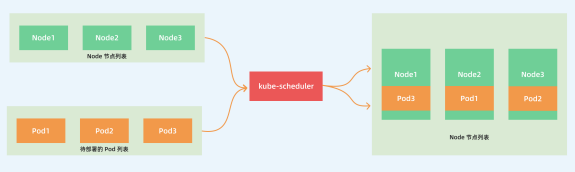
Kubernetes Scheduler Framework

kube-scheduler

1

`kube-scheduler` kubernetes Pod (Node)`kube-scheduler` Pod kubernetes Pod

`kube-scheduler` Pod Node API Server `PodSpec.NodeName` Pod Pod binding



kube-scheduler structrue

-
-
-
-
- Pod
-

kubernetes kubernetes

2

4 Kubernetes

- clone kube-scheduler
- Pod `spec.schedulerName` Pod default Pod Pod Pod Kubernetes Kubernetes API
- Webhook
- Scheduling FrameworkKubernetes v1.15 API API “” v1.16

Schedule Framework

1

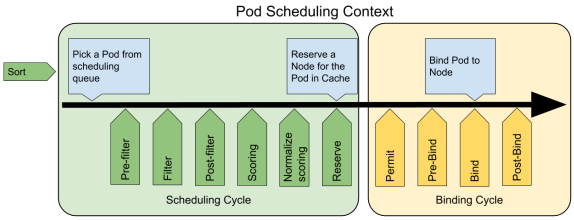
Schedule Framework

Pod Pod Pod**Scheduling Context** Pod Pod

- Pod
-

Pod

2Extension Points



Content Menu

scheduling framework extensions

1. `QueueSort` Pod PodQueueSort `Less(Pod1, Pod2)` Pod `QueueSort`
2. `Pre-filter` Pod Pod `pre-filter` error
3. `Filter` Pod `filter` `filter` `filter` `filter`
4. `Post-filter` `filter` metrics
5. `Scoring` `Soring` `normalize scoring` `scoring`
6. `Normalize scoring` `scoring` `normalize scoring`
7. `Reserve` Pod Pod Pod Pod Pod Pod reserved Unreserve Post-bind
8. `Permit` Pod Permit
    - approve permit approve Pod
    - deny permit deny Pod Pod `Unreserve`
    - wait permit wait Pod permit approvewait denyPod Unreserve
9. `Pre-bind` Pod pre-bind Pod `pre-bind` Pod Unreserve
10. `Bind` Pod
    - pre-bind bind
    - bind bind
    - bind Pod
    - bind Pod bind
11. `Post-bind`
    - Post-bind Pod
    - Post-bind
12. `Unreserve` Pod Pod unreserve Unreserve Pod reserve unreserve

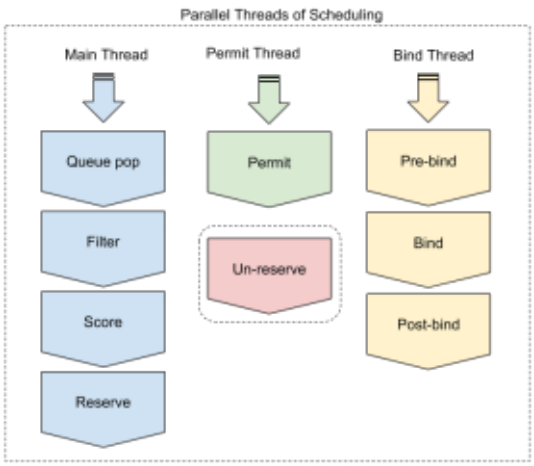
3

1

2

“reserve” “reserve” “reserve”

Pod Permit Bind Unreserve



4

`pkg/scheduler/framework/v1alpha1/interface.go`

```
// Plugin is the parent type for all the scheduling framework plugins.
type Plugin interface {
        Name() string
}
```

```
type QueueSortPlugin interface {
        Plugin
        Less(*PodInfo, *PodInfo) bool
}

// PreFilterPlugin is an interface that must be implemented by "prefilter"
plugins.
// These plugins are called at the beginning of the scheduling cycle.
type PreFilterPlugin interface {
        Plugin
        PreFilter(pc *PluginContext, p *v1.Pod) *Status
}

// FilterPlugin is an interface for Filter plugins. These plugins are
called at the
// filter extension point for filtering out hosts that cannot run a pod.
// This concept used to be called 'predicate' in the original scheduler.
// These plugins should return "Success", "Unschedulable" or "Error" in
Status.code.
// However, the scheduler accepts other valid codes as well.
// Anything other than "Success" will lead to exclusion of the given host
from
// running the pod.
type FilterPlugin interface {
        Plugin
        Filter(pc *PluginContext, pod *v1.Pod, nodeName string) *Status
}

// PostFilterPlugin is an interface for Post-filter plugin. Post-filter is
an
// informational extension point. Plugins will be called with a list of
nodes
// that passed the filtering phase. A plugin may use this data to update
internal
// state or to generate logs/metrics.
type PostFilterPlugin interface {
        Plugin
        PostFilter(pc *PluginContext, pod *v1.Pod, nodes []*v1.Node,
filteredNodesStatuses NodeToStatusMap) *Status
}

// ScorePlugin is an interface that must be implemented by "score" plugins
to rank
// nodes that passed the filtering phase.
type ScorePlugin interface {
        Plugin
        Score(pc *PluginContext, p *v1.Pod, nodeName string) (int, *Status)
}

// ScoreWithNormalizePlugin is an interface that must be implemented by
"score"
// plugins that also need to normalize the node scoring results produced
by the same
// plugin's "Score" method.
type ScoreWithNormalizePlugin interface {
        ScorePlugin
        NormalizeScore(pc *PluginContext, p *v1.Pod, scores NodeScoreList)
*Status
}

// ReservePlugin is an interface for Reserve plugins. These plugins are
called
// at the reservation point. These are meant to update the state of the
plugin.
// This concept used to be called 'assume' in the original scheduler.
// These plugins should return only Success or Error in Status.code.
However,
// the scheduler accepts other valid codes as well. Anything other than
Success
// will lead to rejection of the pod.
type ReservePlugin interface {
```

```
        Plugin
        Reserve(pc *PluginContext, p *v1.Pod, nodeName string) *Status
}

// PreBindPlugin is an interface that must be implemented by "prebind"
plugins.
// These plugins are called before a pod being scheduled.
type PreBindPlugin interface {
        Plugin
        PreBind(pc *PluginContext, p *v1.Pod, nodeName string) *Status
}

// PostBindPlugin is an interface that must be implemented by "postbind"
plugins.
// These plugins are called after a pod is successfully bound to a node.
type PostBindPlugin interface {
        Plugin
        PostBind(pc *PluginContext, p *v1.Pod, nodeName string)
}

// UnreservePlugin is an interface for Unreserve plugins. This is an
informational
// extension point. If a pod was reserved and then rejected in a later
phase, then
// un-reserve plugins will be notified. Un-reserve plugins should clean up
state
// associated with the reserved Pod.
type UnreservePlugin interface {
        Plugin
        Unreserve(pc *PluginContext, p *v1.Pod, nodeName string)
}

// PermitPlugin is an interface that must be implemented by "permit"
plugins.
// These plugins are called before a pod is bound to a node.
type PermitPlugin interface {
        Plugin
        Permit(pc *PluginContext, p *v1.Pod, nodeName string) (*Status,
time.Duration)
}

// BindPlugin is an interface that must be implemented by "bind" plugins.
Bind
// plugins are used to bind a pod to a Node.
type BindPlugin interface {
        Plugin
        Bind(pc *PluginContext, p *v1.Pod, nodeName string) *Status
}
```

KubeSchedulerConfiguration reserve preBind foo

```
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration

...

plugins:
  reserve:
    enabled:
    - name: foo
    - name: bar
    disabled:
    - name: baz
  preBind:
    enabled:
    - name: foo
    disabled:
    - name: baz

pluginConfig:
- name: foo
  args: >
    foo
```

-
-
- KubeSchedulerConfiguration enabled
- enabled

foo reserve bar foo foo bar foo

```
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration

...

plugins:
  reserve:
    enabled:
    - name: bar
    - name: foo
    disabled:
    - name: foo
```

pkg/scheduler/framework/plugins/examples

5

Kubernetes https://github.com/kubernetes-sigs/scheduler-plugins

```
func NewRegistry() Registry {
        return Registry{
                // FactoryMap:
                // New plugins are registered here.
                // example:
                // {
                //  stateful_plugin.Name: stateful.
NewStatefulMultipointExample,
                //  fooplugin.Name: fooplugin.New,
                // }
        }
}
```

NewRegistry() kube-scheduler kubernetes/cmd/kube-scheduler/app/server.go N
ewSchedulerCommand Option Option

```
// Option configures a framework.Registry.
type Option func(framework.Registry) error

// NewSchedulerCommand creates a *cobra.Command object with default
parameters and registryOptions
func NewSchedulerCommand(registryOptions ...Option) *cobra.Command {
  ......
}
```

WithPlugin Option

```
// WithPlugin creates an Option based on plugin name and factory.
func WithPlugin(name string, factory framework.PluginFactory) Option {
        return func(registry framework.Registry) error {
                return registry.Register(name, factory)
        }
}
```

```
func main() {
        rand.Seed(time.Now().UTC().UnixNano())

        command := app.NewSchedulerCommand(
                app.WithPlugin(sample.Name, sample.New),
        )

        logs.InitLogs()
        defer logs.FlushLogs()

        if err := command.Execute(); err != nil {
                _, _ = fmt.Fprintf(os.Stderr, "%v\n", err)
                os.Exit(1)
        }
}
```

app.WithPlugin(sample.Name, sample.New) WithPlugin sample.New framework.PluginFactory PluginFactory

```
type PluginFactory = func(configuration *runtime.Unknown, f
FrameworkHandle) (Plugin, error)
```

sample.New PreFilterFilterPreBind

```
//
const Name = "sample-plugin"

type Args struct {
        FavoriteColor  string `json:"favorite_color,omitempty"`
        FavoriteNumber int    `json:"favorite_number,omitempty"`
        ThanksTo       string `json:"thanks_to,omitempty"`
}

type Sample struct {
        args   *Args
        handle framework.FrameworkHandle
}

func (s *Sample) Name() string {
        return Name
}

func (s *Sample) PreFilter(pc *framework.PluginContext, pod *v1.Pod)
*framework.Status {
        klog.V(3).Infof("prefilter pod: %v", pod.Name)
        return framework.NewStatus(framework.Success, "")
}

func (s *Sample) Filter(pc *framework.PluginContext, pod *v1.Pod, nodeName
string) *framework.Status {
        klog.V(3).Infof("filter pod: %v, node: %v", pod.Name, nodeName)
        return framework.NewStatus(framework.Success, "")
}

func (s *Sample) PreBind(pc *framework.PluginContext, pod *v1.Pod,
nodeName string) *framework.Status {
        if nodeInfo, ok := s.handle.NodeInfoSnapshot().NodeInfoMap
[nodeName]; !ok {
                return framework.NewStatus(framework.Error, fmt.Sprintf
("prebind get node info error: %+v", nodeName))
        } else {
                klog.V(3).Infof("prebind node info: %+v", nodeInfo.Node())
                return framework.NewStatus(framework.Success, "")
        }
}

//type PluginFactory = func(configuration *runtime.Unknown, f
FrameworkHandle) (Plugin, error)
func New(configuration *runtime.Unknown, f framework.FrameworkHandle)
(framework.Plugin, error) {
        args := &Args{}
        if err := framework.DecodeInto(configuration, args); err != nil {
                return nil, err
        }
        klog.V(3).Infof("get plugin config args: %+v", args)
        return &Sample{
                args: args,
                handle: f,
        }, nil
}
```

> *https://github.com/cnych/sample-scheduler-framework*

6

`Deployment` `RBAC` `--config` `KubeSchedulerConfiguration` `plugins` `pluginConfig`

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
```

```
  name: sample-scheduler-clusterrole
rules:
  - apiGroups:
      - ""
    resources:
      - endpoints
      - events
    verbs:
      - create
      - get
      - update
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - pods
    verbs:
      - delete
      - get
      - list
      - watch
      - update
  - apiGroups:
      - ""
    resources:
      - bindings
      - pods/binding
    verbs:
      - create
  - apiGroups:
      - ""
    resources:
      - pods/status
    verbs:
      - patch
      - update
  - apiGroups:
      - ""
    resources:
      - replicationcontrollers
      - services
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - apps
      - extensions
    resources:
      - replicasets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - apps
    resources:
      - statefulsets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - policy
```

```
    resources:
      - poddisruptionbudgets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - persistentvolumeclaims
      - persistentvolumes
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - "storage.k8s.io"
    resources:
      - storageclasses
      - csinodes
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - "coordination.k8s.io"
    resources:
      - leases
    verbs:
      - create
      - get
      - list
      - update
  - apiGroups:
      - "events.k8s.io"
    resources:
      - events
    verbs:
      - create
      - patch
      - update
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: sample-scheduler-sa
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: sample-scheduler-clusterrolebinding
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: sample-scheduler-clusterrole
subjects:
- kind: ServiceAccount
  name: sample-scheduler-sa
  namespace: kube-system
---
apiVersion: v1
```

```
kind: ConfigMap
metadata:
  name: scheduler-config
  namespace: kube-system
data:
  scheduler-config.yaml: |
    apiVersion: kubescheduler.config.k8s.io/v1alpha1
    kind: KubeSchedulerConfiguration
    schedulerName: sample-scheduler
    leaderElection:
      leaderElect: true
      lockObjectName: sample-scheduler
      lockObjectNamespace: kube-system
    plugins:
      preFilter:
        enabled:
        - name: "sample-plugin"
      filter:
        enabled:
        - name: "sample-plugin"
      preBind:
        enabled:
        - name: "sample-plugin"
    pluginConfig:
    - name: "sample-plugin"
      args:
        favorite_color: "#326CE5"
        favorite_number: 7
        thanks_to: "thockin"
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-scheduler
  namespace: kube-system
  labels:
    component: sample-scheduler
spec:
  replicas: 1
  selector:
    matchLabels:
      component: sample-scheduler
  template:
    metadata:
      labels:
        component: sample-scheduler
    spec:
      serviceAccount: sample-scheduler-sa
      priorityClassName: system-cluster-critical
      volumes:
        - name: scheduler-config
          configMap:
            name: scheduler-config
      containers:
        - name: scheduler-ctrl
          image: cnych/sample-scheduler:v0.1.6
          imagePullPolicy: IfNotPresent
          args:
            - sample-scheduler-framework
            - --config=/etc/kubernetes/scheduler-config.yaml
            - --v=3
          resources:
            requests:
              cpu: "50m"
          volumeMounts:
            - name: scheduler-config
              mountPath: /etc/kubernetes
```

sample-scheduler

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-scheduler
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-scheduler
  template:
    metadata:
      labels:
        app: test-scheduler
    spec:
      schedulerName: sample-scheduler
      containers:
      - image: nginx
        imagePullPolicy: IfNotPresent
        name: nginx
        ports:
        - containerPort: 80
```

schedulerName sample-scheduler

```
$ kubectl get pods -n kube-system -l component=sample-scheduler
NAME                                READY   STATUS    RESTARTS   AGE
sample-scheduler-7c469787f-rwhhd   1/1     Running   0          13m
$ kubectl logs -f sample-scheduler-7c469787f-rwhhd -n kube-system
I0104 08:24:22.087881       1 scheduler.go:530] Attempting to schedule
pod: default/test-scheduler-6d779d9465-rq2bb
I0104 08:24:22.087992       1 plugins.go:23] prefilter pod: test-scheduler-
6d779d9465-rq2bb
I0104 08:24:22.088657       1 plugins.go:28] filter pod: test-scheduler-
6d779d9465-rq2bb, node: ydzs-node1
I0104 08:24:22.088797       1 plugins.go:28] filter pod: test-scheduler-
6d779d9465-rq2bb, node: ydzs-node2
I0104 08:24:22.088871       1 plugins.go:28] filter pod: test-scheduler-
6d779d9465-rq2bb, node: ydzs-node3
I0104 08:24:22.088946       1 plugins.go:28] filter pod: test-scheduler-
6d779d9465-rq2bb, node: ydzs-node4
I0104 08:24:22.088992       1 plugins.go:28] filter pod: test-scheduler-
6d779d9465-rq2bb, node: ydzs-master
I0104 08:24:22.090653       1 plugins.go:36] prebind node info: &Node
{ObjectMeta:{ydzs-node3   /api/v1/nodes/ydzs-node3 1ff6e228-4d98-4737-b6d3-
30a5d55ccdc2 15466372 0 2019-11-10 09:05:09 +0000 UTC <nil> <nil> ......}
I0104 08:24:22.091761       1 factory.go:610] Attempting to bind test-
scheduler-6d779d9465-rq2bb to ydzs-node3
I0104 08:24:22.104994       1 scheduler.go:667] pod default/test-scheduler-
6d779d9465-rq2bb is bound successfully on node "ydzs-node3", 5 nodes
evaluated, 4 nodes were found feasible. Bound node resource: "Capacity:
CPU<4>|Memory<8008820Ki>|Pods<110>|StorageEphemeral<17921Mi>; Allocatable:
CPU<4>|Memory<7906420Ki>|Pods<110>|StorageEphemeral<16912377419>.".
```

Pod Pod schedulerName

```
$ kubectl get pods
NAME                                     READY   STATUS    RESTARTS   AGE
test-scheduler-6d779d9465-rq2bb          1/1     Running   0          22m
$ kubectl get pod test-scheduler-6d779d9465-rq2bb -o yaml
......
restartPolicy: Always
schedulerName: sample-scheduler
securityContext: {}
serviceAccount: default
......
```

Kubernetes v1.17 Scheduler Framework

- https://www.qikqiak.com/post/custom-kube-scheduler/
- https://kubernetes.io/zh/docs/concepts/scheduling-eviction/scheduling-framework/
- https://github.com/kubernetes/enhancements/blob/master/keps/sig-scheduling/624-scheduling-framework/README.md