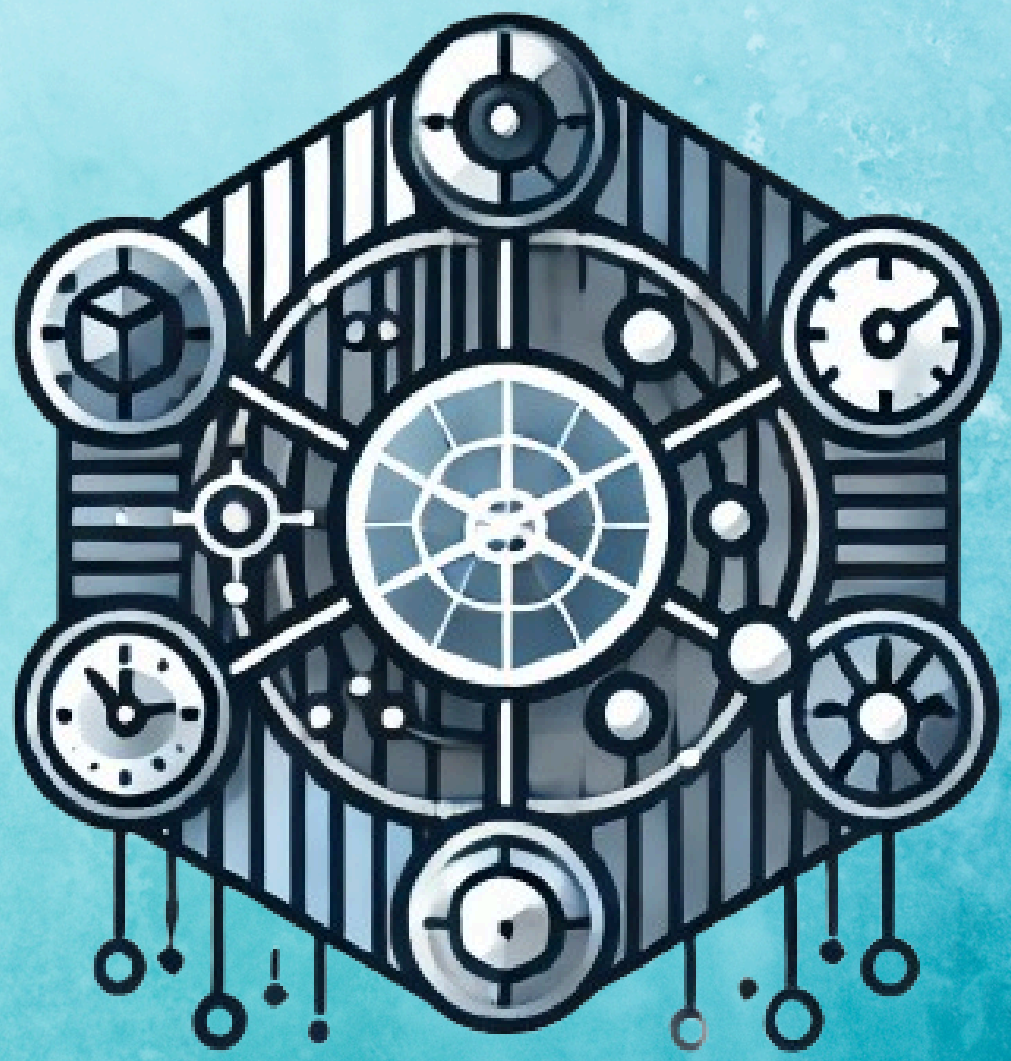
KUBERNETES

Scheduling & Resources



Manual Scheduling

- By default, nodeName is not set. Kubernetes adds it automatically.
- The scheduler goes through all the PODs and looks for those that don't have this set.
  - Identify the Node.
  - Schedule the POD on the Node.
  - Create 'Bind Object': bind the POD to the Node.
- The PODs will stay in 'Pending' status if there is no scheduler.
- Create a binding object and send a post request to the POD binging API that mimics the actual scheduler.

```yaml
apiVersion: v1
kind: Binding
metadata:
  name: nginx
target:
  apiVersion: v1
  kind: Node
  name: <Node-Name>
```

Labels and Selectors

- Labels: properties attached to each item.
- Selectors: helps to filter the labels.

How?

- In the POD definition file

- You can search from label:

$ kubectl <get-pods> --selector <lable-type>=<label>

```yaml
apiVersion: v1 # String
kind: Pod # String
metadata: # Dictionary
  name: my-app
  labels: # Dictionary
    app: my-app
    type: front-end
```

Taints and Tolerations

- Taints and Tolerations are used to set restrictions on what PODs can be scheduled on a Node.

- To taint a Node:
  $ kubectl taint nodes <node-name> <key>=<value>: <taint-effect>

3 taint effects:

- NoSchedule
- ReferNoSchedule
- NoExecute

- See all taints Nodes:
  $ kubectl describe node kubemaster | grep Taint

NodeSelector

- If we want a small a small POD will go to a small Node and a big POD will go to a big Node.
- We label them in the POD definition file with NodeSelector

```yaml
apiVersion: v1 # String
kind: Pod # String
metadata: # Dictionary
  name: my-app
  labels: # Dictionary
    app: my-app
    type: front-end
spec:
  containers: # List of Dictionaries
    - name: my-app-container # String
      image: myregistry.com/my-app:1.0 # String

  # -------------- NODE SELECTOR -------------- #
  nodeSelector: # List of
    size: Large # String
```

NodeAffinity

- Provide us with advanced capabilities to limit POD placement on specific Nodes.
- Makes sure PODs are hosted on a Node.

```yaml
apiVersion: v1 # String
kind: Pod # String
metadata: # Dictionary
  name: my-app
  labels: # Dictionary
    app: my-app
    type: front-end
spec:
  containers: # List of Dictionaries
    - name: my-app-container # String
      image: myregistry.com/my-app:1.0 # String
  # ------------------- NODE AFFINITY -------------- #
  affinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelector: # Dictionary
        - matchExpressions:
            - key: size
              operator: In
              values:
                - Large
```

2 Types of NodeAffinity

1. Available:
- requirdDuringSchedulingIdnoredDuringExecution
- preferredDuringSchedulingIgnoredDuringExecution
2. Planned:
- requiredDuringSchedulingRequiredDuringExecution

NodeAffinity vs Taints & Tolerations

- Each alone can’t guarantee clean POD to Node.
- Both together can make sure clean POD to Node

Resource Requirement and Limits

- ‘kube-scheduler’ decides what POD goes to each Node.
- If there is no available resources, the POD will be in ‘Pending’ state.
- You can specify the amount of resources in the POD-Definition.yaml file

```yaml
apiVersion: v1 # String
kind: Pod # String
metadata: # Dictionary
  name: my-app
  labels: # Dictionary
    app: my-app
    type: front-end
spec:
  containers: # List of Dictionaries
    - name: my-app-container # String
      image: myregistry.com/my-app:1.0 # String
      # ------------------- RESOURCES -------------- #
      resources:
        requests:
          memory: "1Gi"
          cpu: 1
        limits:
          memory: "2Gi"
          cpu: 2
```

Limit Ranges

- Can help to define default values to be set for containers in PODs that are created without a request or limit specified in the 'POD-Definition.yaml'

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-resource-constraint
spec:
  limits:
    - default:
        cpu: 500m # Limit
      defaultRequest:
        cpu: 500m # Request
      max:
        cpu: "1" # Limit
      min:
        cpu: 100m # Request
      type: Container

```

DaemonSets

- Like ReplicaSets, DaemonSets helps you to deploy multiple instances of PODs.
    - Runs each copy of a POD on each Node
- Whenever a new Node is added to the cluster, a replica of the POD is auto-added to the Node. When a Node is removed, the POD is removed as well.
- DaemonSets makes sure the POD is in each Node in the Cluster.

Uses:

- Monitor agent
- Log collector
- Networking

DaemonSets-definition.yaml

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: monitoring-Daemon
spec:
  selector:
    matchLabels:
      app: monitoring-agent
  template:
    metadata:
      labels:
        app: monitoring-agent
    spec:
      containers:
        - name: monitoring-agent
          image: <your-docker-image-registry>/<your-docker-image-name>:latest

```

Static PODs

- If there is no: master, cluster, or even Nodes can the POD stand alone?
- Without kube-apiserver, the kublet is configured to read the POD.yaml files from a directory in a server that stores all the information about the PODs.
- The kublet checks the directory for changes: edit/add/delete.

Change the directory the kublet checks:

- In kublet.service:
  --pod-manifest-path=/path/to/dir \\
- In kubeconfig.yaml:
  staticPodPath:/path/to/dir
- The API server can create both static PODs and Cluster Nodes and can notice the changes in the static Nodes.

Multiple Schedulers

- We can make a custom scheduler and also make it the default scheduler.

```yaml
apiVersion: kubernetes.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
  - schedulerName: Custom-Scheduler
```

- Also, we can add an additional scheduler by editing ‘my-scheduler.service’

--config-/etc/kubernetes/config/<my-scheduler.yaml>

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-custom-scheduler
  namespace: kube-system
spec:
  containers:
   command:
   - kube-scheduler
   - --address=127.0.0.1
   - --kubeconfig=/path/to/my-custom-scheduler
   - --config=/path/to/my-custom-scheduler.yaml

  image: k8s.gcr.io/kube-scheduler-amd64:v1.11.3
  name: kube-scheduler
```

Schedule Profiles

- Schedule Queue: The place where the PODs are waiting to be created.
  - PODs are stored based on priority.
- Filter: if there are any resources to use
- Scoring: Nodes are scored in different weights- based on a free space.
- Binding: the prosses of a POD bind to a Node.

- Also, we can add an additional scheduler by editing ‘my-scheduler.service’

Scheduling Plugins

Scheduling Queue:

- Queuesort
- PrioritySort

Filtering:

- NodeResourcesFit
- NodeName
- NodeUnschedulable
- NodeResourcesFit
- TaintToleration
- NodePorts
- NodeAffinity

Scoring:

- NodeResourceFit
- ImageLocality

Binding:

DefaultBinder