Tencent Kubernetes Engine

TKE General Cluster Guide

Product Documentation

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

TKE General Cluster Guide

TKE General Cluster Overview

Last updated：2023-05-06 19:41:07

Overview

Tencent Kubernetes Engine (TKE) is a container management service with high scalability and performance that enables you to easily run applications in a managed CVM instance cluster. This service frees you from installation, operations, and expansion of the cluster management infrastructure. In addition, it allows you to launch and terminate Docker applications, query the status of the cluster, and use various Tencent Cloud services through simple API calls. You can arrange containers in your cluster based on resource and availability requirements to meet your business or application-specific needs.

Based on native Kubernetes, TKE provides a container-oriented solution that solves operating environment issues during development, testing, and OPS and helps reduce costs and improve efficiency. TKE is fully compatible with the native Kubernetes APIs and extends Kubernetes plug-ins such as CBS and CLB on the Tencent Cloud. In addition, TKE provides network solutions with high reliability and performance based on Tencent Cloud VPC.

Glossary

The following describes the key terms related to TKE:

Cluster: The collection of cloud resources required to run containers, including several Tencent Cloud resources such as CVM instances and CLBs.

Pod: A group of one or more associated containers that share the same storage and network space.

Workload: A Kubernetes resource object that is used to manage the creation, scheduling, and automatic control of Pod replicas throughout the entire lifecycle.

Service: A group of microservices consisting of multiple Pods with the same configuration and the rules for accessing these Pods.

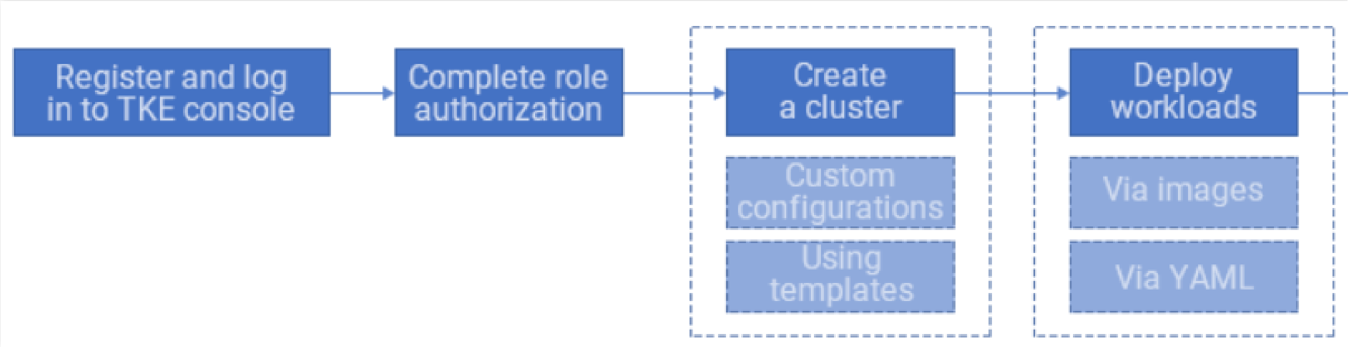
Ingress: A collection of rules for routing external HTTP(S) traffic to a Service.

Application: The features related to Helm 3.0 that are integrated in TKE. They enable you to create products and services such as Helm Chart, container images, and software services.

Image repository: It stores Docker images that are used to deploy TKE.

Directions

The following figure shows you how to use TKE:



1. Perform role authorization.

Register and log in to the TKE console, perform authorization to obtain operation permissions on relevant resources.

2. Create a cluster.

You can customize a cluster or create a cluster with a template.

3. Deploy workloads.

You can deploy workloads by deploying images or orchestrating YAML files. For more information, see Workload.

4. Manage the lifecycle of Pods by performing operations such as monitoring, upgrade, and scaling.

Pricing

TKE charges cluster management fees based on the specifications of the managed clusters, and charges cloud resources fees based on the actual usage. For more information about the billing modes and prices, see TKE Billing Overview.

Additional Services

You can purchase several CVM instances to form a TKE cluster. The containers will run on the CVMs. For more information, see Cloud Virtual Machine.

A cluster can be created in a VPC. CVM instances in the cluster can be allocated to subnets in different availability zones. For more information, see Virtual Private Cloud.

You can use CLB to automatically allocate the request traffic of clients across CVM instances and then forward the traffic to the containers running on the CVM instances. For more information, see Cloud Load Balancer.

You can use Tencent Cloud Observability Platform to monitor the operation statistics of TKE clusters and Pods. For more information, see Tencent Cloud Observability Platform.

Purchase a TKE General Cluster

TKE Billing Overview

Last updated：2022-09-16 10:31:08

Billable Items

The service fees for TKE consists of two parts, **cluster management fees** and **Tencent Cloud service resources fees**.

Cluster management fees

Managed clusters incur the cluster management fees based on their cluster models. For more information, see Cluster Management Fees.

Tencent Cloud service resources fees

Other Tencent Cloud services resources (such as CVM, CBS and CLB) created during the usage of TKE will be charged based on the billing mode for each resource. For more information, see Tencent Cloud Services Resources Fees.

Cluster Management Fees

Billing Mode

The billing mode of pay-as-you-go is usually adopted for TKE.

Billing Item	Billing Mode	Payment Method	Billing Unit
Number of clusters	Pay-as-you-go	Freeze the fees at the time of purchase, and the service is billed at an hourly basis	USD/hour

Recommendations for small clusters

If your cluster has only a small number of nodes (less than 20), we highly recommend you use TKE Serverless Cluster . With TKE Serverless cluster, you can deploy workloads and pay for actual container usage, with no need to purchase nodes and pay cluster management fees.

You can choose to migrate your existing TKE general clusters as needed in the following ways:

Conduct smooth business migration through super nodes to reduce the number of nodes in the TKE general cluster and thereby lower the cluster management fees (such fees are not charged for super nodes; for more information, see Pricing below).

Completely migrate the TKE general cluster to the TKE serverless cluster through the migration tool as instructed in Guide on Migrating Resources in a TKE Managed Cluster to an TKE Serverless Cluster. If you encounter any problems, submit a ticket for assistance.

Pricing

Note:

The unit prices are varied depending on the region. Please refer to the prices displayed in the console.

Read the Purchase Instructions carefully before you select the specification.

The billing cycle of a cluster starts when the cluster is created. You can view the cluster creation time by going to the **TKE console** > **Basic Information**.

Cluster Specification	Price (USD/hour)
L5	0.02040816
L20	0.06279435
L50	0.11459969
L100	0.19152276
L200	0.40031397
L500	0.8021978
L1000	1.47252747
L3000	2.44897959
L5000	4.40188383

Tencent Cloud Service Resources Fees

Other Tencent Cloud service resources (such as CVM, CBS and CLB) created during the usage of TKE will be charged based on each billing mode. For more information, see billing description for each resource.

Tencent Cloud Service	Documentation
CVM	CVM Billing Mode
CBS	CBS Billing Overview
CLB	CLB Billing Description

Note:

TKE is a declarative service based on Kubernetes. When you do not need CLB, CBS or other IaaS service resources created by TKE, you must delete them in TKE console, otherwise, TKE will re-create them and continue to charge fees. For example, if you delete a CLB instance in CLB console instead of in TKE console, TKE will re-create a CLB instance based on declarative APIs.

Purchase Instructions

Last updated：2022-10-25 11:20:31

Purchase Notes

Note：
If the service is unavailable due to the failure of you to abide by the following suggestions, the corresponding service downtime shall not be counted towards the service unavailability period. For more information, see TKE Service Level Agreement.

The availability of TKE clusters are relevant to the number of resources (such as Pod, ConfigMap, CRD and Event) in the cluster, as well as QPS of Get/List read operations and Patch/Delete/Create/Update write operations of the resources. To improve the cluster availability, do not initiate List-like operations for clusters with large amount of resources, and do not write too many ConfigMap/CRDs/EndPoints into the cluster.

The common **List-like operations** are as follows (take Pod resources as an example):

- Query with a label

```
kubectl get pod -l app=nginx
```

- Query for a specified namespace

```
kubectl get pod -n default
```

- Query the Pods across the cluster

```
kubectl get pod --all-namespaces
```

- Initiate a List request through client-go

```
k8sClient.CoreV1().Pods("").List(metav1.ListOptions{})
```

If you want to **query all resources in the cluster**, it is recommended that you use the **informer mechanism** to query through local cache. In some simple scenarios, you can add the ResourceVersion parameter in List to query in kube-apiserver cache. For example, `k8sClient.CoreV1().Pods("").List(metav1.ListOptions{ResourceVersion: "0"})`. Note: When you query in kube-apiserver cache, if you initiate List requests frequently to many resources, it still causes high pressure on kube-apiserver memory. It is recommended that you use this query method for low-frequency requests.

Recommended Configuration

Refer to the recommended configuration below and choose the model that best suits your requirements, so as to prevent cluster unavailability caused by heavy load of the control plane.

Assume that you want to deploy 50 nodes in a cluster and need 2,000 Pods. You need to select a model with the maximum number of nodes of 100, but not 50.

> Note：
>
> - The nodes indicate Kubernetes nodes, including CVM nodes, BM nodes and external nodes. **Supernodes are excluded**.
> - The number of Pods includes Pods in all namespaces and in any status, and excludes the Pods related to system components such as cni-agent.
> - ConfigMap does not include the Pods related to system components such as cni-agent.
> - **Maximum other resources** refers to the number of resources in the manged cluster excluding the Pods, nodes and ConfigMap. For example, for a L100 cluster, the number of the resources, such as ClusterRole, Services and Endpoints, cannot exceed 2,500 respectively.
> - It is recommended that the size of all objects under each type of resource does not exceed 800 MiB, and the size of each object does not exceed 100 KB.

Cluster specification	Max nodes	Max Pods (recommended)	Max ConfigMap (recommended)	Maximum CRDs/Maximum other resources (recommended)
L5	5	150	128	150
L20	20	600	256	600
L50	50	1,500	512	1,250
L100	100	3,000	1,024	2,500
L200	200	6,000	2,048	5,000
L500	500	15,000	4,096	10,000

Cluster specification	Max nodes	Max Pods (recommended)	Max ConfigMap (recommended)	Maximum CRDs/Maximum other resources (recommended)
L1000	1,000	30,000	6,144	20,000
L3000	3,000	90,000	8,192	50,000
L5000	5,000	150,000	10,240	100,000

Payment Overdue

Last updated：2023-07-21 11:06:21

Note：

If you are a customer of a Tencent Cloud partner, the rules regarding resources when there are overdue payments are subject to the agreement between you and the partner.

Overdue Payment

Your managed clusters will be processed as instructed below since **your account balance falls below 0**.

- Within 24 hours: Your TKE managed clusters can be used and are billed.
- After 24 hours: All managed clusters under the account are in **isolated** status and are not billed. You cannot access the API Server. Applications deployed on the nodes are not affected.

Note：

If your account balance is below 0 **before 10:00, April 1, 2022 (UTC +8)**, the **TKE managed clusters created before 10:00, March 21, 2022 (UTC +8)** are isolated after 10:00, April 1, 2022 (UTC +8).

Processing for Overdue Payments

When the managed clusters are in **isolated** status, Tencent Cloud will take the following actions:

Note：

The following description of overdue payment is only for managed clusters. For Overdue Payment of CVM ,CLB Instances, see the corresponding descriptions.

Time Since Isolation	Description
≤ 15 days	If your account is topped up to a positive balance, billing will resume and the clusters will automatically resume the running status.
	If your account is not topped up to a positive balance, the clusters will remain isolated.
＞ 15	If overdue payment of your account persists, the clusters will be deleted and cannot be recovered.

days	All nodes remaining in the clusters will be removed. We will notify the creator and all collaborators of the Tencent Cloud account through email and SMS when the clusters under the account are deleted.

Regions and Availability Zones

Last updated：2023-05-06 19:41:07

Regions

Overview

A region is the physical location of an IDC. In Tencent Cloud, regions are fully isolated from each other, ensuring cross-region stability and fault tolerance. We recommend that you choose the region closest to your end users to minimize access latency and improve access speed.
You can view the following table or use the DescribeRegions API to get a complete region list.

Characteristics

The networks of different regions are fully isolated. Tencent Cloud services in different regions **cannot communicate via a private network by default**.
Tencent Cloud services in different regions can communicate with each other through public IPs over the internet, while those in different VPCs can communicate with each other through Cloud Connect Network, which is faster and more stable.
Cloud Load Balancer currently supports intra-region traffic forwarding and being bound to CVM instances in the same region by default. If you enable the cross-region binding feature, a CLB instance can be bound to CVM instances in another region.

Availability Zones

Overview

An availability zone (AZ) is a physical IDC of Tencent Cloud with independent power supply and network in the same region. It can ensure business stability, as failures (except for major disasters or power failures) in one AZ are isolated without affecting other AZs in the same region. By starting an instance in an independent availability zone, users can protect their applications from being affected by a single point of failure.
You can view the following table or use the DescribeZones API to get a complete availability zone list.

Characteristics

Tencent Cloud services in the same VPC are interconnected via the private network, which means they can communicate using private IPs, even if they are in different availability zones of the same region.

Note

Private network interconnection refers to the interconnection of resources under the same account. Resources under different accounts are completely isolated on the private network.

China

Region	Availability Zone
South China (Guangzhou) ap-guangzhou	Guangzhou Zone 1 (Sold out) ap-guangzhou-1
	Guangzhou Zone 2 (Sold out) ap-guangzhou-2
	Guangzhou Zone 3 ap-guangzhou-3
	Guangzhou Zone 4 ap-guangzhou-4
	Guangzhou Zone 6 ap-guangzhou-6
	Guangzhou Zone 7 ap-guangzhou-7
East China (Shanghai) ap-shanghai	Shanghai Zone 1 (Sold out) ap-shanghai-1
	Shanghai Zone 2 ap-shanghai-2
	Shanghai Zone 3 ap-shanghai-3
	Shanghai Zone 4 ap-shanghai-4
	Shanghai Zone 5 ap-shanghai-5
	Shanghai Zone 8 ap-shanghai-8
East China (Nanjing) ap-nanjing	Nanjing Zone 1 ap-nanjing-1

	Nanjing Zone 2 ap-nanjing-2
	Nanjing Zone 3 ap-nanjing-3
North China (Beijing) ap-beijing	Beijing Zone 1 (Sold out) ap-beijing-1
	Beijing Zone 2 ap-beijing-2
	Beijing Zone 3 ap-beijing-3
	Beijing Zone 4 ap-beijing-4
	Beijing Zone 5 ap-beijing-5
	Beijing Zone 6 ap-beijing-6
	Beijing Zone 7 ap-beijing-7
Southwest (Chengdu) ap-chengdu	Chengdu Zone 1 ap-chengdu-1
	Chengdu Zone 2 ap-chengdu-2
Southwest (Chongqing) ap-chongqing	Chongqing Zone 1 ap-chongqing-1
Hong Kong/Macao/Taiwan (Hong Kong, China) ap-hongkong	Hong Kong Zone 1 (Nodes in Hong Kong, China can cover services in Hong Kong/Macao/Taiwan regions) (Sold out) ap-hongkong-1
	Hong Kong Zone 2 (Nodes in Hong Kong, China can cover services in Hong Kong/Macao/Taiwan) ap-hongkong-2
	Hong Kong Zone 3 (Nodes in Hong Kong, China can cover services in Hong Kong/Macao/Taiwan) ap-hongkong-3

Note:

The product is in beta test for Jinan, Hangzhou, Fuzhou, Wuhan, Changsha, and Shijiazhuang regions. To try it out, contact the sales rep for application.

Other Countries and Regions

Region	Availability Zone
Southeast Asia Pacific (Singapore) ap-singapore	Singapore Zone 1 (Singapore nodes cover services in Southeast Asia) ap-singapore-1
	Singapore Zone 2 (Singapore nodes cover services in Southeast Asia) ap-singapore-2
	Singapore Zone 3 (Singapore nodes cover services in Southeast Asia) ap-singapore-3
	Singapore Zone 4 (Singapore nodes cover services in Southeast Asia) ap-singapore-4
Southeast Asia (Jakarta) ap-jakarta	Jakarta Zone 1 (Jakarta nodes cover services in Southeast Asia) ap-jakarta-1
	Jakarta Zone 2 (Jakarta nodes cover services in Southeast Asia) ap-jakarta-2
Northeast Asia (Seoul) ap-seoul	Seoul Zone 1 (Seoul nodes cover services in Northeast Asia) ap-seoul-1
	Seoul Zone 2 (Seoul nodes cover services in Northeast Asia) ap-seoul-2
Northeast Asia (Tokyo) ap-tokyo	Tokyo Zone 1 (Tokyo nodes cover services in Northeast Asia) ap-tokyo-1
	Tokyo Zone 2 (Tokyo node AZs cover services in Northeast Asia) ap-tokyo-2
Southern Asia Pacific (Mumbai) ap-mumbai	Mumbai Zone 1 (Mumbai nodes cover services in South Asia) ap-mumbai-1
	Mumbai Zone 2 (Mumbai nodes cover services in South Asia) ap-mumbai-2
Southeast Asia Pacific (Bangkok)	Bangkok Zone 1 (Bangkok nodes cover services in Southeast Asia)

ap-bangkok	ap-bangkok-1
	Bangkok Zone 2 (Bangkok nodes cover services in Southeast Asia) ap-bangkok-1
North America (Toronto) na-toronto	Toronto Zone 1 (Toronto nodes cover services in North America) na-toronto-1
South America (São Paulo) sa-saopaulo	São Paulo Zone 1 (São Paulo nodes cover services in South America) sa-saopaulo-1
West US (Silicon Valley) na-siliconvalley	Silicon Valley Zone 1 (Silicon Valley nodes cover services in West US) na-siliconvalley-1
	Silicon Valley Zone 2 (Silicon Valley nodes cover services in West US) na-siliconvalley-2
East US (Virginia) na-ashburn	Virginia Zone 1 (Virginia nodes cover services in East US) na-ashburn-1
	Virginia Zone 2 (Virginia nodes cover services in East US) na-ashburn-2
Europe (Frankfurt) eu-frankfurt	Frankfurt Zone 1 (Frankfurt nodes cover services in Europe) eu-frankfurt-1
	Frankfurt Zone 2 (Frankfurt nodes cover services in Europe) eu-frankfurt-2

How to Select Regions and Availability Zones

When selecting a region and availability zone, take the following into consideration:

Your location, the location of your users, and the region of the CVM instances.

We recommend that you choose the region closest to your end users when purchasing CVM instances to minimize access latency and improve access speed.

Other Tencent Cloud services you use.

When you select other Tencent Cloud services, we recommend you try to locate them all in the same region and availability zone to allow them to communicate with each other through the private network, reducing access latency and increasing access speed.

High availability and disaster recovery.

Even if you have just one VPC, we still recommend that you deploy your businesses in different availability zones to prevent a single point of failure and enable cross-AZ disaster recovery.

There may be network latency among different availability zones. We recommend that you assess your business requirements and find the optimal balance between high availability and low latency.
If you need access to servers in other countries or regions, we recommend that you select an instance in those other countries or regions. If you use a CVM instance in China to access servers in other countries or regions, you may encounter much higher network latency.

Resource Availability

The following table describes which Tencent Cloud resources are global, which are regional, and which are specific to availability zones.

Resource	Resource ID Format (8-Digit String of Numbers and Letters)	Type	Description
User account	Unlimited	Globally unique	Users can use the same account to access Tencent Cloud resources around the world.
SSH key	skey-xxxxxxxx	Global	Users can use an SSH key to bind a CVM in any region under the account.
CVM instance	ins-xxxxxxxx	Available in a single availability zone of a region	CVM instances can only be created in a specific availability zone.
Custom image	img-xxxxxxxx	Available in multiple availability zones of a region	Custom images created for the instance are available to all availability zones of the same region. Use **Copy Image** to copy a custom image if you need to use it in other regions.
EIP	eip-xxxxxxxx	Available in multiple availability zones of a region	EIPs can only be associated with instances in the same region.
Security group	sg-xxxxxxxx	Available in multiple availability	Security groups can only be associated with instances in the same region. Tencent Cloud automatically creates three default security groups for users.

		zones of a region	
Cloud Block Storage	disk-xxxxxxxx	Available in a single availability zone of a region	Users can only create a Cloud Block Storage disk in a specific availability zone and attach it to instances in the same availability zone.
Snapshot	snap-xxxxxxxx	Available in multiple availability zones of a region	A snapshot created from a cloud disk can be used for other purposes (such as creating cloud disks) in this region.
Cloud Load Balancer	clb-xxxxxxxx	Available in multiple availability zones of a region	Cloud Load Balancer can be bound with CVMs in different availability zones of a single region for traffic forwarding.
VPC	vpc-xxxxxxxx	Available in multiple availability zones of a region	A VPC in one region can have resources created in different availability zones of the region.
Subnet	subnet-xxxxxxxx	Available in a single availability zone of a region	Users cannot create subnets across availability zones.
Route table	rtb-xxxxxxxx	Available in multiple availability zones of a region	When creating a route table, users need to specify a VPC. Therefore, route tables are regional as well.

Related Operations

Migrating an instance to another availability zone

Once launched, an instance cannot be migrated to another availability zone. However, you can create a custom image of the CVM instance and use the image to launch or update an instance in a different availability zone.

1. Create a custom image for the current instance. For more information, see Creating a Custom Image.

2. If the instance is on a VPC network environment and you want to retain its current private IP address after the migration, first delete the subnet in the current availability zone and then create a subnet in the new availability zone with the same IP address range. Note that a subnet can be deleted only when it contains no available instances. Therefore, all the instances in the current subnet should be migrated to the new subnet.
3. Create a new instance in the new availability zone by using the custom image you have just created. You can choose the same type and configuration as the original instance, or choose new settings. For more information, see Creating Instances via CVM Purchase Page.
4. If an elastic IP is associated with the original instance, dissociate it from the old instance and associate it with the new instance. For more information, see EIPs.
5. (Optional) If the original instance is pay-as-you-go, you can choose to terminate it. For more information, see Terminating Instances.

Copying images to other regions

Operations such as launching and viewing instances are region-specific. If the image of the instance that you need to launch does not exist in the region, copy the image to the desired region. For more information, see Copying Images.

Quotas and Limits

Last updated：2024-05-07 15:31:26

While using TKE services, you need to consider the service quota applied to TKE, CVM, and managed clusters.

TKE Quota Limit

The default TKE quota for each user is as follows. If you want to increase the quota, submit a ticket for application.

Note:

From October 21, 2019, the maximum node quota for a cluster has been adjusted to at least 5,000.

Item	Default Value	Where to Check	Quota Increase Allowed or Not
Clusters in a region	20	Bottom-right section of the TKE overview page	Yes, the upper limit of quota is unlimited
Nodes in a cluster	5,000		
Image namespaces in a region	10		
Image repositories in a region	500		
Tags of an image	100		

CVM Quota Limit

CVM instances generated by TKE are subject to purchase limits. For more information, see Purchase Limits. If you need more quotas than the default, submit a ticket for application.

Item	Default Value	Where to Check	Quota Increase Allowed or Not
Pay-as-you-go CVM instances in an AZ	30 or 60	CVM Instances page - Resources in each region	Yes

Cluster Configuration Limit

Note:

Cluster configuration limits the cluster size and cannot be modified currently.

Item	IP Address Range	Affected Scope	Where to Check	Modification Allowed or Not
VPC network - Subnet	Custom	Number of nodes that can be added to the subnet	VPC subnet list page for the cluster - Number of available IP addresses	No You can use a new subnet
Container CIDR block	Custom	Maximum number of nodes per cluster Maximum number of services per cluster Maximum number of Pods per node	Basic information page for the cluster - Container CIDR block	No

K8s Resource Quota Description

Note:

The following quotas are automatically applied from April 30, 2022 (UTC +8) and cannot be adjusted. **You can increase the resource quota by upgrading the cluster model**.

To adjust your quota, submit a ticket for application.

Run the following command to check the quota:

```
kubectl get resourcequota tke-default-quota -o yaml
```

To check the `tke-default-quota` object of a specified namespace, add `--namespace` to specify the namespace.

Note:

Other K8s resource limit means that the number of all K8s resources in the cluster except Pod, Node, and ConfigMap **cannot** exceed this value. For example, for an L100 cluster, the number of ClusterRole, Service, Endpoint, and other K8s resources **cannot** exceed 10,000.

CRD refers to **the sum of all CRDs** in the cluster. If the number of some CRDs increases, the number of other CRDs will decrease.

Cluster Model	Pods	ConfigMap	CRDs/Other K8s Resources
L5	600	256	1,250
L20	1,500	512	2,500
L50	3,000	1,024	5,000
L100	6,000	2,048	10,000
L200	15,000	4,096	20,000
L500	30,000	6,144	50,000
L1000	90,000	8,192	100,000
L3000	150,000	10,240	150,000
L5000	200,000	20,480	200,000

Namespace quota

By default, **each namespace has the same margin (margin = quota for the current cluster level - amount already used by the entire cluster). If you create resources in a namespace, the margin will decrease, and the amount available in other namespaces will decrease accordingly after a certain period of time.**

If you want to customize the allocation ratio, you can create a `tke-quota-config` ConfigMap under `kube-system` to specify the **margin** allocation ratio for each namespace.

The following example sets the **margin** allocation ratio to `50%` for the `default` namespace, `40%` for the `kube-system` namespace, and `10%` for the rest of the namespaces. **If the sum of the set percentages exceeds `100%` , TKE considers the ratio invalid and will use the default allocation policy**.

```
apiVersion: v1
data:
  default: "50"
  kube-system: "40"
kind: ConfigMap
metadata:
  name: tke-quota-config
  namespace: kube-system
```

Container Node Disk Settings

Last updated：2019-09-02 16:35:15

Description

When creating or scaling a TKE cluster, you can set the type and size of the system disks and data disks of the container nodes to meet your actual business needs.

Suggestions

1. The directory of the container is stored in the system disk. We recommend creating a system disk with a capacity of 50 GB.
2. If you have specific requirements for the system disk, you can move the Docker's directory to the data disk when initializing the cluster.

Notes on the Public IP of a TKE Node

Last updated：2023-05-23 11:38:21

If you don't want to avoid exposing your company's IP while accessing the public network, you can use Tencent Cloud NAT Gateway. This document describes how to access the public network via an NAT gateway.

Public IP

When a cluster is created, public IPs are assigned to the nodes in the cluster by default. With these public IPs, you can:

- Log in to the nodes in the cluster.
- Access services on the public network.

Public Network Bandwidth

When a service is created on the public network, the public network CLB uses the bandwidth and traffic of the nodes. If the public network service is required, the nodes need to have public network bandwidth. You can choose not to purchase public network bandwidth if it is not needed.

NAT Gateway

The CVM instance is not bound to an EIP, and all the traffic accessing the internet is forwarded via an NAT gateway. In this way, the traffic accessing the internet of the instance is forwarded to the NAT gateway over the private network. This means that the traffic is not subject to the upper limit of public network bandwidth specified when you purchase the instance, and the traffic generated from the NAT gateway does not occupy the public network bandwidth egress of the instance. To access the internet via an NAT gateway, follow the steps below:

Step 1. Create an NAT gateway

1. Log in to the VPC Console and click **NAT Gateway** in the left sidebar.
2. On the NAT gateway management page, click **Create**.
3. In the **Create an NAT Gateway** window that pops up, enter the following parameters.

- Gateway Name: Custom.
- Network: Select the VPC of the NAT gateway service;

- Gateway Type: Select based on actual needs. The type of the gateway can be changed after it is created.
- Outbound Bandwidth Cap: Set based on actual needs.
- Elastic IP: Assign an EIP to the NAT gateway. You can choose an existing EIP or purchase a new one.

4. Click **Create** to complete the creation of the NAT gateway.

> Note：
> The rental fee of 1 hour will be frozen during the creation of the NAT gateway.

Step 2. Configure the route table associated with the subnet

> Note：
> After the NAT gateway is created, you need to configure the routing rules on the route table page in the VPC Console to redirect the subnet traffic to the NAT gateway.

1. Click **Route Table** in the left sidebar.
2. In the route table list, click the route table ID/name associated with the subnet that needs to access the internet.
3. In the "Routing Policy" section, click **+ New routing policies**.
4. In the **Add routing** page, enter the **Destination**, select **NAT gateway** for **Next Hop Type**, and select the ID of the created NAT gateway for **Next Hop**.
5. Click **OK**.
   Now, the traffic generated when the CVM instance in the subnet associated with the route table accesses the internet will be directed to the NAT gateway.

Other Solutions

Solution 1. Use an EIP

The CVM instance is only bound with an EIP but does not use an NAT gateway. With this solution, all the traffic of the instance accessing the internet goes out through the EIP and is subject to the upper limit of public network bandwidth specified when you purchase the instance. The fees for accessing the internet are charged based on the billing method of the instance's network.
For more information, see Elastic Public IP.

Solution 2. Use both an NAT gateway and an EIP

If both an NAT gateway and an EIP are used, all the traffic of the CVM instance accessing the internet is forwarded to the NAT gateway over the private network, and the response packets are returned to the instance through the NAT gateway. This means that the traffic is not subject to the upper limit of public network bandwidth specified when you purchase the instance, and the traffic generated by the NAT gateway does not occupy the public network bandwidth egress of the instance. If the traffic from the internet proactively accesses the EIP of the instance, the response packets of the instance are all returned through the EIP. In this case, the resulting outbound public network traffic is subject to the upper limit of public network bandwidth specified when you purchase the instance. The fees for accessing the public network are charged based on the billing method of the instance's network.

> Note：
> If the bandwidth package (BWP) feature is activated in your account, fees of the outbound traffic generated by the NAT gateway will be deducted from the BWP (which means the network traffic will not be repeatedly billed). It is recommended that you limit the outbound bandwidth of the NAT gateway so as to avoid high BWP fees due to excessive outbound bandwidth.

TKE Security Group Settings

Last updated：2023-10-24 14:31:28

Security is a matter of utmost importance. Tencent Cloud considers security as a top priority in product design and requires all its products to be fully isolated and provides multiple layers of security protection with its basic network. TKE is a typical example. It adopts VPC as the underlying network of container services. This document describes the best practice of security group usage in TKE to help you select the most appropriate security group policy.

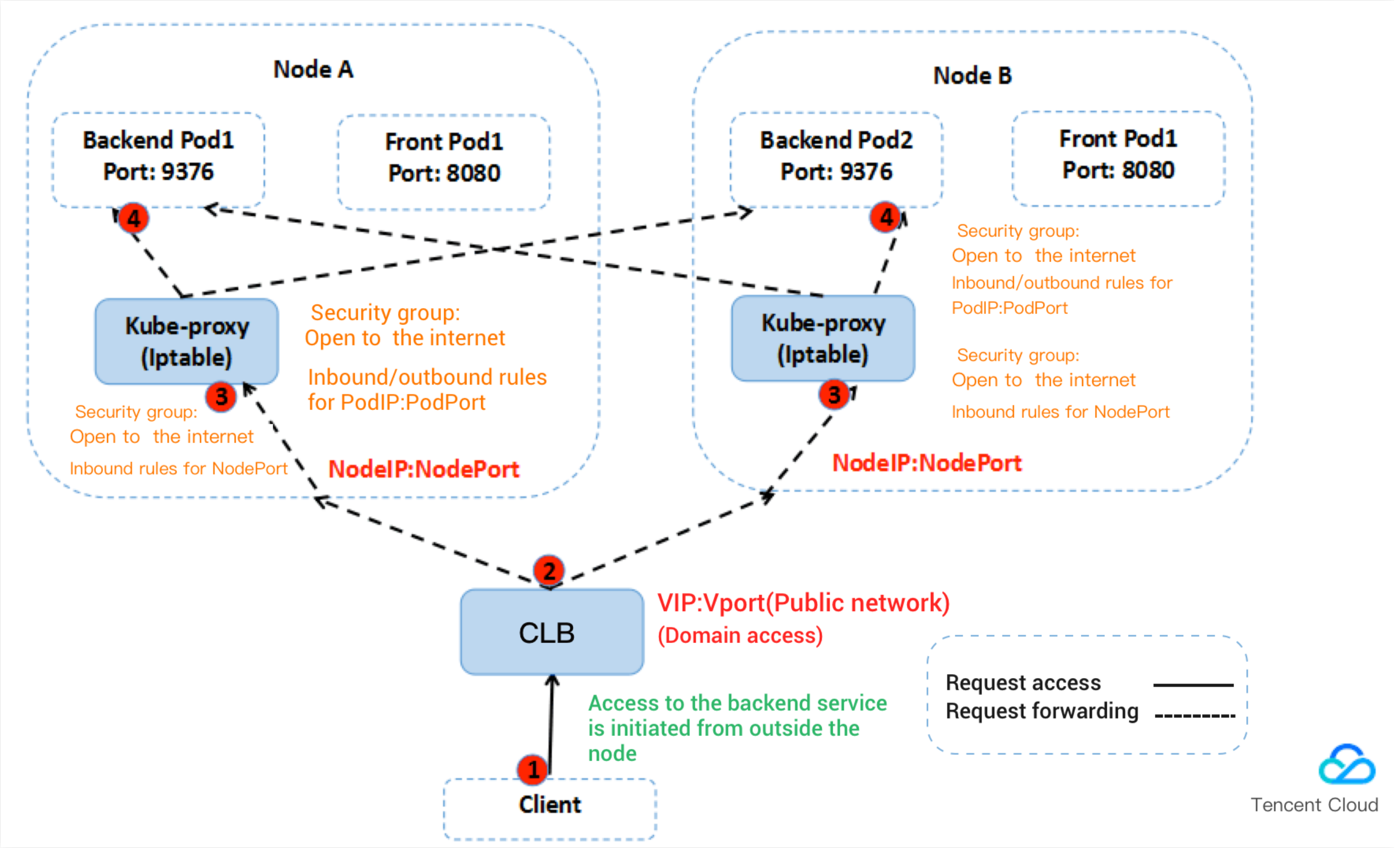
Security Groups

A security group is a virtual firewall capable of filtering stateful packets. As an important network security isolation means provided by Tencent Cloud, it can be used to configure network access control for one or more CVM instances. For more information, see Security Group.

How to Select a Security Group for TKE

- In a container cluster, service pods are distributed on different nodes. We recommend that you bind all CVM instances in one cluster to the same security group and do not add non-clustered CVMs to a security group for a cluster.
- A security group only grants the minimum permission externally.
- You must enable the following rules for using TKE:
- Open the container pod network and the cluster node network to the Internet.
  When a node receives a service access request, the node forwards the request to a service pod according to the iptables rule configured by the kube-proxy module. If the service pod is on another node, cross-node access occurs. For example, the destination IP addresses of the access request include the IP address of the service pod, IP addresses of other nodes in the cluster, and the IP address of the cluster’s cbr0 bridge on the node. In this case, the container pod network and the cluster node network on the peer node must be open to the Internet.
- If clusters in the same VPC need to communicate with each other, you must open the container networks and node networks of the corresponding clusters to the Internet.
- Open port 22 to the Internet if SSH login is required.
- Open ports 30000 to 32768 on nodes to the Internet.
  In the access path, you must use a load balancer to forward data packets to NodeIP:NodePort of the container cluster. NodeIP is the CVM instance IP of any node in the cluster. NodePort is assigned by the container cluster by default when the service is created. NodePort ranges from 30000 to 32768.

The following figure uses service access from the public network as an example.



Default Security Group Rules for TKE

Default security group rules for node

Some ports must be opened to the Internet to ensure normal communication between cluster nodes. To avoid cluster creation failures due to binding to invalid security groups, TKE provides default security group rules, as described in the following table.

> Note：
>
> If the current default security group cannot meet your service requirements and you have created a cluster bound to this security group, you can view and modify the security group rules for the cluster. For more information, please see Managing Security Group Rules.

Inbound rules

Protocol	Port Number	Source IP Address	Rule	Description
All	All	CIDR of the container network	Allow	Enable the communication between pods in the container network.
All	All	CIDR of the cluster network	Allow	Enable the communication between nodes in the cluster network.
tcp	30000 - 32768	0.0.0.0/0	Allow	Open NodePort to the Internet (Services in LoadBalancer type need to be forwarded through NodePort).
udp	30000 - 32768	0.0.0.0/0	Allow	Open NodePort to the Internet (Services in LoadBalancer type need to be forwarded through NodePort).
ICMP	-	0.0.0.0/0	Allow	Enable the support for Internet Control Message Protocol (ICMP) and ping operations.

Outbound rules

Protocol	Port Number	Source IP Address	Rule
All	All	0.0.0.0/0	Allow

Note：

- To customize outbound rules, you need to open the node IP range and container IP range.
- If you configure this rule for container nodes, the services in the cluster can be accessed using different access methods.
- For more information on how to access a service in a cluster, please see "Service Access" in Overview.

Default security group rules for master node in self-deployed cluster

When you create a self-deployed cluster, the default TKE security group will be bound to the master node by default to reduce the risks where the master node cannot communicate with other nodes normally or Services cannot be accessed normally. The configuration rules of default security group are as detailed below:

Note：

The security group creation permission is inherited from the TKE service role. For more information, see Description of Role Permissions Related to Service Authorization.

Inbound rules

Protocol	Port	IP Range	Policy	Remarks
ICMP	All	0.0.0.0/0	Supported	Ping operations are supported.
TCP	30000–32768	Cluster network CIDR	Supported	It is used to open NodePort to the Internet (Services in LoadBalancer type need to be forwarded through NodePort).
UDP	30000–32768	Cluster network CIDR	Supported	It is used to open NodePort to the Internet (Services in LoadBalancer type need to be forwarded through NodePort).
TCP	60001, 60002, 10250, 2380, 2379, 53, 17443, 50055, 443, 61678	Cluster network CIDR	Supported	It is used to open API Server communication to the Internet.
TCP	60001, 60002, 10250, 2380, 2379, 53, 17443	Container network CIDR	Supported	It is used to open API Server communication to the internet.
TCP	30000–32768	Container network CIDR	Supported	It is used to open NodePort to the Internet (Services in LoadBalancer type need to be forwarded through NodePort).
UDP	30000–32768	Container network CIDR	Supported	It is used to open NodePort to the Internet (Services in LoadBalancer type need to be forwarded through NodePort).

Protocol	Port	IP Range	Policy	Remarks
UDP	53	Container network CIDR	Supported	It is used to open CoreDNS communication to the internet.
UDP	53	Cluster network CIDR	Supported	It is used to open CoreDNS communication to the internet.

Outbound rules

Protocol	Port Number	Source IP Address	Rule
All	All	0.0.0.0/0	Allow

Project of New Resources

Last updated：2022-08-26 11:18:00

Overview

To conduct financial accounting by projects, please take the following into the consideration:

1. Clusters are not project-specific, but CVMs, load balancers and other resources in a cluster are project-specific.
2. Project of New-added Resource: Only resources newly added to the cluster are allocated to the project.

Notes

1. We recommend you allocate all the resources in a cluster to the same project.
2. If you need to distribute the CVMs in a cluster to different projects, go to the CVM Console to migrate projects. For more information, see Adjusting Project Configuration.
3. If CVMs belong to different projects, they belong to different `security group instances` . Try to configure the same `security group rules` for the CVMs in the same cluster. For more information, see Changing Security Group.

High-risk Operations of Container Service

Last updated：2020-10-10 12:07:57

When deploying or running business, you may trigger high-risk operations at different levels, leading to service failures to different degrees. To help you estimate and avoid operational risks, this document describes the consequences of the high-risk operations and corresponding solutions. Below you can find the high-risk operations you may trigger when dealing with clusters, networking and load balancing, logs, and cloud disks.

Clusters

Category	High-risk Operation	Consequence	Solution
Master and etcd nodes	Modifying the security groups of nodes in a cluster	Master node may become unavailable	Configure security groups as recommended by Tencent Cloud
	Node expires or is terminated	The master node becomes unavailable	Unrecoverable
	Reinstalling operating system	Master components get deleted	Unrecoverable
	Upgrading master or etcd component version on your own	Cluster may become unavailable	Roll back to the original version
	Deleting or formatting core directory data such as node `/etc/kubernetes`	The master node becomes unavailable	Unrecoverable
	Changing node IP	The master node becomes unavailable	Change back to the old IP
	Modifying parameters of core components, e.g. etcd, kube-apiserver, docker, etc., on your own	Master node may become unavailable	Configure parameters as recommended by Tencent Cloud
	Changing master or etcd certificate on your own	Cluster may become unavailable	Unrecoverable
Worker node	Modifying the security groups of nodes in a cluster	Nodes may become unavailable	Configure security groups as recommended by Tencent Cloud

	Node expires or is terminated	The node becomes unavailable	Unrecoverable
	Reinstalling operating system	Node components get deleted	Remove the node and add it back to the cluster
	Upgrading node component version on your own	Node may become unavailable	Roll back to the original version
	Changing node IP	Node becomes unavailable	Change back to the old IP
	Modifying parameters of core components, e.g. etcd, kube-apiserver, docker, etc., on your own	Node may become unavailable	Configure parameters as recommended by Tencent Cloud
	Modifying operating system configuration	Node may become unavailable	Try to restore the configurations or delete the node and purchase a new one
Others	Modifying permissions in CAM	Some cluster resources, such as cloud load balancers, may not be able to be created	Restore the permissions

Networking and Load Balancing

High-risk Operation	Consequence	Solution
Modifying kernel parameters `net.ipv4.ip_forward=0`	Network not connected	Modify kernel parameters to `net.ipv4.ip_forward=1`
Modifying kernel parameter `net.ipv4.tcp_tw_recycle = 1`	NAT exception	Modify kernel parameter `net.ipv4.tcp_tw_recycle = 0`
Container CIDR’s UDP port 53 is not opened to the Internet in the security group configuration of the node	In-cluster DNS cannot work normally	Configure security groups as recommended by Tencent Cloud
Modifying or deleting LB tags added in TKE	A new LB is purchased	Restore the LB tags
Creating custom listeners in TKE-managed LB	Modification gets	Automatically create listeners

through LB console	reset by TKE	through service YAML
Binding custom backend rs in TKE-managed LB through LB console		Prohibit manual binding of backend rs
Modifying certificate of TKE-managed LB through LB console		Automatically manage certificate through ingress YAML
Modifying TKE-managed LB listener name through LB console		Prohibit modification of TKE-managed LB listener name

Logs

High-risk Operation	Consequence	Solution	Notes
Deleting the `/tmp/ccs-log-collector/pos` directory of the host	Log gets collected again	None	Files in Pod record where they are collected
Deleting the `/tmp/ccs-log-collector/buffer` directory of the host	Log gets lost	None	Buffer contains log cache file

Cloud Disks

High-risk Operation	Consequence	Solution
Manually unmounting cloud disks through console	Writing to Pod reports IO errors	Delete the mount directory of the node and reschedule the Pod
Unmounting disk mounting path on the node	Pod gets written to the local disk	Re-mount the corresponding directory onto Pod
Directly operating CBS block device on the node	Pod gets written to the local disk	None

Deploying Containerized Applications in the Cloud

Last updated：2023-05-06 19:41:07

Overview

All cloud users want their migrations to the cloud to be efficient, stable, and highly available, but this depends on system availability, data reliability, and OPS stability. This document describes the check items for deploying containerized applications to the cloud from three perspectives: evaluation item, impact, and reference. This will help ensure you experience a smooth and efficient migration to Tencent Kubernetes Engine (TKE).

Check Items

System availability

Category	Item	Type	Impact	Reference
Cluster	Before creating a cluster, plan the node network and container network to suit your application scenario to prevent restricted capacity scaling in the future.	Network planning	If you have small-scale subnets or container IP ranges, your cluster may support fewer nodes than your application actually needs.	Network Planning Container Network Overview
	Before creating a cluster, review your planning of direct connect, peering connection, container IP ranges, and subnet IP ranges to prevent IP range conflicts and impacts on your applications.	Network planning	For simple networking scenarios, follow the instructions on the page to configure cluster-related IP ranges to avoid conflicts. For complex networking scenarios, such as peering connection, direct connect, and VPN, improper network planning can affect the normal communication within your application.	-
	When you create a cluster,	Deployment	Security groups provide an	TKE Security

	a new security group is automatically bound to the cluster. You can also set custom security group rules to meet the needs of your application.		important means of security isolation. Improper security policy configuration may lead to security-related risks, service connectivity issues, and other problems.	Group Settings
	As the runtime components currently supported by TKE, Containerd and Docker suit different scenarios. When creating a cluster, select the appropriate container runtime component according to your application scenarios.	Deployment	Once the cluster is created, modifications to the runtime component and version only take effect to new nodes that are not assigned to any node pool. Existing nodes are not affected.	How to Choose Containerd and Docker
	By default, Kube-proxy uses iptables to balance the load between Service and Pod. When creating a cluster, you can quickly enable IPVS for traffic distribution and load balancing.	Deployment	You can enable IPVS when creating a cluster. It will take effect for the entire cluster and cannot be disabled.	Enabling IPVS for a Cluster
	When creating a cluster, choose the independent cluster mode or managed cluster mode as needed.	Deployment	The Master and Etcd of the managed cluster are not user resources and are managed and maintained by Tencent Cloud's technical team. You cannot modify the deployment scale and service parameters of Master and Etcd. If you do need to modify them, choose the independent deployment mode.	Cluster Overview Cluster Hosting Modes Introduction
Workload	When creating a workload, set the CPU and memory limits to improve the robustness of your application.	Deployment	When multiple applications are deployed on one node, if an application without resource upper and lower limits encounters a resource leak, exceptions will occur in other applications on the	Setting the Resource Limit of Workload

			same node due to the lack of resources, and they will report monitoring information errors.	
	When creating a workload, you can configure container health checks, which are "liveness check" and "readiness check".	Reliability	If container health checks are not configured, when application exceptions occur, the pod will not be able to detect them to automatically restart the application for recovery. In this case, while the pod seems normal, the application in the pod will behave abnormally.	Setting the Health Check for a Workload
	When creating a service, choose the appropriate service access method as needed. Four access methods are currently supported: Via Internet, Intra-cluster, Via VPC, and Node Port Access.	Deployment	An improper access method may cause access logic confusion and waste resources inside and outside the service.	Service Management
	When creating a workload, do not set the number of pod replicas to 1. Set a node scheduling policy based on the needs of your application.	Reliability	Setting the number of pod replicas to 1 incurs service exceptions when node exceptions or pod exceptions occur. To ensure that your pod can be scheduled successfully, ensure that the node has resources available for container scheduling after setting the scheduling rules.	Adjusting Pod quantity Setting the Scheduling Rule for a Workload

Data reliability

Category	Item	Type	Impact	Reference
Container data persistence	Apply pod data storage and choose an appropriate volume type as needed.	Reliability	When a node fails to be restored following an exception, the data in the local disk cannot be restored. However, cloud storage can provide extremely high data reliability in this situation.	Instructions for Other Storage Volumes

Ops stability

Category	Item	Type	Impact	Reference
Engineering	Check whether the quotas of resources such as CVMs, VPCs, subnets, and CBS disks can meet customer needs.	Deployment	Insufficient quotas will cause resource creation to fail. If you have enabled auto scaling, ensure that you have sufficient quotas for your Tencent Cloud services.	Quotas and Limits Quota Limit
	We recommend that you do not modify the kernel parameters, system configurations, versions of cluster core components, security groups, and LB parameters on the nodes in your cluster.	Deployment	This may cause TKE cluster features or Kubernetes components installed on the node to fail, making the node unavailable for application deployment.	High-risk Operations of Container Service
Proactive Ops	TKE provides multidimensional monitoring and alarm features, along with basic resource monitoring provided by Cloud Monitor, to provide more refined metrics. Configuring monitoring and alarm helps you receive prompt alarms and locate faults in case of exceptions.	Monitoring	If the monitoring and alarm features are not configured, no normal standard can be established for container cluster performance, and alarms will not be promptly received when an exception occurs. In this case, you will have to manually inspect your environment.	Overview of Monitoring and Alarms Viewing Monitoring Data List of Monitoring and Alarm Metrics

Kubernetes API Operation Guide

Last updated：2020-11-03 17:05:17

Overview

This document describes how to use Kubernetes APIs to perform operations in Tencent Kubernetes Engine (TKE) clusters. For example, you can use the APIs to view all namespaces in a cluster, view all pods in a specified namespace, and add, delete, or query a pod in the specified namespace.

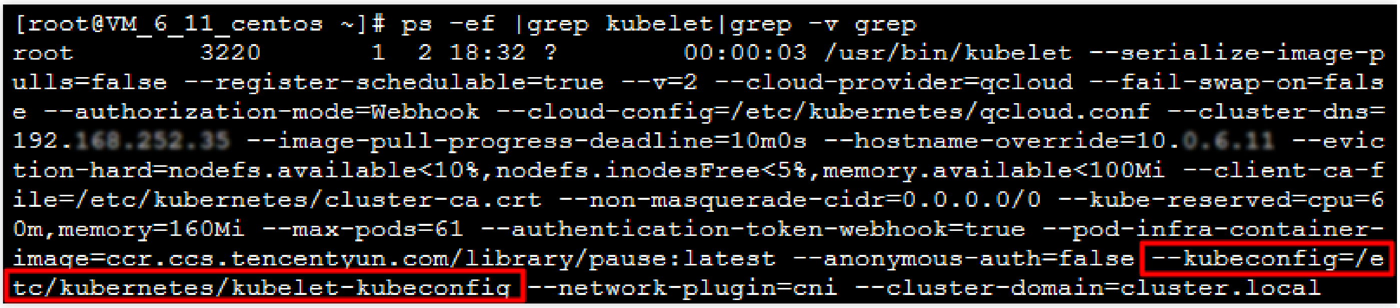
Directions

Obtaining the kubeconfig cluster access credential

1. Log in to the cluster node by referring to Log into Linux Instance Using Standard Login Method.
2. Run the following command to obtain the location of the cluster access credential (kubeconfig) file:

```
ps -ef |grep kubelet|grep -v grep
```

The following figure shows the output of the command, where the location of the access credential file is `/etc/kubernetes/kubelet-kubeconfig`.



```
[root@VM_6_11_centos ~]# ps -ef |grep kubelet|grep -v grep
root         3220        1  2 18:32 ?        00:00:03 /usr/bin/kubelet --serialize-image-p
ulls=false --register-schedulable=true --v=2 --cloud-provider=qcloud --fail-swap-on=fals
e --authorization-mode=Webhook --cloud-config=/etc/kubernetes/qcloud.conf --cluster-dns=
192.[redacted] --image-pull-progress-deadline=10m0s --hostname-override=10.[redacted] --evic
tion-hard=nodefs.available<10%,nodefs.inodesFree<5%,memory.available<100Mi --client-ca-f
ile=/etc/kubernetes/cluster-ca.crt --non-masquerade-cidr=0.0.0.0/0 --kube-reserved=cpu=6
0m,memory=160Mi --max-pods=61 --authentication-token-webhook=true --pod-infra-container-
image=ccr.ccs.tencentyun.com/library/pause:latest --anonymous-auth=false --kubeconfig=/e
tc/kubernetes/kubelet-kubeconfig --network-plugin=cni --cluster-domain=cluster.local
```

3. Run the following command to go to the `kubernetes` directory:

```
cd /etc/kubernetes
```

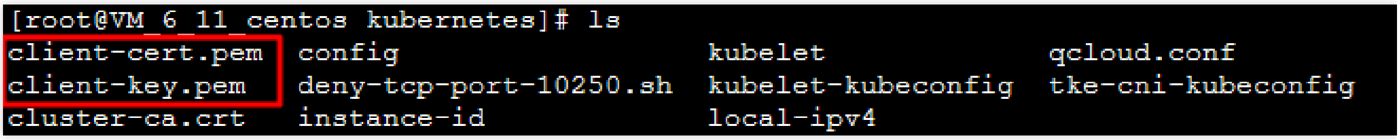
4. Run the following commands in sequence to obtain the CA, key, and apiserver information from the `kubeconfig` file, respectively:

```
cat ./kubelet-kubeconfig |grep client-certificate-data | awk -F ' ' '{print
$2}' |base64 -d > client-cert.pem
```

```
cat ./kubelet-kubeconfig |grep client-key-data | awk -F ' ' '{print $2}' |base6
4 -d > client-key.pem
```

```
APISERVER=`cat ./kubelet-kubeconfig |grep server | awk -F ' ' '{print $2}'`
```

Run the `ls` command. Then, you can find the generated `client-cert.pem` and `client-key.pem` files in the kubernetes directory, as shown in the following figure:



```
[root@VM_6_11_centos kubernetes]# ls
client-cert.pem  config                   kubelet             qcloud.conf
client-key.pem   deny-tcp-port-10250.sh   kubelet-kubeconfig  tke-cni-kubeconfig
cluster-ca.crt   instance-id              local-ipv4
```

Calling Kubernetes APIs by using CURL commands

1. Run the following command to view all namespaces in the current cluster:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/api/v1/namespace
s
```

> **Note：**
>
> If an error stating insufficient permissions occurs when you run the `curl` command, you can resolve the error by referring to Granting cluster permissions.

2. Run the following command to view all pods in the kube-system namespace:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/api/v1/namespace
s/kube-system/pods
```

Managing pod lifecycles

> **Note：**
>
> The files created in the following steps and their content are for demonstration purposes only. You can customize them based on your actual requirements.

Creating a pod in the JSON format

1. Run the following command to create and open a JSON file:

```
vim nginx-pod.json
```

2. Copy the following content into the JSON file:

```
{
"apiVersion":"v1",
"kind":"Pod",
"metadata":{
"name":"nginx",
"namespace": "default"
},
"spec":{
"containers":[
{
"name":"nginx-test",
"image":"nginx",
"ports":[
{
"containerPort": 80
}
]
}
]
}
}
```

3. Run the following command to create a pod:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/api/v1/namespaces/default/pods -X POST --header 'content-type: application/json' -d@nginx-pod.json
```

Creating a pod in the YAML format

1. Run the following command to create and open a YAML file:

```
vim nginx-pod.json
```

2. Copy the following content into the YAML file:

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
namespace: default
spec:
containers:
- name: nginx-test
```

```
image: nginx
ports:
- containerPort: 80
```

3. Run the following command to create a pod:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/api/v1/namespaces/default/pods -X POST --header 'content-type: application/yaml' --data-binary @nginx-pod.yaml
```

Querying the status of a pod

Run the following command to query the status of a pod:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/api/v1/namespaces/default/pods/nginx
```

Querying the logs of a pod

Run the following command to query the logs of a pod:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/api/v1/namespaces/default/pods/nginx/log
```

Querying the metrics of a pod

Run the following command to query the metrics of a pod through the metric-server API:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/apis/metrics.k8s.io/v1beta1/namespaces/default/pods/nginx
```

Deleting a pod

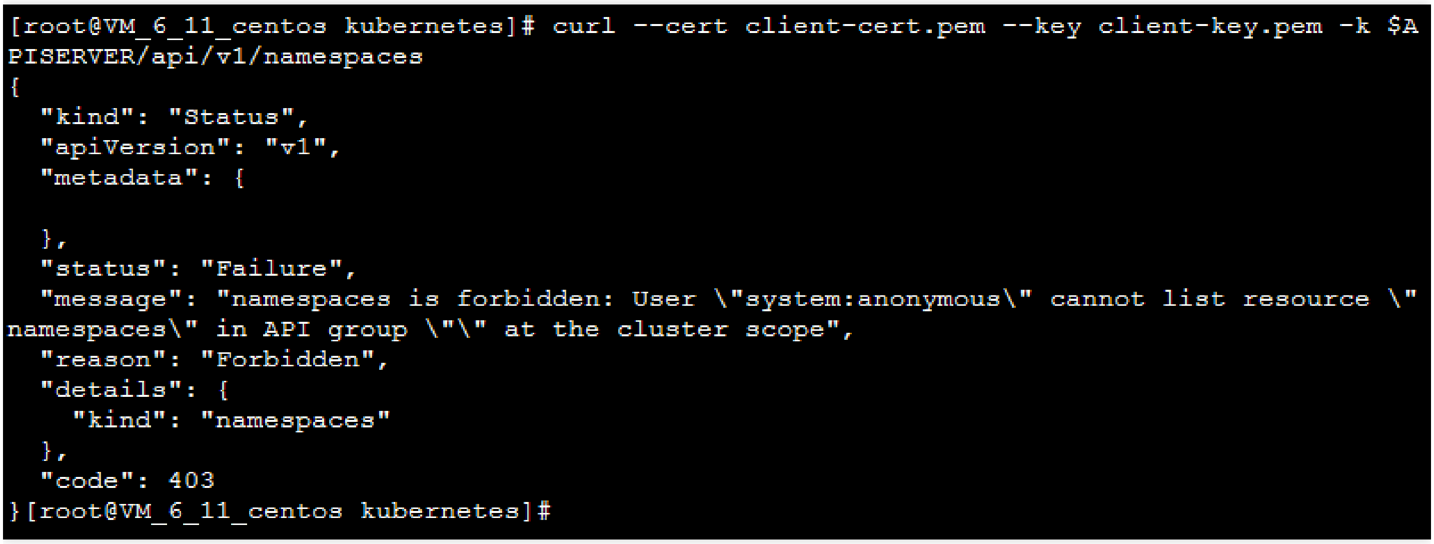
Run the following command to delete a pod:

```
curl --cert client-cert.pem --key client-key.pem -k $APISERVER/api/v1/namespaces/default/pods/nginx -X DELETE
```

Relevant Operations

Granting cluster permissions

If the following error occurs when you run the `curl` command, you must grant cluster access permissions.



```
[root@VM_6_11_centos kubernetes]# curl --cert client-cert.pem --key client-key.pem -k $A
PISERVER/api/v1/namespaces
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "namespaces is forbidden: User \"system:anonymous\" cannot list resource \"
namespaces\" in API group \"\" at the cluster scope",
  "reason": "Forbidden",
  "details": {
    "kind": "namespaces"
  },
  "code": 403
}[root@VM_6_11_centos kubernetes]#
```

You can perform authorization by using the following two methods:

- Method 1 (recommended): perform RBAC authorization in the TKE console. For more information, see Authorizing by using preset identities and Authorizing by using custom policies.
- Method 2: Run the following command to perform authorization. We recommend that you not grant an account the cluster-admin permissions in a production cluster.

```
kubectl create clusterrolebinding cluster-system-anonymous --clusterrole=cluster-
admin --user=system:anonymous
```

Open Source Components

Last updated：2020-07-10 16:39:32

tencentcloud-cloud-controller-manager

`tencentcloud-cloud-controller-manager` is the Cloud Controller Manager implementation for the Tencent Kubernetes Engine (TKE). This component implements the following features in Kubernetes clusters built by Tencent Cloud CVMs:

- nodecontroller: updates relevant `addresses` information for Kubernetes nodes.
- routecontroller: creates routes within pod IP ranges in a VPC.
- servicecontroller: creates a CLB when a load balancer-type service is created in a cluster.

For more information about installation and usage, see Kubernetes Cloud Controller Manager for Tencent Cloud on GitHub.

kubernetes-csi-tencentcloud

`kubernetes-csi-tencentcloud` is a plugin for the Tencent Cloud CBS service that complies with CSI standards. This component allows you to use Cloud Block Storage in Kubernetes clusters built by Tencent Cloud CVMs.
This plugin is suitable for plugins that use CBS when building a Kubernetes cluster, which is different from the `provisioner: cloud.tencent.com/qcloud-cbs` that comes with the TKE cluster.

For more information about installation and usage, please see kubernetes-csi-tencentcloud on GitHub.

Permission Management

Overview

Last updated：2022-06-10 16:48:46

If you have multiple users managing the TKE service, and they all share your Tencent Cloud account access key, you may face the following problems:

- The risk of your key being compromised is high since multiple users are sharing it.
- Your users might introduce security risks from maloperations due to the lack of user access control.

To solve these problems, create sub-accounts for other users and these users use sub-accounts to log in and manage their services. By default, sub-accounts do not have permission to use TKE. You need to create a policy to grant the required permissions to sub-accounts.

Overview

Cloud Access Management (CAM) is a web-based Tencent Cloud service that helps you securely manage and control access permissions of your Tencent Cloud resources. Using CAM, you can create, manage, and terminate users (groups), and control the Tencent Cloud resources that can be used by the specified user through identity and policy management.

When using CAM, you can associate a policy with a user or user group to allow or forbid them to use specified resources to complete specified tasks. For more information on CAM policies, see Element Reference. For more information on how to use CAM policies, see Policy.
You can skip this section if you don't need to manage permissions to CAM resources for sub-accounts. This will not affect your understanding and use of the other sections of the document.

Getting Started

A CAM policy must authorize or deny the use of one or more TKE operations. At the same time, it must specify the resources that can be used for the operations (which can be all resources or partial resources for certain operations). A policy can also include the conditions set for the manipulated resources.

Some TKE APIs do not support resource-level permissions. This means that you cannot specify certain resources when performing such API operations, and these operations are performed on all the resources.

Description of Role Permissions Related to Service Authorization

Last updated：2022-05-09 11:40:29

When you use Tencent Kubernetes Engine (TKE), you need to authorize services to use relevant cloud resources. Each scenario usually contains policies that are defined for different roles in advance. The main roles involved are `TKE_QCSRole` and `IPAMDofTKE_QCSRole` . This document introduces the details of each authorization policy, and the authorization scenarios and authorization steps for each role.

> Note：
> The sample role in this document does not contain the authorization policy related to container image repositories. For more information about TKE image related permissions, see TKE Image Registry Resource-level Permission Settings.

TKE_QCSRole

After TKE is activated, Tencent Cloud grants your account the permissions of the role `TKE_QCSRole` , which is associated with multiple preset policies by default. To obtain relevant permissions, you need to perform the corresponding preset policy authorization operations in specific authorization scenarios. After these operations are completed, the corresponding policy will appear in the role's list of authorized policies. The preset policies associated with `TKE_QCSRole` by default include:

The default associated preset policies

- `QcloudAccessForTKERole` : The permission for TKE to access cloud resources
- `QcloudAccessForTKERoleInOpsManagement` : The permission for Ops management, including the log service

Other associated preset policies

- `QcloudAccessForTKERoleInCreatingCFSStorageclass` : The permission for TKE to operate on Cloud File Storage (CFS), including adding/deleting/querying CFS systems, and querying the mount targets of a file system.
- `QcloudCVMFinanceAccess` : CVM finance permission

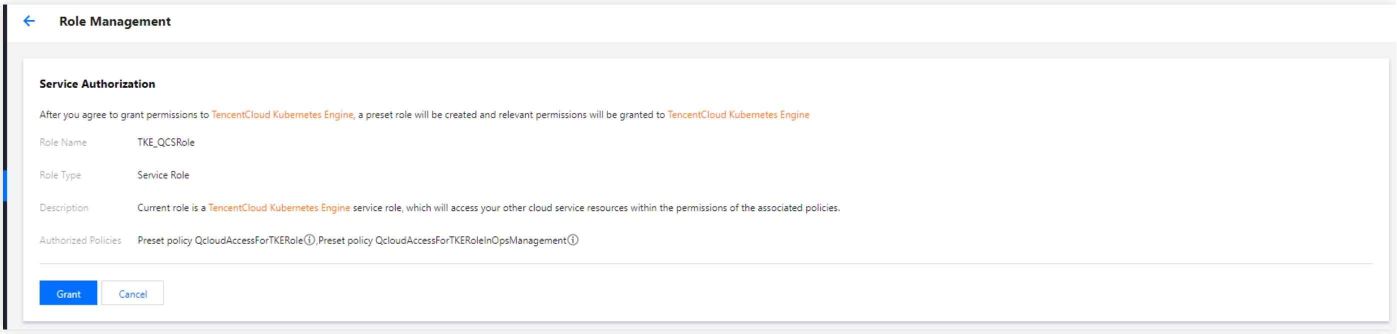
Preset policy QcloudAccessForTKERole

Authorization scenario

When you log in to the TKE console for the first time after registering and logging in to a Tencent Cloud account, you need to go to the "Cloud Access Management" page to grant the current account TKE permissions for operating on CVMs, CLBs, CBS, and other cloud resources.

Authorization steps

1. Log in to the TKE console and click **Cluster** in the left sidebar to pop up the **Service authorization** window.
2. Click **Go to Cloud Access Management** to enter the **Role management** page.
3. Click **Grant** to complete authentication.



Role Management

Service Authorization

After you agree to grant permissions to TencentCloud Kubernetes Engine, a preset role will be created and relevant permissions will be granted to TencentCloud Kubernetes Engine

Role Name TKE_QCSRole

Role Type Service Role

Description Current role is a TencentCloud Kubernetes Engine service role, which will access your other cloud service resources within the permissions of the associated policies.

Authorized Policies Preset policy QcloudAccessForTKERoleⓘ,Preset policy QcloudAccessForTKERoleInOpsManagementⓘ

Permission content

- CVM

Permission Name	Permission Description
`cvm:DescribeInstances`	Querying the list of server instances
`cvm:*Cbs*`	CBS-related permissions

- Tag

Permission Name	Permission Description
`tag:*`	All features related to tags

- CLB

Permission Name	Permission Description
`clb:*`	All features related to CLB

- TKE

Permission Name	Permission Description
`ccs:DescribeCluster`	Querying a cluster list
`ccs:DescribeClusterInstances`	Querying cluster node information

Preset policy QcloudAccessForTKERoleInOpsManagement

Authorization scenario

This policy is associated with `TKE_QCSRole` by default. After TKE is activated and `TKE_QCSRole` is granted, you have the permissions of various Ops-related features, including log features.

Authorization steps

This policy and the preset policy QcloudAccessForTKERole are authorized at the same time, so no extra operation is needed.

Permission content

Log service

Permission Name	Permission Description
`cls:listTopic`	Displaying the list of log topics under a specified logset
`cls:getTopic`	Viewing log topic information
`cls:createTopic`	Creating a log topic
`cls:modifyTopic`	Modifying a log topic
`cls:deleteTopic`	Deleting a log topic
`cls:listLogset`	Displaying the logset list
`cls:getLogset`	Viewing logset information
`cls:createLogset`	Creating a logset
`cls:modifyLogset`	Modifying a logset
`cls:deleteLogset`	Deleting a logset
`cls:listMachineGroup`	Displaying the server group list

Permission Name	Permission Description
`cls:getMachineGroup`	Viewing server group information
`cls:createMachineGroup`	Creating a server group
`cls:modifyMachineGroup`	Modifying a server group
`cls:deleteMachineGroup`	Deleting a server group
`cls:getMachineStatus`	Viewing server group status
`cls:pushLog`	Uploading logs
`cls:searchLog`	Querying logs
`cls:downloadLog`	Downloading logs
`cls:getCursor`	Getting the cursor based on time
`cls:getIndex`	Viewing indexes
`cls:modifyIndex`	Modifying indexes
`cls:agentHeartBeat`	Heartbeat
`cls:getConfig`	Getting the pusher configuration information

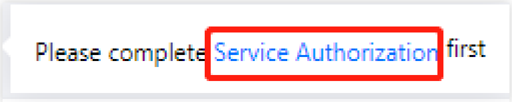
Preset policy QcloudAccessForTKERoleInCreatingCFSStorageclass

Authorization scenario

The Tencent Cloud CFS add-on can help you use file storage in TKE clusters. When using this add-on for the first time, you need to authorize relevant resources, such as file systems in CFS, via TKE.

Authorization steps

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the "Cluster management" page, select the region and ID of the target cluster to go to the cluster details page.
3. Select **Add-on management** and click **Create**.
4. On the **Add-on management** page, if the add-on is selected as "CFS" for the first time, click **Service Authorization** at the bottom of the page.



Please complete Service Authorization first

5. In the "Service authorization" window that pops up, click **Cloud Access Management**.
6. On the "Role management" page, click **Grant** to complete authentication.

Permission content

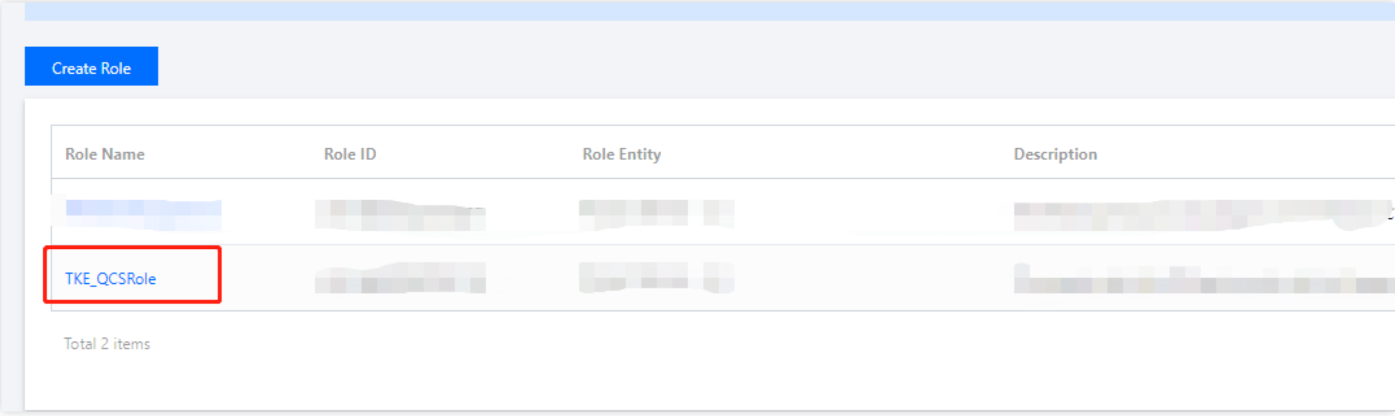
File storage

Permission Name	Permission Description
cfs:CreateCfsFileSystem	Creating a file system
cfs:DescribeCfsFileSystems	Querying a file system
cfs:DescribeMountTargets	Querying mount targets of a file system
cfs:DeleteCfsFileSystem	Deletes a file system

Preset policy QcloudCVMFinanceAccess

Authorization steps

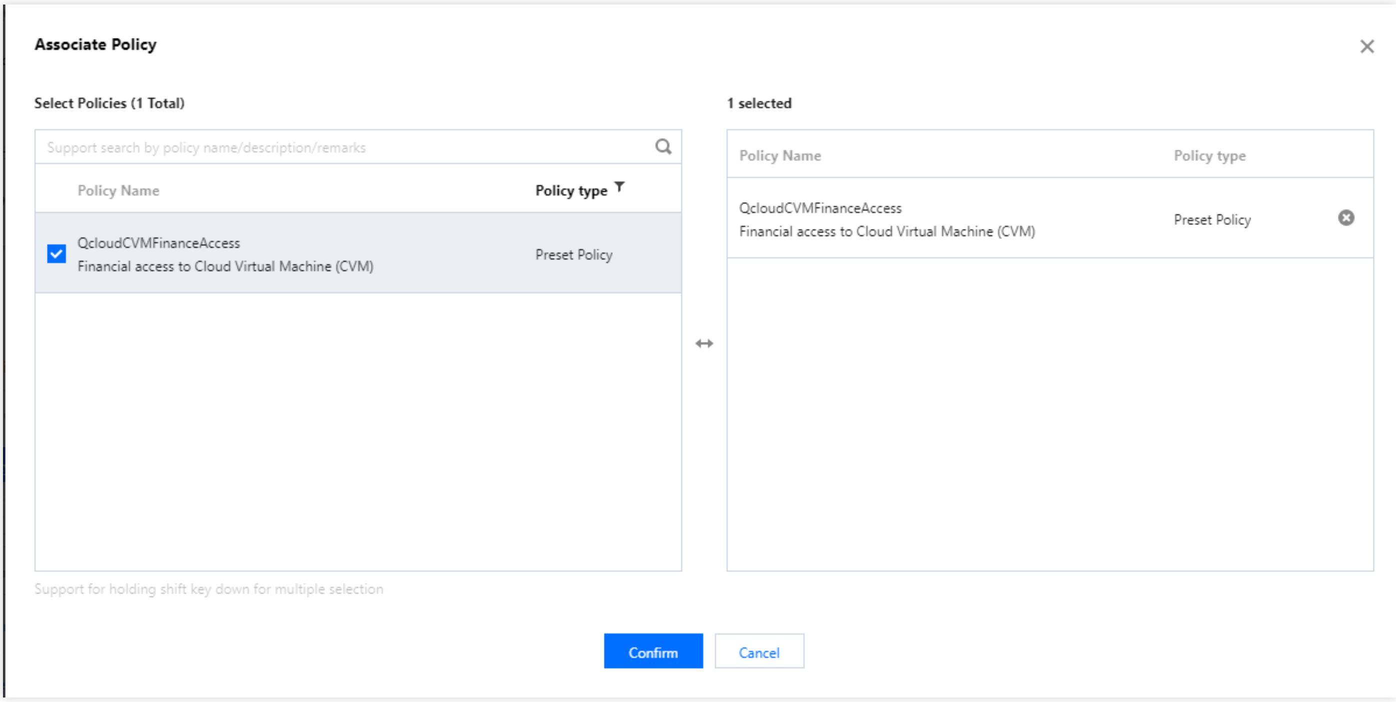
1. Log in to the CAM console, and select **Roles** in the left sidebar.
2. On the role list page, click **TKE_QCSRole** to enter the role management page.



Role Name	Role ID	Role Entity	Description
TKE_QCSRole			

3. Select **Associate policy** on the **TKE_QCSRole** page, and confirm the operation in the "Risk tips" pop-up window.

4. In the "Associate policy" window that pops up, find the policy `QcloudCVMFinanceAccess` and select it.



Associate Policy

Select Policies (1 Total)

	Policy Name	Policy type
☑	QcloudCVMFinanceAccess Financial access to Cloud Virtual Machine (CVM)	Preset Policy

Support for holding shift key down for multiple selection

1 selected

Policy Name	Policy type
QcloudCVMFinanceAccess Financial access to Cloud Virtual Machine (CVM)	Preset Policy

5. Click **Confirm** to complete the process.

Permission content

Permission Name	Permission Description
`finance:*`	CVM finance permission

IPAMDofTKE_QCSRole

`IPAMDofTKE_QCSRole` is the TKE IPAMD support service role. After the permissions of this role are granted, you need to associate preset policies in the authorization scenarios described in this document. After these operations are completed, the following policies will appear in the list of authorized policies of the role:

`QcloudAccessForIPAMDofTKERole` : The permission for TKE IPAMD to access cloud resources

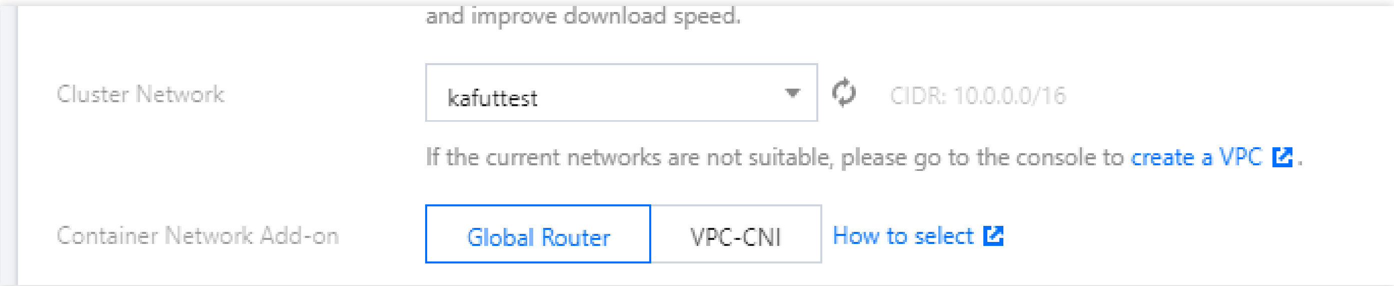
Preset policy QcloudAccessForIPAMDofTKERole

Authorization scenario

When using the VPC-CNI network mode to create a cluster for the first time, you need to grant permission for TKE IPAMD to access cloud resources, so that you can use the VPC-CNI network mode normally.

Authorization steps

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the "Cluster Management" page, click **Create** or **Create with a template** above the cluster list.
3. On the "Create cluster" page, select **VPC-CNI** for **Container network add-on** in "Cluster information" section, and click "Service Authorization".



and improve download speed.

Cluster Network: kafuttest

CIDR: 10.0.0.0/16

If the current networks are not suitable, please go to the console to create a VPC.

Container Network Add-on: Global Router | VPC-CNI

4. In the displayed "Service authorization" window, click **Go to Cloud Access Management**.
5. On the "Role management" page, click **Grant** to complete authentication.

Permission content

- CVM

Permission Name	Permission Description
`cvm:DescribeInstances`	Viewing the list of instances

- Tag

Permission Name	Permission Description
`tag:GetResourcesByTags`	Querying the resource list by tag
`tag:ModifyResourceTags`	Batch modifying tags associated with a resource
`tag:GetResourceTagsByResourceIds`	Querying tags associated with a resource

- VPC

Permission Name	Permission Description
`vpc:DescribeSubnet`	Querying the list of subnets
`vpc:CreateNetworkInterface`	Creating an ENI
`vpc:DescribeNetworkInterfaces`	Querying the list of ENIs
`vpc:AttachNetworkInterfac` e	Binding an ENI with a CVM

Permission Name	Permission Description
`vpc:DetachNetworkInterface`	Unbinding an ENI from a CVM
`vpc:DeleteNetworkInterface`	Deleting an ENI
`vpc:AssignPrivateIpAddresses`	Applying for private IP addresses for an ENI
`vpc:UnassignPrivateIpAddresses`	Returning the private IP addresses of an ENI
`vpc:MigratePrivateIpAddress`	Migrating the private IP addresses of an ENI
`vpc:DescribeSubnetEx`	Querying the list of subnets
`vpc:DescribeVpcEx`	Querying peering connection
`vpc:DescribeNetworkInterfaceLimit`	Querying the ENI quota
`vpc:DescribeVpcPrivateIpAddresses`	Querying the private IP address of a VPC

Controlling TKE cluster-level permissions
Using TKE Preset Policy Authorization

Last updated：2020-09-04 10:53:28

This document describes the preset policies offered by Tencent Kubernetes Engine (TKE). It shows you how to associate sub-accounts with preset policies to grant specific permissions. You can use this document as a reference to configure preset policies that are in line with your particular business needs.

TKE Preset Policies

You can grant relevant permissions to sub-accounts by using the following preset policies:

Policy	Description
`QcloudTKEFullAccess`	This policy grants full read/write access permissions for TKE, including permissions for TKE and related CVMs, CLBs, VPCs, monitors, and user groups.
`QcloudTKEInnerFullAccess`	This policy grants full access permission for TKE. However, since TKE involves the use of many products, we recommend configuring `QcloudTKEFullAccess` .
`QcloudTKEReadOnlyAccess`	This policy grants read-only permission for TKE.

The following preset policies are used to grant permissions for specific TKE services. We do not recommend associating the following preset policies with a sub-account:

Policy	Description
`QcloudAccessForCODINGRoleInAccessTKE`	This policy grants the relevant TKE permissions for the Coding service.
`QcloudAccessForIPAMDofTKERole`	This policy grants the relevant ENI permissions for the TKE service.
`QcloudAccessForIPAMDRoleInQcloudAllocateEIP`	This policy grants the relevant EIP permissions for the TKE service.
`QcloudAccessForTKERole`	This policy grants the relevant CVM, Tag, CLB, and CLS permissions for the TKE service.

`QcloudAccessForTKERoleInCreatingCFSStorageclass`	This policy grants the relevant CFS permissions for the TKE service.
`QcloudAccessForTKERoleInOpsManagement`	This policy associates the TKE service role (TKE_QCSRole) so that TKE can access other Tencent Cloud services, including CLS.

Associating Sub-accounts with Preset Policies

When setting user permissions during the creation of a sub-account, you can associate preset policies with the sub-account by direct association or association via group.

Direct association

By directly associating your sub-account with a policy, the sub-account obtains the permissions contained in the policy. The direct association process is outlined below:

1. Log in to the CAM console and select **Users** -> **User List** on the left sidebar.
2. On the **User List** page, find the target sub-account and click **Grant Permission** in the **Operation** column.
3. On the **Associate Policies** page, select the policies that you want to associate.
4. Click **OK**.

Association via group

By adding your sub-account to a user group, the sub-account automatically obtains the permissions that are associated with the user group. To disassociate the sub-account from the policies of the group, you simply need to remove the sub-account from the user group. The group association process is outlined below:

1. Log in to the CAM console and select **Users** -> **User List** on the left sidebar.
2. On the **User List** page, find the target sub-account and choose **More** -> **Add to Group** in the **Operation** column.
3. On the **Add to Group** page, select the target user group.
4. Click **OK**.

Logging in to the sub-account for verification

Log in to the TKE console to verify that the features corresponding to the associated policies can be used and/or accessed properly. If so, this indicates that the sub-account was successfully authorized.

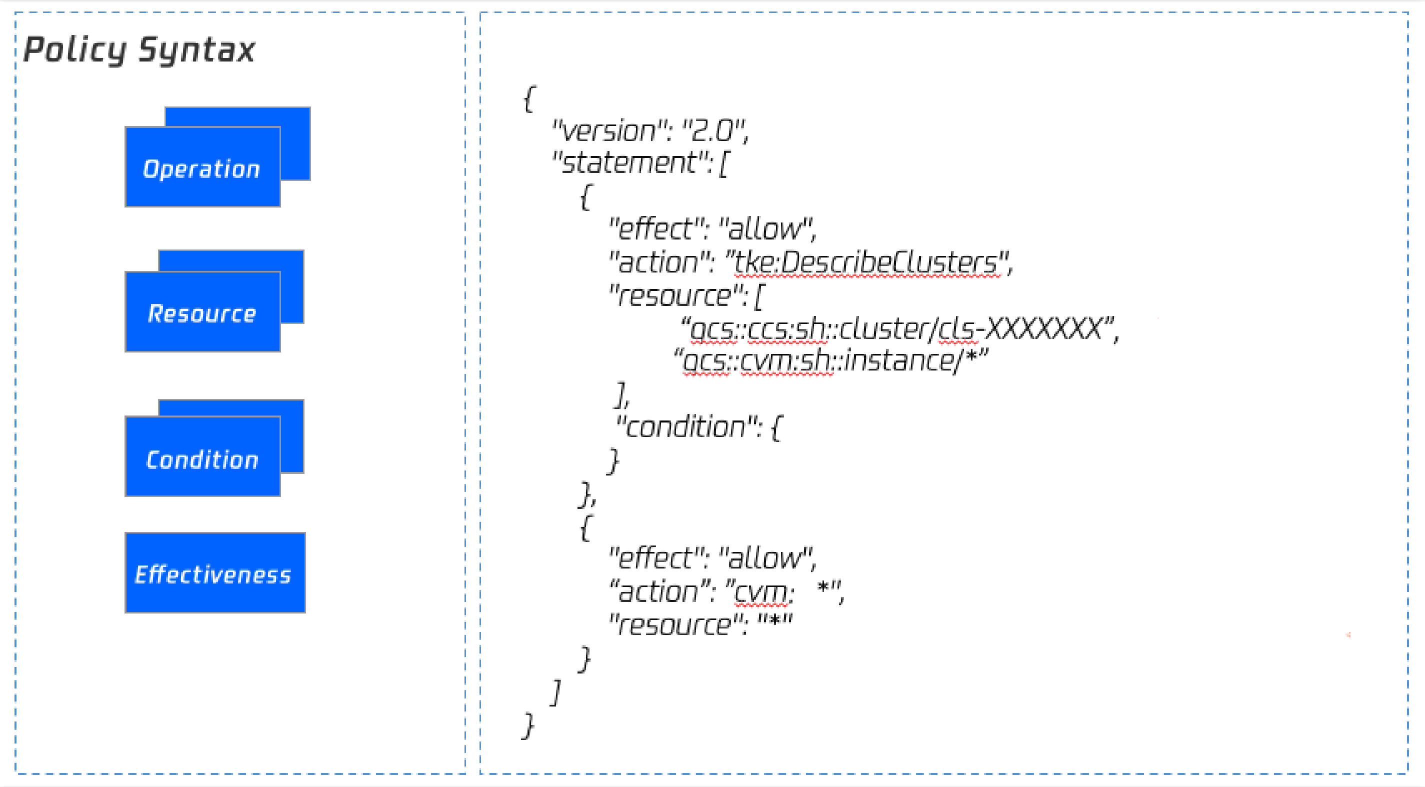
Authorizing by using custom policies

Last updated：2020-10-10 15:04:42

This document describes how to configure custom policies in Tencent Kubernetes Engine (TKE) and grant sub-accounts specific permissions. Reference this document to create custom policies that best fit your business requirements.

Policy Syntax Description

The following figure shows the structure of the policy syntax.



```
{
  "version": "2.0",
  "statement": [
    {
      "effect": "allow",
      "action": "tke:DescribeClusters",
      "resource": [
            "qcs::ccs:sh::cluster/cls-XXXXXXX",
           "qcs::cvm:sh::instance/*"
      ],
       "condition": {
      }
    },
    {
      "effect": "allow",
      "action": "cvm:  *",
      "resource": "*"
    }
  ]
}
```

- **action**: indicates an API.
- **resource**: indicates a resource.

> **Note：**
>
> You can define the policy syntax on your own, or create a custom policy by using the policy generator in CAM. You can configure a custom policy based on the following example.
>
> - Configuring a Sub-account's Administrative Permissions for a Single TKE Cluster

- Use tags to grant full permissions for a batch of clusters to a sub-account

Configuring TKE API Permissions

This section describes multiple features, their sub-features, corresponding Tencent Cloud APIs, APIs for indirect calls, resource levels for permission control, and Action fields of clusters and node modules.

Cluster modules

The following table describes the mappings between features and APIs.

Feature	Sub-feature	Corresponding Tencent Cloud API	API for Indirect Calls
Creating an empty cluster	• Selecting a Kubernetes version • Selecting a runtime component • Selecting a VPC • Setting a container network • Selecting a custom image • Setting IPVS	tke:CreateCluster	cam:GetRole account:DescribeUserData account:DescribeWhiteList tag:GetTagKeys cvm:GetVmConfigQuota vpc:DescribeVpcEx cvm:DescribeImages
Using an existing CVM to create a managed cluster	• Creating an empty cluster to include features • Using an existing CVM as a node • Mounting a security		cvm:DescribeInstances vpc:DescribeSubnetEx cvm:DescribeSecurityGroups vpc:DescribeVpcEx cvm:DescribeImages cvm:ResetInstance cvm:DescribeKeyPairs

	group • Mounting a data disk • Enabling automatic adjustment		
Using an existing CVM to create a self-deployed cluster	• Creating an empty cluster to include features • Using an existing CVM as a node • Using an existing CVM as Control Plane & ETCD • Mounting a security group • Mounting a data disk • Enabling automatic adjustment		cvm:DescribeInstances vpc:DescribeSubnetEx cvm:DescribeSecurityGroups vpc:DescribeVpcEx cvm:DescribeImages cvm:ResetInstance cvm:DescribeKeyPairs
Automatically creating a CVM to create a managed cluster	• Creating an empty cluster to include features • Purchasing a CVM as a node • Mounting a security group • Mounting a data disk • Enabling automatic		cvm:DescribeSecurityGroups cvm:DescribeKeyPairs cvm:RunInstances vpc:DescribeSubnetEx vpc:DescribeVpcEx cvm:DescribeImages

	adjustment		
Automatically creating a CVM to create a self-deployed cluster	• Creating an empty cluster to include features • Purchasing a CVM as a node • Purchasing a CVM as Control Plane & ETCD • Mounting a security group • Mounting a data disk • Enabling automatic adjustment		cvm:DescribeSecurityGroups cvm:DescribeKeyPairs cvm:RunInstances vpc:DescribeSubnetEx vpc:DescribeVpcEx cvm:DescribeImages
Querying a cluster list	-	tke:DescribeClusters	-
Displaying cluster credentials	-	tke:DescribeClusterSecurity	-
Enabling/Disabling the private network/Internet access URL of a cluster	• Creating an Internet access port for a managed cluster • Creating a cluster access port	tke:CreateClusterEndpointVip tke:CreateClusterEndpoint tke:ModifyClusterEndpointSP tke:DescribeClusterEndpointVipStatus tke:DescribeClusterEndpointStatus tke:DeleteClusterEndpointVip tke:DeleteClusterEndpoint	-

	• Modifying security policies for the Internet access port of a managed cluster • Querying the Internet access port enabling status of a managed cluster • Deleting the Internet access port of a managed cluster • Deleting a cluster access port		
Deleting a cluster	-	tke:DeleteCluster	tke:DescribeClusterInstances tke:DescribeInstancesVersion tke:DescribeClusterStatus

Node modules

The following table describes the mappings between features and APIs.

Feature	Sub-feature	Corresponding Tencent API	API for Indirect Calls	Resource Level for Permission Control
Adding an existing node	• Adding an existing node to a cluster	tke:AddExistedInstances	cvm:DescribeInstances vpc:DescribeSubnetEx cvm:DescribeSecurityGroups vpc:DescribeVpcEx cvm:DescribeImages cvm:ResetInstance	• Cluster-level permissions are required for adding

	- Resetting a data disk - Setting a security group		cvm:DescribeKeyPairs cvm:ModifyInstancesAttribute tke:DescribeClusters	an existing node. - CVM-level permissions are required for obtaining a CVM list.
Creating a node	- Creating a node and adding it to a cluster - Resetting a data disk - Setting a security group	tke:CreateClusterInstances	cvm:DescribeSecurityGroups cvm:DescribeKeyPairs cvm:RunInstances vpc:DescribeSubnetEx vpc:DescribeVpcEx cvm:DescribeImages tke:DescribeClusters	Cluster-level permissions are required for creating a node.
Node list	Viewing a cluster node list	tke:DescribeClusterInstances	cvm:DescribeInstances tke:DescribeClusters	- Cluster-level permissions are required for viewing a node list. - CVM-level permissions are required for obtaining a CVM list.
Deleting a node	-	tke:DeleteClusterInstances	cvm:TerminateInstances tke:DescribeClusters	- Cluster-level permissions are required for viewing a node list. - CVM-level permissions are required for

				obtaining a CVM list. - The termination policy of a node is required for terminating the node.

Usage Examples
Using Labels to Configure Sub-accounts with Full Read/Write Permissions for Batch Clusters

Last updated：2023-02-02 17:05:22

Overview

You can grant a user permissions to view and use specific resources in the TKE console by using a Cloud Access Management (CAM) policy. This document describes how to grant a sub-account the permissions for a cluster with the specified tag in the console.

Directions

1. Log in to the CAM console. Click **Create custom policy** in the upper-left corner.
2. On the **Select Policy Creation Method** page that pops up, select **Authorize by Tag**.
3. In the service and action addition area of the Visual Policy Generator, enter the following information, and edit an authorization statement.
   - **Service** (required): select TKE.
   - **Action** (required): select the actions you want to authorize.
4. In the tag selection area, select the tags to authorize. Authorized sub-accounts will have the full read/write permissions for the resources with the specified tag key and tag value.
5. Click **Next**. On the **Bind User**/**Group**/**Role** page displayed, enter the policy name and description. The policy name is `policygen` by default, which is generated automatically in the console. The suffix number is generated based on the creation date. This is customizable.
6. Authorize users/groups/roles. Authorized sub-accounts will have the full read/write permissions for the resources with the specified tag key and tag value.

- **Authorized User**: select the target sub-accounts as required.
- **Authorized User Group**: select the user group to which the target sub-accounts belongs.
- **Authorized Role**: select the role to which the target sub-accounts belong.

7. Click **Complete**.

Configuring a Sub-account's Administrative Permissions to a Single TKE Cluster

Last updated：2021-06-08 11:21:00

Overview

You can grant a user the permissions to view and use specific resources in the TKE console by using a CAM policy. This document describes how to configure the CAM policy of a single cluster in the console.

Directions

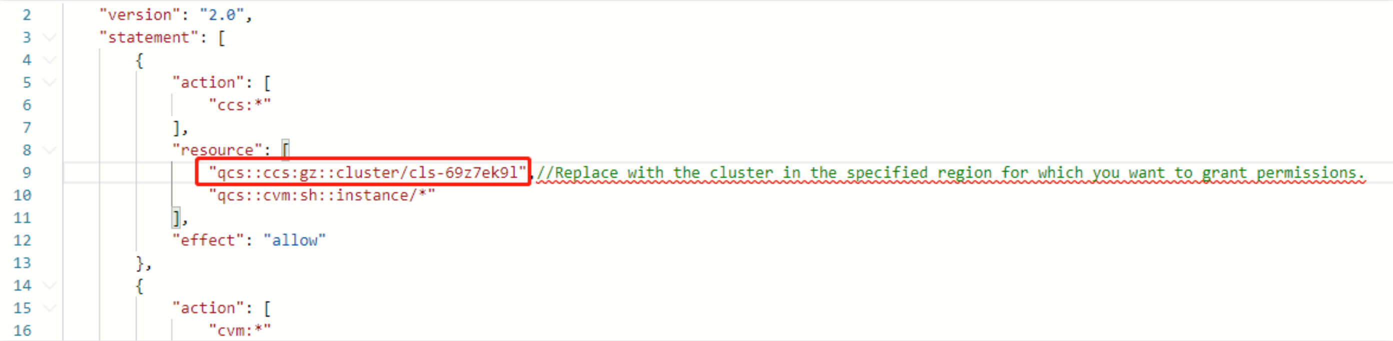
Configuring full read/write permission for a single cluster

1. Log in to the CAM console.
2. In the left sidebar, click Policies to go to the policy management page.
3. Click **Create Custom Policy** and select the "Create by Policy Syntax" method.
4. Select the "Blank template" type and click **Next**.
5. Enter a custom policy name and replace "Edit policy content" with the following content.

```
json
{
"version": "2.0",
"statement": [
{
"action": [
"ccs:*"
],
"resource": [
"qcs::ccs:sh::cluster/cls-XXXXXXX",
"qcs::cvm:sh::instance/*"
],
"effect": "allow"
},
{
"action": [
"cvm:*"
],
"resource": "*",
"effect": "allow"
},
{
"action": [
"vpc:*"
],
"resource": "*",
"effect": "allow"
},
{
"action": [
"clb:*"
],
"resource": "*",
"effect": "allow"
},
{
"action": [
"monitor:*",
"cam:ListUsersForGroup",
"cam:ListGroups",
"cam:GetGroup",
"cam:GetRole"
],
"resource": "*",
"effect": "allow"
}
]
}
```

6. In "Edit policy content", modify `qcs::ccs:sh::cluster/cls-XXXXXXX` to the cluster in the specified region for which you want to grant permissions, as shown below:

   For example, if you need to grant full read/write permission for the cls-69z7ek9l cluster in Guangzhou, modify `qcs::ccs:sh::cluster/cls-XXXXXXX` to `"qcs::ccs:gz::cluster/cls-69z7ek9l"` .



```
    "version": "2.0",
    "statement": [
        {
            "action": [
                "ccs:*"
            ],
            "resource": [
                "qcs::ccs:gz::cluster/cls-69z7ek9l",//Replace with the cluster in the specified region for which you want to grant permissions.
                "qcs::cvm:sh::instance/*"
            ],
            "effect": "allow"
        },
        {
            "action": [
                "cvm:*"
```

Note：

Replace with the ID of the cluster in the specified region for which you want to grant permissions. If you want to allow sub-accounts to scale the cluster, you also need to configure the user payment permission for the sub-accounts.

7. Click **Create a policy** to complete the configuration of full read/write permission for a single cluster.

Configuring read-only permission for a single cluster

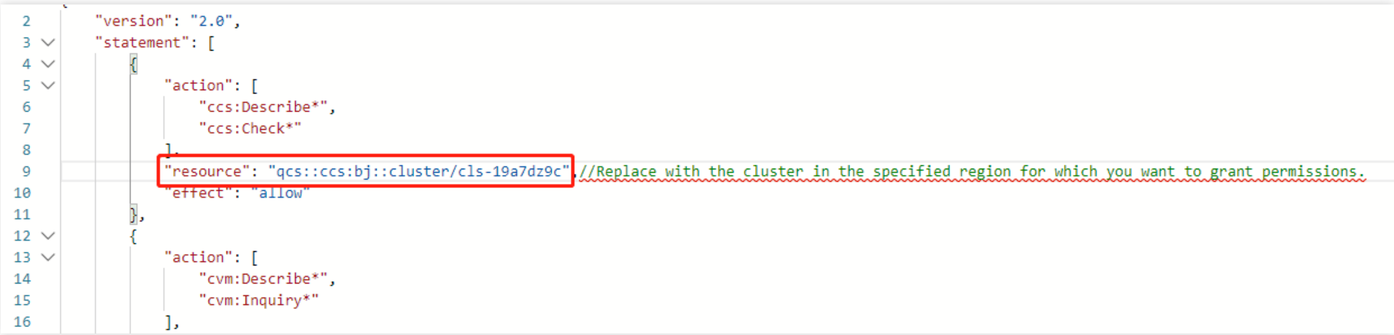
1. Log in to the CAM console.
2. In the left sidebar, click Policies to go to the policy management page.
3. Click **Create Custom Policy** and select the "Create by Policy Syntax" method.
4. Select the "Blank template" type and click **Next**.
5. Enter a custom policy name and replace "Edit policy content" with the following content.

```
json
{
"version": "2.0",
"statement": [
{
"action": [
"ccs:Describe*",
"ccs:Check*"
],
"resource": "qcs::ccs:gz::cluster/cls-1xxxxxx",
"effect": "allow"
},
{
"action": [
"cvm:Describe*",
"cvm:Inquiry*"
],
"resource": "*",
"effect": "allow"
},
{
"action": [
"vpc:Describe*",
"vpc:Inquiry*",
"vpc:Get*"
],
"resource": "*",
"effect": "allow"
},
{
```

```
"action": [
"clb:Describe*"
],
"resource": "*",
"effect": "allow"
},
{
"effect": "allow",
"action": [
"monitor:*",
"cam:ListUsersForGroup",
"cam:ListGroups",
"cam:GetGroup",
"cam:GetRole"
],
"resource": "*"
}
]
}
```

6. In "Edit policy content", modify `qcs::ccs:gz::cluster/cls-1xxxxxx` to the cluster in the specified region for which you want to grant permissions, as shown below:

   For example, if you need to grant ready-only permission for the cls-19a7dz9c cluster in Beijing, modify `qcs::ccs:gz::cluster/cls-1xxxxxx` to `qcs::ccs:bj::cluster/cls-19a7dz9c` .



```
    "version": "2.0",
    "statement": [
        {
            "action": [
                "ccs:Describe*",
                "ccs:Check*"
            ],
            "resource": "qcs::ccs:bj::cluster/cls-19a7dz9c",//Replace with the cluster in the specified region for which you want to grant permissions.
            "effect": "allow"
        },
        {
            "action": [
                "cvm:Describe*",
                "cvm:Inquiry*"
            ],
```

7. Click **Create a policy** to complete the configuration of read-only permission for a single cluster.

Configuring a Sub-account's Full Read/write or Read-only Permission to TKE

Last updated：2023-02-02 17:05:22

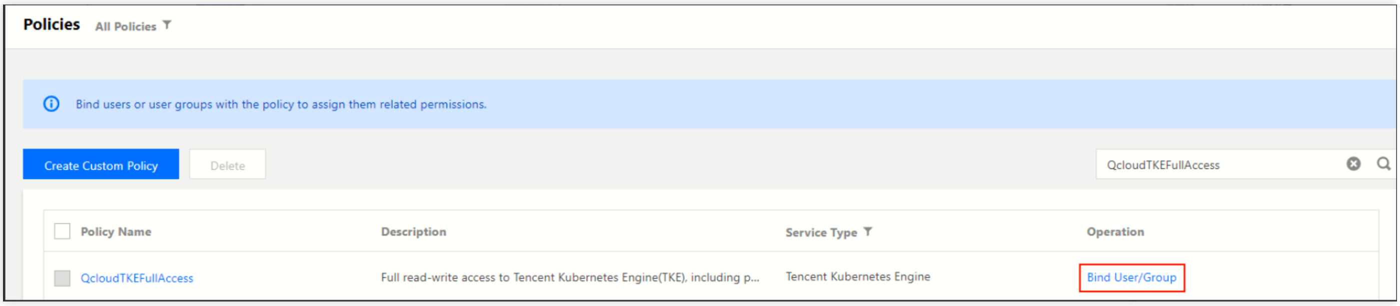
Overview

You can grant a user the permissions to view and use specific resources in the TKE console by using a CAM policy. This document describes how to configure certain permission policies in the console.

Directions

Configuring Full Read/write Permission

1. Log in to the CAM console and select Policies in the left sidebar.
2. On the **Policies** page, click **Bind User**/**Group**/**Role** in the **Operation** column of the **QcloudTKEFullAccess** policy.



Bind users or user groups with the policy to assign them related permissions.

Policy Name	Description	Service Type	Operation
QcloudTKEFullAccess	Full read-write access to Tencent Kubernetes Engine(TKE), including p...	Tencent Kubernetes Engine	Bind User/Group

3. In the **Bind User**/**Group**/**Role** window that pops up, select the accounts that need full read/write permission for the TKE service, and click **OK** to grant full read/write permission for the TKE service to the sub-accounts.
4. On the **Policies** page, click **Bind User**/**Group**/**Role** in the **Operation** column of the **QcloudCCRFullAccess** policy.
5. In the **Bind User**/**Group**/**Role** window that pops up, select the accounts that need full read/write permission for Image Registry, and click **OK** to grant full read/write permission for Image Registry to the sub-accounts.

> Note：
> If you want to use the trigger and automatic building features of Image Registry, you also need to configure additional permissions for TKE - continuous integration (CCB).

Configuring Read-only Permission

1. Log in to the CAM console and select **Policies** in the left sidebar.
2. On the **Policies** page, click **Bind User**/**Group**/**Role** in the **Operation** column of the **QcloudTKEReadOnlyAccess** policy.
3. In the **Bind User**/**Group**/**Role** window that pops up, select the accounts that need the read-only permission for the TKE service, and click **OK** to grant the read-only permission for the TKE service to the sub-accounts.
4. On the **Policies** page, click **Bind User**/**Group**/**Role** in the **Operation** column of the **QcloudCCRReadOnlyAccess* policy.
5. In the **Bind User**/**Group**/**Role** window that pops up, select the accounts that need the read-only permission for Image Registry, and click **OK** to grant the read-only permission for Image Registry to the sub-accounts.

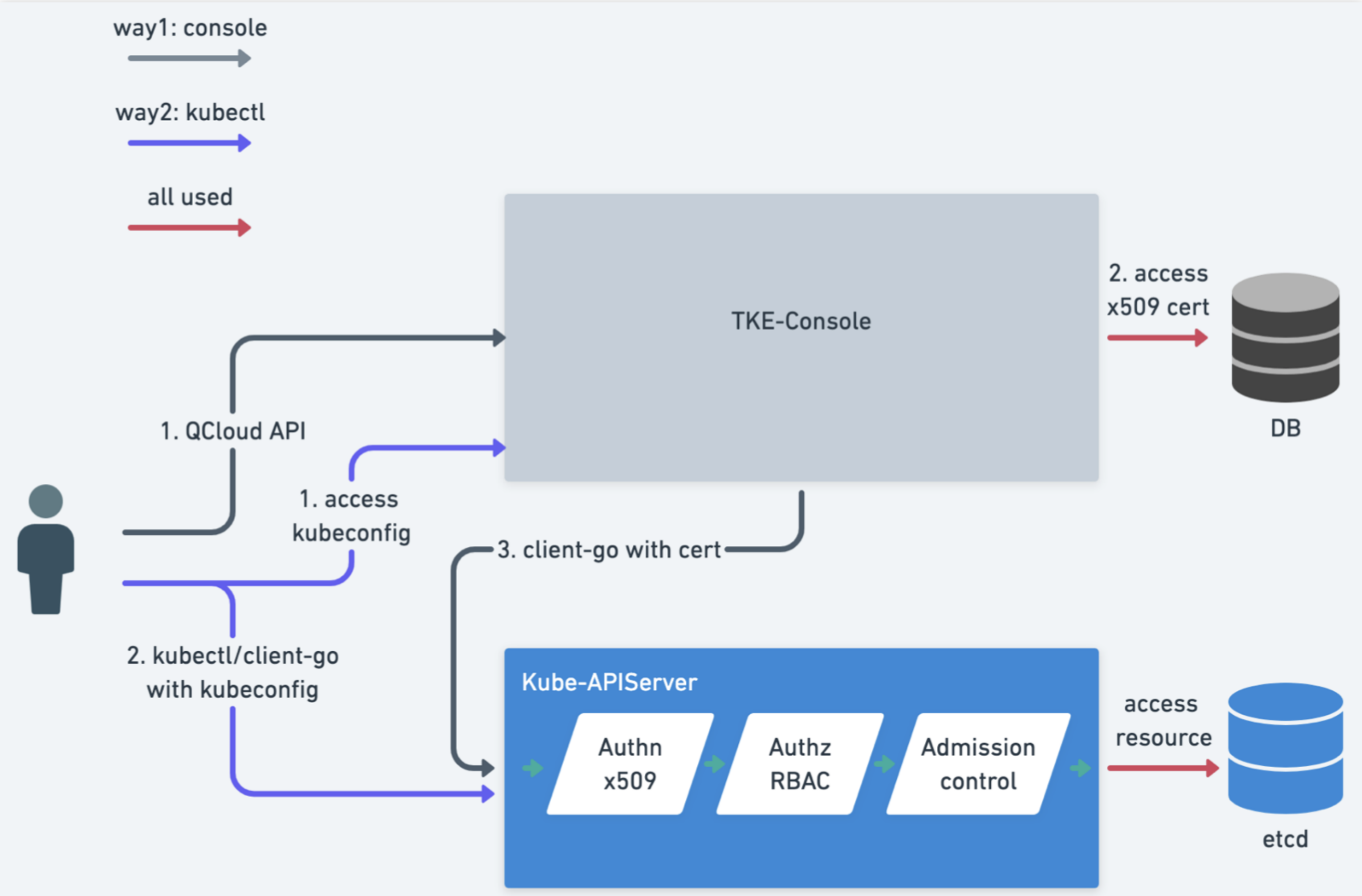
> Note：
> If you want to use the trigger and automatic building features of Image Registry, you also need to configure additional permissions for TKE - continuous integration (CCB).

TKE Kubernetes Object-level Permission Control

Overview

Last updated：2020-12-24 10:56:17

TKE supports the Kubernetes RBAC authorization method, allowing you to perform fine-grained access control for sub-accounts. With this authorization method, you can access resources in a cluster through the TKE console and kubectl. For more information, see the following figure.



Glossary

RBAC

Role-Based Access Control (RBAC) associates users and permissions with roles to indirectly grant permissions to users.

In Kubernetes, RBAC is implemented through the `rbac.authorization.k8s.io` API group, which allows cluster administrators to dynamically configure policies through Kubernetes APIs.

Role

A Role is used to define access permissions for resources in a single namespace.

ClusterRole

A ClusterRole is used to define access permissions for resources in an entire cluster.

RoleBinding

RoleBinding grants the permissions defined in a role to a user or group of users in order to grant authorization for a namespace.

ClusterRoleBinding

ClusterRoleBinding grants the permissions defined in a role to a user or group of users in order to grant cluster-wide authorization.

For more information, see the Kubernetes official documentation.

Solutions for TKE Kubernetes Object-Level Permission Control

Verification method

Kubernetes API servers support various verification policies, such as x509 certificates, bearer tokens, and basic auth. Of these, only individual bearer token verification policies support verification based on the tokens of specified known-token csv files, such as bearer tokens, serviceaccount tokens, OIDC tokens, and webhook token servers.

With implementation complexity and different usage scenarios in mind, TKE has chosen to use x509 certificates as the verification method. This verification method offers the following advantages:

- This method is easier for users to understand.
- No complex changes need to be made to existing clusters.
- You can sort by users and groups, which facilitates subsequent scaling.

TKE implements the following features based on x509 certificate verification:

- Each sub-account has a unique client certificate used for accessing Kubernetes API servers.
- When a sub-account accesses Kubernetes resources on the console, the backend uses the sub-account’s client certificate to access the Kubernetes API server by default.
- A sub-account can update its unique client certificate to prevent credential disclosure.

- A root account or an account that has `tke:admin` permission for a cluster can view and update the certificates of other sub-accounts.

Authorization method

Kubernetes supports two main authorization methods: RBAC and Webhook Server. In order to provide a consistent experience for users who are familiar with and work with native Kubernetes, TKE has chosen RBAC as its authorization method. This authorization method provides preset Roles and ClusterRoles. You only need to create the corresponding RoleBinding or ClusterRoleBinding to implement authorizations for a cluster or namespace. This authorization method has the following advantages:

- It is friendly to users who have a basic knowledge of Kubernetes.
- It allows you to reuse Kubernetes RBAC capabilities and supports various verb-based permission controls for namespaces, API groups, and resources.
- It supports custom policies.
- It allows you to manage custom extended API resources.

Features of TKE Kubernetes Object-level Permission Control

By using the authorization management feature provided by TKE, you can perform more fine-grained permission control. For example, you can configuring sets of permissions such as assigning read-only permissions to a sub-account or assigning read/write permissions to only a certain namespace under a sub-account. For more information on configuring more fine-grained sets of permissions for sub-accounts, see the following documents:

- Using a preset identity for authorization
- Using custom policies for authorization

Comparison of Authorization Modes

Last updated：2020-09-18 10:44:42

Tencent Kubernetes Engine (TKE) currently supports both new and old authorization methods. However, old authorization methods cannot be used to perform Kubernetes-level authorization. We recommend you upgrade the authorization method for your cluster so that you can perform fine-grained permission control for the Kubernetes resources in the cluster.

Comparison of New and Old Authorization Methods

Item	Old Authorization Method	New Authorization Method
Kubeconfig	admin token	x509 certificate unique to each sub-account
Access to cluster resources on the console	No fine-grained permissions, and sub-accounts are granted full read/write permission	Incorporates Kubernetes RBAC resource control

Upgrading the Authorization Method for Existing Clusters

Upgrading the authorization method

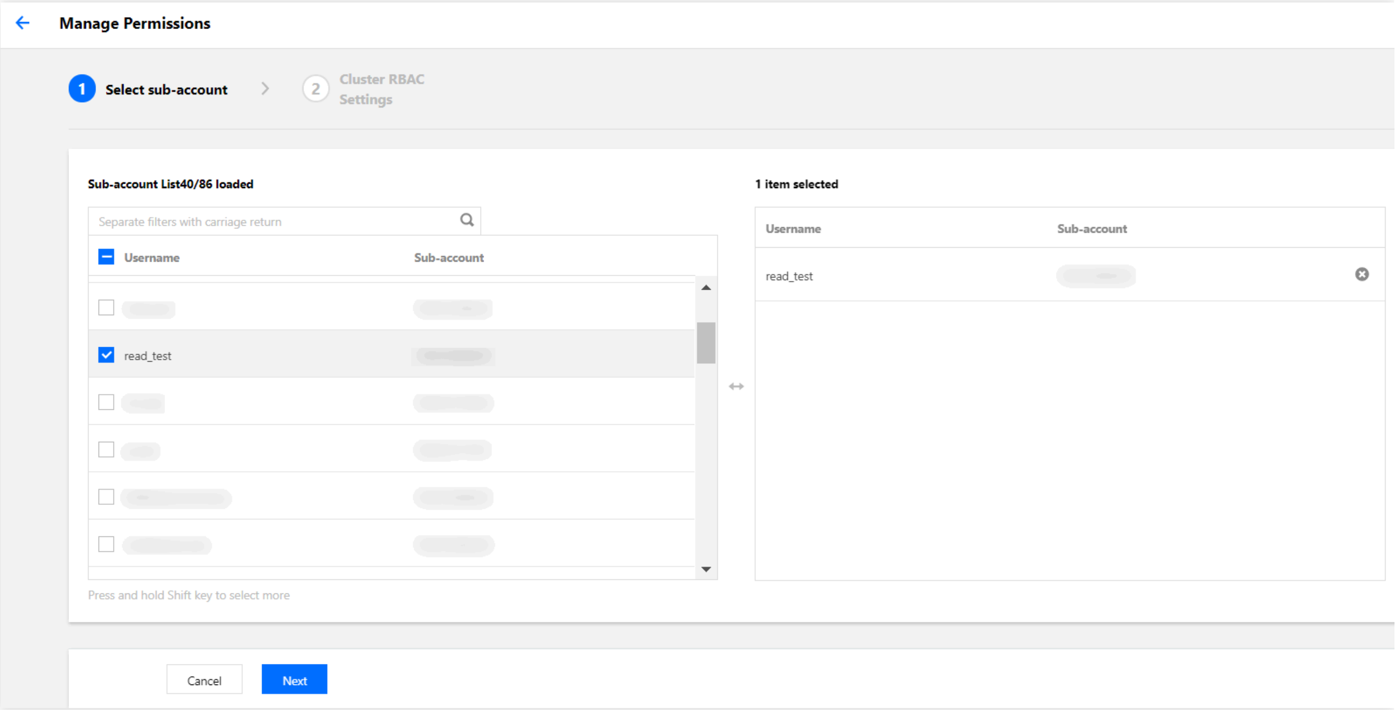
To upgrade a cluster that uses an old authorization method, perform the following steps:

1. Log in to the TKE console and click **Cluster** on the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster.
3. On the cluster details page, select **Authorization Management** -> **ClusterRole** on the left sidebar.
4. On the **ClusterRole** page, click **RBAC Policy Generator**.
5. On the **Switch Permission Management Mode** page, click **Switch Permission Management Mode** to upgrade the authorization method.
   To ensure that the new authorization method is compatible with the old one, TKE will perform the following operations during the upgrade:
   vi. Creating the default preset administrator ClusterRole: `tke:admin` .
   vii. Obtaining the sub-account list.
   viii. Generating the x509 client certificates for Kubernetes API server authentication for each sub-account.
   ix. Binding the `tke:admin` role to each sub-account to ensure compatibility with existing features.
   x. Completing the upgrade.

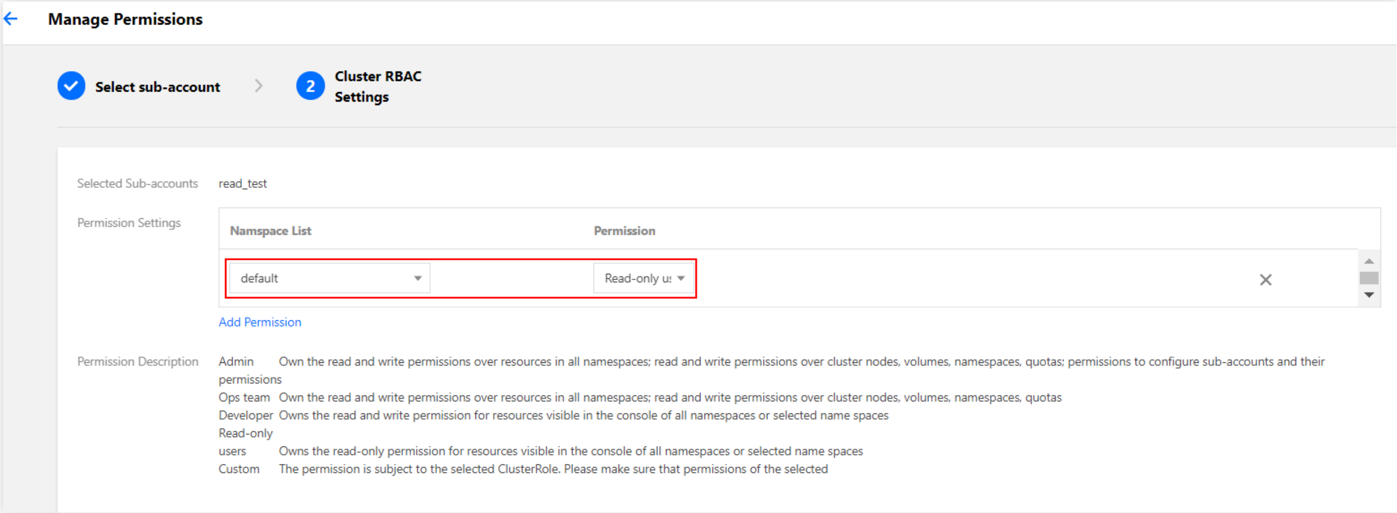
Repossessing sub-account permissions

After the authorization method of a cluster is upgraded, the cluster administrator (often the root account administrator or OPS person who created the cluster) can repossess the cluster permissions granted to sub-accounts as required. The steps are as follows:

1. Select an item under the cluster's **Authorization Management** page, and click **RBAC Policy Generator** on the corresponding management page.
2. When you select a sub-account on the **Administration Permissions** page, select the sub-account whose permissions you want to repossess and then click **Next**, as shown in the following figure.



3. When you set the cluster RBAC, you can also set permissions. For example, select **Read-only Users** as the **Permission Setting** for the **default** namespace, as shown in the following figure.



Manage Permissions

Select sub-account > 2 Cluster RBAC Settings

Selected Sub-accounts read_test

Permission Settings

Namspace List	Permission
default	Read-only us

Add Permission

Permission Description

Admin Own the read and write permissions over resources in all namespaces; read and write permissions over cluster nodes, volumes, namespaces, quotas; permissions to configure sub-accounts and their permissions
Ops team Own the read and write permissions over resources in all namespaces; read and write permissions over cluster nodes, volumes, namespaces, quotas
Developer Owns the read and write permission for resources visible in the console of all namespaces or selected name spaces
Read-only users Owns the read-only permission for resources visible in the console of all namespaces or selected name spaces
Custom The permission is subject to the selected ClusterRole. Please make sure that permissions of the selected

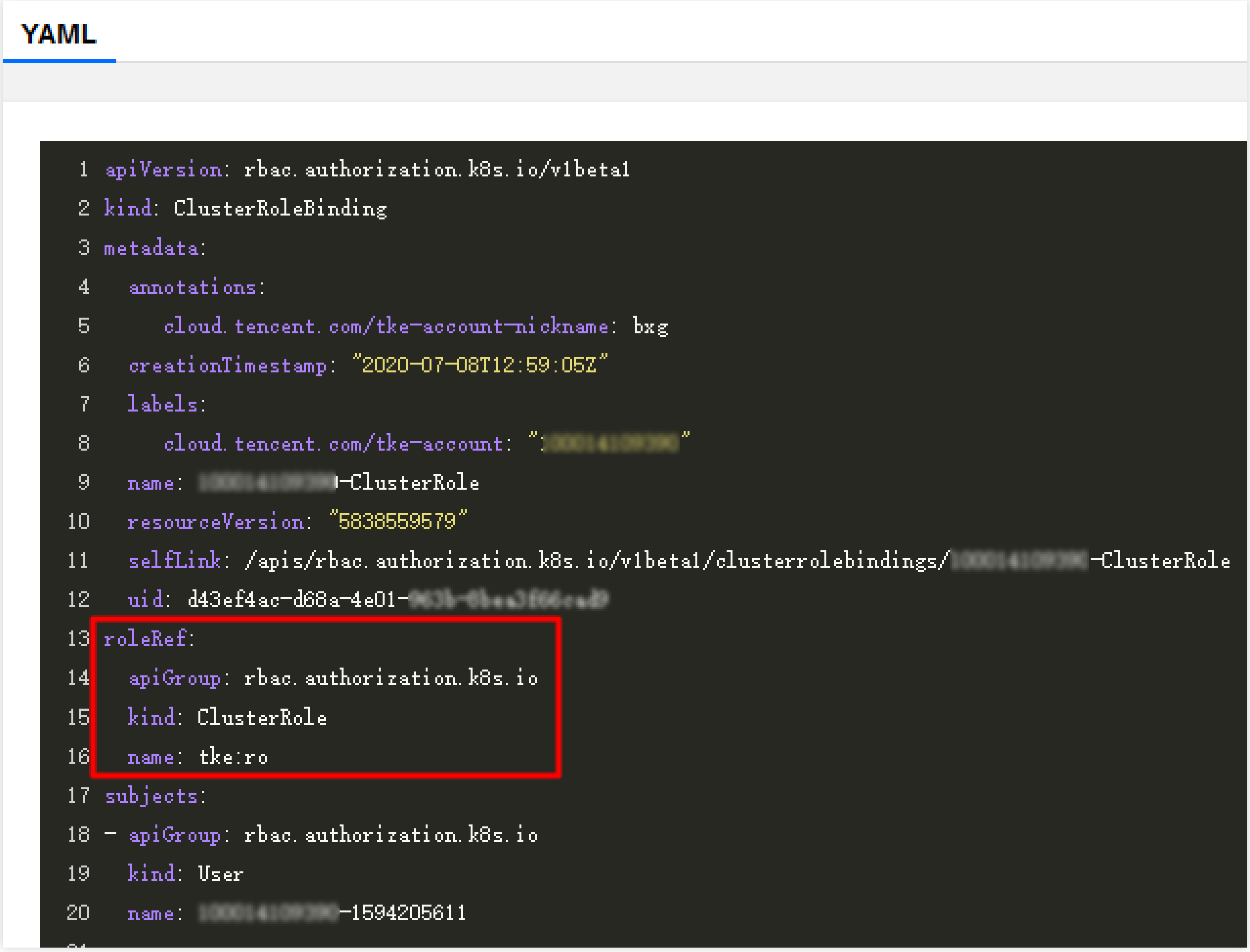
4. Click **Done** to complete the repossession.

Verifying permissions of sub-accounts

After you repossess the permissions of a sub-account, you can verify the current permissions as follows:

1. On the cluster details page, select **Authorization Management** -> **ClusterRoleBinding** on the left sidebar to enter the **ClusterRoleBinding** page.
2. Select the sub-account whose permissions have been repossessed to go to the YAML file page.
   The sub-account has `tke:admin` permission by default. After permissions are repossessed, you can view the

change in the YAML file, as shown in the following figure.



```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  annotations:
     cloud.tencent.com/tke-account-nickname: bxg
  creationTimestamp: "2020-07-08T12:59:05Z"
  labels:
     cloud.tencent.com/tke-account: "[redacted]"
  name: [redacted]-ClusterRole
  resourceVersion: "5838559579"
  selfLink: /apis/rbac.authorization.k8s.io/v1beta1/clusterrolebindings/[redacted]-ClusterRole
  uid: d43ef4ac-d68a-4e01-[redacted]
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: tke:ro
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: [redacted]-1594205611
```

New Authorization Method FAQ

For a cluster that is created using the new authorization method, who has admin permission?

The cluster creator and the root account always have `tke:admin` ClusterRole permission.

Can I control the permissions of the current account?

TKE currently does not allow you to perform permission operations on the current account. You can perform these operations by using kubectl.

Can I directly perform operations on ClusterRoleBindings and ClusterRoles?

Please do not directly modify or delete ClusterRoleBindings and ClusterRoles.

How can I create a client certificate?

When you access cluster resources on the console through a sub-account, TKE will obtain the client certificate of the sub-account. If no certificate is obtained, TKE will create a client certificate for the sub-account.

After a sub-account is deleted on the CAM console, will TKE automatically repossess the relevant permissions?

TKE will automatically repossess the permissions, so you do not need to perform any additional operations.

How can I grant “authorization management” permission to another account?

You can use the default admin role `tke:admin` to grant “authorization management” permission.

Using Preset Identity Authorization

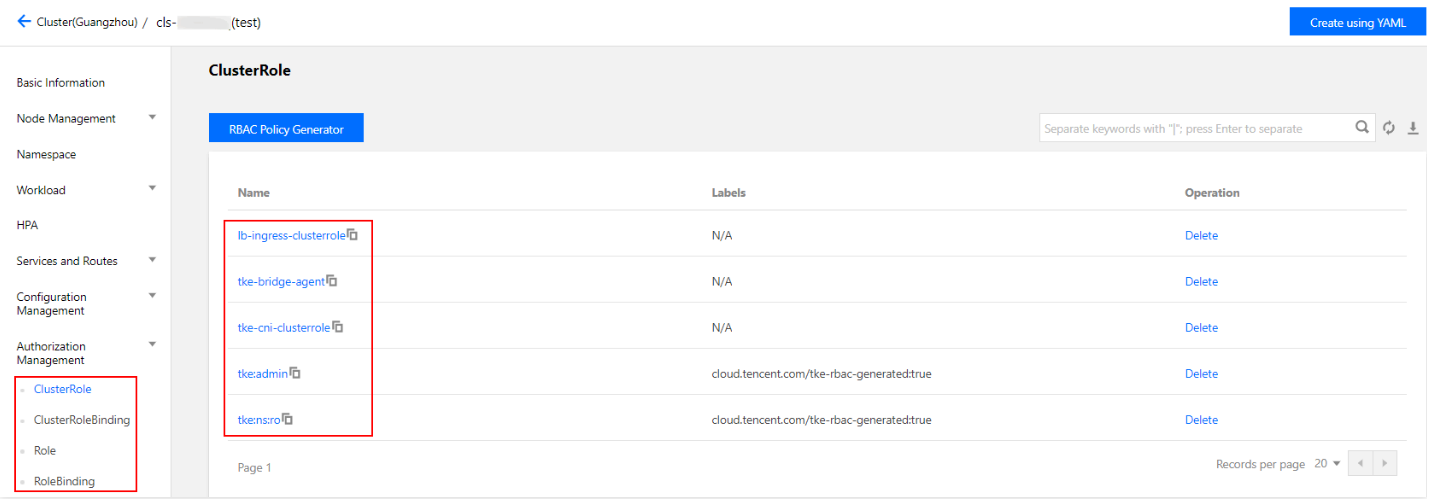
Last updated：2022-12-06 11:23:25

Description of Preset Roles

The Tencent Kubernetes Engine (TKE) console provides fine-grained permission control for Kubernetes resources based on Kubernetes' native Role-Based Access Control (RBAC) authorization policies. It also provides the preset roles `Role` and `ClusterRole` , which are described below:

Role

The TKE console provides an access management page for which the **root account** and **cluster creator** by default have administrator permissions and can manage sub-accounts that have the DescribeCluster Action permission for a given cluster. See the following figure for more information.



Name	Labels	Operation
lb-ingress-clusterrole	N/A	Delete
tke-bridge-agent	N/A	Delete
tke-cni-clusterrole	N/A	Delete
tke:admin	cloud.tencent.com/tke-rbac-generated:true	Delete
tke:ns:ro	cloud.tencent.com/tke-rbac-generated:true	Delete

ClusterRole

- **For all namespaces**:
- **Administrators (tke:admin)**: have read/write permission for the resources in all namespaces, read/write permission for cluster nodes, storage volumes, namespaces, and quotas, and read/write permission for sub-account configurations.
- **OPS personnel (tke:ops)**: have read/write permission for the resources visible on the console in all namespaces and read/write permission for cluster nodes, storage volumes, namespaces, and quotas.
- **Developers (tke:dev)**: have read/write permission for the resources visible on the console in all namespaces.
- **Restricted personnel (tke:ro)**: have read-only permission for the resources visible on the console in all namespaces.

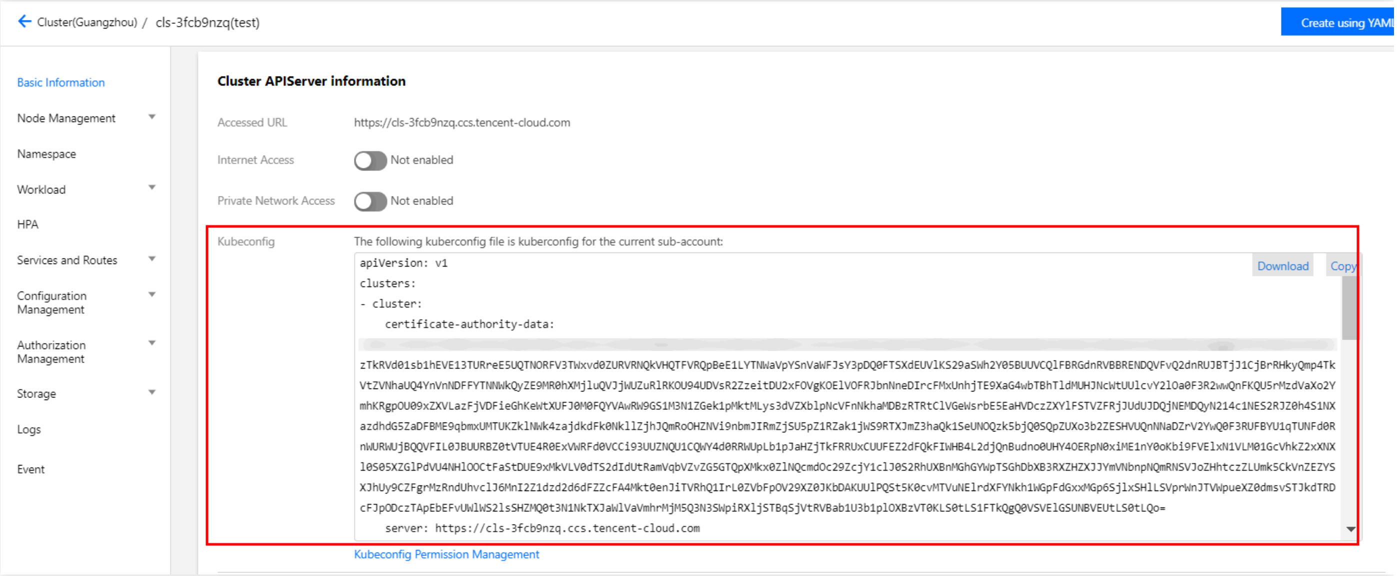
- **Custom:** user-defined ClusterRole.
- **For a specified namespace**:
  - **Developers (tke:ns:dev)**: have read/write permission for the resources visible on the console in a specified namespace.
  - **Read-only users (tke:ns:ro)**: have read-only permission for the resources visible on the console in a specified namespace.
- All the preset ClusterRole policies contain the fixed label: `cloud.tencent.com/tke-rbac-generated: "true"` .
- All the preset ClusterRoleBinding policies contain the fixed annotation: `cloud.tencent.com/tke-account-nickname: yournickname` and the label: `cloud.tencent.com/tke-account: "yourUIN"` .

Directions

Obtaining credentials

TKE will create independent credentials for each sub-account by default. You only need to access the cluster details page or call the Tencent Cloud API DescribeClusterKubeconfig to obtain the credential file `Kubeconfig` of the current account. The procedure for obtaining the file on the console is as follows:

1. Log in to the TKE console and click **Cluster** on the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster.
3. On the cluster details page, select **Basic Information** on the left sidebar. Then, you can view and download the Kubeconfig file in the **Cluster APIServer information** section, as shown in the following figure.



Cluster APIServer information

Accessed URL https://cls-3fcb9nzq.ccs.tencent-cloud.com

Internet Access ○ Not enabled

Private Network Access ○ Not enabled

Kubeconfig The following kuberconfig file is kuberconfig for the current sub-account:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
[redacted]
zTkRVd01sb1hEVE13TURreE5UQTNORFV3TWxvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTjJ1CjBrRHkyQmp4Tk
VtZVNhaUQ4YnVnNDFFYTNNWkQyZE9MR0hXMjluQVJjWUZuRlRKOU94UDVsR2ZzeitDU2xFOVgKOElVOFRJbnNneDIrcFMxUnhjTE9XaG4wbTBhTldMUHJNcWtUUlcvY2lOa0F3R2wwQnFKQU5rMzdVaXo2Y
mhKRgpOU09xZXVLazFjVDFieGhKeWtXUFJ0M0FQYVAwRW9GS1M3N1ZGek1pMktMLys3dVZXblpNcVFnNkhaMDBzRTRtClVGeWsrbE5EaHVDczZXYlFSTVZFRjJUdUJDQjNEMDQyN214c1NES2RJZ0h4S1NX
azdhdG5ZaDFBME9qbmxUMTUKZklNWk4zajdkdFk0NkllZjhJQmRoOHZNVi9nbmJIRmZjSU5pZ1RZak1jWS9RTXJmZ3haQk1SeUNOQzk5bjQ0SQpZUXo3b2ZESHVUQnNNaDZrV2YwQ0F3RUFBYU1qTUNFd0R
nWURWUjBQQVFIL0JBUURBZ0tVTUE4R0ExVWRFd0VCCi93UUZNQU1CQWY4d0RRWUpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFIWHB4L2djQnBudno0UHY4OERpN0xiME1nY0oKbi9FVElxN1VLM01GcVhkZ2xXNX
l0S05XZGlPdVU4NHlOOCtFaStDUE9xMkVLV0dTS2dIdUtRamVqbVZvZG5GTQpXMkx0ZlNQcmdOc29ZcjY1clJ0S2RhUXBnMGhGYWpTSGhDbXB3RXZHZXJJYmVNbnpNQmRNSVJoZHhtczZLUmk5CkVnZEZYS
XJhUy9CZFgrMzRndUhvclJ6MnI2Z1dzd2d6dFZZcFA4Mkt0enJiTVRhQ1IrL0ZVbFpOV29XZ0JKbDAKUUlPQSt5K0cvMTVuNElrdXFYNkh1WGpFdGxxMGp6SjlxSHlLSVprWnJTVWpueXZ0dmsvSTJkdTRD
cFJpODczTApEbEFvUWlWS2lsSHZMQ0t3N1NkTXJaWlVaVmhrMjM5Q3N3SWpiRXljSTBqSjVtRVBab1U3b1plOXBzVT0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
    server: https://cls-3fcb9nzq.ccs.tencent-cloud.com
```

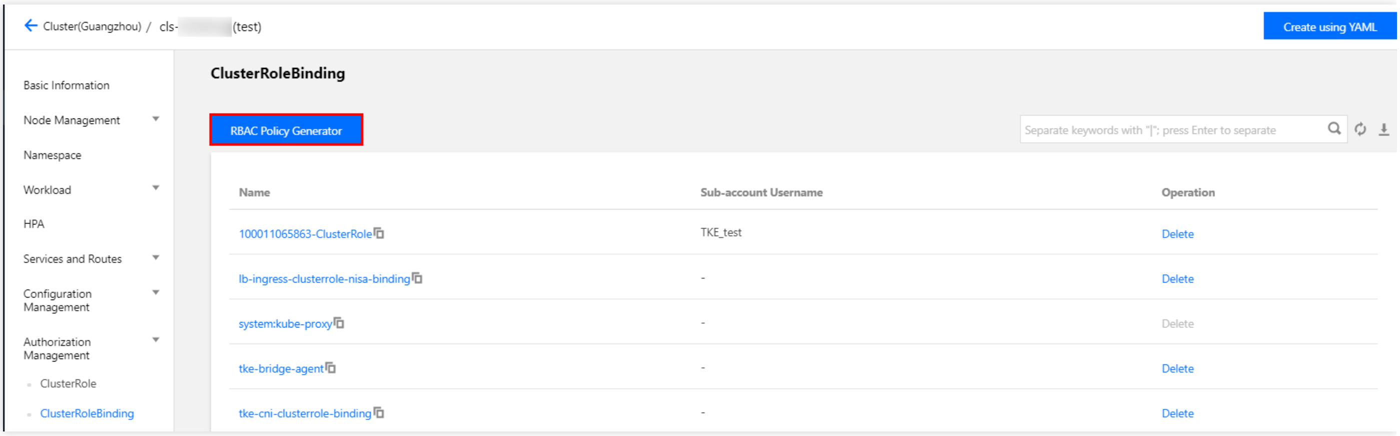
Managing credentials

Cluster administrators can access the credential management page to view and update the cluster credentials of all accounts. For more information, see Updating the TKE cluster access credentials of sub-accounts.

Authorization

> Note：
> Please contact cluster administrators (root accounts, cluster creators, or users with the admin role) for authorization.

1. On the **Cluster Management** page, click the ID of the target cluster.
2. On the cluster details page, select **Authorization Management** -> **ClusterRoleBinding** on the left sidebar.
3. On the **ClusterRoleBinding** page, click **RBAC Policy Generator**, as shown in the following figure.



ClusterRoleBinding

Name	Sub-account Username	Operation
100011065863-ClusterRole	TKE_test	Delete
lb-ingress-clusterrole-nisa-binding	-	Delete
system:kube-proxy	-	Delete
tke-bridge-agent	-	Delete
tke-cni-clusterrole-binding	-	Delete

4. When you select a sub-account on the **Administration Permissions** page, select the target sub-account and click **Next**.
5. When you set the cluster RBAC, set the permissions as follows:

- **Namespace List**: specify the namespaces for which the permissions apply.
- **Permissions**: please reference the descriptions provided on the page and set permissions as needed.

> Note：
> You can also click **Add Permission** to set custom permissions.

Authentication

Log in to your sub-account and verify that the sub-account has the permissions in question. If so, this indicates that the authorization was successful.

Custom Policy Authorization

Last updated：2021-08-17 15:54:03

This document describes how to grant specified permissions to a sub-account by customizing ClusterRoles and Roles in Kubernetes to fit your specific business requirements.

Policy Syntax Description

You can write your own policy syntax or use the Cloud Access Management (CAM) policy generator to create custom policies. An example YAML is shown below:

Role: for a namespace

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
name: testRole
namespace: default
rules:
- apiGroups:
- ""
resources:
- pods
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
```

ClusterRole: for a cluster

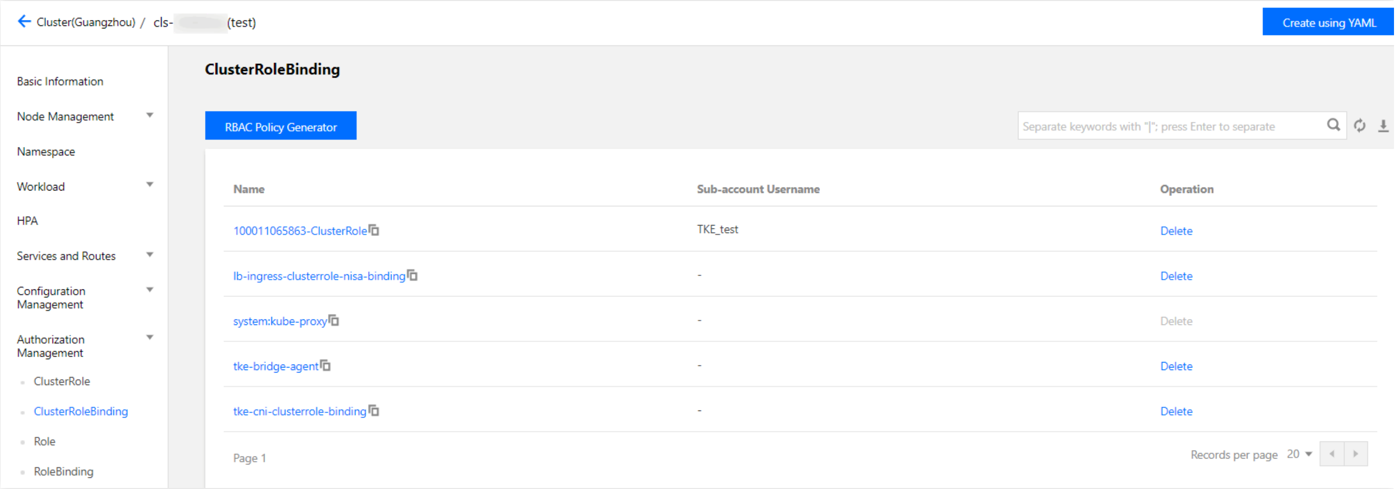
```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: testClusterRole
rules:
- apiGroups:
```

```
- ""
resources:
- pods
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
```

Directions

> Note：
> This section describes how to bind a custom ClusterRole policy to a sub-account. This operation is basically the same as that for binding a Role policy. Following the directions below, you can bind policies to fit your specific business requirements.

1. Log in to the TKE console and click **Cluster** on the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster.
3. On the cluster details page, select **Authorization Management** -> **ClusterRole** on the left sidebar, as shown in the following figure.



Cluster(Guangzhou) / cls-[redacted](test)

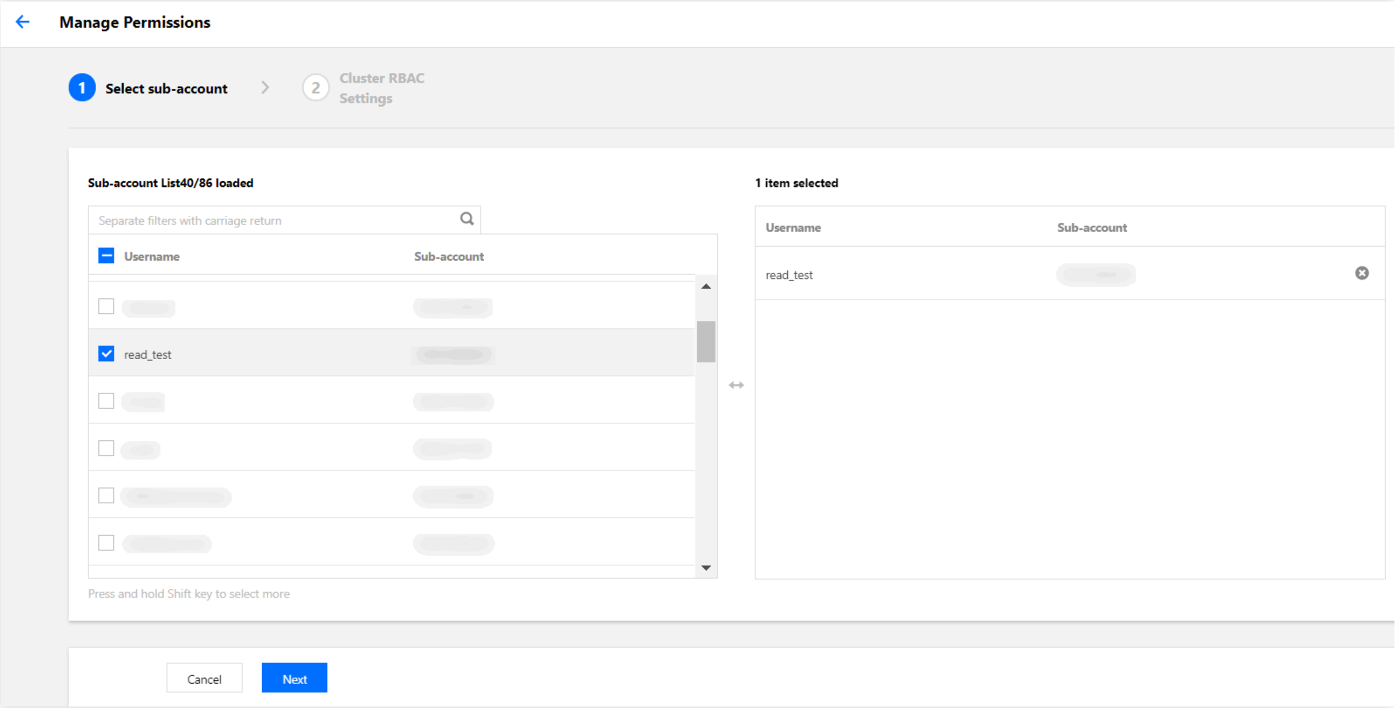
ClusterRoleBinding

Name	Sub-account Username	Operation
100011065863-ClusterRole	TKE_test	Delete
lb-ingress-clusterrole-nisa-binding	-	Delete
system:kube-proxy	-	Delete
tke-bridge-agent	-	Delete
tke-cni-clusterrole-binding	-	Delete

Page 1

4. On the **ClusterRole** page, select **Create using YAML** in the upper-right corner.

5. On the editing page, enter the YAML content of the custom policy and then click **Complete** to create the ClusterRole policy.
   For this step, the ClusterRole: for a cluster YAML is used as an example. After the policy is created, you can view the custom permission `testClusterRole` on the **ClusterRole** page.
6. On the **ClusterRoleBinding** page, click **RBAC Policy Generator**.
7. When you select a sub-account on the **Administration Permissions** page, select the target sub-account and click **Next**, as shown in the following figure.



Manage Permissions

1 Select sub-account > 2 Cluster RBAC Settings

Sub-account List40/86 loaded

Separate filters with carriage return

	Username	Sub-account
□	[redacted]	[redacted]
☑	read_test	[redacted]
□	[redacted]	[redacted]
□	[redacted]	[redacted]
□	[redacted]	[redacted]
□	[redacted]	[redacted]

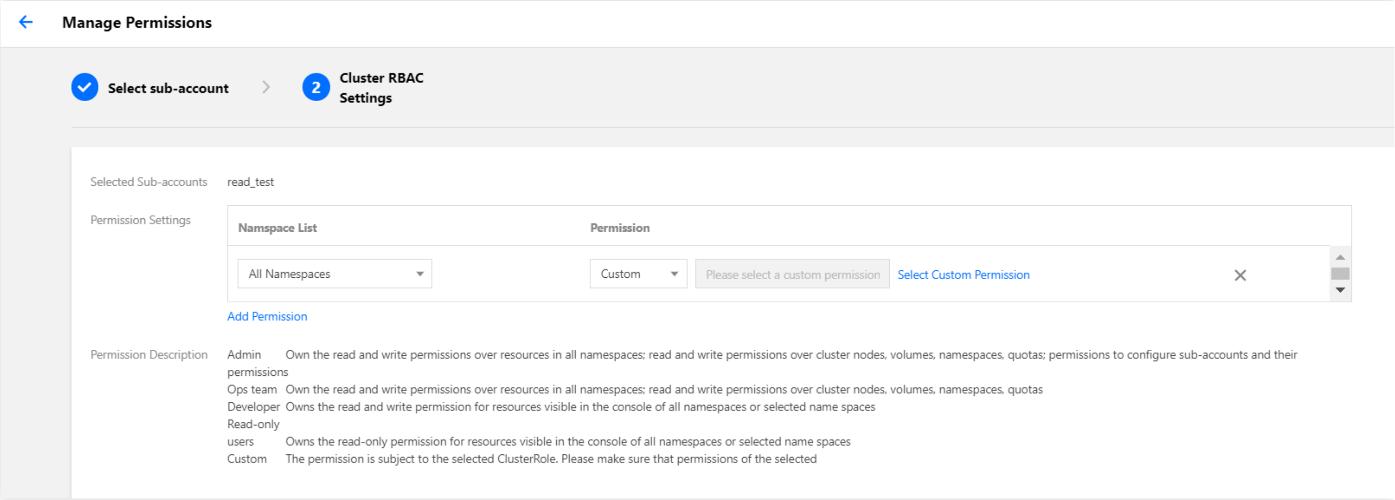
Press and hold Shift key to select more

1 item selected

Username	Sub-account
read_test	[redacted]

Cancel Next

8. On the **Cluster RBAC Setting** page, set the permissions as instructed, as shown in the following figure.



Manage Permissions

✓ Select sub-account > 2 Cluster RBAC Settings

Selected Sub-accounts read_test

Permission Settings

Namspace List	Permission
All Namespaces	Custom · Please select a custom permission · Select Custom Permission

Add Permission

Permission Description
Admin Own the read and write permissions over resources in all namespaces; read and write permissions over cluster nodes, volumes, namespaces, quotas; permissions to configure sub-accounts and their permissions
Ops team Own the read and write permissions over resources in all namespaces; read and write permissions over cluster nodes, volumes, namespaces, quotas
Developer Owns the read and write permission for resources visible in the console of all namespaces or selected name spaces
Read-only users Owns the read-only permission for resources visible in the console of all namespaces or selected name spaces
Custom The permission is subject to the selected ClusterRole. Please make sure that permissions of the selected

- **Namespace List**: specify the namespaces for which the permissions apply.

- **Permissions**: select **Custom** and click **Select Custom Permissions**. Then, select the desired permissions from the custom permission list. Here, we select the previously created custom permission `testClusterRole` as an example.

  > Note：
  > You can also click **Add Permission** to continue customizing the permissions.

9. Click **Done** to complete the authorization.

For your Reference

For more information, see the Kubernetes official documentation: Using RBAC for authorization.

Updating the TKE Cluster Access Credentials of Sub-accounts

Last updated：2022-03-30 18:09:30

Access Credentials

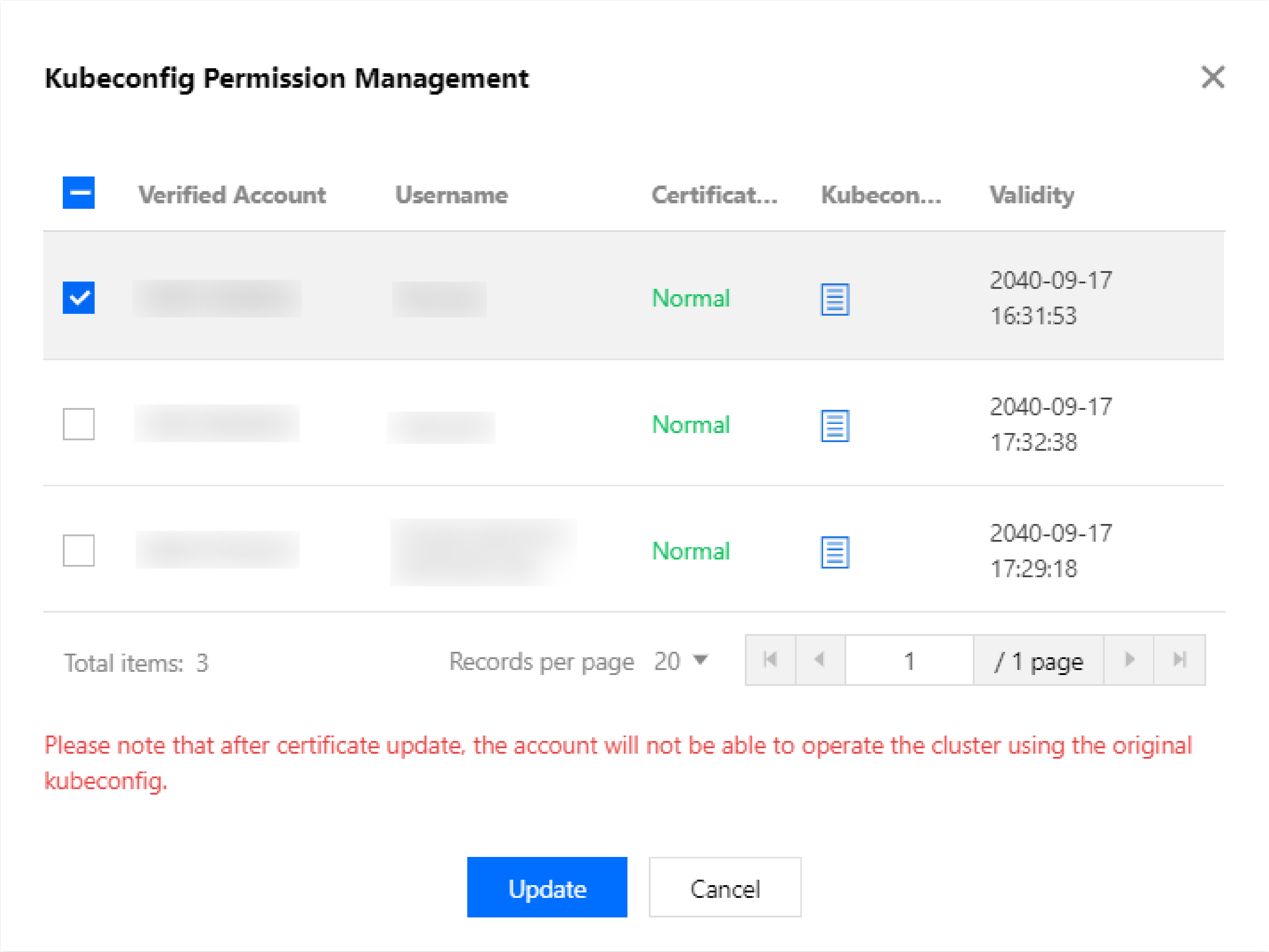
Tencent Kubernetes Engine (TKE) implements the following features based on x509 certificates:

- Each sub-account has a unique client certificate used for accessing Kubernetes API servers.
- Under the new authorization method adopted by TKE, when different sub-accounts obtain access credentials for a cluster (i.e., for accessing the basic information page of the cluster or calling the DescribeClusterKubeconfig API), they will obtain a unique x509 client certificate, which is issued by the self-signed CA of each cluster.
- When a sub-account accesses Kubernetes resources on the console, the backend uses the sub-account’s client certificate to access the Kubernetes API server by default.
- A sub-account can update its unique client certificate to prevent credential disclosure.
- A root account or an account that has `tke:admin` permission for a cluster can view and update the certificates of other sub-accounts.

Directions

1. Log in to the TKE console and click **Cluster** on the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster.
3. On the cluster details page, click **Basic Information** on the left sidebar. In the **Cluster APIServer information** section, click **Kubeconfig**.

4. On the **Kubeconfig** page, select the authentication account and click **Update**, as shown in the following figure.



Kubeconfig Permission Management

☐	Verified Account	Username	Certificat...	Kubecon...	Validity
☑	[redacted]	[redacted]	Normal	🗎	2040-09-17 16:31:53
☐	[redacted]	[redacted]	Normal	🗎	2040-09-17 17:32:38
☐	[redacted]	[redacted]	Normal	🗎	2040-09-17 17:29:18

Total items: 3 Records per page 20 ▾ 1 / 1 page

Please note that after certificate update, the account will not be able to operate the cluster using the original kubeconfig.

Cluster Management

Cluster Overview

Last updated：2022-04-25 12:28:55

Cluster Overview

A cluster is a collection of cloud resources required for running a container, including several CVMs and CLBs. You can run your applications in your cluster.

Cluster Architecture

A TKE cluster is compatible with Kubernetes, which includes the following components:

- Master: used to control nodes of the management plane of a cluster.
- Etcd: used to retain the status information of the entire cluster.
- Node: worker nodes used to run applications.

Cluster Types

TKE supports the following cluster types:

Cluster Type	Description
Managed cluster	Master and Etcd are managed by TKE
Self-deployed cluster	Master and Etcd nodes are built on your owned servers

For more information, see Cluster Hosting Mode Instruction.

Cluster Lifecycle

For more information on TKE cluster lifecycle, see Cluster Lifecycle.

Cluster-related Operations

- Creating a Cluster
- Changing the Cluster Operating System
- Cluster Scaling
- Connecting to a Cluster
- Upgrading a Cluster
- Enabling IPVS for a Cluster
- Enabling GPU Scheduling for a Cluster
- How to Choose TKE Network Mode
- Deleting a Cluster
- Custom Kubernetes Component Launch Parameters

Cluster Hosting Modes Introduction

Last updated：2022-05-12 14:44:44

Managed Master Mode

Overview

Tencent Kubernetes Engine (TKE) provides management services for Kubernetes clusters where Master and Etcd nodes are fully managed.
In this mode, the Master and Etcd nodes of your Kubernetes cluster will be centrally managed and maintained by the Tencent Cloud technical team. You only need to purchase the cluster and run worker nodes required for your business load, with no need to worry about management and maintenance issues.

Notes on the managed master mode

- TKE charges cluster management fees based on the specifications of the managed clusters, and charges cloud resources (CVM, persistent storage, and CLB) fees based on the actual usage. For more information about the billing modes and prices, see TKE Billing Overview.
- Master and Etcd nodes in this mode are not user-specific resources. Therefore, you cannot modify their deployment scale and service parameters. If you need to do so, Select the Independent Master Deployment Mode.
- In this mode, even if you delete all the worker nodes from the cluster, the cluster will still persistently attempt to run workloads and services that have not been deleted. This process may incur fees. If you need to stop the services and prevent cluster fees, delete the cluster.

Independent Master Deployment Mode

Overview

TKE also provides an independent Master deployment mode, which gives you full control over your Kubernetes cluster.
In this mode, the Master and Etcd nodes of the Kubernetes cluster are deployed on your Cloud Virtual Machines (CVMs). You have all permissions to manage and operate the Kubernetes cluster.

Notes on the independent master deployment mode

- This mode is available only for Kubernetes 1.10.x or later.
- You need to purchase resources for deploying Master and Etcd of the Kubernetes cluster.
- If your cluster contains a large number of nodes, it is recommended that you select a high-spec CVM model. The following table provides reference for model selection.

Cluster Size	Recommended Master Node Configuration	Recommended Node Quantity
Around 100 nodes	8-core 16 GB SSD system disk	Three or more
Around 500 nodes	16-core 32 GB SSD system disk	Three or more
1,000 nodes or more	Submit a ticket	Three or more

Purchase Requirements

To ensure the high availability of clusters and services and improve cluster performance, it is recommended that you comply with the following requirements in independent deployment mode:

- Deploy at least 3 Master and Etcd nodes.
- Use models with at least 4 cores for Master and Etcd nodes.
- Use SSD disks as system disks of Master and Etcd nodes.

Notes

To ensure the stability of your cluster and improve failback efficiency, we recommend that you note the following:

- In the independent Master deployment mode:
- Do not delete the core components of the Master node that support the operation of the Kubernetes cluster.
- Do not modify the configuration parameters of the core components of the Master node.
- Do not modify or delete the core resources in the Kubernetes cluster.
- Do not modify or delete the relevant certificate files (with .crt and .key extensions) of the Master node.
- Unless otherwise required:
- Do not modify the Docker version of any node.
- Do not modify the OS components, such as kernel and nfs-utils, of any node.

Note：

- Core components include kube-APIserver, kube-scheduler, kube-controller-manager, tke-tools, systemd, and cluster-container-agent.
- Configuration parameters of core components include kube-APIserver parameters, kube-scheduler parameters, and kube-controller-manager parameters.
- Core resources in the cluster (including but not limited to): hpa endpoint, master service account, kube-dns, auto-scaler, master cluster role, and master cluster role binding.

If you have any questions about the above suggestions, submit a ticket.

Cluster Lifecycle

Last updated：2022-12-14 15:36:26

Notes on Cluster Lifecycle Status

Status	Description
Creating	The cluster is being created and is applying for Tencent Cloud resources.
Scaling	The number of nodes in the cluster is being changed or nodes are being added or terminated.
Running	The cluster is running normally.
Upgrading	The cluster is being upgraded.
Deleting	The cluster is being deleted.
Abnormal	There are exceptions in the cluster, such as node network inaccessibility.
Isolated	The managed cluster is being isolated due to overdue payments exceeding 24 hours. Billing for cluster management has stopped.

Note：

TKE is based on Kubernetes and is a declarative service. If you have created IaaS resources such as CLB instances or CBS cloud disks in TKE and no longer need to use them, delete the corresponding Service and PersistentVolumeClaim objects in the TKE console. If you only delete the load balancers in the CLB console or the cloud disks in the CBS console, TKE will recreate them and fees will continue to be incurred.

Creating a Cluster

Last updated：2023-07-07 17:48:59

This document describes how to create a general cluster and configure the VPCs, subnets and security groups in the TKE console.

Prerequisites

Before creating a cluster, you need to complete the following preparations:

Sign up for a Tencent Cloud account.

When you log in to the TKE console for the first time, you need to grant the current account TKE permissions for operating on CVMs, CLBs, CBS, and other cloud resources. For more information, see Description of Role Permissions Related to Service Authorization.

To create a cluster whose network type is virtual private cloud (VPC), you need to create a VPC in the target region and create a subnet in the target availability zone under the VPC.

If you do not use the default security group, you need to create a security group in the target region and add a security group rule that meets your business requirements.

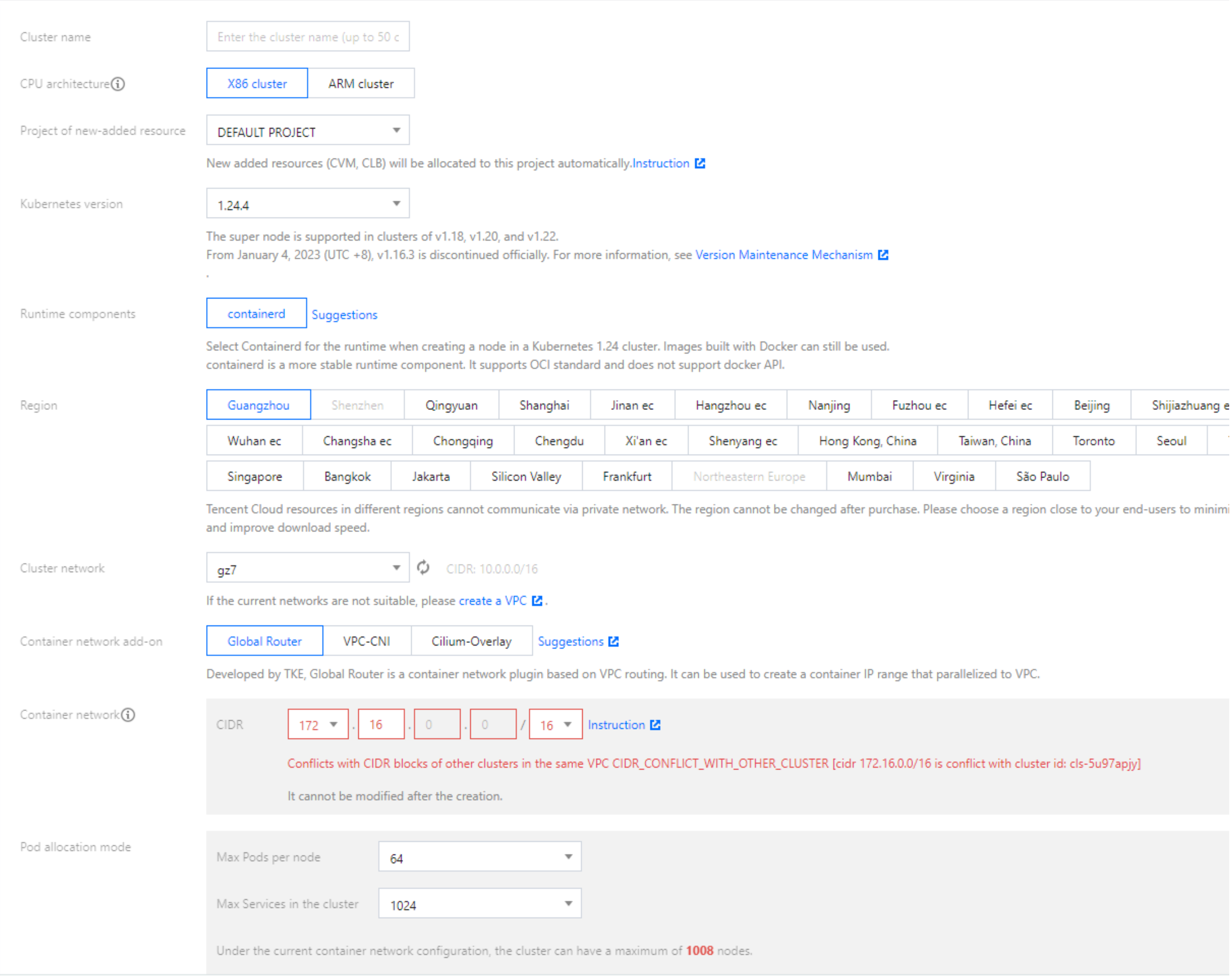
To bind an SSH key pair when creating a Linux instance, you need to create SSH keys for the target project.

When you create a cluster, you will use the resources such as VPCs, subnets, and security groups. Each region has a resource quota. For more information, see Quota Limits for Cluster Purchase.

Create a Cluster in the TKE Console

1. Enter the cluster information

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the "Cluster management" page, click **Create** above the cluster list.
3. Select **General cluster**, and click **Create**.
4. On **Create Cluster** page, configure the basic information of the cluster as shown in the figure below:



Cluster name: Enter the cluster name (up to 50 c

CPU architecture: X86 cluster | ARM cluster

Project of new-added resource: DEFAULT PROJECT

New added resources (CVM, CLB) will be allocated to this project automatically.Instruction

Kubernetes version: 1.24.4

The super node is supported in clusters of v1.18, v1.20, and v1.22.
From January 4, 2023 (UTC +8), v1.16.3 is discontinued officially. For more information, see Version Maintenance Mechanism
.

Runtime components: containerd Suggestions

Select Containerd for the runtime when creating a node in a Kubernetes 1.24 cluster. Images built with Docker can still be used.
containerd is a more stable runtime component. It supports OCI standard and does not support docker API.

Region: Guangzhou

Tencent Cloud resources in different regions cannot communicate via private network. The region cannot be changed after purchase. Please choose a region close to your end-users to minimi
and improve download speed.

Cluster network: gz7 CIDR: 10.0.0.0/16

If the current networks are not suitable, please create a VPC.

Container network add-on: Global Router | VPC-CNI | Cilium-Overlay Suggestions

Developed by TKE, Global Router is a container network plugin based on VPC routing. It can be used to create a container IP range that parallelized to VPC.

Container network: CIDR 172 . 16 . 0 . 0 / 16 Instruction

Conflicts with CIDR blocks of other clusters in the same VPC CIDR_CONFLICT_WITH_OTHER_CLUSTER [cidr 172.16.0.0/16 is conflict with cluster id: cls-5u97apjy]

It cannot be modified after the creation.

Pod allocation mode:

Max Pods per node: 64

Max Services in the cluster: 1024

Under the current container network configuration, the cluster can have a maximum of **1008** nodes.

Cluster name: Set the name of the cluster with up to 50 characters.

Project of new-added resource: Select a project as needed. The newly added resources will be automatically assigned to this project.

Kubernetes version: Multiple Kubernetes versions are available. For feature comparison between different versions, see Supported Versions of the Kubernetes Documentation.

Runtime components: Select **docker** or **containerd**. For more information, see How to Choose Containerd and Docker.

Region: It is recommended that you select a region that is close to your location. For more information, see Regions and Availability Zones.

Cluster network: Assigns IP addresses that are within the node network address range to CVMs in the cluster. For details, see Network Settings for Containers and Nodes.

Container network add-on: GlobalRouter, VPC-CNI and Cilium-Overlay are provided. For more information, see How to Choose a TKE Network Mode.

Container network: Assigns IP addresses that are within the container network IP range to containers in the cluster. For details, see Container Network Overview.

Image provider: You can select a public image or custom image. For more information, see Image Overview.

Operating system: Select the operating system based on your requirements.

Cluster description: Enter information about the cluster, which will be displayed on the **Cluster information** page.

Advanced settings (optional):

Tencent Cloud tags: After binding tags to the cluster, you can categorize the resources. For more information, see Querying Resources by Tag.

Deletion protection: When it's enabled, the cluster will not be deleted by misoperation in the console or via the API.

Kube-proxy proxy mode: Select **iptables** or **ipvs**. IPVS mode is applicable to large-scale services. You cannot disable it once it is enabled. For more information, see Enabling IPVS for a Cluster.

Custom parameters: Configure the cluster with custom parameters. For more information, see Custom Kubernetes Component Launch Parameters.

Runtime version: Select the version of the container runtime component.

5. Click **Next**.

2. Select a model

On the **Select model** page, confirm the billing mode, select an AZ and the corresponding subnet, and confirm the node model.

1. Select **Add node** or **Existing nodes** for **Node source**.

Adding a Node

Existing nodes

Create a cluster by adding nodes (that is, by adding CVMs). The details are as follows:

Cluster type: You can select **Managed cluster** or **Self-deployed cluster**.

Managed cluster: The Master and Etcd of the Kubernetes cluster will be managed and maintained by Tencent Cloud.

Self-deployed cluster: The Master and Etcd of the Kubernetes cluster will be deployed on the CVM instance you purchased.

Cluster specification: Select an appropriate cluster specification as needed. For more information, see Purchase Instructions. You can adjust the cluster specification manually, or enable Auto Cluster Upgrade to have it adjusted automatically.

Billing Mode: **Pay-as-you-go** is supported. For more information, see Billing Plans.

Worker configurations: If you select **Add node** for **Node source** and **Managed cluster** for **Cluster type**, all configuration items in this module are set to the default values. You can modify them as needed.

Availability zone: You can select multiple availability zones at the same time to deploy your Master or Etcd nodes to ensure higher availability of the cluster.

Node network: You can select multiple subnet resources at the same time to deploy your Master or Etcd nodes to ensure higher availability of the cluster.

Model: Choose a model higher than CPU 4-core. For details, see Instance Types.

System disk: The default value is “HDD cloud disk - 50 GB”. You can select local disk, HDD cloud disk, SSD cloud disk, or premium cloud disk based on your actual model. For details, see Storage Overview.

Data disk: As it is not recommended to deploy other applications on the Master and Etcd nodes, no data disk is configured for them by default. You can purchase one and add it if needed.

Public network bandwidth: Select **Assign free public IP** and the system will assign a public IP address for free. Two billing methods are available. For more information, see Public Network Billing.

Node name: The name of the computer in the OS (the node name displayed by running the `kubectl get nodes` command). It is a cluster attribute. The node name can be named in the following two modes:

Auto-generated: The node hostname defaults to the private IP of the node.

Custom name: You can use sequential numbering or custom format string. It can contain lower-case letters, numbers, hyphens ("-") and periods ("."). Symbols cannot be placed at the beginning nor end, and cannot be used consecutively. For more naming rules, see Batch Sequential Naming or Pattern String-Based Naming.

Note

Due to the naming restriction of kubernetes node, you can only use the lower-case letters when customizing the hostname, for example, 'cvm {R:13}-big{R:2}-test'.

Instance name: The CVM instance name displayed in the console, which is determined by the naming mode of the hostname.

When the node hostname is automatically generated, it supports sequential numbering or custom format for multiple instances. The instance name is automatically generated by default in the format of `tke_cluster id_worker`.

When the node hostname is customized, the instance name is the same as the hostname, without the need to reconfigure it.

CVM quantity: Set the number of instances as needed.

Note

If **Self-deployed cluster** is selected for **Cluster type**, you can also refer to “Worker configurations” to set the Master and Etcd nodes. Deploy at least three instances, which can be in different availability zones.

Create a cluster using the existing nodes (that is, by using the existing CVMs). The details are as follows:

Note

The selected CVMs will be reinstalled and all data in the system disk will be cleared.

The selected CVMs will be migrated to the project of the cluster. All related security groups will be unbound. You need to bind them manually again.

If you set the data disk mounting parameters when configuring the CVM, this parameter will be applied to **all Master and Worker nodes**. For more information, see the **Mount data disk** section in Adding an existing node.

Cluster type: You can select **Managed cluster** or **Self-deployed cluster**.

Managed cluster: The Master and Etcd of the Kubernetes cluster will be managed and maintained by Tencent Cloud.

Self-deployed cluster: The Master and Etcd of the Kubernetes cluster will be deployed on the CVM instance you purchased.

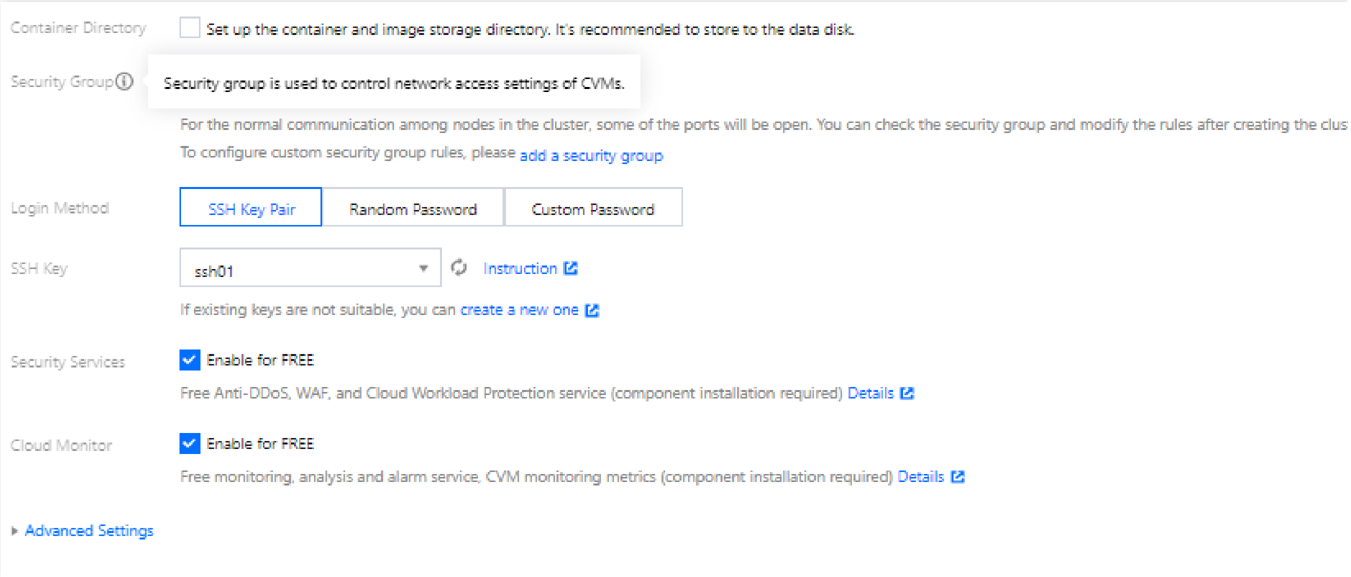
Cluster specification: Select an appropriate cluster specification as needed. For more information, see Purchase Instructions. You can adjust the cluster specification manually, or enable Auto Cluster Upgrade to have it adjusted automatically.

Worker configurations: Select the existing CVMs based on actual needs.

2. Click **Next** to start configuring a CVM.

3. Configure CVM

1. In the "CVM configuration" step, configure a CVM based on the following information:



Container Directory □ Set up the container and image storage directory. It's recommended to store to the data disk.

Security Group ⓘ Security group is used to control network access settings of CVMs.

For the normal communication among nodes in the cluster, some of the ports will be open. You can check the security group and modify the rules after creating the clus

To configure custom security group rules, please add a security group

Login Method: SSH Key Pair | Random Password | Custom Password

SSH Key: ssh01 Instruction

If existing keys are not suitable, you can create a new one

Security Services ☑ Enable for FREE

Free Anti-DDoS, WAF, and Cloud Workload Protection service (component installation required) Details

Cloud Monitor ☑ Enable for FREE

Free monitoring, analysis and alarm service, CVM monitoring metrics (component installation required) Details

▸ Advanced Settings

qGPU sharing: When it is enabled, GPU sharing is enabled for all added GPU nodes in the cluster by default. You can enable or disable GPU sharing through the Label. Note that the qGPU add-on must be installed if you want to use GPU sharing.

Container Directory: Select this option to set up the container and image storage directory. We recommend that you store to the data disk, such as `/var/lib/docker`.

Security Group: The security group works as a firewall to control access to the CVM network. The following settings are supported:

Create and bind the default security group. You can preview the default security group rules.

Add a security group to configure custom security group rules according to your actual needs. For details, see TKE Security Group Settings.

Login Method: Three login methods are available.

SSH Key Pair: A key pair is a pair of parameters generated by an algorithm. It is a way to log in to a CVM instance that is more secure than regular passwords. For more details, see SSH Key.

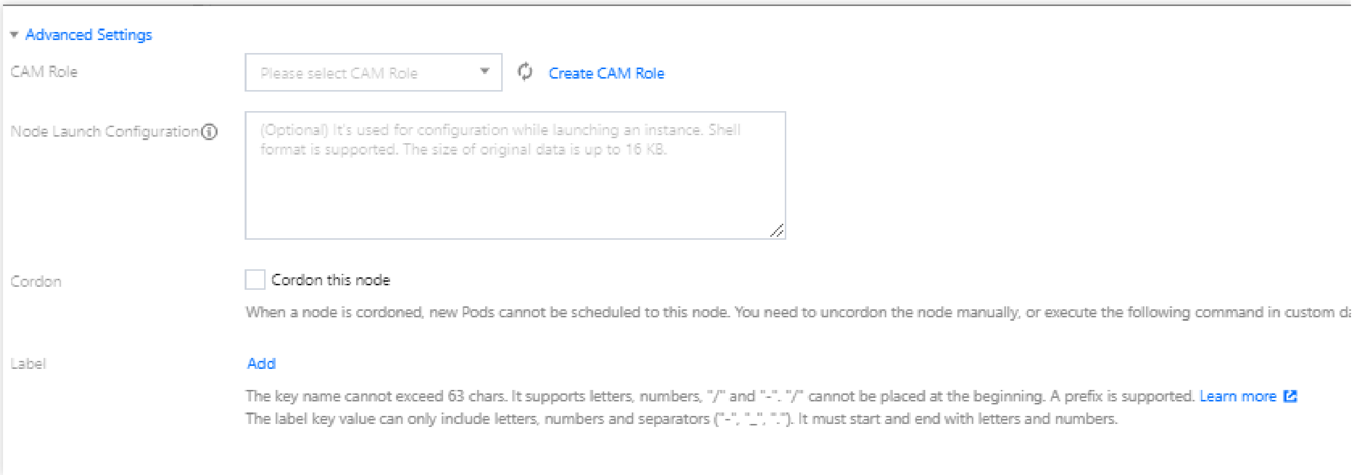
Random Password: The system sends an automatically generated password to your Message Center.

Custom Password: Set a password as prompted.

Security Services: Free DDoS, Web Application Firewall (WAF) and Cloud Workload Protection (CWP) are activated by default. For more information, see Cloud Workload Protection.

Cloud Monitor: Free monitoring, analysis, and alarms are activated by default, and components are installed to obtain CVM monitoring metrics. For more information, see Tencent Cloud Observability Platform.

2. (Optional) Click **Advanced Settings** to view or configure more information.



▾ Advanced Settings

CAM Role: Please select CAM Role ▾ Create CAM Role

Node Launch Configuration: (Optional) It's used for configuration while launching an instance. Shell format is supported. The size of original data is up to 16 KB.

Cordon: □ Cordon this node

When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually, or execute the following command in custom d

Label: Add

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more

The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

CAM Role: You can bind all the nodes created this time to the same CAM role, and grant the authorization policy bound to the role to the nodes. For more information, see Managing Roles.

Node Launch Configuration: Specify custom data to configure the node, that is, to run the configured script when the node is launched. You need to ensure the reentrant and retry logic of the script. The script and its log files can be viewed at the node path: `/usr/local/qcloud/tke/userscript` .

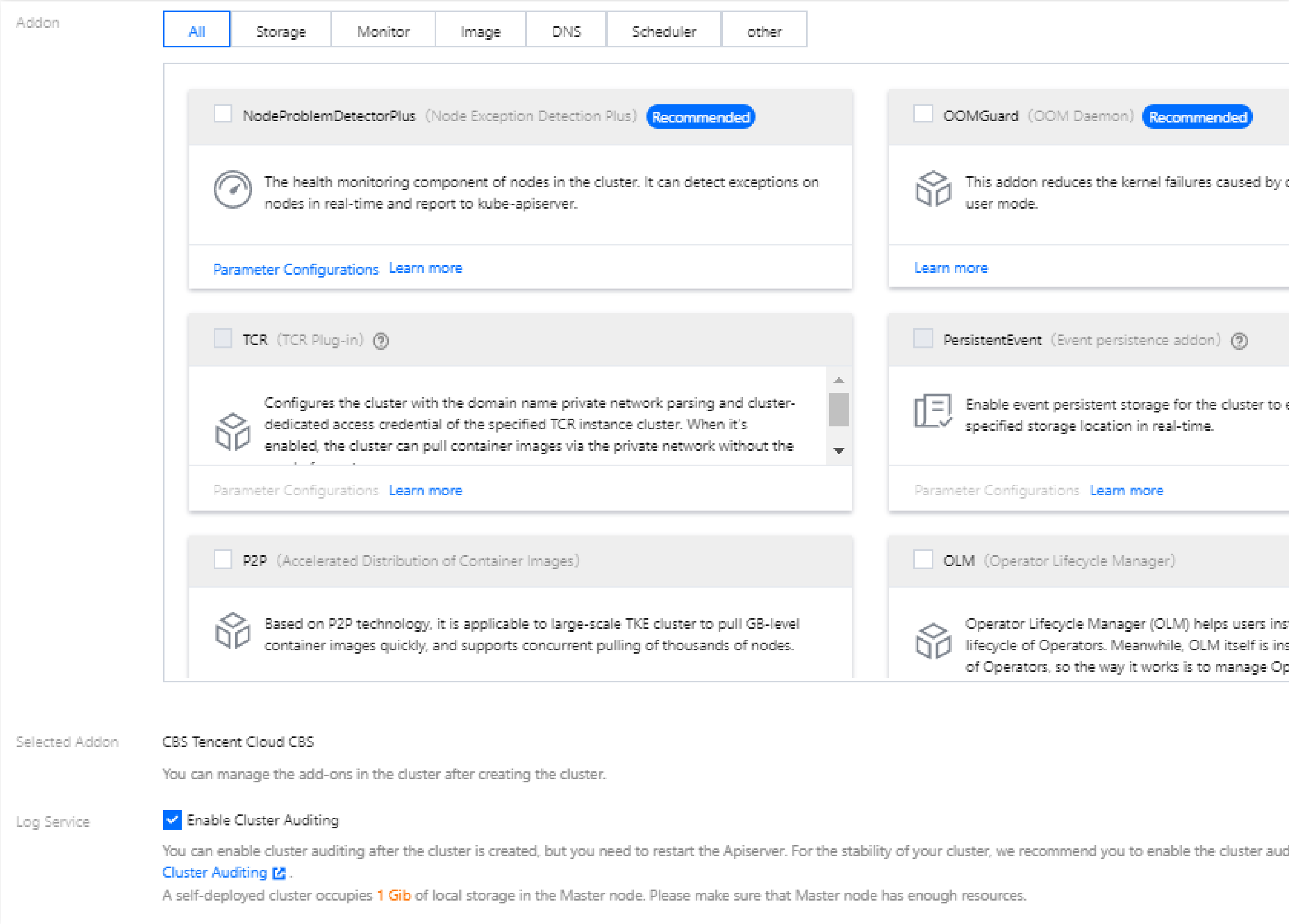
Cordon: After you check **Cordon this node**, new Pods cannot be scheduled to this node. You can uncordon the node manually, or execute the uncordon command in custom data as needed.

Label: Click **Add** to customize the label. The label set here will be automatically added to the initial nodes of the cluster, and is used to filter and manage nodes in the future.

3. Click **Next**.

4. Configure add-ons

1. Configure add-ons based on the following information:



Addon

All | Storage | Monitor | Image | DNS | Scheduler | other

□ NodeProblemDetectorPlus (Node Exception Detection Plus) Recommended

The health monitoring component of nodes in the cluster. It can detect exceptions on nodes in real-time and report to kube-apiserver.

Parameter Configurations Learn more

□ OOMGuard (OOM Daemon) Recommended

This addon reduces the kernel failures caused by c user mode.

Learn more

□ TCR (TCR Plug-in) ?

Configures the cluster with the domain name private network parsing and cluster-dedicated access credential of the specified TCR instance cluster. When it's enabled, the cluster can pull container images via the private network without the

Parameter Configurations Learn more

□ PersistentEvent (Event persistence addon) ?

Enable event persistent storage for the cluster to e specified storage location in real-time.

Parameter Configurations Learn more

□ P2P (Accelerated Distribution of Container Images)

Based on P2P technology, it is applicable to large-scale TKE cluster to pull GB-level container images quickly, and supports concurrent pulling of thousands of nodes.

□ OLM (Operator Lifecycle Manager)

Operator Lifecycle Manager (OLM) helps users ins lifecycle of Operators. Meanwhile, OLM itself is ins of Operators, so the way it works is to manage Op

Selected Addon CBS Tencent Cloud CBS

You can manage the add-ons in the cluster after creating the cluster.

Log Service ☑ Enable Cluster Auditing

You can enable cluster auditing after the cluster is created, but you need to restart the Apiserver. For the stability of your cluster, we recommend you to enable the cluster aud Cluster Auditing .

A self-deployed cluster occupies 1 Gib of local storage in the Master node. Please make sure that Master node has enough resources.

- **Addon**: You can select the add-ons such as storage, monitor, and image as needed. For more information, see Add-on Overview.

TMP: When it is enabled, you can configure data collection rules and alarm rules based on your needs. Then, you can check monitoring data on the Grafana dashboard. For more information, see TMP Overview.

Log Service: The cluster auditing is enabled by default. For more information, see Cluster Audit.

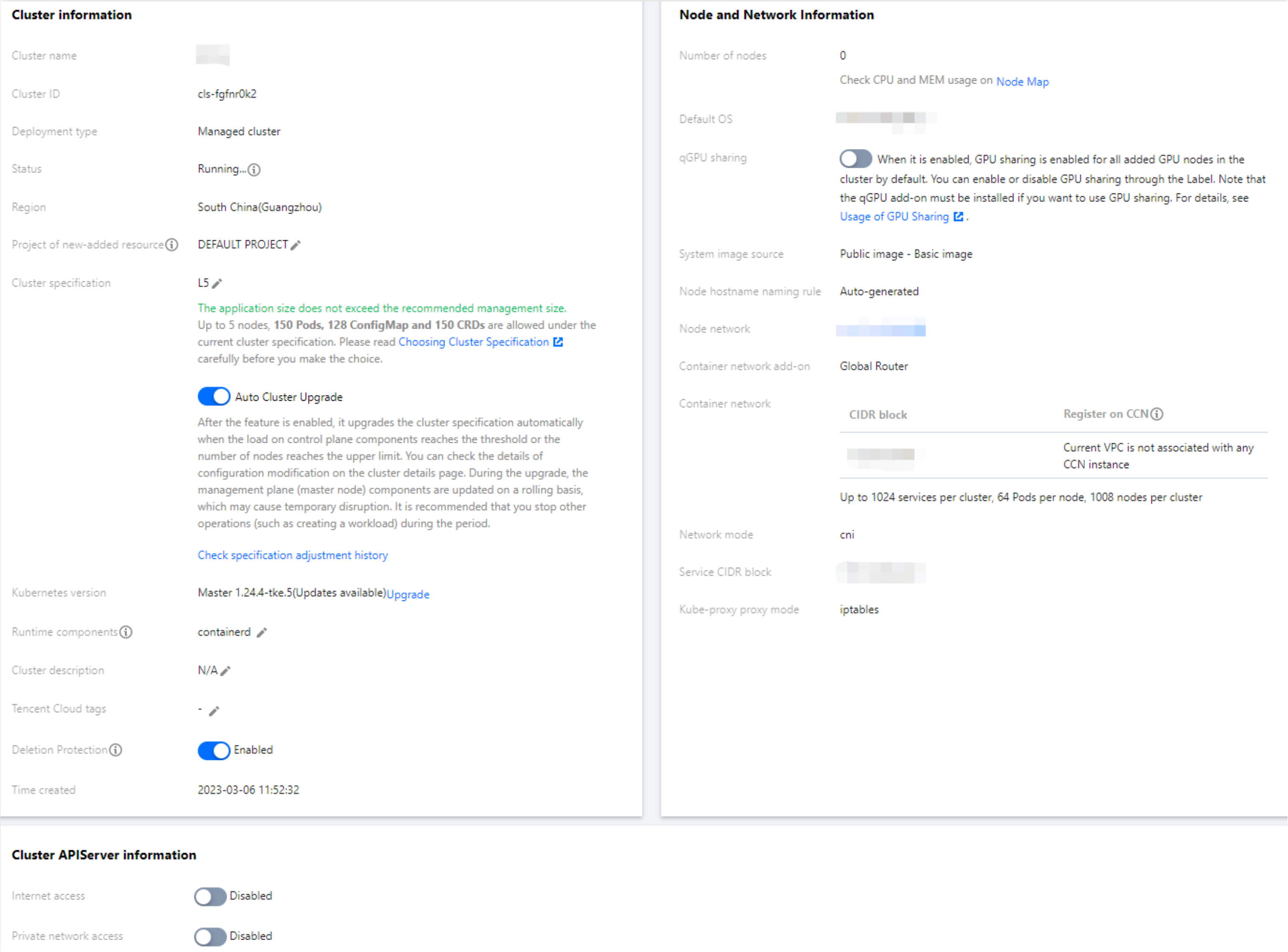
2. Click **Next**.

5. Confirm the information

On the **Confirm information** page, confirm the configuration and billing information for the cluster, and select **I have read and agree to** TKE Service Level Agreement**. Click** Done to complete the process.

6. View the cluster

You can view clusters that have been created in the cluster list. You can click the cluster ID to enter the details page, and then view the cluster, node, and network information on the "Basic information" page.



Cluster information

Cluster name	[redacted]
Cluster ID	cls-fgfnr0k2
Deployment type	Managed cluster
Status	Running...
Region	South China(Guangzhou)
Project of new-added resource	DEFAULT PROJECT
Cluster specification	L5 The application size does not exceed the recommended management size. Up to 5 nodes, **150 Pods, 128 ConfigMap and 150 CRDs** are allowed under the current cluster specification. Please read Choosing Cluster Specification carefully before you make the choice. Auto Cluster Upgrade After the feature is enabled, it upgrades the cluster specification automatically when the load on control plane components reaches the threshold or the number of nodes reaches the upper limit. You can check the details of configuration modification on the cluster details page. During the upgrade, the management plane (master node) components are updated on a rolling basis, which may cause temporary disruption. It is recommended that you stop other operations (such as creating a workload) during the period. Check specification adjustment history
Kubernetes version	Master 1.24.4-tke.5(Updates available)Upgrade
Runtime components	containerd
Cluster description	N/A
Tencent Cloud tags	-
Deletion Protection	Enabled
Time created	2023-03-06 11:52:32

Node and Network Information

Number of nodes	0 Check CPU and MEM usage on Node Map
Default OS	[redacted]
qGPU sharing	When it is enabled, GPU sharing is enabled for all added GPU nodes in the cluster by default. You can enable or disable GPU sharing through the Label. Note that the qGPU add-on must be installed if you want to use GPU sharing. For details, see Usage of GPU Sharing .
System image source	Public image - Basic image
Node hostname naming rule	Auto-generated
Node network	[redacted]
Container network add-on	Global Router
Container network	CIDR block: [redacted] Register on CCN: Current VPC is not associated with any CCN instance Up to 1024 services per cluster, 64 Pods per node, 1008 nodes per cluster
Network mode	cni
Service CIDR block	[redacted]
Kube-proxy proxy mode	iptables

Cluster APIServer information

Internet access	Disabled
Private network access	Disabled

Create a Cluster via the API

You can also use the `CreateCluster` API to create a cluster. For more information, see CreateCluster API Documentation.

Deleting a Cluster

Last updated：2022-08-02 17:19:05

Overview

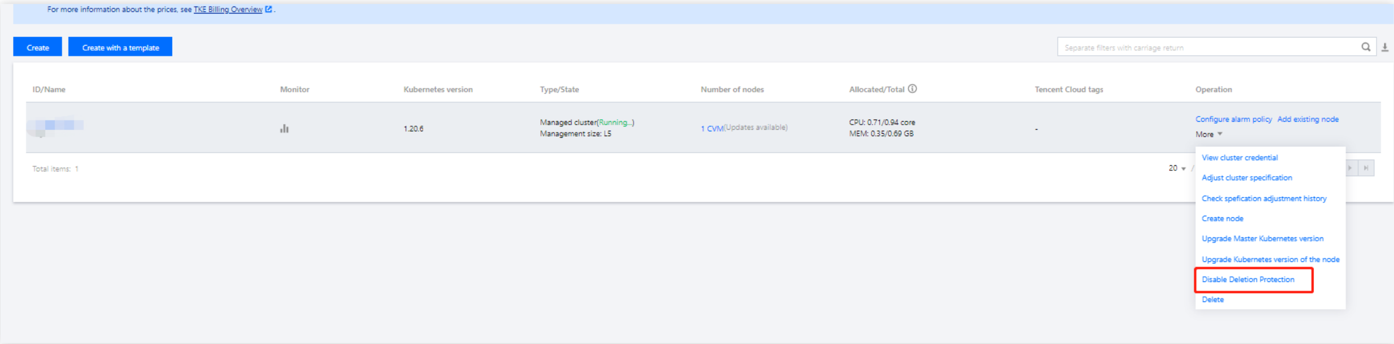
This document describes how to delete clusters that are no longer needed from the TKE console to avoid unnecessary costs. On the **Delete cluster** page, you can view all resources of a cluster, terminated resources, and choose whether to retain some resources as needed. Ensure that you are well aware of the operation risks before deleting clusters.

Directions

Disabling cluster deletion protection

Method 1

1. Log in to the TKE console, and click **Cluster** in the left sidebar.
2. On the “Cluster management” page, locate the desired cluster, and select **More** > **Disable deletion protection** in the "Operation" column.


For more information about the prices, see TKE Billing Overview.

ID/Name	Monitor	Kubernetes version	Type/State	Number of nodes	Allocated/Total	Tencent Cloud tags	Operation
[redacted]		1.20.6	Managed cluster(Running...) Management size: L5	1 CVM(Updates available)	CPU: 0.71/0.94 core MEM: 0.35/0.69 GB	-	Configure alarm policy Add existing node More

Total items: 1

Create
Create with a template
Separate filters with carriage return
20
View cluster credential
Adjust cluster specification
Check spefication adjustment history
Create node
Upgrade Master Kubernetes version
Upgrade Kubernetes version of the node
Disable Deletion Protection
Delete

3. Click **OK** in the pop-up window.

Method 2

1. Log in to the TKE console, and click **Cluster** in the left sidebar.
2. On the “Cluster management” page, click the name of the desired cluster to open the cluster details page.

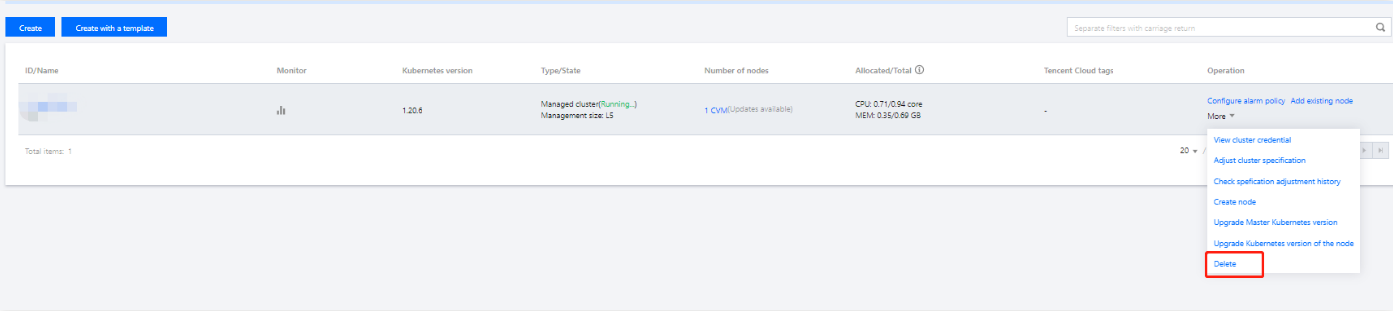
3. On the cluster's "Basic information" page, disable the deletion protection.

Deletion protection: Disabled

4. Click **OK** in the pop-up window.

Deleting a cluster

1. Log in to the TKE console, and click **Cluster** in the left sidebar.
2. On the **Cluster management** page, choose **More** > **Delete** on the right of the target cluster.



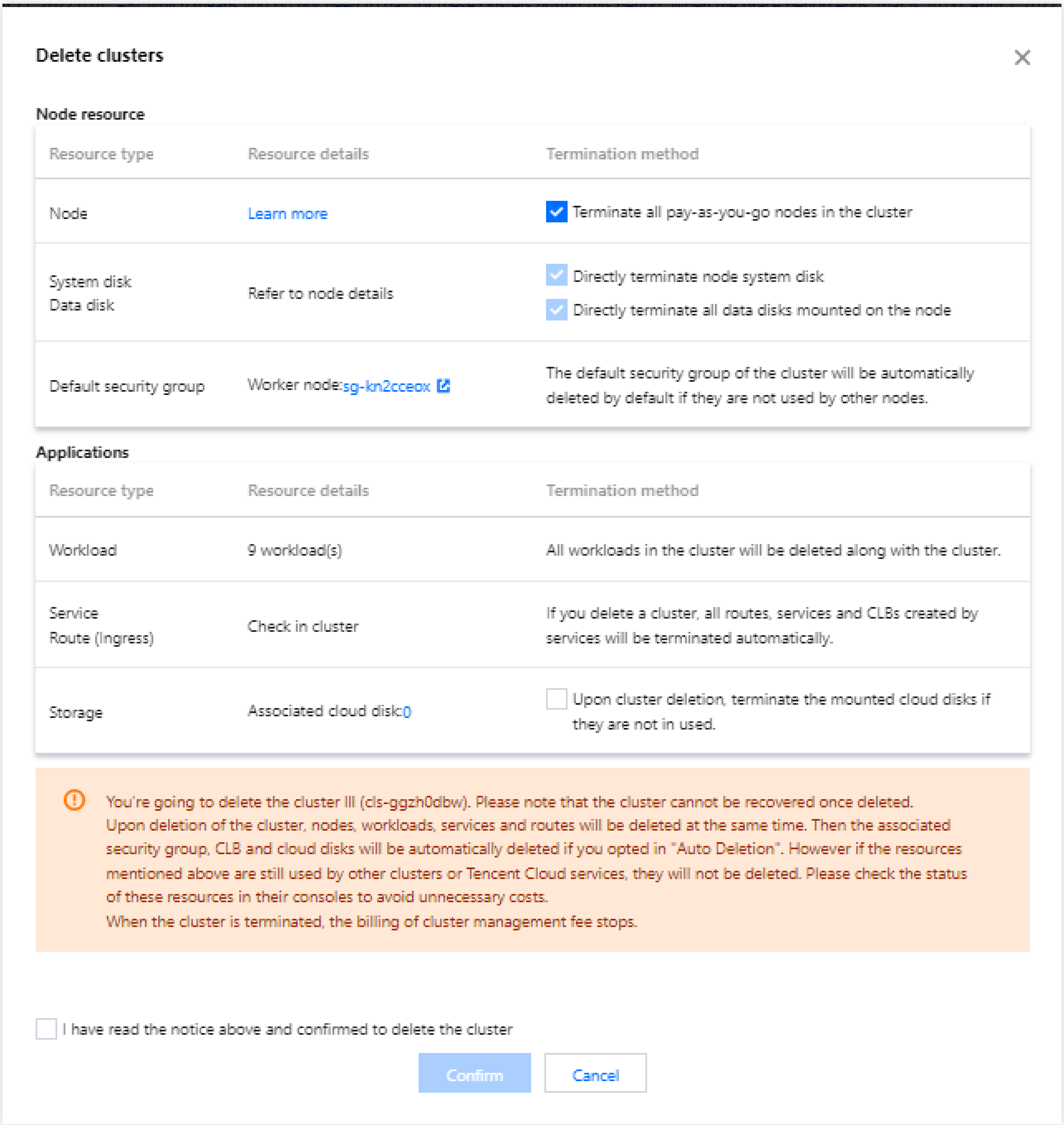
Create
Create with a template
Separate filters with carriage return

ID/Name	Monitor	Kubernetes version	Type/State	Number of nodes	Allocated/Total ⓘ	Tencent Cloud tags	Operation
[redacted]		1.20.6	Managed cluster(Running...) Management size: L5	1 CVM(Updates available)	CPU: 0.71/0.94 core MEM: 0.35/0.69 GB	-	Configure alarm policy Add existing node More

Total items: 1

20
View cluster credential
Adjust cluster specification
Check spefication adjustment history
Create node
Upgrade Master Kubernetes version
Upgrade Kubernetes version of the node
Delete

3. In the **Delete cluster** window that appears, choose to retain or delete existing resources of the cluster as needed.



Delete clusters

Node resource

Resource type	Resource details	Termination method
Node	Learn more	☑ Terminate all pay-as-you-go nodes in the cluster
System disk Data disk	Refer to node details	☑ Directly terminate node system disk ☑ Directly terminate all data disks mounted on the node
Default security group	Worker node:sg-kn2cceox	The default security group of the cluster will be automatically deleted by default if they are not used by other nodes.

Applications

Resource type	Resource details	Termination method
Workload	9 workload(s)	All workloads in the cluster will be deleted along with the cluster.
Service Route (Ingress)	Check in cluster	If you delete a cluster, all routes, services and CLBs created by services will be terminated automatically.
Storage	Associated cloud disk:0	□ Upon cluster deletion, terminate the mounted cloud disks if they are not in used.

You're going to delete the cluster lll (cls-ggzh0dbw). Please note that the cluster cannot be recovered once deleted.
Upon deletion of the cluster, nodes, workloads, services and routes will be deleted at the same time. Then the associated security group, CLB and cloud disks will be automatically deleted if you opted in "Auto Deletion". However if the resources mentioned above are still used by other clusters or Tencent Cloud services, they will not be deleted. Please check the status of these resources in their consoles to avoid unnecessary costs.
When the cluster is terminated, the billing of cluster management fee stops.

□ I have read the notice above and confirmed to delete the cluster

4. Read the risks of the cluster deletion operation, and tick "I have read the notice above and confirmed to delete the cluster".
5. Click **OK** to delete the cluster.

Cluster Scaling

Last updated：2023-06-06 11:16:57

Overview

This document describes how to manually or automatically scale a cluster to meet the resource requirements of the applications. You can scale a cluster in one of the following ways:

- Manually adding/removing a node
- Automatically adding/removing a node via auto scaling
- Completing scaling of application layer via supernode (without scaling node)

Prerequisites

1. You have logged in to the TKE console.
2. You have created a cluster.

Directions

Manually adding/removing a node

To scale out a cluster, you can manually add a node by creating a node or adding an existing node. To scale in a cluster, you can remove a node.

Creating a node

When creating a node, you can configure a new CVM on the **Create Node** page for cluster scale-out.
For details, see Adding a Node.

Adding an existing node

> Note：
>
> - Currently, you can only add CVMs in the same VPC.
> - If you choose to add an existing node to the cluster, the operating system of the CVM will be reinstalled according to your settings.

- If you choose to add an existing node to the cluster, the project of the CVM will be migrated to the project set for the cluster.
- When adding a node with only one data disk to the cluster, you can choose to set the related parameters of data disk mounting.

When adding a node, you can select and configure the CVM you want to add to the cluster on the **Add Existing Node** page.
For details, see Adding a Node > Adding an Existing Node.

Removing a node

For directions on how to scale in a cluster, see Removing a Node.

Automatically adding/removing a node via auto scaling

Auto scaling relies on the community component Cluster Autoscaler (CA), which can dynamically adjust the number of nodes in a cluster to meet your resource requirements. For details on auto scaling, see Node Pool Overview.

Scaling out via supernode

Supernode is a kind of scheduling capability. It supports scheduling the Pods in a standard Kubernetes cluster to a supernode that does not occupy the cluster server resource to implement dynamic scaling out when resources are insufficient. For more information, see supernode Overview.

FAQs

For issues related to scaling, see Auto-scaling Related.

Changing the Cluster Operating System

Last updated：2023-05-22 16:05:37

Operating System Description

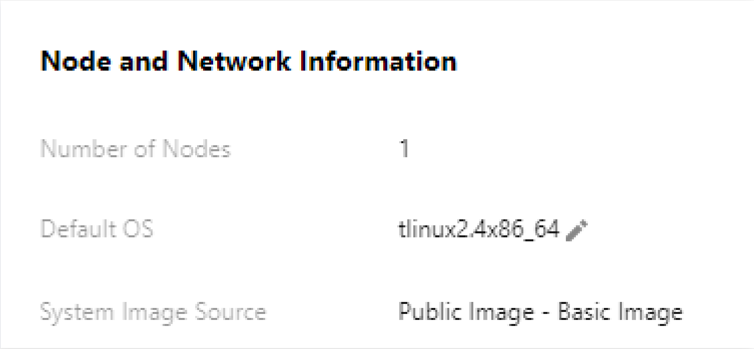
- Changes to the OS only affect new nodes and reinstalled nodes, not existing nodes.
- Nodes in the same cluster can use different OS versions without affecting cluster functionality.
- One script does not suit all operating systems. After configuring a node with a script, run a test to make sure it is compatible with the operating system.
- If you want to use custom images, submit a ticket for further assistance.

Operation Directions

Changing the default OS of a cluster

> Note：
> Before changing the default OS of a cluster, read Operating System Description carefully to learn about the relevant risks.

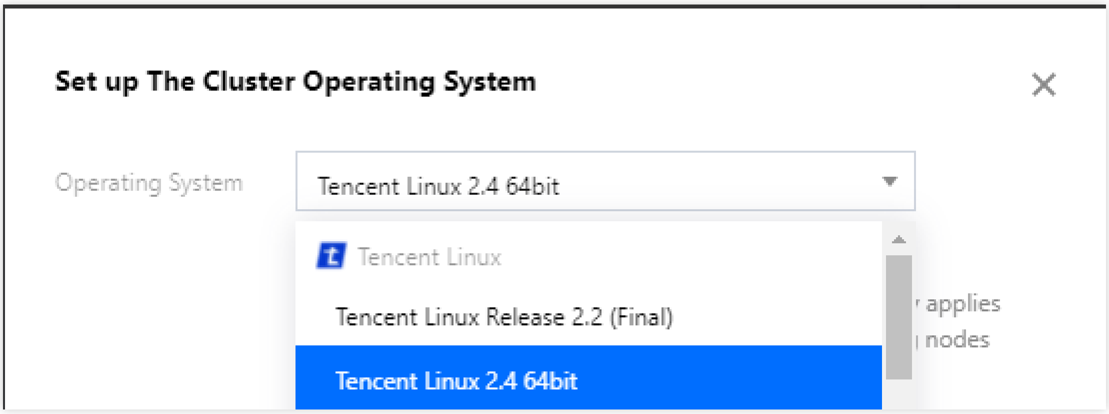
1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, select **View Cluster Credential** for the target cluster to go to the basic information page of the cluster.
3. In **Node and Network Information**, click ✎ next to **Default OS**. See the figure below:



Node and Network Information

Number of Nodes	1
Default OS	tlinux2.4x86_64 ✎
System Image Source	Public Image - Basic Image

4. On the **Set Cluster OS** page, change the OS. See the figure below:



Set up The Cluster Operating System

Operating System: Tencent Linux 2.4 64bit

5. Click **Submit**.

Connecting to a Cluster

Last updated：2023-04-17 15:21:26

Scenario

This document shows how to connect a local client to a TKE cluster by using kubectl, a Kubernetes command line tool.

Note

To improve access security and stability of TKE clusters, TKE plans to officially discontinue the domain name “cls-xxx.ccs.tencent-cloud.com” from August 10, 2022 (UTC +8).

TKE will upgrade the cluster access from July 20, 2022 (UTC +8). After the upgrade, a new architecture will be used when you enable public/private network access for a cluster. Please re-enable cluster access during July 20 to August 10 to migrate to the new architecture (the migration is required for existing clusters). For details, see Enabling Cluster Access and Obtaining Kubeconfig.

Differences between the old and new architectures are listed below. Cluster access APIs are compatible with the new architecture. You can migrate to the new architecture by calling the APIs. Please make adaption before **July 20, 2022 (UTC +8)** if you plan to call the APIs.

1. The public network resolution feature is not available under the new architecture. You can pass in a domain name, for which a security signature will be provided. To ensure access security, you must configure the public network resolution on your own.
2. When enabling public network access under the new architecture, you must set a security group to configure access control polices.
3. When enabling public network access under the new architecture, a public CLB is created under your account. For pricing of public CLB, see Billing for Bill-by-IP Accounts.

Solution 1. Connecting to a Cluster via Cloud Shell

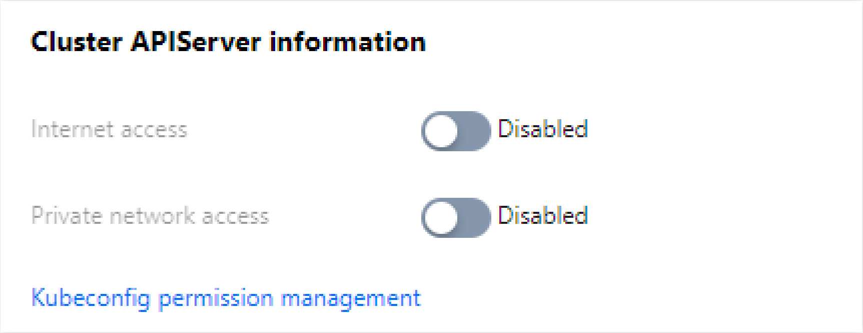
TKE integrates Tencent Cloud's Cloud Shell. You can implement the capability to quickly connect to a cluster in the Tencent Cloud console and use kubectl to flexibly manage clusters.

Directions

Step 1. Enable public network access

1. Log in to the TKE console and select Cluster in the left sidebar.
2. On the **Cluster Management** page, select the region where the target cluster resides and click the ID of the cluster to go to the cluster details page.

3. On the cluster basic information page, check whether cluster access is enabled.



Cluster APIServer information

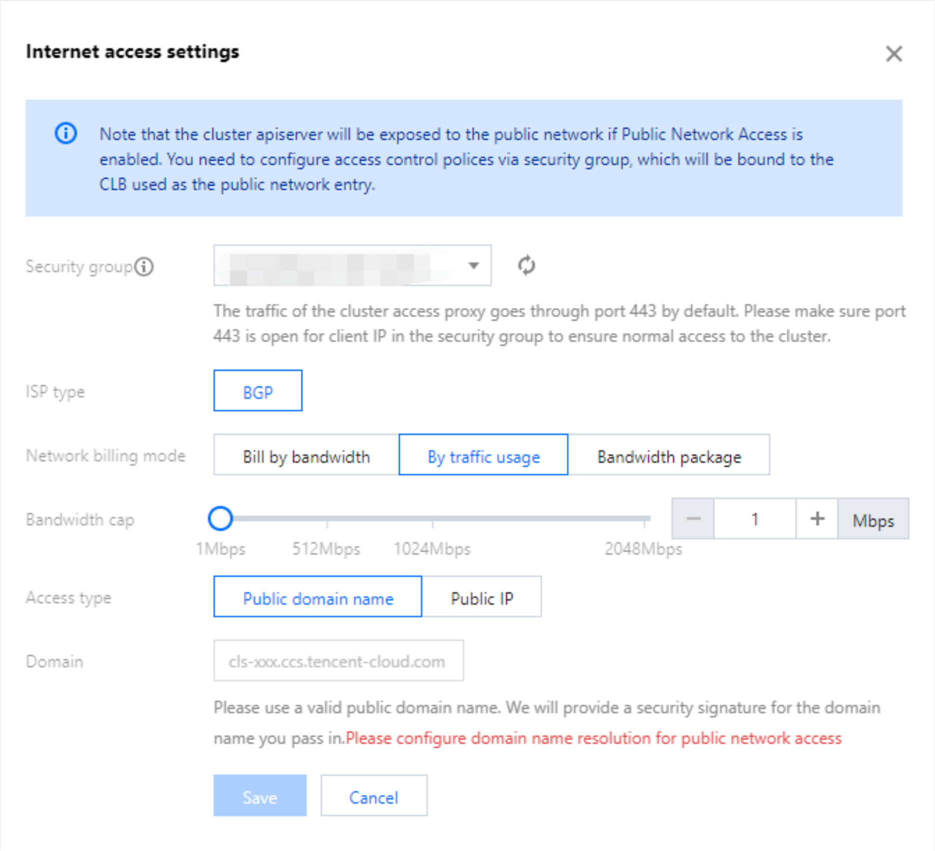
Internet access ○ Disabled

Private network access ○ Disabled

Kubeconfig permission management

4. Click

to enable public network access. You need to configure relevant parameters to enable public network access.



Internet access settings

ⓘ Note that the cluster apiserver will be exposed to the public network if Public Network Access is enabled. You need to configure access control polices via security group, which will be bound to the CLB used as the public network entry.

Security group ⓘ

The traffic of the cluster access proxy goes through port 443 by default. Please make sure port 443 is open for client IP in the security group to ensure normal access to the cluster.

ISP type: BGP

Network billing mode: Bill by bandwidth | By traffic usage | Bandwidth package

Bandwidth cap: 1Mbps 512Mbps 1024Mbps 2048Mbps − 1 + Mbps

Access type: Public domain name | Public IP

Domain: cls-xxx.ccs.tencent-cloud.com

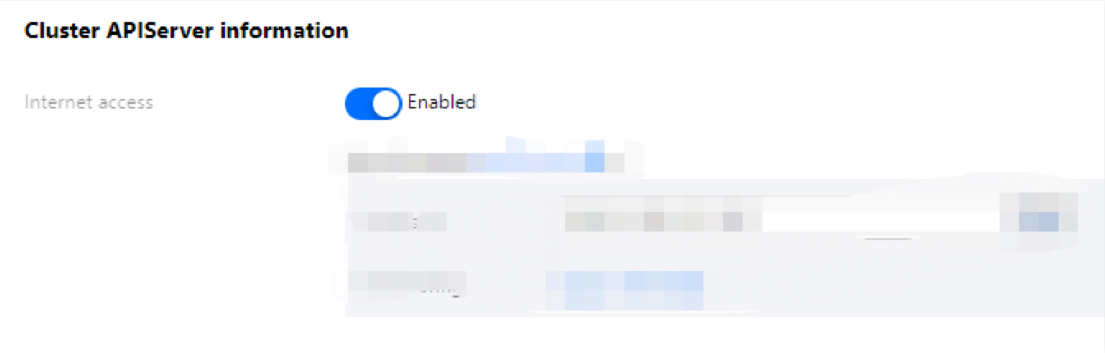
Please use a valid public domain name. We will provide a security signature for the domain name you pass in.Please configure domain name resolution for public network access

Security group: A public CLB is automatically assigned after the public network access is enabled. You can configure access control policies via a security group. We will bind the security group to the public CLB to control access.

ISP type, **Network billing mode**, **Bandwidth cap**: Configure these parameters based on your actual needs. For more information, see Creating CLB Instances.

Access type: If you select **Public domain name**, you need to pass in a custom domain name, for which we will provide a security signature. You must configure public network resolution on your own. If you select the default CLB domain name, you do not need to manually configure operations such as domain name resolution.

5. Check that public network access is enabled. See the figure below:



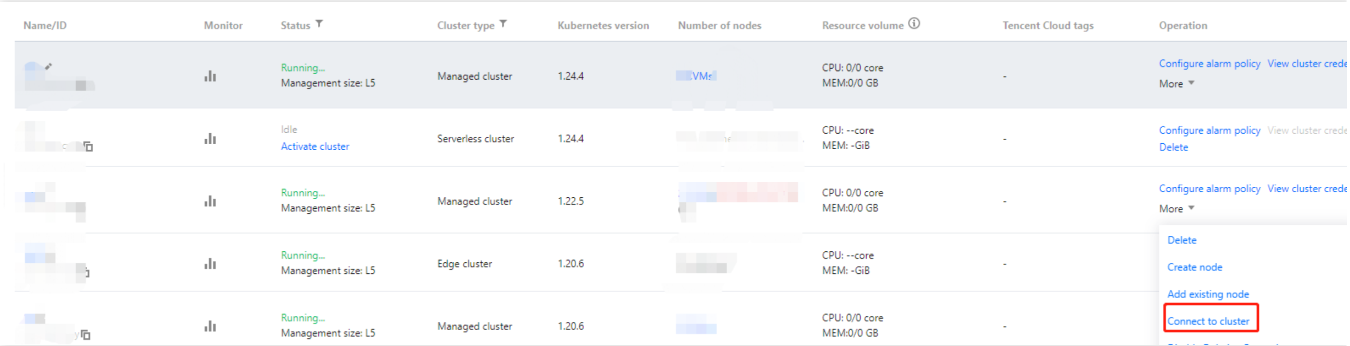
Cluster APIServer information

Internet access Enabled

Step 2. Use Cloud Shell to connect to the cluster

1. Log in to the TKE console and select Cluster in the left sidebar.

2. On the **Cluster Management** page, select the region where the target cluster resides, and select **More** > **Connect to a Cluster**. See the figure below:



Name/ID	Monitor	Status	Cluster type	Kubernetes version	Number of nodes	Resource volume	Tencent Cloud tags	Operation
		Running... Management size: L5	Managed cluster	1.24.4	VMs	CPU: 0/0 core MEM:0/0 GB	-	Configure alarm policy View cluster crede More
		Idle Activate cluster	Serverless cluster	1.24.4		CPU: --core MEM: -GiB	-	Configure alarm policy View cluster crede Delete
		Running... Management size: L5	Managed cluster	1.22.5		CPU: 0/0 core MEM:0/0 GB	-	Configure alarm policy View cluster crede More
		Running... Management size: L5	Edge cluster	1.20.6		CPU: --core MEM: -GiB	-	
		Running... Management size: L5	Managed cluster	1.20.6		CPU: 0/0 core MEM:0/0 GB	-	

3. The Cloud Shell entry is displayed at the lower part of the console. You can directly enter a kubectl command in the command box.

Solution 2. Connecting to a Cluster via a Local Computer

Prerequisites

Install curl.

Directions

Step 1. Install kubectl

1. Install kubectl as instructed in Installing and Setting up kubectl. You can select an appropriate way to obtain kubectl based on the OS type:

Note

If you have already installed kubectl, skip this step.

Replace "v1.18.4" in the command line with the kubectl version required by your application based on actual needs. The version of kubectl on the client must be consistent with the latest version of Kubernetes on the service end. You can check the Kubernetes version on the **Cluster information** section of the **Basic information** page.

macOS X

Linux

Windows

Run the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/darw
```

Run the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/linu
```

Run the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/wind
```

2. Here we take Linux as an example. Run the following command to grant permissions to use kubectl.

```
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
```

3. Run the following command to verify whether the installation is successful.

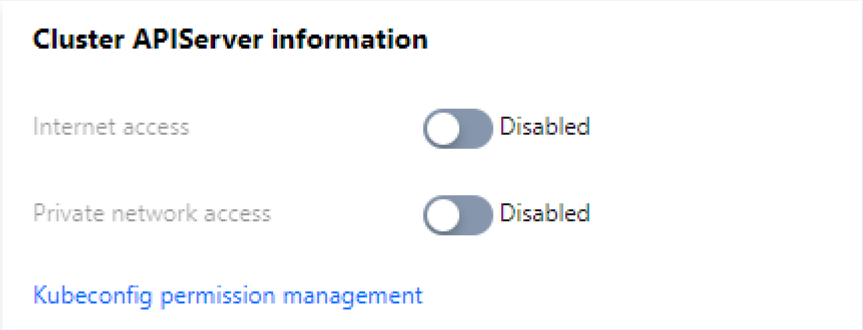
```
kubectl version
```

If the output shows the version information, the installation is successful.

```
Client Version: version.Info{Major:"1", Minor:"5", GitVersion:"v1.5.2", GitCommit:"
```

Step 2. Enable cluster access

1. Log in to the TKE console and select Cluster in the left sidebar.
2. On the **Cluster Management** page, select the region where the target cluster resides and click the ID/name of the cluster to go to the cluster details page.
3. On the cluster basic information page, check whether cluster access is enabled.



Cluster APIServer information

Internet access: Disabled

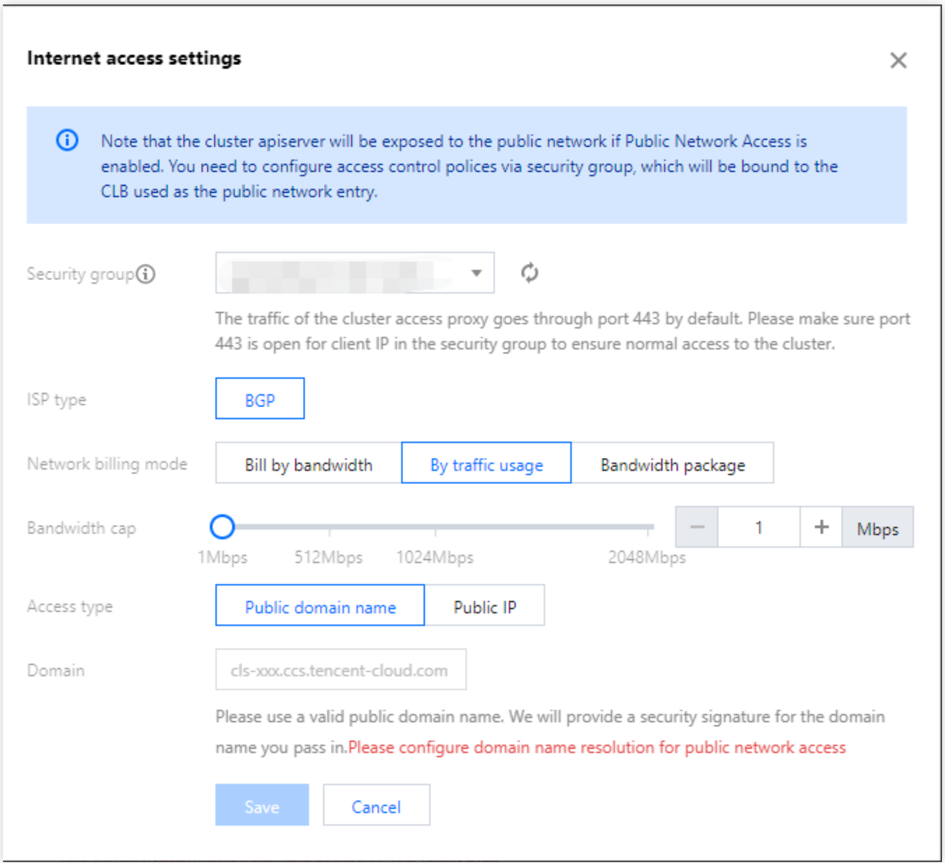
Private network access: Disabled

Kubeconfig permission management

Enabling public network access

Enabling private network access

You need to configure relevant parameters to enable the public network access.



Internet access settings

Note that the cluster apiserver will be exposed to the public network if Public Network Access is enabled. You need to configure access control polices via security group, which will be bound to the CLB used as the public network entry.

Security group: [redacted]

The traffic of the cluster access proxy goes through port 443 by default. Please make sure port 443 is open for client IP in the security group to ensure normal access to the cluster.

ISP type: BGP

Network billing mode: Bill by bandwidth | By traffic usage | Bandwidth package

Bandwidth cap: 1 Mbps

Access type: Public domain name | Public IP

Domain: cls-xxx.ccs.tencent-cloud.com

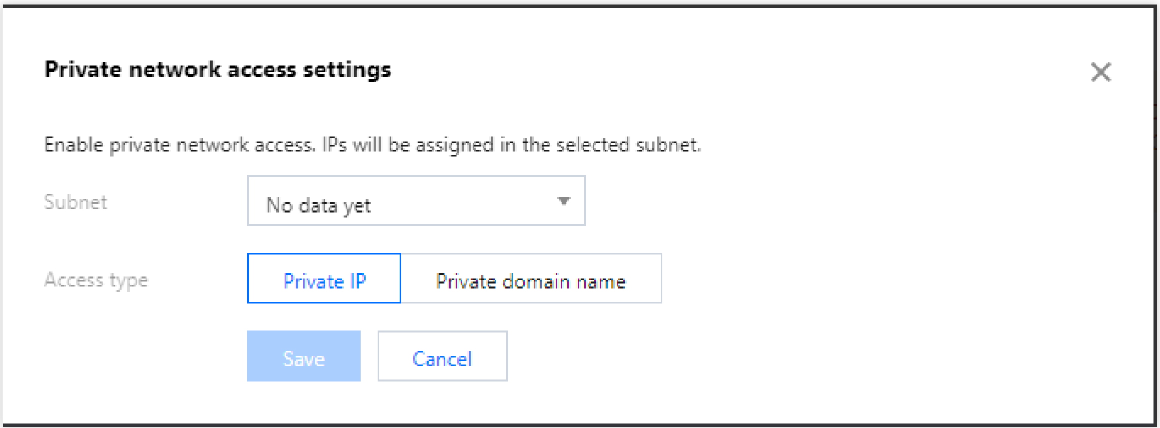
Please use a valid public domain name. We will provide a security signature for the domain name you pass in.Please configure domain name resolution for public network access

Security group: A public CLB is automatically assigned after the public network access is enabled. You can configure access control policies via a security group. We will bind the security group to the public CLB to control access.

ISP type, **Network billing mode**, **Bandwidth cap**: Configure these parameters based on your actual needs. For more information, see Creating CLB Instances.

Access type: If you select **Public domain name**, you need to pass in a custom domain name, for which we will provide a security signature. You must configure public network resolution on your own. If you select the default CLB domain name, you do not need to manually configure operations such as domain name resolution.

You need to configure relevant parameters to enable private network access.



Private network access settings

Enable private network access. IPs will be assigned in the selected subnet.

Subnet: No data yet

Subnet: It is disabled by default. To enable the private network access, you need to configure a subnet. IP addresses are assigned from the configured subnet after the private network access is successfully enabled.

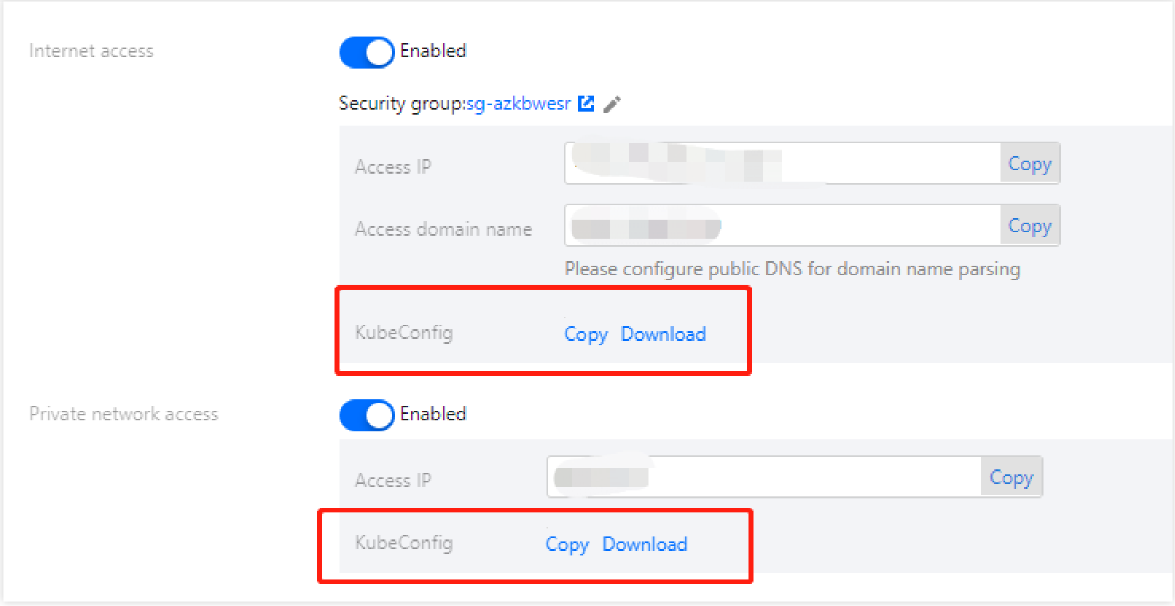
Access type: If **Private domain name** is selected, you need to pass in a custom domain name, for which we will provide a security signature. You must configure private network resolution on your own. If **Private IP** is selected, we will assign a private IP and provide a security signature for it.

Using the service IP address of Kubernetes: On the cluster details page, select **Service and route > Service** in the left sidebar to obtain the service IP address of Kubernetes in the default namespace. Replace the clusters.cluster.server field in the Kubeconfig file with https://< `IP` >:443. **Note**: Kubernetes service is under the ClusterIP mode and is only applicable to access within the cluster.

Step 3. Obtain kubeconfig

TKE provides two types of KubeConfig for use in public network access and private network access, respectively. After the cluster access is enabled, you can follow the steps below to obtain the corresponding Kubeconfig:

1. Check “Cluster APIServer information” on the “Basic information” page of the cluster.
2. Copy or download **Kubeconfig** under the corresponding access type, or check the security group, access domain name (configured when the access is enabled) and access IP for the public network access.



Internet access: Enabled

Security group:sg-azkbwesr

Access IP: [redacted] Copy

Access domain name: [redacted] Copy

Please configure public DNS for domain name parsing

KubeConfig: Copy Download

Private network access: Enabled

Access IP: [redacted] Copy

KubeConfig: Copy Download

Step 4. Configure kubeconfig and access the Kubernetes cluster

1. Configure the cluster credential as needed.

Before configuration, check whether the access credential for any cluster has been configured on the current client.

If no, the `~/.kube/config` file is empty, and you can copy the obtained kubeconfig access credential to the file. If the `~/.kube/config` file does not exist on the client, you can create one.

If yes, you can download the obtained kubeconfig to a specified location and run the following commands in sequence to merge the config files of multiple clusters.

```
KUBECONFIG=~/.kube/config:~/Downloads/cls-3jju4zdc-config kubectl config view --mer
```

```
export KUBECONFIG=~/.kube/config
```

where, `~/Downloads/cls-3jju4zdc-config` is the kubeconfig file path of the current cluster. Replace it with the actual local path of the file.

2. After kubeconfig is completed, run the following commands in sequence to view the contexts and switch contexts to access the cluster.

```
kubectl config get-contexts
kubectl config use-context cls-3jju4zdc-context-default
```

3. Run the following command to check whether the cluster can be accessed.

```
kubectl get node
```

If you cannot connect to the cluster, check whether public network access or private network access is enabled, and ensure that the access client is in the specified network environment.

Notes

Introduction to the kubectl CLI

Kubectl is a CLI tool for performing operations on Kubernetes clusters. This document covers the `kubectl` syntax, common command operations, and examples. For more information on each command (including all main commands and sub-commands), see the kubectl reference document or run the `kubectl help` command to view help information.

Upgrading a Cluster

Last updated：2023-05-25 15:52:18

Overview

Tencent Cloud TKE allows you to upgrade the Kubernetes version. You can use this feature to upgrade a running Kubernetes cluster. The upgrade process includes pre-upgrade checking, upgrading the Master, and upgrading the node.

Upgrade Notice

- **The upgrading action is irreversible. Perform this operation with caution.**

- Before upgrading the cluster, check whether the cluster is healthy. If the cluster is abnormal, you can fix it yourself or consult online.

- **Upgrade sequence**: when upgrading a cluster, you must first upgrade the Master, and then upgrade the node as quickly as possible. During the upgrade process, we recommend that you do not perform any operations in the cluster.

- **Only upgrading to the next Kubernetes version offered by TKE is supported.** Upgrading across multiple versions (such as upgrading from 1.8 to 1.12, skipping 1.10) is not supported. You can upgrade to the next version only if the Master and node versions are consistent.

- **Incompatibility of CSI-CFS add-on**: as for the CSI add-ons: COS CSI and CFS CSI, the CSI add-on versions adapted to different Kubernetes versions have the following differences. Therefore, we recommend that you reinstall the CSI add-on in add-on management page when you upgrade the cluster to TKE 1.14 and later version. The rebuilding of the add-on does not affect COS and CFS storage already in use.
  - The version of the CSI add-on adapted for Kubernetes 1.10 and Kubernetes 1.12 is **0.3**.
  - The CSI add-on version for Kubernetes 1.14 and later is **1.0**.

- **The failure of HPA**: before Kubernetes 1.18, the apiversion of the deployment object referenced in HPA may be `extensions/v1beta1` , but after Kubernetes 1.18, the apiversion of deployment is only apps/v1, which may cause the failure of HPA after the cluster is upgraded to Kubernetes 1.18.

If you use the HPA feature, we recommend that you run the following command to switch the apiVersion in the HPA object to `apps/v1` before upgrading.

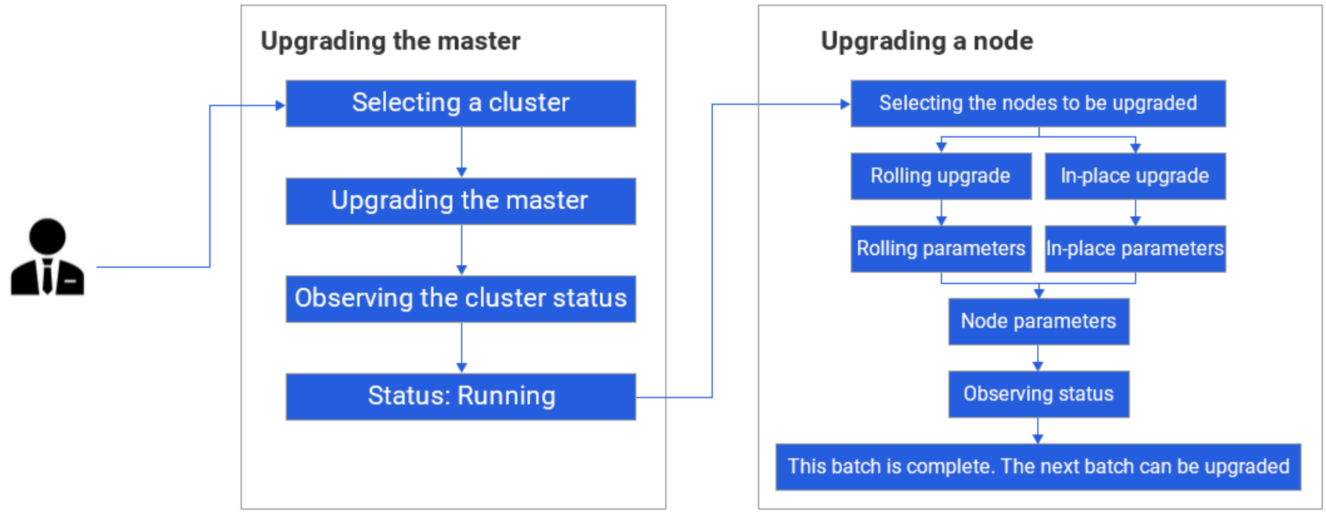
```
kubectl patch hpa test -p '{"spec":{"scaleTargetRef":{"apiVersion":"apps/v1"
}}}'
```

- **The failure of Helm applications**: each application, including those installed through the application marketplace or third parties, supports different versions of Kubernetes. Before upgrading a cluster, you are advised to view the list of applications installed in the cluster and check the range of cluster versions supported by the applications. Some applications are adaptable to higher versions of Kubernetes, and you may need to upgrade them. Some applications may not be adaptable to higher versions of Kubernetes, and in which case, upgrade the cluster with caution.

- **Nginx-ingress version issue**: extensions/v1beta1 and networking.k8s.io/v1beta1 ingress APIs are no longer provided in v1.22. For more information, see here. If the Nginx-ingress version in your cluster is too early, upgrade the Nginx-ingress add-on on the add-on management page after you upgrade Kubernetes to v1.22 or later.

How It Works

Upgrading a cluster involves two steps. The first step is to upgrade the Master Kubernetes version and the second is to upgrade the node Kubernetes version. See below for details:



Upgrading the Master Kubernetes version

> Note：
> **Currently, Master version upgrades of managed clusters and self-deployed clusters are supported**, and the upgrade takes 5-10 minutes, during which you are unable to operate your cluster.

Notes for Master major version and minor version upgrade

Currently, the upgrade of Master supports the **major version upgrade** (for example, from 1.14 to 1.16), and the **minor version upgrade** (for example, from 1.14.3 to 1.14.6, or from v1.18.4-tke.5 to v1.18.4-tke.6). We strongly recommend that you check the corresponding feature release records before upgrading:

- Before upgrading the major version of kubernetes, we recommend that you check the Update Notes of TKE Kubernetes Major Versions.
- Before upgrading the minor version of kubernetes, we recommend that you check the TKE Kubernetes Revision Version History.

> Note
> - For major version upgrades (for example, from 1.12 to 1.14), the original custom parameters will not be retained, you need to reconfigure them for the new version. For more information, see Custom Kubernetes Component Launch Parameters.
> - For minor version upgrades, the custom parameters will be retained, and you do not need to reconfigure them.

Points for attention

- **Before upgrading, read the Upgrade Notice.**
- For TKE clusters of the v1.7.8, the network mode is bridge. Upgrading the cluster does not automatically switch the network mode to cni.
- Upgrading the cluster does not switch kube-dns to core-dns.
- When you upgrade a cluster to v1.10 and 1.12, some features configured when the cluster is created, such as support for ipvs, will become unavailable.
- After the upgrade of an existing cluster, if the master version is 1.10 or later, and the node version is V1.8 or earlier, the PVC feature will be unavailable.
- After upgrading the master version, we recommend that you upgrade the node version as soon as possible.

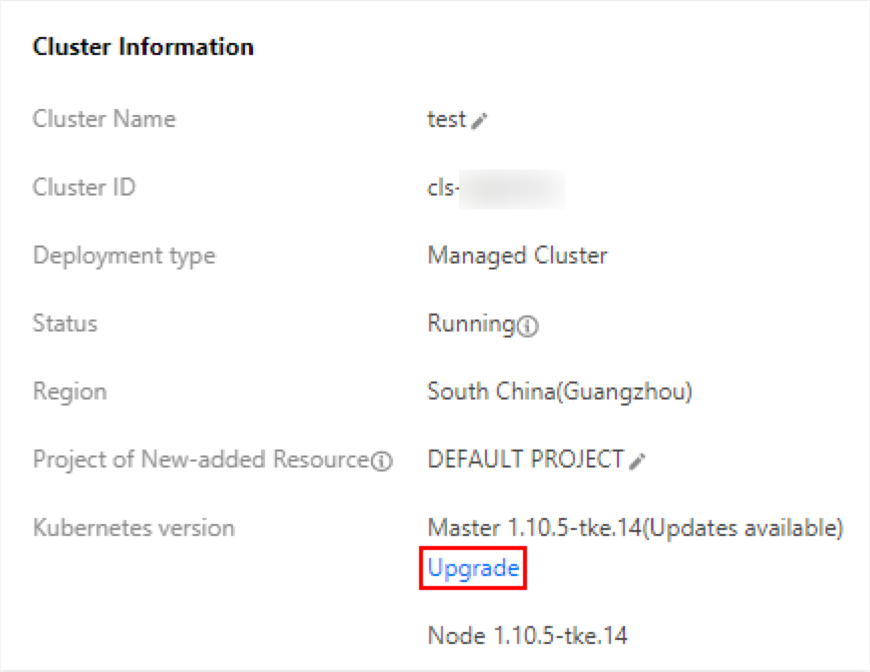
Technical principles of Master upgrade

The master node upgrade involves 3 steps: pre-dependent component upgrade, Master node component upgrade, and post-dependent component upgrade.

- **Pre-dependent component upgrade**: the pre-dependent components, such as monitoring components, will be upgraded to prevent component exceptions due to compatibility problems.
- **Master node component upgrade**: all corresponding components of Masters will be upgraded in sequence. When the previous component is upgraded successfully, the upgrade of next component starts.
  TKE will first upgrade kube-apiserver, then kube-controller-manager and kube-scheduler, and finally kubelet. The specific steps are as follows:
  - Regenerate the yaml file corresponding to the static Pod of the kube-apiserver component.
  - Check whether the current kube-apiserver Pod is healthy and whether the kubernetes version is normal.
  - Similarly, upgrade kube-controller-manager and kube-scheduler in sequence.
  - Upgrade kubelet and check whether the master node is ready.
- **Post-dependent component upgrade**:
  - Upgrade the post-dependent components as needed, such as kube-proxy (and change its rolling update strategy to on delete), and cluster-autoscaler components.
  - Perform some compatibility operations related to post-dependent components to prevent component exceptions due to compatibility problems.

Master upgrade

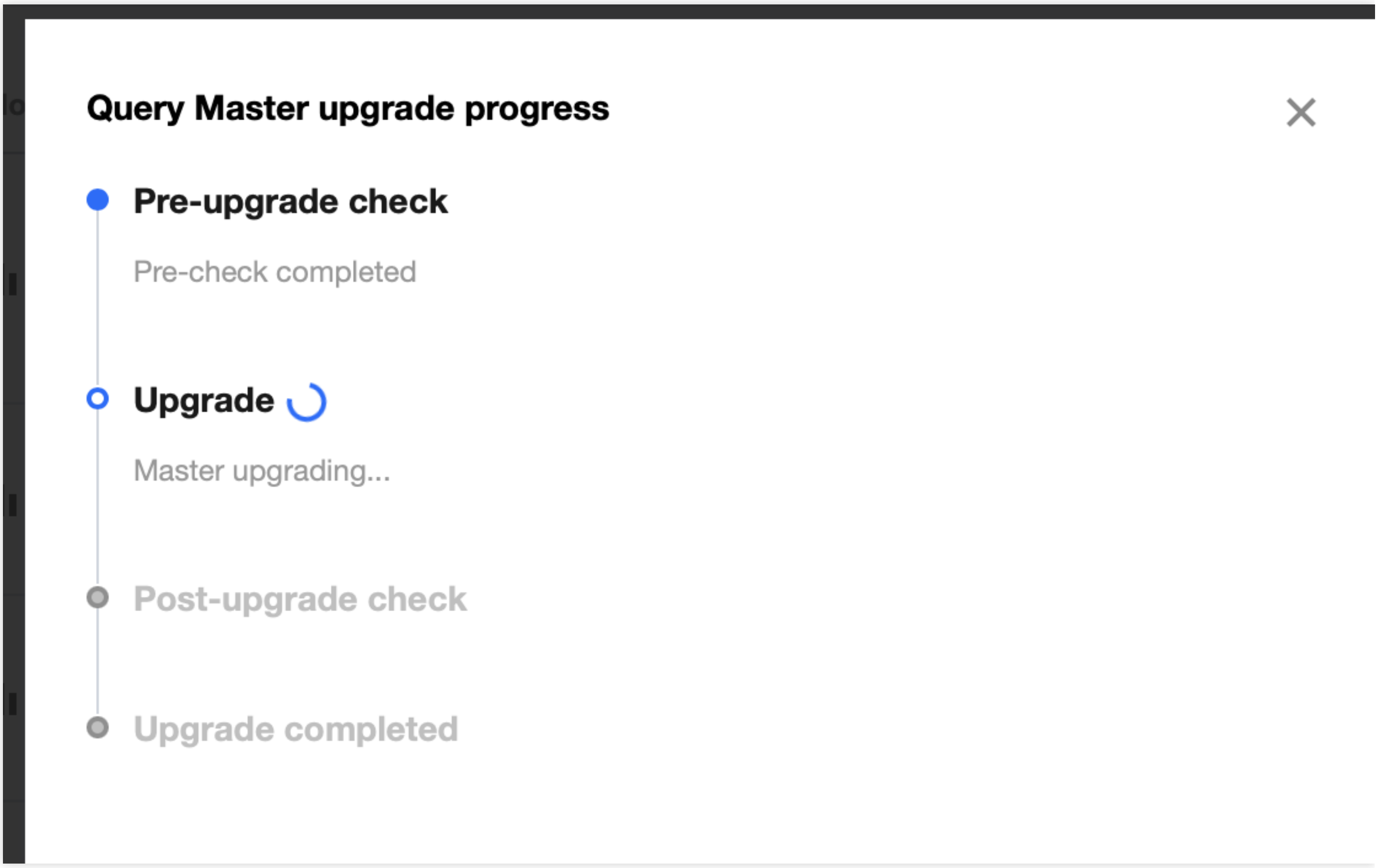
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, select the ID of the desired cluster, and enter the cluster details page.
3. On the cluster details page, click **Basic Information** on the left.
4. In the cluster information module on the cluster's **Basic Information** page, click **Upgrade** to the right of the Master version, as shown in the figure below:



Cluster Information

Cluster Name	test
Cluster ID	cls-[redacted]
Deployment type	Managed Cluster
Status	Running
Region	South China(Guangzhou)
Project of New-added Resource	DEFAULT PROJECT
Kubernetes version	Master 1.10.5-tke.14(Updates available) Upgrade Node 1.10.5-tke.14

5. In the pop-up window, click **Submit** and wait until the upgrade is complete.
6. You can view the upgrade progress in the cluster status of the cluster management page, or you can view the current upgrade progress, master node upgrade progress (the managed cluster does not display the specific Master node list), and the upgrade duration in the upgrade progress pop-up window, as shown below:



Query Master upgrade progress

Pre-upgrade check

Pre-check completed

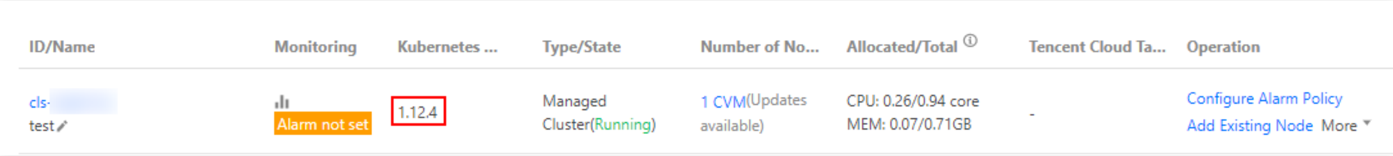
Upgrade

Master upgrading...

Post-upgrade check

Upgrade completed

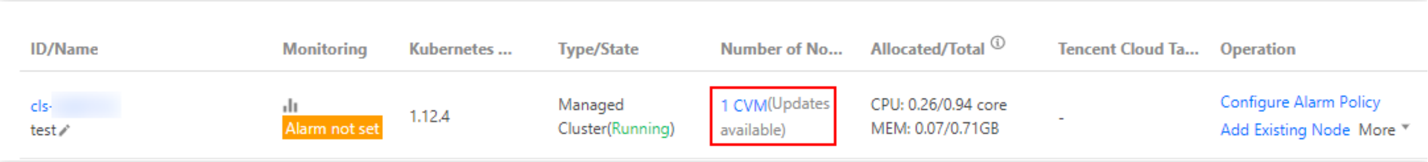
7. In this example, the original master version of the cluster is 1.10.5. After the upgrade, the Master version is 1.12.4. This is shown in the following figure:



ID/Name	Monitoring	Kubernetes ...	Type/State	Number of No...	Allocated/Total ⓘ	Tencent Cloud Ta...	Operation
cls-[redacted] test	Alarm not set	1.12.4	Managed Cluster(Running)	1 CVM(Updates available)	CPU: 0.26/0.94 core MEM: 0.07/0.71GB	-	Configure Alarm Policy Add Existing Node More

Upgrading the node Kubernetes version

After the cluster’s Master Kubernetes version is upgraded, the cluster list page will show that an upgrade is available for the cluster node, as shown below:



ID/Name	Monitoring	Kubernetes ...	Type/State	Number of No...	Allocated/Total ⓘ	Tencent Cloud Ta...	Operation
cls-[redacted] test	Alarm not set	1.12.4	Managed Cluster(Running)	1 CVM(Updates available)	CPU: 0.26/0.94 core MEM: 0.07/0.71GB	-	Configure Alarm Policy Add Existing Node More

Points for attention

- **Before upgrading, read the Upgrade Notice.**
- You can upgrade the node when it’s running.

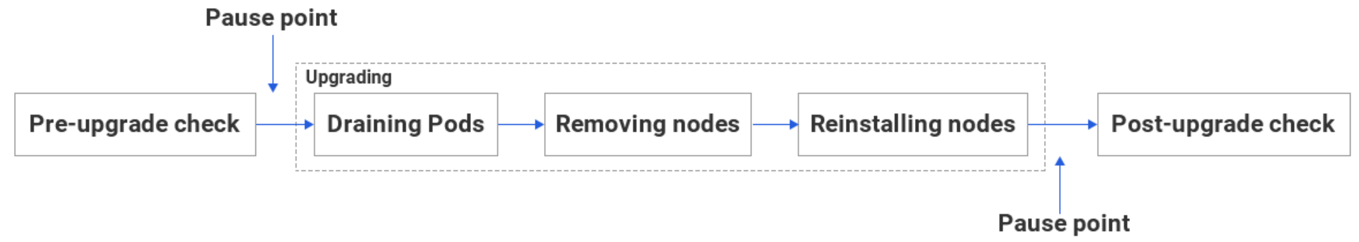
Selecting an upgrade method

You can upgrade the node Kubernetes version in two ways: reinstall and rolling upgrade and in-place rolling upgrade.

- **Reinstall and rolling upgrade**: reinstall the node to upgrade the node version. This method only supports major version upgrades, such as upgrades from version 1.10 to version 1.12.
- **In-place rolling upgrade**: upgrade directly without re-installation. This only replaces components such as Kubelet and kube-proxy. Currently, this method supports both major and minor upgrades, such as from version 1.10 to version 1.12, or from version 1.14.3 to version 1.14.8.

Principles of reinstall and rolling upgrade

Rolling upgrade based on the reinstalled node. Only one node is upgraded at a time. When the previous node is upgraded successfully, the upgrade of next node starts. See below:



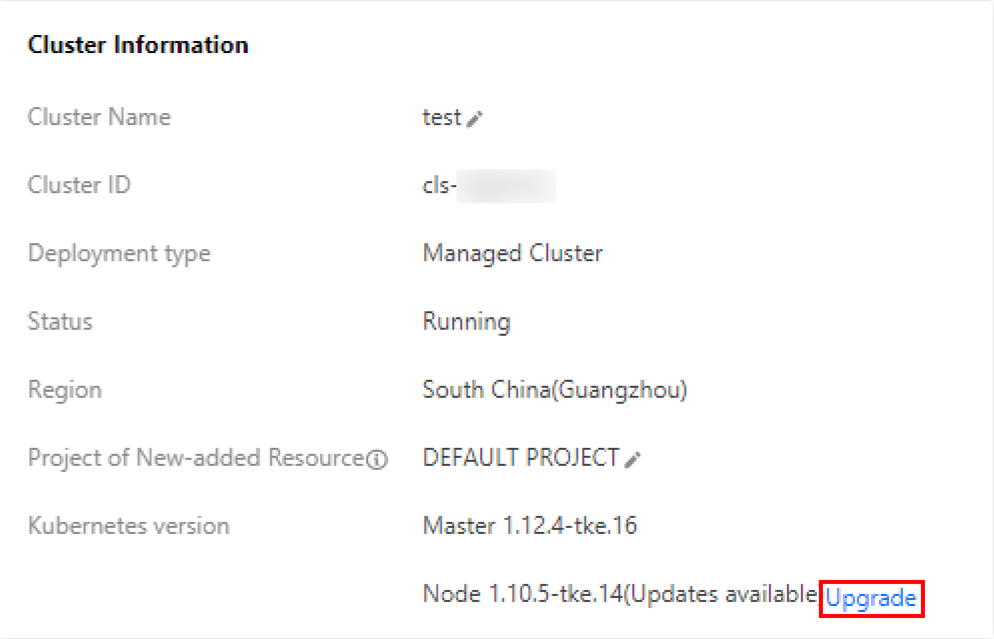
- **Pre-upgrade checking**: check the Pods on a node before draining. The specific items for pre-upgrade checking are as follows:
  - Calculate the number of pods of all workloads in this node. If the Pod number of any workload changes to 0 after the node is drained, then the check fails, and the upgrade cannot be performed.
  - The following system control plane workloads will be ignored:
    - l7-lb-controller
    - cbs-provisioner
    - hpa-metrics-server
    - service-controller

  - cluster-autoscaler

- **Evicting Pods**: first mark the node as unschedulable. Then, evict or delete all Pods from the node.
- **Removing nodes**: remove the node from the cluster. This step only performs basic cleanup, and will not delete the node instance of the node in the cluster. Therefore, the node's attributes such as label and taint are retained.
- **Reinstalling nodes**: reinstall the node's operating system and reinstall the new version of kubelet.
- **Post-upgrade checking**: check whether the node is ready and schedulable, and check whether the current proportion of unavailable Pods exceeds the max limit.

Reinstall and rolling upgrade (node Kubernetes version)

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, select the ID of the cluster for node Kubernetes version upgrade to enter the cluster details page.
3. In the cluster information module, click **Upgrade** to the right of the Node Kubernetes version, as shown in the figure below:

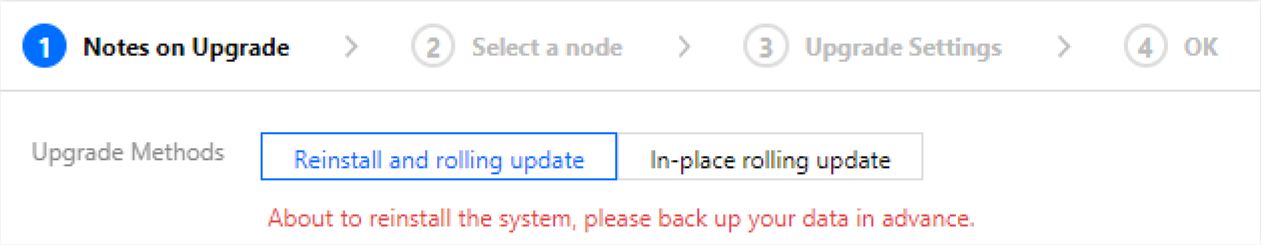


Cluster Information

Cluster Name	test
Cluster ID	cls-[redacted]
Deployment type	Managed Cluster
Status	Running
Region	South China(Guangzhou)
Project of New-added Resource	DEFAULT PROJECT
Kubernetes version	Master 1.12.4-tke.16
	Node 1.10.5-tke.14(Updates available Upgrade

4. In the **Notes on Upgrade** step, select **Reinstall and rolling upgrade** as the upgrade method. Read the upgrade notice carefully, check the checkbox of **I have read and agree to the terms above**, and then click **Next**.

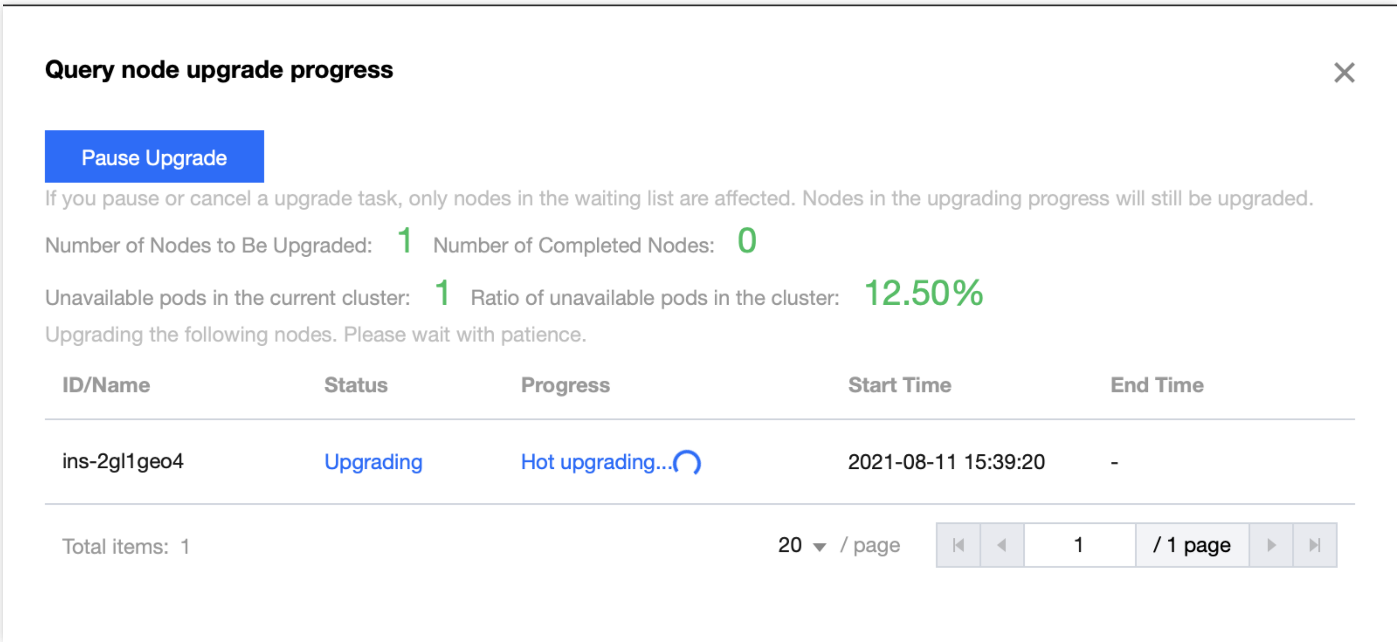
> Note：
> This upgrade method will reinstall the operating system, and the original data will be cleared. Back up the data in advance.



Upgrade Methods: Reinstall and rolling update

About to reinstall the system, please back up your data in advance.

5. In the **Select nodes** step, select the nodes to be upgraded, and click **Next**.
6. In the **Upgrade Settings** step, enter the node information as required, and click **Next**.
7. In the **Confirmation** step, confirm the information and click **Finish** to start the upgrade.
8. View the progress of the node upgrade until all the nodes are upgraded.



If you pause or cancel a upgrade task, only nodes in the waiting list are affected. Nodes in the upgrading progress will still be upgraded.

Number of Nodes to Be Upgraded: 1 Number of Completed Nodes: 0

Unavailable pods in the current cluster: 1 Ratio of unavailable pods in the cluster: 12.50%

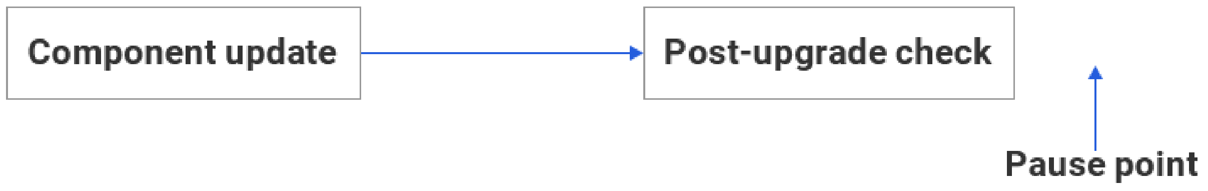
Upgrading the following nodes. Please wait with patience.

ID/Name	Status	Progress	Start Time	End Time
ins-2gl1geo4	Upgrading	Hot upgrading...	2021-08-11 15:39:20	-

Total items: 1

Principles of in-place rolling upgrade

Node in-place upgrade uses the rolling upgrade method, meaning that it will only upgrade one node at a time, with the next node not being upgraded until the current node upgrade is successful. In-place upgrade currently supports the upgrades of major version and different minor versions of the major version. This is shown in the following figure:

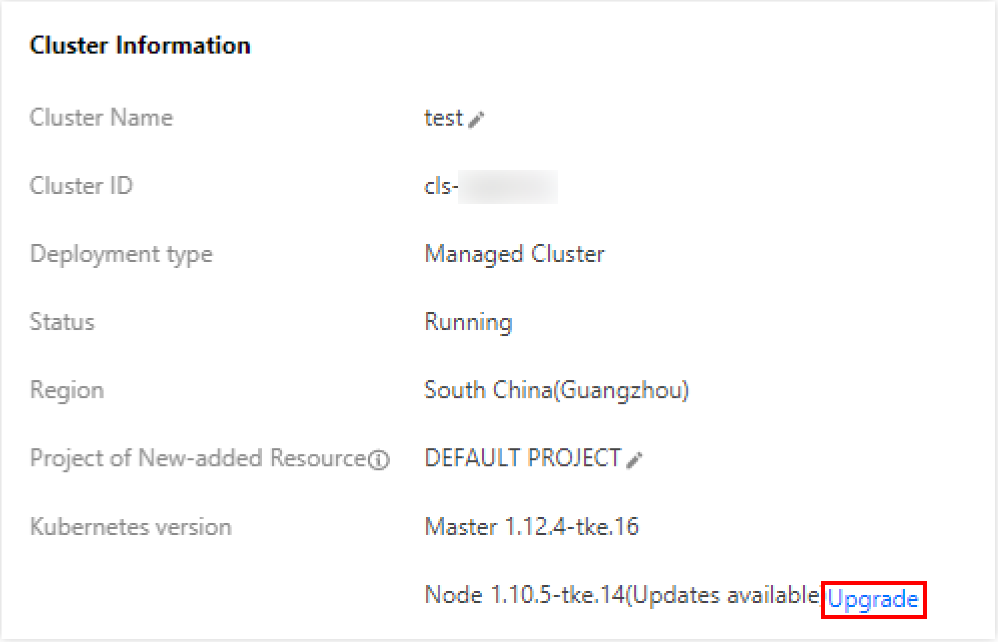


The steps are described as follows:

- **Component updating**: replace and restart the kubelet and kube-proxy components on the node.
- **Post-upgrade checking**: check whether the node is ready and check whether the proportion of currently unavailable Pods exceeds the max limit.

In-place rolling upgrade

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, select the ID of the desired cluster and enter the cluster details page.
3. In the cluster information module, click **Upgrade** to the right of the Node Kubernetes version, as shown in the figure below:

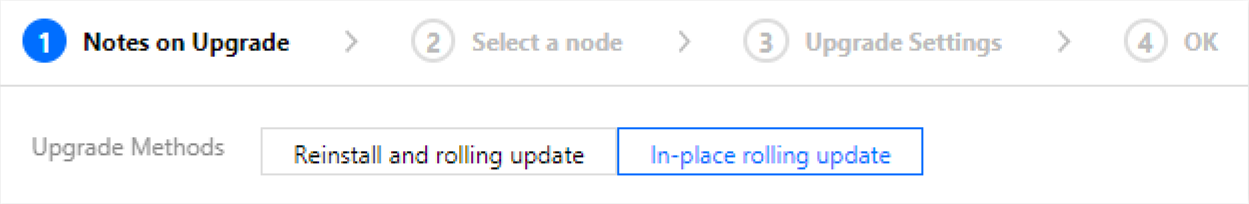


Cluster Information

Cluster Name	test
Cluster ID	cls-[redacted]
Deployment type	Managed Cluster
Status	Running
Region	South China(Guangzhou)
Project of New-added Resource	DEFAULT PROJECT
Kubernetes version	Master 1.12.4-tke.16
	Node 1.10.5-tke.14(Updates available Upgrade

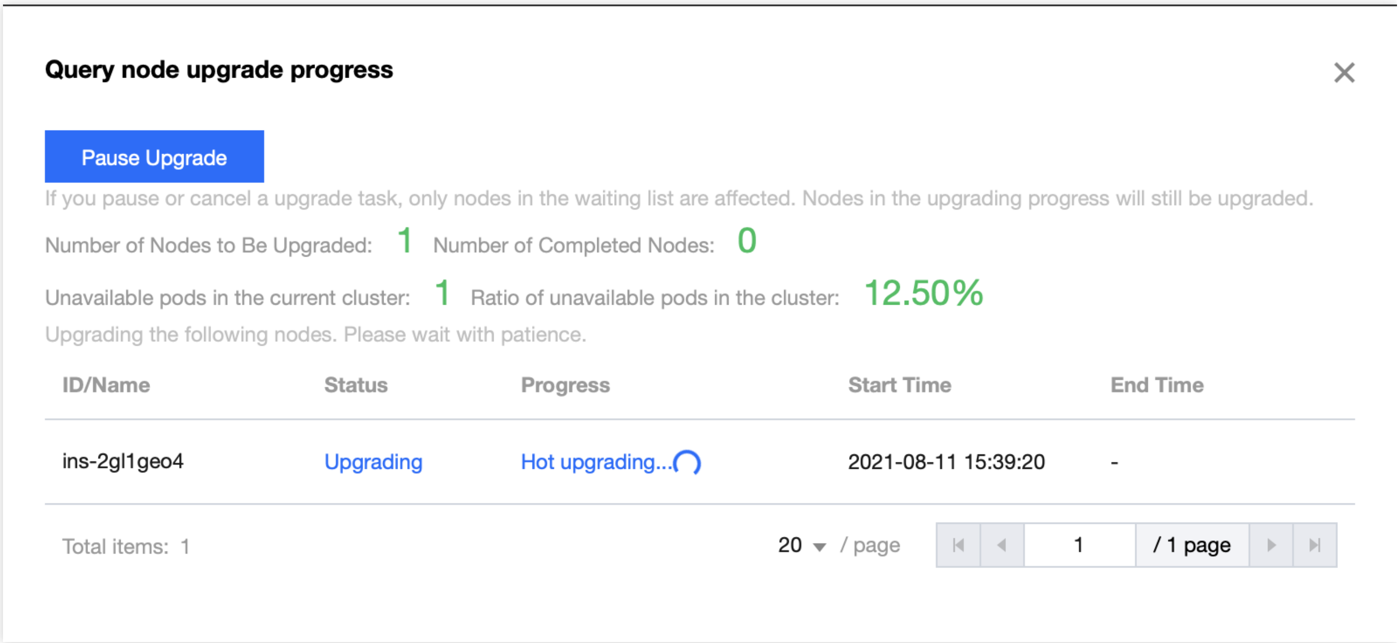
4. In the **Notes on Upgrade** step, select **In-place rolling upgrade** as the upgrade method. Read the upgrade notice carefully, place a check mark next to **I have read and agree to the terms above**, and then click **Next**, as

shown in the figure below:



Upgrade Methods: Reinstall and rolling update [✓ In-place rolling update]

5. In the **Select nodes** step, select the nodes to be upgraded, and click **Next**.
6. In the **Confirmation** step, confirm the information and click **Finish** to start the upgrade.
7. View the progress of the node upgrade until all the nodes are upgraded.



Query node upgrade progress

If you pause or cancel a upgrade task, only nodes in the waiting list are affected. Nodes in the upgrading progress will still be upgraded.

Number of Nodes to Be Upgraded: 1 Number of Completed Nodes: 0

Unavailable pods in the current cluster: 1 Ratio of unavailable pods in the cluster: 12.50%

Upgrading the following nodes. Please wait with patience.

ID/Name	Status	Progress	Start Time	End Time
ins-2gl1geo4	Upgrading	Hot upgrading...	2021-08-11 15:39:20	-

Total items: 1 20 / page 1 / 1 page

Enabling IPVS for a Cluster

Last updated：2022-01-13 16:23:38

Operation Scenario

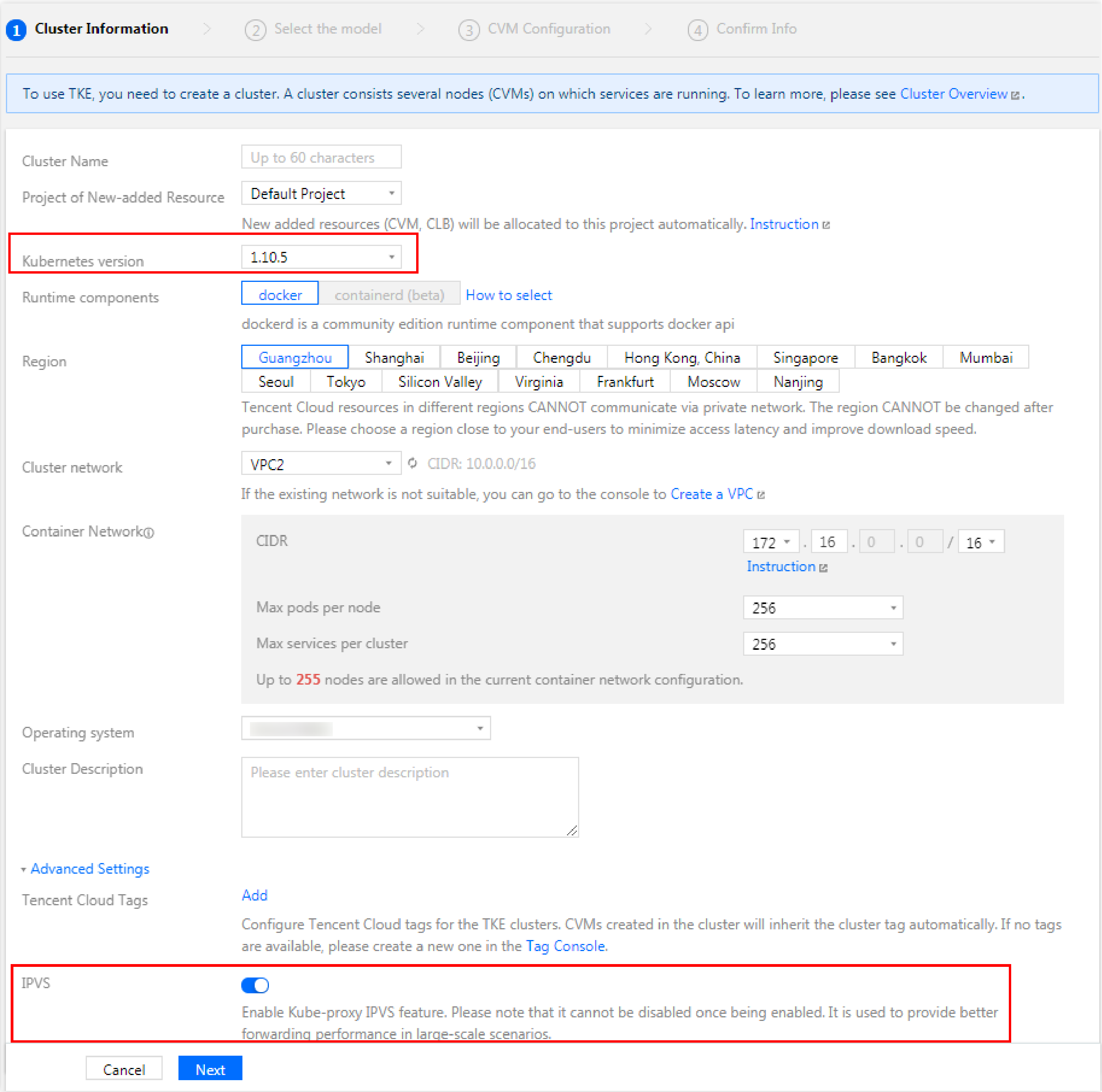
By default, Kube-proxy uses iptables to balance the load between Service and Pod. TKE supports fast enabling of IPVS-based traffic distributing and load balancing. IPVS is suitable for large-scale clusters by providing better scalability and performance.

Precautions

- This feature can be enabled only when the cluster is created but not for an existing cluster.
- After enabling, IPVS takes effect for the entire cluster. It is recommended not to manually modify the IPVS in the cluster or use it together with iptables.
- IPVS cannot be disabled once enabled in the cluster.
- IPVS is only available for TKE clusters running Kubernetes v1.10 or higher.

Steps

1. Log in to the TKE console.
2. Follow the steps in Creating a Cluster.On the "Create a cluster" page, set the "Kubernetes version" to v1.10 or higher, click **Advanced settings**, and enable "IPVS support". See the figure below:



To use TKE, you need to create a cluster. A cluster consists several nodes (CVMs) on which services are running. To learn more, please see Cluster Overview.

Cluster Name: Up to 60 characters

Project of New-added Resource: Default Project

New added resources (CVM, CLB) will be allocated to this project automatically. Instruction

Kubernetes version: 1.10.5

Runtime components: docker | containerd (beta) | How to select

dockerd is a community edition runtime component that supports docker api

Region: Guangzhou | Shanghai | Beijing | Chengdu | Hong Kong, China | Singapore | Bangkok | Mumbai | Seoul | Tokyo | Silicon Valley | Virginia | Frankfurt | Moscow | Nanjing

Tencent Cloud resources in different regions CANNOT communicate via private network. The region CANNOT be changed after purchase. Please choose a region close to your end-users to minimize access latency and improve download speed.

Cluster network: VPC2 CIDR: 10.0.0.0/16

If the existing network is not suitable, you can go to the console to Create a VPC

Container Network

CIDR: 172 . 16 . 0 . 0 / 16

Instruction

Max pods per node: 256

Max services per cluster: 256

Up to 255 nodes are allowed in the current container network configuration.

Operating system:

Cluster Description: Please enter cluster description

Advanced Settings

Tencent Cloud Tags: Add

Configure Tencent Cloud tags for the TKE clusters. CVMs created in the cluster will inherit the cluster tag automatically. If no tags are available, please create a new one in the Tag Console.

IPVS: ●

Enable Kube-proxy IPVS feature. Please note that it cannot be disabled once being enabled. It is used to provide better forwarding performance in large-scale scenarios.

3. Follow the on-screen prompts to complete the cluster creation.

Enabling GPU Scheduling for a Cluster

Last updated：2020-04-26 19:12:27

Scenario

If your business involves scenarios such as deep learning and high-performance computing, you can use TKE to support the GPU feature, which can help you quickly use a GPU container.

You can enable GPU scheduling in multiple ways:

- Adding a GPU node to the cluster
  - Creating a GPU instance
  - Adding an existing GPU instance
- Creating a GPU service container
  - Creating a GPU service container in the console
  - Creating a GPU service container by using an application or kubectl command

Prerequisites

You have logged in to the TKE console.

Notes

- GPU scheduling is supported only when the Kubernetes version of the cluster is **1.8.*** or later.
- GPUs are not shared among containers. A container can request one or more GPUs, but not part of a GPU.
- We recommend that you use the GPU feature together with affinity scheduling.

Directions

Adding a GPU node to a cluster

You can add a GPU node in either of the following ways:

- Creating a GPU instance
- Adding an existing GPU instance

Creating a GPU instance

1. Click **Clusters** in the left sidebar to go to the "Cluster Management" page.

2. Click **Create a Node** for the cluster in which the GPU instance is to be created.

3. On the "Select the Model" page, select **GPU Model** as the instance "Family" and select "GPU Compute GN2" as the "Model".

4. Complete the remainder of the process as instructed.

> **Note：**
>
> During CVM configuration, TKE automatically performs the initial processes such as GPU driver installation based on the selected model, and you can ignore the basic image.

Adding an existing GPU instance

1. Click **Clusters** in the left sidebar to go to the "Cluster Management" page.

2. Click **Add Existing Node** for the cluster in which an existing GPU instance is to be added.

3. On the "Select Nodes" page, select an existing GPU node and click **Next**.

4. Complete the remainder of the process as instructed.

> **Note：**
>
> During CVM configuration, TKE automatically performs the initial processes such as GPU driver installation based on the selected model, and you can ignore the basic image.

Creating a GPU service container

You can create a GPU service container in either of the following ways:

- Creating a GPU service container in the console
- Creating a GPU service container by using an application or kubectl command

Creating a GPU service container in the console

1. Click **Clusters** in the left sidebar to go to the "Cluster Management" page.
2. Click the ID or name of the cluster for which the workload is to be created to go to the cluster management page for this workload.

3. Select a workload type under "Workload" to go to the corresponding information page. For example, choose **Workload** > **DaemonSet** to go to the DaemonSet information page.
4. Click **Create** to go to the "Create Workload" page.
5. Specify information such as the workload name and namespace as instructed.
6. Click **Create Workload** to create the workload.

Creating a GPU service container by using an application or kubectl command

You can add a GPU field in the YAML file by using an application or kubectl command.

Custom Kubernetes Component Launch Parameters

Last updated：2022-04-22 09:42:02

Overview

To facilitate the configuration and management of Kubernetes component parameters in TKE clusters, Tencent Cloud supports custom Kubernetes component parameters. This document describes how to configure custom Kubernetes component parameters in clusters.

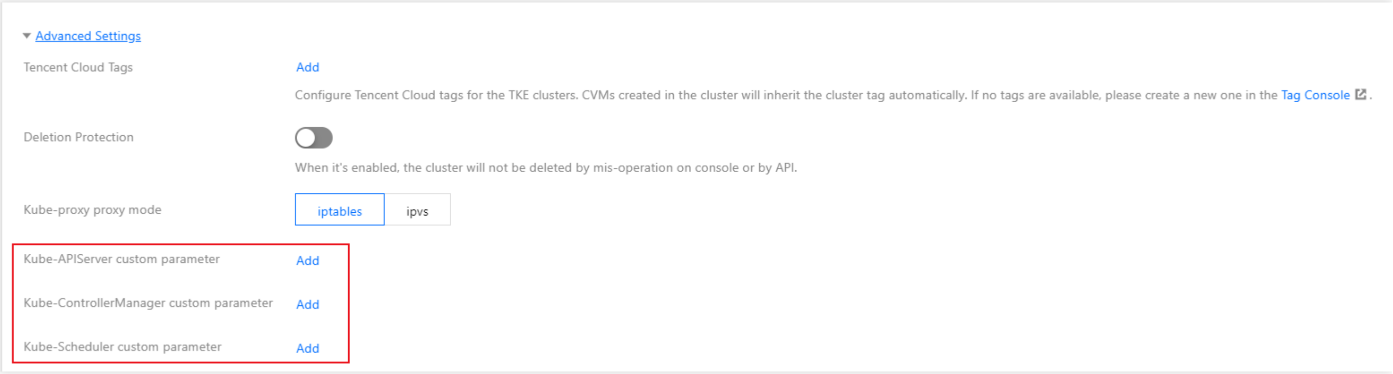
Notes

- To use custom Kubernetes component launch parameters, you need to submit a ticket to apply for it.
- While submitting the ticket, you need to provide custom Kubernetes component launch parameters, including your account ID, cluster ID, and the component and component parameters.
- For Kubernetes cluster version upgrade, due to the potential incompatibility of launch parameters after a Kubernetes version upgrade, major version upgrades will not retain the custom Kubernetes component parameters from your original cluster version. Therefore, you need to reconfigure custom Kubernetes component parameters.

Directions

Configuring custom Kubernetes component parameters when creating a cluster

1. Log in to the Tencent Cloud TKE console, and click **Clusters** in the left sidebar.
2. On the "Cluster Management" page, click **Create** above the cluster list.
3. On the "Create a cluster” page, choose **Advanced Settings** > **Configure Custom Kubernetes Component Parameters**, as shown in the figure below:



▼ Advanced Settings

Tencent Cloud Tags: Add

Configure Tencent Cloud tags for the TKE clusters. CVMs created in the cluster will inherit the cluster tag automatically. If no tags are available, please create a new one in the Tag Console.

Deletion Protection

When it's enabled, the cluster will not be deleted by mis-operation on console or by API.

Kube-proxy proxy mode: iptables | ipvs

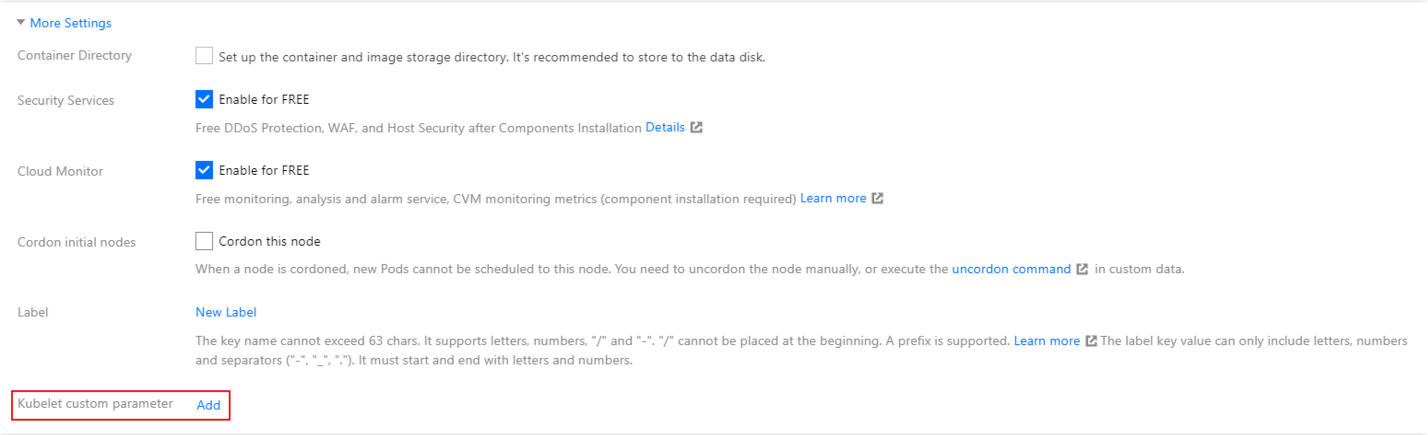
Kube-APIServer custom parameter: Add

Kube-ControllerManager custom parameter: Add

Kube-Scheduler custom parameter: Add

Configuring the custom Kubelet parameters of a node

On the **Create a cluster node**, **Add existing nodes**, **Create a node pool**, and **Add nodes** pages, you can configure the custom Kubelet parameters of a node, as shown in the figure below:



▼ More Settings

Container Directory: □ Set up the container and image storage directory. It's recommended to store to the data disk.

Security Services: ☑ Enable for FREE

Free DDoS Protection, WAF, and Host Security after Components Installation Details

Cloud Monitor: ☑ Enable for FREE

Free monitoring, analysis and alarm service, CVM monitoring metrics (component installation required) Learn more

Cordon initial nodes: □ Cordon this node

When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually, or execute the uncordon command in custom data.

Label: New Label

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Kubelet custom parameter: Add

Configuring custom Kubernetes component parameters when upgrading a cluster

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, select the ID of the desired cluster, and enter the cluster details page.
3. On the cluster's **Basic Information** page, click **Upgrade** to the right of the Kubernetes version. At the same time, set the Kubernetes component launch parameters.

Using KMS for Kubernetes Data Source Encryption

Last updated：2023-02-02 17:05:22

Overview

Tencent Cloud TKE-KMS Plugin integrates the rich key management features of Key Management Service (KMS) to provide powerful encryption/decryption capabilities for Secret in Kubernetes cluster. This document describes how to encrypt data for Kubernetes cluster via KMS.

Concepts

Key Management Service (KMS)

Key Management Service (KMS) is a security management solution that leverages a third-party certified hardware security module (HSM) to generate and protect keys so you can easily create and manage keys, helping you to meet your key management and compliance needs in multi-application and multi-business scenarios.

Prerequisites

You have created a TKE **self-deployed cluster** that meets the following conditions:

- Kubernetes v1.10.0 or later.
- Etcd v3.0 or later.

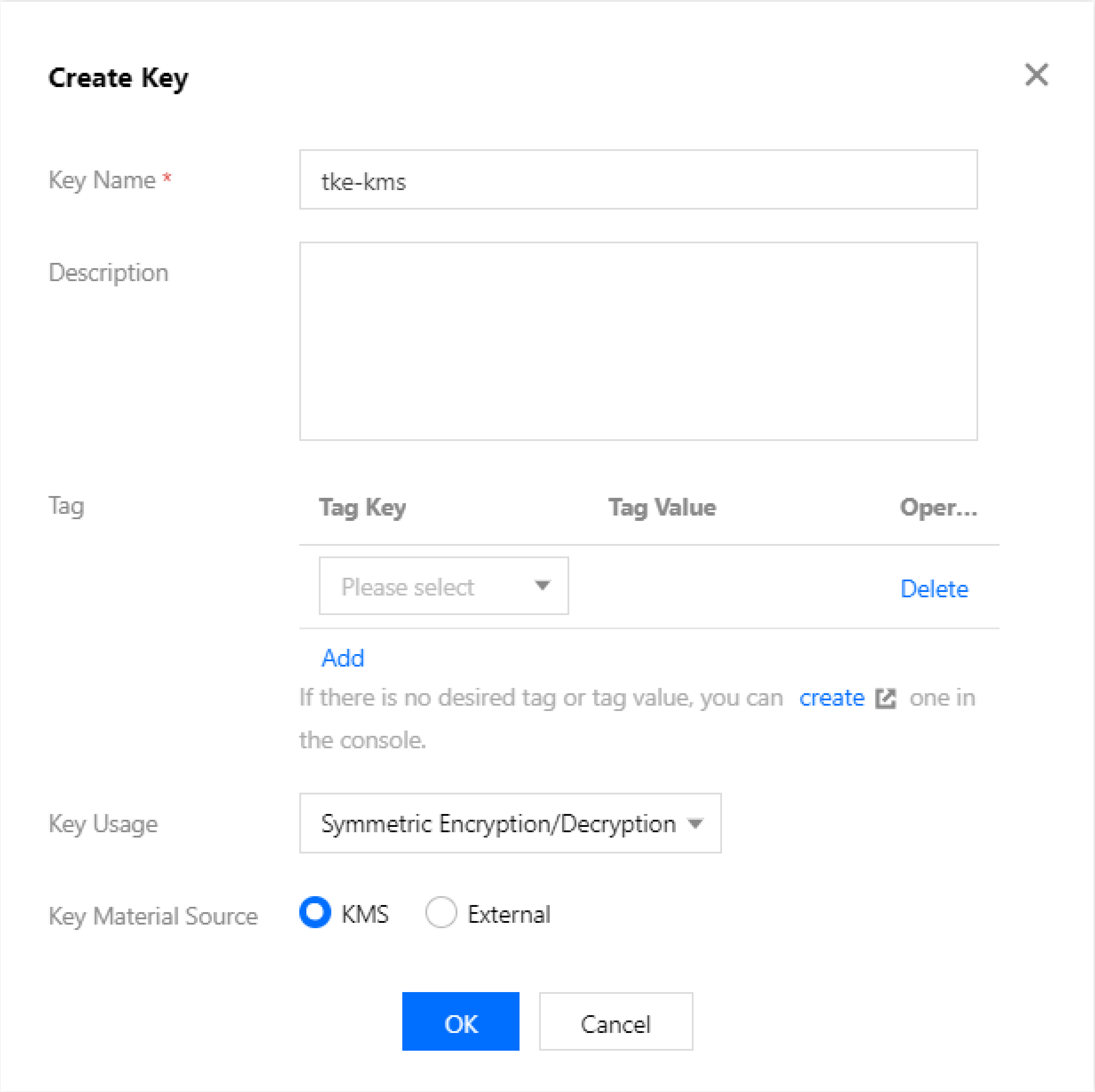
> Note：
> If you want to check the version, you can go to Cluster Management page and select the cluster ID to go to the **Basic Information** page to view.

Directions

Creating a KMS key and obtaining the ID

1. Log in to the KMS Console, and go to **Customer Managed CMK** page.

2. At the top of the **Customer Managed CMK** page, select the region for which you want to create a key, and click **Create**.
3. On the pop-up window, configure the parameters according to the following information, as shown below:



Create Key

Key Name * tke-kms

Description

Tag

Tag Key	Tag Value	Oper...
Please select		Delete

Add

If there is no desired tag or tag value, you can create one in the console.

Key Usage: Symmetric Encryption/Decryption

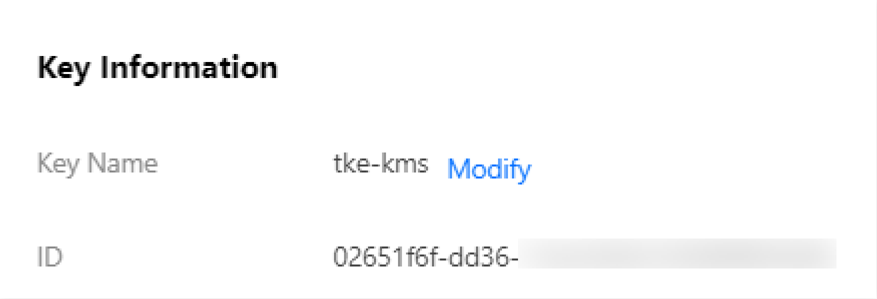
Key Material Source: ● KMS ○ External

The key parameters are as follows. Retain the default settings for other parameters.

- **Key Name**: this is required and must be unique within the region. It can contain letters, numbers, `_`, `-`, and cannot begin with `KMS-`. In this document, we take `tke-kms` as an example.
- **Description**: this is optional and used to specify the type of data to be protected, or the application to be used in conjunction with the CMK.
- **Key Usage**: select **Symmetric encryption and decryption**.

- **Key Material Source**: select **KMS** or **External** based on the actual needs. In this document, we take **KMS** as an example.

4. Click **OK** to go back to **Customer Managed CMK** page to view the created keys.
5. Click the key ID to go to **Key Information** page, you can view the complete ID of the key on this page. See the figure below:



Key Information

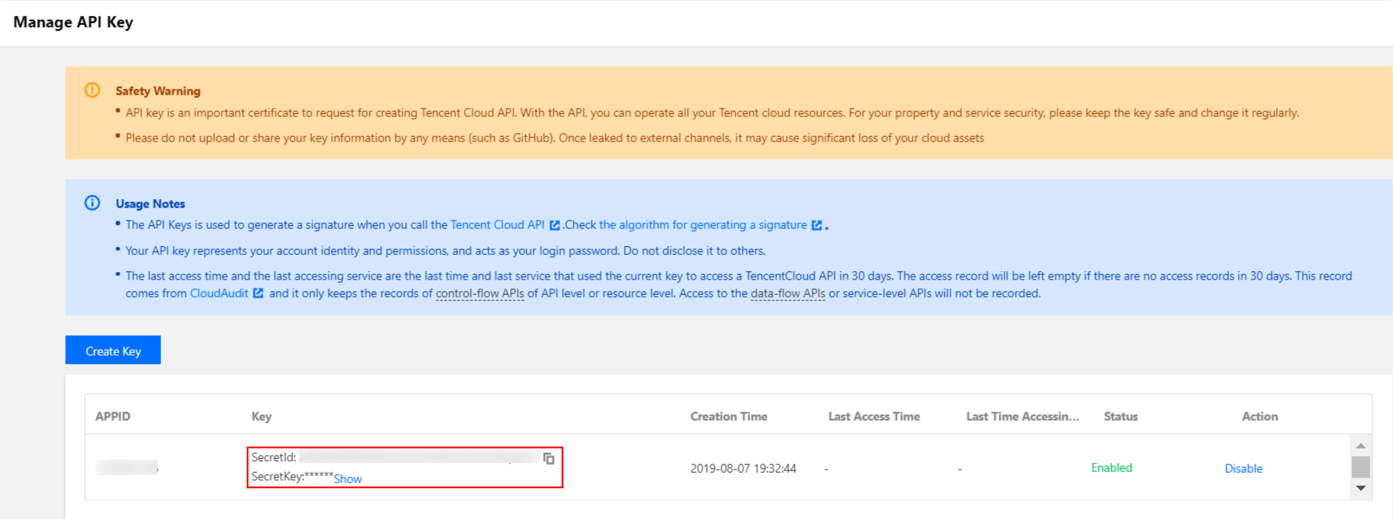
Key Name tke-kms Modify

ID 02651f6f-dd36-[redacted]

Creating and obtaining access key

Before using TKE for the first time, go to the TencentCloud API Key page to apply for `SecretId` and `SecretKey`. If you already have them, skip this procedure.

1. Log in to the CAM console and select **Access Key** > **Manage API Key** in the left sidebar to go to the **Manage API Key** page.
2. On the **Manage API Key** page, click **Create Key** to create a pair of `SecretId` / `SecretKey`.
3. You can check the key's information including `SecretId` and `SecretKey` on **Manage API Key** page when the creation is completed. See the figure below:



Manage API Key

Safety Warning
- API key is an important certificate to request for creating Tencent Cloud API. With the API, you can operate all your Tencent cloud resources. For your property and service security, please keep the key safe and change it regularly.
- Please do not upload or share your key information by any means (such as GitHub). Once leaked to external channels, it may cause significant loss of your cloud assets

Usage Notes
- The API Keys is used to generate a signature when you call the Tencent Cloud API .Check the algorithm for generating a signature .
- Your API key represents your account identity and permissions, and acts as your login password. Do not disclose it to others.
- The last access time and the last accessing service are the last time and last service that used the current key to access a TencentCloud API in 30 days. The access record will be left empty if there are no access records in 30 days. This record comes from CloudAudit and it only keeps the records of control-flow APIs of API level or resource level. Access to the data-flow APIs or service-level APIs will not be recorded.

APPID	Key	Creation Time	Last Access Time	Last Time Accessin...	Status	Action
[redacted]	SecretId: [redacted] SecretKey:******Show	2019-08-07 19:32:44	-	-	Enabled	Disable

Creating a DaemonSet and deploying tke-kms-plugin

1. Log in to the TKE console and click **Cluster** in the left sidebar.

2. On the **Cluster Management** page, click the ID of the cluster that meet the conditions to go to the cluster details page.

3. Select **Create Via YAML** at the top right corner on any interface of the cluster to go to **Create Via YAML** page. Enter the parameters for `tke-kms-plugin.yaml` , as shown below:

> Note：
> Enter values for the following parameters based on the actual needs:
>
> - `{{REGION}}` : the region where KMS key resides. You can check Region List for the valid values.
> - `{{KEY_ID}}` : enter the KMS key ID obtained in the step of creating a KMS key and obtaining the ID.
> - `{{SECRET_ID}}` and `{{SECRET_KEY}}` : enter the SecretID and SecretKey created in the step of creating and obtaining access key.
> - `images: ccr.ccs.tencentyun.com/tke-plugin/tke-kms-plugin:1.0.0` : tke-kms-plugin image address. If you want to use the self-created tke-kms-plugin image, you can replace it.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
name: tke-kms-plugin
namespace: kube-system
spec:
selector:
matchLabels:
name: tke-kms-plugin
template:
metadata:
labels:
name: tke-kms-plugin
spec:
nodeSelector:
node-role.kubernetes.io/master: "true"
hostNetwork: true
restartPolicy: Always
volumes:
- name: tke-kms-plugin-dir
hostPath:
path: /var/run/tke-kms-plugin
type: DirectoryOrCreate
```

```
tolerations:
- key: node-role.kubernetes.io/master
effect: NoSchedule
containers:
- name: tke-kms-plugin
image: ccr.ccs.tencentyun.com/tke-plugin/tke-kms-plugin:1.0.0
command:
- /tke-kms-plugin
- --region={{REGION}}
- --key-id={{KEY_ID}}
- --unix-socket=/var/run/tke-kms-plugin/server.sock
- --v=2
livenessProbe:
exec:
command:
- /tke-kms-plugin
- health-check
- --unix-socket=/var/run/tke-kms-plugin/server.sock
initialDelaySeconds: 5
failureThreshold: 3
timeoutSeconds: 5
periodSeconds: 30
env:
- name: SECRET_ID
value: {{SECRET_ID}}
- name: SECRET_KEY
value: {{SECRET_KEY}}
volumeMounts:
- name: tke-kms-plugin-dir
mountPath: /var/run/tke-kms-plugin
readOnly: false
```

4. Click **Done** and wait for the DaemonSet to be created.

Configuring kube-apiserver

1. Log in to each Master node of the cluster by referring to Logging in to Linux Instance Using Standard Login Method.

> Note：
> Master node security group defaults to close port 22. You need to open port 22 on the security group interface before logging in to the node. For more information, see Adding a Security Group Rule.

2. Run the following command to create and open the YAML file.

```
vim /etc/kubernetes/encryption-provider-config.yaml
```

3. Press **i** to switch to the edit mode and edit the YAML file. Enter the followings according to the K8s version that you actually use:

- K8S v1.13+:

```
apiVersion: apiserver.config.k8s.io/v1
kind: EncryptionConfiguration
resources:
- resources:
- secrets
providers:
- kms:
name: tke-kms-plugin
timeout: 3s
cachesize: 1000
endpoint: unix:///var/run/tke-kms-plugin/server.sock
- identity: {}
```

- K8S v1.10 - v1.12:

```
apiVersion: v1
kind: EncryptionConfig
resources:
- resources:
- secrets
providers:
- kms:
name: tke-kms-plugin
timeout: 3s
cachesize: 1000
endpoint: unix:///var/run/tke-kms-plugin/server.sock
- identity: {}
```

4. After editing is completed, press **Esc** and enter **:wq** to save the file and go back.

5. Run the following command to edit the YAML file.

```
vi /etc/kubernetes/manifests/kube-apiserver.yaml
```

6. Press **i** to switch to the edit mode and add the followings to `args` according to the K8s version you actually use.

> Note：
> Self-deployed cluster of K8s v1.10.5. You need to remove `kube-apiserver.yaml` from the `/etc/kubernetes/manifests` directory and move it back to the directory after you have completed the editing.

- K8S v1.13+:

```
--encryption-provider-config=/etc/kubernetes/encryption-provider-config.yaml
```

- K8S v1.10 - v1.12:

```
--experimental-encryption-provider-config=/etc/kubernetes/encryption-provider-c
onfig.yaml
```

7. Add Volume command to `/var/run/tke-kms-plugin/server.sock` . The location and content for adding is as follows:

> Note：
> `/var/run/tke-kms-plugin/server.sock` is a unix socket that is listened when tke kms server is launched. kube apiserver will access tke kms server by accessing the socket.

Add the followings for `volumeMounts:` :

```
- mountPath: /var/run/tke-kms-plugin
name: tke-kms-plugin-dir
```

Add the followings for `volume:` :

```
- hostPath:
path: /var/run/tke-kms-plugin
name: tke-kms-plugin-dir
```

8. When the editing is finished, press **Esc**, enter **:wq** and save the `/etc/kubernetes/manifests/kube-apiserver.yaml` file. Wait for kube-apiserver to restart.

Verification

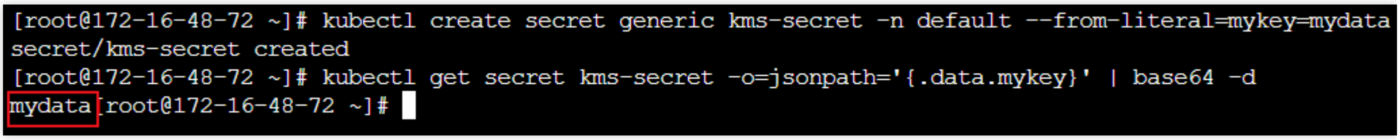
1. Log in to the node of the cluster and run the following command to create a Secret.

```
kubectl create secret generic kms-secret -n default --from-literal=mykey=mydata
```

2. Run the following command to verify if the Secret has been decrypted correctly.

```
kubectl get secret kms-secret -o=jsonpath='{.data.mykey}' | base64 -d
```

3. If the output value is `mydata` , i.e. it is equal to the value of Secret, it means Secret has been decrypted correctly. See the figure below:



```
[root@172-16-48-72 ~]# kubectl create secret generic kms-secret -n default --from-literal=mykey=mydata
secret/kms-secret created
[root@172-16-48-72 ~]# kubectl get secret kms-secret -o=jsonpath='{.data.mykey}' | base64 -d
mydata[root@172-16-48-72 ~]#
```

References

For more information about Kubernetes KMS, see Using a KMS provider for data encryption.

Images

Image Overview

Last updated：2023-02-01 16:10:50

Overview

This document describes two types of images supported by TKE and their respective use cases and instructions. For more information, see Image Types.

> Note：
>
> - TKE provides SLA guarantees only for public images.
> - For custom images in non-standard operating environments without TKE's compatibility adaptation, you need to ensure the image availability in Kubernetes environments. In principle, TKE does not provide SLA and technical support for these images.

- **Public image**: They are images officially provided by Tencent Cloud. Each image contains an operating system and initialization components provided by Tencent Cloud, and is available to all users.
- **Custom image**: It is created by using the image creation feature or imported by using the image import feature. A custom image is only available to the creator and the people they share it with. It is a non-standard environment that doesn't come with official support and ongoing maintenance from Tencent Cloud.

Limits

- There are two levels of operating systems, including **cluster level** and **node pool level**.
  - OS configured at the cluster level is used when creating a node, adding an existing node, and upgrading a node in a cluster.
  - When adding existing nodes or expanding the node capacity inside the node pool, you will use the OS at the node pool level.
- Changes to the OS only apply to new nodes and reinstalled nodes, **but not existing nodes**.

List of Public Images Supported by TKE

TKE offers the following **public images** that you can choose as needed.

> Note：
> Whenever TKE plans to adjust the image logic, we will notify you **at least one week** in advance via Message Center, SMS, and email.
> Image logic changes will not affect the existing nodes previously created by using an earlier image. For better results, we recommend you use a later basic image.

Image ID	OS Name	OS Name Displayed in the Console	OS Type	Release Status	Notes
img-9axl1k53	tlinux2.4(tkernel4)x86_64	TencentOS Server 2.4(TK4)	Tencent OS Server	Full release	Kernel version: 5.4.119
img-3la7wgnt	centos7.8.0_x64	CentOS 7.8	CentOS	Full release	CentOS 7.8 public kernel
img-eb30mz89	tlinux3.1x86_64	TencentOS Server 3.1(TK4)	Tencent OS Server	Full release	• We recommend you use the latest release of TencentOS Server. • Kernel version: 5.4.119
img-hdt9xxkt	tlinux2.4x86_64	TencentOS Server 2.4 Formerly known as Tencent linux release 2.4 (Final)	Tlinux	Full release	Kernel version: 4.14.105
img-22trbn9x	ubuntu20.04x86_64	Ubuntu Server 20.04.1 LTS 64bit	Ubuntu	It is in beta. To join it, please submit a ticket to apply.	Ubuntu 20.04.1 public kernel

Image ID	OS Name	OS Name Displayed in the Console	OS Type	Release Status	Notes
img-pi0ii46r	Ubuntu18.04.1x86_64	Ubuntu 18.04 LTS 64bit	Ubuntu	Full release	Ubuntu 18.04.1 public kernel
img-25szkc8t	centos8.0x86_64	CentOS 8.0	CentOS	It is in beta. To join it, please submit a ticket to apply.	CentOS 8.0 public kernel
img-9qabwvbn	CentOS7.6.0_x64	CentOS 7.6	CentOS	Full release	CentOS 7.6 public kernel

TKE-Optimized Series Images

Last updated：2023-05-06 17:30:07

Tencent Linux, a public image of Tencent Cloud that contains TencentOS-kernel (a customized kernel maintained by the Tencent Cloud team), is available in TKE as a default image.

TKE-Optimized images are once used for improving image stability and providing more features, but the images are no longer available for the clusters in TKE console after the Tencent Linux public image is launched.

Note

The clusters that are still using TKE-Optimized images are not affected and can continue to use the images. But it is recommended that you switch to Tencent Linux 2.4. The new nodes in the cluster will use Tencent Linux 2.4 while the existing nodes are not affected.

The CentOS 7.6 TKE-Optimized image is fully compatible with Tencent Linux 2.4.

The user space tools of the Ubuntu 18.04 TKE-Optimized image are not fully compatible with Tencent Linux. If you have used these tools in your script, you need to modify the script after switching to Tencent Linux.

Worker node introduction

Node Overview

Last updated：2020-12-24 10:07:44

Introduction

A node is a basic element of a container cluster. It can be either a virtual machine or a physical machine, depending on the service. Each node contains the basic components required for running a pod, including Kubelet and Kube-proxy.

Node-Related Operations

- Adding a node
- Removing a node
- Draining or cordoning a node
- Configuring the startup script of a node
- Using a GPU node
- Setting a Node Label

Node Lifecycle

Last updated：2020-02-24 18:33:29

Notes on Node Lifecycle Status

Status	Description
Healthy	The node is running normally and connected to the cluster.
Exceptional	The node is exceptional and not connected to the cluster.
Cordoned	The node is cordoned and no new Pods can be scheduled to this node.
Draining	The node is draining the Pod to another node.
Other	See CVM Lifecycle.

Node Resource Reservation Description

Last updated：2022-09-22 11:01:21

TKE clusters occupy node resources to run add-ons (such as kubelet, kube-proxy, and runtime). Therefore, **the total number of node resources** and **the number of allocable resources in a cluster** may differ from each other. This document describes the policies and notes in terms of node resource reservation in TKE clusters so that you can set reasonable numbers of requested resources and limited resources for Pods when deploying an application.

Policy for Calculating Allocable Node Resources

Calculation formula

Allocable = Capacity - Reserved - Eviction - Threshold

Node CPU reservation rules

Node CPU	Reservation Rule	Description
1c <= CPU <= 4c	0.1c is reserved.	-
4c < CPU <= 64c	0.1c is reserved for the 4c part, and 2.5% for the excessive part.	For example, if CPU = 32c, reserved resources = 0.1 + (32 - 4) * 2.5% = 0.8c.
64c < CPU <= 128c	0.1c is reserved for the 4c part, 2.5% for the 4c to 64c part, and 1.25% for the excessive part.	For example, if CPU = 96c, reserved resources = 0.1 + (64 - 4) * 2.5% + (96 - 64) * 1.25% = 2c.
CPU > 128c	0.1c is reserved for the 4c part, 2.5% for the 4c to 64c part, 1.25% for the 64c to 128c part, and 0.5% for the excessive part.	For example, if CPU = 196c, reserved resources = 0.1 + (64 - 4) * 2.5% + (128 - 64) * 1.25% + (196 - 128) * 0.5% = 2.74c.

Node memory reservation rules

Node Memory	Reservation Rule	Description
1 GB <= Memory <= 4 GB	25% is reserved.	For example, if memory = 2 GB, reserved resources = 2 * 25% = 512 MB.

Node Memory	Reservation Rule	Description
4 GB < Memory <= 8 GB	25% is reserved for the 4 GB part, and 20% for the excessive part.	For example, if memory = 8 GB, reserved resources = 4 * 25% + (8 - 4) * 20% = 1,843 MB.
8 GB < Memory <= 16 GB	25% is reserved for the 4 GB part, 20% for the 4 GB to 8 GB part, and 10% for the excessive part.	For example, if memory = 12 GB, reserved resources = 4 * 25% + (8 - 4) * 20% + (12 - 8) * 10% = 2,252 MB.
16 GB < Memory <= 128 GB	25% is reserved for the 4 GB part, 20% for the 4 GB to 8 GB part, 10% for the 8 GB to 16 GB part, and 6% for the excessive part.	For example, if memory = 32 GB, reserved resources = 4 * 25% + (8 - 4) * 20% + (16 - 8) * 10% + (32 - 16) * 6% = 3,645 MB.
Memory > 128 GB	25% is reserved for the 4 GB part, 20% for the 4 GB to 8 GB part, 10% for the 8 GB to 16 GB part, 6% for the 16 GB to 128 GB part, and 2% for the excessive part.	For example, if memory = 320 GB, reserved resources = 4 * 25% + (8 - 4) * 20% + (16 - 8) * 10% + (128 - 16) * 6% + (320 - 128) * 2% = 13,475 MB.

Note：

You can use custom kubelet parameters to modify `kube-reserved` for node resource reservation. We recommend you reserve sufficient CPU and memory resources for add-ons to ensure node stability.

Viewing Allocable Node Resources

Run the following command (replace `NODE_NAME` with the actual node name) to check the allocable node resources in a cluster. The output result contains `Capacity` and `Allocatable` fields, along with measurements of CPU, memory, and temporary storage.

```
kubectl describe node NODE_NAME | grep Allocatable -B 7 -A 6
```

Notes

- The reservation policy automatically takes effect for K8s v1.16 or later and nodes created after June 24, 2022, without no manual configuration required.
- To ensure your business stability, the reservation policy won't take effect for existing nodes. This is because allocable resources may become fewer based on the calculation method, which means possible node eviction for nodes requiring a large number of resources.
- If you want to apply the reservation policy to existing nodes, remove them from the cluster without termination and then add them in the TKE console. In this case, they become newly added nodes subject to the policy by default.

Adding a Node

Last updated：2023-05-19 11:26:14

Overview

You can add a node to your cluster in the following ways:

Creating a node

Adding an existing node

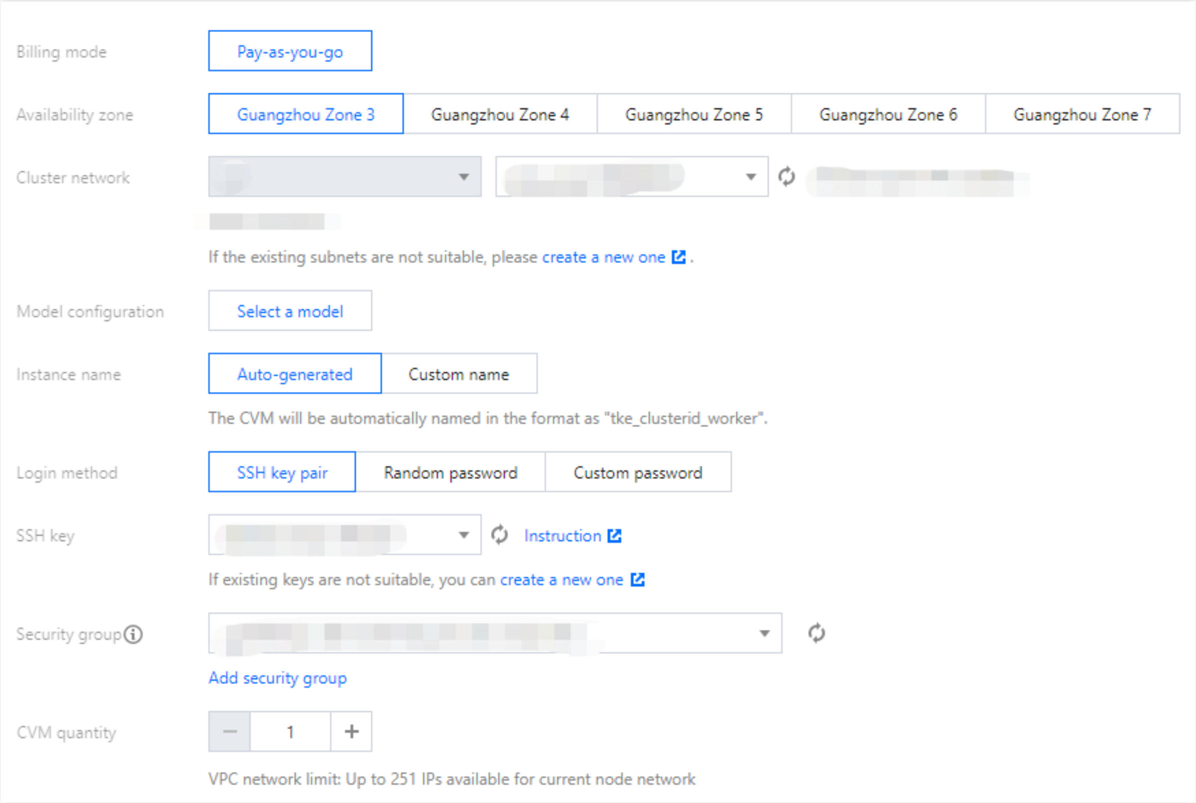
Prerequisites

You have logged in to the TKE console.

Directions

Creating a node

1. In the left sidebar, click Cluster to go to the **Cluster Management** page.
2. Click the ID of the target cluster to go to the details page of the cluster.
3. Choose **Node Management** > **Node** in the left sidebar to go to the node list page, and click **Create Node**.
4. On the **Create Node** page, configure the parameters as instructed below.



Billing mode: Pay-as-you-go

Availability zone: Guangzhou Zone 3 | Guangzhou Zone 4 | Guangzhou Zone 5 | Guangzhou Zone 6 | Guangzhou Zone 7

Cluster network: [redacted] [redacted] [redacted]

[redacted]

If the existing subnets are not suitable, please create a new one .

Model configuration: Select a model

Instance name: Auto-generated | Custom name

The CVM will be automatically named in the format as "tke_clusterid_worker".

Login method: SSH key pair | Random password | Custom password

SSH key: [redacted] Instruction

If existing keys are not suitable, you can create a new one

Security group: [redacted]

Add security group

CVM quantity: − 1 +

VPC network limit: Up to 251 IPs available for current node network

The main parameters are described as follows:

Billing Mode: **Pay-as-you-go** is supported. For more information, see Billing Plans.

Availability Zone: This parameter is used to filter the available subnet list under the available zone.

Cluster Network: Select the subnet that assigns IP to the created node. A single node creation only supports a single subnet.

Model Configuration: Click **Select a Model**. On the **Model Configuration** page, select the values as needed based on the following descriptions:

Model: Select the model by specifying the number of CPU cores, memory size, and instance type. For more information, see Families and Models.

System disk: Controls the storage and schedules the operating of Cloud Virtual Machines (CVMs). You can view the system disk types available for the selected model and select the system disk as required. For more information, see Cloud Disk Types.

Data disk: Stores all user data.

Instance Name: The CVM instance name displayed on the console, which is determined by the naming mode of host name. The following two naming methods are provided:

Auto-generated: The host name will be automatically named. It supports sequential numbering or custom format for multiple instances. Up to 60 characters allowed. The instance name is automatically generated by default in the format of `tke_cluster id_worker` .

Custom Name: The host name is manually configured. The instance name is the same as the host name without reconfiguration.

Login Method: Select any one of the following login methods as required:

SSH Key Pair: A key pair is a pair of parameters generated by an algorithm. It is a way to log in to a CVM instance that is more secure than regular passwords. For more details, see SSH Key.

SSH Key: This parameter displays only when **SSH Key Pair** is selected. Select an existing key in the drop-down list. For how to create a key, see Creating an SSH key.

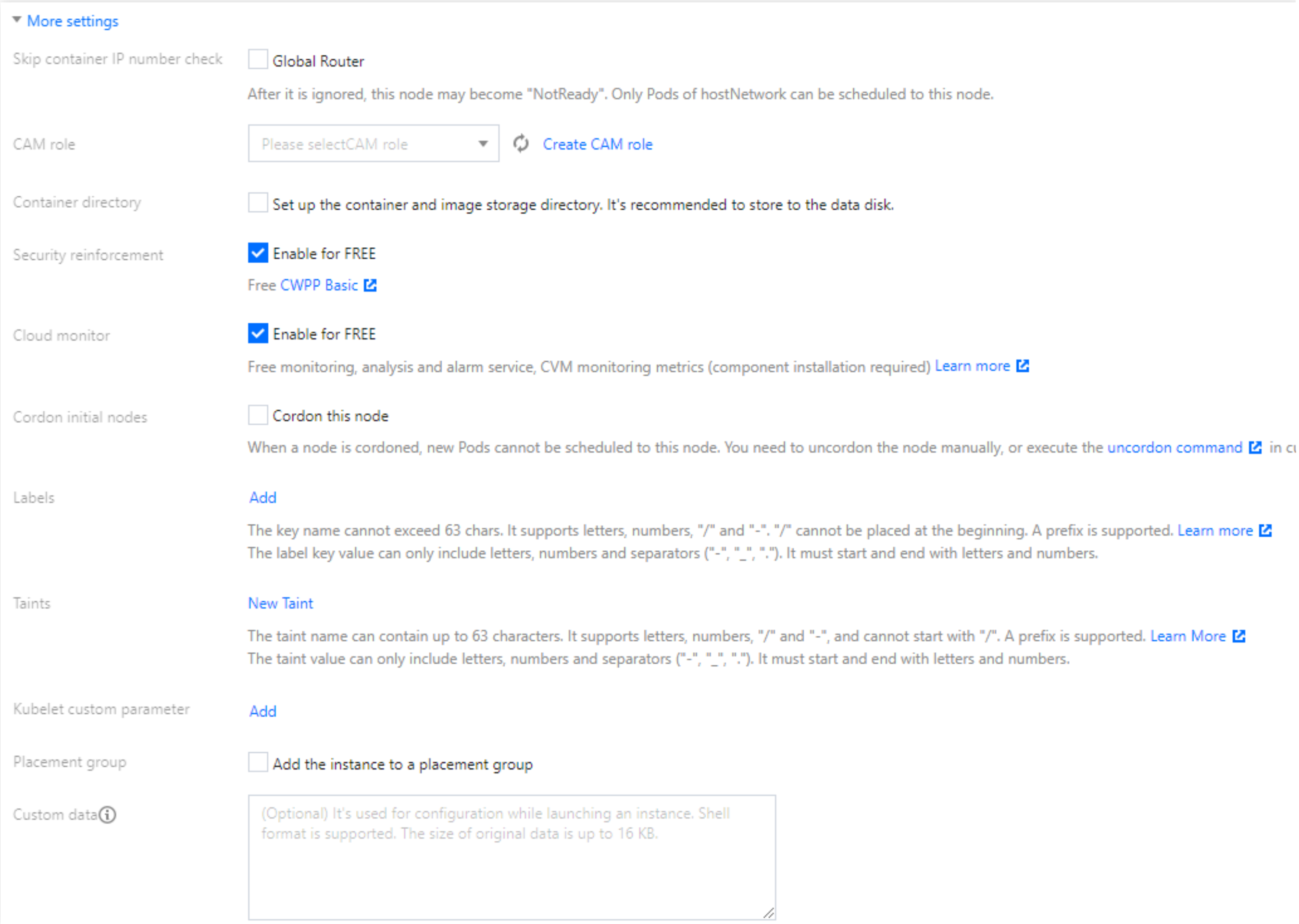
Random Password: The system sends an automatically generated password to your Message Center.

Custom Password: Set a password as prompted.

Security Group: The default value is the security group specified when the cluster is created. You can replace the security group or add a security group as required.

Amount: Specify the desired capacity as needed.

5. (Optional) Click **More Settings** on the **Create Node** page to view or configure more information.



▾ More settings

Skip container IP number check	□ Global Router After it is ignored, this node may become "NotReady". Only Pods of hostNetwork can be scheduled to this node.
CAM role	Please selectCAM role ▾ Create CAM role
Container directory	□ Set up the container and image storage directory. It's recommended to store to the data disk.
Security reinforcement	☑ Enable for FREE Free CWPP Basic
Cloud monitor	☑ Enable for FREE Free monitoring, analysis and alarm service, CVM monitoring metrics (component installation required) Learn more
Cordon initial nodes	□ Cordon this node When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually, or execute the uncordon command in c
Labels	Add The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.
Taints	New Taint The taint name can contain up to 63 characters. It supports letters, numbers, "/" and "-", and cannot start with "/". A prefix is supported. Learn More The taint value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.
Kubelet custom parameter	Add
Placement group	□ Add the instance to a placement group
Custom data ⓘ	(Optional) It's used for configuration while launching an instance. Shell format is supported. The size of original data is up to 16 KB.

CAM Role: You can bind all the nodes created this time to the same CAM role, and grant the authorization policy bound to the role to the nodes. For more information, see Managing Roles.

Container Directory: Select this option to set up the container and image storage directory. We recommend that you store to the data disk, such as `/var/lib/docker` .

Security Services: Free DDoS, Web Application Firewall (WAF) and Cloud Workload Protection (CWP) are activated by default. For more information, see Cloud Workload Protection.

Cloud Monitor: Free monitoring, analysis, and alarms are activated by default, and components are installed to obtain CVM monitoring metrics. For more information, see Tencent Cloud Observability Platform.

Cordon initial nodes: After you check **Cordon this node**, new Pods cannot be scheduled to this node. You can uncordon the node manually, or execute the uncordon command in custom data as needed.

Label: Click **New Label** to custom a label, which is used to filter or manage nodes.

Custom Data: Specify custom data to configure the node, that is, to run the configured script when the node is started up. You need to ensure the reentrant and retry logic of the script. The script and its log files can be viewed at the node path: `/usr/local/qcloud/tke/userscript` .

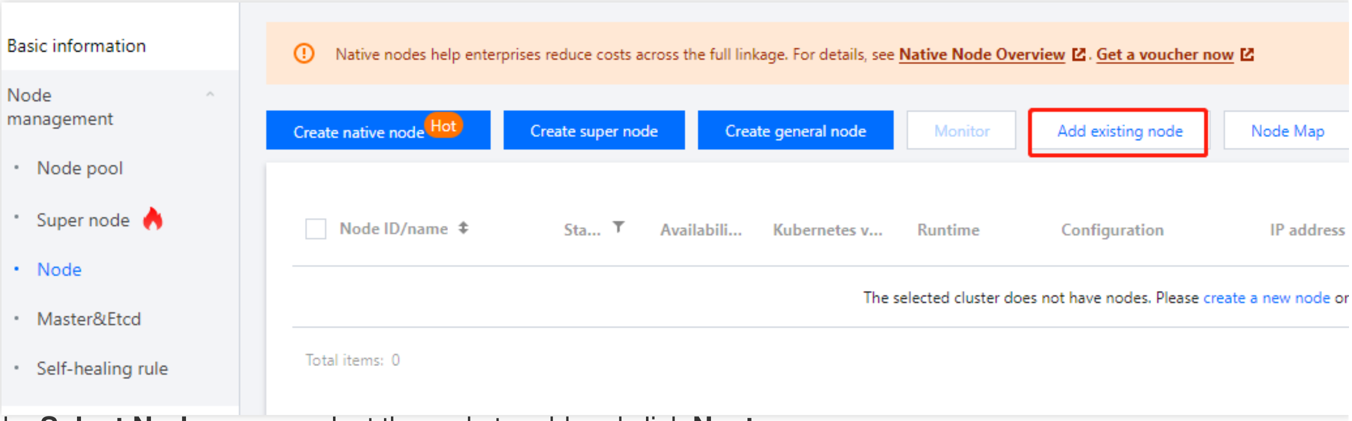
Adding an existing node

Note

Only CVM instances in the same VPC can be added to a cluster.

Do not add public gateway CVMs to the cluster. A DNS exception occurs when this type of CVM is reinstalled and added to the cluster, and the node becomes unavailable.

1. In the left sidebar, click Cluster to go to the **Cluster Management** page.
2. Click the ID of the target cluster to go to the details page of that cluster.
3. Choose **Node Management** > **Node** to go to the node list page, and click **Add Existing Node**.



4. On the **Select Nodes** page, select the node to add and click **Next**.
5. On the **CVM Configuration** page, configure the CVM instance to add to the cluster.

Mount Data Disk: The related settings for formatting the mounting. Enter the device name and mount point, and select whether to format the system or not.

Note

If you need to mount NVMe data disks to a high I/O, high-performance HCC model, you are advised to set file system volume labels for the data disks and add them to the cluster independently, without adding them to other models at the same time.

Back up the important data in advance. If you have formatted the disk, you don’t need to format the system, just enter the mount point.

The settings for formatting the mounting will take effect for the selected nodes. Please ensure that the entered device name, for example, /dev/vdb meets your expectations (If you have performed hot swapping and other operations on CBS, the device name may change).

If you have created partitions or are using LVM, please enter the partition name or logical volume name in the device name, and configure the corresponding parameters for formatting the mounting.

If you enter the incorrect device name, an error will occur and the node initialization will be terminated.

If the entered mount point does not exist, a corresponding directory will be created, and no error will occur.

Do not check: Do not set the data disk mounting. You can manually mount or use the script to mount.

Check: You need to set the device name, format system (you can select **Do not format**), and mount point. If you want to format the device "/dev/vdb" into "ext4" and mount it to the `/var/lib/docker` directory, you can set the device name to `/dev/vdb` , select `ext4` for the format system, and set the mount point to `/var/lib/docker` .

Container Directory: Set up the container and image storage directory. It's recommended to store to the data disk.

Operating System: You can modify the OS setting in the cluster details page. After the modification, the newly added or reinstalled nodes will use the new operating system.

Login Method:

Custom Password: Set a password as prompted.

SSH Key Pair: A key pair is a pair of parameters generated by an algorithm. Logging in to a CVM using a key pair is more secure than using regular passwords. For more information, see SSH Keys.

Random Password: A password will be automatically generated and sent to you through the Message Center.

Security Group: Configure network access control for the CVM instance as needed. You can click **Add Security Group** to open other ports to the internet.

6. Click **Done**.

Removing a Node

Last updated：2022-04-27 15:08:30

Scenario

This document guides you through the process of removing a node from a cluster.

Considerations

- A pay-as-you-go node can be retained or terminated as needed, but if it is not terminated, fees will continue to be incurred.
- Keep in mind that if a node is removed from and then added back to the cluster, the system will be reinstalled.

Directions

1. Log in to the TKE console and select **Clusters** in the left sidebar.
2. In the Cluster Management list page, click the ID/name of the cluster of the node to be removed to go to the details page.
3. On the left sidebar, select **Node Management** > **Node** to go to the **Node List** page.
4. In the node list, select the row of the node to be removed and click **Remove**.
5. When the "Are you sure you want to remove the following nodes?" window pops up, click **OK** to complete the removal.

Draining or Cordoning a Node

Last updated：2022-04-22 11:55:50

Overview

This document explains how to drain or cordon a node.

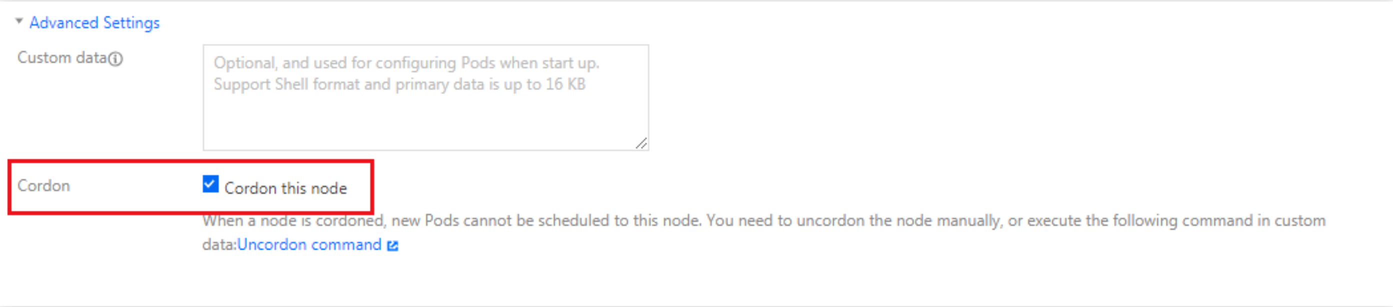
Directions

Cordoning a Node

After cordoning a node, new Pods cannot be scheduled to it. If you want to schedule a Pod to the node, you need to uncordon the node manually. If a node has been bound as backend target node, it will be removed from the target node list after it is condoned. You can cordon a node with one of the following two methods:

- Method A
- Method B

- When adding a node, on the **CVM Configuration** page, click **Advanced Settings** and select **Cordon this node**.



▾ Advanced Settings

Custom data: Optional, and used for configuring Pods when start up. Support Shell format and primary data is up to 16 KB

Cordon: ☑ Cordon this node

When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually, or execute the following command in custom data:Uncordon command

Uncordoning a Node

After a node is uncordoned, new Pods can be scheduled to it. You can uncordon a node with one of the following two methods:

- Method A
- Method B

When you create a node by running a script, you can uncordon it by adding a command for uncordoning the node in the script. Below is an example:

```
#!/bin/sh
# your initialization script
echo "hello world!"
# If you set unschedulable when you create a node,
# after executing your initialization script,
# use the following command to make the node schedulable.
node=`ps -ef|grep kubelet|grep -oE 'hostname-override=\S+'|cut -d"=" -f2`
#echo ${node}
kubectl uncordon ${node} --kubeconfig=/root/.kube/config
```

The `kubectl uncordon` command indicates uncordoning the node.

Draining a Node

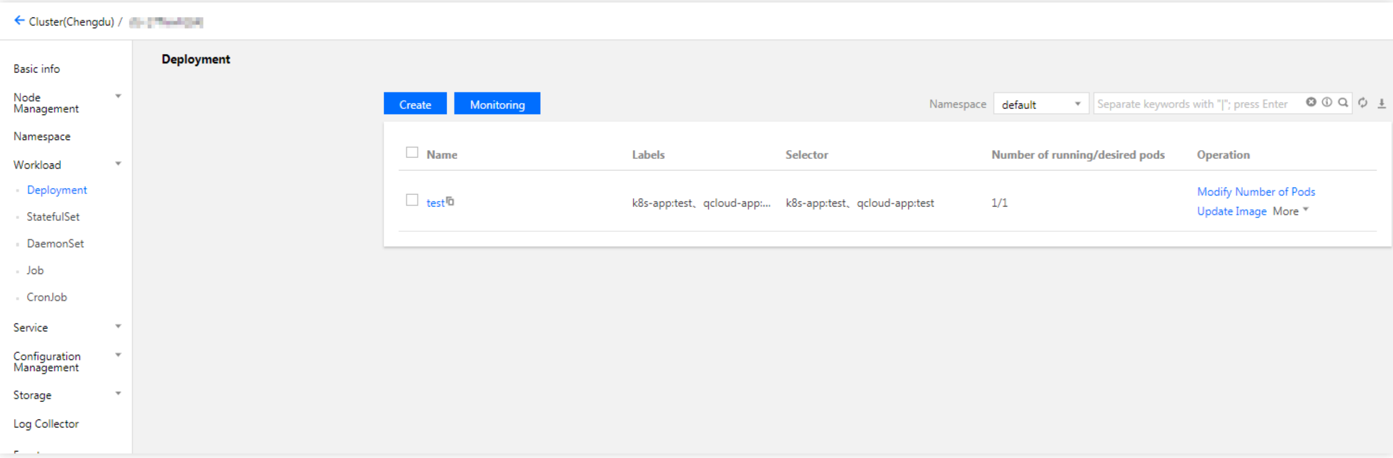
Overview

Before performing maintenance on a node, you can safely drain a Pod from a node by draining the node. After the node is drained, all Pods (excluding those managed by DaemonSet) in the node will be automatically drained to other nodes in the cluster, and the drained node will be set to cordoned status.

> Note：
> For locally stored Pods, data will be lost after they are drained. Please be cautious when doing so.

Directions

1. Log in to the TKE console.
2. In the left sidebar, click **Clusters** to go to the cluster management page.
3. Click the ID/name of the cluster where to drain the node to go to the management page of the cluster. See the figure below:



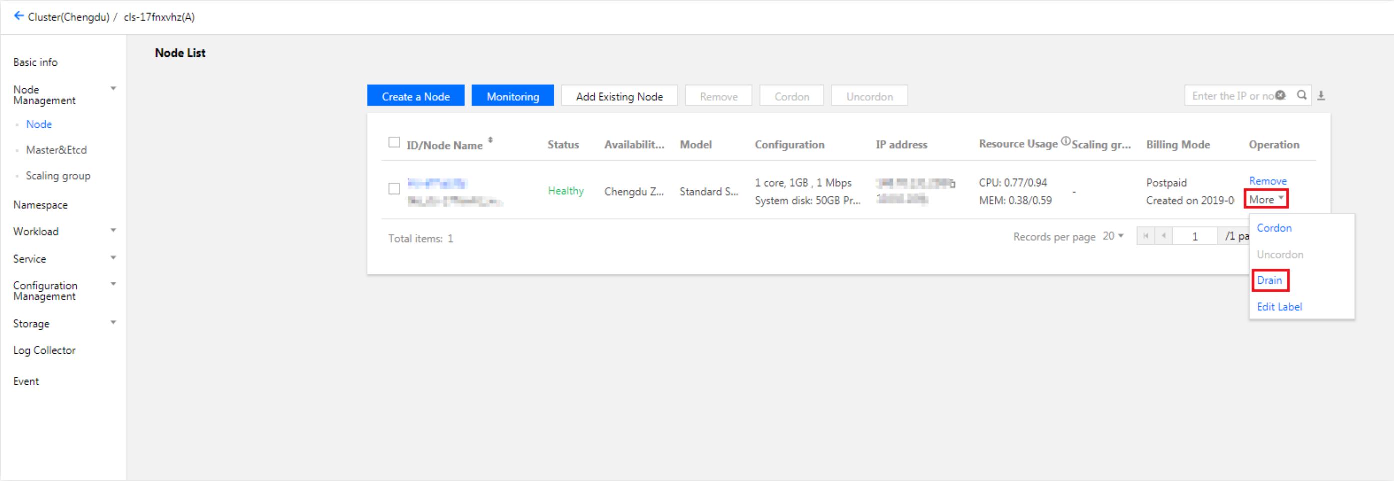
Cluster(Chengdu) / [redacted]

Deployment

Namespace: default

Name	Labels	Selector	Number of running/desired pods	Operation
test	k8s-app:test、qcloud-app:...	k8s-app:test、qcloud-app:test	1/1	Modify Number of Pods Update Image More

4. In the left sidebar, select **Node Management** > **Nodes** to go to the **Node List** page.
5. Click **More** > **Drain** in the row of the node to be drained. See the figure below:



ID/Node Name	Status	Availabilit...	Model	Configuration	IP address	Resource Usage	Scaling gr...	Billing Mode	Operation
[redacted]	Healthy	Chengdu Z...	Standard S...	1 core, 1GB , 1 Mbps System disk: 50GB Pr...	[redacted]	CPU: 0.77/0.94 MEM: 0.38/0.59	-	Postpaid Created on 2019-0	Remove More

Total items: 1 Records per page 20 1 /1 pa

6. In the pop-up dialog box, click **OK** to complete the draining.

Setting the Startup Script of a Node

Last updated：2023-05-25 15:31:54

Scenario

The startup script of a node can help you initialize the node before the node becomes ready. The configured script is executed when the node starts. If you purchase multiple CVM instances at a time, the custom data will run on all of them.

Use Limits

- Do not modify configurations such as those for kubelet, kube-proxy, and docker on the TKE node in the startup script.
- If the startup script fails to be executed, it will not be executed again. Therefore, you must ensure the executability of the script or ensure that a retry mechanism is available.
- You can view the script and its log file in the `/usr/local/qcloud/tke/userscript` path of the node.

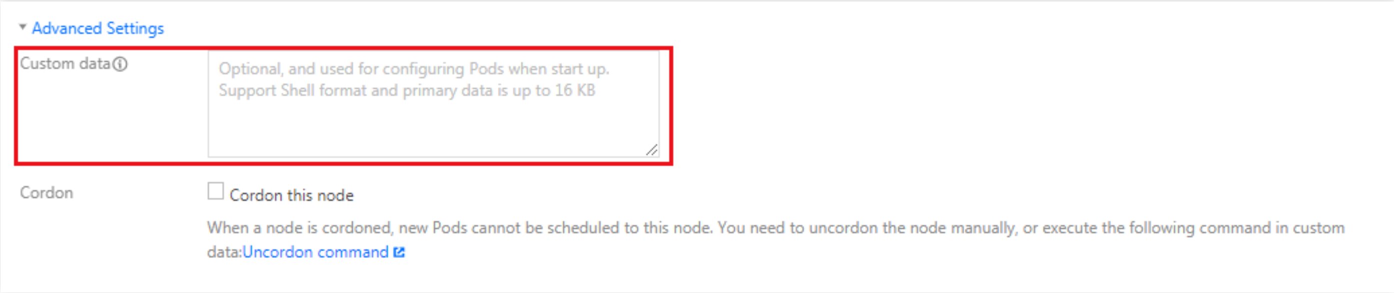
Directions

You can configure the startup script for a node in the following scenarios:

- Configuring the node startup script when creating a cluster or a node
- Configuring the node startup script when adding an existing node
- Configuring the node startup script when creating a scaling group

When creating a cluster or a node

- When creating a cluster and adding a node, click **Advanced Settings** on the "CVM Configuration" page and complete custom data as a startup script, as shown in the following figure:



▾ Advanced Settings

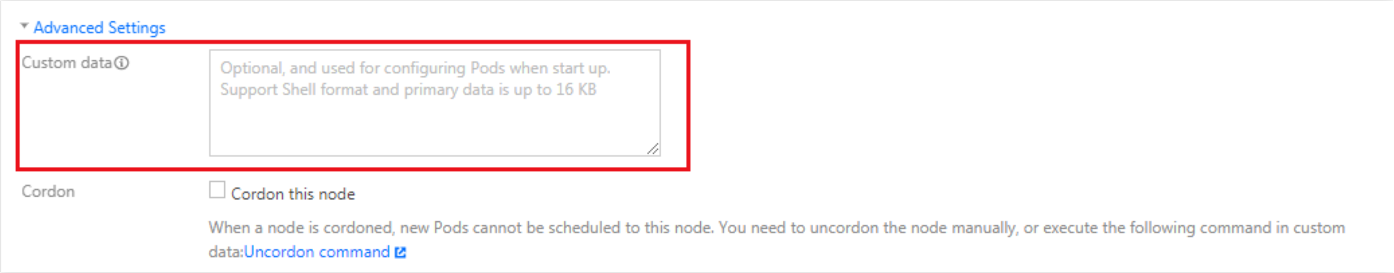
Custom data ⓘ: Optional, and used for configuring Pods when start up. Support Shell format and primary data is up to 16 KB

Cordon: □ Cordon this node

When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually, or execute the following command in custom data:Uncordon command

When adding an existing node

- When adding an existing node, click **Advanced Settings** on the "CVM Configuration" page and complete custom data as a startup script, as shown in the following figure:



▾ Advanced Settings

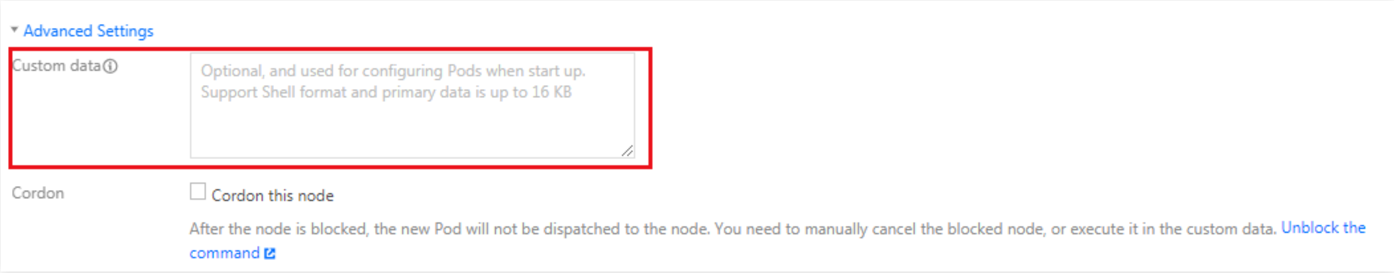
Custom data ⓘ: Optional, and used for configuring Pods when start up. Support Shell format and primary data is up to 16 KB

Cordon: □ Cordon this node

When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually, or execute the following command in custom data:Uncordon command

When creating a scaling group

- When creating a scaling group, click **Advanced Settings** on the "Launch Configuration" page and complete custom data as a startup script, as shown in the following figure:



▾ Advanced Settings

Custom data ⓘ: Optional, and used for configuring Pods when start up. Support Shell format and primary data is up to 16 KB

Cordon: □ Cordon this node

After the node is blocked, the new Pod will not be dispatched to the node. You need to manually cancel the blocked node, or execute it in the custom data. Unblock the command

Using GPU Node

Last updated：2022-06-10 16:48:44

Overview

If your business involves scenarios such as deep learning and high-performance computing, you can use TKE to support the GPU feature, which can help you quickly use a GPU container.
There are many ways to create a GPU CVM instance:

- Create a GPU CVM instance
- Add an existing GPU CVM instance
- Create a node pool

Usage Limits

- You need to select the GPU model for the added node. You can have the GPU driver automatically installed as needed. For more information, see GPU Driver.
- TKE supports GPU scheduling only if the Kubernetes version of the cluster is later than **1.8.***.
- By default, GPUs are not shared among containers. A container can request one or more GPUs, but not part of a GPU.
- The **master node** in a self-deployed cluster currently does not support the GPU model setting.

Directions

Creating GPU CVM instance

For more information, see Adding a Node. When creating a GPU, you should pay special attention to the following parameters:

Model

On the **Select Model** page, set **Model** in **Node Model** to GPU.

GPU driver, CUDA version, and cuDNN version

After setting the model, you can select the GPU driver version, CUDA version, and cuDNN version as needed.

- If you select **Automatically install GPU driver on the backend**, it will be installed automatically during system start, taking 15–25 minutes.
- The supported driver versions are determined by both the operating system and the GPU model.
- If you do not select **Automatically install GPU driver on the backend**, the GPU driver will be installed by default for some operating systems of earlier versions to ensure the normal use. The complete default driver version information is as shown below:

Operating System	Default Driver Version Installed
CentOS 7.6, Ubuntu 18, Tencent Linux2.4	450
CentOS 7.2 (not recommended)	384.111
Ubuntu 16 (not recommended)	410.79

MIG

With multi-instance GPU (MIG) enabled, an A100 GPU will be divided into seven separate GPU instances to help you improve the GPU utilization when multiple jobs are running. For more information, see NVIDIA Multi-Instance GPU User Guide.

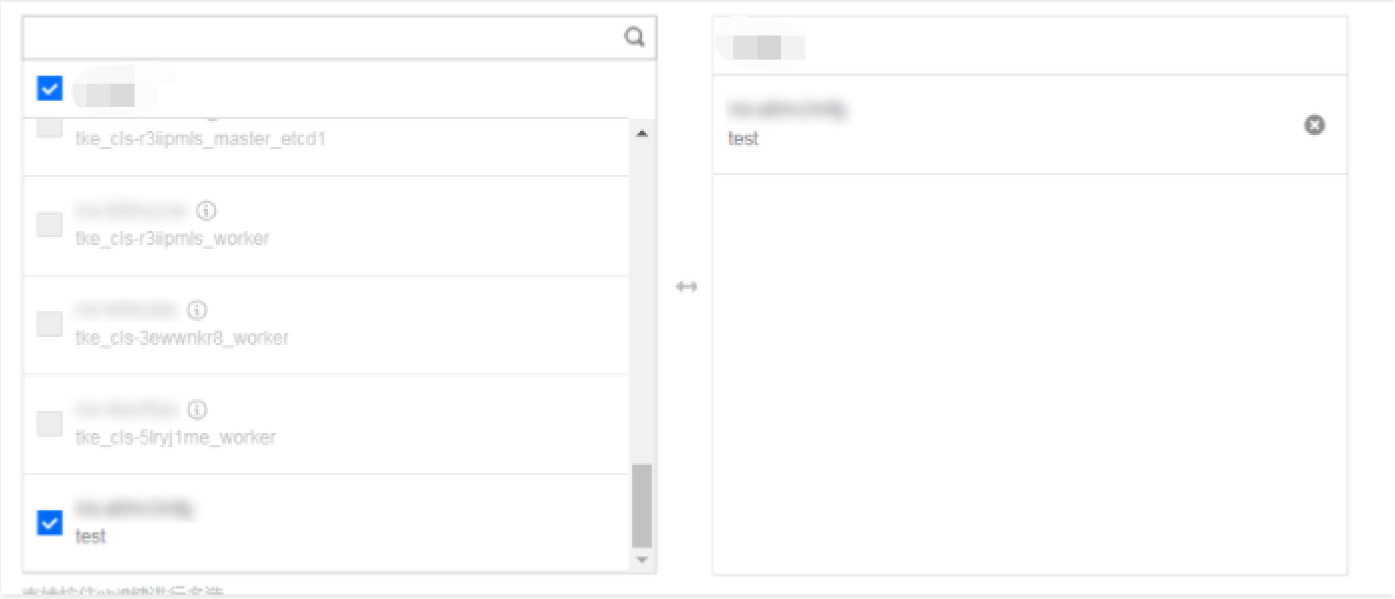
To use the MIG feature, make sure the following conditions are met:

- The GPU model is GT4.
- You have selected **Automatically install GPU driver on the backend** in the console and configured the GPU, CUDA, and cuDNN versions.

Adding existing GPU CVM instance

For detailed directions, see Adding a Node. When adding a node, you should pay attention to the following:

- On the **Select Node** page, select an existing GPU node as shown below:



☑

☐ tke_cls-r3iipmls_master_etcd1

☐ tke_cls-r3iipmls_worker

☐ tke_cls-3ewwnkr8_worker

☐ tke_cls-5lryj1me_worker

☑ test

test

- Configure the automatic installation of the GPU driver and MIG as needed.

Setting a Node Label

Last updated：2021-08-17 15:32:42

Operation Scenario

This document guides you through the process of setting a node Label.

Usage Restrictions

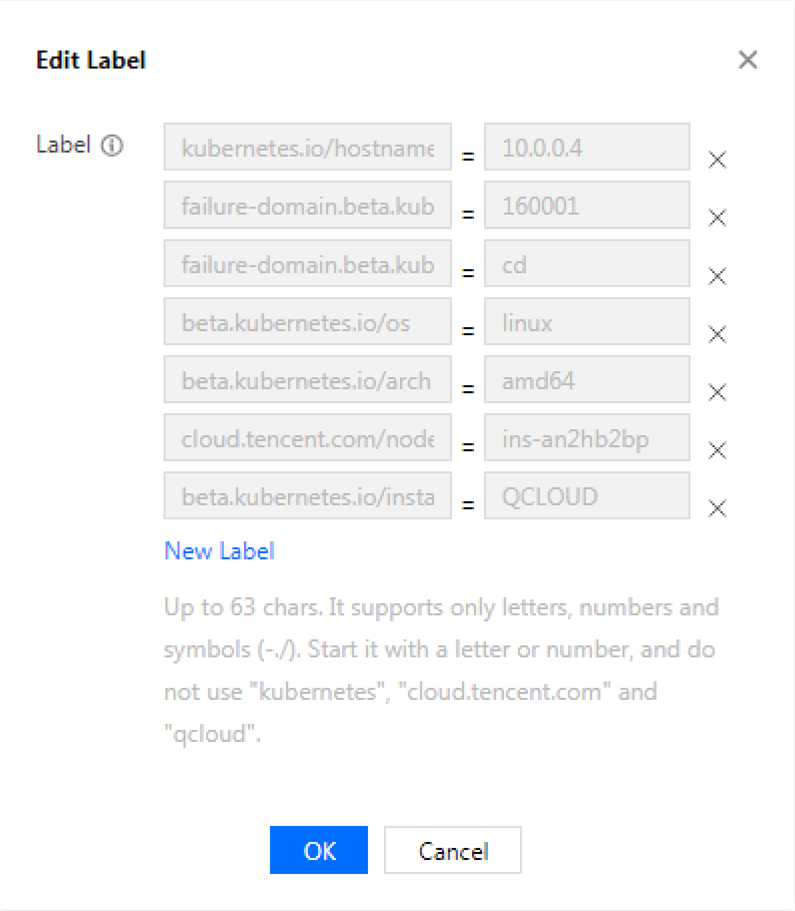
- Labels related to *kubernetes* and *qcloud* cannot be edited or deleted.
- *kubernetes* and *qcloud* labels are reserved keys and cannot be added.
- Currently, you can set Labels for one single node at a time, and batch setting is not supported.

Directions

- Setting a Node Label in the Console
- Using kubectl to Set a Node Label

1. Log in to the TKE console.
2. In the left sidebar, click **Clusters** to go to the cluster management page.
3. Select the ID/name of the cluster for which to set the node Label to go to the cluster details page.
4. In the left sidebar, select "Node Management" > "Nodes" to go to the "Node list" page.
5. Select the row of the node for which to set the Label and click **More** > **Edit a Label**.

6. In the "Edit a Label" window that pops up, edit the Label and click **Submit**. See the figure below:



Edit Label

Label

kubernetes.io/hostname	=	10.0.0.4
failure-domain.beta.kub	=	160001
failure-domain.beta.kub	=	cd
beta.kubernetes.io/os	=	linux
beta.kubernetes.io/arch	=	amd64
cloud.tencent.com/node	=	ins-an2hb2bp
beta.kubernetes.io/insta	=	QCLOUD

Up to 63 chars. It supports only letters, numbers and symbols (-./). Start it with a letter or number, and do not use "kubernetes", "cloud.tencent.com" and "qcloud".

Normal Node Management

Supported CVM Models for General Nodes

Last updated：2024-06-14 16:28:43

Background

To enhance the provision of container services, TKE conducts availability tests for container environments on the device models supported by general nodes. These tests primarily cover multiple application modules such as container network modes, storage, public images, node initialization, and GPU drivers. The following table lists the general node device models that can be created on the TKE console.

Note:

The range of device models supported by the TKE console for creating general nodes does not correspond directly with those of the CVM console. If your business requires adaptation to new device models, you may submit a service ticket.

Supported Device Models for General Nodes

Instance Type	Device Model
Standard type	S1、S2、S2ne、S3、S3ne、S4、S4m、S5、S5se、S5t、S6、S6t、SA1、SA2、SA2a、SA3、SK1、SN3ne、SR1、SW3a、SW3b、SW3ne、SA4、SA5、S8
Premium type	RS2t、RS3t、RS4t、RS5t
Computational type	C2、C3、C4、C5、C6、TC3、CN3
High-I/O type	I1、I2、I3、I6t、IT2、IT3、IT3a、IT3b、IT3c、IT5、ITA5
Memory type	M1、M2、M3、M4、M5、M6、M6ce、M6mp、M6p、MA2、MA3、MA4、MA5
High-performance type	HCCG5v、HCCIC5、HCCPNV4h、HCCTG5v、HCCPNV4sne（HCCPNV4sn）、HCCPNV5v、HCCPNV5vp、HCCPNV5、HCCPNV5x
GPU model	GI1、GI3X、GN10S、GN10X、GN10Xp、GN6、GN6S、GN7、GN8、GNV4、GT4、PNV4、PNV4ne、PNV5、GC49、PNV5b、PNV5i
Big data type	D1、D2、D3
Bare metal	BMD2、BMD3、BMD3c、BMD3s、BMDA2、BMI5、BMIA2、BMIA2m、BMM5r、

	BMS4、BMSA2、BMSC4、BMM6i、BMTGC39me（BMGC39me）、BMG5e、BMG5n、BMG5i、BMG5t、BMGY5、BMGNV4、BMSA3、BMIA3、BMS5
Others	CHC、CN10X、BC1

Note:

1. Before using Cloud Bare Metal (CBM), navigate to Instance Specification to confirm the device model configurations, such as whether the mounting of elastic NICs is supported and whether the mounting of cloud disks is supported.

1.1 For device models that do not support the mounting of elastic NICs, general nodes cannot be added to clusters in VPC-CNI network mode. In this case, you can use the GR mode.

1.2 For device models that do not support the mounting of cloud disks, PVCs cannot be bound to pods.

2. Regarding high-performance HCC models:

2.1 HCCG5v, HCCIC5, HCCPNV4h, and HCCTG5v only support public images of CentOS 7.6, Ubuntu 18.04.1, and TencentOS Server 2.4 (TK4).

2.2 HCCPNV5vp only supports custom images. Please submit a ticket and contact CVM after-sales support to provide them.

2.3 HCCPNV5 and HCCPNV5x only support public images of TencentOS Server 3.1 (TK4) UEFI. Please submit a ticket and contact CVM after-sales support to enable an allowlist.

3. ARM device models SR1 and SK1 only support images of TencentOS Server 2.4 for ARM 64 (TK4) and CentOS 8.2 (ARM).

4. Consumer-grade card models (such as GC49) require manual driver installation. You can specify the driver installation script when creating a node or use a pre-installed driver custom image; otherwise, the node initialization might fail due to undetected drivers.

5. Only the SA2, S5, C3, and C4 device models support Red Hat images.

6. The qGPU feature is only available for native nodes, currently supporting T4 and V100 card models. For details, see Using qGPU.

Node Pool Overview

Last updated：2023-05-06 19:41:07

Overview

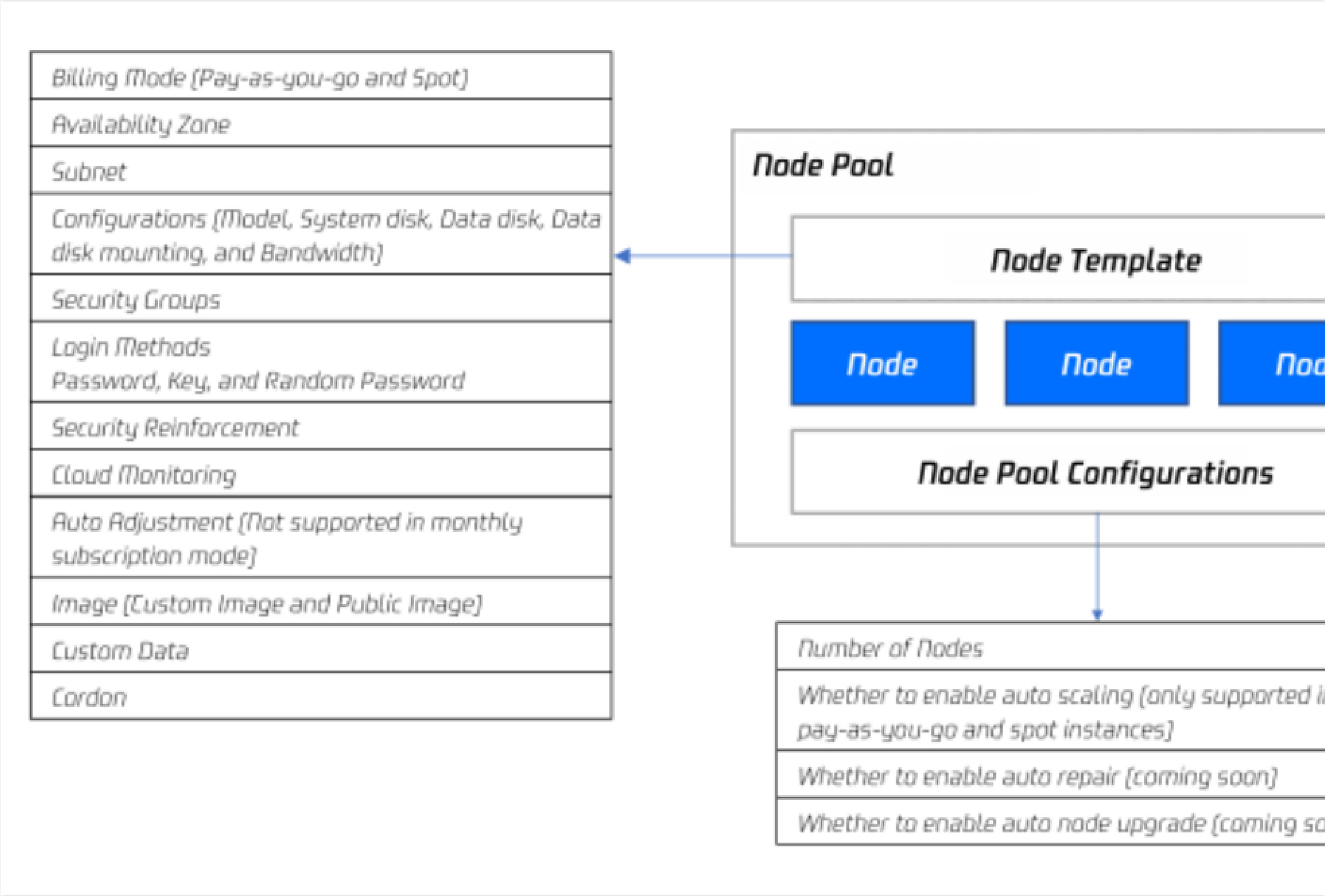
To help you efficiently manage nodes in a Kubernetes cluster, TKE introduced the concept of node pool. Basic node pool features allow you to conveniently and quickly create, manage, and terminate nodes and dynamically scale nodes in or out.

When a Pod in the cluster cannot be scheduled due to insufficient resources, scale-out will be triggered automatically, reducing labor costs.

When a scale-down condition such as node idleness is met, scale-down will be triggered automatically, which reduces the resource costs.

Product Architecture

The overall architecture of a node pool is as follows:


Billing Mode (Pay-as-you-go and Spot)
Availability Zone
Subnet
Configurations (Model, System disk, Data disk, Data disk mounting, and Bandwidth)
Security Groups
Login Methods
Password, Key, and Random Password
Security Reinforcement
Cloud Monitoring
Auto Adjustment (Not supported in monthly subscription mode)
Image (Custom Image and Public Image)
Custom Data
Cordon
Node Pool
Node Template
Node
Node
Noc
Node Pool Configurations
Number of Nodes
Whether to enable auto scaling (only supported i pay-as-you-go and spot instances)
Whether to enable auto repair (coming soon)
Whether to enable auto node upgrade (coming sc

Generally, all nodes in a node pool share the following attributes:

Node-level operating system

Billing type (currently pay-as-you-go and spot instances are supported)

CPU/memory/GPU

Launch parameters of Kubernetes components for nodes

Custom launch script for nodes

Kubernetes Label and Taints settings for nodes

In addition, TKE extends the following features for a node pool:

Supports managing node pool with CRD

Maximum number of Pods for each node in a specific node pool

Node-pool-level automatic repair and upgrade

Use Cases

When you need to use a large-scale cluster, we recommend that you use a node pool to manage nodes to improve the usability of the large-scale cluster. The following table describes multiple use cases for managing large-scale clusters and shows the effect of node pools in each use case.

Use Case	Effect
A cluster includes many heterogeneous nodes with different model configurations.	A node pool allows you to manage the nodes by groups in a unified manner.
A cluster needs to frequently scale nodes in and out.	A node pool improves OPS efficiency and reduces operating costs.
The scheduling rules for applications in a cluster are complex.	Node pool labels allow you to quickly specify service scheduling rules.
Routine maintenance is required for nodes in a cluster.	A node pool allows you to conveniently upgrade the Kubernetes version and the Docker version.

Concepts

TKE Auto Scaling is implemented based on Tencent Cloud AutoScaling and the cluster-autoscaler of the Kubernetes community. The relevant concepts are described as follows:
CA: cluster-autoscaler, an open-source component of the community, is mainly responsible for the auto scaling of the cluster.
AS: AutoScaling, the Tencent Cloud auto scaling service
ASG: AutoScaling Group, a specific scaling group (The node pool depends on the scaling group provided by the auto scaling service. A node pool corresponds to a scaling group. You only need to care about the node pool)
ASA: AS activity, a scaling activity
ASC: AS config, the AS launch configuration, namely the node template

Node Types in a Node Pool

To meet the requirements of different scenarios, the nodes in a node pool can be classified into the following two types:

Note:

It is not recommended to add existing nodes. If you do not have the permission to create nodes, you can add the existing nodes to scale out the cluster. However, some parameters of adding existing nodes may be inconsistent with the template of the node you defined, and the auto scaling cannot be performed.

Node Type	Node Source	Supporting Auto Scaling	Mode of Removal from Node Pool	Is Node Quantity Affected by **Adjust Node Number**
Nodes in the	Auto scale-out or	Yes	Auto scale-in or	Yes

scaling group	manual adjustment of the node quantity		manual adjustment of the node quantity	
Nodes outside the scaling group	Manually added to the node pool by users	No	Manually removed by users	No

How the Node Pool Auto Scaling Works

Please read how the node pool auto scaling works before using it.

How the node pool auto scale-out works

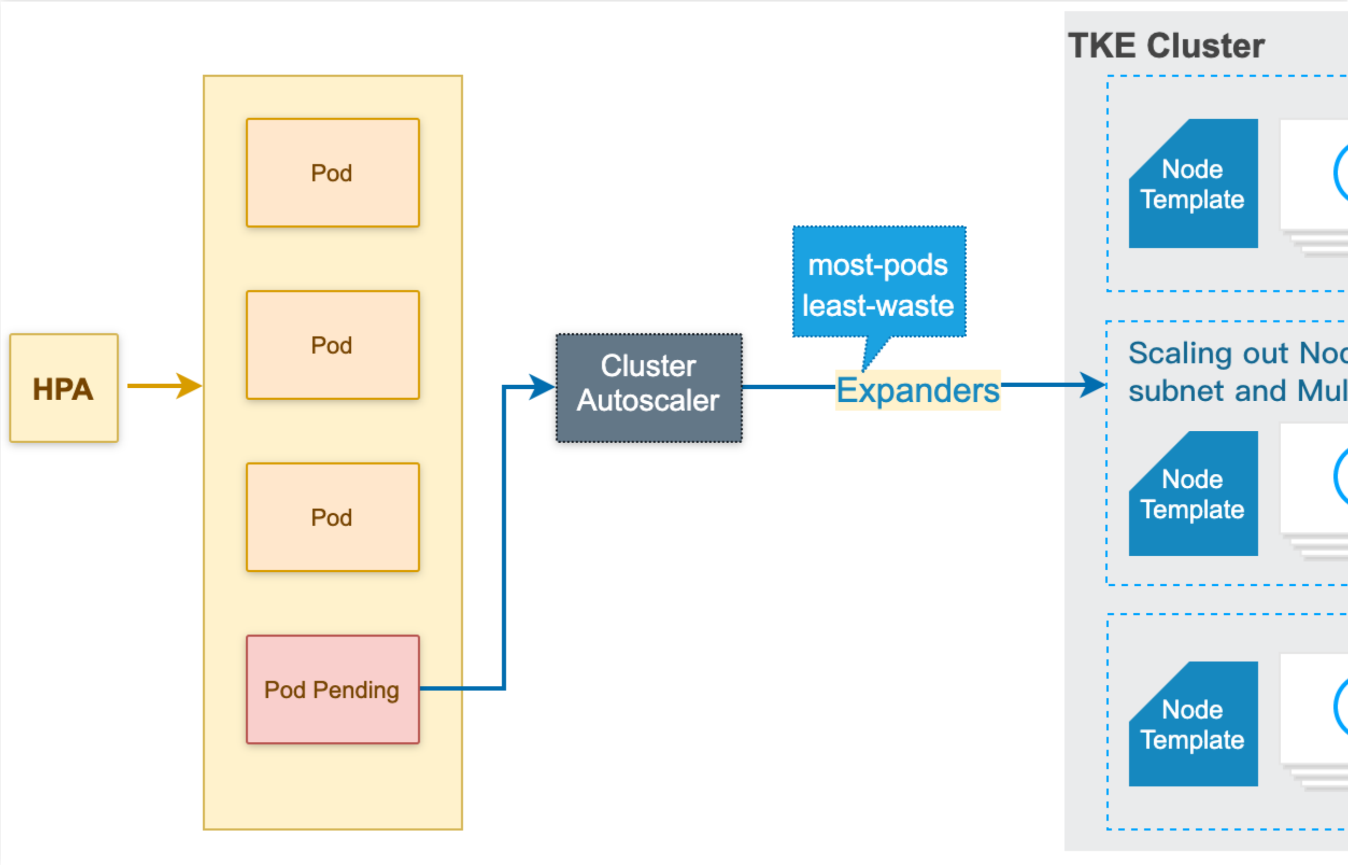
1. When the resources in the cluster are insufficient (the computing/storage/network resources of the cluster cannot meet the Pod request/affinity rules), the CA (Cluster Autoscaler) will detect the pending Pods due to the scheduling failure.
2. CA makes scheduling judgments based on the node template of each node pool, and selects the appropriate node template.
3. If there are multiple suitable templates, that is, multiple node pools are available for scale-out, CA will call **expanders** to select the optimal template from the multiple templates and scale out the corresponding node pool.
4. The specified node pool will be scaled out (based on the multi-subnet and multi-model policy), and two retry policies (set during the creation of the node pool) are provided. When the scale-out fails, it will retry based on the configured retry policies.

Note

The scale-out of a specific node pool is performed based on the subnet you set when creating the node pool and the subsequent multi-model configuration. Generally, **the multi-model policy shall prevail, and the next is the multi-zone/subnet policy.**

For example, if you configure multiple models A, B, multiple subnets 1, 2, 3, the scale-out is performed based on A1, A2, A3, B1, B2, and B3 in sequence. If A1 is sold out, the scale-out performed based on A2 instead of B1.

The following figure shows how the node pool auto scale-out works:



How the node pool auto scale-in works

1. Cluster Autoscaler (CA) detects that the allocation rate (value of `Request` , which takes the maximum value of CPU allocation rate and MEM allocation rate) is lower than the set node. When calculating the allocation rate, you can set the Daemonset type to not be included in the resources occupied by the Pod.

2. CA determines whether the scale-in can be triggered in the current cluster status. The following requirements must be met:

The node idle time is met (10 minutes by default).

The buffer time for cluster scale-out is met (10 minutes by default).

3. CA judges whether the node meets the scale-in conditions. You can set the **nodes not be scaled in** as needed (the nodes that meet the conditions will not be scaled in by CA).

Nodes with local storage

Nodes with Pods in kube-system namespace and not managed by DaemonSet

Note

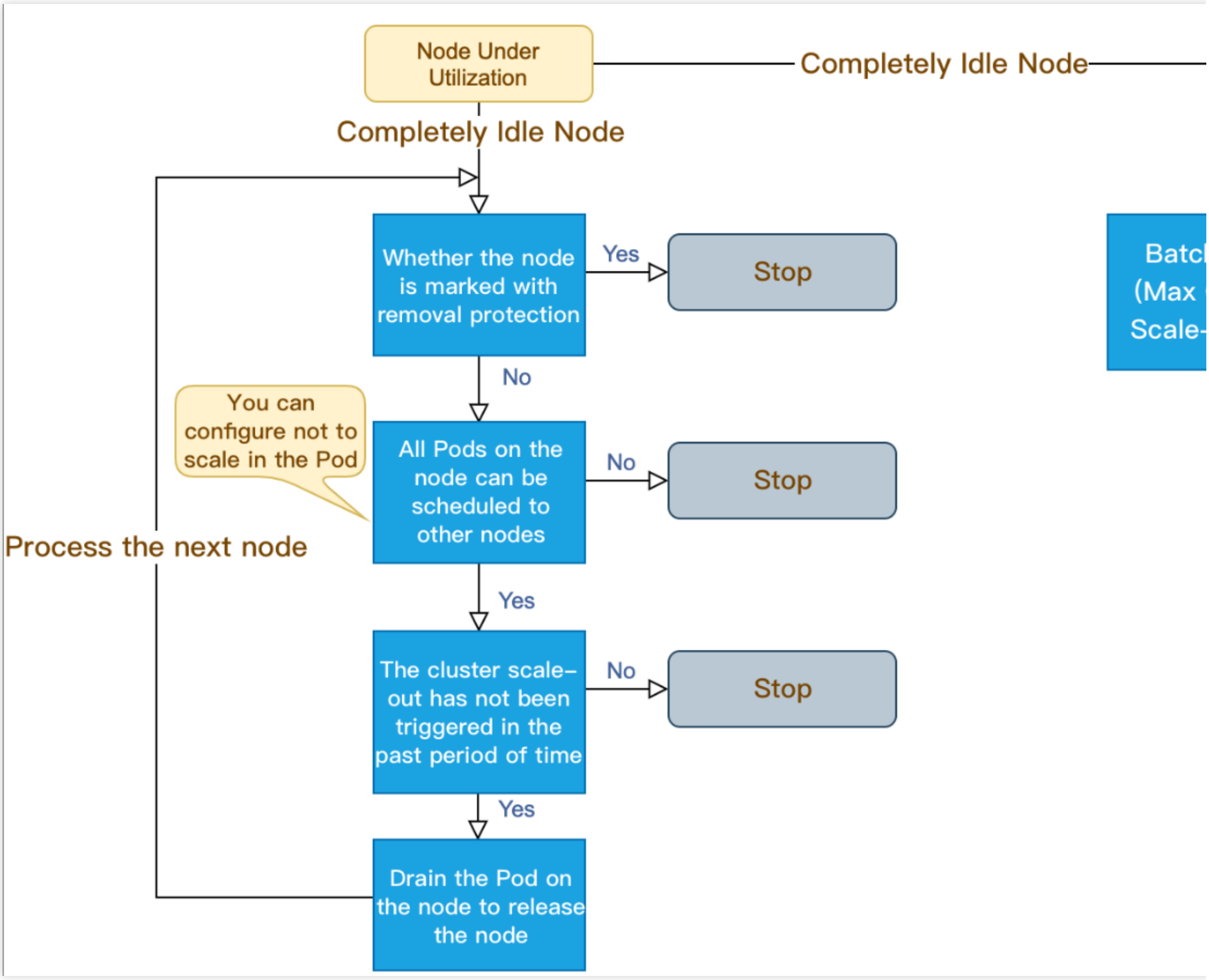
The nodes not be scaled in only take effect in cluster level. If you need more fine-grained protection of nodes from being scaled in, you can use the **removal protection** feature.

4. CA drains the Pod on the node, and then releases/shuts down the node.

The completely idle nodes can be scaled in concurrently. (You can set the max concurrent scale-in volume.)

The non-completely idle nodes are scaled in one by one.

The following figure shows how the node pool auto scale-in works:


Node Under Utilization
Completely Idle Node
Completely Idle Node
Whether the node is marked with removal protection
Yes
Stop
No
You can configure not to scale in the Pod
All Pods on the node can be scheduled to other nodes
No
Stop
Process the next node
Yes
The cluster scale-out has not been triggered in the past period of time
No
Stop
Yes
Drain the Pod on the node to release the node
Batc
(Max
Scale-

Features and Notes

Feature	Description	Remarks
Creating a Node Pool	Adds a node pool	We recommend that you create no more than 20 node pools for a single cluster.
Deleting a Node Pool	When you delete a node pool, you can select whether to terminate nodes in the node pool. No matter whether the node is terminated or not, the node will not be retained in the cluster.	When you delete a node pool, if you select to terminate nodes, the nodes will not be retained. You can create new nodes later if required.
Enabling auto scaling for a node pool	After you enable auto scaling for a node pool, the number of nodes in the node pool	Do not enable or disable auto scaling on the scaling group page in the console.

	is automatically adjusted according to the workload of the cluster.	
Disabling auto scaling for a node pool	After you disable auto scaling for a node pool, the number of nodes in the node pool is not automatically adjusted based on the workload of the cluster.	
Adjusting the number of nodes in a node pool	You can directly adjust the number of nodes in a node pool. If you decrease the number of nodes, the nodes in the scaling group are scaled in based on the node removal policy (the earliest node will be removed by default). Note: the scale-in is performed by the scaling group. TKE cannot detect the specific scale-in nodes and cannot drain or cordon in advance.	After you enable auto scaling, we recommend that you do not manually adjust the size of a node pool. Do not directly adjust the desired capacity of a scaling group in the console. Please scale in the node pool through auto scaling. During auto scaling, the node is first marked as unschedulable, all Pods on the node are drained or deleted, and then the node is released.
Adjusting a Node Pool	You can modify the node pool name, the operating system, the number of nodes in the scaling group, and the Kubernetes label and taint.	Modifications of the label and taint take effect for all the nodes in a node pool and may cause Pods to be rescheduled.
Adding an existing node	You can add Pods that do not belong to the cluster to the node pool. The following conditions are required: The Pods and the cluster belong to the same VPC. The Pods are not used by other clusters and have the same model and billing mode configurations as the node pool. You can add nodes in the cluster that do not belong to any node pool. It requires the node Pods and the node pool must be configured with the same model and billing mode.	In normal cases, we recommend that you directly create a node pool instead of adding an existing node to a node pool.
Removing a node from a node pool	You can remove any node from the node pool and you can choose to whether to retain the node in the cluster.	Do not add nodes to the scaling group in the console, which may result in data inconsistency.
Converting an existing scaling group into a node pool	You can convert an existing scaling group into a node pool. After the conversion, the node pool inherits all the features of the original scaling group, and the information	This operation is irreversible. Ensure that you are familiar with the features of node pools before conversion.

	about the scaling group will not be displayed. After all existing scaling groups in a cluster are converted into node pools, this feature will be disabled.	

Related Operations

You can log in to the Tencent Kubernetes Engine console and perform node pool-related operations according to the following documents:

Creating a Node Pool

Viewing a Node Pool

Adjusting a Node Pool

Deleting a Node Pool

Creating a Node Pool

Last updated：2023-06-01 15:26:20

Overview

This document describes how to create a node pool in a cluster via the TKE console and describes node pool-related operations, such as viewing, managing, and deleting a node pool.

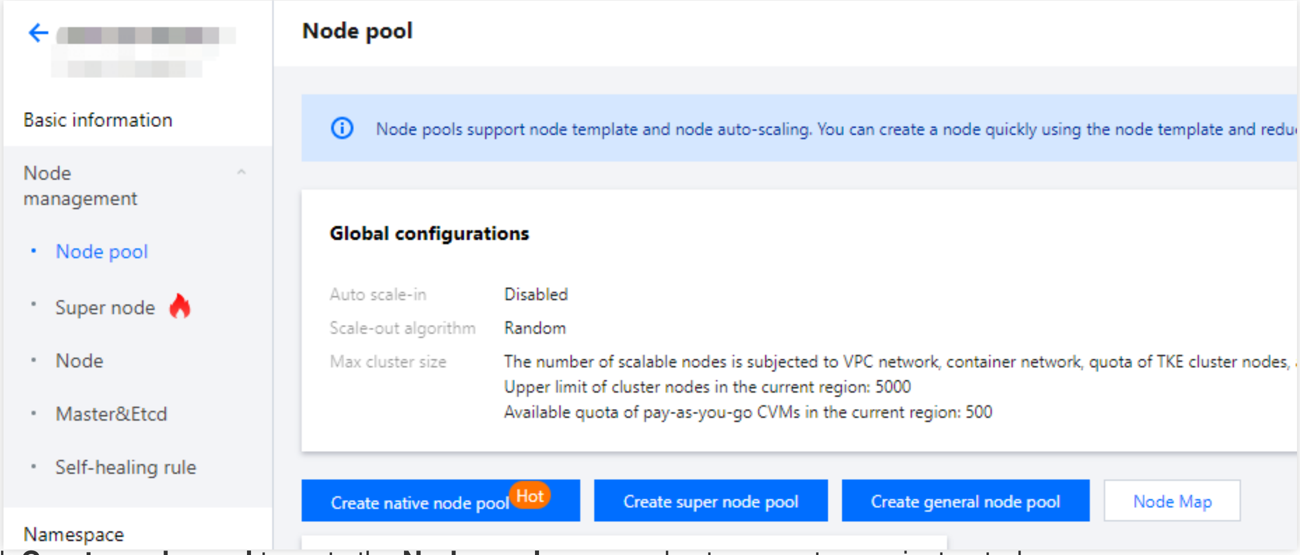
Prerequisites

You are familiar with the basic concepts of a node pool.

You have created a cluster as instructed in Creating a Cluster.

Directions

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. Click **Node management** > **Node pool** in the left sidebar to go to the node pool list page.



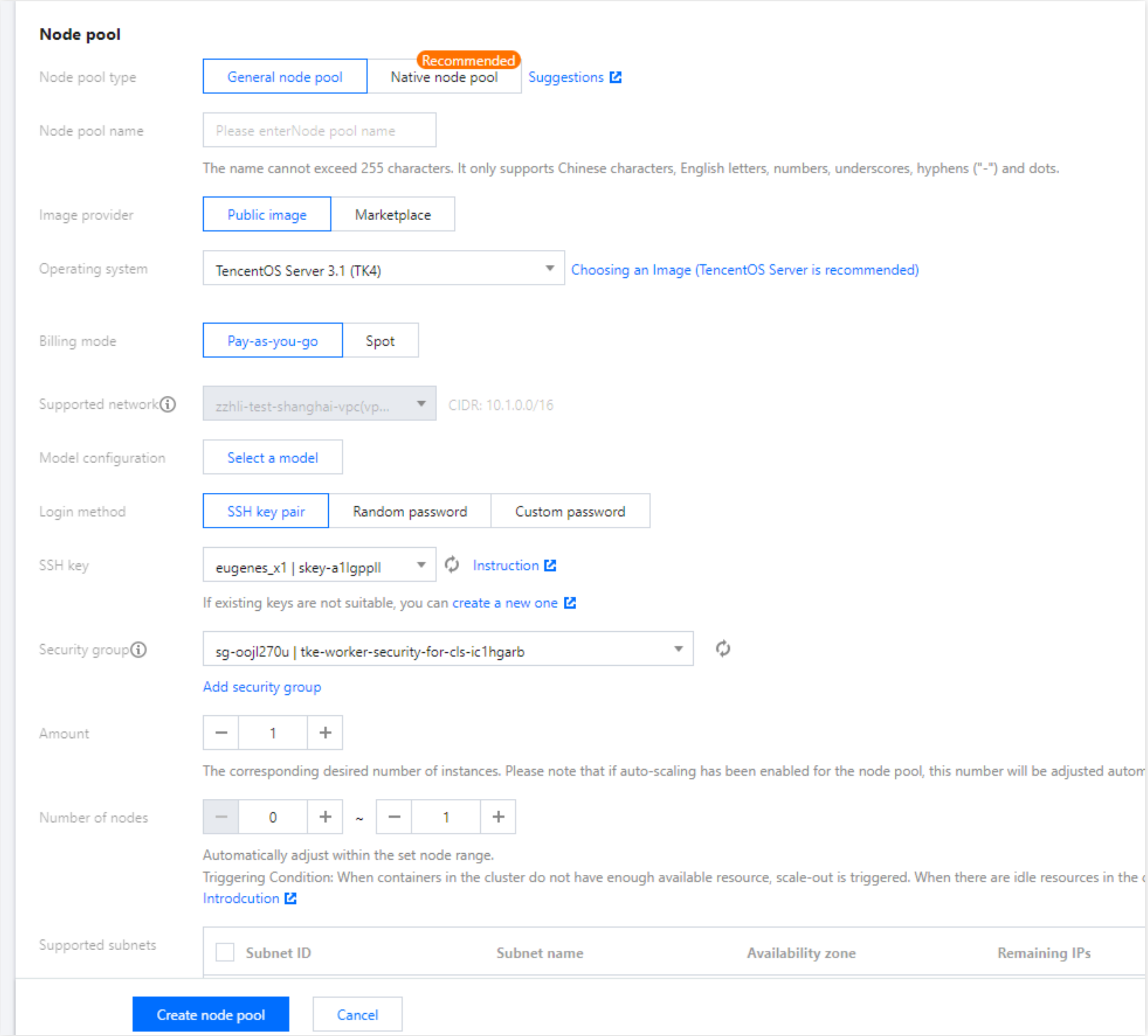
Node pool

Node pools support node template and node auto-scaling. You can create a node quickly using the node template and redu

Global configurations

Auto scale-in Disabled
Scale-out algorithm Random
Max cluster size The number of scalable nodes is subjected to VPC network, container network, quota of TKE cluster nodes, .
Upper limit of cluster nodes in the current region: 5000
Available quota of pay-as-you-go CVMs in the current region: 500

4. Click **Create node pool** to go to the **Node pool** page, and set parameters as instructed.



Node pool

Node pool type: General node pool | Native node pool (Recommended) | Suggestions

Node pool name: Please enterNode pool name

The name cannot exceed 255 characters. It only supports Chinese characters, English letters, numbers, underscores, hyphens ("-") and dots.

Image provider: Public image | Marketplace

Operating system: TencentOS Server 3.1 (TK4) Choosing an Image (TencentOS Server is recommended)

Billing mode: Pay-as-you-go | Spot

Supported network: zzhli-test-shanghai-vpc(vp... CIDR: 10.1.0.0/16

Model configuration: Select a model

Login method: SSH key pair | Random password | Custom password

SSH key: eugenes_x1 | skey-a1lgppll Instruction

If existing keys are not suitable, you can create a new one

Security group: sg-oojl270u | tke-worker-security-for-cls-ic1hgarb

Add security group

Amount: 1

The corresponding desired number of instances. Please note that if auto-scaling has been enabled for the node pool, this number will be adjusted autom

Number of nodes: 0 ~ 1

Automatically adjust within the set node range.

Triggering Condition: When containers in the cluster do not have enough available resource, scale-out is triggered. When there are idle resources in the c

Introdcution

Supported subnets:

☐ Subnet ID	Subnet name	Availability zone	Remaining IPs

Node pool name: Enter a custom pool name.

Operating system: Select an OS based on actual needs. This OS takes effect on the node pool level and can be modified. After modification, the new OS only take effect for the new nodes in the node pool, rather than the existing nodes.

Billing mode: Valid values include **Pay-as-you-go** and **Spot**. You can select a value as required. For more information, see Billing Mode.

Supported network: The system will assign IP addresses within the address range of the node network for servers in the cluster.

Note

This field is specified at the cluster level, which cannot be modified after configuration.

Model configuration: Click **Select a model**. On the **Model Configuration** page, select the values as needed based on the following descriptions:

Availability zone: Launch configurations do not contain availability zone information. This option is only used to filter available instance types in the availability zone.

Model: Select the model by specifying the number of CPU cores, memory size, and instance type. For more information, see Families and Models.

System disk: Controls the storage and schedules the operating of Cloud Virtual Machines (CVMs). You can view the system disk types available for the selected model and select the system disk as required. For more information, see Cloud Disk Types.

Data disk:Stores all the user data. You can specify the values according to the following descriptions. Each model corresponds to different data disk settings. For more information, see the following table:

Model	Data Disk Settings
Standard, Memory Optimized, Computing, and GPU	No option is selected by default. If you select any of these options, you must specify the cloud disk settings and formatting settings.
High I/O and Big Data	These options are selected by default and cannot be cleared. You can customize the formatting settings for the default local disks.
Batch-based	This option is selected by default, but can be cleared. If this option is selected, you can purchase only default local disks. You can customize the formatting settings for the default local disks.

Add Data Disk (optional): click Add Data Disk and specify the settings according to the table above.

Public Bandwidth: **Assign free public IP** is selected by default, indicating that the system will assign a public IP for free. Billing by traffic or by bandwidth can be selected as needed. For billing details, see Public Network Billing. You can customize the network speed.

Login method: Select any one of the following login methods as required:

SSH key pair: A key pair is a pair of parameters generated by an algorithm. It is a way to log in to a CVM instance that is more secure than regular passwords. For more details, see SSH Key.

SSH key: This parameter displays only when **SSH Key Pair** is selected. Select an existing key in the drop-down list. For how to create a key, see Creating an SSH key.

Random password: The system sends an automatically generated password to your Message Center.

Custom password: Set a password as prompted.

Security group: The default value is the security group specified when the cluster is created. You can replace the security group or add a security group as required.

Amount: Specify the desired capacity as needed.

Note

If auto scaling has been enabled for the node pool, this quantity will be automatically adjusted based on the loads of the cluster.

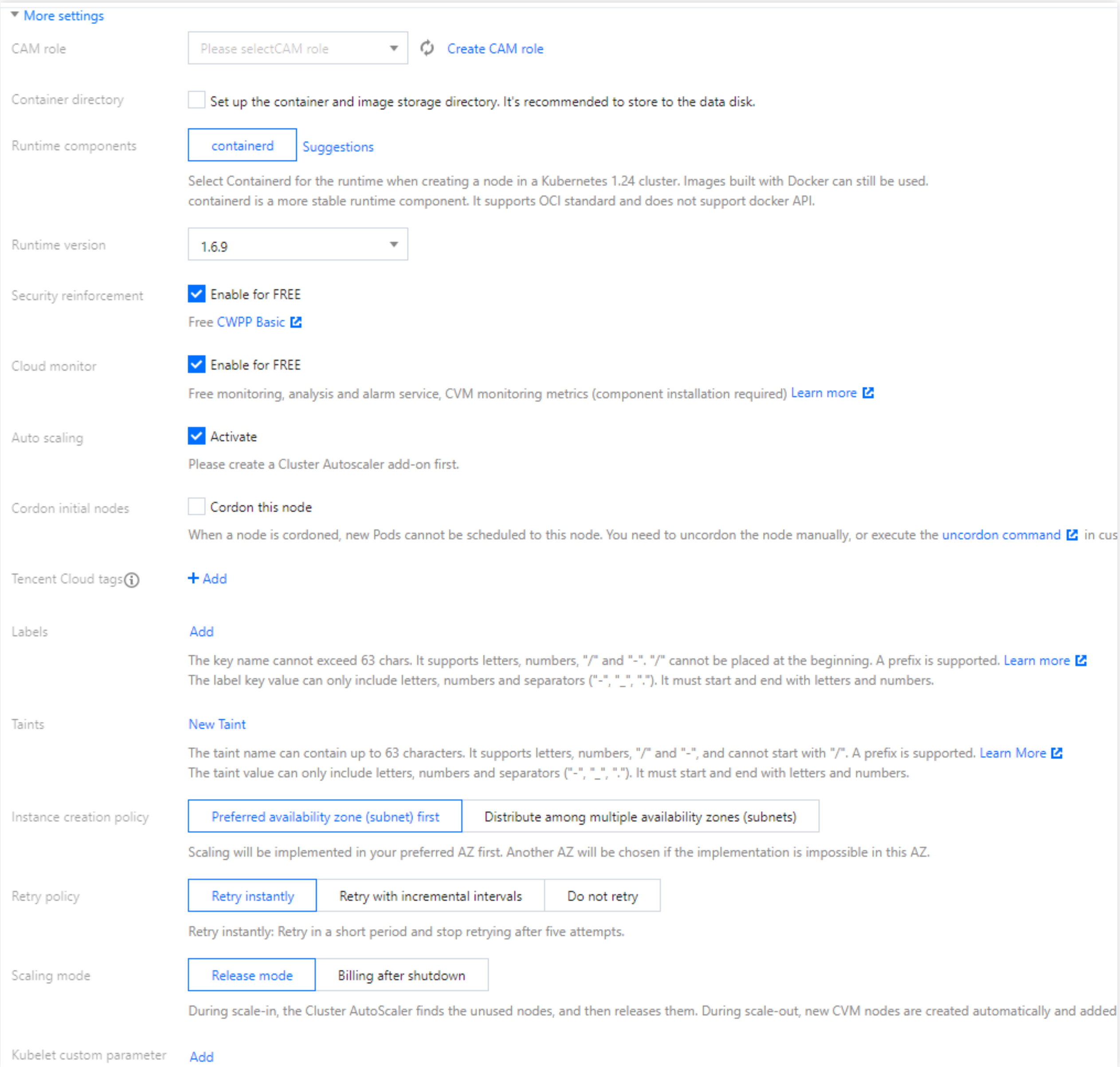
Number of nodes: The number of nodes will be automatically adjusted within the specified node quantity range, which will not exceed the specified range.

Supported subnets: Select an available subnet as needed.

Note

The default multiple subnets scale-out policy of node pool is that if you have configured multiple subnets, when the node pool performs scale-out (manual scale-out and auto scaling), it creates nodes based on the priority determined by the order in the subnet list. If a node can be successfully created in the subnet of the highest priority, all nodes will be created in the subnet.

(Optional) Click **More Settings** to view or configure more information.



▾ More settings

CAM role: Please selectCAM role ▾ | Create CAM role

Container directory: □ Set up the container and image storage directory. It's recommended to store to the data disk.

Runtime components: containerd | Suggestions
Select Containerd for the runtime when creating a node in a Kubernetes 1.24 cluster. Images built with Docker can still be used.
containerd is a more stable runtime component. It supports OCI standard and does not support docker API.

Runtime version: 1.6.9 ▾

Security reinforcement: ☑ Enable for FREE
Free CWPP Basic

Cloud monitor: ☑ Enable for FREE
Free monitoring, analysis and alarm service, CVM monitoring metrics (component installation required) Learn more

Auto scaling: ☑ Activate
Please create a Cluster Autoscaler add-on first.

Cordon initial nodes: □ Cordon this node
When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually, or execute the uncordon command in cus

Tencent Cloud tags: + Add

Labels: Add
The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Taints: New Taint
The taint name can contain up to 63 characters. It supports letters, numbers, "/" and "-", and cannot start with "/". A prefix is supported. Learn More
The taint value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Instance creation policy: Preferred availability zone (subnet) first | Distribute among multiple availability zones (subnets)
Scaling will be implemented in your preferred AZ first. Another AZ will be chosen if the implementation is impossible in this AZ.

Retry policy: Retry instantly | Retry with incremental intervals | Do not retry
Retry instantly: Retry in a short period and stop retrying after five attempts.

Scaling mode: Release mode | Billing after shutdown
During scale-in, the Cluster AutoScaler finds the unused nodes, and then releases them. During scale-out, new CVM nodes are created automatically and added

Kubelet custom parameter: Add

CAM role: You can bind all the nodes of the node pool to the same CAM role, and grant the authorization policy bound to the role to the nodes. For more information, see Managing Roles.

Container directory: Select this option to set up the container and image storage directory. We recommend that you store to the data disk, such as `/var/lib/docker` .

Security Services: Free DDoS, Web Application Firewall (WAF) and Cloud Workload Protection (CWP) are activated by default. For more information, see Cloud Workload Protection.

Cloud monitor: Free monitoring, analysis, and alarms are activated by default, and components are installed to obtain CVM monitoring metrics. For more information, see Tencent Cloud Observability Platform.

Auto scaling: **Enable** is selected by default.

Cordon initial nodes: After you check **Cordon this node**, new Pods cannot be scheduled to this node. You can uncordon the node manually, or execute the uncordon command in custom data as needed.

Labels: Click **Add** and customize the settings of the label. The specified label here will be automatically added to nodes created in the node pool to help filter and manage the nodes using the label.

Taints: This is a node-level attribute and is usually used with `Tolerations` . You can specify this parameter for all the nodes in the node pool, so as to stop scheduling Pods that do not meet the requirements to these nodes and drain such Pods from the nodes.

Note

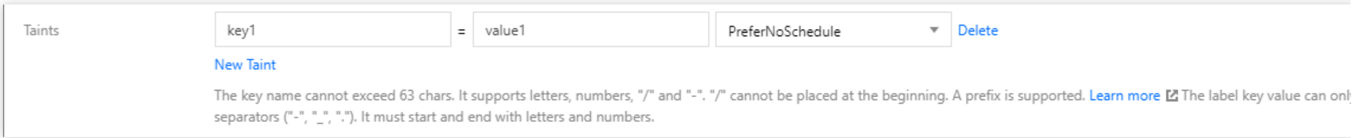
The value of **Taints** usually consists of `key` , `value` , and `effect` . Valid values of `effect` are as follows:

PreferNoSchedule: Optional condition. Try not to schedule a Pod to a node with a taint that cannot be tolerated by the Pod.

NoSchedule: When a node contains a taint, a Pod without the corresponding toleration must not be scheduled.

NoExecute: When a node contains a taint, a Pod without the corresponding toleration to the taint will not be scheduled to the node and any such Pods already on the node will be drained.

Assume that **Taints** is set to `key1=value1:PreferNoSchedule` , the configuration in the TKE console is as below:



Taints key1 = value1 PreferNoSchedule

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more The label key value can onl separators ("-", "_", "."). It must start and end with letters and numbers.

Retry policy: Select either of the following policies as needed.

Retry instantly: Retry immediately. The system stops retrying after failing five times in a row.

Retry with incremental intervals: The retry interval extends as the number of consecutive failures increases. The value ranges from seconds to one day.

Scaling mode: select either of the following two scaling modes as needed.

Release mode: If this mode is selected, the system automatically releases idle nodes as determined by Cluster AutoScaler during scale-in and automatically creates and adds a node to scaling groups during scale-out.

Billing after shutdown: If this mode is selected, during scale-out, the system preferably starts nodes that have been shut down, and if the number of nodes still fails to meet requirements, the system creates the desired number of nodes. During scale-in, the system shuts down idle nodes. If the nodes support the No Charges When Shut Down feature, the nodes that are shut down will not be billed, but remaining nodes are still billed. For more information, see No Charges When Shut down for Pay-as-You-Go Instances Details.

Custom data: Specify custom data to configure the node, that is, to run the configured script when the node is started up. You need to ensure the reentrant and retry logic of the script. The script and its log files can be viewed at the node path: `/usr/local/qcloud/tke/userscript` .

5. Click **Create Node Pool**.

Related Operations

After a node pool is created, you can manage the node pool according to the following documents:

Viewing a Node Pool

Adjusting a Node Pool

Deleting a Node Pool

Viewing a Node Pool

Last updated：2022-06-14 15:19:11

Introduction

This document describes how to view an existing node pool of a cluster and query the details of the node pool via the TKE console for subsequent node pool management.

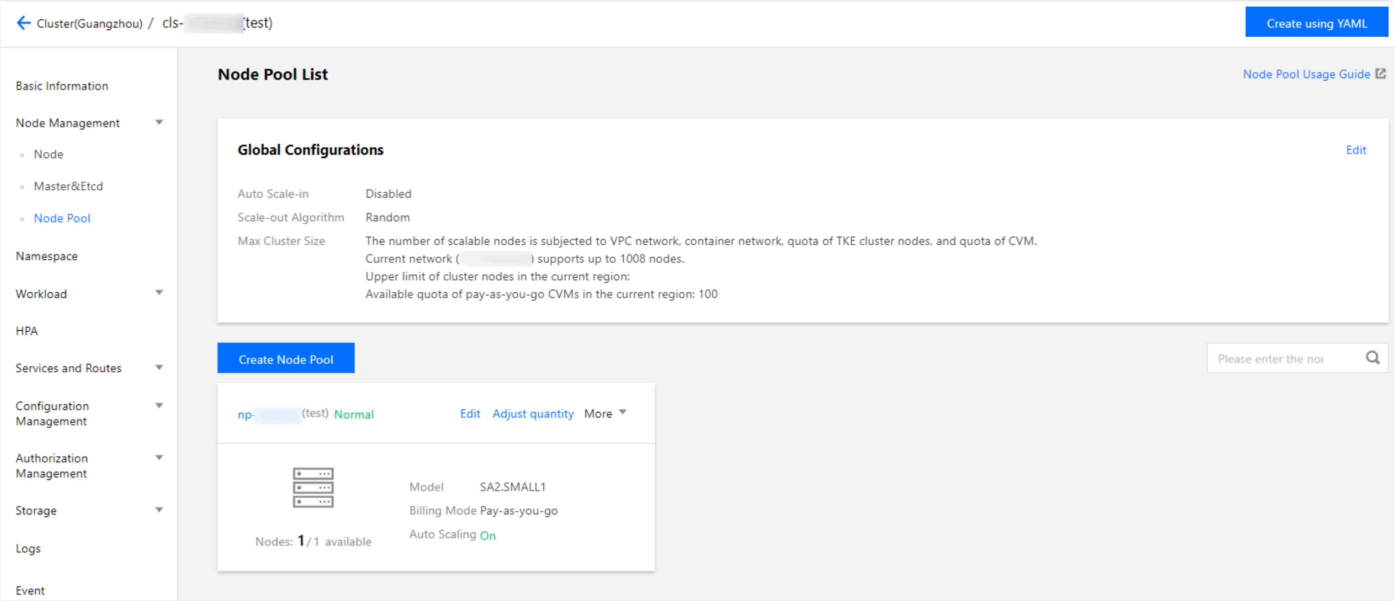
Prerequisites

You have created a node pool for the cluster. For more information on how to create a node pool, see Creating a Node Pool.

Directions

Viewing the node pool list page

1. Log in to the Tencent Kubernetes Engine console and click **Cluster** in the left sidebar.
2. On the “Cluster Management” page, click the desired cluster ID to open the “Deployment” page.
3. On the left sidebar, choose **Node Management** -> **Node Pool** to open the “Node pool list” page. You can view existing node pools and global node pool configurations, as shown in the following figure.



Node Pool List

Global Configurations

Edit

Auto Scale-in: Disabled
Scale-out Algorithm: Random
Max Cluster Size: The number of scalable nodes is subjected to VPC network, container network, quota of TKE cluster nodes, and quota of CVM.
Current network ([redacted]) supports up to 1008 nodes.
Upper limit of cluster nodes in the current region:
Available quota of pay-as-you-go CVMs in the current region: 100

np-[redacted](test) Normal

Nodes: 1/1 available
Model: SA2.SMALL1
Billing Mode: Pay-as-you-go
Auto Scaling: On

Node pool information and configurations are as follows:

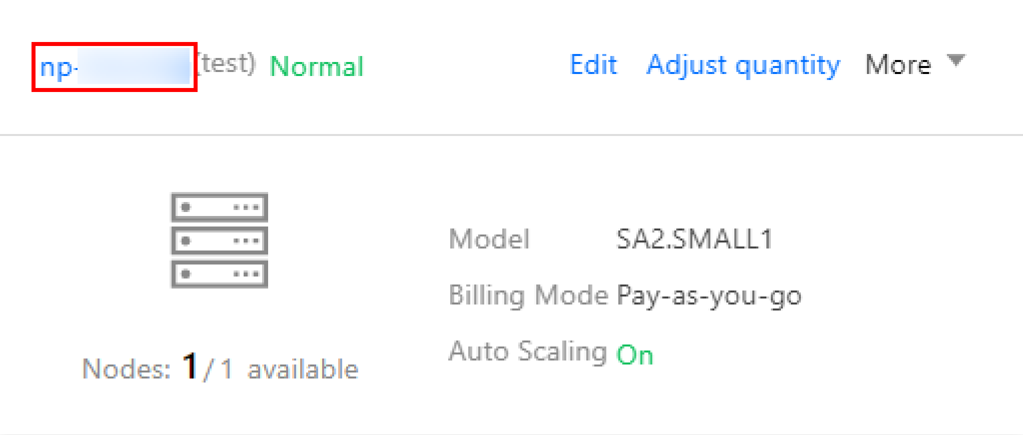
- **Global Configuration**: the common configuration items for all node pools of the cluster. You can click **Edit** in the upper-right corner of the module to modify the values. For more information, see Adjusting Global Node Pool Configurations.
  - **Auto scale-down**: this feature is disabled in this example. If this feature is enabled, auto scale-down is triggered when a large amount of node resources in the cluster is idle. For more information, see Cluster Scaling.
  - **Scale-up Algorithm**: the value is “Random” by default in this example, indicating that a scaling group in the node pool is randomly selected for scale-up. TKE further supports the following scale-up algorithms and you can modify the value as required:
    - **most-pods**: scales up the scaling group that can schedule the most Pods.
    - **least-waste**: scales up the scaling group that has the fewest remaining resources after Pod scheduling.
  - **Max cluster size**: displays the size information of the current cluster. When you adjust the number of existing node pools or recreate a node pool, note the size limit specified by this parameter and properly specify the number of nodes in a node pool.
- **Node pool card page**: the node pool sorting area is located under the Global Configuration area. Each node pool is displayed as a card that includes the following information:

> Note：
> When many node pools are displayed, you can enter the node pool ID or node pool name in the search box in the upper-right corner of the area to filter node pools.

- **Node pool ID (which is the name of the node pool)**: the value is np-***(test) in this example. Click the ID to open the details page of the node pool. You can view more information about the node pool on the page.
- **Node pool status**: the value is “Normal” in this example, indicating that the node pool is in the normal state.
- **Node pool operation**: the operations include **Edit**, **Adjust Number**, and **More**. For more information, see Adjusting Node Pool Configurations.
- **Number of available nodes/Total number of nodes in the node pool**: the value is “1 available/1 in total” in this example.
- **Model**: the model of all the nodes in the node pool.
- **Billing Mode**: the billing mode of all the nodes in the node pool. The value is “Pay-as-you-go” in this example, indicating that the instance is billed based on actual usage. For more information on billing, see Payment Modes.
- **Auto scaling**: the value is “On” in this example.

Viewing a single node pool

1. On the “Node pool card” page, click the desired node pool ID, as shown in the following figure.



np-[redacted](test) Normal

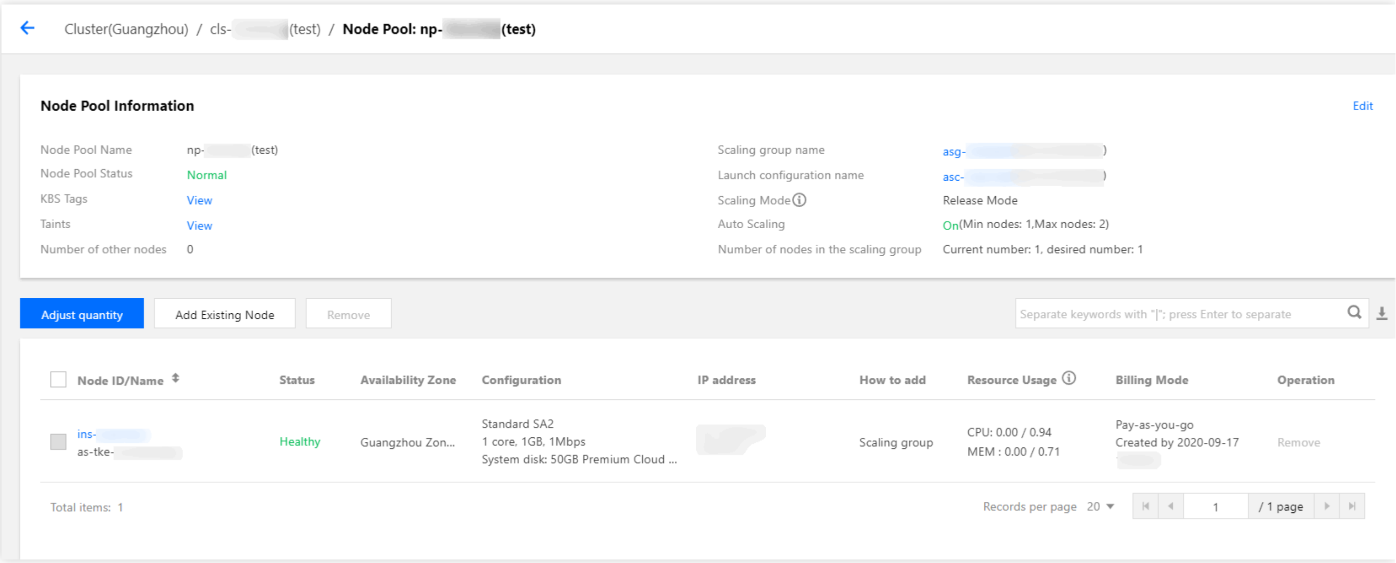
Model SA2.SMALL1

Billing Mode Pay-as-you-go

Auto Scaling On

Nodes: 1 / 1 available

2. On the details page of the node pool, you can view more basic information and node information of the node pool, as shown in the following figure.



Cluster(Guangzhou) / cls-[redacted](test) / **Node Pool: np-[redacted](test)**

Node Pool Information

Edit

Node Pool Name	np-[redacted](test)	Scaling group name	asg-[redacted])
Node Pool Status	Normal	Launch configuration name	asc-[redacted])
KBS Tags	View	Scaling Mode ⓘ	Release Mode
Taints	View	Auto Scaling	On(Min nodes: 1,Max nodes: 2)
Number of other nodes	0	Number of nodes in the scaling group	Current number: 1, desired number: 1

□ Node ID/Name	Status	Availability Zone	Configuration	IP address	How to add	Resource Usage ⓘ	Billing Mode	Operation
□ ins-[redacted] as-tke-[redacted]	Healthy	Guangzhou Zon...	Standard SA2 1 core, 1GB, 1Mbps System disk: 50GB Premium Cloud ...	[redacted]	Scaling group	CPU: 0.00 / 0.94 MEM : 0.00 / 0.71	Pay-as-you-go Created by 2020-09-17 [redacted]	Remove

Total items: 1

Relevant Operations

For more information on the features of node pools, see the following documents:

- Creating a Node Pool
- Adjusting a Node Pool
- Deleting a Node Pool

Adjusting a Node Pool

Last updated：2022-03-23 18:17:29

Overview

This document describes how to adjust the configuration of a node pool through the TKE console, including adjusting the global node pool configurations, configurations of the node pool, and the number of nodes in the node pool, enabling/disabling auto scaling, and setting removal protection for a node.

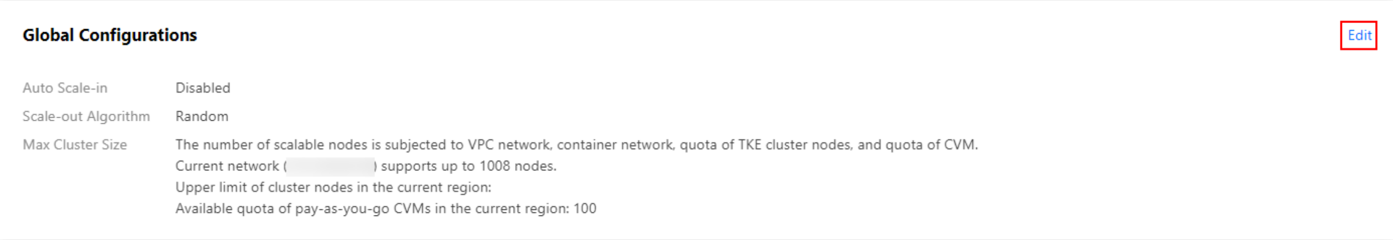
Prerequisites

- You have created an available node pool. For more information, please see Creating a Node Pool.
- You have opened the Node pool list page. For more information, see Viewing a Node Pool.

Directions

Adjusting the global configuration of node pool

1. On the “Node Pool List” page, click **Edit** in the top-right corner of the **Global Configuration** module as shown below:



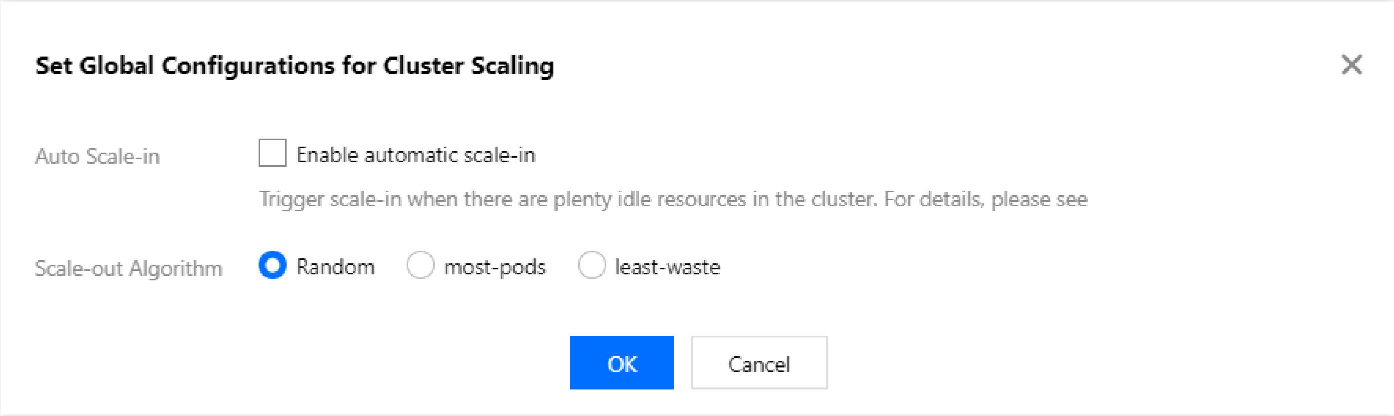
Global Configurations

Auto Scale-in Disabled

Scale-out Algorithm Random

Max Cluster Size The number of scalable nodes is subjected to VPC network, container network, quota of TKE cluster nodes, and quota of CVM.
Current network ([redacted]) supports up to 1008 nodes.
Upper limit of cluster nodes in the current region:
Available quota of pay-as-you-go CVMs in the current region: 100

2. In the “Set Cluster Scaling Global Configuration” window, specify the configuration settings according to the following descriptions.



Set Global Configurations for Cluster Scaling

Auto Scale-in □ Enable automatic scale-in

Trigger scale-in when there are plenty idle resources in the cluster. For details, please see

Scale-out Algorithm ● Random ○ most-pods ○ least-waste

The main parameters are described as follows:

- **Auto Scale-down**: this option is not selected by default. If this option is selected, auto scale-in is triggered when a large amount of node resources in the cluster is idle. For more information, see Cluster Scaling.
- **Scale-down Configuration**: this configuration item appears only when Auto Scale-down is selected. Specify the configuration settings as required.
    - **Max Concurrent Scale-in Volume**: this parameter indicates the maximum number of nodes that can be scaled in concurrently. The default value is 10. You can specify the value as required.

    > Note：
    > Only completely idle and empty nodes are removed here. If any node in the node pool contains a Pod, a maximum of one node will be removed each time.

    - **When Occupied Resources/Allocable Resources Are Smaller Than**: you can specify this value to trigger a check of scale-in conditions when the ratio of **Resources occupied by Pods or allocable resources** is less than the value. Value range: 0 to 80.
    - **Node Idle Time Threshold**: you can specify this parameter to terminate a node when the continuous idle time of a node exceeds the specified value by several minutes.
    - **Rechecking Scale-up Conditions**: this parameter allows you to specify the time when the cluster first checks the scale-up conditions.
    - **Nodes Not to Be Scaled Down**: select the following configuration items as required to ensure that the following types of nodes will not be scaled in:
        - Nodes with locally stored Pods
        - Nodes with Pods that are located in the kube-system namespace and are not managed by DaemonSet
- **Scale-up Algorithm**: the algorithm based on which the cluster is scaled up. Valid values:
    - **Random**: randomly scales up a node pool when there are multiple node pools.

- **most-pods**: scales up the node pool that can schedule the most Pods when there are multiple node pools.
- **least-waste**: scales up the node pool that can ensure the fewest remaining resources after Pod scheduling when there are multiple node pools.

3. Click **OK**.

Adjusting configurations of a node pool

Adjusting node pool operating system, alternative model, and container runtime

1. On the **Node Pool List** page, click a node pool ID to enter the node pool details page.
2. On the node pool basic information page, you can modify the node pool attributes.

Show All

Operating system

展开&收起

You can click ✎ on the right of the operating system to change the node pool operating system.
The change of operating system takes effect only for newly added, reinstalled, and upgraded nodes but not running nodes.

Model

展开&收起

You can click ✎ on the right of **Model** to change the alternative models of the node pool (the primary model cannot be changed). Setting alternative models can effectively reduce the risks of scale-out failure when the primary models are sold out.

- The order of alternative models corresponds to the priority order of the models. Set the model order as needed and confirm it at the bottom of the pop-up window.
- The alternative models must have the same specification (CPU, memory, and CPU architecture) as the primary model.
- You can select up to 10 models (including the primary model) in the same node pool. Plan the models as needed.

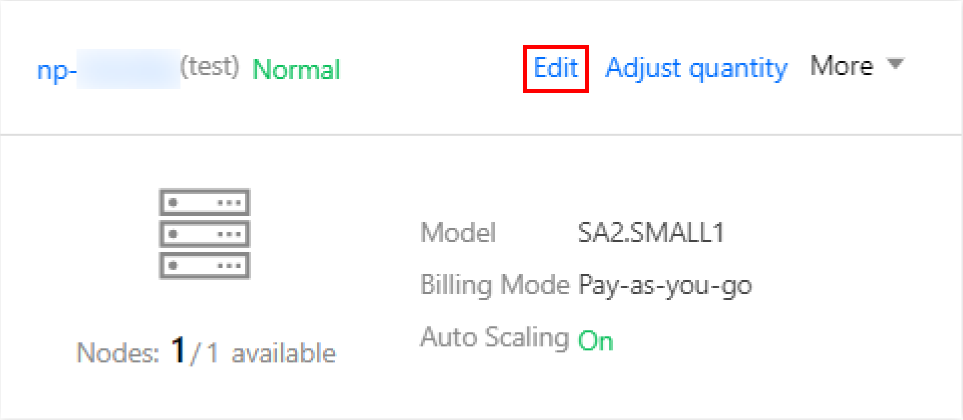
Runtime component

展开&收起

Click ✎ on the right of **Runtime Component** to change the runtime component and version of the node pool. For more information, see How to Choose Containerd and Docker.

Adjusting the number of nodes, label, and taints

1. On the card page of the target node pool, click **Edit** in the top-right corner.



np-[redacted](test) Normal

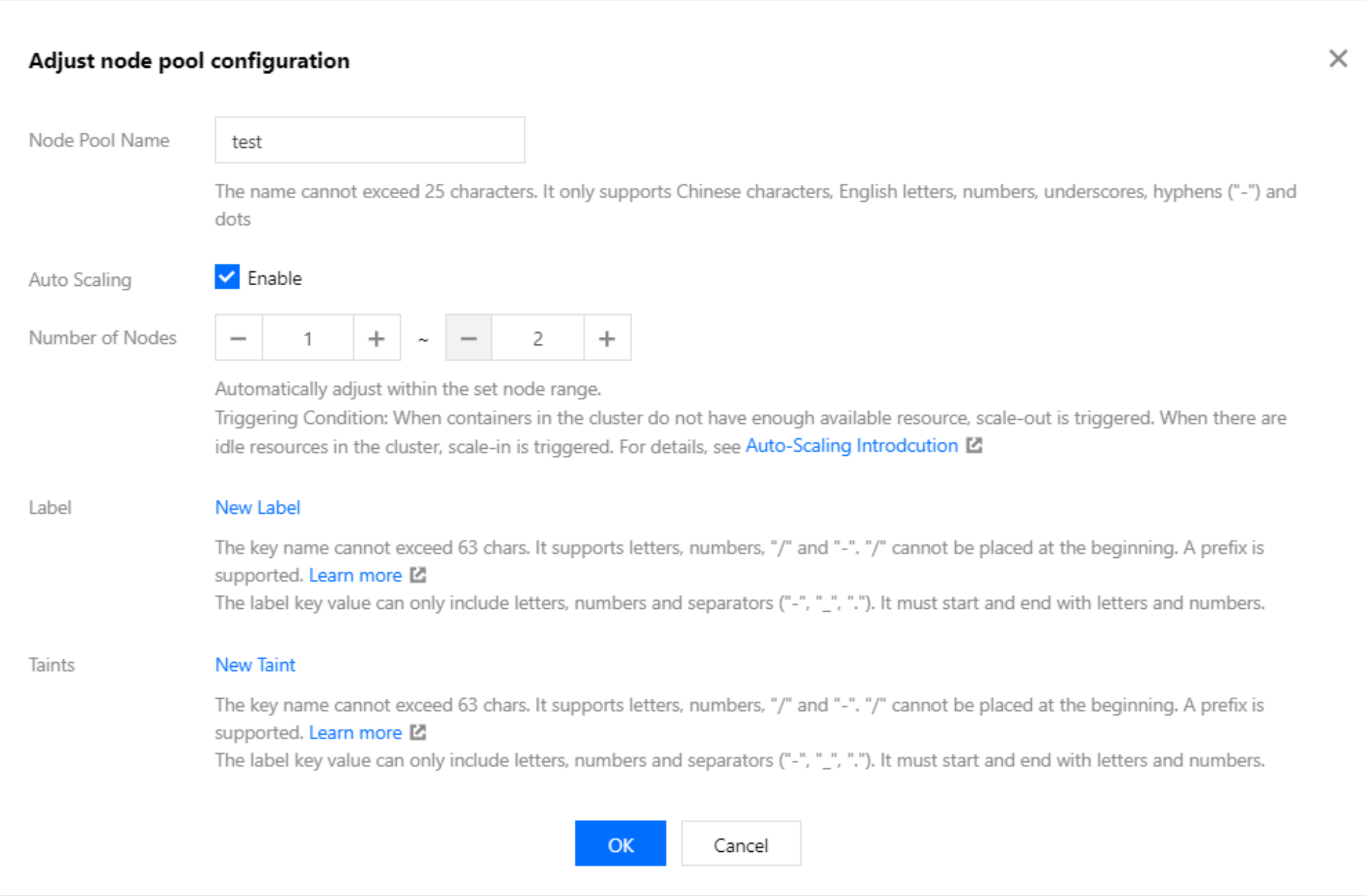
Model SA2.SMALL1

Billing Mode Pay-as-you-go

Auto Scaling On

Nodes: 1 / 1 available

2. On the “Adjust node pool configurations” page, specify the configuration settings according to the following descriptions.



Adjust node pool configuration

Node Pool Name: test

The name cannot exceed 25 characters. It only supports Chinese characters, English letters, numbers, underscores, hyphens ("-") and dots

Auto Scaling: ☑ Enable

Number of Nodes: 1 ~ 2

Automatically adjust within the set node range.
Triggering Condition: When containers in the cluster do not have enough available resource, scale-out is triggered. When there are idle resources in the cluster, scale-in is triggered. For details, see Auto-Scaling Introdcution

Label: New Label

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Taints: New Taint

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

- **Node pool name**: you can customize the name of the node pool based on service requirements to facilitate subsequent resource management.
- **Auto scaling**: you can select this option as required.
- **Node Quantity Range**: the number of nodes will be automatically adjusted within the specified node quantity range, which will not exceed the specified range.

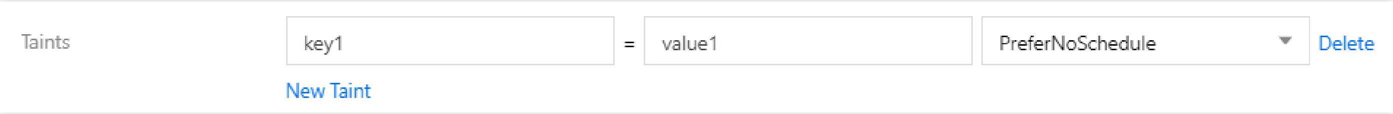
> Note：
> This number range affects the operation of Adjusting the number of nodes in a node pool. For example, when the number of nodes in the current node pool has reached the maximum value within the range, the number of nodes cannot be increased again.

- **Label**: the specified label here will be automatically added to nodes created in the node pool to help filter and manage nodes using labels. Click **New Label** to customize the label.
- **Taints**: the attributes of the node. This parameter is usually used with `Tolerations` . You can specify this parameter for all the nodes of the node pool so that Pods that do not meet conditions cannot be scheduled to these

nodes and are drained from these nodes.

> Note：
> The value of Taints usually consists of `key` , `value` , and `effect` . Valid values of `effect` :
>
> - **PreferNoSchedule**: optional condition. A Pod is not likely to be scheduled to a node with a taint that cannot be tolerated by the Pod.
> - **NoSchedule**: when a node contains a taint, a Pod without the corresponding toleration to the taint will never be scheduled to the node.
> - **NoExecute**: when a node contains a taint, a Pod without the corresponding toleration to the taint will not be scheduled to the node and will be drained from the node if any.

Assume that Taints is set to `key1=value1:PreferNoSchedule` . The following figure shows the configurations in the TKE console:

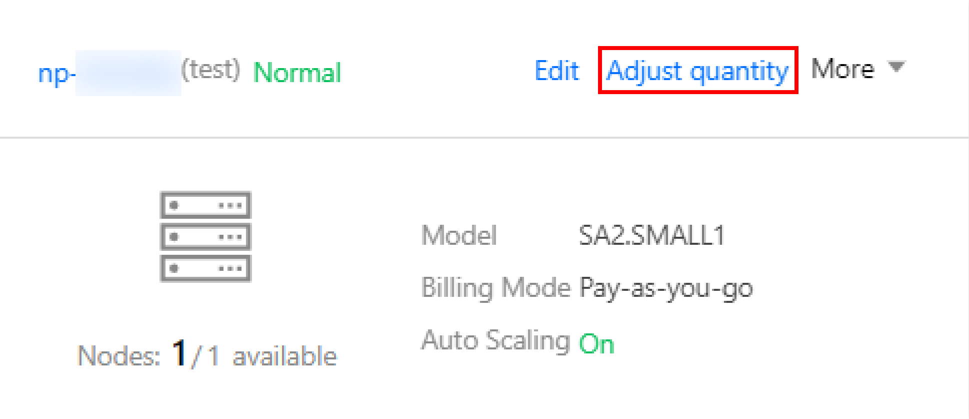


Taints key1 = value1 PreferNoSchedule

3. Click **OK** and wait until the update is completed.

Adjusting the number of nodes in node pool

1. On the card page of the desired node pool, click **Adjust quantity** in the top-right corner as shown below:

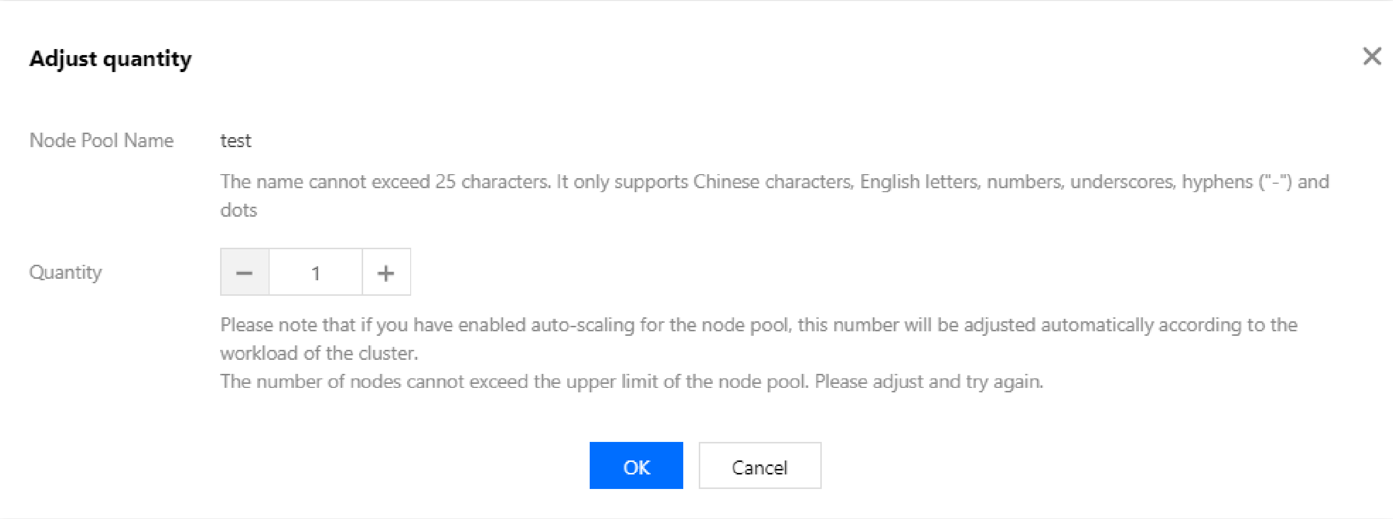


np-[redacted](test) Normal

Model SA2.SMALL1
Billing Mode Pay-as-you-go
Auto Scaling On
Nodes: 1 / 1 available

2. On the **Adjust quantity** page that pops up, adjust the node quantity as needed. The quantity must be within the range of node quantity set in the node pool as shown below:

Note：
When auto scaling is enabled for the node pool, this number is automatically adjusted according to the workload of the cluster. However, the actual number of nodes may be inconsistent with the number that is specified during adjustment.



Adjust quantity

Node Pool Name test

The name cannot exceed 25 characters. It only supports Chinese characters, English letters, numbers, underscores, hyphens ("-") and dots

Quantity 1

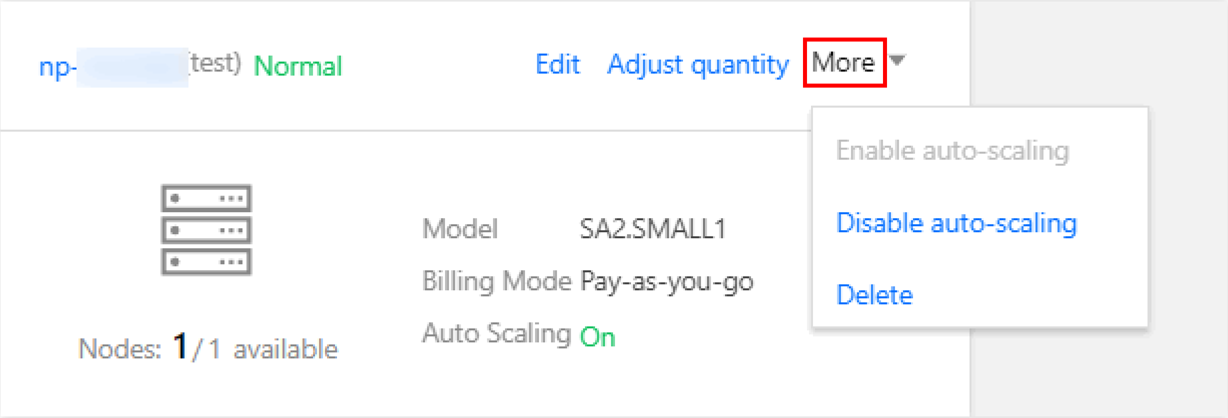
Please note that if you have enabled auto-scaling for the node pool, this number will be adjusted automatically according to the workload of the cluster.
The number of nodes cannot exceed the upper limit of the node pool. Please adjust and try again.

3. Click **OK** and wait until the number is adjusted.

Enabling or disabling auto scaling

Note：
We recommend you enable or disable auto scaling in the node pool on the TKE side to ensure that the status can be synced to Cluster-autoscaler.

1. On the card page of the target node pool, click **More** in the top-right corner as shown below:



Nodes: 1 / 1 available

Model SA2.SMALL1

Billing Mode Pay-as-you-go

Auto Scaling On

2. Select **Enable auto-scaling** or **Disable auto-scaling** based on your actual conditions and click **OK** in the pop-up window.

Related Operations

For more information on the features and operations of node pools, see the following documents:

- Creating a Node Pool
- Viewing a Node Pool
- Deleting a Node Pool

Deleting a Node Pool

Last updated：2023-05-24 16:29:04

Introduction

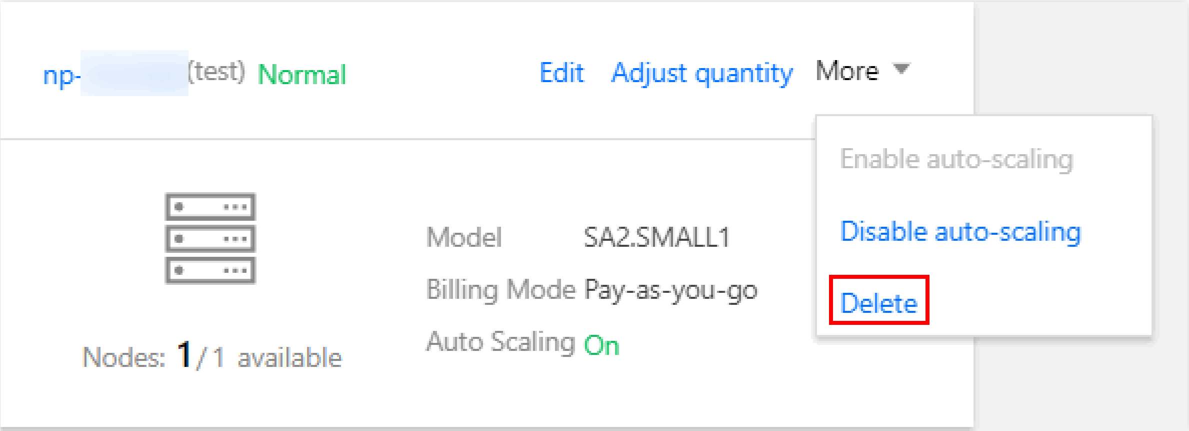
This document describes how to delete an existing node pool from a cluster via the TKE console. You can delete idle node pools to reduce the waste of resources.

Prerequisites

- You have created an available node pool. For more information, see Creating a Node Pool.
- You have opened the “Node pool list” page. For more information, see Viewing a Node Pool.

Directions

1. On the card page of the desired node pool, choose **More** -> **Delete** in the upper-right corner, as shown in the following figure:



np-[redacted](test) Normal

Model SA2.SMALL1

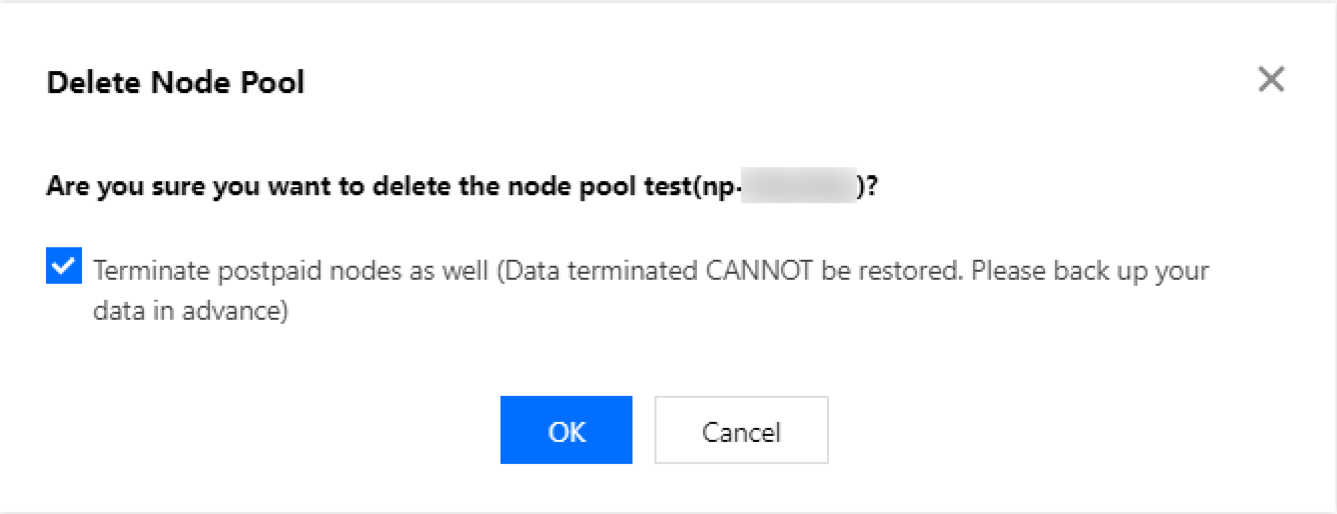
Billing Mode Pay-as-you-go

Auto Scaling On

Nodes: 1/1 available

2. In the “Delete node pool” window, specify whether to retain the nodes based on requirements.

> Note：
>
> - Terminate postpaid node is selected by default. You can clear the option as required.
> - Once terminated, pay-as-you-go nodes cannot be restored. Proceed with caution and back up data in advance.



Delete Node Pool

Are you sure you want to delete the node pool test(np-[redacted])?

☑ Terminate postpaid nodes as well (Data terminated CANNOT be restored. Please back up your data in advance)

3. Click **OK** and wait until the node pool is deleted.

Relevant Operations

For more information on the features and operations of node pools, see the following documents:

- Creating a Node Pool
- Viewing a Node Pool
- Adjusting a Node Pool

Viewing Node Pool Scaling Logs

Last updated：2020-11-23 10:12:25

Overview

This document describes how to view scaling records of node pools, which can help you:

- See traffic changes in business and configure node pools to more efficiently meet demands.
- See expenditures to manage costs more efficiently.
- See the reasons for scaling failures to manage risks. For example, scale-out may fail because all resources in a region are sold out.
- See global scaling records and scaling records of a specific node.

Note：

- When multiple node pools exist, Cluster Autoscaler (CA) selects a proper node pool for scaling. Global scaling records can be obtained based on CA events.
- If you are only interested in the scaling records of a specific node pool and do not care about the CA behavior, go to the node pool details page to view scaling records of this node pool.

Prerequisites

- You have created an available node pool. For more information, please see Creating a Node Pool.
- You have opened the "Node pool list" page. For more information, please see Viewing a Node Pool.

Directions

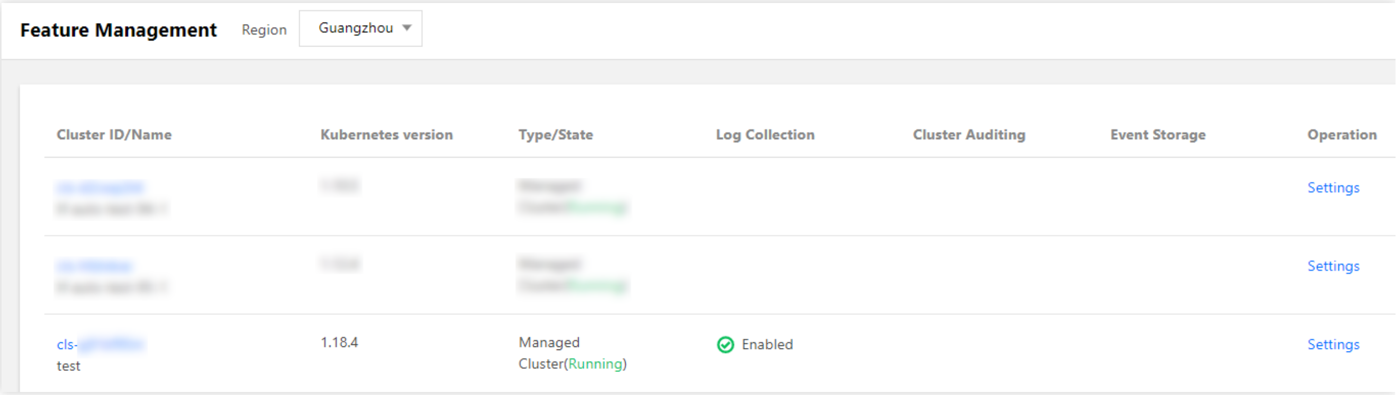
Viewing global scaling records

The community open-source component CA stores relevant information of each scaling activity under a specific pod or node as a Kubernetes event. However, Kubernetes events are stored in the backend for only 1 hour by default. If you want to query and review the scaling records of a node pool, we recommend that you enable event persistence to persistently store Kubernetes events.

Enabling event persistence

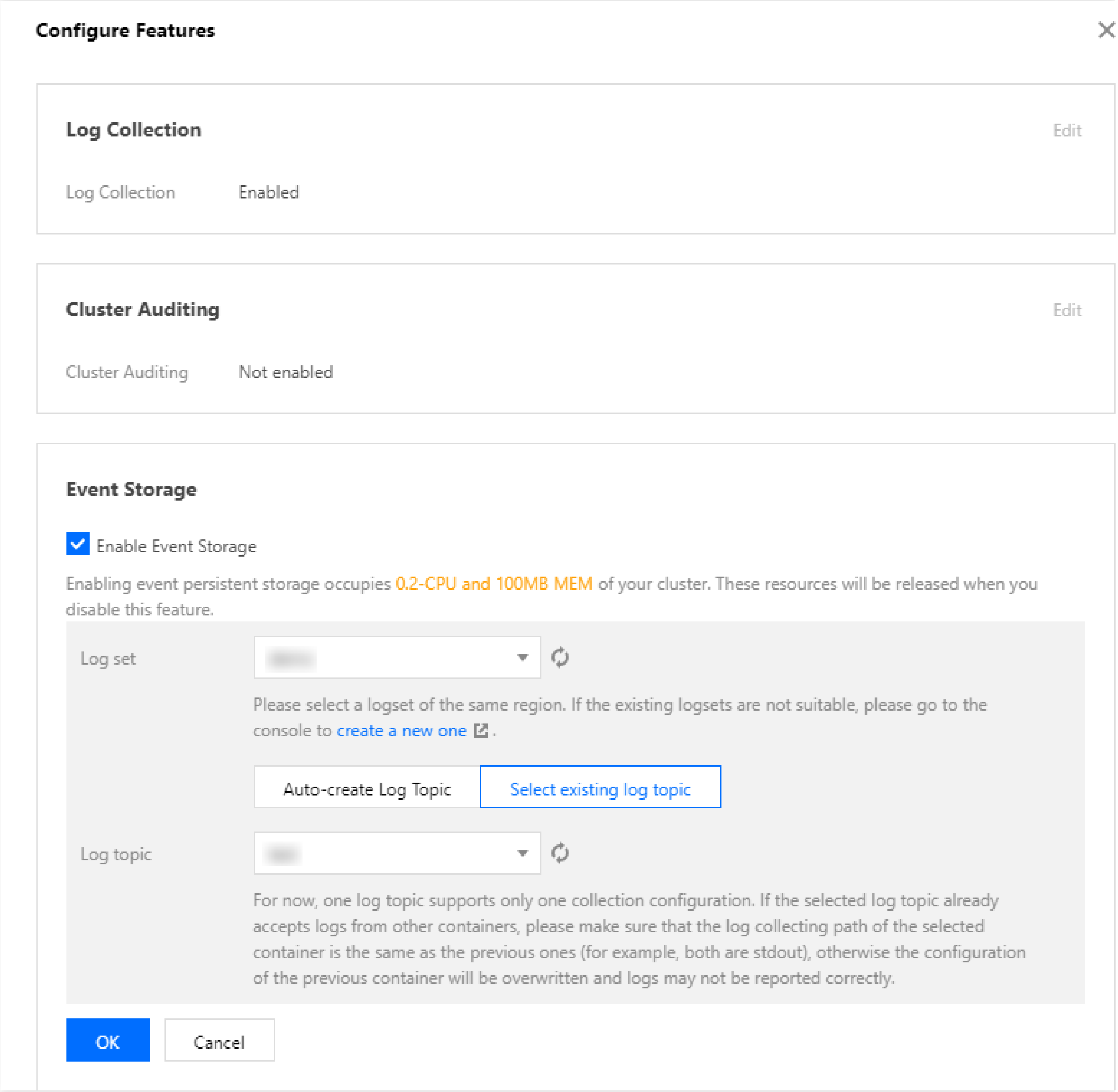
1. Log in to the TKE console.

2. Choose **Cluster OPS** > **Feature Management** in the left sidebar to go to the **Feature Management** page.
3. At the top of the **Feature Management** page, select a region. Click **Set** next to the cluster for which you want to enable event persistence, as shown in the figure below.



Cluster ID/Name	Kubernetes version	Type/State	Log Collection	Cluster Auditing	Event Storage	Operation
[redacted]	[redacted]	[redacted]				Settings
[redacted]	[redacted]	[redacted]				Settings
cls-[redacted] test	1.18.4	Managed Cluster(Running)	✓ Enabled			Settings

4. In the **Configure Features** window that appears, click **Edit** next to the **Event Storage** feature.
5. Select **Enable Event Storage** and select the logset and log topic for event persistence, as shown in the figure below.



Configure Features

Log Collection

Log Collection Enabled

Cluster Auditing

Cluster Auditing Not enabled

Event Storage

☑ Enable Event Storage

Enabling event persistent storage occupies 0.2-CPU and 100MB MEM of your cluster. These resources will be released when you disable this feature.

Log set

Please select a logset of the same region. If the existing logsets are not suitable, please go to the console to create a new one.

Auto-create Log Topic | Select existing log topic

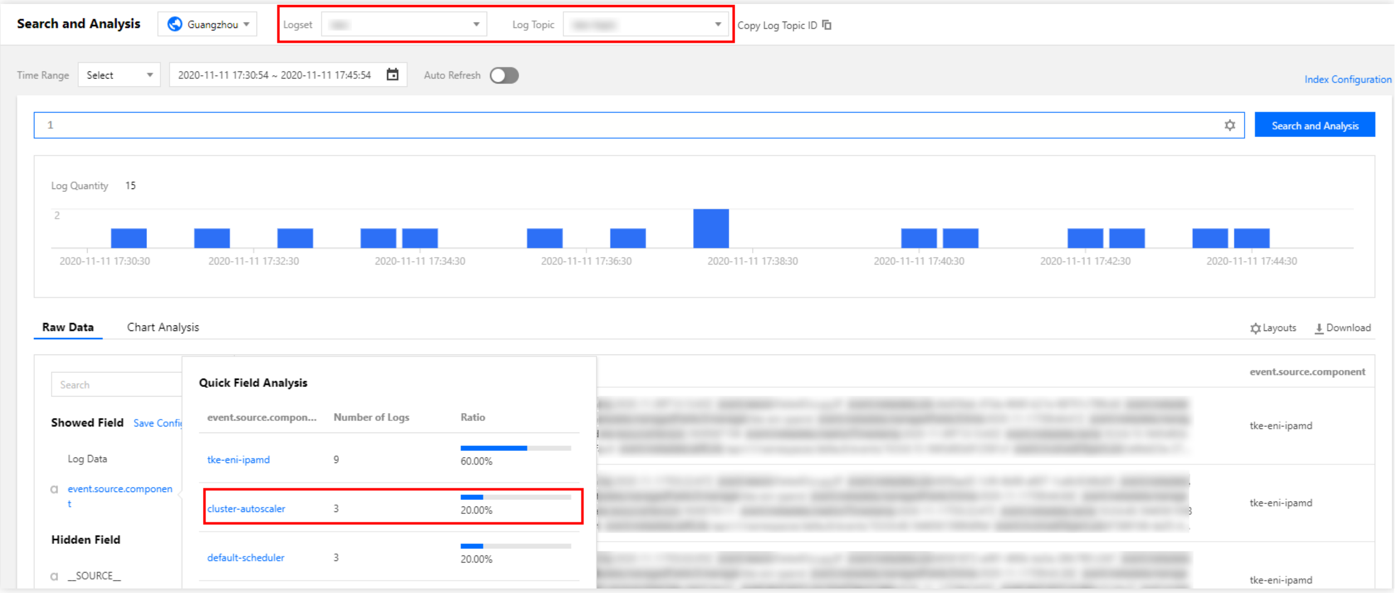
Log topic

For now, one log topic supports only one collection configuration. If the selected log topic already accepts logs from other containers, please make sure that the log collecting path of the selected container is the same as the previous ones (for example, both are stdout), otherwise the configuration of the previous container will be overwritten and logs may not be reported correctly.

6. Click **OK**.

Querying event persistence

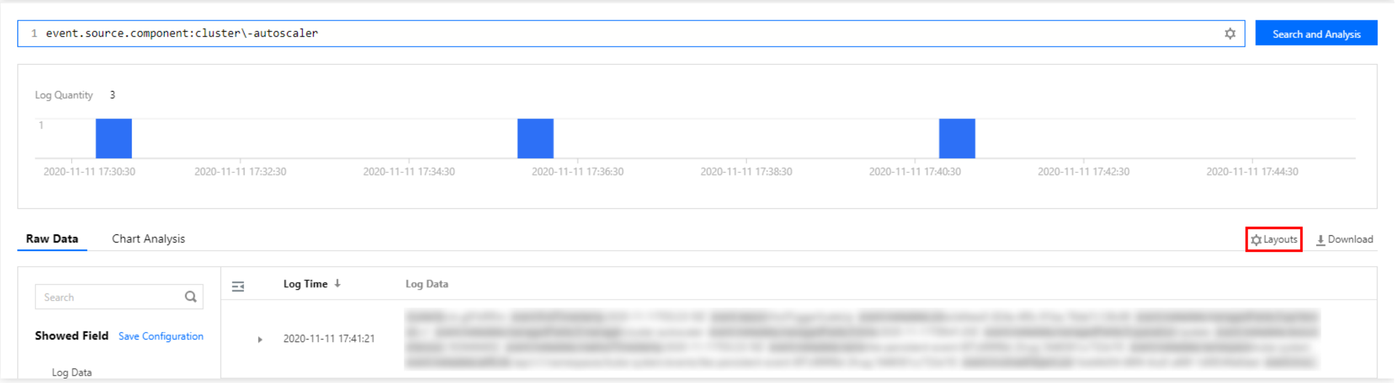
1. Log in to the CLS console.
2. Click **Search and Analyze** in the left sidebar to go to the **Search and Analyze** management page.
3. At the top of the **Search and Analyze** page, select a region and select the event persistence logset and log topic that you want to view.
4. Select **event.source.component:cluster-autoscaler** and click **Search and Analyze**, as shown in the figure below.



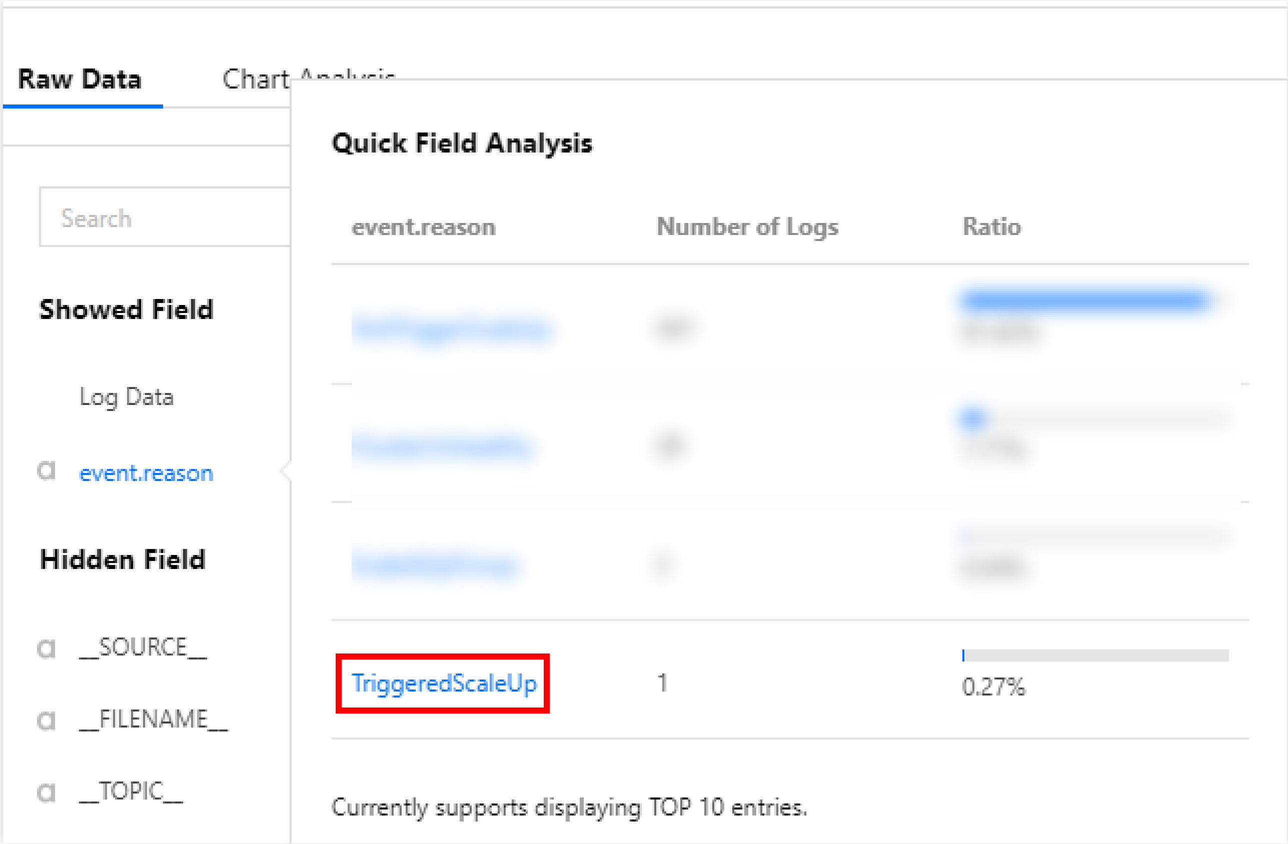
Quick Field Analysis

event.source.compon...	Number of Logs	Ratio
tke-eni-ipamd	9	60.00%
cluster-autoscaler	3	20.00%
default-scheduler	3	20.00%

5. Configure data columns in **Column Settings** on the right and visualize the desired columns, as shown in the figure below.


```
1 event.source.component:cluster\-autoscaler
```

Specify an event type. For example, if you only want to view scale-out events, select **TriggeredScaleUp** for the search, as shown in the figure below.

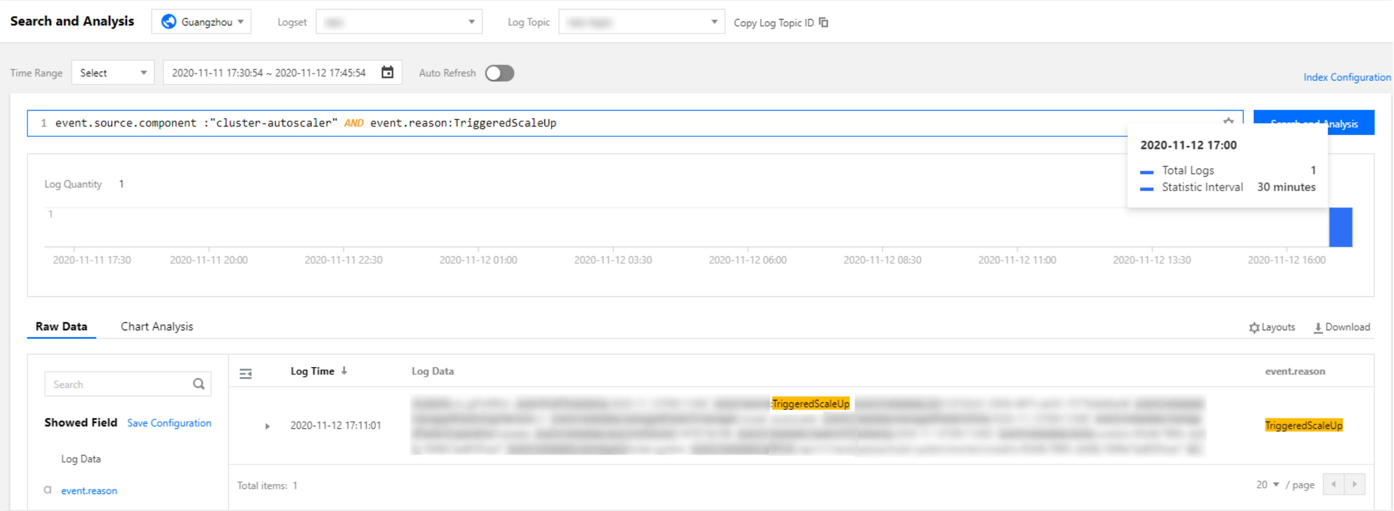


Quick Field Analysis

event.reason	Number of Logs	Ratio
TriggeredScaleUp	1	0.27%

Currently supports displaying TOP 10 entries.

6. The scaling log querying result (including all node pool scale-out logs) is as follows:



Time Range: Select 2020-11-11 17:30:54 ~ 2020-11-12 17:45:54 Auto Refresh

```
event.source.component :"cluster-autoscaler" AND event.reason:TriggeredScaleUp
```

2020-11-12 17:00
Total Logs 1
Statistic Interval 30 minutes

Log Time	Log Data	event.reason
2020-11-12 17:11:01	TriggeredScaleUp	TriggeredScaleUp

Total items: 1 20 / page

Search guide

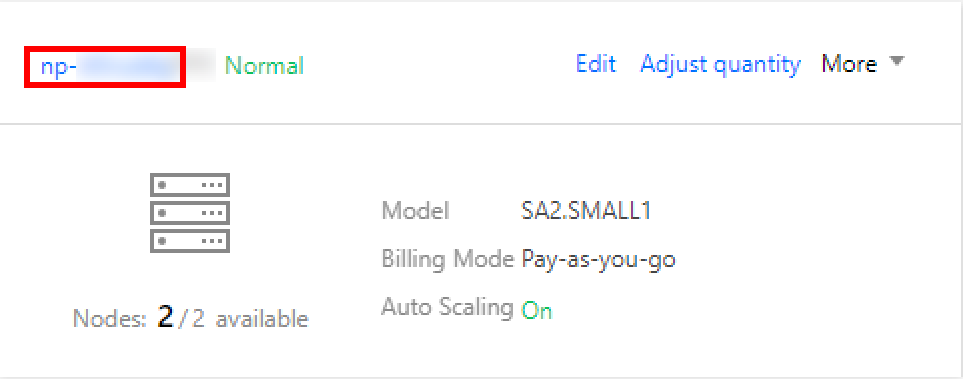
You can refer to the following documents to view a more detailed scaling activity list:

- Syntax and Rules
- CA FAQ

- For CA scaling events, the value of the Reason field may be any of the following: TriggeredScaleUp, NotTriggerScaleUp, ScaledUpGroup, FailedToScaleUpGroup, ScaleDown, ScaleDownFailed, and ScaleDownEmpty. For more information, see Detailed Field Description.

Querying scaling logs of a specific node pool

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the desired cluster ID to open the **Deployment** page.
3. In the left sidebar, choose **Node Management** > **Node Pool** to open the **Node Pool List** page.
4. On the node pool page, click the desired node pool ID, as shown in the figure below.

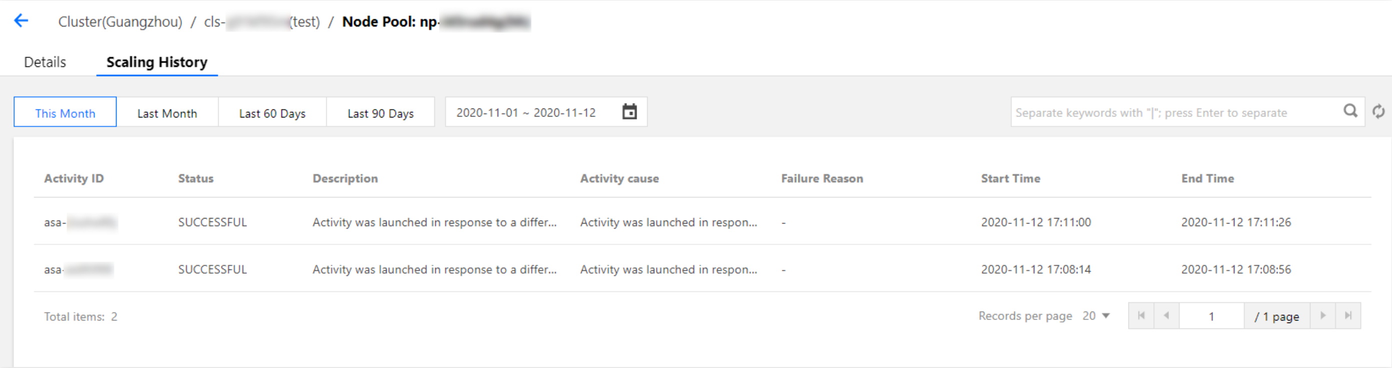


Model SA2.SMALL1

Billing Mode Pay-as-you-go

Auto Scaling On

5. On the node pool details page, click the **Scaling Logs** tab on the top, as shown in the figure below.


Cluster(Guangzhou) / cls-[redacted](test) / Node Pool: np-[redacted]
Details
Scaling History
This Month
Last Month
Last 60 Days
Last 90 Days
2020-11-01 ~ 2020-11-12
Separate keywords with "|"; press Enter to separate

Activity ID	Status	Description	Activity cause	Failure Reason	Start Time	End Time
asa-[redacted]	SUCCESSFUL	Activity was launched in response to a differ...	Activity was launched in respon...	-	2020-11-12 17:11:00	2020-11-12 17:11:26
asa-[redacted]	SUCCESSFUL	Activity was launched in response to a differ...	Activity was launched in respon...	-	2020-11-12 17:08:14	2020-11-12 17:08:56

Total items: 2

Records per page 20 ▾ 1 / 1 page

The scaling log fields are as follows:

- **Activity ID**: ID of a scaling activity
- **Status**: status of a scaling activity
- **Description*: description of a scaling activity, displaying the number of scale-out/scale-in nodes
- **Activity Cause**: causes for triggering a scaling activity
- **Failure Cause**: if a scaling activity fails, this column displays the causes of failure.
- **Start Time**: time when a scaling activity starts, in seconds
- **End Time**: time when a scaling activity ends, in seconds

References

For more information on the features and operations of node pools, please see the following documents:

- Creating a Node Pool
- Viewing a Node Pool
- Adjusting a Node Pool

Node Pool FAQs

Last updated：2023-09-26 17:40:16

This document lists the common questions and issues during the usage of general node pools.

What is the relationship between a node pool and a scaling group?

A node pool is a collection of nodes with similar specifications, configurations, and attributes. You can manage the nodes in batch through the TKE console, such as configuring node specifications, labels, taints, scripts and other parameters. You can create node pools with different billing modes (such as pay-as-you-go, spot) in a cluster. The underlying implementation of a node pool relies on Auto Scaling. There are two main concepts:
A **scaling group** contains a collection of CVM instances that follow the same policies and are used for the same snario. Attributes of a scaling group include the maximum and minimum numbers of CVM instances and more.
Launch configuration is the template for automatic creation of CVMs. It specifies the CVM instance type, system disk/data disk types and capacities, key pair, security group, etc.
Each node pool corresponds to a scaling group, and each scaling group corresponds to a launch configuration. You can check the bound launch configuration and scaling group in the node pool details page in the TKE console.

Which node pool parameters can I modified?

Reminder:
To ensure the proper scaling of node pools, DO NOT modify parameters in the AS console unless they are stated in this document.

Parameter	How to modify
Node pool name	TKE console > Node pool information
Auto Scaling	
Node range	
Cloud tag, Deletion protection	
Labels, Taints	TKE console > Node pool information: You can specify whether to apply the modifications to existing nodes.
Operating system	TKE console > Node configuration details: The modifications only apply to nodes newly added to the node pool.
Runtime components	
Secondary models	TKE console > Node configuration details A node pool can contain up to 10 models (including the primary model).

	The list shows only instance types available in the AZ of the node pool subnet. If a GPU-model taken as the primary model, , the driver is specified upon node pool creation. The secondary model must also be a GPU model.
Custom data	TKE console > Node configuration details: Modifications only apply to nodes newly added to the node pool. **Note:** **DO NOT** directly modify the custom data in the launch configuration of AS, which may cause the failure of adding nodes to the cluster.
Security group	AS console > Launch configuration details:. The modifications only apply to nodes newly added to the node pool.
Instance name (node name)	AS console > Launch configuration > Advanced: The modifications only apply to nodes newly added to the node pool.
Subnet	AS console > Scaling group details: Modifications only apply to nodes newly added to the node pool.
Instance creation policy/Retry policy	AS console > Scaling groups > Policy information: Modifications take effect from the next auto scaling task.

Which node pool parameters should not be modified

Parameter	Description
Billing mode	You cannot modify the billing mode in the TKE console. In addition, we suggest NOT** to modify the instance billing mode of a launch configuration in the AS console.
Data disk	The data disk cannot be modified. In addition, it is recommended **not** to modify the capacity of data disks, and not to add or delete a data disk in the launch configuration of AS. Otherwise, the pending Pods may fail to be scheduled to the newly added nodes.
Supported network (VPC)	The VPC cannot be modified. In addition, it is recommended **not** to modify the supported network on the scaling group details page of AS, otherwise it may cause the failure of node expansion.

Note:

Each node pool corresponds to a unique scaling group and launch configuration. The launch configuration cannot be bound to other scaling groups, otherwise the node pool may fail to be deleted.

It is recommended **not** to modify other parameters of the scaling group in the AS console

Native Node Management

Overview

Last updated：2023-05-08 18:12:58

Definition

Native nodes are a new type of nodes that TKE provides for Kubernetes environments. Developed based on Tencent Cloud's technological experience in operations of ten million-core containers, native nodes provide users with native, highly stable, and fast-responding Kubernetes node management capabilities.

Benefits

Integrated with the FinOps principle, native nodes boost cost optimization of cloud resources

Provide the HouseKeeper visualized resource dashboard to improve resource utilization, reduce costs, and enhance efficiency.
Support request recommendation to avoid resource idling.
Provide dedicated dynamic scheduling capabilities with the following features:
Load balancing: Better balance the resource load on nodes based on the current load and load history.
Packing improvement: Increase the amount of schedulable node CPU and memory resources and improve the node packing rate to over 100%.
Business adjustment: Allow you to set an expected resource utilization rate and ensure continuous node scheduling to realize more centralized business resource deployment.

Native nodes provide multi-dimensional management capabilities to reduce operations workload

New Kubernetes operations model: Provide the declarative infrastructure API for users to manage nodes in the same way as workload management. You can manage nodes using the Kubernetes API, the Tencent Cloud API, or the TKE console.
Tencent’s intelligent operations system: Support operating system-, runtime-, and Kubernetes-level fault detection and automatic upgrade to reduce operations workload for users.
Based on Tencent Cloud’s cloud-native technology practices, parameters are tuned and adapted comprehensively at the operating system, runtime, and Kubernetes levels, significantly enhancing the node initialization stability.

Native Nodes vs. Normal Nodes

In general, native nodes provide all the capabilities that normal nodes have, with better performance. The following table compares the two types of nodes in details.

Module	Native Nodes	Normal Modes
Management mode	Node HouseKeeper mode: This mode provides resource management and stable operations capabilities to assist users in decision-making.	Serverful mode: Analysis, decision-making, and action are all conducted by users.
Declarative management of infrastructure	Supported	Not supported
In-place Pod configuration adjustment	Supported	Not supported
Custom configuration entries for kernel parameter tuning and other purposes	Supported	Not supported
Node self-heal	Tencent's operating system-, Kubernetes-, and runtime-level fault detection and self-heal capabilities	NPD add-on (discontinued)
Scheduler	Schedulers dedicated for native nodes, supporting virtual scale-outs of schedulable resources	Dynamic Scheduler and DeScheduler
Request Recommendation	Recommendation and quick update	Not supported
Job occupation	Supported	Not supported
Manage Node	Kernel parameter configuration, nameserver parameter configuration, host parameter configuration, pre-request scripts, and post-request scripts	Not supported

Billing Modes

Native nodes support multiple types of Cloud Virtual Machine (CVM) instances. You can select appropriate instances to deploy based on your application scale and business characteristics. TKE charges resources (including CPU, memory, GPU, and system disk resources) consumed by native nodes according to the node type and resource specifications.

Native nodes supported the **pay-as-you-go** billing mode.

Billing Mode	**Pay-as-you-go**
Payment method	Postpaid (Fees are frozen upon purchase and settled hourly)
Billing unit	USD per second
Price	High
Minimum use duration	Billing per second and settlement per hour; resource purchase and release allowed at any time
Use case	Suitable for periodic computing scenarios such as transcoding, big data, and e-commerce flash sale campaigns, or tidal online service scenarios where Horizontal Pod Autoscaler (HPA) is enabled.

Regions and Availability Zones

You can use native nodes in the following regions.

China

Region			Region Abbreviation
China	Regions on the public cloud	Beijing	ap-beijing
		Nanjing	ap-nanjing
		Shanghai	ap-shanghai
		Guangzhou	ap-guangzhou
		Chengdu	ap-chengdu
		Chongqing	ap-chongqing
		Hong Kong (China)	ap-hongkong
		Taipei (China)	ap-taipei

Other countries and regions

Region			Region Abbreviation

Asia Pacific	Regions on the public cloud	Singapore	ap-singapore
		Mumbai	ap-mumbai
		Jakarta	ap-jakarta
		Seoul	ap-seoul
		Bangkok	ap-bangkok
		Tokyo	ap-tokyo
North America		Silicon Valley	na-siliconvalley
		Virginia	na-ashburn
		Toronto	na-toronto
South America		São Paulo	sa-saopaulo
Europe		Frankfurt	eu-frankfurt

References

Note

For the convenience of management, we recommend that you use a **node pool** to create and manage a group of native nodes with the same parameter settings.

Creating Native Nodes: You can create a native node by using the TKE console, calling the Kubernetes API, or calling the TencentCloud API.

Delete Native Nodes

Self-Heal Rules

Declarative Operation Practice

Native Node Scaling

In-Place Pod Configuration Adjustment

Management Parameters

Purchasing Native Nodes

Native Node Pricing

Last updated：2024-08-08 16:15:34

Pricing

Note:

Native node is a TKE cloud product billed separately. Its pricing is different from that of that of the Cloud Virtual Machine (CVM) service. The billing information of different specifications of native nodes is subject to that displayed on the console.

TKE native nodes support multiple types of CVM instances. You can select appropriate instances to deploy based on your application scale and business characteristics. TKE charges resources (including CPU, memory, GPU, and system disk resources) consumed by native node instances according to the instance type and resource specifications.

Billing formula: Fees = **Unit price of the native node instance resource configuration** * **Running time**

Billing Item	Description
Instance type	Instance types supported by the current native node are as follows: (For more information, see CVM instances) Standard: S2, S4, S5, SA2, SA3, S6, SA5, SA4, S7, S8 Computing: C3, C4, C5, C6 Memory optimized: M3, MA3, M5, M6 GPU: GN7, GNV4, PNV4, GN10X, GN10Xp, GT4 High I/O: IT5
Resource specifications	Resource specifications supported by the current native node are as follows: (Subject to those displayed on the console) CPU: Two cores or more Memory: 2 GB or more System disk: Premium Cloud Storage or SSD, 50-1024 GB GPU: Only GPU native node instances contain GPU resources and support only entire GPUs.

Billing Modes

TKE provides only one billing mode for native node instances: pay-as-you-go. Details are as follows:

Billing Mode	**Pay-as-you-go**

Payment method	Postpaid (Fees are frozen upon purchase and settled hourly)
Billing unit	USD/second
Unit price	High
Minimum usage duration	Billed by second and settled by hour. You can purchase or release resources at any time.
Use cases	Suitable for periodic computing scenarios such as transcoding, big data, and e-commerce flash sale campaigns, or tidal online service scenarios where Horizontal Pod Autoscaler (HPA) is enabled.

Pay-as-you-go

Billing formula: Fees = Number of requested native node instances × Unit price of the native node instance resource configuration (by usage) × Usage duration

When you activate a pay-as-you-go native node instance, the fees (including CPU, memory, GPU, and system disk fees) for 1-hour usage based on the resource configuration unit price will be frozen in your account balance as a deposit. You will then be billed by the hour (Beijing time) for your usage over the past hour. When you purchase a native node instance, the unit price will be listed as an hourly fee. However, you will actually be billed by the second and the fees will be rounded to the nearest two decimal places. Billing starts from the second the native node instance is purchased and stops the second the instance is terminated. When the instance is terminated, the frozen amount will be unfrozen.

References

Overdue Payment

Payment Overdue

Last updated：2023-02-23 18:34:01

If your account has overdue payment, a notification will be sent to you. Once receiving it, please go to the Billing Center in the console and pay the past due charges in time to prevent your business from being affected. This document will provide detailed information on overdue payment of native nodes.

Alerting

Reminder Type	Description
Expiration	Seven days before cluster resource expiration, the system will send an expiration alert to your Tencent Cloud account creator, global resource collaborators, and financial collaborators by email and SMS.
Overdue payment	On the day of and after cluster resource expiration, the system will send an overdue payment alert to your Tencent Cloud account creator and all collaborators by email and SMS.

Repossession

Note：

- Suspension: When the resource of a user is suspended, the user cannot use the resource any more. However, data of the resource is not deleted, and the user can resume the resource.
- Release: Data of a resource is deleted, and the resource cannot be resumed.

Pay-as-you-go instance repossession mechanism

- Starting from the point when your account balance becomes negative, the native node instances are still available for the next **two hours**, and the billing continues. Two hours later, if your account is still not topped up to a balance greater than zero, the native node instances will be suspended and released (the node resources will be terminated, and data cannot be restored).
- If your account is topped up to a balance greater than zero two hours after your account has overdue payments, the fees of the native node instances will be calculated and deducted according to the pay-as-you-go billing rules.

Lifecycle of a Native Node

Last updated：2023-05-05 11:05:32

Lifecycle Description

Status	Description
Healthy	The node is running normally and connected to the cluster.
Exception	The node is running abnormally and not connected to the cluster.
Creating	The node is being created and not connected to the cluster. After **machine purchase**, **component installation**, and **node registration** are complete, the node can be connected to the cluster.
Draining...	The node is draining the Pod to another node.
Restarting	The node is restarting and cannot be connected to the cluster. No new Pods can be scheduled to this node.
Cordoned	The node is cordoned and no new Pods can be scheduled to this node.

Native Node Parameters

Last updated：2024-06-14 16:28:43

Parameter	Support Details	Description
Model	Standard: S2, S4, S5, SA2, SA3, S6, SA5, SA4, S7, and S8 Compute optimized: C3, C4, C5, and C6 Memory optimized: M3, MA3, M5, and M6 GPU optimized: GN7, GNV4, PNV4, GN10X, GN10Xp, and GT4 High-IO type: IT5	The models displayed in the console vary with the remaining resources in the selected availability zone. If you need other models, you can submit a ticket to seek for help. Moreover, native node pools **support several alternative models with the same specifications**. You can go to the node pool details page to configure.
System disk	Premium Cloud Disk and SSD	It is recommended that the system disk be at least 100 GB.
Data disk	Premium Cloud Disk, SSD, Balanced SSD, and Enhanced SSD	It is recommended that the data disk be at least 50 GB. By default, no data disk is bound.
Public network bandwidth	EIP binding is supported.	By default, the node's public network bandwidth is not enabled. For details, see Enable the Public Network Access.
Operating system	TencentOS Server	Based on the Tencent Cloud virtualization platform, by using techniques such as kernel optimization, the operating system supports cloud-native resource isolation and performs comprehensive parameter tuning for container applications.
Node login	SSH login	By default, node login is enabled. You can enable and distribute an SSH key in the console for the node to be logged in. For details, see Enable SSH Key Login.
GPU driver	Drivers 450/470/515/525	When creating a node pool, you can select the target driver below model selection.
Runtime	Only Containerd is supported.	For Kubernetes clusters 1.16 and later versions, the supported Containerd version is 1.6.9.
Kubernetes	K8s major version 1.16 or later	Some K8s versions have specific minor version requirements including: v1.16.3-tke.28 or later;

		v1.18.4-tke.26 or later; v1.20.6-tke.21 or later.
Operation and Maintenance Parameters	The setting of parameters for Kubelet, Kernel, Hosts, and Nameservers is supported.	For details, see Management Parameters.
Initialization Script	Two phases are supported: before node initialization and after node initialization.	You can provide an initialization script when creating a node pool.
Node Auto Scaling	Supported.	For details, see Native Node Scaling.
Node Specification Enlargement	Supported. A scalable dedicated scheduler for native nodes is provided.	For details, see Dedicated Scheduler Product Description.
qGPU	Supported.	For details, see qGPU Overview.
Memory Compression	Supported.	For details, see the Memory Compression User Guide.
Preemptible Jobs	Supported.	For details, see the preemptible Job feature description.
QoSAgent	Supported.	Multi-dimensional fine scheduling capabilities such as CPU Priority, CPU Burst, Memory/Network/Disk IO QoS enhancements are provided, improving cluster resource utilization while ensuring quality stability. For details, see Fine Scheduling.
Fault Self-Healing	Supported.	Based on TKE's self-developed intelligent operations product Cloud Explorer, it provides multi-dimensional fault self-healing capabilities, comprehensively enhancing node stability and improving operational efficiency. For details, see Fault Self-Healing Rules.
In-Place Scaling	Supported.	For details, see Enable Pod In-Place Scaling.

Creating Native Nodes

Last updated：2024-06-14 16:28:43

This document describes how to create native nodes in the Tencent Kubernetes Engine (TKE) console or by using the YAML configuration file.

Prerequisites

You have logged in to the TKE console.

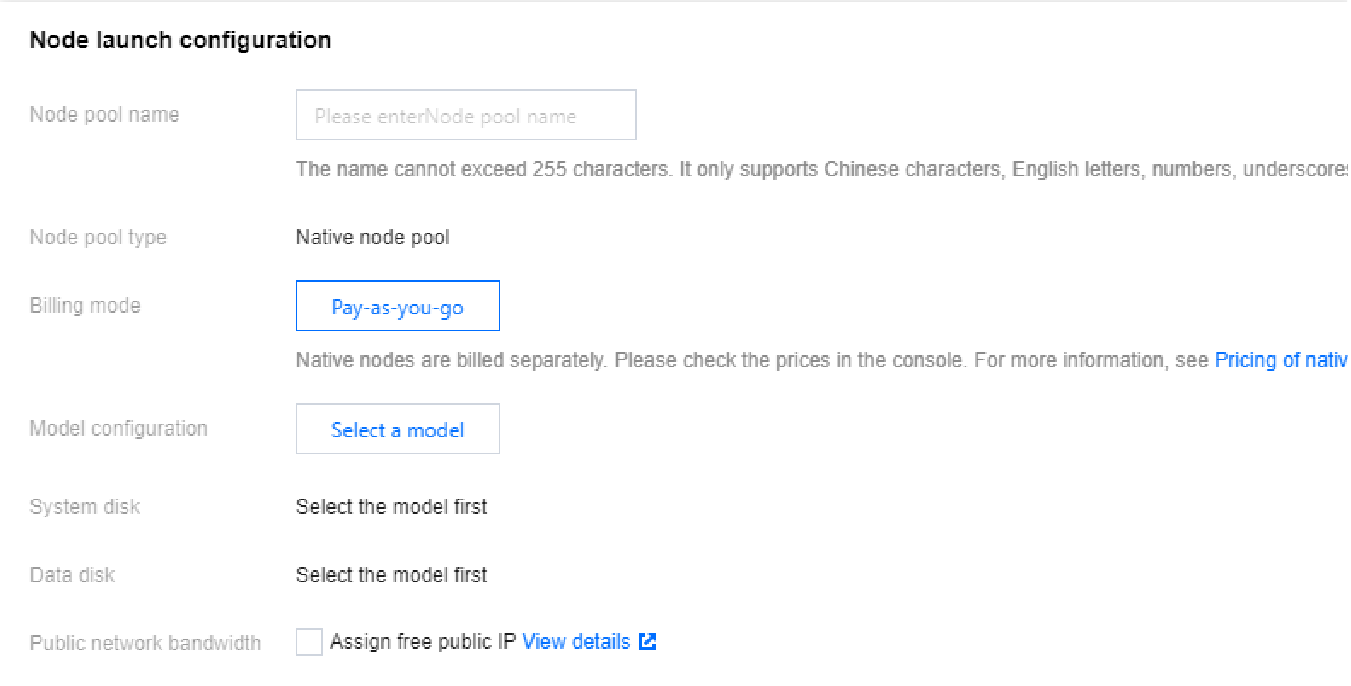
You have created a standard TKE cluster. For more information, see Quickly Creating a Standard Cluster.

Note

You can manage native nodes only at the **node pool** level.

Using the Console

1. Log in to the TKE console and choose **Cluster** from the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Node pool** from the left sidebar to go to the **Node pool list** page.
4. Click **Create node pool**. On the **Create node pool** page, set the parameters as shown in the following figure. For more information about the parameters, see Parameters.



Node launch configuration

Node pool name: Please enterNode pool name

The name cannot exceed 255 characters. It only supports Chinese characters, English letters, numbers, underscore

Node pool type: Native node pool

Billing mode: Pay-as-you-go

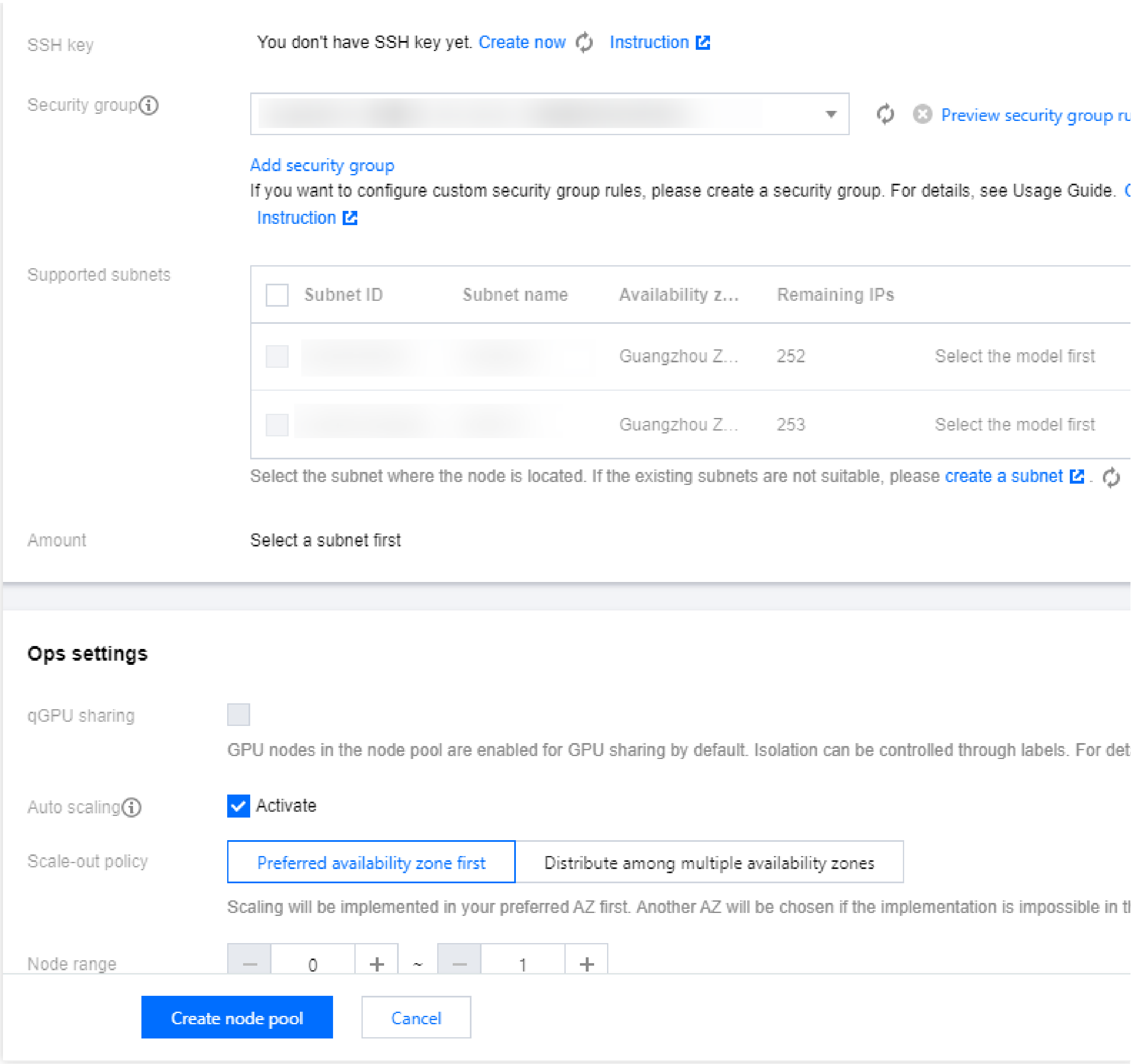
Native nodes are billed separately. Please check the prices in the console. For more information, see Pricing of nativ

Model configuration: Select a model

System disk: Select the model first

Data disk: Select the model first

Public network bandwidth: □ Assign free public IP View details



SSH key: You don't have SSH key yet. Create now Instruction

Security group: [redacted] Preview security group ru

Add security group

If you want to configure custom security group rules, please create a security group. For details, see Usage Guide. (

Instruction

Supported subnets

□	Subnet ID	Subnet name	Availability z...	Remaining IPs	
□	[redacted]	[redacted]	Guangzhou Z...	252	Select the model first
□	[redacted]	[redacted]	Guangzhou Z...	253	Select the model first

Select the subnet where the node is located. If the existing subnets are not suitable, please create a subnet .

Amount: Select a subnet first

Ops settings

qGPU sharing: □

GPU nodes in the node pool are enabled for GPU sharing by default. Isolation can be controlled through labels. For det

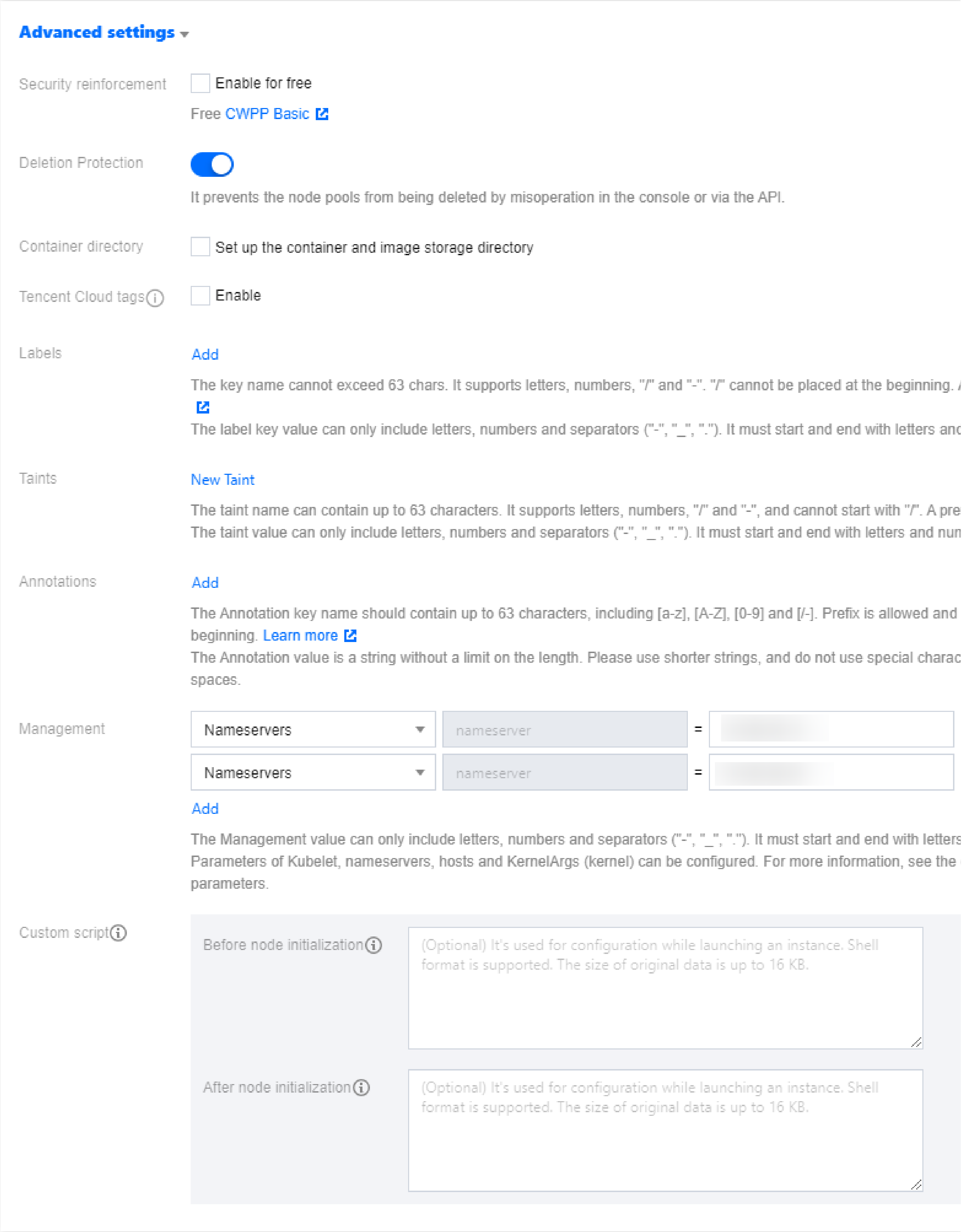
Auto scaling: ☑ Activate

Scale-out policy: Preferred availability zone first | Distribute among multiple availability zones

Scaling will be implemented in your preferred AZ first. Another AZ will be chosen if the implementation is impossible in tl

Node range: 0 ~ 1

5. (Optional) Click **Advanced settings** to view or configure more settings, as shown in the following figure:



Advanced settings ▾

Security reinforcement □ Enable for free

Free CWPP Basic

Deletion Protection ●

It prevents the node pools from being deleted by misoperation in the console or via the API.

Container directory □ Set up the container and image storage directory

Tencent Cloud tags □ Enable

Labels Add

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A

The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and

Taints New Taint

The taint name can contain up to 63 characters. It supports letters, numbers, "/" and "-", and cannot start with "/". A pre
The taint value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and nun

Annotations Add

The Annotation key name should contain up to 63 characters, including [a-z], [A-Z], [0-9] and [/-]. Prefix is allowed and beginning. Learn more
The Annotation value is a string without a limit on the length. Please use shorter strings, and do not use special charac spaces.

Management	Nameservers ▾	nameserver	= [redacted]
	Nameservers ▾	nameserver	= [redacted]

Add

The Management value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters
Parameters of Kubelet, nameservers, hosts and KernelArgs (kernel) can be configured. For more information, see the parameters.

Custom script

Before node initialization: (Optional) It's used for configuration while launching an instance. Shell format is supported. The size of original data is up to 16 KB.

After node initialization: (Optional) It's used for configuration while launching an instance. Shell format is supported. The size of original data is up to 16 KB.

6. Click **Create node pool**.

Using YAML

The following sample code shows the specifications of Kubernetes resources in a native node pool. For more information about the parameters in the YAML configuration file, see Parameters.

```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  type: Native
  displayName: mstest
  replicas: 2
  autoRepair: true
  deletePolicy: Random
  healthCheckPolicyName: test-all
  instanceTypes:
  - C3.LARGE8
  subnetIDs:
  - subnet-xxxxxxxx
  - subnet-yyyyyyyy
  scaling:
    createPolicy: ZonePriority
    maxReplicas: 100
  template:
    spec:
      displayName: mtest
      runtimeRootDir: /var/lib/containerd
      unschedulable: false
      metadata:
        labels:
          key1: "val1"
          key2: "val2"
      providerSpec:
        type: Native
        value:
          instanceChargeType: PostpaidByHour
          lifecycle:
            preInit: "echo hello"
            postInit: "echo world"
          management:
            hosts:
            - Hostnames:
              - test
              IP: 22.22.22.22
            nameservers:
            - 183.60.83.19
            - 183.60.82.98
            - 8.8.8.8
          metadata:
            creationTimestamp: null
          securityGroupIDs:
          - sg-xxxxxxxx
          systemDisk:
            diskSize: 50
            diskType: CloudPremium
```

Parameters

Parameter Module	Parameter	YAML Field	Note
Launch Configuration	Node Pool Type	Field name: spec.type Field value: Native	Nativ node
	Node Pool Name	Field name: spec.displayname Field value: demo-machineset (custom)	Cust it bas and facili reso
	Billing Mode	Field name: spec.template.spec.providerSpec.value.instanceChargeType Field value: PostpaidByHour (pay-as-you-go)/PrepaidCharge (monthly subscription)	Both mont supp acco need
	Model Configuration	**Model:** Field name: spec.instanceTypes Field value: S2.MEDIUM4 (Refer to the console for other model specifications.) **System Disk:** Field name: spec.template.spec.providerSpec.value.systemDisk.diskSize/diskType Field value: diskSize: 50 (Customizable. The size must be a multiple of 10 and the minimum value is 50 GB.) diskType: CloudPremium/CloudSSD (System disk type. The options include Premium Cloud Disk and SSD.)	Sele by re infor Conf Avail insta unde zone Mode filtere mem type. Syst the s stora sche size be g
	Data disk	Field name: spec.template.spec.providerSpec.value.dataDisks Field value: diskSize: same as system disk diskType: same as system disk fileSystem: ext3/ext4/xfs	This data form.

		mountTarget: /var/lib/containerd (mount path)	
	Public network bandwidth	Field name: spec.template.spec.providerSpec.value.internetAccessible Field value: For details, see Enabling Public Network Access for Native Nodes	Publ enab acce EIP. Enab Acce
	hostname	Display field: metadata.annotation key: "node.tke.cloud.tencent.com/hostname-pattern" value: "custom"	Com oper the ir used will b host Nam 1. Ba and s nami name chara chara lowe hyph Sym the b must cons 2. Th Cust pool indic insta this a the c work hostn batcl pool ID.1, Note durir node clust

			nami manu
	SSH Key	Field name: spec.template.spec.providerSpec.value.keyIDs Field value: skey-asxxxx (SSH key ID)	The If the suita
	Security Group	Field name: spec.template.spec.providerSpec.value.securityGroupIDs Field value: sg-a7msxxx (security group ID)	By d set d clust secu creat
	Quantity	Field name: spec.replicas Field value: 7 (custom)	Expe main corre it acc neec to 5, creat node
	Container Network	Field name: spec.subnetIDs Field value: subnet-i2ghxxxx (container subnet ID)	Sele avail your 1. W the n will t acco subn subn orde creat alwa subn 2. If a for th be cr subn scali confi
OPS feature	Fault Self-Healing	Field name: spec.autoRepair Field value: true (enabled)/false (disabled)	Optic to en featu anon

			real- self-l inclu kube
	Check and Self-healing Rules	Field name: spec.healthCheckPolicyName Field value: test-all (binding fault self-healing CR name)	You self-l pools howe only
	Auto Scaling	Field name: spec.scaling	After for th comp perfo pool. Rem featu deve conta featu relies featu
	Node Quantity Range	Field name: spec.scaling.maxReplicas/minReplicas Field value: maxReplicas: 7 (custom) minReplicas: 2 (custom)	The the n rang value enab quan be a withi
	Scaling Policy	Field name: spec.scaling.createPolicy Field value: ZonePriority (preferred availability zone first)/ ZoneEquality (distribute among multiple availability zones)	1. Pr first: prior your zone not p zone perfo zone 2. Di avail made node

			spec zone in the polic multi confi
Advanced Parameters	Labels	Field name: spec.template.spec.metadata.labels Field value: key1: "value1" (The label's key/value is customizable.)	Node filteri node will b for th this r
	Taints	Field name: spec.template.spec.metadata.taints Field value: effect: NoSchedule/PreferNoSchedule/NoExecute (Fill in the type of taints.)	Node typic with that p cond sche confi autor node node
	Container Directory	Field name: spec.template.spec.runtimeRootDir Field value: /var/lib/containerd	Chec and i for e
	Management	Field name: spec.template.spec.providerSpec.value.management.**kubeletArgs/kernelArgs/hosts/nameservers** Field value: For details, see Management Parameter Description.	The **Kern Nam** are c see Mana Desc
	Custom Script	Field name: spec.template.spec.providerSpec.value.lifecycle.preInit/postInit Field value: preInit: "echo hello" (Script executed before node initialization, which is customizable.) postInit: "echo world" (Script executed after node initialization, which is customizable.)	Spec confi scrip **initi node** provi reen The gene

			view /usr/l （Pc

Deleting Native Nodes

Last updated：2024-06-27 11:09:15

This document describes how to delete native nodes from node pools.

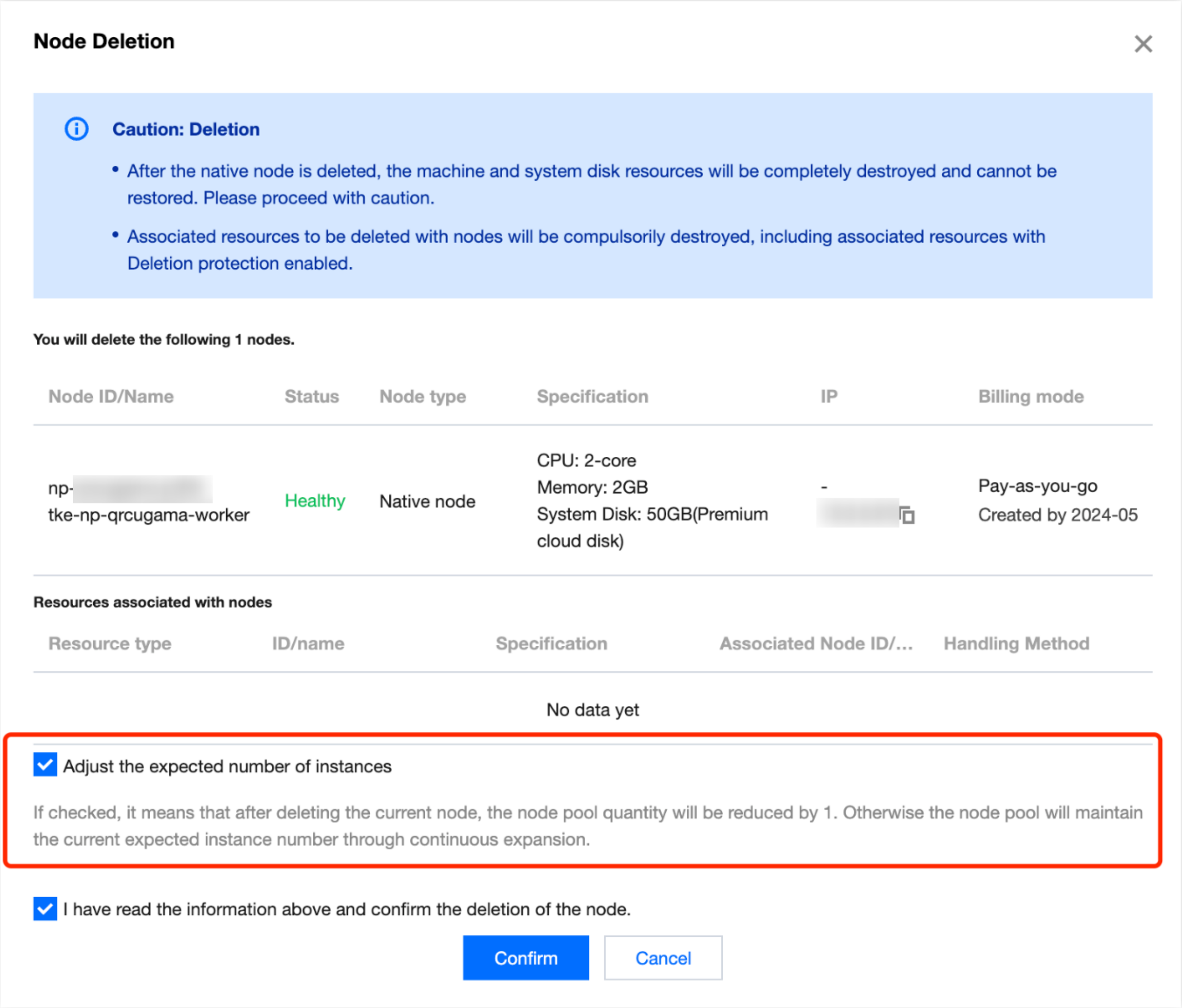
Note

Native nodes support declarative management. If you delete nodes while not modifying the required number of nodes in a node pool, new nodes are created immediately after you delete nodes.
Native nodes are managed in node pools and cannot be managed directly in clusters.
Deleting a native node releases all the resources on the node and does not support restoration. We recommend that you proceed with caution.

Directions

Pay-as-you-go node pool

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Select **Node Management** > **Worker Node** > **Node Pool** in the left sidebar , and click the node pool ID in the node pool list to enter the node pool details page.
4. In the node list, select the target node and then click **Delete** . In the pop-up window, check **Adjust the expected number of instances** (If checked, it means that after deleting the current node, the node pool quantity will be reduced by 1. Otherwise the node pool will maintain the current expected instance number through continuous expansion).



Node Deletion

Caution: Deletion

- After the native node is deleted, the machine and system disk resources will be completely destroyed and cannot be restored. Please proceed with caution.
- Associated resources to be deleted with nodes will be compulsorily destroyed, including associated resources with Deletion protection enabled.

You will delete the following 1 nodes.

Node ID/Name	Status	Node type	Specification	IP	Billing mode
np-[redacted] tke-np-qrcugama-worker	Healthy	Native node	CPU: 2-core Memory: 2GB System Disk: 50GB(Premium cloud disk)	- [redacted]	Pay-as-you-go Created by 2024-05

Resources associated with nodes

Resource type	ID/name	Specification	Associated Node ID/...	Handling Method
No data yet				

☑ Adjust the expected number of instances

If checked, it means that after deleting the current node, the node pool quantity will be reduced by 1. Otherwise the node pool will maintain the current expected instance number through continuous expansion.

☑ I have read the information above and confirm the deletion of the node.

5. Click **Confirm** to complete the node deletion.

Note:

For the pay-as-you-go node pool, when no specific nodes are designated for deletion and you choose to reduce the number of nodes directly by clicking **Adjust quantity**, the system will **randomly delete** redundant node replicas.

Monthly Subscription Type

1. Log in to the TKE console, and select **Clusters** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Select **Node Management > Worker Node > Node Pool** in the left sidebar , and click the node pool ID in the node pool list to enter the node pool details page.
4. In the node list, click **Adjust quantity** , to reduce the number of nodes in the pop-up window. In the subsequent confirmation window and refund window, separately click **Confirm** to complete the deletion of monthly subscription nodes. For example, if the current node pool has 5 native monthly subscription nodes and you wish to delete 2 specific nodes, first adjust the quantity to 3, and then select 2 monthly subscription nodes from the current node list for deletion.

5. In the node list, you can also select a monthly subscription node or a batch of nodes to be deleted, and then click **Delete** .

Note:

Since monthly subscription nodes involve refund processes, they cannot be deleted randomly like pay-as-you-go nodes. You must confirm the pop-up window information before deletion.

Refunds for monthly subscription nodes before expiration will be calculated according to the pay-as-you-go price. Proceed with caution.

Handling of Resources Associated with Nodes

Machines and system disks: After the native node is deleted, the machine and system disk resources will be completely terminated and cannot be restored. Proceed with caution.

EIP: If your node was configured with public network bandwidth at the time of creation, the bound EIP will be terminated along with the deletion of the node and cannot be restored. Proceed with caution.

Data disk: If your node was bound to a data disk at the time of creation, it will be terminated along with the deletion of the node and cannot be restored. It is recommended to back up your data in advance before deletion.

Self-Heal Rules

Last updated：2023-05-05 11:05:32

Overview

The instability of infrastructure and uncertainty of environment often trigger system failures at different levels. To relieve the Ops workload, the Tencent Kubernetes Engine (TKE) team has developed the self-heal feature for the Node-Problem-Detector-Plus add-on to help Ops engineers locate system exceptions and take minimal self-heal actions for various check items based on preset experiential Ops rules. Characteristics of the self-heal feature:

The system detects persistent faults that require human intervention in real time.

The scope of detection includes dozens of check items, such as check items on the operating system, Kubernetes environment, and runtime.

The feature quickly responds to faults based on preset experiential rules, such as executing a fix script and rebooting an add-on.

Check Items

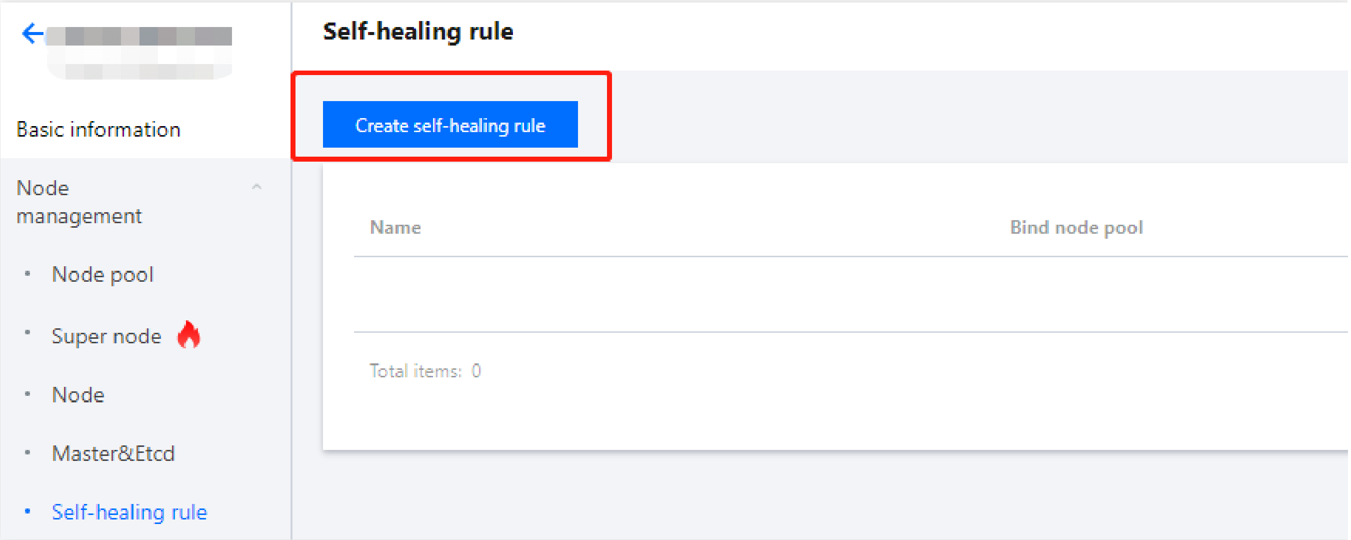
Check Item	Description	Risk Level	Self-Heal Action
FDPressure	Too many files opened. This is to check whether the number of file descriptors of the server has reached 90% of the maximum value.	low	-
RuntimeUnhealthy	List containerd task failed	low	RestartRuntime
KubeletUnhealthy	Call kubelet healthz failed	low	RestartKubelet
ReadonlyFilesystem	Filesystem is readonly	high	-
OOMKilling	Process has been oom-killed	high	-
TaskHung	Task blocked more then beyond the threshold	high	-
UnregisterNetDevice	Net device unregister	high	-
KernelOopsDivideError	Kernel oops with divide error	high	-
KernelOopsNULLPointer	Kernel oops with NULL pointer	high	-

Ext4Error	Ext4 filesystem error	high	-
Ext4Warning	Ext4 filesystem warning	high	-
IOError	IOError	high	-
MemoryError	MemoryError	high	-
DockerHung	Task blocked more then beyond the threshold	high	-
KubeletRestart	Kubelet restart	low	-

Enabling the Self-Heal Feature for Nodes

Enabling the feature in the TKE console

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Fault self-heal rule** in the left sidebar to go to the **Fault self-heal rule list** page.
4. Click **Create rule** to create a new self-heal rule. See the figure below:



5. Return to the node pool list page.
6. Click the ID of the target node pool to go to the details page of the node pool.
7. In the **Ops information** section of the details page, click **Edit** to enable the self-heal feature for the node pool.
8. View the details of real-time fault detection in the **Ops records** section. If the status of a check item is **Failed**, the check item failed.

Enabling the feature by using YAML

1. Create self-heal rules.

Specify the YAML configuration file as follows and run the `kubectl ceate -f demo-HealthCheckPolicy.yaml` command to create self-heal rules for a cluster:

```
apiVersion: config.tke.cloud.tencent.com/v1
kind: HealthCheckPolicy
metadata:
  name: test-all
  namespace: cls-xxxxxxxx (the ID of the cluster)
spec:
  machineSetSelector:
    matchLabels:
      key: fake-label
  rules:
  - action: RestartKubelet
    enabled: true
    name: FDPressure
  - action: RestartKubelet
    autoRepairEnabled: true
    enabled: true
    name: RuntimeUnhealthy
  - action: RestartKubelet
    autoRepairEnabled: true
    enabled: true
    name: KubeletUnhealthy
  - action: RestartKubelet
    enabled: true
    name: ReadonlyFilesystem
  - action: RestartKubelet
    enabled: true
    name: OOMKilling
  - action: RestartKubelet
    enabled: true
    name: TaskHung
  - action: RestartKubelet
    enabled: true
    name: UnregisterNetDevice
  - action: RestartKubelet
    enabled: true
    name: KernelOopsDivideError
  - action: RestartKubelet
    enabled: true
    name: KernelOopsNULLPointer
  - action: RestartKubelet
    enabled: true
    name: Ext4Error
  - action: RestartKubelet
    enabled: true
    name: Ext4Warning
  - action: RestartKubelet
    enabled: true
    name: IOError
  - action: RestartKubelet
    enabled: true
    name: MemoryError
  - action: RestartKubelet
    enabled: true
    name: DockerHung
  - action: RestartKubelet
```

```
    enabled: true
    name: KubeletRestart
```

2. Enable the self-heal feature.

Set the value of the `MachineSet` parameter to `healthCheckPolicyName: test-all` in the YAML configuration file:

```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
```

```
spec:
  type: Hosted
  displayName: demo-machineset
  replicas: 2
  autoRepair: true
  deletePolicy: Random
  healthCheckPolicyName: test-all
  instanceTypes:
  - C3.LARGE8
  subnetIDs:
  - subnet-xxxxxxxx
  - subnet-yyyyyyyy
......
```

Declarative Operation Practice

Last updated：2024-06-27 11:09:15

Operations Supported by Kubectl

CRD Type	Operation
MachineSet	Creating a native node pool `kubectl create -f machineset-demo.yaml`
	Viewing the list of native node pools `kubectl get machineset`
	Viewing the YAML details of a native node pool `kubectl describe ms machineset-name`
	Deleting a native node pool `kubectl delete ms machineset-name`
	Scaling out a native node pool `kubectl scale --replicas=3 machineset/machineset-name`
Machine	Viewing native nodes `kubectl get machine`
	Viewing the YAML details of a native node `kubectl describe ma machine-name`
	Deleting a native node `kubectl delete ma machine-name`
HealthCheckPolicy	Creating a fault self-healing rule `kubectl create -f demo-HealthCheckPolicy.yaml`
	Viewing the list of fault self-healing rules `kubectl kubectl get HealthCheckPolicy`
	Viewing the YAML details of a fault self-healing rule `kubectl describe HealthCheckPolicy HealthCheckPolicy-name`
	Deleting a fault self-healing rule `kubectl delete HealthCheckPolicy HealthCheckPolicy-name`

Using CRD via YAML

MachineSet

For the parameter settings of a native node pool, refer to the Description of Parameters for Creating Native Nodes.

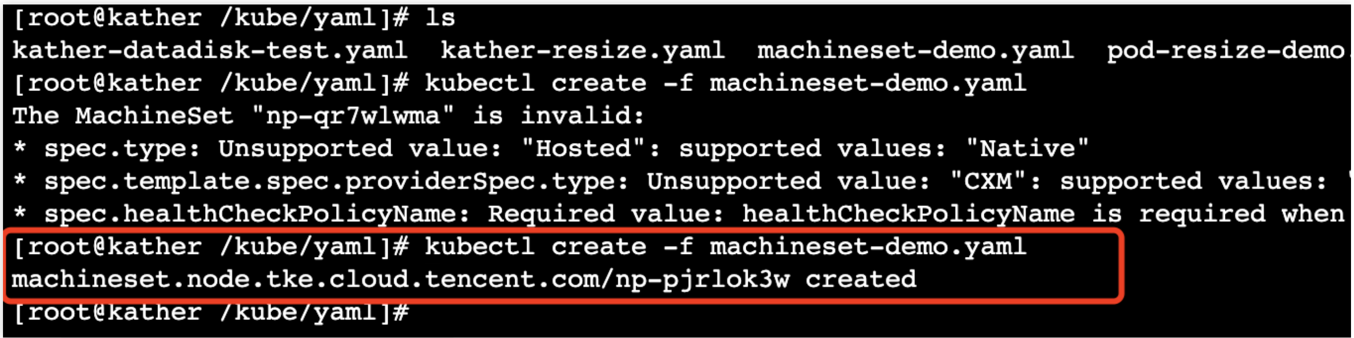
```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
```

```
spec:
  autoRepair: false #Fault self-healing switch
  displayName: test
  healthCheckPolicyName:  #Self-healing rule name
  instanceTypes: #Model specification
  - S5.MEDIUM2
  replicas: 1  #Node quantity
  scaling: #Auto-scaling policy
    createPolicy: ZonePriority
    maxReplicas: 1
  subnetIDs: #Node pool subnet
  - subnet-nnwwb64w
  template:
    metadata:
      annotations:
        node.tke.cloud.tencent.com/machine-cloud-tags: '[{"tagKey":"xxx","tagValue"
    spec:
      displayName: tke-np-mpam3v4b-worker #Custom display name
      metadata:
        annotations:
          annotation-key1: annotation-value1 #Custom annotations
        labels:
          label-test-key: label-test-value #Custom labels
      providerSpec:
        type: Native
        value:
          dataDisks: #Data disk parameters
          - deleteWithInstance: true
            diskID: ""
            diskSize: 50
            diskType: CloudPremium
            fileSystem: ext4
            mountTarget: /var/lib/containerd
          instanceChargeType: PostpaidByHour #Node billing mode
          keyIDs: #Node login SSH parameters
          - skey-xxx
          lifecycle: #Custom script
            postInit: echo "after node init"
            preInit: echo "before node init"
          management: #Settings of Management parameters, including kubelet\\kernel
          securityGroupIDs: #Security group configuration
          - sg-xxxxx
          systemDisk: #System disk configuration
            diskSize: 50
            diskType: CloudPremium
      runtimeRootDir: /var/lib/containerd
      taints: #Taints, not required
      - effect: NoExecute
        key: taint-key2
        value: value2
   type: Native
```

Kubectl Operation Demo

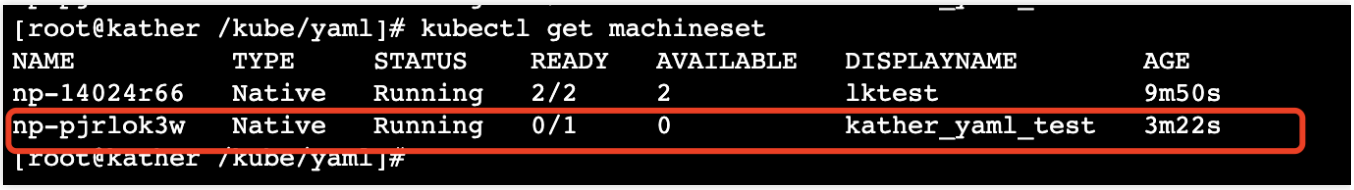
MachineSet

1. Run the `kubectl create -f machineset-demo.yaml` command to create a MachineSet based on the preceding YAML file.

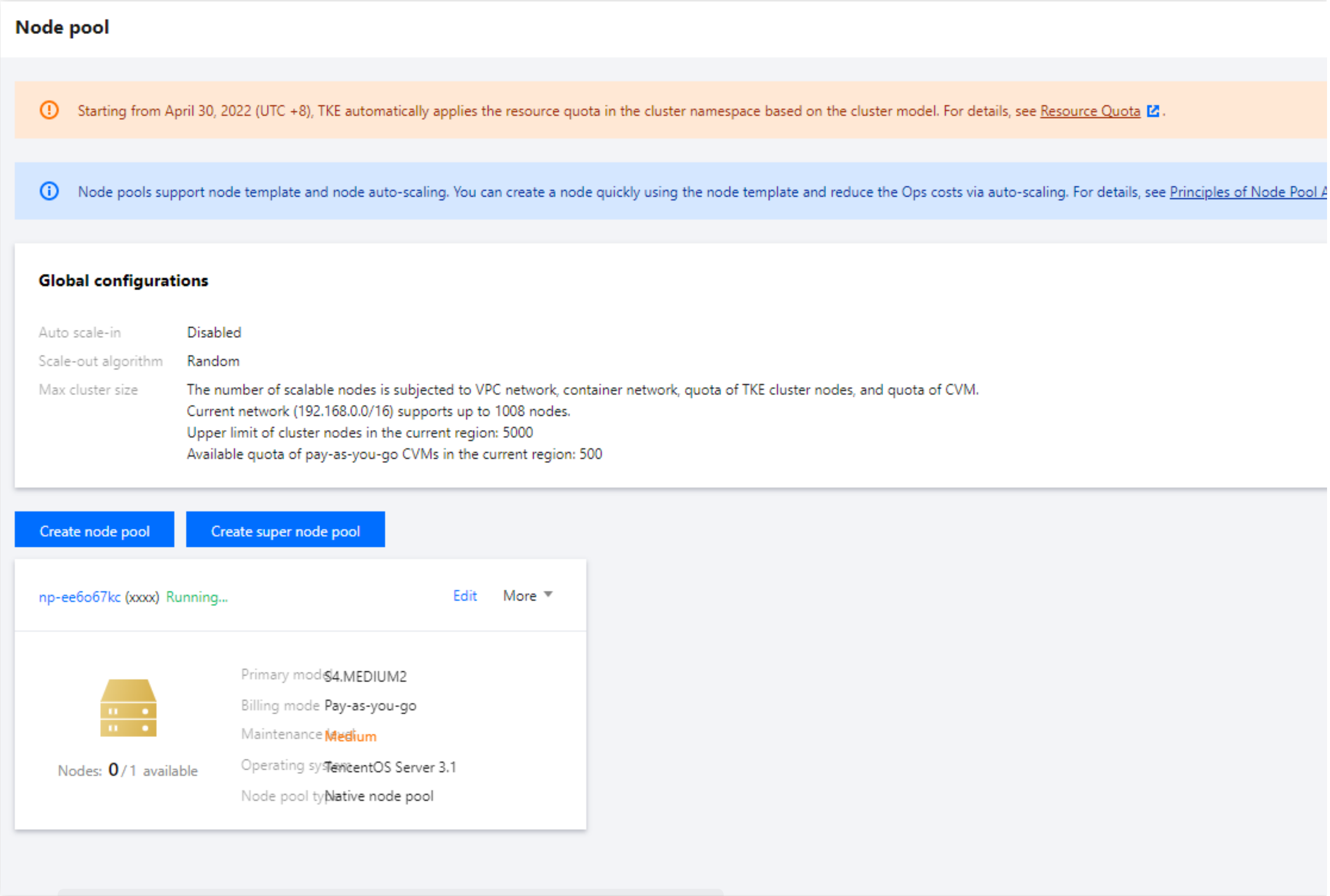


```
[root@kather /kube/yaml]# ls
kather-datadisk-test.yaml  kather-resize.yaml  machineset-demo.yaml  pod-resize-demo.
[root@kather /kube/yaml]# kubectl create -f machineset-demo.yaml
The MachineSet "np-qr7wlwma" is invalid:
* spec.type: Unsupported value: "Hosted": supported values: "Native"
* spec.template.spec.providerSpec.type: Unsupported value: "CXM": supported values:
* spec.healthCheckPolicyName: Required value: healthCheckPolicyName is required when
[root@kather /kube/yaml]# kubectl create -f machineset-demo.yaml
machineset.node.tke.cloud.tencent.com/np-pjrlok3w created
[root@kather /kube/yaml]#
```

2. Run the `kubectl get machineset` command to view the status of the MachineSet np-pjrlok3w. At this time, the corresponding node pool already exists in the console, and its node is being created.



```
[root@kather /kube/yaml]# kubectl get machineset
NAME          TYPE     STATUS    READY   AVAILABLE   DISPLAYNAME       AGE
np-14024r66   Native   Running   2/2     2           lktest            9m50s
np-pjrlok3w   Native   Running   0/1     0           kather_yaml_test  3m22s
[root@kather /kube/yaml]#
```



Node pool

Starting from April 30, 2022 (UTC +8), TKE automatically applies the resource quota in the cluster namespace based on the cluster model. For details, see Resource Quota .

Node pools support node template and node auto-scaling. You can create a node quickly using the node template and reduce the Ops costs via auto-scaling. For details, see Principles of Node Pool A

Global configurations

Auto scale-in Disabled

Scale-out algorithm Random

Max cluster size The number of scalable nodes is subjected to VPC network, container network, quota of TKE cluster nodes, and quota of CVM.
Current network (192.168.0.0/16) supports up to 1008 nodes.
Upper limit of cluster nodes in the current region: 5000
Available quota of pay-as-you-go CVMs in the current region: 500

np-ee6o67kc (xxxx) Running...

Nodes: 0 / 1 available

Primary mod S4.MEDIUM2

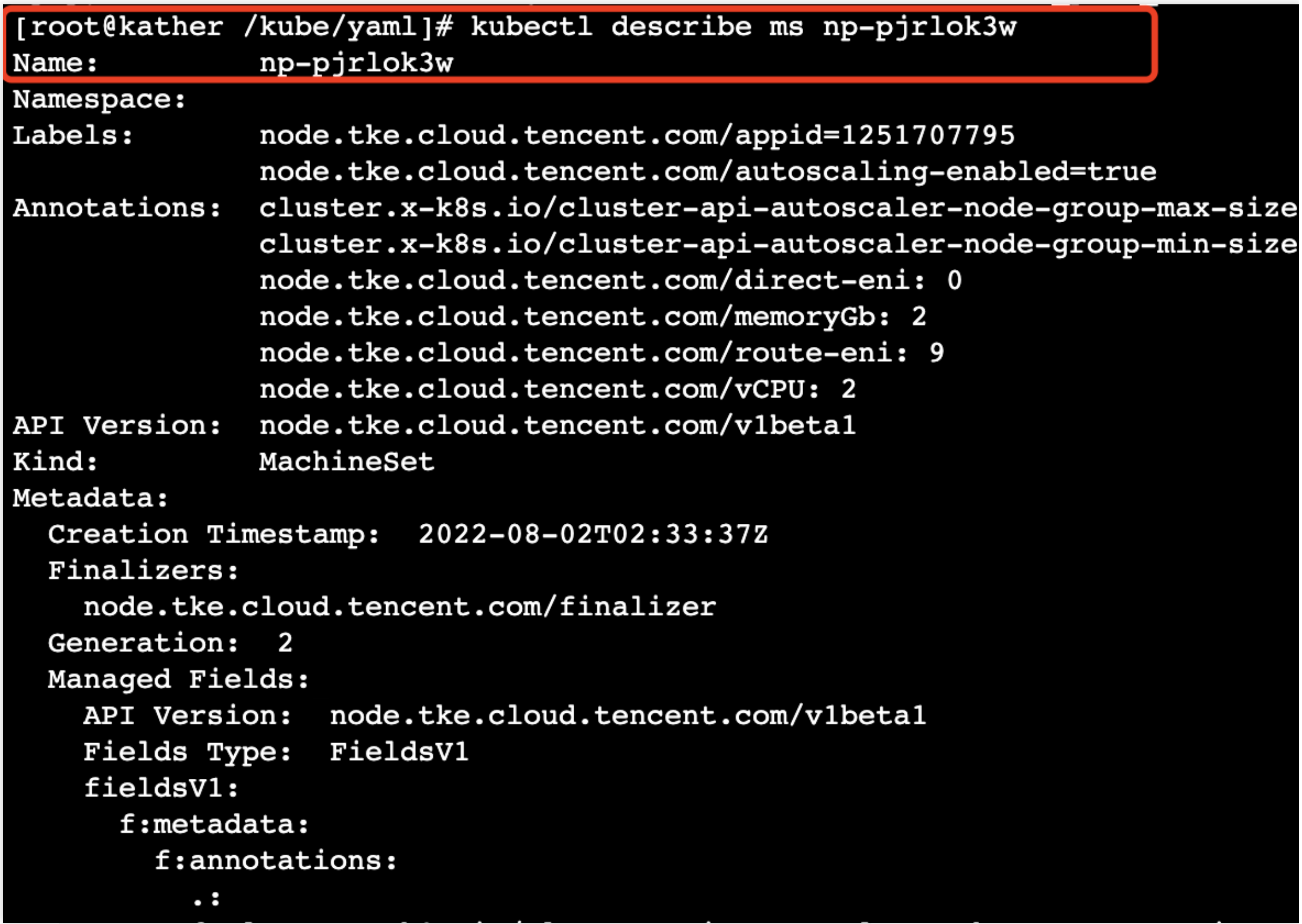
Billing mode Pay-as-you-go

Maintenance level Medium

Operating sys TencentOS Server 3.1

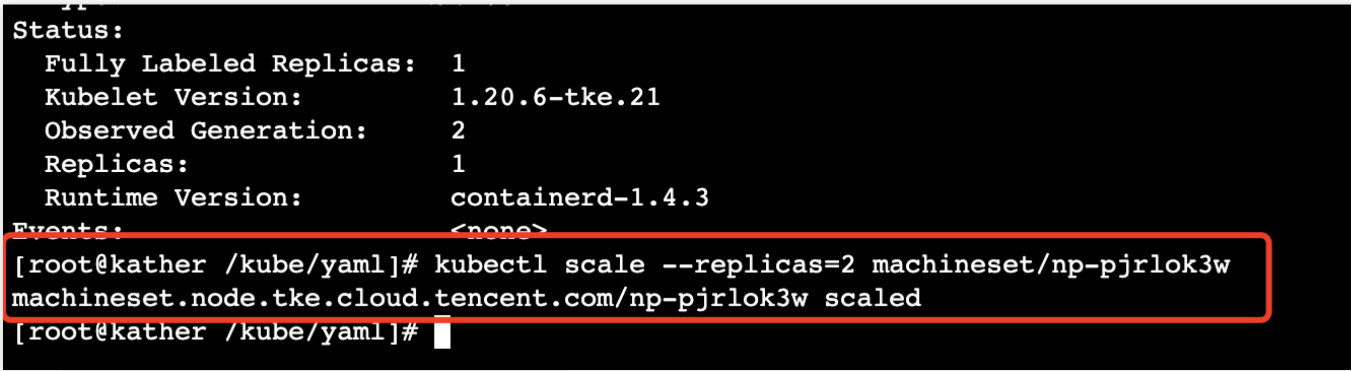
Node pool ty Native node pool

3. Run the `kubectl describe machineset np-pjrlok3w` command to view the description of the MachineSet np-pjrlok3w.



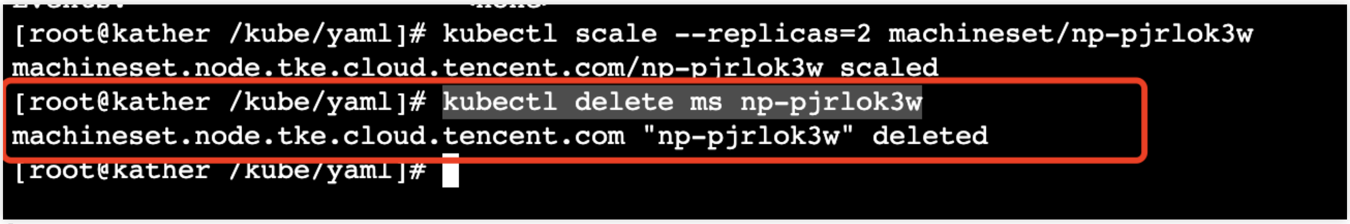
```
[root@kather /kube/yaml]# kubectl describe ms np-pjrlok3w
Name:         np-pjrlok3w
Namespace:
Labels:       node.tke.cloud.tencent.com/appid=1251707795
              node.tke.cloud.tencent.com/autoscaling-enabled=true
Annotations:  cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size
              cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size
              node.tke.cloud.tencent.com/direct-eni: 0
              node.tke.cloud.tencent.com/memoryGb: 2
              node.tke.cloud.tencent.com/route-eni: 9
              node.tke.cloud.tencent.com/vCPU: 2
API Version:  node.tke.cloud.tencent.com/v1beta1
Kind:         MachineSet
Metadata:
  Creation Timestamp:  2022-08-02T02:33:37Z
  Finalizers:
    node.tke.cloud.tencent.com/finalizer
  Generation:  2
  Managed Fields:
    API Version:  node.tke.cloud.tencent.com/v1beta1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .:
```

4. Run the `kubectl scale --replicas=2 machineset/np-pjrlok3w` command to execute scaling of the node pool.



```
Status:
  Fully Labeled Replicas:  1
  Kubelet Version:         1.20.6-tke.21
  Observed Generation:     2
  Replicas:                1
  Runtime Version:         containerd-1.4.3
Events:                    <none>
[root@kather /kube/yaml]# kubectl scale --replicas=2 machineset/np-pjrlok3w
machineset.node.tke.cloud.tencent.com/np-pjrlok3w scaled
[root@kather /kube/yaml]#
```

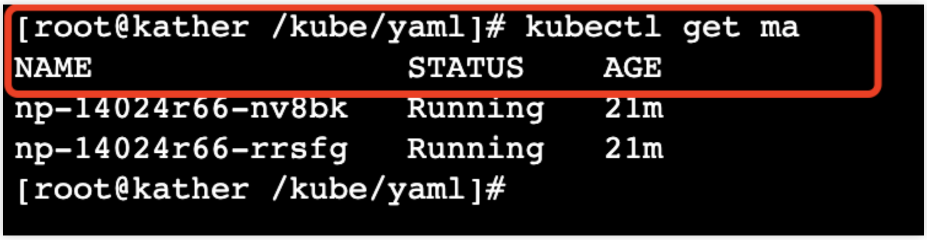
5. Run the `kubectl delete ms np-pjrlok3w` command to delete the node pool.



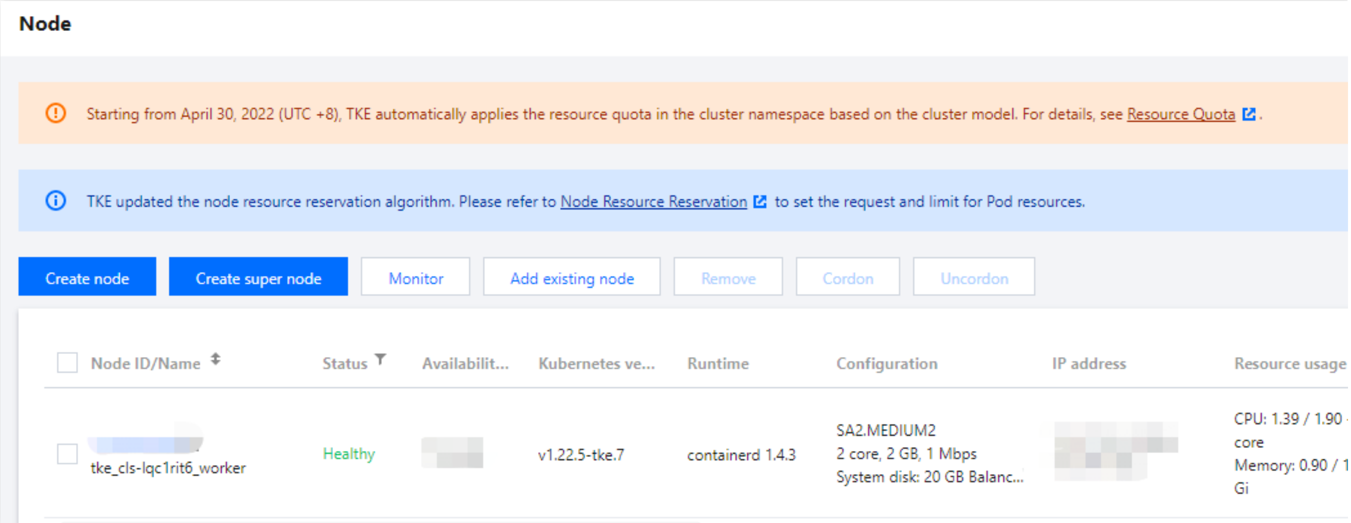
```
[root@kather /kube/yaml]# kubectl scale --replicas=2 machineset/np-pjrlok3w
machineset.node.tke.cloud.tencent.com/np-pjrlok3w scaled
[root@kather /kube/yaml]# kubectl delete ms np-pjrlok3w
machineset.node.tke.cloud.tencent.com "np-pjrlok3w" deleted
[root@kather /kube/yaml]#
```

Machine

1. Run the `kubectl get machine` command to view the machine list. At this time, the corresponding node already exists in the console.



```
[root@kather /kube/yaml]# kubectl get ma
NAME                   STATUS    AGE
np-14024r66-nv8bk      Running   21m
np-14024r66-rrsfg      Running   21m
[root@kather /kube/yaml]#
```



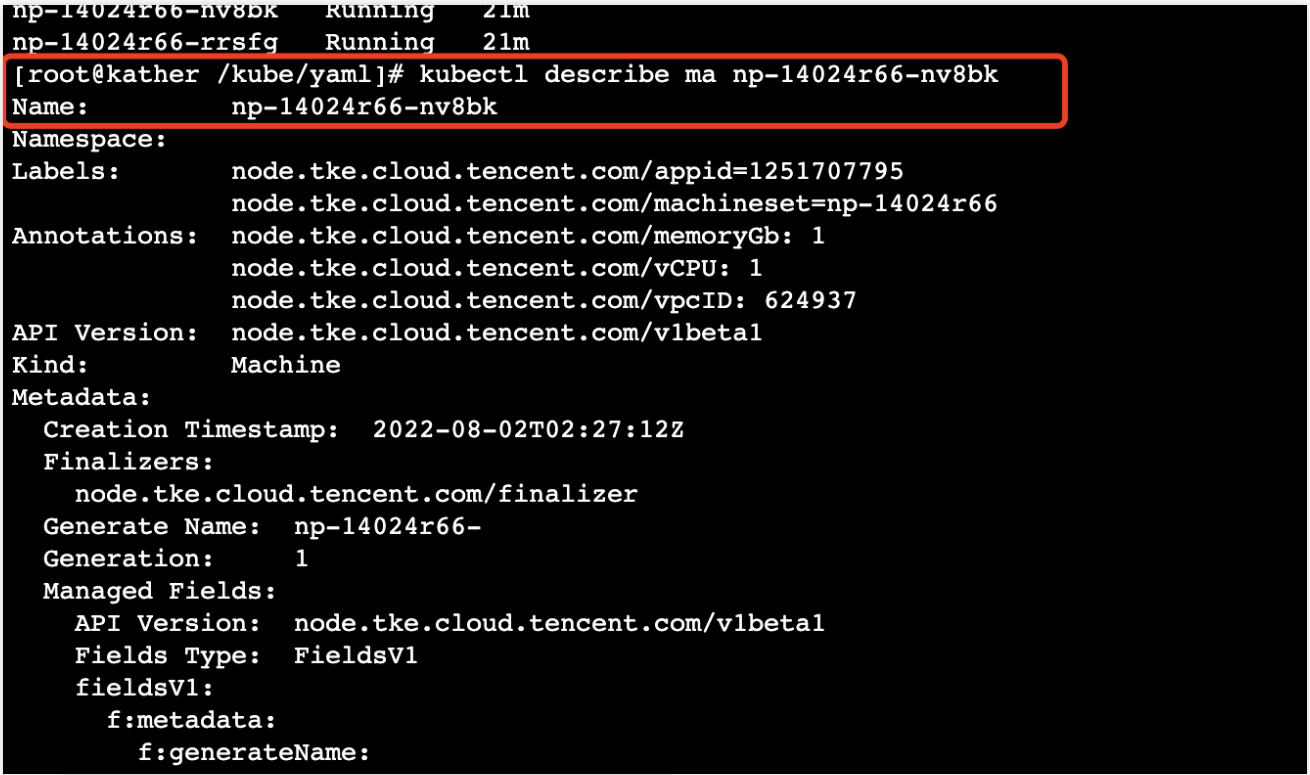
Node

Starting from April 30, 2022 (UTC +8), TKE automatically applies the resource quota in the cluster namespace based on the cluster model. For details, see Resource Quota.

TKE updated the node resource reservation algorithm. Please refer to Node Resource Reservation to set the request and limit for Pod resources.

□	Node ID/Name	Status	Availabilit...	Kubernetes ve...	Runtime	Configuration	IP address	Resource usage
□	[redacted] tke_cls-lqc1rit6_worker	Healthy	[redacted]	v1.22.5-tke.7	containerd 1.4.3	SA2.MEDIUM2 2 core, 2 GB, 1 Mbps System disk: 20 GB Balanc...	[redacted]	CPU: 1.39 / 1.90 core Memory: 0.90 / 1 Gi

2. Run the `kubectl describe ma np-14024r66-nv8bk` command to view the description of the machine np-14024r66-nv8bk.



```
np-14024r66-nv8bk   Running   21m
np-14024r66-rrsfg   Running   21m
[root@kather /kube/yaml]# kubectl describe ma np-14024r66-nv8bk
Name:         np-14024r66-nv8bk
Namespace:
Labels:       node.tke.cloud.tencent.com/appid=1251707795
              node.tke.cloud.tencent.com/machineset=np-14024r66
Annotations:  node.tke.cloud.tencent.com/memoryGb: 1
              node.tke.cloud.tencent.com/vCPU: 1
              node.tke.cloud.tencent.com/vpcID: 624937
API Version:  node.tke.cloud.tencent.com/v1beta1
Kind:         Machine
Metadata:
  Creation Timestamp:  2022-08-02T02:27:12Z
  Finalizers:
    node.tke.cloud.tencent.com/finalizer
  Generate Name:  np-14024r66-
  Generation:     1
  Managed Fields:
    API Version:  node.tke.cloud.tencent.com/v1beta1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:generateName:
```

3. Run the `kubectl delete ma np-14024r66-nv8bk` command to delete the node.

Note:

If you delete the node directly without adjusting the expected number of nodes in the node pool, the node pool will detect that the actual number of nodes does not meet the declarative number of nodes, and then create a new node and add it to the node pool. It is recommended to delete a node with the method as follows:

1. Run the `kubectl scale --replicas=1 machineset/np-xxxxx` command to adjust the expected number of nodes.

2. Run the `kubectl delete machine np-xxxxxx-dtjhd` command to delete the corresponding node.

Native Node Scaling

Last updated：2024-06-27 11:09:15

Note

The auto-scaling of a native node is implemented by Tencent Kubernetes Engine (TKE). The auto-scaling of a normal node relies on Auto Scaling (AS).

If auto-scaling is not enabled for a native node pool:

The number of initialized nodes is specified by the **Nodes** parameter in the console, or the `replicas` parameter in the YAML configuration file.

You can manually adjust the number of nodes as needed. However, the number of nodes is limited by the maximum number, which is 500 by default, and the number of IP addresses in the container subnet.

If auto-scaling is enabled for a native node pool:

The number of initialized nodes is specified by the **Nodes** parameter in the console, or the `replicas` parameter in the YAML configuration file.

You must specify the **Number of Nodes** parameter in the console, or the `minReplicas` and `maxReplicas` parameters in the YAML configuration file to set the range for the number of nodes. Cluster Autoscaler (CA) adjusts the number of nodes in the current node pool within the specified range.

You cannot manually adjust the number of nodes as needed.

Note:

At a same moment, the auto-scaling of the node pool can be controlled only by 1 role in the console. If auto-scaling is enabled, the instance quantity cannot be adjusted manually. If you wish to manually adjust the instance quantity, first disable auto-scaling.

Enabling the Auto-scaling Feature for Nodes

Parameter description

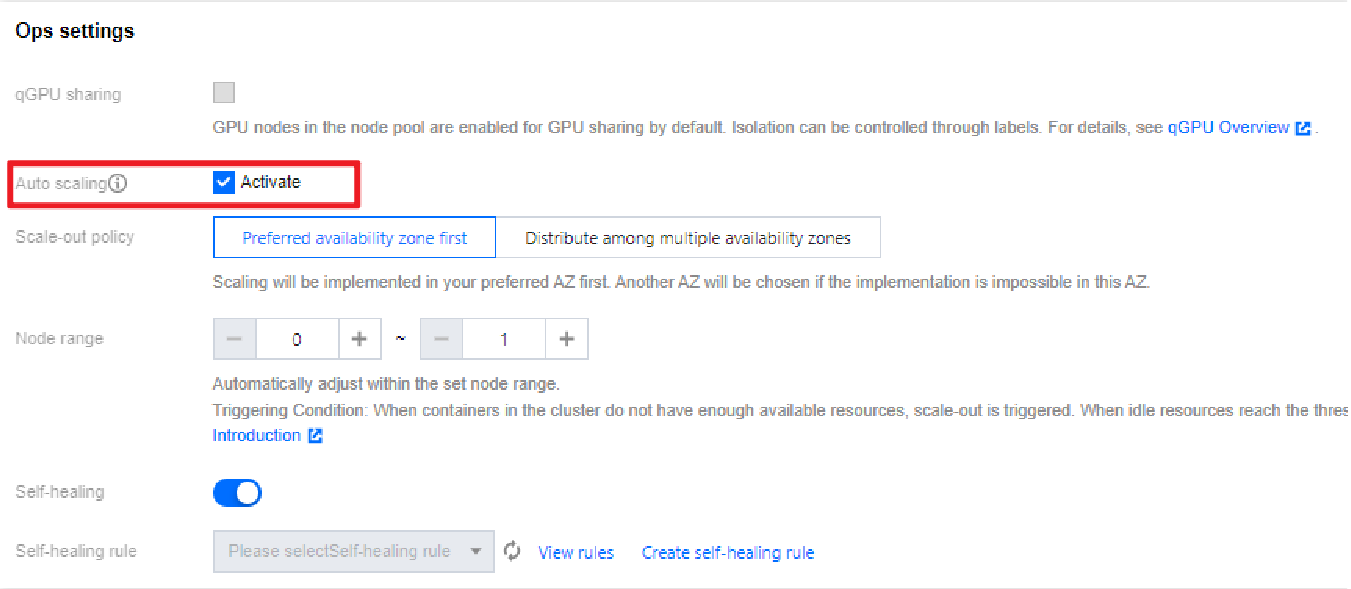
Function	Parameter and Values	Description
Auto Scaling	Parameter: spec.scaling	The auto-scaling feature is enabled by default. If the auto-scaling feature is enabled for a node pool, CA automatically scales in or out the node pool.
Number of Nodes	Parameter: spec.scaling.maxReplicas and spec.scaling.minReplicas	The number of nodes in the node pool cannot exceed the specified range. If auto-scaling is enabled for a node pool, the

	Valid values: The value is customizable.	number of native nodes in the node pool can be automatically adjusted within the specified range.
Scaling policy	Parameter: spec.scaling.createPolicy Example values: **Zone priority** in the console, or `ZonePriority` in the YAML configuration file. **Zone equality** in the console, or `ZoneEquality` in the YAML configuration file.	If you specify **Zone priority**, the auto-scaling feature performs scaling in the preferred zone first. If the preferred zone cannot be scaled, other zones are used. If you specify **Zone equality**, the auto-scaling feature distributes node instances evenly among the zones, or subnets, specified in the scaling group. This policy takes effect only if you have configured multiple subnets.

Enabling the feature in the TKE console

Method 1: Enabling auto-scaling on the node pool creation page

1. Log in to the TKE console and create a node pool in the cluster. For more information, see Creating Native Nodes.
2. On the **Create node pool** page, select **Enable** for **Auto-scaling**. See the following figure:



Ops settings

qGPU sharing □
GPU nodes in the node pool are enabled for GPU sharing by default. Isolation can be controlled through labels. For details, see qGPU Overview.

Auto scaling ☑ Activate

Scale-out policy: Preferred availability zone first | Distribute among multiple availability zones
Scaling will be implemented in your preferred AZ first. Another AZ will be chosen if the implementation is impossible in this AZ.

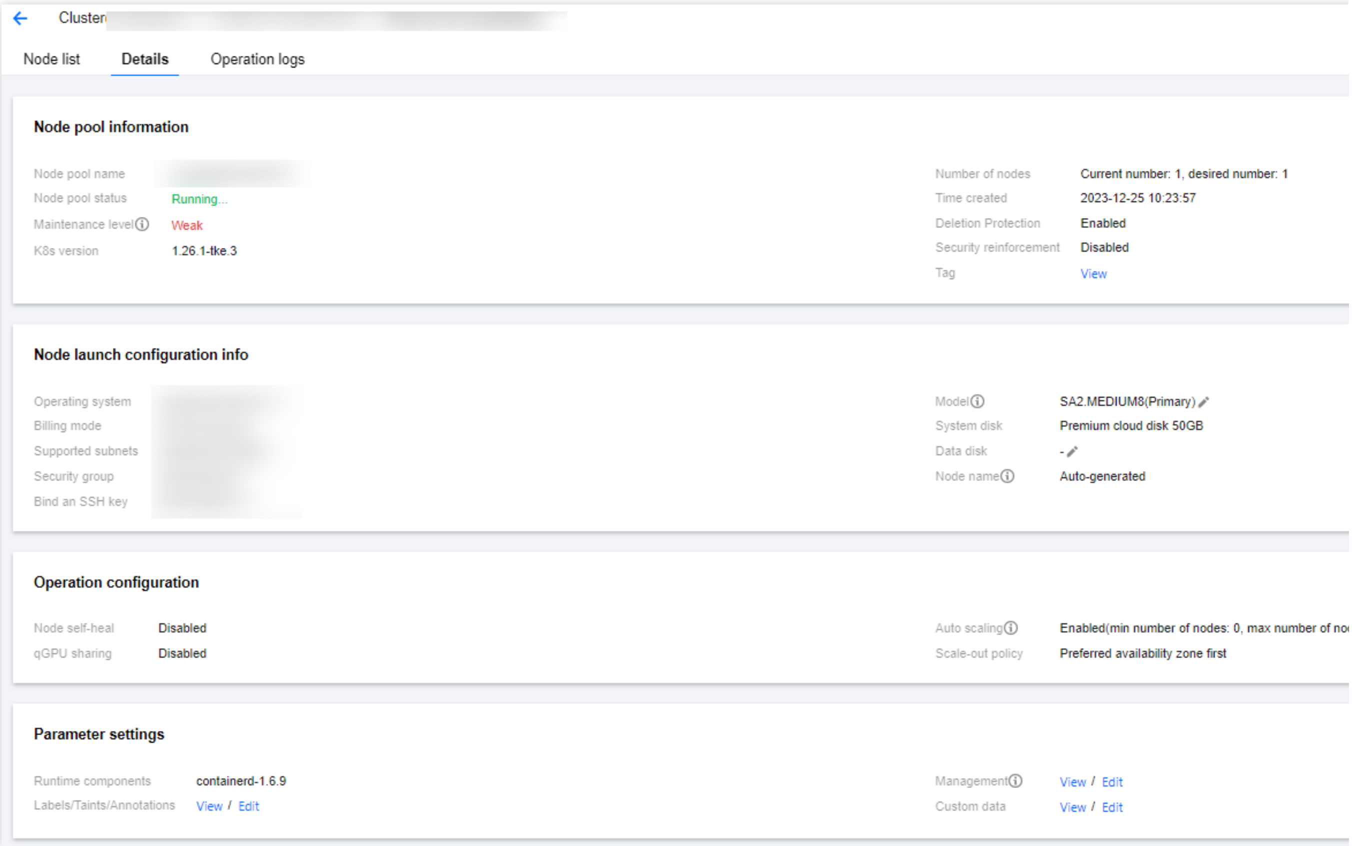
Node range: 0 ~ 1
Automatically adjust within the set node range.
Triggering Condition: When containers in the cluster do not have enough available resources, scale-out is triggered. When idle resources reach the thres
Introduction

Self-healing: on

Self-healing rule: Please selectSelf-healing rule   View rules   Create self-healing rule

Method 2: Enabling auto-scaling on the details page of a node pool

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Select **Node Management > Worker Node** in the left sidebar, and click the node pool ID in the **node pool** to enter the node pool details page.
4. On the node pool details page, click **Edit** on the right side of **Operation configuration**, as shown in the following figure:



Cluster [redacted]

Node pool information

Node pool name	[redacted]	Number of nodes	Current number: 1, desired number: 1
Node pool status	Running...	Time created	2023-12-25 10:23:57
Maintenance level ⓘ	Weak	Deletion Protection	Enabled
K8s version	1.26.1-tke.3	Security reinforcement	Disabled
		Tag	View

Node launch configuration info

Operating system	[redacted]	Model ⓘ	SA2.MEDIUM8(Primary)
Billing mode		System disk	Premium cloud disk 50GB
Supported subnets		Data disk	-
Security group		Node name ⓘ	Auto-generated
Bind an SSH key			

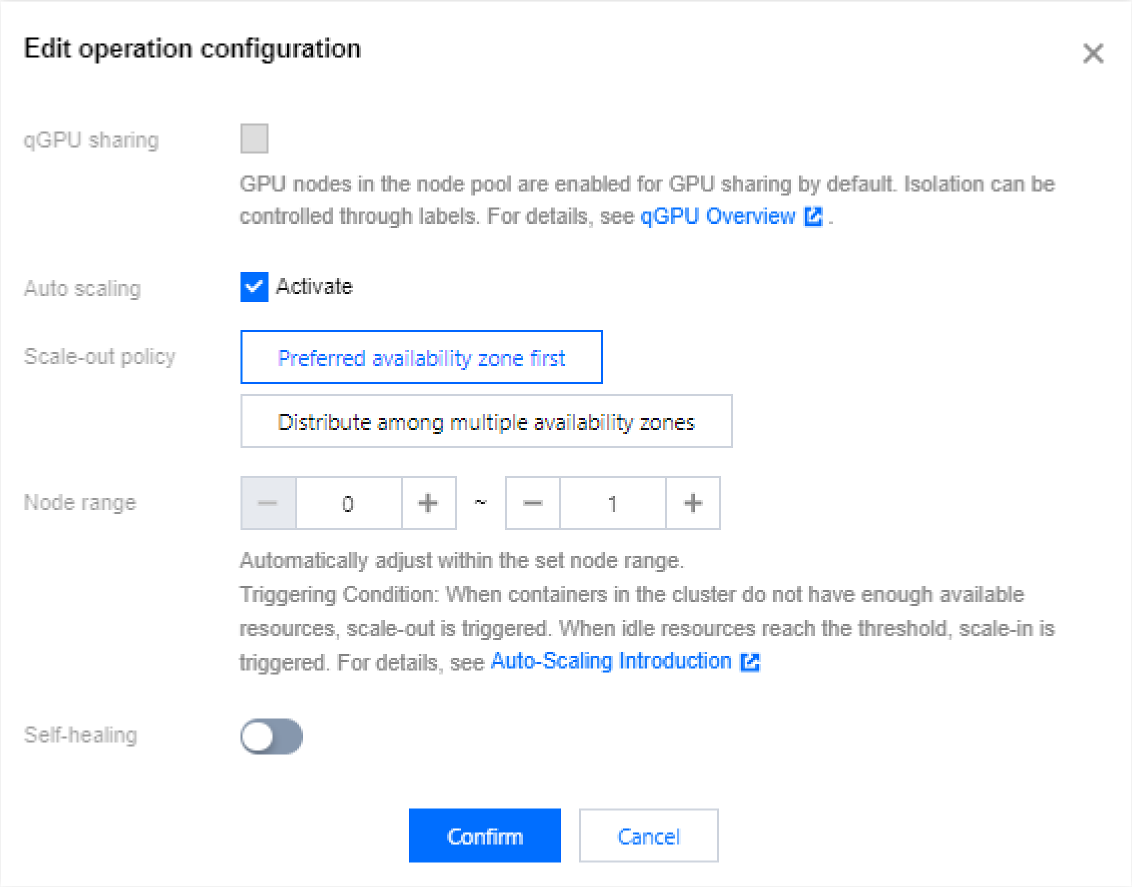
Operation configuration

Node self-heal	Disabled	Auto scaling ⓘ	Enabled(min number of nodes: 0, max number of no
qGPU sharing	Disabled	Scale-out policy	Preferred availability zone first

Parameter settings

Runtime components	containerd-1.6.9	Management ⓘ	View / Edit
Labels/Taints/Annotations	View / Edit	Custom data	View / Edit

5. Check **Activate Auto-scaling** , and click **Confirm** to enable auto-scaling.



Edit operation configuration

qGPU sharing □

GPU nodes in the node pool are enabled for GPU sharing by default. Isolation can be controlled through labels. For details, see qGPU Overview .

Auto scaling ☑ Activate

Scale-out policy: Preferred availability zone first / Distribute among multiple availability zones

Node range: 0 ~ 1

Automatically adjust within the set node range.
Triggering Condition: When containers in the cluster do not have enough available resources, scale-out is triggered. When idle resources reach the threshold, scale-in is triggered. For details, see Auto-Scaling Introduction

Self-healing

Enabling the feature by using YAML

Specify the `scaling` parameter in the YAML configuration file for a node pool.

```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  type: Native
  displayName: mstest
  replicas: 2
  autoRepair: true
  deletePolicy: Random
  healthCheckPolicyName: test-all
  instanceTypes:
  - C3.LARGE8
```

```
    subnetIDs:
    - subnet-xxxxxxxx
    - subnet-yyyyyyyy
    scaling:
      createPolicy: ZonePriority
      minReplicas: 10
      maxReplicas: 100
    template:
      spec:
        displayName: mtest
        runtimeRootDir: /var/lib/containerd
        unschedulable: false
  ......
```

Viewing the scaling records

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Node pool** in the left sidebar to go to the **Node pool list** page.
4. Click the ID of the target node pool to go to the details page of the node pool.
5. View the scaling records on the **Ops records** page.

Introduction to Scale-out Principles

This document will explain the scale-out principles of native nodes with examples under conditions of multiple models and multiple subnets.

Scenario 1: The scale-out policy is Preferred availability zone first when auto-scaling is enabled

Algorithm :

1. Determine the preferred availability zone based on the subnet arrangement sequence.
2. Select a model with the highest current inventory from multiple models for scale-out, and check the inventory in real time after scale-out of each machine, to ensure that the machine is scaled out successfully in the preferred availability zone as much as possible.

Example :

Assume that the node pool is configured with Models A/B (A has an inventory of 5 units, and B has an inventory of 3 units) and Subnets 1/2/3 (3 subnets are in different availability zones, with Subnet 1 being preferred). The arrangement sequences of models and subnets are valid during algorithm judgment. At this moment, CA triggers the scale-out of 10 machines in the node pool. The background judgment process is as follows:

2.1 Based on the subnet arrangement sequence, identify the subnet of the preferred availability zone as **Subnet 1**.
2.2 Check the real-time inventory status of all models, and scale out 1 node. Then repeat this process.
2.3 After the scale-out of 8 nodes, if no resources are available to continue scaling out in Subnet 1, proceed to Step 2.1 and switch the subnet of preferred availability zone to Subnet 2.

Scenario 2: The scale-out policy is Distribute among multiple availability zones when auto-scaling is enabled

Algorithm:
1. Based on the distribution status of the existing nodes within the node pool in the availability zones, determine the expected scale-out quantity for each availability zone, to ensure that the number of nodes distributed in each zone is as uniform as possible after scaling out.
2. After the availability zone is determined, select a model with the highest current inventory from multiple models for scale-out, and check the inventory in real time after scale-out of each machine, to ensure that the machine is successfully scaled out in the current availability zone as much as possible.
Example:
Assume that the node pool is configured with Models A/B (A has an inventory of 5 units, and B has 3 units) and Subnets 1/2/3 (3 subnets are in different availability zones, with Subnet 1 as preferred). The arrangement sequences of the models and subnets are valid during algorithm evaluation, and the node pool has 5 nodes which are deployed in Availability Zone 1. At this moment, CA triggers the scale-out of 10 machines in the node pool. The background judgment process is as follows:
2.1 Based on the deployment status of the existing nodes, it is expected to scale out 5 machines respectively in Availability Zones 2 and 3.
2.2 Based on the subnet sequence, identify the subnet of currently operated availability zone, namely **Subnet 2**.
2.2.1 Check the real-time inventory status of all models, and scale out 1 node. Then repeat this process.
2.2.2 After the scale-out in Availability Zone 2 is completed, proceed to Step 2.2 and switch the subnet of the availability zone to be scaled out currently to **Subnet 3**.

Scenario 3: Manually increase the number of node pools when auto-scaling is disabled

At this time, the default scale-out policy is **Distribute among multiple availability zones**, and the principle is as same as that of Scenario 2.

In-place Pod Configuration Adjustment

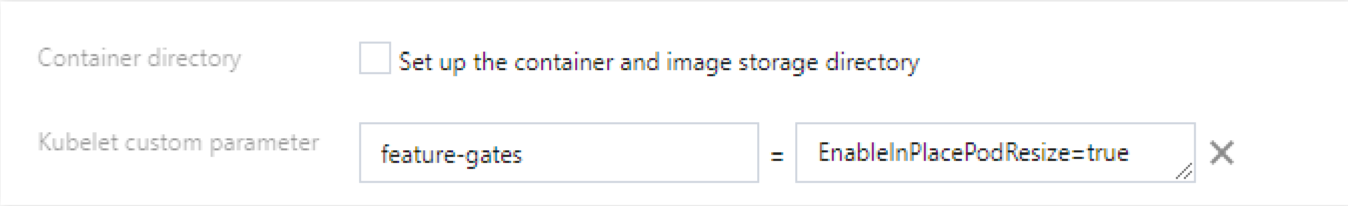
Last updated：2024-08-06 16:27:32

Overview

This document introduces the use cases, working principles, and usage modes of the in-place Pod resource update feature. According to the Kubernetes specifications, to modify container parameters when the Pod is running, you need to update the `PodSpec` and submit it again. This will delete and rebuild the Pod. TKE native nodes provide in-place Pod configuration adjustment capability, with which you can adjust the request/limit values of the Pod CPU and memory without restarting the Pod.

Prerequisites

This feature is available only to native nodes.
Supported cluster versions: Kubernetes v1.16 or later with minor versions as follows:
kubernetes-v1.16.3-tke.30 or later
kubernetes-v1.18.4-tke.28 or later
kubernetes-v1.20.6-tke.24 or later
kubernetes-v1.22.5-tke.15 or later
kubernetes-v1.24.4-tke.7 or later
kubernetes-v1.26.1-tke.2 or later
During node creation, set the custom kubelet parameter: **feature-gates** = **EnableInPlacePodResize=true**. See the figure below:



Container directory □ Set up the container and image storage directory

Kubelet custom parameter feature-gates = EnableInPlacePodResize=true ✕

Warning:
Adding the current feature-gates parameter to the existing node will **trigger restarting the Pod on the node** . It is recommended to evaluate the impact on the business before execution.

Use Cases

1. Deal with traffic burst and ensure business stability

Scenario: Dynamically modifying the Pod resource parameter is suitable for temporary adjustments. For example, when the memory usage of the Pod gradually increases, to avoid triggering Out Of Memory (OOM) Killer, you can increase the memory limit without restarting the Pod.
Solution: Increase the limit value of the CPU or memory.

2. Increase CPU utilization to reduce business costs

Scenario: To ensure the stability of online applications, admins usually reserve a considerable amount of resource buffers to cope with the load fluctuations of upstream and downstream links. The container's request configuration will be far higher than its actual resource utilization, resulting in low resource utilization of the cluster and a large amount of resource waste.
Solution: Decrease the request value of the CPU or memory.

Case Demo

Verification

Increase the memory's limit value of a running business Pod from 128Mi to 512Mi, and check that the new limit value takes effect and the Pod is not rebuilt.

Verification steps

1. Kubectl creates the `pod-resize-demo.yaml` file with the following content. The memory's request value is set to 64Mi and the limit value is 128Mi.

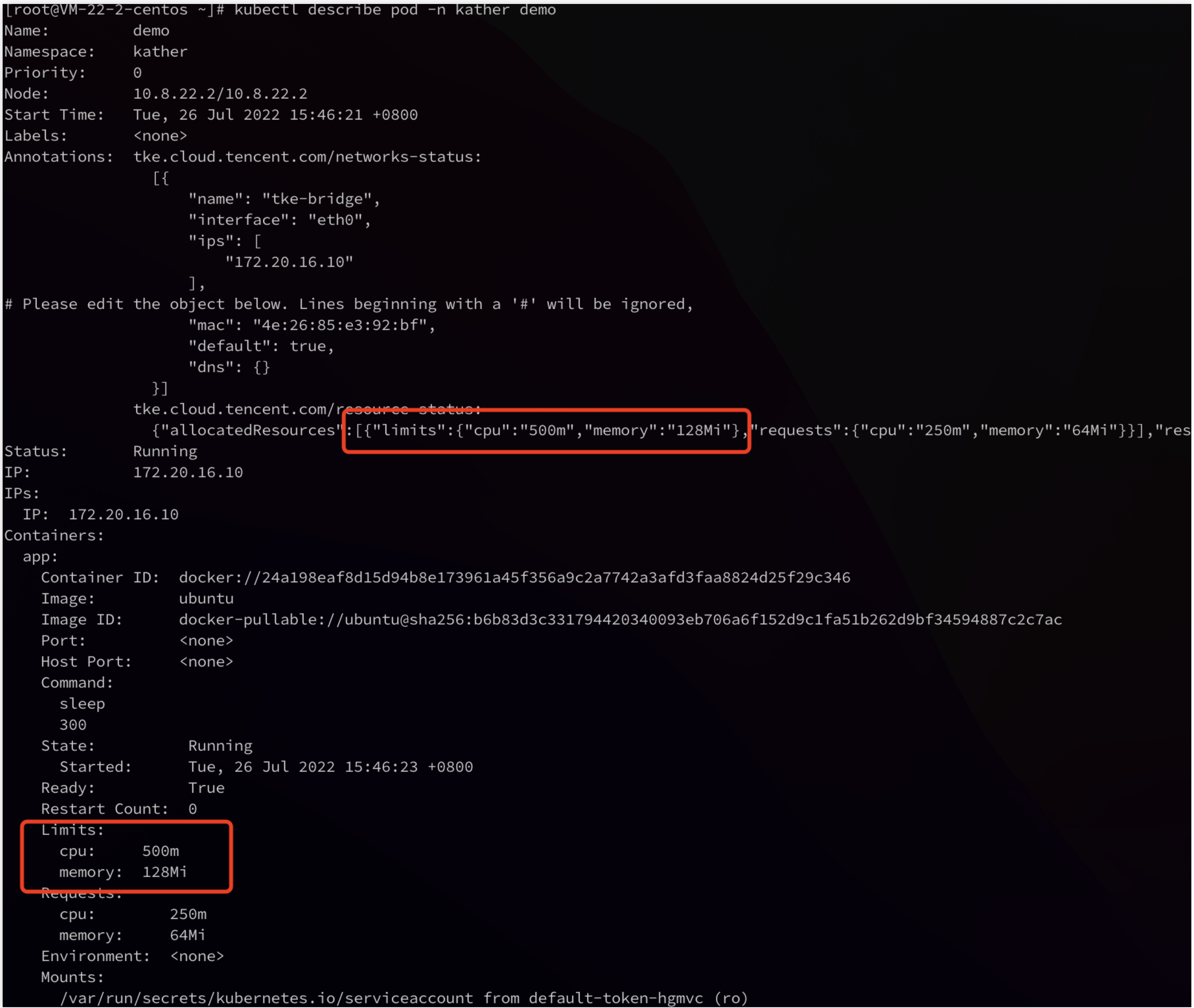
```
# Kubectl command:
kubectl apply -f pod-resize-demo.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: demo
  namespace: kather
spec:
  containers:
  - name: app
    image: ubuntu
    command: ["sleep", "3600"]
    resources:
      limits:
        memory: "128Mi"
        cpu: "500m"
      requests:
        memory: "64Mi"
        cpu: "250m"
```

2. Check the resource value of the Pod whose configuration is to be adjusted.

```
# Kubectl command:
kubectl describe pod -n kather demo
```

As shown in the figure below, the `Annotations` part of a Pod whose configuration is adjustable contains the **tke.cloud.tencent.com/resource-status** field, which marks the Pod's current in-use resources and configuration adjustment status. The expected Pod resource values will be marked on each container.



```
[root@VM-22-2-centos ~]# kubectl describe pod -n kather demo
Name:         demo
Namespace:    kather
Priority:     0
Node:         10.8.22.2/10.8.22.2
Start Time:   Tue, 26 Jul 2022 15:46:21 +0800
Labels:       <none>
Annotations:  tke.cloud.tencent.com/networks-status:
                [{
                    "name": "tke-bridge",
                    "interface": "eth0",
                    "ips": [
                        "172.20.16.10"
                    ],
# Please edit the object below. Lines beginning with a '#' will be ignored,
                    "mac": "4e:26:85:e3:92:bf",
                    "default": true,
                    "dns": {}
                }]
              tke.cloud.tencent.com/resource-status:
                {"allocatedResources":[{"limits":{"cpu":"500m","memory":"128Mi"},"requests":{"cpu":"250m","memory":"64Mi"}}],"res
Status:       Running
IP:           172.20.16.10
IPs:
  IP:  172.20.16.10
Containers:
  app:
    Container ID:  docker://24a198eaf8d15d94b8e173961a45f356a9c2a7742a3afd3faa8824d25f29c346
    Image:         ubuntu
    Image ID:      docker-pullable://ubuntu@sha256:b6b83d3c331794420340093eb706a6f152d9c1fa51b262d9bf34594887c2c7ac
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
      300
    State:          Running
      Started:      Tue, 26 Jul 2022 15:46:23 +0800
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     500m
      memory:  128Mi
    Requests:
      cpu:        250m
      memory:     64Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-hgmvc (ro)
```

3. Update Pod configuration.

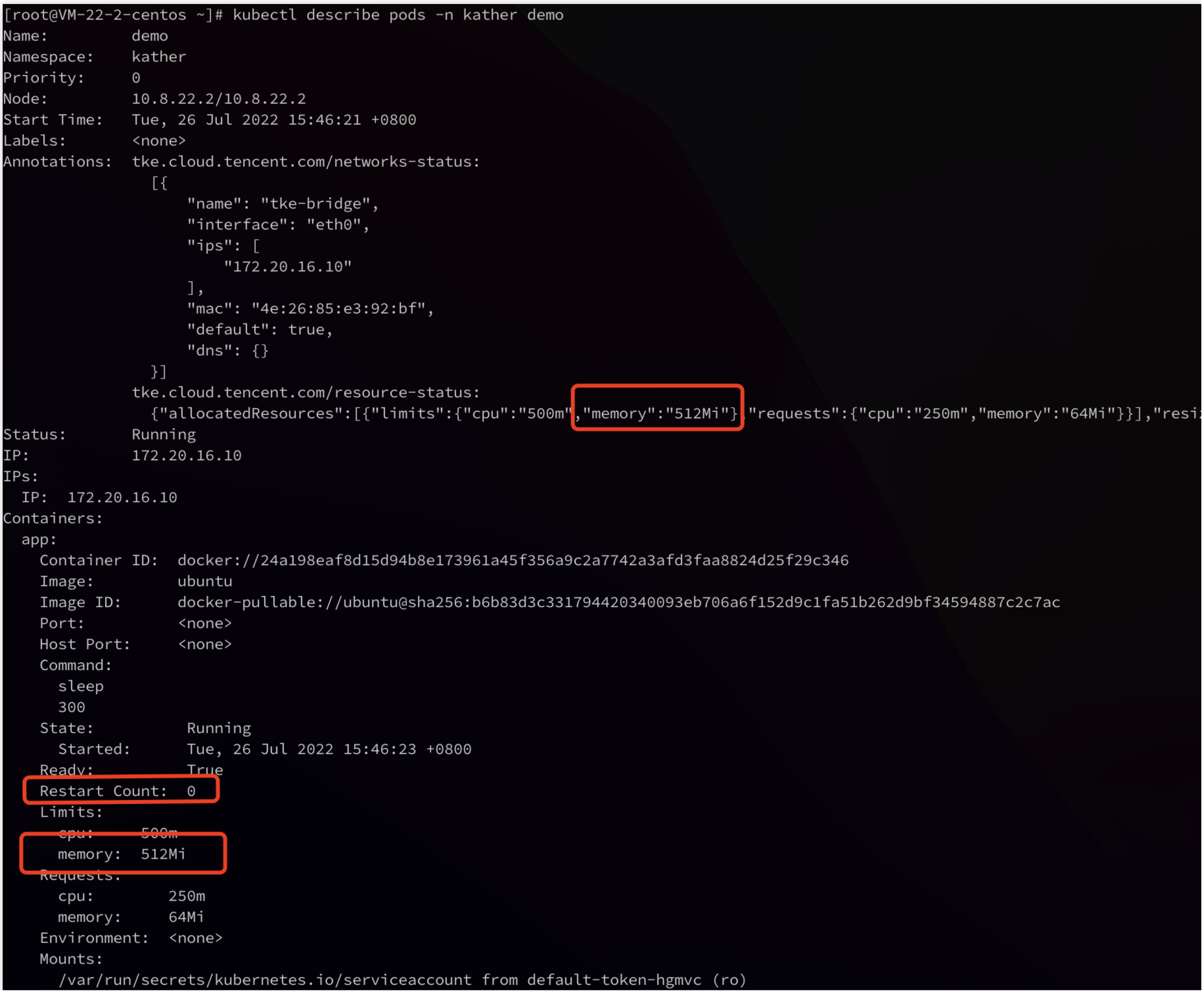
Assume that you want to increase the limit value of the Pod memory. Use kubectl to change the value of the `pod.spec.containers.resources.limits.memoy` field from 128Mi to 512Mi.

```
# Kubectl command:
kubectl edit pod -n kather demo
```

4. Run the following command to check the running status of the Pod.

```
# Kubectl command:
kubectl describe pod -n kather demo
```

As shown in the figure below, the memory values in `Pod spec` and `Annotations` are changed to the expected value "512Mi", and the value of `Restart Count` is 0.



```
[root@VM-22-2-centos ~]# kubectl describe pods -n kather demo
Name:         demo
Namespace:    kather
Priority:     0
Node:         10.8.22.2/10.8.22.2
Start Time:   Tue, 26 Jul 2022 15:46:21 +0800
Labels:       <none>
Annotations:  tke.cloud.tencent.com/networks-status:
                [{
                    "name": "tke-bridge",
                    "interface": "eth0",
                    "ips": [
                        "172.20.16.10"
                    ],
                    "mac": "4e:26:85:e3:92:bf",
                    "default": true,
                    "dns": {}
                }]
              tke.cloud.tencent.com/resource-status:
                {"allocatedResources":[{"limits":{"cpu":"500m","memory":"512Mi"},"requests":{"cpu":"250m","memory":"64Mi"}}],"resi
Status:       Running
IP:           172.20.16.10
IPs:
  IP:  172.20.16.10
Containers:
  app:
    Container ID:  docker://24a198eaf8d15d94b8e173961a45f356a9c2a7742a3afd3faa8824d25f29c346
    Image:         ubuntu
    Image ID:      docker-pullable://ubuntu@sha256:b6b83d3c331794420340093eb706a6f152d9c1fa51b262d9bf34594887c2c7ac
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
      300
    State:          Running
      Started:      Tue, 26 Jul 2022 15:46:23 +0800
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     500m
      memory:  512Mi
    Requests:
      cpu:        250m
      memory:     64Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-hgmvc (ro)
```

5. Verify the in-place Pod configuration adjustment.

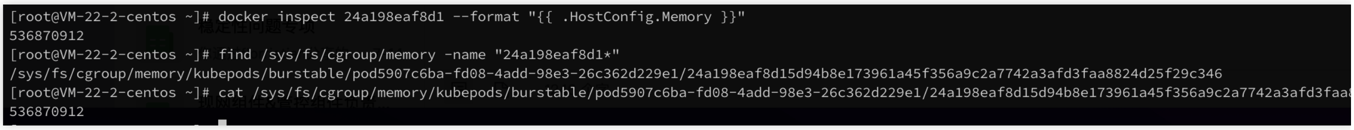
After you locate the container by using the `docker` or `ctr` command, you can see that the memory limit in the container metadata has been changed. If you enter `memory cgroup` , you will see that the memory limit has also been changed to the expected value "512Mi".

```
docker inspect <container-id> --format "{{ .HostConfig.Memory }}"
```

```
find /sys/fs/cgroup/memory -name "<container-id>*"
```

```
cat <container-memory-cgroup>/memory.limit_in_bytes
```

As shown below:



```
[root@VM-22-2-centos ~]# docker inspect 24a198eaf8d1 --format "{{ .HostConfig.Memory }}"
536870912
[root@VM-22-2-centos ~]# find /sys/fs/cgroup/memory -name "24a198eaf8d1*"
/sys/fs/cgroup/memory/kubepods/burstable/pod5907c6ba-fd08-4add-98e3-26c362d229e1/24a198eaf8d15d94b8e173961a45f356a9c2a7742a3afd3faa8824d25f29c346
[root@VM-22-2-centos ~]# cat /sys/fs/cgroup/memory/kubepods/burstable/pod5907c6ba-fd08-4add-98e3-26c362d229e1/24a198eaf8d15d94b8e173961a45f356a9c2a7742a3afd3faa
536870912
```

How it Works

1. Kubelet stores the Pod's current in-use resources and configuration adjustment status in JSON format in `tke.cloud.tencent.com/resource-status` in `Annotations` , where, the `resizeStatus` field indicates the configuration adjustment status. For more information, see Configuration Adjustment Status.

```
Annotations: tke.cloud.tencent.com/resource-status:
  {"allocatedResources":[{"limits":{"cpu":"500m","memory":"128Mi"},"requests":{"cpu
```

2. Resources in `pod.spec.containers.resources` are the expected resources, which are expected to be allocated to the Pod. If the current expected resource is modified, kubelet will try to modify the Pod's actual resource and write the final result to `Annotations` .

Note

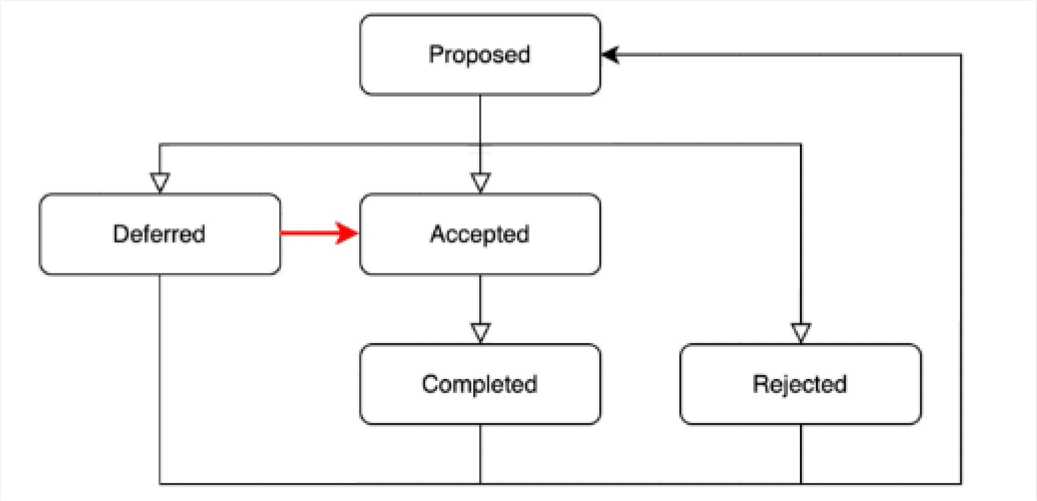
The implementation of in-place Pod configuration adjustment is based on Kubernetes Enhancement Proposal 1287.

Configuration Adjustment Status

The Kubernetes community adds some fields to `Pod.Status` in later versions to display the configuration adjustment status. This status works with Kube Scheduler to implement scheduling. Similar fields, including the Pod's actual resources and current configuration adjustment operation status, are added to Pod `Annotations` of TKE native nodes.

Status	Description	Remarks
Proposed	The Pod configuration adjustment request is submitted.	-
Accepted	Kubelet discovers that Pod resources have been modified and the node resources are sufficient to admit the Pod after the configuration adjustment.	-
Rejected	The Pod configuration adjustment request is rejected.	Rejection reason: The expected `requests` resource value after Pod configuration adjustment is greater than the node's `allocated` value.
Completed	The Pod resource is successfully modified, and the adjusted resource is set in the container.	-
Deferred	The current configuration adjustment is deferred due to certain issues and will be triggered again upon next Pod status change.	Possible issues: The current node resources are insufficient: Allocated amount of node resources - Amount of resources occupied by other Pods < Amount of resources required by the Pod whose configuration is to be adjusted. Failed to store the status.

The execution status of Pod configuration adjustment is as shown below:



Use Limits

To give priority to ensuring the stability of business Pod operation, it is necessary to carry out some operational restrictions on the in-place Pod configuration adjustment capability:

1. Allow only the CPU and memory resources of the Pod to be modified.
2. Allow Pod resource modification only when `PodSpec.Nodename` is not empty.
3. The resource modification scope is as follows:

The limit value of each container in a Pod can be increased or decreased: Decreasing the CPU limit value may cause traffic to slow down, and decreasing the memory limit value may fail (kubelet will retry decreasing memory in a subsequent syncLoop).

The request value of each container in the Pod can be increased or decreased, but the highest container request value cannot be greater than the container limit value.

4. For containers with no request or limit value specified:

For a container with no limit value specified, a new value cannot be set.

For a container with no request value specified, the lowest value cannot be smaller than 100m.

5. When a request or limit value is modified, the QoS type cannot be changed between burstable and guaranteed. You need to modify both the request and limit values to maintain the same QoS during configuration adjustment.

For example, [cpu-request: 30, cpu-limit: 50] can only be adjusted to [cpu-request: 49, cpu-limit: 50] and cannot be adjusted to [cpu-request: 50, cpu-limit: 50].

Enabling SSH Key Login for a Native Node

Last updated：2023-05-05 11:05:32

Overview

This document describes how to enable SSH key login, assign SSH keys, and modify SSH keys for node pools or native nodes.

Directions

For node pools

Assign an SSH key when creating a node pool

Modify the SSH key associated with an existing node pool

You can assign an SSH key to a node pool during the creation of the node pool. After a scale-out of the node pool, the SSH key is automatically assigned to the new nodes.

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Node pool** in the left sidebar. On the **Node pool list** page, click **Create node pool**.
4. On the **Create node pool** page, select **SSH Key**.
5. Click **Create node pool**.

Go to the details page of the node pool and modify the associated SSH key in the launch configuration section. The new SSH key takes effect only for new nodes in the node pool and does not affect the existing native nodes.

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Node pool** in the left sidebar. On the **Node pool list** page, click the ID of the node pool.
4. On the details page of the node pool, click ✏ next to **Associate SSH key** in the launch configuration section.
5. On the **Set node pool SSH key** page, select an SSH key.
6. Click **OK**.

For nodes

Enable or modify SSH key login for a node

Enable or modify node login for multiple nodes at a time

If no SSH key is assigned to a node pool, you can enable and modify SSH key login for a single node in the node pool.

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Node pool** in the left sidebar. On the **Node pool list** page, click the ID of the node pool.
4. On the details page of the node pool, choose **More** > **Node Login** in the row of the node.

You can assign an SSH key to multiple nodes at a time.

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Node pool** in the left sidebar. On the **Node pool list** page, click the ID of the node pool.
4. On the details page of the node pool, select multiple nodes as needed and choose **More** > **Batch set SSH key**.

Note:

1. Modifying the SSH key assigned to a node does not require the node to restart.
2. When you enable or modify node login for the first time or modify an SSH key that has been assigned to a node, the operation takes one to two minutes.
3. We recommend that the number of nodes for which you enable or modify SSH key login at a time do not exceed 20.
4. Batch enabling or modification of node login does not override the SSH keys that have been assigned to nodes.

Management Parameters

Last updated：2024-06-27 11:09:15

Overview

The Management parameters provide a unified entry for common custom configurations of nodes, which allow you to optimize the underlying kernel parameter KernelArgs for native nodes, and also supports setting Kubelet\\Nameservers\\Hosts to meet the environmental requirements of business deployment.

Management Parameters

Parameter Item	Description
KubeletArgs	Sets the Kubelet-related custom parameters required for business deployment.
Nameservers	Sets the DNS server address required for the business deployment environment.
Hosts	Sets the Hosts required for the business deployment environment.
KernelArgs	The business performance can be optimized by setting kernel parameters (if the parameters currently available for configuration do not meet your needs, you can submit a ticket for support).

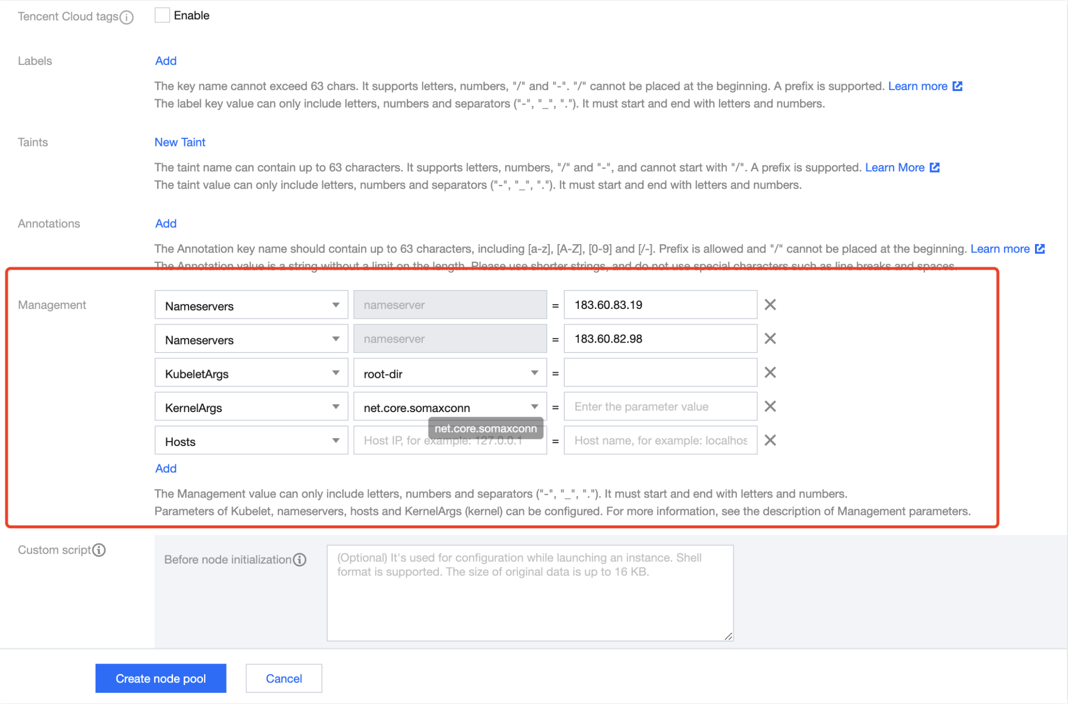
Note:

1. The Kubelet parameters supporting custom configuration are related to the cluster version. If the Kubelet parameters available in the console for the current cluster cannot meet your needs, submit a ticket for support.
2. To ensure the normal installation of system components, native nodes are by default injected with the official documentation library addresses of Tencent Cloud, namely `nameserver = 183.60.83.19` , and `nameserver = 183.60.82.98` .

Operation Using the Console

Method 1: Setting Management Parameters for a New Node Pool

1. Log in to the TKE console, and create a native node by reference to the Creating Native Nodes documentation.
2. On the **Create** page, click **Advanced Settings**, to set the Management parameters for the node, as shown in the figure below:



Tencent Cloud tags ⓘ □ Enable

Labels Add

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Taints New Taint

The taint name can contain up to 63 characters. It supports letters, numbers, "/" and "-", and cannot start with "/". A prefix is supported. Learn More
The taint value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Annotations Add

The Annotation key name should contain up to 63 characters, including [a-z], [A-Z], [0-9] and [/-]. Prefix is allowed and "/" cannot be placed at the beginning. Learn more
The Annotation value is a string without a limit on the length. Please use shorter strings, and do not use special characters such as line breaks and spaces.

Management

Nameservers	nameserver	=	183.60.83.19	✕
Nameservers	nameserver	=	183.60.82.98	✕
KubeletArgs	root-dir	=		✕
KernelArgs	net.core.somaxconn	=	Enter the parameter value	✕
Hosts	Host IP, for example: 127.0.0.1	=	Host name, for example: localhos	✕

Add

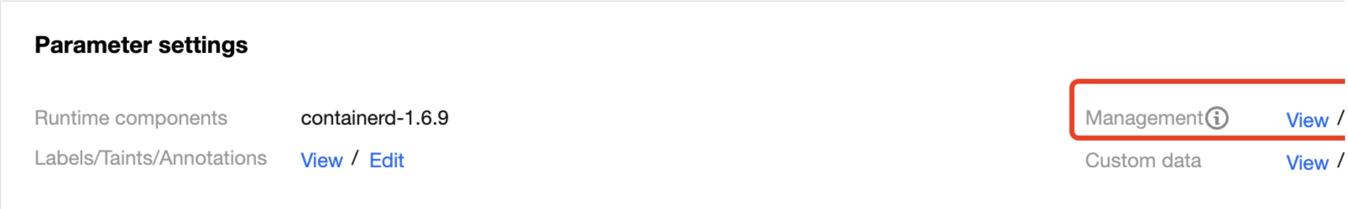
The Management value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.
Parameters of Kubelet, nameservers, hosts and KernelArgs (kernel) can be configured. For more information, see the description of Management parameters.

Custom script ⓘ Before node initialization ⓘ (Optional) It's used for configuration while launching an instance. Shell format is supported. The size of original data is up to 16 KB.

3. Click **Create node pool**.

Method 2: Setting Management Parameters for an Existing Node Pool

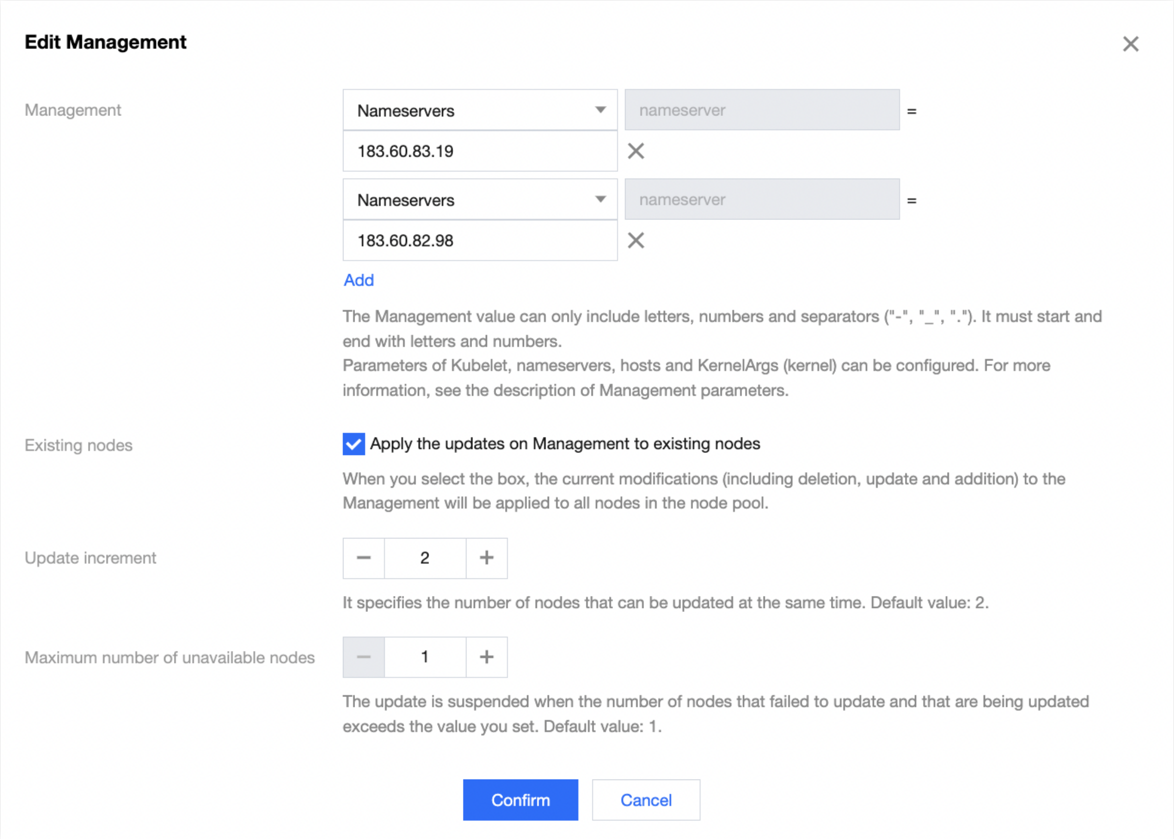
1. Log in to the TKE console, and select **Clusters** in the left sidebar.
2. On the cluster list page, click on the target cluster ID to enter the cluster details page.
3. Select **Node Management** > **Worker Node** in the left sidebar, and click on the node pool ID in the **Node Pool** to enter the node pool details page.
4. On the node pool details page, click **Parameter Settings** > **Management** > **Edit** to modify the Management parameters, as shown in the figure below:



Parameter settings

Runtime components containerd-1.6.9
Labels/Taints/Annotations View / Edit
Management ⓘ View /
Custom data View /

5. You can check **Apply the updates on management to existing nodes**, to apply the parameter changes to the existing nodes in the node pool. Once checked, the changes (including deletion, update, and addition) on Management will apply to all nodes (including the existing and new nodes) in the node pool, as shown in the figure below:
Update increment: The system will update Ops parameters in batches. This parameter defines the number of nodes that can be updated simultaneously in each batch.
Maximum number of unavailable nodes: If the number of nodes that failed to update (including those being updated) exceeds this set value, the system will pause the update operation.



Edit Management

Management: Nameservers nameserver = 183.60.83.19

Nameservers nameserver = 183.60.82.98

Add

The Management value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.
Parameters of Kubelet, nameservers, hosts and KernelArgs (kernel) can be configured. For more information, see the description of Management parameters.

Existing nodes: ☑ Apply the updates on Management to existing nodes

When you select the box, the current modifications (including deletion, update and addition) to the Management will be applied to all nodes in the node pool.

Update increment: 2

It specifies the number of nodes that can be updated at the same time. Default value: 2.

Maximum number of unavailable nodes: 1

The update is suspended when the number of nodes that failed to update and that are being updated exceeds the value you set. Default value: 1.

6. Click **Confirm**. You can view the update progress and results in the **Ops Records** of the node pool.

Warning:

Certain Kubelet parameters, when applied to the existing nodes, may trigger the restart of business Pods, such as "feature-gates: EnableInPlacePodResize=true", "kube-reserved: xxx", "eviction-hard: xxx", and "max-pods: xxx". It is recommended to operate with caution after assessing the risks.

Operation Using YAML

```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  autoRepair: false
  deletePolicy: Random
  displayName: xxxxxx
  healthCheckPolicyName: ""
  instanceTypes:
  - SA2.2XLARGE8
  replicas: 2
  scaling:
    createPolicy: ZoneEquality
  subnetIDs:
  - subnet-xxxxx
  - subnet-xxxxx
  template:
    spec:
      displayName: tke-np-bqclpywh-worker
      providerSpec:
        type: Native
        value:
          instanceChargeType: PostpaidByHour
          keyIDs:
          - skey-xxxxx
          lifecycle: {}
          management:
            kubeletArgs:
            - feature-gates=EnableInPlacePodResize=true
            - allowed-unsafe-sysctls=net.core.somaxconn
            - root-dir=/var/lib/test
            - registry-qps=1000
            hosts:
            - Hostnames:
              - static.fake.com
              IP: 192.168.2.42
            nameservers:
            - 183.60.83.19
            - 183.60.82.98
            kernelArgs:
            - kernel.pid_max=65535
            - fs.file-max=400000
            - net.ipv4.tcp_rmem="4096 12582912 16777216"
            - vm.max_map_count="65535"
          metadata:
            creationTimestamp: null
          securityGroupIDs:
          - sg-b3a93lhv
          systemDisk:
            diskSize: 50
            diskType: CloudPremium
      runtimeRootDir: /var/lib/containerd
  type: Native
```

KubeletArgs Parameter Description

1. The kubelet parameters supporting configuration are inconsistent for different accounts and different cluster versions. If the current parameters do not meet your needs, you can submit a ticket to contact us.
2. The following parameters are not recommended for modification, for they will highly probably affect the normal operation of business on the node:
container-runtime
container-runtime-endpoint
hostname-override
kubeconfig
root-dir

KernelArgs Parameters

OS parameters supporting adjustment and their allowable values are listed below.

Sockets and Network Optimization

For proxy nodes expected to process a large amount of concurrent sessions, you can use the following TCP and network options for adjustment.

No.	Parameter	Default Value	Allowable Values/Range	Parameter Type	Description
1	"net.core.somaxconn"	32768	4096 - 3240000	int	The maximum leng the listening queue each port in the sys
2	"net.ipv4.tcp_max_syn_backlog"	8096	1000 - 3240000	int	The maximum leng the TCP SYN queu
3	"net.core.rps_sock_flow_entries"	8192	1024 - 536870912	int	The maximum size hash table for RPS
4	"net.core.rmem_max"	16777216	212992 - 134217728	int	The maximum size bytes, of the socke receiving buffer.
5	"net.core.wmem_max"	16777216	212992 - 134217728	int	The maximum size bytes, of the socke sending buffer.
6	"net.ipv4.tcp_rmem"	"4096 12582912 16777216"	1024 - 2147483647	string	The minimum/default/m size of TCP socket receiving buffer.
7	"net.ipv4.tcp_wmem"	"4096	1024 -	string	The

		12582912 16777216"	2147483647		minimum/default/m size of TCP socket sending buffer.
8	"net.ipv4.neigh.default.gc_thresh1"	2048	128 - 80000	int	The minimum numl entries that can be retained. If the nun entries is less than value, the entries w be recycled.
9	"net.ipv4.neigh.default.gc_thresh2"	4096	512 - 90000	int	When the number c entries exceeds thi the GC will clear th entries longer than seconds.
10	"net.ipv4.neigh.default.gc_thresh3"	8192	1024 - 100000	int	The maximum allov number of non-perr entries.
11	"net.ipv4.tcp_max_orphans"	32768	4096 - 2147483647	int	The maximum num TCP sockets not at to any user file han held by the system Increase this paran value properly to av "Out of socket men error when the loac
12	"net.ipv4.tcp_max_tw_buckets"	32768	4096 - 2147483647	int	The maximum num timewait sockets he the system simultar Increase this paran value properly to av "TCP: time wait buc table overflow" errc

File Descriptor Limits

Large amounts of traffic in service usually come from a large number of local files. You can slightly adjust the following kernel settings and built-in limits so that only a part of the system memory is used to handle larger traffic.

No.	Parameter	Default Value	Allowable Values/Range	Parameter Type	Description
1	"fs.file-max"	3237991	8192 -	int	The limit on the total

			12000500		number of file descriptors (FDs), including sockets, in the entire system.
2	"fs.inotify.max_user_instances"	8192	1024 - 2147483647	int	The limit on the total number of inotify instances.
3	"fs.inotify.max_user_watches"	524288	781250 - 2097152	int	The limit on the total number of inotify watches. Increase this parameter value to avoid the "Too many open files" error.

Virtual Memory

The following setting can be used to adjust the operations of the Linux kernel virtual memory (VM) subsystem and the writeout of dirty data on disks.

No.	Parameter	Default Value	Allowable Values/Range	Parameter Type	Description
1	"vm.max_map_count"	262144	65530 - 262144	int	The maximum number of memory map areas a process can have.

Worker Thread Limits

No.	Parameter	Default Value	Allowable Values/Range	Parameter Type	Description
1	"kernel.threads-max"	4194304	4096 - 4194304	int	The system-wide limit on the number of threads (tasks) that can be created on the system.
2	"kernel.pid_max"	4194304	4096 - 4194304	int	The system-wide limit on the total number of processes and threads. PIDs greater than this value are not allocated.

Modifying Native Nodes

Last updated：2024-06-14 16:28:43

This document describes how to modify common parameters for native nodes.

Feature Items	Modify Location and Description
Cloud Tag	**Location Description**: Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Click Edit at the top right corner of the Node Pool Basic Information module, and modify the Tencent Cloud tag in Edit Parameter Settings. **Effective Range**: existing nodes and newly added nodes.
Model	**Location Description**: Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Select the Node Startup Configuration Information module and edit the model information. **Effective Range**: newly added nodes. **Note:** The host model does not support deletion. You can add other alternative models with the same specifications. 1. You can select up to 10 models (including the host model) in the same node pool. 2. The list of available instance types is filtered based on the availability zone of the node pool subnet and the existing network resource margin. 3. If the host model of the node pool is GPU instance, adding non-GPU instance types as alternative models is not supported.
System disk	The console does not currently support modifying system disk types and capacities. You can edit the relevant fields in the machinesets objects corresponding to the node pool using kubectl. (For details, see Creation Parameter Description). Changes will only apply to newly added nodes. **Note:** Existing nodes do not support modifications to system disk types and capacities.
Data disk	**Location Description:** Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select

	Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Select the Node Startup Configuration Information module and edit the data disk information. **Effective Range**: newly added nodes.
Public network bandwidth	The console does not currently support modifying Public Network Bandwidth Binding. You can edit the relevant fields in the machinesets objects corresponding to the node pool using kubectl. (For details, see Enabling Public Network Access). Changes will only apply to newly added nodes. **Note:** Existing nodes do not support modifications to the public network activation status.
Security group	**Location Description:** Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Select the Node Startup Configuration Information module and edit the security group information. **Effective Range**: If you check "Inventory Update", then the changes (including deletion, updating, and appending) on this security group will apply to all nodes (including existing and newly added nodes) in the node pool. You are advised to perform the operation with caution. If you do not check "Inventory Update", the changes will only apply to newly added nodes.
Host name	**Location Description:** Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Select the Node Startup Configuration Information module and edit the host name information. **Effective Range**: newly added nodes. **Note:** 1. By default, the host naming pattern is consistent with the cluster's node hostname naming pattern. 2. When the cluster's node hostname naming pattern is set to automatic naming, this parameter cannot be modified. By default, the intranet IP address is used as the host name and the node's hostname. 3. When the cluster's node hostname naming pattern is set to manual naming, this parameter can be modified. You can use the intranet IP address or a custom name as the host name and the node's hostname.
Supports subnet	**Location Description:** Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the

	target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Select the Node Startup Configuration Information module and edit the subnet information. **Effective Range**: newly added nodes.
Elastic scaling	**Location Description**: Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Click Edit at the top right corner of the Operation Information module and modify AS. **Effective Range**: newly added nodes. **Note**: For details on the principles of use and operational limitations, see Node Scaling.
qGPU	**Location Description**: Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Click Edit at the top right corner of the Operation Information module and modify qGPU. **Effective Range**: newly added nodes. **Note**: Whether the qGPU can be enabled depends on the model and the driver. For details, see Using qGPU.
Label/Taint/Annotation	**Location Description**: Log in to [Container Service Console] and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a node pool to go to the node pool details page. Select the Parameter Setting module and edit the Label/Taint/Annotation information. **Effective Range**: If you check "Inventory Update", then the changes (including deletion, updating, and appending) on Label/Annotation/Taint will apply to all nodes (including existing and newly added nodes) in the node pool. You are advised to perform the operation with caution. If you do not check "Inventory Update", the changes will only apply to newly added nodes.
Kubelet/Kernel/Nameserver/Host	**Location Description**: Log in to Container Service Console and choose Cluster from the left sidebar. On the cluster list page, click the ID of the target cluster to go to the details page. Choose Node Management, select Worker Node, and click on the Node Pool tab. On the tab page, select a

	node pool to go to the node pool details page. Select the Parameter Setting module and edit Management information. **Effective Range**: If you check the "Inventory Update" field, then the changes (including deletion, updating, and appending) on Management will apply to all nodes (including existing and newly added nodes) in the node pool. You are advised to perform the operation with caution. If you do not check "Inventory Update", the changes will only apply to newly added nodes. **Note:** Some Kubelet parameters may trigger a business pod restart when you apply the parameter settings to existing nodes. You are advised to perform the operation with caution after assessing the risks. For details, see Management Parameter Introduction.

Enabling Public Network Access for a Native Node

Last updated：2023-05-05 11:05:32

Note:

Bill-by-CVM accounts cannot be used to enable public network access for native nodes. For more information, see Account Types. If you use a bill-by-CVM account, you can submit a ticket to upgrade your account.

This document describes how to bind a node to an elastic IP (EIP) and enable public network access for the node in the TKE console or using YAML.

Note

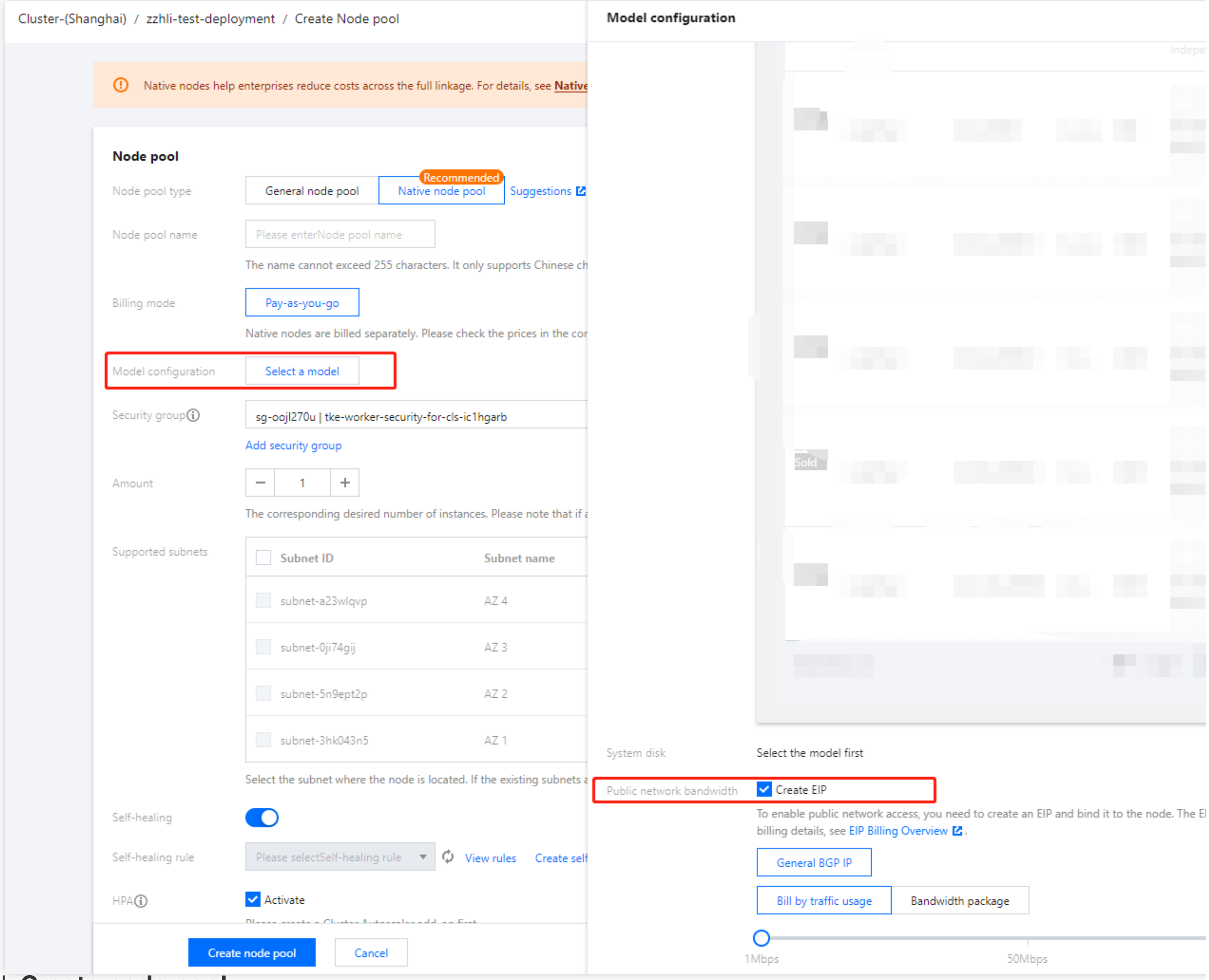
In a node pool with public network access enabled, each time a native node is created, an EIP is automatically created and bound to the node.

The EIP bound to a node has the same lifecycle as the node.

When you use native nodes, no fees are charged for EIPs.

Using the Console to Enable Public Network Access for Native Nodes

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the details page.
3. Choose **Node management** > **Node pool** in the left sidebar. On the **Node pool list** page, click **Create node pool**.
4. On the **Create node pool** page, click **Model configuration**. On the **Model configuration** page, select **Create EIP**, as shown below:



Cluster-(Shanghai) / zzhli-test-deployment / Create Node pool

Native nodes help enterprises reduce costs across the full linkage. For details, see Native

Node pool

Node pool type: General node pool | Native node pool (Recommended) | Suggestions

Node pool name: Please enterNode pool name

The name cannot exceed 255 characters. It only supports Chinese ch

Billing mode: Pay-as-you-go

Native nodes are billed separately. Please check the prices in the cor

Model configuration: Select a model

Security group: sg-oojl270u | tke-worker-security-for-cls-ic1hgarb

Add security group

Amount: − 1 +

The corresponding desired number of instances. Please note that if a

Supported subnets:

Subnet ID	Subnet name
subnet-a23wlqvp	AZ 4
subnet-0ji74gij	AZ 3
subnet-5n9ept2p	AZ 2
subnet-3hk043n5	AZ 1

Select the subnet where the node is located. If the existing subnets a

Self-healing: ●

Self-healing rule: Please selectSelf-healing rule | View rules | Create self

HPA: ☑ Activate

Create node pool | Cancel

Model configuration

System disk: Select the model first

Public network bandwidth: ☑ Create EIP

To enable public network access, you need to create an EIP and bind it to the node. The EI billing details, see EIP Billing Overview.

General BGP IP

Bill by traffic usage | Bandwidth package

1Mbps 50Mbps

5. Click **Create node pool**.

Using YAML to Enable Public Network Access for Native Nodes

Fields

Field Name	Field Value	Description
spec.template.spec.providerSpec.value.internetAccessible	addressType	`EIP` : If this field is left er standard EIP is used. `HighQualityEIP` : A E (dedicated EIP) is used.
	chargeType	Billing mode: `TrafficPostpaidByH` Postpaid by traffic on an ho basis.

		`BandwidthPostpaidB` Postpaid by bandwidth on hourly basis. `BandwidthPackage` : shared bandwidth packag EIP must be in the allowlis bandwidth package.
	maxBandwidthOut	Maximum bandwidth in Mb
	bandwidthPackageID	ID of the shared bandwidth package.

Note

BGP IPs support only **bill-by-IP accounts** in the **Hong Kong (China)** region. You can go to the Virtual Private Cloud (VPC) console to create a shared bandwidth package.

YAML sample

```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  deletePolicy: Random
  displayName: HighQualityEIP-test
  instanceTypes:
  - SA2.MEDIUM2
  replicas: 1
  scaling:
    createPolicy: ZonePriority
    maxReplicas: 4
```

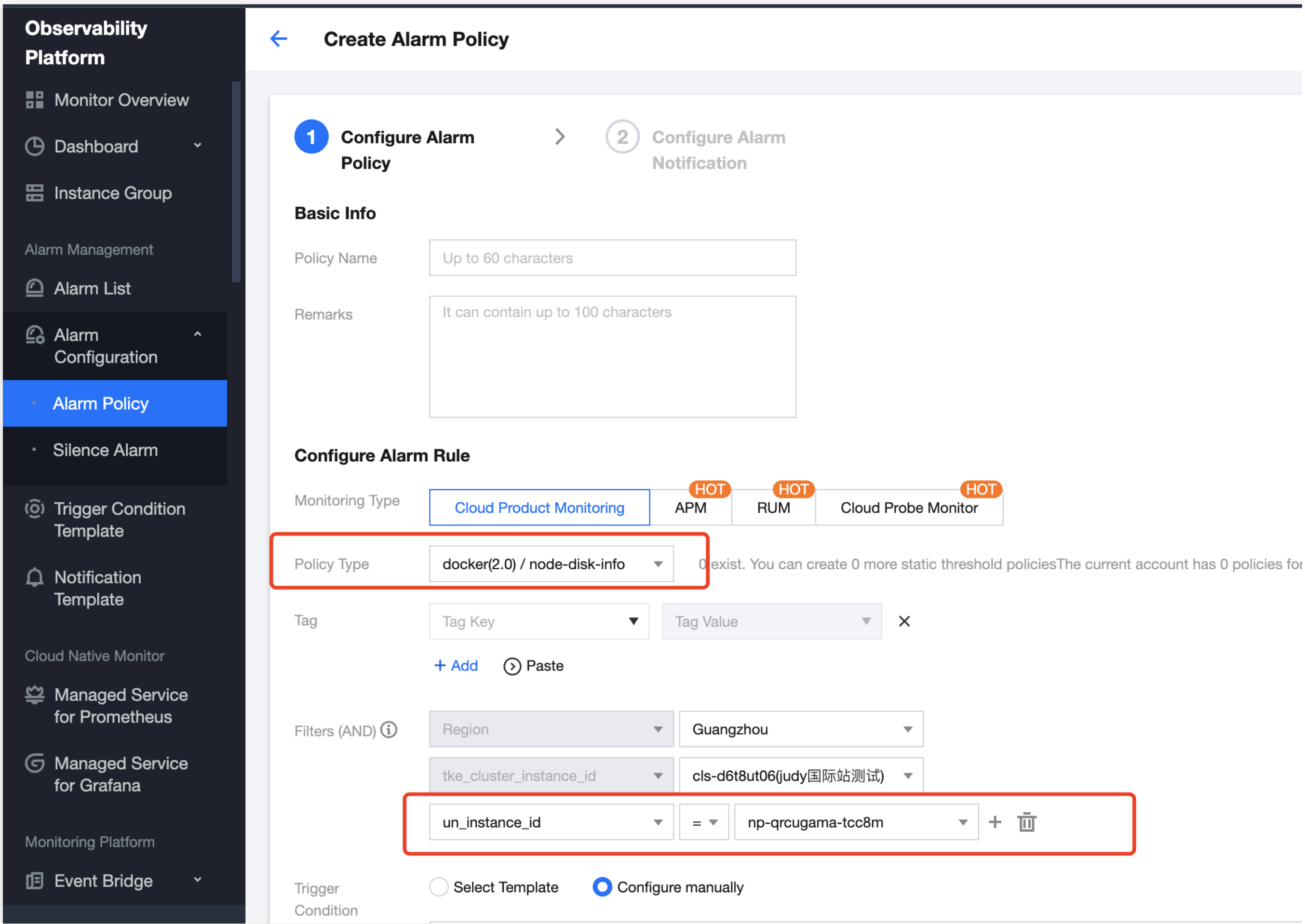
```
subnetIDs:
- subnet-xxxxxxx
template:
  metadata:
    labels:
      node.tke.cloud.tencent.com/machineset: np-ohh7gaek
  spec:
    providerSpec:
      type: Native
      value:
        instanceChargeType: PostpaidByHour
        lifecycle: {}
        management:
          nameservers:
          - 183.60.83.19
          - 183.60.82.98
        metadata:
          creationTimestamp: null
        securityGroupIDs:
        - sg-5lxe2r2p
        systemDisk:
          diskSize: 50
          diskType: CloudPremium
        internetAccessible:
          chargeType: BandwidthPackage
          bandwidthPackageID: bwp-95xr2686
          maxBandwidthOut: 100
          addressType: HighQualityEIP
    runtimeRootDir: /var/lib/containerd
type: Native
```

FAQs for Native Nodes

Last updated：2024-06-14 16:28:43

How Can We Configure System Disk Monitoring Alerts?

1. Log in to TCOP, and choose **Alert Management > Policy Management**.
2. Click **Create New Policy** and configure alerts.
3. On the alert configuration page, select **Cloud Product Monitoring** as the monitoring type, and select either **Container Service (2.0)/Node disk information** or **Container Service (2.0)/Node disk IO** as the policy type.
4. Sequentially select **Region, Cluster** and **Node ID**, and set the target native node as the alert object, as shown in the following figure:



Basic Info

Policy Name: Up to 60 characters

Remarks: It can contain up to 100 characters

Configure Alarm Rule

Monitoring Type: Cloud Product Monitoring | HOT APM | HOT RUM | HOT Cloud Probe Monitor

Policy Type: docker(2.0) / node-disk-info — 0 exist. You can create 0 more static threshold policiesThe current account has 0 policies for

Tag: Tag Key | Tag Value ✕

+ Add Paste

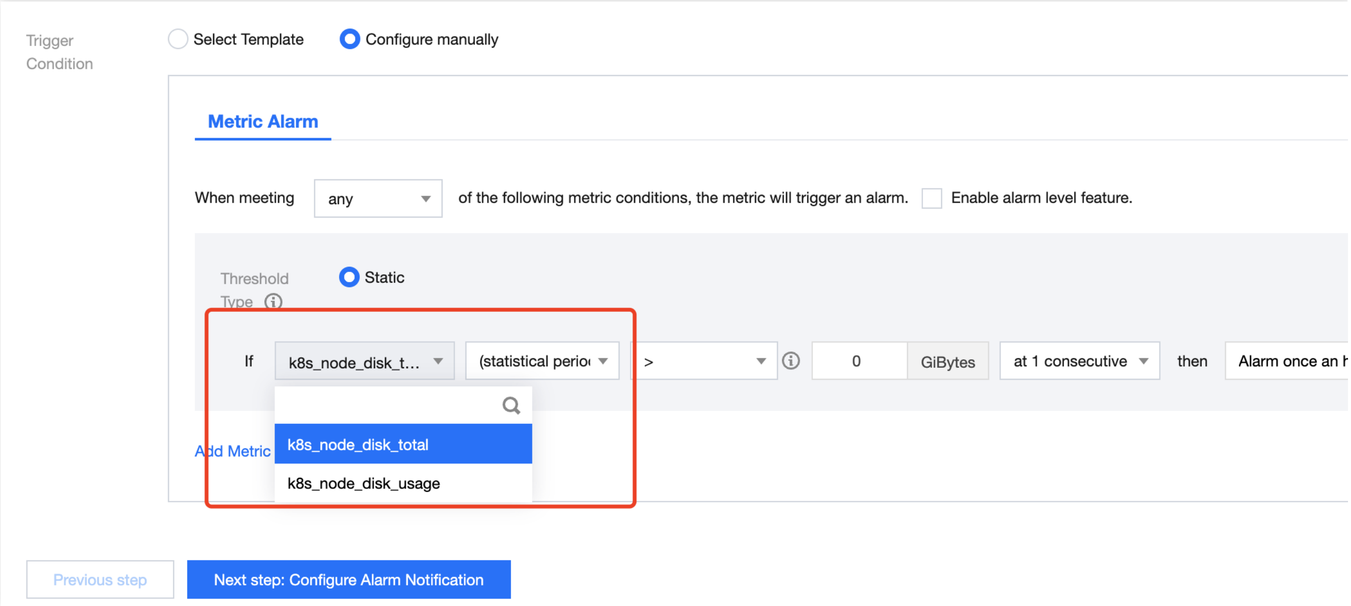
Filters (AND): Region = Guangzhou

tke_cluster_instance_id = cls-d6t8ut06(judy国际站测试)

un_instance_id = np-qrcugama-tcc8m

Trigger Condition: ○ Select Template ● Configure manually

5. Select **Total Disk Capacity, Disk Utilization, and Disk Write Bandwidth** as alert metrics, as shown in the following figure:


Trigger Condition ○ Select Template ● Configure manually

Metric Alarm

When meeting any of the following metric conditions, the metric will trigger an alarm. □ Enable alarm level feature.

Threshold Type ● Static

If k8s_node_disk_t... (statistical perio > 0 GiBytes at 1 consecutive then Alarm once an h

k8s_node_disk_total

k8s_node_disk_usage

Add Metric

Previous step | Next step: Configure Alarm Notification

6. Click **Next: Configure Alarm Notification**. For alert notification configuration details, see Notification Template.
7. Click **Finish**.

How Can We View the Resource Quantity of a Container Within Itself?

Background

When viewing container resources in standard containers (for example, with the top command), the resources of the entire machine are visible. This can cause confusion for some monitoring, troubleshooting, and even some businesses (such as deciding the queue length based on certain parameters under proc). The native node image has the capability to control container resource visibility. By mounting cgroupfs to the host directory and then using bind mount to mount the directory's files to the corresponding /proc location in the container during creation, you can view the container's own resource quantity within itself. That is, commands like free, top, and loadavg within the container will display the container's own values.

Usage

You need to set the following annotation on the pod:

Field annotation	Meaning
cloud.tencent.com/cgroupfs="*"	This means all containers within the pod apply the cgroupfs capability.
cloud.tencent.com/cgroupfs="container1,container2"	This means that only container 1 and container 2 in the pod apply the cgroupfs capability.

Note:

This feature requires the containerd version to be later than 1.4.3-tke.3 or 1.6.9-tke.3.

Supernode management

Super Node Overview

Last updated：2023-09-22 18:16:28

Overview

A super node is not an actual node, but a kind of scheduling capability based on the native K8s. It supports scheduling the Pods in a standard Kubernetes cluster to a super node that does not occupy the cluster server resource. In the TKE cluster that has enabled the super node feature, the Pods that meet the scheduling conditions will be scheduled to the super node maintained by Tencent Kubernetes Engine for Serverless.

Pods deployed on the super nodes have the same security isolation as CVM, and have the same network isolation and network connectivity as Pods deployed on existing nodes in the cluster, as shown in the figure below:

Concepts

Elastic container

If a cluster has deployed a super node, the Pods scheduled to the super node are the elastic containers, which do not occupy the node resources of cluster server, nor is it restricted by the upper limit of server node resources.

Node pool

To help you efficiently manage nodes in a Kubernetes cluster, TKE introduced the concept of node pool. Basic node pool features allow you to conveniently and quickly create, manage, and terminate nodes and dynamically scale nodes in or out.

Benefits

Making elasticity faster and more efficient

With compared to node pool and scaling group, scaling out and in process for super nodes simplifies server purchase, initialization and returning. This improves the speed of elasticity greatly, reduces possible failures in scaling out to the largest extent, and makes elasticity more efficient.

- The scaling out process of super node is in seconds.

- The scaling in process of super node is short. It can scale in instantaneously with non-stop service.

Cost saving

Super nodes have the advantages of second-level elasticity and serverless, on-demand product form, which makes them have a great advantage in terms of costs.

- Use on demand to reduce the cluster resource buffer. Because the specifications of the Pods scheduled to real nodes cannot completely match the node specifications, there will always be some fragmented resources that cannot be used but are still billed. Super nodes are used on demand to avoid the generation of fragmented resources, therefore, it can improve the resource utilization of the overall cluster, reduce buffer and save costs.

- Reduce the billing duration of elastic resources and save costs. Because the super node is scaling out in seconds and scaling in instantaneously, it will greatly reduce the costs in scaling.

Billing

Fees are not charged for super nodes, but charged based on the Pod resources scheduled to the super nodes.

Elastic container on the super node is a pay-as-you-go service. The fees are calculated based on the configured amount of resources and the actual period of using them. Fees will be calculated based on the specifications of the CPU, GPU, and memory for a workload and the running time of the workload. For more information, see Product Pricing.

Notes on scheduling

Generally, a cluster with super node enabled will automatically scale out Pods to a super node when the server node resources are insufficient, and scale in the Pods on the super node first when the server node resources are sufficient. You can also schedule the Pods to a super node manually. For details, please see Notes on Scheduling Pod to Super Node.

Operations

Scaling out in seconds to deal with burst traffic easily

For irregular burst traffic, it is difficult to guarantee timely node scaling. If resource specifications are configured with high traffic as a baseline, only a small portion of resources will be used when the traffic is stable, which is a serious waste of resources. It is recommended to configure super nodes without additional preset resources to deal with burst traffic.

- High elasticity: scale out in seconds, and deal with burst traffic easily. It will automatically terminate the Pods when service traffic drops. Scale in with non-stop service.
- Low costs: avoid resources idling costs and improve utilization of resources.

Reducing the cluster resource buffer for long-term running service peaks

For long-term running applications with tidal resource loads, super nodes can quickly deploy a large number of Pods without occupying cluster server node resources. When the Pods need to scale out at the service peak, they will be automatically scheduled to the nodes first, consuming the reserved node resources, and then be scheduled to the super nodes to supply more temporary resources to the cluster. These resources will be automatically returned as the Pods scale in.

- High elasticity: scale out in seconds, automatically terminate Pods when service traffic drops, and scale in with non-stop service.
- Low costs: you can reduce the reserved cluster buffer, make the usage and reservation of cluster node resources more reasonable, improve the resource utilization, and reduce costs.

Replacing node scaling for short-term running tasks

For tasks that run in a short term and have high resource demands, it is generally necessary to manually scale out a large number of nodes to ensure resources, then schedule the Pods, and return the server after the task is completed. Node resources have buffer, which causes a waste of resource. It is recommended to use super nodes and schedule Pods to super nodes manually without the need of node management.

- No node scaling is required: No need to scale out and in the cluster nodes before and after deployment of these loads, which reduces the time period and maintenance cost of nodes scaling. When the task is completed and the Pod exits, resources will be returned automatically and billing will be stopped. No need for human or program intervention.
- Use on demand to reduce cost: creating Pods based on resources required to avoid resource buffer.

Purchasing a Super Node
Super Node Pricing

Last updated：2023-07-07 14:50:26

The following pricing takes effect from 00:00 on July 1, 2023 (UTC +8).

Pay-as-You-Go Pricing

TKE Serverless Cluster is billed based on the resource specification and usage. **Fee** = **Billable item specification** × **Resource price per unit time** × **Running time (in seconds)**. For details on specifications of billable items, see Resource Specifications.

Intel Pod Pricing

Region	Billable items			
	CPU (USD/core/second)	Memory (USD/GiB/second)	Premium cloud disk (USD/GB/second) Free of charge of the first 20 GB	SSD cloud disk (USD/GB/second) Free of charge of the first 20 GB
Shanghai, Beijing, Guangzhou, Nanjing, Chongqing, Chengdu, Qingyuan, Wuhan, Changsha, Zhengzhou, Xi'an, Shenyang, Fuzhou, Hefei, Jinan, Hangzhou, Shijiazhuang, Shanghai Finance, Shenzhen Finance,	0.000004976	0.000002073	0.00000004	0.000000104

Beijing Finance, Shanghai ADC, Japan, Thailand, Silicon Valley, India, Virginia, Sao Paulo				
Hong Kong (China), Taipei (China), Singapore	0.000004976	0.000002248	0.00000002	0.000000010
Seoul, Frankfurt	0.000005224	0.000002612	0.00000002	0.000000010
Jakarta	0.000005804	0.000002902	0.00000002	0.000000010

AMD Pod Pricing

Region	Billable items			
	CPU (USD/core/second)	Memory (USD/GiB/second)	Premium cloud disk (USD/GB/second) Free of charge of the first 20 GB	SSD cloud disk (USD/GB/second) Free of charge of the first 20 GB
Shanghai, Beijing	0.00000269	0.00000133	0.00000004	0.000000104
Guangzhou, Qingyuan, Wuhan, Changsha, Zhengzhou, Xi’an, Shenyang, Fuzhou, Hefei, Jinan, Hangzhou, Shijiazhuang	0.00000253	0.00000133	0.00000002	0.000000010
Shanghai Finance, Shenzhen	0.00000410	0.00000203	0.00000002	0.000000010

Finance, Beijing Finance, Shanghai Auto-Driving Cloud				
Hong Kong (China), Taipei (China), Japan, Singapore, Thailand, India, Virginia, Sao Paulo, Frankfurt	0.00000361	0.00000182	0.00000002	0.000000010
Silicon Valley	0.00000299	0.00000149	0.00000002	0.000000010

Running Time

The running time is calculated in seconds, staring when a Pod fetches the first container image until it stops. The Pod is billed based on the resource usage during this period.

Billing Examples

Example 1

Assume that there is a Deployment in Virginia, whose resource specification is 2C4G with 2 replicas. It takes 5 minutes (300 seconds) from the time when the Deployment is started to the time it ends.

In this case, the fee of the Deployment = 2 × (2 × 0.000004976 + 4 × 0.000002073) × 300 = 0.019495 USD.

Example 2

Assume that a CronJob in Silicon Valley needs to start 10 4C8G AMD Pods each time and terminate the Pods 10 minutes (600 seconds) later. If the CronJob executes the job twice a day, the task is billed as follows:

Daily task fee = 2 × 10 × (4 × 0.00000299 + 8 × 0.00000149) × 600 = 0.28656 USD.

Creating Super Node

Last updated：2022-11-02 16:02:34

This document describes how to deploy super nodes in a cluster in the TKE console.

Prerequisites

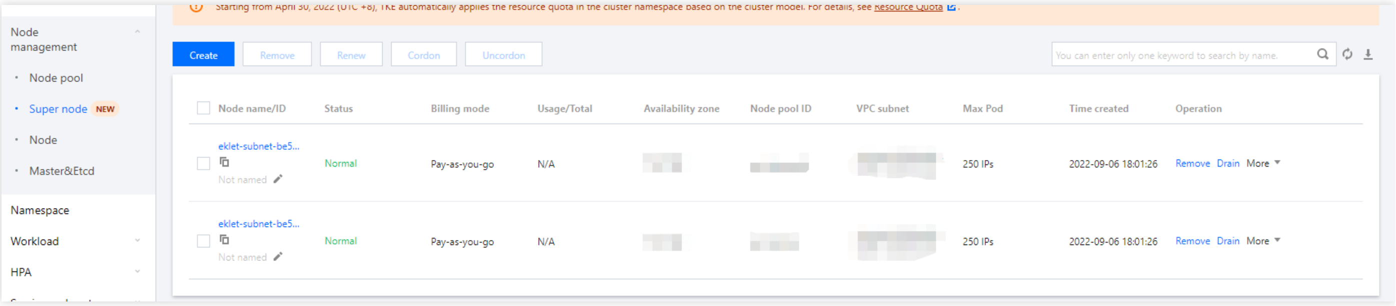
- You have created a cluster.
- Kubernetes is on v1.16 or later.

Directions

1. Log in to the TKE console and click **Cluster** on the left sidebar.

2. On the cluster list page, click the target cluster ID to enter the cluster details page.

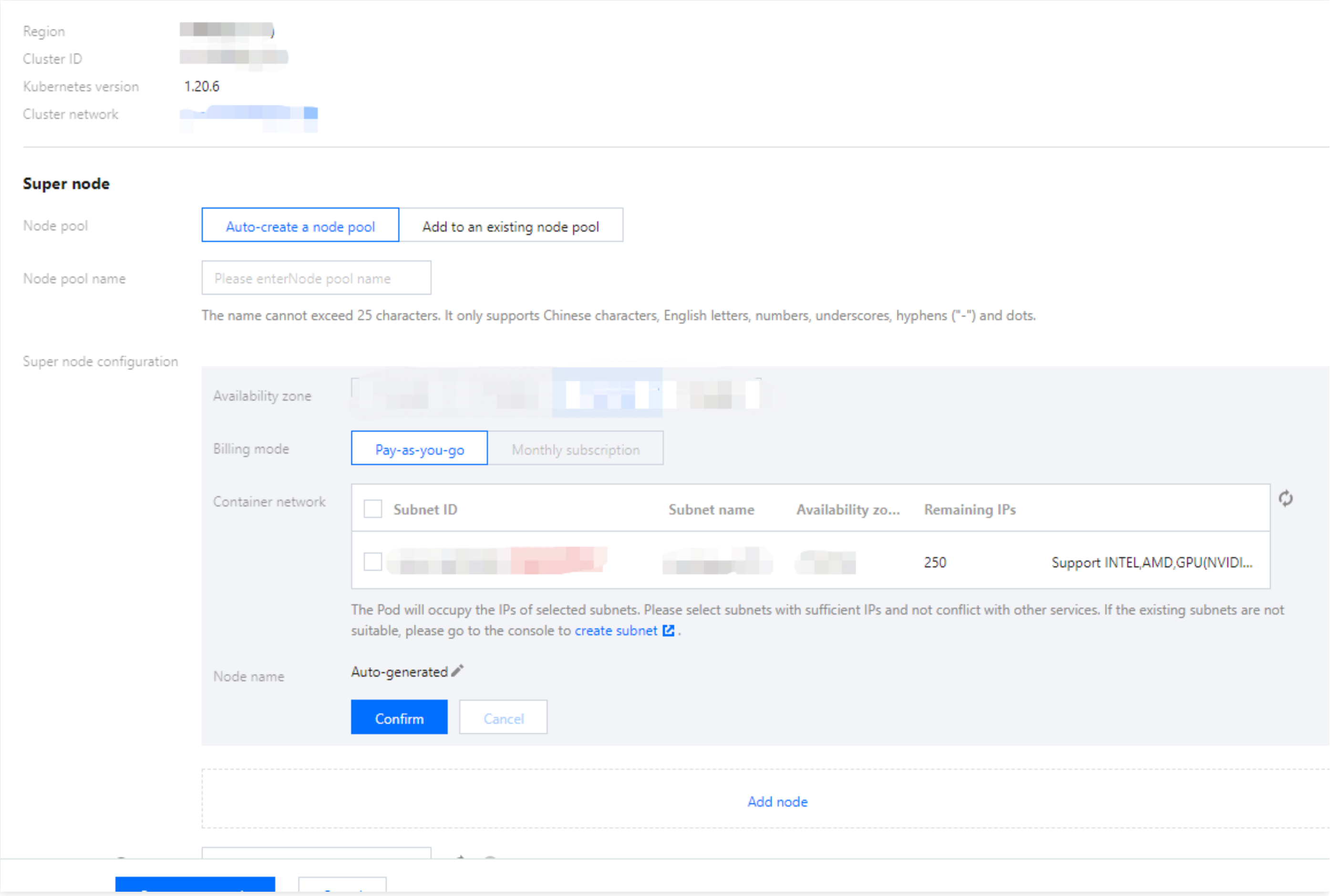
3. Select **Node Management** > **Super Node** on the left sidebar.



Starting from April 30, 2022 (UTC +8), TKE automatically applies the resource quota in the cluster namespace based on the cluster model. For details, see Resource Quota.

Node name/ID	Status	Billing mode	Usage/Total	Availability zone	Node pool ID	VPC subnet	Max Pod	Time created	Operation
eklet-subnet-be5... Not named	Normal	Pay-as-you-go	N/A	[redacted]	[redacted]	[redacted]	250 IPs	2022-09-06 18:01:26	Remove Drain More
eklet-subnet-be5... Not named	Normal	Pay-as-you-go	N/A	[redacted]	[redacted]	[redacted]	250 IPs	2022-09-06 18:01:26	Remove Drain More

4. Click **Create** and specify the parameters as prompted.



Region	[redacted]
Cluster ID	[redacted]
Kubernetes version	1.20.6
Cluster network	[redacted]

Super node

Node pool: Auto-create a node pool | Add to an existing node pool

Node pool name: Please enterNode pool name

The name cannot exceed 25 characters. It only supports Chinese characters, English letters, numbers, underscores, hyphens ("-") and dots.

Super node configuration

Availability zone: [redacted]

Billing mode: Pay-as-you-go | Monthly subscription

Container network:

□ Subnet ID	Subnet name	Availability zo...	Remaining IPs	
□ [redacted]	[redacted]	[redacted]	250	Support INTEL,AMD,GPU(NVIDI...

The Pod will occupy the IPs of selected subnets. Please select subnets with sufficient IPs and not conflict with other services. If the existing subnets are not suitable, please go to the console to create subnet .

Node name: Auto-generated

Confirm | Cancel

Add node

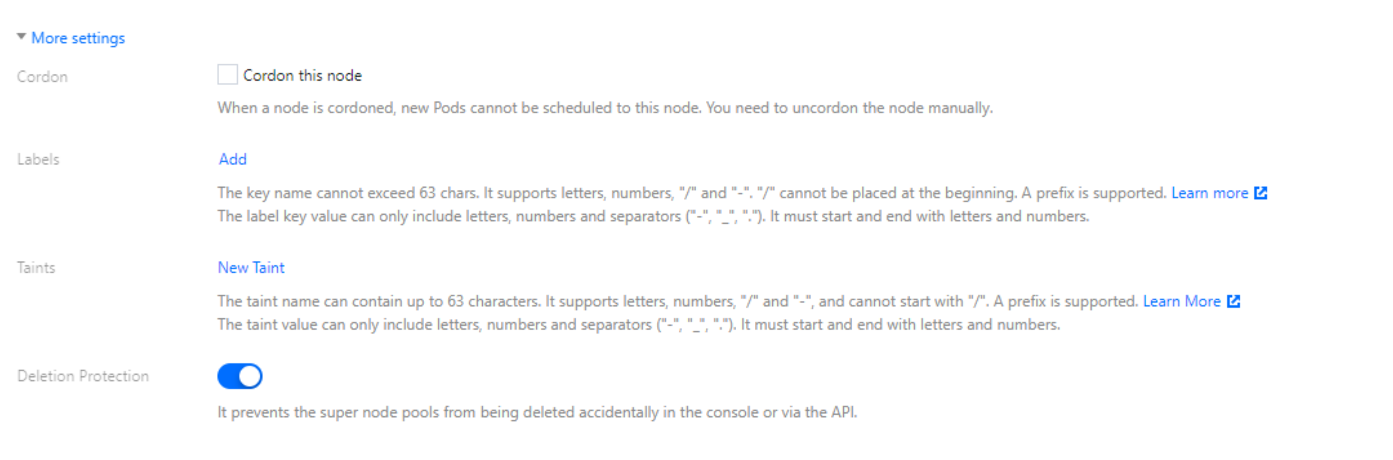
- **Node Pool**: **Auto-create a node pool** and **Add to an existing node pool** are available. For the former, a super node pool will be created when a super node is created; for the latter, a new super node can be added to an existing node pool for unified management. Super nodes in different billing modes and AZs and with different specifications can be added to a node pool.
- **Node Pool Name**: If a node pool is automatically created, its name can be managed.
- **Super node configuration**:
  - **Availability zone**: Select the availability zone where the super node is located.
  - **Billing mode**: The pay-as-you-go billing mode is supported in all regions.
  - **Container network**: Assign IPs within the container IP range to containers in the cluster. Pods on super nodes will occupy VPC subnet IPs. The number of Pods that can be scheduled to super nodes is limited by the number of remaining IPs. Select subnets with sufficient available IPs and not in conflict with those of other services. Pods on super nodes will run in the specified VPC. Each Pod is bound to an ENI in the specified VPC. You can check the ENI associated with the Pod in the ENI list.
    Multiple subnets can be selected in the pay-as-you-go billing mode, and a pay-as-you-go super node is created for a subnet. The super nodes are billed based on the Pod specifications and running duration after the workloads are created and the Pods are scheduled to the super nodes.

> Note
> - We recommend that you configure multiple availability zones for the container network so that your workloads can be automatically distributed to multiple availability zones, which improves usability.
> - Ensure that the subnet assigned to the container network has sufficient available IPs, so as to prevent pod creation failure caused by insufficient IPs when creating a large-scale workload.

- **Security group configuration**: For more information, see TKE Security Group Settings.

> Note
> - The security group configured for a super node is directly associated with its Pods. Configure the network rules as required by the Pods. For example, you need to open port 80 if the Pods provide services via port 80.
> - The security group is the default group to which Pods scheduled to the super node are bound. You can specify another security group to overwrite the default one during scheduling.

5. (Optional) Click **More Settings** to view or configure more information.



▾ More settings

Cordon
□ Cordon this node
When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually.

Labels
Add
The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Taints
New Taint
The taint name can contain up to 63 characters. It supports letters, numbers, "/" and "-", and cannot start with "/". A prefix is supported. Learn More
The taint value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Deletion Protection
●
It prevents the super node pools from being deleted accidentally in the console or via the API.

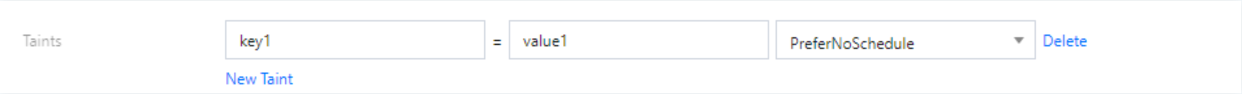
- **Cordon initial nodes**: If **Cordon this node** is selected, new Pods cannot be scheduled to this node. You can uncordon the node manually, or execute the uncordon command in custom data as needed.
- **Labels**: Click **New Label** and customize the label settings. The specified label here will be automatically added to super nodes created in the node pool to help filter and manage super nodes by label.
- **Taints**: It is a node attribute usually used with tolerations. You can specify this parameter for all the super nodes in the node pool, so as to prevent Pods that don't meet the requirements from being scheduled to these nodes and drain such Pods from the nodes.

Note

The value of **Taints** usually consists of `key` , `value` , and `effect` . Valid values of `effect` :

- **PreferNoSchedule**: Optional. Try not to schedule a Pod to a node with a taint that cannot be tolerated by the Pod.
- **NoSchedule**: When a node contains a taint, a Pod without the corresponding toleration must not be scheduled.
- **NoExecute**: When a node contains a taint, a Pod without the corresponding toleration won't be scheduled to the node and any such Pods on the node will be drained.

Assume that Taints is set to `key1=value1:PreferNoSchedule` . The following figure shows the configurations in the TKE console:



Taints key1 = value1 PreferNoSchedule

6. Click **Create Super Node**. If you select **Auto-create a node pool**, the super node pool of the super node will be created synchronously.

Pod Schedulable to Super Node

Last updated：2022-11-02 16:02:34

Billing Mode

Pods scheduled to super nodes are billed on a pay-as-you-go or spot basis.

Kubernetes Versions

- Pay-as-you-go super nodes are supported by clusters on v1.16 or later.

Specifications of Pods Schedulable to Super Nodes

Make sure that you understand the specifications and configurations of Pods supported by super nodes, as they are the basis for billing available resources and services during container running.

Pay-as-you-go mode

- 0.25C–16C Pods are supported (for nonstandard Pods, their specifications are automatically upgraded).
- Pods with a CPU to memory ratio of more than 1:8 are supported.

List of specifications supported by nodes:

> Note：
>
> For nonstandard Pods, their specifications are automatically upgraded.

CPU (Cores)	Memory Range (GiB)	Granularity of Memory Range (GiB)
0.25	0.5, 1, 2	-
0.5	1, 2, 3, 4	-
1	1 - 8	1
2	4 - 16	1

CPU (Cores)	Memory Range (GiB)	Granularity of Memory Range (GiB)
4	8 - 32	1
8	16 - 32	1
12	24 - 48	1
16	32 - 64	1

Super Node Configuration

Pod temporary storage

When each Pod scheduled to a super node is created, a temporary image storage of 20 GiB will be allocated.

> Note：
>
> - Temporary image storage will be deleted when the Pod lifecycle ends. Therefore, do not store important data in it.
> - The actual available storage will be less than 20 GiB due to the stored images.
> - Annotations can be used to scale out system disk resources.
> - We recommend you mount important data and large files to Volume for persistent storage.

Pod network

Pods scheduled to super nodes are in the same VPC as Tencent Cloud services such as CVM and TencentDB. Each Pod occupies a VPC subnet IP.
A Pod can connect to other Pods or Tencent Cloud services in the same VPC without any performance losses.

Pod isolation

Pods scheduled to super nodes have the same security isolation as CVM instances. Pods are scheduled and created on the underlying physical server of Tencent Cloud, and the resource isolation between Pods is guaranteed by virtualization technology during the creation.

Other special Pod configurations

You can define `template annotation` in a YAML file to implement capabilities such as binding security groups, allocating resources, and allocating EIPs for Pods scheduled to super nodes. For configuration details, see the following table:

Note：

- If no security group is specified, a Pod will be bound to the specified security group of the node pool by default. Make sure that the network policy of the security group doesn't affect the normal operation of the Pod. For example, you need to open port 80 if the Pod provides services via port 80.
- To allocate CPU resources, you must specify the `cpu` and `mem` annotations in line with the CPU specifications in Resource Specifications.
- To allocate GPU resources through the method specified in the annotation, you must specify the `gpu-type` and `gpu-count` annotations and ensure that their values meet the GPU specifications in Resource Specifications.

Annotation Key	Annotation Value and Description	Required
eks.tke.cloud.tencent.com/security-group-id	Default security group bound to a workload. Security group ID: You can enter multiple IDs and separate them by comma, for example, `sg-id1,sg-id2` . Network policies take effect based on the sequence of security groups.	No. If you don't specify it, the security group specified by the node pool is bound by default. If you specify it, make sure that the security group ID already exists in the same region.
eks.tke.cloud.tencent.com/cpu	Number of CPU cores required by a Pod. For more information, see Resource Specifications. The unit is cores by default and doesn't need to be specified.	No. If you specify it, make sure that the specification is supported, and you need to enter the `cpu` and `mem` parameters.
eks.tke.cloud.tencent.com/mem	Amount of memory required by a Pod. For more information, see Resource Specifications. You need to specify the unit, for example, 512 MiB, 0.5 GiB, or 1 GiB.	No. If you specify it, make sure that the specification is supported, and you need to enter the `cpu` and `mem` parameters.

Annotation Key	Annotation Value and Description	Required
eks.tke.cloud.tencent.com/cpu-type	CPU model required by a Pod. Currently, Intel and AMD are supported. For more information on configurations supported by S4, S3, and other models, see Resource Specifications.	No. If you don't specify it, the system will match the most suitable specification according to Specifying resource specifications. If the matched specification is supported by both Intel and AMD, the Intel specification is preferred.
eks.tke.cloud.tencent.com/gpu-type	GPU model required by a Pod. Currently, V1001/4*T41/2*T4T4 are supported. You can specify the model by priority, for example, "T4,V100" indicates T4 Pods will be created first. If the T4 resources in the selected region are insufficient, V100 Pods will be created. For more information on configurations, see Resource Specifications.	It is required if you need a GPU. When specifying it, make sure that the GPU model is supported; otherwise, an error will be reported.
eks.tke.cloud.tencent.com/gpu-count	Number of GPUs required by a Pod. For more information, see Resource Specifications. The unit is cards by default and doesn't need to be specified.	No. If you specify it, make sure that the specification is supported.
eks.tke.cloud.tencent.com/retain-ip	Static IP of a Pod. Enter the value of `"true"` to enable this feature. If a Pod with the static IP enabled is terminated, its IP will be retained for 24 hours by default. If the Pod is rebuilt within 24 hours, its IP can still be used; otherwise, its IP may be occupied by other Pods. **It is valid only for StatefulSet and raw Pods.**	No
eks.tke.cloud.tencent.com/retain-ip-hours	Modifies the default retention period of a Pod's static IP. Enter a number. The unit is hours, and the default value is 24 hours. An IP can be retained for up to one year. **It is valid only for StatefulSet and raw Pods.**	No

Annotation Key	Annotation Value and Description	Required
eks.tke.cloud.tencent.com/eip-attributes	Indicates that the Pod of the workload needs to be associated with an EIP. When the value is "", the default EIP configuration is used. You can enter the cloud API parameter `json` of the EIP in "" to customize the configuration. For example, if the value of `annotation` is '{"InternetMaxBandwidthOut":2}', the bandwidth is 2 Mbps. Note that this cannot be used for non-bill-by-IP accounts.	No
eks.tke.cloud.tencent.com/eip-claim-delete-policy	Indicates whether to repossess the EIP after the Pod is deleted. `Never` indicates not to repossess. The default value is to repossess. This parameter takes effect only when `eks.tke.cloud.tencent.com/eip-attributes` is specified. Note that this cannot be used for non-bill-by-IP accounts.	No
eks.tke.cloud.tencent.com/eip-id-list	If the workload is a StatefulSet, you can also specify one or multiple existing EIPs, such as "eip-xx1,eip-xx2". Note that the number of StatefulSet Pods must be less than or equal to the number of EIP IDs specified in this annotation; otherwise, Pods that cannot be allocated with EIPs will be in the "Pending" status. Note that this cannot be used for non-bill-by-IP accounts.	No

For samples, see Annotation.

Default Quota

By default, up to 500 Pods can be scheduled to a pay-as-you-go super node in each cluster. If the desired number of Pods exceeds the quota limit, you can submit a ticket to apply for a higher quota. Tencent Cloud will assess your actual needs and increase your quota as appropriate.

Applying for a higher quota

1. Submit a ticket to enter the ticket page.

2. In the **Problem description** field, enter a description such as "I want to apply for a higher quota for the Pods of cluster super node." Specify the target region and quota. Enter your mobile number and other information as instructed.
3. After providing all the necessary information, click **Submit Ticket**.

Pod Limits

Workload limits

The Pods for DaemonSet workloads won't be scheduled to super nodes.

Service limits

For cluster services in GlobalRouter mode, if externaltrafficpolicy = local, the traffic won't be forwarded to Pods scheduled to super nodes.

Volume limits

The emptyDir, PVC, Secret, NFS, ConfigMap, DownloadAPI, and hostPath volumes are supported.
For PVC volumes:

- PV type: Only NFS, CephFS, hostPath, static CBS types are supported (CSI not supported).
- StorageClass type: Only user-defined and `cloud.tencent.com/qcloud-cbs` types are supported (CFS not supported).

GPU limits

You must specify the `gpu-type` field in the annotation; otherwise, scheduling to super nodes is not supported. Different GPU Pod types come with different CPU and memory specifications, which don't need to be specified. If you need to specify them, make sure that they are identical to those supported by the GPU; otherwise, scheduling will fail.

Other limits

- The super node feature is not available for clusters without any server nodes.
- Pods with the static IP enabled cannot be scheduled to super nodes.
- Pods that with hostIP specified use the Pod IP as the hostIP by default.
- Pods scheduled to super nodes are strictly isolated. If hard anti-affinity is enabled, scheduling to super nodes won't take effect, and it happens that multiple Pods of the same workload are scheduled to the same super node.
- Pods in the `tke-eni-ip-webhook` namespace cannot be scheduled to super nodes.

Scheduling Pod to Super Node

Last updated：2023-05-12 17:06:12

This document describes how to schedule a Pod to a super node in a TKE cluster. There are two ways to do that:

- Auto scaling out
- Manually schedule

Auto scaling out

If the cluster is configured with super nodes, the Pod will be scheduled to a super node automatically when available node resources are insufficient at the service peak. No need to purchase a server. When service regains stability, Pod resources in the super node are released automatically. No server returning is required.

If Cluster Scaling and super node are enabled for the cluster at the same time, Pods will be scheduled to the super node first, and the scaling out will not be triggered. If the Pod cannot be scheduled to the super node due to the above scheduling limits, the node scaling out will be triggered normally. When the server node resources are sufficient, the cluster will scale in the Pods on the super node first.

Manually schedule

You can schedule a Pod to a super node manually. By default, a super node automatically adds taints to lower the scheduling priority. If you want to manually schedule a Pod to a (specified) super node, you need to add corresponding tolerations for the Pod. However, not all the Pods can be scheduled to super nodes. For more information, see Notes for Scheduling Pod to Super Node. For the sake of convenience, you can specify `nodeselector` in Pod Spec, as shown below:

```
spec:
nodeSelector:
node.kubernetes.io/instance-type: eklet
```

Or specify `nodename` in Pod Spec, as shown below:

```
spec:
nodeName: $super node name
```

TKE’s control components will judge whether the Pod can be scheduled to a super node. If not, the Pod will not be scheduled to the super node.

Super Node Annotation Description

Last updated：2022-08-02 18:03:01

By defining annotations in the YAML file, you can leverage the rich customization capabilities of super nodes. For more information on how to configure super nodes, see Annotation Description.

Collecting Logs of the Pod on the Supernodes

Last updated：2022-10-12 16:05:09

This document describes how to collect logs from a Pod scheduled to a super node in a TKE cluster.

- Collect logs to CLS
- Collect logs to Kafka

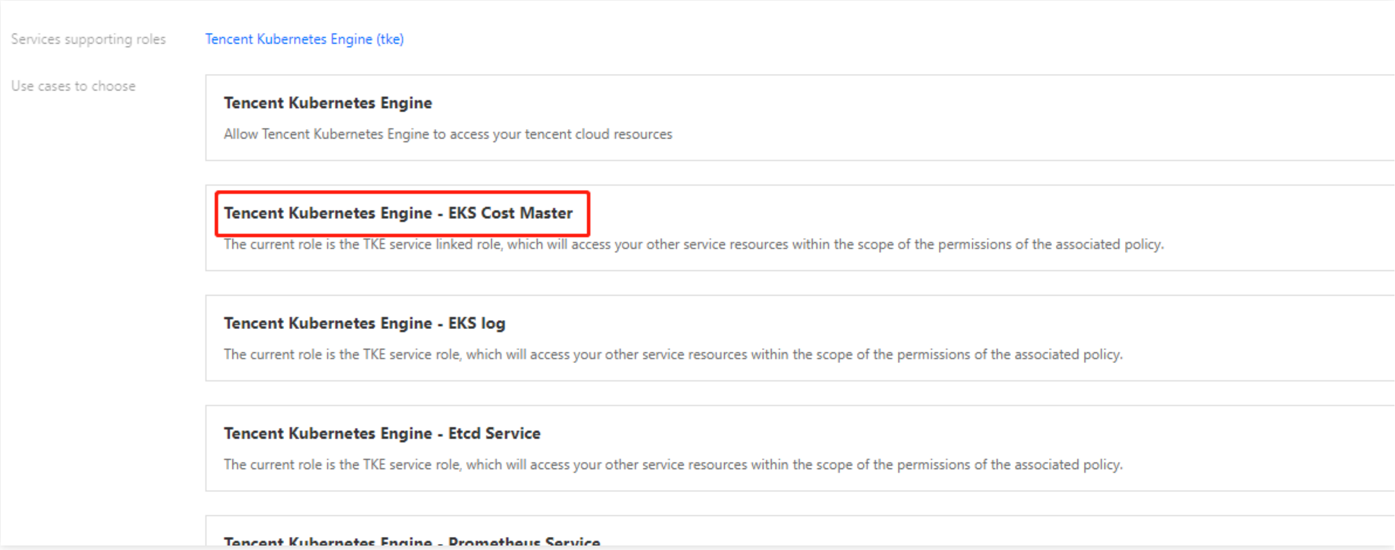
Collecting Logs to CLS

Authorizing a role to the service

Before collecting logs of a Pod on a super node to CLS, you need to authorize a role to the service to ensure that logs can be uploaded to CLS normally:

Follow the steps below:

1. Log in to **CAM console** > **Role**.
2. Click **Create Role** on the "Role" page.
3. In the **Select role entity** dialog box, click **Tencent Cloud Product Service** > **TKE** > **TKE - EKS log collection**, and click **Next**, as shown below:



Services supporting roles   Tencent Kubernetes Engine (tke)

Use cases to choose

Tencent Kubernetes Engine
Allow Tencent Kubernetes Engine to access your tencent cloud resources

Tencent Kubernetes Engine - EKS Cost Master
The current role is the TKE service linked role, which will access your other service resources within the scope of the permissions of the associated policy.

Tencent Kubernetes Engine - EKS log
The current role is the TKE service role, which will access your other service resources within the scope of the permissions of the associated policy.

Tencent Kubernetes Engine - Etcd Service
The current role is the TKE service role, which will access your other service resources within the scope of the permissions of the associated policy.

Tencent Kubernetes Engine - Prometheus Service

4. Confirm role policy, and click **Next**.
5. Review role policy, and click **Done** to complete role configuration.

Configuring log collection

You need to enable TKE log collection feature and configure corresponding log collection rule when you finished service role authorization. For example, you need to specify workload collection and Pod labels collection. For more information, see Using CRD to Configure Log Collection via the Console.

Collecting Logs to Kafka

To collect logs of a Pod on a super node to a self-built Kafka or CKafka cluster, you can configure the log collection rules in the console or CRD to define the collection source and consumer. After CRD configuration, the Pod collector will collect logs according to the configured rules.
The specific configuration of CRD is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig ## Default value
metadata:
name: test ## CRD resource name, unique in the cluster
spec:
kafkaDetail:
brokers: xxxxxx # A required item, broker address, generally it is domain name:port. If there are more than one address, separate them with ",".
topic: xxxxxx # A required item, topicID
messageKey: # An optional item. You can specify the Pod field as the key to upload to the specified partition.
valueFrom:
fieldRef:
fieldPath: metadata.name
timestampKey: # The key of timestamp. Default value is @timestamp.
timestampFormat: # The format of timestamp. Default value is double.
inputDetail:
type: container_stdout ## Log collection type, including container_stdout (container standard output) and container_file (container file).

containerStdout: ## Container standard output
namespace: default ## The Kubernetes namespace of the container to be collected. If this parameter is not specified, it indicates all namespaces.
allContainers: false ## Whether to collect the standard output of all containers in the specified namespace
container: xxx ## Name of the container to be collected. This item can be left empty.
includeLabels: ## Only Pods that contain the specified labels will be collected.
k8s-app: xxx ## Only the logs generated by Pods with the configuration of "k8s-app=xxx" in the Pod labels will be collected. This parameter cannot be specified at the same time as workloads and allContainers=true.
workloads: ## Kubernetes workload to which the container Pod to be collected bel
```

```
ongs
- namespace: prod ## Workload namespace
name: sample-app ## Workload name
kind: deployment ## Workload type. Supported values include deployment, daemonset, statefulset, job, and cronjob.
container: xxx ## Name of the container to be collected. If this item is left empty, it indicates all containers in the workload Pod will be collected.

containerFile: ## File in the container
namespace: default ## The Kubernetes namespace of the container to be collected. A namespace must be specified.
container: xxx ## Name of the container to be collected. You can enter a * for this item.
includeLabels: ## Only Pods that contain the specified labels will be collected.
k8s-app: xxx ## Only the logs generated by Pods with the configuration of "k8s-app=xxx" in the Pod labels are collected. This parameter cannot be specified at the same time as workload.
workload: ## Kubernetes workload to which the container Pod to be collected belongs
name: sample-app ## Workload name
kind: deployment ## Workload type. Supported values include deployment, daemonset, statefulset, job, and cronjob.
logPath: /opt/logs ## Log folder. Wildcards are not supported.
filePattern: app_*.log ## Log file name. It supports the wildcards "*" and "?". "*" matches multiple random characters, and "?" matches a single random character.
```

FAQs

Last updated：2022-12-09 18:03:11

- How do I prohibit a Pod from being scheduled to a pay-as-you-go supernodes?
- How do I prohibit ordinary TKE clusters from automatically scheduling a Pod to a pay-as-you-go supernodes in case of resource inadequacy?
- How do I manually schedule a Pod to a pay-as-you-go supernodes?
- How do I forcibly schedule a Pod to a pay-as-you-go supernodes, no matter whether the pay-as-you-go supernodes supports the Pod?
- How do I customize DNS configuration for a pay-as-you-go supernodes?

Daemonset Running Supported on Supernodes

Last updated：2024-02-04 09:15:47

Feature Overview

Given that super nodes lack the concept of actual nodes, resource objects running on regular nodes, such as DaemonSets, cannot operate as expected. Consequently, certain system-level application capabilities supported by DaemonSet, like log collection and resource monitoring services, cannot be uniformly supported on super nodes. The common industry solution is to implement DaemonSet related capabilities through sidecar injection. However, this leads to an inconsistent user experience compared to regular nodes and is functionally compromised. For example, updates to the sidecar can affect the lifecycle of business Pods. In response, Tencent Cloud has launched a new DaemonSet Pod injection scheme for running DaemonSet on super nodes. As the only scheme in the industry that supports the operation of DaemonSet Pods under a Nodeless architecture, this solution has the following advantages:
Full Compatibility: Fully compatible with the native use of DaemonSet Pods.
Zero Intrusion: DaemonSet Pods and business Pods have individual lifecycles. Any changes to the DaemonSet will not affect business Pods.
Observability: The console supports the monitoring of Daemonset Pod, and the querying of Daemonset Pod logs, events, etc.

Application scenario

TKE Managed Cluster: In the TKE Managed Cluster, if you have added a super node and the Pod running on the super node expects to support the same Daemonset capability as conventional nodes.
TKE Serverless Cluster: In the TKE Serverless Cluster, if you want to run the Daemonset resources, you can refer to this document.
IDC Cluster: in the IDC Cluster, if you have added a super node and the Pod executing on the super node expects to support the same Daemonset capability with IDC nodes.

Preparation

Check if the control plane master components have been upgraded to the following specific versions or higher:

Kubernetes Version	Version Requirements for Control Plane Master Component

v1.26	v1.26.1-tke.1 or higher version
v1.24	v1.24.4-tke.4 or higher version
v1.22	v1.22.5-tke.8 or higher version
v1.20	v1.20.6-tke.30 or a later version
v1.18	v1.18.4-tke.34 or a later version
v1.16	v1.16.3-tke.33 or higher version

Verify that the super node's version has been upgraded to v2.11.20 or higher.

1. Creating Daemonsets and label the Daemonsets that need to run on the super nodes

The purpose of this step is to declare which Daemonsets should run on the super nodes.

If you desire to run the same DaemonSet on both normal nodes and supernodes in TKE, we recommend creating a duplicate DaemonSet and tagging it accordingly. This will not affect the original DaemonSet running on the normal nodes and facilitates management.

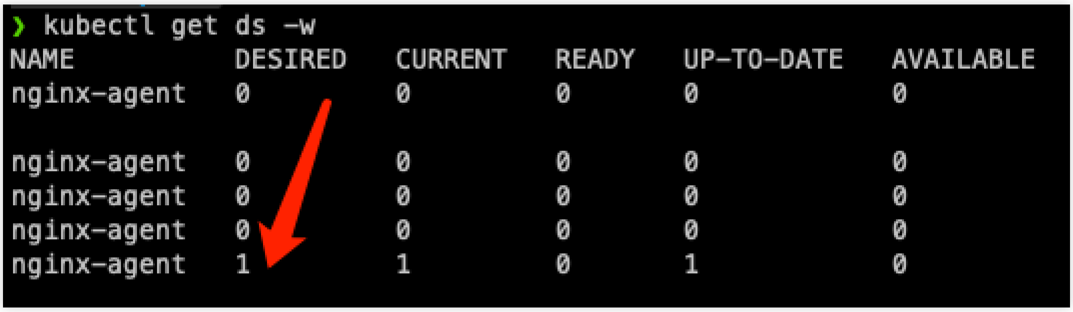
As shown in the file below, tag the DaemonSet to be injected in `eks.tke.cloud.tencent.com/ds-injection: "true"` . For pay-as-you-go supernodes, since they come with default taints (key=eks.tke.cloud.tencent.com/eklet,effect=NoSchedule), they will need to tolerate taints to support creating DaemonSet Pods.

Note:

During the scheduling process, the resources defined in the DaemonSet will not be effective, nor will they generate any additional charges.

```
spec:
  template:
    metadata:
      annotations:
        eks.tke.cloud.tencent.com/ds-injection: "true"
    spec:
      tolerations:
      - key: eks.tke.cloud.tencent.com/eklet
        operator: Exists
        effect: NoSchedule
```

Through the steps above, if there are super nodes in the cluster, you can run the `kubectl get ds` command to view the new replica expansion situation. The cluster will display as many replicas as there are super nodes, as shown in the figure below:



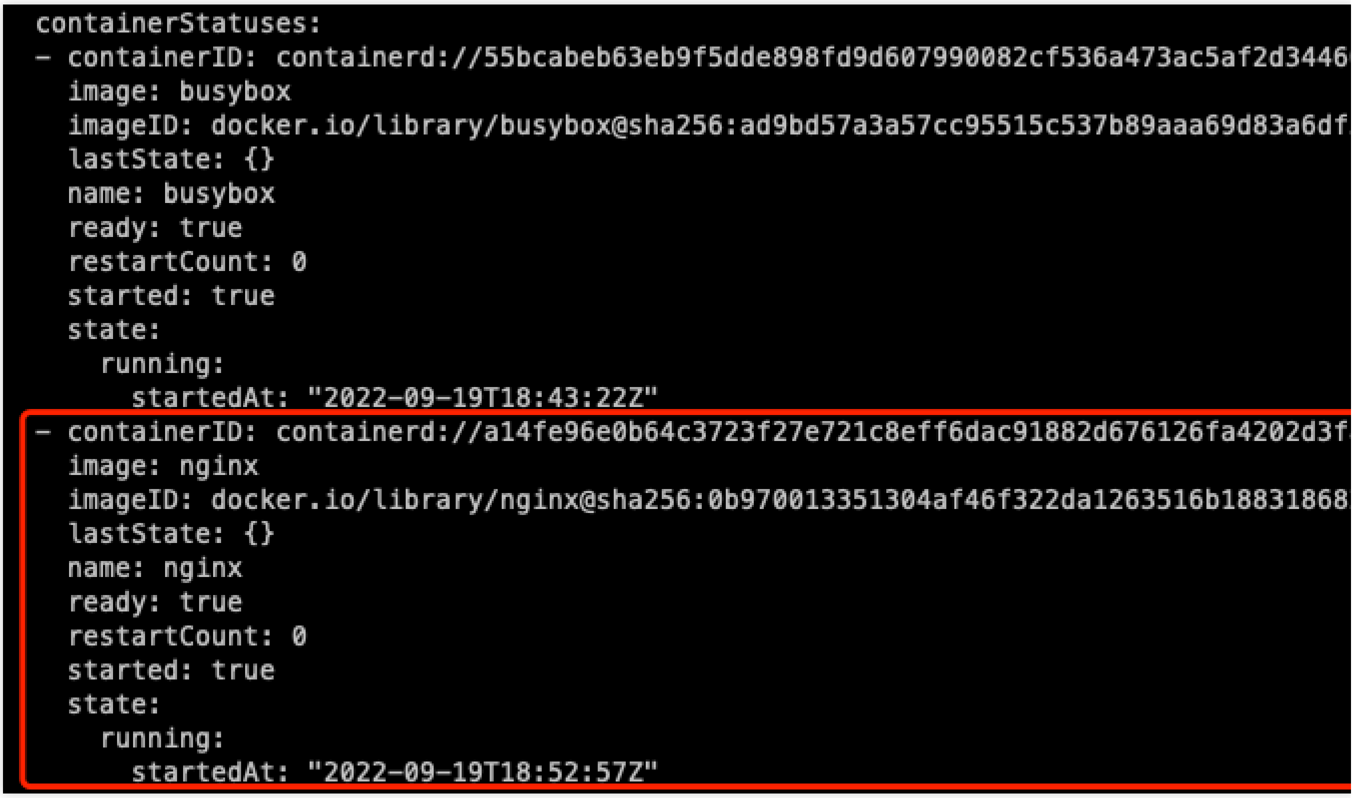
```
❯ kubectl get ds -w
NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
nginx-agent   0         0         0       0            0

nginx-agent   0         0         0       0            0
nginx-agent   0         0         0       0            0
nginx-agent   0         0         0       0            0
nginx-agent   1         1         0       1            0
```

2. Activating DaemonSet Pods

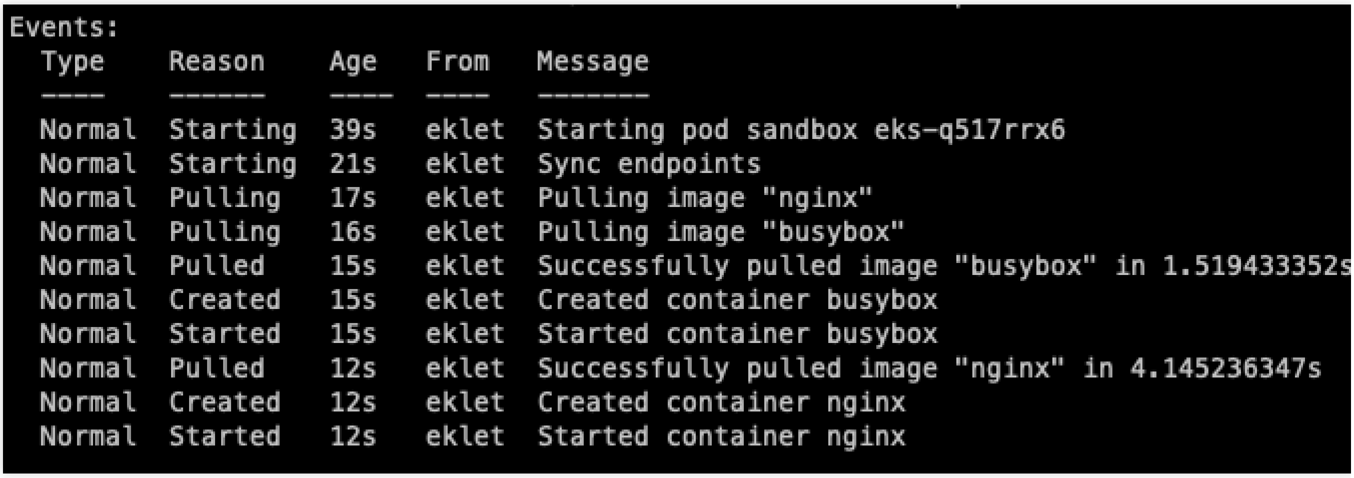
Upon the declaration of DaemonSet running on the supernodes is completed, it implies that every business Pod on the current supernode will be automatically injected with a DaemonSet Pod. Subsequently, any business Pod that is launched will also be automatically injected (excluding Pods under the **kube-system** namespace).

The YAML configuration of the injected Pods themselves will remain unchanged. However, the status of the injected DaemonSet Pod containers will be added to the Container Status for more convenient observation of the status of the DaemonSet Pods, as shown in the figure below:



```
containerStatuses:
- containerID: containerd://55bcabeb63eb9f5dde898fd9d607990082cf536a473ac5af2d3446
  image: busybox
  imageID: docker.io/library/busybox@sha256:ad9bd57a3a57cc95515c537b89aaa69d83a6df
  lastState: {}
  name: busybox
  ready: true
  restartCount: 0
  started: true
  state:
    running:
      startedAt: "2022-09-19T18:43:22Z"
- containerID: containerd://a14fe96e0b64c3723f27e721c8eff6dac91882d676126fa4202d3f
  image: nginx
  imageID: docker.io/library/nginx@sha256:0b970013351304af46f322da1263516b18831868
  lastState: {}
  name: nginx
  ready: true
  restartCount: 0
  started: true
  state:
    running:
      startedAt: "2022-09-19T18:52:57Z"
```

Events of the DaemonSet Pod containers, such as pulling images, launching, abnormal exit, and others, will be reported to the injected Pods.



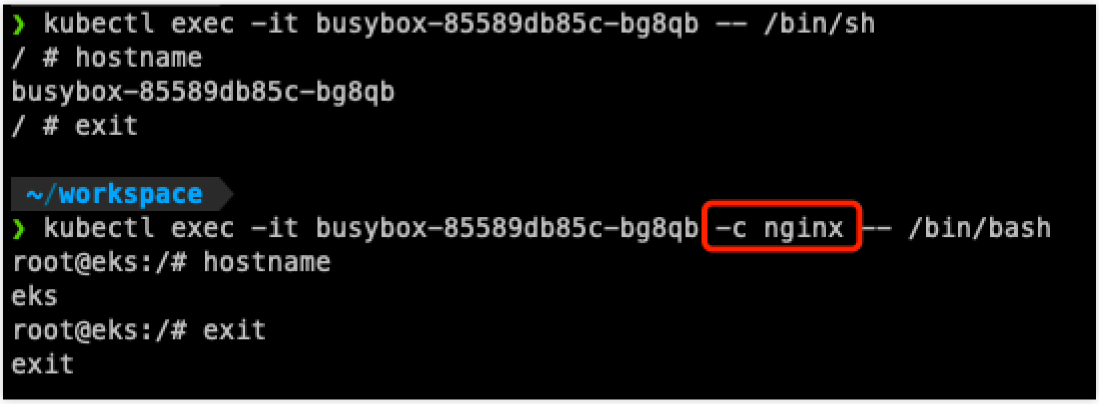
```
Events:
  Type    Reason    Age   From   Message
  ----    ------    ----  ----   -------
  Normal  Starting  39s   eklet  Starting pod sandbox eks-q517rrx6
  Normal  Starting  21s   eklet  Sync endpoints
  Normal  Pulling   17s   eklet  Pulling image "nginx"
  Normal  Pulling   16s   eklet  Pulling image "busybox"
  Normal  Pulled    15s   eklet  Successfully pulled image "busybox" in 1.519433352s
  Normal  Created   15s   eklet  Created container busybox
  Normal  Started   15s   eklet  Started container busybox
  Normal  Pulled    12s   eklet  Successfully pulled image "nginx" in 4.145236347s
  Normal  Created   12s   eklet  Created container nginx
  Normal  Started   12s   eklet  Started container nginx
```

3. Logging in to Daemonset Pods

The `kubectl exec` command allows you to log in to the containers of the original Pods. To log in to the containers of the DaemonSet Pods, direct the `exec` towards the injected Pods, then use the `-c` parameter to specify the DaemonSet Pod containers you want to log in. The command is as follows:

```
kubectl exec -it <daemonset-pod-name> -c <daemonset-container-name> -- /bin/bash
```

There is an example below:



```
❯ kubectl exec -it busybox-85589db85c-bg8qb -- /bin/sh
/ # hostname
busybox-85589db85c-bg8qb
/ # exit

 ~/workspace
❯ kubectl exec -it busybox-85589db85c-bg8qb -c nginx -- /bin/bash
root@eks:/# hostname
eks
root@eks:/# exit
exit
```

Note:

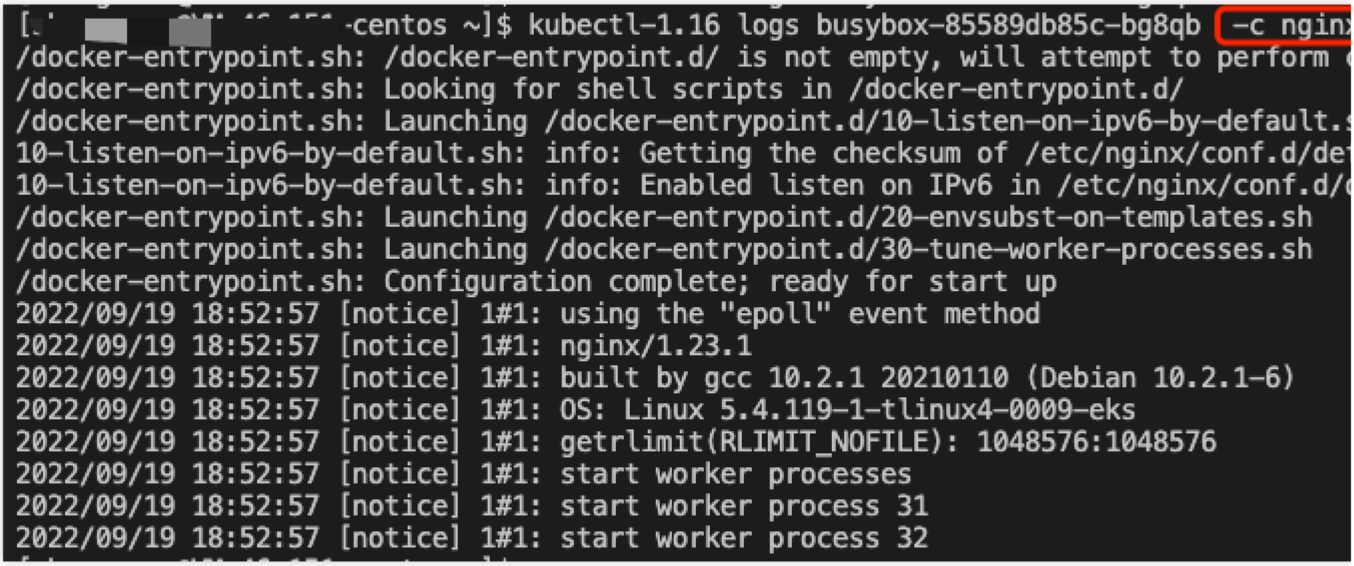
If the container names of the DaemonSet Pods are the same with those of the original Pods, the `exec` command will give priority to the original Pods, hence it becomes impossible to log in to the containers of the DaemonSet Pods. However, the containers of the DaemonSet Pods continue to function normally. For the containers of the DaemonSet Pods to be observable, it's recommended to differentiate the container names of the DaemonSet Pods from those of the Pods.

4. Viewing Daemonset Pod logs

Similar to using the `kubectl exec` command, you can utilize the `kubectl logs` command combined with the `-c` parameter to view the logs of the DaemonSet Pod containers. The command is as follows:

```
kubectl logs <daemonset-pod-name> -c <daemonset-container-name>
```

There is an example below:



```
[[redacted]-centos ~]$ kubectl-1.16 logs busybox-85589db85c-bg8qb -c nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform c
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.s
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/de
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/09/19 18:52:57 [notice] 1#1: using the "epoll" event method
2022/09/19 18:52:57 [notice] 1#1: nginx/1.23.1
2022/09/19 18:52:57 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/09/19 18:52:57 [notice] 1#1: OS: Linux 5.4.119-1-tlinux4-0009-eks
2022/09/19 18:52:57 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/09/19 18:52:57 [notice] 1#1: start worker processes
2022/09/19 18:52:57 [notice] 1#1: start worker process 31
2022/09/19 18:52:57 [notice] 1#1: start worker process 32
```

Please note that since clients of version 1.18 and above of kubectl will verify whether the Pod Spec contains the specified containers before initiating a logs request. If the specified container is absent, the following error is returned directly:

```
container xxx is not valid for pod xxx
```

In cases like these, you can switch your kubectl version to 1.16, or use the released version we will provide in the future.The download link for the kubectl 1.16 version is as follows:

```
curl -LO "https://dl.k8s.io/release/v1.16.0/bin/linux/amd64/kubectl"
```

5. Injection of rules

If it is declared that the DaemonSet Pods have been enabled on the super nodes, in addition to the Pods under the kube-system namespace, the other Pods on the super nodes will be injected with the DaemonSet Pods. If you have special Pods that are not expected to be injected with the DaemonSet Pods, you can configure the following annotation:

```
eks.tke.cloud.tencent.com/ds-injection: "false"
```

If you hope that all Pods under a specific namespace are not injected with DaemonSet Pods, you can configure the following annotation for the namespace.

```
eks.tke.cloud.tencent.com/ds-injection: "false"
```

If you hope that Pods under the kube-system namespace are injected with DaemonSet Pods, it is necessary to explicitly declare the following annotation during the creation of these Pods.

```
eks.tke.cloud.tencent.com/ds-injection: "true"
```

6. Special capabilities

Adjusting the Startup Sequence of DaemonSet Pods

If the injected DaemonSet Pods need to start before the business Pods, you can configure the following annotation to the business Pods, allowing them to start after the DaemonSet Pods.

```
eks.tke.cloud.tencent.com/start-after-ds: "true"
```

Opening ports

If the injected DaemonSet Pod needs to expose ports externally, this must be declared additionally through annotations. The method of declaration is as follows.

```
eks.tke.cloud.tencent.com/metrics-port: "9100,8080,3000-5000"
```

By default, only port 9100 is exposed, providing monitoring data metrics. You can add other ports after 9100. The declaration of a port range is supported, and multiple ports are separated by commas. Note that port 9100 can be altered but not removed, because the first port is typically utilized for monitoring data metrics by default.

When accessing the port of the DaemonSet Pod from outside the container, the IP accessed is the business Pod IP and the port is the port of the DaemonSet.

```
curl "http://<pod-ip>:<ds-port>/"
```

Communication between business Pods and local DaemonSet Pods

If the business Pods need to actively access the injected DaemonSet Pods, it is necessary to obtain the virtual IPs used by the injected DaemonSet Pods. By adding the following annotation, the business Pods can obtain the virtual IPs used by the DaemonSet Pods through env from hostIP.

```
eks.tke.cloud.tencent.com/env-host-ip: "true"
```

Similar env from hostIP is used in the container:

```
env:
- name: HOST_IP
  valueFrom:
    fieldRef:
      fieldPath: status.hostIP
```

Registered Node Management

Registered Node Overview

Last updated：2024-05-10 14:41:58

Overview

Registered nodes (third-party nodes) are a newly upgraded node product form of Tencent Kubernetes Engine (TKE) in hybrid cloud deployment. It allows users to host non-Tencent Cloud servers in the TKE cluster. Users provide computing resources while TKE is responsible for the lifecycle management of the cluster.

Note:

Registered nodes now support two product modes: the Direct Connect (DC) version (connected through DC and Cloud Connect Network (CCN)) and the public network version (connected through the Internet). Users can choose the proper version according to different scenarios as needed.

Use Cases

Resource Reuse

An enterprise needs to migrate to the cloud. However, it has investment in local data centers, and has existing server resources (CPU resources and GPU resources) in the Internet data center (IDC). By using the feature of Registered Node, users can add IDC resources to the TKE cluster, so that the existing server resources can be effectively utilized during cloudification.

Cluster Hosting and OPS

Tencent Cloud conducts unified OPS and control on the local deployment and OPS cost of Kubernetes clusters, so that users only need to maintain their local servers.

Hybrid Deployment Scheduling

Both registered nodes and Tencent Cloud Cloud Virtual Machine (CVM) nodes can be simultaneously scheduled within a single cluster, facilitating the expansion of IDC services to CVM without the need to introduce multi-cluster management.

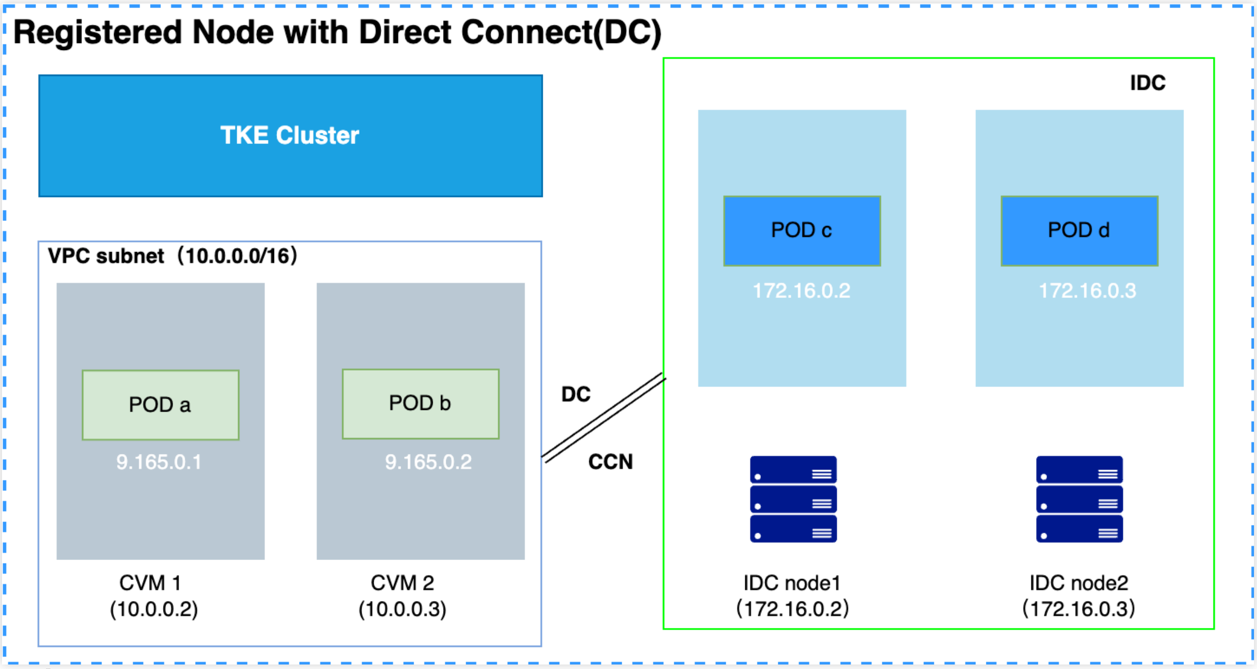
Seamless Integration with Cloud Services

Registered nodes seamlessly integrate with cloud-native services of Tencent Cloud, covering cloud-native capabilities such as logs, monitoring, auditing, storage, and container security.

Registered Nodes with DC

Architecture

Users can connect their own IDC environment to Tencent Cloud Virtual Private Cloud (VPC) through DC and CCN, and then connect the IDC nodes to the TKE cluster through the private network, achieving unified management of IDC nodes and cloud-based CVM nodes. The architecture diagram is as follows:



Constraints

To ensure the stability of registered nodes, users can access registered nodes only through DC or CCN (the virtual private network (VPN) is currently not supported).

Registered nodes must use TencentOS Server 3.1 or TencentOS Server 2.4 (TK4).

Graphics processing unit (GPU): Only NVIDIA series of GPUs are supported, including Volta (such as V100), Turing (such as T4), and Ampere (such as A100 and A10).

The registered node feature is only available for TKE clusters of **v1.18 or later versions** and the cluster must contain at least one CVM node.

For scenarios where CVM nodes and IDC nodes are deployed, the TKE team has launched a hybrid cloud container network scheme based on Cilium-Overlay.

Port Connectivity Configuration of Nodes

To ensure the connectivity between CVM nodes and IDC nodes in a hybrid cloud cluster, a series of ports need to be configured on the CVM nodes and IDC nodes.

CVM nodes: CVM nodes must use the security group settings that meet TKE requirements. If the TKE cluster uses the Cilium-Overlay network mode, additional security group rules need to be added.

Inbound rule

Protocol	Port	Source	Policy	Remarks
UDP	8472	Cluster CIDR IDC CIDR	Allow	VXLAN communication is allowed for cluster nodes.

Outbound rule

Protocol	Port	Destination	Policy	Remarks
UDP	8472	Cluster CIDR IDC CIDR	Allow	VXLAN communication is allowed for cluster nodes.

IDC nodes: Configure ports on nodes for firewall rules.

Inbound rule

Protocol	Port	Source	Policy	Remarks
UDP	8472	Cluster CIDR IDC CIDR	Allow	VXLAN communication is allowed for cluster nodes.
TCP	10250	VXLAN communication is allowed for cluster nodes.	Allow	API server communication is allowed.

Outbound rule

Protocol	Port	Destination	Policy	Remarks
UDP	8472	Cluster CIDR IDC CIDR	Allow	VXLAN communication is allowed for cluster nodes.
TCP	80, 443, 9243, 10250, 60002	VPC subnet CIDR with proxy	Allow	Tencent Cloud proxy communication is allowed.

Network Mode

For different network modes of TKE clusters, there are certain limitations of pod network capabilities as follows:

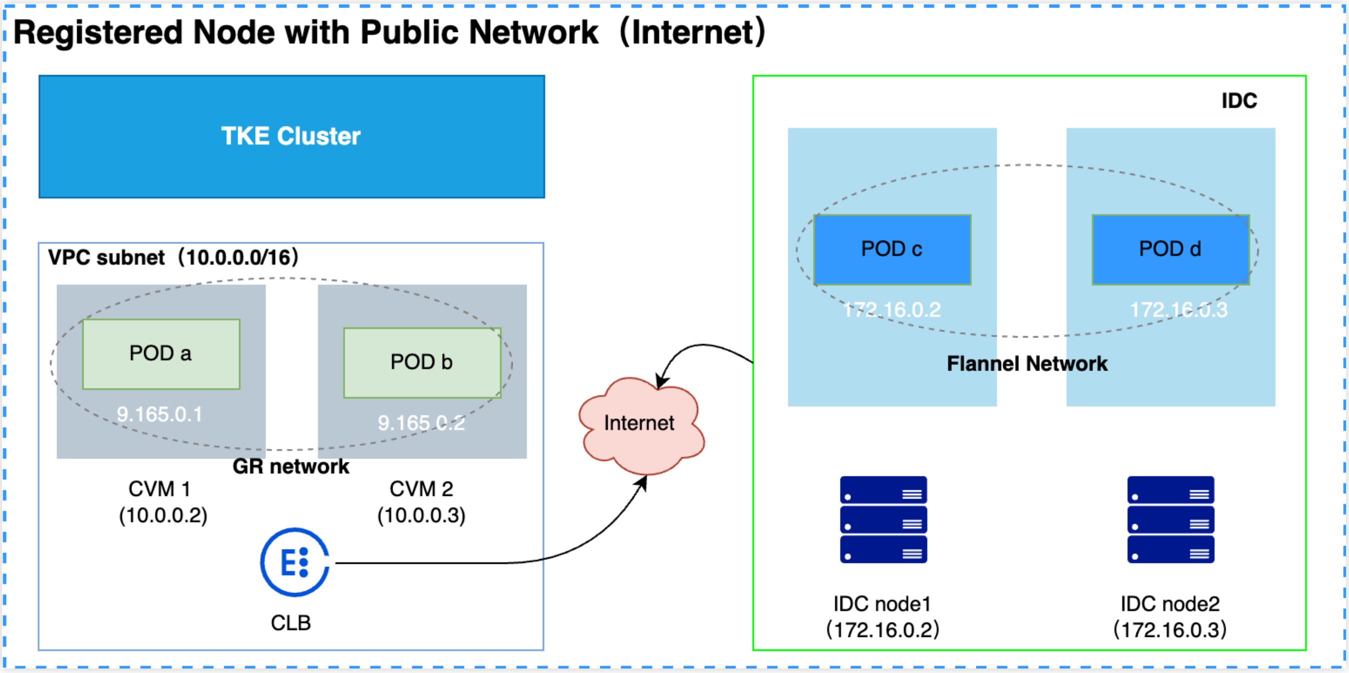
For clusters using the GlobalRouter or VPC-CNI exclusive ENI mode: Pods on IDC nodes can only use the hostNetwork network mode. In this mode, pods of IDC nodes can communicate with CVM nodes only through the host network.

For clusters using the Cilium-Overlay network mode: This is a container network solution specially designed by the TKE team for hybrid cloud scenarios. Both CVM pods and IDC pods are on the same overlay network plane and can communicate with each other.

Registered Node with Public Network (Internet Version)

Architecture

If a user fails to establish DC between the IDC and Tencent Cloud due to certain objective factors, but still wants to manage the IDC nodes through TKE to reduce the deployment and OPS cost of Kubernetes, the user can use the registered nodes of the public network version to register the IDC nodes to TKE for unified management over the Internet. The architecture diagram is as follows:



Note:

Unlike the DC version, the public network version can only communicate with TKE through the Internet. By default, the CVM nodes and IDC nodes are two completely isolated partitions, and the pods in CVM nodes cannot communicate with pods in IDC nodes through networks. Therefore, it is recommended that users manage and schedule IDC nodes as a separate node pool to prevent communication between CVM pods and IDC pods.

Constraints

Before using registered nodes of the public network version, it is necessary to ensure that the environment meets the constraint requirements. Otherwise, the product features may be abnormal.

Operating system: Registered nodes of the public network version must use TencentOS Server 3.1 and TencentOS Server 2.4 (TK4).

TKE cluster:

The Kubernetes version must be 1.20 or later.

Set "Container network add-on" to **Global Router**.

There must be at least one CVM node.

Network: IDC nodes can communicate with Tencent Cloud Cloud Load Balancer (CLB) and can access TCP ports 443 and 9000 of CLB.

Hardware (GPU): GPU nodes are currently not supported.

Port Connectivity Configuration of Nodes

To ensure the connectivity between the Tencent Cloud and IDC through Internet, a series of ports need to be configured on the CVM nodes and IDC nodes.

CVM nodes: CVM nodes must use the security group settings that meet TKE requirements. If the TKE cluster uses the Cilium-Overlay network mode, additional security group rules need to be added.

IDC nodes: Configure ports on nodes for firewall rules to allow access to ports and also configure rules to allow access to the public network image repository.

Image repository: Ensure that **ccr.ccs.tencentyun.com** and **superedge.tencentcloudcr.com** can be accessed on IDC nodes.

Inbound rule

Protocol	Port	Source	Policy	Remarks
UDP	8472	Cluster CIDR IDC CIDR	Allow	VXLAN communication is allowed for cluster nodes.
TCP	10250	VXLAN communication is allowed for cluster nodes.	Allow	API server communication is allowed.

Outbound rule

Protocol	Port	Destination	Policy	Remarks
UDP	8472	Cluster CIDR IDC CIDR	Allow	VXLAN communication is allowed for cluster nodes.
TCP	443, 9000	IP address of CLB	Allow	The CLB address can be accessed to provide the

				node registration and cloud-edge tunnel services.

Network Mode

For the public network version of registered nodes, the network between CVM and IDC are naturally isolated. Therefore, CVM nodes use the Global Router (GR) network to achieve pod communication between CVM nodes, while the IDC nodes use the Flannel network to achieve pod communication between IDC nodes. By default, CVM pods are isolated from IDC pods.

Comparison of Capabilities Between Registered Nodes and TKE Nodes

Class	Capability	TKE Node	Registered Node (DC)	Registered Node (Public Network)
Node management	Node adding	✔	✔	✔
	Node removal	✔	✔	✔
	Setting of node tags and taints	✔	✔	✔
	Node draining and cordoning	✔	✔	✔
	Batch node management in the node pool	✔	✔	✔
	Kubernetes upgrade	✔	Partial support	Partial support
Storage volume	Local storage (emptyDir, hostPath, etc)	✔	✔	✔
	Kubernetes API (ConfigMap, Secret, etc)	✔	✔	✔
	Cloud Block Storage (CBS)	✔	-	-
	Cloud File Storage (CFS)	✔	✔	-
	Cloud Object Storage	✔	✔	-

	(COS)			
Observability	Prometheus monitoring	✔	✔	-
	Cloud product monitoring	✔	-	-
	CLS	✔	✔	-
	Cluster audit	✔	✔	✔
	Event storage	✔	✔	✔
Service	ClusterIP service	✔	✔	✔
	NodePort service	✔	✔	✔
	LoadBalancer service	✔	✔	-
	CLB ingress	✔	✔	-
	Nginx ingress	✔	✔	-
Others	qGPU	✔	-	-

Creating a Registered Node

Last updated：2024-05-10 14:42:14

Directions

Installing Operating Systems for Registered Nodes

Currently, registered nodes must use [TencentOS Server 3.1] or [TencentOS Server 2.4 (TK4)]. Details are as follows:

Operating System	Description	Download URL
TencentOS Server 3.1	It is compatible with the CentOS 8 user mode and uses the T-Kernel 4 deeply optimized based on LTS kernel 5.4.	Download URL
TencentOS Server 2.4 (TK4)	It is compatible with the CentOS 7 user mode and uses the T-Kernel 4 deeply optimized based on LTS kernel 5.4.	Download URL

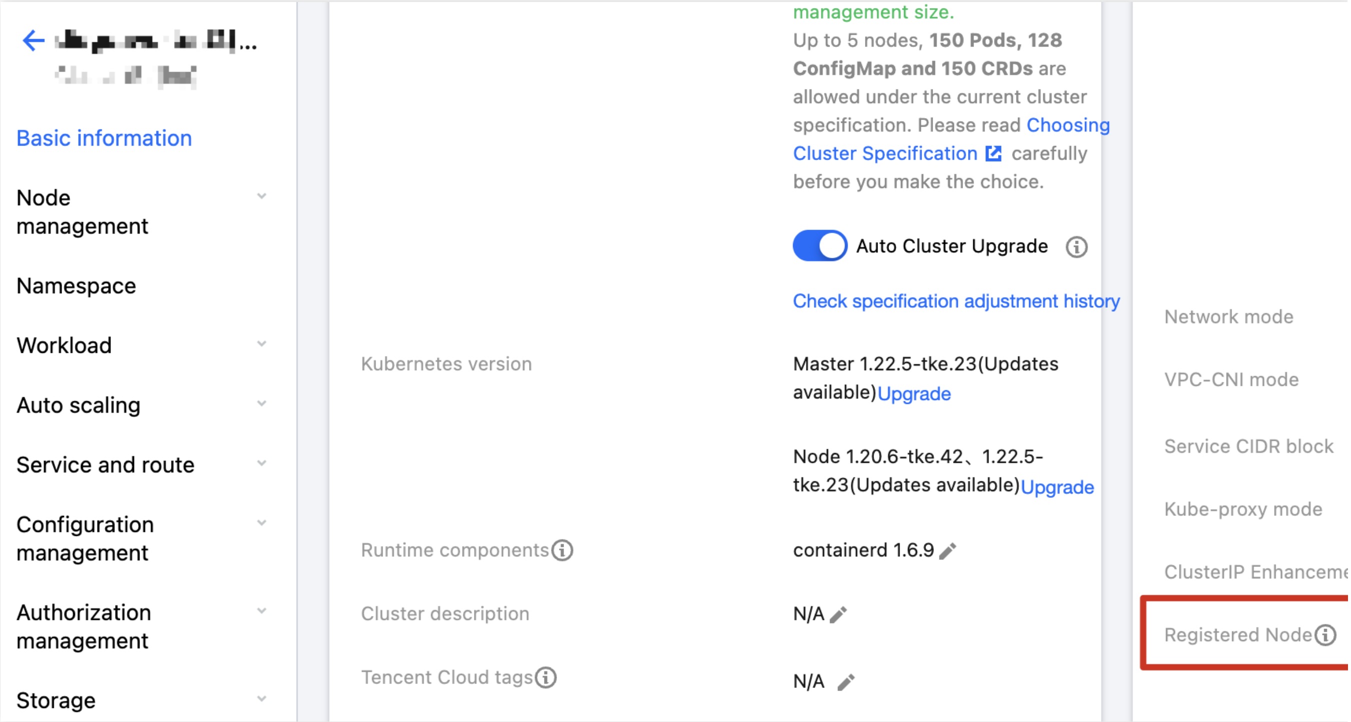
Note:
TencentOS Server is the Linux operating system designed by Tencent for cloud scenarios. With specific features and optimized performance, it provides a high-performance, secure, and reliable operating environment for applications in Cloud Virtual Machine (CVM) instances.

Enabling Support for Registered Nodes

Cluster of the Global Router Mode

If your cluster is networked in Global Router (GR) mode, you can enable registered nodes of the Direct Connect (DC) version and the public network version.

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the desired cluster ID to go to the **Basic information** page.
3. Click the Registered Node switch.



← [redacted]

Basic information

Node management

Namespace

Workload

Auto scaling

Service and route

Configuration management

Authorization management

Storage

management size.
Up to 5 nodes, **150 Pods, 128 ConfigMap and 150 CRDs** are allowed under the current cluster specification. Please read Choosing Cluster Specification carefully before you make the choice.

Auto Cluster Upgrade

Check specification adjustment history

Kubernetes version: Master 1.22.5-tke.23(Updates available)Upgrade

Node 1.20.6-tke.42、1.22.5-tke.23(Updates available)Upgrade

Runtime components: containerd 1.6.9

Cluster description: N/A

Tencent Cloud tags: N/A

Network mode

VPC-CNI mode

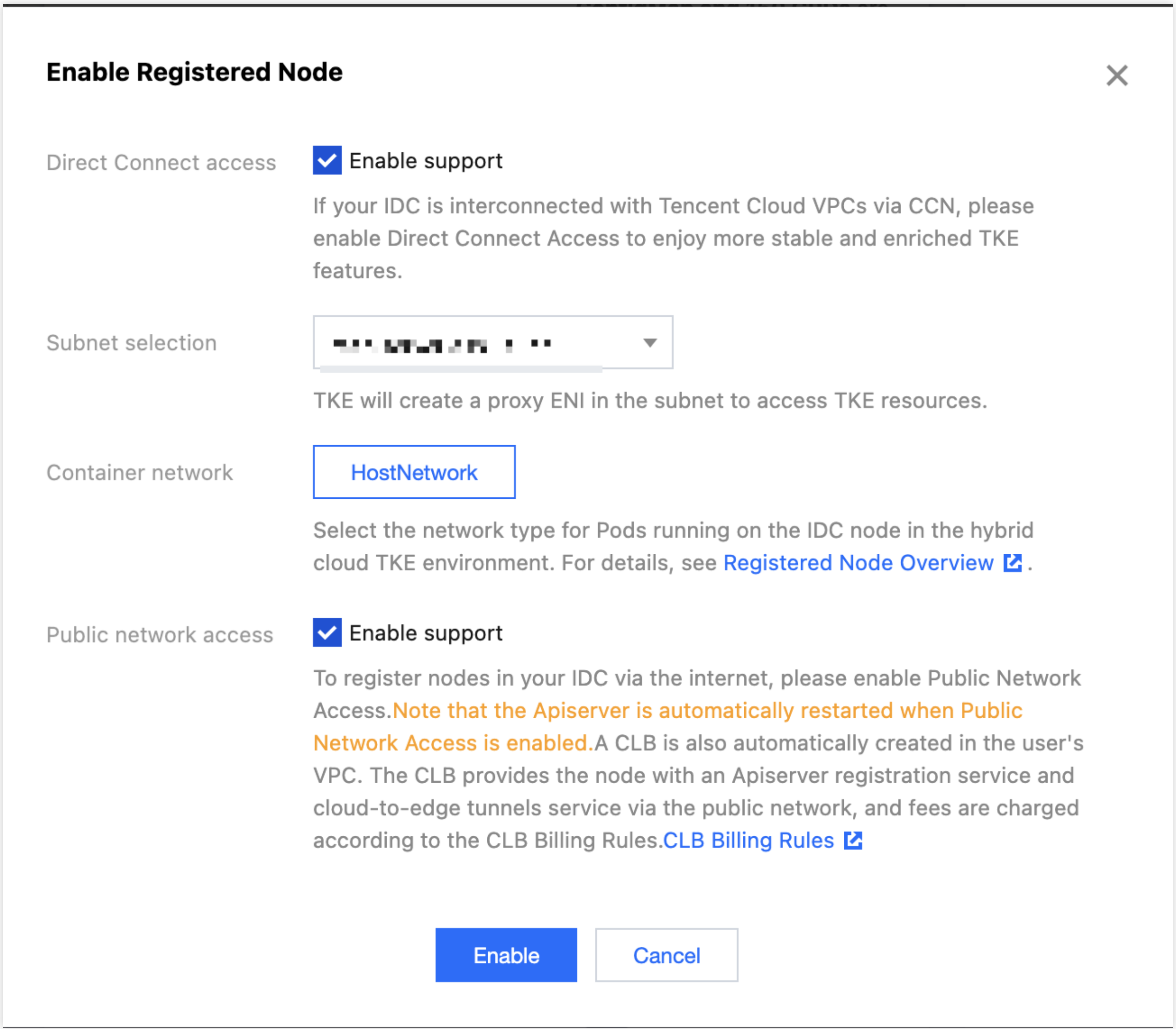
Service CIDR block

Kube-proxy mode

ClusterIP Enhanceme

Registered Node

4. You can enable direct connect access and public network access seperately.



Enable Registered Node

Direct Connect access ☑ Enable support

If your IDC is interconnected with Tencent Cloud VPCs via CCN, please enable Direct Connect Access to enjoy more stable and enriched TKE features.

Subnet selection [redacted]

TKE will create a proxy ENI in the subnet to access TKE resources.

Container network HostNetwork

Select the network type for Pods running on the IDC node in the hybrid cloud TKE environment. For details, see Registered Node Overview .

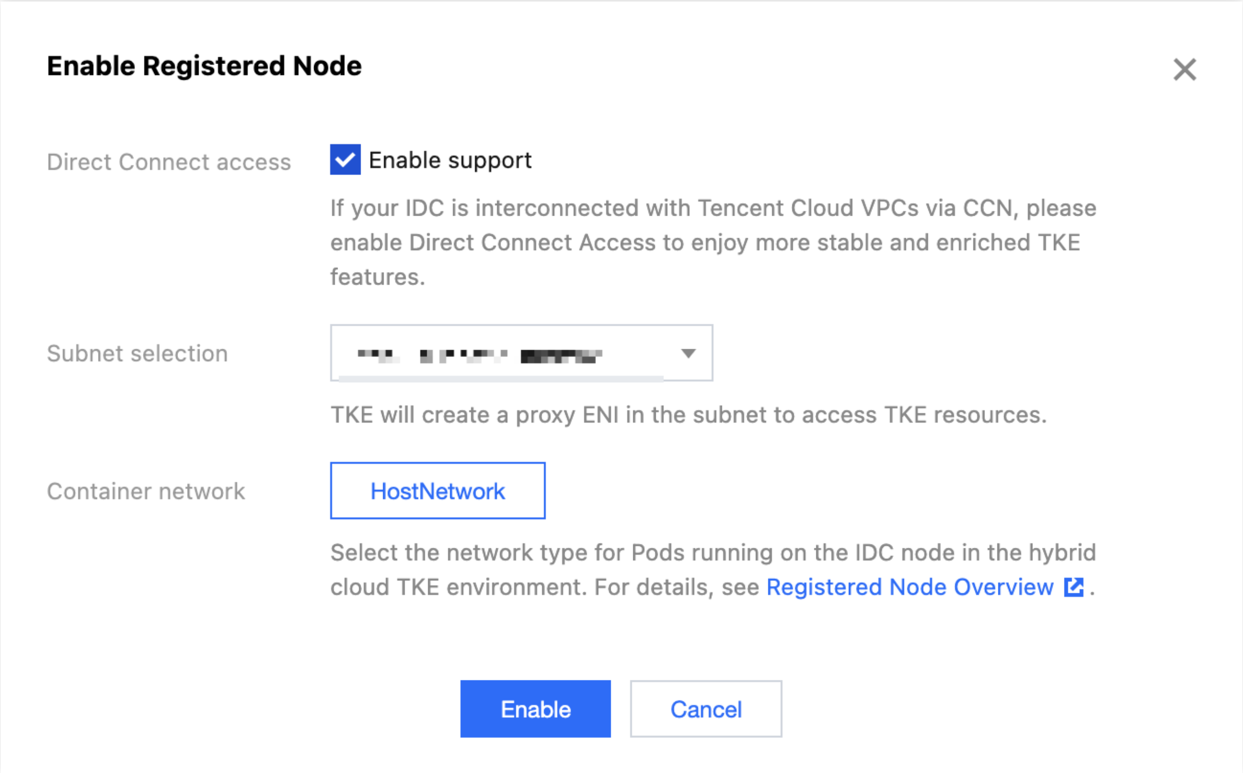
Public network access ☑ Enable support

To register nodes in your IDC via the internet, please enable Public Network Access.Note that the Apiserver is automatically restarted when Public Network Access is enabled.A CLB is also automatically created in the user's VPC. The CLB provides the node with an Apiserver registration service and cloud-to-edge tunnels service via the public network, and fees are charged according to the CLB Billing Rules.CLB Billing Rules

5. Click **Enable**.

Cluster of the VPC-CNI Mode

If your cluster is networked in VPC-CNI mode, you can only enable registered nodes of the DC version as follows:



Enable Registered Node

Direct Connect access ☑ Enable support

If your IDC is interconnected with Tencent Cloud VPCs via CCN, please enable Direct Connect Access to enjoy more stable and enriched TKE features.

Subnet selection [redacted]

TKE will create a proxy ENI in the subnet to access TKE resources.

Container network HostNetwork

Select the network type for Pods running on the IDC node in the hybrid cloud TKE environment. For details, see Registered Node Overview.

Cluster of the Cilium-Overly Mode

If your cluster is networked in Cilium-Overlay mode, registered nodes can be automatically added to the cluster. Set **Container network add-on** to **Cilium-Overlay**.

Subnet: TKE will create a proxy ENI in the selected subnet for registered nodes to access cloud resources.

After the Cilium-Overlay cluster is created, the registered node feature is enabled by default. To query related information, log in to the TKE console and choose **Basic information** > **Node management** > **Worker node**.

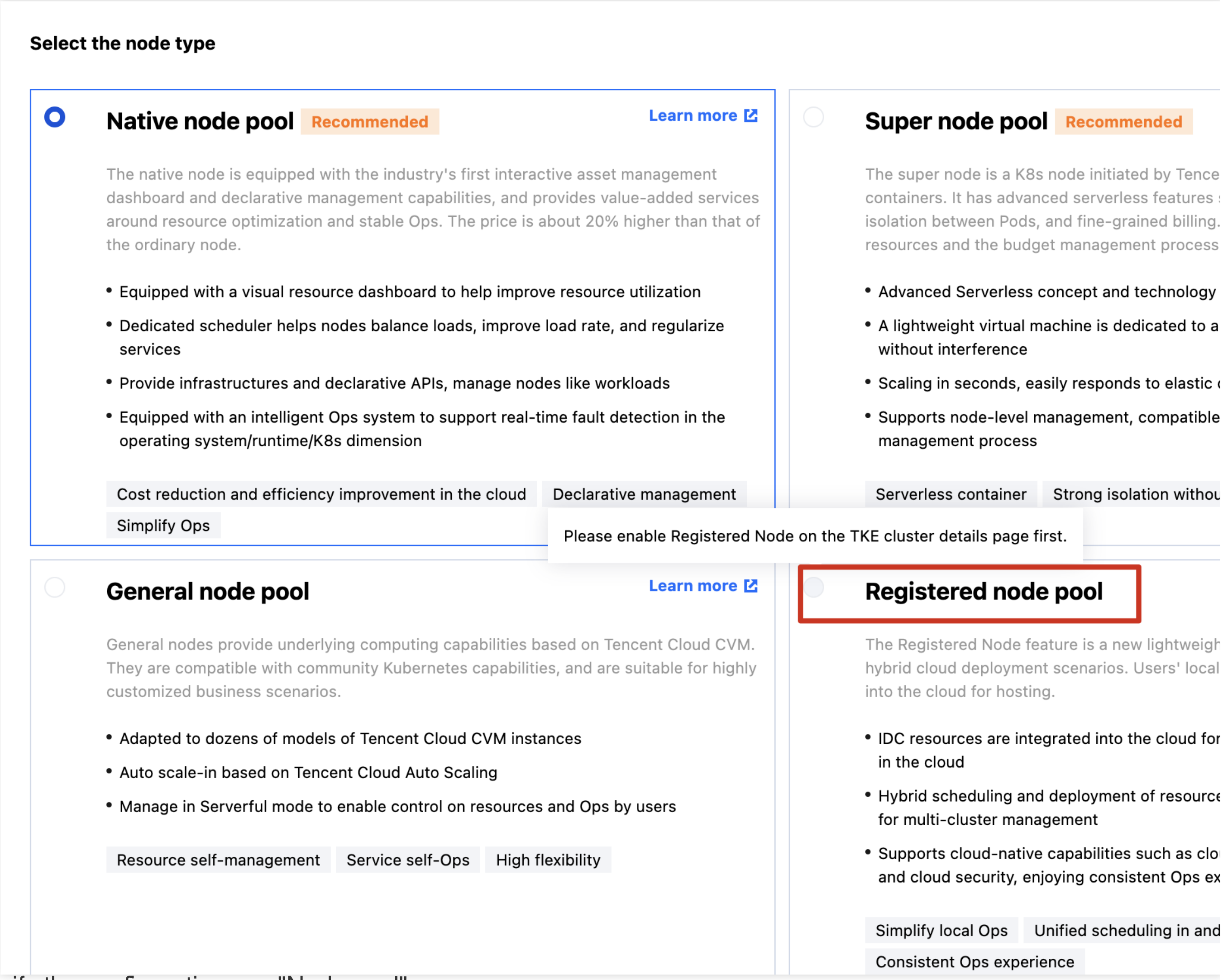
Adding a Registered Node

Creating a Registered Node Pool

Note:

Registered nodes can be managed only through the **registered node pool**.

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the desired cluster ID to go to the **Basic information** page.
3. Choose **Node management** > **Worker node** in the left sidebar to go to the **Node pool** tag page.
4. Click **Create** to open the "Select the node type" page, and select **Registered node pool**.



Select the node type

● Native node pool Recommended Learn more

The native node is equipped with the industry's first interactive asset management dashboard and declarative management capabilities, and provides value-added services around resource optimization and stable Ops. The price is about 20% higher than that of the ordinary node.

- Equipped with a visual resource dashboard to help improve resource utilization
- Dedicated scheduler helps nodes balance loads, improve load rate, and regularize services
- Provide infrastructures and declarative APIs, manage nodes like workloads
- Equipped with an intelligent Ops system to support real-time fault detection in the operating system/runtime/K8s dimension

Cost reduction and efficiency improvement in the cloud | Declarative management | Simplify Ops

○ Super node pool Recommended

The super node is a K8s node initiated by Tence containers. It has advanced serverless features isolation between Pods, and fine-grained billing. resources and the budget management process

- Advanced Serverless concept and technology
- A lightweight virtual machine is dedicated to a without interference
- Scaling in seconds, easily responds to elastic
- Supports node-level management, compatible management process

Serverless container | Strong isolation withou

Please enable Registered Node on the TKE cluster details page first.

○ General node pool Learn more

General nodes provide underlying computing capabilities based on Tencent Cloud CVM. They are compatible with community Kubernetes capabilities, and are suitable for highly customized business scenarios.

- Adapted to dozens of models of Tencent Cloud CVM instances
- Auto scale-in based on Tencent Cloud Auto Scaling
- Manage in Serverful mode to enable control on resources and Ops by users

Resource self-management | Service self-Ops | High flexibility

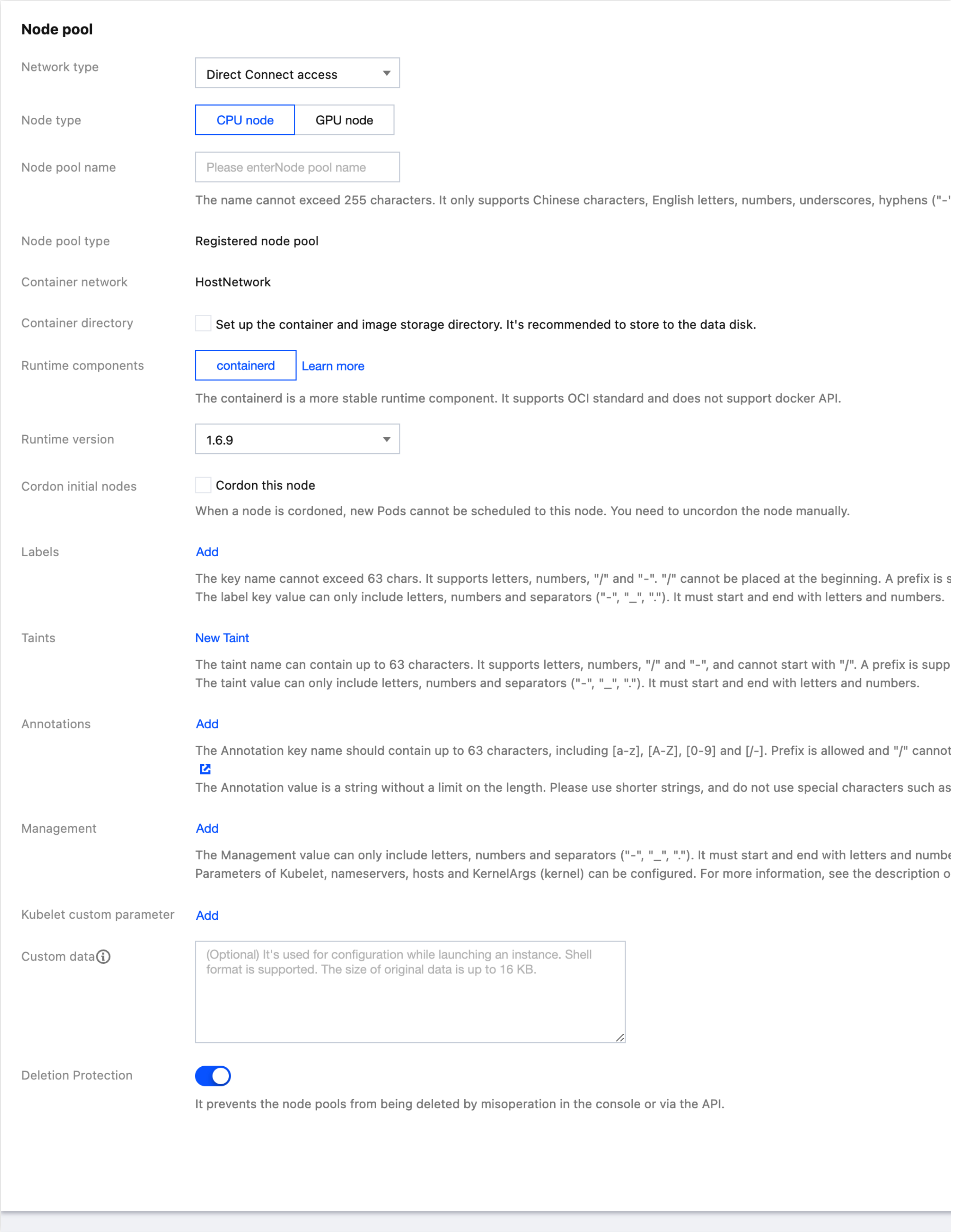
○ Registered node pool

The Registered Node feature is a new lightweigh hybrid cloud deployment scenarios. Users' local into the cloud for hosting.

- IDC resources are integrated into the cloud for in the cloud
- Hybrid scheduling and deployment of resource for multi-cluster management
- Supports cloud-native capabilities such as clo and cloud security, enjoying consistent Ops ex

Simplify local Ops | Unified scheduling in and | Consistent Ops experience

5. Specify the configurations on "Node pool" page.



Node pool

Network type: Direct Connect access

Node type: CPU node | GPU node

Node pool name: Please enterNode pool name

The name cannot exceed 255 characters. It only supports Chinese characters, English letters, numbers, underscores, hyphens ("-'

Node pool type: Registered node pool

Container network: HostNetwork

Container directory: □ Set up the container and image storage directory. It's recommended to store to the data disk.

Runtime components: containerd Learn more

The containerd is a more stable runtime component. It supports OCI standard and does not support docker API.

Runtime version: 1.6.9

Cordon initial nodes: □ Cordon this node

When a node is cordoned, new Pods cannot be scheduled to this node. You need to uncordon the node manually.

Labels: Add

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is s
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Taints: New Taint

The taint name can contain up to 63 characters. It supports letters, numbers, "/" and "-", and cannot start with "/". A prefix is supp
The taint value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Annotations: Add

The Annotation key name should contain up to 63 characters, including [a-z], [A-Z], [0-9] and [/-]. Prefix is allowed and "/" cannot
The Annotation value is a string without a limit on the length. Please use shorter strings, and do not use special characters such as

Management: Add

The Management value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbe
Parameters of Kubelet, nameservers, hosts and KernelArgs (kernel) can be configured. For more information, see the description o

Kubelet custom parameter: Add

Custom data: (Optional) It's used for configuration while launching an instance. Shell format is supported. The size of original data is up to 16 KB.

Deletion Protection: ●

It prevents the node pools from being deleted by misoperation in the console or via the API.

Network type: Select **Direct Connect access** or **Public Network access**.

Node type: Select **CPU node** or **GPU node**. This selection is available only DC access.

Node pool name: It specifies the name of the node pool.

Container directory: Select this option to set up the container and image storage directory. It is recommended to store to the data disk, such as `/var/lib/docker` .

Runtime components: It specifies the runtime component of the container. **docker** and **containerd** are supported.

Runtime version: It specifies the version of the runtime component.

Cordon initial nodes: If **Cordon this node** is selected, new pods cannot be scheduled to this node. You can uncordon the node manually, or run the uncordon command in custom data as needed.

Labels: Click **Add** and customize the settings of the label. The specified label will be automatically added to nodes created in the node pool to help filter and manage external nodes using labels.

Taints: This is a node-level attribute and is usually used with `Tolerations` . You can specify this parameter for all the nodes in the node pool, so as to stop scheduling pods that do not meet the requirements to these nodes and drain such pods from the nodes.

The value of **Taints** usually consists of `key` , `value` , and `effect` . Valid values of `effect` :

PreferNoSchedule: Optional. Try not to schedule a pod to a node with a taint that cannot be tolerated by the pod.

NoSchedule: When a node contains a taint, a pod without the corresponding toleration must not be scheduled.

NoExecute: When a node contains a taint, a pod without the corresponding toleration to the taint are not be scheduled to the node and any such pods already on the node are drained.

Annotations: You can use Kubernetes annotations to attach arbitrary non-identifying metadata to objects.

Management: Parameters of kubelet, nameservers, hosts, and KernelArgs (kernel) can be configured.

Kubelet custom parameter: You can customize kubelet parameters.

Custom data: Specify custom data to configure the node, that is, to run the configured script when the node is started. You need to ensure the reentrant and retry logic of the script. The script and its log files can be viewed in the node path: `/usr/local/qcloud/tke/userscript` .

6. Click **Create**.

Adding a Registered Node

Do the following to add registered nodes to the node pool:

1. On the node pool page, click the desired node pool ID.
2. On the node pool details page, click **Create node** to get the script.
3. In the **CPU node initialization script** window, select a value for Node initialization method, and copy or download the script. When this is a **"Direct Connect access"** node pool:

Public network: Selected by default. For an IDC node, directly download the installation script file (31 KB in size) over the public network.

Private network: If your IDC node cannot access the public network, access the private network through a DC line to download the installation script file.

4. When this is a **"Public Network access"** node pool, the script is generated automatically.
5. Run the script on your server.

Note:

The script download link is valid for 1 hour. Since the script is downloaded by using COS, you need to ensure that the IDC node can access COS through the private/public network.

6. Run the following command to add the **"Direct Connect access"** node:

```
./ add2tkectl-cls-m57oxxxp-np-xxxx install
```

Note:

If the node fails to be added because the related Docker and containerd add-ons are installed on an external node, run the following command to delete the add-ons and add the node again:

```
./add2tkectl-cls-m57oxxxp-np-xxxx clear
```

7. Run the following command to add the **"Public Network access"** node:

```
./ edgectl install -n [nodeName]
```

Note:

If the node fails to be added because the related Docker and containerd add-ons are installed on an external node, run the following command to delete the add-ons and add the node again:

```
./edgectl clear
```

Memory Compression Instructions

Instructions

Last updated：2024-06-14 16:28:43

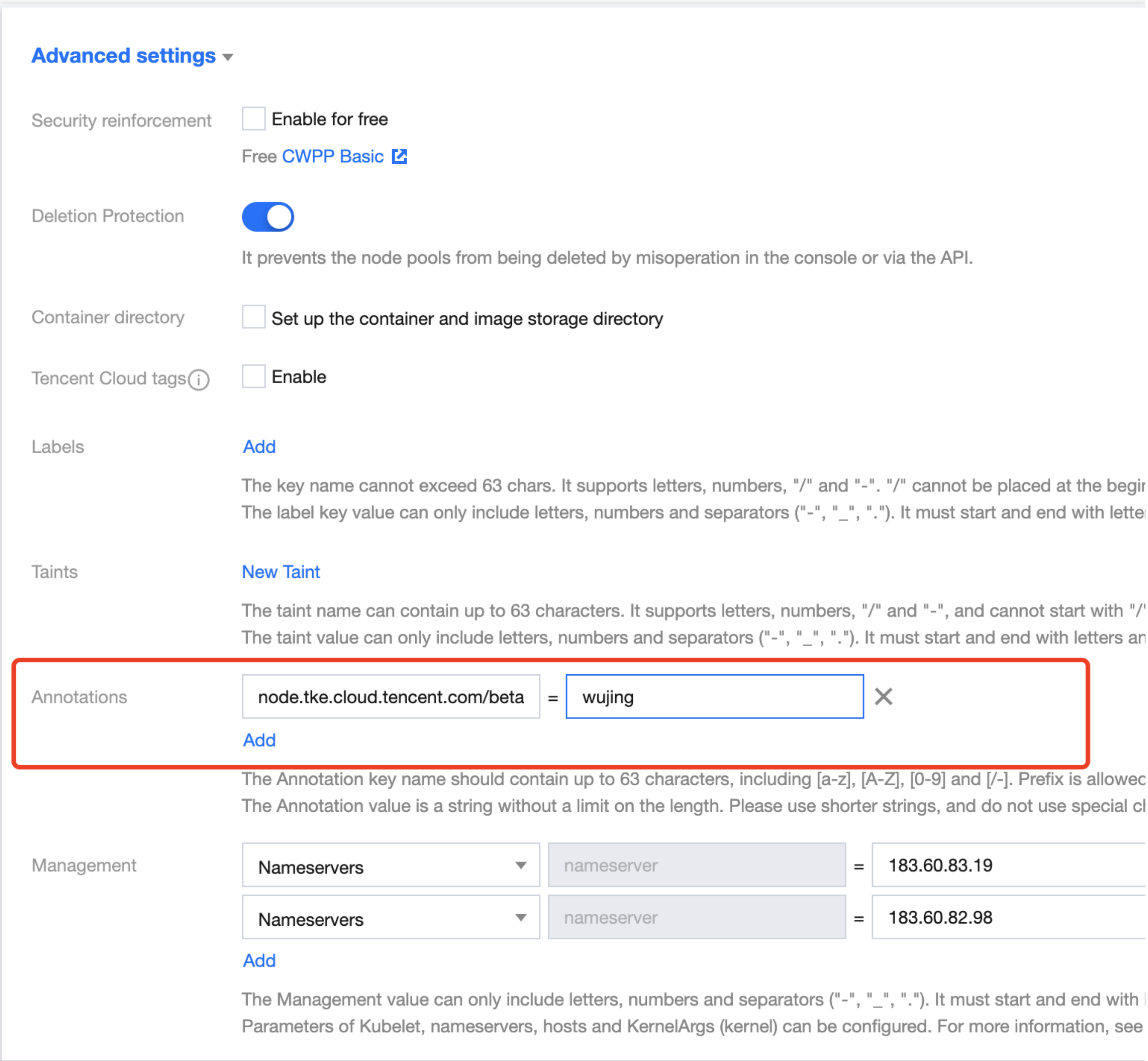
This document describes how to activate and use the memory compression feature based on native nodes.

Environment preparations

The memory compression feature requires updating the kernel of the native node image to the latest version (**5.4.241-19-0017**), which can be achieved using the following methods:

Adding Native Nodes

1. Log in to the Tencent Kubernetes Engine Console, and choose **Cluster** from the left navigation bar.
2. In the cluster list, click on the desired cluster ID to access its details page.
3. Choose **Node Management > Worker Nodes**, select the **Node Pool** tab, and click **Create.**
4. Select **native nodes**, and click **Create.**
5. In **Advanced Settings** of the Create Node Pool page, find the **Annotations** field and set "**node.tke.cloud.tencent.com/beta-image = wujing**", as shown in the following figure:



Advanced settings ▾

Security reinforcement □ Enable for free
Free CWPP Basic

Deletion Protection ●
It prevents the node pools from being deleted by misoperation in the console or via the API.

Container directory □ Set up the container and image storage directory

Tencent Cloud tags □ Enable

Labels Add
The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the begi
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with lette

Taints New Taint
The taint name can contain up to 63 characters. It supports letters, numbers, "/" and "-", and cannot start with "/'
The taint value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters an

Annotations node.tke.cloud.tencent.com/beta = wujing ✕
Add
The Annotation key name should contain up to 63 characters, including [a-z], [A-Z], [0-9] and [/-]. Prefix is allowec
The Annotation value is a string without a limit on the length. Please use shorter strings, and do not use special cl

Management Nameservers ▾ nameserver = 183.60.83.19
Nameservers ▾ nameserver = 183.60.82.98
Add
The Management value can only include letters, numbers and separators ("-", "_", "."). It must start and end with l
Parameters of Kubelet, nameservers, hosts and KernelArgs (kernel) can be configured. For more information, see

6. Click **Create node pool**.

Note:

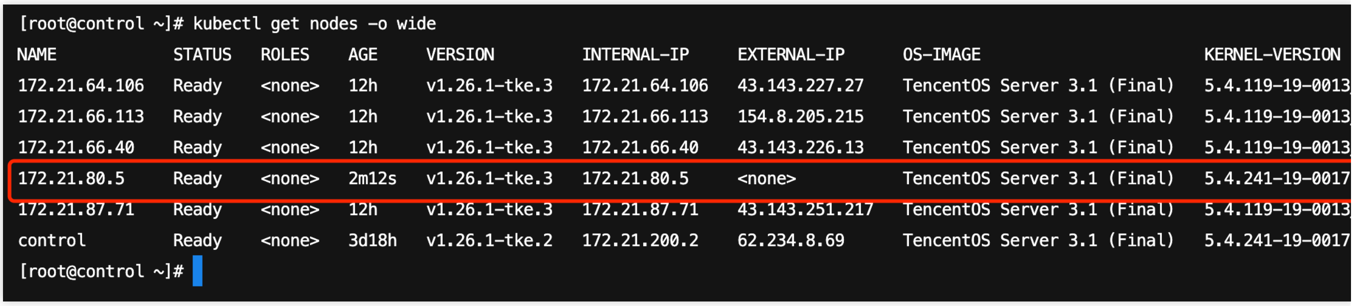
By default, the image with the latest kernel version (**5.4.241-19-0017**) will be installed on the native nodes added to this node pool.

Existing Native Nodes

The kernel versions of existing native nodes can be updated using RPM packages. You can contact us through Submit a Ticket.

Kernel Version Verification

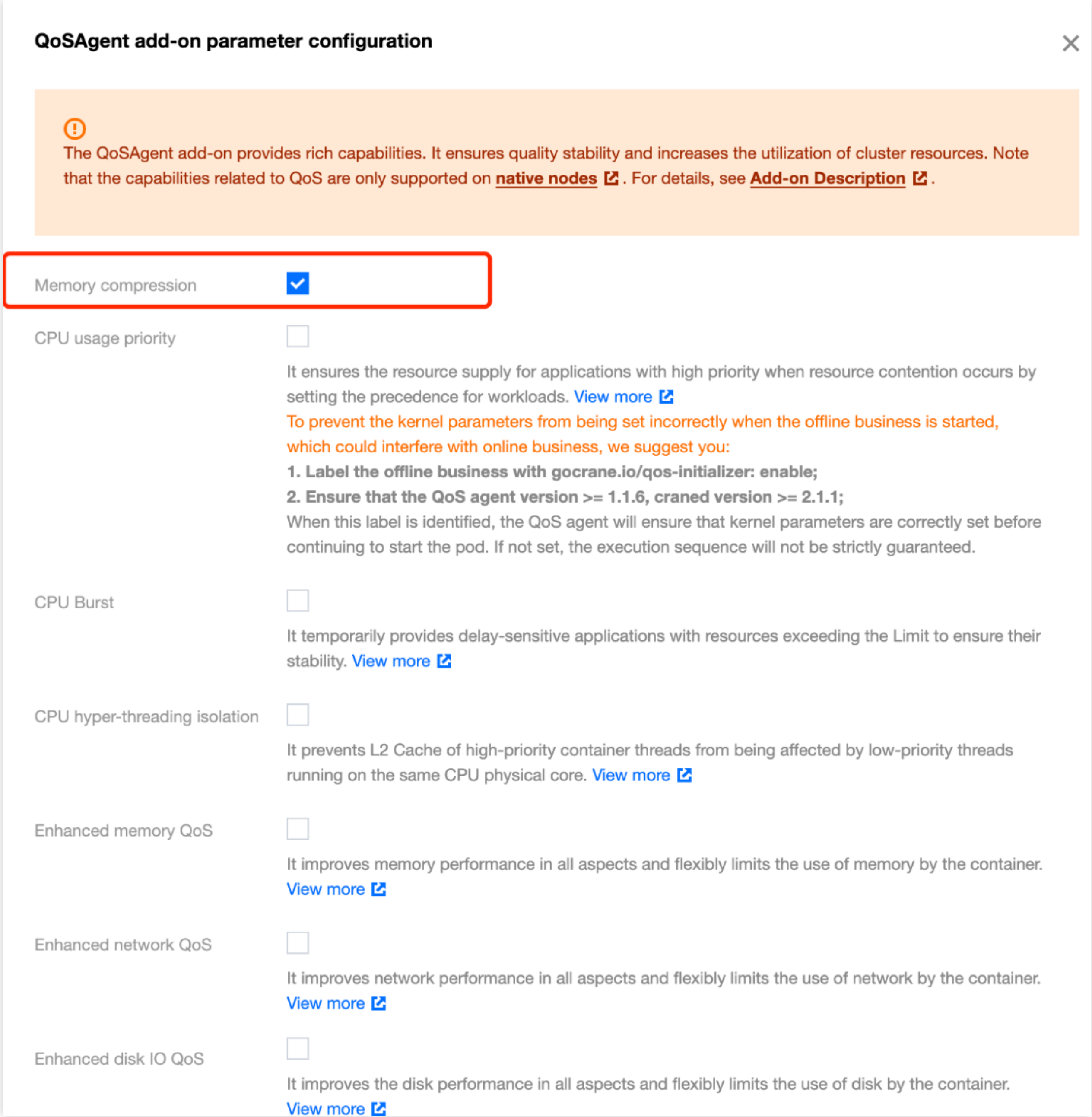
You can run the `kubectl get nodes -o wide` command to verify that the node's KERNEL-VERSION has been updated to the latest version **5.4.241-19-0017.1_plus**.



```
[root@control ~]# kubectl get nodes -o wide
NAME            STATUS   ROLES    AGE     VERSION         INTERNAL-IP     EXTERNAL-IP      OS-IMAGE                      KERNEL-VERSION
172.21.64.106   Ready    <none>   12h     v1.26.1-tke.3   172.21.64.106   43.143.227.27    TencentOS Server 3.1 (Final)  5.4.119-19-0013
172.21.66.113   Ready    <none>   12h     v1.26.1-tke.3   172.21.66.113   154.8.205.215    TencentOS Server 3.1 (Final)  5.4.119-19-0013
172.21.66.40    Ready    <none>   12h     v1.26.1-tke.3   172.21.66.40    43.143.226.13    TencentOS Server 3.1 (Final)  5.4.119-19-0013
172.21.80.5     Ready    <none>   2m12s   v1.26.1-tke.3   172.21.80.5     <none>           TencentOS Server 3.1 (Final)  5.4.241-19-0017
172.21.87.71    Ready    <none>   12h     v1.26.1-tke.3   172.21.87.71    43.143.251.217   TencentOS Server 3.1 (Final)  5.4.119-19-0013
control         Ready    <none>   3d18h   v1.26.1-tke.2   172.21.200.2    62.234.8.69      TencentOS Server 3.1 (Final)  5.4.241-19-0017
[root@control ~]#
```

Installing the QosAgent Component

1. Log in to the Tencent Kubernetes Engine Console, and choose **Cluster** from the left navigation bar.
2. In the cluster list, click on the desired cluster ID to access its details page.
3. Choose **Component Management** from the left-side menu, and click **Create** on the Component Management page.
4. On the **New Component Management** page, select **QoS Agent** and check **Memory Compression** in the parameter configurations, as shown in the following figure:



QoSAgent add-on parameter configuration

The QoSAgent add-on provides rich capabilities. It ensures quality stability and increases the utilization of cluster resources. Note that the capabilities related to QoS are only supported on **native nodes**. For details, see **Add-on Description**.

Memory compression ☑

CPU usage priority □
It ensures the resource supply for applications with high priority when resource contention occurs by setting the precedence for workloads. View more
To prevent the kernel parameters from being set incorrectly when the offline business is started, which could interfere with online business, we suggest you:
1. Label the offline business with gocrane.io/qos-initializer: enable;
2. Ensure that the QoS agent version >= 1.1.6, craned version >= 2.1.1;
When this label is identified, the QoS agent will ensure that kernel parameters are correctly set before continuing to start the pod. If not set, the execution sequence will not be strictly guaranteed.

CPU Burst □
It temporarily provides delay-sensitive applications with resources exceeding the Limit to ensure their stability. View more

CPU hyper-threading isolation □
It prevents L2 Cache of high-priority container threads from being affected by low-priority threads running on the same CPU physical core. View more

Enhanced memory QoS □
It improves memory performance in all aspects and flexibly limits the use of memory by the container. View more

Enhanced network QoS □
It improves network performance in all aspects and flexibly limits the use of network by the container. View more

Enhanced disk IO QoS □
It improves the disk performance in all aspects and flexibly limits the use of disk by the container. View more

5. Click **OK**.

6. On the **New Component Management** page, click **Complete** to install the component.

Note:

The QosAgent component of the version 1.1.5 or later supports memory compression. If the component has been installed in your cluster, perform the following steps:

1. On the cluster's **Component Management** page, find the successfully deployed QosAgent and click the right side **Upgrade.**

2. After the upgrade, click **Update Configuration** and select **Memory Compression**.

3. Click **Complete.**

Selecting Nodes to Enable Memory Compression

To facilitate Gray Box Testing, QosAgent does not enable kernel configurations required for memory compression on all native nodes by default. You need to use **NodeQOS** to specify which nodes can have Compression Capability enabled.

Deploying the NodeQOS Object

1. **Deploy the NodeQOS object.** Use `spec.selector.matchLabels` to specify on which nodes to enable memory compression, as shown in the following example:

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

```
metadata:
  name: compression
spec:
  selector:
    matchLabels:
      compression: enable
  memoryCompression:
    enable: true
```

2. **Label the node to associate the node with NodeQOS.** Perform the following steps:

2.1 Log in to the Tencent Kubernetes Engine Console, and choose **Cluster** from the left navigation bar.

2.2 In the cluster list, click on the desired cluster ID to access its details page.

2.3 Choose **Node Management** > **Worker Nodes**, select the **Node Pool** tab, and click **Edit** on the Node Pool tab page.

2.4 On the **Adjust Node Pool Configuration** page, modify the label and check **Apply this update to existing nodes**. In the example, the label is `compression: enable` .

2.5 Click **OK**.

Validation of Effectiveness

After enabling memory compression on the node, you can use the following command to obtain the node's YAML configuration and confirm whether memory compression is correctly enabled through the node's annotation. The following is an example:

```
kubectl get node <nodename> -o yaml | grep "gocrane.io/memory-compression"
```

After logging into the node, check zram, swap, and kernel parameters in turn to confirm that memory compression is correctly enabled. The following is an example:

```
# Confirm zram device initialization.
# zramctl
NAME       ALGORITHM DISKSIZE DATA COMPR TOTAL STREAMS MOUNTPOINT
/dev/zram0 lzo-rle       3.6G   4K   74B   12K       2 [SWAP]

# Confirm settings for swap.
# free -h
              total        used        free      shared  buff/cache   available
Mem:          3.6Gi       441Mi       134Mi       5.0Mi       3.0Gi       2.9Gi
Swap:         3.6Gi       0.0Ki       3.6Gi
```

```
# sysctl vm.force_swappiness
vm.force_swappiness = 1
```

Selecting Services to Enable Memory Compression

Deploying the PodQOS Object

1. **Deploy the PodQOS object.** Enable memory compression on specific pods using `spec.labelSelector.matchLabels` , as shown in the following example:

```
apiVersion: ensurance.crane.io/v1alpha1
kind: PodQOS
metadata:
  name: memorycompression
spec:
  labelSelector:
    matchLabels:
      compression: enable
  resourceQOS:
    memoryQOS:
      memoryCompression:
        compressionLevel: 1
        enable: true
```

Note:

compressionLevel represents the compression level. The value ranges from 1 to 4, corresponding to the algorithms lz4, lzo-rle, lz4hc, zstd, in order of decreasing compression ratio and increasing performance loss.

2. **Create a workload matching the labelSelector in PodQOS**, as shown in the following example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: memory-stress
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: memory-stress
  template:
    metadata:
      labels:
        app: memory-stress
        compression: enable
    spec:
      containers:
        - command:
            - bash
            - -c
            - "apt update && apt install -yq stress && stress --vm-keep --vm 2 --vm
          image: ccr.ccs.tencentyun.com/ccs-dev/ubuntu-base:20.04
          name: memory-stress
          resources:
            limits:
              cpu: 500m
              memory: 1Gi
            requests:
              cpu: 100m
              memory: 100M
      restartPolicy: Always
```

Note:

All containers in a pod must have a memory limit.

Validation of Effectiveness

Verify the memory compression feature through pod annotation (gocrane.io/memory-compression), process memory usage, zram or swap usage, and cgroup memory usage, ensuring that memory compression has been correctly enabled for the pod.

```
# QosAgent will set an annotation for the pod with memory compression enabled.
kubectl get pods -l app=memory-stress -o jsonpath="{.items[0].metadata.annotations.

# zramctl
NAME       ALGORITHM DISKSIZE  DATA  COMPR TOTAL STREAMS MOUNTPOINT
/dev/zram0 lzo-rle       3.6G  163M 913.9K  1.5M       2 [SWAP]

# free -h
              total        used        free      shared  buff/cache   available
Mem:          3.6Gi       1.4Gi       562Mi       5.0Mi       1.7Gi       1.9Gi
Swap:         3.6Gi       163Mi       3.4Gi
```

```
#Check memory.zram.{raw_in_bytes,usage_in_bytes} in cgroup (usually in /sys/fs/cgro
cat memory.zram.{raw_in_bytes,usage_in_bytes}
170659840
934001

#Calculate the difference to obtain the size of memory saved. In the example, 170Mi
cat memory.zram.{raw_in_bytes,usage_in_bytes} | awk 'NR==1{raw=$1} NR==2{compressed
170659840
```

Compression Monitoring

Last updated：2024-06-14 16:28:43

QosAgent provides a series of metrics on port 8084 for monitoring node and pod memory compression, as well as memory and CPU pressure. Users can configure Prometheus and Grafana for monitoring. In addition, we also offer a Grafana monitoring dashboard template for businesses to quickly check the efficiency of memory compression. (Please submit a ticket to obtain the template.)

Key Metrics Introduction

Object	Metric Name	Meaning
Pod	pod_pressure_total	Pod-level PSI, which displays the waiting duration for each pod due to the lack of resources such as CPU, memory, and IO.
	pod_memory_info	Memory status of a pod, including statistics on the following memory metrics for pods and containers: RSS, anonymous memory, file pages, active memory, and inactive memory.
	pod_memory_page_fault_info	Page fault situation of a pod (including file page faults, anonymous page faults, major page faults, and minor page faults).
	pod_memory_oom_kill	Out of Memory (OOM) statistics of a pod.
Node	node_pressure_total	Node CPU, IO, and memory PSI metrics (indicating whether certain resources are constrained).
	node_memory_page_fault_distance	Refault frequency, indicating the situation where "hot" pages are swapped out.
	node_memory_page_fault_major	Number of page faults caused by disk reads.
	node_disk_io_time_seconds_total	Node disk IO total time (via zram0 device metrics, you can observe the situations of swapping out and swapping in to zram0).
	node_disk_read_bytes_total	Disk and zram0 device IO read bandwidth.
	node_disk_reads_completed_total	Disk and zram0 device IO read count

		(indirectly indicating the situation of anonymous memory page faults caused by memory compression).
	node_disk_writes_completed(time_seconds)_total	Disk/zram0 device write operations and total time consumption.
	node_memory_oom_kill	Disk and zram0 device write status.

Which services can be compressed

Supports collecting the ratio of "cold" pages in services as an estimated compressible value. "Cold" pages include cold anonymous pages and cold file pages. Based on the estimated compression value and business attributes, it can be determined which services can be compressed, as well as the estimated amount of compression.

Workingset Saved: kubelet-perspective observation of "Inactive anon" savings value.

Memory Saved: monitoring-perspective observation of "Inactive anon + Inactive File" savings value.

Save memory amount

Memory size saved per pod = Size before zram compression - Size after zram compression

Is memory reclamation accurate?

Observe "node_memory_page_fault_major" and PSI metrics. Low values for node_memory_page_fault_major and Memory PSI metrics indicate accurate reclamation.

Is the business stable?

Monitor the changes of the number of OOM occurrences, PSI, and Zram0 device IO for a pod/node (e.g., "node_disk_read_bytes_total", "node_disk_reads_completed_total", and "node_disk_writes_completed(time_seconds)_total"). The increases of the number of OOM occurrences, PSI, and Zram0 device IO all indicate instability.

Integrate with Tencent Cloud Prometheus Monitoring

1. log in to the Prometheus monitoring service console.
2. In the Prometheus instance list, click the newly created **Instance ID/Name**.
3. Enter the Prometheus Management Center and click **Data Acquisition** in the top navigation bar.
4. On the Integrated Container Service page, click **Associate Cluster** to associate the cluster with the Prometheus instance. For details, see Associated Cluster.
5. In the cluster list, click **Data Collection Configuration** on the right side of the cluster, select **Create Custom Monitoring**, and fill in the configuration information:

Monitoring type: Workload monitoring

Namespace: kube-system

Workload type: DaemonSet

Workload: qos-agent

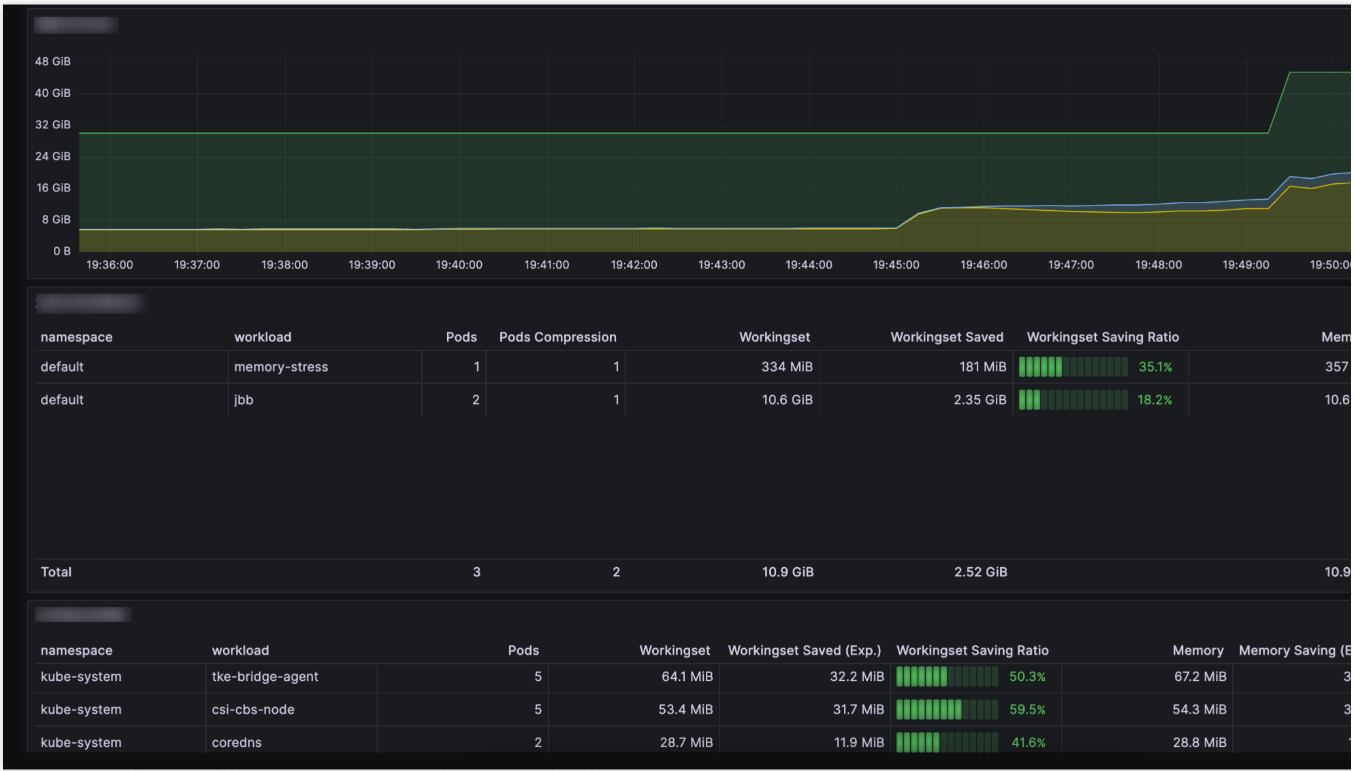
targetPort: 8084

metricsPath: /metrics

6. Click **OK**.

7. Import the following two panels in Grafana:

Cluster Dimension Panel. You can submit a ticket to obtain it, as shown below:



[redacted]

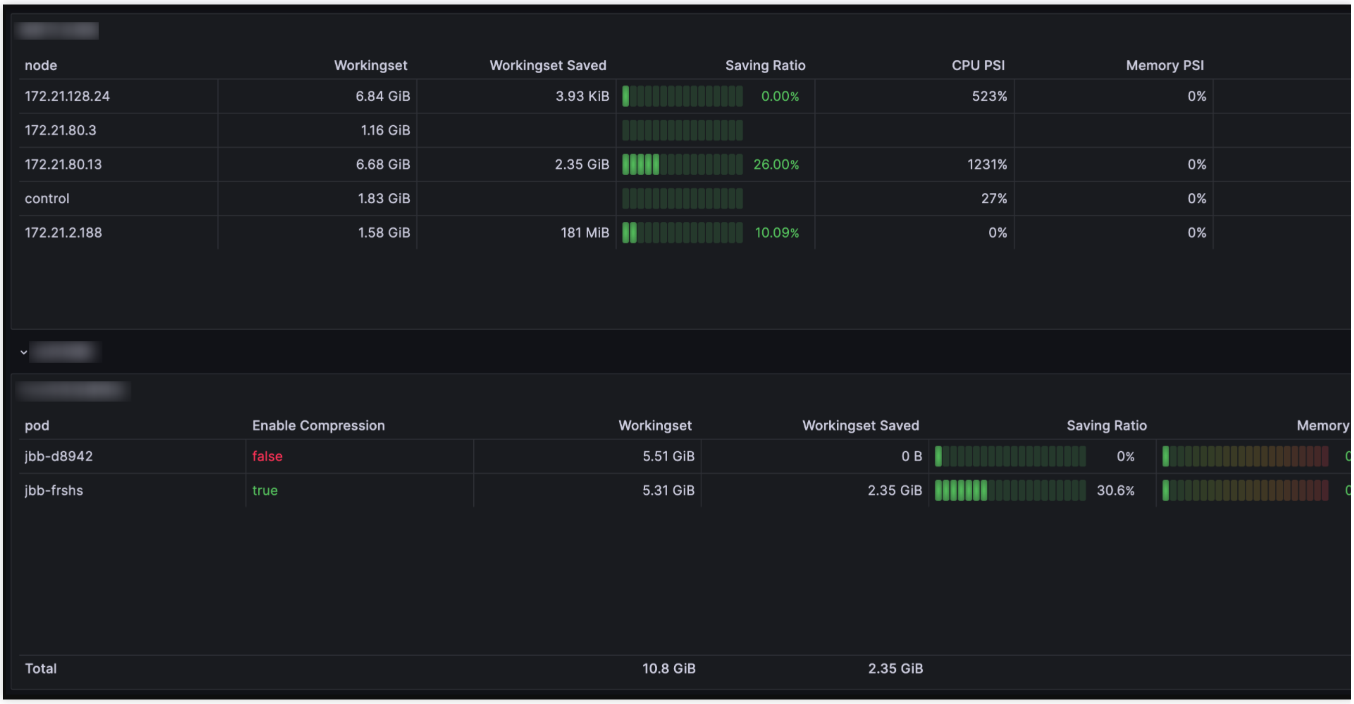
[redacted]

namespace	workload	Pods	Pods Compression	Workingset	Workingset Saved	Workingset Saving Ratio	Mem
default	memory-stress	1	1	334 MiB	181 MiB	35.1%	357
default	jbb	2	1	10.6 GiB	2.35 GiB	18.2%	10.6
Total		3	2	10.9 GiB	2.52 GiB		10.9

[redacted]

namespace	workload	Pods	Workingset	Workingset Saved (Exp.)	Workingset Saving Ratio	Memory	Memory Saving (E
kube-system	tke-bridge-agent	5	64.1 MiB	32.2 MiB	50.3%	67.2 MiB	3
kube-system	csi-cbs-node	5	53.4 MiB	31.7 MiB	59.5%	54.3 MiB	3
kube-system	coredns	2	28.7 MiB	11.9 MiB	41.6%	28.8 MiB	

Node Dimension Panel. You can submit a ticket to obtain it, as shown below:



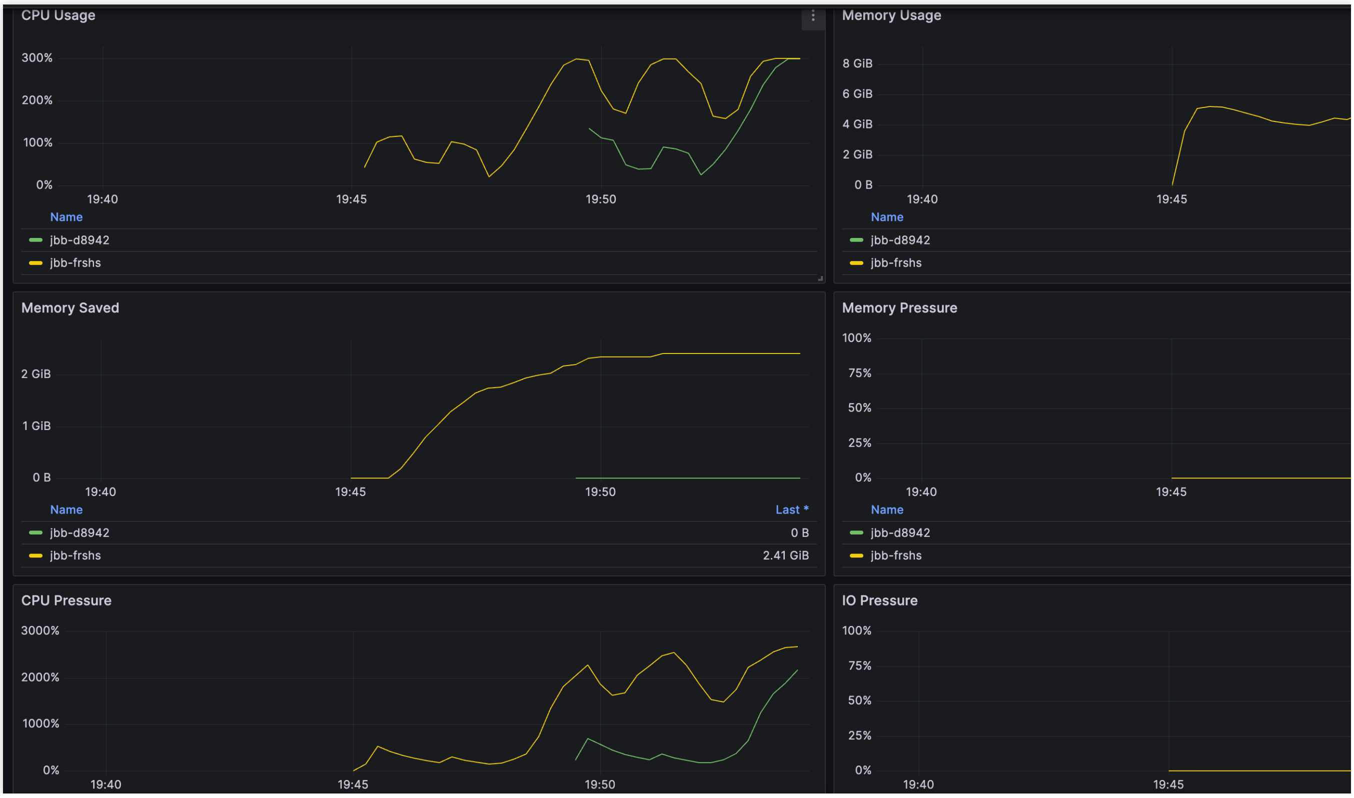
[redacted]

node	Workingset	Workingset Saved	Saving Ratio	CPU PSI	Memory PSI
172.21.128.24	6.84 GiB	3.93 KiB	0.00%	523%	0%
172.21.80.3	1.16 GiB				
172.21.80.13	6.68 GiB	2.35 GiB	26.00%	1231%	0%
control	1.83 GiB			27%	0%
172.21.2.188	1.58 GiB	181 MiB	10.09%	0%	0%

[redacted]

[redacted]

pod	Enable Compression	Workingset	Workingset Saved	Saving Ratio	Memory
jbb-d8942	false	5.51 GiB	0 B	0%	
jbb-frshs	true	5.31 GiB	2.35 GiB	30.6%	
Total		10.8 GiB	2.35 GiB		



GPU Share

qGPU Overview

Last updated：2024-06-27 11:09:15

TKE GPU Virtualization

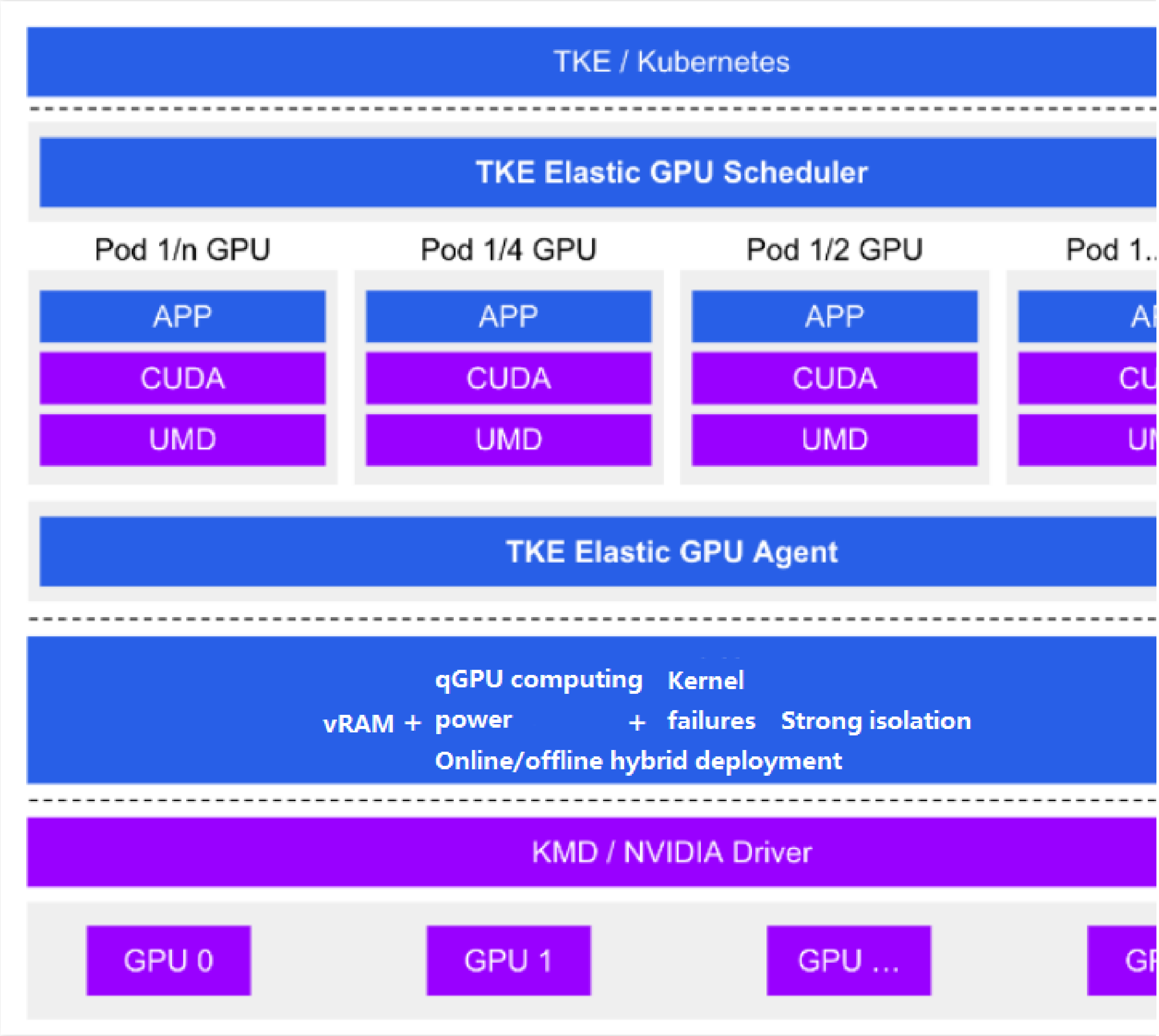
Tencent Kubernetes Engine qGPU (TKE qGPU) is a GPU virtualization service launched by Tencent Cloud. It supports sharing a GPU card among multiple containers and provides the capacity to finely isolate the vRAM and computing power among containers. Also, it provides the unique online/offline hybrid deployment capability to increase GPU utilization and help users significantly save their GPU resource costs on the basis of finely segmenting GPU resources and on the premise of ensuring their business stability.

Based on TKE's open-source Elastic GPU framework, qGPU can schedule GPU computing power and vRAM at a fine granularity, share a GPU card among multiple containers, and allocate GPU resources across GPU cards. Plus, relying on the powerful underlying isolation technology, it can strongly isolate vRAM and computing power to ensure that business performance and resource use are not affected by GPU sharing.

Note:

The qGPU feature is available only for TKE native nodes. No guarantee of effective service is provided for other node types.

Solution Framework Diagram



qGPU Strengths

Flexibility: Finely configure GPU computing capacity and vRAM size.

Strong isolation: vRAM and computing power can be strictly isolated.

Hybrid deployment: Supports online and offline hybrid deployment to maximize the GPU utilization.

Coverage: Supports popular architectures including Volta (such as V100), Turing (such as T4) and Ampere (such as A100 and A10).

Cloud nativeness: Standard Kubernetes and NVIDIA Docker solutions are supported.

Compatibility: No need to rewrite application code or replace CUDA libraries, and it is easily deployed in a way imperceptible to the application.

High performance: Virtualization is applied at the underlying layer of GPU devices, realizing efficient convergence and nearly zero loss of throughput.

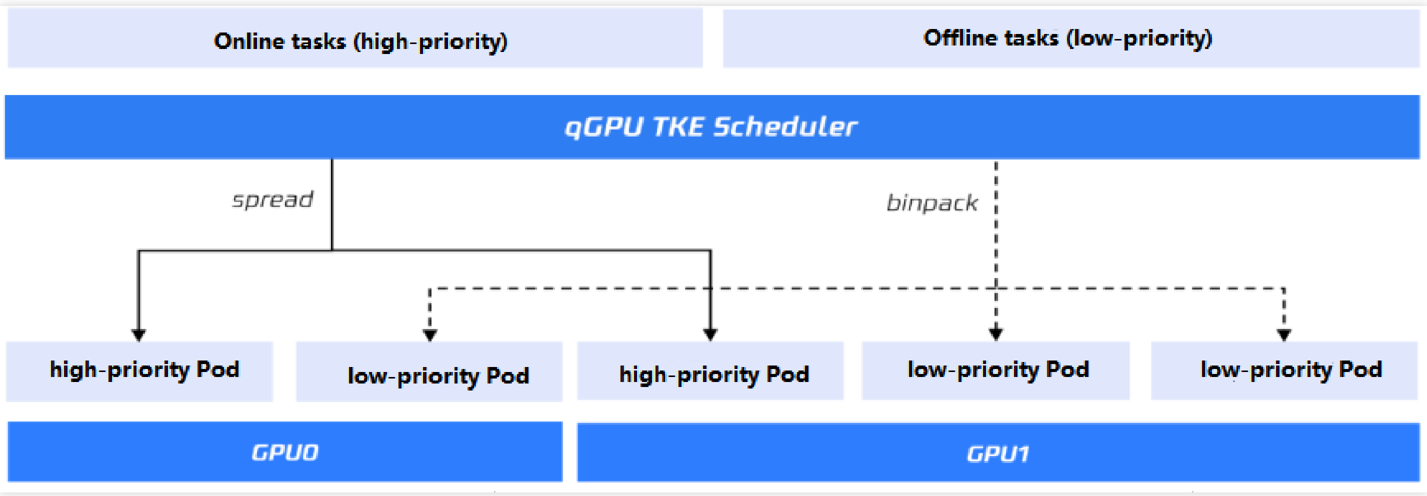
qGPU Online/Offline Hybrid Deployment Description

Last updated：2022-12-08 17:25:19

Feature Overview

Generally, qGPU Pods fairly use physical GPU resources, and a qGPU kernel driver allocates equivalent GPU time slices to each task. As different GPU computing tasks have different running characteristics and importance levels, they have different requirements for GPU resources and use the resources differently. For example, real-time inference is sensitive to GPU resources and latency. It needs to get GPU resources as quickly as possible for computing, but the utilization is usually low. Model training requires a large amount of GPU resources but is insensitive to latency, which means it can endure a certain period of suppression.

Again this backdrop, Tencent Cloud has launched qGPU online/offline hybrid deployment, an innovative technology for deploying and scheduling online and offline GPU resources. Specifically, both online (high-priority) and offline (low-priority) tasks are deployed on the same GPU card, guaranteeing 100% utilization of the idle computing power by low-priority tasks and absolute preemption by high-priority tasks at the kernel and driver levels. This technology achieves 100% GPU utilization and minimizes costs.



Strengths

qGPU online/offline hybrid deployment keeps GPU computing power under absolute control and pushes the utilization to the limit:

- 100% utilization of the idle computing power of high-priority tasks: All GPU computing power can be used by low-priority tasks when it is not occupied by high-priority tasks.
- 100% preemption of the computing power of low-priority tasks: Busy high-priority tasks can preempt GPU computing power from low-priority tasks.

Typical Use Cases

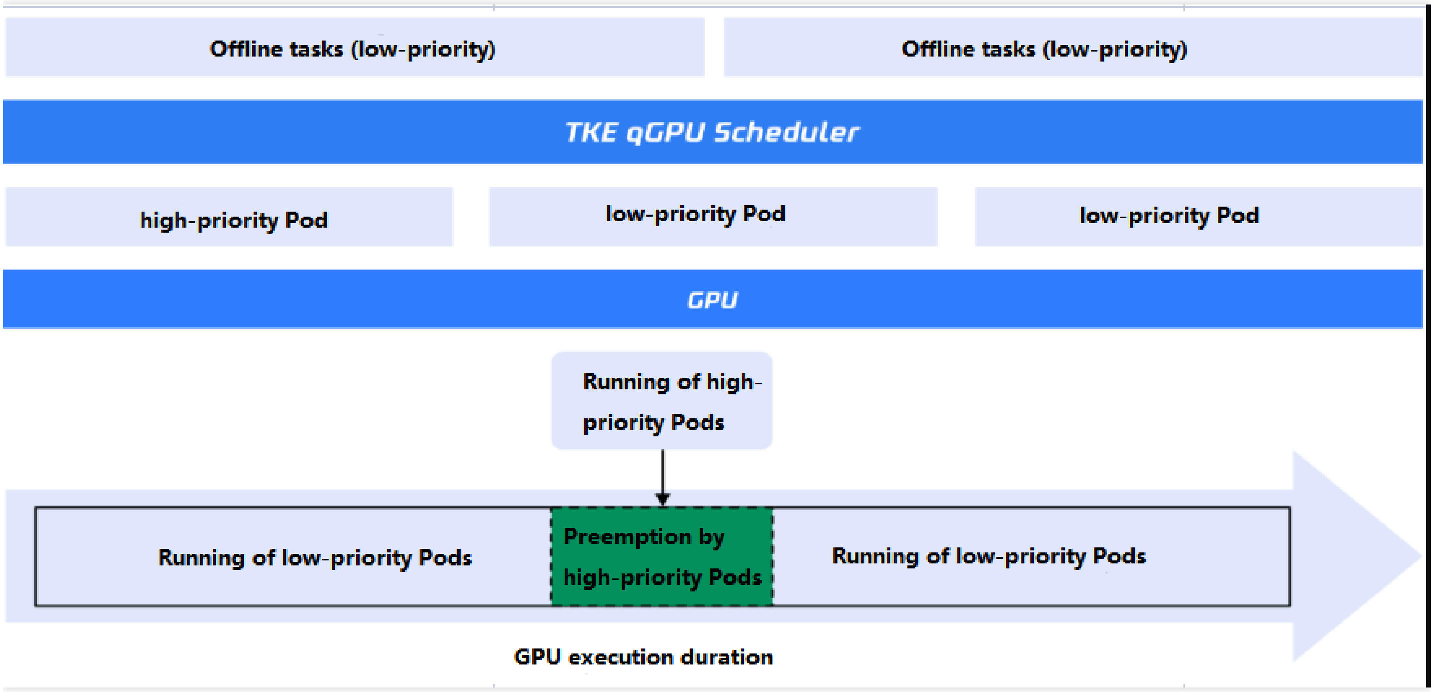
Hybrid deployment of online and offline inference

Search and recommendation support online services and are sensitive to the real-timeness of GPU computing power, while data preprocessing supports offline data cleansing and processing and is insensitive to the real-timeness of GPU computing power. The former can be set as a high-priority task and the latter as a low-priority one for deployment on the same GPU card.

Hybrid deployment of online inference and offline training

Real-time reference is sensitive to the availability of GPU computing power and uses a relatively small amount of resources, while model training consumes a large amount of resources and is insensitive to the availability of GPU computing power. Therefore, the former can be set as a high-priority task and the latter as a low-priority one for deployment on the same GPU card.

How It Works


Offline tasks (low-priority)
Offline tasks (low-priority)
TKE qGPU Scheduler
high-priority Pod
low-priority Pod
low-priority Pod
GPU
Running of high-priority Pods
Running of low-priority Pods
Preemption by high-priority Pods
Running of low-priority Pods
GPU execution duration

qGPU online/offline hybrid deployment can be enabled through the online/offline scheduling policy of the TKE cluster to allow online (high-priority) and offline (low-priority) tasks to share physical GPU resources more efficiently. qGPU online/offline hybrid deployment technology has two features:

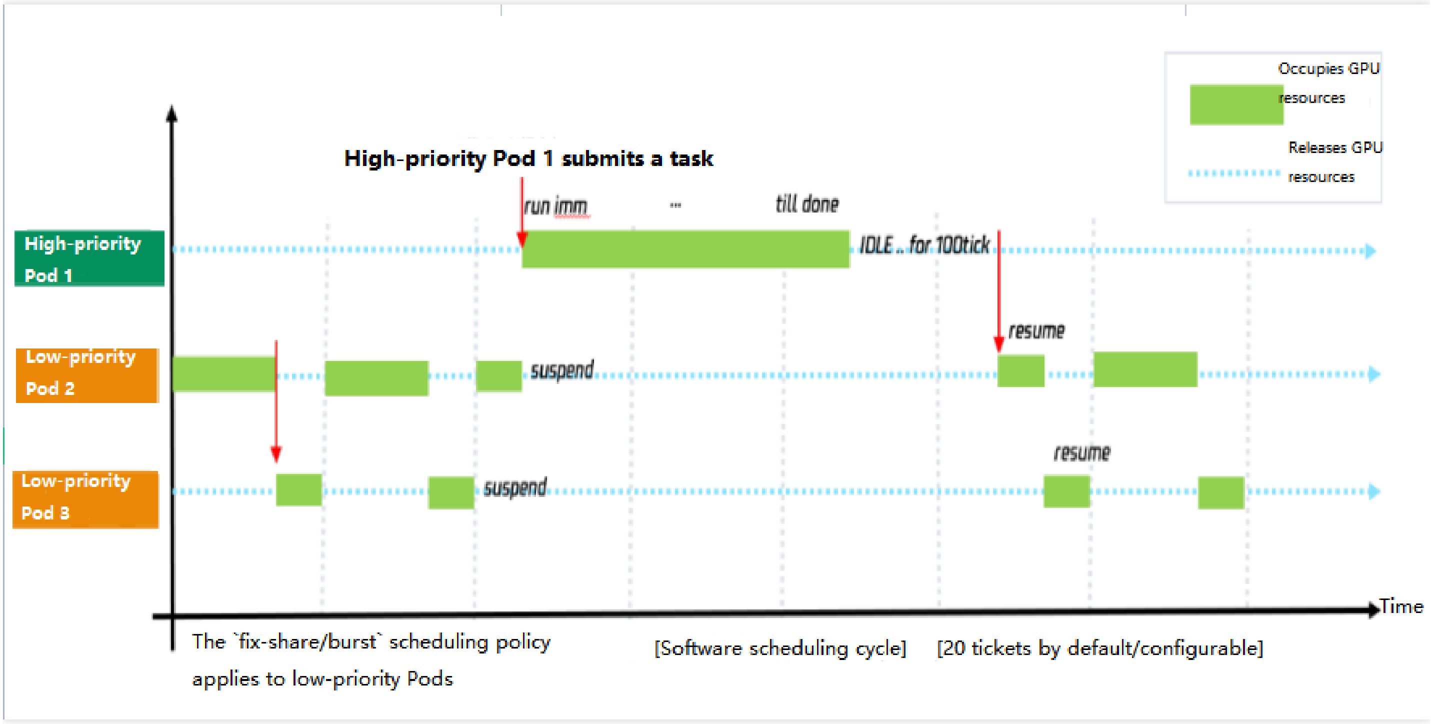
Feature 1: 100% utilization of the idle computing power by low-priority Pods

After low-priority Pods are scheduled to the node GPU, if the GPU computing power is not occupied by high-priority Pods, low-priority Pods can use all the computing power. When multiple low-priority Pods share the GPU computing power, the qGPU policy applies. When there are multiple high-priority Pods, resource competition applies instead of a specific policy.

Feature 2: 100% preemption of the computing power of low-priority Pods

qGPU online/offline hybrid deployment provides a priority-based preemption capability, which ensures that high-priority Pods can immediately and completely use the GPU computing power when they are busy. This is implemented through a priority-based preemption and scheduling policy at the qGPU driver layer:
First, the qGPU driver perceives the requirements of high-priority Pods for GPU computing power and provides all computing power to them within one millisecond after they submit computing tasks. When high-priority Pods have no running tasks, the driver will release the occupied computing power after 100 milliseconds and allocate it to offline Pods.
Second, the qGPU driver supports suspending and resuming computing tasks. When a high-priority Pod has a running computing task, the low-priority Pod that occupies GPU computing power will be suspended immediately to release the computing power. When the task of the high-priority Pod ends, the low-priority Pod will be woken up to resume the computing task from where it ends. The sequence diagram of computing tasks at different priorities is as

shown below:



Scheduling policy

On a general qGPU node, you can set the policy for scheduling Pods on the same card. In the online/offline hybrid deployment feature, the policy affects only the scheduling of low-priority Pods.

- Low-priority Pods
  When high-priority Pods are sleeping and low-priority Pods are running, low-priority Pods are scheduled based on the policy. When high-priority Pods use GPU computing power, all low-priority ones will be suspended immediately until the high-priority task ends, after which low-priority tasks resume based on the policy.
- High-priority Pods
  When high-priority Pods have computing tasks, they preempt the GPU computing power immediately. High-priority Pods always preempt resources from low-priority Pods, and high-priority Pods compete for GPU computing power with each other, both of which are not subject to a specific policy.

Using qGPU Online/Offline Hybrid Deployment

Last updated：2022-12-08 17:25:19

This document describes how to use qGPU online/offline hybrid deployment.

Step 1. Deploy add-ons

You need to deploy nano-gpu-scheduler and nano-gpu-agent.

Deploying nano-gpu-scheduler

nano-gpu-scheduler involves `ClusterRole` and `ClusterRoleBinding` as well as `Deployment` and `Service` . Deploy it by using the following YAML.

Below is the scheduling policy:

- By default, online Pods are preferentially scheduled to GPU cards without offline Pods according to the spread algorithm.
- By default, offline Pods are preferentially scheduled to GPU cards without online Pods according to the bin packing algorithm.

```
kind: Deployment
apiVersion: apps/v1
metadata:
name: qgpu-scheduler
namespace: kube-system
spec:
replicas: 1
selector:
matchLabels:
app: qgpu-scheduler
template:
metadata:
labels:
app: qgpu-scheduler
annotations:
scheduler.alpha.kubernetes.io/critical-pod: ''
spec:
hostNetwork: true
tolerations:
- effect: NoSchedule
operator: Exists
```

```
key: node-role.kubernetes.io/master
serviceAccount: qgpu-scheduler
containers:
- name: qgpu-scheduler
image: ccr.ccs.tencentyun.com/lionelxchen/mixed-scheduler:v61
command: ["qgpu-scheduler", "--priority=binpack"]
env:
- name: PORT
value: "12345"
resources:
limits:
memory: "800Mi"
cpu: "1"
requests:
memory: "800Mi"
cpu: "1"
---
apiVersion: v1
kind: Service
metadata:
name: qgpu-scheduler
namespace: kube-system
labels:
app: qgpu-scheduler
spec:
ports:
- port: 12345
name: http
targetPort: 12345
selector:
app: qgpu-scheduler
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qgpu-scheduler
rules:
- apiGroups:
- ""
resources:
- nodes
verbs:
- get
- list
- watch
- apiGroups:
- ""
```

```
resources:
- events
verbs:
- create
- patch
- apiGroups:
- ""
resources:
- pods
verbs:
- update
- patch
- get
- list
- watch
- apiGroups:
- ""
resources:
- bindings
- pods/binding
verbs:
- create
- apiGroups:
- ""
resources:
- configmaps
verbs:
- get
- list
- watch
---
apiVersion: v1
kind: ServiceAccount
metadata:
name: qgpu-scheduler
namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qgpu-scheduler
namespace: kube-system
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
name: qgpu-scheduler
subjects:
```

```
- kind: ServiceAccount
name: qgpu-scheduler
namespace: kube-system`
```

Deploying nano-gpu-agent

nano-gpu-agent involves `ClusterRole` and `ClusterRoleBinding` as well as `Deployment` and `Service` . Deploy it by using the following YAML.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
name: qgpu-manager
namespace: kube-system
spec:
selector:
matchLabels:
app: qgpu-manager
template:
metadata:
annotations:
scheduler.alpha.kubernetes.io/critical-pod: ""
labels:
app: qgpu-manager
spec:
serviceAccount: qgpu-manager
hostNetwork: true
nodeSelector:
qgpu-device-enable: "enable"
initContainers:
- name: qgpu-installer
image: ccr.ccs.tencentyun.com/lionelxchen/mixed-manager:v27
command: ["/usr/bin/install.sh"]
securityContext:
privileged: true
volumeMounts:
- name: host-root
mountPath: /host
containers:
- image: ccr.ccs.tencentyun.com/lionelxchen/mixed-manager:v27
command: ["/usr/bin/qgpu-manager", "--nodename=$(NODE_NAME)", "--dbfile=/host/va
r/lib/qgpu/meta.db"]
name: qgpu-manager
resources:
limits:
memory: "300Mi"
```

```
cpu: "1"
requests:
memory: "300Mi"
cpu: "1"
env:
- name: KUBECONFIG
value: /etc/kubernetes/kubelet.conf
- name: NODE_NAME
valueFrom:
fieldRef:
fieldPath: spec.nodeName
securityContext:
privileged: true
volumeMounts:
- name: device-plugin
mountPath: /var/lib/kubelet/device-plugins
- name: pod-resources
mountPath: /var/lib/kubelet/pod-resources
- name: host-var
mountPath: /host/var
- name: host-dev
mountPath: /host/dev
volumes:
- name: device-plugin
hostPath:
path: /var/lib/kubelet/device-plugins
- name: pod-resources
hostPath:
path: /var/lib/kubelet/pod-resources
- name: host-var
hostPath:
type: Directory
path: /var
- name: host-dev
hostPath:
type: Directory
path: /dev
- name: host-root
hostPath:
type: Directory
path: /
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qgpu-manager
rules:
```

```
- apiGroups:
- ""
resources:
- "*"
verbs:
- get
- list
- watch
- apiGroups:
- ""
resources:
- events
verbs:
- create
- patch
- apiGroups:
- ""
resources:
- pods
verbs:
- update
- patch
- get
- list
- watch
- apiGroups:
- ""
resources:
- nodes/status
verbs:
- patch
- update
---
apiVersion: v1
kind: ServiceAccount
metadata:
name: qgpu-manager
namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: qgpu-manager
namespace: kube-system
roleRef:
apiGroup: rbac.authorization.k8s.io
kind: ClusterRole
```

```
name: qgpu-manager
subjects:
- kind: ServiceAccount
name: qgpu-manager
namespace: kube-system
```

Step 2. Configure the node label

All qGPU nodes in the cluster will be labeled "qgpu-device-enable=enable". In addition, you need to add the "mixed-qgpu-enable=enable" label to nodes that require online/offline deployment.

Step 3. Configure business attributes

- Offline Pods
- Online Pods
- General Pods

You can use `tke.cloud.tencent.com/app-class: offline` to identify an offline Pod and use `tke.cloud.tencent.com/qgpu-core-greedy` to apply for computing power for it. Note that an offline Pod doesn't support multiple cards, and the computing power applied for must be no more than 100 cores.

```
apiVersion: v1
kind: Pod
annotations:
tke.cloud.tencent.com/app-class: offline
spec:
 containers:
 - name: offline-container
   resources:
     requests:
  tke.cloud.tencent.com/qgpu-core-greedy: xx // Offline computing power
   tke.cloud.tencent.com/qgpu-memory: xx
```

Using qGPU

Last updated：2024-02-28 18:02:54

Notes

Supported Kubernetes Versions	TKE version ≥ v1.14.x
Supported Node Types	Support only native nodes. Native nodes, equipped with FinOps concepts and paired with qGPU, can substantially improve the utilization of GPU/CPU resources.
Supported GPU Card Architectures	Volta (e.g., V100), Turing (e.g., T4), and Ampere (e.g., A100, A10) are supported.
Supported Driver Versions	The nvidia driver minor version (end version number, e.g., 450.102.04, where the minor version corresponds to 04) needs to satisfy the following conditions: 450: <= 450.102.04 470: <= 470.161.03 515: <= 515.65.01 525: <= 525.89.02
Shared Granularity	Each qGPU is assigned a minimum of 1G of vRAM, with a precision unit of 1G. Computing capacity is allocated at a minimum of 5 (representing 5% of one card), up to 100 (a whole card), with a precision unit of 5 (i.e., 5, 10, 15, 20...100).
Complete Card Allocation	Nodes with qGPU capability enabled can allocate whole cards in the manner of tke.cloud.tencent.com/qgpu-core: 100 | 200 | ... (N * 100, N is the number of whole cards). It is recommended to differentiate the NVIDIA allocation method or convert to qGPU usage through TKE's node pool capability.
Quantity Limits	Up to 16 qGPU devices can be created on one GPU. It is recommended to determine the number of qGPU shareable per GPU card based on the size of the vRAM requested by the container deployment.

Note:

If you have upgraded the Kubernetes Master version of your TKE cluster, please pay attention to the following points:

For a managed cluster, you do not need to reconsider configuring this plugin.

In case of a self-deployed cluster (with a self-maintained Master), an upgrade to the Master version might reset the configuration of all components on the Master. This could potentially affect the configuration of the qGPU-scheduler plugin as a Scheduler Extender. Therefore, it is required to uninstall the qGPU plugin first, and then reinstall it.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Requested Resource	Namespace
qgpu-manager	DaemonSet	Each GPU node with one Memory: 300 M, CPU: 0.2	kube-system
qgpu-manager	ClusterRole	-	-
qgpu-manager	ServiceAccount	-	kube-system
qgpu-manager	ClusterRoleBinding	-	kube-system
qgpu-scheduler	Deployment	A single replica Memory: 800 M, CPU: 1	kube-system
qgpu-scheduler	ClusterRole	-	-
qgpu-scheduler	ClusterRoleBinding	-	kube-system
qgpu-scheduler	ServiceAccount	-	kube-system
qgpu-scheduler	Service	-	kube-system

qGPU 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.
It is required to install qgpu ko kernel files and create, manage, and delete qgpu device, thus the initiation of the privileged level container is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission

Track the status changes of a pod, access pod information, and clean up resources such as qgpu devices when a pod is deleted.	pods	get/list/watch
Monitor the status changes of a node, access node information, and add labels to nodes based on gpu card driver and version information as well as qgpu version information.	nodes	get/list/watch/update
The qgpu-scheduler is an extender-based scheduler specifically developed for qgpu resources, based on the Kubernetes scheduler extender mechanism. **The permissions it requires align with those of other community scheduler components** (such as Volcano), including tracking and access to pod information, updating scheduling results to pod labels and annotations, tracking and access to node information, accessing configuration via the configmap, and creating scheduling events.	pods	get/list/update/patch
	nodes	get/list/watch
	configmaps	get/list/watch
	events	create/patch
gpu.elasticgpu.io is qgpu's proprietary CRD resource (**this feature has been deprecated, but to be compatible with earlier versions, the resource definition must be retained**) for recording GPU resource information, managed by qgpu-manager and qgpu-scheduler. It requires permissions for a full range of operations, including creation, deletion, modification, and queries.	gpu.elasticgpu.io and gpu.elasticgpu.io/status	All Permissions

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: qgpu-manager
rules:
  - apiGroups:
      - ""
    resources:
      - pods
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - update
      - get
      - list
      - watch
  - apiGroups:
      - "elasticgpu.io"
    resources:
      - gpus
      - gpus/status
    verbs:
      - '*'

---

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: qgpu-scheduler
rules:
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - create
      - patch
  - apiGroups:
      - ""
    resources:
      - pods
    verbs:
      - update
      - patch
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - bindings
      - pods/binding
    verbs:
      - create
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - "elasticgpu.io"
    resources:
      - gpus
      - gpus/status
    verbs:
      - '*'
```

Direction

Step 1: Install the qGPU scheduling component

1. Log in to 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 **Component Management** from the left menu bar, and click **Create** on the Component Management page.
4. Select the QGPU (GPU Isolation Component) on the **Create Component Management** page.
5. Click **Parameter Configuration**, and set the scheduling policies of qgpu-scheduler.

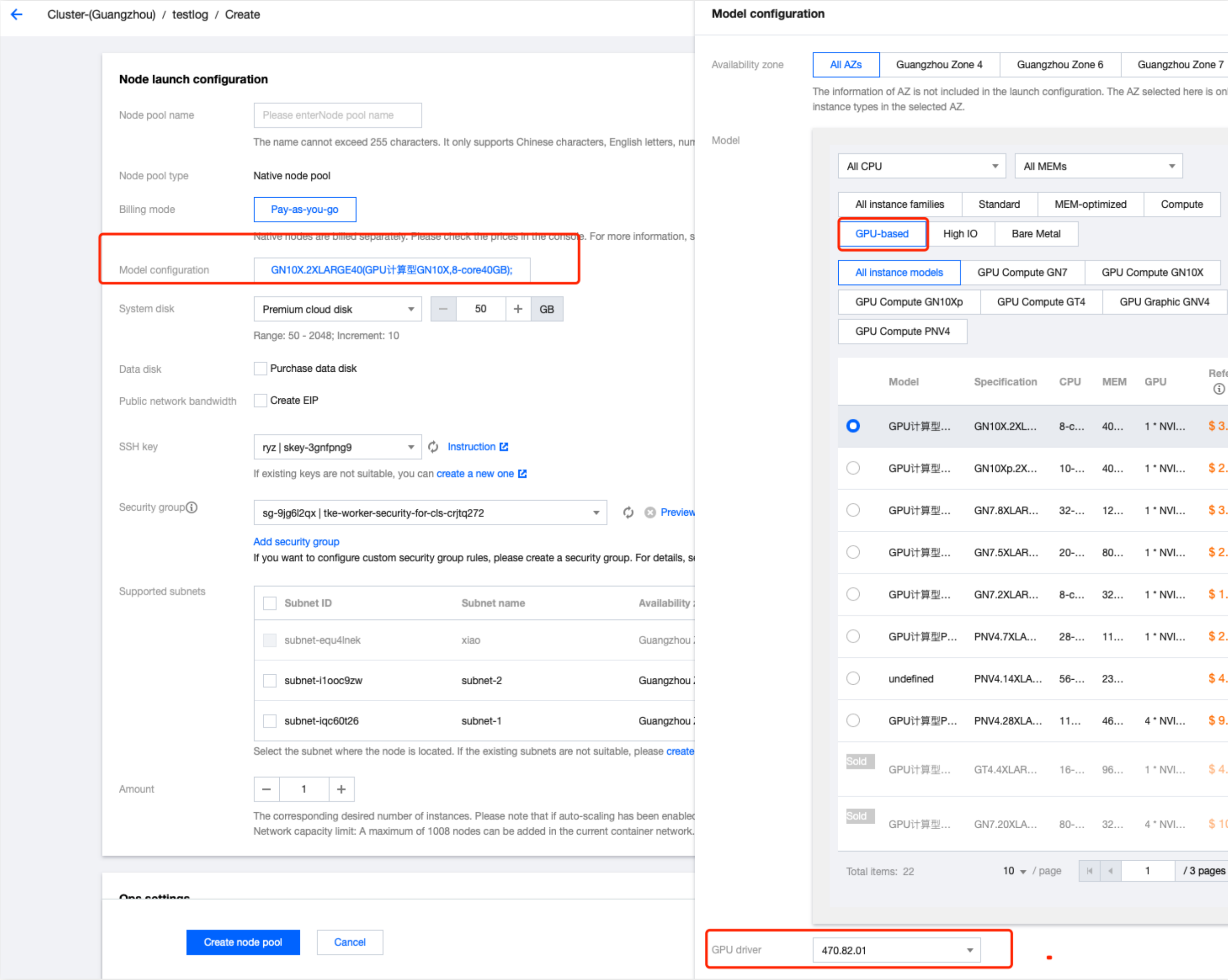
Spread: Multiple Pods would be distributed across different nodes and GPU cards, prioritizing nodes with greater residual resources. It applies to high-availability scenarios to avoid placing replicas of the same application on the same device.

Binpack: Multiple Pods would primarily use the same node, making this suitable for scenarios aiming to increase GPU utilization.

6. Click **Complete** to create the component. Once installed, GPU resources need to be prepared for the cluster.

Step 2: Prepare GPU resources and activate qGPU sharing

1. In the Cluster list, click the desired Cluster ID to access its detailed page.
2. In **Node Management** > **Worker Node**, select the **Node Pool** tab**,** then click **New**.
3. Select **Native Node**, then click **Create**.
4. In the **Create** page, select the corresponding GPU model and choose the driver version supported by qgpu as shown below:



Cluster-(Guangzhou) / testlog / Create

Node launch configuration

Node pool name: Please enterNode pool name
The name cannot exceed 255 characters. It only supports Chinese characters, English letters, num

Node pool type: Native node pool

Billing mode: Pay-as-you-go
Native nodes are billed separately. Please check the prices in the console. For more information, s

Model configuration: GN10X.2XLARGE40(GPU计算型GN10X,8-core40GB);

System disk: Premium cloud disk 50 GB
Range: 50 - 2048; Increment: 10

Data disk: □ Purchase data disk

Public network bandwidth: □ Create EIP

SSH key: ryz | skey-3gnfpng9 Instruction
If existing keys are not suitable, you can create a new one

Security group: sg-9jg6l2qx | tke-worker-security-for-cls-crjtq272 Preview
Add security group
If you want to configure custom security group rules, please create a security group. For details, s

Supported subnets:

□ Subnet ID	Subnet name	Availability
□ subnet-equ4lnek	xiao	Guangzhou
□ subnet-i1ooc9zw	subnet-2	Guangzhou
□ subnet-iqc60t26	subnet-1	Guangzhou

Select the subnet where the node is located. If the existing subnets are not suitable, please create

Amount: 1
The corresponding desired number of instances. Please note that if auto-scaling has been enabled
Network capacity limit: A maximum of 1008 nodes can be added in the current container network.

Ops settings

Create node pool Cancel

Model configuration

Availability zone: All AZs | Guangzhou Zone 4 | Guangzhou Zone 6 | Guangzhou Zone 7
The information of AZ is not included in the launch configuration. The AZ selected here is onl instance types in the selected AZ.

Model: All CPU | All MEMs

All instance families | Standard | MEM-optimized | Compute | GPU-based | High IO | Bare Metal

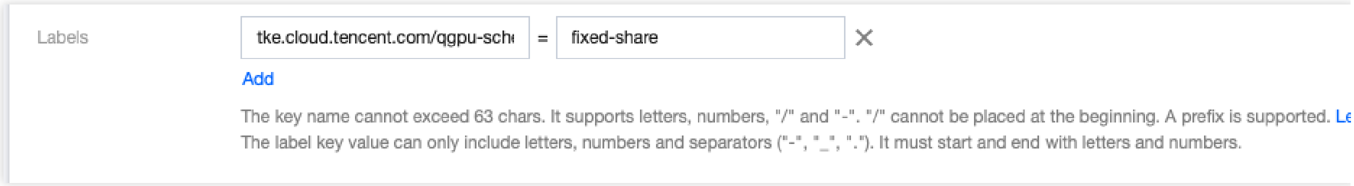
All instance models | GPU Compute GN7 | GPU Compute GN10X | GPU Compute GN10Xp | GPU Compute GT4 | GPU Graphic GNV4 | GPU Compute PNV4

	Model	Specification	CPU	MEM	GPU	Refe
●	GPU计算型...	GN10X.2XL...	8-c...	40...	1 * NVI...	$ 3.
○	GPU计算型...	GN10Xp.2X...	10-...	40...	1 * NVI...	$ 2.
○	GPU计算型...	GN7.8XLAR...	32-...	12...	1 * NVI...	$ 3.
○	GPU计算型...	GN7.5XLAR...	20-...	80...	1 * NVI...	$ 2.
○	GPU计算型...	GN7.2XLAR...	8-c...	32...	1 * NVI...	$ 1.
○	GPU计算型P...	PNV4.7XLA...	28-...	11...	1 * NVI...	$ 2.
○	undefined	PNV4.14XLA...	56-...	23...		$ 4.
○	GPU计算型P...	PNV4.28XLA...	11...	46...	4 * NVI...	$ 9.
Sold	GPU计算型...	GT4.4XLAR...	16-...	96...	1 * NVI...	$ 4.
Sold	GPU计算型...	GN7.20XLA...	80-...	32...	4 * NVI...	$ 10

Total items: 22 10 / page 1 / 3 pages

GPU driver: 470.82.01

5. In **Operation Feature Setting**, click the switch on the right of **qGPU Sharing**. After the switch is enabled, **all new GPU native nodes in the node pool** will enable GPU sharing by default. You can control whether to enable isolation capability through Labels.
6. In **Advanced Settings** > **Labels**, set Labels via the advanced configuration of the node pool, designating qGPU isolation policies:



Labels tke.cloud.tencent.com/qgpu-sche = fixed-share ✕

The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Le
The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Label Key: `tke.cloud.tencent.com/qgpu-schedule-policy`
Label Value: `fixed-share` (The full name or abbreviation of the label value can be provided, more values are available in the table below)

Currently, qGPU supports the following isolation policies:

Label Value	Abbreviation	Name	Meaning
best-effort (Default value)	be	Best Effort	Default value. The computing capacity for each Pod is limitless and can be used as long as there is remaining computing capacity on the card. If a total of N Pods are enabled, each Pod bearing a substantial workload, the result would eventually be a computing capacity of 1/N.
fixed-share	fs	Fixed Share	Each Pod is granted a fixed compute quota that cannot be exceeded, even if the GPU still possesses unused computing capacity.
burst-share	bs	Guaranteed Share with Burst	The scheduler ensures each Pod has a minimum compute quota but as long as the GPU has spare capacity, it may be used by a Pod. For instance, when the GPU has unused capacity (not assigned to other Pods), a Pod can use the computing capacity beyond its quota. Please note that when this portion of the unused capacity is reassigned, the Pod will revert to its computing quota.

7. Click **Create a node pool**.

Step 3: Allocate shared GPU resources to the application

Setting the qGPU corresponding resources to containers can enable the Pod to use qGPU. You can assign GPU resources to your application via the console or YAML.

Note:

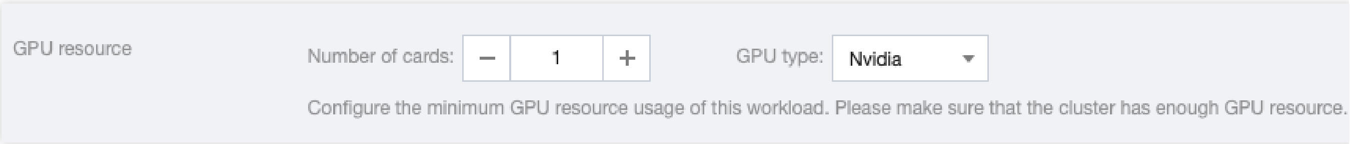
If the application requires to use the whole card resources, just fill in the card quantity; there is no need to fill in the vRAM (it will automatically use all the vRAM on the allocated GPU card).

If the application requires to use the decimal card resources (i.e., sharing the same card with other applications); it requires to fill in both the card quantity and the vRAM simultaneously.

Setting via the Console

Setting via YAML

1. Choose **Workload** in the left navigation bar of the cluster. Click **Create** on any workload object type page. This document takes **Deployment** as an example.
2. On the **Create Deployment** page, select **Instance Container** and fill in the GPU-related resources as shown below:



GPU resource

Number of cards: − 1 +

GPU type: Nvidia

Configure the minimum GPU resource usage of this workload. Please make sure that the cluster has enough GPU resource.

Setting related qGPU resources through YAML:

```
spec:
  containers:
    resources:
      limits:
        tke.cloud.tencent.com/qgpu-memory: "5"
        tke.cloud.tencent.com/qgpu-core: "30"
      requests:
        tke.cloud.tencent.com/qgpu-memory: "5"
        tke.cloud.tencent.com/qgpu-core: "30"
```

Where:

The resource values related to qGPU in the requests and limits must be consistent (According to the rules of K8S, the setting for qGPU in the requests can be omitted, in this case, the requests will be automatically set to the same value as the limits).

tke.cloud.tencent.com/qgpu-memory indicates the vRAM (Unit: G) requested by the container, **allocated in integer values, decimal values are not supported**.

tke.cloud.tencent.com/qgpu-core represents the computing capacity applied by the container. Each GPU card can provide 100% computing capacity, **the setting of qgpu-core should be less than 100**. If the setting value exceeds the remaining computing capacity ratio value, then the setting fails. After setting, the container can obtain a CPU card with n% computing capacity.

Kubernetes Object Management Overview

Last updated：2022-12-13 17:10:52

Object Management Instructions

You can operate native Kubernetes objects such as Deployment and DaemonSet in the console.
The Kubernetes objects are persistent entities in clusters and are used to host services running within the clusters. Different Kubernetes objects can represent diffrent entites:

- Running applications
- Resources available to applications
- Policies associated with applications

You can use Kubernetes objects directly in the TKE console or through the Kubernetes APIs such as kubectl.

Types of Objects

Common Kubernetes objects can be divided into the following types:

Object Types		Object Descriptions	Object Management Operations
Workload	Deployment	Use to manage the Pod of which the scheduling rules are specified.	Deployment Management
	StatefulSet	Manage the application workload API object, and the application is stateful.	StatefulSet Management
	DaemonSet	Ensure the Pods are running on all or some of the nodes, such as log collection program.	DaemonSet Management
	Job	One or more Pods are created for one Job, until the end of running.	Job Management
	CronJob	Job tasks that runs on a regular basis.	CronJob Management
Sevice	Service	A Kubernetes object that provides access to Pod. You can	Service

		define different types based on your needs.	Management
	Ingress	A Kubernetes object that manages external access to services in a cluster.	Ingress Management
Configuration	ConfigMap	Use to store configuration information.	ConfigMap Management
	Secret	Use to store sensitive information, such as password and token.	Secret Management
Storage	Volume	Use to store data related to container access.	Storage Management
	Persistent Volumes （PV）	A piece of storage configured in a Kubernetes cluster.	
	Persistent Volumes Claim （PVC）	A statement of storage request. If you consider a PV as a Pod, then PVC is equivalent to workload.	
	StorageClass	Use to describe the type of storage. When a PVC is created, storage of the specified type, i.e., storage template, is created using StorageClass.	

There are dozens of other Kubernetes objects such as Namespace, HPA, and ResourceQuotas. You can use different objects based on your business needs. Available objects vary depending on Kubernetes version. For more information, visit Kubernetes website.

Resource Quota

TKE applies the following resource limits to all **managed clusters** by using ResourceQuota/tke-default-quota. If you need more quota, please submit a ticket to apply.

Cluster Size	Quota	
	Pod	ConfigMap
Number of nodes ≤ 5	4000	3000
5 < Number of nodes ≤ 20	8000	6000
Number of nodes > 20	None	None

Namespace

Last updated：2022-12-13 16:43:33

A Namespace is the object of logical environment division in the same cluster in Kubernetes. You can manage the division of multiple teams or projects using Namespaces. In a Namespace, the name of a Kubernetes object must be unique. You can use ResourceQuotas to allocate available resources and control the access to different Namespace networks.

How to Use

- Use in the TKE console: You can add, delete, change, and query Namespaces in the TKE console.
- Use with kubectl: For more information, see Kubernetes' official documentation.

Setting Usage Quota of a Namespace Resource with a ResourceQuota

You can have multiple ResourceQuota resources under one Namespace, and each ResourceQuota can set usage constraints for each Namespace resource. You can set the following usage constraints for Namespace resources:

- Compute resource quotas, such as CPU and memory.
- Storage resource quotas, such as total storage of requests.
- Kubernetes object count quotas, such as the number of Deployments.

The quota settings supported by ResourceQuota vary slightly by Kubernetes version. For more information, see Kubernetes' official document about ResourceQuota.
Below is an example of ResourceQuota:

```
apiVersion: v1
kind: ResourceQuota
metadata:
name: object-counts
namespace: default
spec:
hard:
configmaps: "10" ## Up to 10 ConfigMaps
replicationcontrollers: "20" ## Up to 20 ReplicationControllers
secrets: "10" ## Up to 10 Secrets
services: "10" ## Up to 10 Services
```

```
services.loadbalancers: "2" ## Up to 2 Services in LoadBalancer mode
cpu: "1000" ## Up to 1,000 CPU resources can be used under this Namespace
memory: 200Gi ## Up to 200 Gi of memory can be used under this Namespace
```

Setting Access Control for a Namespace Network Using a NetworkPolicy

NetworkPolicy is a resource provided by Kubernetes (K8s) to define a Pod-based network isolation policy. You can not only restrict Namespaces, but also control network access among Pods, i.e., controlling whether a group of Pods can communicate with another group or other network endpoints.

For details about how to deploy a NetworkPolicy Controller in a cluster and implement network access control among Namespaces using a NetworkPolicy, see Using NetworkPolicy for Network Access Control.

Workload

Deployment Management

Last updated：2023-02-02 17:05:22

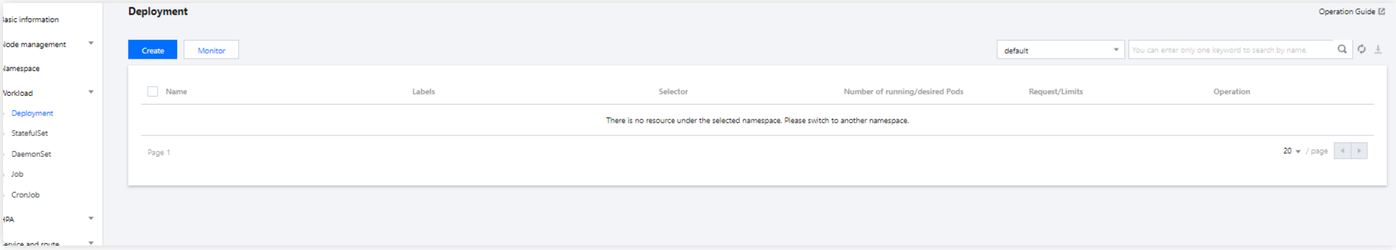
Overview

A Deployment declares the Pod template and Pod running policies. It is used to deploy stateless applications. You can specify the number of replicas, scheduling policy, and update policy for Pods running in the Deployment as needed.

Operation Guide for Deployments in the Console

Creating a Deployment

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster where Deployment needs to be created to enter the cluster management page. See the figure below:



Name	Labels	Selector	Number of running/desired Pods	Request/Limits	Operation

There is no resource under the selected namespace. Please switch to another namespace.

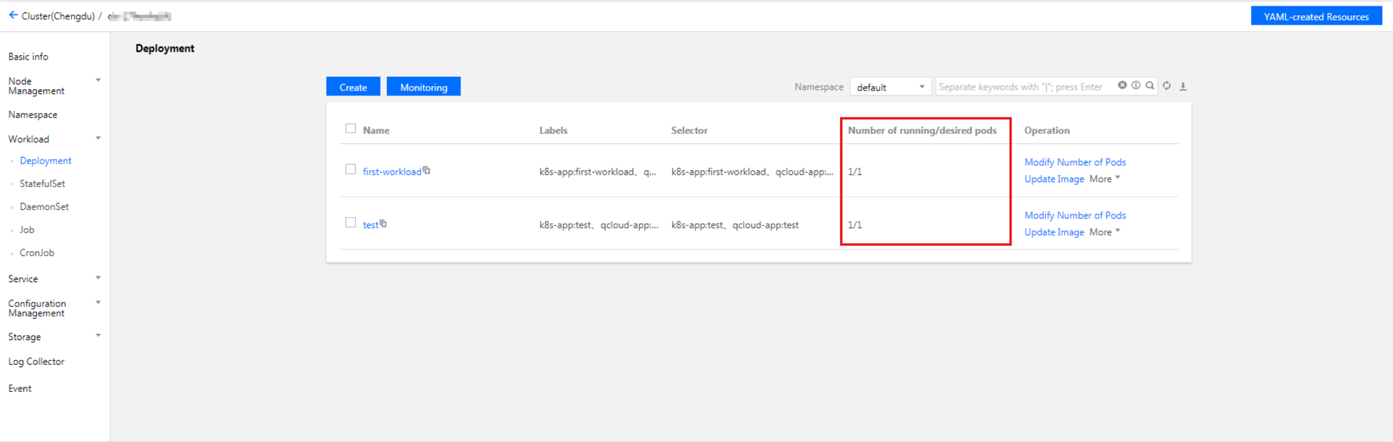
3. Click **Create** to go to the **Create Deployment** page and set deployment parameters as needed. Key parameters are described as follows:

- **Workload**: enter the customized name.
- **Label**: a key-value pair, which is used for classified management of resources. For more information, see Querying Resources by Tag.
- **Namespace**: select a namespace based on your requirements.
- **Volume (optional)**: provides storage for the container. It can be a temp path, CVM path, CBS volume, file storage NFS, configuration file and PVC, and it must be mounted to the specified path of the container.
- **Containers in the Pod**: set one or more different containers for a Pod of the Deployment based on actual needs.
  - **Name**: custom.
  - **Image**: select as needed.

- **Image Tag**: fill as needed.
- **Image Pull Policy**: the following three policies are available. Select as needed.
  If you do not set any image pull policy and **Image Tag** is left empty or set to `latest` , the `Always` policy is used. Otherwise, the `IfNotPresent` policy is used.
  - **Always**: always pull the image from the remote end.
  - **IfNotPresent**: a local image is used by default. If no local image is available, the image is pulled remotely.
  - **Never**: only use a local image. If no local image is available, an exception occurs.
- **Environment Variable**: set the container variables.
- **CPU/Memory Limit**: set the CPU and memory limit according to Kubernetes' resource limits to improve the robustness of the business.
- **GPU Resource**: you can configure the least GPU resource used by the workload.
- **Advanced Settings**: parameters such as "**working directory**", "**run commands**", "**run parameters**", "**container health check**", and "**privilege level**" can be set.

- **Image Access Credential**: a container image is private by default. You need to select the image access credential for the TCR instance when creating a workload.
- **Number of Pods**: select the adjustment method and set the number of Pods based on actual needs.
  - **Manual Adjustment**: set the number of Pods. You can adjust it by clicking **+** or **-**.
  - **Auto Adjustment**: automatically adjust the number of Pods if any of the set conditions are met. For details, see Automatic Scaling Basic Operations.

4. Click **Create Workload** to complete the creation. See the figure below:
   When the running quantity is equal to the expected quantity, all Pods under the Deployment have been created.



☐ Name	Labels	Selector	Number of running/desired pods	Operation
☐ first-workload	k8s-app:first-workload、q...	k8s-app:first-workload、qcloud-app:...	1/1	Modify Number of Pods Update Image More
☐ test	k8s-app:test、qcloud-app:...	k8s-app:test、qcloud-app:test	1/1	Modify Number of Pods Update Image More

Updating a Deployment

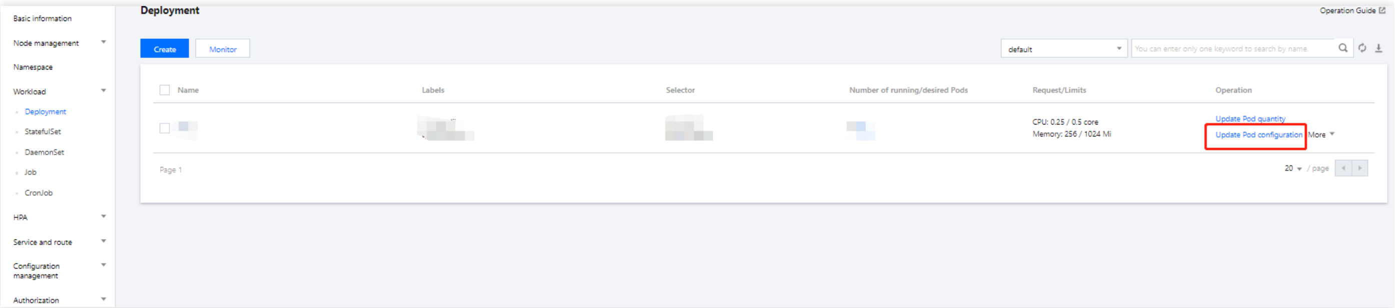
Updating YAML

1. Log in to the TKE console and select **Cluster** in the left sidebar.

2. Click the ID of the cluster for which to update the Deployment to go to the management page of the cluster.
3. In the row of the Deployment for which YAML should be updated, click **More** > **Edit YAML** to go to the Deployment updating page.
4. On the **Update a Deployment** page, edit the YAML and click **Done** to update the YAML.

Updating Pod configuration

1. On the cluster management page, click the ID of the cluster for which Pod configuration should be updated to go to the management page of the cluster.
2. In the Deployment row for which Pod configuration needs to be updated, click **Update Pod Configuration**, as shown below:

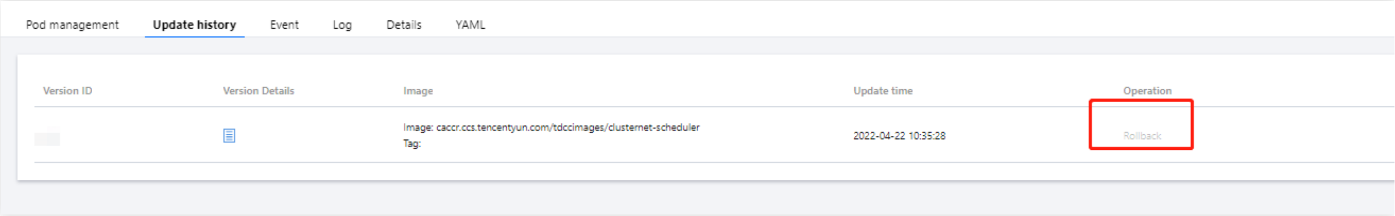


Name	Labels	Selector	Number of running/desired Pods	Request/Limits	Operation
[redacted]	[redacted]	[redacted]	[redacted]	CPU: 0.25 / 0.5 core Memory: 256 / 1024 Mi	Update Pod quantity Update Pod configuration More

3. On the **Update Pod Configuration** page, modify the updating method and set parameters as needed.
4. Click **Update Pod Configuration**.

Rolling back a Deployment

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster for which to roll back the Deployment to go to the management page of the cluster.
3. Click the name of the Deployment to be rolled back to go to the Deployment information page.
4. Select the **Modification History** tab, and click **Rollback** in the row of the version for which rollback is needed, as shown below:



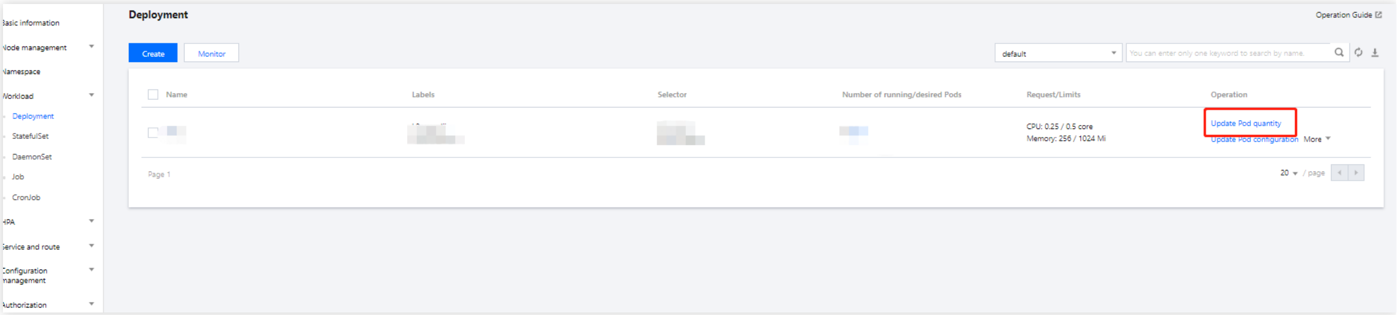
Version ID	Version Details	Image	Update time	Operation
[redacted]		Image: caccr.ccs.tencentyun.com/tdccimages/clusternet-scheduler Tag:	2022-04-22 10:35:28	Rollback

5. Click **OK** in the **Rollback Resources** prompt box to complete the process.

Adjusting Pod quantity

1. Log in to the TKE console and select **Cluster** in the left sidebar.

2. Click the ID of the cluster for which to adjust the Pod quantity to go to the management page of the cluster.
3. In the row of the Deployment for which the Pod quantity should be adjusted, click **Update Pod Quantity** to go to the Pod quantity updating page, as shown below:



Deployment

□ Name	Labels	Selector	Number of running/desired Pods	Request/Limits	Operation
□ [redacted]	[redacted]	[redacted]	[redacted]	CPU: 0.25 / 0.5 core Memory: 256 / 1024 Mi	Update Pod quantity Update Pod configuration More

Page 1

4. Adjust the Pod quantity based on actual needs and click **Update Number of Instance** to complete the adjustment.

Using kubectl to Manipulate Deployments

YAML sample

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
name: nginx-deployment
namespace: default
labels:
app: nginx-deployment
spec:
replicas: 3
selector:
matchLabels:
app: nginx-deployment
template:
metadata:
labels:
app: nginx-deployment
spec:
containers:
- name: nginx
image: nginx:latest
ports:
- containerPort: 80
```

- **kind**: this identifies the Deployment resource type.

- **metadata**: basic information such as Deployment name, Namespace, and Label.
- **metadata.annotations**: an additional description of the Deployment. You can set additional enhancements to TKE through this parameter.
- **spec.replicas**: the number of Pods managed by the Deployment.
- **spec.selector**: the label of the Pod selected by the selector managed by the Deployment.
- **spec.template**: detailed template configuration of the Pod managed by the Deployment.

For more details about the parameters, see Kubernetes' official document about Deployment.

Using kubectl to create a Deployment

1. See the YAML sample to prepare the Deployment YAML file.
2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
3. Run the following command to create the Deployment YAML file.

   ```
   kubectl create -f Deployment YAML filename
   ```

For example, to create a Deployment YAML file named nginx.yaml, run the following command:

```
kubectl create -f nginx.yaml
```

4. Run the following command to check whether the Job is successfully created.

   ```
   kubectl get deployments
   ```

If a message similar to the following is returned, the creation is successful.

```
NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
first-workload 1 1 1 0 6h
ng 1 1 1 1 42m
```

Using kubectl to update a Deployment

You can update the Deployment through Kubectl in three ways. The method 1 and method 2 support both **Recreate** and **RollingUpdate** update policies.

- The Recreate update policy is to first terminate all Pods and then recreate the Deployment.
- The RollingUpdate is the rolling update policy, which is used to update the Pods of the Deployment one by one on a rolling basis.

- Method 1

- Method 2
- Method 3

Run the following command to update a Deployment.

```
kubectl edit deployment/[name]
```

This method applies to simple debugging verification. It is not recommended to use it in production environments. You can update any Deployment parameters in this way.

Using kubectl to rollback a Deployment

1. Run the following command to view the update history of the Deployment.

   ```
   kubectl rollout history deployment/[name]
   ```

2. Run the following command to view the details of the specified version.

   ```
   kubectl rollout history deployment/[name] --revision=[REVISION]
   ```

3. Run the following command to roll back to the earlier version.

   ```
   kubectl rollout undo deployment/[name]
   ```

To specify the rollback version, run the following command.

```
kubectl rollout undo deployment/[name] --to-revision=[REVISION]
```

Using kubectl to adjust Pod quantity

- Manually updating the Pod quantity
- Automatically updating the Pod quantity

Run the following command to manually update the Pod quantity.

```
kubectl scale deployment [NAME] --replicas=[NUMBER]
```

Using kubectl to delete a Deployment

Run the following command to delete a Deployment.

```
kubectl delete deployment [NAME]
```

StatefulSet Management

Last updated：2022-04-22 11:03:12

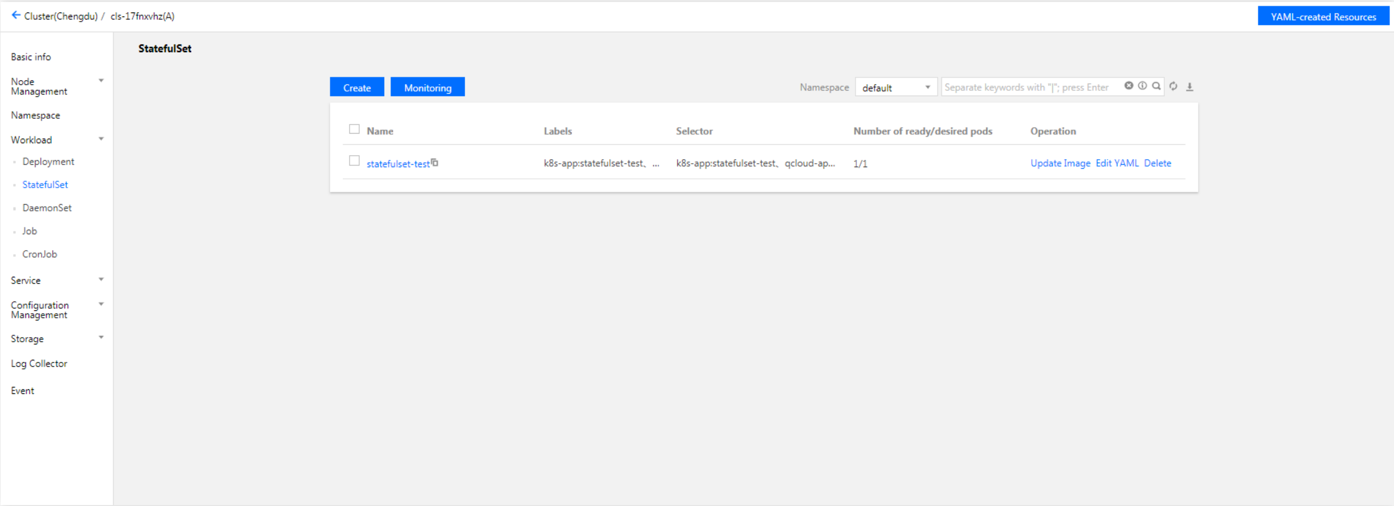
Overview

A StatefulSet is used to manage stateful applications. A Pod created by a StatefulSet has a persistent identifier that is created according to the specifications. The identifier will be retained after the Pod is migrated, terminated, or restarted. When using persistent storage, you can map storage volumes to identifiers. If your application does not require any persistent identifier, we recommend that you use a Deployment to deploy the application.

Operation Guide for StatefulSets in the Console

Creating a StatefulSet

1. Log in to the TKE console and select **Clusters** in the left sidebar.
2. Click the ID of the cluster where StatefulSet needs to be created to enter the cluster management page.
3. Choose **Workload** > **StatefulSet** to go to the StatefulSet management page, as shown below:



Name	Labels	Selector	Number of ready/desired pods	Operation
statefulset-test	k8s-app:statefulset-test、...	k8s-app:statefulset-test、qcloud-ap...	1/1	Update Image Edit YAML Delete

4. Click **Create** to open the **Create Workload** page.

   Set the StatefulSet parameters as needed. Key parameters are as follows:

- **Workload**: enter the customized name.
- **Label**: a key-value pair, which is used for classified management of resources.
- **Namespace**: select a namespace based on your requirements.
- **Type**: Select **StatefulSet (run the Pod in a stateful manner)**.

- **Volume (optional)**: provides storage for the container. It can be a temp path, CVM path, CBS volume, file storage NFS, configuration file and PVC, and it must be mounted to the specified path of the container.
- **Containers in the Pod**: set one or more different containers for a Pod of the StatefulSet as needed.
  - **Name**: custom.
  - **Image**: select as needed.
  - **Image tag**: fill as needed.
  - **Image Pull Policy**: the following three policies are available. Select as needed.
    If you do not set any image pull policy and **Image Tag** is left empty or `latest`, the `Always` policy is used. Otherwise, the `IfNotPresent` policy is used.
    - **Always**: always pull the image from the remote end.
    - **IfNotPresent**: a local image is used by default. If no local image is available, the image is pulled remotely.
    - **Never**: only use a local image. If no local image is available, an exception occurs.
  - **CPU/memory limits**: set the CPU and memory limit according to Kubernetes' resource limits to improve the robustness of the business.
  - **GPU Resource**: you can configure the least GPU resource used by the workload.
  - **Advanced settings**: parameters such as "**working directory**", "**run commands**", "**run parameters**", "**container health check**", and "**privilege level**" can be set.
- **Image Access Credential**: a container image is private by default. You need to select the image access credential for the TCR instance when creating a workload.
- **Number of Pods**: select the adjustment method and set the number of Pods based on actual needs.
- **Node Scheduling Policy**: the Pod can be scheduled to the node of the Label that meets the expectation according to the scheduling rules.
- **Access Settings**: set Service parameters according to actual needs. For more information, see Service Access.

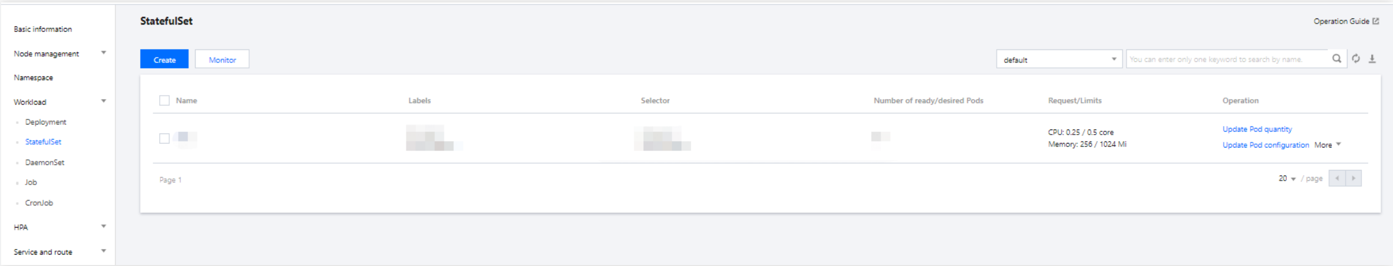
5. Click **Create Workload** to complete the process.

Updating a StatefulSet

Updating YAML

1. Log in to the TKE console and select **Clusters** in the left sidebar.
2. Click the ID of the cluster for which to update the YAML to go to the management page of the cluster.

3. Select **Workload** > **StatefulSet** to go to the StatefulSet information page, as shown below:



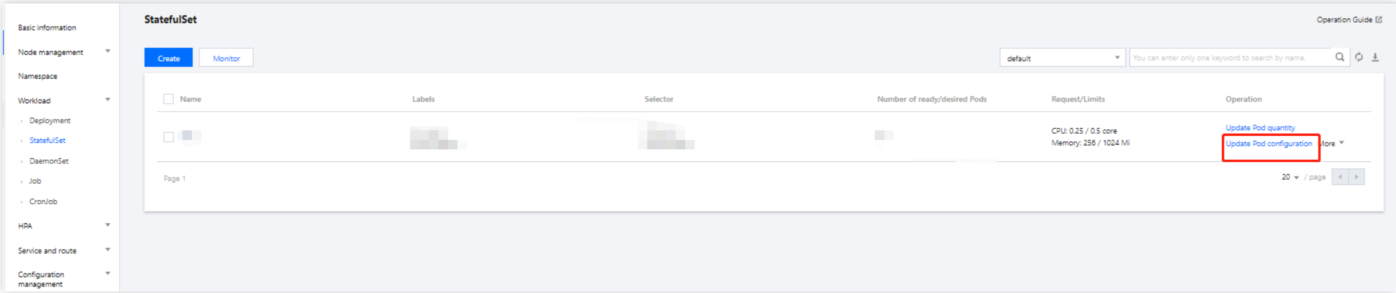
☐ Name	Labels	Selector	Number of ready/desired Pods	Request/Limits	Operation
☐ [redacted]	[redacted]	[redacted]	[redacted]	CPU: 0.25 / 0.5 core Memory: 256 / 1024 Mi	Update Pod quantity Update Pod configuration More

Page 1 20 / page

4. In the row of the StatefulSet for which YAML should be updated, click **More** > **Edit YAML** to go to the StatefulSet updating page.
5. On the **Update a StatefulSet** page, edit the YAML and click **Done** to update the YAML.

Updating Pod configuration

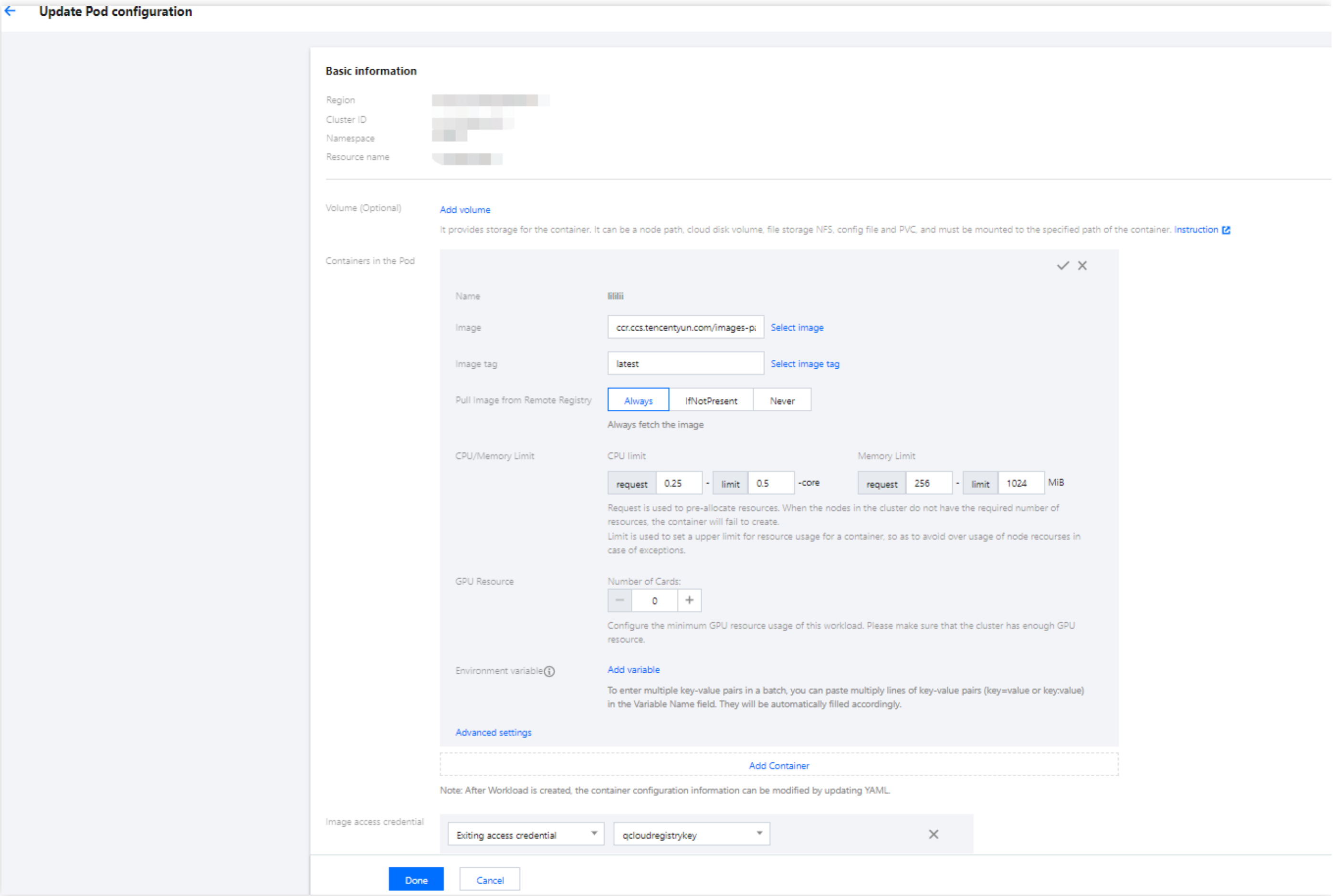
1. On the cluster management page, click the ID of the StatefulSet cluster for which the Pod configuration needs to be updated to enter the StatefulSet cluster management page.
2. In the StatefulSet row for which Pod configuration needs to be updated, click **Update Pod Configuration**, as shown below:



☐ Name	Labels	Selector	Number of ready/desired Pods	Request/Limits	Operation
☐ [redacted]	[redacted]	[redacted]	[redacted]	CPU: 0.25 / 0.5 core Memory: 256 / 1024 Mi	Update Pod quantity Update Pod configuration More

Page 1 20 / page

3. On the **Update Pod Configuration** page, modify the updating method and set parameters as needed, as shown below:



Update Pod configuration

Basic information

Region [redacted]
Cluster ID [redacted]
Namespace [redacted]
Resource name [redacted]

Volume (Optional) Add volume
It provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container. Instruction

Containers in the Pod

Name: lililii
Image: ccr.ccs.tencentyun.com/images-p: Select image
Image tag: latest Select image tag
Pull Image from Remote Registry: Always | IfNotPresent | Never
Always fetch the image
CPU/Memory Limit:
CPU limit: request 0.25 - limit 0.5 -core
Memory Limit: request 256 - limit 1024 MiB
Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail to create.
Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node recourses in case of exceptions.
GPU Resource: Number of Cards: 0
Configure the minimum GPU resource usage of this workload. Please make sure that the cluster has enough GPU resource.
Environment variable: Add variable
To enter multiple key-value pairs in a batch, you can paste multiply lines of key-value pairs (key=value or key:value) in the Variable Name field. They will be automatically filled accordingly.
Advanced settings

Add Container

Note: After Workload is created, the container configuration information can be modified by updating YAML.

Image access credential: Exiting access credential | qcloudregistrykey

Done Cancel

4. Click **Done** to update the Pod configuration.

Using Kubectl to Manipulate StatefulSets

YAML sample

```
apiVersion: v1
kind: Service ## Create a Headless Service to control the network domain
metadata:
name: nginx
namespace: default
labels:
app: nginx
spec:
ports:
- port: 80
name: web
clusterIP: None
selector:
```

```
app: nginx
---
apiVersion: apps/v1
kind: StatefulSet ### Create a Nginx StatefulSet
metadata:
name: web
namespace: default
spec:
selector:
matchLabels:
app: nginx
serviceName: "nginx"
replicas: 3 # by default is 1
template:
metadata:
labels:
app: nginx
spec:
terminationGracePeriodSeconds: 10
containers:
- name: nginx
image: nginx:latest
ports:
- containerPort: 80
name: web
volumeMounts:
- name: www
mountPath: /usr/share/nginx/html
volumeClaimTemplates:
- metadata:
name: www
spec:
accessModes: [ "ReadWriteOnce" ]
storageClassName: "cbs"
resources:
requests:
storage: 10Gi
```

- **kind**: this identifies the StatefulSet resource type.
- **metadata**: basic information such as StatefulSet name and Label.
- **metadata.annotations**: an additional description of the StatefulSet. You can set additional enhancements to TKE through this parameter.
- **spec.template**: detailed template configuration of the Pod managed by the StatefulSet.
- **spec.volumeClaimTemplates**: provides a templates for creating PVCs and PVs.

For more details about the parameters, see Kubernetes' official document about StatefulSet.

Creating a StatefulSet

1. Prepare the StatefulSet YAML file as instructed by YAML sample.
2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
3. Run the following command to create the StatefulSet YAML file.

```
kubectl create -f <StatefulSet YAML filename>
```

For example, to create a StatefulSet YAML file named web.yaml, run the following command:

```
kubectl create -f web.yaml
```

4. Run the following command to check whether the Job is successfully created.

```
kubectl get StatefulSet
```

If a message similar to the following is returned, the creation is successful.

```
NAME DESIRED CURRENT AGE
test 1 1 10s
```

Updating a StatefulSet

Run the following command to view the update policy type of the StatefulSet.

```
kubectl get ds/<daemonset-name> -o go-template='{{.spec.updateStrategy.type}}{{
"\n"}}'
```

StatefulSet has the following two update policy types:
–OnDelete: the default upgrade policy. With this policy, after the StatefulSet is updated, you have to manually delete the old StatefulSet Pod to create a new one.

- RollingUpdate: Kubernetes 1.7 or later is supported. With this policy, after the StatefulSet template is updated, the old StatefulSet Pod will be terminated, and a new StatefulSet Pod will be created in a rolling update manner (only for Kubernetes v1.7 or later).

Method 1

Run the following command to update a StatefulSet.

```
kubectl edit StatefulSet/[name]
```

This method applies to simple debugg verification. It is not recommended to use it in production environments. You can update any StatefulSet parameters in this way.

Method 2

Run the following command to update the image of the specified container.

```
kubectl patch statefulset <NAME> --type='json' -p='[{"op": "replace", "path": "/s
pec/template/spec/containers/0/image", "value":"<newImage>"}]'
```

It is recommended to keep other StatefulSet parameters unchanged and only update the container image when the business is updated.

If the StatefulSet is roll updated, you can view the update status by running the following command:

```
kubectl rollout status sts/<StatefulSet-name>
```

Deleting a StatefulSet

Run the following command to delete a StatefulSet.

```
kubectl delete StatefulSet [NAME] --cascade=false
```

The --cascade=false parameter indicates that Kubernetes only deletes the StatefulSet but not the Pods. Run the following command if you need to delete Pod.

```
kubectl delete StatefulSet [NAME]
```

For more information about StatefulSet operations, see Kubernetes' official guide.

DaemonSet Management

Last updated：2022-04-22 10:54:28

Overview

DaemonSet is used to deploy resident background programs in the cluster. For example, it can collect logs for a node. DaemonSet ensures that specified Pods are running on all or certain nodes. When you add new nodes to a cluster, Pods are deployed automatically. When nodes are removed from the cluster, Pods are recovered automatically.

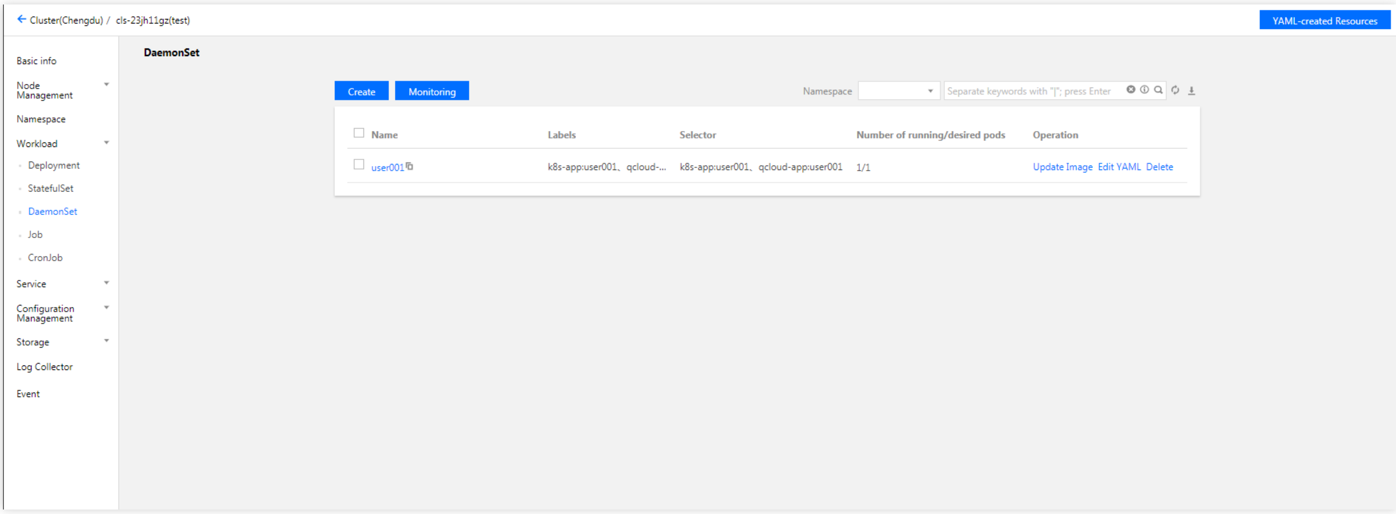
Notes on Scheduling

If nodeSelector or affinity parameters are configured on the Pod, the Pod managed by DaemonSet is scheduled based on the specified rules. If the nodeSelector or affinity parameters are not configured on the Pod, the Pod will be deployed on all nodes.

Operation Guide for DaemonSet in the Console

Creating a DaemonSet

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster where DaemonSet needs to be created to enter the cluster management page.
3. Select **Workload** > **DaemonSet** to go to the DaemonSet information page. See the figure below:



Name	Labels	Selector	Number of running/desired pods	Operation
user001	k8s-app:user001、qcloud-...	k8s-app:user001、qcloud-app:user001	1/1	Update Image Edit YAML Delete

4. Click **Create** to open the **Create Workload** page.
   Set DaemonSet parameters as needed. The key parameters are as follows:

- **Workload**: enter the customized name.
- **Label**: a key-value pair, which is used for classified management of resources.
- **Namespace**: select a namespace based on your requirements.
- **Type**: select **DaemonSet (run the Pod on each server)**.
- **Volume (optional)**: provides storage for the container. It can be a temp path, CVM path, CBS volume, file storage NFS, configuration file and PVC, and it must be mounted to the specified path of the container.
- **Containers in the Pod**: set one or more different containers for a Pod of the DaemonSet based on actual needs.
  - **Name**: custom.
  - **Image**: select as needed.
  - **Image Tag**: fill as needed.
  - **Image Pull Policy**: the following three policies are available. Select as needed.
    If you do not set any image pull policy and **Image Tag** is left empty or `latest` , the `Always` policy is used. Otherwise, the `IfNotPresent` policy is used.
    - **Always**: always pull the image from the remote end.
    - **IfNotPresent**: a local image is used by default. If no local image is available, the image is pulled remotely.
    - **Never**: only use a local image. If no local image is available, an exception occurs.
  - **CPU/Memory Limit**: set the CPU and memory limit according to Kubernetes' resource limits to improve the robustness of the business.
  - **GPU Resource**: you can configure the least GPU resource used by the workload.
  - Advanced Settings: you can set the parameters such as **Working Directory**, **Running Command**, **Running Parameter**, **Container Health Check**, and **Privileged Container**.
- **Image Access Credential**: a container image is private by default. You need to select the image access credential for the TCR instance when creating a workload.
- **Node Scheduling Policy**: the Pod can be scheduled to the node of the Label that meets the expectation according to the scheduling rules.

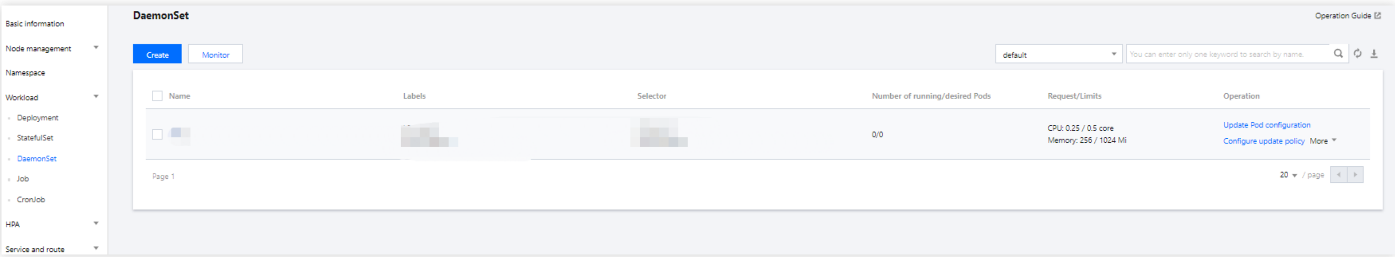
5. Click **Create Workload** to complete the process.

Updating a DaemonSet

Updating YAML

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster for which to update the YAML to go to the management page of the cluster.

3. Select **Workload** > **DaemonSet** to go to the DaemonSet information page. See the figure below:



Name	Labels	Selector	Number of running/desired Pods	Request/Limits	Operation
[redacted]	[redacted]	[redacted]	0/0	CPU: 0.25 / 0.5 core Memory: 256 / 1024 Mi	Update Pod configuration Configure update policy More

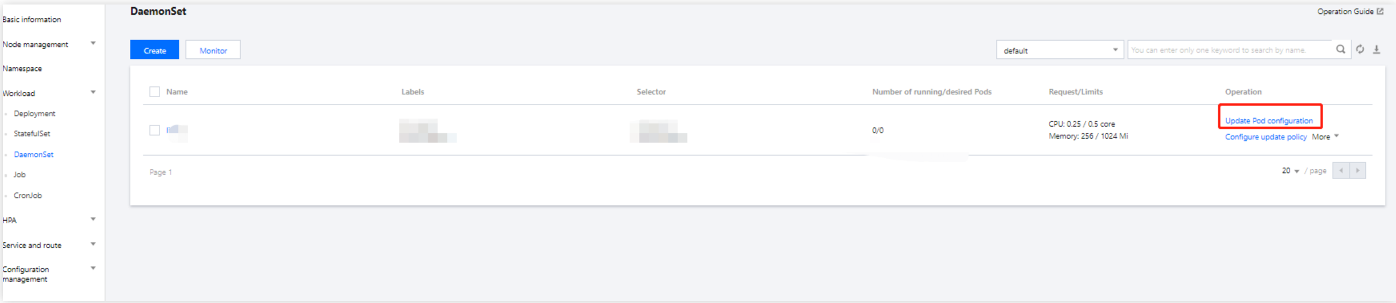
Page 1 20 / page

4. In the row of DaemonSet for which YAML is to be updated, click **More** > **Edit YAML** to go to the DaemonSet updating page.
5. On this page, edit the YAML and click **Done** to update the YAML.

Updating Pod configuration

> Note：
>
> Rolling update of DaemonSet is only supported in Kubernetes v1.6 or higher.

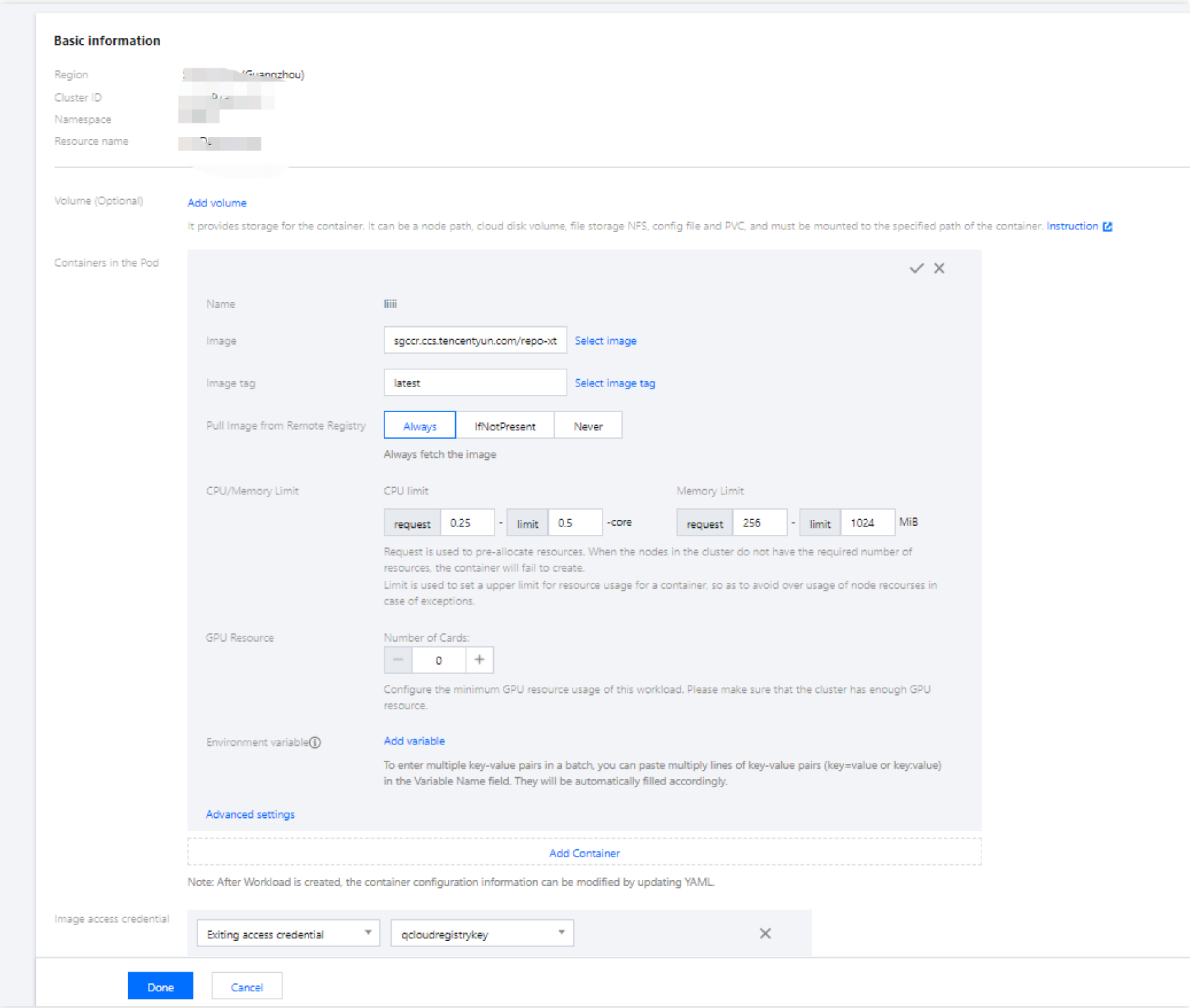
1. On the cluster management page, click the ID of the cluster for which Pod configuration should be updated, and go to the management page of the cluster.
2. In the DaemonSet row for which Pod configuration needs to be updated, click **Update Pod Configuration**, as shown below:



Name	Labels	Selector	Number of running/desired Pods	Request/Limits	Operation
[redacted]	[redacted]	[redacted]	0/0	CPU: 0.25 / 0.5 core Memory: 256 / 1024 Mi	Update Pod configuration Configure update policy More

Page 1 20 / page

3. On the **Update Pod Configuration** page, modify the updating method and set parameters as needed, as shown below:



Basic information

Region [redacted](Guangzhou)

Cluster ID [redacted]

Namespace [redacted]

Resource name [redacted]

Volume (Optional) Add volume

It provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container. Instruction

Containers in the Pod

Name liiii

Image sgccr.ccs.tencentyun.com/repo-xt Select image

Image tag latest Select image tag

Pull Image from Remote Registry [✓ Always] IfNotPresent Never

Always fetch the image

CPU/Memory Limit

CPU limit request 0.25 - limit 0.5 -core

Memory Limit request 256 - limit 1024 MiB

Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail to create.
Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node recourses in case of exceptions.

GPU Resource Number of Cards: 0

Configure the minimum GPU resource usage of this workload. Please make sure that the cluster has enough GPU resource.

Environment variable Add variable

To enter multiple key-value pairs in a batch, you can paste multiply lines of key-value pairs (key=value or key:value) in the Variable Name field. They will be automatically filled accordingly.

Advanced settings

Add Container

Note: After Workload is created, the container configuration information can be modified by updating YAML.

Image access credential Exiting access credential qcloudregistrykey

Done Cancel

4. Click **Done** to update the Pod configuration.

Using kubectl to Manipulate DaemonSet

YAML sample

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
name: fluentd-elasticsearch
namespace: kube-system
labels:
k8s-app: fluentd-logging
```

```
spec:
selector:
matchLabels:
name: fluentd-elasticsearch
template:
metadata:
labels:
name: fluentd-elasticsearch
spec:
tolerations:
- key: node-role.kubernetes.io/master
effect: NoSchedule
containers:
- name: fluentd-elasticsearch
image: k8s.gcr.io/fluentd-elasticsearch:1.20
resources:
limits:
memory: 200Mi
requests:
cpu: 100m
memory: 200Mi
volumeMounts:
- name: varlog
mountPath: /var/log
- name: varlibdockercontainers
mountPath: /var/lib/docker/containers
readOnly: true
terminationGracePeriodSeconds: 30
volumes:
- name: varlog
hostPath:
path: /var/log
- name: varlibdockercontainers
hostPath:
path: /var/lib/docker/containers
```

Note：

Note: The YAML sample described above comes from `https://kubernetes.io/docs/concepts/workloads/controllers/daemonset` . The container image may not be got during creation. The sample is only used to describe the composition of DaemonSet.

- **kind**: this identifies the DaemonSet resource type.
- **metadata**: basic information such as the DaemonSet name and label.

- **metadata.annotations**: an additional description of the DaemonSet. You can set additional enhancements to TKE through this parameter.
- **spec.template**: detailed template configuration of the Pod managed by the DaemonSet.

For more information, see Kubernetes' official document about DaemonSet.

Using kubectl to create a DaemonSet

1. Prepare the StatefulSet YAML file as instructed by YAML sample.
2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
3. Run the following command to create the DaemonSet YAML file.

```
kubectl create -f <DaemonSet YAML filename>
```

For example, to create a StatefulSet YAML file named fluentd-elasticsearch.yaml, run the following command:

```
kubectl create -f fluentd-elasticsearch.yaml
```

4. Run the following command to check whether the Job is successfully created.

```
kubectl get DaemonSet
```

If a message similar to the following is returned, the creation is successful.

```
NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
frontend 0 0 0 0 0 app=frontend-node 16d
```

Using kubectl to update a DaemonSet

Run the following command to view the update policy type of the DaemonSet.

```
kubectl get ds/<daemonset-name> -o go-template='{{.spec.updateStrategy.type}}{{
"\n"}}'
```

DaemonSet has the following two update policy types:

-OnDelete: default update policy. With this policy, after the DaemonSet is updated, you have to manually delete the old DaemonSet Pod to create a new one.

-RollingUpdate: Kubernetes 1.6 or later is supported. With this policy, after the DaemonSet template is updated, the old DaemonSet Pod will be killed, and a new one will be created in a rolling update manner.

Method 1

Run the following command to update a DaemonSet.

```
kubectl edit DaemonSet/[name]
```

This method applies to simple debugg verification. It is not recommended to use it in production environments. You can update any DaemonSet parameters in this way.

Method 2

Run the following command to update the image of the specified container.

```
kubectl set image ds/[daemonset-name][container-name]=[container-new-image]
```

It is recommended to keep other DaemonSet parameters unchanged and only update the container image when the business is updated.

Using kubectl to rollback a DaemonSet

1. Run the following command to view the update history of the DaemonSet.

   ```
   kubectl rollout history daemonset /[name]
   ```

2. Run the following command to view the details of the specified version.

   ```
   kubectl rollout history daemonset /[name] --revision=[REVISION]
   ```

3. Run the following command to roll back to the earlier version.

   ```
   kubectl rollout undo daemonset /[name]
   ```

To specify the rollback version, run the following command.

```
kubectl rollout undo daemonset /[name] --to-revision=[REVISION]
```

Using kubectl to delete a DaemonSet

Run the following command to delete a DaemonSet.

```
kubectl delete DaemonSet [NAME]
```

Job Management

Last updated：2022-04-22 11:07:23

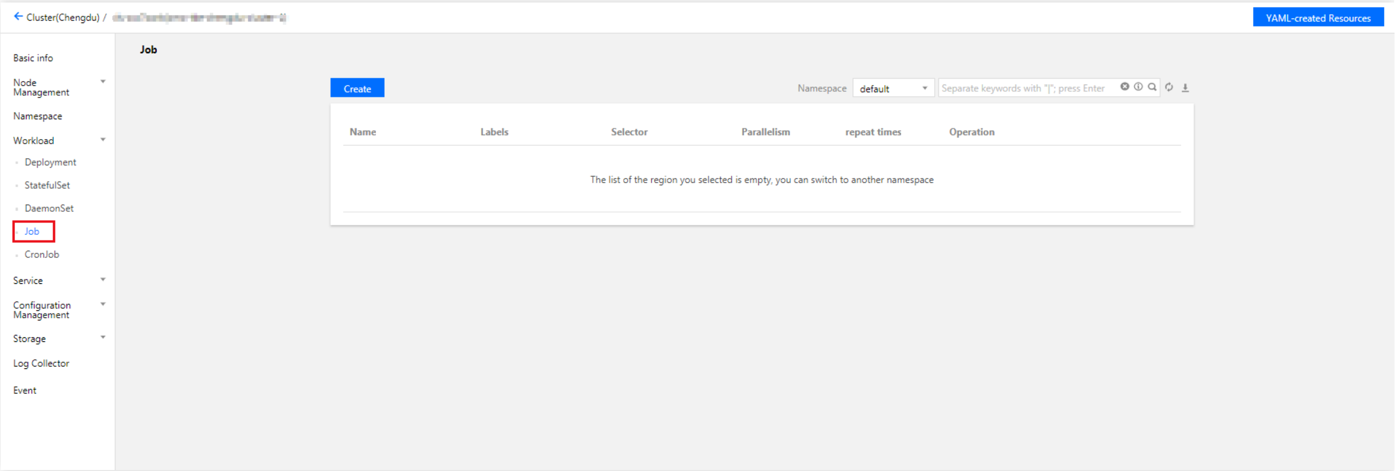
Overview

A Job creates one or more Pods and ensures that these pods run according to the specified rules until a specified number of them successfully terminate. Jobs can be used in many scenarios, such as batch computing and data analysis. You can specify the number of repeated runs, the level of parallelism and the restart policy as needed.
A Job will keep existing Pods and not create new Pods after it is complete. You can view the logs of completed Pods in "Logs". Deleting a Job will clean up the Pods it created as well as the logs of those Pods.

Managing Jobs in the Console

Creating a Job

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to go to the cluster management page.
3. Click the ID of the cluster where Job needs to be created to enter the cluster management page.
4. Select **Workload** > **Job** to go to the Job information page, as shown below:



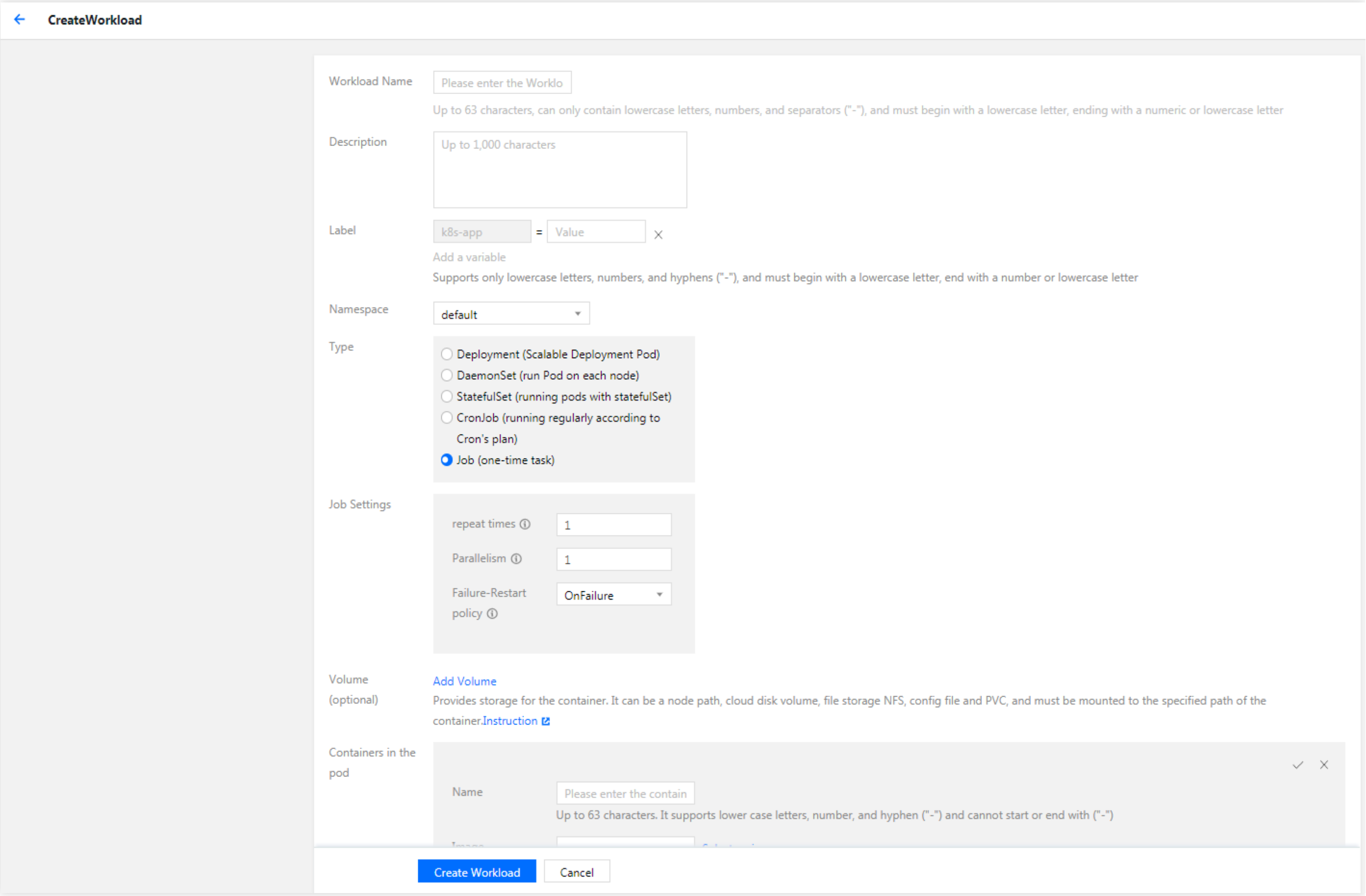
Namespace: default

Separate keywords with "|"; press Enter

Name	Labels	Selector	Parallelism	repeat times	Operation

The list of the region you selected is empty, you can switch to another namespace

5. Click **Create** to go to **Create Workload** page, as shown below:



CreateWorkload

Workload Name: Please enter the Worklo
Up to 63 characters, can only contain lowercase letters, numbers, and separators ("-"), and must begin with a lowercase letter, ending with a numeric or lowercase letter

Description: Up to 1,000 characters

Label: k8s-app = Value ×
Add a variable
Supports only lowercase letters, numbers, and hyphens ("-"), and must begin with a lowercase letter, end with a number or lowercase letter

Namespace: default

Type:
- ○ Deployment (Scalable Deployment Pod)
- ○ DaemonSet (run Pod on each node)
- ○ StatefulSet (running pods with statefulSet)
- ○ CronJob (running regularly according to Cron's plan)
- ● Job (one-time task)

Job Settings:
- repeat times ⓘ: 1
- Parallelism ⓘ: 1
- Failure-Restart policy ⓘ: OnFailure

Volume (optional): Add Volume
Provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container.Instruction

Containers in the pod:
Name: Please enter the contain
Up to 63 characters. It supports lower case letters, number, and hyphen ("-") and cannot start or end with ("-")

Create Workload Cancel

.

6. Set the Job parameters based on your actual needs. The key parameters are as follows:

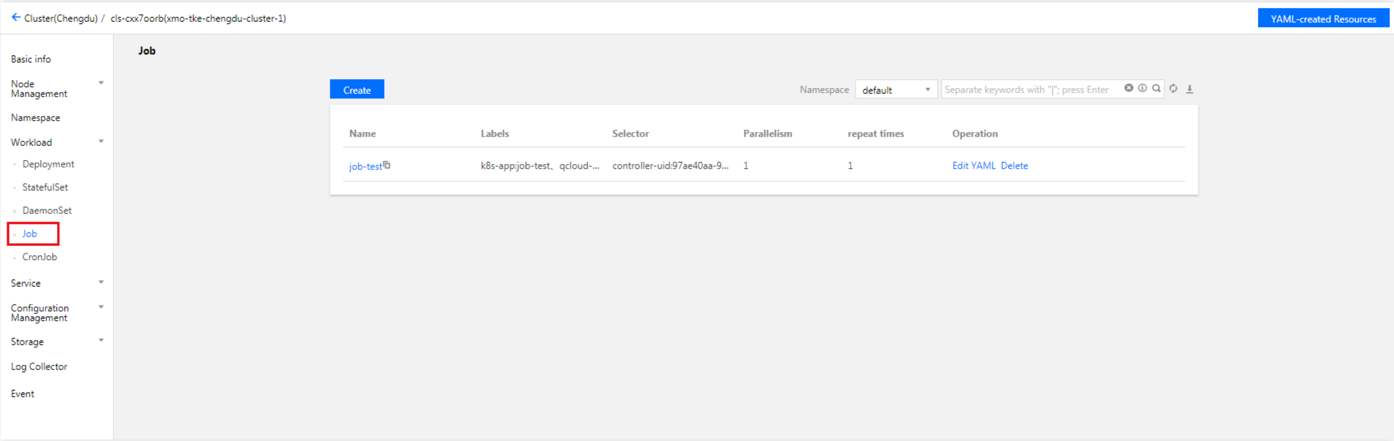
- **Workload Name**: custom.
- **Label**: a key-value pair, which is used for classified management of resources.
- **Namespace**: select a namespace based on your requirements.
- **Type**: select **Job (One-time Task)**.
- **Job Settings**: set one or more containers for a Pod of the Job as needed.
  - **Repeat Times**: set the times of repeated executions of Pods under this Job.
  - **Concurrent Pods**: set the number of parallel Pods in this Job.
  - **Restart Policy**: set the restart policy applied when containers under the Pod abnormally exit.
    - **Never**: do not restart the container until all the containers under the Pod exit.
    - **OnFailure**: the Pod continues to run while the container will be restarted.
- **Volume (optional)**: provides storage for the container. It can be a temp path, CVM path, CBS volume, file storage NFS, configuration file and PVC, and it must be mounted to the specified path of the container.
- **Containers in the Pod**: set one or more containers for a Pod of the Job as needs.
  - **Name**: custom.

  - **Image**: select as needed.
  - **Image Tag**: enter the tag based on your actual needs.
  - **CPU/memory limits**: set the CPU and memory limit according to Kubernetes' resource limits to improve the robustness of the business.
  - **GPU Resource**: you can configure the least GPU resource used by the workload.
  - **Advanced Settings**: you can set the parameters such as **Working Directory**, **Running Command**, **Running Parameter**, **Container Health Check**, and **Privileged Container**.
- **Image Access Credential**: a container image is private by default. You need to select the image access credential for the TCR instance when creating a workload.
- **Node Scheduling Policy**: the Pod can be scheduled to the node of the Label that meets the expectation according to the scheduling rules.

7. Click **Create Workload** to complete the process.

Viewing Job Status

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to go to the cluster management page.
3. Click the ID of the cluster for which you want to view the Job status to enter the cluster management page.
4. Select **Workload** > **Job** to go to the Job information page, as shown below:



Name	Labels	Selector	Parallelism	repeat times	Operation
job-test	k8s-app:job-test、qcloud-...	controller-uid:97ae40aa-9...	1	1	Edit YAML Delete

5. To view the Job's details, click its name.

Deleting a Job

A Job will keep existing Pods and not create new Pods after it is complete. You can view the logs of completed Pods in "Logs". Deleting a Job will clean up the Pods it created as well as the logs of those Pods.

Managing Jobs via Kubectl

YAML sample

```
apiVersion: batch/v1
kind: Job
metadata:
name: pi
spec:
completions: 2
parallelism: 2
template:
spec:
containers:
- name: pi
image: perl
command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
restartPolicy: Never
backoffLimit: 4
```

- kind: identifies the Job resource type.
- metadata: the basic information such as Job name and label.
- metadata.annotations: the additional description of the Job. You can set additional enhancements to TKE through this parameter.
- spec.completions: the times of repeated executions of Pods under this Job.
- spec.parallelism: the number of parallel Pods in this Job.
- spec.template: the detailed template configuration for Pod of the Job.

Creating a Job

1. See the YAML sample to prepare the Job YAML file.
2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
3. Create the Job YAML file.

   ```
   kubectl create -f Job YAML filename
   ```

For example, to create a Job YAML file named pi.yaml, run the following command:

```
kubectl create -f pi.yaml
```

4. Run the following command to check whether the Job is successfully created.

   ```
   kubectl get job
   ```

If a message similar to the following is returned, the creation is successful.

```
NAME DESIRED SUCCESSFUL AGE
job 1 0 1m
```

Deleting a Job

Run the following command to delete a Job.

```
kubectl delete job [NAME]
```

CronJob Management

Last updated：2022-04-22 11:10:46

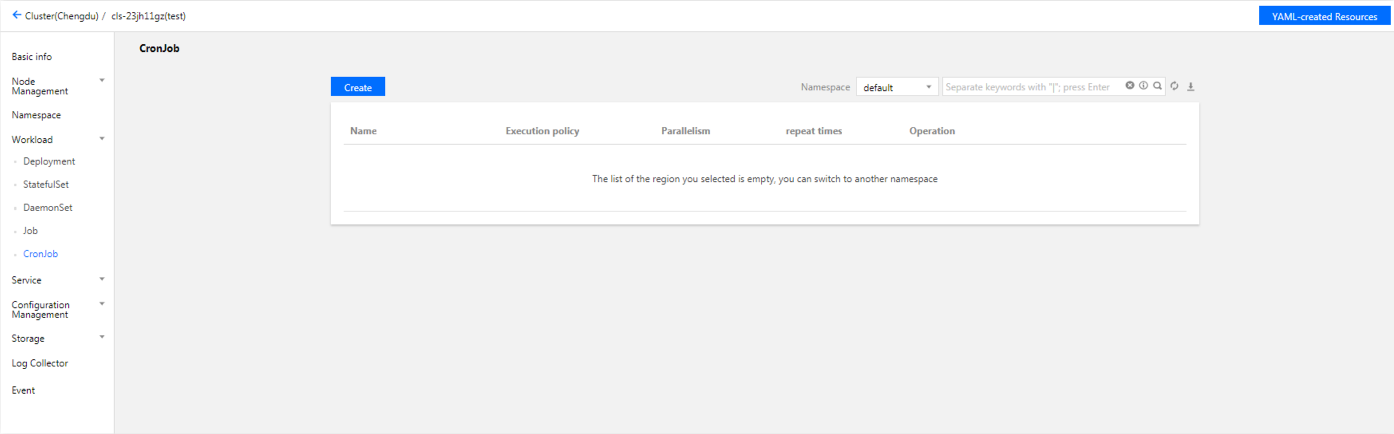
Overview

A CronJob object is similar to a line in a crontab (cron table) file. It periodically runs a Job according to the specified schedule.

Operation Guide for CronJobs in the Console

Creating a CronJob

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to go to the cluster management page.
3. Click the ID of the cluster where CronJob needs to be created to enter the cluster management page.
4. Select **Workload** > **CronJob** to go to the CronJob information page. See the figure below:



Name	Execution policy	Parallelism	repeat times	Operation

The list of the region you selected is empty, you can switch to another namespace

5. Click **Create** to open the **Create Workload** page.
6. Set the CronJob parameters based on your actual needs. The key parameters are as follows:

- **Workload Name**: custom.
- **Label**: a key-value pair, which is used for classified management of resources.
- **Namespace**: select a namespace based on your requirements.
- **Type**: select **CronJob (running according to a cron schedule)**.
- **Scheduling Rules**: select a periodic execution policy based on your business requirements.

- **Completed Jobs Retained**: it corresponds to .spec.successfulJobsHistoryLimit. For more information, see Jobs History Limits.
- **Failed Jobs Retained**: it corresponds to .spec.failedJobsHistoryLimit. For more information, see Jobs History Limits.
- **Job Settings**:
  - **Repeat Times**: set the number of times the Pod in the Job needs to be repeated.
  - **Concurrent Pods**: set the number of Pods that the Job runs in parallel.
  - **Restart Policy**: set the restart policy applied when containers under the Pod exits exceptionally.
    - Never: do not restart the container until all containers in the Pod exit.
    - OnFailure: the Pod continues to run and the container will be restarted.
- **Volume (optional)**: provides storage for the container. It can be a temp path, CVM path, CBS volume, file storage NFS, configuration file and PVC, and it must be mounted to the specified path of the container.
- **Containers in the Pod**: set one or more different containers for a Pod of the CronJob based on actual needs.
  - **Name**: custom.
  - **Image**: select as needed.
  - **Image Tag**: enter the tag based on your actual needs.
  - **Image Pull Policy**: the following three policies are available. Select as needed.
    If you do not set any image pull policy and **Image Tag** is left empty or set to `latest` , the `Always` policy is used. Otherwise, the `IfNotPresent` policy is used.
    - **Always**: always pull the image from the remote end.
    - **IfNotPresent**: a local image is used by default. If no local image is available, the image is pulled remotely.
    - **Never**: only use a local image. If no local image is available, an exception occurs.
  - **CPU/Memory Limit**: set the CPU and memory limit according to Kubernetes' resource limits to improve the robustness of the business.
  - **GPU Resource**: you can configure the least GPU resource used by the workload.
  - **Advanced Settings**: you can set the parameters such as **Working Directory**, **Running Command**, **Running Parameter**, **Container Health Check**, and **Privileged Container**.
- **Image Access Credential**: a container image is private by default. You need to select the image access credential for the TCR instance when creating a workload.
- **Node Scheduling Policy**: the Pod can be scheduled to the node of the Label that meets the expectation according to the scheduling rules.

7. Click **Create Workload** to complete the process.

Viewing CronJob status

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to go to the cluster management page.

3. Click the ID of the cluster where CronJob status needs to be checked to enter the cluster management page.
4. Select **Workload** > **CronJob** to go to the CronJob information page.
5. Click the name of the CronJob for which to view the status to view its details.

Using kubectl to Manipulate CronJobs

YAML sample

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
name: hello
spec:
schedule: "*/1 * * * *"
jobTemplate:
spec:
template:
spec:
containers:
- name: hello
image: busybox
args:
- /bin/sh
- -c
- date; echo Hello from the Kubernetes cluster
restartPolicy: OnFailure
```

- kind: this identifies the CronJob resource type.
- metadata: basic information such as CronJob name and Label.
- metadata.annotations: an additional description of the CronJob. You can set additional enhancements to TKE through this parameter.
- spec.schedule: the cron policy run by the CronJob.
- spec.jobTemplate: the Job template run by Cron.

Creating a CronJob

Method 1

1. See the YAML sample to prepare the CronJob YAML file.
2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
3. Run the following command to create the CronJob YAML file.

```
kubectl create -f CronJob YAML filename
```

For example, to create a CronJob YAML file named cronjob.yaml, run the following command:

```
kubectl create -f cronjob.yaml
```

Method 2

1. Quickly create a CronJob by running the `kubectl run` command.

   For example, to quickly create a CronJob that does not need to write full configuration information, run the following command:

   ```
   kubectl run hello --schedule="*/1 * * * *" --restart=OnFailure --image=busybox
   -- /bin/sh -c "date; echo Hello"
   ```

2. Run the following command to see if the item is successfully created.

   ```
   kubectl get cronjob [NAME]
   ```

If a message similar to the following is returned, the creation is successful.

```
NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE
cronjob * * * * * False 0 <none> 15s
```

Deleting a CronJob

Note：

- Before running this deletion command, please confirm whether there is a Job being created; if yes, running this command will terminate that Job.
- When you run this deletion command, created Jobs and completed Jobs will not be terminated or deleted.
- To delete a Job created by CronJob, please do so manually.

Run the following command to delete a CronJob.

```
kubectl delete cronjob [NAME]
```

Setting the Resource Limit of Workload

Last updated：2022-04-25 14:51:21

Request and Limit

Request: This refers to the minimum requirement of resources used by a container, which serves as a judgment criterion of resource allocation when the container is scheduled. The container is allowed to be scheduled to a node only when the amount of resources available on the node is greater than or equal to the container resource request quantity. However, the Request parameter does not limit the maximum value of resources available to the container.
Limit: This is the maximum value of resources available to a container.

Note：
For more information about the **Limit** and **Request** parameters, click here.

CPU Limit Description

For CPU resources, you can set the amount of resources for CPU Request and CPU Limit, which is in cores (U) and can be decimals.

Note：

- CPU Request is used as the basis for scheduling. When a container is created, CPU resources are allocated to it on the node, which are called "allocated CPU" resources.
- CPU Limit is the upper limit of the container's CPU resources. If it is not set, there will be no limit (CPU Limit >= CPU Request).

Memory Limit Description

For memory resources, you can only limit the maximum amount of memory available to a container. It is in MiB and can be decimals.

Note：

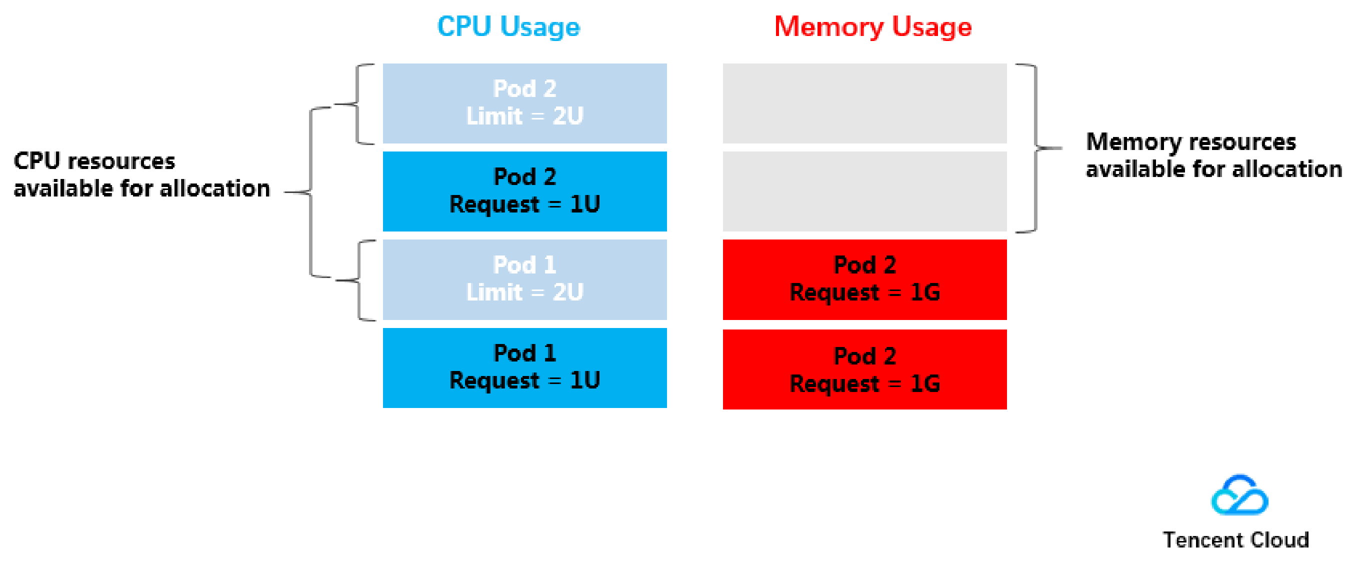
- Memory Request is used as the basis for scheduling. When a container is created, memory resources are allocated to it on the node, which is called the "allocated memory" resources.
- Memory resources are non-scalable. There will be a risk of OOM if the memory resources used by all containers on the node exceed the limit. Therefore, if Limit is not set, containers can use all resources available on the node, which causes resources of other containers to be occupied and the Pod on which this type of containers located is easy to be drained. It is recommended to set Limit = Request.

CPU Usage and CPU Utilization

- The CPU usage is an absolute value indicating the number of physical CPU cores actually used. Both the CPU resource Request and CPU resource Limit are judged based on the CPU usage.
- CPU utilization is a relative value indicating the ratio of CPU usage to a single CPU core (or total CPU cores on the node).

Examples

Below is a simple example that illustrates the roles of Request and Limit. The test cluster consists of one 4U4G node, two deployed Pods (Pod1 and Pod2), and the resources of each Pod are set as (CPU Request, CPU Limit, Memory Request, Memory Limit) = (1U, 2U, 1G, 1G). (1.0G = 1000MiB)
The usage of CPU and memory resources on the node is as shown in the figure below:



The allocated CPU resources are 1U (for Pod1) + 1U (for Pod2) = 2U, and the remaining CPU resources available for allocation are 2U.
The allocated memory resources are 1G (for Pod1) + 1G (for Pod2) = 2G, and the remaining memory resources available for allocation are 2G.
Therefore, one more (CPU Request, Memory Request) = (2U, 2G) Pod or 2 more (CPU Request, Memory Request) = (1U, 1G) Pods can be deployed on the node.

In terms of resource limitations, the upper limit of the resources used by Pod1 and Pod2 is (2U, 1G), which means that the maximum CPU resources available to the Pods are 2U if there are idle resources.

Recommended Service Resource Limitations

TKE will recommend the Request and Limit values based on the historical load of your current container image. Using the recommended values will ensure that your container runs more smoothly and reduce the probability of exceptions.

Recommendation algorithm:
The algorithm first takes the values of load per minute in the current container image in the past seven days, and then uses the 95th percentile value to determine the recommended Request, which is half of the Limit.

```
Request = Percentile(actual load[7d],0.95)
Limit = Request * 2
```

If the current sample size (actual load) does not meet the quantity requirement of recommendation calculation, the algorithm will expand the sample value range accordingly and try to recalculate. For example, it will retry after removing filter criteria such as image tag, namespace, and serviceName. If no valid values are obtained after multiple calculations, the recommended values will be blank.

Blank recommended values:
During normal use, you may find that there are no recommendations for some values, which may be caused by the following:

1. The current data does not meet the calculation requirements. At least 1440 samples (actual load), i.e., the data of one day, should be present.
2. The recommended value is less than the Request or Limit that has already been configured for your current container.

Note：

1. As the recommended values are calculated based on the historical load, in principle, the longer the container image runs real businesses, the more accurate the recommended values.
2. When you create a service using the recommended values, container scheduling may fail due to insufficient cluster resources. When saving the service, you should carefully check the remaining resources in the current cluster.
3. The recommended values are only for reference. You can make adjustments based on the actual business needs.

References

The Request and Limit values of containers need to be set based on service types, demands and scenarios. For more information, see Setting Request and Limit.

Setting the Scheduling Rule for a Workload

Last updated：2023-05-25 11:22:18

Overview

You can specify the scheduling of the Pods under a workload in the cluster by setting the scheduling policy in the advanced settings of the workload. The following are the typical use cases:

- Run the Pods on a specific node.
- Run the Pods on nodes in a specific scope (which can be an availability zone, a model and so on).

How to Configure Scheduling policies

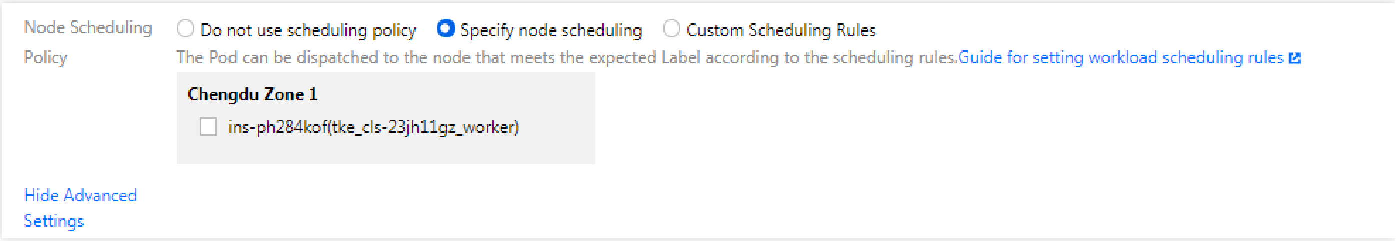
Prerequisites

- Kubernetes 1.7 or later, and scheduling policies are configured. Scheduling policies can be found under the Advanced Settings of a workload.
- To ensure that your Pods can be scheduled successfully, the node should have resources available for container scheduling after the scheduling policy is configured.
- You need node labels if you use custom scheduling features. For details, refer to Setting Node Label.

Configuring a scheduling policy

If your cluster is created using Kubernetes 1.7 or later, you can set a scheduling policy when creating a workload. Select one of the following scheduling types:

- **Schedule to a specific node**: schedule Pods to a specific node with matching node labels.



Node Scheduling Policy ○ Do not use scheduling policy ● Specify node scheduling ○ Custom Scheduling Rules

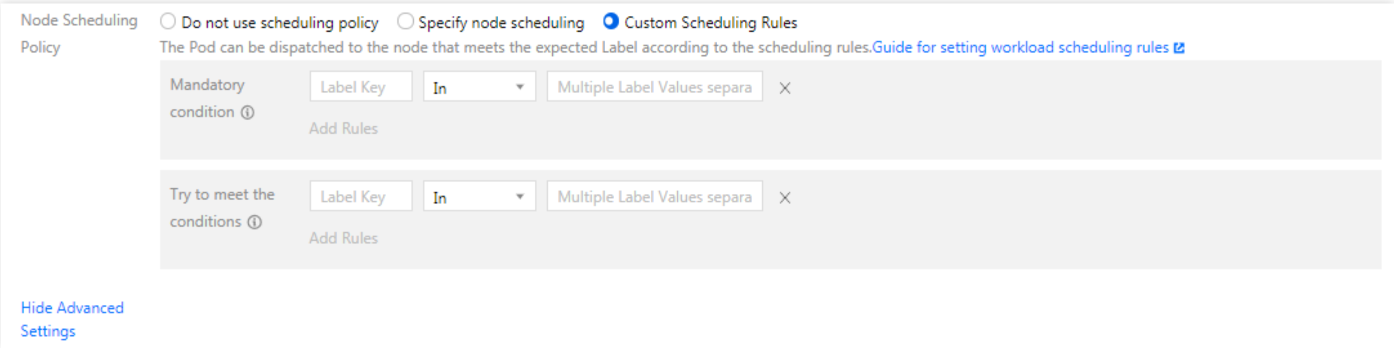
The Pod can be dispatched to the node that meets the expected Label according to the scheduling rules.Guide for setting workload scheduling rules

Chengdu Zone 1

□ ins-ph284kof(tke_cls-23jh11gz_worker)

Hide Advanced Settings

- **Custom scheduling**: customize how Pods are scheduled by matching Pod labels.



Node Scheduling Policy ○ Do not use scheduling policy ○ Specify node scheduling ● Custom Scheduling Rules

The Pod can be dispatched to the node that meets the expected Label according to the scheduling rules.Guide for setting workload scheduling rules

Mandatory condition ⓘ　Label Key　In ▾　Multiple Label Values separa　×

Add Rules

Try to meet the conditions ⓘ　Label Key　In ▾　Multiple Label Values separa　×

Add Rules

Hide Advanced Settings

Custom scheduling policies have the following modes:

- Conditions that must be met: if a node meets the affinity conditions, the Pods are scheduled to the corresponding node. If not, the scheduling fails.
- Conditions that should be met if possible: if a node meets the affinity conditions, the Pods are scheduled to the corresponding node. If not, the pods are scheduled to a random node.

Multiple custom scheduling policies can be added. The following is a list of operators:

- In: the value of the label is in the list.
- NotIn: the value of the label is not in the list.
- Exists: the key of the label exists.
- DoesNotExits: the key of the label does not exist.
- Gt: the value of the label is greater than the listed value (string match).
- Lt: the value of the label is less than the listed value (string match).

How It Works

Kubernetes uses YAML files to distribute scheduling policies and the affinity and anti-affinity mechanism ensures that Pods are scheduled according to rules. For more information on this mechanism, refer to Kubernetes official documentation.

Setting the Health Check for a Workload

Last updated：2021-01-27 14:49:19

The Tencent Cloud TKE kernel is based on Kubernetes, which periodic probes containers. It then uses the results to decide the health status of the containers and perform actions if needed.

Health Check Types

Health checks are divided into the following types:

- **Liveness check**: it checks whether the container is alive, which is similar to checking whether a process exists using ps. If the liveness check fails, the cluster will restart the container. No action is performed if the liveness check succeeds.
- **Readiness check**: it checks whether the container is ready to handle user requests. For example, if an application takes a while to start up because it relies on a data disk mount or an external module startup to provide service, Kubernetes will use a readiness probe to check if the application has finished starting. If the readiness probe fails, requests to that container are blocked until the readiness probe succeeds.

Health Check Methods

TCP port probes

TCP port probe works as follows:
The cluster periodically tries to establish TCP connections to containers that provide TCP services. If the connection is established, the probe is successful. Otherwise, the probe fails. For TCP probes to work, you must specify the listening port of the container.
For example, you have a Redis service running on port 6379 and a TCP probe is set up to examine port 6379. The cluster will periodically try to establish a TCP connection to the container on port 6379. If the connection is established, the probe succeeds. If not, the probe fails.

HTTP request probes

HTTP request probes are for containers that provide HTTP/HTTPS services, where the cluster periodically initiates HTTP/HTTPS GET requests to the containers. If the return code of the HTTP/HTTPS response is in the range of 200 - 399, the probe succeeds; otherwise, it fails. When the HTTP request probe is used, be sure to specify the listening port and the HTTP/HTTPS request path.
For example, for a container that provides HTTP services with a service port of 80 and HTTP check path of

`/health-check` , the cluster will periodically initiate a `GET http://containerIP:80/health-check` request to the container.

Execute command check

It is a powerful check method. After the user specifies an executable command within the container, the cluster periodically executes the command. If the result is 0, the check succeeds. Otherwise, the check fails.
You can replace both TCP port probes and HTTP request probes with execute command checks:

- To replace TCP port probes, you can write a specific program to connect to a port of a container. If the connection succeeds, the script returns 0; otherwise, it returns -1.
- To replace HTTP request probes, you can write a script to execute wget to retrieve content from the container and check the return code of the response. For example, `wget http://127.0.0.1:80/health-check` . If the return code is in the range of 200 - 399, the script returns 0; otherwise, it returns -1.

Considerations

- The program to be run must be part of the image of the container. Otherwise, the execute command check will fail because the program cannot be found.
- If the execute command check uses a shell script, you cannot use the script as the command. Use a script interpreter to perform the check. For example, if the script is `/data/scripts/health_check.sh` , use the following:

  ```
  sh
  /data/scripts/health_check.sh
  ```

  The following shows how to set up health checks using the TKE Console:

  i. In the **Deployment** page of the cluster, click **Create**. The **CreateWorkload** page appears.
  ii. Select **Advanced Settings** in the **Containers in a pod** section.
  iii. Select **Liveness Check** for **Container Health Check** and configure the following parameters:
    - **Checking Method**: select **Execute Command Check**.
    - **Execute Command**: enter the following:

      ```
      sh
      /data/scripts/health_check.sh
      ```

  iv. For other parameters, refer to Deployment Management.

Other Common Parameters

- **Start-up Latency**: unit: seconds. This specifies the time to wait before starting a probe after a container is launched. For example, if the start delay is set to 5, the health check will start 5 seconds after the container is launched.

- **Interval**: unit: second. It specifies the frequency of health checks. For example, if the interval is set to 10, then health checks are run once every 10 seconds.
- **Response timeout**: unit: second. It specifies the timeout period for health check. It indicates the TCP connection timeout period, the HTTP request response timeout period, and the execute command timeout period for TCP port probe, HTTP request probe, and execute command check, respectively.
- **Healthy Threshold**: unit: time. It specifies the times of consecutive health check successes before the container is determined to be healthy. For example, if the healthy threshold is set to 3, the container will be considered healthy only if the probe succeeds three times consecutively.

> ⚠ **Note：**
>
> For liveness checks, the healthy threshold can only be 1. Other values are invalid.

- **Unhealthy Threshold**: unit: time. It specifies the times of consecutive health check failures before the container is determined to be unhealthy. For example, if the unhealthy threshold is set to 3, the container will be considered unhealthy only if the probe fails three times consecutively.

Setting the Run Command and Parameter for a Workload

Last updated：2022-04-18 15:06:22

Overview

When you create a workload, an image is used to specify the process of the container in the pod. By default, the image runs a default command. To run a specific command or to rewrite the default values of the image, you need to use the following three settings:

- Working directory (workingDir): this specifies the current working directory.
- Run command (command): this controls the actual command run by the image.
- Command argument (args): this is the parameter passed to the running command.

Working Directory

WorkingDir specifies the current working directory. If it does not exist, a working directory will be automatically created. If this parameter is not specified, the default value of the container runtime will be used. If WORKDIR is not specified in the image or console, the default value of workingDir will be “/”.

Command and Parameter Usage

For information on how to adapt the docker run command to Tencent Cloud TKE, see docker run Parameter Adaptation.

Docker images contain metadata related to image information storage. If you do not specify any command or parameter for the container, the container may run the default command and parameter used when the image is created. By default, they are `Entrypoint` and `CMD` in Docker. For more information, see Entrypoint and CMD from Docker.
If you enter the run commands and parameters for the container when creating a service, TKE will override the default commands (i.e., "Entrypoint" and "CMD") when the image is created. The rules are as follows:

Image Entrypoint	Image CMD	Container's run command	Container's run parameter	Final run
[ls]	[/home]	Not set	Not set	[ls / home]

Image Entrypoint	Image CMD	Container's run command	Container's run parameter	Final run
[ls]	[/home]	[cd]	Not set	[cd]
[ls]	[/home]	Not set	[/data]	[ls / data]
[ls]	[/home]	[cd]	[/data]	[cd / data]

Note：

- Docker entrypoint corresponds to the run command on the TKE console, and the parameter CMD of Docker run corresponds to the run parameter on the TKE console. If multiple parameters exist, enter them in the parameter field of TKE with each parameter on its own row.
- For examples on how to use the TKE console to set the running command and parameter for a container, see Commands and Args.

Using a Container Image in a TCR Enterprise Instance to Create a Workload

Last updated：2021-11-26 10:10:18

Overview

The Tencent Container Registry (TCR) Enterprise Edition provides enterprise-grade exclusive and secure image hosting services for enterprise-grade container customers who have strict data security and compliance requirements, businesses distributed across multiple regions, and large cluster scales. Compared with the TCR Personal Edition, the TCR Enterprise Edition supports container image secure scanning, cross-region automatic synchronization, Helm chart hosting, network access control, and other features. For more information, see Tencent Container Registry.

This document describes how to use a private image hosted in TCR to deploy applications in Tencent Kubernetes Engine (TKE).

Prerequisite

Before you use a private image hosted in TCR to deploy applications in TKE, ensure that you have completed the following operations:

- You have created a TCR Enterprise Edition instance in the TCR console. If you have not created a TCR Enterprise Edition instance, create one. For more information, see Creating an Enterprise Edition Instance.
- If you are using a sub-account, you must have granted the sub-account operation permissions for the corresponding instance. For more information, see Example of Authorization Solution of the Enterprise Edition.

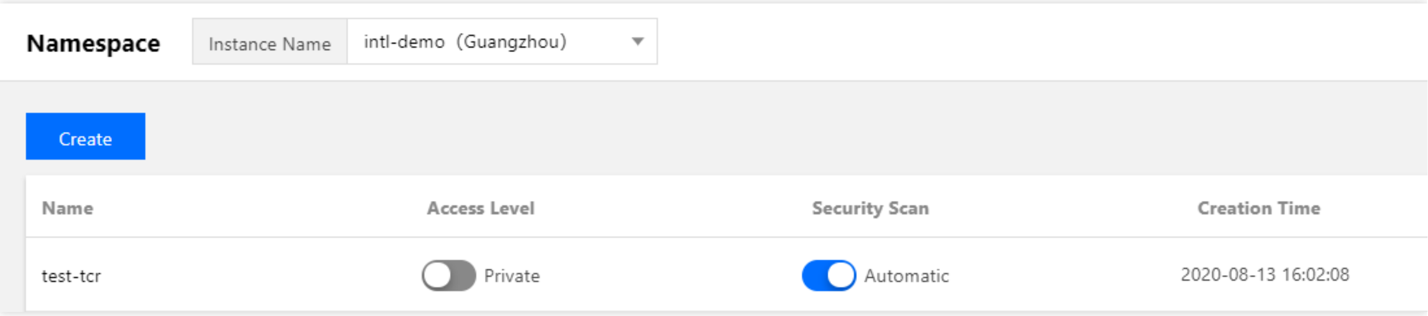
Directions

Preparing a container image

Creating a namespace

A new TCR Enterprise Edition instance does not have a default namespace, and a namespace cannot be automatically created through the pushed image. Therefore, create a namespace as required. For more information, see Managing Namespaces.
We recommend that the namespace be named based on the project or team name. In this document, `docker` is

used as an example. The following page appears after the namespace is created.



Namespace

Instance Name: intl-demo (Guangzhou)

Name	Access Level	Security Scan	Creation Time
test-tcr	Private	Automatic	2020-08-13 16:02:08

Creating an image repository (optional)

Container images are hosted in specific image repositories. Create an image repository as required. For more information, see Creating an Image Repository. Set the image repository name to the name of the container image to be deployed. In this document, `getting-started` is used as an example. The following page appears after the image repository is created.

> Note：
> Use Docker CLI or another image tool, such as jenkins, to push the image to the TCR Enterprise Edition instance. If no image repository exists, an image repository will be automatically created. You do not need to create one in advance.

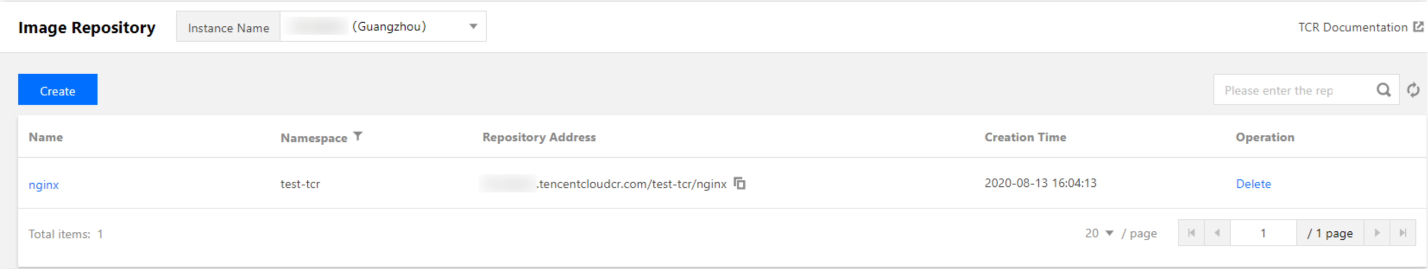


Image Repository

Instance Name: [redacted] (Guangzhou)

Name	Namespace	Repository Address	Creation Time	Operation
nginx	test-tcr	[redacted].tencentcloudcr.com/test-tcr/nginx	2020-08-13 16:04:13	Delete

Total items: 1

Pushing a container image

You can use Docker CLI or another image, such as jenkins to push an image to a specific image repository. Here, the Docker CLI is used to push images. To push a container image, you need to use a CVM or physical server with Docker installed and ensure that the Client is allowed to access the instance. For more information, see Network Access Control Overview.

1. Obtain an access credential for the TCR Enterprise Edition instance and run the Docker login command to log in to the instance. For more information on how to obtain an instance access credential, see Obtaining an Instance

Access Credential.

2. After successful login, create a container image on the local server or obtain a public image from Docker Hub for testing.

   This document uses the latest Nginx image on the official Docker Hub website as an example. In the command-line tool, run the following commands sequentially to push this image. Note to replace `demo-tcr` , `docker` , and `getting-started` with the actual instance, namespace, and image repository names that you created.

   ```
   docker tag getting-started:latest demo-tcr.tencentcloudcr.com/docker/getting-started:latest
   ```

   ```
   docker push demo-tcr.tencentcloudcr.com/docker/getting-started:latest
   ```

   After the image is pushed, you can go to the "Image Repository" page in the TCR console and select a repository name to view details.

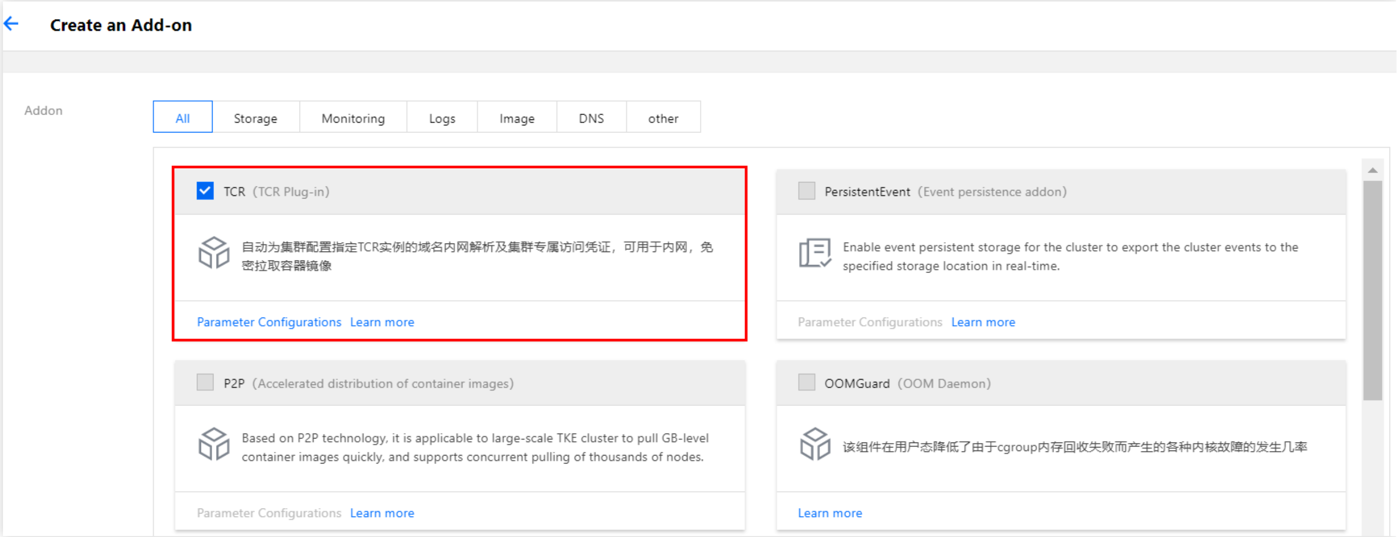
Configure TKE cluster access TCR instance

TCR Enterprise Edition instances support network access control and deny all external access by default. You can select a public network or private network for a TKE cluster to access a specific instance and pull the container image based on the network configuration of the TKE cluster. If the TKE cluster and TCR instance are deployed in the same region, we recommend that the TKE cluster pull the container image through a private network to accelerate pulling and reduce public network traffic costs.

Using the TCR add-on for quick access configuration (recommended)

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, click **Add-on Management** in the left sidebar to go to the **Add-on Management** page and click **Create**.
4. On the **Create an add-on** page, select **TCR**, as shown below.

> Note：
> Currently, the TCR add-on only supports clusters in Kubernetes 1.12, 1.14, 1.16, 1.18 and 1.20. If you are using another cluster version, manually configure the access method or upgrade the cluster version.



Create an Add-on

Addon

☑ TCR (TCR Plug-in)

自动为集群配置指定TCR实例的域名内网解析及集群专属访问凭证，可用于内网，免密拉取容器镜像

Parameter Configurations Learn more

□ PersistentEvent (Event persistence addon)

Enable event persistent storage for the cluster to export the cluster events to the specified storage location in real-time.

Parameter Configurations Learn more

□ P2P (Accelerated distribution of container images)

Based on P2P technology, it is applicable to large-scale TKE cluster to pull GB-level container images quickly, and supports concurrent pulling of thousands of nodes.

Parameter Configurations Learn more

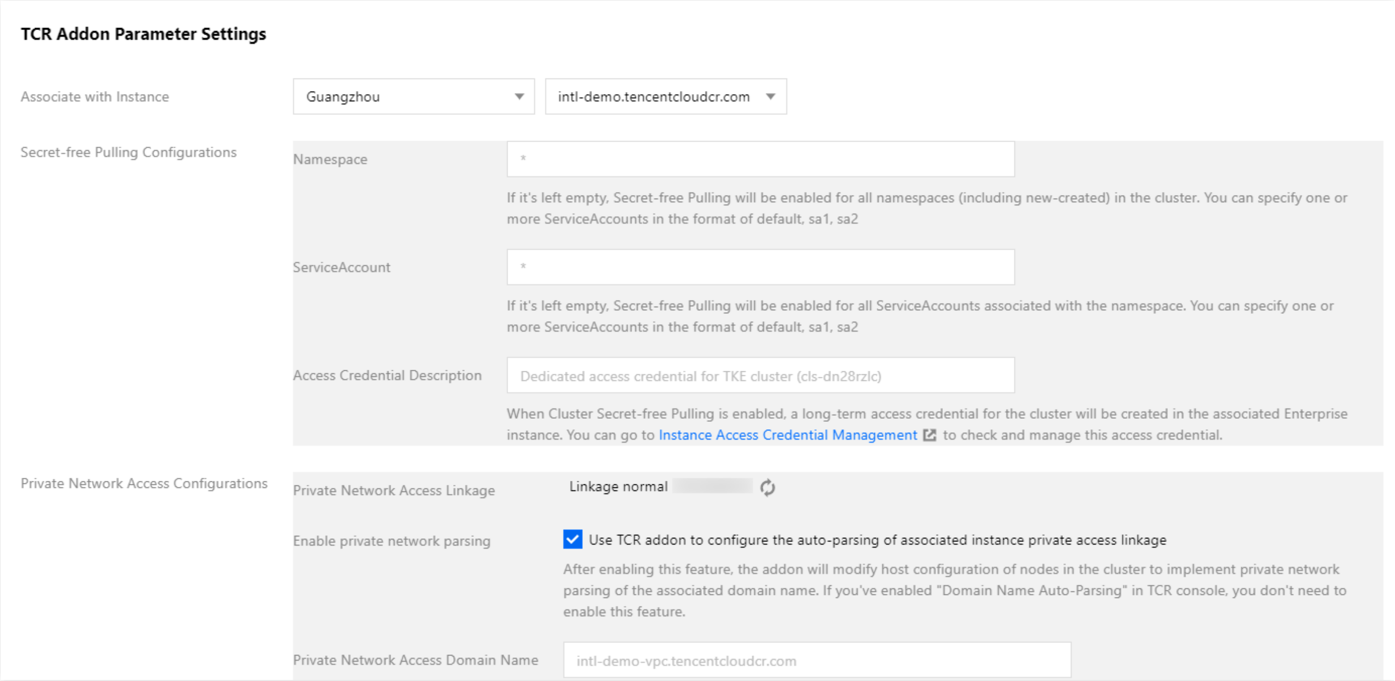
□ OOMGuard (OOM Daemon)

该组件在用户态降低了由于cgroup内存回收失败而产生的各种内核故障的发生几率

Learn more

- Click **View Details** to view the add-on features and configuration description.
- Click **Parameter Configurations** to configure the add-on.

5. On the **TCR Add-on Parameter Settings** page, configure related parameters based on the add-on configuration method described in **View Details**, as shown below:



TCR Addon Parameter Settings

Associate with Instance: Guangzhou ▼ intl-demo.tencentcloudcr.com ▼

Secret-free Pulling Configurations

Namespace: *

If it's left empty, Secret-free Pulling will be enabled for all namespaces (including new-created) in the cluster. You can specify one or more ServiceAccounts in the format of default, sa1, sa2

ServiceAccount: *

If it's left empty, Secret-free Pulling will be enabled for all ServiceAccounts associated with the namespace. You can specify one or more ServiceAccounts in the format of default, sa1, sa2

Access Credential Description: Dedicated access credential for TKE cluster (cls-dn28rzlc)

When Cluster Secret-free Pulling is enabled, a long-term access credential for the cluster will be created in the associated Enterprise instance. You can go to Instance Access Credential Management to check and manage this access credential.

Private Network Access Configurations

Private Network Access Linkage: Linkage normal [redacted]

Enable private network parsing: ☑ Use TCR addon to configure the auto-parsing of associated instance private access linkage

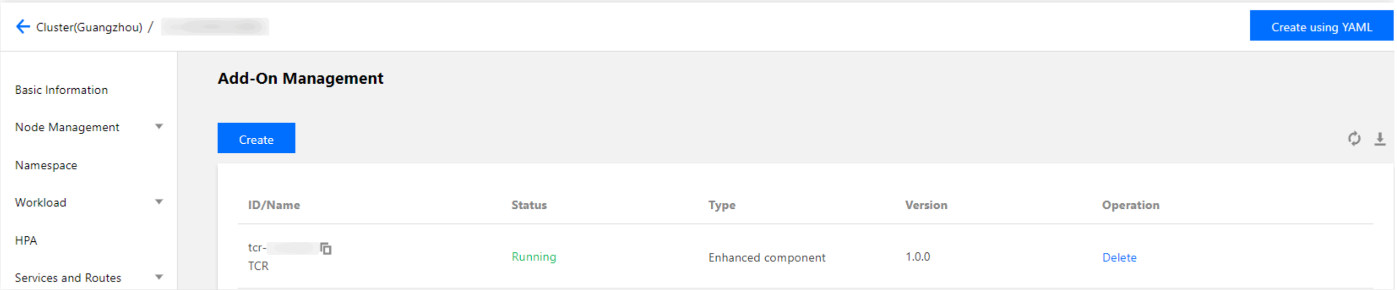
After enabling this feature, the addon will modify host configuration of nodes in the cluster to implement private network parsing of the associated domain name. If you've enabled "Domain Name Auto-Parsing" in TCR console, you don't need to enable this feature.

Private Network Access Domain Name: intl-demo-vpc.tencentcloudcr.com

- **Associate with Instance**: select a TCR instance in the same region as the TKE cluster.
- **Password-free Pulling**: retain the default setting.
- **Private Network Access Configurations**: this is an optional feature. If the TCR instance has accessed to the VPC where the cluster is in and has enabled auto-parsing, the nodes in the cluster can access the TCR instance through the private network without using this feature. Because the auto-parsing feature of the TCR instance is dependent on PrivateDNS, you can use this configuration to implement private network access if PrivateDNS

has not been supported in the region where the cluster is located. If "Linkage normal" is not displayed in the private network access linkage, you need to configure the private network linkage of the VPC where the TCR instance and the TKE cluster are located in. For more information, see Private Network Access Control.

6. Click **OK** to go back to add-on selection page.
7. On the add-on selection page, click **Done** to install the TCR add-on for the cluster.
8. After the add-on is installed, the cluster can pull images from the associated instance without needing a password through a private network, as shown below.



Add-On Management

ID/Name	Status	Type	Version	Operation
tcr-[redacted] TCR	Running	Enhanced component	1.0.0	Delete

Manually configuring private network access and the access credential

1. Configuring private network access

1. Configure a private network linkage of the VPC where the TCR instance and the TKE cluster are located in and enable auto-parsing. For more information, see Private Network Access Control.
2. If auto-parsing is not supported in the region where the current TCR instance is located, you can configure the domain name parsing for the TCR instance in the TKE cluster. You can choose from the following solutions based on the actual needs:
   - **Configuring the node host when creating the cluster**
     In the "CVM Configuration" step during the TKE cluster creation process, select **Advanced Settings** and enter the following content in "Node Launch Configuration":

     ```
     echo '172.21.17.69 demo.tencentcloudcr.com' >> /etc/hosts
     ```

   - **Configuring the node host for an existing cluster**
     Log in to the cluster nodes and run the following command:

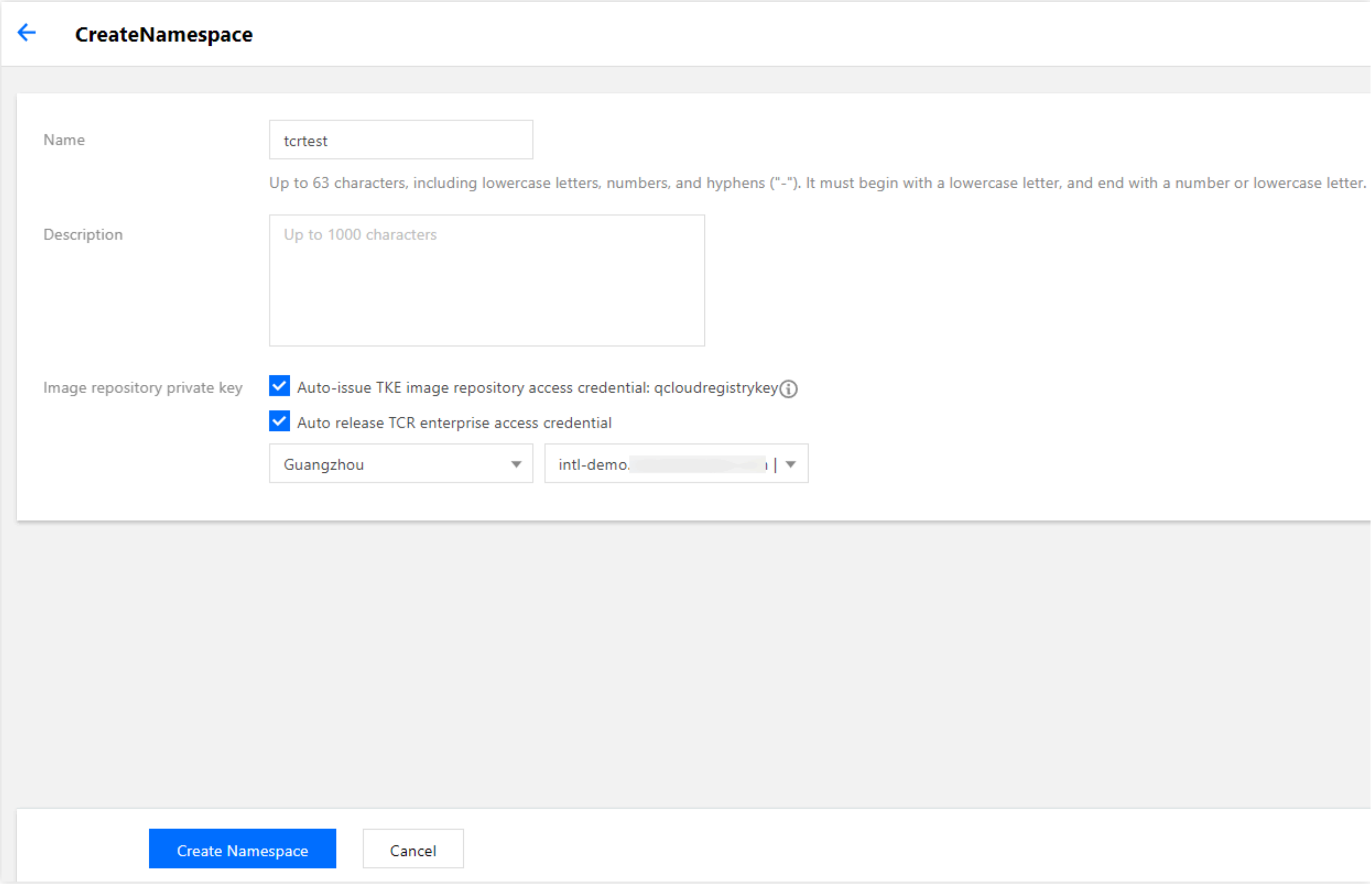
     ```
     echo '172.21.17.69 demo.tencentcloudcr.com' >> /etc/hosts
     ```

     Replace `172.21.17.69` and `demo.tencentcloudcr.com` with the private network resolution IP address and TCR instance domain name that you use.

2. Configuring access credential

When creating a namespace, follow the steps below to deliver an access credential:

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. Click **Namespace** in the left sidebar to go to the **Namespace** page and click **Create**.
4. On the **CreateNamespace** page, select **Auto-issue TKE image repository access credential** and select the TCR instance that the cluster needs to access, as shown below.



CreateNamespace

Name: tcrtest

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

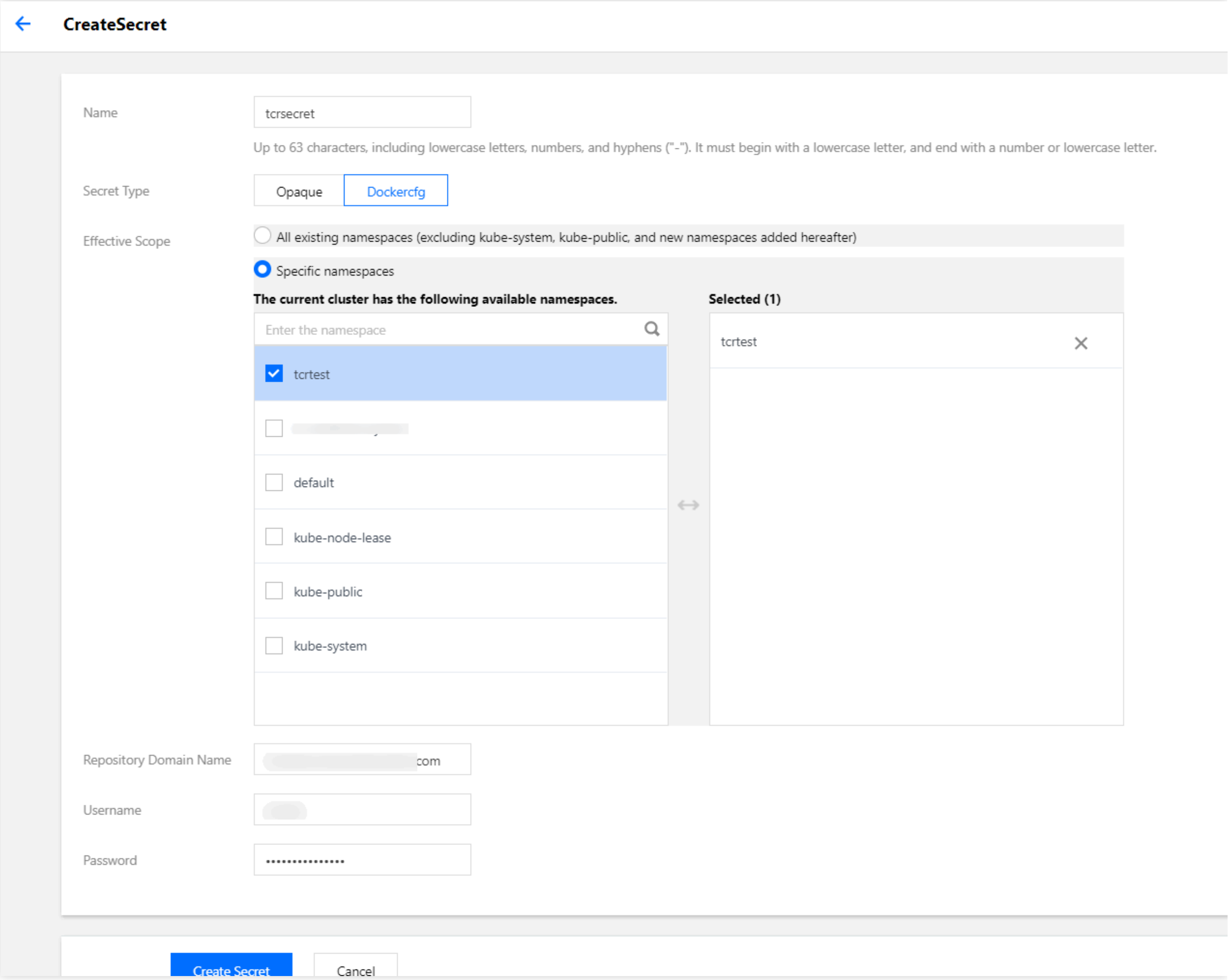
Description: Up to 1000 characters

Image repository private key:
☑ Auto-issue TKE image repository access credential: qcloudregistrykey
☑ Auto release TCR enterprise access credential
Guangzhou | intl-demo.[redacted]

5. Click **Create Namespace**.
   After the namespace is created, the access credential of the instance is automatically delivered to the namespace. To view the access credential, for example, `1000090225xx-tcr-m3ut3qxx-dockercfg` , choose **Configuration Management** > **Secret**. `1000090225xx` indicates the UIN of the sub-account used to create the namespace, and `tcr-m3ut3qxx` indicates the ID of the selected instance.

Perform the following steps to deliver the access credential to an existing namespace:

1. Obtain the username and password used to log in to the instance. For more information, see Obtaining an Instance Access Credential.
2. On the cluster details page, choose **Configuration Management** > **Secret** in the left sidebar to go to the **Secret** page.
3. On the **Secret** page, click **Create** to go to the **CreateSecret** page, as shown below. Refer to the following information to deliver the access credential.



CreateSecret

Name: tcrsecret

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Secret Type: Opaque | Dockercfg

Effective Scope:

○ All existing namespaces (excluding kube-system, kube-public, and new namespaces added hereafter)

● Specific namespaces

The current cluster has the following available namespaces.

Enter the namespace

- ☑ tcrtest
- □ [redacted]
- □ default
- □ kube-node-lease
- □ kube-public
- □ kube-system

Selected (1)

tcrtest ✕

Repository Domain Name: [redacted]com

Username: [redacted]

Password: ••••••••••••••••

Create Secret | Cancel

The main parameters are described as follows:

- **Secret Type**: select **Dockercfg**.
- **Effective Scope**: select the namespace to which the access credential is delivered.
- **Repository Domain Name**: enter the access domain name of the TCR instance.
- **Username and Password**: enter the username and password obtained in Step 1.

4. Click **Create Secret** to deliver the access credential.

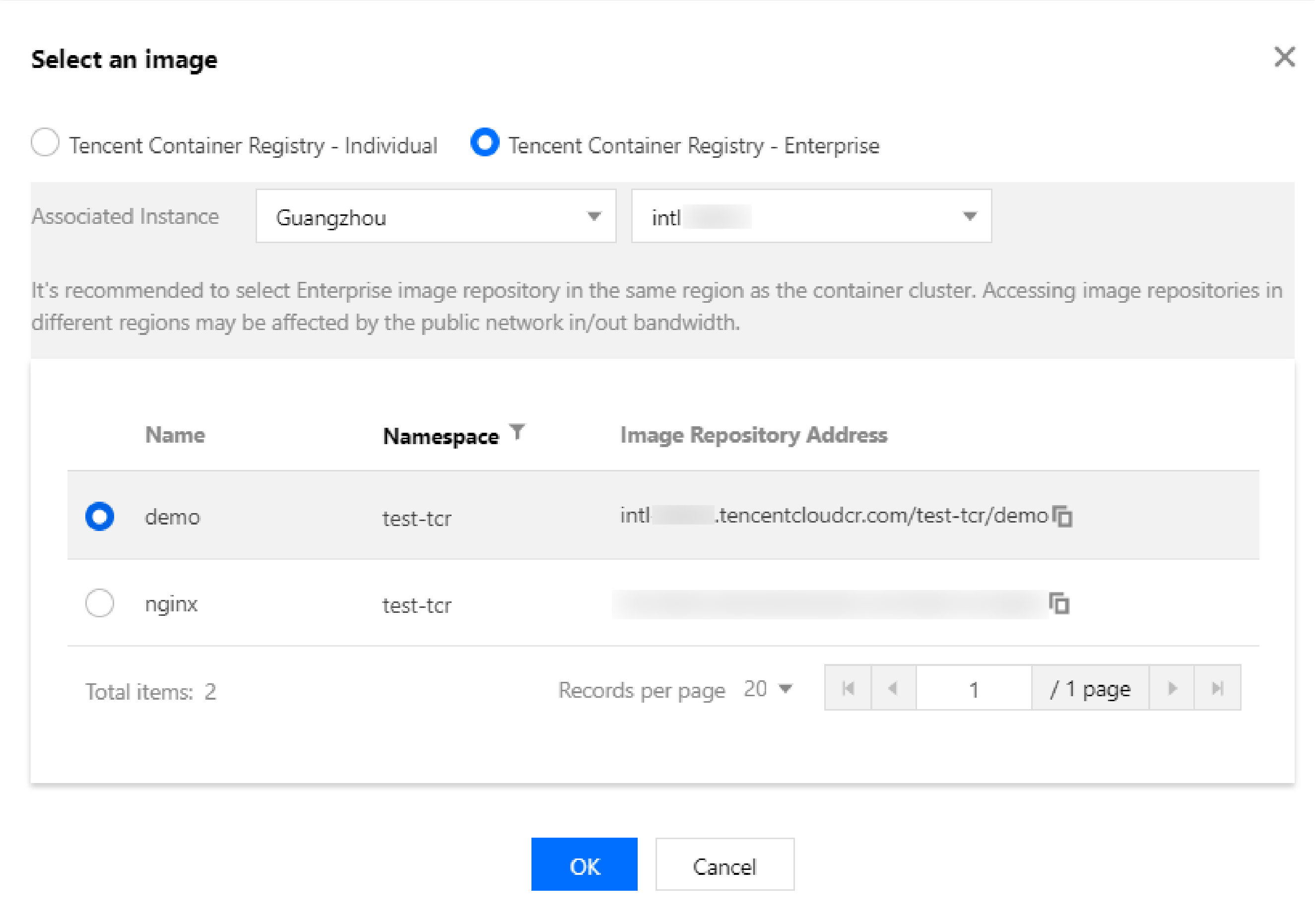
Using the container image in the TCR instance to create a workload

1. On the cluster details page, select **Workload** > **Deployment** in the left sidebar.
2. On the **Deployment** page, click **Create**.
3. On the **CreateWorkload** page, set the following parameters to create a workload.

   The main parameters are described as follows:

   - **Namespace**: select the namespace to which the access credential is delivered.
   - **Containers in the Pod**:

- **Image**: click **Select Image**, select **Tencent Container Registry - Enterprise** in the pop-up, and select region, instance and image repository based on your needs. See the figure below:



Select an image

○ Tencent Container Registry - Individual ● Tencent Container Registry - Enterprise

Associated Instance: Guangzhou | intl[redacted]

It's recommended to select Enterprise image repository in the same region as the container cluster. Accessing image repositories in different regions may be affected by the public network in/out bandwidth.

	Name	Namespace	Image Repository Address
●	demo	test-tcr	intl[redacted].tencentcloudcr.com/test-tcr/demo
○	nginx	test-tcr	[redacted]

Total items: 2 Records per page 20 1 / 1 page

- **Image Tag**: after you select an image, click **Select Image Tag**, and select a tag for the image repository based on your needs in the pop-up. If you do not select, the `latest` will be used by default.

- **Image Access Credential**:
  - If the cluster has the TCR add-on installed, it does not need to be configured.
  - If the cluster does not have the TCR add-on installed, click **Add Image Access Credential** and select the access credential delivered in the step of Configuring the access credential. See the figure below:

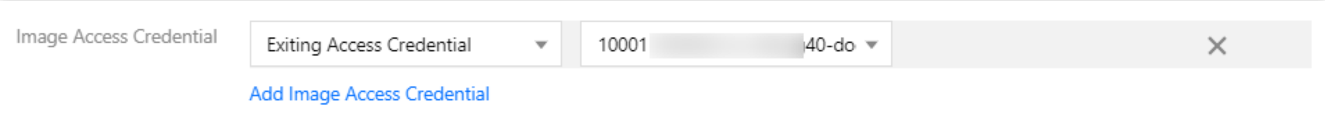
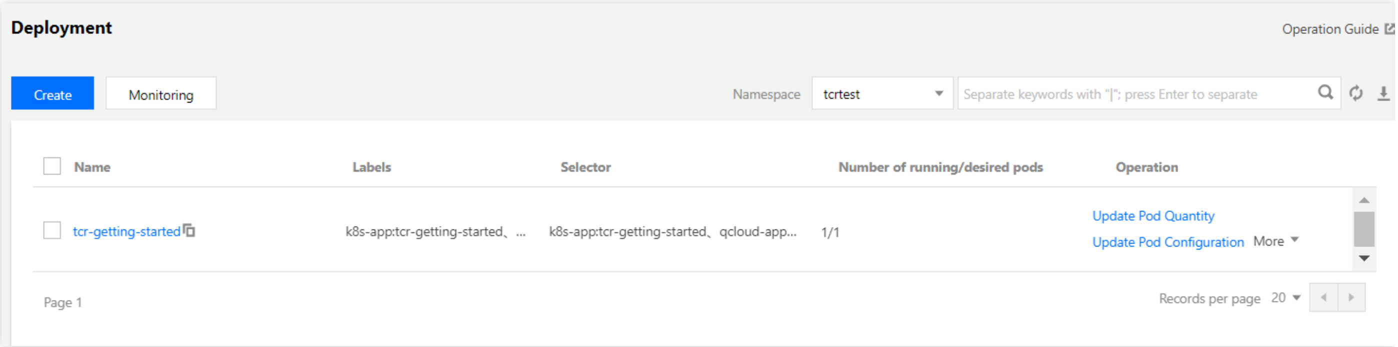


Image Access Credential: Exiting Access Credential | 10001[redacted]40-do

4. After other parameters are set, click **Create Workload** and view the workload deployment progress.
   After the workload is deployed, "Number of Running/Desired Pods" for the workload becomes "1/1" on the

Deployment page, as shown in the figure below:



Namespace tcrtest

	Name	Labels	Selector	Number of running/desired pods	Operation
□	tcr-getting-started	k8s-app:tcr-getting-started、...	k8s-app:tcr-getting-started、qcloud-app...	1/1	Update Pod Quantity Update Pod Configuration More

Auto Scaling

Automatic Scaling Basic Operations

Last updated：2023-02-02 17:05:22

Overview

Horizontal Pod Autoscaler (HPA) can automatically scale the number of Pods for services according to the average CPU utilization of target Pods and other metrics. This document describes how to implement Pod autoscaling via Tencent Cloud TKE console.

How it Works

The HPA backend components pull monitoring metrics of containers and Pods from Tencent Cloud's Cloud Monitor every 15 seconds and calculate the desired number of replicas based on the current monitoring data, the current number of replicas, and the desired value of the metrics. When there is a gap between the desired number and the actual number of replicas, HPA will trigger a Deployment to adjust the number of Pod replicas, thereby achieving auto-scaling.

Take CPU utilization as an example. Suppose there are two Pods with an average CPU utilization of 90%, and the target CPU utilization is set to 60% for autoscaling. Then the number of Pods will be automatically adjusted as follows: 90% × 2 / 60% = 3 Pods.

> Note：
> If you set multiple auto scaling metrics, HPA will separately calculate the target numbers of replicas according to each metric and then take the maximum number to use for auto scaling.

Notes

- If you choose **CPU utilization (by request)** as the metric type, a CPU request must be set for the container.
- Set reasonable targets for the policy metrics. For example, set 70% for containers and applications and leave 30%.
- Keep Pods and nodes healthy; avoid frequently recreating Pods.
- Ensure that the load balancer works stably.

- If the gap between the actual number and desired number of replicas is smaller than 10%, HPA will not adjust the number of replicas.
- If the value of Deployment.spec.replicas corresponding to the service is 0, HPA will not work.
- If multiple HPAs are bound to a single Deployment, the HPAs will take effect simultaneously, which will cause workload replicas to be repeatedly scaled.

Prerequisites

- You have registered a Tencent Cloud account.
- You have logged in to the TKE console.
- You have created a cluster. For more information, see Creating a Cluster.

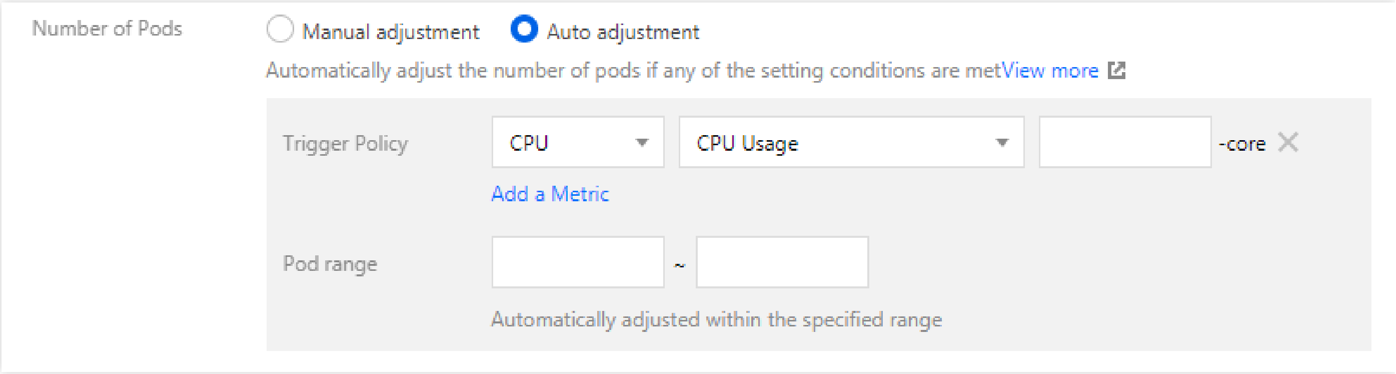
Directions

Enabling Auto Scaling

You can enable auto scaling in one of the following ways.

Setting auto-adjustment of the number of Pods

1. On the **Cluster Management** page, click the cluster ID for which a scaling group is to be created.
2. Select **Workload** > **Deployment**. On the **Deployment** page, click **Create**.
3. On the **Create Deployment** page, select **Auto adjustment** for the number of Pods as shown below:



Number of Pods ○ Manual adjustment ● Auto adjustment

Automatically adjust the number of pods if any of the setting conditions are metView more

Trigger Policy: CPU | CPU Usage | -core

Add a Metric

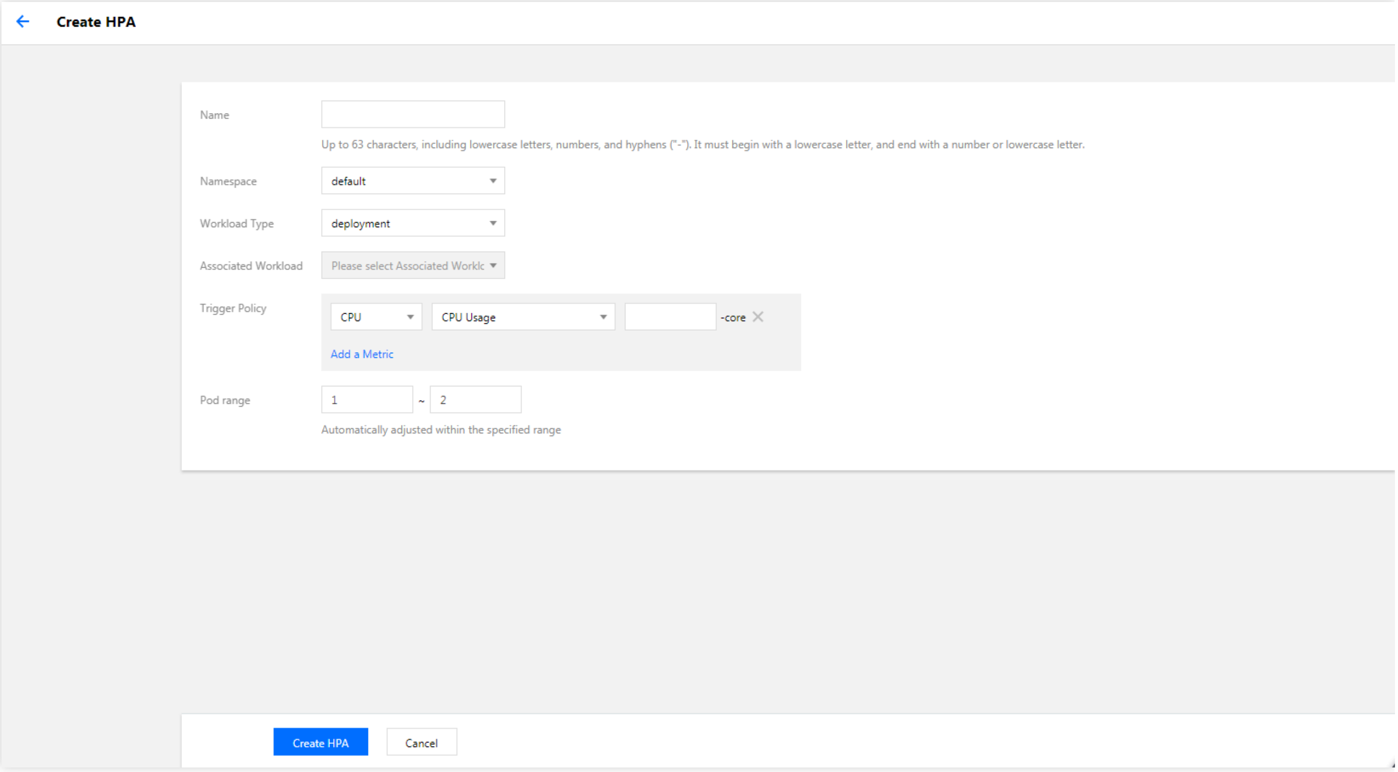
Pod range: ~

Automatically adjusted within the specified range

- **Trigger Policy**: the policy metrics that trigger the auto-scaling. For details, see Metric Type.
- **Number of Pods**: enter the minimum and maximum numbers according to your needs. The number of Pods will be auto-adjusted within the range.

Creating auto-scaling group

1. On the **Cluster Management** page, click the cluster ID for which a scaling group is to be created.
2. Select **Auto Scaling** > **HorizontalPodAutoscaler**. On the **HorizontalPodAutoscaler** page, and click **Create**.
3. On the **Create HPA** page, configure the HPA as needed.



Name
Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Namespace: default

Workload Type: deployment

Associated Workload: Please select Associated Worklc

Trigger Policy: CPU | CPU Usage | -core

Pod range: 1 ~ 2
Automatically adjusted within the specified range

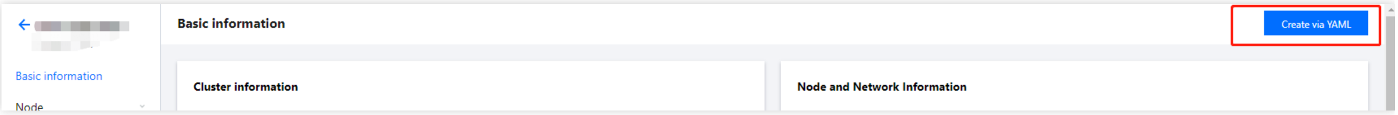
- **Name**: Enter the name of the auto scaling group to be created.
- **Namespace**: select based on your needs.
- **Workload Type**: select based on your needs.
- **Associated Workload**: select based on your needs. The value cannot be empty.
- **Trigger Policy**: the policy metrics that trigger the auto-scaling. For details, see Metric Type.
- **Number of Pods**: enter the minimum and maximum numbers according to your needs. The number of Pods will be auto-adjusted within the range.

4. Click **Create HPA**.

Creating using YAML

1. On the **Cluster Management** page, click the cluster ID for which a scaling group is to be created.

2. On the cluster basic information page, click **Creating using YAML** in the top right corner.



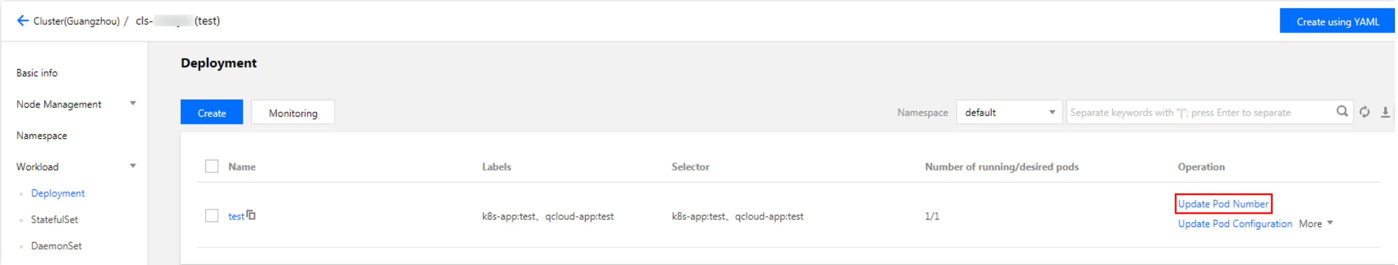
3. Edit the content according to your needs and click **Complete** to create the HPA.

Updating Auto Scaling Rules

You can update auto scaling rules in one of the following ways.

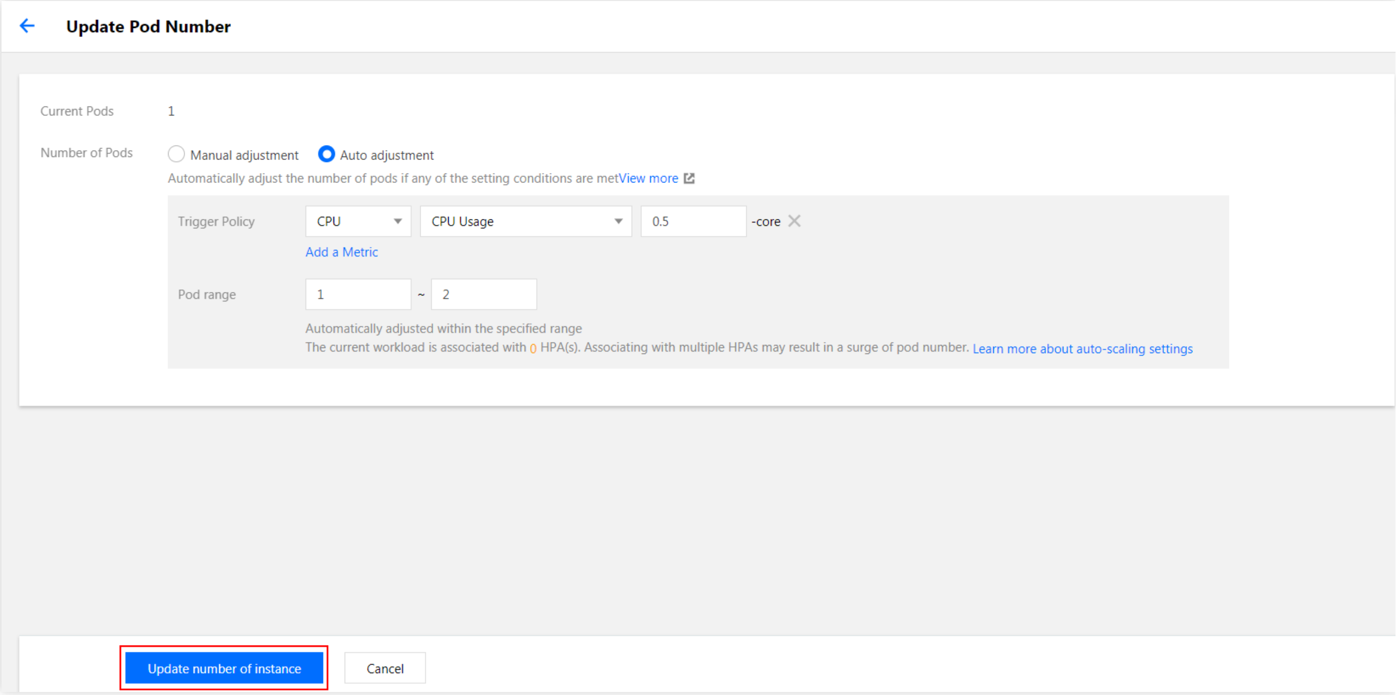
Updating the Pod Quantity

1. On the **Cluster Management** page, click the target cluster ID.
2. Select **Workload** > **Deployment** to enter the **Deployment** page and click **Update Pod Number**.



Name	Labels	Selector	Number of running/desired pods	Operation
test	k8s-app:test、qcloud-app:test	k8s-app:test、qcloud-app:test	1/1	Update Pod Number Update Pod Configuration More

3. On the **Update Pod Number** page, select **Auto adjustment** and set parameters as needed.



Current Pods 1

Number of Pods ○ Manual adjustment ● Auto adjustment

Automatically adjust the number of pods if any of the setting conditions are metView more

Trigger Policy CPU CPU Usage 0.5 -core

Add a Metric

Pod range 1 ~ 2

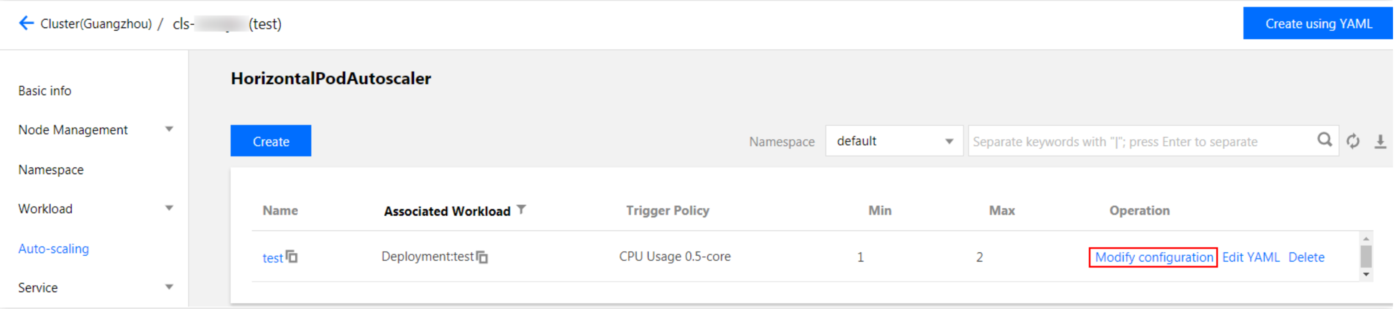
Automatically adjusted within the specified range

The current workload is associated with 0 HPA(s). Associating with multiple HPAs may result in a surge of pod number. Learn more about auto-scaling settings

4. Click **Update number of instance**.

Modifying HPA Configuration

1. On the **Cluster Management** page, click the cluster ID for which a scaling group is to be created.
2. Select **Auto Scaling** > **HorizontalPodAutoscaler**. On the **HorizontalPodAutoscaler** page, click **Modify configuration** in the **Operation** column of the HPA whose configuration is to be updated.

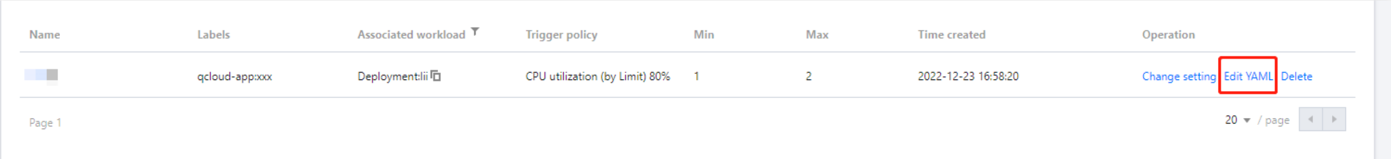


Name	Associated Workload	Trigger Policy	Min	Max	Operation
test	Deployment:test	CPU Usage 0.5-core	1	2	Modify configuration Edit YAML Delete

3. On the **Update Configuration** page, change the settings according to your needs and click **Update HPA**.

Editing YAML

1. On the **Cluster Management** page, click the cluster ID for which a scaling group is to be created.
2. Select **Auto Scaling** > **HorizontalPodAutoscaler**. On the **HorizontalPodAutoscaler** page, click **Edit YAML** in the **Operation** column of the HPA whose configuration is to be updated.



Name	Labels	Associated workload	Trigger policy	Min	Max	Time created	Operation
[redacted]	qcloud-app:xxx	Deployment:lii	CPU utilization (by Limit) 80%	1	2	2022-12-23 16:58:20	Change setting Edit YAML Delete

3. On the **Edit YAML** page, edit parameters as needed and click **Complete**.

Metric Type

For more information on metrics and types, see HPA Metrics.

HPA Metrics

Last updated：2023-03-30 18:26:22

Horizontal Pod Autoscaler (HPA) can automatically scale the number of Pods for services according to the average CPU utilization of target Pods and other metrics. You can set auto-scaling triggering metrics in the console including CPU, memory, disk, network, and GPU metrics. You can also use these metrics when creating and editing HPAs with YAML files. This document provides an example of configuring a YAML file.

Autoscaling Metrics

The following tables list the details of the autoscaling metrics:

Note

Each variable under `metricName` has its own unit which is listed in the default unit column. You can omit such units when compiling the YAML file.

CPU metrics

Metric Name in the Console	Unit in the Console	Remarks	Type	metricName	Default Unit
CPU Usage	Core	Number of CPU cores used by the Pod	Pods	k8s_pod_cpu_core_used	Core
CPU Utilization (per node)	%	Percentage of total CPU of the node used by the Pod	Pods	k8s_pod_rate_cpu_core_used_node	%
CPU Utilization (per Request)	%	Ratio of the total number of CPU cores used by the Pod and the value of Request specified by the container in the Pod	Pods	k8s_pod_rate_cpu_core_used_request	%
CPU	%	Ratio of the total	Pods	k8s_pod_rate_cpu_core_used_limit	%

Utilization (per Limit)		number of CPU cores used by the Pod and the sum of Limit specified by the container in the Pod			

Disk metrics

Metric Name in the Console	Unit in the Console	Remarks	Type	metricName	Default Unit
Disk Write Traffic	KB/s	Pod's disk write rate	Pods	k8s_pod_fs_write_bytes	B/s
Disk Read Traffic	KB/s	Pod's disk read rate	Pods	k8s_pod_fs_read_bytes	B/s
Disk Read IOPS	Times/s	Number of I/O times Pod reads data from disk	Pods	k8s_pod_fs_read_times	Times/s
Disk Write IOPS	Times/s	Number of I/O times Pod writes data to disk	Pods	k8s_pod_fs_write_times	Times/s

Network metrics

Metric Name in the Console	Unit in the Console	Remarks	Type	metricName	Default Unit
Network bandwidth in	Mbps	Sum of inbound bandwidth of all containers per Pod	Pods	k8s_pod_network_receive_bytes_bw	Bps
Network bandwidth out	Mbps	Sum of outbound bandwidth of	Pods	k8s_pod_network_transmit_bytes_bw	Bps

		all containers per Pod			
Network traffic in	KB	Sum of inbound traffic of all containers per Pod	Pods	k8s_pod_network_receive_bytes	B
Network traffic out	KB	Sum of outbound traffic of all containers per Pod	Pods	k8s_pod_network_transmit_bytes	B
Network packets in	Count/s	Sum of inbound packets of all containers per Pod	Pods	k8s_pod_network_receive_packets	Count/s
Network packets out	Count/s	Sum of outbound packets of all containers per Pod	Pods	k8s_pod_network_transmit_packets	Count/s

Memory metrics

Metric Name in the Console	Unit in the Console	Remarks	Type	metricName	Default Unit
Memory Usage	MiB	Amount of memory used by the Pod	Pods	k8s_pod_mem_usage_bytes	B
Memory Usage (excluding cache)	MiB	Pod memory usage, excluding cache	Pods	k8s_pod_mem_no_cache_bytes	B
Memory Utilization (per node)	%	Percentage of total memory of the node	Pods	k8s_pod_rate_mem_usage_node	%

		used by the Pod			
Memory Utilization (per node, excluding cache)	%	Percentage of total memory of the node used by the Pod, excluding cache	Pods	k8s_pod_rate_mem_no_cache_node	%
Memory Utilization (per Request)	%	Percentage of total memory of the Request used by the Pod	Pods	k8s_pod_rate_mem_usage_request	%
Memory Utilization (per Request, excluding cache)	%	Percentage of total memory of the Request used by the Pod, excluding cache	Pods	k8s_pod_rate_mem_no_cache_request	%
Memory Utilization (per Limit)	%	Percentage of Pod memory usage to the Limit value	Pods	k8s_pod_rate_mem_usage_limit	%
Memory Utilization (per Limit, excluding cache)	%	Percentage of Pod memory usage to the Limit value, excluding cache	Pods	k8s_pod_rate_mem_no_cache_limit	%

GPU

Note

The following GPU-related triggering metrics can only be used in TKE Serverless clusters.

Metric Name in the Console	Unit in the Console	Remarks	Type	metricName	Default Unit
GPU Usage	CUDA Core	Pod GPU usage	Pods	k8s_pod_gpu_used	CUDA Core

GPU Applications	CUDA Core	Pod GPU applications	Pods	k8s_pod_gpu_request	CUDA Core
GPU Utilization (per Request)	%	Percentage of GPU usage to the Request value	Pods	k8s_pod_rate_gpu_used_request	%
GPU Utilization (per node)	%	GPU usage percentage in the node	Pods	k8s_pod_rate_gpu_used_node	%
GPU Memory Usage	MiB	Pod GPU memory usage	Pods	k8s_pod_gpu_memory_used_bytes	B
GPU Memory Applications	MiB	Pod GPU memory applications	Pods	k8s_pod_gpu_memory_request_bytes	B
GPU Memory Utilization (per Request)	%	Percentage of GPU memory usage to the Request value	Pods	k8s_pod_rate_gpu_memory_used_request	%
GPU Memory Utilization (per node)	%	GPU memory usage percentage in the node	Pods	k8s_pod_rate_gpu_memory_used_node	%

Creating and Editing an HPA by Using a YAML File

You can create and edit an HPA by using a YAML file. The following example shows a configuration file that defines an HPA named "example". The HPA enables the system to trigger HPA for 1 or 2 Pods when the CPU usage reaches 1.

Note

TKE is compatible with the native resource types.

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: example
  namespace: default
  labels:
    qcloud-app: example
spec:
  minReplicas: 1
  maxReplicas: 2
  metrics:
  - type: Pods# Support using Resource
    pods:
      metricName: k8s_pod_cpu_core_used
      targetAverageValue: "1"
  scaleTargetRef:
    apiVersion: apps/v1beta2
    kind: Deployment
    name: nginx
```

Configuration

ConfigMap Management

Last updated：2023-02-02 17:05:22

Overview

ConfigMap allows you to decouple configuration artifacts from images to ensure that the application is more portable. ConfigMap is a key-value pair. You can create a corresponding ConfigMap object using kubectl in the console, and use a ConfigMap by mounting a volume, through environment variables, or in the container's run command.

Using the Console

Creating a ConfigMap

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to open the TKE cluster list page.
3. Click the ID of the cluster where ConfigMap needs to be created to go to the cluster management page.
4. Select **Configuration Management** > **ConfigMap** to go to the ConfigMap information page.
5. Click **Create** to go to the **Create ConfigMap** page.
6. Set the ConfigMap parameters based on actual needs. Key parameters are as follows:

- Name: customize the name of the container in the Pod.
- Namespace: Select the namespace type and set the variable name and value based on actual needs.
- Content: Add the variable name and variable value.

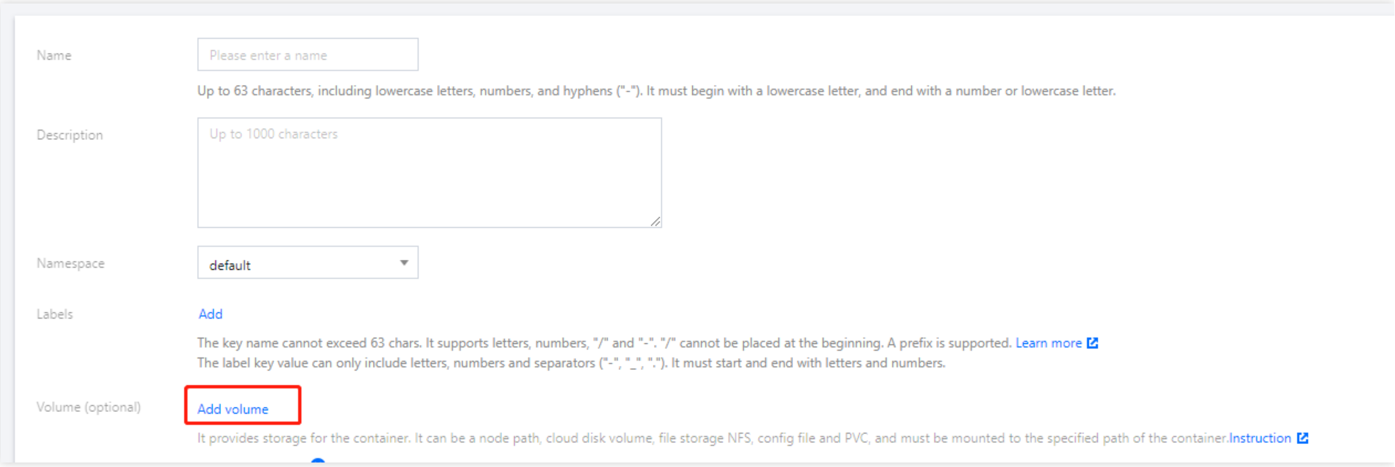
7. Click **Create ConfigMap**.

Using a ConfigMap

Method 1: Using the ConfigMap Type for a Volume

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to open the TKE cluster list page.
3. Click the ID of the cluster where Workload needs to be deployed to go to the cluster management page.
4. Under **Workload**, select any workload type to go to the relevant information page. For example, select **Workload** > **DaemonSet** to go to the DaemonSet information page.
5. Click **Create** to go to the **Create DaemonSet** page.

6. Set the workload name, namespace and other information as instructed. In **Volume**, click **Add Volume**.



Name: Please enter a name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Description: Up to 1000 characters

Namespace: default

Labels: Add

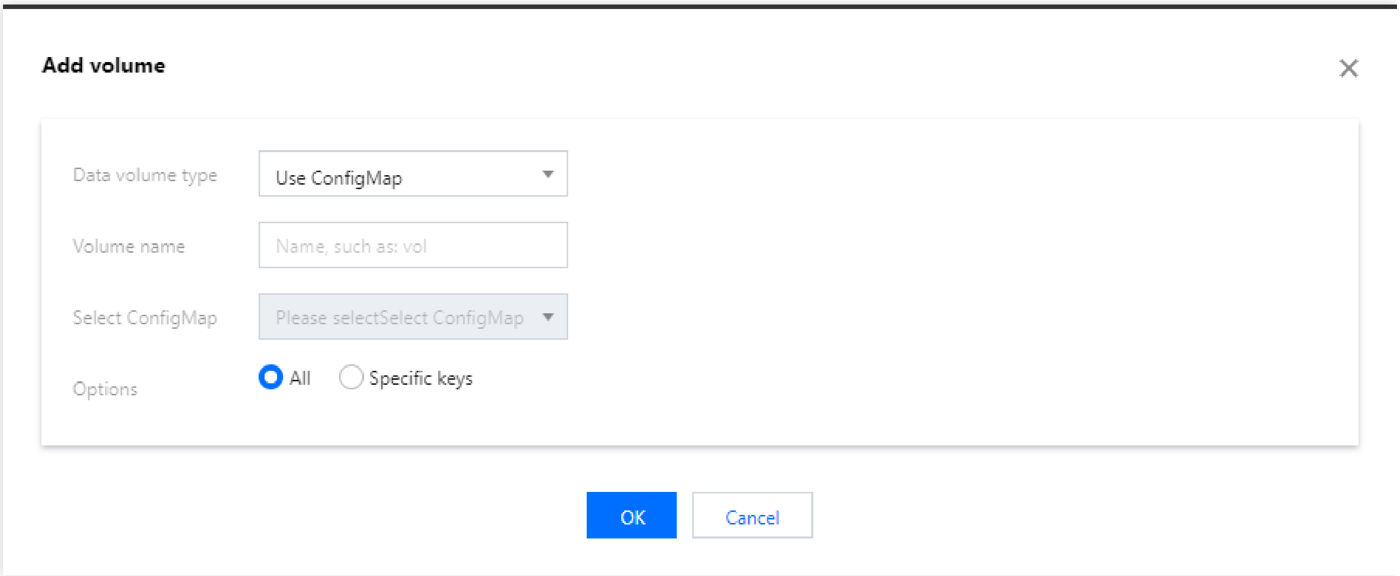
The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more

The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Volume (optional): Add volume

It provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container.Instruction

7. In the **Add volume** pop-up window, configure the mounting point and click **OK**.
   Set **Data volume type** to **Use ConfigMap**, enter the volume name, and click the **Select ConfigMap** drop-down list to select an option.



Add volume

Data volume type: Use ConfigMap

Volume name: Name, such as: vol

Select ConfigMap: Please selectSelect ConfigMap

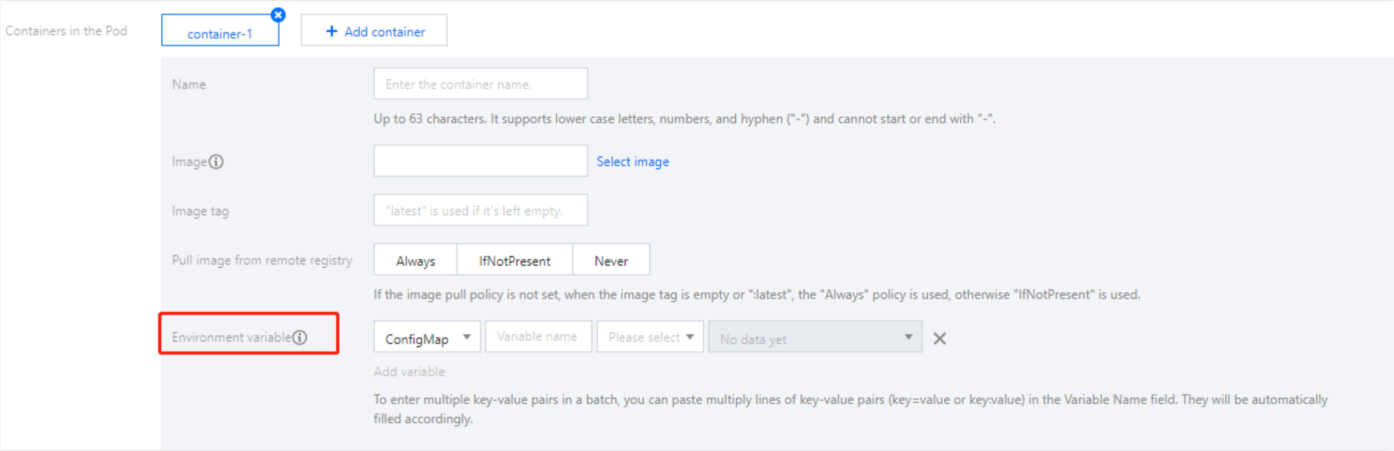
Options: ● All ○ Specific keys

- **Data volume type**: Select **Use ConfigMap**.
- **Volume name**: Enter a custom name.
- **Select ConfigMap**: Select as needed.
- **Options**: **All** and **Specific keys** are available.
- **Items**: if you select **Specific keys**, you can mount to a specific path by adding an item. For example, if the mounting point is /data/config, and the file name is filename, the value of the key-value pair will be stored under /data/config/filename.

8. Click **OK**. Click **Create Workload**.

Method 2: Using the ConfigMap Type for an Environmental Variable

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to open the TKE cluster list page.
3. Click the ID of the cluster where Workload needs to be deployed to go to the cluster management page.
4. Under **Workload**, select any workload type to go to the relevant information page. For example, select **Workload** > **DaemonSet** to go to the DaemonSet information page.
5. Click **Create** to go to the **Create DaemonSet** page.
6. Set the workload name, namespace and other information as instructed. In **Environment Variable** under **Containers in the Pod**, click **Add variable**.



Containers in the Pod: container-1 | + Add container

Name: Enter the container name.

Up to 63 characters. It supports lower case letters, numbers, and hyphen ("-") and cannot start or end with "-".

Image: Select image

Image tag: "latest" is used if it's left empty.

Pull image from remote registry: Always | IfNotPresent | Never

If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.

Environment variable: ConfigMap | Variable name | Please select | No data yet ✕

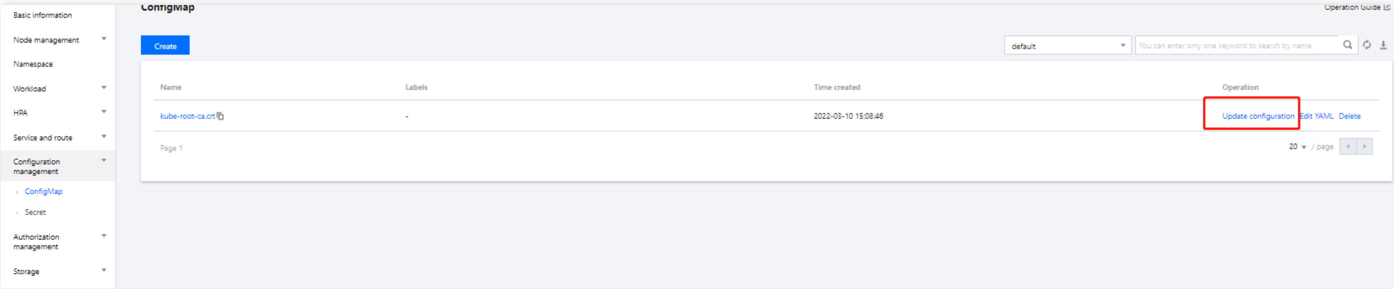
Add variable

To enter multiple key-value pairs in a batch, you can paste multiply lines of key-value pairs (key=value or key:value) in the Variable Name field. They will be automatically filled accordingly.

7. Select "ConfigMap" for the environment variable and select the resource based on actual needs.
8. Click **Create Workload** to complete the process.

Updating a ConfigMap

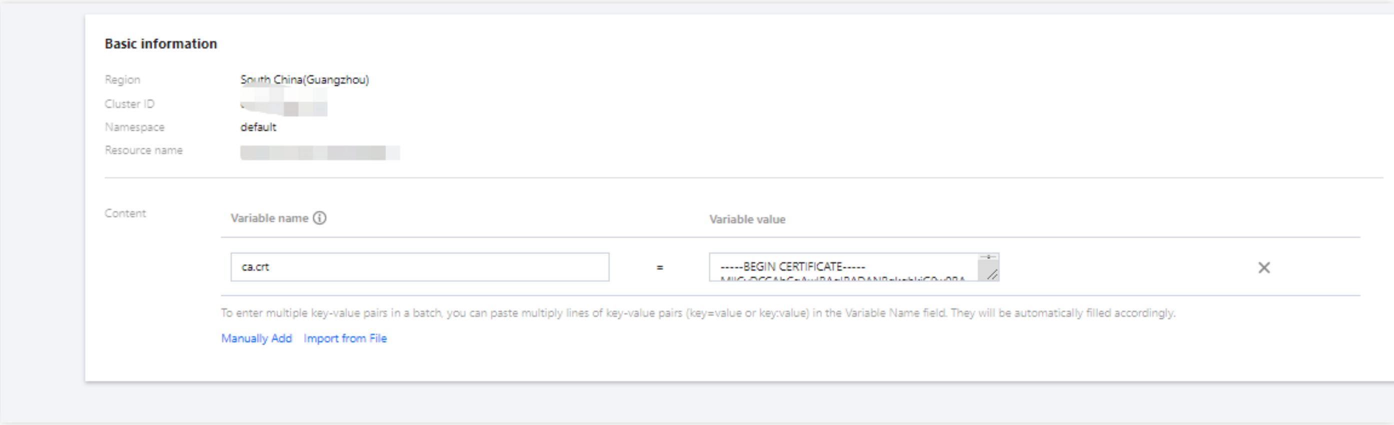
1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to open the TKE cluster list page.
3. Click the ID of the cluster where ConfigMap needs to be updated to go to the cluster management page.
4. Select **Configuration Management** > **ConfigMap** to go to the ConfigMap information page.
5. Click **Update configuration** in the **Operation** column of the ConfigMap whose configuration needs to be updated.



Name	Labels	Time created	Operation
kube-root-ca.crt	-	2022-03-10 15:08:46	Update configuration Edit YAML Delete

Page 1   20 / page

6. On the **Update configuration** page, edit the key-value pair and click **Update ConfigMap**.



Basic information

Region South China(Guangzhou)

Cluster ID [redacted]

Namespace default

Resource name [redacted]

Content

Variable name ⓘ Variable value

ca.crt = -----BEGIN CERTIFICATE-----

To enter multiple key-value pairs in a batch, you can paste multiply lines of key-value pairs (key=value or key:value) in the Variable Name field. They will be automatically filled accordingly.

Via kubectl

YAML sample

```
apiVersion: v1
data:
key1: value1
key2: value2
key3: value3
kind: ConfigMap
metadata:
name: test-config
namespace: default
```

- **data**: The data of ConfigMap presented as key-value.
- **kind**: This identifies the ConfigMap resource type.
- **metadata**: Basic information such as ConfigMap name and Label.
- **metadata.annotations**: An additional description of the ConfigMap. You can set additional enhancements to TKE through this parameter.

Creating a ConfigMap

Method 1: Creating Using the YAML Sample File

1. See the YAML sample to prepare the ConfigMap YAML file.

2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.

3. Run the following command to create the ConfigMap YAML file.

   ```
   kubectl create -f ConfigMap YAML filename
   ```

   For example, to create a ConfigMap YAML file named web.yaml, run the following command:

   ```
   kubectl create -f web.yaml
   ```

4. Run the following command to check whether the Job is successfully created.

   ```
   kubectl get configmap
   ```

   If a message similar to the following is returned, the creation is successful.

   ```
   NAME DATA AGE
   test 2 39d
   test-config 3 18d
   ```

Method 2: Creating by Running a Command

Run the following command to create the ConfigMap in the directory.

```
kubectl create configmap <map-name> <data-source>
```

- <map-name>: Name of the ConfigMap.
- <data-source>: Directory, file, or literal.

For more details about the parameters, see Kubernetes' official document about ConfigMap.

Using a ConfigMap

Method 1: Using the ConfigMap Type for a Volume

Below is a YAML sample:

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
```

```
containers:
- name: nginx
image: nginx:latest
volumeMounts:
name: config-volume
mountPath: /etc/config
volumes:
name: config-volume
configMap:
name: test-config ## Set the ConfigMap source
## items: ## Set the key mounting of the specified ConfigMap
## key: key1 ## Select the specified key
## path: keys ## Mount to the specified subpath
restartPolicy: Never
```

Method 2: Using the ConfigMap Type for an Environmental Variable

Below is a YAML sample:

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
env:
- name: key1
valueFrom:
configMapKeyRef:
name: test-config ## Set the filename of the source ConfigMap
key: test-config.key1 ## Set the value source of the environment variable
restartPolicy: Never
```

Secret Management

Last updated：2023-02-02 17:24:19

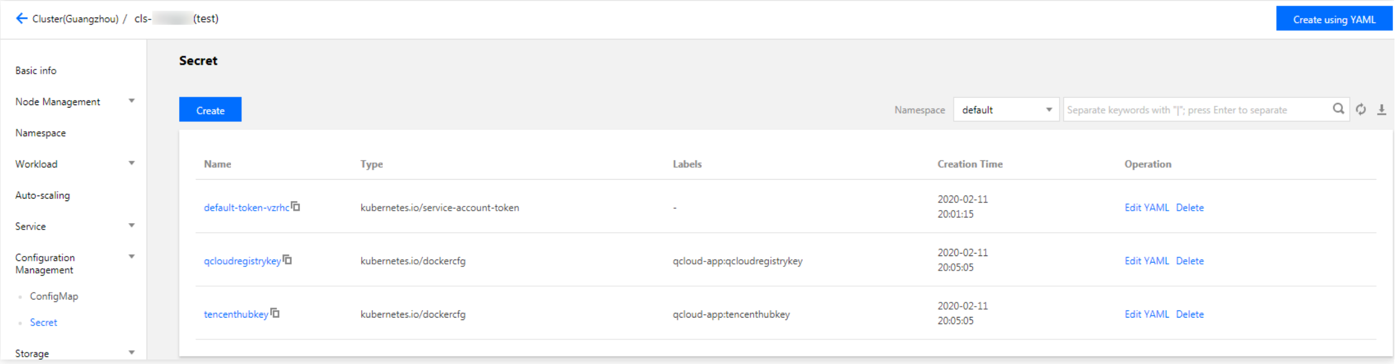
Overview

A secret is a key-value pair that can store sensitive information such as passwords, tokens, and keys to help you lower the risk of information exposure. You can create a secret object using kubectl in the console, and use a secret by mounting a volume, through environment variables, or in the container's run command.

Using the Console

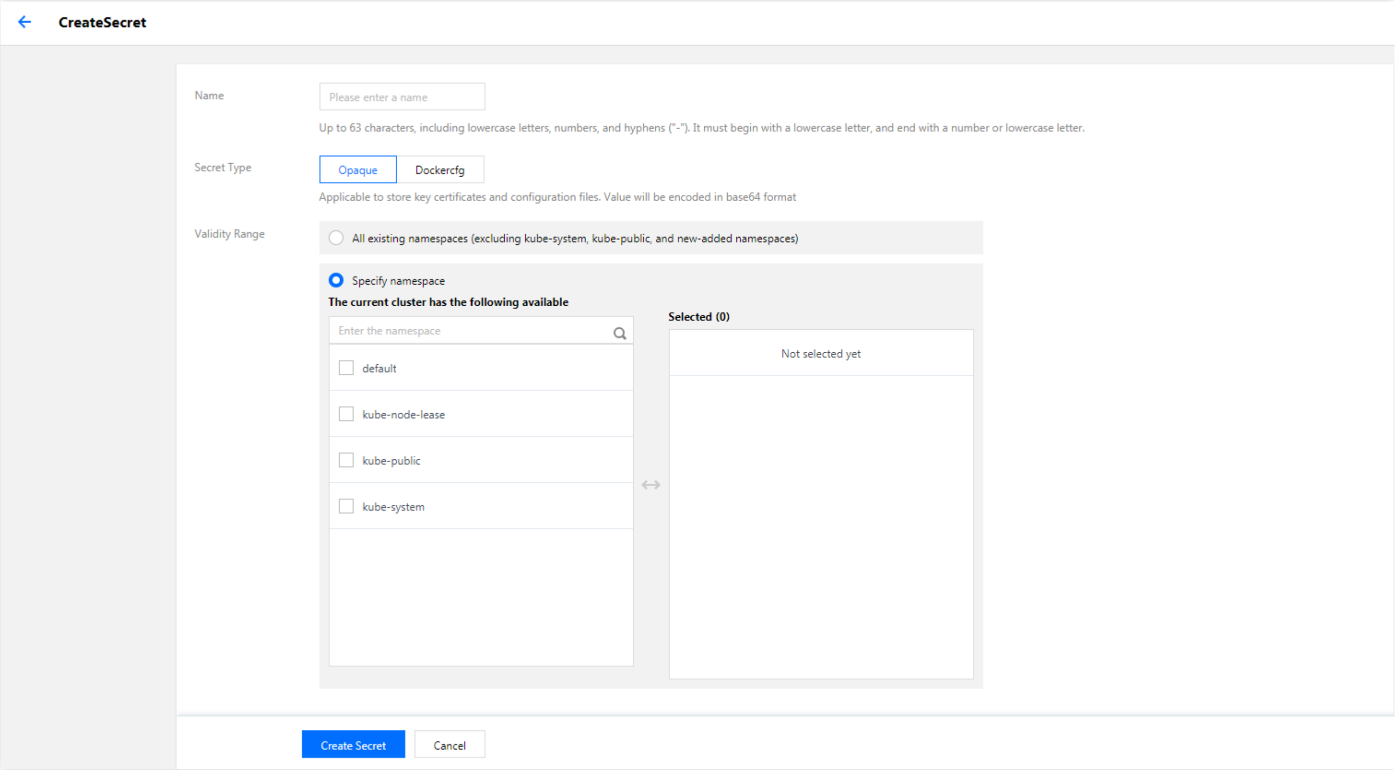
Creating a Secret

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Select the ID of the cluster where you want to create a Secret to enter the cluster management page.
3. Select **Configuration Management** > **Secret** in the left sidebar to go to the Secret page as shown below:



Name	Type	Labels	Creation Time	Operation
default-token-vzrhc	kubernetes.io/service-account-token	-	2020-02-11 20:01:15	Edit YAML Delete
qcloudregistrykey	kubernetes.io/dockercfg	qcloud-app:qcloudregistrykey	2020-02-11 20:05:05	Edit YAML Delete
tencenthubkey	kubernetes.io/dockercfg	qcloud-app:tencenthubkey	2020-02-11 20:05:05	Edit YAML Delete

4. Click **Create**. On the **Create Secret** page, configure parameters as needed.



CreateSecret

Name: Please enter a name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Secret Type: Opaque | Dockercfg

Applicable to store key certificates and configuration files. Value will be encoded in base64 format

Validity Range:

○ All existing namespaces (excluding kube-system, kube-public, and new-added namespaces)

● Specify namespace

The current cluster has the following available

Enter the namespace

□ default

□ kube-node-lease

□ kube-public

□ kube-system

Selected (0)

Not selected yet

Create Secret | Cancel

- **Name**: enter a name.
- **Secret Type**: select **Opaque** or **Dockercfg** as needed.
  - **Opaque**: suitable for storing key certificates and configuration files. The value will be base64-encoded.
  - **Dockercfg**: suitable for storing the verification information of private Docker Registry.
- **Effective Scope**: please select one from the following two options based on your needs.
  - **All existing namespaces**: excluding kube-system, kube-public, and new namespaces added hereafter.
  - **Specific namespaces**: you can specify one or more available namespaces in the current cluster.
- **Content**: make configuration according to your secret type.
  - If the secret type is **Opaque**: set the variable name and value as needed.
  - If the secret type is **Dockercfg**:
    - Repository domain name: enter the domain name or IP as applicable.
    - Username: enter the username for the third-party repository according to your needs.
    - Password: enter the login password for the third-party repository according to your needs.

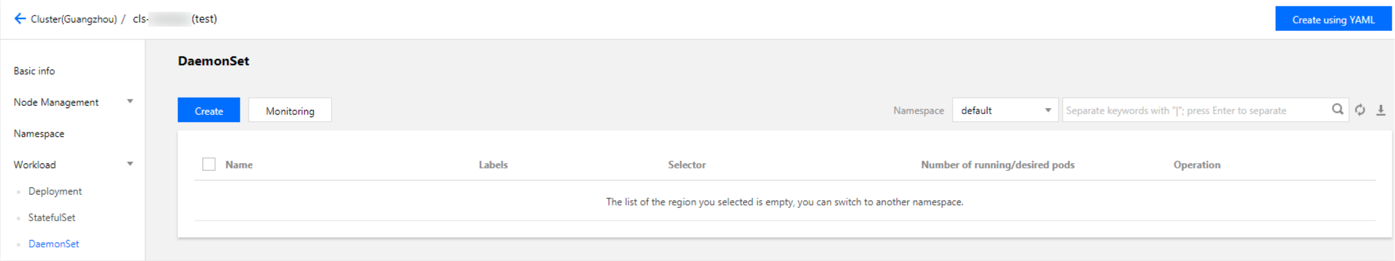
> Note：
>
> If this is the first time you log in to the system, an account will be created and the related information will be written to the `~/.dockercrg` file.

6. Click **Create Secret** to complete the creation.

Using a Secret

Method 1: Using Secret as a volume

1. Log in to the TKE console and select Cluster in the left sidebar.
2. Click the ID of the cluster where you want to deploy the workload to enter the cluster management page.
3. Under **Workload**, select a workload type to go to the corresponding information page.
   For example, select **Workload** > **DaemonSet** to go to the DaemonSet information page. See the figure below:



Cluster(Guangzhou) / cls-[redacted](test)

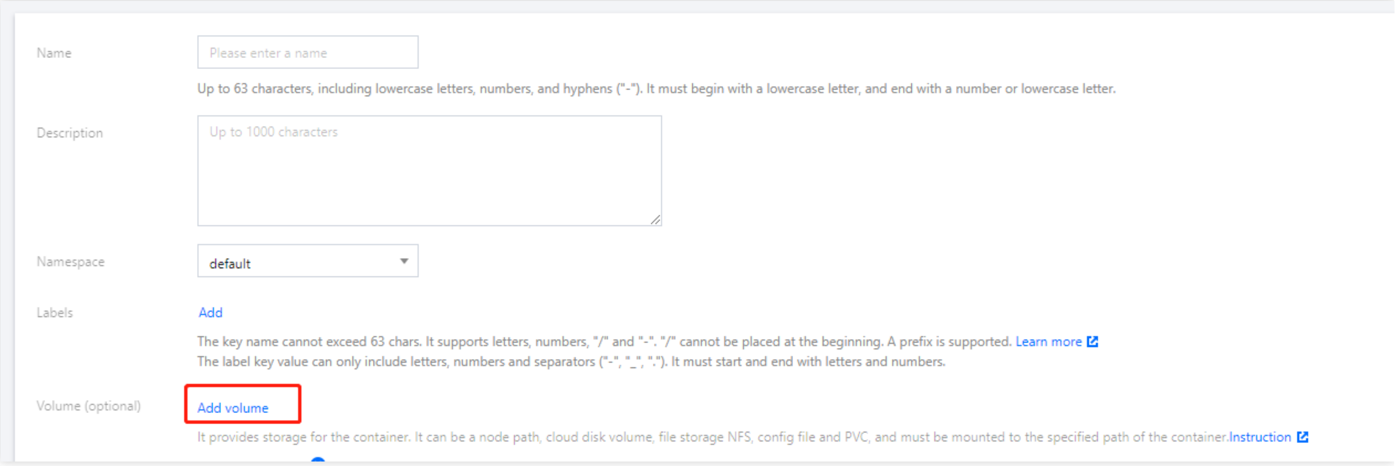
DaemonSet

Namespace: default

Name	Labels	Selector	Number of running/desired pods	Operation

The list of the region you selected is empty, you can switch to another namespace.

4. Click **Create** to open the **Create Workload** page.
5. Set the workload name, namespace and other information as instructed. In **Volume**, click **Add Volume**.



Name: Please enter a name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Description: Up to 1000 characters

Namespace: default

Labels: Add

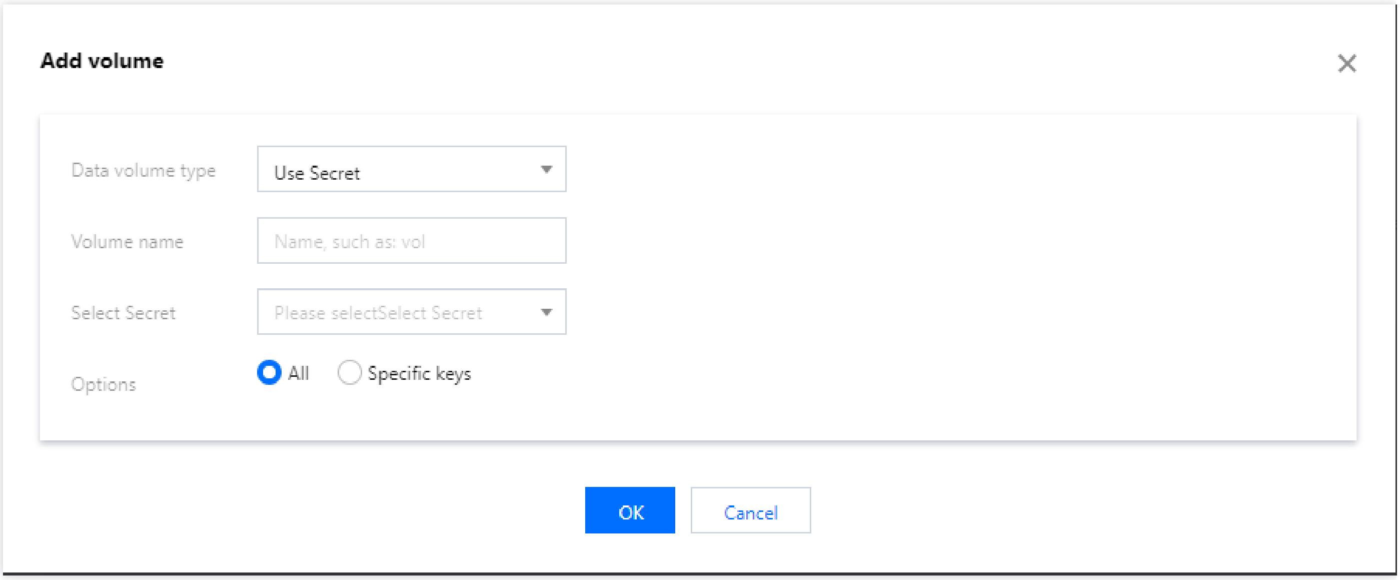
The key name cannot exceed 63 chars. It supports letters, numbers, "/" and "-". "/" cannot be placed at the beginning. A prefix is supported. Learn more

The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

Volume (optional): Add volume

It provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container.Instruction

6. Select **Use Secret** in the drop-down menu, enter a name, and click **Select Secret**.



Add volume

Data volume type: Use Secret

Volume name: Name, such as: vol

Select Secret: Please selectSelect Secret

Options: ● All ○ Specific keys

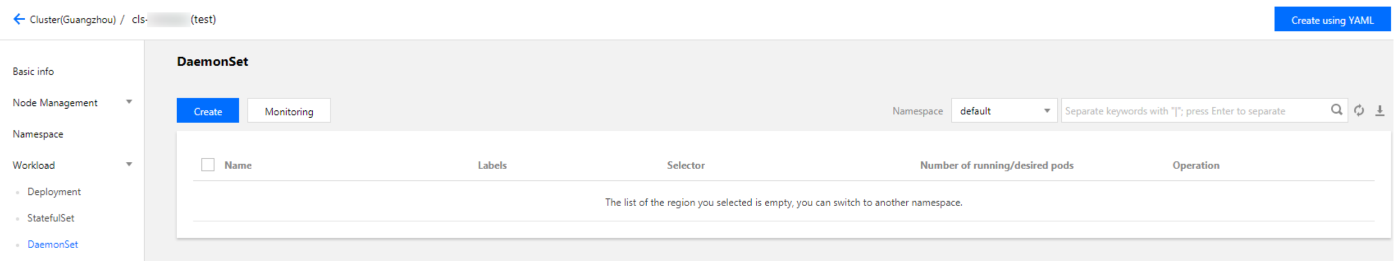
- **Select a secret**: select a Secret as needed.
- **Options**: **All** and **Specific keys** are available.
- **Items**: if you select the **Specific keys** option, you can mount the Secret to a specific path by adding an item. For example, if the mounting point is `/data/config` , the sub-path is `dev` , it will finally be saved under `/data/config/dev` .

8. Click **Create Workload** to complete the process.

Method 2: Using a Secret as an environmental variable

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the ID of the cluster where you want to deploy the workload to enter the cluster management page.
3. Under **Workload**, select a workload type to go to the corresponding information page.

   For example, select **Workload** > **DaemonSet** to go to the DaemonSet information page. See the figure below:



Cluster(Guangzhou) / cls-[redacted](test)

DaemonSet

Namespace: default

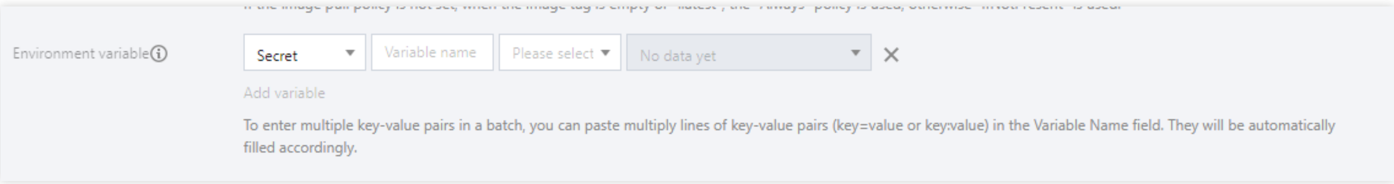
Separate keywords with "|"; press Enter to separate

□ Name	Labels	Selector	Number of running/desired pods	Operation

The list of the region you selected is empty, you can switch to another namespace.

4. Click **Create** to open the **Create Workload** page.

5. Set the workload name, namespace and other information as instructed. In **Environment Variable** under **Containers in the Pod**, select **Secret** for the environment variable and select resources as needed.



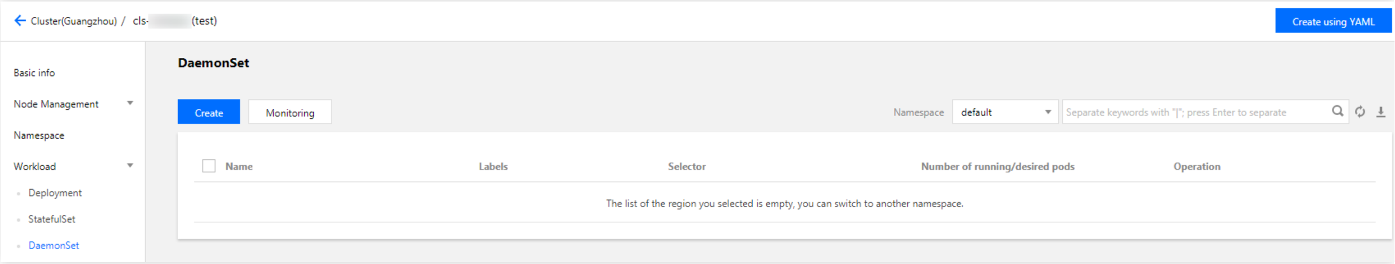
Environment variable: Secret

To enter multiple key-value pairs in a batch, you can paste multiply lines of key-value pairs (key=value or key:value) in the Variable Name field. They will be automatically filled accordingly.

6. Click **Create Workload** to complete the process.

Method 3: Referencing a Secret when using third-party image repositories

1. Log in to the TKE console and select Cluster in the left sidebar.
2. Click the ID of the cluster where you want to deploy the workload to enter the cluster management page.
3. Under **Workload**, select a workload type to go to the corresponding information page.
   For example, select **Workload** > **DaemonSet** to go to the DaemonSet information page. See the figure below:



Cluster(Guangzhou) / cls-[redacted](test)

DaemonSet

Name	Labels	Selector	Number of running/desired pods	Operation

The list of the region you selected is empty, you can switch to another namespace.

4. Click **Create** to open the **Create Workload** page.
5. Set the workload name, namespace and other information as instructed, and select **Image access credential** as needed.
6. Click **Create Workload** to complete the process.

Updating a Secret

1. Log in to the TKE console and select Cluster in the left sidebar.
2. Select the ID of the cluster for which you want to update the YAML to go to the cluster management page.
3. Select **Configuration Management** > **Secret** to go to the Secret information page.
4. In the row of the Secret for which you want to update the YAML, click **Edit YAML** to go to the Secret updating page.
5. On the **Update Secret** page, edit the YAML and click **Complete**.

> Note：
>
> To modify key-values, edit the parameter values of data in YAML and click **Finish** to complete the update.

Via kubectl

Creating a Secret

Method 1: Creating a Secret with a specified file

1. Run the following commands to obtain the username and password of the Pod.

```
$ echo -n 'username' > ./username.txt
$ echo -n 'password' > ./password.txt
```

2. Run the following kubectl command to create a Secret.

```
$ kubectl create secret generic test-secret --from-file=./username.txt --from-file=./password.txt
secret "testSecret" created
```

3. Run the following command to view the details about the Secret.

```
kubectl describe secrets/ test-secret
```

Method 2: Manually creating a Secret with a YAML file

> Note：
>
> To manually create a Secret using YAML, you need to Base64-encode the data of the Secret in advance.

```
apiVersion: v1
kind: Secret
metadata:
name: test-secret
type: Opaque
data:
username: dXNlcm5hbWU= ## Generated by echo -n 'username' | base64
password: cGFzc3dvcmQ= ## Generated by echo -n 'password' | base64
```

Using a Secret

Method 1: Using a Secret as a volume

Below is a YAML sample:

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
volumeMounts:
name: secret-volume
mountPath: /etc/config
volumes:
name: secret-volume
secret:
name: test-secret ## Set the Secret source
## items: ## Set the key mounting of the specified Secret
## key: username ## Select the specified key
## path: group/user ## Mount to the specified subpath
## mode: 256 ## Set file permission
restartPolicy: Never
```

Method 2: Using a Secret as an environmental variable

Below is a YAML sample:

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
env:
- name: SECRET_USERNAME
valueFrom:
secretKeyRef:
name: test-secret ## Set the filename of the source Secret
key: username ## Set the value source of the environment variable
restartPolicy: Never
```

Method 3: Referencing a Secret when using third-party image repositories

Below is a YAML sample:

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
imagePullSecrets:
- name: test-secret ## Set the filename of the source Secret
restartPolicy: Never
```

Service Management Overview

Last updated：2023-05-06 17:36:46

Basic Concepts of Service

You can deploy various containers in Kubernetes. Some of them provide layer-7 network services externally over HTTP or HTTPS, and others provide layer-4 network services over TCP or UDP. Service resources defined in Kubernetes are used to manage access to layer-4 network services in a cluster.

You can specify the Service type with Kubernetes `ServiceType` , which defaults to `ClusterIP` .
`ServiceType` values and their behaviors are described as follows:

Value	Description
ClusterIP	Exposes the Service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster. This is the default value of `ServiceType` .
NodePort	Exposes the Service on each node's IP at a static port (NodePort). The NodePort Service will be routed to the ClusterIP Service that will be created automatically. It can be accessed from outside the cluster through the `<NodeIP>:<NodePort>` request. We recommend you not provide external and even public network services directly through cluster nodes in the production environment, as using NodePort will expose cluster nodes directly to attacks. Generally, cluster nodes are dynamic and can be added or removed, and using NodePort will couple them with addresses providing external services.
LoadBalancer	Exposes the Service externally or privately by using a CLB instance. The CLB can be routed to NodePort or directly forwarded to containers in the VPC-CNI network.

ClusterIP and NodePort Services usually behave in the same way in external clusters or those provided by cloud vendors. A LoadBalancer services is exposed by using a cloud vendor's load balancer and will have additional load balancer capabilities provided by the cloud vendor, for example, control of the network type of the load balancer and adjustment of weights of bound real servers. For more information, see Service Management.

Service Access Methods

You can use the following service access methods provided by TKE based on the above definition of `ServiceTypes` :

Access Method	Service Type	Description
Public network	LoadBalancer	In Loadbalance mode of the Service, public IPs can directly access backend Pods. This method is applicable to web frontend Services. A created Service can be accessed from outside the cluster with the "CLB instance domain name or IP + Service port" or from within the cluster with the "Service name + Service port". Note: The architecture of CLB was upgraded on March 6, 2023. After the upgrade, public network CLB instances deliver services through **domain names**. The **VIP** of a CLB instance is no longer displayed in the console. This is because as service traffic increases, the VIP changes dynamically. For more information, see [March 6, 2023] Notice: Domain Name-Based CLB Available on Public Networks. For CLB users registered after the upgrade, the domain name-based CLB architecture is adopted by default. If your account was registered before the upgrade, you can choose whether to use the original CLB architecture. To use the upgraded architecture, you need to upgrade both CLB and TKE, or else, public network Service/Ingress add-ons will not be properly synchronized in TKE. For information about how to upgrade CLB, see Directions for Upgrading to Domain Name-Based CLB. For information about the upgraded TKE Service/Ingress add-on versions, please submit a ticket.
VPC	LoadBalancer	In Loadbalance mode of the Service, private IPs can directly access backend Pods by specifying the `service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx` annotation. A created Service can be accessed from outside the cluster with the "CLB instance domain name or IP + Service port" or from within the cluster with the "Service name + Service port".
Access through the node port	NodePort	This access method maps node ports to containers and is supported for TCP, UDP, and Ingress. It can be used for customizing upper-layer load balancer forwarding to nodes. A created Service can be accessed with the "CVM instance IP + node port".
Access from within the cluster only	ClusterIP	In ClusterIP mode of the Service, Service IPs are automatically assigned for access from within the cluster. This method can be used for database services such as MySQL, so as to ensure service network isolation. A created Service can be accessed with the "Service name + Service port".

CLB Concepts

How a Service works

In a Tencent Cloud container cluster, the Service Controller add-on syncs your Service resources when you create, modify, or delete Service resources, cluster nodes or service endpoints change, or add-on containers drift or restart. The Service Controller add-on will create CLB resources and configure listeners and real servers based on the description of the Service resource. When you delete the Service resource, it will repossess the CLB resources.

Service lifecycle management

The external service capabilities of a Service rely on the resources provided by the CLB instance. Service resource management matters to a Service, which will use the following labels during the resource lifecycle management:

`tke-createdBy-flag = yes` : Indicates that the resource is created by TKE.

If this label exists, the corresponding resource will be deleted when the Service is terminated.

If this label does not exist, only the listener resources in the CLB instance but not the CLB instance itself will be deleted when the Service is terminated.

`tke-clusterId = <ClusterId>` : Identifies the cluster that uses the resource.

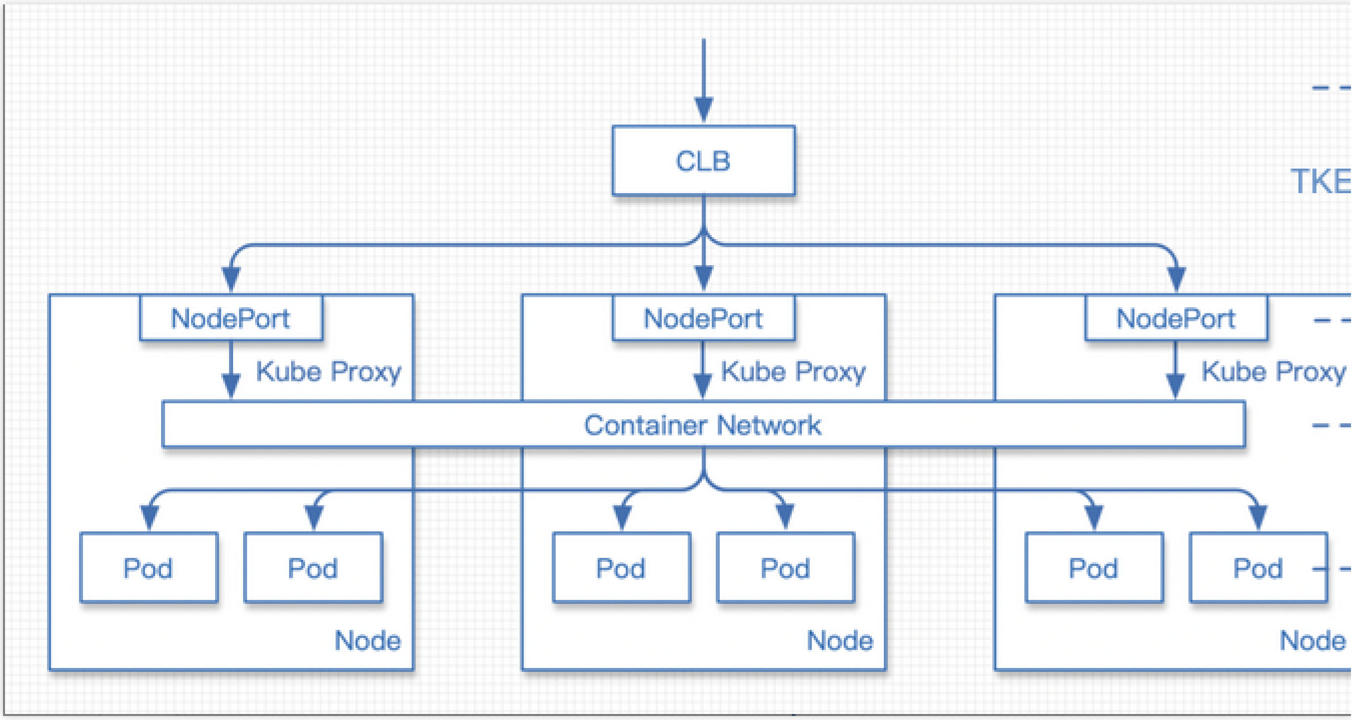
If the `ClusterId` is correct, the corresponding label will be deleted when the Service is terminated.

Note

If you use an existing CLB instance, the Service will only use but not delete the instance.

If deletion protection is enabled or a private connection is used for the CLB, the CLB will not be deleted when services are deleted.

When a Service of the LoadBalancer type is created, the lifecycle of the corresponding CLB instance starts; when the Service is deleted or the CLB instance is recreated, the lifecycle ends. During the lifecycle, the CLB instance will be continuously synchronized based on the description of the Service. **The CLB instance will be recreated or terminated if you switch the network for accessing the Service (public network > VPC, VPC > public network, or between VPC subnets) or replace the CLB instance.** A Service of the LoadBalancer type works as follows:



Service precautions

Each Service has the `.spec.externalTrafficPolicy` field. The kube-proxy filters the endpoints it routes to based on the `.spec.externalTrafficPolicy` setting. When the field is set to `Local` , Kubernetes considers only a node's local endpoints. When the field is set to `Cluster` (default value), or is not set, Kubernetes considers all endpoints. For more information, see the Service Internal Traffic Policy Kubernetes document.

If the Service uses the `Local` method, there will be a stream interruption when a Pod is scheduled from a TKE node to a super node, or from a super node to a TKE node, because the Service will select only a local service endpoint.

High-risk operations on a Service

Use a traditional CLB instance (not recommended).

Modify or delete a CLB instance label added by TKE, purchase a new CLB instance, and recover the label.

Rename a CLB listener managed by TKE in the CLB console.

Service features

For more information on Service operations and features, see the following documents:

Basic Features

Service CLB Configuration

Using Existing CLBs

Service Backend Selection

Service Cross-region Binding

Graceful Service Shutdown

Using Services with CLB-to-Pod Direct Access Mode

Multiple Services Sharing a CLB

Service Extension Protocol

Service Annotation

References

For more information, see the Service Kubernetes document.

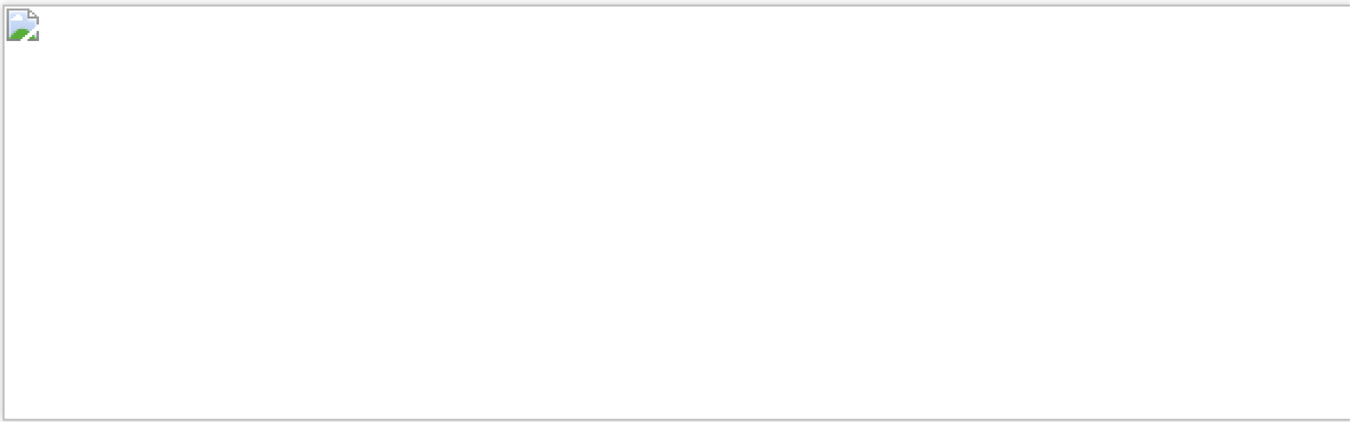
Basic Features

Last updated：2023-03-30 18:26:22

Managing a Service in the Console

Creating a service

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster** page, click the ID of the cluster for which you need to create a Service to go to the cluster management page.
3. Select **Services and Routes** > **Service** to go to the Service management page.



4. Click **Create** to enter the **Create Service** page. Set the Service parameters as needed. Key parameters are as follows:

Service Name: Customize a name.

Namespace: Select a namespace based on your requirements.

Access Settings: Set it as needed and as instructed in Service Access Methods.

(Optional) **Advanced Settings**:

External Traffic Policy:

Cluster: Defaults to averagely forward all Pods of the workload.

Local: Retain the client IP, and ensure that traffic is only forwarded within the node if the access mode is public network, VPC private network (LoadBalancer) and node port (NodePort). If you choose **Local**, the health check for nodes without Pods may fail, raising the risk of unbalanced traffic forwarding

Note

If the Service uses the `Local` method, there will be a stream interruption when a Pod is scheduled from a TKE node to a super node, or from a super node to a TKE node, because the Service will select only a local service endpoint.

Session Affinity: If you want to ensure that connections from a particular client are passed to the same Pod every time, you can set session affinity based on the client IP address by setting the Service's `.spec.sessionAffinity` to ClientIP (the default value is `None`).

Workload binding: Reference an existing workload or customize a label. Then, the Service will select workloads with the label.

Note

To use an existing CLB instance, see Using Existing CLBs.

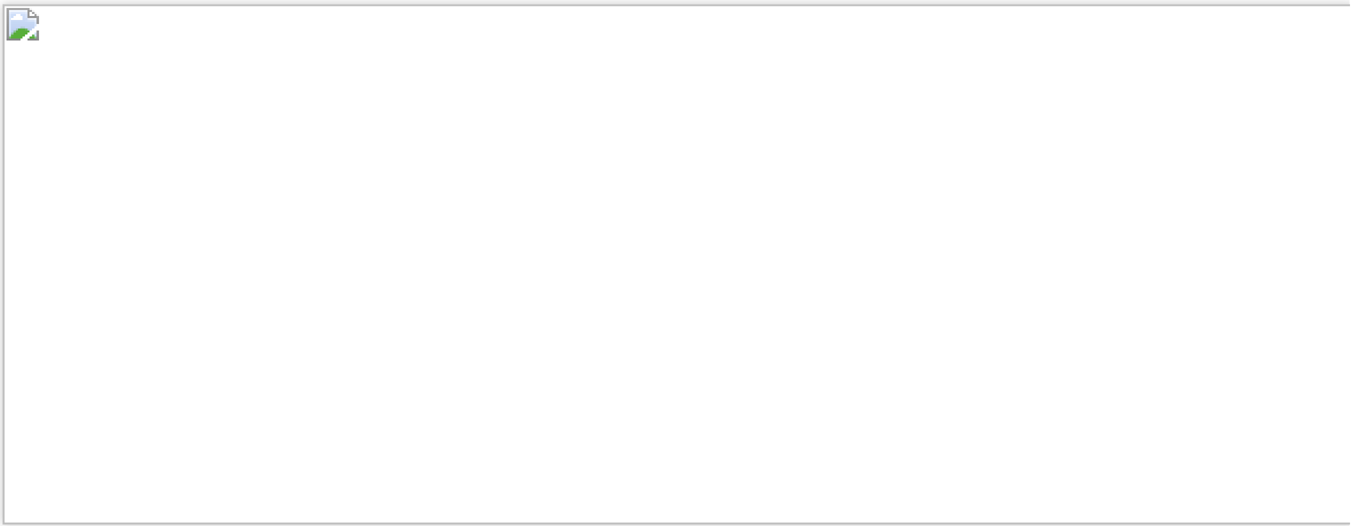
As a layer-4 CLB instance has only **the unique quadruple of CLB VIP, listener protocol, backend RS VIP, and backend RS port** and doesn't contain a CLB listener port, scenarios with different CLB listener ports but the same protocol and RS are not supported. In addition, TKE doesn't support opening different ports of the same protocol for the same business.

5. Click **Create Service**.

Updating a Service

Updating configuration

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the target cluster ID to enter the cluster's basic information page.
3. Select **Services and Routes** > **Service**. On the **Service** page, locate the target service and click **Update configuration** on the right.



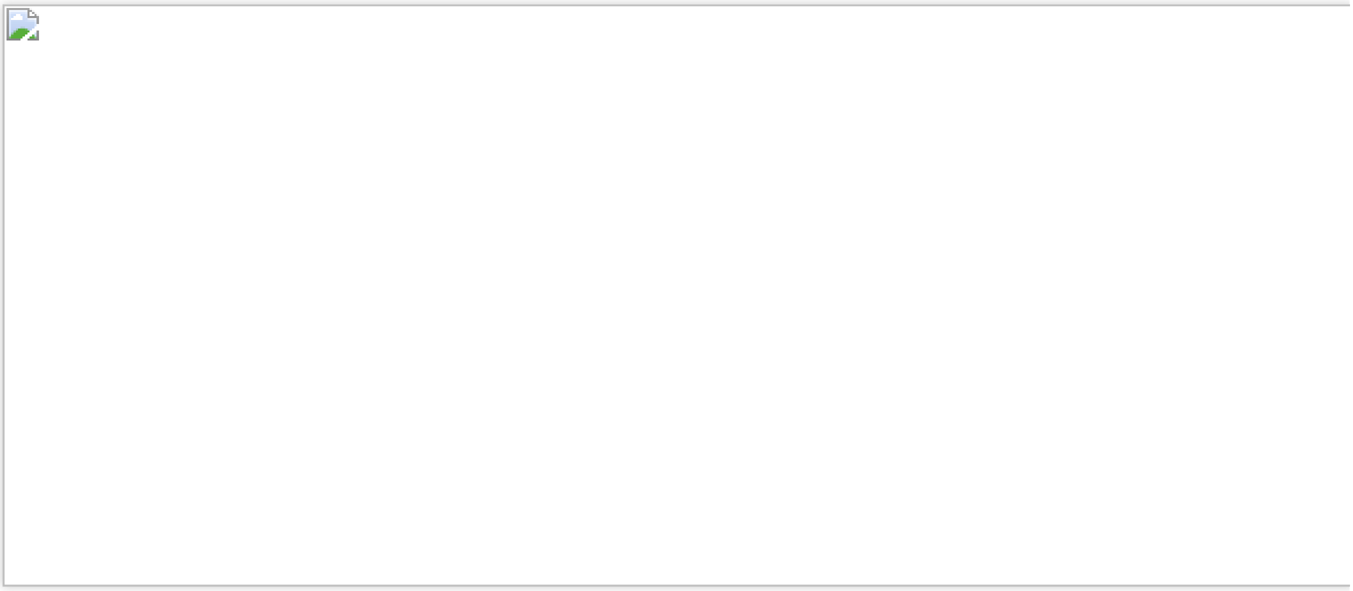
4. On the **Update access method** page, configure the access method as needed.
5. Click **Update access method**.

Editing YAML

1. Select **Services and Routes** > **Service**. On the **Service** page, locate the target service and click **Edit YAML** on the right.
2. On the **Edit YAML** page, edit the YAML and click **Done**.

Deleting a Service

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the target cluster ID to enter the cluster's basic information page.
3. Select **Services and Routes** > **Service**. On the **Service** page, locate the target service and click **Delete** on the right.



Managing a Service Using kubectl

YAML sample

```
kind: Service
apiVersion: v1
metadata:
  ## annotations:
  ## service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx #
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
```

```
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

Note:

kind: Service resource type.

metadata: Basic information such as Service name and label.

metadata.annotations: Additional description of the Service. You can set additional enhancements to TKE through this parameter.

spec.selector: The Service will select workloads with the label in the label selector.

spec.type: Mode for accessing the Service.

ClusterIP: The Service is made public in the cluster for internal access.

NodePort: The node port mapped to the backend Service. External access to the cluster can be implemented through `IP:NodePort` .

LoadBalancer: The Service is made public through the Tencent Cloud CLB instance. A public network CLB instance is created by default, and a private network CLB can be created by specifying annotations.

By default, you can create up to 100 public network or private network CLB instances. If you need more, submit a ticket to increase the quota.

The management and sync of configurations between Service and CLB instances are based on the resource object of the `LoadBalancerResource` type named the CLB ID. Do not perform any operations on this CRD; otherwise, the Service may fail.

ExternalName: The Service is mapped to DNS, which applies to only kube-dns 1.7 or later.

Creating a service

1. Prepare the Service YAML file as instructed in the YAML sample.
2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
3. Run the following command to create the Service YAML file.

```
kubectl create -f Service YAML filename
```

For example, to create a Service YAML file named `my-service.yaml` , run the following command:

```
kubectl create -f my-service.yaml
```

4. Run the following command to check whether the creation is successful:

```
kubectl get services
```

If a message similar to the following is returned, the creation is successful.

```
NAME          TYPE         CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes ClusterIP 172.16.255.1 <none> 443/TCP 38d
```

Updating a Service

Method 1

Run the following command to update a Service:

```
kubectl edit service/[name]
```

Method 2

1. Manually delete the old Service.

2. Run the following command to create a new Service:

```
kubectl create/apply
```

Deleting a Service

Run the following command to delete a Service:

```
kubectl delete service [NAME]
```

Service CLB Configuration

Last updated：2024-08-08 14:50:28

TkeServiceConfig

TkeServiceConfig is a Custom Resource Definition (CRD) provided by TKE. TkeServiceConfig can help you configure LoadBalancer-type Services more flexibly and manage various CLB configurations in them.

Use cases

CLB parameters and features that cannot be defined by Service YAML semantics can be configured through TkeServiceConfig.

Configuration instructions

TkeServiceConfig can help you quickly perform CLB configuration. Through the Service annotation `service.cloud.tencent.com/tke-service-config:<config-name>` , you can specify the target configuration and apply it to the Service.

Note：

TkeServiceConfig resources and the Service need to be in the same namespace.

TkeServiceConfig does not help you directly configure or modify protocols and ports. You need to describe protocols and ports in the configuration in order to deliver the specified configuration to the listener. You can declare multiple sets of listener configurations in a single TkeServiceConfig. Currently, configurations are mainly provided for CLB health check and backend access.

When the protocol and port are specified, the configuration will be accurately delivered to the corresponding listener:

`spec.loadBalancer.l4Listeners.protocol` : Layer-4 protocol

`spec.loadBalancer.l4Listeners.port` : Listening port

Associated Actions Between Service and TkeServiceConfig

1. During the creation of a Loadbalancer-type Service, if you set annotation `service.cloud.tencent.com/tke-service-config-auto: "true"` , `<ServiceName>-auto-service-config` will be automatically created. Alternatively, you can specify your own created TkeServiceConfig through **service.cloud.tencent.com/tke-service-config:<config-name>**. These two annotations cannot be used at the same time, and the manually specified `<config-name>` cannot be suffixed with `-auto-service-config` and `-auto-ingress-config` .

2. The automatically created `TkeServiceConfig` has the following sync behaviors:

When a layer-4 listener is added during Service resource update, if there is no corresponding TkeServiceConfig configuration segment for the listener or forwarding rule, Service-Controller will automatically add the corresponding TkeServiceConfig configuration segment.

When a layer-4 listener is deleted, Service-Controller will automatically delete the corresponding TkeServiceConfig segment.

When Service resources are deleted, the corresponding TkeServiceConfig will also be deleted.

When you modify the default TkeServiceConfig of the Service, the TkeServiceConfig content will also be applied to the CLB.

3. You can also refer to the following complete TkeServiceConfig configuration and create your own desired CLB configuration. Services will import the configuration through the annotation **service.cloud.tencent.com/tke-service-config:<config-name>**.

4. A manually created `TkeServiceConfig` has the following sync behaviors:

When you add a configuration annotation in the Service, the CLB will immediately set synchronization.

When you delete a configuration annotation in the Service, the CLB will remain unchanged.

When you modify the TkeServiceConfig configuration, the CLB of the Service that imported the configuration will set synchronization based on the new TkeServiceConfig.

If the Service listener does not find the corresponding configuration, the listener will not be modified.

If the Service listener finds the corresponding configuration but the configuration does not contain specified attributes, the listener will not be modified.

Complete Configuration Reference

```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
  name: sample # Configuration name
  namespace: default # Configuration namespace
spec:
  loadBalancer:
    l4Listeners: # Layer-4 rule configuration, applicable to Service listener confi
    - protocol: TCP # Layer-4 rule for protocol ports anchoring the Service. Requir
      port: 80 # Required. Value range: 1-65535.
      deregisterTargetRst: true # Optional. Boolean. Bidirectional RST switch. Reco
```

```
        session: # Configuration related to session persistence. Optional.
          enable: true # Indicates whether to enable session persistence. Required. B
          sessionExpireTime: 100 # Session persistence duration. Optional. Default va
        healthCheck: # Configuration related to health check. Optional.
          enable: true # Indicates whether to enable health check. Required. Boolean.
          checkType: "TCP" # Health check type. Optional. Enumerated value: TCP|HTTP|
          intervalTime: 10 # Health check probe interval. Optional. Default value: 5.
          healthNum: 2 # Healthy threshold, indicating the number of consecutive heal
          unHealthNum: 3 # Unhealthy threshold, indicating the number of consecutive
          timeout: 10 # Health check response timeout threshold. This should be less
          httpCode: 31 # Health check status code. Optional. Default value: 31. Value
          httpCheckPath: "/" # Health check path. Optional. Only applicable to HTTP/H
          httpCheckDomain: "" # Health check domain. Optional. Default is the domain
          httpCheckMethod: "HEAD" # Health check method (only applicable to HTTP/HTTP
          httpVersion: "HTTP/1.1" # Custom probe related parameters. When the health
          sourceIpType: 0 # Health check probe source. 0 (VIP as source IP) 1 (100.64
        scheduler: WRR # Request forwarding method. WRR, LEAST_CONN, and IP_HASH indi
```

Example

Sample deployment: jetty-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: jetty
  name: jetty-deployment
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: jetty
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: jetty
    spec:
      containers:
      - image: jetty:9.4.27-jre11
        imagePullPolicy: IfNotPresent
        name: jetty
        ports:
        - containerPort: 80
          protocol: TCP
        - containerPort: 443
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

Sample Service: jetty-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/tke-service-config: jetty-service-config
    # Specify the existing tke-service-config
    # service.cloud.tencent.com/tke-service-config-auto: "true"
    # Automatically create a `tke-service-config`
  name: jetty-service
  namespace: default
spec:
  ports:
  - name: tcp-80-80
    port: 80
    protocol: TCP
    targetPort: 80
  - name: tcp-443-443
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    app: jetty
  type: LoadBalancer
```

This sample includes the following configurations:

The Service is of the public network LoadBalancer type, with two TCP services declared: one on port 80 and the other on port 443.

The `jetty-service-config` CLB configuration is used.

TkeServiceConfig sample: jetty-service-config.yaml

```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
  name: jetty-service-config
  namespace: default
spec:
  loadBalancer:
    l4Listeners:
    - protocol: TCP
      port: 80
      deregisterTargetRst: true
```

```
        healthCheck:
          enable: false
      - protocol: TCP
        port: 443
        session:
          enable: true
          sessionExpireTime: 3600
        healthCheck:
          enable: true
          intervalTime: 10
          healthNum: 2
          unHealthNum: 2
          timeout: 5
        scheduler: WRR
```

This sample includes the following configurations:

The name is `jetty-service-config` , and in the layer-4 listener configuration, two configuration segments are declared:

1. The TCP listener of port 80 will be configured. Health check is disabled.
2. The TCP listener of port 443 will be configured.

Health check is enabled, with the health check interval set to 10s, the healthy threshold set to 2 times, the unhealthy threshold also set to 2 times, and the timeout threshold set to 5s.

The session persistence feature is enabled, with the timeout period set to 3,600s.

The forwarding policy is configured as "weighted round robin".

kubectl configuration commands

```
$ kubectl apply -f jetty-deployment.yaml
$ kubectl apply -f jetty-service.yaml
$ kubectl apply -f jetty-service-config.yaml

$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
jetty-deployment-8694c44b4c-cxscn   1/1     Running   0          8m8s
jetty-deployment-8694c44b4c-mk285   1/1     Running   0          8m8s
jetty-deployment-8694c44b4c-rjrtm   1/1     Running   0          8m8s

$ kubectl get service jetty-service
```

```
NAME     TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)
jetty    LoadBalancer   10.127.255.209   150.158.220.237   80:31338/TCP,443:32373/TC

# Get the `TkeServiceConfig` configuration list
$ kubectl get tkeserviceconfigs.cloud.tencent.com
NAME                     AGE
jetty-service-config     52s

# Update and modify the `TkeServiceConfig` configuration
$ kubectl edit tkeserviceconfigs.cloud.tencent.com jetty-service-config
tkeserviceconfig.cloud.tencent.com/jetty-service-config edited
```

Using Existing CLBs

Last updated：2020-12-28 16:59:04

Tencent Kubernetes Engine (TKE) supports existing Cloud Load Balancers (CLBs) by using the `service.kubernetes.io/tke-existed-lbid: <LoadBalanceId>` annotation. You can use this annotation to specify a CLB instance to be associated with cluster service resources. TKE also provides the feature of **CLB sharing by multiple services**, which allows you to specify multiple services to share an existing CLB. To configure this feature, refer to the sample configuration in this document.

Synchronization for Using Existing CLBs

- When an existing CLB is used, the network-type annotation of the specified service does not work.
- When a service no longer uses an existing CLB, the listener corresponding to the service will be deleted, but the CLB will be retained.
  When a listener is deleted, the name of the listener is checked to verify whether it has been modified by the user. If yes, the listener is considered as created by the user and therefore cannot be automatically deleted.
- If a service is using an automatically created CLB, adding the annotation for using an existing CLB to this service terminates the lifecycle of the current CLB, which will be released accordingly. The configuration of the service will be synchronized with the CLB. Likewise, if the annotation for using an existing CLB is deleted from a service, the Service Controller component will create a CLB for the service and perform synchronization.

Tencent Cloud Tag Synchronization for Using Existing CLBs

- By default, the tag `tke-createdBy-flag = yes` is configured for all CLBs created by services. When a service is terminated, the corresponding resources are deleted. If an existing CLB is used, this tag is not configured, and the corresponding resources are not deleted when the service is terminated.
- The tag `tke-clusterId =` is configured for all services. If the ClusterId is correct, the tag is deleted when the service is terminated.
- For clusters created after August 17, 2020, the feature of sharing the same CLB among multiple services is disabled by default. For the changes and details of CLB tag configuration rules created by services in clusters before and after the aforementioned date, see Multiple Services Sharing a CLB.

Notes:

- The specified CLB must be in the same VPC as the cluster.

- Ensure that your TKE service and CVM service do not share the same CLB.
- You cannot use the CLB console to manage the listeners and servers bound to the TKE-managed CLBs. Your modification will be overwritten during automatic synchronization by TKE.
- When existing CLBs are used:
  - The Service Controller is not responsible for the release and repossession of the existing CLBs.
  - Only CLBs created in the CLB console can be used. CLBs automatically created by TKE cannot be shared because this affects the CLB lifecycle management of other services.
- When CLBs are **shared**:
  - A CLB cannot be shared across clusters.
  - When you need to use this **sharing** feature, we recommend that you implement robust listener port management. Otherwise, chaos may occur when a CLB is shared by multiple services.
  - In case of port conflict, CLB sharing is disabled. If a conflict occurs during modification, synchronization may be improper at the listener backend.
  - For services that share a CLB, local access is disabled (restriction for Classic CLBs).
  - When you delete a service that shares an existing CLB, you must manually unbind the real server that is bound to the CLB. The tag `tke-clusterId: cls-xxxx` is retained for the CLB, and can only be cleared manually.

Service Example

```
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
name: nginx-service
spec:
ports:
- name: 80-80-no
port: 80
protocol: TCP
targetPort: 80
selector:
app: nginx
type: LoadBalancer
```

ⓘ **Note：**

- `service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx` indicates that the service uses an existing CLB for configuration.
- Note that the service type must be set to `LoadBalancer` .

Use Cases

Using a monthly subscription CLB to provide services to external users

When the Service Controller component manages CLB lifecycles, it only supports the purchase of pay-as-you-go CLBs. When you need to use a CLB for a long term, the monthly subscription mode is more cost-effective. In such cases, you can purchase and manage CLBs independently, use annotations to control the use of existing CLBs by services, and remove CLB lifecycle management from the Service Controller component.

Opening the TCP and UDP services in the same port

According to the official Kubernetes restrictions in service design, when multiple port protocols are opened under the same service, these protocols must be the same. In many game scenarios, users need to simultaneously open the TCP and UDP services in the same port. Tencent CLBs support simultaneous listening on UDP and TCP over the same port. This demand can be met through CLB sharing by multiple services.

For example, in the following service configuration, `game-service` is described as two service resources. The descriptions are basically the same except for the protocols for listening. Both services specify the use of an existing CLB `lb-6swtxxxx` through annotations. By applying the resources to a cluster through kubectl, multiple protocols can be exposed over the same CLB port.

```
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
name: game-service-a
spec:
ports:
- name: 80-80-tcp
port: 80
protocol: TCP
targetPort: 80
selector:
app: game
type: LoadBalancer
------------------------------------------------
apiVersion: v1
kind: Service
```

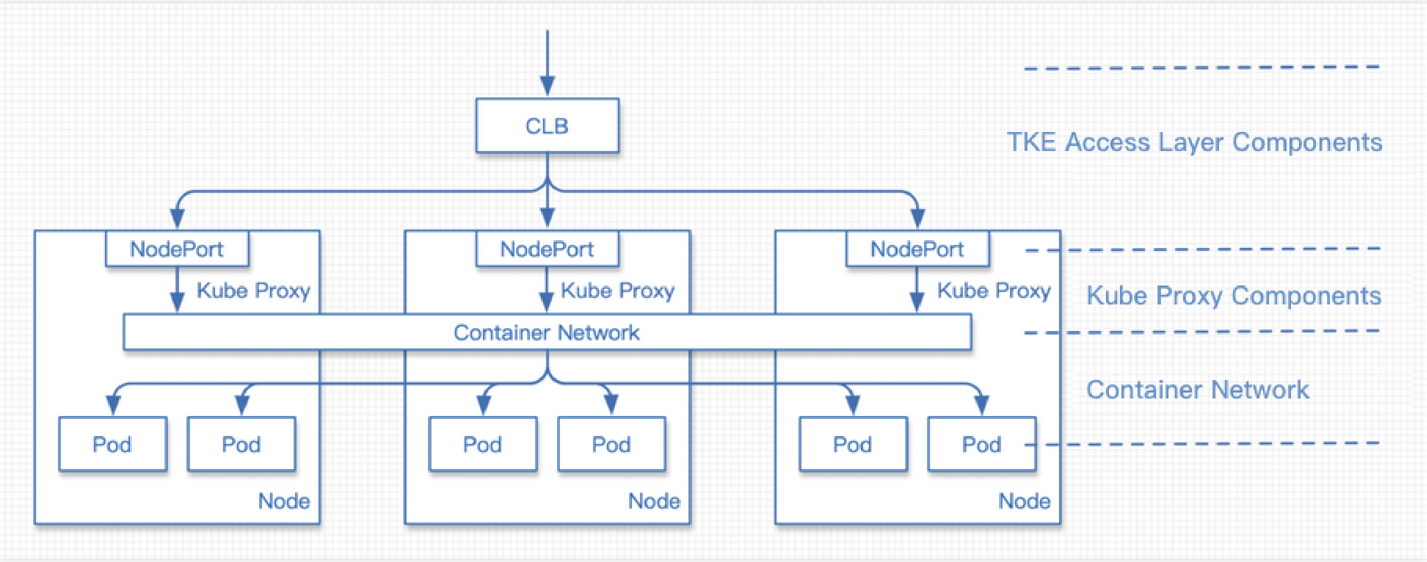
```
metadata:
annotations:
service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
name: game-service-b
spec:
ports:
- name: 80-80-udp
port: 80
protocol: UDP
targetPort: 80
selector:
app: game
type: LoadBalancer
```

Service Backend Selection

Last updated：2022-06-10 19:32:52

Selecting Default Backend

By default, a Service configures the NodePort of a cluster node as the CLB backend, as shown in the TKE Access Layer Components section below. This solution is highly fault-tolerant, where traffic from a CLB instance to any NodePort will be forwarded to a random Pod. This is also the most basic network access layer solution proposed by Kubernetes, as shown below:



`TKE Service Controller` does not use the following nodes as the CLB backend by default:

- Master nodes (which cannot be used for loads at the network access layer).
- Nodes in the `NotReady` status (unhealthy nodes).

> Note：
> `TKE Service Controller` can bind nodes in the `Unschedulable` status. They can be used as the traffic ingress, as they will forward the received traffic in the container network and will not discard it, as shown above.

Specifying Access Layer Backend

For some very large clusters, a Service-managed CLB instance will mount the NodePort of almost all cluster nodes as the backend, which may cause the following problems:

- A limit is imposed on the number of the CLB backends.
- A CLB instance performs a health check on each NodePort, and all health check requests are sent to the backend workload.

Such problems can be solved in the following ways:
For some large clusters, you can specify some nodes to be bound by using the `service.kubernetes.io/qcloud-loadbalancer-backends-label` annotation, which contains a label selector that allows you to bind matching nodes after they are labeled. This process is synced, which means when a node changes so that it is selected or no longer selected, `TKE Service Controller` will add or remove the corresponding backend on the CLB instance. For more information, see Labels and Selectors.

Notes

- When the selector in the `service.kubernetes.io/qcloud-loadbalancer-backends-label` does not select any node, the Service backend will be emptied, interrupting the service. This feature requires cluster node label management.
- Adding a compliant node or changing an existing node will trigger a Controller update.

Use cases

Test application in large cluster

Deploy a test application containing only one or two Pods under a large cluster. When the service is exposed through Service, the CLB instance will perform a health check on all the backend NodePorts, and the number of such requests has a huge impact on the test application. In this case, you can specify a small number of nodes in the cluster as the backends by using labels to relieve the pressure brought by health checks. For more information, see High Health Check Frequency.

Sample

```
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/qcloud-loadbalancer-backends-label: "group=access-layer"
name: nginx-service
spec:
ports:
- name: 80-80-no
port: 80
```

```
protocol: TCP
targetPort: 80
selector:
app: nginx
type: LoadBalancer
```

This sample contains the following configuration:

- Describes a service exposure for public network CLB instances.
- The `service.kubernetes.io/qcloud-loadbalancer-backends-label` annotation declares the backend selector, and only cluster nodes labeled `group=access-layer` will be used as the CLB backends.

Service Local Mode

Kubernetes provides the `ExternalTrafficPolicy` Service feature. When it is set to `Local` , traffic will not be forwarded between nodes through NAT, reducing NAT operations and retaining the source IP. NodePort will only forward traffic to the Pod of the current node. The `Local` mode has the following characteristics:

- Strengths:

- Avoids the performance loss caused by inter-node forwarding through NAT Gateway.

2. Retains the request source IP for the server.

- Shortcomings:
  - NodePort cannot serve nodes without a workload.

Notes

- CLB sync takes time. When the number of Local service workloads is small, the drifting or rolling updates of the workloads are fast. If the updates are not synced to the backend promptly, the backend service may become unavailable.
- It is only suitable for handling low-traffic, low-load businesses and not recommended in the production environment.

Sample: Enabling Local forwarding for Service (externalTrafficPolicy: Local)

```
apiVersion: v1
kind: Service
metadata:
name: nginx-service
spec:
externalTrafficPolicy: Local
```

```
ports:
- name: 80-80-no
port: 80
protocol: TCP
targetPort: 80
selector:
app: nginx
type: LoadBalancer
```

Default backend in Local mode

By default, when the Local mode is enabled for a Service, the NodePorts of almost all nodes will be mounted as the backends. The CLB instance will not forward traffic to backend nodes without workloads based on the health check result. To prevent backends without workloads from being bound, you can specify nodes with workloads as the backends in Local mode by using the `service.kubernetes.io/local-svc-only-bind-node-with-pod: "true"` annotation. For more information, see Using Source IP.

Sample: Enabling Local forwarding and binding for Service

```
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/local-svc-only-bind-node-with-pod: "true"
name: nginx-service
spec:
externalTrafficPolicy: Local
ports:
- name: 80-80-no
port: 80
protocol: TCP
targetPort: 80
selector:
app: nginx
type: LoadBalancer
```

In Local mode, the request traffic to a node is not forwarded among nodes. Therefore, when nodes have different numbers of workloads, using the same backend weight may make the load on each node uneven. In this case, you can perform weighted balancing by using the `service.cloud.tencent.com/local-svc-weighted-balance: "true"` annotation, where the weight of the NodePort backend will be determined by the number of workloads on the node, thus avoiding load unevenness caused by the different numbers of workloads on different nodes. Here, **Local weighted balancing must be used in conjunction with Local binding** as shown below:

Sample: Enabling Local forwarding, binding, and weighted balancing for Service

```
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/local-svc-only-bind-node-with-pod: "true"
service.cloud.tencent.com/local-svc-weighted-balance: "true"
name: nginx-service
spec:
externalTrafficPolicy: Local
ports:
- name: 80-80-no
port: 80
protocol: TCP
targetPort: 80
selector:
app: nginx
type: LoadBalancer
```

Service Cross-region Binding

Last updated：2022-12-23 10:47:33

Overview

When you use the Service of public network CLB type, the CLB is generated for random availability zone in the VPC where the cluster resides by default. Currently, TKE Service of public network CLB allows you to specify availability zones, including availability zones in other regions. This document describes how to bind and specify availability zones for CLB Service across regions via the console and YAML.

Use Cases

- The cross-region access or cross-VPC access of CLB must be supported. That is, the VPC where the CLB resides and the VPC where the cluster resides are not in the same VPC.
- The availability zone of CLB must be specified to realize unified management of resources.

Note：

1. Cross-region binding is only available to bill-by-IP accounts.
2. If you need to use the CLB that is not in the same VPC as this cluster, you need to connect the VPCs of the current cluster and the CLB via CCN.
3. After the VPCs are connected, please submit a ticket to apply for this feature.
4. You should enter the region ID in the following YAML. You can check the region ID in Regions and Availability Zones.

Directions

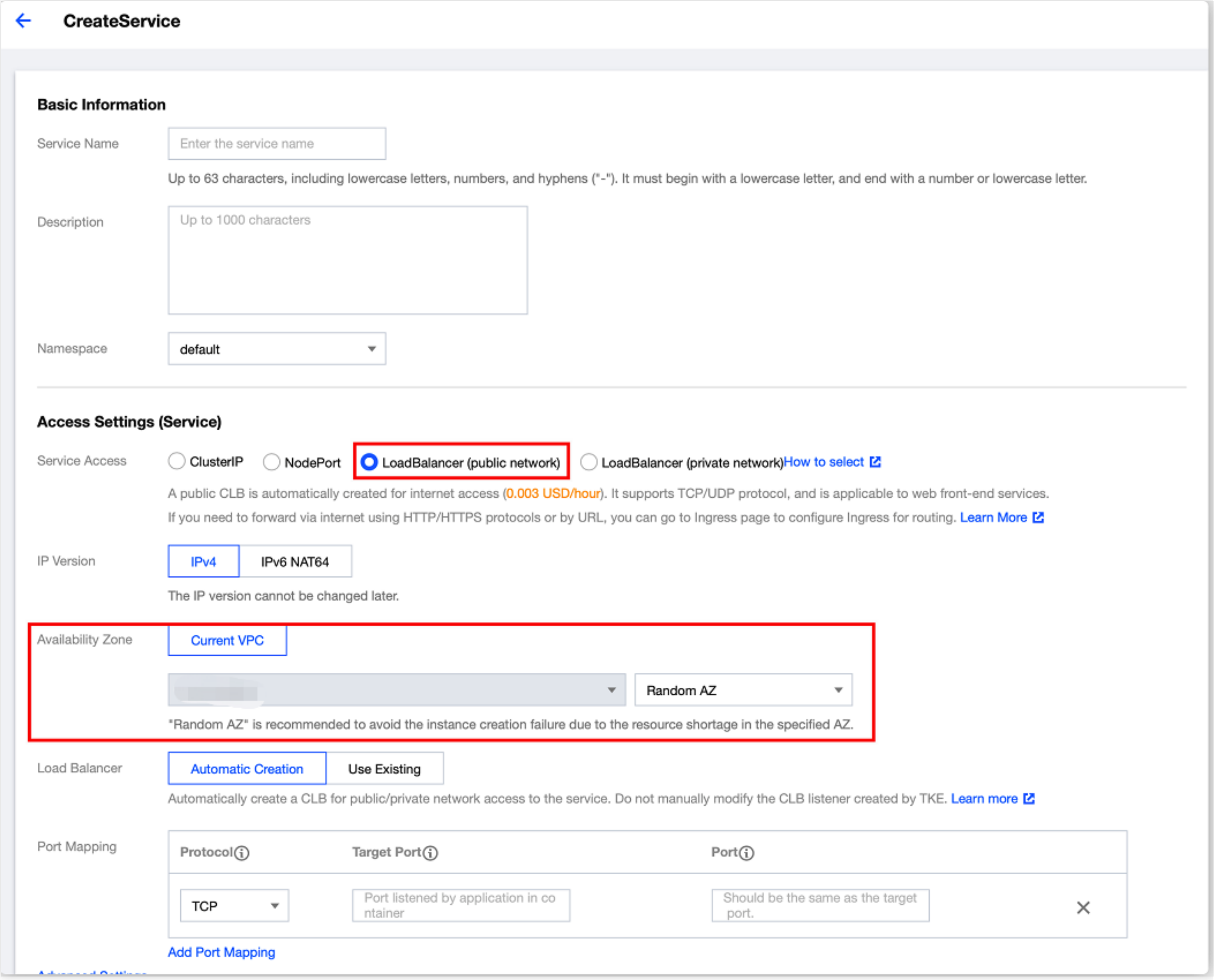
You can bind and specify availability zones for CLB Service across regions via the console and YAML.

- Console
- YAML

1. Log in to the TKE console and click **Cluster** in the left sidebar.

2. In the **Cluster** page, click the ID of the cluster for which you need to create a Service to go to the cluster management page.
3. Select **Services and Routes** > **Service** to go to the **Service** management page and click **Create**.
4. Configure the availability zone rules in the “Create Service” page. The configuration rules are as follows:

- **Service Access**: select **LoadBalancer (public network)**.



← CreateService

Basic Information

Service Name: Enter the service name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Description: Up to 1000 characters

Namespace: default

Access Settings (Service)

Service Access: ○ ClusterIP ○ NodePort ● LoadBalancer (public network) ○ LoadBalancer (private network)How to select

A public CLB is automatically created for internet access (0.003 USD/hour). It supports TCP/UDP protocol, and is applicable to web front-end services.
If you need to forward via internet using HTTP/HTTPS protocols or by URL, you can go to Ingress page to configure Ingress for routing. Learn More

IP Version: IPv4 | IPv6 NAT64

The IP version cannot be changed later.

Availability Zone: Current VPC

[redacted] Random AZ

"Random AZ" is recommended to avoid the instance creation failure due to the resource shortage in the specified AZ.

Load Balancer: Automatic Creation | Use Existing

Automatically create a CLB for public/private network access to the service. Do not manually modify the CLB listener created by TKE. Learn more

Port Mapping:

Protocol	Target Port	Port	
TCP	Port listened by application in container	Should be the same as the target port.	×

Add Port Mapping

Graceful Service Shutdown

Last updated：2022-09-26 16:12:49

Overview

In scenarios where a Pod is connected directly at the access layer, when the backend performs a rolling update, or the backend Pod is deleted, if you delete the Pod directly from the CLB backend, unprocessed requests that have been received by it cannot be processed.
Particularly, in persistent connection scenarios, such as meeting business, if the Pod of the workload is updated or deleted directly, the meeting will be interrupted.

Use Cases

- When a Pod quits gracefully during a workload update, the client will not perceive the jitters and errors generated during the update (if any).
- When a Pod needs to be deleted, it can process the received requests, and inbound traffic is turned off while outbound traffic is still on. Outbound traffic will not be turned off until all existing requests are processed and the Pod is deleted.

> Note：
> This is only effective in the direct access mode. Check whether your cluster supports direct access.

Directions

Step 1. Use an annotation to indicate the use of graceful shutdown

The following is an example of using an annotation to indicate the use of graceful shutdown. For detailed Service annotations, see Service Annotation.

```
kind: Service
apiVersion: v1
metadata:
annotations:
service.cloud.tencent.com/direct-access: "true" ## Enable CLB-to-Pod direct acce
ss
```

```
service.cloud.tencent.com/enable-grace-shutdown: "true" # Indicate the use of gr
aceful shutdown
name: my-service
spec:
selector:
app: MyApp
```

Step 2. Use `preStop` and `terminationGracePeriodSeconds`

Step 2 involves using `preStop` and `terminationGracePeriodSeconds` in the workload that requires graceful shutdown.

Container termination process

The following describes the container termination process in a Kubernetes environment:

1. If a deleted Pod contains `DeletionTimestamp` and is in **Terminating** status, the weight of the Pod on the CLB backend is adjusted to `0` .
2. kube-proxy updates the forwarding rule and removes the Pod from the endpoint list of the Service, and new traffic will no longer be forwarded to the Pod.
3. If a `preStop` hook is configured for the Pod, it will be executed.
4. kubelet will send a SIGTERM signal to each container in the Pod to ask the container processes to stop gracefully.
5. Wait for the container processes to stop completely. If a process has not stopped completely after `terminationGracePeriodSeconds` (30s by default) elapses, a SIGKILL signal will be sent to forcibly stop it.
6. All container processes are terminated, and the Pod resources are cleared.

Specific steps

1. **Use `preStop`**

   To implement graceful termination, you must process the SIGTERM signal in your business code. The main logic is to stop accepting new traffic, continue to process existing traffic, and quit after all connections are closed. For more information, see Go by Example: Signals.

   If the SIGTERM signal is not processed in your business code, or if you cannot control the used third-party library or system to add the logic of graceful termination, you can also try configuring `preStop` for the Pod to implement such logic as shown below:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
   name: lifecycle-demo
   spec:
   ```

```
containers:
- name: lifecycle-demo-container
image: nginx
lifecycle:
preStop:
exec:
command:
- /clean.sh
...
```

For more information on how to configure `preStop`, see Lifecycle.

In certain extreme cases, new connections may still be forwarded within a short period of time after the Pod is deleted. This is because kubelet and kube-proxy watch that the Pod is deleted at the same time, and kubelet may have stopped the containers before kube-proxy syncs the rules. Normally, an application will no longer accept new connections after it receives `SIGTERM` , and it will only keep the existing connections for processing, which may cause some requests to fail at the moment when the Pod is deleted.

In view of the above, you can use `preStop` to make the Pod sleep for a short while first and then start to stop the container processes after kube-proxy completes the rule sync as shown below:

```
apiVersion: v1
kind: Pod
metadata:
name: lifecycle-demo
spec:
containers:
- name: lifecycle-demo-container
image: nginx
lifecycle:
preStop:
exec:
command:
- sleep
- 5s
```

2. **Use `terminationGracePeriodSeconds` to adjust the termination grace period**

If you need a long termination grace period (`preStop` and the business process termination may take more than 30s in total), you can customize `terminationGracePeriodSeconds` as shown below based on the actual situation so as to avoid being stopped by SIGKILL prematurely:

```
apiVersion: v1
kind: Pod
metadata:
name: grace-demo
spec:
terminationGracePeriodSeconds: 60 # The termination grace period is 30s by default, and you can set a longer period.
containers:
- name: lifecycle-demo-container
image: nginx
lifecycle:
preStop:
exec:
command:
- sleep
- 5s
...
```

Capabilities

Graceful shutdown sets the weight on the CLB backend to `0` only when a Pod is deleted. If a running Pod becomes unhealthy, setting the weight to `0` on the backend can reduce the risk of service unavailability.
You can use the `service.cloud.tencent.com/enable-grace-shutdown-tkex: "true"` annotation to implement graceful shutdown.
The annotation will check whether an endpoint in the Endpoint object is `not-ready` , and if so, the annotation will set the weight on the CLB backend to `0` .

References

- Troubleshooting: No Graceful Unbinding on the Backend of NGINX Ingress Controller

Using Services with CLB-to-Pod Direct Access Mode

Last updated：2023-03-17 12:00:45

Overview

For a service in native LoadBalancer mode, a Cloud Load Balancer (CLB) can be automatically created. It first forwards traffic to a cluster through the Nodeport of the cluster, and then forwards it again through iptable or ipvs. Services in this mode can meet users’ needs in most scenarios, but in the following scenarios, **services in CLB-to-Pod direct access mode** are recommended:

The source IP needs to be obtained (local forwarding must be enabled for non-direct access mode)

Higher forwarding performance is required (there are two layers of CLBs when the CLB and service are in non-direct access mode, so performance loss is inevitable).

Complete health checks and session persistence are required for the Pod layer (there are two layers of CLBs when the CLB and service are in non-direct access mode, so health checks and session persistence are difficult to configure).

Description

If your cluster is a Serverless cluster, it defaults to the CLB-to-Pod direct access mode, and you don't need to do anything.

The CLB-to-Pod direct access mode is available for both GlobleRouter and VPC-CNI container network modes. Click the cluster ID in the cluster list to go to the cluster details page. In the **Basic Information** page, you can find the container network add-on used by the current cluster.

VPC-CNI Mode

Use limits

The Kubernetes version of the cluster must be 1.12 or later.

The VPC-CNI ENI mode must be enabled for the cluster network mode.

The workloads used by a service in direct access mode must adopt the VPC-CNI ENI mode.

Up to 200 workload replicas can be bound to the CLB backend by default. If you need to bind more replicas, please submit a ticket to increase the quota.

The feature limits of a CLB bound to an ENI must be satisfied. For more information, see Binding an ENI.

When workloads in CLB-to-Pod direct access mode are updated, a rolling update is performed based on the health check status of the CLB, which will affect the update speed.

HostNetwork type workloads are not supported.

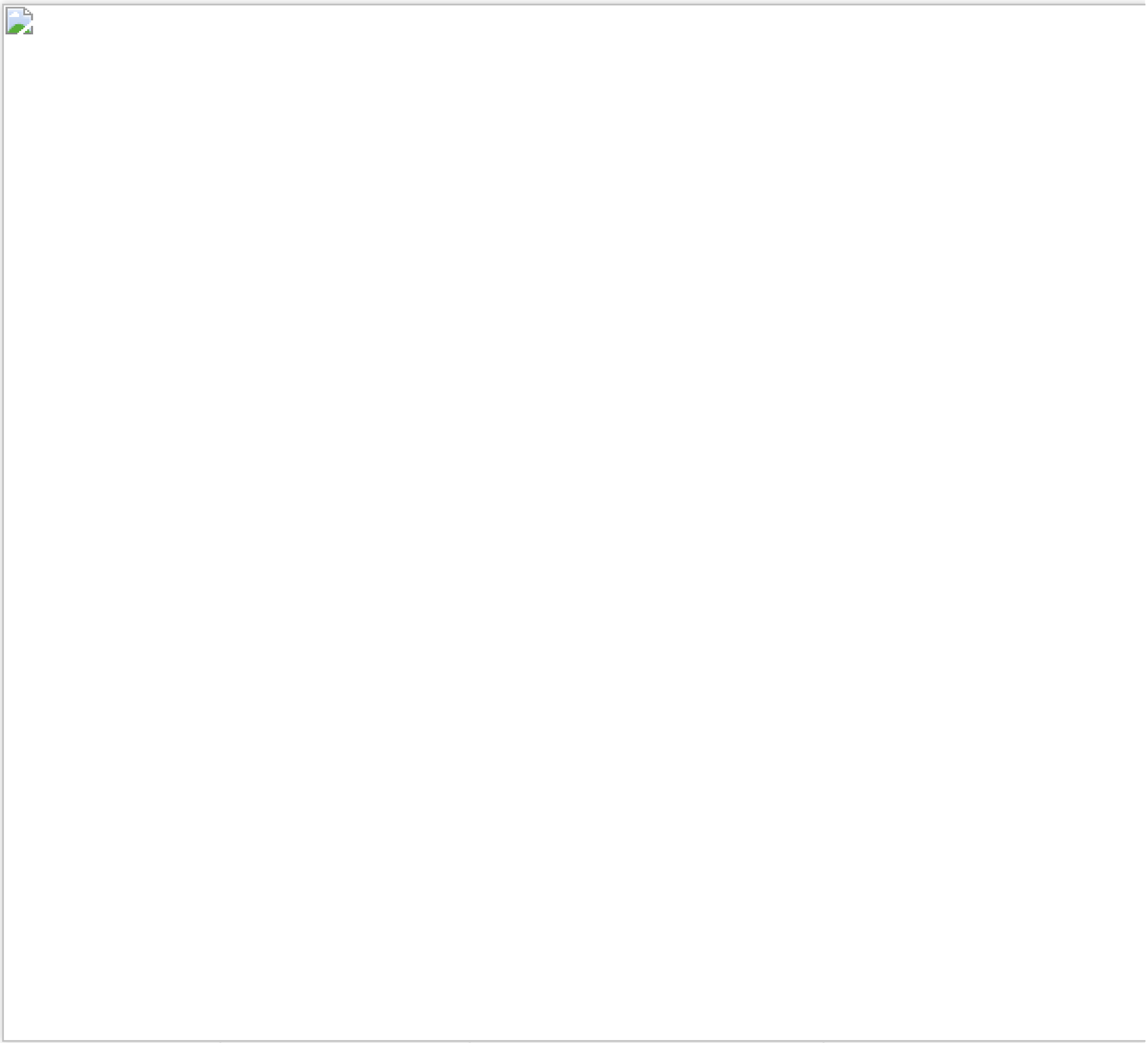
Directions

Console

YAML

1. Log in to the TKE console.
2. Refer to the steps of Creating a service in the console to go to the **Create a service** page and set the service parameters as required.

Some key parameters need to be set as follows:



Service access method: Select **Public network CLB access** or **Private network CLB access**.

Network mode: Select **Enable CLB-to-Pod direct access**.

Workload binding: Select **Reference workload**.

3. Click **Create service**.
The YAML configuration for a service in CLB-to-Pod direct access mode is the same as that for a common service. In this example, the annotation indicates whether to enable the CLB-to-Pod direct access mode.

```
kind: Service
apiVersion: v1
metadata:
  annotations:
    service.cloud.tencent.com/direct-access: "true" ##Enable CLB-to-Pod direct acce
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

Annotation extension

For the CLB configuration, see TkeServiceConfig. The annotation configuration is as follows:

```
service.cloud.tencent.com/tke-service-config: [tke-service-configName]
```

Must-Knows

Ensuring the availability during rolling update

ReadinessGate, provided by the official Kubernetes, is mainly used to control the status of Pod, and requires the cluster version to be later than 1.12. By default, a Pod has the following conditions: PodScheduled, Initialized, ContainersReady, when these statuses are all Ready, the Pod Ready passes the conditions. However, in the cloud native scenario, the status of Pods needs to be judged in combination with other factors. `ReadinessGate` provides a mechanism that allows you to add a fence for the Pod's status judgment, which is judged and controlled by a third party. In this way, the status of the Pod is associated with the third party.

Changes in the rolling update of CLB-to-Pod direct access mode

When users start the rolling update of an app, Kubernetes will perform the rolling update according to the update policy. However, the identification that it uses to judge whether a batch of Pods have started only includes the status of the Pods themselves, and does not consider whether the Pods are configured with health check in the CLB and have passed it. If such Pods cannot be scheduled in time when the access layer components are under high load, the Pods with successful rolling update may not be providing services to external users, thus resulting in service interruption.
In order to associate the backend status of the CLB and rolling update, the TKE access layer components introduced a new feature: `ReadinessGate` , which was introduced in Kubernetes 1.12. Only when the TKE access layer components confirm that the backend binding is successful and the health check is passed, will it configure the state of `ReadinessGate` , so that Pods can reach the Ready state and the rolling update of the entire workload can be facilitated.

Using ReadinessGate in a cluster

Kubernetes clusters provide a service registration mechanism. You only need to register your services to a cluster in the form of `MutatingWebhookConfigurations` resources. When a Pod is created, the cluster will deliver notifications according to the configured callback path. At this time, the pre-creation operation can be performed for the Pod, that is, `ReadinessGate` can be added to the Pod. This callback process must be based on HTTPS. That is, the CA that issues requests needs to be configured in `MutatingWebhookConfigurations` , and a certificate issued by the CA needs to be configured on the server.

Disaster recovery of the ReadinessGate mechanism

The service registration or certificates in user clusters may be deleted by users, although these system component resources should not be modified or destroyed by users. However, such problems will inevitably occur because of users’ exploration of clusters or misoperations. Therefore, the integrity of the above resources will be checked when the access layer component is started, and the resources will be rebuilt if the integrity is damaged to strengthen the robustness of the system. For more information, see Pod readiness.

GlobalRouter Mode

Use limits

A workload can only run in one network mode. You can choose VPC-CNI ENI mode or GlobalRouter mode for the workloads used by a service in direct access mode.

It is only available for the bill-by-IP accounts.

Up to 200 workload replicas can be bound to the CLB backend by default. If you need to bind more replicas, please submit a ticket to increase the quota.

When the CLB-to-Pod direct access mode is used, the network linkage is restricted by the security group of CVM. Please confirm whether the security group configuration opens the corresponding protocol and port. **The port corresponding to the workload on the CVM needs to be opened.**

After the CLB-to-Pod direct access mode is enabled, the ReadinessGate will be enabled by default. It will check whether the traffic from the load balancer is normal during the rolling update of Pod. You also need to configure the correct health check configuration for the application. For details, see TkeServiceConfig.

YAML operation instructions

The YAML configuration for a service in CLB-to-Pod direct access mode is the same as that for a common service. In this example, the annotation indicates whether to enable the CLB-to-Pod direct access mode.

Prerequisites

Add `GlobalRouteDirectAccess: "true"` to the `kube-system/tke-service-controller-config` ConfigMap to enable the direct access capability of GlobalRoute.

Enable the direct access mode in the Service's YAML

```
kind: Service
apiVersion: v1
metadata:
  annotations:
    service.cloud.tencent.com/direct-access: "true" ##Enable CLB-to-Pod direct acce
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
```

```
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

Annotation extension

For the CLB configuration, see TkeServiceConfig. The annotation configuration is as follows:

```
service.cloud.tencent.com/tke-service-config: [tke-service-configName]
```

Multiple Services Sharing a CLB

Last updated：2023-03-30 16:26:40

Scenario

You can use the feature for sharing the same CLB among multiple Services to support the simultaneous opening of TCP and UDP on the same port for the same VIP.

Note

This feature is not recommended for other scenarios.

Notes

For TKE clusters created before Aug. 17, 2020, the CLBs created by their Services support the sharing of the same CLB by default.

For TKE clusters created after Aug. 17, 2020, the feature of multiple Services sharing the same CLB is disabled by default.

If you need to reuse CLB instances for Services, submit a ticket for application.

If it is a TKE Serverless cluster, the CLB sharing is enabled by default. Notes:

1.1 Only CLB instances purchased manually can be reused, and those purchased automatically by a serverless cluster cannot. If those purchased automatically are reused, an error will be reported. This is to protect them from being repossessed by the serverless cluster.

1.2 The following two annotations must be added to the Service once the CLB is purchased:

service.kubernetes.io/qcloud-share-existed-lb:"true"

service.kubernetes.io/tke-existed-lbid:lb-xxx

The management and sync of configurations between Service and CLB instances are based on the resource object of the `LoadBalancerResource` type named the CLB ID. Do not perform any operations on this CRD; otherwise, the Service may fail.

Use Limits

In Service reuse scenarios, the number of listeners managed by a CLB instance is subject to the `TOTAL_LISTENER_QUOTA` of the CLB instance. For more information, see DescribeQuota.

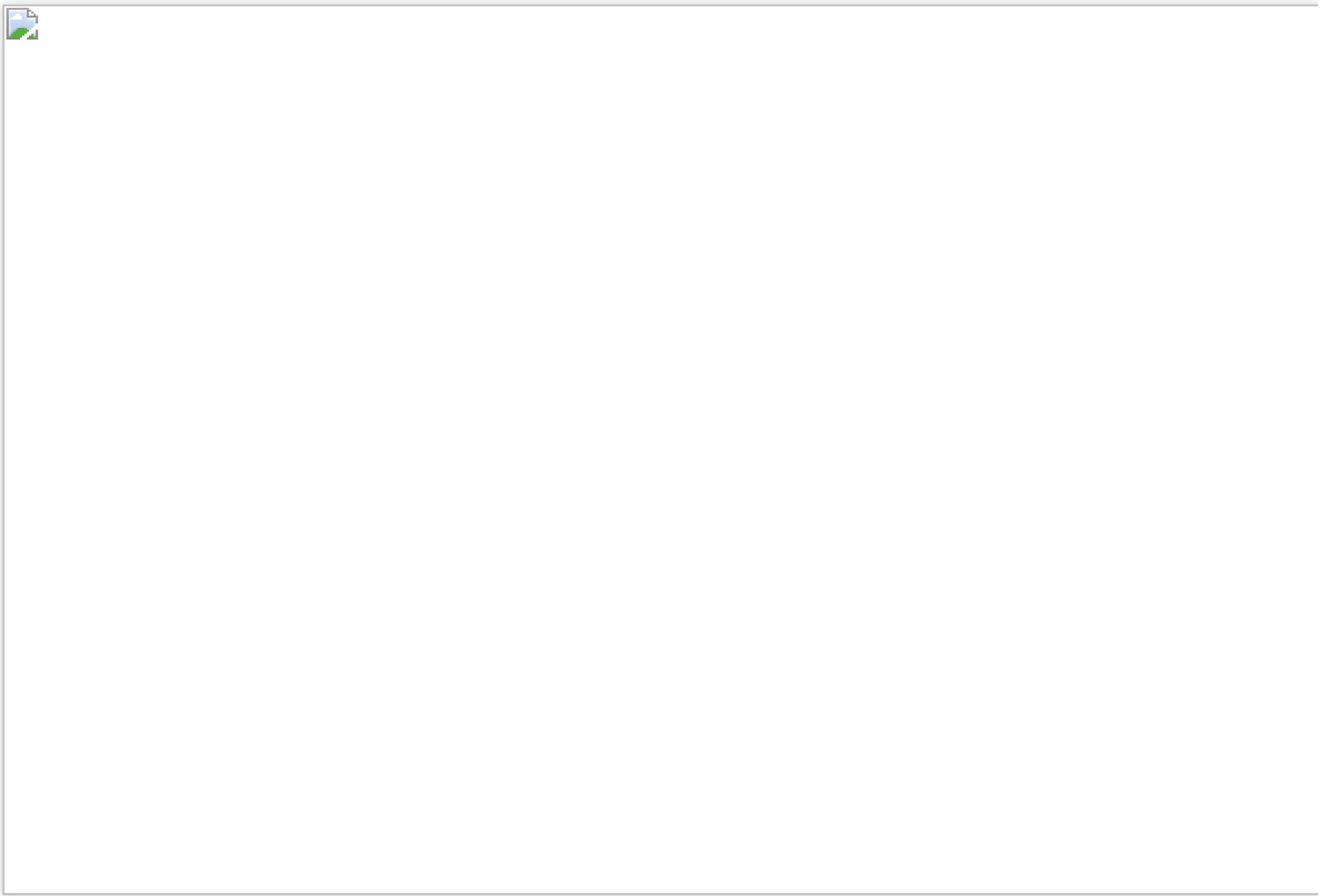
In scenarios where a Service is reused, only the user-created Cloud Load Balancer (CLB) can be used. This is because when the CLB created in the TKE cluster is reused, CLB resources may not be released, leading to a resource leak.

Note

After reusing CLB resources created by the current TKE, you need to manually manage the CLB resources, because the CLB's life cycle will not be controlled by the TKE due to the lack of the tag.

Directions

1. Refer to Creating CLB Instances to create a public or private CLB in the VPC where the cluster is located.
2. Refer to Creating a Deployment or Creating a Service to create a Service of the Loadbalancer type. Select **Use existing** for load balancer and choose the CLB instance created in Step 1.



3. Repeat Step 2 to share the same CLB among multiple Services.

Service Extension Protocol

Last updated：2023-05-23 10:31:13

Protocols Supported by Services by Default

A Service is a mechanism and abstraction through which Kubernetes exposes applications outside the cluster. You can access the applications in a cluster through a Service.

Notes

For access in direct access mode, there are no restrictions on the use of extension protocols, and TCP and UDP protocols can be used together.

In non-direct access scenarios, `ClusterIP` and `NodePort` modes can be used together. However, the community has restrictions on Services of the `LoadBalancer` type, and only protocols of the same type can be used currently.

When `LoadBalancer` is declared as TCP, the port can use the capabilities of extension protocols to change the protocol of CLB to TCP_SSL, HTTP, or HTTPS.

When `LoadBalancer` is declared as UDP, the port can use the capabilities of extension protocols to change the protocol of CLB to UDP.

TKE Extension of Service Forwarding Protocols

In addition to the rules of the protocols supported by a native Service, a Service needs to support the hybrid use of TCP and UDP as well as the TCP SSL, HTTP, and HTTPS protocols in certain scenarios. TKE extends the support for more protocols in `LoadBalancer` mode.

Prerequisites

Extension protocols are only effective for Services in `LoadBalancer` mode.

An extension protocol describes the relationship between the protocol and the port through an annotation.

The relationship between the extension protocol and the annotation is as follows:

When the port described in `Service Spec` is not covered in the annotation of the extension protocol, `Service Spec` will be configured according to your declaration.

When the port described in the annotation of the extension protocol does not exist in `Service Spec`, the configuration will be ignored.

When the port described in the annotation of the extension protocol exists in `Service Spec`, the protocol configuration declared in `Service Spec` will be overwritten.

Annotation name

service.cloud.tencent.com/specify-protocol

Sample annotations of extension protocols

Sample for TCP_SSL

Sample for HTTP

Sample for HTTPS

Sample for TCP/UDP

Sample for hybrid use

QUIC

```
{"80":{"protocol":["TCP_SSL"],"tls":"cert-secret"}}
```

```
{"80":{"protocol":["HTTP"],"hosts":{"a.tencent.com":{},"b.tencent.com":{}}}}
```

```
{"80":{"protocol":["HTTPS"],"hosts":{"a.tencent.com":{"tls":"cert-secret-a"},"b.te
```

```
{"80":{"protocol":["TCP","UDP"]}} # Only supported in direct access mode. For more
```

```
{"80":{"protocol":["TCP_SSL","UDP"],"tls":"cert-secret"}} # Only supported in dire
```

```
{"80":{"protocol":["QUIC"],"tls":"cert-secret"}}
```

Notes

The field `cert-secret` in TCP_SSL and HTTPS indicates that a certificate must be specified when you use the protocol. The certificate is an Opaque type Secret, the key of Secret is qcloud_cert_id, and the value is the certificate ID. For details, see Ingress Certificate Configuration.

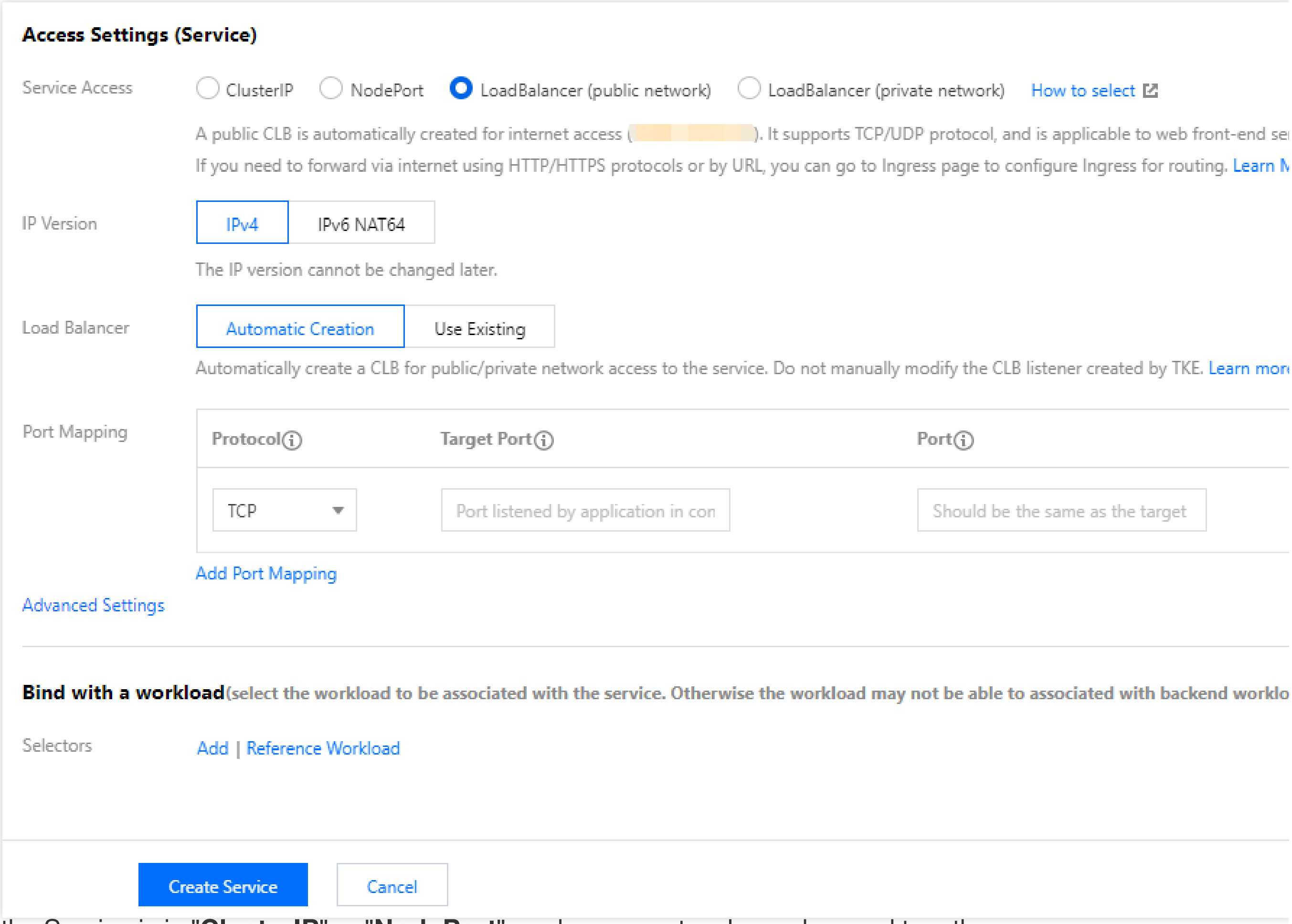
Extension protocol use instructions

Use instructions of extension protocol `YAML`

Use instructions of extension protocols in the console

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["TCP_SSL"],"tls
  name: test
    ....
```

If you expose a Service in the form of "**public network CLB**" or "**private network CLB**" when creating it, in modes other than direct access mode, only TCP and TCP SSL can be used together in **Port Mapping** as shown below:



Access Settings (Service)

Service Access ○ ClusterIP ○ NodePort ● LoadBalancer (public network) ○ LoadBalancer (private network) How to select

A public CLB is automatically created for internet access ([redacted]). It supports TCP/UDP protocol, and is applicable to web front-end se

If you need to forward via internet using HTTP/HTTPS protocols or by URL, you can go to Ingress page to configure Ingress for routing. Learn M

IP Version: IPv4 | IPv6 NAT64

The IP version cannot be changed later.

Load Balancer: Automatic Creation | Use Existing

Automatically create a CLB for public/private network access to the service. Do not manually modify the CLB listener created by TKE. Learn more

Port Mapping

Protocol	Target Port	Port
TCP	Port listened by application in con	Should be the same as the target

Add Port Mapping

Advanced Settings

Bind with a workload(select the workload to be associated with the service. Otherwise the workload may not be able to associated with backend worklo

Selectors: Add | Reference Workload

When the Service is in "**ClusterIP**" or "**NodePort**" mode, any protocols can be used together.

If you are using services with CLB-to-Pod direct access mode, hybrid use of any protocols is supported.

Cases

A native Service does not support hybrid use of protocols. Upon some special modifications, TKE supports hybrid use of protocols in CLB-to-Pod direct access mode.

Please note that the same protocol is used in YAML, but you can specify the protocol type for each port via the annotation. In the sample below, port 80 uses the TCP protocol, and port 8080 uses the UDP protocol.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/direct-access: "true"  # TKE Serverless clusters defa
    service.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["TCP"]},"8080":
  name: nginx
spec:
  externalTrafficPolicy: Cluster
  ports:
  - name: tcp-80-80
```

```
      nodePort: 32150
      port: 80
      protocol: TCP
      targetPort: 80
    - name: udp-8080-8080
      nodePort: 31082
      port: 8080
      protocol: TCP # Note: Only the same type of protocols can be used because of th
      targetPort: 8080
    selector:
      k8s-app: nginx
      qcloud-app: nginx
    sessionAffinity: None
    type: LoadBalancer
```

Service Annotation

Last updated：2023-04-07 20:04:01

You can use the following annotations to configure Services to enrich CLB capabilities.

Annotation Usage

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
  name: test
........
```

Annotation Collection

service.kubernetes.io/loadbalance-id

Note:

This is a read-only annotation that provides the `LoadBalanceId` imported by the current Service. You can go to Tencent Cloud CLB console to view the IDs of the CLB instances in the same VPC with the cluster.

service.kubernetes.io/qcloud-loadbalancer-internal-subnetid

Note:

This annotation is used to specify the creation of a private network CLB instance. Its value is the subnet ID.

Use case:

`service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx`

service.kubernetes.io/tke-existed-lbid

Note:

When you use an existing CLB instance, you should note that different usages have different impacts on Tencent Cloud tags.

Use case:

For the detailed usage, see Using Existing CLBs.

service.kubernetes.io/local-svc-only-bind-node-with-pod

Note:

In Service Local mode, only nodes with Pods are bound.

Use case:

For the detailed usage, see Service Local Mode.

service.cloud.tencent.com/local-svc-weighted-balance

Note:

It is used with the annotation `service.kubernetes.io/local-svc-only-bind-node-with-pod` .

The weight of the CLB backend is determined by the number of workloads on the nodes.

Use case:

For the detailed usage, see Service Local Mode.

service.kubernetes.io/qcloud-loadbalancer-backends-label

Note:

This annotation is used to specify a tag for setting the nodes to be bound to the CLB backend.

Use case:

For the detailed usage, see Specifying the Access-Layer Backend.

service.cloud.tencent.com/direct-access

Note:

This annotation is used to connect a CLB instance directly to a Pod.

Use case:

For the detailed usage, see Using Services with CLB-to-Pod Direct Access Mode.

service.cloud.tencent.com/tke-service-config

Note:

This annotation is used to configure CLB through `tke-service-config` .

Use case:

For the detailed usage, see Service CLB Configuration.

service.cloud.tencent.com/tke-service-config-auto

Note:

This annotation is used to automatically create a `TkeServiceConfig` .

Use case:

For the detailed usage, see Associated Actions Between Service and TkeServiceConfig.

service.kubernetes.io/loadbalance-nat-ipv6

Note:

This is a read-only annotation. When you create an NAT64 IPv6 CLB instance, its IPv6 address will be displayed in the annotation.

Use case:

```
service.kubernetes.io/loadbalance-nat-ipv6: "2402:4e00:1402:7200:0:9223:5842:2a44"
```

service.kubernetes.io/loadbalance-type (it will be disused soon)

Note:

This annotation is used to control the type of the automatically created CLB instance: classic CLB or CLB.

Valid values: yunapi_clb (classic), classic (classic), yunapiv3_forward_clb (CLB)

Default value: yunapiv3_forward_clb (CLB)

Note

Without special needs, we don't recommend you use classic CLB, which has ceased to be iterated and lacks many features.

service.cloud.tencent.com/specify-protocol

Note:

This annotation is used to configure TCP, UDP, TCP SSL, HTTP, or HTTPS for the specified listening port.

Use case:

For the detailed usage, see Service Extension Protocol.

service.kubernetes.io/service.extensiveParameters

Note:

This annotation uses the parameters configured when the CLB was created. It can only be configured at the time of creation and cannot be modified after the creation.

Refer to Creating a CLB Instance to add custom parameters for the created CLB instance.

Use case:

Creating a NAT64 IPv6 instance:

service.kubernetes.io/service.extensiveParameters: '{"AddressIPVersion":"IPV6"}'

Purchasing a CTCC CLB:

service.kubernetes.io/service.extensiveParameters: '{"VipIsp":"CTCC"}'

Creating a CLB with a custom name:

service.kubernetes.io/service.extensiveParameters: '{"LoadBalancerName":"my_cutom_lb_name"}'

service.cloud.tencent.com/enable-grace-shutdown

Note:

This annotation is used to shut down CLB instances gracefully in direct access mode. If a deleted Pod contains `DeletionTimestamp` and is in **Terminating** status, the weight of the Pod on the CLB backend is adjusted to `0` .

Use case:

It is only supported in direct access mode and needs to be used together with `service.cloud.tencent.com/direct-access` . For more information on how to use it, please see Graceful Service Shutdown.

service.cloud.tencent.com/enable-grace-shutdown-tkex

Note:

This annotation is used to shut down CLB instances gracefully in direct access mode. If the endpoints in the Endpoint object are `not-ready` , the weights on the CLB backend are adjusted to `0` .

Use case:

It is only supported in direct access mode and needs to be used together with `service.cloud.tencent.com/direct-access` . For more information on how to use it, see Graceful Service Shutdown.

service.kubernetes.io/qcloud-loadbalancer-internet-charge-type

Note:

The billing type of CLB can only be configured at the time of creation, and cannot be modified after the creation.

This annotation is used to specify the CLB payment mode when a CLB is created. Please use it with `service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out` annotation.

Valid values:

`BANDWIDTH_POSTPAID_BY_HOUR` : Postpaid by bandwidth on an hourly basis

`TRAFFIC_POSTPAID_BY_HOUR` : Postpaid by traffic on an hourly basis

Use case:

service.kubernetes.io/qcloud-loadbalancer-internet-charge-type `: "TRAFFIC_POSTPAID_BY_HOUR"`

service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out

Note:

CLB bandwidth can only be configured at the time of creation, and cannot be modified after the creation.

This annotation is used to specify the maximum outbound bandwidth of the CLB when a CLB is created, which applies only to public network CLB instances. Please use it with `service.kubernetes.io/qcloud-loadbalancer-internet-charge-type` annotation.

Valid values:

Value range: 1-2,048 Mbps

Use case:

```
service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out: "2048"
```

service.cloud.tencent.com/security-groups

Note:

This annotation is used to bind security groups to CLB-type Services. Up to five security groups can be bound to a CLB.

Note:

For more information, see Use Limits of CLB security groups.

Usually, the "Allow Traffic by Default" feature must be enabled, with which the traffic forwarding between CLB and CVM is allowed by default. Traffic coming from the CLB only needs to be verified by the security group bound to the CLB. The annotation is `service.cloud.tencent.com/pass-to-target` .

When Using Existing CLBs, logic conflicts may occur if different security groups are configured for multiple Services.

Use case:

```
service.cloud.tencent.com/security-groups: "sg-xxxxxx,sg-xxxxxx"
```

service.cloud.tencent.com/pass-to-target

Note:

This annotation is used to configure the "Allow Traffic by Default" feature for the CLB-type Services. The traffic forwarding between CLB and CVM is allowed by default. Traffic coming from the CLB only needs to be verified by the security group bound to the CLB.

Note:

For more information, see Use Limits of CLB security groups.

Usually, the feature of binding a security group is required. The annotation is `service.cloud.tencent.com/security-groups` .

When Using Existing CLBs, logic conflicts may occur if different “Allow Traffic” configurations are configured for multiple Services.

Use case:

```
service.cloud.tencent.com/pass-to-target: "true"
```

Ingress Management

Ingress Controllers

Last updated：2023-05-06 19:41:07

Ingress Controllers

Application CLB

Application CLB is a TKE Ingress Controller based on the Tencent Cloud Load Balancer (CLB), which can implement the access of different services in the cluster with different URLs. CLB directly forwards the traffic to a Pod through the NodePort (the traffic is forwarded to a Pod in the CLB-to-Pod direct access mode). One Ingress configuration is bound to one CLB instance (IP), which is suitable for scenarios that only require simple routing management and are insensitive to IP address convergence. For more information, see CLB Type Ingress.

Istio Ingress Gateway

Istio Ingress Gateway is an Ingress Controller based on Tencent Cloud CLB and Istio Ingress Gateway (provided by Tencent Cloud TCM). The control plane and related supporting components are maintained by Tencent Cloud. You only need to deploy the containerized data plane that performs traffic forwarding in the cluster. You can use native Kubernetes Ingress or Istio API that provides more refined traffic management capabilities. A layer of proxy (envoy) is added after CLB, which is suitable for scenarios where there are more requirements for access layer routing management, IP address convergence, and entrance traffic management of cross-cluster and heterogeneous deployment service.

Dedicated API Gateway

Dedicated API Gateway is a TKE Ingress Controller based on a dedicated Tencent Cloud API Gateway instance. It is suitable for scenarios where multiple TKE clusters require a unified access layer or the access layer requires authentication and traffic throttling. For more information, see API Gateway Type Ingress. It has the following strengths:

API Gateway is directly connected to the Pods of the TKE cluster without any intermediate nodes.

An API Gateway TKE tunnel can connect multiple TKE services at the same time, among which the traffic is distributed based on the weighted round robin algorithm.

Advanced extended capabilities provided by API Gateway can be used, such as authentication, traffic throttling, canary traffic distribution, caching, and downgrade upon circuit breaking.

Supported by a dedicated API Gateway instance, the underlying physical resources of a user are exclusive to the user, with a stable performance and high SLA delivered.

Nginx Ingress Controller

Nginx Ingress Controller is an Ingress controller based on Tencent Cloud CLB and Nginx reverse proxy (containerized deployment in cluster). It extends the features of native Kubernetes Ingress through Annotations, and adds a layer of proxy (nginx) after CLB, which is suitable for scenarios where there are more requirements for access layer routing management and IP address convergence. For more information, see Nginx Type Ingress.

Ingress Controllers Comparison

Module	Description	Application CLB	Istio Ingress Gateway (Provided by Tencent Cloud TCM)	Dedicated API Gateway	Nginx Ingress Controller
Traffic management	Supported protocols	HTTP and HTTPS	HTTP, HTTPS, HTTP2, GRPC, TCP, and TCP + TLS	HTTP, HTTPS, HTTP2, and GRPC	HTTP, HTTPS, HTTP2, GRPC, TCP, and UDP
	IP Management	One Ingress rule corresponds to one IP (CLB).	Multiple Ingress rules correspond to one IP (CLB). IP address convergence is supported.	Multiple Ingress rules correspond to one IP (Dedicated API Gateway). IP address convergence is supported.	Multiple Ingress rules correspond to one IP (CLB). IP address convergence is supported.
	Attribute route	HOST and URL	More attributes are supported, such as header, method, query, and parameter.	More attributes are supported, such as header, method, query, and parameter.	More attributes are supported, such as header and cookie.
	Traffic behavior	Not supported	Behaviors such as rewrite and	Redirection, custom request, and custom response are supported.	Behaviors such as rewrite and redirection

			redirection are supported.		are supported.
	Region-aware load balancing	Not supported	Supported	Not supported	Not supported
Application access addressing	Service discovery	Single Kubernetes cluster	Multiple Kubernetes clusters + heterogeneous service	Multiple Kubernetes clusters	Single Kubernetes cluster
Security	SSL configuration	Supported	Supported	Supported	Supported
	Authentication authorization	Not supported	Supported	Supported	Supported
Observability	Monitoring metrics	Supported (View in CLB)	Supported (Cloud native monitoring or Tencent Cloud Observability Platform)	Supported (View in API Gateway)	Supported (Cloud native monitoring)
	Call tracing	Not supported	Supported	Not supported	Not supported
	Add-on Ops	The associated CLB has been managed. You only need to run TKE Ingress Controller in the cluster.	The control plane has been managed. You only need to run the data plane Ingress Gateway.	You don't need to run the control plane in the Kubernetes cluster. Instead, simply enable the private network access feature in the cluster.	You need to run Nginx Ingress Controller in the cluster (control plane + data plane).

CLB Type Ingress
Overview

Last updated：2022-12-13 18:23:37

Services expose TKE in clusters based on the layer-4 network. Exposed service types, such as ClusterIP, NodePort, and LoadBalancer, are all based on the access entry of layer-4 network services. They lack layer-7 network capabilities, such as load balancing, SSL, and name-based virtual hosts. An Ingress exposes HTTP and HTTPS services in the layer-7 network and provides common layer-7 network capabilities.

Basic Ingress Concepts

An Ingress is a collection of rules that allow access to services of a cluster. You can configure different forwarding rules to allow different URLs to access different services. To properly run Ingress resources, the cluster must run an Ingress controller. TKE enables the CLB-based TKE Ingress Controller by default in the cluster.

Ingress Lifecycle Management

The external service capability of an Ingress depends on resources provided by the CLB. Service resource management is one of the important feature of an Ingress. The following table describes the labels that an Ingress will use for resource lifecycle management.

Label	Description
`tke-createdBy-flag = yes`	• Indicates that the resource was created by TKE. When an Ingress with this label is deleted, the corresponding resources are also deleted. • When an Ingress without this label is destroyed, only the CLB listener is deleted and the CLB will not be deleted.
`tke-clusterId = <clusterId>`	• Identifies the cluster that uses the resource. • When the Ingress is deleted, the corresponding label (with correct ClusterId) will be deleted.
`tke-lb-ingress-uuid = <Ingress UUID>`	• Identifies the Ingress that uses the resource. • Currently, an Ingress cannot reuse a CLB with other Ingresses. If you specify that an Ingress use an existing CLB but the label value is incorrect, the request will be rejected. • When the Ingress is deleted, the corresponding label (with correct Ingress UUID) will be deleted.

Ingress Controller Usage Method

In addition to TKE Ingress Controller provided by Tencent Cloud, the Kubernetes community has various third-party Ingress controllers. These Ingress controllers expose services in the layer-7 network. The Kubernetes community allows you to use the `kubernetes.io/ingress.class` annotation to distinguish different Ingress controllers and determine the controller that processes an ingress. TKE Ingress Controller also supports this annotation. The detailed rules and use suggestions are as follows:

- When an Ingress does not have the `kubernetes.io/ingress.class` annotation, TKE Ingress Controller will manage the Ingress.
- When an Ingress has the `kubernetes.io/ingress.class` annotation and its value is `qcloud` , TKE Ingress Controller will manage the Ingress.
- When an Ingress modifies the `kubernetes.io/ingress.class` annotation content, TKE Ingress Controller will add the Ingress to or remove it from its management scope based on the annotation content. This operation will create or release an Ingress.
- When TKE Ingress Controller is not required, you can change the number of `Deployment` (`kube-system:l7-lb-controller`) replicas in the cluster to 0 to disable the TKE Ingress Controller feature.

> Note：
>
> - Before disabling the TKE Ingress Controller feature, ensure that no Ingress is managed by TKE Ingress Controller to prevent CLB release failures.
> - If **Deletion Protection** is enabled or a **private connection** is used for the CLB, the CLB will not be deleted when services are deleted.

Ingress Operations

For more information about Ingress-related operations and features, see the following documents:

- Basic Ingress Features
- Using an Existing CLB for Direct Pod Connection
- Using TKEServiceConfig to Configure CLBs
- Mixed Use of HTTP and HTTPS Protocols through Ingress
- Ingress Certificate Configuration

Basic Ingress Features

Last updated：2023-05-06 19:41:07

Overview

Ingress is a collection of rules that allow access to Services of a cluster. You can configure different forwarding rules to allow different URLs to access different Services.

To properly run Ingress resources, the cluster must run an Ingress controller. TKE enables the CLB-based `l7-lb-controller` by default in the cluster. It supports HTTP and HTTPS as well as other self-built Ingress controllers in the cluster. You can select different Ingress types based on your business needs.

Notes

The architecture of Tencent Cloud Load Balancer (CLB) has been upgraded on March 6, 2023. After the upgrade, public network CLB instances deliver services through domain names. As service traffic increases, the VIP changes dynamically. Therefore, the VIP of a CLB instance is no longer displayed in the console. For more information, see Launch of Domain Name-Based Public CLB Instances.

For new Tencent Cloud users, the upgraded domain name-based CLB instances are used by default.

Existing users can choose to continue to use the original CLB instances, which are not affected by the upgrade. If you need to upgrade the CLB service, you need to upgrade both CLB and TKE. Otherwise, the synchronization of all public network Service/Ingress add-ons in TKE may be affected. For how to upgrade CLB, see Upgrading to Domain Name-based CLB. For how to upgrade TKE Service/Ingress add-ons, submit a ticket.

Ingress API version support: extensions/v1beta1 and networking.k8s.io/v1beta1 ingress APIs are no longer provided in v1.22. networking.k8s.io/v1 APIs are available since v1.19 (which is v1.20 for TKE because TKE supports only even versions). For more information, see Kubernetes documentation.

Do not use the same CLB for TKE and CVM.

For a CLB managed by TKE, you cannot modify its listeners, forward paths, certificates, and backend-bound servers on the CLB console. Changes made on the CLB console will be automatically overwritten by TKE.

When using an existing CLB:

You can only use load balancers created through the CLB console, not balancers automatically created by TKE.

Do not use one CLB for multiple Ingresses.

Do not use the same CLB for Ingress and Service.

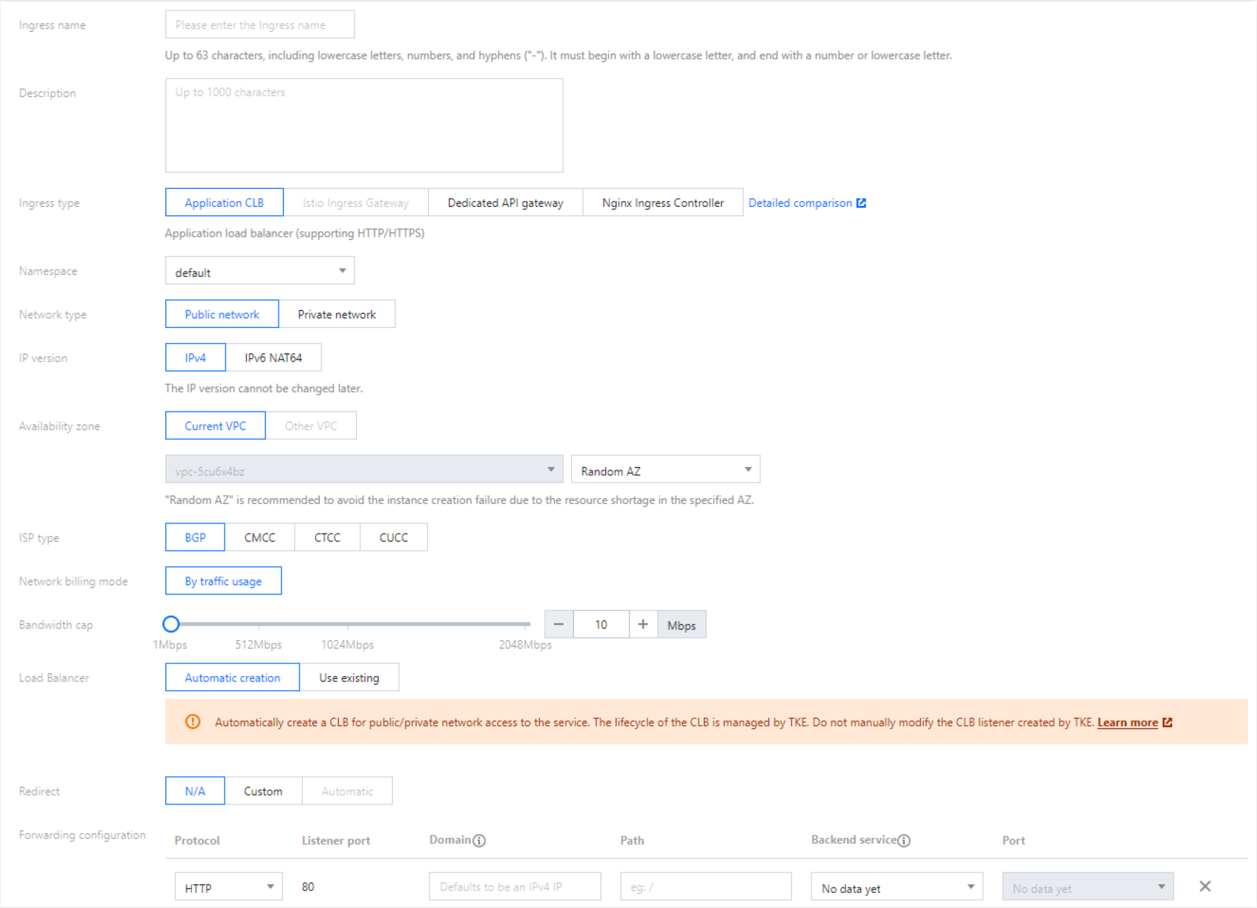
After you delete an Ingress, the real server bound to the reused CLB will need to be unbound manually. `tag tke-clusterId: cls-xxxx` will be kept for the CLB and will need to be cleared manually.

By default, you can create up to 50 forwarding rules under a single CLB instance. If you need more, submit a ticket to increase the quota.
The management and sync of configurations between Ingress and CLB instances are based on the resource object of the `LoadBalancerResource` type named the CLB ID. Do not perform any operations on this CRD; otherwise, the Ingress may fail.

Managing Ingress in Console

Creating an Ingress

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to go to the cluster management page.
3. Click the cluster ID where the Ingress needs to be created to go to the cluster management page.
4. Select **Service** > **Ingress** to go to the Ingress information page.
5. Click **Create** to go to the **Create an Ingress** page.



Ingress name	Please enter the Ingress name Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.
Description	Up to 1000 characters
Ingress type	[✓ Application CLB] Istio Ingress Gateway Dedicated API gateway Nginx Ingress Controller Detailed comparison Application load balancer (supporting HTTP/HTTPS)
Namespace	default
Network type	[✓ Public network] Private network
IP version	[✓ IPv4] IPv6 NAT64 The IP version cannot be changed later.
Availability zone	[✓ Current VPC] Other VPC vpc-5cu6x4bz Random AZ "Random AZ" is recommended to avoid the instance creation failure due to the resource shortage in the specified AZ.
ISP type	[✓ BGP] CMCC CTCC CUCC
Network billing mode	[✓ By traffic usage]
Bandwidth cap	1Mbps 512Mbps 1024Mbps 2048Mbps − 10 + Mbps
Load Balancer	[✓ Automatic creation] Use existing Automatically create a CLB for public/private network access to the service. The lifecycle of the CLB is managed by TKE. Do not manually modify the CLB listener created by TKE. Learn more
Redirect	[✓ N/A] Custom Automatic

Forwarding configuration

Protocol	Listener port	Domain	Path	Backend service	Port	
HTTP	80	Defaults to be an IPv4 IP	eg: /	No data yet	No data yet	×

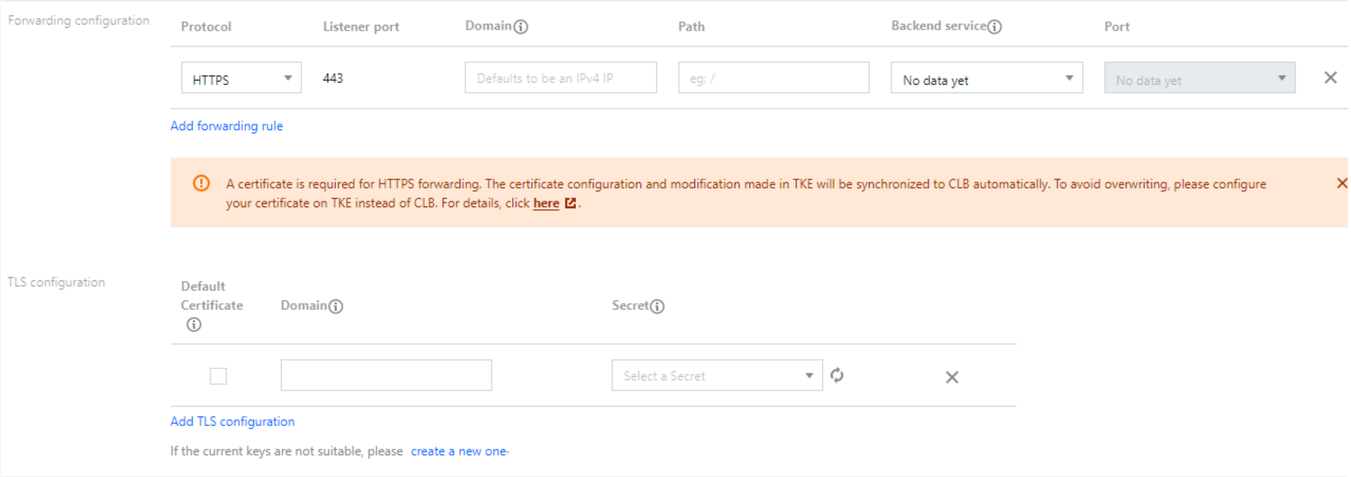
6. Set the Ingress parameters based on your actual needs. The key parameters are as follows:
Ingress name: Custom.
Network type: The default value is `Public network` . Select another network if needed.
IP Version: You can select IPv4 or IPv6 NAT64 as needed.

Load balancer: Create one automatically or use an existing CLB.

Namespace: Select the namespace based on your actual needs.

Forwarding Configuration: The default value of **Protocol** is **Http**. You can select a protocol as needed.

If you select **Https**, you need to bind the server certificate to ensure access security.



Forwarding configuration

Protocol	Listener port	Domain	Path	Backend service	Port
HTTPS	443	Defaults to be an IPv4 IP	eg: /	No data yet	No data yet

Add forwarding rule

A certificate is required for HTTPS forwarding. The certificate configuration and modification made in TKE will be synchronized to CLB automatically. To avoid overwriting, please configure your certificate on TKE instead of CLB. For details, click here.

TLS configuration

Default Certificate	Domain	Secret
□		Select a Secret

Add TLS configuration

If the current keys are not suitable, please create a new one.

For more information, see Certificate Requirements and Certificate Format Conversion.

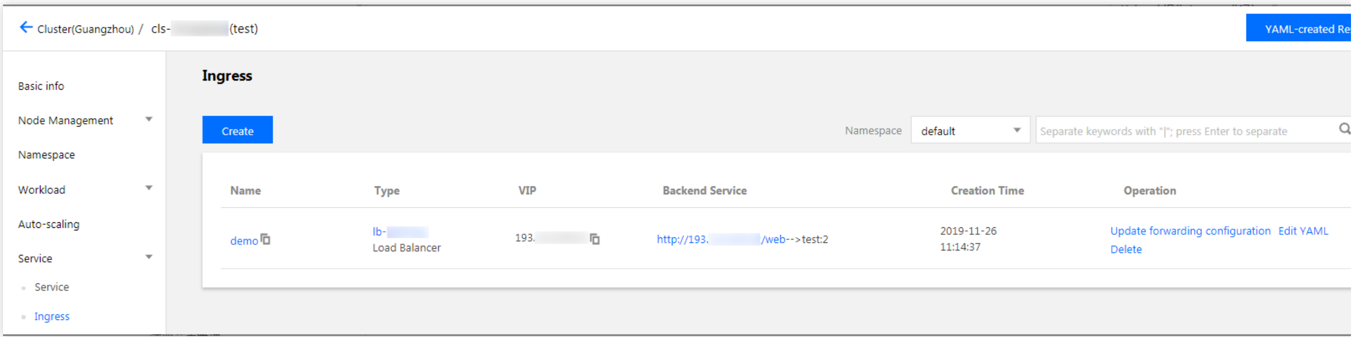
Forwarding configuration: Set this parameter as needed.

7. Click **Create Ingress** to create an Ingress.

Updating an Ingress

Updating YAML

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to go to the cluster management page.
3. Click the cluster ID for which you want to update the YAML to go to the cluster management page.
4. Select **Service** > **Ingress** to go to the Ingress information page.



Cluster(Guangzhou) / cls-[redacted](test)

Ingress

Namespace: default

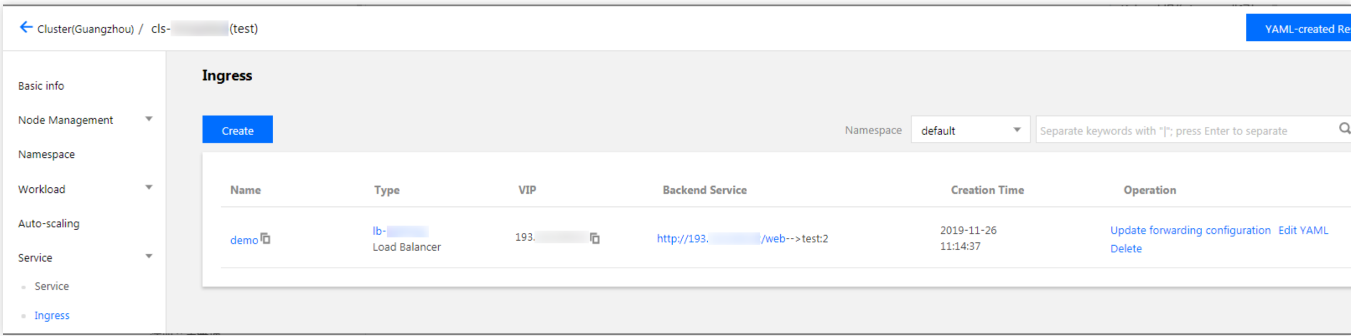
Separate keywords with "|"; press Enter to separate

Name	Type	VIP	Backend Service	Creation Time	Operation
demo	lb-[redacted] Load Balancer	193.[redacted]	http://193.[redacted]/web-->test:2	2019-11-26 11:14:37	Update forwarding configuration Edit YAML Delete

5. In the row of the Ingress for which you want to update YAML, click **Edit YAML** to go to the **Update an Ingress** page.
6. On the **Update an Ingress** page, edit YAML and click **Complete** to update YAML.

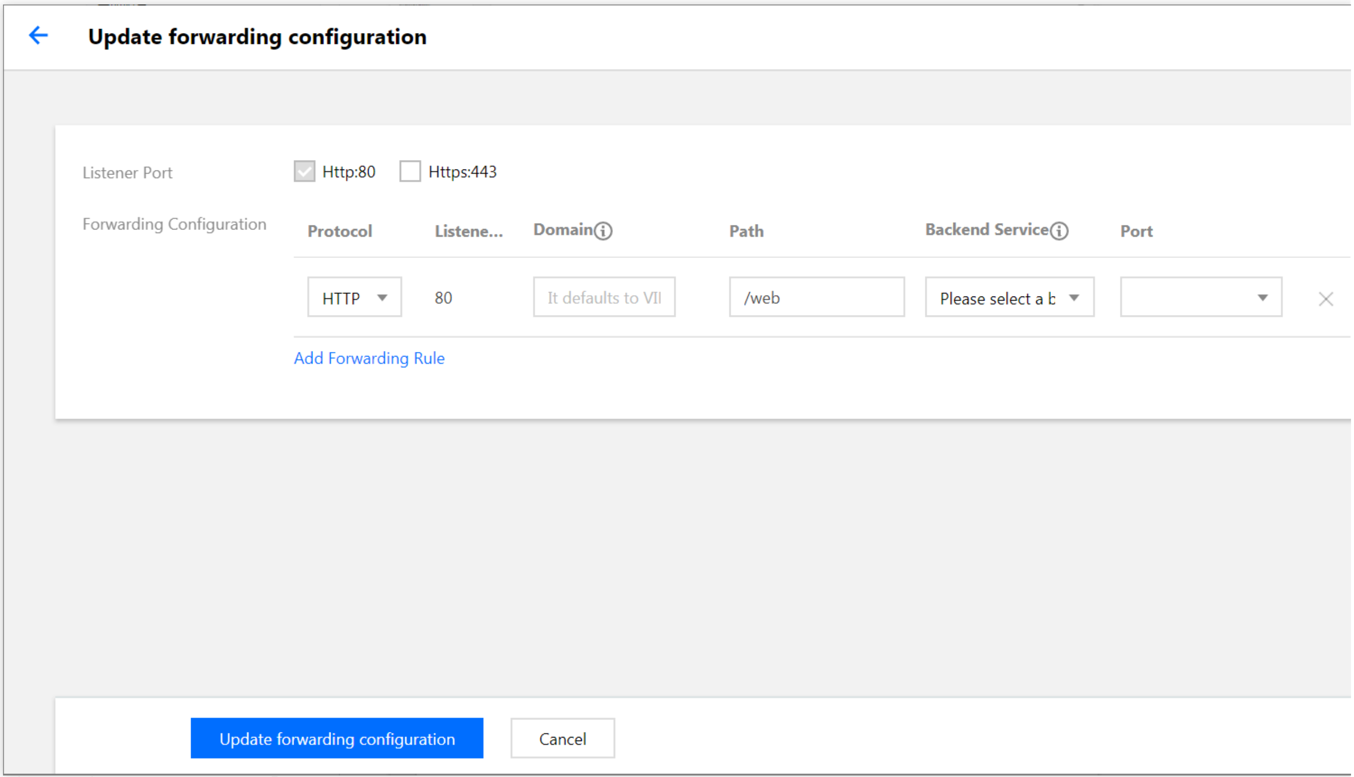
Updating a forwarding rule

1. On the cluster management page, click the cluster ID for which you want to update the YAML to go to the cluster management page.
2. Select **Service** > **Ingress** to go to the Ingress information page.



Name	Type	VIP	Backend Service	Creation Time	Operation
demo	lb-[redacted] Load Balancer	193.[redacted]	http://193.[redacted]/web-->test:2	2019-11-26 11:14:37	Update forwarding configuration Edit YAML Delete

3. In the row of the Ingress for which you want to update the forwarding rule, click **Update the forwarding configuration** to go to the **Update forwarding configuration** page as shown in the figure below:


Update forwarding configuration
Add Forwarding Rule
Update forwarding configuration
Cancel

Listener Port: ☑ Http:80 □ Https:443

Forwarding Configuration:

Protocol	Listene...	Domain	Path	Backend Service	Port
HTTP	80	It defaults to VII	/web	Please select a b	

4. Modify the forwarding configuration based on your actual needs and click **Update forwarding configuration** to complete the update.

Managing Ingresses Using Kubectl

YAML sample

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: qcloud ## Options: qcloud (CLB-type Ingress), ngin
    ## kubernetes.io/ingress.existLbId: lb-xxxxxxxx  ## Specify an existing load ba
    ## kubernetes.io/ingress.subnetId: subnet-xxxxxxxx  ## If you are creating a CL
  name: my-ingress
  namespace: default
spec:
  rules:
```

```
  - host: localhost
    http:
      paths:
      - backend:
          serviceName: non-service
          servicePort: 65535
        path: /
```

kind: Ingress resource type.

metadata: Basic information such as Ingress name and Label.

metadata.annotations: An additional description of the Ingress. You can set additional enhancements for TKE through this parameter.

spec.rules: Ingress forwarding rule, which can be configured to implement a simple routing service, domain name-based simple fan-out routing, default domain name for simple routing, and a securely configured routing service.

annotations: create an Ingress for public/private network access using an existing load balancer

If the existing application CLB is idle and you want to use it for an Ingress created by TKE or you want to use the same CLB within the cluster, you can set it using the following annotations:

Note

Please read the Notes before use.

```
metadata:
  annotations:
    kubernetes.io/ingress.existLbId: lb-6swtxxxx
```

annotations: create a private network Ingress of the CLB type

If you need to use a private network CLB, set it with the following annotations:

```
metadata:
  annotations:
    kubernetes.io/ingress.subnetId: subnet-xxxxxxxx
```

Notes

If you are using an account with **IP bandwidth packages**, you need to specify the following two annotations when creating a service accessible to the public network:

`kubernetes.io/ingress.internetChargeType` identifies the public network bandwidth billing method. Options include:

TRAFFIC_POSTPAID_BY_HOUR (bill-by-traffic)

BANDWIDTH_POSTPAID_BY_HOUR (bill-by-bandwidth)

`kubernetes.io/ingress.internetMaxBandwidthOut` identifies the bandwidth cap (value range: [1, 2000] Mbps).

Example:

```
metadata:
annotations:
  kubernetes.io/ingress.internetChargeType: TRAFFIC_POSTPAID_BY_HOUR
  kubernetes.io/ingress.internetMaxBandwidthOut: "10"
```

For more information on **IP bandwidth packages**, see Bandwidth Package Types.

Creating an Ingress

1. Prepare the Ingress YAML file as instructed in the YAML sample.
2. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
3. Run the following command to create the Ingress YAML file:

```
kubectl create -f Ingress YAML filename
```

For example, to create an Ingress YAML file named “my-ingress.yaml”, run the following command:

```
kubectl create -f my-ingress.yaml
```

4. Run the following command to check whether the creation is successful:

```
kubectl get ingress
```

If a message similar to the following is returned, the creation is successful.

```
NAME          HOSTS        ADDRESS    PORTS      AGE
clb-ingress   localhost               80         21s
```

Updating an Ingress

Method 1

Run the following command to update an Ingress:

```
kubectl edit  ingress/[name]
```

Method 2

1. Manually delete the old Ingress.
2. Run the following command to recreate an Ingress:

```
kubectl create/apply
```

Using an Existing CLB for Direct Pod Connection

Last updated：2022-12-13 16:06:31

Tencent Kubernetes Engine (TKE) allows you to use the `kubernetes.io/ingress.existLbId:<loadbalanceid>` annotation to specify an existing CLB instance associated with an ingress.

> Note：
> Different from services that can reuse the same CLB instance, ingresses cannot use the same CLB instance.

Notes

- Your TKE containers and CVMs cannot share a CLB.
- You cannot operate on CLB listeners and backend servers managed by `Ingress Controller` in the CLB console. Your updates will be overwritten by `Ingress Controller` .
- When an existing CLB is used:
  - Multiple ingresses cannot reuse the same CLB.
  - The specified CLB cannot contain any existing listeners. If a listener exists, delete it in advance.
  - Only CLBs created in the CLB console can be used. CLBs automatically created and managed by `Service Controller` cannot be used. This means a service and an ingress cannot use the same CLB.
  - `Ingress Controller` does not manage CLB resources. This means that, when an ingress is deleted, the CLB will not be deleted or recycled.

Use Cases

Using a monthly subscription CLB to provide external services

When `Ingress Controller` manages the CLB lifecycle, you can only purchase pay-as-you-go CLBs. However, monthly subscription CLBs have price advantages and are preferentially selected by users who need to continuously use a CLB.

In such scenarios, you can independently purchase and manage a CLB, use an annotation to enable an ingress to use an existing CLB, and detach CLB lifecycle management from `Ingress Controller` . The following shows a sample.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.existLbId: lb-mgzu3mpx
name: nginx-ingress
spec:
rules:
- http:
paths:
- backend:
serviceName: nginx-service
servicePort: 80
path: /
```

The `kubernetes.io/ingress.existLbId: lb-mgzu3mpx` annotation indicates that the ingress will use the existing CLB `lb-mgzu3mpx` to configure the ingress service.

Using TKEServiceConfig to Configure CLBs

Last updated：2023-02-03 16:33:03

TkeServiceConfig

`TkeServiceConfig` is a custom resource definition (CRD) provided by TKE to help you manage the various configurations of CLB with an Ingress more flexibly.

Use cases

The CLB parameters and features that cannot be defined by the semantics of `Ingress YAML` can be configured through `TkeServiceConfig`.

Configuration instructions

`TkeServiceConfig` helps you quickly configure CLB. You can specify a target configuration for application to an Ingress through the Ingress annotation **`ingress.cloud.tencent.com/tke-service-config:&lt;config-name>`**.

> Note：
> The `TkeServiceConfig` resource needs to be in the same namespace as the Ingress.

`TkeServiceConfig` doesn't help you configure and modify the protocol, port, domain name, and forwarding path; instead, you need to describe them in the configuration to specify the forwarding rule for delivery by the configuration.

There can be multiple domain names under each layer-7 listener and multiple forwarding paths under each domain name. Therefore, you can declare multiple combinations of domain name and forwarding rule configurations in `TkeServiceConfig`. Currently, configurations are mainly provided for CLB health check and backend access.

- The configuration can be accurately delivered to the corresponding listener by specifying the protocol and port:
- `spec.loadBalancer.l7Listeners.protocol` : layer-7 protocol
- `spec.loadBalancer.l7Listeners.port` : listening port
- By specifying the protocol, port, domain name, and access path, you can set configurations at the forwarding rule level, such as for backend health check and load balancing methods.
- `spec.loadBalancer.l7Listeners.protocol` : layer-7 protocol
- `spec.loadBalancer.l7Listeners.port` : listening port
- `spec.loadBalancer.l7Listeners.domains[].domain` : domain name

- `spec.loadBalancer.l7Listeners.domains[].rules[].url` : forwarding path
- `spec.loadBalancer.l7listeners.protocol.domain.rules.url.forwardType` : specified backend protocol
  - A backend protocol is the protocol between a CLB instance and the real server. If you select HTTP as the backend protocol, you need to deploy HTTP service for the real server. If you select HTTPS as the backend protocol, you need to deploy HTTPS service for the real server. Encryption and decryption of HTTPS service will consume more resources. For more information, see Configuring a HTTPS Listener for a CLB Instance.

> Note：
> When your domain name is configured as the default value, i.e., public or private VIP, you can configure by entering a null value in the `domain` field.

Association between Ingress and TkeServiceConfig

1. If you set `**ingress.cloud.tencent.com/tke-service-config-auto:&lt;true>**` when creating an Ingress, `&lt;IngressName>-auto-ingress-config` will be created automatically. You can also specify the `TkeServiceConfig` you created on your own directly through `**ingress.cloud.tencent.com/tke-service-config:&lt;config-name>**` . The two annotations cannot be used at the same time.
2. The name of the custom configuration you use for a Service/Ingress cannot be suffixed with `-auto-service-config` or `-auto-ingress-config` .
3. The automatically created `TkeServiceConfig` has the following sync behaviors:

- When a layer-7 forwarding rule is added during Ingress resource update, `Ingress-Controller` will automatically add the corresponding `TkeServiceConfig` configuration segment for the rule if it doesn't exist.
- When a layer-7 forwarding rule is deleted, the `Ingress-Controller` component will automatically delete the corresponding `TkeServiceConfig` segment.
- When an Ingress resource is deleted, the `TkeServiceConfig` will also be deleted.
- When you modify the default `TkeServiceConfig` of the Ingress, the `TkeServiceConfig` content will also be applied to CLB.

4. You can also create the desired CLB configuration as instructed in the following `TkeServiceConfig` configuration reference, which is imported by the Service through the `**ingress.cloud.tencent.com/tke-service-config:&lt;config-name>**` annotation.
5. A manually created `TkeServiceConfig` has the following sync behaviors:

- When you use a configuration annotation in the Ingress, CLB will immediately set sync.
- When you delete a configuration annotation in the Ingress, CLB will remain unchanged.
- When you modify the `TkeServiceConfig` configuration, CLB of the Ingress that imports the configuration will set sync based on the new `TkeServiceConfig` .
- If the Ingress listener cannot find the corresponding configuration, the listener will not be modified.
- If the Ingress listener finds the corresponding configuration, but the configuration doesn't contain declared attributes, the listener will not be modified.

License request example

Sample deployment: jetty-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: jetty
name: jetty-deployment
namespace: default
spec:
progressDeadlineSeconds: 600
replicas: 3
revisionHistoryLimit: 10
selector:
matchLabels:
app: jetty
strategy:
rollingUpdate:
maxSurge: 25%
maxUnavailable: 25%
type: RollingUpdate
template:
metadata:
creationTimestamp: null
labels:
app: jetty
spec:
containers:
- image: jetty:9.4.27-jre11
imagePullPolicy: IfNotPresent
name: jetty
ports:
- containerPort: 80
```

```
protocol: TCP
- containerPort: 443
protocol: TCP
resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
dnsPolicy: ClusterFirst
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
```

Sample Service: jetty-service.yaml

```
apiVersion: v1
kind: Service
metadata:
name: jetty-service
namespace: default
spec:
ports:
- name: tcp-80-80
port: 80
protocol: TCP
targetPort: 80
- name: tcp-443-443
port: 443
protocol: TCP
targetPort: 443
selector:
app: jetty
type: NodePort
```

This example contains the following configuration:

Service `NodePort` type, with two TCP services declared, one on port 80 and the other on port 443.

Sample Ingress: jetty-ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.rule-mix: "true"
kubernetes.io/ingress.http-rules: '[{"path":"/health","backend":{"serviceName":"jetty-service","servicePort":"80"}}]'
kubernetes.io/ingress.https-rules: '[{"path":"/","backend":{"serviceName":"jetty-service","servicePort":"443","host":"sample.tencent.com"}}]'
ingress.cloud.tencent.com/tke-service-config: jetty-ingress-config
# Specify the existing `tke-service-config`
# ingress.cloud.tencent.com/tke-service-config-auto: "true"
# Automatically create a `tke-service-config`
name: jetty-ingress
namespace: default
spec:
rules:
- http:
paths:
- backend:
serviceName: jetty-service
servicePort: 80
path: /health
- host: "sample.tencent.com"
http:
paths:
- backend:
serviceName: jetty-service
servicePort: 443
path: /
tls:
- secretName: jetty-cert-secret
```

This example contains the following configuration:

- Two different protocols are used together. The default domain name (public IP) is used to expose an HTTP service, and the `sample.tencent.com` domain name is used to expose an HTTPS service.
- The forwarding path of the HTTP service is `/health` , and that of the HTTPS service is `/` .
- The `jetty-ingress-config` CLB configuration is used.

Sample `TkeServiceConfig` : jetty-ingress-config.yaml

```
apiVersion: cloud.tencent.com/v1alpha1
kind: TkeServiceConfig
metadata:
name: jetty-ingress-config
namespace: default
spec:
loadBalancer:
l7Listeners:
- protocol: HTTP
port: 80
domains:
```

```
- domain: "" # When `domain` is null, the VIP is used as the domain name
rules:
- url: "/health"
forwardType: HTTP # It specifies HTTP as the backend protocol
healthCheck:
enable: false
- protocol: HTTPS
port: 443
defaultServer: "sample.tencent.com" # Default domain name
keepaliveEnable: 1 # Enable persistent connection for the listener
domains:
- domain: "sample.tencent.com"
rules:
- url: "/"
forwardType: HTTPS # It specifies HTTPS as the backend protocol
session:
enable: true
sessionExpireTime: 3600
healthCheck:
enable: true
intervalTime: 10 # `intervalTime` must be greater than `timeout`; otherwise, an error will occur.
timeout: 5 # `timeout` must be smaller than `intervalTime`; otherwise, an error will occur.
healthNum: 2
unHealthNum: 2
httpCheckPath: "/checkHealth"
httpCheckDomain: "sample.tencent.com" # Note: the health check must use a fixed domain name for detection. If you enter a wildcard domain name in `.spec.loadBalancer.l7Listeners.protocol.domains.domain`, be sure to use the `httpCheckDomain` field to specify the domain name that requires health check; otherwise, the wildcard domain name does not support health check.
httpCheckMethod: HEAD
scheduler: WRR
```

This example contains the following configuration:

The name of the `TkeServiceConfig` is `jetty-ingress-config` , and in the layer-7 listener configuration, two configuration segments are declared:

1. The HTTP listener of port 80 will be configured, including the configuration of domain name, which is the default domain name and corresponds to the VIP of CLB.
   The health check feature under the `/health` path is disabled.
2. The HTTPS listener of port 443 will be configured, including the configuration of domain name, which is `sample.tencent.com` . Under this domain name, only a forwarding rule configuration with the forwarding path of `/` is described, which contains the following:

- The health check feature is enabled, with the health check interval set to 10s, the healthy threshold to 2 times, and the unhealthy threshold also to 2 times, health checks are performed through `HEAD` requests, the check path is `/checkHealth` , and the check domain name is `sample.tencent.com` .
- The session persistence feature is enabled, with the timeout period set to 3,600s.
- The forwarding policy is configured as "weighted round robin".

kubectl configuration commands

```
➜ kubectl apply -f jetty-deployment.yaml
➜ kubecti apply -f jetty-service.yaml
➜ kubectl apply -f jetty-ingress.yaml
➜ kubectl apply -f jetty-ingress-config.yaml
➜ kubectl get pods
NAME READY STATUS RESTARTS AGE
jetty-deployment-8694c44b4c-cxscn 1/1 Running 0 8m8s
jetty-deployment-8694c44b4c-mk285 1/1 Running 0 8m8s
jetty-deployment-8694c44b4c-rjrtm 1/1 Running 0 8m8s
# Get the `TkeServiceConfig` configuration list
➜ kubectl get tkeserviceconfigs.cloud.tencent.com
NAME AGE
jetty-ingress-config 52s
# Update and modify the `TkeServiceConfig` configuration
➜ kubectl edit tkeserviceconfigs.cloud.tencent.com jetty-ingress-config
tkeserviceconfigs.cloud.tencent.com/jetty-ingress-config edited
```

Ingress Cross-region Binding

Last updated：2022-03-30 17:52:59

Overview

When you use the Ingress of the CLB type, the CLB is generated for random availability zone in the VPC where the cluster resides by default. Currently, the CLB Ingress of TKE allows you to specify availability zones, including availability zones in other regions. This document describes how to bind and specify availability zones for CLB Ingress across regions via the console and YAML.

Operations

- The cross-region access or cross-VPC access of CLB must be supported. That is, the VPC where the CLB resides and the VPC where the cluster resides are not in the same VPC.
- The availability zone of CLB has realized unified management of resources.

> Note：
>
> 1. If you need to use the CLB that is not in the same VPC as this cluster, you need to connect the VPCs of the current cluster and the CLB via CCN.
> 2. After the VPCs are connected, please submit a ticket to apply for this feature.
> 3. You should enter the region ID in the following YAML. You can check the region ID in Regions and Availability Zones.

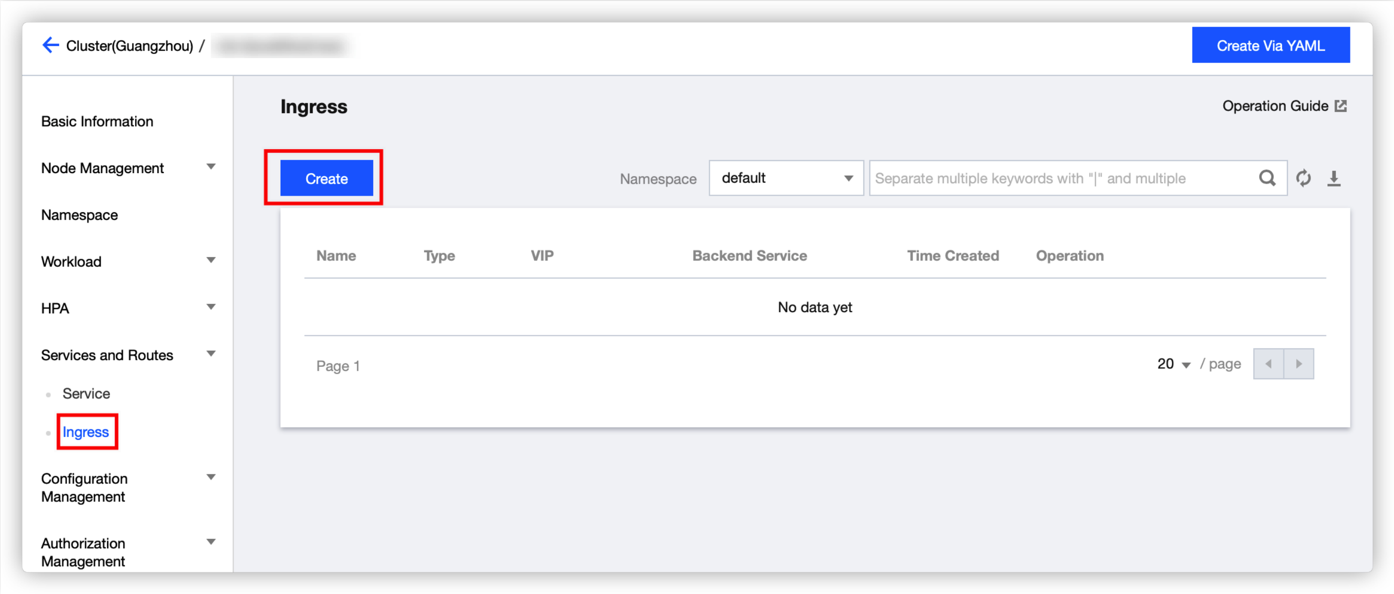
Directions

You can bind and specify availability zones for CLB Ingress across regions via the console and YAML.

- Console
- YAML

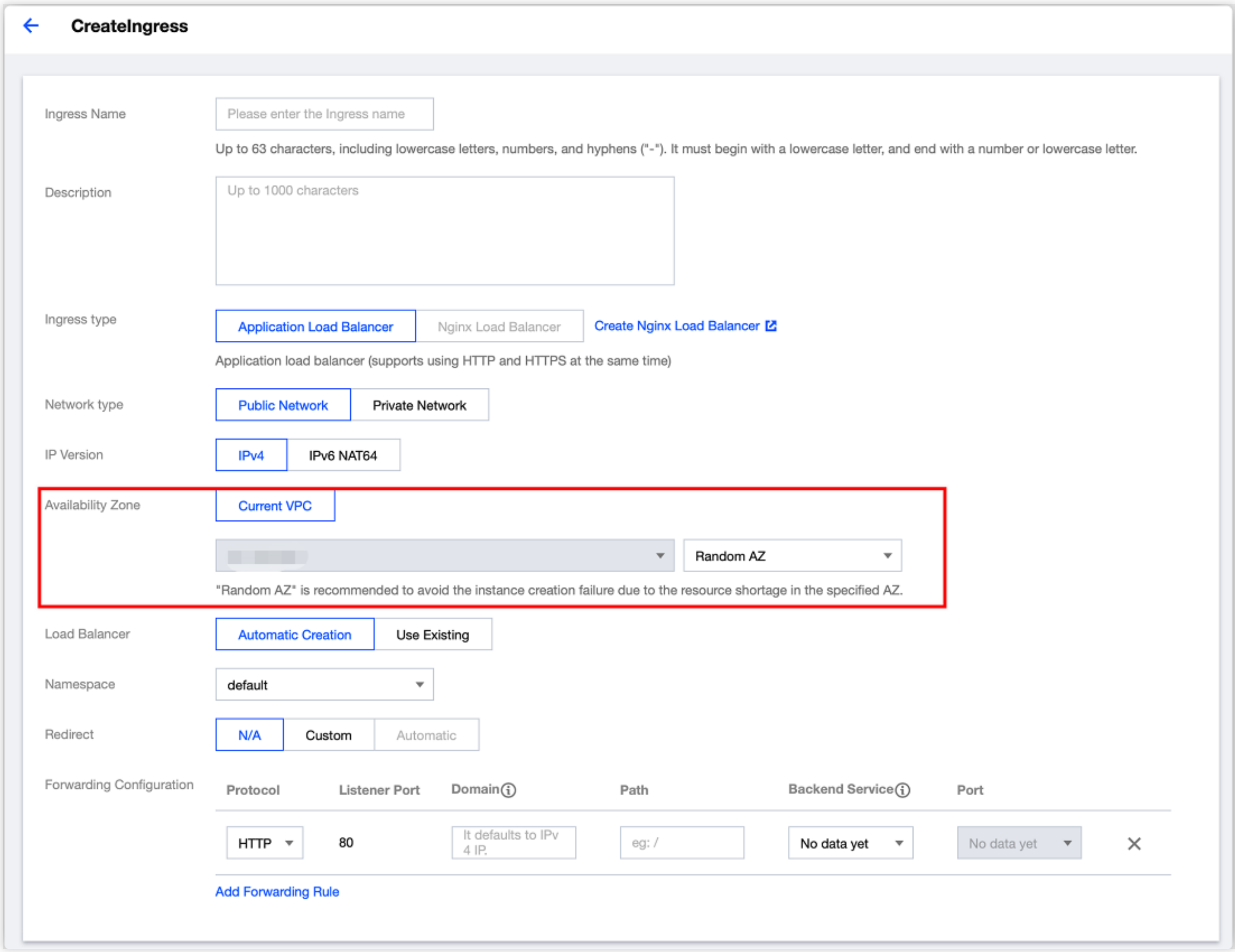
1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the "Cluster Management" page, click the cluster ID whose Ingress object needs to be modified.

3. On the cluster details page, select **Services and Routes** > **Ingress** in the left sidebar, as shown in the figure below:



4. Click **Create** and configure the availability zone rules on the **Create Ingress** page. The configuration rules are as follows:
   - **Current VPC**: use the CLB in the VPC where the cluster resides. **Random AZ** is recommended to avoid the instance creation failure due to the resource shortage in the specified availability zone.
   - **Other VPC**: it only supports the VPC that has connected to the VPC of the current cluster via CCN. **Random AZ** is recommended to avoid the instance creation failure due to the resource shortage in the specified availability

zone.



CreateIngress

Ingress Name: Please enter the Ingress name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Description: Up to 1000 characters

Ingress type: Application Load Balancer | Nginx Load Balancer | Create Nginx Load Balancer

Application load balancer (supports using HTTP and HTTPS at the same time)

Network type: Public Network | Private Network

IP Version: IPv4 | IPv6 NAT64

Availability Zone: Current VPC

[redacted] | Random AZ

"Random AZ" is recommended to avoid the instance creation failure due to the resource shortage in the specified AZ.

Load Balancer: Automatic Creation | Use Existing

Namespace: default

Redirect: N/A | Custom | Automatic

Forwarding Configuration

Protocol	Listener Port	Domain	Path	Backend Service	Port
HTTP	80	It defaults to IPv 4 IP.	eg: /	No data yet	No data yet

Add Forwarding Rule

Graceful Ingress Shutdown

Last updated：2022-09-26 16:12:50

Overview

In scenarios where a Pod is connected directly at the access layer, when the backend performs a rolling update, or the backend Pod is deleted, if you delete the Pod directly from the CLB backend, unprocessed requests that have been received by it cannot be processed.
Particularly, in persistent connection scenarios, such as meeting business, if the Pod of the workload is updated or deleted directly, the meeting will be interrupted.

Use Cases

Note：
This is only effective in the direct access mode. Check whether your cluster supports direct access.

- When a Pod quits gracefully during a workload update, the client will not perceive the jitters and errors generated during the update (if any).
- When a Pod needs to be deleted, it can process the received requests, and inbound traffic is turned off while outbound traffic is still on. Outbound traffic will not be turned off until all existing requests are processed and the Pod is deleted.

Directions

Step 1. Use an annotation to indicate the use of graceful shutdown

Below is an example of using an annotation to indicate the use of graceful shutdown. For the detailed Ingress annotations, see Ingress Annotation.

```
kind: Ingress
apiVersion: v1
metadata:
annotations:
ingress.cloud.tencent.com/direct-access: "true" ## Enable CLB-to-Pod direct acce
ss.
```

```
ingress.cloud.tencent.com/enable-grace-shutdown: "true"` # It indicates the usag
e of graceful shutdown.
name: my-Ingress
spec:
selector:
app: MyApp
...
```

Step 2. Use `preStop` and `terminationGracePeriodSeconds`

Step 2 involves using `preStop` and `terminationGracePeriodSeconds` in the workload that requires graceful shutdown.

Container termination process

The following describes the container termination process in a Kubernetes environment:

1. If a deleted Pod contains `DeletionTimestamp` and is in **Terminating** status, the weight of the Pod on the CLB backend is adjusted to `0` .
2. kube-proxy updates the forwarding rule and removes the Pod from the endpoint list of the Ingress instance.
3. If a `preStop` hook is configured for the Pod, it will be executed.
4. kubelet will send a SIGTERM signal to each container in the Pod to ask the container processes to stop gracefully.
5. Wait for the container processes to stop completely. If a process has not stopped completely after `terminationGracePeriodSeconds` (30s by default) elapses, a SIGKILL signal will be sent to forcibly stop it.
6. All container processes are terminated, and the Pod resources are cleared.

Specific steps

1. **Use `preStop`**

   To implement graceful termination, you must process the SIGTERM signal in your business code. The main logic is to stop accepting new traffic, continue to process existing traffic, and quit after all connections are closed. For more information, see Go by Example: Signals.

   If the SIGTERM signal is not processed in your business code, or if you cannot control the used third-party library or system to add the logic of graceful termination, you can also try configuring `preStop` for the Pod to implement such logic as shown below:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
   name: lifecycle-demo
   spec:
   ```

```
containers:
- name: lifecycle-demo-container
image: nginx
lifecycle:
preStop:
exec:
command:
- /clean.sh
```

For more information on how to configure `preStop`, see Lifecycle.

In certain extreme cases, new connections may still be forwarded within a short period of time after the Pod is deleted. This is because kubelet and kube-proxy watch that the Pod is deleted at the same time, and kubelet may have stopped the containers before kube-proxy syncs the rules. Normally, an application will no longer accept new connections after it receives `SIGTERM` , and it will only keep the existing connections for processing, which may cause some requests to fail at the moment when the Pod is deleted.

In view of the above, you can use `preStop` to make the Pod sleep for a short while first and then start to stop the container processes after kube-proxy completes the rule sync as shown below:

```
apiVersion: v1
kind: Pod
metadata:
name: lifecycle-demo
spec:
containers:
- name: lifecycle-demo-container
image: nginx
lifecycle:
preStop:
exec:
command:
- sleep
- 5s
```

2. **Use `terminationGracePeriodSeconds` to adjust the termination grace period**

If you need a long termination grace period (`preStop` and the business process termination may take more than 30s in total), you can customize `terminationGracePeriodSeconds` as shown below based on the actual situation so as to avoid being stopped by SIGKILL prematurely:

```
apiVersion: v1
kind: Pod
```

```
metadata:
name: grace-demo
spec:
terminationGracePeriodSeconds: 60 # The termination grace period is 30s by default, and you can set a longer period.
containers:
- name: lifecycle-demo-container
image: nginx
lifecycle:
preStop:
exec:
command:
- sleep
- 5s
```

Capabilities

Graceful shutdown sets the weight on the CLB backend to `0` only when a Pod is deleted. If a running Pod becomes unhealthy, setting the weight to `0` on the backend can reduce the risk of service unavailability.

You can use the `ingress.cloud.tencent.com/enable-grace-shutdown-tkex: "true"` annotation to implement graceful shutdown.

The annotation will check whether an endpoint in the Endpoint object is `not-ready` , and if so, the annotation will set the weight on the CLB backend to `0` .

Ingress Redirection

Last updated：2021-10-13 14:29:33

Overview

Domain name redirection means when you access a URL in a browser, the web server is set to automatically redirect to another URL.

Use Cases

- The website supports HTTP and HTTPS; for example, `http://tencent.com` and `https://tencent.com` need to point to the same web service.
- The domain name of the website has changed, for example, from `https://tengxun.com` to `https://tencent.com` , and the two domain names point to the same web service.
- The website content has been partially adjusted and is no longer accessible at the original URL; in this case, it can be redirected to a new URL that provides services.

> Note
> - After you use redirection, there will be an additional annotation, which indicates that the forwarding rule of the Ingress is managed by TKE and cannot be deleted or modified subsequently; otherwise, it will conflict with the redirection rule set in CLB.
>
> ```
> ingress.cloud.tencent.com/rewrite-support: "true"
> ```
>
> - If a letter is used to represent the domain name address, and A has been redirected to B, then:
>   - A cannot be redirected to C (unless you delete the old redirection relationship first and then create a new one).
>   - B cannot be redirected to any other address.
>   - A cannot be redirected to A.

There are two redirection methods:

- **Automatic redirection**: you need to create an `HTTPS:443` listener first and then create a forwarding rule under it. When you call this API, the system will automatically create an `HTTP:80` listener (if it doesn't exist) and create a forwarding rule under it, which correspond to the various configurations under the `HTTPS:443` listener, such as the domain name.

- **Manual redirection**: you can manually configure the original access address and redirection address, and the system will automatically redirect requests made to the original access address to the destination address at the corresponding path. Multiple paths can be configured under the same domain name as a redirection policy to implement automatic redirection of requests between HTTP and HTTPS.

Notes

- If you don't have the TKE Ingress redirection annotation declaration, the original logic that doesn't manage redirection rules will be compatible; that is, you can configure redirection rules in the CLB console, and TKE Ingress doesn't process such rules.
- If you don't have the TKE Ingress redirection annotation declaration, due to the redirection protection restrictions of CLB, if the forwarding configuration A is redirected to the forwarding configuration B, you must delete the redirection rule first before you can delete the forwarding configuration B.
- If you use the TKE Ingress redirection annotation declaration, all redirection rules under CLB are managed by TKE Ingress and take effect only in the relevant annotations in TKE Ingress. In this case, if you modify the redirection configuration in the CLB console, the configuration will eventually be overwritten by the forwarding rule configured in TKE Ingress.

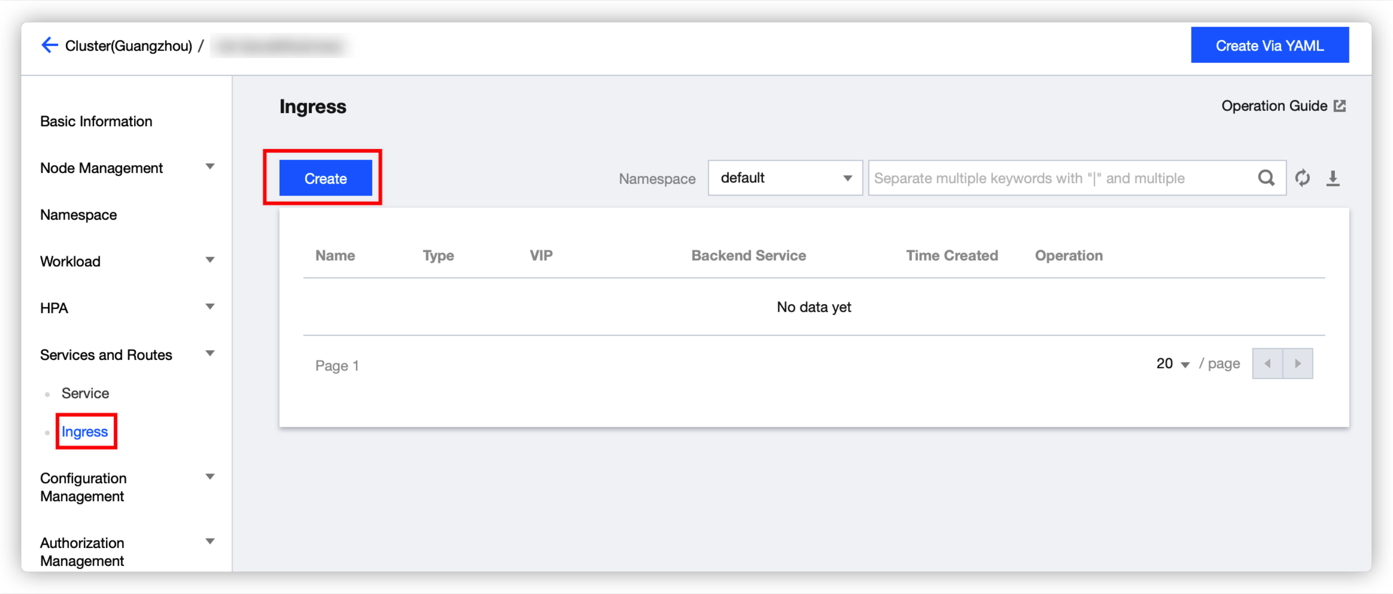
Directions

An Ingress supports two redirection configuration ways: console and YAML, as detailed below:

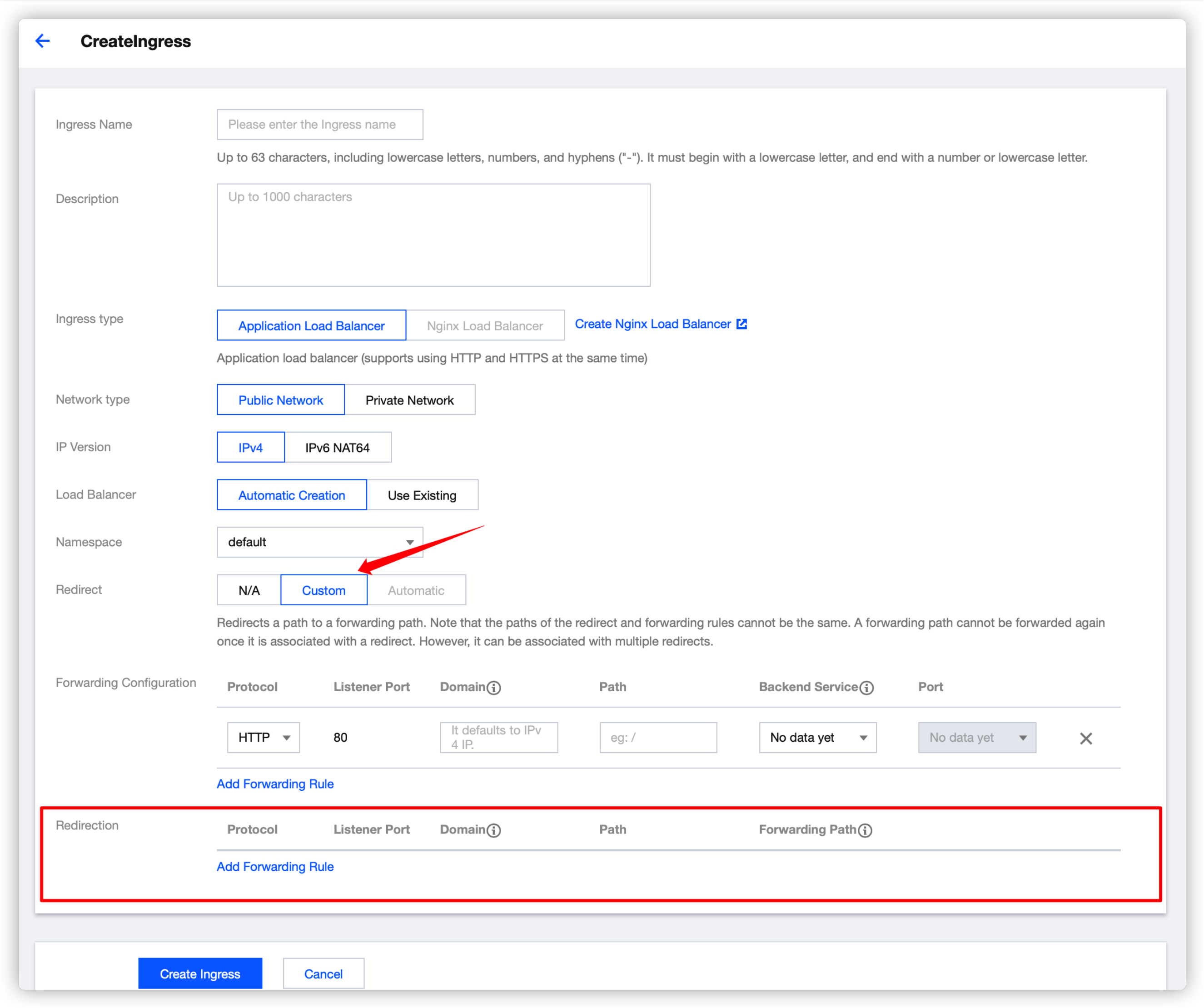
- Console
- YAML

1. Log in to the TKE console and select **Cluster** on the left sidebar.
2. On the **Cluster Management** page, select the ID of the cluster for which to modify the Ingress.

3. On the cluster details page, select **Services and Routes** > **Ingress** as shown below:



4. Click **Create** and configure the relevant redirection rule on the **Create Ingress** page as shown below:
   - **None**: no redirection rules are used.
   - **Manual**: the **Redirection Forwarding Configuration** section will appear under **Forwarding Configuration**.
     - You can enter information in **Forwarding Configuration** just like for a general Ingress, with the backend being a certain service.
     - You can also enter information in **Redirection Forwarding Configuration** just like for a general Ingress, but the backend is a certain path in a certain "forwarding configuration".



CreateIngress

Ingress Name: Please enter the Ingress name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Description: Up to 1000 characters

Ingress type: Application Load Balancer | Nginx Load Balancer | Create Nginx Load Balancer

Application load balancer (supports using HTTP and HTTPS at the same time)

Network type: Public Network | Private Network

IP Version: IPv4 | IPv6 NAT64

Load Balancer: Automatic Creation | Use Existing

Namespace: default

Redirect: N/A | Custom | Automatic

Redirects a path to a forwarding path. Note that the paths of the redirect and forwarding rules cannot be the same. A forwarding path cannot be forwarded again once it is associated with a redirect. However, it can be associated with multiple redirects.

Forwarding Configuration

Protocol	Listener Port	Domain	Path	Backend Service	Port
HTTP	80	It defaults to IPv4 IP.	eg: /	No data yet	No data yet

Add Forwarding Rule

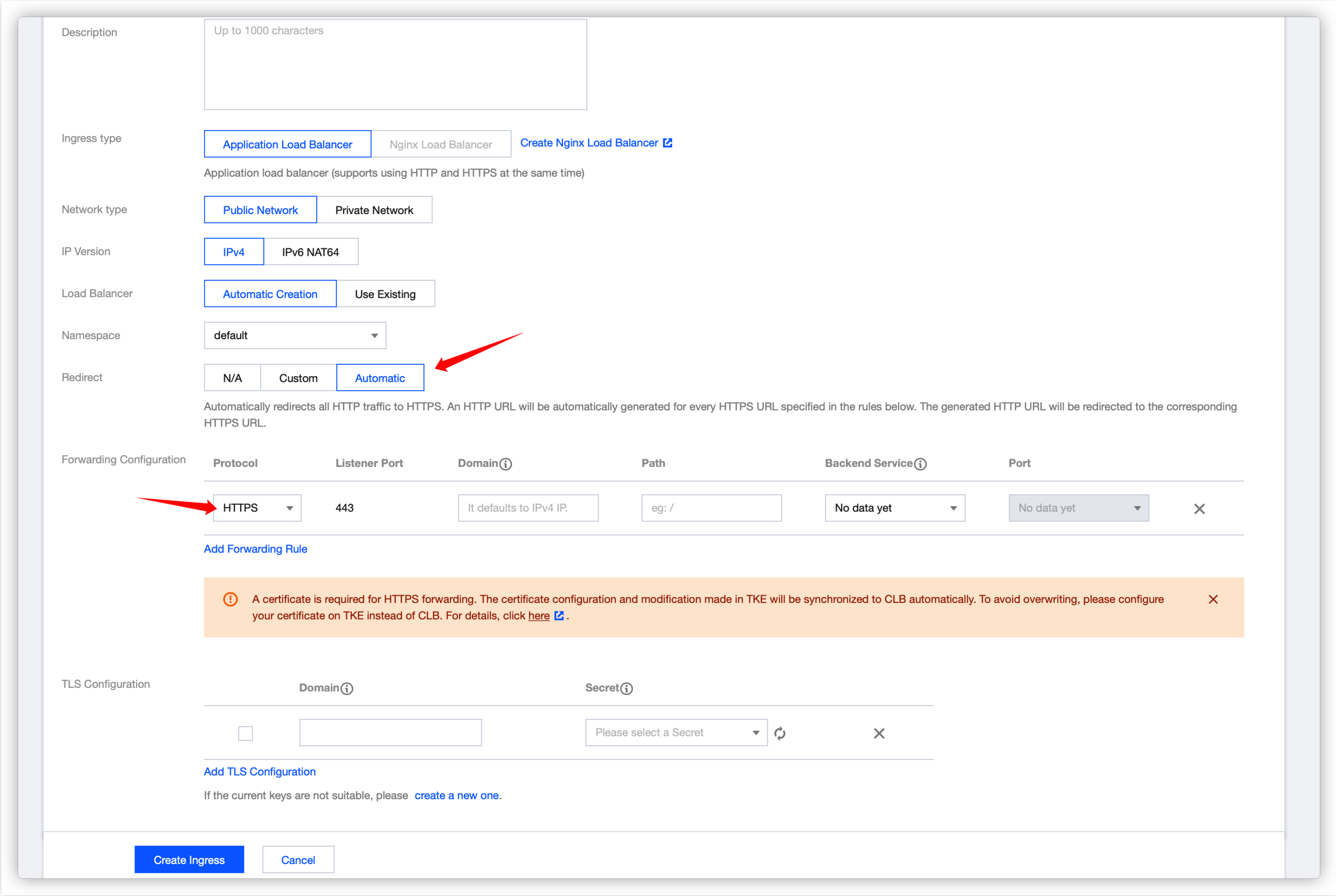
Redirection

Protocol	Listener Port	Domain	Path	Forwarding Path

Add Forwarding Rule

Create Ingress | Cancel

- **Auto**: it only takes effect for the "HTTPS" path in **Forwarding Configuration**. The same path with the "HTTP" protocol will be generated automatically, and only the protocols are different. The forwarding rule for the "HTTP" path is automatically redirected to the "HTTPS" path.



Description: Up to 1000 characters

Ingress type: Application Load Balancer | Nginx Load Balancer | Create Nginx Load Balancer

Application load balancer (supports using HTTP and HTTPS at the same time)

Network type: Public Network | Private Network

IP Version: IPv4 | IPv6 NAT64

Load Balancer: Automatic Creation | Use Existing

Namespace: default

Redirect: N/A | Custom | Automatic

Automatically redirects all HTTP traffic to HTTPS. An HTTP URL will be automatically generated for every HTTPS URL specified in the rules below. The generated HTTP URL will be redirected to the corresponding HTTPS URL.

Forwarding Configuration

Protocol	Listener Port	Domain	Path	Backend Service	Port
HTTPS	443	It defaults to IPv4 IP.	eg: /	No data yet	No data yet

Add Forwarding Rule

A certificate is required for HTTPS forwarding. The certificate configuration and modification made in TKE will be synchronized to CLB automatically. To avoid overwriting, please configure your certificate on TKE instead of CLB. For details, click here .

TLS Configuration

	Domain	Secret
□		Please select a Secret

Add TLS Configuration

If the current keys are not suitable, please create a new one.

Create Ingress | Cancel

Ingress Certificate Configuration

Last updated：2023-05-05 10:38:21

Overview

This document describes how to use an Ingress certificate. You can configure an Ingress certificate in the following scenarios:

If HTTPS is selected as the listener protocol when you create an Ingress, selecting a proper server certificate helps ensure access security.

If you bind the same certificate for all HTTPS domain names, you can configure a certificate with all HTTPS rules in the Ingress to simplify updates.

Binding different certificates to different domain names helps improve the SSL/TLS performance of the server and client.

Reminders

You need to create the certificate to be configured in advance. For more information, see Creating a server certificate in the console.

You need to set the Ingress certificate by using a Secret. Tencent Kubernetes Engine (TKE) Ingress creates a Secret with the same name as the certificate by default, which contains the certificate ID.

If you want to change the certificate, it is recommended that you create a certificate on the certificate platform and update the Secret certificate ID. Because the cluster add-on is synced based on the Secret statement, if you update the certificate on other certificate services or CLB services, the update will be restored by Secret.

The Secret certificate resource must be in the same namespace as the Ingress resource.

When you create the Ingress certificate, a Secret certificate resource with the same name will be created automatically by the console. If the Secret resource name already exists, the Ingress certificate cannot be created.

Generally, when you create an Ingress certificate, the certificate resource associated with a Secret will not be reused. However, you still can create an Ingress certificate that reuses the certificate resource associated with the Secret. When the Secret is updated, all Ingress certificates that are associated with the Secret are also updated.

Once you add a matching certificate for a domain name, the CLB SNI feature is enabled and cannot be disabled. If the domain name corresponding to the certificate is deleted, the certificate will match the HTTPS domain name corresponding to the Ingress by default.

Classic CLB does not support domain name- or URL-based forwarding. An Ingress that is created through Classic CLB cannot be configured with multiple certificates.

Example

TKE allows you to configure a certificate for a CLB HTTPS listener that is created for an Ingress by using the `spec.tls` field in the Ingress. Where, `secretName` indicates a Kubernetes Secret resource that contains a Tencent Cloud certificate ID, as shown in the following example:

Ingress

Creating via YAML:

```
spec:
    tls:
    - hosts:
      - www.abc.com
      secretName: secret-tls-2
```

Secret

Creating via YAML

Creating via the console

```
apiVersion: v1
```

```
stringData:
    qcloud_cert_id: Xxxxxxxx ## Set the certificate ID as Xxxxxxxx
    qcloud_ca_cert_id: Xxxxxxxx ## Set the certificate ID as Xxxxxxxx. It’s only re
kind: Secret
metadata:
    name: tencent-com-cert
    namespace: default
type: Opaque
```

You can create a Secret in the **TKE console**. For more information, see Creating a Secret.

The main parameters of a Secret are as follows:

Name: A custom name. This document uses `cos-secret` as an example.

Secret type: Select **Opaque**. This type is suitable for saving key certificates and configuration files. The value is Base64-coded.

Validity range: Select a range as required and ensure that the Secret is in the same namespace as the Ingress.

Content: Set the variable name to `qcloud_cert_id` and the variable value to the server certificate ID.

Important

If you want to configure a mutual authentication certificate, you need to add both the server certificate and the client CA certificate to the Secret. Also, you need to add a key pair to the Secret: The variable name is `qcloud_ca_cert_id`, and the variable value is the client CA certificate ID.

Ingress Certificate Configuration

If only one `spec.secretName` is set and no hosts are configured, the certificate will be configured for all HTTPS forwarding rules, as shown in the following example:

```
spec:
    tls:
    - secretName: secret-tls
```

You can configure a level-1 domain name with a wildcard, as shown in the following example:

```
spec:
    tls:
    - hosts:
      - *.abc.com
      secretName: secret-tls
```

If you configure a certificate and a wildcard certificate at the same time, TKE selects a certificate based on priority. As shown in the following example, `www.abc.com` will use the certificate that is described in `secret-tls-2` .

```
spec:
    tls:
    - hosts:
      - *.abc.com
      secretName: secret-tls-1
    - hosts:
      - www.abc.com
      secretName: secret-tls-2
```

When you update an Ingress that has used multiple certificates, the TKE Ingress controller will perform the following judgments:

If no host matches rules.host in HTTPS, the update cannot be submitted.

If one TLS host matches rules.host in HTTPS, the update can be submitted and the certificate corresponding to the Secret can be configured for the host.

When a SecretName of TLS is changed, only the existence of the SecretName but not the Secret content will be verified. In this case, the update can be submitted as long as the Secret exists.

Note

Make sure that the certificate ID in the Secret meets requirements.

Directions

Creating a server certificate in the console

Description

Skip this step if you already have the target certificate.

1. Log in to the CLB console and click Certificate Management in the left sidebar.
2. On the **Certificate management** page, click **Create**.
3. On the **Create a new certificate** page, set the parameters based on the following description:

Certificate name: Set a custom certificate name.

Certificate type: Select **Server certificate**, which indicates an SSL certificate. An encrypted HTTP protocol based on the SSL certificate for secure data transmission enables a site to be switched from HTTP to HTTPS.

Certificate content: Enter the certificate content based on your actual requirements. For more information on certificate format requirements, see SSL Certificate Format.

Key content: It is displayed only when **Certificate type** is **Server certificate**. For more information, see SSL Certificate Format.

4. Click **Submit**.

Creating an Ingress object that uses the certificate

Reminders

When HTTPS service is enabled for an Ingress object created in the console, a Secret resource with the same name will be created to store the certificate ID. Then, this Secret is used and referenced to in the Ingress.

The mappings between domain names and certificates that can be configured in TLS are as follows:

A level-1 domain name with the wildcard can be configured.

If a domain name matches several certificates, a certificate is randomly selected. We recommend that you not use different certificates for the same domain name.

You must configure certificates for all HTTPS domain names. Otherwise, the Ingress object may fail to be created.

Steps

Create an Ingress object with a **Https:443** listener. For details, see Creating an Ingress.

Modifying Certificates

Reminders

To modify a certificate, you need to verify all Ingress objects that use the certificate. If multiple Ingress objects are configured with the same Secret resource, the CLB certificates of these Ingress objects will be modified simultaneously.
You need to modify a certificate by modifying its Secret because the Secret content includes your Tencent Cloud certificate ID.

Steps

1. Run the following command to open the Secret to be modified in the default editor. Note that you need to replace `[secret-name]` with the name of the target secret.

```
kubectl edit secrets [secret-name]
```

2. Modify the Secret resource and change the value of `qcloud_cert_id` to the new certificate ID.

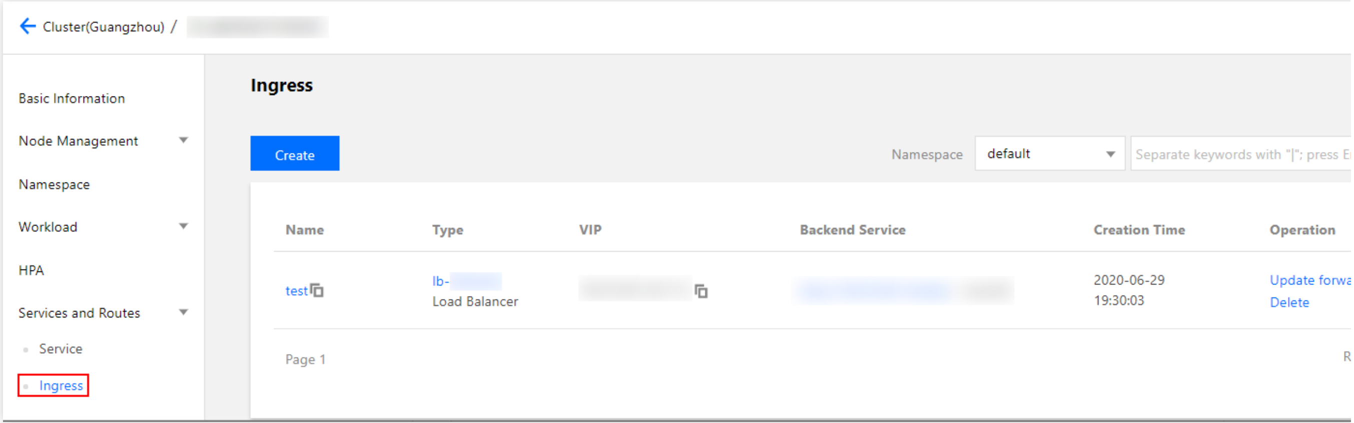
Similar to the creation of a Secret, modifying a Secret certificate ID requires Base64 encoding. Select Base64 manual encoding or specify `stringData` to perform Base64 automatic encoding based on your actual needs.

Updating Ingress objects

Updating an Ingress object in the console

Updating an Ingress object by using YAML

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the target cluster ID.
3. On the cluster details page, select **Service and route** > **Ingress** in the left sidebar.



Name	Type	VIP	Backend Service	Creation Time	Operation
test	lb-[redacted] Load Balancer	[redacted]	[redacted]	2020-06-29 19:30:03	Update forwa Delete

4. Find the target Ingress object, and click **Update forwarding configuration** in the **Operation** column.
5. On the **Update forwarding configuration** page, update the forwarding configuration rules as required.
6. Click **Update forwarding configuration** to complete the update.

Run the following command to open the Ingress object to be modified in the default editor. Modify the YAML file and save the modification.

```
kubectl edit ingress <ingressname> -n <namespaces>
```

Ingress Annotation

Last updated：2022-09-26 16:12:50

You can use the following annotations to configure Ingress to enrich CLB capabilities.

Annotation Usage

```
apiVersion:
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.class: "qcloud"
name: test
........
```

Annotation Collection

kubernetes.io/ingress.class

Note:

This annotation is used to configure the Ingress type for the component management that is not configured with the annotation or the Ingress resource whose annotation content is qcloud.

Use case:

`kubernetes.io/ingress.class: "qcloud"`

kubernetes.io/ingress.qcloud-loadbalance-id

Note:

This is a read-only annotation that provides the LoadBalanceId referenced by the current Ingress.

Use case:

`kubernetes.io/ingress.qcloud-loadbalance-id: "lb-3imskkfe"`

ingress.cloud.tencent.com/loadbalance-nat-ipv6

Note:

This is a read-only annotation that provides IPv6 address when the user configures or applies for NAT IPv6 CLB.

ingress.cloud.tencent.com/loadbalance-ipv6

Note:

This is a read-only annotation that provides IPv6 address when the user configures or applies for FullStack IPv6 CLB.

kubernetes.io/ingress.internetChargeType

Note:

The billing type of CLB can only be configured at the time of creation, and cannot be modified after the creation.
This annotation is used to specify the CLB payment mode when a CLB instance is created. You need to use it with `kubernetes.io/ingress.internetMaxBandwidthOut` annotation.

Valid values:

- TRAFFIC_POSTPAID_BY_HOUR: Postpaid by traffic on an hourly basis.
- BANDWIDTH_POSTPAID_BY_HOUR: Postpaid by bandwidth on an hourly basis.

Use case:

```
kubernetes.io/ingress.internetChargeType: "TRAFFIC_POSTPAID_BY_HOUR"
```

kubernetes.io/ingress.internetMaxBandwidthOut

Note:

CLB bandwidth can only be configured at the time of creation, and cannot be modified after the creation.
This annotation is used to specify the maximum outbound bandwidth of the CLB instance when a CLB instance is created, which applies only to public network CLB instances. You need to use it with `kubernetes.io/ingress.internetChargeType` annotation.

Valid values:

Value range: 1-2,048 Mbps

Use case:

```
kubernetes.io/ingress.internetMaxBandwidthOut: "2048"
```

kubernetes.io/ingress.extensiveParameters

Note:

This annotation uses the parameters configured when the CLB was created. It can only be configured at the time of creation and cannot be modified after the creation.

Refer to Creating a CLB Instance to add custom parameters for the created CLB instance.

Use case:

- Creating a NAT64 IPv6 instance:
  `kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPV6"}'`
- Creating an IPv6 Instance
  `kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPv6FullChain"}'`
- Purchasing a CTCC CLB:
  `kubernetes.io/ingress.extensiveParameters: '{"VipIsp":"CTCC"}'`
- Specifying a availability zone to create
  `kubernetes.io/ingress.extensiveParameters: '{"ZoneId":"ap-guangzhou-1"}'`

kubernetes.io/ingress.subnetId

Note:

This annotation is used to specify the creation of a private network CLB, and specify the subnet where the CLB locates.

Use case:

`kubernetes.io/ingress.subnetId: "subnet-3swgntkk"`

kubernetes.io/ingress.existLbId

Note:

This annotation is used to specify the use of the existing CLB as the entry resource of the access layer.

> Note：
> When using an existing CLB, you need to ensure that it does not include other listeners.

Use case:

`kubernetes.io/ingress.existLbId: "lb-342wppll"`

kubernetes.io/ingress.rule-mix:

kubernetes.io/ingress.http-rules:

kubernetes.io/ingress.https-rules:

Note:

The first annotation is used to configure hybrid protocols. The second and third annotations are used to configure the forwarding rules, which support forwarding via both http and https protocols. These annotations can be used to configure manual redirection.

Use case:

For more information on usage, see Mixed Use of HTTP and HTTPS Protocols through Ingress.

ingress.cloud.tencent.com/direct-access

Note:

You can use this annotation to achieve CLB-to-Pod direct access in layer 7. Pay attention to the service dependency of directly access under different networks.

Use case:

For details, see Using Services with CLB-to-Pod Direct Access Mode.

ingress.cloud.tencent.com/tke-service-config

Note:

This annotation is used to set the CLB related configurations through tke-service-config, including listeners, forwarding rules, etc.

Use case:

`ingress.cloud.tencent.com/tke-service-config: "nginx-config"` . For more information, see Using TkeServiceConfig to Configure CLBs.

ingress.cloud.tencent.com/tke-service-config-auto

Note:

You can use this annotation to automatically create the TkeServiceConfig resource and provide a configuration template for user to configure as needed.

Use case:

`ingress.cloud.tencent.com/tke-service-config-auto: "true"` . For more information, see Using TKEServiceConfig to Configure CLBs.

ingress.cloud.tencent.com/rewrite-support

Note:

- This annotation can be used to configure manual redirection together with `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules` .
- This annotation can be used to configure automatic redirection together with `ingress.cloud.tencent.com/auto-rewrite` .

Use case:

`ingress.cloud.tencent.com/rewrite-support: "true"`

ingress.cloud.tencent.com/auto-rewrite

Note:

This annotation is used to provide automatic redirection for the HTTP port. All forwarding rules declared on the HTTPS port will create corresponding redirection rules. It is used together with the `ingress.cloud.tencent.com/rewrite-support` annotation to enable redirection management capabilities.

Use case:

`ingress.cloud.tencent.com/auto-rewrite: "true"`

ingress.cloud.tencent.com/cross-region-id

Note:

This annotation is used for Ingress cross-region binding, and is used to specify which region to access. It is used together with `kubernetes.io/ingress.existLbId` or `ingress.cloud.tencent.com/cross-vpc-id` .

Use case:

- Creating a CLB for remote access
  `ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"`
  `ingress.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"`

- Selecting an existing CLB for remote access
  `ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"`
  `kubernetes.io/ingress.existLbId: "lb-342wppll"`

ingress.cloud.tencent.com/cross-vpc-id

Note:

This annotation is used for Ingress cross-region binding, and is used to specify which VPC to access. It can be used together with `ingress.cloud.tencent.com/cross-region-id` annotation to specify the VPC of other region.

> Note：
> This annotation applies to the CLB created and managed by TKE. It is invalid for scenarios that use the existing CLB.

Use case:

Creating CLB for remote access:

`ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"`

`ingress.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"`

ingress.cloud.tencent.com/enable-grace-shutdown

Note:

This annotation is used to shut down CLB instances gracefully in direct access mode. If a deleted Pod contains `DeletionTimestamp` and is in **Terminating** status, the weight of the Pod on the CLB backend is adjusted to `0` .

Use case:

It is only supported in direct access mode and needs to be used together with `ingress.cloud.tencent.com/direct-access` . For more information on how to use it, see Graceful Ingress Shutdown.

ingress.cloud.tencent.com/enable-grace-shutdown-tkex

Note:

This annotation is used to shut down CLB instances gracefully in direct access mode. If the endpoints in the Endpoint object are `not-ready` , the weights on the CLB backend are adjusted to `0` .

Use case:

It is only supported in direct access mode and needs to be used together with `ingress.cloud.tencent.com/direct-access` . For detailed directions, see Graceful Ingress Shutdown.

ingress.cloud.tencent.com/security-groups

Note:

This annotation is used to bind security groups to CLB-type Ingresses. Up to five security groups can be bound to a CLB.

Notes:

- For more information, see Use Limits of CLB security groups.
- Usually, the "Allow Traffic by Default" feature must be enabled, with which the traffic forwarding between CLB and CVM is allowed by default. Traffic coming from the CLB only needs to be verified by the security group bound to the CLB. The annotation is `ingress.cloud.tencent.com/pass-to-target` .

Use case:

`ingress.cloud.tencent.com/security-groups: "sg-xxxxxx,sg-xxxxxx"`

ingress.cloud.tencent.com/pass-to-target

Note:

This annotation is used to configure the "Allow Traffic by Default" feature for the CLB-type Ingress. The traffic forwarding between CLB and CVM is allowed by default. Traffic coming from the CLB only needs to be verified by the security group bound to the CLB.

Notes:

- For more information, see Use Limits of CLB security groups.
- Usually, the feature of binding a security group is required. The annotation is `ingress.cloud.tencent.com/security-groups` .

Use case:

`ingress.cloud.tencent.com/pass-to-target: "true"`

Mixed Use of HTTP and HTTPS Protocols through Ingress

Last updated：2022-04-25 15:29:02

Mixed Rules

In default scenarios, if TLS is not configured in Ingress, the service will be exposed though HTTP protocol. If TLS is configured in Ingress, the service will be exposed though HTTPS protocol. The service described by Ingress is only able to be exposed though one type of protocol. To deal with the limitations of this rule, TKE provides support for mixed use of both protocols.

If you need to expose services through HTTP and HTTPS protocols simultaneously, you can refer to this document to enable mixed protocols and configure all forwarding rules to `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules` annotations.

Rule Format

The rule format of `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules` is a `Json Array` . The format for each object is as below:

```
{
"host": "<domain>",
"path": "<path>",
"backend": {
"serviceName": "<service name>",
"servicePort": "<service port>"
}
}
```

Configuration Steps

`TKE Ingress Controller` supports mixed configuration of `HTTP` and `HTTPS` rules. The steps are as follows:

1. **Enable mixed rules**
   Add the `kubernetes.io/ingress.rule-mix` annotation in Ingress and set it to `true` .

2. **Match rules**

   Match each forwarding rule in Ingress with `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules` , and add them to the corresponding rule set. If a corresponding rule is not found in Ingress annotation, it is added to the HTTPS rule set by default.

3. **Verify matches**

   When matching rules, verify Host, Path, ServiceName, and ServicePort (Host defaults to `VIP` , and Path defaults to `/`).

Example

Ingress example: sample-ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
annotations:
kubernetes.io/ingress.http-rules: '[{"host":"www.tencent.com","path":"/","backen
d":{"serviceName":"sample-service","servicePort":"80"}}]'
kubernetes.io/ingress.https-rules: '[{"host":"www.tencent.com","path":"/","backen
d":{"serviceName":"sample-service","servicePort":"80"}}]'
kubernetes.io/ingress.rule-mix: "true"
name: sample-ingress
namespace: default
spec:
rules:
- host: www.tencent.com
http:
paths:
- backend:
serviceName: sample-service
servicePort: 80
path: /
tls:
- secretName: tencent-com-cert
```

This example contains the following configuration:

- It describes the default certificate. The certificate ID should exist in the Secret resource `tencent-com-cert` .
- It enables mixed protocols, and describes the forwarding rule that described in `ingress.spec.rule` in both `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules` .

3. At this point, CLB will configure forwarding rule in both HTTP and HTTPS to expose a service.

API Gateway Type Ingress

API Gateway TKE Tunnel Configuration

Last updated：2023-03-31 10:34:01

Scenario

You can directly access Pods in a TKE cluster through API Gateway without passing through CLB. This document describes how to create a TKE tunnel and configure it as the backend type of an API in the console, so that requests from API Gateway go directly to the corresponding Pod of the TKE tunnel.

Feature strengths

API Gateway is directly connected to the Pods of the TKE cluster, reducing intermediate nodes (such as CLB).

A TKE tunnel can connect multiple TKE clusters at the same time.

Note

TKE tunnels are currently only supported by **dedicated** API Gateway instances.

Prerequisites

1. You have a dedicated instance.
2. You have a TKE cluster and have obtained its admin role.

Directions

Step 1. Create a TKE tunnel

1. Log in to the API Gateway console.
2. Select **Backend Tunnel** on the left sidebar and click **Create**.
3. On the **Create Backend Tunnel** page, enter the following information:

Backend Tunnel Name: Enter a custom name.

Tunnel Type: Select **TKE tunnel**.

VPC: Select a VPC.

Service List: Up to 20 services can be configured in the service list. The weighted round robin algorithm is used to distribute traffic among multiple Pods. The steps to configure a service are as follows:

3.1.1 Enter the weight ratio of each Pod of the service.

3.1.2 Select the cluster. If the cluster has not been authorized, API Gateway will request authorization.

3.1.3 Select a namespace in the cluster.

3.1.4 Select the service and its port.

3.1.5 Advanced options: Select additional node labels.

Backend Type: Select **HTTP** or **HTTPS**.

Host Header: Optional. It is the value of host in the request header carried in the HTTP/HTTPS request when API Gateway accesses the backend service.

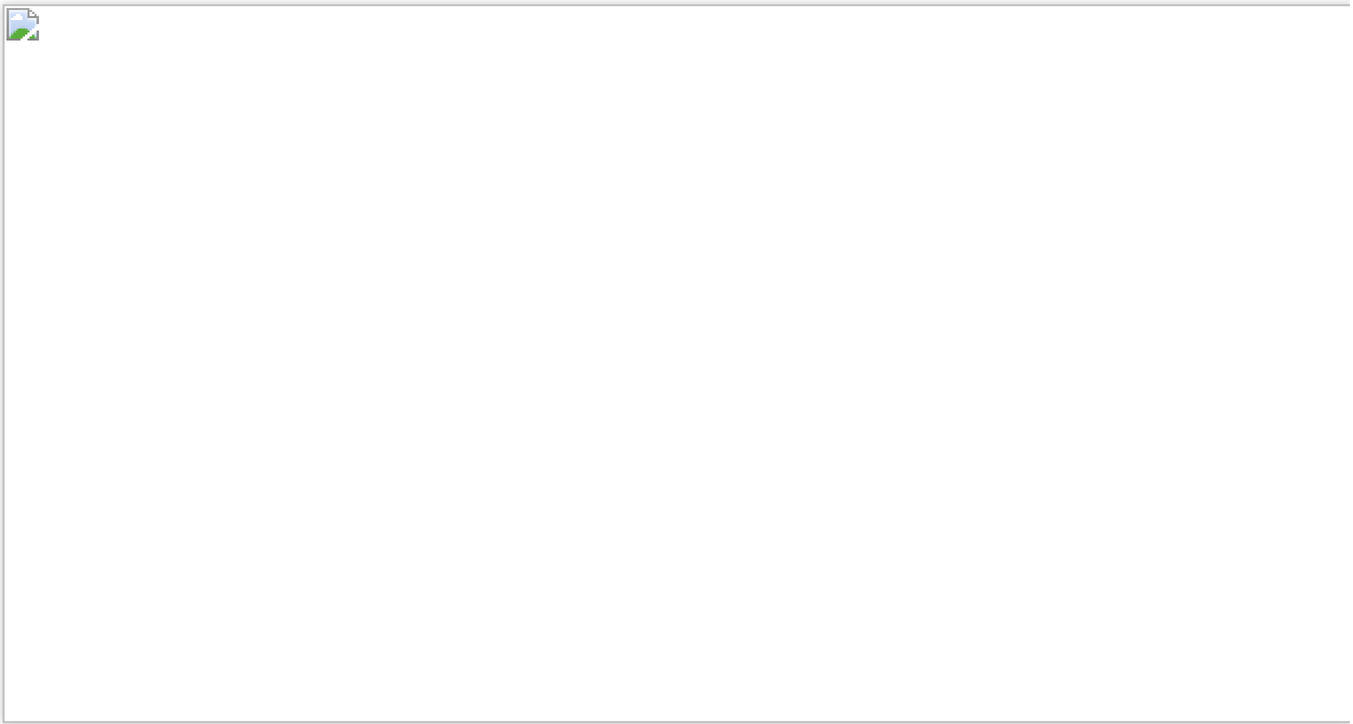
Tag: Optional. A tag is used to manage resources by category from different dimensions.

Step 2. Connect the API backend to the TKE tunnel

1. On the Service page in the API Gateway console, click the target service ID to enter the API management page.
2. Click **Create** to create a general API.
3. Enter the frontend configuration information and click **Next**.
4. Select **VPC resources** as the backend type, select **TKE tunnel** as the backend tunnel type, and click **Next**.
5. Set the response result and click **Complete**.

Network Architecture

After the TKE tunnel is bound to the API, the architecture of the entire network is as follows:



API Gateway directly accesses the Pods in the TKE cluster without passing through CLB. The YAML configuration file of the cluster's httpbin service is as follows, where the `selector` indicates that the Pod with the tag key `app` and tag value `httpbin` is selected as the node of the TKE tunnel. Therefore, Pods on versions 1/2/3 are also nodes of the TKE tunnel.

```
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
    service: httpbin
spec:
  ports:
  - name: http
    port: 8000
```

```
      targetPort: 80
  selector:
    app: httpbin
```

Reminders

A TKE tunnel can connect up to 20 TKE services.

You should have the admin role of the TKE cluster.

The TKE tunnel and the dedicated API Gateway instance must be in the same region. Currently, API Gateway doesn't support cross-VPC access.

Granting TKE Cluster Permissions to API Gateway

Last updated：2023-03-31 10:34:01

Scenario

This document describes how to authorize API Gateway to access the API server of a TKE cluster, offers solutions to authorization issues, and lists the permissions obtained by API Gateway in an YAML file.

Prerequisites

1. You have logged in to the API Gateway console.
2. You have a TKE cluster and have obtained its admin role.

Directions

In the TKE tunnel configuration of API Gateway, if you reference a TKE cluster for the first time, you need to grant API Gateway the access to the cluster's API server and ensure that the cluster has private network access enabled.
When the TKE tunnel is configured, the API Gateway system will automatically check whether the cluster has been authorized, and if not, it will prompt you for authorization.
If the cluster access has already been granted to API Gateway, the system will display **Authorized API Gateway**. Each cluster only needs to be authorized for API Gateway once and doesn't require repeated authorizations for subsequent operations.

How It Works

The process for API Gateway to get the authorization is as follows:
1. Under the `kube-system` namespace, create a ServiceAccount named `apigw-ingress` and a ClusterRole named `apigw-ingress-clusterrole` .
2. Bind `apigw-ingress` and `apigw-ingress-clusterrole` through ClusterRoleBinding. Then, the permission of the `apigw-ingress` ServiceAccount is obtained by API Gateway to access the API server of the cluster.

The permission of the `apigw-ingress` ServiceAccount is stored in the Secret prefixed with `apigw-ingress-token-`.

For more information on the permissions obtained by API Gateway and the specific method, see the YAML used to create resources:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: apigw-ingress-clusterrole
rules:
  - apiGroups:
      - ""
    resources:
      - services
      - namespaces
      - endpoints
      - nodes
      - pods
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - apps
    resources:
      - deployments
      - replicasets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - configmaps
      - secrets
    verbs:
      - "*"
  - apiGroups:
      - extensions
    resources:
      - ingresses
      - ingresses/status
    verbs:
      - "*"
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - create
      - patch
      - list
      - update
  - apiGroups:
      - apiextensions.k8s.io
    resources:
      - customresourcedefinitions
    verbs:
      - "*"
  - apiGroups:
      - cloud.tencent.com
    resources:
      - tkeserviceconfigs
    verbs:
      - "*"
---
apiVersion: v1
kind: ServiceAccount
metadata:
  namespace: kube-system
  name: apigw-ingress
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: apigw-ingress-clusterrole-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: apigw-ingress-clusterrole
subjects:
  - kind: ServiceAccount
    name: apigw-ingress
    namespace: kube-system
```

Reminders

After you successfully grant API Gateway the access to the TKE cluster, you cannot modify the resources reserved by API Gateway, including:

The ServiceAccount named `apigw-ingress` under the `kube-system` namespace.

The ClusterRole named `apigw-ingress-clusterrole` under the `kube-system` namespace.

The ClusterRoleBinding named `apigw-ingress-clusterrole-binding` under the `kube-system` namespace.

The Secret prefixed with `apigw-ingress-token-` in the `kube-system` namespace.

FAQs

Problem: During authorization, it is found that the private network access feature is not enabled for the TKE cluster.

Solution: Enable the TKE cluster's private network access feature and then click **Retry**.

Use of additional node labels

Last updated：2023-03-31 10:34:01

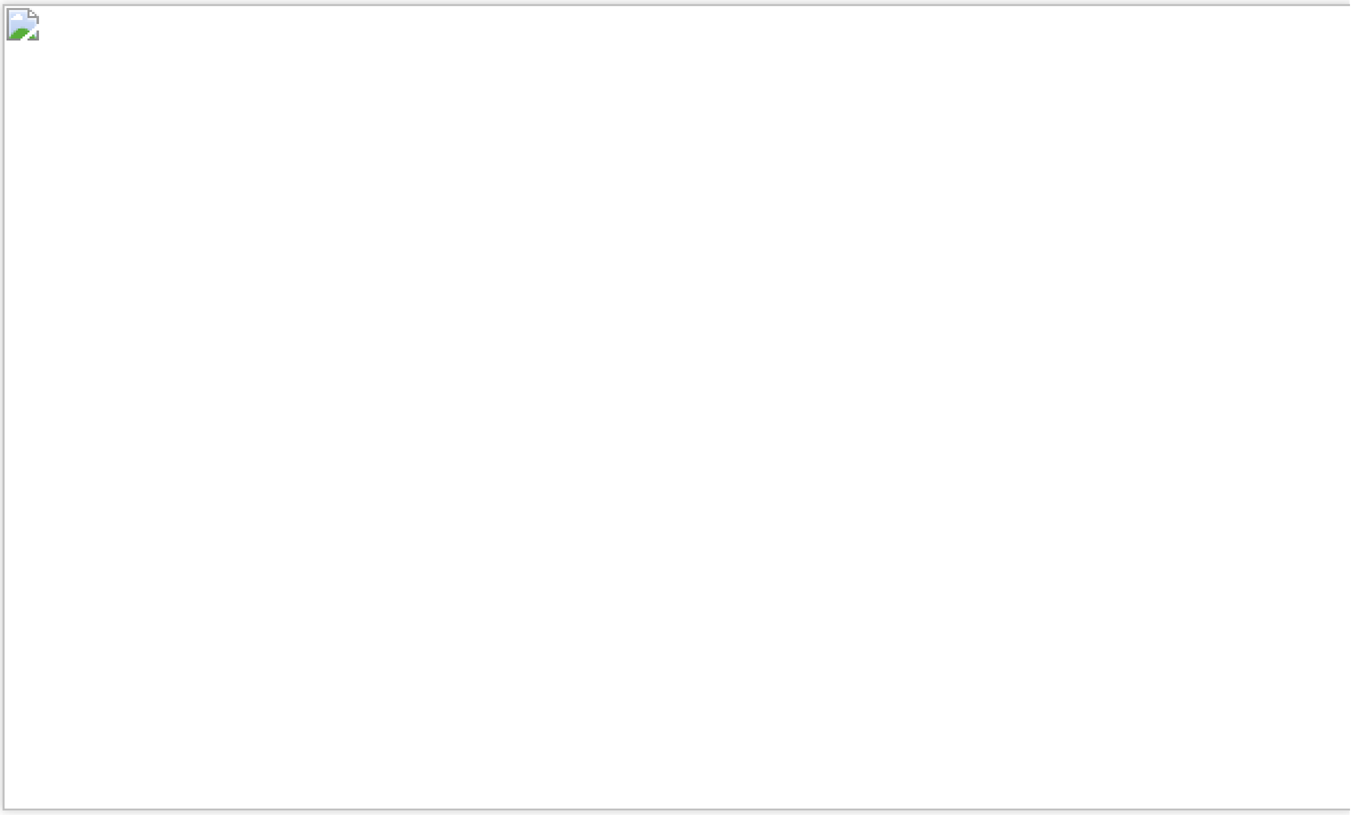
Use Cases

By using extra node label, you can directly forward a request to a Pod with the specified label under a service so as to finely control the target Pod.
For example, in the `default` namespace, there are a Pod with labels of `app: httpbin` and `version: v1`, a Pod with labels of `app: httpbin` and `version: v2`, as well as an httpbin service with selector of `app: httpbin`. If you want API Gateway to only forward requests to the Pod whose labels are `app: httpbin` and `version: v1`, you can use the extra node label and add the configuration of `version: v1` to achieve this.

Directions

1. When configuring the TKE tunnel, manually enter the extra node label.
2. Click **Save** to create or modify the TKE tunnel.

The final effect of forwarding is as follows:



How It Works

In a TKE cluster, the service itself is configured with selectors. For example, in the httpbin service, the selector is `app: httpbin` , but the extra node label provided by API Gateway will be combined with the selector in the httpbin service, so the final label combination will be `app: httpbin` and `version: v1` . Therefore, only the http Pod with the `version: v1` label will appear as this TKE tunnel's node.

If you enter a label key that already exists in the httpbin service in the extra node label, the extra node label will be ignored, and the value of the label existing in the selector will prevail. For example, if you enter `app: not-httpbin` in the extra label, it will conflict with the selector of the httpbin service, so `app: not-httpbin` will be ignored.

The YAML of the httpbin service is as follows:

```
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
    service: httpbin
spec:
  ports:
  - name: http
    port: 8000
    targetPort: 80
  selector:
    app: httpbin
```

Reminders

Extra node label is an advanced feature and requires you to make sure that the label exists during value input. If a wrong label is entered, the number of nodes in the TKE tunnel will become 0.

If the selector of the service and the extra node label have the same key, the configuration in the selector will prevail.

If the service port is modified (for example, from 80 to 8080), the change needs to be synced to API Gateway. If the service port is not modified while only the target port is modified, the change will be automatically synced to API Gateway.

Nginx Type Ingress

Overview

Last updated：2022-07-26 16:03:16

Nginx-ingress Introduction

Nginx can be used as a reverse proxy, load balancer, and for HTTP caching.

Nginx-ingress is an Ingress controller for Kubernetes that uses Nginx as a reverse proxy and load balancer. You can deploy and use Nginx-ingress add-on in the cluster. TKE provides productization capabilities to help you install and use Nginx-ingress in the cluster.

Nginx-ingress Concepts

- **Nginx-ingress add-on**: You can use Nginx-ingress in TKE through Nginx-ingress add-on. Install and deploy Nginx-ingress add-on with one click on the **Add-on management** page.
- **Nginx-ingress instance**: You can deploy multiple Nginx-ingress instances in a cluster (for example, one for the public network and one for the private network). Each instance corresponds to a CRD resource in Kubernetes. When a Nginx-ingress instance is created, Nginx-ingress-controller, service, configmap and other Kubernetes resources will be automatically created in the cluster.
- **Nginx-ingress-controller**: The actual Nginx load. The controller will watch the kubernetes ingress object and update the changes in the cluster. The forwarding configuration of Nginx load is the `nginx.conf` file.

Nginx-ingress Relevant Operations

For details of Nginx-ingress relevant operations and features, see the documents below:

- Installing Nginx-ingress
- Using Nginx-ingress Object to Access External Traffic of the Cluster
- Nginx-ingress Monitoring Configuration
- Nginx-ingress Log Configuration

Installing Nginx-ingress Instance

Last updated：2023-05-23 15:52:02

Installing the Nginx-ingress Add-on

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on management** to go to the **Add-on list** page.
4. On the **Add-On List** page, select **Create**. On the **Create add-on** page, select NginxIngress.
5. Click **Done**. You can view the **Addon Details** under **Services and Routes** > **NginxIngress**.

Reminders

The architecture of Tencent Cloud Load Balancer (CLB) has been upgraded on March 6, 2023. After the upgrade, public network CLB instances deliver services through domain names. As service traffic increases, the VIP changes dynamically. Therefore, the VIP of a CLB instance is no longer displayed in the console. For more information, see Launch of Domain Name-Based Public CLB Instances.

For new Tencent Cloud users, the upgraded domain name-based CLB instances are used by default.

Existing users can choose to continue to use the original CLB instances, which are not affected by the upgrade. If you need to upgrade the CLB service, you need to upgrade both CLB and TKE. Otherwise, the synchronization of all public network Service/Ingress add-ons in TKE may be affected. For information about upgrading CLB, see Upgrading to Domain Name-based CLB. For information about upgrading TKE Service/Ingress add-ons, submit a ticket.

Installation Methods

You can use the following installation methods to install Nginx-ingress in TKE based on the requirements of your business scenarios:

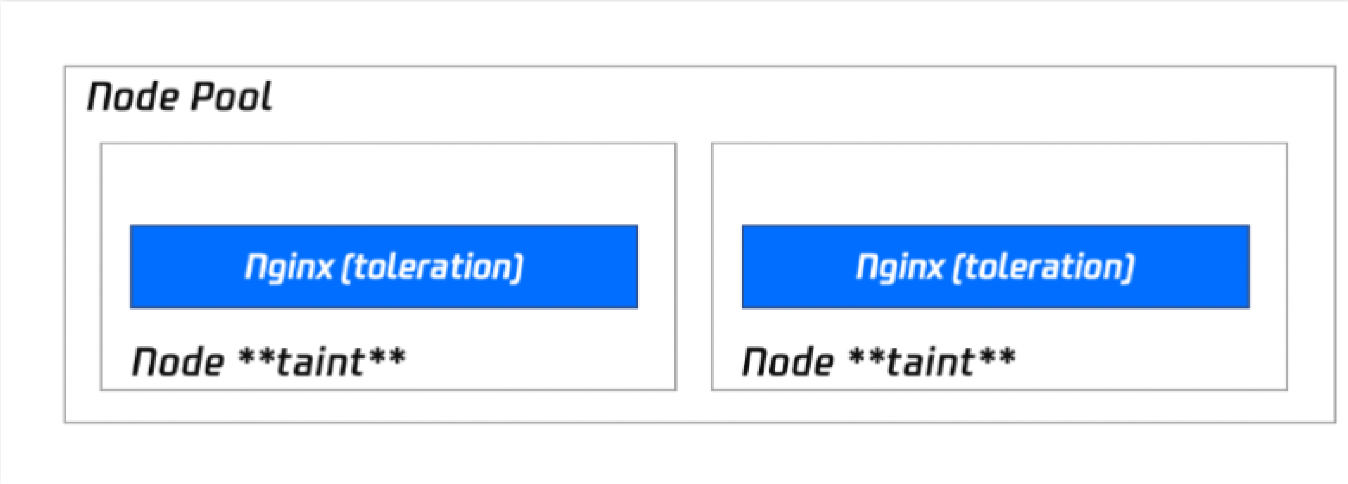
Deploying via specifying a node pool as a DaemonSet

Deploying via "Deployment + HPA" and specifying a scheduling policy

Deploying via Nginx frontend accessing an LB

Deploying via specifying a node pool as a DaemonSet (recommended)

As Nginx is a key traffic Ingress gateway, we recommend you deploy Nginx-ingress in the specified node pool rather than on the same node with other businesses. The deployment architecture is as shown below:

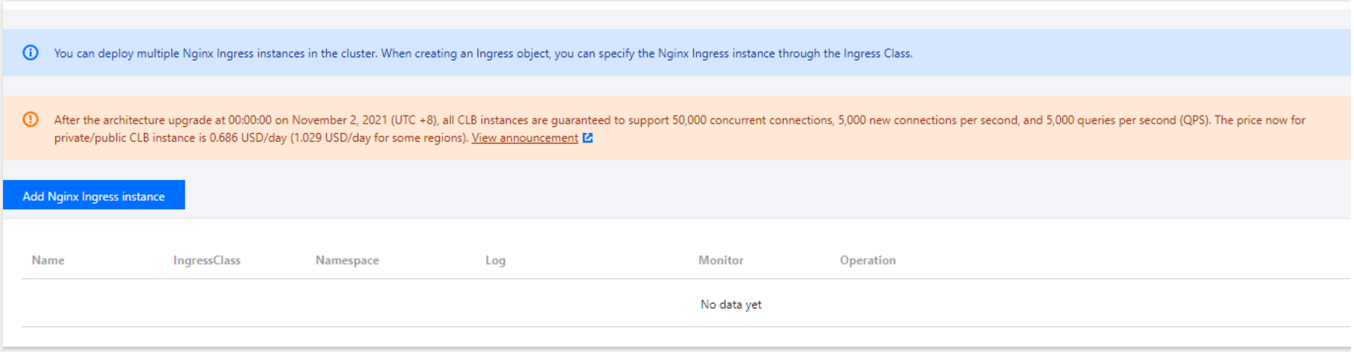


Installation procedure

Note

If you use this installation method, you can enjoy the complete scaling capabilities of the node pool and can remove and add Nginx replicas simply by adjusting the number of nodes in the node pool subsequently.

1. Prepare the node pool for deploying Nginx-ingress, and set the taint to prevent other Pods from scheduling this node pool. For information about how to deploy node pool, see Node Pool Overview.
2. Install the Nginx-ingress add-on in the cluster.
3. On the cluster information page, choose **Services and Routes** > **NginxIngress** and click **Add Nginx Ingress instance**. A cluster can have multiple Nginx instances.

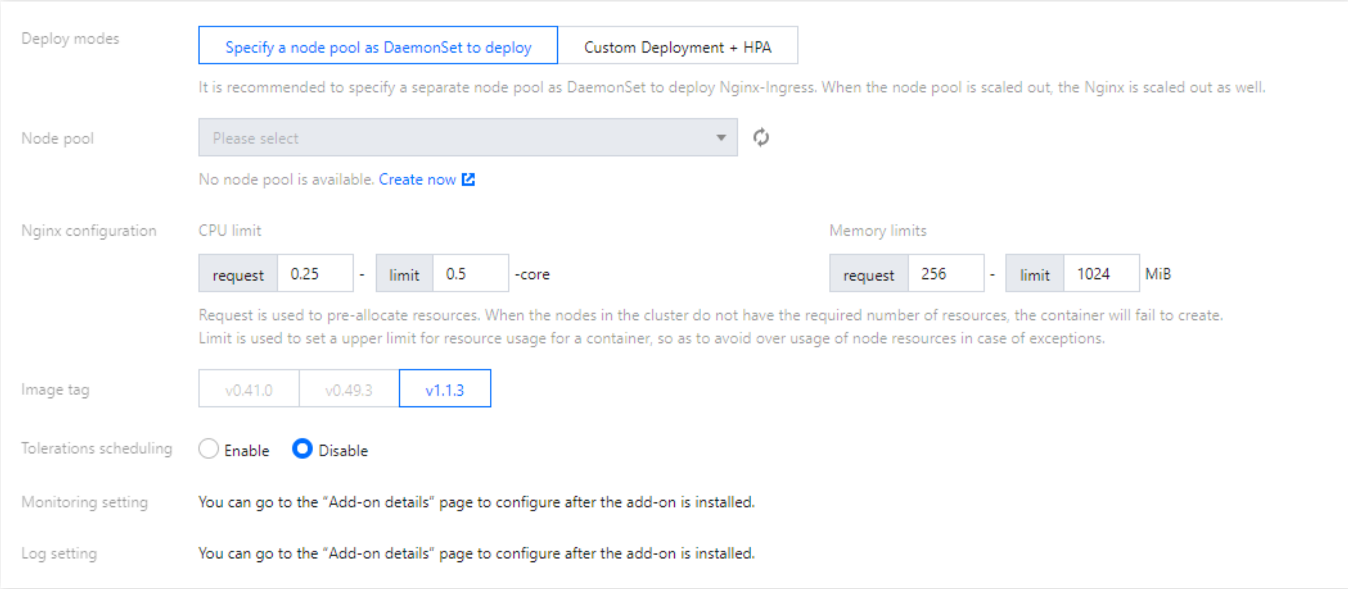


You can deploy multiple Nginx Ingress instances in the cluster. When creating an Ingress object, you can specify the Nginx Ingress instance through the Ingress Class.

After the architecture upgrade at 00:00:00 on November 2, 2021 (UTC +8), all CLB instances are guaranteed to support 50,000 concurrent connections, 5,000 new connections per second, and 5,000 queries per second (QPS). The price now for private/public CLB instance is 0.686 USD/day (1.029 USD/day for some regions). View announcement

Name	IngressClass	Namespace	Log	Monitor	Operation
				No data yet	

4. In the **Create NginxIngress** pop-up window, select **Specify a node pool as DaemonSet to deploy** for **Deploy modes** and set other parameters as needed.



Deploy modes: Specify a node pool as DaemonSet to deploy | Custom Deployment + HPA

It is recommended to specify a separate node pool as DaemonSet to deploy Nginx-Ingress. When the node pool is scaled out, the Nginx is scaled out as well.

Node pool: Please select

No node pool is available. Create now

Nginx configuration: CPU limit: request 0.25 - limit 0.5 -core; Memory limits: request 256 - limit 1024 MiB

Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail to create.
Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node resources in case of exceptions.

Image tag: v0.41.0 | v0.49.3 | v1.1.3

Tolerations scheduling: ○ Enable ● Disable

Monitoring setting: You can go to the "Add-on details" page to configure after the add-on is installed.

Log setting: You can go to the "Add-on details" page to configure after the add-on is installed.

Node pool: Configure the node pool.

Nginx configuration: You need to set **request** to a value lower than the model configuration of the node pool (as the nodes have reserved resources). **limit** is optional.

Image tag:

Kubernetes Versions	Versions Supported by Nginx-ingress	Image Versions Supported by Nginx Instances
<=1.18	1.1.0 and 1.2.0	v0.41.0 and v0.49.3
	1.0.0	v0.41.0
1.20	1.1.0 and 1.2.0	v1.1.3
	1.0.0	v0.41.0
>=1.22	1.1.0 and 1.2.0	v1.1.3

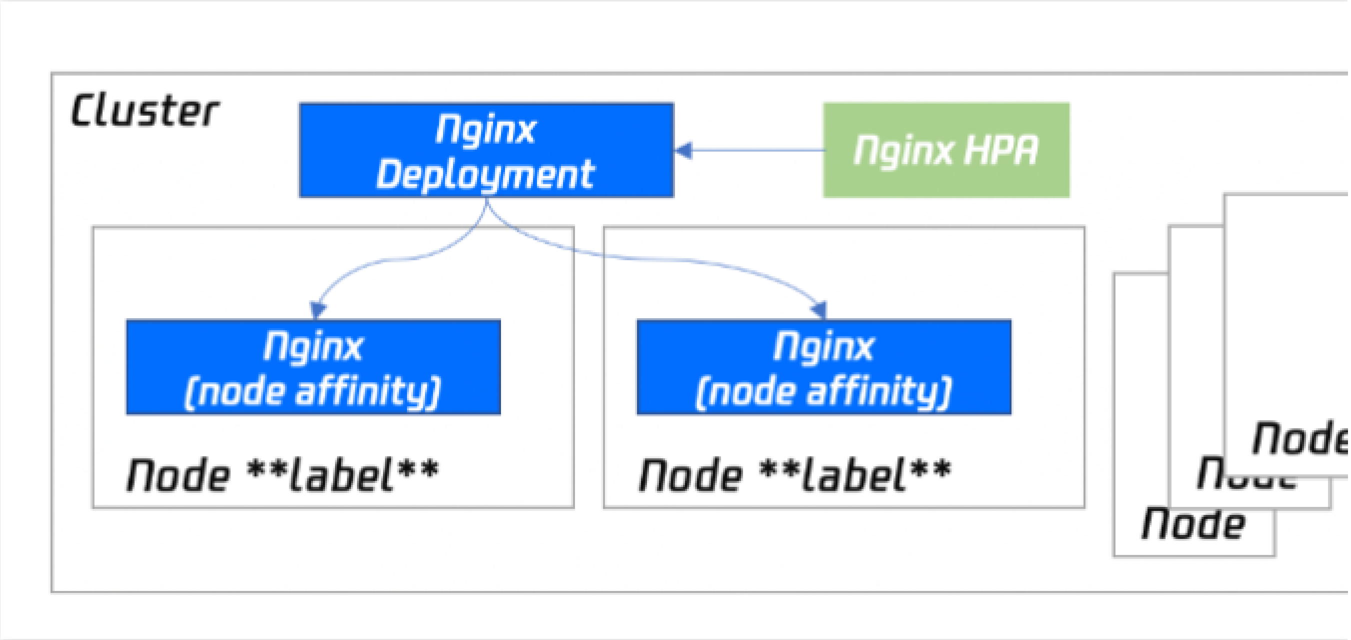
Note

1. **Note on Elastic IP (EIP) usage**: The 1.0.0 and 1.1.0 versions of the Nginx-ingress add-on rely on EIP and the 1.2.0 and later versions no longer require EIP. If your EIP service is subject to use limits, we recommend that you upgrade your Nginx-ingress add-on. The upgrade does not affect existing Nginx Ingress instances, service access, or data security.
2. **Note on upgrades**: For more information about Nginx instance versions, visit GitHub. For more information about how to upgrade a cluster, see Upgrading a Cluster. For more information about how to upgrade the Nginx-ingress add-on, see Add-On Lifecycle Management.

5. Click **OK**.

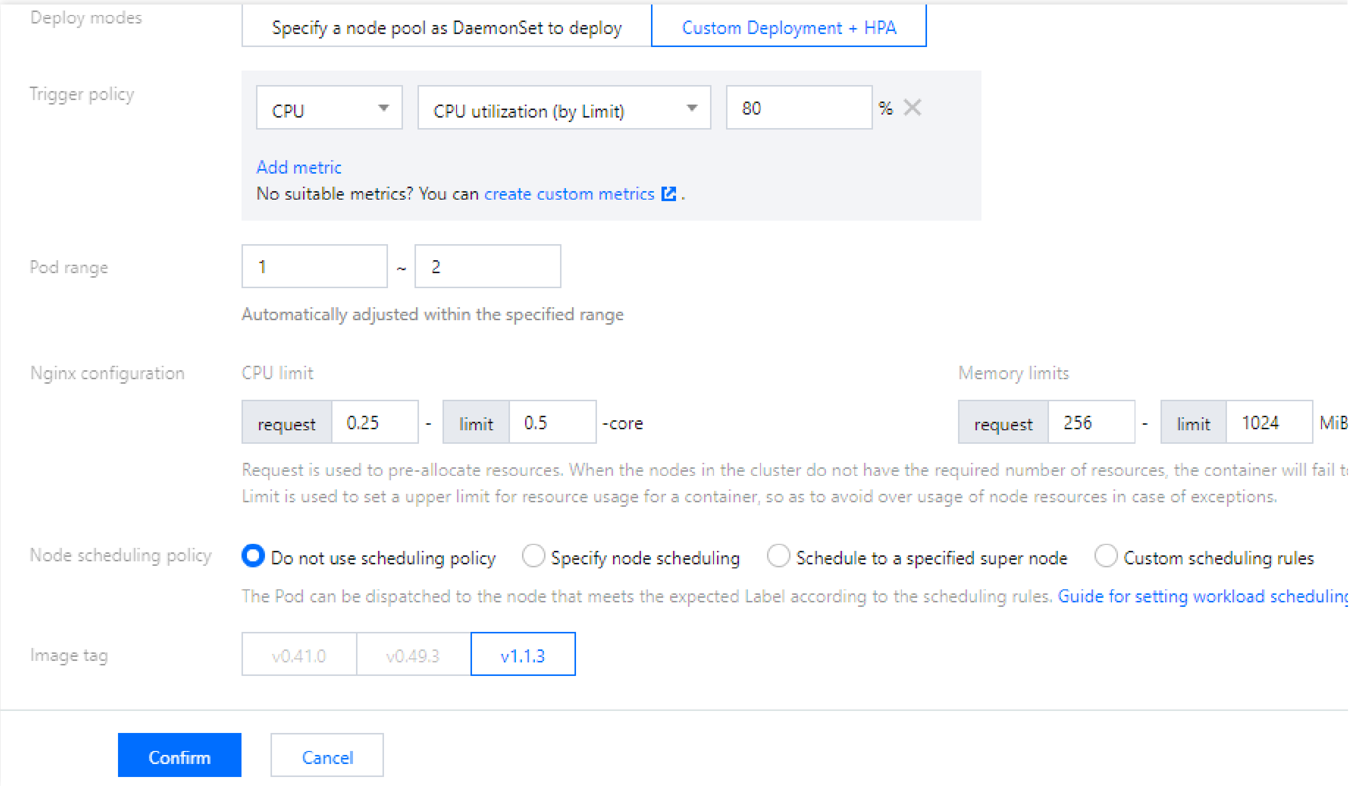
Deploying via "Deployment + HPA" and specifying a scheduling policy

If you use the Deployment + HPA mode to deploy Nginx-ingress, you can configure a taint and tolerance to deploy Nginx instances and business Pods in a distributed manner based on your business needs. With HPA, you can configure auto scaling for Nginx instances based on metrics such as CPU and memory utilization. The deployment architecture is as shown below:



Installation procedure

1. Prepare the node pool for deploying Nginx-ingress, and set the taint to prevent other Pods from scheduling this node pool. For information about how to deploy node pool, see Node Pool Overview.
2. Install the Nginx-ingress add-on in the cluster.
3. On the cluster information page, choose **Services and Routes** > **NginxIngress** and click **Add Nginx Ingress Instance**. A cluster can have multiple Nginx instances.
4. In the **Create NginxIngress** pop-up window, select **Custom Deployment + HPA** for **Deploy modes** and set other parameters as needed.



Deploy modes: Specify a node pool as DaemonSet to deploy | Custom Deployment + HPA

Trigger policy: CPU | CPU utilization (by Limit) | 80 %

Add metric

No suitable metrics? You can create custom metrics .

Pod range: 1 ~ 2

Automatically adjusted within the specified range

Nginx configuration: CPU limit: request 0.25 - limit 0.5 -core; Memory limits: request 256 - limit 1024 MiB

Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail t

Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node resources in case of exceptions.

Node scheduling policy: ● Do not use scheduling policy ○ Specify node scheduling ○ Schedule to a specified super node ○ Custom scheduling rules

The Pod can be dispatched to the node that meets the expected Label according to the scheduling rules. Guide for setting workload scheduling

Image tag: v0.41.0 | v0.49.3 | v1.1.3

Nginx configuration: You need to set **request** to a value lower than the model configuration of the node pool (as the nodes have reserved resources). **limit** is optional.

Node scheduling policy: Specify a policy as needed.

Image tag:

For Kubernetes clusters on v1.20 or earlier, the Nginx-ingress add-on is on v1.0.0, and the Nginx instance image can only be on v41.0.

For Kubernetes clusters on v1.20 or earlier, the Nginx-ingress add-on is on v1.1.0, and the Nginx instance image can only be on v41.0 or v49.3.

For Kubernetes clusters on v1.22 or later, the Nginx-ingress add-on can only be on v1.1.0, and the Nginx instance image can only be on v1.1.3.

Note

For more information about Nginx instance versions, visit GitHub. For more information about how to upgrade a cluster, see Upgrading a Cluster. For more information about how to upgrade the Nginx-ingress add-on, see Add-On Lifecycle Management.

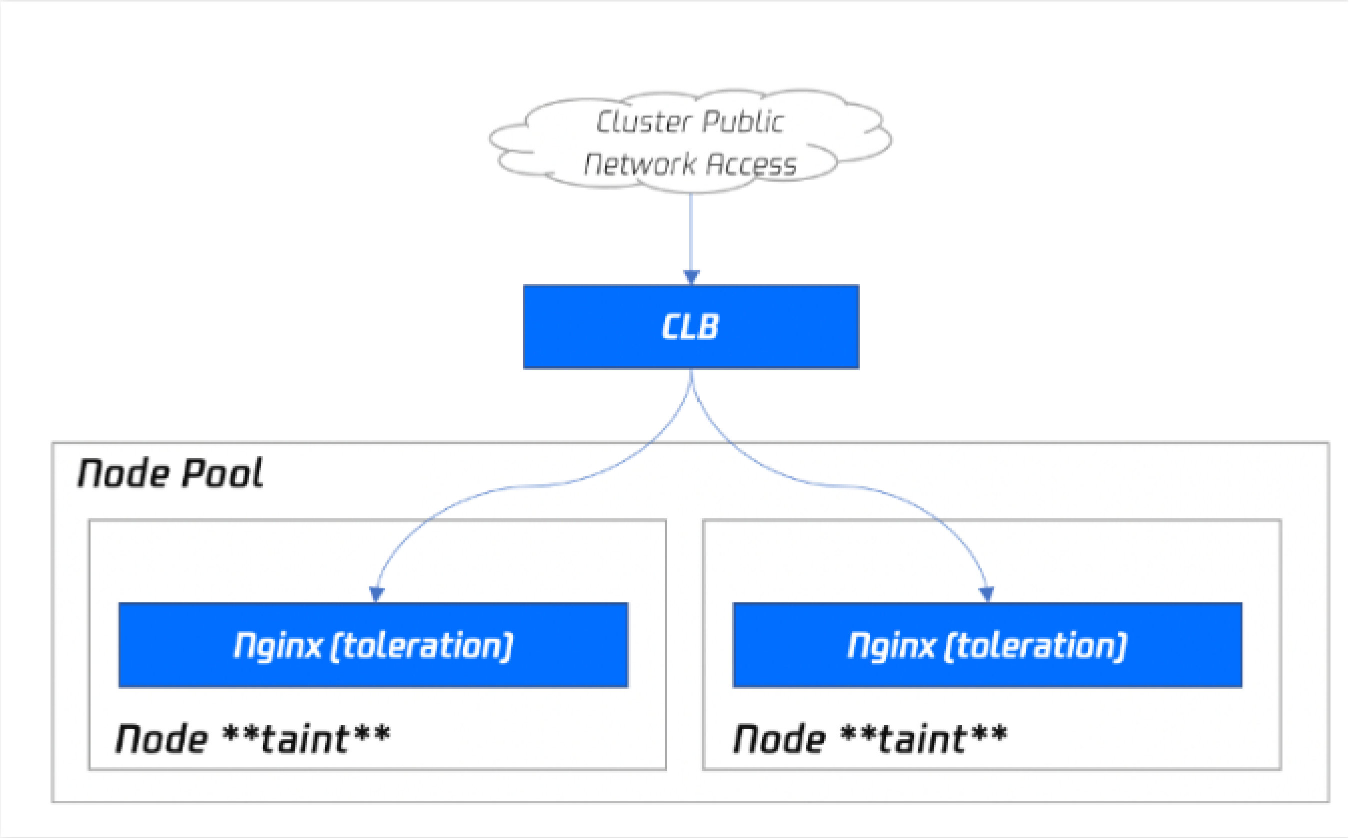
5. Click **OK**.

Deploying via Nginx frontend accessing an LB

If only Nginx is deployed in the cluster, external traffic cannot be received, so you also need to configure the Nginx frontend load balancer. TKE currently provides a product-like installation capability, and you can also select different deployment modes based on your business needs.

Directly connecting cluster in VPC-CNI mode to Nginx Service (recommended)

If your cluster is in VPC-CNI mode, we recommend you directly connect to the Nginx Service through CLB. The load of node pool deployment is used as an example in the following figure:



This solution, with high performance and without manual maintenance of CLB, is the optimal solution. It requires the cluster to enable VPC-CNI. This solution is recommended for the cluster that has configured the VPC-CNI network plug-in, or the Global Router network plug-in with VPC-CNI enabled (both modes are enabled).

Using common Service in LoadBalancer mode in cluster in Global Router mode

If your cluster does not support the VPC-CNI mode, you can also use a common Service in LoadBalancer mode for traffic access. Currently, Services in LoadBalancer mode in TKE are implemented based on NodePorts by default. CLB is bound to the NodePort of a node to use the NodePort as a real server and forwards the traffic to the NodePort, and then the request is routed to the backend Pod of the Service through iptables or IPVS on the node. This scheme is the simplest, but the traffic passes through the NodePort, which means that there is one more layer for forwarding, and the following problems may exist:

The forwarding path is relatively long. After reaching the NodePort, traffic goes through the CLB within Kubernetes and is then forwarded through iptables or ipvs to Nginx. This increases network time consumption.

Passing through the NodePort will necessarily cause SNAT. If traffic is too concentrated, port exhaustion or conntrack insertion conflicts can easily occur, leading to packet loss and causing some traffic exceptions.

The NodePort of each node also serves as a CLB. If the CLB is bound to the NodePorts of large numbers of nodes, the CLB status is distributed among each node, which can easily cause global load imbalance.

The CLB carries out health probes on NodePorts, and probe packets are ultimately forwarded to the Pods of Nginx-ingress. If the CLB is bound to too many nodes, and the number of Pods of Nginx-ingress is small, the probe packets will cause immense pressure on Nginx-ingress.

Using HostNetwork + load balancer mode

The console does not support setting this mode currently. You can manually modify the YAML file of the Nginx workload to set the network mode to HostNetwork and manually create a CLB instance to bind the node port exposed by Nginx.
Note that when you use HostNetwork, to avoid port listening conflicts, Nginx-ingress Pods cannot be scheduled to the same node.

Default Parameters for Nginx-ingress Installation in TKE

Setting Nginx-ingress parameters

In the details page of Nginx-ingress add-on, you can select an Nginx-ingress instance to edit YAML in **Nginx Configuration** tab.
Note
By default, Nginx won't restart after parameter configuration, and it will take a short while for the parameters to take effect.
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on management** to go to the **Add-on list** page.
4. Click **Update Nginx Configuration** on the right of the target add-on to enter the **Nginx Configuration** page.
5. Select the target Nginx-ingress instance and click **Edit YAML**.
6. On the **Update ConfigMap** page, edit the YAML file and click **Done**.

Parameter configuration sample code

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: alpha-ingress-nginx-controller
  namespace: kube-system
data:
  access-log-path: /var/log/nginx/nginx_access.log
  error-log-path: /var/log/nginx/nginx_error.log
  log-format-upstream: $remote_addr - $remote_user [$time_iso8601] $msec "$request"
  keep-alive-requests: "10000"
  max-worker-connections: "65536"
```

```
    upstream-keepalive-connections: "200"
```

Note

Do not modify `access-log-path` , `error-log-path` , or `log-format-upstream` . Otherwise, CLS log collection will be affected.

If you need to configure different parameters for your business, refer to Official Document.

Using Nginx-ingress Object to Access External Traffic of the Cluster

Last updated：2023-02-02 17:05:22

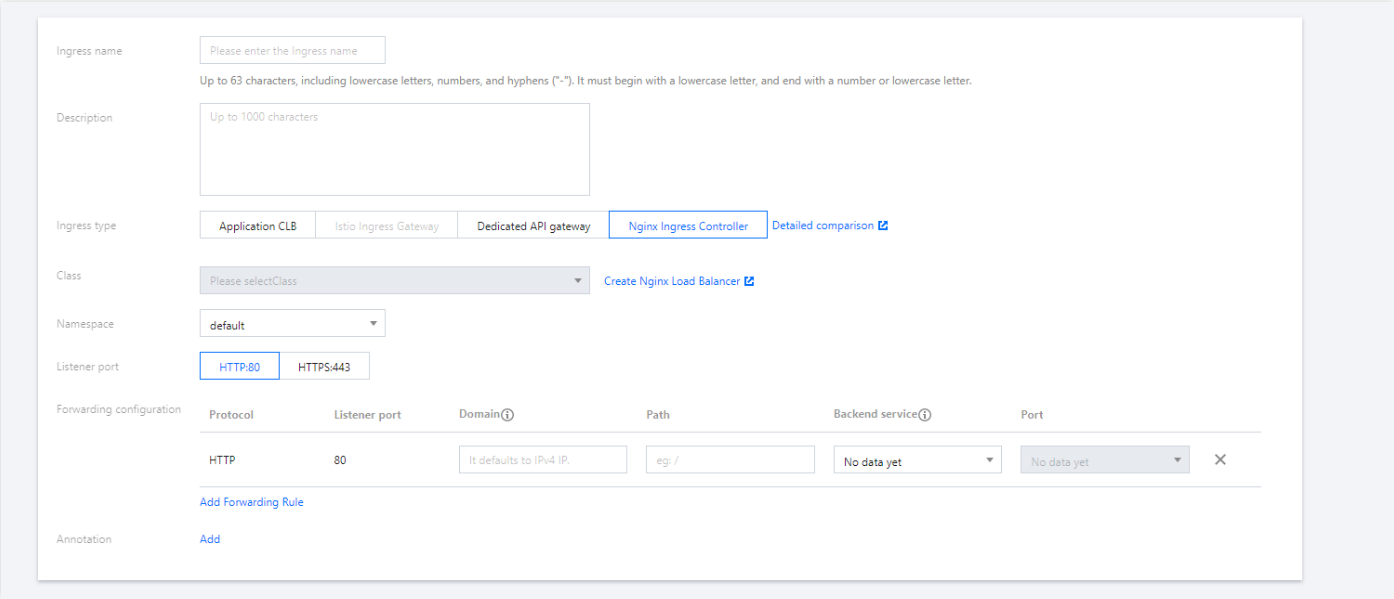
Prerequisites

- You have logged in to the TKE console.
- You have deployed Nginx-ingress Addon in the cluster.
- You have installed and created the Nginx-ingress instance required for the business.

How to Use

Nginx-ingress usage on console

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. Click the cluster ID that has installed the Nginx-ingress addon to go to the cluster details page.
3. Select **Services and Route** > **Ingress** to go to the Ingress information page.
4. Click **Create** to go to the **Create an Ingress** page.
5. Set the Ingress parameters based on actual needs, as shown in the figure below:



Ingress name: Please enter the Ingress name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Description: Up to 1000 characters

Ingress type: Application CLB | Istio Ingress Gateway | Dedicated API gateway | Nginx Ingress Controller Detailed comparison

Class: Please selectClass Create Nginx Load Balancer

Namespace: default

Listener port: HTTP:80 | HTTPS:443

Forwarding configuration:

Protocol	Listener port	Domain	Path	Backend service	Port
HTTP	80	It defaults to IPv4 IP.	eg: /	No data yet	No data yet

Add Forwarding Rule

Annotation: Add

- Ingress type: select **Nginx Ingress Controller**.

- Forwarding configuration: configure forwarding rules as needed.

6. Click **Create Ingress**.

Managing Nginx-ingress using Kubectl

Before importing the IngressClass resource and the ingressClassName field in Kubernetes, the Ingress class was specified by the `kubernetes.io/ingress.class` annotation in Ingress.
Sample:

```
metadata:
name:
annotations:
kubernetes.io/ingress.class: "nginx-pulic". ## The corresponding Nginx-ingress in
stance name of the Nginx-ingress addon in the TKE cluster.
```

Relevant Operations

You can configure annotations for Nginx Ingress objects. For details, see Official Document.

Usage model of Nginx-ingress object

When multiple Ingress objects apply to one Nginx entity:

- Sort the Ingress rules by the CreationTimestamp field, that is, the previous rules shall prevail.
- If the same path is defined for the same host in multiple Ingresses, the most previous rules shall prevail.
- If multiple Ingresses contain the TLS part of the same host, the most previous rules shall prevail.
- If multiple Ingresses define an annotation that affects the configuration of the Server block, the most previous rules shall prevail.
- Create NGINX Server based on each hostname.
- If multiple Ingresses define different paths for the same host, the ingress-controller merges these definitions.
- Multiple Ingresses can define different annotations. These definitions are not shared among Ingresses.
- Ingress annotations will be applied to all paths in Ingress.

Triggering nginx.conf update mechanism

The following describes the situation where nginx.conf needs to be reloaded:

- Create an Ingress object.
- Add a new TLS for Ingress.

- The modification of Ingress annotation not only affects the upstream configuration, but also has a greater impact. For example, the load-balance annotation does not need to be reloaded.
- Add/delete paths for Ingress.
- Delete Ingress and the Service and Secret of Ingress.
- The status of the object associated with the Ingress is unknown, such as Service or Secret.
- Update a Secret.

Nginx-ingress Log Configuration

Last updated：2023-08-10 11:05:41

Integrating with CLS, TKE provides a complete set of productized capabilities to implement Nginx-ingress log collection and consumption capabilities.

Nginx-ingress Log Types

Nginx Controller collects and provides the following logs to users:

- **Nginx Controller Log**: major. The control plane logs, which record the modification of the Nginx Controller control plane. It is mainly used for control plane troubleshooting, for example, due to the incorrect configuration of the Ingress template, the synchronization is not performed.
- **AccessLog Log**: major. User data plane logs, which record the relevant information of user's layer-7 request. It is mainly used for users to perform data analysis, audit, business troubleshooting, etc.
- **ErrorLog Log**: minor. The internal error log of Nginx.

By default, the AccessLog and Nginx Controller logs will be mixed into the standard output stream, and there will be difficulties for log collection. This document describes how to distinguish log paths and collect logs separately.

Prerequisites

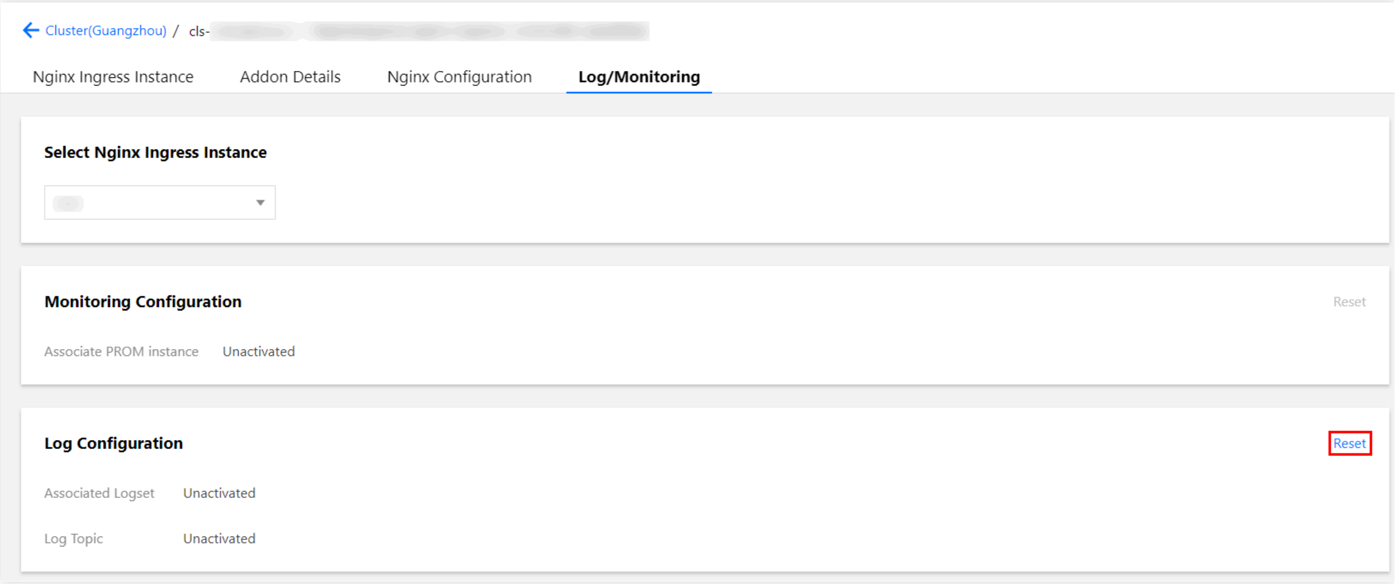
You have enabled Log Collection in **Feature Management** in the TKE console. For more information, see Enabling Log Collection.

TKE Nginx-ingress Log Collection

Log collection directions

1. Install Nginx-ingress Addon for the target cluster.
2. On **Services and Routes** > **NginxIngress** page, select an installed add-on to go to its details page.

3. On the **Log Monitoring** page, click **Reset** in the upper-right corner of the **Log Configuration** area.



Cluster(Guangzhou) / cls-[redacted]

Nginx Ingress Instance | Addon Details | Nginx Configuration | **Log/Monitoring**

Select Nginx Ingress Instance

[redacted]

Monitoring Configuration Reset

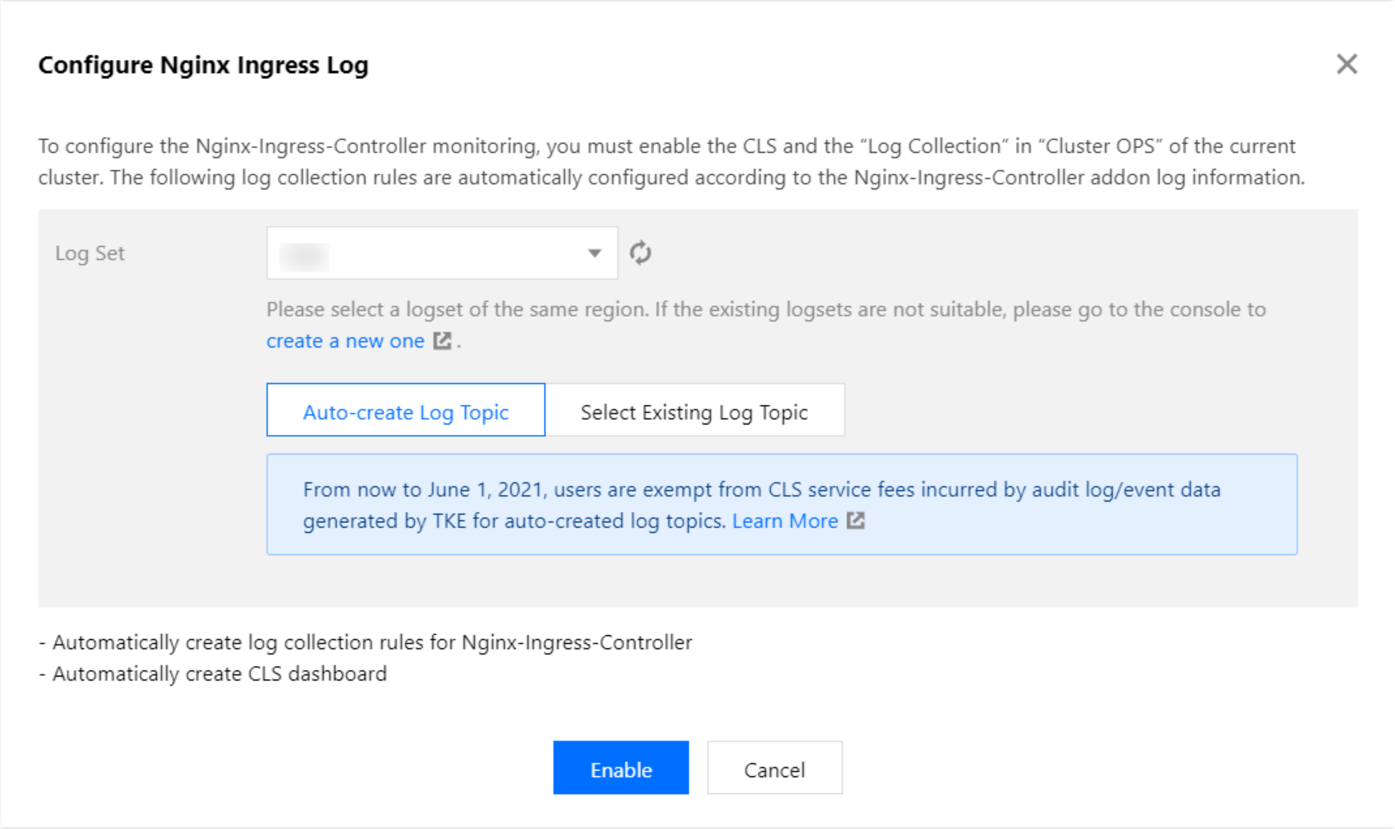
Associate PROM instance Unactivated

Log Configuration Reset

Associated Logset Unactivated

Log Topic Unactivated

4. Select or create a logset in the pop-up window, as shown in the figure below:



Configure Nginx Ingress Log

To configure the Nginx-Ingress-Controller monitoring, you must enable the CLS and the “Log Collection” in “Cluster OPS” of the current cluster. The following log collection rules are automatically configured according to the Nginx-Ingress-Controller addon log information.

Log Set [redacted]

Please select a logset of the same region. If the existing logsets are not suitable, please go to the console to create a new one .

Auto-create Log Topic | Select Existing Log Topic

From now to June 1, 2021, users are exempt from CLS service fees incurred by audit log/event data generated by TKE for auto-created log topics. Learn More

- Automatically create log collection rules for Nginx-Ingress-Controller
- Automatically create CLS dashboard

Enable | Cancel

5. Click **Enable**.

Note：

> For information on CLS billing rules and billing standards, see Billing Overview.

Log collection metrics

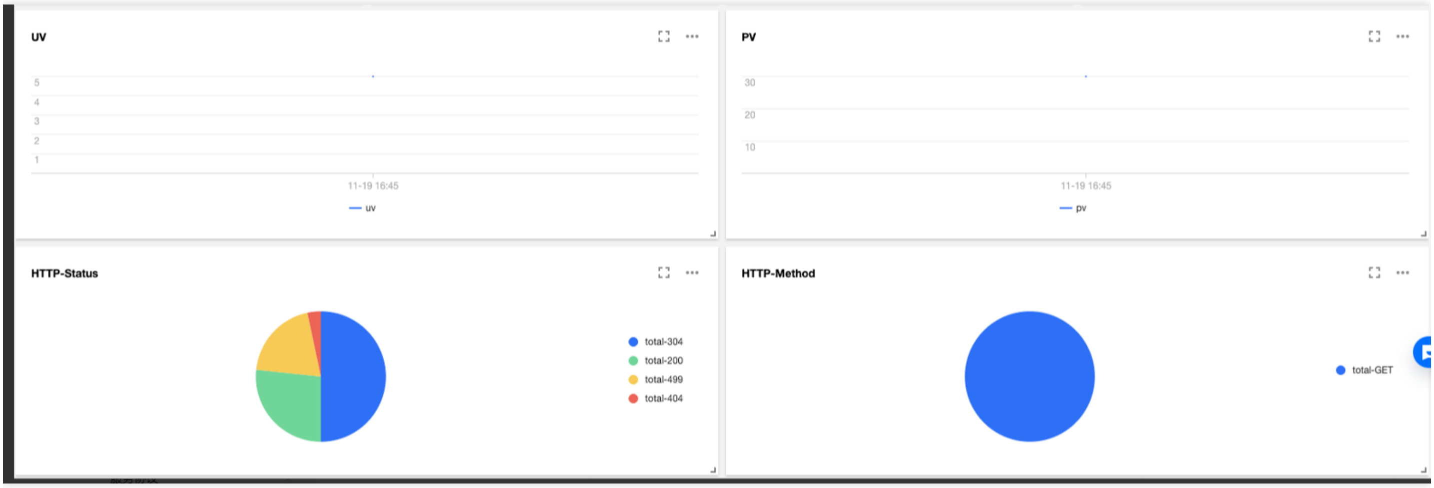
The log collection metrics are as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
name: nginx-ingress-test
resourceVersion: "7169042"
selfLink: /apis/cls.cloud.tencent.com/v1/logconfigs/nginx-ingress-test
uid: 67c96f86-4160-****-****-f6faf8d544dc
spec:
clsDetail:
extractRule:
beginningRegex: (\S+)\s-\s(\S+)\s\[(\S+)\]\s(\S+)\s\"(\w+)\s(\S+)\s([^\"]+)\"\s(\S+)\s(\S+)\s\"([^"]*)\"\s\"([^"]*)\"\s(\S+)\s(\S+)\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s(\S+)
keys:
- remote_addr
- remote_user
- time_local
- timestamp
- method
- url
- version
- status
- body_bytes_sent
- http_referer
- http_user_agent
- request_length
- request_time
- proxy_upstream_name
- proxy_alternative_upstream_name
- upstream_addr
- upstream_response_length
- upstream_response_time
- upstream_status
- req_id
logRegex: (\S+)\s-\s(\S+)\s\[(\S+)\]\s(\S+)\s\"(\w+)\s(\S+)\s([^\"]+)\"\s(\S+)\s(\S+)\s\"([^"]*)\"\s\"([^"]*)\"\s(\S+)\s(\S+)\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s\[([^\]]*)\]\s(\S+)
logType: fullregex_log
topicId: 56766bad-368e-****-****-ed77ebcdefa8
```

```
inputDetail:
containerFile:
container: controller
filePattern: nginx_access.log
logPath: /var/log/nginx
namespace: default
workload:
kind: deployment
name: nginx-ingress-nginx-controller
type: container_file
```

Nginx-ingress log dashboard

TKE will automatically create a standard log dashboard once Nginx-ingress log collection enabled. You can also configure the chart on the CLS console based on your business needs, as shown in the figure below:



References

If you need to customize log collection rules and indexes, see Custom Nginx Ingress Log.

Nginx-ingress Monitoring Configuration

Last updated：2020-12-30 10:32:08

TKE Nginx-ingress Monitoring Introduction

Nginx Controller now provides monitoring data of the addon running status. You can enable Nginx-ingress monitoring capabilities by configuring Nginx-ingress monitoring.

Prerequisites

- The cluster has associated with cloud native monitoring PROM instance.
- Cloud native monitoring PROM instance needs to be on the same network plane as Nginx.

Collection Metrics

TKE Nginx-ingress automatically configures the following collection metrics:

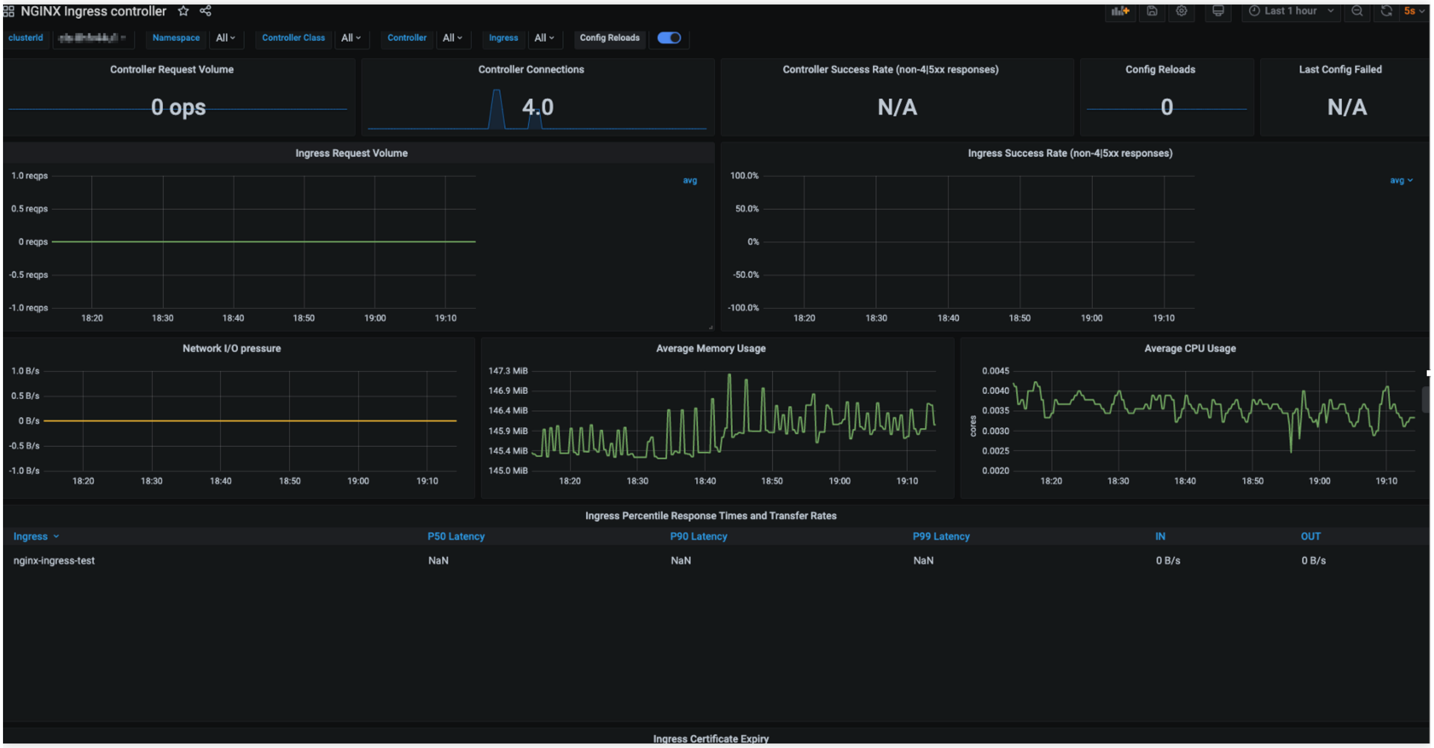
- **Nginx status**
  - nginx_ingress_controller_connections_total
  - nginx_ingress_controller_requests_total
  - nginx_ingress_controller_connections
- **Processes**
  - nginx_ingress_controller_num_procs
  - nginx_ingress_controller_cpu_seconds_total
  - nginx_ingress_controller_read_bytes_total
  - nginx_ingress_controller_write_bytes_total
  - nginx_ingress_controller_resident_memory_bytes
  - nginx_ingress_controller_virtual_memory_bytes
  - nginx_ingress_controller_oldest_start_time_seconds
- **Sockets**
  - nginx_ingress_controller_request_duration_seconds
  - nginx_ingress_controller_request_size
  - nginx_ingress_controller_response_duration_seconds
  - nginx_ingress_controller_response_size
  - nginx_ingress_controller_bytes_sent

- nginx_ingress_controller_ingress_upstream_latency_seconds

You can also configure monitoring collection metrics based on your business needs. For metric details, see Official Document.

Grafana Dashboard of Nginx-ingress Monitoring

After TKE Nginx-ingress has enabled the monitoring feature, it will associate with the cloud native monitoring PROM instance. Cloud native monitoring PROM instance provides a Grafana dashboard. You can directly go to the corresponding Grafana dashboard on the Nginx-ingress addon page, as shown in the figure below:



Ingress Percentile Response Times and Transfer Rates

Ingress	P50 Latency	P90 Latency	P99 Latency	IN	OUT
nginx-ingress-test	NaN	NaN	NaN	0 B/s	0 B/s

Installing Nginx Add-on and Instance with Terraform

Last updated：2023-06-09 15:23:55

Background

The environment configuration for the demo in this document is as follows.

The Kubernetes cluster version is v1.22.5.

The Nginx add-on version is v1.2.0.

The Nginx instance version is v1.1.3.

Step 1. Install Terraform

You can run the following command to download and install Terraform.

```
wget https://releases.hashicorp.com/terraform/1.4.6/terraform_1.4.6_linux_amd64.zip
```

The release address of v1.4.6 is https://releases.hashicorp.com/terraform/1.4.6/. You can select the appropriate installation package as needed.

Step 2. Install Nginx Add-on in the Cluster

First, install the Nginx add-on, which is an installation management tool for Nginx. Then, install the Nginx instance by using this add-on.

A sample provider.tf file is as follows.

```
# Tencent Cloud provider
terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
      version = "1.80.6"
    }
  }
}

# Tencent Cloud information (change the key pair "secret_id" and "secret_key")
provider "tencentcloud" {
    secret_id  = "********"
    secret_key = "********"
    region     = "ap-shanghai"
}

# Install the Nginx add-on (change the cluster ID "cluster_id")
resource "tencentcloud_kubernetes_addon_attachment" "addon_ingressnginx" {
  cluster_id   = "cls-xxxxxxxx"
  name = "ingressnginx"
  request_body = "{\\"kind\\":\\"App\\",\\"spec\\":{\\"chart\\":{\\"chartName\\":\\
}
```

Step 3. Declarative Installation of the Nginx Instance

For more information about Kubernetes Provider configuration, see the Official Document.

The Nginx instance configuration can be modified as needed.

IngressClass configuration (demo is used in the sample)

HPA configuration

Requests/limits configuration

A sample provider.tf file is as follows.

```
provider "kubernetes" {
  config_path = "~/.kube/config"
}

resource "kubernetes_manifest" "nginxingress_demo" {
  manifest = {
    "apiVersion" = "cloud.tencent.com/v1alpha1"
    "kind" = "NginxIngress"
    "metadata" = {
      "name" = "demo"
    }
    "spec" = {
      "ingressClass" = "demo"
      "service" = {
        "annotation" = {
          "service.kubernetes.io/service.extensiveParameters" = "{\\"InternetAccess
        }
        "type" = "LoadBalancer"
      }
      "workLoad" = {
        "hpa" = {
          "enable" = true
          "maxReplicas" = 2
          "metrics" = [
            {
              "pods" = {
                "metricName" = "k8s_pod_rate_cpu_core_used_limit"
                "targetAverageValue" = "80"
              }
              "type" = "Pods"
            },
          ]
          "minReplicas" = 1
        }
        "template" = {
          "affinity" = {}
          "container" = {
            "image" = "ccr.ccs.tencentyun.com/paas/nginx-ingress-controller:v1.1.3"
            "resources" = {
              "limits" = {
                "cpu" = "0.5"
                "memory" = "1024Mi"
              }
              "requests" = {
                "cpu" = "0.25"
                "memory" = "256Mi"
              }
            }
          }
        }
        "type" = "deployment"
      }
    }
  }
}
```

Storage Management Overview

Last updated：2021-04-19 16:23:47

Cluster storage management is an important part of preserving business data. At present, Tencent Kubernetes Engine (TKE) supports multiple classes of storage.

Storage Class

Storage Class	Description	Usage
Tencent Cloud Block Storage (CBS)	CBS provides persistent storage at the data block level. It is typically used as the primary storage device for data that requires frequent and fine-grained updates (such as file systems and databases), and it's featured with high availability, reliability, and performance.	Create persistent volumes (PVs) and persistent volume claims (PVCs), and then mount dynamic or static volumes to workloads. For more information, see Using CBS.
Tencent Cloud File Storage (CFS)	CFS provides standard NFS and CIFS/SMB file system access protocols as well as shared data sources for multiple CVM instances or other computing services. It supports elastic capacity and performance expansion. As a highly available and reliable distributed file system, CFS is suitable for scenarios such as big data analysis, media processing, and content management.	Creat PVs and PVCs, and then mount dynamic or static volumes to workloads. For more information, see Using CFS.
Tencent Cloud Object Storage (COS)	COS is a distributed storage service provided by Tencent Cloud for massive file storage. You can use COS to upload, download, and manage files in different formats.	Create PVs and PVCs, and then mount static volumes to workloads.
Others	-	TKE also supports the following following local storage options for workloads: host path, NFS disk, ConfigMap, and Secret. For more information, see Volume Management.

Note：

We recommend that you use the cloud storage service. Otherwise, when a node encounters an exception and cannot be restored, the locally stored data cannot be restored.

Related Concepts

- **PersistentVolume (PV)**: this is a storage resource in a cluster. A PV is independent of the pod lifecycle. You can create PVs of different types depending on the StorageClass type.
- **PersistentVolumeClaim (PVC)**: this is a request for storage in a cluster. For example, if a PV is a node resource used by a pod, the PVC states that the PV resource is used. If PV resources are insufficient, the PVC can create a PV dynamically.

Using COS

Last updated：2023-04-07 15:17:49

Overview

Tencent Kubernetes Engine (TKE) allows you to use Cloud Object Storage (COS) by creating PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) and mounting volumes to workloads. This document describes how to mount a COS bucket to a workload in a TKE cluster.

Preparations

1. Installing the COS add-on

Note

If your cluster has been installed with the COS-CSI add-on, skip this step.

1. Log in to the TKE console. In the left sidebar, click **Cluster**.
2. On the **Cluster** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-On Management**. On the **Add-On Management** page, click **Create**.
4. On the **Create add-on** page, select **Tencent Cloud COS**.
5. Click **Done**.

2. Creating an access key

Note

To avoid loss to your cloud assets due to root account key leakage, we recommend that you disable your root account from logging in to the console, or use the root account key to access cloud APIs but use a sub-account or collaborator account with the relevant management permissions to operate related resources. For more information, see Security Best Practice.

This document describes how to create or view an access key by using a sub-user with the relevant access and management permissions. For more information about how to create a sub-user and grant access and management permissions to the sub-user, see Creating Sub-User.

1. Use a sub-account to log in to the Cloud Access Management (CAM) console. In the left sidebar, choose **Access Key** > **API Keys**.
2. On the **API Key Management** page, click **Create Key** and wait until the key is created.

Note

One sub-user can have at most two API keys.

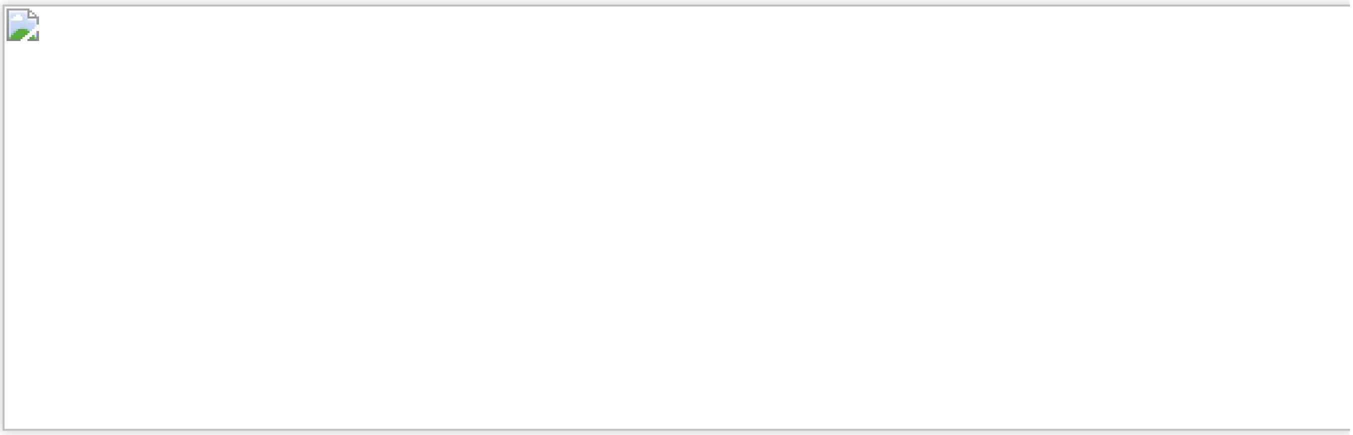
An API key is an important credential for creating Tencent Cloud API requests. To keep your assets and services secure, store your keys appropriately and change them regularly. Delete old keys when new ones are created.

3. Creating a bucket

Log in to the COS console and create a bucket. For more information, see Creating Bucket. After the bucket is created, you can find it in the bucket list.

4. Getting the bucket subdirectory

1. On the **Bucket List** page, click the name of the created bucket to go to the bucket details page.
2. Click **File List** in the left sidebar, and click the subfolder to be mounted to go to its details page. Get the subdirectory `/costest` in the top-right corner of the page as shown below:



Directions

Using COS via the console

Step 1. Create a Secret that can access COS

1. Log in to the TKE console. In the left sidebar, click **Cluster**.
2. On the **Cluster** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, choose **Configuration Management** > **Secret** in the left sidebar. On the **Secret** page, click **Create**.
4. On the **Create Secret** page, set the parameters as required, as shown below:

Name: Enter a custom name. This document uses `cos-secret` as an example.

Secret Type: Select **Opaque**. This type is suitable for saving key certificates and configuration files. The value is Base64-coded.
Effective Scope: Select **Specific Namespace**. Make sure that the Secret is created under the `kube-system` namespace.
Content: Specify the access key required by the Secret to access the bucket. The value must contain the variable names `SecretId` and `SecretKey` and their corresponding variable values. Refer to Creating an access key to create an access key, and go to the API Key Management page to get its details.
5. Click **Create Secret**.

Step 2. Create a PV that supports COS-CSI dynamic configuration

Note
This step requires a bucket. If no bucket is available in the current region, create one. For more information, see Creating Bucket.

1. On the details page of the target cluster, choose **Storage** > **PersistentVolume** in the left sidebar. On the **PersistentVolume** page, click **Create**.
2. On the **Create PersistentVolume** page, set the parameters as required, as shown below:

The main parameters are described as follows:

Creation Method: Select **Manual**.

Name: Enter a custom name. This document uses `cos-pv` as an example.

Provisioner: Select **COS**.

R/W permission: COS only supports **Multi-computer read and write**.

Note

Single machine read and write: Currently, CBS disks can only be mounted to the same machine, and therefore, only data read and write on a single machine can be processed.

Multi-computer read and write: CFS/COS can be mounted to multiple machines at a time, and therefore, data read and write on multiple machines can be processed.

Secret: Select the Secret created in Step 1. This document uses `cos-secret` as an example. Make sure that the Secret is created under the `kube-system` namespace.

Buckets list: The list of buckets used to save COS objects. You can select an available bucket as required.

Storage bucket subfolder: Enter the bucket subdirectory obtained in Getting the bucket subdirectory. This document uses `/costest` as an example. If the entered subdirectory does not exist, the system will automatically create it for you.

Domain: The default domain name is displayed. You can use this domain name to access the bucket.

Mounting options: The COSFS tool allows a bucket to be mounted locally. After the bucket is mounted, you can directly operate on the COS objects in it. This parameter is used to set related restrictions. The mounting option `-oensure_diskfree=20480` in this example indicates that when the free space of the disk where the cache files are stored is less than 20,480 MB, the COSFS tool will fail to be started.

Note

Different mounting options must be separated with spaces. For more mounting options, see Common Mounting Options.

3. Click **Create PersistentVolume**.

Step 3. Create a PVC to bind a PV

Note

Do not bind a PV that is in the bound state.

1. On the details page of the target cluster, choose **Storage** > **PersistentVolumeClaim** in the left sidebar. On the **PersistentVolumeClaim** page, click **Create**.
2. On the **Create PersistentVolumeClaim** page, set the parameters as required, as shown below:

Name: Enter a custom name. This document uses `cos-pvc` as an example.

Namespace: Select **kube-system**.

Provisioner: Select **COS**.

R/W permission: COS only supports **Multi-computer read and write**.

PersistentVolume: Select the PV that you created in Step 2. This document uses `cos-pv` as an example.

3. Click **Create PersistentVolumeClaim**.

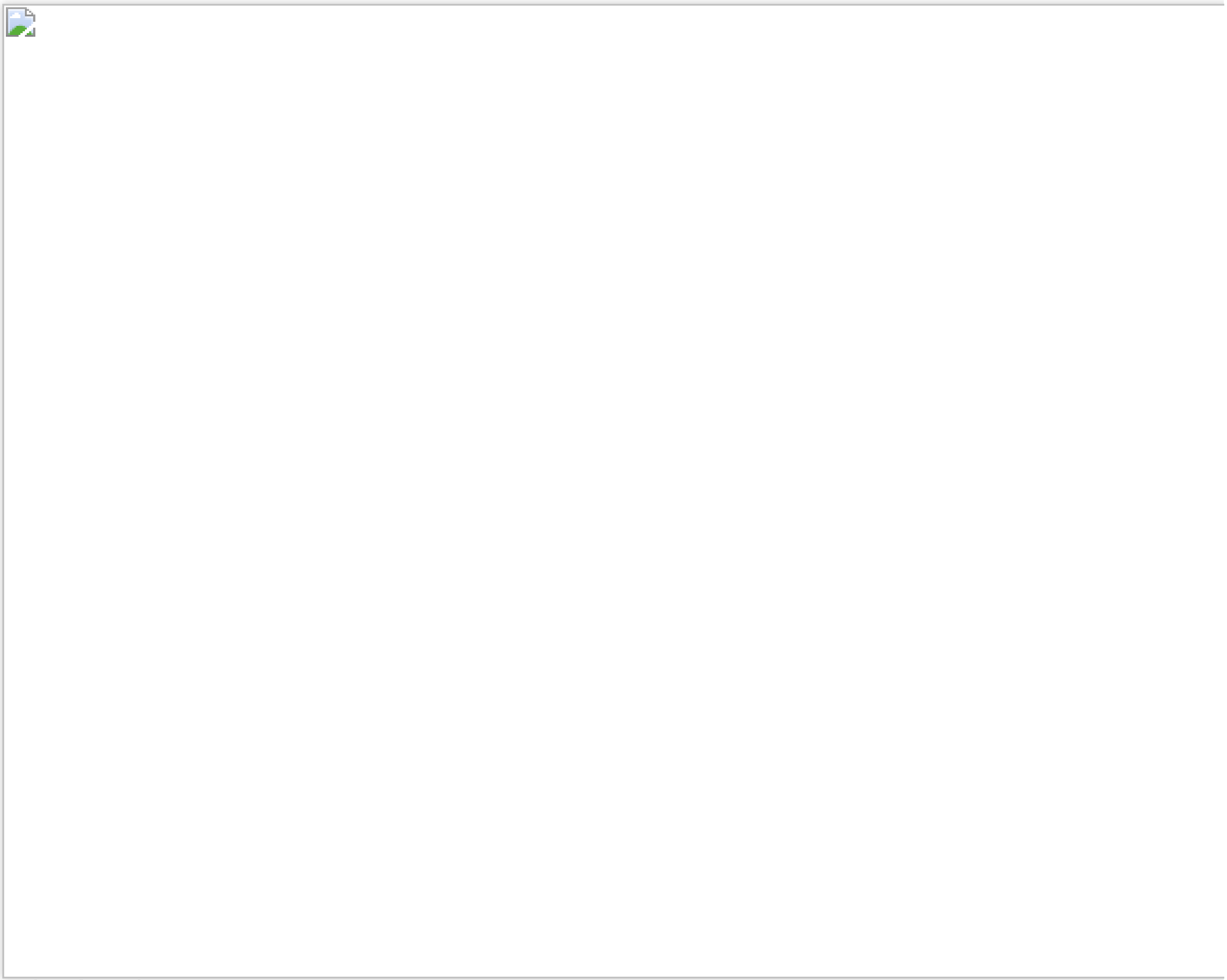
Step 4. Create a Pod that uses a PVC

Note

This step creates a Deployment workload as an example.

1. On the details page of the target cluster, choose **Workload** > **Deployment**. On the **Deployment** page, click **Create**.

2. On the **Create Deployment** page, set the parameters as required to create a Deployment. For more information, see Creating a Deployment. Then, mount a volume as required, as shown below:



Volume (optional):

Mount method: Select **Use existing PVC**.

Volume name: Enter a custom name. This document uses `cos-vol` as an example.

Select PVC: Select the PVC that you created in Step 3. This document uses `cos-pvc` as an example.

Containers in the Pod: Click **Add mount point** to set a mount point.

Volume: Select the volume `cos-vol` that you added in this step.

Target path: Enter a destination path. This document uses `/cache` as an example.

Sub-path: Mount only a sub-path or a single file in the selected volume, such as `./data` or `data` .

3. Click **Create Deployment**.

Using COS via a YAML file

Creating a secret that can access COS

You can create a secret that can access COS by using a YAML file. The YAML file template is as follows:

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: cos-secret
  # Replaced by your secret namespace.
  namespace: kube-system
data:
  # Replaced by your temporary secret file content. You can generate a temporary se
  # Note: the value must be encoded by base64.
  SecretId: VWVEJxRk5Fb0JGbDA4M...(base64 encode)
  SecretKey: Qa3p4ZTVCMFlQek...(base64 encode)
```

Creating a PV that supports COS-CSI dynamic configuration

You can create a PV to support COS-CSI dynamic configuration by using a YAML file. The YAML file template is as follows:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cos-pv
```

```
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  csi:
    driver: com.tencent.cloud.csi.cosfs
    nodePublishSecretRef:
      name: cos-secret
      namespace: kube-system
    volumeAttributes:
      # Replaced by the url of your region.
      url: http://cos.ap-XXX.myqcloud.com
      # Replaced by the bucket name you want to use.
      bucket: XXX-1251707795
      # You can specify sub-directory of bucket in cosfs command in here.
      path: /costest
       # You can specify any other options used by the cosfs command in here.
    # additional_args: "-oallow_other"# Specify a unique volumeHandle like bucket n
    volumeHandle: XXX
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
```

Creating a PVC that binds a PV

You can create a PVC that binds the preceding PV by using a YAML file. The YAMl file template is as follows:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cos-pvc
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  # You can specify the pv name manually or just let kubernetes to bind the pv and
  # volumeName: cos-pv
  # Currently cos only supports static provisioning, the StorageClass name should b
  storageClassName: ""
```

Creating a pod that uses a PVC

You can create a pod by using a YAML file. The YAML file template is as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-cos
spec:
  containers:
  - name: pod-cos
    command: ["tail", "-f", "/etc/hosts"]
    image: "centos:latest"
    volumeMounts:
    - mountPath: /data
      name: cos
    resources:
      requests:
        memory: "128Mi"
        cpu: "0.1"
  volumes:
  - name: cos
    persistentVolumeClaim:
      # Replaced by your pvc name.
      claimName: cos-pvc
```

More Information

For more information about how to use COS, see README_COSFS.md.

Use File to Store CFS

CFS Instructions

Last updated：2020-10-19 10:50:26

Overview

Tencent Kubernetes Engine (TKE) allows you to use Cloud File Storage (CFS) by creating persistent volumes (PVs) and persistent volume claims (PVCs) and mounting volumes to workloads. This document describes how to mount a CFS disk to a workload in a cluster by using the following two methods:

- Method 1: dynamically creating a CFS disk
- Method 2: using an existing CFS disk

Preparations

Installing the CFS add-on

Note：

If your cluster has been installed with the CFS-CSI add-on, skip this step.

1. Log in to the TKE console.
2. Click **Cluster** in the left sidebar to go to the **Cluster Management** page.
3. Choose the ID of the cluster for which you want to create an add-on and click **Add-On Management** in the left sidebar on the cluster details page.
4. On the **Add-On Management** page, click **Create** to go to the **Create Add-On** page.
5. Select **CFS** and click **Done**.

Directions

Dynamically creating a CFS disk

To dynamically create a CFS disk, complete the following steps:

1. Create a StorageClass of the CFS type and define a CFS template.
2. Create a PVC by using the StorageClass and further define the CFS parameters.

3. Select the created PVC when creating a workload volume and configure the container mount point.

Using an existing CFS disk

To use an existing CFS disk, complete the following steps:

1. Create a PV by using an existing CFS disk.
2. When creating a PVC, set the same StorageClass and capacity as that for the preceding PV.
3. When creating a workload, select the preceding PVC.

Related Information

For more information on how to use CFS, see README_CFS.md.

Managing CFS Templates by Using a StorageClass

Last updated：2022-12-13 18:23:37

Overview

A cluster admin can use StorageClass to define different storage classes for Tencent Kubernetes Engine (TKE) clusters. TKE provides the block storage StorageClass by default. You can use both StorageClass and PersistentVolumeClaim to dynamically create required storage resources.

This document describes how to create a StorageClass of the Cloud File Storage (CFS) type by using the console and Kubectl as well as how to customize the template required by CFS disks.

Prerequisites

1. Install the CFS add-on

If your cluster has been installed with the CFS-CSI add-on, skip this step; otherwise, install it as instructed in CFS Instructions.

2. Create a subnet

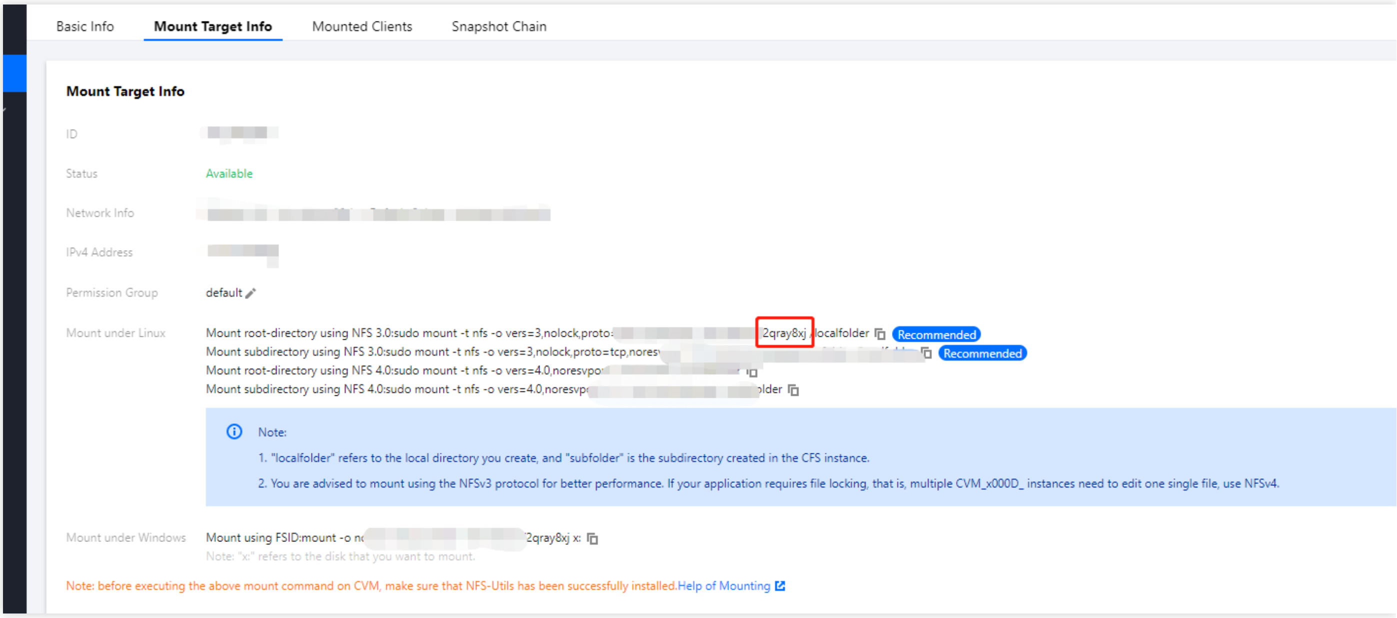
When creating a StorageClass, you need to set the CFS subnet to ensure that every AZ in the CFS's VPC has a suitable subnet. We recommend you create a subnet in advance as instructed in Creating Subnets.

3. Create a permission group and add permission group rules

When creating a StorageClass, you need to configure a suitable permission group for the file system. We recommend you create a permission group in advance as instructed in Managing Permissions.

4. Get the file system FSID

1. In the CFS console, click the ID of the file system for which you want to obtain the FSID. The details page of the file system appears.
2. Select the **Mount target info** tab and get the file system FSID next to **Mount to Linux** such as `a43qadkl` as shown below:



Mount Target Info

ID [redacted]

Status Available

Network Info [redacted]

IPv4 Address [redacted]

Permission Group default

Mount under Linux

Mount root-directory using NFS 3.0:sudo mount -t nfs -o vers=3,nolock,proto:[redacted]2qray8xj /localfolder Recommended
Mount subdirectory using NFS 3.0:sudo mount -t nfs -o vers=3,nolock,proto=tcp,nores[redacted] Recommended
Mount root-directory using NFS 4.0:sudo mount -t nfs -o vers=4.0,noresvpo[redacted]
Mount subdirectory using NFS 4.0:sudo mount -t nfs -o vers=4.0,noresvp[redacted]older

Note:
1. "localfolder" refers to the local directory you create, and "subfolder" is the subdirectory created in the CFS instance.
2. You are advised to mount using the NFSv3 protocol for better performance. If your application requires file locking, that is, multiple CVM_x000D_ instances need to edit one single file, use NFSv4.

Mount under Windows

Mount using FSID:mount -o n[redacted]2qray8xj x:
Note: "x:" refers to the disk that you want to mount.

Note: before executing the above mount command on CVM, make sure that NFS-Utils has been successfully installed.Help of Mounting

Note：

For better stability, when you create a PV by YAML and use the NFSv3 protocol for mounting, you need to specify the FSID of the file system to be mounted.

Console

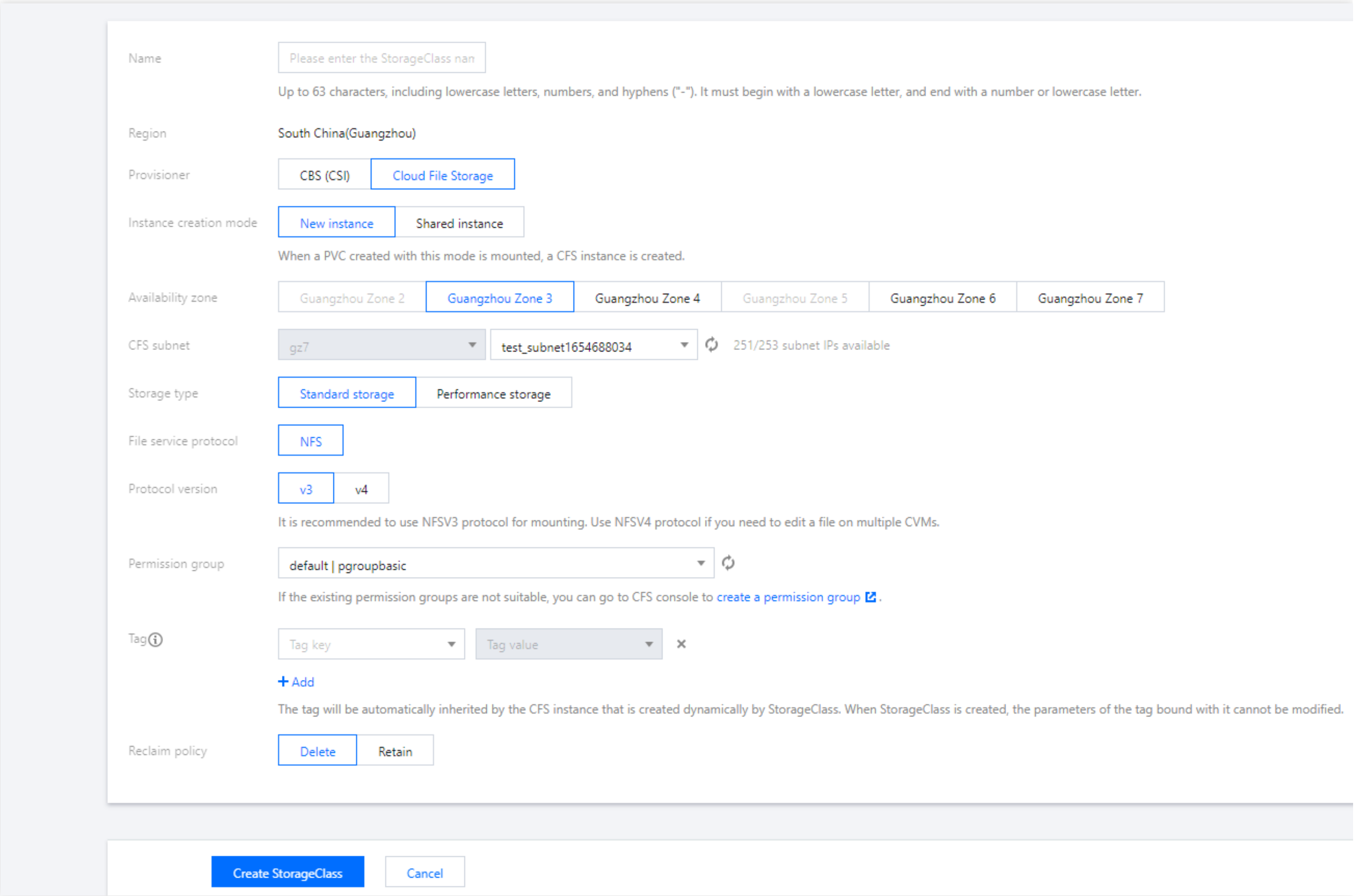
Creating a StorageClass

1. Log in to the TKE console and select **Cluster** on the left sidebar.

2. On the **Cluster management** page, click the ID of the target cluster to go to the cluster details page.

3. Select **Storage** > **StorageClass** on the left sidebar.

4. Click **Create** to enter the **Create StorageClass** page, where you can configure StorageClass parameters.



Name: Please enter the StorageClass nan

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Region: South China(Guangzhou)

Provisioner: CBS (CSI) | Cloud File Storage

Instance creation mode: New instance | Shared instance

When a PVC created with this mode is mounted, a CFS instance is created.

Availability zone: Guangzhou Zone 2 | Guangzhou Zone 3 | Guangzhou Zone 4 | Guangzhou Zone 5 | Guangzhou Zone 6 | Guangzhou Zone 7

CFS subnet: gz7 | test_subnet1654688034 251/253 subnet IPs available

Storage type: Standard storage | Performance storage

File service protocol: NFS

Protocol version: v3 | v4

It is recommended to use NFSV3 protocol for mounting. Use NFSV4 protocol if you need to edit a file on multiple CVMs.

Permission group: default | pgroupbasic

If the existing permission groups are not suitable, you can go to CFS console to create a permission group.

Tag: Tag key | Tag value

The tag will be automatically inherited by the CFS instance that is created dynamically by StorageClass. When StorageClass is created, the parameters of the tag bound with it cannot be modified.

Reclaim policy: Delete | Retain

Configuration Item	Description
Name	Enter the StorageClass name, for example, `cfs-storageclass`.
Region	It is the region of the cluster by default.
Provisioner	**Provisioner** can be **CBS (CSI)** or **Cloud File Storage**. Here, **Cloud File Storage** is selected.

Configuration Item	Description
Instance creation mode	It can be **New instance** or **Shared instance**. ◦ New instance: During mounting, a CFS instance is created for each PVC by default. ◦ Shared instance: During mounting, PVCs correspond to different sub-directories of the same CFS instance. The shared CFS instance and its sub-directories are created automatically by the system. Note The **shared instance** mode is supported by the CFS-CSI add-on on v1.0.1 or later. Upgrade your add-on in time. Use instructions are as follows: ◦ For a StorageClass in shared instance mode, the **Reclaim policy** is **Retain**. ◦ When the StorageClass is used to dynamically create a PVC for the first time, a CFS instance will be created by default, along with its sub-directories to implement isolated mounting of PVCs. ◦ CFS instances created by different StorageClasses in shared instance mode are different. We recommend you limit the number of instances.
AZ	In the current region, select an AZ that supports CFS. Different AZs in the same region support different storage classes. For more information, see Recommended Regions .
CFS subnet	Set the subnet range of the CFS in the current AZ.
Storage Type	CFS provides **Standard Storage** and **Performance Storage**. Different AZs in the same region support different storage types. Select one as needed. ◦ Standard Storage: It features cost-effectiveness and large capacity, making it suitable for scenarios such as data backup, file sharing, and log storage. ◦ Performance Storage: It features high throughput and IOPS, making it suitable for IO-intensive workloads such as high-performance computing, media asset rendering, machine learning, DevOps, and OA.
File service protocol	It is **NFS** by default to allow for pass-through access to files and file systems on the server.
Protocol version	We recommend you use NFSv3 for better performance. If your application relies on file locking (that is, multiple CVM instances are needed to edit a file), use NFSv4 for mounting.
Permission Group	Configure a permission group for the file system, which is used to manage the access and read/write permissions of clients that access the file system over the same network. Select a permission group as needed. If no such permission group is available, create one on the Permission Group page.

Configuration Item	Description
Reclaim policy	It can be **Delete** or **Retain**. The latter is recommended out of data security considerations. ◦ Delete: If a PV is dynamically created through a PVC, the PV and storage instance bound to the PVC will be automatically terminated when the PVC is terminated. ◦ Retain: If a PV is dynamically created through a PVC, the PV and storage instance bound to the PVC will be retained when the PVC is terminated.
Label	Select the cloud tag to be bound to the CFS instance. The tag will be automatically inherited by the CFS instance that is created dynamically by a StorageClass. After creation, the parameters of the bound tag cannot be modified. If the existing tags are not suitable, create one in the Tag console.

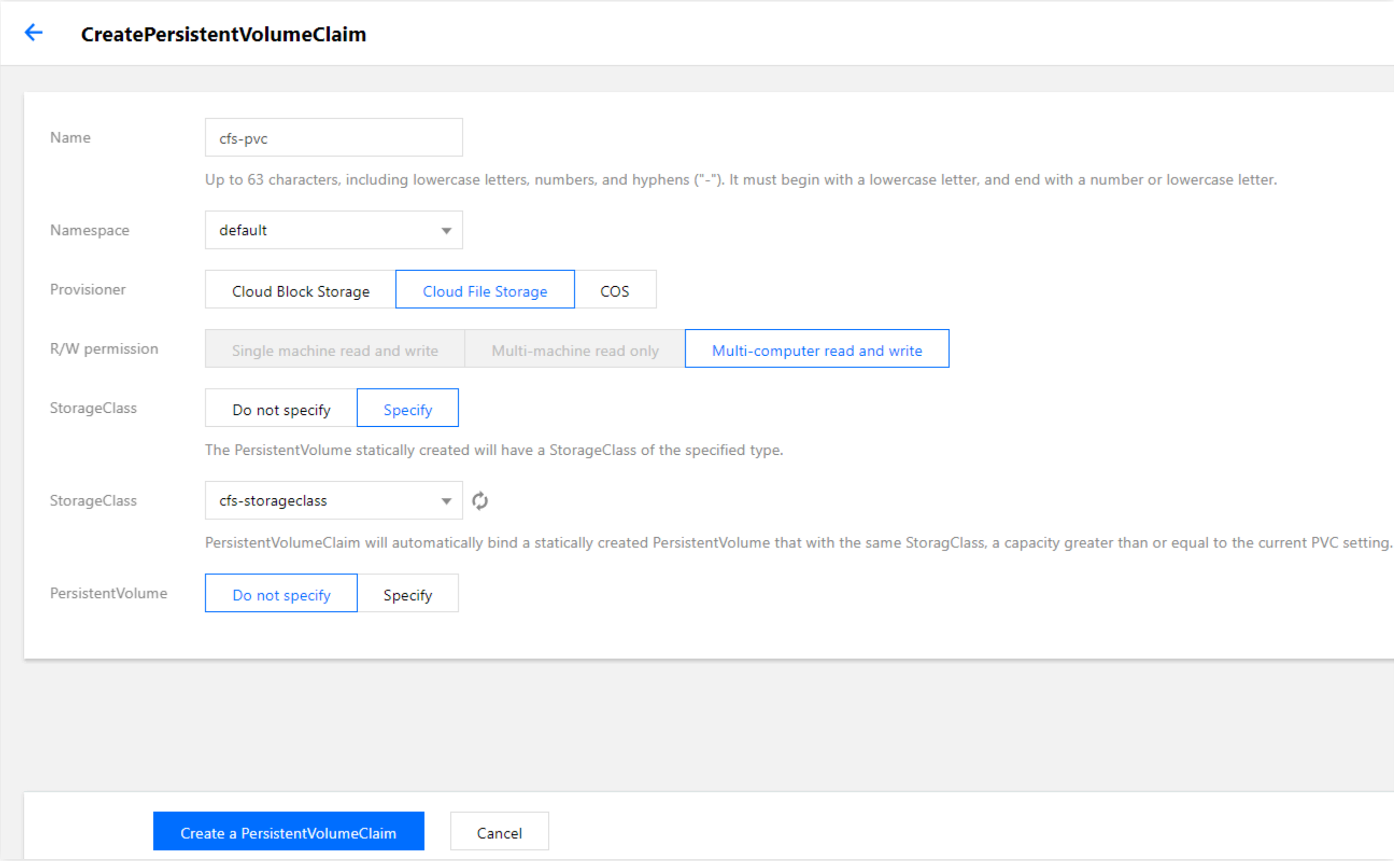
5. Click **Create StorageClass** to complete the process.

Creating a PVC by using the specified StorageClass

1. On the **Cluster management** page, select the ID of the cluster for which a PVC needs to be created.

2. On the cluster details page, select **Storage** > **PersistentVolumeClaim** on the left sidebar.

3. Click **Create** to enter the **Create PersistentVolumeClaim** page, where you can set key PVC parameters.



CreatePersistentVolumeClaim

Name: cfs-pvc

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Namespace: default

Provisioner: Cloud Block Storage | Cloud File Storage | COS

R/W permission: Single machine read and write | Multi-machine read only | Multi-computer read and write

StorageClass: Do not specify | Specify

The PersistentVolume statically created will have a StorageClass of the specified type.

StorageClass: cfs-storageclass

PersistentVolumeClaim will automatically bind a statically created PersistentVolume that with the same StoragClass, a capacity greater than or equal to the current PVC setting.

PersistentVolume: Do not specify | Specify

Configuration Item	Description
Name	Enter the `PersistentVolumeClaim` name, for example, `cfs-pvc`.
Namespaces	A namespace is used to assign cluster resources. Here, **default** is selected.
Provisioner	Select **Cloud File Storage**.
Read/write permission	CFS only supports **Multi-computer read and write**.

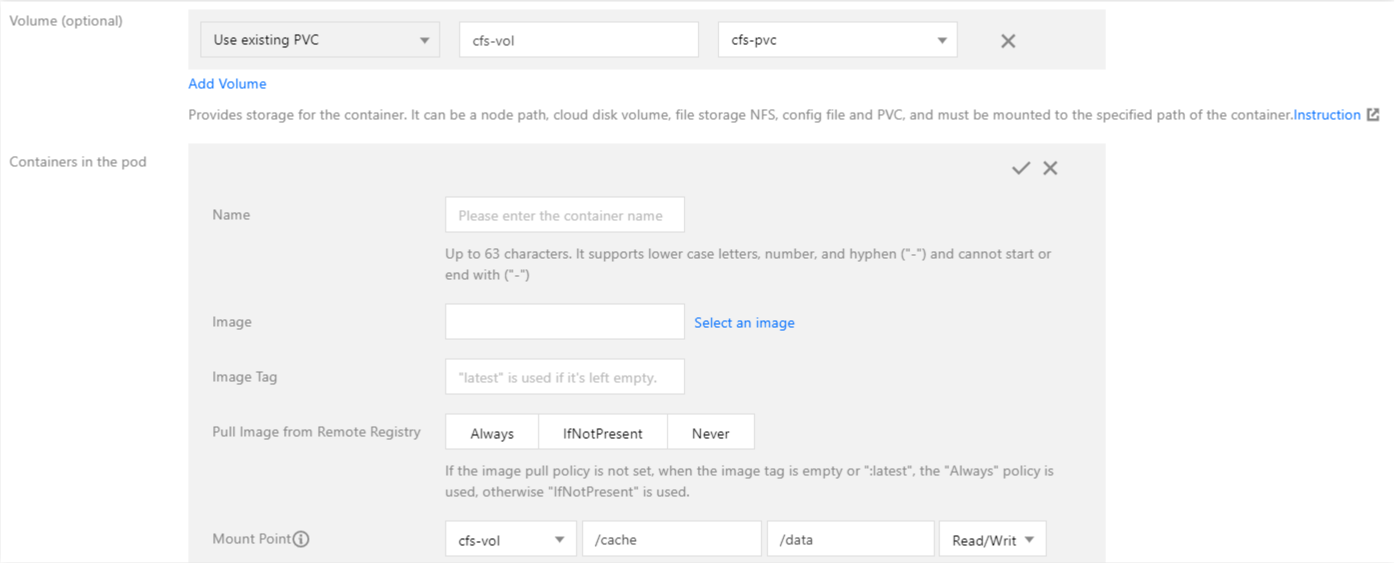
Configuration Item	Description
StorageClass	Specify the StorageClass as needed. Here, **Specify** is selected. `cfs-storageclass` created in the Creating a StorageClass step is used as an example. Note ◦ The PVC and PV will be bound to the same StorageClass. ◦ If you select **Do not specify**, the value of `StorageClass` for the corresponding PVC is null, and the value of the `storageClassName` field in the corresponding YAML file is a null string.
PersistentVolume	Specify the PersistentVolume as needed. Here, **Do not specify** is selected. Note ◦ The system first searches the current cluster for PVs that meet the binding rules. If there are no such PVs, the system dynamically creates a PV to be bound based on the PVC and StorageClass parameters. ◦ Either the StorageClass or PersistVolume should be specified. ◦ For more information on **Do not specify** for **PersistentVolume**, see PV and PVC binding rules.

4. Click **Create PersistentVolumeClaim**.

Creating a workload to use a PVC volume

> Note：
>
> This step creates a Deployment workload as an example.

1. On the **Cluster management** page, select the target cluster ID to go to the **Deployment** page of the cluster for which the workload needs to be deployed.
2. Click **Create** to enter the **Create Workload** page. For detailed directions, see Deployment Management. Then, mount a volume based on the following information.



Volume (optional): Use existing PVC | cfs-vol | cfs-pvc

Add Volume

Provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container.Instruction

Containers in the pod

Name: Please enter the container name

Up to 63 characters. It supports lower case letters, number, and hyphen ("-") and cannot start or end with ("-")

Image: Select an image

Image Tag: "latest" is used if it's left empty.

Pull Image from Remote Registry: Always | IfNotPresent | Never

If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.

Mount Point: cfs-vol | /cache | /data | Read/Writ

- **Volume (optional)**:
  - **Mount method**: Select **Use existing PVC**.
  - **Volume name**: Set a custom name. This document uses `cfs-vol` as an example.
  - **Select PVC**: Select `cfs-pvc` , which you created in the step of Creating a PVC.
- **Containers in the Pod**: Click **Add mount target** to set a mount target.
  - **Volume**: Select the volume `cfs-vol` that you added in this step.
  - **Destination path**: Enter a destination path. This document uses `/cache` as an example.
  - **Sub-path**: Mount only a sub-path or a single file in the selected volume, such as `/data` or `/test.txt` .

3. Click **Create Workload** to complete the process.

> Note：
>
> If you use the PVC mount method of CFS, the volume can be mounted to multiple nodes.

kubectl

Creating a StorageClass

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: cfs
```

```
parameters:
# subdir-share: "true"
vpcid: vpc-xxxxxxxx
subnetid: subnet-xxxxxxxx
vers: "3"
resourcetags: ""
provisioner: com.tencent.cloud.csi.cfs
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The following parameters can be configured:

Parameter	Required	Description
zone	No	It defines the region for the CFS instance.
pgroupid	No	It defines the permission group for the CFS instance.
storagetype	No	It defaults to Standard Storage (SD). Valid values: SD (Standard Storage) HP (High-Performance Storage).
subdir-share	Yes	It indicates the shared instance mode for instance creation by StorageClass.
vpcid	Yes	It indicates the ID of the VPC where the file is stored.
subnetid	Yes	It indicates the ID of the subnet where the file is stored.
vers	Yes	It indicates the version of the protocol used by the add-on to connect to the file system. The dynamically created PVs inherit this parameter. The versions "3" and "4" are supported.
resourcetags	Yes	It indicates the cloud tag of the file system. A corresponding Tencent Cloud tag is applied on the generated file system. Multiple tags are separated by comma. For example, "a:b,c:d".

Creating a PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: cfs
namespace: default
spec:
accessModes:
- ReadWriteMany
resources:
```

```
requests:
storage: 10Gi
storageClassName: cfs
volumeMode: Filesystem
volumeName: XXX # You don't need to specify it for dynamic creation. For static c
reation, you need to specify the PV instance ID.
```

Parameter	Required	Description
spec.accessModes	No	The cfs storage supports Multiple-Read-Multiple-Write.
spec.resources.requests.storage	Yes	The storage capacity only depends on the type of the file system.

Note：

1. CFS supports expanding the storage capacity of the file system according to the file size. The requests and applications are not interrupted during the expansion. The default CFS instance capacity is 10 GiB, and the upper limit of the capacity depends on the product type. For details, see System Restrictions.
2. The PVs dynamically created through a PVC inherit the parameters configured in StorageClass. The parameters are generated automatically by the storage add-on.

Managing CFS by Using PVs and PVCs

Last updated：2022-04-06 10:29:27

Overview

Tencent Kubernetes Engine (TKE) allows you to create PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) and use existing PVCs when creating workloads and adding volumes so that you can manage a file system by using the PVs and PVCs.

> Note：
> Different regions support different file storage capabilities. You need to select a region based on your requirement. For more information, see Storage Class and Performance Specification.

Preparations

Installing the CFS add-on

> Note：
> If your cluster has been installed with the CFS-CSI add-on, skip this step.

1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to go to the **Cluster Management** page.
3. Click the ID of the cluster for which you want to create an add-on to go to the **Cluster Details** page.
4. Select **Add-On Management** > **Create** to go to **Create an Add-On** page.
5. On the **Create Add-on** page, select **CFS** and click **Done**.

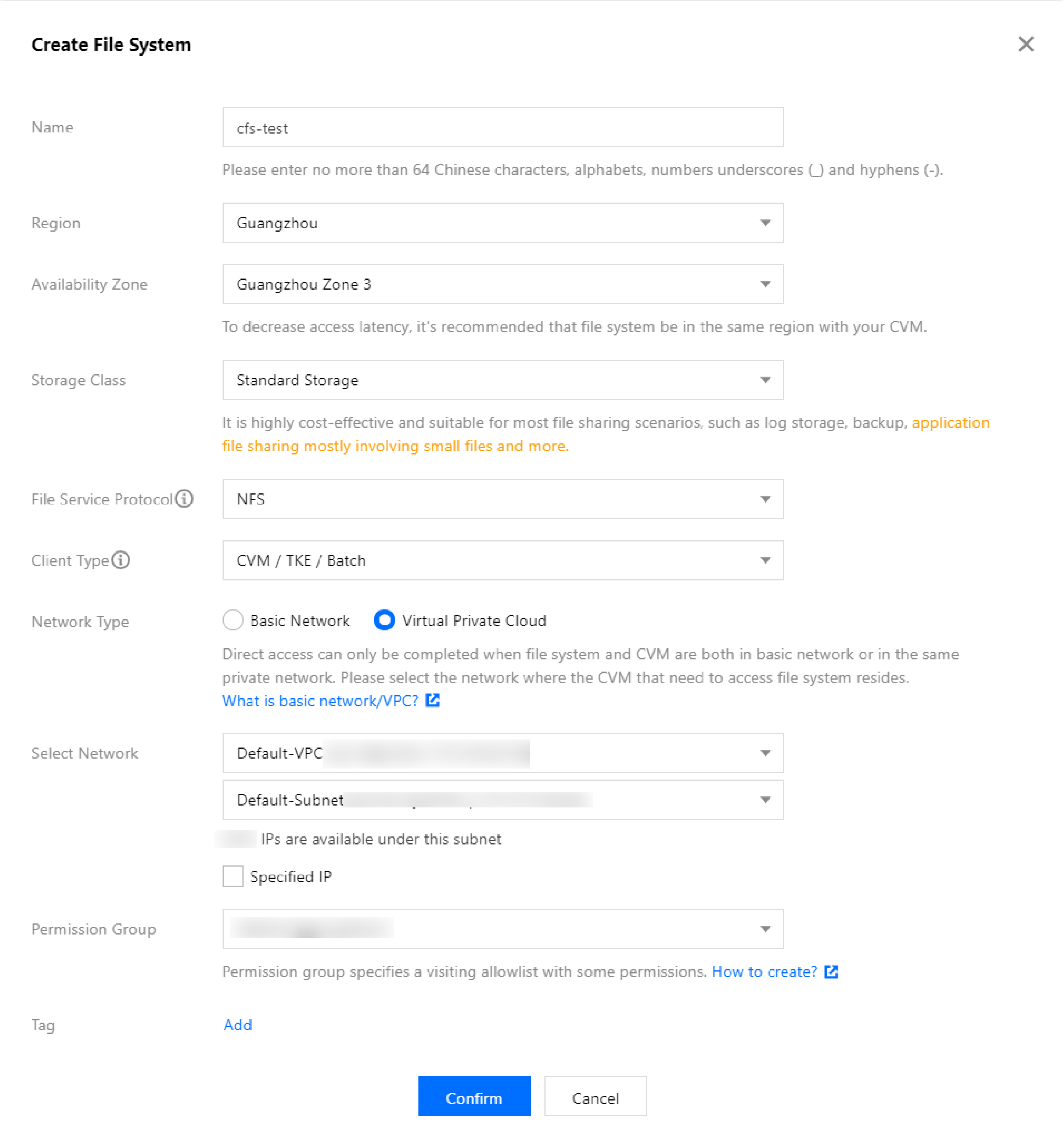
Creating a StorageClass in the console

To statically create a PV of the file storage type, you need to bind an available StorageClass of the same type. For more information, see Creating a StorageClass via the Console.

Creating a CFS file system

1. Log in to the CFS console and go to the **File System** page.

2. Click **Create**. Select the file system type first: **Standard Storage** or **Performance Storage**. The supported types vary by AZ. For more information, see Available Regions. Then, configure the detailed settings:



Create File System

Name: cfs-test

Please enter no more than 64 Chinese characters, alphabets, numbers underscores (_) and hyphens (-).

Region: Guangzhou

Availability Zone: Guangzhou Zone 3

To decrease access latency, it's recommended that file system be in the same region with your CVM.

Storage Class: Standard Storage

It is highly cost-effective and suitable for most file sharing scenarios, such as log storage, backup, application file sharing mostly involving small files and more.

File Service Protocol: NFS

Client Type: CVM / TKE / Batch

Network Type: ○ Basic Network ● Virtual Private Cloud

Direct access can only be completed when file system and CVM are both in basic network or in the same private network. Please select the network where the CVM that need to access file system resides.
What is basic network/VPC?

Select Network: Default-VPC [redacted]

Default-Subnet [redacted]

[redacted] IPs are available under this subnet

□ Specified IP

Permission Group: [redacted]

Permission group specifies a visiting allowlist with some permissions. How to create?

Tag: Add

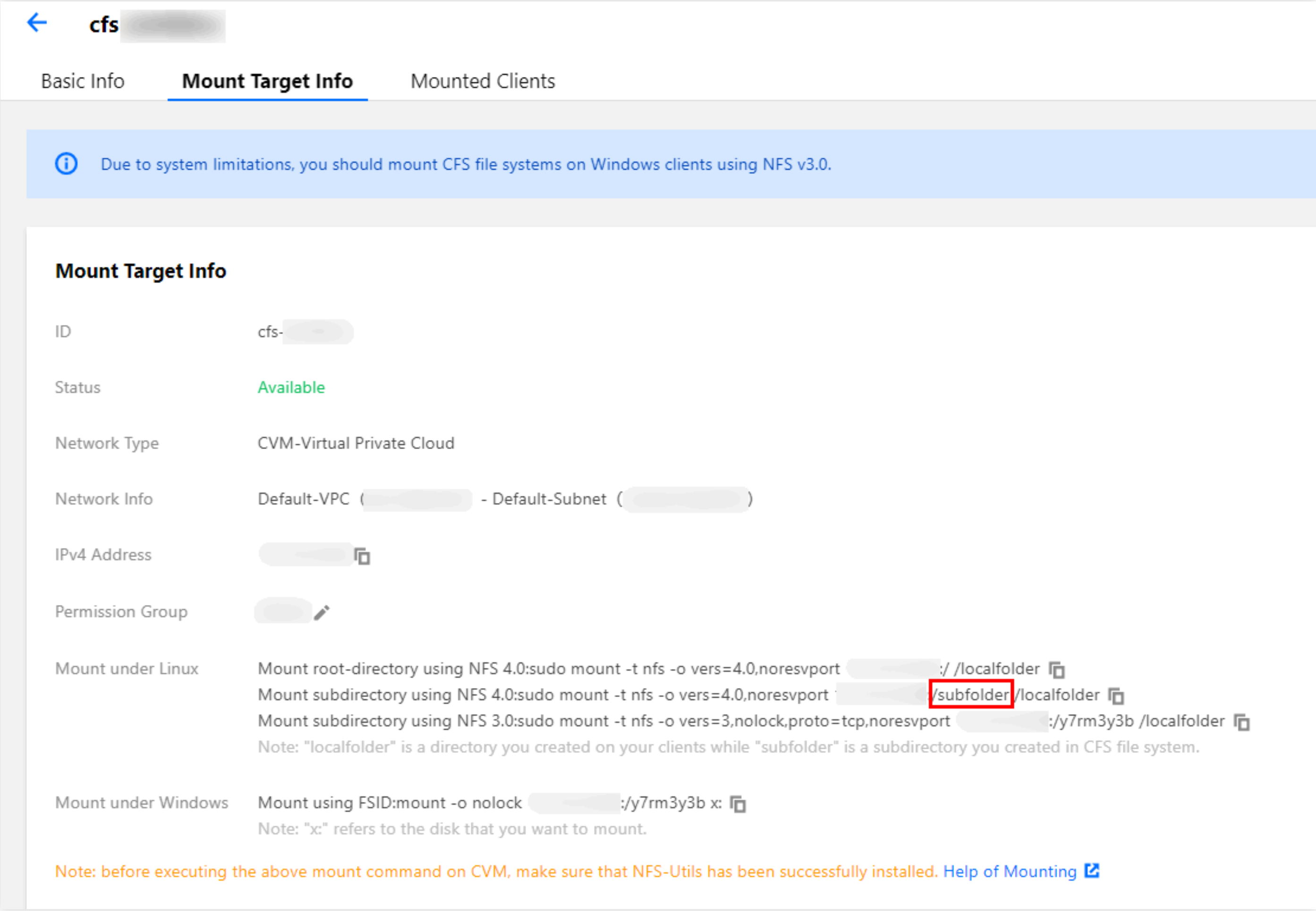
- **Name**: set a custom name. This document uses `cfs-test` as an example.
- **Region*: select a region in which to create the file system and ensure that the file system and the cluster are in the same region.
- **Availability Zone**: select an availability zone in which to create the file system.

- **File Service Protocol**: select a protocol type for the file system. Valid values include **NFS** and **CIFS/SMB**.
  - NFS: better suited to Linux and Unix clients.
  - CIFS/SMB: better suited to Windows clients.
- **Data Source**: you can create a file system from a snapshot.
- **Select Network**: select a network to ensure that the file system and the cluster that uses the file system are in the same VPC.
- **Permission Group**: each file system must be bound to a permission group. The permission group specifies an allowlist that can access the file system and lists the read and write permissions.
- **Tag**:
  - If you have a tag, you can add it to the file system.
  - If you do not have a tag, log in to the Tag console to create the required tag, and then bind the tag to the file system. You can also add a tag to the file system after the file system is created.

3. Click **Buy Now** and wait for the creation to succeed.

Getting the file system subdirectory

1. On the **File System** page, click the ID of the file system for which you want to obtain the destination subdirectory. The details page of the file system appears.
2. Select the **Mount Point Info** tab and obtain the subdirectory `/subfolder` of this file system from **Mount to Linux** as shown below:



cfs [redacted]

Due to system limitations, you should mount CFS file systems on Windows clients using NFS v3.0.

Mount Target Info

ID	cfs-[redacted]
Status	Available
Network Type	CVM-Virtual Private Cloud
Network Info	Default-VPC ([redacted]) - Default-Subnet ([redacted])
IPv4 Address	[redacted]
Permission Group	[redacted]
Mount under Linux	Mount root-directory using NFS 4.0:sudo mount -t nfs -o vers=4.0,noresvport [redacted]:/ /localfolder Mount subdirectory using NFS 4.0:sudo mount -t nfs -o vers=4.0,noresvport [redacted]/subfolder/localfolder Mount subdirectory using NFS 3.0:sudo mount -t nfs -o vers=3,nolock,proto=tcp,noresvport [redacted]:/y7rm3y3b /localfolder Note: "localfolder" is a directory you created on your clients while "subfolder" is a subdirectory you created in CFS file system.
Mount under Windows	Mount using FSID:mount -o nolock [redacted]:/y7rm3y3b x: Note: "x:" refers to the disk that you want to mount.

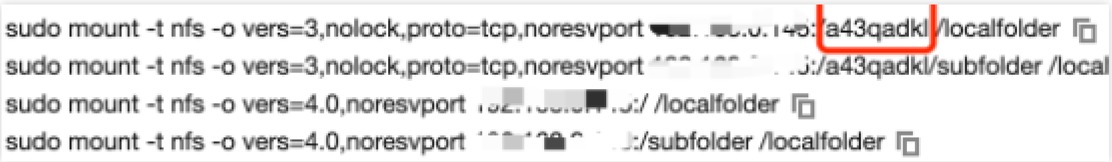
Note: before executing the above mount command on CVM, make sure that NFS-Utils has been successfully installed. Help of Mounting

- `localfolder` : indicates the local directory that you created.
- `subfolder` : it indicates a subdirectory that you created in the CFS file system. The subdirectory of the file system is `/subfolder` .

Getting the file system FSID

Note：
For better stability, when using the NFSv3 protocol to mount, you need to specify the FSID of the file system to be mounted.

1. In the CFS console, click the ID of the file system for which you want to obtain the FSID. The details page of the file system appears.
2. Select the **Mount Target Info** tab and get the file system FSID next to **Mount to Linux** such as `a43qadkl` as shown below:



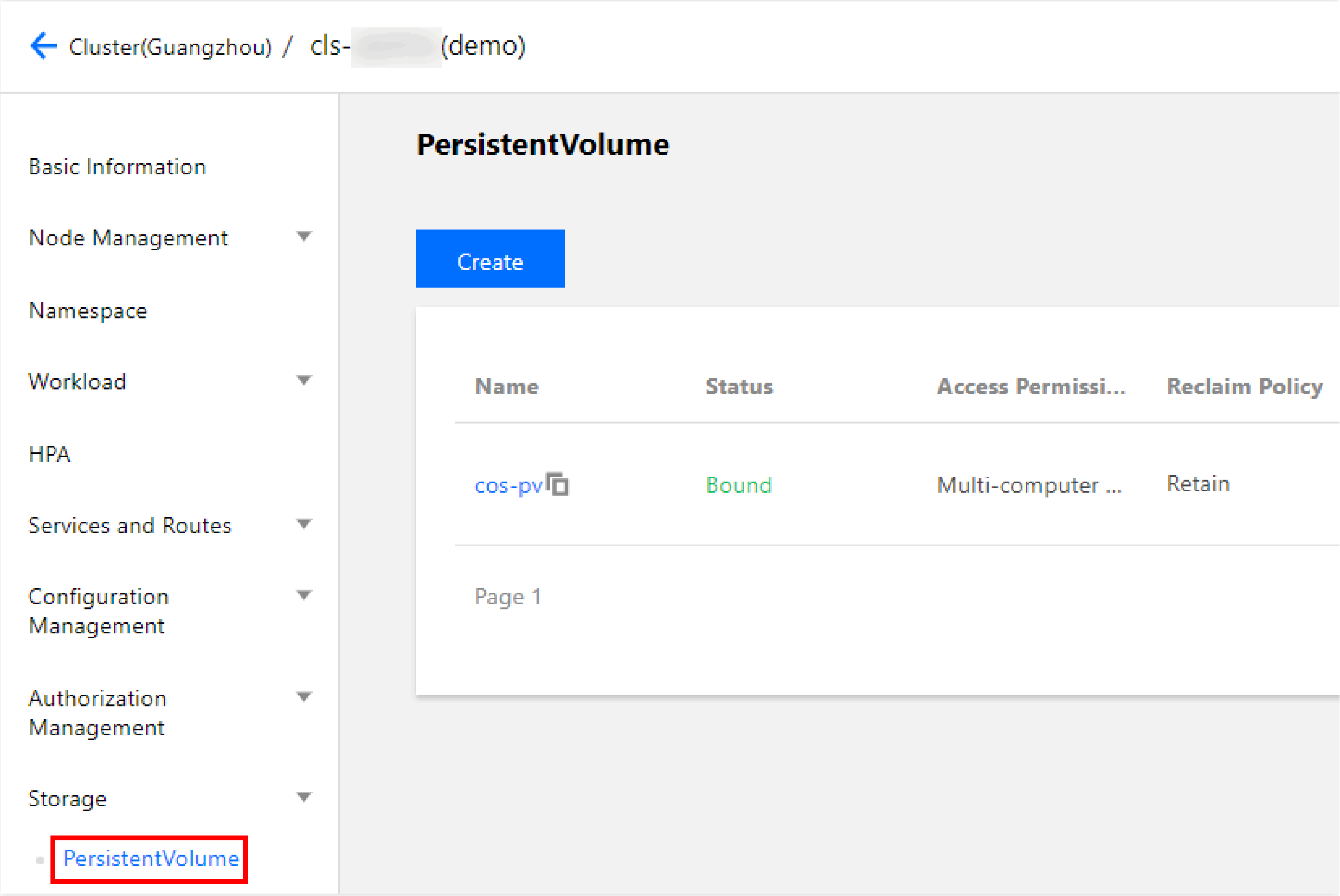
```
sudo mount -t nfs -o vers=3,nolock,proto=tcp,noresvport [redacted]:/a43qadkl /localfolder
sudo mount -t nfs -o vers=3,nolock,proto=tcp,noresvport [redacted]:/a43qadkl/subfolder /local
sudo mount -t nfs -o vers=4.0,noresvport [redacted]:/ /localfolder
sudo mount -t nfs -o vers=4.0,noresvport [redacted]:/subfolder /localfolder
```

Directions

Creating a PV statically

> Note：
> A statically created PV is suitable for scenarios where file storage already contains data and is used in a cluster.

1. Log in to the TKE console and select **Clusters** on the left sidebar.
2. On the **Cluster Management** page, select the ID of the cluster where the PV needs to be created. The cluster management page of the PV to create appears.
3. Choose **Storage** > **PersistentVolume** in the left sidebar to go to the **PersistentVolume** page, as shown in the following figure.

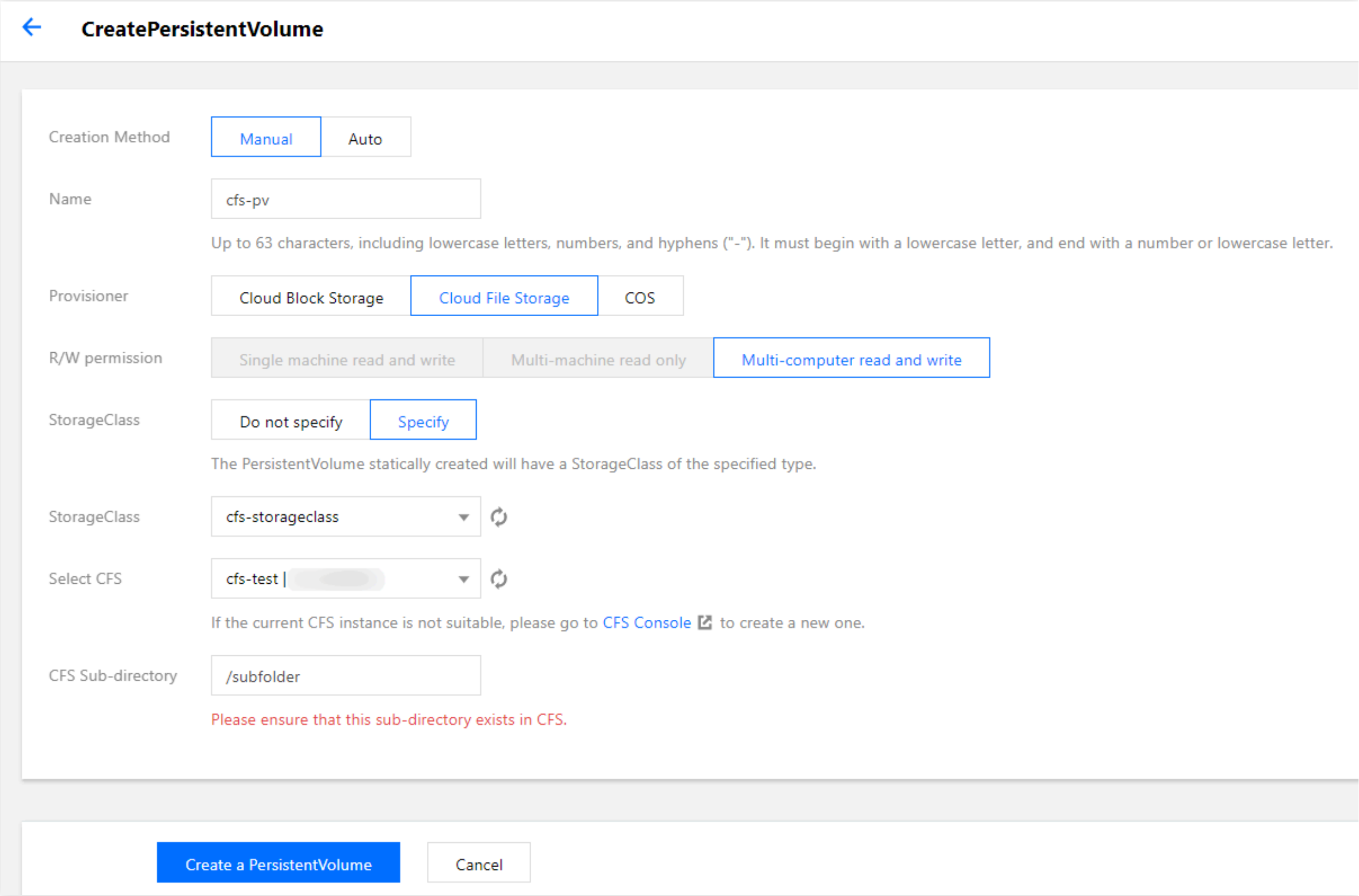


PersistentVolume

Name	Status	Access Permissi...	Reclaim Policy
cos-pv	Bound	Multi-computer ...	Retain

Page 1

4. Click **Create** to go to the **Create PersistentVolume** page, where you can set PV parameters as required as shown below:



CreatePersistentVolume

Creation Method: Manual | Auto

Name: cfs-pv

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Provisioner: Cloud Block Storage | Cloud File Storage | COS

R/W permission: Single machine read and write | Multi-machine read only | Multi-computer read and write

StorageClass: Do not specify | Specify

The PersistentVolume statically created will have a StorageClass of the specified type.

StorageClass: cfs-storageclass

Select CFS: cfs-test | [redacted]

If the current CFS instance is not suitable, please go to CFS Console to create a new one.

CFS Sub-directory: /subfolder

Please ensure that this sub-directory exists in CFS.

- **Creation Method**: select **Manual**.
- **Name**: set a custom name. This document uses `cfs-pv` as an example.
- **Provisioner**: select **Cloud File Storage**.
- **R/W permission**: CFS only supports multi-server read and write.
- **StorageClass**: select a StorageClass as required. This document uses `cfs-storageclass` , which you created in the step of Creating a StorageClass via the console, as an example.

> Note：
>
> - The PVC and PV will be bound to the same StorageClass.
> - If you do not specify a StorageClass, the value of `StorageClass` for the corresponding PV is empty, and the value of the `storageClassName` field in the corresponding YAML file is a null string.

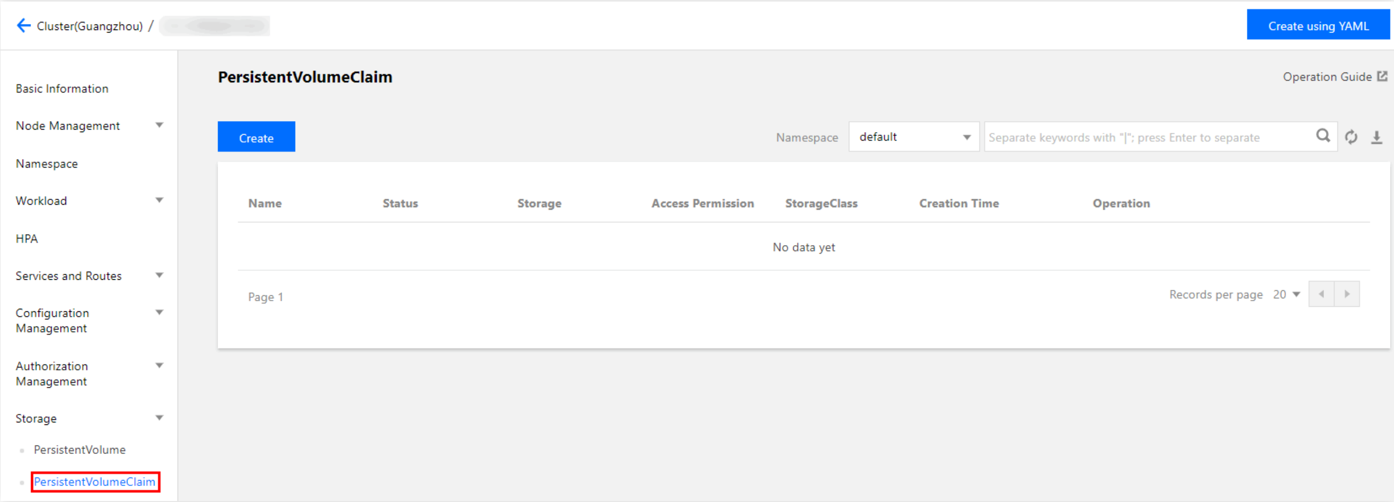
- **Select CFS**: ensure that the CFS and the current cluster are in the same VPC. This document uses `cfs-test` , which you created in the step of Creating CFS, as an example.

- **CFS Subfolder**: enter the file system subdirectory that you obtained in the step of Obtaining the file system subdirectory. This document uses `/subfolder` as an example.

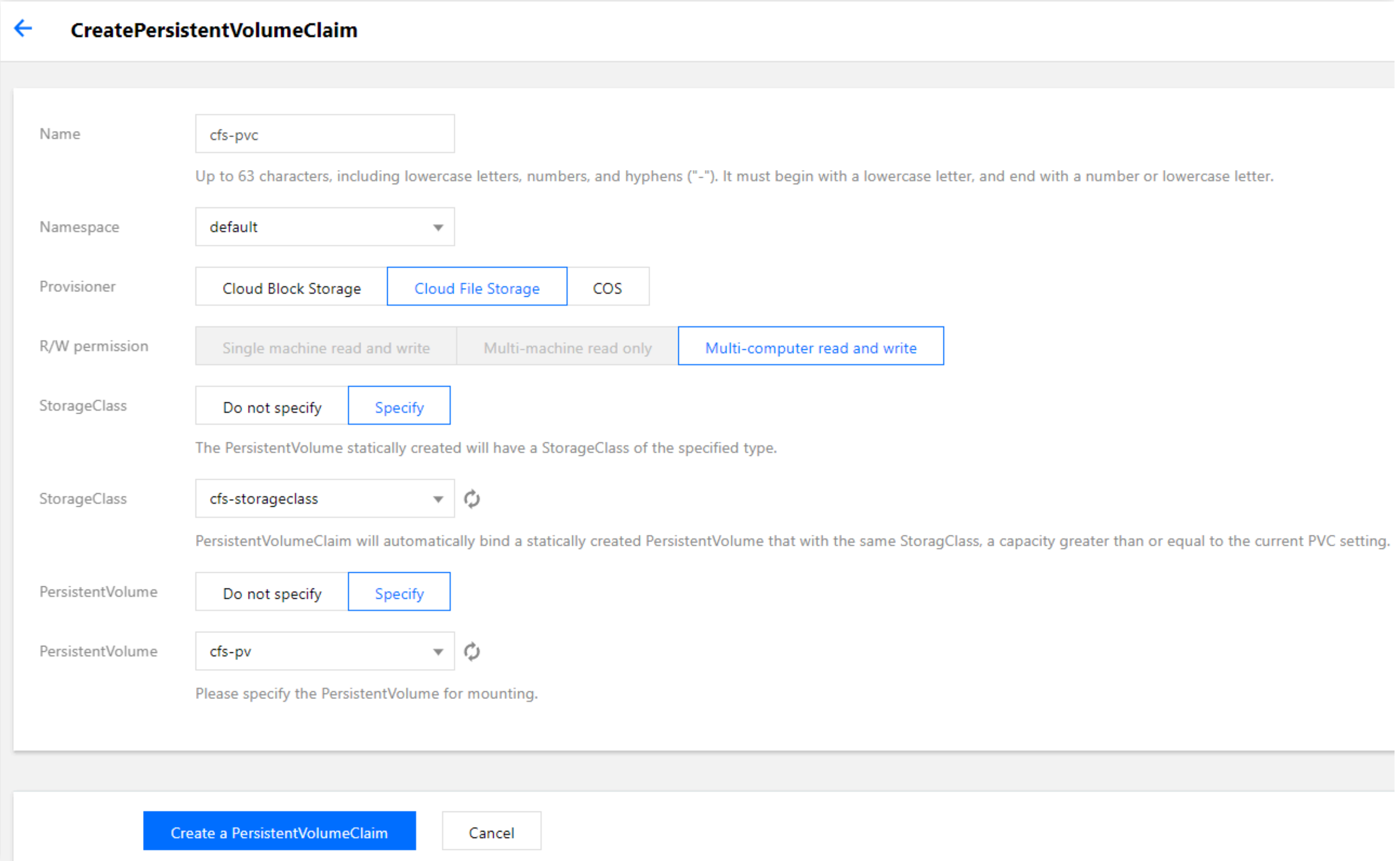
5. Click **Create PersistentVolume** to complete the creation.

Creating a PVC

1. On the target cluster details page, choose **Storage** > **PersistentVolumeClaim** in the left sidebar to go to the **PersistentVolumeClaim** page, as shown in the following figure.



2. Click **Create** to go to the **Create PersistentVolumeClaim** page, where you can set key PVC parameters as required, as shown in the following figure.



CreatePersistentVolumeClaim

Name: cfs-pvc

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Namespace: default

Provisioner: Cloud Block Storage | Cloud File Storage | COS

R/W permission: Single machine read and write | Multi-machine read only | Multi-computer read and write

StorageClass: Do not specify | Specify

The PersistentVolume statically created will have a StorageClass of the specified type.

StorageClass: cfs-storageclass

PersistentVolumeClaim will automatically bind a statically created PersistentVolume that with the same StoragClass, a capacity greater than or equal to the current PVC setting.

PersistentVolume: Do not specify | Specify

PersistentVolume: cfs-pv

Please specify the PersistentVolume for mounting.

- **Name**: set a custom name. This document uses `cfs-pvc` as an example.
- **Namespace**: select **default**.
- **Provisioner**: select **Cloud File Storage**.
- **R/W permission**: CFS only supports multi-server read and write.
- **StorageClass**: select a StorageClass as required. This document uses `cfs-storageclass` , which you created in the step of Creating a StorageClass via the console, as an example.

> Note：
>
> - The PVC and PV will be bound to the same StorageClass.
> - If you do not specify a StorageClass, the value of `StorageClass` for the corresponding PVC is empty, and the value of the `storageClassName` field in the corresponding YAML file is a null string.

- **PersistVolume**: specify a PersistentVolume as required. This document uses the `cfs-pv` created in the Creating a PV statically step as an example.

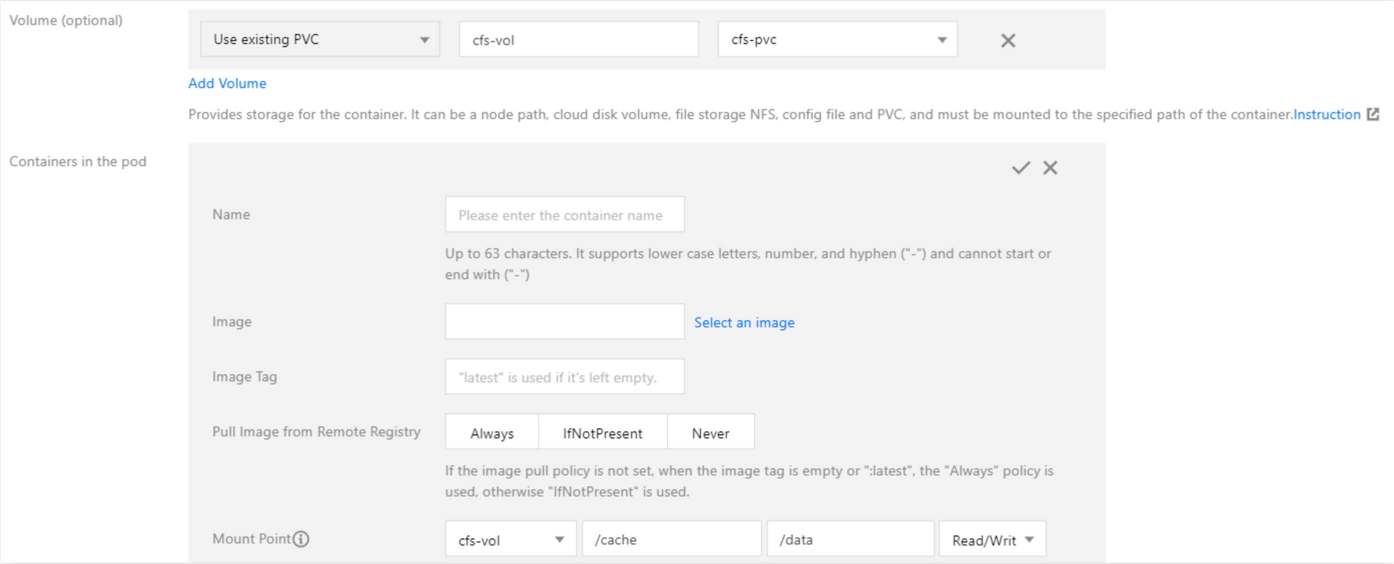
> Note：
>
> - Only PVs in the specified StorageClass and in the Available or Released statuses can be selected. If no PV in the current cluster meets the conditions, select **Do not specify** in **Specify PersistVolume**.
> - If the status of the selected PV is Released, you need to manually delete the `claimRef` field in the corresponding YAML configuration file of the PV so that the PV can be successfully bound with the PVC. For more information, see PV and PVC Binding Rules.

3. Click **Create PersistentVolumeClaim** to complete the creation process.

Creating a workload to use a PVC volume

> Note：
>
> This step creates a Deployment workload as an example.

1. On the **Cluster Management** page, select the target cluster ID to go to the **Deployment** page of the cluster for which the workload needs to be deployed.
2. Click **Create** to go to the **Create Workload** page. For more information on how to create a workload, see Creating a Deployment. Then, mount a volume as required, as shown in the following figure.



Volume (optional): Use existing PVC | cfs-vol | cfs-pvc

Add Volume

Provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container.Instruction

Containers in the pod

Name: Please enter the container name

Up to 63 characters. It supports lower case letters, number, and hyphen ("-") and cannot start or end with ("-")

Image: Select an image

Image Tag: "latest" is used if it's left empty.

Pull Image from Remote Registry: Always | IfNotPresent | Never

If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.

Mount Point: cfs-vol | /cache | /data | Read/Writ

- **Volume (optional)**:
  - **Mount method**: select **Use existing PVC**.
  - **Volume name**: set a custom name. This document uses `cfs-vol` as an example.
  - **Select PVC**: select `cfs-pvc` , which you created in the step of Creating a PVC.

- **Containers in the Pod**: click **Add Mount Target** to set a mount target.
  - **Volume**: select the volume `cfs-vol` that you added in this step.
  - **Destination Path**: enter a destination path. This document uses `/cache` as an example.
  - **Sub-path**: mount only a sub-path or a single file in the selected volume, such as `/data` or `/test.txt` .

3. Click **Create Workload** to complete the process.

> Note：
> If you use the PVC mount method of CFS, the volume can be mounted to multiple nodes.

Kubectl operation instructions

Creating a PV

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: cfs
spec:
accessModes:
- ReadWriteMany
capacity:
storage: 10Gi
csi:
driver: com.tencent.cloud.csi.cfs
volumeAttributes:
fsid: XXXXXX
host: 192.168.XX.XX
path: /
vers: "3"
volumeHandle: cfs
persistentVolumeReclaimPolicy: Retain
storageClassName: XXX
volumeMode: Filesystem
```

Parameter	Required	Description
fsid	Yes	The file system's FSID (rather than ID), which can be viewed in the mount target information of the file system.
host	Yes	The file system's IP address, which can be viewed in the mount target information of the file system.

Parameter	Required	Description
path	Yes	A subdirectory of the file system. After mounting, the workload will not be able to access the upper-level directory of this subdirectory.
vers	Yes	The version of the protocol used by the add-on to connect to the file system. Currently supported versions are "3" and "4".

Note：
If you specify the protocol version as `vers: "3"` in the YAML file of the static PV, you also need to specify the FSID of the file system to be mounted (see Getting the file system FSID); otherwise, the mount will fail. `vers: "4"` does not require the FSID.

Use Cloud Disk CBS

CBS Instructions

Last updated：2021-11-09 10:46:26

Overview

Tencent Cloud Tencent Kubernetes Engine (TKE) allows you to use Cloud Block Storage (CBS) disks by creating PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) and mounting volumes to workloads. This document describes how to mount a CBS disk to a workload in a cluster by using the following two methods:

> Note：
> When CBS is used through PV and PVC, a cloud disk only supports the creation of one PV, and it can only be mounted by one cluster node at any time.

- Method 1: dynamically creating a CBS disk
- Method 2: using an existing CBS disk

Directions

Dynamically creating a CBS disk

To dynamically create a CBS disk, you generally need to complete the following steps:

1. Create a StorageClass of the CBS type and define a CBS template.

   > Note：
   > - TKE provides a default StorageClass named cbs, which is configured with a Premium Cloud Storage cloud disk in a randomly selected availability zone in pay-as-you-go mode.
   > - You can customize a StorageClass as required.

2. Create a PVC by using the StorageClass and further define the CBS parameters.
3. Select the created PVC when creating a workload volume and set the container mount point.
   For more information, please see Managing CBS templates by using a StorageClass.

Using an existing CBS disk

You can use an existing CBS disk as follows:

1. Use an existing CBS disk to create a PV.
2. When creating a PVC, use the same StorageClass and capacity as that for the existing PV.
3. When creating a workload, select the PVC.
   For more information, please see Managing CBS by using PVs and PVCs.

Managing CBS Templates by Using a StorageClass

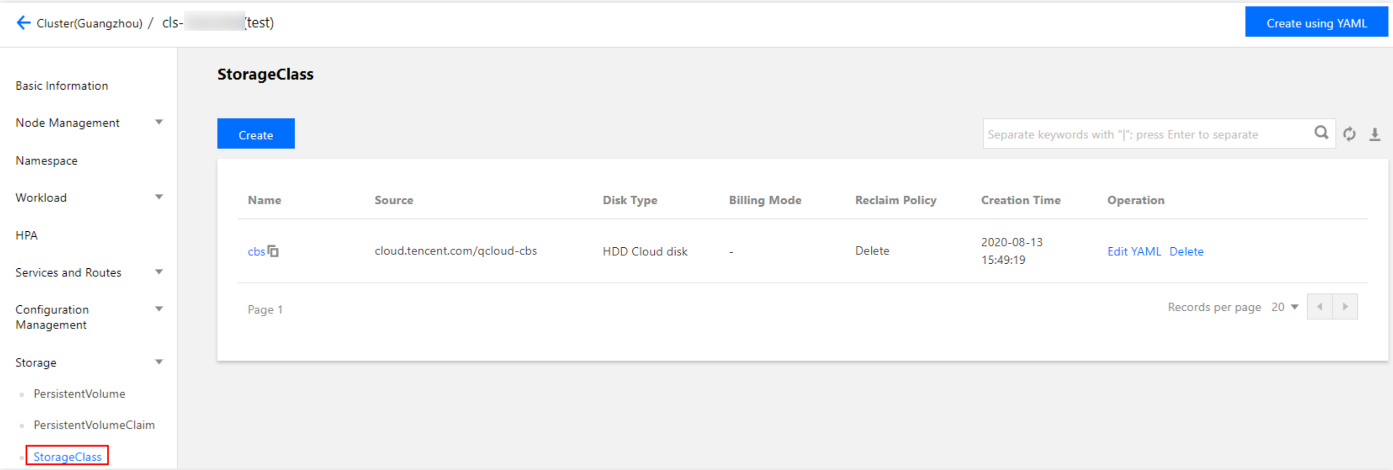
Last updated：2022-11-17 15:07:10

A cluster admin can use StorageClass to define different storage classes for Tencent Kubernetes Engine (TKE) clusters. TKE provides the block storage StorageClass by default. You can use both StorageClass and PersistentVolumeClaim to dynamically create required storage resources. This document describes how to create a StorageClass of the Cloud Block Storage (CBS) type by using the console and Kubectl, and how to customize the template required by CBS disks.

Console Operation Directions

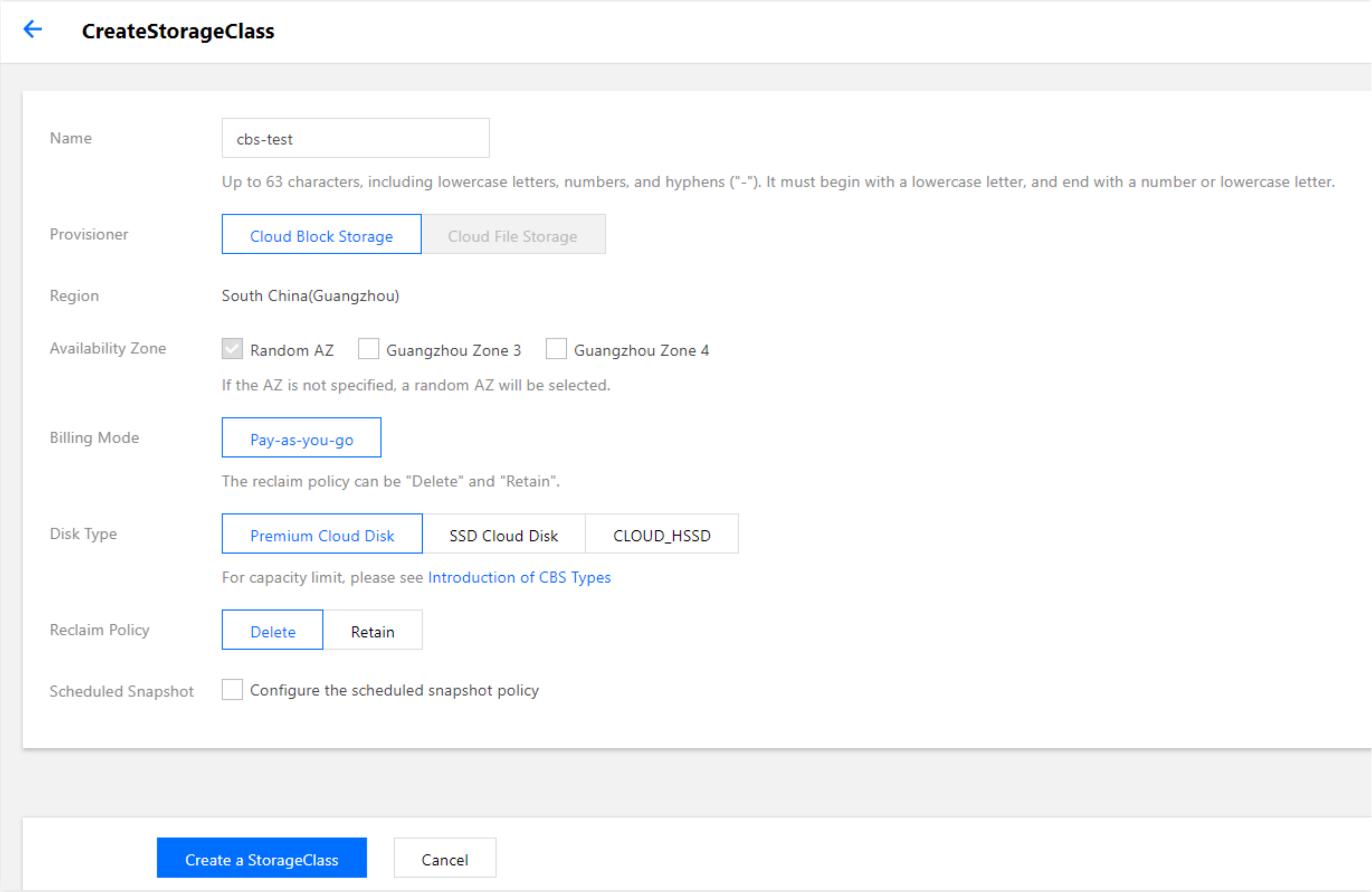
Creating StorageClass

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. Click the ID of the cluster for which a StorageClass needs to be created to go to the cluster details page.
3. Click **Storage** > **StorageClass** in the left sidebar, as shown in the following figure.



Name	Source	Disk Type	Billing Mode	Reclaim Policy	Creation Time	Operation
cbs	cloud.tencent.com/qcloud-cbs	HDD Cloud disk	-	Delete	2020-08-13 15:49:19	Edit YAML Delete

4. Click **Create** to go to the **Create StorageClass** page, where you can set the parameters as required, as shown in the following figure.



CreateStorageClass

Name: cbs-test

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Provisioner: Cloud Block Storage | Cloud File Storage

Region: South China(Guangzhou)

Availability Zone: ☑ Random AZ □ Guangzhou Zone 3 □ Guangzhou Zone 4

If the AZ is not specified, a random AZ will be selected.

Billing Mode: Pay-as-you-go

The reclaim policy can be "Delete" and "Retain".

Disk Type: Premium Cloud Disk | SSD Cloud Disk | CLOUD_HSSD

For capacity limit, please see Introduction of CBS Types

Reclaim Policy: Delete | Retain

Scheduled Snapshot: □ Configure the scheduled snapshot policy

The main parameters are described as follows:

- **Name**: Set a custom name. This document uses `cbs-test` as an example.
- **Provisioner**: Select **Cloud Block Storage**.
- **Region**: The region where the current cluster is located.
- **Availability zone**: Select the availability zones that support CBS disks in the current region as required.
- **Billing mode**: The **pay-as-you-go** billing mode is provided. It allows you to enable and terminate instances at any time. The instances are billed based on actual usage, and the **Delete** and **Retain** reclaim policies are supported.
- **Disk type**: **Premium Cloud Disk**, **SSD Cloud Disk**, and **Enhanced SSD Cloud Disk** are supported. Different availability zones may have different disk types. For more information, see Cloud Disk Types. Select a disk type as prompted by the console.
- **Reclaim policy**: The reclaim policy for cloud disks. Generally, the **Delete** and **Retain** reclaim policies are provided, which depends on the selected billing mode. For data security, we recommend that you select **Retain**.
- **Volume binding mode**: The modes of **Bind now** and **Wait for scheduling** are available. Different modes support different volume binding policies. Refer to the following information to select the appropriate mode:
  - **Bind now**: PVCs created via the storageclass will be directly bound with the PV and allocated.
  - **Wait for scheduling**: PVCs created via the storageclass will not be bound with the PV and allocated until the pod that uses the PVCs is created.

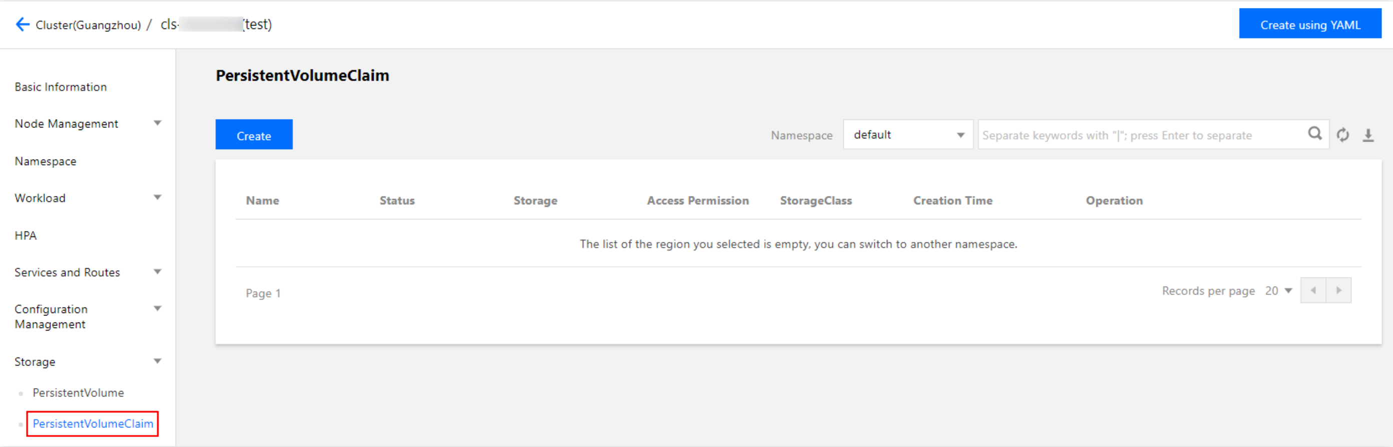
- **Scheduled snapshot**: Setting scheduled snapshot policy can effectively protect data security, but data backup will generate certain fees. For more information, see Snapshot Overview.

> Note：
> The default-policy configuration provided by TKE for backup includes the date of backup execution, time point of backup execution, and backup retention period.

5. Click **Create a StorageClass** to complete the process.

Creating a PVC by using a specified StorageClass

1. On the **Cluster management** page, select the ID of the cluster for which a PVC needs to be created.
2. On the cluster details page, choose **Storage** > **PersistentVolumeClaim** in the left sidebar to go to the **PersistentVolumeClaim** page, as shown in the following figure.



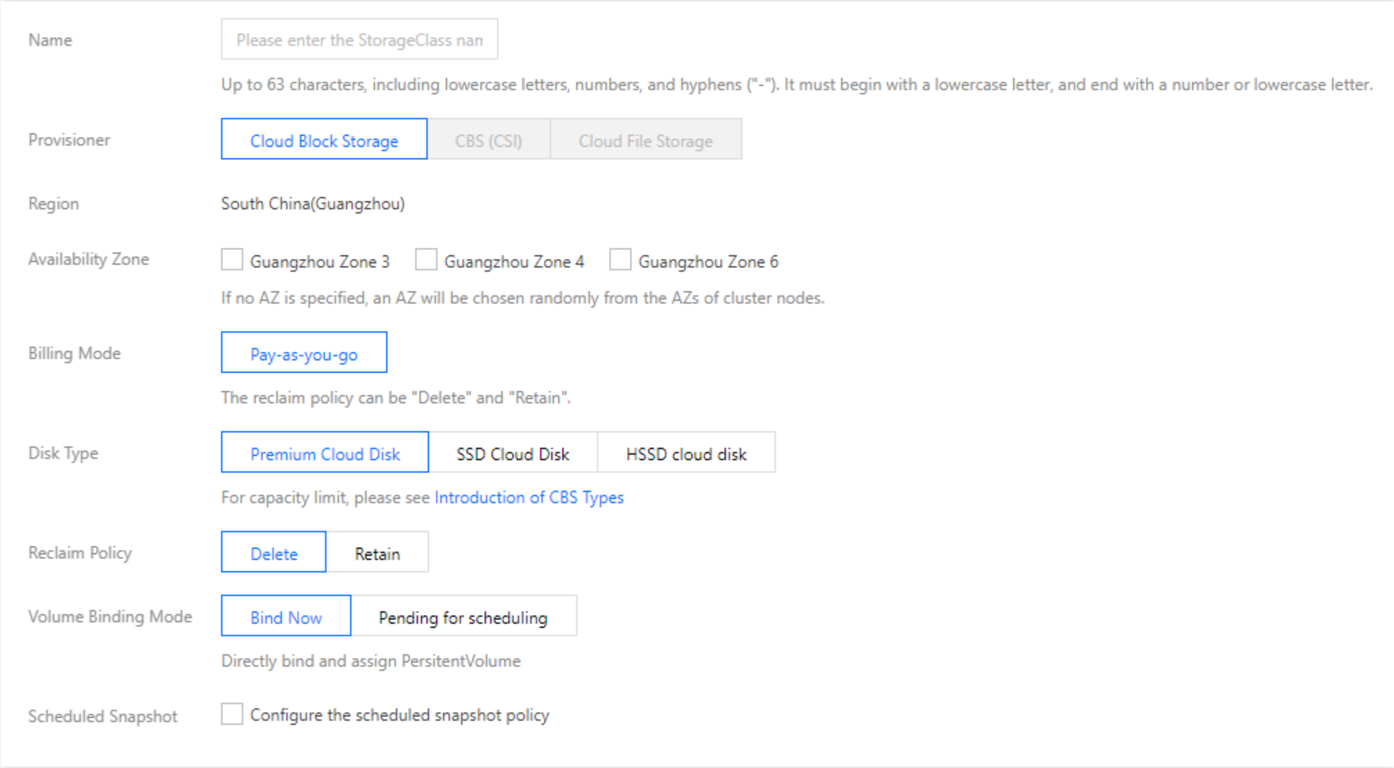
PersistentVolumeClaim

Name	Status	Storage	Access Permission	StorageClass	Creation Time	Operation

The list of the region you selected is empty, you can switch to another namespace.

Page 1   Records per page 20

3. Click **Create** to go to the **Create a PersistentVolumeClaim** page, where you can set key PVC parameters as required, as shown in the following figure.



Name	Please enter the StorageClass nan Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.
Provisioner	Cloud Block Storage / CBS (CSI) / Cloud File Storage
Region	South China(Guangzhou)
Availability Zone	□ Guangzhou Zone 3 □ Guangzhou Zone 4 □ Guangzhou Zone 6 If no AZ is specified, an AZ will be chosen randomly from the AZs of cluster nodes.
Billing Mode	Pay-as-you-go The reclaim policy can be "Delete" and "Retain".
Disk Type	Premium Cloud Disk / SSD Cloud Disk / HSSD cloud disk For capacity limit, please see Introduction of CBS Types
Reclaim Policy	Delete / Retain
Volume Binding Mode	Bind Now / Pending for scheduling Directly bind and assign PersitentVolume
Scheduled Snapshot	□ Configure the scheduled snapshot policy

The main parameters are described as follows:

- **Name**: set a custom name. This document uses `cbs-pvc` as an example.
- **Namespace**: Select **default**.
- **Provisioner**: Select **Cloud Block Storage**.
- **R/W permission**: CBS disks only support **Single machine read and write**.
- **StorageClass**: Specify a StorageClass as required. This document uses the `cbs-test` created in the step of Creating a StorageClass as an example.

> Note：
>
> - The PVC and PV will be bound to the same StorageClass.
> - If you do not specify a StorageClass, the value of `StorageClass` for the corresponding PVC is empty, and the value of the `storageClassName` field in the corresponding YAML file is a null string.

- **PersistVolume**: Specify a PersistentVolume as required. In the example in this document, no PersistentVolume is specified.

> Note：

> - The system first searches the current cluster to see whether there are PVs that meet the binding rules. If no, the system dynamically creates a PV to be bound based on the PVC and the selected StorageClass.
> - If `StorageClass` is not specified, then `PersistVolume` must be specified.
> - No PersistentVolume is specified. For more information, see PV and PVC Binding Rules.

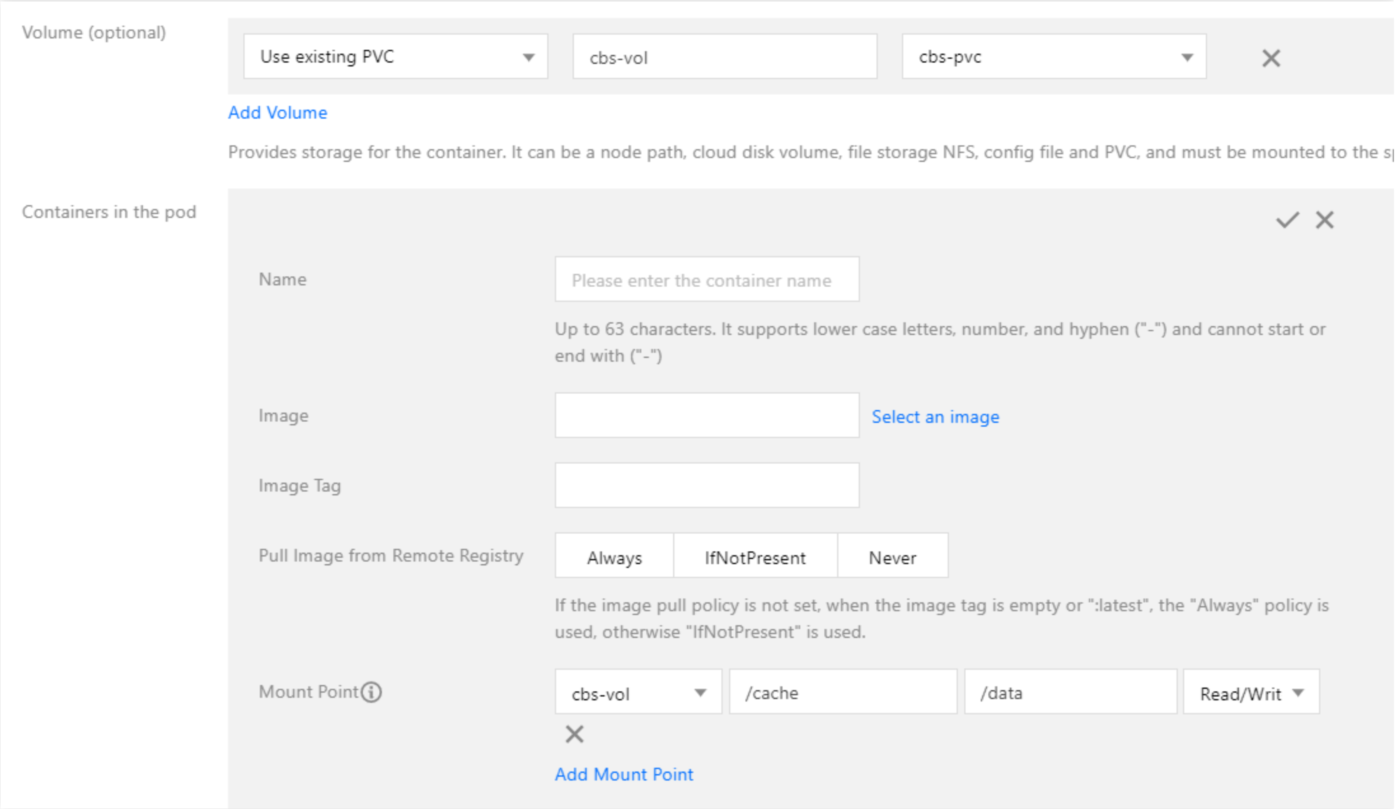
- **Disk type**: Based on the selected StorageClass, the available disk types are displayed: **Premium Cloud Disk**, **SSD Cloud Disk** and **Enhanced SSD Cloud Disk**.
- **Capacity**: When PersistentVolume is not specified, you need to indicate the desired capacity of the cloud disk. The capacity must be a multiple of 10. For premium cloud disk, the minimum capacity is 10 GB, and for SSD cloud disk and enhance SSD cloud disk, the minimum capacity is 20 GB.
- **Cost**: Based on the above parameters, calculate the cost of the corresponding cloud disk. For more information, see Billing Modes.

4. Click **Create a PersistentVolumeClaim** to complete the creation.

Creating a StatefulSet to mount a PVC volume

> Note：
>
> This step creates a StatefulSet workload as an example.

1. On the details page of the desired cluster, choose **Workload** > **StatefulSet** in the left sidebar to go to the **StatefulSet** page.
2. Click **Create** to go to the **Create Workload** page. For more information, see Creating a StatefulSet. Then, mount a volume as required, as shown in the following figure.



Volume (optional): Use existing PVC | cbs-vol | cbs-pvc

Add Volume

Provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the s

Containers in the pod

Name: Please enter the container name

Up to 63 characters. It supports lower case letters, number, and hyphen ("-") and cannot start or end with ("-")

Image: Select an image

Image Tag

Pull Image from Remote Registry: Always | IfNotPresent | Never

If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.

Mount Point: cbs-vol | /cache | /data | Read/Writ

Add Mount Point

- **Volume (optional)**:
  - **Mount method**: Select **Use existing PVC**.
  - **Volume name**: Set a custom name. This document uses `cbs-vol` as an example.
  - **Select PVC**: Select an existing PVC. This document uses the `cbs-pvc` , which you created in the step of Creating a PVC by using a specified StorageClass, as an example.
- **Containers in the Pod**: Click **Add mount target** to set a mount target.
  - **Volume**: Select the volume `cbs-vol` that you added in this step.
  - **Destination path**: Enter a destination path. This document uses `/cache` as an example.
  - **Sub-path**: Mount only a sub-path or a single file in the selected volume, such as `/data` or `/test.txt` .

3. Click **Create Workload** to complete the process.

> Note：
>
> If you use the PVC mount method of CBS, the volume can be mounted to only one node.

Kubectl Operation Directions

You can use the sample template in this document to create a StorageClass by using Kubectl.

Creating a StorageClass

The following sample YAML file is used to create a StorageClass with the default name of cbs in a cluster.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
# annotations:
# storageclass.beta.kubernetes.io/is-default-class: "true"
# If this line is present, it will become the default-class, and if you do not specify a type when creating a PVC, this type will be used automatically.
name: cloud-premium
# If the CBS-CSI add-on is installed for the TKE cluster, enter `com.tencent.cloud.csi.cbs` for `provisioner`.
# If the CBS-CSI add-on is not installed, enter `cloud.tencent.com/qcloud-cbs` for `provisioner` (this capability is deprecated in v1.20 and later versions).
provisioner: com.tencent.cloud.csi.cbs
parameters:
type: CLOUD_PREMIUM
renewflag: NOTIFY_AND_AUTO_RENEW
paymode: POSTPAID_BY_HOUR
aspid: asp-123
reclaimPolicy: Retain
volumeBindingMode: WaitForFirstConsumer
```

The following table lists the supported parameters.

Parameter	Description
type	This includes CLOUD_PREMIUM (Premium cloud disk), CLOUD_SSD (SSD cloud disk) and CLOUD_HSSD (enhanced SSD cloud disk).
zone	Availability zone. If an availability zone is specified, the cloud disk is created in this availability zone. If no availability zone is specified, the availability zones of all nodes are obtained and one is selected at random. For the identifiers of all Tencent Cloud regions, see Regions and Availability Zones.
paymode	The billing method of the cloud disk. The default value is `POSTPAID_BY_HOUR` (pay-as-you-go), which supports the **Retain** and **Delete** reclaim policies. **Retain** is only available in clusters later than V1.8.
volumeBindingMode	The volume binding mode. Two modes are supported: **Immediate** (bind now) and **WaitForFirstConsumer** (wait for scheduling).
reclaimPolicy	The reclaim policy. Two policies are supported: **Delete** and **Retain**.
renewflag	CBS renewl mode. The default value is `NOTIFY_AND_MANUAL_RENEW` .

	• `NOTIFY_AND_AUTO_RENEW` indicates that the created CBS supports notifications upon expiration and automatic renewal by month. • `NOTIFY_AND_MANUAL_RENEW` indicates that the created CBS supports notifications upon expiration but not automatic renewal. • `DISABLE_NOTIFY_AND_MANUAL_RENEW` indicates that the created CBS does not support notifications upon expiration or automatic renewal.
aspid	Snapshot policy ID. The created cloud disk will be automatically bound with this policy. Binding failure does not affect the creation of the cloud disk.

Creating a multi-Pod StatefulSet

You can use a cloud disk to create a multi-pod StatefulSet. The sample YAML file is as follows:

> The apiVersion of the resource object varies based on the cluster Kubernetes version. You can run the command `kubectl api-versions` to view the apiVersion of the current resource object.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
name: web
spec:
selector:
matchLabels:
app: nginx
serviceName: "nginx"
replicas: 3
template:
metadata:
labels:
app: nginx
spec:
terminationGracePeriodSeconds: 10
containers:
- name: nginx
image: nginx
ports:
- containerPort: 80
name: web
volumeMounts:
- name: www
mountPath: /usr/share/nginx/html
```

```
volumeClaimTemplates: # the system automatically creates a PVC and then automatic
ally creates a PV.
- metadata:
name: www
spec:
accessModes: [ "ReadWriteOnce" ]
storageClassName: cloud-premium
resources:
requests:
storage: 10Gi
```

Managing CBS by using PVs and PVCs

Last updated：2023-05-06 19:58:58

Overview

Tencent Kubernetes Engine (TKE) allows you to create persistent volumes (PVs) and persistent volume claims (PVCs) and use existing PVCs when creating workloads and adding volumes so that you can manage Cloud Block Storage (CBS) disks by using the PVs and PVCs.

Note

CBS disks cannot be mounted across availability zones. If a pod with a CBS-type PV mounted is migrated to another availability zone, the mounting will fail.

To expand a cloud disk, you need to go to the Cloud Block Storage console. For more information, see Expanding Cloud Disks.

Directions

Console operation directions

Creating a StorageClass

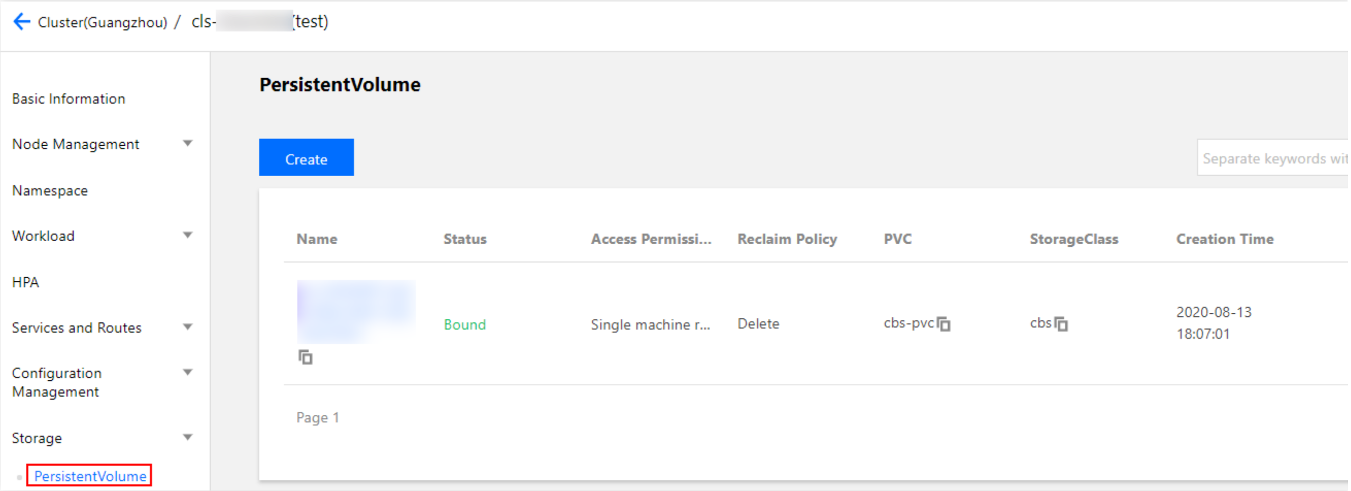
To manually create a PV of the CBS type, you need to bind an available StorageClass of the same type. For more information, see Creating StorageClass.

Creating a PV manually

Note

This approach is applicable to scenarios where there are already existing cloud disks used in the cluster.

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. Choose **Storage** > **PersistentVolume** in the left sidebar to go to the **PersistentVolume** page, as shown in the following figure.



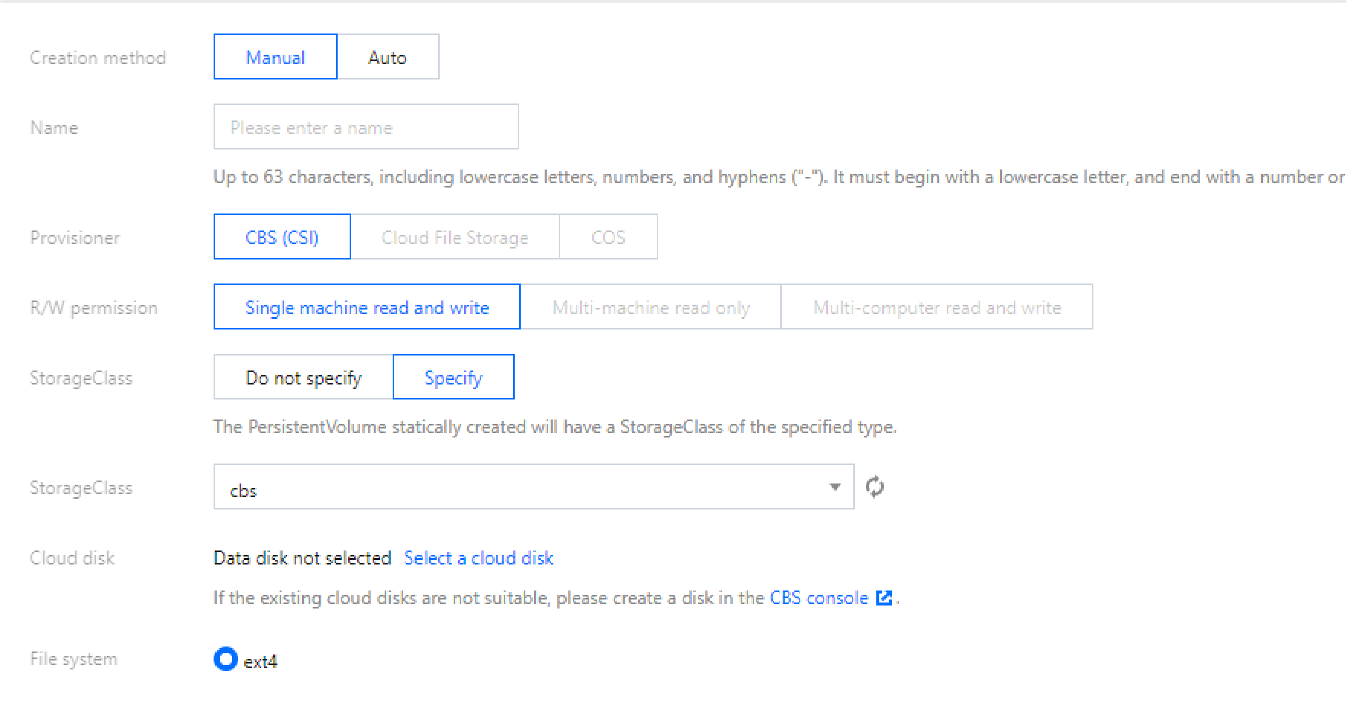
Cluster(Guangzhou) / cls-[redacted](test)

PersistentVolume

Name	Status	Access Permissi...	Reclaim Policy	PVC	StorageClass	Creation Time
[redacted]	Bound	Single machine r...	Delete	cbs-pvc	cbs	2020-08-13 18:07:01

Page 1

4. Click **Create** to go to the **Create PersistentVolume** page, where you can set PV parameters as required, as shown in the following figure.



Creation method: ☑ Manual □ Auto

Name: Please enter a name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or

Provisioner: ☑ CBS (CSI) □ Cloud File Storage □ COS

R/W permission: ☑ Single machine read and write □ Multi-machine read only □ Multi-computer read and write

StorageClass: □ Do not specify ☑ Specify

The PersistentVolume statically created will have a StorageClass of the specified type.

StorageClass: cbs

Cloud disk: Data disk not selected Select a cloud disk

If the existing cloud disks are not suitable, please create a disk in the CBS console.

File system: ● ext4

The main parameters are described as follows:

Creation Method: Select **Manual**.

Name: Set a custom name. This document uses `cbs-pv` as an example.

Provisioner: Select **Cloud Block Storage**.

R/W permission: CBS disks only support **Single machine read and write**.

StorageClass: Select a StorageClass as required. This document uses `cbs-test` , which you created in the step of Creating a StorageClass, as an example.

Note

The PVC and PV will be bound to the same StorageClass.

If you do not specify a StorageClass, the value of `StorageClass` for the corresponding PV is empty, and the value of the `storageClassName` field in the corresponding YAML file is a null string.

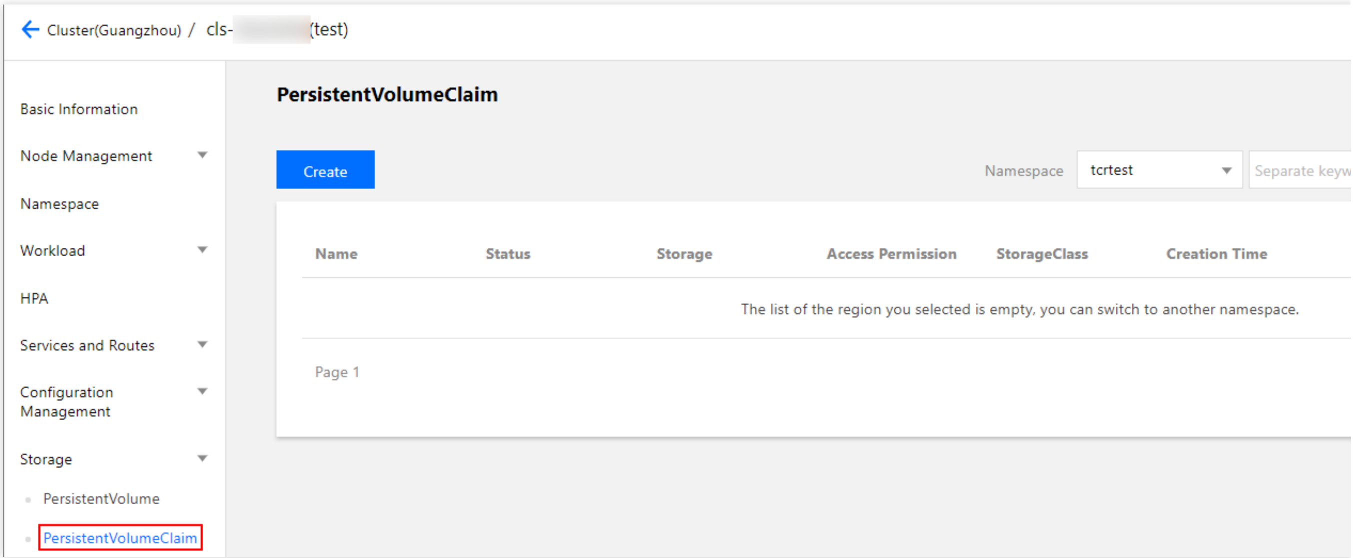
Cloud Disk: Select a created cloud disk.

File System: The default value is **ext4**.

5. Click **Create a PersistentVolume** to complete the creation.

Creating a PVC

1. On the cluster details page, choose **Storage** > **PersistentVolumeClaim** in the left sidebar to go to the **PersistentVolumeClaim** page, as shown in the following figure.

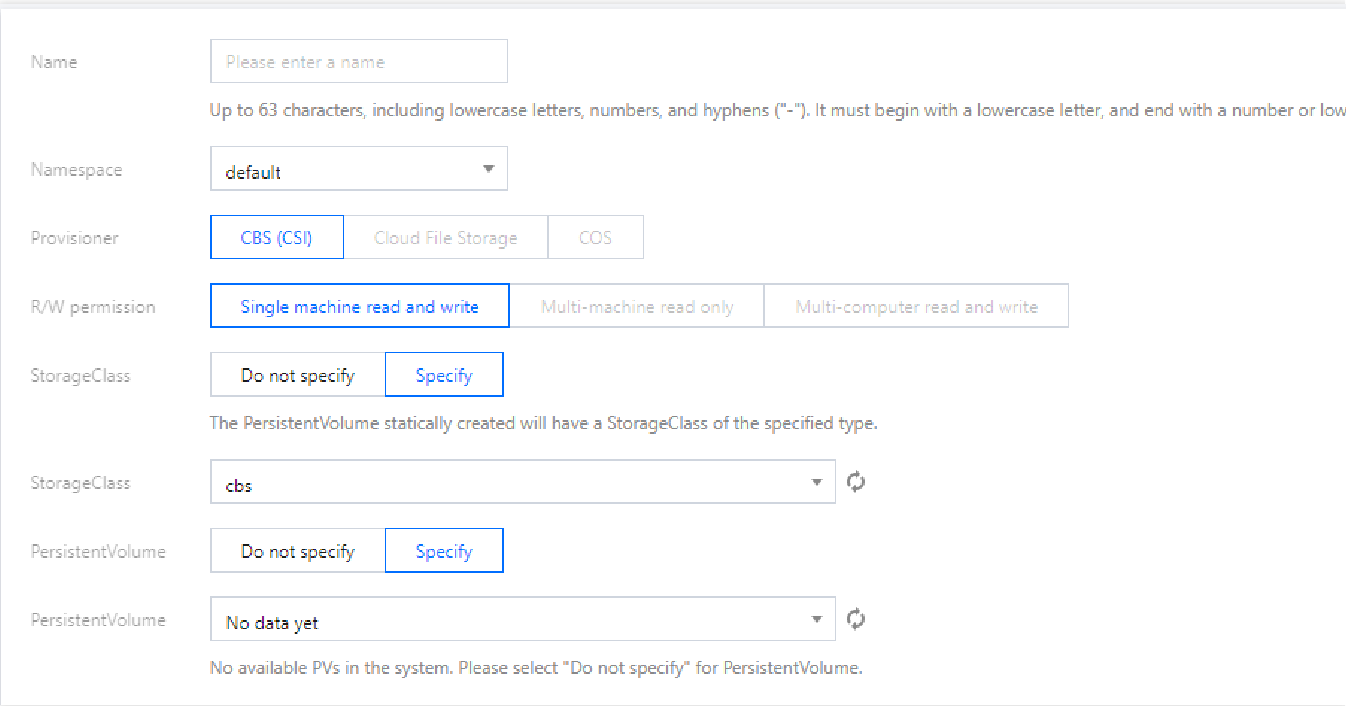


Name	Status	Storage	Access Permission	StorageClass	Creation Time

The list of the region you selected is empty, you can switch to another namespace.

Page 1

2. Click **Create** to go to the **Create PersistentVolumeClaim** page, where you can set PVC parameters as required, as shown in the following figure.



Name: Please enter a name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or low

Namespace: default

Provisioner: CBS (CSI) | Cloud File Storage | COS

R/W permission: Single machine read and write | Multi-machine read only | Multi-computer read and write

StorageClass: Do not specify | Specify

The PersistentVolume statically created will have a StorageClass of the specified type.

StorageClass: cbs

PersistentVolume: Do not specify | Specify

PersistentVolume: No data yet

No available PVs in the system. Please select "Do not specify" for PersistentVolume.

The main parameters are described as follows:

Name: Set a custom name. This document uses `cbs-pvc` as an example.

Namespace: Select **default**.

Provisioner: Select **Cloud Block Storage**.

R/W permission: CBS disks only support **Single machine read and write**.

StorageClass: Select a StorageClass as required. This document uses `cbs-test` , which you created in the step of Creating a StorageClass, as an example.

Note

The PVC and PV will be bound to the same StorageClass.

If you do not specify a StorageClass, the value of `StorageClass` for the corresponding PVC is empty, and the value of the `storageClassName` field in the corresponding YAML file is a null string.

PersistVolume: specify a PersistentVolume as required. This document uses use the `cbs-pv` created in the step of Creating a PV manually as an example.

Note

Only PVs in the specified StorageClass and in the Available or Released statuses can be selected. If no PV in the current cluster meets the conditions, select **Do not specify** in **Specify PersistVolume**.

If the status of the selected PV is Released, you need to manually delete the `claimRef` field in the corresponding YAML configuration file of the PV so that the PV can be successfully bound with the PVC. For more information, see Rules for Binding PVs and PVCs.

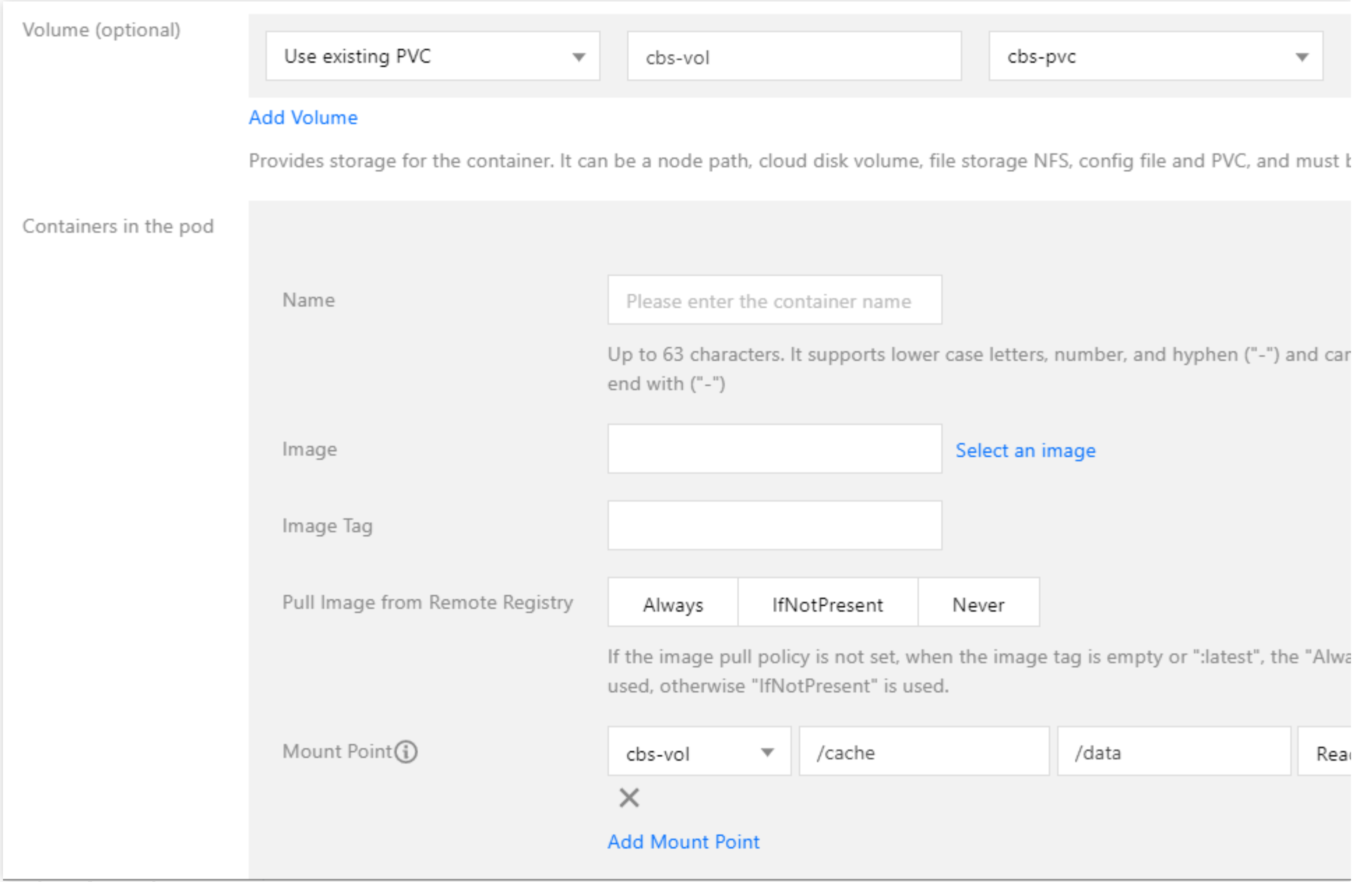
3. Click **Create a PersistentVolumeClaim** to complete the creation.

Creating a workload to use a PVC volume

Note

This step creates a Deployment workload as an example.

1. On the **Cluster Management** page, click the ID of the target cluster to go to the **Deployment** page of the cluster.
2. Click **Create** to go to the **Create Workload** page. For more information, see Creating a Deployment. Then, mount a volume as required, as shown in the following figure.



Volume (optional)

Use existing PVC ▾ | cbs-vol | cbs-pvc ▾

Add Volume

Provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must b

Containers in the pod

Name: Please enter the container name

Up to 63 characters. It supports lower case letters, number, and hyphen ("-") and car
end with ("-")

Image: Select an image

Image Tag

Pull Image from Remote Registry: Always | IfNotPresent | Never

If the image pull policy is not set, when the image tag is empty or ":latest", the "Alwa
used, otherwise "IfNotPresent" is used.

Mount Point: cbs-vol ▾ | /cache | /data | Rea

×

Add Mount Point

Volume (optional):

Mount method: Select **Use existing PVC**.

Volume name: Set a custom name. This document uses `cbs-vol` as an example.

Select PVC: Select `cbs-pvc` , which you created in the step of Creating a PVC.

Containers in the Pod: Click **Add mount point** to set a mount point.

Volume: Select the volume `cbs-vol` that you added in this step.

Target path: Enter a destination path. This document uses `/cache` as an example.

Sub-path: Mount only a sub-path or a single file in the selected volume, such as `/data` or `/test.txt` .

3. Click **Create Workload** to complete the process.

Note

If you use the PVC mount method of CBS, the volume can be mounted to only one node.

Kubectl operation directions

You can use the following sample YAML file to perform creation by using Kubectl.

(Optional) Creating a PV

You can create a PV by using an existing CBS disk, or directly create a PVC. The system automatically creates the PV. The sample YAML file is as follows:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cbs-test
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 10Gi
  csi:
    driver: com.tencent.cloud.csi.cbs
    fsType: ext4
    readOnly: false
    volumeHandle: disk-xxx # Specify an existing CBS ID
  storageClassName: cbs
```

Creating a PVC

If you did not create a PV, the system automatically creates the corresponding PV when creating a PVC. The sample YAML file is as follows:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
   name: nginx-pv-claim
spec:
   storageClassName: cbs
   accessModes:
     - ReadWriteOnce
   resources:
     requests:
       storage: 10Gi
```

The capacity of the cloud disk must be a multiple of 10.

The minimum capacity of a premium cloud disk is 10 GB, and the minimum capacity of an SSD cloud disk or enhanced SSD cloud disk is 20 GB. For details, see the Creating Cloud Disks.

Using a PVC

You can create a workload to use a PVC volume. The sample YAML file is as follows:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
   name: nginx-deployment
spec:
   replicas: 1
   selector:
     matchLabels:
       qcloud-app: nginx-deployment
   template:
     metadata:
       labels:
         qcloud-app: nginx-deployment
     spec:
       containers:
       - image: nginx
         imagePullPolicy: Always
         name: nginx
         volumeMounts:
         - mountPath: "/opt/"
           name: pvc-test
       volumes:
       - name: pvc-test
         persistentVolumeClaim:
           claimName: nginx-pv-claim # An existing PVC
```

Instructions for Other Storage Volumes

Last updated：2022-04-18 14:56:06

Introduction

Volume types

Volume Type	Description
Use temporary path	/
Use host path	Mount the file directory of the host where the container resides to the path specified by the container (corresponding to HostPath in Kubernetes). You can also choose not to set the source path (corresponding to EmptyDir in Kubernetes). If the source path is not specified, the system mounts the temporary directory of the assigned host to the mount target of the container.**Local disk volumes that have specified source paths are suitable for persisting data to the host where the container resides, whereas EmptyDir is suitable for temporary storage for containers.**
Use NFS disk	Simply enter the NFS path. You can use Tencent Cloud's Cloud File Storage (CFS) or user-built NFS file storage. **An NFS volume is suitable for persistent storage with frequent reads and writes in scenarios such as big data analysis, media processing, and content management.**
Use existing PersistentVolumeClaim	Use the storage of the existing PersistentVolumeClaim to declare the storage for workloads, and automatically assign or create a PersistentVolume and mount it to the corresponding pod. This is suitable for stateful applications created by StatefulSet.
Use ConfigMap	A ConfigMap is mounted to a pod as a file system. You can mount the custom ConfigMap entries to a specific path. For more information, see ConfigMap Management.
Use Secret	A Secret is mounted to a pod as a file system. You can mount custom Secret entries to a specific path. For more information, see Secret Management.

Notes on volumes

- **After creating a volume, you need to set the mount target of the container in the "Containers in the Pod" module.**
- Under the same service, the name of the volume and the set mount target must be unique.
- When the source path of a local disk volume is empty, the system will assign a temporary directory `/var/lib/kubelet/pods/pod_name/volumes/kubernetes.io~empty-dir` , and the lifecycle of the

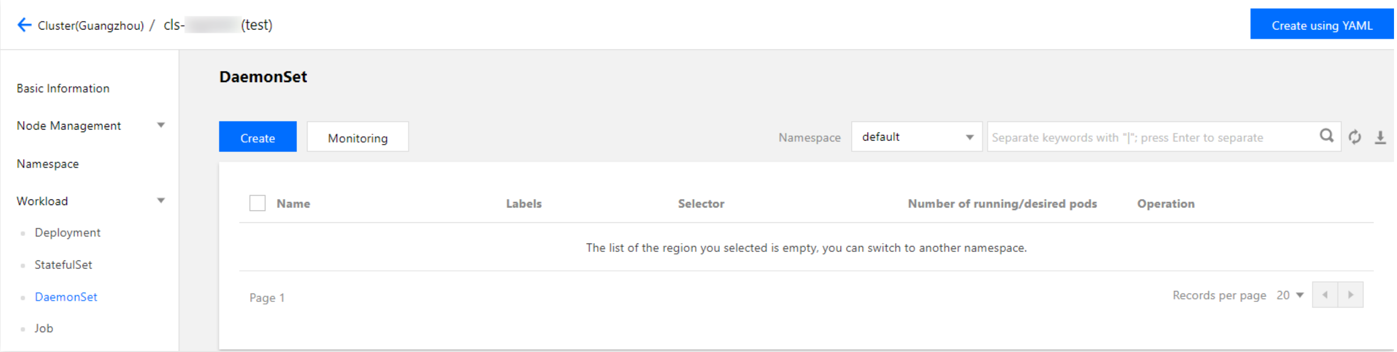
volume using the temporary directory is the same as the lifecycle of the Pod.

- If no permission is set for volume mount, the default setting will be read/write permission.

Operation Guide for Volumes in the Console

Creating a workload to mount a volume

1. Log in to the TKE console and select **Clusters** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the cluster where you want to deploy a workload to go to the cluster management page.
3. Under **Workload**, select a workload type to go to the corresponding information page.
   For example, select **Workload** -> **DaemonSet** to go to the DaemonSet page as shown in the following figure:

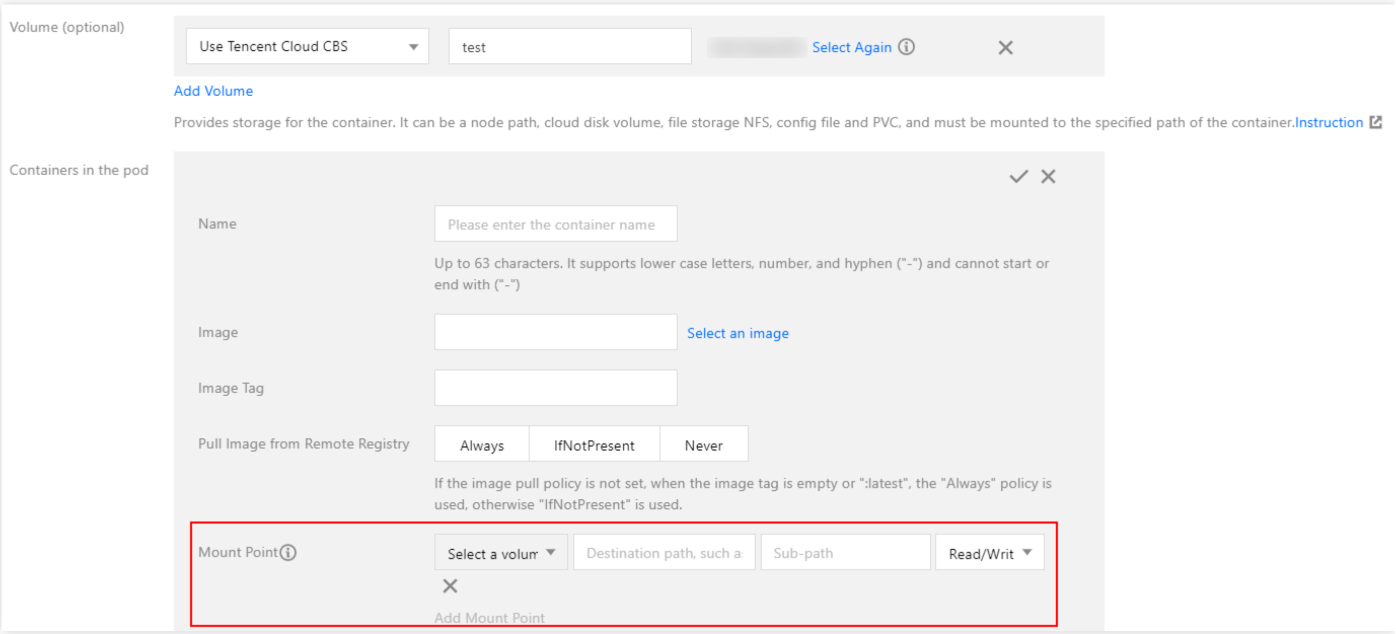


Namespace: default

Name	Labels	Selector	Number of running/desired pods	Operation

The list of the region you selected is empty, you can switch to another namespace.

4. Click **Create** to go to **Create Workload** page.
5. Set the workload name, namespace and other information as instructed. In **Volume**, click **Add Volume**.
6. Select a storage method for the volume. **Use Tencent Cloud CBS** is selected in this case.
7. Configure the mount point in **Mount Target** under **Containers in the Pod**, as shown in the following figure:
   You can configure the mount point only after selecting **Add Volume** in Step 5.



Volume (optional)

Use Tencent Cloud CBS ▾ | test | [redacted] Select Again ⓘ ✕

Add Volume

Provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container.Instruction

Containers in the pod

✓ ✕

Name: Please enter the container name

Up to 63 characters. It supports lower case letters, number, and hyphen ("-") and cannot start or end with ("-")

Image: Select an image

Image Tag

Pull Image from Remote Registry: Always | IfNotPresent | Never

If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.

Mount Pointⓘ: Select a volum ▾ | Destination path, such a | Sub-path | Read/Writ ▾ ✕

Add Mount Point

8. Set other options as needed and click **Create Workload** to finish the creation.

Configurations for mounting different volumes

This table shows the details of the use of different volumes. **When you are creating a workload and have selected **Add Volume**, you can refer to the following content to add the volume and set the mount point:

Volume			Mount Target		
Type	Name	Others	Destination Path	Sub-path	Read and Write Permissions
Temporary path	Custom	/	Specify this path as needed, for example, `/cache`.	Mount only the sub-path or a single file in the selected volume, for example, `/data` or `/test.txt`.	Select as needed. • Read-only: the container path volume can only be read and data modifications must be performed on the host. • Read and write: data

Volume			Mount Target		can be read from and
Type	Name	Others	Destination Path	Sub-path	Read and Write Permissions
Node path		Set the node path. • Node Path: the node path cannot be empty. For example, if the container needs to access Docker, the node path can be set to `/var/lib/docker` . • Check Type: TKE provides many check types such as NoChecks and DirectoryOrCreate. Read the description of each type in the console and select a check type as needed.			container path volume.
NFS disk		NFS Path: enter the CFS address or user-built NFS address. • To create a file system, see Creating File Systems and Mount Targets. • `10.0.0.161:/` is an example NFS path. To obtain the NFS path, please log in to the CFS console, click the ID of the target file system and find it in **Mount under Linux** on the **Mount Target Info** tab.			
Existing PVC		Choose a PVC as needed.			
Tencent Cloud CBS		Select a cloud disk as needed.			
ConfigMap		• Select a ConfigMap: select a ConfigMap as needed.			

Volume			Mount Target		
Type	Name	Others	Destination Path	Sub-path	Read and Write Permissions
SECRET		• Options: **All** and **Specific keys**. • Items: if you select **Specific keys**, you can mount it to a specific path by adding items. For example, if the mount point is `/data/config` and the sub-path is `dev` , the data will be stored under `/data/config/dev` .			

Using kubectl to Manipulate Volumes

The following is a sample where which you can perform creation by using kubectl directly.

Sample YAML for Mounting a Volume to a Pod

```
apiVersion: v1
kind: Pod
metadata:
name: test-pd
spec:
containers:
- image: k8s.gcr.io/test-webserver
name: test-container
volumeMounts:
- mountPath: /cache
name: cache-volume
volumes:
- name: cache-volume
emptyDir: {}
```

- **spec.volumes**: set the name, type, and parameters of the volume.
  - **spec.volumes.emptyDir**: set the temporary path
  - **spec.volumes.hostPath**: set the host path
  - **spec.volumes.nfs**: set the NFS disk

  - **spec.volumes.persistentVolumeClaim**: set the existing PersistentVolumeClaim.
- **spec.volumeClaimTemplates**: if this declaration is used, PersistentVolumeClaim and PersistentVolume will be automatically created based on the content of the declaration
- **spec.containers.volumeMounts**: enter the mount point of the volume.

PV and PVC binding rules

Last updated：2022-11-10 10:26:13

PV Status Introduction

PV Status	Description
Available	When a created PV is not bound with a PVC, the PV is in the `Available` status.
Bound	After a PVC is bound with a PV, the PV is in the `Bound` status.
Released	For a PV with the Retain reclaim policy, after its bound PVC is deleted, the status of the PV will change from `Bound` to `Released`. **Note:** for a PV in the `Released` status, you need to manually delete the claimRef field in the YAML configuration file so that the PV can be successfully bound with a new PVC.

PVC Status Introduction

PVC Status	Description
Pending	When no eligible PV can be bound with a PVC, the PVC is in the `Pending` status.
Bound	After a PVC is bound with a PV, the PVC is in the `Bound` status.

Binding Rules

When binding a PVC with a PV, consider the following parameters to check whether PVs that meet the binding rules are available in the current cluster.

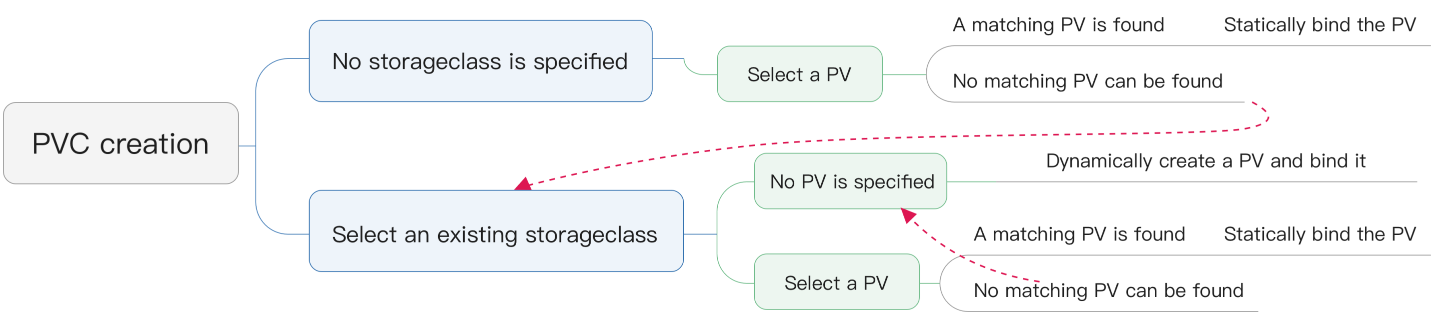
Parameter	Description
VolumeMode	Specifies whether the volume is of the `FileSystem` type or `Block` type. The PV to be selected must match the PVC in terms of the VolumeMode label.
Storageclass	The `storageclass` of the PV and PVC must be the same (or both empty).
AccessMode	Specifies the volume access mode. The AccessMode of the PV and that of the PVC must be the same.

Size	Specifies the storage capacity of the volume. The specified capacity of the PVC must be less than or equal to that of the PV. If multiple eligible PVs are available, bind the PV with the smallest capacity to the PVC.

> Note：
> After a PVC is created, the system will bind an eligible PV with the PVC based on the above parameters. If the PV resources in the current cluster are insufficient, the system will dynamically create an eligible PV and bind it with the PVC.

StorageClass Selection and PV/PVC Binding

The following figure shows the principle of StorageClass selection and PV/PVC binding on TKE platform:


PVC creation
No storageclass is specified
Select a PV
A matching PV is found
Statically bind the PV
No matching PV can be found
Select an existing storageclass
No PV is specified
Dynamically create a PV and bind it
Select a PV
A matching PV is found
Statically bind the PV
No matching PV can be found

Application and Add-On Feature Management Description

Last updated：2023-05-18 10:33:00

Add-on management description

TKE provides a variety of add-ons and rich cluster features to enhance cluster performance and overall stability. Three types of add-ons are available:

Add-On Type	Description	Documentation
System add-on	It is the default and core add-on in a cluster, such as CBS-CSI, IPAMD, monitoring add-on, and log add-on. If it is abnormal, the cluster may fail.	-
Enhanced add-on	It is also called the extended add-on, an extended feature package provided by TKE. You can select one as needed.	Add-on Overview
Application market	It provides Helm 3.0 features integrated by TKE and supports creating various products and services such as Helm charts, container images, and software services.	Application Market

Note：

Disclaimer: Tencent Cloud supports your proper installation of applications from the application market for supported cluster types and Kubernetes versions, but is not responsible for other matters such as application issues during running, application exceptions due to custom configuration modification, or lack of support for specified cluster types and Kubernetes versions.

Tencent Cloud aftersales team provides applications from the application market specifically for experienced system admins or IT personnel. Tencent Cloud doesn't provide debugging or SLAs for such applications. If you encounter any problems when using an application, contact us via the reference link provided in the application details or at the official website of the application.

For more information on the TKE SLA, see Service Level Agreement.

The three types of add-ons are managed as follows:

Add-On Type	Scope	Service Level	Upgrade Method
System add-on(Default installation that cannot be deleted)	CBS-CSI, IPAMD, monitoring add-on, log add-on, etc.	TKE prioritizes the stability of system add-ons and promptly releases security and compatibility updates and fixes on the backend.	TKE will not update certain versions with special features, which need to be manually updated as needed. For detailed directions, see Add-

			On Version Maintenance Description.
Enhanced add-on(Custom installation)	For more information on the list, see Add-on Overview.	TKE guarantees the stability of enhanced add-ons and promptly releases security and compatibility updates and fixes on the backend. Each enhanced add-on comes with a list of supported versions and may fail if you don't update it promptly.	TKE will not update certain versions with special features, which need to be manually updated as needed. For detailed directions, see Add-On Version Maintenance Description.
Application market(Custom installation)	For more information on the list, see here.	TKE only supports the installation and deployment of applications for supported cluster types and Kubernetes versions.	TKE provides application update methods and may push new charts from time to time. For detailed directions, see Use the application.

Feature management description

Certain features are identified as "preview" by TKE in the documentation and in the console, indicating that they are try-outs and are not covered by the Service Level Agreement.

Add-On Management

Add-on Overview

Last updated：2023-02-01 16:10:50

Add-ons are extended feature packages provided by Tencent Cloud TKE. You can deploy add-ons based on your business requirements. Add-ons can help you manage Kubernetes components in clusters, including component deployment, upgrades, configuration updates, and removal.

Add-on Types

There are two types of add-ons: basic add-ons and advanced add-ons.

Basic add-ons

Basic add-ons are software packages that TKE features depend on. For example, the CLB add-ons `Service-controller` and `CLB-ingress-controller`, and the TKE network add-on `tke-cni-agent`.

> Note：
>
> - The upgrade and configuration management of basic add-ons are fully managed by TKE. We recommend that you do not modify basic add-ons.
> - When basic add-ons are updated, you will be notified by email and SMS.

Advanced add-ons

Advanced add-ons are optional add-ons provided by TKE. You can deploy such add-ons to use the advanced features supported by TKE. The following table describes the advanced add-ons:

Add-On	Use Case	Description
OOMGuard (OOM daemon)	Monitoring	This add-on reduces the kernel failures caused by cgroup memory reclamation failures in user mode.
NodeProblemDetectorPlus (node exception detection plus)	Monitoring	This add-on detects various exceptions on nodes in real time and reports the detection results to kube-apiserver.
NodeLocalDNSCache (local DNS cache add-on)	DNS	This add-on runs the DNS cache proxy as a DaemonSet on the cluster node to improve the cluster DNS performance.

Add-On	Use Case	Description
DNSAutoscaler (DNS horizontal scaling add-on)	DNS	This add-on gets the numbers of nodes and cores of a cluster via a Deployment and then automatically adds or removes DNS replicas according to the preset scaling policy.
COS-CSI (COS)	Storage	This add-on implements the CSI API, which can help container clusters use COS.
CFS-CSI (CFS)	Storage	This add-on implements the CSI API, which can help container clusters use CFS.
CBS-CSI (CBS)	Storage	This add-on implements the CSI API to allow you to select the storage class for TKE clusters and create PVs and PVCs of the corresponding CBS cloud disk types in the console.
TCR (TCR plug-in)	Image	This add-on automatically configures the cluster with the domain name private network parsing and cluster-dedicated access credential of the specified TCR instance cluster. When it's enabled, the cluster can pull container images over the private network without a secret.
P2P (accelerated distribution of container images)	Image	Based on P2P technology, this add-on can be used to accelerate the pulling of GB-level container images in large-scale TKE clusters and supports concurrent pulling of thousands of nodes.
Ceberus (image signature verification add-on)	Image	This add-on performs signature verification on container images in the TCR repository to ensure that only container images signed by trusted authorizing parties are deployed, thereby mitigating the risks of running exceptions or malicious code.
Dynamic Scheduler (dynamic scheduling add-on)	Scheduling	Dynamic Scheduler is an add-on provided by TKE for pre-selection and preferential selection based on actual node loads. It is implemented based on the native Kube-scheduler Extender mechanism of Kubernetes. After being installed in a TKE cluster, this add-on will effectively prevent node load imbalances caused by the native scheduler through the request and limit scheduling mechanisms.
Descheduler (rescheduling add-on)	Scheduling	After being installed in a TKE cluster, this add-on will work with Kube-scheduler to monitor the high-load nodes in the cluster in real time and drain low-priority Pods. We recommend you use it together with the TKE Dynamic Scheduler add-on to ensure cluster load balancing in multiple dimensions.

Add-On	Use Case	Description
NetworkPolicy Controller (network policy controller add-on)	Others	Network Policy is a resource provided by Kubernetes. This add-on provides a controller for implementing resources of this type.
Nginx-Ingress (community Ingress add-on)	Others	Nginx can be used as a reverse proxy, load balancer, and for HTTP caching. Nginx-ingress is an Ingress controller for Kubernetes that uses Nginx as a reverse proxy and load balancer.
OLM (Operator Lifecycle Manager)	Others	OLM (Operator Lifecycle Manager) is part of the Operator Framework, which helps users install, update, and manage the lifecycle of Operators.
HPC (modifying the number of replicas periodically)	Others	HorizontalPodCronscaler (HPC) is an add-on to modify the number of replicas of K8s workloads periodically. Used in conjunction with HPC CRD resources, it can support scheduled actions in seconds.

Add-On Lifecycle Management

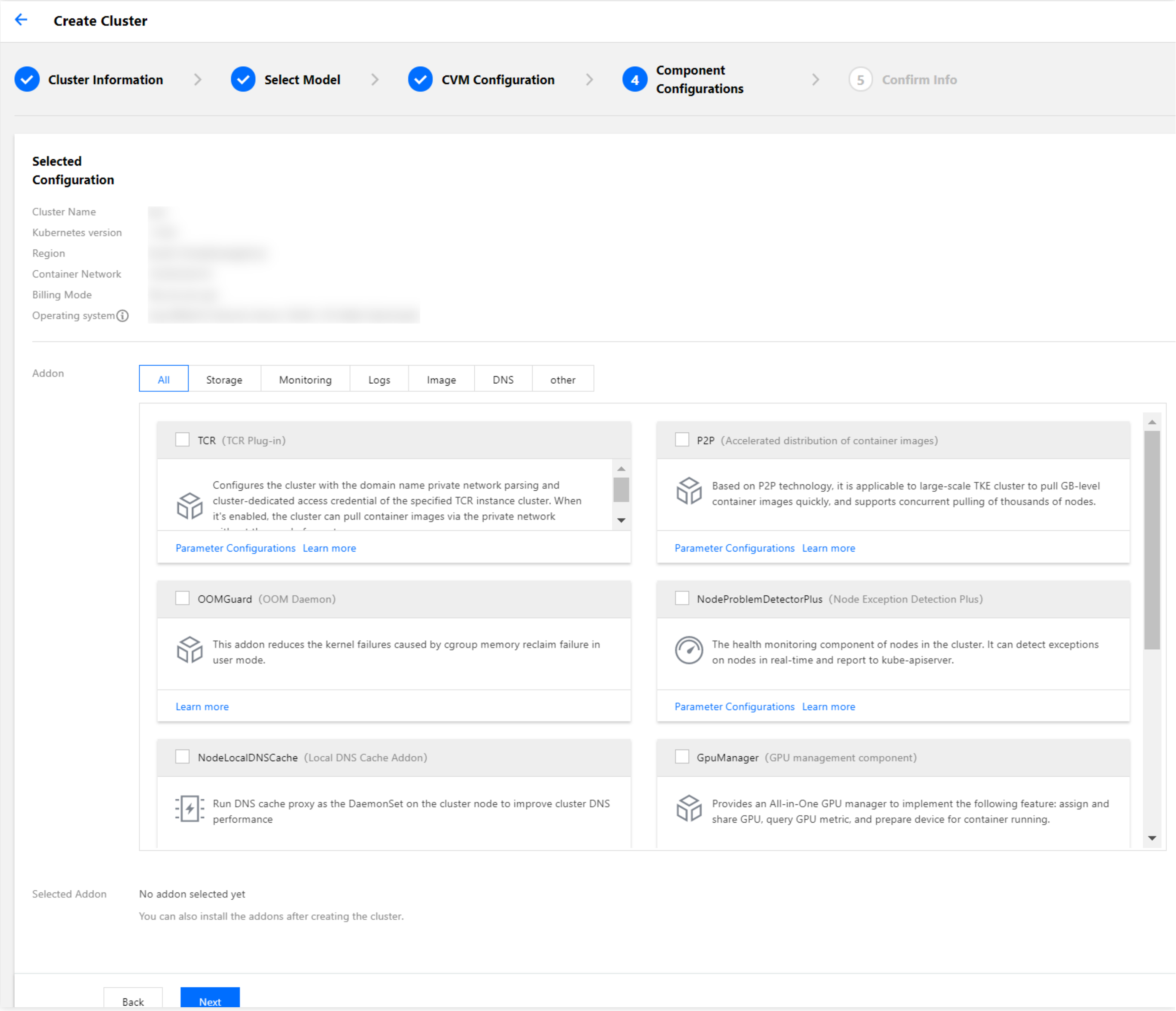
Last updated：2022-12-12 15:08:35

Installing an Add-on

You can install add-ons with the Installing on the cluster creation page or Installing on the add-on management page.

Installing on the cluster creation page

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the "Cluster management" page, click **Create** above the cluster list.
 3 On the "Create a cluster" page, configure the parameters in the **Basic information**, **Select a model**, **CVM configuration**, and **Add-on configuration** steps in sequence.



Create Cluster

Cluster Information > Select Model > CVM Configuration > 4 Component Configurations > 5 Confirm Info

Selected Configuration

Cluster Name [redacted]
Kubernetes version [redacted]
Region [redacted]
Container Network [redacted]
Billing Mode [redacted]
Operating system [redacted]

Addon: All | Storage | Monitoring | Logs | Image | DNS | other

□ TCR (TCR Plug-in)
Configures the cluster with the domain name private network parsing and cluster-dedicated access credential of the specified TCR instance cluster. When it's enabled, the cluster can pull container images via the private network
Parameter Configurations Learn more

□ P2P (Accelerated distribution of container images)
Based on P2P technology, it is applicable to large-scale TKE cluster to pull GB-level container images quickly, and supports concurrent pulling of thousands of nodes.
Parameter Configurations Learn more

□ OOMGuard (OOM Daemon)
This addon reduces the kernel failures caused by cgroup memory reclaim failure in user mode.
Learn more

□ NodeProblemDetectorPlus (Node Exception Detection Plus)
The health monitoring component of nodes in the cluster. It can detect exceptions on nodes in real-time and report to kube-apiserver.
Parameter Configurations Learn more

□ NodeLocalDNSCache (Local DNS Cache Addon)
Run DNS cache proxy as the DaemonSet on the cluster node to improve cluster DNS performance

□ GpuManager (GPU management component)
Provides an All-in-One GPU manager to implement the following feature: assign and share GPU, query GPU metric, and prepare device for container running.

Selected Addon: No addon selected yet
You can also install the addons after creating the cluster.

You can select the add-ons to install based on your business deployment and click **View details** to view the add-on description. Some add-ons require you to complete **Parameter configuration** first.

Note：

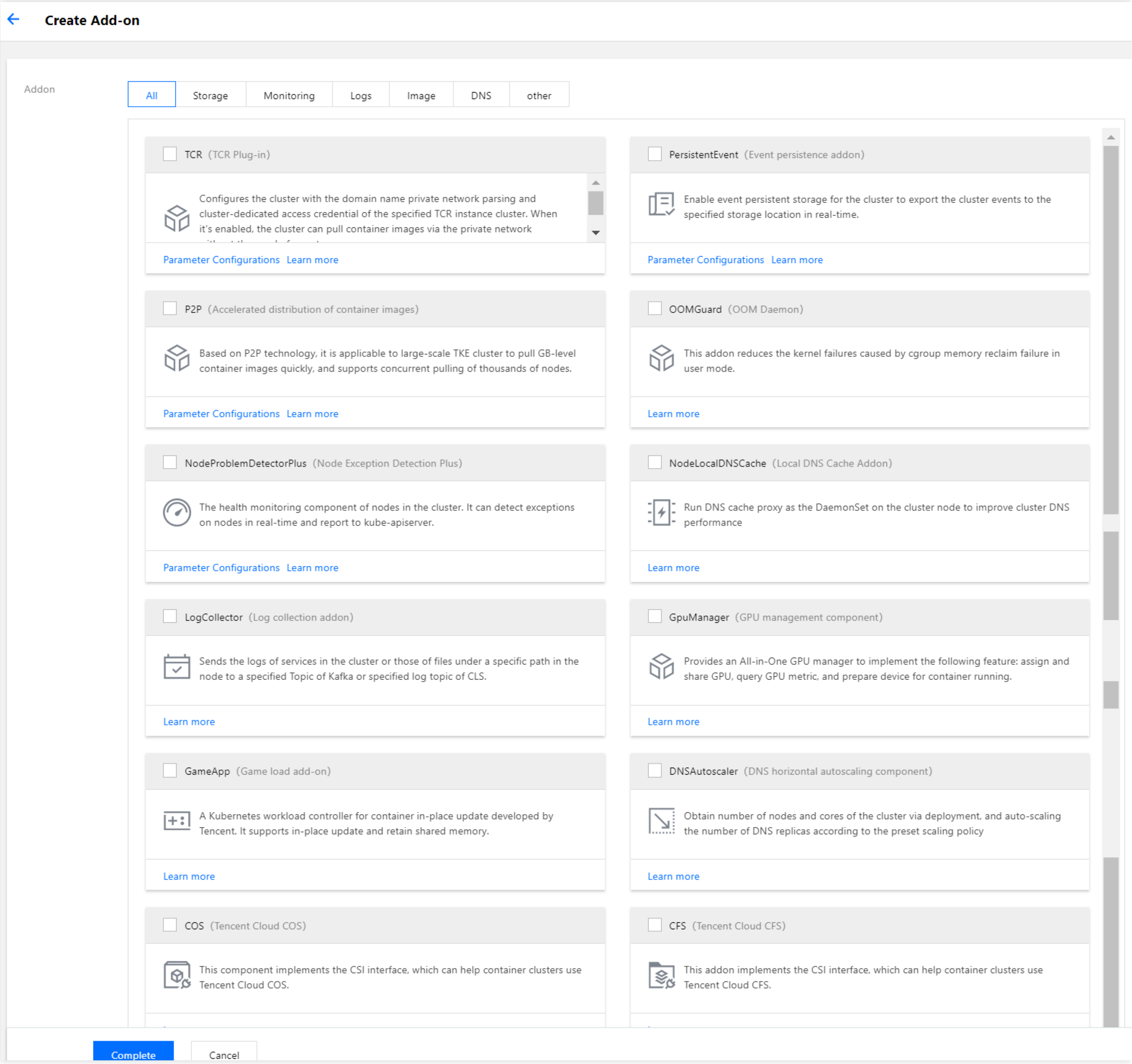
- Add-on installation is not a critical path in cluster creation. An add-on installation failure does not affect the cluster creation.
- Add-on installation consumes some cluster resources. The specific amount of resources consumed varies depending on the add-ons. For more information, click **View details** for a specific add-on.

3. Click **Next**. Check and confirm the cluster configuration information.

4. Click **OK** to complete the process.

Installing on the add-on management page

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on management** to go to the “Add-on list” page.
4. On the "Add-on list" page, click **Create** to go to the add-on installation page.



□ TCR (TCR Plug-in)

Configures the cluster with the domain name private network parsing and cluster-dedicated access credential of the specified TCR instance cluster. When it's enabled, the cluster can pull container images via the private network

Parameter Configurations Learn more

□ PersistentEvent (Event persistence addon)

Enable event persistent storage for the cluster to export the cluster events to the specified storage location in real-time.

Parameter Configurations Learn more

□ P2P (Accelerated distribution of container images)

Based on P2P technology, it is applicable to large-scale TKE cluster to pull GB-level container images quickly, and supports concurrent pulling of thousands of nodes.

Parameter Configurations Learn more

□ OOMGuard (OOM Daemon)

This addon reduces the kernel failures caused by cgroup memory reclaim failure in user mode.

Learn more

□ NodeProblemDetectorPlus (Node Exception Detection Plus)

The health monitoring component of nodes in the cluster. It can detect exceptions on nodes in real-time and report to kube-apiserver.

Parameter Configurations Learn more

□ NodeLocalDNSCache (Local DNS Cache Addon)

Run DNS cache proxy as the DaemonSet on the cluster node to improve cluster DNS performance

Learn more

□ LogCollector (Log collection addon)

Sends the logs of services in the cluster or those of files under a specific path in the node to a specified Topic of Kafka or specified log topic of CLS.

Learn more

□ GpuManager (GPU management component)

Provides an All-in-One GPU manager to implement the following feature: assign and share GPU, query GPU metric, and prepare device for container running.

Learn more

□ GameApp (Game load add-on)

A Kubernetes workload controller for container in-place update developed by Tencent. It supports in-place update and retain shared memory.

Learn more

□ DNSAutoscaler (DNS horizontal autoscaling component)

Obtain number of nodes and cores of the cluster via deployment, and auto-scaling the number of DNS replicas according to the preset scaling policy

Learn more

□ COS (Tencent Cloud COS)

This component implements the CSI interface, which can help container clusters use Tencent Cloud COS.

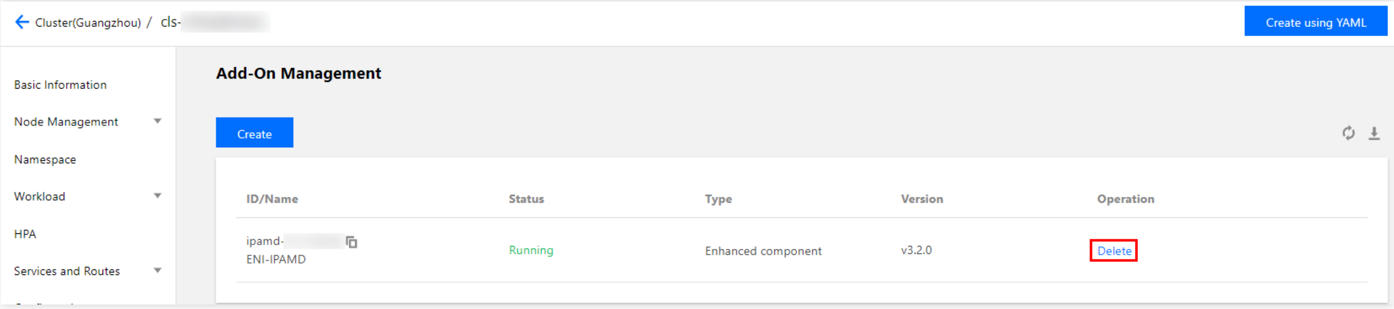
□ CFS (Tencent Cloud CFS)

This addon implements the CSI interface, which can help container clusters use Tencent Cloud CFS.

5. Select the add-ons to install and click **OK**.

Uninstalling an Add-on

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on management** to go to the “Add-on list” page.
4. On the "Add-on list" page, click **Delete** to the right of the target add-on.

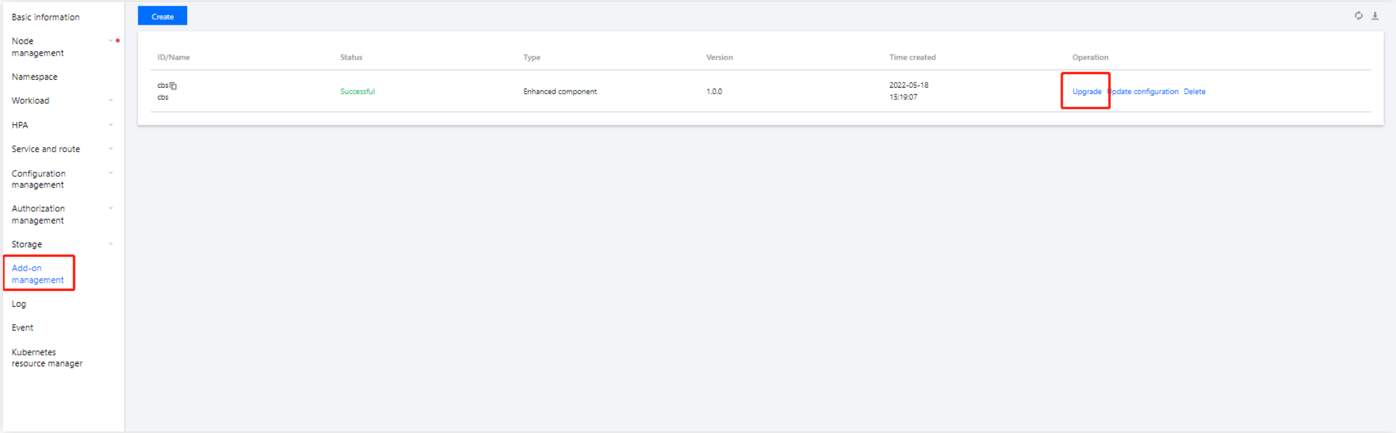


ID/Name	Status	Type	Version	Operation
ipamd-[redacted] ENI-IPAMD	Running	Enhanced component	v3.2.0	Delete

5. In the displayed "Delete resource" window, click **Confirm** to uninstall the add-on.

Upgrading an Add-on

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on management** to go to the “Add-on list” page.
4. On the "Add-on list" page, click **Upgrade** to the right of the target add-on.



ID/Name	Status	Type	Version	Time created	Operation
cbs cbs	Successful	Enhanced component	1.0.0	2022-05-18 15:19:07	Upgrade Update configuration Delete

5. In the displayed "Upgrade add-on" window, click **Confirm** to upgrade the add-on.

CBS-CSI Description
CBS-CSI

Last updated：2024-02-05 10:22:05

Operation Scenario

The CBS-CSI add-on allows you to select the storage class and create the corresponding PVs and PVCs of the CBS type in a TKE cluster on the console. This document introduces the features of the CBS-CSI add-on and some common use cases.

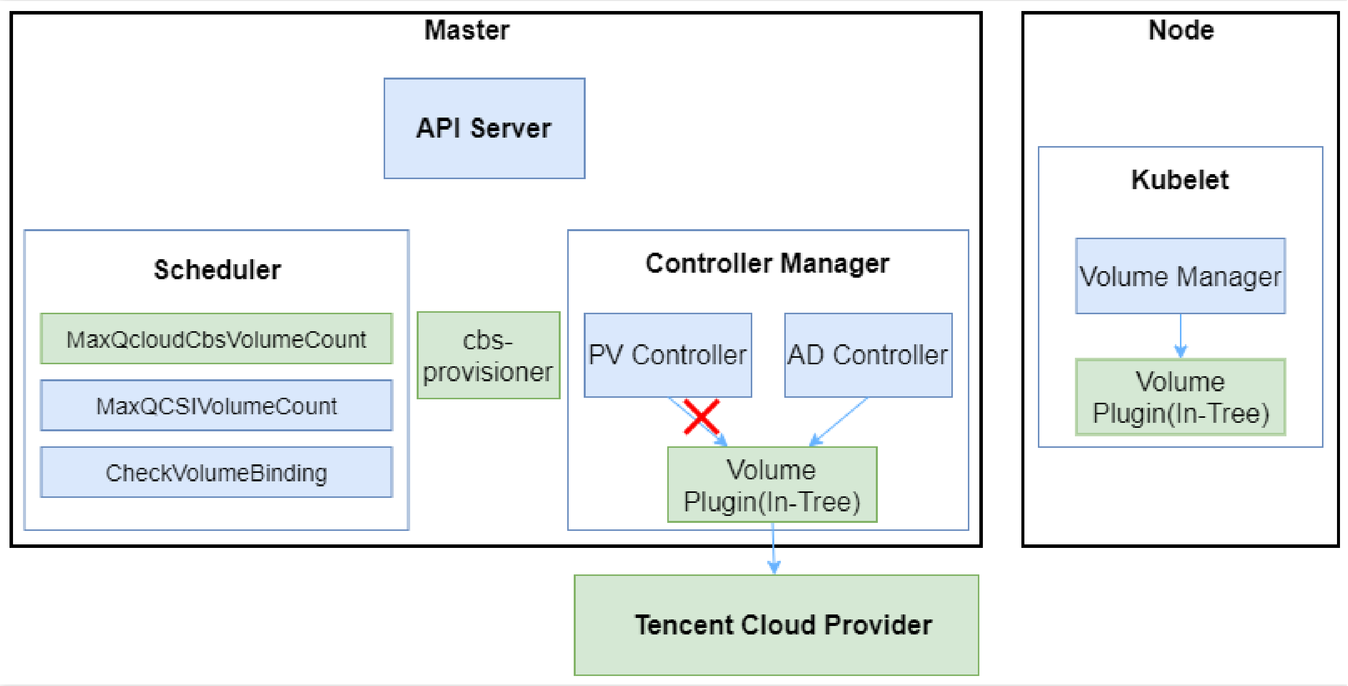
Features

Feature	Description
Static volume	Supports manual creation of volumes, PV objects, and PVC objects.
Dynamic volume	Supports configuration, creation, and deletion of volumes and PV objects through StorageClass.
Storage topology awareness	CBS does not support cross-AZ mounting. In a cluster with multiple AZs, the CBS-CSI add-on will schedule pods first, and then volumes will be created in the AZ of the node where the pods are scheduled.
Scheduler awareness of node maxAttachLimit	By default, one Tencent CVM instance can mount up to 20 cloud disks. When scheduling pods, the scheduler will filter out nodes where the number of mounted cloud disks has exceeded the limit.
Online volume expansion	You can modify the PVC capacity field to implement online expansion (only the CBS type is supported).
Volume snapshot and restoration	Supports the creation of volumes through snapshots.

Component Description

After it is deployed in a cluster, the CBS-CSI add-on contains the following components:

DaemonSet (NodePlugin): each node provides a DaemonSet. It consists of two containers, CBS-CSI Driver and node-driver-registrar. It is used to register the driver for the node and provide the ability to mount.
StatefulSet and Deployment (Controller): consists of a driver and multiple sidecars (external-provisioner, external-attacher, external-resizer, external-snapshotter, and snapshot-controller). It provides functions, such as o create or delete volumes, attach or detach, expand, and take snapshot.



Limits

TKE cluster version 1.14 or later
You can expand cloud disks online and create snapshots in a TKE cluster only after using the CBS-CSI add-on.
You can continue to use QcloudCbs (In-Tree plugin) in your TKE cluster. (It will be integrated to CBS-CSI through Volume Migration in the future.)

CBS-CSI 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.
The related directory /var/lib/kubelet on the host machine needs to be mounted to the container to accomplish volume mount/unmount, hence the activation of the privileged-level container is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permis
Perceiving the maximum number of disks that can be mounted on a node from the providerID in the Access Node resources	node	get/list
Executing disk creation and deletion based on pvc/pv information	pv/pvc/storageclasses/csinode	get/list/watch/create/update/p
Completing disk mounting and uninstallation based on volumeattachments resource objects	volumeattachments/volumesnapshotclasses	create/get/list/watch/update/c
Expanding disk capacity via snapshot	pod/volumesnapshotclasses/volumesnapshots/configmap	get/list/watch

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cbs-csi-controller-role
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "update", "patch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims/status"]
    verbs: ["update", "patch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "list", "watch", "create", "update", "patch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["csinodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["coordination.k8s.io"]
    resources: ["leases"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
  - apiGroups: ["csi.storage.k8s.io"]
    resources: ["csinodeinfos"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["volumeattachments", "volumeattachments/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshotclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshotcontents"]
    verbs: ["create", "get", "list", "watch", "update", "delete"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshots"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["apiextensions.k8s.io"]
    resources: ["customresourcedefinitions"]
    verbs: ["create", "list", "watch", "delete"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshotcontents/status"]
    verbs: ["update"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshots/status"]
    verbs: ["update"]
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list", "watch", "update", "patch", "create", "delete"]
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cbs-csi-node-role
  namespace: kube-system
rules:
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list"]
```

Use Cases

Avoid attaching cloud disk across availability zones through cbs-csi

Online Expansion of Cloud Disk

Creating Snapshot and Using It to Restore Volume

Avoid attaching cloud disk across availability zones through cbs-csi

Last updated：2022-12-12 10:28:24

Overview

CBS cloud disks do not support cross-AZ mounting to nodes. Therefore, in cross-AZ clusters, we recommend that you use the CBS-CSI **topology awareness** feature to avoid cross-AZ mounting problems.

How it works

Topology-aware scheduling requires the cooperation of multiple Kubernetes components, including the Scheduler, PV controller, and external-provisioner. The detailed process is as follows:

1. The PV controller observes PVC objects and checks whether VolumeBindingMode of the Storageclass is **WaitForFirstConsumer**. If yes, it does not process the PVC creation event but waits for the Scheduler to process it.
2. After the Scheduler schedules the pod, it will mark the nodeName on the PVC object as an annotation: `volume.kubernetes.io/selected-node: 10.0.0.72` .
3. After the PV controller obtains the update event of the PVC object, it processes the annotation (`volume.kubernetes.io/selected-node`), obtains the node object based on the nodeName, and then passes it to the external-provisioner.
4. The external-provisioner obtains the AZ based on the label of the passed node object (`failure-domain.beta.kubernetes.io/zone`), and then creates the PV in the corresponding AZ. In this way, it can be in the same AZ as the pod, and you can prevent cloud disk mounting failure caused by cloud disks and the node being in different AZs.

Prerequisites

- You have installed a TKE cluster of v1.14 or later versions. For more information, see Creating a Cluster.
- You have updated CBS-CSI or In-Tree to the latest version.

Directions

Use the following YAML to set volumeBindingMode to **WaitForFirstConsumer** in the Storageclass. Below is a sample:

```
kind: StorageClass
metadata:
name: cbs-topo
parameters:
type: cbs
provisioner: com.tencent.cloud.csi.cbs
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

Note：
Both CBS-CSI and In-Tree support this operation.

Online Expansion of Cloud Disk

Last updated：2023-12-20 09:25:18

Overview

TKE supports online expansion of PVs, as well as the corresponding CBS and file system. Expansion can be completed without the need to restart pods. To ensure the stability of the file system, we recommend that you perform this operation when the CBS file system is not mounted.

Prerequisites

You have created a TKE cluster of v1.16 or later versions. For more information, see Creating a Cluster.
You have updated CBS-CSI to the latest version.
(Optional) You can use snapshot to back up data before expansion to avoid data loss due to expansion failure.
PV of non-CBS-CSI type in earlier versions of cluster 1.20 does not support online expansion.

Directions

Creating a StorageClass that allows expansion

You can use the following YAML to create a StorageClass that allows expansion. Set `allowVolumeExpansion` to `true` in the Storageclass. Below is an example:

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cbs-csi-expand
parameters:
  diskType: CLOUD_PREMIUM
provisioner: com.tencent.cloud.csi.cbs
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Online expansion

Two expansion methods are provided:

Expansion Method	Description
Online expansion with restarting the Pod	The CBS document system to be expanded is not mounted, and expansion errors and issues in method 2 can be avoided. **We recommend that you use this method for expansion**.
Online expansion without restarting the Pod	The CBS document system to be expanded is mounted on the node. If there is an I/O process, a document system expansion error may occur.

Online expansion with restarting the Pod

Online expansion without restarting the Pod

1. Run the following command to confirm the status of the PV and the document system before expansion. In the following example, the size of both PV and document system is 30G.

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/vdd        30832548 44992  30771172   1% /usr/share/nginx/html

$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   30Gi       RWO            Delete
```

2. Run the following command to tag the PV object with an invalid zone label, which aims to make the Pod unable to be scheduled to a node after it is restarted in the next step. Below is an example:

```
$ kubectl label pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c failure-domain.beta.kub
```

3. Run the following command to restart the Pod. The Pod will be in the `Pending` status because the label of the PV corresponding to the Pod indicates that it is in an invalid zone. Below is an example:

```
$ kubectl delete pod ivantestweb-0

$ kubectl get pod ivantestweb-0
NAME            READY   STATUS    RESTARTS   AGE
ivantestweb-0   0/1     Pending   0          25s

$ kubectl describe pod ivantestweb-0
Events:
Type     Reason             Age                  From                 Message

----     ------             ----                 ----                 -------
```

```
Warning  FailedScheduling  40s (x3 over 2m3s)  default-scheduler  0/1 nodes are ava
```

4. Run the following command to expand the capacity of the PVC object to 40G. Below is an example:

```
kubectl patch pvc www1-ivantestweb-0 -p '{"spec":{"resources":{"requests":{"storage
```

Caution:

The PVC object capacity after expansion must be a multiple of 10. For more information on the storage capacity specifications supported by different cloud disk types, see Creating Cloud Disks.

5. Run the following command to remove the label of the PVC object. In this way, the Pod can be scheduled successfully. Below is an example:

```
$ kubectl label pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c failure-domain.beta.kub
persistentvolume/pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c labeled
```

6. Run the following command. You can see that the status of the Pod is `Running` , and the size of both the corresponding PV and document system has expanded from 30G to 40G. Below is an example:

```
$ kubectl get pod ivantestweb-0
NAME            READY   STATUS    RESTARTS   AGE
ivantestweb-0   1/1     Running   0          17m

$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   40Gi       RWO            Delete

$ kubectl get pvc www1-ivantestweb-0
NAME                 STATUS   VOLUME                                     CAPACITY
www1-ivantestweb-0   Bound    pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   40Gi
```

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/vdd        41153760 49032  41088344   1% /usr/share/nginx/html
```

1. Run the following command to confirm the status of the PV and the document system before expansion. In the following example, the size of both PV and document system is 20G.

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem     1K-blocks  Used Available Use% Mounted on
```

```
/dev/vdd        20511312 45036  20449892   1% /usr/share/nginx/html

$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   20Gi       RWO            Delete
```

2. Run the following command to expand the capacity of the PVC object to 30G. Below is an example:

```
$ kubectl patch pvc www1-ivantestweb-0 -p '{"spec":{"resources":{"requests":{"stora
```

Caution:

The PVC object capacity after expansion must be a multiple of 10. For more information on the storage capacity specifications supported by different cloud disk types, see Creating Cloud Disks.

3. Run the following command. You can see that the size of both PV and document system has expanded to 30G. Below is an example:

```
$ kubectl exec ivantestweb-0 df /usr/share/nginx/html
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/vdd        30832548 44992  30771172   1% /usr/share/nginx/html
```

```
$ kubectl get pv pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   30Gi       RWO            Delete
```

Creating Snapshot and Using It to Restore Volume

Last updated：2022-11-11 11:15:47

Overview

If you need to create a snapshot of the PVC data disk to back up data, or to restore the backup snapshot data to a new PVC, you can use the CBS-CSI add-on. This document describes how to use the CBS-CSI add-on to implement data backup and restoration of PVC.

Prerequisites

- You have created a TKE cluster on v1.18 or later versions. For more information, see Creating a Cluster.
- You have installed the latest version of CBS-CSI.

Directions

Backing up PVC

Creating `VolumeSnapshotClass`

1. Use the following YAML to create a `VolumeSnapshotClass` object:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
name: cbs-snapclass
driver: com.tencent.cloud.csi.cbs
deletionPolicy: Delete
```

2. Run the following command to see if the `VolumeSnapshotClass` is created successfully:

```
$ kubectl get volumesnapshotclass
NAME DRIVER DELETIONPOLICY AGE
cbs-snapclass com.tencent.cloud.csi.cbs Delete 17m
```

Creating PVC snapshot `VolumeSnapshot`

1. This document takes `new-snapshot-demo` as an example to use the following YAML to create a `VolumeSnapshot` object.

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
name: new-snapshot-demo
spec:
volumeSnapshotClassName: cbs-snapclass
source:
persistentVolumeClaimName: csi-pvc
```

2. Run the following command to check whether the `Volumesnapshot` and `Volumesnapshotcontent` objects have been created successfully. If `READYTOUSE` is `true`, the creation is successful.

```
$ kubectl get volumesnapshot
NAME READYTOUSE SOURCEPVC SOURCESNAPSHOTCONTENT RESTORESIZE SNAPSHOTCLASS SNAPS
HOTCONTENT CREATIONTIME AGE
new-snapshot-demo true www1-ivantestweb-0 10Gi cbs-snapclass snapcontent-ea11a7
97-d438-4410-ae21-41d9147fe610 22m 22m
```

```
$ kubectl get volumesnapshotcontent
NAME READYTOUSE RESTORESIZE DELETIONPOLICY DRIVER VOLUMESNAPSHOTCLASS VOLUMESNAPS
HOT AGE
snapcontent-ea11a797-d438-4410-ae21-41d9147fe610 true 10737418240 Delete com.tenc
ent.cloud.csi.cbs cbs-snapclass new-snapshot-demo 22m
```

3. Run the following command to obtain the snapshot ID of the `Volumesnapshotcontent` object. The field is `status.snapshotHandle` (here takes `snap-e406fc9m` as an example). You can log in to the CVM console > Snapshot List and use the snapshot ID to check whether the snapshot exists, as shown below:

```
$ kubectl get volumesnapshotcontent snapcontent-ea11a797-d438-4410-ae21-41d9147
fe610 -oyaml
```

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotContent
metadata:
creationTimestamp: "2020-11-04T08:58:39Z"
finalizers:
- snapshot.storage.kubernetes.io/volumesnapshotcontent-bound-protection
```

```
name: snapcontent-ea11a797-d438-4410-ae21-41d9147fe610
resourceVersion: "471437790"
selfLink: /apis/snapshot.storage.k8s.io/v1beta1/volumesnapshotcontents/snapconten
t-ea11a797-d438-4410-ae21-41d9147fe610
uid: 70d0390b-79b8-4276-aa79-a32e3bdef3d6
spec:
deletionPolicy: Delete
driver: com.tencent.cloud.csi.cbs
source:
volumeHandle: disk-7z32tin5
volumeSnapshotClassName: cbs-snapclass
volumeSnapshotRef:
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
name: new-snapshot-demo
namespace: default
resourceVersion: "471418661"
uid: ea11a797-d438-4410-ae21-41d9147fe610
status:
creationTime: 1604480319000000000
readyToUse: true
restoreSize: 10737418240
snapshotHandle: snap-e406fc9m
```

Restoring data from the snapshot to a new PVC

1. This document takes the `VolumeSnapshot` object `new-snapshot-demo` created in the previous step as an example and uses the following YAML to restore volume from the snapshot.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: restore-test
spec:
storageClassName: cbs-csi
dataSource:
name: new-snapshot-demo
kind: VolumeSnapshot
apiGroup: snapshot.storage.k8s.io
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 10Gi
```

2. Run the following command to check whether the restored PVC has been created successfully. You can view the corresponding `diskid` in the PV (here takes `disk-gahz1kw1` as an example).

```
$ kubectl get pvc restore-test
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
restore-test Bound pvc-80b98084-29a3-4a38-a96c-2f284042cf4f 10Gi RWO cbs-csi 97s
```

```
$ kubectl get pv pvc-80b98084-29a3-4a38-a96c-2f284042cf4f -oyaml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
annotations:
pv.kubernetes.io/provisioned-by: com.tencent.cloud.csi.cbs
creationTimestamp: "2020-11-04T12:08:25Z"
finalizers:
- kubernetes.io/pv-protection
name: pvc-80b98084-29a3-4a38-a96c-2f284042cf4f
resourceVersion: "474676883"
selfLink: /api/v1/persistentvolumes/pvc-80b98084-29a3-4a38-a96c-2f284042cf4f
uid: 5321df93-5f21-4895-bafc-71538d50293a
spec:
accessModes:
- ReadWriteOnce
capacity:
storage: 10Gi
claimRef:
apiVersion: v1
kind: PersistentVolumeClaim
name: restore-test
namespace: default
resourceVersion: "474675088"
uid: 80b98084-29a3-4a38-a96c-2f284042cf4f
csi:
driver: com.tencent.cloud.csi.cbs
fsType: ext4
volumeAttributes:
diskType: CLOUD_PREMIUM
storage.kubernetes.io/csiProvisionerIdentity: 1604478835151-8081-com.tencent.cloud.csi.cbs
volumeHandle: disk-gahz1kw1
nodeAffinity:
required:
```

```
nodeSelectorTerms:
- matchExpressions:
- key: topology.com.tencent.cloud.csi.cbs/zone
operator: In
values:
- ap-beijing-2
persistentVolumeReclaimPolicy: Delete
storageClassName: cbs-csi
volumeMode: Filesystem
status:
phase: Bound
```

Note：

If `StorageClass` uses topology awareness (to schedule the Pod before creating the PV), that is, to specify `volumeBindingMode: WaitForFirstConsumer` , you need to deploy the Pod (mount the PVC) to trigger the PV creation (create a CBS from the snapshot and bind it to the PV).

UserGroupAccessControl

Last updated：2023-08-01 17:07:54

Overview

Add-on description

With UserGroupAccessControl, you can integrate Kubernetes RBAC into a Tencent Cloud CAM user group to control sub-account access in a refined manner.

Kubernetes objects deployed in a cluster

Kubernetes object name	Type	Specification	Namespaces
user-group-access-control	ServiceAccount	-	kube-system
user-group-access-control	ClusterRole	-	kube-system
user-group-access-control	ClusterRoleBinding	-	kube-system
user-group-access-control	Service	-	kube-system
user-group-access-control	ConfigMap	-	kube-system
user-group-access-control	Deployment	0.5C1G (for new Kubernetes objects)	kube-system

Use Cases

A CAM user group is a collection of multiple users (sub-accounts) with similar roles. It can provide authorization and set subscription messages in batches. UserGroupAccessControl can help setting the same Kubernetes object access permissions for sub-accounts with the same function in a TKE general cluster.

Limits

Supported K8s cluster versions: v1.16 and later versions.

Directions

Note:

To use the UserGroupAccessControl add-on, please submit a ticket.

Step 1. Create a user group

Create a user group in CAM. For details, see Creating User Group. If you already have a user group, skip this step.

Step 2. Install the add-on

1. Log in to the TKE console. In the left sidebar, click **Cluster**.
2. On the **Cluster** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on management**. On the page that appears, click **Create**.
4. On the **Create add-on** page, select the **Authentication authorization** module and select **UserGroupAccessControl**.
5. Click **Service authorization**. Associate the "TKE_QCSRole" role with the preset policy "QcloudAccessForTKERoleInGroupsForUser" to allow TKE access information of user groups under your account. On the **Service authorization** page, confirm the role name and authorization policy, and click **Grant**.
6. Go back to the **Create add-on** page, click **Complete**. Now, you can view the add-on details on the **Add-on management** page.

Step 3. Create a role and bind the policy to the user group

1. In the left sidebar, click **Authorization Management** > **ClusterRole**. Click **RBAC Policy Generator** on the **ClusterRole** page.
2. Select **User group** for account type, and select the target user group.
3. Click **Next**. In **Cluster RBAC settings**, set Kubernetes object access permissions for the specified user group.
4. Click **Complete**.

Step 4: View the role binding policy

In the left sidebar, click **Authorization management** > **ClusterRoleBinding**. Check the policy that is named starting with the user group ID.

Note:

To manage permissions for Tencent Cloud resources (such as migrating sub-accounts, adding/removing permission for cloud resources), you only need to make changes in the CAM user group. The policy associated with the created role will be updated at the same time. For details, see Managing User Groups.

COS-CSI

Last updated：2024-02-01 10:03:01

Overview

Add-on description

The Kubernetes-csi-tencentcloud COS-CSI plug-in allows you to use Tencent Cloud Object Storage (COS) in your TKE cluster.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Default Resource Consumption	Namespaces
csi-coslauncher	DaemonSet	-	kube-system
csi-cosplugin	DaemonSet	-	kube-system
csi-cos-tencentcloud-token	Secret	-	kube-system

Use Cases

COS is a distributed storage service provided by Tencent Cloud to store massive files. You can store and view data at any time over a network. Tencent Cloud COS provides scalable, affordable, reliable, and secure data storage services for all users.
With the COS-CSI add-on, you can quickly use COS as COSFS in your cluster through standard native Kubernetes. For more information, see COSFS.

Limits

Supports clusters with Kubernetes version 1.10 and later.
For Kubernetes 1.12 clusters, the following kubelet configuration must be added: `--feature-gates=KubeletPluginsWatcher=false` .
For more information on the limits of COSFS, see COSFS.
To use COS in TKE, you must install this add-on in your cluster, which consumes some system resources.

COS-CSI Permission

Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.
The related directory /var/lib/kubelet on the host machine needs to be mounted to the container to accomplish volume mount/unmount, hence the activation of the privileged-level container is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
Supporting COS bucket mounting in lite mode	PersistentVolume	get/watch/list/update
	pod	get/create/delete/update
Storing related COS configuration in the lite mounting method	configmap	get/create/delete/update

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-cos-tencentcloud
rules:
  - apiGroups: [""]
    resources: ["events", "persistentvolumes"]
    verbs: ["get", "watch", "update", "list"]
  - apiGroups: [""]
    resources: ["pods", "configmaps"]
    verbs: ["get", "create", "delete", "update"]
```

Usage

Installing the COS add-on

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the "Cluster Management" page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-On Management** to go to the "Add-On List" page.
4. On the "Add-On List" page, click **Create**. On the "Create an Add-On" page that appears, select **COS**.
5. Click **Finish** to create the add-on.

Using COS

You can mount COS for workloads in a TKE cluster. For more information, see Using COS.

CFS-CSI

Last updated：2024-02-01 10:05:00

Overview

Add-on description

The Kubernetes-csi-tencentcloud CFS-CSI plug-in allows you to use Tencent Cloud File Storage in your TKE cluster.

Note:

For clusters of version 1.12, you need to modify the kubelet configuration by adding `\\--feature-gates=KubeletPluginsWatcher=false\\` .

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Default Resource Occupation	Namespace
csi-provisioner-cfsplugin	StatefulSet	-	kube-system
csi-nodeplugin-cfsplugin	DaemonSet	-	kube-system
csi-provisioner-cfsplugin	Service	1C2G	kube-system

Use Cases

Cloud File Storage (CFS) provides a scalable shared file storage service that can be used with Tencent Cloud services such as CVM, TKE, and BatchCompute. CFS offers standard NFS and CIFS/SMB file system access protocols to provide shared data sources for multiple CVM instances or other computing services. It supports elastic capacity expansion and performance scaling. CFS can be mounted on existing applications without modification. As a highly available and reliable distributed file system, CFS is suitable for various scenarios such as big data analysis, media processing, and content management.

CFS is easy to integrate. You do not need to adjust your business structure or make complex configurations. You can integrate and use CFS in three steps: create a file system, launch a file system client on the server, and mount the created file system. With the CFS-CSI add-on, you can quickly use CFS through the standard native Kubernetes in your TKE cluster. For more information, see CFS Usage.

Limits

For the limits of CFS, see CFS System Limits.
To use CFS in TKE, you need to install this add-on in your cluster, which will occupy some system resources.

CFS-CSI 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.
The related directory /var/lib/kubelet on the host machine needs to be mounted to the container to accomplish volume mount/unmount, hence the activation of the privileged-level container is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
It is required to support the dynamic creation of CFS instances.	persistentvolumeclaims/persistentvolumes	All operations
	storageclasses	get/list/watch
Supporting the cfs instance under the shared pattern	tcfs	get/list/watch/create/update/delete/patch
	deployment	get/list/watch/create/update/delete
	node	get/list

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-cfs-controller-role
rules:
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list"]
  - apiGroups: [""]
    resources: ["services", "events", "configmaps", "endpoints"]
    verbs: ["get","list","create","update","patch","delete"]
  - apiGroups: [""]
    resources: ["services/status", "events/status"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete", "update"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update", "patch", "create"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["volumeattachments", "volumeattachments/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["extensions"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "update", "patch", "create"]
  - apiGroups: ["extensions"]
    resources: ["ingresses/status"]
    verbs: ["get"]
  - apiGroups: ["apps"]
    resources: ["deployments"]
    verbs: ["get", "list", "delete", "update", "create", "watch"]
  - apiGroups: ["apps"]
    resources: ["deployments/status"]
    verbs: ["get"]
  - apiGroups: ["tcfsoperator.k8s.io"]
    resources: ["tcfs", "tcfs/status"]
    verbs: ["get", "list", "watch", "create", "delete", "update", "patch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: tcfs-subdir-external-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "update", "patch"]
```

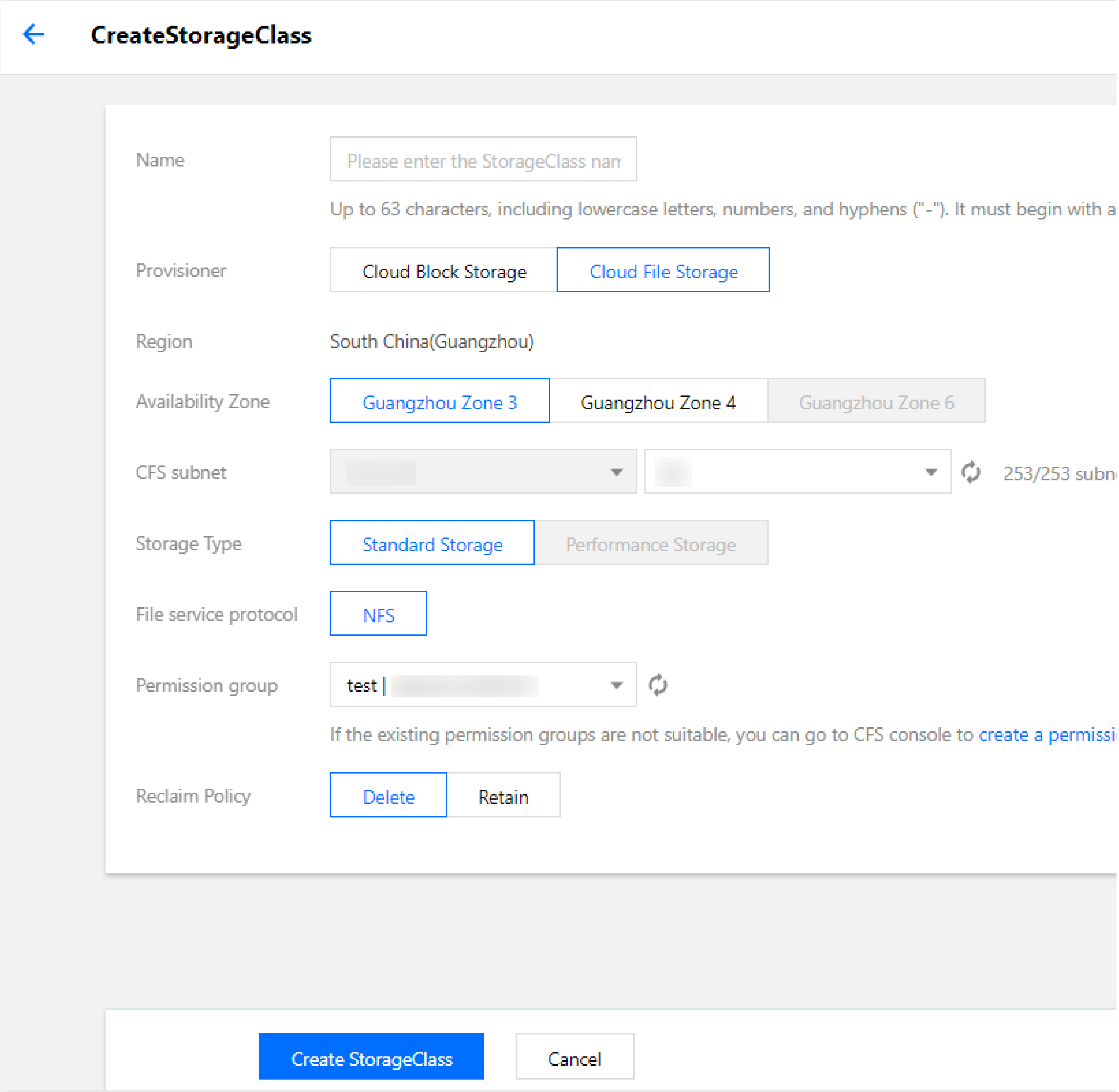
Directions

Installing and setting the CFS add-on

1. Log in to the TKE Console and select **Cluster** in the left sidebar.
2. On the "Cluster Management" page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on Management** to go to the "Add-on List" page.
4. On the "Add-on List" page, click **Create**. On the displayed "Create an Add-on" page, select **CFS**.
5. Click **Finish** to create the add-on.

Creating a CFS-type StorageClass

1. On the "Cluster Management" page, click the ID of the cluster that uses CFS to go to the cluster details page.
2. In the left sidebar, choose **Storage** > **StorageClass** and click **Create** to go to the "Create a StorageClass" page.
3. Create a CFS-type StorageClass based on your actual requirements, as shown in the figure below:



CreateStorageClass

Name: Please enter the StorageClass nam

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a

Provisioner: Cloud Block Storage | Cloud File Storage

Region: South China(Guangzhou)

Availability Zone: Guangzhou Zone 3 | Guangzhou Zone 4 | Guangzhou Zone 6

CFS subnet: [redacted] [redacted] 253/253 subn

Storage Type: Standard Storage | Performance Storage

File service protocol: NFS

Permission group: test | [redacted]

If the existing permission groups are not suitable, you can go to CFS console to create a permissi

Reclaim Policy: Delete | Retain

4. Click **Create a StorageClass** to complete the creation.

Creating a PersistentVolumeClaim

1. On the "Cluster Management" page, click the ID of the cluster that uses CFS to go to the cluster details page.
2. In the left sidebar, choose **Storage** > **PersistentVolumeClaim** and click **Create** to go to the "Create a PersistentVolumeClaim" page.
3. Create a CFS-type PersistentVolumeClaim based on your actual requirements and select the StorageClass created above.
4. Click **Create a PersistentVolumeClaim** to complete the creation process.

Creating a workload

1. On the "Cluster Management" page, click the ID of the cluster that uses CFS to go to the cluster details page.
2. In the left sidebar, choose **Workload** > **Deployment** and click **Create** to go to the "Create a Workload" page.
3. Based on your actual requirements, select **Use an existing PVC** for volumes and select the PVC created above.
4. Mount the PVC to the specified container path and click **Create a Workload** to complete the creation process.

P2P

Last updated：2020-11-25 11:49:11

Overview

Add-on description

The P2P add-on is a Kubernetes plug-in provided by the Tencent Container Registry (TCR) service for the accelerated distribution of container images. Based on P2P technology, this add-on can accelerate the pulling of container images in the gigabytes by massive TKE clusters. It supports concurrent pulling by thousands of nodes.

This add-on consists of `p2p-agent` , `p2p-proxy` , and `p2p-tracker` .

- p2p-agent is deployed on each node in a cluster. It serves as the agent for receiving image pull requests from each node and forwarding these requests to each peer (node) in the P2P network.
- p2p-proxy is deployed on some nodes in a cluster. It serves as the raw seed to connect to the image repository to be accelerated. In addition to serving as seeds, proxy nodes need to pull raw data from the target image repository.
- p2p-tracker is deployed on some nodes in a cluster. It serves as the tracker service of the open-source BitTorrent protocol.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Requested Resource	Namespace
p2p-agent	DaemonSet	Per node: 0.2 CPU cores and 0.2 GB memory	kube-system
p2p-proxy	Deployment	Per node: 0.5 CPU cores and 0.5 GB memory	kube-system
p2p-tracker	Deployment	Per node: 0.5 CPU cores and 0.5 GB memory	kube-system
p2p-proxy	Service	-	kube-system
p2p-tracker	Service	-	kube-system
agent	Configmap	-	kube-system
proxy	Configmap	-	kube-system
tracker	Configmap	-	kube-system

Use Cases

This add-on can accelerate the pulling of container images in the gigabytes by massive TKE clusters and supports concurrent pulling by thousands of nodes. Its recommended use cases are as follows:

- The cluster has 500 to 1,000 nodes, and local disks are used to store pulled container images. In this scenario, nodes in the cluster support a maximum concurrent pull speed of 100 MB/s.
- The cluster has 500 to 1,000 nodes, CBS cloud disks are used to store pulled container images, and the cluster is located in a major Chinese region, such as Guangzhou, Beijing, or Shanghai. In this scenario, nodes in the cluster support a maximum concurrent pull speed of 20 MB/s.

Limits

- If the P2P add-on is enabled in a large-scale cluster to pull container images, a high read/write pressure is imposed on node data disks, which may affect existing businesses in the cluster. If cluster nodes use CBS cloud disks to store pulled container images, select a proper download speed limit based on the region of the cluster or contact your after-sales agent/architect to prevent read/write overload of cloud disks from interrupting existing businesses in the cluster during image pulling or even affecting the normal operations of other users in the region.
- To enable the P2P add-on, you must reserve some resources. During image pull acceleration, the P2P add-on consumes the CPU and memory resources of nodes. These resources are released after the acceleration is completed.
  - The limit of p2p-proxy is 4 CPU cores and 4 GB memory.
  - The limit of p2p-agent is 4 CPU cores and 2 GB memory.
  - The limit of p2p-tracker is 2 CPU cores and 4 GB memory.
- You need to estimate the number of p2p-proxies to be launched based on the node scale of the cluster. The minimum node configuration for running a p2p-proxy is 4 CPU cores and 8 GB memory, with a private network bandwidth of 1.5 GB/s. A single p2p-proxy supports up to 200 cluster nodes.
- You need to select the nodes on which to deploy p2p-proxy and p2p-tracker by manually attaching Kubernetes labels. For more information, see Usage. The nodes where p2p-proxy and p2p-tracker are located must be able to access the origin server of the repository.
- P2p-agent uses port 5004 of the nodes and P2P dedicated communication ports 6881 (for p2p-agent) and 6882 (for p2p-proxy). P2p-agent and p2p-proxy will create the local work directories `/p2p_agent_data` and `/p2p_proxy_data` respectively to cache container images. Ensure in advance that nodes have reserved sufficient storage space.

Usage

1. Select proper nodes on which to deploy and run p2p-proxy.
   You can label nodes by running `kubectl label nodes XXXX proxy=p2p-proxy` . Then, p2p-proxy will be

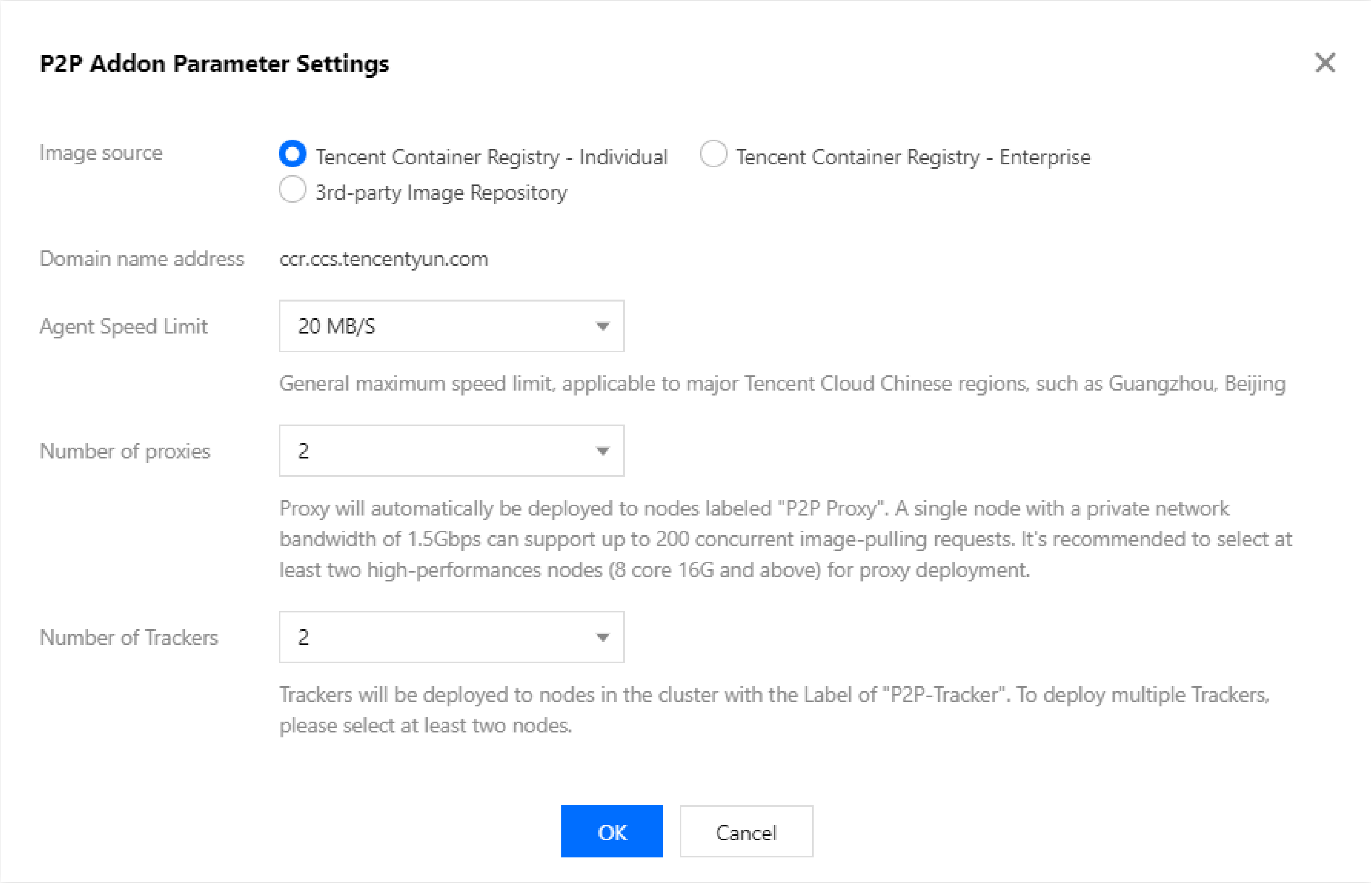
automatically deployed on these nodes upon installation of the P2P add-on. After the installation, if you want to adjust the number of p2p-proxies, you can add or delete the labels on the specified nodes and then change the number of p2p-proxy workload replicas under the kube-system namespace of the cluster.

2. Select proper nodes on which to deploy and run p2p-tracker.
   You can label nodes by running `kubectl label nodes XXXX tracker=p2p-tracker` . Then, p2p-tracker will be automatically deployed on these nodes upon installation of the P2P add-on. After the installation, if you need to adjust the number of p2p-trackers, you can add or delete the label on the specified nodes, and then change the number of p2p-tracker workload replicas under the kube-system namespace of the cluster.
3. The following configuration must be added to the node security group: in inbound rules, open ports 30000-32768 of TCP and UDP, and open all IP addresses in the VPC to the Internet. In outbound rules, open all ports (the default security group of TKE cluster work nodes already meets this requirement).
4. Select the specified cluster to activate the P2P add-on. Enter the domain name of the image repository to be accelerated, node pull speed limit, number of p2p-proxies, and number of p2p-trackers. After installation, if you want to adjust the maximum download speed, modify downloadRate and uploadRate in p2p-agent configmap.
5. In the business namespace, create a dockercfg for pulling images where the repository domain name is localhost:5004, and the username and password are the original access credential of the target image repository.
6. Modify the business YAML file by changing the domain name address of the image repository to be accelerated to localhost:5004 (for example, localhost:5004/p2p-test/test:1.0) and using the newly created dockercfg as ImagePullSecret.
7. Use the business YAML file to deploy updated workloads and monitor the image pull speed and the read/write load of the node disks in real time. Adjust the download speed limit of nodes in time to achieve optimal acceleration.

Directions

1. Log in to the Tencent Kubernetes Engine console and click **Cluster** in the left sidebar.
2. On the "Cluster Management" page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on Management** to go to the "Add-on List" page.
4. On the "Add-on List" page, click **Create**. On the "Create an Add-on" page that appears, select **P2P**.
5. Choose **Parameter Configuration**. In the displayed "P2P Add-on Parameter Settings" window, specify the domain name of the image repository to be accelerated, node pull speed limit, number of p2p-proxies, and number

of p2p-trackers, as shown in the figure below:



P2P Addon Parameter Settings

Image source: ● Tencent Container Registry - Individual ○ Tencent Container Registry - Enterprise ○ 3rd-party Image Repository

Domain name address: ccr.ccs.tencentyun.com

Agent Speed Limit: 20 MB/S

General maximum speed limit, applicable to major Tencent Cloud Chinese regions, such as Guangzhou, Beijing

Number of proxies: 2

Proxy will automatically be deployed to nodes labeled "P2P Proxy". A single node with a private network bandwidth of 1.5Gbps can support up to 200 concurrent image-pulling requests. It's recommended to select at least two high-performances nodes (8 core 16G and above) for proxy deployment.

Number of Trackers: 2

Trackers will be deployed to nodes in the cluster with the Label of "P2P-Tracker". To deploy multiple Trackers, please select at least two nodes.

OOMGuard

Last updated：2024-02-05 16:08:13

Overview

Note:
This add-on reduces the chance of various kernel failures triggered by cgroup memory reclamation failures in the user mode. It applies only to native kernel defects of CentOS 7.2/7.6. For other image versions, you do not need to install this add-on.

Add-on Description

Out of Memory (OOM) indicates the programs run with more memory than the maximum memory available because memory cannot be repossessed or is used too much in the application system. When cgroup memory is insufficient, Linux kernel triggers cgroup OOM to kill some processes, so as to repossess some memory to keep continuous operation of the system. As many bugs may occur while Linux kernel (especially the earlier versions such as v3.10) processes cgroup OOM, frequent cgroup OOM occurrence may result in node failures (crash, restart, and unkillable abnormal processes).
OOM-Guard is an add-on provided by TKE for processing container cgroup OOM in user mode. When cgroup OOM occurs, before the kernel kills the container process, OOM-Guard kills the excessive container in the user space. This reduces the chance of various node failures triggered by memory repossessing failures in kernel mode.
Before the OOM threshold is triggered, OOM-Guard writes `memory.force_empty` to trigger relevant cgroup memory repossessing. If `memory.stat` still contains a large amount of cache data, no subsequent processing policies will be triggered. After a container is killed due to cgroup OOM, the add-on reports the `OomGuardKillContainer` event to Kubernetes. You can query the event by running the `kubectl get event` command.

How It Works

The core concept is to kill the excessive containers in user space before kernel kills the container processes due to cgroup OOM. This reduces the chance of various kernel errors triggered by code branches that encounter repossessing failure of kernel cgroup memory.
OOM-Guard will set "threshold notify" mechanism for memory cgroup to receive notifications from the kernel. For more information, see threshold notify.

Sample

For example, the memory limit set for a pod is 1000M, OOM-Guard will calculate margin based on the configuration parameters.

```
margin = 1000M * margin_ratio = 20M // the default value of margin_ratio is 0.02
```

In addition, the minimum value of margin is min_margin (1M) and maximum value is max_margin (50M). If it exceeds the limit, min_margin or max_margin is applied.

Calculate the threshold:

```
threshold = limit - margin // i.e. 1000M - 20M = 980M
```

980M is the threshold that is set to the kernel. When the memory used by the pod reaches 980M, OOM-Guard will receive a notification sent by the kernel.

Before threshold is triggered, OOM-Guard writes `memory.force_empty` to trigger relevant cgroup memory repossessing. In addition, if threshold is triggered and `memory.stat` of relevant cgroup still contains a large amount of cache data, the subsequent processing policies will not be triggered. Thus, when cgroup memory reaches the limit, kernel still triggers cgroup OOM.

Processing policy applied when threshold is reached

You can control the processing policies by setting the `--policy` parameter. The following three policies are available for now. The default policy is "container".

Policy	Description
process	It uses a policy the same as the cgroup OOM killer. It selects a process with the highest value of oom_score inside the cgroup, and kills the process by "SIGKILL" sent from OOM-Guard.
container	It selects a docker container under this cgroup and kills the whole container.
noop	It only records logs but does not take any action.

Kubernetes objects deployed in a cluster

Kubernetes Object	Type	Required Resources	Namespaces
oomguard	ServiceAccount	-	kube-system
system:oomguard	ClusterRoleBinding	-	-
oom-guard	DaemonSet	0.02-core CPU, 120 MB memory	kube-system

Overview

This add-on is suitable for Kubernetes clusters where the node memory pressure is high and node failures are often caused by business container OOM.

Limits

The containerd service socket path is not changed, and the default path of TKE is retained:
docker runtime: `/run/docker/containerd/docker-containerd.sock`
containerd runtime: `/run/containerd/containerd.sock`
The mount target of the cgroup memory subsystem is not changed, and the default mount target `/sys/fs/cgroup/memory` is retained.

Component Permission Description

Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.
The OOM guard necessitates transmitting the occurrence of OOM through events when it arises, thus it requires create/patch/update permissions of the events.

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:oomguard
rules:
```

```
- apiGroups:
  - ''
  resources:
  - 'events'
  verbs:
  - create
  - patch
  - update
```

How to Use

1. Log into the TKE console and click **Cluster** in the left sidebar.
2. In the **Cluster** list, click the ID of the target cluster to go to the cluster details page.
3. Select **Add-on management** in the left sidebar, and click **Create** on the Add-on management page.
4. On the **Create an Add-On** page, select **OOM-Guard**.
5. Click **Complete** to install the component.

TCR Introduction

Last updated：2024-02-05 16:00:10

Overview

Add-on Description

TCR Addon is a plug-in provided by the Tencent Container Registry (TCR) service for private-network and Secret-free pulling of container images. After this plug-in is installed in a TKE cluster, cluster nodes can pull container images from Enterprise Edition instances over the private network, without the need for explicit configuration of ImagePullSecret in the cluster resource YAML file. This plug-in can accelerate image pulling in TKE clusters and simplify image configuration.

Note:

The TKE cluster version must be v1.10.x or later. We recommend that you use this add-on in TKE v1.12.x or later.

The startup parameters of the Kubernetes `controller manager` component must contain `authentication-kubeconfig` and `authorization-kubeconfig` (enabled by default in TKE v.12.x).

Kubernetes objects deployed in a cluster

Name	Type	Resource Amount	Namespace
tcr-assistant-system	Namespace	1	-
tcr-assistant-manager-role	ClusterRole	1	-
tcr-assistant-manager-rolebinding	ClusterRoleBinding	1	-
tcr-assistant-leader-election-role	Role	1	tcr-assistant-system
tcr-assistant-leader-election-rolebinding	RoleBinding	1	tcr-assistant-system
tcr-assistant-webhook-server-cert	Secret	1	tcr-assistant-system
tcr-assistant-webhook-service	Service	1	tcr-assistant-

			system
tcr-assistant-validating-webhook-configuration	ValidatingWebhookConfiguration	1	tcr-assistant-system
imagepullsecrets.tcr.tencentcloudcr.com	CustomResourceDefinition	1	tcr-assistant-system
tcr.ips*	ImagePullSecret CRD	(2-3)	tcr-assistant-system
tcr.ips*	Secret	(2-3)*{Namespace No.}	tcr-assistant-system
tcr-assistant-controller-manager	Deployment	1	tcr-assistant-system
updater-config	ConfigMap	1	tcr-assistant-system
hosts-updater	DaemonSet	{Node No.}	tcr-assistant-system

Component resource usage

Component	Resource Usage	Instance Quantity
tcr-assistant-controller-manager	CPU：500m memory：512Mi	1
hosts-updater	CPU：100m memory：100Mi	Number of worker nodes

Use Cases

Pulling images without a Secret

For a Kubernetes cluster to pull private images, you must create access credential Secret resources, configure the ImagePullSecret attribute in the YAML file of the resources, and explicitly specify the created Secret. The overall configuration process is complicated, and image pull will fail if the Secret is not configured or the specified Secret is incorrect.
In face of this problem, you can install the TCR add-on in the cluster. The add-on automatically obtains the access credential of the specified TCR Enterprise Edition instance and delivers it to the specified namespace of the TKE cluster. When using the YAML file to create or update resources, you do not need to configure ImagePullSecret. Instead, the cluster automatically uses the delivered access credential to pull images from the TCR Enterprise Edition instance.

Pulling images over the private network

The add-on automatically creates a DaemonSet workload, host-updater, which is used to update the host configuration of nodes in the cluster. Note that this configuration is only recommended for test scenarios. For resolving, you can use the private network linkage provided by TCR, PrivateDNS, or your own DNS service.

Limits

For use cases of secret-free image pulling:
Users must have the permission to obtain the access credential of the specified TCR Enterprise Edition instance, that is, the permission to call the CreateInstanceToken API. We recommend that users with TCR admin permissions configure this add-on.
After the add-on is installed and takes effect, do not repeatedly specify ImagePullSecret in the resource YAML file. Otherwise, nodes may use the incorrect image pull access credential, leading to pull failures.

Component Permission Description

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

Requiring/supporting the feature of password-free pull of images, i.e. proactively managing image credentials (secret) for clients.	Secret	watch, create, update, patch, and delete

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: tcr-assistant-leader-election-role
  namespace: tcr-assistant-system
rules:
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
  - apiGroups:
      - ""
    resources:
      - configmaps/status
    verbs:
      - get
      - update
      - patch
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: tcr-assistant-manager-role
  namespace: tcr-assistant-system
rules:
  - apiGroups:
      - ""
    resources:
      - secrets
    verbs:
      - create
      - delete
      - patch
      - update
      - watch
  - apiGroups:
      - admissionregistration.k8s.io
    resources:
      - validatingwebhookconfigurations
    verbs:
      - create
      - get
      - patch
  - apiGroups:
      - certificates.k8s.io
    resources:
      - certificatesigningrequests
    verbs:
      - create
      - delete
      - get
  - apiGroups:
      - certificates.k8s.io
    resources:
      - certificatesigningrequests/approval
    verbs:
      - update
  - apiGroups: ["certificates.k8s.io"]
    resources:
      - "signers"
    #   # resourceNames:
    #   #   # Support legacy versions, before signerName was added
    #   #   - "kubernetes.io/legacy-unknown"
    verbs:
      - approve
  - apiGroups:
      - ""
    resources:
      - namespaces
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - namespaces/status
    verbs:
      - get
  - apiGroups:
      - ""
    resources:
      - serviceaccounts
    verbs:
      - get
      - list
      - patch
      - update
      - watch
  - apiGroups:
      - ""
    resources:
      - serviceaccounts/status
    verbs:
      - get
      - patch
      - update
  - apiGroups:
      - tcr.tencentcloudcr.com
    resources:
      - imagepullsecrets
    verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
  - apiGroups:
      - tcr.tencentcloudcr.com
    resources:
      - imagepullsecrets/status
    verbs:
      - get
      - patch
      - update
```

Operation step

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** on the Add-on management page.
4. On the **Create an Add-On** page, select **TCR**, and click **Parameter Configuration**.
5. On the **TCR Component Parameter Configuration** page, refer to the following information for configuration:

Select an associated instance: select an existing instance of the TCR Enterprise edition under the currently logged-in account and confirm that the currently logged-in user has permission to create a long-term access credential for the instance. If you need to create a new Enterprise Edition instance, create it in the region where the current cluster is located.

Configure password-free pulling (enabled by default): you can choose to issue the access credential automatically for the current user, or specify the username and password. You can also configure the namespace and ServiceAccount for the desired Secret-free pulling. We recommend that you keep the default configuration to make sure this feature works on the new namespace.

Configure private network resolving (advanced feature): make sure that there is already a private network linkage between the cluster and the associated TCR instance, and enable the private network resolving feature. Note that this configuration is only recommended for test scenarios. For resolving, you can use the private network linkage provided by TCR, orPrivateDNS, or your own DNS service.

6. Click **Complete** to create the component. After the component is created, if you need to modify the related configuration, please delete the component to reconfigure and install.

Note:

When the TCR add-on is deleted, the created dedicated access credential is not automatically deleted. You can go to the TCR console to manually disable or delete the credential.

How It Works

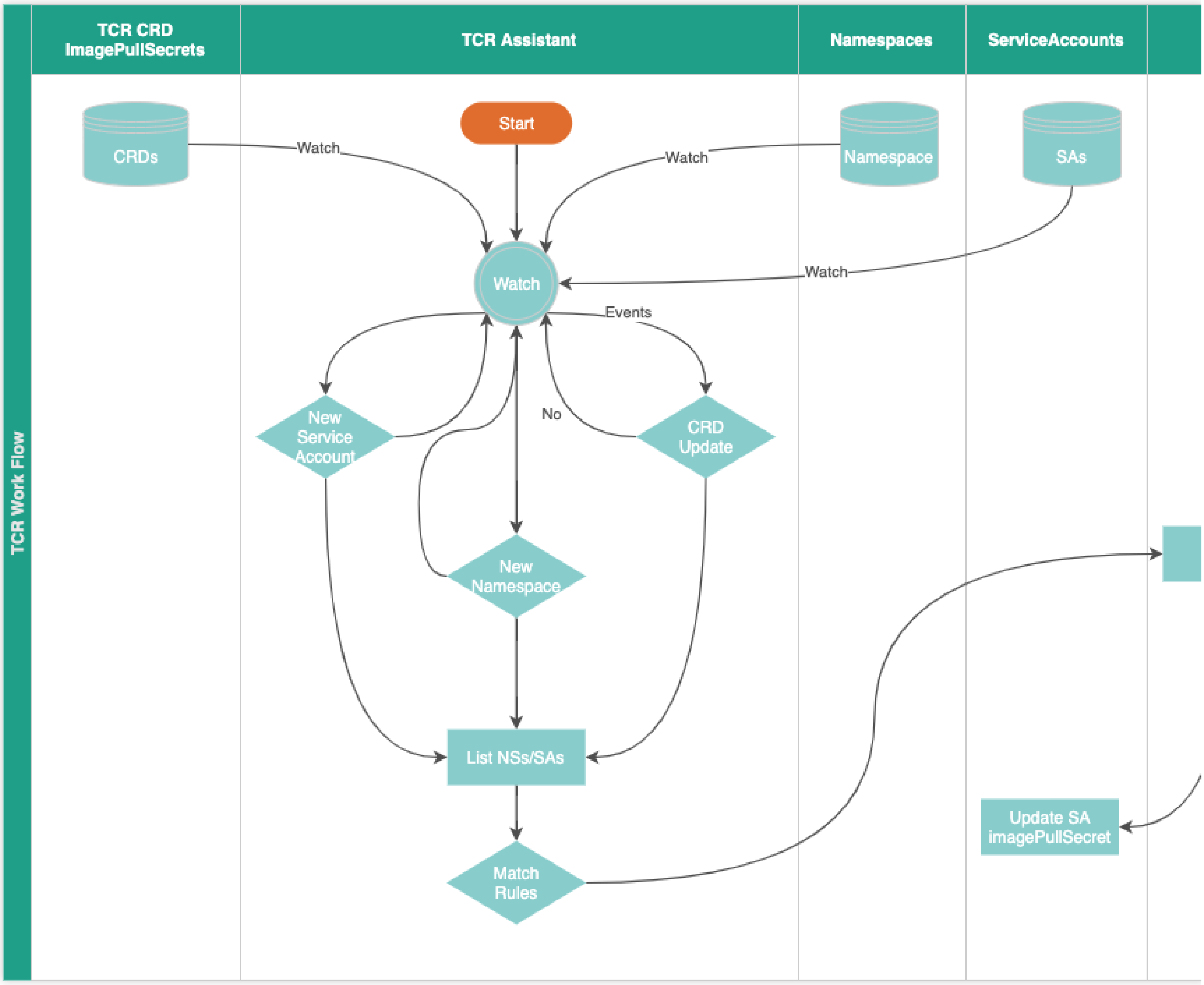
Overview

TCR Assistant helps you implement the auto-process of deploying k8s `imagePullSecret` to any namespace, and associate it with the `ServiceAccount` of the namespace. If you do no **explicitly specify** the `imagePullSecret` and `serviceAccount` when create the workload, K8s will try find the matched `imagePullSecret` from the `ServiceAccount` named `default` under the namespace.

Glossary

Name	Alias	Description
ImagePullSecret	ips, ipss	The CRD defined by TCR Assistant. It’s used to store the username and password of the image repository, and issue the target `Namespace` and `ServiceAccount` .

How it works



TCR Assistant is a classic K8s Operator. Upon deployment of TCR Assistant, the CRD object `imagepullsecrets.tcr.tencentcloudcr.com` is created automatically. This CRD's `kind` is `ImagePullSecret`, and its version is `tcr.tencentcloudcr.com/v1`, with the alias as `ips` or `ipss`. TCR Assistant keeps watching the resource status of `Namespace` and `ServiceAccount` in the cluster. When there are resource changes, it checks whether the changes match the rules set in `ImagePullSecret`. If yes, it automatically deploys the Secret required to pull the **private image repository**. TCR Assistant is usually deployed in a K8s cluster, and accesses K8s master API in `in cluster` mode.

Creating CRD resources

When TCR Assistant is deployed, the Secret used to pull TCR image is not deployed in the target K8s cluster. You need to create `ImagePullSecret` using kubectl or Client Go.

```
# Create ImagePullSecret resource
$ kubectl create -f allinone/imagepullsecret-sample.yaml

imagepullsecret.tcr.tencentcloudcr.com/imagepullsecret-sample created
```

`ImagePullSecret` resource sample file (allinone/imagepullsecret-sample.yaml):

```
apiVersion: tcr.tencentcloudcr.com/v1
kind: ImagePullSecret
metadata:
  name: imagepullsecret-sample
spec:
  namespaces: "*"
  serviceAccounts: "*"
  docker:
    username: "100012345678"
    password: tcr.jwt.token
    server: fanjiankong-bj.tencentcloudcr.com
```

Description of `ImagePullSecret` spec fields:

Field	Description	Remarks
namespaces	`NameSpace` matching rule	Match any namespace: `*` or blank; Match any of multiple namespaces: enter the resource names and separate them with `,` . **Note**: Expressions are not supported. Please enter the exact resource name.
serviceAccounts	`serviceAccounts` matching rule	Match any namespace: `*` or blank; Match any of multiple namespaces: enter the resource names and separate them with `,` . **Note**: Expressions are not supported. Please enter the exact resource name.
docker.server	Image repository domain name	Please enter only the repository domain name
docker.username	Image repository username	Make sure the user has all the required permissions
docker.password	Password of the image repository username	-

After the creation, you can run the following command to check execution result of TCR Assistant:

```
# List ImagePullSecret information
$ kubectl get ipss
NAME                         NAMESPACES    SERVICE-ACCOUNTS    SECRETS-DESIRED    SECRETS-
imagepullsecret-sample       *             *                   10                 10

# Check details
$  kubectl describe ipss
Name:          imagepullsecret-sample
Namespace:
Labels:        <none>
Annotations:   <none>
```

```
API Version:  tcr.tencentcloudcr.com/v1
Kind:         ImagePullSecret
Metadata:
  Creation Timestamp:  2021-12-01T06:47:34Z
  Generation:          1
    Manager:      kubectl-client-side-apply
    Operation:    Update
    Time:         2021-12-01T06:47:34Z
    API Version:  tcr.tencentcloudcr.com/v1
    Manager:         manager
    Operation:       Update
    Time:            2021-12-01T06:47:38Z
  Resource Version:  30389349
  UID:               2109f384-240b-405c-9ce8-73ce938a7c2f
Spec:
  Docker:
    Password:        tcr.jwt.token
    Server:          fanjiankong-bj.tencentcloudcr.com
    Username:        100012345678
  Namespaces:        *
  Service Accounts:  *
Status:
  S As Desired:  47
  S As Success:  1
  Secret Update Successful:
    Namespaced Name:  kube-public/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  devtools/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  demo/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  kube-system/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  tcr-assistant-system/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  kube-node-lease/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  cert-manager/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  default/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:36Z
    Namespaced Name:  afm/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:37Z
    Namespaced Name:  lens-metrics/tcr.ipsimagepullsecret-sample
    Updated At:       2021-12-01T06:47:37Z
  Secrets Desired:    10
  Secrets Success:    10
```

```
    Service Accounts Modify Successful:
      Namespaced Name:  default/default
      Updated At:       2021-12-01T06:47:38Z
  Events:               <none>
```

Note:

If it's necessary to update the `Secret` resource of the TCR Assistant deployment, there's no need to delete and rebuild the `ImagePullSecret` resource. Simply editing the `docker.username` and `docker.password` fields will make it effective. For instance:

```
$ kubectl edit ipss imagepullsecret-sample
```

Namespace updates

When TCR Assistant detects new K8s `Namespace` , it checks whether the name of the resource matches the `namespaces` field of `ImagePullSecret` . If the names are not matched, it goes to the next step. If the names are matched, K8s API is invoked to create a `Secret` resource, and the `Secret` name is added to the `imagePullSecrets` of `ServiceAccount` . See below for examples:

```
# Check the Secret automatically deployed under newns
```

```
$ kubectl get secrets -n newns
NAME                            TYPE                                  DATA   AGE
tcr.ipsimagepullsecret-sample   kubernetes.io/dockerconfigjson        1      7m2s
default-token-nb5vw             kubernetes.io/service-account-token   3      7m2s

# Check the Secret automatically associated with the `ServiceAccount` resource name
$ kubectl get serviceaccounts default -o yaml -n newns
apiVersion: v1
imagePullSecrets:
- name: tcr.ipsimagepullsecret-sample
kind: ServiceAccount
metadata:
  creationTimestamp: "2021-12-01T07:09:56Z"
  name: default
  namespace: newns
  resourceVersion: "30392461"
  uid: 7bc67144-3685-4666-ba41-b1447bbbaa38
secrets:
- name: default-token-nb5vw
```

ServiceAccount updates

When TCR Assistant detects new K8s `ServiceAccount` , it checks whether the name of the resource matches the `serviceAccounts` field of `ImagePullSecret` . If the names are **not matched**, it goes to the next step. If the names are matched, K8s API is invoked to create or update `Secret` resource, and the `Secret` name is added to the `imagePullSecrets` field of `ServiceAccount` . See below for examples:

```
# Create ServiceAccount resource under newns
$ kubectl create sa kung -n newns
serviceaccount/kung created

# Check the Secret automatically associated with the newly-created `ServiceAccount`
$ kubectl get serviceaccounts kung -o yaml -n newns
apiVersion: v1
imagePullSecrets:
- name: tcr.ipsimagepullsecret-sample
kind: ServiceAccount
metadata:
  creationTimestamp: "2021-12-01T07:19:12Z"
  name: kung
  namespace: newns
  resourceVersion: "30393760"
  uid: e236829e-d88e-4feb-9e80-5e4a40f2aea2
secrets:
- name: kung-token-fljt8
```

TCR Hosts Updater

Last updated：2021-12-02 15:01:15

[TOC]

Overview

TCR Hosts Updater, a component of TCR Addon, helps you deploy the hosts file on K8s cluster worker nodes in regions without VPC DNS service.

How It Works

Hosts Updater mounts a specific `ConfigMaps` in the K8s cluster as the `Volume` of a worker node, watches the changes of the config file via inotify (inode notify) of Linux OS, and updates the `/etc/hosts` file of the worker node accordingly.
As we need to edit the `/etc/hosts` file of the worker node, `Daemonset` workload is usually used for deployment, and it should be run on each node that you need to update hosts. To allow the DaemonSet workload runs on nodes in the cluster as many as possible, the following tolerations are set in the YAML file of the DaemonSet:

```
tolerations:
- key: node-role.kubernetes.io/master
effect: NoSchedule
- key: node.kubernetes.io/disk-pressure
effect: NoSchedule
- key: node.kubernetes.io/memory-pressure
effect: NoSchedule
- key: node.kubernetes.io/network-unavailable
effect: NoSchedule
- key: node.kubernetes.io/not-ready
effect: NoExecute
- key: node.kubernetes.io/pid-pressure
effect: NoSchedule
- key: node.kubernetes.io/unreachable
effect: NoExecute
```

```
- key: node.kubernetes.io/unschedulable
effect: NoSchedule
```

Deployment and Usage

Before the deployment, please create a `ConfigMaps` resource named `updater-config` under `Namespace` of Host Updater (e.g. `kube-system`), and then create `DaemonSet` .

To add or delete entries in hosts, simply edit the `updater-config` . See below for examples:

```
apiVersion: v1
kind: ConfigMap
metadata:
name: updater-config
namespace: kube-system
data:
hosts.yaml: |
- domain: demo.tencentcloudcr.com
ip: 10.0.0.2
disabled: false
- domain: vpc-demo.tencentcloudcr.com
ip: 10.0.0.2
disabled: false
```

Note: As K8s `ConfigMaps Volume` is used, after the modification of `updater-config` , the effective time of the hosts update depends on the `sync period` of worker node kubelet (default to 1 min) and the `Cache TTL` of `ConfigMaps` (default to 1 min). For details, see https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#mounted-configmaps-are-updated-automatically.

DNSAutoscaler

Last updated：2024-02-01 10:17:07

Overview

Add-on Description

DNSAutoscaler is an add-on for DNS horizontal auto scaling. It obtains the number of nodes and cores of a cluster through a Deployment and then automatically scales the number of DNS replicas according to preset scaling policies. Two scaling modes are supported: Linear mode and Ladder mode.

Linear Mode

Sample ConfigMap configuration is as follows:

```
    }
```

```
data:
  linear: |-
    {
      "coresPerReplica": 2,
      "nodesPerReplica": 1,
      "min": 1,
      "max": 100,
      "preventSinglePointFailure": true
    }
```

Formula for calculating the number of target replicas:

replicas = max(ceil(cores × 1/coresPerReplica) , ceil(nodes × 1/nodesPerReplica))

replicas = min(replicas, max)

replicas = max(replicas, min)

Ladder Mode

Sample ConfigMap configuration is as follows:

```
data:
  ladder: |-
```

```
      {
        "coresToReplicas":
        [
          [ 1, 1 ],
          [ 64, 3 ],
          [ 512, 5 ],
          [ 1024, 7 ],
          [ 2048, 10 ],
          [ 4096, 15 ]
        ],
        "nodesToReplicas":
        [
          [ 1, 1 ],
          [ 2, 2 ]
        ]
      }
```

Calculating the quantity of target replicas:

Assume that the above configuration is applied in a cluster with 100 nodes and 400 cores, then: nodesToReplicas = 2 (100>2), coresToReplicas = 3 (64<400<512), the greater value of the two is 3, so replica = 3.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Requirement	Requested Resource	Namespace
tke-dns-autoscaler	Deployment	20 M CPU and 10 Mi memory per node	kube-system
dns-autoscaler	ConfigMap	-	kube-system
tke-dns-autoscale	ServiceAccount	-	kube-system
tke-dns-autoscaler	ClusterRole	-	kube-system
tke-dns-autoscaler	ClusterRoleBinding	-	kube-system

Limits

The add-on supports only clusters with Kubernetes version 1.8 and later.

The workload of the DNS server in the cluster should be Deployment or CoreDNS.

Notes

During CoreDNS horizontal scaling, some CoreDNS replicas may be unavailable for a period of time. We recommend that you optimize related configurations to maximize the DNS service availability. For more information, see Configuring Smooth Upgrade.

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
Monitoring changes in node resources within the cluster	node	list/watch
Modifying the number of coredns replicas deployed by the deployment	replicationcontrollers/scale, deployments/scale, and replicasets/scale	get/update
Retrieving parameter configurations from the configmap. In the absence of configured parameters, a configmap with default parameters will be created.	configmap	get/create

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: tke-dns-autoscaler
rules:
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - list
      - watch
  - apiGroups:
      - ""
    resources:
      - replicationcontrollers/scale
    verbs:
      - get
      - update
  - apiGroups:
      - extensions
      - apps
    resources:
      - deployments/scale
      - replicasets/scale
    verbs:
      - get
      - update
  - apiGroups:
      - ""
    resources:
      - configmaps
    verbs:
      - get
      - create
```

How to Use

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on management** to go to the **Add-on list** page.
4. On the **Add-on list** page, click **Create**. On the **Create add-on** page, select **DNSAutoscaler**.

The default scaling configuration of this add-on is as follows:

```
data:
  ladder: |-
    {
      "coresToReplicas":
      [
        [ 1, 1 ],
        [ 128, 3 ],
        [ 512, 4 ]
      ],
      "nodesToReplicas":
      [
          [ 1, 1 ],
          [ 2, 2 ]
        ]
      }
```

After the add-on is created successfully, you can modify its configuration by modifying `configmap/tke-dns-autoscaler` under the kube-system namespace. For more information about the configuration, see the official documentation.

5. Click **Done**.

NodeProblemDetectorPlus Add-on

Last updated：2024-02-01 10:15:37

Overview

Add-on description

Node-Problem-Detector-Plus is an add-on that monitors the health status of Kubernetes cluster nodes. It runs in the TKE environment as a DaemonSet to help users detect various exceptions on nodes in real time and report the detection results to the upstream Kube-apiserver.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Resource Amount	Namespaces
node-problem-detector	DaemonSet	0.5C 80M	kube-system
node-problem-detector	ServiceAccount	-	kube-system
node-problem-detector	ClusterRole	-	-
node-problem-detector	ClusterRoleBinding	-	-

Use Cases

Node-Problem-Detector-Plus can be used to monitor the running status of nodes, including kernel deadlocks, OOM, system thread pressure, system file descriptor pressure, and other metrics. It reports such information to the API Server as Node Conditions and Events.
You can estimate the resource pressure of nodes by detecting the corresponding metrics and then manually release or scale out node resources before nodes start draining pods. In this way, you can prevent potential losses resulted from Kubernetes resource repossessing or node unavailability.

Limits

To use NPD in your cluster, you need to install this add-on in your cluster. The system resources used by NPD containers is restricted to 0.5 CPU core and 80 MB memory.

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
It is required to report fault information when a node encounters a malfunction and modify its condition.	nodestatus	patch
It is required to send event notifications to the cluster.	event	create/patch/update

Permission Definition

```
rules:
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - ""
  resources:
  - nodes/status
  verbs:
  - patch
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - patch
  - update
```

Usage

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the “**Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on Management** to go to the **Add-on List** page.
4. On the **Add-on List** page, click **Create** to go to the **Create Add-on** page, and select **NodeProblemDetectorPlus**.
5. Click **Complete**. After the installation is successful, the corresponding node-problem-detector resources are available in your cluster, and the corresponding conditions will be added to Node Conditions.

Appendix

Node Conditions

After the NPD plug-in is installed, the following specific Conditions will be added to nodes:

Condition	Default Value	Description
ReadonlyFilesystem	False	Indicates whether the file system is read-only.
FDPressure	False	Queries whether the number of file descriptors of the host reaches 80% of the max value.
FrequentKubeletRestart	False	Indicates whether Kubelet has restarted more than 5 times in 20 minutes.
CorruptDockerOverlay2	False	Indicates whether the DockerImage is faulty.
KubeletProblem	False	Indicates whether the Kubelet service is Running.
KernelDeadlock	False	Indicates whether a deadlock exists in the kernel.

FrequentDockerRestart	False	Indicates whether Docker has restarted more than 5 times in 20 minutes.
FrequentContainerdRestart	False	Indicates whether Containerd has restarted more than 5 times in 20 minutes.
DockerdProblem	False	Indicates whether the Docker service is Running (if the node runtime is Containerd, the value is always False).
ContainerdProblem	False	Indicates whether the Containerd service is Running (if the node runtime is Docker, the value is always False).
ThreadPressure	False	Indicates whether the current number of threads of the system reaches 90% of the max value.
NetworkUnavailable	False	Indicates whether the NTP service status is Running.
SerfFailed	False	Detects the node network health status in distributed mode.

NodeLocalDNSCache

Last updated：2022-04-06 10:29:27

Overview

Add-on Description

NodeLocal DNSCache runs on cluster nodes in the form of a DaemonSet and as a DNS cache proxy to enhance the DNS performance of clusters. In the current system architecture, pods in ClusterFirst DNS mode can connect to kube-dns serviceIP to perform DNS query and be converted to kube-dns/CoreDNS endpoints according to the iptables rules added by kube-proxy. In this new architecture, pods can access the DNS cache proxy running on the same node to eliminate the need of configuring iptables DNAT rules and connection tracking. The local cache proxy queries the kube-dns service to retrieve the cache loss of the cluster host name (suffixed with cluster.local by default).

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Requested Resource	Namespace
node-local-dns	DaemonSet	Per node: 50M CPU and 5Mi memory	kube-system
kube-dns-upstream	Service	-	kube-system
node-local-dns	ServiceAccount	-	kube-system
node-local-dns	Configmap	-	kube-system

Limits

- This add-on is supported only by Kubernetes 1.14 or later versions.
- VPC-CNI supports both the iptables and IPVS modes of kube-proxy. GlobalRouter only supports the iptables mode, and in order for it to support the IPVS mode, the kubelet parameters need to be changed. For more information, see Using NodeLocal DNSCache in Kubernetes clusters.
- For relevant names and labels for which the workloads corresponding to the DNS service have not been adjusted since cluster creation, check that the following workloads related to the DNS service exist under the kube-system namespace of the cluster:
  - service/kube-dns

  - deployment/kube-dns or deployment/coredns, with the "k8s-app: kube-dns" label

- For self-deployed clusters in IPVS mode, make sure that the add-pod-eni-ip-limit-webhook ClusterRole has the following permissions:

```
- apiGroups:
- ""
resources:
```

- configmaps
  - secrets
  - namespaces
  - services
  verbs:

-
  list
  - watch
  - get
  - create
  - update
  - delete
  - patch

- For self-deployed and managed clusters in IPVS mode, make sure that the version of the add-pod-eni-ip-limit-webhook Deployment image under the tke-eni-ip-webhook namespace is greater than or equal to v0.0.6.

Recommended Configuration

After installing NodeLocal DNSCache, we recommend you add the following configuration to CoreDNS:

```
template ANY HINFO . {
rcode NXDOMAIN
}
forward . /etc/resolv.conf {
```

```
prefer_udp
}
```

Directions

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on Management** to go to the **Add-on List** page.
4. On the **Add-On List** page, select **Create**. On the **Create Add-on** page, select NodeLocalDNSCache. For detailed configurations of NodeLocalDNSCache, see Using NodeLocal DNSCache in Kubernetes clusters.
5. Click **Done** to complete the process.

Network Policy

Last updated：2024-02-01 10:16:11

Overview

Add-on description

Network Policy is a resource provided by Kubernetes for defining pod-based network isolation policies. It describes whether a group of pods can communicate with other groups of pods and other network entities. This add-on provides a controller for implementing resources of this type. You can use this add-on if you want to control the network traffic of specific applications at the IP address or port layer (layer 3 or layer 4 of OSI).

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Requested Resource	Namespace
networkpolicy	DaemonSet	Each instance: CPU: 250m, Memory: 250Mi	kube-system
networkpolicy	ClusterRole	-	kube-system
networkpolicy	ClusterRoleBinding	-	kube-system
networkpolicy	ServiceAccount	-	kube-system

Component Permission Description

Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.
Access to the namespaces, pods, services, nodes, endpoints, and networkpolicies within the cluster is required, thus necessitating list/get/watch permission.

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: networkpolicy
rules:
  - apiGroups:
      - ""
    resources:
      - namespaces
      - pods
      - services
      - nodes
      - endpoints
    verbs:
      - list
      - get
      - watch
  - apiGroups:
      - "networking.k8s.io"
    resources:
      - networkpolicies
    verbs:
      - list
      - get
      - watch
  - apiGroups:
      - extensions
    resources:
      - networkpolicies
    verbs:
      - get
      - list
      - watch
```

Directions

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the “**Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on Management** to go to the **Add-on List** page.
4. On the **Add-on List** page, click **Create** and select **NetworkPolicy** in the pop-up **Create Add-on** window. For details of NetworkPolicy configuration, see Best Practices for Network Policy.
5. Click **Done**.

DynamicScheduler

Last updated：2022-09-26 16:18:52

Note

TPS was deactivated on May 16, 2022. For more information, see Notice on TPS Discontinuation on May 16, 2022 at 10:00 (UTC +8). The new Prometheus service will be provided by TMP.

If your Dynamic Scheduler uses TPS as the data source and you don't change it, the Dynamic Scheduler will become invalid. To use TMP as the data source, you need to upgrade the Dynamic Scheduler before associating it with a TMP instance, as TMP adds API authentication capabilities.

If your Dynamic Scheduler uses the self-built Prometheus service, it will not be affected by the TPS deactivation, but you need to guarantee the stability and reliability of the self-built Prometheus service.

Overview

Add-on description

The Dynamic Scheduler is a dynamic scheduler provided by TKE for pre-selection and preferential selection based on actual node loads. It is implemented based on the native kube-scheduler extender mechanism of Kubernetes. After being installed in a TKE cluster, this add-on will work with the kube-scheduler to effectively prevent node load imbalances caused by the native scheduler through the request and limit scheduling mechanisms.

This add-on relies on the Prometheus add-on and rule configuration. We recommend you follow the instructions in Deploying dependencies; otherwise, the add-on may not work properly.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Requested Resource	Namespace
node-annotator	Deployment	100 MB CPU and 100 MiB MEM for each instance; one instance in total	kube-system
dynamic-scheduler	Deployment	400 MB CPU and 200 MiB MEM for each instance; three instances in total	kube-system
dynamic-scheduler	Service	-	kube-system

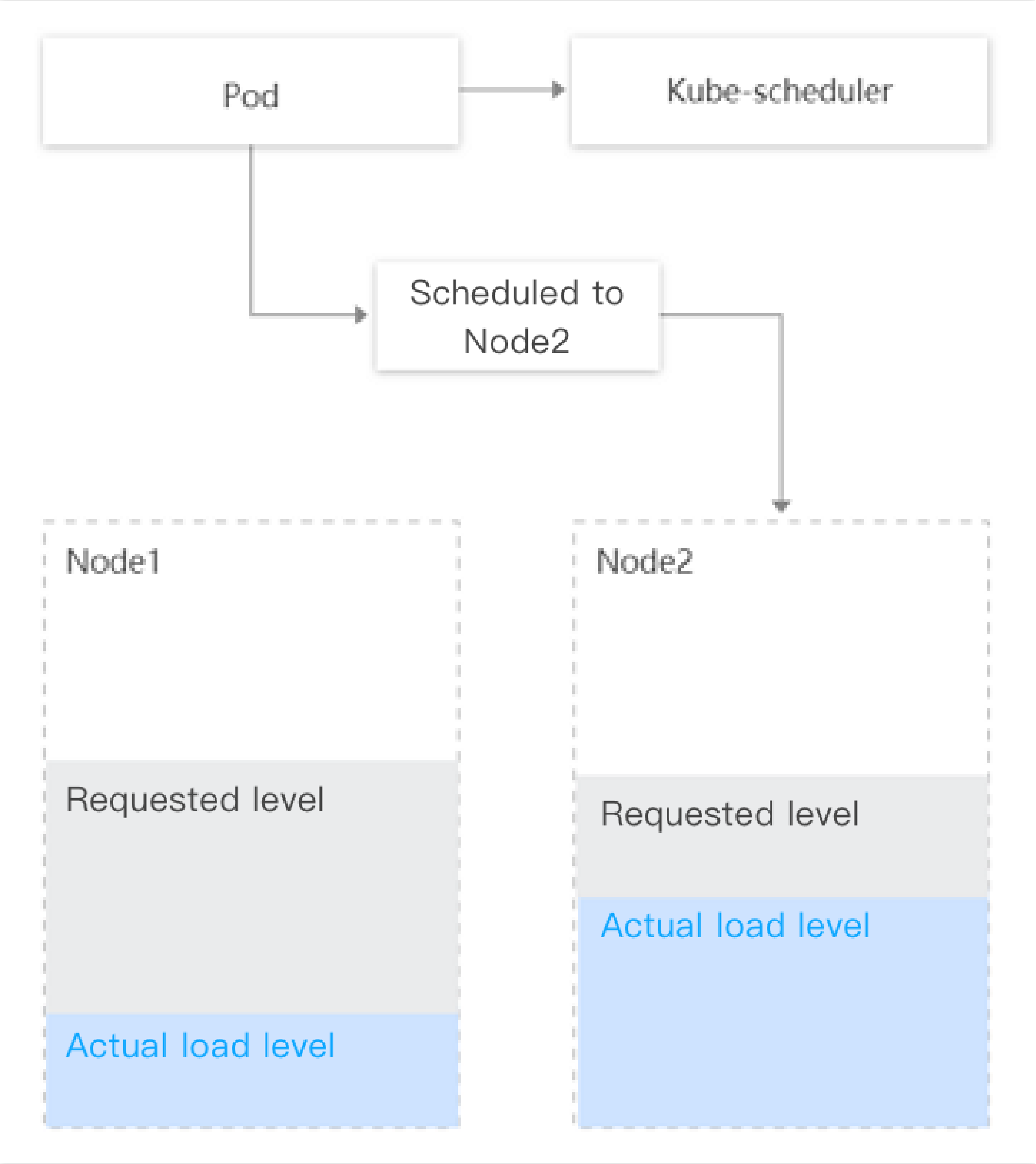
Kubernetes Object Name	Type	Requested Resource	Namespace
node-annotator	ClusterRole	-	kube-system
node-annotator	ClusterRoleBinding	-	kube-system
node-annotator	ServiceAccount	-	kube-system
dynamic-scheduler-policy	ConfigMap	-	kube-system
restart-kube-scheduler	ConfigMap	-	kube-system
probe-prometheus	ConfigMap	-	kube-system

Use Cases

Uneven cluster loads

Most of Kubernetes' native schedulers rely on Pod request resources for scheduling, which means that they cannot make decisions based on the actual node loads in the current and past periods of time and may cause the following problems:
A large number of remaining resources on some nodes of the cluster can be scheduled (value calculated based on the request and limit values of the running Pods on the nodes), but the actual loads are high; on other nodes, a small number of remaining resources can be scheduled, but the actual loads are low. In this case, the kube-scheduler will preferentially schedule Pods to nodes with more remaining resources (based on the `LeastRequestedPriority` policy).

The kube-scheduler will schedule the Pod to node 2, despite the fact that node 1 with a lower actual load level is a better choice.



Avoiding scheduling hotspots

To avoid continuing to schedule Pods from low-load nodes, the Dynamic Scheduler supports a policy to avoid scheduling hotspots, that is, to collect the number of Pods scheduled in the past few minutes and lower the node's

score during preferential selection.
The current policy is as follows:

- If more than two Pods are scheduled to the node in the past minute, the node's score for preferential selection is decreased by 1.
- If more than five Pods are scheduled to the node in the past five minutes, the node's score for preferential selection is decreased by 1.

Risk Control

- This add-on has been interconnected to TKE's monitoring and alarming system.
- We recommend you enable event persistence for the cluster to better monitor the add-on for exceptions and locate the problems.
- Uninstalling the add-on will only delete the scheduling logic of the Dynamic Scheduler and will not affect the scheduling feature of the native kube-scheduler.

Limits

- The TKE is on v1.10.x or later.
- If you need to upgrade the Kubernetes master version:
  - For a managed cluster, you don't need to set the add-on again.
  - For a self-deployed cluster, master version upgrade will reset the configurations of all the add-ons in the master, which affects the configuration of the Dynamic Scheduler add-on as a scheduler extender. Therefore, you need to uninstall the Dynamic Scheduler and install it again.

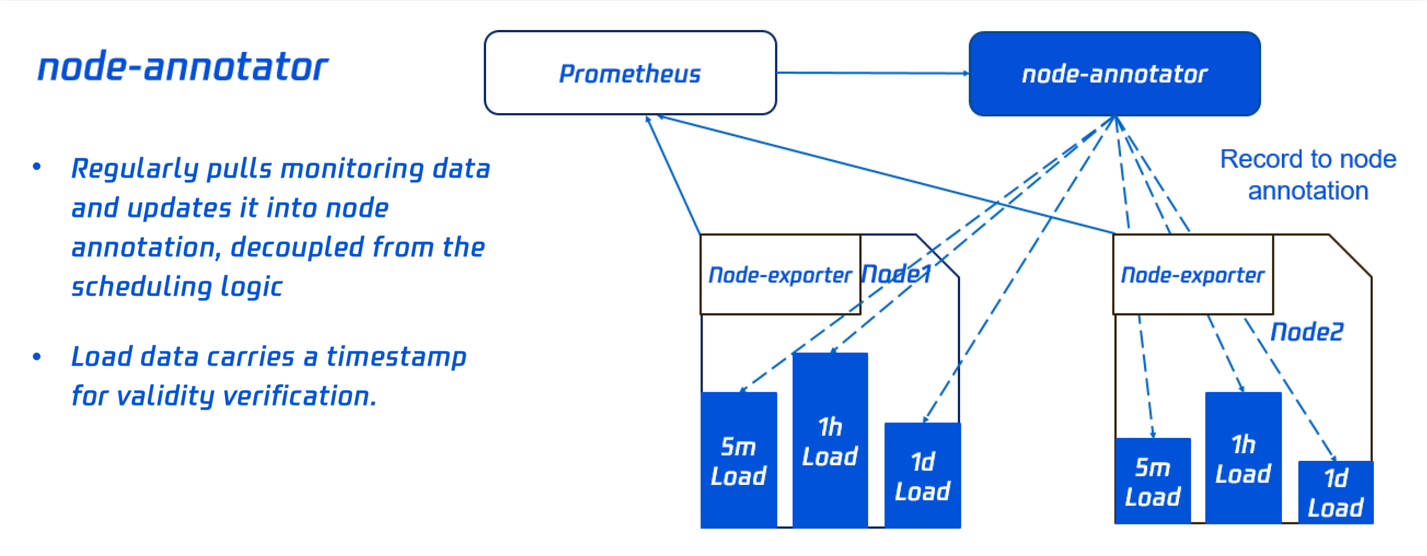
How It Works

The Dynamic Scheduler is based on the scheduler extender mechanism to get the node load from the Prometheus data. It adopts a scheduling policy based on the actual node load and performs intervention during pre-selection and preferential selection, so that Pods are preferentially scheduled to low-load nodes. This add-on consists of node-annotator and the dynamic scheduler.

node-annotator

node-annotator is responsible for regularly pulling the metrics of the node load from the monitoring data and sync them to the annotation of the node.

> Note：
> After the add-on is deleted, the annotation generated by node-annotator will not be cleared automatically and needs to be cleared manually.



node-annotator

- Regularly pulls monitoring data and updates it into node annotation, decoupled from the scheduling logic
- Load data carries a timestamp for validity verification.

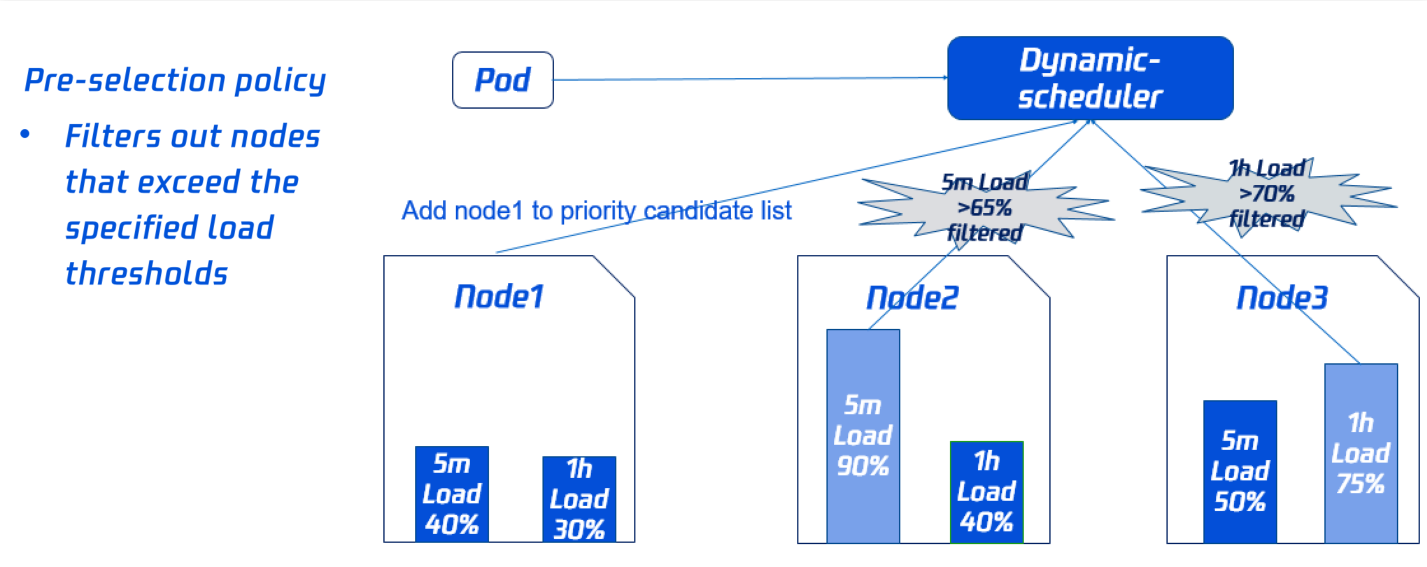
Dynamic scheduler

The dynamic scheduler is a scheduler extender that filters and scores the nodes during pre-selection and preferential selection based on the load data of the node annotation.

Pre-selection policy

To avoid scheduling Pods to high-load nodes, you need to filter out some high-load nodes during pre-selection. You can dynamically configure the filter policy and ratio as instructed in Add-On Parameter Description.

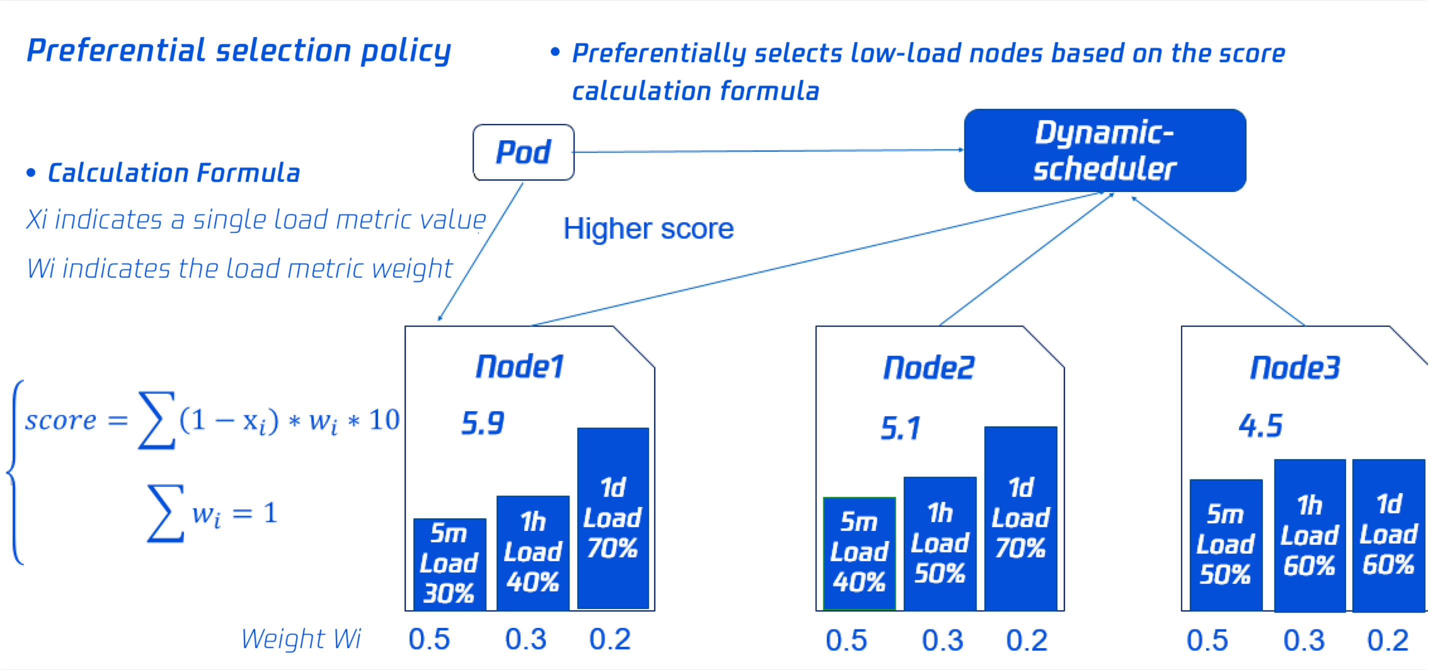
As both node 2's load in the past five minutes and node 3's load in the past hour exceed the threshold, they will not be included in preferential selection.



Preferential selection policy

To balance the loads on each node in the cluster, the dynamic scheduler will score the nodes based on their load data. The lower the load, the higher the score.

Node 1 with the highest score will be preferentially selected for scheduling. You can dynamically configure the scoring policy and weights as instructed in Add-On Parameter Description.



$$\begin{cases} score = \sum(1 - x_i) * w_i * 10 \\ \sum w_i = 1 \end{cases}$$

Add-On Parameter Description

Prometheus data query address

Note：

- To ensure that the required monitoring data can be pulled by the add-on and the scheduling policy can take effect, follow the Configuring the Prometheus rule step in Deploying dependencies to configure the monitoring data collection rules.
- Default values have been set for the pre-selection and preferential selection parameters. If you have no special requirements, you can directly use them.

- If you use the self-built Prometheus service, just enter the data query URL (HTTPS/HTTPS).
- If you use the managed Prometheus service, just select the managed instance ID, and the system will automatically parse the data query URL of the instance.

Pre-selection parameters

Default Value of the Pre-selection Parameter	Description
Average **CPU** utilization threshold in five minutes	If the **average** CPU utilization of the node in the past five minutes exceeds the configured threshold, no Pods will be scheduled to the node.
Maximum **CPU** utilization threshold in an hour	If the **maximum** CPU utilization of the node in the past hour exceeds the configured threshold, no Pods will be scheduled to the node.
Average **memory** utilization threshold in five minutes	If the **average** memory utilization of the node in the past five minutes exceeds the configured threshold, no Pods will be scheduled to the node.
Maximum **memory** utilization threshold in an hour	If the **maximum** memory utilization of the node in the past hour exceeds the configured threshold, no Pods will be scheduled to the node.

Preferential selection parameters

Default Value of the Preferential Selection Parameter	Description
Average **CPU** utilization weight in five minutes	The greater the weight, the bigger impact the **average** CPU utilization in the past five minutes has on the node score.
Maximum **CPU** utilization weight in an hour	The greater the weight, the bigger impact the **maximum** CPU utilization in the past hour has on the node score.

Default Value of the Preferential Selection Parameter	Description
Maximum **CPU** utilization weight in a day	The greater the weight, the bigger impact the **maximum** CPU utilization in the past day has on the node score.
Average **memory** utilization weight in five minutes	The greater the weight, the bigger impact the **average** memory utilization in the past five minutes has on the node score.
Maximum **memory** utilization weight in an hour	The greater the weight, the bigger impact the **maximum** memory utilization in the past hour has on the node score.
Maximum **memory** utilization weight in a day	The greater the weight, the bigger impact the **maximum** memory utilization in the past day has on the node score.

Directions

Deploying dependencies

The Dynamic Scheduler relies on the actual node loads in the current and past periods of time to make scheduling decisions. It needs to get the information of the actual node loads of the system through the Prometheus add-on. Before using the Dynamic Scheduler, you need to deploy the Prometheus add-on. In the TKE, you can use the self-built Prometheus monitoring service or the cloud native monitoring service.

- Self-built Prometheus monitoring service
- Prometheus monitoring service

Deploying the Node Exporter and Prometheus

You can deploy the Node Exporter and Prometheus as needed to monitor node metrics through the Node Exporter.

Configuring aggregation rules

After getting the node monitoring data from the Node Exporter, you need to aggregate and calculate the data collected in the native Node Exporter through Prometheus. To get metrics such as `cpu_usage_avg_5m` , `cpu_usage_max_avg_1h` , `cpu_usage_max_avg_1d` , `mem_usage_avg_5m` , `mem_usage_max_avg_1h` , and `mem_usage_max_avg_1d` required by the Dynamic Scheduler, you need to configure `rules` in Prometheus as follows:

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
name: example-record
```

```
spec:
groups:
- name: cpu_mem_usage_active
interval: 30s
rules:
- record: cpu_usage_active
expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[30s]))
* 100)
- record: mem_usage_active
expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
- name: cpu-usage-5m
interval: 5m
rules:
- record: cpu_usage_max_avg_1h
expr: max_over_time(cpu_usage_avg_5m[1h])
- record: cpu_usage_max_avg_1d
expr: max_over_time(cpu_usage_avg_5m[1d])
- name: cpu-usage-1m
interval: 1m
rules:
- record: cpu_usage_avg_5m
expr: avg_over_time(cpu_usage_active[5m])
- name: mem-usage-5m
interval: 5m
rules:
- record: mem_usage_max_avg_1h
expr: max_over_time(mem_usage_avg_5m[1h])
- record: mem_usage_max_avg_1d
expr: max_over_time(mem_usage_avg_5m[1d])
- name: mem-usage-1m
interval: 1m
rules:
- record: mem_usage_avg_5m
expr: avg_over_time(mem_usage_active[5m])
```

Configuring the Prometheus file

1. The above section defines the `rules` to calculate the metrics required by the Dynamic Scheduler. You need to configure the `rules` to Prometheus as a general Prometheus configuration file. Below is a sample:

   ```
   global:
   evaluation_interval: 30s
   scrape_interval: 30s
   external_labels:
   rule_files:
   ```

```
- /etc/prometheus/rules/*.yml # `/etc/prometheus/rules/*.yml` is the defined `r
ules` file.
```

2. Copy the `rules` configurations to a file (such as `dynamic-scheduler.yaml`) and put the file under `/etc/prometheus/rules/` of the above Prometheus container.
3. Load the Prometheus server to get the metrics required by the Dynamic Scheduler from Prometheus.

> Note
>
> In general, the above Prometheus configuration file and `rules` configuration file are stored via a ConfigMap before being mounted to a Prometheus server's container. Therefore, you only need to modify the ConfigMap.

Installing the add-on

1. Log in to the TKE console and select **Cluster** on the left sidebar.
2. On the **Cluster management** page, click the ID of the target cluster to go to the cluster details page.
3. On the left sidebar, click **Add-On Management**.
4. On the **Add-On List** page, select **Create**. On the **Create Add-on** page, select DynamicScheduler (dynamic scheduler).
5. Click **Parameter Configurations** and enter the parameters required by the add-on as instructed in Add-On Parameter Description.
6. Click **Done**. After the add-on is installed successfully, the Dynamic Scheduler can run normally without extra configurations.

DeScheduler

Last updated：2023-05-06 20:04:31

Note

TPS was deactivated on May 16, 2022. For more information, see Notice on TPS Discontinuation on May 16, 2022 at 10:00 (UTC +8). The new Prometheus service will be provided by TMP.

If your DeScheduler uses TPS as the data source and you don't change it, the DeScheduler will become invalid. To use TMP as the data source, you need to upgrade the DeScheduler before associating it with a TMP instance, as TMP adds API authentication capabilities.

If your DeScheduler uses the self-built Prometheus service, it will not be affected by the TPS deactivation, but you need to guarantee the stability and reliability of the self-built Prometheus service.

Overview

Add-on Description

DeScheduler is a plug-in provided by TKE based on the Kubernetes native community of DeScheduler to implement rescheduling based on actual node loads. After being installed in a TKE cluster, this plug-in will take effect in synergy with Kube-scheduler to monitor in real time high-load nodes in the cluster and drain low-priority pods. We recommend that you use it together with the TKE DynamicScheduler add-on to ensure cluster load balancing in multiple dimensions. This add-on relies on the Prometheus add-on and rule configuration. We recommend you read Deploying dependencies carefully before installing it; otherwise, it may not work properly.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Requested Resource	Namespace
descheduler	Deployment	200 MB CPU and 200 MiB MEM for each instance; one instance in total	kube-system
descheduler	ClusterRole	-	kube-system
descheduler	ClusterRoleBinding	-	kube-system
descheduler	ServiceAccount	-	kube-system
descheduler-policy	ConfigMap	-	kube-

			system
probe-prometheus	ConfigMap	-	kube-system

Use Cases

The DeScheduler addresses the unreasonable running of existing nodes in the cluster through rescheduling. The policy of the community Descheduler is implemented based on the data on the API server, but not actual node loads. Therefore, the policy can be adjusted to rescheduling based on actual loads by monitoring nodes.
TKE's `ReduceHighLoadNode` policy relies on Prometheus and Node Exporter monitoring data. Pods are drained and rescheduled based on node metrics such as CPU utilization, memory utilization, network I/O, and system loadavg, thereby avoiding extreme node loads. The `ReduceHighLoadNode` of the DeScheduler needs to be used together with the Dynamic Scheduler's policy based on actual node loads.

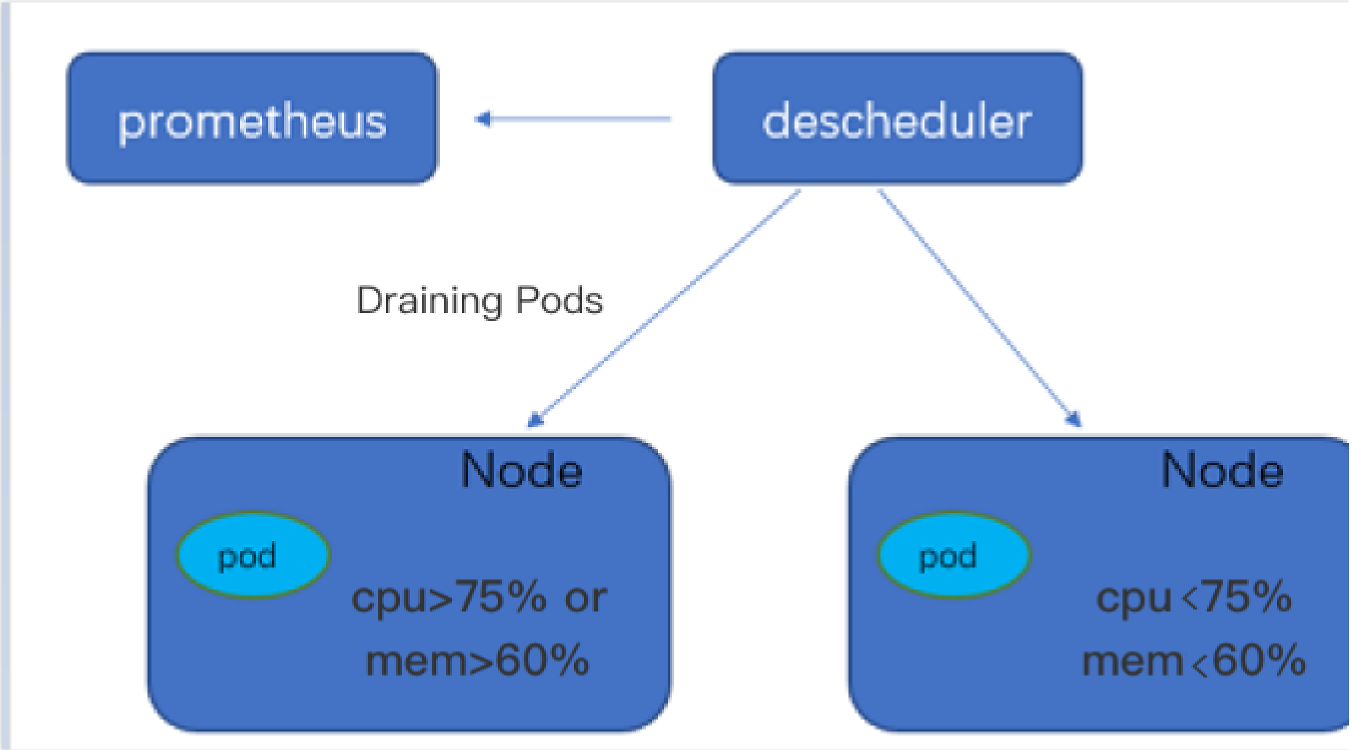
Notes

The Kubernetes is on v1.10.x or later.
In certain cases, some Pods will be scheduled repeatedly to nodes that require rescheduling, which causes Pods to be drained repeatedly. In this case, you can change the nodes to which Pods can be scheduled as needed, or mark the Pods as undrainable.
This add-on has been interconnected to TKE's monitoring and alarming system.
We recommend you enable event persistence for the cluster to better monitor the add-on for exceptions and locate the problems. When the Descheduler drains a Pod, an event will be generated. You can determine whether a Pod is drained repeatedly through the event with the `reason` of "Descheduled".
To prevent the DeScheduler from draining critical Pods, the algorithm is designed not to drain Pods by default. For a Pod that can be drained, its workload needs to be displayed and determined. For StatefulSet and Deployment objects, you can set annotations indicating that Pods can be drained.
Preconditions for Pod draining: To prevent drained Pods from running out of resources, the cluster should contain at least five nodes, and at least four of them should have a load below the **target utilization**.
Pod draining is a high-risk operation. Please perform Pod draining according to node affinity, taint-related configuration, and Pods' requirements for nodes to prevent situations where no node can be scheduled after Pod draining.
If a large number of Pods are drained, the service may become unavailable. Kubernetes provides native PDB objects to prevent a large number of Pods in a workload from becoming unavailable after the draining API is called, but the PDB configuration needs to be created. TKE's DeScheduler includes a guarantee measure to check whether the

number of Pods prepared by a workload is greater than half of the number of the replicas; if not, the draining API will not be called.

How It Works

The DeScheduler is based on the rescheduling concept of the community Descheduler to scan for running Pods on each node that are not in line with the policy and drain them for rescheduling. The community Descheduler provides some of the policies based on the data on the API server, for example, the `LowNodeUtilization` policy that relies on the request and limit values of the Pod. The data can effectively balance the cluster resource allocation and avoid resource fragmentation. However, the community policy lacks the support for the occupation of actual node resources. Specifically, if the same number of resources are allocated from node A and node B, their peak loads will differ significantly due to differences in CPU and memory usage during actual Pod running.
Therefore, TKE has released the DeScheduler, which monitors the actual node loads at the underlying layer for rescheduling. With node load statistics of the cluster from Prometheus and the configured load threshold, it regularly executes the check rules in the policy and drains Pods from high-load nodes.



Add-on Parameter Description

Prometheus data query address

Notes

To ensure that the required monitoring data can be pulled by the add-on and the scheduling policy can take effect, follow the "Configuring the Prometheus file" step in Deploying dependencies to configure the monitoring data collection rules.

If you use the self-built Prometheus service, just enter the data query URL (HTTP/HTTPS).

If you use the managed Prometheus service, just select the managed instance ID, and the system will automatically parse the data query URL of the instance.

Utilization threshold and target utilization

Notes

A default value has been set for the load threshold parameter. If you have no special requirements, you can directly use it.

If the average CPU or memory utilization of the node in the past five minutes exceeds the configured threshold, the Descheduler will identify the node as a high-load node, execute the logic to drain and reschedule Pods to reduce the load below the target utilization.

Directions

Dependency deployment

The DeScheduler add-on relies on the actual node loads in the current and past periods of time to make scheduling decisions. It needs to get the information of the actual node loads of the system through the Prometheus add-on. Before using the DeScheduler add-on, you can use the self-built Prometheus monitoring service or the TKE cloud native monitoring service.

Self-built Prometheus

TMP

Deploying the Node Exporter and Prometheus

You can deploy the Node Exporter and Prometheus as needed to monitor node metrics through the Node Exporter.

Aggregation rule configuration

After getting the node monitoring data from the Node Exporter, you need to aggregate and calculate the data collected in the native Node Exporter through Prometheus. To get metrics such as `cpu_usage_avg_5m` and `mem_usage_avg_5m` required by the DeScheduler, you need to configure `rules` in Prometheus. Below is a sample:

```
groups:
  - name: cpu_mem_usage_active
    interval: 30s
    rules:
    - record: mem_usage_active
      expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
  - name: cpu-usage-1m
    interval: 1m
    rules:
    - record: cpu_usage_avg_5m
      expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m
  - name: mem-usage-1m
    interval: 1m
    rules:
    - record: mem_usage_avg_5m
      expr: avg_over_time(mem_usage_active[5m])
```

Notes

When using TKE's DynamicScheduler, you need to configure the aggregation rules in Prometheus to get node monitoring data. As some of the DynamicScheduler's aggregation rules are identical to those of the DeScheduler, do not overlap rules during configuration. In addition, you should configure the following rules to use the Dynamic Scheduler together with the DeScheduler:

```
groups:
  - name: cpu_mem_usage_active
    interval: 30s
    rules:
    - record: mem_usage_active
      expr: 100*(1-node_memory_MemAvailable_bytes/node_memory_MemTotal_bytes)
  - name: mem-usage-1m
    interval: 1m
    rules:
    - record: mem_usage_avg_5m
      expr: avg_over_time(mem_usage_active[5m])
  - name: mem-usage-5m
    interval: 5m
    rules:
    - record: mem_usage_max_avg_1h
      expr: max_over_time(mem_usage_avg_5m[1h])
    - record: mem_usage_max_avg_1d
      expr: max_over_time(mem_usage_avg_5m[1d])
  - name: cpu-usage-1m
    interval: 1m
    rules:
    - record: cpu_usage_avg_5m
      expr: 100 - (avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m
  - name: cpu-usage-5m
    interval: 5m
    rules:
    - record: cpu_usage_max_avg_1h
      expr: max_over_time(cpu_usage_avg_5m[1h])
    - record: cpu_usage_max_avg_1d
      expr: max_over_time(cpu_usage_avg_5m[1d])
```

Configuring the Prometheus file

1. The above section defines the `rules` to calculate the metrics required by the DeScheduler. You need to configure the `rules` to Prometheus as a general Prometheus configuration file. Below is a sample:

```
global:
   evaluation_interval: 30s
   scrape_interval: 30s
   external_labels:
rule_files:
- /etc/prometheus/rules/*.yml # `/etc/prometheus/rules/*.yml` is the defined `rules
```

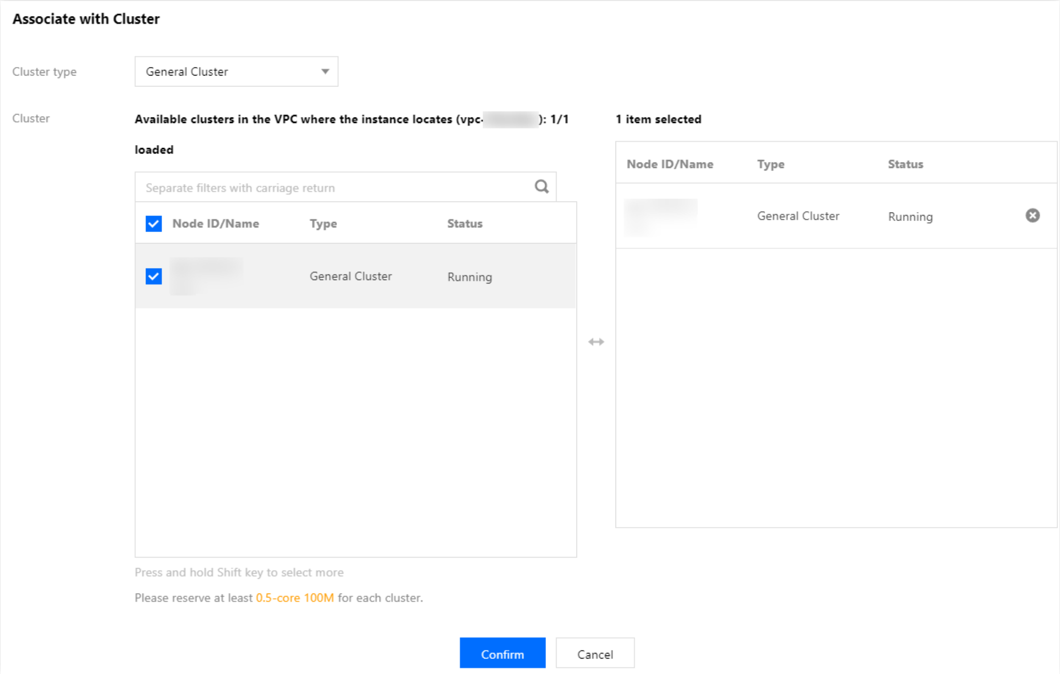
2. Copy the `rules` configurations to a file (such as `de-scheduler.yaml`) and put the file under `/etc/prometheus/rules/` of the above Prometheus container.

3. Reload the Prometheus server to get the metrics required by the Dynamic Scheduler from Prometheus.

Note

In general, the above Prometheus configuration file and `rules` configuration file are stored via a ConfigMap before being mounted to a Prometheus server's container. Therefore, you only need to modify the ConfigMap.

1. Log in to the TKE console and select Prometheus Monitoring in the left sidebar.
2. Create a Prometheus instance under the same VPC as the target cluster, and associate it with the cluster. See the figure below:



Associate with Cluster

Cluster type: General Cluster

Cluster: Available clusters in the VPC where the instance locates (vpc-[redacted]): 1/1 loaded

Separate filters with carriage return

☑ Node ID/Name	Type	Status
☑ [redacted]	General Cluster	Running

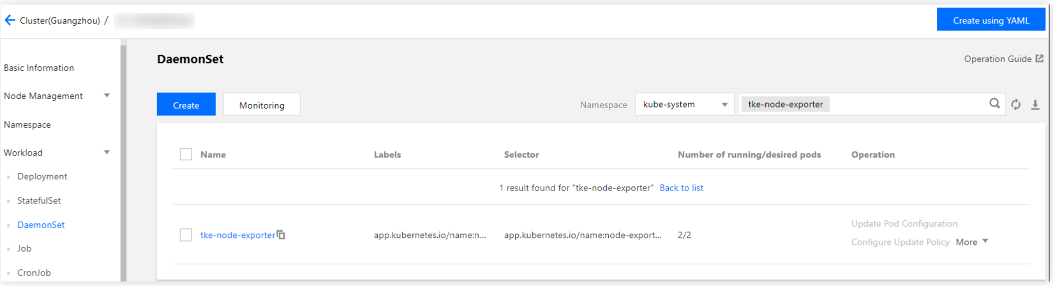
1 item selected

Node ID/Name	Type	Status
[redacted]	General Cluster	Running

Press and hold Shift key to select more

Please reserve at least 0.5-core 100M for each cluster.

3. After associating with the native managed cluster, you can see that the Node Exporter has been installed on each node of the cluster.



Cluster(Guangzhou) / [redacted]

DaemonSet

Namespace: kube-system | tke-node-exporter

□ Name	Labels	Selector	Number of running/desired pods	Operation
1 result found for "tke-node-exporter" Back to list				
□ tke-node-exporter	app.kubernetes.io/name:n...	app.kubernetes.io/name:node-export...	2/2	Update Pod Configuration Configure Update Policy More

4. Set the Prometheus aggregation rules. The content is the same as that configured in Self-built Prometheus monitoring service. The rules take effect immediately after being saved without server reloading.

Installing the add-on

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-On Management**. On the **Add-On List** page, click **Create**.

4. On the **Create Add-on** page, select DynamicScheduler (dynamic scheduler). Click **Parameter Configurations** and enter the parameters required by the add-on as instructed in Add-On Parameter Description.
5. Click **Done**. After the add-on is installed successfully, the DeScheduler can run normally without extra configurations.
6. To drain a workload (such as StatefulSet and Deployment objects), set the following annotation:

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

Nginx-ingress

Last updated：2022-12-12 10:38:32

Introduction

Add-on description

Nginx can be used as a reverse proxy, load balancer, and HTTP buffer. The Nginx-ingress add-on is an Ingress controller for Kubernetes that uses Nginx as a reverse proxy and load balancer. You can deploy and use the Nginx-ingress add-on in your cluster.

Kubernetes objects deployed in a cluster

Deploying the Nginx-ingress add-on in a cluster will deploy the following Kubernetes objects in the cluster:

Kubernetes Object Name	Type	Default Resource Occupation	Namespaces
nginx-ingress	Service	-	Custom
nginx-ingress	Configmap	-	Custom
tke-ingress-nginx-controller-operator	Deployment	0.13-core CPU, 128 MB memory	kube-system
ingress-nginx-controller	Deployment/DaementSet	0.1-core CPU	kube-system
ingress-nginx-controller-hpa	HPA	-	kube-system

Prerequisites

- We recommend that you use Kubernetes 1.16 or later.
- We recommend that you use the TKE node pool feature.
- We recommend that you use Tencent Cloud CLS.

Usage

- Nginx-ingress Overview
- Installing Nginx-ingress

- Using Nginx-ingress Object to Access External Traffic of the Cluster
- Nginx-ingress Log Configuration

HPC

Last updated：2024-02-01 10:16:40

Overview

Add-on description

HorizontalPodCronscaler (HPC) is an add-on to modify the number of replicas of K8S workload. Used in conjunction with HPC CRD resources, it can support scheduled actions in seconds.

Add-on features

Configures "Pod Range" (when the associated object is HPA) or "Desired Number of Pods" (when the associated object is Deployment or StatefulSet).
Sets an “Exceptional Time”. The smallest granularity of the setting is Day. Multiple exceptional times are allowed.
Specifies whether the scheduled task runs only once.

Kubernetes objects deployed in a cluster

Deploying HPC Add-on in a cluster will deploy the following Kubernetes objects in the cluster:

Kubernetes Object	Type	Required Resources	Namesp
horizontalpodcronscalers.autoscaling.cloud.tencent.com	CustomResourceDefinition	-	-
hpc-leader-election-role	Role	-	kube-sy
hpc-leader-election-rolebinding	RoleBinding	-	kube-sy
hpc-manager-role	ClusterRole	-	-
hpc-manager-rolebinding	ClusterRoleBinding	-	-
cronhpa-controller-manager-metrics-service	Service	-	kube-sy
hpc-manager	ServiceAccount	-	kube-sy
tke-hpc-controller	Deployment	100mCPU, 100Mi/pod	kube-sy

Limits

Environment requirements

Note:

If you create a cluster of version 1.12.4 or later, you can use the cluster directly without any parameter changes.
This add-on is supported only by Kubernetes 1.12 or later versions.
The launch parameters of kube-apiserver must be set as follows: `--feature-gates=CustomResourceSubresources=true` .

Node Requirements

The HPC add-on follows the timezone of the associated server. Please make sure that the `/etc/localtime` file exists in the node.
By default, two HPC Pods are installed on different nodes. Therefore at least two nodes are required.

Requirement on resource to be controlled

When you create an HPC resource, please make sure that the workload (K8S resource) to be controlled exists in the cluster.

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
Monitoring changes of horizontalpodcronscalers	horizontalpodcronscalers	create/delete/get/list/patch/watch
The replicas of deployments/statefulsets that required modification	deployments/statefulsets	get/list/patch/watch
Modifying the minReplicas/maxReplicas of horizontalpodautoscalers	horizontalpodautoscalers	get/list/patch/watc
Synchronizing the events of HPC scheduled task execution	events	create/patch

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: null
  name: hpc-manager-role
rules:
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - patch
- apiGroups:
  - apps
  resources:
  - deployments
  verbs:
  - get
  - list
  - patch
  - watch
- apiGroups:
  - apps
  resources:
  - statefulsets
  verbs:
  - get
  - list
  - patch
  - watch
- apiGroups:
  - autoscaling
  resources:
  - horizontalpodautoscalers
  verbs:
  - get
  - list
  - patch
  - watch
- apiGroups:
  - autoscaling.cloud.tencent.com
  resources:
  - horizontalpodcronscalers
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - autoscaling.cloud.tencent.com
  resources:
  - horizontalpodcronscalers/status
  verbs:
  - get
  - patch
  - update
- apiGroups:
  - apiextensions.k8s.io
  resources:
  - customresourcedefinitions
  resourceNames:
  - horizontalpodcronscalers.autoscaling.cloud.tencent.com
  verbs:
  - get
  - list
  - delete
  - watch
```

Directions

Installing HPC

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the “**Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on Management** to go to the **Add-on List** page.
4. On the "Add-on List" page, click **Create**, On the "Create Add-on" page that appears, select **HPC**.
5. Click **Done**.

Examples

Creating a scheduled task resource that associated with Deployment

The example is as follows:

```
apiVersion: autoscaling.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
  name: hpc-deployment
  namespace: default
spec:
  scaleTarget:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
    namespace: default
  crons:
- name: "scale-down"
  excludeDates:
    - "* * * 15 11 *"
    - "* * * * * 5"
  schedule: "30 */1 * * * *"
  targetSize: 1
- name: "scale-up"
  excludeDates:
    - "* * * 15 11 *"
    - "* * * * * 5"
  schedule: "0 */1 * * * *"
  targetSize: 3
```

Creating a scheduled task resource that associated with StatefulSet

The example is as follows:

```
apiVersion: autoscaling.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
  name: hpc-statefulset
  namespace: default
spec:
  scaleTarget:
    apiVersion: apps/v1
    kind: Statefulset
    name: nginx-statefulset
    namespace: default
  crons:
  - name: "scale-down"
    excludeDates:
      - "* * * 15 11 *"
    schedule: "0 */2 * * * *"
    targetSize: 1
  - name: "scale-up"
    excludeDates:
      - "* * * 15 11 *"
    schedule: "30 */2 * * * *"
    targetSize: 4
```

Creating a scheduled task resource that associated with HPA

The example is as follows:

```
apiVersion: autoscaling.cloud.tencent.com/v1
kind: HorizontalPodCronscaler
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: hpc-hpa
spec:
  scaleTarget:
    apiVersion: autoscaling/v1
    kind: HorizontalPodAutoscaler
    name:  nginx-hpa
    namespace: default
  crons:
  - name: "scale-up"
    schedule: "30 */1 * * * *"
    minSize: 2
    maxSize: 6
  - name: "scale-down"
    schedule: "0 */1 * * * *"
    minSize: 1
    maxSize: 5
```

Scheduled duration settings

Field Name	Required	Value Range	Allowed Special Characters
Seconds	Yes	0 - 59	* / , -
Minutes	Yes	0 - 59	* / , -
Hours	Yes	0 - 23	* / , -
Day of month	Yes	1 - 31	* / , - ?
Month	Yes	1 - 12 or JAN - DEC	* / , -
Day of week	Yes	0 - 6 or SUN - SAT	* / , - ?

Description of tke-monitor-agent

Last updated：2024-02-01 10:07:57

Overview

Tencent Cloud upgraded the basic monitoring architecture to improve the stability of the TKE basic monitoring and alarming feature. After the upgrade, a DaemonSet named `tke-monitor-agent` is deployed under the `kube-system` namespace in the cluster, and the K8s resource objects of authentication and authorization are created, including ClusterRole, ServiceAccount, and ClusterRoleBinding. These resource objects are all named `tke-monitor-agent`.

Strengths

This add-on collects the monitoring data of containers, Pods, nodes, and community add-ons. The collected data is used for basic monitoring metrics display, metrics alarming, and metric-based HPA service in the console. By deploying this add-on, you can fix the problem that the monitoring data can't be obtained due to the instability of the basic monitoring service, thereby enjoying more stable monitoring, alarming, and HPA services.

Impact

Deploying this add-on does not affect the normal running of the cluster.

If your **node resources are allocated unreasonably**, **node load is too heavy**, or **node resources are not enough**, deploying the basic monitoring add-on may cause the problem where the Pod corresponding to the `tke-monitor-agent` DaemonSet is in the status of **Pending**, **Evicted**, **OOMKilled** or **CrashLoopBackOff**. The details of the status are as follows:

Pending: The resources on the cluster node are not enough to schedule a Pod. You can schedule the Pod to the node by setting the quantity of requested resources for the `tke-monitor-agent` DaemonSet to `0`. For more information, see Pod Remains in Pending.

Evicted: This status may be caused by insufficient node resources or a heavy load on the node. You can find out the cause and solve the problem in the following ways:

Run `kubectl describe pod -n kube-system <podName>` to check the cause according to the description in the `Message` field.

Run `kubectl describe pod -n kube-system <podName>` to check the cause according to the description in the `Events` field.

CrashLoopBackOff or **OOMKilled**: Run `kubectl describe pod -n kube-system <podName>` to check whether an OOM error occurs. If yes, you can increase the value of `memory limits` , which can't exceed 100 MB. If the error still occurs after the value is set to 100 MB, submit a ticket for assistance.

ContainerCreating: Run `kubectl describe pod -n  kube-system <podName>` to check the `Events` field. If `Failed to create pod sandbox: rpc error: code = Unknown desc = failed to create a sandbox for pod "<pod name >": Error response from daemon: Failed to set projid for /data/docker/overlay2/xxx-init: no space left on device` is displayed, the container data disk is full, and you can clear the data disk to restore it.

Note:

If the problem persists, submit a ticket for assistance.

Quantity of resources consumed in each Pod managed by the DaemonSet (named `tke-monitor-agent`) is positively correlated with the number of Pods and containers running on the node. Below is a sample stress test with low MEM and CPU usage:

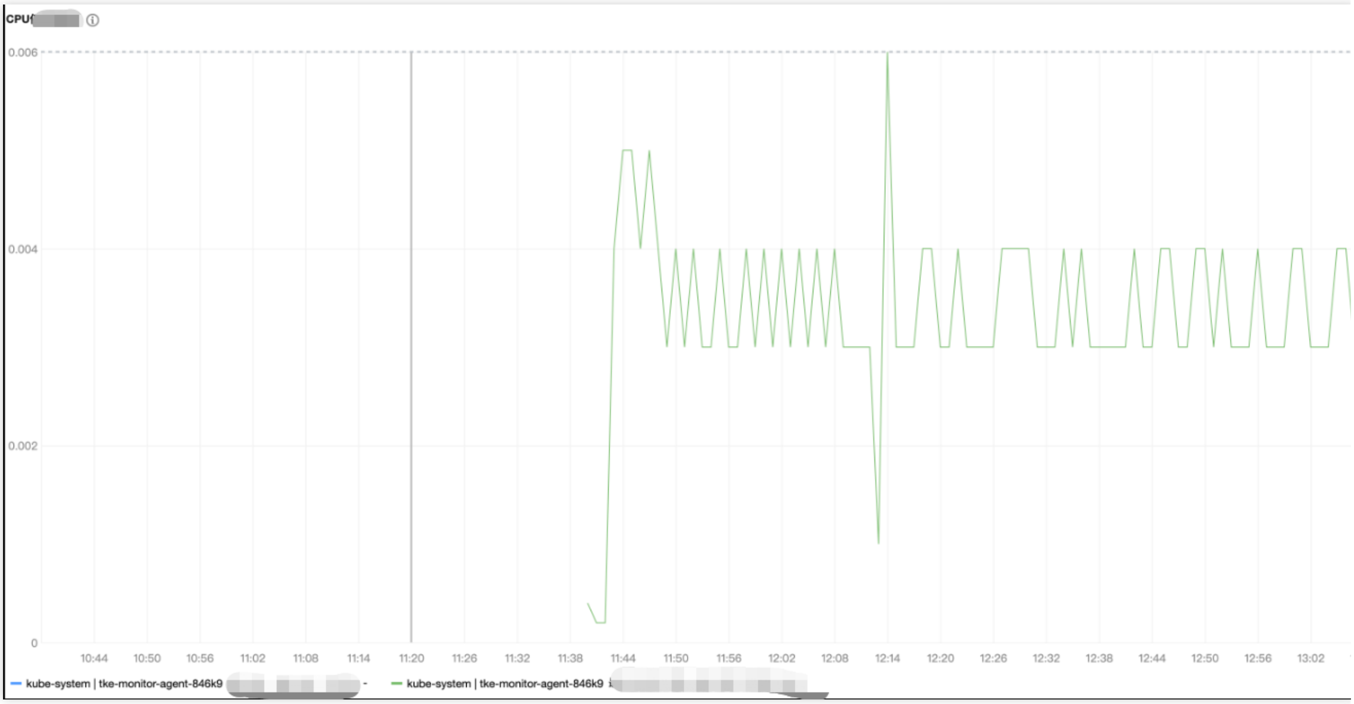
Data volume

220 Pods are deployed on a node, and each Pod contains three containers.

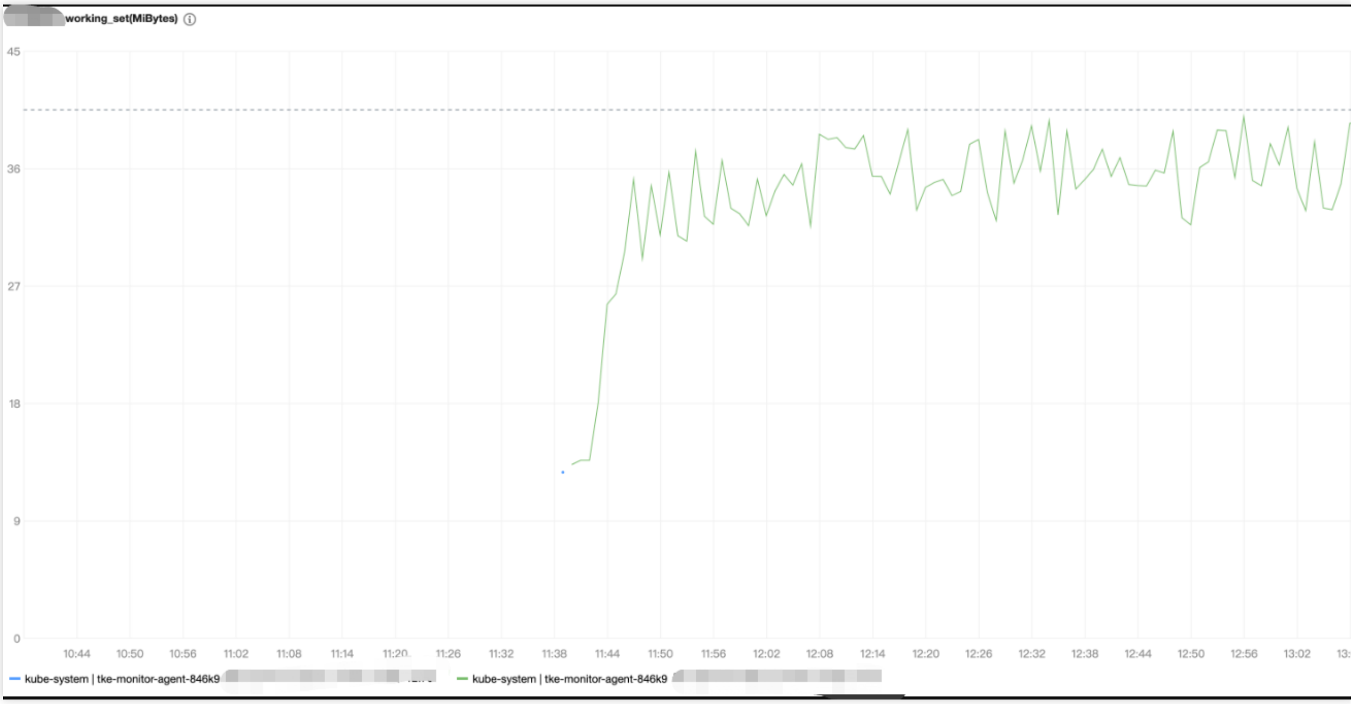
Resources consumed

MEM (peak)	CPU (peak)
About 40 MiB	0.01C

The stress test result of the CPU usage is as shown below:



The stress test result of the memory usage is as shown below:



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
It is required to gather the number of Pods and related information in the cluster.	ReplicaSets, Deployments, and Pods	list/watch
Obtaining the metric information of cadvisor by visiting the /metrics port on the Kubelet of the node.	nodes, nodes/proxy, and nodes/metrics	list/watch/get
Delivering metric data with cluster-monitor	services	list/watch
Reporting metrics to HPA-Metrics-Server	custommetrics	update

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: tke-monitor-agent
rules:
  - apiGroups: ["apps"]
    resources: ["replicasets"]
    verbs: ["list", "watch"]
  - apiGroups: ["apps"]
    resources: ["deployments"]
    verbs: ["list", "watch"]
  - apiGroups: [""]
    resources: ["nodes", "nodes/proxy", "nodes/metrics"]
    verbs: ["list", "watch", "get"]
  - apiGroups: [""]
    resources: ["services"]
    verbs: ["list", "watch"]
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["list", "watch"]
  - apiGroups: ["monitor.tencent.io"]
    resources: ["custommetrics"]
    verbs: ["update"]
```

GPU-Manager Add-on

Last updated：2022-04-20 16:51:38

Overview

Add-on description

GPU Manager is an all-in-one GPU manager. It is implemented based on the Kubernetes Device Plugin system. This manager provides features such as assigning shared GPU, querying GPU metrics, and preparing GPU-related devices before running a container. It supports your use of GPU devices in Kubernetes clusters.

Add-on features

- **Topology assignment**: provides an assignment feature based on GPU topology. When you assign an application with more than one GPU card, it can select the fastest topology link method to assign the GPU device.
- **GPU sharing**: allows you to submit tasks with less than one GPU card, and supplies QoS assurance.
- **Querying application GPU metrics**: You can access the `/metric` path of the CVM port (by default this is 5678) to provide GPU metrics collection feature for Prometheus, and can access the `/usage` path to perform state querying of readable containers.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Recommended Resource Reservation	Namespaces
gpu-manager-daemonset	DaemonSet	Each node: 1-core CPU, 1 Gi memory	kube-system
gpu-quota-admission	Deployment	Each node: 1-core CPU, 1 Gi memory	kube-system

Use Cases

When the GPU application is running in a Kubernetes cluster, it can prevent the waste of resources caused by requesting a whole card in scenarios such as AI training, allowing full utilization of resources.

Limits

- This add-on is implemented through Kubernetes Device Plugin. It can be directly used on clusters of Kubernetes version 1.10 and above.

- Each GPU card is split into 100 shares. The usage of GPU can only be a between 0.1 to 0.9, or an integer. VRAM resources are assigned memory with 256MiB as the smallest unit.
- To use GPU-Manager, the cluster must contain GPU model nodes.

Directions

Installing the add-on

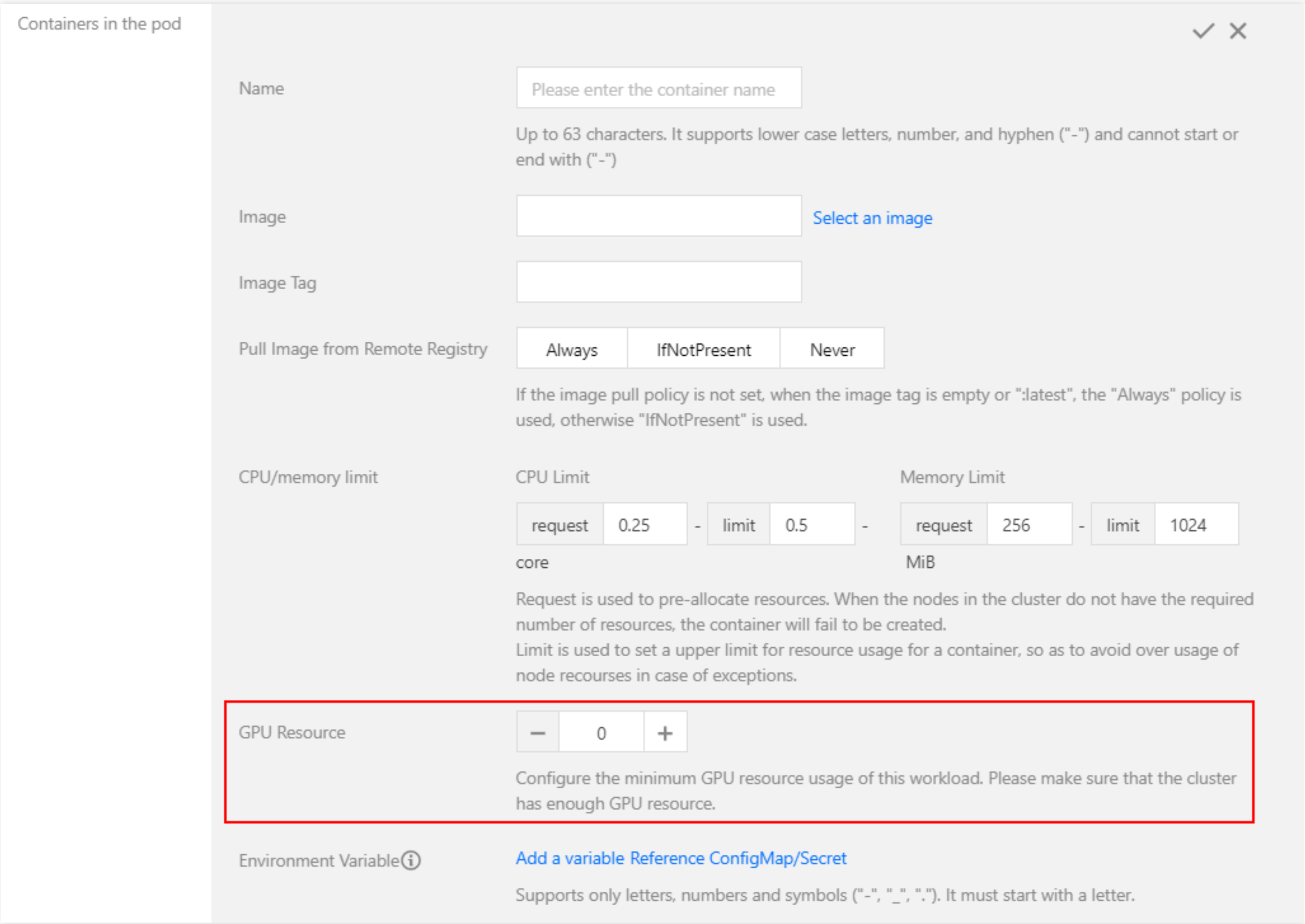
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. In the left sidebar, click **Add-on Management** to go to the **Add-on List** page.
4. On the "Add-On List" page, click **Create**. On the "Create an Add-On" page that appears, select **GpuManager**.
5. Click **Done** to complete the process.

Creating fine-grained GPU workloads

After the GpuManager add-on is successfully installed, you can use the following two methods to create fine-grained GPU workloads.

Method 1: creating through the TKE Console

1. Log in to the TKE console and select **Clusters** in the left sidebar.
2. Select the cluster for which you want to create the GPU application to go to the workload management page, and click **Create**.
3. On the **Create a Workload** page, set the configuration as needed. You can configure a fine-grained GPU workload in **GPU Resource**, as shown below:



Containers in the pod

✓ ✕

Name	Please enter the container name Up to 63 characters. It supports lower case letters, number, and hyphen ("-") and cannot start or end with ("-")
Image	Select an image
Image Tag	
Pull Image from Remote Registry	Always / IfNotPresent / Never If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.
CPU/memory limit	CPU Limit: request 0.25 - limit 0.5 - core Memory Limit: request 256 - limit 1024 - MiB Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail to be created. Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node recourses in case of exceptions.
GPU Resource	− 0 + Configure the minimum GPU resource usage of this workload. Please make sure that the cluster has enough GPU resource.
Environment Variable ⓘ	Add a variable Reference ConfigMap/Secret Supports only letters, numbers and symbols ("-", "_", "."). It must start with a letter.

Method 2: creating through YAML

Note：

When submitting, use YAML to set the GPU resource usage for the container. The core resources must have `tencent.com/vcuda-core` entered in "resource". The VRAM resources must have `tencent.com/vcuda-memory` entered in "resource".

A YAML example is as follows:

- P4 device using 1 card:

```
apiVersion: v1
kind: Pod
...
spec:
```

```
containers:
- name: gpu
resources:
tencent.com/vcuda-core: 100
```

- 5GiB VRAM application using 0.3 cards:

```
apiVersion: v1
kind: Pod
...
spec:
containers:
- name: gpu
resources:
tencent.com/vcuda-core: 30
tencent.com/vcuda-memory: 20
```

Cluster Autoscaler

Last updated：2024-06-14 16:28:43

Overview

Component Overview

The Cluster Autoscaler (CA) component offers auto-scaling (AS) capabilities for clusters based on simulated scheduling algorithms. It supports adding new nodes when resources are scarce and removing old nodes when resources are idle.

Note

This component must be used in conjunction with a node pool, and now it supports native nodes and regular nodes. To use this capability, ensure that the node pool has **AS** enabled.

Kubernetes objects deployed in a cluster

Kubernetes Object Name	Type	Resource Amount	Namespace
cluster-autoscaler	PodDisruptionBudget	-	kube-system
cluster-autoscaler	ServiceAccount	-	kube-system
cluster-autoscaler	Secret	-	kube-system
cluster-autoscaler	ClusterRole	-	-
cluster-autoscaler	ClusterRoleBinding	-	-
cluster-autoscaler	Role	-	kube-system
cluster-autoscaler	RoleBinding	-	kube-system
cluster-autoscaler	Service	-	kube-system
cluster-autoscaler	Deployment	0.5C1G (for new creations)	kube-system

Application Scenario

When instances (pods) in the cluster cannot be scheduled due to insufficient resources, an auto scale-out action is triggered to add nodes of an appropriate type through simulated scheduling, reducing your labor costs.

When scale-in conditions are met, an auto scale-in action is triggered to remove nodes, saving you resource costs.

Limits

Kubernetes cluster version: 1.16 or later

Component Principles

Scale-out Principle

1. When the cluster lacks sufficient resources (the cluster's compute/storage/network resources cannot meet the Request/Affinity rules of pods), CA detects pods that are pending due to unschedulability.
2. CA makes scheduling decisions based on the node template of each node pool, selecting the appropriate node template.
3. If multiple templates are suitable, meaning there are multiple scalable node pools available, CA will call expanders to select the optimal template and scale out the corresponding node pool.

Scale-in Principle

1. CA detects that the allocation rate (namely, the Request value, which takes the maximum value of the CPU allocation rate and MEM allocation rate) is lower than the threshold specified for a node. When calculating the allocation rate, you can set the Daemonset type not to count in a pod's occupied resources.
2. CA checks whether the cluster's status meets the following requirements for triggering a scale-in action:
Node idle duration requirement (default: 10 minutes)
Cluster expansion buffer time requirement (default: 10 minutes)
3. CA checks whether a node meets the conditions for scale-in. You can configure the following do-not-remove conditions as needed (nodes that meet these conditions will not be removed by CA):
Nodes containing local storage.
Nodes containing pods that are not managed by DaemonSet in the kube-system namespace.
4. After evicting the pods on the node, CA releases/shuts down the node.
Completely idle nodes can be removed in batches. (The maximum number of nodes that can be removed in a batch can be set.)
Non-completely idle nodes are removed one by one.

Parameter description

Module	Functional Item	Parameter Value Description
Scale-	Scale-out	**Random (default)**: If there are multiple scalable node pools, any one of

out	Algorithm	them will be selected for scaling out. **Most-pods**: If there are multiple scalable node pools, the node pool running the highest number of pods will be selected for scaling out. **Least-waste**: If there are multiple scalable node pools, the node pool with the least resource waste will be selected for scaling out. **Priority**: If there are multiple scalable node pools, the node pool with higher priority will be selected for scaling out according to your custom ConfigMap (see below for details). This feature is only supported by native node pools and is ineffective for regular node pools.
Scale-in	Maximum Concurrent Scale-downs	The number of nodes that can be removed in a batch during a scale-in action. **Note:** Only completely idle nodes are removed in a batch. Nodes that contain any pods are removed one by one.
	Scale-in Conditions	Threshold: Scale-in conditions are evaluated when the percentage of resources occupied by pods to allocable resources is less than x%.
		Trigger latency: A node is removed after being continuously idle for x minutes.
		Silent period: Scale-in conditions are evaluated x minutes after the cluster is scaled out.
	Do Not Scale Down Nodes	Nodes containing pods with local storage (including hostPath and emptyDir). Nodes containing pods that are not managed by DaemonSet in the kube-system namespace.

###Using a Scale-out Algorithm Based on the Priority in Custom ConfigMap

Note

This feature is only supported by native node pools and is ineffective for regular node pools.

Priority values range from 1 to 100 and must be positive integers.

A node pool ID maps one and only one priority.

If the node pool ID is not configured in ConfigMap, even if scale-out requirements are met, the scale-out action will not be performed because the priority is not configured.

Below is a sample:

```
apiVersion: v1
data:
  priorities: |-
    100:
      - np-l5wmakan     #np-l5wmakan (node pool ID) has a priority of 100.
    50:
      - np-9r9rh7kp     #np-9r9rh7kp (node pool ID) has a priority of 50.
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander     #The name must be set to cluster-a
  namespace: kube-system
```

Related Links

Creating Native Nodes

Creating Regular Nodes

CFSTURBO-CSI

Last updated：2024-02-28 18:04:47

Component Overview

The Kubernetes-csi-tencentcloud CFSTURBO plugin implements the CSI interface, facilitating the usage of Tencent Cloud's CFS Turbo Cloud File Storage within a container cluster.

Use Cases

Cloud File Storage CFS Turbo is ideal for large-scale throughput and mixed workload businesses, providing private protocol mounting. The performance of a single client can match that of the storage cluster. You can pair it with Cloud Virtual Machine (CVM), Tencent Kubernetes Engine (TKE), BatchCompute, and other services.

CFS Turbo is effortlessly accessible. No need to adjust your existing business structure or undertake complex configurations. In just three steps, you can complete the access and usage of the file system: create a file system, enable the file system client on the server, and subsequently mount the created file system. With the CFSTURBO-CSI extension component, CFS Turbo can swiftly be brought into play within a container cluster by using the standard native Kubernetes.

Use Limits

The CFSTURBO-CSI extension component requires a Kubernetes version of 1.14 or later.

For use limits of CFS Turbo, see Use Limits.

To use CFS Turbo in TKE, it is required to install such an extension component in the cluster, which will occupy certain system resources.

csfTurbo-csi Permission

Permission Description

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

It is required to mount the host /var/lib/kubelet related directory to the container to complete the mount/umount of the volume, hence the activation of the privileged-level container is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permissi
Supporting dynamic creation/deletion of cfsturbo subdirectory type pv	persistentvolumeclaims/persistentvolumes/storageclasses	get/list/watch/create/delete/upd
	node	get/list/

Permission Definition

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cfsturbo-csi-controller-role
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete", "update"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "list", "watch", "create", "update", "patch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
```

Directions

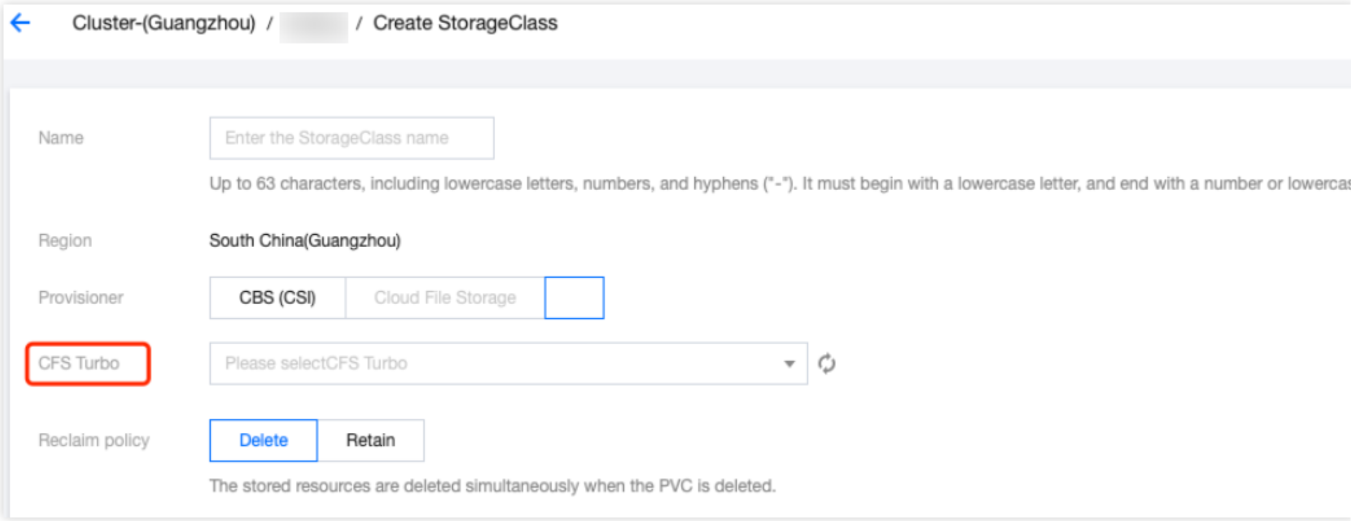
Installing a Component

1. Log in to 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 **Component Management** from the left-side menu, and click **Create** on the Component Management page.
4. In the **Create** page, select the **CFSTurbo**.
5. Click **Done** to install the component.

Specifying StorageClass

Step 1: Create a CFS Turbo Type StorageClass

1. In the cluster list, click a cluster ID to enter the cluster details page.
2. Select **Storage** > **StorageClass** from the left menu bar, then click **Create** on the StorageClass page.
3. In the **Create StorageClass** page, configure the StorageClass parameters as shown below:



Cluster-(Guangzhou) / [redacted] / Create StorageClass

Name: Enter the StorageClass name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercas

Region: South China(Guangzhou)

Provisioner: CBS (CSI) | Cloud File Storage

CFS Turbo: Please selectCFS Turbo

Reclaim policy: Delete | Retain

The stored resources are deleted simultaneously when the PVC is deleted.

Configuration Item	Description
Name	Enter the StorageClass name.
Region	It is the region of the cluster by default.
Provisioner	Select Cloud File Storage CFS Turbo here.
CFS Turbo	Please select the already created CFS Turbo. If you do not have a suitable CFS Turbo, please go to the Cloud File Storage console to create a new one. For details, see Creating File Systems and Mount Targets.
Reclaim Policy	Two reclaim policies are provided: Delete and Retain. For data security, it is recommended to use the Retain reclaim policy. Delete: If a PV is dynamically created through a PVC, the PV and storage instance bound to the PVC will be automatically terminated when the PVC is terminated. Retain: If a PV is dynamically created through a PVC, the PV and storage instance bound to the PVC will be retained when the PVC is terminated.

4. Click **Create StorageClass** to complete the creation.

Step 2: Create a PersistentVolumeClaim

1. In the cluster list, click a cluster ID to enter the cluster details page.
2. Select **Storage** > **PersistentVolumeClaim** from the left menu bar, and click **Create** on the PersistentVolumeClaim page.
3. On the **Create PersistentVolumeClaim** page, set the key parameters of your PVC.

Configuration Item	Description
Name	Enter the name of the PersistentVolumeClaim.

Namespace	A namespace is used to assign cluster resources. Select default here.
Provisioner	Select Cloud File Storage CFS Turbo.
Read/Write Permission	CFS only supports multi-server read and write.
Specify StorageClass or Not	Select **Specify**. **Note:** The PVC and PV will be bound to the same StorageClass. **Do not specify** implies that the StorageClass value of this PVC will be empty, corresponding to a void value under the `storageClassName` field in the YAML file as a character string.
StorageClass	Choose the StorageClass created in the steps above.
Specify a PersistentVolume or Not	Specify the PersistentVolume according to demand. **Note:** The system first searches the current cluster for PVs that meet the binding rules. If there are no such PVs, the system dynamically creates a PV to be bound based on the PVC and StorageClass parameters. Either the StorageClass or PersistVolume should be specified. For more details on **Not Specifying PersistVolume**, see PV and PVC Binding Rules.

4. Click **Create PersistentVolumeClaim**.

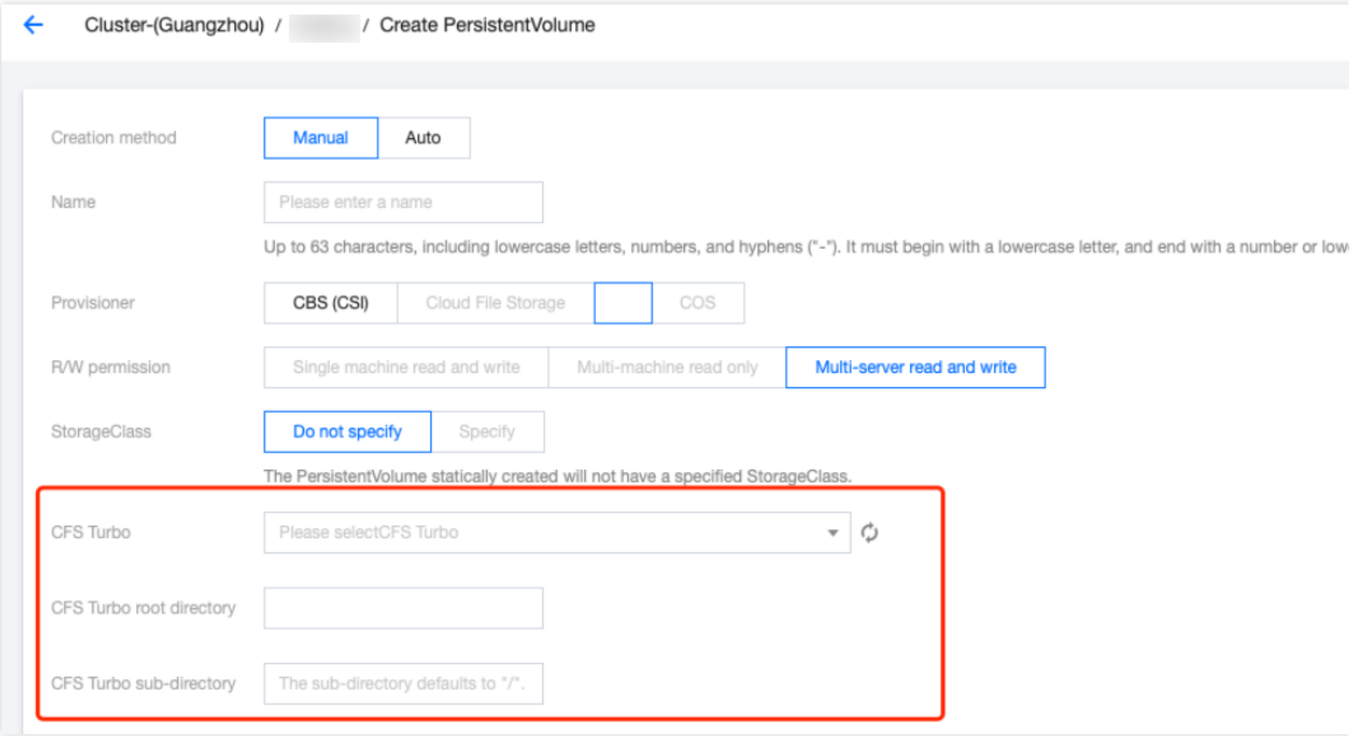
Step 3: Create a workload

1. In the cluster list, click a cluster ID to enter the cluster details page.
2. Select **Workload** > **Deployment** from the left menu bar, and click **Create** on the Deployment page.
3. On the Create Deployment page, configure Workload parameters. For parameter details, see Creating a Deployment. Based on actual requirements, choose **Use existing PVC** for the data volume and select the PVC you have created.
4. After mounting to the specified path in the container, click **Create Deployment**.

Not Specifying StroageClass

Step 1: Create a PersistentVolume

1. In the cluster list, click a cluster ID to enter the cluster details page.
2. Select **Storage** > **PersistentVolume** in the left menu bar, and click **Create** on the PersistentVolume page.
3. On the **Create PersistentVolume** page, configure the key parameters for PV as shown below:



Cluster-(Guangzhou) / [redacted] / Create PersistentVolume

Creation method: Manual | Auto

Name: Please enter a name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or low

Provisioner: CBS (CSI) | Cloud File Storage | | COS

R/W permission: Single machine read and write | Multi-machine read only | Multi-server read and write

StorageClass: Do not specify | Specify

The PersistentVolume statically created will not have a specified StorageClass.

CFS Turbo: Please selectCFS Turbo

CFS Turbo root directory:

CFS Turbo sub-directory: The sub-directory defaults to "/".

Configuration Item	Description
Source Setting	Select **Static Creation**.
Name	Enter the name of the PersistentVolume.
Provisioner	Select Cloud File Storage CFS Turbo.
Read/Write Permission	CFS only supports multi-server read and write.
Specify StorageClass or Not	Select **Do not specify**.
CFS Turbo	Please select the created CFS Turbo. If you do not have a suitable CFS Turbo, please go to the Cloud File Storage console to create a new one. For details, see Creating File Systems and Mount Targets.
CFS Turbo Root Directory	Fill in according to the root directory of the mount point information in CFS Turbo.
CFS Turbo Subdirectory	Fill in according to the subdirectory of the mount point information in CFS Turbo.

4. Click **Create PersistentVolume** to complete the creation.

Step 2: Create a PersistentVolumeClaim

1. In the cluster list, click a cluster ID to enter the cluster details page.
2. Select **Storage** > **PersistentVolumeClaim** in the left menu bar, and click **Create** on the PersistentVolumeClaim page.
3. On the **Create PersistentVolumeClaim** page, configure the key parameters of your PVC.

Configuration Item	Description
Name	Enter the name of the PersistentVolumeClaim.
Namespace	A namespace is used to assign cluster resources. Select default here.
Provisioner	Select Cloud File Storage CFS Turbo.
Read/Write Permission	CFS only supports multi-server read and write.
Specify StorageClass or Not	Select **Do not specify**. **Note:** The PVC and PV will be bound to the same StorageClass. **Do not specify** implies that the StorageClass value of this PVC will be empty, corresponding to a void value under the `storageClassName` field in the YAML file as a character string.
StorageClass	Choose the StorageClass created in the steps above.
Specify a PersistentVolume or Not	Specify the PersistentVolume according to demand. **Note:** The system first searches the current cluster for PVs that meet the binding rules. If there are no such PVs, the system dynamically creates a PV to be bound based on the PVC and StorageClass parameters. Either the StorageClass or PersistVolume should be specified. For details on **Not Specifying PersistVolume**, see PV and PVC Binding Rules.

4. Click **Create PersistentVolumeClaim**.

Step 3: Create a workload

1. In the cluster list, click a cluster ID to enter the cluster details page.
2. Select **Workload** > **Deployment** in the left menu bar, and click **Create** on the Deployment page.
3. In the Create Deployment page, configure the workload parameters. For more details, see Creating a Deployment. According to actual requirements, choose **Use Existing PVC** for the data volume and select the created PVC.

tke-log-agent

Last updated：2024-02-05 16:10:58

Overview

Component Overview

tke-log-agent is a Kubernetes cluster log collection component. It allows users to unobtrusively collect standard output logs from containers, log files within the containers, and node logs.

Resource objects deployed in the cluster

Kubernetes Object Name	Type	Resource Amount	Namespace
tke-log-agent	Daemonset	0.21C126M	kube-system
cls-provisioner	Deployment	0.1C64M	kube-system
logconfigs.cls.cloud.tencent.com	CustomResourceDefinition	-	-
cls-provisioner	ClusterRole	-	-
cls-provisioner	ClusterRoleBinding	-	-
cls-provisioner	ServiceAccount	-	kube-system
tke-log-agent	ClusterRole	-	-
tke-log-agent	ClusterRoleBinding	-	-
tke-log-agent	ServiceAccount	-	kube-system

Application scenarios

When the independent cluster initiates the audit log collection, it will by default install the tke-log-agent and collect the apiserver audit logs.
You can collect standard output logs from containers, log files within the containers, and node logs via the collection rules.

Component principle

1. After detecting that a user has created a collection rule, the cls-provisioner will generate a collection configuration from the CLS side and sync it to the CLS server side based on the configuration information of the collection rule.
2. The tke-log-agent maps log directories to a unified directory based on the collection rules.
3. The loglistener syncs with the CLS server side collection configuration, collecting and reporting logs to the CLS side based on the collection configuration.

Component Permissions Description

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**.

Log-Agent Permission

Permission Description

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

Only standard clusters with log collection enabled will deploy this component, other types of clusters will not deploy.

It requires read and write capabilities in the host directory for metadata files, thus the activation of privileged-level containers is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
Monitoring changes in the log collection rules	logconfig/logconfigpro	watch/patch/get
Obtaining runtime types of the nodes	node	list/watch/get
When collecting logs within the standard output logs/containers, it is required to collect logs from specific namespace pods.	namespace/pod	list/watch/get
When collecting logs within the containers, it is required to obtain the actual storage path of the container logs.	PV/PVC	list/watch/get
	SC	get
Collecting relevant logs related to workloads	Workloads	list/watch/get

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: tke-log-agent
rules:
  - apiGroups: ["cls.cloud.tencent.com"]
    resources: ["logconfigs","logconfigpros"]
    verbs: ["list", "watch", "patch","get"]
  - apiGroups: [""]
```

```
    resources: ["pods", "namespaces", "nodes", "persistentvolumeclaims","configmaps
    verbs: ["list", "watch", "get"]
  - apiGroups: ["apps"]
    resources: ["daemonsets","replicasets","deployments","statefulsets"]
    verbs: ["list", "watch", "get"]
  - apiGroups: ["batch"]
    resources: ["jobs","cronjobs"]
    verbs: ["list", "watch", "get"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get"]
```

cls-provisioner 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
Synchronizing the rule content of log config to the CLS side	logconfig	list/watch/patch/update

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cls-provisioner
rules:
- apiGroups:
  - cls.cloud.tencent.com
  resources:
  - logconfigs
  verbs:
  - list
```

```
    - watch
    - patch
    - update
  - apiGroups:
    - '*'
    resources:
    - events
    - configmaps
    verbs:
    - create
    - patch
    - update
```

Related Links

Enabling Log Collection

Configuring Log Collection via the Console

Configuring Log Collection via Yaml

Using CRD to Collect Logs to Kafka

Helm Application

Overview

Last updated：2020-09-11 10:23:46

Application features refer to the features related to Helm 3.0 integrated in Tencent Kubernetes Engine (TKE), which provide you with various product and service capabilities including Helm Chart, Tencent Container Registry (TCR), and software services. Created applications will run in the specified cluster to offer corresponding capabilities.

Application-related Operations

- Managing Applications
- Connecting to a Cluster Using Local Helm Client

Use the application

Last updated：2023-07-06 11:05:03

This document describes how to create, update, roll back, and delete applications in the TKE console.

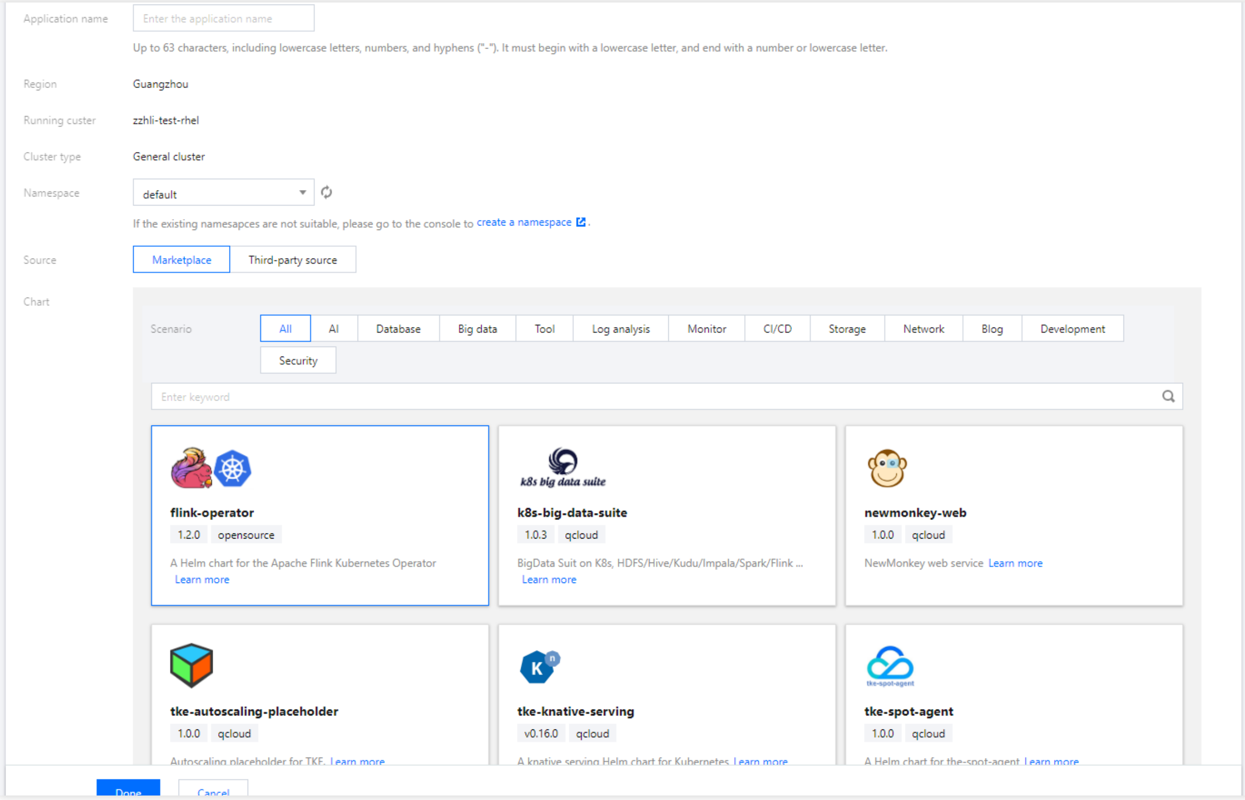
Reminder

Application management applies only to clusters with Kubernetes v1.8 or later versions.

Directions

Creating an application

1. Log in to the TKE console and click **Applications** in the left sidebar.
2. At the top of the **Applications** page, select the cluster and region where to create the application and click **Create**.
3. On the **Create Application** page, set the basic information about the application according to the settings shown in the following figure.



Application name: Enter the application name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Region: Guangzhou

Running custer: zzhli-test-rhel

Cluster type: General cluster

Namespace: default

If the existing namesapces are not suitable, please go to the console to create a namespace.

Source: Marketplace

Chart:

flink-operator 1.2.0 opensource
A Helm chart for the Apache Flink Kubernetes Operator Learn more

k8s-big-data-suite 1.0.3 qcloud
BigData Suit on K8s, HDFS/Hive/Kudu/Impala/Spark/Flink ... Learn more

newmonkey-web 1.0.0 qcloud
NewMonkey web service Learn more

tke-autoscaling-placeholder 1.0.0 qcloud
Autoscaling placeholder for TKE. Learn more

tke-knative-serving v0.16.0 qcloud
A knative serving Helm chart for Kubernetes Learn more

tke-spot-agent 1.0.0 qcloud
A Helm chart for the-spot-agent Learn more

The main parameters are described as follows:

Application Name: Enter a custom application name.

Source: Select **Marketplace**, or **Third-party Source**. For details, see the following table.

Source	Configuration Item
Marketplace	Filter charts by cluster type or application scenario. Select the required application package and chart version. You can also edit the parameters.
Third-party source	Chart address: Official and self-built Helm repositories are supported. Note that the value of this parameter must start with `http` and end with `.tgz`. In this example, the value is `http://139.199.162.50/test/nginx-0.1.0.tgz`. Type: Select **Public** or **Private** as needed. Parameters: Edit the parameters as needed.

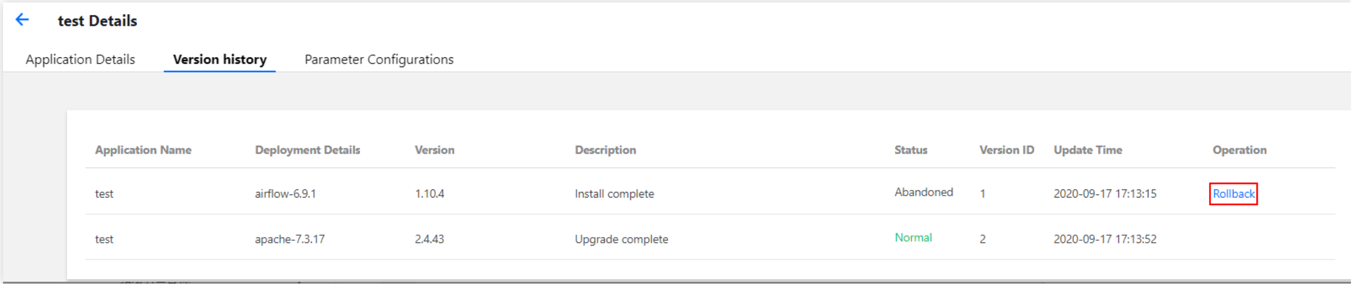
4. Click **Done**.

Updating an application

1. Go to the TKE console and click **Application** in the left sidebar.
2. On the **Application** page, locate the application to update and click **Update Application** on the right.
3. In the displayed **Update Application** window, configure the key information as needed and click **Done**.

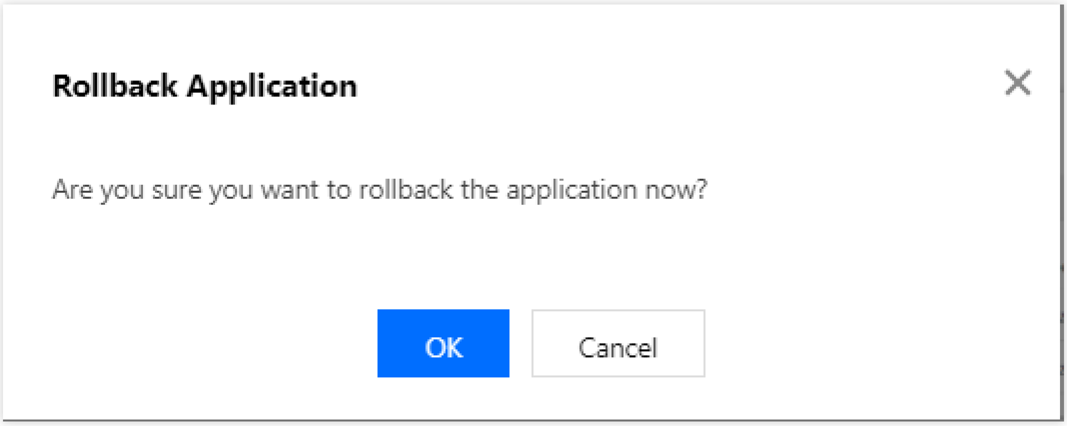
Rolling back an application

1. Go to the TKE console and click **Application** in the left sidebar.
2. On the **Application** page, click the application to update to go to the application details page.
3. On the application details page, click the **Version History** tab, locate the required version, and click **Roll Back** on the right.



Application Name	Deployment Details	Version	Description	Status	Version ID	Update Time	Operation
test	airflow-6.9.1	1.10.4	Install complete	Abandoned	1	2020-09-17 17:13:15	Rollback
test	apache-7.3.17	2.4.43	Upgrade complete	Normal	2	2020-09-17 17:13:52	

4. In the displayed **Rollback Application** window, click **OK**.



Rollback Application

Are you sure you want to rollback the application now?

Deleting an application

1. Go to the TKE console and click **Application** in the left sidebar.
2. On the **Application** page, locate the application to delete and select **Delete** on the right.
3. In the displayed **Delete Application** window, click **OK**.

Connecting to a Cluster Using the Local Helm Client

Last updated：2022-12-12 15:52:46

Operation Scenario

This document explains how to connect to a cluster by using local Helm client.

Directions

Downloading the Helm client

Run the following commands in sequence to download the Helm client. For more information on installing Helm, see Installing Helm.

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
```

```
chmod 700 get_helm.sh
```

```
./get_helm.sh
```

Configuring a Helm Chart repository (optional)

1. Run the following command to configure the official Kubernetes repository.

```
helm repo add stable https://kubernetes-charts.storage.googleapis.com/
```

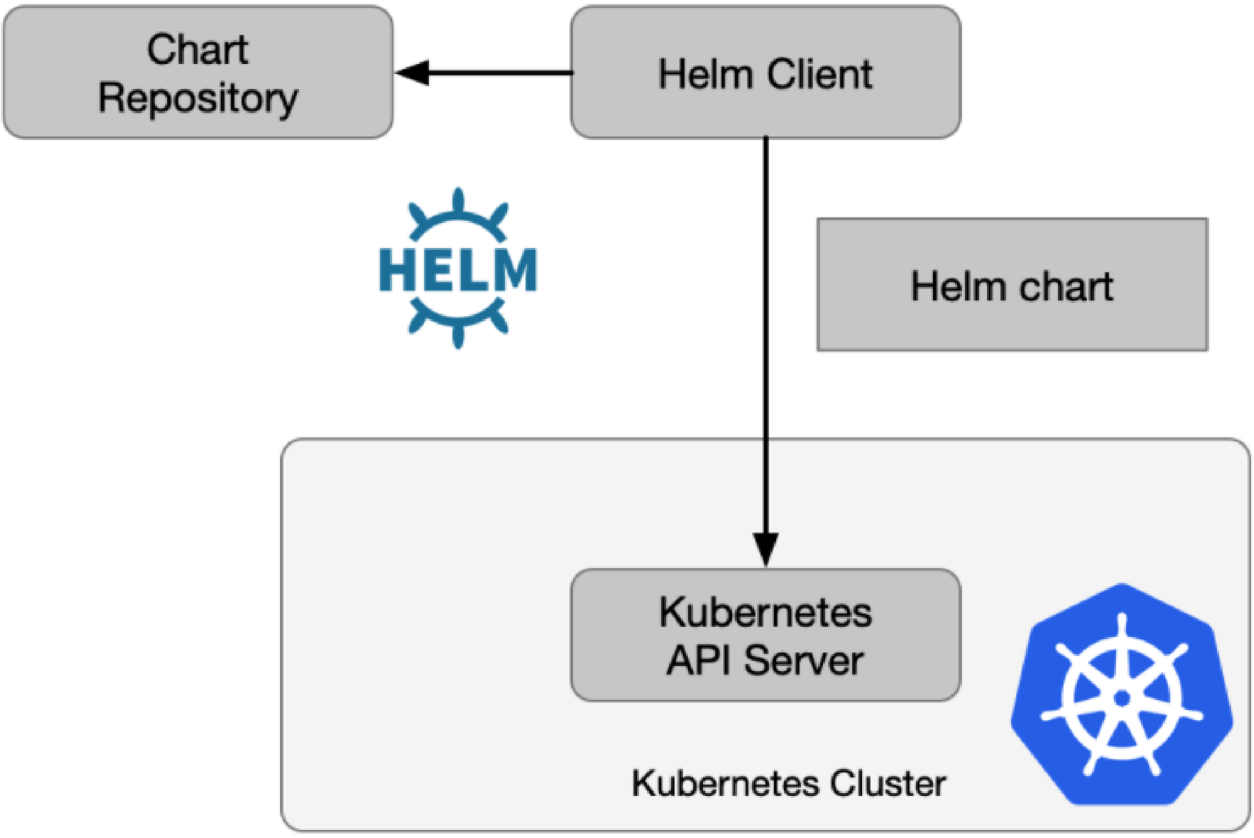
2. Run the following command to configure the Tencent Cloud application market.

```
helm repo add tkemarket https://market-tke.tencentcloudcr.com/chartrepo/opensource-stable
```

3. Configure a TCR private Helm repository.

Connecting to a cluster

Compared with Helm v2, Helm v3 has removed the Tiller component. The Helm client can directly connect to the API servers of clusters. The application-related version data is stored in Kubernetes. See the figure below:



The Helm client uses a client certificate generated by TKE to access clusters. The detailed directions are as follows:

1. Use the TKE console or APIs to obtain an available Kubeconfig for public or private network access.
2. Connect to the target cluster by referring to the following two methods:

- Use the kubeconfig obtained above to configure kubectl config use-context on the device where the Helm client is located.
- Run the following command with the specified parameter to access the target cluster.

```
helm install .... --kubeconfig [path of kubeconfig]
```

Application Market

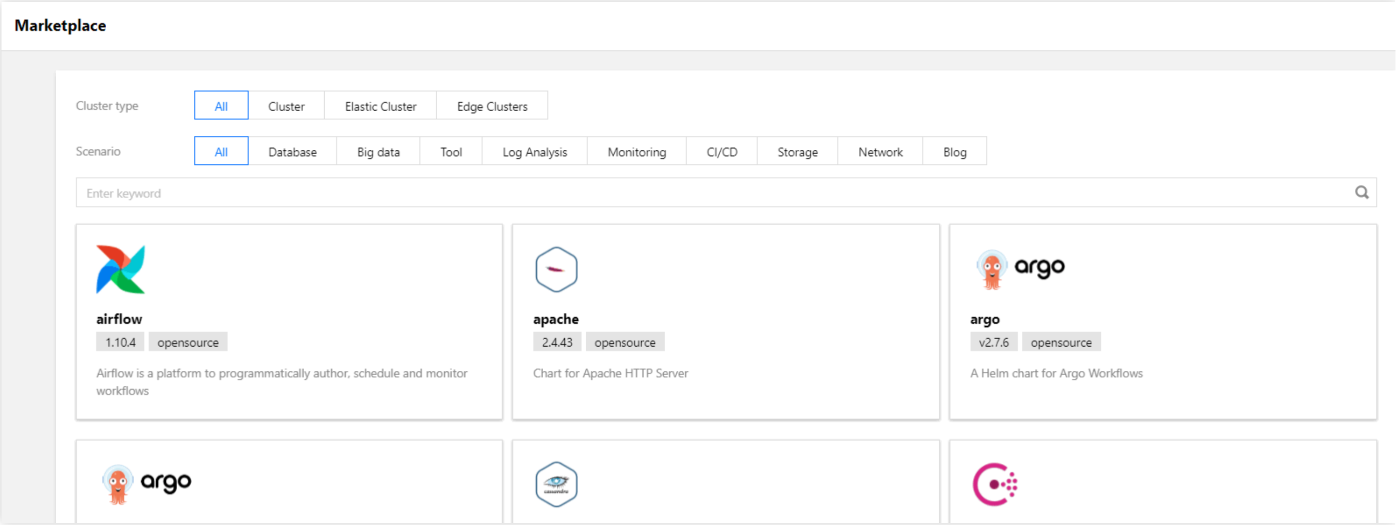
Last updated：2022-10-17 14:32:58

The Tencent Kubernetes Engine (TKE) marketplace provides a variety of products and services including Helm Chart, Tencent Container Registry (TCR), and software services that are classified by cluster type or scenario. This document describes how to create applications quickly through “Marketplace” in the TKE console.

View Applications

1. Log in to the TKE console and click **Marketplace** in the left sidebar.
2. The following operations are available on the “Marketplace” page.

- Filter applications: You can filter applications by cluster type or scenario, or by entering keywords.
  - Cluster type: The types include “Cluster”, “Serverless cluster”, “Edge cluster” and “Registered cluster”.
  - Scenario: The scenarios include “Database”, “Big data”, “Tool”, “Log analysis”, “Monitoring”, “CI/CD”, “Storage”, “Network”, and “Blog”.
- View applications: Click the desired application package to go to the details page.



Marketplace

Cluster type: All | Cluster | Elastic Cluster | Edge Clusters

Scenario: All | Database | Big data | Tool | Log Analysis | Monitoring | CI/CD | Storage | Network | Blog

Enter keyword

airflow
1.10.4 opensource
Airflow is a platform to programmatically author, schedule and monitor workflows

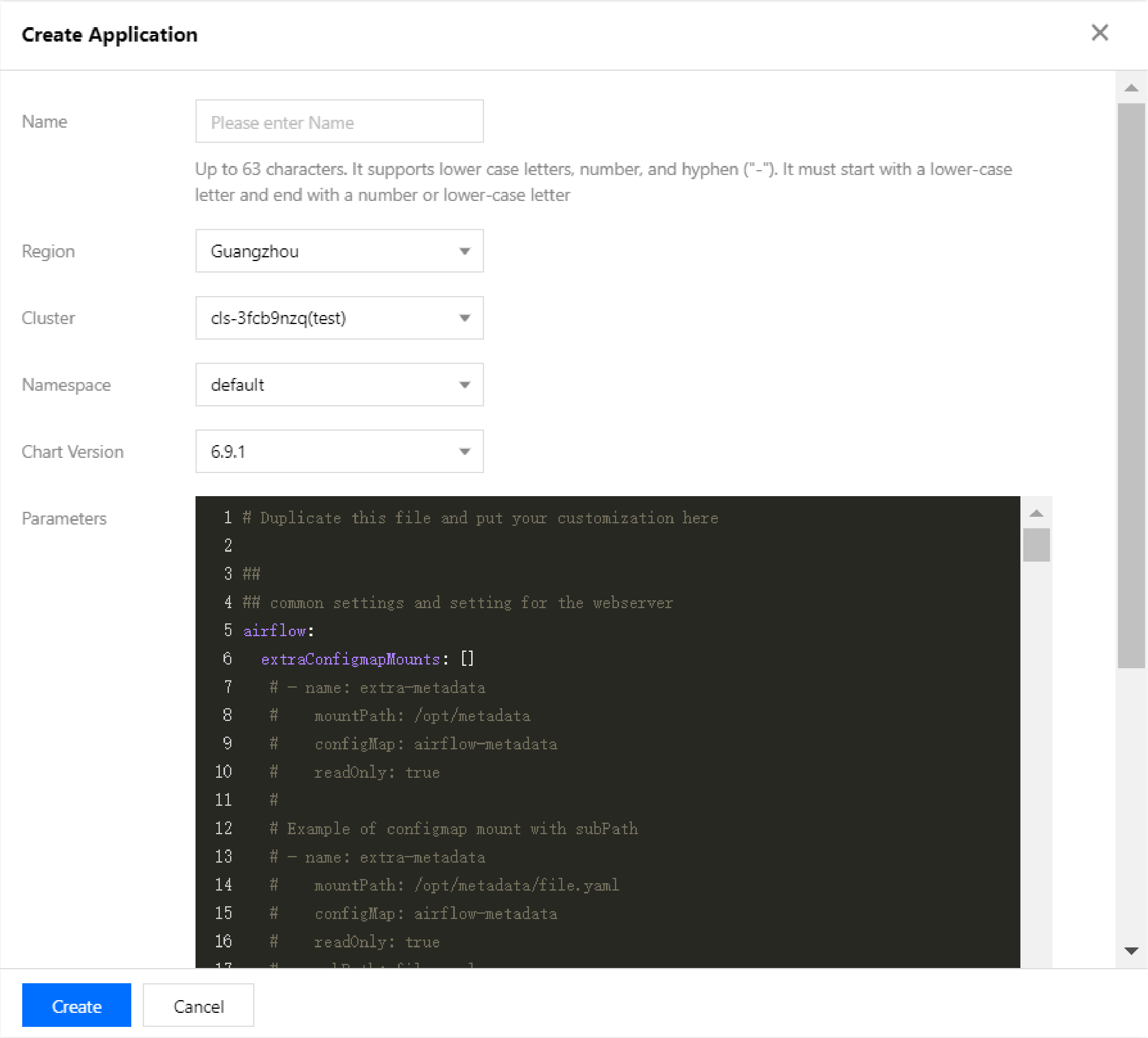
apache
2.4.43 opensource
Chart for Apache HTTP Server

argo
v2.7.6 opensource
A Helm chart for Argo Workflows

Create Applications

1. Log in to the TKE console and click **Marketplace** in the left sidebar.
2. On the **Marketplace** page, select a desired application package to go to the application details page.
3. Click **Create application** on the “Basic information” page.

4. In the “Create application” pop-up window, configure and create the application as needed.



Create Application

Name: Please enter Name

Up to 63 characters. It supports lower case letters, number, and hyphen ("-"). It must start with a lower-case letter and end with a number or lower-case letter

Region: Guangzhou

Cluster: cls-3fcb9nzq(test)

Namespace: default

Chart Version: 6.9.1

Parameters:

```
# Duplicate this file and put your customization here

##
## common settings and setting for the webserver
airflow:
  extraConfigmapMounts: []
   # - name: extra-metadata
   #    mountPath: /opt/metadata
   #    configMap: airflow-metadata
   #    readOnly: true
   #
   # Example of configmap mount with subPath
   # - name: extra-metadata
   #    mountPath: /opt/metadata/file.yaml
   #    configMap: airflow-metadata
   #    readOnly: true
```

5. Click **Create** to complete the process.

Network Management

Container Network Overview

Last updated：2023-03-14 18:19:11

Container Network and Cluster Network Descriptions

The cluster network and the container network are the basic attributes of a cluster. By setting up both networks, you can plan the network partitioning of the cluster.

Relationship between the container network and cluster network

Cluster network: Assigns IPs within the node network address range to CVMs in the cluster. You can select a subnet of a VPC instance as the node network of the cluster. For more information, see VPC Overview.
Container network: Assigns the IPs within the container network address range to containers in the cluster. It includes GlobalRouter mode, VPC-CNI mode and Cilium-Overlay mode.
GlobalRouter mode: You can customize three major private IP ranges to set up the container network. Then, you can automatically assign a CIDR block of an appropriate size to Kubernetes services based on the maximum number of intra-cluster services of your choice. You can also automatically assign an IP range of an appropriate size to each CVM in the cluster for assigning an IP to a Pod based on the maximum number of Pods per node of your choice.
VPC-CNI mode: Selects the subnet in the same VPC as the cluster for container IP assignment.
Cilium-Overlay mode: You can customize three major private IP ranges to set up the container network. Then, you can automatically assign a CIDR block of an appropriate size to Kubernetes services based on the maximum number of intra-cluster services of your choice. You can also automatically assign an IP range of an appropriate size to each node in the cluster for assigning an IP to a Pod based on the maximum number of Pods per node of your choice.

Restrictions on the container network and cluster network

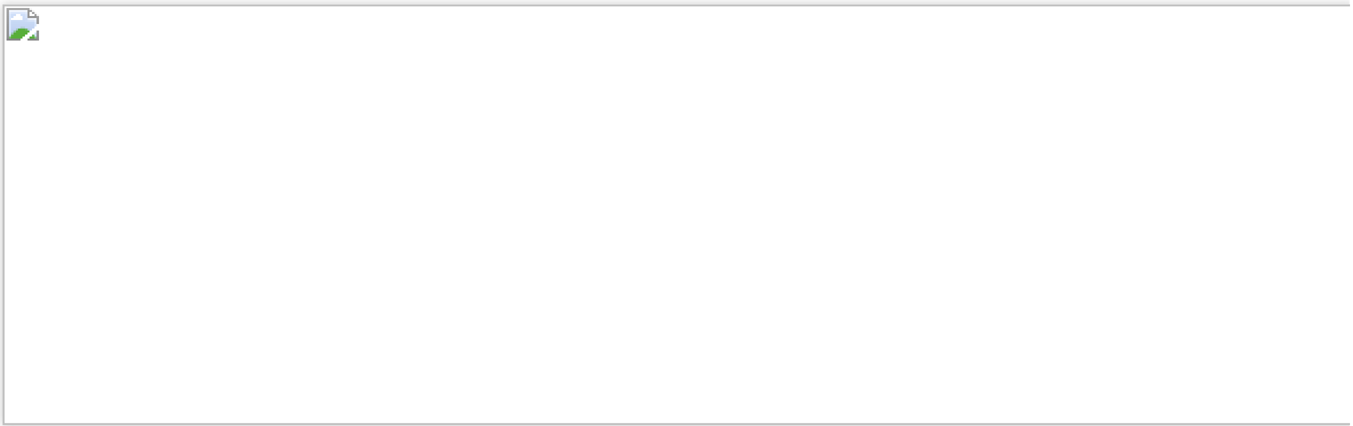
The IP ranges of the cluster network and container network cannot overlap.
The IP ranges of container networks of different clusters in the same VPC instance cannot overlap.
If the container network and the VPC route overlap, traffic will be preferentially forwarded within the container network.

Communication between the cluster network and other Tencent Cloud resources

Containers in the same cluster can communicate with one another.
Containers and nodes in the same cluster can communicate with one another.
Containers in a cluster can communicate via private network with resources in the same VPC, such as TencentDB, TencentDB for Redis, and TencentDB for Memcached.
Note:

When connecting containers in a cluster to other resources in the same VPC instance, check that the security group has opened the container IP range.
The ip-masq component in a TKE cluster prevents containers from accessing the container network and VPC network through SNAT without affecting other IP ranges. Therefore, the container IP range needs to be opened to the Internet when containers access other resources (for example, Redis) in the same VPC instance.
Setting Up Intra-Region Cross-Cluster Communication
Setting Up Cross-Region Cross-Cluster Communication
Setting communication between cluster container and IDC.

Notes on the Container Network



Container CIDR block: This indicates the IP range where the resources such as services and Pods are located.
Max Pods per node: It determines the size of CIDR block assigned to each node.
Description
A TKE cluster creates two kube-dns Pods and one l7-lb-controller Pod by default.
For Pods on a node, three addresses cannot be assigned, which are the network number, broadcast address, and gateway address. Therefore, the maximum number of Pods per node is podMax minus 3.
Max services per cluster: This determines the size of the CIDR block assigned to the service.
Description
A TKE cluster creates 3 services (kubernetes, hpa-metrics-service, and kube-dns) by default, with another 1 broadcast address and 1 network number available. Therefore, the maximum number of available services per cluster is ServiceMax minus 5.
Node: Worker nodes in a cluster.
Description
The calculation formula for the number of nodes is (CIDR IP quantity - Max services per cluster)/Max Pods per node.

How to Choose a TKE Network Mode

Tencent Kubernetes Engine (TKE) provides different network modes for different scenarios. This document gives a detailed description of the GlobalRouter, VPC-CNI and Cilium-Overlay network modes, as well as comparisons on use cases, strengths, and use limits. You can select a network mode based on your business needs.

Description

The network mode cannot be modified after it is selected when creating the cluster.

GlobalRouter mode

GlobalRouter is a global routing capability provided by TKE based on the underlying VPC instance. It implements a routing policy for mutual access between the container network and the VPC instance. For more information, see GlobalRouter Mode.

VPC-CNI mode

VPC-CNI is a container network capability provided by TKE based on CNIs and VPC ENIs. It is suitable for scenarios with high latency requirements. In this network mode, containers and nodes are located on the same network plane, and container IPs are ENI IPs assigned by the IPAMD component. For more information, see VPC-CNI Mode.

Cilium-Overlay mode

The Cilium-Overlay network mode is a container network plugin provided by TKE based on Cilium VXLan. In this mode, you can add external nodes to TKE clusters in distributed cloud scenarios. For more information, see Cilium-Overlay Mode.

Description

Due to performance loss in the Cilium-Overlay mode, this mode only supports the scenarios where external nodes are configured in distributed cloud, and does not support the scenarios where only cloud nodes are configured.

Choosing a network mode

This section compares the GlobalRouter, VPC-CNI and Cilium-Overlay network modes in terms of the use cases, strengths, and use limits. You can choose the network mode that best fits your needs.

Dimension	GlobalRouter	VPC-CNI	Cilium-Overlay
Use Cases	General container businesses Offline computing businesses	Scenarios with high network latency requirements Scenarios where static container IPs are required after a traditional architecture is migrated to a container platform	It supports only scenarios where external nodes are configured in distributed cloud. It does not support scenarios where only cloud nodes are configured.
Benefits	Container routing is performed directly	The container network of the ENI is a VPC subnet	Cloud nodes and external nodes share the specified

	through the VPC instance, and containers and nodes are located on the same network plane. Container IP ranges are dynamically assigned without occupying other IP ranges in the VPC instance. Therefore, available IPs are abundant.	and can be managed as a VPC product. Static IPs and LB-to-Pod direct access are supported. The network performance is better than that of GlobalRouter.	container IP range. Container IP ranges are dynamically assigned without occupying other IP ranges in the VPC instance. Therefore, available IPs are abundant.
Use Limits	You need to make additional configuration for interconnection scenarios such as Direct Connect, Peering Connection, and Cloud Connect Network. Static Pod IPs are not supported.	The container network and node network belong to the same VPC instance. Therefore, IP addresses are limited. The number of Pods on a node are limited by the ENI and the available IPs of the ENI. The static IP address mode does not allow Pods to be scheduled across availability zones.	There is a performance loss of less than 10% when you use the Cilium VXLan tunnel encapsulation protocol. The Pod IP cannot be accessed directly outside the cluster. You must obtain two IPs from the specified subnet to create a private CLB, so that external nodes in IDC can access APIServer and the public cloud services. Static Pod IPs are not supported.
Additional capabilities	Standard Kubernetes features.	TKE supports static Pod IPs. The container network is managed in the VPC console. LBs directly forward requests to Pods, and the Pods can obtain source IPs.	Standard Kubernetes features.

GlobalRouter Mode

GlobalRouter Mode

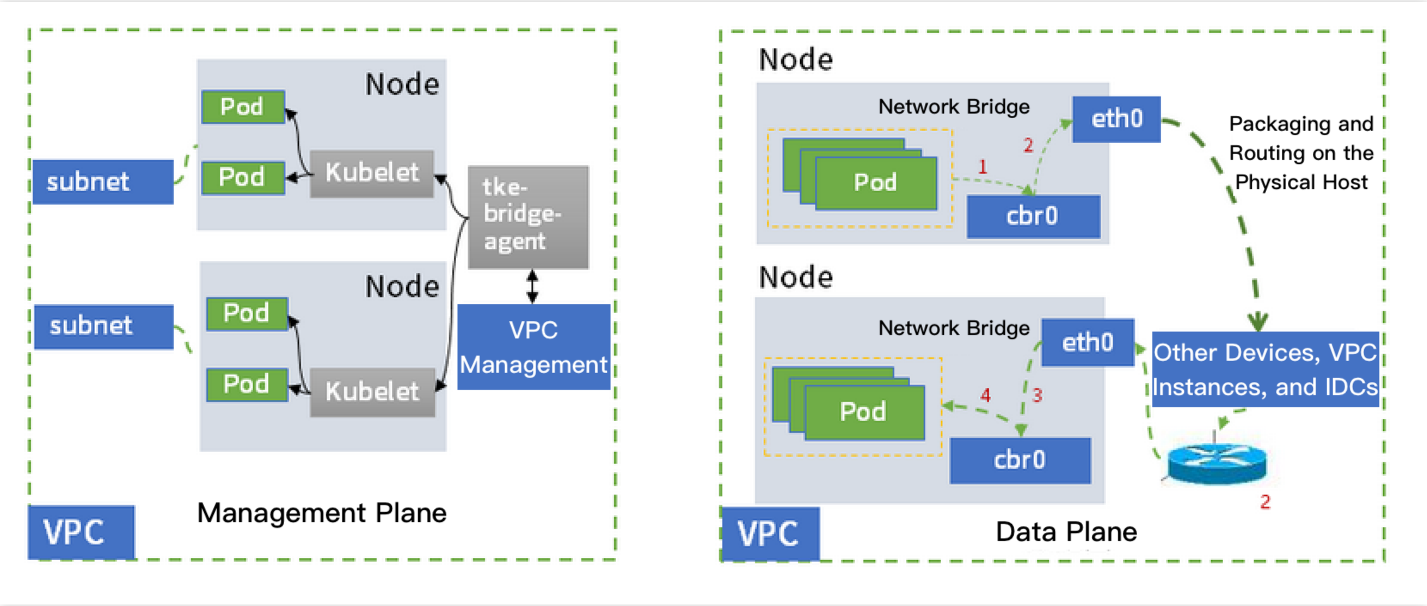
Last updated：2023-05-23 16:09:51

How It Works

GlobalRouter is a global routing capability provided by TKE based on the underlying VPC instance. It implements a routing policy for mutual access between the container network and the VPC instance. This network mode has the following characteristics:

- Container routing is performed directly through the VPC instance.
- Containers and nodes are located on the same network plane.
- Container IP ranges are dynamically assigned without occupying other IP ranges in the VPC instance.

The GlobalRouter mode is suitable for general use cases and can be seamlessly used with standard Kubernetes features. The following diagram illustrates how it works:



Use Limits

- The IP address ranges of the cluster network and container network cannot overlap.
- The IP address ranges of container networks in different clusters within the same VPC instance cannot overlap.

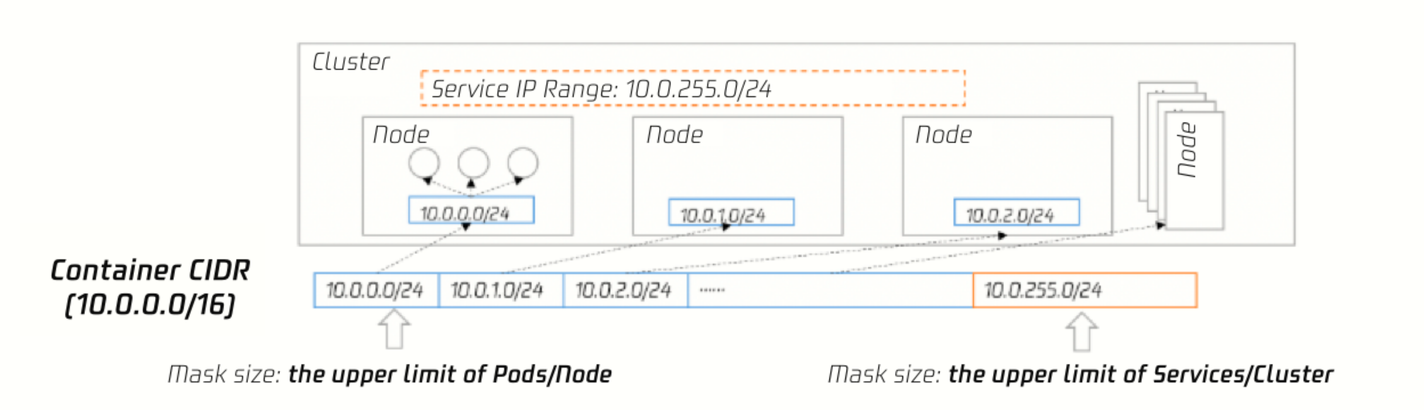
- If the container network and the VPC route overlap, traffic will be preferentially forwarded within the container network.
- The static pod IP addresses are not supported.

Container IP Address Assignment Mechanism

For container network terms and quantity calculation, see Container Network Description.

Pod IP Address Assignment

The following diagram illustrates how it works:



Note:

- Each node of the cluster will use the specified IP range in the container CIDR for the node to assign IP addresses to Pods.
- The Service IP range of the cluster will select the last segment of the specified IP range in the container CIDR for Service IP addresses assignment.
- After the node is released, the corresponding container IP range will be returned to the IP range pool as well.
- The scale out node will automatically select the available IP range in the CIDR range of the container in a loop and sequentially.

Interconnection Between Intra-region and Cross-region Clusters in GlobalRouter Mode

Last updated：2023-02-02 17:05:22

Overview

Peering Connection is a high-bandwidth and high-quality connectivity service that supports communication among Tencent Cloud resources. You can achieve **intra-region and cross-region** communication among different clusters through a peering connection.

Prerequisites

- The directions in this document are based on an existing cluster with nodes. If no such a cluster exists, create one by referring to Quickly Creating a Standard Cluster.
- Create a peering connection by referring to Creating Intra-account Peering Connection. Make sure that the peering connection is successfully created and the CVM instances can communicate with one another. If the peering connection does not work, go to the console and check if the **route table**, **CVM security groups**, and **subnet ACL** are configured properly.

Directions

Note：
To implement interconnection between cross-region clusters, submit a ticket after completing the following steps to configure container routing to interconnect containers.

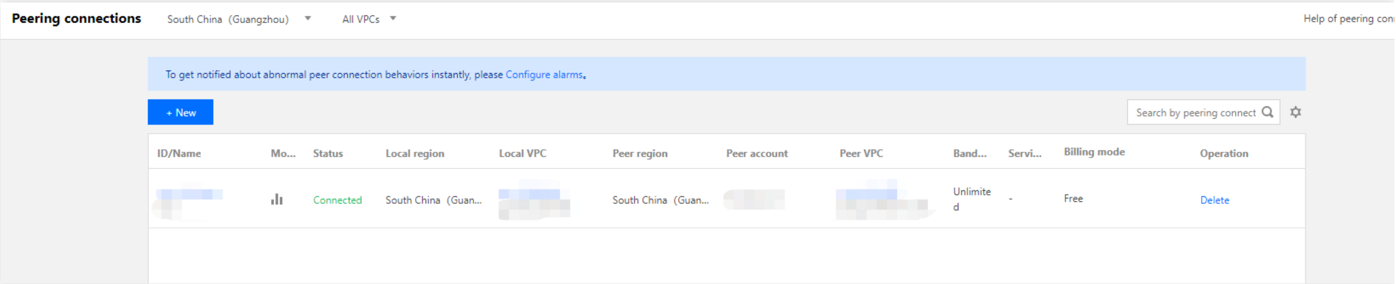
Getting container information

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Click the target cluster ID/name to go to the cluster details page. For example, go to the basic information page of cluster A.
3. Record the following information: **Region**, **Node Network**, and **Container Network**.
4. Repeat step 3 to step 4 for the other container in the different cluster.
For example, go to the basic information page of cluster B, record the **Region**, **Node Network**, and **Container**

Network of the cluster B container.

Configuring route tables

1. Log in to the VPC console and select **Peering connection** from the left sidebar.
2. On the peering connection management page, record the **name/ID** of the peering connection.



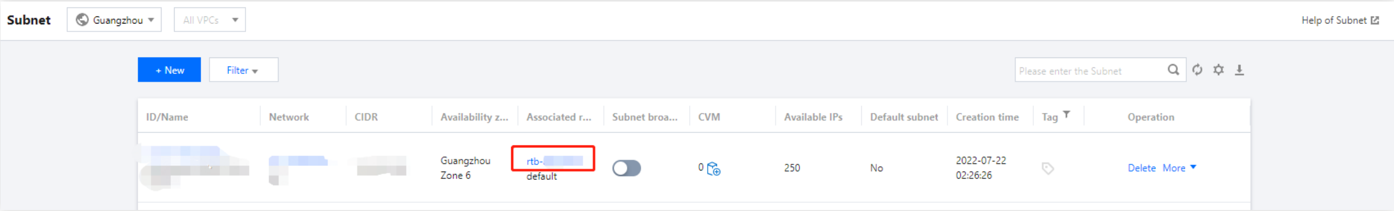
Peering connections South China (Guangzhou) ▾ All VPCs ▾ Help of peering con

To get notified about abnormal peer connection behaviors instantly, please Configure alarms.

+ New Search by peering connect

ID/Name	Mo...	Status	Local region	Local VPC	Peer region	Peer account	Peer VPC	Band...	Servi...	Billing mode	Operation
[redacted]		Connected	South China (Guan...	[redacted]	South China (Guan...	[redacted]	[redacted]	Unlimited	-	Free	Delete

3. Select **Subnet** from the left sidebar to go to the subnet management page.
4. Click the route table associated with the desired subnet.



Subnet Guangzhou ▾ All VPCs ▾ Help of Subnet

+ New Filter ▾ Please enter the Subnet

ID/Name	Network	CIDR	Availability z...	Associated r...	Subnet broa...	CVM	Available IPs	Default subnet	Creation time	Tag	Operation
[redacted]	[redacted]	[redacted]	Guangzhou Zone 6	rtb-[redacted] default		0	250	No	2022-07-22 02:26:26		Delete More ▾

5. On the details page of the associated route table, click **+ New routing policies**.
6. On the **Add a route** page, configure the following parameters:

- Destination: Enter the IP address range of the container in Cluster B.
- Next hop type: Select **Peering connection**.
- Next hop: Select the established peering connection.

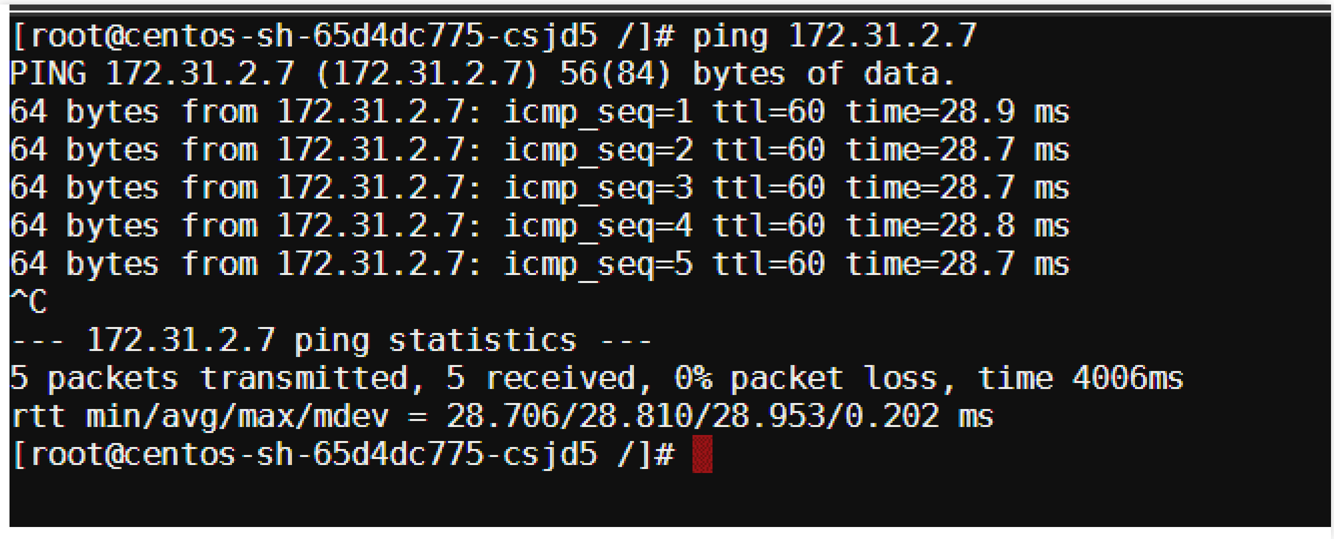
7. Click **OK** to complete the configuration of the local route table.
8. Repeat step 3 to step 7 to configure the route table of the opposite end.

Expected Results

- **Intro-region peering**: the above directions should allow containers in different clusters to communicate.
- **Cross-region cluster**: after the peering connection is established successfully, submit a ticket to configure container routing to interconnect the containers.

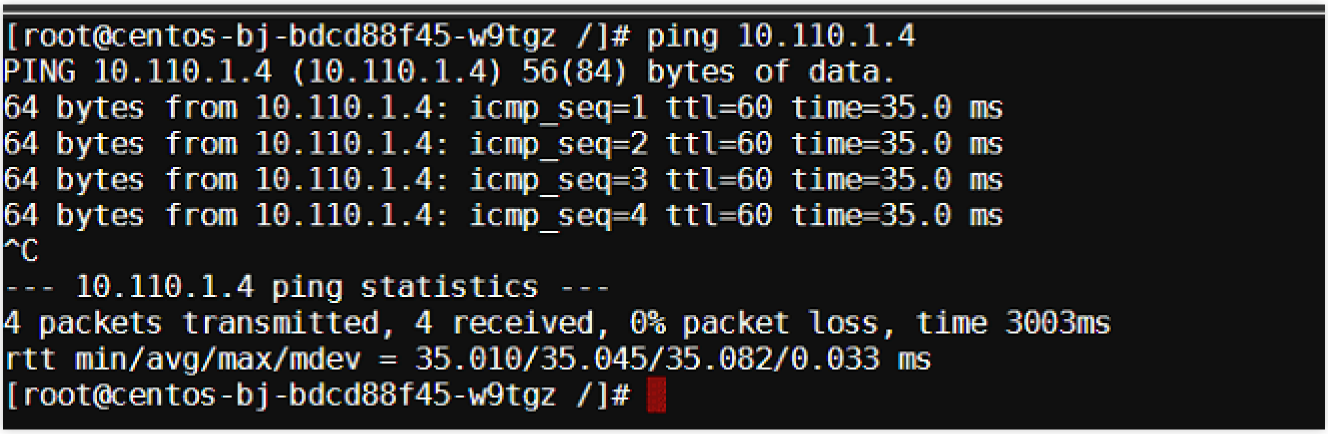
Refer to Basic Remote Terminal Operations on how to log in to a container, and verify the peering connection as instructed below:

1. Log in to the container in Cluster A and access the container in Cluster B.



```
[root@centos-sh-65d4dc775-csjd5 /]# ping 172.31.2.7
PING 172.31.2.7 (172.31.2.7) 56(84) bytes of data.
64 bytes from 172.31.2.7: icmp_seq=1 ttl=60 time=28.9 ms
64 bytes from 172.31.2.7: icmp_seq=2 ttl=60 time=28.7 ms
64 bytes from 172.31.2.7: icmp_seq=3 ttl=60 time=28.7 ms
64 bytes from 172.31.2.7: icmp_seq=4 ttl=60 time=28.8 ms
64 bytes from 172.31.2.7: icmp_seq=5 ttl=60 time=28.7 ms
^C
--- 172.31.2.7 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 28.706/28.810/28.953/0.202 ms
[root@centos-sh-65d4dc775-csjd5 /]#
```

2. Log in to the container in Cluster B and access the container in Cluster A.



```
[root@centos-bj-bdcd88f45-w9tgz /]# ping 10.110.1.4
PING 10.110.1.4 (10.110.1.4) 56(84) bytes of data.
64 bytes from 10.110.1.4: icmp_seq=1 ttl=60 time=35.0 ms
64 bytes from 10.110.1.4: icmp_seq=2 ttl=60 time=35.0 ms
64 bytes from 10.110.1.4: icmp_seq=3 ttl=60 time=35.0 ms
64 bytes from 10.110.1.4: icmp_seq=4 ttl=60 time=35.0 ms
^C
--- 10.110.1.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 35.010/35.045/35.082/0.033 ms
[root@centos-bj-bdcd88f45-w9tgz /]#
```

Interconnection Between Cluster in GlobalRouter Mode and IDC

Last updated：2022-12-23 10:37:05

Overview

Currently, a container cluster can communicate with your IDC mainly through two methods: **Direct Connect** and **IPsec VPN**.

> Note：
>
> - The directions in this document are based on an existing cluster with nodes. For how to create a cluster, see Quickly Creating a Standard Cluster.
> - Make sure that the VPC where TKE resides is successfully connected to your IDC through Direct Connect or VPN Connections. If no tunnel is established, you can establish one as instructed in Features.

Directions

Communication over Direct Connect

1. Apply for a connection as instructed in Applying for Connection.
2. Apply for a tunnel as instructed in Creating a Dedicated Tunnel.
3. Create a Direct Connect gateway as instructed in Creating Direct Connect Gateway.
4. Verify that the container node can communicate with the IDC.

   > Note：
   > When performing this step, please make sure that the container node can communicate with the IDC and the verification is successful.

5. Prepare information such as the region, `appID` , cluster ID, `vpcID` , and Direct Connect gateway ID and submit a ticket to open up the container network.
6. Select the mode of operation based on the type of protocol used by the IDC.
   - If the IDC uses the BGP protocol, the container IP address range route will be automatically synchronized.

- If other protocols are used, you need to configure the next hop of the container IP address range in the IDC to be routed to the Direct Connect gateway.

7. Verify that the container can communicate with the IDC.

Communication over VPN

Configuring an SPD Policy

1. Log in to the VPC console.
2. On the left sidebar, click **VPN Connections** > **VPN Tunnel** to enter the VPN tunnel management page.
3. On the VPN tunnel details page, click **Edit** in the **SPD Policy** column to add the container IP address range.
4. Click **Save**.
5. Repeat step 3 to step 5 to configure the SPD policy for the opposite VPN tunnel.

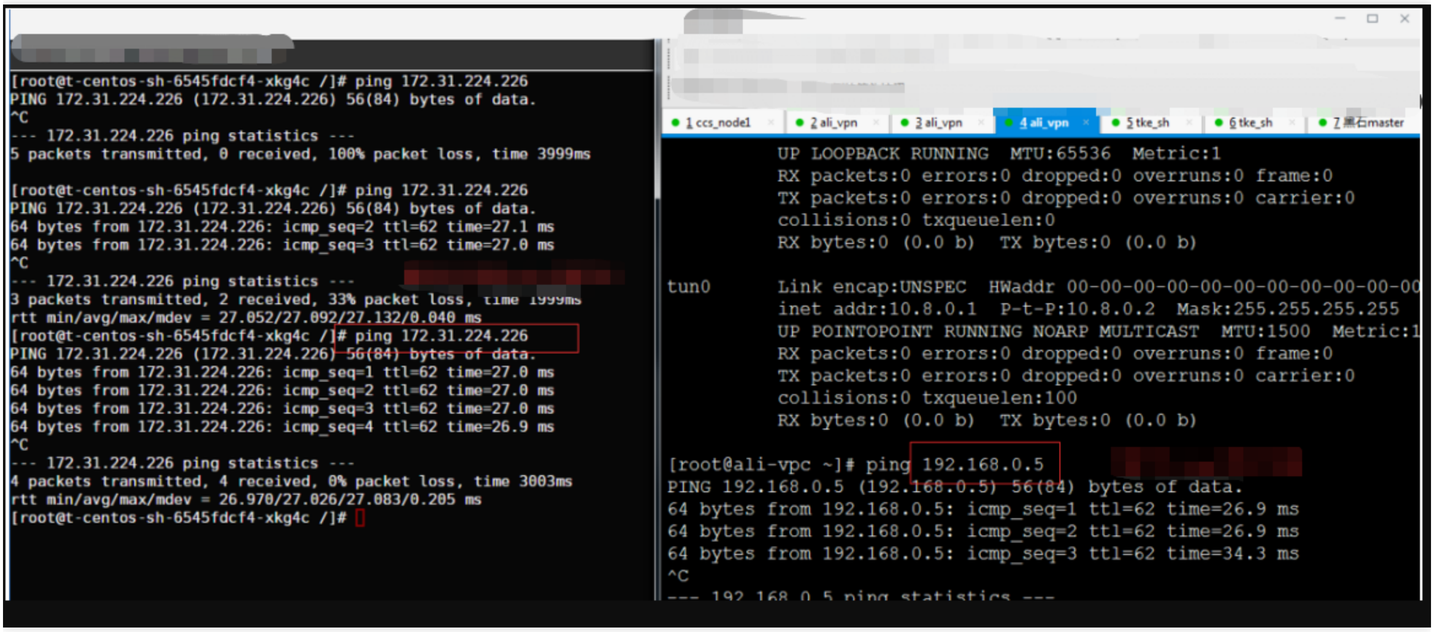
Adding a Container IP Address Range

> Note：
> One subnet can be bound to only one routing table. If multiple routing tables are associated, they will be replaced with the last bound one.

1. On the left sidebar, click **Route Table** to enter the route table management page.
2. Find the route table configured when you set intra-region cross-cluster interconnection or set cross-region cross-cluster interconnection. Click the ID/name of the route table to enter the route table details page.
3. Click **+ New routing policies** to add the container IP range.
4. Select the **Associated Subnets** tab and click **Create Associated Subnet** to associate with the subnet where the CVM instance is.
5. Repeat step 2 to step 4 to add the IP address range where the Tencent Cloud container is located to the opposite routing device.

Expected Result

Containers can communicate with the peer CVM instance as shown below:


[redacted]

```
[root@t-centos-sh-6545fdcf4-xkg4c /]# ping 172.31.224.226
PING 172.31.224.226 (172.31.224.226) 56(84) bytes of data.
^C
--- 172.31.224.226 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 3999ms

[root@t-centos-sh-6545fdcf4-xkg4c /]# ping 172.31.224.226
PING 172.31.224.226 (172.31.224.226) 56(84) bytes of data.
64 bytes from 172.31.224.226: icmp_seq=2 ttl=62 time=27.1 ms
64 bytes from 172.31.224.226: icmp_seq=3 ttl=62 time=27.0 ms
^C
--- 172.31.224.226 ping statistics ---     [redacted]
3 packets transmitted, 2 received, 33% packet loss, time 1999ms
rtt min/avg/max/mdev = 27.052/27.092/27.132/0.040 ms
[root@t-centos-sh-6545fdcf4-xkg4c /]# ping 172.31.224.226
PING 172.31.224.226 (172.31.224.226) 56(84) bytes of data.
64 bytes from 172.31.224.226: icmp_seq=1 ttl=62 time=27.0 ms
64 bytes from 172.31.224.226: icmp_seq=2 ttl=62 time=27.0 ms
64 bytes from 172.31.224.226: icmp_seq=3 ttl=62 time=27.0 ms
64 bytes from 172.31.224.226: icmp_seq=4 ttl=62 time=26.9 ms
^C
--- 172.31.224.226 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 26.970/27.026/27.083/0.205 ms
[root@t-centos-sh-6545fdcf4-xkg4c /]#
```

[redacted]

```
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.1  P-t-P:10.8.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@ali-vpc ~]# ping 192.168.0.5      [redacted]
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=62 time=26.9 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=62 time=26.9 ms
64 bytes from 192.168.0.5: icmp_seq=3 ttl=62 time=34.3 ms
^C
--- 192.168.0.5 ping statistics ---
```

Containers can communicate with VPN opposite servers.

> Note：
> If you want your cloud containers to communicate with your IDC over an IPsec VPN, you need to set the **SPD policy** and **routing table**.

Registering GlobalRouter Mode Cluster to CCN

Last updated：2023-05-06 17:36:46

Cloud Connect Network

Cloud Connect Network (CCN) provides interconnection among Virtual Private Clouds (VPCs) and between VPCs and local IDCs. You can add VPCs and Direct Connect gateways to CCN to achieve single-point access and network-wide resource sharing for a simple, intelligent, secure, and flexible hybrid cloud and globally interconnected network.

Overview

You can register existing container clusters with CCN, and CCN will include the container network into its scope of management. After the container network is fully registered, you can enable or disable the IP range routing of the container network on the CCN side to achieve interconnection between the container clusters and resources in CCN.

Note

After a container cluster is registered with CCN, its IP range will be enabled if the IP range does not conflict with the existing routing in the CCN instance, and will be disabled by default if conflicts exist.

Prerequisites

The VPC of the cluster is already in CCN. For more information about the operations related to CCN, see Operations Overview.

You have evaluated whether the IP range of the cluster network conflicts with the IP ranges of other resources in CCN.

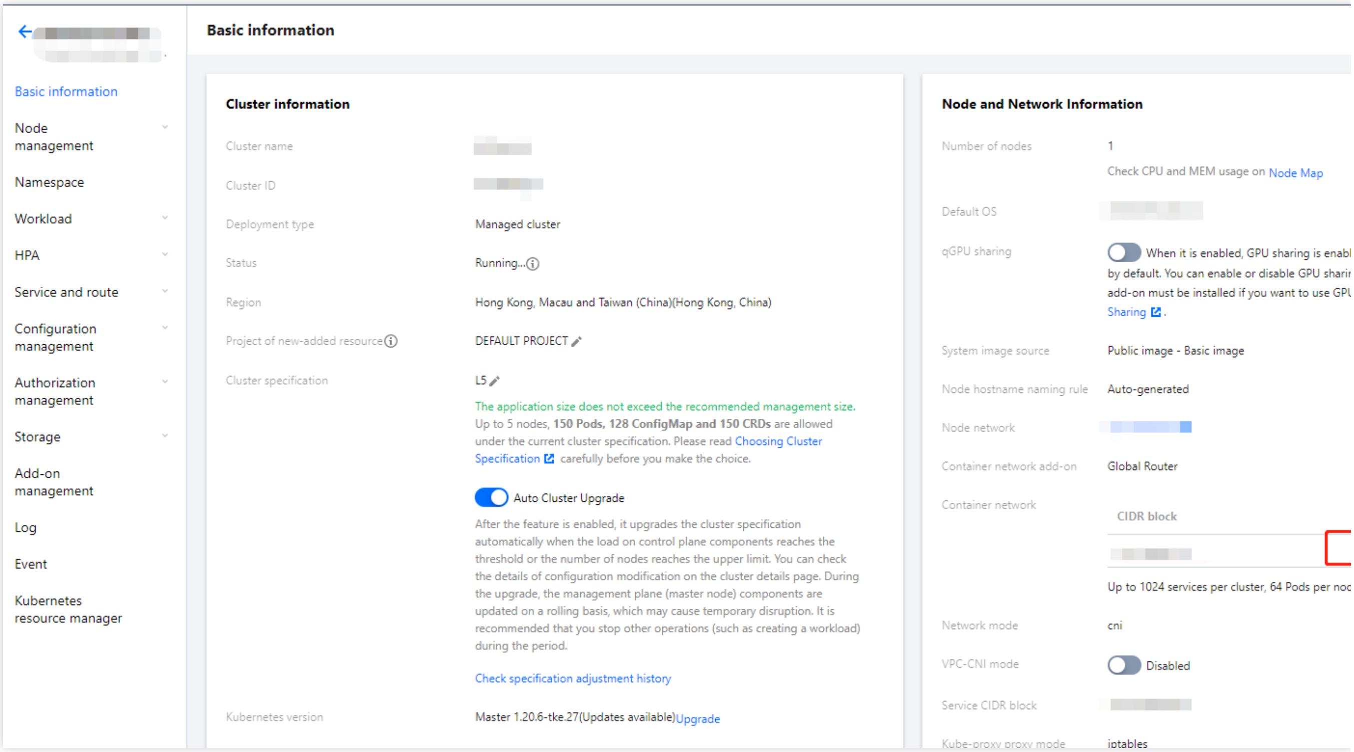
Directions

Registering Container Network to CCN

1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click the ID of the cluster you want to register with CCN, and click **Basic information** on the left to go to the cluster basic information page.
3. Turn on the CCN registration switch to register the container network with CCN as shown below:

Note

This step registers only the container IP range to CCN. Whether or not the routing takes effect depends on your configuration on the CCN side.



Basic information

Cluster information

Cluster name [redacted]
Cluster ID [redacted]
Deployment type Managed cluster
Status Running...
Region Hong Kong, Macau and Taiwan (China)(Hong Kong, China)
Project of new-added resource DEFAULT PROJECT
Cluster specification L5

The application size does not exceed the recommended management size.
Up to 5 nodes, **150 Pods, 128 ConfigMap and 150 CRDs** are allowed under the current cluster specification. Please read Choosing Cluster Specification carefully before you make the choice.

Auto Cluster Upgrade

After the feature is enabled, it upgrades the cluster specification automatically when the load on control plane components reaches the threshold or the number of nodes reaches the upper limit. You can check the details of configuration modification on the cluster details page. During the upgrade, the management plane (master node) components are updated on a rolling basis, which may cause temporary disruption. It is recommended that you stop other operations (such as creating a workload) during the period.

Check specification adjustment history

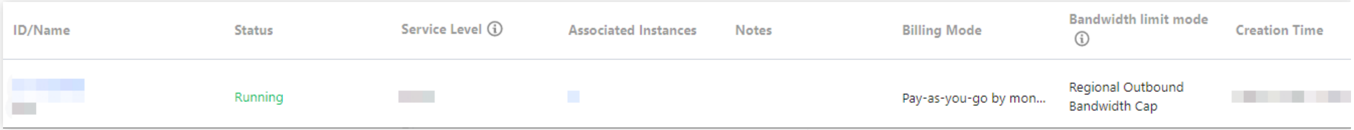
Kubernetes version Master 1.20.6-tke.27(Updates available)Upgrade

Node and Network Information

Number of nodes 1
Check CPU and MEM usage on Node Map
Default OS [redacted]
qGPU sharing When it is enabled, GPU sharing is enabl by default. You can enable or disable GPU sharir add-on must be installed if you want to use GPL Sharing .
System image source Public image - Basic image
Node hostname naming rule Auto-generated
Node network [redacted]
Container network add-on Global Router
Container network CIDR block [redacted]
Up to 1024 services per cluster, 64 Pods per noc
Network mode cni
VPC-CNI mode Disabled
Service CIDR block [redacted]
Kube-proxy proxy mode iptables

Viewing the IP Range Routing of the Container

1. Log in to the VPC console and click CCN in the left sidebar to go to the CCN management page.
2. Click **Manage instances** to the right of the CCN associated with the cluster VPC to go to the CCN instance management page as shown below:



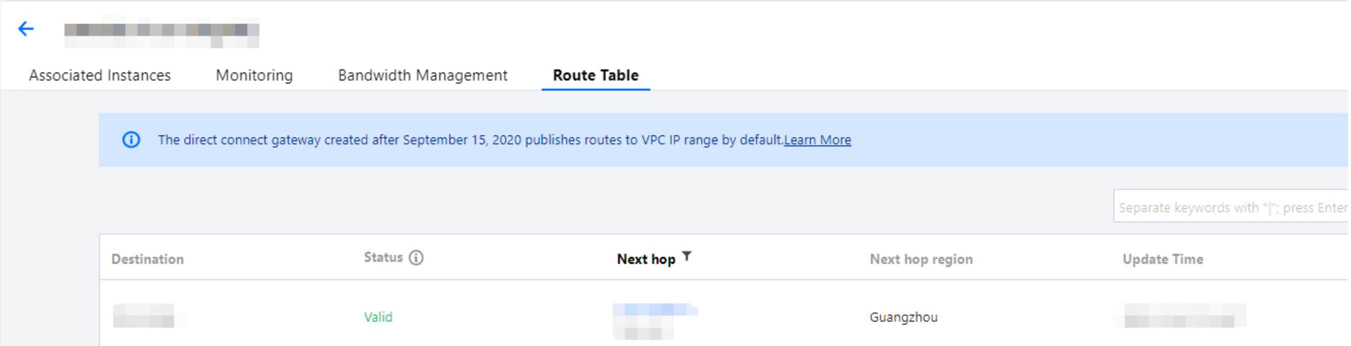
ID/Name	Status	Service Level	Associated Instances	Notes	Billing Mode	Bandwidth limit mode	Creation Time
[redacted]	Running	[redacted]	[redacted]		Pay-as-you-go by mon...	Regional Outbound Bandwidth Cap	[redacted]

3. Click the **Route table** tab to view whether IP range routing is enabled for the container as shown below:

Note

If the IP range of the cluster network does not conflict with the existing routing in the CCN instance, the routing will be enabled by default. If there is an IP range conflict, the routing will be disabled by default.

For more information about routing conflicts, see Use Limits > Route Limits. If you want to enable a route that may lead to routing conflicts, see Enabling Route.



[redacted]

The direct connect gateway created after September 15, 2020 publishes routes to VPC IP range by default.Learn More

Destination	Status	Next hop	Next hop region	Update Time
[redacted]	Valid	[redacted]	Guangzhou	[redacted]

4. Test the interconnection between the container cluster and other resources in CCN.

VPC-CNI Mode

VPC-CNI Mode

Last updated：2022-11-03 15:30:58

How It Works

VPC-CNI is a container network capability provided by TKE based on CNIs and VPC ENIs. It is suitable for scenarios with high latency requirements. In this network mode, containers and nodes are located on the same network plane, and container IP addresses are ENI IP addresses allocated by the IPAMD component.

The VPC-CNI mode includes the shared and exclusive ENI modes for different scenarios, which can be selected as needed.

- Shared ENI mode: Pods share an ENI, and the IPAMD component applies for multiple ENI IP addresses for different Pods. Pod IP addresses can be fixed. For more information, see Static IP Address Features.
- Exclusive ENI mode: Each Pod has an exclusive ENI for higher performance. The number of ENIs that can be used by nodes differs by model. The number is smaller for Pods on a single node.

Use Limits

- We don't recommend subnets in VPC-CNI mode be used by other Tencent Cloud resources such as CVM and CLB instances.
- The nodes in the cluster need to be in the same AZ as the subnet, otherwise, the Pod cannot be scheduled.
- In VPC-CNI mode, the number of Pods that can be scheduled on a node is subject to the maximum number of IP addresses that can be bound to the node ENI and the number of ENIs. The higher the specifications of the node, the more ENIs can be inserted, which can be checked by viewing the **Allocatable** of the node.

Applications

Compared with Global Router, VPC-CNI has the following strengths and use cases:

- It has one layer fewer bridge and 10% higher network forwarding performance and is suitable for latency-sensitive scenarios.
- It supports static Pod IP addresses and is suitable for scenarios that rely on static container IP addresses, for example, migrating a traditional architecture to a container platform and performing security policy restrictions on IP addresses.

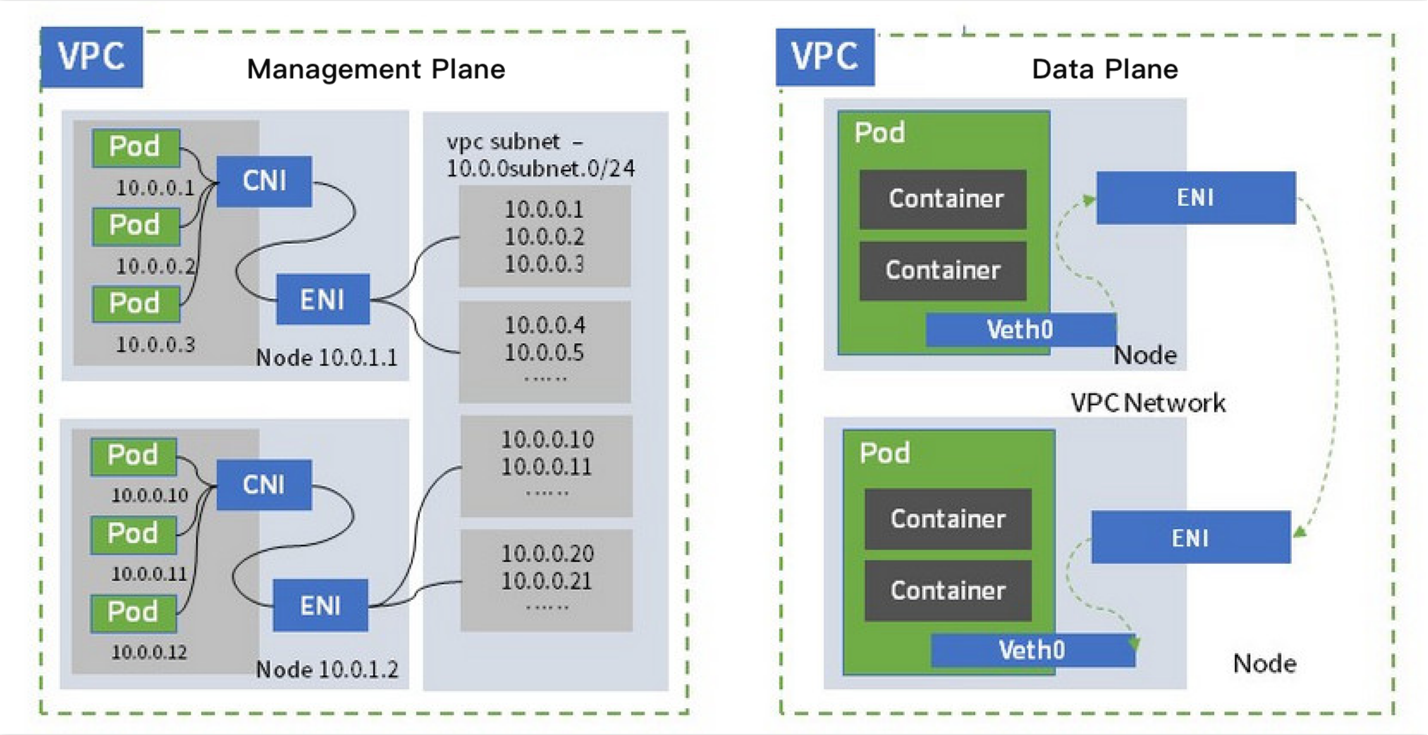
- It supports CLB-to-Pod direct connect.

Multiple Pods with Shared ENI Mode

Last updated：2022-11-03 15:30:58

How It Works

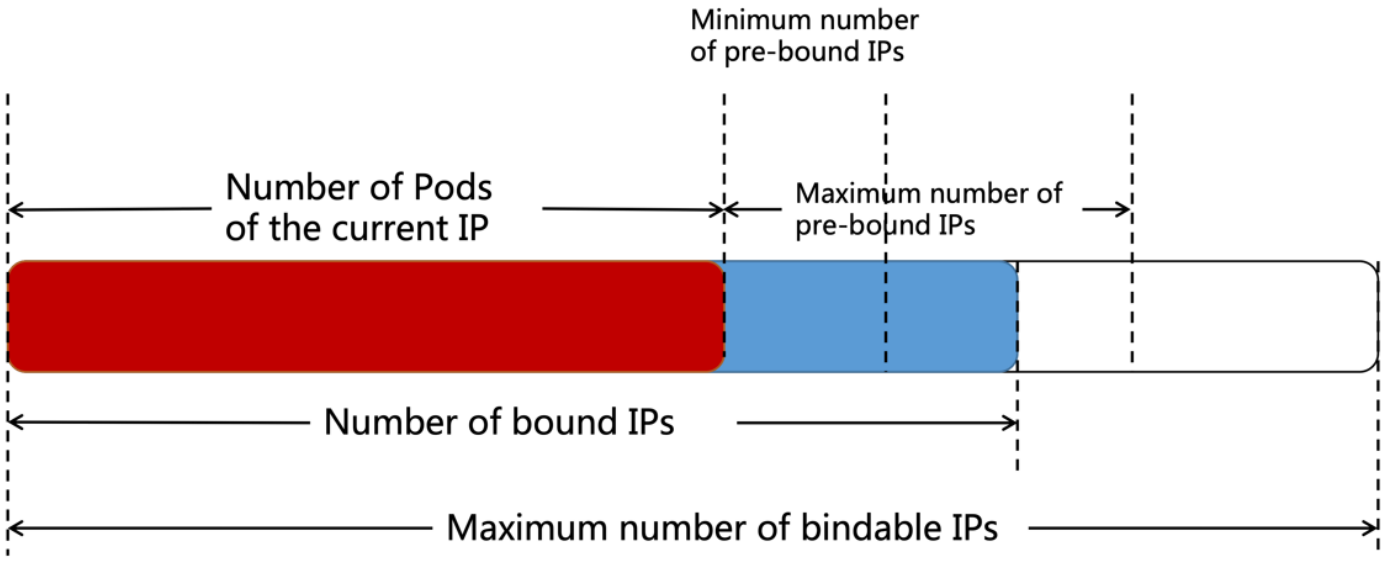
The following diagram illustrates how the multiple Pods with a shared ENI in VPC-CNI mode work.



- The cluster network is the user's VPC, and the nodes and container subnets belong to this VPC.
- The container subnet can select subnets in multiple VPCs.
- You can set whether to enable the static IP address. For more information, see Static IP Address Usage.

IP Address Management Principle

Non-static IP address mode


Minimum number of pre-bound IPs
Number of Pods of the current IP
Maximum number of pre-bound IPs
Number of bound IPs
Maximum number of bindable IPs

- TKE maintains an auto-scaling IP pool on each node. The number of bound IPs ranges from **the number of Pods + the minimum number of pre-bound IPs** to **the number of Pods + the maximum number of pre-bound IPs**.
  - When **the number of bound IPs is less than the number of Pods + the minimum number of pre-bound IPs**, new IPs will be bound to make **the number of bound IPs = the number of Pods + the minimum number of pre-bound IPs**.
  - When **the number of bound IPs is larger than the number of Pods + the maximum number of pre-bound IPs**, an IP will be released about every 2 minutes until **the number of bound IPs = the number of Pods + the maximum number of pre-bound IPs**.
  - When **the maximum number of bindable IPs is less than the number of bound IPs**, the unnecessary idle IPs will be released directly to make **the number of bound IPs equal to the maximum number of bindable IPs**.
- When a Pod with a shared ENI is created, an available IP is randomly allocated from the node's available IP pool.
- When a Pod with a shared ENI is terminated, its IP will be released and returned to the IP pool of the node instead of being released (deleted) in the VPC, so that another Pod can continue to use the IP.
- IPs and ENIs are allocated and released based on the **principle of least ENIs** to ensure that the ENIs in use are as few as possible:
  - **Allocate an IP to a Pod**: preferentially allocate IPs on the ENI with the most allocated IPs.
  - **Release an IP**: preferentially release IPs on the ENI with the least allocated IPs.
  - **Bind to a new ENI**: if the IP quota of the bound ENI is exhausted or the IPs of the subnet where the ENI is located is used up, you can apply for a new ENI.

  - **Release an ENI**: if all secondary IPs on the bound ENI have been unbound and you do not want to add IPs, you can unbind and delete the ENI.
- The expansion resource `tke.cloud.tencent.com/eni-ip` will be registered for the node. The allocatable number of the resource is the number of IPs that have been bound. The capacity is the maximum number of IPs that can be bound to the node. Therefore, if a Pod fails to be scheduled to a node, it indicates that the IPs of the node have been used up.
- Select a subnet for a new ENI: preferentially select the subnet with the most available IPs.
- The maximum number of bindable IPs of each node = the maximum number of bound ENIs * the number of bindable IPs of a single ENI
- Currently, **the minimum number of pre-bound IPs** and **the maximum number of pre-bound IPs** are set to five respectively by default.

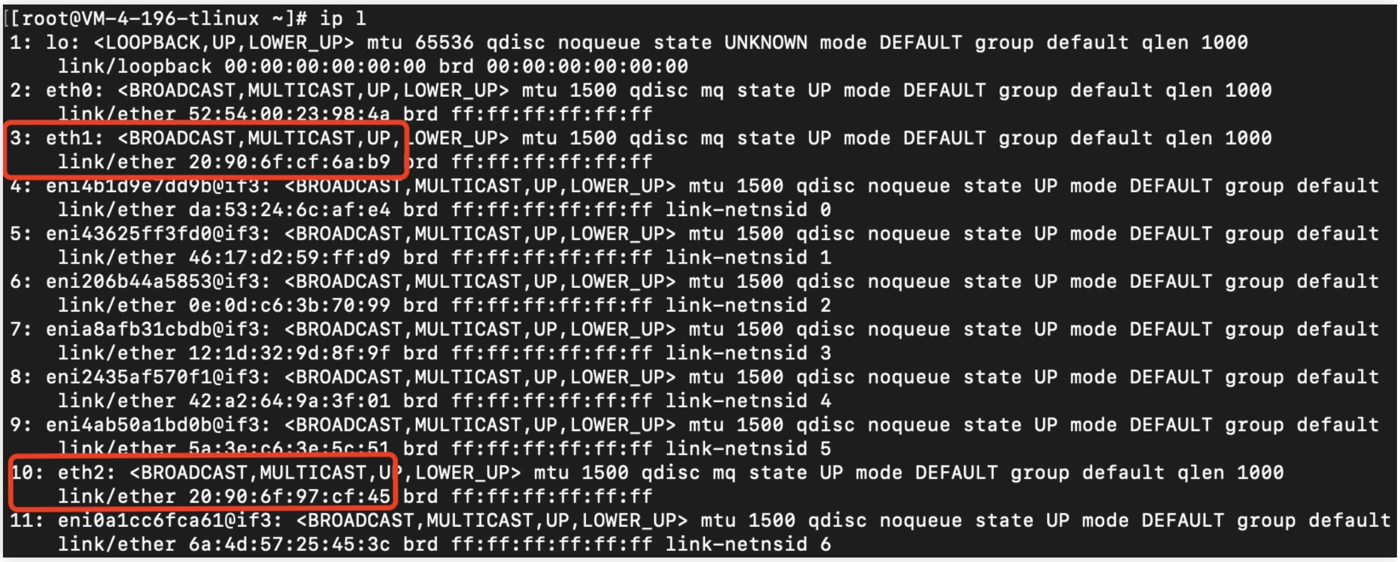
Static IP address mode

- TKE network component maintains an IP pool available at the cluster level.
- Each newly added node in the cluster will not be bound to any secondary IP address or ENI in advance, and IP addresses are totally **allocated on demand**.
- When a Pod in VPC-CNI mode is created, the IPAMD component will find an available ENI for IP address allocation on the corresponding node. The allocation follows the principle of **least ENIs**, that is, the ENI bound to the most number of IP addresses is allocated first.
- If the existing ENI is bound to a maximum number of IP addresses, create an ENI for IP address allocation. The subnet that has the largest number of available IP addresses is preferred for the ENI.
- If the Pod without a static IP address annotation is terminated, the IP address will be returned to the available IP pool of the cluster, the unbinding of the IP address from the ENI will be triggered, and the IP address will be released and returned to the VPC subnet.
- The static IP address of the terminated Pod will be retained in the VPC, and this IP address will be used again when a Pod with the same name as the terminated Pod is created.
- When the node is deleted, the IP addresses occupied by the ENI will be released.
- When there are multiple container subnets, the ENI is preferentially allocated to the subnet that has the largest number of available IP addresses. If there is no such subnet, the ENI binding will fail.

Data Plane Principle for Multiple ENIs

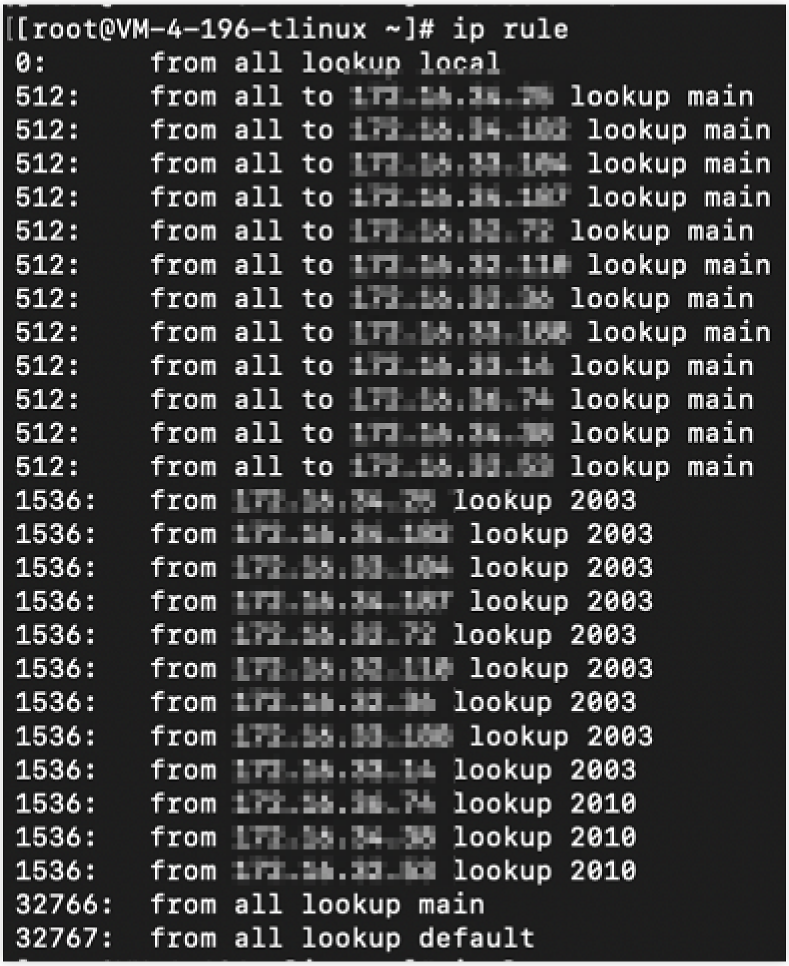
When a node has bound multiple ENIs, the network packets sent from the Pod will be forwarded to the corresponding ENI according to the policy-based routing.

- You can execute `ip link` on the node to view the information of all network devices on the node, and you can learn about the network devices corresponding to the ENI of the node through comparison of the mac address of the ENI. Generally, `eth0` represents primary ENI, `eth1` and `eth2` represent secondary ENIs:



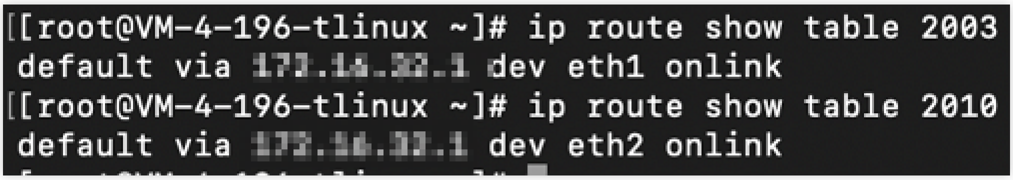
```
[root@VM-4-196-tlinux ~]# ip l
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:23:98:4a brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 20:90:6f:cf:6a:b9 brd ff:ff:ff:ff:ff:ff
4: eni4b1d9e7dd9b@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether da:53:24:6c:af:e4 brd ff:ff:ff:ff:ff:ff link-netnsid 0
5: eni43625ff3fd0@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 46:17:d2:59:ff:d9 brd ff:ff:ff:ff:ff:ff link-netnsid 1
6: eni206b44a5853@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 0e:0d:c6:3b:70:99 brd ff:ff:ff:ff:ff:ff link-netnsid 2
7: enia8afb31cbdb@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 12:1d:32:9d:8f:9f brd ff:ff:ff:ff:ff:ff link-netnsid 3
8: eni2435af570f1@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 42:a2:64:9a:3f:01 brd ff:ff:ff:ff:ff:ff link-netnsid 4
9: eni4ab50a1bd0b@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 5a:3e:c6:3e:5c:51 brd ff:ff:ff:ff:ff:ff link-netnsid 5
10: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 20:90:6f:97:cf:45 brd ff:ff:ff:ff:ff:ff
11: eni0a1cc6fca61@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default
    link/ether 6a:4d:57:25:45:3c brd ff:ff:ff:ff:ff:ff link-netnsid 6
```

- You can execute `ip rule` on the node to view the information of policy-based route table. TKE network component obtains the number of the route table through `<link index="">+2000` of the ENI. Network packets sent from the Pod that is bound to the corresponding ENI IP will be forwarded to this route table. In this example, the route table corresponding to `eth1` is 2003, and the route table corresponding to `eth2` is 2010.



```
[[root@VM-4-196-tlinux ~]# ip rule
 0:      from all lookup local
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 512:    from all to [redacted] lookup main
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2003
 1536:   from [redacted] lookup 2010
 1536:   from [redacted] lookup 2010
 1536:   from [redacted] lookup 2010
 32766:  from all lookup main
 32767:  from all lookup default
```

- The default routing to the corresponding ENI is set for the corresponding route table. You can execute `ip route show table <id>` on the node to view it:



```
[[root@VM-4-196-tlinux ~]# ip route show table 2003
 default via [redacted] dev eth1 onlink
[[root@VM-4-196-tlinux ~]# ip route show table 2010
 default via [redacted] dev eth2 onlink
```

When the network packets that are sent to the Pod reach the node, they will be sent to the Veth ENI of the Pod via the primary route table by following the policy-based routing.

How to Use

To use VPC-CNI, ensure that `rp_filter` is disabled. You can refer to the following code sample:

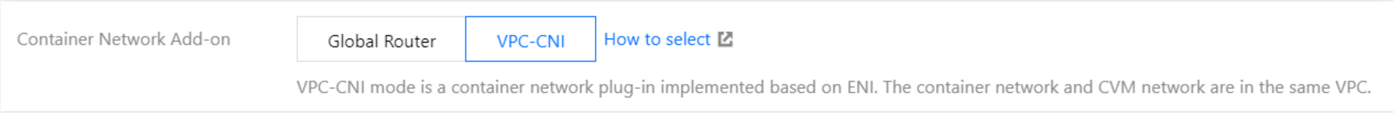
```
sysctl -w net.ipv4.conf.all.rp_filter=0
# Assume that eth0 is the primary ENI
sysctl -w net.ipv4.conf.eth0.rp_filter=0
```

> Note：
> The `tke-eni-agent` component automatically sets the node kernel parameters. If you have manually maintained the kernel parameters and enabled `rpfilter` , network connection would fail.

Enabling VPC-CNI

Enabling VPC-CNI when creating the cluster

1. Log in to the TKE console and select **Cluster** on the left sidebar.
2. On the "Cluster management" page, click **Create** above the cluster list.
3. On "Create Cluster" page, select **VPC-CNI** for **Container Network Add-on**, as shown below:



Container Network Add-on: Global Router | VPC-CNI (selected)

VPC-CNI mode is a container network plug-in implemented based on ENI. The container network and CVM network are in the same VPC.

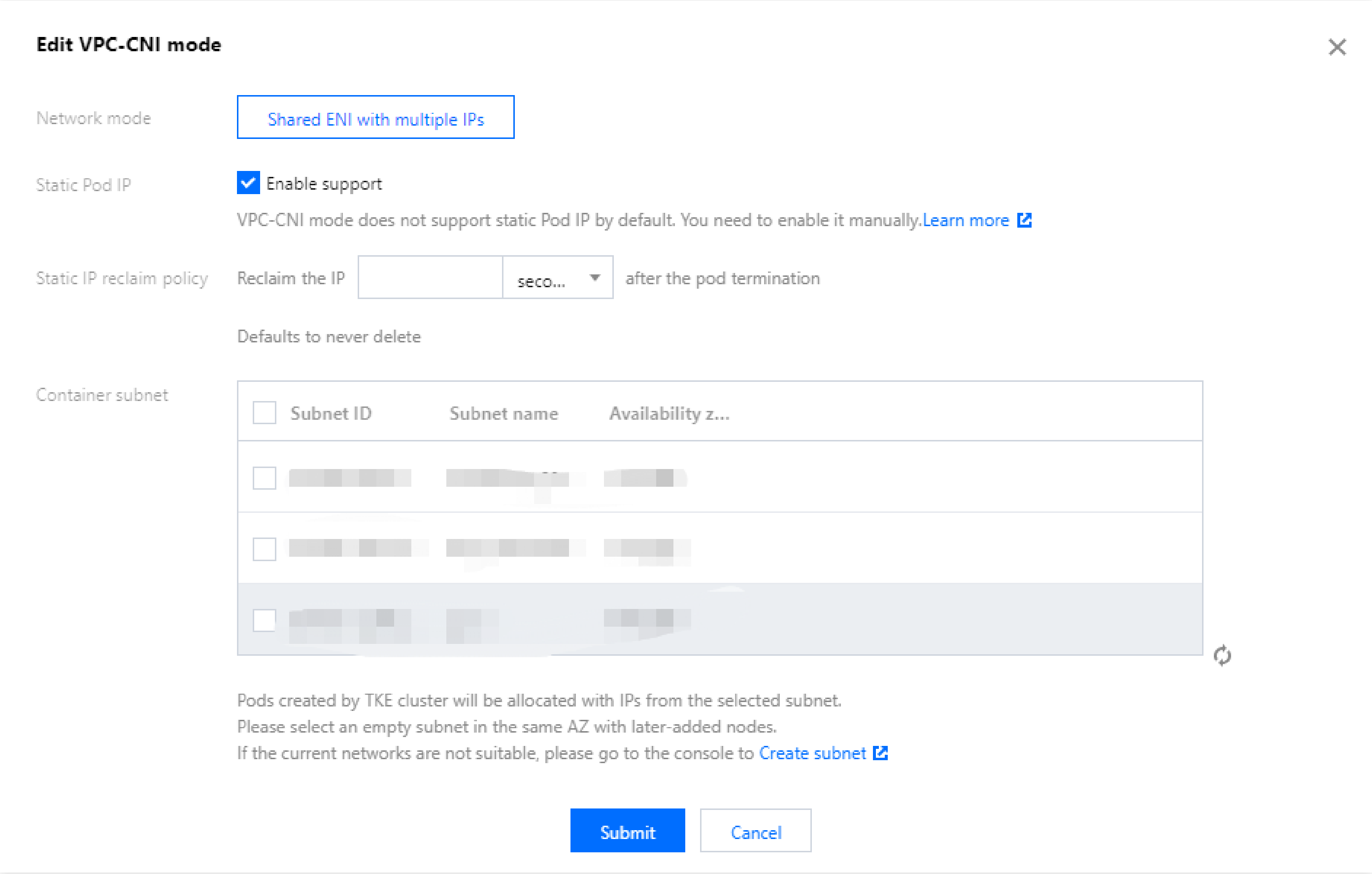
> Note：
> By default, the VPC-CNI mode **does not enable the static IP address**. You can enable the static IP address only when Creating a Cluster. If you need to enable the static Pod IP address for the cluster, see Static IP Address Usage.

Enabling VPC-CNI for the existing clusters

When creating a cluster, select the Global Router network add-on. Then, enable the VPC-CNI mode on the basic information page of the cluster (by default, both modes are enabled).

1. Log in to the TKE console and select **Cluster** on the left sidebar.
2. On "Cluster Management" page, select the ID of the cluster for which VPC-CNI needs to be enabled and go to its details page.
3. On the cluster details page, click **Basic Information** on the left.
4. In the **Node and Network Information** section, enable **VPC-CNI mode**.

5. In the pop-up window, specify whether to support static IP addresses and select the subnet as shown below:



Edit VPC-CNI mode

Network mode: Shared ENI with multiple IPs

Static Pod IP: ☑ Enable support

VPC-CNI mode does not support static Pod IP by default. You need to enable it manually.Learn more

Static IP reclaim policy: Reclaim the IP [ ] seco... after the pod termination

Defaults to never delete

Container subnet:

□ Subnet ID	Subnet name	Availability z...
□ [redacted]	[redacted]	[redacted]
□ [redacted]	[redacted]	[redacted]
□ [redacted]	[redacted]	[redacted]

Pods created by TKE cluster will be allocated with IPs from the selected subnet.
Please select an empty subnet in the same AZ with later-added nodes.
If the current networks are not suitable, please go to the console to Create subnet

> Note：
>
> - For scenarios that use static IP addresses, when enabling VPC-CNI, you need to set the IP reclaiming policy to specify when to reclaim the IP addresses after Pods are terminated.
> - Pods with non-static IP addresses are not affected by these settings because their IP addresses are immediately released upon Pod termination. These IP addresses are not returned to the VPC, but returned to the IP address pool managed by the container.

6. Click **Submit** to enable VPC-CNI mode for the cluster.

Disabling VPC-CNI

1. Log in to the TKE console and select **Cluster** on the left sidebar.
2. On "Cluster Management" page, select the ID of the cluster for which VPC-CNI needs to be enabled and go to its details page.
3. On the cluster details page, click **Basic Information** on the left.

4. In the **Node and Network Information** section, disable the **VPC-CNI mode**.
5. Click **Submit** in the pop-up window to disable the VPC-CNI mode.

Pods with Exclusive ENI Mode

Last updated：2022-06-14 14:52:20

Based on the original single ENI with multiple IP addresses of VPC-CNI mode, the Pod with exclusive ENI mode supports that the container directly uses the ENI exclusively. It can seamlessly connect with all features of Tencent Cloud's VPC products, while making a great improvement in performance.

> Note：
> This feature is currently in beta. If you want to try it out, please submit a ticket.

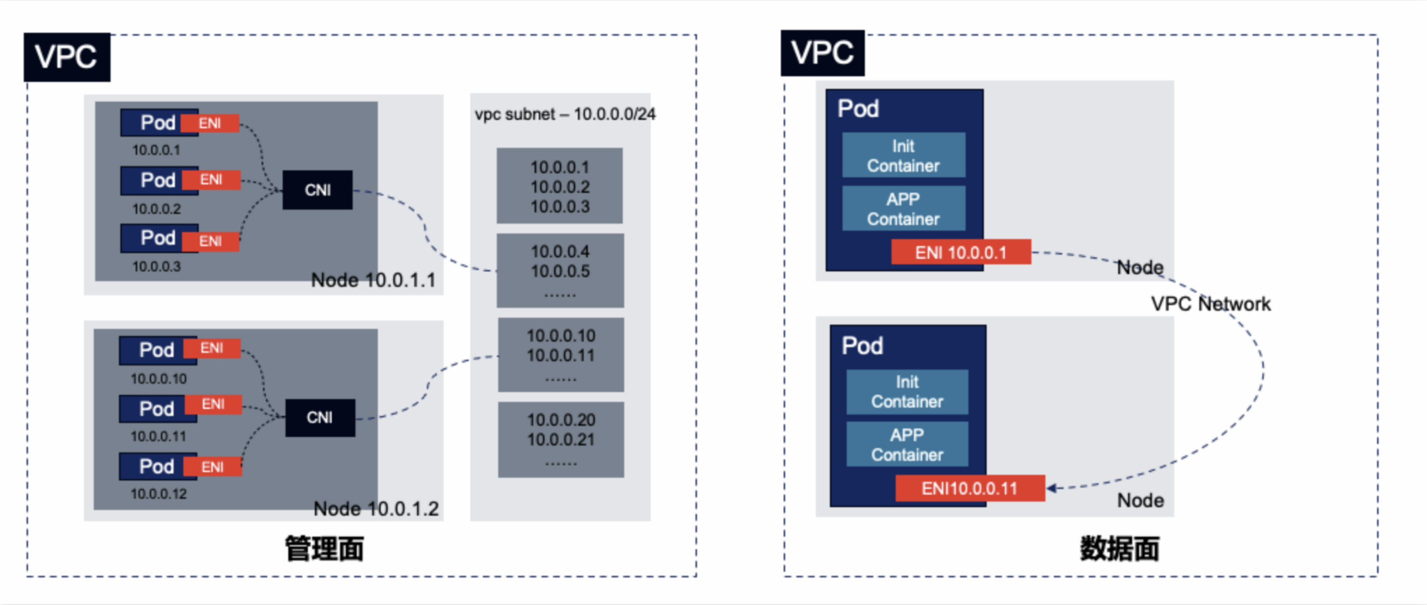
Overview

The new VPC-CNI mode network solution has the following new capabilities:

- Pod can be directly bound to EIP/NAT, and no longer relies on the public network access capability of the node, nor does it need SNAT, which can be applicable to the public network access scenarios with high concurrency and high bandwidth such as Crawler and video conference.
- It supports static IP address based on Pod name. The IP address can remain unchanged after Pod is scheduled and restarted.
- It supports CLB-to-Pod direct connection without forwarding through NodePort, so as to improve forwarding performance and provide a unified CLB view.

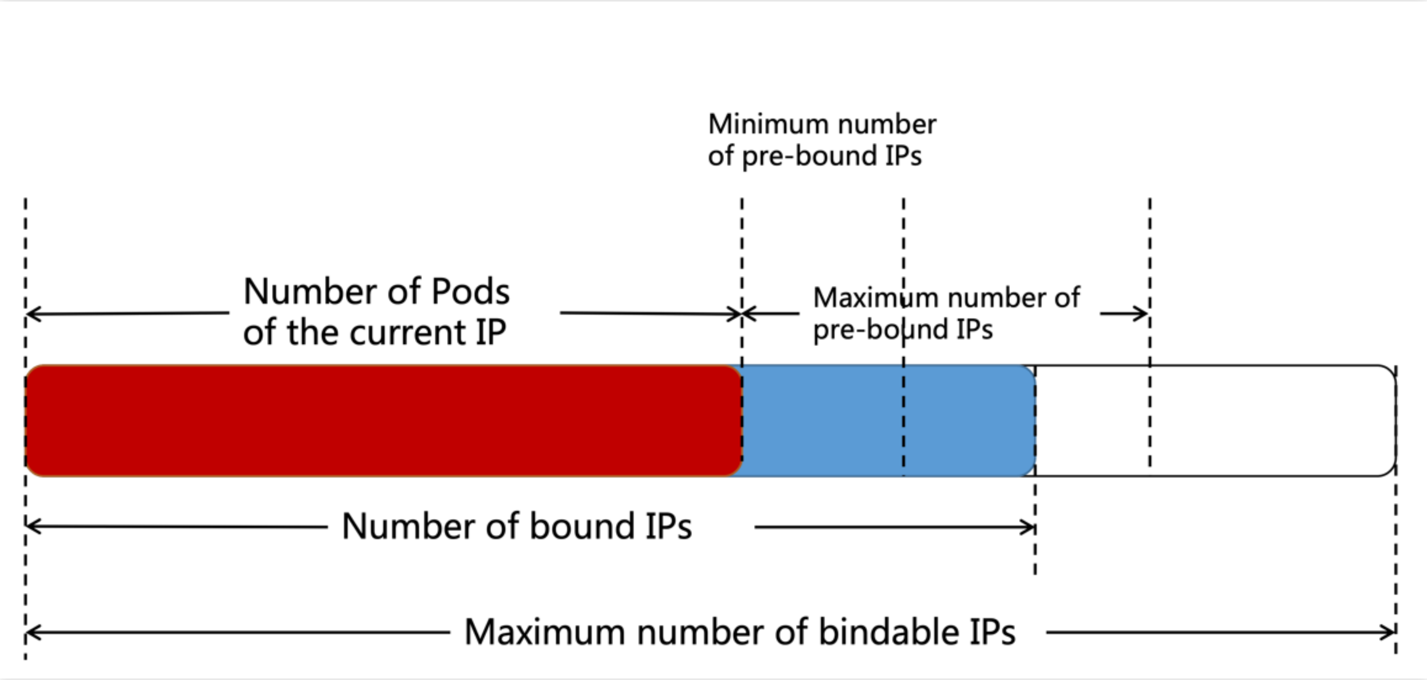
Method

The new VPC-CNI mode network solution is an extension of the original VPC-CNI mode. Relying on the ENI, the ENI bound to the node can be configured to the container network namespace through CNI, so that the container can directly use the ENI exclusively. The principle is shown in the figure below:



IP Address Management Principle

Non-static IP address mode


Minimum number of pre-bound IPs
Number of Pods of the current IP
Maximum number of pre-bound IPs
Number of bound IPs
Maximum number of bindable IPs

- TKE maintains an auto-scaling ENI pool on each node. The number of bound ENIs ranges from **the number of Pods + the minimum number of pre-bound ENIs** to **the number of Pods + the maximum number of pre-**

bound ENIs.
  - When the number of bound ENIs is less than the number of Pods + the minimum number of pre-bound ENIs, new ENIs will be bound to make the number of bound ENIs = the number of Pods + the minimum number of pre-bound ENIs.
  - When the number of bound ENIs is larger than the number of Pods + the maximum number of pre-bound ENIs, an ENI will be released about every 2 minutes until the number of bound ENIs = the number of Pods + the maximum number of pre-bound ENIs.
  - When the maximum number of bindable ENIs is less than the number of bound ENIs, the unnecessary idle ENIs will be released directly to make the number of bound ENIs equal to the maximum number of bindable ENIs.
- When a Pod with an exclusive ENI is created, an available ENI is randomly allocated from the node's available ENI pool.
- When a Pod with an exclusive ENI is terminated, its ENI will be released and returned to the ENI pool of the node instead of being released (deleted) in the VPC, so that another Pod can continue to use the ENI.
- When the node is deleted, all bound ENIs will be released (deleted).
- When there are multiple container subnets, the ENI is preferentially allocated to the subnet with the largest number of available IP addresses.

Static IP address mode

- TKE does not maintain an ENI pool on each node, and the ENI is not pre-bound to the node.
- When a Pod with an exclusive ENI is created, an ENI is directly bound to the node and used by this Pod.
- When a Pod with an exclusive ENI of the non-static IP address is terminated, the ENI used by the Pod will be deleted and released directly in the VPC. When a Pod with an exclusive ENI of the static IP address is terminated, the ENI will only be unbound, but not be deleted and released.
- When the node is deleted, all bound ENIs will be released (deleted).
- When there are multiple container subnets, the ENI is preferentially allocated to the subnet with the largest number of available IP addresses.

Use Limitations

- The network mode is only available to some models such as S5, SA2, IT5 and SA3.
- The number of Pods with exclusive ENIs running on a node is limited by the number of ENIs that can be bound to model. The maximum number of Pods with exclusive ENIs running on a node = the maximum number of bindable ENIs - 1.For more information, see Limits on the Number of Pods in VPC-CNI Mode.
- The new network solution is only suitable for the new TKE clusters.
- There are unified restrictions on the VPC-CNI mode:
  - You need to plan a dedicated subnet for containers, and the subnet is not recommended to be shared with other Tencent Cloud services such as CVMs and CLBs.

  - The nodes in the cluster need to be in the same AZ as the subnet, otherwise, the Pod cannot be scheduled.

Static IP Address Mode Instructions
Static IP Address Usage

Last updated：2022-11-03 15:40:01

Overview

This mode is suitable for scenarios that rely on static container IP addresses, for example, migrating a traditional architecture to a container platform and performing security policy restrictions on IP addresses. The static IP address mode is not recommended for services without IP address limits.

Features and Limitations

- The static IP address is achieved by retaining the associated IP address when the Pod is terminated, or keeping the IP unchanged when the Pod is migrated.
- Pods in a cluster can be in different subnets, but Pods with static IP addresses cannot be scheduled across node availability zones and across subnets.
- The IP address of Pod can automatically associate with EIP, thus Pod can be accessed via internet.
- For the static IP addresses with shared ENI, when the Pod with static IP address is terminated, its IP address is only retained in the cluster. If other clusters or services (such as CVM, CDB, CLB) use the same subnet, the retained static IP address may be occupied, and the Pod will be unable to obtain the IP address when it being restarted. **Please ensure that the container subnet of this mode is exclusively used.**

How to Use

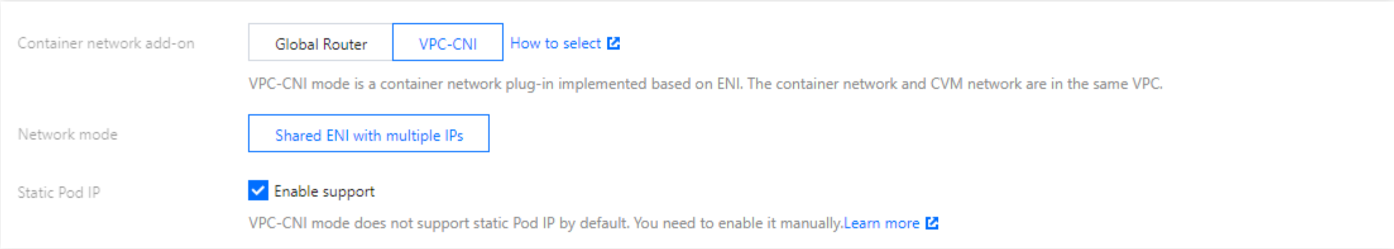
You can enable the static IP address using either of the following methods:

- Select VPC-CNI with static IP address when creating a cluster
- Enable VPC-CNI with static IP address for GlobalRouter mode

Selecting static IP address of VPC-CNI mode when creating a cluster

Note：
If you use this method to enable VPC-CNI, when you create a workload on the console or through YAML, all Pods will use ENIs by default.

1. Log in to the TKE console and select **Cluster** on the left sidebar.
2. On the "Cluster management" page, click **Create** above the cluster list.
3. On "Create Cluster" page, select **VPC-CNI** for **Container Network Add-on**.
4. Select **VPC-CNI** for "Container Network Add-on". Check **Enable Support** for **Static Pod IP**, as shown in the figure below:



Container network add-on: Global Router | VPC-CNI | How to select

VPC-CNI mode is a container network plug-in implemented based on ENI. The container network and CVM network are in the same VPC.

Network mode: Shared ENI with multiple IPs

Static Pod IP: ☑ Enable support

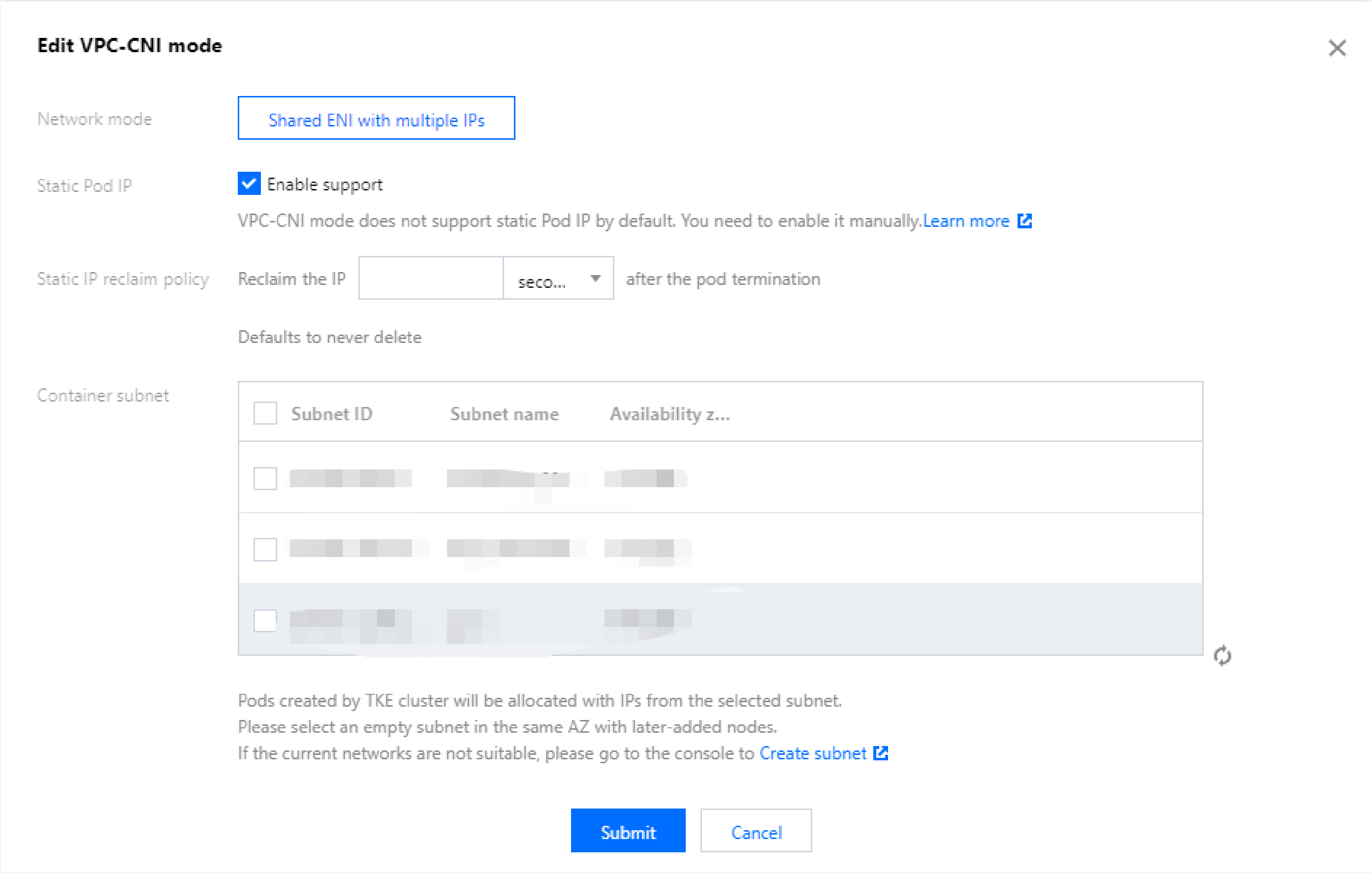
VPC-CNI mode does not support static Pod IP by default. You need to enable it manually.Learn more

Enabling VPC-CNI with static IP address for GlobalRouter mode

Enabling VPC-CNI for the existing clusters

> Note：
>
> - Enable VPC-CNI Mode with static IP address for GlobalRouter, that is, when creating a cluster, you select the Global Router network add-on, and then enable the VPC-CNI mode (both modes can be used at the same time by default) on the basic information page of the cluster.
> - If you use this method to enable VPC-CNI, the Pods cannot use ENIs by default.

1. Log in to the TKE console and select **Cluster** on the left sidebar.
2. On "Cluster Management" page, select the ID of the cluster for which VPC-CNI needs to be enabled and go to its details page.
3. On the cluster details page, click **Basic Information** on the left.
4. In the **Node and Network Information** section, enable **VPC-CNI mode**.
5. In the pop-up window, select **Enable Support**, confirm the IP address reclaim policy, and select the subnet as shown below:



Edit VPC-CNI mode

Network mode: Shared ENI with multiple IPs

Static Pod IP: ☑ Enable support

VPC-CNI mode does not support static Pod IP by default. You need to enable it manually.Learn more

Static IP reclaim policy: Reclaim the IP [ ] seco... after the pod termination

Defaults to never delete

Container subnet:

□	Subnet ID	Subnet name	Availability z...
□	[redacted]	[redacted]	[redacted]
□	[redacted]	[redacted]	[redacted]
□	[redacted]	[redacted]	[redacted]

Pods created by TKE cluster will be allocated with IPs from the selected subnet.
Please select an empty subnet in the same AZ with later-added nodes.
If the current networks are not suitable, please go to the console to Create subnet

Note：

- For scenarios that use static IP addresses, when enabling VPC-CNI, you need to set the IP reclaiming policy to specify when to reclaim the IP addresses after Pods are terminated.
- Pods with non-static IP addresses are not affected by these settings because their IP addresses are immediately released upon Pod termination. These IP addresses are not returned to the VPC, but returned to the IP address pool managed by the container.

6. Click **Submit** to enable VPC-CNI mode for the cluster.

Creating StatefulSets with static Pod IP addresses

In GlobalRouter mode with VPC-CNI enabled, if you have applications to deploy in TKE, which need to use the static Pod IP addresses, you can create a StatefulSets with static IP addresses. Pod created by this type of StatefulSet are assigned with an actual IP address in the VPC through an ENI. The IP addresses are assigned by TKE VPC-CNI add-on. So that when the Pod is restarted or migrated, the IP address can be unchanged.

By using StatefulSets with static IP addresses, you can:

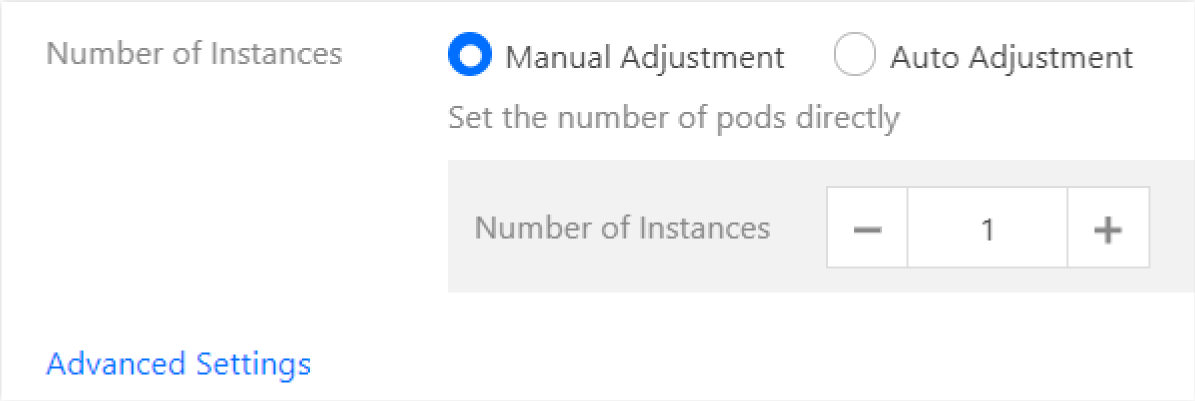
- Authorize based on source IP addresses.
- Review processes based on IP addresses.
- Query logs based on Pod IP addresses.

> Note：
> When StatefulSets with static IP addresses are used, the static IP addresses survive only within the lifecycle of their StatefulSets.

You can create the static IP address using either of the following methods:

- Creating StatefulSets with Static IP Addresses via TKE console

  i. Log in to the TKE console and select **Cluster** on the left sidebar.
  ii. Select a cluster ID that needs to use the static IP address and go to its management page.
  iii. Choose **Workload** > **StatefulSet** to go to the cluster management page for StatefulSet.
  iv. Click **Create** and view **Number of Instances**, as shown in the figure below.



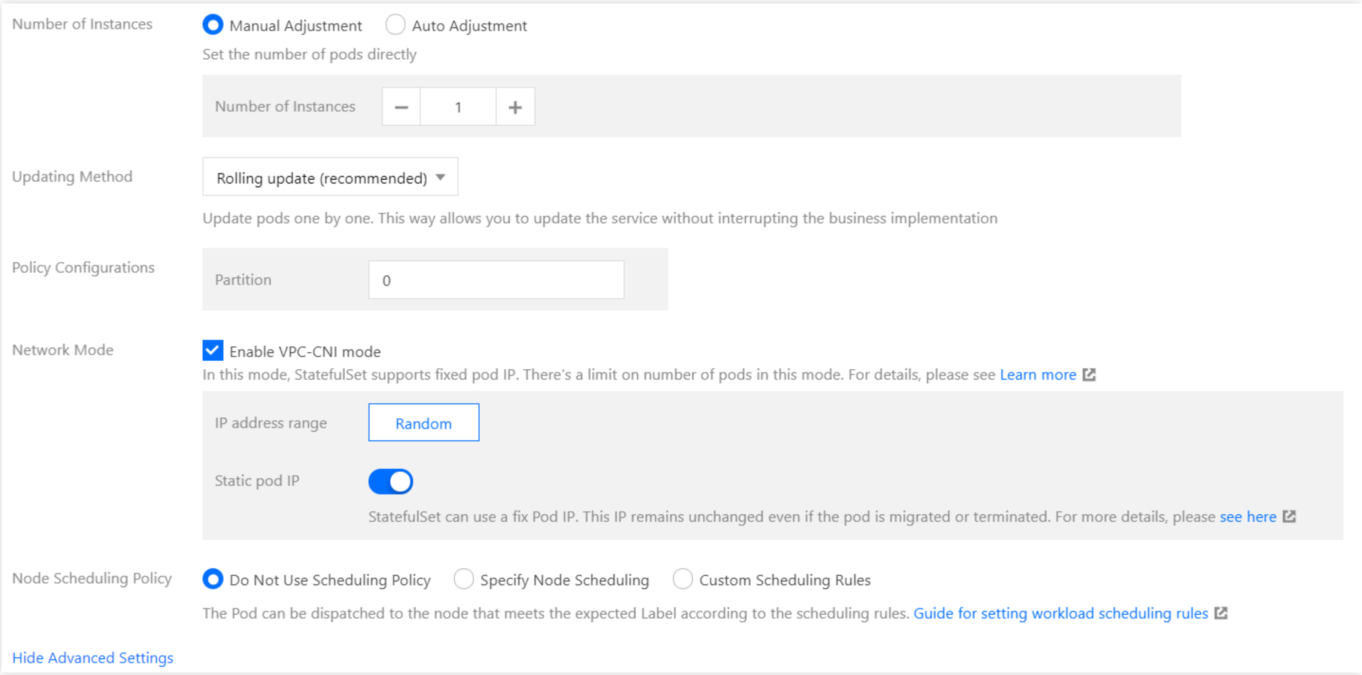
Number of Instances ● Manual Adjustment ○ Auto Adjustment

Set the number of pods directly

Number of Instances − 1 +

Advanced Settings

v. Click **Advanced Settings** and set **StatefulSet** parameters as needed. The key parameters are as follows:



Number of Instances: ● Manual Adjustment ○ Auto Adjustment

Set the number of pods directly

Number of Instances: − 1 +

Updating Method: Rolling update (recommended)

Update pods one by one. This way allows you to update the service without interrupting the business implementation

Policy Configurations: Partition: 0

Network Mode: ☑ Enable VPC-CNI mode

In this mode, StatefulSet supports fixed pod IP. There's a limit on number of pods in this mode. For details, please see Learn more

IP address range: Random

Static pod IP: ●

StatefulSet can use a fix Pod IP. This IP remains unchanged even if the pod is migrated or terminated. For more details, please see here

Node Scheduling Policy: ● Do Not Use Scheduling Policy ○ Specify Node Scheduling ○ Custom Scheduling Rules

The Pod can be dispatched to the node that meets the expected Label according to the scheduling rules. Guide for setting workload scheduling rules

Hide Advanced Settings

- Network mode: select **Enable VPC-CNI mode**.
  - IP address range: currently, only the **Random** value is supported.
  - Static pod IP: select **Enable**.

- Creating via YAML

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
labels:
k8s-app: busybox
name: busybox
namespace: default
spec:
replicas: 3
selector:
matchLabels:
k8s-app: busybox
qcloud-app: busybox
serviceName: ""
template:
metadata:
annotations:
tke.cloud.tencent.com/networks: "tke-route-eni"
tke.cloud.tencent.com/vpc-ip-claim-delete-policy: Never
```

```
creationTimestamp: null
labels:
k8s-app: busybox
qcloud-app: busybox
spec:
containers:
- args:
- "10000000000"
command:
- sleep
image: busybox
imagePullPolicy: Always
name: busybox
resources:
limits:
tke.cloud.tencent.com/eni-ip: "1"
requests:
tke.cloud.tencent.com/eni-ip: "1"
```

- spec.template.annotations: `tke.cloud.tencent.com/networks: "tke-route-eni"` indicates that the Pod uses the VPC-CNI mode with shared ENI. If you use the VPC-CNI mode with independent ENI, please modify the value to `"tke-direct-eni"` .
- spec.template.annotations: to create Pods in VPC-CNI mode, you need to set the annotation `tke.cloud.tencent.com/vpc-ip-claim-delete-policy` . Its default value is “Immediate”, that is, when a Pod is terminated, the associated IP address is also terminated. To use a static IP address, set it to “Never”, that is, a Pod is terminated, but the associated IP address will be retained. When a Pod with the same name as the terminated Pod is pulled the next time, the original IP address is used.
- spec.template.spec.containers.0.resources: to create Pods with shared ENI in VPC-CNI mode, you need to add "requests" and "limits", that is, `tke.cloud.tencent.com/eni-ip` . If you are using the VPC-CNI mode with independent ENI, add `tke.cloud.tencent.com/direct-eni` .

Static IP Address Features

Last updated：2023-02-23 18:34:01

Retaining and Reclaiming a Static IP Address

In static IP address mode, after a Pod in VPC-CNI mode is created and used, the network component will create the CRD object `VpcIPClaim` with the same name as the Pod in the same namespace. This object describes the Pod's requirements for the IP address. The network component will then create the CRD object `VpcIP` based on the object and associate it with the corresponding `VpcIPClaim` . `VpcIP` is the name of the actual IP address, indicating that an actual IP address is occupied.

You can run the following command to view IP address usage in the container subnet of the cluster:

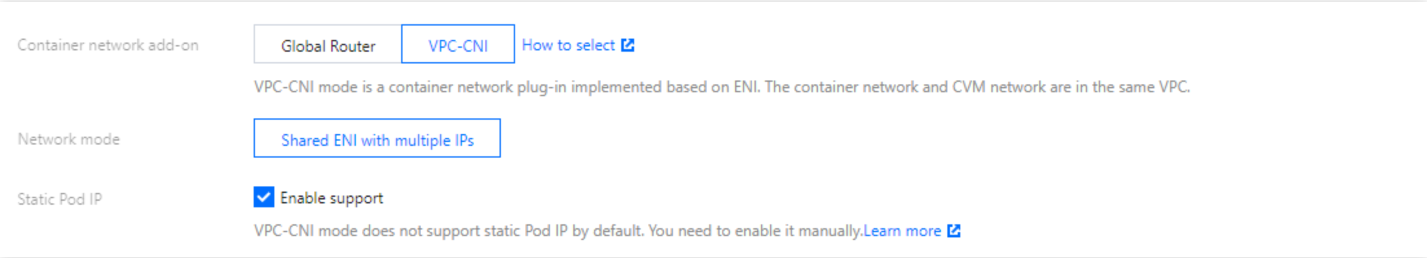
```
kubectl get vip
```

For a Pod to which a non-static IP address is bound, `VpcIPClaim` will be terminated and `VpcIP` will be terminated and reclaimed after the Pod is terminated. For a Pod to which a static IP address is bound, `VpcIPClaim` and `VpcIP` will be retained after the Pod is terminated. After the Pod with the same name is started, it will use the `VpcIP` associated with the `VpcIPClaim` with the same name, so as to retain the IP address.

As the network component will look for available IP addresses based on `VpcIP` during IP address allocation in the cluster, static IP addresses need to be reclaimed promptly if not used (the current default policy indicates never to reclaim); otherwise, IP addresses will be wasted, and no IP addresses will be available. This document describes reclaiming after expiration, manual reclaiming, and cascade reclaiming of an IP address.

Reclaiming after expiration

On Creating a Cluster page, select **VPC-CNI** for **Container Network Add-on** and check **Enable Support** for **Static Pod IP**, as shown in the figure below:

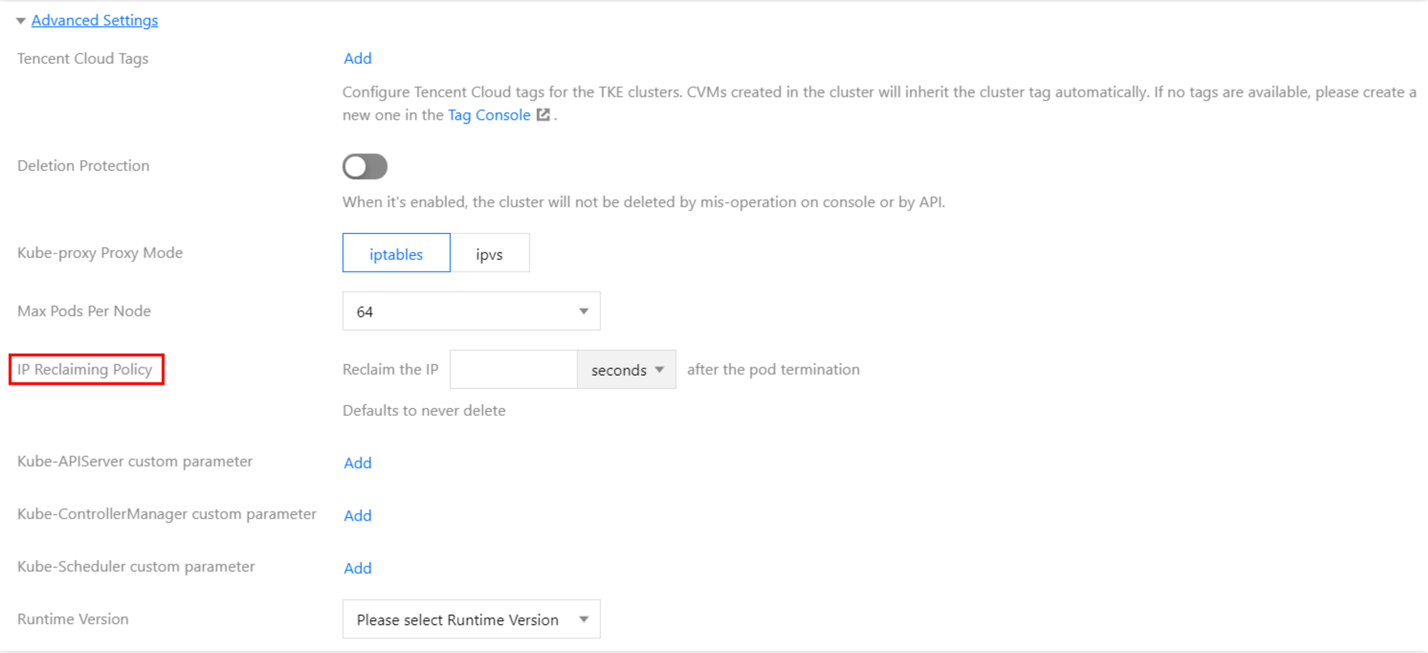


Container network add-on: Global Router | VPC-CNI (selected) How to select
VPC-CNI mode is a container network plug-in implemented based on ENI. The container network and CVM network are in the same VPC.

Network mode: Shared ENI with multiple IPs

Static Pod IP: ☑ Enable support
VPC-CNI mode does not support static Pod IP by default. You need to enable it manually.Learn more

Set **IP Reclaiming Policy** in **Advanced Settings**. You can set how many seconds after the Pod is terminated to reclaim the static IP address.



▾ Advanced Settings

Tencent Cloud Tags: Add

Configure Tencent Cloud tags for the TKE clusters. CVMs created in the cluster will inherit the cluster tag automatically. If no tags are available, please create a new one in the Tag Console .

Deletion Protection

When it's enabled, the cluster will not be deleted by mis-operation on console or by API.

Kube-proxy Proxy Mode: iptables | ipvs

Max Pods Per Node: 64

IP Reclaiming Policy: Reclaim the IP [ ] seconds after the pod termination

Defaults to never delete

Kube-APIServer custom parameter: Add

Kube-ControllerManager custom parameter: Add

Kube-Scheduler custom parameter: Add

Runtime Version: Please select Runtime Version

You can modify the **existing clusters** with the following method:

tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, select **Add-On Management** in the left sidebar.
4. On the **Add-On Management** page, click **Update configuration** in the **Operation** column of the **eniipamd** add-on.
5. On the **Update configuration** page, enter the expiration time in the static IP reclaiming policy, and click **Done**.

tke-eni-ipamd earlier than v3.5.0 or no eniipamd to manage

- Run the command `kubectl edit deploy tke-eni-ipamd -n kube-system` to modify the existing tke-eni-ipamd deployment.
- Run the following command to add the launch parameter to `spec.template.spec.containers[0].args` or modify the launch parameter.

```
- --claim-expired-duration=1h # You can enter a value that is not less than 5m
```

Manual reclaiming

For an IP address that urgently needs to be reclaimed, you need to find its Pod and namespace before running the following command to manually reclaim it:

Note：

You must make sure that the Pod of the IP address to be reclaimed has been terminated; otherwise, the Pod network will become unavailable.

```
kubectl delete vipc <podname> -n <namespace>
```

Cascade reclaiming

Currently, the static IP address is bound to a Pod, regardless of the specific workload (e.g., Deployment, Statefulset). After the Pod is terminated, it is uncertain when to reclaim the static IP address. TKE has implemented that the static IP address was deleted once the workload to which the Pod belongs was deleted.

You can enable cascade reclaiming by the following steps:

tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, select **Add-On Management** in the left sidebar.
4. On the **Add-On Management** page, click **Update configuration** in the **Operation** column of the **eniipamd** add-on.
5. On the **Update configuration** page, select **Cascade reclaiming**, and click **Done**.

tke-eni-ipamd earlier than v3.5.0 or no eniipamd to manage

1. **Run the command `kubectl edit deploy tke-eni-ipamd -n kube-system` to modify the existing tke-eni-ipamd deployment**.
2. Run the following command to add the launch parameter to `spec.template.spec.containers[0].args` .
   ```
   - --enable-ownerref
   ```

After the modification, ipamd will automatically restart and take effect. At that time, a new workload can implement the cascade deletion of the static IP, which is not supported for an existing workload.

FAQs

What should I do if the EIP cannot be bound and Pods cannot be scheduled to a node in shared ENI mode?

After a node is added to a cluster, IPAMD will try binding an EIP from the subnet in the same AZ as the node (the subnet configured for IPAMD) to the node. If IPAMD becomes abnormal or it is not configured with a subnet in the same AZ as the node, IPAMD cannot allocate a secondary ENI to the node. In addition, if the current VPC uses more secondary ENIs than the upper limit, no secondary ENIs can be allocated to the node.
Run the following command for troubleshooting:

```
kubectl get event
```

- If `event` displays `ENILimit` , the quota is not appropriate. You can fix the problem by increasing the quota of ENIs for the VPC.
- Check whether the container subnet in the AZ of the node has sufficient IP addresses, and if not, add some.

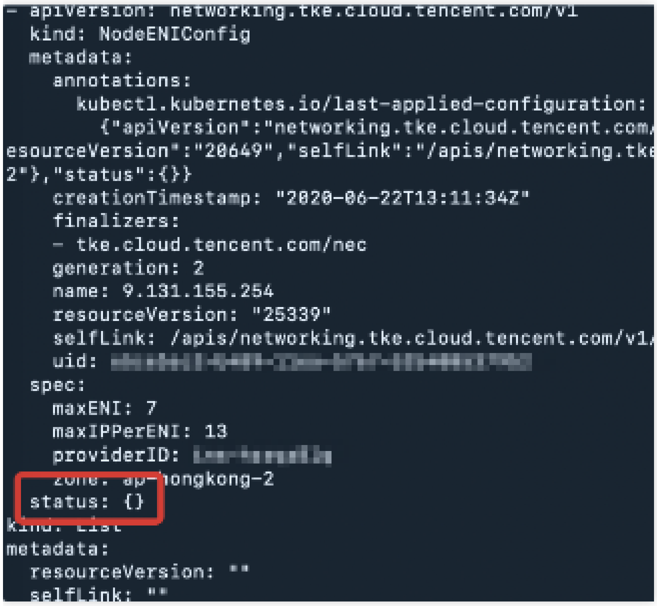
What should I do if the prompt says that the number of ENIs exceeds the upper limit and no ENIs can be allocated for the node?

Symptom

The ENIs configured for the node cannot be bound, and the VIP associated with `nec` failed to be attached. View `nec` , and you can see that its status is empty.
Run the following command to view `nec` :

```
kubectl get nec -o yaml
```

If the `nec` status of the node is empty, the returned result will be as shown below:

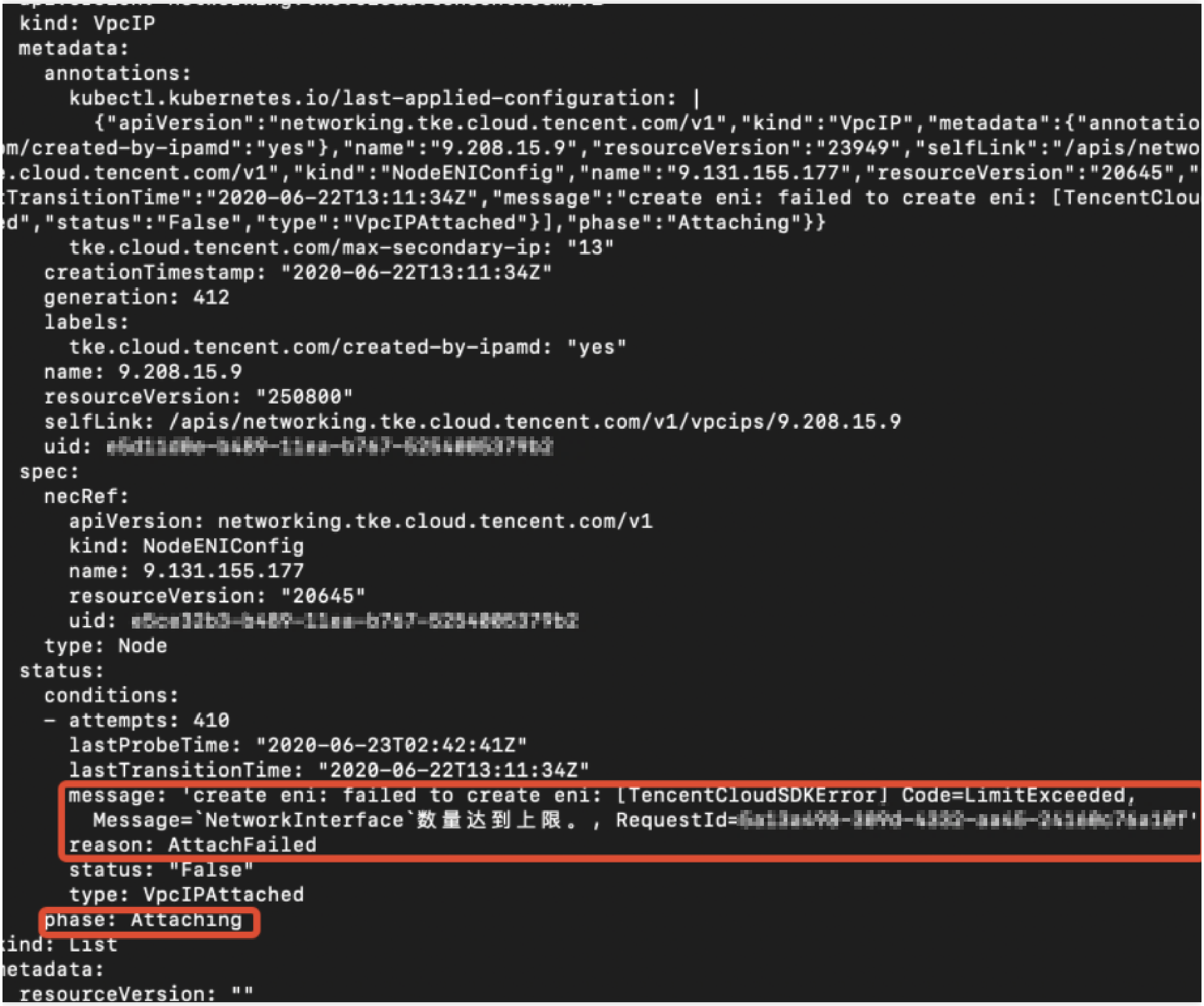


```
- apiVersion: networking.tke.cloud.tencent.com/v1
  kind: NodeENIConfig
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration:
        {"apiVersion":"networking.tke.cloud.tencent.com/
esourceVersion":"20649","selfLink":"/apis/networking.tke
2"},"status":{}}
    creationTimestamp: "2020-06-22T13:11:34Z"
    finalizers:
    - tke.cloud.tencent.com/nec
    generation: 2
    name: 9.131.155.254
    resourceVersion: "25339"
    selfLink: /apis/networking.tke.cloud.tencent.com/v1/
    uid: [redacted]
  spec:
    maxENI: 7
    maxIPPerENI: 13
    providerID: [redacted]
    zone: ap-hongkong-2
  status: {}
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

Run the following command to view the VIP associated with `nec` :

```
kubectl get vip -oyaml
```

If the command returns a success result, the VIP status is `Attaching` . The error message is as shown below:



```
 kind: VpcIP
 metadata:
   annotations:
     kubectl.kubernetes.io/last-applied-configuration: |
       {"apiVersion":"networking.tke.cloud.tencent.com/v1","kind":"VpcIP","metadata":{"annotatio
m/created-by-ipamd":"yes"},"name":"9.208.15.9","resourceVersion":"23949","selfLink":"/apis/netwo
.cloud.tencent.com/v1","kind":"NodeENIConfig","name":"9.131.155.177","resourceVersion":"20645","
TransitionTime":"2020-06-22T13:11:34Z","message":"create eni: failed to create eni: [TencentClou
d","status":"False","type":"VpcIPAttached"}],"phase":"Attaching"}}
     tke.cloud.tencent.com/max-secondary-ip: "13"
   creationTimestamp: "2020-06-22T13:11:34Z"
   generation: 412
   labels:
     tke.cloud.tencent.com/created-by-ipamd: "yes"
   name: 9.208.15.9
   resourceVersion: "250800"
   selfLink: /apis/networking.tke.cloud.tencent.com/v1/vpcips/9.208.15.9
   uid: [redacted]
 spec:
   necRef:
     apiVersion: networking.tke.cloud.tencent.com/v1
     kind: NodeENIConfig
     name: 9.131.155.177
     resourceVersion: "20645"
     uid: [redacted]
   type: Node
 status:
   conditions:
   - attempts: 410
     lastProbeTime: "2020-06-23T02:42:41Z"
     lastTransitionTime: "2020-06-22T13:11:34Z"
     message: 'create eni: failed to create eni: [TencentCloudSDKError] Code=LimitExceeded,
       Message=`NetworkInterface`数量达到上限。, RequestId=[redacted]'
     reason: AttachFailed
     status: "False"
     type: VpcIPAttached
   phase: Attaching
kind: List
etadata:
 resourceVersion: ""
```

Solution

Currently, up to 1,000 ENIs can be bound to a VPC. To increase the quota, submit a ticket for application. The quota will take effect by region.

Non-static IP Address Mode Instructions

Last updated：2023-02-23 18:43:33

Use Cases

The non-static IP address mode is suitable for the scenarios where the service does not rely on static IPs of the container. For example, stateless services and stateless offline applications that can deploy multiple copies.

Features and Limitations

- The node can maintain available ENI/IP pool, so that the Pod can be rebuilt quickly in a massive way.
- It supports the pre-binding policy, so that the Pod can be scaled out quickly.
- It supports auto scaling of ENIs/IPs to avoid waste of IPs and improve their utilization rate.
- `0` cannot be set for the pre-binding. That means fully on-demand allocation is not supported for now, and waste of IPs may occur if there are too many nodes.

IP Address Management Principle

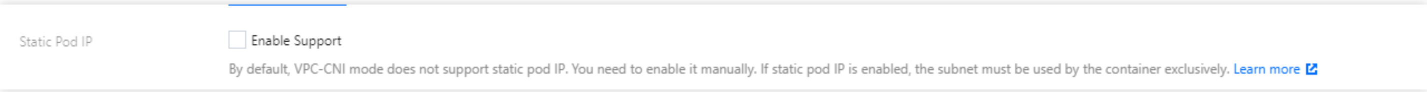
TKE maintains an auto-scaling exclusive ENI/IP pool on each node. The number of bound exclusive ENIs/IPs ranges from **the number of Pods + the minimum number of pre-bound ENIs/IPs** to **the number of Pods + the maximum number of pre-bound ENIs/IPs**.

- When **the number of bound ENIs/IPs is less than the number of Pods + the minimum number of pre-bound ENIs/IPs**, new exclusive ENIs/IPs will be bound to make **the number of bound ENIs/IPs = the number of Pods + the minimum number of pre-bound ENIs/IPs**.
- When **the number of bound ENIs/IPs is larger than the number of Pods + the maximum number of pre-bound ENIs/IPs**, an ENI/IP will be released about every 2 minutes until **the number of bound ENIs/IPs = the number of Pods + the maximum number of pre-bound ENIs/IPs**.
- When **the maximum number of bindable ENIs/IPs is less than the number of bound ENIs/IPs, the unnecessary idle exclusive ENIs/IPs will be released directly to make **the number of bound ENIs/IPs equal to the maximum number of bindable ENIs/IPs**.

How to Use

Enabling non-static IP address

Select non-static IP address of VPC-CNI mode when creating a cluster. That means you should not check **Enable Support** for **Static Pod IP**.



Static Pod IP

□ Enable Support

By default, VPC-CNI mode does not support static pod IP. You need to enable it manually. If static pod IP is enabled, the subnet must be used by the container exclusively. Learn more

Supporting quick release

The ENI/IP pool managed in non-static IP address mode applies the policy of slow release by default. Only one unnecessary idle ENI/IP is released every two minutes. If you want to utilize IPs more efficiently, you need to enable "quick release". In this mode, the ENI/IP pool is checked every two minutes to release unnecessary idle ENIs/IPs until the number of idle ENIs/IPs is equal to the maximum number of pre-bound ENIs/IPs. The methods for enabling quick release are as follows:

tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, select **Add-On Management** in the left sidebar.
4. On the **Add-On Management** page, click **Update configuration** in the **Operation** column of the **eniipamd** add-on.
5. On the **Update configuration** page, select **Quick release**, and click **Done**.

tke-eni-ipamd earlier than v3.5.0 or no eniipamd to manage

- Run the command `kubectl edit ds tke-eni-agent -n kube-system` to modify the existing tke-eni-agent daemonset.
- Add the following launch parameter to `spec.template.spec.containers[0].args` to enable "quick release". The tke-eni-agent will update and make the feature take effect on a rolling basis after the modification.

```
- --enable-quick-release
```

Specifying the number of pre-bound ENIs/IPs on a node

You can specify the number of pre-bound ENIs/IPs by modifying the annotation of CRD `NEC` corresponding to the node.

```
# Specifying the minimum number of pre-bound ENIs/IPs in shared ENI mode
"tke.cloud.tencent.com/route-eni-ip-min-warm-target"
# Specifying the maximum number of pre-bound ENIs/IPs in shared ENI mode
```

```
"tke.cloud.tencent.com/route-eni-ip-max-warm-target"
# Specifying the minimum number of pre-bound ENIs/IPs in exclusive ENI mode
"tke.cloud.tencent.com/direct-eni-min-warm-target"
# Specifying the maximum number of pre-bound ENIs/IPs in exclusive ENI mode
"tke.cloud.tencent.com/direct-eni-max-warm-target"
```

How to modify:

```
# Sample: modify the minimum number of pre-bound IPs on <nodeName> to 1
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/route-eni-ip-min-warm-targ
et"="1" --overwrite
# Sample: modify the maximum number of pre-bound IPs on <nodeName> to 3
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/route-eni-ip-max-warm-targ
et"="3" --overwrite
```

- Dynamic check for pre-binding is triggered immediately after the modification. If the number of pre-bound ENIs/IPs is insufficient, ENIs/IPs will be bound until the number meets your requirement. Otherwise, the ENIs/IPs will be unbound.
- The two annotations must exist at the same time when you perform the modification, and the condition of `0 &lt;= the minimum number of pre-bound ENIs/IPs &lt;= the maximum number of pre-bound ENIs/IPs` must be met. Otherwise, the modification will be failed.

Specifying the maximum number of bindable ENIs/IPs on a node

You can specify the maximum number of bindable ENIs/IPs on a node by modifying the annotation of CRD `nec` corresponding to the node. You can use the following annotations to specify the maximum number of bindable ENIs and the maximum number of bindable IPs on a single ENI.

```
# Specifying the maximum number of bindable ENIs in shared ENI mode
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/route-eni-max-attach"="1"
--overwrite
# Specifying the number of bindable IPs on a single ENI in shared ENI mode
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/max-ip-per-route-eni"="9"
--overwrite
# Specifying the maximum number of bindable exclusive ENIs in exclusive ENI mode
kubectl annotate nec <nodeName> "tke.cloud.tencent.com/direct-eni-max-attach"="5"
--overwrite
```

You must ensure that the value you entered for modification is not less than the number of ENIs/IPs currently in use on the node. Otherwise, the modification will be failed.

Dynamic check for pre-binding is triggered immediately after the modification. If `the number of bound ENIs/IPs > the maximum number of bindable ENIs/IPs`, the ENIs/IPs will be unbound until `the number of bound ENIs/IPs = the maximum number of bindable ENIs/IPs`.

Specifying the default number of pre-bound ENIs/IPs

tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, select **Add-On Management** in the left sidebar.
4. On the **Add-On Management** page, click **Update configuration** in the **Operation** column of the **eniipamd** add-on.
5. On the **Update configuration** page, enter the pre-bound default values for the shared and exclusive ENI modes, and click **Done**.

tke-eni-ipamd earlier than v3.5.0 or no eniipamd to manage

- Run the command `kubectl edit deploy tke-eni-ipamd -n kube-system` to modify the existing tke-eni-ipamd deployment.
- Add the following launch parameter to `spec.template.spec.containers[0].args` to modify the default number of pre-bound ENIs/IPs. The ipamd will restart and take effect automatically after the modification. The default number only affects the added nodes.

```
# The default number of minimum pre-bound ENIs/IPs in shared ENI mode. Default value: 5
- --ip-min-warm-target=3
# The default number of maximum pre-bound ENIs/IPs in shared ENI mode. Default value: 5
- --ip-max-warm-target=3
# The default number of minimum pre-bound ENIs/IPs in exclusive ENI mode. Default value: 1
- --eni-min-warm-target=3
# The default number of maximum pre-bound ENIs/IPs in exclusive ENI mode. Default value: 1
- --eni-max-warm-target=3
```

Interconnection Between VPC-CNI and Other Cloud Resources/IDC Resources

Last updated：2020-12-28 16:19:59

The VPC-CNI mode and container network are IP ranges that can be managed by the VPC. You can achieve the interconnection with other cloud resources and IDC resources through VPC feature configurations.

Tencent Cloud provides an extensive range of solutions to enable instances within a VPC, such as CVMs and databases, to connect to the Internet, connect to instances in other VPC instances, or interconnect with local IDCs.

Security Group of VPC-CNI Mode

Last updated：2023-05-06 19:15:14

You can bind specified security group to the ENI created in VPC-CNI mode through the following methods:

Prerequisites

The version of IPAMD component is v3.2.0 or later version. You can check the version through image tag.

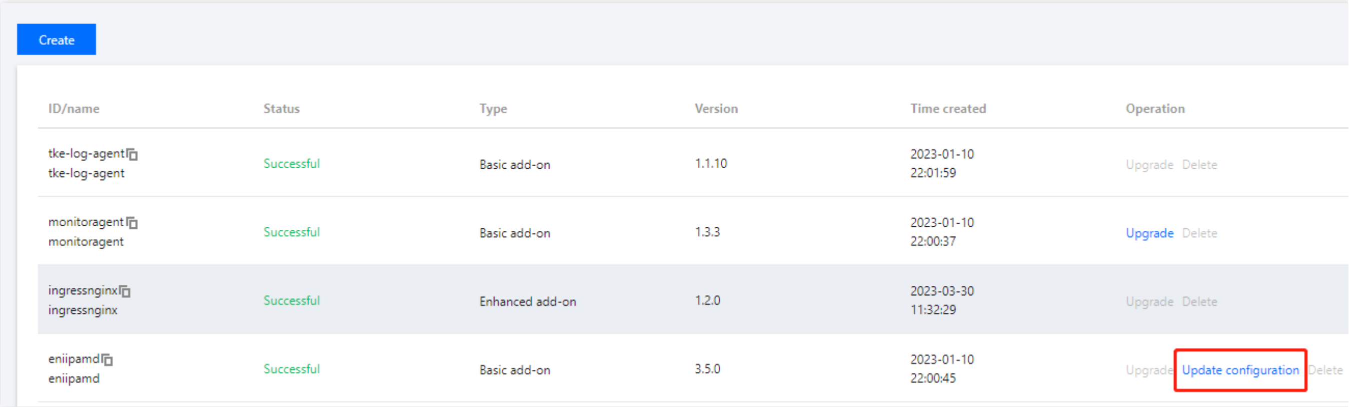
The IPAMD component has enabled the capability of security group (not enabled by default). The launch parameter is `--enable-security-groups` .

Currently, only multiple Pods with shared ENI mode is supported.

Enabling Security Group Feature for the IPAMD Component

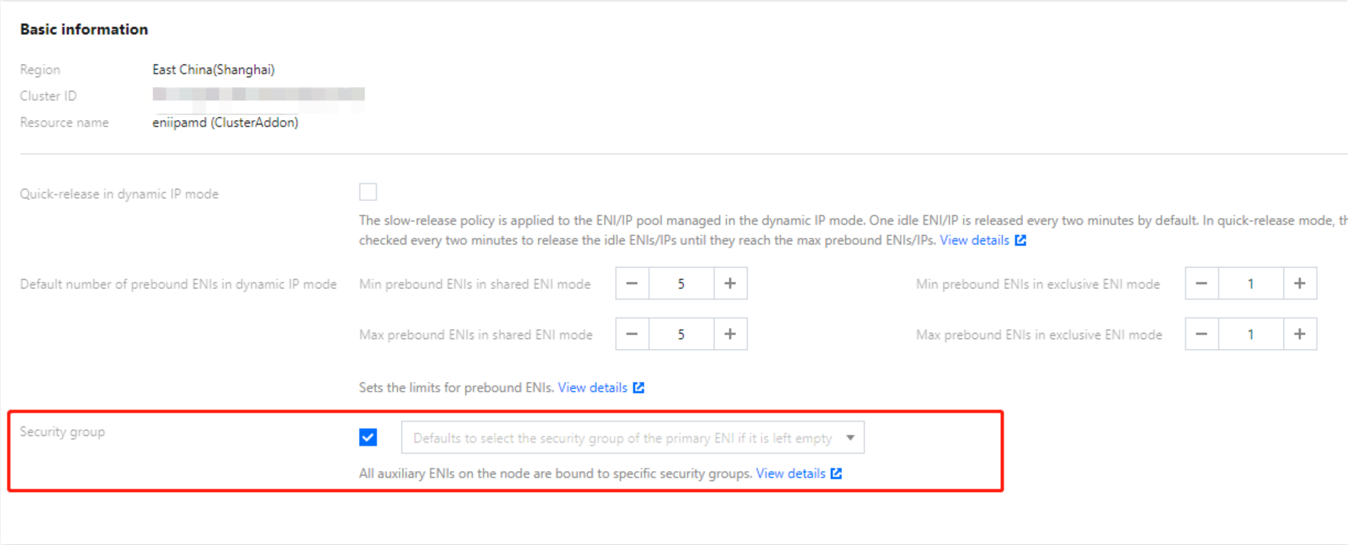
tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, select **Add-On Management** on the left of the page, click **Update configuration** in the **Operation** column of the **eniipamd** add-on.



ID/name	Status	Type	Version	Time created	Operation
tke-log-agent tke-log-agent	Successful	Basic add-on	1.1.10	2023-01-10 22:01:59	Upgrade Delete
monitoragent monitoragent	Successful	Basic add-on	1.3.3	2023-01-10 22:00:37	Upgrade Delete
ingressnginx ingressnginx	Successful	Enhanced add-on	1.2.0	2023-03-30 11:32:29	Upgrade Delete
eniipamd eniipamd	Successful	Basic add-on	3.5.0	2023-01-10 22:00:45	Upgrade Update configuration Delete

4. On the **Update configuration** page, select **Security group**. If you want to use the security group of the primary ENI, do not specify a security group. Otherwise, specify one.



Basic information

Region East China(Shanghai)

Cluster ID [redacted]

Resource name eniipamd (ClusterAddon)

Quick-release in dynamic IP mode □

The slow-release policy is applied to the ENI/IP pool managed in the dynamic IP mode. One idle ENI/IP is released every two minutes by default. In quick-release mode, th
checked every two minutes to release the idle ENIs/IPs until they reach the max prebound ENIs/IPs. View details

Default number of prebound ENIs in dynamic IP mode

Min prebound ENIs in shared ENI mode − 5 +

Min prebound ENIs in exclusive ENI mode − 1 +

Max prebound ENIs in shared ENI mode − 5 +

Max prebound ENIs in exclusive ENI mode − 1 +

Sets the limits for prebound ENIs. View details

Security group ☑ Defaults to select the security group of the primary ENI if it is left empty

All auxiliary ENIs on the node are bound to specific security groups. View details

5. Click **Done**.

tke-eni-ipamd earlier than v3.5.0 or no eniipamd to manage

Modify the existing tke-eni-ipamd deployment:

```
kubectl edit deploy tke-eni-ipamd -n kube-system
```

Run the following command to add the launch parameter to `spec.template.spec.containers[0].args` . After the modification, ipamd will restart and take effect automatically.

For a secondary ENI that is not associated with a security group on an existing node, a security group will be bound to it based on the following policy. If a security group has been bound, strong synchronization will be performed for the set security group unless the feature has been enabled before and a security group has been set on the node. The following security group will be bound to all ENIs on a new node.

```
- --enable-security-groups
# If you want to use the security groups of the primary ENI/instance by default, do
- --security-groups=sg-xxxxxxxx,sg-xxxxxxxx
```

Method of synchronizing ENI security group settings of existing nodes

If you want the security group policy to take effect on the existing nodes that have been set the security groups, you need to manually disable the security group, and then enable the security group again to achieve synchronization. You can operate as follows:

1. Add an annotation to clear and disable the security groups bound to the ENIs of the node. After the annotation is added, the existing ENIs of the node will unbind all security groups:

```
kubectl annotate node <nodeName> --overwrite tke.cloud.tencent.com/disable-node-eni
```

2. (Wait 2-5s after the first step.) After the value is reset to "no", the security groups configured based on the above policy can be rebound.

```
kubectl annotate node <nodeName> --overwrite tke.cloud.tencent.com/disable-node-eni
```

Feature Logic

If the launch parameter `--security-groups` is not set, or its value is empty, the security group of each node will use the security group bound to the node instance (security group bound to the primary ENI). If the feature is enabled, when the security group of a node instance (security group of the primary ENI) changes, the security group of the secondary ENI will not be synchronized automatically. Instead, you need to disable the security group on the node and enable it again for synchronization. For operation details, see Method of synchronizing ENI security group settings of existing nodes.

After the feature is enabled, if `--security-groups` is set, the security group of each node is set to this security group set.

After the feature is enabled, if `--security-groups` is modified, the settings of security groups on new nodes will be synchronized with global parameters, and the settings of security groups on existing nodes will remain unchanged. If you want to synchronize the settings of security groups on existing nodes, you need to disable the security group on the node and enable it again. For operation details, see Method of synchronizing ENI security group settings of existing nodes.

The priority for setting a security group is consistent with the sequence of setting a security group on a node. If the security group of the primary ENI is used, the priority is consistent with that of the primary ENI.

Run the following command to view the security group of the node. The `spec.securityGroups` contains the information of the security group of the node.

```
kubectl get nec <nodeName> -oyaml
```

Run the following command to modify the security group of the node. The modification will take effect immediately.

```
kubectl edit nec <nodeName>
```

After the feature is enabled, if an existing ENI is not bound with a security group, the security group of the node will be bound to it. Security group of an existing ENI will be strong synced with the security group of the node to ensure consistency with the security group of the node. A new ENI will be bound with security group of the node.

Instructions on Binding an EIP to a Pod

Last updated：2023-02-23 18:34:01

You can directly bind an EIP to a Pod that adopts the VPC-CNI mode as instructed below.

Prerequisites and Limitations

- The role policy that is used by IPAMD has been granted with EIP API permission.
- The EIP feature is not available in exclusive ENI with non-static IP address of VPC-CNI mode (it is available in v3.3.9 and later versions).
- The EIPs auto-created in the cluster cannot be reclaimed when the cluster is deleted.

Adding EIP API Access Permission for the IPAMD Component Role

1. Log in to the CAM console, select **Roles** in the left sidebar.
2. In **CAM console** > **Roles**, search for the role `IPAMDofTKE_QCSRole` , and click the role name to go to the role details page.
3. Click **Associate Policies**.
4. In the pop-up window, search for and select the preset policy `QcloudAccessForIPAMDRoleInQcloudAllocateEIP` , and click **OK**. This policy contains all permissions required by the IPAMD component to operate an EIP.

Auto-creating an EIP

See the following Yaml sample to associate with an EIP automatically:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
labels:
k8s-app: busybox
name: busybox
namespace: default
spec:
replicas: 1
selector:
matchLabels:
```

```
k8s-app: busybox
qcloud-app: busybox
serviceName: ""
template:
metadata:
annotations:
tke.cloud.tencent.com/networks: "tke-route-eni"
tke.cloud.tencent.com/eip-attributes: '{"Bandwidth":"100","ISP":"BGP"}'
tke.cloud.tencent.com/eip-claim-delete-policy: "Never"
creationTimestamp: null
labels:
k8s-app: busybox
qcloud-app: busybox
spec:
containers:
- args:
- "10000000000"
command:
- sleep
image: busybox
imagePullPolicy: Always
name: busybox
resources:
limits:
tke.cloud.tencent.com/eni-ip: "1"
tke.cloud.tencent.com/eip: "1"
requests:
tke.cloud.tencent.com/eni-ip: "1"
tke.cloud.tencent.com/eip: "1"
```

- **spec.template.annotations：tke.cloud.tencent.com/eip-attributes: '{"Bandwidth":"100","ISP":"BGP"}'** indicates that the Pod for the workload needs to automatically associate with an EIP. The bandwidth of the EIP is 100 Mbps and the ISP is BGP.
- **spec.template.annotations: tke.cloud.tencent.com/eip-claim-delete-policy: "Never"** indicates that the EIP of the Pod for the workload is a static IP address, and it cannot be changed after the Pod is terminated. If it is not a static IP address, do not add the annotation.
- **spec.template.spec.containers.0.resources**: to associate a Pod with an EIP, you need to add “requests” and “limits”, that is, `tke.cloud.tencent.com/eip` , so that the scheduler can ensure that the node to which the Pod scheduled still have EIPs available.

Key configurations

- The EIPs that each node can bind to are subject to the relevant quota restrictions and the bound number of CVMs. The maximum number of EIPs that each node can bind to is **the bound number of CVMs - 1**.

- **tke.cloud.tencent.com/eip-attributes: '{"Bandwidth":"100","ISP":"BGP"}'**: only "bandwidth" and "ISP" can be configured for now. "ISP" can be set to `BGP` , `CMCC` , `CTCC` or `CUCC` , which corresponds to ordinary BGP IP and static single-line IP (China Mobile, China Telecom and China Unicom) respectively. If the two parameters are left empty, the default values of 100 Mbps and BGP will be used.
- Fees will not be charged on IPs after an auto-created EIP is bound. The default billing method for public network access is `postpaid by traffic on an hourly basis` .

Specifying an EIP

See the following Yaml sample to associate with a specified EIP automatically:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
labels:
k8s-app: busybox
name: busybox
namespace: default
spec:
replicas: 1
selector:
matchLabels:
k8s-app: busybox
qcloud-app: busybox
serviceName: ""
template:
metadata:
annotations:
tke.cloud.tencent.com/networks: "tke-route-eni"
tke.cloud.tencent.com/eip-id-list: "eip-xxx1,eip-xxx2"
creationTimestamp: null
labels:
k8s-app: busybox
qcloud-app: busybox
spec:
containers:
- args:
- "10000000000"
command:
- sleep
image: busybox
imagePullPolicy: Always
name: busybox
```

```
resources:
limits:
tke.cloud.tencent.com/eni-ip: "1"
tke.cloud.tencent.com/eip: "1"
requests:
tke.cloud.tencent.com/eni-ip: "1"
tke.cloud.tencent.com/eip: "1"
```

- **tke.cloud.tencent.com/eip-id-list: "eip-xxx1,eip-xxx2"** indicates that the Pod of the workload needs to be automatically associated with a specified EIP and that the first replica uses the EIP whose `eipID` is `eip-xxx1` and the second uses the EIP whose `eipID` is `eip-xxx2` . According to the current policy, Pods are associated with EIPs in the annotation based on the numbers at the end of their names. If there are no numbers (Deployment type, for example), EIPs are randomly associated with. When conflicts occur, only one Pod can be associated successfully. For Pods without numbers, we recommend you specify only one EIP.
- **spec.template.spec.containers.0.resources**: to associate a Pod with an EIP, you need to add “requests” and “limits”, that is, `tke.cloud.tencent.com/eip` , so that the scheduler can ensure that the node to which the Pod scheduled still have EIPs available.

Making Sure That Active Outbound Traffic Passes EIPs

By default, the current cluster is deployed with the `ip-masq-agent` component, which performs SNAT for node addresses of the active outbound traffic of the Pods in the cluster. In addition, if the VPC is configured with the NAT gateway, the configuration will affect the active outbound traffic of the Pods. Therefore, to let the active outbound traffic of a Pod pass its associated EIP, you need to modify the relevant configuration and routing policy.

Removing SNAT from a cluster

To prevent SNAT from being performed for the active outbound traffic of the Pod associated with the EIP, you need to modify the SNAT rules in the cluster:

```
kubectl -n kube-system edit cm ip-masq-agent-config
```

In the `data.config` field, add a new field whose key is `NonMasqueradeSrcCIDRs` and whose value is the **private IP** range list of the Pod associated with the EIP. For example, if the IP address is `172.16.0.2` , you need to enter `172.16.0.2/32` . Below is a sample:

```
apiVersion: v1
data:
config: '{"NonMasqueradeCIDRs":["172.16.0.0/16","10.67.0.0/16"],"NonMasqueradeSrcCIDRs":["172.16.0.2/32"],"MasqLinkLocal":true,"ResyncInterval":"1m0s","MasqLinkLocalIPv6":false}'
```

```
kind: ConfigMap
metadata:
name: ip-masq-agent-config
namespace: kube-system
```

The saved configuration takes effect immediately after exit and will be hot updated within one minute.

This field prevents the active outbound traffic of Pods within the IP range from SNAT. If a larger IP range is entered, no SNAT will be performed for Pods within the range. Proceed with caution.

Adjusting the priority levels of NAT gateways and EIPs

If the NAT gateway is configured for the VPC of the cluster, make sure that the configurations are correct as instructed in Adjusting the Priorities of NAT Gateways and EIPs; otherwise, the active outbound traffic of the Pod may prefer NAT gateways over EIPs.

Retaining and Reclaiming of an EIP

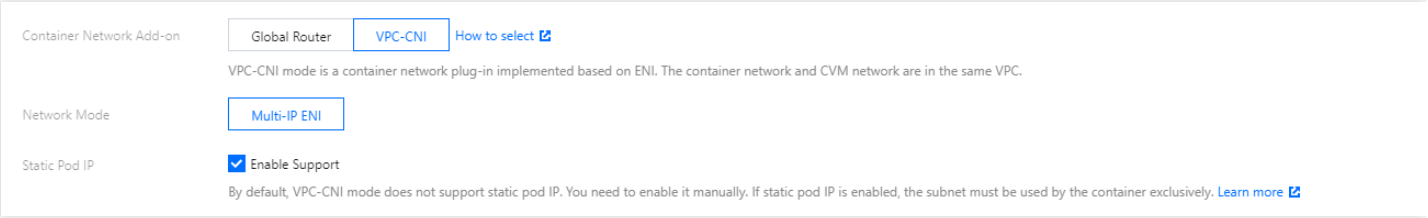
After "auto-associate with an EIP" is enabled for the Pod, the network component will create a CRD object `EIPClaim` with the same name of the Pod in the same namespace. This object describes the Pod's requirements for the EIP.

For a Pod to which a non-static EIP is bound, `EIPClaim` will be terminated and the EIP associated with the Pod will also be terminated and reclaimed after the Pod is terminated. For a Pod to which a static EIP is bound, `EIPClaim` and the EIP will be retained after the Pod is terminated. After the Pod **with the same name** is enabled, it will use the EIP associated with the `EIPClaim` of the same name, so as to retain the EIP.

Below are three methods for reclaiming an EIP, including reclaiming after expiration, manual reclaiming and cascade reclaiming.

Reclaiming after expiration

On Creating a Cluster page, select **VPC-CNI** for **Container Network Add-on** and check **Enable Support** for **Static Pod IP**, as shown in the figure below:


Container Network Add-on: Global Router | VPC-CNI | How to select

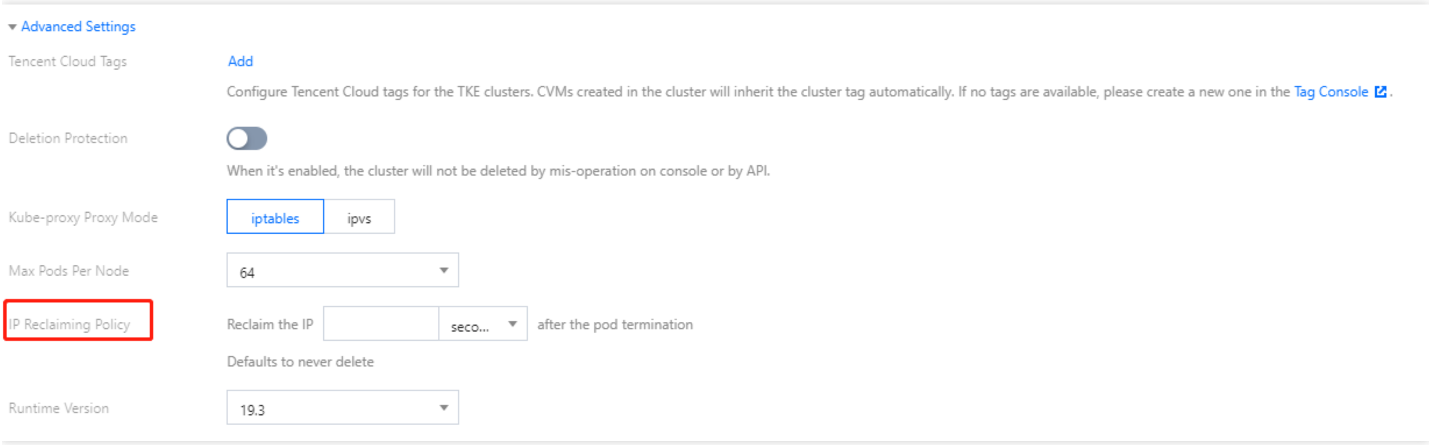
VPC-CNI mode is a container network plug-in implemented based on ENI. The container network and CVM network are in the same VPC.

Network Mode: Multi-IP ENI

Static Pod IP: ☑ Enable Support

By default, VPC-CNI mode does not support static pod IP. You need to enable it manually. If static pod IP is enabled, the subnet must be used by the container exclusively. Learn more

Set **IP Reclaiming Policy** in **Advanced Settings**. You can set how many seconds after the Pod is terminated to reclaim the static IP address.



▾ Advanced Settings

Tencent Cloud Tags: Add

Configure Tencent Cloud tags for the TKE clusters. CVMs created in the cluster will inherit the cluster tag automatically. If no tags are available, please create a new one in the Tag Console.

Deletion Protection: ○

When it's enabled, the cluster will not be deleted by mis-operation on console or by API.

Kube-proxy Proxy Mode: iptables | ipvs

Max Pods Per Node: 64

IP Reclaiming Policy: Reclaim the IP [ ] seco... after the pod termination

Defaults to never delete

Runtime Version: 19.3

You can modify the **existing clusters** with the following method:

tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, select **Add-On Management** in the left sidebar.
4. On the **Add-On Management** page, click **Update configuration** in the **Operation** column of the **eniipamd** add-on.
5. On the **Update configuration** page, enter the expiration time in the static IP reclaiming policy, and click **Done**.

tke-eni-ipamd earlier than v3.5.0 or no eniipamd to manage

- Run the command `kubectl edit deploy tke-eni-ipamd -n kube-system` to modify the existing tke-eni-ipamd deployment.
- Run the following command to add the launch parameter to `spec.template.spec.containers[0].args` or modify the launch parameter.

```
- --claim-expired-duration=1h # You can enter a value that is not less than 5m
```

Manual reclaiming

For an EIP that needs to be reclaimed urgently, you need to find the namespace and name of the corresponding Pod, and run the following command to reclaim it manually.

> Note：
> You must ensure the Pod corresponding to the reclaimed EIP have been terminated. Otherwise, the EIP will be associated with and bound to the Pod again.

```
kubectl delete eipc <podname> -n <namespace>
```

Cascade reclaiming

Currently, the static EIP is strongly bound to the Pod, regardless of the specific workload (e.g., deployment, statefulset). After the Pod is terminated, it is uncertain when to reclaim the static EIP. TKE has implemented that the static EIP is deleted once the workload to which the Pod belongs is deleted. **The version of the IPAMD component needs to be v3.3.9 or later version (you can check the version through image tag)**.

You can enable cascade reclaiming by the following steps:

tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
3. On the cluster details page, select **Add-On Management** in the left sidebar.
4. On the **Add-On Management** page, click **Update configuration** in the **Operation** column of the **eniipamd** add-on.
5. On the **Update configuration** page, select **Cascade reclaiming**, and click **Done**.

tke-eni-ipamd earlier than v3.5.0 or no eniipamd to manage

1. **Run the command `kubectl edit deploy tke-eni-ipamd -n kube-system` to modify the existing tke-eni-ipamd deployment**.
2. Run the following command to add the launch parameter to `spec.template.spec.containers[0].args`.

   ```
   - --enable-ownerref
   ```

After the modification, ipamd will automatically restart and take effect. At that time, a new workload can implement the cascade deletion of the static EIP, which is not supported for an existing workload.

VPC-CNI Component Description

Last updated：2024-02-01 10:04:17

VPC-CNI component contains three kubernetes cluster components: `tke-eni-agent` , `tke-eni-ipamd` and `tke-eni-ip-scheduler` .

tke-eni-agent

It is deployed on each node of the cluster in the form of `daemonset` . The responsibilities are described below.

Copy `tke-route-eni` , `tke-eni-ipamc` and other CNI plugins to the directory of CNI executive file of the node (it is set to `/opt/cni/bin` by default).

Generate CNI configuration file in CNI configuration directory (it is set to `/etc/cni/net.d/` by default).

Configure policy-based routing and an ENI for the node.

It acts as the GRPC Server to be responsible for allocating/releasing Pod IPs.

Conduct IP garbage collection periodically, and reclaim IPs for which the Pods does not on the node.

Set expansion resources of ENIs and IPs through [Device Plugins] of kubernetes(https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/).

tke-eni-ipamd

It is deployed on certain nodes or masters of the cluster in the form of `deployment` . The responsibilities are described below.

Create and manage CRD resources such as nec, vipc, vip and veni.

In non-static IP address mode, create/bind/unbind/delete an ENI and allocate/release an ENI IP based on node requirements and status.

In static IP address mode, create/bind/unbind/delete an ENI and allocate/release an ENI IP based on Pod requirements and status.

Manage the security groups of ENIs of the node.

Create/bind/unbind/delete an EIP based on Pod requirements.

tke-eni-ip-scheduler

It is deployed on certain nodes or masters of the cluster in the form of `deployment` only in static IP address mode to act as an extension plugin for scheduling. The responsibilities are described below.

If there are multiple subnets, it schedule the Pods with static IP addresses to the nodes of the specified subnet.
In static IP address mode, it judges whether the IPs in the subnets corresponding to the node to which the Pod is scheduled are sufficient.

Component Permission Description

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**.

tke-eni-agent Permission

Permission Description

The permission of this component is the minimal dependency required for the current feature to operate.
To modify network-related kernel parameters, such as net.ipv4.ip_forward, net.ipv4.rp_filter, etc., thus the activation of a privileged-level container is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
In the process of IP allocation, obtaining information related to pods and nodes is required.	pods, namespaces, and nodes	get/list/watch
Obtaining the network configuration information	configmaps	get/list/watch
Managing the relevant network extended resources of the nodes, such as tke.cloud.tencent.com/eni-ip, etc.	nodes/status	get/list/watch/patch
Obtaining IP, Network Interface Card, and other network configuration details through a self-defined object, and collaborating with the eni-ipamd component	networking.tke.cloud.tencent.com groups	get/list/watch/delete/update
Exposing the working status of components through events and	events	get/list/watch/create/update/patch

information related to changes in the node network		

Permission Definition

```
kind: ClusterRole
metadata:
  name: tke-eni-agent
rules:
- apiGroups: [""]
```

```
  resources:
  - pods
  - namespaces
  - nodes
  - configmaps
  verbs: ["list", "watch", "get"]
- apiGroups: [""]
  resources:
  - nodes/status
  verbs: ["list", "watch", "get", "patch"]
- apiGroups: ["networking.tke.cloud.tencent.com"]
  resources:
    - underlayips
    - nodeeniconfigs
    - vpcipclaims
    - vpcips
    - vpcenis
  verbs: ["get", "list", "watch", "delete", "update"]
- apiGroups: [""]
  resources:
    - events
  verbs: ["list", "watch", "get", "update", "patch", "create"]
```

tke-eni-ipamd 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
In the process of IP allocation, obtaining information related to pods and nodes is required.	pods, namespaces, nodes, and nodes/status	get/list/watch
In the process of allocating IP to the super node's Pod, it is required to update the allocation information to the Pod's annotation.	pods	update/patch
Under the global routing work pattern, it is required to write the podCIDR assigned to the	nodes, and nodes/status	update/patch

node on the node's object. Simultaneously, when working in conjunction with the node's auto-scaling, it is required to update the node's conditions and taints.		
The multi-replica operation feature is based on LeaderElection, which requires read and write permissions for associated configmaps or endpoints, with operational information exposed via events.	configmaps, endpoints, and events	get/list/watch/create/update/patch
When a Pod with a fixed IP is terminated, it is required to obtain its associated workload information to determine whether the fixed IP needs to be released.	statefulsets and deployments	get/list/watch
Using custom objects to manage relevant network resources (Elastic Network Interface, IP, Security Group, etc.).	customresourcedefinitions	create/update/get
	networking.tke.cloud.tencent.com apiGroups	get/list/watch/create/update/patch/delete
It is required to obtain the native node-related information.	node.tke.cloud.tencent.com apiGroups	get/list/watch
Registration of node-related capabilities requires collaboration with the Cilium component.	cilium.io apiGroups	get/list/watch/create/update/patch/delete

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
# kubernetes versions before 1.8.0 should use rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: tke-eni-ipamd
rules:
- apiGroups: [""]
  resources:
  - pods
  - namespaces
  - nodes
```

```
    - nodes/status
    verbs: ["list", "watch", "get", "patch", "update"]
  - apiGroups: [""]
    resources:
    - configmaps
    - endpoints
    - events
    verbs: ["get", "list", "watch", "update", "create", "patch"]
  - apiGroups: ["apps", "extensions"]
    resources:
    - statefulsets
    - deployments
    verbs: ["list", "watch", "get"]
  - apiGroups: ["apiextensions.k8s.io"]
    resources:
    - customresourcedefinitions
    verbs: ["create", "update", "get"]
  - apiGroups: ["networking.tke.cloud.tencent.com"]
    resources:
    - staticipconfigs
    - underlayips
    - nodeeniconfigs
    - vpcipclaims
    - vpcips
    - eipclaims
    - vpcenis
    verbs: ["create", "update", "delete", "get", "list", "watch", "patch"]
  - apiGroups: ["node.tke.cloud.tencent.com"]
    resources:
    - machines
    verbs: ["get", "list", "watch"]
  - apiGroups: [ "cilium.io" ]
    resources:
    - ciliumnodes
    - ciliumnodes/status
    - ciliumnodes/finalizers
    verbs: [ "create", "update", "delete", "get", "list", "watch", "patch" ]
```

tke-eni-ip-scheduler Permission

Permission Description

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

The related directory /var/lib/kubelet on the host machine needs to be mounted to the container to accomplish volume mount/unmount, hence the activation of the privileged-level container is required.

Permission Scenarios

Feature	Involved Object	Involved Operation Permission
An expansion of bindVerb is required, to address the issue of IP allocation conflicts when binding the Pod concurrency.	pods/binding	get/list/watch/create/update/patch
The multi-replica operation feature is based on LeaderElection, which requires read and write permissions for associated configmaps or endpoints, with operational information exposed via events.	configmaps,endpoints,events	get/list/watch/create/update/patch
During scheduling expansion, it is required to obtain relevant information about pods and nodes.	pods,namespaces,nodes,nodes/status	get/list/watch
During scheduling expansion, it is required to interact with the custom object of the component, thus ensuring the complete allocation of IPs and resolving the conflicts of IP allocation.	networking.tke.cloud.tencent.com groups	get/list/watch/update

Permission Definition

```
apiVersion: rbac.authorization.k8s.io/v1
# kubernetes versions before 1.8.0 should use rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: tke-eni-ip-scheduler
rules:
  - apiGroups: [""]
    resources:
      - pods/binding
    verbs: ["get", "list", "watch", "update", "create", "patch"]
  - apiGroups: [""]
```

```
      resources:
        - ["configmaps", "endpoints", "events"]
      verbs: ["get", "list", "watch", "update", "create", "patch"]
    - apiGroups: [""]
      resources:
        - ["pods", "namespaces", "nodes", "nodes/status"]
      verbs: ["list", "watch", "get"]
    - apiGroups: ["networking.tke.cloud.tencent.com"]
      resources:
        - ["nodeeniconfigs", "vpcipclaims", "vpcips"]
      verbs: ["get", "list", "watch", "update"]
```

Limits on the Number of Pods in VPC-CNI Mode

Last updated：2022-11-03 15:50:16

This document describes the default limits on the number of Pods in different VPC-CNI network modes.

Limits on the Number of Pods with Shared ENI

The number of Pods with shared ENI is limited by the number of ENIs that can be bound to a node and the number of IPs that can be bound on a single ENI. By default, **the maximum number of Pod IPs on a single node with multiple ENIs** = **the maximum number of secondary ENIs that can be bound** * **the number of secondary IPs that can be bound on a single ENI**, and **the maximum number of Pod IPs on a single node with a single ENI = the number of secondary IPs that can be bound on a single ENI**. See the table below for details.

CPU Cores	1	2-6	8-10	>=12
The maximum number of secondary ENIs that can be bound	1	3	5	7
The number of secondary IPs that can be bound on a single ENI	5	9	19	29
The maximum number of Pod IP addresses on a single node (with multiple ENIs)	5	27	95	203
The maximum number of Pod IP addresses on a single node (with a single ENI)	5	9	19	29

Note：
Multi-ENI component versions are supported (v3.3 or later in non-static IP address mode, and v3.4 or later in static IP address mode).

The number of ENIs that can be bound to each model and the number of IPs that can be bound on a single ENI is slightly different. For details, see Use Limits.

Limits on the Number of Pods with Exclusive ENIs

The number of Pods with exclusive ENIs is only limited by the number of ENIs that can be bound to the node. It only supports some models such as S5, SA2, IT5 and SA3. See the table below for details.

CPU Cores **Model**	1	2	4	>=8	>=128
S5	4	9	19	39	23
SA2	4	9	19	39	23
IT5	4	9	19	39	23
SA3	4	9	15	15	15

Cilium-Overlay Mode

Cilium-Overlay Mode

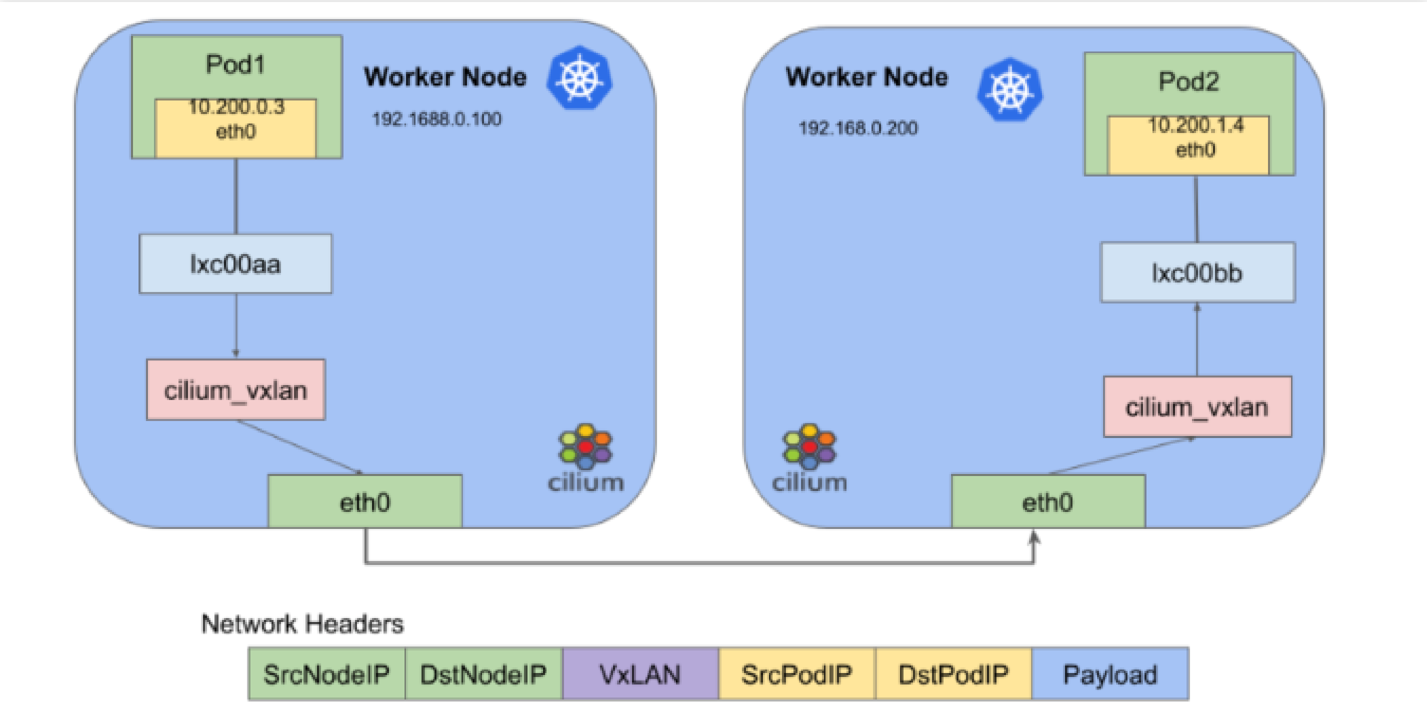
Last updated：2022-08-10 15:53:23

How It Works

The Cilium-Overlay network mode is a container network plugin provided by TKE based on Cilium VXLan. In this mode, you can add external nodes to TKE clusters in distributed cloud scenarios. The features of this mode are as follows:

- Cloud nodes and external nodes share the specified container IP range.
- Container IP ranges are dynamically assigned without occupying other IP ranges in the VPC instance.
- The Cilium VXLan tunnel encapsulation protocol is used to build the Overlay network.

Cross-node Pod access is supported after the VPC network in the cloud and the IDC network of the external node are interconnected through Cloud Connect Network. See the figure below to learn how it works.



Note：

> Due to performance loss in the Cilium-Overlay mode, this mode only supports the scenarios where external nodes are configured in distributed cloud, and does not support the scenarios where only cloud nodes are configured. For more information, see External Node Overview.

Usage Limits

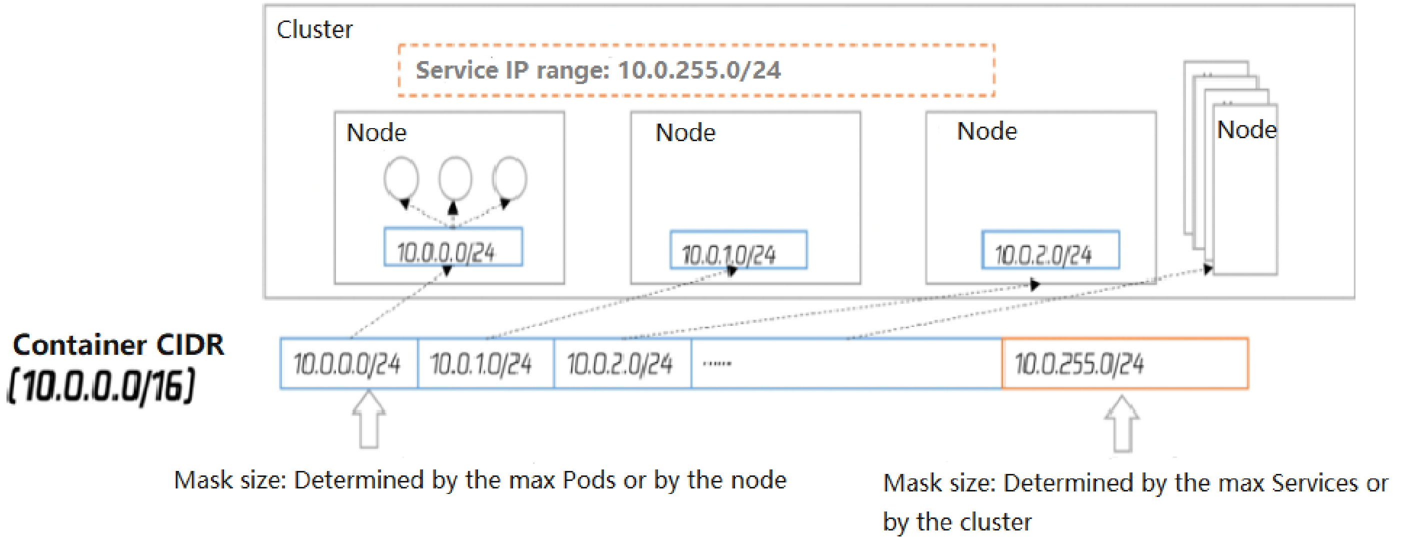
- There is a performance loss of less than 10% when you use the Cilium VXLan tunnel encapsulation protocol.
- The Pod IP cannot be accessed directly outside the cluster.
- You must obtain two IPs from the specified subnet to create a private CLB, so that external nodes in IDC can access APIServer and the public cloud services.
- The IP ranges of the cluster network and container network cannot overlap.
- Static Pod IPs are not supported.

Container IP Assignment Mechanism

For container network terms and quantity calculation, see Container Network Overview.

Pod IP assignment

The following diagram illustrates how it works:


Cluster
Service IP range: 10.0.255.0/24
Node
10.0.0.0/24
Node
10.0.1.0/24
Node
10.0.2.0/24
Node
Container CIDR
(10.0.0.0/16)
10.0.0.0/24
10.0.1.0/24
10.0.2.0/24
......
10.0.255.0/24
Mask size: Determined by the max Pods or by the node
Mask size: Determined by the max Services or by the cluster

- Nodes in a cluster include cloud nodes and external nodes.

- Each node of the cluster will use the specified IP range in the container CIDR block for the node to assign IPs to Pods.
- The Service IP range of the cluster will select the last segment of the specified IP range in the container CIDR block for Service IPs assignment.
- After the node is released, the corresponding container IP range will be returned to the IP range pool as well.
- The added nodes automatically select the available IP ranges in the container CIDR block cyclically and sequentially.

OPS Center

Audit Management

Cluster Audit

Last updated：2023-05-24 10:56:22

Note：

From now to June 30, 2022, users are **exempt from CLS service fees** incurred by audit log/event data generated by TKE for auto-created log topics. For details, see Note on Free Log Storage for TKE Audit and Event Center.

Introduction

Cluster audit is a feature based on Kubernetes Audit that can store and search the records of kube-apiserver JSON logs to generate configurable policies. This feature records the access events of kube-apiserver and records the activities of each user, admin, or system component that has an impact on the cluster in sequence.

Advantages

The cluster audit feature provides a different cluster monitoring dimension from metrics. After cluster audit is enabled, Kubernetes can record the log of every audit operation on the cluster. An audit log is a structured record in JSON format and consists of three parts: metadata, requestObject, and responseObject. The metadata (containing request context information, such as who initiated the request, where it was initiated, and the accessed URI) component is required, whereas requestObject and responseObject are optional, depending on the audit level. You can obtain the following information from logs:

- Activities that occur in the cluster
- Activity occurrence times and objects
- Activity triggering times, triggering positions, and observation points
- Activity results and subsequent processing

Reading the audit log

```
{
"kind":"Event",
"apiVersion":"audit.k8s.io/v1",
"level":"RequestResponse",
"auditID":0a4376d5-307a-4e16-a049-24e017******,
"stage":"ResponseComplete",
// What happened?
"requestURI":"/apis/apps/v1/namespaces/default/deployments",
"verb":"create",
// Who initiated the request?
"user":{
"username":"admin",
"uid":"admin",
"groups":[
"system:masters",
"system:authenticated"
]
},
// Where was it initiated?
"sourceIPs":[
"10.0.6.68"
],
"userAgent":"kubectl/v1.16.3 (linux/amd64) kubernetes/ald64d8",
// What happened?
"objectRef":{
"resource":"deployments",
"namespace":"default",
"name":"nginx-deployment",
"apiGroup":"apps",
"apiVersion":"v1"
},
// What is the result?
"responseStatus":{
"metadata":{
},
"code":201
},
// Request and response details
"requestObject":Object{...},
"responseObject":Object{...},
// When did it start/end?
"requestReceivedTimestamp":"2020-04-10T10:47:34.315746Z",
"stageTimestamp":"2020-04-10T10:47:34.328942Z",
// Reason for accepting/rejecting the request
"annotations":{
"authorization.k8s.io/decision":"allow",
"authorization.k8s.io/reason":""
}
}
```

TKE Cluster Audit Policies

Audit levels (levels)

Unlike common logs, the level of a Kubernetes audit log is more like a verbose configuration, which is used to indicate the degree of detail of the recorded information. There are four audit levels, as described in the following table:

Parameter	Description
None	Nothing is recorded
Metadata	The metadata of the request (for example, the user, time, resources, and operation) is recorded, excluding the message bodies of the request and response
Request	The metadata and request message body are recorded, excluding the response message body
RequestResponse	All information is recorded, including the metadata and the message bodies of the request and response

Audit stages (stages)

Logs can be recorded at different stages, as described in the following table:

Parameter	Description
RequestReceived	The log is recorded when the request is received
ResponseStarted	The log is recorded after the message header of the response is sent. This parameter is applicable only to persistent connection requests, such as watch.
ResponseComplete	The log is recorded after the response is fully sent
Panic	The request is not completed due to an internal server error

TKE audit policies

By default, TKE records audit logs when receiving requests. For most operations, audit logs of the RequestResponse level are recorded, except for the following cases:

- For get, list, and watch, logs of the Request level are recorded.
- For requests of secrets resources, configmaps resources, or tokenreviews resources, logs of the Metadata level are recorded.

Logs will not be recorded for the following requests:

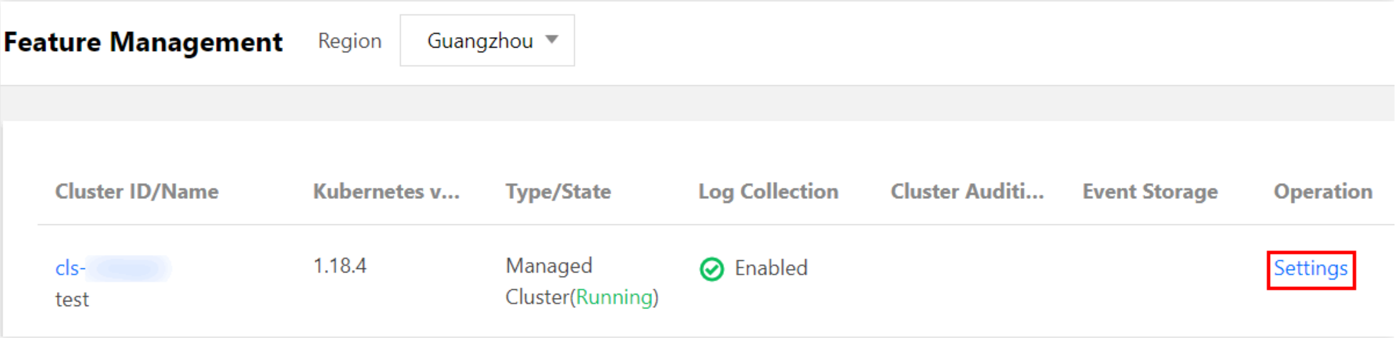
- Requests sent by `system:kube-proxy` for monitoring endpoints resources, services resources, or services/status resources.
- get requests sent by `system:unsecured` for configmaps resources in the kube-system namespace.
- get requests sent by kubelet for nodes resources or nodes/status resources.
- get requests sent by any identity in `system:nodes` for nodes resources or nodes/status resources.
- get and update requests sent by `system:kube-controller-manager` , `system:kube-scheduler` , or `system:serviceaccount:endpoint-controller` for endpoint resources in the kube-system namespace.
- get requests sent by `system:apiserver` for namespaces resources, namespaces/status resources, or namespaces/finalize resources.
- Requests sent to URLs that match `/healthz*` , `/version` , or `/swagger*` .

Directions

Enabling cluster audit

> Note：
>
> - To enable the cluster audit feature, you need to restart kube-apiserver. We recommend that you do not frequently enable and disable the feature.
> - A self-deployed cluster consumes about 1 GB of local storage on the master node. Therefore, ensure that the storage of the master node is sufficient.

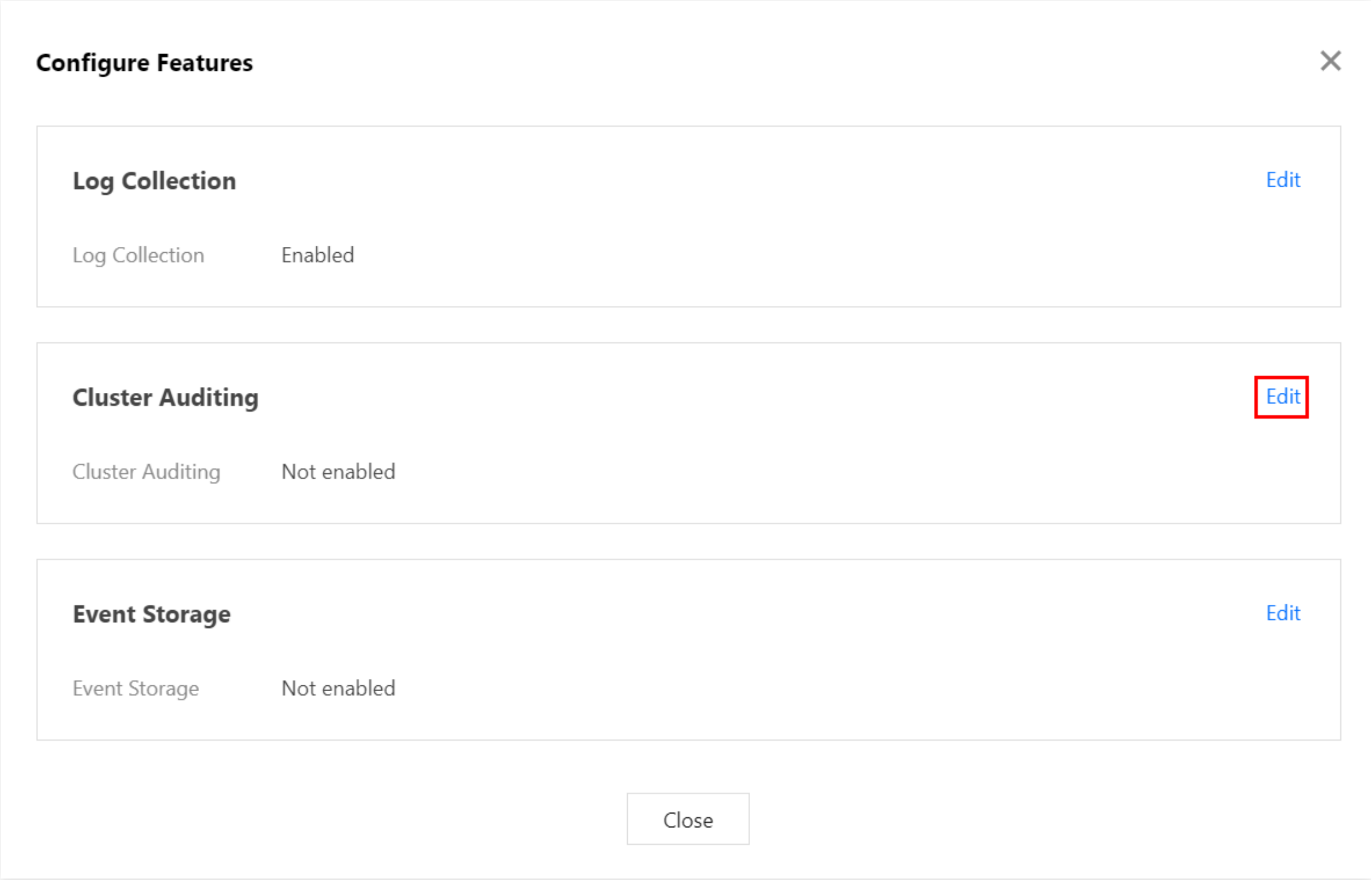
1. Log in to the TKE console.
2. Choose **Cluster OPS** > **Feature Management** in the left sidebar to go to the **Feature Management** page.
3. At the top of the "Feature Management" page, select the region. Click **Set** for the cluster that you want to enable cluster audit, as shown in the figure below:



Feature Management Region Guangzhou ▾

Cluster ID/Name	Kubernetes v...	Type/State	Log Collection	Cluster Auditi...	Event Storage	Operation
cls-[redacted] test	1.18.4	Managed Cluster(Running)	✓ Enabled			Settings

4. In the "Configure a Feature" window that appears, click **Edit** for the "Cluster Audit" feature.



Configure Features ✕

Log Collection Edit

Log Collection Enabled

Cluster Auditing Edit

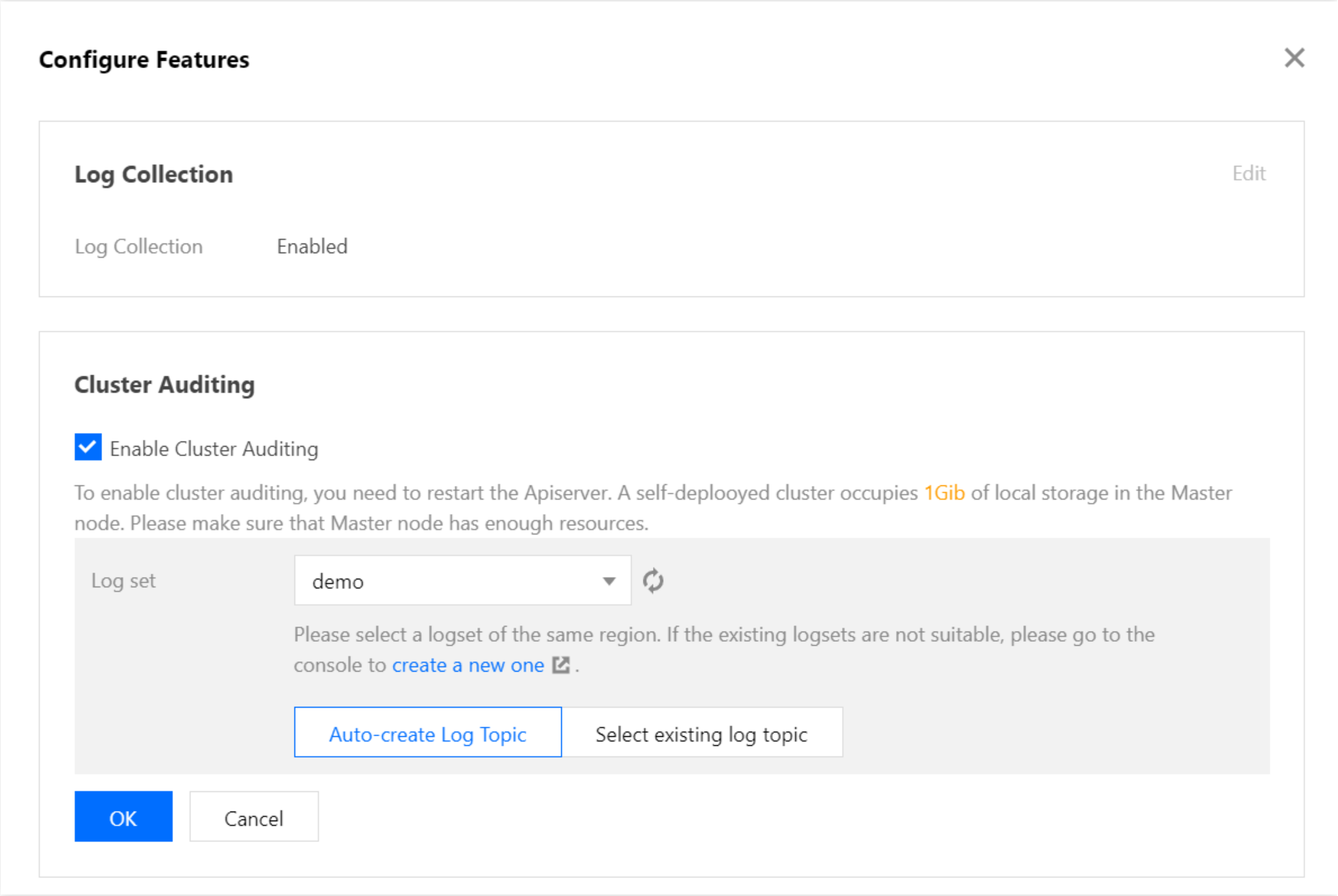
Cluster Auditing Not enabled

Event Storage Edit

Event Storage Not enabled

Close

5. Select **Enable Cluster Audit** and select the logset and log topic for storing audit logs. We recommend that you select **Automatically Create a Log Topic**.



Configure Features

Log Collection

Log Collection Enabled

Cluster Auditing

☑ Enable Cluster Auditing

To enable cluster auditing, you need to restart the Apiserver. A self-deplooyed cluster occupies 1Gib of local storage in the Master node. Please make sure that Master node has enough resources.

Log set: demo

Please select a logset of the same region. If the existing logsets are not suitable, please go to the console to create a new one .

6. Click **OK** to enable the cluster audit feature.

Auditing Dashboard

Last updated：2022-07-21 15:58:03

Overview

TKE provides users with an out-of-the-box audit dashboard and can automatically configure dashboards of auditing overview, node operation overview, K8s object operation overview, and aggregation search for the clusters with the feature of **Cluster Auditing** enabled. With user-defined filter items, and built-in CLS global search, TKE makes it convenient for users to observe and search various cluster operations, so as to find and locate problems in time.

Feature Description

Five dashboards are configured in the **Auditing search**, namely **Auditing overview**, **Node operation overview**, **K8s object operation overview**, **Aggregation search**, and **Global search**. Follow the steps below to go to the **Auditing search** page and use the corresponding features:

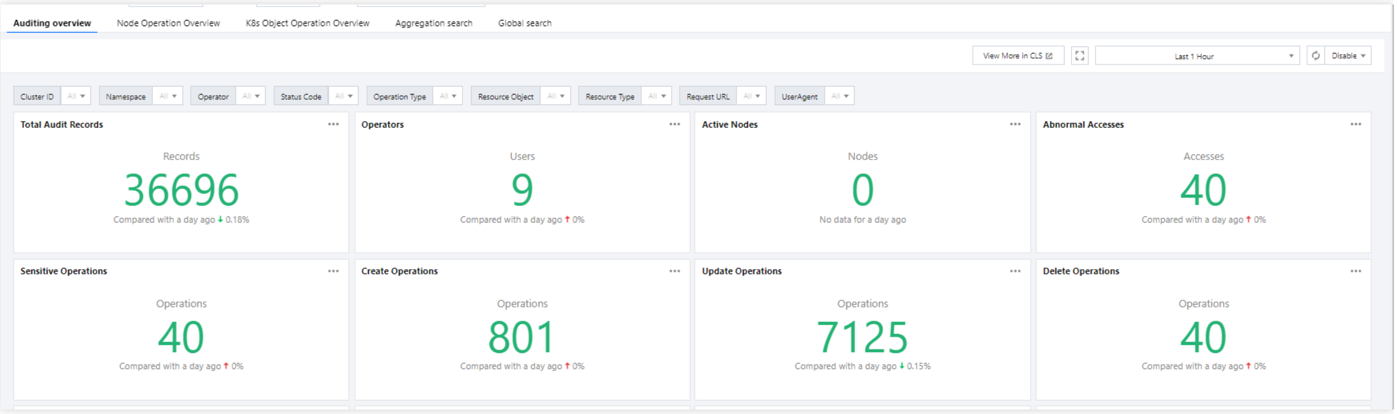
1. Log in to the TKE console.
2. Enable the Cluster Auditing feature. For more information, see Cluster Auditing.
3. Select **Log Management > Audit Logs** in the left sidebar to go to the “Audit log search” page.

Auditing overview

When you want to view the operation of the entire cluster APIserver, you can set filter conditions on the "Auditing overview" page, view the summary statistics of the core audit log, and display the data comparison within a period, for example, core audit log statistics, distribution, important operation trends, etc.

You can view more statistics on this page, as shown below:

- Core audit log statistics dashboard:



Total Audit Records

Records 36696

Compared with a day ago ↓ 0.18%

Operators

Users 9

Compared with a day ago ↑ 0%

Active Nodes

Nodes 0

No data for a day ago

Abnormal Accesses

Accesses 40

Compared with a day ago ↑ 0%

Sensitive Operations

Operations 40

Compared with a day ago ↑ 0%

Create Operations

Operations 801

Compared with a day ago ↑ 0%

Update Operations

Operations 7125

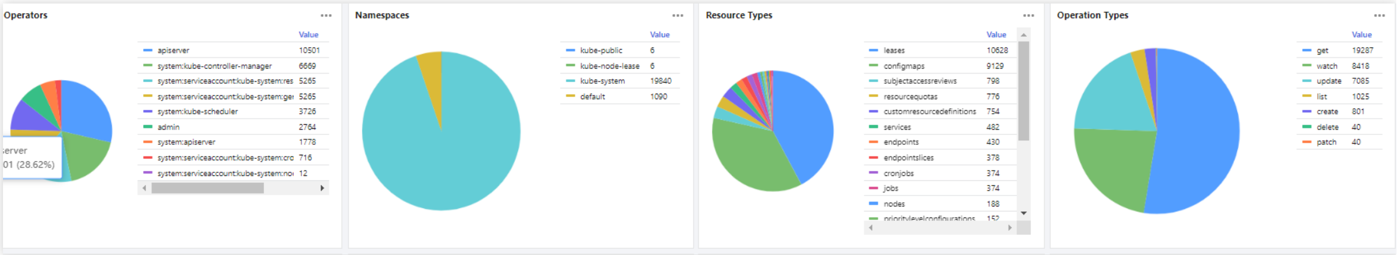
Compared with a day ago ↓ 0.15%

Delete Operations

Operations 40

Compared with a day ago ↑ 0%

- Distribution dashboard:



Operators

	Value
apiserver	10501
system:kube-controller-manager	6669
system:serviceaccount:kube-system:res	5265
system:serviceaccount:kube-system:ge	5265
system:kube-scheduler	3726
admin	2764
system:apiserver	1778
system:serviceaccount:kube-system:cro	716
system:serviceaccount:kube-system:no	12

Namespaces

	Value
kube-public	6
kube-node-lease	6
kube-system	19840
default	1090

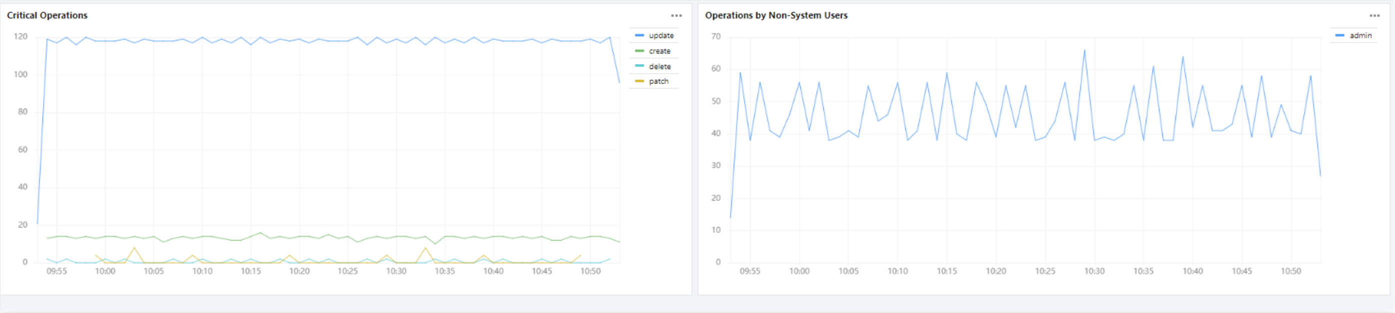
Resource Types

	Value
leases	10628
configmaps	9129
subjectaccessreviews	798
resourcequotas	776
customresourcedefinitions	754
services	482
endpoints	430
endpointslices	378
cronjobs	374
jobs	374
nodes	188
prioritylevelconfigurations	152

Operation Types

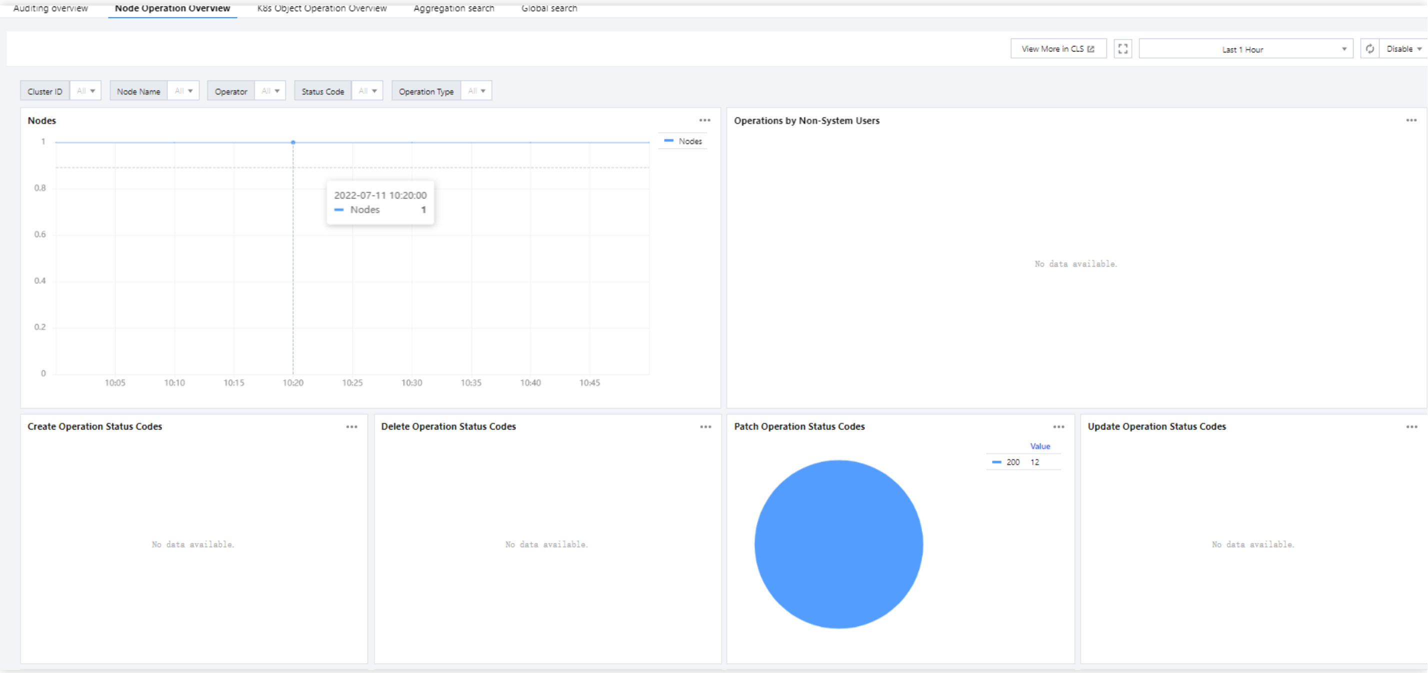
	Value
get	19287
watch	8418
update	7085
list	1025
create	801
delete	40
patch	40

- Important operation trend dashboard:



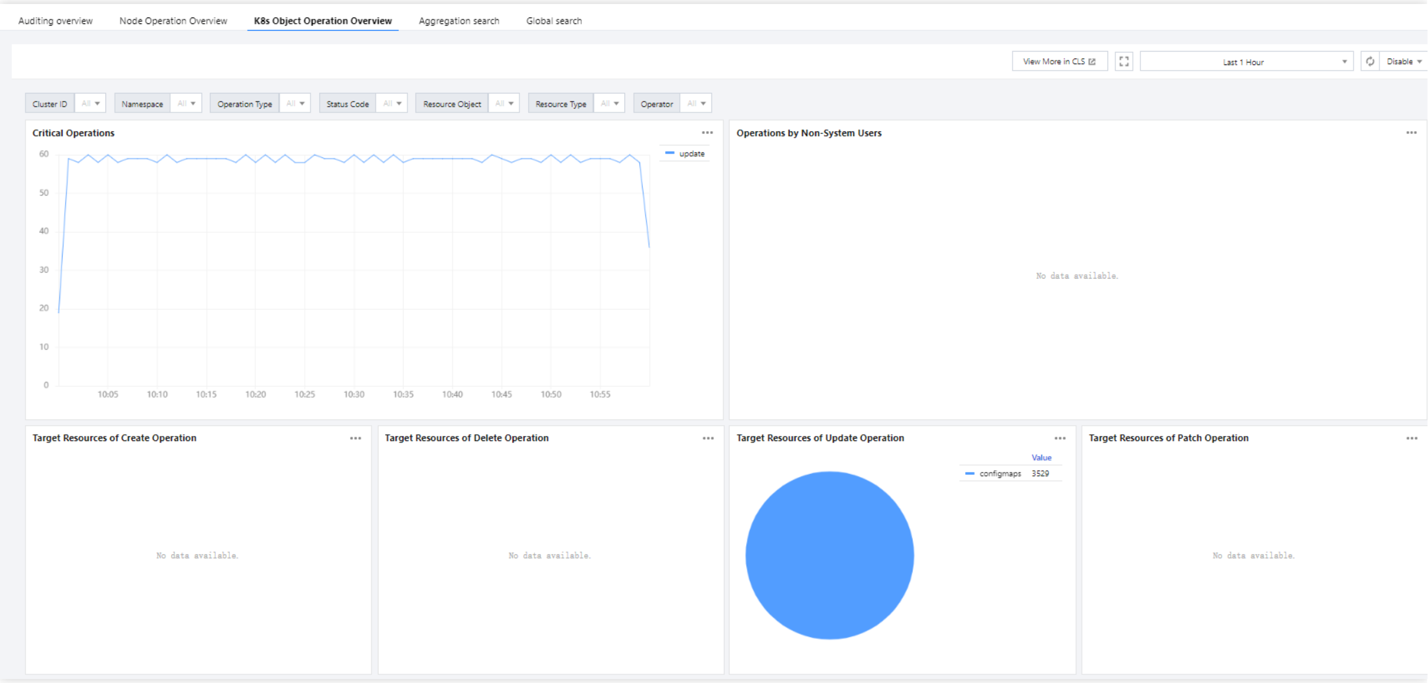
Node operation overview

When you need to troubleshoot node problems, you can set filters on the **Node operation overview** page to view dashboards of various node operations, including `create` , `delete` , `patch` , `update` , `block` , and `evict` .



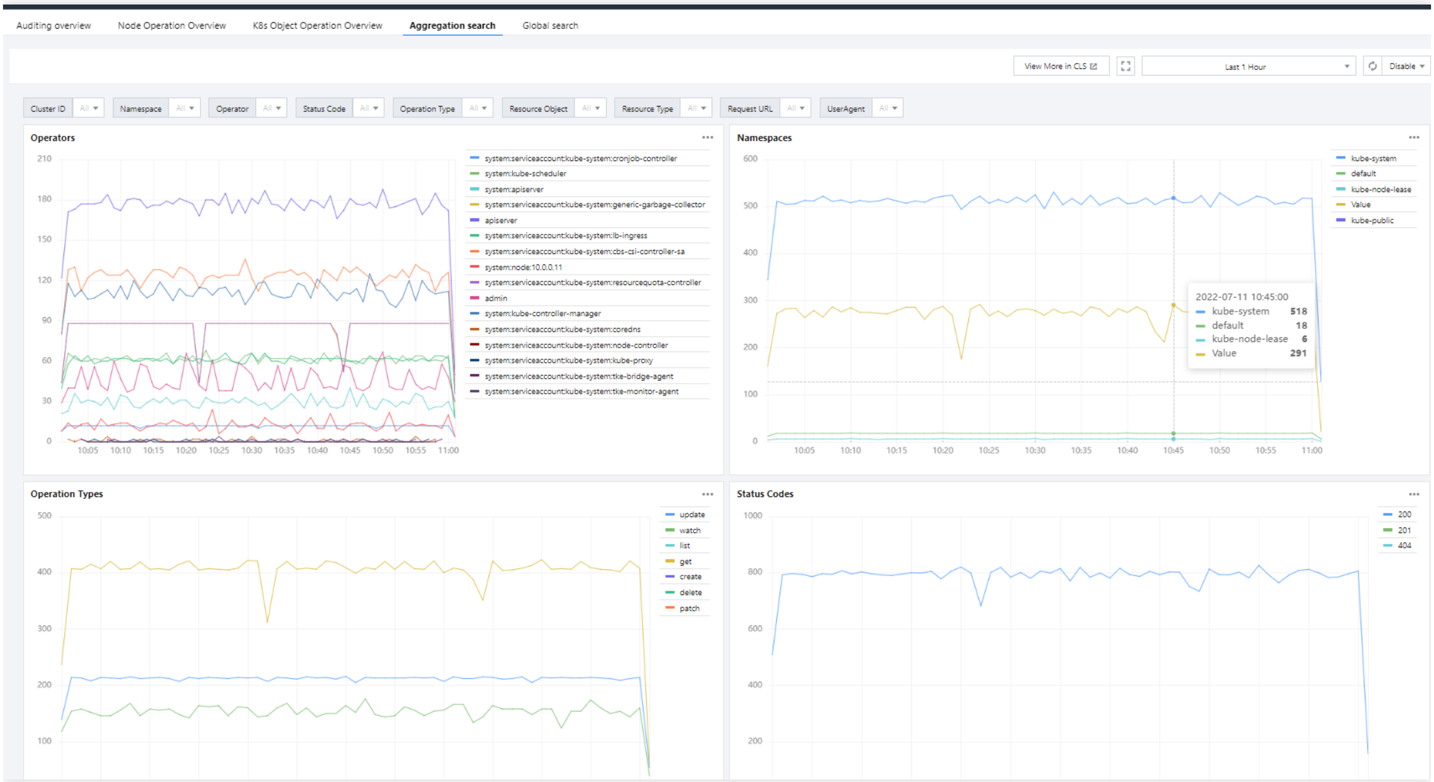
K8s object operation overview

When you need to troubleshoot problems related to K8s objects (such as a certain workload), you can go to the **K8s object operation overview** page, and set filter conditions to view the details of the operation overview, the corresponding users, and the corresponding audit log list of various K8s objects.



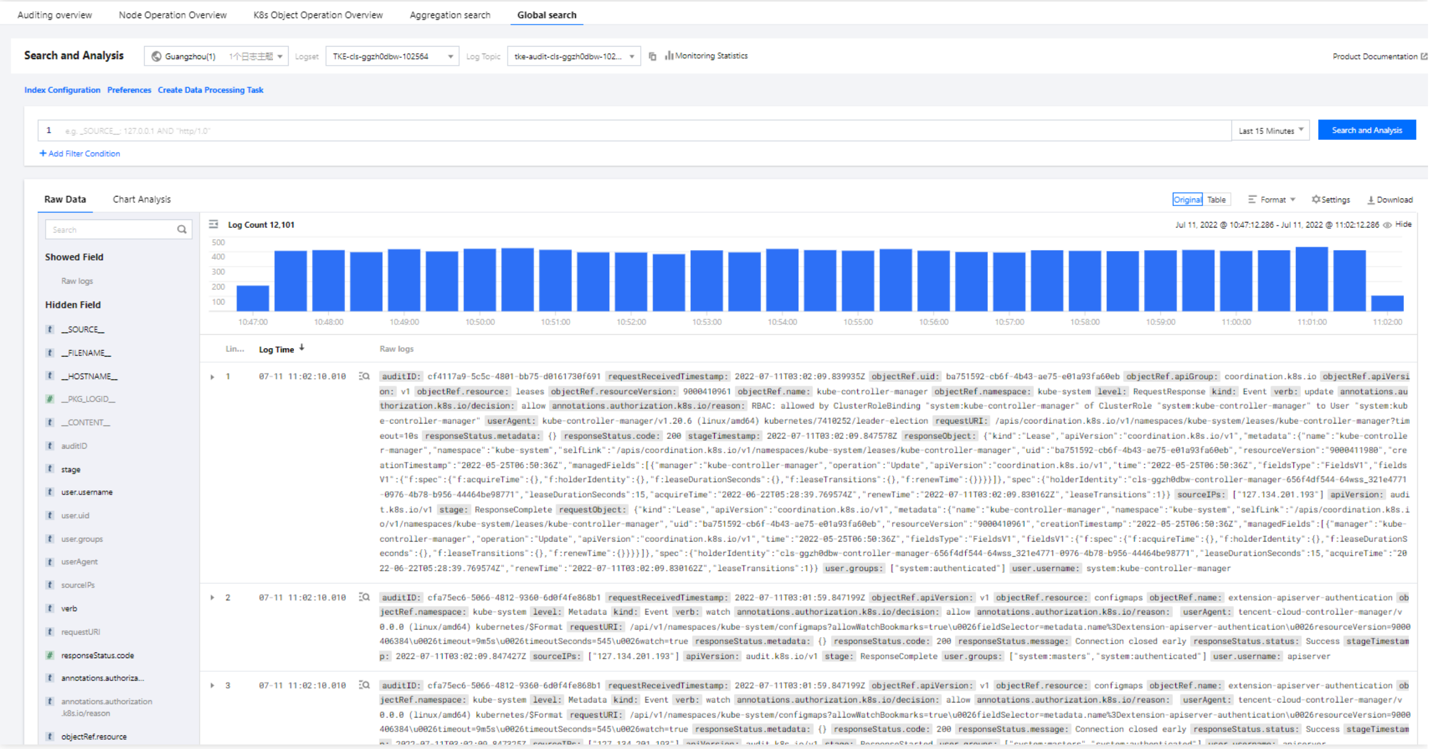
Aggregation search

When you want to view the distribution trend of audit logs in a certain dimension, you can set filter conditions on the "Aggregation search" page to view the sequence diagrams of various important operations. The dimensions include the user, namespace, operation type, status code, resource type, and the corresponding audit log list.



Global search

Global search dashboard, with built-in CLS search analysis page, is convenient for users to quickly search all audit logs in the TKE console.



```
1  07-11 11:02:10.010  auditID: cf4117a9-5c5c-4801-bb75-d0161730f691 requestReceivedTimestamp: 2022-07-11T03:02:09.839935Z objectRef.uid: ba751592-cb6f-4b43-ae75-e01a93fa60eb objectRef.apiGroup: coordination.k8s.io objectRef.apiVersi
on: v1 objectRef.resource: leases objectRef.resourceVersion: 9000410961 objectRef.name: kube-controller-manager objectRef.namespace: kube-system level: RequestResponse kind: Event verb: update annotations.au
thorization.k8s.io/decision: allow annotations.authorization.k8s.io/reason: RBAC: allowed by ClusterRoleBinding "system:kube-controller-manager" of ClusterRole "system:kube-controller-manager" to User "system:kub
e-controller-manager" userAgent: kube-controller-manager/v1.20.6 (linux/amd64) kubernetes/7410252/leader-election requestURI: /apis/coordination.k8s.io/v1/namespaces/kube-system/leases/kube-controller-manager?tim
eout=10s responseStatus.metadata: {} responseStatus.code: 200 stageTimestamp: 2022-07-11T03:02:09.847578Z responseObject: {"kind":"Lease","apiVersion":"coordination.k8s.io/v1","metadata":{"name":"kube-controlle
r-manager","namespace":"kube-system","selfLink":"/apis/coordination.k8s.io/v1/namespaces/kube-system/leases/kube-controller-manager","uid":"ba751592-cb6f-4b43-ae75-e01a93fa60eb","resourceVersion":"9000411980","cre
ationTimestamp":"2022-05-25T06:50:36Z","managedFields":[{"manager":"kube-controller-manager","operation":"Update","apiVersion":"coordination.k8s.io/v1","time":"2022-05-25T06:50:36Z","fieldsType":"FieldsV1","fields
V1":{"f:spec":{"f:acquireTime":{},"f:holderIdentity":{},"f:leaseDurationSeconds":{},"f:leaseTransitions":{},"f:renewTime":{}}}}]},"spec":{"holderIdentity":"cls-ggzh0dbw-controller-manager-656f4df544-64wss_321e4771
-0976-4b78-b956-44464be98771","leaseDurationSeconds":15,"acquireTime":"2022-06-22T05:28:39.769574Z","renewTime":"2022-07-11T03:02:09.830162Z","leaseTransitions":1}} sourceIPs: ["127.134.201.193"] apiVersion: audi
t.k8s.io/v1 stage: ResponseComplete requestObject: {"kind":"Lease","apiVersion":"coordination.k8s.io/v1","metadata":{"name":"kube-controller-manager","namespace":"kube-system","selfLink":"/apis/coordination.k8s.i
o/v1/namespaces/kube-system/leases/kube-controller-manager","uid":"ba751592-cb6f-4b43-ae75-e01a93fa60eb","resourceVersion":"9000410961","creationTimestamp":"2022-05-25T06:50:36Z","managedFields":[{"manager":"kube-
controller-manager","operation":"Update","apiVersion":"coordination.k8s.io/v1","time":"2022-05-25T06:50:36Z","fieldsType":"FieldsV1","fieldsV1":{"f:spec":{"f:acquireTime":{},"f:holderIdentity":{},"f:leaseDurationS
econds":{},"f:leaseTransitions":{},"f:renewTime":{}}}}]},"spec":{"holderIdentity":"cls-ggzh0dbw-controller-manager-656f4df544-64wss_321e4771-0976-4b78-b956-44464be98771","leaseDurationSeconds":15,"acquireTime":"20
22-06-22T05:28:39.769574Z","renewTime":"2022-07-11T03:02:09.830162Z","leaseTransitions":1}} user.groups: ["system:authenticated"] user.username: system:kube-controller-manager

2  07-11 11:02:10.010  auditID: cfa75ec6-5066-4812-9360-6d0f4fe868b1 requestReceivedTimestamp: 2022-07-11T03:01:59.847199Z objectRef.apiVersion: v1 objectRef.resource: configmaps objectRef.name: extension-apiserver-authentication ob
jectRef.namespace: kube-system level: Metadata kind: Event verb: watch annotations.authorization.k8s.io/decision: allow annotations.authorization.k8s.io/reason:  userAgent: tencent-cloud-controller-manager/v
0.0.0 (linux/amd64) kubernetes/$Format requestURI: /api/v1/namespaces/kube-system/configmaps?allowWatchBookmarks=true\u0026fieldSelector=metadata.name%3Dextension-apiserver-authentication\u0026resourceVersion=9000
406384\u0026timeout=9m5s\u0026timeoutSeconds=545\u0026watch=true responseStatus.metadata: {} responseStatus.code: 200 responseStatus.message: Connection closed early responseStatus.status: Success stageTimestam
p: 2022-07-11T03:02:09.847427Z sourceIPs: ["127.134.201.193"] apiVersion: audit.k8s.io/v1 stage: ResponseComplete user.groups: ["system:masters","system:authenticated"] user.username: apiserver

3  07-11 11:02:10.010  auditID: cfa75ec6-5066-4812-9360-6d0f4fe868b1 requestReceivedTimestamp: 2022-07-11T03:01:59.847199Z objectRef.apiVersion: v1 objectRef.resource: configmaps objectRef.name: extension-apiserver-authentication ob
jectRef.namespace: kube-system level: Metadata kind: Event verb: watch annotations.authorization.k8s.io/decision: allow annotations.authorization.k8s.io/reason:  userAgent: tencent-cloud-controller-manager/v
0.0.0 (linux/amd64) kubernetes/$Format requestURI: /api/v1/namespaces/kube-system/configmaps?allowWatchBookmarks=true\u0026fieldSelector=metadata.name%3Dextension-apiserver-authentication\u0026resourceVersion=9000
406384\u0026timeout=9m5s\u0026timeoutSeconds=545\u0026watch=true responseStatus.metadata: {} responseStatus.code: 200 responseStatus.message: Connection closed early responseStatus.status: Success stageTimestam
p: 2022-07-11T03:02:09.847325Z sourceIPs: ["127.134.201.193"] apiVersion: audit.k8s.io/v1 stage: ResponseStarted user.groups: ["system:masters","system:authenticated"] user.username: apiserver
```

Configuring alarms based on the dashboards

You can configure alarms based on the preset dashboards. When the conditions you set are reached, the alarms will be triggered. The steps are as follows:

1. Click **Quickly add alarm** on the right of the target dashboard.
2. Create an alarm policy in **Alarm Policy** in the **CLS console** as instructed in Configuring Alarm Policies.

Event Management

Event Storage

Last updated：2023-05-06 17:36:46

Note

Users are **exempted from CLS service fees** incurred by audit and event data generated by TKE until June 30, 2022. Users can opt to automatically create a new logset or to automatically create log topics within an existing logset.

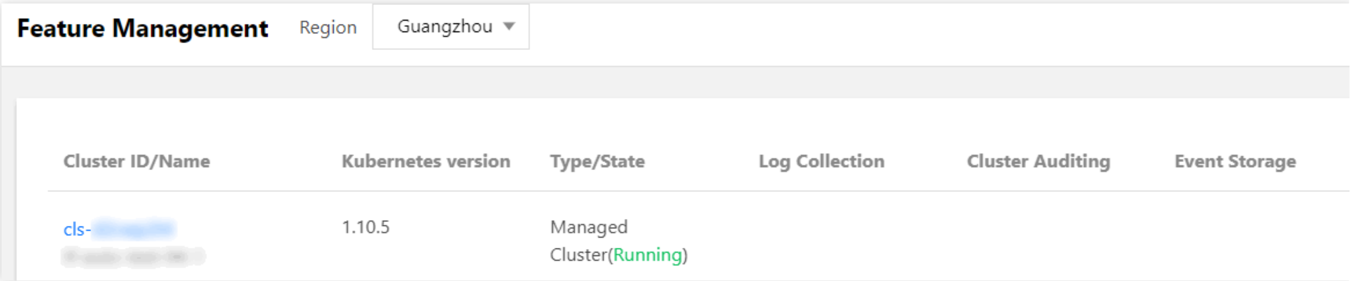
Overview

Kubernetes Events contains information about the operations of Kubernetes clusters and the scheduling of various resources, which can help Ops personnel monitor daily changes in resources and locate problems. TKE supports event persistence for all clusters. After enabling this feature, your cluster events will be exported to the configured storage in real time. TKE also supports the search of event flows by using PaaS services provided by Tencent Cloud or open-source software tools. This document describes how to enable persistent storage of cluster events.

Directions

Enabling event storage

1. Log in to the TKE console.
2. Click **Operation Management** in the left sidebar.
3. At the top of the **Feature Management** page, select a region and a cluster type. Locate the target cluster and click **Settings** on the right, as shown below:

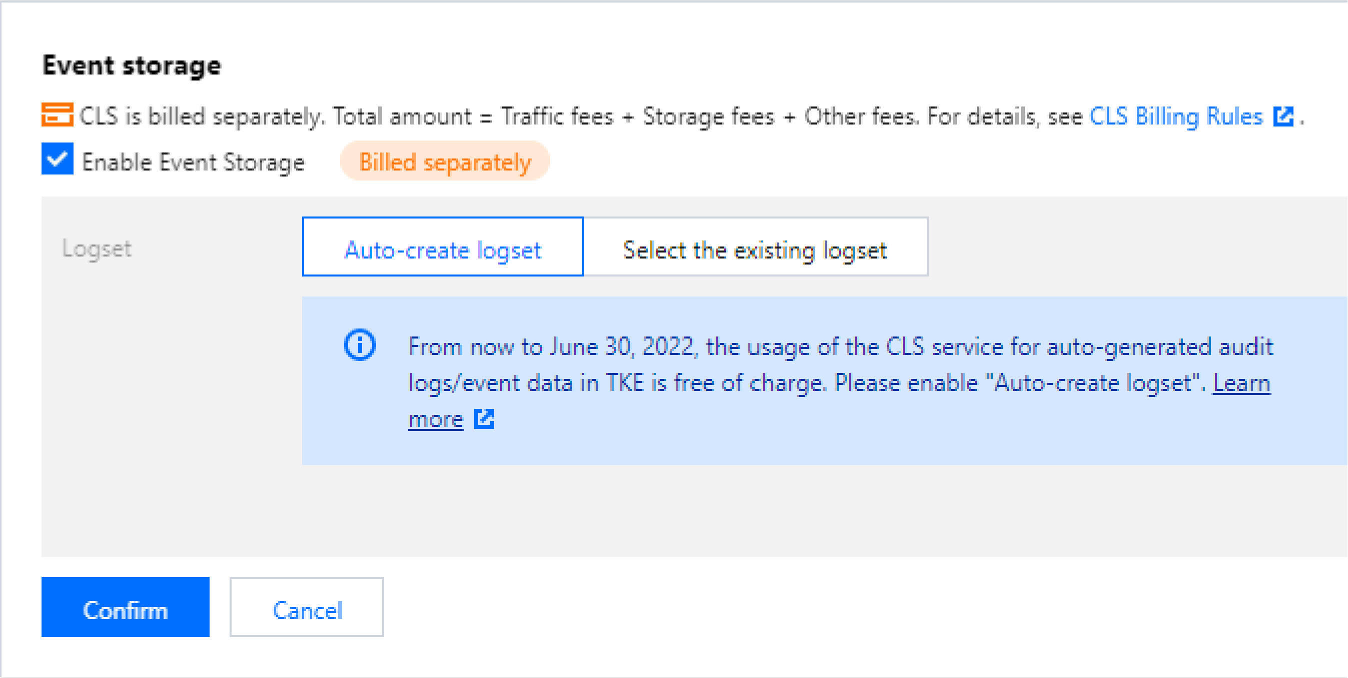


Cluster ID/Name	Kubernetes version	Type/State	Log Collection	Cluster Auditing	Event Storage
cls-[redacted] [redacted]	1.10.5	Managed Cluster(Running)			

4. On the **Configure features** page, click **Edit** on the right for event storage. Select **Enable Event Storage** and configure a logset and a log topic, as shown below:

Note

A logset can contain up to 10 log topics. If you choose automatic creation of log topics, ensure that the selected logset contains less than 10 log topics.



Event storage

CLS is billed separately. Total amount = Traffic fees + Storage fees + Other fees. For details, see CLS Billing Rules.

☑ Enable Event Storage Billed separately

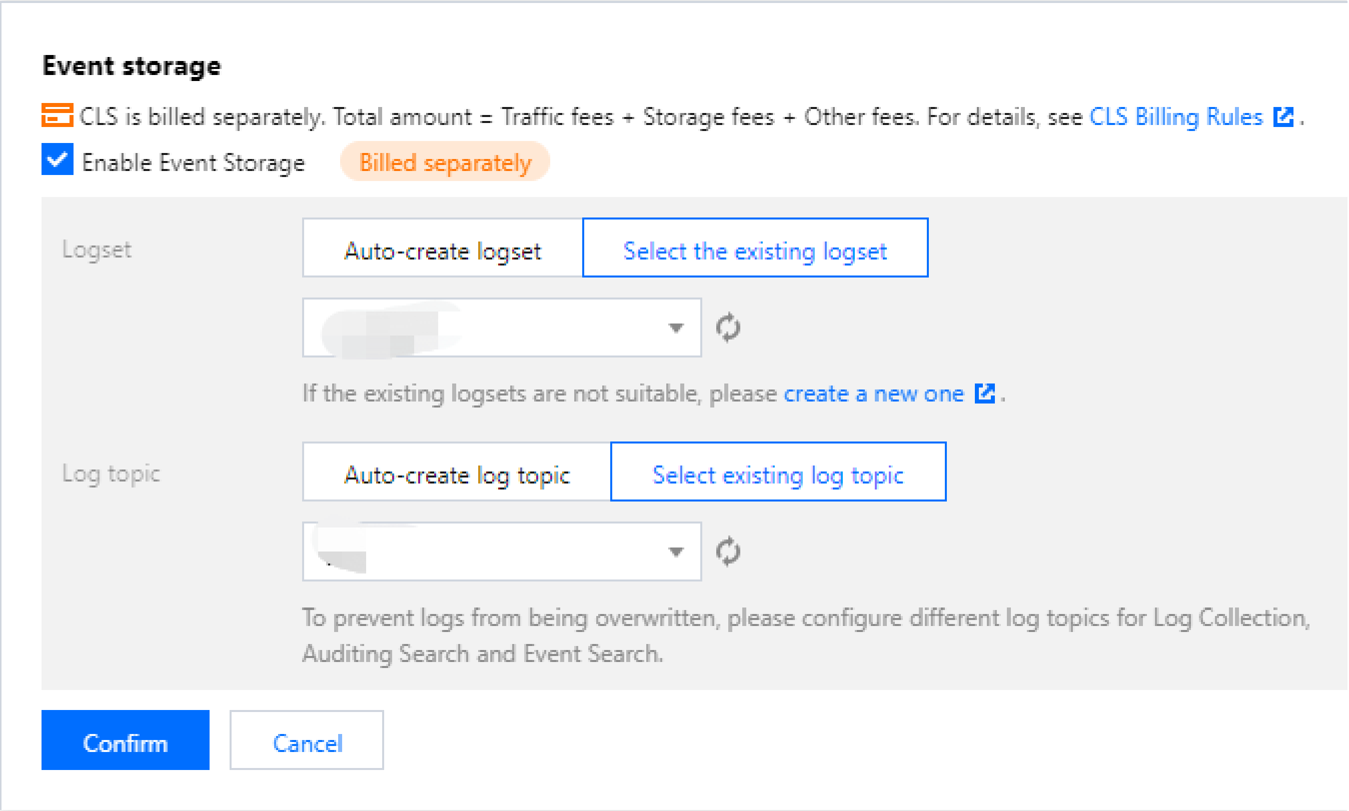
Logset: Auto-create logset | Select the existing logset

From now to June 30, 2022, the usage of the CLS service for auto-generated audit logs/event data in TKE is free of charge. Please enable "Auto-create logset". Learn more

5. Click **OK** to enable event storage.

Updating logsets or log topics

1. Log in to the TKE console.
2. Click **Operation Management** in the left sidebar.
3. At the top of the **Feature Management** page, select a region and a cluster type. Locate the target cluster and click **Settings** on the right.
4. On the **Configure features** page, click **Edit** on the right for event storage. Re-select a logset and a log topic, as shown below:



Event storage

CLS is billed separately. Total amount = Traffic fees + Storage fees + Other fees. For details, see CLS Billing Rules .

☑ Enable Event Storage Billed separately

Logset: Auto-create logset | Select the existing logset

[redacted]

If the existing logsets are not suitable, please create a new one .

Log topic: Auto-create log topic | Select existing log topic

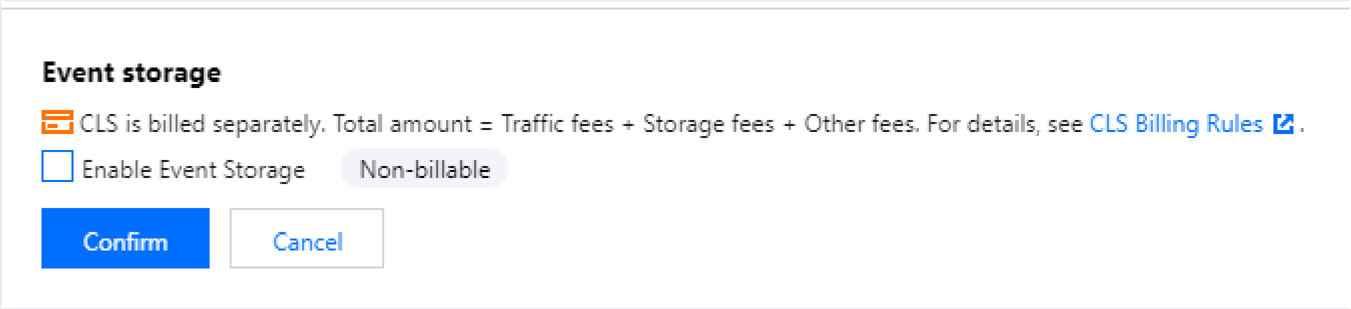
[redacted]

To prevent logs from being overwritten, please configure different log topics for Log Collection, Auditing Search and Event Search.

5. Click **OK** to update the logset and log topic.

Disabling event storage

1. Log in to the TKE console.
2. Click **Operation Management** in the left sidebar.
3. At the top of the **Feature Management** page, select a region and a cluster type. Locate the target cluster and click **Settings** on the right.
4. On the **Configure features** page, click **Edit** on the right for event storage. Deselect **Enable Event Storage**, as shown below:



Event storage

CLS is billed separately. Total amount = Traffic fees + Storage fees + Other fees. For details, see CLS Billing Rules .

□ Enable Event Storage Non-billable

5. Click **OK** to disable event storage.

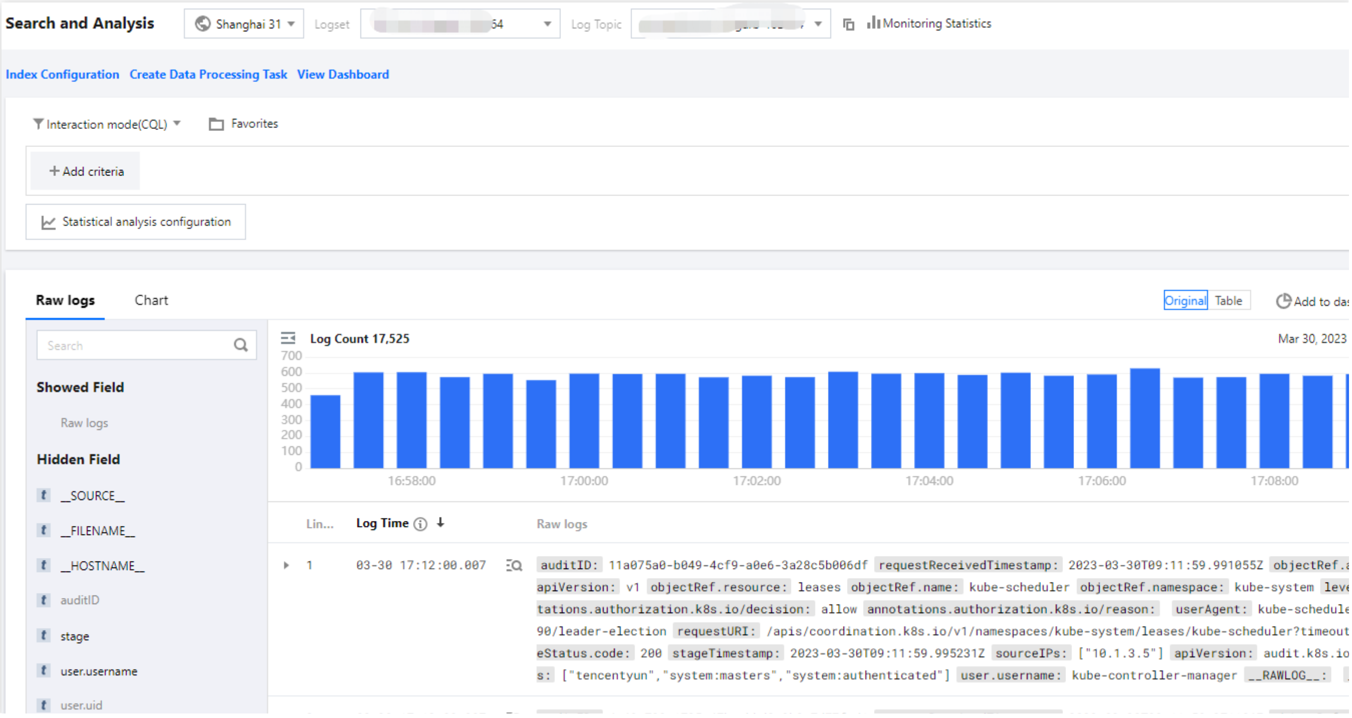
Viewing events in the CLS console

1. Log in to the CLS console.
2. In the left sidebar, click **Search and Analysis**.

3. On the page that appears, select the logset and log topic configured for event storage, configure display fields according to needs, and perform a search and analysis, as shown below:

Note:

When you enable event storage, indexing will be enabled for your log topic by default.



```
1    03-30 17:12:00.007    auditID: 11a075a0-b049-4cf9-a0e6-3a28c5b006df requestReceivedTimestamp: 2023-03-30T09:11:59.991055Z objectRef.a
                           apiVersion: v1 objectRef.resource: leases objectRef.name: kube-scheduler objectRef.namespace: kube-system leve
                           tations.authorization.k8s.io/decision: allow annotations.authorization.k8s.io/reason:  userAgent: kube-schedule
                           90/leader-election requestURI: /apis/coordination.k8s.io/v1/namespaces/kube-system/leases/kube-scheduler?timeout
                           eStatus.code: 200 stageTimestamp: 2023-03-30T09:11:59.995231Z sourceIPs: ["10.1.3.5"] apiVersion: audit.k8s.io
                           s: ["tencentyun","system:masters","system:authenticated"] user.username: kube-controller-manager __RAWLOG__:
```

Event Dashboard

Last updated：2023-05-06 19:41:07

Scenarios

TKE provides users with an out-of-the-box event dashboard and can automatically configure analysis dashboards of event overview and exception events aggregation search for the clusters with the feature of **Event Storage** enabled. With user-defined filter items, and built-in CLS event global search, uses can comprehensively observe, find, analyze, and locate problems in the TKE console.

Description

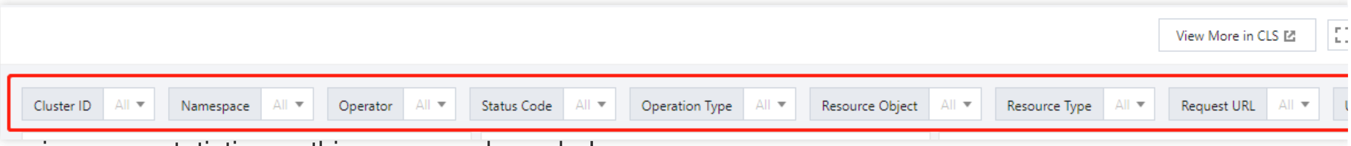
Three dashboards are configured in the **Event Search**, namely **Event Overview**, **Exception Events Aggregation Search**, and **Global Search**. Please follow the steps below to go to the **Event Search** page and use the corresponding features:

1. Log in to the TKE console.
2. Enable **Event Storage**. For more information, see Event Storage.
3. Select **Log Management** > **Event Logs** in the left sidebar.
4. At the top of the **Event Search** page, select the region and cluster type to view the cluster event details.

Event overview

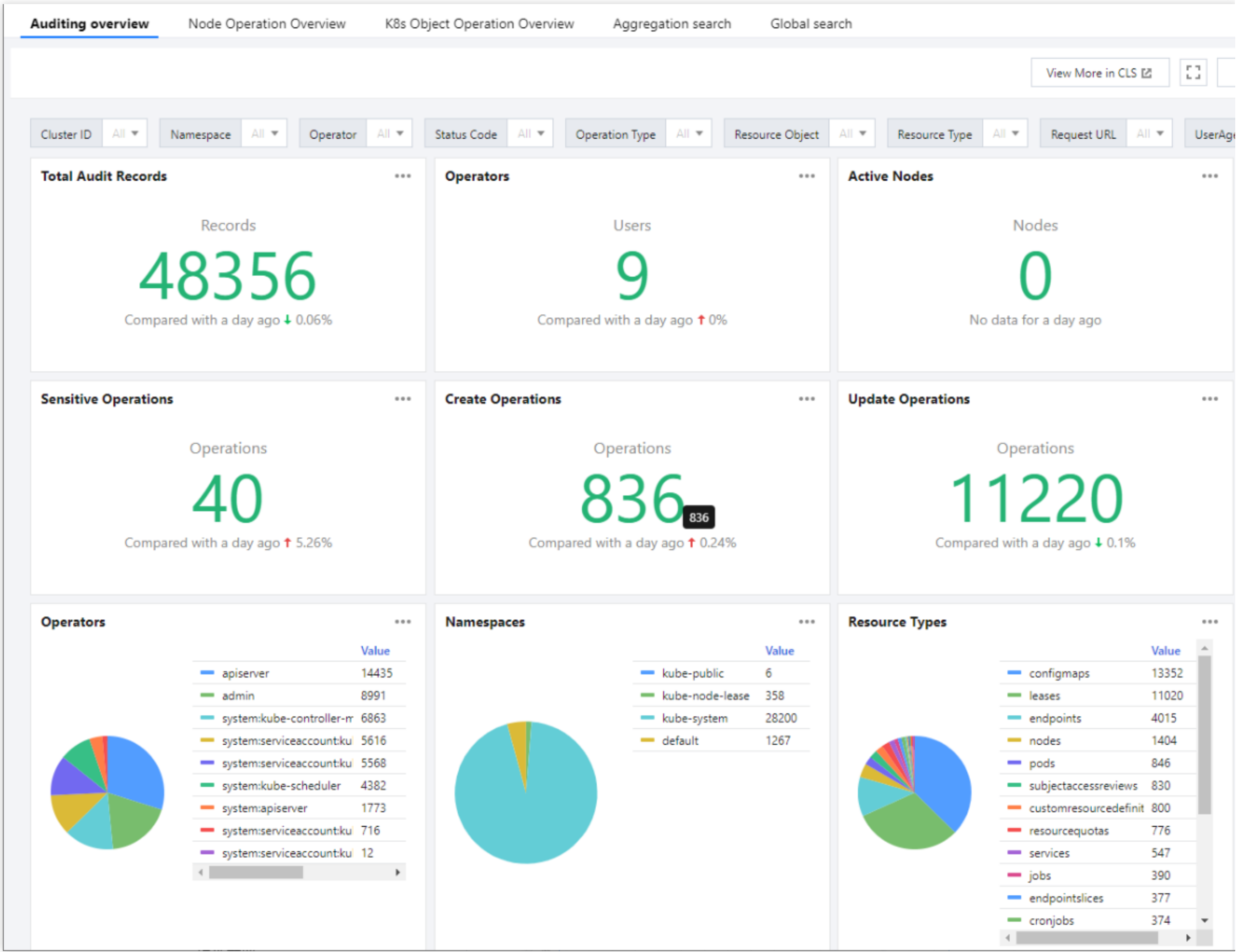
On the **Event Overview** page, you can filter events based on dimensions such as the cluster ID, namespace, level, reason, resource type, resource object event source, view summary statistics of core events, and display data comparison within a period, for example, dashboards of the total number and distribution of events, node exceptions, Pod OOM, important event trends, and top exception event lists.

You can customize the filter items as needed, as shown in the figure below:



You can view more statistics on this page, as shown below:

Total number of events, level distribution, exception event reason and object distribution:



Total Audit Records

Records

48356

Compared with a day ago ↓ 0.06%

Operators

Users

9

Compared with a day ago ↑ 0%

Active Nodes

Nodes

0

No data for a day ago

Sensitive Operations

Operations

40

Compared with a day ago ↑ 5.26%

Create Operations

Operations

836

Compared with a day ago ↑ 0.24%

Update Operations

Operations

11220

Compared with a day ago ↓ 0.1%

Operators

	Value
apiserver	14435
admin	8991
system:kube-controller-m	6863
system:serviceaccount:ku	5616
system:serviceaccount:ku	5568
system:kube-scheduler	4382
system:apiserver	1773
system:serviceaccount:ku	716
system:serviceaccount:ku	12

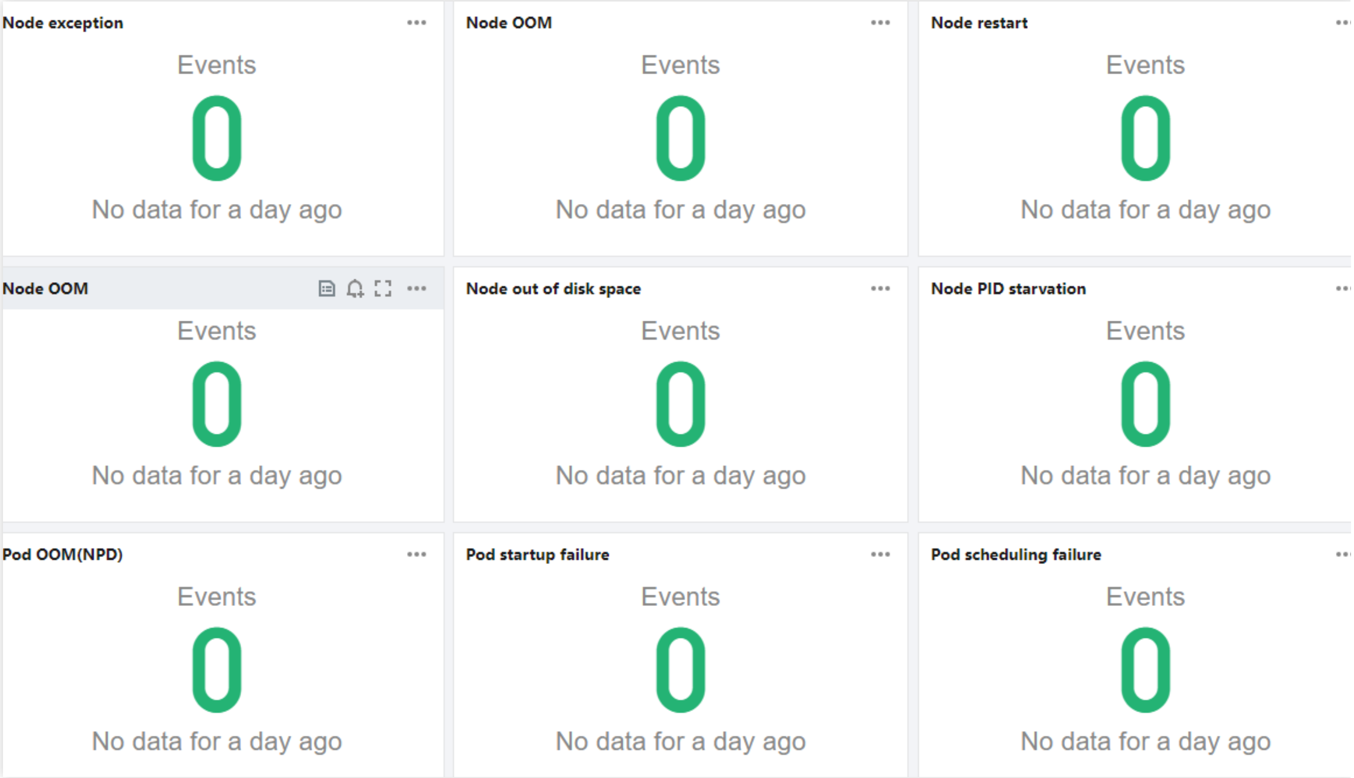
Namespaces

	Value
kube-public	6
kube-node-lease	358
kube-system	28200
default	1267

Resource Types

	Value
configmaps	13352
leases	11020
endpoints	4015
nodes	1404
pods	846
subjectaccessreviews	830
customresourcedefinit	800
resourcequotas	776
services	547
jobs	390
endpointslices	377
cronjobs	374

Summary of various common events:



Node exception
Events
0
No data for a day ago

Node OOM
Events
0
No data for a day ago

Node restart
Events
0
No data for a day ago

Node OOM
Events
0
No data for a day ago

Node out of disk space
Events
0
No data for a day ago

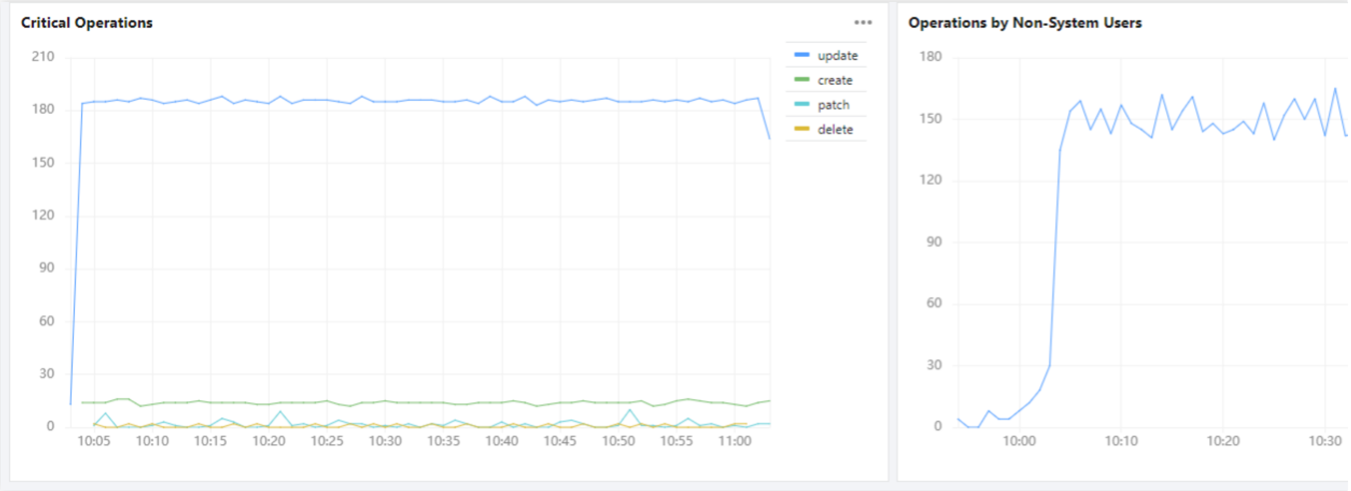
Node PID starvation
Events
0
No data for a day ago

Pod OOM(NPD)
Events
0
No data for a day ago

Pod startup failure
Events
0
No data for a day ago

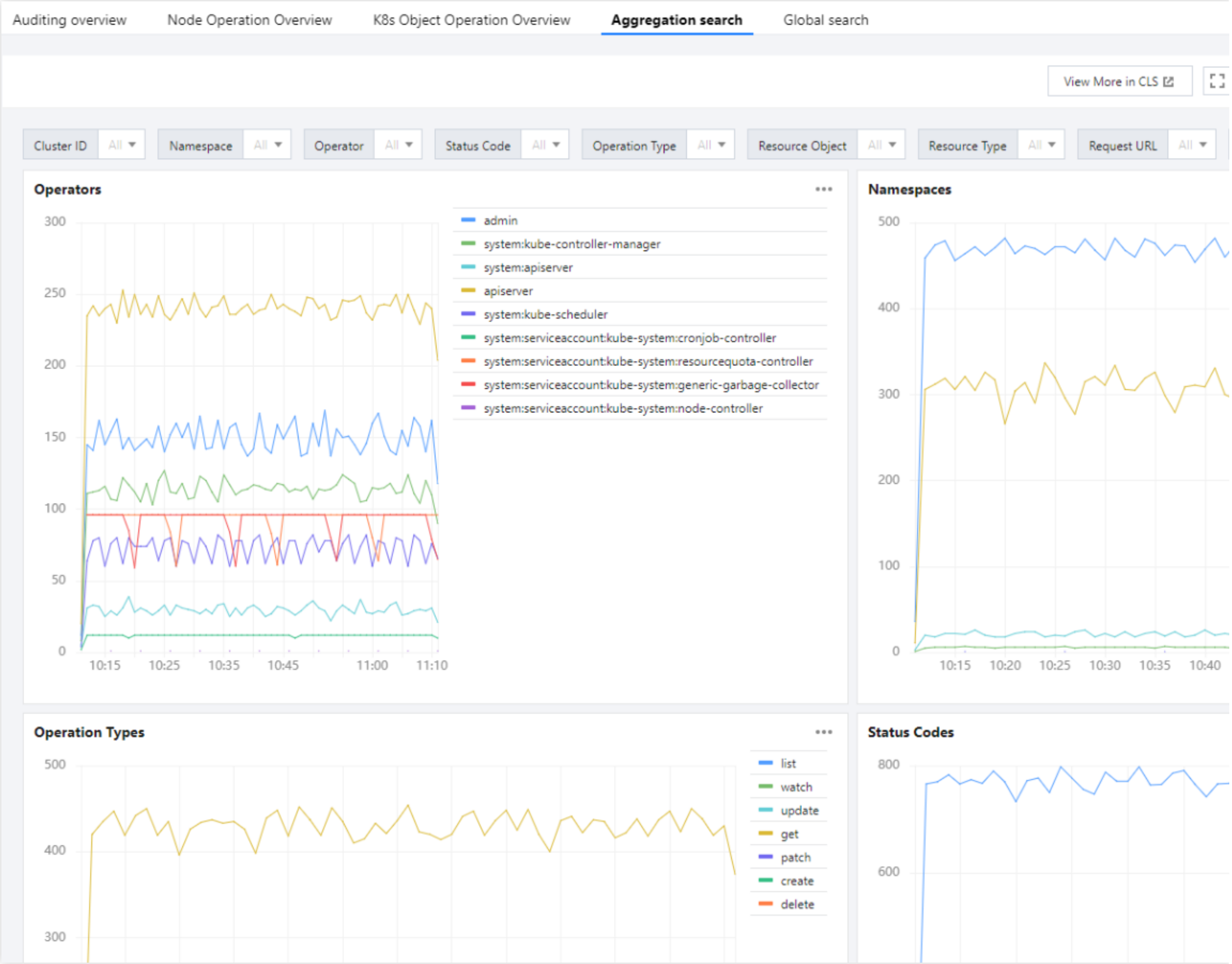
Pod scheduling failure
Events
0
No data for a day ago

Event trends and top exception events list:



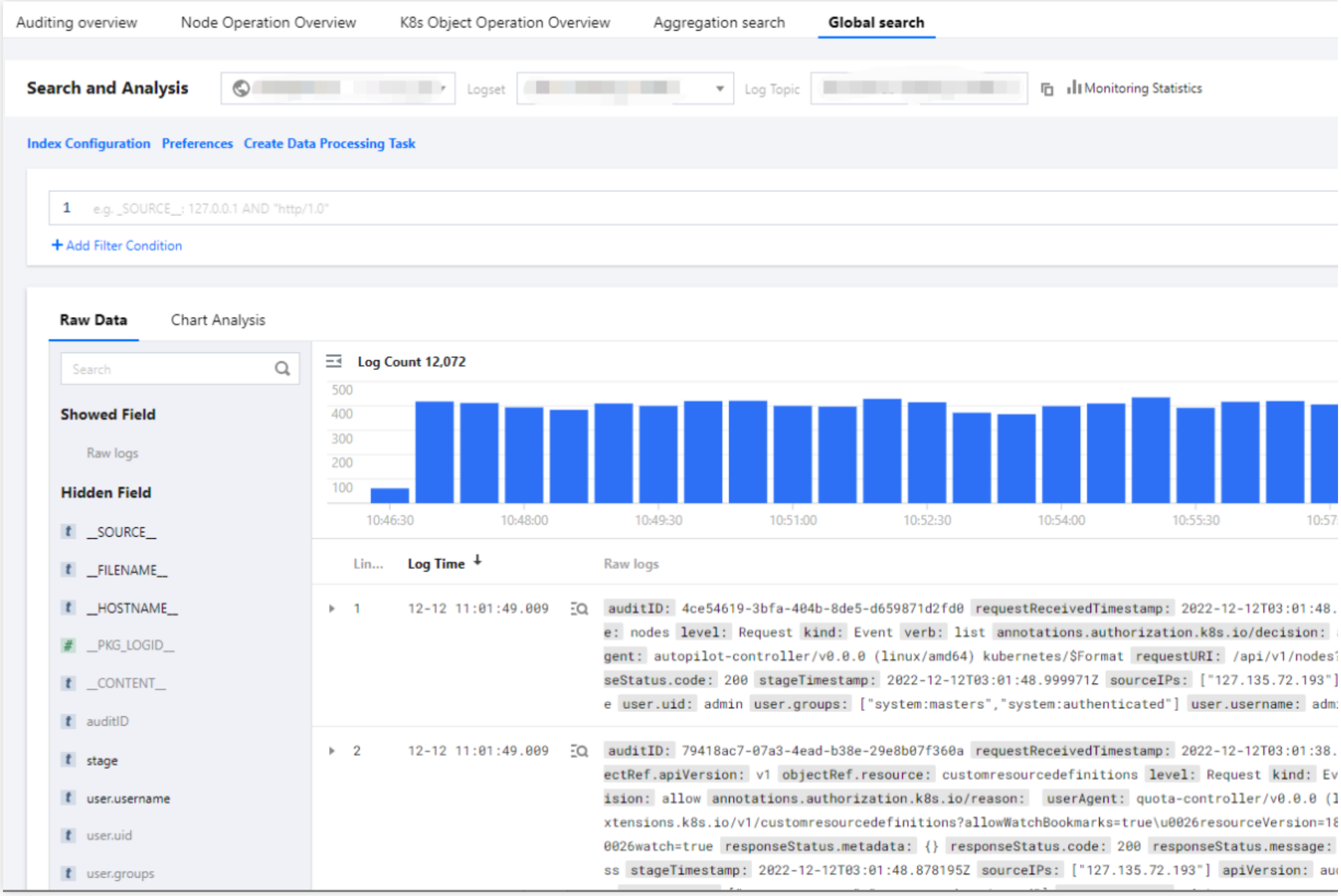
Exception events aggregation search

On the **Exception Events Aggregation Search** page, you can set filter conditions to view the reason and object distribution trends of various exception events in a certain period of time. You can also search the exception events in the list below the trend diagrams to quickly locate the problems as shown below:



Global search

Global search dashboard, with built-in CLS search analysis page, is convenient for users to quickly search all events in the TKE console as shown below:

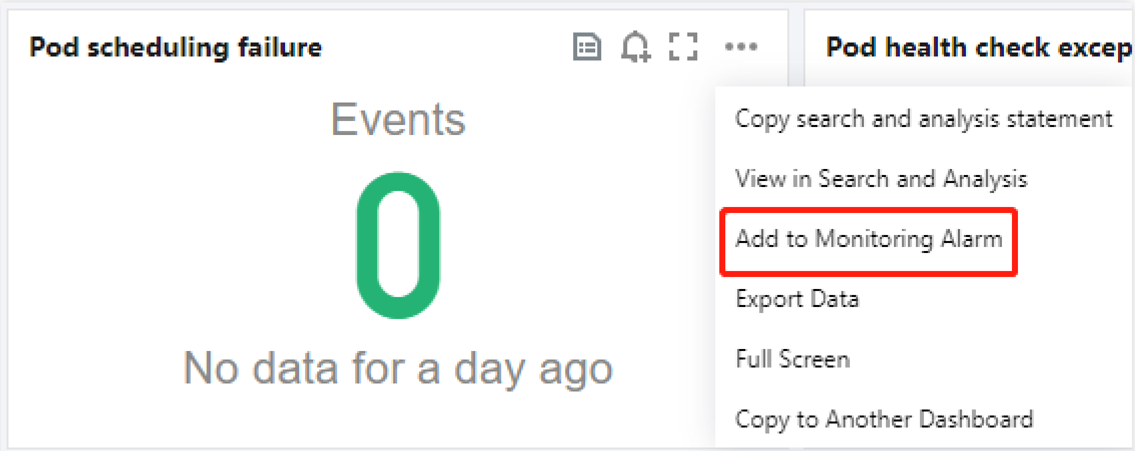


```
1  12-12 11:01:49.009  auditID: 4ce54619-3bfa-404b-8de5-d659871d2fd0 requestReceivedTimestamp: 2022-12-12T03:01:48.
                       e: nodes level: Request kind: Event verb: list annotations.authorization.k8s.io/decision:
                       gent: autopilot-controller/v0.0.0 (linux/amd64) kubernetes/$Format requestURI: /api/v1/nodes
                       seStatus.code: 200 stageTimestamp: 2022-12-12T03:01:48.999971Z sourceIPs: ["127.135.72.193"]
                       e user.uid: admin user.groups: ["system:masters","system:authenticated"] user.username: adm
2  12-12 11:01:49.009  auditID: 79418ac7-07a3-4ead-b38e-29e8b07f360a requestReceivedTimestamp: 2022-12-12T03:01:38.
                       ectRef.apiVersion: v1 objectRef.resource: customresourcedefinitions level: Request kind: Ev
                       ision: allow annotations.authorization.k8s.io/reason: userAgent: quota-controller/v0.0.0 (l
                       xtensions.k8s.io/v1/customresourcedefinitions?allowWatchBookmarks=true\u0026resourceVersion=18
                       0026watch=true responseStatus.metadata: {} responseStatus.code: 200 responseStatus.message:
                       ss stageTimestamp: 2022-12-12T03:01:48.878195Z sourceIPs: ["127.135.72.193"] apiVersion: au
```

Configuring alarms based on the dashboards

You can configure alarms based on the preset dashboards. When the conditions you set are reached, the alarms will be triggered. The steps are as follows:

1. Click **Add to Monitoring Alarm** on the right of the target dashboard.



2. Create an alarm policy in **Alarm Policy** in the CLS console as instructed in Configuring Alarming Policies.

Health Check

Last updated：2021-01-27 15:06:22

Overview

The cluster health check feature is a service provided by Tencent Kubernetes Engine (TKE) for checking the status and health of each resource in a cluster. The resulting check report displays the detailed status and configuration of components, nodes, workloads, and other check items. If an exception is detected, this feature can describe the exception in detail, automatically analyze the severity, cause, and impact, and propose rectification suggestions.

⚠ **Note：**

During the health check, namespace **tke-cluster-inspection** will be automatically created in your cluster, and a Daemonset will be installed to collect node information. Both objects will be automatically deleted after the health check is completed.

Main Check Items

Check Category	Check Item	Check Content	Self-Deployed Clusters Only
Resource status	kube-apiserver status	Check whether the component is running. If the component runs as a pod, the health check feature checks whether it has restarted over the past 24 hours	Yes
	kube-scheduler status		Yes
	kube-controller-manager status		Yes
	etcd status		Yes
	kubelet status		No
	kube-proxy status		No
	dockerd status		No

Check Category	Check Item	Check Content	Self-Deployed Clusters Only
	Master node status	Check whether the node status is Ready and free of any other exceptions, such as insufficient memory and insufficient disk space	Yes
	Worker node status	Check whether the node status is Ready and free of any other exceptions, such as insufficient memory and insufficient disk space	No
	Status of each workload	Check whether the number of currently available pods of the workload meets the expected number of pods	No
Running status	Parameter configuration of kube-apiserver	Check the following parameters based on the master node configuration: • max-requests-inflight: maximum number of non-change requests running in a given period • max-mutating-requests-inflight: maximum number of change requests running in a given period	Yes
	Parameter configuration of kube-scheduler	Check the following parameters based on the master node configuration: • kube-api-qps: QPS of kube-apiserver resquests • kube-api-burst: maximum burst value during communication with kube-apiserver	Yes
	Parameter configuration of kube-controller-manager	Check the following parameters based on the master node configuration: • kube-api-qps: QPS of kube-apiserver requests • kube-api-burst: maximum burst value during communication with kube-apiserver	Yes

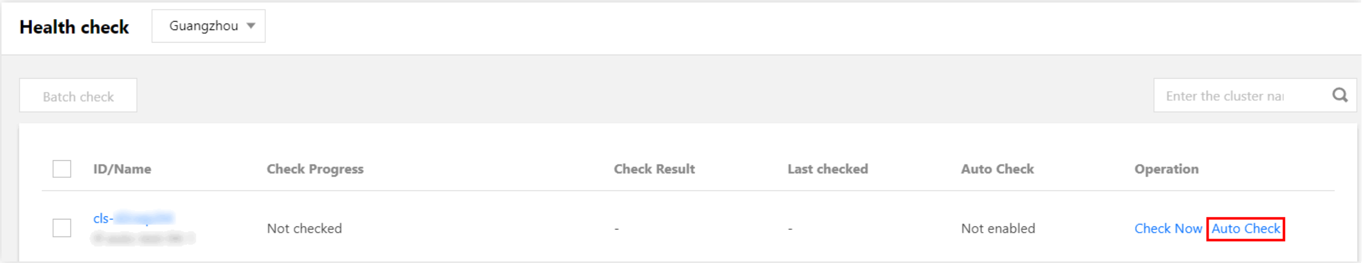
Check Category	Check Item	Check Content	Self-Deployed Clusters Only
	Parameter configuration of etcd	Check the following parameter based on the master node configuration: quota-backend-bytes: storage capacity	Yes
	Reasonability of the master node configuration	Check whether the current master node configuration is sufficient to the current cluster scale	Yes
	High availability of nodes	Check whether the current cluster is a single-node cluster. Check whether current cluster nodes support multi-AZ disaster recovery That is, the health check feature checks whether the total number of resources in other availability zones is sufficient to the current cluster business scale in the event that one availability zone becomes unavailable	No
	Request and Limit configuration of workloads	Check whether workloads have configured resource-limiting containers. Configuring resource limits helps improve resource planning, pod scheduling, cluster availability, and other functions	No
	Anti-affinity configuration of workloads	Check whether workloads have configured affinity or anti-affinity. Configuring anti-affinity helps improve the high availability of business	No
	PDB configuration of workloads	Check whether workloads have configured PDB, which can help prevent your business from becoming unavailable due to eviction.	No
	Health check configuration of workloads	Check whether a health check is configured for workloads. Health check helps detect business exceptions	No
	HPA-IP configuration	Check whether the current number of remaining pod IP addresses in the cluster meets the maximum number for HPA scale-out	No

Directions

1. Log in to the Tencent Kubernetes Engine console and choose **Cluster OPS** > **Health Check** in the left sidebar.
2. On the **Health Check** page that appears, select the health check mode for the cluster.

   In the Health Check page, there’re three options: Batch Check, Check Now, and Auto Check.

   - **Batch Check**: simultaneously checks multiple clusters.
   - **Check Now**: checks only one cluster.
   - **Auto Check**: enables periodic health check for the cluster.

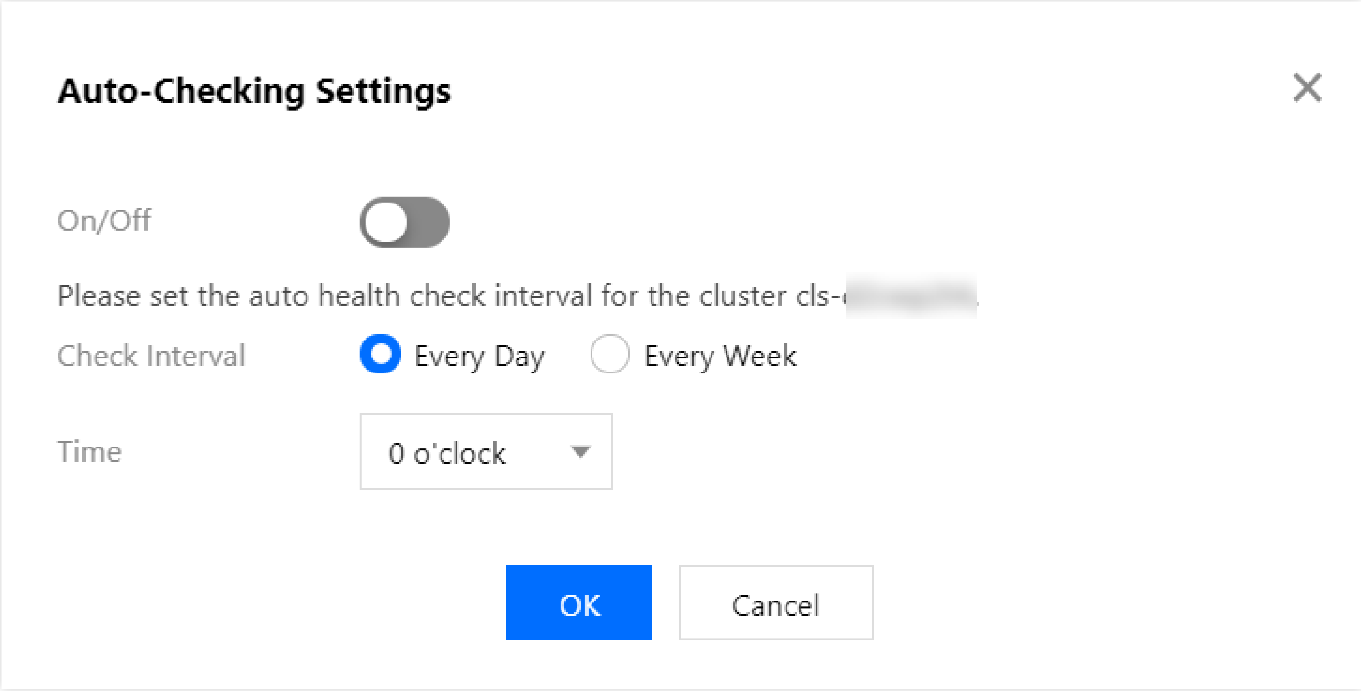


Health check Guangzhou

Batch check Enter the cluster na

	ID/Name	Check Progress	Check Result	Last checked	Auto Check	Operation
□	cls-[redacted] [redacted]	Not checked	-	-	Not enabled	Check Now Auto Check

In the **Auto-Checking Settings** window that appears, you can enable/disable auto-check, and set the check period and time based on your needs.



Auto-Checking Settings

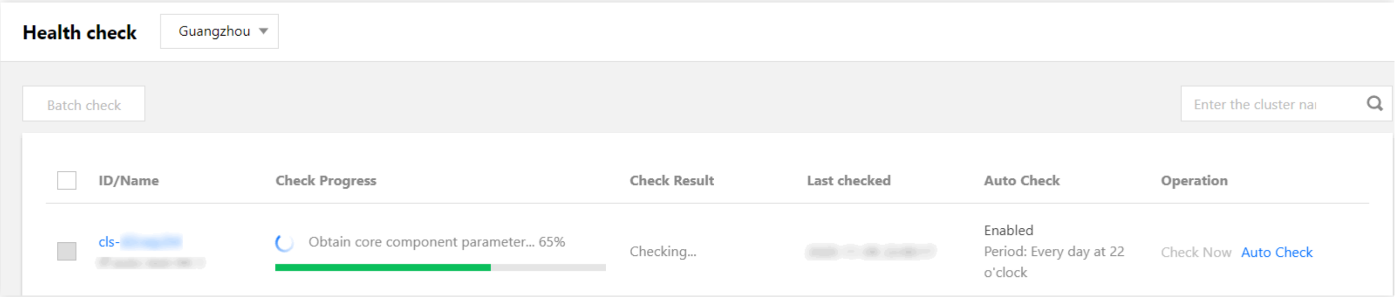
On/Off

Please set the auto health check interval for the cluster cls-[redacted].

Check Interval ● Every Day ○ Every Week

Time 0 o'clock

3. When the health check starts, you can view the check progress.

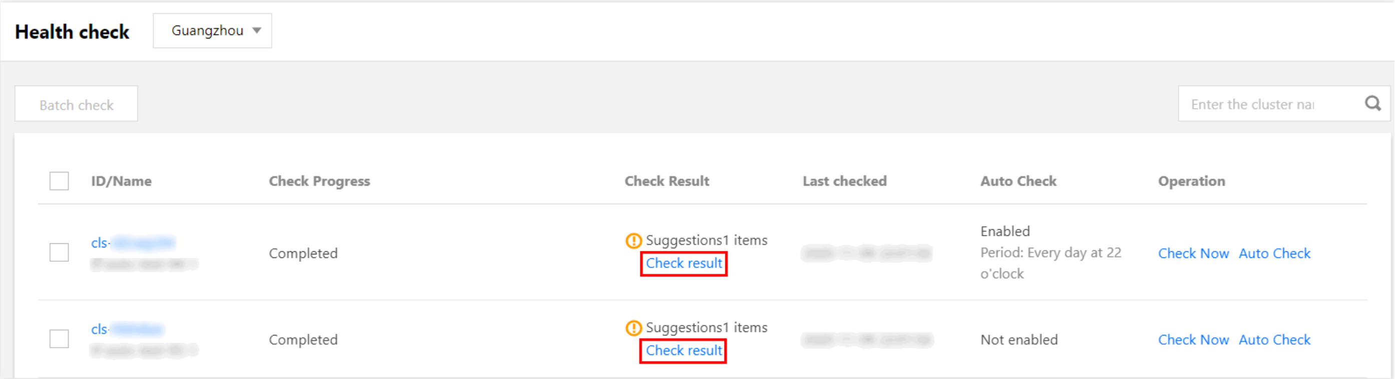


Health check Guangzhou

Batch check | Enter the cluster nai

□	ID/Name	Check Progress	Check Result	Last checked	Auto Check	Operation
□	cls-[redacted] [redacted]	Obtain core component parameter... 65%	Checking...	[redacted]	Enabled Period: Every day at 22 o'clock	Check Now Auto Check

4. After the check is completed, you can click **View Check Result** to view the check report.

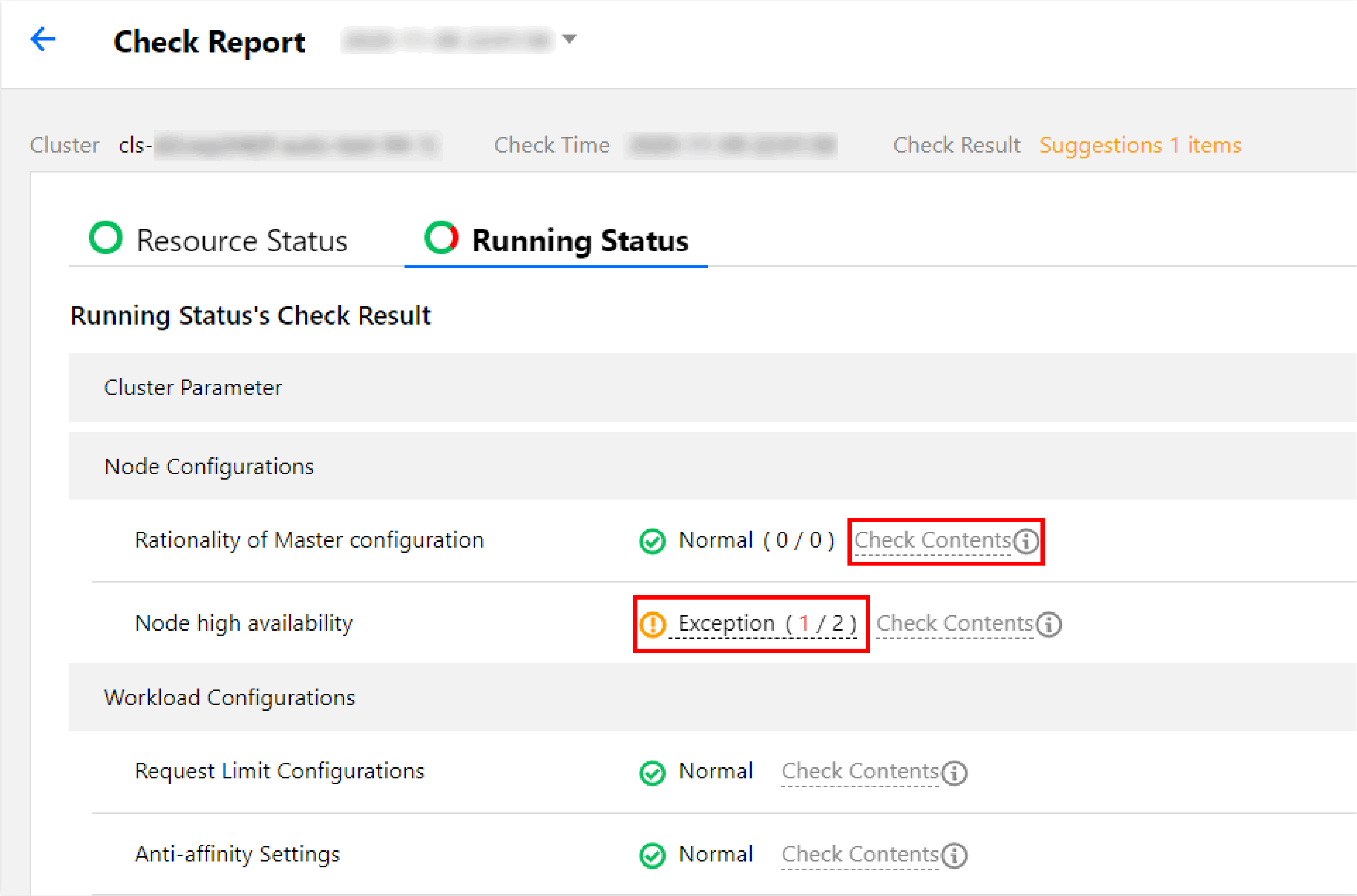


Health check Guangzhou

Batch check | Enter the cluster nai

□	ID/Name	Check Progress	Check Result	Last checked	Auto Check	Operation
□	cls-[redacted] [redacted]	Completed	Suggestions1 items Check result	[redacted]	Enabled Period: Every day at 22 o'clock	Check Now Auto Check
□	cls-[redacted] [redacted]	Completed	Suggestions1 items Check result	[redacted]	Not enabled	Check Now Auto Check

On the check report page, click **Resource Status** and **Running Status** tab to view resource statuses and exceptions respectively. Click **Check Contents** to view the detailed check contents. Click **Exceptions** to view the exception severity, descriptions, causes, impacts, and rectification suggestions.



Check Report [redacted]

Cluster cls-[redacted] Check Time [redacted] Check Result Suggestions 1 items

Running Status's Check Result

Cluster Parameter

Node Configurations

Rationality of Master configuration	✓ Normal (0 / 0)	Check Contents ⓘ
Node high availability	! Exception (1 / 2)	Check Contents ⓘ

Workload Configurations

Request Limit Configurations	✓ Normal	Check Contents ⓘ
Anti-affinity Settings	✓ Normal	Check Contents ⓘ

Monitoring and Alarms

Overview of Monitoring and Alarms

Last updated：2019-10-23 11:18:09

Overview

A healthy monitoring environment ensures the high reliability, high availability, and high performance of Tencent Cloud Tencent Kubernetes Engine (TKE). You can collect monitoring data of different dimensions for different resources, making it easy to understand every resource's usage information and locate errors quickly.
Tencent Cloud TKE features collection and display of monitoring data at 5 levels: cluster, node, workload, pod, and container.
The collection of monitoring data allows you to establish normal standards for cluster performance. By testing performance of a container cluster at different times and under different load conditions, you can better understand the normal performance of a container cluster or service. This allows you to quickly determine if the running status is exceptional based on the current monitoring data, and to find solutions promptly. For example, you can monitor a service's CPU utilization, memory usage, or disk I/O.

Monitoring

For more information on TKE monitoring, please see Viewing Monitoring Data.
For more information on the currently supported monitoring metrics, please see List of Monitoring and Alarm Metrics.

Alarms

We recommend that you set the necessary alarms on all production clusters in order to detect TKE exceptions promptly and ensure the stability and reliability of your business. For more information on setting alarms, please see Setting Alarms.
For more information on the currently supported alarm metrics, please see List of Monitoring and Alarm Metrics.

> TKE monitoring and alarm features primarily cover the core metrics and events of Kubernetes objects. To ensure more fine-grained metric coverage, please use these features in combination with the basic resource monitoring provided by Cloud Monitoring, (e.g. CVM, Cloud Block Storage, Load Balancer, etc.).

Viewing Monitoring Data

Last updated：2023-02-23 18:34:01

Overview

Tencent Cloud TKE provides basic monitoring features to all clusters by default. Use the steps below to view TKE monitoring data.

- Cluster Metrics
- Node Metrics
- Pod Metrics on the Node
- Viewing Workload Metrics
- Viewing Pod Metrics in a Workload
- Viewing Container Metrics in a Pod

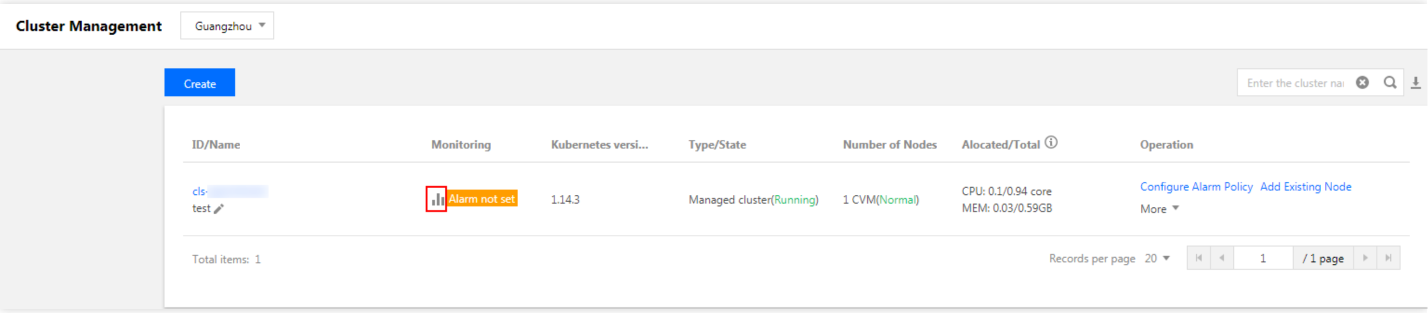
Prerequisites

Log in to TKE console and open the **cluster management** page.

Directions

Viewing Cluster Metrics

Click 📊 in the row of the cluster for which you want to view the monitoring data. You will see that cluster’s monitoring information as shown in the figure below:



ID/Name	Monitoring	Kubernetes versi...	Type/State	Number of Nodes	Alocated/Total	Operation
cls-[redacted] test	Alarm not set	1.14.3	Managed cluster(Running)	1 CVM(Normal)	CPU: 0.1/0.94 core MEM: 0.03/0.59GB	Configure Alarm Policy Add Existing Node More

Total items: 1

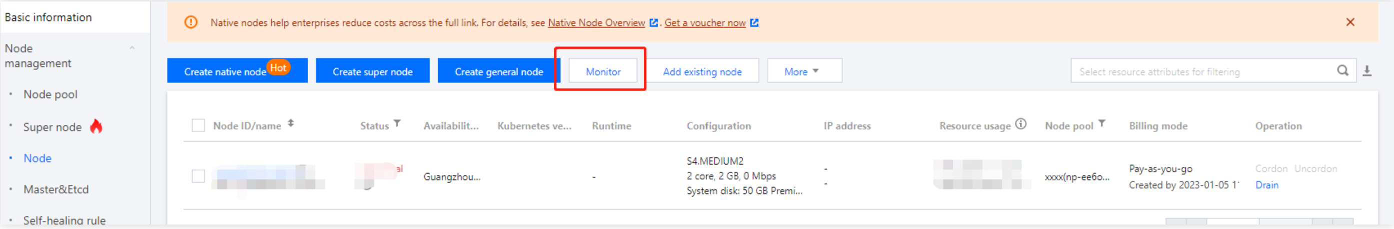
Records per page 20 1 / 1 page

Viewing Node Metrics

Use the steps below to view monitoring information for nodes and Master & Etcd nodes.

1. Click the ID/name of the target cluster to go to the cluster management page.
2. Expand **Node Management** to view monitoring information for ordinary nodes and Master & Etcd nodes.

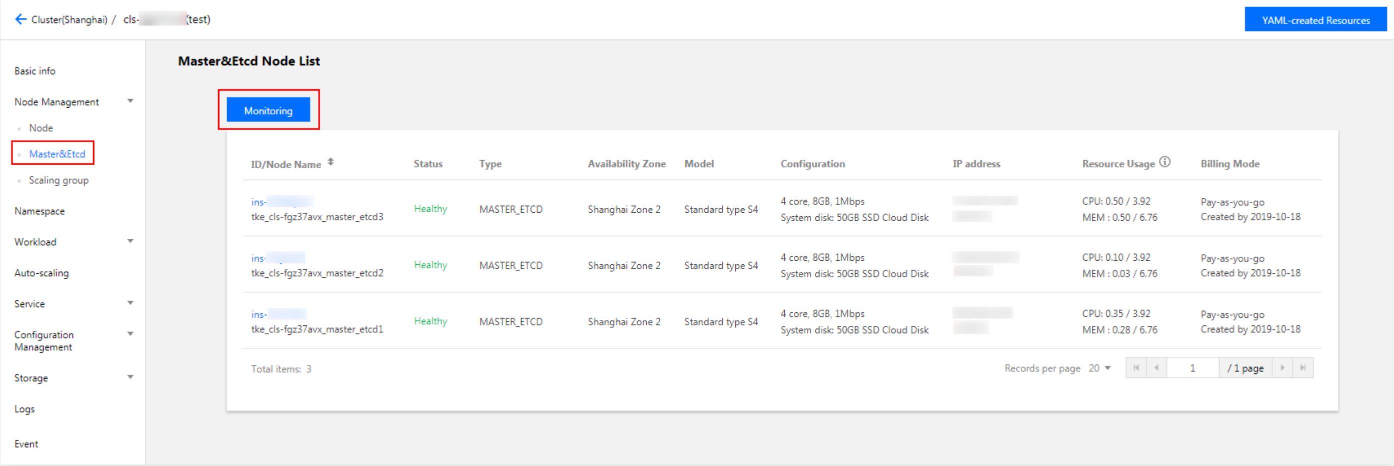
- Select **Node** > **Monitor** to go to the **Node Monitoring** page and view the monitoring information. See the figure below:



Native nodes help enterprises reduce costs across the full link. For details, see Native Node Overview. Get a voucher now

Node ID/name	Status	Availabilit...	Kubernetes ve...	Runtime	Configuration	IP address	Resource usage	Node pool	Billing mode	Operation
[redacted]	[redacted]	Guangzhou...		-	S4.MEDIUM2 2 core, 2 GB, 0 Mbps System disk: 50 GB Premi...	- -	[redacted]	xxxx(np-ee6o...	Pay-as-you-go Created by 2023-01-05 1	Cordon Uncordon Drain

- Select **Master & Etcd** > **Monitoring** to go to the **Master & Etcd Monitoring** page and view the monitoring information. See the figure below:



Master&Etcd Node List

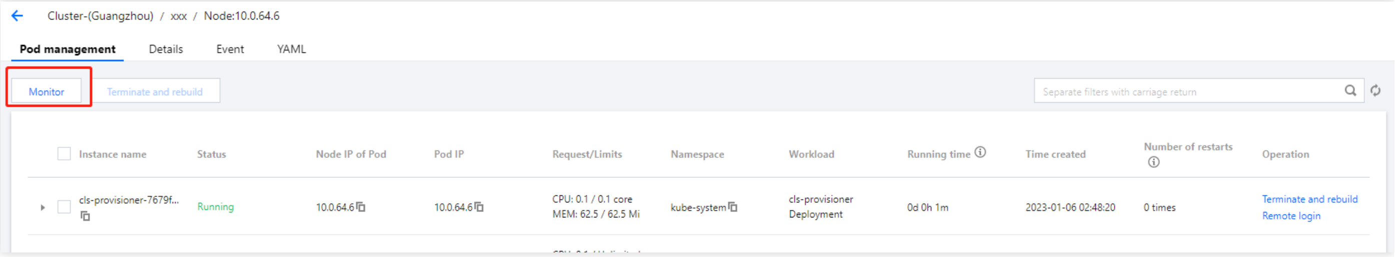
ID/Node Name	Status	Type	Availability Zone	Model	Configuration	IP address	Resource Usage	Billing Mode
ins-[redacted] tke_cls-fgz37avx_master_etcd3	Healthy	MASTER_ETCD	Shanghai Zone 2	Standard type S4	4 core, 8GB, 1Mbps System disk: 50GB SSD Cloud Disk	[redacted]	CPU: 0.50 / 3.92 MEM : 0.50 / 6.76	Pay-as-you-go Created by 2019-10-18
ins-[redacted] tke_cls-fgz37avx_master_etcd2	Healthy	MASTER_ETCD	Shanghai Zone 2	Standard type S4	4 core, 8GB, 1Mbps System disk: 50GB SSD Cloud Disk	[redacted]	CPU: 0.10 / 3.92 MEM : 0.03 / 6.76	Pay-as-you-go Created by 2019-10-18
ins-[redacted] tke_cls-fgz37avx_master_etcd1	Healthy	MASTER_ETCD	Shanghai Zone 2	Standard type S4	4 core, 8GB, 1Mbps System disk: 50GB SSD Cloud Disk	[redacted]	CPU: 0.35 / 3.92 MEM : 0.28 / 6.76	Pay-as-you-go Created by 2019-10-18

Total items: 3

Records per page 20 ▾ 1 / 1 page

Viewing Metrics of Pods in a Node

1. Click the ID/name of the target cluster to go to the cluster management page.
2. Select **Node Management** > **Node** to go to the node list page.
3. Click a node name, click the **Pod Management** tab, and click **Monitor** to view the curve chart of the monitored metrics for the pods on that node. See the figure below:



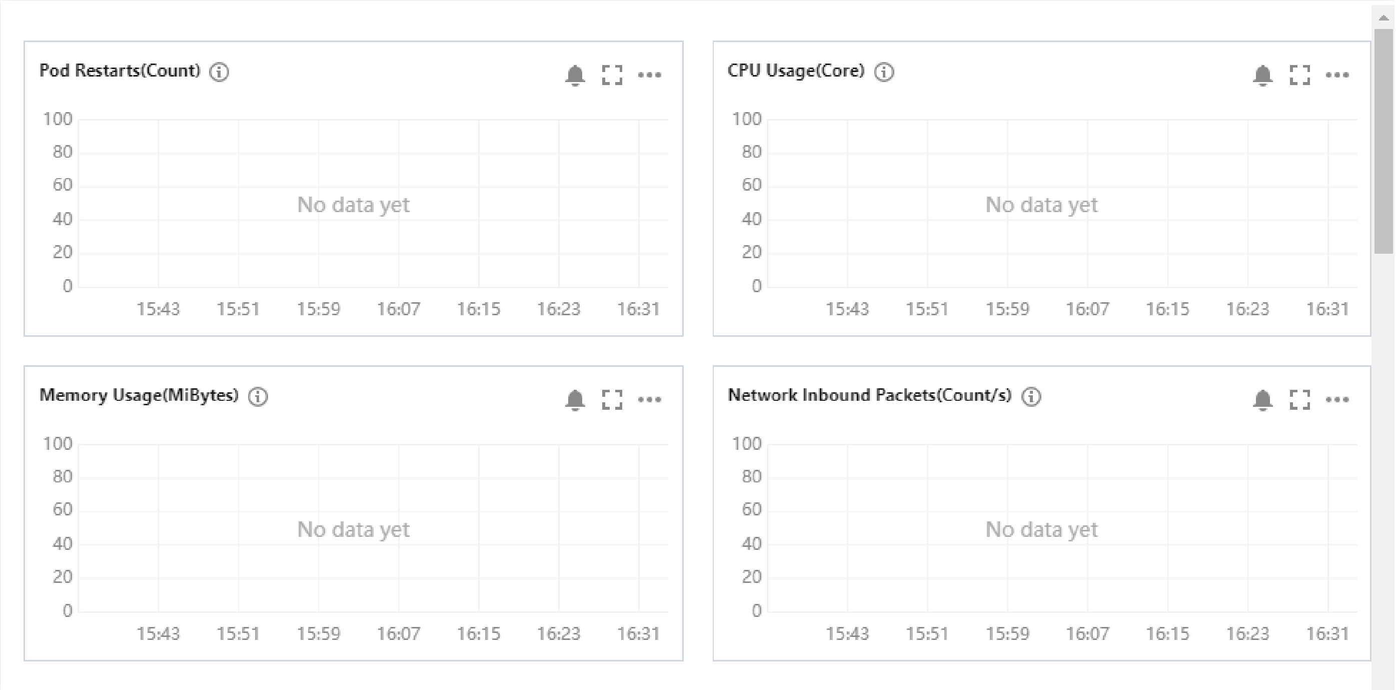
Cluster-(Guangzhou) / xxx / Node:10.0.64.6

Pod management | Details | Event | YAML

Monitor | Terminate and rebuild

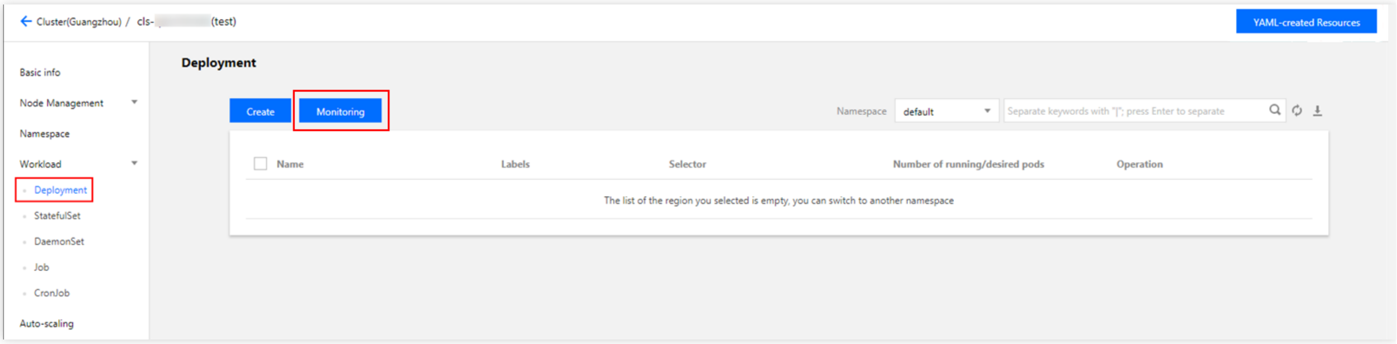
Separate filters with carriage return

☐ Instance name	Status	Node IP of Pod	Pod IP	Request/Limits	Namespace	Workload	Running time	Time created	Number of restarts	Operation
☐ cls-provisioner-7679f...	Running	10.0.64.6	10.0.64.6	CPU: 0.1 / 0.1 core MEM: 62.5 / 62.5 Mi	kube-system	cls-provisioner Deployment	0d 0h 1m	2023-01-06 02:48:20	0 times	Terminate and rebuild Remote login



Viewing Workload Metrics

1. Click the ID/name of the target cluster to go to the cluster management page.
2. Select **Workload** > Any workload type. In this example, **Deployment** is selected.
3. Click **Monitoring** to view the workload monitoring information.



Cluster(Guangzhou) / cls-[redacted](test)

YAML-created Resources

Basic info
Node Management
Namespace
Workload
- Deployment
- StatefulSet
- DaemonSet
- Job
- CronJob

Auto-scaling

Deployment

Create | Monitoring

Namespace: default

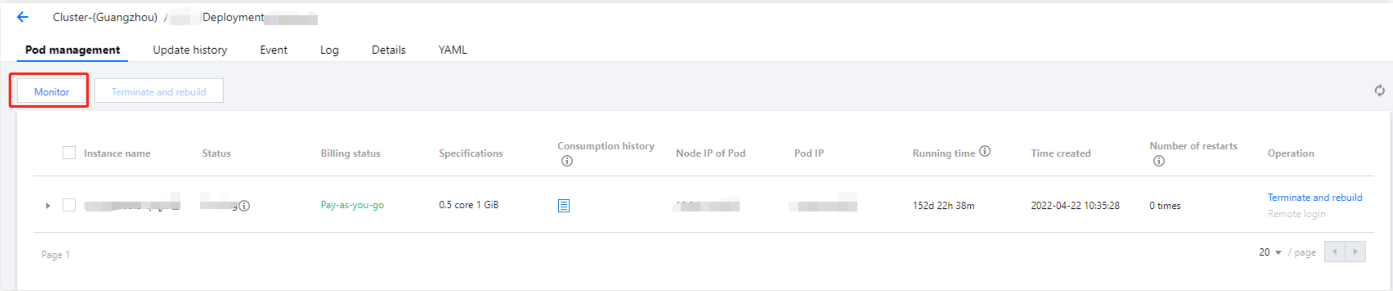
Separate keywords with "|"; press Enter to separate

☐ Name	Labels	Selector	Number of running/desired pods	Operation

The list of the region you selected is empty, you can switch to another namespace

Viewing Metrics of Pods in a Workload

1. Select the ID/name of the cluster you want to view to go to the cluster management page.
2. Select **Workload** > Any workload type. In this example, **Deployment** is selected.
3. Click a workload name. On the **Pod management** tab of the workload, click **Monitor** to view the comparison charts of the monitored metrics of all Pods of the workload.



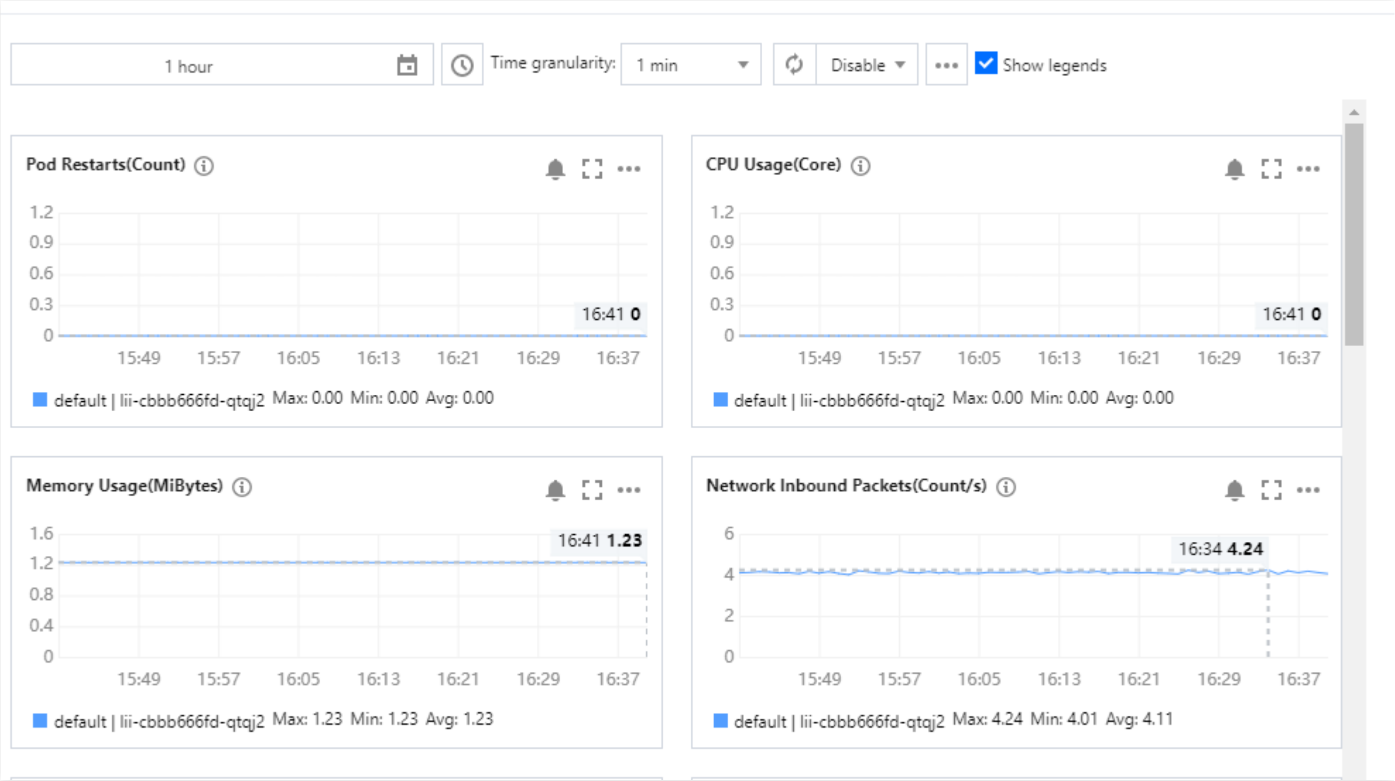
	Instance name	Status	Billing status	Specifications	Consumption history ⓘ	Node IP of Pod	Pod IP	Running time ⓘ	Time created	Number of restarts ⓘ	Operation
▸ □	[redacted]	[redacted] ⓘ	Pay-as-you-go	0.5 core 1 GiB	▤	[redacted]	[redacted]	152d 22h 38m	2022-04-22 10:35:28	0 times	Terminate and rebuild Remote login

Page 1 20 ▾ / page

Viewing Metrics of Containers in a Pod

1. Click the ID/name of the target cluster to go to the cluster management page.
2. Select **Workload** > Any workload type. In this example, **Deployment** is selected.
3. Click a workload name. On the **Pod management** tab of the workload, click ▶ on the left of the instance name to view the container information of the Pod.

4. Click 📊 to view the comparison charts of the monitored metrics of the containers in the Pod.



1 hour | Time granularity: 1 min | Disable | ☑ Show legends

List of Monitoring and Alarm Metrics

Last updated：2021-03-25 16:19:47

Monitoring

TKE currently provides monitoring metrics of the following dimensions. All metrics are **average values** within the granularity.

Monitoring Metrics for Clusters

Monitoring Metric	Unit	Description
CPU Utilization	%	CPU utilization rate of entire cluster
MEM Utilization	%	Memory utilization rate of entire cluster

Monitoring Metrics for Master & Etcd and Ordinary Nodes

Monitoring Metric	Unit	Description
Re-startup of Pods	restarts	Sum of the number of restarts of all pods on the node
Exception	-	Node status: normal or exceptional
CPU Utilization	%	CPU usage of all pods on the node to the total CPU of the node
MEM Utilization	%	Memory usage of all pods on the node to the total memory of the node
Private bandwidth in	bps	Total private network inbound bandwidth of all pods on the node
Private bandwidth out	bps	Total for private network outbound bandwidth of all pods on the node
Public bandwidth in	bps	Total public network inbound bandwidth of all pods on the node
Public bandwidth out	bps	Total public network outbound bandwidth of all pods on the node
TCP Connections Count	connections	Number of TCP connections maintained on the node

For more information on monitoring metrics for cluster nodes, please see Get Monitoring Statistics.

For more information on monitoring metrics for cluster node data disks, please see Monitoring Cloud Disks.

Monitoring Metrics for Workloads

Monitoring Metric	Unit	Description
Re-startup of Pods	restarts	Total for the number of restarts of all pods in the workload
CPU Usage	cores	CPU usage of all pods in the workload
CPU Utilization (% cluster)	%	CPU usage of all pods in the workload to the total CPU of the cluster
MEM Usage	B	Memory usage of all pods in the workload
MEM Utilization (% cluster)	%	Memory usage of all pods in the workload to the total memory of the cluster
Network Inbound Bandwidth	bps	Total inbound bandwidth of all pods in the workload
Network Outbound Bandwidth	bps	Total for outbound bandwidth of all pods in the workload
Network Inbound Traffic	B	Total inbound traffic of all pods in the workload
Network Traffic Out	B	Total outbound traffic of all pods in the workload
Network Inbound Traffic	packets/sec	Total inbound packets of all pods in the workload
Network Outbound Traffic	packets/sec	Total outbound packets of all pods in the workload

If the workload provides services outside the cluster, please see Obtaining Monitoring Data for more information on network monitoring metrics for bound services.

Monitoring Metrics for Pods

Monitoring Metric	Unit	Description
Exception	-	Pod status: normal or exceptional
CPU Usage	cores	CPU usage of the pod
CPU Utilization (% node)	%	CPU usage of the pod to the total CPU of the node
CPU Utilization (% Request)	%	CPU usage of the pod to the Request valude
CPU Utilization (% of Limit)	%	CPU usage of the pod to the Limit value

MEM Usage	B	Memory usage of the pod, including cache
MEM Usage (exclude cache)	B	Actual memory usage (not including cache) of all containers in the pod
MEM Utilization (% node)	%	Memory usage of the pod to the total memory of the node
MEM Utilization (% node, exclude cache)	%	Actual memory usage (not including cache) of all containers in the pod to the total memory of the node
MEM Utilization (% Request)	%	Memory usage of the pod to the Request value
MEM Utilization (% Request, exclude cache)	%	Actual memory usage (not including cache) of all containers in the pod to the Request value
MEM Utilization (% of Limit)	%	Memory usage of the pod to the Limit value
MEM Utilization (% limit, exclude cache)	%	Actual memory usage (not including cache) of all containers in the pod to the Limit value
Network Inbound Bandwidth	bps	Total inbound bandwidth of the pod
Network Outbound Bandwidth	bps	Total outbound bandwidth of the pod
Network Inbound Traffic	B	Total inbound traffic of the pod
Network Traffic Out	B	Total outbound traffic of the pod
Network Inbound Traffic	packets/sec	Total inbound packets of the pod
Network Outbound Traffic	packets/sec	Total outbound packets of the pod

Monitoring Metrics for Containers

Monitoring Metric	Unit	Description
CPU Usage	cores	CPU usage of container
CPU Utilization (% node)	%	CPU usage of the container to the total CPU of the node
CPU Utilization (% Request)	%	CPU usage of the container to the Request value
CPU Utilization (% Limit)	%	CPU usage of the container to the Limit value
MEM Usage	B	Memory usage of the container, including cache
MEM Usage (exclude cache)	B	Actual memory usage of the container (not including cache)

MEM Utilization (% node)	%	Memory usage of the container to the total memory of the node
MEM Utilization (% node, exclude cache)	%	Actual memory usage (not including cache) of the container to the total memory of the node
MEM Utilization (% request)	%	Memory usage of the container to the Request value
MEM Utilization (% Request, excl. cache)	%	Actual memory usage (not including cache) of the container to the Request value
MEM Utilization (% of Limit)	%	Memory usage of the container to the Limit value
MEM Utilization (% limit, exclude cache)	%	Actual memory usage (not including cache) of the container to the Limit value
Block device read bandwidth	B/sec	Throughput of the container to read data from disk
Block device write bandwidth	B/sec	Throughput of the container to write data to disk
Read IOPS of Block Device	operations/sec	Number of times the container read from disk
Write IOPS of Block Device	operations/sec	Number of times the container wrote to disk

Alarms

TKE currently provides alarm metrics of the following dimensions. All metrics are **average values** within the statistical period.

Alarm Metrics for Clusters

Monitoring Metric	Unit	Description
CPU Utilization	%	CPU utilization rate of entire cluster
MEM Utilization	%	Memory utilization rate of entire cluster
CPU Allocation	%	Ratio of the sum of the set CPU Requests from all containers in the cluster to the cluster’s total allocable CPU resources
MEM Allocation	%	Ratio of the sum of the set Requests from all containers in the cluster to the cluster’s total allocable memory resources

Apiserver Normal	-	Apiserver status. By default, alarms when status value is `False` . Only self-deployed clusters support this metric.
Etcd Normal	-	Etcd status. By default, alarms when status value is `False` . Only self-deployed clusters support this metric.
Scheduler Normal	-	Scheduler status. By default, alarms when status value is `False` . Only self-deployed clusters support this metric.
Control Manager Normal	-	Control Manager status. By default, alarms when status value is `False` . Only self-deployed clusters support this metric.

Alarm Metrics for Nodes

Monitoring Metric	Unit	Description
CPU Utilization	%	CPU usage of all pods on the node to the total CPU of the node
MEM Utilization	%	Memory usage of all pods on the node to the total memory of the node
Re-startup of Pods on This Node	Times	Total number of restarts of all pods on the node
Node Ready	-	Node status. By default, alarms when status value is `False` .

For more information on alarm metrics for cluster nodes, please see Get Monitoring Statistics and Create Alarm.

For more information on alarm metrics for cluster node data disks, please see Monitoring Cloud Disks and Create Alarm.

Alarm Metrics for Pods

Monitoring Metric	Unit	Description
CPU Utilization (% node)	%	CPU usage of the pod to the total CPU of the node l
MEM Utilization (% node)	%	Memory usage of the pod to the total memory of the node
Actual MEM Utilization (% node, exclude cache)	%	Actual memory usage (exclude cache) of all containers in the pod to the total memory of the node
CPU Utilization (% limit)	%	CPU usage of the pod to the Limit value
MEM Utilization (% of Limit)	%	Memory usage of the pod to the Limit value

Actual MEM Utilization (% of Limit, exclude cache)	%	Actual memory usage of the pod (exclude cache) to the Limit value
Re-startup of Pods	restarts	Number of pod restarts
Pod Ready	-	Pod status. By default, alarms when status value is `False` .
CPU Usage	cores	CPU usage of the pod
MEM Usage	MB	Memory usage of the pod, including cache
Actual MEM Usage (exclude cache)	MB	Actual memory usage of all containers in the pod, excluding cache

Log Management

Collect container logs to CLS

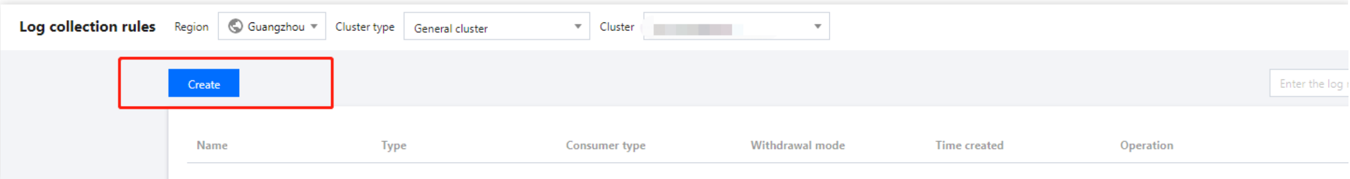
Last updated：2023-05-25 10:41:57

This document introduces how to configure log collection rules in the TKE console and ship logs to Tencent Cloud Log Service (CLS).

Directions

Creating log collection rules

1. Log in to the TKE console and choose **Log Management** > **Log Rules** in the left sidebar.
2. At the top of the **Log Rules** page, select a region and a cluster for which you want to configure log collection rules and click **Create**, as shown below:



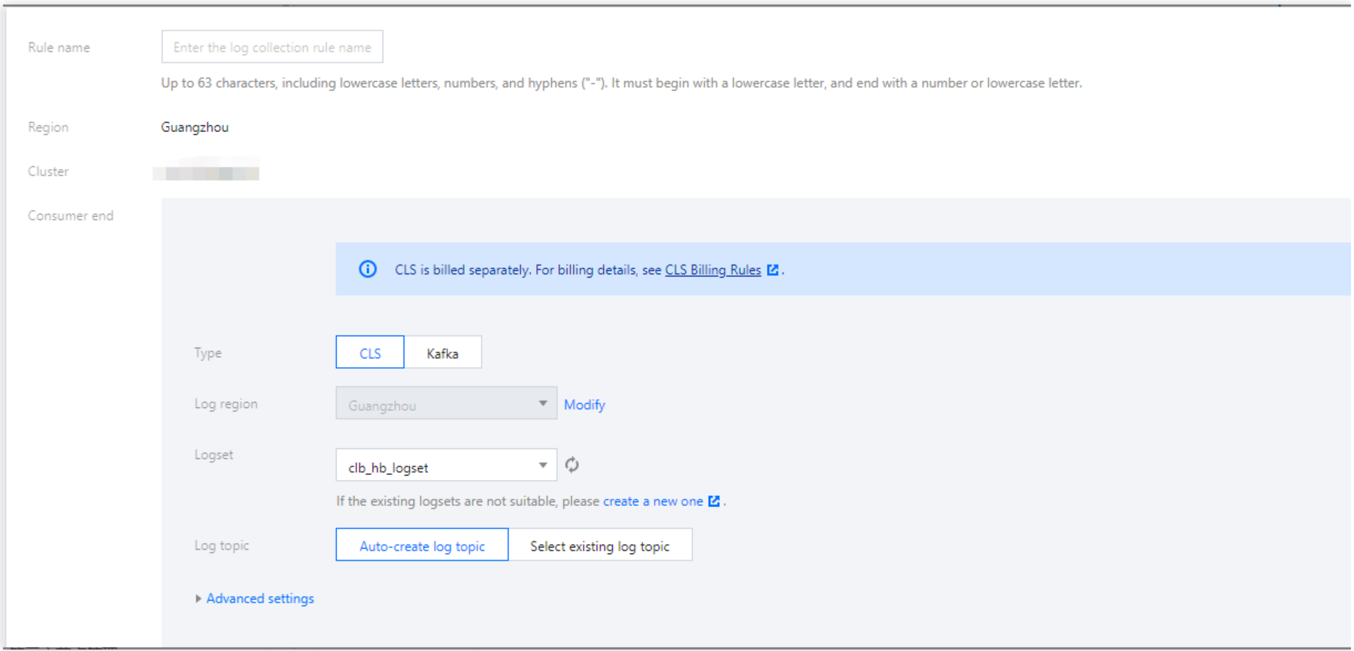
Log collection rules Region Guangzhou Cluster type General cluster Cluster [redacted]

Create

Enter the log

Name	Type	Consumer type	Withdrawal mode	Time created	Operation

3. On the **Create Log Collecting Policy** page, configure the consumer of logs: Set **Type** to **CLS** in the **Consumer end** area, as shown in the figure below:



Rule name: Enter the log collection rule name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Region: Guangzhou

Cluster: [redacted]

Consumer end:

CLS is billed separately. For billing details, see CLS Billing Rules.

Type: CLS | Kafka

Log region: Guangzhou Modify

Logset: clb_hb_logset

If the existing logsets are not suitable, please create a new one.

Log topic: Auto-create log topic | Select existing log topic

▸ Advanced settings

Rule name: You can customize the log collection rule name.

Log region: CLS supports cross-region log shipping. You can click **Modify** to select the destination region for log shipping.

Logset: Created logsets are displayed by log region. If the existing logsets are not suitable, you can create a new one in the CLS console. For operation details, see Creating logset.

Log topic: Select the corresponding log topic under the logset. Two modes are supported: **Auto-create log topic** and **Select existing log topic**.

Advanced settings:

Default metadata: CLS sets the metadata `pod_name` , `namespace` , and `container_name` as indexes for log searches by default.

Custom metadata: You can customize metadata and log indexes.

Note

CLS does not support cross-region log shipping (from the Chinese mainland to outside the Chinese mainland and vice versa). For regions where CLS is not activated, logs can be shipped only to the nearest regions. For example, the container logs collected from a Shenzhen cluster can be shipped only to Guangzhou, and the containers logs collected from a Tianjin cluster can be shipped only to Beijing. You can find more information in the console.

Currently, a log topic supports the collection configuration of only one type of logs. That is, the log, audit, and event types cannot use the same topic. If they use the same topic, logs will be overwritten. Ensure that the selected log topic is not occupied by other collection configurations. A logset can contain up to 500 log topics.

Custom metadata and metadata indexes cannot be modified once created. You can go to the CLS console to modify the configuration.

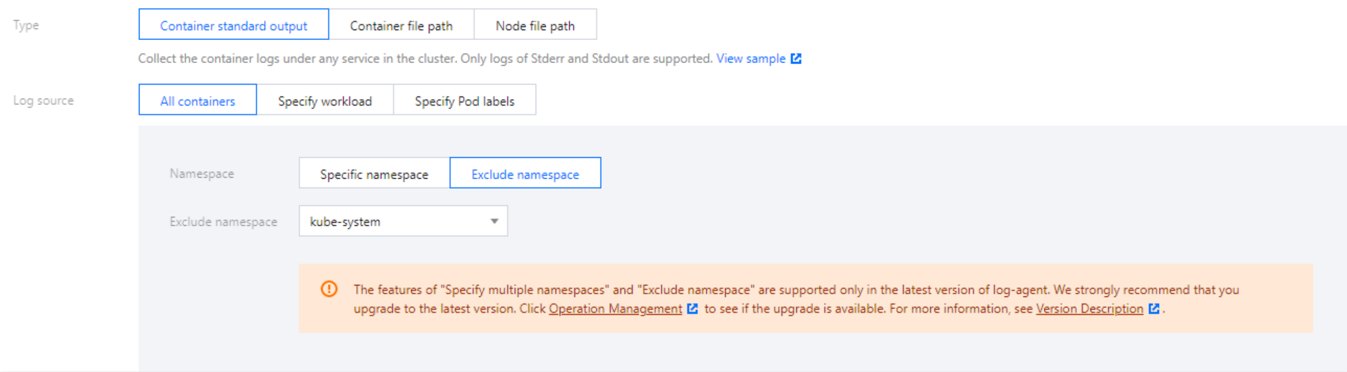
4. Select a collection type and configure a log source. Supported collection types are **Container standard output**, **Container file path**, and **Node file path**.

Container Standard Output Logs

Container File Logs

Node File Logs

Log source supports **All containers**, **Specify workload**, and **Specify Pod Labels**, as shown below:



Type: Container standard output | Container file path | Node file path

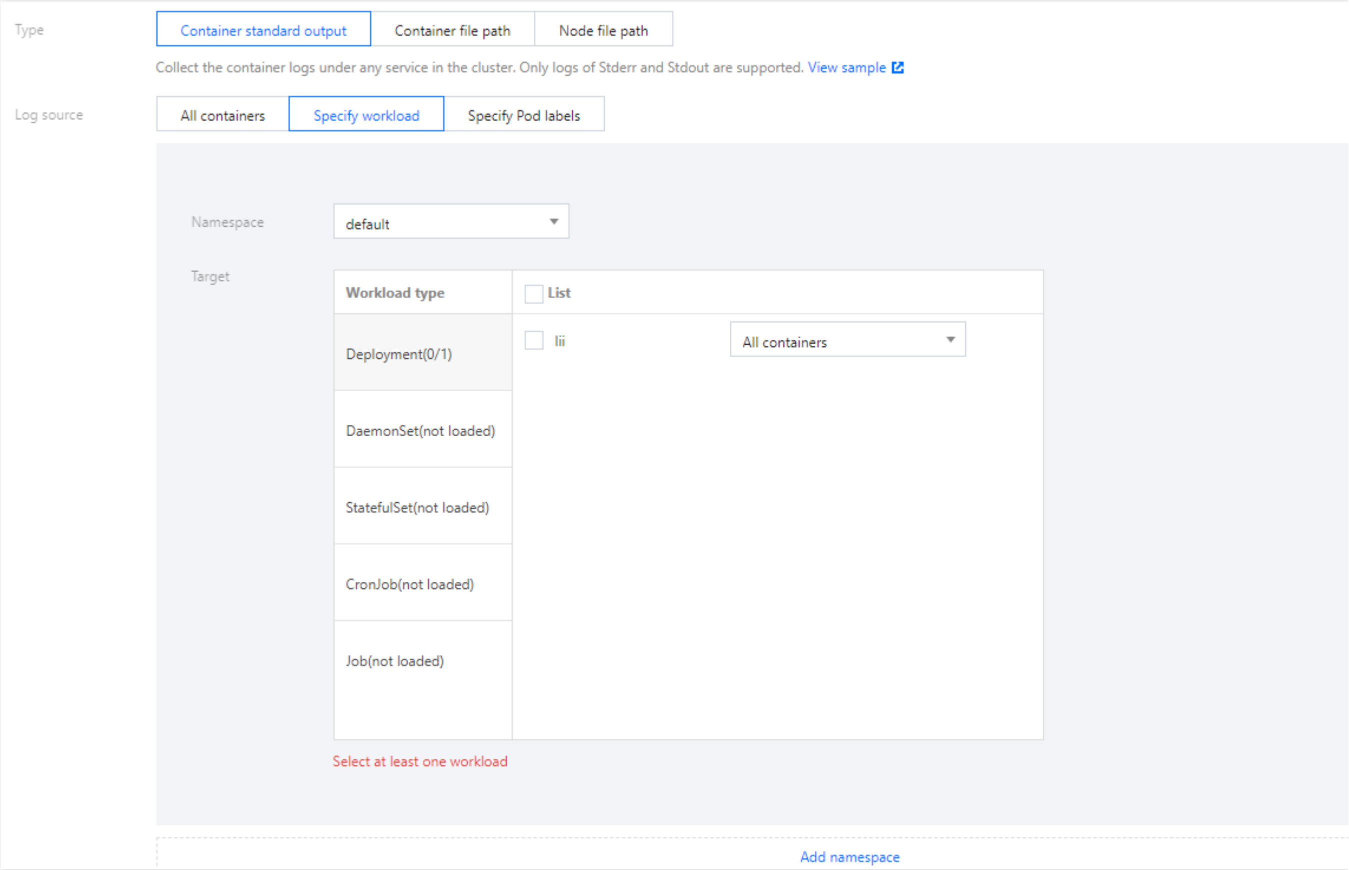
Collect the container logs under any service in the cluster. Only logs of Stderr and Stdout are supported. View sample

Log source: All containers | Specify workload | Specify Pod labels

Namespace: Specific namespace | Exclude namespace

Exclude namespace: kube-system

The features of "Specify multiple namespaces" and "Exclude namespace" are supported only in the latest version of log-agent. We strongly recommend that you upgrade to the latest version. Click Operation Management to see if the upgrade is available. For more information, see Version Description.



Type: Container standard output | Container file path | Node file path

Collect the container logs under any service in the cluster. Only logs of Stderr and Stdout are supported. View sample

Log source: All containers | Specify workload | Specify Pod labels

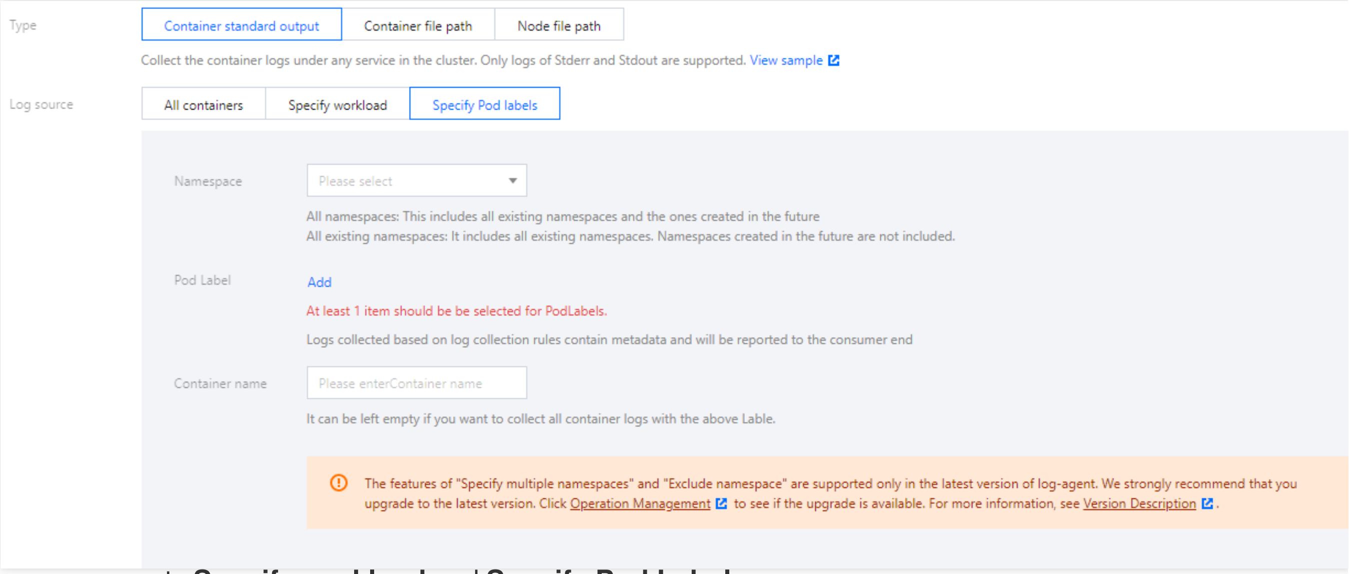
Namespace: default

Target

Workload type	□ List	
Deployment(0/1)	□ lii	All containers
DaemonSet(not loaded)		
StatefulSet(not loaded)		
CronJob(not loaded)		
Job(not loaded)		

Select at least one workload

Add namespace



Type: Container standard output | Container file path | Node file path

Collect the container logs under any service in the cluster. Only logs of Stderr and Stdout are supported. View sample

Log source: All containers | Specify workload | Specify Pod labels

Namespace: Please select

All namespaces: This includes all existing namespaces and the ones created in the future
All existing namespaces: It includes all existing namespaces. Namespaces created in the future are not included.

Pod Label: Add

At least 1 item should be be selected for PodLabels.

Logs collected based on log collection rules contain metadata and will be reported to the consumer end

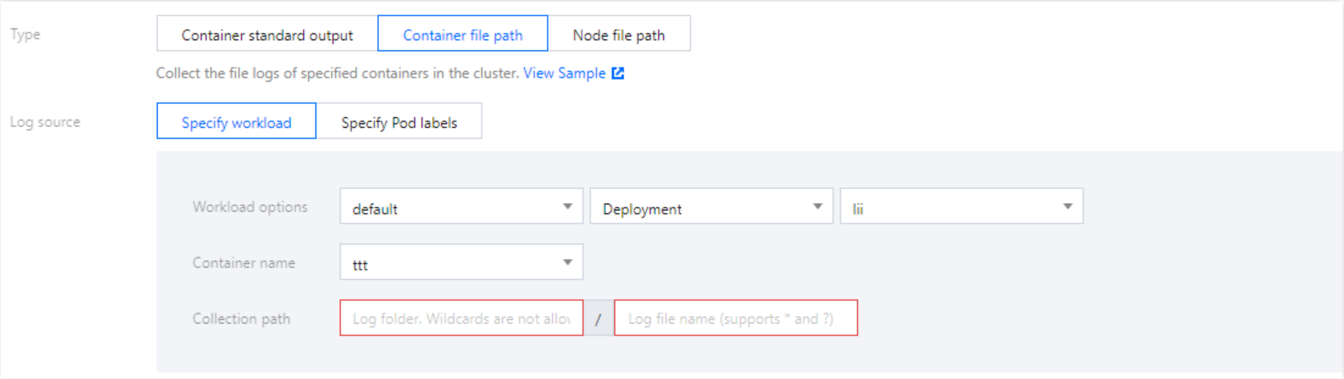
Container name: Please enterContainer name

It can be left empty if you want to collect all container logs with the above Lable.

The features of "Specify multiple namespaces" and "Exclude namespace" are supported only in the latest version of log-agent. We strongly recommend that you upgrade to the latest version. Click Operation Management to see if the upgrade is available. For more information, see Version Description .

Log source supports **Specify workload** and **Specify Pod Labels**.

You can specify a file path or use wildcards for the collection path. For example, when the container file path is `/opt/logs/*.log` , you can specify the collection path as `/opt/logs` and the file name as `*.log` .



Type: Container standard output | Container file path | Node file path

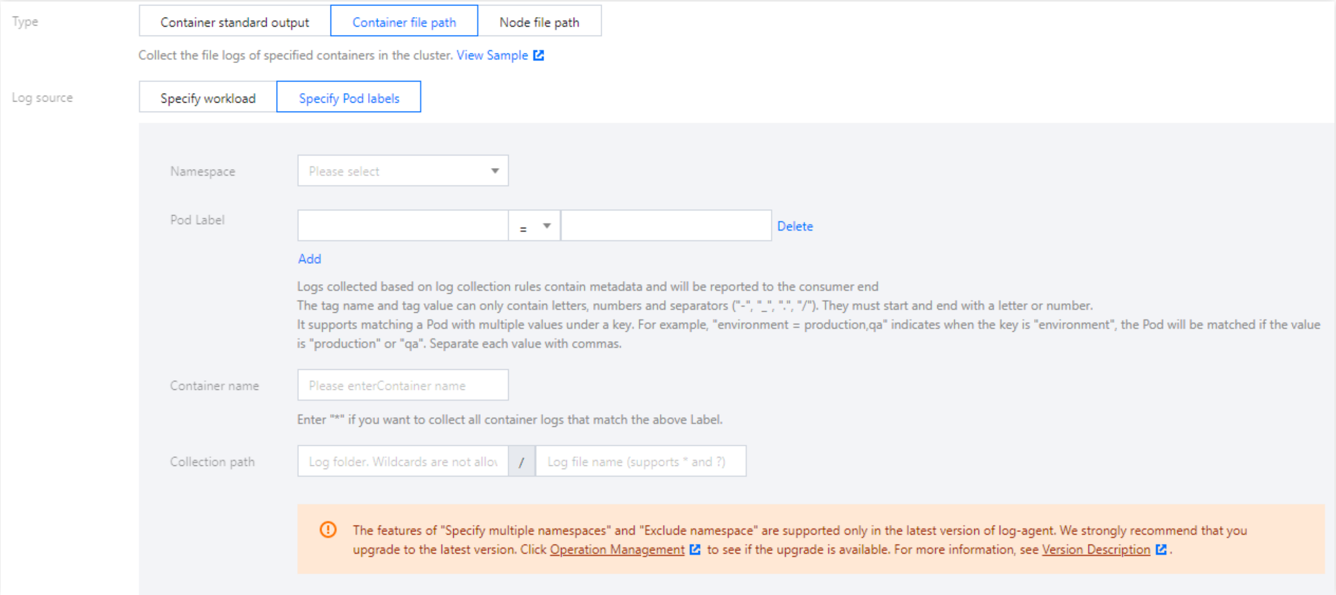
Collect the file logs of specified containers in the cluster. View Sample

Log source: Specify workload | Specify Pod labels

Workload options: default | Deployment | lii

Container name: ttt

Collection path: Log folder. Wildcards are not allov / Log file name (supports * and ?)



Type: Container standard output | Container file path | Node file path

Collect the file logs of specified containers in the cluster. View Sample

Log source: Specify workload | Specify Pod labels

Namespace: Please select

Pod Label: = Delete

Add

Logs collected based on log collection rules contain metadata and will be reported to the consumer end
The tag name and tag value can only contain letters, numbers and separators ("-", "_", ".", "/"). They must start and end with a letter or number.
It supports matching a Pod with multiple values under a key. For example, "environment = production,qa" indicates when the key is "environment", the Pod will be matched if the value is "production" or "qa". Separate each value with commas.

Container name: Please enterContainer name

Enter "*" if you want to collect all container logs that match the above Label.

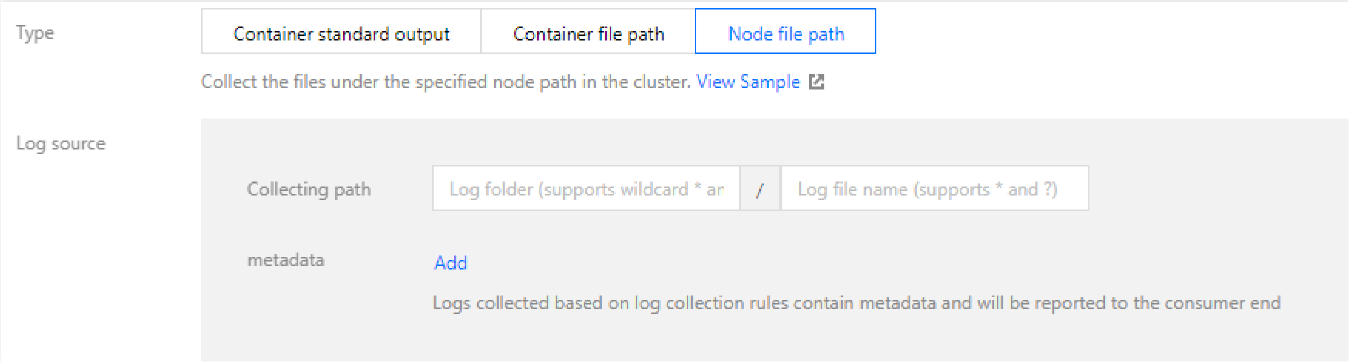
Collection path: Log folder. Wildcards are not allov / Log file name (supports * and ?)

The features of "Specify multiple namespaces" and "Exclude namespace" are supported only in the latest version of log-agent. We strongly recommend that you upgrade to the latest version. Click Operation Management to see if the upgrade is available. For more information, see Version Description .

Note

For **Container file path**, the corresponding path **cannot be a soft link or hard link**. Otherwise, the actual path of the soft link will not exist in the collector's container, resulting in log collection failure.

You can specify a file path or use wildcards. For example, when the container file paths for collection are `/opt/logs/service1/*.log` and `/opt/logs/service2/*.log`, you can specify the folder of the collection path as `/opt/logs/service*` and the file name as `*.log`.
You can attach metadata in the key-value pair format to log records as needed.



Type: Container standard output | Container file path | Node file path

Collect the files under the specified node path in the cluster. View Sample

Log source

Collecting path: Log folder (supports wildcard * ar / Log file name (supports * and ?)

metadata: Add

Logs collected based on log collection rules contain metadata and will be reported to the consumer end

Note:

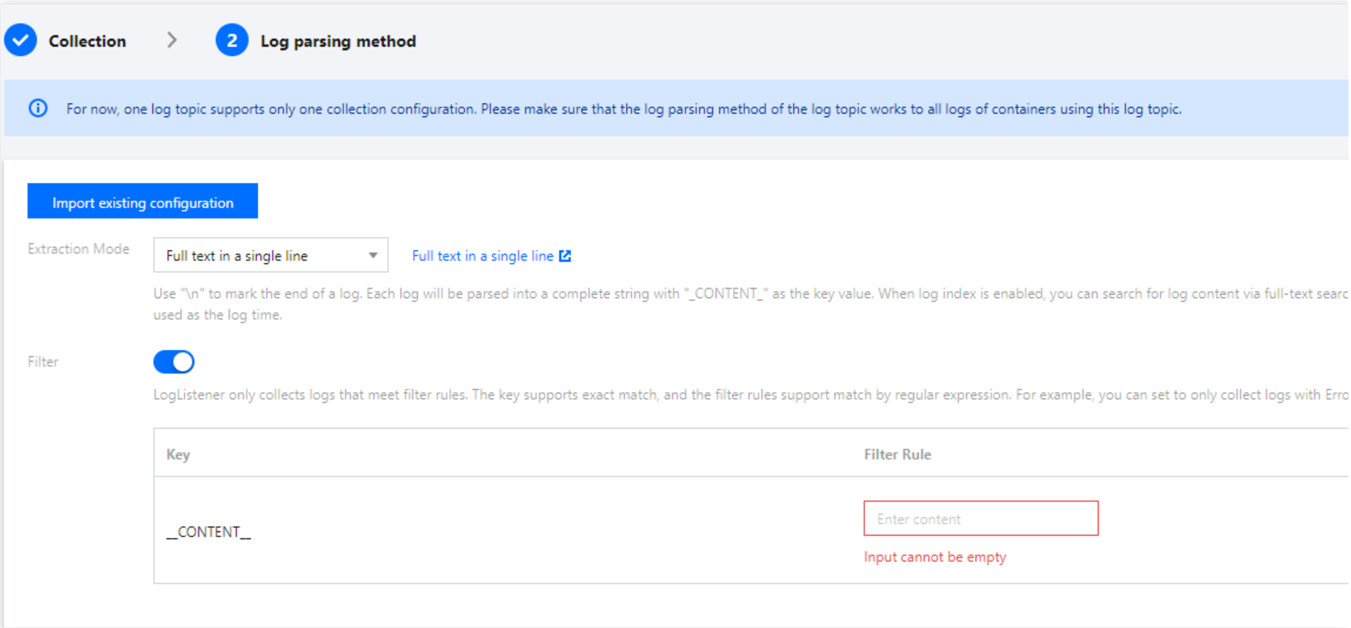
For **Node file path**, the corresponding path **cannot be a soft link or hard link**. Otherwise, the actual path of the soft link will not exist in the collector, resulting in log collection failure.
Each node log file can be collected to only one log topic.

Note

For **Container standard output** and **Container file path** (excluding **Node file path**/not mounted in hostPath), besides the original log content, the metadata related to the container or Kubernetes (such as the ID of the container that generated the logs) will also be reported to the CLS. Therefore, when viewing logs, users can trace the log source or search based on the container identifier or characteristics (such as container name and labels).
The metadata related to the container or Kubernetes is shown in the table below:

Field	Description
container_id	ID of the container to which the log belongs
container_name	Name of the container to which the log belongs
image_name	Image name IP of the container to which the log belongs
namespace	Namespace of the Pod to which the log belongs
pod_uid	UID of the Pod to which the log belongs
pod_name	Name of the Pod to which the log belongs
pod_lable_{label name}	Labels of the Pod to which the log belongs (for example, if a Pod has two labels: `app=nginx` and `env=prod` , the reported log will have two metadata entries attached: `pod_label_app:nginx` and `pod_label_env:prod`)

5. Configure the collection policy as **Full** or **Incremental**.
Full: Collecting logs from the beginning of the log file.
Incremental: Collecting logs 1 MB ahead of the end of the log file. For a log file less than 1 MB in size, incremental collection is equivalent to full collection.
6. Click **Next** and select a log parsing method, as shown below:



For now, one log topic supports only one collection configuration. Please make sure that the log parsing method of the log topic works to all logs of containers using this log topic.

Extraction Mode: Full text in a single line ▾ Full text in a single line

Use "\n" to mark the end of a log. Each log will be parsed into a complete string with "_CONTENT_" as the key value. When log index is enabled, you can search for log content via full-text searc
used as the log time.

Filter: ●

LogListener only collects logs that meet filter rules. The key supports exact match, and the filter rules support match by regular expression. For example, you can set to only collect logs with Erro

Key	Filter Rule
__CONTENT__	Enter content Input cannot be empty

Encoding Mode: Supports **UTF-8** and **GBK**.

Extraction Mode: Supports multiple extraction modes, as described below:

Parsing mode	Description	Reference
Full text in a single line	A log contains only one line of content, and the line break `\\n` marks the end of a log. Each log will be parsed into a complete string with `CONTENT` as the key. When log indexing is enabled, you can search for log content via full-text search. The time attribute of a log is determined by the collection time.	Full Text in a Single Line
Full text in multi lines	A log with full text in multi lines spans multiple lines, and a first-line regular expression is used for matching. When a log in a line matches the preset regular expression, the line is considered as the beginning of a log, and the next matching line will be the end mark of the log. A default key, `CONTENT` , will be set as well. The time attribute of a log is determined by the collection time. The regular expression can be generated automatically.	Full Text in Multi Lines
Single line - full regex	The single-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete log. When configuring the single-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	Full Regular Expression (Single-Line)
Multiple lines - full regex	The multi-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete log that spans multiple lines in a log text file (such as Java program logs) based on a regular expression. When configuring the multi-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding	Full Regular Expression (Multi-Line)

	key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	
JSON	A JSON log automatically extracts the key at the first layer as the field name and the value at the first layer as the field value to implement structured processing of the entire log. Each complete log ends with a line break `\\n` .	JSON Format
Separator	Logs in this format are structured with the specified separator, and each complete log ends with a line break `\\n` . You need to define a unique key for each separate field. Leave the field blank if you don’t want to collect it. At least one field is required.	Separator Format

Filter: LogListener collects only logs that meet the filter rules. **Key** supports exact matching, and **Filter Rule** supports regular expression matching. For example, you can specify to collect only logs where `ErrorCode` is `404` . You can enable the filter feature and configure rules as needed.

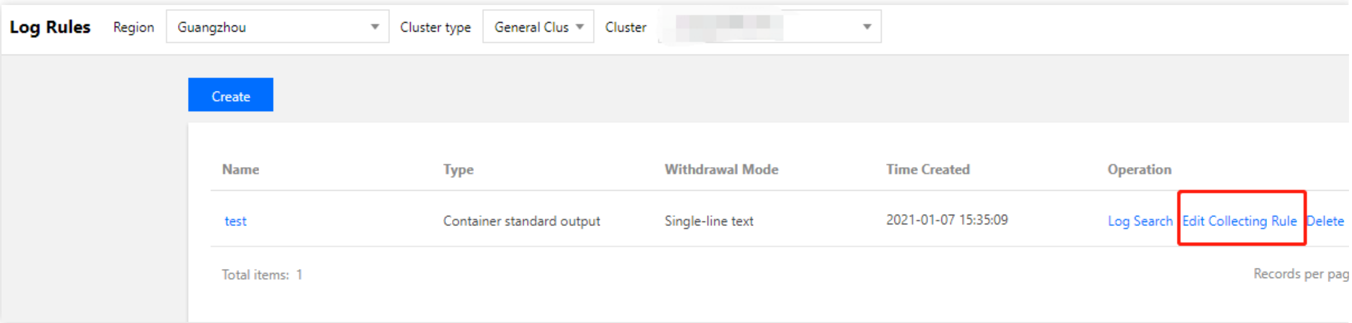
Note

Currently, one log topic supports only one collection configuration. Ensure that all container logs that adopt the log topic can accept the log parsing method that you choose. If you create different collection configurations under the same log topic, the earlier collection configurations will be overwritten.

7. Click **Done**.

Updating the log rules

1. Log in to the TKE console and choose **Log Management** > **Log Rules** in the left sidebar.
2. At the top of the **Log Rules** page, select the region and the cluster where you want to update the log collection rules and click **Edit Collecting Rule** at the right, as shown in the figure below:



Log Rules Region Guangzhou Cluster type General Clus Cluster [redacted]

Name	Type	Withdrawal Mode	Time Created	Operation
test	Container standard output	Single-line text	2021-01-07 15:35:09	Log Search Edit Collecting Rule Delete

Total items: 1

Records per pag

3. Update the configuration as needed and click **Done**.

Note

The logset and log topic of a created rule cannot be modified.

References

Besides using the TKE console, you can also configure log collection by using custom resource definition (CRD). For more information, see Using CRD to Configure Log Collection.

Using CRD to Configure Log Collection

Last updated：2023-05-05 10:38:21

Overview

Besides configuration log collection in the TKE console, you can also configure it by using the Custom Resource Definitions (CRD). CRD supports the collection of container standard outputs, container files, and host files. It also supports multiple log collection formats, and supports shipping logs to different consumers such as CLS and CKafka.

Prerequisites

Activate Log Collection in TKE console.

CRD Overview

Structure overview

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig                    ## Default value
metadata:
  name: test       ## CRD resource name, unique in the cluster
spec:
  clsDetail:                                    ## The configuration for shipping t
    ...
  inputDetail:                  ## Data source configuration for collection
    ...
  kafkaDetail:                                  ## The configuration for shipping t
    ...
```

```
status:                                                 ## CRD resource status
  status: ""
  code: ""                                              ## The error code returned
  reason: ""                                    ## Error cause
```

clsDetail description

Note

The topic cannot be modified once it’s specified.

```
  clsDetail:
    ## If the log topic is created automatically, the names of logset and topic nee
    logsetName: test                       ## CLS logset name. Logset for the name
    topicName: test                        ## CLS log topic name. Log topic for the

    # Select an existing logset and log topic. If the logset is specified but the l
    logsetId: xxxxxx-xx-xx-xx-xxxxxxxx  ## The ID of the CLS logset. The logset nee
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx      ## CLS log topic ID. The log topic needs

    logType: json_log                 ## Log collection format. json_log: json form
    logFormat: xxx                      ## Log formatting method
    period: 30                                          ## Lifecycle in days. Value
    partitionCount:                     ## The number (an integer) of log topic par
    tags:                             ## Tag description list. This parameter is us
     - key: xxx                                                  ## Tag key
     value: xxx                      ## Tag value
    autoSplit: false## Whether to enable automatic split (Boolean type). Default va
    maxSplitPartitions:
    storageType: hot.                ## Log topic storage class. Valid values: `hot
    excludePaths:                      ## Collection path blocklist
      - type: File                                               ## Type. Valid va
          value: /xx/xx/xx/xx.log         ## The value of `type`
    indexs:                                                     ## You can customi
      - indexName: ## When a key value or metafield index needs to be configured fo
          indexType: ## Valid values: `long`, `text`, `double`
          tokenizer:  ## Field delimiter. Each character represents a delimiter. On
          sqlFlag: ## Whether the analysis feature is enabled for the field (Boolea
          containZH:  ## Whether Chinese characters are contained (Boolean)
    region: ap-xxx                     ## Topic region for cross-region shipping
    userDefineRule: xxxxxx             ## Custom collection rule, which is a serial
    extractRule: {}                    ## Extraction and filter rule. If `ExtractRu
```

inputDetail description

```
inputDetail:
  type: container_stdout                ## Log collection type, including contain

  containerStdout:                      ## Container standard output
    namespace: default    ## The Kubernetes namespace of the container to be coll
    excludeNamespace: nm1,nm2   ## The Kubernetes namespace of the container to b
    nsLabelSelector: environment in (production),tier in (frontend) ## Filter nam
    allContainers: false       ## Whether to collect the standard output of all c
    container: xxx               ## Name of the container of which the logs will 
    excludeLabels:  ## Pods with the specified labels will be excluded. This fiel
      key2: value2  ## Pods with multiple values of the same key can be matched.
    includeLabels:  ## Pods with the specified labels will be collected. This fie
      key: value1   ## The `metadata` will be carried in the log collected based

    metadataLabels:            ## Specify the Pod labels to be collected as the m
    - label1
    customLabels:              ## Custom metadata
      label: l1

    workloads:
    container: xxx  ## Name of the container to collect. If this parameter is not
      kind: deployment  ## Workload type. Supported values include deployment, da
      name: sample-app  ## Workload name
      namespace: prod  ## Workload namespace

  containerFile:  ## File in the container
    namespace: default  ## The Kubernetes namespace of the container to be collec
    excludeNamespace: nm1,nm2    ## The Kubernetes namespace of the container to b
    nsLabelSelector: environment in (production),tier in (frontend) ## Filter nam
    container: xxx          ## The name of container of which the logs will be co
    logPath: /var/logs     ## Log folder. Wildcards are not supported.
    filePattern: app_*.log     ## Log file name. It supports the wildcards [*?].
    customLabels:   ## Custom metadata
      key: value
    excludeLabels:  ## Pods with the specified labels will be excluded. This fiel
      key2: value2  ## Pods with multiple values of the same key can be matched.

    includeLabels:  ## Pods with the specified labels will be collected. This fie
      key: value1   ## The `metadata` will be carried in the log collected based
    metadataLabels:            ## Specify the Pod labels to be collected as the m
    - label1               ## pod label
    workload:
      container: xxx           ## Name of the container to collect. If this param
      name: sample-app             ## Workload name

  hostFile:                ## Node file path
    filePattern: '*.log'             ## Log file name. It supports the wildcards
    logPath: /tmp/logs     ## Log file folder. Wildcards are not supported.
    customLabels:   ## Custom metadata
      label1: v1
```

extractRule description

Name	Type	Required	Description
timeKey	String	No	Time field key name. `time_key` and `time_format` must appear in pairs.

timeFormat	String	No	Time field format. For more information, see the output parameters of the time format description of the `strftime` function in C language.
delimiter	String	No	Delimiter for delimited log, which is valid only if `log_type` is `delimiter_log` .
logRegex	String	No	Full log matching rule, which is valid only if `log_type` is `fullregex_log` .
beginningRegex	String	No	First-Line matching rule, which is valid only if `log_type` is `multiline_log` or `fullregex_log` .
unMatchUpload	String	No	Whether to upload the logs that failed to be parsed. Valid values: `true` : yes; `false` : no.
unMatchedKey	String	No	Key of the failure log
backtracking	String	No	Size of the data to be rewound in incremental collection mode. Value: `-1` (collect all) and `0` (collect increment). It default to `-1` .
keys	Array of String	No	Key name of each extracted field. An empty key indicates to discard the field. This parameter is valid only if `log_type` is `delimiter_log` . `json_log` logs use the key of JSON itself.
filterKeys	Array of String	No	Key of the log to filter. It corresponds to `FilterRegex` by the index.
filterRegex	Array of String	No	Regex corresponding to the key of the log to filter. It corresponds to `FilterKeys` by the index.
isGBK	String	No	Whether it's GBK-encoded. Values: `0` (No), `1` (Yes) **Note:** This field may return null, indicating that no valid value was found.
jsonStandard	String	No	Whether it's standard JSON. Values: `0` (No), `1` (Yes). **Note:** This field may return null, indicating that no valid value was found.

kafkaDetail description

```
kafkaDetail:
  brokers: x.x.x.x:p     ## (Required) The broker address. Generally, it is domai
  topic: test      ##
  kafkaType: CKafka     ## Kafka type. Valid values: `CKafka` (CKafka); `SelfBuil
  instanceId: xxxx      ## The ID of the CKafka instance when `kafkaType` is `CKa
     logType: minimalist_log ## The type of the parsed Kafka log. Valid values: `
  timestampFormat: xxx      ## The format of timestamp. It defaults to `double`.
  timestampKey: xxx           ## The key of timestamp. It defaults to `@timestamp
  metadata:
    formatType: default   ## Metadata format. Valid values: `default` (default, t
  messageKey:            ## Specify a key to ship logs to the specified partition
```

```
        value: Field        ## Topic ID, which is required
        valueFrom:
          fieldRef:
            fieldPath: metadata.name ## If the key is `Field`, you can select `metada
```

status description

status	Description
The status is empty.	Initial status
Synced	Configured successfully
Stale	Configuration failed

Sample CRD

Sample CRD for the configuration of the container standard output

All containers

Specifying a workload

Specifying Pod labels

Specify a namespace

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: "test"
spec:
  clsDetail:
    .......
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: true
      namespace: default,kube-public
    type: container_stdout
```

Exclude a namespace

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: "test"
spec:
  clsDetail:
    ........
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: true
      excludeNamespace: kube-system,kube-node-lease
    type: container_stdout
```

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
```

```
metadata:
  name: "test"
spec:
  clsDetail:
    ......
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      allContainers: false
      workloads:
      - container: prod
        kind: deployment
        name: sample-app
        namespace: kube-system
    type: container_stdout
```

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    ......
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
  inputDetail:
    containerStdout:
      container: prod
      excludeLabels:
        key2: v2
      includeLabels:
        key1: v1
      namespace: default,kube-system
    type: container_stdout
```

Sample CRD for the configuration of the container file path

Specifying a workload

Specifying Pod labels

```yaml
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    .......
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    containerFile:
      container: prod
      filePattern: '*.log'
      logPath: /tmp/logs
      namespace: kube-system
      workload:
        kind: deployment
        name: sample-app
    type: container_file
```

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  name: test
spec:
  clsDetail:
    .......
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    containerFile:
      container: prod
      filePattern: '*.log'
      includeLabels:
        key1: v1
      excludeLabels:
        key2: v2
      logPath: /tmp/logs
      namespace: default,kube-public
    type: container_file
```

Sample CRD for the configuration of the node file path

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
metadata:
  creationTimestamp: "2022-03-13T12:48:49Z"
  generation: 4
  name: test
  resourceVersion: "11729531"
  selfLink: /apis/cls.cloud.tencent.com/v1/logconfigs/test
  uid: 233f4b72-cfef-4a43-abb8-e4d033185097
spec:
  clsDetail:
    .......
    topicId: xxxx-xx-xx-xx-xxxx
  inputDetail:
    hostFile:
      customLabels:
        testmetadata: v1
      filePattern: '*.log'
      logPath: /var/logs
    type: host_file
```

Log Add-On Version Upgrade

Last updated：2023-05-19 15:51:28

Overview

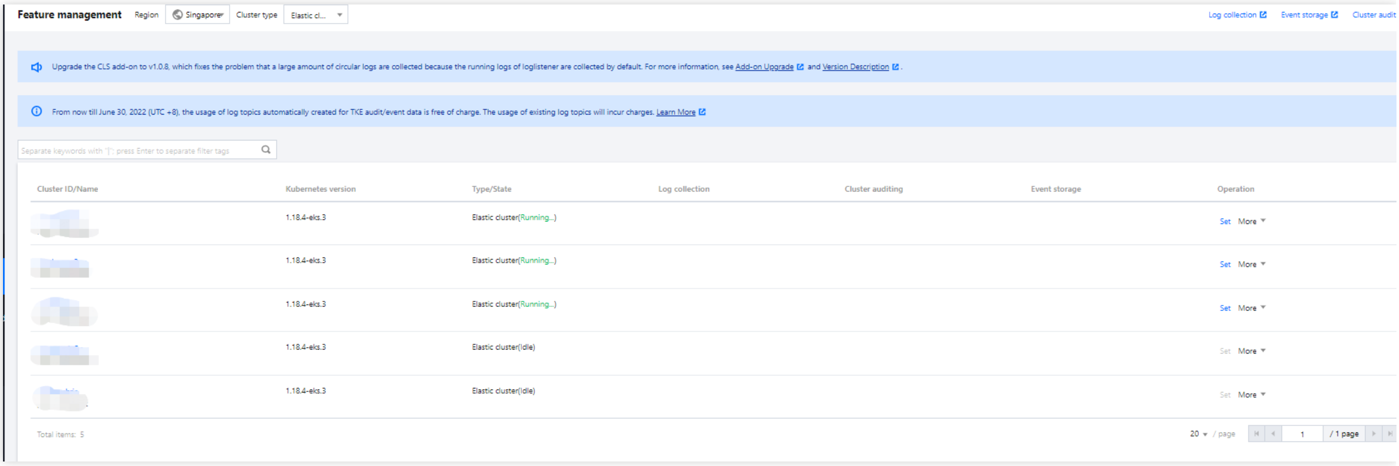
The TKE Ops Center provides the log add-on version upgrade feature. If you have enabled log collection, you can view the current add-on version and perform manual upgrades in **Operation Management** in the TKE console.

Upgrade Notice

- The upgrade is an **irreversible** operation.
- The add-on can only be upgraded to a later version. By default, it is upgraded to the latest version.
- During the upgrade, the console will automatically upgrade the accompanying LogListener and Log-Provisioner and update the CRD resources in your cluster to get the latest log feature.
- For more information on versions, see CLS Add-on Version Description.

Directions

1. Log in to the TKE console and click **Operation Management** on the left sidebar.
2. On the **Operation Management** page, select the **Region** and **Cluster Type** at the top of the cluster list. If your cluster has log collection enabled and the add-on can be upgraded, the console will prompt "The add-on can be upgraded" as shown below:



Feature management Region Singapore Cluster type Elastic cl... Log collection Event storage Cluster audit

Upgrade the CLS add-on to v1.0.8, which fixes the problem that a large amount of circular logs are collected because the running logs of loglistener are collected by default. For more information, see Add-on Upgrade and Version Description .

From now till June 30, 2022 (UTC +8), the usage of log topics automatically created for TKE audit/event data is free of charge. The usage of existing log topics will incur charges. Learn More

Separate keywords with "|"; press Enter to separate filter tags

Cluster ID/Name	Kubernetes version	Type/State	Log collection	Cluster auditing	Event storage	Operation
[redacted]	1.18.4-eks.3	Elastic cluster(Running...)				Set More
[redacted]	1.18.4-eks.3	Elastic cluster(Running...)				Set More
[redacted]	1.18.4-eks.3	Elastic cluster(Running...)				Set More
[redacted]	1.18.4-eks.3	Elastic cluster(Idle)				Set More
[redacted]	1.18.4-eks.3	Elastic cluster(Idle)				Set More

Total items: 5 20 / page 1 / 1 page

3. Select your cluster, click **Settings**, and click **Edit** in the **Log Collection** column.
4. Click **Upgrade Add-on** in the **Log Collection** details.

Backup Center

Overview

Last updated：2024-01-19 14:08:31

TKE Backup Center provides integrated solutions for the backup, restoration and migration of containerized applications. This document describes the use cases and the core add-on of the Backup Center.

Note：

Backup Center is in beta test. To try it out, please submit a ticket.

Use cases

Backup restoration: If the resources in a cluster or namespace are deleted by misoperation, you can quickly restore them with the backup data.

Cross-cloud migration: Application data can be migrated between public clouds or between public cloud and private cloud.

Business compliance: Backup data is fetched periodically for auditing.

Core add-ons

Add-on name	Description
tke-backup	A backup add-on deployed in your cluster. It supports scheduled backup and restoration of Kubernetes resources via CRD based on the open-source tool Velero.

Kubernetes objects deployed in a cluster

Kubernetes object name	Type	Specification	Namespaces
tke-backup	Deployment	At least 0.1 core CPU and 256 MB memory	tke-backup
tke-backup	Service	-	tke-backup
tke-backup	backupstoragelocation	-	-
tke-backup	backup	-	-
tke-backup	restores	-	-

Resource types

TKE customized backup related CRD resources are described as follows:

Resource name	Description
Backup	It specifies the backup policy for resource objects. The backup process is launched when a backup task is created. When a backup task is deleted, the underlying data stored in the backup repository COS are not deleted accordingly.
BackupSchedule	It specifies the scheduled backup policy for resource objects. A scheduled backup task is created based on it.
Restore	It restores backup data to the target TKE cluster. The restoration process is launched when you create a restoration task. When you delete a restoration task, the log of restoration operation is removed from the list.

Directions

1. Log in to the TKE console to create a backup repository. For details, see Creating a Backup Repository.
2. Create a backup policy or scheduled backup policy for the target cluster. For details, see Backup Management.
3. Restore the specified resource objects based on the backup data. For details, see Restoration Management.

Backup Repository

Last updated：2024-01-19 14:08:46

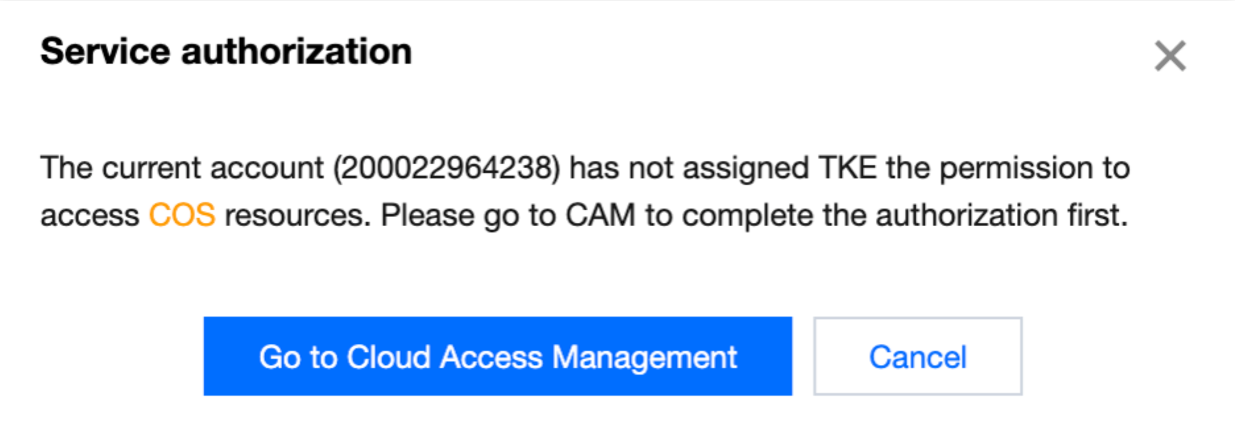
Introduction

TKE Backup Center provides integrated solutions for the backup, restoration and migration of applications. This document describes how to create a backup repository.

Prerequisite

Log in to the COS console. Create a COS bucket, which is used as the underlying storage of the backup repository. A TKE role accesses your COS bucket with the minimum required permission. The bucket name must start with **tke-backup**. For operation details, see Creating A Bucket.

Grant read-write permission on COS objects to the TKE role. Assign the policy `QcloudAccessForTKERoleInCOSObject` to the role `TKE_QCSRole` .



Service authorization

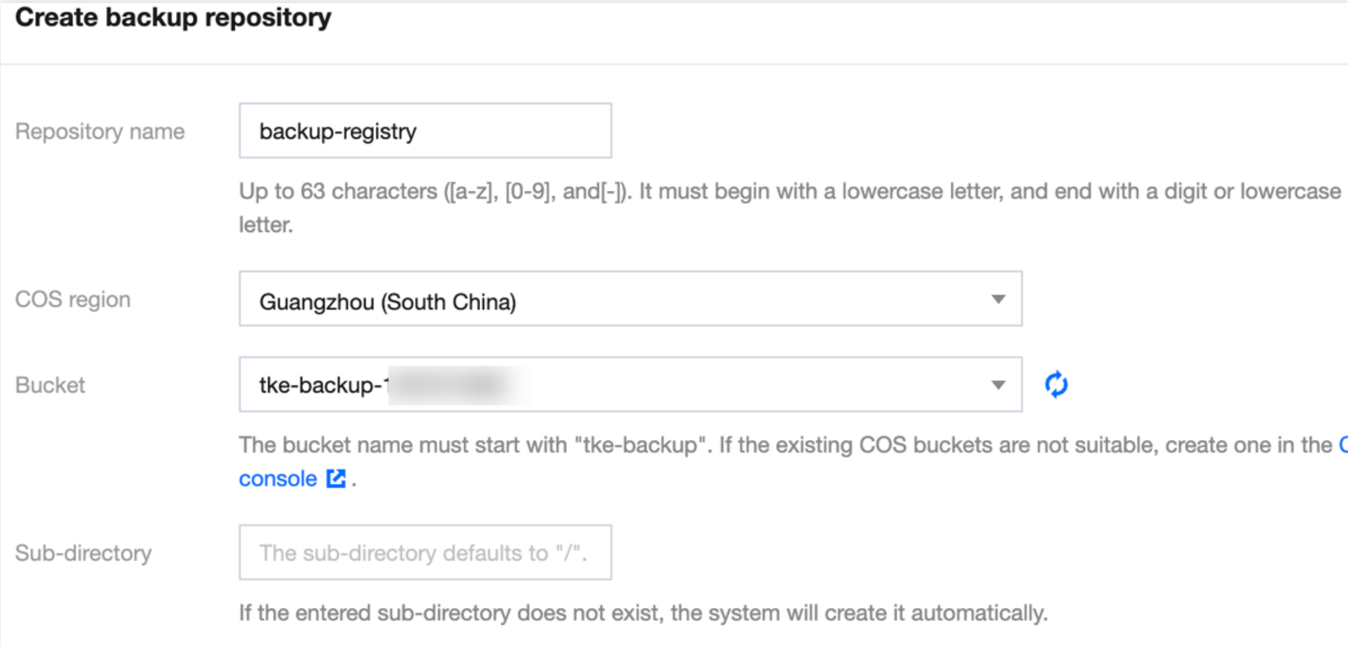
The current account (200022964238) has not assigned TKE the permission to access COS resources. Please go to CAM to complete the authorization first.

Note：

For details on COS billing, see Billing Overview.

Directions

1. Log in to the TKE console and select **Backup Center > Backup Repository** in the left sidebar.
2. Click **Create** on the page that appears.
3. Enter the basic information of the repository.



Create backup repository

Repository name: backup-registry

Up to 63 characters ([a-z], [0-9], and[-]). It must begin with a lowercase letter, and end with a digit or lowercase letter.

COS region: Guangzhou (South China)

Bucket: tke-backup-1

The bucket name must start with "tke-backup". If the existing COS buckets are not suitable, create one in the C console.

Sub-directory: The sub-directory defaults to "/".

If the entered sub-directory does not exist, the system will create it automatically.

Repository name: The name of the backup repository.

COS region: Select the region for COS.

Bucket: The bucket name must start with "tke-backup". If the existing buckets are not suitable, please create one in the COS console.

4. Click **OK**.

Note:

A backup repository can be used by multiple TKE clusters.

When a repository is deleted, the backup resources associated with it cannot be restored normally.

When a repository is deleted, the underlying storage resources are not affected. You can go to the COS console for further operations.

Backup Management

Last updated：2024-01-19 14:09:12

Introduction

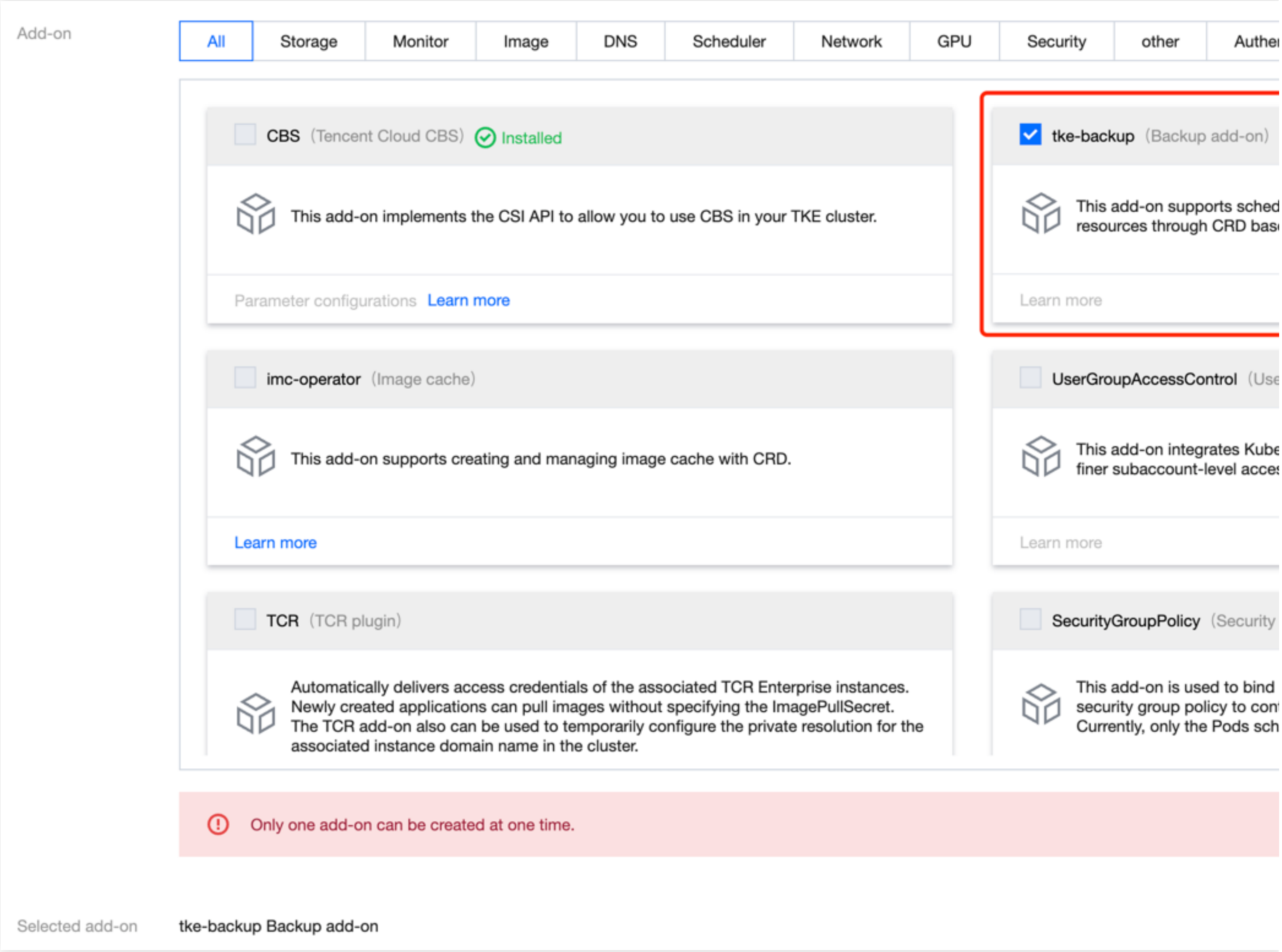
TKE Backup Center provides integrated solutions for the backup, restoration and migration of applications. This document describes how to create a backup task and a scheduled backup policy.

Prerequisite

Note:

If you have installed a community-based open source backup tool, such as Velero, please uninstall it first.

Install the **tke-backup** add-on in the target cluster. You can go to the **Add-on management** module of the cluster to install the add-on. For operation details, see Add-on Installation.



☐ CBS (Tencent Cloud CBS) Installed

This add-on implements the CSI API to allow you to use CBS in your TKE cluster.

☑ tke-backup (Backup add-on)

This add-on supports sched
resources through CRD bas

☐ imc-operator (Image cache)

This add-on supports creating and managing image cache with CRD.

☐ UserGroupAccessControl (Use

This add-on integrates Kube
finer subaccount-level acce

☐ TCR (TCR plugin)

Automatically delivers access credentials of the associated TCR Enterprise instances. Newly created applications can pull images without specifying the ImagePullSecret. The TCR add-on also can be used to temporarily configure the private resolution for the associated instance domain name in the cluster.

☐ SecurityGroupPolicy (Security

This add-on is used to bind
security group policy to con
Currently, only the Pods sch

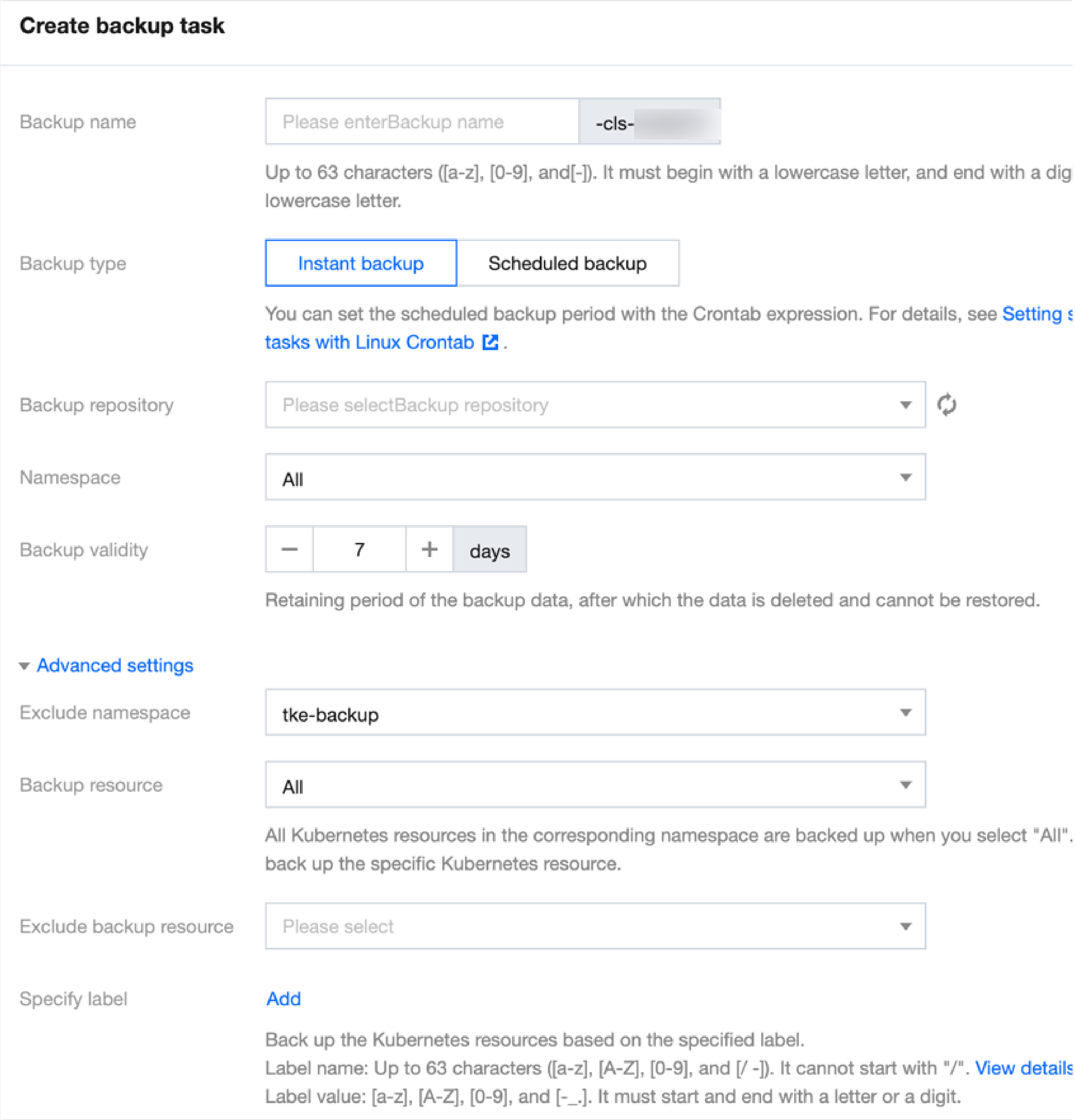
Only one add-on can be created at one time.

Selected add-on tke-backup Backup add-on

Directions

Creating a backup

1. Log in to the TKE console and select **Backup Center > Backup Management** in the left sidebar.
2. Click **Create backup** on the page that appears.
3. Enter backup information.



Create backup task

Backup name: Please enterBackup name -cls-[redacted]

Up to 63 characters ([a-z], [0-9], and[-]). It must begin with a lowercase letter, and end with a dig lowercase letter.

Backup type: Instant backup | Scheduled backup

You can set the scheduled backup period with the Crontab expression. For details, see Setting s tasks with Linux Crontab .

Backup repository: Please selectBackup repository

Namespace: All

Backup validity: – 7 + days

Retaining period of the backup data, after which the data is deleted and cannot be restored.

▾ Advanced settings

Exclude namespace: tke-backup

Backup resource: All

All Kubernetes resources in the corresponding namespace are backed up when you select "All". back up the specific Kubernetes resource.

Exclude backup resource: Please select

Specify label: Add

Back up the Kubernetes resources based on the specified label.
Label name: Up to 63 characters ([a-z], [A-Z], [0-9], and [/ -]). It cannot start with "/". View details
Label value: [a-z], [A-Z], [0-9], and [-_.]. It must start and end with a letter or a digit.

Field description:

Backup name: Enter the backup task name that meets the requirements prompted.

Backup type:

Instant backup: Create a backup task and implement the backup operation instantly.

Scheduled backup: Create the resource object BackupSchedule, which will create backup tasks in a scheduled way based on the rules you set.

Backup repository: Select a backup repository you created.

Namespace: Select a specific namespace or all namespaces, in which all applications are backed up.

Backup validity: Retaining period of the backup data, after which the data is deleted and cannot be restored.

Advanced settings:

Exclude namespace: You can specify the namespace that will not be backed up when you select "All" for **Namespace**.

Backup resource: All Kubernetes resources in the corresponding namespace are backed up when you select "All". Otherwise, back up the specific Kubernetes resource.

Exclude resource: You can specify the resource that will not be backed up when you select "All" for **Backup resource**.

Specify label: Only the applications with the label in the target namespace are backed up.

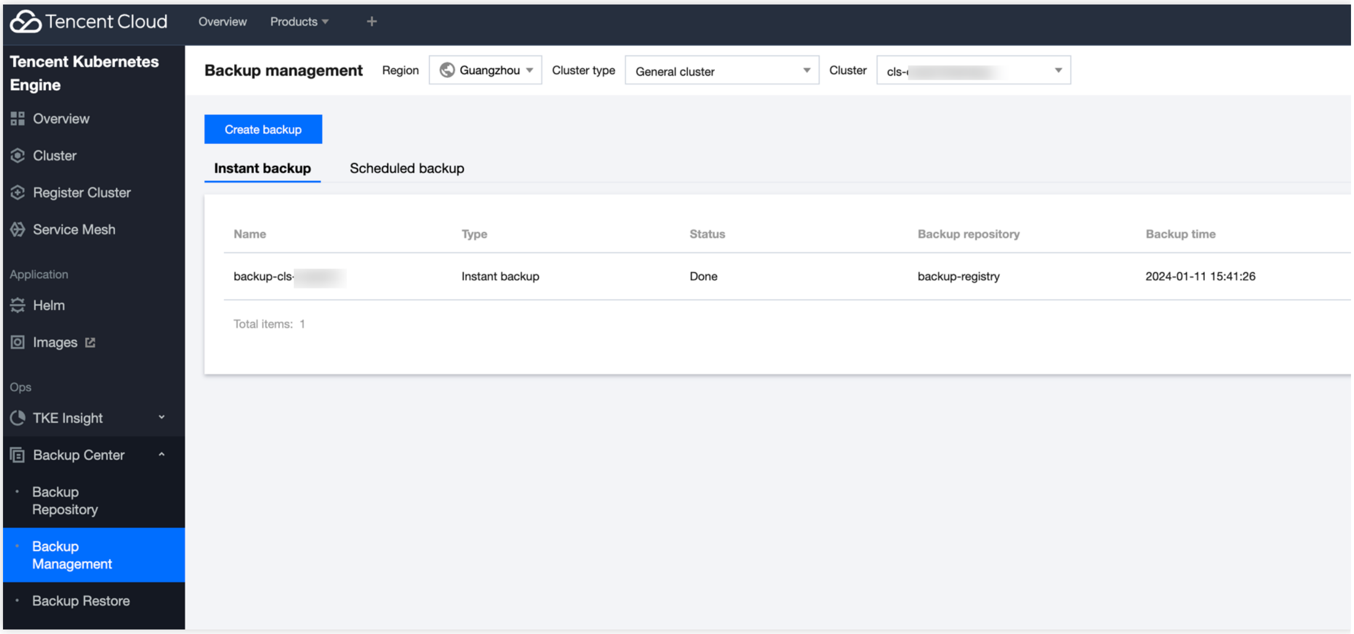
4. Click **OK**.

Note:

The following types of Kubernetes resources are backed up: Deployment, StatefulSet, DaemonSet, Job, CronJob, ConfigMap and Secret.

Viewing backup

You can view the backup on **Instant backup** and **Scheduled backup** tabs on the **Backup management** page.



Backup management Region: Guangzhou Cluster type: General cluster Cluster: cls-[redacted]

Name	Type	Status	Backup repository	Backup time
backup-cls-[redacted]	Instant backup	Done	backup-registry	2024-01-11 15:41:26

Total items: 1

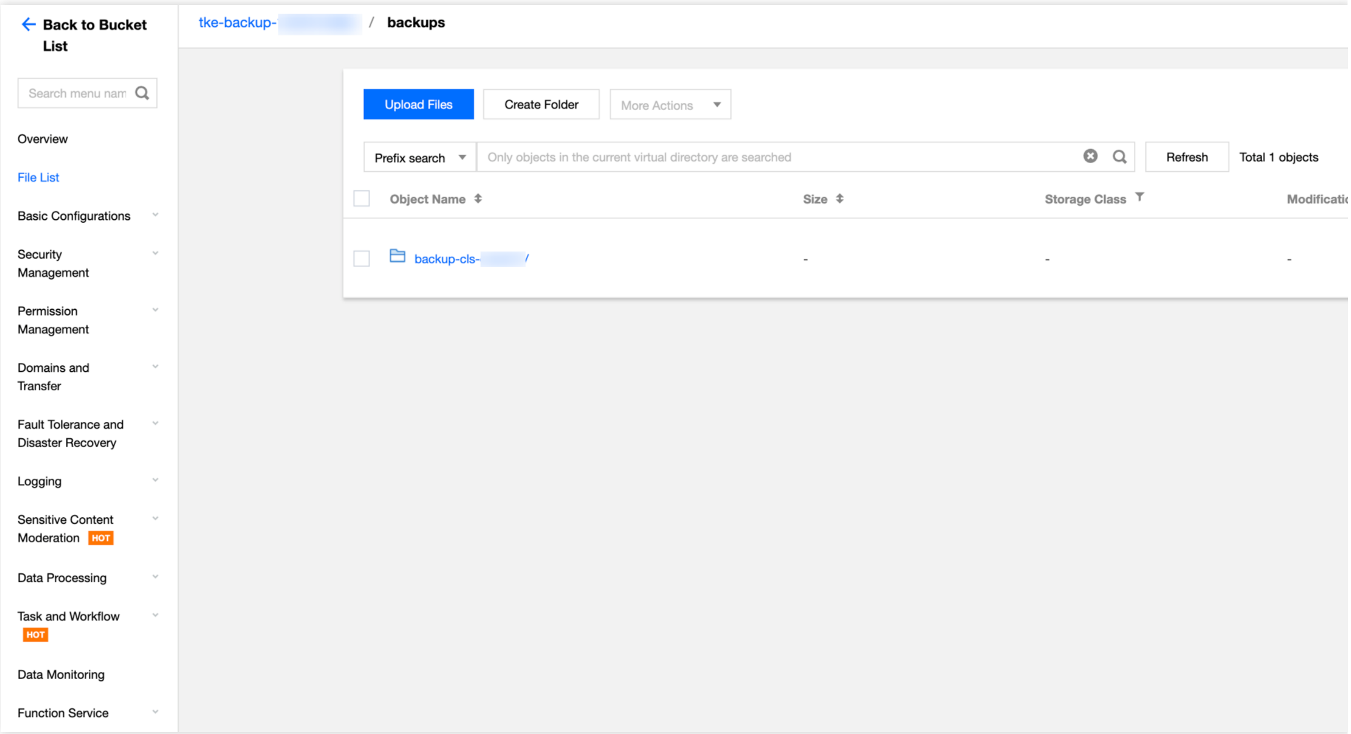
Viewing backup status

Status	Description
Initializing	The backup task is being created.
Executing	The backup task is being implemented.

Acceleration	The backup operation is completed.
Partially failed	Some resource objects are backed up successfully, and others are failed. You can check "Status" in YAML to learn the number of successfully backed up resource objects and the failure reasons.
Failed.	The backup is failed. You can learn the failure reasons in the console or by checking "Status" in YAML.

Viewing backup content

You can view the backup data of the storage in the COS console. Each backup task is named in the format of "Backup name-Cluster name-YYYYMMDDHHmmss".



Restoration Management

Last updated：2024-01-19 14:09:34

Introduction

TKE Backup Center provides integrated solutions for the backup, restoration and migration of applications. This document describes how to restore resources from the target cluster where the backup task has been created.

Note:

Only backup and restoration in a cluster are supported. Cross-cluster migration is not available currently.

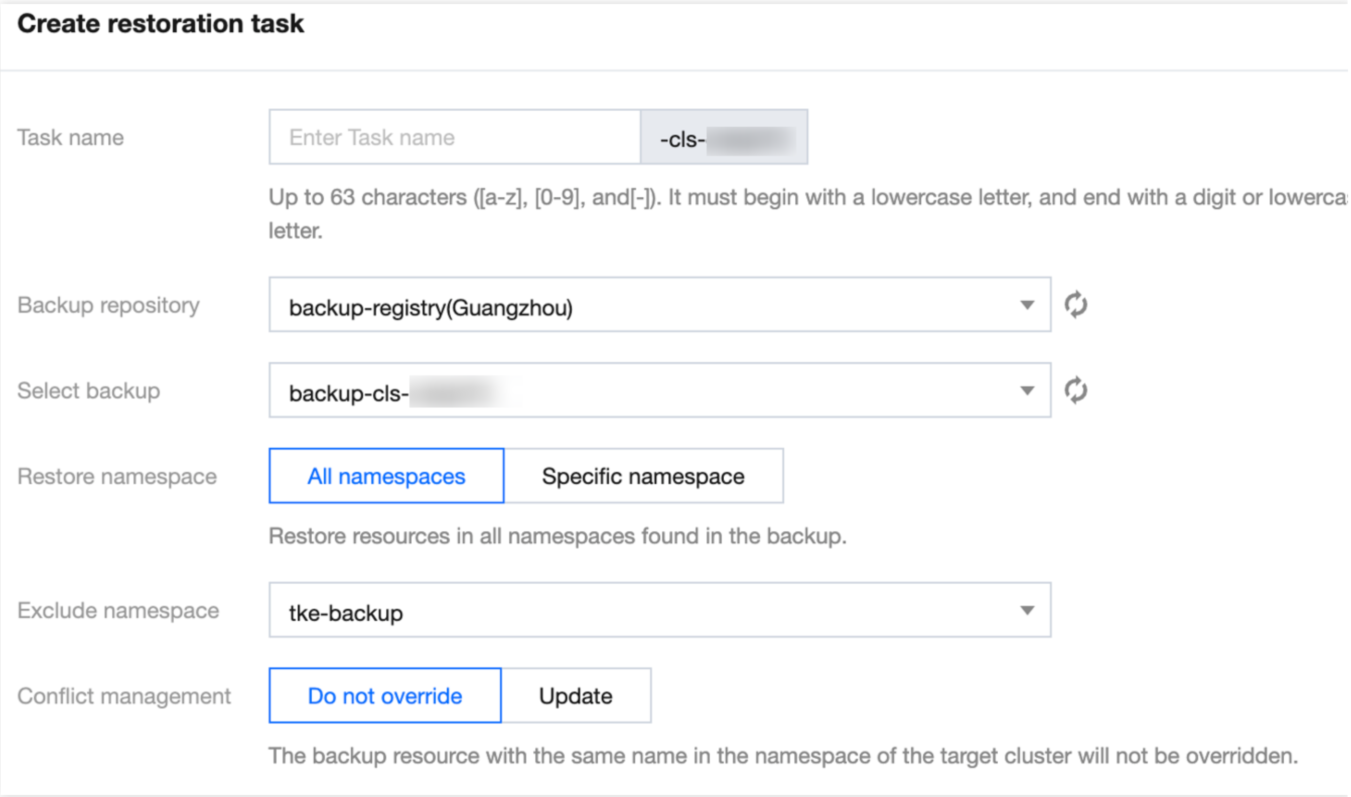
Prerequisite

Create a backup task in the target cluster.

Directions

Creating a restoration task

1. Log in to the TKE console and select Backup Center > Restoration Management in the left sidebar.
2. Click Create restoration task on the page that appears.
3. Enter restoration information.



Create restoration task

Task name: Enter Task name -cls-[redacted]

Up to 63 characters ([a-z], [0-9], and[-]). It must begin with a lowercase letter, and end with a digit or lowerca letter.

Backup repository: backup-registry(Guangzhou)

Select backup: backup-cls-[redacted]

Restore namespace: All namespaces | Specific namespace

Restore resources in all namespaces found in the backup.

Exclude namespace: tke-backup

Conflict management: Do not override | Update

The backup resource with the same name in the namespace of the target cluster will not be overridden.

Field description:

Task name: Enter the restoration task name that meets the requirements prompted.

Backup repository: Select a backup repository you created.

Select backup: Select a backup task to restore.

Restore namespace: Select namespaces from which you want to restore resources.

All namespaces: Restore resources in all namespaces found in the backup.

Specific namespace: Restore resources in the specific namespace that is selected in the backup task.

Conflict treatment:

Do not override (recommended): The backup resource with the same name in the namespace of the target cluster will not be overridden.

Update: The backup resource with the same name in the namespace of the target cluster will be overridden.

4. Click OK.

Note:

The success of the restoration tasks cannot be guaranteed.

Deleting the backup task only removes the operation log of restoration from the restoration list.

Viewing restoration status

Status	Description
Initializing	The restoration task is being created.

Executing	The restoration task is being implemented.
Acceleration	The restoration operation is completed.
Partially failed	Some resource objects are restored successfully, and others are failed. You can check "Status" in YAML to learn the number of successfully restored resource objects and the failure reasons.
Failed	The restoration is failed. You can learn the failure reasons in the console or by checking "Status" in YAML.

Cloud Native Monitoring Overview

Last updated：2021-12-23 16:01:19

Overview

Tencent Prometheus Service (TPS) is a monitoring and alarming solution specially optimized for cloud native service scenarios. It has the full monitoring capabilities of open-source Prometheus and provides a lightweight, stable, and high-availability cloud native monitoring service. It eliminates your need to build a Prometheus monitoring system on your own or care about issues such as data storage, data display, and system OPS, and enables you to enjoy a high-performance multi-cluster cloud native monitoring service after simple configuration.

Overview of Prometheus

Prometheus is an open-source system monitoring and alarming framework. It completely disrupts the testing and alarming models of traditional monitoring systems by forming a new model based on centralized rule computing and unified analysis and alarming. As a project in Cloud Native Computing Foundation with a popularity only second to Kubernetes, it has gradually become a core monitoring component in the era of cloud native thanks to its powerful standalone performance, flexible PromQL, and active community ecosystem.

Strengths of Prometheus

- Support for powerful multidimensional data models.
- Built-in flexible query language PromQL.
- Support for all-around monitoring.
- Great openness.
- Support for target discovery and collection through dynamic service or static configuration.

Shortcomings of open-source Prometheus

- Native (open-source) Prometheus is deployed on a single server and does not provide cluster features, which makes it impossible to use Prometheus to monitor large clusters.
- It cannot easily implement dynamic scaling and load balancing.
- It is technically difficult to deploy and get started with.

Comparison between TPS and open-source Prometheus

Comparison Item	TPS	Open-Source Prometheus

Comparison Item	TPS	Open-Source Prometheus
Scenario	Optimized for container cloud native scenarios	Oriented to multiple scenarios
Weight	Super lightweight	High memory usage
Stability	Higher than native	Not guaranteed
Availability	High	Low
Data storage capability	Unlimited	Subject to local disk capacity
Monitoring of ultra large cluster	Supported	Not supported
Data visualization	Excellent visualization capabilities based on Grafana	Limited visualization capabilities based on native Prometheus UI
Open-Source ecosystem	Full compatibility	Native support
Barrier to use	Low	High
Cost	Low	High

Strengths

Full compatibility with the configurations and core APIs of Prometheus to retain the native features and strengths of Prometheus

TPS supports custom multidimensional data models.

TPS has the built-in flexible query language PromQL.

TPS supports target discovery and collection through dynamic service or static configuration.

TPS is compatible with core Prometheus APIs.

Support for monitoring ultra large clusters

In the performance stress test for a single Prometheus server, when the number of series exceeds 3 million (the length of each label and its value is fixed at 10 characters), the memory usage increases significantly to over 20 GB; therefore, a large-memory server is required for running Prometheus.

TPS can monitor ultra large clusters based on its proprietary sharding technology and unlimited data storage provided by COS.

Support for monitoring multiple clusters in one instance

One TPS instance can be associated with multiple clusters.

Support for template-based management and configuration

TPS allows you to configure templates for monitoring multiple instances and clusters. Then, you can use a template to quickly implement unified multi-cluster monitoring.

Ultra lightweight and non-intrusion monitoring

TPS is lighter than open-source Prometheus. Prometheus uses 16–128 GB memory. In contrast, TPS only requires the deployment of a small agent in your cluster, which uses only 20 MB memory to monitor a cluster with 100 nodes; plus, its memory usage will never exceed 1 GB no matter how large the cluster is.

After you associate your cluster, TPS will automatically deploy the agent in it, so you can start monitoring your businesses without manually installing any component. The ultra lightweight agent has no impact on the businesses and components in your cluster.

Support for real-time dynamic scaling to meet elastic needs

TPS uses Tencent Cloud's proprietary sharding and scheduling technologies to implement real-time dynamic scaling of collection tasks, meeting your elastic needs. It also supports CLB for better load balancing.

High availability

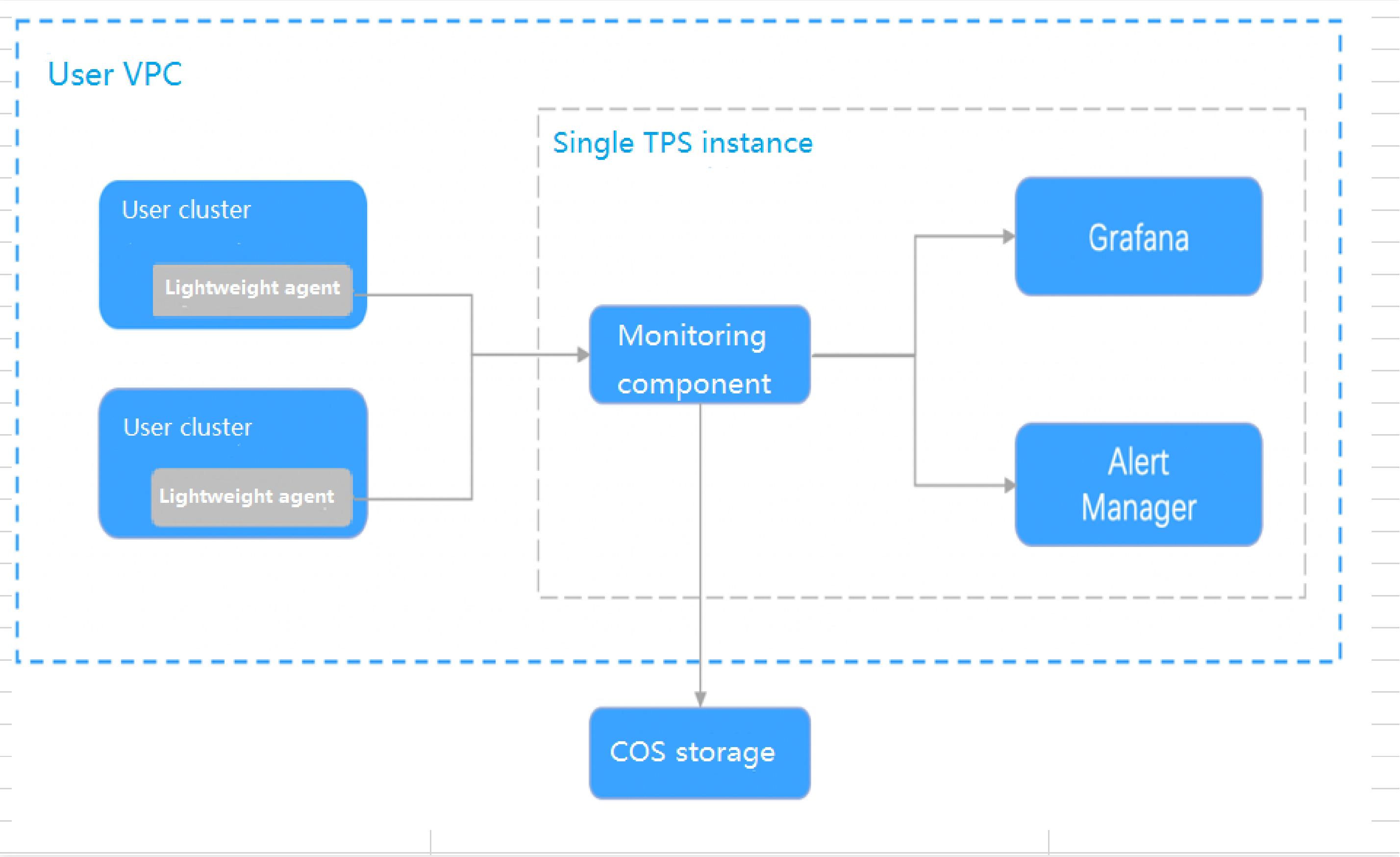
TPS uses technical methods to avoid data breakpoints and losses, so as to secure high availability of the monitoring service.

Low connection costs

You can write configuration files easily in the TPS console, so you don't need to have an extensive knowledge of Prometheus to use TPS. If you already know how to use Prometheus, TPS also allows you to submit configuration information through a native YAML file, making it easier for you to customize advanced features for personalized monitoring.

Product Architecture

As an ultra lightweight, high-availability, and non-intrusion monitoring system, TPS only places a small agent in your cluster. Specifically, the agent in your VPC performs operations such as data collection, remote storage, and query, Grafana visually displays data, and AlertManager is used for alarms. The product architecture is as shown below:



TPS can monitor multiple clusters, businesses outside clusters in the same VPC, and ultra large clusters. It also supports real-time scaling of the monitoring component to secure high availability of the monitoring service.

After you associate a cluster, TPS will add the mainstream collection configuration from the community by default, making it available out of the box without any custom configuration required.

In addition, each TPS instance has a built-in independent Grafana account, which provides a rich variety of preset dashboards and highly customizable monitoring capabilities. In this way, you can implement business-based custom monitoring without caring about the management and scheduling of basic monitoring resources and bottlenecks in the monitoring performance, and enjoy the best monitoring service at the minimum costs.

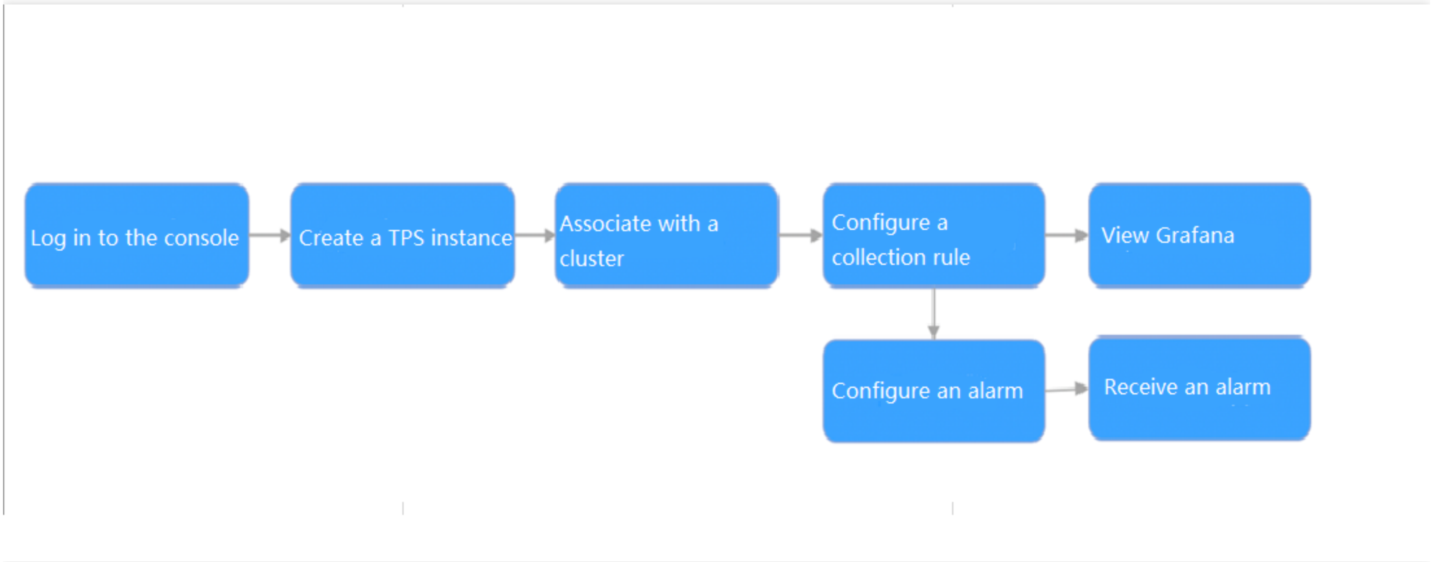
Directions

Log in to your Tencent Cloud account, go to the TPS console, authorize COS as prompted, and use TPS as follows:

1. Create a TPS instance. For more information, see TPS Instance Management.
2. Associate a cluster with the newly created TPS instance. At this point, the system will automatically deploy the agent in your cluster and deploy the monitoring component in your VPC, so you don't need to install any plugins. For

more information, see Associating Cluster.

3. Configure a collection rule. After a cluster is successfully associated, you can flexibly configure the data collection and alarm rules as needed. Then, you can open Grafana to view the monitoring data. For more information, see Configuring Collection Rule and Alarm Configuration.



Key concepts

- **TPS instance**: a TPS instance corresponds to a complete set of monitoring services and has an independent GUI. One instance can be associated with multiple clusters in the same VPC to implement unified multi-cluster monitoring.
- **Cluster:** it generally refers to your TKE or EKS cluster in Tencent Cloud.
- **Cluster association**: it refers to the operation of associating a TPS instance with a cluster.
- **Collection rule:** it refers to a custom monitoring data collection rule.
- **Job:** in Prometheus, a job is a collection task, which defines the public configurations of all monitoring targets in a job workload. Multiple jobs can form the configuration file of a collection task.
- **Target:** it refers to a data collection target obtained through static configuration or service discovery. For example, when a Pod is monitored, the target will be each container in the Pod.
- **Metric:** it is used to record the monitoring metric data. All metrics are time series data and identified by metric name. The sample data collected by each metric contains information in at least three dimensions (metric name, time, and metric value).
- **Series:** it is a metric-label pair and represented as a straight line on a dashboard.

Use Cases

TPS mainly monitors container cloud native business scenarios. In addition to the implementation of mainstream container and Kubernetes monitoring solutions, it also flexibly supports custom monitoring of your businesses, gradually optimizes the preset dashboards in different scenarios, and continuously summarizes industry-specific best practices, in order to help you perform multidimensional analysis and personalized display of monitoring data. It is committed to becoming the best monitoring solution in container scenarios.

Pricing

Currently, TPS is in beta test and free of charge. You only need to pay small storage fees to enjoy the high-quality TPS service. To try it out, go to the TPS console.

Related Services

TPS is responsible for container cloud native monitoring. If you want to use Prometheus to monitor other non-container scenarios, use Managed Service for Prometheus (TMP).

Quick Migration from TPS to TMP

Last updated：2022-06-10 19:32:52

Note
To provide better and more powerful product capabilities, TPS will be merged and upgraded into Tencent Managed Service for Prometheus (TMP). The new TMP service supports cross-region and cross-VPC monitoring and connecting a unified Grafana dashboard to multiple TMP instances for data display in one place. For more information on TMP billing, see Pay-as-You-Go. For Tencent Cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.

TPS will be deactivated on May 16, 2022. For more information, see Announcements. Click here to try out the launched TMP service. TPS instances can no longer be created. You can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency first; otherwise, higher costs may be incurred.

TPS supports quick migration to TMP, where you can migrate a single instance or a batch of instances in the same region. In general, it takes about ten minutes to migrate a TPS instance. **The new TMP instance will be named "old instance name (trans-from-prom-xxx)", where the "old instance name" is the TPS instance name and "xxx" is the TPS instance ID.** After the migration, you can view new monitoring data in the TMP instance and previous monitoring data in the TPS instance. Note that the TPS instance will be deleted upon the end of service.

The migration steps are as follows:

- *1*Migrate the Grafana configuration.
- *2*Create a Grafana instance.
- *3*Create a TMP instance.
- *4*Bind the TMP instance to the cluster associated with the TPS instance.
- *5*Migrate the collection configuration.
- *6*Migrate the alarm policy.
- *7*Migrate the aggregation rule.

Migration Notes

Estimated costs after migration

The capability of **Billable Metric Collection Rate** has been launched for TPS and TMP, which helps you estimate the cost of monitoring by instance, cluster, target, and metric.

> Note：
> Costs will be incurred by TMP instances but not TPS instances.

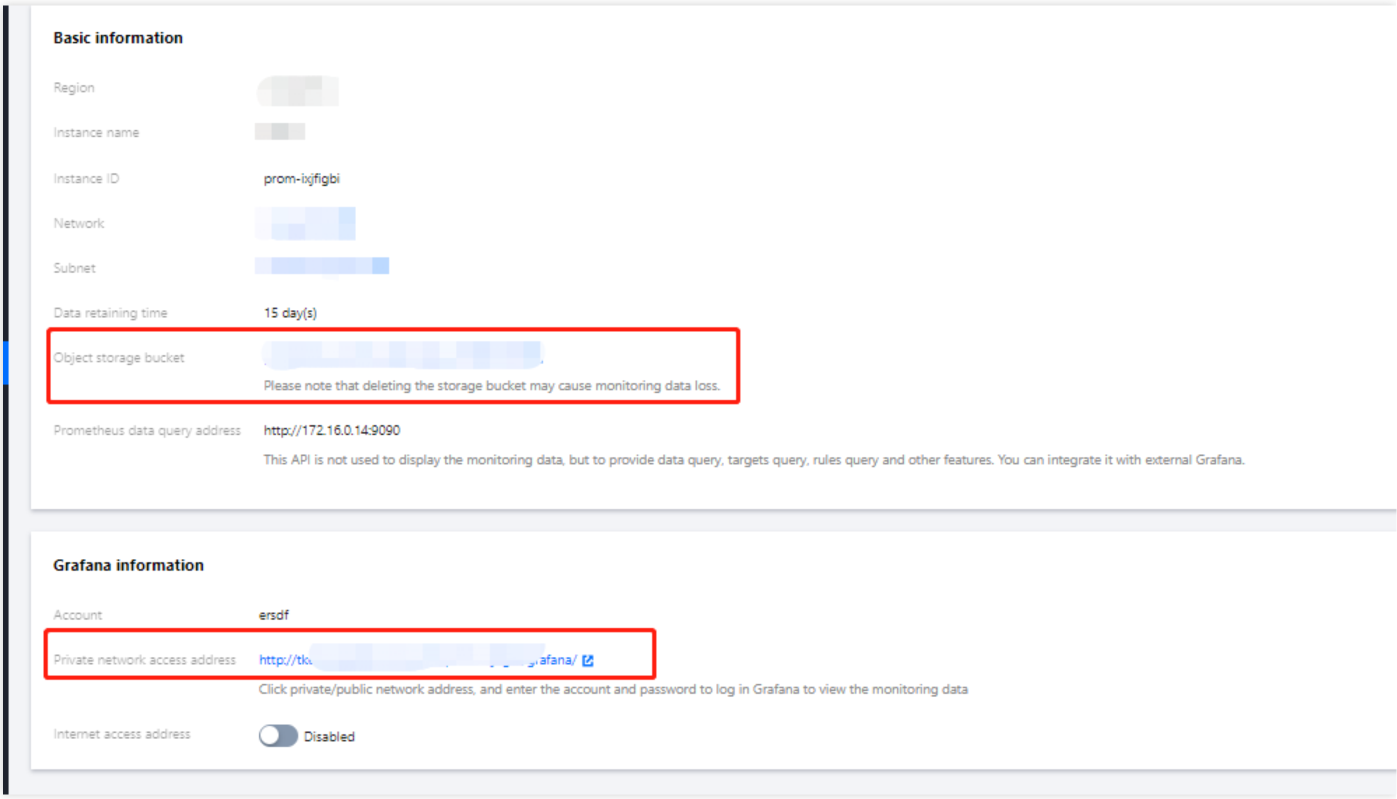
1. Log in to the TKE console and select **Cloud Native Monitoring** on the left sidebar.
2. In the cloud native monitoring list, view the **Billable Metric Collection Rate**, which indicates the collection rate of billable metrics for migration to the TMP instance and is estimated based on your reported metric data volume and the collection frequency. This value multiplied by 86400 is the number of monitoring data points per day, and you can calculate the estimated published price as instructed in Pay-as-You-Go.
   You can also click **Quick Migration** on the right of the instance name to get the estimated price after a TPS instance is migrated to TMP. Or you can view the **Billable Metric Collection Rate** under different dimensions on various pages such as **Associate with Cluster**, **Data Collection Configuration**, and **Metric Details**.

(Old) TPS' Prometheus data query address and Grafana address

If you have application platforms or systems that depend on TPS' **Prometheus data query address and Grafana address**, replace them with the appropriate addresses in TMP promptly after the migration; otherwise, your **Prometheus data query address and Grafana address** will become invalid after the TPS instance is deleted upon the end of service.

1. Log in to the TKE console and select **Cloud Native Monitoring** on the left sidebar.

2. Click the ID of an instance to enter its **Basic Information** page.



Basic information

Region	[redacted]
Instance name	[redacted]
Instance ID	prom-ixjfigbi
Network	[redacted]
Subnet	[redacted]
Data retaining time	15 day(s)
Object storage bucket	[redacted] Please note that deleting the storage bucket may cause monitoring data loss.
Prometheus data query address	http://172.16.0.14:9090 This API is not used to display the monitoring data, but to provide data query, targets query, rules query and other features. You can integrate it with external Grafana.

Grafana information

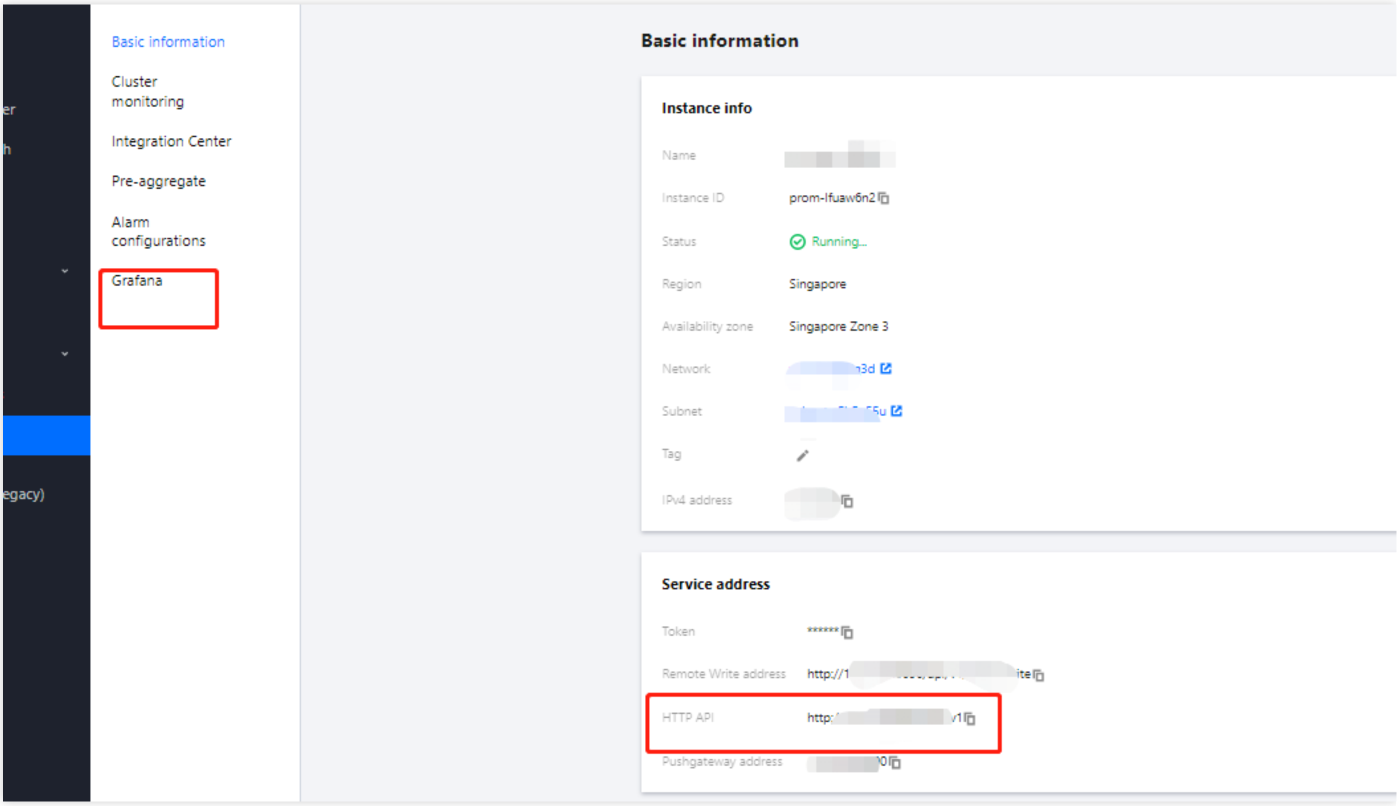
Account	ersdf
Private network access address	http://tk[redacted]afana/ Click private/public network address, and enter the account and password to log in Grafana to view the monitoring data
Internet access address	Disabled

(New) TMP's Prometheus data query address and Grafana address

> Note：
>
> TMP adds authentication to the query API. If you need to connect a TMP instance to your Grafana, use your Tencent Cloud account `APPID` as the username and the token below as the password. For more information, see Querying Monitoring Data.

1. Log in to the TKE console and select **TMP** on the left sidebar.

2. Click the ID of an instance to enter its **Basic Information** page.



Basic information

Instance info

Name [redacted]

Instance ID prom-lfuaw6n2

Status Running...

Region Singapore

Availability zone Singapore Zone 3

Network [redacted]3d

Subnet [redacted]5u

Tag

IPv4 address [redacted]

Service address

Token ******

Remote Write address http://1[redacted]ite

HTTP API http:[redacted]v1

Pushgateway address [redacted]0

> Note：
> After the migration is completed, do not associate new clusters or collection rules with the old TPS instance, as these changes will not be automatically synced to the TMP instance.

Directions

Single instance migration

1. Log in to the TKE console and select **Cloud Native Monitoring** on the left sidebar.
2. On the instance list page, select the region where the instance to be migrated resides.
3. Click **Quick Migration** on the right of the instance.
4. In the pop-up window, select the **Network** and **Data Storage Time** required for the TMP instance.
    - **Network**: The TMP instance has the same VPC and subnet as the TPS instance by default. If you want to select another VPC, make sure that the target VPC is connected to the VPC of the monitored cluster.
    - **Data Storage Time**: **15 days** by default. You can also select **30 days** or **45 days**.
    - **Tag**: Not required. You can select one as needed.

- **Estimated Cost**: During the migration from the current TPS instance to TMP, it displays the **Billable Metric Collection Rate** and estimated daily cost that will take effect after the successful migration.

> Note：
>
> For more information on TMP billing, see Pay-as-You-Go and Tencent Cloud Resource Usage. If the costs are too high, we recommend you streamline monitoring metrics.
>
> 5. Click **OK**. The migration is successful when the TPS instance status changes to **Migrated**.
>
> 6. After the TPS migration is completed, the **TMP console** will display **a new TMP instance named "old instance name (trans-from-xxx)" in the same region, where the "old instance name" is the TPS instance name and "xxx" is the TPS instance ID.**
>
> ? After the migration is completed, do not associate new clusters or collection rules with the TPS instance, as the association will not be automatically synced to the TMP instance.

Batch instance migration

1. Log in to the TKE console and select **Cloud Native Monitoring** on the left sidebar.
2. On the instance list page, select the region where the instances to be migrated reside.
3. Select the instances in the **Not migrated** status and click **Quick Migration** at the top.

> Note：
>
> - You cannot specify the VPC or subnet for TMP instances if you select batch migration. If it is necessary, perform **single instance migration**.
> - Before the migration, see Pay-as-You-Go and Tencent Cloud Resource Usage for TMP billing. If the costs are too high, we recommend you streamline monitoring metrics.

4. Click **OK**. The migration is successful when the TPS instance status changes to **Migrated**.
5. After the TPS migration is completed, the **TMP** console will display **a new TMP instance named "old instance name (trans-from-xxx)" in the same region, where the "old instance name" is the TPS instance name, and "xxx" is the TPS instance ID.**

> Note：
>
> After the migration is completed, do not associate new clusters or collection rules with the TPS instance, as the association will not be automatically synced to the TMP instance.

TMP Instance Management

Last updated：2022-06-22 11:33:00

Note
Tencent Prometheus Service (TPS) has been integrated into TMP, which supports cross-region monitoring in multiple VPCs, and provides a unified Grafana dashboard, allowing for checking of multiple monitoring instances. For more information on TMP billing, see Pay-as-You-Go. For cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.

TPS is discontinued from May 16, 2022 (UTC +8). For more information, see Announcement. Click here to try out TMP. TPS instances can no longer be created, but you can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency; otherwise, higher costs may be incurred.

Overview

In the TKE console, you can easily create a TMP instance and associate it with a cluster in the current region. Clusters associated with the same TMP instance share the same monitoring metrics and alarming policies. Currently, cloud native monitoring supports managed clusters, self-deployed clusters, elastic clusters and edge clusters. This document describes how to create and manage TMP instances in the TKE console.

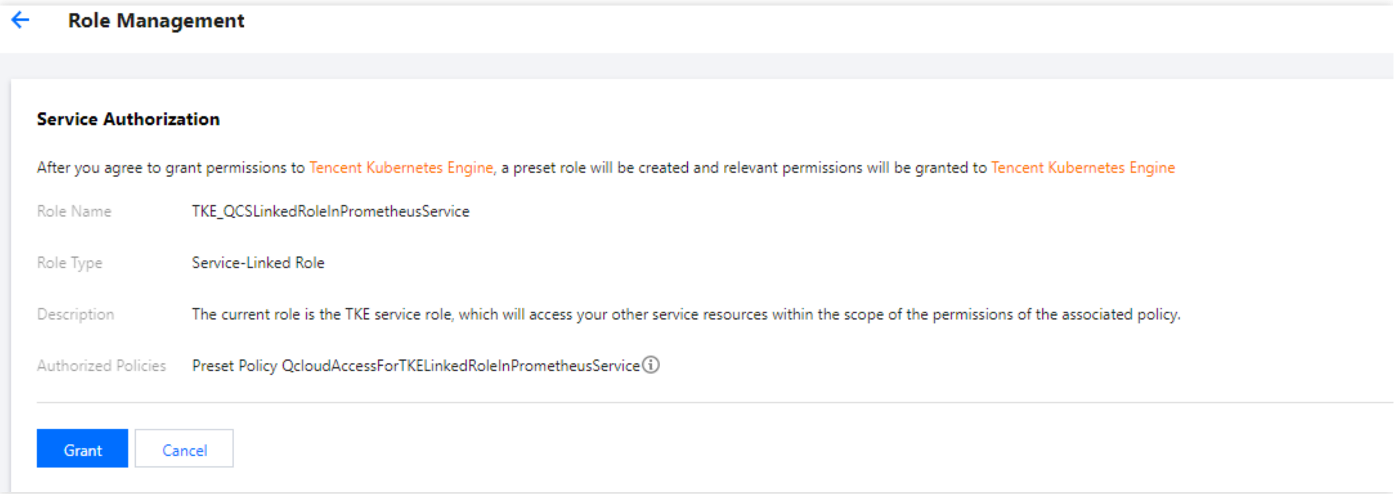
Directions

Service authorization

When using the cloud native monitoring for the first time, you need to assign the `TKE_QCSLinkedRoleInPrometheusService` role to the service, which is used to authorize the cloud native monitoring service to access the COS bucket.

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar to pop up the **Service authorization** window.
2. Click **Go to Cloud Access Management** to enter the **Role management** page.

3. Click **Grant** to complete authentication.



Role Management

Service Authorization

After you agree to grant permissions to Tencent Kubernetes Engine, a preset role will be created and relevant permissions will be granted to Tencent Kubernetes Engine

Role Name TKE_QCSLinkedRoleInPrometheusService

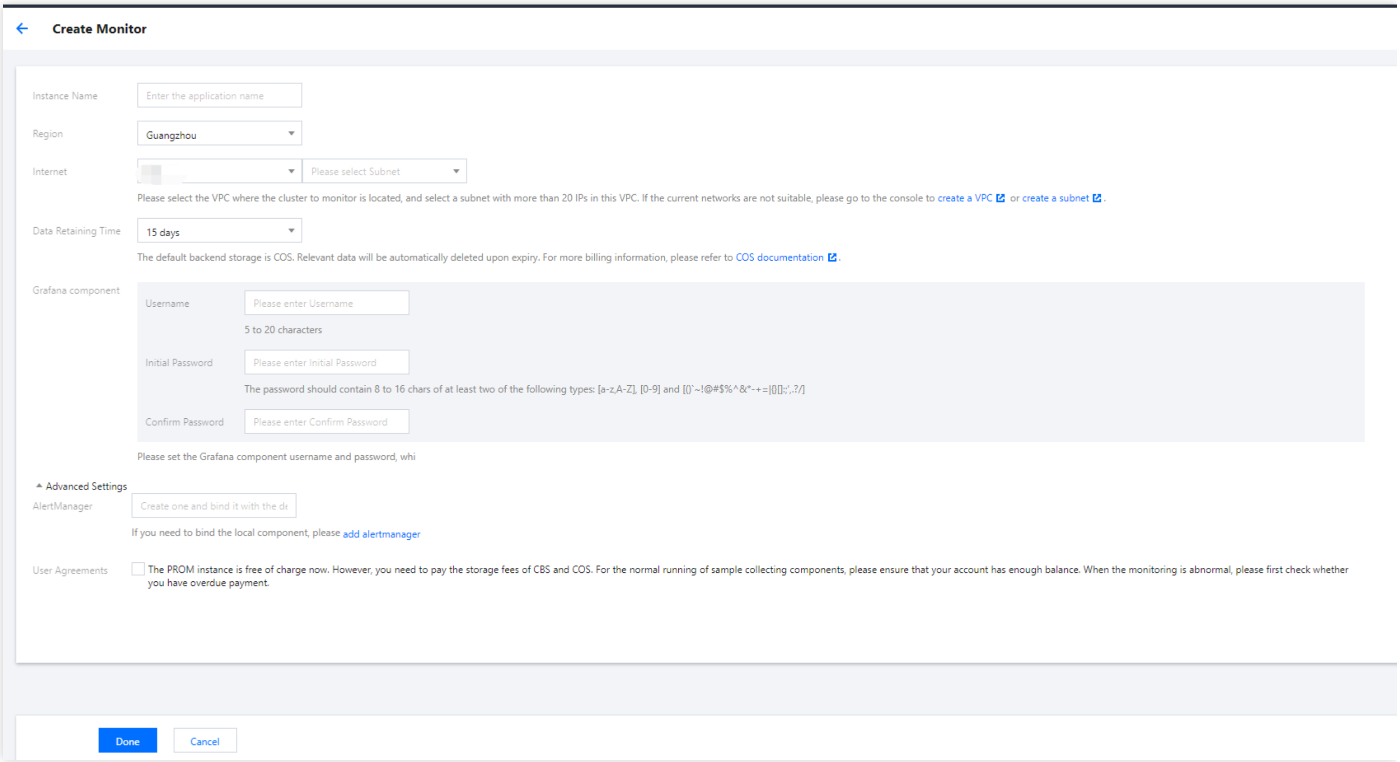
Role Type Service-Linked Role

Description The current role is the TKE service role, which will access your other service resources within the scope of the permissions of the associated policy.

Authorized Policies Preset Policy QcloudAccessForTKELinkedRoleInPrometheusService

Creating TMP instance

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.
2. Click **Create** at the top of the instance list page.
3. On the **Create TMP instance** page, configure the basic information of the instance.



Create Monitor

Instance Name: Enter the application name

Region: Guangzhou

Internet: [redacted] Please select Subnet

Please select the VPC where the cluster to monitor is located, and select a subnet with more than 20 IPs in this VPC. If the current networks are not suitable, please go to the console to create a VPC or create a subnet.

Data Retaining Time: 15 days

The default backend storage is COS. Relevant data will be automatically deleted upon expiry. For more billing information, please refer to COS documentation.

Grafana component

Username: Please enter Username

5 to 20 characters

Initial Password: Please enter Initial Password

The password should contain 8 to 16 chars of at least two of the following types: [a-z,A-Z], [0-9] and [()`~!@#$%^&*-+=|{}[]:;',.?/]

Confirm Password: Please enter Confirm Password

Please set the Grafana component username and password, whi

Advanced Settings

AlertManager: Create one and bind it with the de

If you need to bind the local component, please add alertmanager

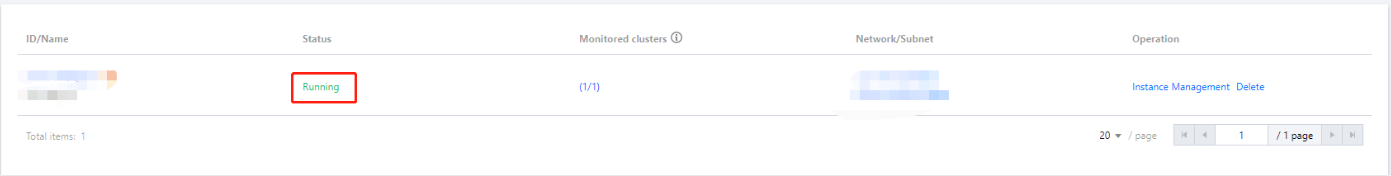
User Agreements: □ The PROM instance is free of charge now. However, you need to pay the storage fees of CBS and COS. For the normal running of sample collecting components, please ensure that your account has enough balance. When the monitoring is abnormal, please first check whether you have overdue payment.

- **Instance name**: Enter a custom instance name within 60 characters.

- **Region**: select a region to deploy this instance. For now, it only supports Beijing, Shanghai and Guangzhou regions. The region cannot be modified after the instance is created. We recommend that you choose the region closest to your business to minimize access latency and improve reporting speed.
- **Network**: Select an existing VPC and its subnet in the current region, which cannot be modified after the instance is created. If there are no VPC resources in the region, you can go to the VPC console to create a VPC. For more information, see Network Settings for Containers and Nodes.
- **Data retaining time**: Select the data storage time (30 days/3 months/6 months/1 year). After the instance is successfully created, a COS bucket will be automatically created and billed based on actual resource usage. For more information, see Billing Overview.
- **Grafana component**: Select whether to enable Grafana access. If you enable it, you need to set the username and password for Grafana logins. After the instance is created, the Grafana username and password cannot be modified. You can enable Grafana public network access according to your business needs.
- **AlertManager**: You can add a custom AlertManager address to send the alarms generated by the instance to your AlertManager.

> Note：
> After the instance is successfully created, you can monitor kubernetes clusters under the VPC to which the instance belongs. If you need to monitor clusters in multiple regions or under different VPCs, you need to create the instance in the same VPC.

4. Click **Complete**.
5. You can check the instance creation progress on the **Cloud native monitoring** page. If the instance status changes to "Running", the instance was successfully created and is running properly.



ID/Name	Status	Monitored clusters ⓘ	Network/Subnet	Operation
[redacted]	Running	(1/1)	[redacted]	Instance Management Delete

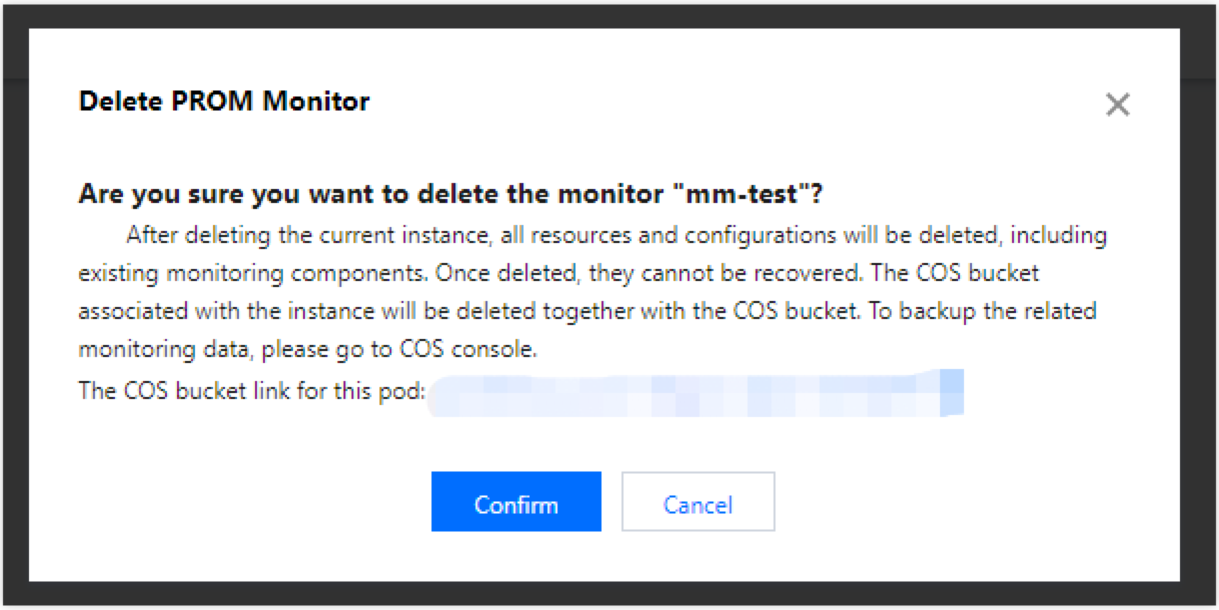
Total items: 1

> Note：
> If it takes too long to create an instance or the displayed status is abnormal, please submit a ticket.

Deleting TMP instance

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.

2. On the instance list page, click **Delete** on the right of an instance to be deleted.
3. In the **Delete TMP instance** pop-up window, click **OK** to delete the instance.



Delete PROM Monitor

Are you sure you want to delete the monitor "mm-test"?

After deleting the current instance, all resources and configurations will be deleted, including existing monitoring components. Once deleted, they cannot be recovered. The COS bucket associated with the instance will be deleted together with the COS bucket. To backup the related monitoring data, please go to COS console.

The COS bucket link for this pod: [redacted]

Note：

When the current instance is deleted, the monitoring components installed in the cluster and the COS bucket associated with the instance will be deleted at the same time by default. To backup the related monitoring data, please go to COS console.

Associating with Cluster

Last updated：2021-12-23 16:01:19

Overview

This document describes how to associate clusters with TPS instances in the cloud native monitoring. When the association is established, you can edit configurations such as data collection rules. Currently, cloud native monitoring only supports the association of the TPS instance and the TKE self-deployed cluster, managed cluster, and elastic cluster under the same VPC to which the instance belongs.

Prerequisites

- You have logged in to the TKE console and created a self-deployed cluster.
- You have created a TPS instance in the VPC of the cluster.

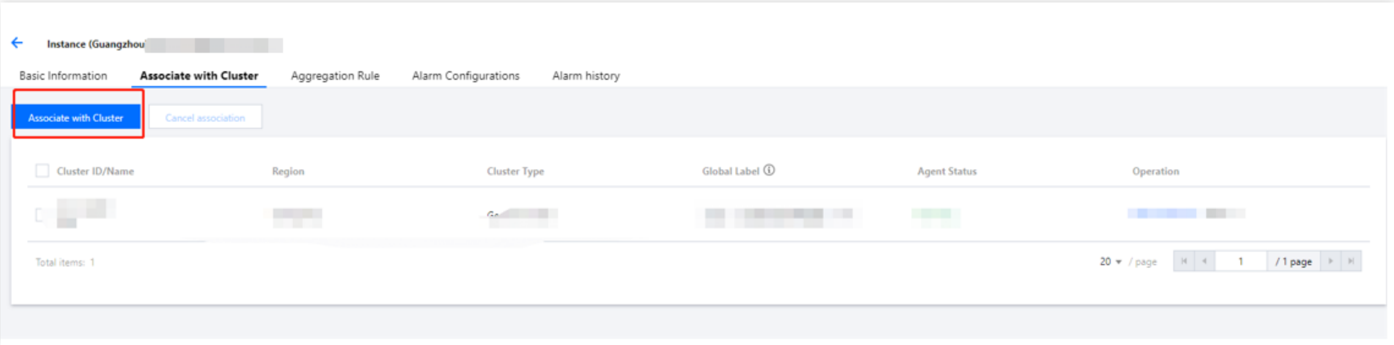
Directions

Associating with cluster

> Note：
> After the cluster is successfully associated, the monitoring data collection add-on will be installed in the cluster, which will be deleted when the cluster is disassociated.

1. Log in to the TKE console and click **Cloud Native Monitoring** on the left sidebar.
2. On the instance list page, select an instance name that needs to associate with the cluster to go to its details page.

3. On **Associate with Cluster** page, click **Associate with Cluster** as shown in the figure below:



Instance (Guangzhou [redacted]

Basic Information | **Associate with Cluster** | Aggregation Rule | Alarm Configurations | Alarm history

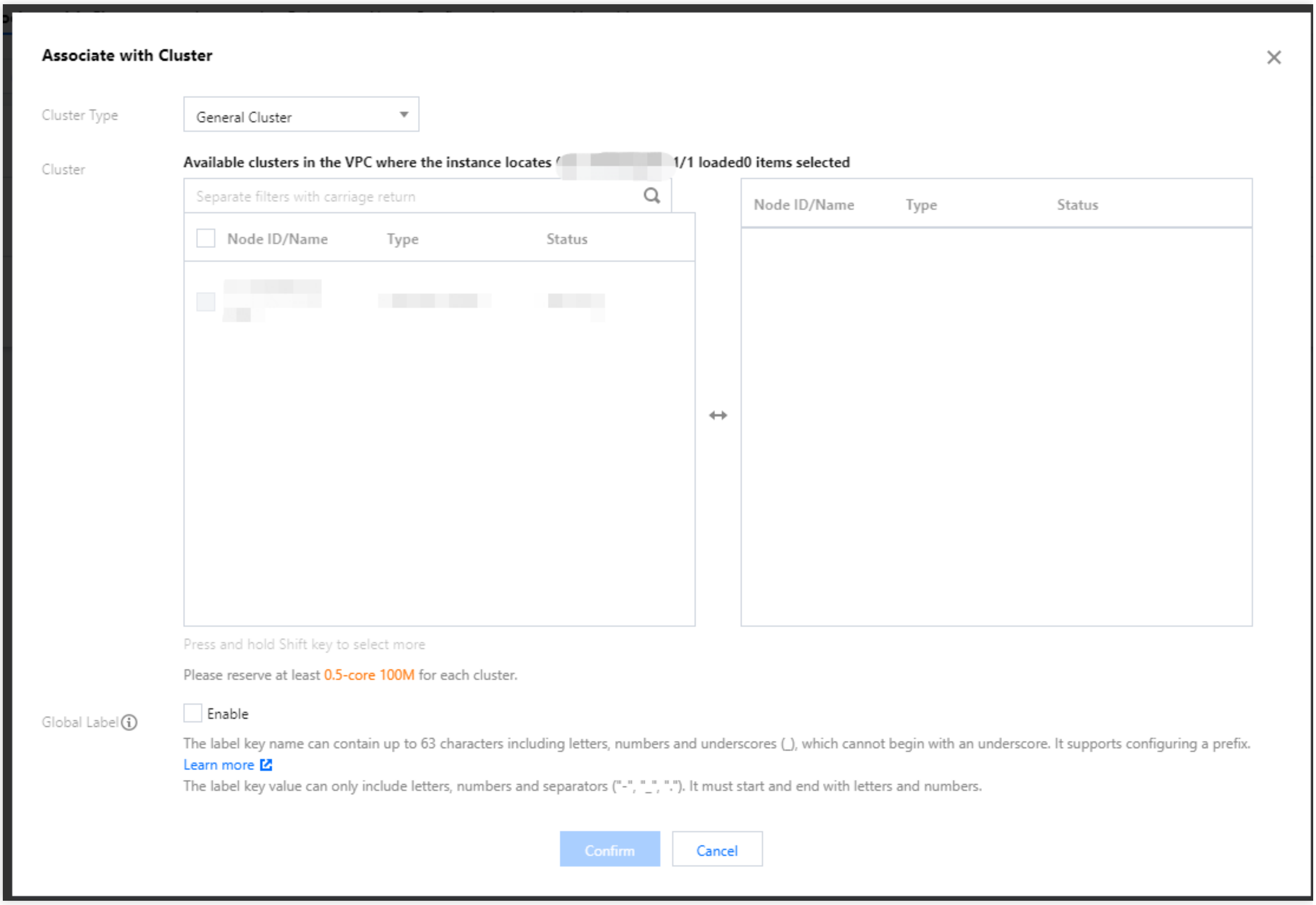
Associate with Cluster | Cancel association

□ Cluster ID/Name	Region	Cluster Type	Global Label ⓘ	Agent Status	Operation
□ [redacted]	[redacted]	Ge[redacted]	[redacted]	[redacted]	[redacted]

Total items: 1

20 ▾ / page 1 / 1 page

4. In the pop up **Associate with Cluster** window, select the cluster to associate with under the current VPC, as shown in the figure below:



Associate with Cluster

Cluster Type: General Cluster ▾

Cluster: Available clusters in the VPC where the instance locates ([redacted] 1/1 loaded0 items selected

Separate filters with carriage return

□ Node ID/Name	Type	Status
□ [redacted]	[redacted]	[redacted]

↔

Node ID/Name	Type	Status

Press and hold Shift key to select more

Please reserve at least 0.5-core 100M for each cluster.

Global Label ⓘ: □ Enable

The label key name can contain up to 63 characters including letters, numbers and underscores (_), which cannot begin with an underscore. It supports configuring a prefix. Learn more

The label key value can only include letters, numbers and separators ("-", "_", "."). It must start and end with letters and numbers.

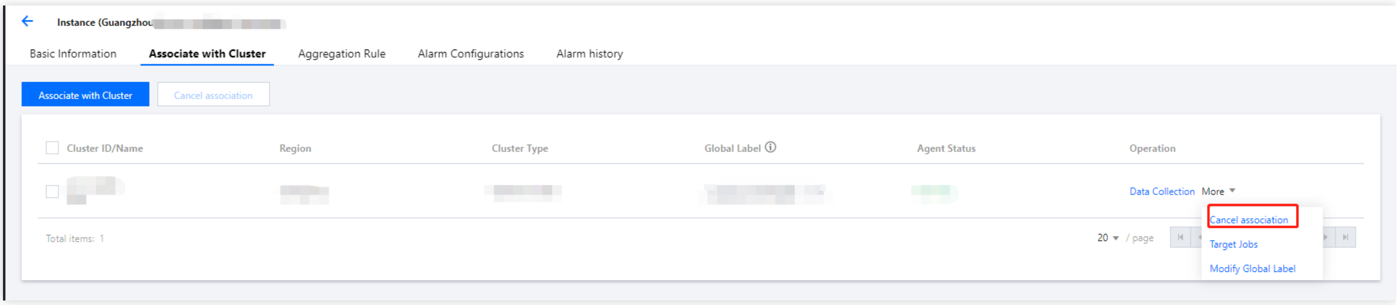
Confirm | Cancel

5. Click **OK**.

Canceling association

1. Log in to the TKE console and click **Cloud Native Monitoring** on the left sidebar.
2. On the instance list page, select an instance name that needs to cancel association to go to its details page.

3. On **Associate with Cluster** page, click **Cancel association** on the right side of the instance as shown below:



4. Click **OK** in the pop up **Disassociate cluster** window.

Data Collection Configurations

Last updated：2022-05-16 15:39:26

> Note
>
> Tencent Prometheus Service (TPS) has been integrated into TMP, which supports cross-region monitoring in multiple VPCs, and provides a unified Grafana dashboard, allowing for checking of multiple monitoring instances. For more information on TMP billing, see Pay-as-You-Go. For cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.
>
> TPS will be discontinued soon. Click here to try out TMP. TPS instances can no longer be created, but you can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency; otherwise, higher costs may be incurred.

Overview

This document describes how to configure monitoring collection items for the associated cluster.

Prerequisites

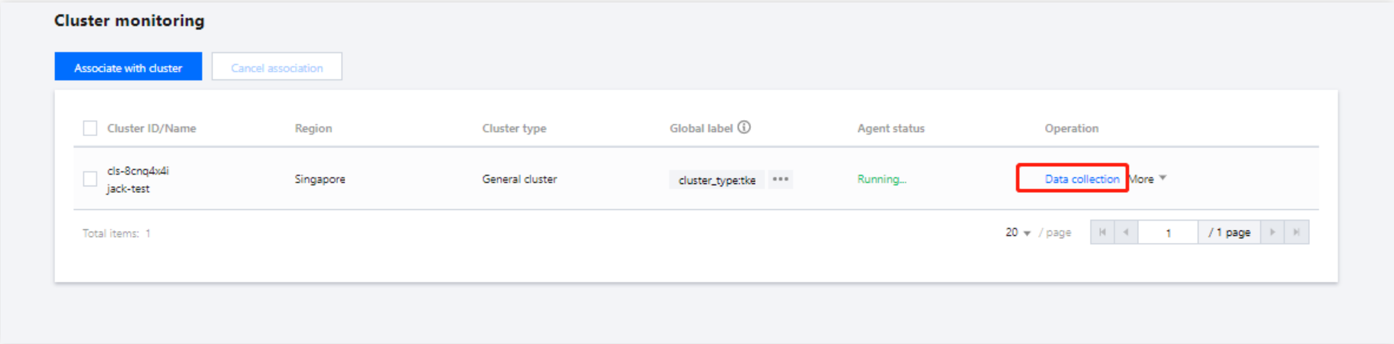
Before configuring monitoring collection items, you need to perform the following operations:

- You've created a TMP instance.
- The cluster to be monitored has been associated with the corresponding instance as instructed in Associating with Cluster.

Directions

Configuring data collection

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.
2. On the instance list page, select the target instance to enter its details page.
3. On the **Associate with cluster** page, click **Data collection** on the right of the instance to enter the collection configuration list page.



Cluster ID/Name	Region	Cluster type	Global label	Agent status	Operation
cls-8cnq4x4i jack-test	Singapore	General cluster	cluster_type:tke •••	Running...	Data collection More

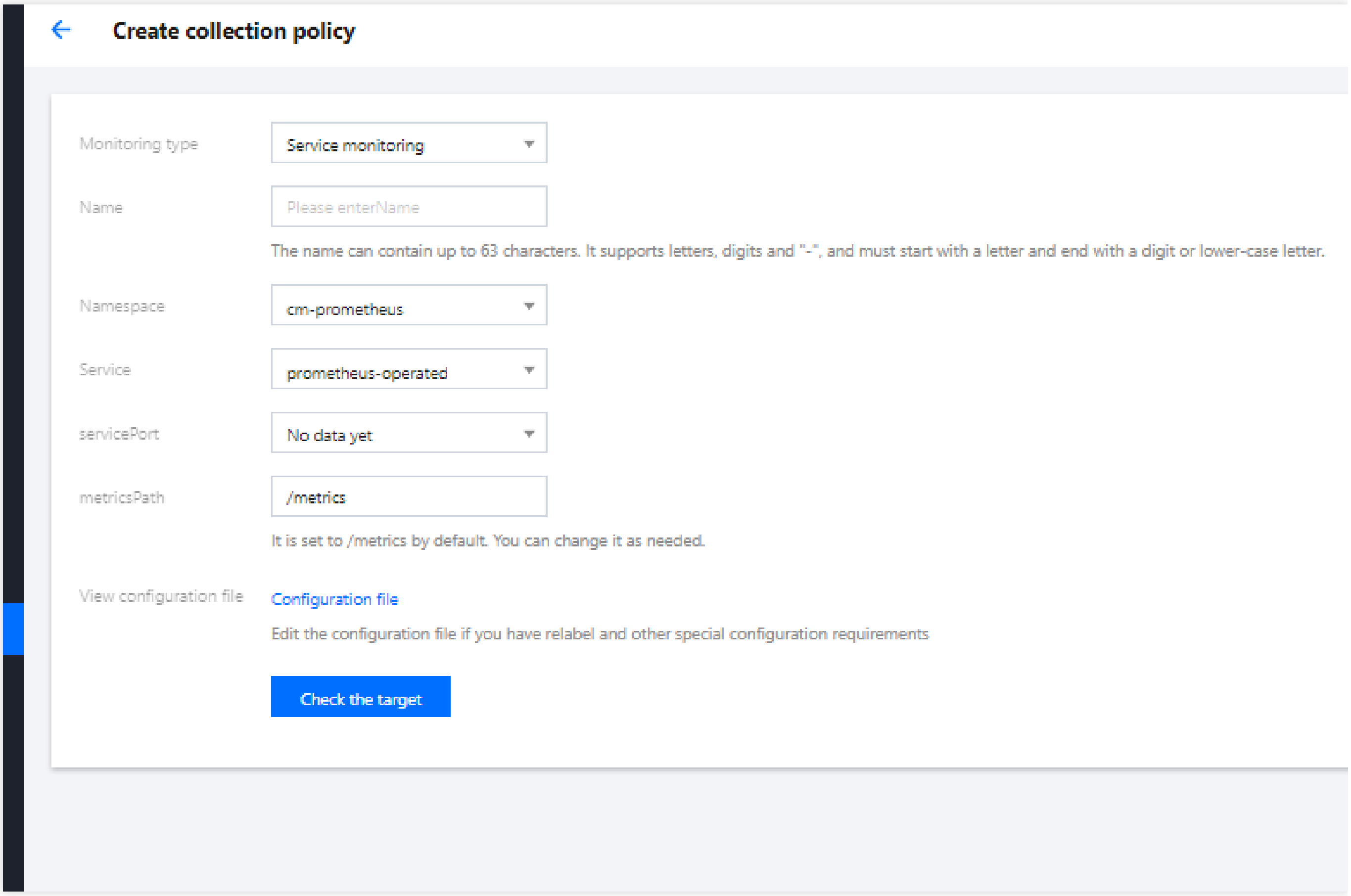
Total items: 1

20 / page 1 / 1 page

4. On the **Data collection** page, add the data collection configuration. The cloud native monitoring has preset some collection configuration files to collect regular monitoring data. You can configure new data collection rules to monitor your business data by using the following two methods:
   - Adding configuration in the console
   - Adding configuration via yaml

Monitoring service

   i. Click **Add**.
   ii. In the **Create collection policy** pop-up window, enter the configuration information.



Monitoring type: Service monitoring

Name: Please enterName

The name can contain up to 63 characters. It supports letters, digits and "-", and must start with a letter and end with a digit or lower-case letter.

Namespace: cm-prometheus

Service: prometheus-operated

servicePort: No data yet

metricsPath: /metrics

It is set to /metrics by default. You can change it as needed.

View configuration file: Configuration file

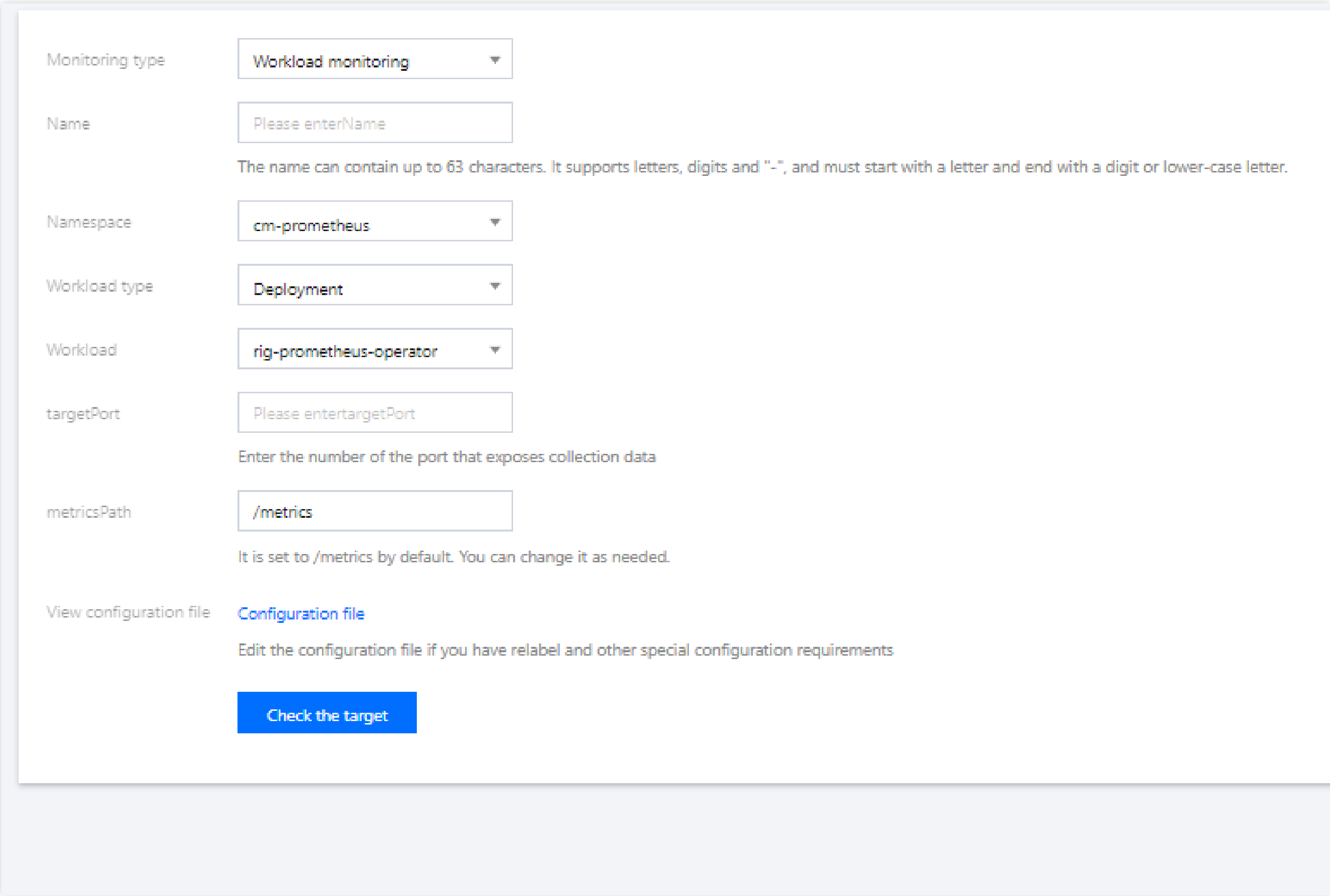
Edit the configuration file if you have relabel and other special configuration requirements

- **Monitoring type**: Select **Service monitoring**.
- **Name**: Enter the rule name.
- **Namespace**: Select the namespace to which the service belongs.
- **Service**: Select the service to be monitored.
- **ServicePort**: Select the corresponding port.
- **MetricsPath**: Defaults to `/metrics`. You can directly enter the collection API as needed.
- **View configuration file**: Click **Configuration file** to view the current configuration file. If you have special configuration requirements such as relabel, you can edit them in the configuration file.
- **Check the target**: Click **Check the target** to view a list of all targets that can be collected under the current collection policy, and confirm whether the collection policy meets your expectations.

Monitoring workload

i. Click **Add**.

ii. In the **Create collection policy** pop-up window, enter the configuration information.



Monitoring type: Workload monitoring

Name: Please enterName

The name can contain up to 63 characters. It supports letters, digits and "-", and must start with a letter and end with a digit or lower-case letter.

Namespace: cm-prometheus

Workload type: Deployment

Workload: rig-prometheus-operator

targetPort: Please entertargetPort

Enter the number of the port that exposes collection data

metricsPath: /metrics

It is set to /metrics by default. You can change it as needed.

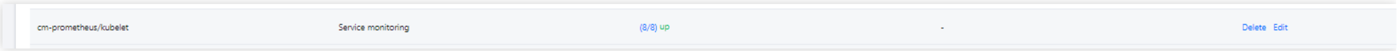
View configuration file: Configuration file

Edit the configuration file if you have relabel and other special configuration requirements

- **Monitoring type**: Select **Workload monitoring**.
- **Name**: Enter the rule name.
- **Namespace**: Select the namespace to which the workload belongs.
- **Workload type**: Select the workload type to be monitored.

- **Workload**: Select the workload to be monitored.
- **targetPort**: Enter the target port that exposes the collection metrics through which the collection target can be found. If the port is incorrect, the collection target will not be obtained correctly.
- **MetricsPath**: Defaults to `/metrics` . You can directly enter the collection API as needed.
- **View configuration file**: Click **Configuration Ffile** to view the current configuration file. If you have special configuration requirements such as relabel, you can edit them in the configuration file.
- **Check the target**: Click **Check the target** to view a list of all targets that can be collected under the current collection policy, and confirm whether the collection policy meets your expectations.

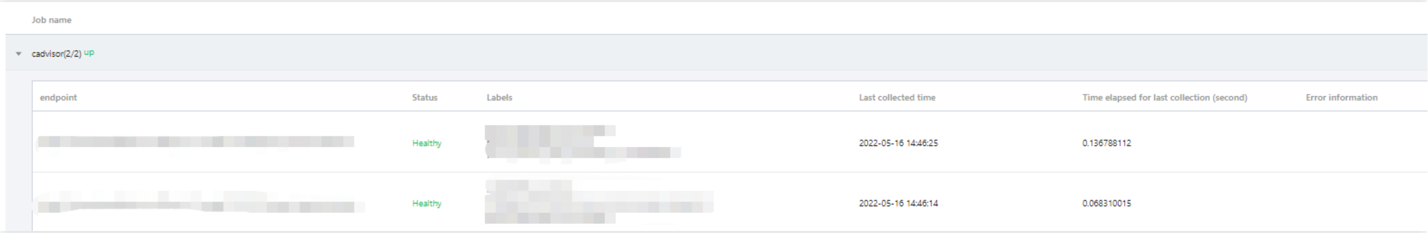
5. Click **OK**.
6. You can view the status of the collection target on the **Data collection** page of the instance.



cm-prometheus/kubelet	Service monitoring	(8/8) up	-	Delete Edit

targets (1/1) indicates one actually captured target/one checked collection target. When the number of actual captured targets equals to the number of checked targets, the status will be "up", which means that the current capture is normal. When the number of actual captured targets is less than the number of checked targets, the status will be "down", which means that some endpoints capture failed.

Click the field value (1/1) to view the details of the collection target.

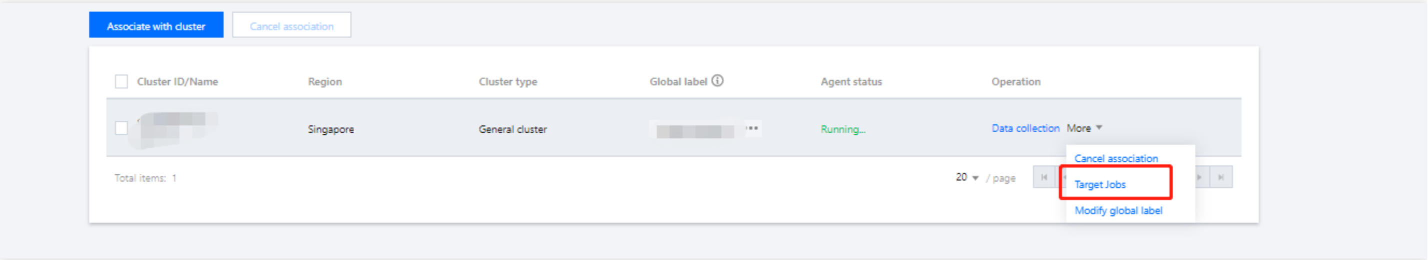


Job name

cadvisor(2/2) up

endpoint	Status	Labels	Last collected time	Time elapsed for last collection (second)	Error information
[redacted]	Healthy	[redacted]	2022-05-16 14:46:25	0.136788112	
[redacted]	Healthy	[redacted]	2022-05-16 14:46:14	0.068310015	

On the **Associate with cluster** tab of the instance, click **More** > **Target Jobs** on the right of the cluster name to view all the collection targets of this cluster.



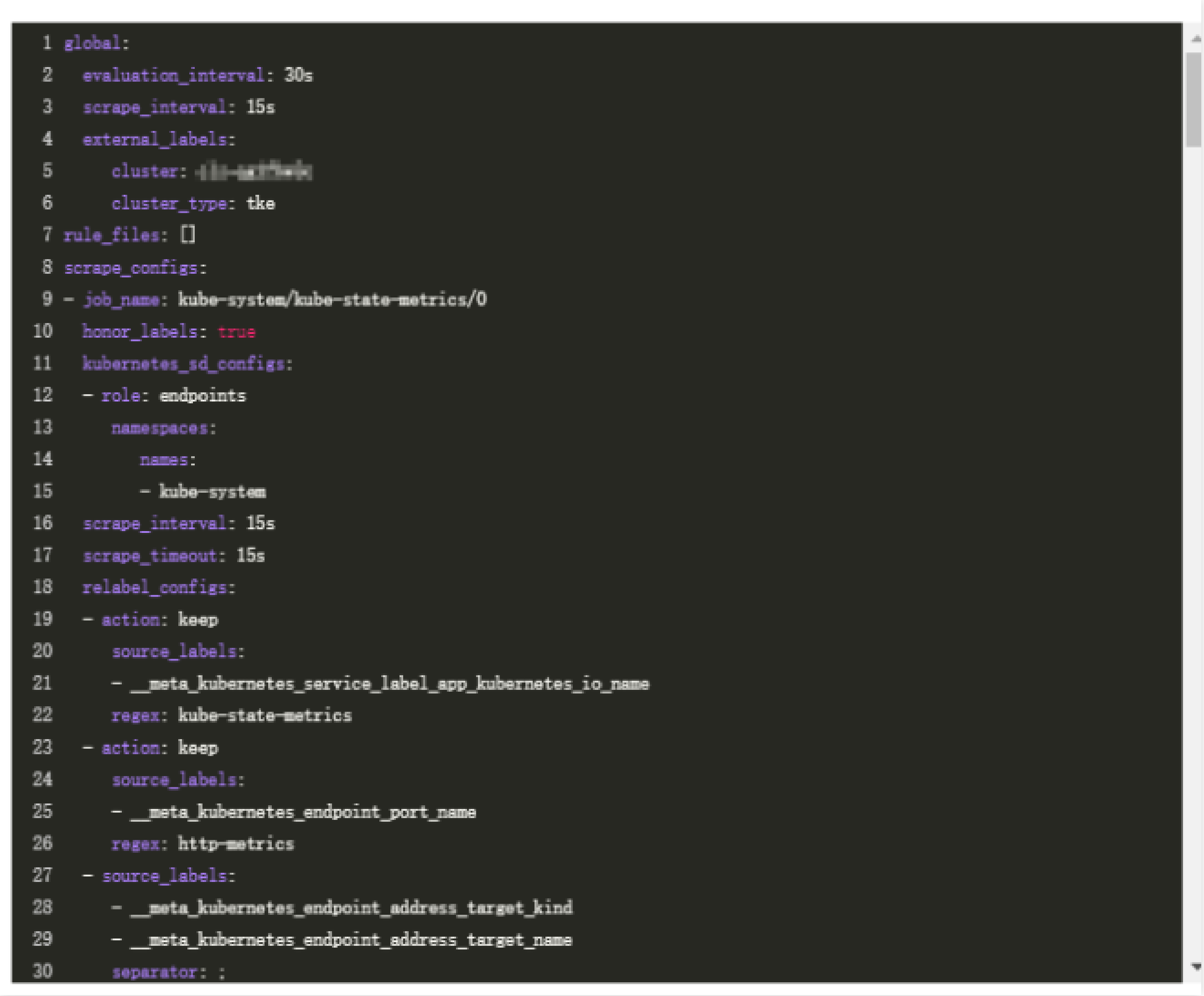
Cluster ID/Name	Region	Cluster type	Global label ⓘ	Agent status	Operation
[redacted]	Singapore	General cluster	[redacted] •••	Running...	Data collection More ▾

Total items: 1

Viewing existing configuration

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.
2. On the instance list page, click **View configuration** in the upper-right corner.

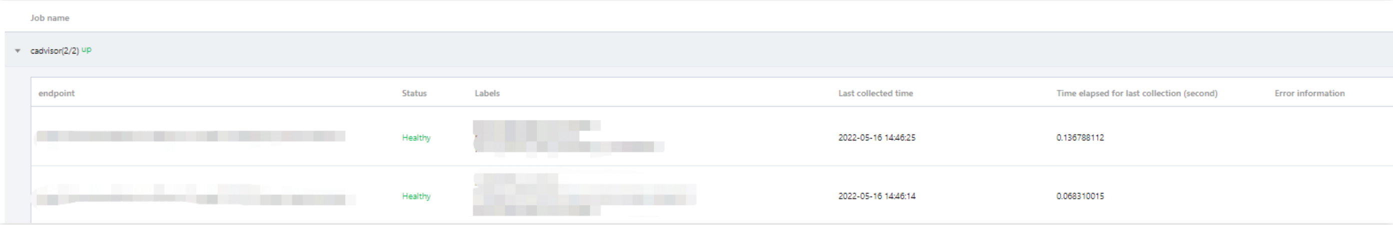
3. In the **View configuration** pop-up window, view all monitoring metrics configured in the yaml file.



```
global:
  evaluation_interval: 30s
  scrape_interval: 15s
  external_labels:
     cluster: [redacted]
     cluster_type: tke
rule_files: []
scrape_configs:
- job_name: kube-system/kube-state-metrics/0
  honor_labels: true
  kubernetes_sd_configs:
  - role: endpoints
     namespaces:
        names:
        - kube-system
  scrape_interval: 15s
  scrape_timeout: 15s
  relabel_configs:
  - action: keep
     source_labels:
     - __meta_kubernetes_service_label_app_kubernetes_io_name
     regex: kube-state-metrics
  - action: keep
     source_labels:
     - __meta_kubernetes_endpoint_port_name
     regex: http-metrics
  - source_labels:
     - __meta_kubernetes_endpoint_address_target_kind
     - __meta_kubernetes_endpoint_address_target_name
     separator: ;
```

Viewing collection targets

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.
2. On the instance list page, select the target instance to enter its details page.
3. On the **Associate with cluster** tab, click **Target Jobs** on the right of the instance.
4. On the targets list page, view the collection status of current data.



Job name

▾ cadvisor(2/2) up

endpoint	Status	Labels	Last collected time	Time elapsed for last collection (second)	Error information
[redacted]	Healthy	[redacted]	2022-05-16 14:46:25	0.136788112	
[redacted]	Healthy	[redacted]	2022-05-16 14:46:14	0.068310015	

- The endpoints in the status of "Unhealthy" are displayed at the top of the list by default.
- You can filter a target by resource attribute on the collection target page.

Related Operations

Mounting file to collector

When configuring the collection item, if you need to provide some files for the configuration, such as a certificate, you can mount the file to the collector in the following way, and the update of the file will be synchronized to the collector in real time.

- **prometheus.tke.tencent.cloud.com/scrape-mount = "true"**
  Add the above label to the configmap under the `prom-xxx` namespace, and all the keys will be mounted to the collector path `/etc/prometheus/configmaps/[configmap-name]/` .

- **prometheus.tke.tencent.cloud.com/scrape-mount = "true"**
  Add the above label to the secret under the `prom-xxx` namespace, and all the keys will be mounted to the collector path `/etc/prometheus/secrets/[secret-name]/` .

Streamlining Monitoring Metrics

Last updated：2022-06-10 19:32:52

Note
To provide better and more powerful product capabilities, TPS will be merged and upgraded into Tencent Managed Service for Prometheus (TMP). The new TMP service supports cross-region and cross-VPC monitoring and connecting a unified Grafana dashboard to multiple TMP instances for data display in one place. For more information on TMP billing, see Pay-as-You-Go. For Tencent Cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.

TPS will be deactivated on May 16, 2022. For more information, see Announcements. Click here to try out the launched TMP service. TPS instances can no longer be created. You can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency first; otherwise, higher costs may be incurred.

Overview

This document describes how to streamline the TPS **collection metrics** to avoid unnecessary expenses after the migration to TMP.

Prerequisites

Before configuring monitoring collection items, you need to perform the following operations:

- You have logged in to the TKE console and created a self-deployed cluster.
- You have created a TMP instance in the VPC of the cluster.

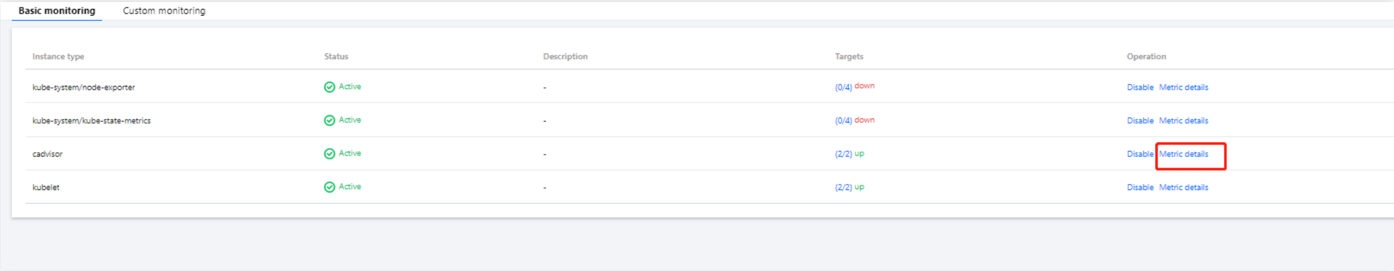
Streamlining Metrics

Streamlining metrics in the console

TMP offers more than 100 free basic monitoring metrics as listed in Free Metrics in Pay-as-You-Go Mode.

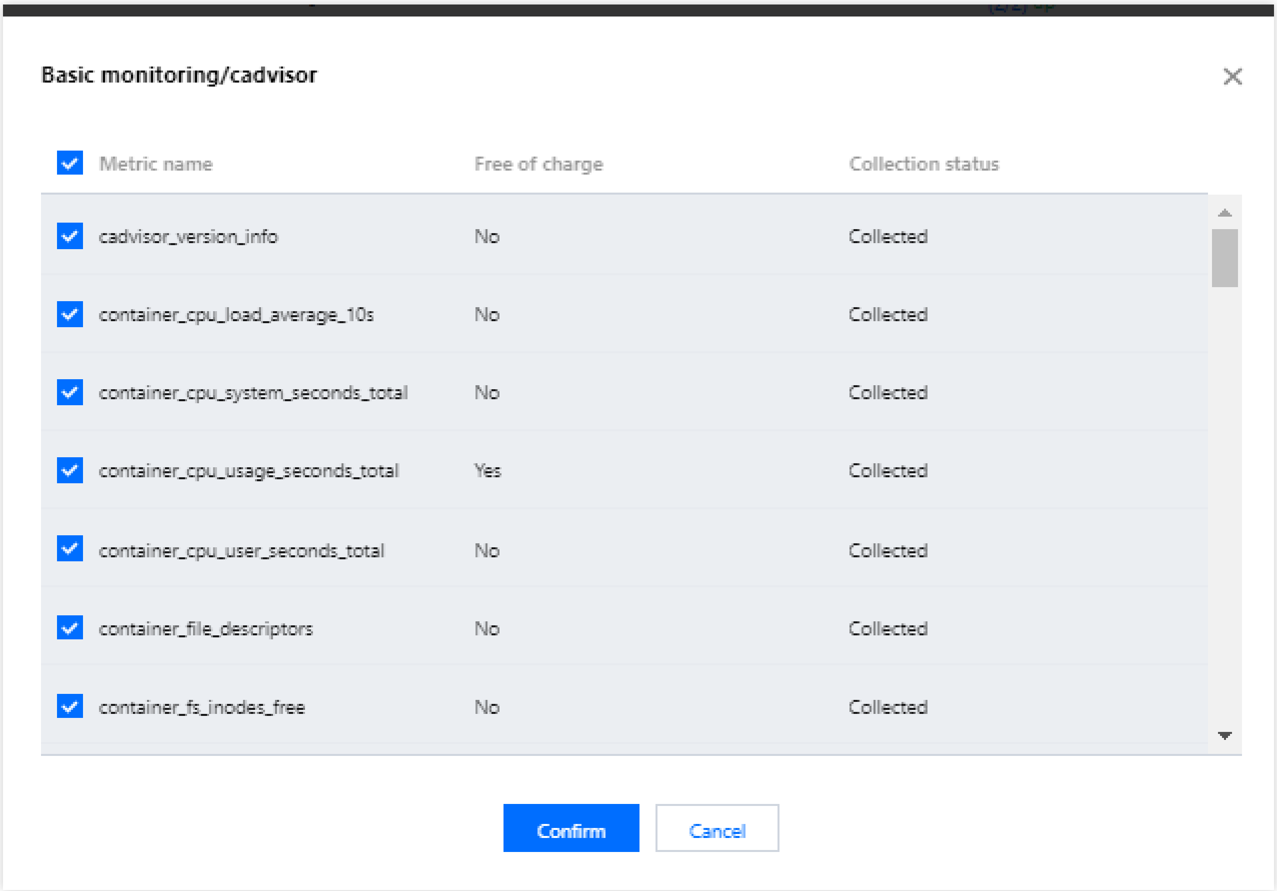
1. Log in to the TKE console and select **Cloud Native Monitoring** on the left sidebar.
2. On the instance list page, select the target instance to enter its details page.

3. On the **Associate with Cluster** page, click **Data Collection Configuration** on the right of the cluster to enter the collection configuration list page.
4. You can add or remove targets of basic metrics on the productized page. Click **Metric Details** on the right.



Instance type	Status	Description	Targets	Operation
kube-system/node-exporter	Active	-	(0/4) down	Disable Metric details
kube-system/kube-state-metrics	Active	-	(0/4) down	Disable Metric details
cadvisor	Active	-	(2/2) up	Disable Metric details
kubelet	Active	-	(2/2) up	Disable Metric details

5. The following shows whether the metrics are free. If you select a metric, it will be collected. We recommend you deselect paid metrics to avoid additional costs after the migration to TMP. Only metrics for basic monitoring are free of charge. For more information on free metrics, see Free Metrics in Pay-as-You-Go Mode. For more information on paid metrics, see Pay-as-You-Go.



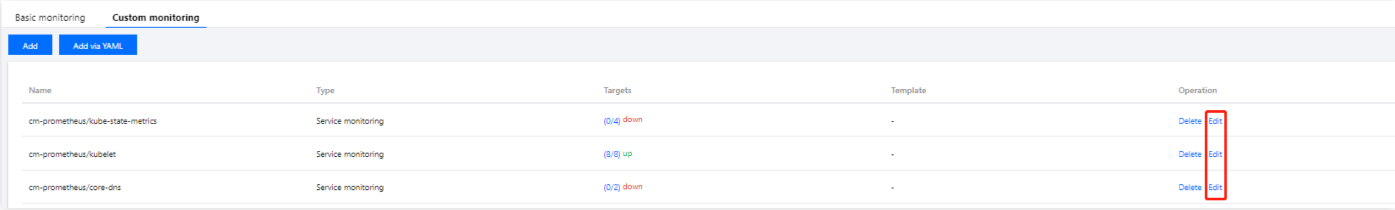
☑ Metric name	Free of charge	Collection status
☑ cadvisor_version_info	No	Collected
☑ container_cpu_load_average_10s	No	Collected
☑ container_cpu_system_seconds_total	No	Collected
☑ container_cpu_usage_seconds_total	Yes	Collected
☑ container_cpu_user_seconds_total	No	Collected
☑ container_file_descriptors	No	Collected
☑ container_fs_inodes_free	No	Collected

Streamlining metrics through YAML

Currently, TMP is billed by the number of monitoring data points. We recommend you optimize your collection configuration to collect only required metrics and filter out unnecessary ones. This will save costs and reduce the overall reported data volume. For more information on the billing mode and Tencent Cloud resource usage, see here.

The following describes how to add filters for ServiceMonitors, PodMonitors, and RawJobs to streamline custom metrics.

1. Log in to the TKE console and select **Cloud Native Monitoring** on the left sidebar.
2. On the instance list page, select the target instance to enter its details page.
3. On the **Associate with Cluster** page, click **Data Collection Configuration** on the right of the cluster to enter the collection configuration list page.
4. Click **Edit** as shown below:



Name	Type	Targets	Template	Operation
cm-prometheus/kube-state-metrics	Service monitoring	(0/4) down	-	Delete Edit
cm-prometheus/kubelet	Service monitoring	(8/8) up	-	Delete Edit
cm-prometheus/core-dns	Service monitoring	(0/2) down	-	Delete Edit

ServiceMonitor and PodMonitor

A ServiceMonitor and a PodMonitor use the same filtering fields, and this document uses a ServiceMonitor as an example.
Sample for ServiceMonitor:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
labels:
app.kubernetes.io/name: kube-state-metrics
app.kubernetes.io/version: 1.9.7
name: kube-state-metrics
namespace: kube-system
spec:
endpoints:
- bearerTokenSecret:
key: ""
interval: 15s # This parameter is the collection frequency. You can increase it to reduce the data storage costs. For example, you can set it to `300s` for less important metrics, which can reduce the amount of monitoring data collected by 20 times.
port: http-metrics
scrapeTimeout: 15s
jobLabel: app.kubernetes.io/name
namespaceSelector: {}
selector:
```

```
matchLabels:
app.kubernetes.io/name: kube-state-metrics
```

To collect `kube_node_info` and `kube_node_role` metrics, you need to add the `metricRelabelings` field to the Endpoint list of the ServiceMonitor. Note that it is **`metricRelabelings`** but not `relabelings`.

Sample for adding `metricRelabelings`:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
labels:
app.kubernetes.io/name: kube-state-metrics
app.kubernetes.io/version: 1.9.7
name: kube-state-metrics
namespace: kube-system
spec:
endpoints:
- bearerTokenSecret:
key: ""
interval: 15s # This parameter is the collection frequency. You can increase it to reduce the data storage costs. For example, you can set it to `300s` for less important metrics, which can reduce the amount of monitoring data collected by 20 times.
port: http-metrics
scrapeTimeout: 15s # This parameter is the collection timeout period. TMP configuration requires that this value not exceed the collection interval, i.e., `scrapeTimeout` <= `interval`.
# The following four lines are added:
metricRelabelings: # Each collected item is subject to the following processing.
- sourceLabels: ["__name__"] # The name of the label to be detected. `__name__` indicates the name of the metric or any label that comes with the item.
regex: kube_node_info|kube_node_role # Whether the above label satisfies this regex. Here, `__name__` should satisfy the requirements of `kube_node_info` or `kube_node_role`.
action: keep # Keep the item if it meets the above conditions; otherwise, drop it.
jobLabel: app.kubernetes.io/name
namespaceSelector: {}
selector:
```

RawJob

If Prometheus' RawJob is used, see the following method for metric filtering.

Sample job:

```
scrape_configs:
- job_name: job1
scrape_interval: 15s # This parameter is the collection frequency. You can increase it to reduce the data storage costs. For example, you can set it to `300s` for less important metrics, which can reduce the amount of monitoring data collected by 20 times.
static_configs:
- targets:
- '1.1.1.1'
```

If you only need to collect `kube_node_info` and `kube_node_role` metrics, add the `metric_relabel_configs` field. Note that it is **`metric_relabel_configs`** but not `relabel_configs`.

Sample for adding `metric_relabel_configs`:

```
scrape_configs:
- job_name: job1
scrape_interval: 15s # This parameter is the collection frequency. You can increase it to reduce the data storage costs. For example, you can set it to `300s` for less important metrics, which can reduce the amount of monitoring data collected by 20 times.
static_configs:
- targets:
- '1.1.1.1'
# The following four lines are added:
metric_relabel_configs: # Each collected item is subject to the following processing.
- source_labels: ["__name__"] # The name of the label to be detected. `__name__` indicates the name of the metric or any label that comes with the item.
regex: kube_node_info|kube_node_role # Whether the above label satisfies this regex. Here, `__name__` should satisfy the requirements of `kube_node_info` or `kube_node_role`.
action: keep # Keep the item if it meets the above conditions; otherwise, drop it.
```

Blocking certain targets

Blocking the monitoring of the entire namespace

TPS will manage all the ServiceMonitors and PodMonitors in a cluster by default after the cluster is associated. If you want to block the monitoring of a namespace, you can label it with `tps-skip-monitor: "true"` as instructed in Labels and Selectors.

Blocking certain targets

TPS collects monitoring data by creating CRD resources of ServiceMonitor and PodMonitor types in your cluster. If you want to block the collection of the specified ServiceMonitor and PodMonitor resources, you can add the label of `tps-skip-monitor: "true"` to these CRD resources as instructed in Labels and Selectors.

Creating Aggregation Rules

Last updated：2022-06-10 16:48:44

Note
To provide better and more powerful product capabilities, TPS will be merged and upgraded into Tencent Managed Service for Prometheus (TMP). The new TMP service supports cross-region and cross-VPC monitoring and connecting a unified Grafana dashboard to multiple TMP instances for data display in one place. For more information on TMP billing, see Pay-as-You-Go. For cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.

TPS will be deactivated soon. Click here to try out the recently launched TMP service. TPS no longer supports instance creation, and you can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency; otherwise, higher costs may be incurred.

Overview

This document describes how to configure aggregation rules to improve query efficiency when dealing with complex query scenarios.

Prerequisites

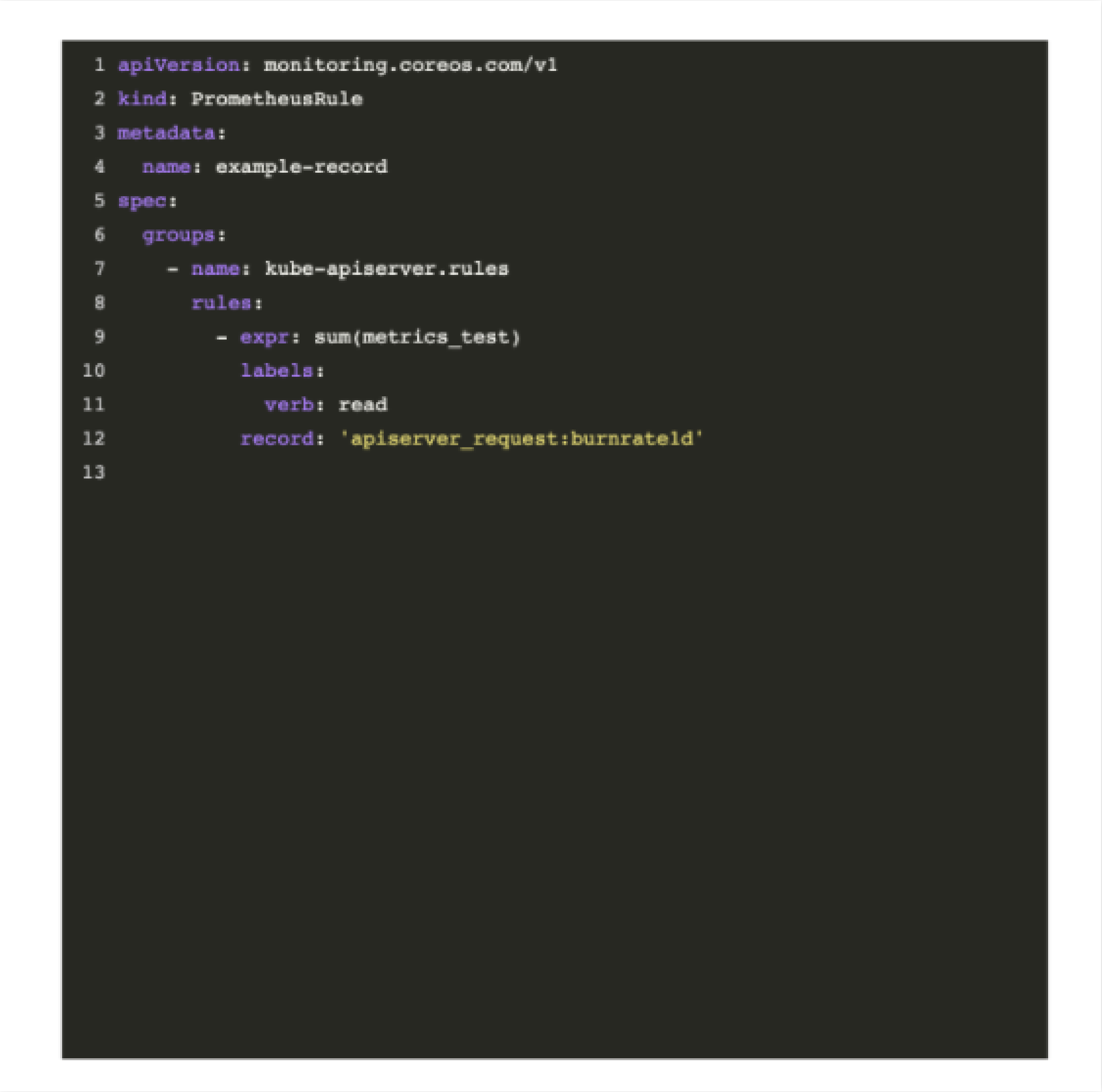
Before configuring aggregation rules, you need to perform the following operations:

- You have logged in to the TKE console and created a self-deployed cluster.
- You have created a TMP instance in the VPC of the cluster.

Directions

1. Log in to the TKE console and click **Cloud Native Monitoring** on the left sidebar.
2. On the instance list page, select the target instance to enter its details page.
3. On the **Aggregation Rule** page, click **Create Aggregation Rule** .

4. In the **Add Aggregation Rule** pop-up window, edit the aggregation rule as shown below:



```yaml
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: example-record
spec:
  groups:
    - name: kube-apiserver.rules
      rules:
        - expr: sum(metrics_test)
          labels:
            verb: read
          record: 'apiserver_request:burnrate1d'

```

5. Click **OK**.

Alarm Configurations

Last updated：2022-05-16 16:02:28

Note
Tencent Prometheus Service (TPS) has been integrated into TMP, which supports cross-region monitoring in multiple VPCs, and provides a unified Grafana dashboard, allowing for checking of multiple monitoring instances. For more information on TMP billing, see Pay-as-You-Go. For cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.

TPS will be discontinued on May 16, 2022,see Announcements. Click here to try out TMP. TPS instances can no longer be created, but you can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency; otherwise, higher costs may be incurred.

Overview

This document describes how to configure alarm policies in cloud native monitoring.

Prerequisites

Before configuring alarm policies, you need to perform the following operations:

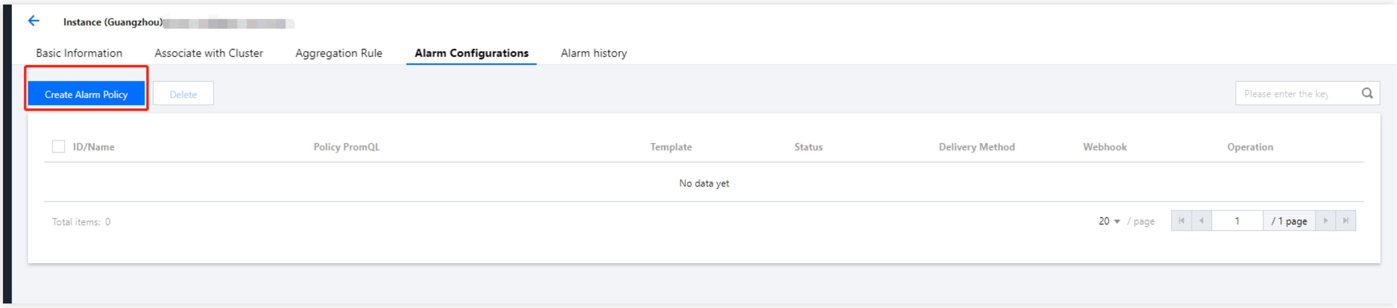
- You've created a TMP instance.
- The cluster to be monitored has been associated with the corresponding instance as instructed in Associating with Cluster.
- The information that needs to be collected has been added to the data collection configuration of the cluster.

Directions

Configuring alarm policies

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.
2. On the instance list page, select the target instance to enter its details page.

3. On the **Alarm configurations** page, click **Create alarm policy**.



Instance (Guangzhou)[redacted]

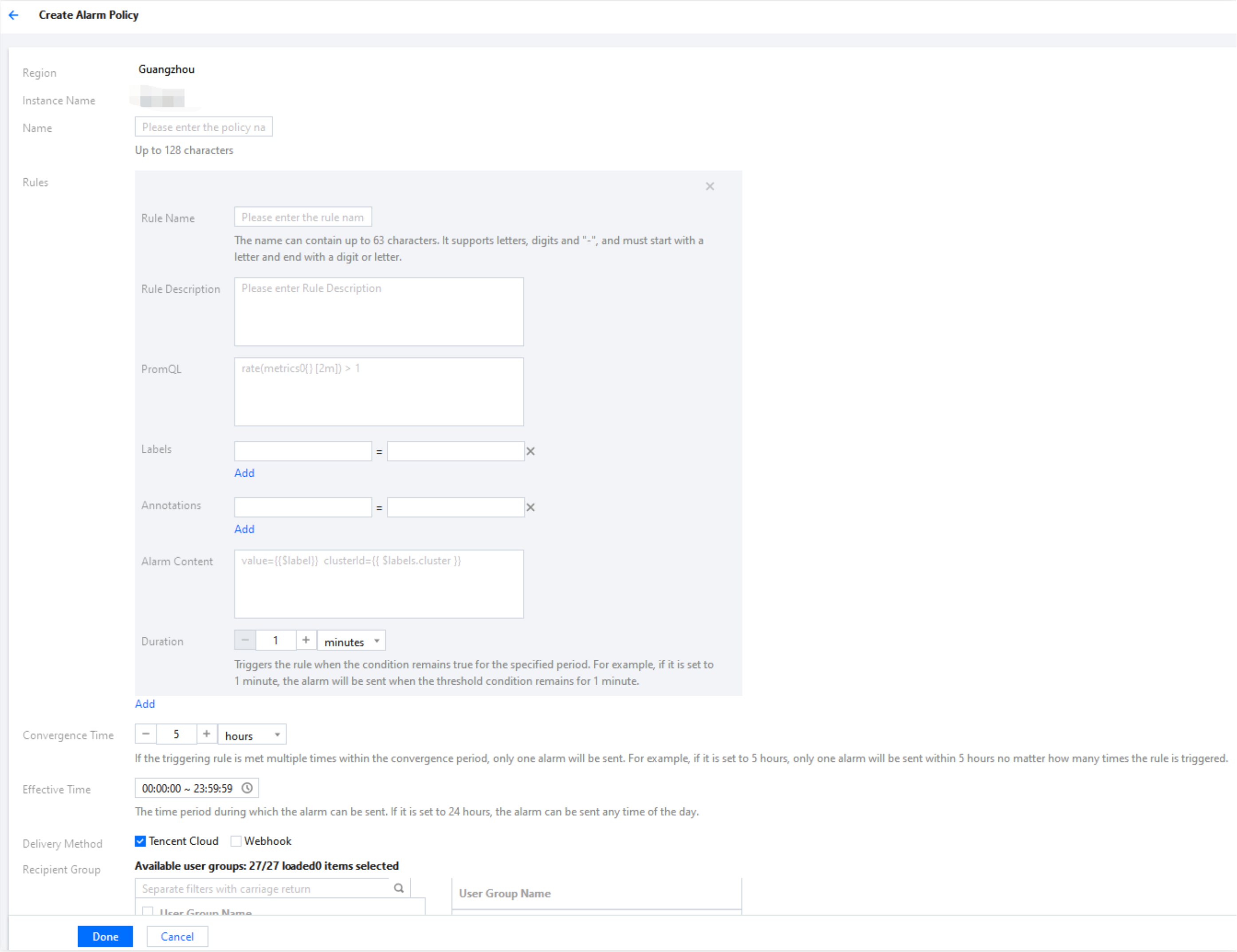
Basic Information | Associate with Cluster | Aggregation Rule | Alarm Configurations | Alarm history

Create Alarm Policy | Delete | Please enter the key

ID/Name	Policy PromQL	Template	Status	Delivery Method	Webhook	Operation
No data yet						

Total items: 0 | 20 / page | 1 / 1 page

4. On the **Create alarm policy** page, add the details of the alarm policy.



Create Alarm Policy

Region: Guangzhou

Instance Name: [redacted]

Name: Please enter the policy na

Up to 128 characters

Rules

Rule Name: Please enter the rule nam

The name can contain up to 63 characters. It supports letters, digits and "-", and must start with a letter and end with a digit or letter.

Rule Description: Please enter Rule Description

PromQL: rate(metrics0{} [2m]) > 1

Labels: =

Add

Annotations: =

Add

Alarm Content: value={{$label}} clusterId={{ $labels.cluster }}

Duration: 1 minutes

Triggers the rule when the condition remains true for the specified period. For example, if it is set to 1 minute, the alarm will be sent when the threshold condition remains for 1 minute.

Add

Convergence Time: 5 hours

If the triggering rule is met multiple times within the convergence period, only one alarm will be sent. For example, if it is set to 5 hours, only one alarm will be sent within 5 hours no matter how many times the rule is triggered.

Effective Time: 00:00:00 ~ 23:59:59

The time period during which the alarm can be sent. If it is set to 24 hours, the alarm can be sent any time of the day.

Delivery Method: ☑ Tencent Cloud □ Webhook

Recipient Group: Available user groups: 27/27 loaded0 items selected

Separate filters with carriage return

□ User Group Name | User Group Name

Done | Cancel

- **Rule name**: Name of alarm rule (up to 40 characters).
- **PromQL**: Alarm rule statement.
- **Duration**: When the condition described in the above statement reaches the duration specified here, an alarm will be triggered.

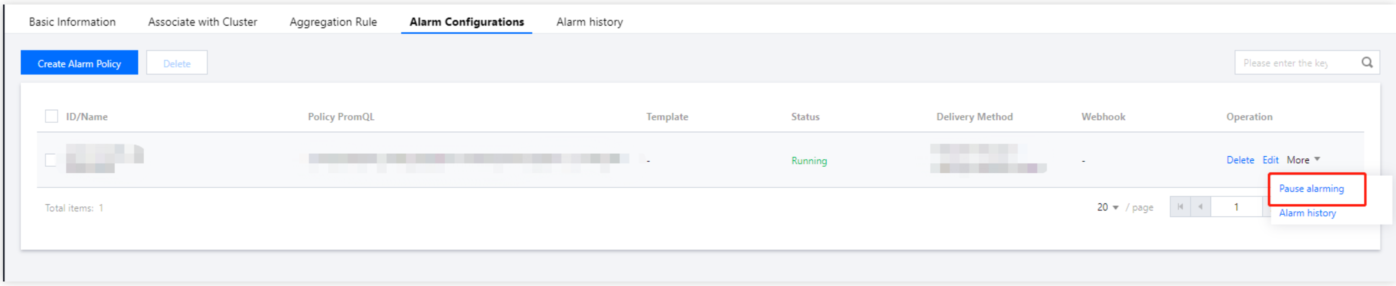
- **Label**: Prometheus labels of each rule.
- **Alarm content**: The alarm notifications to be sent to recipients through email and SMS when an alarm is triggered.
- **Convergence time**: In this specified period, if the alarm condition is met multiple times, only one notification is sent.
- **Effective time**: Alarm notifications can only be sent during the effective period.
- **Recipient group**: Notifications will be sent to the specified contact group.
- **Delivery method**: The delivery channel of alarm notifications.

6. Click **Complete**.

> Note：
>
> The alarm policy will take effect by default once created.

Pausing alarming

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.
2. On the instance list page, select the target instance to enter its details page.
3. On the **Alarm configurations** page, click **More** > **Pause alarming** on the right side of the instance.



ID/Name	Policy PromQL	Template	Status	Delivery Method	Webhook	Operation
[redacted]	[redacted]	-	Running	[redacted]	-	Delete Edit More

Total items: 1

4. In the **Disable alarm policy** pop-up window, click **OK**.

Alarm History

Last updated：2022-05-16 15:52:50

> Note
>
> Tencent Prometheus Service (TPS) has been integrated into TMP, which supports cross-region monitoring in multiple VPCs, and provides a unified Grafana dashboard, allowing for checking of multiple monitoring instances. For more information on TMP billing, see Pay-as-You-Go. For cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.
>
> TPS will be discontinued on May 16, 2022,see Announcements. Click here to try out TMP. TPS instances can no longer be created, but you can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency; otherwise, higher costs may be incurred.

Overview

This document describes how to query the alarm history in cloud native monitoring.

Prerequisites

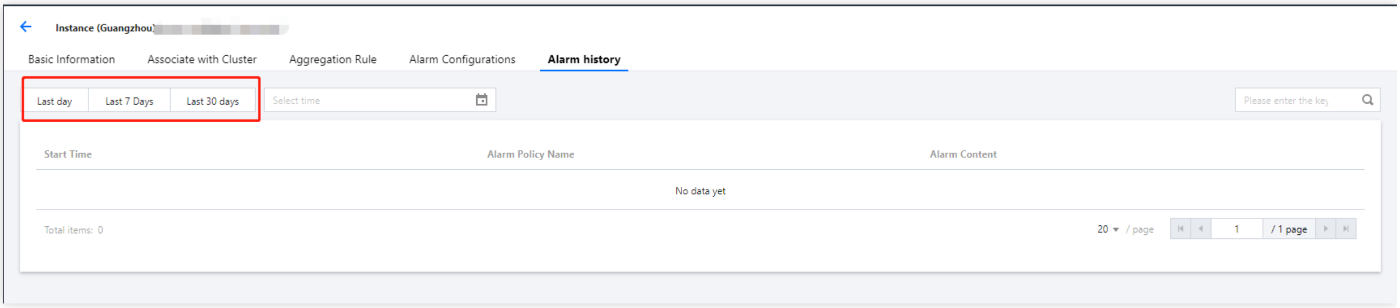
Before querying alarm history, you need to perform the following operations:

- You've created a TMP instance.
- The cluster to be monitored has been associated with the corresponding instance as instructed in Associating with Cluster.
- The information that needs to be collected has been added to the data collection configuration of the cluster.
- Alarm policies are configured. Refer to Configuring Alarm Policy.

Directions

1. Log in to the TKE console and click **Cloud Native Monitoring** in the left sidebar.
2. On the instance list page, select the target instance to enter its details page.

3. On the **Alarm history** page, select a time range to query the alarm history.



Start Time	Alarm Policy Name	Alarm Content
	No data yet	

Total items: 0

Resource Usage of TPS

Last updated：2022-06-22 11:32:31

Note
Tencent Prometheus Service (TPS) has been integrated into TMP, which supports cross-region monitoring in multiple VPCs, and provides a unified Grafana dashboard, allowing for checking of multiple monitoring instances. For more information on TMP billing, see Pay-as-You-Go. For cloud resource usage details, see Billing Mode and Resource Usage. Free metrics for basic monitoring will not be billed.

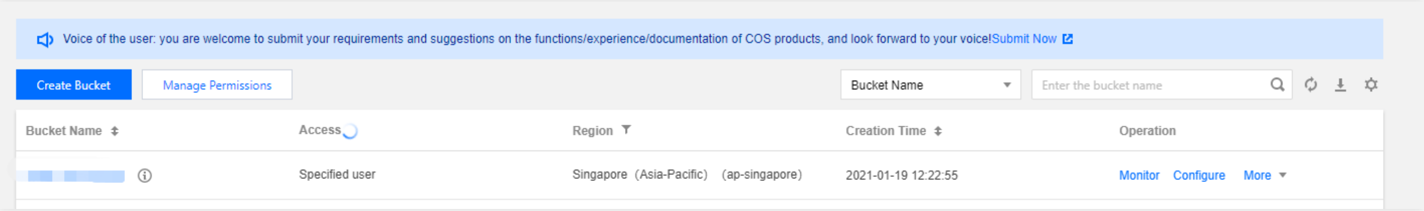
TPS is discontinued from May 16, 2022 (UTC +8). For more information, see Announcement. Click here to try out TMP. TPS instances can no longer be created, but you can use our quick migration tool to migrate your TPS instances to TMP. Before the migration, streamline monitoring metrics or reduce the collection frequency; otherwise, higher costs may be incurred.

TPS is currently in free beta test. When you use TPS, storage resources such as COS and CBS, as well as public and private network CLB resources will be created in your account and billed based on the actual usage. This document describes these resources and their billing methods.

Resource List

COS

The COS will be enabled in your account for persistent storage of metrics when a TPS instance is created. You can view the resource information in the COS console.



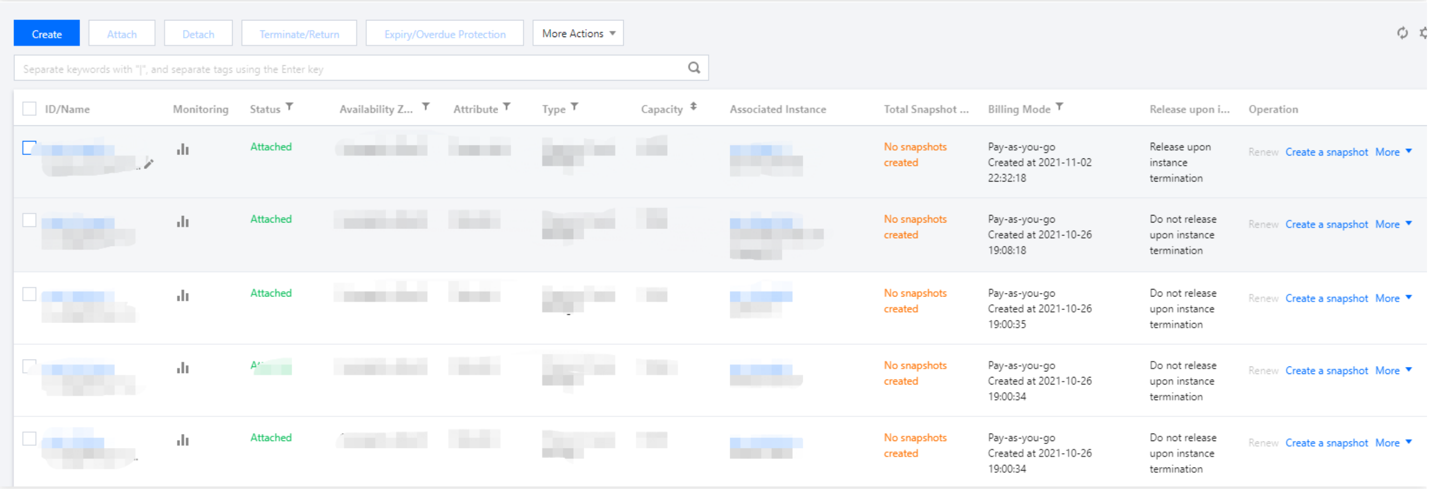
Voice of the user: you are welcome to submit your requirements and suggestions on the functions/experience/documentation of COS products, and look forward to your voice!Submit Now

Bucket Name	Access	Region	Creation Time	Operation
[redacted]	Specified user	Singapore（Asia-Pacific）（ap-singapore）	2021-01-19 12:22:55	Monitor Configure More

This resource is billed based on the actual storage volume and period, which is defined by you at the time of instance creation. For more information about billing for COS, see Pay-as-You-Go.

CBS

Five Premium Cloud Storage will be purchased in your account for temporary storage of metrics when a TPS instance is created. You can view the information about CBS resource and specification in the CBS console.



Create
Attach
Detach
Terminate/Return
Expiry/Overdue Protection
More Actions
Separate keywords with "|", and separate tags using the Enter key

ID/Name	Monitoring	Status	Availability Z...	Attribute	Type	Capacity	Associated Instance	Total Snapshot ...	Billing Mode	Release upon i...	Operation
[redacted]		Attached	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	No snapshots created	Pay-as-you-go Created at 2021-11-02 22:32:18	Release upon instance termination	Renew Create a snapshot More
[redacted]		Attached	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	No snapshots created	Pay-as-you-go Created at 2021-10-26 19:08:18	Do not release upon instance termination	Renew Create a snapshot More
[redacted]		Attached	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	No snapshots created	Pay-as-you-go Created at 2021-10-26 19:00:35	Do not release upon instance termination	Renew Create a snapshot More
[redacted]		A[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	No snapshots created	Pay-as-you-go Created at 2021-10-26 19:00:34	Do not release upon instance termination	Renew Create a snapshot More
[redacted]		Attached	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	No snapshots created	Pay-as-you-go Created at 2021-10-26 19:00:34	Do not release upon instance termination	Renew Create a snapshot More

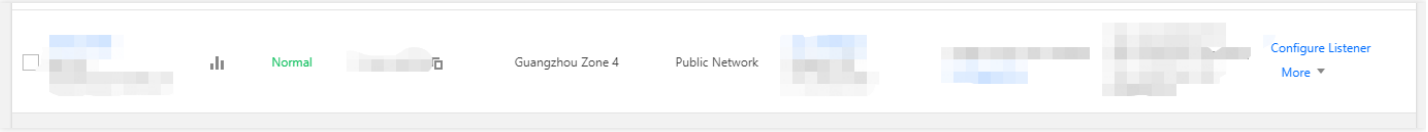
Where:

- Specification of the CBS used for Grafana is 10 GB.
- Specification of the CBS used for Thanos Rule component is 50 GB.
- Specification of the CBS used for Thanos Store component is 200 GB.
- Specification of the CBS used for AlertManager is 10 GB.
- Specification of the CBS used for Prometheus is varied depending on the actual metrics. 10 GB is used for about 30w series (about 30 nodes).

This resource is billed based on the actual usage. For billing details of CBS, see Price Overview.

CLB

Two private network LBs will be created in your account when a TPS instance is created. One LB will be added when one more cluster is associated. If you want to access Grafana via the public network, you need to create a corresponding public network LB. This resource is charged. You can view the information about the LBs in the CLB console.



[redacted]		Normal	[redacted]	Guangzhou Zone 4	Public Network	[redacted]	[redacted]	[redacted]	[redacted]	Configure Listener More

This resource is billed based on the actual usage. For billing details of CLB, see Billing for Standard Account.

Resource Termination

You cannot delete these resources in the corresponding consoles. You need to terminate the TPS instances in the TPS console to terminate all these resources. Tencent Cloud does not repossess TPS instances proactively. If you do not use TPS anymore, you need to delete the TPS instances immediately to avoid additional charges.

Remote Terminals

Remote Terminal Overview

Last updated：2019-08-06 10:36:07

Remote Terminal help users debug their containers quickly and connect to the containers for troubleshooting. It supports file copy/paste/upload/download operations, and can also solve the problem regarding long container login paths and reduce the difficulty in debugging.

Help Topics

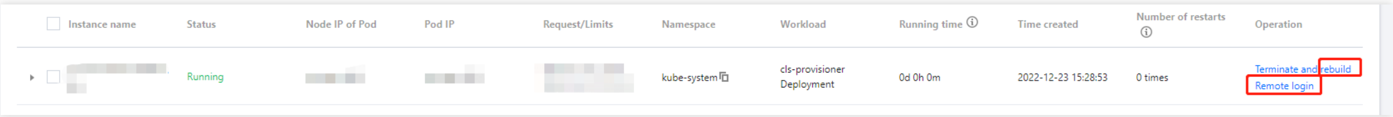
- Basic Operations of Remote Terminal
- Other Container Login Methods

Basic Remote Terminal Operations

Last updated：2023-02-02 17:05:22

Connecting to a Container through Remote Terminal

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the cluster ID (cls-xxx) to go to the cluster details page.
3. In the left sidebar, select **Node Management** > **Node**. On the **Node List** page, click the node ID to go to the Pod management page.
4. In the instance list, click **Remote login** in the **Operation** column of the instance.



□ Instance name	Status	Node IP of Pod	Pod IP	Request/Limits	Namespace	Workload	Running time ⓘ	Time created	Number of restarts ⓘ	Operation
□ [redacted]	Running	[redacted]	[redacted]	[redacted]	kube-system	cls-provisioner Deployment	0d 0h 0m	2022-12-23 15:28:53	0 times	Terminate and rebuild Remote login

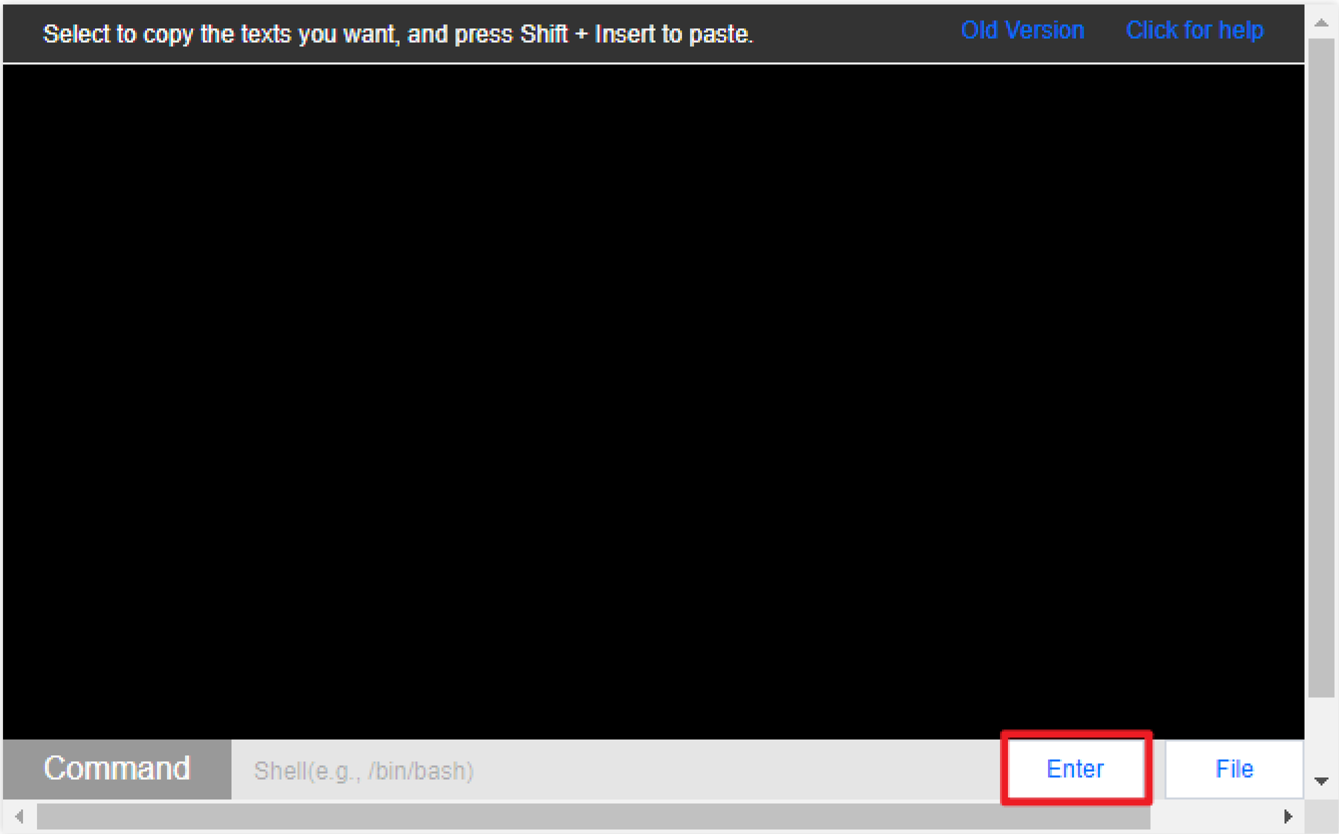
> Note
>
> Containers meeting any of the following conditions do not support remote login:
>
> - The namespace is kube-system.
> - A bash is not built in the container image.
>
>   For FAQs about the remote terminal, see here.

5. In the container login pop-up window, select shell and select **Login** on the right side of the container you want to log in.

Running Commands for Containers Without Shell

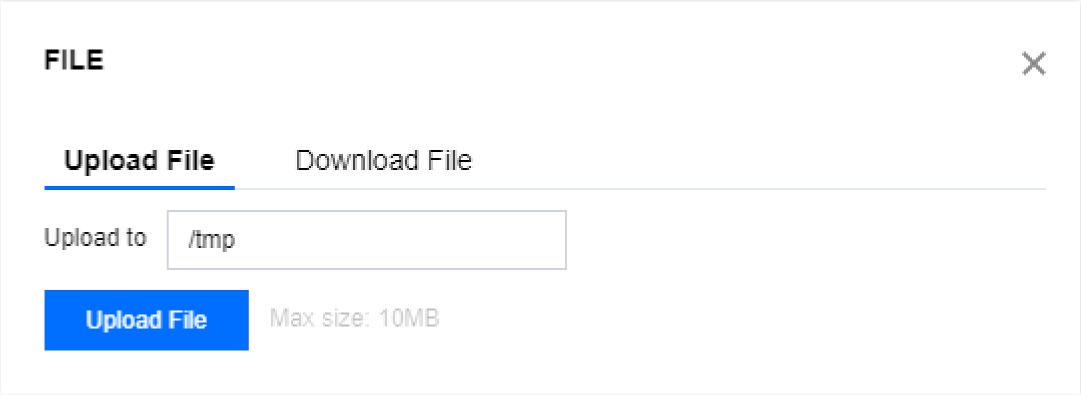
1. Go to the remote terminal page.

2. Enter the command to be run below and click **Enter**, as shown in the following figure:


Select to copy the texts you want, and press Shift + Insert to paste.
Old Version
Click for help
Command
Shell(e.g., /bin/bash)
Enter
File

Uploading and Downloading Files

1. Go to the remote terminal page.
2. Click **File** and select **Upload File** or **Download File**.



- **Upload File**: Specify the directory to which the files are to be uploaded.
- **Download File**: Specify the path of the files to be downloaded.

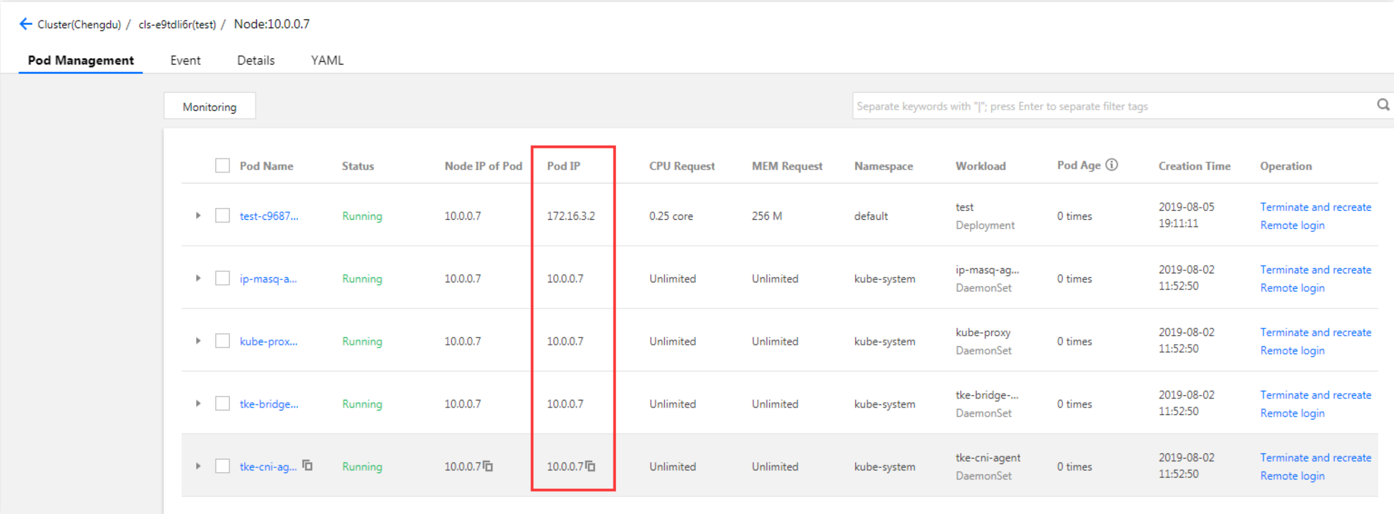
Other Login Methods

Last updated：2023-02-03 14:54:24

Logging In to the Container Through SSH

If the SSH server is installed on your container, you can log in to the container through SSH.

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the cluster ID (cls-xxx) to go to the cluster details page.
3. On the cluster details page, select **Node Management** > **Node** in the left sidebar.
4. On the **Node List** page, click the node name to go to the Pod management page.
5. In the instance list, obtain the IP address of the instance, as shown below:



Cluster(Chengdu) / cls-e9tdli6r(test) / Node:10.0.0.7

Pod Management
Event
Details
YAML
Monitoring
Separate keywords with "|"; press Enter to separate filter tags

Pod Name	Status	Node IP of Pod	Pod IP	CPU Request	MEM Request	Namespace	Workload	Pod Age	Creation Time	Operation
test-c9687...	Running	10.0.0.7	172.16.3.2	0.25 core	256 M	default	test Deployment	0 times	2019-08-05 19:11:11	Terminate and recreate Remote login
ip-masq-a...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	ip-masq-ag... DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login
kube-prox...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	kube-proxy DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login
tke-bridge...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	tke-bridge-... DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login
tke-cni-ag...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	tke-cni-agent DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login

6. Log in to any node in the cluster. For more information, see here.
7. Log in to the container through SSH.

Logging In to a Container Through the Container's Node

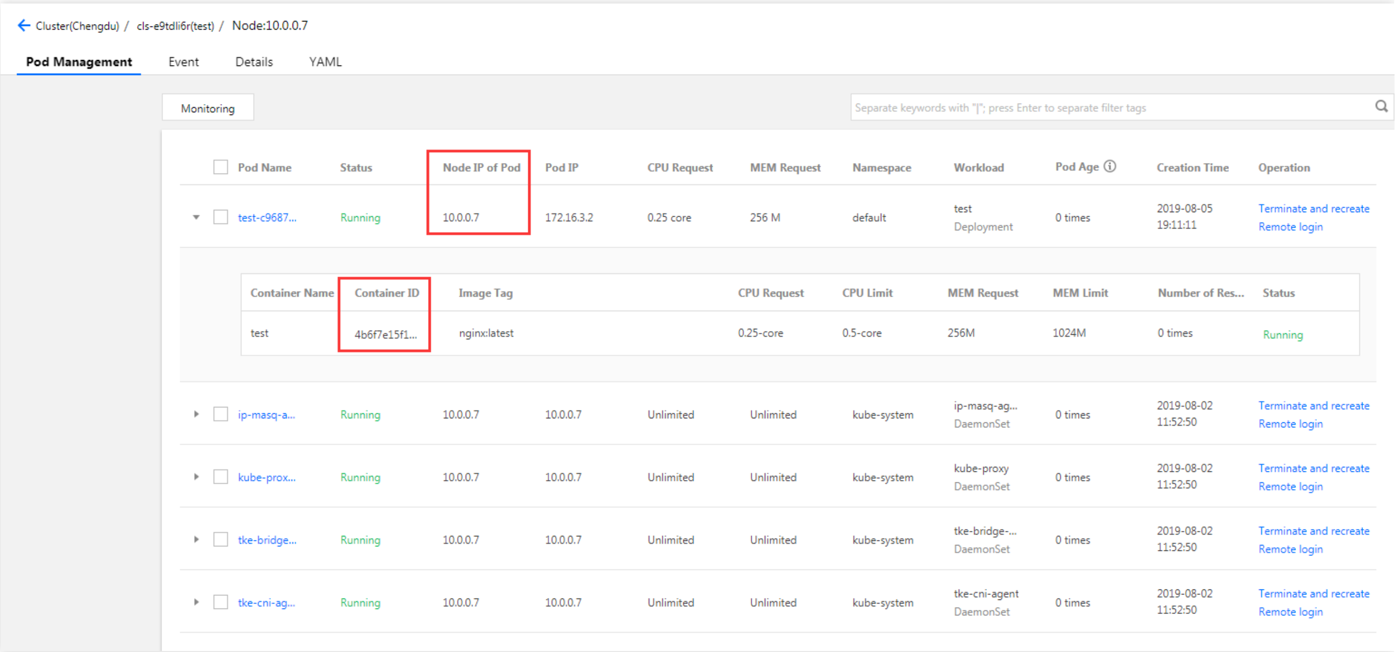
1. Log in to the TKE console and select **Cluster** in the left sidebar.

2. On the **Cluster Management** page, click the cluster ID (cls-xxx) to go to the cluster details page.

3. On the cluster details page, select **Node Management** > **Node** in the left sidebar.

4. On the **Node List** page, click the node name to go to the Pod management page.

5. In the instance list, obtain the IP address of the node to which the container belongs and the container ID.


Cluster(Chengdu) / cls-e9tdli6r(test) / Node:10.0.0.7
Pod Management
Event
Details
YAML
Monitoring
Separate keywords with "|"; press Enter to separate filter tags

Pod Name	Status	Node IP of Pod	Pod IP	CPU Request	MEM Request	Namespace	Workload	Pod Age	Creation Time	Operation
test-c9687...	Running	10.0.0.7	172.16.3.2	0.25 core	256 M	default	test Deployment	0 times	2019-08-05 19:11:11	Terminate and recreate Remote login
ip-masq-a...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	ip-masq-ag... DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login
kube-prox...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	kube-proxy DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login
tke-bridge...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	tke-bridge-... DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login
tke-cni-ag...	Running	10.0.0.7	10.0.0.7	Unlimited	Unlimited	kube-system	tke-cni-agent DaemonSet	0 times	2019-08-02 11:52:50	Terminate and recreate Remote login

Container Name	Container ID	Image Tag	CPU Request	CPU Limit	MEM Request	MEM Limit	Number of Res...	Status
test	4b6f7e15f1...	nginx:latest	0.25-core	0.5-core	256M	1024M	0 times	Running

6. Log in to the node. For more information, see here.

7. Run the `docker ps` command to view the container you want to log in to.

```
[root@VM_88_88_centos ~]# docker ps | grep 75b3b15af61a
75b3b15af61a nginx:latest "nginx -g 'daemon off" About a minute ago Up About a
minute k8s_worid.e8b44cc_worid-24bn2_default_81a59654-aa14-11e6-8a18-52540093c4
0b_42c0b746
```

8. Run the `docker exec` command to log in to the container.

```
[root@VM_0_60_centos ~]# docker ps | grep 75b3b15af61a
75b3b15af61a nginx:latest "nginx -g 'daemon off" 2 minutes ago Up 2 minutes k8s
_worid.e8b44cc_worid-24bn2_default_81a59654-aa14-11e6-8a18-52540093c40b_6b389dd
2
[root@VM_0_60_centos ~]# docker exec -it 75b3b15af61a /bin/bash
root@worid-24bn2:/# ls
bin boot devetc home liblib64 media mnt optproc root run sbin srv sys tmp usr v
ar
```

Policy Management

Last updated：2024-05-08 09:58:32

Overview

Native Kubernetes has a cascade deletion mechanism. If a resource is deleted, other related resources will be automatically deleted. For example, when a namespace is deleted, all the related resources such as pods, services, and ConfigMaps under this namespace will be deleted accordingly, which may cause business disruption.
To solve this problem, TKE provides the policy management module implemented by the Gatekeeper based on the Open Policy Agent (OPA). This function helps you define and execute consistent policies in multiple clusters to gain a serious safe and reliable system.

Policy Description

Policy Classification

Cluster deletion protection: It is not allowed to delete a cluster that still contains working nodes.
Cluster resource deletion protection: It is not allowed to delete the cluster scoped or namespace scoped Kubernetes resource that may cause cascading deletion for other system resources.

Support Boundary

Cluster deletion protection policy: It supports all versions of TKE standard clusters and TKE serverless clusters, but does not support registered clusters and edge clusters.
Cluster resource deletion protection policy: It support kubernetes version 1.16 and later for both TKE standard clusters and TKE serverless clusters, but does not support registered clusters and edge clusters.

Policy Type

Baseline policy: It is mandatory and cannot be disabled.
Preferred policy: It is enabled by default, but can be disabled by the user.
Optional policy: It is disabled by default, but can be enabled by the user.

Policy Library

TKE Policy

Classification	Policy Name	Policy Description	Policy Type

Cluster policy	If there are nodes in the cluster, the cluster cannot be deleted.	If there are regular nodes, native nodes, or registered nodes in the cluster, the nodes must be eliminated before the cluster can be deleted.	Baseline policy
Namespace policy	If there are workloads, services and routes, or storage objects under the namespace, the namespace cannot be deleted.	If there are pods, services, ingresses, and PVCs within the namespace, clear the aforementioned resources before deleting the namespace.	Preferred policy
Configuration-related policy	Disallow deletion if a CRD has associated CR resources	If a CRD defines CR resources, the CR resources must be deleted first before the CRD can be deleted.	Preferred policy

OPA Standard Library Policy

Type	Policy Name	Policy Description	Policy Type
General	k8sallowedrepos	Requires container images to begin with a string from the specified list.	Optional Policy
General	k8spspautomountserviceaccounttokenpod	Controls the ability of any Pod to enable automountServiceAccountToken.	Optional Policy
General	k8sblockendpointeditdefaultrole	Many Kubernetes installations by default have a system:aggregate-to-edit ClusterRole which does not properly restrict access to editing Endpoints. This ConstraintTemplate forbids the system:aggregate-to-edit ClusterRole from granting permission to create/patch/update Endpoints.	Optional Policy
General	k8sblockloadbalancer	Disallows all Services with type LoadBalancer.	Optional Policy
General	k8sblocknodeport	Disallows all Services with type NodePort.	Optional Policy
General	k8sblockwildcardingress	Users should not be able to create Ingresses with a blank or wildcard (*) hostname since that would enable them to intercept traffic for other	Optional Policy

		services in the cluster, even if they don't have access to those services.	
General	k8scontainerlimits	Requires containers to have memory and CPU limits set and constrains limits to be within the specified maximum values.	Optional Policy
General	k8scontainerrequests	Requires containers to have memory and CPU requests set and constrains requests to be within the specified maximum values.	Optional Policy
General	k8scontainerratios	Sets a maximum ratio for container resource limits to requests.	Optional Policy
General	k8srequiredresources	Requires containers to have defined resources set.	Optional Policy
General	k8sdisallowanonymous	Disallows associating ClusterRole and Role resources to the system:anonymous user and system:unauthenticated group.	Optional Policy
General	k8sdisallowedtags	Requires container images to have an image tag different from the ones in the specified list.	Optional Policy
General	k8sexternalips	Restricts Service externalIPs to an allowed list of IP addresses.	Optional Policy
General	k8simagedigests	Requires container images to contain a digest.	Optional Policy
General	noupdateserviceaccount	Blocks updating the service account on resources that abstract over Pods. This policy is ignored in audit mode.	Optional Policy
General	k8sreplicalimits	Requires that objects with the field spec.replicas (Deployments, ReplicaSets, etc.) specify a number of replicas within defined ranges.	Optional Policy
General	k8srequiredannotations	Requires resources to contain specified annotations, with values	Optional Policy

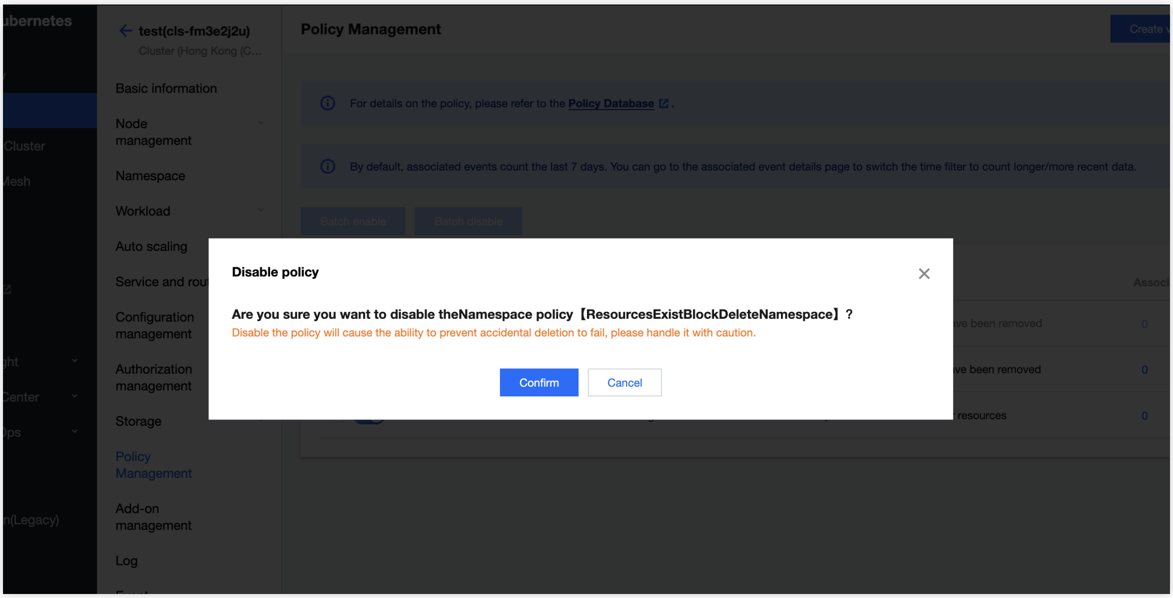
		matching provided regular expressions.	
General	k8srequiredlabels	Requires resources to contain specified labels, with values matching provided regular expressions.	Optional Policy
General	k8srequiredprobes	Requires Pods to have readiness and/or liveness probes.	Optional Policy
Pod Security Policy	k8spspallowprivilegeescalationcontainer	Controls restricting escalation to root privileges. Corresponds to the allowPrivilegeEscalation field in a PodSecurityPolicy.	Optional Policy
Pod Security Policy	k8spspapparmor	Configures an allow-list of AppArmor profiles for use by containers. This corresponds to specific annotations applied to a PodSecurityPolicy.	Optional Policy
Pod Security Policy	k8spspcapabilities	Controls Linux capabilities on containers. Corresponds to the allowedCapabilities and requiredDropCapabilities fields in a PodSecurityPolicy.	Optional Policy
Pod Security Policy	k8spspflexvolumes	Controls the allowlist of FlexVolume drivers. Corresponds to the allowedFlexVolumes field in PodSecurityPolicy.	Optional Policy
Pod Security Policy	k8spspforbiddensysctls	Controls the sysctl profile used by containers. Corresponds to the allowedUnsafeSysctls and forbiddenSysctls fields in a PodSecurityPolicy. When specified, any sysctl not in the allowedSysctls parameter is considered to be forbidden.	Optional Policy
Pod Security Policy	k8spspfsgroup	Controls allocating an FSGroup that owns the Pod's volumes. Corresponds to the fsGroup field in a PodSecurityPolicy.	Optional Policy
Pod Security	k8spsphostfilesystem	Controls usage of the host filesystem. Corresponds to the allowedHostPaths	Optional Policy

Policy		field in a PodSecurityPolicy.	
Pod Security Policy	k8spsphostnamespace	Disallows sharing of host PID and IPC namespaces by pod containers. Corresponds to the hostPID and hostIPC fields in a PodSecurityPolicy.	Optional Policy
Pod Security Policy	k8spsphostnetworkingports	Controls usage of host network namespace by pod containers. Specific ports must be specified. Corresponds to the hostNetwork and hostPorts fields in a PodSecurityPolicy.	Optional Policy
Pod Security Policy	k8spspprivilegedcontainer	Controls the ability of any container to enable privileged mode.	Optional Policy
Pod Security Policy	k8spspprocmount	Controls the allowed procMount types for the container. Corresponds to the allowedProcMountTypes field in a PodSecurityPolicy.	Optional Policy
Pod Security Policy	k8spspreadonlyrootfilesystem	Requires the use of a read-only root file system by pod containers.	Optional Policy
Pod Security Policy	k8spspseccomp	Controls the seccomp profile used by containers.	Optional Policy
Pod Security Policy	k8spspselinuxv2	Defines an allow-list of seLinuxOptions configurations for pod containers.	Optional Policy
Pod Security Policy	k8spspallowedusers	Controls the user and group IDs of the container and some volumes.	Optional Policy
Pod Security Policy	k8spspvolumetypes	Restricts mountable volume types to those specified by the user.	Optional Policy

Operation Description

Enabling/Disabling Policy

1. Log in to the TKE console, and select **Cluster** in the left sidebar.
2. On the cluster management page, select the target cluster ID to enter the basic information page for the cluster.
3. Select **Policy Management** from the left navigation bar to enter the policy management page, select a policy, and click **Enable**/**Disable**. Disabling a policy requires a second confirmation, while enabling it does not. See below:



Policy Management

For details on the policy, please refer to the Policy Database .

By default, associated events count the last 7 days. You can go to the associated event details page to switch the time filter to count longer/more recent data.

Disable policy

Are you sure you want to disable theNamespace policy【ResourcesExistBlockDeleteNamespace】?

Disable the policy will cause the ability to prevent accidental deletion to fail, please handle it with caution.

Verifying Policy Effect

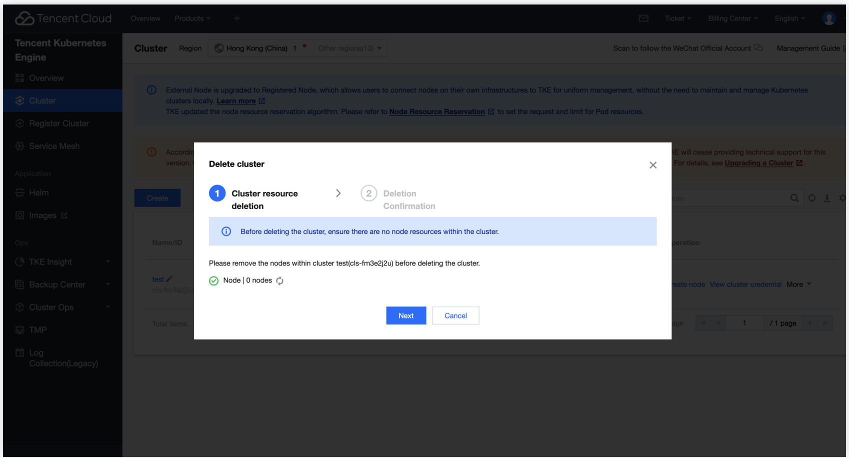
Taking the cluster deletion policy as an example, create a TKE standard cluster and verify whether a deletion request will be intercepted when there are nodes in the cluster.

1. Create a TKE standard cluster with nodes. For detailed steps, see Create Cluster.
2. Initiate a cluster deletion request.

Delete via console

Delete through API

1. Delete the cluster. For detailed steps, see Delete Cluster.
2. A window prompt indicates that nodes must be removed before you proceed with cluster deletion. See below:



External Node is upgraded to Registered Node, which allows users to connect nodes on their own infrastructures to TKE for uniform management, without the need to maintain and manage Kubernetes clusters locally. Learn more

TKE updated the node resource reservation algorithm. Please refer to Node Resource Reservation to set the request and limit for Pod resources.

Delete cluster

1 Cluster resource deletion > 2 Deletion Confirmation

Before deleting the cluster, ensure there are no node resources within the cluster.

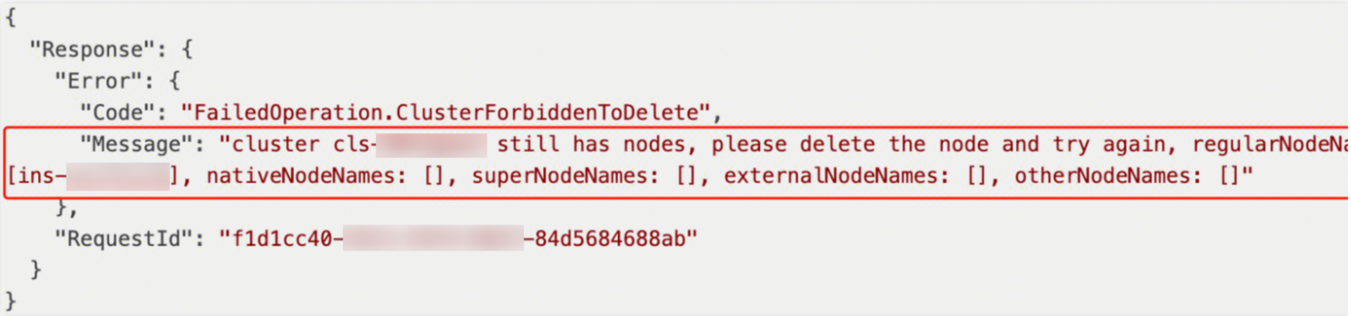
Please remove the nodes within cluster test(cls-fm3e2j2u) before deleting the cluster.

✓ Node | 0 nodes

Next Cancel

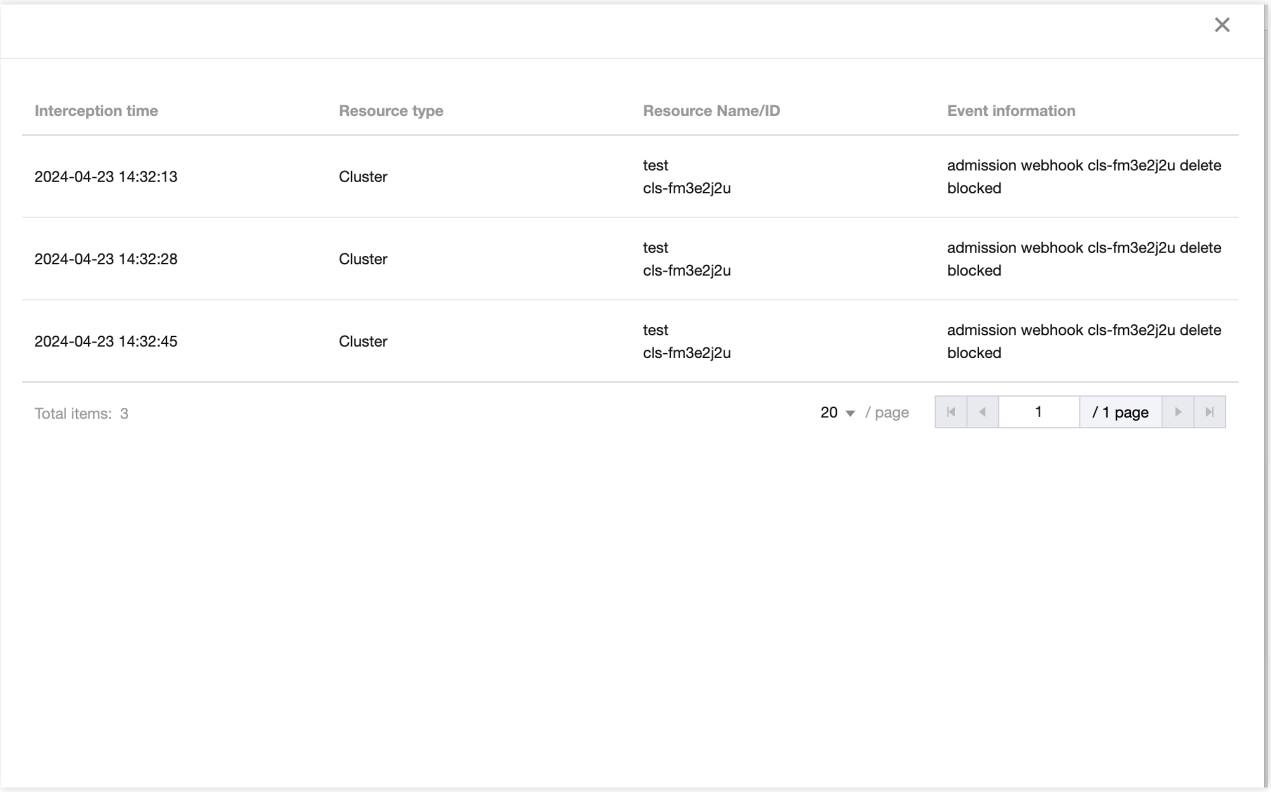
1. Delete the cluster through API. For how to call the API, see the API document Delete Cluster.

2. Calling the API to delete the cluster failed. The error message returned includes a list of existing nodes in the cluster. See below:



```
{
  "Response": {
    "Error": {
      "Code": "FailedOperation.ClusterForbiddenToDelete",
      "Message": "cluster cls-[redacted] still has nodes, please delete the node and try again, regularNodeNa
[ins-[redacted]], nativeNodeNames: [], superNodeNames: [], externalNodeNames: [], otherNodeNames: []"
    },
    "RequestId": "f1d1cc40-[redacted]-84d5684688ab"
  }
}
```

3. On the **Policy Management** page, click the number of related events to view the interception event information. See below:



Interception time	Resource type	Resource Name/ID	Event information
2024-04-23 14:32:13	Cluster	test cls-fm3e2j2u	admission webhook cls-fm3e2j2u delete blocked
2024-04-23 14:32:28	Cluster	test cls-fm3e2j2u	admission webhook cls-fm3e2j2u delete blocked
2024-04-23 14:32:45	Cluster	test cls-fm3e2j2u	admission webhook cls-fm3e2j2u delete blocked

Total items: 3