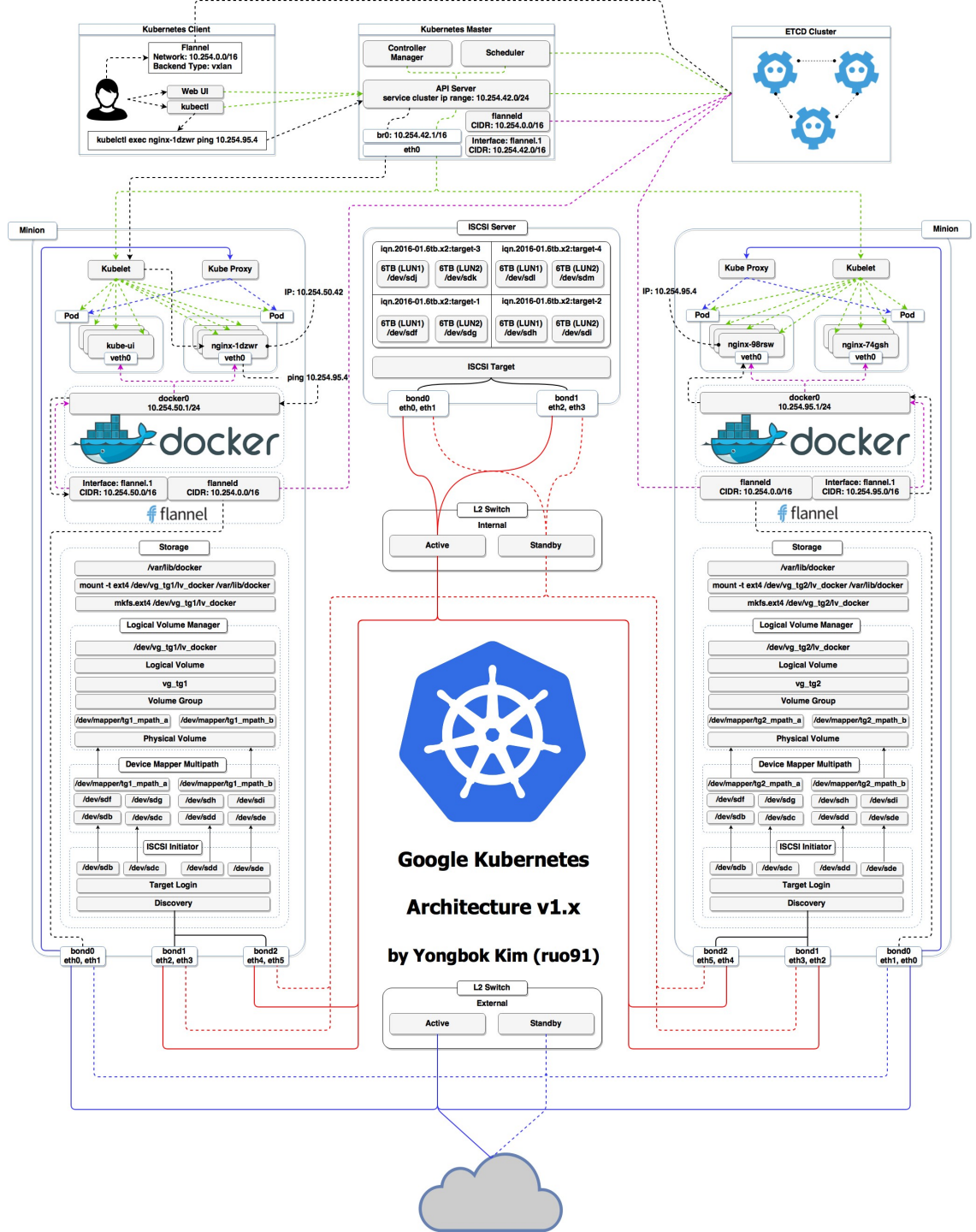

kubernetes

Agenda

- Kubernetes Architecture Review
- Pod Scheduling
- Custom Scheduler
- Kubelet Configuration
- ConfigMap

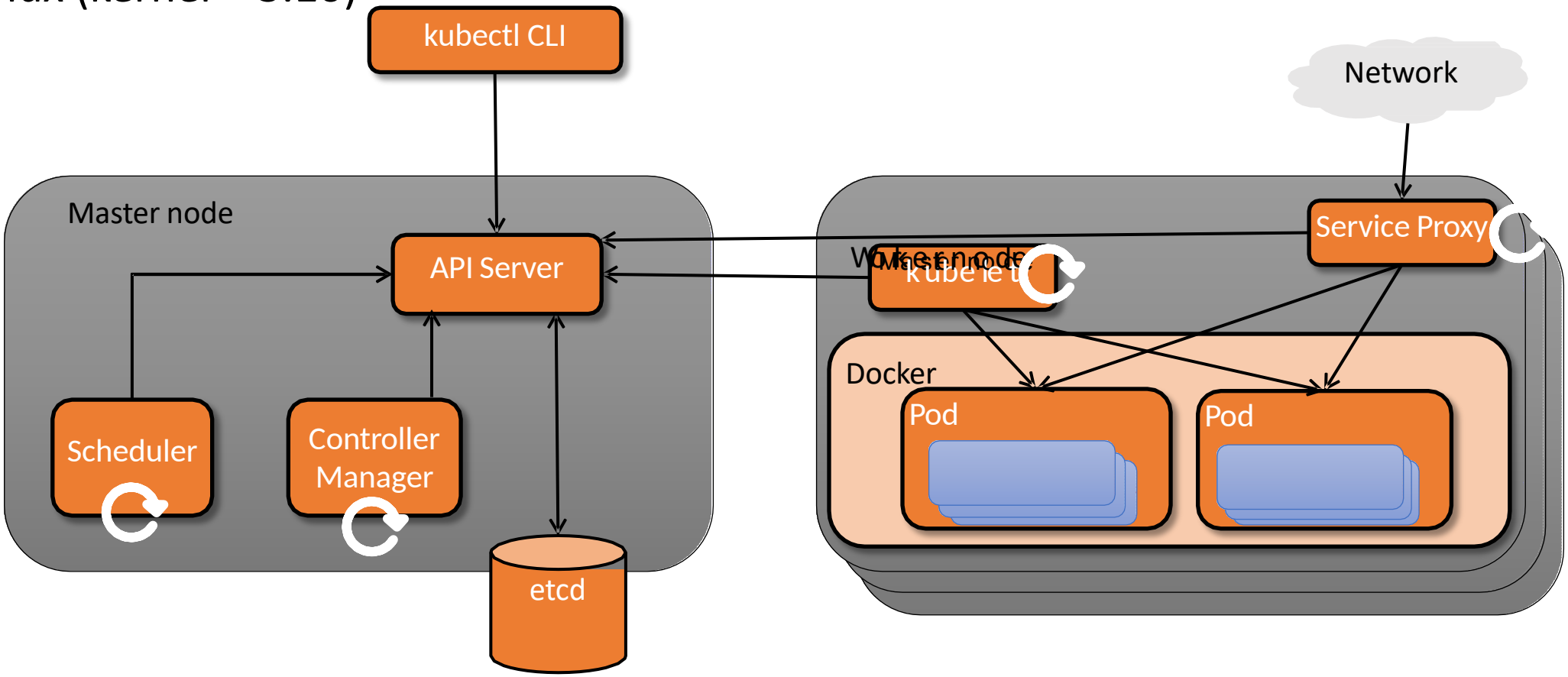
Kubernetes Architecture

Kubernetes Architecture

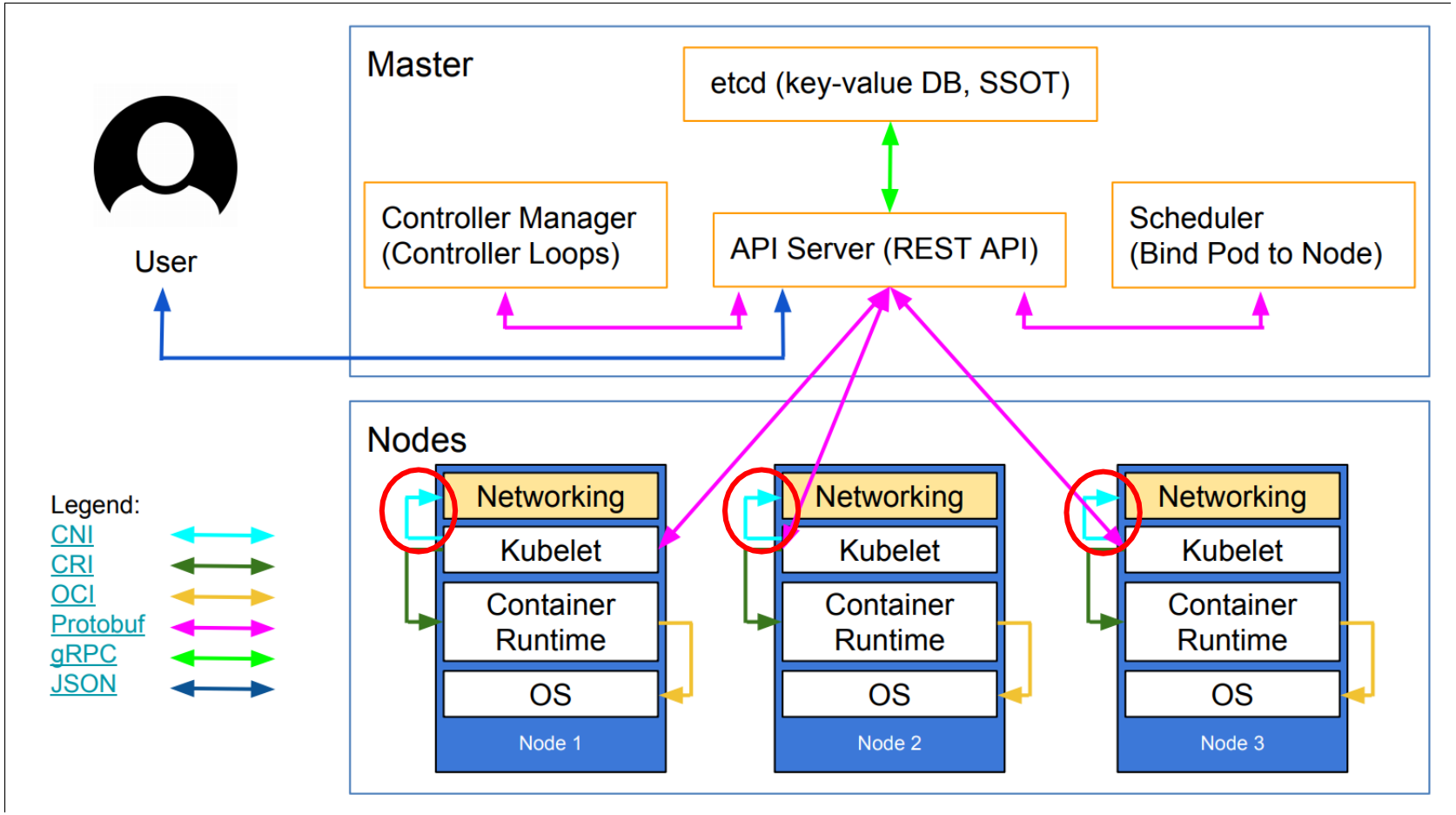

Kubernetes Client
Flannel
Network: 10.254.0.0/16
Backend Type: vxlan
Web UI
kubectl
kubelctl exec nginx-1dzwr ping 10.254.95.4
Kubernetes Master
Controller Manager
Scheduler
API Server
service cluster ip range: 10.254.42.0/24
flanneld
CIDR: 10.254.0.0/16
br0: 10.254.42.1/16
eth0
Interface: flannel.1
CIDR: 10.254.42.0/16
ETCD Cluster
Minion
Kubelet
Kube Proxy
IP: 10.254.50.42
Pod
kube-ui
veth0
nginx-1dzwr
veth0
ping 10.254.95.4
docker0
10.254.50.1/24
docker
Interface: flannel.1
CIDR: 10.254.50.0/16
flanneld
CIDR: 10.254.0.0/16
flannel
Storage
/var/lib/docker
mount -t ext4 /dev/vg_tg1/lv_docker /var/lib/docker
mkfs.ext4 /dev/vg_tg1/lv_docker
Logical Volume Manager
/dev/vg_tg1/lv_docker
Logical Volume
vg_tg1
Volume Group
/dev/mapper/tg1_mpath_a
/dev/mapper/tg1_mpath_b
Physical Volume
Device Mapper Multipath
/dev/sdf
/dev/sdg
/dev/sdh
/dev/sdi
/dev/sdb
/dev/sdc
/dev/sdd
/dev/sde
ISCSI Initiator
Target Login
Discovery
bond0
eth0, eth1
bond1
eth2, eth3
bond2
eth4, eth5
ISCSI Server
iqn.2016-01.6tb.x2:target-3
iqn.2016-01.6tb.x2:target-4
6TB (LUN1)
/dev/sdj
6TB (LUN2)
/dev/sdk
6TB (LUN1)
/dev/sdl
6TB (LUN2)
/dev/sdm
iqn.2016-01.6tb.x2:target-1
iqn.2016-01.6tb.x2:target-2
6TB (LUN1)
/dev/sdf
6TB (LUN2)
/dev/sdg
6TB (LUN1)
/dev/sdh
6TB (LUN2)
/dev/sdi
ISCSI Target
bond0
eth0, eth1
bond1
eth2, eth3
L2 Switch
Internal
Active
Standby
Google Kubernetes
Architecture v1.x
by Yongbok Kim (ruo91)
L2 Switch
External
Active
Standby
IP: 10.254.95.4
nginx-98rsw
nginx-74gsh
/var/lib/docker
mount -t ext4 /dev/vg_tg2/lv_docker /var/lib/docker
mkfs.ext4 /dev/vg_tg2/lv_docker
/dev/vg_tg2/lv_docker
vg_tg2
/dev/mapper/tg2_mpath_a
/dev/mapper/tg2_mpath_b
bond2
eth5, eth4
bond1
eth3, eth2
bond0
eth1, eth0

Kubernetes Cluster Architecture

- Kubernetes nodes can be physical hosts or VM's running a container-friendly Linux (kernel > 3.10)

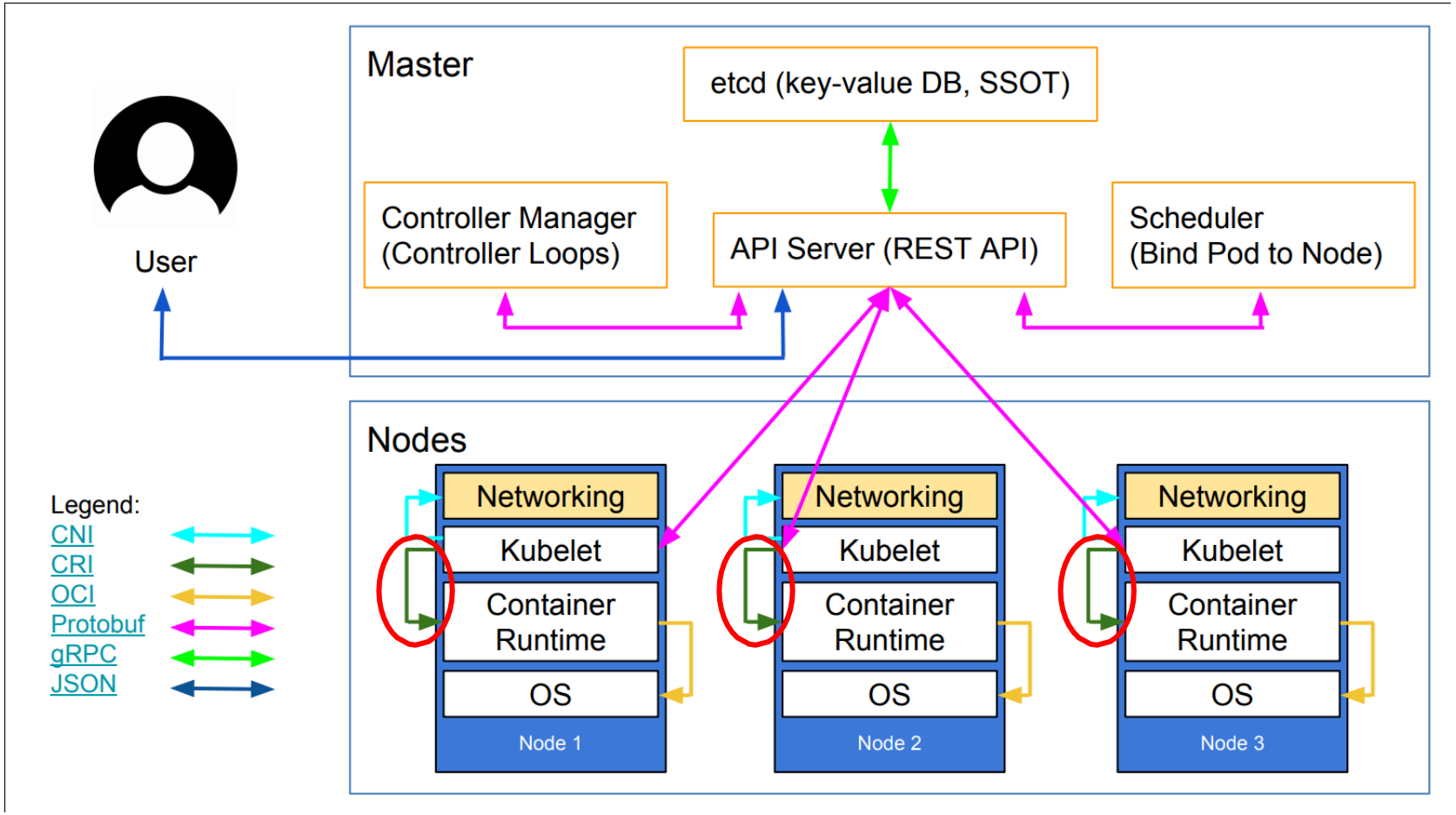


Components Communication



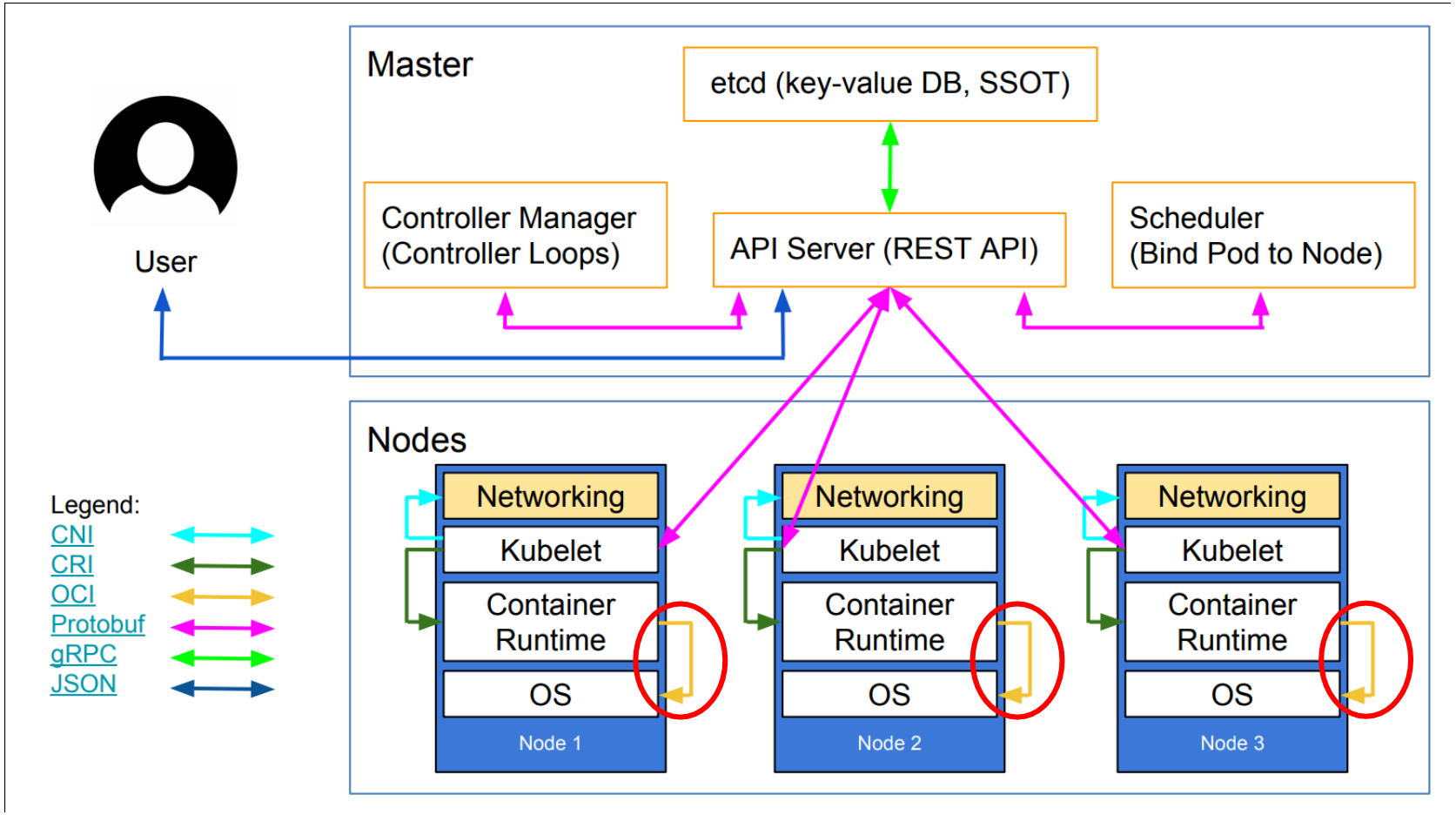
- CNI = Container Network Interface

Components Communication



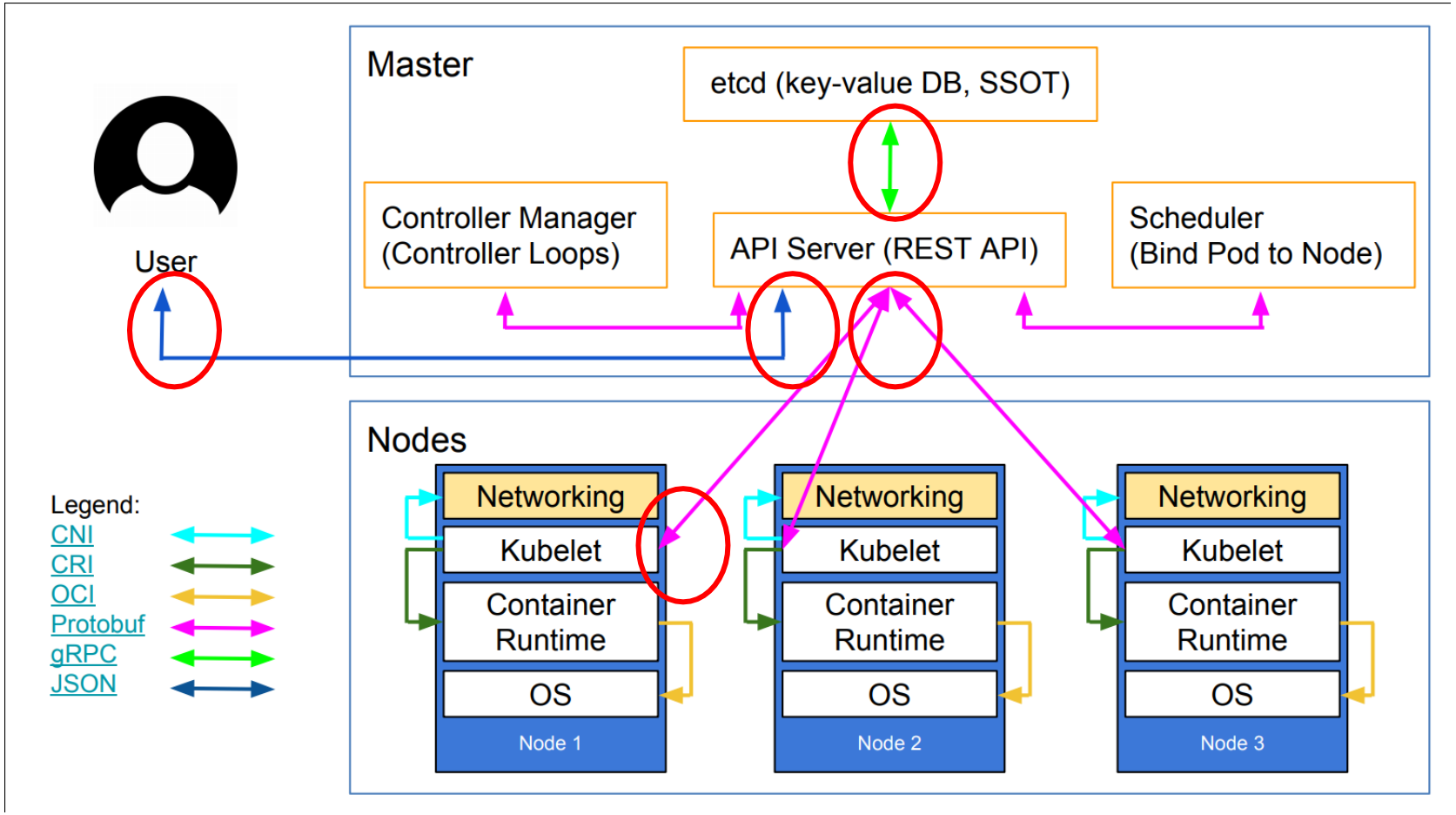
- CNI = Container Network Interface
- CRI = Container Runtime Interface
  - Docker, CRI-O, Containerd

Components Communication



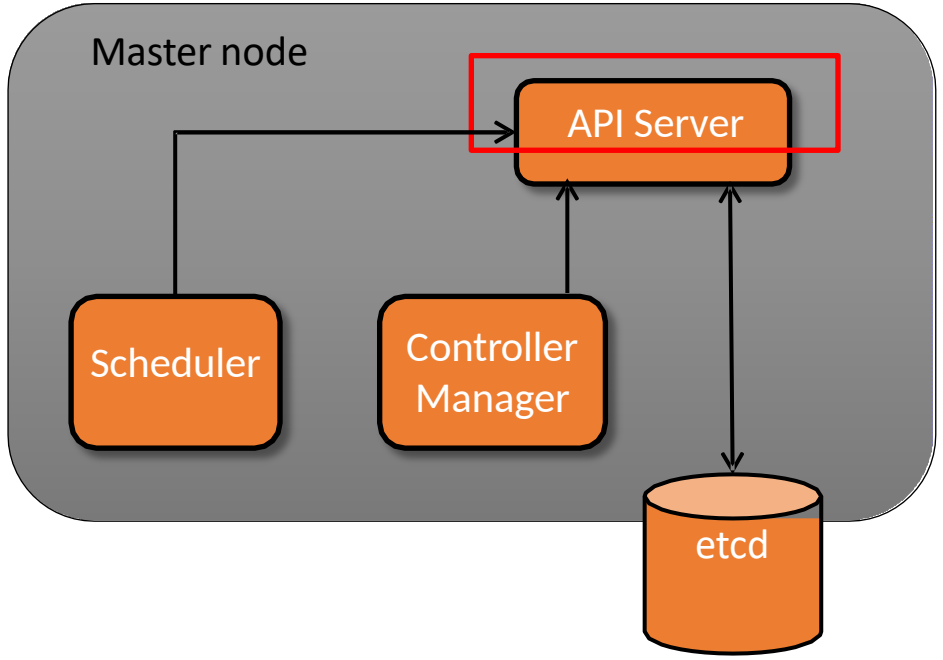
- CNI = Container Network Interface
- CRI = Container Runtime Interface
- Docker, CRI-O, Containerd
- OCI = Open Container Initiative

Components Communication



- CNI = Container Network Interface
- CRI = Container Runtime Interface
- Docker, CRI-O, Containerd
- OCI = Open Container Initiative
- Protobuf
- gRPC
- JSON

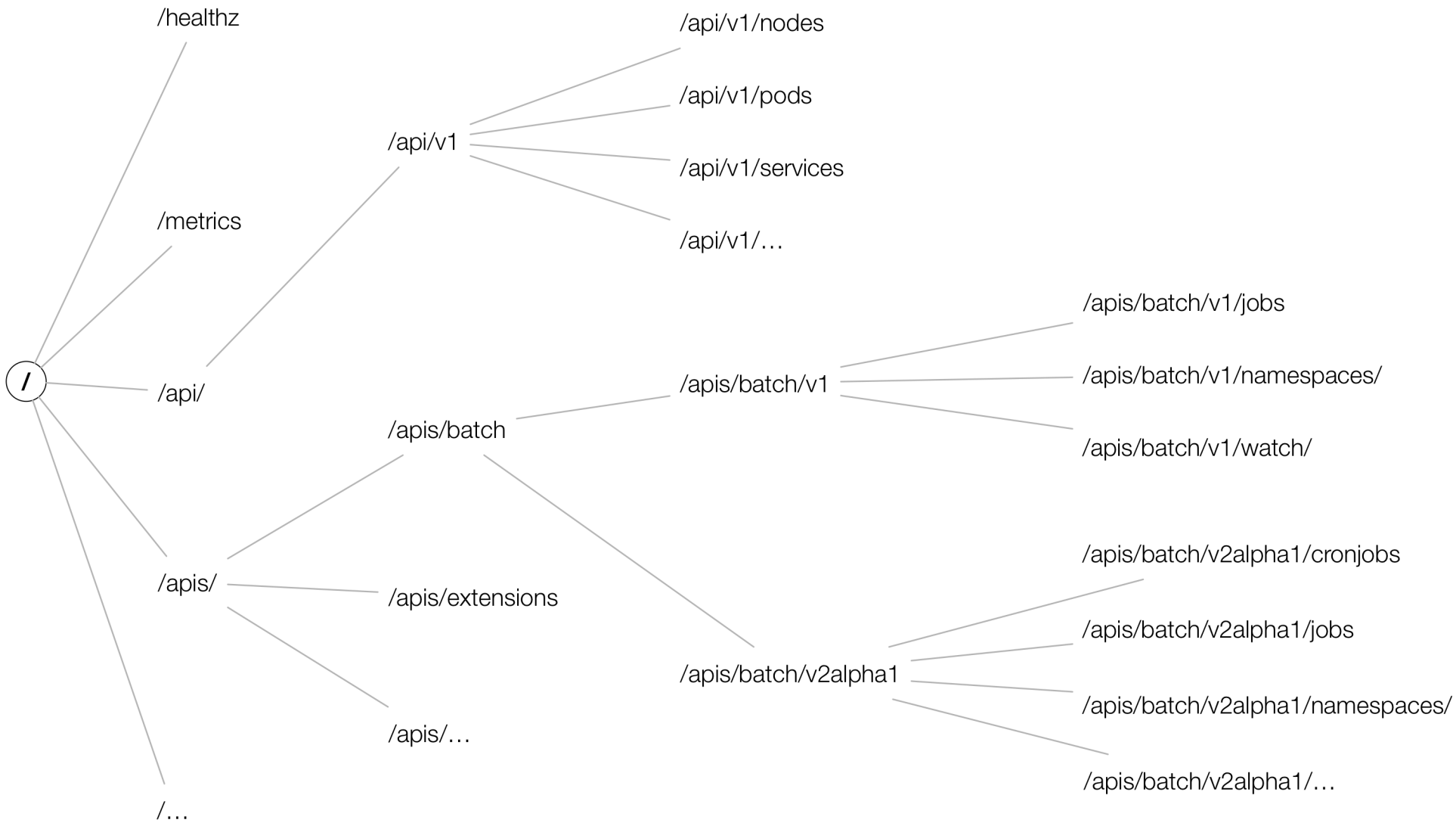
Kubernetes Master Node Components



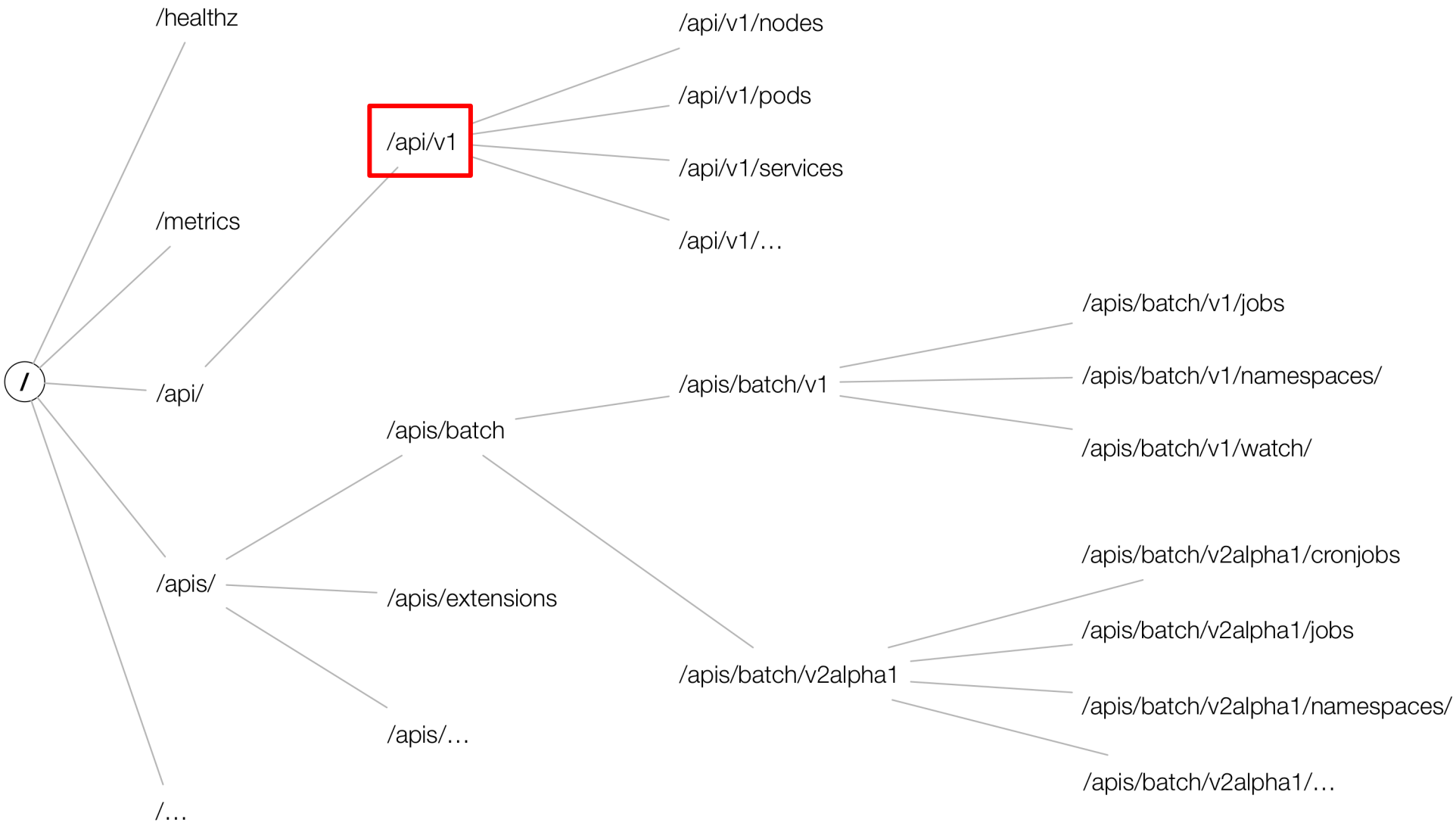
- **API Server (kube-apiserver)**: exposes the Kubernetes REST API, and can be scaled horizontally
- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on
- **Controller manager (kube-controller-manager)**: runs background controller processes for the system to enforce declared object states, e.g. Node Controller, Replication Controller, …
- **Persistent data store (etcd)**: all K8s system data is stored in a distributed, reliable key-value store. etcd may run on separate nodes from the master

K8s components written in Go (golang.org)

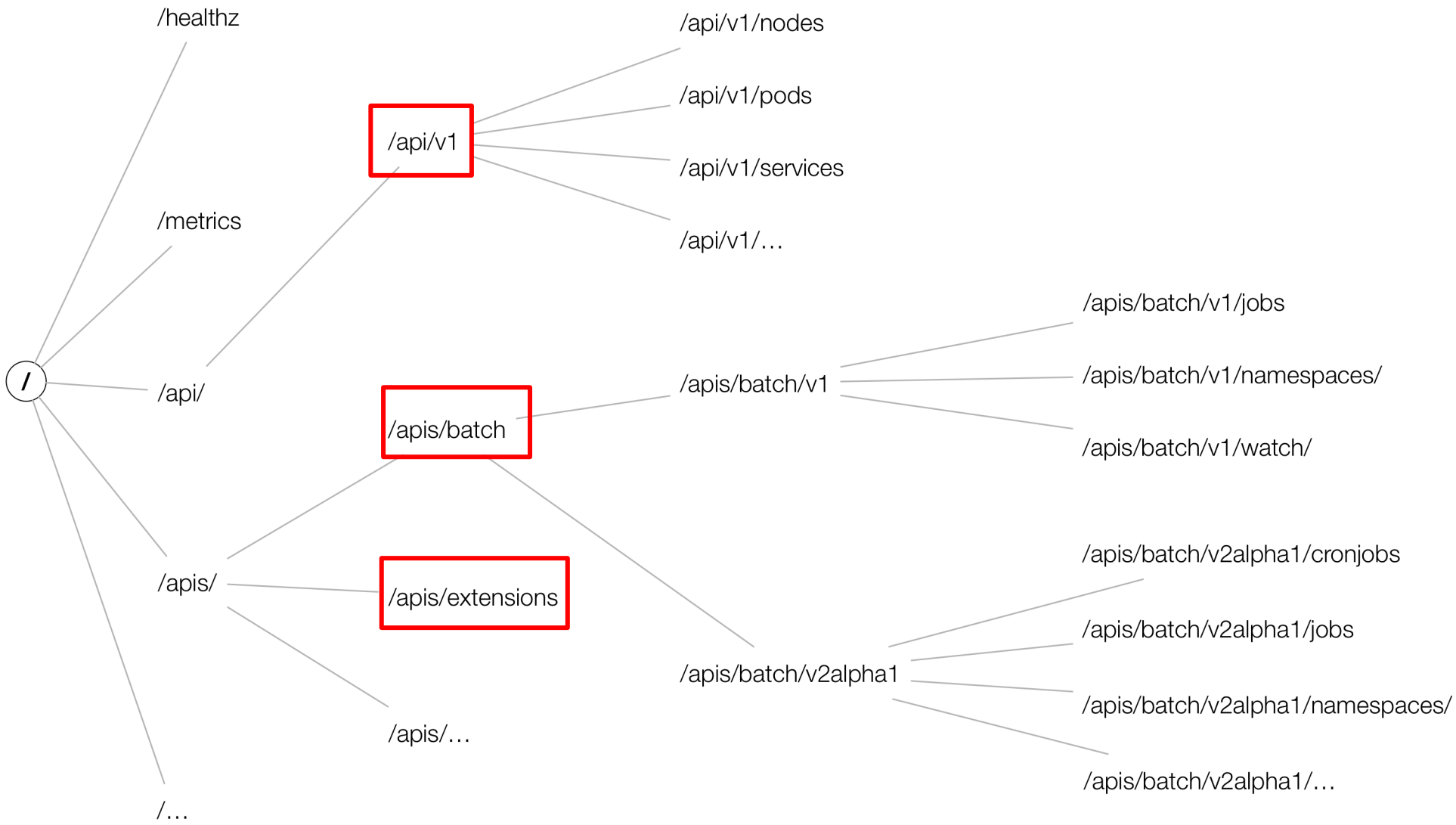
API Deep Dive



API Groups


/
/healthz
/metrics
/api/
/api/v1
/api/v1/nodes
/api/v1/pods
/api/v1/services
/api/v1/…
/apis/
/apis/batch
/apis/batch/v1
/apis/batch/v1/jobs
/apis/batch/v1/namespaces/
/apis/batch/v1/watch/
/apis/extensions
/apis/batch/v2alpha1
/apis/batch/v2alpha1/cronjobs
/apis/batch/v2alpha1/jobs
/apis/batch/v2alpha1/namespaces/
/apis/batch/v2alpha1/…
/apis/…
/…

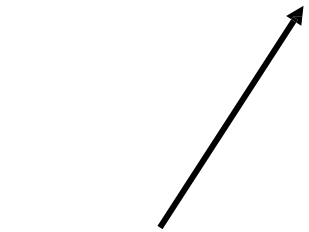
API Groups


/
/healthz
/metrics
/api/
/apis/
/…
/api/v1
/apis/batch
/apis/extensions
/apis/…
/api/v1/nodes
/api/v1/pods
/api/v1/services
/api/v1/…
/apis/batch/v1
/apis/batch/v2alpha1
/apis/batch/v1/jobs
/apis/batch/v1/namespaces/
/apis/batch/v1/watch/
/apis/batch/v2alpha1/cronjobs
/apis/batch/v2alpha1/jobs
/apis/batch/v2alpha1/namespaces/
/apis/batch/v2alpha1/…

API Group

- Collection of Kinds that are logically related
  - Job, ScheduledJob in batch API Group

/apis/batch/v1/jobs



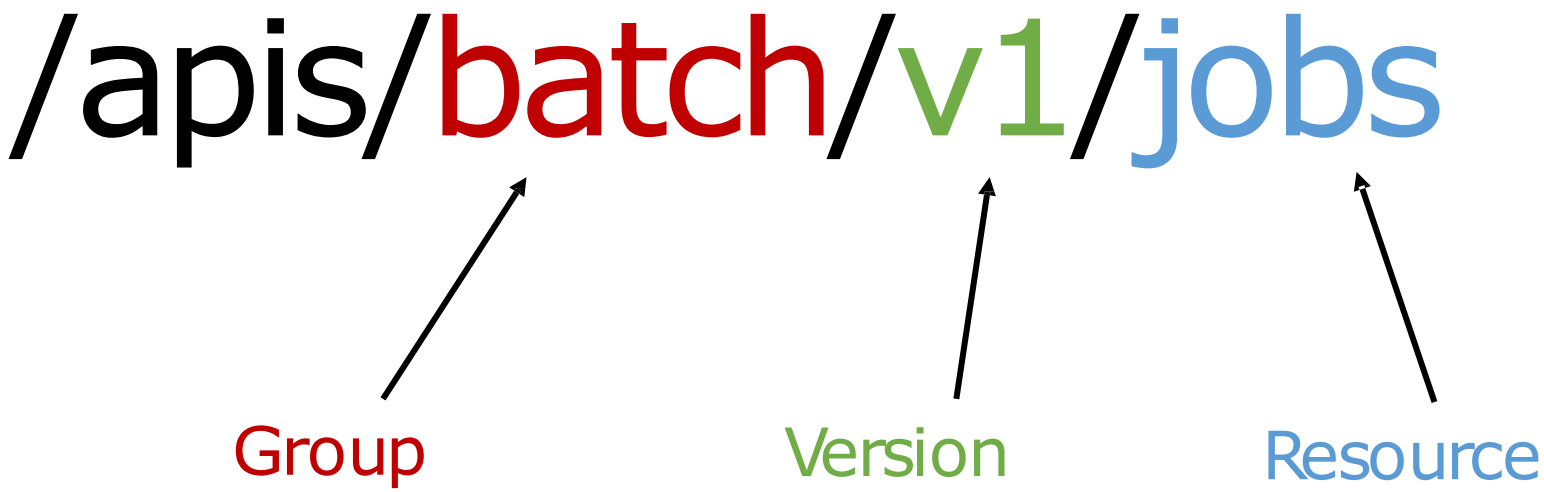
Group

API Version

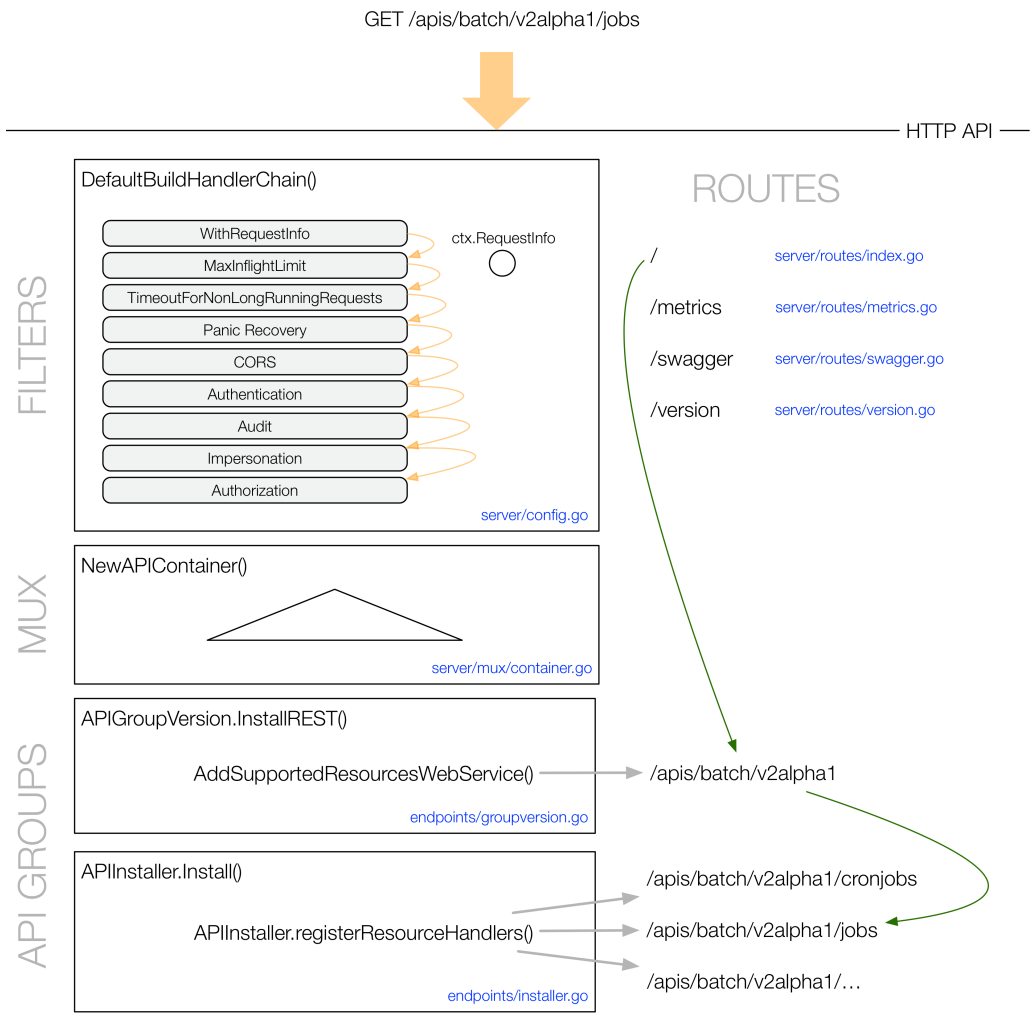
- Each API Group can be part of multiple versions
  - v1alpha1 -> v1beta1 -> v1

API Resource

- System entity being manipulated as JSON over HTTP



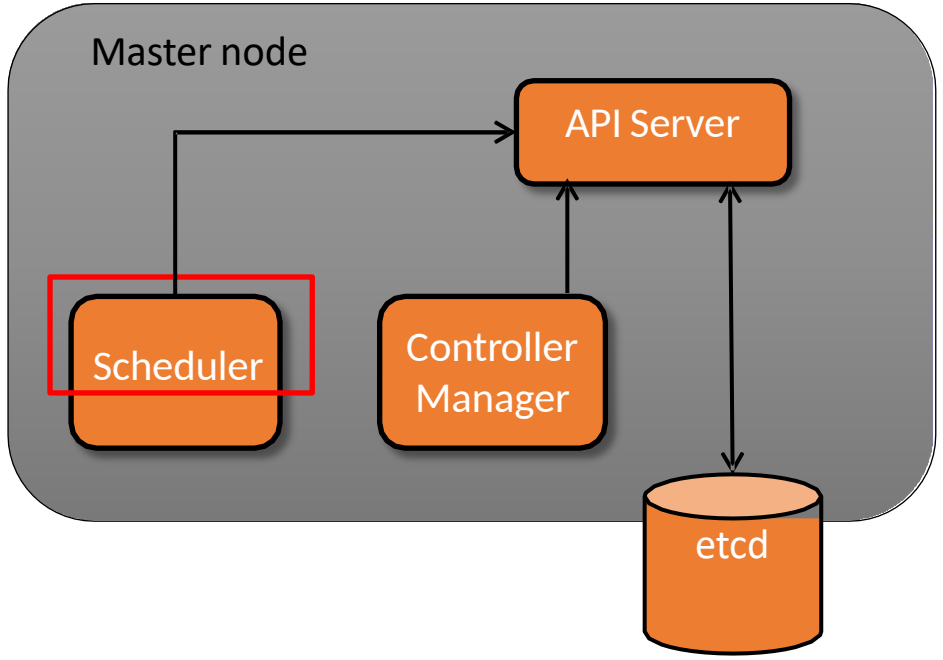
Request Flow and Processing



1. HTTP request is processed
2. Multiplexer routes the HTTP request to handler depending on path
3. Routes connect handlers with HTTP paths
4. The handler, registered per API Group takes the HTTP request context (user, rights etc.) delivers the requested object from storage (etcd)

base for Go packages: k8s.io/apiserver/pkg/

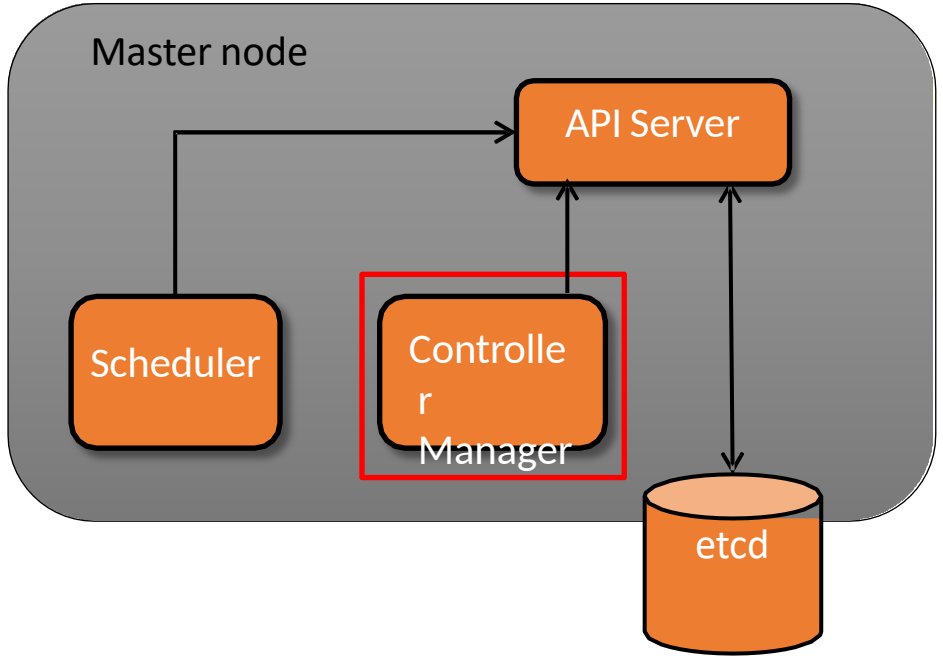
Kubernetes Master Node Components



- **API Server (kube-apiserver)**: exposes the Kubernetes REST API, and can be scaled horizontally
- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on
- **Controller manager (kube-controller-manager)**: runs background controller processes for the system to enforce declared object states, e.g. Node Controller, Replication Controller, …
- **Persistent data store (etcd)**: all K8s system data is stored in a distributed, reliable key-value store. etcd may run on separate nodes from the master

K8s components written in Go (golang.org)

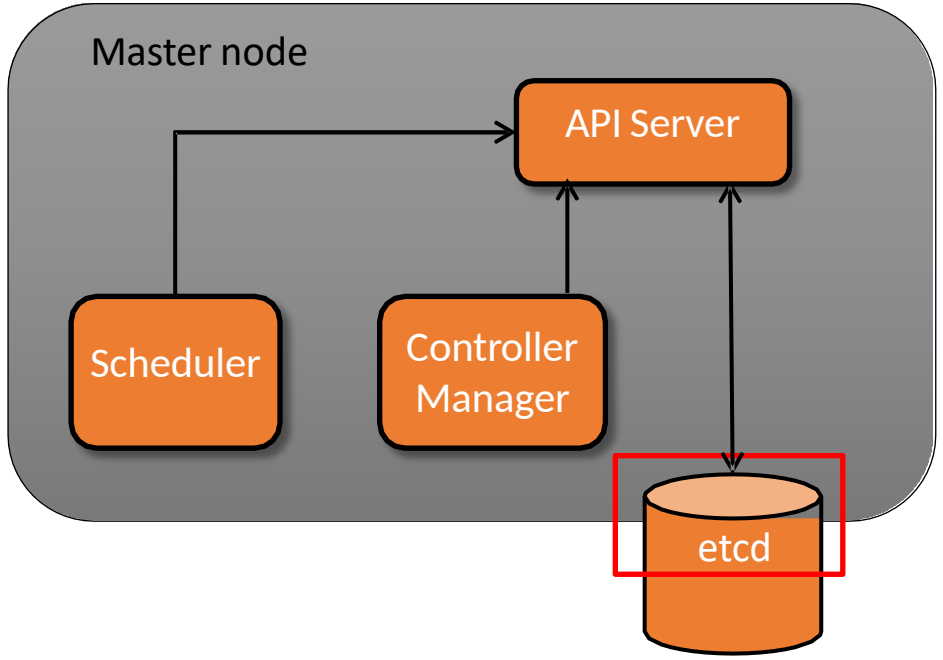
Kubernetes Master Node Components



- **API Server (kube-apiserver)**: exposes the Kubernetes REST API, and can be scaled horizontally
- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on
- **Controller manager (kube-controller-manager)**: runs background controller processes for the system to enforce declared object states, e.g. Node Controller, Replication Controller, …
- **Persistent data store (etcd)**: all K8s system data is stored in a distributed, reliable key-value store. etcd may run on separate nodes from the master

K8s components written in Go (golang.org)

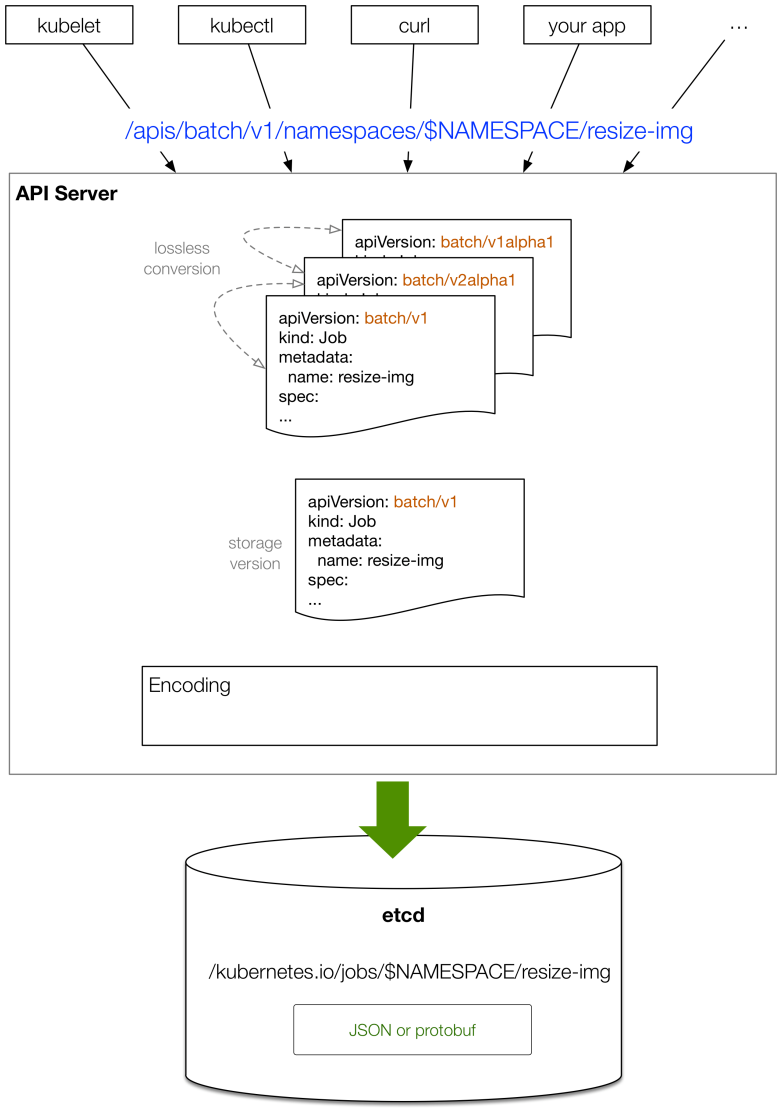
Kubernetes Master Node Components



- **API Server (kube-apiserver)**: exposes the Kubernetes REST API, and can be scaled horizontally
- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on
- **Controller manager (kube-controller-manager)**: runs background controller processes for the system to enforce declared object states, e.g. Node Controller, Replication Controller, …
- **Persistent data store (etcd)**: all K8s system data is stored in a distributed, reliable key-value store. etcd may run on separate nodes from the master

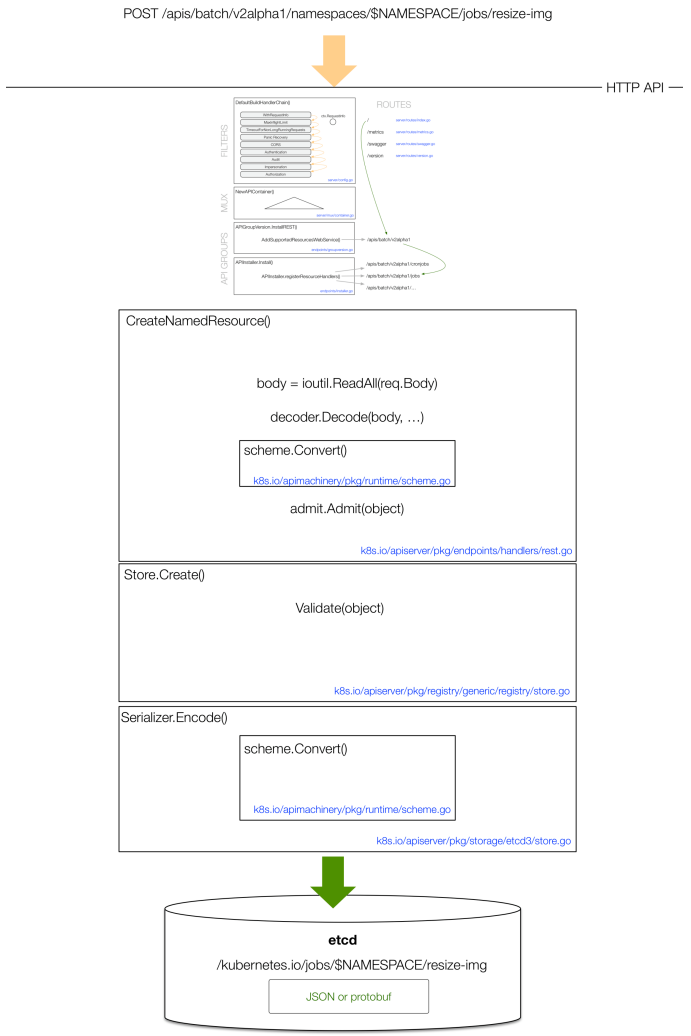
K8s components written in Go (golang.org)

Etcd flow



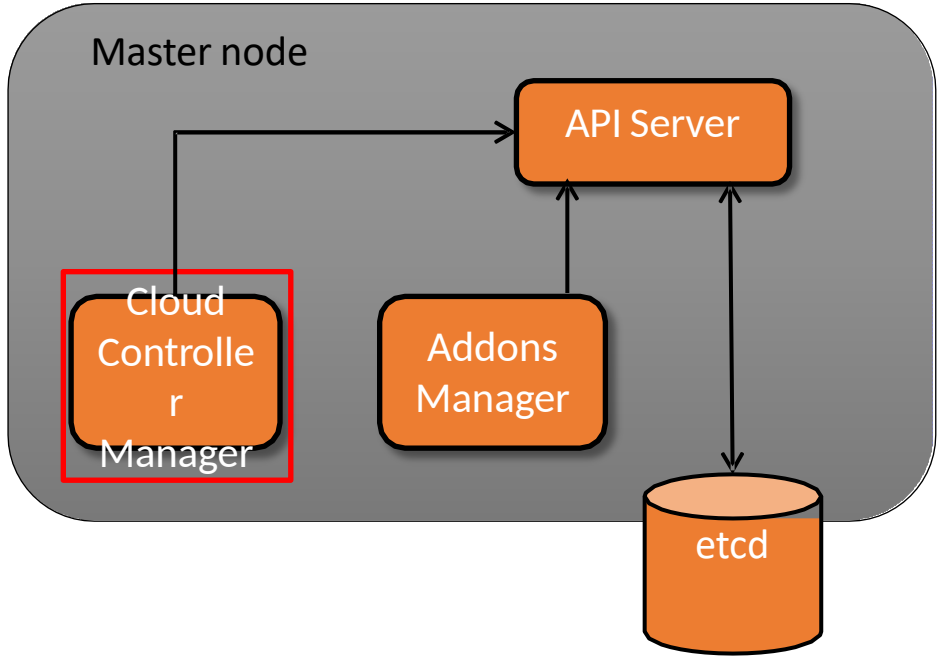
- **Etcd flow:**
  - client provides desired object state in YAML or JSON
  - kubectl converts YAML to JSON and sends it to API
  - API server turns input object state into canonical storage version
  - storage process in etcd, at a certain key, into a value with the encoding to JSON or protobuf.

Serialization of State Flow



- **Serialization flow:**
  - API Server keeps all known k8s object kinds in a Go registry (Scheme).
  - Version of kinds defined along with how they can be:
    - converted
    - new objects created
    - objects encoded/decoded to JSON/protobuf

Master Node Additional Components



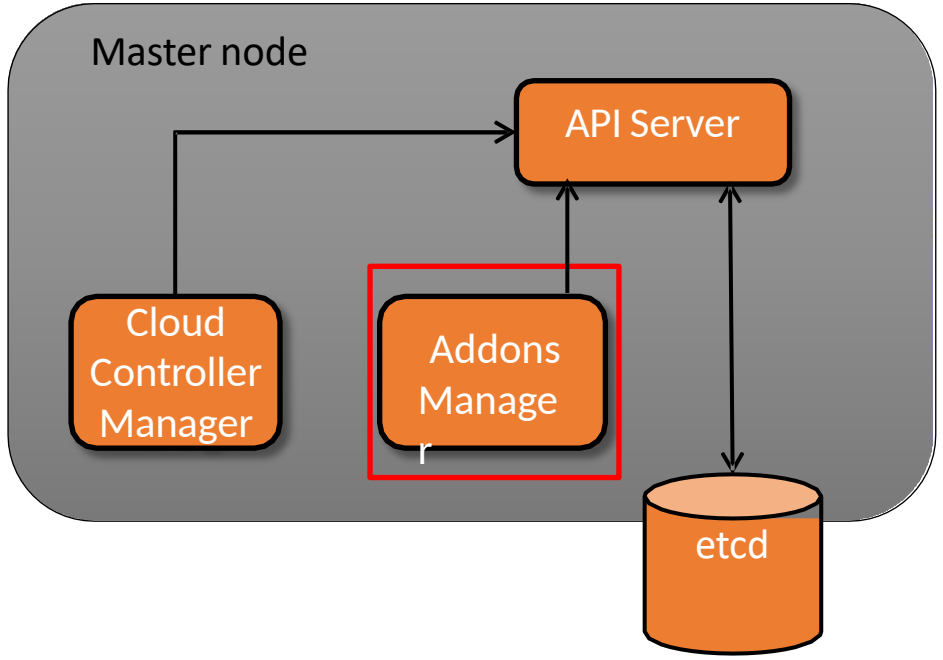
- **cloud-controller-manager**: runs controllers interacting with underlying IaaS providers – alpha feature

  Allows cloud vendor-specific code to be separate from main K8s system components

- **addons-manager**: creates and maintains cluster addon resources in 'kube-system' namespace, e.g.
  - **Kubernetes Dashboard**: general web UI for application and cluster management
  - **kube-dns**: serves DNS records for K8s services and resources
  - Container resource monitoring and cluster-level logging

K8s components written in Go (golang.org)

Master Node Additional Components



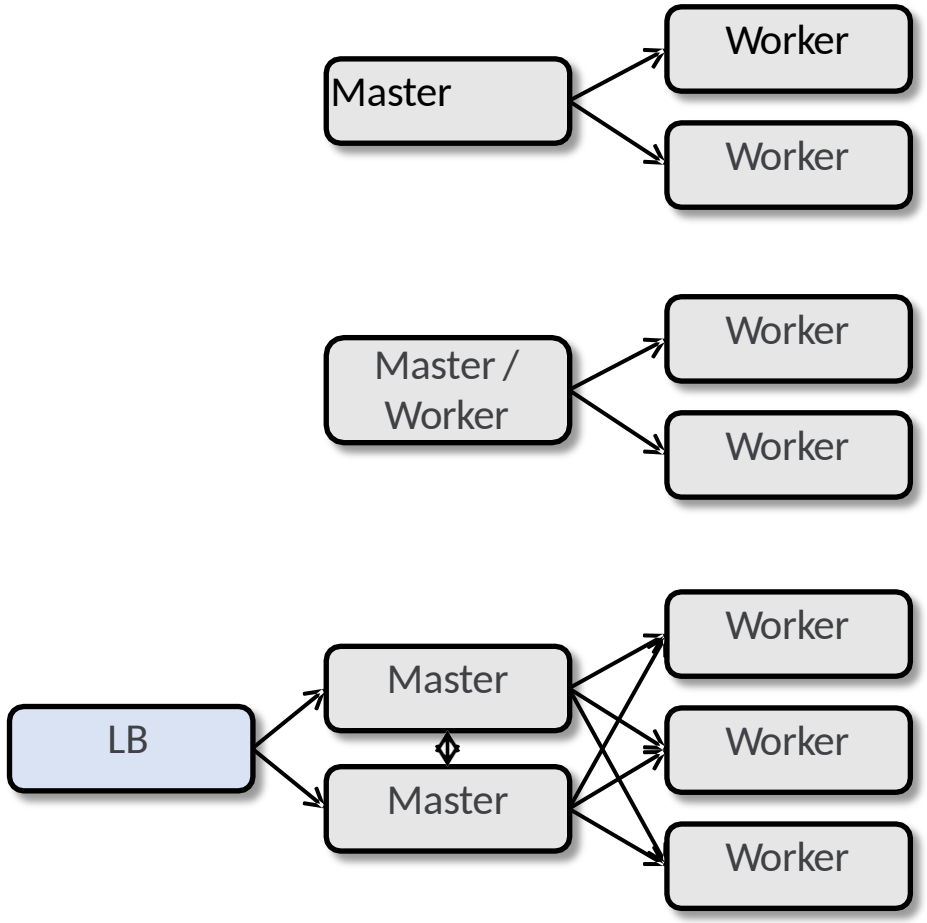
- **cloud-controller-manager**: runs controllers interacting with underlying IaaS providers – alpha feature
  - Allows cloud vendor-specific code to be separate from main K8s system components

- **addons-manager**: creates and maintains cluster addon resources in 'kube-system' namespace, e.g.
  - **Kubernetes Dashboard**: general web UI for application and cluster management
  - **kube-dns**: serves DNS records for K8s services and resources
  - Container resource monitoring and cluster-level logging

K8s components written in Go (golang.org)

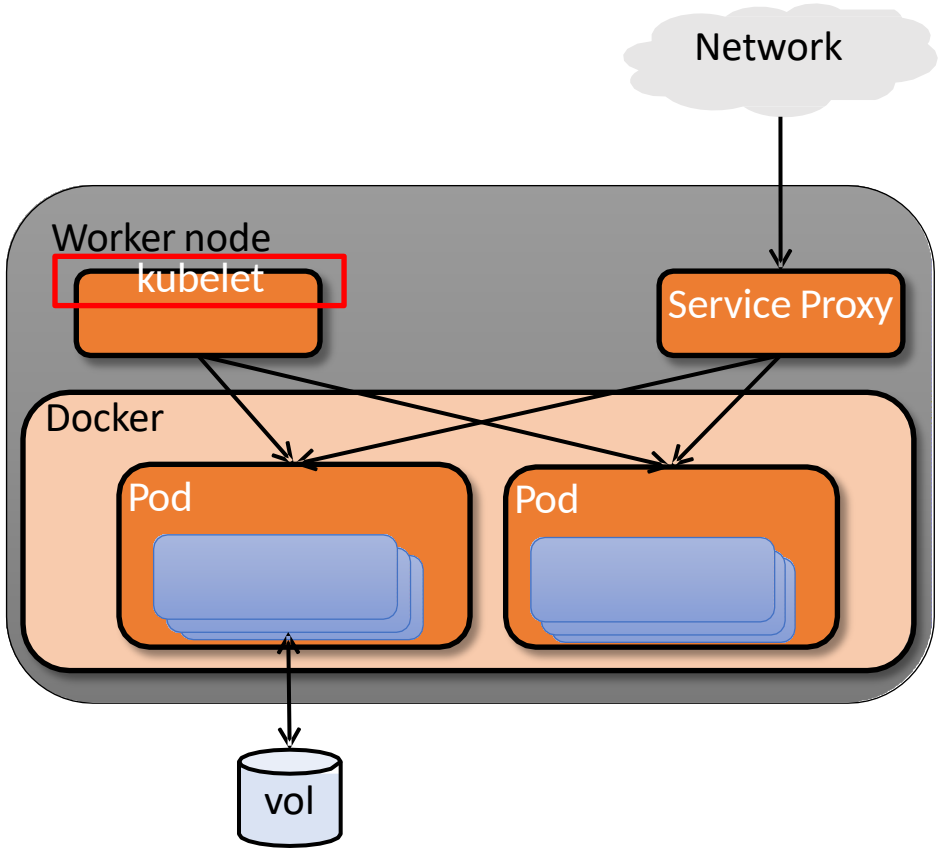
Kubernetes Master Node Deployment Options

- Simple cluster has a single master node, with single API/scheduler/controller
- At small scale, master may also be a worker node
- Can create a resilient control plane for K8s using a cluster of master node replicas behind a loadbalancer
  - Kube-apiserver and etcd scale out horizontally
  - Kube-scheduler and kube-controller-manager use master election to run single instances at a time



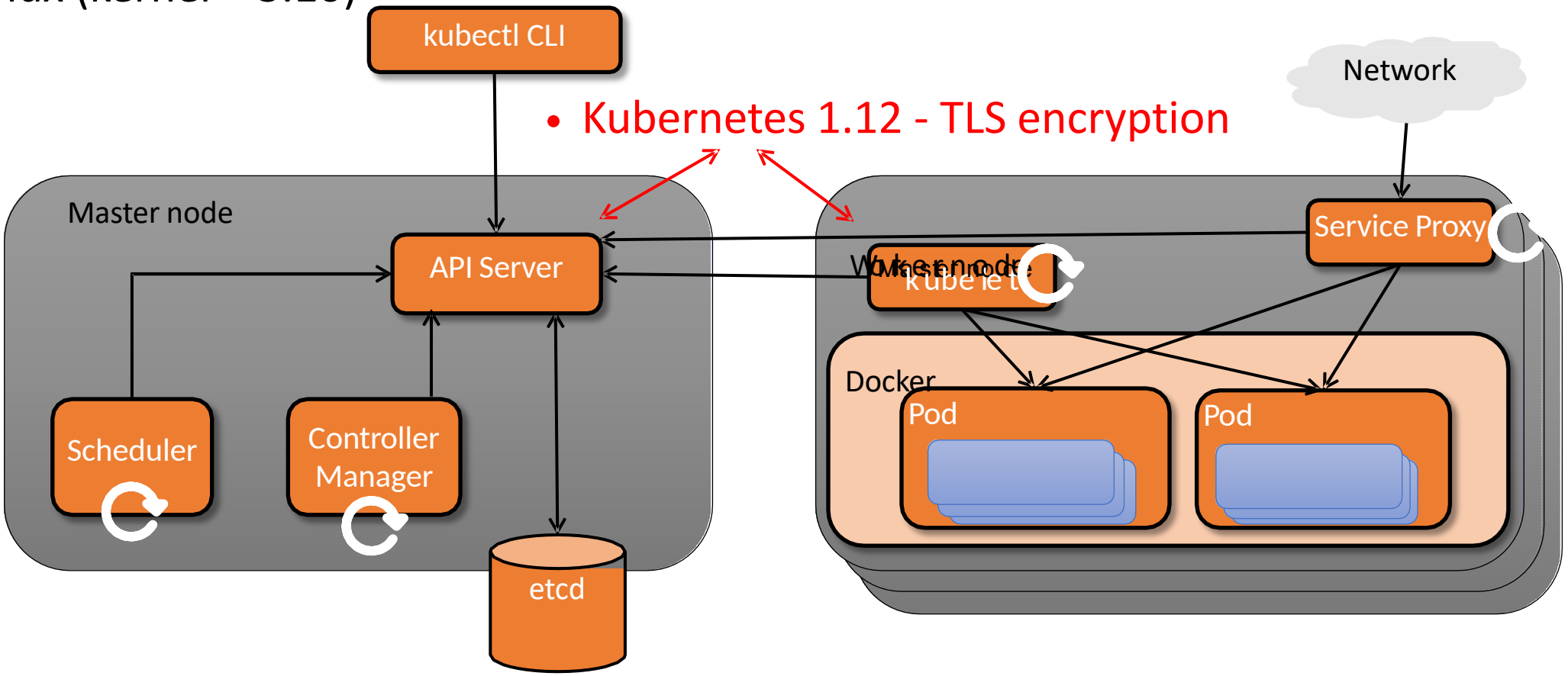
Kubernetes Worker Node Components

- **kubelet**: local K8s agent that is responsible for operations on the node, including
  - Watching for pod assignments
  - Mounting pod required volumes
  - Running a pod's containers
  - Executing container liveness probes
  - Reporting pod status to system
  - Reporting node status to system
- **Service proxy (kube-proxy):** enables K8s service abstractions by maintaining host network rules and forwarding connections
- **Docker:** runs the containers



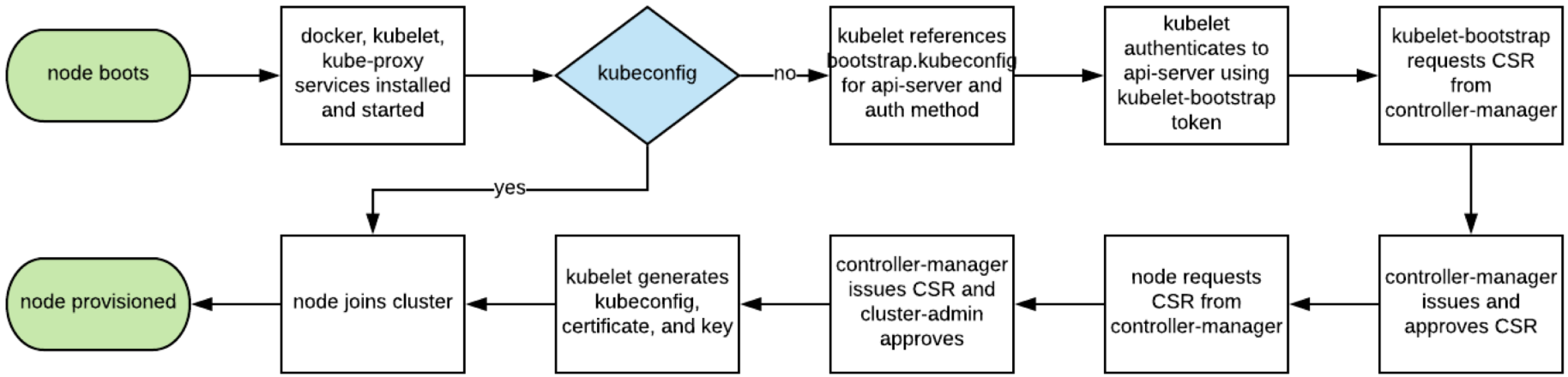
Kubernetes Cluster Architecture

- Kubernetes nodes can be physical hosts or VM's running a container-friendly Linux (kernel > 3.10)



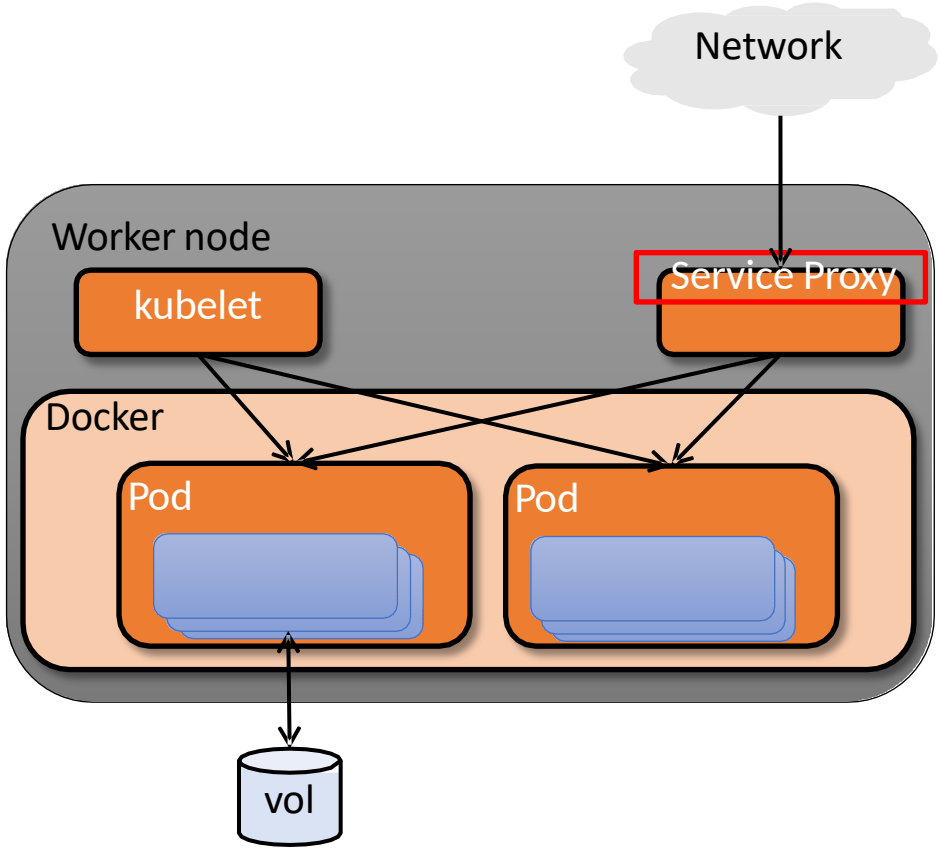
Kubernetes Cluster Architecture

- Kubernetes with TLS bootstrap process



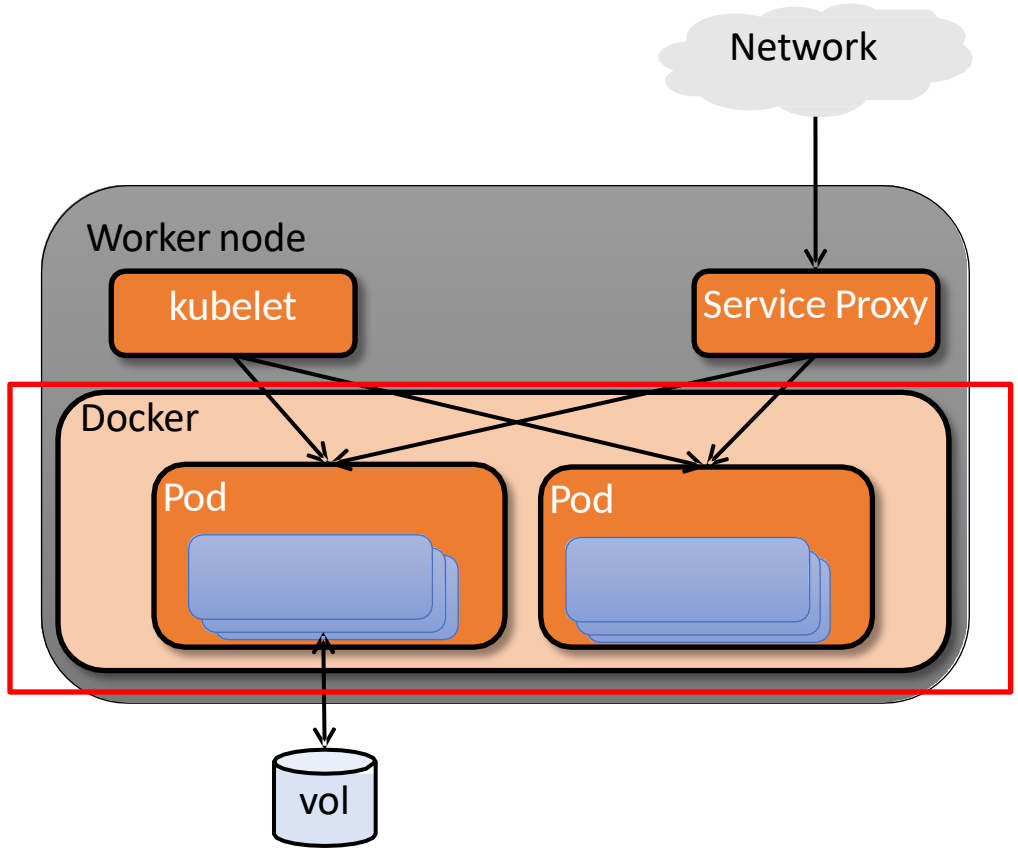
Kubernetes Worker Node Components

- **kubelet**: local K8s agent that is responsible for operations on the node, including
  - Watching for pod assignments
  - Mounting pod required volumes
  - Running a pod's containers
  - Executing container liveness probes
  - Reporting pod status to system
  - Reporting node status to system
- **Service proxy (kube-proxy):** enables K8s service abstractions by maintaining host network rules and forwarding connections
- **Docker:** runs the containers



Kubernetes Worker Node Components

- **kubelet**: local K8s agent that is responsible for operations on the node, including
  - Watching for pod assignments
  - Mounting pod required volumes
  - Running a pod's containers
  - Executing container liveness probes
  - Reporting pod status to system
  - Reporting node status to system
- **Service proxy (kube-proxy):** enables K8s service abstractions by maintaining host network rules and forwarding connections
- **Docker:** container runtime



Kubernetes Installation

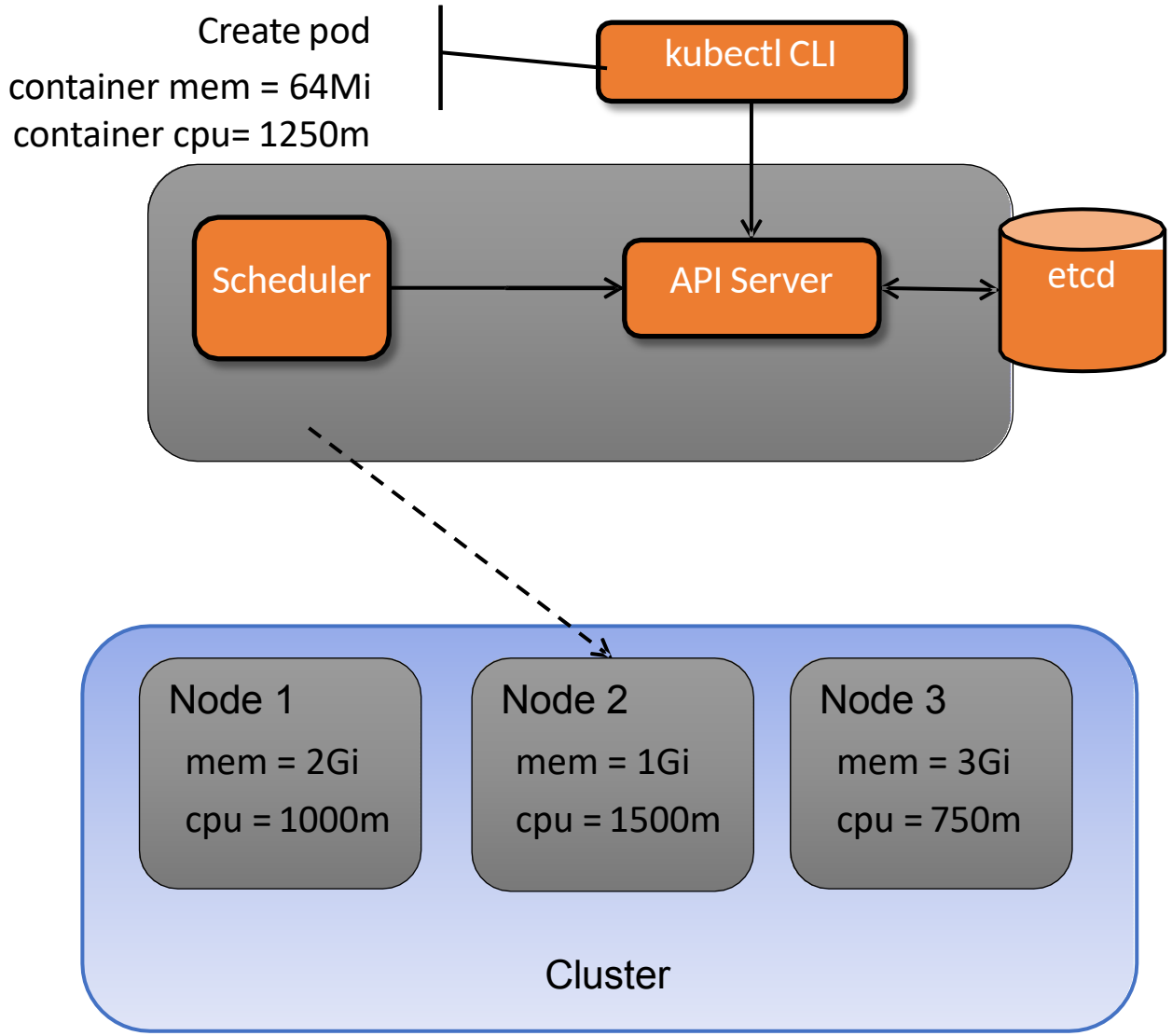
Install Kubernetes

- Install pre-requisite packages on all 3 nodes
- Initialize master
- Join nodes to cluster

Kubernetes POD Scheduling

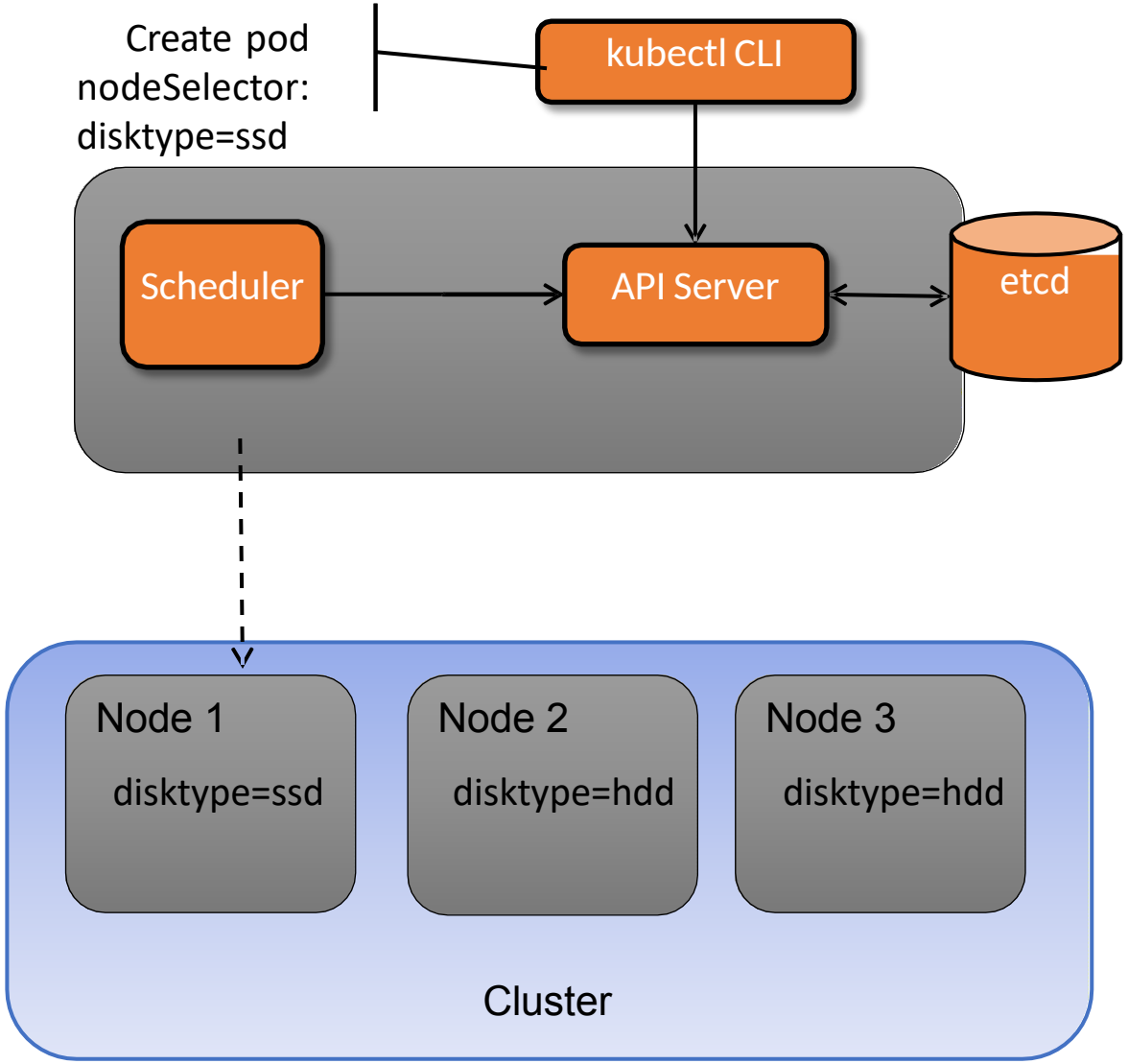
Pod Scheduling Overview

- kube-scheduler on the master node assigns new pod requests to appropriate worker nodes
- Default scheduler takes account of
  - Available node CPU/RAM
  - Resource requests from new pod – sum of resource requests of pod containers
- Default scheduler will automatically
  - Schedule pod on node with sufficient free resources
  - Spread pods across nodes in the cluster
- Can specify custom schedulers for pods



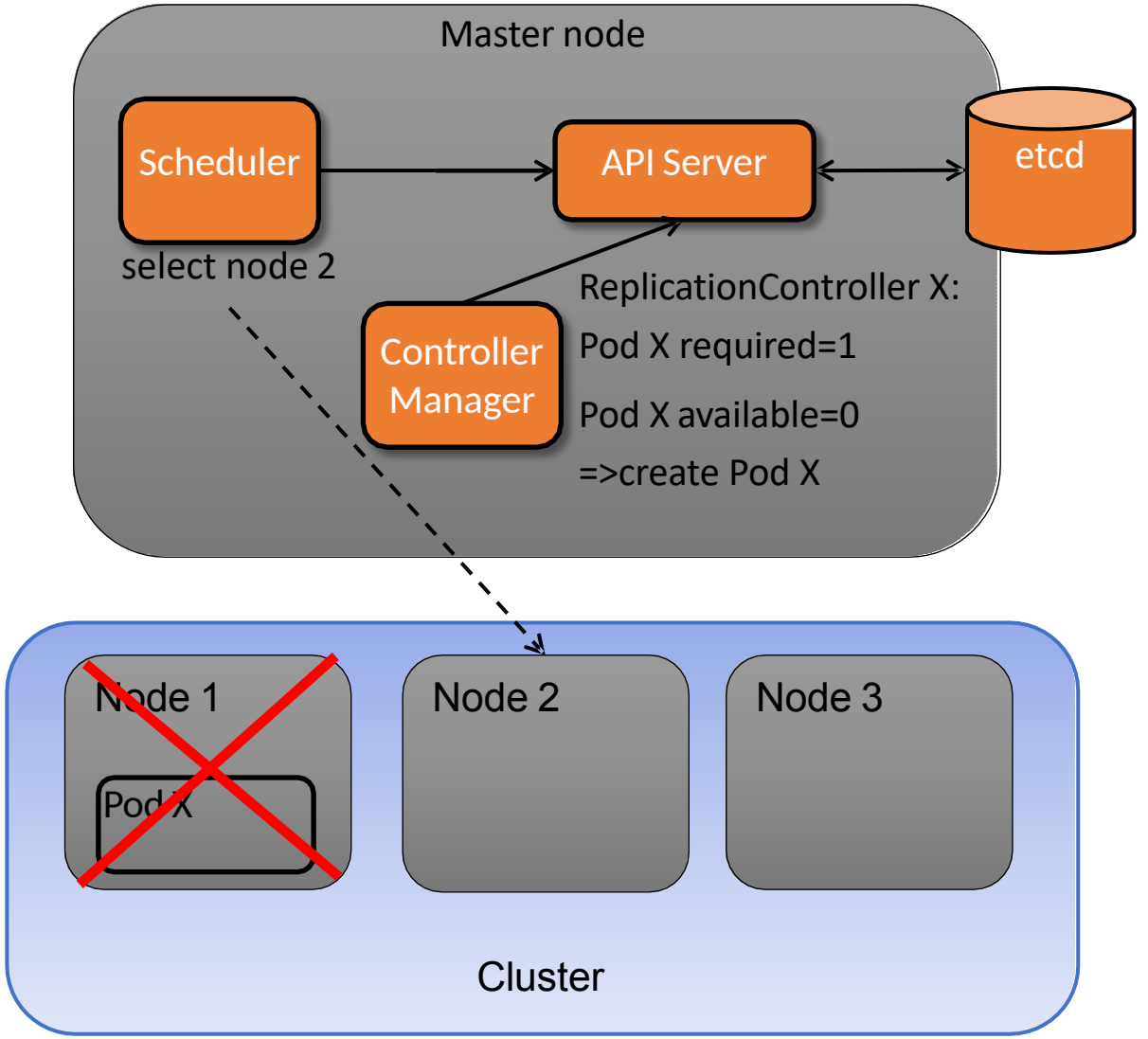
Controlling Pod Scheduling with Default Scheduler

- Nodes carry labels indicating topology and other resource notes
- Users can require pods to be scheduled on nodes with specific label(s) via a nodeSelector in container spec



Scheduling Pod Re-creation Driven by Control Loops

- kube-scheduler performs same node selection operation when new pod created due to e.g. node loss
- kube-controller-manager runs controllers like ReplicationController managing number of pod instances available
- kube-controller-manager will initiate request for new pods as needed, which will be scheduled by kube-scheduler per pod/container spec



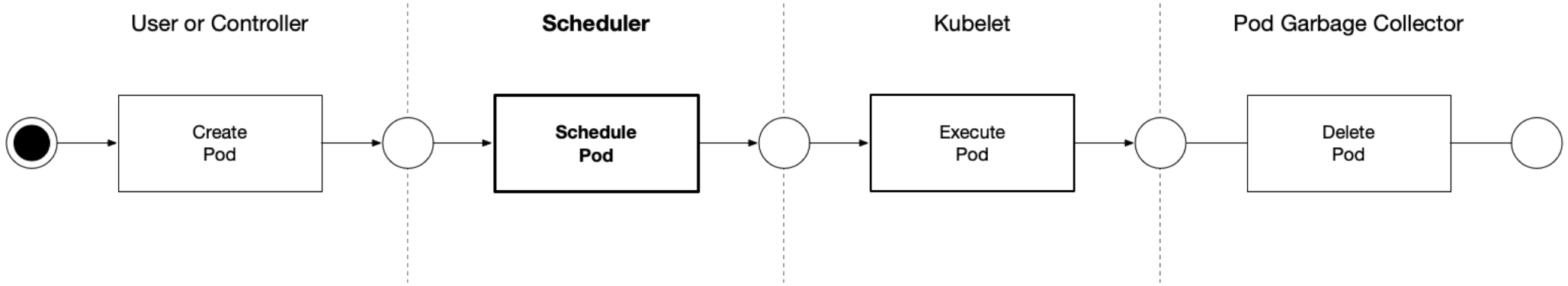
Kubernetes Scheduling

Node Selector

```
apiVersion: v1
kind: Pod
metadata:
    name: nginx
    labels:
      env: test
spec:
  container:
  - name: nginx
    image: nginx
    nodeSelector:
      disktype: ssd
```

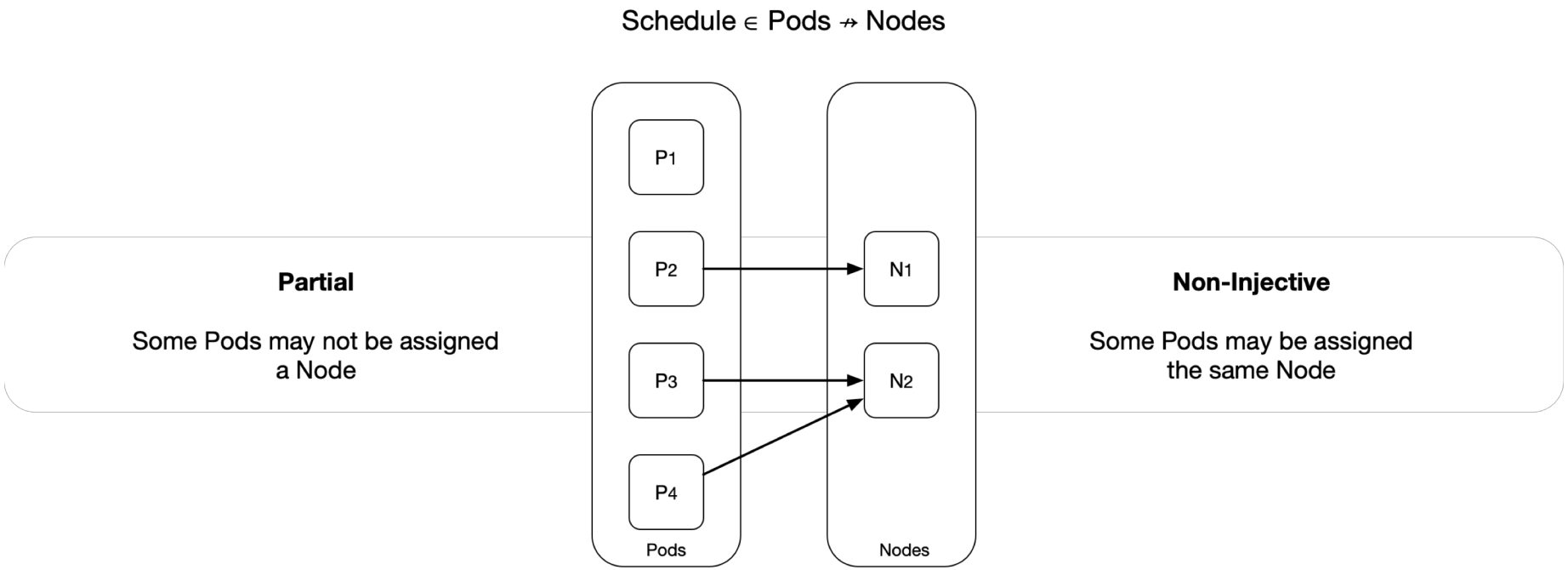
Scheduling flow

- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on, this is called a "placement"
  - Placement: a partial, non-injective assignement of a set of Pods to a set of Nodes.



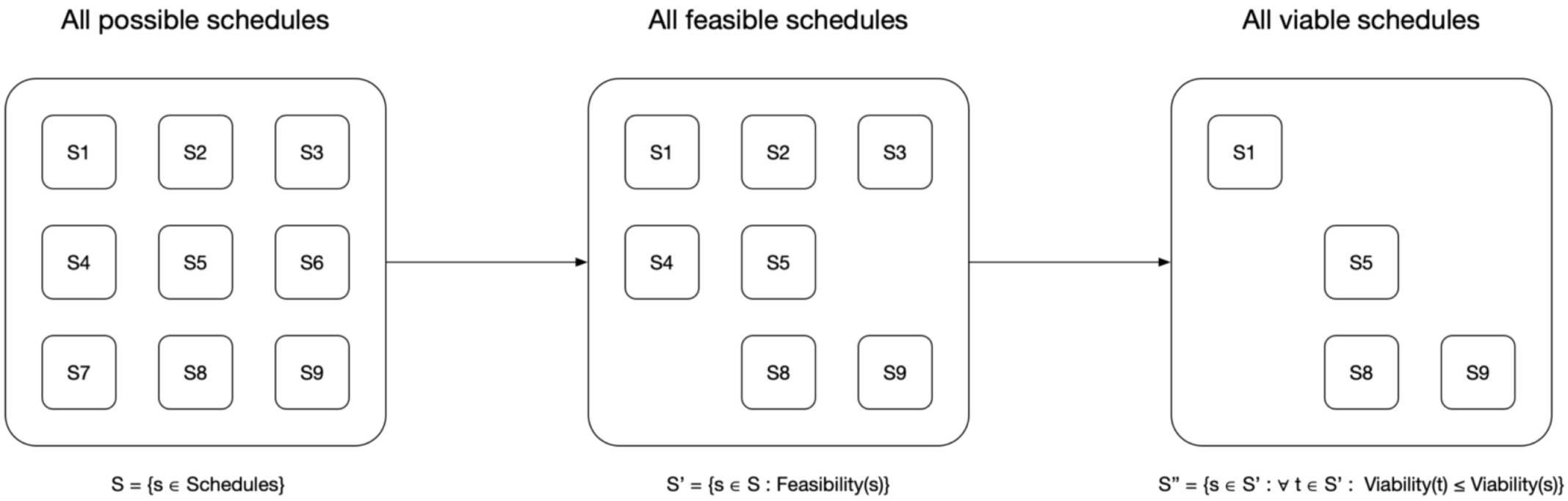
Scheduler details

- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on, this is called a "placement"
  - Placement: a partial, non-injective assignement of a set of Pods to a set of Nodes.



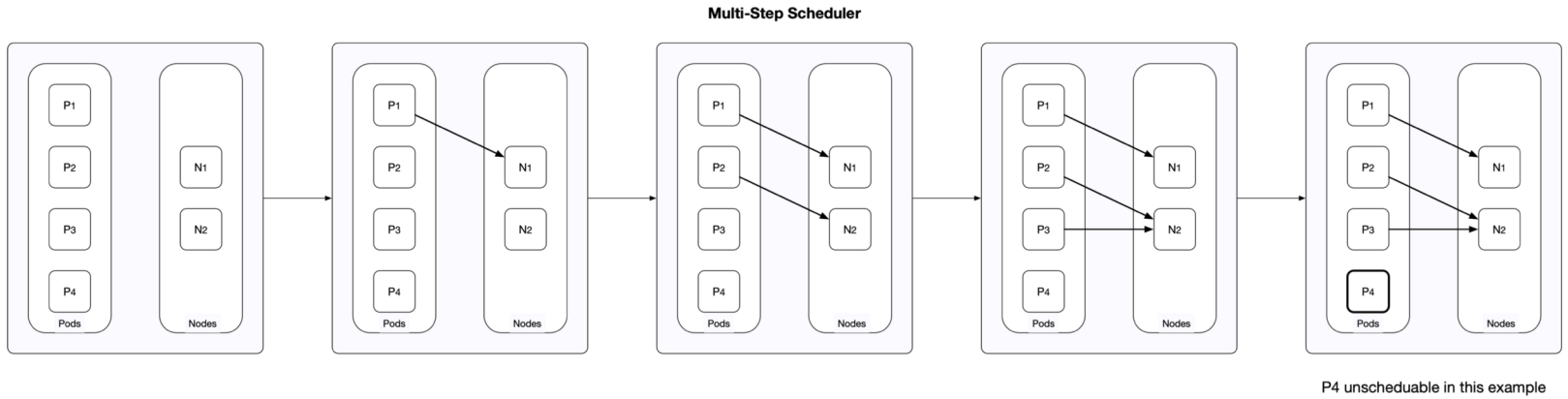
Scheduler decisions

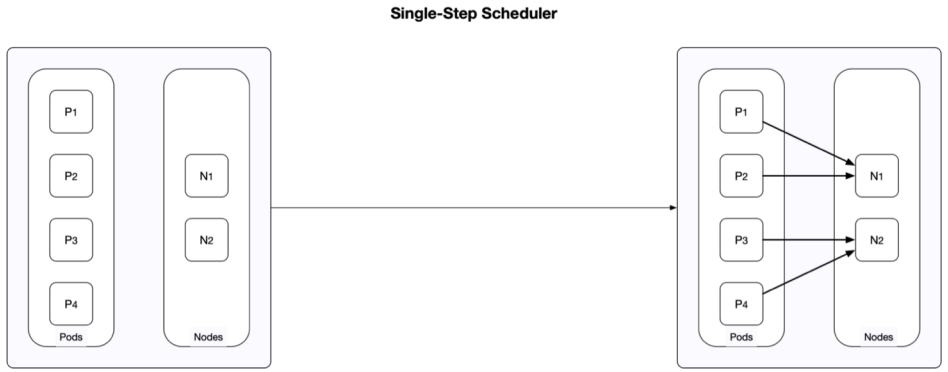
- **Scheduler (kube-scheduler)**: selects nodes for newly created pods to run on, this is called a "placement"
  - Placement: a partial, non-injective assignement of a set of Pods to a set of Nodes.



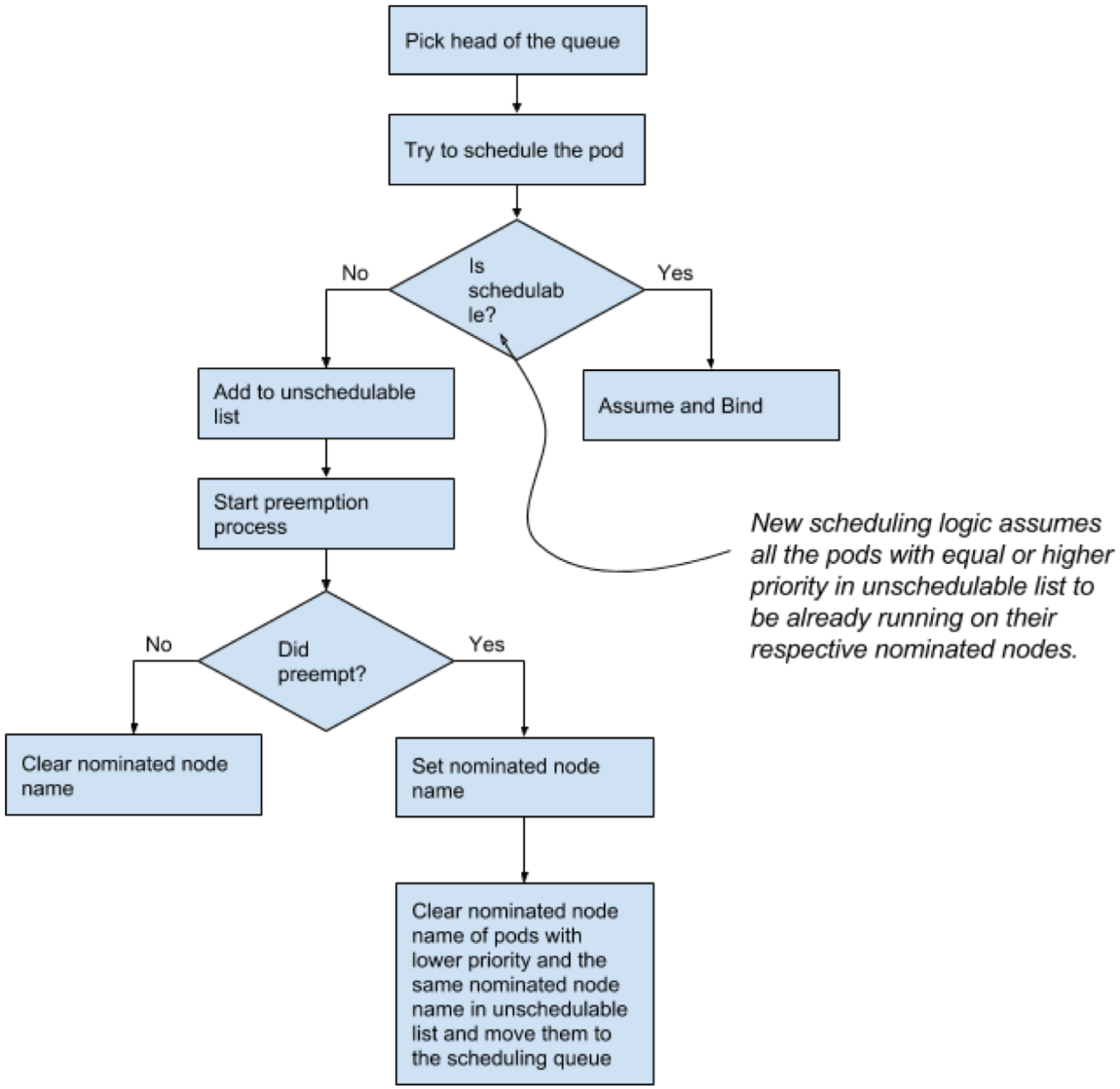
Scheduler decisions

- **Multi-step scheduler**
  - local optimum instead of global optimum



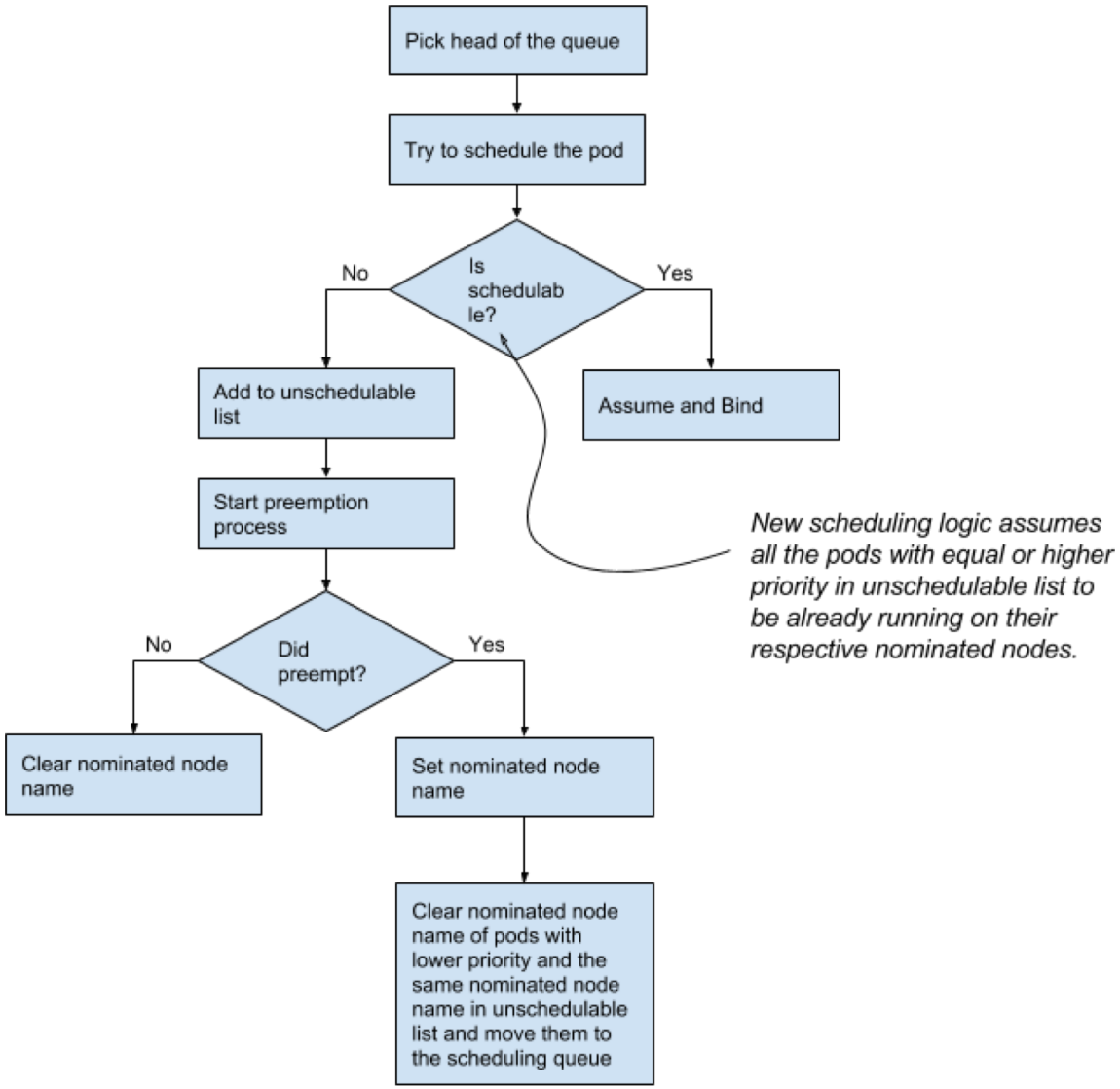


Scheduler preemption

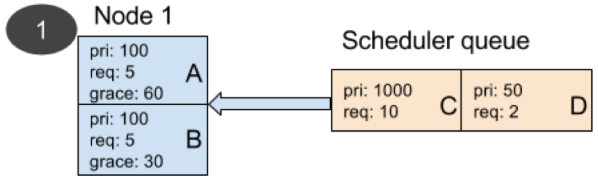


- **Preemption**
  - Lower priority Pods are destroyed to free up resources for higher priority Pods.

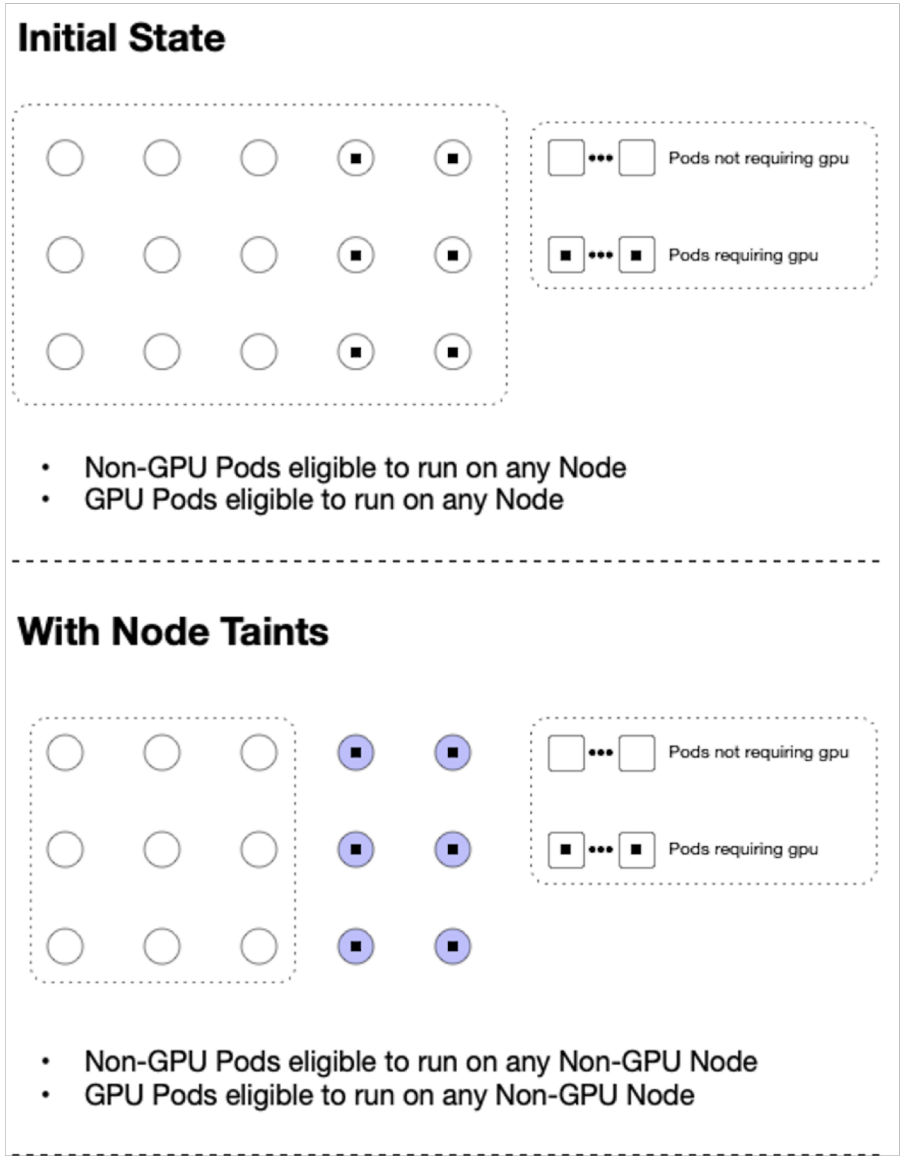
Scheduler preemption



- **Example**
  - 1 node in the cluster with capacity 10 units
  - 2 pods (A,B) running on the node with priority 100 and each using 5 units
  - Scheduler has 2 pods (C,D) in queue. Pod C needs 10 units and priority is 1000. Pod D needs 2 units and it's priority is 50
- Scheduler determines that Pod C has highest priority and destroys (A,B) so it can schedule.



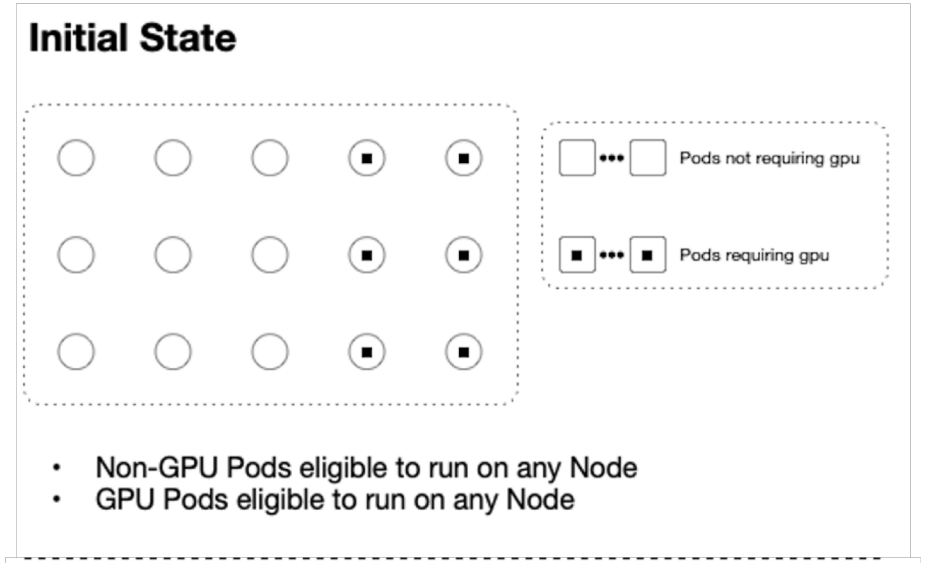
Scheduler placements

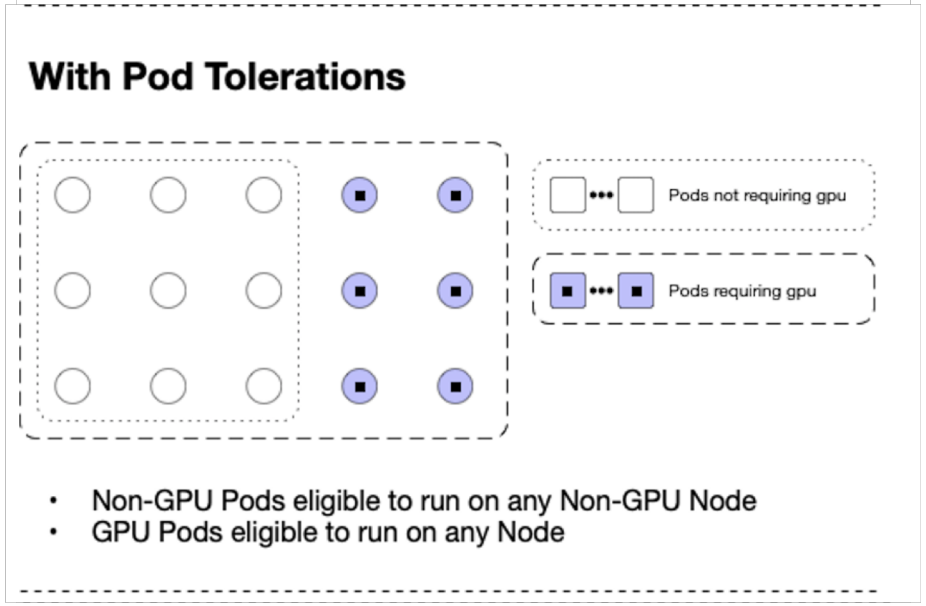


- **Case Study**
  - 9 nodes without GPUs
  - 6 nodes with GPUs
  - We need to schedule Pods that require GPUs only on nodes with GPUs and Pods that do not require GPUs on nodes without GPUs.

- **Node Taints**
  - An option is to use taints which will allow you to stop Pods from being deployed on certain nodes.

Scheduler placements

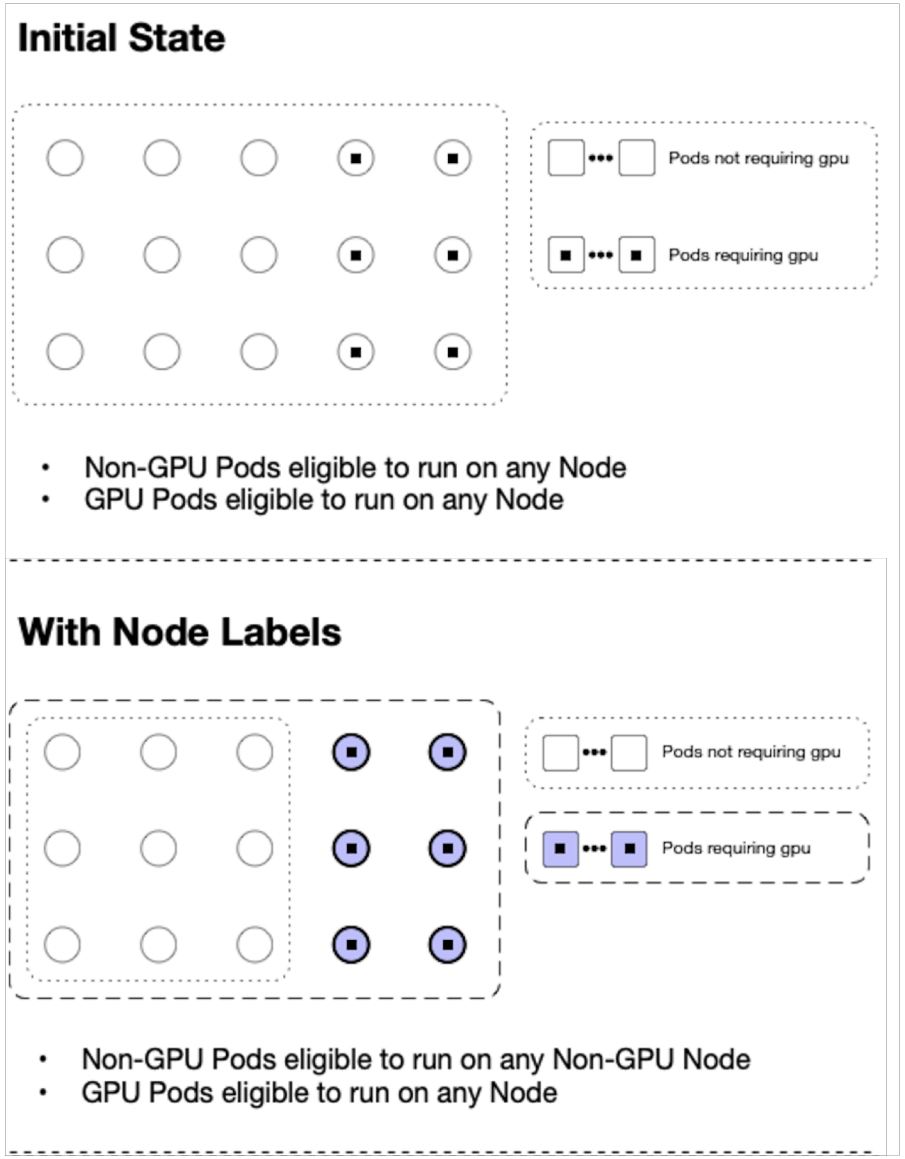




- **Case Study**
  - 9 nodes without GPUs
  - 6 nodes with GPUs
  - We need to schedule Pods that require GPUs only on nodes with GPUs and Pods that do not require GPUs on nodes without GPUs.

- **Tolerations**
  - Tolerations allow us to say specific Pods can run on tainted nodes.

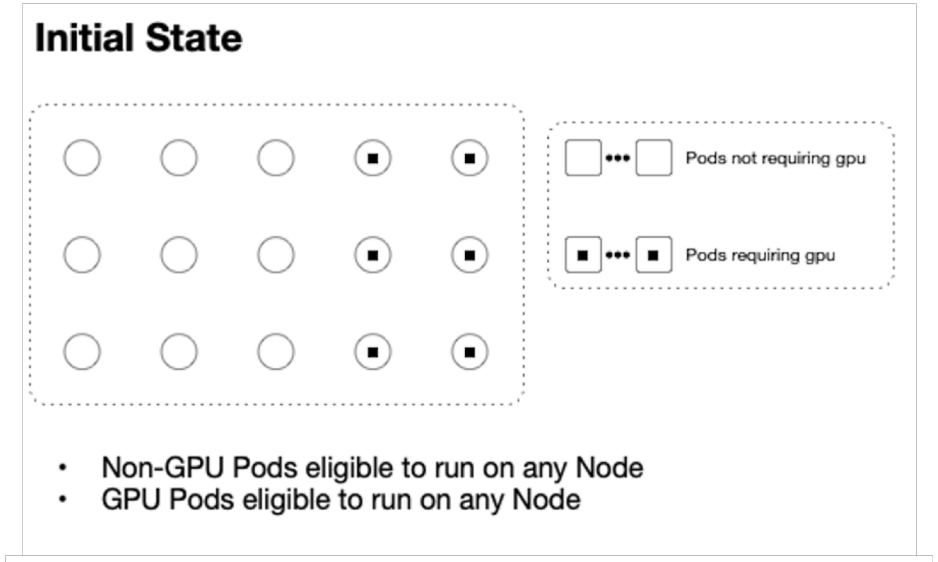
Scheduler placements

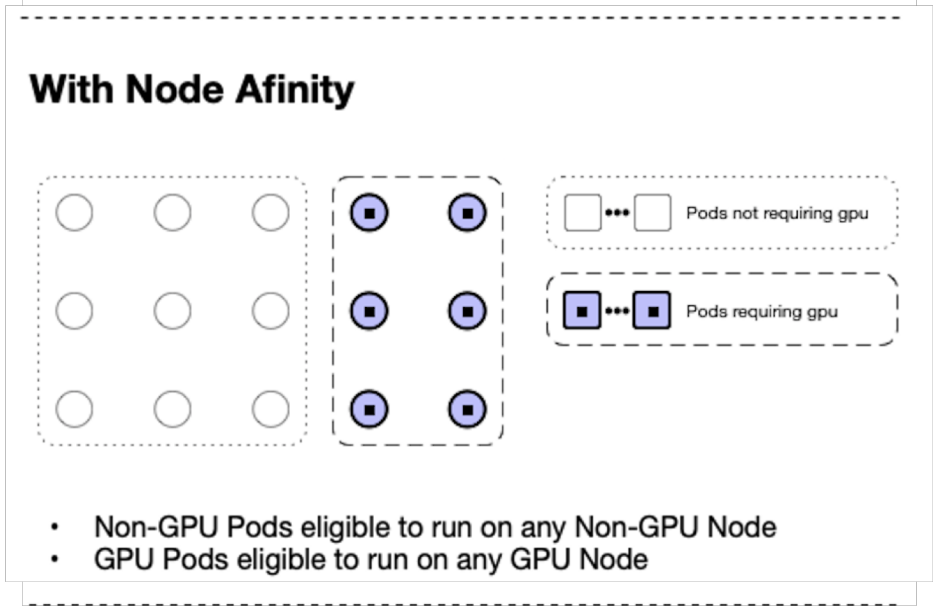


- **Case Study**
  - 9 nodes without GPUs
  - 6 nodes with GPUs
  - We need to schedule Pods that require GPUs only on nodes with GPUs and Pods that do not require GPUs on nodes without GPUs.

- **Node labels**
  - Labels tell Pods which nodes they should be scheduled on.

Scheduler placements





- **Case Study**
  - 9 nodes without GPUs
  - 6 nodes with GPUs
  - We need to schedule Pods that require GPUs only on nodes with GPUs and Pods that do not require GPUs on nodes without GPUs.

- **Node affinity**
  - Similar to node labels but more expressive
    - In/NotIn/Exists/DoesNotExist/Lt,Gt
    - Required or preferred
    - Schedule based on Pods running on other nodes.

Built-in Node Labels

- kubernetes.io/hostname
- failure-domain.beta.kubernetes.io/zone
- failure-domain.beta.kubernetes.io/region
- beta.kubernetes.io/instance-type
- beta.kubernetes.io/os
- beta.kubernetes.io/arch

Kubernetes Advanced POD Scheduling

Kubernetes Scheduling

- Node Selector
- Node Affinity
- Pod Affinity
- Pod Anti Affinity
- 3rd party schedulers

Kubernetes Scheduling

- K8s 1.6 adds more advanced scheduling options, including
  - Node affinity/anti-affinity
    - generalizing the nodeSelector feature
  - Node taints
    - prevent scheduling of pods to nodes unless pod ‘tolerates’ the taint
  - Pod affinity/anti-affinity
    - control relative placement of pods

Node Affinity

The affinity/anti-affinity feature greatly expands the types of constraints you can express.

The key enhancements are:

- More expressive language (not just “AND of exact match”)
- Rule is “soft”/”preference” so if it doesn’t match Pods are still scheduled.

Kubernetes Scheduling

Node Affinity

```
affinity:
 nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
     - matchExpressions:
       - key: "failure-domain.beta.kubernetes.io/zone"
         operator: In
         values: ["us-central1-a"]
```

Kubernetes Scheduling

Node Affinity

```
affinity:
 nodeAffinity:
   preferredDuringSchedulingIgnoredDuringExecution:
     nodeSelectorTerms:
       - matchExpressions:
         - key: "failure-domain.beta.kubernetes.io/zone"
           operator: In
           values: ["us-central1-a"]
```

Kubernetes Scheduling

Weight example

```
affinity:
 nodeAffinity:
  preferredDuringSchedulingIgnoredDuringExecution:
  - weight: 5
    preference:
      - matchExpressions:
        - key: env
          operator: In
          values:
          - dev
   preferredDuringSchedulingIgnoredDuringExecution:
  - weight: 1
    preference:
      - matchExpressions:
        - key: team
          operator: In
          values:
          - engineering
```

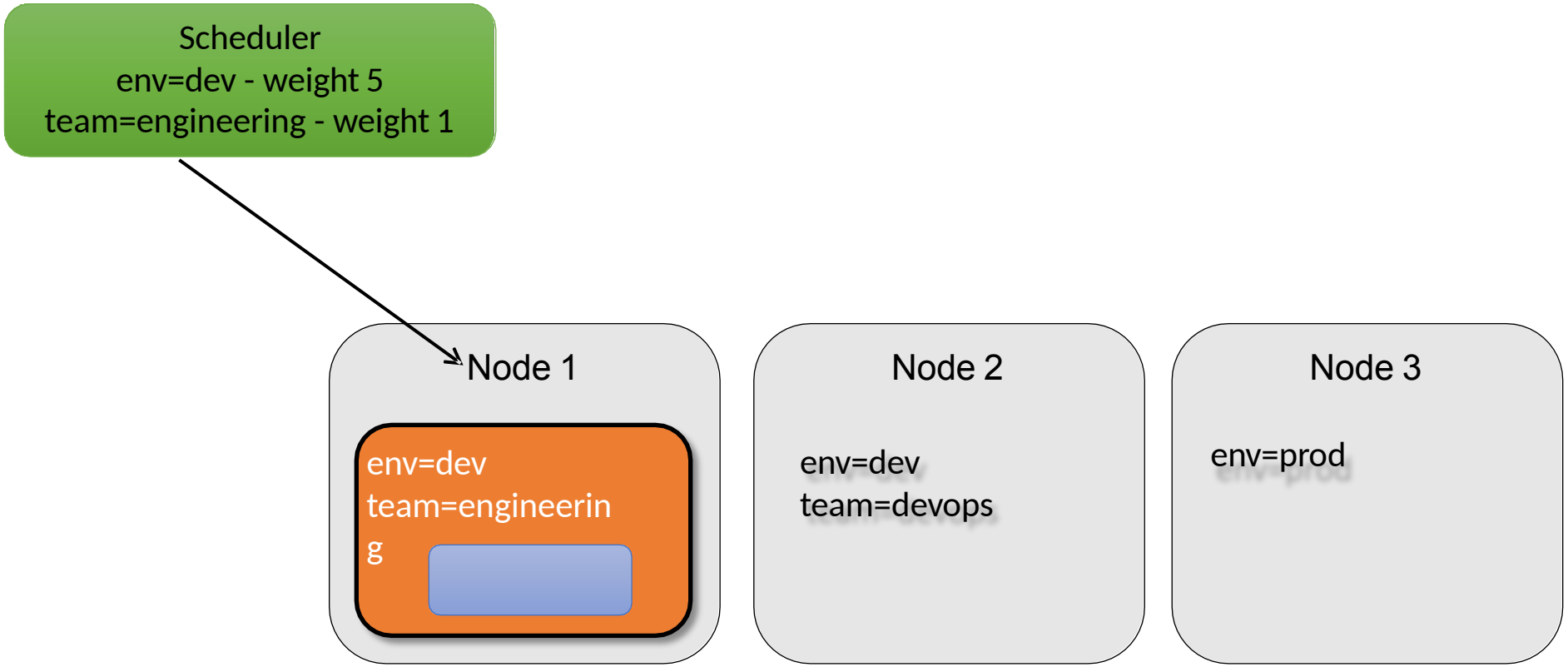
Node Affinity

Kubernetes supports "weighted scheduling"

- The higher this weighting the more weight is given to that rule.
- Scoring is done by summarizing weighting per node.
  - 2 rules weights 1 and 5
  - If both match = score of 6
  - If only 1 rule matches = score of 1
- Node with highest total weight score receives Pod.

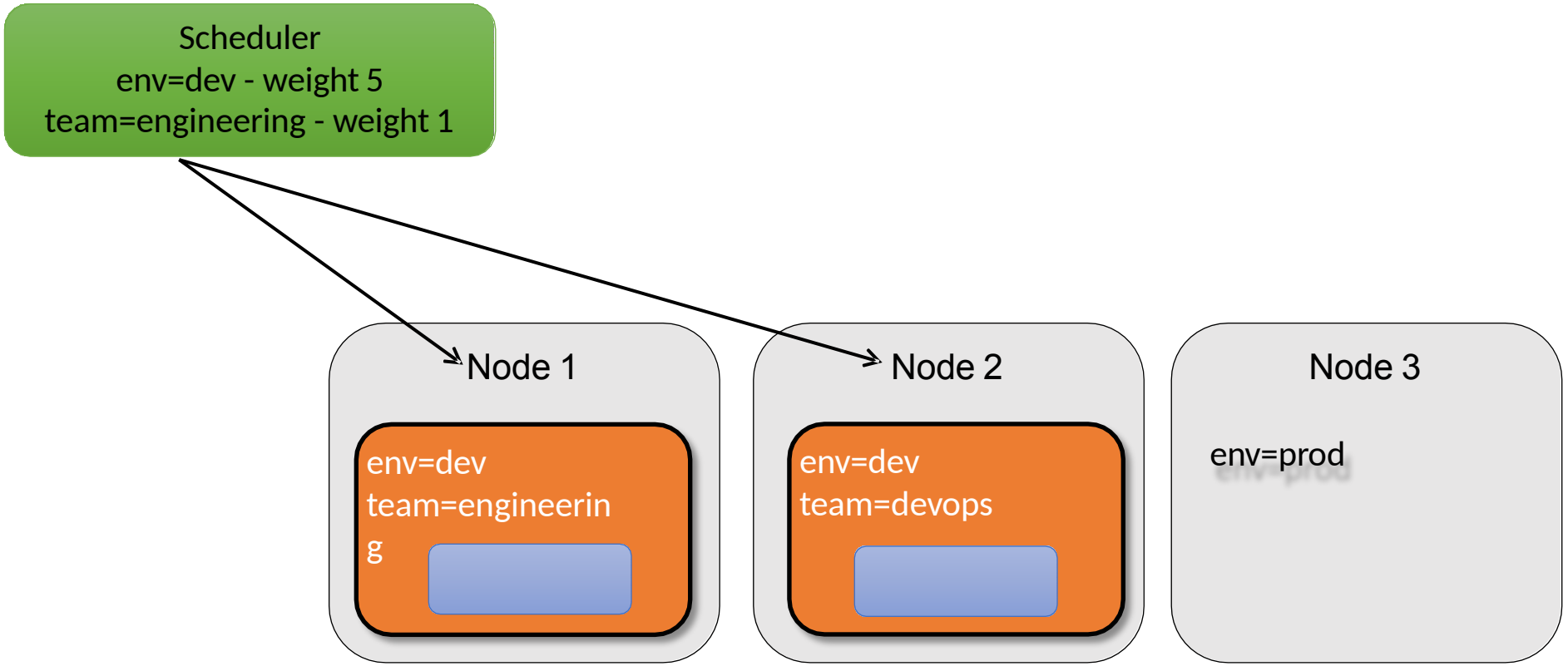
Node Affinity: Weight

- Schedule Pods onto nodes that have labels
  - env = dev
  - team = engineering



Node Affinity: Weight

- Schedule Pods onto nodes that have labels
  - env = dev
  - team = engineering



Kubernetes Scheduling

Node Anti-Affinity

```
affinity:
 nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
     - matchExpressions:
       - key: "failure-domain.beta.kubernetes.io/zone"
         operator: NotIn
         values: ["us-central1-a"]
```

Operators

- Valid operators
  - In
  - NotIn
  - Exists
  - DoesNotExist
  - Gt (Greater Than)
  - Lt (Less Than)

Node Taints

- Allows you to mark (“taint”) a node
  - No pods can be scheduled onto tainted nodes
  - Useful when all or most PODs should not be scheduled on a node
  - Mark your master node as schedulable only by Kubernetes system components
  - Keep regular pods away from nodes that have special hardware so as to leave room for pods that need the special hardware.

Node Tolerations

- To allow a POD to be scheduled onto a ‘tainted’ node it must have:

```
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

Pod Affinity

- Define how pods should be placed relative to one another
- Spread or pack pods within a service or relative to pods in other services

```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: service
          operator: In
          values: [“S1”]
      topologyKey: failure-domain.beta.kubernetes.io/zone
```

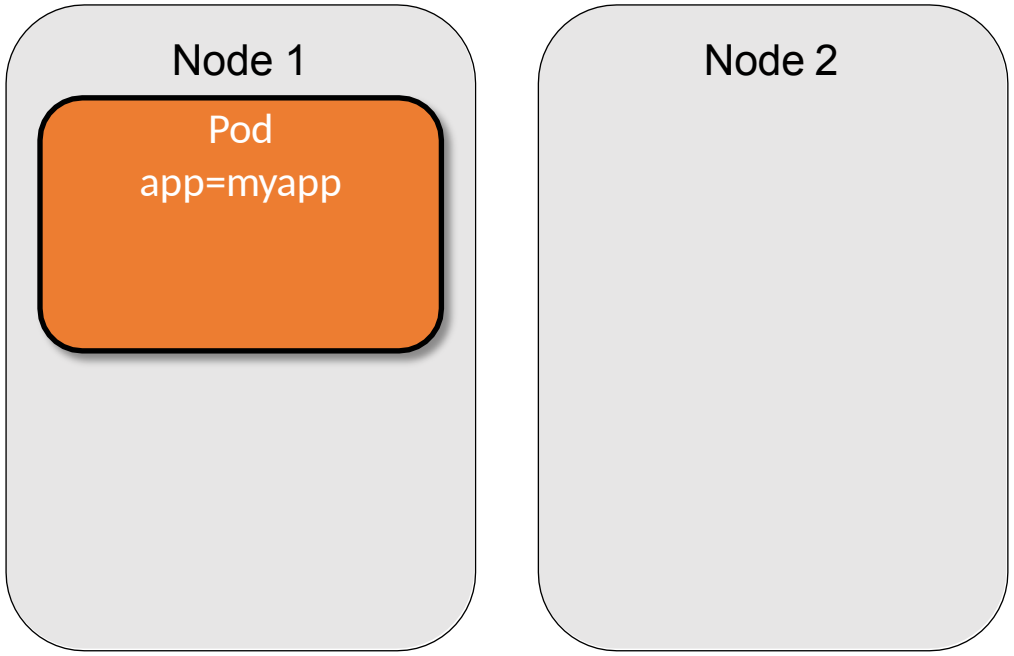
Pod Affinity

- Define how pods should be placed relative to one another
- Spread or pack pods within a service or relative to pods in other services

```
affinity:
  podAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: service
          operator: In
          values: [“S1”]
      topologyKey: failure-domain.beta.kubernetes.io/zone
```

Pod Affinity

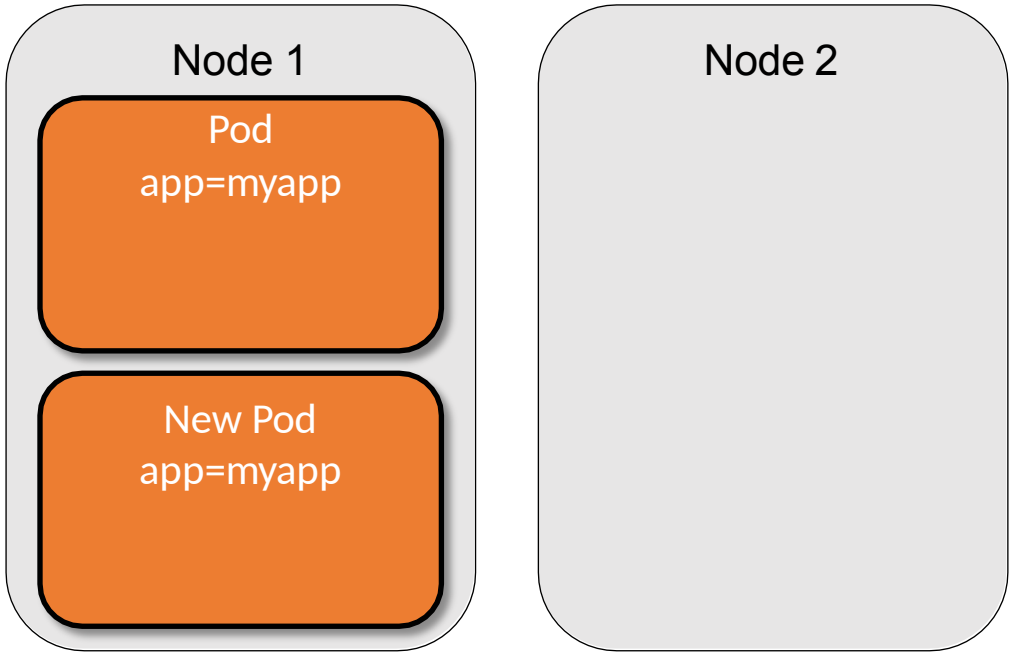
- Schedule Pods onto nodes that have same labels
  - app=myapp
  - operator: In
  - kubernetes.io/hostname




```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: "app"
          operator: In
          values:
          - myapp
      topologyKey: "kubernetes.io/hostname"
```

Pod Affinity

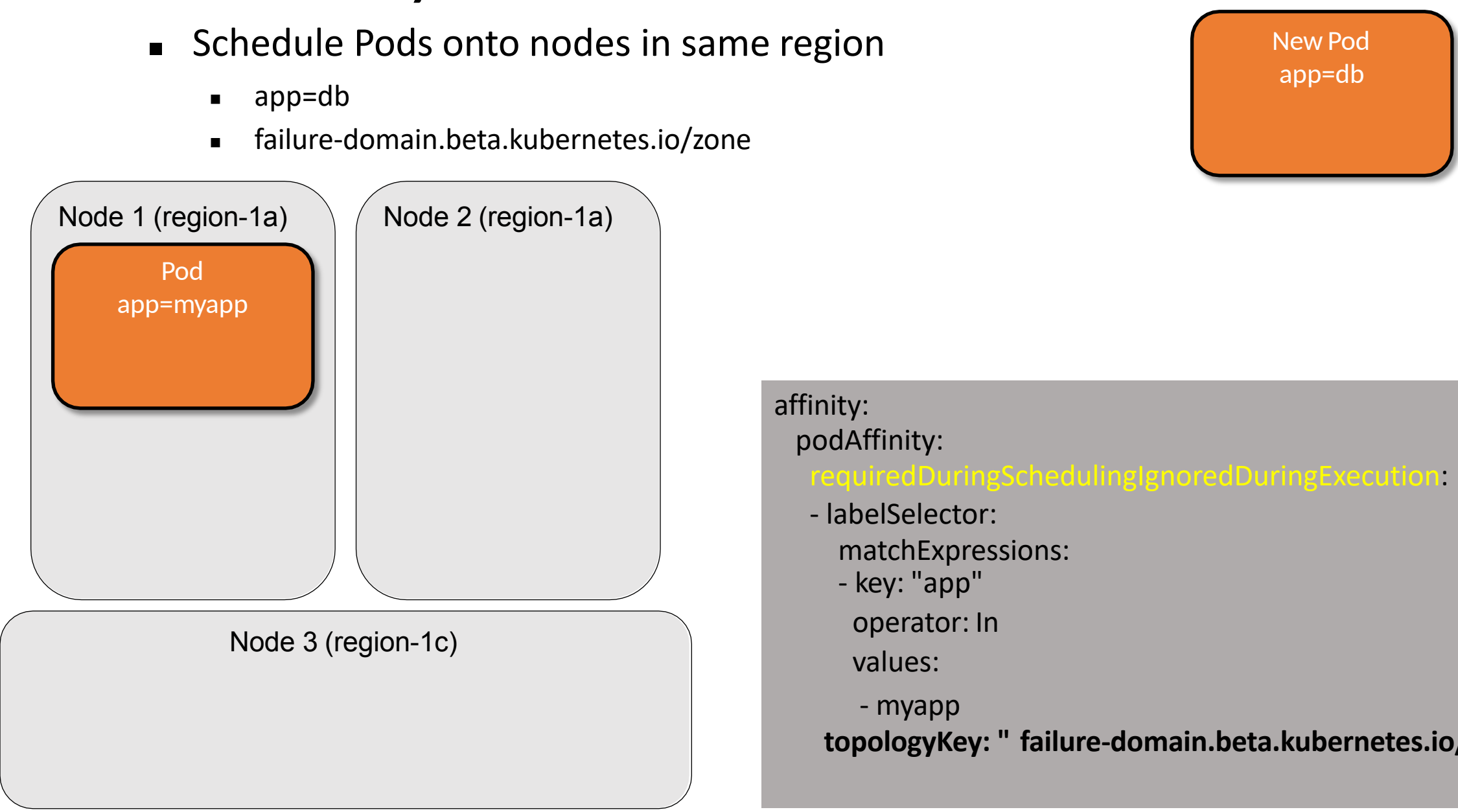
- Schedule Pods onto nodes that have same labels
  - app=myapp
  - operator: In
  - kubernetes.io/hostname




```
affinity:
  podAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: "app"
          operator: In
          values:
          - myapp
      topologyKey: "kubernetes.io/hostname"
```

Pod Affinity

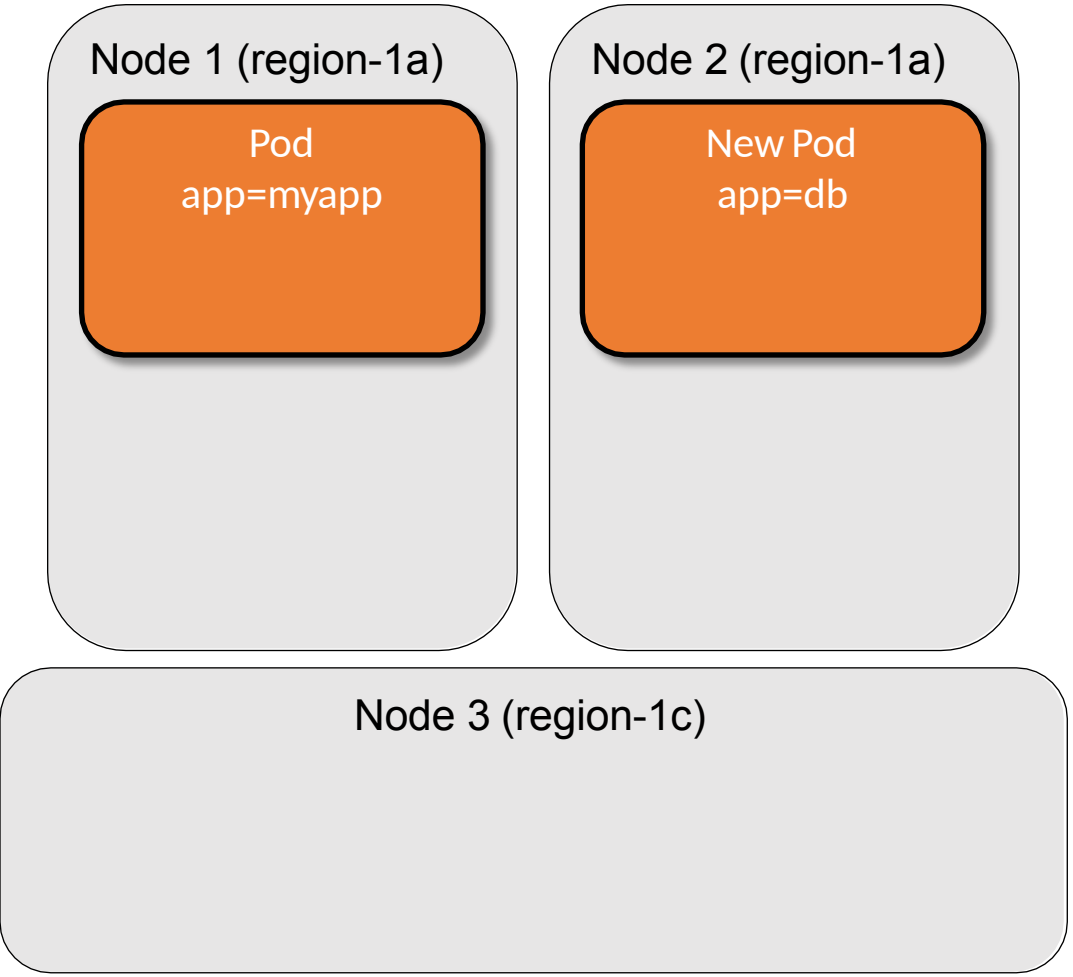
- Schedule Pods onto nodes in same region
  - app=db
  - failure-domain.beta.kubernetes.io/zone




```
affinity:
  podAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   - labelSelector:
       matchExpressions:
       - key: "app"
         operator: In
         values:
         - myapp
     topologyKey: " failure-domain.beta.kubernetes.io/zone"
```

Pod Affinity

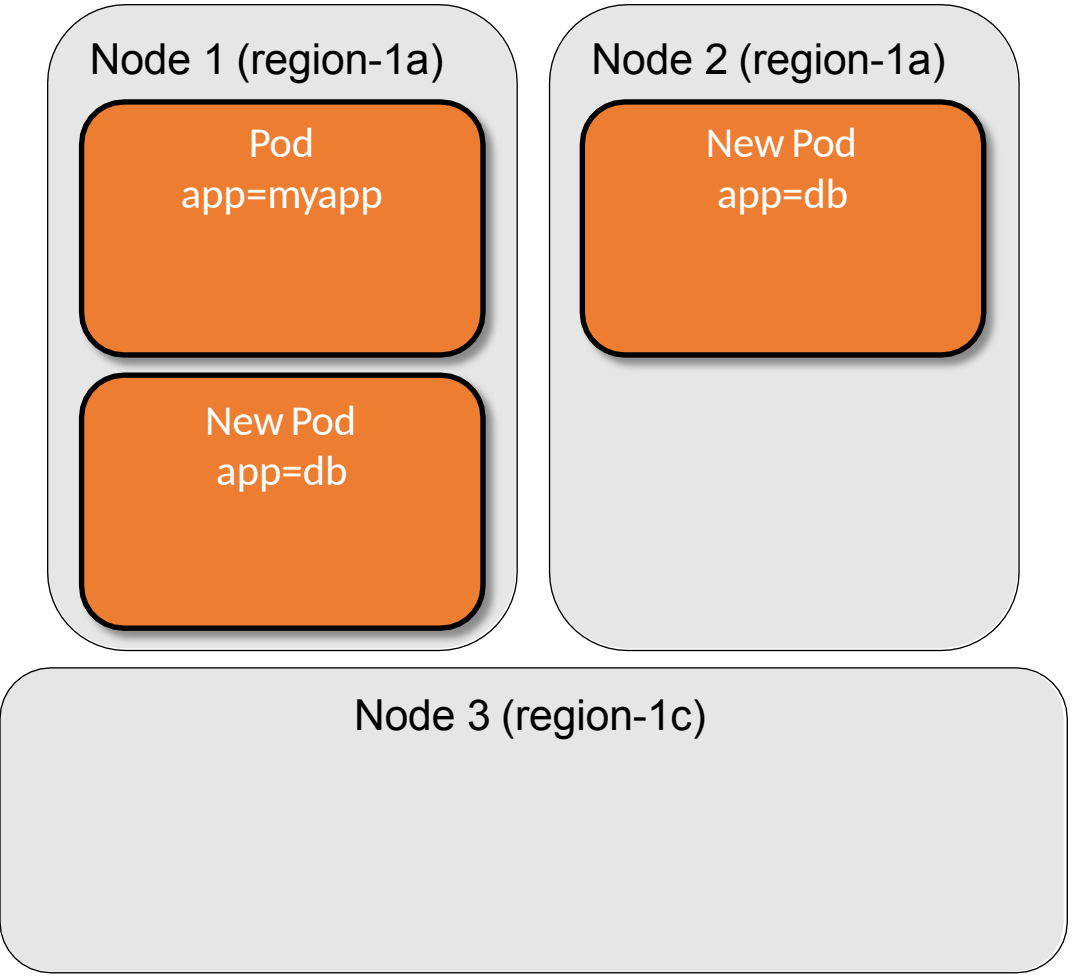
- Schedule Pods onto nodes in same region
  - app=db
  - failure-domain.beta.kubernetes.io/zone




```
affinity:
  podAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   - labelSelector:
       matchExpressions:
       - key: "app"
         operator: In
         values:
         - myapp
     topologyKey: " failure-domain.beta.kubernetes.io/zone"
```

Pod Affinity

- Schedule Pods onto nodes in same region
  - app=db
  - failure-domain.beta.kubernetes.io/zone




```
affinity:
  podAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   - labelSelector:
       matchExpressions:
       - key: "app"
         operator: In
         values:
         - myapp
     topologyKey: " failure-domain.beta.kubernetes.io/zone"
```

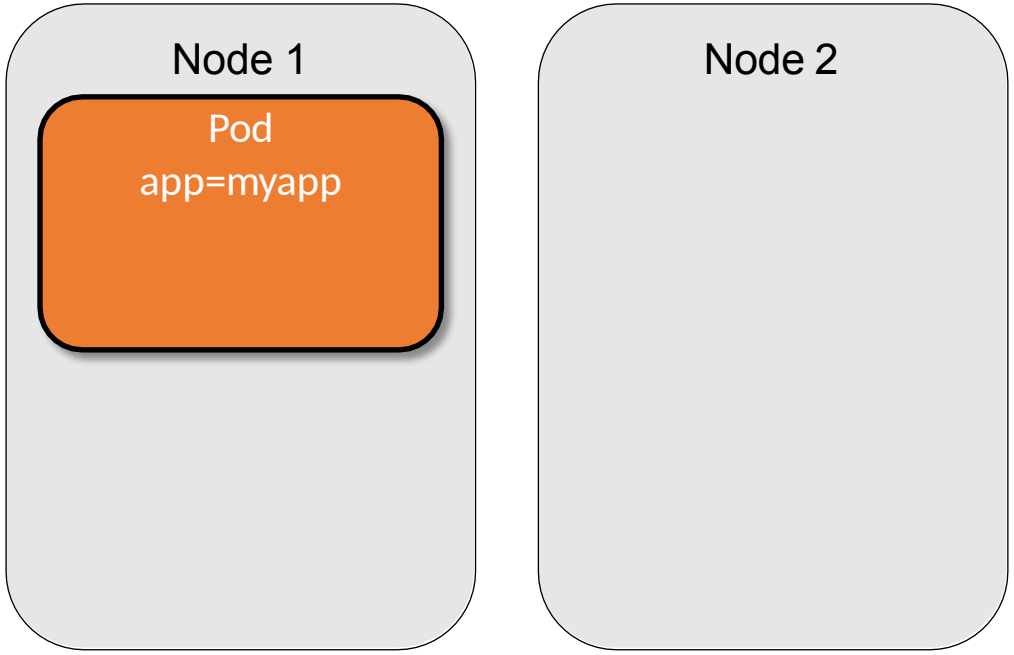
Pod Anti-Affinity

- Define how pods should be placed relative to one another
- PODs of different services run on different nodes.

```
affinity:
  podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: service
          operator: In
          values: [“S1”]
      topologyKey: kubernetes.io/hostname
```

Pod Anti-Affinity

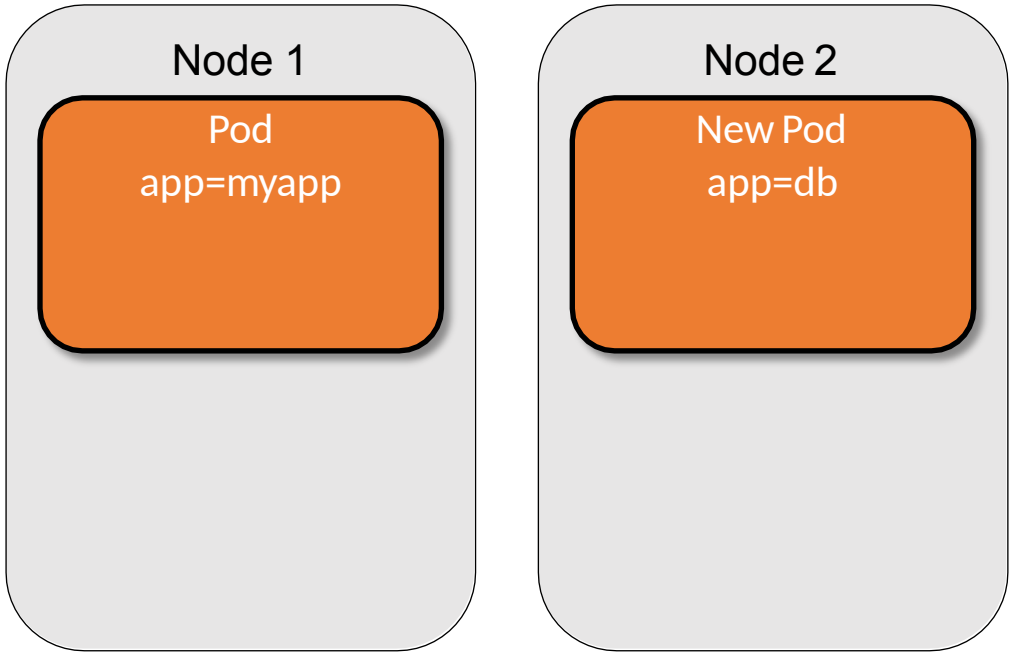
- Do not schedule Pods onto nodes with matching labels.
  - app=db
  - operator: In
  - kubernetes.io/hostname




```
affinity:
  podAntiAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   - labelSelector:
      matchExpressions:
      - key: "app"
        operator: In
        values:
         - myapp
     topologyKey: "kubernetes.io/hostname"
```

Pod Anti-Affinity

- Do not schedule Pods onto nodes with matching labels.
  - app=db
  - operator: In
  - kubernetes.io/hostname




```
affinity:
  podAntiAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   - labelSelector:
      matchExpressions:
      - key: "app"
        operator: In
        values:
         - myapp
     topologyKey: "kubernetes.io/hostname"
```

Affinity Lab02

- Set label on nodes
- Update POD YAML with nodeSelector
- Apply Affinity/Anti-Affinity YAML
- Confirm PODs are scheduled as expected

Taint Lab

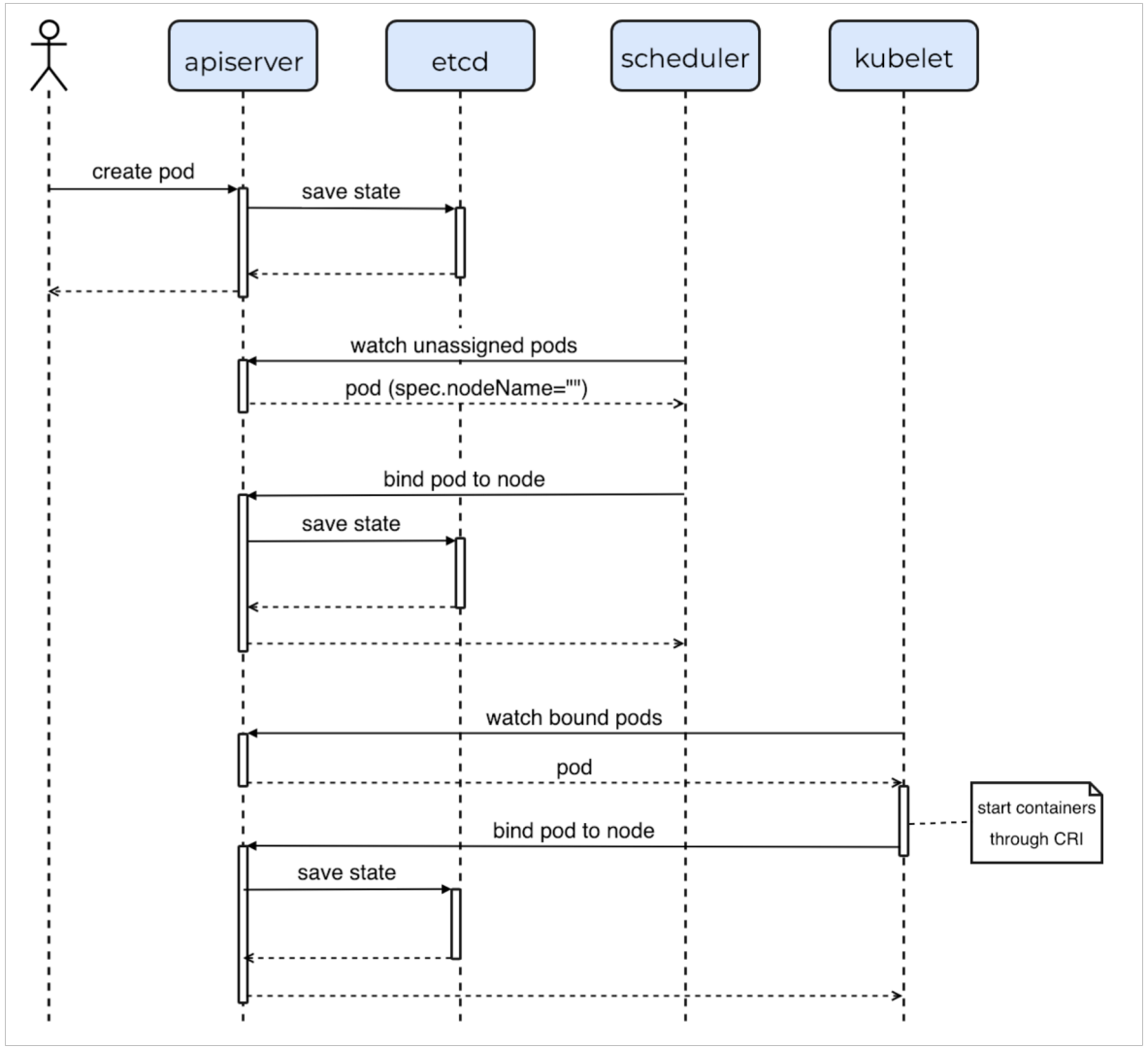
- Set Taint on node
- Create deployment without toleration
- Update YAML with toleration
- Create deployment with toleration
- Confirm PODs scheduled as expected

Custom Scheduler(s)

Custom Schedulers

- Create your own schedulers
- Run alongside or replace default scheduler
- Beta in Kubernetes

Custom Schedulers



1. Pod created and state saved in etcd without node
2. Scheduler notices new Pod without node
3. Finds node with available resources
4. Tells apiserver to bind pod to node and apiserver updates etcd with node assignment
5. Kubelet sees new pod request and creates it on node.

Scheduler flow

1. A loop to watch the unbound pods in the cluster through querying the apiserver
2. Custom logic that finds the best node for a pod.
3. A request to the bind endpoint on the apiserver.

Custom Scheduler Example

- Custom scheduler defined in YAML
- Default scheduler ignores PODs

```
apiVersion: v1
kind: Pod
metadata:
    name: nginx
    labels:
      app: nginx
spec:
  schedulerName: my-scheduler
  containers:
  - name: nginx
    image: nginx:1.10
```

Custom Schedulers

- Can be written in any language
- Simple or complex
- 3rd party schedulers available.

Custom Scheduler Code (Bash)

```
#!/bin/bash
SERVER='localhost:8001'
while true;
do
  for PODNAME in $(kubectl --server $SERVER get pods -o json | jq '.items[] | select(.spec.schedulerName == "my-scheduler") | select(.spec.nodeName == null) | .metadata.name' | tr -d '"')
;
  do
    NODES=($(kubectl --server $SERVER get nodes -o json | jq '.items[].metadata.name' | tr -d '"'))
    NUMNODES=${#NODES[@]}
    CHOSEN=${NODES[$[ $RANDOM % $NUMNODES ]]}
    curl --header "Content-Type:application/json" --request POST --data '{"apiVersion":"v1", "kind": "Binding", "metadata": {"name": "'$PODNAME'"}, "target": {"apiVersion": "v1", "kind"
: "Node", "name": "'$CHOSEN'"}}' http://$SERVER/api/v1/namespaces/default/pods/$PODNAME/binding/
    echo "Assigned $PODNAME to $CHOSEN"
  done
  sleep 1
done
```

Custom Scheduler v1 (Go)

```
watch, err := s.clientset.CoreV1().Pods("").Watch(metav1.ListOptions{
    FieldSelector: fmt.Sprintf("spec.schedulerName=%s,spec.nodeName=", schedulerName),
})

 ...

for event := range watch.ResultChan() {
   if event.Type != "ADDED" {
                continue
   }
   p := event.Object.(*v1.Pod)
   fmt.Println("found a pod to schedule:", p.Namespace, "/", p.Name)

...
}
```

Custom Scheduler v1 (Go)

```
nodes, _ := s.clientset.CoreV1().Nodes().List(metav1.ListOptions{})
return &nodes.Items[rand.Intn(len(nodes.Items))], nil
```

- Find a fitting node (random)
- We are querying the apiserver for list of nodes on every schedule event.
- Bad for performance.

Custom Scheduler v1 (Go)

```
s.clientset.CoreV1().Pods(p.Namespace).Bind(&v1.Binding{
    ObjectMeta: metav1.ObjectMeta{
          Name:       p.Name,
    Namespace:       p.Namespace,
    },
   Target: v1.ObjectReference{
         APIVersion:  "v1",
         Kind:        "Node",
         Name:        randomNode.Name,
   },
})
```

- After finding node we use the 'Bind' function to tell the apiserver so it can update etcd and have the kubelet create the Pod.

Custom Scheduler v1 (Go)

```
timestamp := time.Now().UTC()
s.clientset.CoreV1().Events(p.Namespace).Create(&v1.Event{
        Count:          1,
        Message:        message,
        Reason:         "Scheduled",
        LastTimestamp: metav1.NewTime(timestamp),
        FirstTimestamp: metav1.NewTime(timestamp),
        Type:           "Normal",
        Source: v1.EventSource{
                Component: schedulerName,
        },
        InvolvedObject: v1.ObjectReference{
                Kind:      "Pod",
                Name:      p.Name,
                Namespace: p.Namespace,
                UID:       p.UID,
        },
        ObjectMeta: metav1.ObjectMeta{
                GenerateName: p.Name + "-",
        },
})
```

- Add scheduled events so we can track when Pod is scheduled.

Custom Scheduler v2 (Go)

- Improve performance:
  - SharedInformers provide hooks to receive notifications of adds, updates, and deletes for a particular resource. They also provide convenience functions for accessing shared caches.

```
nodeInformer := factory.Core().V1().Nodes()
nodeInformer.Informer().AddEventHandler(cache.ResourceEventHandlerFuncs{
        AddFunc: func(obj interface{}) {
                node := obj.(*v1.Node)
                log.Printf("New Node Added to Store: %s", node.GetName())
        },
})
factory.Start(quit)
return nodeInformer.Lister()
```

Custom Scheduler v2 (Go)

- Update 'Bind' code so it looks at cached node list.

```
nodes, err := s.nodeLister.List(labels.Everything())
return nodes[rand.Intn(len(nodes))], nil
```

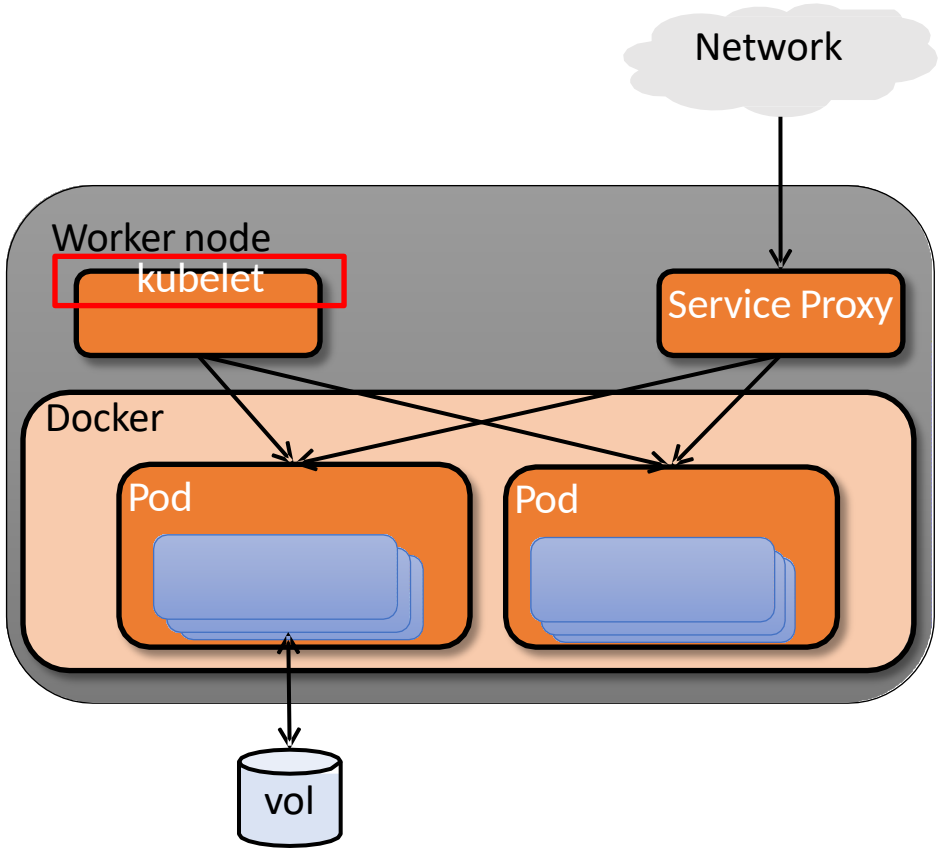
Custom Scheduler Lab03

- Deploy custom Go scheduler
- Update YAML manifest to use custom scheduler
- Deploy application and confirm it uses custom scheduler

Kubelet Configuration

Kubelet Architecture

- **kubelet**: local K8s agent that is responsible for operations on the node, including
  - Watching for pod assignments
  - Mounting pod required volumes
  - Running a pod's containers
  - Executing container liveness probes
  - Reporting pod status to system
  - Reporting node status to system



Kubelet Configuration

- Kubelet requires PodSpec
- PodSpec: YAML or JSON object describes a POD
- Manages PODs created with Kubernetes

Kubelet Configuration

There are 4 ways a container manifest can be provided to the Kubelet.

- PodSpec from API server (most common)
- File: Path passed as a flag on the command line.
- HTTP endpoint: passed as a parameter on command line.
- HTTP server: Listen for HTTP and respond to simple API to submit new manifests

Kubelet Path

```
root     15375  1.9  1.9 427928 79988 ?        Ssl  02:07   0:01 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernete
s/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --pod-manifest-path=/etc/kubernetes/manifests --allo
w-privileged=true --network-plugin=cni --cni-conf-dir=/etc/cni/net.d --cni-bin-dir=/opt/cni/bin --cluster-dns=10.96.0.1
0 --cluster-domain=cluster.local --authorization-mode=Webhook --client-ca-file=/etc/kubernetes/pki/ca.crt --cadvisor-po
rt=0 --rotate-certificates=true --cert-dir=/var/lib/kubelet/pki
```

Kubelet API

```
ubuntu@ip-10-0-100-134:~$ ps auxww|grep -i [k]ubelet
root     12691  1.7  2.3 513000 93516 ?        Ssl  01:43   2:57 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf
 --pod-manifest-path=/etc/kubernetes/manifests --allow-privileged=true --network-plugin=cni --cni-conf-dir=/etc/cni/net.d --cni-bin-dir=/opt/cni/bin --cluster-dns=10.96.0.10 --cluster-d
omain=cluster.local --authorization-mode=Webhook --client-ca-file=/etc/kubernetes/pki/ca.crt --cadvisor-port=0 --rotate-certificates=true --cert-dir=/var/lib/kubelet/pki
root     13140  2.0  6.7 372616 271444 ?       Ssl  01:43   3:28 kube-apiserver --tls-private-key-file=/etc/kubernetes/pki/apiserver.key --admission-control=Initializers,NamespaceLifecy
cle,LimitRanger,ServiceAccount,DefaultStorageClass,DefaultTolerationSeconds,NodeRestriction,ResourceQuota --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname --requestheade
r-group-headers=X-Remote-Group --service-cluster-ip-range=10.96.0.0/12 --client-ca-file=/etc/kubernetes/pki/ca.crt --requestheader-username-headers=X-Remote-User --tls-cert-file=/etc/ku
bernetes/pki/apiserver.crt --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt --insecure-port=0 --requestheader-extra-headers-prefix=X-Remote-Extra- --secure-
port=6443 --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt --kubelet-client-key=/etc/kubernetes/
pki/apiserver-kubelet-client.key --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key --enable-bootstrap-token-auth=true --allow-privileged=true --requestheader-allowed-na
mes=front-proxy-client --advertise-address=10.0.100.134 --service-account-key-file=/etc/kubernetes/pki/sa.pub --authorization-mode=Node,RBAC --etcd-servers=http://127.0.0.1:2379
```

Kubelet Lab

- Install Kubelet in standalone mode
- Deploy v1 application YAML
- Update to v2 application YAML

ConfigMap

ConfigMap

- Many applications require configuration via:
  - Config Files
  - Command-Line Arguments
  - Environment Variables
- These need to be decoupled from images to keep portable
- ConfigMap API provides mechanisms to inject containers with configuration data
- Store individual properties or entire config files/JSON blobs
- Key-Value Pairs

ConfigMap

- Not meant for sensitive information
- PODs or controllers can use ConfigMaps

1. Populate the value of environment variables
2. Set command-line arguments in a container
3. Populate config files in a volume

```
kind: ConfigMap
apiVersion: v1
metadata:
    creationTimestamp: 2016-02-18T19:14:38Z
    name: example-config
    namespace: default
data:
    example.property.1: hello
    example.property.2: world
    example.property.file: |-
        property.1=value-1
        property.2=value-2
        property.3=value-3
```

ConfigMap from directory

- 2 files in docs/user-guide/configmap/kubectl
  - game.properties
  - ui.properties

game.properties

```
enemies=aliens
lives=3
enemies.cheat=true
enemies.cheat.level=noGoodRotten
secret.code.passphrase=UUDDLRLRBABAS
secret.code.allowed=true
secret.code.lives=30
```

ui.properties

```
color.good=purple
color.bad=yellow
allow.textmode=true
how.nice.to.look=fairlyNice
```

ConfigMap from directory

```
kubectl create configmap game-config --from-file=docs/user-guide/configmap/kubectl
```

```
kubectl describe configmaps game-config
Name: game-config
Namespace: default
Labels: <none>
Annotations: <none>
Data
====
game.properties: 121 bytes
ui.properties: 83 bytes
```

ConfigMap from directory

```
kubectl get configmaps game-config -o yaml
```

```
apiVersion: v1
data:
    game.properties: |-
        enemies=aliens
        lives=3
        …
ui.properties: |
       color.good=purple
       …
kind: ConfigMap
metadata:
    creationTimestamp: 2016-02-18T18:34:05Z
    name: game-config
    namespace: default
    resourceVersion: "407"-
    selfLink: /api/v1/namespaces/default/configmaps/game-config
    uid: 30944725-d66e-11e5-8cd0-68f728db1985
```

ConfigMap from files

```
kubectl create configmap \
game-config-2 \
--from-file=file1 \
--from-file=file2
```

```
kubectl get configmaps \
game-config-2 \
-o yaml
```

```
apiVersion: v1
data:
    game.properties: |-
        enemies=aliens
        lives=3
        …
ui.properties: |
       color.good=purple
       …
kind: ConfigMap
metadata:
    creationTimestamp: 2016-02-18T18:52:05Z
    name: game-config-2
    namespace: default
    resourceVersion: ”516"-
    selfLink: /api/v1/namespaces/default/configmaps/game-config-2
    uid: b4952dc3-d670-11e5-8cd0-68f728db1985
```

ConfigMap options

- --from-file=/path/to/directory
- --from-file=/path/to/file1 (/path/to/file2)
- Literal key=value: --from-literal=special.how=very

ConfigMap in PODs

- Populate Environment Variables

```
apiVersion: v1
kind: ConfigMap
metadata:
    name: special-config
    namespace: default
data:
    special.how: very
    special.type: charm
```

OUTPUT

```
SPECIAL_LEVEL_KEY=very
SPECIAL_TYPE_KEY=charm
```

```
apiVersion: v1
kind: Pod
metadata:
    name: dapi-test-pod
spec:
  containers:
      - name: test-container
        image: busybox
        command: [“/bin/sh”, “-c, “env” ]
        env:
            - name: SPECIAL_LEVEL_KEY
              valueFrom:
                 configMapKeyRef:
                     name: special-config
                     key: special.how
  restartPolicy: Never
```

ConfigMap Lab

- Create a ConfigMap to configure Redis as a cache
- Look at ConfigMap in YAML format
- Create a POD to use ConfigMap
- Confirm POD used ConfigMap settings
- Setup Node.js web app
- Configure Nginx reverse proxy