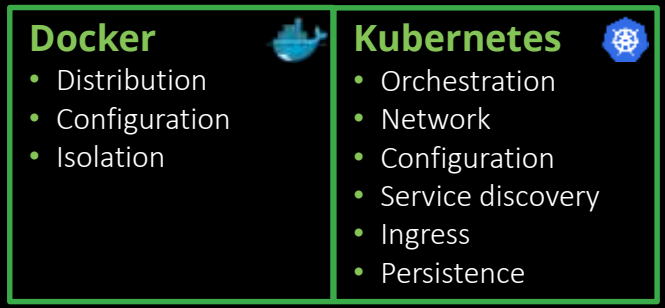
Scheduling in Kubernetes
October, 2017

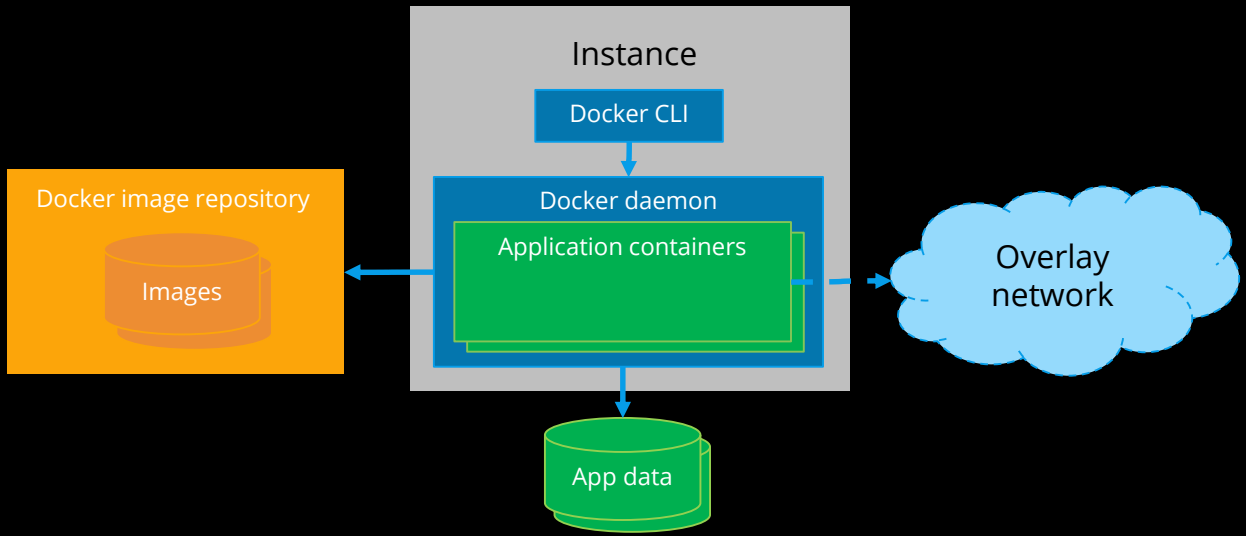
What to look for

- Kubernetes overview
- Scheduling algorithm
- Scheduling controls
- Advanced scheduling techniques
- Examples and use cases

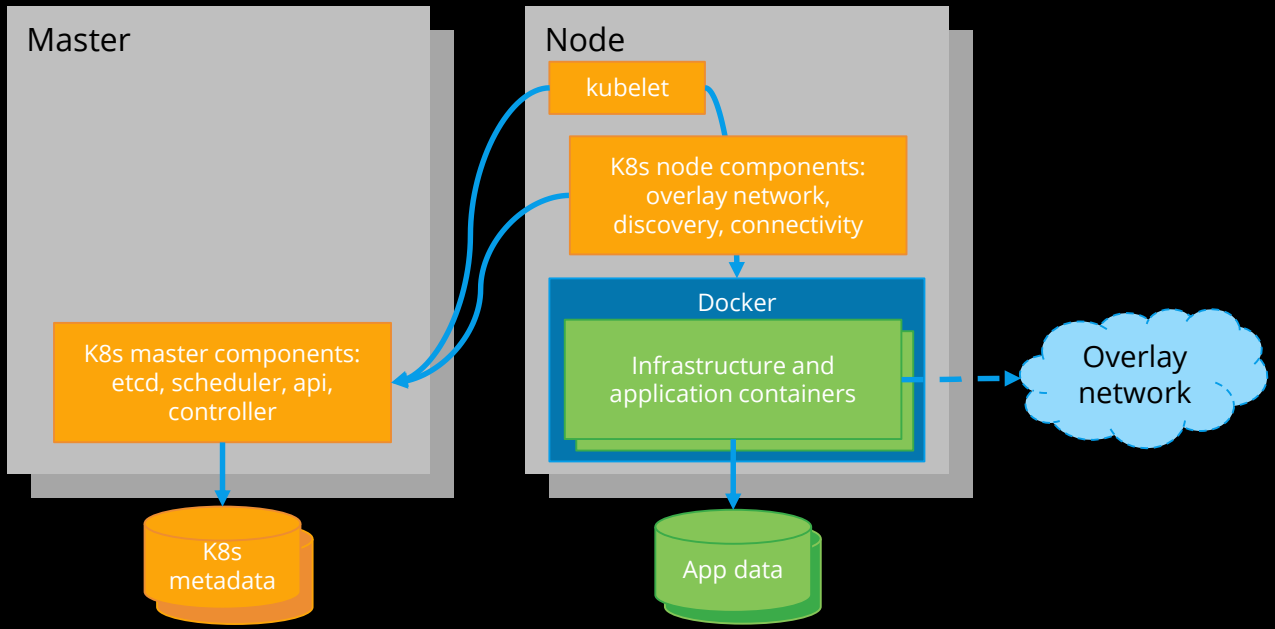
Kubernetes | Technology stack



Docker | Architecture

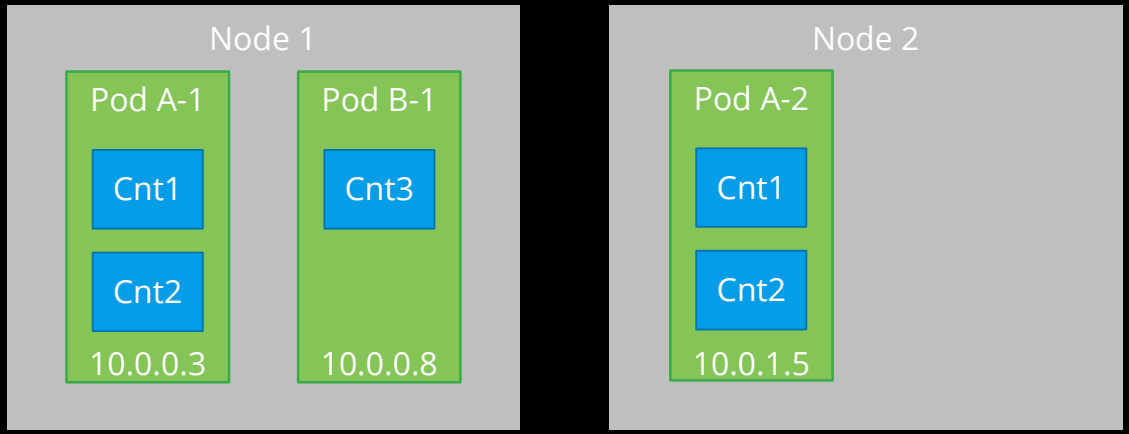


Kubernetes | Architecture

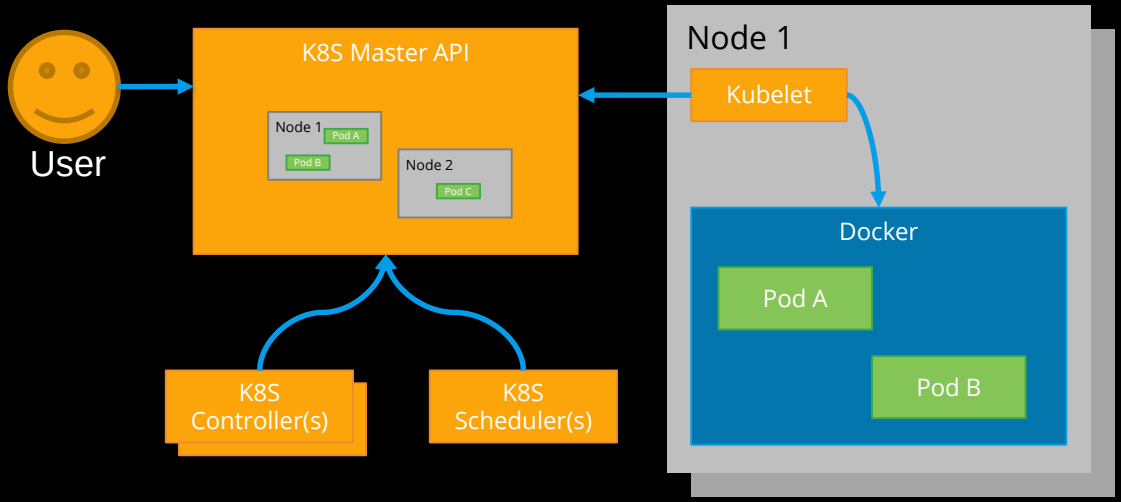


kublr

Kubernetes | Nodes and Pods

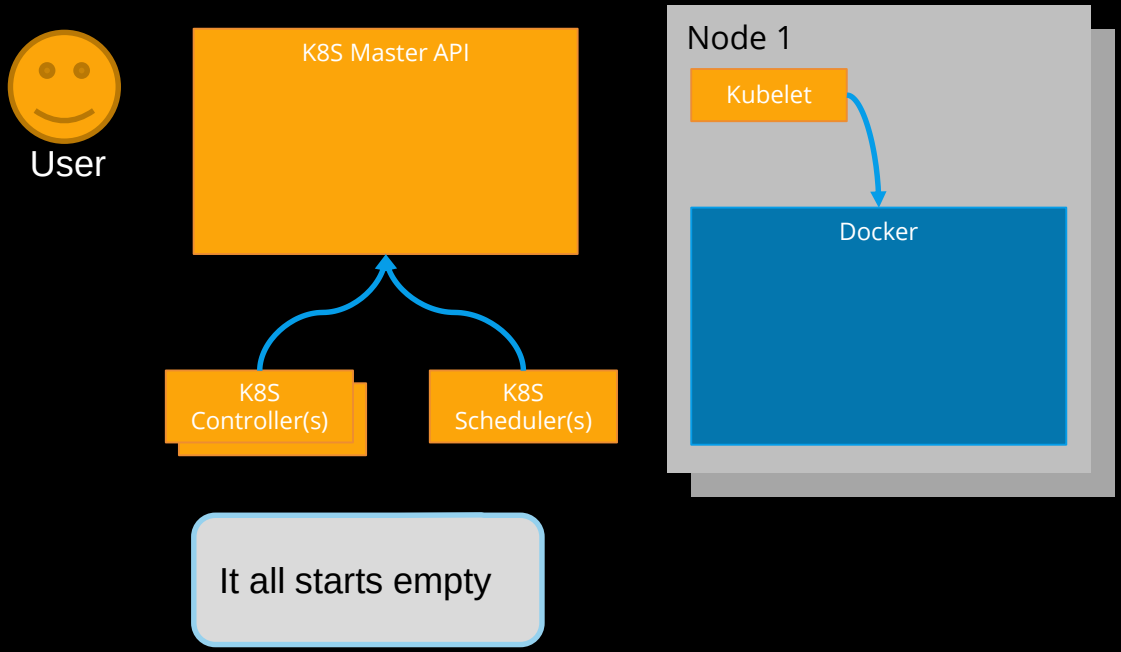


Kubernetes | Container Orchestration



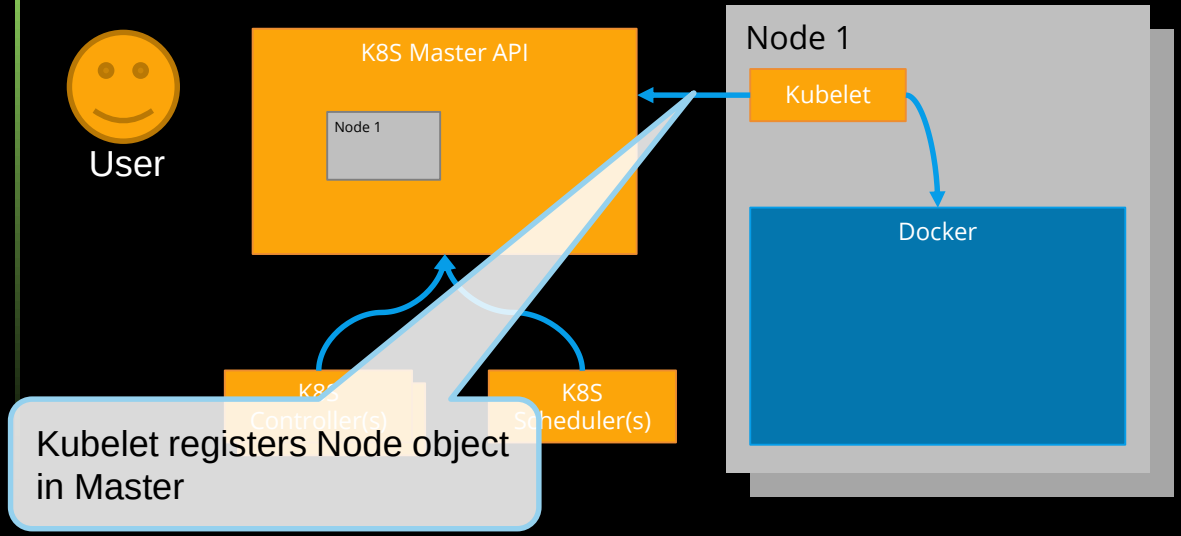
kublr

Kubernetes | Container Orchestration



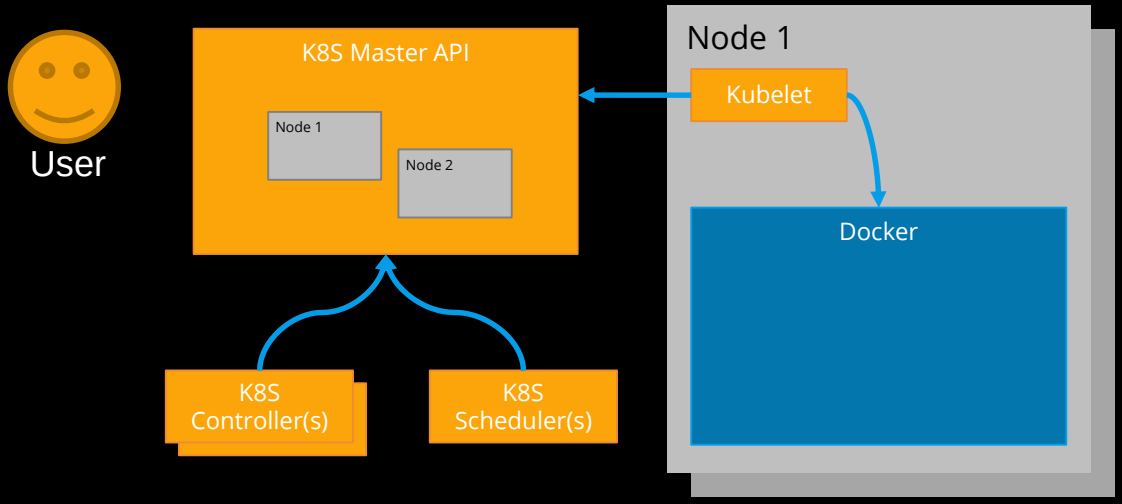
kublr

Kubernetes | Container Orchestration



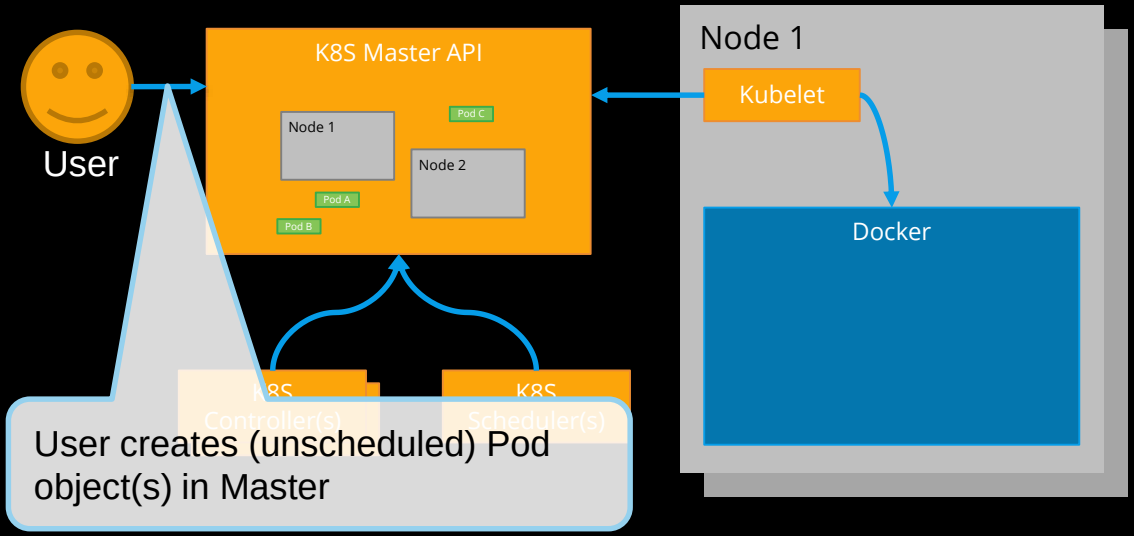
kublr

Kubernetes | Container Orchestration



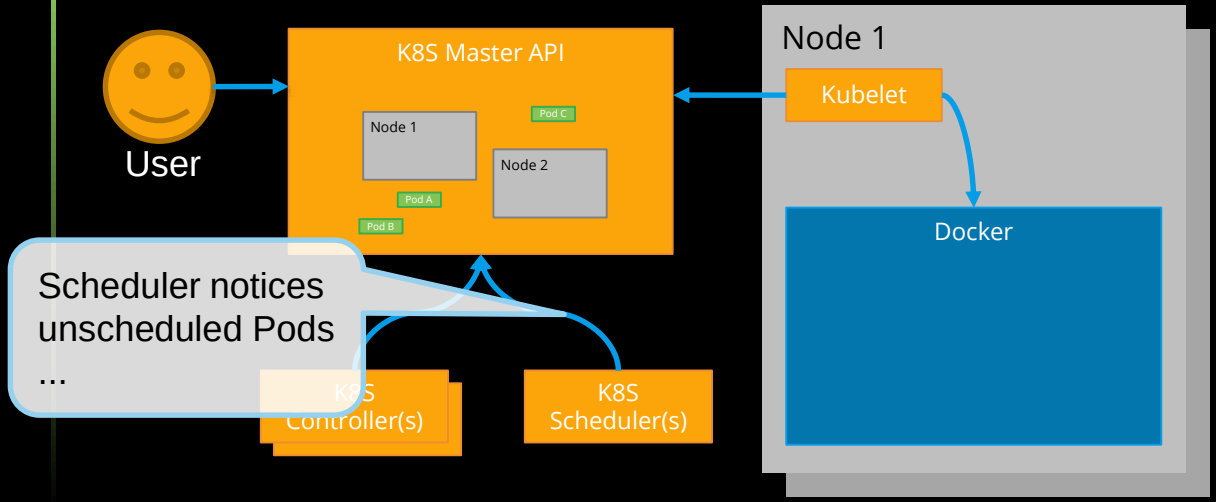
kublr

Kubernetes | Container Orchestration



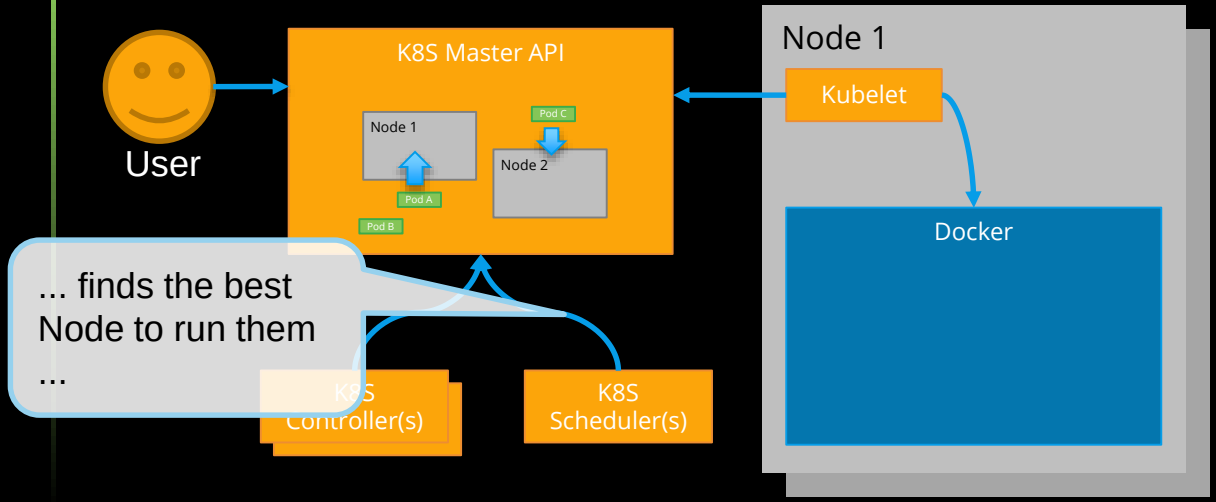
kublr

Kubernetes | Container Orchestration



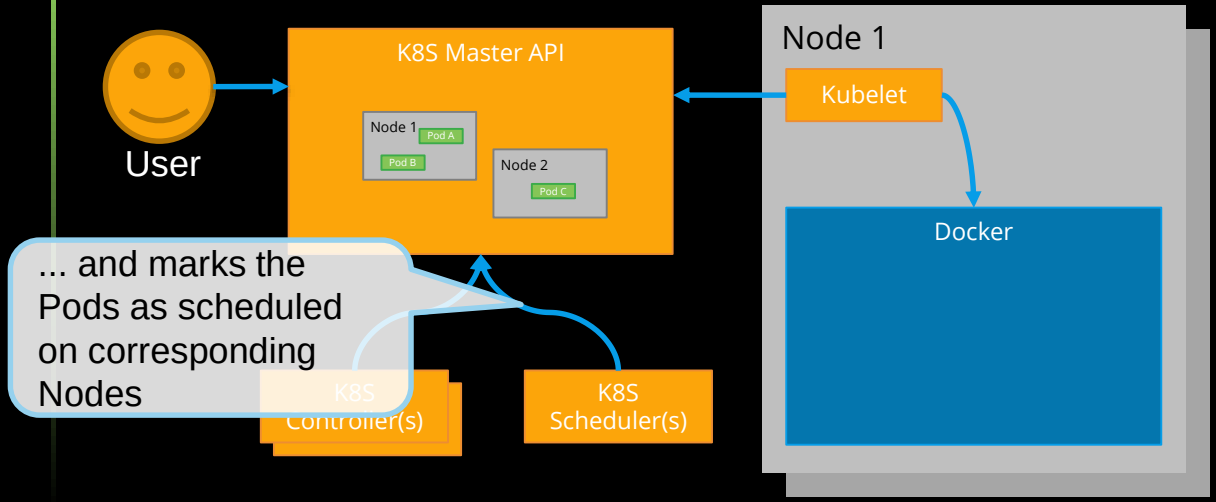
kublr

Kubernetes | Container Orchestration



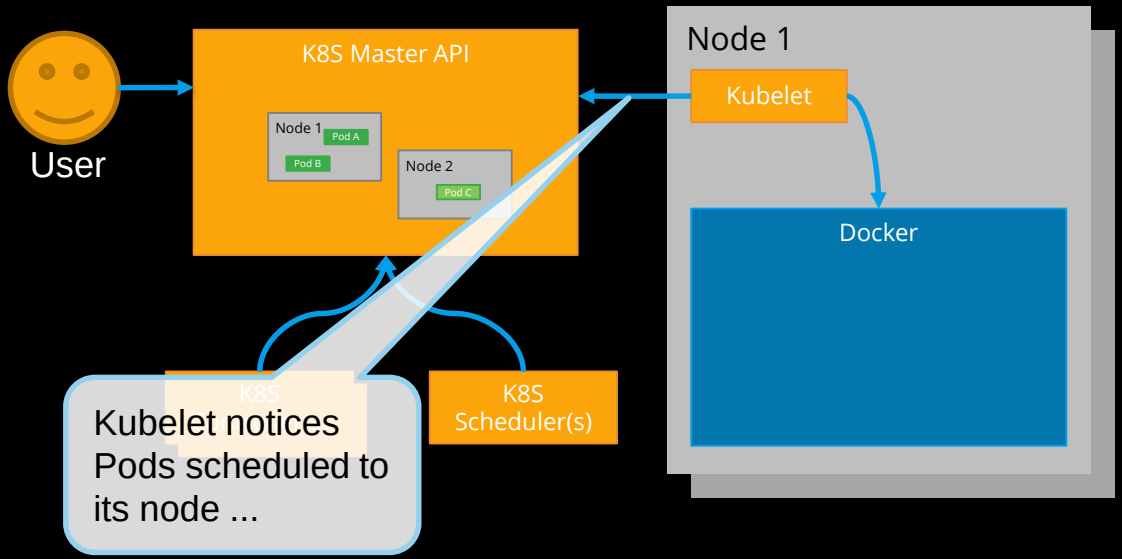
kublr

Kubernetes | Container Orchestration



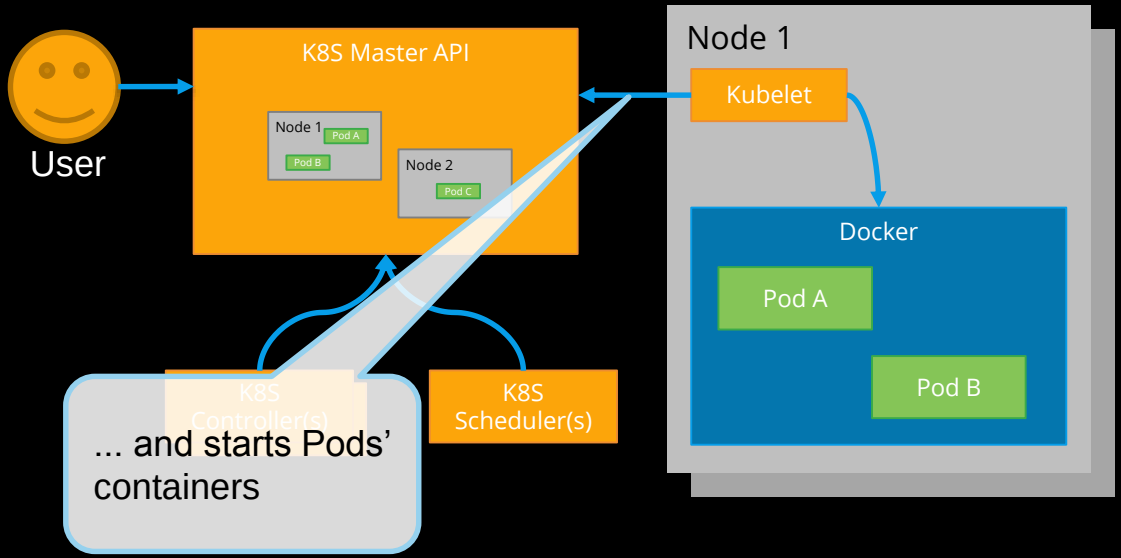
kublr

Kubernetes | Container Orchestration



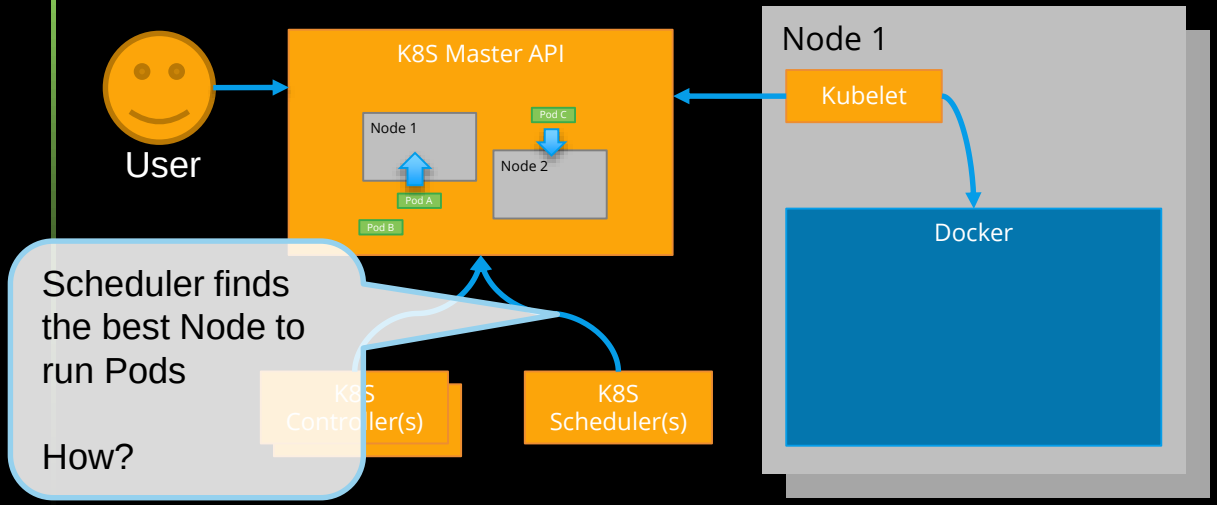
kublr

Kubernetes | Container Orchestration



kublr

Kubernetes | Scheduling Algorithm



kublr

Kubernetes | Scheduling Algorithm

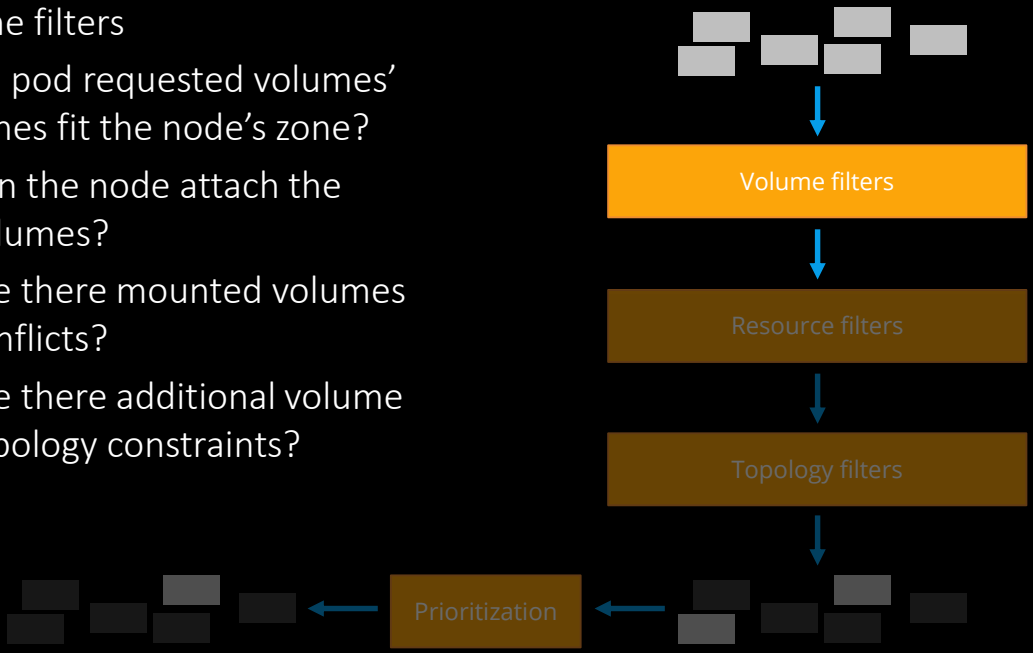
For each pod that needs scheduling:

1. Filter nodes
2. Calculate nodes priorities
3. Pick node with the highest priority

Scheduling Algorithm | Filters

Volume filters

- Do pod requested volumes' zones fit the node's zone?
- Can the node attach the volumes?
- Are there mounted volumes conflicts?
- Are there additional volume topology constraints?

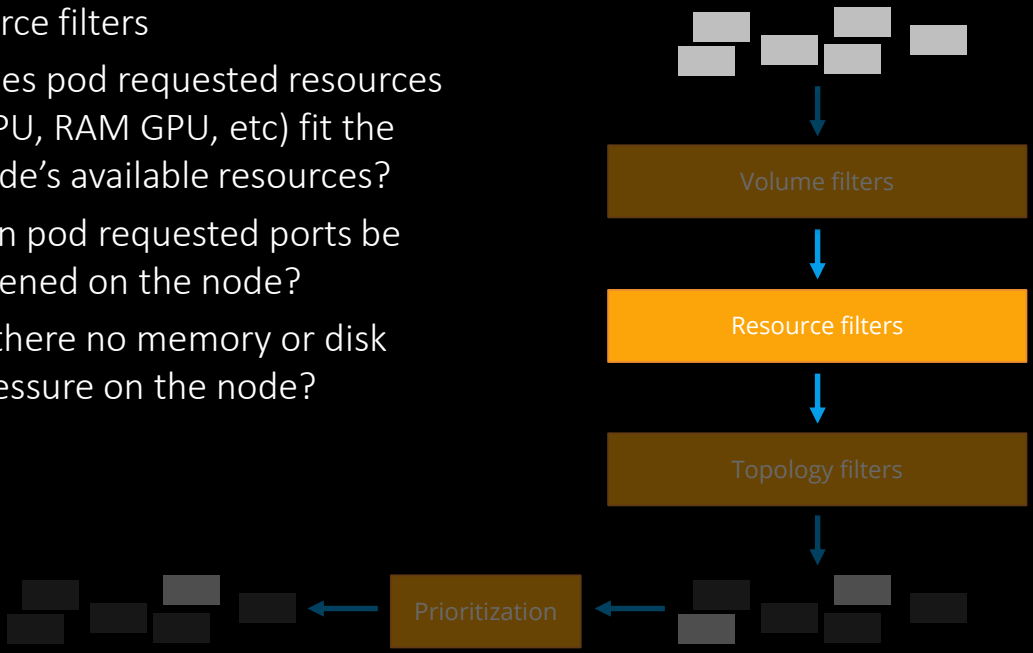


kublr

Scheduling Algorithm | Filters

Resource filters

- Does pod requested resources (CPU, RAM GPU, etc) fit the node's available resources?
- Can pod requested ports be opened on the node?
- Is there no memory or disk pressure on the node?

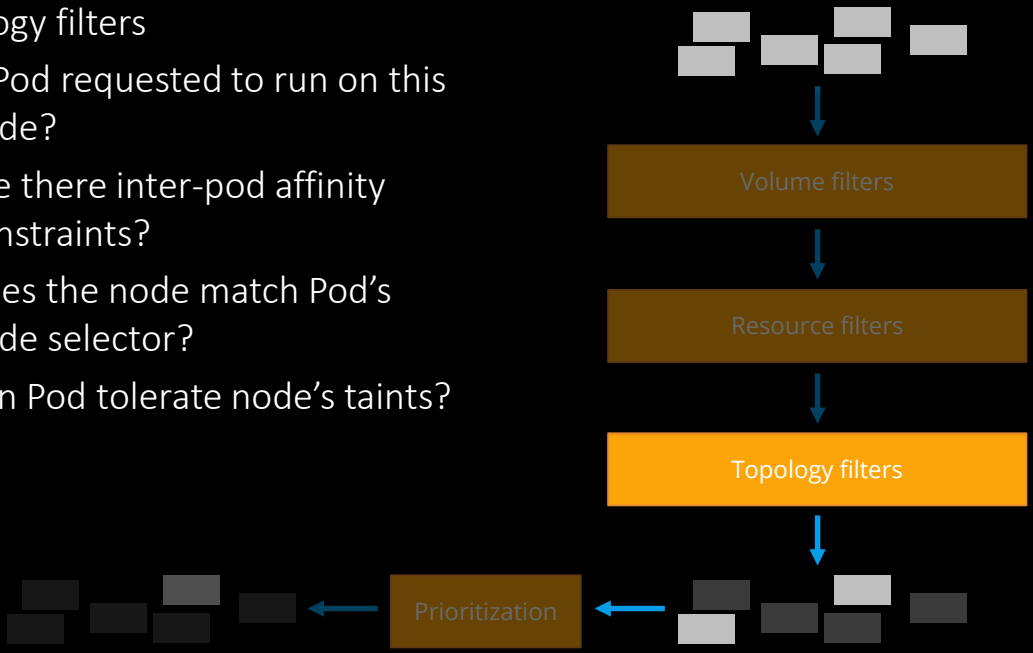


kublr

Scheduling Algorithm | Filters

Topology filters

- Is Pod requested to run on this node?
- Are there inter-pod affinity constraints?
- Does the node match Pod's node selector?
- Can Pod tolerate node's taints?

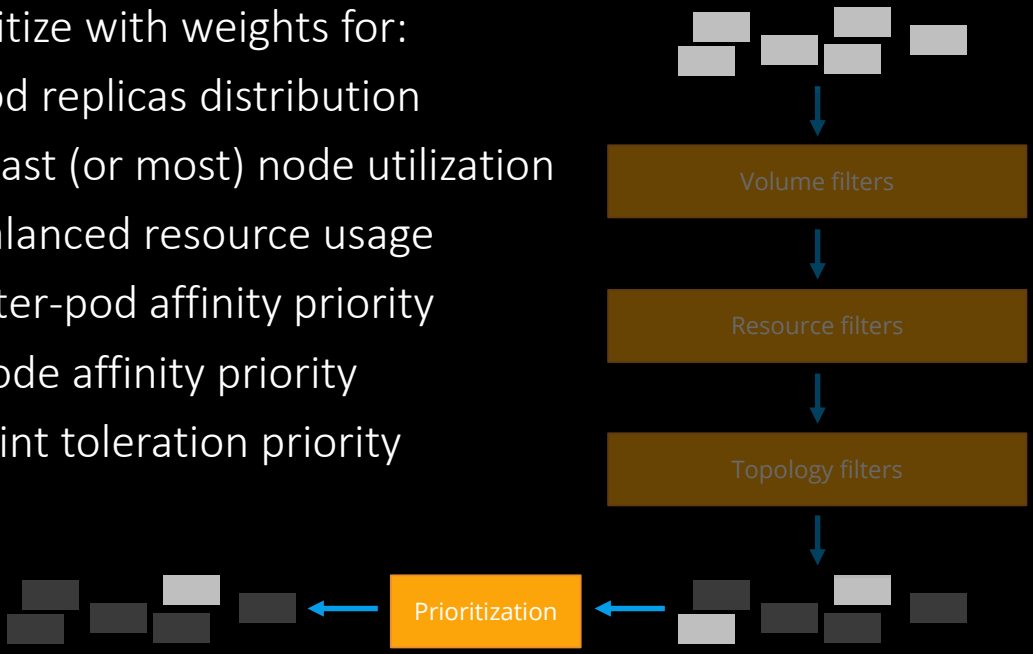


kublr

Scheduling Algorithm | Priorities

Prioritize with weights for:

- Pod replicas distribution
- Least (or most) node utilization
- Balanced resource usage
- Inter-pod affinity priority
- Node affinity priority
- Taint toleration priority



kublr

Scheduling | Controlling Pods Destin~~y~~ation

- Specify resource requirements
- Be aware of volumes
- Use node constraints
- Use affinity and anti-affinity

- Scheduler configuration
- Custom / multiple schedulers

Scheduling Controlled | Resources

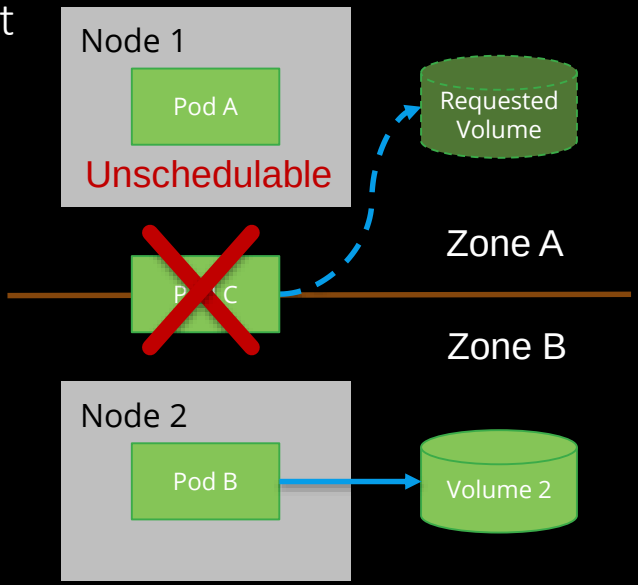
- CPU, RAM, other (GPU)
- Requests and limits
- Reserved resources

```
kind: Pod
spec:
  containers:
  - name: main
    resources:
      requests:
        cpu: 100m
        memory: 1Gi
```

```
kind: Node
status:
  allocatable:
    cpu: "4"
    memory: 8070796Ki
    pods: "110"
  capacity:
    cpu: "4"
    memory: 8Gi
    pods: "110"
```

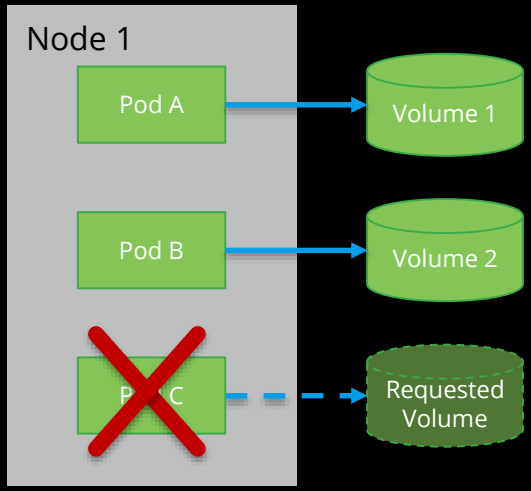
Scheduling Controlled | Volumes

- Request volumes in the right zones
- Make sure node can attach enough volumes
- Avoid volume location conflicts
- Use volume topology constraints (alpha in 1.7)



Scheduling Controlled | Volumes

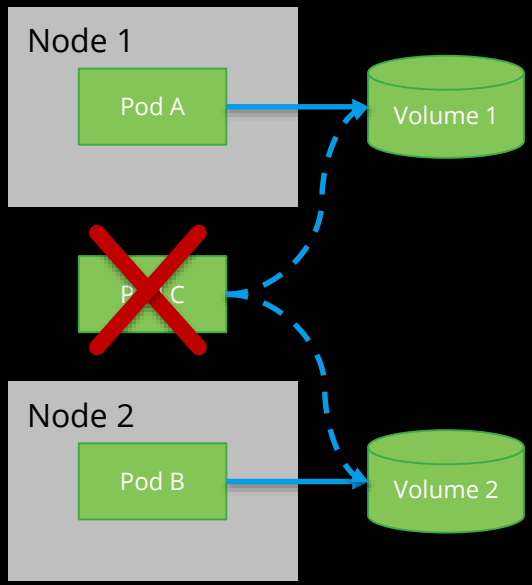
- Request volumes in the right zones
- Make sure node can attach enough volumes
- Avoid volume location conflicts
- Use volume topology constraints (alpha in 1.7)



kublr

Scheduling Controlled | Volumes

- Request volumes in the right zones
- Make sure node can attach enough volumes
- Avoid volume location conflicts
- Use volume topology constraints (alpha in 1.7)



Scheduling Controlled | Volumes

- Request volumes in the right zones
- Make sure node can attach enough volumes
- Avoid volume location conflicts
- Use volume topology constraints (alpha in 1.7)

```
annotations:
  "volume.alpha.kubernetes.io/node-affinity": '{
"requiredDuringSchedulingIgnoredDuringExecution": {
  "nodeSelectorTerms": [{
    "matchExpressions": [{
      "key": "kubernetes.io/hostname",
      "operator": "In",
      "values": ["docker03"]
    }]
  }]
}}'
```

Scheduling Controlled | Node Constraints

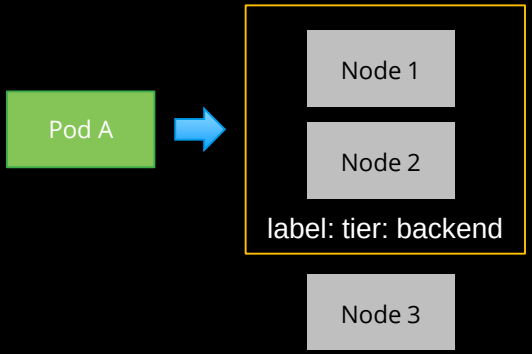
- Host constraints
- Labels and node selectors
- Taints and tolerations


```
kind: Pod
spec:
  nodeName: node1
```

```
kind: Node
metadata:
  name: node1
```

Scheduling Controlled | Node Constraints

- Host constraints
- Labels and node selectors
- Taints and tolerations



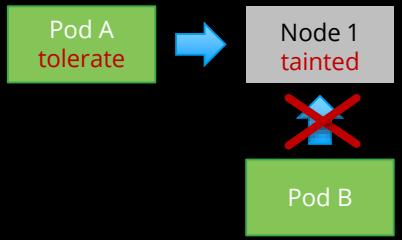

```
kind: Pod
spec:
  nodeSelector:
    tier: backend
```

```
kind: Node
metadata:
  labels:
    tier: backend
```

kublr

Scheduling Controlled | Node Constraints

- Host constraints
- Labels and node selectors
- Taints and tolerations




```
kind: Pod
spec:
  tolerations:
  - key: error
    value: disk
    operator: Equal
    effect: NoExecute
    tolerationSeconds: 60
```

```
kind: Node
spec:
  taints:
  - effect: NoSchedule
    key: error
    value: disk
    timeAdded: null
```

kublr

Scheduling Controlled | Taints

Taints communicate node conditions

- Key – condition category
- Value – specific condition
- Operator – value wildcard
  - Equal
  - Exists
- Effect
  - NoSchedule – filter at scheduling time
  - PreferNoSchedule – prioritize at scheduling time
  - NoExecute – filter at scheduling time, evict if executing
- TolerationSeconds – time to tolerate “NoExecute” taint

```
kind: Pod
spec:
  tolerations:
  - key: <taint key>
    value: <taint value>
    operator: <match operator>
    effect: <taint effect>
    tolerationSeconds: 60
```

Scheduling Controlled | Affinity

- Node affinity
- Inter-pod affinity
- Inter-pod anti-affinity

```
kind: Pod
spec:
  affinity:
    nodeAffinity: { ... }
    podAffinity: { ... }
    podAntiAffinity: { ... }
```

Scheduling Controlled | Node Affinity

- Scope
  - Preferred during scheduling, ignored during execution
  - Required during scheduling, ignored during execution

```
kind: Pod
spec:
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 10
        preference: { <node selector term> }
      - ...
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - { <node selector term> }
        - ...
```

Interlude | Node Selector vs Node Selector Term

```
...
nodeSelector:
  <label 1 key>: <label 1 value>
  ...
```

```
...
<node selector term>:
  matchExpressions:
  - key: <label key>
    operator: In | NotIn | Exists | DoesNotExist | Gt | Lt
    values:
    - <label value 1>
    ...
  ...
```

Scheduling Controlled | Inter-pod Affinity

- Scope
  - Preferred during scheduling, ignored during execution
  - Required during scheduling, ignored during execution

```
kind: Pod
spec:
  affinity:
    podAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 10
        podAffinityTerm: { <pod affinity term> }
      - ...
      requiredDuringSchedulingIgnoredDuringExecution:
      - { <pod affinity term> }
      - ...
```

Scheduling Controlled | Inter-pod Anti-affinity

- Scope
  - Preferred during scheduling, ignored during execution
  - Required during scheduling, ignored during execution

```
kind: Pod
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 10
        podAffinityTerm: { <pod affinity term> }
      - ...
      requiredDuringSchedulingIgnoredDuringExecution:
      - { <pod affinity term> }
      - ...
```

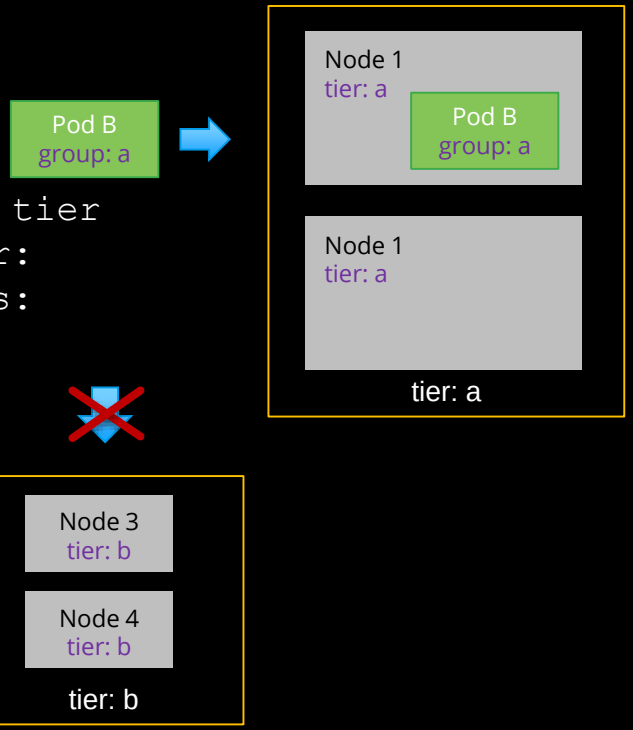
Scheduling Controlled | Pod Affinity Term

- topologyKey – nodes' label key defining co-location
- labelSelector and namespaces – select group of pods

```
<pod affinity term>:
  topologyKey: <topology label key>
  namespaces: [ <namespace>, ... ]
  labelSelector:
    matchLabels:
      <label key>: <label value>
      ...
    matchExpressions:
    - key: <label key>
      operator: In | NotIn | Exists | DoesNotExist
      values: [ <value 1>, ... ]
    ...
```

Scheduling Controlled | Affinity Example

```
affinity:
  topologyKey: tier
  labelSelector:
    matchLabels:
      group: a
```



Scheduling Controlled | Scheduler Configuration

```
kube-scheduler

--scheduler-name=default-scheduler

--algorithm-provider=DefaultProvider
--algorithm-provider=ClusterAutoscalerProvider
```

Scheduling Controlled | Multiple Schedulers

```
kind: Pod
Metadata:
  name: pod2
spec:
  schedulerName: my-scheduler
```

```
kind: Pod
Metadata:
  name: pod1
spec:
  ...
```

Scheduling Controlled | Custom Scheduler

Naive implementation

- In an infinite loop:
  - Get list of Nodes: `/api/v1/nodes`
  - Get list of Pods: `/api/v1/pods`
  - Select Pods with `status.phase == Pending` and `spec.schedulerName == our-name`
  - For each pod:
    - Calculate target Node
    - Create a new Binding object: `POST /api/v1/bindings`

```
apiVersion: v1
kind: Binding
Metadata:
  namespace: default
  name: pod1
target:
  apiVersion: v1
  kind: Node
  name: node1
```

Scheduling Controlled | Custom Scheduler

Better implementation

- Watch Pods: `/api/v1/pods`
- On each Pod event:
  - Process if the Pod with `status.phase == Pending` and `spec.schedulerName == our-name`
  - Get list of Nodes: `/api/v1/nodes`
  - Calculate target Node
  - Create a new Binding object: `POST /api/v1/bindings`

```
apiVersion: v1
kind: Binding
Metadata:
  namespace: default
  name: pod1
target:
  apiVersion: v1
  kind: Node
  name: node1
```

kublr

Scheduling Controlled | Custom Scheduler

Even better implementation

- Watch Nodes: `/api/v1/nodes`
- On each Node event:
  - Update Node cache
- Watch Pods: `/api/v1/pods`
- On each Pod event:
  - Process if the Pod with `status.phase == Pending` and `spec.schedulerName == our-name`
  - Calculate target Node
  - Create a new Binding object: `POST /api/v1/bindings`

```
apiVersion: v1
kind: Binding
Metadata:
  namespace: default
  name: pod1
target:
  apiVersion: v1
  kind: Node
  name: node1
```

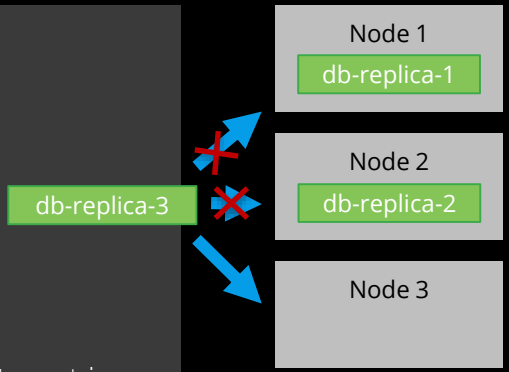
Custom Scheduler | Standard Filters

- Minimal set of filters
- kube-scheduler
  - Extend
  - Re-implement

GitHub kubernetes/kubernetes
plugin/pkg/scheduler/scheduler.go
plugin/pkg/scheduler/algorithm/predicates/predicates.go

Use Case | Distributed Pods

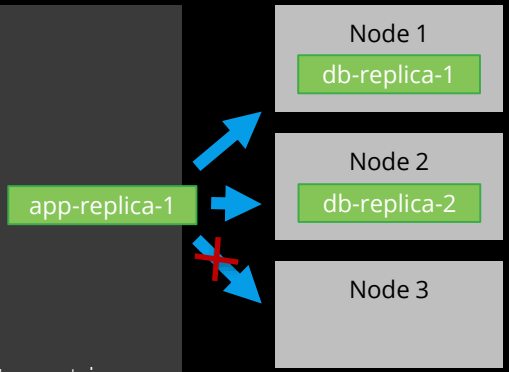
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-replica-3
  labels:
    comp: db
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - topologyKey: kubernetes.io/hostname
        labelSelector:
        matchExpressions:
        - key: comp
          operator: In
          values: [ "db" ]
```



kublr

Use Case | Co-located Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app-replica-1
  labels:
    comp: app
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - topologyKey: kubernetes.io/hostname
        labelSelector:
        matchExpressions:
        - key: comp
          operator: In
          values: [ "db" ]
```



kublr

Use Case | Reliable service on spot nodes

- “fixed” node group
  Expensive, more reliable, fixed number
  Tagged with label `nodeGroup: fixed`
- “spot” node group
  Inexpensive, unreliable, auto-scaled
  Tagged with label `nodeGroup: spot`
- Scheduling rules:
  - At least two pods on “fixed” nodes
  - All other pods favor “spot” nodes
- Custom scheduler

Thank you

Oleg Chunikhin
Chief Technology Officer
oleg@kublr.com

kublr.com