod
metadata:
    name : simple-webapp-color
spec:
    priorityClassName: high-priority
    Containers:
    - name: simple-webapp-color
      image: simple-webapp-color
      resources:
```

⇒ Using the above "pod-definition.yaml" a pod got Created waiting to be scheduled in one of the 4 nodes in the cluster

```
        requests:
            memory: "1Gi"
            cpu: 10
```

⇒ It has a resource requirement of 10 CPU, so its gonna be scheduled on a node that has 10 CPU remaining & See the available CPU on all of those nodes



⇒ Now, the pod is not done waiting to be scheduled there're other pods too.

Scheduling queue:-

⇒ So, the first thing that happens is that, when these pods are created, the pods ends up in a "scheduling queue".

⇒ This is where the pods wait to be scheduled. So at this stage, pods are sorted based on the priority defined on the pods. In this case our pod has a high priority set.

To set a priority

⇒ First, create a priority class that looks like this below & set it a "name" & set it a "priority value", in this case, it is set to "1 million"

apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
description: "this priority class should be used for xyz service pods only"

"1 million" is really a very high priority, this is how the pod with very high priority gets to the beginning of the queue to be scheduled first, so that sorting happens in this scheduling phase.

Filtering:-

⇒ Then, our pod enters the "filter" phase

⇒ This is where nodes that cannot run the pod are filtered out

⇒ In our case node01 & node02 do not have sufficient resources [do not have 10CPU remaining] so they are filtered out

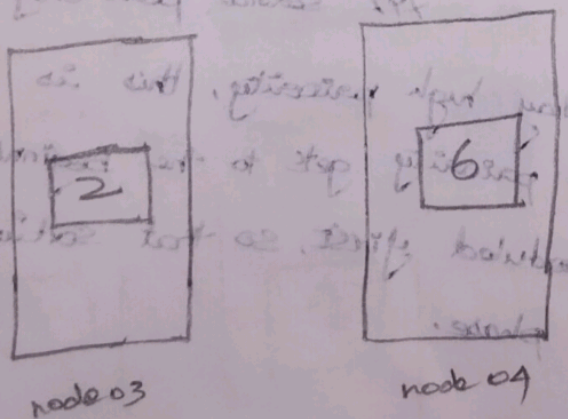
The next phase is Scoring phase.

Scoring phase:-

⇒ In this phase, nodes are scored with different weights

⇒ From the 2 remaining nodes, node03 and node04, the "scheduler" associates a score to each node based on the free space that it will have after reserving the CPU required for that pod.

* Currently node03 has CPU limit of 12, as the pod definition file it requires 10 CPU limit, so after reserving the cpu limit 10, it will have 2 similarly for node04 it will have 6 after reserving the cpu limit 10



* node03 left with 2 whereas node04 left with 6

⇒ Since after reserving the pod to that node node 04 has maximum resources left, hence the "node 04" gets the highest score

⇒ Hence "node 04" gets picked up

Binding:-

⇒ And finally in Binding phase, the pod is finally bound to a node with the highest score.

⇒ Now, all of the below operations are achieved via certain plugins

Scheduling Queue → Filtering → Scoring → Binding

Eg:- while in "scheduling phase" (ie) scheduling queue its the "priority Sort" plugin that sorts the pods in an order based on the priority configured on the pods, this is how pods with priority class gets higher priority over the other pods when scheduling

In the Filtering stage, its the NodeResourcesFit plugin that identifies the nodes that has "sufficient resources" required by the pods & filters out the nodes that doesn't

⇒ Other plugin in "filtering stage" - NodeName plugin that checks if a pod has a node name mentioned in the pods spec. & filters out all the nodes that does not match this name.

pod.definition.yaml

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

nodeName: node02

Node Unschedulable plugin:-

kubectl describe node controlplane

```
Name:               controlplane
Roles:              control-plane
CreationTimeStamp:  Thu, 06 oct 2020 06:19:57 -400
Taints:             node.kubernetes.io/unschedulable:NoSchedule
Unschedulable:      true
Lease:
```

⇒ This plugin filters out nodes that has the unschedulable flag set to "true"

⇒ For the nodes, which has this "unschedulable flag" is set to "true", this particular

plugin : Node Unschedulable" makes sure that no pods are set on those nodes

Scoring phase - NodeResourcesFit

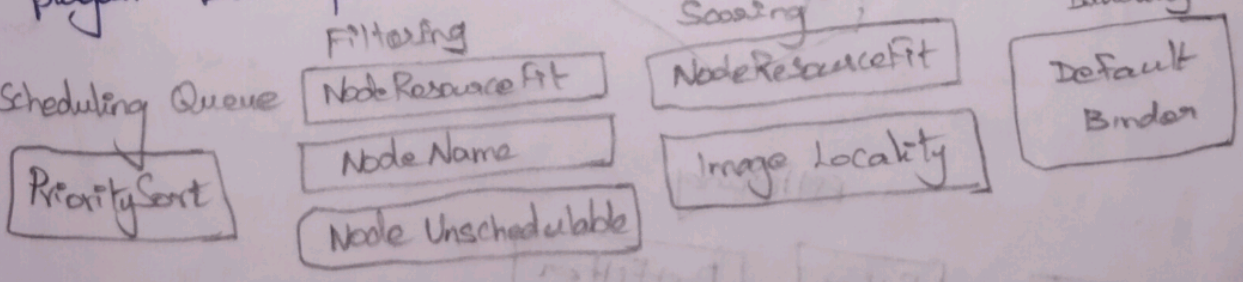
⇒ In the scoring phase, the "node resource fit" plugin associates a score to each node based on the resource available on it & after the pod is allocated to it.

⇒ So, as you can see, a single plugin can be associated in multiple different phases

⇒ Another eg of plugin in this stage would be the image locality plugin that associates a high score to the nodes that already has the container image used by the pods among the different nodes at this place, plugins do not really reject the pod placement on a particular node, for eg:- in case of image locality node, it ensures that pod are placed on a node that already has the image but if there are no nodes available, it'll anyway place the pod on a node that does not even have the image. So its just that scoring that happens at this stage.

Binding phase:-

⇒ In binding phase, you have the default binder plugin that provides the binding mechanism



⇒ The K8s is highly extensible. In nature makes it possible for us to customize what plugin go where

& for us to write our own plugin and plug them in here, that is achieved with the help of "extension points"

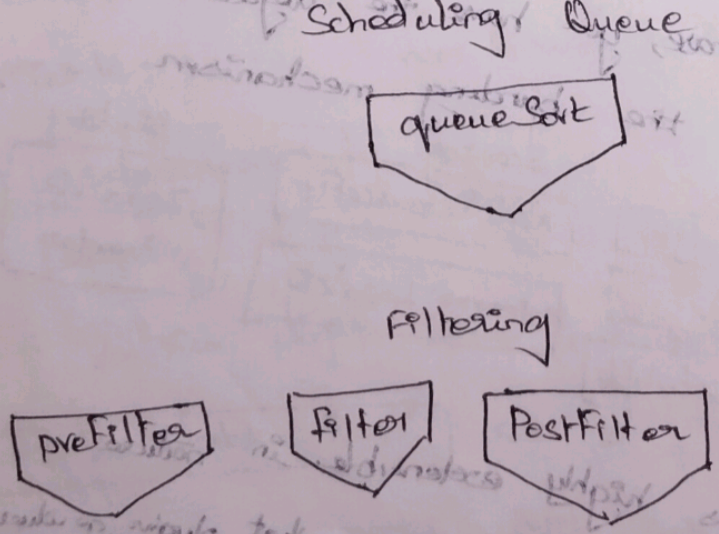
⇒ At each stage, there is an extension point to which a plugin can be plugged to.

⇒ In the scheduling queue, we've a queue sort extension to which the priority sort plugin is plugged to

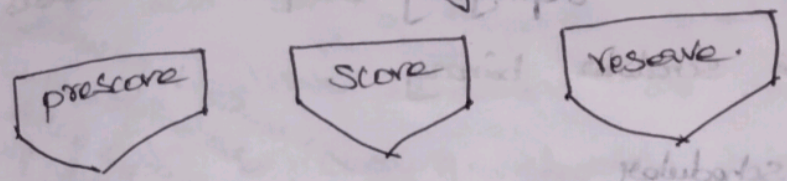
⇒ And then we have the filter extension, the score & the bind extension to which each of these plugins that we just talked about are plugged to

⇒ There're extensions before entering the filter phase, called "pre-filter extension" & after the filter phase called post filter & then there are pre-score before the score extension point & reserve after the extension point the score extension point

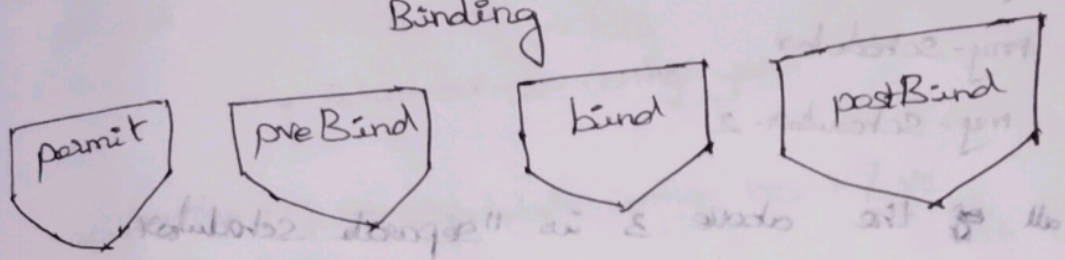
⇒ And there is permit & preBind before the bind & post bind after the "binding phase"



Scoring



Binding



⇒ Basically we can get a custom code of your own to run anywhere in those points by just creating a plugin & plugging it into the respective kind of point that you want to plug it to.

There are additional plugins that come by default that are associated with the different extension points

* Some of the plugins span across multiple extension points

* Some of them are just within a specific extension point

Lets look at how we can change the default behaviour of how those plugins are called & how we can get our own plugins in there if its really needed.

⇒ Earlier, we talked about deploying 3 separate schedulers each with separate scheduler binary

⇒ We've 3 schedulers

1) Default Scheduler
2) my-scheduler
3) my-scheduler-2

kubeadm - pod - 1. config & API server location
2. kubeconfig - location of binary & custom scheduler config file

Now all of the above 3 is "separate scheduler binaries" that run with "separate scheduler config file" associated with each of them. This is one way of deploying "multiple schedulers"

The problem here is "since these are separate processes there is an additional effort required to maintain these separate processes, & more importantly, since they are separate processes they may run into race conditions while making scheduling decision

Eg:- One scheduler may schedule a workload on a node without knowing that there's another scheduler scheduling a workload on that same node at the same time

⇒ So with the 1.18 release of k8s a feature to support multiple profiles in a single scheduler was introduced, so now you can configure multiple profiles within a single scheduler in the configuration file by adding more entries to the list of profiles & for each profile, specify a separate scheduler name.

So this creates a separate profile for each scheduler which acts as a "separate scheduler" itself except that now multiple schedulers are run in the same binary as opposed to creating separate binaries for each scheduler.

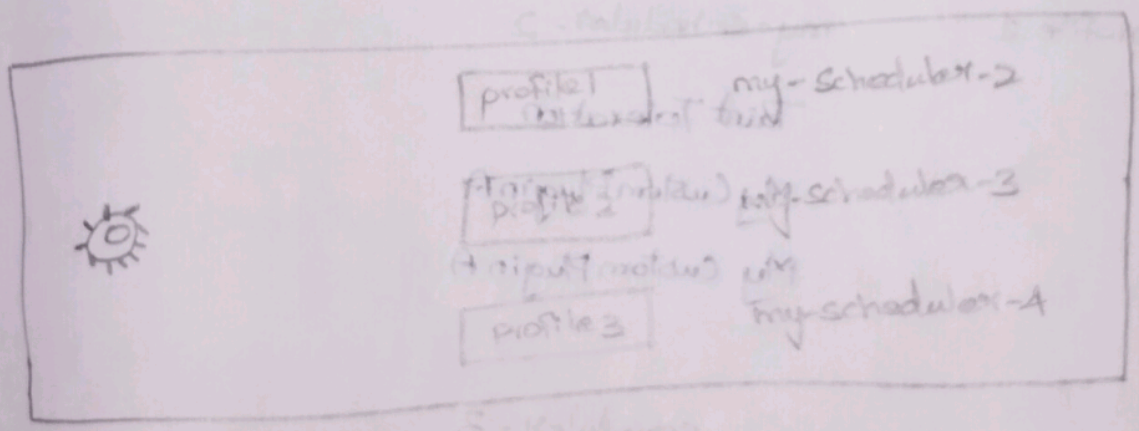
my-scheduler-2-config.yaml

apiVersion: kubescheduler.config.k8s.io/v1

kind: KubeSchedulerConfiguration

profiles:

- schedulerName: my-scheduler-2
- schedulerName: my-scheduler-3
- schedulerName: my-scheduler-4



How to configure these different scheduler profile. To work differently, because right now all of them has different names. So they are gonna work just like the "default scheduler"

How do you configure them to work differently?

Under each scheduler profile we can configure the plugins the way we want to

For eg:- for the myScheduler2 profile, we're going to disable certain plugins like the taint & toleration plugin & enable my own custom plugin

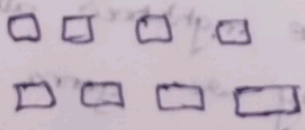
⇒ For the my scheduler 3 profile, we're going to disable all the "pre score" and "score" plugins. so, this is how it is going to look, under the plug-in section, specify the extension point and enable or disable the plug-ins by name or a pattern as shown

profile 1 — my-scheduler-2

- Taint Toleration
- MyCustomPluginA
- MyCustomPluginA

profile 2 — my-scheduler-3

preScore score



profile 3 — my-scheduler-4

my-scheduler-2-config.yaml

```
apiVersion: kubeScheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration

profiles:
- schedulerName: my-scheduler-2
  plugins:
    score:
      disabled:
        - name: TaintToleration
      enabled:
        - name: MyCustomPluginA
        - name: MyCustomPluginB

- schedulerName: my-scheduler-3
  plugins:
    preScore:
      disabled:
        - name: '*'
    score:
      disabled:
        - name: '*'

- schedulerName: my-scheduler-4
```

To read more about this checkout the k8s enhancement proposal that introduced multi scheduling profiles - 1451-multi-scheduling-profiles