Kubernetes Scheduler

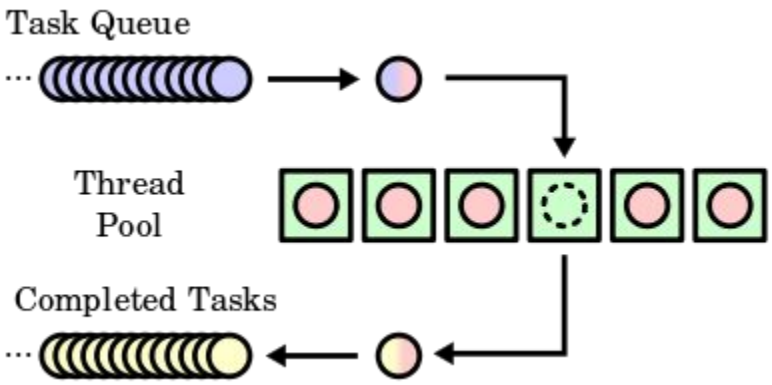
Kanagaraj Manickam

Scheduling

In computing, **scheduling** is the action of assigning *resources* to perform *tasks*. The *resources* may be processors, network links or expansion cards. The *tasks* may be threads, processes or data flows.

The scheduling activity is carried out by a process called **scheduler**. Schedulers are often designed so as to keep all computer resources busy (as in load balancing), allow multiple users to share system resources effectively, or to achieve a target quality-of-service.

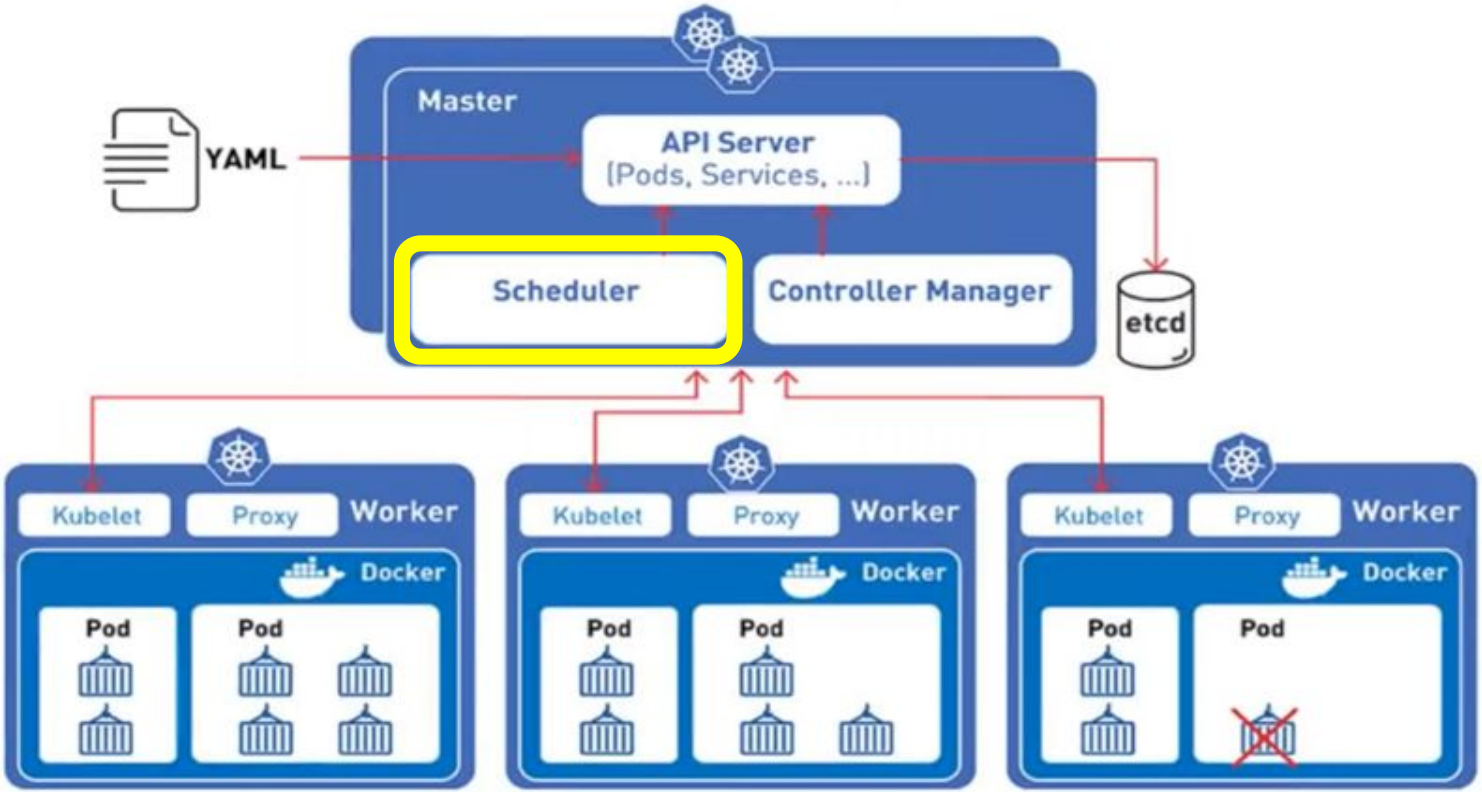
Scheduling is fundamental to computation itself, and an intrinsic part of the execution model of a computer system; the concept of scheduling makes it possible to have computer multitasking with a single central processing unit (CPU).



Agenda

- Architecture
- Resource Request and Limits
- Manual Scheduling
- Node Selector & Affinity
- Pod Affinity
- Taint & Tolerations
- DemonSet
- Static Pod
- Multiple scheduler

Kubernetes Architecture



Resource Request & Limits

- Sets boundary for resource consumption of containers in a pod
- CPU, Memory & hugepage
- > Limit => OOM, so evict the running pod
-

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
spec:
  limits:
  - default:
      cpu: 1
    defaultRequest:
      cpu: 0.5
    type: Container
```

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: mem-limit-range
spec:
  limits:
  - default:
      memory: 512Mi
    defaultRequest:
      memory: 256Mi
    type: Container
```

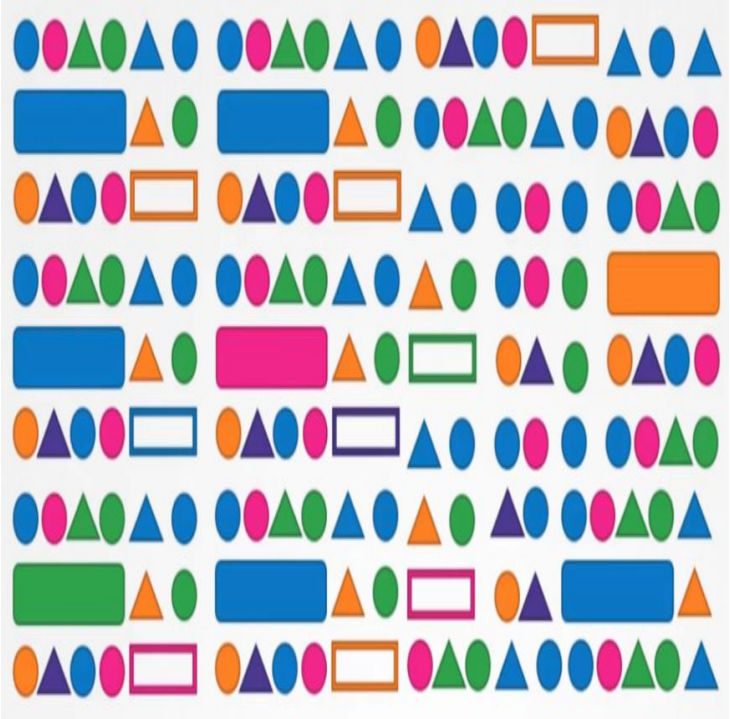
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    resources:
      requests:
        memory: "1Gi"
        cpu: 1
      limits:
        memory: "2Gi"
        cpu: 2
```

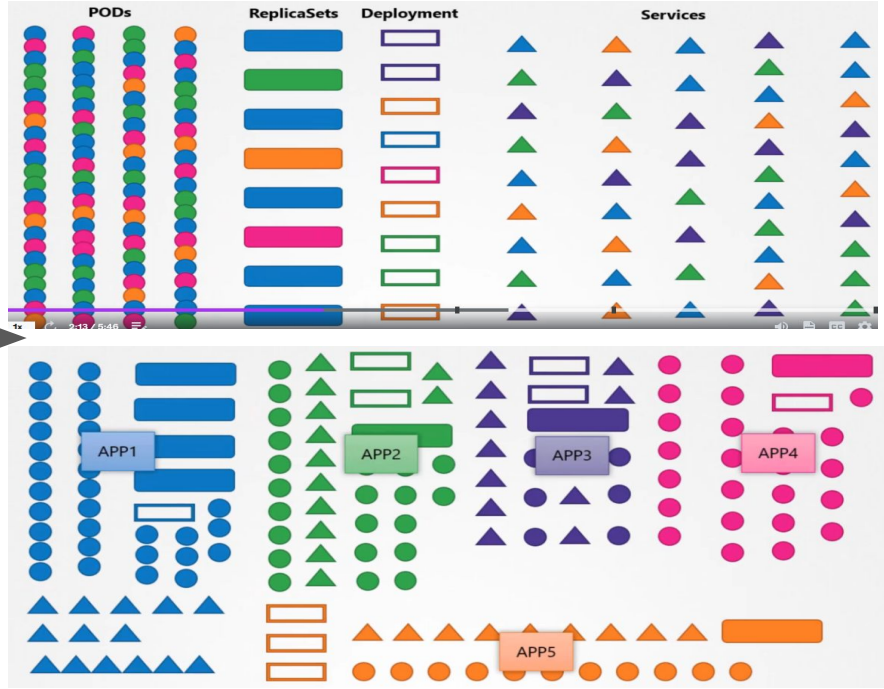
Manual Scheduling

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 8080
  nodeName: node02
```

- Direct placement of pod
- Ex: CI/CD
- Pod State: Pending -> Ready

Label & Selector





Label & Selector

Labels

- Tags
- Categorize
- Dynamic

Selectors

- Choose Objects based on concern
- Grouping

```
controlplane ~ ➜  kubectl label pod auth a=b
pod/auth labeled

controlplane ~ ➜  kubectl get pod -l a=b
NAME    READY   STATUS    RESTARTS   AGE
auth    0/1     Pending   0          109s
```

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: simple-webapp
  labels:
    app: App1
    function: Front-end
spec:
  replicas: 3
  selector:
    matchLabels:
      app: App1
  template:
    metadata:
      labels:
        app: App1
        function: Front-end
    spec:
      containers:
      - name: simple-webapp
        image: simple-webapp
```

Node Selector

- Direct approach to associate node of having given label
- Very limited filtering ability

  Kubectl label nodes node1 disktype=ssd

- Explicitly specify the labels to select in the pod definition

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
```

Node Affinity

- More **expressive** than simple node selector
- Taken care of pod while there is change in env
-

	DuringScheduling	DuringExecution
Type 1	Required	Ignored
Type 2	Preferred	Ignored
Type 3	Required	Required

```yaml
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/os
          operator: In
          values:
          - linux
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 1
      preference:
        matchExpressions:
        - key: another-node-label-key
          operator: In
          values:
          - another-node-label-value
```

||

&&

```yaml
apiVersion:
kind:
metadata:
 name: myapp-pod
spec:
 containers:
  - name: data-processor
    image: data-processor
 affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: size
          operator: In
          values:
          - Large
```

`In`, `NotIn`, `Exists`, `DoesNotExist`, `Gt` and `Lt`.

Pod Affinity/Anti-affinity

Inter-pod affinity and anti-affinity rules take the form "this Pod should (or, in the case of anti-affinity, should not) run in an X if that X is already running one or more Pods that meet rule Y", where X is a topology domain like node, rack, cloud provider zone or region, or similar and Y is the rule Kubernetes tries to satisfy

X - Label Selector

Y - TopologyKey

- `requiredDuringSchedulingIgnoredDuringExecution`
- `preferredDuringSchedulingIgnoredDuringExecution`

node-1	node-2	node-3
webserver-1	*webserver-2*	*webserver-3*
cache-1	*cache-2*	*cache-3*

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-server
spec:
  selector:
    matchLabels:
      app: web-store
  replicas: 3
  template:
    metadata:
      labels:
        app: web-store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - web-store
            topologyKey: "kubernetes.io/hostname"
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
```

Taint (Node) & Tolerations (Pod)

Taints allow a node to repel a set of pods unless they are toleran

Tolerations allow the scheduler to schedule pods with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes.

```
kubectl taint nodes node-name key=value:taint-effect
```

What happens to PODs that DO NOT TOLERATE this taint?

NoSchedule | PreferNoSchedule | NoExecute

1

```
kubectl taint nodes node1 example-key=value1:NoSchedule
```

2

```yaml
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  tolerations:
  - key: "example-key"
    operator: "Exists"
    effect: "NoSchedule"
```

DemonSet

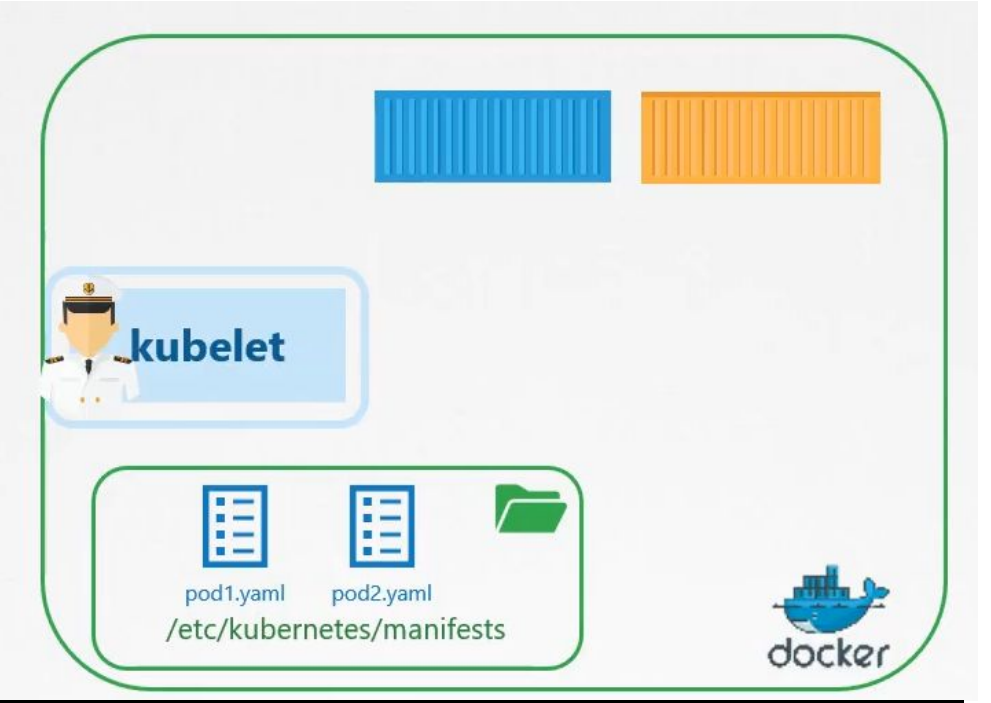
- Per Node, only one instance of Pod
- Monitoring, logging kind of use cases
- Same as ReplicaSet definition YAML.
- Ex: kube-proxy, wavenet
- Alternative : Static pod / direct pod ??

```yaml
spec:
  tolerations:
  # these tolerations are to have the daemonset runnable on control plane nodes
  # remove them if your control plane nodes should not run pods
  - key: node-role.kubernetes.io/control-plane
    operator: Exists
    effect: NoSchedule
  - key: node-role.kubernetes.io/master
    operator: Exists
    effect: NoSchedule
```

```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: monitoring-agent
```

Static pod

- Deployment of multi-node masters
- Kubelet creates and manage Static pods
- No control by master controllers such as ReplicaSet
- No control by scheduler




```
ExecStart=/usr/local/bin/kubelet \\
  --container-runtime=remote \\
  --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \\
  --pod-manifest-path=/etc/Kubernetes/manifests \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2
```

Multiple Scheduler

- Default-scheduler
- Custom specific scheduling
- Bound pod to specific scheduler

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  -  image: nginx
     name: nginx

   schedulerName: my-custom-scheduler
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-custom-scheduler
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-scheduler
    - --address=127.0.0.1
    - --kubeconfig=/etc/kubernetes/scheduler.conf
    - --leader-elect=true
    - --scheduler-name=my-custom-scheduler

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

Thank you