Managing and Controlling the Kubernetes Scheduler

Kien Bui
DevOps & Platform Engineer

Course Overview

Configuring and Managing Storage in Kubernetes

Configuration as Data - Environment Variables, Secrets and ConfigMaps

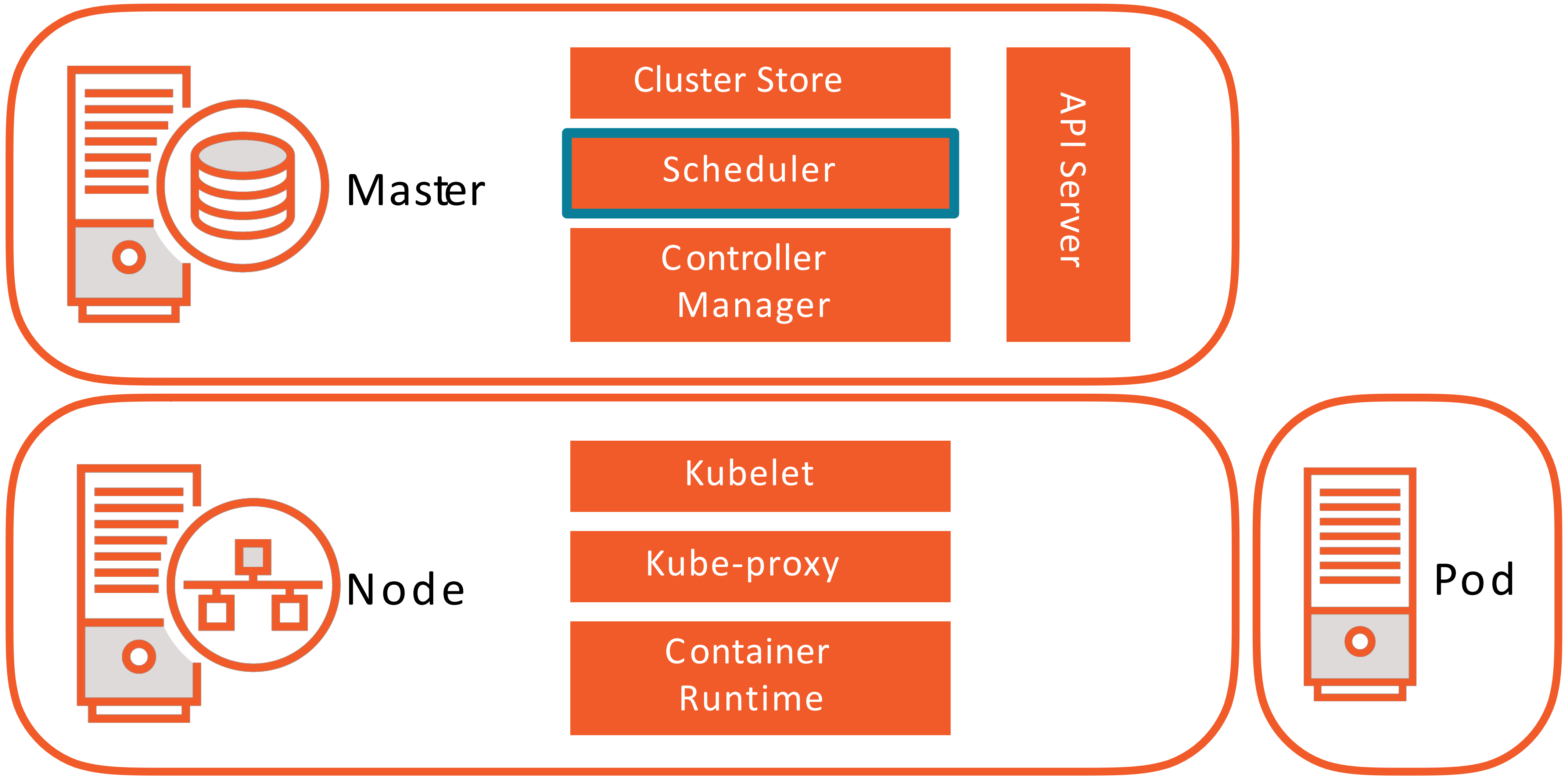
Managing and Controlling the Kubernetes Scheduler

Overview

Scheduling in Kubernetes

Controlling scheduling in Kubernetes

Master Node - Control Plane



Kubernetes has one job…
starting Pods on Nodes

Scheduling in Kubernetes

Selecting a Node to start a Pod on

`kube-scheduler`

Scheduling in Kubernetes

Resources

Policy

Scheduling Process

Watches the API Server for Unscheduled Pods

Node selection

Update `nodeName` in the Pod object

Nodes' kubelets watch API Server for work

Signal container runtime to start container(s)

Node Selection

Filtering

- From all Nodes
- Apply Filters
- Filtered Nodes
- Hard constraints

Scoring

- Scoring functions
- Feasible Nodes
- Policy constraints

Binding

- Selected Nodes List
- Ties are broken
- Update API Object

`nodeName: c1-node1`

Pod

Resource Requests

Setting `requests` will cause the scheduler to find a Node to fit the workload/Pod

- `requests` are guarantees
  - CPU
  - Memory
- Allocatable resources per Node
- Pods that need to be scheduled but there not enough resources available will go `Pending`

Demo

Scheduling in action

Scheduling Pods with `requests`

Controlling Scheduling

Node Selector

Affinity

Taint and Tolerations

Node Cordoning

Manual Scheduling

Node Selector

`nodeSelector` -assign Pods to Nodes using Labels and Selectors

Apply Labels to Nodes

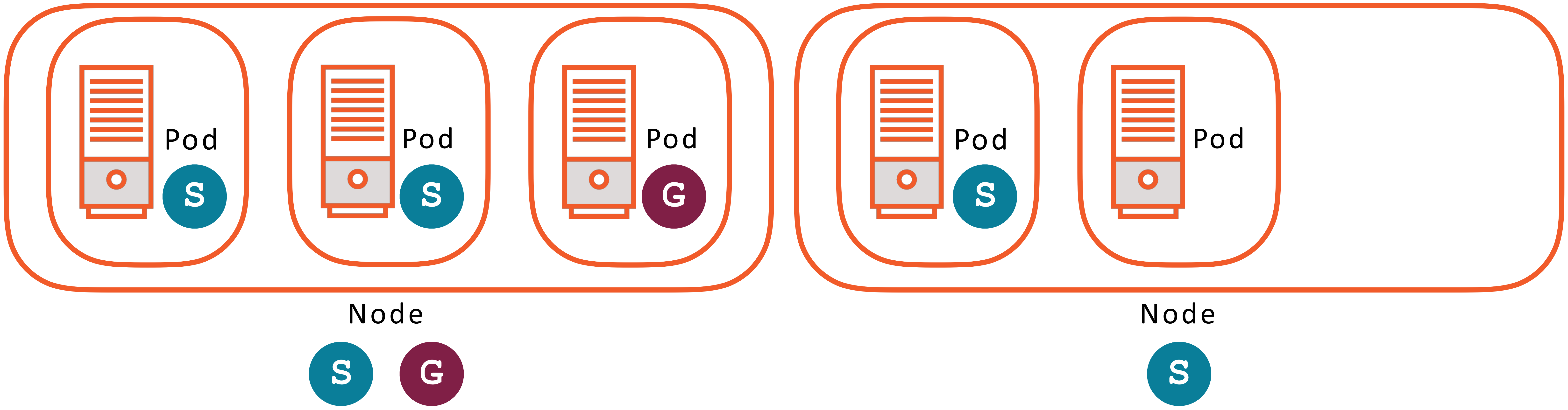
Scheduler will assign Pods a to a Node with a matching Label

Simple key/value check based on `matchLabels`

Often used to map Pods to Nodes based on...

- Special hardware requirements
- Workload isolation

Scheduling - Node Selector



Assigning Pods to Nodes using Node Selectors

```
kubectl label node c1-node3 hardware=local_gpu
```

```
spec:
  containers:
  - name: hello-world
    image: gcr.io/google-samples/hello-app:1.0
    ports:
    - containerPort: 8080
  nodeSelector:
    hardware: local_gpu
```

Affinity and Anti-Affinity

`nodeAffinity`-uses Labels on Nodes to make a scheduling decision with `matchExpressions`

`requiredDuringSchedulingIgnoredDuringExecution`

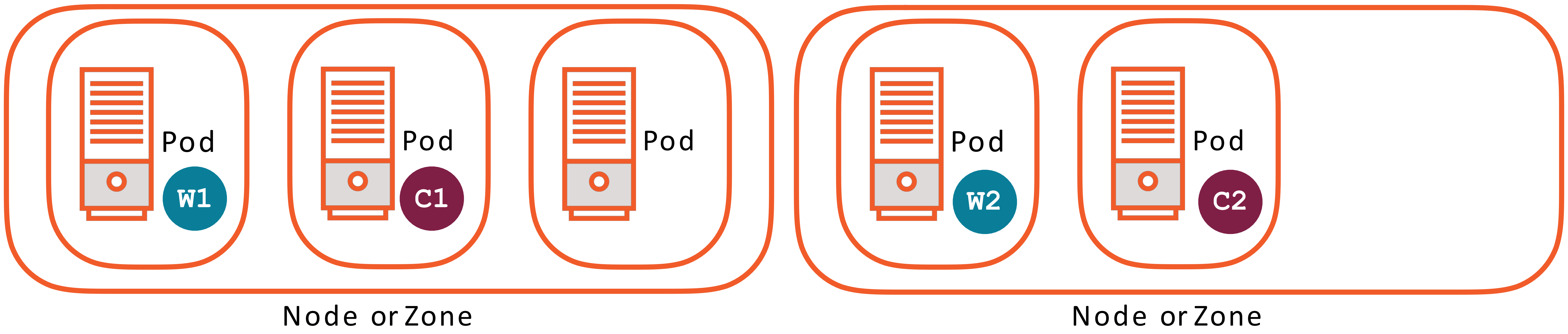
`preferredDuringSchedulingIgnoredDuringExecution`

`podAffinity` -schedule Pods onto the same Node, Zone as some other Pod

`podAntiAffinity` - schedule Pods onto the different Node, Zone as some other Pod

https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity

Scheduling - Pod Affinity/Anti-Affinity



Using Affinity to Control Pod Placement

```
spec:
  containers:
  - name: hello-world-cache
  ...
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - hello-world-web
        topologyKey: "kubernetes.io/hostname"
```

Taints and Tolerations

Taints - ability to control which Pods are scheduled to Nodes

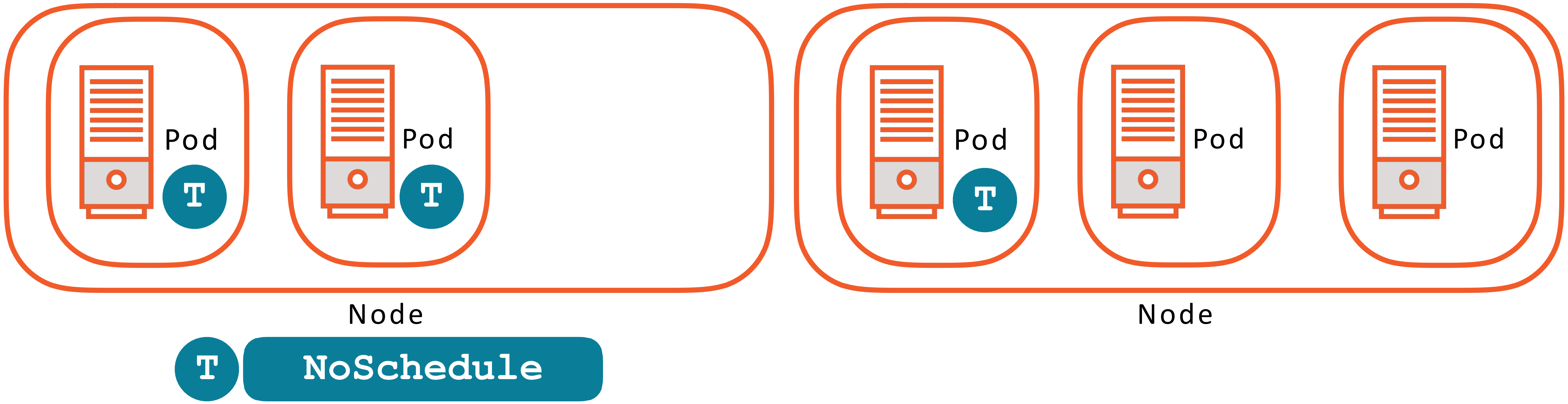
Tolerations - allows a Pod to ignore a Taint and be scheduled as normal on Tainted Nodes

Useful in scenarios where the cluster administrator needs to influence scheduling without depending on the user

```
key=value:effect
```

```
kubectl taint nodes c1-node1 \
 key=MyTaint:NoSchedule
```

Scheduling - Taints and Tolerations



Adding a Taint to a Nodes and a Toleration to a Pod

```
kubectl taint nodes c1-node1 key=MyTaint:NoSchedule
```

```
spec:
  containers:
  - name: hello-world
    image: gcr.io/google-samples/hello-app:1.0
    ports:
    - containerPort: 8080
  tolerations:
  - key: "key"
    operator: "Equal"
    value: "MyTaint"
    effect: "NoSchedule"
```

Demo

Using Affinity and Anti-Affinity to schedule Pods to Nodes

Controlling Pod placement with Taints and Tolerations

Node Cordoning

Marks a Node as unschedulable

Prevents new Pods from being scheduled to that Node

Does not affect any existing Pods on the Node

This is useful as a preparatory step before a Node reboot or maintenance

```
kubectl cordon c1-node3
```

If you want to gracefully evict your Pods from a Node…

```
kubectl drain c1-node3 --ignore-daemonsets
```

Manually Scheduling a Pod

Scheduler populates `nodeName`

If you specify `nodeName` in your Pod definition the Pod will be started on that node

Node's name must exist

Still subject to Node resource constraints

Configuring Multiple Schedulers

Implement your own scheduler

Run multiple schedulers concurrently

- Define in your Pod Spec which scheduler you want

Deploy your scheduler as a system Pod in the cluster

https://kubernetes.io/docs/tasks/administer-cluster/configure-multiple-schedulers/

Demo

Node Cordoning

Manually scheduling a Pod

Review

Scheduling in Kubernetes

Controlling scheduling in Kubernetes