Tencent Kubernetes Engine

TKE Scheduling

Product Documentation

Contents

TKE Scheduling
Job Scheduling
Native Node Dedicated Scheduler
Overview

Last updated：2024-04-24 15:55:36

Overview

Component Overview

Kubernetes' scheduling logic operates **based on the Pod's Request**. The schedulable resources on the node are occupied by the Pod's Request amount and cannot free up. The native node dedicated scheduler is a scheduling plugin developed by Tencent Kubernetes Engine (TKE) based on the native Kube-scheduler Extender mechanism of Kubernetes, which can virtually magnify the capacity of the node, resolving the issue of the node's resources being occupied while maintaining a low utilization rate.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Requested Resource	Belonging Namespace
crane-scheduler-controller	Deployment	Each instance is endowed with 200m CPU and 200Mi memory, totaling one instance	kube-system
crane-descheduler	Deployment	Each instance is endowed with 200m CPU and 200Mi memory, totaling one instance	kube-system
crane-scheduler	Deployment	Each instance is endowed with 200m CPU and 200Mi memory,	kube-system

		totaling three instances	
crane-scheduler-controller	Service	-	kube-system
crane-scheduler	Service	-	kube-system
crane-scheduler	ClusterRole	-	kube-system
crane-descheduler	ClusterRole	-	kube-system
crane-scheduler	ClusterRoleBinding	-	kube-system
crane-descheduler	ClusterRoleBinding	-	kube-system
crane-scheduler-policy	ConfigMap	-	kube-system
crane-descheduler-policy	ConfigMap	-	kube-system
ClusterNodeResourcePolicy	CRD	-	-
CraneSchedulerConfiguration	CRD	-	-
NodeResourcePolicy	CRD	-	-
crane-scheduler-controller-mutating-webhook	MutatingWebhookConfiguration	-	-

Application Scenarios

Scenario 1: Resolving the issue of high node box rate but low utilization

Note:

The fundamental concepts are as follows.

Box Rate: It refers to the ratio of the sum of Requests of all Pods on a node to the actual specifications of the node.

Utilization: It refers to the ratio of the total actual usage of all Pods on a node to the actual specifications of the node.

The native Kubernetes scheduler schedules based on the Request resources of Pod. Therefore, even if the actual usage on the node is low at this time, if the sum of Requests of all Pods on the node is close to the actual specifications of the node, new Pods cannot be scheduled, resulting in substantial resource waste. Moreover, businesses tend to apply for surplus resources to ensure the stability of their services, that is, a large Request, leading to the occupation of node resources, unable to free up. At this point, the node's box rate is substantial, but the actual resource utilization is comparatively low.

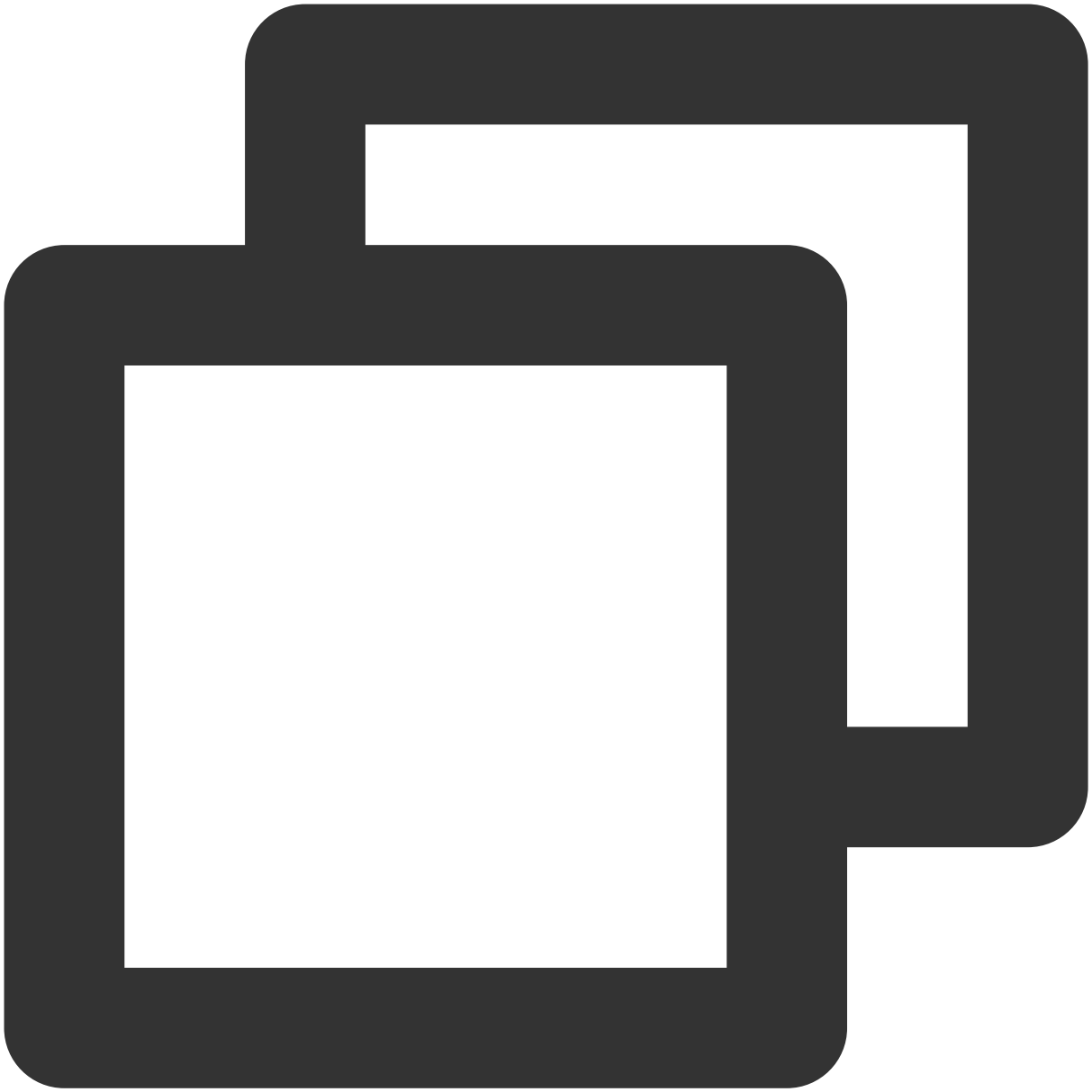
At such times, you can use the dedicated native node scheduler to virtually enhance the specifications of CPU and memory on a node, thus amplifying its scheduler resources. More pods can thereby be scheduled.

Scenario 2: Setting the watermark of the nodes

The watermark setting of the node is to ensure the stability of the node and set the node's target utilization rate:
Control of the watermark during scheduling: This step determines the native node's target resource utilization rate to guarantee stability. While scheduling the Pods, nodes with resources above this watermark will not be selected. Moreover, from nodes meeting the watermark requirements, as shown in the following figure, nodes with lower actual load watermarks have priority to balance the cluster node's utilization distribution.
Control of the watermark during runtime: This step determines the current target resource utilization rate for native nodes to guarantee stability. At runtime, nodes with resources above this watermark could trigger evictions. Given that eviction is a high-risk action, bear in mind the following notes.

Notes

1. To avoid draining important Pods, this feature is set not to evict Pods by default. For Pods that can be safely drained, it is essential for users to explicitly determine the workload to which the Pod belongs. For example, StatefulSet, Deployment, and other objects can be set as drainable annotations:



```
descheduler.alpha.kubernetes.io/evictable: 'true'
```

2. It is recommended to enable event persistence for the cluster, to better monitor component abnormalities and troubleshoot. When evicting a Pod, corresponding events will be generated. You can observe if the Pod is being repeatedly evicted based on the Descheduled event.
3. The eviction action has requirements for nodes: a cluster is required to have 3 or more low-load native nodes, where a low-load definition refers to a Node's load that is lesser than its operational water-level control.
4. After filtering at the node dimension, evacuation begins on the workload on the Node. This necessitates the constraint that the replica count of the workload should be equal to or greater than 2, or at least half of the Workload

spec replicas.
5. At the Pod dimension level, if a Pod's load exceeds the eviction watermark of the node, eviction is forbidden to prevent the overloading of other nodes by relocating them there.

Scenario 3: Pods under specified Namespace shall be allocated only to native nodes upon the subsequent scheduling

Native nodes, the newly-launched node types, are introduced by the TKE Tencent Kubernetes Engine team of Tencent Cloud. They are built upon the technical excellence derived from Tencent Cloud's tens of millions of core container operations, thereby delivering native-like, high-stability, and rapid-response K8s node management capabilities. Native nodes, with amplifiable node specifications and recommended Request capabilities, are hence highly advisable for exploiting its advantages fully by scheduling your workload to them. While enabling the native node scheduler, you can opt for Namespace. Consequently, Pods under the specified Namespace shall be scheduled exclusively to native nodes in the following scheduling.

Note:

If the native node resources are insufficient at this stage, it would result in Pod Pending.

Limits

This feature is only supported by the native node. For more information, see Native Node Overview.
It is required to ensure that the Kubernetes version is v1.22.5-tke.8, v1.20.6-tke.24, v1.18.4-tke.28,v1.16.3-tke.30 or higher. For cluster versions upgrade, see Upgrading a Cluster.

Risk Control

After the uninstallation of this component, only the scheduling logic associated with the native node-dedicated scheduler will be eliminated, leaving the scheduling capability of the native Kube-Scheduler untouched. The already scheduled Pods on the native node will not be affected due to their pre-set schedule. However, a reboot of kubelet on the native node might trigger Pod eviction as the sum of Pods' Requests on the native node could exceed the genuine specifications of the native node.
In the event of the amplification coefficient being adjusted downwards, the existing Pods on the native node, due to their already prescribed schedule, will remain unaffected. Nonetheless, if the kubelet on the native node restarts, it might trigger Pod eviction since the aggregate of Pods' Requests on the native node could surpass the amplified specifications of the native node after the amplification.
Users witness the inconsistency between Node resources in the Kubernetes cluster and corresponding CVM node resources.
In the future, issues related to excessive load and instability could possibly arise.

After the amplification of the node specifications, the node kubelet layer and the resource QoS-related modules might be affected. For instance, kubelet's binding cores, when a 4-core node is treated as an 8-core node for scheduling, the Pods' binding cores could possibly be impacted.

Component Permission Description

Crane Scheduler Permission

Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
It is required to keep track of the updates and changes to the node, as well as the utilization of the access node.	nodes	get/watch/list
Track the updates and changes of pods, and determine the scheduling priority of nodes based on the recent scheduling situation of pods within the cluster.	pods/namespaces	get/watch/list
It is required to update node utilization to node resources, thereby achieving the decoupling of scheduling and query logic.	nodes/status	patch
It is required to support multiple replicas to ensure component availability.	leases	create/get/update
It is required to track the updates and changes of the configmap, implementing the feature of scheduling specified pods to native nodes.	configmap	get/list/watch

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: crane-scheduler
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - nodes
  - namespaces
```

```
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - nodes/status
  verbs:
  - patch
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - extensions
  - apps
  resources:
  - deployments/scale
  verbs:
  - get
  - update
- apiGroups:
  - coordination.k8s.io
  resources:
  - leases
  verbs:
  - create
  - get
  - update
- apiGroups:
  - "scheduling.crane.io"
  resources:
  - clusternoderesourcepolicies
  - noderesourcepolicies
  - craneschedulerconfigurations
  verbs:
  - get
  - list
  - watch
  - update
  - create
  - patch
```

Crane Descheduler Permission

Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
It is required to keep track of the updates and changes to the node, as well as the utilization of the access node.	nodes	get/watch/list
Track the updates and changes of the pods, determining the pods to be evicted first based on the information of the pods within the clusters.	pods	get/watch/list
Drain the pod.	pods/eviction	create
It is required to determine whether the number of ready workloads where the pod resides constitutes half or more of the total requirements to decide whether to drain the pod.	replicasets/deployments/statefulsets/statefulsetpluses/job	get
Report events when draining Pods.	create	events

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: crane-descheduler
  namespace: kube-system
rules:
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "watch", "list"]
  - apiGroups: [""]
    resources: ["pods"]
```

```
    verbs: ["get", "watch", "list"]
  - apiGroups: [""]
    resources: ["nodes/status"]
    verbs: ["patch"]
  - apiGroups: [""]
    resources: ["pods/eviction"]
    verbs: ["create"]
  - apiGroups: ["*"]
    resources: ["replicasets"]
    verbs: ["get"]
  - apiGroups: ["*"]
    resources: ["deployments"]
    verbs: ["get"]
  - apiGroups: ["apps"]
    resources: ["statefulsets"]
    verbs: ["get"]
  - apiGroups: ["platform.stke"]
    resources: ["statefulsetpluses"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create"]
  - apiGroups: ["*"]
    resources: ["jobs"]
    verbs: ["get"]
  - apiGroups: [ "coordination.k8s.io" ]
    resources: [ "leases"
```

Request Recommendation

Last updated：2024-03-04 15:26:47

Overview

Component Overview

Kubernetes can efficiently improve business orchestration capabilities and resource utilization. With no additional capabilities for support, this enhancement remains substantially limited. The average resource utilization of a TKE node is merely about 14% according to the previous statistics by the TKE team.
The main reason for the poor resource utilization rate of a Kubernetes cluster is adherence to Kubernetes' resource scheduling logic. When creating Kubernetes workloads, it's typical to configure suitable resource Requests and Limits for the workload, indicating resource concession and restriction. Among these, the Requests have the most significant impact on the utilization rate. To prevent the resources employed by their workload from being occupied by others, or to cater to the resource demands during peak traffic, users tend to set larger values for Request. The disparity between the Requests and the actual utilized resources cannot be employed by other workloads, resulting in wastage. The unreasonable setting of Request values leads to a low resource utilization rate in the Kubernetes cluster.
Tencent Kubernetes Engine (TKE) supports the installation of **Request Recommendation** component in the cluster. Request Recommendation allows for the suggestion of Request/Limit values for container-level resources in Kubernetes workloads, reducing resource wastage.

Resource objects deployed in a cluster

By enabling Request Recommendation in a cluster, it will deploy the following Kubernetes objects within a cluster:

Kubernetes Object Name	Type	Default Resource Occupation	Associated Namespace
analytics.analysis.crane.io	CustomResourceDefinition	-	-
recommendations.analysis.crane.io	CustomResourceDefinition	-	-
crane-system	Namespace	-	-
housekeeper-default	Analytics	-	crane-system
recommendation-config	ConfigMap	-	crane-system
craned	ClusterRole	-	-
craned	ClusterRoleBinding	-	-
craned	Service	-	crane-system

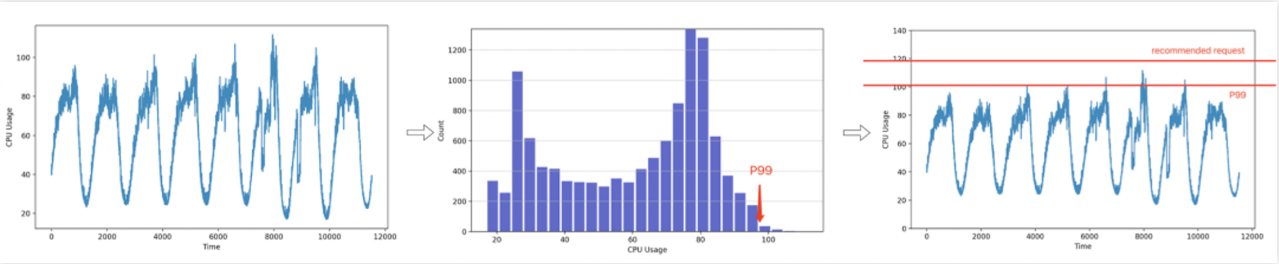
craned	ServiceAccount	-	crane-system
craned	Deployment	-	crane-system

Feature Overview

It supports recommending suitable Request/Limit values of resources for each container in Deployment, StatefulSet, and DaemonSet.
It supports one-click update of the resource values for containers in the initial workload with recommended values.
It supports maintaining the Request/Limit ratio. The recommended Request/Limit will preserve the proportion between the Reqeust/Limit in the initial Workload Container Settings. If the Limit is not set upon Workload creation, a Limit recommendation won't be provided.
The console's one-click update capability for Request recommendations will add a nodeSelector attribute to the workload by default. During workload updates, Pods can only be scheduled on native nodes. If native node resources are insufficient, it will lead to a pending of the Pod.

Principles of Request Recommendation

The component creates an Analytics CR object under the crane-system Namespace, covering all native Kubernetes workloads (Deployment, DaemonSet, StatefulSet) in all clusters. It analyzes workload data for up to 14 days, updating recommended values every 12 hours.
It then produces a Recommendation CR object for each workload within the cluster based on Analytics, purposed for data storage of recommendations.
If recommendation CR generates recommendation data, it will inscribe this information into the corresponding workload's Annotation.



Notes

Environment Requirements

Kubernetes version: 1.10+

Node Requirements

The **One-Click Update Workload Request** feature in the Tencent Kubernetes Engine Console will migrate the workload to the native node. If your cluster's native node lacks resources, it could result in a pending of the Pod.

Requirements on the Controlled Resources

It supports Deployment, StatefulSet, and DaemonSet.
It does not support Job and CronJob, as well as the Pods that are not managed by a workload.

Recommended Threshold

Suggested minimum values: The recommended minimum value for CPU per container is 0.125 core, i.e. 125 m; the minimum memory is 125 Mi.

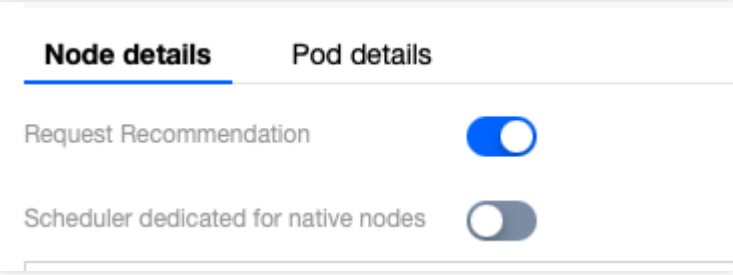
Instructions

Installing a Component

1. Log in to the Tencent Kubernetes Engine Console.
2. Select **TKE Insight** > **Node Map** on the left.

Note:
You can also undertake the installation in **TKE Insight** > **Workload Map**.

3. On the Node Map page, hover your mouse over a Node at the bottom of the page, and click **Details**.
4. In the top right corner of the Node details page, enable the **Request Recommendation** switch to configure the scheduler's parameters.



Note:
This feature comprises a global switch at the cluster level. After the feature is enabled, it will automatically analyze the historical monitoring data of workloads to recommend appropriate Request values.

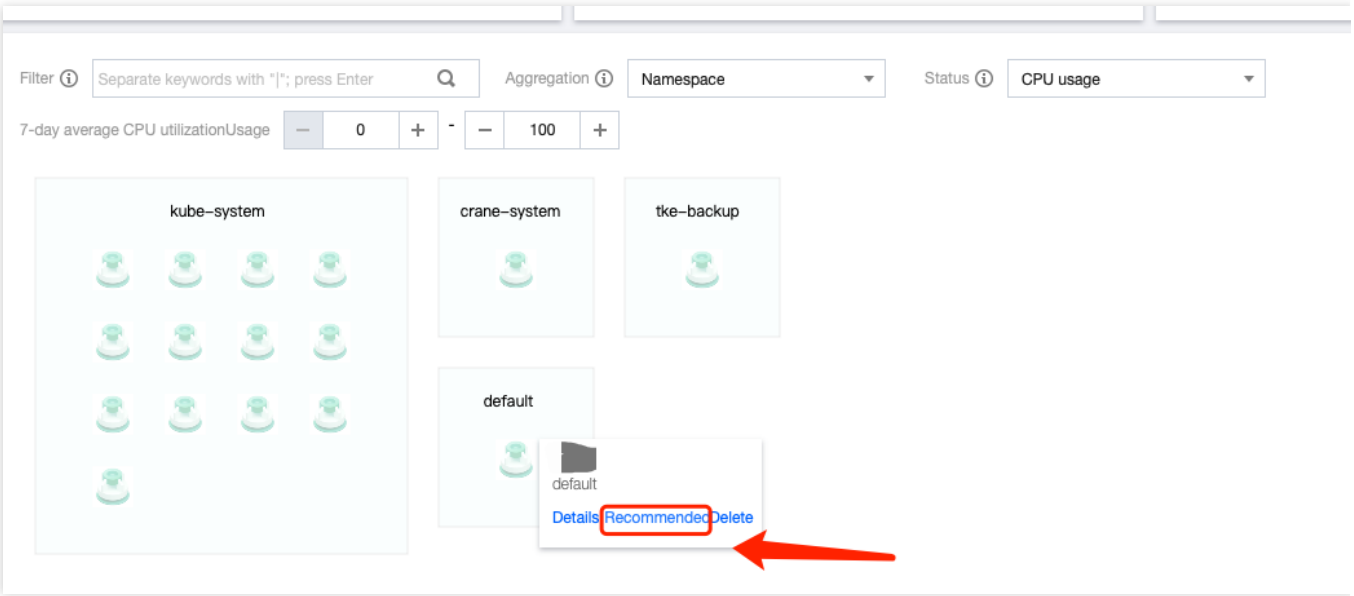
This feature does not take effect immediately after enabling. The system will analyze the resource usage history to provide accurate recommended values.
The period for calculation may vary for different workloads. One workload within a cluster may potentially impact another.
After this feature is enabled, values will be recommended for the workloads that run at least for one day.
For workloads created after this feature is enabled, it usually takes one day to recommend values.
It is recommended to update the Workload with the recommended values after the workload has been running stably for a while.

Using a Component

1. Log in to the Tencent Kubernetes Engine Console.
2. Select **TKE Insight > Workload Map** on the left.

Note:
Workload Map mainly displays various states and metrics of workloads through a visual interface, assisting users in comprehending the current configuration volume of the workload and its actual usage, thereby helping in analyzing potential issues within the workload. For more information, see the Workload Map documentation.

3. On the Workload Map page, hover your mouse over a workload at the bottom of the page, and click **Recommended**.



4. In the pop-up window, click **Confirm** to use the suggested Request value for updating the original value in the Workload.

Note:
The **One-Click Update Workload Request** feature in the Tencent Kubernetes Engine Console will migrate the workload to the native nodes. If your native nodes in your cluster lack resources, it will result in a pending of the Pod.

Accessing Recommended Values in the Background

The Request Recommendation engine stores the recommended values in the YAML file of each workload. You can use the standard Kubernetes API to access these recommended values for each workload and then integrate them into your business's deployment system. The following demonstrates how to peek into the recommended Request amount for each container under a workload:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
```

```
    analysis.crane.io/resource-recommendation: |
      containers:
      # If a Pod contains multiple containers, each container has recommended value
      - containerName: nginx
        target:
          cpu: 125m
          memory: 125Mi #If unit is missing herein, a character string "58243235" w
```

Note:

The component itself does not recommend a Limit. When updating the Workload using the Request recommendation value in the console, it will maintain the ratio of the Workload's Request and Limit to ensure the Quality of Service (QoS) remains constant. If you access the recommended value of the Request in the background, you can consider it as a reference to update the resource configuration of the original Workload.

Component Permission Description

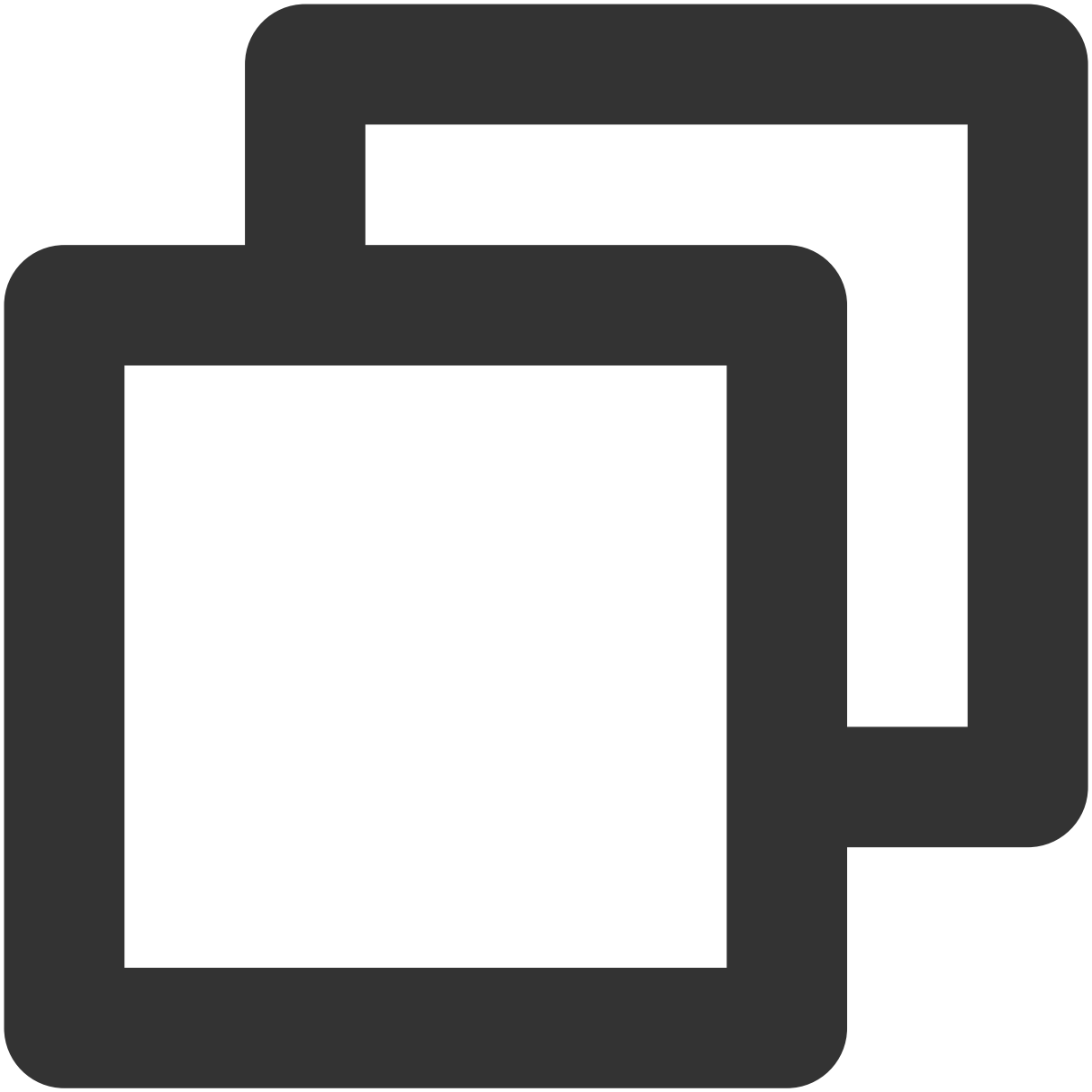
Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
Recording the oom record of the pod	pod	get/list/watch
Searching and recommending idle nodes based on the node	node	get/list/watch
It is required to record the exception information in the form of events.	event	create/patch/update
Monitoring changes in related recommendation resources, and making resource recommendation	analysis.crane.io	All permissions

Permission Definition



```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: craned
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  - pods
  - nodes
```

```
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - analysis.crane.io
  resources:
  - "*"
  verbs:
  - "*"
- apiGroups:
  - apps
  resources:
  - daemonsets
  - deployments
  - deployments/scale
  - statefulsets
  - statefulsets/scale
  verbs:
  - get
  - list
  - watch
- apiGroups:
    - apps
  resources:
    - daemonsets/status
    - deployments/status
    - deployments/scale
    - statefulsets/status
    - statefulsets/scale
  verbs:
    - update
- apiGroups:
  - autoscaling
  resources:
  - horizontalpodautoscalers
  verbs:
  - '*'
- apiGroups:
  - autoscaling.crane.io
  resources:
  - '*'
  verbs:
  - '*'
- apiGroups:
  - ""
  resources:
```

```
  - events
  verbs:
  - create
  - patch
  - update
- apiGroups:
  - prediction.crane.io
  resources:
  - '*'
  verbs:
  - '*'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: craned
  namespace: crane-system
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
      - secrets
    verbs:
      - create
  - apiGroups:
      - ""
    resourceNames:
      - craned
    resources:
      - configmaps
    verbs:
      - get
      - patch
      - update
  - apiGroups:
      - ""
    resourceNames:
      - clusters-secret-store
    resources:
      - secrets
    verbs:
      - get
  - apiGroups:
      - coordination.k8s.io
    resources:
      - leases
```

```
    verbs:
      - get
      - patch
      - update
      - create
```

Fine Scheduling

QoSAgent

Last updated：2024-02-05 16:28:54

QoS Agent is an extended component enhanced by Tencent Cloud based on quality of service, offering an array of capabilities. It ensures stability while increasing the utilization rate of cluster resources.

Note:

QoS capabilities are only supported on native nodes. If your nodes are not native, or your workload does not reside on native nodes, these capabilities will not be effective.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Default Resource Occupation	Associated Namespaces
avoidanceactions.ensurance.crane.io	CustomResourceDefinition	-	-
nodeqoss.ensurance.crane.io	CustomResourceDefinition	-	-
podqoss.ensurance.crane.io	CustomResourceDefinition	-	-
timeseriespredictions.prediction.crane.io	CustomResourceDefinition	-	-
kube-system	Namespace	-	-
all-be-pods	PodQOS	-	kube-system
qos-agent	ClusterRole	-	-
qos-agent	ClusterRoleBinding	-	-
crane-agent	Service	-	kube-system
qos-agent	ServiceAccount	-	kube-system
qos-agent	Daemonset	-	kube-system

Feature Overview

Feature	Description
Priority of CPU Usage	The feature of setting CPU usage priority ensures a sufficient supply of resources for high-priority tasks during resource competition, thereby suppressing low-priority tasks.
CPU Burst	CPU Burst permits temporary provision of resources beyond the limit for latency-sensitive applications, ensuring their stability.
CPU Hyperthreading Isolation	Preventing L2 Cache of high-priority container threads from being affected by low-priority threads running on the same CPU physical core.
Memory QoS Enhancement	A comprehensive enhancement of memory performance, along with the flexible limitations on the memory usage of the container.
Network QoS Enhancement	A comprehensive enhancement of network performance, along with flexible limitations on the network usage of the container.
Disk IO QoS Enhancement	A comprehensive enhancement of disk performance, along with flexible limitations on the disk usage of the container.

QoS Agent Permission

Note:

The **Permission Scenarios** section only lists the permissions related to the core features of the components, for a complete permission list, please refer to the **Permission Definition**.

Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
Reading podqos, nodeqos, time series, and other configurations	podqoss / nodeqoss / avoidanceactions	get/list/watch/update
Viewing the pod information of the current node	pod	get/list/watch
Enabling isolation capability based on Podqos/ Modifying node resources to increase offline resources	pod status	update/patch
Adding a taint to the node	node	get/list/watch/update

Sending events based on the status of isolation and resource interference	event	All Permissions

Permission Definition

```
rules:
 - apiGroups:
      - ""
    resources:
      - pods
```

```
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - pods/status
    verbs:
      - update
      - patch
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - get
      - list
      - watch
      - update
  - apiGroups:
      - ""
    resources:
      - nodes/status
      - nodes/finalizers
    verbs:
      - update
      - patch
  - apiGroups:
      - ""
    resources:
      - pods/eviction
    verbs:
      - create
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
```

```
      - "*"
  - apiGroups:
      - "ensurance.crane.io"
    resources:
      - podqoss
      - nodeqoss
      - avoidanceactions
    verbs:
      - get
      - list
      - watch
      - update
  - apiGroups:
      - "prediction.crane.io"
    resources:
      - timeseriespredictions
      - timeseriespredictions/finalizers
    verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
  - apiGroups:
      - "topology.crane.io"
    resources:
      - "noderesourcetopologies"
    verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
```

Deployment Methods

1. Log into the Tencent Kubernetes Engine Console, and choose **Cluster** from the left navigation bar.
2. In the Cluster list, click the desired Cluster ID to access its detailed page.
3. Select **Add-on management** from the left-side menu, and click **Create** within the Component Management page.
4. On the **Create Add-on management** page, tick the box for **QoS Agent**.
5. Click **Complete** to install the add-on.

Please Note:

With the completion of the deployment, you need to manually select the corresponding driver due to potential differences in cgroup driver of the cluster. The instructions are as follows:

1. Within the **Add-on** in your cluster, locate the successfully deployed QoS Agent, and click **Update configuration** on the right.
2. On the add-on configuration page of QoS Agent, select the dropdown box to the right of the cgroupDrive option, and choose cgroupDrive that matches your cluster.
3. Click **Complete**.

FAQs

How to confirm the cgroupDrive of a cluster?

The cgroupDrive of a cluster can only be either cgroupfs or systemd. The confirmation method is as follows:
Initially, the operation of peekcluster can be viewed in the "basic information" page of the cluster, specifically in the "operating add-on", by determining whether the current cluster serves as a docker or containerd.
If the operating cluster is docker, on any node in the cluster, execute `docker info` and view the field content of `Cgroup Driver` .
If the operating cluster is containerd, in the file of /etc/containerd/config.toml on any node in the cluster, the presence of the field: `SystemdCgroup = true` signifies a systemd, otherwise, it is a cgroup.

How to select the operating business or node?

Choosing a specific resource object via `label` or `scope` is supported.

Note:

When both of the following selectors exist concurrently, the operation used is an "and", i.e. all conditions must be met.

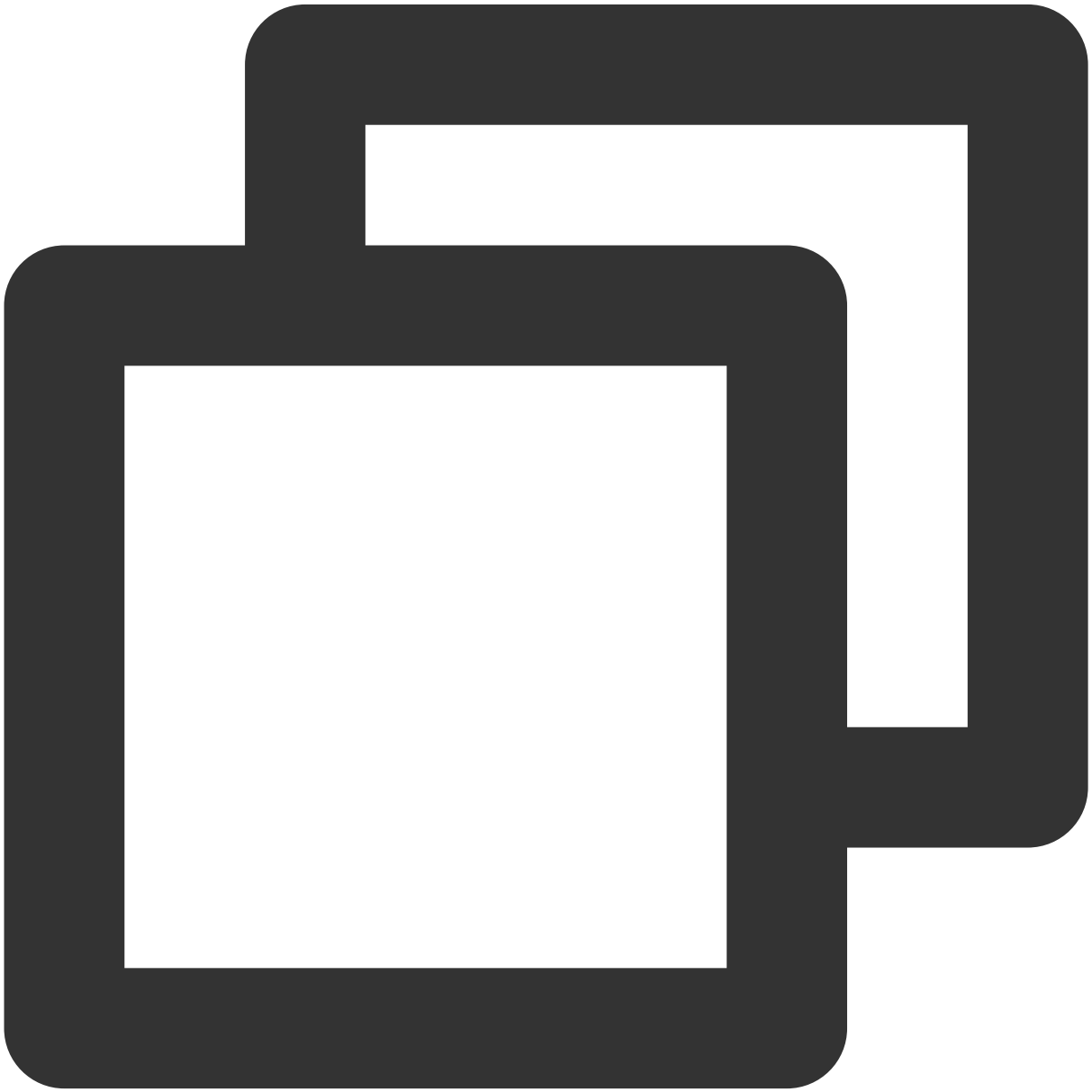
labelSelector

The labelSelector filters resources by associating them with the resource labels of the object. The usual method of usage is to attach a specific tag to the designated workloads on the business end. This Tag is then given to the operation team. When creating a PodQOS, the operation team associates this tag through the labelSelector field, effectively granting different QoS capabilities to different businesses.

scopeSelector

The scopeSelector is composed of multiple MatchExpressions. The relationship between these MatchExpressions is an "and". There are three fields in MatchExpressions, namely ScopeName, Operator, and Values corresponding to ScopeName;
The ScopeName includes three types: QOSClass, Priority, and Namespace;

QOSClass refers to a desired Workload that is associated with a specific QOSClass. The Values can be one or more among Guaranteed, Burstable, and BestEffort;
Priority refers to a desired Workload that is associated with a specific Priority. The Values can be specific priority values, such as ["1000", "2000-3000"], supporting a range of priorities;
Namespace refers to a desired Workload that is associated with a specific Namespace. The Values can be one or more.
Operator includes two types, specifically In and NotIn. If left it blank, the default type is In.
As illustrated below, it denotes that the BestEffortPod meets a condition of app-type=offline, with a CPU priority of 7:



```
apiVersion: ensurance.crane.io/v1alpha1
```

```
kind: PodQOS
metadata:
  name: offline-task
spec:
  allowedActions:
  - eviction
  resourceQOS:
    cpuQOS:
      cpuPriority: 7
  scopeSelector:
    matchExpressions:
    - operator: In
      scopeName: QOSClass
      values:
      - BestEffort
  labelSelector:
    matchLabels:
      app-type: offline
```