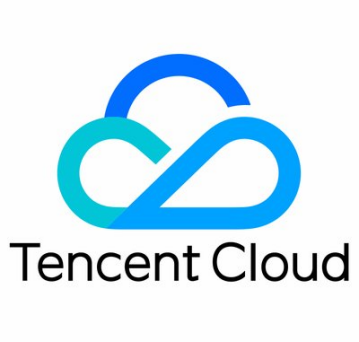
Tencent Kubernetes Engine

TKE General Cluster Guide



Contents

TKE General Cluster Guide
TKE General Cluster Overview

Last updated: 2023-09-26 09:56:16

Introduction

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.
The following video introduces Tencent Kubernetes Engine (TKE), its core concepts, and the usage process:
Watch video

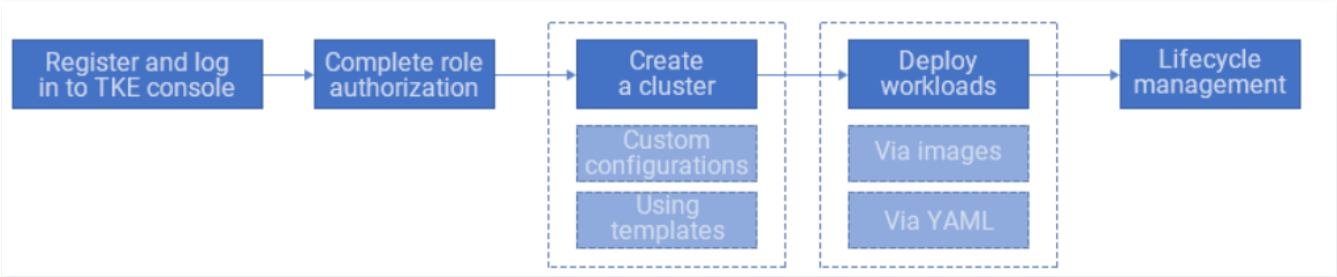
Definitions

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.

How to Use a Security Group

The following figure shows you how to use TKE:



1. Role Authorization
   Register and log in to the TKE console, complete the service authorization to obtain relevant resource operation permissions, and start using the container service product.
2. Create a cluster
   You can customize a new cluster or create a cluster using a template.
3. Deploy Workloads
   Support deployment of workloads using image deployment and YAML file orchestration. For more information, refer to Workload Management.
4. Manage the lifecycle of Pods by performing operations such as monitoring, upgrade, and scaling.

Product 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 the CVM Documentation.
- 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 the VPC Documentation.
- You can use CLB to automatically allocate the request traffic of clients across CVM instances and then forward it to the containers running on the CVM instances. For more information, see the CLB Documentation.
- 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.

High-risk Operations of Container Service

Last updated: 2023-09-15 15:29:05

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

Type	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 component 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 nodes	Modifying the security groups of nodes in a cluster	Node may become unavailable	Configure security groups as recommended by Tencent Cloud
	Modifying Node Instance Specification	Forced shutdown of the server, node unavailability	Remove the node and add it back to the cluster
	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,	Restore the permissions

		such as cloud load balancers, may not be able to be created	

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 through LB console	Modification gets reset by TKE	Automatically create listeners 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	Remarks
Deleting the /tmp/ccs-log-collector/pos directory of the host	Log gets collected again	-	The Pos file contains the record of the file's collection location.
Deleting the /tmp/ccs-log-collector/buffer directory of the host	Log gets lost	-	The buffer contains cached log files waiting to be consumed.

Cloud Disk

High-risk Operation	Consequence	Solution
Manually unmounting cloud disks through console	Pod encounters an I/O error while writing	Delete the mount directory of the node and reschedule the Pod
Unmounting disk mounting path on the node	Pod writing to local disk	Re-mount the corresponding directory onto Pod
Directly operating CBS block device on the node	Pod writing to local disk	-

Deploying Containerized Applications in the Cloud

Last updated: 2023-09-26 09:57:26

Feature 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	Local Disk Types	Impact Description	Reference
Clusters	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 and Node Network Configuration
	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.	VPC Connectivity
	When you create a cluster, 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.	Deployment	Security groups provide an important means of security isolation. Improper security policy configuration may lead to security-related risks, service connectivity issues, and other problems.	TKE Security 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	Deployment	The Master and Etcd of the managed cluster are not user resources and are managed and	• Cluster Overview • Explanation

	managed cluster mode as needed.		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.	of Cluster Management Modes
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 same node due to the lack of resources, and they will report monitoring information errors.	Setting the Resource Limit of Workload
	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 the number of Pods • Setting the Scheduling Rule for a Workload

Data reliability

Category	Item	Local Disk Types	Impact Description	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	Local Disk Types	Impact Description	Reference
Engineering	Check whether the quotas of resources such as CVMs, VPCs,	Deployment	Insufficient quotas will cause resource creation to fail. If you have enabled auto scaling,	• Purchase Cluster

	subnets, and CBS disks can meet customer needs.		ensure that you have sufficient quotas for your Tencent Cloud services.	Quota Limit • Quota Limits
	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.	• Configure Alarms • View Monitoring Data • List of Monitoring and Alarm Metrics

Open Source Components

Last updated: 2023-09-26 09:57:59

tencentcloud-cloud-controller-manager

`tencentcloud-cloud-controller-manager` is the Cloud Controller Manager implementation for the Tencent Kubernetes Engine (TKE). This component is used to implement the following functions on Kubernetes clusters built by Tencent Cloud CVMs:

- nodecontroller: updates relevant `addresses` information for Kubernetes nodes
- routecontroller: responsible for creating routes within the pod IP range in a private network.
- servicecontroller: creates corresponding load balancers when services of type LoadBalancer are created within the cluster.

For more installation and usage instructions, see GitHub tencentcloud-cloud-controller-manager.

kubernetes-csi-tencentcloud

kubernetes-csi-tencentcloud is a CSI-compliant implementation plugin for Tencent Cloud Block Storage services. With this component, you can use cloud block storage in Kubernetes clusters built on Tencent Cloud CVMs.
This plugin is suitable for scenarios where cloud disks are used in self-built Kubernetes clusters, and is distinct from the built-in `provisioner cloud.tencent.com/qcloud-cbs` in container service clusters.
For more installation and usage instructions, see GitHub kubernetes-csi-tencentcloud.

Permission Management Overview

Last updated: 2023-09-15 15:32:08

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:

- Your key will be easily compromised because it is shared by several users.
- Your users might introduce security risks from maloperations due to the lack of user access control.

To address these issues, you can use sub-accounts to delegate management of different services to different users. By default, sub-accounts do not have permissions to use TKE. You need to create policies that grant sub-accounts the necessary permissions for their tasks.

Feature Overview

Cloud Access Management (CAM) is a web-based service provided by Tencent Cloud, primarily designed to help customers securely manage access permissions to resources within their Tencent Cloud accounts. With CAM, you can create, manage, and terminate users (groups), and control which individuals can access specific Tencent Cloud resources through identity and policy management.

When using CAM, you can associate a policy with a user or user group to allow or deny them access to specified resources for completing specified tasks. For more information on CAM policy basics, see Policy Syntax. For more information on how to use CAM policies, see Policy.

If you do not need to manage access to CAM-related resources for sub-accounts, you can skip this section. Skipping these parts will not affect your understanding and use of the rest of the documentation.

Get 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.

TCR Image Repository Resource-Level Permission Configuration

Last updated: 2023-09-26 10:05:59

TCR Permissions Overview

The address format for Tencent Cloud container images is: `ccr.ccs.tencentyun.com/${namespace}/${name}:${tag}`.
Permissions for image repositories are configured around the following two fields:

- `${namespace}`: The namespace to which the image repository belongs.
- `${name}`: The name of the image repository.

> **Note**
> The namespace `${namespace}` and image name `${name}` must not contain a forward slash "/".
> The `${tag}` field currently only supports authorization for deletion operations. Please refer to Image Tag Permissions.

Using the `${namespace}` and `${name}` fields, administrators can create detailed permission schemes for collaborators, enabling flexible permission management. For example:

- Permit collaborator A to pull images
- Forbid collaborator A from deleting images
- Forbid collaborator B from pulling the images in the namespace ns1

If you do not require detailed management of image repository permissions, you can use Preset Policy Authorization.
For more granular control over collaborator permissions, use Custom Policy Authorization.
TCR permissions are managed through Tencent Cloud CAM. You can learn more about using CAM in the following sections: User Management, Policy Management, and Authorization Management.

Preset Policy Authorization

To simplify TCR permissions management, two preset policies are configured in TCR:

- Full Read/Write Access to Container Registry (CCR)
  This preset policy configures all permissions for the container image service. If a collaborator is associated with this preset policy, they will have the same image repository permissions as the administrator. For more information, see Permission List.
- Image Repository (CCR) Read-Only Access
  This preset policy includes read-only permissions for the container image service. If a collaborator is **only** associated with this preset policy in the container image service, the following actions will be prohibited:
  - `docker push` Pushing an image
  - Create a namespace of image registry
  - Delete a namespace of image registry
  - Create an image repository
  - Delete an image repository
  - Delete an image tag

If you do not know how to associate a collaborator with a preset policy, please see Policy and Authorization Management.

Custom Policy Authorization

With custom policies, administrators can associate different permissions for various collaborators.
When assigning permissions, please consider the following factors:

- **Resource**: Which image repositories are associated with this permission policy, for example, all image repositories are described as `qcs::ccr:::repo/*`. For more information, see CAM Resource Description Method.
- **Action**: The operations that the permission policy performs on the resource, such as deletion, creation, etc., usually described using an API.
- **Effect**: The effect the permission policy has on the collaborator (Allow/Deny)

When you have planned the permission settings, you can assign the permissions. The following example shows how to "permit collaborators to create an image repository":

Step 1. Creating a custom policy

1. On the Policies page of the CAM console, click **Create Custom Policy** in the upper left corner.
2. In the pop-up window for selecting the creation method, click **Create by Policy Syntax** to proceed to the Select Policy Template page.
3. On the policy template selection page, choose **Create by Policy Syntax** > **Blank Template**.
4. Click **Next** to proceed to the Edit Policy page.
5. On the Edit Policy page, set the policy name to `ccr-policy-demo` and enter the following content into the "Edit Policy Content" input box.

```
{
  "version": "2.0",
  "statement": [{
    "action": "ccr:CreateRepository",
    "resource": "qcs::ccr:::repo/*",
    "effect": "allow"
  }]
}
```

6. Click **Complete**.

Step 2: Associate a Custom Policy

Upon completing the policy creation (ccr-policy-demo) in Step 1, you can associate it with any collaborator. For more information, see Authorization Management. Once the policy is associated, the collaborator will have the **permission to create image repositories in any namespace**.

`"resource": "qcs::ccr:::repo/*"` format explanation:

- `qcs::ccr:::` is a fixed format, indicating the developer's TCR service.
- `repo` is a fixed prefix, representing the resource type, which is an image repository here.
- The `*` following the slash (`/`) represents a match for all image repositories.

For a detailed description of resource, see CAM Resource Description Method.

Authorizing by resource

You can grant permissions for multiple resources at a time. For example, to "permit the deletion of the image repositories in namespaces `foo` and `bar` ", you can create the following custom policy:

```
{
  "version": "2.0",
  "statement": [{
    "action": [
      "ccr:BatchDeleteRepository",
      "ccr:DeleteRepository"
    ],
    "resource": [
      "qcs::ccr:::repo/foo/*",
      "qcs::ccr:::repo/bar/*"
    ],
    "effect": "allow"
  }]
}
```

> **Note**
> - `foo/*` in `qcs::ccr:::repo/foo/*` means all images in the image repository namespace `foo` .
> - `bar/*` in `qcs::ccr:::repo/bar/*` means all images in the image repository namespace `bar` .

Authorizing by action (API)

You can configure multiple `actions` for a single resource to achieve unified management of resource permissions. For example, to "allow creation, deletion, and push of image repositories in namespace foo," you can create the following custom policy:

```
{
  "version": "2.0",
  "statement": [{
    "action": [
      "ccr:CreateRepository",
      "ccr:BatchDeleteRepository",
      "ccr:DeleteRepository",
      "ccr:push"
    ],
    "resource": "qcs::ccr:::repo/foo/*",
    "effect": "allow"
  }]
}
```

Permission List

docker client permissions

resource: `qcs::ccr:::repo/${namespace}/${name}`
action:

- `ccr:pull` : Use the docker command line to pull an image
- `ccr:push` : Use the docker command line to push an image

Namespace permissions

resource: `qcs::ccr:::repo/${namespace}`
action:

- `ccr:CreateCCRNamespace` Create an image repository namespace
- `ccr:DeleteUserNamespace` Delete an image repository namespace

Image repository permissions

resource: `qcs::ccr:::repo/${namespace}/${name}`
action:

- `ccr:CreateRepository` Create an image repository
- `ccr:DeleteRepository` Delete an image repository
- `ccr:BatchDeleteRepository` Batch delete image repositories
- `ccr:GetUserRepositoryList` View the list of image repositories

> **Note**
> To prevent a collaborator from deleting certain images, configure multiple actions.

For example, to prohibit deleting any image repository:

```
{
  "version": "2.0",
  "statement": [{
    "action": [
      "ccr:BatchDeleteRepository",
      "ccr:DeleteRepository"
    ],
    "resource": "qcs::ccr:::repo/*",
    "effect": "deny"
  }]
}
```

Image Tag Permissions

resource: `qcs::ccr:::repo/${namespace}/${name}:${tag}`
action: `ccr:DeleteTag` Delete image tag permissions

Controlling TKE Cluster-Level Permissions

Using TKE Preset Policy Authorization

Last updated: 2023-09-26 10:06:24

This document describes the preset policies of Tencent Kubernetes Engine (TKE). It shows you how to associate sub-accounts with preset policies and grant sub-accounts specific permissions. You can configure preset policies based on this document and your actual business requirements.

TKE Preset Policies

You can grant your sub-account the necessary permissions by using the following preset policies:

Rule	Description
QcloudTKEFullAccess	This policy grants full read/write and access permissions for TKE, including permissions for TKE and related CVMs, CLBs, VPCs, monitors, and user groups.
QcloudTKEInnerFullAccess	TKE Full Access: TKE involves numerous products; it is recommended to configure the QcloudTKEFullAccess permission.
QcloudTKEReadOnlyAccess	This policy grants read-only permission for TKE.

The following preset policies are used to grant permissions to the TKE service itself when you use the TKE service. We do not recommend you associate the following preset policies with your sub-account:

Rule	Description
QcloudAccessForCODINGRoleInAccessTKE	This policy grants the relevant TKE permissions to the Coding service.
QcloudAccessForIPAMDofTKERole	This policy grants the relevant ENI permissions to the TKE service.
QcloudAccessForIPAMDRoleInQcloudAllocateEIP	This policy grants the relevant EIP permissions to the TKE service.
QcloudAccessForTKERole	This policy grants the relevant CVM, Tag, CLB, and CLS permissions to the TKE service.
QcloudAccessForTKERoleInCreatingCFSStorageclass	This policy grants the relevant CFS permissions to 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

In the step for setting user permissions when creating a sub-account, you can associate preset policies with the sub-account by direct association or association via group.

Direct Association

You can directly associate your sub-account with a policy to obtain the permissions contained in the policy.

1. Log in to the Cloud Access Management console and select **Users** > User List on the left sidebar.
2. In the "User List" management page, select **Authorize** on the right side of the row where the sub-account requiring permission settings is located.
3. On the **Associate Policies** page, select the policies that you want to associate.
4. Click **OK**.

Association with Group

You can add your sub-account to a user group. Then, the sub-account automatically obtains the permissions that are associated with this 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.

1. Log in to the Cloud Access Management console and select **Users** > User List on the left sidebar.
2. On the "User List" management page, select **More Actions** > **Add to Group** on the right side of the row where the sub-account

requiring permission settings is located.
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. If they can, this indicates that the sub-account was successfully authorized.

Authorizing By Using Custom Policies

Last updated: 2023-09-26 18:01:32

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 policy syntax structure is shown below:

- **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
> - Grant full read/write permissions to sub-accounts for multiple clusters using tags

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.

SDK	Sub-features included	Corresponding TencentCloud APIs	Indirect API Invocation	Resource-level Permission Control	Action Field
Creating an empty cluster	• Kubernetes Version Selection • Runtime Component Selection	tke:CreateCluster	cam:GetRole account:DescribeUserData account:DescribeWhiteList tag:GetTagKeys	• Create Cluster is an API-level permission control. • To obtain the VPC list, you	"tke:CreateCluster", "cam:GetRole", "tag:GetTagKeys", "cvm:GetVmConfigQuota", "vpc:DescribeVpcE

	• Select a VPC Network • Configuring Container Network • Custom Image Selection • IPVS Settings		cvm:GetVmConfigQuota vpc:DescribeVpcEx cvm:DescribeImages	need the resource permissions for VPC.	x", "cvm:DescribeImages"
Using an existing CVM to create a managed cluster	• Features included in creating an empty cluster • Using an Existing CVM as a Node • Attach Security Group • Mounting a data disk • Enable auto-scaling		cvm:DescribeInstances vpc:DescribeSubnetEx cvm:DescribeSecurityGroups vpc:DescribeVpcEx cvm:DescribeImages cvm:ResetInstance cvm:DescribeKeyPairs	• Create Cluster is an API-level permission control. • To obtain the CVM list, CVM resource permissions are required.	"tke:CreateCluster", "cvm:DescribeInstances", "vpc:DescribeSubnetEx", "cvm:DescribeSecurityGroups", "vpc:DescribeVpcEx", "cvm:DescribeImages", "cvm:ResetInstance", "cvm:DescribeKeyPairs"
Using an existing CVM to create a self-deployed cluster	• Features included in creating an empty cluster • Using an Existing CVM as a Node • Using an existing CVM as Master&ETCD • Attach Security Group • Mounting a data disk • Enable auto-scaling		cvm:DescribeInstances vpc:DescribeSubnetEx cvm:DescribeSecurityGroups vpc:DescribeVpcEx cvm:DescribeImages cvm:ResetInstance cvm:DescribeKeyPairs	• Create Cluster is an API-level permission control. • To obtain the VPC list, you need the resource permissions for VPC. • To obtain the CVM list, CVM resource permissions are required.	"tke:CreateCluster", "cvm:DescribeInstances", "vpc:DescribeSubnetEx", "cvm:DescribeSecurityGroups", "vpc:DescribeVpcEx", "cvm:DescribeImages", "cvm:ResetInstance", "cvm:DescribeKeyPairs"
Automatically creating a CVM to create a managed cluster	• Features included in creating an empty cluster • Purchasing CVM as a node • Attach Security Group • Mounting a data disk • Enable auto-scaling		cvm:DescribeSecurityGroups cvm:DescribeKeyPairs cvm:RunInstances vpc:DescribeSubnetEx vpc:DescribeVpcEx cvm:DescribeImages	• Create Cluster is an API-level permission control. • To obtain the VPC list, you need the resource permissions for VPC.	"cvm:DescribeSecurityGroups", "cvm:DescribeKeyPairs", "cvm:RunInstances", "vpc:DescribeSubnetEx", "vpc:DescribeVpcEx", "cvm:DescribeImages", "tke:CreateCluster"
Automatically creating a CVM to create a	• Features included in creating an empty cluster • Purchasing		cvm:DescribeSecurityGroups cvm:DescribeKeyPairs	• Create Cluster is an API-level permission control. • To obtain the	"cvm:DescribeSecurityGroups", "cvm:DescribeKeyPairs", "cvm:RunInstances

self-deployed cluster	CVM as a node • Purchasing CVM as Master&ETCD • Attach Security Group • Mounting a data disk • Enable auto-scaling		cvm:RunInstancesvpc:DescribeSubnetEx vpc:DescribeVpcEx cvm:DescribeImages	VPC list, you need the resource permissions for VPC.	", "vpc:DescribeSubnetEx", "vpc:DescribeVpcEx", "cvm:DescribeImages", "tke:CreateCluster"
Querying a cluster list	-	tke:DescribeClusters	-	Cluster-level permissions are required for obtaining a cluster list.	"tke:DescribeClusters"
Displaying cluster credentials	-	tke:DescribeClusterSecurity	-	Cluster-level permissions are required for displaying cluster credentials.	"tke:DescribeClusterSecurity"
Enabling/Disabling the private network/Internet access URL of a cluster	• Creating a public network access port for a managed cluster • Creating a cluster access endpoint • Modifying the security policy for the public network port of a managed cluster • Query the status of enabling public network ports for managed clusters • Deleting the public network access port of a managed cluster • Deleting a cluster access endpoint	tke:CreateClusterEndpointVip tke:CreateClusterEndpoint tke:ModifyClusterEndpointSP tke:DescribeClusterEndpointVipStatus tke:DescribeClusterEndpointStatus tke:DeleteClusterEndpointVip tke:DeleteClusterEndpoint	-	Cluster-level permissions are required for enabling or disabling cluster access.	-
Deleting a Cluster	-	tke:DeleteCluster	tke:DescribeClusterInstances tke:DescribeInstancesVersion tke:DescribeClusterStatus	Cluster-level permissions are required for deleting a cluster.	"tke:DescribeClusterInstances", "tke:DescribeInstancesVersion", "tke:DescribeClusterStatus", "tke:DeleteCluster"

Node modules

The following table describes the mappings between features and APIs.

SDK	Sub-features included	Corresponding TencentCloud APIs	Indirect API Invocation	Resource-level Permission Control	Action Field
Adding an Existing Node	• Adding an existing node to the cluster • Resetting the Data Disk • Configuring a security group	tke:AddExistedInstances	cvm:DescribeInstances vpc:DescribeSubnetEx cvm:DescribeSecurityGroups vpc:DescribeVpcEx cvm:DescribeImages cvm:ResetInstance cvm:DescribeKeyPairs cvm:ModifyInstancesAttribute tke:DescribeClusters	• To add an existing node, the corresponding cluster resource permissions are required. • To obtain the CVM list, CVM resource permissions are required.	"cvm:DescribeInstances", "vpc:DescribeSubnetEx", "cvm:DescribeSecurityGroups", "vpc:DescribeVpcEx", "cvm:DescribeImages", "cvm:ResetInstance", "cvm:DescribeKeyPairs", "tke:DescribeClusters", "tke:AddExistedInstances"
Creating a node	• Add a new node to the cluster • Resetting the Data Disk • Configuring 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.	"cvm:DescribeSecurityGroups", "cvm:DescribeKeyPairs", "cvm:RunInstances", "vpc:DescribeSubnetEx", "vpc:DescribeVpcEx", "cvm:DescribeImages", "tke:DescribeClusters"
Node list	Viewing a cluster node list	tke:DescribeClusterInstances	cvm:DescribeInstances tke:DescribeClusters	• Viewing the node list requires the corresponding cluster's resource permissions. • To obtain the CVM list, CVM resource permissions are required.	"cvm:DescribeInstances", "tke:DescribeClusters", "tke:DescribeClusterInstances"
Removing a node	-	tke:DeleteClusterInstances	cvm:TerminateInstances tke:DescribeClusters	• Viewing the node list requires the corresponding cluster's resource permissions. • To obtain the CVM list, CVM resource permissions are required. • To delete a node, the corresponding	"cvm:TerminateInstances", "tke:DescribeClusters", "tke:DeleteClusterInstances"

				node termination policy is required.	

Use Case

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

Last updated: 2023-09-15 15:42:53

Scenario

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.

Instructions

1. On the Policies page of the CAM console, click **Create Custom Policy** in the upper left corner.
2. In the pop-up window for selecting the creation method, click **Authorize by Tag** to navigate to the page for tag-based authorization.
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** to proceed to the Associate User/User Group/Role page. On this page, supplement the policy name and description information. The policy name is automatically generated by the console, with the default being "policygen" and the suffix number generated based on the creation date. You can customize it as needed.
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 **Finish**.

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

Last updated: 2023-09-15 15:43:25

Scenario

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.

Instructions

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 Create by Policy Syntax.
4. Select the "Blank Template" type and click **Next**.
5. Enter a custom policy name and replace "Edit policy content" with the following content.

```
{
 "version": "2.0",
 "statement": [
   {
     "action": [
       "tke:*"
     ],
     "resource": [
       "qcs::tke: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", replace `qcs::tke:sh::cluster/cls-XXXXXXX` with the cluster in the specified region you want to grant permissions to. For example, if you need to grant full read/write permissions to the cls-69z7ek9l cluster in the Guangzhou region, change `qcs::tke:sh::cluster/cls-XXXXXXX` to `"qcs::tke:gz::cluster/cls-69z7ek9l"`.

> **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 Policy** to complete the configuration of full read/write permissions 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 Create by Policy Syntax.
4. Select the "Blank Template" type and click **Next**.
5. Enter a custom policy name and replace "Edit policy content" with the following content.

```
{
 "version": "2.0",
 "statement": [
    {
        "action": [
            "tke:Describe*",
            "tke:Check*"
        ],
        "resource": "qcs::tke: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::tke:gz::cluster/cls-1xxxxxx` to the cluster in the specified region you want to grant permissions to. For example, if you need to grant read-only permission to the cls-19a7dz9c cluster in the Beijing region, change `qcs::tke:gz::cluster/cls-1xxxxxx` to `qcs::tke:bj::cluster/cls-19a7dz9c`.
7. Click **Create Policy** to complete the configuration of read-only permissions for a single cluster.

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

Last updated: 2023-09-26 10:11:56

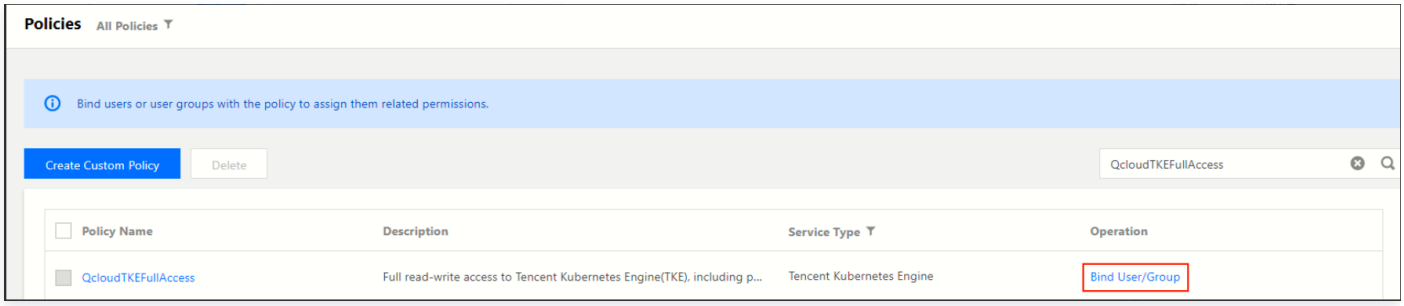
Scenario

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.

Instructions

Configuring Full Read/write Permission

1. Log in to the Access Management console and select Policies from the left navigation bar.
2. On the "Policies" management page, select **Associate User/Group/Role** for the **QcloudTKEFullAccess** policy row, as shown below:



3. In the "Associate User/User Group/Role" pop-up window, select the account that requires full read/write permissions for TKE service, and click **Confirm** to complete the configuration of full read/write permissions for the sub-account on TKE service.
4. On the Policy Management page, click **Associate Users/User Groups/Roles** in the **QcloudCCRFullAccess** policy row.
5. In the "Associate Users/User Groups/Roles" pop-up window, select the accounts that require full read/write permissions for the image repository, and click **OK** to complete the configuration of full read/write permissions for the sub-account.

> **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

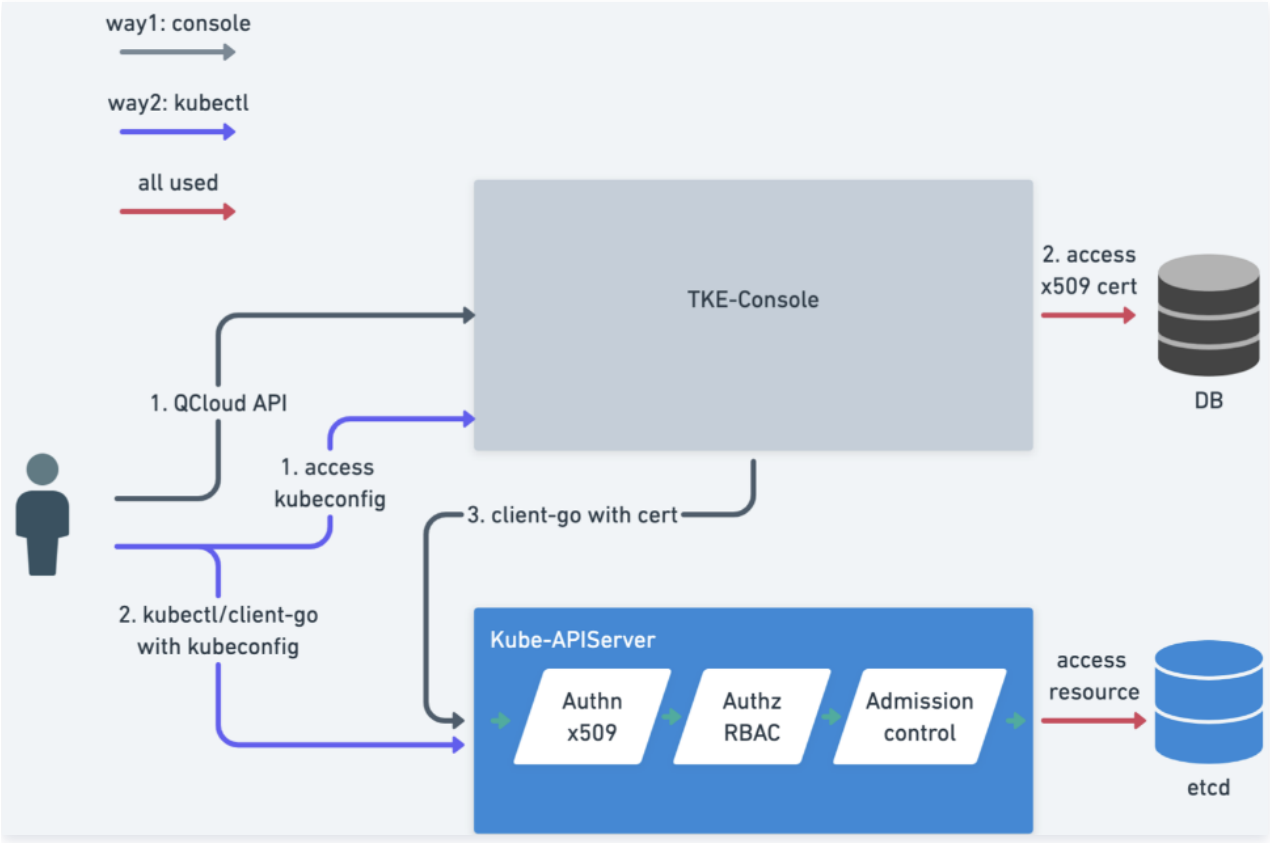
1. Log in to the Access Management console and select Policies from the left navigation bar.
2. On the "Policies" management page, select **Associate Users/User Groups/Roles** for the **QcloudTKEReadOnlyAccess** policy row.
3. In the "Associate User/User Group/Role" pop-up window, select the account that requires read-only access to TKE services and click **Confirm** to complete the configuration of read-only permissions for the sub-account on TKE services.
4. On the Policy Management page, click **Associate Users/User Groups/Roles** in the **QcloudCCRReadOnlyAccess** policy row.
5. In the "Associate Users/User Groups/Roles" pop-up window, select the accounts that require read-only access to the image repository and click **OK** to complete the configuration of read-only permissions for the sub-account.

> **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: 2023-09-26 10:12:37

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



Definitions

RBAC (Role-Based Access Control)

Role-based access control (RBAC) is a method of indirectly granting permissions to users by associating them with roles and associating roles with permissions. In Kubernetes, RBAC is implemented through the `rbac.authorization.k8s.io` API Group, allowing cluster administrators to dynamically configure policies via the Kubernetes API.

Role

Role defines the permissions of a role in a namespace.

ClusterRole

ClusterRole defines the permissions of roles in a cluster.

RoleBinding

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

ClusterRoleBinding

ClusterRoleBinding grants the permissions of a role to a user or group of users in order to grant cluster authorization to users. For more information, see the Kubernetes official documentation.

Solution for TKE Kubernetes Object-Level Permission Control

Verification method

Kubernetes APIServer supports a wide variety of authentication strategies, such as x509 certificates, bearer tokens, and basic authentication. Among these, only the bearer token authentication strategy supports multiple token authentication methods, including known-token CSV file bearer tokens, service account tokens, OIDC tokens, and webhook token servers.
After considering implementation complexity and usage in different scenarios, TKE chose to use x509 certificates as the verification method. This verification method has the following advantages:

- This method is easier for users to understand.
- No complex changes are necessary for 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 in the console, the backend uses the client certificate of the sub-account to access the user's Kubernetes API server by default.
- A sub-account can update its unique client certificate to prevent disclosure of the credentials.
- A root account or an account that has the `tke:admin` permission for a cluster can view and update the certificates of other sub-accounts.

Authorization Method

Kubernetes includes two mainstream authorization methods: RBAC and Webhook Server. To provide a consistent experience for users familiar with Kubernetes and to integrate with native Kubernetes, TKE has chosen to use the RBAC method. This method offers predefined Roles and ClusterRoles, and users only need to create corresponding RoleBindings and ClusterRoleBindings within the cluster to implement authorization changes. The advantages of this method are as follows:

- 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 assign read-only permissions to a sub-account or assign read/write permissions to only a certain namespace of a sub-account. To perform more fine-grained permission configurations for sub-accounts, see the following documents:

- Using Predefined Identity Authorization
- Custom Policy Authorization

Using Preset Identity Authorization

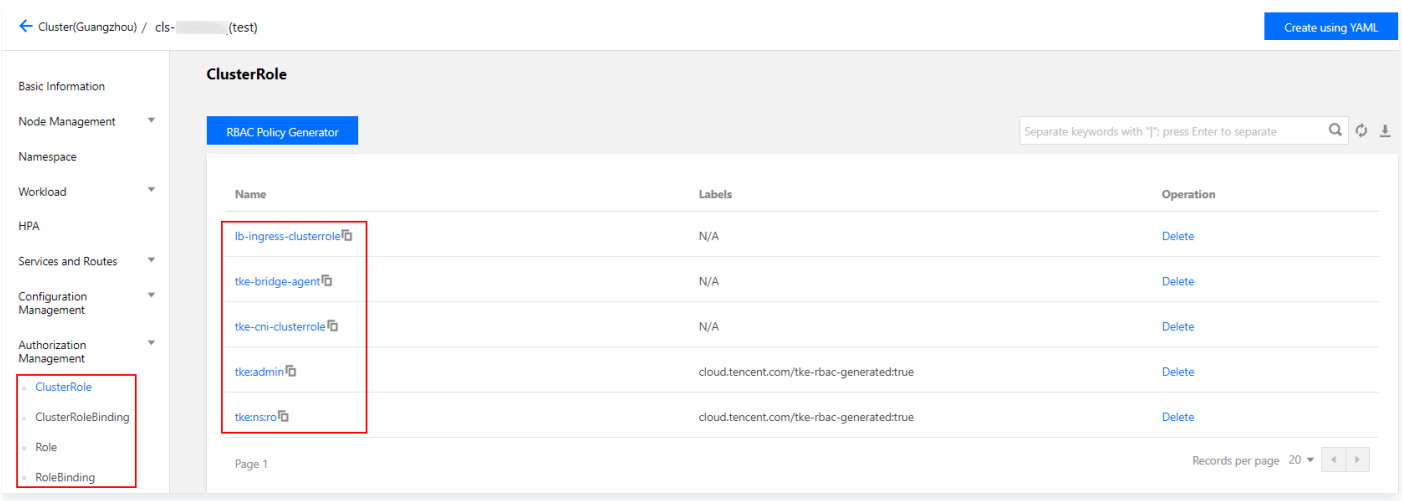
Last updated: 2023-09-15 15:52:25

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 Description

The TKE console provides an authorization management page. By default, the **primary account** and **cluster creator** have administrator privileges. You can manage permissions for other sub-accounts that have the DescribeCluster Action permission for the cluster, as shown in the following image:



ClusterRole Description

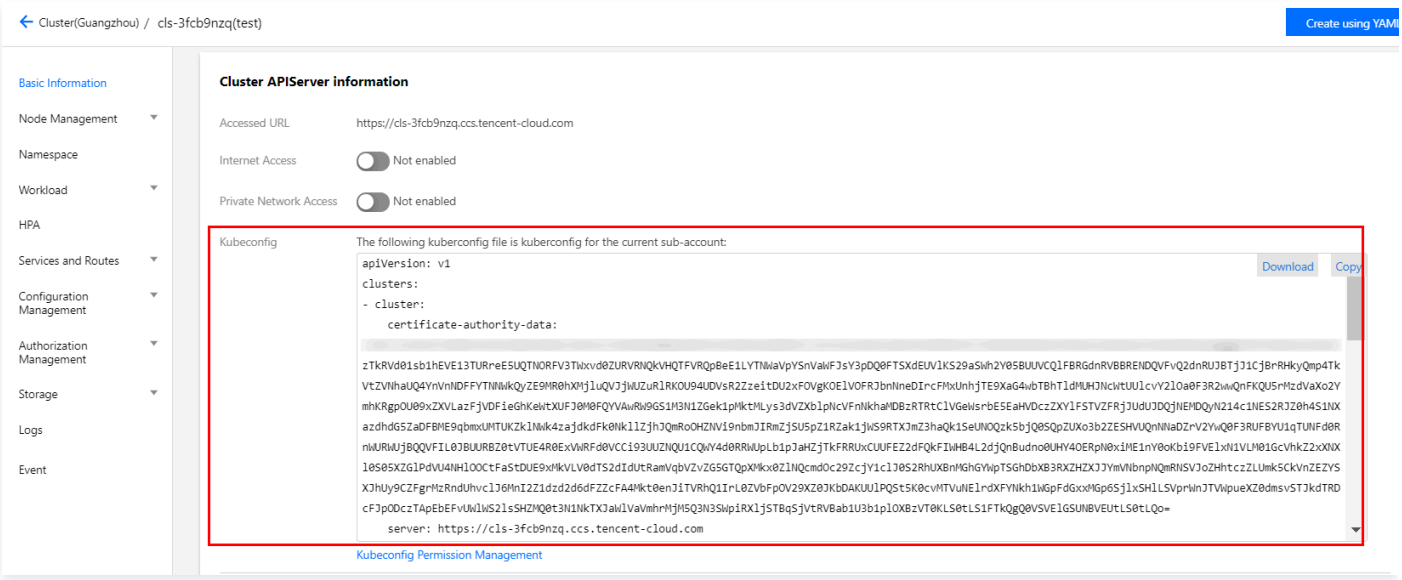
- **For all namespaces:**
- **Administrators (tke:admin)**: have read/write permissions for the resources in all namespaces, read/write permissions for cluster nodes, storage volumes, namespaces, and quotas, and read/write permissions for sub-account configuration.
- **OPS personnel (tke:ops)**: have read/write permissions for the resources visible in the console in all namespaces and read/write permissions for cluster nodes, storage volumes, namespaces, and quotas.
- **Developers (tke:dev)**: have read/write permissions for the resources visible in the console in all namespaces.
- **Restricted personnel (tke:ro)**: have read-only permission for the resources visible in the console in all namespaces.
- **Custom:** user-defined ClusterRole.
- **For a specified namespace:**
  - **Developers (tke:ns:dev)**: have read/write permissions for the resources visible in the console for the specified namespace.
  - **Read-only users (tke:ns:ro)**: have read-only permissions for the resources visible in the console for the 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 annotations: `cloud.tencent.com/tke-account-nickname: yournickname` and the label: `cloud.tencent.com/tke-account: "yourUIN"`.

Instructions

Obtaining credentials

TKE automatically creates separate credentials for each sub-account. Users can obtain the Kubeconfig file containing the credential information of the current account by accessing the cluster details page or calling the cloud API DescribeClusterKubeconfig.
The steps to obtain the Kubeconfig file through the console are as follows:

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.
3. On the cluster details page, select **Basic Information** in the left sidebar to view and download the Kubeconfig file in the "Cluster API Server Information" section, as shown in the following image:



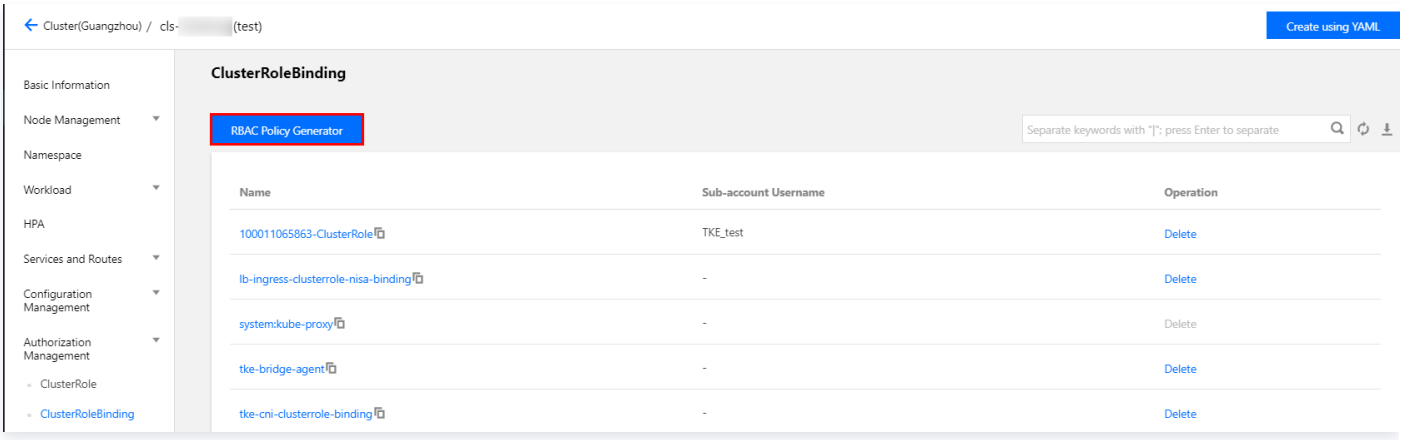
Managing credentials

Cluster administrators can access the credential management page to view and update the credentials for clusters under all accounts. For more information, please refer to Updating TKE Cluster Access Credentials for Sub-accounts.

Authorize

> **Note**
> Please contact the cluster administrator (root account, cluster creator, or user with the admin role) for authorization.

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



4. In the "Select Sub-account" step of the "Manage Permissions" page, select the sub-account to be authorized and click **Next**.
5. When you set the cluster RBAC, set the permissions as follows:
   - **Namespace List**: specify the namespaces for which the permissions take effect.
   - **Permissions**: set the permissions as required based on the permission description on the page.

> **Note**

> You can also click **Add Permission** to continue customizing the permission settings.

Authentication

Log in to your sub-account and verify that the sub-account has the permissions, which indicates that the authorization is successful.

Custom Policy Authorization

Last updated: 2023-09-15 16:15:56

This document describes how to customize ClusterRoles and Roles in Kubernetes to grant a sub-account specified permissions. You can perform the operations based on your business requirements.

Policy Syntax Description

You can write a policy syntax or access 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

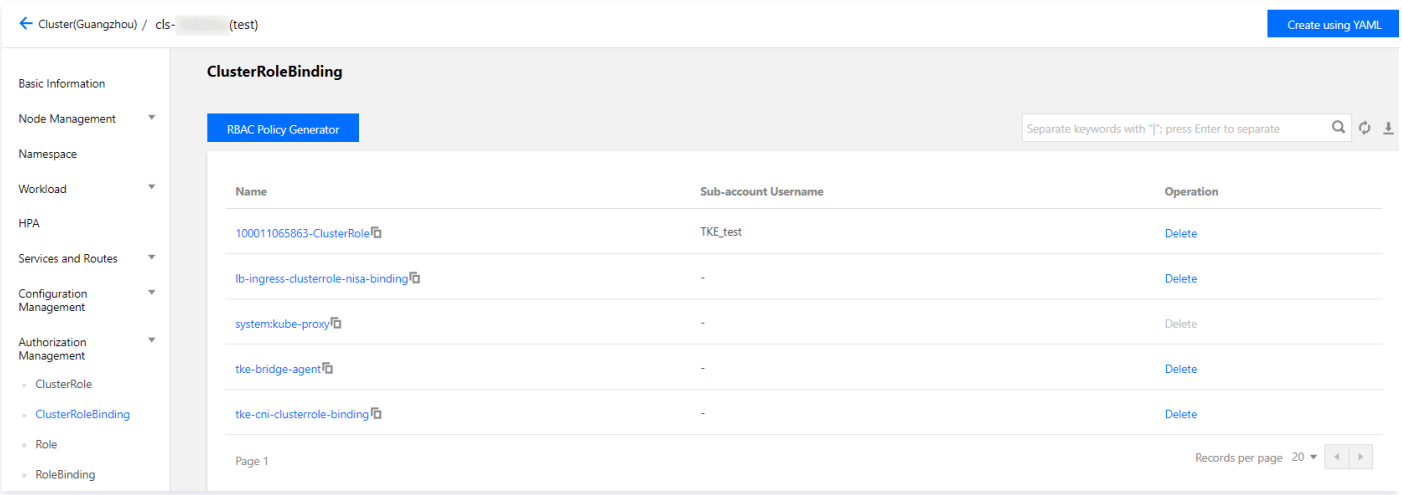
```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: testClusterRole
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
```

Instructions

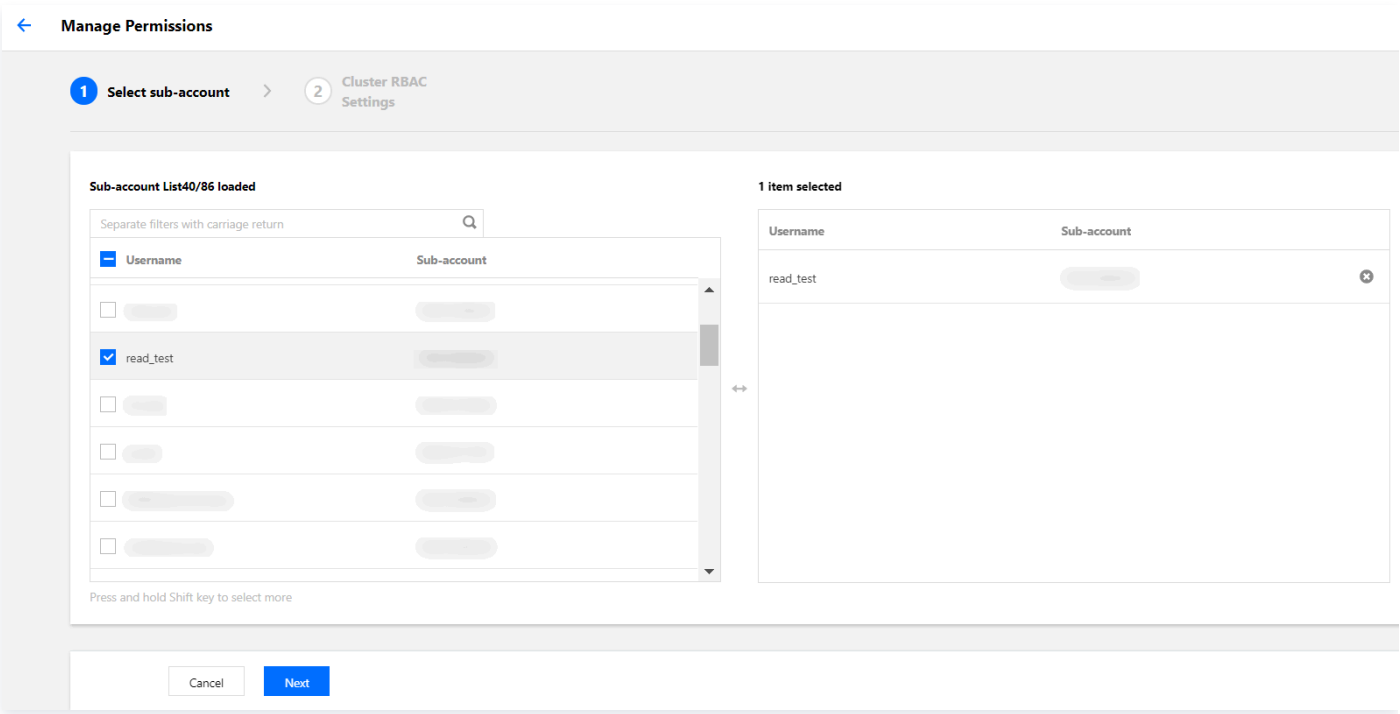
Note

This step takes binding a custom ClusterRole to a sub-account as an example. The process is similar to binding a Role. You can perform the operations based on your actual requirements.

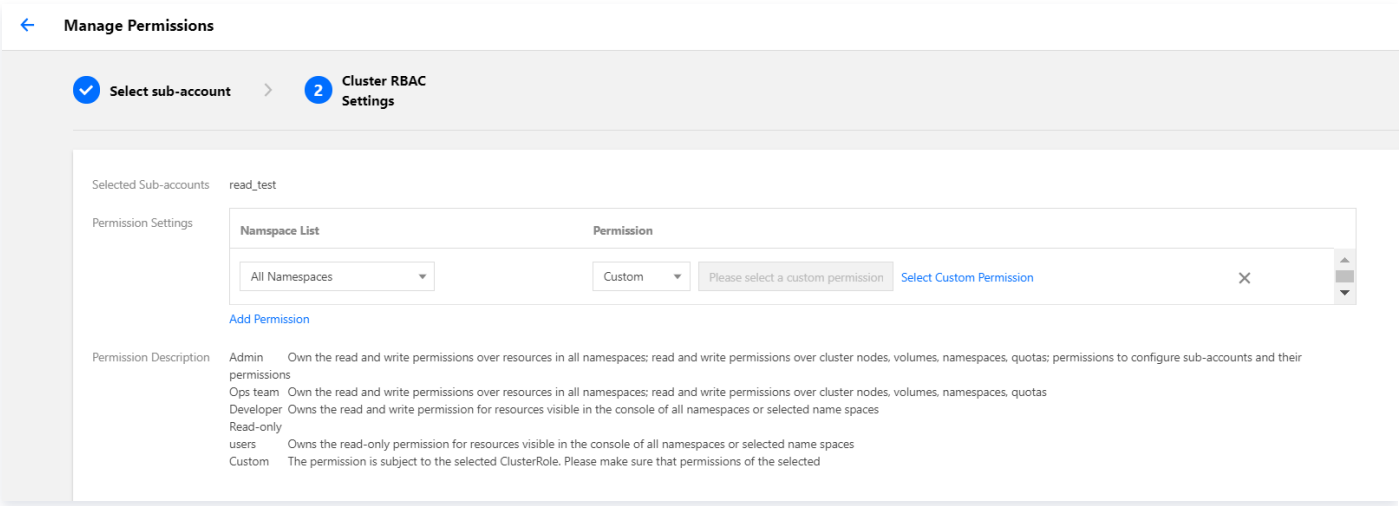
1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the **Cluster Management** page, click the ID of the target cluster.
3. In the cluster details page, select **Authorization Management** > **ClusterRoleBinding** in the left sidebar, as shown below:



4. In the "ClusterRole" management page, select **Create Resource with YAML** in the top-right corner.
5. Enter the custom policy YAML content in the editing interface and click **Done** to create a ClusterRole. This step uses the ClusterRole: for a cluster YAML as an example. After creation, you can view the custom permission "testClusterRole" in the ClusterRole management page.
6. In the "ClusterRoleBinding" management page, click on **RBAC Policy Generator**.
7. In the "Select sub-account" step on the "Manage Permissions" page, check the sub-account to be authorized and click **Next**. As shown in the image below:



8. Navigate to the "Cluster RBAC settings" page and follow the instructions below to configure permissions, as shown in the figure:



- **Namespace List**: specify the namespaces for which the permissions take effect.
- **Permission**: Select "Custom" and click **Choose Custom Permission**. Select the desired permissions from the custom permission list. In this example, we choose the previously created custom permission "testClusterRole".

> **Note**
> You can also click **Add Permission** to continue customizing the permission settings.

9. Click **Finish** to complete the authorization process.

References

For more information, refer to the Kubernetes official documentation: Using RBAC Authorization

Updating the TKE Cluster Access Credentials of Sub-accounts

Last updated: 2023-09-15 16:17:16

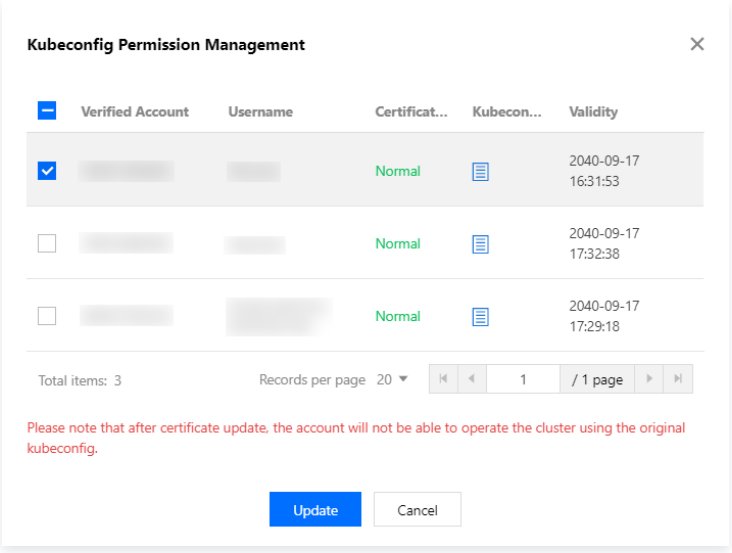
Access Credentials

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.
- In the new authorization method adopted by TKE, when different sub-accounts obtain the credentials for accessing a cluster (accessing the basic information page of the cluster or calling Tencent API DescribeClusterKubeconfig), the sub-accounts can obtain their unique x509 client certificates, which are issued by the self-signed CA of each cluster.
- When a sub-account accesses Kubernetes resources in the console, the backend uses the client certificate of the sub-account to access the user's Kubernetes API server by default.
- A sub-account can update its unique client certificate to prevent disclosure of the credentials.
- A root account or an account that has the `tke:admin` permission for a cluster can view and update the certificates of other sub-accounts.

Instructions

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.
3. On the cluster details page, select **Basic Information** in the left sidebar, and click **Kubeconfig Permission Management** in the "Cluster API Server Information" section.
4. In the displayed "Kubeconfig Permission Management" window, select the required authentication accounts and click **Update** as needed. As shown in the following figure:



Cluster Management
Cluster Overview

Last updated: 2023-09-15 16:18:03

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 the managing node of a cluster.
- Etcd: Used to retain the status information of the entire cluster.
- Node: This is the working node where the business is run.

Cluster Type

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: 2023-09-15 16:18:40

Managed Master Deployment Mode

Feature Overview

Tencent Kubernetes Engine (TKE) offers a fully managed Kubernetes cluster management service, including Master and Etcd components. In this mode, the Kubernetes cluster's Master and Etcd are centrally managed and maintained by the Tencent Cloud technical team. You only need to purchase the cluster and provide the worker nodes required to run the workloads, without worrying about cluster management and maintenance.

Notes on the managed master deployment 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, please 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, please delete the cluster.

Independent Master Deployment Mode

Feature Overview

Tencent Cloud TKE also offers a self-deployed master deployment mode, providing you with full control over the cluster. In this mode, the Kubernetes cluster's Master and Etcd components will be deployed on the CVM instances you purchased. You will have complete management and operational control over the Kubernetes cluster.

Notes on the independent master deployment mode

- This mode is only available for Kubernetes v1.10.x and later.
- In this mode, you need to purchase additional resources to deploy the Master and Etcd nodes of the Kubernetes cluster.
- If your cluster contains a large number of nodes, we recommend that you select a high-spec model for the CVMs. When selecting models, see the following table for reference:

Cluster Scale	Recommended Master Node Configuration	Recommended Node Quantity
Approximately 100 nodes	8 cores, 16 GB memory, SSD system disk	3 or more nodes
Approximately 500 nodes	16 cores, 32 GB memory, SSD system disk	3 or more nodes
Over 1,000 nodes	Contact Us	3 or more nodes

Purchase Requirements

To ensure the high availability and performance of your cluster and services, you need to meet the following requirements in independent deployment mode:

- At least three CVMs are deployed for the Master and Etcd nodes.
- The model of the CVMs configured for Master and Etcd nodes contains at least 4 cores.
- SSDs are deployed as system disks on the CVMs for the Master and Etcd nodes.

Supports and Limits

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: kube-APIserver, kube-scheduler, kube-controller-manager, tke-tools, systemd, and cluster-container-agent.
- Core component configuration parameters: 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, please contact us.

Cluster Lifecycle

Last updated: 2023-09-15 16:19:11

Notes on Cluster Lifecycle Status

Status	Note
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 properly.
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

Tencent Container Service is based on Kubernetes and is a declarative service. If you have created IaaS resources such as Cloud Load Balancers (CLBs) and Cloud Block Storage (CBS) in the container service and no longer need to use them, please delete the corresponding Service and PersistentVolumeClaim objects in the Container Service Console. If you only delete the CLB in the CLB Console or the CBS in the CBS Console, the container service will recreate new CLBs and CBSs, and continue to charge related fees.

Creating a Cluster

Last updated: 2023-09-15 16:30:29

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

Preparations

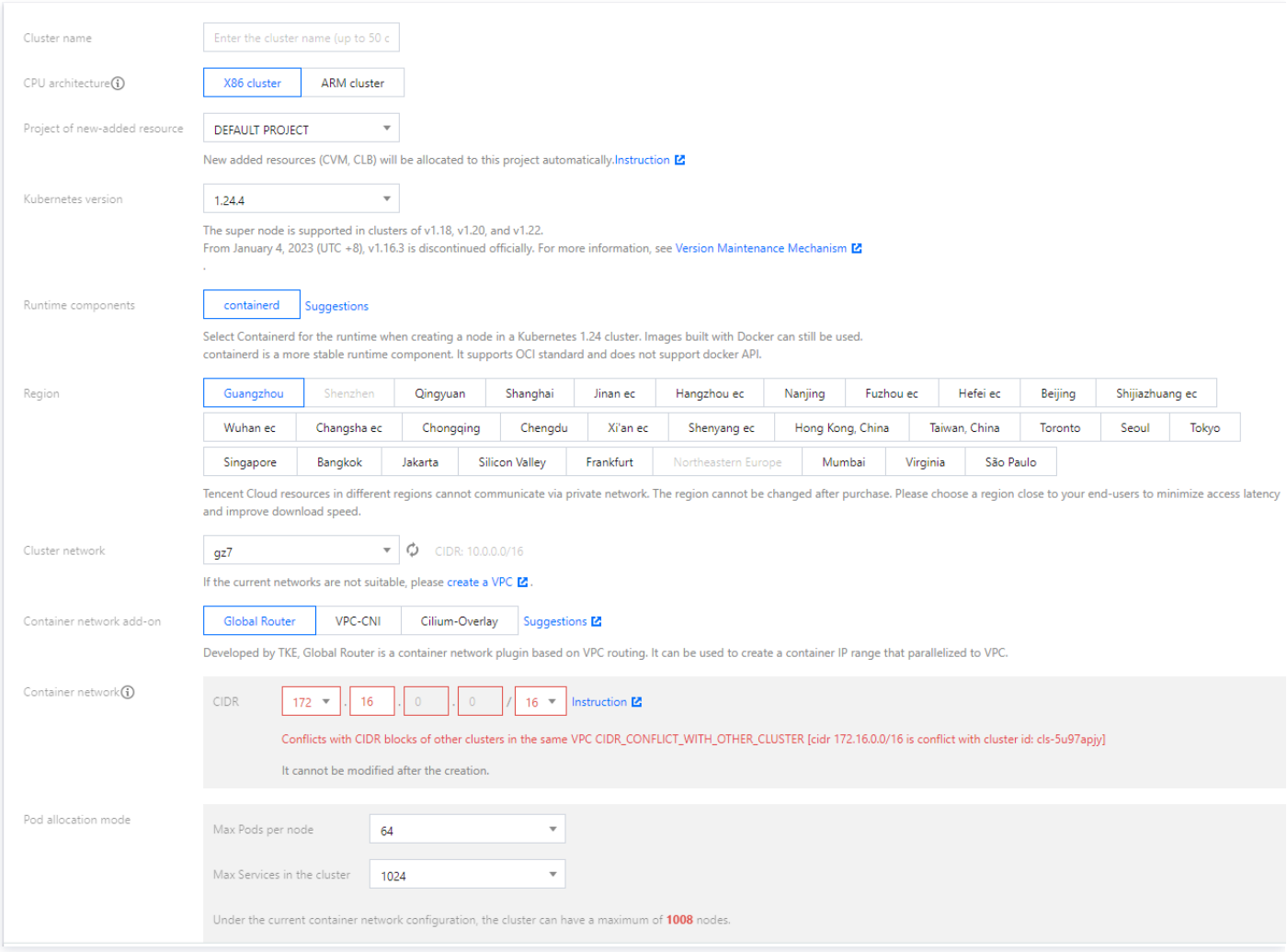
Before creating a cluster, you need to complete the following preparations:

- Sign up for a Tencent Cloud account and complete identity verification.
- 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 an SSH key 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 Cluster Information

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. Select **General cluster**, and click **Create**.
4. On the "Create Cluster" page, configure the basic information for the cluster, as shown in the following image:



- **Cluster name**: Set the name of the cluster with up to 50 characters.
- **Project of new-added resource***: 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 address range to containers in the cluster. For details, see Network Settings for Containers and Nodes.
- **Image Provider**: Supports three types of images – public images, custom images, and marketplace images. 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 mis-operation in the console or by 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 details, refer to 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**.

Add node

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**: Both **Pay-as-you-go** and **Monthly Subscription** are supported. For details, see Billing Mode.
- **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**: Select a model with more than four CPU cores. For more information, see Instance Types and Customizing Linux CVM Configurations.
    - **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 Kubernetes node naming restrictions, manually naming hostnames only supports lowercase letters, such as `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 AZs.

Existing nodes

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

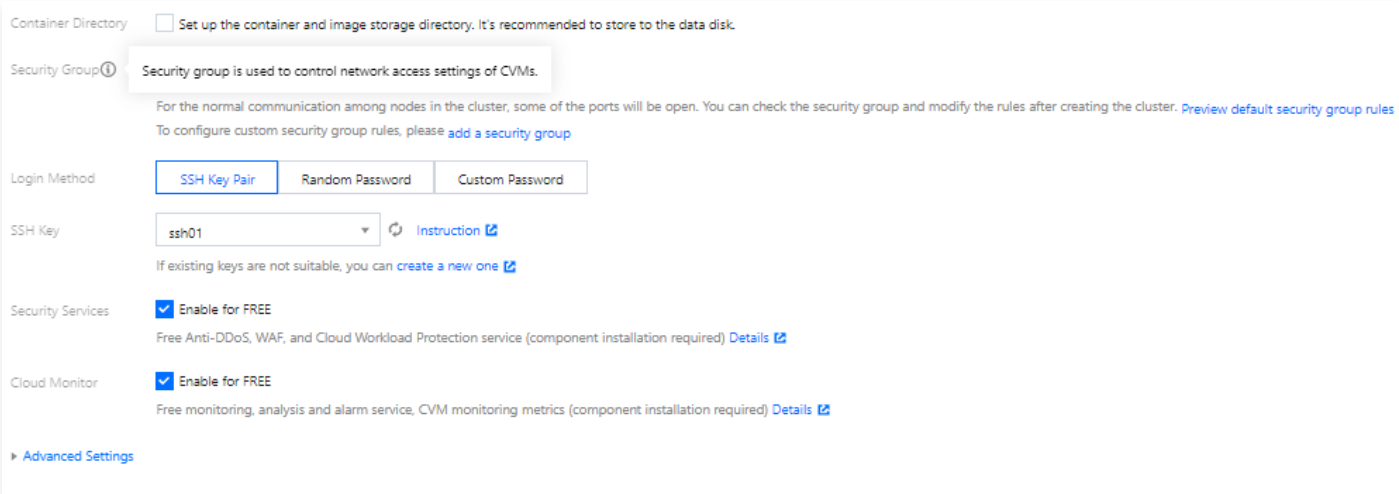
> **Note**
> - The selected CVMs will be reinstalled and all data in the system disk will be cleared.
> - The selected CVMs will be migrated to the project of the cluster. All related security groups will be unbound. You need to bind them manually again.
> - If you specify data disk mounting parameters when configuring CVM instances, these parameters will apply to **both Master and Worker nodes**. For more considerations, please refer to the Adding Existing Nodes section and the **Data Disk Mounting** parameter description.

- **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:



- **qGPU Sharing**: When enabled, all incremental GPU nodes in the cluster will have GPU sharing capabilities by default. You can control the isolation capability using Labels. Please note that the qGPU component must be installed to use GPU sharing scheduling capabilities properly. For more information, see How 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 network access of the CVM. 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 Methods**: three login methods are available.
  - **SSH Key Pair**: a key pair is a pair of parameters generated by an algorithm. Compared to regular passwords, it is a more secure way to log in to a CVM. For details, see SSH Key.
  - **Random Password**: an automatically generated password will be sent 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.

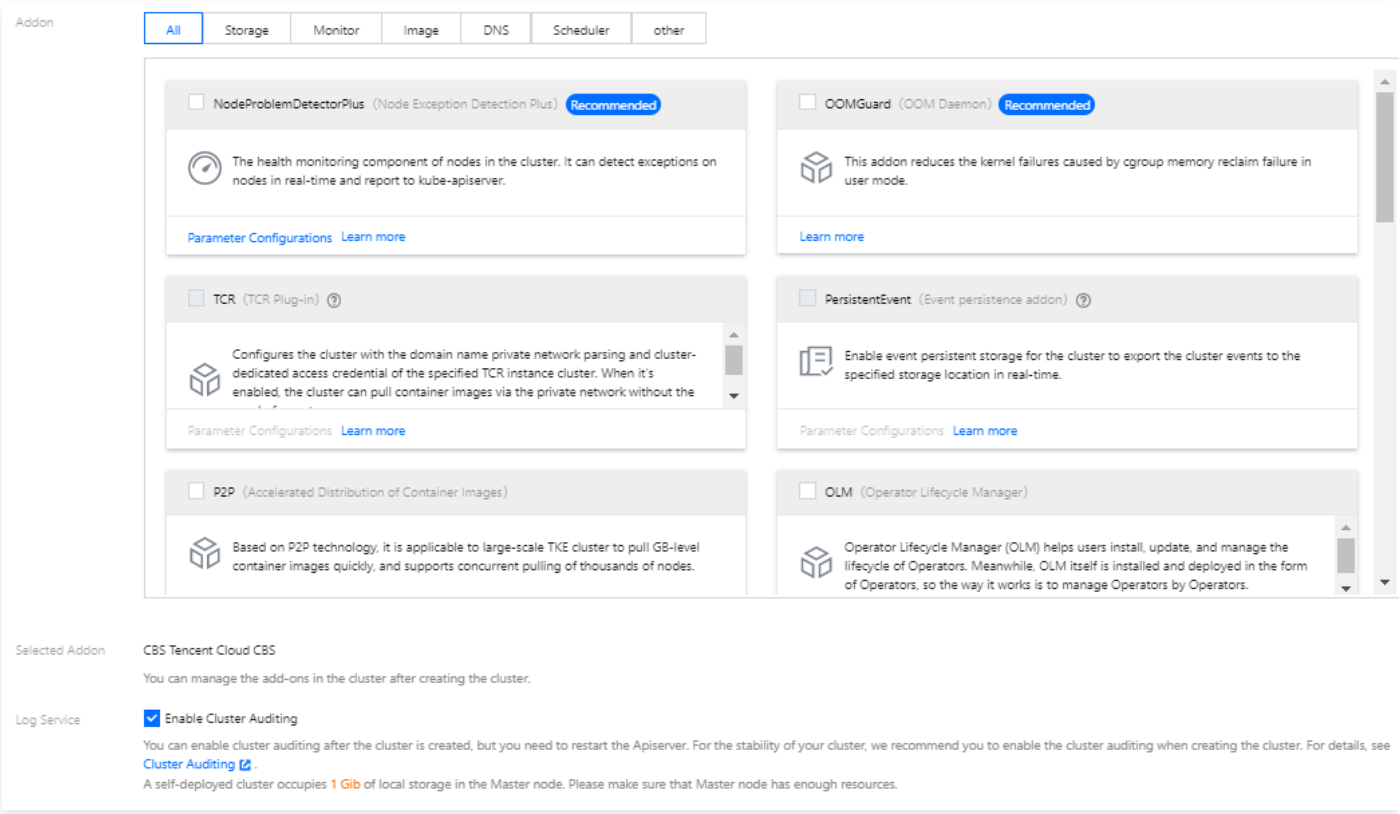
2. (Optional) Click **Advanced Settings** to view or configure more information.

- **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 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`.
- **Cordon**: When "Cordon this node" is selected, new Pods will not be scheduled to this node. To unlock the node, you can either manually cancel the lock or execute the Unlock command in the custom data. Please set 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. Add-on Configurations

1. In the "Add-on Configuration" step, configure add-ons based on the following information, as shown in the figure below:



- **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.

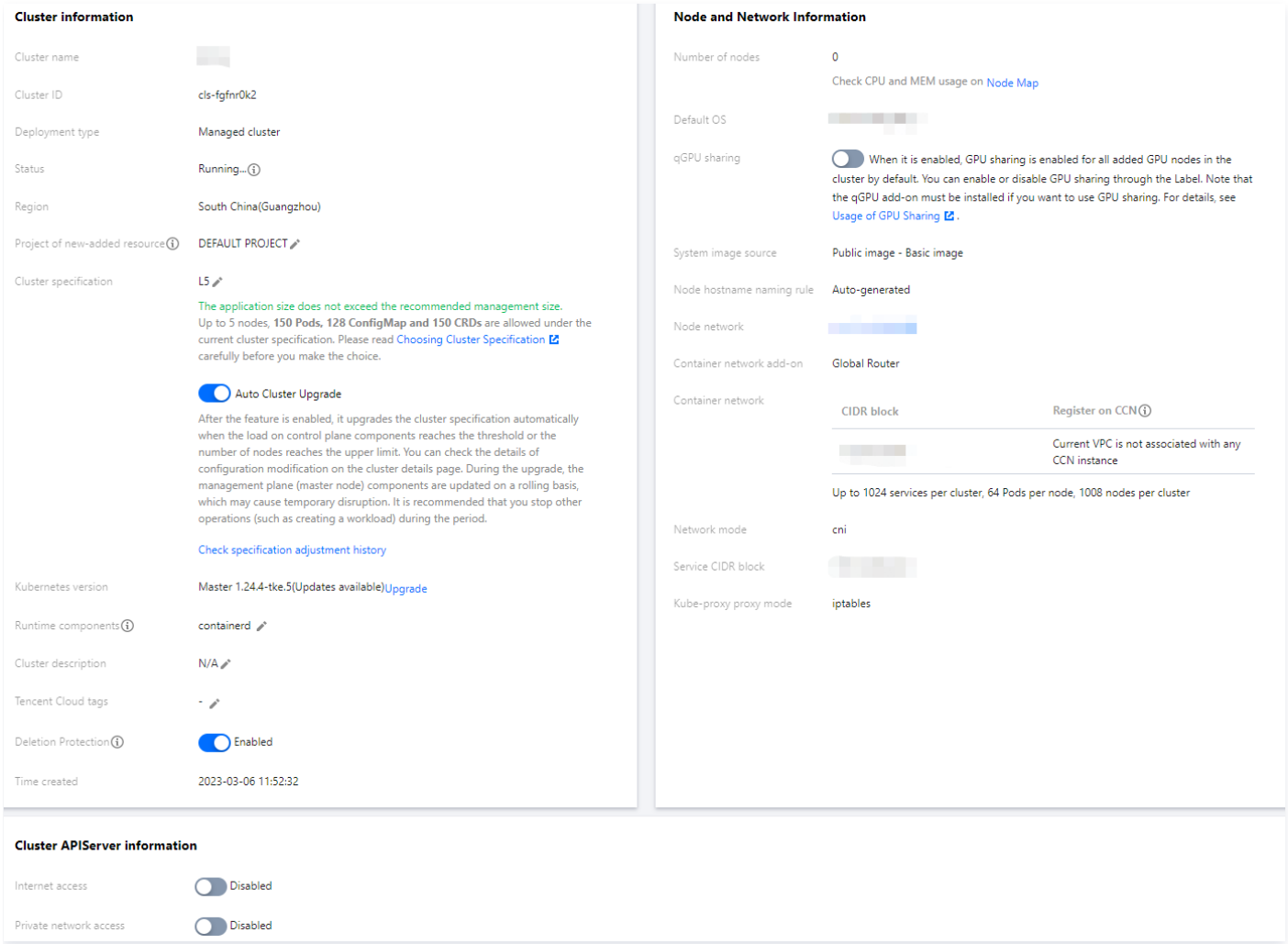
2. Click **Next**.

5. Confirm the information

On the "Information Confirmation" page, review the selected configuration details and costs for the cluster, and check the box next to **I have read and agree to the** TKE Service Level Agreement**. Click Finish** to create a cluster.

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.



Create a Cluster via the API

You can also create a cluster using the CreateCluster API. For more information, please refer to the CreateCluster API documentation.

Changing the Cluster Operating System

Last updated: 2023-09-15 17:18:53

Notes on the OS

- Modifying the operating system only affects subsequently added or reinstalled nodes, and has no impact on the existing nodes' OS.
- Within the same cluster, nodes using different versions of the operating system will not affect the cluster's functionality.
- A single script may not be suitable for all operating systems. It is recommended that you verify the compatibility of the node's OS with the script after configuring the script for the node.
- To use the custom image feature, please contact us online to apply.

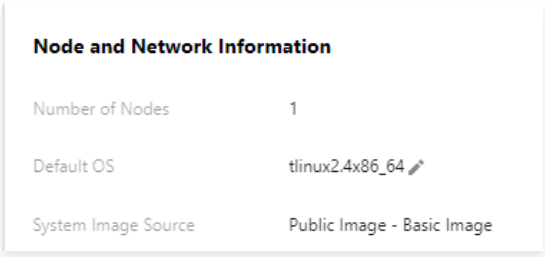
> **Note**
> If you need to use the custom image feature, create custom images using the basic image provided by the container service. For more information, see Custom Image Instructions.

Instructions

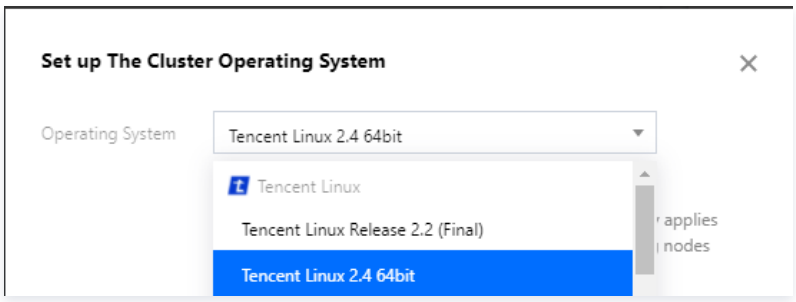
Changing the Default Cluster Operating System

> **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 select **Cluster** in the left sidebar.
2. On the "Cluster Management" page, select **View Cluster Credentials** on the right side of the target cluster's row to access the cluster's basic information page.
3. In the "Node and Network Information" section, click on the icon to the right of **Default OS**. See the image below:



4. In the pop-up "Set up The Cluster Operating System" window, change the operating system as shown in the figure below:



5. Click **Submit**.

Cluster Scaling

Last updated: 2023-09-15 17:23:42

Scenario

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 add or remove nodes.
- Automatically add or remove nodes using auto-scaling.
- Scale applications using super nodes without the need for node scaling.

Preparations

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

Instructions

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

During the node creation process, you can configure a CVM instance on the "Create Node" page to scale out the cluster. For detailed instructions, please refer to Create 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 you 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.

During the process, you can select and configure the CVMs to be added to the cluster on the "Add Existing Nodes" page to scale out the cluster. For detailed instructions, please refer to Add Existing Nodes.

Removing a node

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

Automatically add or remove nodes using elastic 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.

Expand business operations using super nodes

Super nodes are a scheduling capability that allows Pods in standard Kubernetes clusters to be scheduled on resources outside of the cluster server nodes, providing dynamic resource replenishment when resources are insufficient. For more information, see Super Node Overview.

FAQs Overview

For issues related to scaling, see FAQs for Scaling.

Connecting to Cluster

Last updated: 2023-09-15 17:31:52

Scenario

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

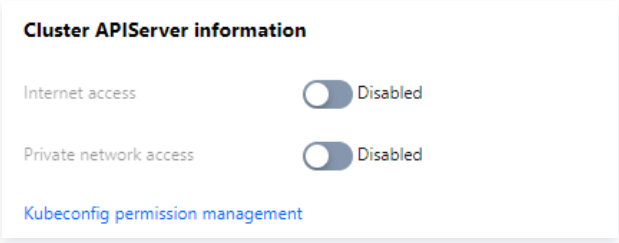
Solution 1: Connecting to a Cluster via Cloud Shell

TKE integrates Tencent Cloud's Cloud Shell, enabling you to effortlessly connect to a cluster with a single click in the Tencent Cloud console and use kubectl for flexible cluster management.

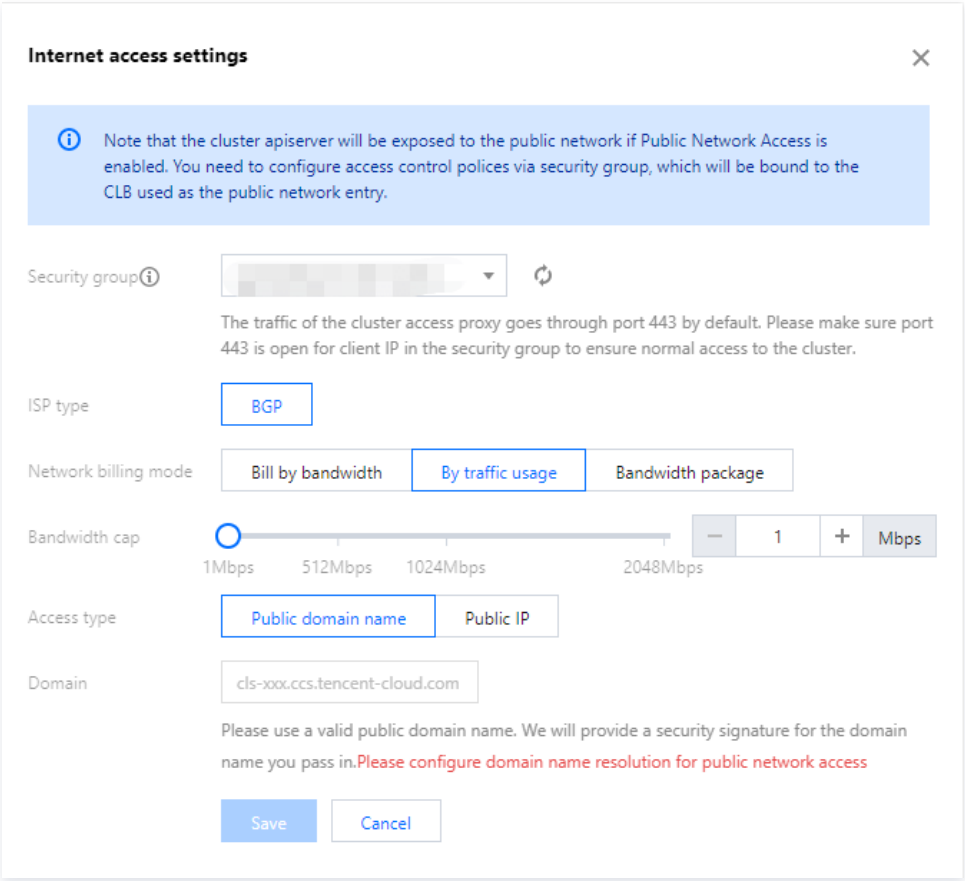
Instructions

Step 1: Enable public network access for the cluster

1. Log in to the TKE console and click 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 the cluster access status, as shown in the figure below:

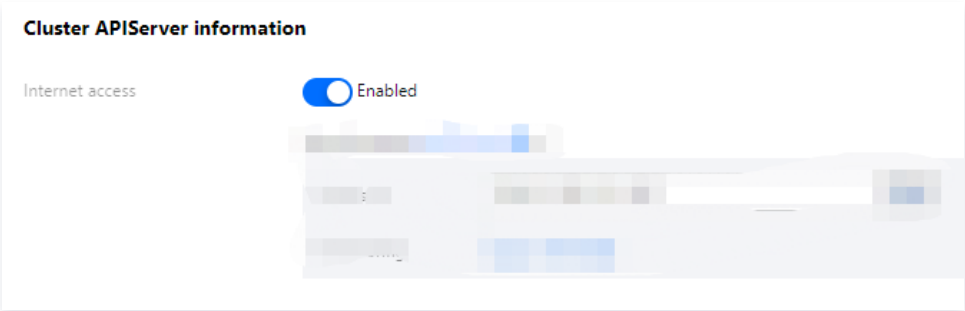


4. Click to enable public network access. When enabling public network access, you need to configure the relevant parameters, as shown in the following image:



- **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**: For CLB-related parameters, please refer to the CLB Creation Guide and set them according to your actual requirements.
- **Access type**: After selecting a public domain name, you need to provide a custom domain name. We will generate a security signature for the domain name you provide, and you must configure the public network resolution yourself. If you choose the default CLB domain name, there is no need to manually configure the DNS record or any other operations.

5. Ensure that public network access is enabled, as shown in the figure below:



Step 2: Use Cloud Shell to connect to the cluster

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the **Cluster Management** page, select the region where the target cluster resides, and click **More > Connect to cluster** on the right, as shown in the figure below:

3. At the bottom of the console, you will find the Cloud Shell entry. You can directly enter kubectl commands in the command box.

Solution 2: Connecting to a Cluster via a Local Computer

Preparations

You have installed curl.

Instructions

Step 1: Install kubectl tool

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.
> - Please replace "v1.18.4" in the command line with the kubectl version required for your business needs. The client's kubectl and server's Kubernetes versions should be consistent. You can view the Kubernetes version in the "Cluster Information" section under **Basic Information**.

For Mac OS X

Run the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/darwin/amd64/kubectl
```

Linux system

Run the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/linux/amd64/kubectl
```

For Windows systems

Run the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/windows/amd64/kubectl.exe
```

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.

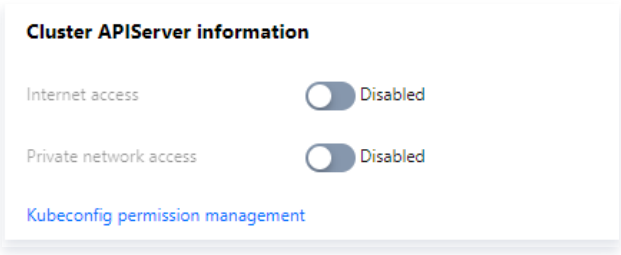
```
kubectl version
```

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

```
Client Version: version.Info{Major:"1", Minor:"5", GitVersion:"v1.5.2",
GitCommit:"08e099554f3c31f6e6f07b448ab3ed78d0520507", GitTreeState:"clean", BuildDate:"2017-01-12T04:57:25Z",
GoVersion:"go1.7.4", Compiler:"gc", Platform:"linux/amd64"}
```

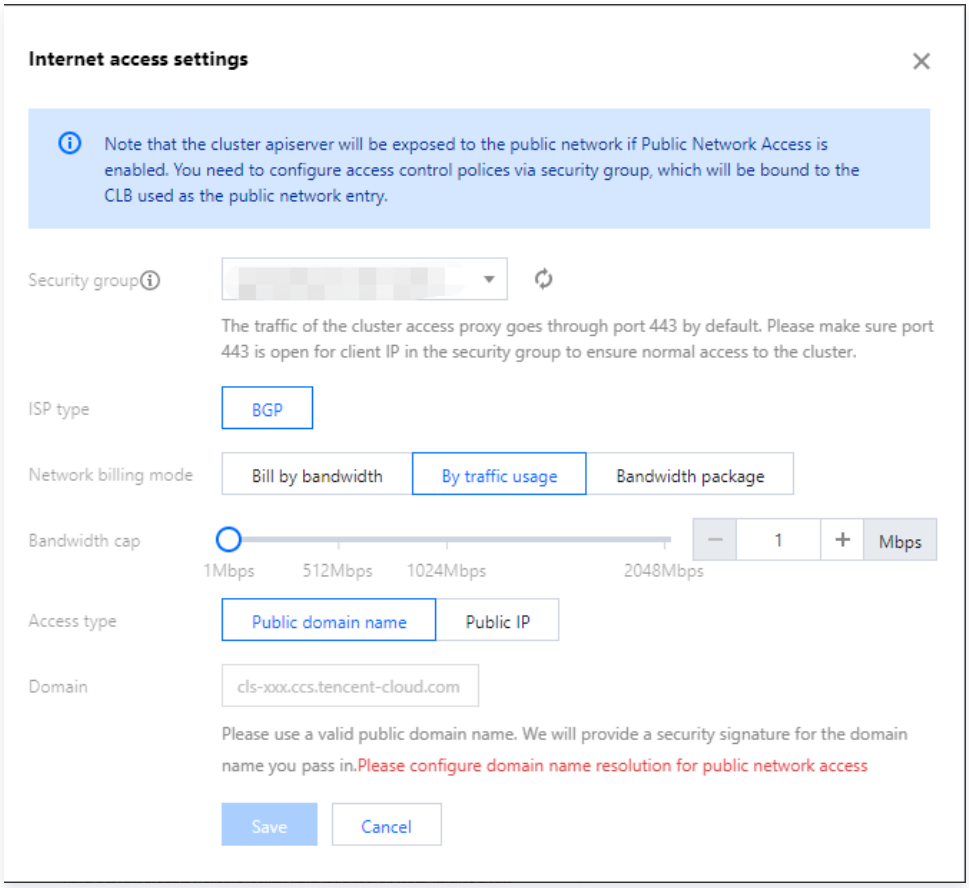
Step 2: Enable Cluster Access

1. Log in to the TKE console and click Cluster in the left sidebar.
2. In the **Cluster Management** page, select the region where the cluster is located, and click on the target **Cluster ID/Name** to enter the cluster details page.
3. On the Cluster Basic Information page, check the cluster access status as shown in the figure below:



Enabling public network access

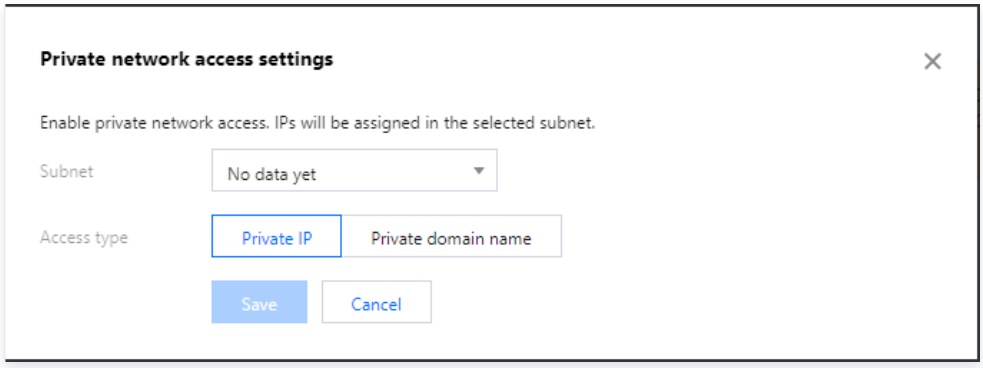
You need to configure relevant parameters to enable the public network access.



- **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**: For CLB-related parameters, please refer to the CLB Creation Guide and set them according to your actual requirements.
- **Access type**: After selecting a public domain name, you need to provide a custom domain name. We will generate a security signature for the domain name you provide, and you must configure the public network resolution yourself. If you choose the default CLB domain name, there is no need to manually configure the DNS record or any other operations.

Enabling private network access

You need to configure relevant parameters to enable private network access, as shown in the following image:

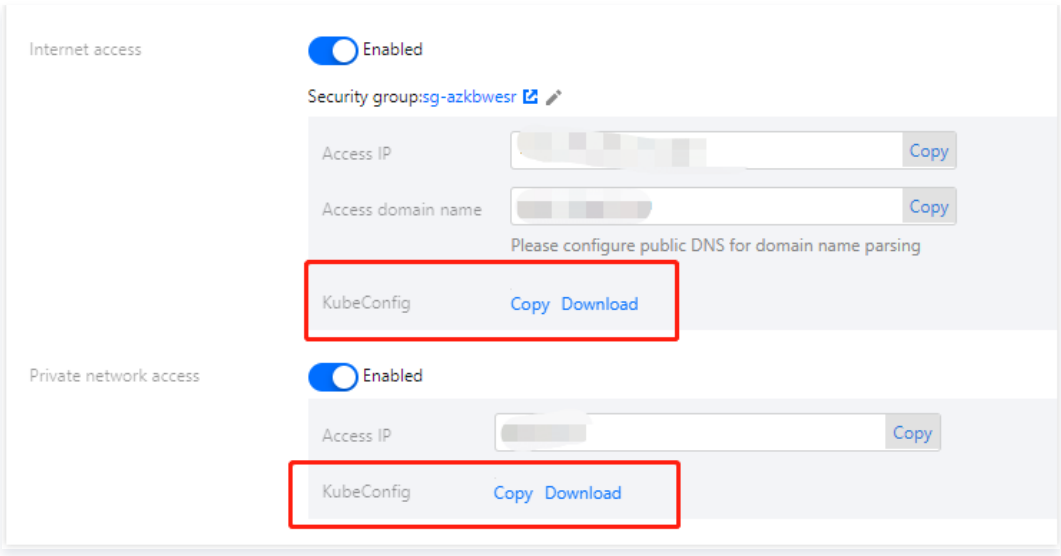


- **Subnet**: It is disabled by default. To enable the private network access, you need to configure a subnet. IP addresses are assigned from the configured subnet after the private network access is successfully enabled.
- **Access type**: If "private domain name" is selected, you need to pass in a 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.

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. In **Cluster Details > Basic Information**, view the "Cluster APIServer Information."
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. As shown in the image below:



Step 4: Configure kubeconfig and access the Kubernetes cluster

1. Configure the cluster credentials based on your specific requirements.
   Before configuration, determine whether the current access client has already been configured with access credentials for any cluster:
   - **If not**, and the `~/.kube/config` file is empty, you can directly copy the obtained Kubeconfig access credential content and paste it into `~/.kube/config`. If the client does not have a `~/.kube/config` file, you can create one directly.
   - **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 --merge --flatten > ~/.kube/config
     ```

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

     In this case, `~/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 completing the kubeconfig configuration, execute the following commands in sequence to view and switch contexts for accessing the current cluster.

   ```
   kubectl config get-contexts
   ```

   ```
   kubectl config use-context cls-3jju4zdc-context-default
   ```
3. Execute the following command to test if the cluster can be accessed normally.

   ```
   kubectl get node
   ```

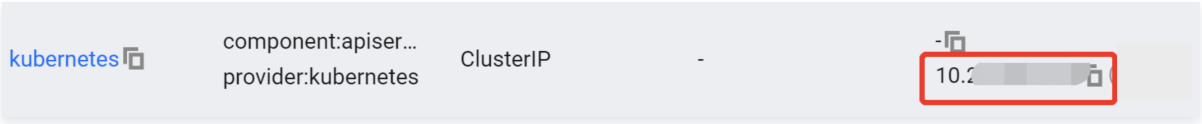
   If you cannot connect, please check whether the public network access or private network access entry has been enabled, and ensure that the access client is within the specified network environment.

Solution 3: Connecting to a Cluster via an Internal Node

Instructions

Step 1: Obtain the service IP of Kubernetes

1. Log in to the TKE console and click 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 details page, select **Service and Route > Service** in the left sidebar to obtain the Kubernetes service IP under the default namespace, as shown in the following image:



> **Note**
> Kubernetes service in ClusterIP mode is only suitable for access within the cluster.

Step 2: Configure kubeconfig and access the Kubernetes cluster

1. Log in to any node within the cluster and replace the clusters.cluster.server field in the `~/.kube/config` file with https://<IP>:443 using the Kubernetes service IP obtained in step 1, as shown below:

2. Execute the following command to test if the cluster can be accessed normally.

```
kubectl get node
```

Note

Overview of kubectl CLI

Kubectl is a command-line tool for operating Kubernetes clusters. This article covers kubectl syntax, common command operations, and provides examples. For detailed information on each command, including all main and subcommands, refer to the kubectl reference documentation or use the `kubectl help` command to view detailed help.

Upgrading a Cluster

Last updated: 2023-09-15 17:41:44

Scenario

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

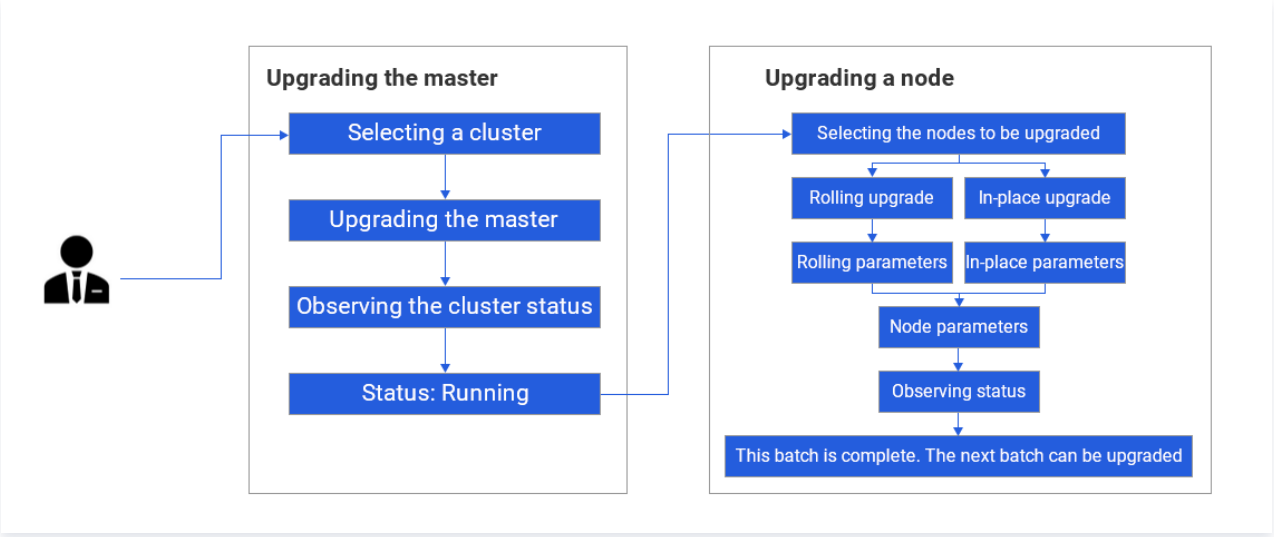
- **The upgrading action is irreversible. Perform this operation with caution.**
- Before upgrading the cluster, check whether the cluster is healthy. If the cluster is abnormal, you can fix it yourself or consult online.
- **Upgrade sequence**: when upgrading a cluster, you must first upgrade the Master, 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**.
- **HPA Invalidation Issue**: In Kubernetes versions prior to 1.18, the apiversion of the deployment object referenced in HPA might be `extensions/v1beta1`. However, in Kubernetes 1.18 and later, the apiversion of deployment is only `apps/v1`, which may cause HPA to become invalid after upgrading to Kubernetes 1.18.
  If you are using HPA, it is recommended to execute the following command before upgrading to switch the apiVersion in the HPA object to `apps/v1`.

```
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.

Instructions

The two steps to upgrade a cluster are upgrading the Master Kubernetes version and upgrading the Node Kubernetes version. The details are shown in the following image:



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:**
> - When upgrading major versions (e.g., from 1.12 to 1.14), if you have set custom parameters, you need to reset the custom parameters for the new version. The original parameters will not be retained. 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.

Supports and Limits

- **Before upgrading, please read the Upgrade Notice carefully.**
- 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.

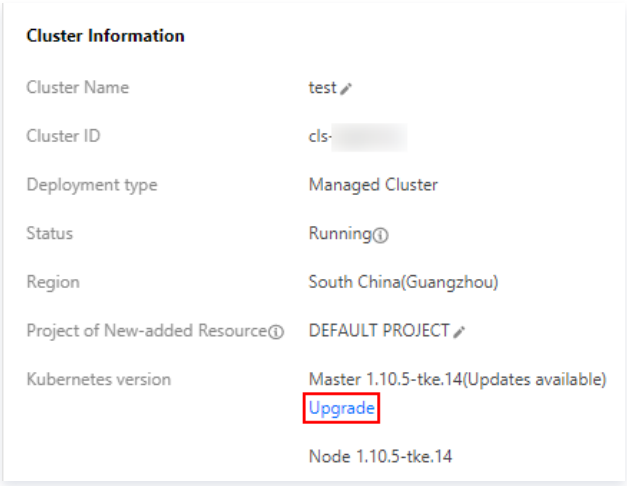
- **Pre-dependent component upgrade**: the pre-dependent components, such as monitoring components, will be upgraded to

prevent component exceptions due to compatibility problems.

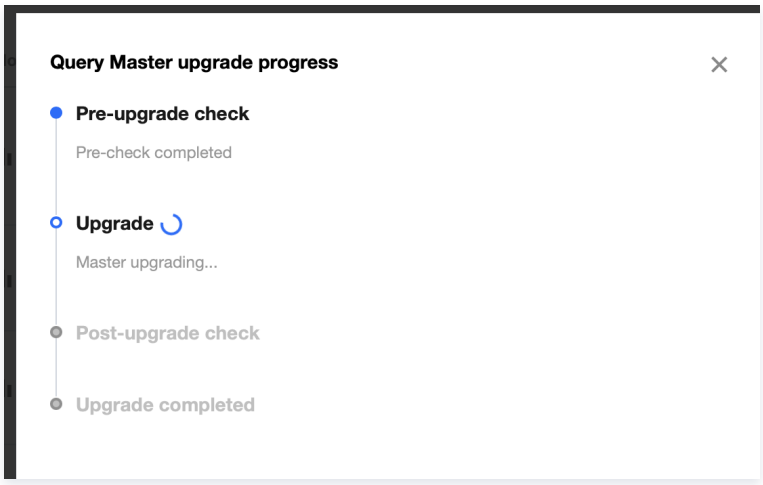
- **Master node component upgrade**: The corresponding components of all Masters will be upgraded in sequence. After all Masters' components are upgraded, the next component will be upgraded. TKE will first upgrade kube-apiserver, followed by kube-controller-manager and kube-scheduler, and finally kubelet. 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

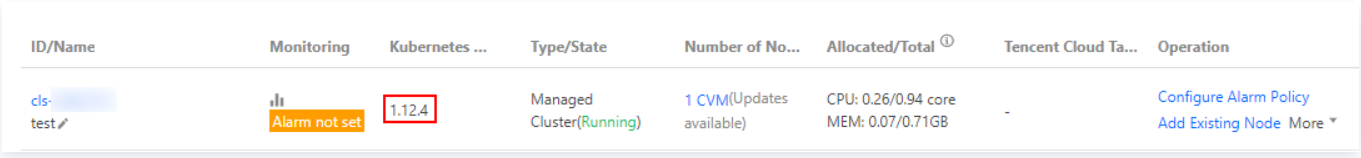
1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the **Cluster Management** page, select the ID of the desired cluster, and enter the cluster details page.
3. On the cluster details page, click **Basic Information** on the left.
4. In the "Cluster Information" page, click **Upgrade**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



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 on the cluster management page, as well as in the upgrade progress pop-up window. This includes the current upgrade progress, Master node upgrade progress (managed clusters do not display specific Master node lists), and upgrade duration. As shown in the image below:

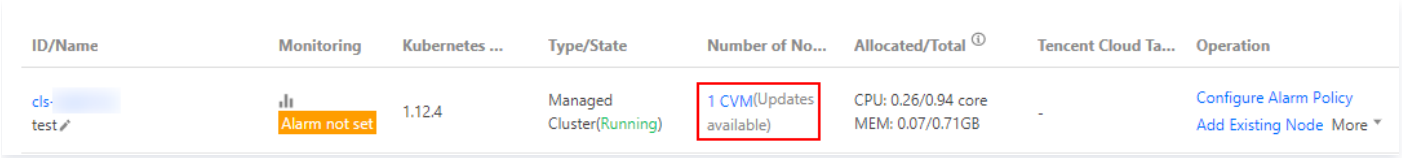


7. In this example, the Kubernetes version of the Master is 1.10.5 before the upgrade, and 1.12.4 after the upgrade, as shown below:



Upgrading the node Kubernetes version

After the cluster Master Kubernetes version is upgraded, the cluster list page will display that the cluster nodes have available upgrades, as shown below:



Supports and Limits

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

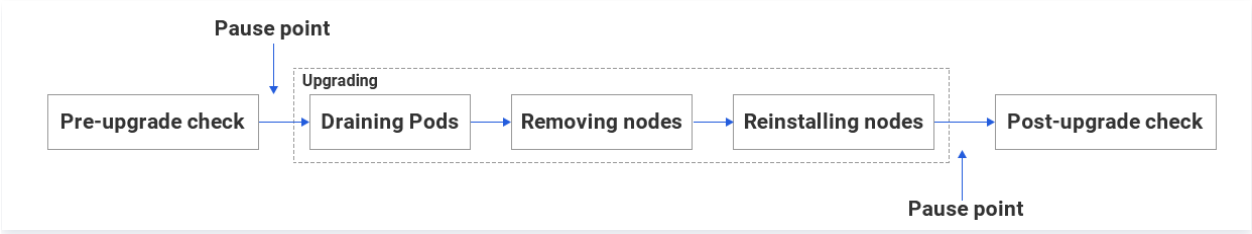
Selecting an upgrade method

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.

Reinstallation Rolling Upgrade Execution Principle

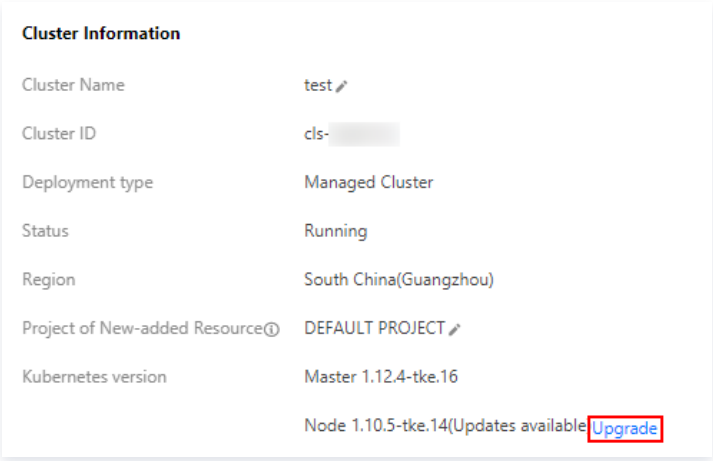
The node upgrade based on reinstallation uses a rolling upgrade method. Only one node will be upgraded at a time, and the next node will be upgraded only after the current node is successfully upgraded, as shown below:



- **Pre-upgrade check**: 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 check**: check whether the node is ready and schedulable, and check whether the current proportion of unavailable pods exceeds the max limit.

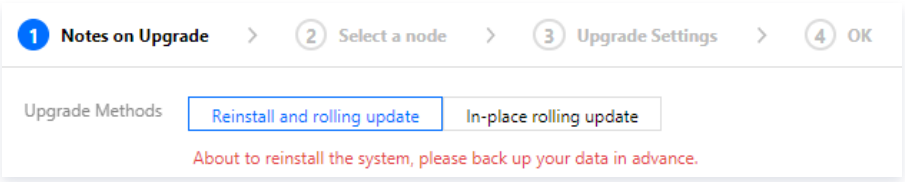
Reinstall and rolling upgrade (node Kubernetes version)

1. Log in to the TKE console and click 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:

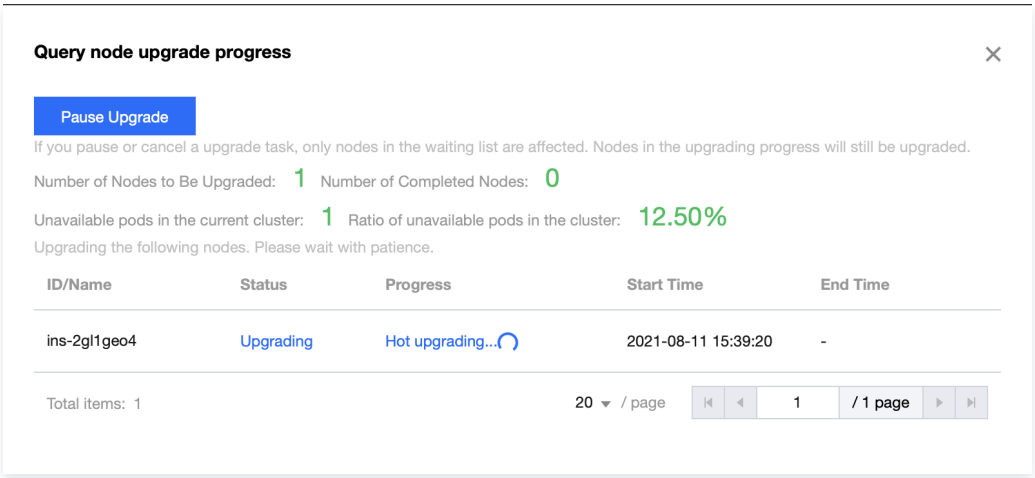


4. In the "**Notes on Upgrade**" step, select the upgrade method as **Reinstall and rolling upgrade** and carefully read the upgrade notice. Check **I have read and agreed to the above technical terms** and click **Next** as shown below:

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

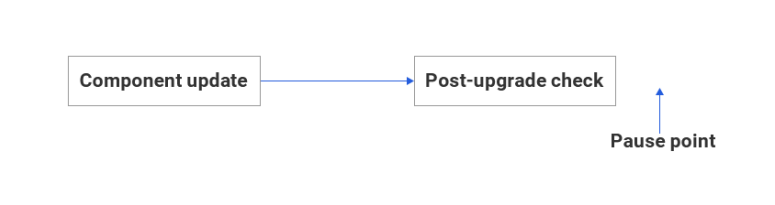


5. In the "Select a node" step, choose the nodes to be upgraded in this batch and click **Next**.
6. In the "Upgrade Settings" step, fill in the node information as needed and click **Next**.
7. In the "Confirmation" step, verify the information and click **Finish** to start the upgrade.
8. Monitor the node upgrade progress until all nodes are successfully upgraded.



In-place Rolling Upgrade Execution Principle

Nodes are upgraded in place using a rolling upgrade method, where only one node is upgraded at a time, and the next node is upgraded only after the current node is successfully upgraded. In-place upgrades currently support both major version upgrades and minor version upgrades within the same major version. As shown in the following diagram:

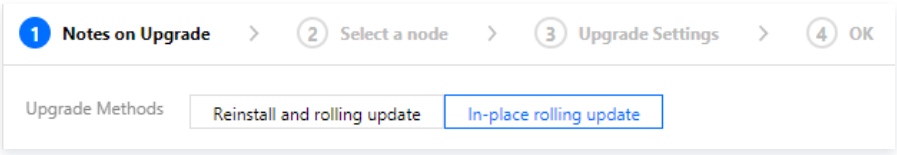


The steps are described as follows:

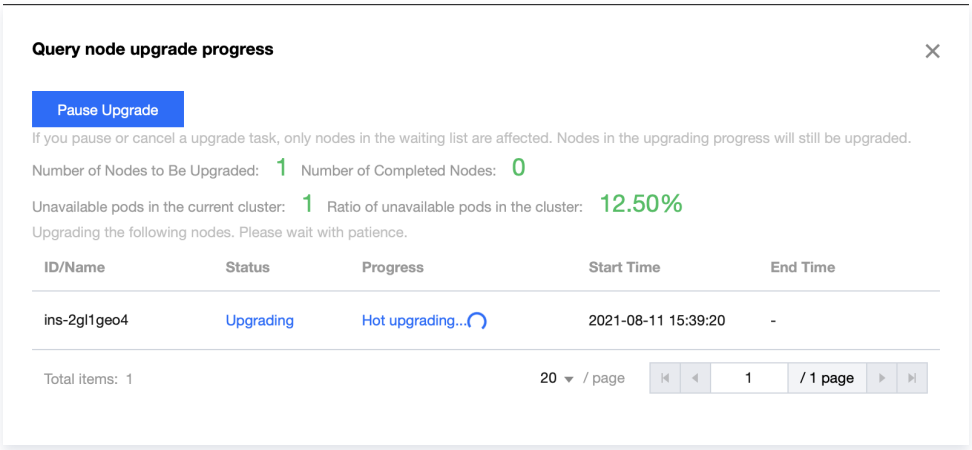
- **Component update**: replace and restart the kubelet and kube-proxy components on the node.
- **Post-upgrade check**: check whether the node is ready and check whether the proportion of currently unavailable pods exceeds the max limit.

In-place rolling upgrade

1. Log in to the TKE console and 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. In the cluster's "Basic Information" page, click **Upgrade** on the right side of the Node Kubernetes version.
4. In the **Notes on Upgrade**, select the upgrade method as "In-place rolling upgrade" and carefully read the upgrade notice. Check **I have read and agree to the above technical terms** and click **Next**. As shown below:



5. In the "Select a node Selection" step, choose the nodes to be upgraded in this batch and click **Next**.
6. In the "Confirmation" step, verify the information and click **Finish** to start the upgrade.
7. Monitor the node upgrade progress until all nodes are successfully upgraded.



Enabling IPVS for a Cluster

Last updated: 2023-09-15 17:42:09

Scenario

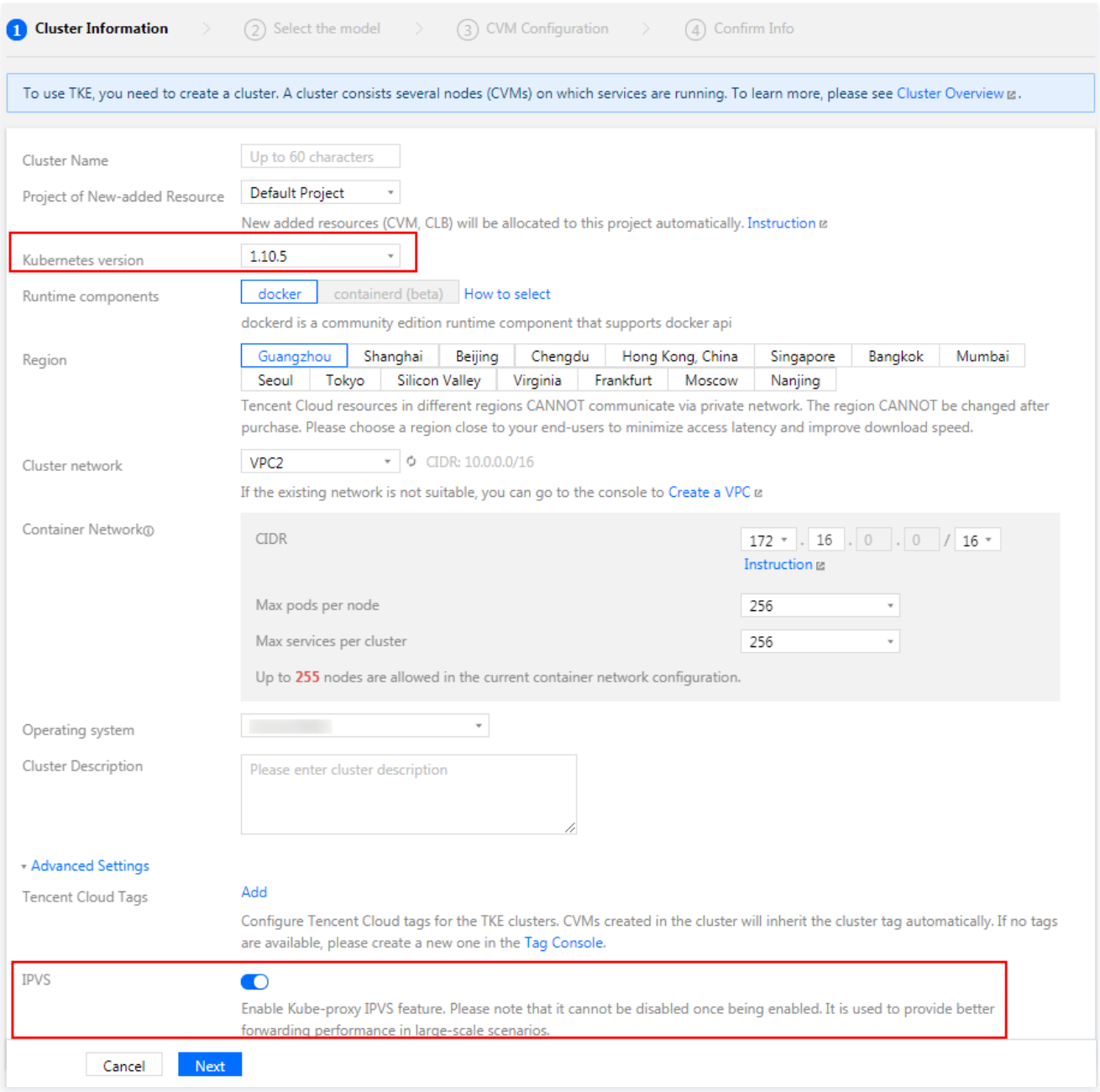
By default, Kube-proxy uses iptables to implement load balancing between Services and Pods. TKE supports quickly enabling IPVS-based traffic handling and load balancing. Enabling IPVS is suitable for large-scale clusters, providing better scalability and performance.

Supports and Limits

- 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.

Instructions

1. Log in to the TKE console.
2. Refer to Create Cluster, on the "Create Cluster" page, set the "Kubernetes Version" to a version higher than 1.10, and click **Advanced Settings** to enable "IPVS Support". As shown in the image below:



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

Custom Kubernetes Component Launch Parameters

Last updated: 2023-09-15 17:46:14

Scenario

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.

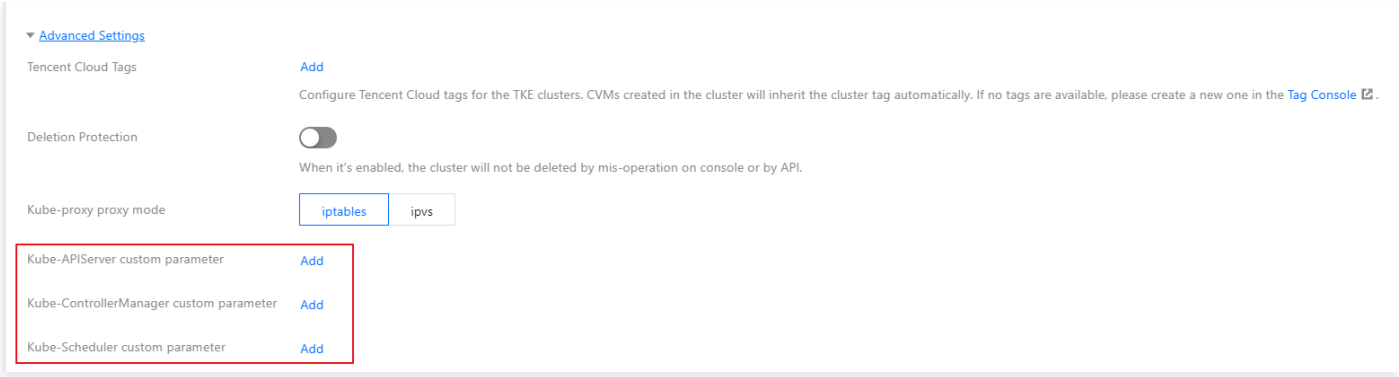
Supports and Limits

- To use the custom Kubernetes component launch parameters feature, please contact us online for application.
- 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.

Notes

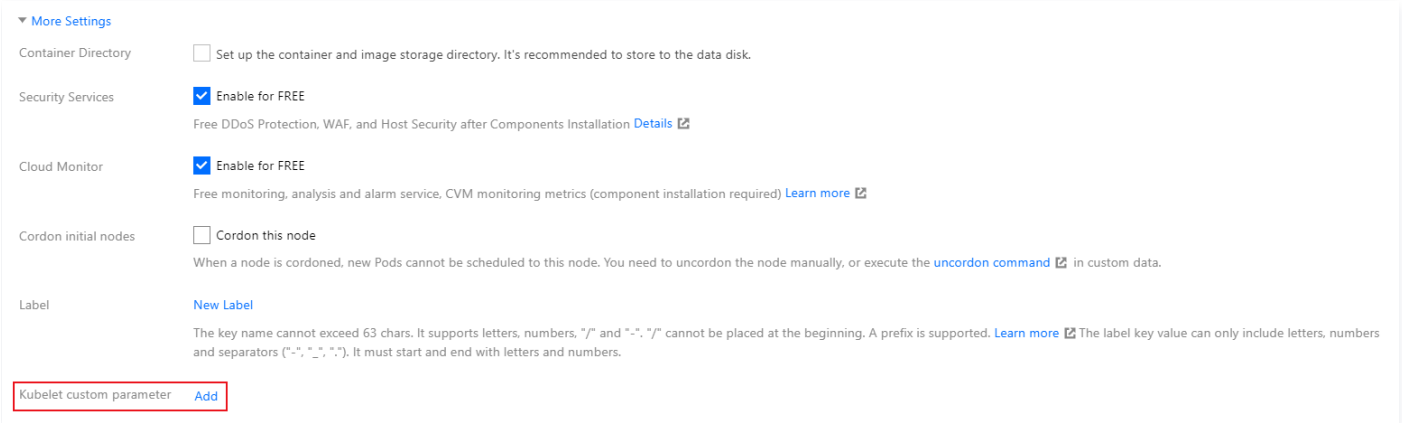
Configuring custom Kubernetes component parameters when creating a cluster

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



Configuring the custom Kubelet parameters of a node

On the "Create Cluster Node", "Add Existing Node", "Add Node Pool", and "Add Node" pages, you can set custom Kubelet parameters for nodes, as shown in the following figure:



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. In the **Basic Information** of the cluster, click **Upgrade** next to the Kubernetes version. At the same time, configure the Kubernetes component launch parameters.

Image
Image Overview

Last updated: 2023-09-15 17:46:55

Overview

This document describes three 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.
> - Custom and marketplace images are non-standard operating environments. TKE has not undergone compatibility adaptation for these images, and users are responsible for ensuring their usability in a Kubernetes environment. In principle, TKE does not provide SLA services or technical support for such 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.
- **Market image**: It is provided for specific use cases, such as qGPU sharing. It is available to all users and integrated with certain applications in addition to the operating system.

Supports and 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**
> If TKE plans to adjust the image logic in the future, we will notify you at least one week in advance via site messages, text messages, and emails. Please feel confident using the service. Changes in image logic will not affect existing nodes created with older image versions. For optimal results, we recommend using the latest base images.

Image ID	Os Name	OS Name Displayed in the Console	OS Type	Release Status	Remarks
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	It is recommended to use the latest release of TencentOS Server. Kernel version: 5.4.119
img-hdt9xxkt	tlinux2.4x86_64	TencentOS Server 2.4Formerly known as	Tlinux	Full release	Kernel version: 4.14.105

		Tencent linux release 2.4 (Final)			
img-22trbn9x	ubuntu20.04x86_64	Ubuntu Server 20.04.1 LTS 64bit	Ubuntu	In beta testing, please submit a ticket to apply.	Ubuntu 20.04.1 Public Kernel
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	In beta testing, 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: 2023-09-15 17:50:31

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 Management
Node Overview

Last updated: 2023-09-15 17:50:50

Feature Overview

Nodes are the fundamental elements of container clusters. Depending on the business requirements, nodes can be either virtual machines or physical machines. Each node contains the essential components required to run Pods, including Kubelet, Kube-proxy, and others.

Node-related Operations

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

Node Lifecycle

Last updated: 2023-09-15 17:51:09

Notes on Node Lifecycle Status

Status	Note
Healthy	The node is running normally and connected to the cluster.
Abnormal	The node is running abnormally 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 States	Please refer to Cloud Server Lifecycle.

Removing a node

Last updated: 2023-09-15 17:58:22

Scenario

This document guides you on how to remove nodes from a cluster.

Supports and Limits

- A prepaid node will not be terminated after it is removed from the cluster.
- When removing a pay-as-you-go node, you can choose whether to terminate it or not. If you choose not to terminate it, billing will continue.
- Keep in mind that if a node is removed from and then added back to the cluster, the system will be reinstalled.

Instructions

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the "Cluster Management" list page, click the ID or name of the cluster from which you want to remove a node, and enter the cluster details page.
3. Select **Node Management** > **Nodes** from the left navigation menu to access the "Node List" page.
4. In the node list, select the row of the node you want to remove and click **Remove**.
5. In the "Are you sure you want to remove the following nodes?" pop-up window, click **Confirm** to complete the removal.

Draining or Cordoning a Node

Last updated: 2023-09-26 10:16:43

Scenario

This document explains how to drain or cordon a node.

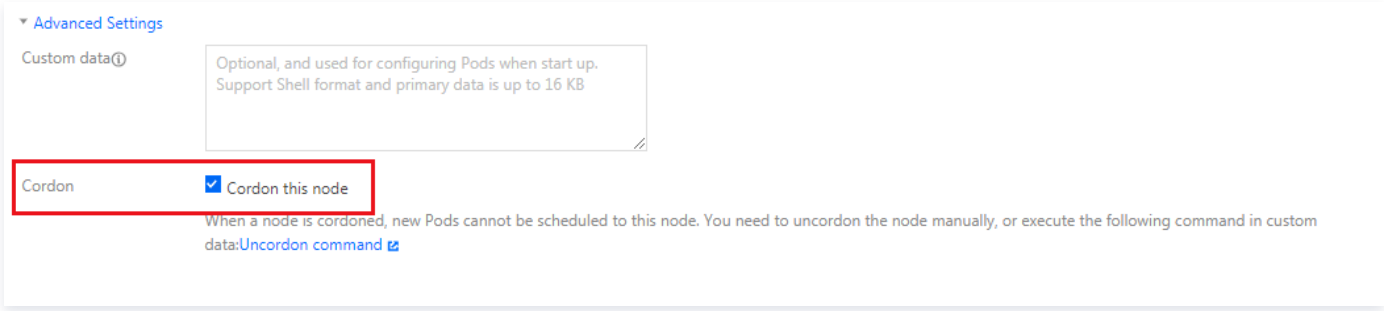
Instructions

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:

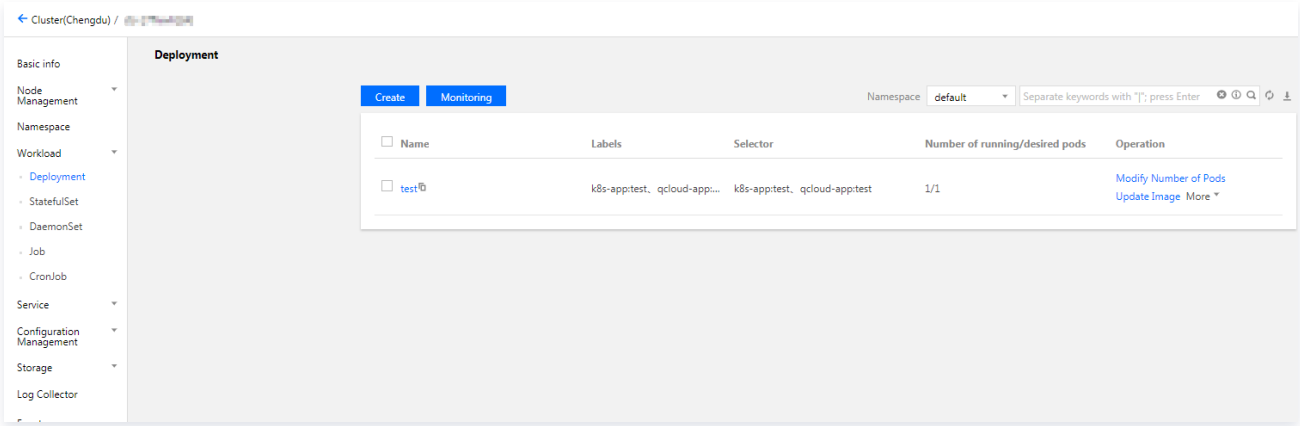
Method A

When adding a new node, on the "CVM Configuration" page, click **Advanced Settings** and select "Enable Cordon".

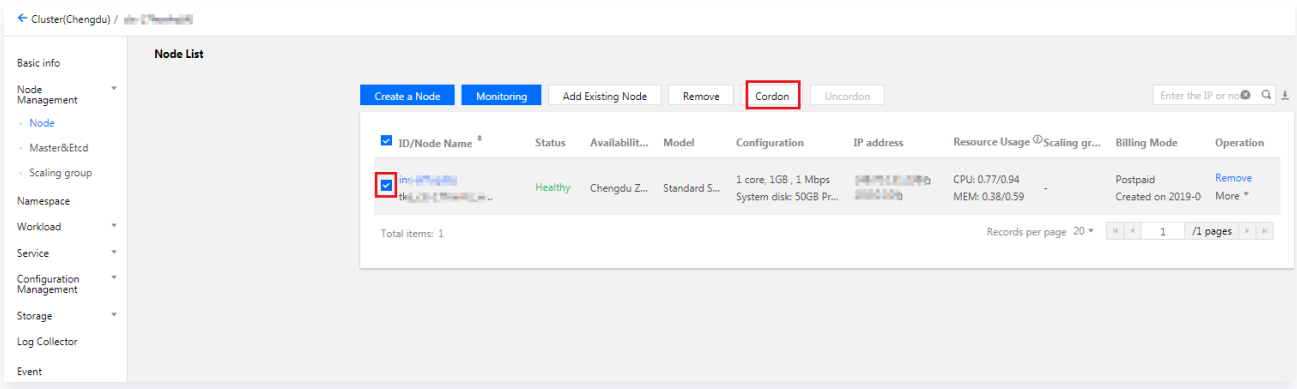


Method B

1. Log in to the TKE console.
2. In the left navigation pane, click Clusters to access the cluster management page.
3. Click the ID or name of the cluster containing the node you want to cordon, and enter the cluster management page, as shown below:



4. In the left navigation pane, select **Node Management > Nodes** to access the "Node List" page.
5. In the node list, select the row of the node you want to cordon and click **Cordon**. As shown in the following image:



6. In the pop-up dialog box, click **Confirm** to complete the cordon process.

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

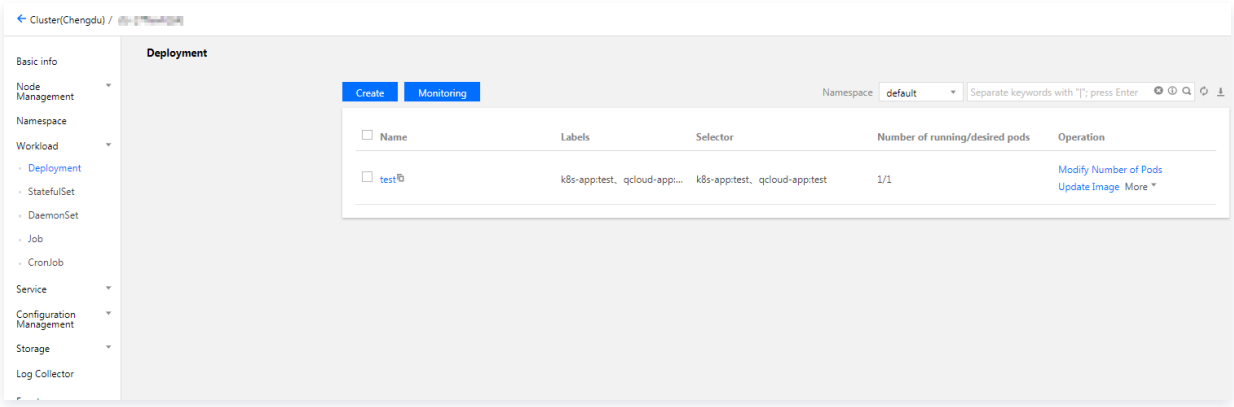
When you create a node by running a script, you can uncordon it by adding a command for uncordoning the node in the script. Below is an example:

```
#!/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
```

`kubectl uncordon` command is used to uncordon a node.

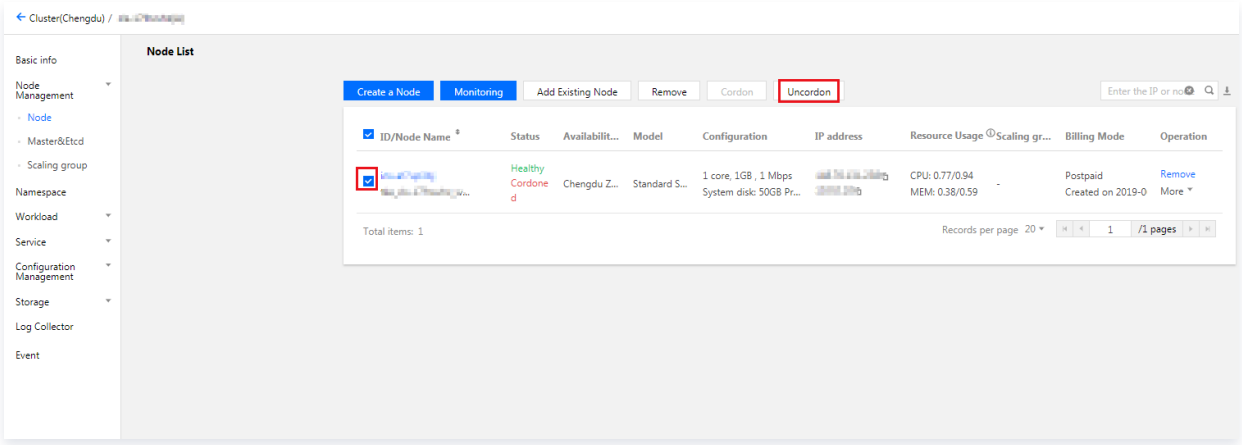
Method B

1. Log in to the TKE console.
2. In the left navigation pane, click Clusters to access the cluster management page.
3. Click the ID or name of the cluster containing the node you want to uncordon, and go to the management page of that cluster, as shown below:



4. In the left navigation pane, select **Node Management > Nodes** to access the "Node List" page.

5. In the node list, select the row of the node you want to uncordon, and click **Uncordon**. As shown in the following image:



6. In the pop-up dialog box, click **Confirm** to complete the uncordoning process.

Draining a Node

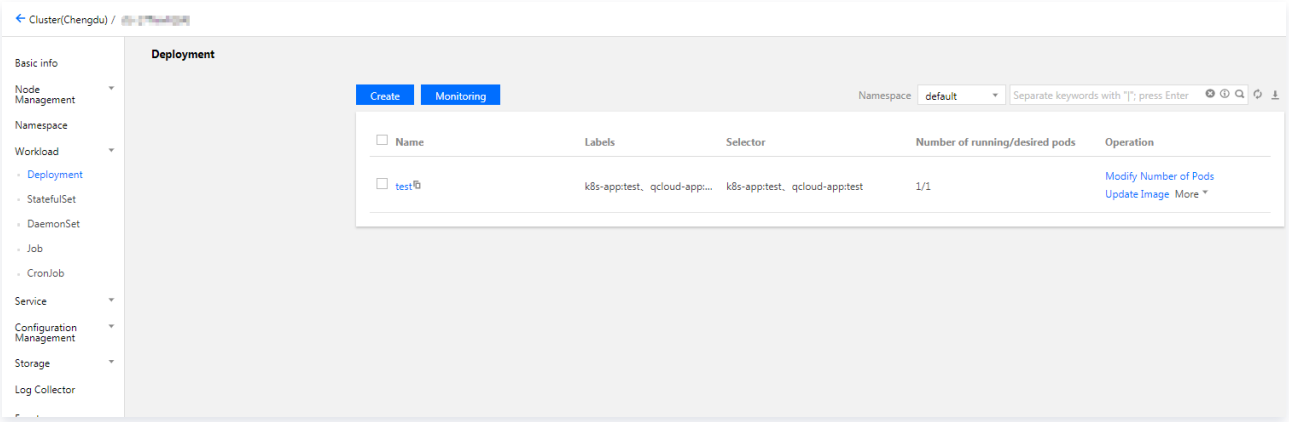
Overview

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.

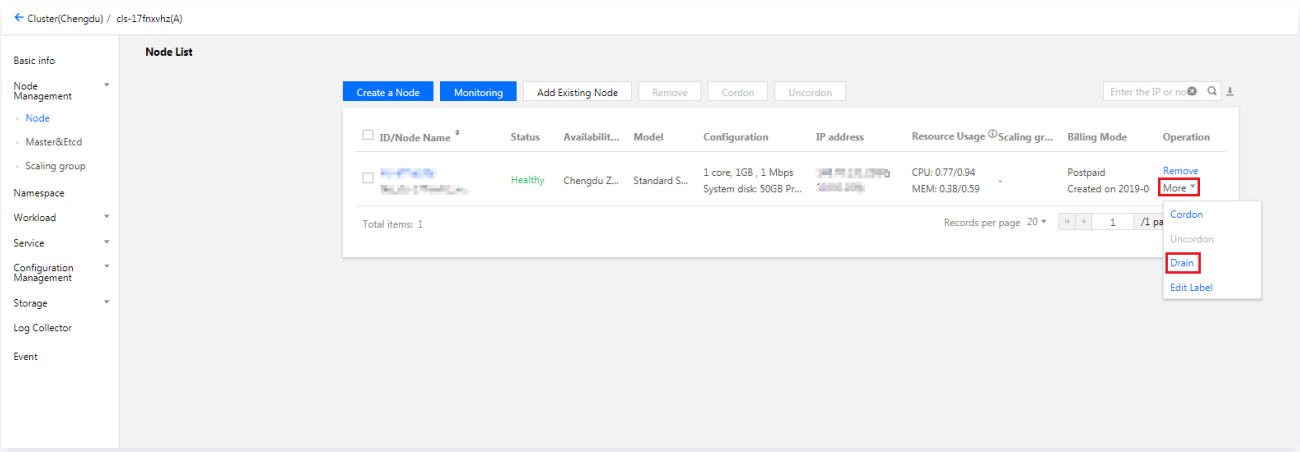
> **Note**
> For locally stored Pods, data will be lost after they are drained. Please be cautious when doing so.

Procedure

1. Log in to the TKE console.
2. In the left navigation pane, click Clusters to access the cluster management page.
3. Click the ID or name of the cluster containing the node you want to drain, and go to the cluster management page, as shown below:



4. In the left navigation pane, select **Node Management > Nodes** to access the "Node List" page.
5. In the row of the node you want to drain, click **More > Drain**, as shown below:



6. In the pop-up dialog box, click **Confirm** to complete the eviction.

Setting the Startup Script of a Node

Last updated: 2023-09-26 10:17:03

Scenario

Configuring a node's startup script can help you perform initialization tasks on your node before it becomes ready. The configured script will run when the node starts up. If you purchase multiple cloud servers at once, the custom data will be executed on all of them.

Usage Limits

- It is recommended that you restrain yourself from modifying configurations such as Kubelet, kube-proxy, and docker on the TKE node through the startup script.
- If the startup script fails to run, it will not be retried. You should ensure the executability and retry mechanism of the script.
- The script and its generated log files can be viewed at the node's path: `/usr/local/qcloud/tke/userscript` .

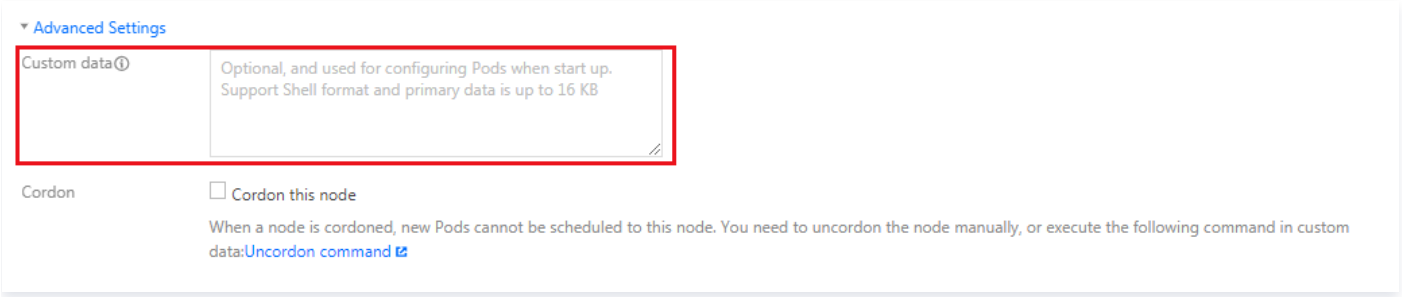
Instructions

You can set the node startup script in the following three scenarios:

- Setting the node startup script when creating a cluster or node
- Setting the node startup script when adding an existing node
- Setting the node startup script when creating a scaling group

When creating a cluster or adding new nodes

On the Create Cluster and Add Node pages, navigate to the "Cloud Server Configuration" section, click **Advanced Settings**, and enter custom data and startup scripts, as shown in the figure below:

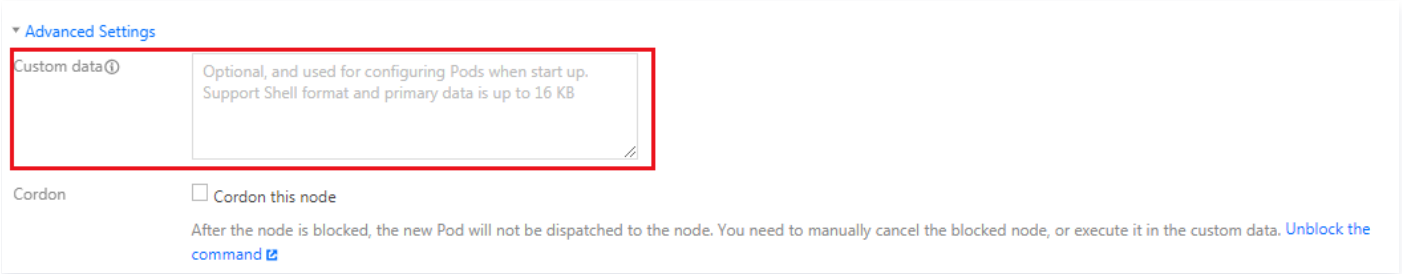


When adding an existing node:

Add existing node: On the "Cloud Server Configuration" page, click **Advanced Settings** and enter custom data for the startup script.

When creating a scaling group

When creating a scaling group, on the "Launch Configuration" page, click **Advanced Settings** and fill in the custom data and startup script. See the following figure:



Setting a Node Label

Last updated: 2023-09-26 10:17:29

Scenario

This document guides you through the process of setting a node Label.

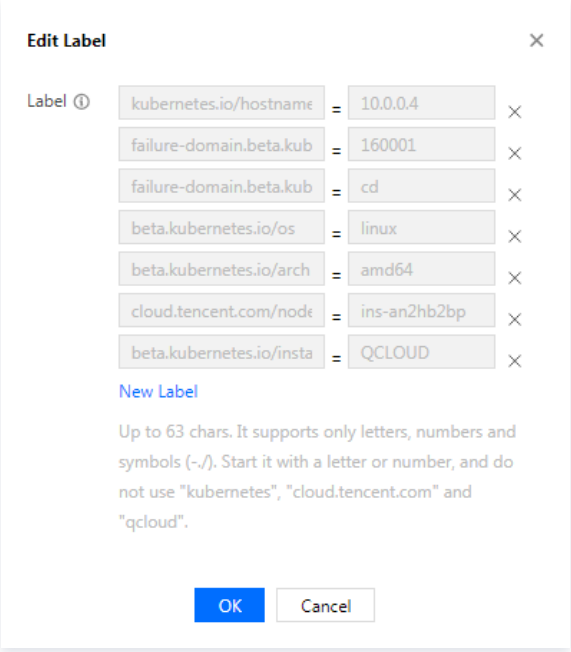
Usage Limits

- Labels related to *kubernetes* and *qcloud* cannot be edited or deleted.
- *kubernetes* and *qcloud* 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.

Instructions

Setting Node Labels via Console

1. Log in to the TKE console.
2. In the left navigation bar, click Cluster to enter 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 navigation pane, select **Node Management** > **Nodes** to access the "Node List" page.
5. Choose **More** > **Edit Label** on the right of the target node.
6. In the "Edit Label" pop-up window, edit the Label and click **OK**, as shown in the image below:



Setting Node Label with Kubectl

1. Install kubectl and connect to a cluster. For detailed operations, see Connecting to a Cluster.
2. Run the following command to set a node Label.

```
kubectl label nodes <node-name> <label-key>=<label-value>
```

3. Run the following command to view the node Label.

```
kubectl get nodes --show-labels
```

Information similar to the following is returned:

```
NAME          STATUS    ROLES     AGE       VERSION        LABELS
172.17.124.5   Ready     <none>    12d       v1.10.5-tke.3
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=QCLOUD,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=sh,failure-
domain.beta.kubernetes.io/zone=200001,kubernetes.io/hostname=172.17.124.5
172.17.124.8   Ready     <none>    12d       v1.10.5-tke.3
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=QCLOUD,beta.kubernetes.io/os=linux,failure-
domain.beta.kubernetes.io/region=sh,failure-
domain.beta.kubernetes.io/zone=200001,kubernetes.io/hostname=172.17.124.8
```

Normal Node Management
Supported CVM Instance Types for Ordinary Nodes

Last updated: 2023-09-26 10:19:42

Example

To better provide container services, TKE conducts availability tests for container environments on supported instance types of ordinary nodes. The tests mainly cover multiple application modules such as container network modes, storage, public images, node initialization, and GPU drivers. The currently supported ordinary node instance types that can be created in the container service console are shown in the table below.

Note
The instance types supported for creating ordinary nodes in the container service console do not correspond one-to-one with those in the CVM console. If your business requires adaptation to new instance types, you can submit a ticket for assistance.

Supported Instance Types for Ordinary Nodes

Instance Type	Model
Standard	S1、S2、S2ne、S3、S3ne、S4、S4m、S5、S5se、S5t、S6、S6t、SA1、SA2、SA2a、SA3、SK1、SN3ne、SR1、SW3a、SW3b、SW3ne
Exclusive	RS2t、RS3t、RS4t
Compute	C2、C3、C4、C5、C6、TC3、CN3
High I/O Type	I1、I2、I3、I6t、IT2、IT3、IT3a、IT3b、IT3c、IT5
Memory optimized	M1、M2、M3、M4、M5、M6、M6ce、M6mp、M6p、MA2、MA3
High performance	HCCG5v、HCCIC5、HCCPNV4h、HCCTG5v、HCCPNV4sne（HCCPNV4sn）、HCCPNV5v、
GPU model	GI1、GI3X、GN10S、GN10X、GN10Xp、GN6、GN6S、GN7、GN8、GNV4、GT4、PNV4、PNV4ne
Big Data	D1、D2、D3
Bare Metal	BMD3c、BMD3s、BMDA2、BMI5、BMIA2、BMIA2m、BMM5r、BMS4、BMSA2、BMSC4、BMM6i、BMTGC39me（BMGC39me）、BMG5e、BMG5n、BMG5i、BMGY5、BMGNV4
Others	CHC、CN10X、BC1

Note:

1. For bare metal cloud servers, please visit Instance Specifications to confirm the model configuration, such as whether it supports attaching elastic network interfaces and cloud disks. Models that do not support attaching elastic network interfaces cannot be added as ordinary nodes to clusters using the VPC-CNI network mode.
2. HCCG5v, HCCIC5, HCCPNV4h, and HCCTG5v only support public images CentOS 7.6, Ubuntu18.04.1, and TencentOS Server 2.4 (TK4).
3. ARM-based instance types SR1 and SK1 only support TencentOS Server 2.4 for ARM 64 (TK4) and CentOS 8.2 (ARM) images.
4. The qGPU feature is only supported on T4 and V100 card models. For more information, please refer to Using qGPU.
5. Red Hat images only support SA2, S5, C3, and C4 instance types.

Node Pool Overview

Last updated: 2023-09-26 10:20:21

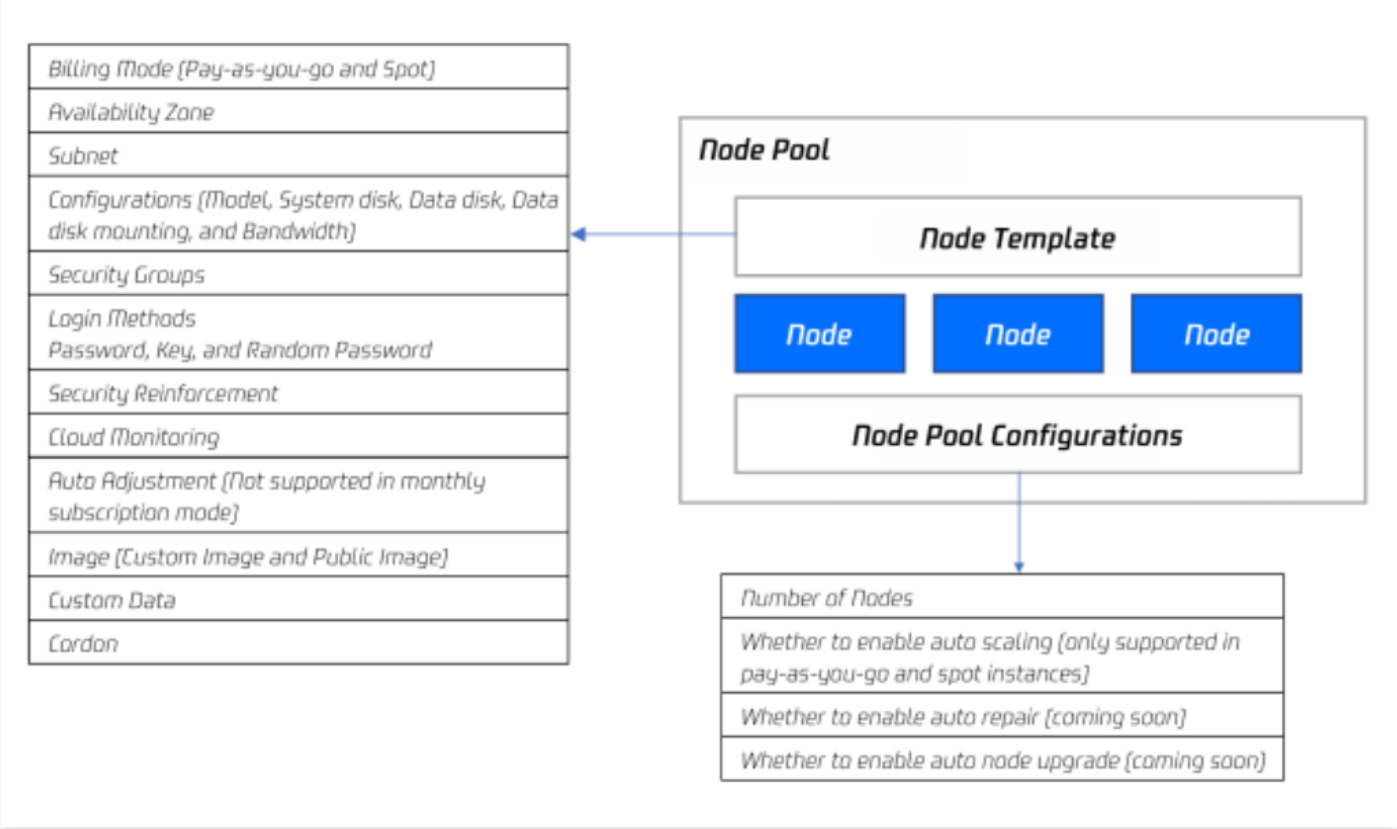
Feature Overview

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 the scale-in conditions are met, for example, when a node is idle, scale-in will be triggered automatically, reducing resource costs.

Product Architecture

The overall architecture of a node pool is as follows:



Under normal circumstances, nodes within a node pool share the following common attributes:

- Node-level operating system
- Billing type (currently supports pay-as-you-go, spot instances, and monthly subscription).
- 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

Scenarios

When your business requires large-scale clusters, it is recommended to use node pools for node management to improve the usability of large-scale clusters. The table below describes various large-scale cluster management scenarios and demonstrates the role of node pools in each scenario:

Scenes	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.

Relevant 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 scaling group	Auto scale-out or manual adjustment of the node quantity	Supported	Auto scale-in or manual adjustment of the node quantity	Supported
Nodes outside the scaling group	Manually added to the node pool by users	Not required	Manually removed by users	Not required

How the Node Pool Auto Scaling Works

Please read how the node pool auto scaling works before using it.

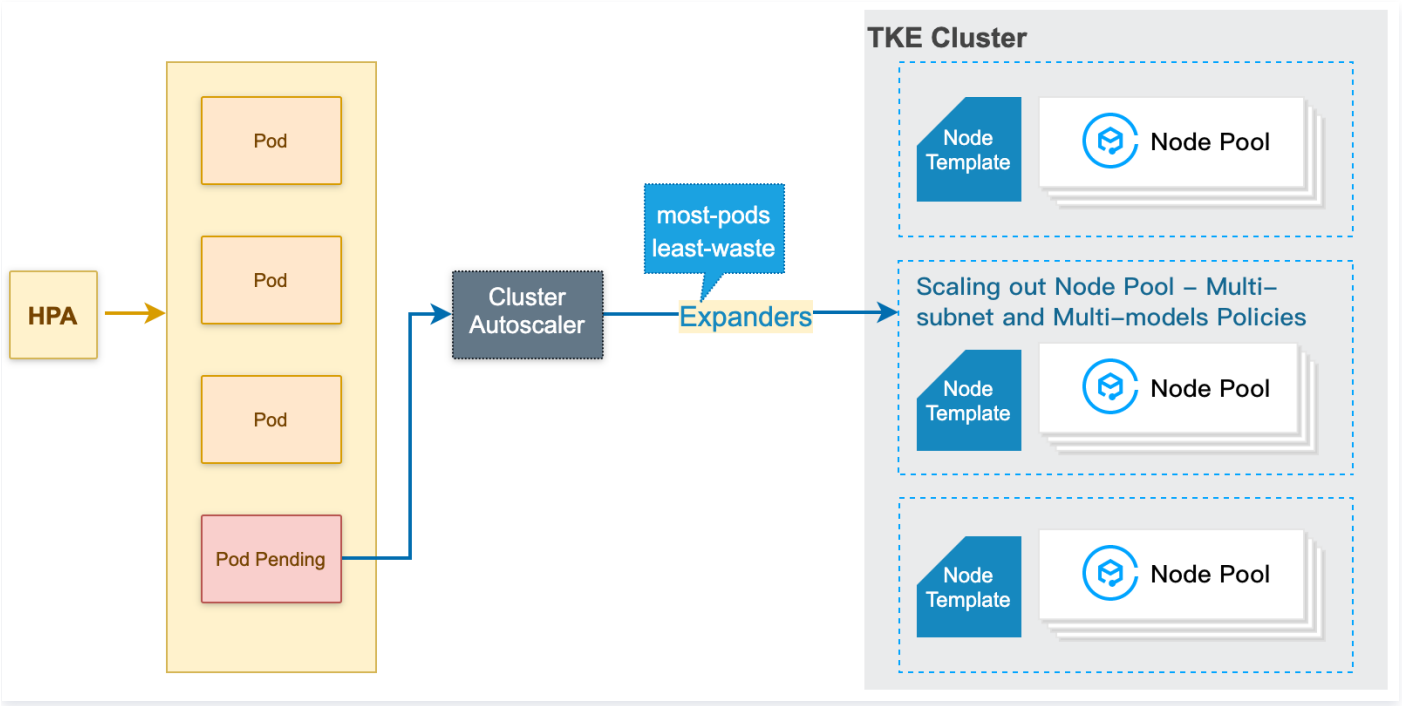
How the node pool auto scale-out works

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**
> When scaling out a specific node pool, the expansion will be based on the subnets set during the creation of the node pool and the subsequent multi-model configurations. Generally, the **multi-model policy is prioritized, followed by the multi-**

availability zone/subnet policy.
For example, if you have configured multiple models A and B, and multiple subnets 1, 2, and 3, the system will attempt to scale out in the order of A1, A2, A3, B1, B2, and B3. If A1 is sold out, the system will try A2 instead of B1.

The following figure shows how the node pool auto scale-out works:



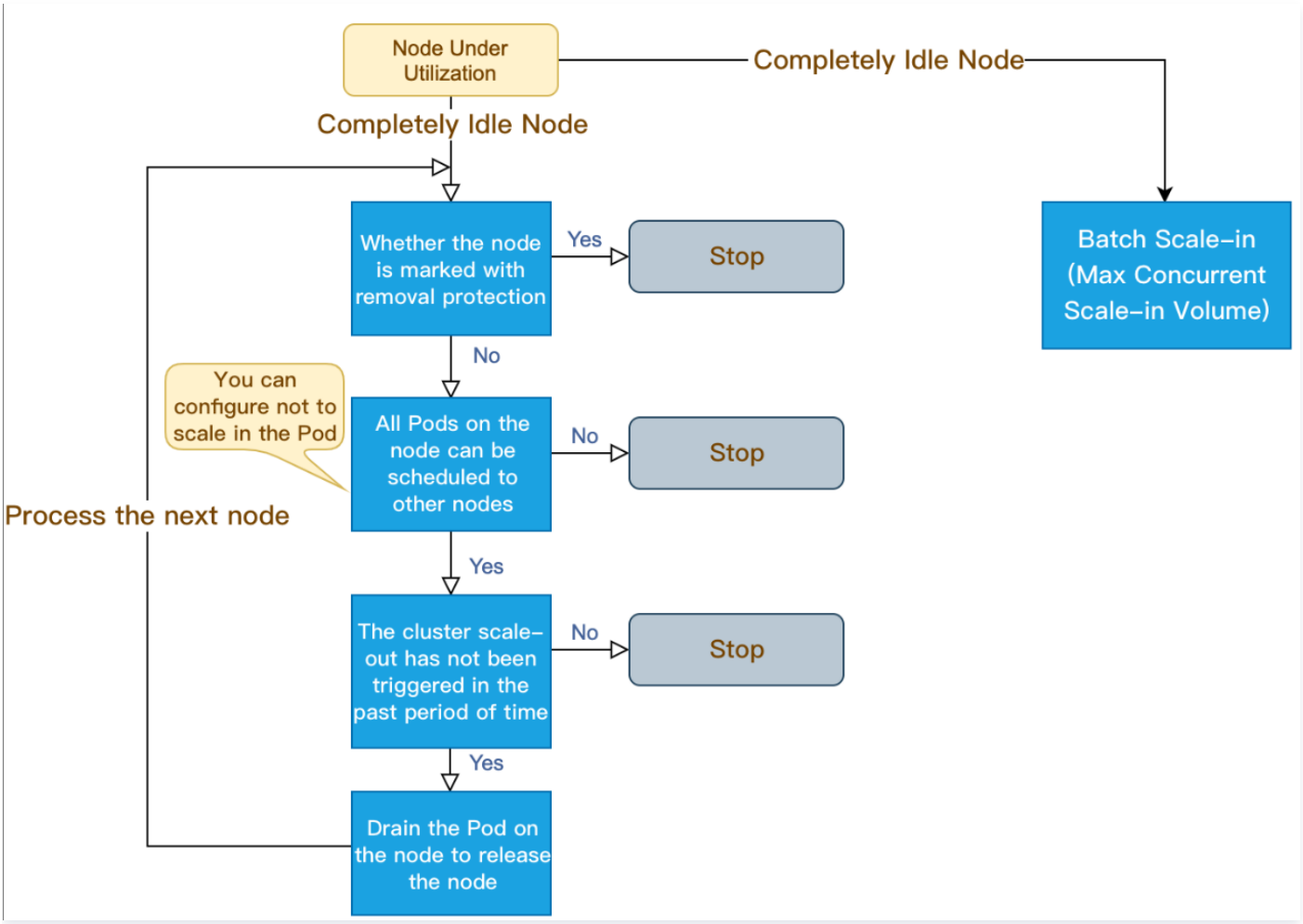
How the node pool auto scale-in works

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:
   - Node idle duration requirement (default is 10 minutes).
   - Cluster scale-out buffer time requirement (default is 10 minutes).
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 the kube-system namespace that are 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. After CA drains the Pods from a node, it releases or shuts down the node (excluding pay-as-you-go nodes).
   - 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:



Features and Notes

Feature	Note	Supports and Limits
Creating a Node Pool	Adds a node pool	• We recommend that you create no more than 20 node pools for a single cluster. • Billing mode is an attribute of the node pool. For details on monthly subscription node pools, please refer to Creating Monthly Subscription Node Pool Instructions. Do not convert pay-as-you-go nodes in the node pool to monthly subscription nodes; it is recommended to create a new monthly subscription node pool.
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 is automatically adjusted according to the workload of the cluster.	Do not enable or disable auto scaling in the Scaling Group page.
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. • Please 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 Taints attributes 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.	Please 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 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.	This operation is irreversible. Ensure that you are familiar with the features of node pools before conversion.

Related Actions

You can log in to the TKE 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: 2023-09-26 10:20:49

Scenario

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.

Preparations

- You are familiar with the Node Pool Concepts.
- You have created a cluster.

Notes

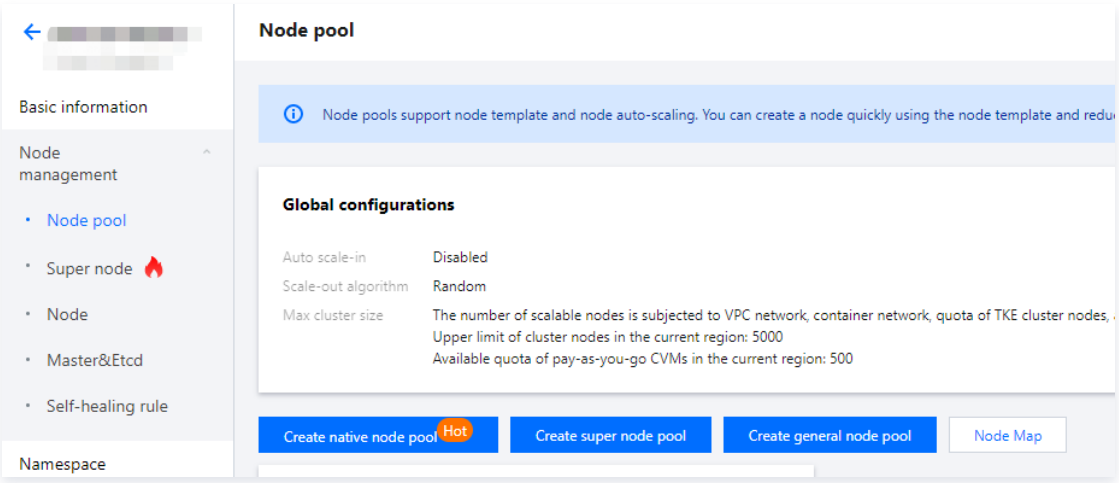
Monthly Subscription Node Pool

Nodes in a monthly subscription node pool are prepaid machines, so there will be some limitations on their auto-scaling capabilities, as described below:

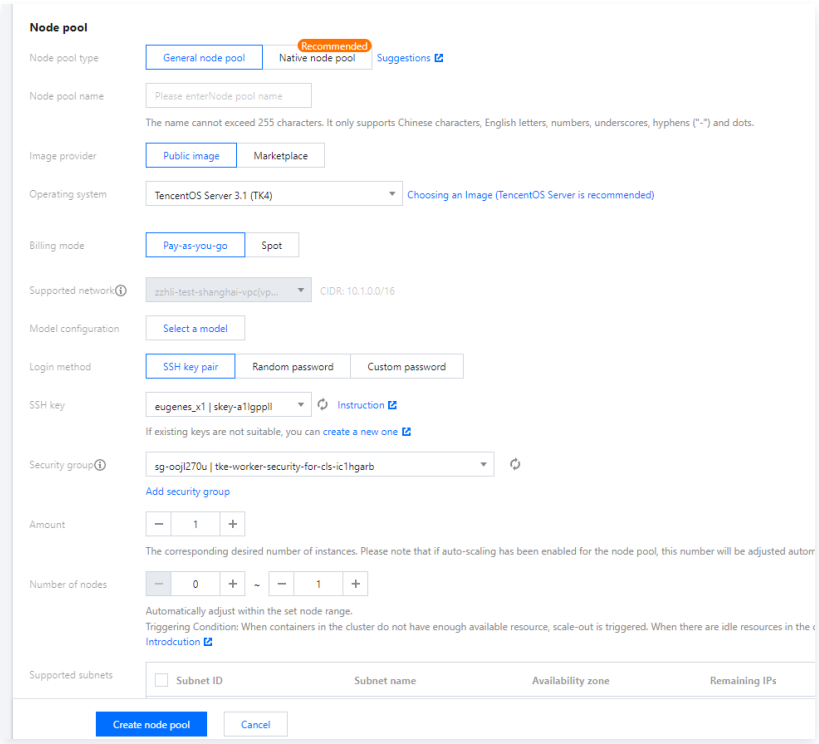
- Monthly subscribed node pools support auto scale-out.
- Monthly subscription node pools do not support elastic scale-in (elastic scale-in has a certain degree of randomness; we recommend managing prepaid machines through **manual removal** of nodes).
- When removing monthly subscribed nodes from a node pool, you can choose whether to keep them in the cluster. TKE **will not terminate the monthly subscribed nodes for you**. Please remove the nodes from the cluster and terminate them in the CVM console. For more information, see Refund for Monthly Subscribed Instances.

Instructions

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. Select **Node Management** > **Node Pool** in the left sidebar to access the "Node Pool List" page, as shown below:



4. Click **Create Node Pool** to navigate to the "New Node Pool" page, and configure the settings as per the instructions below. See the following image:



- **Node Pool Name**: you can customize the name of the node pool based on service requirements to facilitate subsequent resource management.
- **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**: Three billing modes are available: **Pay-as-you-go**, **Spot pricing**, and **Monthly subscription**. Please choose according to your actual needs. For more information, see Billing Mode Comparison.

> **Note**
> Monthly subscription node pools do not support auto-scaling. You can expand the capacity by manually adjusting the number of nodes in the node pool.

- **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**: you can select the model by specifying the number of CPU cores, memory size, and instance type. For more information, see Instance Types and Customizing Linux CVM Configurations.
  - **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 needed. 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,	No option is selected by default. If you select any of these options, you must specify the cloud disk settings and formatting settings.

and GPU	
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.

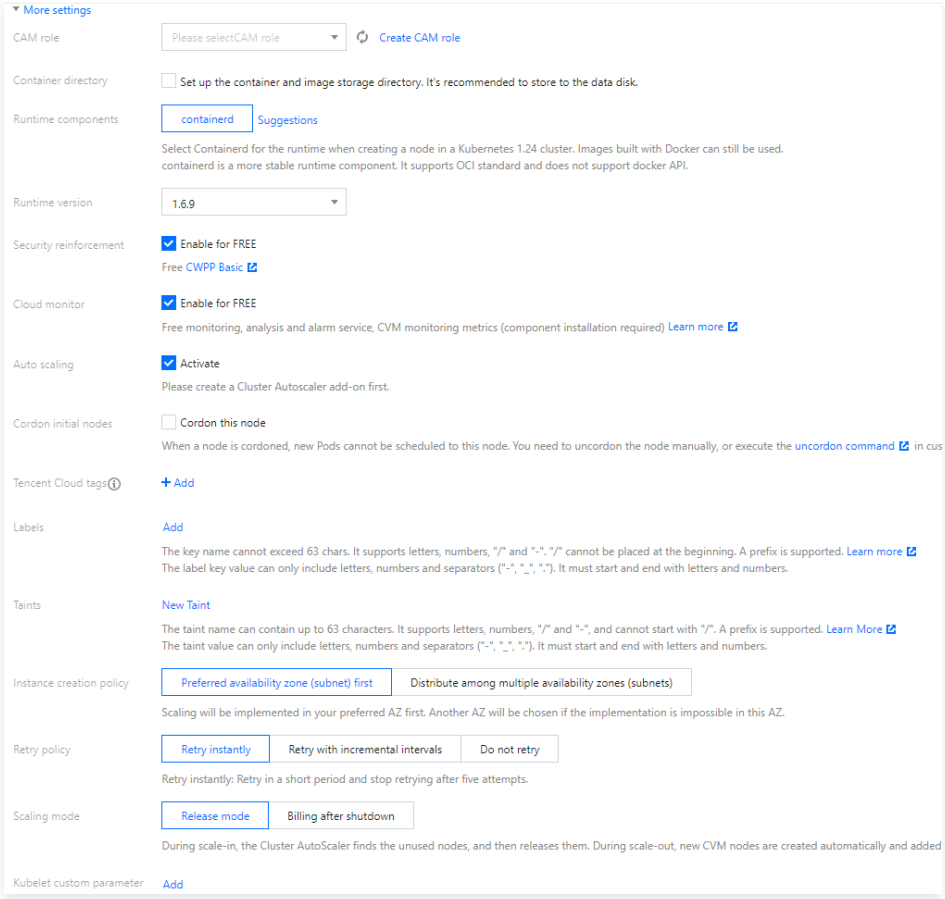
- **Add Data Disk (optional)**: click **Add Data Disk** and specify the settings according to the table above.
  - **Public Bandwidth**: **Assign free public IP** is selected by default, indicating that the system will assign a public IP for free. Billing by traffic or by bandwidth can be selected as needed. For billing details, see Public Network Billing. You can customize the network speed.
- **Login Method**: you can 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. Compared to regular passwords, it is a more secure way to log in to a CVM. For 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**: an automatically generated password will be sent to your Message Center.
  - **Custom Password**: set a password as prompted.
- **Security Groups**: 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.
- **Quantity**: the desired capacity. You can specify this value as required.

> **Note**
> If auto scaling has been enabled for the node pool, this quantity will be automatically adjusted based on the loads of the cluster.

- **Node Quantity Range**: 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 required.

> **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.

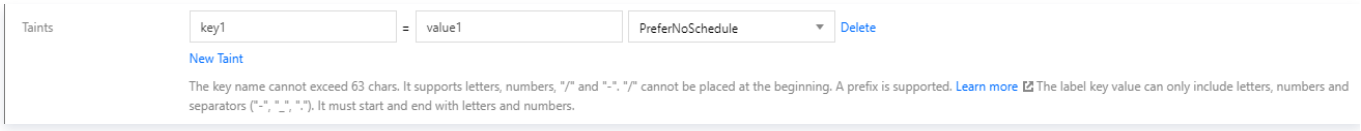
5. (Optional) Click **More Settings** to view or configure more information as shown below:



- **CAM Role**: binds the same CAM role to all nodes in the node pool, granting the authorization policy of this role to the nodes. For more information, please 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**: 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 **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**: taints are node attributes. 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 the relevant conditions cannot be scheduled to these nodes and any such Pods already on these nodes will be drained.

> **Note**
> Taints typically consist of three elements: `key`, `value`, and `effect`. The possible values for `effect` usually include the following three types:
> - **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 cannot 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.

For example, to set Taints `key1=value1:PreferNoSchedule`, configure the console as shown in the figure below:



- **Retry Policy**: select one of the following policies as required.
  - **Try again**: 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 one of the following two scaling modes as required.
  - **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 Actions

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

Deleting a Node Pool

Last updated：2023-09-26 11:10:04

Scenario

This document describes how to delete a created node pool in a cluster using the container service console. You can follow the instructions in this document to delete unused node pools and reduce unnecessary resource waste.

Preparations

- 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.

Instructions

1. Select **More** > **Delete** in the upper right corner of the target node pool card. As shown in the following figure:

2. In the "Delete Node Pool" window that appears, set whether to retain the nodes as needed. See the following figure:

> **Note**
> - By default, pay-as-you-go nodes are selected for termination. You can uncheck this option based on your actual needs.
> - Pay-as-you-go nodes cannot be recovered once terminated, so please operate with caution and back up your data in advance.

3. Click **Confirm** and wait for the deletion to complete.

Related Actions

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

Adding a Node

Last updated: 2023-09-15 17:58:06

Scenario

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

- Creating a node
- Adding an existing node

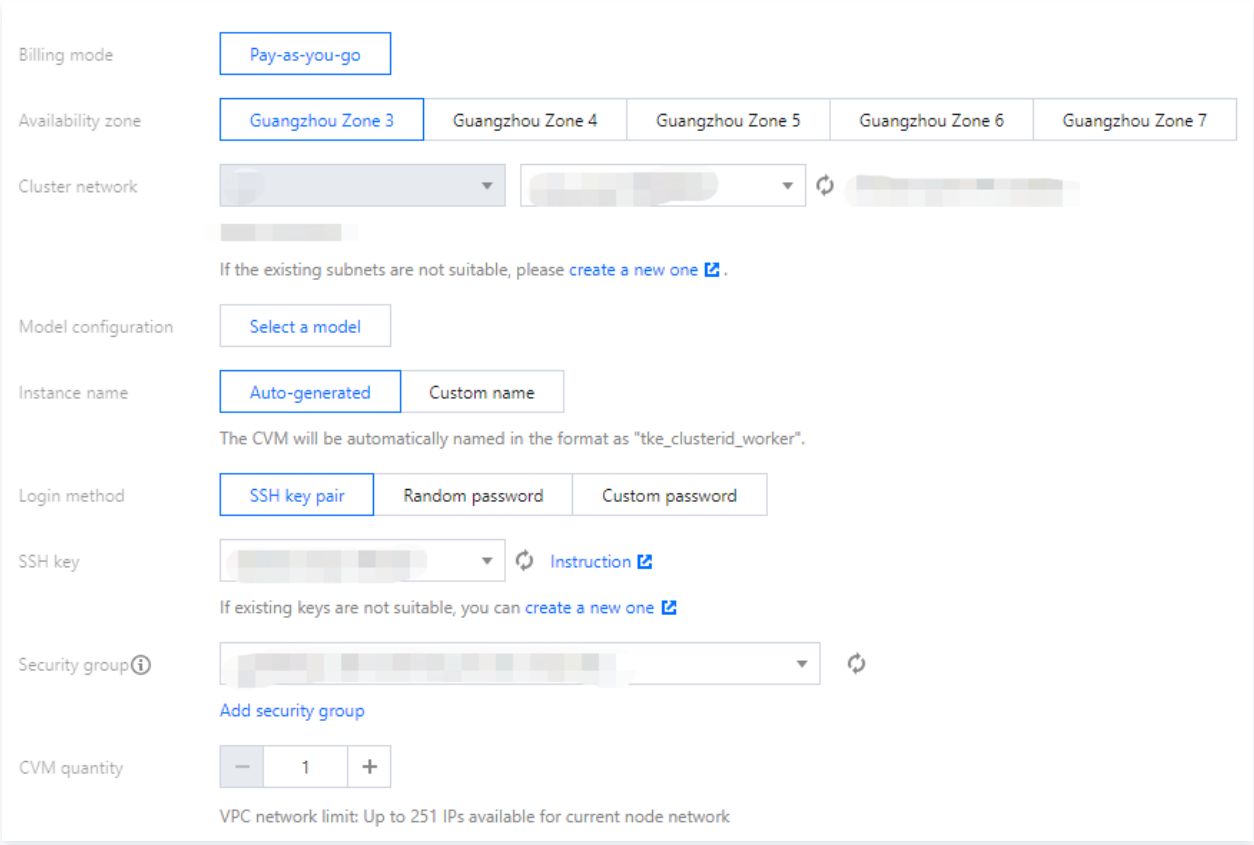
Preparations

You have logged in to the TKE console.

Instructions

Creating a node

1. In the left navigation bar, click Cluster to enter the "Cluster Management" page.
2. Click the ID of the cluster in which the CVM instance is to be created 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 relevant parameters according to your actual needs, as shown in the following image:

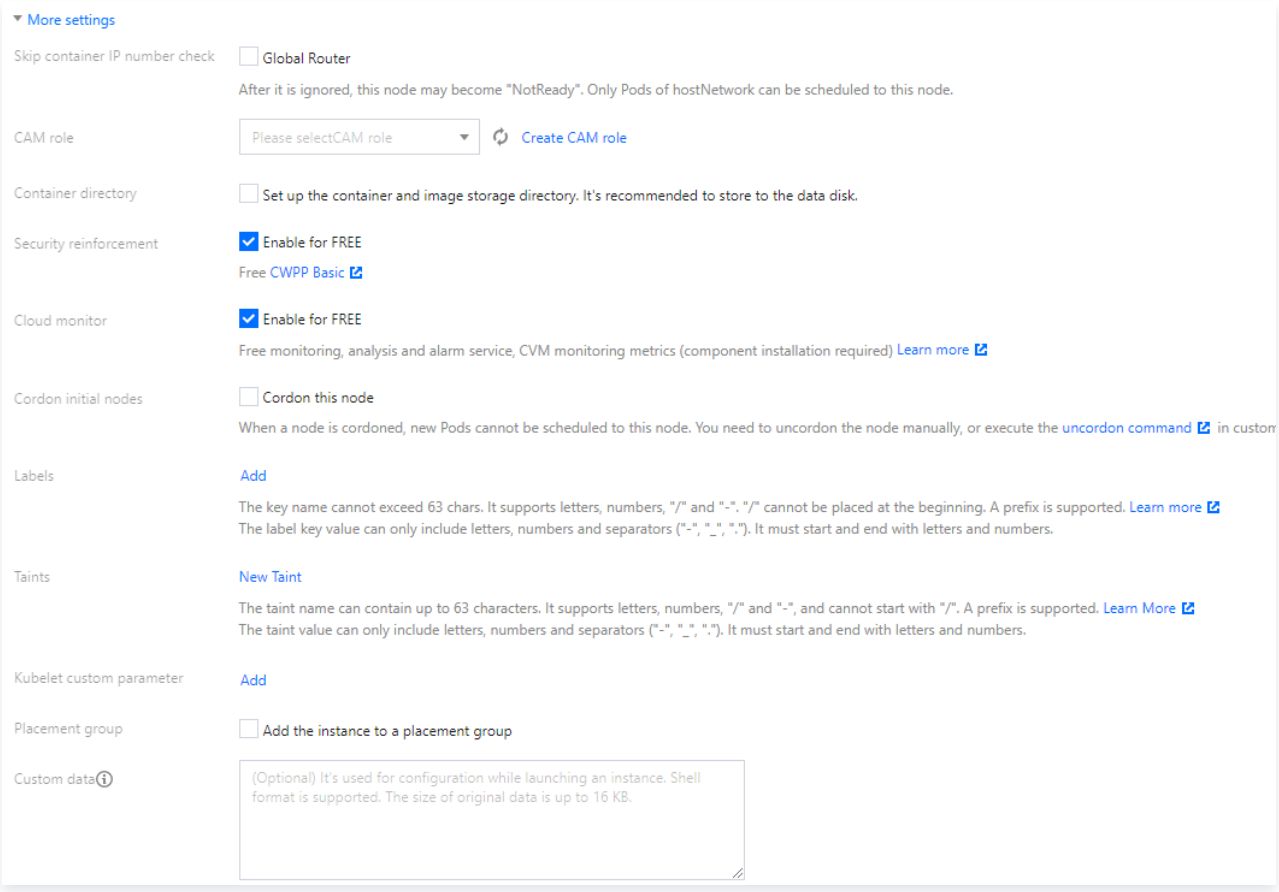


The main parameter information is as follows:

- **Billing mode**: Both **Pay-as-you-go** and **Monthly Subscription** are supported. For details, see Billing Mode.
- **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**: you can select the model by specifying the number of CPU cores, memory size, and instance type. For more information, see Instance Types and Customizing Linux CVM Configurations.
  - **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 needed. 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**: you can 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. Compared to regular passwords, it is a more secure way to log in to a CVM. For 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**: an automatically generated password will be sent 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.
- **CVM quantity**: the number of instances to create. You can specify this value as needed.

5. (Optional) Click **More settings** in the "Create Node" page to view or configure additional information, as shown in the image below:



- **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`.

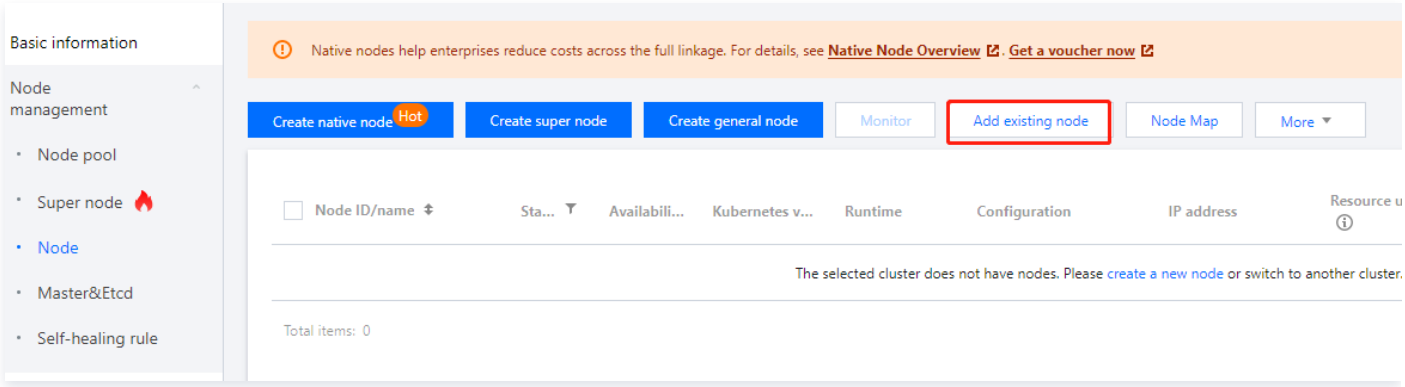
- **Security reinforcement**: Free DDoS, Web Application Firewall (WAF) and Cloud Workload Protection (CWP) are activated by default. For more information, see Cloud Workload Protection.
- **Cloud monitor**: Free monitoring, analysis, and alarms are activated by default, and components are installed to obtain CVM monitoring metrics. For more information, see Tencent Cloud Observability Platform.

  Cordon initial nodes: By enabling "Block", new Pods will not be scheduled on this node. To unblock the node, you need to manually cancel the block or execute the unblock command in custom data. Please set as needed.
- **Labels**: 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**
> - Currently, you can only add CVM instances in the same VPC.
> - 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 navigation bar, click Cluster to enter 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. Setting File System Volume Label for Data Disks
     > - 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 the box: Enter the device name, format the system (you can choose not to format), and set the mount point. If you want to format the /dev/vdb device as ext4 and mount it under the /var/lib/docker directory, you can set it as follows: Device name `/dev/vdb`, format system `ext4`, mount point `/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 **Finish**.

Node Pool FAQ

Last updated: 2023-09-26 11:10:43

This document primarily addresses common issues encountered while using general node pools.

What is the relationship between node pools and scaling groups?

A node pool is a collection of nodes with similar specifications, configurations, and attributes. You can perform batch operations on these nodes in the Tencent Cloud Container Service Console, such as setting node specifications, Labels, Taints, scripts, and other parameters. Within the same cluster, you can create node pools with different billing types (pay-as-you-go, yearly/monthly subscription, and spot instances). The underlying implementation of node pools relies on the cloud product Auto Scaling (AS) and mainly includes the following two concepts:

- **Scaling groups** are collections of cloud server instances that follow the same rules and target the same scenarios. Scaling groups define attributes such as the maximum and minimum number of CVM instances within the group.
- **Launch configurations** are templates for automatically creating cloud server instances, including instance types, system disk and data disk types and capacities, key pairs, security groups, and more.

Each node pool corresponds to a scaling group, and each scaling group corresponds to a launch configuration. You can obtain the links to the bound AS "launch configuration" and "scaling group" in the Tencent Cloud Container Service Console node pool details page.

Which parameters can be modified in a node pool?

Reminder:
Aside from the parameters mentioned in this document that can be modified in the AS console, it is not recommended to adjust other parameters, as doing so may cause the auto-scaling functionality of the node pool to malfunction.

Parameter items	Recommended modification methods
Node pool name	TKE Console: Node pool basic information page is editable.
Elastic scaling capability	
Number of Nodes	
Cloud tags, deletion protection	
Labels、Taints	In the TKE Console, the basic information page of the node pool is editable, allowing you to choose whether to apply the modifications to the existing nodes.
Operating System	TKE Console: The node configuration details page is editable, and changes will only apply to new nodes added under the node pool.
Runtime component	
Alternative models	TKE Console: Node configuration details page can be edited: • A single node pool can have up to 10 different instance models (including the primary model), so please plan accordingly. • The available instance types in the current list are filtered based on the availability zone of the node pool's subnet and the remaining resources in the network. • If your node pool consists of GPU-type instances, the driver installation will be based on the specifications set during node pool creation, and non-GPU-type instances cannot be added as alternative models.
Custom data	In the TKE Console: The node configuration details page allows you to view and edit configurations. Any modifications will only apply to newly added nodes within the node pool. **Note:** The TKE platform needs to inject the node initialization Agent installation script into the "Custom Data." Therefore, it is **not recommended** to directly modify the "Custom Data"

	in the AS launch configuration, as it may affect the normal addition of nodes to the cluster.
Security Group	AS Console: The launch configuration details page can be edited, and the changes will only apply to newly added nodes in the node pool.
Instance Name (Node Name)	AS Console: Launch configuration advanced settings can be edited, and modifications will only apply to newly added nodes in the node pool.
Subnets	AS Console: The scaling group details page is editable, and modifications will only apply to newly added nodes in the node pool.
Instance Creation Strategy/Retry Policy	AS Console: The scaling group policy information page is editable, and modifications will take effect for the next scaling activity.

Parameters that cannot be modified

Parameter items	Impact Description
Billing	Node pools do not support changing the billing mode, and it is also **not recommended** to modify the "instance billing mode" in the AS launch configuration.
Data Disk	Node pools do not support modifying data disks, and it is also **not recommended** to change the capacity, add or remove data disks in the AS launch configuration. Otherwise, it may affect the Cluster Autoscaler component's judgment on the expansion node template, resulting in pending pods being unable to be scheduled on the newly expanded nodes.
Supports Virtual Private Cloud (VPC)	Node pools do not support modifying VPCs, and it is also **not recommended** to modify the "supported network" in the AS scaling group details page, as this may cause node expansion to fail.

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 deletion will fail.
- Instances scaled out by setting **alarm-triggered policies** or **scheduled tasks** for scaling groups through the AS Console cannot be detected by node pools, which may affect the elastic scaling judgment of cluster components like Cluster Autoscaler (CA). Therefore, it is not recommended to modify other parameters of scaling groups in the AS Console.

Native Node Management Overview

Last updated: 2023-09-26 11:17:42

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.

Strengths

Integrated with the FinOps principle, native nodes boost cost optimization of cloud resources

- Equipped with HouseKeeper visualized resource dashboard, native nodes enhance resource utilization, reduce costs, and improve efficiency in the cloud.
- Intelligent Request Recommendation for load balancing, reducing resource idling.
- Offering 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	Unavailable
In-place Pod configuration adjustment	Supported	Unavailable
Custom configuration entries for kernel parameter tuning and other purposes	Supported	Unavailable
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	Supports recommendation and one-click updates	Unavailable
Job occupation	Supported	Unavailable
Manage Node	Kernel parameter configuration, nameserver parameter configuration, host parameter configuration, pre-request scripts, and post-request scripts	Unavailable

Billing

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 support both **pay-as-you-go** and **monthly subscription** billing modes.

Billing Mode	Monthly Subscription	Pay-as-you-go
Payment Method	Prepaid	Postpaid (Fees are frozen upon purchase and settled hourly)
Billing Unit	CNY/month	USD/second
Unit Price	Lower unit price	High
Minimum Usage Duration	At least one month	Billed by the second, settled hourly, available for purchase and release at any time.
Use Cases	Ideal for mature businesses with long-term stable node resource requirements.	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.

Region and Availability Zone

You can use native nodes in the following regions.

China

Regions			Abbreviation
China	All regions	Beijing	ap-beijing
		Nanjing	ap-nanjing
		Shanghai	ap-shanghai
		Guangzhou	ap-guangzhou
		Chengdu	ap-chengdu
		Chongqing	ap-chongqing
		Hong Kong (China)	ap-hongkong
		Taipei	ap-taipei
	Finance cloud regions	Shenzhen Finance	ap-shenzhen-fsi
		Shanghai Finance	ap-shanghai-fsi
		Beijing Finance	ap-beijing-fsi

Other Countries and Regions

Regions			Abbreviation
Asia Pacific	All regions	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		Sao 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

Native Node Parameters

Last updated: 2023-09-26 11:18:40

Category	Supported Items	Description
Model	• Standard: S2, S4, S5, SA2, SA3, and S6 • Computing: C3 and C4 • Memory-optimized: MA3 and M5 • GPU: GN7, GNV4, PNV4, GN10X, and GN10Xp • High I/O: IT5	The models displayed in the console are related to the remaining resources in the selected availability zone. For additional model requirements, you can submit a ticket for assistance. Moreover, native node pools **support configuring multiple alternative models with the same specifications**. You can set this up on the node pool details page.
System Disk	Premium Cloud Disk and SSD	We recommend specifying the system disk space as at least 100 GB.
Data Disk	Premium Cloud Disk and SSD	We recommend a minimum data disk size of 50 GB; the data disk is not bound by default.
Public Network Bandwidth	EIP	By default, public network bandwidth is not enabled for nodes. For more information, refer to Enabling Public Network Access.
Operating System	TencentOS Server	Leveraging Tencent Cloud's virtualization platform and employing kernel optimization techniques, the system provides support for cloud-native resource isolation while conducting comprehensive parameter tuning tailored to container application scenarios.
Node login	SSH login	Node login is enabled by default. You can enable and deliver an SSH key for the node you want to log in to via the console. For more information, refer to Enable SSH Key Login.
GPU driver	450/470/515 Driver	When creating a node pool, you can select the target driver under the model selection.
Runtime	Supports only Containerd, with version 1.6.9 for clusters v1.24 and later, and 1.4.3 for the rest.	The container team optimized the performance of Containerd logs, increasing the CPU utilization of Nginx containers from 50% to 90% (this feature has been merged into the community).
Kubernetes	Kubernetes major version 1.16 or later	Certain Kubernetes versions have specific requirements for minor version numbers, as follows: v1.16.3-tke.28 or later; v1.18.4-tke.26 or later; v1.20.6-tke.21 or later.
Operational Parameters	Supports Kubelet, Kernel, Hosts, and Nameservers parameter configuration.	For more information, see Management Parameter Overview.
Initialization Script	Supports two stages: before and after node initialization	You can provide an initialization script when creating a node pool.
Node Auto-scaling	This feature is supported.	For details, see Native Node Auto Scaling.
Fault self-heal	This feature is supported.	TKE's self-developed intelligent operations product, Cloud Probe, offers multi-dimensional self-healing capabilities, comprehensively enhancing node stability and operational efficiency. For more information, refer to Self-Healing Rules.
In-Place Configuration Adjustment	This feature is supported.	For more information, refer to Enabling In-Place Pod Configuration Adjustment.

Node specification amplification	Supported, offering scalable dedicated scheduler for native nodes	For more information, refer to Dedicated Scheduler Product Description.
Preemptible Job	This feature is supported.	For more information, see Preemptible Job Feature Description.
QoSAgent	This feature is supported.	Offers multi-dimensional fine-tuning capabilities, such as CPU priority, CPU burst, and enhanced memory/network/disk IO QoS, ensuring quality stability while increasing cluster resource utilization. For more information, refer to Fine-tuning Scheduling.

Lifecycle of a Native Node

Last updated: 2023-09-26 11:18:24

Lifecycle Description

Status	Note
Healthy	The node is running normally and connected to the cluster.
Abnormal	The node is running abnormally and not connected to the cluster.
Creating	The node is being created and has not yet connected to the cluster. Upon completing the actions of **purchasing the machine, installing components, and node registration** , the node will connect to the cluster normally.
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.

Creating Native Nodes

Last updated: 2023-09-26 11:18:53

This document describes how to create native nodes in the Tencent Kubernetes Engine (TKE) console or by using the YAML configuration file.

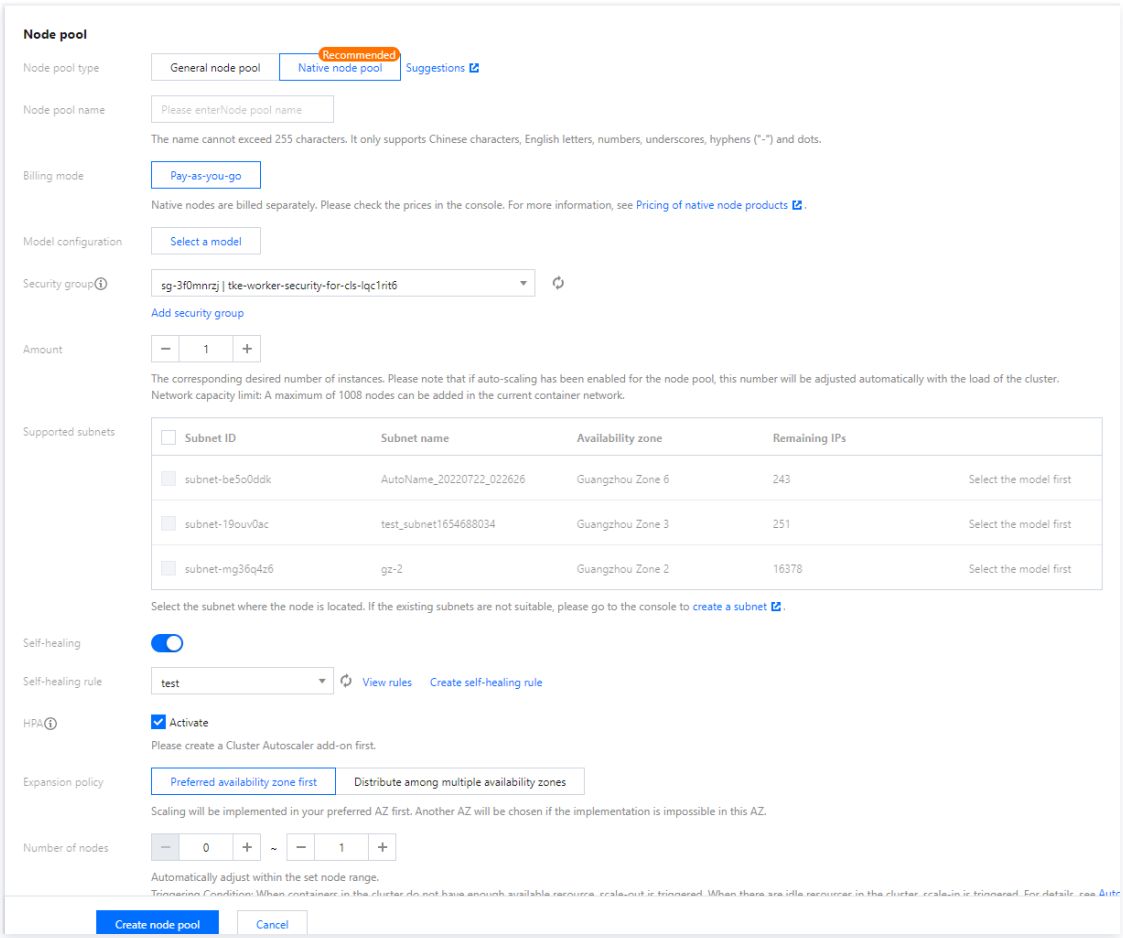
Preparations

- You have logged in to the TKE console.
- You have created a standard TKE cluster. For more information, see Quickly Creating a Standard Cluster.

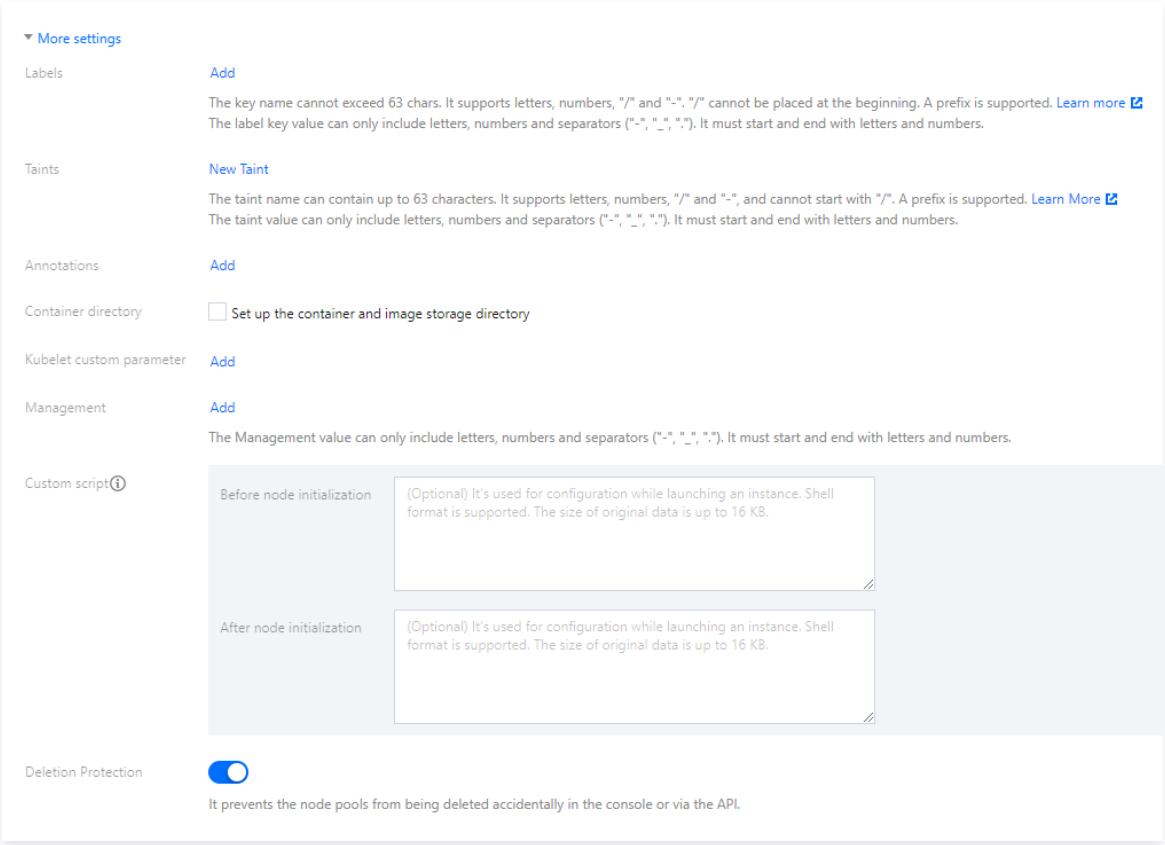
> **Note**
> You can manage native nodes only at the **node pool** level.

Creating a Role via 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. Select **Node Management > Node Pool** from the left menu, and click **Create Node Pool** on the **Node Pool List** page.
4. On the **Create Node Pool** page, refer to the Parameter Description for guidance on setting up, as shown in the following figure:



5. (Optional) Click **More Settings** to view or configure more information, as shown in the following figure:



6. Click **Create node pool**.

Creating via 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:
          - lkongtest
          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	Configuration Item	Parameter in YAML	Note
Launch configuration	Node pool type	Parameter: spec.type Valid value: Native	`Native` indicates a native node pool.
	Node pool name	Parameter: metadata.name Example value: demo-machineset	The node pool name is customizable.
	Billing	Parameter: spec.template.spec.providerSpec.value.instanceChargeType Field value: PostpaidByHour (Pay-as-you-go) / PrepaidCharge (Monthly subscription)	Two billing modes are available: pay-as-you-go and monthly subscription. Please choose according to your actual needs.
	Model Specification	Model: Parameter: spec.instanceTypes Example value: S2.MEDIUM4. For more information about supported specifications, see the **Model Specification** pop-up window in the console. System disk: Parameter: spec.template.spec.providerSpec.value.systemDisk.diskSize/diskType Example values: diskSize: Size of the system disk in GB. You can specify a custom value, which must be a multiple of 10. Default value: 50. diskType: Type of the system disk. Valid values: `CloudPremium` and `CloudSSD`.	You can specify the model specifications in the **Model Specification** pop-up window based on the following information: • Zone: Filters the instances available in the selected zone. For more information, see Regions and Availability Zones. • Model: Filters instances by the number of CPU cores, memory size, and instance type. • System disk: Controls the storage and schedules the node operation. We recommend that you set the system disk size to a value larger than 100 GB.
	Security Group	Parameter: spec.template.spec.providerSpec.value.securityGroupIDs Example value: sg-a7msxxx (ID of the security group)	The default value is the ID of the security group specified when the cluster is created. You can replace the security group or add a security group as needed.

	Amount	Parameter: spec.replicas Example value: 7 (The value is customizable).	You use this parameter to specify the number of nodes to be maintained in the node pool. For example, if you set the value to 5, then five nodes can be maintained in the node pool.
	Container network	Parameter: spec.subnetIDs Example value: subnet-i2ghxxxx (ID of a container subnet)	Select available subnets as needed. 1. When you manually adjust the number of nodes, the system tries to create nodes as per the order of the subnets. For example, if the system creates a node on the first subnet in the list, the system creates all other nodes on the first subnet. 2. If you enable auto-scaling for the node pool, the system selects the appropriate subnets to create nodes based on to the specified scaling policy.
Ops features	Fault self-heal	Parameter: spec.autoRepair Valid values: `true` and `false`.	Optional. We recommend that you enable the fault self-heal feature by setting the value to `true`. This feature detects exceptions of native nodes in real time, such as the OS, runtime, and kubelet exceptions, and takes self-heal actions.
	Self-heal rule	Parameter: spec.healthCheckPolicyName Example value: test-all (name of the bound self-heal rule)	You can bind a different self-heal rule to each node pool.
	Automatic Upgrade (Alpha Version)	-	Optional. This feature is in alpha testing. TKE supports automatic upgrade during the specified Ops window periods to simplify the version Ops, such as the iteration and maintenance of Kubernetes, runtime components, and OS kernel. This feature also timely fix security vulnerabilities based on the Tencent Security intelligence.
	Upgradeable items	-	The system will upgrade the version of the specified items. A specific upgrade package is provided for each upgradeable item. We will release the version upgrade notes in the console and documentation at least seven days before the version upgrade.
	Ops windows	-	The system automatically upgrades the upgradeable items within the specified Ops window periods.
	Maximum number of upgrade nodes	-	Within the Ops windows, the number of upgrade nodes increases exponentially with 2 as the base, starting from 1. The number of nodes that are concurrently upgraded will not exceed the specified maximum number of upgrade nodes.
	HPA	Parameter: spec.scaling	If the auto-scaling feature is enabled, CA automatically scales in or out the node pool.

			Note: The auto-scaling of a native node is implemented by TKE. The auto-scaling of a normal node relies on Auto Scaling (AS).
	Number of Nodes	Parameter: spec.scaling.maxReplicas / minReplicas Example values: maxReplicas: 7 (The value indicates the maximum number of nodes and is customizable). minReplicas: 2 (The value indicates the minimum number of nodes and is customizable).	The number of nodes in the node pool cannot exceed the specified range. If auto-scaling is enabled for a node pool, the number of native nodes in the node pool can be automatically adjusted within the specified range.
	Scaling policy	Parameter: spec.scaling.createPolicy Valid values: **Zone priority** and **Zone equality** in the console. `ZonePriority` and `ZoneEquality` in the YAML configuration file.	1. 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. 2. 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.
Advanced parameters	Labels	Parameter: spec.template.spec.metadata.labels Valid values: Custom labels in the `key1: "val1"` format.	A node-level attribute for the filtering and easy management of nodes. The specified labels are automatically added to all nodes in the node pool.
	Taints	Parameter: spec.template.spec.metadata.taints Valid values: `NoSchedule`, `PreferNoSchedule`, and `NoExecute`. The value indicates the type of taints.	A node-level attribute that is often used with `Tolerations`. You can specify this parameter for a node in the node pool, so as to stop scheduling Pods that do not meet the requirements to the node. The specified type of taints are automatically added to all nodes in the node pool.
	Container directory	Parameter: spec.template.spec.runtimeRootDir Example value: /data/containerd (custom container directory).	Specify this parameter to set the container and image storage directory. Another example value: `/var/lib/`.
	Kubelet custom parameter	This parameter is controlled based on the allowlist. To enable this parameter, submit a ticket or contact Technical Support.	This parameter allows you specify common Kubelet parameters.
	Management	Parameter: spec.template.spec.providerSpec.value.management.hosts/nameservers/KernelArgs Example values: hosts: Hostnames: ['test'], IP: '22.22.22.22' nameservers: ['183.60.83.19', '183.60.82.98'] KernelArgs: This field is controlled by an allowlist. You can submit a ticket for assistance.	Supported Management parameters include Nameservers, Hosts, and KernelArgs.
	Custom scripts	Parameter: spec.template.spec.providerSpec.value.lifecycle.preInit/postInit Example values: preInit: `"echo hello"` (custom script executed before the node initialization).	Configure the node by specifying custom scripts that are executed before and after the node initialization. You need to ensure the reentrant and retry logic of the scripts. The scripts and their log files can be viewed at the path `/usr/local/qcloud/tke/userscript`.

		postInit: `"echo world"` (custom script executed after the node initialization).	

Declarative Operation Practice

Last updated: 2023-09-26 11:19:35

Operations Supported by Kubectl

CRD Type	Action Items
MachineSet	Creating a Native Node Pool `kubectl create -f machineset-demo.yaml`
	Viewing the Native Node Pool List `kubectl get machineset`
	Viewing the YAML Details of a Native Node Pool `kubectl describe ms machineset-name`
	Deleting a Native Node Pool (Monthly and Annual Node Pools must be deleted via the console) `kubectl delete ms machineset-name`
	Expanding the Native Node Pool `kubectl scale --replicas=3 machineset/machineset-name`
Machine	View Native Nodes `kubectl get  machine`
	Viewing the Native Node YAML Details `kubectl describe  ma machine-name`
	Directions `kubectl delete ma machine-name`
HealthCheckPolicy	Creating Fault Detection and Self-Healing Rules `kubectl create -f demo-HealthCheckPolicy.yaml`
	Viewing the Fault Self-Heal Rule List `kubectl kubectl get HealthCheckPolicy`
	Viewing the Fault Self-Heal Rule YAML Details `kubectl describe HealthCheckPolicy HealthCheckPolicy-name`
	Delete self-healing rule `kubectl delete HealthCheckPolicy HealthCheckPolicy-name`

Using CRD via YAML

MachineSet

```
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:
            - lkongtest
            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
```

Kubectl Operation Demo

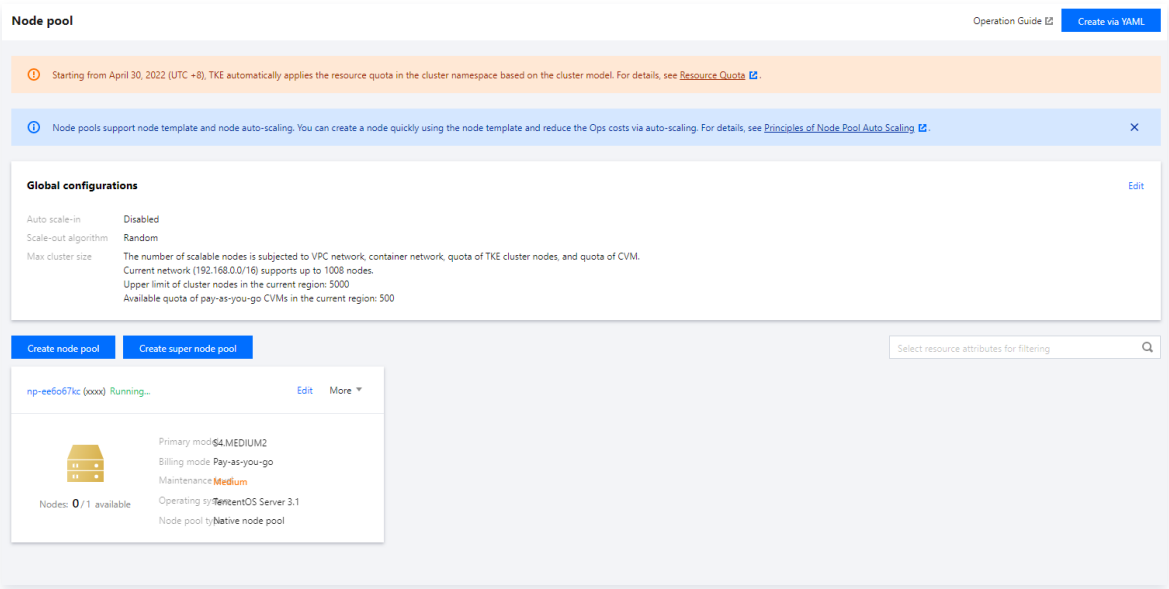
MachineSet

1. Execute the command `kubectl create -f machineset-demo.yaml` to create a MachineSet based on the provided YAML:

```
[root@kather /kube/yaml]# ls
kather-datadisk-test.yaml  kather-resize.yaml  machineset-demo.yaml  pod-resize-demo.yaml
[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: "CVM", "Native"
* spec.healthCheckPolicyName: Required value: healthCheckPolicyName is required when `autoRepair` is true
[root@kather /kube/yaml]# kubectl create -f machineset-demo.yaml
machineset.node.tke.cloud.tencent.com/np-pjrlok3w created
[root@kather /kube/yaml]#
```

2. Check the status of MachineSet np-pjrlok3w using the command `kubectl get machineset`. At this point, the corresponding node pool appears in the console, and the nodes are 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]#
```



3. View the description of MachineSet np-pjrlok3w using the command `kubectl describe 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: 3
              cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size: 0
              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. Execute node pool scaling using the command `kubectl scale --replicas=2 machineset/np-pjrlok3w`:

```
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]#
```

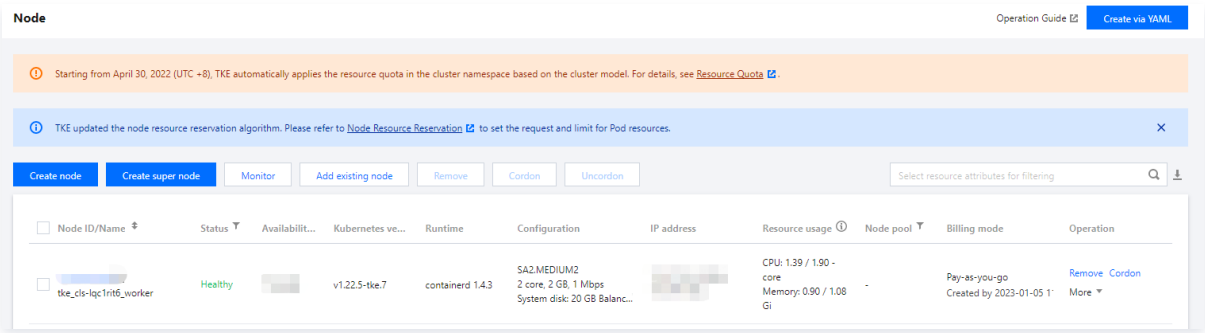
5. Delete the node pool using the command `kubectl delete ms np-pjrlok3w`.

```
[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. Use the command `kubectl get machine` to view the Machine list. At this point, the corresponding nodes already exist 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]#
```



2. View the description of Machine np-14024r66-nv8bk using the command `kubectl describe ma 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 a node without adjusting the desired node count in the node pool, the node pool will detect that the actual number of nodes is insufficient to meet the declared node count, and will create a new node to be added to the pool. Therefore, we recommend following these steps for node deletion:

- Run the `kubectl scale --replicas=1 machineset/np-xxxxx` command to adjust the expected number of nodes.
- Run the `kubectl delete machine np-xxxxxx-dtjhd` command to delete the corresponding node.

- For monthly-subscribed nodes, direct deletion via kubectl is not possible due to the involvement of node termination and refunds.

Enabling Public Network Access for a Native Node

Last updated: 2023-09-26 11:20:37

> **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.

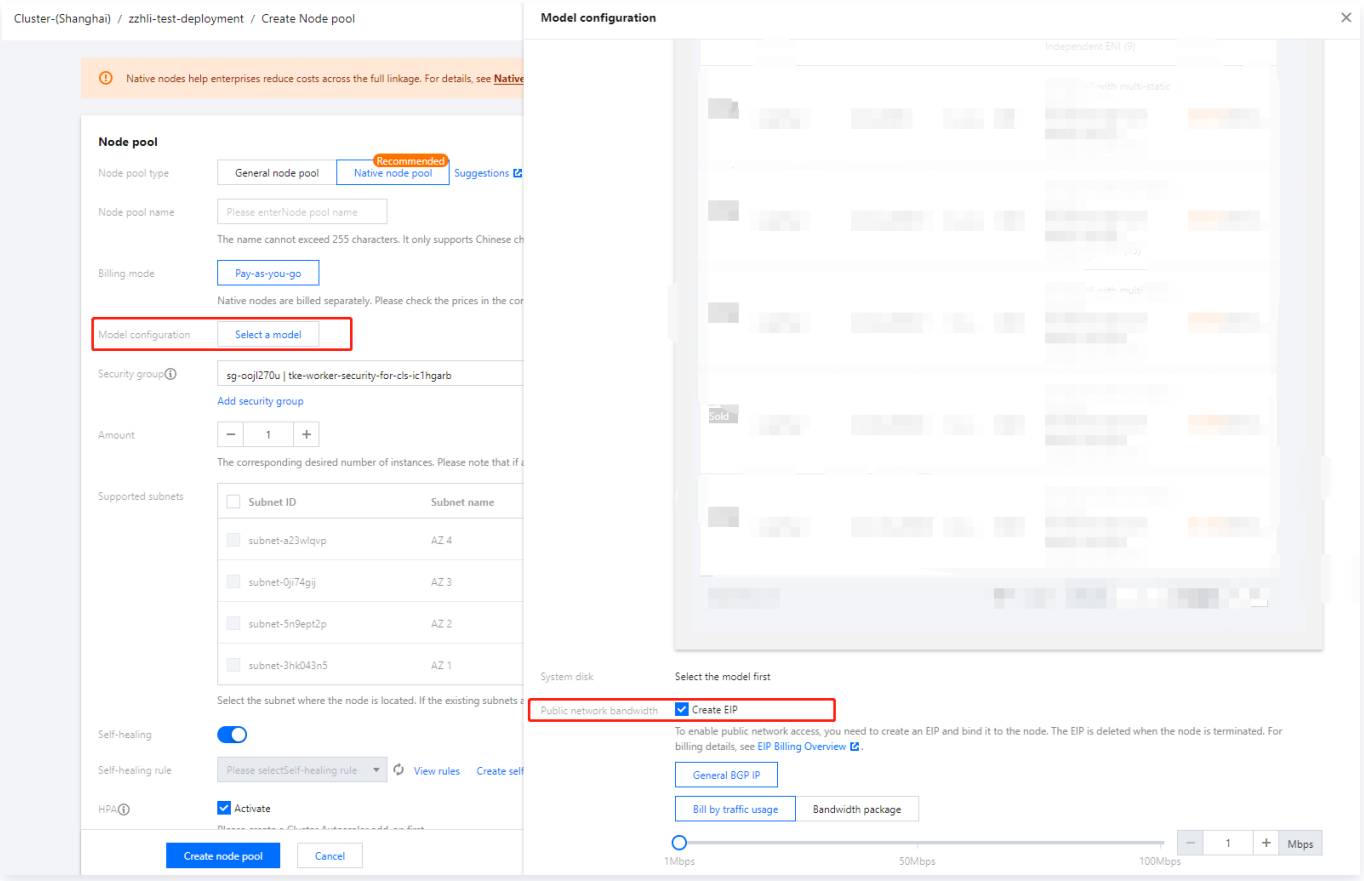
This document describes how to bind a node to an elastic IP (EIP) and enable public network access for the node in the TKE console or using YAML.

Supports and Limits

- 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.
- Native nodes do not incur additional charges for EIPs. For EIP billing details, please refer to Elastic IP Billing Description.

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. In the left sidebar, select **Node Management > Node Pools**. On the **Node Pools** page, click **Create Native Node Pool**.
4. On the **Create Node Pool** page, click **Instance Configuration**. In the **Instance Configuration page**, select the **Create Elastic IP** checkbox, as shown in the following image:



5. Click **Create node pool**.

Using YAML to Enable Public Network Access for Native Nodes

Fields

Field	Field Value	Description
spec.template.spec.providerSpec.value.internetAccessible	addressType	• `EIP`: If this field is left empty, a standard EIP is used. • `HighQualityEIP`: A BGP IP (dedicated EIP) is used.
	chargeType	Billing mode: `TrafficPostpaidByHour`: Postpaid by traffic on an hourly basis. `BandwidthPostpaidByHour`: Postpaid by bandwidth on an hourly basis. `BandwidthPackage`: Paid by shared bandwidth package. The EIP must be in the allowlist of the bandwidth package.
	maxBandwidthOut	Maximum bandwidth in Mbps.
	bandwidthPackageID	ID of the shared bandwidth package.

Note

- For more information on EIP types and considerations, please refer to Applying for an EIP.
- Premium EIPs are currently only supported for **Bill-by-IP accounts**, available in the **Hong Kong, China** region, and billed using **Shared Bandwidth Packages**. If you do not have a premium BGP bandwidth package, you can create one in the Virtual Private Cloud Console > **Shared Bandwidth Packages**.

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
```

Management Parameters

Last updated: 2023-09-26 11:20:50

Overview

Management parameters provide a unified entry for frequently used custom configurations of nodes. You can use this entry to tune the underlying KernelArgs kernel parameters for native nodes. You can also set the Nameservers and Hosts parameters to meet the requirements of the service deployment environment.

Management Parameter Classification

Configuration Item	Description
Nameservers	DNS server addresses required by service deployment.
Hosts	Hosts required by service deployment.
KernelArgs	Kernel parameters for performance tuning. This feature is available only for accounts in the allowlist. You can submit a ticket to apply for the feature.

Note
To ensure normal installation of system components, native nodes are automatically injected with Tencent Cloud's database addresses: `nameserver = 183.60.83.19` and `nameserver = 183.60.82.98`.

KernelArgs parameters

Supported OS parameters and their valid 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 socket and network settings for tuning.

No.	Category	Default value	Valid Values/Range	Parameter Type	Description
1	"net.core.somaxconn"	32768	4096 – 3240000	int	The maximum length of the listening queue for each port in the system.
2	"net.ipv4.tcp_max_syn_backlog"	8096	1000 – 3240000	int	The maximum length of tcp SYN queue length.
3	"net.core.rps_sock_flow_entries"	8192	1024 – 536870912	int	The maximum size of hash table for RPS.
4	"net.core.rmem_max"	16777216	212992 – 134217728	int	The maximum size, in bytes, of the receive socket buffer.
5	"net.core.wmem_max"	16777216	212992 –	int	The maximum size, in bytes, of the send socket buffer.

			13421 7728		
6	"net.ipv4.tcp_rmem"	"4096 12582912 16777216"	1024 – 21474 83647	string	The min/default/max size of tcp socket receive buffer.
7	"net.ipv4.tcp_wmem"	"4096 12582912 16777216"	1024 – 21474 83647	string	The min/default/max size of tcp socket send buffer.
8	"net.ipv4.neigh.default.gc_thresh1"	2048	128 – 80000	int	The minimum number of entries that can be retained. If the number of entries is less than this value, the entries will not be recycled.
9	"net.ipv4.neigh.default.gc_thresh2"	4096	512 – 90000	int	When the number of entries exceeds this value, the GC will clear the entries longer than 5 seconds.
10	"net.ipv4.neigh.default.gc_thresh3"	8192	1024 – 100000	int	Maximum allowable number of non-permanent entries.
11	"net.ipv4.tcp_max_orphans"	32768	4096 – 21474 83647	int	Maximal number of TCP sockets not attached to any user file handle, held by system. Increase this parameter properly to avoid the 'Out of socket memory' error when the load is high.
12	"net.ipv4.tcp_max_tw_buckets"	32768	4096 – 21474 83647	int	Maximal number of timewait sockets held by system simultaneously. Increase this parameter properly to avoid "TCP: time wait bucket table overflow" error.

File descriptor limits:

Large amounts of traffic 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.	Category	Default value	Valid Values/Range	Parameter Type	Description
1	"fs.file-max"	3237 991	8192 – 1200050 0	int	Limit on the total number of fd, including socket, in the entire system.
2	"fs.inotify.max_user_instances"	8192	1024 – 2147483 647	int	Limit on the total number of inotify instances.
3	"fs.inotify.max_user_watches"	5242 88	781250 – 2097152	int	The total number of inotify watches is limited. Increase this parameter to avoid "Too many open files" errors.

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.	Category	Default value	Valid Values/Range	Parameter Type	Description
1	"vm.max_map_	262144	65530 –	int	The maximum number of memory map areas a

	count"		262144		process may have.

Worker thread limits:

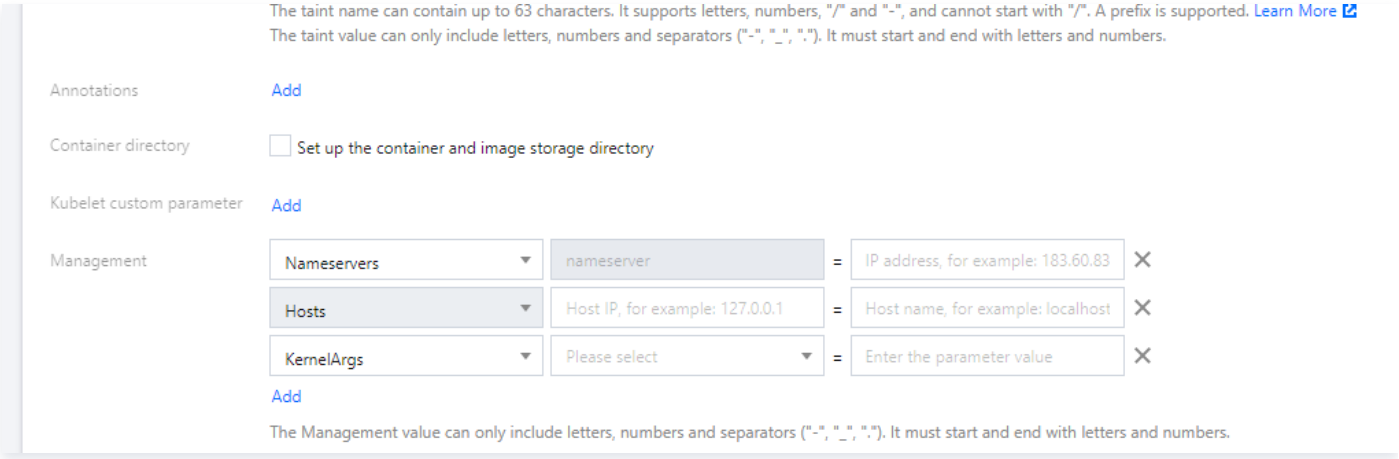
No.	Category	Default value	Valid Values/Range	Parameter Type	Description
1	"kernel.threads-max"	4194304	4096 – 4194304	int	The system-wide limit on the number of threads (tasks) that can be created on the system.
2	"kernel.pid_max"	4194304	4096 – 4194304	int	PIDs greater than this value are not allocated; thus, the value in this file also acts as a system-wide limit on the total number of processes and threads.

Setting the Management Parameters for a Node

Via the Console

Method 1: Setting the Management parameters for a new node

1. Log in to the TKE console and create a native node. For more information, see Creating Native Nodes.
2. On the "Create Node Pool" page, click **More Settings** to set Management parameters for the nodes, as shown in the following image:



3. Click **Create node pool**.

Method 2: Setting the Management parameters for an existing node

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 > Node Pool** in the left sidebar to navigate to the "Node Pool List" page.
4. Click the ID of the node pool to go to the **Node list** page.
5. In the "Details" tab, click **Parameter Settings > Edit** to modify the Management parameters.

Enabling the feature by using YAML

```
apiVersion: node.tke.cloud.tencent.com/v1beta1
kind: MachineSet
spec:
  type: Native
  displayName: mstest
  replicas: 2
  autoRepair: true
  deletePolicy: Random
```

```
instanceTypes:
- C3.LARGE8
subnetIDs:
- subnet-xxxxxxxx
template:
  spec:
    displayName: mtest
    providerSpec:
      type: Native
      value:
        instanceChargeType: PostpaidByHour
        Enter the management parameter information here
        management:
          hosts:
          - Hostnames:
            - static.fake.com
            IP: 192.168.2.42
          - Hostnames:
            - common.fake.com
            IP: 192.168.2.45
          nameservers:
          - 183.60.83.19
          - 183.60.82.98
          - 8.8.8.8
          kernelArgs:
          - kernel.pid_max=65535
          - fs.file-max=400000
          - net.ipv4.tcp_rmem="4096 12582912 16777216"
          - vm.max_map_count="65535"
```

Super Node Management
Super Node Overview

Last updated: 2023-09-26 11:21:13

Super Node Overview

Super nodes are Tencent Cloud's newly upgraded node product form, providing users with availability zone-level, customizable specification node capabilities. Using a super node is similar to using an extra-large CVM, making resource management and scaling simpler. Super nodes support both **annual/monthly subscription** and **pay-as-you-go** billing modes.

- In the annual/monthly subscription mode, users can customize node specifications and purchase the total customized node computing power on a prepaid basis. The annual/monthly subscription mode restricts user-scheduled Pods to have a scale between 1C and 8C, with a CPU-to-memory ratio less than 1:4. This mode is suitable for fixed computing power and always-on businesses. Compared to regular nodes, the per-core price of super nodes in the annual/monthly subscription mode is lower, allowing users to migrate all eligible Pods to super nodes to reduce the per-core cost of fixed resources.
- In pay-as-you-go mode, you don't need to specify node specifications, and node resources are elastic and billed by CPU and memory usage. This mode applies to elastic computing power scenarios, where pay-as-you-go super nodes can be added to support resource elasticity during peak hours, so as to further reduce cluster resource costs.

Pods deployed on super nodes have the same security isolation as cloud servers and the same network isolation as Pods deployed on existing regular nodes in the cluster. However, since super nodes do not have separate public IP addresses, access to the internet requires using NAT or binding EIP mode.

Strengths

Super nodes have the following advantages over general nodes:

Reduced costs

Purchasing an annual/monthly subscription super node offers a lower per-core price compared to purchasing a regular node.

- CPU and memory pricing is lower. For details, please refer to Super Node Pricing.
- The annual/monthly subscription super nodes have a larger specification limit, allowing for a wider range of schedulable Pods and avoiding the waste of boundary resources when purchasing regular nodes with an annual/monthly subscription.

Pay-as-you-go super nodes feature elasticity within seconds and on-demand use, making them quite competitive for cost reduction in elastic business scenarios.

- Pay-as-you-go super nodes offer true on-demand usage, reducing resource fragmentation and improving overall cluster resource utilization. By minimizing resource buffer, costs are lowered.
- The billable usage of elastic resources is shortened to save costs. As pay-as-you-go super nodes are scaled out within seconds and scaled in instantaneously, scaling costs are greatly reduced.

Easy Management

Managing resources becomes easier as you only need to manage one node per availability zone instead of multiple nodes. Annual/monthly subscription super nodes support on-demand upgrades and partial downgrades, making it as simple as managing an oversized CVM node for scaling up or down. Elastic resources can be achieved by adding pay-as-you-go super nodes, providing faster and more efficient elasticity compared to regular node pools and auto-scaling groups. This not only better meets elastic demands but also saves costs for users.

Node management comparison:

Scenario	General Node	Super Node
Purchase	Node types, specifications, and quantities need to be planned and purchased multiple times.	Only the total resource specifications in an AZ need to be collected, and only one super node with custom specifications needs to be purchased.
Resource scale-out	Nodes need to be added or node specifications	• Long-term expansion: Upgrade the super node configuration and adjust the total specification size.

	need to be batch adjusted.	• Short-term scale-out: Use pay-as-you-go super nodes, with the elastic portion scheduled to pay-as-you-go super nodes by default.
Resource scale-in	Nodes need to be returned, and losses need to be borne.	• Long-term scaling down: Downgrade the super node configuration by adjusting the total specification size (downgrade is supported once per month), and up to 3 super nodes can be downscaled. • Short-term scaling down: No action is required when using pay-as-you-go super nodes.
Manage Node	Multiple nodes need to be managed.	Only one node needs to be managed in an AZ.

More efficient elasticity

Compared with node pools and scaling groups, pay-as-you-go super nodes simplify server purchase, initialization, and returning during scaling processes. This greatly accelerates elasticity and minimizes possible scaling failures.

- Pay-as-you-go super nodes shorten the general scaling process of 4-6 minutes to seconds, greatly increasing the efficiency.
- Pay-as-you-go super nodes avoid the CA, cordoning, and Pod draining processes, truly implementing lossless and instantaneous scale-in.

Billing

Super nodes support both pay-as-you-go and annual/monthly subscription billing modes.

Instance Billing Method	Monthly Subscription	Pay-as-You-Go
Payment Method	Prepaid	Deposit upon purchase, billed hourly
Billing Unit	CNY/month	USD/second
Unit Price	Lower unit price	High
Minimum Usage Duration	Minimum usage of one month	Billed by the second, settled hourly, available for purchase and release at any time.
Configuration adjustment	Supports Configuration Adjustment	There are no specification limits
Use Cases	Suitable for mature online businesses with long-term stable computing power requirements.	It is suitable for scenarios where the demand for computing power fluctuates greatly

Note:

- When adding an annual/monthly subscription super node, the total price will be calculated based on the total node specifications, CPU and memory unit price, and the purchase duration for prepaid payment.
- Adding pay-as-you-go super nodes is free of charge, and fees will be charged after Pods are scheduled to super nodes. The backend will calculate the fees based on the CPU, GPU, and memory resources requested by workloads and the execution duration of workloads. You don't need to pay any fees in advance.

Region and Availability Zone

Users can utilize annual/monthly subscription super nodes in the following availability zones:

China

Regions	Availability Zones
South China (Guangzhou) ap-guangzhou	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 2 ap-shanghai-2
	Shanghai Zone 3 ap-shanghai-3
	Shanghai Zone 4 ap-shanghai-4
	Shanghai Zone 5 ap-shanghai-5
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 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

Kubernetes Version

- Pay-as-you-go super nodes are supported by clusters on v1.16 or later.
- Annual/monthly subscription super nodes currently support clusters with version 1.18.4 and later.

Pods Schedulable to Super Nodes

Clusters with super nodes support scheduling Pods with different specifications based on different billing modes.

Annual/Monthly Subscription Super Node

- Supports scheduling 1C to 8C standard specification Pods (if non-standard, they will be automatically converted to standard specifications). For specifications, please refer to Super Node Schedulable Pod Description.
- Supports scheduling Pods with a CPU-to-memory ratio less than 1:4.

Pay-as-you-go super node

- 0.25C-16C Pods are supported.
- Pods with a CPU to memory ratio of up to 1:8 are supported.
- GPU Pods are supported.

For more information, see Notes on Pod Scheduled to Supernodes.

Scheduling to Super Nodes

Annual/monthly subscribed super nodes and annual/monthly subscribed TKE regular nodes have equal scheduling priority.
In TKE clusters with pay-as-you-go super nodes added, when annual/monthly subscribed node resources are insufficient, Pods will be automatically scheduled to pay-as-you-go super nodes. If node resources are sufficient, Pods on pay-as-you-go super nodes will be prioritized for scaling down. Additionally, manual scheduling of Pods to super nodes is also supported.
For more information, see How to Schedule a Pod to a Supernodes.

Scenarios

Super nodes are suitable for all business scenarios. The annual/monthly subscription mode is applicable to all always-on businesses with Pod specifications ranging from 1C to 8C, while the pay-as-you-go mode is suitable for elastic scenarios.

Using Super Nodes for Always-On Businesses

Advantages: Cost-effective, easy to manage
Compared to regular nodes with annual/monthly subscriptions, super nodes offer approximately 20% lower per-core prices. For always-on services with fixed computing power requirements, if most of the current Pod specifications are below 8C, you can use the annual/monthly subscription mode for super nodes. Purchase super nodes based on the total Pod specifications as needed, schedule all 1C~8C Pods to super nodes, and reduce the per-core price of cluster resources to achieve cost reduction goals.

Pay-as-you-go super nodes for elastic businesses

Strengths: Low costs and high elasticity
In elastic business scenarios, pay-as-you-go super nodes allow for scale-out within seconds to accommodate traffic surges and require less resource buffer to reduce costs.

- 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: 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.

Pod Schedulable to Super Node

Last updated: 2023-09-26 11:22:04

Billing Mode

Pods scheduled on super nodes support two billing modes: prepaid and postpaid (pay-as-you-go and spot pricing).

Kubernetes Versions

- Pay-as-you-go super nodes are supported by clusters on v1.16 or later.
- Annual and monthly subscription super nodes currently support the highest minor versions of v1.18, v1.20, and v1.24 clusters.

Specifications of Pods Schedulable to Super Nodes

The resource specifications of Pods on super nodes are the basis for available resources during container runtime and service billing. It is essential to understand the resource specifications of Pods on super nodes. Different billing modes for super nodes support different schedulable Pod specifications.

Monthly Subscription Mode

- Supports scheduling Pods with standard specifications ranging from 1C to 8C.
- Supports scheduling Pods with CPU values greater than 1/4 of the memory values.

List of specifications supported by nodes:

> **Note**
> For nonstandard Pods, their specifications are automatically upgraded.

CPU/Core	Memory Range (GiB)	Granularity of Memory Range (GiB)
1	1 – 4	1
2	2 – 8	1
4	8 – 16	1
8	16 – 32	1

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/Core	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
4	8 – 32	1
8	16 – 32	1
12	24 – 48	1

16	32 – 64	1
32	64、128、256	–
64	128、192、256、512	–

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.
- You are recommended to mount important data and large files to Volume for persistent storage.

Pod network

Pods scheduled on super nodes use a VPC network parallel to cloud servers, cloud databases, and other cloud products. Each Pod occupies a VPC subnet IP. Communication between Pods, and between Pods and other cloud products within the same VPC, occurs directly through the VPC network without performance loss.

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 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 by 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	For the default security group bound to the workload, please enter the Security Group ID(s): Multiple IDs can be entered, separated by `commas`. For example, `sg-id1,sg-id2`. Network policies take effect in the order of the 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/m	Amount of memory required by a Pod. For more	No. If you specify it, make sure that

em	information, see Resource Specifications. You need to specify the unit, for example, 512 MiB, 0.5 GiB, or 1 GiB.	the specification is supported, and you need to enter the `cpu` and `mem` parameters.
eks.tke.cloud.tencent.com/cpu-type	The required CPU resource model for Pods is currently available in the following models: Intel and AMD specific models, such as S4 and S3. For detailed configurations supported by each model, please refer to 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	The GPU resource models required for Pods currently include the following: V100, 1/4 T4, 1/2 T4, and T4. The priority order can be written as "T4, V100", which means that a T4 resource Pod will be created first. If the available T4 resources in the selected region are insufficient, a V100 resource Pod will be created instead. For specific configurations supported by each model, please refer to the 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. Make sure that the entered specification is supported.
eks.tke.cloud.tencent.com/retain-ip	Pod Static IP: Set the value to `"true"` to enable this feature. For Pods with this feature enabled, the Pod's IP will be retained for 24 hours after the Pod is terminated. If the Pod is recreated within 24 hours, it can still use the same IP. After 24 hours, the IP may be taken by other Pods. **This is only effective for StatefulSets and raw Pods.**	Not required
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.**	Not required
eks.tke.cloud.tencent.com/eip-attributes	This indicates that the Pods of the workload need to be associated with an EIP. When the value is `""`, it means that the default EIP configuration is used. You can fill in the EIP cloud API parameter JSON within `""` to implement custom configurations. For example, if the annotation value is '{"InternetMaxBandwidthOut":2}', it means using a 2M bandwidth. Note that this is not available for non-bandwidth upgraded accounts.	Not required
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.	Not required
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,	Not required

	Pods that cannot be allocated with EIPs will be in the "Pending" status. Note: this cannot be used for non-bill-by-IP accounts.	

For samples, please see Annotation.

Default Quota

When purchasing an annual or monthly subscription super node, a default quota will be allocated based on the total specifications. For pay-as-you-go super nodes, by default, each cluster can schedule up to 500 Pods on the super node. If you require resources exceeding the above quota, you can submit a quota increase application. Tencent Cloud will assess your actual needs, and upon approval, your quota will be increased.

Applying for a higher quota

1. Please submit a ticket, select **Manual Support** or **Other Issues** > **Create Now** to enter the ticket information submission 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. Click **Online consulting**.

Pod Limits

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

1. Supported volume types include EmptyDir, PVC, Secret, NFS, ConfigMap, Downward API, HostPath, GitRepo, ISCSI, DownwardAPI, Projected, CSI, and Ephemeral.
2. For PVC volumes:
   - PV Types: Supports NFS, CephFS, HostPath, and static CBS types; other types are not supported.
   - StorageClass types: Custom, /cloud.tencent.com/qcloud-cbs, com.tencent.cloud.csi.cbs, and com.tencent.cloud.csi.cfs are supported, while others are not.

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 limitations

- The super node feature is not available for clusters without any server nodes.
- The Pods that have specified the hostIP will use the Pod IP as the value of 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: 2023-09-26 11:22:20

This article primarily discusses how to automatically or manually schedule Pods to supernodes within a Tencent Kubernetes Engine (TKE) cluster.

Automatic scheduling

- If a cluster is configured with yearly or monthly subscribed super nodes, Pods that meet the super node scheduling rules will be scheduled equally between the subscribed super nodes and regular nodes.
- If a cluster is configured with pay-as-you-go super nodes, Pods will be automatically scheduled to super nodes during peak business hours when the existing yearly or monthly subscribed node resources are insufficient, without the need to purchase servers. When the business returns to normal, the Pod resources in the super nodes will be automatically released, eliminating the need for server return operations.
- If both the cluster scaling and pay-as-you-go super node features are enabled for a cluster, Pods will be scheduled to pay-as-you-go super nodes first, and cluster scale-out won't be triggered. If Pods cannot be scheduled to super nodes due to scheduling limits, the scale-out will be triggered normally. When the server node resources are sufficient, the cluster will release Pods on super nodes first.

Manually schedule

Users can manually schedule Pods to super nodes. By default, pay-as-you-go super nodes automatically add Taints to lower their scheduling priority. To manually schedule Pods to super nodes or specify super node scheduling, you usually need to add corresponding Tolerations to the Pods. However, not all Pods can be scheduled to super nodes. For more information, please refer to Super Node Scheduling Instructions. For ease of use, you can specify a nodeselector in the Pod Spec. An example is provided below:

```
spec:
  nodeSelector:
    node.kubernetes.io/instance-type: eklet
```

TKE's control components will determine whether the Pod can be scheduled to a super node. If not, the Pod won't be scheduled to the super node.
Yearly or monthly subscribed super nodes currently only support scheduling Pods with specified specifications and CPU-to-memory ratios. If the Pods do not meet these rules, the scheduling will not be successful.

Super Node Annotation Description

Last updated: 2023-09-26 11:22:30

By defining Annotations in a YAML file, you can achieve a wide range of customization capabilities for super nodes. You can learn more about common configuration operations for super nodes using Annotations from the Annotation Description.

Collecting Logs of the Pod on the Supernodes

Last updated: 2023-09-26 11:23:47

This document describes how to collect logs from a Pod scheduled to a super node in a TKE cluster.

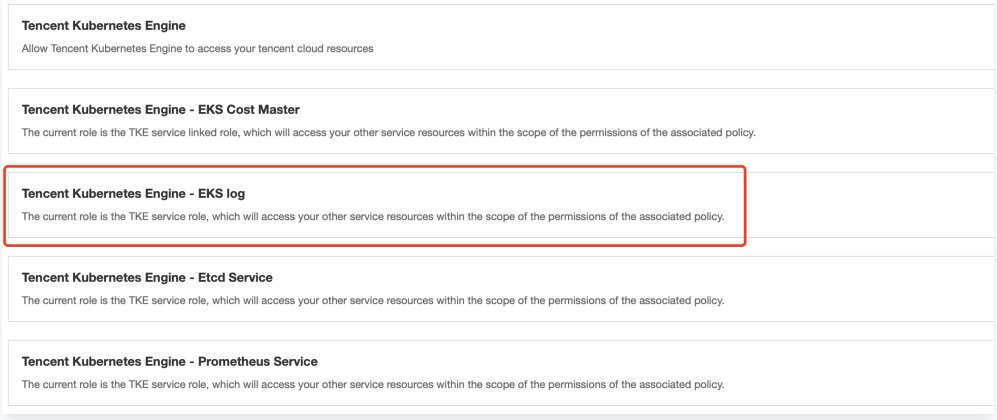
- Collect logs to CLS
- Collect logs to Kafka

Collect 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 the **Access Management Console** > Roles.
2. Click **Create Role** on the "Role" page.
3. In "Select Role Entity", choose **Tencent Cloud Services** > **Container Service (TKE)** > **Container Service – EKS Log Collection**, and click **Next**. As shown in the image below:



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.

Collect logs to Kafka

To collect logs from Pods on super nodes to self-built Kafka or CKafka, you can configure the corresponding log collection rules in the console or configure the CRD yourself, defining the collection source and consumer. Once the CRD configuration is complete, the built-in collector in the Pod will collect logs according to the rules.
The specific CRD configuration is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig                    ## Default value
metadata:
  name: test                       ## CRD resource name, unique within the cluster
spec:
  kafkaDetail:
    brokers: xxxxxx      ## (Required) The broker address. Generally, it is domain name:port. If there are more than one address, separate them with ",".
    topic: xxxxxx        # Topic ID, which is required
    messageKey:          # Optional, specify the Pod field as the key to upload to the designated partition
      valueFrom:
```

```
      fieldRef:
        fieldPath: metadata.name
    timestampKey:          ## The key of timestamp. It defaults to @timestamp
    timestampFormat:       ## The format of timestamp. It defaults to 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 not specified, it
represents all namespaces.
      allContainers: false              ## Whether to collect the standard output of all containers in the specified namespace
      container: xxx                    ## Name of the container for log collection, can be left empty here
      includeLabels:                    ## Collect Pods with the specified labels
        k8s-app: xxx                    ## Collect logs only from pods with the label "k8s-app=xxx". This field cannot be specified if
workloads or allContainers=true are specified.
      workloads:                        ## Kubernetes workloads to which the Pods containing the containers to be collected belong
      - 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 collect. If left blank, 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                    ## The name of the container of which the logs will be collected. The * can be used here.
      includeLabels:                    ## Collect Pods with the specified labels
        k8s-app: xxx                    ## Collect logs only from pods with the label "k8s-app=xxx". This field cannot be specified if
workload is specified.
      workload:                         ## The Kubernetes workload to which the container's Pod belongs for log collection
        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: 2023-09-26 11:23:12

Registered Node Management
Registered Node Overview

Last updated: 2023-09-26 14:10:41

What is a Registered Node?

Registered Nodes (formerly known as Third-Party Nodes) are a newly upgraded node product form by Tencent Cloud Container Service team, specifically designed for hybrid cloud deployment scenarios. It allows users to host non-Tencent Cloud hosts in TKE clusters, with users providing computing resources and TKE being responsible for cluster lifecycle management.

Use Cases

Resource Reutilization

Enterprises need to migrate to the cloud, but they have already invested in local data centers and have existing server resources (CPU resources, GPU resources) in the IDC. By using the Registered Node feature, IDC host resources can be added to the TKE public cloud cluster, ensuring that existing server resources are effectively utilized during the cloud migration process.

Cluster Hosting and Maintenance

Eliminate the cost of building and maintaining a local K8s cluster, as Tencent Cloud provides unified operation and management. Users only need to maintain their local servers.

Hybrid Deployment Scheduling

Supports simultaneous scheduling of Registered Nodes and CVM nodes in a single cluster, making it easy to expand on-premises services to the cloud without the need for multi-cluster management.

Seamless Integration with Cloud Services

Registered Nodes seamlessly integrate with Tencent Cloud's cloud-native services, covering logging, monitoring, auditing, storage, container security, and other cloud-native capabilities.

Notes

Constraints

Before using the Registered Node capability, ensure that the environment meets the constraint requirements; otherwise, the Registered Node product features cannot be used properly.

- Operating System Constraints: The operating system of the Registered Node must use TencentOS Server 3.1 and TencentOS Server 2.4 (TK4).
- Hardware Constraints (GPU): Only NVIDIA series graphics cards are supported, including: Volta (e.g., V100), Turing (e.g., T4), and Ampere (e.g., A100, A10).
- Cloud-to-ground Connectivity: CCN + Direct Connect (VPN method is currently not supported).
- TKE Cluster Constraints: **v1.18 and later versions**, there must be at least one CVM node in the cloud (clusters with only registered nodes are not supported for now).

Network mode

For different network types in TKE clusters, there are certain limitations on the Pod network capabilities of Registered Nodes, as described below:

- For GlobalRouter and VPC-CNI exclusive ENI mode clusters: Pods on on-premises IDC nodes only support the hostNetwork network mode.
- VPC-CNI Shared Network Card Mode Cluster: Pods on the on-premises IDC nodes support either hostNetwork or ciliumBGP network mode (choose one).
- Cilium-Overlay Network Mode Cluster: A container network solution specifically introduced by the TKE team for hybrid cloud scenarios, where Pods in both cloud and on-premises environments are on the same overlay network plane.

For more information on network descriptions, see Network Modes.

Comparison of Capabilities between Registered Nodes and Cloud Nodes

Type	Feature	Cloud Nodes	Registered node
Manage Node	Add Node	✔	✔
	Removing a Node	✔	✔
	Configuring Node Labels and Taints	✔	✔
	Node Cordon and Drain	✔	✔
	Batch Management through Node Pools	✔	✔
	Kubernetes Version Upgrade	✔	/
Storage volume	Local Storage (emptyDir, hostPath, etc.)	✔	✔
	Kubernetes API (ConfigMap、Secret…)	✔	✔
	Tencent Cloud Block Storage (CBS)	✔	–
	Tencent Cloud File Storage (CFS)	✔	✔
	Tencent Cloud COS	✔	/
Observability	Supports Prometheus Monitoring Service	✔	✔
	Supports Cloud Monitoring	✔	/
	Log Integration with CLS Supported	✔	✔
	Enabling Cluster Audit	✔	✔
	Event Storage Support	✔	✔
Traffic Access	Supports ClusterIP type Service	✔	✔
	Supports NodePort Type Service	✔	✔
	Supports LoadBalancer Type Service	✔	✔ Based on Tencent Cloud Load Balancer (CLB)
	Ingress Supporting CLB Types	✔	✔
	Supports Nginx-type Ingress	✔	✔
Others	Supports qGPU	✔	✔

Remove registered nodes

Last updated: 2023-09-26 14:11:32

This article explains how to remove registered nodes.

Instructions

You can remove registered nodes in bulk by deleting the registered node pool, or you can specify individual nodes within the node pool for removal.

Delete registered node pool

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the desired cluster ID to go to the **Basic information** page.
3. Select **Node Management** > **Node Pool** in the left menu bar to access the "Node Pool List" page.
4. In the "Node Pool Card" section, click **Delete** in the upper right corner of the node pool card you plan to remove.
5. In the "Delete Registered Node Pool" pop-up window, click **Confirm** to delete the node pool.

Delete specified registered node

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster management** page, click the desired cluster ID to go to the **Basic information** page.
3. Select **Node Management** > **Node Pool** in the left menu bar to access the "Node Pool List" page.
4. In the "Node Pool Card", click the ID of the target node pool to go to the details page of the node pool.
5. Select the registered node you want to delete and click **Remove**.

Note

- After deleting a registered node, the node will only be removed from the cluster and the Pods running on the node will not be cleared.
- For security reasons, after deleting the node, it is recommended to reinstall the node or run the following command to delete the kube-apiserver access configuration on the node:

```
rm - rf /etc/kubernetes $HOME/.kube
```

Traffic Access

Last updated: 2023-09-26 14:11:46

This document primarily explains how to use Tencent Cloud Load Balancer (CLB) to implement traffic access for registered nodes at both layer-4 and layer-7.

Prerequisites

- You have already created and registered nodes in the cluster.
- To utilize Tencent Cloud CLB for registered node traffic access, ensure that the hybrid cloud deployment feature is enabled, as it is based on CLB's hybrid cloud deployment capabilities. For more information, please refer to Hybrid Cloud Deployment.

Service

Kubernetes Service resources abstract the strategy for accessing a group of Pods, shielding the dynamic changes of backend instances and load balancing across multiple instances. They are used to manage layer-4 network service access within the cluster. Kubernetes ServiceTypes allow specifying the Service type, primarily supporting three types: `ClusterIP`, `NodePort`, and `LoadBalancer`. The behavior of ClusterIP and NodePort type Services is generally consistent across different cloud providers and self-built clusters. However, `LoadBalancer` type Services, which utilize the cloud provider's load balancing for service exposure, offer various additional features based on the capabilities of the cloud provider's load balancer.
Here, we introduce how to implement layer-4 traffic access for registered nodes using LoadBalancer type Service based on Tencent Cloud CLB.

Directions

When creating a LoadBalancer Service, add the corresponding Annotation to the Service:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/hybrid-type: CCN
    service.cloud.tencent.com/snat-pro-info: '{"snatIps": [{"subnetId":"subnet-12345678","ip":"192.168.0.1"}]}'
  name: test
  ........
```

Annotation

Here, we will only introduce parameters related to registered node traffic access:

- `service.cloud.tencent.com/hybrid-type`
  This annotation enables the feature of registered node traffic access in distributed cloud scenarios. You need to specify the traffic forwarding method, which currently only supports Cloud Connect Network (CCN). Valid parameter value: CCN.
- `service.cloud.tencent.com/snat-pro-info`
  This annotation is used to configure the SnatIP for CLB, otherwise, the CLB backend cannot bind to the IDC IP. You need to specify the subnet and IP where the SnatIP is located. The IP is optional, and if not provided, it will be assigned by default from the specified subnet.

> **Note**
> The description of SNAT IP is as follows:
> - A SNAT IP is mainly used in hybrid cloud deployment where requests are forwarded to IDC servers. It must be assigned when you bind a CLB instance to an IP in the IDC that is interconnected with CNN, and serves as the private IP of your VPC.
> - A maximum of 10 SNAT IPs can be configured for each CLB instance.
> - For a single CLB instance with a single rule configured with one SNAT IP and bound to one backend service, the maximum number of connections is 55,000. If you increase the number of SNAT IPs or backend services, the number of connections increases proportionally. For example, if a CLB instance is configured with two SNAT IPs and has ten

backend ports bound, the total number of connections for that instance would be: 2 x 10 x 55,000 = 1,100,000. You can assess the allocation of SNAT IPs based on the number of connections.

- **Note that deleting a SNAT IP disconnects all connections on the IP.** Please proceed with caution.

Sample Code

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/hybrid-type: CCN
    service.cloud.tencent.com/snat-pro-info: '{"snatIps": [{"subnetId":"xxxxxxxx"}]}'
  name: nginx-svc
spec:
  ports:
  - port: 80
    targetPort: 80
  selector:
    k8s-app: nginx
  sessionAffinity: None
  externalTrafficPolicy: Local
  type: LoadBalancer
```

Note:

- If `service.cloud.tencent.com/hybrid-type` and `service.cloud.tencent.com/snat-pro-info` are not specified, the CLB will only mount cloud-based nodes.
- It is recommended to use the Service Local mode by setting `ExternalTrafficPolicy` to `Local` to avoid traffic forwarding between nodes through NAT settings and minimize cross-line traffic traversal. For more information on Service Local mode, please refer to Service Local Mode.

Ingress

An Ingress is a collection of rules that allow access to services within a cluster. You can configure different forwarding rules to allow different URLs to access different services within the cluster. For the Ingress resource to function properly, the cluster must run an Ingress Controller. The container service has enabled the TKE Ingress Controller by default within the cluster, which is based on Tencent Cloud Load Balancer (CLB).
Here, we introduce how to use CLB type Ingress based on Tencent Cloud Load Balancer (CLB) to implement layer-7 traffic access for registered nodes.

Directions

When creating a CLB type Ingress, add the corresponding Annotation to the Ingress:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: qcloud
    ingress.cloud.tencent.com/hybrid-type: CCN
    ingress.cloud.tencent.com/snat-pro-info: '{"snatIps": [{"subnetId":"subnet-12345678","ip":"192.168.0.1"}]}'
  name: test-ingress
  ........
```

Annotation

- `kubernetes.io/ingress.class`
  Configure the Ingress type. Here, we use the CLB-type Ingress, with a valid value of **qcloud**.
- `ingress.cloud.tencent.com/hybrid-type`
  This annotation enables the feature of registered node traffic access in distributed cloud scenarios. You need to specify the

traffic forwarding method, which currently only supports Cloud Connect Network (CCN). Valid parameter value: **CCN**.

- `ingress.cloud.tencent.com/snat-pro-info`
  This annotation configures the SnatIP for CLB; otherwise, the CLB backend cannot bind to the IDC IP. You need to specify the subnet where the SnatIP is located and the IP itself, although the IP is optional. If not provided, it will be assigned by default from the specified subnet.

Sample Code

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: qcloud
    ingress.cloud.tencent.com/hybrid-type: CCN
    ingress.cloud.tencent.com/snat-pro-info: '{"snatIps": [{"subnetId":"subnet-xxxxxx"}]}'
  name: nginx-ingress
  namespace: default
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: nginx-svc
          servicePort: 80
        path: /
        pathType: ImplementationSpecific
```

Registered Node FAQs

Last updated: 2023-09-26 14:11:55

Cluster-related

- Does the node registration feature support independent deployment of clusters?
- Why does the node registration feature require the presence of cloud nodes within the cluster?

Node-related

- What are the differences in capabilities between registered nodes and cloud nodes?
- Which operating systems are supported by registered nodes?
- How to handle the failure of adding a node due to docker or containerd-related software on the node?

Network & Traffic Access-related

- How do containers on registered nodes expose services externally?

Ops

- How can registered node logs be integrated with CLS?
- How can registered nodes be integrated with Prometheus monitoring service?

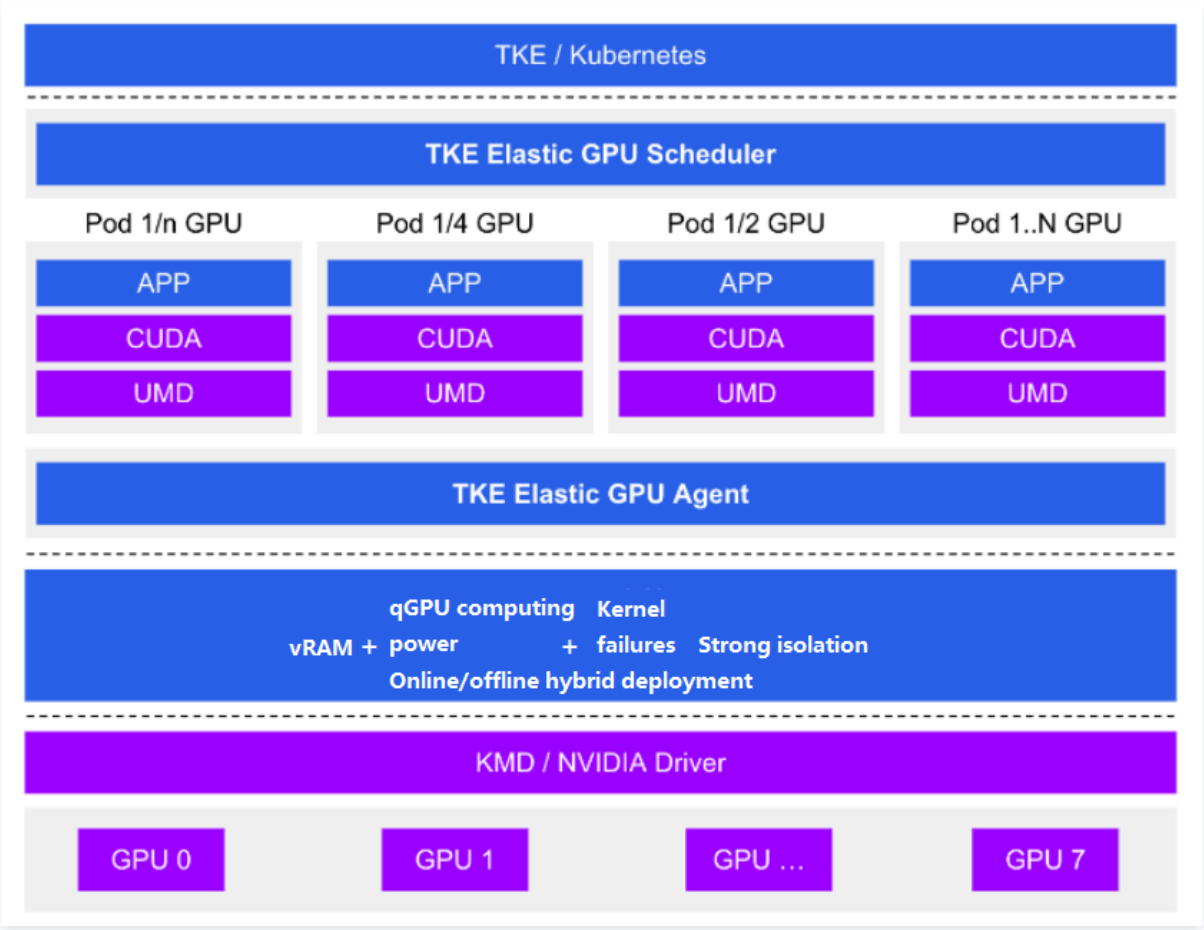
GPU Share
qGPU Overview

Last updated: 2023-09-26 14:12:10

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.

Solution Framework Diagram



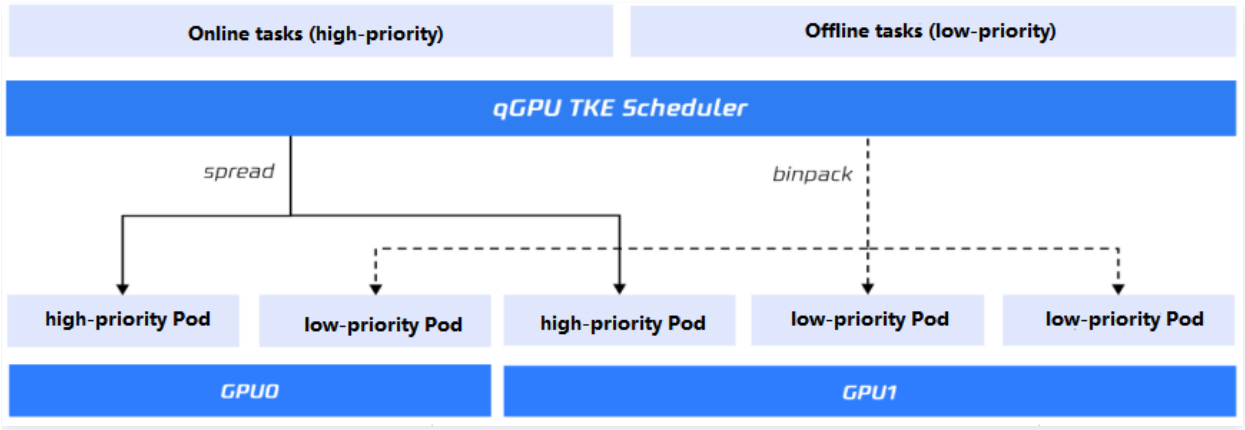
qGPU Online/Offline Hybrid Deployment Description

Last updated: 2023-09-26 14:12:55

Feature Overview

In typical scenarios, qGPU Pods fairly utilize physical GPU resources, with the qGPU kernel driver allocating equivalent GPU time slices for each task. However, different GPU computing tasks have varying characteristics and importance, leading to diverse GPU resource usage and requirements. For example, real-time inference is sensitive to GPU resources, requiring low latency and quick access to GPU resources for computation, but its GPU resource utilization is usually not high. Model training requires a larger amount of GPU resources but has a lower sensitivity to latency and can tolerate some suppression.

In this context, Tencent Cloud has introduced the qGPU online/offline hybrid deployment feature. qGPU online/offline hybrid deployment is an innovative GPU scheduling technology launched by Tencent Cloud, supporting the simultaneous mixed deployment of online (high-priority) tasks and offline (low-priority) tasks on the same GPU card. At the kernel and driver level, it achieves 100% utilization of idle computing power for low-priority tasks and 100% preemption for high-priority tasks. Relying on qGPU online/offline hybrid scheduling technology, users' GPU resources can be further utilized, increasing GPU utilization to 100% and minimizing GPU usage 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

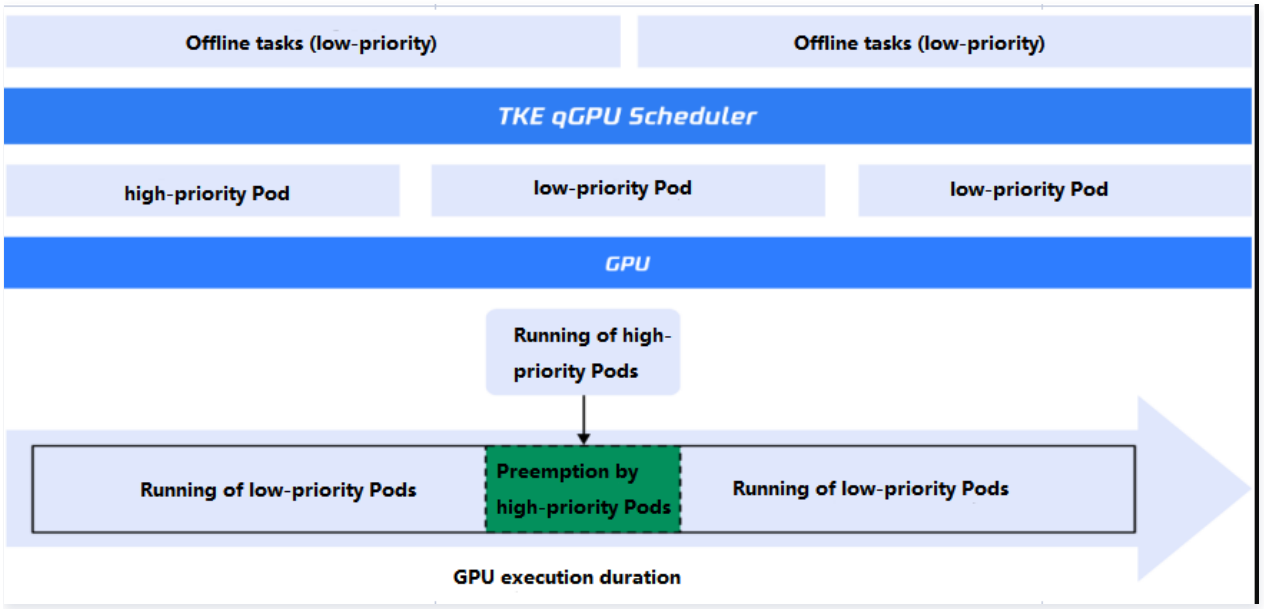
Hybrid deployment of online and offline inference

Search/recommendation inference tasks support online services and require real-time GPU computing power. Data preprocessing inference tasks support offline data cleaning and processing and have lower real-time requirements for GPU computing power. By setting online inference tasks as high-priority and offline inference tasks as low-priority, they can be hybrid deployed 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.

Technical Principles



With the online/offline scheduling strategy provided by TKE clusters, qGPU online/offline hybrid deployment capability can be enabled, helping online tasks (high-priority) and offline tasks (low-priority) to share physical GPU resources more efficiently. qGPU online/offline hybrid deployment technology mainly includes 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.

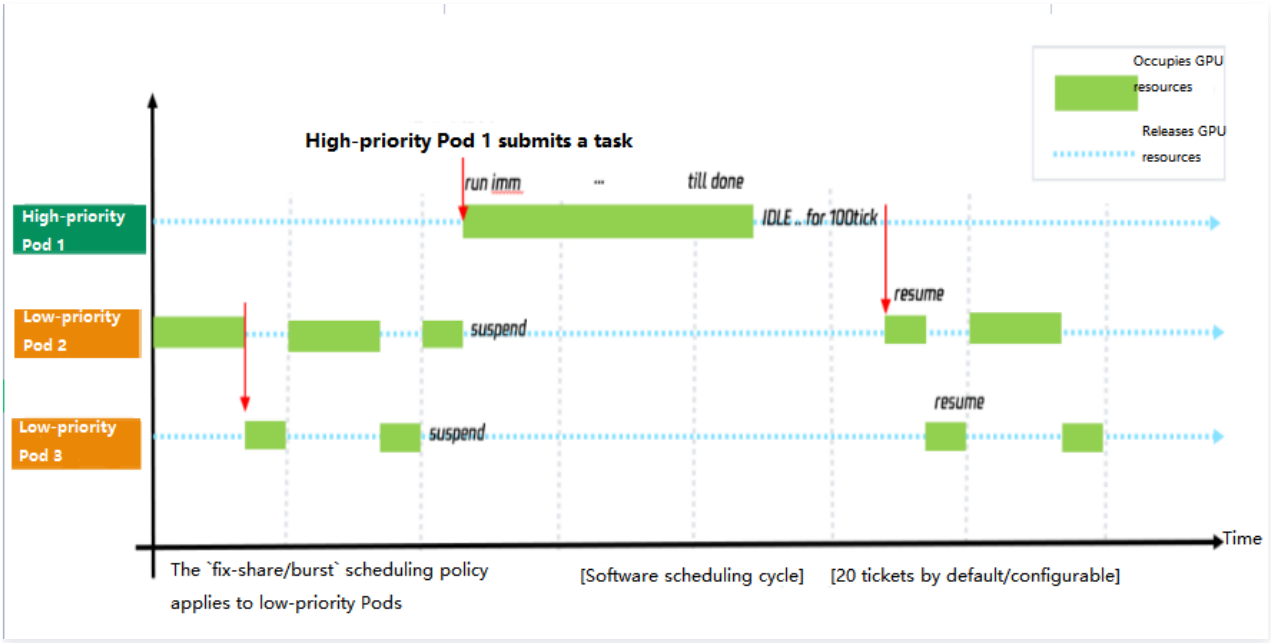
Feature 2: 100% preemption of the computing power of low-priority Pods

qGPU online/offline hybrid deployment provides a priority preemption capability, ensuring that high-priority Pods can immediately and fully utilize GPU computing resources when busy. This is achieved through a priority preemption scheduling strategy. We have implemented this absolute preemption capability at the qGPU driver level:

Firstly, the qGPU driver can sense the demand for GPU computing power from high-priority Pods. As soon as a high-priority Pod submits a computing task involving GPU power, the qGPU driver will provide all the computing power to the high-priority Pod in the shortest time possible, with a response time controlled within 1ms. When the high-priority Pod has no tasks running, the driver will release the occupied computing power within 100ms and reallocate it to the offline Pods.

Secondly, the qGPU driver can support the pause and resume of computing tasks. When a high-priority Pod has a computing task running, the low-priority Pod that originally occupied the GPU will be immediately paused, releasing the GPU computing power for the high-priority Pod. When the high-priority Pod task is completed, the low-priority Pod will be promptly awakened and continue

computing from the interruption point. The timing diagram of the computing tasks running at various priority levels is 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 in a dormant state, low-priority Pods are running, and scheduling between low-priority Pods still follows the policy strategy. As soon as high-priority Pods start using GPU computing power, all low-priority Pods will be paused immediately until the high-priority Pods' computation tasks are completed. Then, low-priority tasks will resume running according to the policy strategy.
- High-priority Pods: High-priority Pods will immediately preempt GPU computing power when they have computing tasks. The relationship between high-priority Pods and low-priority Pods is absolute preemption, unaffected by specific policies. The allocation of GPU computing power among multiple high-priority Pods is a competitive mode, not controlled by specific policy strategies.

Using qGPU Online/Offline Hybrid Deployment

Last updated: 2024-06-14 14:55:18

This document describes how to use qGPU online/offline hybrid deployment.

Step 1: Preliminary Preparations

1. Before using qGPU and its offline hybrid deployment, you need to have a cluster. If you don't have one, please refer to Creating a TKE Cluster.
2. Proceed sequentially with Step 1: Installing the qGPU Scheduling Component, Step 2: Enabling qGPU Sharing in the Cluster, and Step 3: Preparing GPU Resources for using qGPU.

Step 2: Reconstruct Component

> **Note:**
> Ensure that the application is scheduled to the node only after the reconstruction action below is completed; otherwise, the offline and online hybrid function will not work properly.
> If you have created the workload before creating the nodes, it is recommended to block the nodes when creating the node pool. After completing the reconstruction steps below, unblock the nodes to prevent scheduling applications to the nodes during the reconstruction process.

1. After the node is created, it is necessary to rebuild the qgpu-manager Pod and the qgpu-scheduler Pod on the node for the configuration to take effect. The command is as follows:

```
$ kubectl delete pod qgpu-manager-6mcrk -n kube-system
$ kubectl delete po -n kube-system -l app=qgpu-scheduler
```

2. To check if the node has qGPU low-priority computing resources, execute the `kubectl describe node` command and check for the following resources:

```
tke.cloud.tencent.com/qgpu-core-greedy:  100
tke.cloud.tencent.com/qgpu-memory:       15
```

Step 3: Configure Business Operations

Offline Pods

The `tke.cloud.tencent.com/app-class: offline` label indicates an offline Pod, and the `tke.cloud.tencent.com/qgpu-core-greedy` is used to apply for offline computing power. It should be noted that offline Pods do not support multiple cards, and both low-priority computing power and memory need to be set simultaneously:

- Low-priority computing power: tke.cloud.tencent.com/qgpu-core-greedy, measured in percentage.
- Memory: tke.cloud.tencent.com/qgpu-memory, measured in GB.

```
apiVersion: v1
kind: Pod
annotations:
tke.cloud.tencent.com/app-class: offline // Low-priority label
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
```

Online Pods

You can use `tke.cloud.tencent.com/app-class: online` to identify an online Pod. You need to apply for only video memory but not computing power.

> **Note:**
> The video memory is statically partitioned. High-priority Pods will not preempt the video memory resources of low-priority Pods.

```
apiVersion: v1
kind: Pod
annotations:
  tke.cloud.tencent.com/app-class: online
spec:
  containers:
     - name: online-container
    resources:
      requests:
        tke.cloud.tencent.com/qgpu-memory: xx
```

General Pods

The `tke.cloud.tencent.com/app-class` annotation is not involved. A general Pod supports multiple cards.

```
apiVersion: v1
kind: Pod
spec:
  containers:
  - name: common-container
    resources:
      requests:
        tke.cloud.tencent.com/qgpu-core: xx
        tke.cloud.tencent.com/qgpu-memory: xx
```

Kubernetes Object Management Overview

Last updated: 2023-09-26 18:07:26

Object Management Instructions

You can directly operate native Kubernetes objects in the console, such as Deployment, DaemonSet, etc.
Kubernetes objects are persistent entities within the cluster, used to support the running services. Different Kubernetes objects can express different meanings:

- Running applications
- Resources available to applications
- Policies associated with applications

You can directly use Kubernetes objects through the TKE console or Kubernetes API, such as Kubectl.

Object Types

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 define different types based on your needs.	Service 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	Types used to describe storage. When creating a PVC, a specific type of storage is created through StorageClass, which serves as a template for storage.	

Kubernetes objects also include dozens of types such as Namespaces, HPA, Resource Quotas, etc. You can use different Kubernetes objects according to your business needs. Different versions of Kubernetes support different objects. For more information, visit the Kubernetes official website.

Resource limits

TKE applies the following resource restrictions to all **managed clusters** using ResourceQuota/tke-default-quota. If you need more quota items, please contact us online to apply.

Cluster Scale	Quota	
	Pod	ConfigMap
Number of nodes ⩽ 5	4000	3000
5 < Node Count ⩽ 20	8000	6000
Number of nodes > 20	None	None

Workload
Deployment Management

Last updated: 2023-09-26 18:08:10

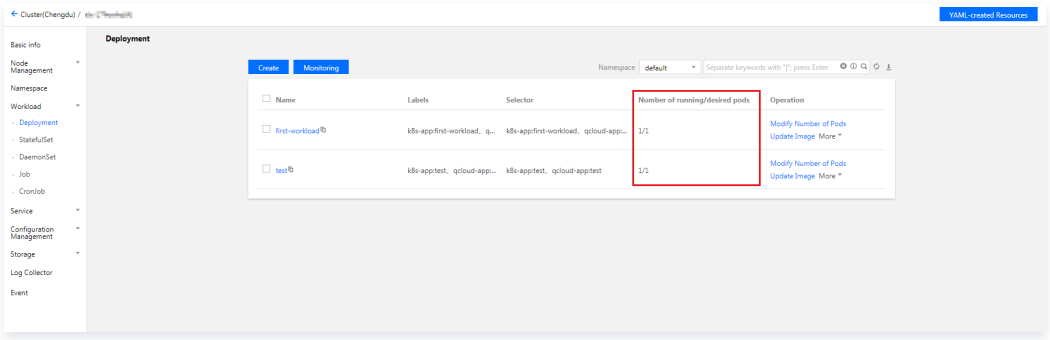
Feature Overview

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 click Cluster in the left sidebar.
2. Click the ID of the target cluster to go to the details page of the cluster where the Deployment needs to be created.
3. Click **Create** to enter the "New Deployment" page. Set the Deployment parameters according to your requirements. Key parameter information is as follows:
   - **Name**: Customize the workload 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 actual 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 an image based on your requirements.
     - **Image Tag**: fill as needed.
     - **Image Pull Policy**: Three policies are available, please choose according to your needs.
       If the image pull policy is not set, the Always policy will be used when the image version is empty or `latest`, otherwise the IfNotPresent policy will be 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 instances by clicking "+" or "−" to control the instance count.
     - **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, as shown in the figure below:
   When the number of running Pods equals the desired number, it indicates that all Pods under the Deployment have been created.



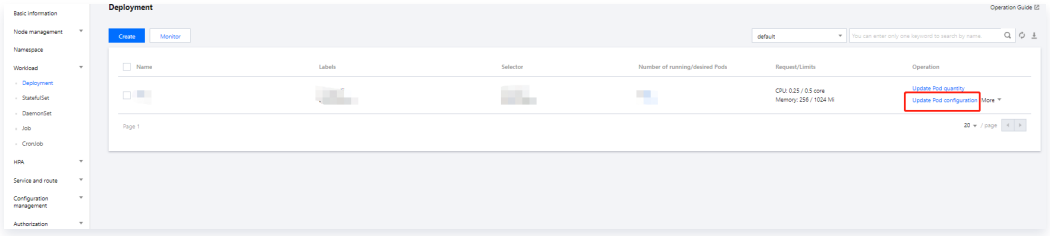
Updating a Deployment

Updating YAML

1. Log in to the TKE console and click 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 Deployment" page, edit the YAML and click **Finish** to update the YAML.

Updating Pod configuration

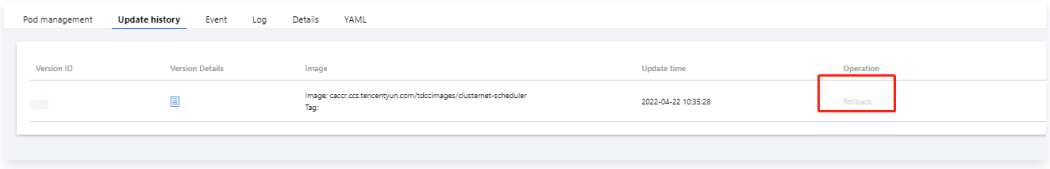
1. On the cluster management page, click the ID of the cluster for which Pod configuration should be updated to go to the management page of the cluster.
2. In the row of the Deployment for which you want to update the Pod configuration, click **Update Pod Configuration**, as shown in the figure below:



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

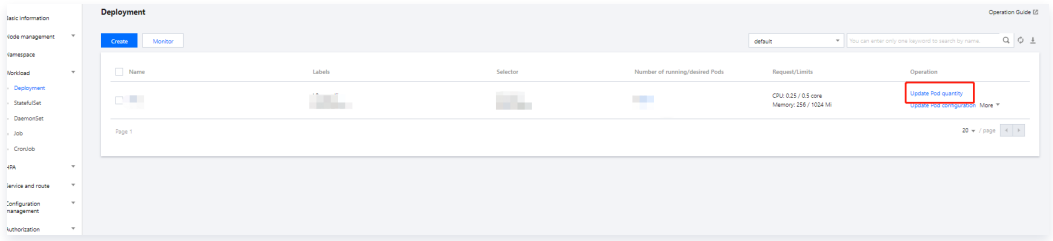
1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click the ID of the cluster for which to roll back the Deployment to go to the management page of the cluster.
3. Click the name of the Deployment to be rolled back to go to the Deployment information page.
4. Select the **Revision History** tab, locate the version you want to roll back, and click **Roll Back**. See the image below:



5. In the "Roll Back Resources" pop-up, click **OK** to complete the rollback.

Adjusting Pod quantity

1. Log in to the TKE console and click 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 you want to adjust the number of Pods, click **Update Pod Quantity** to go to the Update Pod Quantity page, as shown in the figure below:



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**: Deployment resource type.
- **metadata**: basic information such as Deployment name, Namespace, and Label.
- **metadata.annotations**: Additional description of the Deployment. You can set additional enhancements for TKE through this parameter.
- **spec.replicas**: The number of Pods managed by the Deployment.
- **spec.selector**: The Deployment manages the Pods with Labels selected by the Selector.
- **spec.template**: Detailed template configuration for Pods 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

There are three methods to update a Deployment using Kubectl. Both Method 1 and Method 2 support the **Recreate** and **RollingUpdate** update strategies.

- The Recreate update policy is to first terminate all Pods and then recreate the Deployment.
- The RollingUpdate update strategy is a rolling update policy that updates Pods in a Deployment one by one. RollingUpdate also supports pausing and setting update time intervals.

Method A

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.

Method B

Run the following command to update the image of the specified container.

```
kubectl set image deployment/[name] [containerName]=[image:tag]
```

For updates, we recommend that you change none of the Deployment parameters but the one for container's image.

Method C

Run the following command to roll update the specified resource.

```
kubectl rolling-update [NAME] -f FILE
```

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 rollback to the earlier version.

```
kubectl rollout undo deployment/[name]
```

To specify the version number to rollback, run the following command.

```
kubectl rollout undo deployment/[name] --to-revision=[REVISION]
```

Using kubectl to adjust Pod quantity

Manually updating the Pod quantity

Run the following command to manually update the Pod quantity.

```
kubectl scale deployment [NAME] --replicas=[NUMBER]
```

Automatically updating the Pod quantity

Prerequisites
The HPA features of the cluster is enabled. By default, these features have been enabled for clusters created by TKE.
Operation Steps
Run the following command to set automatic scaling for the Deployment.

```
kubectl autoscale deployment [NAME] --min=10 --max=15 --cpu-percent=80
```

Using kubectl to delete a Deployment

Run the following command to delete a Deployment.

```
kubectl delete deployment [NAME]
```

StatefulSet Management

Last updated: 2023-09-26 14:18:00

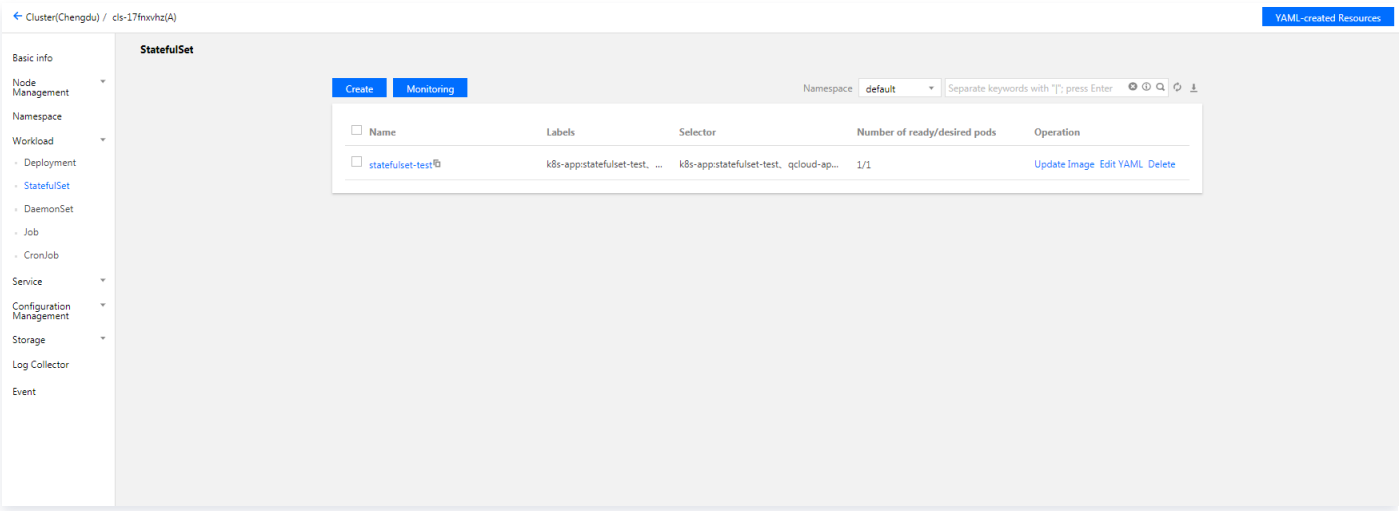
Feature Overview

StatefulSet is primarily used for managing stateful applications and can create Pods with persistent identifiers. These identifiers are retained even after the Pod is migrated or terminated and restarted. When persistent storage is required, you can map storage volumes to identifiers one-to-one. If your application does not require a persistent identifier, it is recommended to use Deployment for deploying the application.

Operation Guide for StatefulSets in the Console

Creating a StatefulSet

1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click the ID of the cluster where StatefulSet needs to be created to enter the cluster management page.
3. Select **Workload** > **StatefulSet** to access the StatefulSet management page, as shown below:



4. Click **New** to enter the "Create Workload" page. Set the StatefulSet parameters according to your requirements. Key parameter information is as follows:
   - **Workload Name**: Enter a custom name.
   - **Label**: a key-value pair, which is used for classified management of resources.
   - **Namespace**: select a namespace based on your actual 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 an image based on your requirements.
     - **Image Tag**: fill as needed.
     - **Image Pull Policy**: Three policies are available, please choose according to your needs.
       If the image pull policy is not set, the Always policy is used when the image version is empty or `latest`, 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**: parameters such as "**working directory**", "**run commands**", "**run parameters**", "**container health check**", and "**privilege level**" can be set.

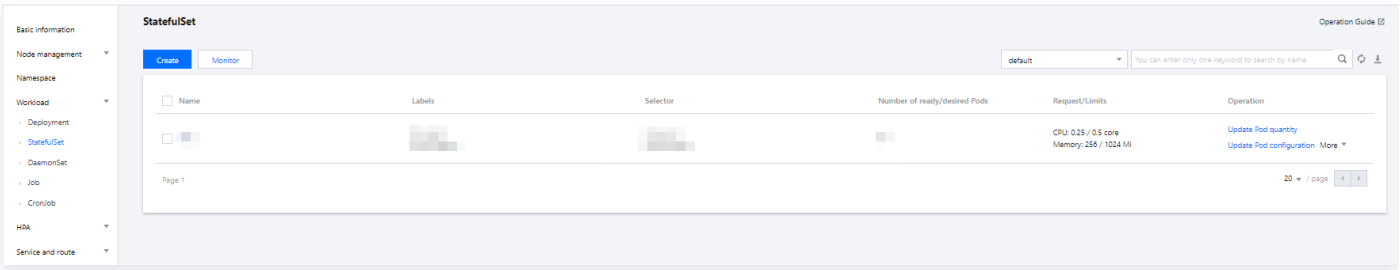
- **Image Access Credential**: a container image is private by default. You need to select the image access credential for the TCR instance when creating a workload.
- **Number of Pods**: select the adjustment method and set the number of Pods based on actual needs.
- **Node Scheduling Policy**: the 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**.

Updating a StatefulSet

Updating YAML

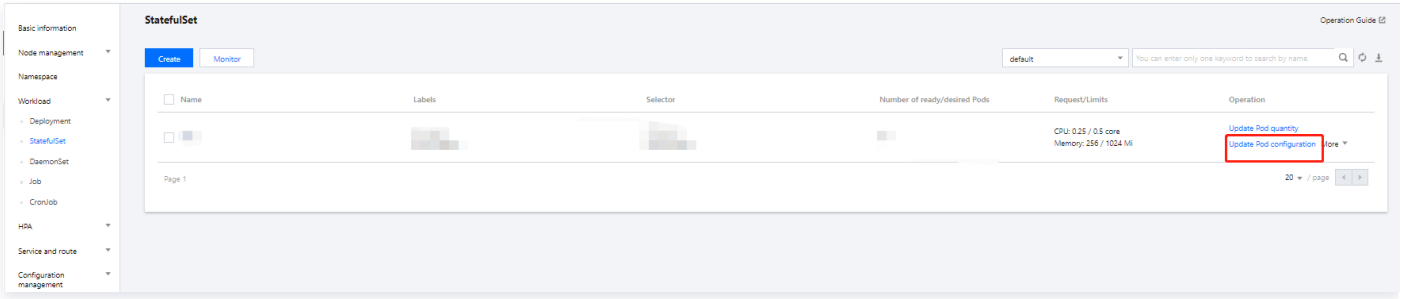
1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click the cluster ID for which you want to update the YAML to go to the cluster management page.
3. Select **Workload** > **StatefulSet** to access the StatefulSet information page, as shown in the figure below:



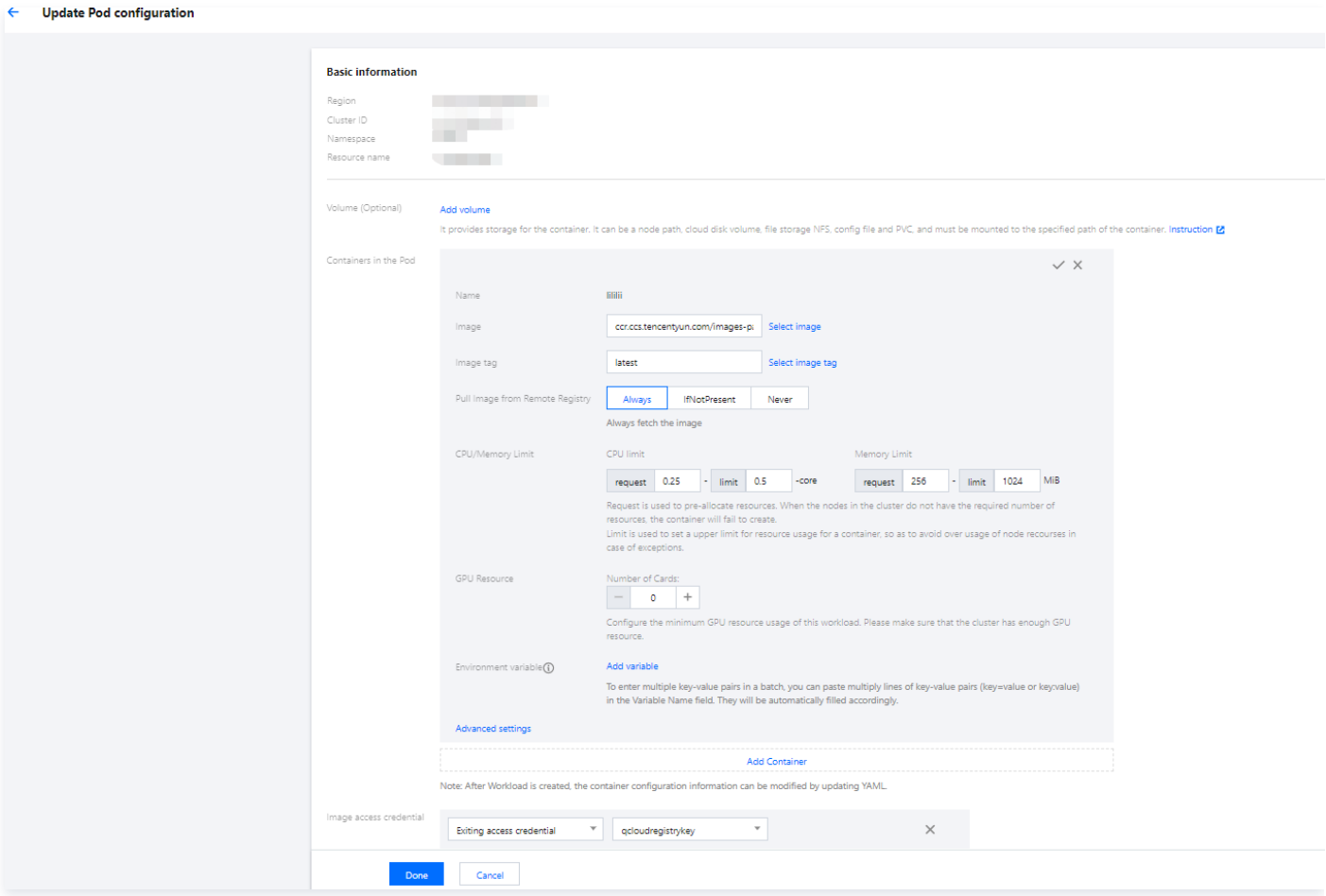
4. In the row of the StatefulSet for which YAML should be updated, click **More** > **Edit YAML** to go to the StatefulSet updating page.
5. On the "Update StatefulSet" page, edit the YAML and click **Finish** to update the YAML.

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 row of the StatefulSet for which you want to update the Pod configuration, click **Update Pod Configuration**. As shown in the image below:



3. On the "Update Pod Configuration" page, modify the update method and set parameters according to your actual requirements, as shown in the image below:



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 an 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**: StatefulSet resource type.
- **metadata**: Basic information such as StatefulSet name and Label.
- **metadata.annotations**: Additional description for the StatefulSet. You can set additional enhancements for Tencent Cloud TKE through this parameter.
- **spec.template**: Detailed template configuration for Pods managed by the StatefulSet.
- **spec.volumeClaimTemplates**: A template for creating PVC & PV.

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 update strategy. With this strategy, after updating the StatefulSet, you need to manually delete the old StatefulSet Pod before a new StatefulSet Pod is created.
- RollingUpdate: Supports Kubernetes 1.7 or higher versions. With this update strategy, after updating the StatefulSet template, the old StatefulSet Pods will be terminated, and new StatefulSet Pods will be created using a rolling update method (Kubernetes 1.7 or higher versions).

Method A

Run the following command to update a StatefulSet.

```
kubectl edit StatefulSet/[name]
```

This method is suitable for simple debugging and validation, but not recommended for direct use in production environments. You can use this method to update any StatefulSet parameter.

Method B

Run the following command to update the image of the specified container.

```
kubectl patch statefulset <NAME> --type='json' -p='[{"op": "replace", "path": "/spec/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 will only delete the StatefulSet and not any Pods. To delete Pods, execute the following command:

```
kubectl delete  StatefulSet [NAME]
```

For more information about StatefulSet operations, see Kubernetes' official guide.

DaemonSet Management

Last updated: 2023-09-26 14:18:40

Feature Overview

DaemonSet is primarily used for deploying background processes that reside within the cluster, such as node log collection. DaemonSet ensures that the specified Pod runs on all or a subset of nodes. When new nodes are added to the cluster, Pods are automatically deployed; when nodes are removed from the cluster, Pods are automatically reclaimed.

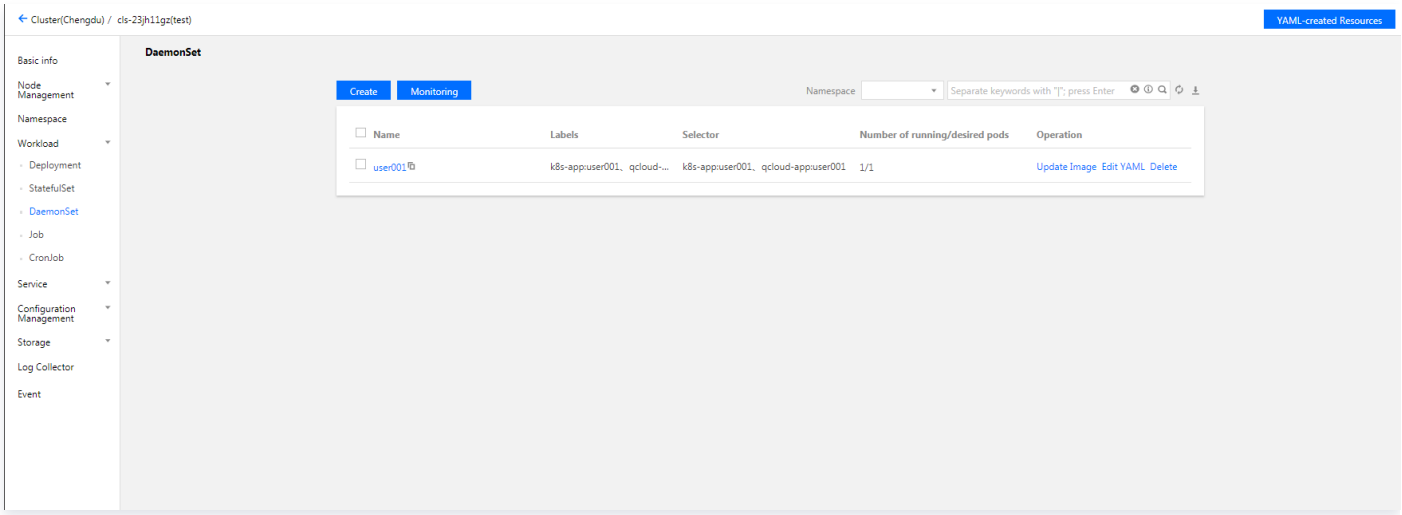
Notes on Scheduling

If the nodeSelector or affinity parameters are configured for a Pod, the DaemonSet-managed Pod will be scheduled according to the specified rules. If the nodeSelector or affinity parameters are not configured for a 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 click 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 access the DaemonSet information page. As shown below:



4. Click **Create** to enter the "New Workload" page.
   Set the DaemonSet parameters according to your requirements. Key parameter information is as follows:
   - **Workload Name**: Enter a custom name.
   - **Label**: a key-value pair, which is used for classified management of resources.
   - **Namespace**: select a namespace based on your actual 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 an image based on your requirements.
     - **Image Tag**: fill as needed.
     - **Image Pull Policy**: Three policies are provided, please choose according to your needs.
       If the image pull policy is not set, the Always policy will be used when the image version is empty or `latest`; otherwise, the IfNotPresent policy will be 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**.

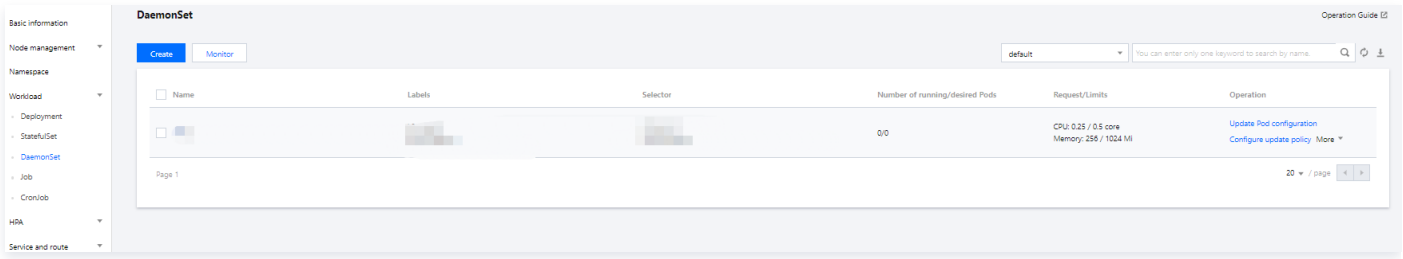
- **Image Access Credential**: a container image is private by default. You need to select the image access credential for the TCR instance when creating a workload.
- **Node Scheduling Policy**: the Pod can be scheduled to the node of the Label that meets the expectation according to the scheduling rules.

5. Click **Create Workload**.

Updating a DaemonSet

Updating YAML

1. Log in to the Container Service Console and select ** Cluster ** in the left sidebar.
2. Click the cluster ID for which you want to update the YAML to go to the cluster management page.
3. Select **Workload** > **DaemonSet** to access the DaemonSet information page. As shown below:

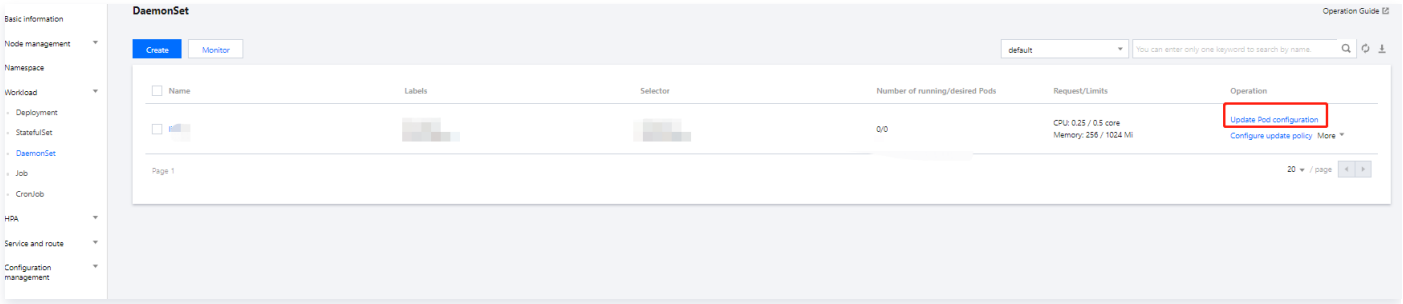


4. In the row of DaemonSet for which YAML is to be updated, click **More** > **Edit YAML** to go to the DaemonSet updating page.
5. On this page, edit the YAML and click **Done** to update the YAML.

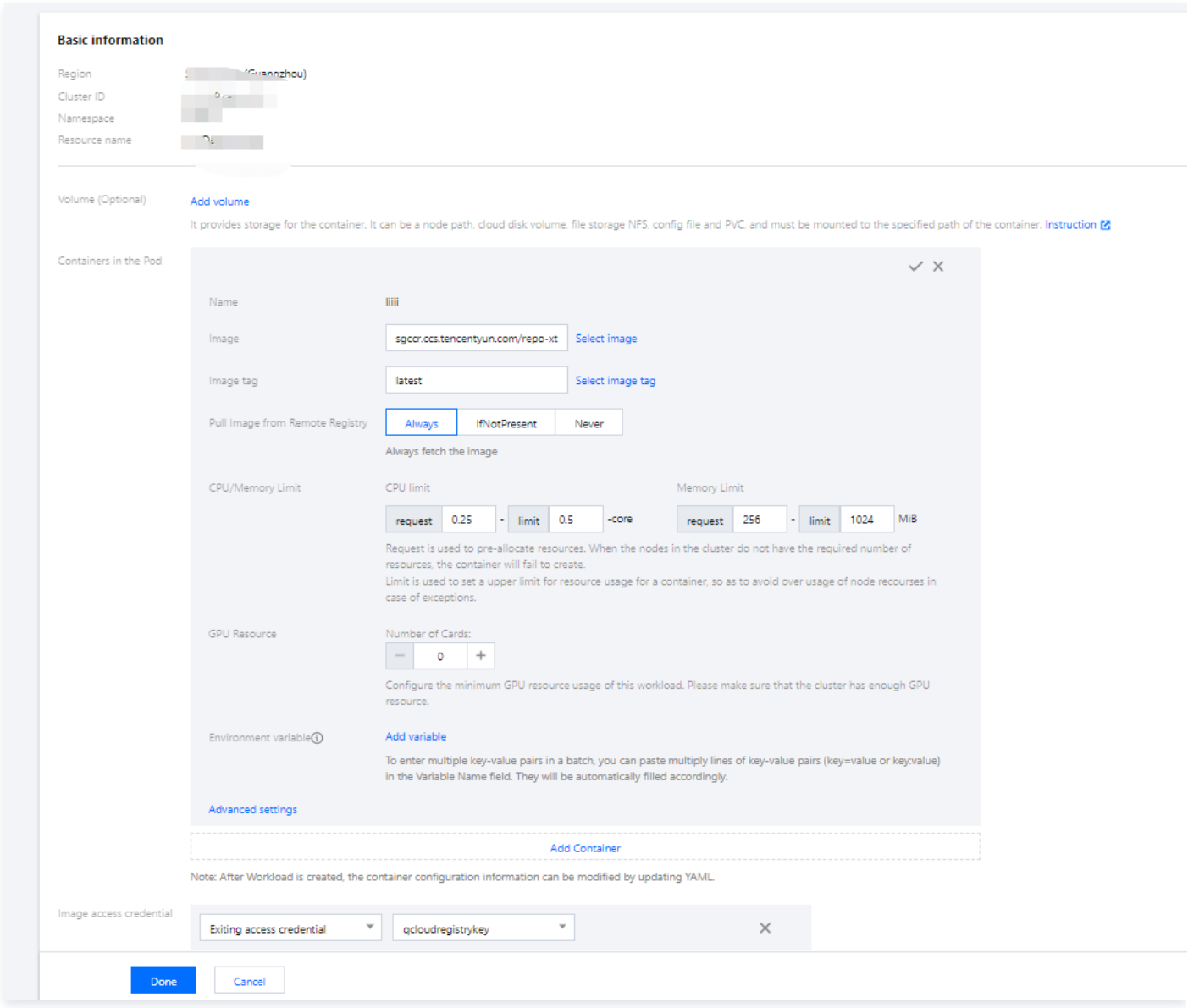
Updating Pod configuration

> **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 row of the DaemonSet for which you want to update the Pod configuration, click **Update Pod Configuration**. See the image below:



3. On the "Update Pod Configuration" page, modify the update method and set parameters according to your actual needs, as shown in the following figure:



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**: Identifies the DaemonSet resource type.
- **metadata**: Basic information such as DaemonSet name and Label.
- **metadata.annotations**: Additional description of the DaemonSet. You can set additional enhancements for Tencent Cloud TKE through this parameter.
- **spec.template**: Detailed template configuration for the Pod managed by DaemonSet.

For more information, see the Official Kubernetes DaemonSet Documentation.

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 strategy. With this strategy, after updating the DaemonSet, you must manually delete the old DaemonSet Pod before a new DaemonSet Pod is created.
- RollingUpdate: Supported in Kubernetes v1.6 or higher versions. With this update strategy, after updating the DaemonSet template, the old DaemonSet Pod will be terminated, and new DaemonSet Pods will be created using a rolling update approach.

Method A

Run the following command to update a DaemonSet.

```
kubectl edit DaemonSet/[name]
```

This method is suitable for simple debugging and validation, but it is not recommended for direct use in production environments. You can use this method to update any DaemonSet parameter.

Method B

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 rollback to the earlier version.

   ```
   kubectl rollout undo daemonset /[name]
   ```

   To specify the version number to rollback, 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: 2023-09-26 14:19:04

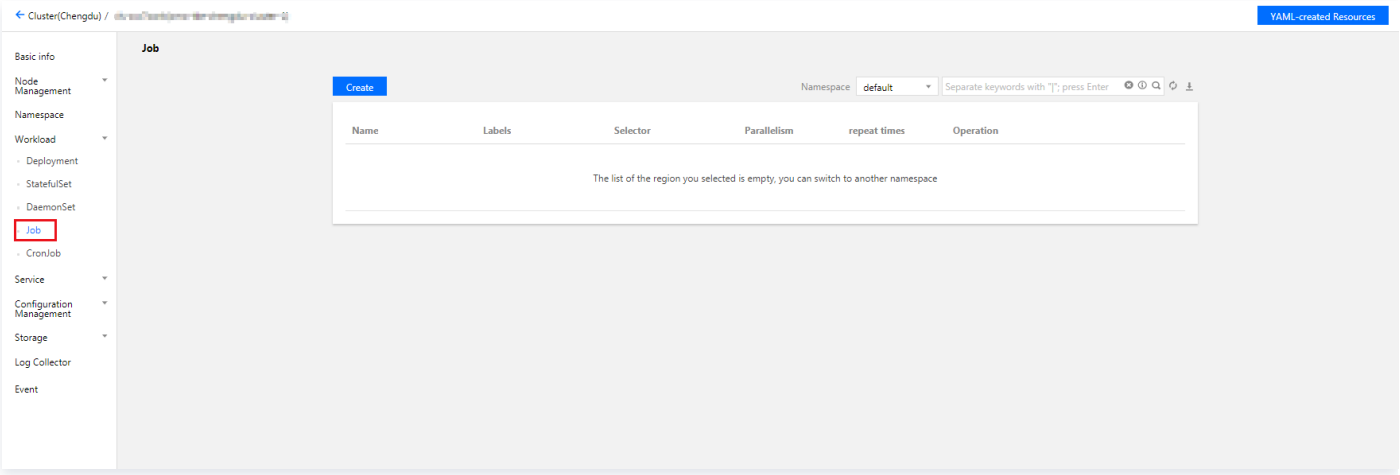
Feature Overview

The Job controller creates 1-N Pods that run according to specified rules until they complete. Jobs are suitable for batch processing, data analysis, and other scenarios. You can meet business requirements by setting the number of repetitions, parallelism, and restart policies.
Once a Job is completed, it will not create new Pods nor delete existing ones. You can view the logs of completed Pods in the "Logs" section. If you delete a Job, the Pods created by the Job will also be deleted, and you will not be able to view the logs of the Pods created by that Job.

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 access the Job information page, as shown in the figure below:



5. Click **Create** to enter the "Create Workload" page, as shown in the figure below:
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 actual 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 an image based on your requirements.
     - **Image Tag**: enter the tag based on your actual needs.

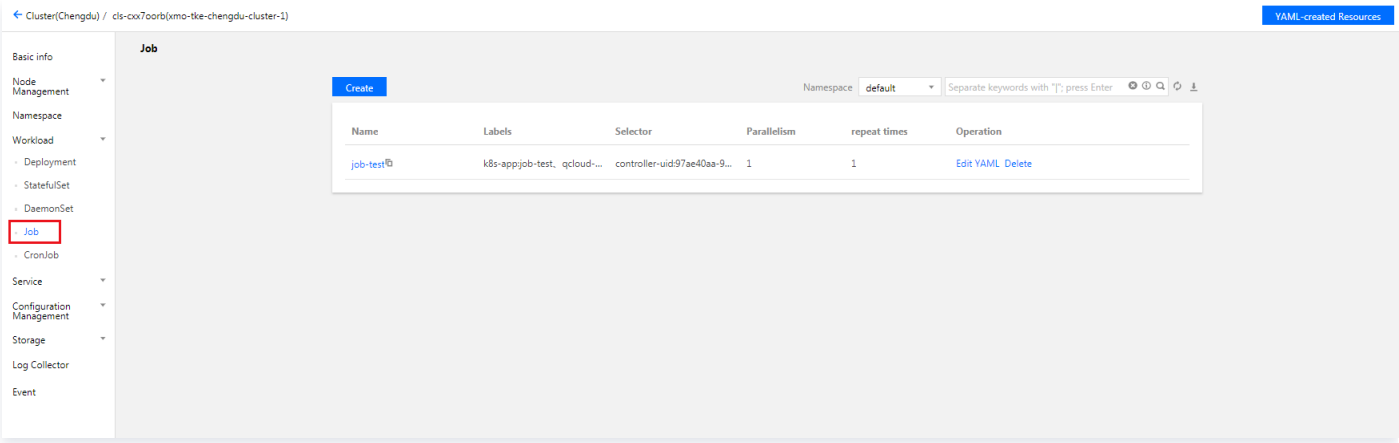
- **CPU/Memory Limit**: set the CPU and memory limit according to Kubernetes' resource limits to improve the robustness of the business.
- **GPU Resource**: you can configure the least GPU resource used by the workload.
- **Advanced Settings**: You can configure parameters such as "Working Directory", "Run Command", "Run Parameters", "Container Health Check", and "Privilege Level".

- **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**.

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. Navigate to "Workloads" > "Job" to access the Job information page, as shown in the figure below:



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]
```

Setting the Resource Limit of Workload

Last updated: 2023-09-26 14:19:19

Requests and Limits

Request: The minimum resource requirement for a container, used as a basis for resource allocation during container scheduling. A container is only scheduled to a node if the available resources on the node >= the container's resource request. However, the Request parameter does not limit the maximum resource usage of the container.
Limit: The maximum amount of resources a container can use.
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.

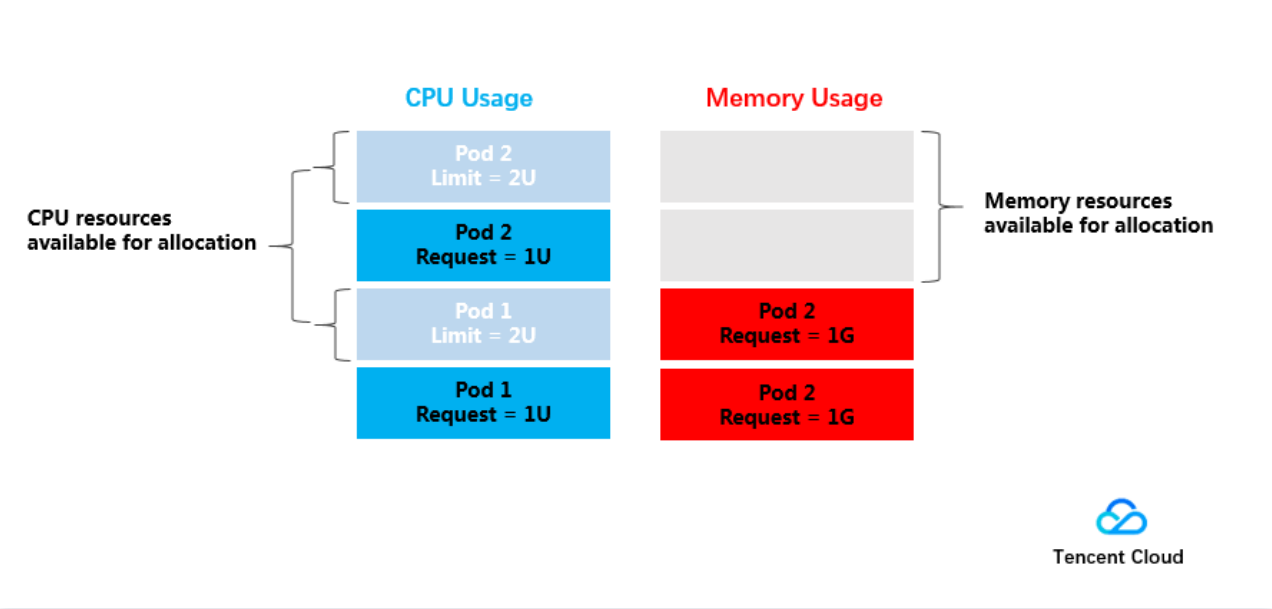
CPU Usage and CPU Utilization

- 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).

Sample Code

Here is a simple example to illustrate the role of Request and Limit. The test cluster includes one 4U4G node and two deployed Pods (Pod1, Pod2). Each Pod's resource settings are (CPU Request, CPU Limit, Memory Request, Memory Limit) = (1U, 2U, 1G, 1G). (1.0G = 1000MiB)

The CPU and memory resource usage on the node is shown below:



Allocated CPU resources: 1U (allocated to Pod1) + 1U (allocated to Pod2) = 2U, leaving 2U of available CPU resources.
Allocated memory resources: 1G (allocated to Pod1) + 1G (allocated to Pod2) = 2G, leaving 2G of available memory resources.

Thus, the node can deploy an additional Pod with (CPU Request, Memory Request) = (2U, 2G) or two Pods with (CPU Request, Memory Request) = (1U, 1G) each.
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:
First, we extract the minute-level workload of the current container image for the past 7 days and use the 95th percentile value for statistical analysis to determine the recommended Request. The Limit is set to twice the Request value.

```
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.
Recommended value is empty:
During usage, you may find that some values have no recommended values, which could be due to the following reasons:

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**
> - 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.
> - 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.
> - The recommended values are only for reference. You can make adjustments based on the actual business needs.

Documentation

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 Health Check for a Workload

Last updated: 2023-09-26 14:19:46

Tencent Cloud container clusters are built on the Kubernetes kernel. Kubernetes supports periodic probing of containers, determining their health status based on the probe results, and performing additional actions accordingly.

Health Check Types

Health checks are divided into the following types:

- **Container Liveness Check**: Used to detect whether a container is alive, similar to running the ps command to check if a process exists. If the container's liveness check fails, the cluster will restart the container. If the container's liveness check succeeds, no action will be taken.
- **Container Readiness Check**: Used to determine whether a container is ready to handle user requests. For example, when an application takes a long time to start, it may need to load disk data or depend on an external module to start before it can provide services. In this case, a container readiness check can be used to inspect the application process and confirm whether it has started successfully. If the container's readiness check fails, the cluster will block requests to access the container. If the container's readiness check succeeds, access to the container will be allowed.

Health Check Methods

TCP Port Probe

The principle of TCP port probing is as follows:
For containers providing TCP communication services, the cluster periodically establishes TCP connections to the container. If the connection is successful, the probe is considered successful; otherwise, it fails. When choosing TCP port probing, the listening port of the container must be specified.

For example, a Redis container has a service port of 6379. If we configure TCP port probing for this container and specify the probing port as 6379, the cluster will periodically initiate TCP connections to the container's 6379 port. If the connection is successful, the check is considered successful; otherwise, it fails.

HTTP Request Probe

HTTP request probing is designed for containers providing HTTP/HTTPS services, and the cluster periodically sends HTTP/HTTPS GET requests to the container. If the HTTP/HTTPS response code falls within the range of 200 – 399, the probe is considered successful; otherwise, it fails. When using HTTP request probing, you must specify the container's listening port and the HTTP/HTTPS request path.

For example, for a container providing HTTP services with a server port of 80 and an HTTP check path of `/health-check`, the cluster will periodically send `GET http://containerIP:80/health-check` requests to the container.

Run Command Check

Executable command check is a powerful inspection method that requires users to specify an executable command within a container. The cluster will periodically execute this command inside the container. If the command returns a result of 0, the check is successful; otherwise, it fails.
For TCP port probing and HTTP request probing, both can be replaced by using the executable command check method:

- For TCP port probing, you can write a program to connect to the container's port. If the connection is successful, the script returns 0; otherwise, it returns -1.
- For HTTP request probes, you can create a script to perform a wget on the container and check the response code. For example, `wget http://127.0.0.1:80/health-check`. If the response code is within the range of 200 – 399, the script returns 0; otherwise, it returns -1.

Supports and Limits

- The program to be run must be placed in the image of the container; otherwise, the run will fail as the program cannot be found.
- If the command being executed is a shell script, you cannot directly specify the script as the execution command; instead, you need to add the script interpreter. For example, if the script is `/data/scripts/health_check.sh`, then when using the execution command check, the specified program should be:

```
sh
```

```
/data/scripts/health_check.sh
```

Setting steps using the TKE console to create a Deployment as an example:

1.1 On the "Deployment" page of the cluster, click **Create**.

1.2 Navigate to the "Create Workload" page and select **Show Advanced Settings** located below the "Container Instance" module.

1.3 In "Container Health Check", using **Liveness Check** as an example, set the following parameters.

- **Check Method**: Select "Executable Command Check".
- **Execute Command**: Enter the following content.

```
sh
```

```
/data/scripts/health_check.sh
```

1.4 For other parameter settings, please refer to Deployment Management.

Other Common Parameters

- **Startup Delay**: Measured in seconds. Specifies the duration after the container starts before probing begins. For example, if the startup delay is set to 5, health checks will begin 5 seconds after the container starts.
- **Interval Time**: Measured in seconds. Specifies the frequency of health checks. For example, if the interval time is set to 10, the cluster will perform a check every 10 seconds.
- **Response Timeout**: Unit in seconds. Specifies the timeout duration for health probes. For TCP port probing, HTTP request probing, and command execution checks, this represents the TCP connection timeout, HTTP request response timeout, and command execution timeout, respectively.
- **Healthy Threshold**: Measured in occurrences. Specifies how many consecutive successful health checks are required before a container is considered healthy. For example, if the healthy threshold is set to 3, it means that the container is considered healthy only if it passes 3 consecutive probes successfully.

> ⚠ **Note**
> If the type of health check is a survival check, then the healthy threshold can only be 1, and other values you set will be considered invalid.

- **Unhealthy Threshold**: Measured in occurrences. Specifies how many consecutive health check failures are required before a container is considered unhealthy. For example, if the unhealthy threshold is set to 3, it means that the container is considered unhealthy only after failing the probe 3 consecutive times.

Setting the Run Command and Parameter for a Workload

Last updated: 2023-09-26 14:20:01

Overview

When creating a workload, you typically specify the process to run in the container instances through an image. By default, the image runs the default command. If you need to run a specific command or override the image's default values, you will need to use the following three settings:

- Working Directory (workingDir): Specifies the current working directory.
- Run Command (command): Controls the actual command executed by the image.
- Command Arguments (args): Parameters passed to the run command.

Working Directory Description

WorkingDir specifies the current working directory. If it does not exist, it will be automatically created. If not specified, the default value for the container runtime is used. If WORKDIR is not specified in the image and not specified in the console, the default workingDir is "/".

Command and Parameter Usage

To adapt the docker run command to Tencent Cloud Container Service, please refer to docker run parameter adaptation.
Docker images have metadata for storing image information. If no run command and parameters are provided, the container will run the default command and parameters provided during image creation. The native Docker fields are "Entrypoint" and "CMD". For more information, see Docker's Entrypoint documentation and CMD documentation.
If you provide a run command and parameters for the container when creating a service, the container service will override the default command set during image creation (i.e., "Entrypoint" and "CMD"). 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]
[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 in the container service console, and Docker run's CMD parameter corresponds to the Run Arguments in the container service console. When there are multiple run arguments, you need to enter them in the container service's Run Arguments section, with each argument on a separate line.
- For an example of setting container run commands and parameters through the TKE console, please refer to Command and Args.

Auto Scaling
HPA Metrics

Last updated: 2023-09-26 18:10:10

Horizontal Pod Autoscaler (HPA) can automatically adjust 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 through YAML files. This document describes how to configure a YAML file.

Auto Scaling Metrics

The following tables list the details of the auto scaling 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

Description in the Console	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 utilization (per Limit)	%	Ratio of the total number of CPU cores used by the Pod and the sum of Limit specified by the container in the Pod	Pods	k8s_pod_rate_cpu_core_used_limit	%

Disk metrics

Description in the Console	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	Reads/second	Number of I/O times Pod reads data from disk	Pods	k8s_pod_fs_read_times	Reads/second
Disk write IOPS	Reads/second	Number of I/O times Pod writes data to disk	Pods	k8s_pod_fs_write_times	Reads/second

Networking

Description in the Console	in the Console	Remarks	type	metricName	Default Unit

Inbound Network Bandwidth	Mbps	Sum of inbound bandwidth of all containers per Pod	Pods	k8s_pod_network_receive_bytes_bw	Bps
Outbound Network Bandwidth	Mbps	Sum of outbound bandwidth of all containers per Pod	Pods	k8s_pod_network_transmit_bytes_bw	Bps
Inbound Network Traffic	KB	Sum of inbound traffic of all containers per Pod	Pods	k8s_pod_network_receive_bytes	B
Outbound Network Traffic	KB	Sum of outbound traffic of all containers per Pod	Pods	k8s_pod_network_transmit_bytes	B
Network Packets In	/second	Sum of inbound packets of all containers per Pod	Pods	k8s_pod_network_receive_packets	/second
Network Packets Out	/second	Sum of outbound packets of all containers per Pod	Pods	k8s_pod_network_transmit_packets	/second

Memory

Description in the Console	in the Console	Remarks	type	metricName	Default Unit
MEM 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 used by the Pod	Pods	k8s_pod_rate_mem_usage_node	%
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 the total amount of memory specified by Limit that is used by the Pod	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.

Description in the Console	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 HPA Through YAML

You can create and edit an HPA through a YAML file. Below is a sample configuration file that defines an HPA named `example`, which will be triggered when the CPU usage reaches `1`. The instance range is 1-2.

> **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  # Supports using Resource
    pods:
      metricName: k8s_pod_cpu_core_used
      targetAverageValue: "1"
  scaleTargetRef:
    apiVersion: apps/v1beta2
    kind: Deployment
    name: nginx
```

Configuration
ConfigMap Management

Last updated: 2023-09-26 18:10:44

Feature Overview

ConfigMap allows you to decouple configurations from images to make the application more portable. ConfigMap is a key-value pair. You can create a ConfigMap object by using kubectl in the console, and use a ConfigMap by mounting a volume, through environment variables, or by running commands in the container.

Via the console

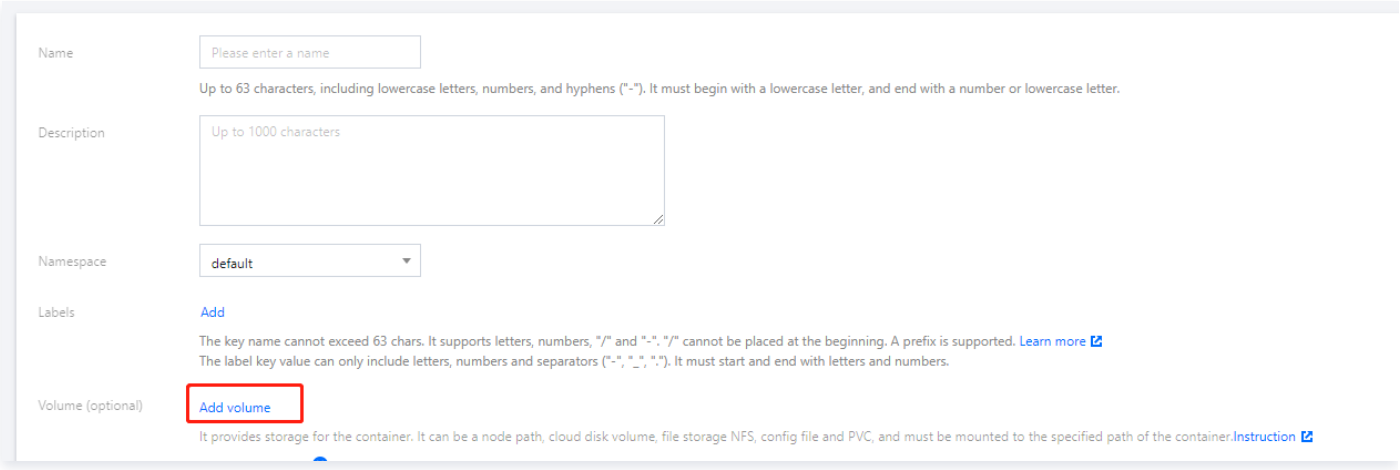
Creating a ConfigMap

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 enter the "New 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 as needed.
   - Content: Add the variable name and variable value.
7. Click **Create ConfigMap**.

Using ConfigMap

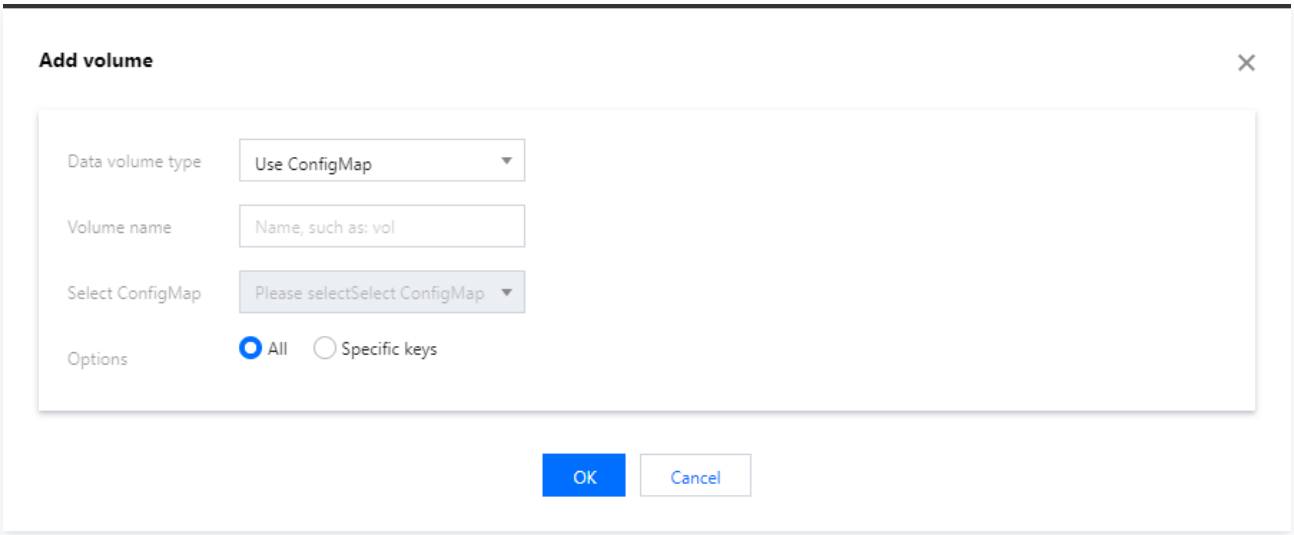
Method 1: Using ConfigMap as a Volume Type

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 enter its information page. For example, choose **Workload** > **DaemonSet** to access the DaemonSet information page.
5. Click **Create** to enter the "Create DaemonSet" page.
6. Based on the information provided, set the workload name, namespace, and other details. In the "Volumes" section, click **Add Volume**. Refer to the figure below:



7. In the "Add Volume" dialog box, configure the mount point according to the following information and click **Confirm**. As shown below:

Select "Use ConfigMap" method, enter the name, and click **Select Configuration Item**. As shown below:

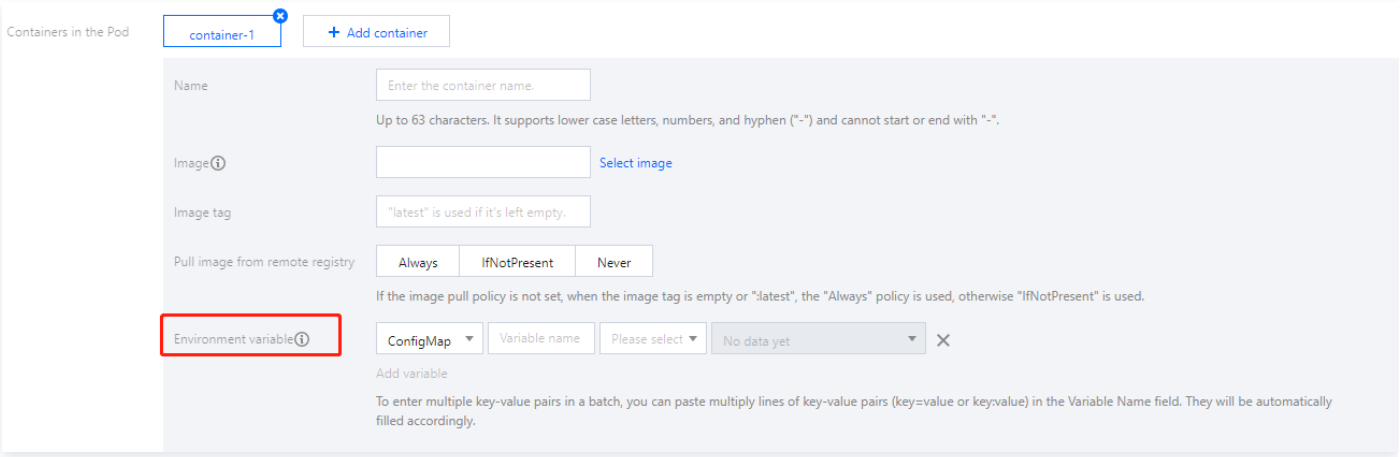


- **Volume Type**: Select "Use ConfigMap" method.
- **Volume name**: Enter a custom name.
- **Select ConfigMap**: Select as needed.
- **Options**: Offers two choices – "All" and "Specify some keys".
- **Items**: When selecting the "Specify certain keys" option, you can add an item to mount to a specific path. For example, if the mount point is /data/config and the file name is filename, the value of the key–value pair will be stored in /data/config/filename.

8. Click **OK**. Click **Create Workload**.

Method 2: Using ConfigMap type in environment variables

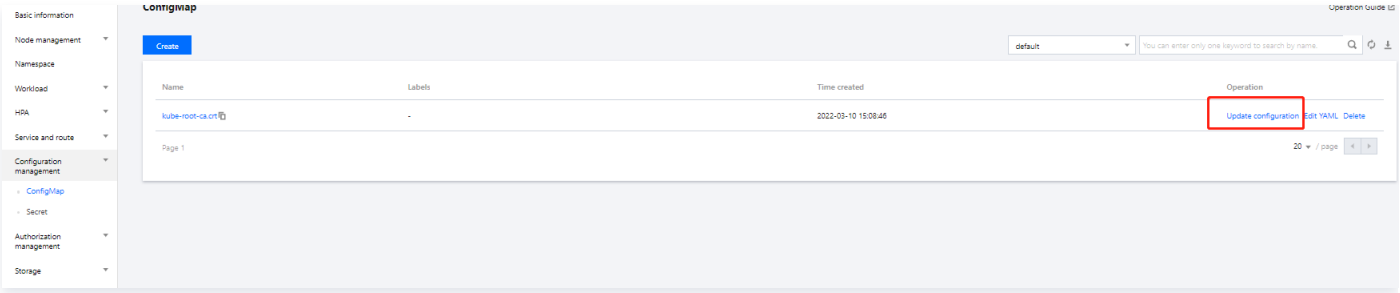
1. Log in to the TKE console.
2. In the left sidebar, click **Cluster** to open the TKE cluster list page.
3. Click the ID of the cluster where Workload needs to be deployed to go to the cluster management page.
4. Under "Workload", select any Workload type to enter its information page. For example, choose **Workload** > **DaemonSet** to access the DaemonSet information page.
5. Click **Create** to enter the "Create DaemonSet" page.
6. Based on the information on the page, set the workload name, namespace, and other details. In the "Containers in the Pod" section, under "Environment Variables," click **Add Variable**. Refer to the figure below:



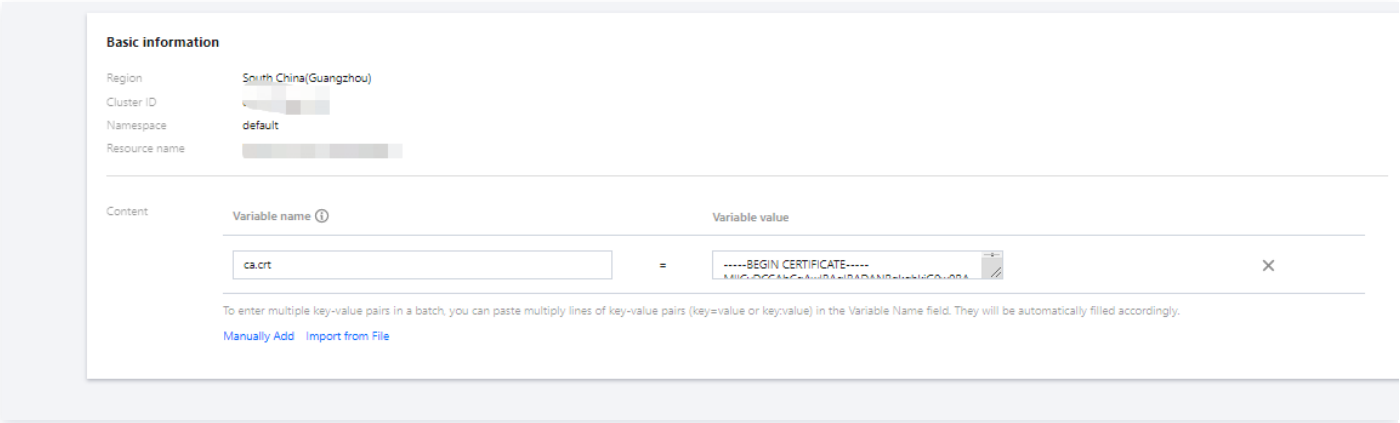
7. Select **ConfigMap** for the environment variable and select the resources as needed.
8. Click **Create Workload**.

Updating 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. In the row of the ConfigMap you want to update, click **Update Configuration** on the right to go to the Update ConfigMap page.



6. On the "Update Configuration" page, edit the key–value pairs and click **Update ConfigMap**.



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: Create using a sample YAML 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
```

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: Create by executing commands

Run the following command to create the ConfigMap in the directory.

```
kubectl create configmap <map-name> <data-source>
```

- <map-name>: Represents the name of the ConfigMap.
- <data-source>: Represents a directory, file, or literal value.

For more information about the parameters, see Kubernetes' official document about ConfigMap.

Using ConfigMap

Method 1: Using ConfigMap as a Volume Type

Below are the sample YAML files:

```
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 specified ConfigMap Key to mount
         ##   key: key1  ## Select specified Key
         ##   path: keys ## Mount to the specified sub-path
   restartPolicy: Never
```

Method 2: Using ConfigMap type in environment variables

Below are the sample YAML files:

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:latest
      env:
        - name: key1
          valueFrom:
            configMapKeyRef:
              name: test-config ## Set the source ConfigMap file name
              key: test-config.key1  ## Set the source item for this environment variable's value
  restartPolicy: Never
```

Service Management Overview

Last updated: 2023-09-26 14:23:54

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	Note
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	Expose the service through the IP and static port (NodePort) on each cluster node. NodePort services route to ClusterIP services, which are automatically created. By requesting `<NodeIP>:<NodePort>`, the NodePort service can be accessed from outside the cluster. It is not recommended to provide services directly through cluster nodes or even the public network in production environments, except for testing and non-production environments. From a security perspective, using this type exposes cluster nodes directly, making them vulnerable to attacks. Cluster nodes are generally considered dynamic and scalable, and using this type creates coupling between the service's external address and the cluster nodes.
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. LoadBalancer Services are 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	Note
Public network forwarding cluster	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 Tencent Cloud Load Balancer (CLB) instances was upgraded on March 6, 2023. After the upgrade, public network CLB instances provide services through **domain names**. The **VIP** dynamically changes with business requests, and the console no longer displays VIP addresses. Please refer to the Announcement of Domain Name-Based Public Network CLB Launch. • For CLB users registered after the upgrade, the domain name-based CLB architecture is adopted by default. • Existing users can choose to continue using their current load balancing services without being affected by the upgrade. If you need to upgrade your load balancing service, you must upgrade both Tencent Cloud CLB and TKE; otherwise, the synchronization of all public network Service/Ingress types in TKE may be affected. For details on CLB upgrade operations, see Domain Name-Based Load Balancer Upgrade Guide. To upgrade the Service/Ingress component version in TKE, please contact us through Online Consultation.

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

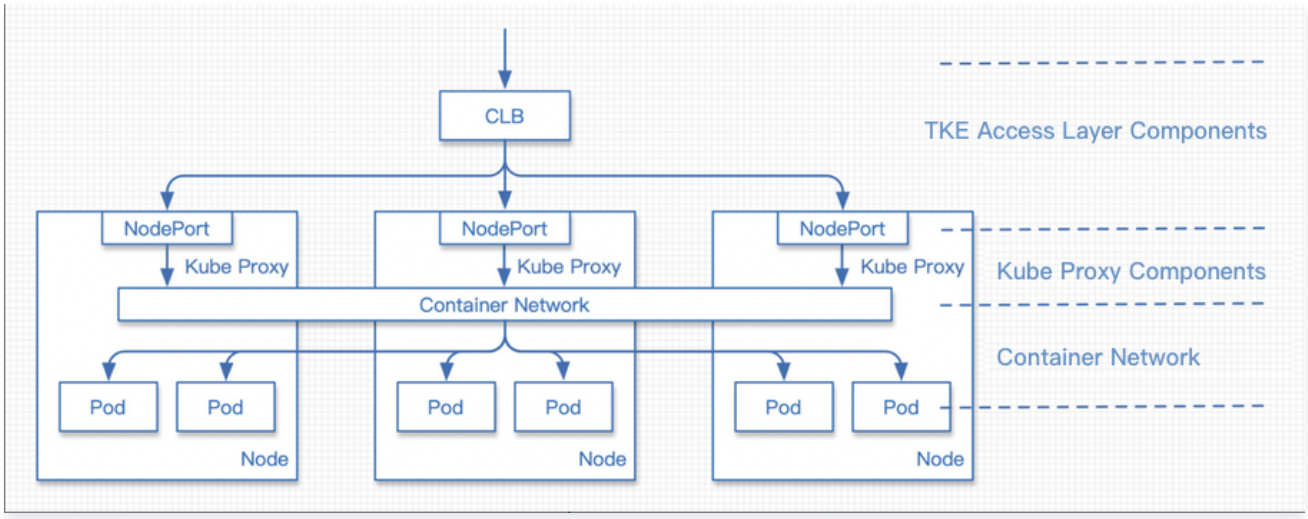
The external service capabilities of a Service rely on the resources provided by the CLB instance. Service resource management matters to a Service, which will use the following labels during the resource lifecycle management:

- `tke-createdBy-flag = yes` : Indicates that the resource is created by TKE.
- `tke-clusterId = <ClusterId>` : Identifies the cluster that uses the resource.

> **Note**
> - The above tags are read-only for users. Please do not modify or delete them, as doing so may result in resource leakage.
> - 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 load balancer, the load balancer will not be deleted when the Service is removed.

When a LoadBalancer type Service resource is created in the cluster, the corresponding load balancer's lifecycle begins. The load balancer's lifecycle ends when the Service resource is deleted or the load balancer is rebuilt. During this period, the load balancer continuously synchronizes with the Service resource description. **When users switch the network access of the Service, such as from public network to Cluster IP, VPC private network to Cluster IP, or Cluster IP to an existing load balancer, these operations involve the deletion or termination of the load balancer.** The working principle of the LoadBalancer type Service is illustrated in the

following diagram:



Service precautions

Service has a field: `.spec.externalTrafficPolicy`. kube-proxy filters the target service endpoints based on the `spec.internalTrafficPolicy` setting. When its value is set to `Local`, only the local service endpoints on the node are selected. When its value is set to `Cluster` or left as default, Kubernetes selects all service endpoints. For more information, see Kubernetes documentation.
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 of Service
- Service Load Balancing Configuration
- Using Existing CLBs
- Service Backend Selection
- Service Cross-Domain Binding
- Graceful Shutdown of Service
- Using Services with CLB-to-Pod Direct Access Mode
- Multiple Services Sharing a CLB
- Service Extension Protocol
- Service Annotation Description

References

For more information, see the open-source document Kubernetes Service.

Basic Features

Last updated: 2023-09-26 18:12:09

Console Operation Guide

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**. On the **Service** page, click **Create**. As shown below:

4. On the **Create Service** page, set the Service parameters according to your actual needs. Key parameters are as follows:
   - **Service Name**: Customize a name.
   - **Namespace**: select a namespace based on your actual requirements.
   - **Access Settings**: Set it as needed and as instructed in Service Access Methods.
   - (Optional) **Advanced Settings**:
     - **External TrafficPolicy**:
       - **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**: To ensure that connections from a specific client are always directed to the same Pod, you can set session affinity based on the client IP address by configuring the Service's `.spec.sessionAffinity` to ClientIP (default 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.

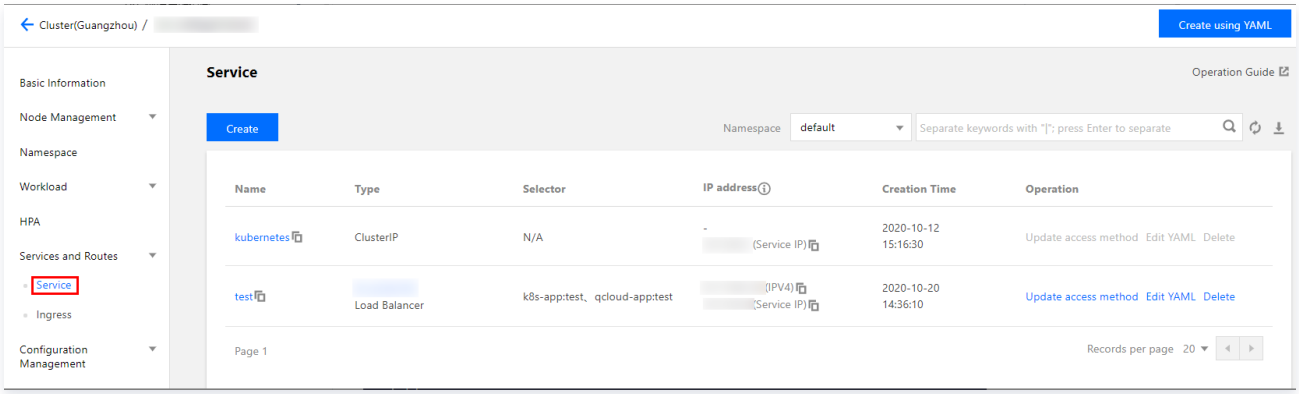
5. Click **Create Service**.

Updating a Service

> **Note**
> To prevent resource anomalies when a Service switches between different CLBs, such as CLB detaching from TKE control causing resource leaks or failing to create the corresponding CLB resulting in service interruptions, TKE has imposed the following restrictions on the lifecycle changes of a Service:
> 1. Disallow switching between Public Network LB Access and Private Network LB Access in the service access method.
> 2. When selecting public LB access, switching the current VPC's availability zone and other VPCs is prohibited; switching between automatically creating and using existing load balancers in the load balancer is also not allowed.
> 3. When selecting private network LB access, switching between automatically creating and using existing load balancers is prohibited, as well as changing the subnet where the LB is located.
> 4. When selecting an existing CLB, switching to another existing CLB is not allowed.
> 5. Prohibit switching between Classic CLB and Application CLB.
>
> If you attempt to switch the above actions, the frontend will impose restrictions, and the backend will intercept. If you have a need for the above switches, you can try:
> 1. First, switch the service access method to either Cluster-Internal Access or Host Port Access.
> 2. Reconfigure to the target parameter settings.
>
> For more information, see Announcement.

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. As shown below:



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. As shown below:

Managing a Service Using kubectl

YAML sample

```
kind: Service
apiVersion: v1
metadata:
  ## annotations:
  service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx ## If you are creating a Service for private network access, you need to specify this annotation.
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

Storage fees description

- **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, users can create up to 100 private or public network CLB instances. If you require more than 100 instances, you can contact support to increase the load balancing CLB 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 Job is successfully created.

```
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: 2023-09-26 14:24:09

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 Notes

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 directly configure and modify protocols and ports. You need to describe the protocol and port in the configuration to specify the listener for the configuration. Multiple listener configurations can be declared in a single TkeServiceConfig, mainly focusing on CLB health checks and backend access configurations.
By specifying the protocol and port, the configuration can 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. When creating a LoadBalancer-mode Service, set the annotation `service.cloud.tencent.com/tke-service-config-auto: "true"` to automatically create <ServiceName>-auto-service-config. You can also directly specify your own TkeServiceConfig by using **service.cloud.tencent.com/tke-service-config:<config-name>**. Both annotations cannot be used simultaneously.
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 configuration.
    - protocol: TCP # Layer-4 rule for protocol ports anchoring the Service. Required. Enumerated value: TCP|UDP.
      port: 80 # Required. Value range: 1-65535.
      deregisterTargetRst: true # Optional, boolean. Two-way RST switch.
      session: # Configuration related to session persistence. Optional.
        enable: true # Indicates whether to enable session persistence. Required. Boolean.
        sessionExpireTime: 100 # Session persistence duration. Optional. Default value: 30. Value range: 30-3600. Unit: second.
      healthCheck: # Configuration related to health check. Optional.
        enable: true # Indicates whether to enable session persistence. Required. Boolean.
        intervalTime: 10 # Health check probe interval. Optional. Default value: 5. Value range: 5-300. Unit: second.
        healthNum: 2 # Healthy threshold, indicating the number of consecutive healthy health check results that it takes to indicate normal forwarding. Optional. Default value: 3. Value range: 2-10. Unit: times.
        unHealthNum: 3 # Unhealthy threshold, indicating the number of consecutive unhealthy health check results that it takes to indicate a forwarding exception. Optional. Default value: 3. Value range: 2-10. Unit: times.
        timeout: 10 # Health check response timeout threshold. This should be less than the health check interval. Optional. Default value: 2. Value range: 2-60. Unit: second.
      scheduler: WRR # Request forwarding method configuration. WRR and LEAST_CONN represent weighted round robin and least connections, respectively. Optional, enumerated values: WRR | LEAST_CONN.
```

Sample

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
NAME    TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)                 AGE
jetty   LoadBalancer   10.127.255.209   150.158.220.237   80:31338/TCP,443:32373/TCP   2m47s

# 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: 2023-09-26 14:24:16

Tencent Kubernetes Engine (TKE) supports using existing Cloud Load Balancer (CLB) instances through the annotation `service.kubernetes.io/tke-existed-lbid: <LoadBalanceId>`. You can use this annotation to specify the CLB instance associated with the cluster Service resource. Additionally, TKE offers the **Service Load Balancer Reuse** feature, allowing multiple Services to use the same existing CLB. You can refer to this document for configuration guidance.

Synchronization behavior when using an existing load balancer

- When using an existing load balancer, the annotation specifying the Service's network type will not take effect.
- When a Service no longer uses an existing load balancer, the corresponding listener described in the Service will be deleted, but the load balancer will be retained. During the deletion of the listener, the listener name will be checked for modifications. If the user has modified the listener name, it is assumed that the listener may have been created by the user, and it will not be deleted automatically.
- If a Service is currently using an automatically created load balancer, adding an annotation to use an existing load balancer will terminate and release the current load balancer's lifecycle, and the Service configuration will be synchronized with the existing load balancer. Conversely, if the annotation for the existing load balancer being used by the Service is removed, the Service Controller component will create a new load balancer for the Service and synchronize it accordingly.

Synchronization behavior of Tencent Cloud tags when using an existing load balancer

- By default, the CLB created by the Service will be configured with the `tke-createdBy-flag = yes` label, and the Service will delete the corresponding resources upon termination. If an existing CLB is used, this label will not be configured, and the corresponding resources will not be deleted when the Service is terminated.
- All Services will be configured with the `tke-clusterId =` label. If the ClusterId is correct, the corresponding label will be deleted when the Service is terminated.
- For clusters created after August 17, 2020, the feature of multiple Services sharing the same CLB is disabled by default. For changes in the CLB label configuration rules and detailed information regarding Services created within clusters before and after this date, please refer to Multiple Services Sharing a CLB.

Supports and Limits

- The specified load balancer must be in the same VPC as the cluster.
- Please ensure that your container-based services do not share a load balancer with your Cloud Virtual Machine (CVM) services.
- Modifications made to the listeners and backend server bindings of a load balancer managed by TKE through the load balancer console are not supported, as your changes will be overwritten by TKE's automatic synchronization.
- When using an existing load balancer:
  - The Service Controller will not be responsible for the release and recovery of the existing load balancer.
  - Only load balancers created through the Cloud Load Balancer console are supported for reuse. Reusing load balancers automatically created by TKE is not supported, as it may disrupt the lifecycle management of other Service load balancers.
- When **reusing** a load balancer:
  - Cross-cluster load balancer reuse is not supported.
  - When utilizing the **reuse** feature, it is recommended to have a clear listener port management strategy to avoid confusion in managing the load balancer when it is used by multiple Services.
  - When port conflicts occur while reusing a load balancer, the operation will be rejected. If conflicts arise during modifications, the synchronization of the conflicting listener's backend cannot be guaranteed to be accurate.
  - Services reusing a load balancer do not support enabling Local access (due to classic load balancer limitations).
  - Upon deleting a Service, the real server bound to the reused load balancer will need to be unbound manually. A tag `tke-clusterId: cls-xxxx` will be retained, and it must be cleared manually.

Sample Service

```
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` annotation indicates that this Service will be configured using an existing load balancer.
- Please note that the Service type should be set to `LoadBalancer` type.

Use Cases Example

Using a monthly or yearly subscription load balancer to provide external services

The Service Controller component only supports managing the lifecycle of pay-as-you-go load balancer resources. When users require long-term use of load balancers, the monthly subscription billing mode offers certain price advantages. In such scenarios, users can independently purchase and manage load balancers, then control the Service to use existing load balancers through annotations, effectively separating the load balancer lifecycle management from the Service Controller component.

Exposing TCP and UDP services on the same port

In the official Kubernetes Service design, there is a limitation: all exposed ports under a Service must use the same protocol. Many gaming scenarios require users to expose both TCP and UDP services on the same port simultaneously. Tencent Cloud Load Balancer supports listening to both UDP and TCP protocols on the same port, and this requirement can be addressed through Service Load Balancer Reuse.

For example, in the following Service configuration, `game-service` is described as two Service resources with nearly identical content, except for the listening protocol. Both Services use the annotation to specify the existing load balancer `lb-6swtxxxx`. By applying these resources to the cluster using kubectl, you can achieve the goal of exposing multiple protocols on the same load balancer 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:
```

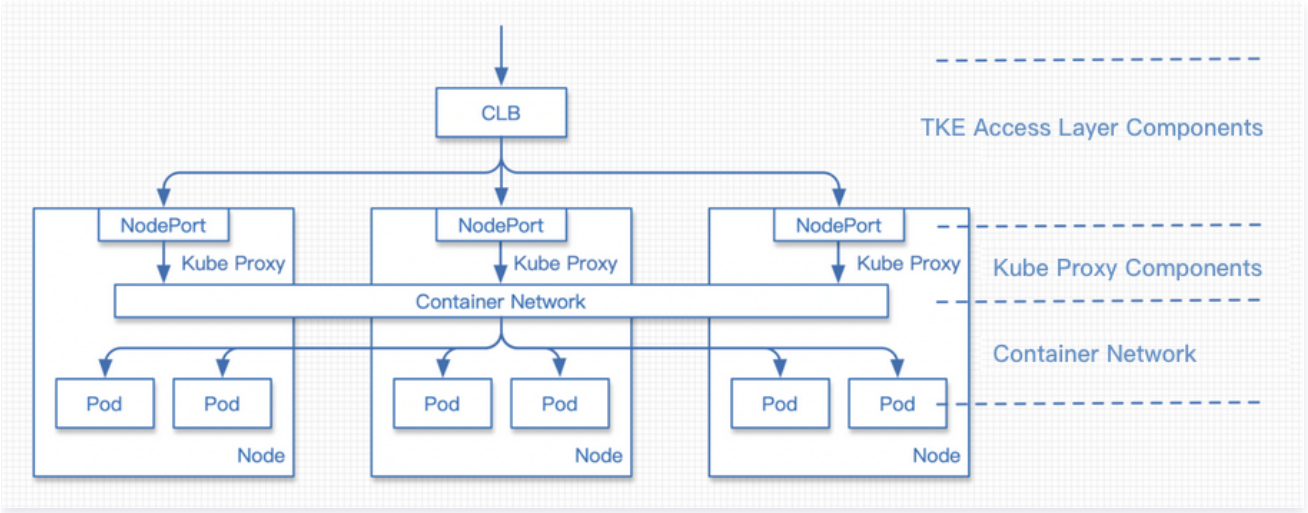
```
  annotations:
    service.kubernetes.io/tke-existed-lbid: lb-6swtxxxx
  name: game-service-b
spec:
  ports:
    - name: 80-80-udp
      port: 80
      protocol: UDP
      targetPort: 80
  selector:
    app: game
  type: LoadBalancer
```

Service Backend Selection

Last updated: 2023-09-26 14:24:23

Selecting Default Backend

By default, the Service configures the load balancer's backend to the cluster node's NodePort, as shown in the TKE Access Layer Components section below. This solution has very high fault tolerance; after traffic passes through the load balancer to any NodePort, the NodePort will randomly select a Pod to forward the traffic. This is also the most basic network access layer solution proposed by Kubernetes. As shown in the following diagram:



`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. Unschedulable nodes can also serve as traffic entry points because, after traffic enters the node, another layer of container network traffic forwarding is performed. Traffic within Unschedulable nodes will not be discarded, as shown in the diagram 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.

This issue can be resolved in the following ways:
In large-scale cluster scenarios, users can use the `service.kubernetes.io/qcloud-loadbalancer-backends-label` annotation to specify a subset of nodes for binding. The content of `service.kubernetes.io/qcloud-loadbalancer-backends-label` is a label selector. Users can mark labels on cluster nodes and then select matching nodes for binding through the label selector described in the Service. This synchronization will continue; when node changes cause them to be selected or no longer selected, the Service Controller will add or remove the corresponding backend on the load balancer accordingly. For more information, please refer to Kubernetes Labels and Selectors.

Supports and Limits

- 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 example contains the following configuration:

- Describes a service exposure for public network CLB instances.
- `service.kubernetes.io/qcloud-loadbalancer-backends-label` annotation specifies the backend selector, and only nodes with the `group=access-layer` Label in the cluster will be used as the backend for this load balancer.

Service Local Mode

Kubernetes offers the Service feature `ExternalTrafficPolicy`. When `ExternalTrafficPolicy` is set to Local, it avoids traffic forwarding between nodes through NAT, reducing NAT operations and preserving the source IP. NodePort will only forward traffic to Pods on the current node. The characteristics of Local mode are as follows:

Strengths:

- Avoids the performance loss caused by inter-node forwarding through NAT Gateway.
- Retains the request source IP for the server.

Shortcomings:

NodePort cannot serve nodes without a workload.

Supports and Limits

- Since Service Local mode inherently has node selection capabilities:
  - Please avoid using the capability to specify access layer backends in Service Local mode.
  - Please avoid using the `node.kubernetes.io/exclude-from-external-load-balancers` annotation to evict backend nodes on the Node when using Service Local mode.
- 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 incoming traffic on a node is not forwarded between nodes. Therefore, when the number of workloads on nodes is inconsistent, the same backend weight may result in uneven load distribution across nodes. In this case, users can use the `service.cloud.tencent.com/local-svc-weighted-balance: "true"` annotation for weighted balancing. When this annotation is used, the weight of the NodePort backend is determined by the number of workloads on the node, thus avoiding load imbalance issues caused by different workload quantities on different nodes. Note that **Local weighted balancing must be used in conjunction with Local binding.** An example is 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: 2023-09-26 14:24:31

Feature 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.

Scenarios

- 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

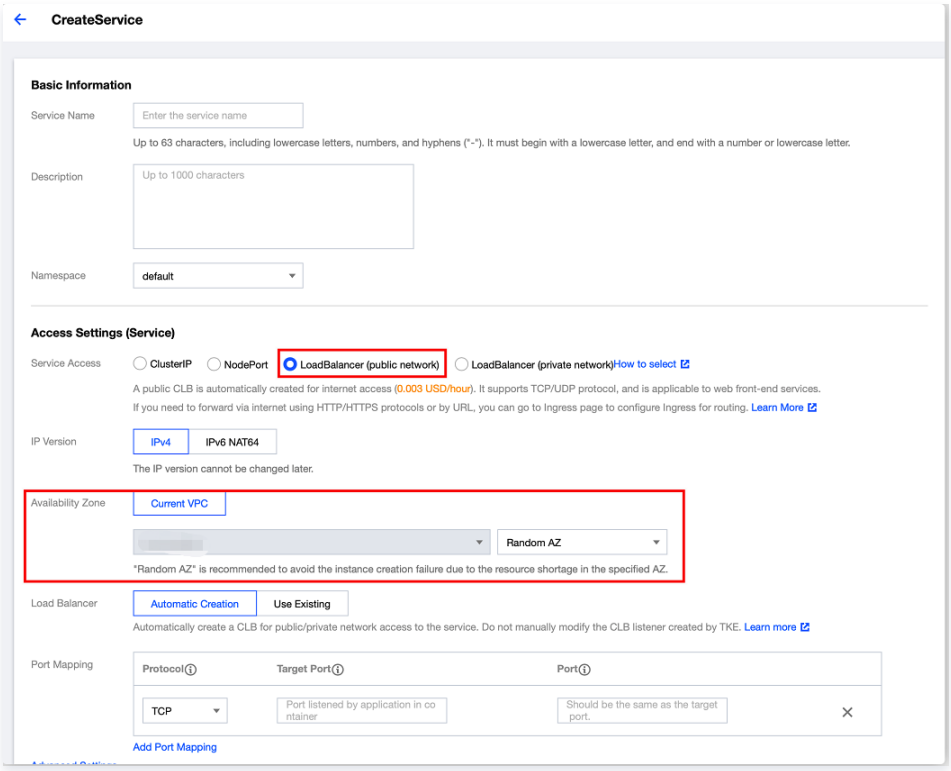
1. Cross-region binding is only available for bill-by-IP accounts. To check your account type, see Checking Account Type.
2. If you need to use the CLB that is not in the same VPC as this cluster, you need to connect the VPCs of the current cluster and the CLB via CCN.
   2.1 The IP address ranges of VPCs in different regions must be planned in advance for CCN, and conflicts should be avoided. Otherwise, conflicting routing rules will not take effect, resulting in the inability to forward data.
   2.2 The VPC where the cluster resides cannot join multiple CCNs simultaneously, as this may result in non-unique routes and prevent data plane forwarding.
3. After ensuring that the VPCs are connected, please contact us online 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.

Instructions

You can bind and specify availability zones for CLB Service across regions via the console and YAML.

Using the Console

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 cluster for which you need to create a Service to go to the cluster management page.
3. Select **Services and Routes** > **Service** to access the Service page, and click **Create**.
4. Configure the relevant availability zone rules in the **Create Service** page. The configuration rules are as follows:
   - **Service Access Method**: Select "Public Network CLB Access".



- **Availability Zone**: Select "Other VPC".

> **Note:**
> Only supports other VPCs connected to the current cluster's VPC through Cloud Connect Network (CCN).

5. Other parameters can be set according to the configuration during Service creation.

YAML Method

> **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 ensuring that the VPCs are connected, please contact us online to apply for this feature.

Example 1

If you only need to specify the availability zone of the VPC where the cluster resides, for example, if the VPC of the cluster is located in Guangzhou, and you need to specify the CLB of Guangzhou Zone 1 for CLB Service, you can add the following annotations to the YAML of the Service:

```
service.kubernetes.io/service.extensiveParameters: '{"ZoneId":"ap-guangzhou-1"}'
```

Example 2

If you need to use a CLB that is not in the VPC of the cluster, you can add the following annotations to the YAML of the Service:

```
service.cloud.tencent.com/cross-region-id: "ap-guangzhou"
service.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"
```

> **Note**
> If you need to specify the availability zone, you also need to add the annotations of sample 1.

Example 3

Select an existing load balancer for remote access, as shown below:

```
service.cloud.tencent.com/cross-region-id: "ap-guangzhou"
service.kubernetes.io/tke-existed-lbid: "lb-342wppll"
```

Example 4

The annotation in the service YAML is as follows:

```
Creating Cross-Region Access Load Balancer
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/cross-region-id: "ap-chongqing"
    service.cloud.tencent.com/cross-vpc-id: "vpc-mjekzyps"
  name: echo-server-service
  namespace: default
spec:
  ......
---
# Scenarios for users reusing load balancers from other regions
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/cross-region-id: "ap-chongqing"
    service.kubernetes.io/tke-existed-lbid: "lb-o8ugf2wb"
  name: echo-server-service
  namespace: default
spec:
  ......
```

For detailed Service annotations, please see Service Annotation.

Graceful Service Shutdown

Last updated: 2023-09-26 14:24:38

Feature Overview

In scenarios where Pods are directly connected to the access layer, when the backend performs a rolling update or a backend Pod is deleted, removing the Pod from the load balancer's backend directly would result in unprocessed requests that the Pod has already received. This is particularly relevant in long-lived connection scenarios, such as conference services, where updating or deleting a workload's Pod would cause the conference to be interrupted immediately.

Scenarios

- 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. Please check whether your cluster supports direct access.

Instructions

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 access mode
    service.cloud.tencent.com/enable-grace-shutdown: "true"  # Indicates the use of graceful 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 achieve graceful termination, it is essential to handle the SIGTERM signal in your application code. The main logic is to stop accepting new traffic, continue processing existing traffic, and exit only when all connections are closed. For more information, see Example.

   If your application code does not handle the SIGTERM signal, or you cannot control the third-party libraries or systems to add graceful termination logic, you can also try configuring a preStop for the Pod to implement graceful termination logic. Here's an example:

```
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 configuring preStop, please refer to the Kubernetes API documentation.
   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. **Adjust the graceful duration using terminationGracePeriodSeconds**
   If a longer graceful termination time is required (preStop + business process stop may exceed 30s), you can customize the terminationGracePeriodSeconds according to the actual situation to avoid being stopped by SIGKILL prematurely. Here's an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: grace-demo
```

```
spec:
  terminationGracePeriodSeconds: 60 # Graceful shutdown defaults to 30s, but you can set a longer duration.
  containers:
  - name: lifecycle-demo-container
    image: nginx
    lifecycle:
      preStop:
        exec:
          command:
          - sleep
          - 5s
...
```

Capabilities

Graceful shutdown only sets the weight of the CLB backend to 0 when the Pod is deleted. If a Pod becomes unhealthy during its operation, setting the weight of the backend to 0 can reduce the risk of service unavailability.

You can use the Annotation: `service.cloud.tencent.com/enable-grace-shutdown-tkex: "true"` to enable this graceful exit capability.

This Annotation will set the weight of the not-ready CLB backend to 0 based on whether the endpoints in the Endpoint object are not-ready.

Documentation

Troubleshooting: No Graceful Unbinding on the Backend of NGINX Ingress Controller

Using Services with CLB-to-Pod Direct Access Mode

Last updated: 2023-09-26 14:24:47

Scenario

Native LoadBalancer mode Service can automatically create a load balancer CLB and forward it to the cluster through the Nodeport, followed by a secondary forwarding via iptable or ipvs. This mode of Service can meet most use cases, but in the following scenarios, it is recommended to use **Direct-to-Pod mode Service**:

- 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).

Note

- If your cluster is a Serverless cluster, it defaults to the Direct-to-Pod access mode, and you don't need to do anything.
- Both GlobalRouter and VPC-CNI container network modes currently support Direct-to-Pod mode. You can click on the cluster ID in the Cluster List to access the cluster details page, and view the network plugin used by the current cluster in the "Basic Information" page.

VPC-CNI Mode

Usage Limits

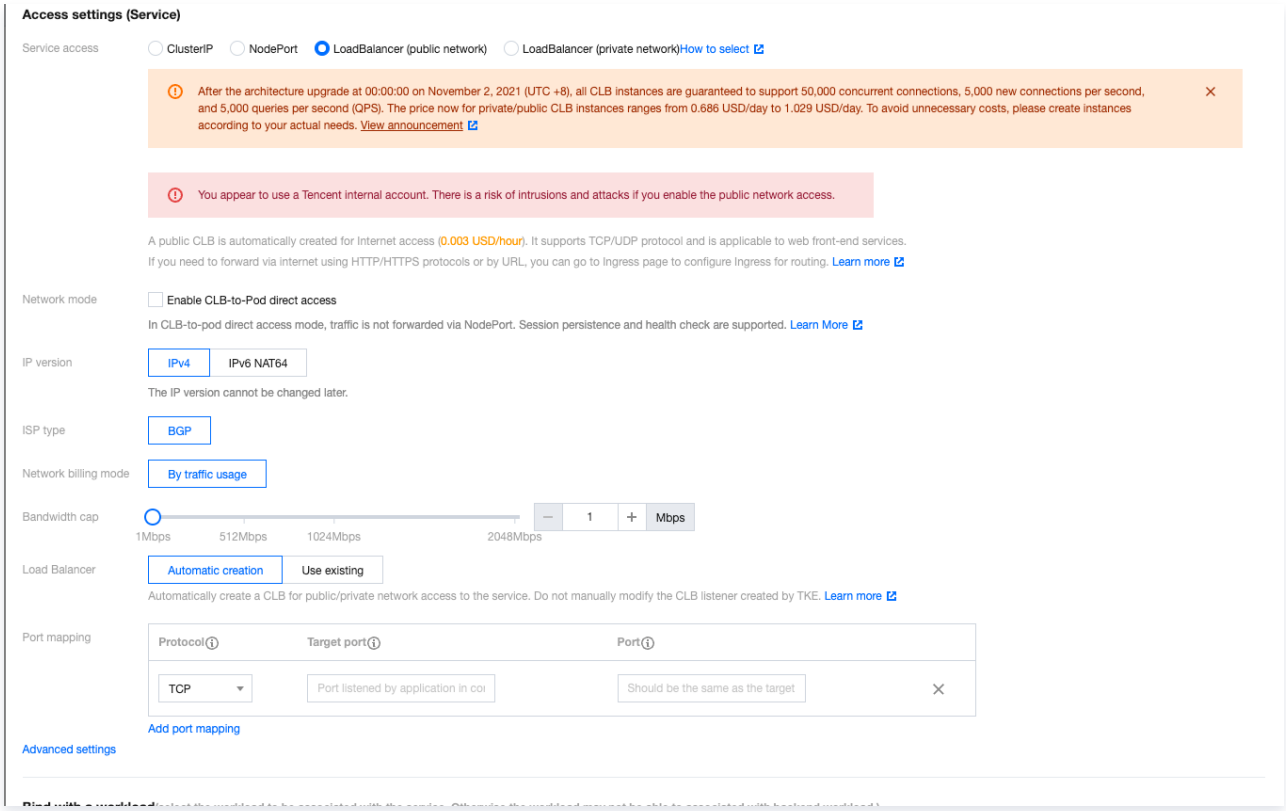
- The Kubernetes version of the cluster must be 1.12 or later.
- The VPC-CNI ENI mode must be enabled for the cluster network mode.
- The workloads used by a service in direct access mode must adopt the VPC-CNI ENI mode.
- The default limit for backend instances in a CLB is 200. If the number of workload replicas you need to bind exceeds 200, you can contact support online to increase the CLB 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.

Instructions

Console Operation Guide

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**.

YAML Operation Guide

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 Direct-to-Pod mode
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  type: LoadBalancer
```

Annotation extension

For the CLB configuration, see Service CLB Configuration. The annotation configuration is as follows:

```
service.cloud.tencent.com/tke-service-config: [tke-service-configName]
```

Supports and Limits

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 initiate a rolling update for their applications, Kubernetes performs the update according to the update strategy. However, the criteria for determining a batch of Pods to start only includes the Pod's own status and does not consider whether the Pod has a health check configured and passed on the load balancer. In high-load scenarios, if such Pods cannot be scheduled promptly, the successfully updated Pods may not be providing services externally, leading to service interruptions.

To associate rolling updates with the backend status of the load balancer, TKE introduces the new feature `ReadinessGate` from Kubernetes 1.12. The TKE ingress component only configures the `ReadinessGate` status to make the Pod reach the Ready state when it confirms that the backend binding is successful and the health check has passed. This promotes the rolling update of the entire workload.

> **Note:**
> `ReadinessGate` is only responsible for traffic readiness checks and has a different role from `Endpoints`. Therefore, the health status of a Pod after it becomes ready is not reflected in the ReadinessGate Status.

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 maloperations. 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

Usage Limits

- A workload can only run in one network mode. You can choose VPC-CNI ENI mode or GlobalRoute mode for the workloads used by a service in direct access mode.
- It is only available for bill-by-IP accounts. If your current account is a traditional account type (non-upgraded bandwidth), please refer to Account Type Upgrade Instructions.
- The default limit for backend instances in a CLB is 200. If the number of workload replicas you need to bind exceeds 200, you can contact support online to increase the CLB 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 (readiness check) 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 Guide

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 Direct-to-Pod mode
  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: 2023-09-26 14:24:56

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, contact us through online consultation for application.
- **If your cluster is a TKE serverless cluster, CLB reuse is enabled by default, but you need to keep the following in mind:**
  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.

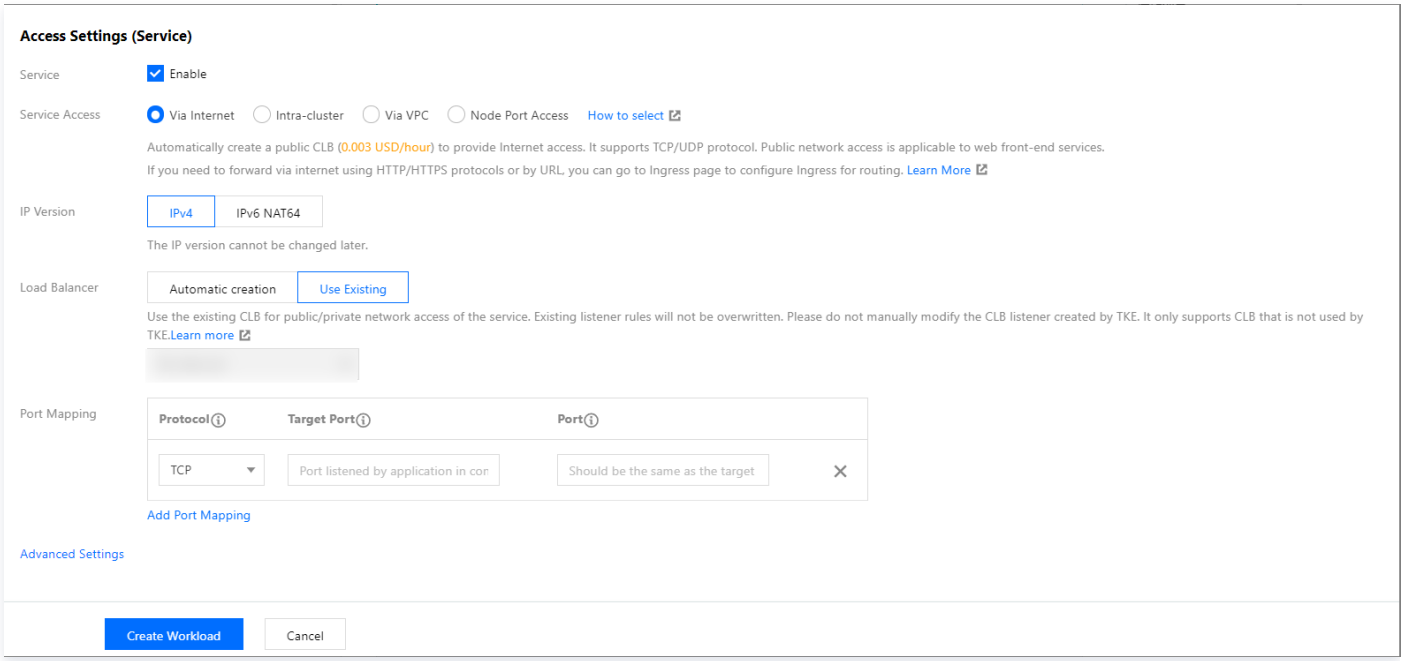
Usage Limits

- In the Service sharing scenario, the number of listeners managed by a single load balancer is limited by the CLB's TOTAL_LISTENER_QUOTA. For more information, please refer to the documentation.
- 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.

Instructions

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: 2023-09-26 14:25:04

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.

Note

- In the direct connection scenario, there are no restrictions on the use of extended protocols, and both TCP and UDP protocols can be employed concurrently.
- In non-direct access scenarios, ClusterIP and NodePort modes can be used together. However, for Services of the LoadBalancer type, the community currently only supports the use of protocols of the same type.
- When the LoadBalancer is declared as TCP, the port can utilize the extended protocol capabilities, changing the load balancing protocol to TCP_SSL, HTTP, or HTTPS.
- When the LoadBalancer is declared as UDP, the port can utilize the extended protocol capabilities to change the load balancing protocol to UDP.

TKE Extension of Service Forwarding Protocols

In native Service protocol support, there are scenarios where both TCP and UDP need to be supported simultaneously on a Service, and the Service must support TCP SSL, HTTP, and HTTPS protocols. TKE has extended support for more protocols in the 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

```
{"80":{"protocol":["TCP_SSL"],"tls":"cert-secret"}}
```

Sample for HTTP

```
{"80":{"protocol":["HTTP"],"hosts":{"a.tencent.com":{},"b.tencent.com":{}}}}
```

Sample for HTTPS

```
{"80":{"protocol":["HTTPS"],"hosts":{"a.tencent.com":{"tls":"cert-secret-a"},"b.tencent.com":{"tls":"cert-secret-b"}}}}
```

Sample for TCP/UDP

```
{"80":{"protocol":["TCP","UDP"]}} # Only supported in direct access mode. For more information, visit https://cloud.tencent.com/document/product/457/41897.
```

Sample for hybrid use

```
{"80":{"protocol":["TCP_SSL","UDP"],"tls":"cert-secret"}} # Only supported in direct access mode. For more information, visit https://cloud.tencent.com/document/product/457/41897.
```

QUIC

```
{"80":{"protocol":["QUIC"],"tls":"cert-secret"}}
```

Note

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.

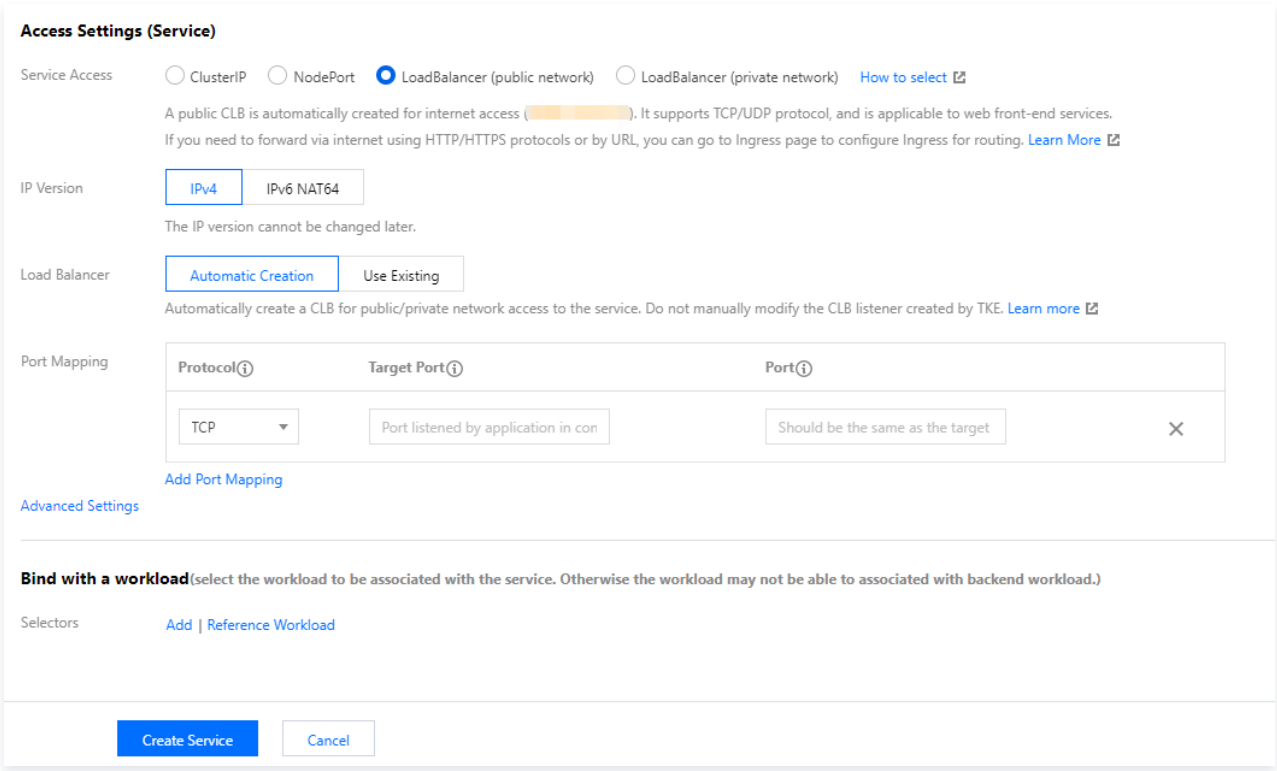
Extension protocol use instructions

Use instructions of extension protocol `YAML`

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["TCP_SSL"],"tls":"cert-secret"}}' # To use other protocols, change the value in the key-value pair to the above content
  name: test
    ....
```

Use instructions of extension protocols in the console

- When creating a Service, if it is exposed as "**Public Network LB**" or "**Private Network LB**", in non-direct connection mode scenarios, only TCP and TCP SSL can be used together in "Port Mapping". As shown in the following image:



- When the Service is in "**ClusterIP**" or "**NodePort**" mode, any protocols can be used together.
- Direct access mode supports the combination of any protocols.

Case Description

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 default to use the CLB-to-Pod direct access mode. For TKE clusters, you must enable the CLB-to-Pod direct access mode with reference to the document.
    service.cloud.tencent.com/specify-protocol: '{"80":{"protocol":["TCP"]},"8080":{"protocol":["UDP"]}}'  # It specifies that port 80 uses the TCP protocol, and port 8080 uses the UDP protocol.
  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 the limits of Kubernetes Service Controller.
    targetPort: 8080
  selector:
    k8s-app: nginx
    qcloud-app: nginx
```

```
  sessionAffinity: None
  type: LoadBalancer
```

Service Annotation

Last updated: 2023-09-26 14:25:11

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 of the CLB referenced by the current Service. You can view the CLB instance ID in the same VPC as the cluster in the Tencent Cloud CLB console.

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 using an existing CLB, be aware of the impact of different usage methods 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 the labels for 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:
Configure the CLB using the 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 supports configuring TCP, UDP, TCP SSL, HTTP, and HTTPS for specified listening ports.
Use case:
For the detailed usage, see Service Extended 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 NAT64 IPv6 Instance:
  service.kubernetes.io/service.extensiveParameters: '{"AddressIPVersion":"IPV6"}'
- Purchasing a CTCC CLB:
  service.kubernetes.io/service.extensiveParameters: '{"VipIsp":"CTCC"}'
- Create a custom CLB name during creation:
  service.kubernetes.io/service.extensiveParameters: '{"LoadBalancerName":"my_custom_lb_name"}'

service.cloud.tencent.com/enable-grace-shutdown

Note:
Graceful shutdown support for CLB direct connection mode. When a Pod is deleted, it has a DeletionTimestamp and its status is set to Terminating. At this point, the weight of the CLB to the Pod 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:
Enable graceful exit for direct connection mode in CLB. Determine whether the endpoints in the Endpoint object are not-ready, and set the weight of not-ready CLB backends 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.kubernetes.io/qcloud-loadbalancer-internet-charge-type

Note:
The CLB billing type can only be configured at the time of creation and cannot be modified after creation. Modifying this annotation after creation will have no effect.
Specify the billing type for the CLB when creating it. Please use this annotation in conjunction with `service.kubernetes.io/qcloud-loadbalancer-internet-max-bandwidth-out`.
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 settings can only be configured at the time of creation and cannot be modified after creation. Modifying this annotation after creation will have no effect. When creating a CLB, specify the maximum outbound bandwidth for the CLB, which is only applicable to public network LBs. This annotation must be used in conjunction with `service.kubernetes.io/qcloud-loadbalancer-internet-charge-type`.
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 allows you to bind security groups to CLB-type Services. A single CLB can be bound to up to 5 security groups.
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: 2023-09-27 09:24:15

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 the Pod through the NodePort (the traffic is forwarded to 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 according to the weighted round robin algorithm.
- Advanced extended capabilities provided by API Gateway can be used, such as authentication, traffic throttling, grayscale traffic distribution, caching, and downgrade upon circuit breaking.
- Supported by the dedicated API Gateway instance, the underlying physical resources are exclusive to one 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	SDK	Application CLB	Istio Ingress Gateway (provided by Tencent Cloud TCM)	Dedicated API Gateway	Nginx Ingress Controller
Traffic management	Supported protocol	http，https	http，https，http2，grpc，tcp，tcp + tls	http，https，http2，grpc	http，https，http2，grpc，tcp，udp
	IP management	One Ingress rule corresponds to one IP (CLB)	Multiple Ingress rules correspond to one IP (CLB). IP	Multiple Ingress rules correspond to one IP (Dedicated API Gateway). IP address	Multiple Ingress rules correspond to one IP (CLB). IP

			address convergence is supported.	convergence is supported.	address convergence is supported.
	Attribute route	host，URL	More attributes are supported: header, method, query, parameter, etc.	More attributes are supported: header, method, query, parameter, etc.	More attributes are supported such as header, cookie.
	Traffic behavior	Unavailable	Rewrite, redirection, etc. are supported.	Redirection, custom request, custom response, etc. are supported.	Rewrite, redirection, etc. are supported.
	Region-aware load balancing	Unavailable	This feature is supported.	Unavailable	Unavailable
Application access addressing	Service discovery	Single Kubernetes cluster	Multiple Kubernetes clusters + heterogeneous service	Multiple Kubernetes clusters	Single Kubernetes cluster
Secure service	SSL configuration	This feature is supported.	This feature is supported.	This feature is supported.	This feature is supported.
	Authentication and authorization	Unavailable	This feature is supported.	This feature is supported.	This feature is supported.
Observability	Monitored metrics	Supported. View in CLB.	Supported (Cloud native monitoring or Tencent Cloud Observability Platform)	Supported. View in API Gateway.	Supported. View in cloud native monitoring.
	Call tracking	Unavailable	This feature is supported.	Unavailable	Unavailable
	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: 2023-09-26 14:37:08

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 within a cluster. By configuring forwarding rules, different URLs can access different services within the cluster. To ensure the proper functioning of Ingress resources, the cluster must run an Ingress Controller. The container service in the cluster has the TKE Ingress Controller enabled by default, which is based on the Tencent Cloud Load Balancer.

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.

Tag	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.

Note
The above tags are read-only for users. Please do not modify or delete them, as doing so may result in resource leakage. If you use an existing CLB instance, the Service will only use but not delete the instance.

Ingress Controller Usage Method

In addition to the TKE Ingress Controller provided by Tencent Cloud, the Kubernetes community offers various types of third-party Ingress Controllers. These controllers are designed to expose services at the layer-7 network level. The Kubernetes community generally supports the use of the kubernetes.io/ingress.class annotation to differentiate between various Ingress Controllers, determining which controller should handle the current Ingress resource. The TKE Ingress Controller also supports this annotation, with specific rules and usage recommendations as follows:

- When the Ingress resource does not have the annotation `kubernetes.io/ingress.class`, the TKE Ingress Controller will manage the current Ingress resource.
- When an Ingress resource has the annotation `kubernetes.io/ingress.class` with the value `qcloud`, the TKE Ingress Controller will manage the current Ingress resource.
- When the Ingress resource modifies the content of the annotation `kubernetes.io/ingress.class`, the TKE Ingress Controller will include or exclude it from the management scope based on the annotation content. This operation involves the creation and release of resources.
- When you are certain that you no longer need the TKE Ingress Controller, you can adjust the number of replicas for the

Deployment (kube-system:l7-lb-controller) in the cluster to 0, effectively disabling the TKE Ingress Controller functionality.

> **Note**
> - Before disabling this feature, please ensure that there are no Ingress resources managed by the TKE Ingress Controller in the cluster to avoid issues with load balancer resource release failure.
> - 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: 2023-09-27 09:25:55

Feature Overview

An Ingress is a collection of rules that allow access to services within a cluster. By configuring forwarding rules, you can enable different URLs to access different services within the cluster. For the Ingress resource to function properly, the cluster must run an Ingress-controller. TKE service has enabled the `l7-lb-controller` by default within the cluster, which is based on Tencent Cloud Load Balancer and supports HTTP and HTTPS. Additionally, you can deploy other Ingress controllers within the cluster. You can choose different Ingress types based on your business requirements.

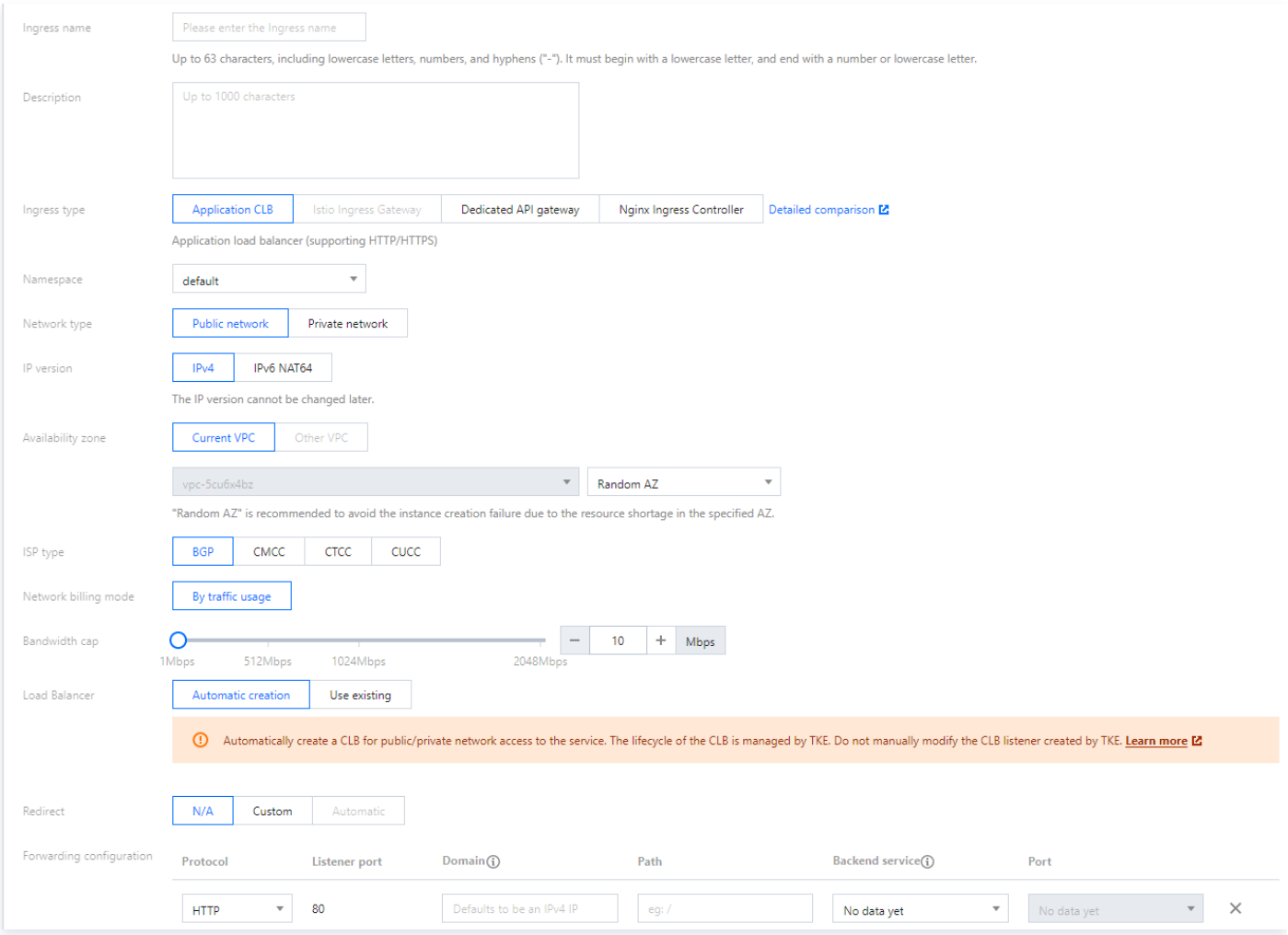
Supports and Limits

- 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 CLB users registered after the upgrade, the domain name-based CLB architecture is adopted by default.
  - Existing users can choose to continue using their current load balancing services without being affected by the upgrade. If you need to upgrade your load balancing service, you must upgrade both Tencent Cloud CLB and TKE; otherwise, the synchronization of all public network Service/Ingress types in TKE may be affected. For details on CLB upgrade operations, see Domain Name-Based Load Balancer Upgrade Guide. To upgrade the Service/Ingress component version in TKE, please contact us through Online Consultation.
- Ingress apiVersion support:
  - The extensions/v1beta1 and networking.k8s.io/v1beta1 API versions of Ingress are not available in v1.22 and later versions. They can only be used in TKE Kubernetes versions ⩽ 1.20.
  - The networking.k8s.io/v1 API is available starting from v1.19 (in TKE scenarios, only even-numbered versions are supported, so it starts from TKE v1.20). This means it can only be used in TKE Kubernetes versions ⩾ 1.20.
  - For more information, please refer to the 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, contact support 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.

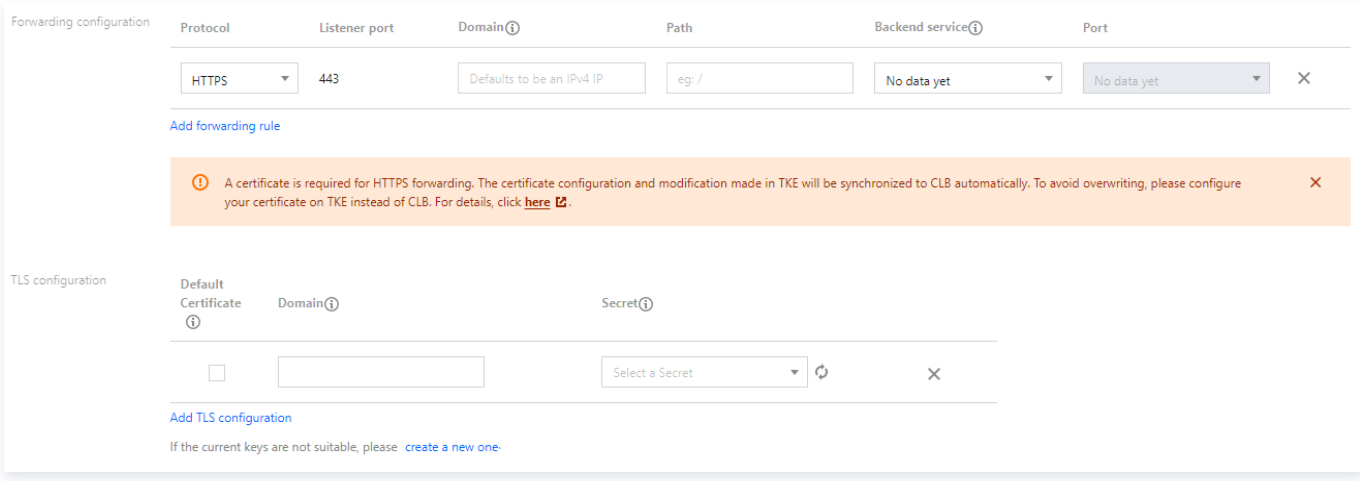
Managing Ingress in the Console

Creating an Ingress

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster management page, click the cluster ID where the Ingress needs to be created.
3. On the cluster details page, select **Service and route** > **Ingress**.
4. On the Ingress page, click **Create**.
5. On the **Create Ingress** page, set the Ingress parameters based on your actual needs. The key parameters are as follows:



- Ingress name: Custom.
- Namespace: select the namespace based on your actual needs.
- Ingress Type: Choose according to your actual requirements. For more information, see Ingress Controllers Comparison.
- Network type: The default value is `Public network`. Select another network if needed.
- Load balancer: Create one automatically or use an existing CLB.
- IP Version: You can select IPv4 or IPv6 NAT64 as needed.
- Forwarding Configuration: The default protocol is **Http**. Please choose the appropriate protocol based on your requirements. If you select **Https**, you need to bind a server certificate to ensure secure access, as shown in the image below:



For more information, see SSL Certificate Format Requirements and Conversion Instructions.

- Forwarding configuration: Set this parameter as needed.

6. Click **Create Ingress** to create an Ingress.

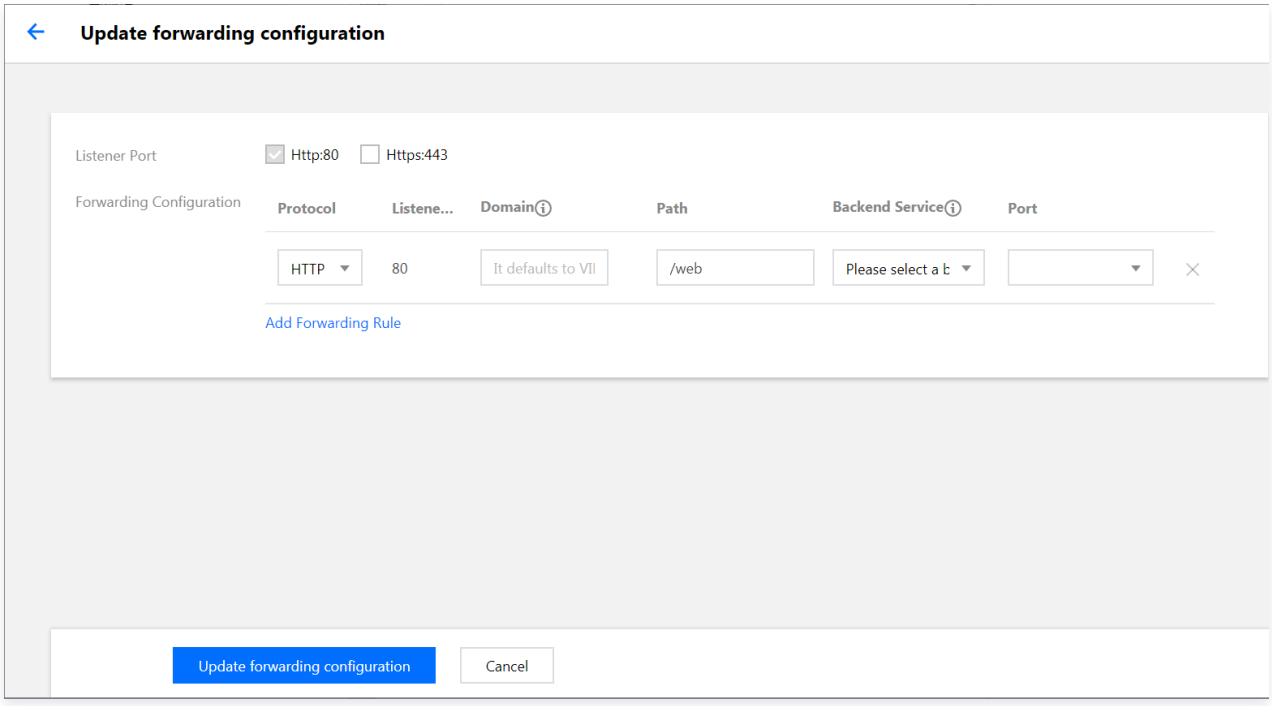
Updating an Ingress

Updating YAML

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster management page, click the cluster ID to enter the management page of the cluster awaiting YAML updates.
3. On the cluster details page, select **Service and route** > **Ingress**.
4. In the row of the Ingress for which you want to update YAML, click **Edit YAML**.
5. On the **Update Ingress** page, edit the YAML and click **Complete**.

Updating a forwarding rule

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the cluster management page, click the cluster ID to enter the management page of the cluster awaiting YAML updates.
3. On the cluster details page, select **Service and route** > **Ingress**.
4. In the row of the Ingress for which you want to update YAML, click **Update forwarding configuration**.
5. On the **Update forwarding configuration** page, modify the forwarding configuration based on your actual needs, as shown in the following image:



6. 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), nginx (nginx-ingress), traefik
    kubernetes.io/ingress.existLbId: lb-xxxxxxxx  ## Specify an existing load balancer to create an Ingress for public/private network access
    kubernetes.io/ingress.subnetId: subnet-xxxxxxxx  ## If you are creating a CLB-type private network Ingress, you need to specify this annotation.
  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 using it.

```
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` Bandwidth cap, range: [1, 2,000] Mbps.
  For 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 by 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 Job is successfully created.

```
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 A

Run the following command to update an Ingress:

```
kubectl edit  ingress/[name]
```

Method B

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: 2023-09-26 14:37:21

Tencent Cloud Container Service TKE supports using existing load balancers through the `kubernetes.io/ingress.existLbId: <LoadBalanceId>` annotation. You can use this annotation to specify the load balancer instance associated with the Ingress.

> **Note**
> The difference between Ingress and Service: Ingress does not support multiple instances using the same load balancer instance, meaning it does not support the reuse feature.

Supports and Limits

- Please ensure that your container business does not share a load balancer resource with your Cloud Virtual Machine (CVM) business.
- Modifications made to the load balancer listeners and backend servers managed by the `Ingress Controller` through the load balancer console are not supported, as they will be automatically overwritten by the `Ingress Controller`.
- When the existing CLBs are used:
  - Do not use one load balancer for multiple Ingresses.
  - The specified load balancer must not have any existing listeners. If there are any, please remove them in advance.
  - This feature only supports load balancers created through the CLB console and does not support load balancers automatically created and managed by the `Service Controller`. In other words, Service and Ingress cannot share the same load balancer.
  - `Ingress Controller` is not responsible for managing load balancer resources; that is, when an Ingress resource is deleted, the load balancer resource will not be deleted and reclaimed.

Use Cases

Using a monthly or yearly subscription load balancer to provide external services

`Ingress Controller` only supports purchasing pay-as-you-go resources when managing the load balancer lifecycle. However, due to the cost advantages of monthly subscription load balancers, users often prefer to purchase them when requiring long-term use of load balancing services.
In such scenarios, users can independently purchase and manage load balancers. By controlling Ingress to use existing load balancers through annotations, the lifecycle management of load balancers is separated from the `Ingress Controller`. An example is shown below:

```
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: /
```

`kubernetes.io/ingress.existLbId: lb-mgzu3mpx` annotation indicates that this Ingress will use the existing load balancer `lb-mgzu3mpx` for Ingress service configuration.

Using TKEServiceConfig to Configure CLBs

Last updated: 2023-09-26 14:37:30

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 Notes

`TkeServiceConfig` can help you quickly configure the load balancer. By using the Ingress annotation **ingress.cloud.tencent.com/tke-service-config:<config-name>**, you can specify the target configuration to be applied to the Ingress.

> **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. When creating an Ingress, set **ingress.cloud.tencent.com/tke-service-config-auto: "true";** to automatically create <IngressName>-auto-ingress-config. You can also directly specify your own `TkeServiceConfig` by using **ingress.cloud.tencent.com/tke-service-config:<config-name>**. Both annotations cannot be used simultaneously.
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 refer to the following complete `TkeServiceConfig` configuration reference to create your own desired CLB configuration. Services will import the configuration through the annotation **ingress.cloud.tencent.com/tke-service-config: <config-name>**.
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.

Sample

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 includes the following configurations:
The Service is of the NodePort type, with two TCP services declared: one on port 80 and the other on port 443.

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: ""     # An empty domain indicates using VIP as the domain name
        rules:
        - url: "/health"
          forwardType :  HTTP # Specifies the backend protocol as HTTP.
          healthCheck:
            enable: false
    - protocol: HTTPS
      port: 443
      defaultServer: "sample.tencent.com" # Default Domain
      keepaliveEnable :  1 # Enable persistent connections for the listener (non-keepalive allowlist users, please do not declare this field)
      domains:
      - domain: "sample.tencent.com"
        rules:
        - url: "/"
          forwardType :  HTTPS # Specifies the backend protocol as HTTPS
          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 less than intervalTime, otherwise an error will occur.
            healthNum: 2
            unHealthNum: 2
            httpCheckPath: "/checkHealth"
            httpCheckDomain: "sample.tencent.com" #Note: Health checks must use a fixed domain name for detection. If you have entered a wildcard domain in .spec.loadBalancer.l7Listeners.protocol.domains.domain, be sure to use the httpCheckDomain field to specify the exact domain name for health checks, as wildcard domains do not support health checks.
            httpCheckMethod: HEAD
            httpCode: 31 # Optional values: 1~31, default 31. 1 means a 1xx response indicates health, 2 means a 2xx response indicates health, 4 means a 3xx response indicates health, 8 means a 4xx response indicates health, and 16 means a 5xx response indicates health. If you want multiple response codes to represent health, add the corresponding values.
          scheduler: WRR
```

This sample includes the following configurations:

The name of the `TkeServiceConfig` is `jetty-ingress-config`, and in the layer-7 listener configuration, two configuration segments are

declared:

1. An HTTP listener on port 80 will be configured, including domain settings, with the default domain corresponding to the CLB's VIP.
   `/health` The health check for the 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:
   - Enable health check, set the health check interval to 10s, the healthy threshold to 2 times, and the unhealthy threshold to 2 times. Perform health checks using HEAD requests, with the check path set to `/checkHealth` and the check domain set to `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
➜ kubectl 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: 2023-09-26 14:37:39

Feature 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.

Scenarios

- 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

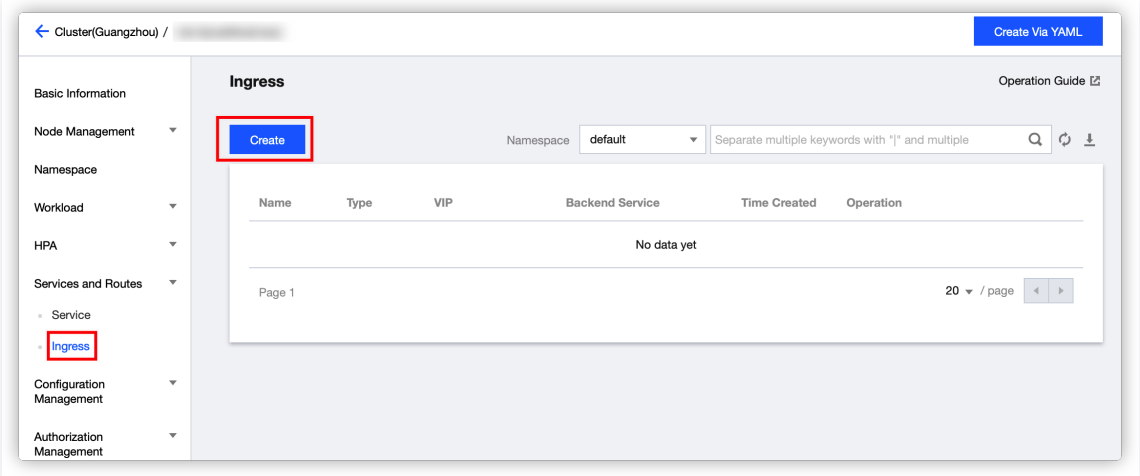
1. If you need to use the CLB that is not in the same VPC as this cluster, you need to connect the VPCs of the current cluster and the CLB via CCN.
   1.1 The IP address ranges of VPCs in different regions must be planned in advance for CCN, and conflicts should be avoided. Otherwise, conflicting routing rules will not take effect, resulting in the inability to forward data.
   1.2 The VPC where the cluster resides cannot join multiple CCNs simultaneously, as this may result in non-unique routes and prevent data plane forwarding.
2. After ensuring that the VPCs are connected, please contact us online 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.

Instructions

You can bind and specify availability zones for CLB Ingress across regions via the console and YAML.

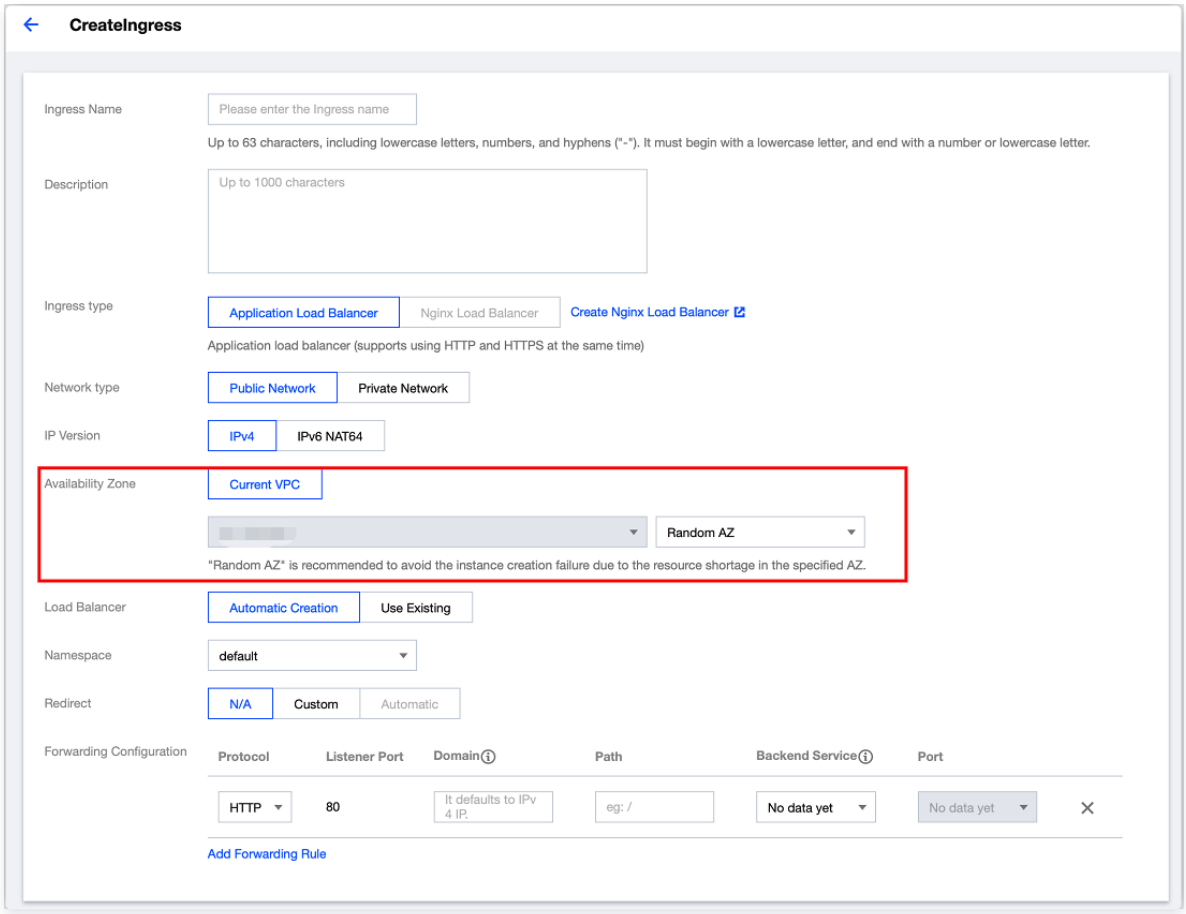
Using the Console

1. Log in to the TKE console and select **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 **Service and route** > **Ingress** in the left sidebar.



4. Click **Create** and configure the relevant availability zone rules in the "Create Ingress" page. The configuration rules are as follows:
   - **Current VPC**: Use the CLB within the VPC where the cluster resides. It is recommended to use a random availability zone, as specifying an availability zone with exhausted resources will prevent the creation of related instances.
   - **Other VPCs**: Only supports other VPCs connected to the current cluster's VPC through Cloud Connect Network. It is

recommended to use random availability zones, as specifying an availability zone with exhausted resources will prevent the creation of related instances.



YAML Method

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 ensuring that the VPCs are connected, please contact us online to apply for this feature.

Example 1

If you only need to specify the availability zone of the VPC where the cluster resides, for example, if the VPC of the cluster is located in Guangzhou, and you need to specify the CLB of Guangzhou Zone 1 for CLB Ingress, you can add the following annotations to the YAML of the Ingress:

```
kubernetes.io/ingress.extensiveParameters: '{"ZoneId":"ap-guangzhou-1"}'
```

Sample 2

If you need to use a CLB that is not in the VPC of the cluster, you need to add the following annotations:

```
ingress.cloud.tencent.com/cross-region-id:
ingress.cloud.tencent.com/cross-vpc-id:
```

The specific examples are as follows:

- To create a CLB for remote access, you need to add the following annotations:

```
ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"
ingress.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"
```

- To select an existing CLB for remote access, you need to add the following annotations:

```
ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"
kubernetes.io/ingress.existLbId: "lb-342wppll"
```

> **Note:**
> If you need to specify the availability zone, you also need to add the annotations of sample 1.

For the detailed Ingress annotations, please see Ingress Annotation.

Ingress Redirection

Last updated: 2023-09-27 09:27:37

Feature Overview

Domain name redirection means when you access a URL in a browser, the web server is set to automatically redirect to another URL.

Scenarios

- 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.

Redirect mode

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**: After users manually configure the original access address and redirection address, the system automatically redirects requests from the original access address to the destination address of the corresponding path. To implement automatic redirection between HTTP and HTTPS requests, you can configure multiple paths as the redirection policies under the same domain name.

Supports and Limits

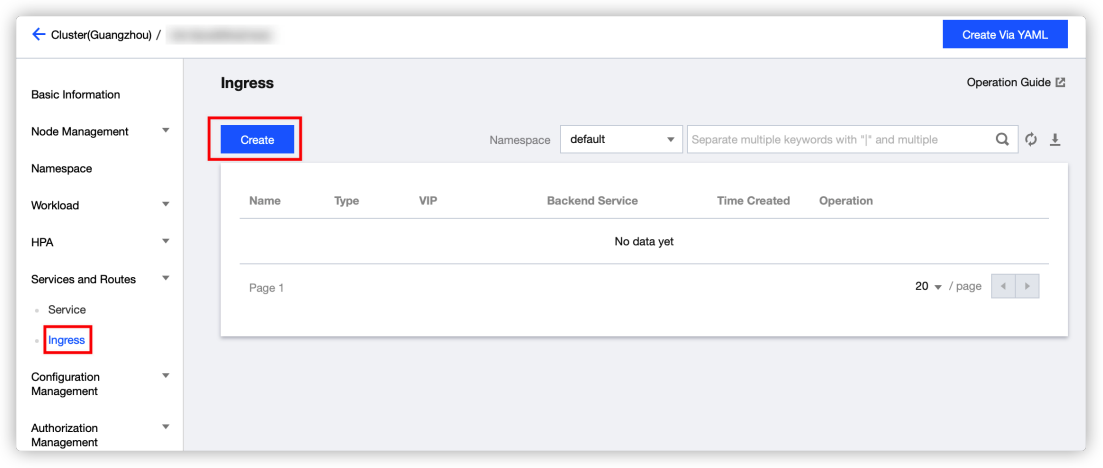
- If you don't have the TKE Ingress redirection annotation declaration, the original logic that doesn't manage redirection rules will be compatible; that is, you can configure redirection rules in the CLB console, and TKE Ingress doesn't process such rules.
- If you don't have the TKE Ingress redirection 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.

Instructions

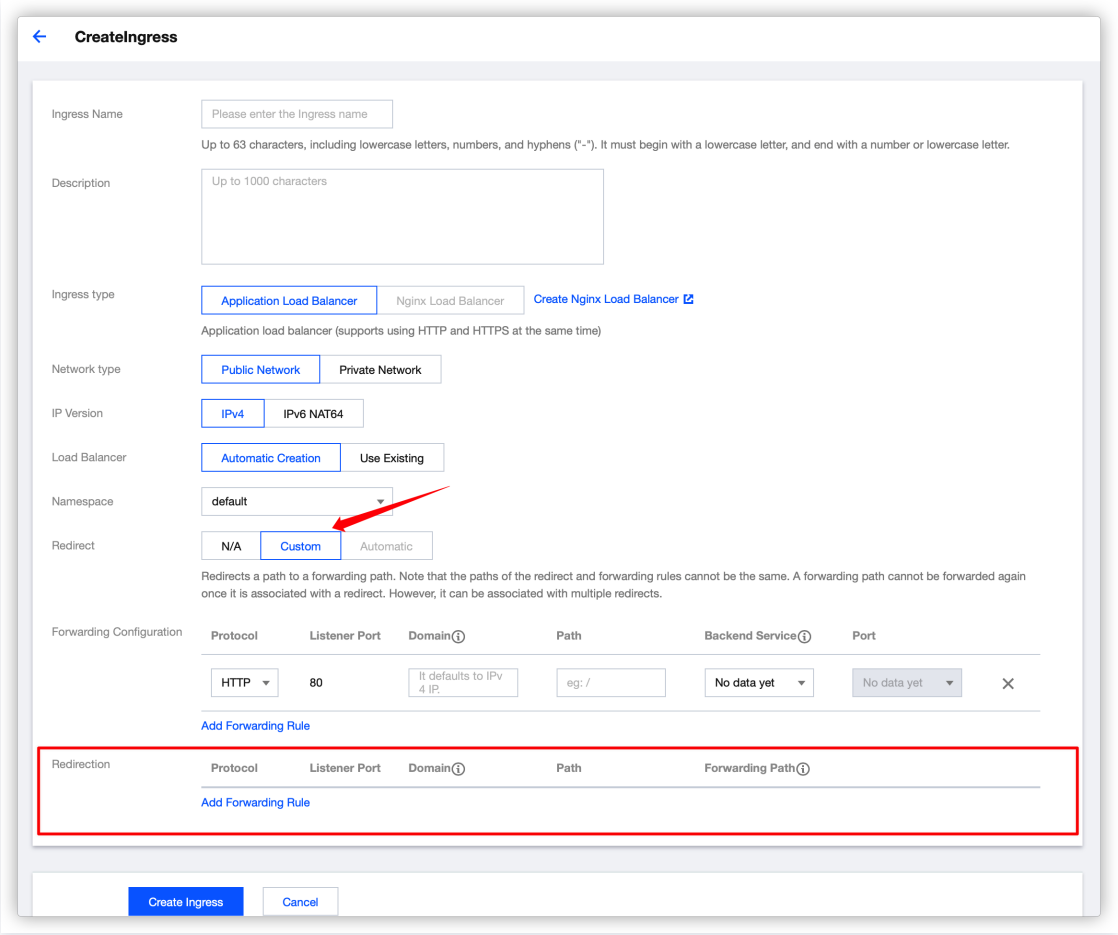
An Ingress supports two redirection configuration ways: console and YAML, as detailed below:

Via the console

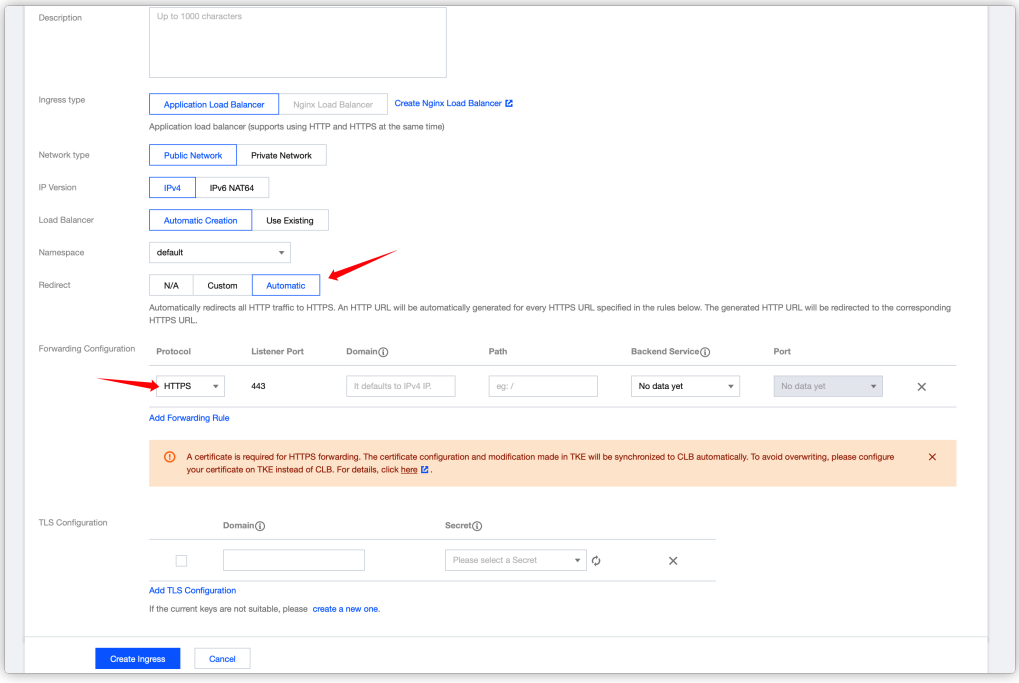
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the "Cluster management" page, click the cluster ID whose Ingress object needs to be modified.
3. On the cluster details page, select **Service and route** > **Ingress** in the left sidebar.



4. Click **Create** and configure the relevant redirection rules in the "Create Ingress" page. The configuration rules are as follows:
   - **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.
     - The configuration method in "**Redirect forwarding configuration**" is the same as that of a regular Ingress forwarding configuration, but the backend points to a specific path within a "forwarding configuration."



- **Automatic**: Applies only to paths with the "HTTPS" protocol in the "**Forwarding Configuration**". An "HTTP" path will be automatically generated for each, with identical paths but different protocols. The "HTTP" path's forwarding rules will automatically redirect to the "HTTPS" path.



Using YAML

Automatic redirection: HTTP redirection to HTTPS

> **Note**
> It takes effect only for the forwarding rule of the HTTPS protocol.

Configure the following annotation in Ingress YAML:

```
ingress.cloud.tencent.com/rewrite-support: "true"  # means redirection is allowed
ingress.cloud.tencent.com/auto-rewrite: "true"     # Indicates automatic redirection
```

After the annotation is configured, the layer-7 forwarding rules that exist in all HTTPS listeners at the forwarding path will be mapped to the HTTP listeners as redirection rules, with the domain name and the path intact.

Manual Redirection

Manual redirection requires adding an annotation and modifying an annotation:

- Added annotation:

```
ingress.cloud.tencent.com/rewrite-support: "true"  # means redirection is allowed
```

- Modified annotation:

```
# Original Annotation Format:
{
  "host": "<domain>",
  "path": "<path>",
  "backend": {
    "serviceName": "<service name>",
    "servicePort": "<service port>"
  }
}

# New Annotation Format:
{
  "host": "<domain>",
  "path": "<path>",
  "backend": {
    "serviceName": "<service name>",
    "servicePort": "<service port>"
  },
  "rewrite": {
    "port": "<rewrite-port>",
    "host": "<rewrite-domain>",
    "path": "<rewrite-path>"
  }
}
```

Sample

A service was previously accessed at `121.4.25.44/path2`, and after a new version is released, it is expected to be accessed at `121.4.25.44/v2/path2`. To do this, you can make the following modification:

1. Add an annotation:

```
ingress.cloud.tencent.com/rewrite-support: "true"  # means redirection is allowed
```

2. Modify the original annotation:

```
# Replace /v1/path1 with /path1 on port 80; replace /v2/path2 with /path2 on port 80. Note: the host can be omitted.
kubernetes.io/ingress.http-rules: '
[{
      "path": "/path1",
      "backend": {
         "serviceName": "path1",
         "servicePort": "80"
      }
   },
   {
      "path": "/path2",
      "backend": {
         "serviceName": "path2",
         "servicePort": "80"
      }
   },
   {
      "path": "/v1/path1",
      "rewrite": {
         "port": 80,
         "path": "/path1"
      }
   },
   {
      "path": "/v2/path2",
      "rewrite": {
         "port": 80,
         "path": "/path2"
      }
   }
]'
```

Modified YAML:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    description: test
    ingress.cloud.tencent.com/rewrite-support: "true"
    kubernetes.io/ingress.class: qcloud
    kubernetes.io/ingress.http-rules: '[{"path":"/path1","backend":{"serviceName":"path1","servicePort":"80"}},
{"path":"/path2","backend":{"serviceName":"path2","servicePort":"80"}},{"path":"/v1/path1","rewrite":
{"port":80,"path":"/path1"}},{"path":"/v2/path2","rewrite":{"port":80,"path":"/path2"}}]'
    kubernetes.io/ingress.https-rules: "null"
    kubernetes.io/ingress.rule-mix: "true"
  name: test
  namespace: default
spec:
  rules:
  - http:
      paths:
      - backend:
          serviceName: path1
          servicePort: 80
        path: /path1
        pathType: ImplementationSpecific
  - http:
      paths:
```

```
      - backend:
          serviceName: path2
          servicePort: 80
        path: /path2
        pathType: ImplementationSpecific
status:
  loadBalancer:
    ingress:
    - ip: 121.4.25.44
```

For the detailed Ingress annotations, please see Ingress Annotation.

Mixed Use of HTTP and HTTPS Protocols through Ingress

Last updated: 2023-09-26 14:37:53

Mixed Rules

In default scenarios, if TLS is not configured in Ingress, services will be exposed through HTTP protocols. If TLS is configured in Ingress, services will be exposed through HTTPS protocols. You can only expose services described by Ingress through one type of the protocol. To deal with this limitation, TKE supports the mixed use of both protocols.
This document describes how to expose services through HTTP and HTTPS protocols simultaneously. Specifically, 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 for `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules` is a `Json Array`. The format of each object is as follows:

```
{
 "host": "<domain>",
 "path": "<path>",
 "backend": {
  "serviceName": "<service name>",
  "servicePort": "<service port>"
 }
}
```

Configuration Directions

`TKE Ingress Controller` supports mixed configuration of `HTTP` and `HTTPS` rules. The steps are as follows:

1. **Enable mixed rules** by adding the annotation `kubernetes.io/ingress.rule-mix` in Ingress and setting it to true.
2. **Rule Matching** matches each forwarding rule in Ingress with `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules`, adding them to the corresponding rule sets. If no matching rule is found in the Ingress annotations, it will be added to the HTTPS rule set by default.
3. **Validation Match** When matching, please verify Host, Path, ServiceName, and ServicePort. The default Host is `VIP`, and the default Path is `/`.

> **Note**
> IPv6 load balancing does not have an IPv4 address and cannot provide the default domain name functionality.

Sample

Sample Ingress: sample-ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.http-rules: '[{"host":"www.tencent.com","path":"/","backend":{"serviceName":"sample-service","servicePort":"80"}}]'
    kubernetes.io/ingress.https-rules: '[{"host":"www.tencent.com","path":"/","backend":{"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 is described in `ingress.spec.rule` in both `kubernetes.io/ingress.http-rules` and `kubernetes.io/ingress.https-rules`.

In this case, CLB will configure forwarding rules in both HTTP and HTTPS protocols to expose services.

Graceful Ingress Shutdown

Last updated: 2023-09-26 14:38:00

Feature Overview

In scenarios where Pods are directly connected to the access layer, when the backend performs a rolling update or a backend Pod is deleted, removing the Pod from the load balancer's backend directly would result in unprocessed requests that the Pod has already received. This is particularly relevant in long-lived connection scenarios, such as conference services, where updating or deleting a workload's Pod would cause the conference to be interrupted immediately.

Scenarios

> **Note**
> This is only effective in the direct access mode. Please 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 all received requests before being removed. At this point, incoming traffic is stopped, but outgoing traffic can still be transmitted normally. The ingress and egress traffic will only be closed once all existing requests have been processed and the Pod is actually deleted.

Instructions

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 access mode
    ingress.cloud.tencent.com/enable-grace-shutdown: "true"  # Indicates graceful shutdown is enabled
  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. **Using preStop**
   To achieve graceful termination, it is essential to handle the SIGTERM signal in your application code. The primary logic is to stop accepting new traffic, continue processing existing traffic, and exit only when all connections are closed. For more information, see the example.
   If your application code does not handle the SIGTERM signal, or you cannot control the third-party libraries or systems to add graceful termination logic, you can also try configuring a preStop for the Pod to implement graceful termination logic, as shown in the following example:

```
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 configuring preStop, please refer to the Kubernetes API documentation.
   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. **Adjust the graceful termination duration using terminationGracePeriodSeconds**
   If a longer graceful termination period is required (preStop + business process termination may exceed 30s), you can customize the terminationGracePeriodSeconds according to the actual situation to avoid being stopped by SIGKILL prematurely. Here is an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: grace-demo
spec:
  terminationGracePeriodSeconds: 60 # Graceful shutdown defaults to 30s, but you can set a longer duration.
  containers:
```

```
- name: lifecycle-demo-container
  image: nginx
  lifecycle:
    preStop:
      exec:
        command:
        - sleep
        - 5s
```

Capabilities

Graceful shutdown only sets the weight of the CLB backend to 0 when the Pod is deleted. If the Pod becomes unhealthy during its operation, setting the weight of the backend to 0 can reduce the risk of service unavailability. You can use the Annotation: `ingress.cloud.tencent.com/enable-grace-shutdown-tkex: "true"` to achieve this graceful exit capability. This Annotation sets the weight of the not-ready CLB backend to 0 based on whether the endpoints in the Endpoint object are not-ready.

Ingress Certificate Configuration

Last updated: 2023-09-26 14:38:08

Scenario

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.

Supports and Limits

- 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.

Sample

TKE supports configuring certificates for the CLB HTTPS listener created by Ingress through the `spec.tls` field. The secretName refers to the Kubernetes Secret resource containing the Tencent Cloud certificate ID. An example is shown below:

Ingress

Creating via YAML:

```
spec:
  tls:
  - hosts:
    - www.abc.com
    secretName: secret-tls-2
```

Secret

Creating via YAML

```
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 required for two-way certificates. See the Important below for more information.
kind: Secret
metadata:
  name: tencent-com-cert
  namespace: default
type: Opaque
```

Creating a Role via Console

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:** Customized, using "cos-secret" as an example in this document.
- **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.

> **Note**
> 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.

Instructions

Creating a server certificate in the console

> **Note**
> 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 on how to add key content, see SSL Certificate Format.
4. Click **Submit**.

Creating an Ingress object that uses the certificate

Notes:

- 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

Notes:

- 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.

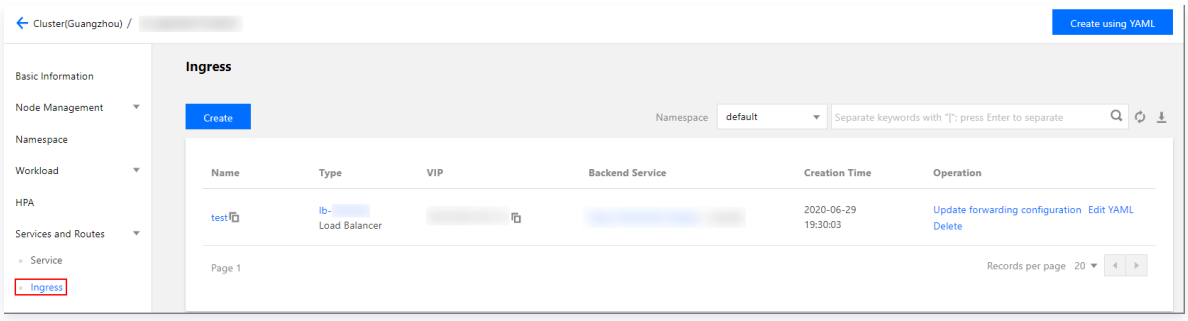
```
kubectl edit secrets [secret-name]
```

2. Modify the Secret resource by changing the value of `qcloud_cert_id` to the new certificate ID. Similar to creating a Secret, modifying the Secret certificate ID requires Base64 encoding. Choose between manual Base64 encoding or specifying `stringData` for automatic Base64 encoding based on your needs.

Updating Ingress objects

Updating an Ingress object in the console

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 **Service and route** > **Ingress** in the left sidebar.



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.

Updating an Ingress object by using YAML

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: 2023-09-26 14:38:15

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:
Configure the Ingress type. The current component management has not configured this annotation, or the annotation content is a Tencent Cloud Ingress resource.
Use case:
`kubernetes.io/ingress.class: "qcloud"`

kubernetes.io/ingress.qcloud-loadbalance-id

Note:
Read-only annotation, the component provides the LoadBalancerId of the current Ingress-referenced CLB instance.
Use case:
`kubernetes.io/ingress.qcloud-loadbalance-id: "lb-3imskkfe"`

ingress.cloud.tencent.com/loadbalance-nat-ipv6

Note:
This is a read-only annotation. It provides an IPv6 address when the user configures or applies for an NAT IPv6 CLB instance.

ingress.cloud.tencent.com/loadbalance-ipv6

Note:
This is a read-only annotation. When users configure or apply for a FullStack IPv6 CLB instance, it provides an IPv6 address.

kubernetes.io/ingress.internetChargeType

Note:
The billing type of the load balancer is only supported during creation and cannot be modified afterward. Changing this annotation after creation will have no effect.
Specify the billing type of the load balancer when creating it. Please use this annotation in conjunction with the `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 settings can only be configured at the time of creation and cannot be modified after creation. Modifying this annotation after creation will have no effect.
Specify the maximum outbound bandwidth for the CLB when creating it. This is only applicable to public network LBs. It should be used in conjunction with the `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"}'`
- Create IPv6 Instance: (The `SubnetId` is required and an IPv6 CIDR block must be assigned. The `MixIpTarget` can provide mixed binding capabilities for IPv4 backends. If your backend is not IPv6, please add this configuration)
  `kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPv6FullChain","SubnetId": "subnet-fqduxxxx"}'`
  `kubernetes.io/ingress.extensiveParameters: '{"AddressIPVersion":"IPv6FullChain","SubnetId": "subnet-fqduxxxx","MixIpTarget":true}'`
- Purchasing a CTCC CLB:
  `kubernetes.io/ingress.extensiveParameters: '{"VipIsp":"CTCC"}'`
- Create in a specified availability zone:
  `kubernetes.io/ingress.extensiveParameters: '{"ZoneId":"ap-guangzhou-1"}'`
- Customize CLB name during creation:
  `kubernetes.io/ingress.extensiveParameters: '{"LoadBalancerName":"my_custom_lb_name"}'`

kubernetes.io/ingress.subnetId

Note:
Specify the creation of a private CLB and designate the subnet to which the CLB belongs.
Use case:
`kubernetes.io/ingress.subnetId: "subnet-3swgntkk"`

kubernetes.io/ingress.existLbId

Note:
Specify the use of an existing CLB as the access layer entry resource.

> **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:
Supports mixed protocol configuration, allowing forwarding paths to be simultaneously carried out on both HTTP and HTTPS. Manual redirection rules configuration is also supported.
Use case:
For the detailed usage, see Ingress with Mixed HTTP and HTTPS Protocols.

ingress.cloud.tencent.com/direct-access

Note:
Supports Layer 7 direct connection to user load balancing. It is important to note the service dependencies for direct connections under various network conditions.
Use case:
For the detailed usage, see Using Services with CLB-to-Pod Direct Access Mode.

ingress.cloud.tencent.com/tke-service-config

Note:
Configure CLB-related settings through tke-service-config, including listeners, forwarding rules, and more.
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:
This annotation is used to automatically create a TkeServiceConfig resource and provide a configuration template, allowing users to configure as needed.
Use case:
`ingress.cloud.tencent.com/tke-service-config-auto: "true"` – For more information, refer to Ingress using TkeServiceConfig to configure CLB.

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:
Provide automatic redirection capability for HTTP ports. All forwarding rules declared on HTTPS ports will create corresponding redirection rules. This requires the use of 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:
Ingress cross-region binding feature, specifying the region to access from. This needs to be used in conjunction with

`kubernetes.io/ingress.existLbId` or `ingress.cloud.tencent.com/cross-vpc-id`.

Use case:

- Create a cross-region load balancer:
  `ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"` `ingress.cloud.tencent.com/cross-vpc-id: "vpc-646vhcjj"`
- Bind to an existing load balancer for cross-region access:
  `ingress.cloud.tencent.com/cross-region-id: "ap-guangzhou"` `kubernetes.io/ingress.existLbId: "lb-342wppll"`

ingress.cloud.tencent.com/cross-vpc-id

Note:

Ingress cross-domain binding feature, specifying the VPC to be accessed. It can be used in conjunction with the `ingress.cloud.tencent.com/cross-region-id` annotation to specify VPCs in other regions.

> **Note**
> This annotation applies to the CLB created and managed by TKE. It is invalid for scenarios that use the existing CLB.

Use case:

Create a cross-region CLB:
`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:

Graceful shutdown support for CLB direct connection mode. When a Pod is deleted, it has a DeletionTimestamp and its status is set to Terminating. At this point, the weight of the CLB to the Pod 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, please see Graceful Ingress Shutdown.

ingress.cloud.tencent.com/enable-grace-shutdown-tkex

Note:

Enable graceful exit for direct connection mode in CLB. Determine whether the endpoints in the Endpoint object are not-ready, and set the weight of not-ready CLB backends 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 capabilities.

ingress.cloud.tencent.com/security-groups

Note:

This annotation allows you to bind security groups to CLB-type Ingress. A single CLB can be bound to up to 5 security groups.

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 `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.

Note:

- 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"`

Nginx Type Ingress Overview

Last updated: 2023-09-26 14:39:06

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**: The entry point for using Nginx-ingress in TKE. You can easily install and deploy Nginx-ingress from the cluster's 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: 2023-09-27 09:27:08

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**. On the **Add-On Management** page, click **Create**.
4. On the **Create Add-on** page, select NginxIngress.
5. Click **Done** to install the add-on. After installation, you can view the add-on details in **Add-On Management**.

Supports and Limits

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 CLB users registered after the upgrade, the domain name-based CLB architecture is adopted by default.
- Existing users can continue to use their current load balancing services without being affected by the upgrade. If you need to upgrade your load balancing service, you must upgrade both Tencent Cloud CLB and TKE; otherwise, the synchronization of all public network type Service/Ingress in TKE may be affected. For details on CLB upgrade operations, see Domain Name-Based Load Balancer Upgrade Guide. To upgrade the Service/Ingress component version in TKE, please contact us through Online Consultation.

Installation Methods

You can use the following installation methods to install Nginx-ingress in TKE based on the requirements of your business scenarios:

- Deploy using DaemonSet in a specified node pool
- 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 a critical traffic gateway, it is not recommended to deploy Nginx alongside other services on the same node. It is advised to use a designated node pool for deploying Nginx-ingress. The deployment architecture is illustrated below:

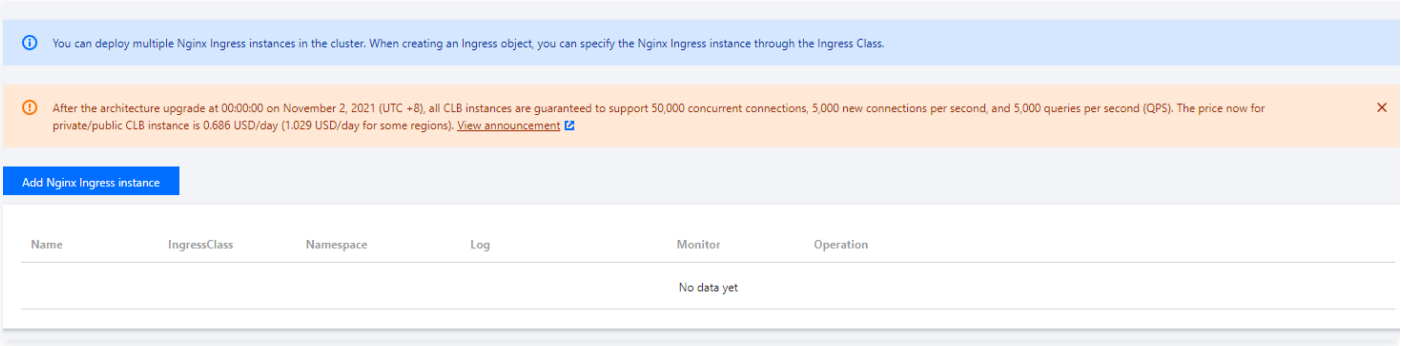
Installation

> **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.

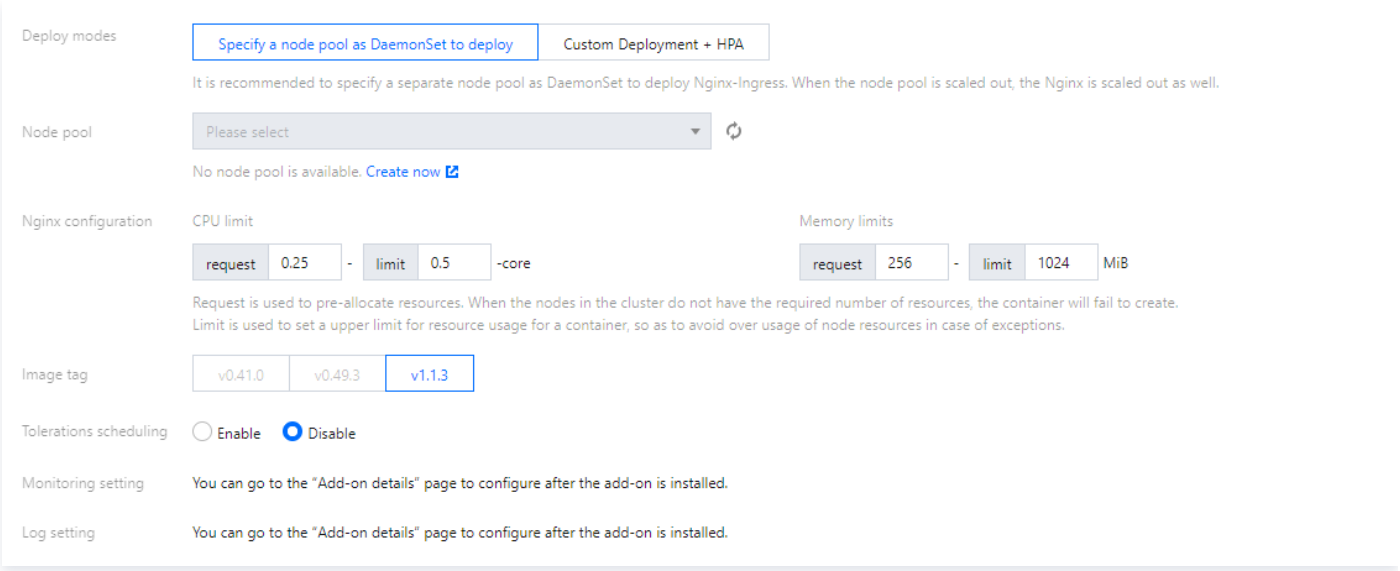
1. Prepare a node pool for Nginx-ingress deployment and set a taint to prevent other Pods from being scheduled to the node pool.

For more information on the deployment node pool, see Node Pool Overview.

2. Install the NginxIngress Add-on in the cluster.
3. On the cluster information page, select **Service and Route > NginxIngress** and click **Create New Nginx Ingress Instance** (multiple Nginx instances can coexist within a cluster).



4. In **Create NginxIngress**, choose **Deploy as DaemonSet in specified node pool** under deployment options, and configure other parameters as needed. As shown in the image below:



- **Node pool**: configure the node pool.
- **Nginx Configuration**: Set the Request value smaller than the node pool's configuration (as nodes have reserved resources). Limit does not need to be set.
- **Image tag**:

Kubernetes Versions	Versions Supported by Nginx-ingress	Image Versions Supported by Nginx Instances
<=1.18	1.1.0、1.2.0	v0.41.0、v0.49.3
	1.0.0	v0.41.0
1.20	1.1.0、1.2.0	v1.1.3
	1.0.0	v0.41.0
>=1.22	1.1.0、1.2.0	v1.1.3

Note

1. **EIP Usage Notes for Add-on**: Nginx Ingress add-on versions 1.0.0 and 1.1.0 rely on Tencent Cloud Elastic Public IP (EIP) service. From v1.2.0 onwards, the add-on no longer depends on EIP. If you have EIP usage restrictions, it is

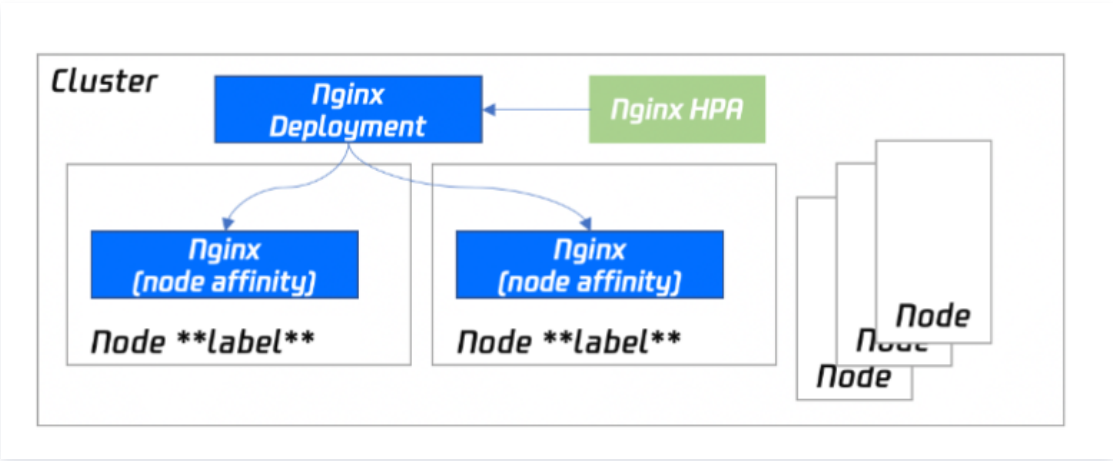
recommended to upgrade the Nginx Ingress add-on. Upgrading the add-on does not affect existing Nginx Ingress instances, has no impact on business access, and does not compromise data security.

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**.

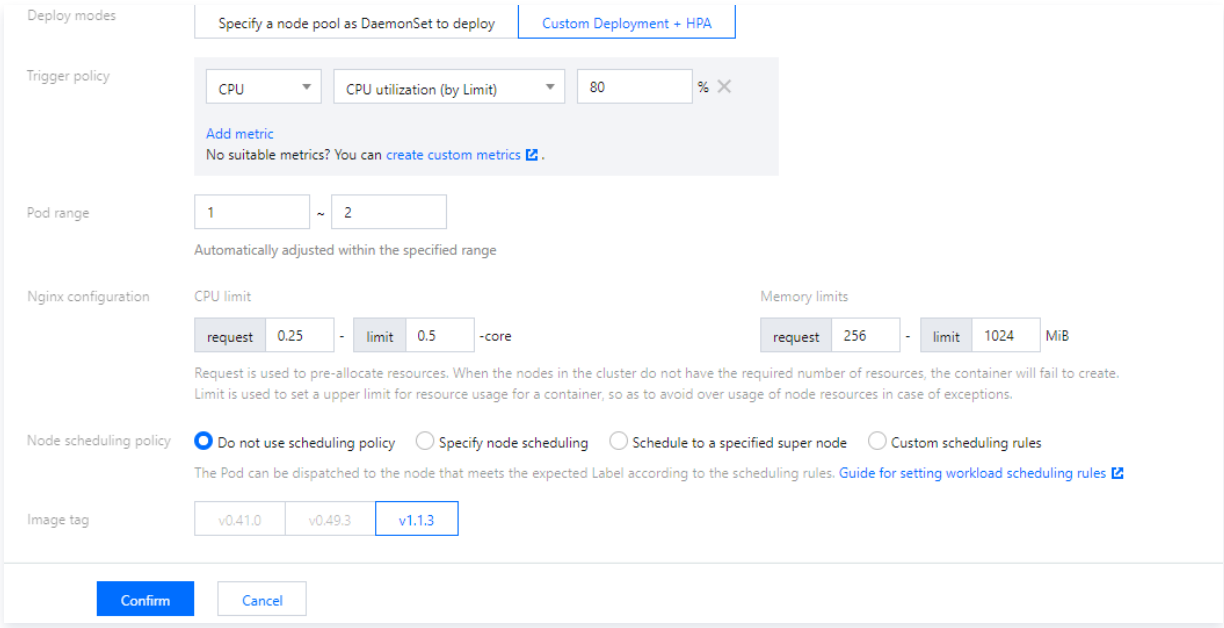
Deploying via "Deployment + HPA" and specifying a scheduling policy

By deploying Nginx-ingress using the Deployment + HPA method, you can configure taints and tolerations to distribute Nginx and business Pods based on your business requirements. In conjunction with HPA, you can set Nginx to scale elastically based on metrics such as CPU and memory usage. The deployment architecture is illustrated in the following diagram:



Installation

1. Prepare a node pool for Nginx-ingress deployment and set a taint to prevent other Pods from being scheduled to the node pool. For more information on the deployment node pool, see Node Pool Overview.
2. Install the NginxIngress Add-on in the cluster.
3. On the cluster information page, select **Service and Route > NginxIngress** and click **Create New Nginx Ingress Instance** (multiple Nginx instances can coexist within a cluster).
4. In "Create NginxIngress", choose **Custom Deployment+HPA Deployment** in the deployment options, and configure other parameters as needed. As shown in the image below:



- **Nginx Configuration**: Set the Request value smaller than the node pool's configuration (as nodes have reserved resources). Limit does not need to be set.
- **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**
> - Version notes: For information about Nginx instance versions, see ingress-nginx documentation.
> - Upgrading a Cluster: If you need to upgrade your cluster, please refer to Upgrade Cluster Procedure.
> - Upgrade Nginx Ingress Add-on: If you need to upgrade the Nginx Ingress add-on, please refer to the Add-on Upgrade Steps.

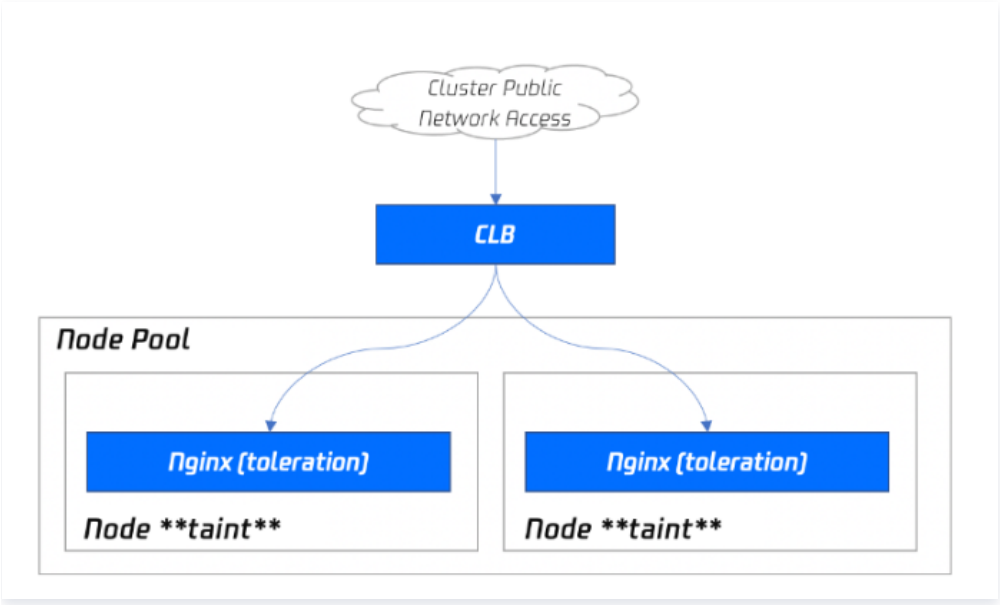
5. Click **OK**.

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, it is recommended to use CLB to directly connect to Nginx Service. The following image shows a load balancing example deployed with a node pool.



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 NodePort, traffic goes through the CLB within K8s and is then forwarded through iptables or ipvs to Nginx. This increases network time consumption.
- Passing through 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 with 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 NodePort, and probe packets are ultimately forwarded to the Pods of Nginx-ingress. If the CLB is bound with 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 the 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 Management" page, click the ID of the target cluster to enter the cluster details page.
3. In the left menu, click **Add-On Management** to access the "Add-on List" page.
4. Click **Update Nginx Configuration** on the right side of the component that requires parameter settings, and enter the "Nginx Configuration" page.
5. Select the target Nginx-ingress instance and click **Edit YAML**.
6. Edit the settings on the "Update ConfigMap" page and click **Finish** to configure the parameters.

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" $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
  keep-alive-requests: "10000"
  max-worker-connections: "65536"
  upstream-keepalive-connections: "200"
```

> **Note**
> - Do not modify `access-log-path` , `error-log-path` , and `log-format-upstream` ; otherwise, CLS log collection will be affected.
> - If you need to configure different parameters for your business, see Official Document .

Using Nginx-ingress Object to Access External Traffic of the Cluster

Last updated: 2023-09-26 14:39:20

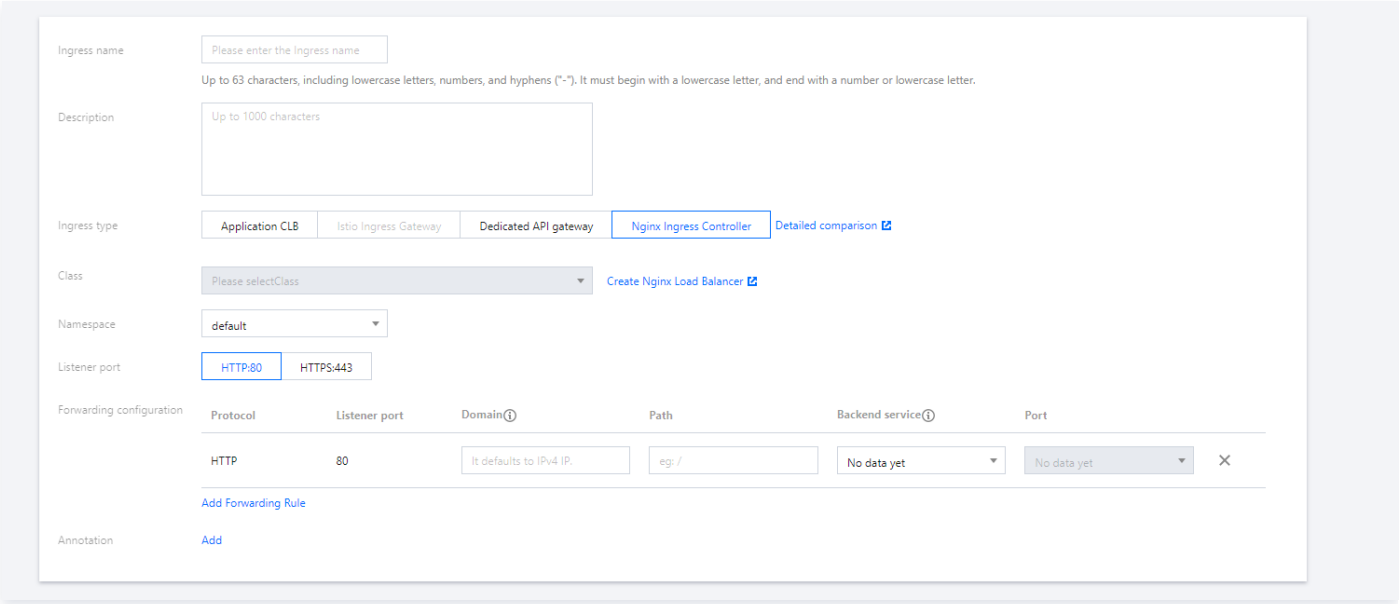
Preparations

- 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 on **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 enter the Create Ingress page.
5. Configure the Ingress parameters based on your actual needs, as shown in the image below:



   - Ingress type: select **Nginx Ingress Controller**.
   - Class: Select an Nginx Ingress instance, or click **Create Nginx Load Balancer** on the right if none is available.
   - Forwarding configuration: configure forwarding rules as needed.
6. Click **Create Ingress**.

Managing Nginx-ingress using Kubectl

Before introducing the IngressClass resource and ingressClassName field in Kubernetes, the Ingress class was specified by the `kubernetes.io/ingress.class` annotation in the Ingress. Here's an example:

```
metadata:
  name:
  annotations:
    kubernetes.io/ingress.class: "nginx-public". ## Corresponds to the Nginx-ingress instance name in the TKE cluster's Nginx-ingress component.
```

Related Actions

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: 2023-09-26 14:39:27

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.

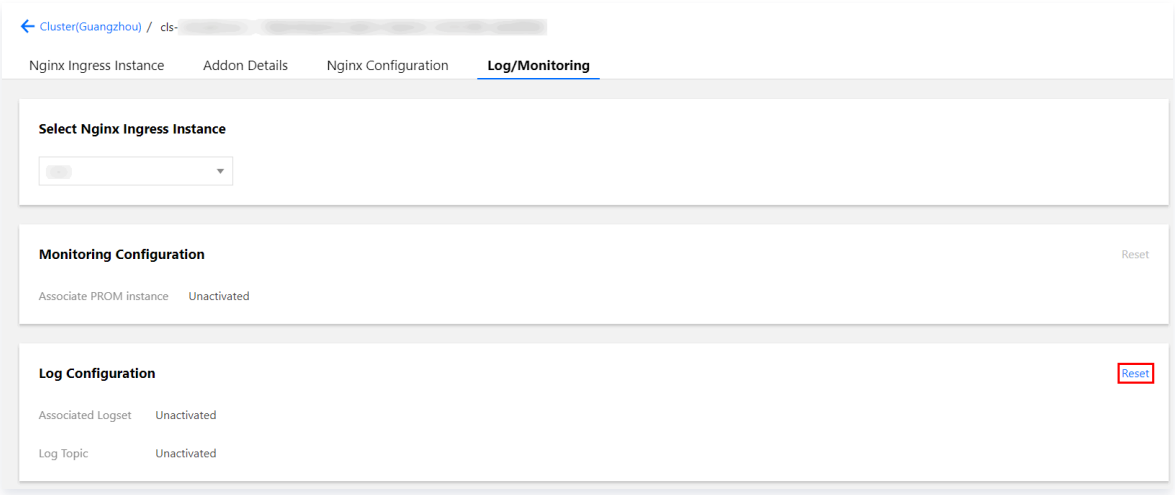
Preparations

Log collection has been enabled in the Feature Management section of the TKE console. For more information, see Enabling Log Collection.

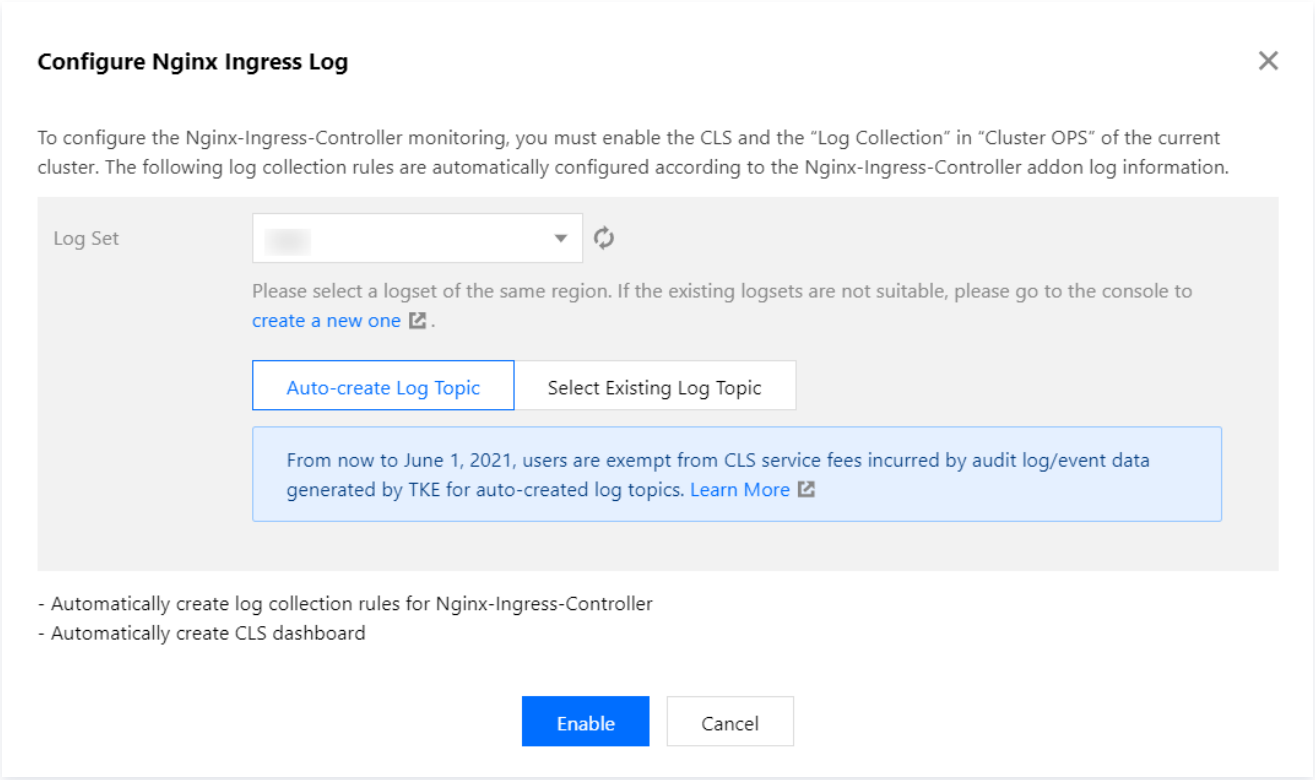
TKE Nginx-ingress Log Collection

Log collection directions

1. Install Nginx-ingress Addon for the target cluster.
2. In **Services and Routes > NginxIngress**, select the installed instance name to enter the add-on details page.
3. In the **Operations** tab, select **Reset** on the right side of the log configuration. As shown below:



4. In the pop-up window, select the specified logset. If not specified, a new logset will be created, as shown below:



5. Click **Enable**.

> **Note**
> For information on CLS billing rules and billing standards, see Billing Overview.

Log collection metrics

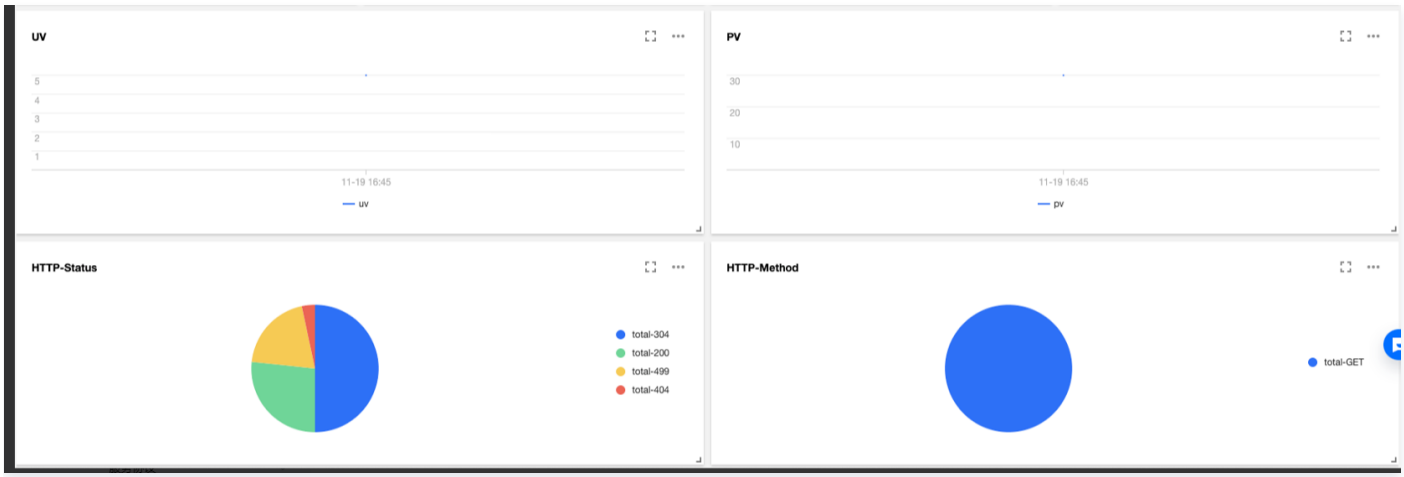
The log collection metrics are as follows:

```
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+)
    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

Enabling the log collection feature in TKE Nginx-ingress will automatically create a standard log dashboard for you. You can also configure charts in the CLS console according to your business needs, as shown below:



Documentation

If you need to customize log collection rules and indexes, see Custom Nginx Ingress Log.

Installing Nginx Add-on and Instance with Terraform

Last updated: 2023-09-26 14:39:35

Preamble

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 the 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. For detailed instructions, please refer to the Tencent Cloud Terraform Application Guide.
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\":\"ingressnginx\",\"chartVersion\":\"1.2.0\"}}}"
}
```

Step 3: Declarative Installation of the Nginx Instance

For more information about Kubernetes Provider configuration, see the Official Documentation.
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" = "{\"InternetAccessible\":
{\"InternetChargeType\":\"TRAFFIC_POSTPAID_BY_HOUR\",\"InternetMaxBandwidthOut\":10}}"
        }
        "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: 2023-09-26 14:48:36

Cluster storage management is an important part of preserving business data. At present, Tencent Kubernetes Engine (TKE) supports multiple classes of storage.

Storage Type

Storage Type	Note	How to Use
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.	TKE supports using Cloud Block Storage (CBS) by creating PVs/PVCs and mounting dynamic (static) volumes to workloads. For more information, please refer to Using Cloud Block Storage CBS.
Tencent Cloud File Storage (CFS)	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. 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.	TKE supports using File Storage (CFS) by creating PVs/PVCs and mounting dynamic (static) volumes to workloads. For more information, please refer to Using File Storage 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.	TKE supports using Cloud Object Storage (COS) by creating PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) and mounting static volumes to workloads. For more information, please refer to Using Cloud Object Storage (COS).
Others	-	When creating workloads, TKE also supports using other types of local storage, such as host paths, NFS disks, ConfigMaps, and Secrets. For more information, please refer to Using Other Storage Volumes.

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.

Relevant Concepts

- **PersistentVolume (PV)**: Storage resources within a cluster. PVs are independent of a Pod's lifecycle and can be created with different types based on the StorageClass.
- **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: 2023-09-27 09:28:12

Scenario

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.

Prepare.

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 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**. On the **Add-On Management** page, click **Create**.
4. On the **Create add-on** page, select **Tencent Cloud COS**.
5. Click **Finish**.

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 on how to create a sub-user and grant access and management permissions to the sub-user, see Creating a Custom Sub-user.

1. Log in to the Access Management Console with a sub-account, and select **Access Key** > **Manage API Key** in the left sidebar.
2. On the **API Key Management** page, click **Create Key** and wait until the key is created.

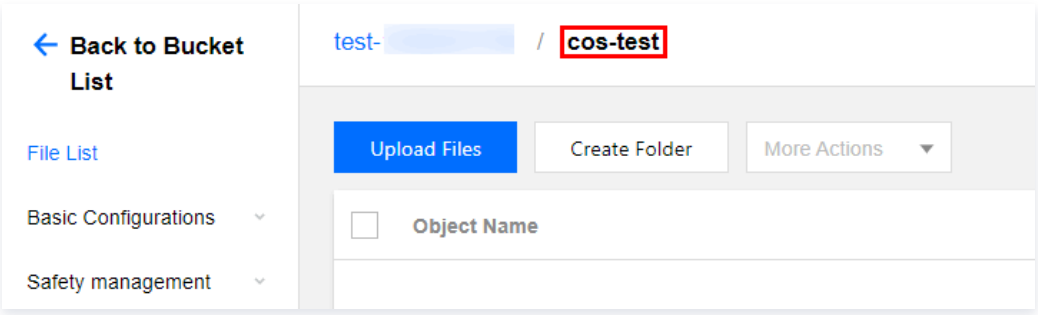
> **Note**
> - One sub-user can have at most two API keys.
> - An API key is an important credential for creating Tencent Cloud API requests. To keep your assets and services secure, store your keys appropriately and change them regularly. Delete old keys when new ones are created.

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. Retrieving the bucket subdirectory

1. On the **Bucket List** page, click the name of the created bucket to go to the bucket details page.
2. Select **File List** in the left sidebar. In the file list, choose the subfolder you want to mount and enter its details page. In the top right corner, obtain the subdirectory path `/costest`, as shown in the image below:

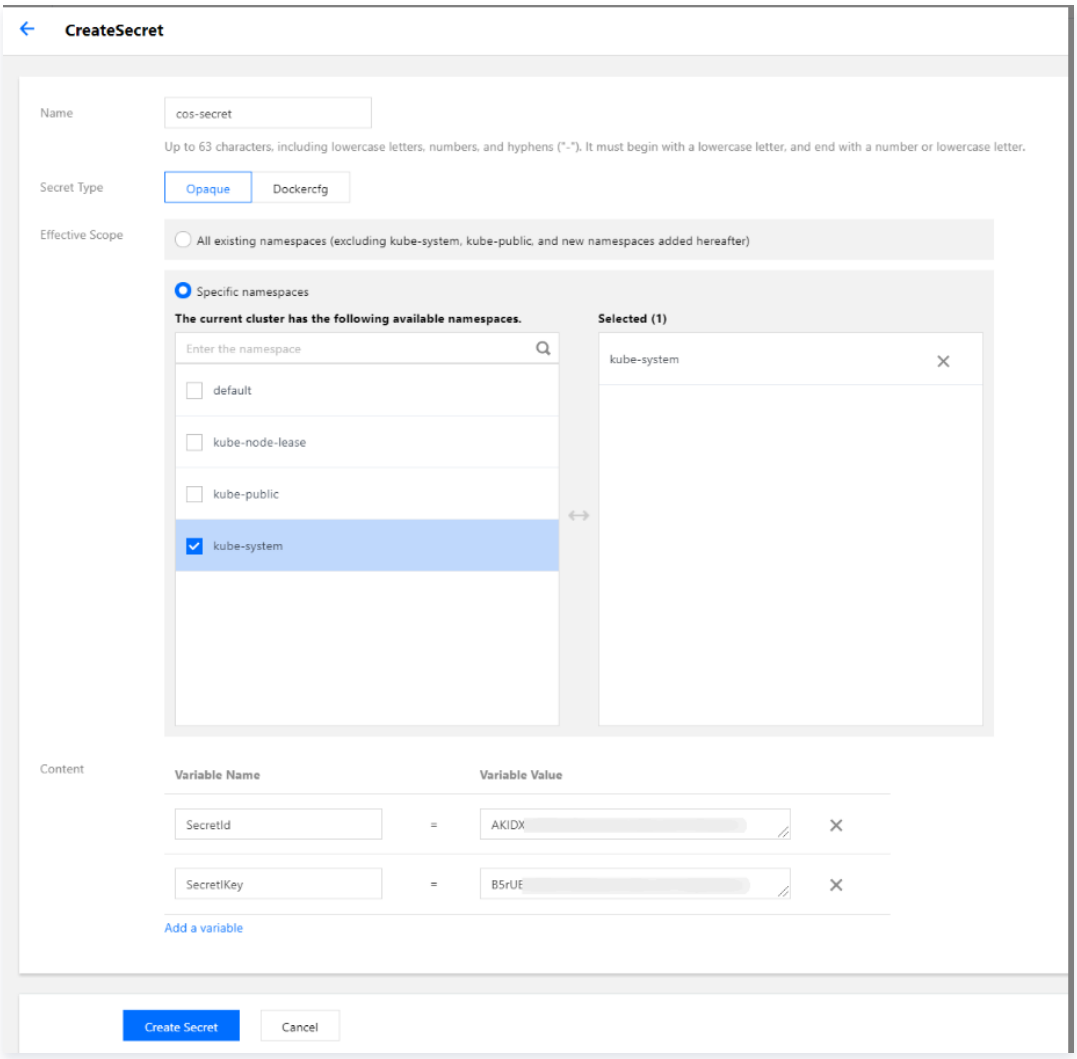


Instructions

Using COS via the console

Step 1. Create a Secret that can access COS

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 **Configuration Management** > **Secret**. On the Secret page, click **Create**.
4. On the **Create Secret** page, configure the settings according to the information provided below, as shown in the following figure:



- **Name**: set 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.

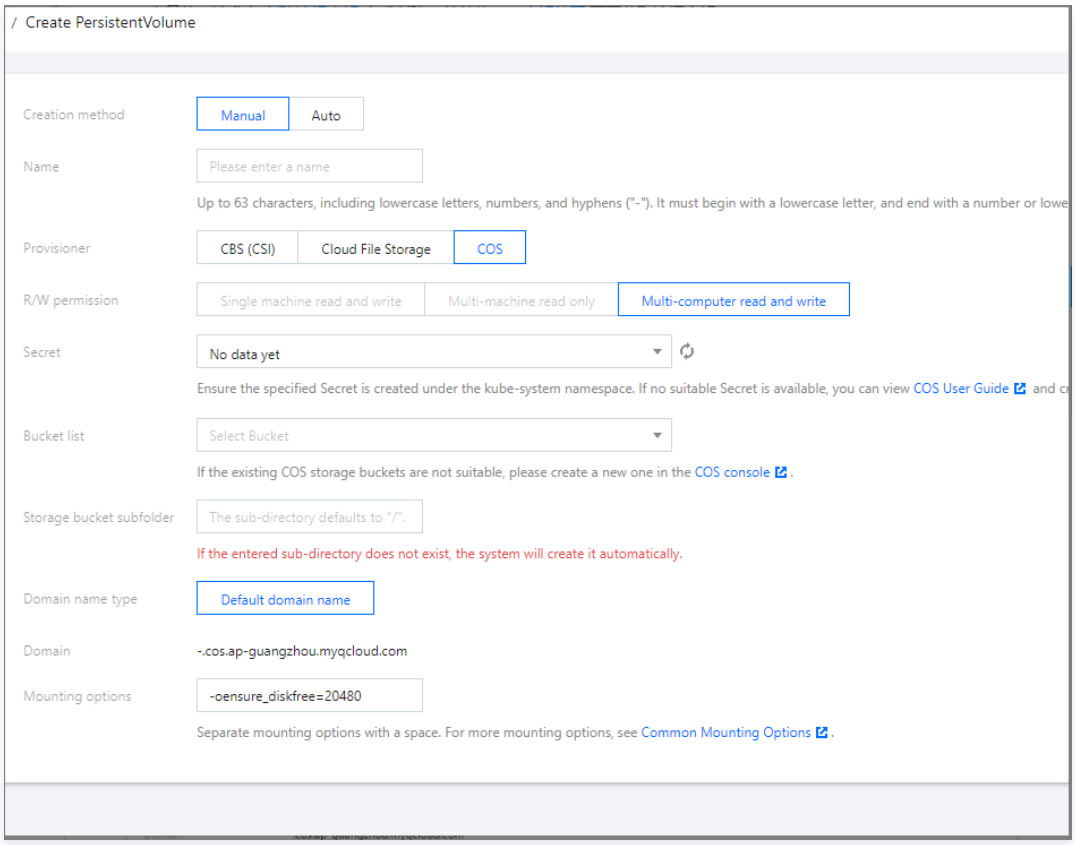
- **Content**: Specify the access key required by the Secret to access the bucket. The value must contain the variable names `SecretId` and `SecretKey` and their corresponding variable values. Refer to Creating an access key to create an access key, and go to the API Key Management page to get its details.

5. Click **Create Secret**.

Step 2: Create a PV that supports COS-CSI static configuration

> **Note**
> This step requires a bucket. If no bucket is available in the current region, create one. For more information, see Creating a 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 parameter information is as follows:

- **Creation Method**: select **Manual**.
- **Name**: set a custom name. This document uses `cos-pv` as an example.
- **Provisioner**: select **COS**.
- **R/W permission**: COS only supports multi-host read and write.

> **Note**
> - **Single machine read and write**: Currently, CBS disks can only be mounted to the same machine, and therefore, CBS supports only data read and write processing on a single machine.
> - **Multi computer read and write**: CFS/COS objects can be mounted to multiple machines, and therefore, CFS/COS supports data read and write processing on multiple machines.

- **Secret**: Select the Secret created in Step 1. This document uses `cos-secret` as an example (ensure the Secret is created under the `kube-system` namespace).
- **Bucket List**: the list of buckets used to save COS objects. You can select an available bucket as required.

- **Storage bucket subdirectory**: 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.
- **Mount Option**: 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 option is used to set related restrictions. The mount 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 20480MB, the COSFS tool will fail to start.

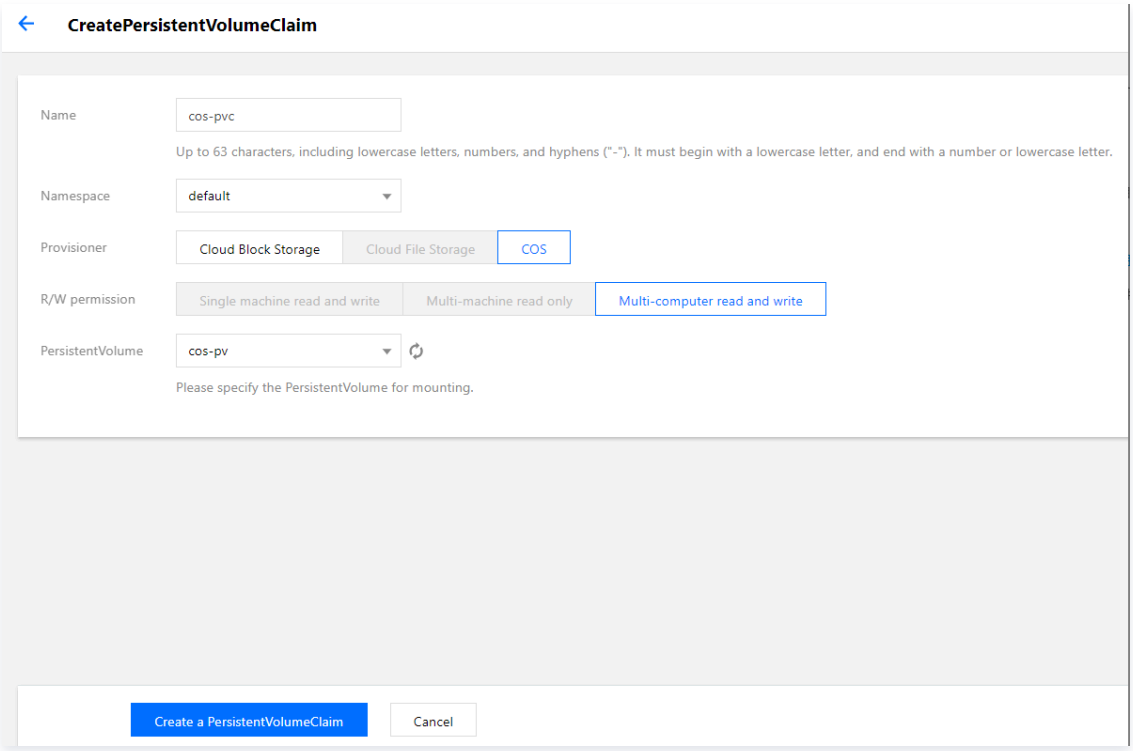
> **Note**
> Different mounting options must be separated with spaces. For more mounting options, see Common Mounting Options.

3. Click **Create PersistentVolume**.

Step 3. Create a PVC to bind a PV

> **Note**
> You cannot 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**: set 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-host read and write.
- **PersistentVolume**: Select the PV created in Step 2. In this example, we use `cos-pv`.

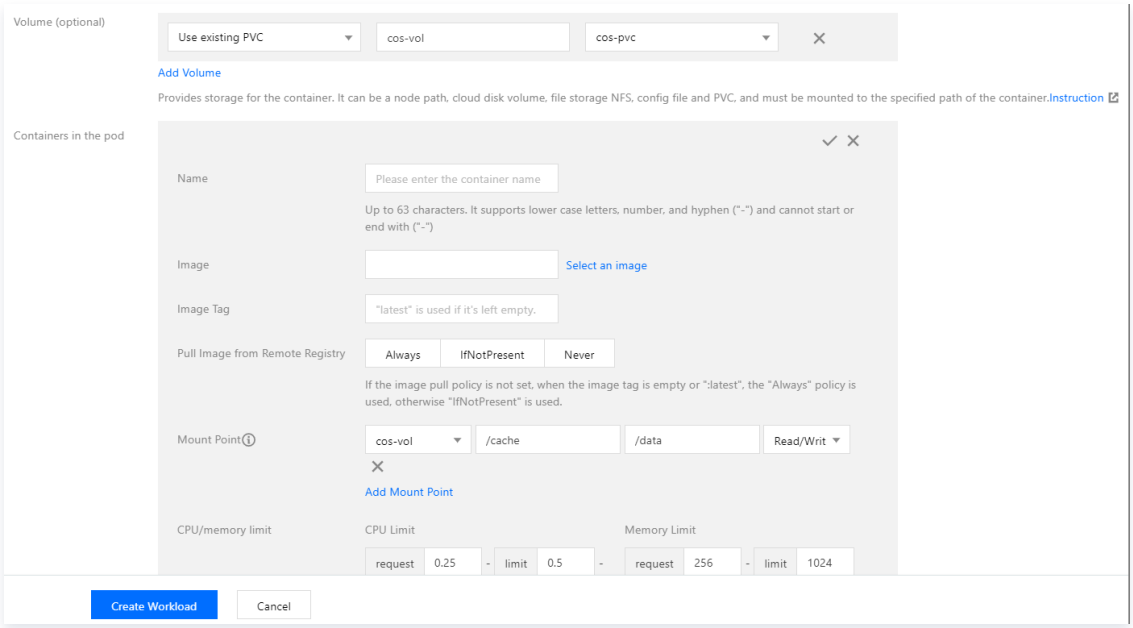
3. Click **Create PersistentVolumeClaim**.

Step 4. Create a Pod that uses a PVC

> **Note**

This step creates a Deployment workload as an example.

1. In the target cluster details page, select **Workloads** > **Deployment** from the left menu, and click **Create** on the Deployment page.
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**: set a custom name. This document uses `cos-vol` as an example.
     - **Select PVC**: Choose the PVC created in Step 3. This document uses `cos-pvc` as an example.
   - **Containers in the Pod**: Click **Add mount target** to set a mount target.
     - **Volume**: Select the volume "cos-vol" 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 `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 secret key with these docs:
  # 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 name.(this value must different from other
pv's volumeHandle)
    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 pvc.
  # volumeName: cos-pv
  # Currently cos only supports static provisioning, the StorageClass name should be empty.
  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
```

Relevant Information

For more information on how to use COS, see README_COSFS.md.

Use File to Store CFS
CFS Instructions

Last updated: 2023-09-26 14:49:22

Scenario

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: Dynamic CFS Creation
- Method 2: Using Existing CFS

Prerequisites

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. Select **Cluster** from the left sidebar to enter the cluster management interface.
3. Select the ID of the cluster for which you want to create an add-on and click **Add-On Management** on the left on the cluster details page.
4. On the "Add-On Management" page, click **Create** to enter the "New Add-On" page.
5. Select **Tencent Cloud CFS** and click Finish.

Instructions

Dynamically Creating Cloud Object Storage

If you need to dynamically create a CFS disk, you can perform the following steps:

1. Create a StorageClass of the CFS type and define the CFS template to use.
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 set the container mount point. For detailed operation steps, please refer to Managing CFS Templates with StorageClass.

Using Existing CFS

When you need to use an existing CFS, follow these steps:

1. Create a PV using an existing CFS.
2. When creating a PVC, set the StorageClass and capacity to match the PV created earlier.
3. When creating a workload, select the aforementioned PVC.
   For detailed operation steps, please refer to Managing CFS with PV and PVC.

Relevant Information

For more information on how to use CFS, please refer to README_CFS.md.

Managing CFS Templates by Using a StorageClass

Last updated: 2023-09-27 09:28:43

Scenario

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 Cloud File Storage 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.

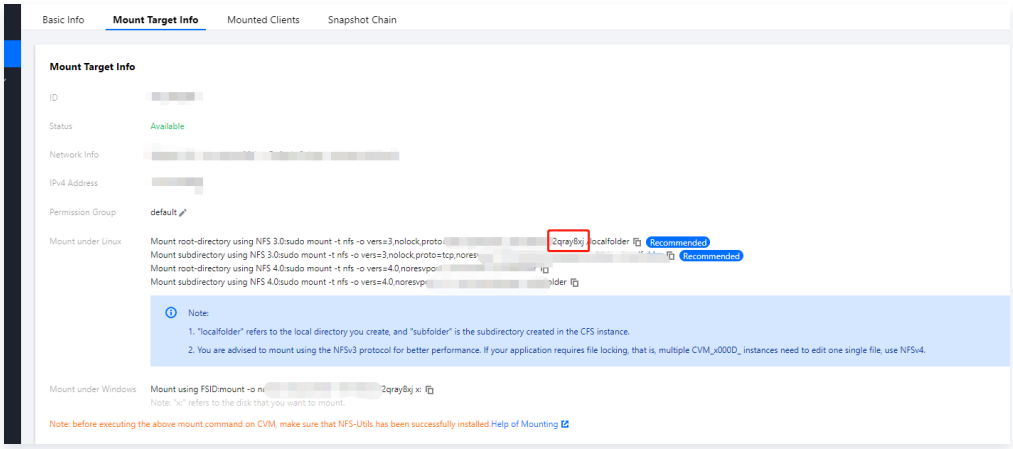
When creating a StorageClass, you need to set the CFS subnet to ensure that every AZ in the CFS's VPC has a suitable subnet. We recommend you create a subnet in advance as instructed in Creating Subnets.

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. Obtain the File System ID (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 Information** tab and obtain the FSID of the file system from "Mount under Linux". As shown in the figure below, `a43qadkl` is the FSID of this file system.



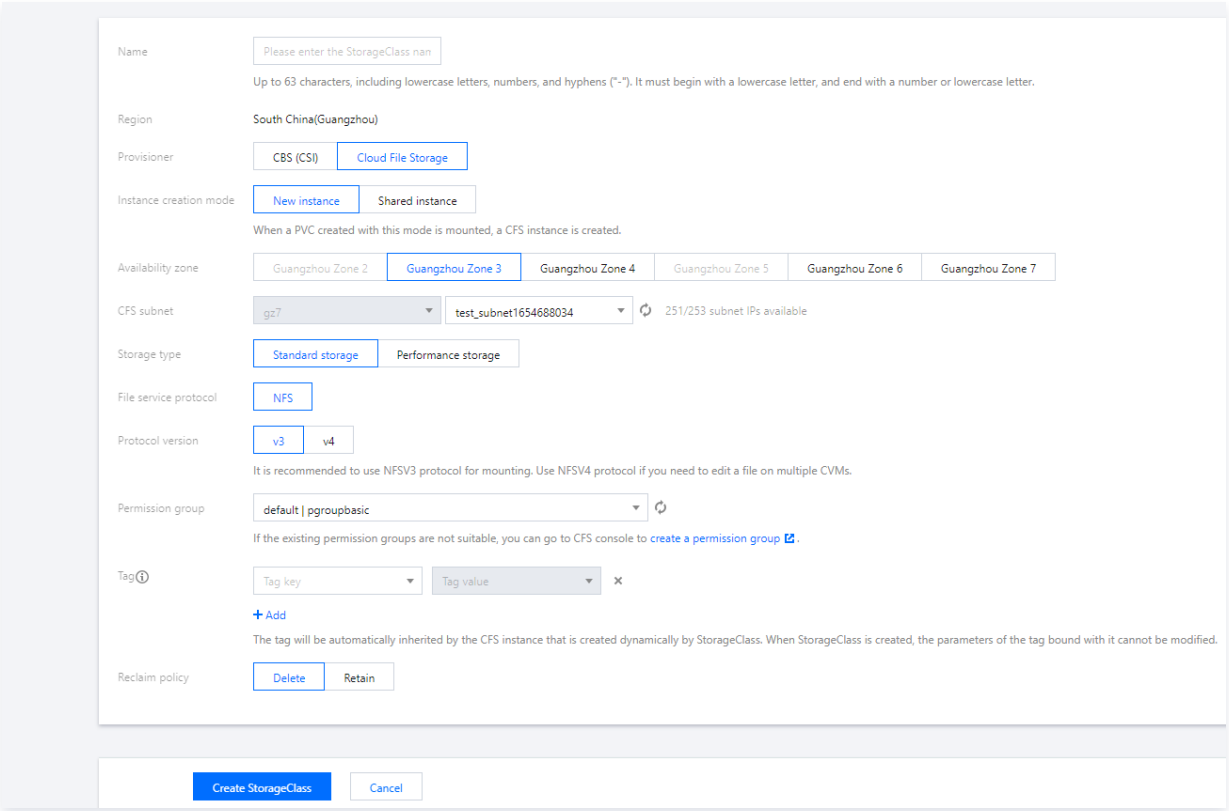
> **Note**
> For better stability, when you create a PV by YAML and use the NFSv3 protocol for mounting, you need to specify the FSID of the file system to be mounted.

Console Operation Guide

Creating a StorageClass

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 enter the cluster details page.
3. Choose **Storage > StorageClass** in the left sidebar to go to the "StorageClass" page.

4. Click **Create**. On the "Create StorageClass" page, configure the StorageClass parameters as shown below:



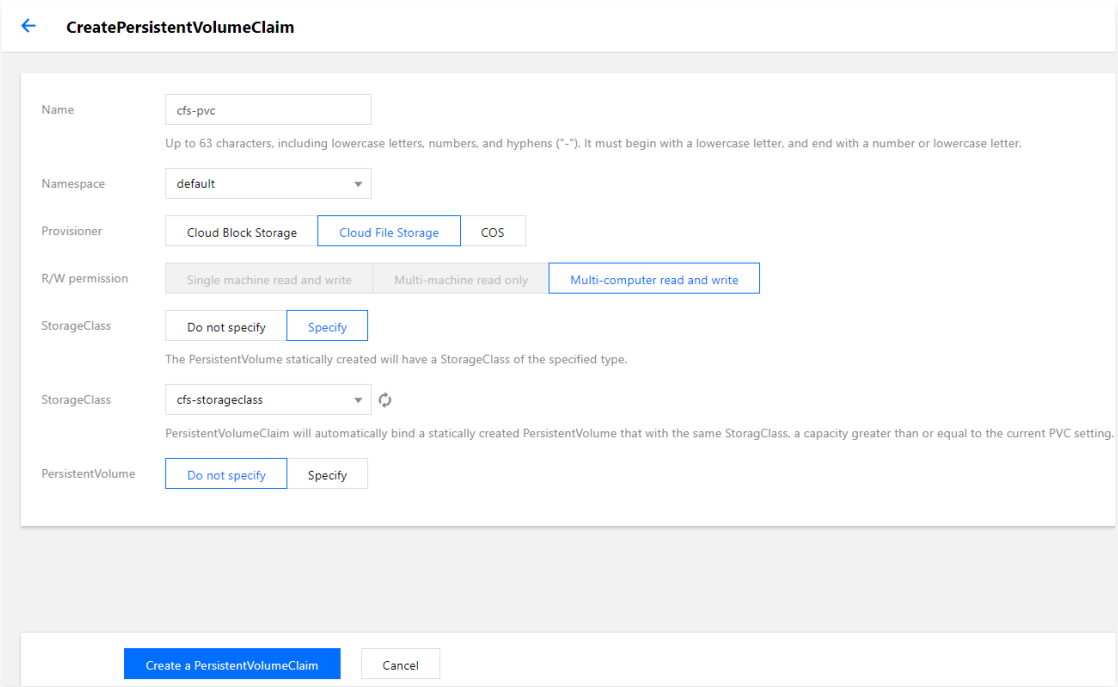
Configuration items	Description
Name	Enter the StorageClass name, for example, `cfs-storageclass`.
Regions	It is the region of the cluster by default.
Provisioner	**Provisioner** can be CBS (CSI) or Cloud File Storage. Here, Cloud File Storage is selected.
Instance creation mode	Offers two modes: **Create new instance** and **Shared instance**. • Create a new instance: By default, a CFS instance is created for each PVC when mounted. • Shared Instance: Each PVC will share different subdirectories of the same CFS instance when mounted. The shared CFS instance and subdirectories are automatically created 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.
Availability Zones	Select an Availability Zone (AZ) within the current region that supports Cloud File Storage. The storage types available in different AZs within each region may vary. Please refer to Supported Regions for selection.
CFS subnet	Set the subnet range of the CFS in the current AZ.
Storage Type	Cloud File Storage offers two types of file systems: **Standard Storage** and **Performance Storage**. The available storage types may vary across different availability zones within a region. Please make your selection based on the actual situation in the console. • Standard Storage: Low-cost, high-capacity storage suitable for cost-sensitive and large-capacity businesses, such as data backup, file sharing, and log storage scenarios.

	• Performance storage: High throughput and high IOPS, suitable for I/O-intensive workloads. Examples include high-performance computing, media rendering, machine learning, DevOps, and office automation scenarios.
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 to manage the access and read/write permissions of clients within the same network. Please choose a suitable permission group based on your requirements. If none are available, go to the Permission Group page to create one.
Reclaim policy	There are two reclaim policies available: **Delete** and **Retain**. For data security purposes, it is recommended to use the **Retain** policy. • Deletion: When a PVC is destroyed, the PV and storage instance bound to it will also be automatically destroyed if the PV was dynamically created using the PVC. • Retain: When a PVC is deleted, the PV and storage instance bound to it will be retained.
Tag	Select the cloud tag to be bound to the CFS instance. The tag will be automatically inherited by the CFS instance created dynamically by StorageClass. The tag parameters bound to the StorageClass cannot be modified after creation. If the existing tags do not meet your requirements, please go to the Tag Console to make changes.

5. Click **Create 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, select **Storage** > **PersistentVolumeClaim** on the left sidebar.
3. Click **Create** to enter the "New PersistentVolumeClaim" page, and configure the key PVC parameters, as shown below:



Configuration items	Description
Name	Enter the `PersistentVolumeClaim` name, for example, `cfs-pvc`.
Namespace	A namespace is used to assign cluster resources. Here, default is selected.
Provisioner	Select **Cloud File Storage**.

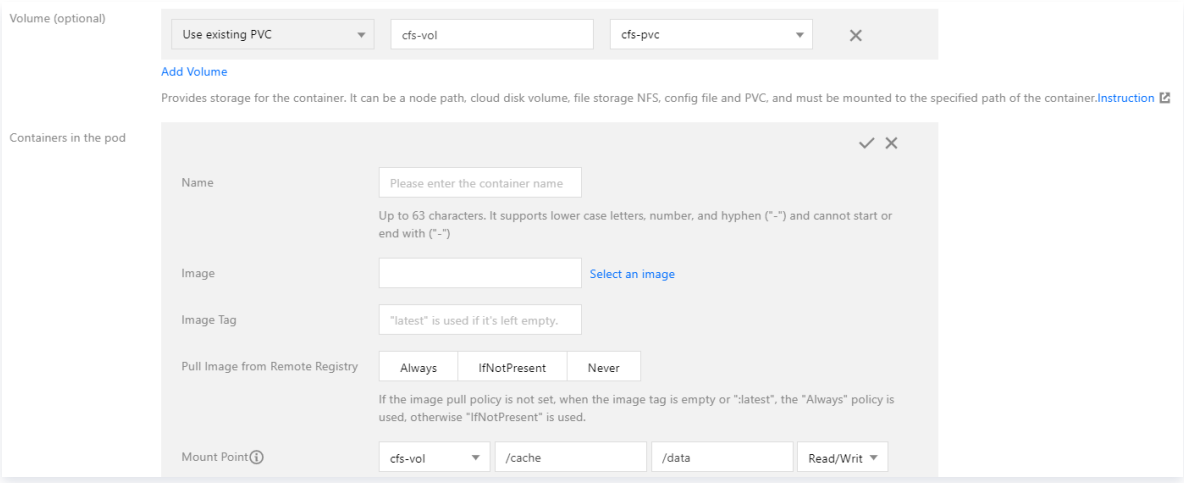
Read & Write Permissions	CFS only supports **Multi-computer read and write**.
StorageClass	Select a StorageClass as required. In this document, we use **Specifying StorageClass** as an example, with the cfs-storageclass created in the Creating StorageClass step. **Note:** • The PVC and PV will be bound to the same StorageClass. • **Not specifying StorageClass** implies that 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 the PersistentVolume as needed. In this document, we choose **Do not specify PersistentVolume**. **Note:** • The system first searches the current cluster to see whether there are PVs that meet the binding rules. If not, 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. • For a detailed introduction on **not specifying PersistentVolume**, please refer to Viewing 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. Select **Create** to enter the "Create Workload" page. Follow the instructions in Creating a Deployment to create a Deployment and refer to the information below for mounting data volumes, as shown in the following image:



   - **Volume (optional)**:
     - **Mount method**: Select "Use existing PVC".
     - **Volume name**: Set a custom name. This document uses `cfs-vol` as an example.
     - **Select PVC**: Choose the "cfs-pvc" that was created in the Create PVC step.
   - **Containers in the Pod**: Click **Add mount target** to set a mount target.
     - **Volume**: Select the "cfs-vol" volume that was 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 Guide

Creating a StorageClass

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cfs
parameters:
  pgroupid: pgroup-xxxxxxxx
  storagetype: SD
  #subdir-share: "true"
  vpcid: vpc-xxxxxxxx
  subnetid: subnet-xxxxxxxx
  vers: "3"
  resourcetags: ""
  zone: ap-guangzhou-3
provisioner: com.tencent.cloud.csi.cfs
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The following parameters can be configured:

Category	Required	Description
zone	Not required	It defines the location where the file is stored.
pgroupid	Not required	It defines the permission group for the file storage.
storagetype	Not required	By default, it is set to STANDARD storage (SD). The available values and descriptions are as follows: • SD: Standard Storage • HP: High-performance storage.
subdir-share	Supported	It indicates the shared instance mode for instance creation by StorageClass.
vpcid	Supported	It indicates the ID of the VPC where the file is stored.
subnetid	Supported	It indicates the ID of the subnet where the file is stored.
vers	Supported	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	Supported	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 commas. 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  # No need to fill in for dynamic creation; for static creation, specify the PV instance ID in this field.
```

Category	Required	Description
spec.accessModes	Not required	The cfs storage supports Multiple-Read-Multiple-Write.
spec.resources.requests.storage	Supported	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.

Use Cloud Disk CBS
CBS Instructions

Last updated: 2023-09-26 14:51:08

Scenario

Tencent Kubernetes Engine (TKE) supports using Cloud Block Storage (CBS) by creating PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs) and mounting volumes to workloads. This document describes how to mount a CBS volume to a workload in a cluster 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

Instructions

Dynamically creating a cloud 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 detailed operation steps, please refer to Managing CBS Templates with 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 PVC.
   For detailed operation steps, please refer to Managing CBS with PV and PVC.

Managing CBS Templates by Using a StorageClass

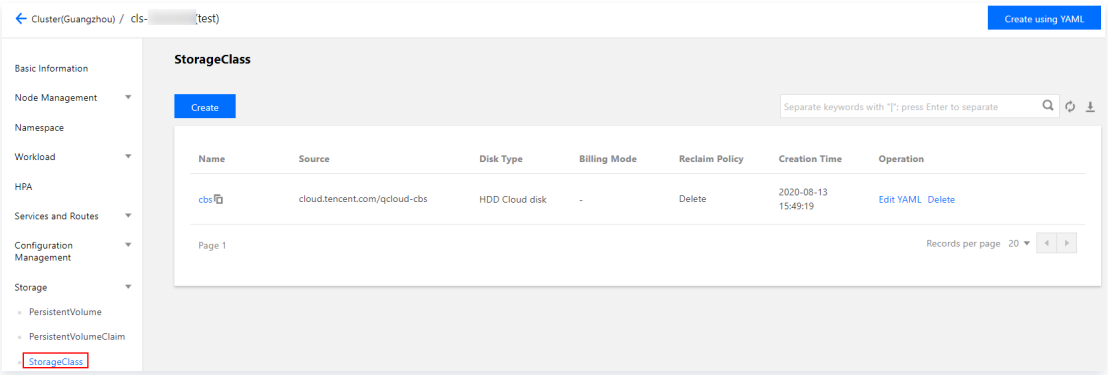
Last updated: 2023-09-26 14:51:40

A cluster admin can use a StorageClass to define different storage classes for 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 type by using the console or kubectl, and how to customize the template required by cloud disks.

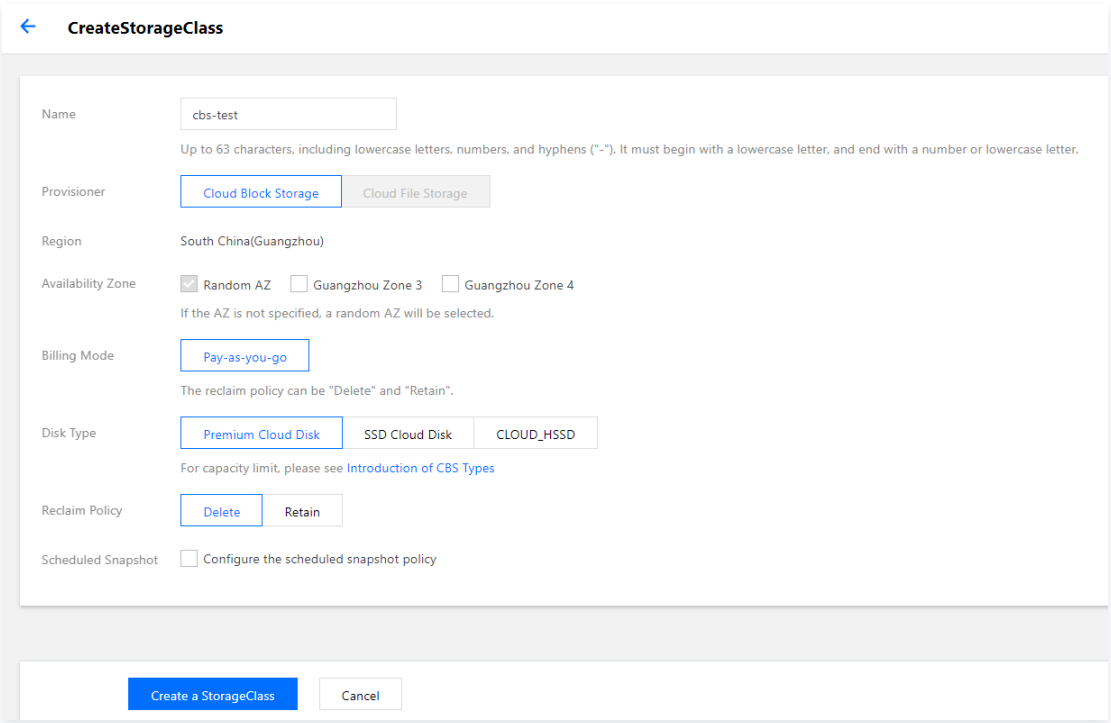
Console Operation Guide

Creating a StorageClass

1. Log in to the TKE console and select **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. Choose **Storage** > **StorageClass** in the left sidebar, as shown in the following figure:



4. Click **Create** to enter the "Create StorageClass" page. Set the parameters as required, as shown below:



The main parameter information is as follows:

- **Name**: set a custom name. This document uses `cbs-test` as an example.
- **Region**: the region where current cluster is located.
- **Provisioner**: Select **CBS (CSI)**.
- **Availability zone**: Select the availability zones that support CBS disks in the current region as required.

- **Billing Mode**: Two billing modes are available: **Pay-as-you-go** and **Monthly subscription**. Different billing modes support different reclaim policies. Please refer to the following information for selection:
  - **Pay-as-you-go**: A flexible billing mode that allows you to create and terminate instances at any time, and pay based on the actual usage of the instances. Supports both deletion and retention as the reclaim policy.
  - **Monthly Subscription**: A prepaid billing mode where you pay the storage fees for one month in advance, with support for automatic monthly renewal. Only the Retain reclaim policy is supported.

    > **Note**
    > - If you need to purchase a monthly or yearly cloud disk, go to the Roles page and add the `QcloudCVMFinanceAccess` policy to the `TKE_QCSRole` role to configure payment permissions. Otherwise, creating a PVC based on a monthly or yearly StorageClass may fail due to payment permission issues.
    > - Only cloud disks with a monthly subscription billing mode can be renewed, and the auto-renewal feature is set to renew monthly by default. Users can go to the details page of the created PVC to enable or disable the auto-renewal feature. For more billing information, please refer to Cloud Disk Billing Issues.

- **Disk type**: **Premium Cloud Disk**, **SSD Cloud Disk**, **Enhanced SSD Cloud Disk**, and **Balanced SSD** 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.
  - **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.

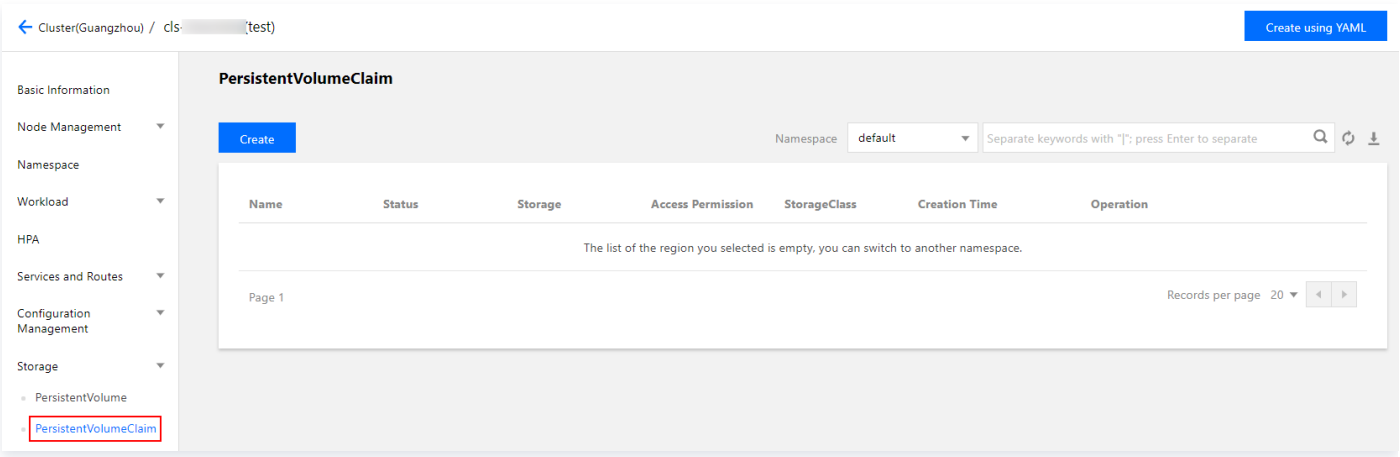
    > **Note**
    > The default-policy configuration provided by TKE for backup includes the date of backup execution, time point of backup execution, and backup retention period.

- **Online expansion**: Specify whether to enable online expansion. For more information about online expansion, see Online Expansion of Cloud Disk.
- **Reclaim Policy**: The reclaim policy for cloud disks, usually offering **Delete** and **Retain** options. The specific choice depends on the selected billing mode. For data security purposes, it is recommended to use the Retain reclaim policy.

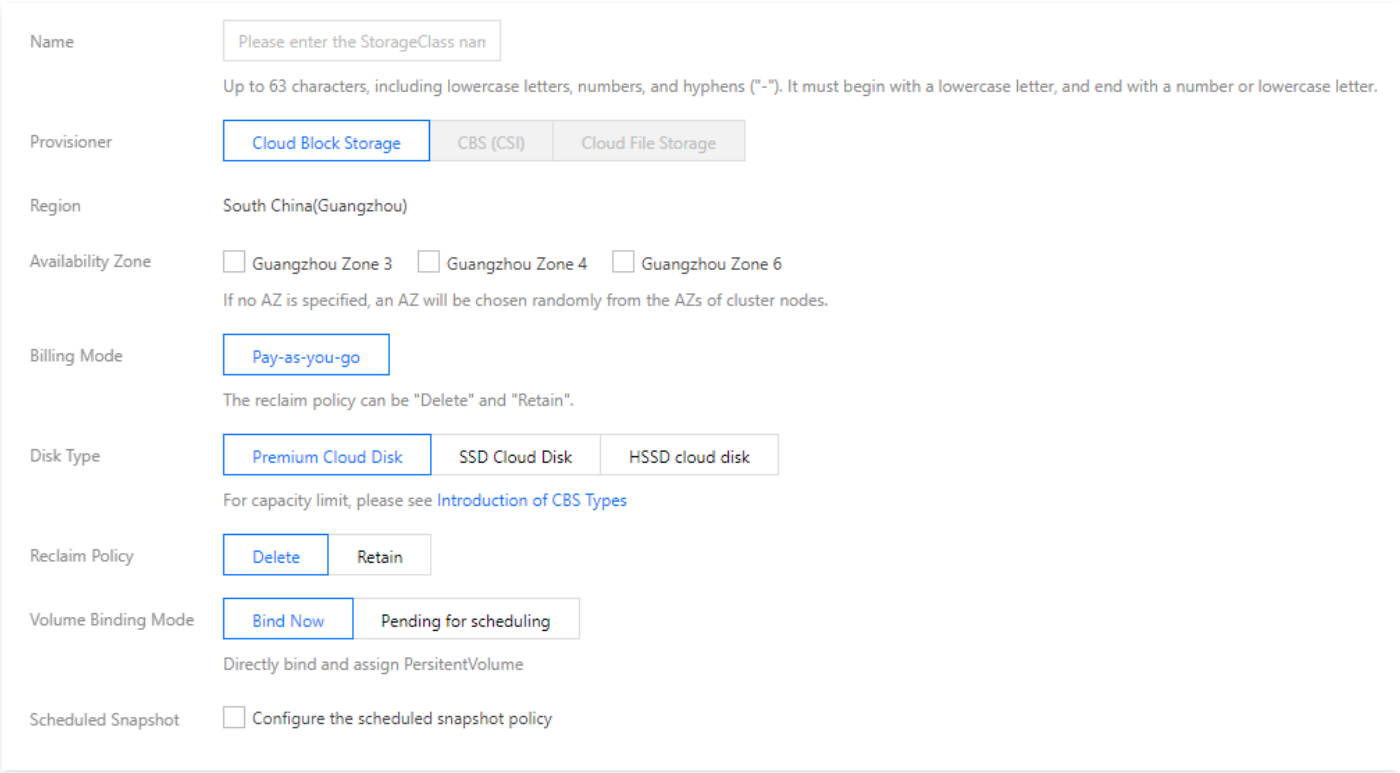
5. Click **Create a StorageClass** to complete the process.

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. In the cluster details page, select **Storage** > **PersistentVolumeClaim** from the left menu to access the "PersistentVolumeClaim" information page, as shown below:



3. Click **Create** to enter the "Create PersistentVolumeClaim" page, and set the key PVC parameters as required, as shown below:



The main parameter information is as follows:

- **Name**: set a custom name. This document uses `cbs-pvc` as an example.
- **Namespace**: Select `default`.
- **Provisioner**: Select **CBS (CSI)**.
- **R/W permission**: Cloud disks only support single-machine read and write.

> **Note**
> - **Single machine read and write**: Currently, CBS disks can only be mounted to the same machine, and therefore, CBS supports only data read and write processing on a single machine.
> - **Multi computer read and write**: CFS/COS objects can be mounted to multiple machines, and therefore, CFS/COS supports data read and write processing on multiple machines.

- **StorageClass**: Specify the StorageClass as needed. In this document, we use cbs-test created in the `Creating a StorageClass` step as an example.

> **Note**

- The PVC and PV will be bound to the same StorageClass.
- Not specifying a StorageClass means that the value of the 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.

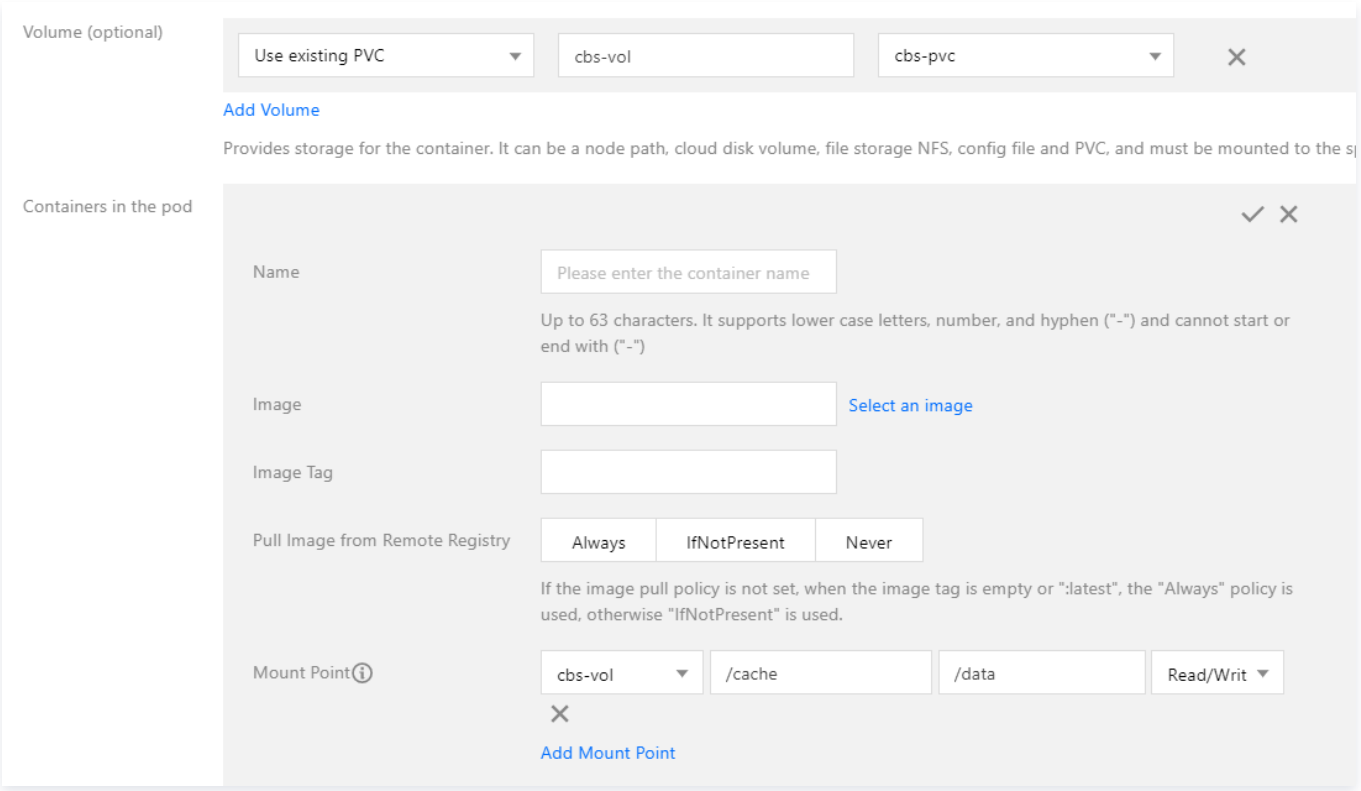
- **Disk Type**: based on the selected StorageClass, the available disk types are displayed: **Premium Cloud Disk**, **SSD Cloud Disk** and **Enhanced SSD Cloud Disk**.
- **Capacity**: when PersistentVolume is not specified, you need to indicate the desired capacity of the cloud disk. The capacity must 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 target cluster details page, select **Workload** > **StatefulSet** in the left sidebar to go to the StatefulSet page.
2. Click **Create** to enter the "Create Workload" page. Follow the Creating a StatefulSet guide and refer to the information below for mounting data volumes, as shown in the following image:



- **Volume (optional)**:

- **Mount method**: Select "Use existing PVC".
- **Volume name**: set a custom name. This document uses `cbs-vol` as an example.
- **Select PVC**: Choose an existing PVC. This document uses cbs-pvc, which was created in the step of `Creating a PVC with a specified StorageClass`, as an example.

- **Containers in the Pod**: Click **Add mount target** to set a mount target.
  - **Data Volume**: Select the data volume "cbs-vol" that has been 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 Guide

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. When creating a PVC without specifying a type, this type will be used automatically.
  name: cloud-premium

For TKE clusters with the CBS-CSI component installed, set the provisioner to com.tencent.cloud.csi.cbs.
# If the CBS-CSI component is not installed, set the provisioner to cloud.tencent.com/qcloud-cbs (this feature is deprecated in version 1.20 and later)
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.

Category	Description
type	Including CLOUD_PREMIUM (Premium Cloud Disk), CLOUD_SSD (SSD Cloud Disk), and CLOUD_HSSD (Enhanced SSD Cloud Disk).
zone	Used to specify the availability zone. If specified, the cloud disk will be created in this availability zone. If not specified, the availability zone information of all nodes will be fetched and randomly selected. For Tencent Cloud region identifiers, please refer to Regions and Availability Zones.
paymode	The default billing mode for cloud disks is POSTPAID_BY_HOUR, which is pay-as-you-go and supports both Retain and Delete policies. Retain is only effective in cluster versions higher than 1.8. Alternatively, you can set it to PREPAID mode, which is a monthly subscription and only supports the Retain policy.
volumeBindingMode	Volume binding modes include Immediate (bind immediately) and WaitForFirstConsumer (delayed

	scheduling).
reclaimPolicy	Reclaim policy supports Delete (removal) and Retain (retention).
renewflag	The renewal mode for cloud disks. The default mode is `NOTIFY_AND_MANUAL_RENEW`. • The `NOTIFY_AND_AUTO_RENEW` mode indicates that the created cloud disk supports expiration notifications and monthly auto-renewal. • The `NOTIFY_AND_MANUAL_RENEW` mode indicates that the created cloud disk supports expiration notifications but does not auto-renew. • The `DISABLE_NOTIFY_AND_MANUAL_RENEW` mode indicates that the created cloud disks will not notify about expiration and will not auto-renew.
aspid	Specify the snapshot ID to automatically bind the snapshot policy after creating the cloud disk. Binding failure will not affect the creation process.

Creating a Multi-Pod StatefulSet

You can use a cloud disk to create a multi-pod StatefulSet. The sample YAML file is as follows:

> **Note**
> 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.

```yaml
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:  # Automatically create PVC, which in turn creates PV automatically
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: cloud-premium
      resources:
        requests:
          storage: 10Gi
```

Managing CBS by Using PVs and PVCs

Last updated: 2023-09-26 14:51:50

Scenario

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.

Instructions

Console Operation Guide

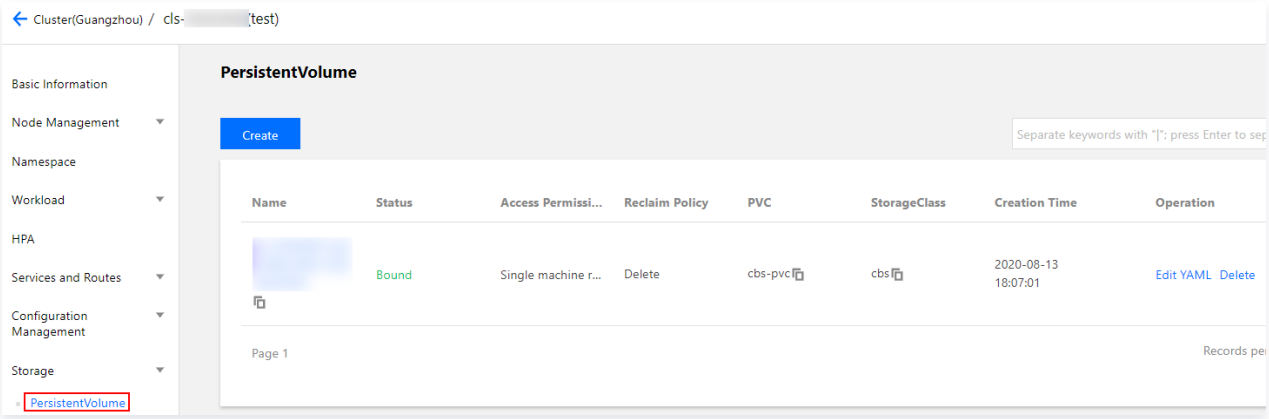
Creating a StorageClass via the console

To statically create a PV of the CBS type, you need to bind an available StorageClass of the same type. For more information, see Creating a StorageClass.

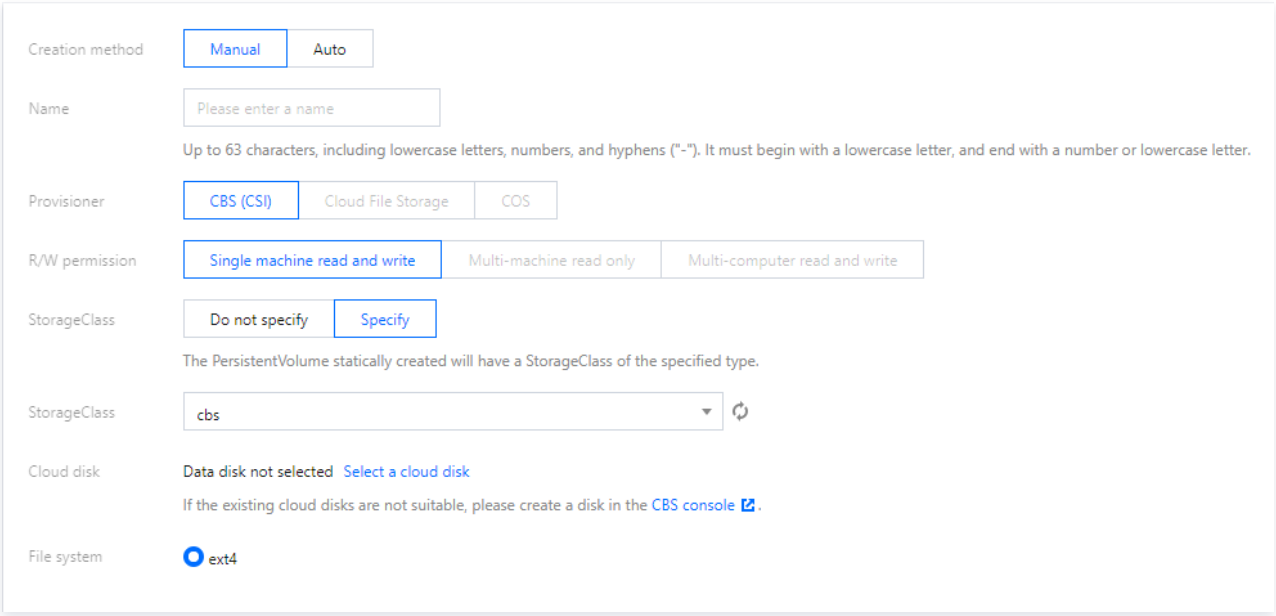
Creating a PV manually

Note
Creating a PV statically is suitable for scenarios where there're 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. Select **Storage** > **PersistentVolume** from the left menu to access the "PersistentVolume" page, as shown below:



4. Click **Create** to enter the "Create PersistentVolume" page, and set the parameters as required, as shown below:



The main parameter information is 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**: Cloud 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.

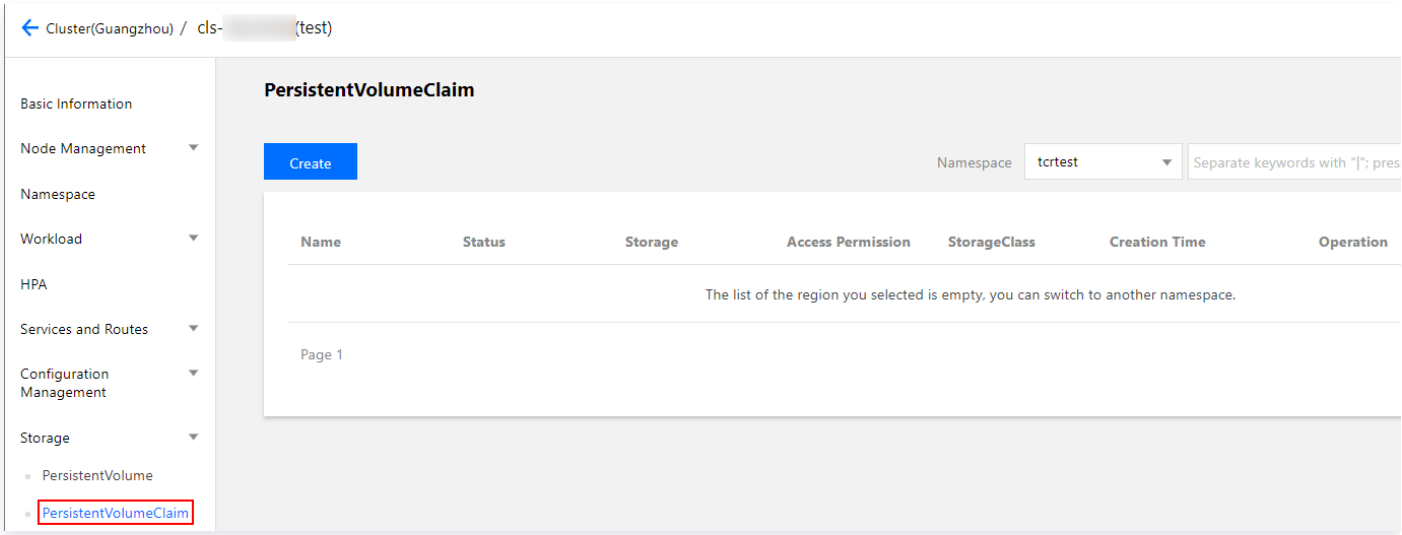
  > **Note**
  > - The PVC and PV will be bound to the same StorageClass.
  > - Not specifying means that the StorageClass value for the corresponding PV is empty, and the `storageClassName` field in the YAML file has an empty string value.

- **Cloud Disk**: select a created cloud disk.
- **File System**: the default value is ext4.

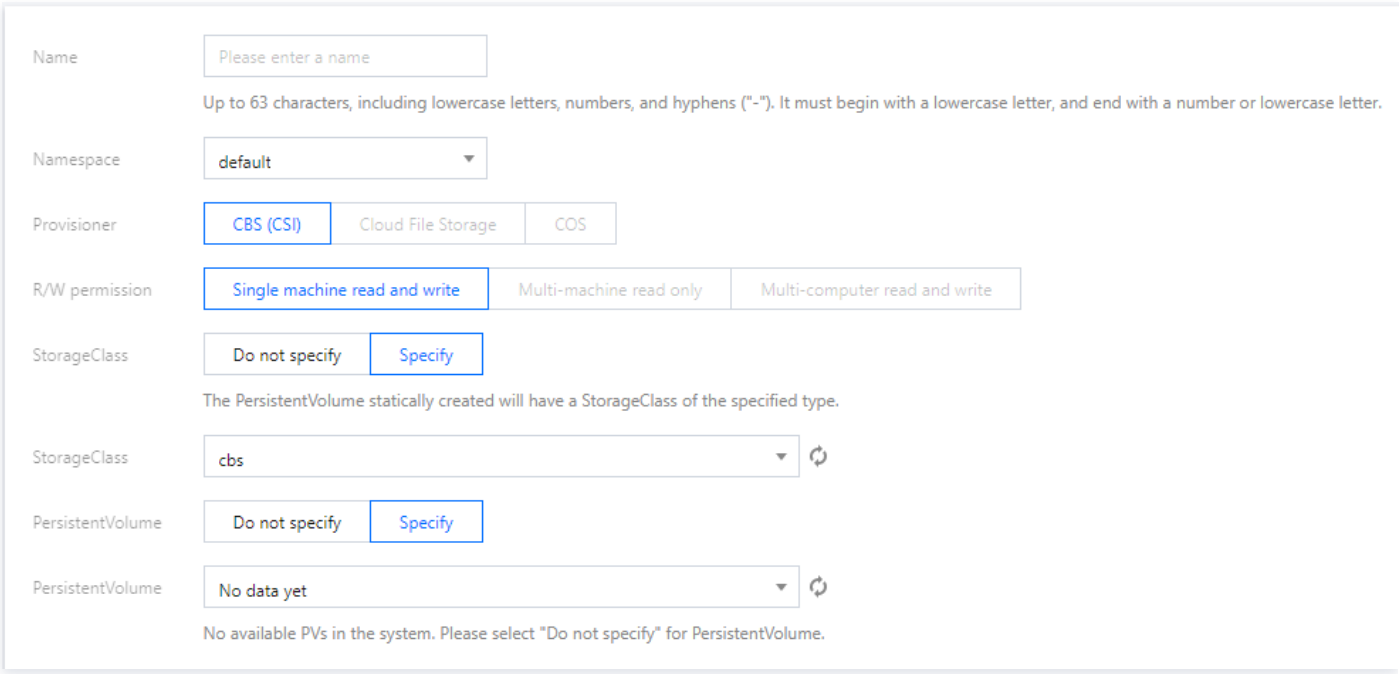
5. Click **Create a PersistentVolume** to complete the creation.

Creating a PVC

1. In the cluster details page, select **Storage** > **PersistentVolumeClaim** from the left menu to access the "PersistentVolumeClaim" page, as shown below:



2. Click **Create** to enter the "Create PersistentVolumeClaim" page, and set the parameters as required, as shown below:



The main parameter information is 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.
  > - Not specifying means that the StorageClass value for the PVC is empty, which corresponds to an empty string for the `storageClassName` field in the YAML file.

- **PersistentVolume**: Specify a PersistentVolume as required. This document uses cbs-pv, which you created in the step of `Static PV Creation`, as an example.

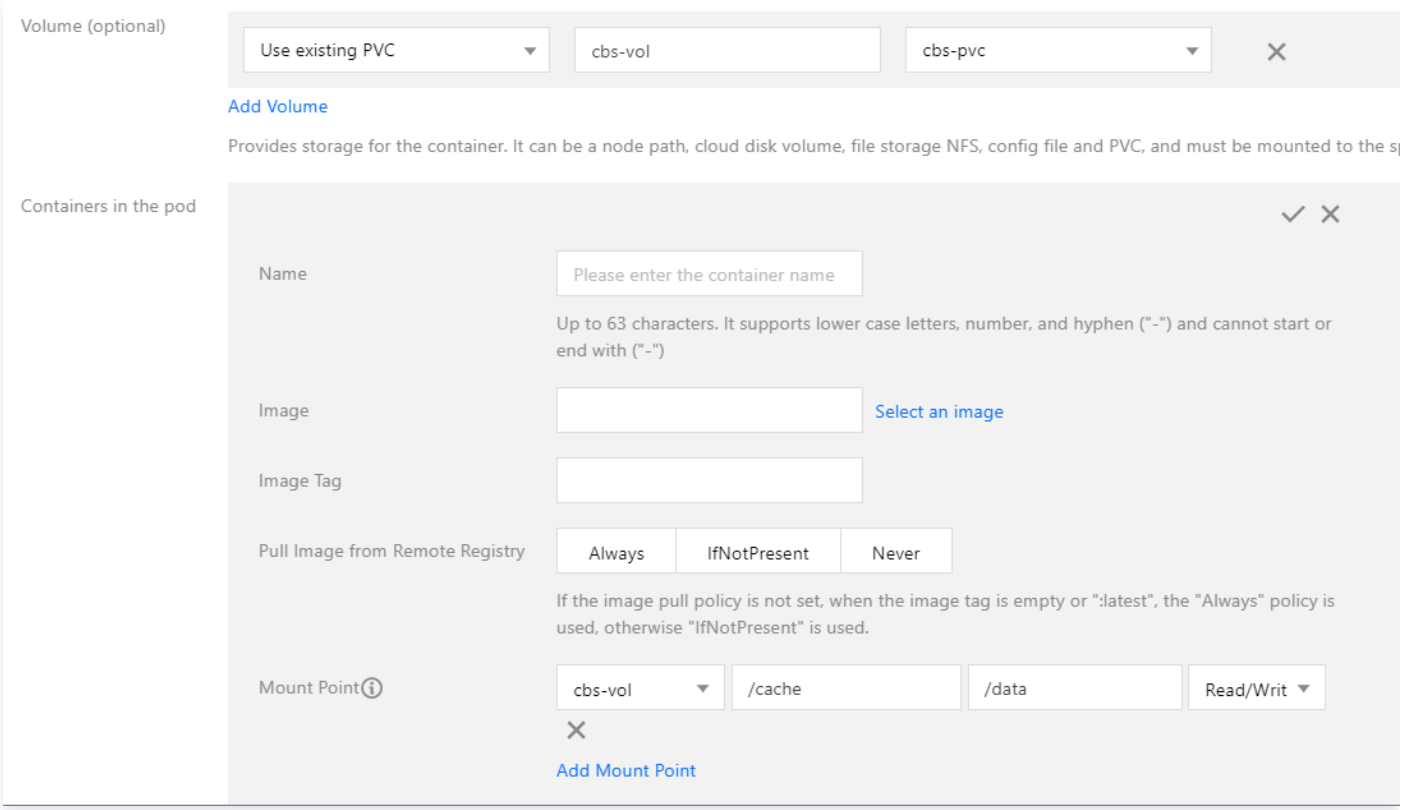
> **Note**
> - Only PVs in the specified StorageClass and in the Available or Released statuses can be selected. If no PV in the current cluster meets the conditions, select "No PersistVolume is specified".
> - If the status of the selected PV is Released, you need to manually delete the `claimRef` field 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, 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. Follow the instructions in Creating a Deployment to create a deployment, and refer to the information below to mount a volume, as shown in the following image:



   - **Volume (optional)**:
     - **Mount method**: Select "Use existing PVC".
     - **Volume name**: set a custom name. This document uses `cbs-vol` as an example.
     - **Select PVC**: Choose the "cbs-pvc" created in the Create PVC step.
   - **Containers in the Pod**: Click **Add mount target** to set a mount target.
     - **Volume**: Select the added volume "cbs-vol" 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 Guide

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: 2023-09-27 09:30:29

Feature Overview

Volume type

Volume type	Description
Use temporary path	/
Use the server 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 fill in the NFS path. You can use Tencent Cloud's Cloud File Storage (CFS) or your own NFS storage. Using NFS data volumes is suitable for persistent storage with multiple reads and writes, and is also suitable for scenarios such as big data analysis, media processing, and content management.
Use existing PersistentVolumeClaim	Utilize existing PersistentVolumeClaim declarations for workload storage, automatically allocating or creating new PersistentVolumes to mount under corresponding Pods. This is primarily suitable for stateful applications created by StatefulSet.
Using ConfigMap	ConfigMap is mounted to the Pod in the form of a file system, supporting custom ConfigMap entries to be mounted to specific paths. For more details, please refer to ConfigMap Management.
Using a Secret	Secret is mounted to the Pod as a file system, supporting custom Secret entries to be mounted to specific paths. For more information, see Secret Management.

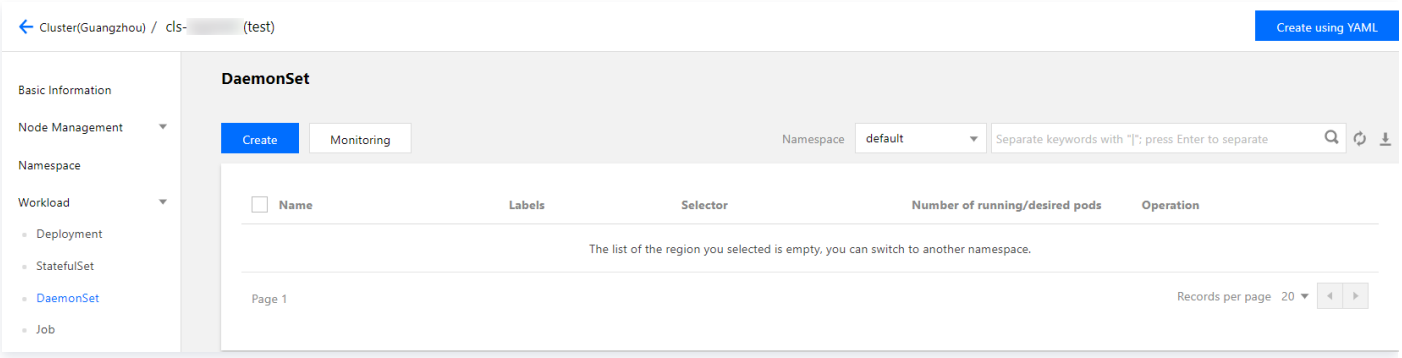
Notes on Volumes

- After creating a volume, you need to set the container mount point in the **Containers in the Pod** module.
- Under the same service, the name of the volume and the mount point set for the container 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 volume using the temporary directory is the same as that of the Pod.
- If no permission is set for volume mount, the default setting will be read/write permission.

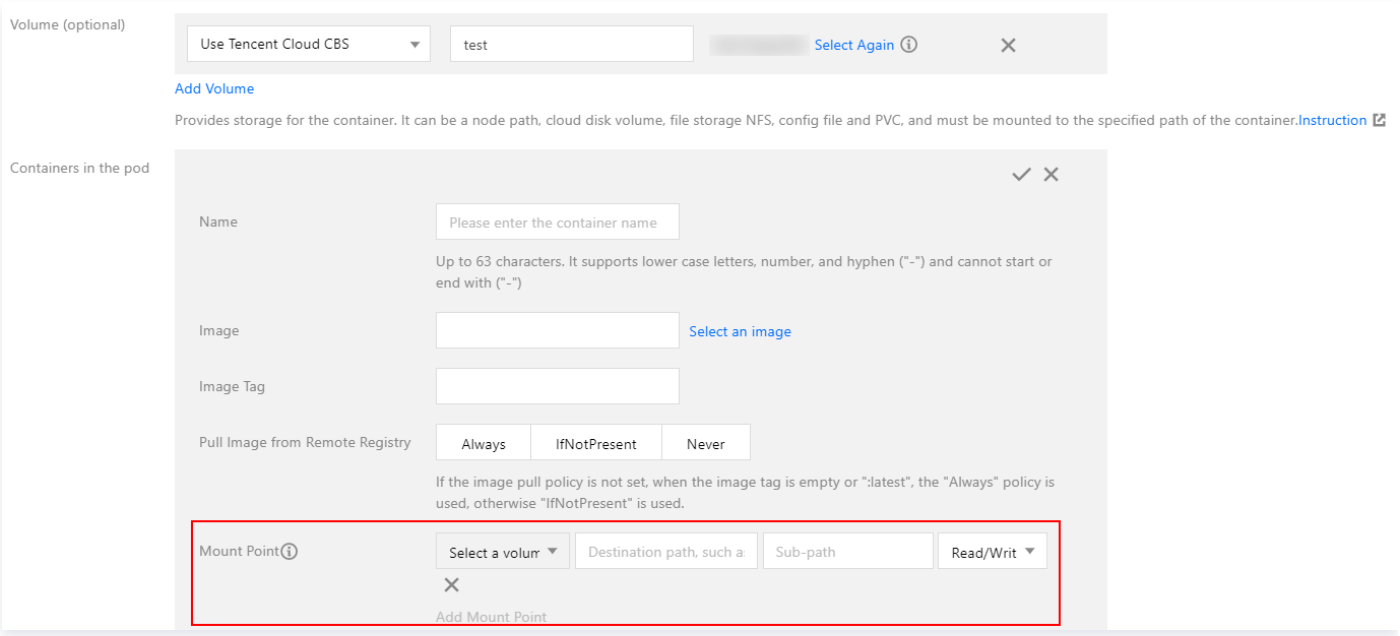
Operation Guide for Volumes in the Console

Creating a Workload to Mount a Volume

1. Log in to the Container Service Console and select Cluster 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 any Workload type to access its information page.
   For example, choose **Workload** > **DaemonSet** to enter the DaemonSet information page, as shown in the figure below:



4. Click **Create** to enter the "New Workload" page.
5. Based on the information provided on the page, set the workload name, namespace, and other details. In the "Volumes" section, click **Add Volume** to add a volume.
6. Based on your actual requirements, choose the storage method for the data volume. This article uses **Tencent Cloud Disk** as an example.
7. In the **Mount Points** section of "Containers in the Pod", configure the mount points as shown below:
   You can configure the mount points only after selecting **Add Volume** in Step 5.



8. Configure the remaining options as needed, and click **Create Workload** to complete the creation.

Configurations for mounting different volumes

The table below shows the usage details of different volume types. **When creating a workload and selecting "Add Volume"**, refer to the following content to add a volume and set the mount point:

Data Volume			Mount Point		
Local Disk Types	Name	Others	Destination Path	Sub-path	Read & Write Permissions
Temporary path	Custom	/	Please fill in according to your actual requirements. Example: `/cache`.	Mount only a sub-path or a single file in the selected volume, such as `/data` or `/test.txt`.	Select one as needed. • Read-only: Only allows reading the data volume of the container path; data
Host path		Configure the server path. • Server path: This path cannot be empty. For example, when the container needs to access Docker, the server path can be set to `/var/lib/docker`.			

		• Check type: TKE offers various check types such as NoChecks, DirectoryOrCreate, etc. Please carefully review the descriptions of each type on the console and select according to your actual requirements.			modification is only permitted on the host machine. • Read-write: Allows reading and saving modifications to the data volume at the container path.
NFS disk		NFS Path: Enter the CFS file system or custom NFS address. • To create a file system, please refer to Create a File System and Mount Target. • NFS path example: 10.0.0.161:/. This path can be obtained by logging into the File System Console, clicking on the target file system ID, and retrieving it from the "Linux Mount Directory" in the **Mount Point Information** tab.			
Existing PVC		Select PVC: Choose the PVC based on your actual needs.			
ConfigMap		• Select ConfigMap: Choose based on your actual requirements. • Options: Provide two choices – "All" and "Specify Part of the Key". • Items: When selecting the "Specify certain keys" option, you can add an item to mount to a specific path. For example, if the mount point is `/data/config` and the sub-path is `dev`, the final storage location will be `/data/config/dev`.			
Secret					

Using kubectl to Manipulate Volumes

The following is a sample where which you can perform creation directly using kubectl.

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 other 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 target of the volume

PV and PVC binding rules

Last updated: 2023-09-26 15:05:08

PV Status Introduction

PV Status	Description
Avaliable	When a created PV is not bound with a PVC, the PV is in the `Available` status.
Bound	Upon binding with a PVC, the PV transitions to the `Bound` state.
Released	A PV with a `Retain` reclaim policy will transition from the `Bound` state to the `Released` state when its bound PVC is deleted. **Note:** To successfully bind a `Released` state PV with a PVC, you must manually delete the `claimRef` field in the YAML configuration file.

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

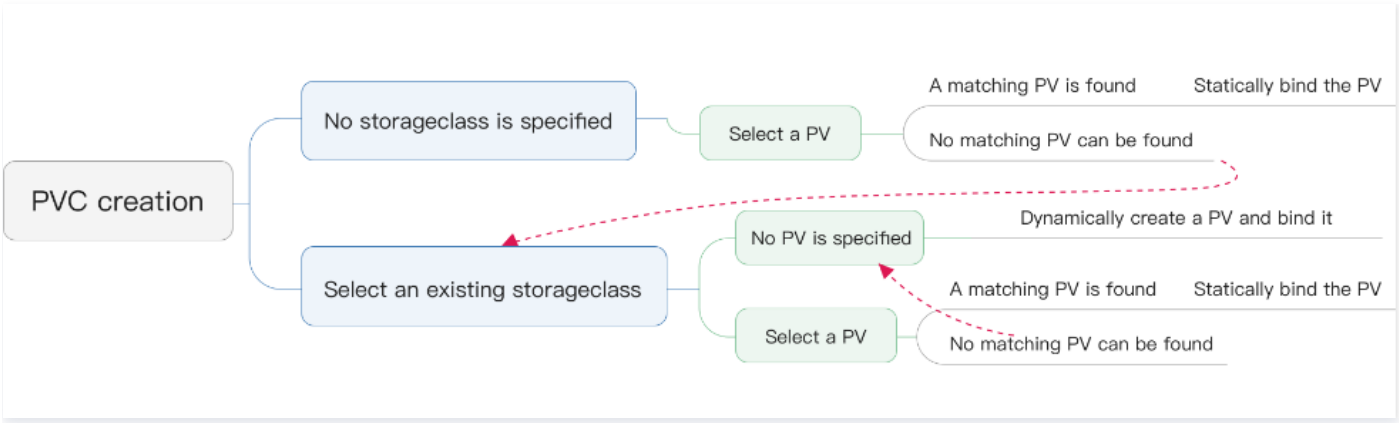
When binding a PVC with a PV, consider the following parameters to check whether PVs that meet the binding rules are available in the current cluster.

Category	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

In Tencent Kubernetes Engine (TKE) platform operations, the relationship between StorageClass selection and PV/PVC binding is illustrated in the following diagram:



Add-on Management
Add-on Overview

Last updated: 2023-09-26 15:09:32

Add-ons are extended feature packages provided by Tencent Cloud Container Service (TKE), allowing you to deploy the desired add-ons based on your business requirements. These add-ons assist in managing Kubernetes components within your cluster, including component deployment, upgrades, configuration updates, and uninstallation.

Add-on Types

Add-ons are categorized into two types: Basic Components and Enhanced Components.

Basic add-on

Basic components are software packages that TKE relies on for its functionality. Examples include the load balancing components Service-controller, CLB-ingress-controller, and the container network plugin tke-cni-agent.

Note

- Upgrades and configuration management of basic components are centrally managed and maintained by TKE. It is not recommended to modify these basic components.
- Updates and releases for basic components will be notified through channels such as email and SMS.

Enhanced add-on

Enhanced components are optional add-ons provided by TKE, which can be deployed to access enhanced features supported by TKE. The types of enhanced components are as follows:

Component name	Overview	Component Description
OOMGuard OOM Daemon	Monitor	This add-on reduces the likelihood of various kernel failures caused by cgroup memory reclaim failure in user mode.
NodeProblemDetectorPlus Node Anomaly Detection Plus	Monitor	This component can detect various anomalies on nodes in real-time and report the results to kube-apiserver.
NodeLocalDNSCache Local DNS Cache Add-on	DNS	This add-on improves cluster DNS performance by running a DNS cache proxy as a DaemonSet on the cluster nodes.
DNSAutoscaler DNS Horizontal Scaling Add-on	DNS	This component obtains the number of nodes and cores in the cluster through a deployment and automatically scales the number of DNS replicas horizontally based on the preset scaling policy.
COS-CSI Tencent Cloud COS	Storage	This component implements the CSI interface, assisting container clusters in utilizing Tencent Cloud Object Storage.
CFS-CSI Tencent Cloud CFS	Storage	This component implements the CSI interface, assisting container clusters in utilizing Tencent Cloud File Storage.
CBS-CSI Tencent Cloud Block Storage (CBS)	Storage	This component implements the CSI interface, enabling TKE clusters to quickly select storage types through the console and create corresponding block storage cloud disk types of PersistentVolumes (PVs) and PersistentVolumeClaims (PVCs).
TCR	Video	This add-on automatically configures the private network domain resolution and dedicated

TCR plugin	flipping	access credentials for the specified TCR instance within the cluster, enabling secret-free pulling of container images through the private network.
P2P Accelerated distribution of container images	Video flipping	This component is based on P2P technology and can be applied to large-scale TKE clusters for rapid retrieval of GB-level container images, supporting concurrent pulling by thousands of nodes.
Cerberus Image signature verification add-on	Video flipping	This add-on performs signature verification on container images in the TCR repository, ensuring that only images signed by trusted authorizers are deployed, thereby reducing the risk of running unexpected or malicious code.
Dynamic Scheduler Dynamic Scheduling Component	scheduling	Dynamic Scheduler is a dynamic scheduling plugin developed by Tencent Cloud Container Service (TKE) based on the native Kubernetes Kube-scheduler Extender mechanism. It performs pre-selection and prioritization based on the actual node loads. After installing this component, it effectively mitigates the issue of uneven node loads caused by the native scheduler's request and limit scheduling mechanism.
Descheduler Rescheduler Component	scheduling	Upon installing this plugin in a TKE cluster, it works in tandem with Kube-scheduler to monitor high-load nodes in real-time and evict low-priority Pods. It is recommended to use this plugin in conjunction with the TKE Dynamic Scheduler (dynamic scheduling add-on) to ensure multi-dimensional load balancing within the cluster.
NetworkPolicy Controller Network Policy Controller Add-on	Others	Network Policy is a resource provided by Kubernetes, and this add-on offers a Controller implementation for this resource.
Nginx-Ingress Community Ingress Component	Others	Nginx can be utilized as a reverse proxy, load balancer, and HTTP cache. The Nginx-ingress add-on is a Kubernetes Ingress controller that employs Nginx as a reverse proxy and load balancer.
OLM Operator Lifecycle Management	Others	OLM (Operator Lifecycle Manager), as a part of the Operator Framework, assists users in automatically installing, upgrading, and managing the lifecycle of Operators.
HPC Scheduled Replica Count Modification	Others	HorizontalPodCronscaler (HPC) is a custom-developed component that allows for scheduled modifications to the replica count of K8s workloads. Used in conjunction with HPC CRD, it supports time-based tasks with a minimum granularity of seconds.
QoSAgent Description	Others	QoS Agent is an add-on provided by Tencent Cloud, designed to enhance service quality. It offers a wide range of capabilities, ensuring quality stability while increasing the utilization of cluster resources.
CraneScheduler Description	Others	The scheduler dedicated for native nodes is a scheduler plugin developed by Tencent Cloud Container Service (TKE) based on the Kubernetes native Kube-scheduler Extender mechanism. It can virtually scale up node capacity to address issues where node resources are fully occupied but have low utilization rates.
Craned Description	Others	Request Intelligent Recommendation can suggest Request/Limit values for container-level resources in Kubernetes workloads, reducing resource wastage.
PodIdentityWebhook Description	Others	Leverage Secrets Manager (SSM) and Tencent Cloud Access Management (CAM) to streamline the process of accessing Tencent Cloud databases, thereby eliminating security risks associated with verifying database usernames and keys. Additionally, the periodic rotation of access credentials by Secrets Manager (SSM) indirectly alleviates the burden of manual operations.

Add-on Lifecycle Management

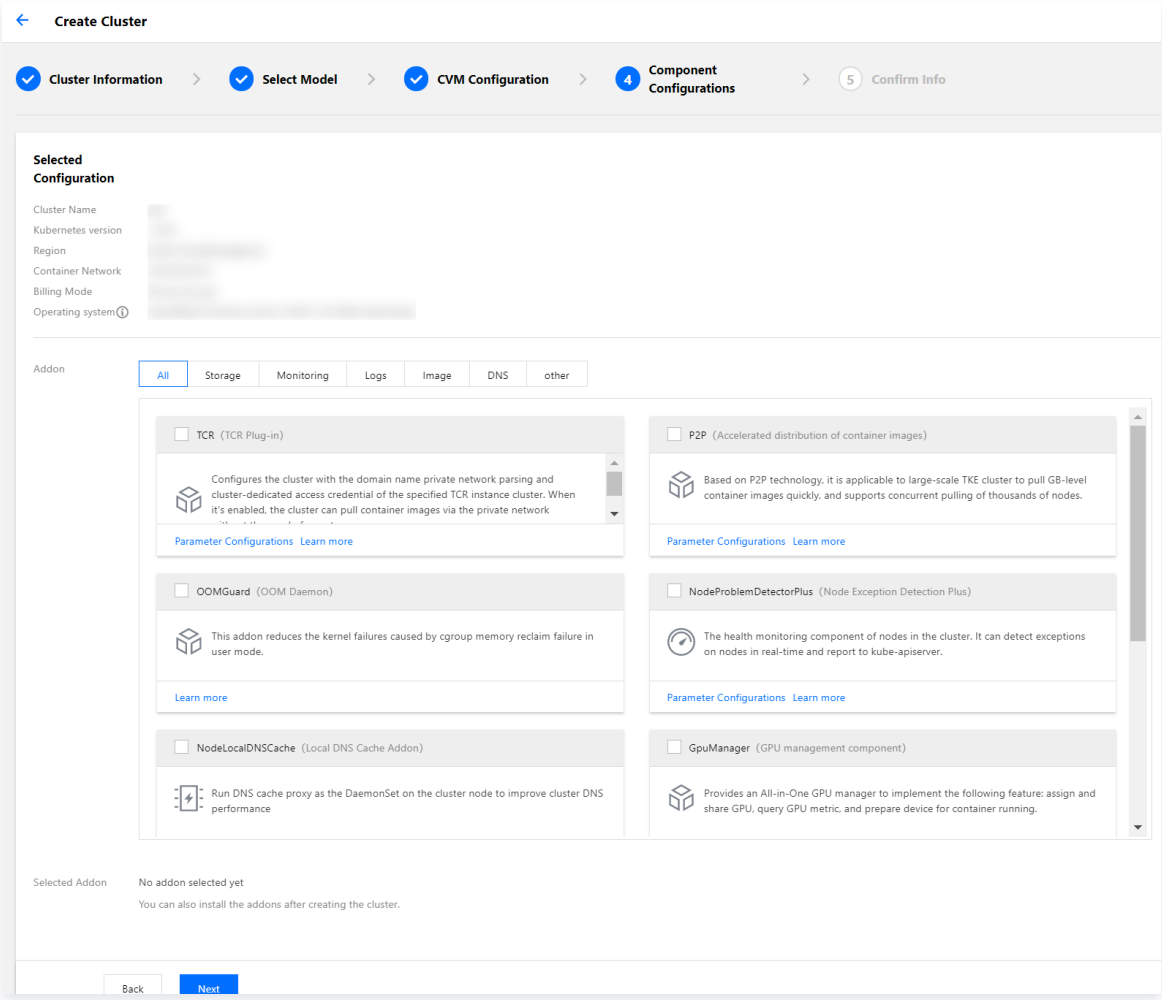
Last updated: 2023-09-26 15:09:50

Component Installation

You can install add-ons either through the cluster creation page or through the add-on management page.

Installing via 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 Cluster" page, fill in the cluster's **Cluster Information**, **Select Instance Type**, **CVM Configuration**, and **Add-on Configuration** in sequence, as shown in the figure below:



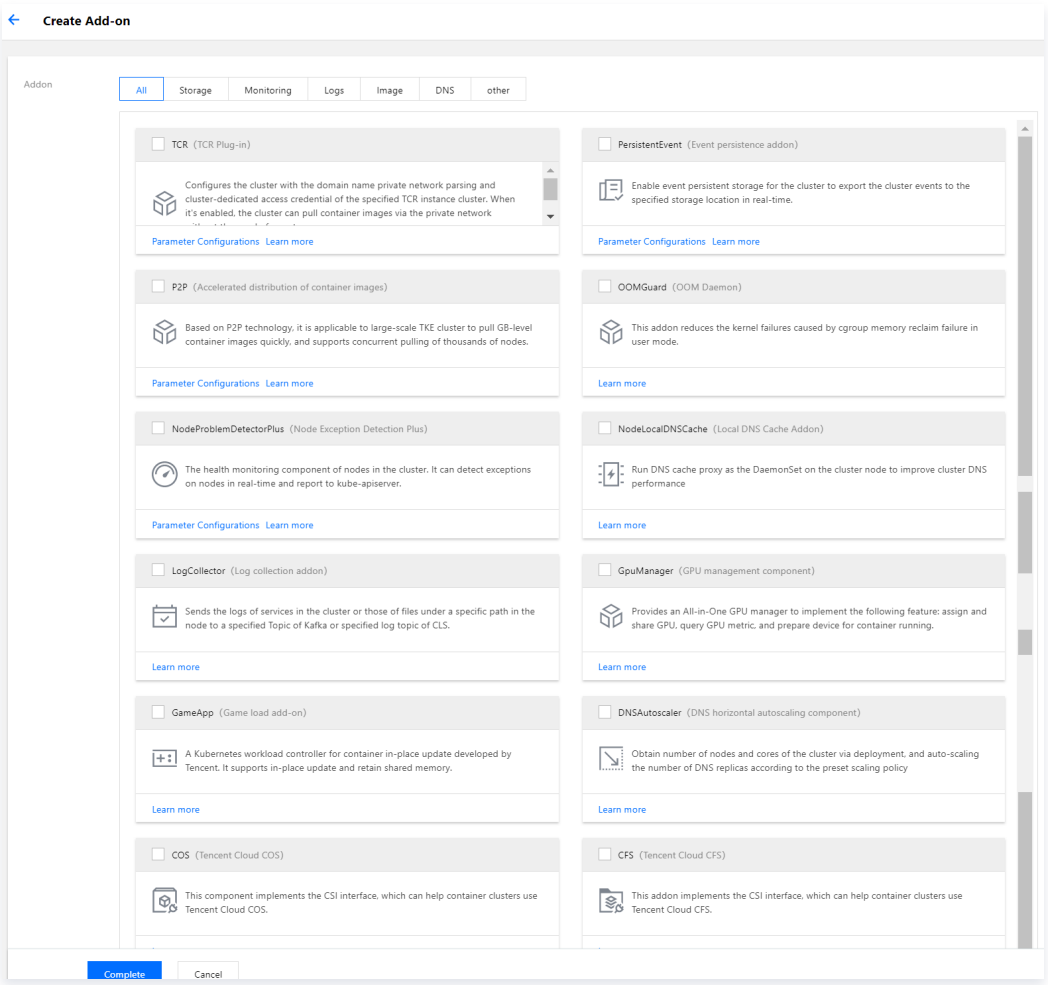
You can choose the appropriate add-ons to install based on your business deployment needs. Click **View Details** on each add-on card to see its introduction. Some add-ons require you to complete **Parameter Configuration** first.

> **Note**
> - Component installation is a non-critical path for cluster creation; failure to install will not affect the creation of the cluster.
> - Component installation requires a certain amount of cluster resources. Different components have different resource usage. Click **View Details** to see detailed information for each component.

4. Click **Next** to review and confirm the cluster configuration details.
5. Click **Done** to complete the creation.

Installing via the add-on management page

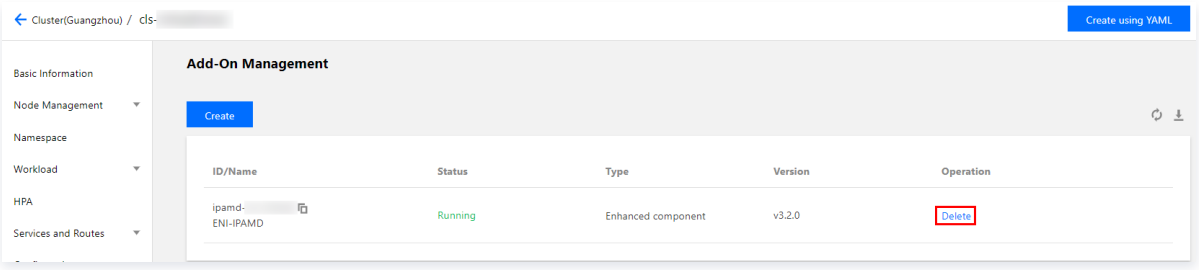
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the "Cluster Management" page, click the ID of the target cluster to enter the cluster details page.
3. Select **Add-on Management** in the left menu bar to enter the "Add-on List" page.
4. On the "Add-On List" page, select **Create** to enter the component installation page, as shown below:



5. Select the required components and click **Finish** to complete the installation.

Component Uninstallation

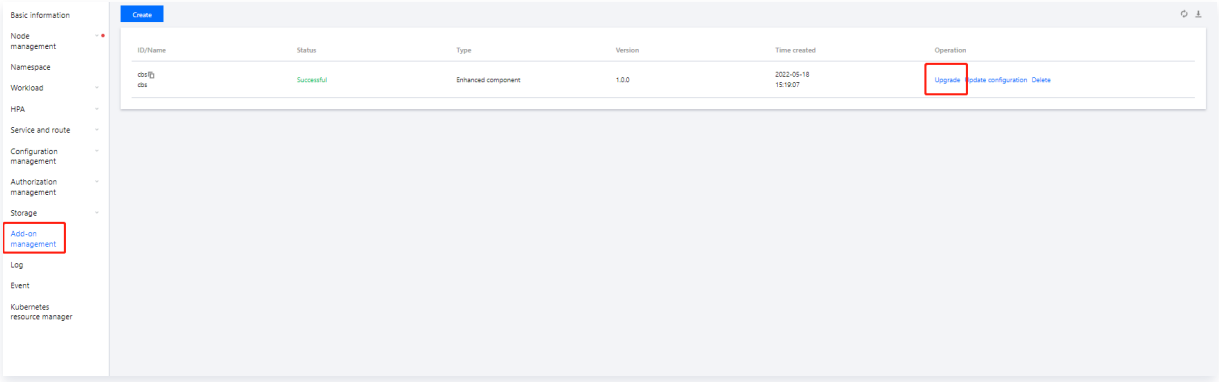
1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the "Cluster Management" page, click the ID of the target cluster to enter the cluster details page.
3. Select **Add-on Management** in the left menu bar to enter the "Add-on List" page.
4. On the "Add-On List" page, click **Delete** on the right side of the row containing the component you want to remove, as shown below:



5. In the "Delete Resource" pop-up window, click **Confirm** to complete the component uninstallation.

Add-on upgrade

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 enter the cluster details page.
3. Select **Add-on Management** in the left menu bar to enter the "Add-on List" page.
4. On the "Add-On List" page, click **Upgrade** on the right side of the row for the component that needs to be upgraded, as shown below:



5. In the "Component Upgrade" pop-up window, click **Confirm** to complete the upgrade.

> **Note**
> For details on add-on version updates, please refer to Add-On Version Maintenance Description.

OOMGuard

Last updated: 2023-09-26 15:10:20

Feature Overview

> **Note**
> This add-on reduces the chance of various kernel failures triggered by cgroup memory reclamation failures in the user mode. It is applicable only to native kernel defects of CentOS 7.2/7.6. For other image versions, you do not need to install this add-on.

Component 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 a component provided by Tencent Kubernetes Engine (TKE) to handle container cgroup OOM in user mode. When cgroup OOM occurs, OOM-Guard kills containers exceeding memory limits directly in user space before the system kernel kills the related container processes. This reduces the probability of various node failures triggered by memory reclamation 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".

Rule	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 within a cluster

Kubernetes Object Name	Local Disk Types	Default Resource Consumption	Namespaces
oomguard	ServiceAccount	-	kube-system
system:oomguard	ClusterRoleBinding	-	-
oom-guard	DaemonSet	0.02-core CPU, 120 MB memory	kube-system

Use Cases

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.

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 enter the cluster details page.
3. Select **Component Management** from the left menu bar to access the "Component List" page.
4. In the "Component List" page, select **Create**, and check OOM-Guard in the "Create Component" page.
5. Click **Done** to complete the process.

NodeProblemDetectorPlus Add-on

Last updated: 2023-09-26 15:10:33

Feature Overview

Component 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 within a cluster

Kubernetes Object Name	Local Disk Types	Resource Amount	Namespaces
node-problem-detector	DaemonSet	0.5C80M	kube-system
node-problem-detector	ServiceAccount	-	kube-system
node-problem-detector	ClusterRole	-	-
node-problem-detector	ClusterRoleBinding	-	-

Use Cases

The Node-Problem-Detector-Plus add-on enables monitoring of node working status, including kernel deadlock, OOM, system thread count pressure, and system file descriptor pressure. These metrics are reported to the Apiserver in the form of Node Condition and Event.

By monitoring these metrics, you can anticipate resource pressure on nodes and manually release or scale up node resources before the node starts evicting Pods. This helps prevent potential losses caused by Kubernetes resource reclamation 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.

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 enter the cluster details page.
3. Select **Component Management** from the left sidebar menu to access the "Component List" page.
4. In the "Component List" page, select **Create** and check Node-Problem-Detector-Plus in the "Create Component" page.
5. Click **Done** to create the component. After successful installation, the corresponding node-problem-detector resources will be available in your cluster, and additional entries will be added to the Node's Condition.

See Also

Node Conditions

After the NPD plug-in is installed, the following specific Conditions will be added to nodes:

Condition Type	Default value	Description
ReadonlyFilesystem	False	Indicate whether the file system is read-only
FDPressure	False	Query whether the number of file descriptors of the host reaches 80% of the maximum value

FrequentKubeletRestart	False	Indicates whether Kubelet has restarted more than 5 times in 20 minutes.
CorruptDockerOverlay2	False	Indicate whether DockerImage is faulty
KubeletProblem	False	Indicates whether the Kubelet service is Running.
KernelDeadlock	False	Indicate whether a deadlock occurs 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	Indicate whether the current number of threads of the system reaches 90% of the maximum value
NetworkUnavailable	False	Indicates whether the NTP service status is Running.
SerfFailed	False	Detect the node network health in distributed mode

NodeLocalDNSCache

Last updated: 2023-09-26 15:10:45

Feature Overview

Component 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 within a cluster

Kubernetes Object Name	Local Disk Types	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
        }
```

Instructions

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 enter the cluster details page.
3. Select **Component Management** from the left sidebar menu to access the "Component List" page.
4. In the "Component List" page, select **Create** and check NodeLocalDNSCache in the "Create Component" page. For detailed configuration of NodeLocalDNSCache, please refer to the official documentation.
5. Click **Done** to complete the process.

DNSAutoscaler

Last updated: 2023-09-26 15:10:56

Feature Overview

Component 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. Currently, 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
    }
```

Target replica calculation formula:
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 within a cluster

Kubernetes Object Name	Local Disk Types	Requested Resource	Namespace

tke-dns-autoscaler	Deployment	Per node: 20M CPU and 10Mi memory	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.

Note

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.

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 menu bar, select **Add-On Management**. Navigate to the **Add-On List** page and click **Create**.
4. On the **Create add-on** page, select DNSAutoscaler. The default scaling configuration of this component 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** to complete the process.

COS-CSI

Last updated: 2023-09-26 15:11:10

Feature Overview

Component Description

The Kubernetes-csi-tencentcloud COS plugin implements the CSI interface, enabling you to utilize Tencent Cloud Object Storage (COS) within container clusters.

Kubernetes objects deployed within a cluster

Kubernetes Object Name	Local Disk Types	Default Resource Consumption	Namespaces
csi-coslauncher	DaemonSet	-	kube-system
csi-cosplugin	DaemonSet	-	kube-system
csi-cos-tencentcloud-token	Secret	-	kube-system

Use Cases

Cloud Object Storage (COS) is a powerful Tencent Cloud distributed storage service that features low costs and high scalability, reliability, and security. It enables you to store a massive number of files and view them on the cloud anytime.
With the COS-CSI add-on, you can quickly use COS in container clusters as COSFS through standard native Kubernetes. For more information, see Introduction to COSFS Tool.

Limits

- Supports clusters with Kubernetes version 1.10 and above.
- For Kubernetes 1.12 clusters, you need to add the following kubelet configuration: `--feature-gates=KubeletPluginsWatcher=false`.
- COSFS inherent limitations, for more information, see COSFS Limitations.
- To use COS in TKE, you need to install this add-on within the cluster, which will consume some system resources.

How to Use

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 enter the cluster details page.
3. Select **Component Management** from the left sidebar menu to access the "Component List" page.
4. In the "Component List" page, select **Create** and check the COS box in the "Create Component" page.
5. Click **Done** to complete the process.

Using COS

You can mount object storage to workloads in a TKE cluster. For more information, please refer to Using Cloud Object Storage (COS).

CFS-CSI

Last updated: 2023-09-26 15:11:45

Feature Overview

Component Description

The Kubernetes-csi-tencentcloud CFS plugin implements the CSI interface, enabling you to use Tencent Cloud File Storage in container clusters.

> **Note**
> For 1.12 clusters, modify the kubelet configuration by adding `--feature-gates=KubeletPluginsWatcher=false`.

Kubernetes objects deployed within a cluster

Kubernetes Object Name	Local Disk Types	Default Resource Consumption	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) offers a scalable shared file storage service, which can be seamlessly integrated with Tencent Cloud's CVM, TKE, and BatchCompute services. CFS provides standard NFS and CIFS/SMB file system access protocols, serving as a shared data source for multiple CVM instances or other computing services. It supports elastic capacity and performance scaling, allowing existing applications to be mounted without any modifications. As a highly available and reliable distributed file system, CFS is well-suited for scenarios such as big data analysis, media processing, and content management.
CFS is easy to integrate, eliminating your need to adjust your business structure or make complex configurations. To integrate and use CFS, simply complete three steps: creating a file system, launching a file system client on a server, and mounting the created file system. With the CFS-CSI extension, you can quickly use CFS in container clusters through standard native Kubernetes. For more information, please refer to CFS Use Cases.

Limits

- For CFS-specific limitations, refer to CFS System Limitations.
- To use CFS in TKE, you need to install this extension component within the cluster, which will consume some system resources.

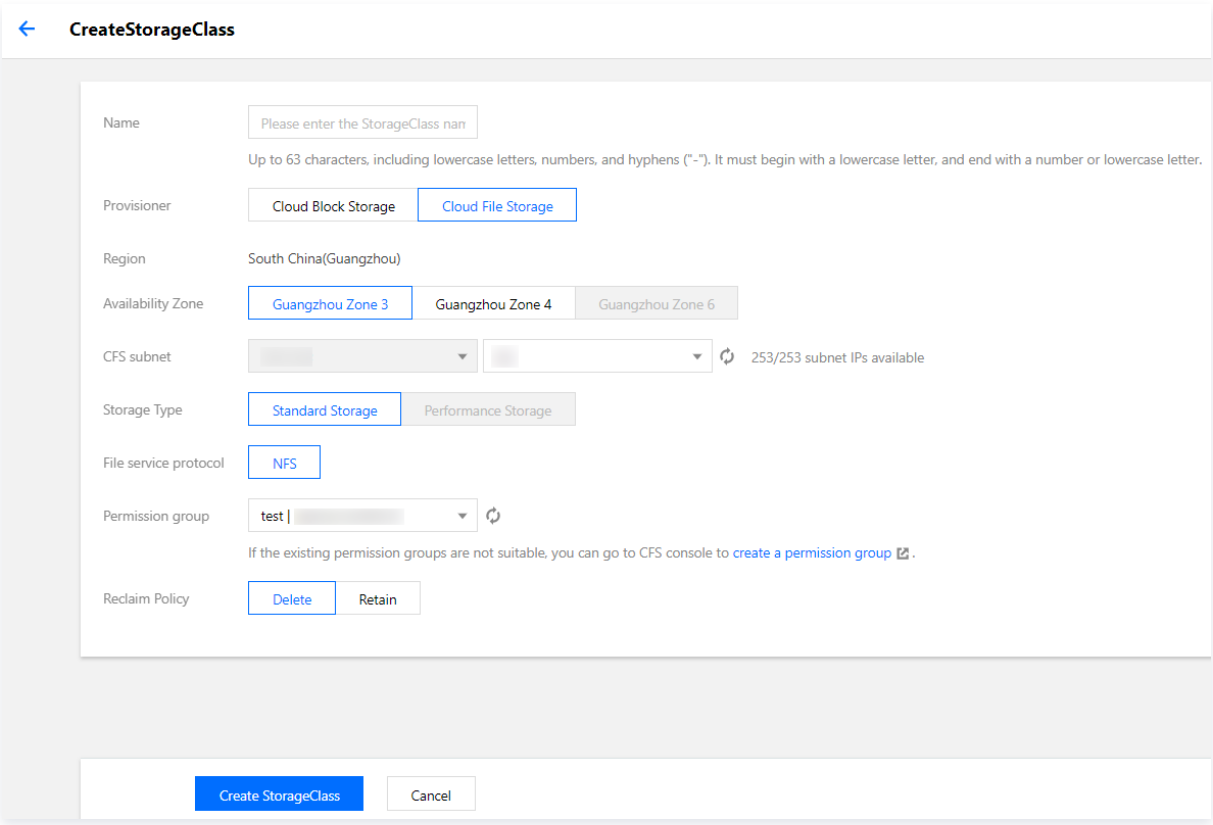
Instructions

Install and Configure 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 enter the cluster details page.
3. Select **Component Management** from the left sidebar to access the "Component List" page.
4. On the "Component List" page, select **Create** and check CFS on the "Create Component" page.
5. Click **Done** to complete the process.

Creating a CFS-type StorageClass

1. On the "Cluster Management" page, click the ID of the cluster using CFS to enter the cluster details page.
2. In the left navigation bar, select **Storage** > **StorageClass**, and click **Create** to access the "New StorageClass" page.
3. Create a CFS-type StorageClass based on your actual requirements, as shown below:



4. Click **Create StorageClass** to complete the creation.

Create a PersistentVolumeClaim

1. On the "Cluster Management" page, click the ID of the cluster using CFS to enter the cluster details page.
2. In the left navigation bar, select **Storage** > **PersistentVolumeClaim**, and click **Create** to enter the "New PersistentVolumeClaim" page.
3. Based on your actual requirements, create a CFS-type PersistentVolumeClaim and select the StorageClass created in the previous steps.
4. Click **Create PersistentVolumeClaim** to complete the creation.

Creating a workload

1. On the "Cluster Management" page, click the ID of the cluster using CFS to enter the cluster details page.
2. In the left navigation bar, select **Workload** > **Deployment**, and click **Create** to enter the "New Workload" page.
3. Based on your specific requirements, select **Use existing PVC** for the volume and choose the previously created PVC.
4. After mounting to the specified container path, click **Create Workload** to complete the creation.

CBS-CSI Description
CBS-CSI

Last updated: 2023-09-26 15:13:27

Scenario

CBS-CSI Component enables TKE clusters to quickly select storage types through the console and create corresponding block storage cloud disk type PVs and PVCs. This document provides an overview of the CBS-CSI component's features and introduces several common use case examples.

Item

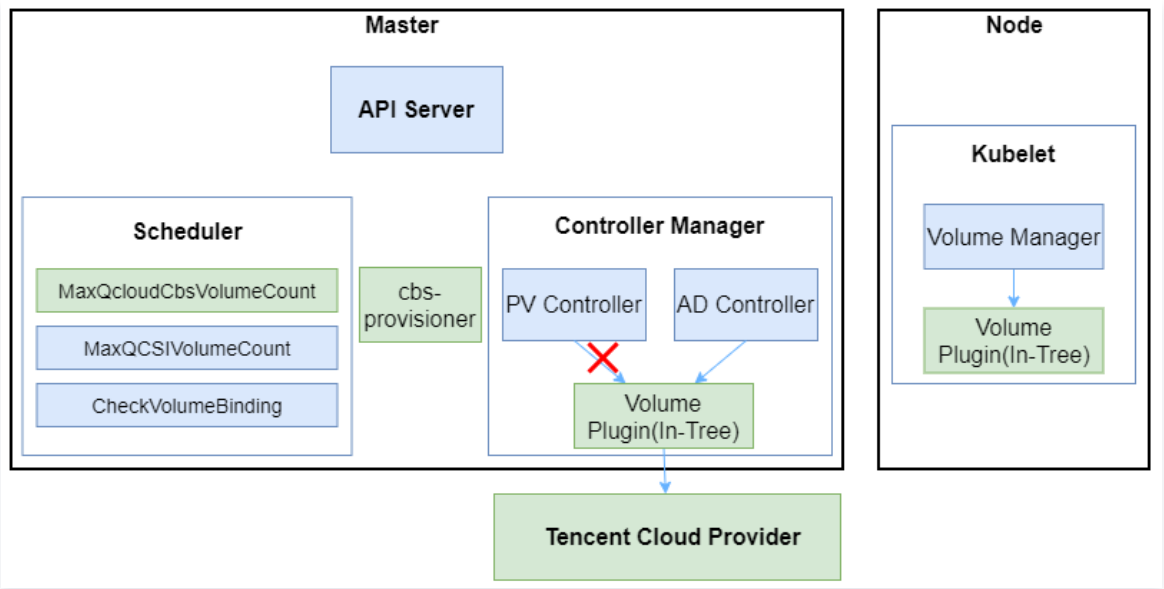
SDK	Note
Static volume	Supports manual creation of volumes, PV objects, and PVC objects.
Dynamic volume	Supports configuration, creation, and deletion of volumes and PV objects through StorageClass.
Storage topology awareness	CBS does not support cross-AZ mounting. In a cluster with multiple AZs, the 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 can mount up to 20 cloud disks. When scheduling pods, the scheduler will filter out nodes where the maximum number of mounted cloud disks has been exceeded.
Volume online expansion	You can modify the capacity field of PVC to implement online expansion (only CBS types are supported).
Volume snapshot and restoration	Supports creation of volumes through snapshots.

Description

After deployment 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 mounting capability.
- StatefulSet and Deployment (Controller): consists of a Driver and multiple Sidecars (external-provisioner, external-attacher, external-resizer, external-snapshotter, and snapshot-controller). It provides the capabilities to create or delete volumes, attach or detach, expand, snapshot, etc.

The example diagram is shown below:



Limits

- TKE cluster version ⩾ 1.14
- 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.)

Sample Code

- Avoid cross-AZ mounting of cloud disks with CBS-CSI
- Online expansion of cloud disks
- Create snapshots and restore volumes using snapshots

Avoid attaching cloud disk across availability zones through cbs-csi

Last updated: 2023-09-26 15:13:58

Scenario

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 to Implement

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.

Preparations

- You have installed a TKE cluster of v1.14 or later versions. For more information, see Creating a Cluster .
- The CBS-CSI or In-Tree components have been updated to the latest version.

Instructions

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: 2023-09-26 15:14:07

Scenario

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.

Preparations

- 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.

Instructions

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 capacity expansion

Two expansion methods are provided:

Scaling	Note
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

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   STATUS   CLAIM
STORAGECLASS   REASON   AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   30Gi       RWO            Delete           Bound    default/www1-ivantestweb-
0   cbs-csi                 20h
```

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.kubernetes.io/zone=nozone
```

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 available: 1 node(s) had no available
volume zone.
```

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":"40Gi"}}}}'
```

> **Note**
> 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.kubernetes.io/zone-
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   STATUS   CLAIM
STORAGECLASS   REASON   AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   40Gi       RWO            Delete           Bound    default/www1-ivantestweb-0
cbs-csi                 20h
$ kubectl get pvc www1-ivantestweb-0
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
www1-ivantestweb-0   Bound    pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   40Gi       RWO            cbs-csi        20h
$ 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
```

Online expansion without restarting the Pod

1. Run the following command to check the status of the PV and file system before expansion. In the example below, both the

PV and file system sizes are 20GB:

```
$ 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   STATUS   CLAIM
STORAGECLASS   REASON   AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   20Gi       RWO            Delete           Bound    default/www1-ivantestweb-
0   cbs-csi             20h
```

2. Run the following command to modify the capacity in the PVC object and expand the capacity to 30GB. Example:

```
$ kubectl patch pvc www1-ivantestweb-0 -p '{"spec":{"resources":{"requests":{"storage":"30Gi"}}}}'
```

> **Note**
> 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   STATUS   CLAIM
STORAGECLASS   REASON   AGE
pvc-e193201e-6f6d-48cf-b96d-ccc09225cf9c   30Gi       RWO            Delete           Bound    default/www1-ivantestweb-0
cbs-csi             20h
```

Creating Snapshot and Using It to Restore Volume

Last updated: 2023-09-26 15:14:14

Scenario

If you need to create a snapshot of the PVC data disk to backup 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.

Preparations

- You have created a TKE cluster of v1.18 or later versions. For more information, see Creating a Cluster.
- You have installed the latest version of CBS-CSI add-on. For more information, see CBS CSI Documents.

Instructions

Backing up PVC

Creating a VolumeSnapshotClass

1. Use the following YAML to create a VolumeSnapshotClass object, as shown below:

```
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
```

Create a PVC snapshot object VolumeSnapshot

1. In this example, we use the snapshot name `new-snapshot-demo` to create a VolumeSnapshot object using the following YAML. The example is as follows:

```
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, as shown below:

```
$ kubectl get volumesnapshot
NAME              READYTOUSE   SOURCEPVC         SOURCESNAPSHOTCONTENT   RESTORESIZE   SNAPSHOTCLASS
SNAPSHOTCONTENT                           CREATIONTIME   AGE
new-snapshot-demo   true       www1-ivantestweb-0                  10Gi       cbs-snapclass   snapcontent-ea11a797-
d438-4410-ae21-41d9147fe610   22m         22m
```

```
$ kubectl get volumesnapshotcontent
NAME                                   READYTOUSE   RESTORESIZE   DELETIONPOLICY   DRIVER
VOLUMESNAPSHOTCLASS   VOLUMESNAPSHOT     AGE
snapcontent-ea11a797-d438-4410-ae21-41d9147fe610   true        10737418240   Delete          com.tencent.cloud.csi.cbs
cbs-snapclass         new-snapshot-demo   22m
```

3. Execute the following command to obtain the snapshot ID of the VolumeSnapshotContent object. The field is `status.snapshotHandle` (e.g., snap-e406fc9m). You can confirm whether the snapshot exists in the Cloud Service Console > Snapshot List based on the snapshot ID. Example:

```
$ kubectl get volumesnapshotcontent snapcontent-ea11a797-d438-4410-ae21-41d9147fe610 -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/snapcontent-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. In this document, we use the VolumeSnapshot object named `new-snapshot-demo` created in the steps above as an example. The following YAML demonstrates how to restore a volume from a 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 (schedule the Pod first and then create the PV), that is, to specify `volumeBindingMode: WaitForFirstConsumer`, you need to deploy the Pod (to mount the PVC) to trigger the creation of the PV (create a new CBS from the snapshot and bind it to the PV).

TCR Introduction

Last updated: 2023-09-26 15:14:47

Feature Overview

Component 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 within a cluster

Name	Local Disk Types	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	Number of instances
tcr-assistant-controller-manager	CPU: 100m memory: 30Mi	1
hosts-updater	CPU: 100m memory: 100Mi	Number of worker nodes

Use Cases

Pulling images without a Secret

Pulling private images in a Kubernetes cluster requires creating an access credential Secret resource and configuring the ImagePullSecret attribute in the resource YAML file, explicitly specifying the created Secret. The overall configuration process can be cumbersome and may cause image pulling failures due to incorrect or missing ImagePullSecret configurations.

To address these issues, you can install the TCR add-on in the cluster. The add-on will automatically obtain the access credentials for the specified TCR Enterprise Edition instance and distribute them to the designated namespace within the TKE cluster. When creating or updating resources using YAML, there is no need to configure ImagePullSecret, as the cluster will automatically use the distributed access credentials to pull images from the TCR Enterprise Edition instance.

Pulling images over the private network

The component will automatically create a DaemonSet workload called host-updater, which updates the Host configuration of cluster nodes and resolves the associated instance domain name to the dedicated private IP of the established private network access link. Please note that this configuration is intended for testing scenarios only. It is recommended to use the automatic private network link resolution provided by TCR, or configure private domain resolution using the PrivateDNS product, or manage resolution using a self-built 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.

How to Use

1. Select an associated instance: select an existing TCR Enterprise Edition instance under the current logged-in account and confirm that the current logged-in user has the 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.
2. Configure Secret-free Pulling (enabled by default): You can choose to automatically distribute the current user's access credentials or specify a username and password. You can also optionally configure the namespaces and ServiceAccounts for which Secret-free Pulling is enabled. We recommend using the default settings to avoid issues with this feature when new namespaces are created.
3. Configure Private Network Parsing (Advanced Feature): Ensure that the cluster and associated TCR instance have established a private network access link and enabled the private network parsing function. Please note that this configuration is for testing scenarios only. It is recommended to use the automatic private network link parsing provided by TCR, or directly use the PrivateDNS product for private domain parsing configuration, or manage parsing using a self-built DNS service.
4. After the TCR add-on is created, if you need to modify its configuration, delete the add-on and reconfigure and reinstall it.

> ⚠ **Note**
> Deleting the add-on will not automatically delete the dedicated access credentials created. You can manually disable or delete them in the Tencent Container Registry console.

How It Works

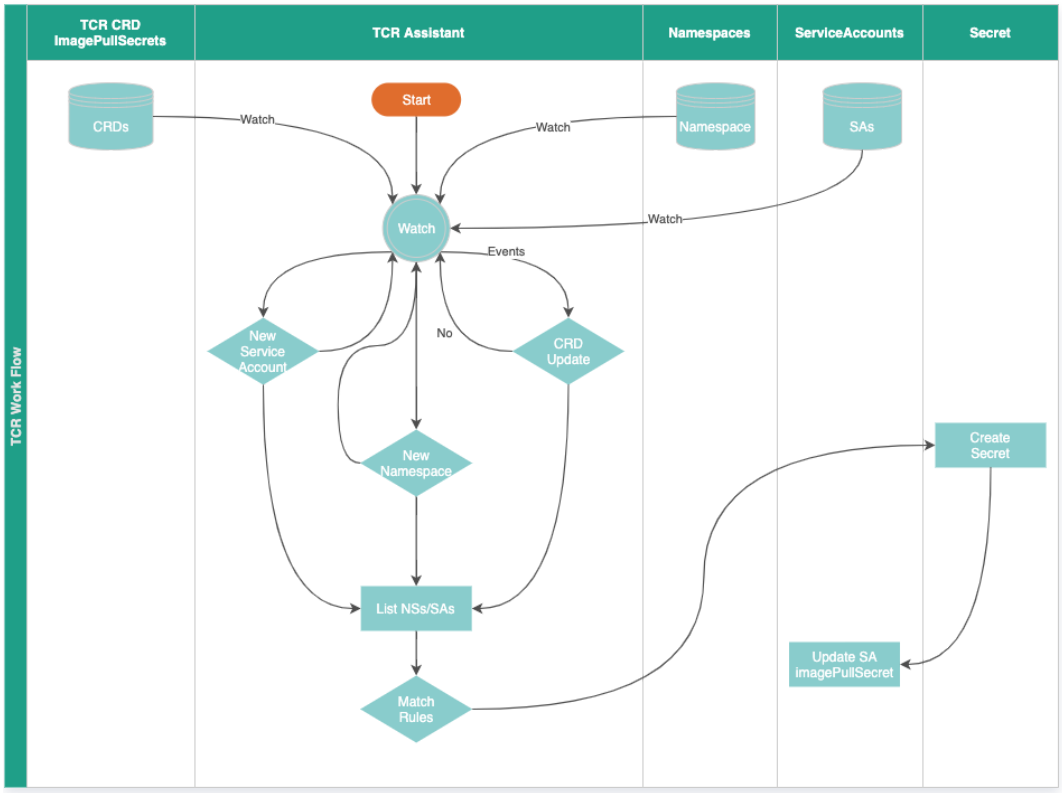
Overview

TCR Assistant is designed to help users automatically deploy k8s `imagePullSecret` to any `Namespace` and associate it with the `ServiceAccount` in that namespace. When a user-created workload does not explicitly specify an **imagePullSecret** and a `serviceAccount`, k8s will attempt to find and match the appropriate `imagePullSecret` from the `ServiceAccount` resource named `default` in the current 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 to Implement



TCR Assistant is a typical Kubernetes Operator. When deploying TCR Assistant, we create a Custom Resource Definition (CRD) object in the target Kubernetes cluster: `imagepullsecrets.tcr.tencentcloudcr.com`. The kind of this CRD is `ImagePullSecret`, the version is `tcr.tencentcloudcr.com/v1`, and the abbreviation is `ips` or `ipss`.
TCR Assistant continuously observes (watches) the k8s cluster's `Namespace` and `ServiceAccount` resources. When these resources change, it checks whether the resource changes match the rules set in `ImagePullSecret` to automatically deploy the Secret resources needed to pull **private image repositories**. The program is typically deployed within the k8s cluster and accesses the k8s master API using the `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
```

The explanation of the `ImagePullSecret` spec fields is as follows:

Parameter	Effect	Remarks
namespaces	`NameSpace` matching rule	`*` or an empty character represents a match for any value; to match multiple `NameSpaces`, use `,` to separate resource names. **Note**: Expressions are not supported; you must explicitly specify the resource names.
serviceAccounts	`serviceAccounts` matching rule	`*` or an empty character represents a match for any value. To match multiple `ServiceAccounts`, use a `,` to separate resource names. **Note**: Expressions are not supported; you must explicitly specify the resource names.
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-SUCCESS
imagepullsecret-sample                      10                10

# Viewing Detailed Information
$  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

To update the `Secret` resource deployed by TCR Assistant, there is no need to delete and recreate the `ImagePullSecret` resource. Simply edit the `docker.username` and `docker.password` fields to take effect. For example:

```
$ kubectl edit ipss imagepullsecret-sample
```

Namespace updates

Upon detecting the creation of a new k8s `Namespace` resource, the TCR Assistant will first check if the name matches the `ImagePullSecret` resource's `namespaces` field. If the resource name **does not match**, the subsequent process is skipped. If the resource name matches, the k8s API will be called to create a `Secret` resource and add the `Secret` resource name to the `ServiceAccount` resource's `imagePullSecrets` field. An example is shown below:

```
# View the automatically deployed Secret in the newns namespace
$ 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

# View the Secret automatically associated with the ServiceAccount resource "default" in the "newns" namespace
$ 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

Upon detecting the creation of a new k8s `ServiceAccount` resource, the TCR Assistant will first check if the name matches the `ImagePullSecret` resource's `serviceAccounts` field. If the resource name **does not match**, the subsequent process is skipped. If the resource name matches, the TCR Assistant will call the k8s API to create or update the `Secret` resource and add the `Secret` resource name to the `ServiceAccount` resource's `imagePullSecrets` field. An example is shown below:

```
# Create a ServiceAccount resource in the newns namespace
$ kubectl create sa kung -n newns
serviceaccount/kung created

# View the Secret automatically associated with the newly created ServiceAccount resource named "kung" in the "newns"
namespace.
$ 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
```

P2P

Last updated: 2023-09-26 15:14:59

Feature Overview

Component Description

P2P Addon is a container image acceleration and distribution plugin based on P2P technology, introduced by Tencent Container Registry (TCR). It is designed for large-scale TKE clusters to quickly pull GB-level container images and supports concurrent pulling for thousands of nodes.
The component consists of `p2p-agent`, `p2p-proxy`, and `p2p-tracker`:

- p2p-agent: Deployed on each node in the cluster, it proxies image pull requests for each node and forwards them to various peers (node nodes) within the P2P network.
- p2p-proxy: Deployed on some nodes in the cluster, it serves as the original seed connection for the accelerated image repository. Proxy nodes are responsible for both seeding and pulling original data from the target image repository.
- p2p-tracker: Deployed on some nodes in the cluster, it is an open-source BitTorrent protocol tracker service.

Kubernetes objects deployed within a cluster

Kubernetes Object Name	Local Disk Types	Requested Resource	Namespace
p2p-agent	DaemonSet	0.2 cores CPU and 0.2 GB memory per node	kube-system
p2p-proxy	Deployment	0.5-core CPU and 0.5 GB memory per node	kube-system
p2p-tracker	Deployment	0.5-core CPU and 0.5 GB memory per node	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

Ideal for large-scale TKE clusters to quickly pull GB-level container images and support concurrent pulling for thousands of nodes, the recommended use cases are as follows:

- In a cluster with 500 to 1000 nodes, using local disk storage for pulled container images, the nodes within the cluster can support a maximum concurrent pull speed of 100 MB/s.
- In a cluster with 500-1000 nodes, using CBS cloud disk storage to pull container images, and located in major domestic regions such as Guangzhou, Beijing, and Shanghai, the nodes within the cluster can support a maximum concurrent pull speed of 20 MB/s.

Limits

- When enabling P2P Addon to pull container images in large-scale clusters, it may cause high read and write pressure on the node data disks, potentially affecting existing services within the cluster. If the nodes in the cluster use CBS cloud disks to store pulled container images, please select an appropriate download speed limit based on the cluster's region or contact your after-sales support or architect to avoid disruptions to live services within the cluster due to high cloud disk read and write loads during image pulling, or even affecting the normal usage of other users in the region.
- Enabling the P2P plugin requires reserving certain resources. The P2P component will occupy CPU and memory resources of the nodes during the image acceleration process, and will no longer consume resources after the acceleration is completed. Specifically:
  - The proxy's limit is set to: 4 cores CPU and 4 GB memory.
  - The limit for the Agent is: 4-core CPU and 2 GB memory.

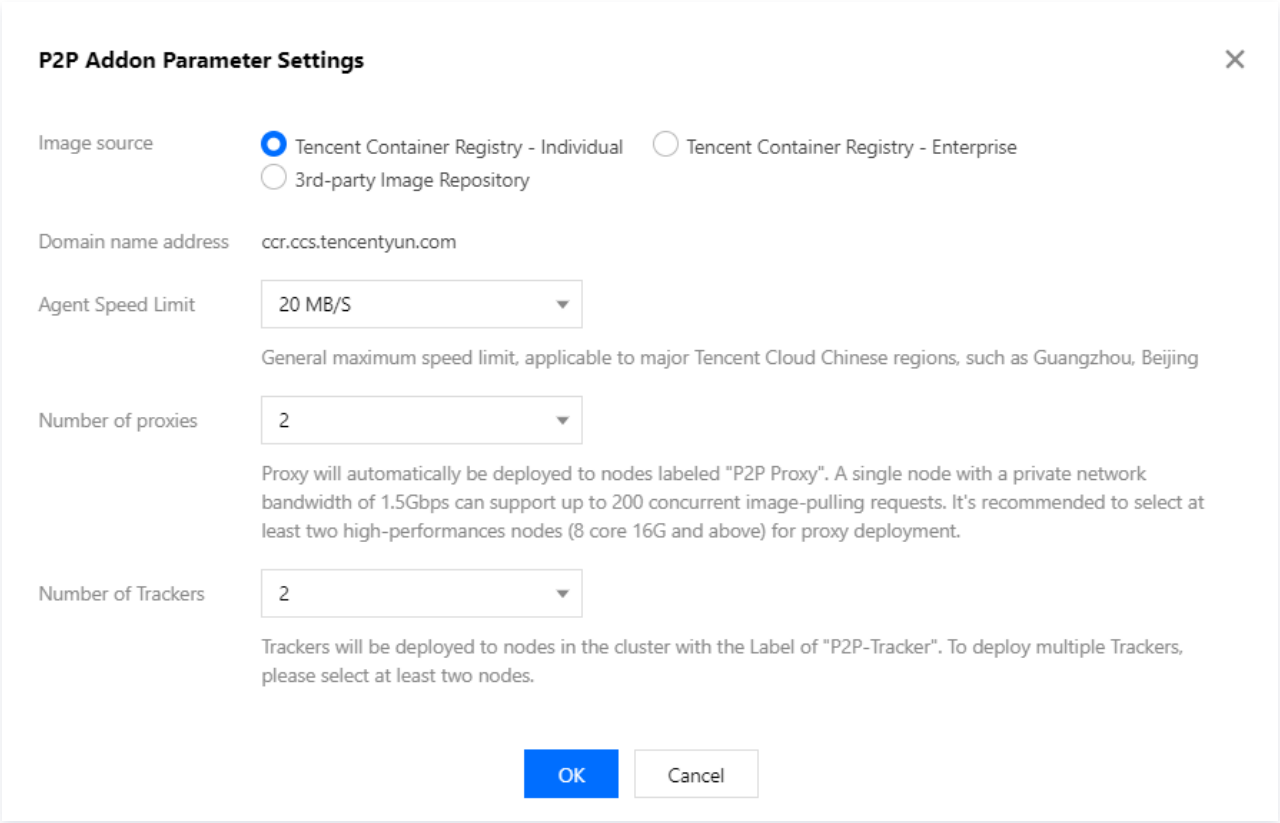
- The limit for Tracker is set to: 2 cores CPU and 4 GB memory.
- Based on the scale of the cluster nodes, estimate the number of Proxy instances to be launched. The minimum configuration for a Proxy running node is 4C8G with an internal network bandwidth of 1.5GB/s. A single Proxy service can support up to 200 cluster nodes.
- You need to actively select deployment nodes for the Proxy and Tracker components by manually adding K8S labels to the nodes. For more information, see Usage Method. The nodes where Proxy and Agent are located need to have access to the repository source.
- The Agent component will occupy port 5004 on the node, as well as P2P dedicated communication ports 6881 (Agent) and 6882 (Proxy). The Agent and Proxy components will create local working directories `/p2p_agent_data` and `/p2p_proxy_data` for caching container images, so please ensure that sufficient storage space has been reserved on the node in advance.

How to Use

1. Select appropriate nodes to deploy and run the Proxy component. You can use the `kubectl label nodes XXXX proxy=p2p-proxy` command to mark nodes, and the plugin will automatically deploy the component on these nodes during installation. If you need to adjust the number of Proxy components after installation, you can add or remove the label on the specified node and modify the number of replicas for the p2p-proxy workload in the kube-system namespace within the cluster.
2. Select appropriate nodes to deploy and run the Tracker component. You can use the `kubectl label nodes XXXX tracker=p2p-tracker` command to mark nodes, and the plugin will automatically deploy the component on these nodes during installation. After installation, if you need to adjust the number of Trackers, you can add or remove the label on the specified node and modify the number of replicas for the p2p-tracker workload in the kube-system namespace within the cluster.
3. The security group configuration required for the nodes is as follows: Inbound rules should allow TCP and UDP ports 30000 - 32768, and allow all IPs within the VPC. Outbound rules should allow all traffic (the default security group for TKE cluster worker nodes already meets these requirements).
4. Select a specific cluster to enable the P2P Addon plugin. Enter the domain name of the image repository to be accelerated, the node pull rate limit, the number of Proxies, and the number of Trackers. If you need to adjust the maximum download speed after installation, you can modify the downloadRate and uploadRate in the 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 service YAML, changing the domain name address of the image repository that needs acceleration to localhost:5004, such as localhost:5004/p2p-test/test:1.0, and use the newly created dockercfg as the 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.

Instructions

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 enter the cluster details page.
3. Select **Component Management** from the left menu bar to enter the "Component List" page.
4. In the "Component List" page, select **Create** and check P2P in the "Create Component" page.
5. Select "Parameter Configuration". In the pop-up "P2P Add-on Parameter Settings" window, fill in the image repository domain name to be accelerated, node pull rate limit, number of Proxy instances, and number of Tracker instances, as shown below:



Network Policy

Last updated: 2023-09-26 15:15:37

Feature Overview

Component Description

Network Policy is a resource provided by Kubernetes for defining network isolation policies based on Pods. It describes whether a group of Pods can communicate with other Pod groups and other Network Entities. This component provides a Controller implementation for this resource. If you wish to control the network traffic of specific applications at the IP address or port level (OSI Layer 3 or Layer 4), you may consider using this component.

Kubernetes objects deployed within a cluster

Kubernetes Object Name	Local Disk Types	Requested Resource	Namespace
networkpolicy	DaemonSet	250m CPU and 250 MiB memory for each instance	kube-system
networkpolicy	ClusterRole	-	kube-system
networkpolicy	ClusterRoleBinding	-	kube-system
networkpolicy	ServiceAccount	-	kube-system

Instructions

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 enter the cluster details page.
3. Select **Component Management** from the left sidebar menu to access the "Component List" page.
4. In the "Component List" page, select **Create**, and check NetworkPolicy in the "Create Component" page. For detailed configuration of NetworkPolicy, please refer to Network Policy Best Practices.
5. Click **Done** to complete the process.

Nginx-ingress

Last updated: 2023-09-26 15:15:56

Feature Overview

Component Description

Nginx can be utilized as a reverse proxy, load balancer, and HTTP cache. The Nginx-ingress add-on is a Kubernetes Ingress controller that employs Nginx as a reverse proxy and load balancer. You can deploy the Nginx-ingress add-on and use Nginx-ingress within your cluster.

Kubernetes objects deployed within a cluster

Deploying the Nginx-ingress Add-on within the cluster will result in the following Kubernetes objects being deployed inside the cluster:

Kubernetes Object Name	Local Disk Types	Default Resource Consumption	Namespaces
nginx-ingress	Service	-	Custom settings
nginx-ingress	Configmap	-	Custom settings
tke-ingress-nginx-controller-operator	Deployment	0.13 cores CPU, 128 MB memory	kube-system
ingress-nginx-controller	Deployment/DaementSet	0.1 core CPU	kube-system
ingress-nginx-controller-hpa	HPA	-	kube-system

Preparations

- Kubernetes 1.16 and later.
- We recommend using the TKE Node Pool feature.
- We recommend using Tencent Cloud Log Service (CLS).

How to Use

- Nginx-ingress Overview
- Installing Nginx-ingress
- Using Nginx-ingress Object to Access External Traffic of the Cluster
- Nginx-ingress Log Configuration

HPC

Last updated: 2023-09-26 15:18:14

Feature Overview

Component Description

HorizontalPodCronscaler (HPC) is a custom-developed component that allows for scheduled modifications to the replica count of K8s workloads. Used in conjunction with HPC CRD, it supports time-based tasks with a minimum granularity of 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 within a cluster

Deploying HPC Add-on in a cluster will deploy the following Kubernetes objects in the cluster:

Kubernetes Object Name	Local Disk Types	Default Resource Consumption	Namespace
horizontalpodcronscalers.autoscaling.cloud.tencent.com	CustomResourceDefinition	-	-
hpc-leader-election-role	Role	-	kube-system
hpc-leader-election-rolebinding	RoleBinding	-	kube-system
hpc-manager-role	ClusterRole	-	-
hpc-manager-rolebinding	ClusterRoleBinding	-	-
cronhpa-controller-manager-metrics-service	Service	-	kube-system
hpc-manager	ServiceAccount	-	kube-system
tke-hpc-controller	Deployment	100mCPU/pod、100Mi/pod	kube-system

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.

Instructions

Install 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 enter the cluster details page.
3. Navigate to the "Component Management" section in the left-hand menu, and enter the "Component List" page.
4. In the "Add-on List" page, select **Create** and check the HPC box in the "Create Add-on" page.
5. Click **Done** to complete the process.

Examples

Creating a scheduled task resource that associated with Deployment

Sample:

```
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

Sample:

```
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

Sample:

```
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	Required	Value Range	Allowed Special Characters
Seconds	Supported	0 – 59	* / , –
Minutes	Supported	0 – 59	* / , –
Hours	Supported	0 – 23	* / , –
Day of month	Supported	1 – 31	* / , – ?
Month	Supported	1 – 12 or JAN – DEC	* / , –
Day of week	Supported	0 – 6 or SUN – SAT	* / , – ?

TKE-log-agent Description

Last updated: 2023-09-26 15:18:50

Feature Overview

Component Description

TKE-log-agent is a log collection component for Kubernetes clusters, allowing users to non-intrusively collect container standard output logs, container internal logs, and node logs.

Resource objects deployed in the cluster

Kubernetes Object Name	Local Disk Types	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

Use Cases

- When enabling audit log collection for a self-deployed cluster, the TKE-log-agent is installed by default and collects Apiserver audit logs.
- Collect container standard output logs, container internal logs, and node logs through log collection rules.

How It Works

1. Upon detecting that a user has created a collection rule, the CLS-provisioner generates a CLS-side collection configuration based on the rule's settings and synchronizes it with the CLS-side server.
2. TKE-log-agent maps log directories to a unified directory according to the collection rules.
3. Loglistener synchronizes the collection configuration from the CLS server and collects logs according to the configuration, reporting them to the CLS side.

Related documentation

- Enable Log Collection
- Configuring Log Collection via Console
- Configure Log Collection via YAML
- Collect container logs through collection rules and report to Kafka

tke-event-collector Description

Last updated: 2023-09-26 15:18:57

Component Description

The tke-event-collector listens to cluster events and reports them to CLS to achieve cluster event persistence. This component is managed as a Deployment within the container service.

Use Cases

With tke-event-collector, you can collect cluster events and report them to CLS. Users can access Container Service Console and select **Log Management > Event Logs** to view event overviews and aggregated exception event searches on the event search page. Additionally, users can perform event searches based on their specific needs in the global search.

How to Use

1. Log in to the Container Service Console and select **Operation Management** in the left sidebar.
2. On the **Operation Management** page, click **Settings** on the right side of the cluster.
3. In the **Configure Features** window, click **Edit** on the right side of Event Storage, and select **Enable Event Storage**.
4. Select the **Logset** and **Log Topic**, then click **Confirm** to enable the event storage feature.
5. After enabling event storage, select **Log Management > Event Logs** in the left sidebar to view the event dashboard and search logs.

Application Management Overview

Last updated: 2023-09-26 15:19:23

Application features refer to the Helm 3.0 related capabilities integrated within Tencent Kubernetes Engine (TKE), enabling you to create various products and services such as Helm charts, container images, and software services. Created applications will run in the cluster you specify, providing you with corresponding capabilities.

Application-related operations

- Application Management
- Connecting to a cluster using the local Helm client

Use the application

Last updated: 2023-09-27 09:31:12

This document describes how to create, update, roll back, and delete applications in the TKE console.

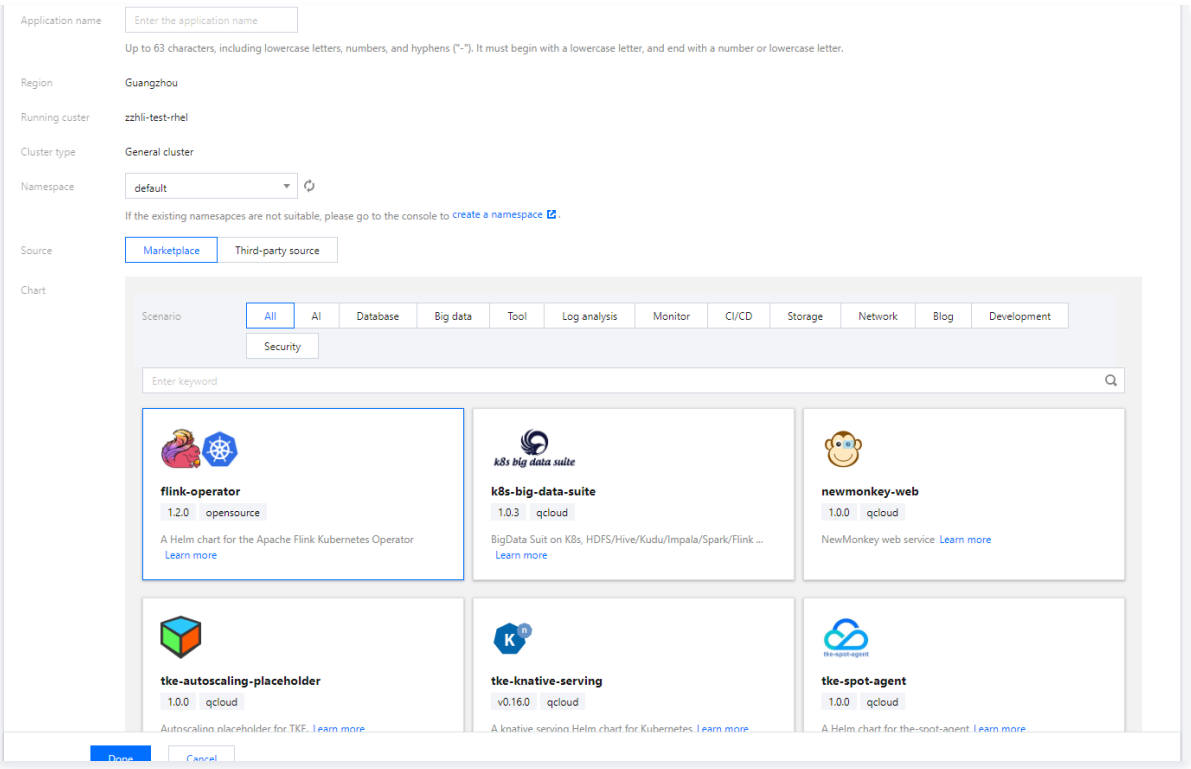
Notes

Application management applies only to clusters with Kubernetes v1.8 or later.

Instructions

Creating an application

1. Sign in to the TKE console and select Applications from the left navigation menu.
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.



The main parameter information is as follows:

- **Application Name**: Enter a custom application name.
- **Source**: Select **Marketplace**, **TCR Private Repository**, or **Third-party Source**. For details, see the following table.

> **Note:**
> The feature of using **TCR private repository** as the source was discontinued on March 31, 2023. After the discontinuation, you will not be able to install new applications through TCR private repository. Existing installed applications will not be affected but cannot be updated. Tencent Container Registry's hosted Helm Chart fully covers this capability and provides more practical and secure deployment services, enabling the simultaneous use of container images and Helm Chart cloud-native deliverables in business projects.

Source	Configuration items
Marketplace	Filter charts by cluster type or application scenario. Select the required application package and chart version. You can also edit the parameters.
TCR private repository	• TCR Instance Name: Select the desired Tencent Cloud Container Registry (TCR) Enterprise Edition instance as needed.

	• Namespace: Select a namespace of the specified TCR instance as needed. After the namespace is specified, charts under the namespace will be displayed on the application list page. • Chart version and parameters: Select the applicable 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.

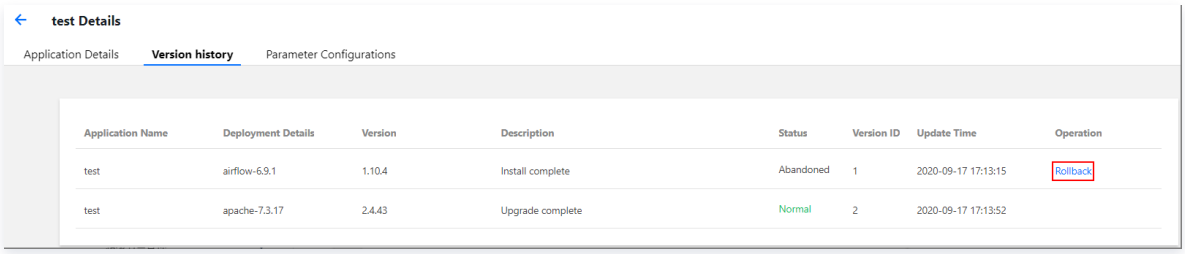
4. Click **Done**.

Updating an application

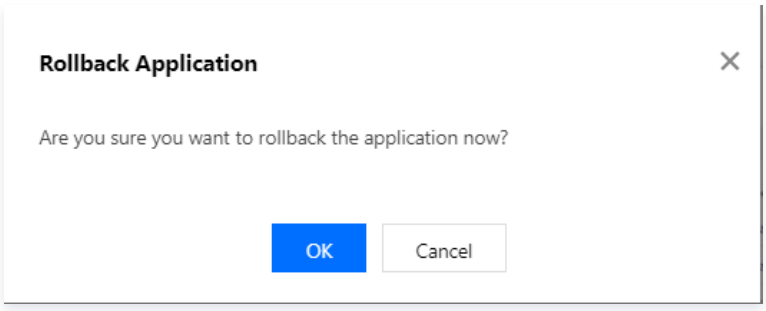
1. Go to the TKE console and click **Application** in the left sidebar.
2. On the **Application** page, locate the application to update and click **Update Application** on the right.
3. In the displayed **Update Application** window, configure the key information as needed and click **Done**.

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.



4. On the **Roll Back Application** page, click **Confirm** as shown below:



Deleting an application

1. Go to the Application Console and select **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: 2023-09-26 15:19:36

Scenario

This document explains how to connect to a cluster using local Helm client.

Instructions

Downloading the Helm Client

Run the following commands in sequence to download and install the Helm client. For more information, 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 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/
```

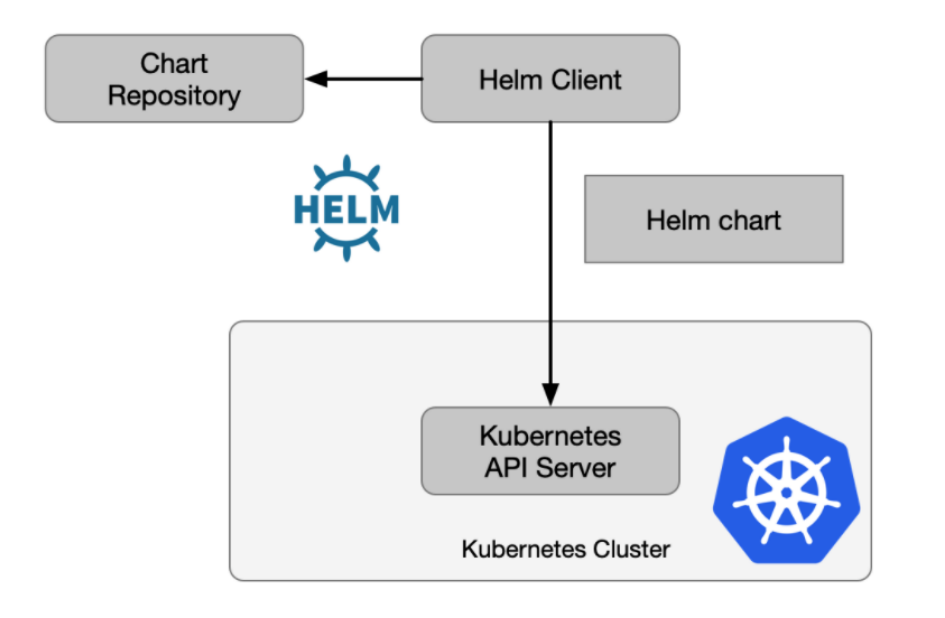
2. Run the following command to configure the Tencent Cloud App Market.

```
helm repo add tkemarket https://market-tke.tencentcloudcr.com/chartrepo/opensource-stable
```

3. Configure TCR Private Helm Repository.

Connecting to a Cluster

Compared to Helm v2, Helm v3 has removed the Tiller component, allowing the Helm client to directly connect to the cluster's ApiServer. Application-related version data is stored directly in Kubernetes, as shown in the following diagram:



The Helm Client uses the client certificate generated by TKE to access the cluster. The specific steps are as follows:

1. Obtain the Kubeconfig with public or private network access through the TKE console or API.
2. You can connect to the target cluster using either of the following methods:
   - Using the obtained kubeconfig, configure the kubectl config use-context on the machine where the Helm Client is located.
   - Run the following command to access the target cluster by specifying parameters.

```
helm install .... --kubeconfig [path to kubeconfig]
```

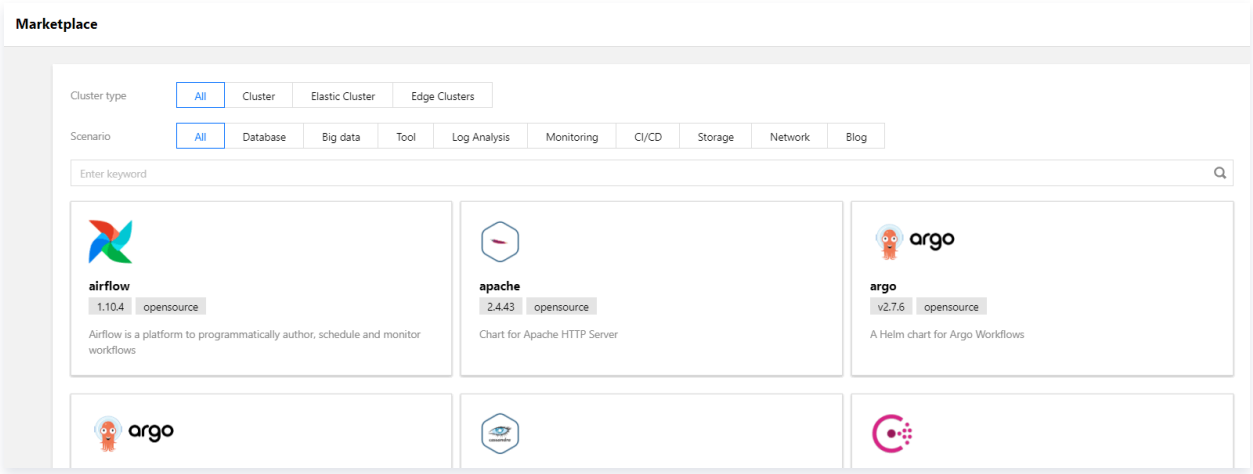
Application Market

Last updated: 2023-09-26 15:19:48

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.

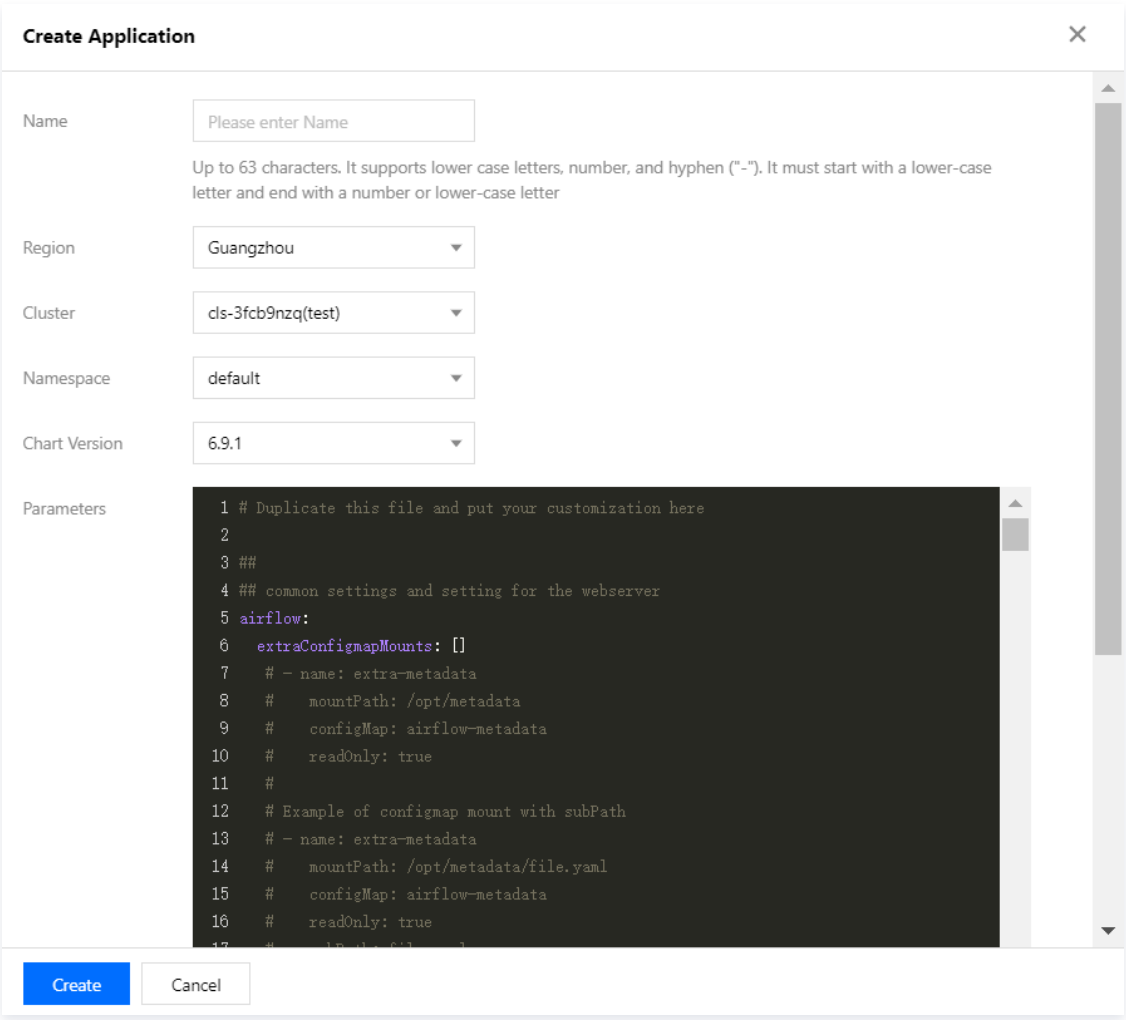
View Applications

1. Log in to the TKE console and select **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: Includes 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 application: Click on the desired application package to navigate to its details page.



Creating an application

1. Log in to the TKE console and select **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" window that appears, configure and create the application as needed, as shown in the following figure:



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

Network Management
GlobalRouter mode
GlobalRouter Mode

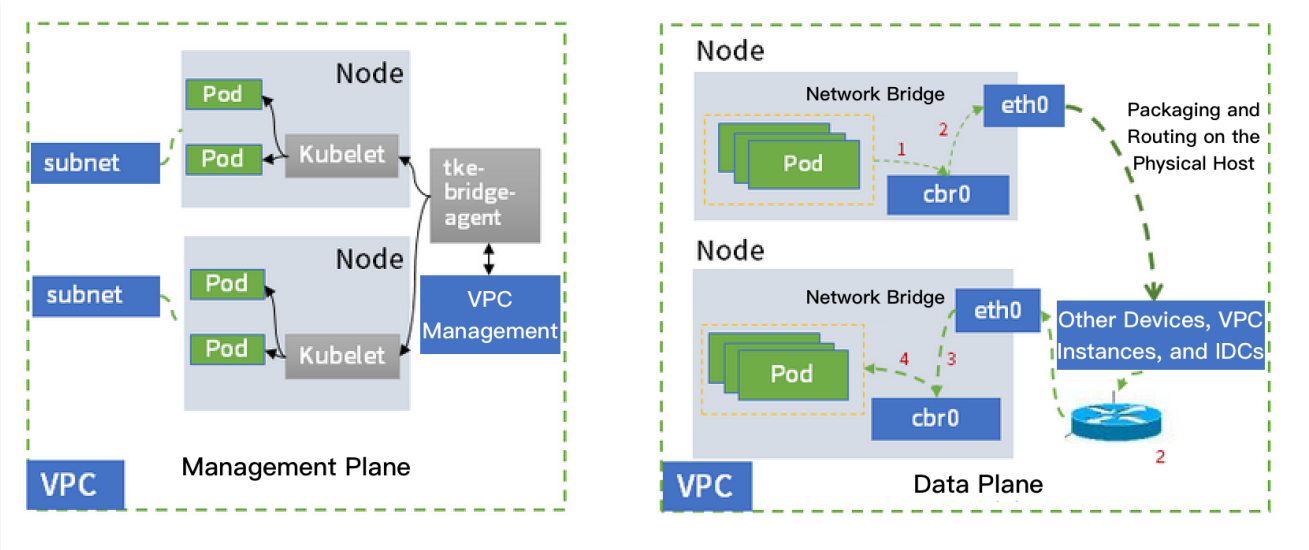
Last updated: 2023-09-26 15:20:28

How It Works

The GlobalRouter network mode is a global routing capability provided by Tencent Kubernetes Engine (TKE) based on the underlying Virtual Private Cloud (VPC). It implements a routing policy for mutual access between the container network and the VPC. The features of this network mode include the following:

- 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 network mode is suitable for general scenarios and can be seamlessly integrated with standard Kubernetes features. The following diagram illustrates the working principle:



Usage Limits

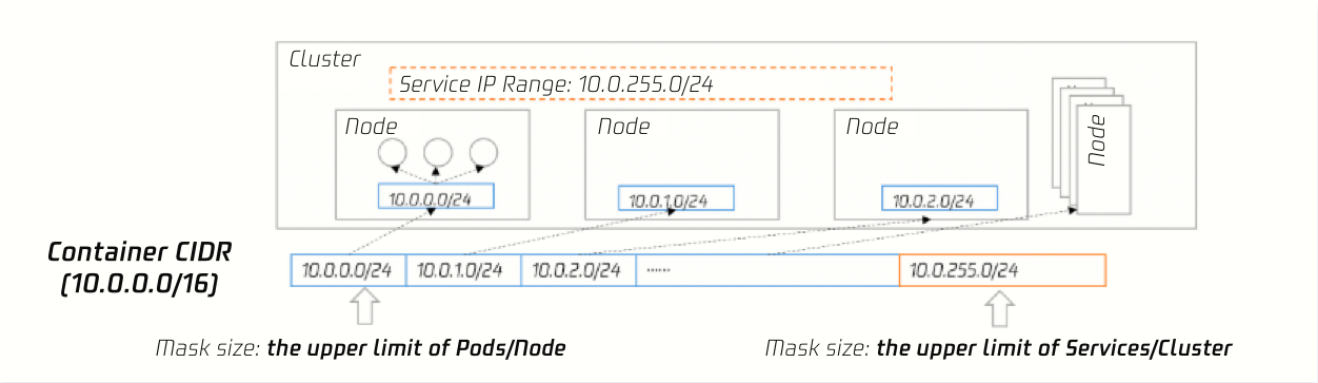
- The IP ranges of the cluster network and container network cannot overlap.
- The IP ranges of container networks of different clusters in the same VPC instance cannot overlap.
- If the container network and VPC route overlap, the traffic will be preferentially forwarded within the container network.
- Static Pod IPs are not supported.

Container IP Assignment Mechanism

For container network terms and quantity calculation, see Container Network Overview.

Pod IP assignment

As shown below:



Note:

- 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.

Interconnection Between Intra-region and Cross-region Clusters in GlobalRouter Mode

Last updated: 2023-09-27 09:32:36

Scenario

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.

Preparations

- 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.
- Refer to Creating a Peering Connection to establish a peering connection. Ensure that the peering connection is successfully established and that the instances can communicate with each other. If there are issues with the peering connection, check whether the **console route table entries, CVM security group, and subnet ACL** settings are configured correctly.

Instructions

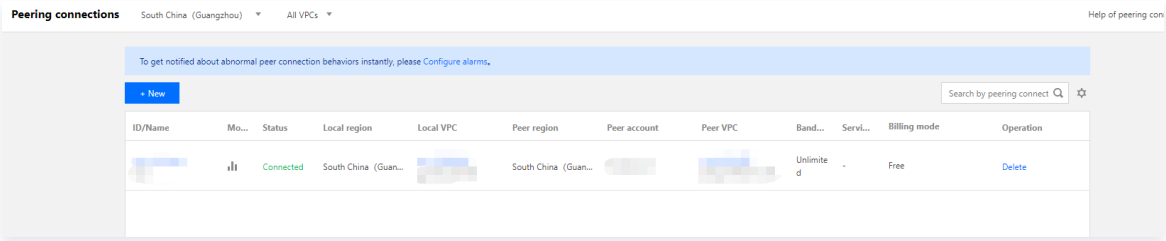
> **Note**
> To achieve cross-region communication between clusters, please complete the following steps and then contact support to request the container routes to be connected, enabling container-to-container communication.

Getting container information

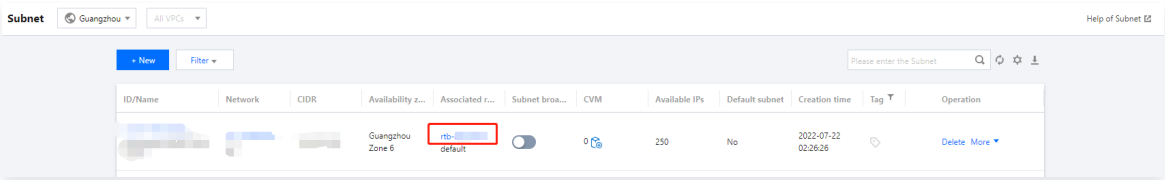
1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click the ID or name of the cluster for which you need to set up inter-cluster communication to go to the details page of that cluster. For example, navigate to the "Basic Information" page of Cluster A.
3. Record the following information: **Region**, **Node Network**, and **Container Network**.
4. Repeat Step 3 – Step 4 to record the "Region", "Node Network", and "Container Network" information of another cluster's containers. For example, navigate to the "Basic Information" page of Cluster B and record the "Region", "Node Network", and "Container Network" information of Cluster B's containers.

Configuring route tables

1. Log in to the VPC console and select Peering Connection in the left sidebar.
2. On the Peering Connection Management page, note down the **ID/Name** of the peering connection, as shown below:



3. In the left sidebar, click Subnets to access the subnet management page.
4. Click the associated route table for the local subnet of the peering connection, as shown below:



5. On the "Default Details" page of the associated route table, click **+Add Routing Policy**.
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 Result

- **Intra-region peering**: the above directions should allow containers in different clusters to communicate.
- **Inter-region clusters**: After the peering connection is successfully established, please contact support to enable container routing and achieve communication between 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 of Cluster A and access the container of Cluster B from within Cluster A's container, as shown in the following diagram:

```
[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 from within the container in Cluster B, as shown in the following diagram:

```
[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 /]#
```

Registering GlobalRouter Mode Cluster to CCN

Last updated: 2023-09-26 20:57:30

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.

Scenario

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 to CCN, its IP range will be enabled when it does not conflict with the existing routing in the CCN instance, and it will be disabled by default when there are conflicts.

Preparations

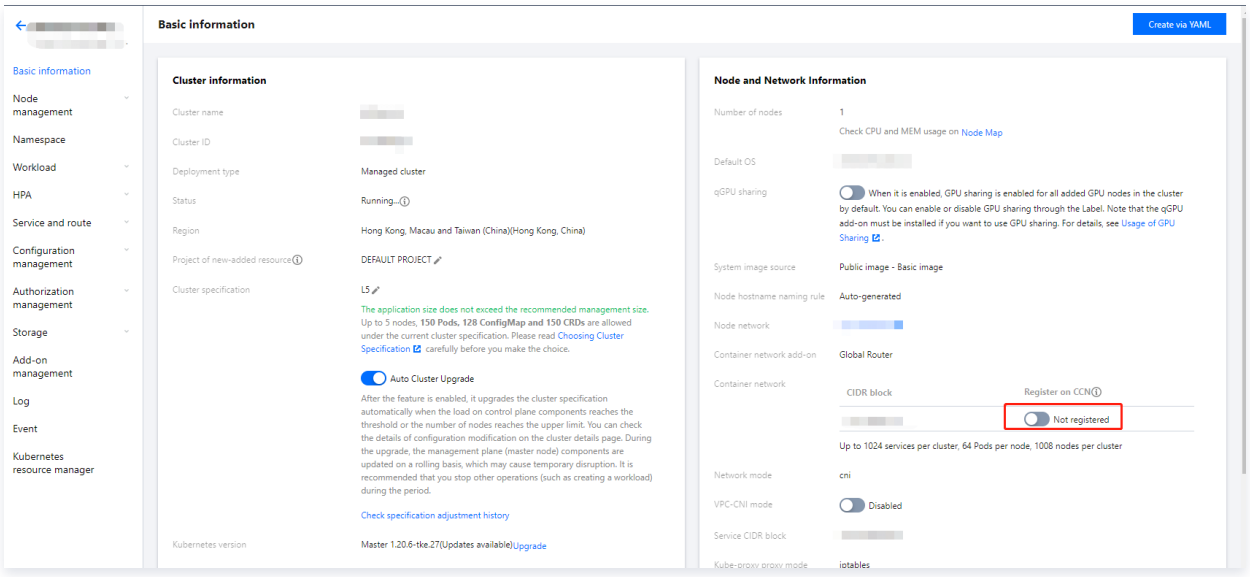
- The VPC of the cluster is already in CCN. For more information about the operations related to CCN, see Operations Overview.
- You have evaluated whether the IP range of the cluster network conflicts with the IP ranges of other resources in CCN.

Instructions

Registering Container Network to CCN

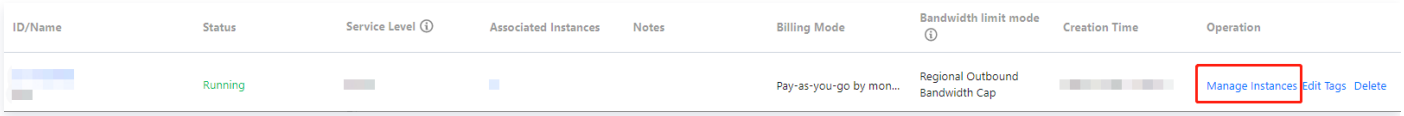
1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click the ID of the cluster you want to register with CCN, and click **Basic information** on the left to go to the cluster basic information page.
3. Turn on the CCN registration switch to register the container network with CCN as shown below:

> **Note**
> This step only registers the container IP range to the CCN. Whether or not the routing takes effect depends on your configuration on the CCN side.



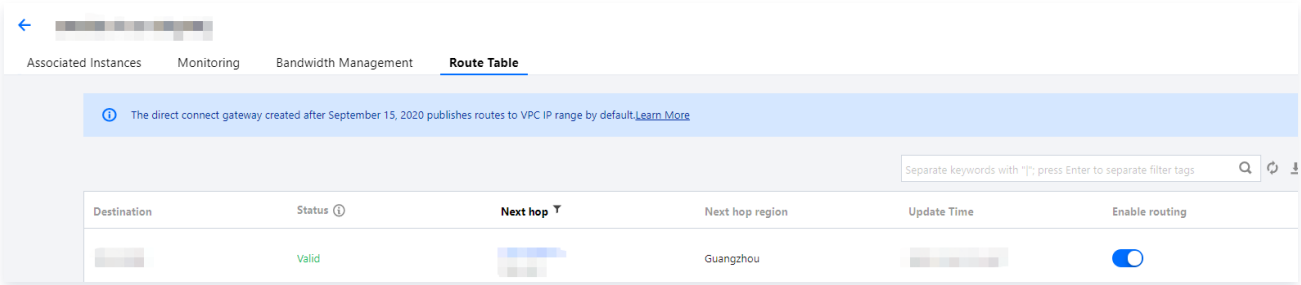
Viewing the IP Range Routing of the Container

1. Log in to the VPC console and click CCN in the left sidebar to go to the CCN management page.
2. Click **Manage Instance** on the right side of the row where the CCN associated with the cluster VPC is located to enter the CCN instance management page, as shown below:



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.



4. Test the interconnection between the container cluster and other resources in CCN.

VPC-CNI mode
VPC-CNI Mode

Last updated: 2023-09-26 15:50:59

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.

Usage 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.

Scenarios

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.

Pods with Exclusive ENI Mode

Last updated: 2023-09-26 15:56:06

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 testing. You can access it by applying for the beta test.

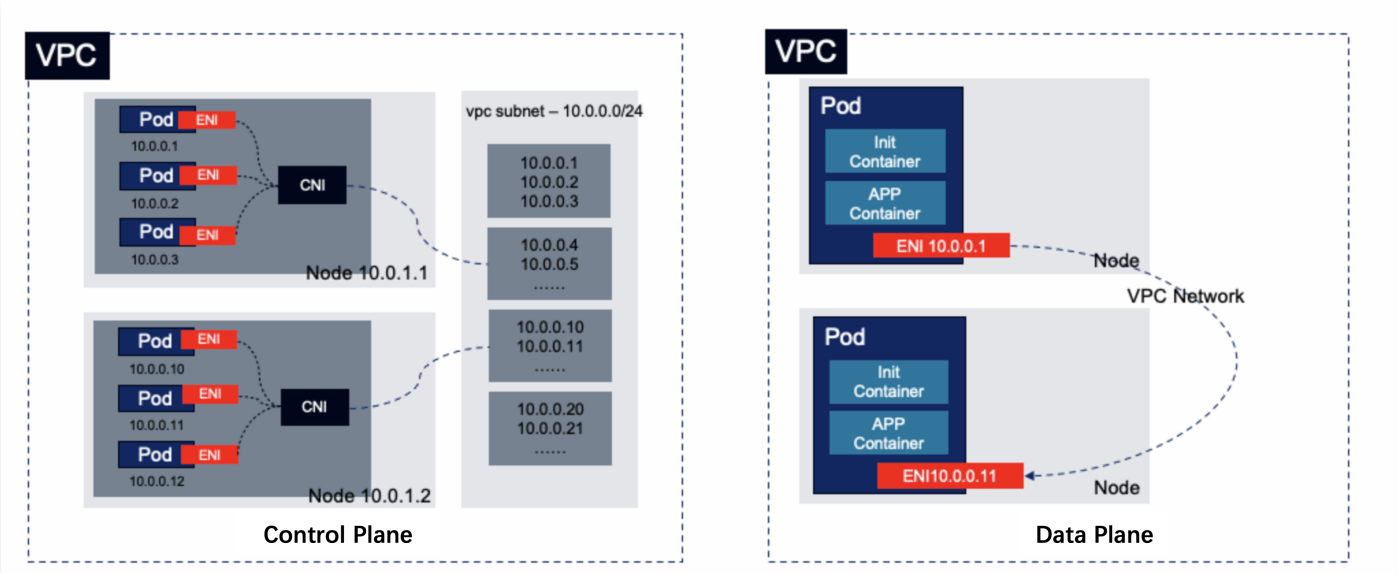
Overview

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.

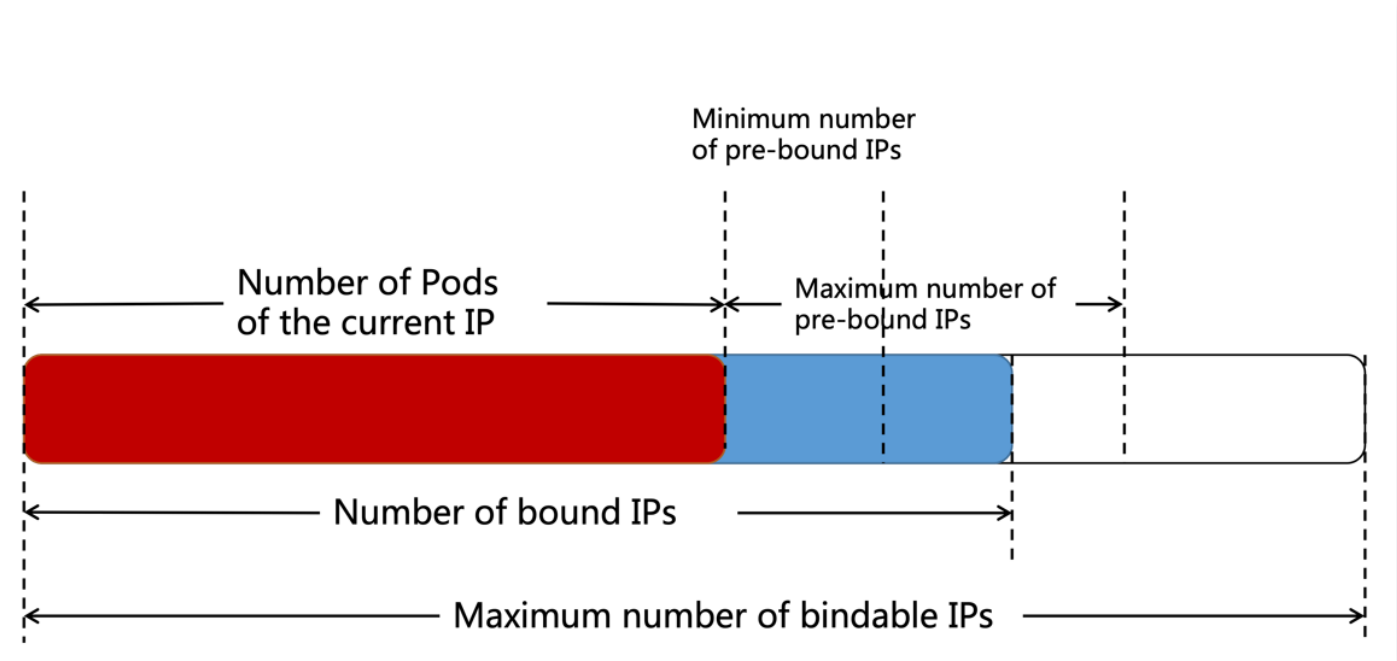
Implementation Method

The new generation solution expands upon the original VPC-CNI mode, relying on ENIs to bind the ENIs to nodes and configure them into the container network namespace through CNI, enabling containers to directly use the ENIs exclusively. The implementation principle is shown in the following diagram:



IP Address Management Principle

Non-static IP address mode



- 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.

Feature limits

- The network mode is only available to some models such as S5, SA2, IT5 and SA3.
- The number of Pods running on a node with an independent ENI solution is limited by the number of ENIs that can be bound to the instance type. The maximum number is the maximum number of bindable ENIs minus 1. For more information, see Pod Limitations 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: 2023-09-26 15:56:45

Use Cases

Applicable for scenarios that rely on static container IPs, such as migrating traditional architectures to container platforms and implementing security policies based on IP restrictions. For services without IP restrictions, it is not recommended to use the fixed IP mode.

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.
- Supports multiple subnets, but does not allow scheduling of Pods with static IPs across subnets. Therefore, Pods in static IP mode cannot be scheduled across availability zones.
- 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

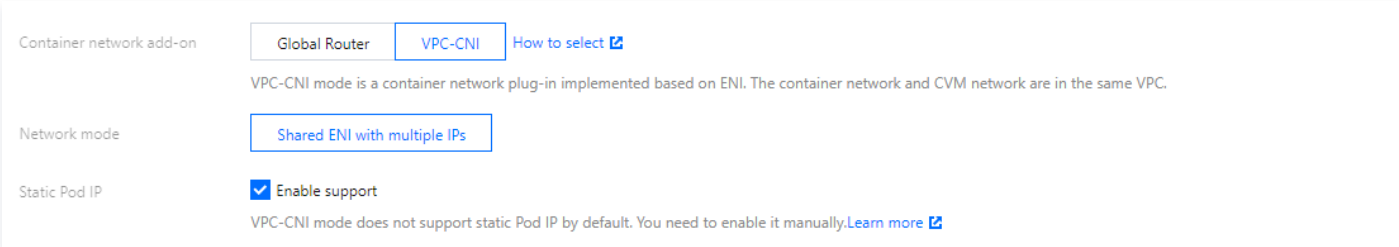
You can enable the static IP address using either of the following methods:

- 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** in the left sidebar.
2. On the **Cluster Management** page, click **New** at the top of the cluster list.
3. On the **Create Cluster** page, select the container network add-on as **VPC-CNI** and check the "Enable Support" for fixed Pod IP as shown in the image below:

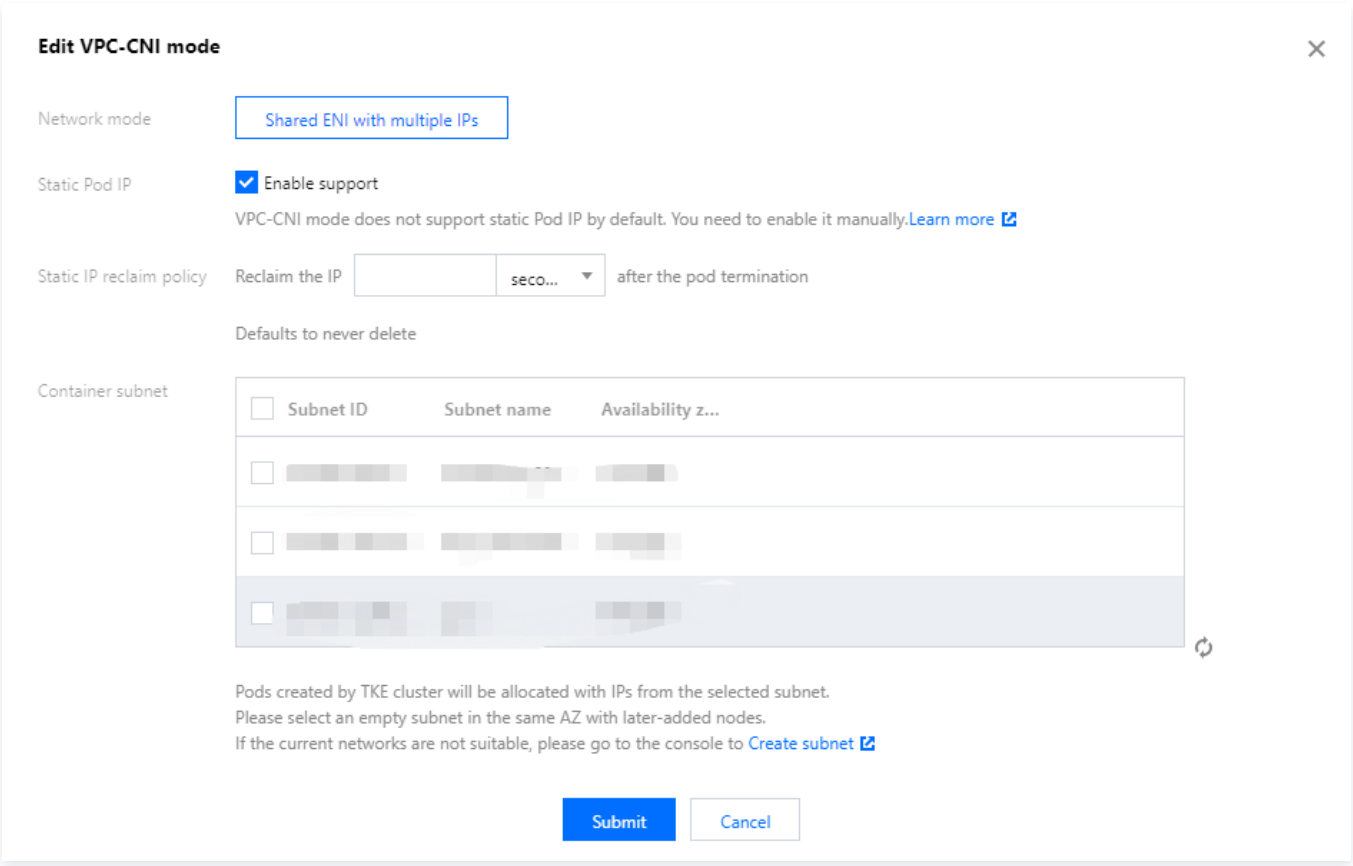


Enabling VPC-CNI with static IP address for GlobalRouter mode

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** in the left sidebar.
2. On the **Cluster Management** page, select the cluster ID for which you want to enable VPC-CNI, and proceed to the cluster details page.
3. In the **Cluster Details** page, select **Basic Information** on the left. In the Cluster Information section, locate the VPC-CNI field and click **Enable**.
4. In the pop-up window, check "Enable Support" for fixed Pod IP, confirm the IP reclaim policy, and select a subnet, as shown in the figure below:



> **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.

5. 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.

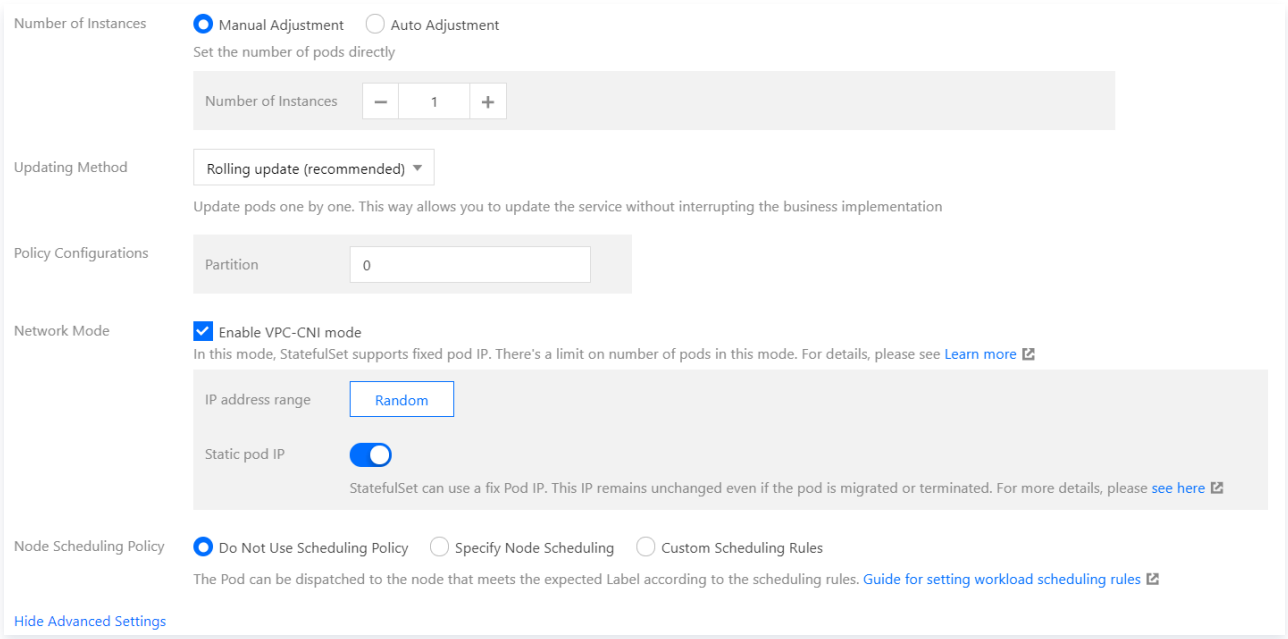
> **Note**

> 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:

Via the console

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. Select a cluster ID that needs to use the static IP address and go to its management page.
3. Select **Workload** > **StatefulSet** to access the **StatefulSet** cluster management page.
4. Click **Create**. In the new StatefulSet, select **Network Mode** > **Enable VPC-CNI mode** and enable **Fixed Pod IP**, as shown in the image below:



   - IP address range: currently, only the **Random** value is supported.
   - Static pod IP: select **Enable**.

Using 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 shared ENI VPC-CNI mode. If using the independent ENI VPC-CNI mode, please change 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 a Pod in VPC-CNI mode with a shared ENI, add requests and limits restrictions, i.e.,** `tke.cloud.tencent.com/eni-ip` **. For VPC-CNI mode with an independent ENI, add** `tke.cloud.tencent.com/direct-eni` **. This resource only needs to be added to the first container of the Pod, and other containers do not require it. Additionally, this resource will be added automatically by default, and you only need to declare the** `tke.cloud.tencent.com/networks` **annotation.**

Interconnection Between VPC-CNI and Other Cloud Resources/IDC Resources

Last updated: 2023-09-26 15:57:38

VPC-CNI mode and container networks belong to the manageable network segments of VPC, enabling direct communication with other cloud resources and IDC resources through VPC product feature configurations.
Tencent Cloud offers a wide range of solutions to meet the needs of connecting CVMs, databases, and other instances within a VPC to the public internet, other VPC instances, or local data centers (IDCs). For more information on VPC and its various connection methods, please refer to VPC Connection Solutions Overview.

Security Group of VPC-CNI Mode

Last updated: 2023-09-26 15:59:05

You can bind specified security group to the ENI created in VPC-CNI mode through the following methods:

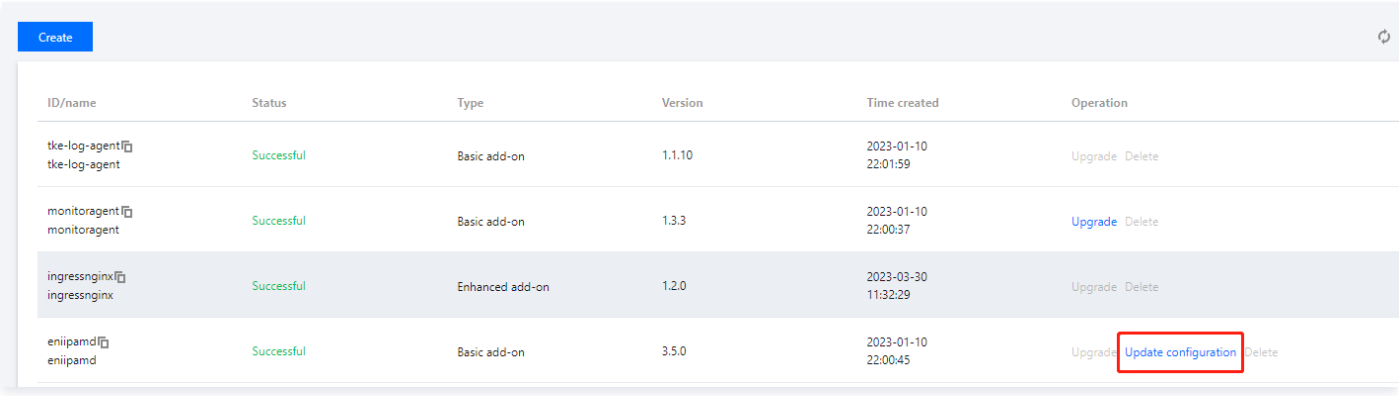
Preparations

- 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.

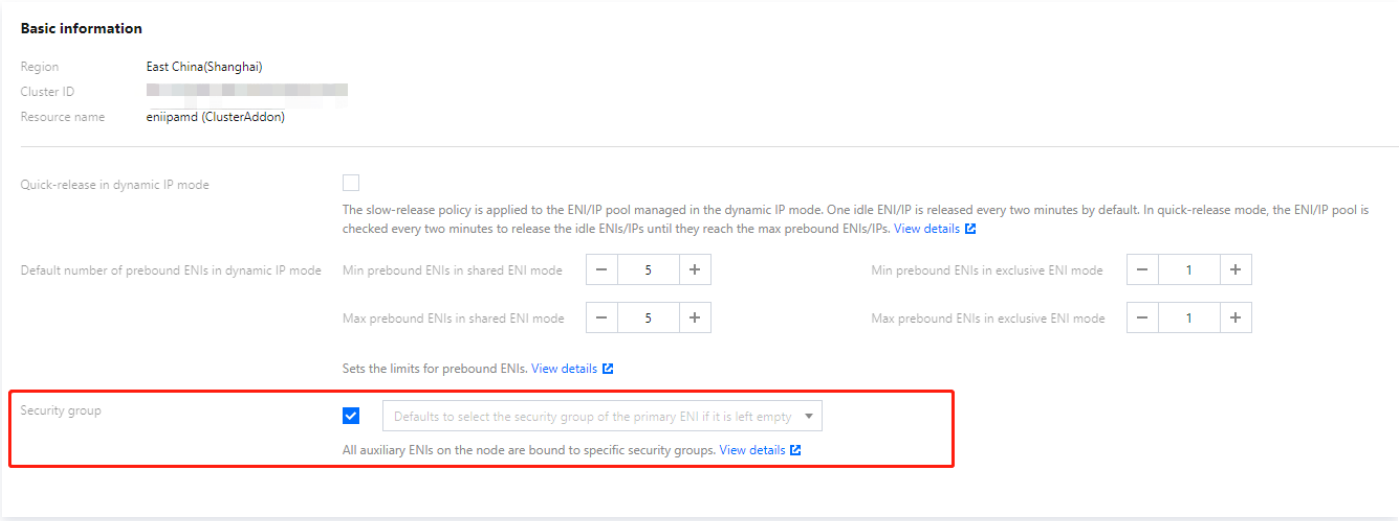
Enabling Security Group Feature for the IPAMD Component

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. In the cluster details page, select **Component Management** on the left side. In the Component Management page, click **eniipamd** component and then click **Update Configuration** on the right side.



4. On the configuration update page, select **Security Group**. If you wish to inherit the primary ENI's security group, do not specify a security group; otherwise, you need to specify one.



5. Click **Finish**.

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
```

- Execute the following command to add the launch parameter in `spec.template.spec.containers[0].args`. After modification, the IPAMD will automatically restart and take effect.

  Once effective, the auxiliary ENIs on existing nodes without associated security groups will be bound to the security group according to the following policy. If they are already bound, they will be strongly synchronized with the set security group, unless the feature has been previously enabled and the node security group is already set. The ENIs on new nodes will be bound to the following security groups.

```
- --enable-security-groups
# If you want to inherit the security group from the primary ENI/instance by default, do not add the security-groups
parameter.
- --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-security-groups="yes"
```

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-security-groups="no"
```

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: 2023-09-26 21:01:08

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 Access Management console and select **Roles** on the left sidebar.
2. In the **Access Management Console**, search for the IPAMD component's related role `IPAMDofTKE_QCSRole` under Roles. Click on the role name to enter the role details page.
3. Click **Associate Policies**.
4. In the pop-up window for associating policies, search for `QcloudAccessForIPAMDRoleInQcloudAllocateEIP` in the search box, and then select the created preset policy `QcloudAccessForIPAMDRoleInQcloudAllocateEIP`. Click **OK** to complete adding EIP API access permission for the IPAMD component role. This policy includes all the required permissions for the IPAMD component to operate EIPs.

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 of this Workload needs to be automatically associated with an EIP, with a bandwidth of 100 Mbps and a BGP line type.
- `spec.template.annotations: tke.cloud.tencent.com/eip-claim-delete-policy: "Never"` indicates that the EIP of the Workload's Pod should remain fixed and not change after the Pod is terminated. If a fixed EIP is not required, do not add this annotation.
- For Pods that require an associated EIP, you need to add requests and limits constraints in `spec.template.spec.containers.0.resources`, specifically `tke.cloud.tencent.com/eip`, to ensure that the scheduler places the Pod on a node with available EIP resources.

Key configurations

- The EIP resources that can be bound to each node are subject to relevant quota restrictions and the binding limits of CVMs. For more information, see EIP Usage Limits.
  The maximum number of EIPs that can be bound to each node is **CVM binding limit − 1**.
- `tke.cloud.tencent.com/eip-attributes: '{"Bandwidth":"100","ISP":"BGP"}'`: Currently, only bandwidth and line type parameters are supported. The `ISP` parameter can be set to `BGP`, `CMCC`, `CTCC`, or `CUCC`, corresponding to general BGP IP, static single-line IP (network operators China Mobile, China Telecom, and China Unicom). If not specified, the default values are 100Mbps and BGP.
- After the auto-applied EIP is bound, no IP resource fees are charged. The default billing method for accessing the public network is **Pay-as-you-go by hourly traffic**. For more details, see EIP Billing Overview.

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 Pods of this workload should automatically associate with the specified EIPs, with the first replica using the EIP with eipID `eip-xxx1` and the second replica using the EIP with eipID `eip-xxx2`. The current parsing policy is: Pods select EIPs from the annotations based on the index at the end of their names. If there is no index (such as in Deployment), a random EIP is selected, and only one Pod can associate successfully in case of conflict. It is recommended that Pods without an index specify only a single EIP.
- For Pods that require an associated EIP, you need to add requests and limits constraints in `spec.template.spec.containers.0.resources`, specifically `tke.cloud.tencent.com/eip`, to ensure that the scheduler places the Pod on a node with available EIP resources.

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
```

Add a new field with the key `NonMasqueradeSrcCIDRs` in the `data.config` field, and set the value to the list of private IP address ranges of Pods associated with EIPs, such as **internal IP**. For example, if the IP is `172.16.0.2`, enter `172.16.0.2/32`. Here 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

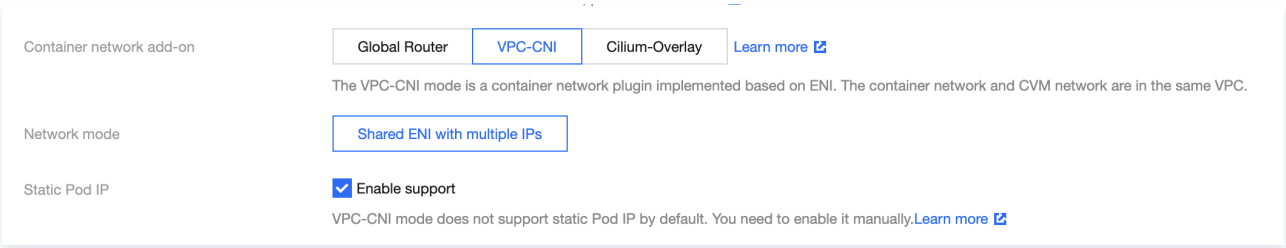
After "auto-associate with an EIP" is enabled for the Pod, the network component will create a CRD object `EIPClaim` with the same name of the Pod in the same namespace. This object describes the Pod's requirements for the EIP.
For Pods with non-static EIPs, the `EIPClaim` will be destroyed upon Pod termination, and the associated EIP will be destroyed and reclaimed. However, for Pods with static EIPs, the `EIPClaim` will be retained after the Pod is terminated, and the EIP will also be

retained. When a **same-named** Pod is launched, it will use the EIP associated with the same-named `EIPClaim`, thus retaining the EIP.
Below are three methods for reclaiming an EIP, including reclaiming after expiration, manual reclaiming and cascade reclaiming.

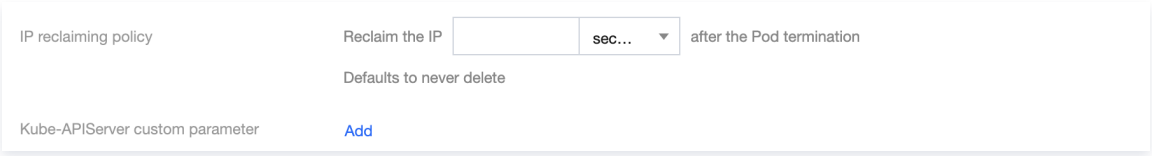
Reclaiming after expiration

Create cluster

On the Create Cluster page, select the **VPC-CNI** mode for the container network plugin and check **Enable Support** for static Pod IP, as shown below:



In Advanced Settings, configure the IP reclaim policy by setting the time to reclaim the retained static IPs upon Pod termination, as shown below:

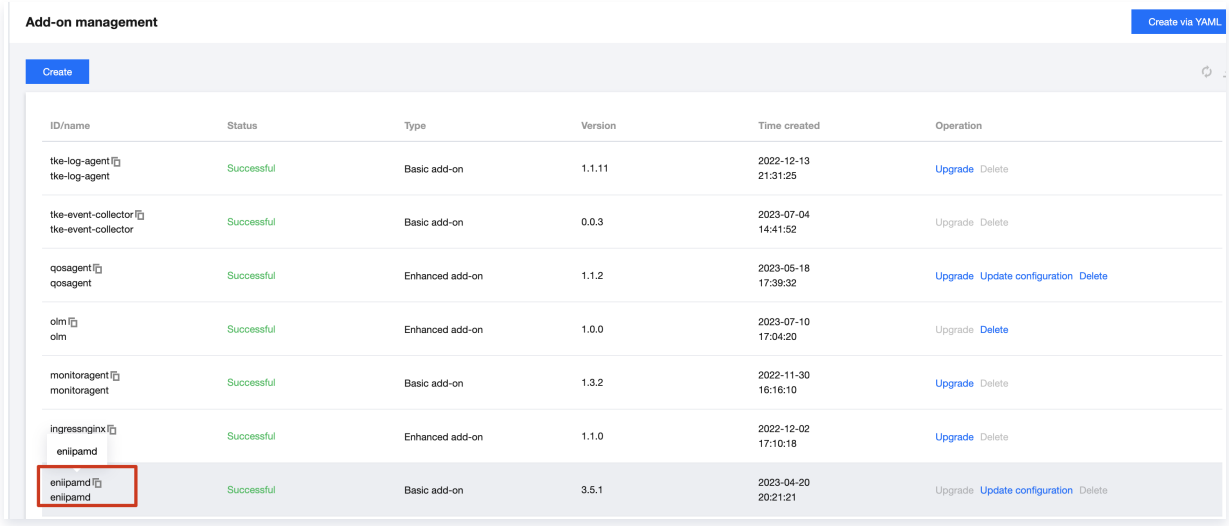


Legacy Cluster

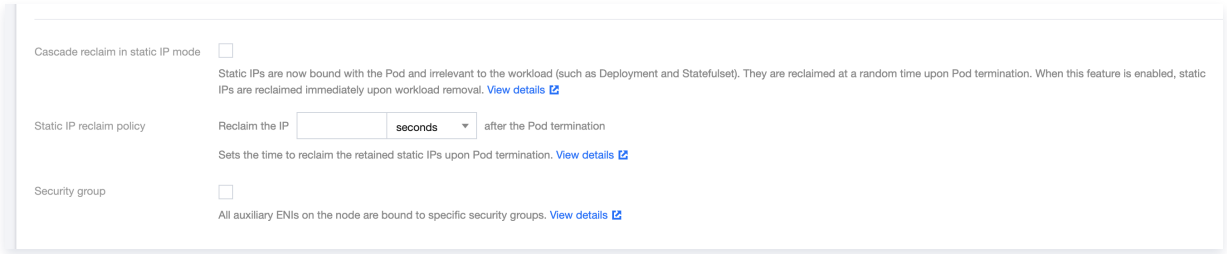
You can modify the **existing clusters** with the following method:

tke-eni-ipamd v3.5.0 or later

1. Log in to the TKE console and 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** in the left sidebar.
4. On the Add-On Management page, locate the **eniipamd** component and select **Update Configuration**.



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 # Enter any value greater than or equal to 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.

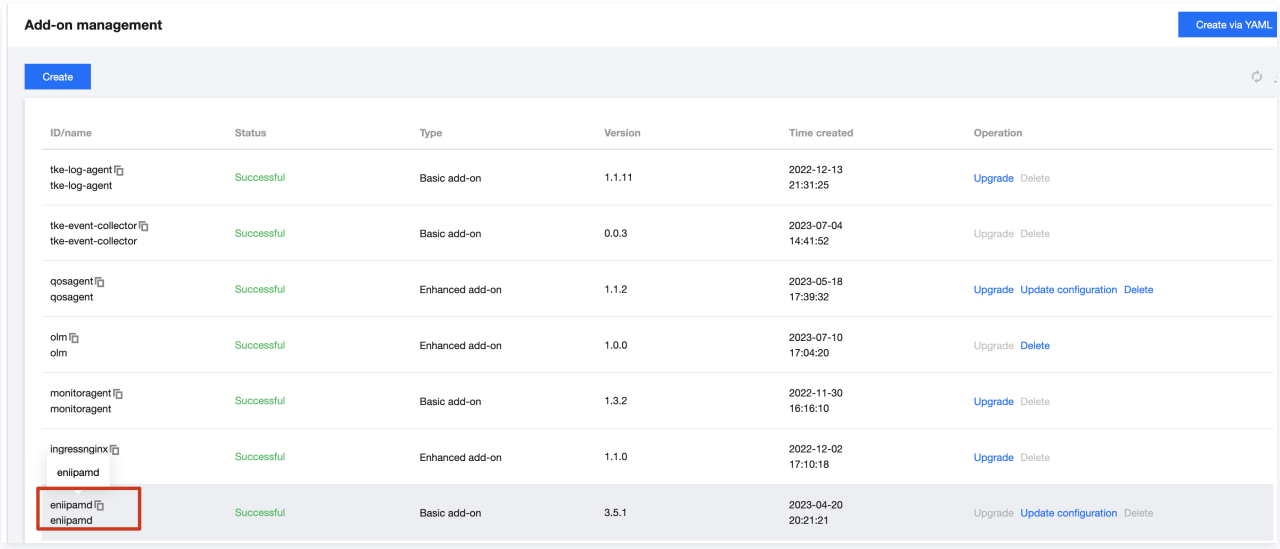
```
kubectl delete eipc <podname> -n <namespace>
```

Cascade reclaiming

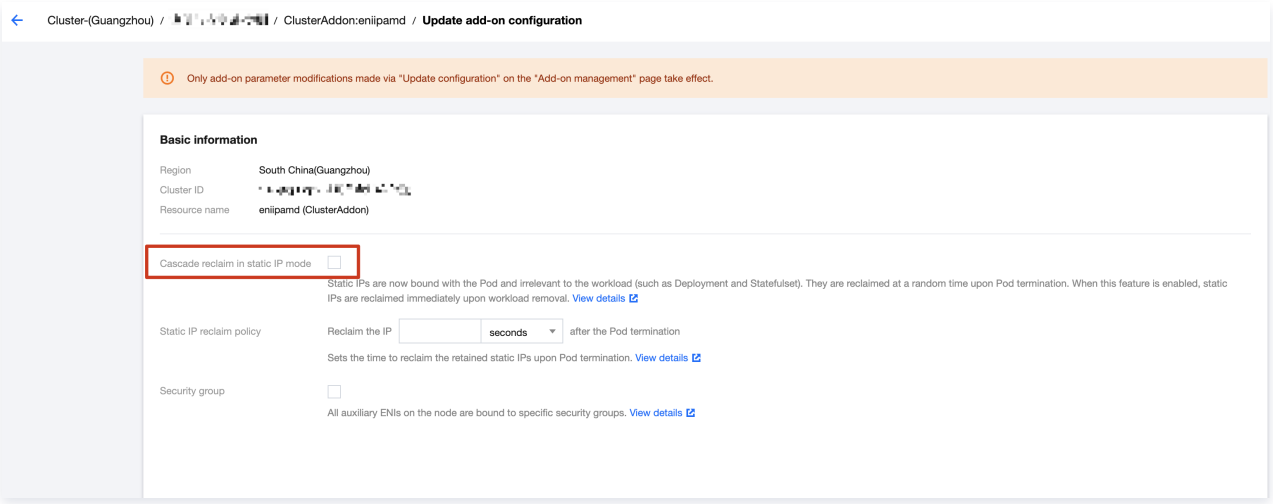
Currently, the static EIP is strongly bound to the Pod, regardless of the specific workload (e.g., deployment, statefulset). After the Pod is terminated, it is uncertain when to reclaim the static EIP. TKE has implemented that the static EIP is deleted once the workload to which the Pod belongs is deleted. **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 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** in the left sidebar.
4. On the Add-On Management page, locate the **eniipamd** component and select **Update Configuration**.



5. On the Update Configuration page, check the **Enable Tiered Recycling** option and click Finish.



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 modification, ipamd will automatically restart and take effect. Once effective, incremental Workloads can support cascading deletion of static EIPs, while existing Workloads cannot support this feature temporarily.

Configuring EIP public address on the ENI within the Pod

You can use the downwardAPI and Volumes to map the EIP address bound to a Pod to the network interface within the Pod, allowing the EIP public address to be obtained within the Pod. Refer to the following YAML configuration:

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    tke.cloud.tencent.com/eip-attributes: ''
  labels:
    app: busybox-eip
  name: busybox-eip
spec:
```

```
initContainers:
- name: get-eip
  image: busybox
  imagePullPolicy: IfNotPresent
  command: ["sh", "-c"]
  args:
  - while true; do
      if [[ -e /etc/podinfo/eip ]]; then
        ip=$(cat /etc/podinfo/eip);
        echo "$ip\n";
        if [ $ip ];then
          echo "found eip ${ip}, set it to eth0";
          ip addr add $ip dev eth0;
          break;
        else
          echo "wait eip associated";
        fi;
      else
        echo "eip file not found";
      fi;
      sleep 1;
    done;
  securityContext:
    privileged: true
  volumeMounts:
  - mountPath: /etc/podinfo
    name: podinfo
containers:
- command: ["sleep", "2300000"]
  image: busybox
  imagePullPolicy: IfNotPresent
  name: busybox
  resources:
    requests:
      tke.cloud.tencent.com/eip: "1"
    limits:
      tke.cloud.tencent.com/eip: "1"
volumes:
  - name: podinfo
    downwardAPI:
      items:
        - path: "eip"
          fieldRef:
            fieldPath: metadata.annotations['tke.cloud.tencent.com/eip-public-ip']
```

- Once the EIP is successfully bound to the Pod, the public IP address will be written to the Pod annotation `tke.cloud.tencent.com/eip-public-ip`. It can be mounted within the Pod using downwardAPI and volumes.
- The initContainer executes the script code, polling the EIP public address file until it exists, and then configures the address on the eth0 network interface.

VPC-CNI Component Description

Last updated: 2023-09-26 15:59:51

The VPC-CNI component comprises three Kubernetes cluster components: `tke-eni-agent`, `tke-eni-ipamd`, and `tke-eni-ip-scheduler`.

tke-eni-agent

Deployed as a `daemonset` on each node within the cluster, its responsibilities include:

- 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 up the ENI and IP extended resources using Kubernetes' device-plugin mechanism.

tke-eni-ipamd

Deployed as a `deployment` on specific nodes or the master within the cluster, its responsibilities include:

- 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

Deployed as a `deployment` on specific nodes or master within the cluster, it is only deployed in fixed IP mode as a scheduler extension plugin, with responsibilities including:

- 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.

Limits on the Number of Pods in VPC-CNI Mode

Last updated: 2023-09-26 16:00:08

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
Maximum Number of Secondary ENIs that can be Bound	1	3	5	7
Number of Secondary IPs that can be Bound on a Single ENI	5	9	19	29
Max Pod IPs per Node (Multiple ENIs)	5	27	95	203
Max Pod IPs per Node (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

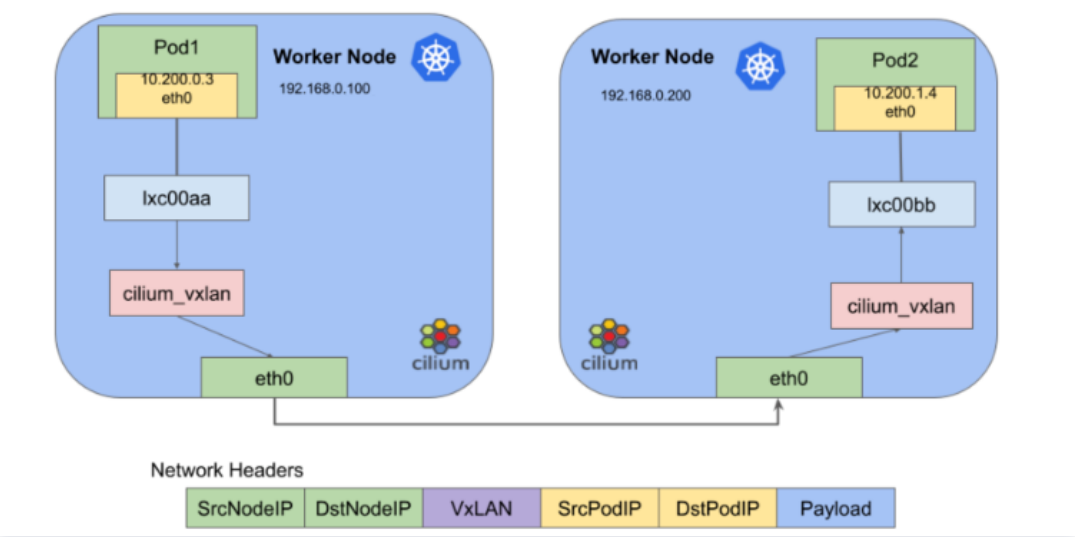
Last updated: 2023-09-26 16:00:23

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.
- Cilium mode does not support Webhooks.

Usage Limits

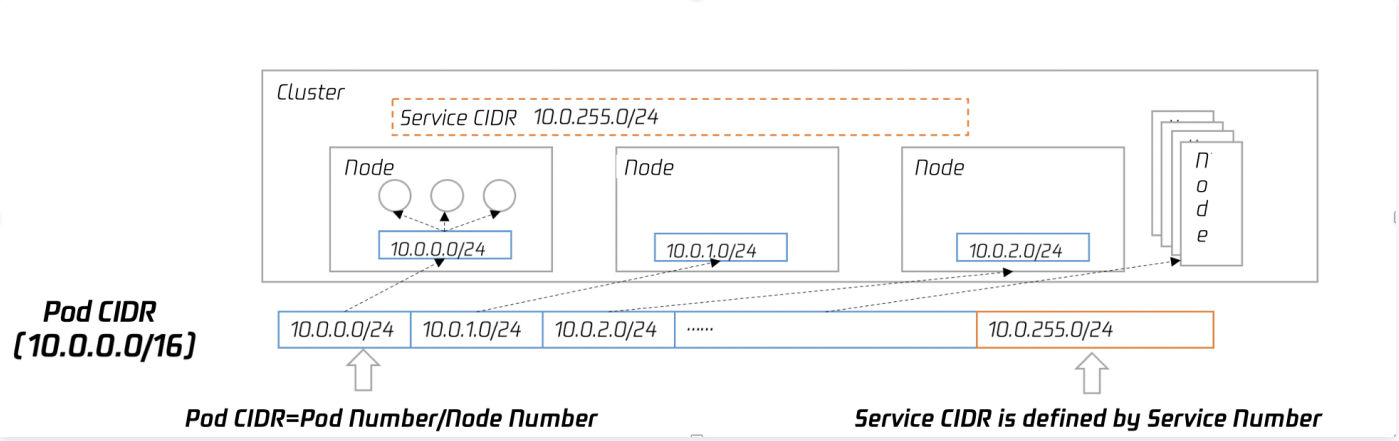
- There is a performance loss of less than 10% when you use the Cilium VXLan tunnel encapsulation protocol.
- The Pod IP cannot be accessed directly outside the cluster.
- You must obtain two IPs from the specified subnet to create a private CLB, so that external nodes in IDC can access APIServer and 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

As shown below:



- 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.

Cluster Operations
Audit Management
Cluster Audit

Last updated: 2023-09-26 21:03:57

Reminder
CLS offers **free services** for all audit and event data generated by TKE until June 30, 2022. Please select "Auto-create logset" or choose "Auto-create log topic" in an existing logset. For more information, refer to TKE Container Service Audit and Event Center Log Free Description.

Feature Overview

Cluster audit is a feature based on Kubernetes Audit that can store and search the records of JSON logs with configurable policies generated by kube-apiserver. 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.

Strengths

The cluster audit feature provides another cluster monitoring dimension different from metrics. After cluster audit is enabled, Kubernetes can record every audit log that operates on the cluster. An audit log is a structured record in JSON format, and includes three parts: metadata, requestObject, and responseObject. The metadata (containing the request context information, such as who initiated the request, where it was initiated, and the accessed URI) is a required part. requestObject and responseObject are optional, depending on the audit level. You can learn the following information from logs:

- Activities that occur in the cluster.
- Occurrence time and objects of an activity.
- Activity triggering time, triggering positions, and observation points.
- Activity results and subsequent processing.

Reading audit logs

```
{
 "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's 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 level (level)

Unlike common logs, the level of Kubernetes audit logs is more like a kind of verbose configuration, which is used to indicate the degree of detail of the recorded information. There are four audit levels, as listed in the following table:

Category	Note
None	Nothing is recorded.
Metadata	The metadata of the request (for example: 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 stage (stage)

Logs can be recorded at different stages, as listed in the following table:

Category	Note
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 only applies to persistent connection requests, such as watch.
ResponseComplete	The log is recorded after the response is completely 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 exceptions:

- For get, list, and watch, logs of the Request level are recorded.
- Requests for Secrets, ConfigMaps, or TokenReviews are logged at the Metadata level.

Logs will not be recorded for the following requests:

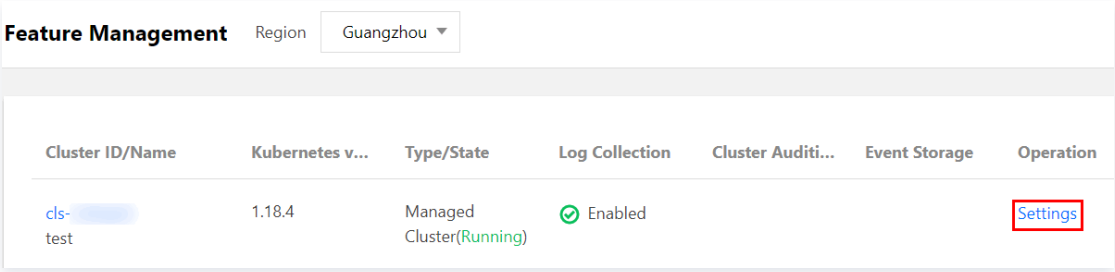
- `system:kube-proxy` issues requests to monitor endpoints resources, services resources, or services/status resources.
- `system:unsecured` GET requests targeting configmaps resources in the kube-system namespace.
- get requests sent by kubelet for nodes resources or nodes/status resources.
- `system:nodes` group issues GET requests for node or node/status resources.
- GET and UPDATE requests for endpoints resources in the kube-system namespace, issued by `system:kube-controller-manager`, `system:kube-scheduler`, or `system:serviceaccount:endpoint-controller`.
- `system:apiserver` GET requests for namespaces, namespaces/status, or namespaces/finalize resources.
- Requests sent to URLs that match `/healthz*`, `/version`, or `/swagger*`.

Instructions

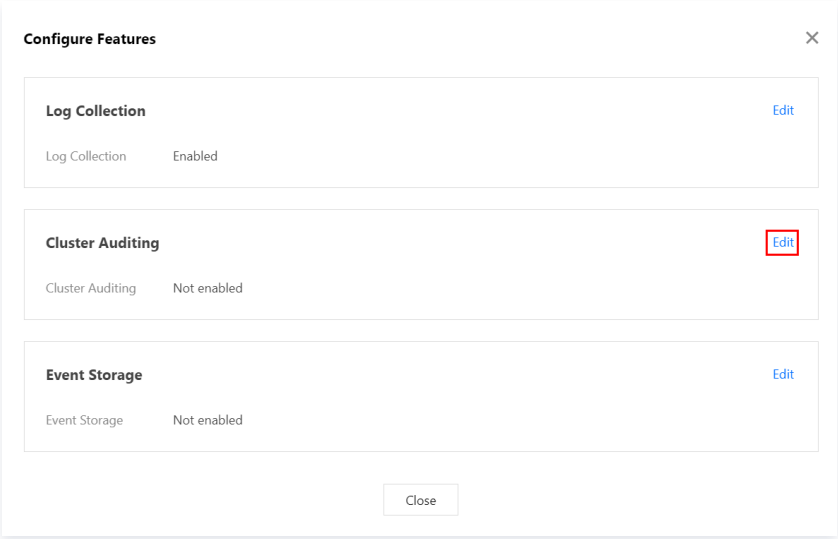
Enabling cluster audit

> **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 occupies about 1 Gib of local storage on the master node. Please ensure that the storage of the master node is sufficient.

1. Log in to the TKE console.
2. In the left sidebar, choose **Operation Management** > **Feature Management**.
3. At the top of the "Feature Management" page, select a region and click **Settings** on the right side of the cluster you wish to enable cluster auditing for, as shown below:

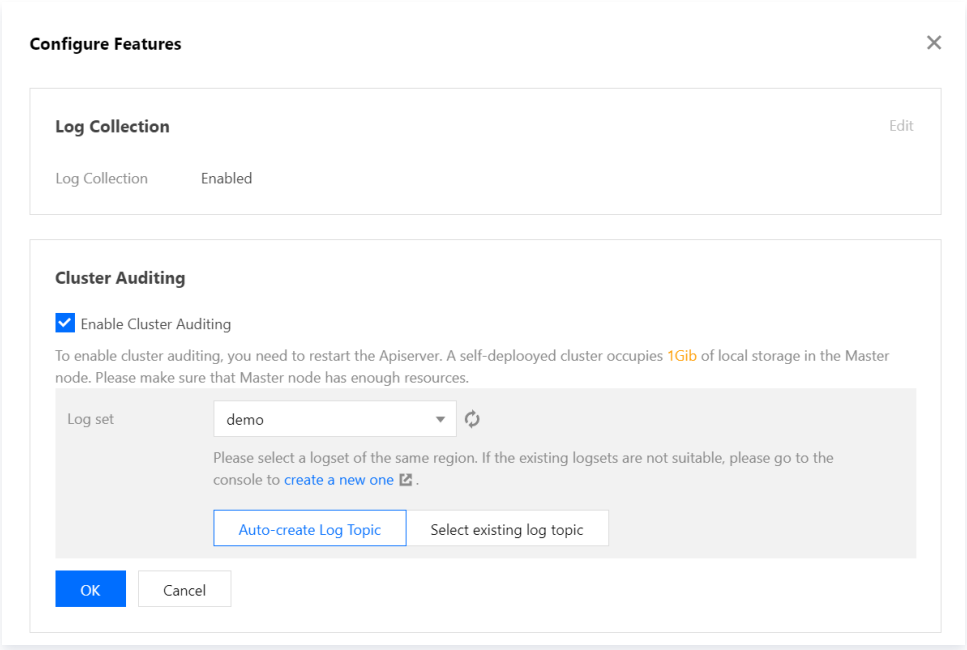


4. In the "Settings" pop-up window, click **Edit** on the right side of the "Cluster Audit" feature.



5. Check **Enable Cluster Auditing**. Select the logset and log topic for storing audit logs. We recommend that you select **Auto-create**

Log Topic.



6. Click **OK** to enable the cluster audit feature.

Event Management
Event Storage

Last updated: 2023-09-26 21:04:58

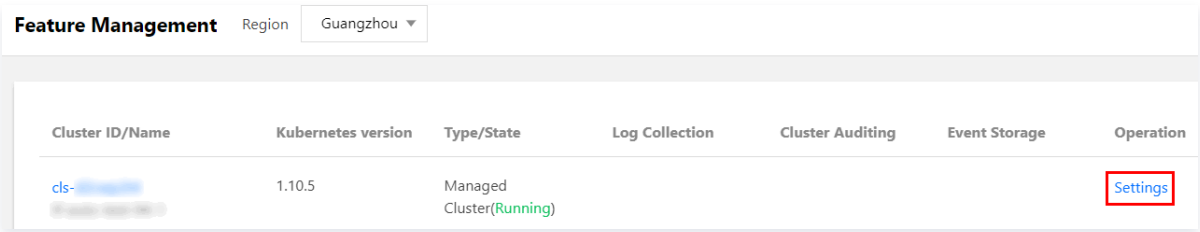
Scenario

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.

Instructions

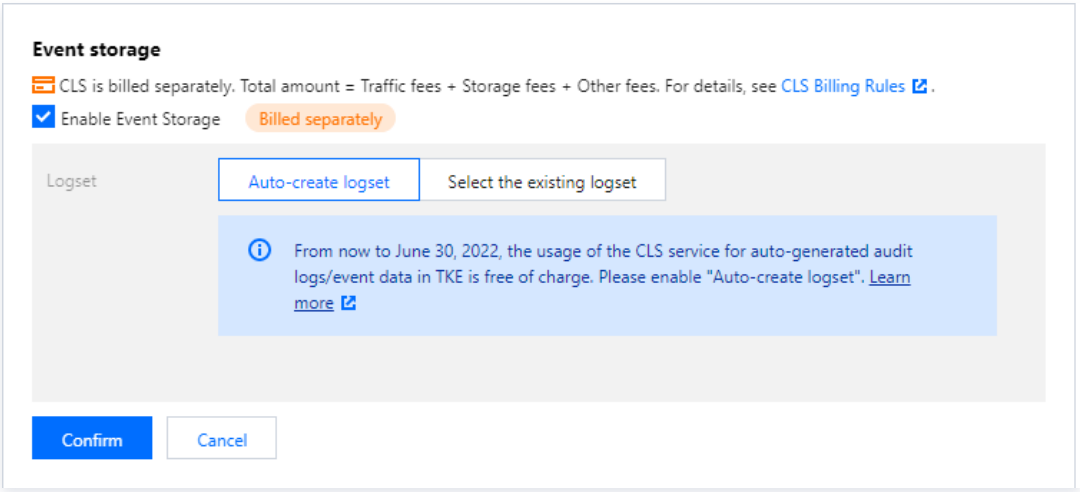
Enabling event storage

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:



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 logset contains less than 10 log topics.

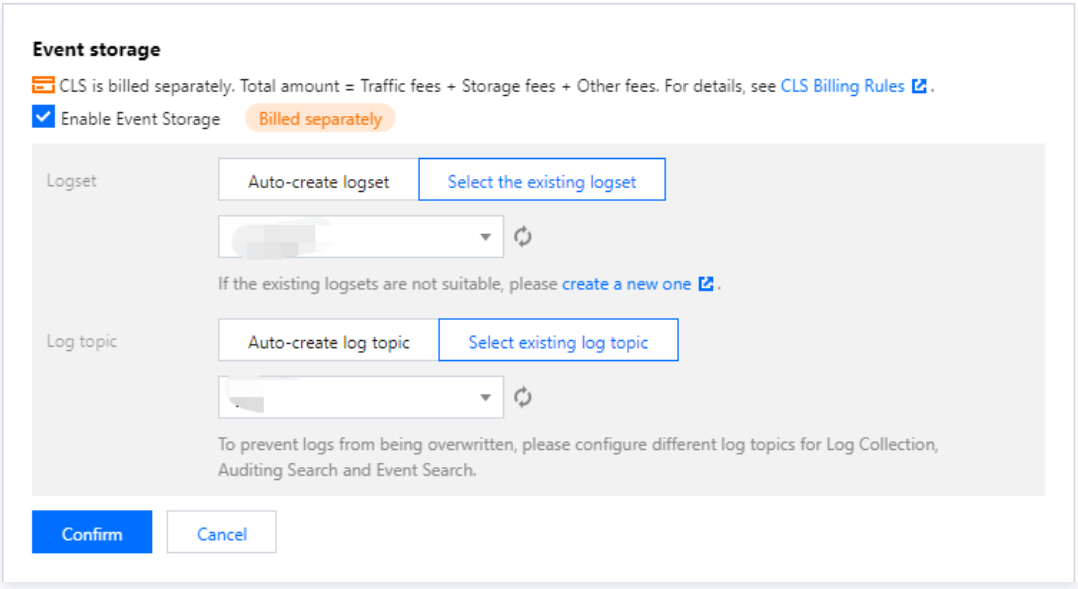


5. Click **OK** to enable event storage.

Updating logsets or log topics

1. Log in to the TKE console.
2. Click **Operation Management** in the left sidebar.

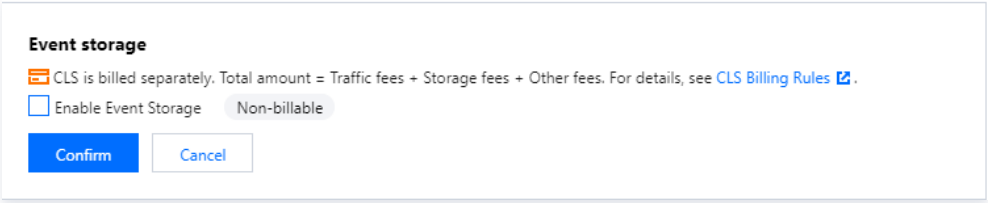
3. At the top of the **Feature Management** page, select a region and a cluster type. Locate the target cluster and click **Settings** on the right.
4. On the **Configure features** page, click **Edit** on the right for event storage, and reselect the logset and log topic, as shown below:



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:

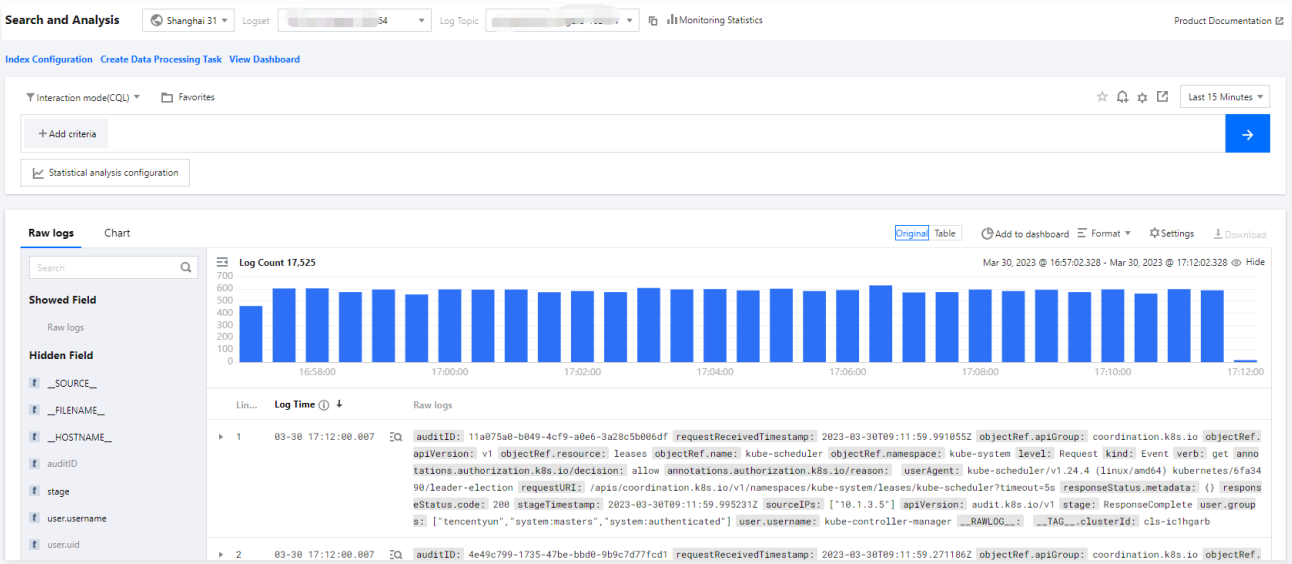


5. Click **OK** to disable event storage.

Viewing event in the CLS console

1. Log in to the Cloud Log Service console.
2. In the left sidebar, click **Search and Analysis**.
3. On the **Search and Analysis** page, select the logset and log topic configured in event storage, configure the display fields as needed, and perform search and analysis, as shown below:

> **Note:**
> When you enable event storage, indexing will be enabled for your log topic by default. For more information, see Log Search and Analysis.



Event Dashboard

Last updated: 2023-09-26 21:05:57

Scenario

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.

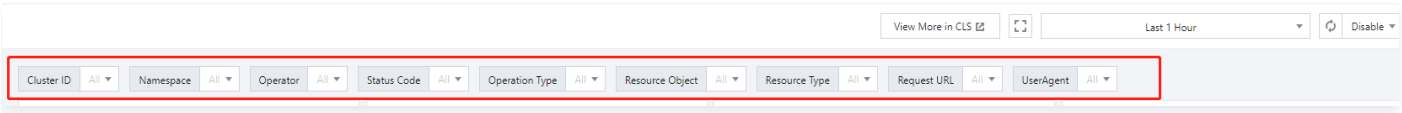
Feature Overview

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** feature. For more information, please refer to Event Storage.
3. In the left navigation pane, select **Log Management > Event Logs**.
4. At the top of the **Event Search** page, select the region and cluster type to view the cluster event details.

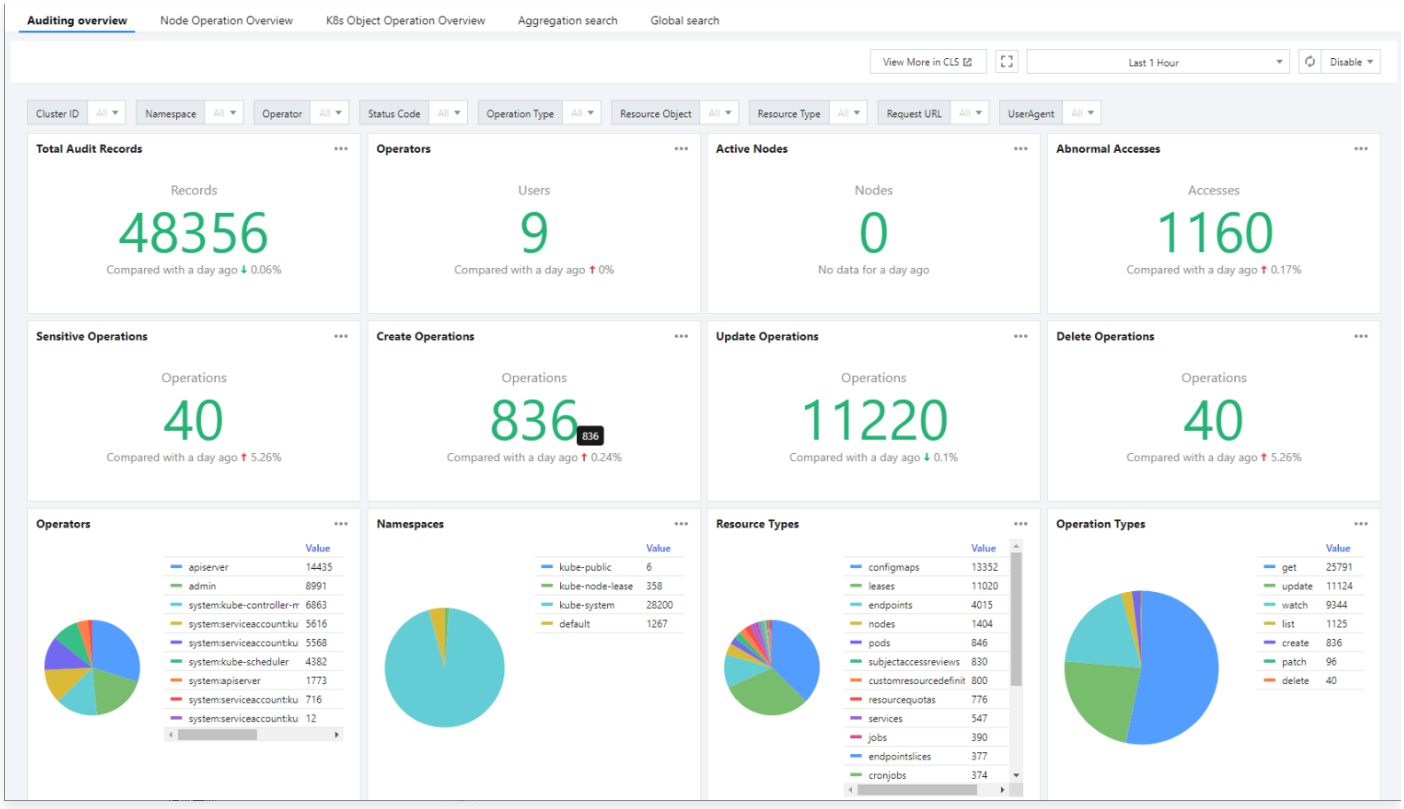
Event overview

On the **Event Overview** page, you can filter events based on dimensions such as cluster ID, namespace, level, reason, resource type, and resource object event source. You can view the summary statistics of core events and display data comparisons within a period. For example, the total number of events and distribution, node anomalies, Pod OOM, important event trends, and other dashboards, as well as the exception TOP event list.
In the filter items, you can customize the configuration according to your needs, as shown below:

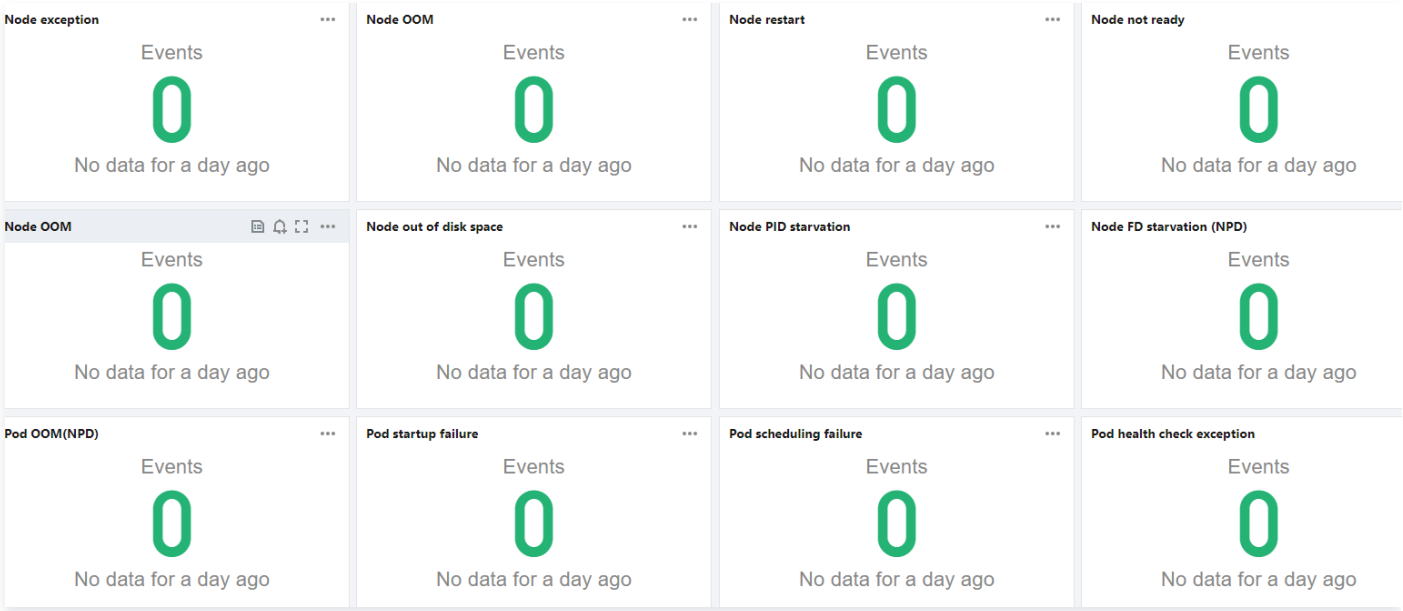


You can view more statistics on this page as shown below:

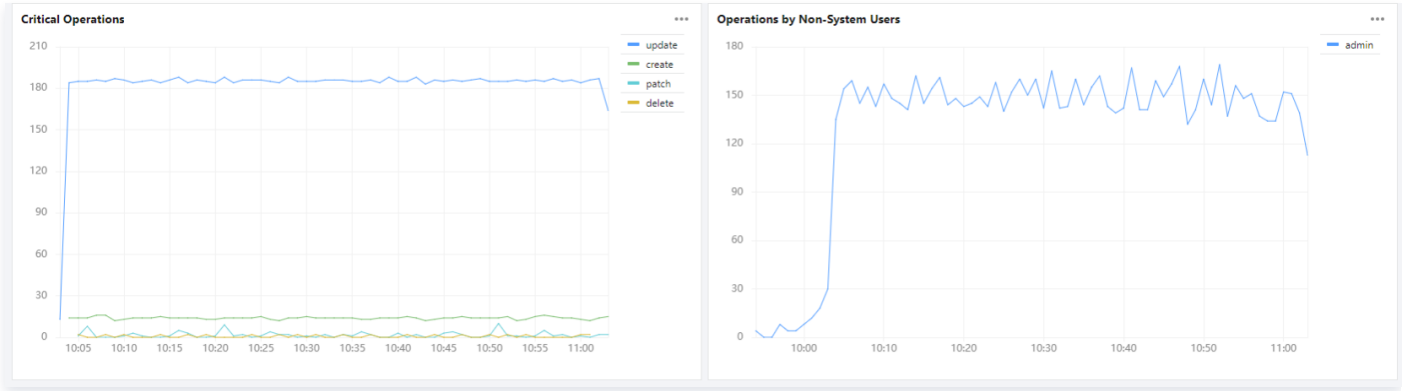
- Total number of events, level distribution, exception event reason and object distribution are shown in the figure below:



- The summary of various common events is shown in the figure below:



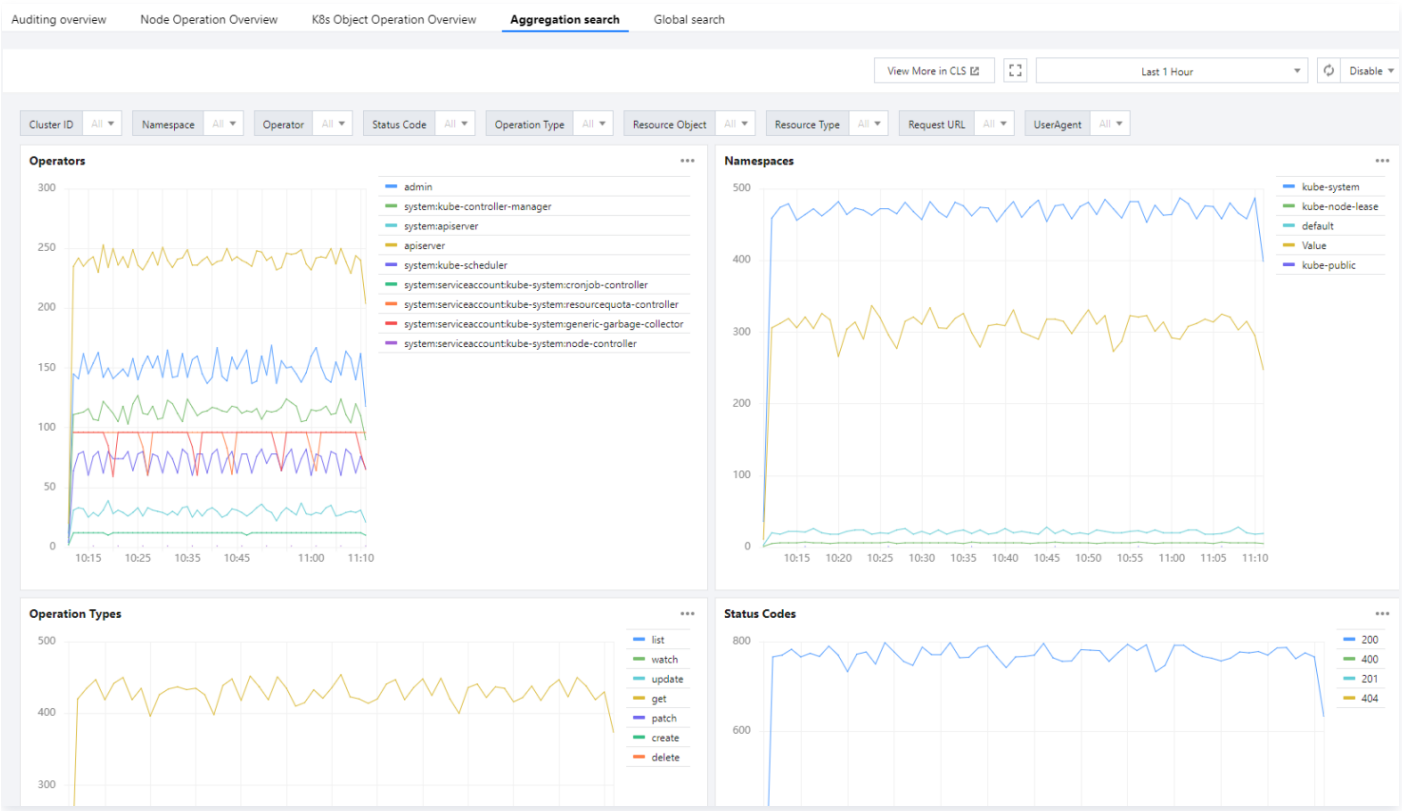
- Event trends and top exception events list:



Exception events aggregation search

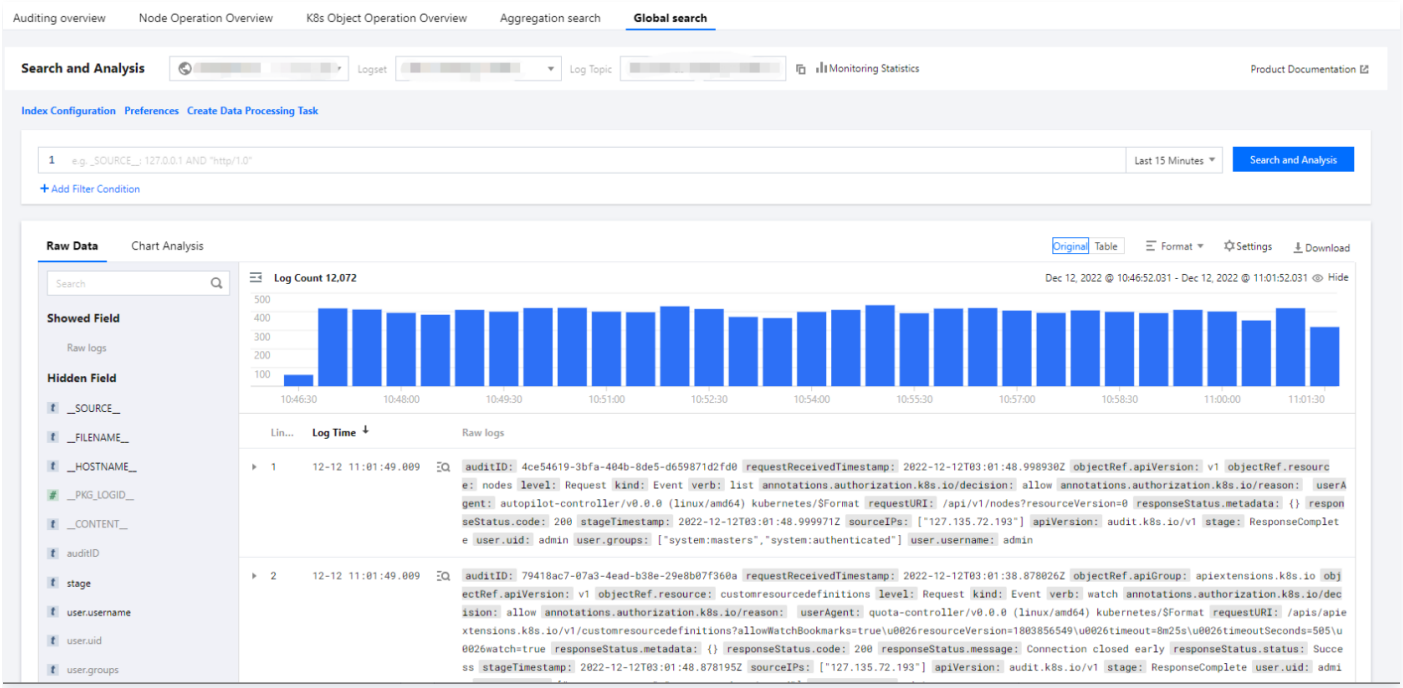
On the **Exception Events Aggregation Search** page, you can set filter conditions to view the reason and object distribution trends of various exception events in a certain period of time. You can also search the exception events in the list below the trend diagrams

to quickly locate the problems as shown below:



Global search

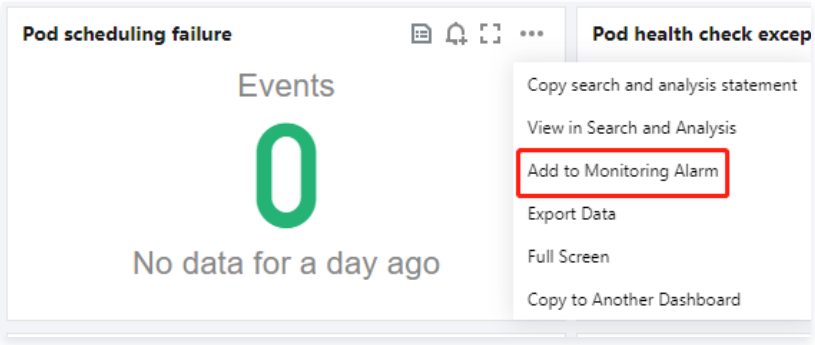
Global search dashboard, with built-in CLS search analysis page, is convenient for users to quickly search all events in the TKE console as shown below:



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, as shown in the following image:



2. Create an alarm policy in Alarm Policy in the **CLS console** as instructed in Configuring Alarming Policies.

Health check

Last updated: 2023-09-26 21:07:11

Scenario

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.

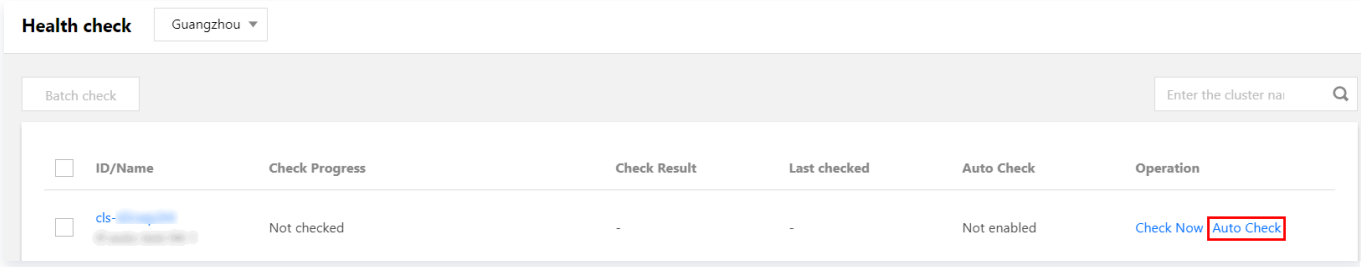
Main Check Items

Check Category	Check Items	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
	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	The following parameters are checked based on the Master node configuration: max-requests-inflight: The maximum number of non-change requests running within a specified period. max-mutating-requests-inflight: The maximum number of mutating requests running within a specified time period.	Yes
	Parameter configuration	The following parameters are checked based on the Master node configuration: kube-api-qps: The QPS used when requesting kube-apiserver.	Yes

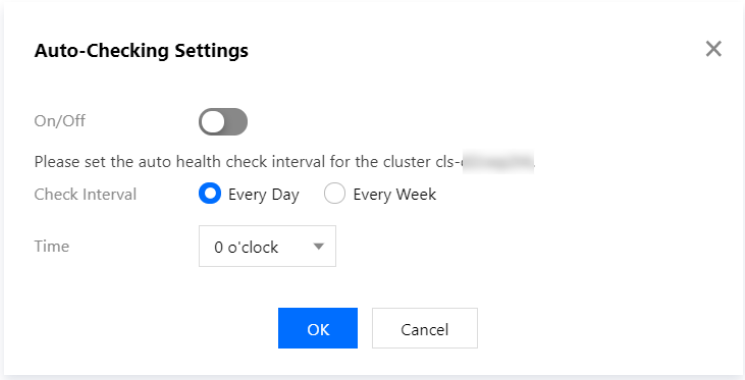
of kube-scheduler	kube-api-burst: maximum burst value during communication with kube-apiserver	
Parameter configuration of kube-controller-manager	The following parameters are checked based on the Master node configuration: kube-api-qps: The QPS used when requesting kube-apiserver. kube-api-burst: maximum burst value during communication with kube-apiserver	Yes
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; verify if the cluster nodes support multi-AZ disaster recovery, i.e., when one availability zone becomes unavailable, whether the total resources of the remaining availability zones are sufficient to support the current cluster's workload.	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

Instructions

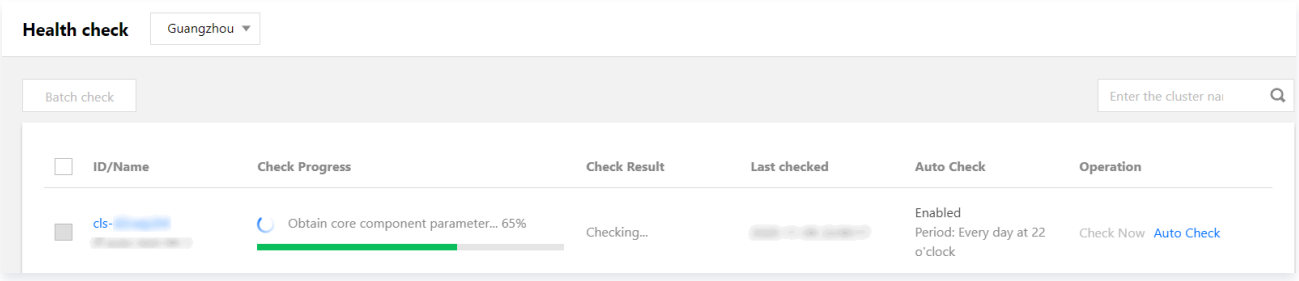
1. Log in to the TKE console and select **Operation Center** > **Health Check** in the left sidebar.
2. Navigate to the "Health Check" page, select the cluster that requires a health check, and choose an appropriate check method. There are three health check methods: Batch Check, Immediate Check, and Automatic Check.
   - **Batch Check**: simultaneously checks multiple clusters.
   - **Check Now**: checks only one cluster.
   - **Auto-check**: Suitable for clusters that require periodic checks. Select the cluster that needs periodic checks and click **Auto-check**, as shown below:



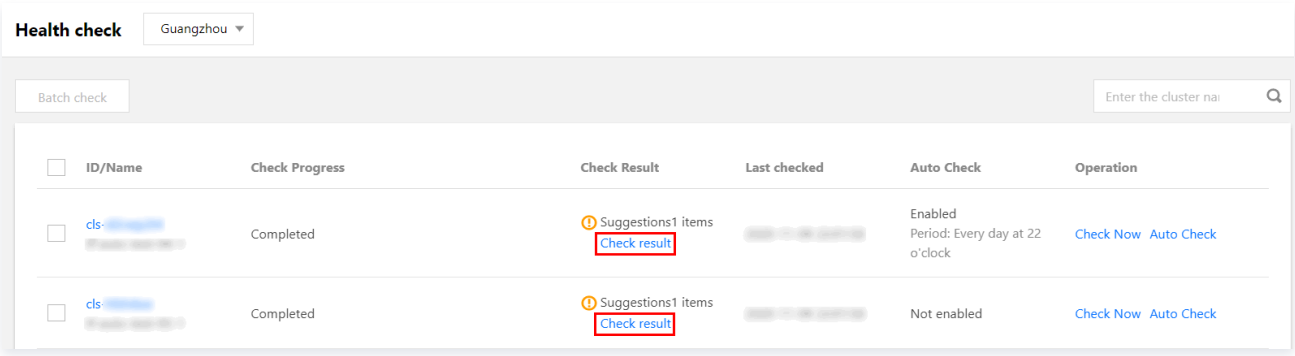
In the "Auto Check Settings" pop-up window, you can configure the enablement status, check cycle, and time according to your needs, as shown in the figure below:



3. After selecting the check method, wait for the check to complete and view the progress. As shown below:

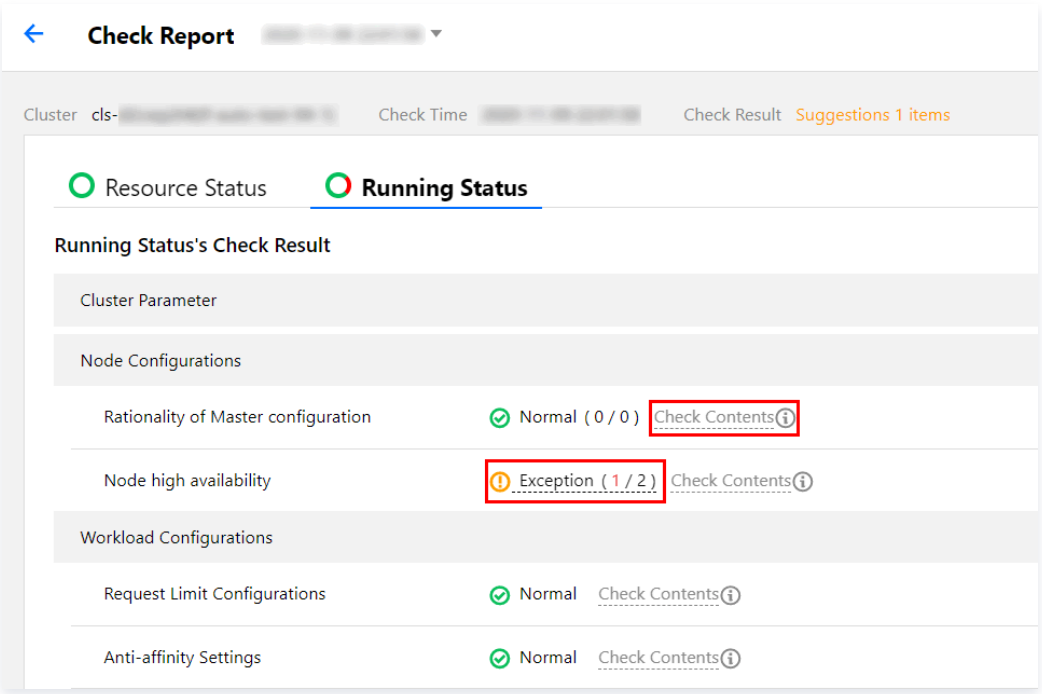


4. After the check is completed, click **View Results** to view the check report, as shown below:



On the check report page, select **Resource Status** and **Running Status** to view the resource status and exceptions, respectively. Click **Check Items** to display the specific check items, and click **Exceptions** to view the exception level, description, cause,

impact, and rectification suggestions, as shown below:



Monitoring and Alarming
Overview of Monitoring and Alarms

Last updated: 2023-09-26 16:03:46

Overview

Tencent Cloud TKE offers monitoring data collection and display capabilities at five levels: cluster, node, workload, Pod, and Container. A robust monitoring environment ensures high reliability, availability, and performance for Tencent Cloud TKE. By configuring alarms, you can collect monitoring data from various dimensions for different resources, allowing you to easily understand resource usage and swiftly pinpoint errors.
Collecting monitoring data helps you establish a baseline for the normal performance of your container clusters. By measuring the performance of container clusters under different load conditions and at various times, and collecting historical monitoring data, you can gain a clear understanding of the normal performance of your clusters and services. This enables you to quickly determine if the services are operating abnormally based on current monitoring data and promptly identify solutions to any issues. For example, you can monitor CPU utilization, memory usage, and disk I/O for your services.

Monitoring

For guidance on using the monitoring features of the container service, please refer to View Monitoring Data.
For the currently covered monitoring metrics, please refer to Monitoring and Alarm Metrics List.

Alarm

To promptly detect anomalies in your container service and ensure the stability and reliability of your business, it is recommended that you configure necessary alarms for all production clusters. For guidance on configuring alarms, please refer to Setting Alarms.
For the currently covered alarm metrics, please refer to Monitoring and Alarm Metrics List.

Note

The monitoring and alarm features provided by the container service primarily cover core metrics or events of Kubernetes objects. Please use them in conjunction with the basic resource monitoring provided by the Tencent Cloud Observability Platform (such as cloud servers, block storage, load balancing, etc.) to ensure more refined metric coverage.
If the basic monitoring capabilities provided by Tencent Cloud Container Service do not meet your requirements, you can use the Prometheus Monitoring service offered by Tencent Cloud. Prometheus Monitoring is dedicated to providing lightweight, stable, and highly available services. It retains the native features of Prometheus, supports custom metric collection, multi-cluster monitoring, reporting of millions of metrics, excellent visualization capabilities based on Grafana with default panels, stable multi-channel alarming capabilities, and a non-intrusive architecture that minimally impacts your cluster resources. The highly customizable configuration options help you build the most suitable monitoring platform for cloud-native scenarios. For specific operations, please refer to Tencent Cloud Prometheus One-Click Association with Container Service Monitoring.

List of Monitoring and Alarm Metrics

Last updated: 2023-09-26 16:04:15

Currently, TKE provides monitoring and alarming metrics in the following dimensions, with all metrics being the **average value** within the statistical period.

Cluster Monitoring and Alarm Metrics

Metrics	<p>	Note
Number of Pods	Connections	Number of Pods in the cluster
Number of Nodes	Connections	Number of Nodes in the Cluster
Total CPU Configuration	Core	Total CPU Configuration of the Cluster
CPU Usage	Core	Cluster CPU Usage
CPU utilization	%	Cluster CPU Utilization
CPU Usage (Elastic Container)	Core	Elastic container CPU usage (if using virtual nodes in a node pool)
Block Device Read Size	Mbytes	Total Disk Usage of the Cluster
Block device read count	Threads	Total Disk Reads in the Cluster
Block Device Write Size	Mbytes	Cluster Disk Write Data Volume
Block Device Write Count	Threads	Total Disk Writes in the Cluster
Total MEM	Gbytes	Total Memory of the Cluster
MEM Usage	Mbytes	Total Cluster Memory Usage
Memory Utilization	%	Cluster Memory Utilization
Memory usage (EKS)	Mbytes	Elastic container memory usage (if using virtual nodes in a node pool)
Memory Usage (Elastic Container, excluding Cache)	Mbytes	Elastic container memory usage (excluding cache) if using node pool virtual nodes
Inbound Network Traffic	Mbytes	Cluster Network Traffic In
Network bandwidth	Mbps	Cluster Network Bandwidth
Network Packets In	count/s	Cluster Network Packets In
Outbound Network Traffic	Mbytes	Cluster Outbound Network Traffic
Network Packets Out	count/s	Cluster Network Packets Out
GPU Memory Total	Gbytes	Total GPU Memory of the Cluster
GPU Memory Usage	Mbytes	Cluster GPU Memory Total Usage
Total GPU Quantity	cards	Total Cluster GPU
GPU Usage	cards	CPU utilization rate of entire cluster
vRAM utilization	%	GPU vRAM Utilization
GPU Utilization	%	Cluster GPU Utilization

Master&Etcd and General Node Monitoring and Alarm Metrics

Metrics	<p>	Note
Re-startup of Pods	Threads	Sum of the number of restarts of all pods on the node
Node Status	-	Node status: normal or exceptional
CPU utilization	%	CPU usage of all pods on the node to the total CPU of the node
CPU Allocation	Core	Total CPU allocation for all Pods within the node
Memory Utilization	%	Ratio of the working set memory usage of all Pods in a node to the total node capacity.
Memory Allocation	Mbps	Total memory allocation for all Pods within the node
Private network bandwidth in	Mbps	Total private network inbound bandwidth of all pods on the node
Private network bandwidth out	Mbps	Total for private network outbound bandwidth of all pods on the node
Public network bandwidth in	Mbps	Total public network inbound bandwidth of all pods on the node
Public network bandwidth out	Mbps	Total public network outbound bandwidth of all pods on the node
TCP Connection Count	Connections	Number of TCP connections maintained by the node
GPU Usage	cards	Total GPU usage of all Pods within the node
GPU Memory Usage	Mbps	Total GPU memory usage of all Pods within the node
GPU Memory Utilization	%	Ratio of the total GPU memory usage of all Pods in the node to the node's overall GPU memory capacity
GPU Utilization	%	Ratio of GPU usage by all Pods within the node to the total GPU capacity of the node
Node's ENI-IP Allocation	Connections	Number of IPs allocated on the ENI of a Node
Direct-ENI allocation for Nodes	Connections	Number of IPs assigned to the direct-eni on the Node
Number of IPs allocated to Node Pod CIDR in GlobalRouter mode cluster	Connections	In a K8S cluster with GlobalRouter mode, the number of IPs allocated in a node's Pod CIDR.
Number of IP addresses that can be assigned to nodes in a GlobalRouter mode cluster	Connections	In a K8s cluster with GlobalRouter mode, the total number of IPs that can be allocated to a node.

For more detailed monitoring and alarming of cluster nodes, please refer to Cloud Server Monitoring and Create Alarm Policy.
For more detailed monitoring and alarming of cluster node data disks, please refer to Cloud Disk Monitoring and Create Alarm Policy.

Workload Monitoring and Alarm Metrics

Metrics	<p>	Note
Workload Anomalies	–	Whether the workload is in an abnormal state, non-zero indicates abnormality.
Number of Pods	Connections	Sum of all Pod counts within the workload
Re-startup of Pods	Threads	Total for the number of restarts of all pods in the workload
CPU Usage	Core	CPU usage of all pods in the workload
CPU utilization	%	Ratio of the total CPU usage of all Pods within the workload to the overall usage
MEM Usage	Mbytes	Sum of memory usage for all Pods within the workload
Memory Usage (excluding Cache)	Mbytes	The sum of memory usage (excluding Cache) for all Pods within the workload.
Memory Usage (working_set)	Mbytes	Sum of working set memory usage for all Pods within the workload
Memory Utilization	%	Ratio of memory usage for all Pods within the workload to the total amount
Memory Utilization (excluding Cache)	%	The proportion of memory usage (excluding Cache) for all Pods within the workload to the total memory of all Pods.
Memory Utilization (working_set)	%	Ratio of working set memory usage to total for all Pods within the workload
Inbound Network Bandwidth	bps	Total inbound bandwidth of all pods in the workload
Outbound Network Bandwidth	bps	Total for outbound bandwidth of all pods in the workload
Inbound Network Traffic	B	Total inbound traffic of all pods in the workload
Outbound Network Traffic	B	Total outbound traffic of all pods in the workload
Network Packets In	count/s	Total inbound packets of all pods in the workload
Network Packets Out	count/s	Total outbound packets of all pods in the workload
Block Device Read Size	Mbytes	Sum of block device read sizes for all Pods within the workload
Block device read count	Threads	Sum of block device read operations for all Pods within the workload
Block Device Write Size	Mbytes	Sum of block device write sizes for all Pods within the workload
Block Device Write Count	Threads	Sum of block device write counts for all Pods within the workload
GPU Usage	cards	Sum of GPU usage for all Pods within the workload
GPU Memory Usage	Mbps	Sum of GPU memory usage for all Pods within the workload
GPU Memory Utilization	%	Ratio of GPU memory usage to total GPU memory for all Pods within the workload
GPU Utilization	%	Ratio of the total GPU usage of all Pods within the workload and the total GPU capacity.

If the workload provides services outside the cluster, please see Obtaining Monitoring Data for more information on network monitoring metrics for bound services.

Pod Monitoring and Alarm Metrics

Metrics	<p>	Note
Re-startup of Pods	Threads	Number of pod restarts
Exception	-	Pod status: normal or exceptional
CPU Usage	Core	Number of CPU cores used by the Pod
CPU Utilization(per node)	%	Percentage of total CPU of the node used by the Pod
CPU utilization(% of request)	%	Percentage of the total number of CPU cores specified by Request that is used by the Pod
CPU Utilization(per Limit)	%	Percentage of the total number of CPU cores specified by Limit that is used by the Pod
MEM Usage	Mbytes	Sum of memory usage (including cache) of Containers within a Pod (source: container_memory_usage_bytes)
Memory Usage(excluding cache)	Mbytes	Sum of memory usage by Containers within a Pod, excluding cache (Source: container_memory_usage_bytes - container_memory_cache)
Memory Usage (working_set)	Mbytes	Memory usage of the working set in the Pod's Container (Source: container_memory_working_set_bytes)
Memory Utilization(per node)	%	Percentage of total memory of the node used by the Container within the Pod (including cache)
Memory Utilization(per node, excluding cache)	%	Ratio of memory usage (excluding cache) by Containers within a Pod to the total memory of the node
Memory Utilization (per node, working_set)	%	Ratio of the working set memory usage of the Container in the Pod to the total memory of the node
Memory Utilization (per Request)	%	Ratio of the memory usage of the Container in the Pod and the value of Request specified
MEM Utilization (% Request, excl. cache)	%	Ratio of memory usage (excluding cache) by the Container in the Pod and the value of Request specified
Memory Utilization (per Request, working_set)	%	Ratio of the working set memory usage of the Container in the Pod to the configured Request value
Memory Utilization (per Limit)	%	Ratio of memory usage by the Container in the Pod to the specified Limit value
Memory Utilization(per Limit, excluding cache)	%	Ratio of memory usage (excluding cache) by the Container in the Pod to the set Limit value
Memory Utilization (by Limit, working_set)	%	Ratio of the working set memory usage of the Container in the Pod to the specified Limit value
Inbound Network Bandwidth	Mbps	Sum of Pod’s ingress bandwidth
Outbound Network Bandwidth	Mbps	Sum of Pod’s egress bandwidth
Inbound Network Traffic	Mbytes	Sum of Pod’s ingress traffic
Outbound Network Traffic	Mbytes	Sum of Pod’s egress traffic
Network Packets In	count/s	Sum of Pod’s inbound packets

Network Packets Out	count/s	Sum of Pod’s outbound packets
Pod TCP Connections	Connections	Number of TCP connections for a Pod
Block Device Read Size	Mbytes	Pod's block device read size
Block device read count	Threads	Pod's block device read count
Block Device Write Size	Mbytes	Pod's block device write size
Block Device Write Count	Threads	Pod's block device write count
Rootfs Usage	Byte	Rootfs usage in the Pod
GPU Applications	cards	Pod GPU Memory Applications
GPU Memory Utilization (per node)	%	Ratio of GPU memory usage in the Pod to the total GPU memory of the node
GPU Memory Utilization (per Request)	%	Ratio of GPU memory usage in the Pod to the GPU memory request value
GPU Utilization (per node)	%	Pod GPU usage as a percentage of the total node GPU capacity
GPU Utilization(per Request)	%	Ratio of GPU usage in the Pod to the GPU request amount
GPU Memory Applications	Mbytes	Pod GPU Memory Request
GPU Memory Usage	Mbytes	GPU Memory Usage in Pods
GPU Usage	cards	Pod GPU Usage
GPU vRAM Utilization	%	Percentage of GPU vRAM usage in the Pod relative to the total vRAM capacity
GPU Encoding Resource Utilization Rate	%	Pod GPU Encoding Resource Usage Rate
GPU Decoding Resource Usage Rate	%	Pod GPU decoding resource usage rate
GPU Stream Processor Utilization	%	Pod GPU Stream Processor Utilization

Container Monitoring and Alarm Metrics

Metrics	<p>	Note
CPU Usage	Core	CPU usage of container
CPU Utilization(per node)	%	CPU usage of the container to the total CPU of the node
CPU utilization(% of request)	%	CPU usage of the container to the Request value
CPU Utilization(per Limit)	%	CPU usage of the container to the Limit value
MEM Usage	Mbytes	Memory usage of the Container, including cache (Source: container_memory_usage_bytes)

Memory Usage(excluding cache)	Mbytes	Memory usage of the container, excluding cache (Source: container_memory_usage_bytes − container_memory_cache)
Memory Usage (working_set)	Mbytes	Container working set memory usage (Source: container_memory_working_set_bytes)
Memory Utilization(per node)	%	Percentage of total node memory used by the container (including cache)
Memory Utilization(per node, excluding cache)	%	Percentage of total node memory used by the container (excluding cache)
Memory Utilization (per node, working_set)	%	Percentage of the working set memory usage of the Container relative to the total node capacity
Memory Utilization (per Request)	%	Memory usage of the container to the Request value
MEM Utilization (% Request, excl. cache)	%	Ratio of the memory usage (excluding cache) of the Container and the value of Request set
Memory Utilization (per Request, working_set)	%	Ratio of the working set memory usage of the Container to the value of Request set
Memory Utilization (per Limit)	%	Memory usage of the container to the Limit value
Memory Utilization(per Limit, excluding cache)	%	Ratio of the container's memory usage (excluding cache) to the set Limit value
Memory Utilization (by Limit, working_set)	%	Ratio of the working set memory usage of the Container to the specified Limit value
Block device read bandwidth	B/s	Throughput of the container to read data from disk
Block device write bandwidth	B/s	Throughput of the container to write data to disk
Read IOPS of Block Device	Reads/second	Number of times the container read from disk
Write IOPS of Block Device	Reads/second	Number of times the container wrote to disk

GPU Monitoring Metrics Acquisition

Last updated: 2023-09-26 16:04:28

Component Description

TKE has developed the elastic-gpu-exporter component for obtaining GPU-related monitoring metrics, which mainly include:

- GPU Card Utilization
- Pod / Container GPU Resource Utilization

Deployment directions

The elastic-gpu-exporter is deployed into the cluster using a DaemonSet.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: elastic-gpu-exporter
  namespace: kube-system
  labels:
    app: elastic-gpu-exporter
spec:
  updateStrategy:
    type: RollingUpdate
  selector:
    matchLabels:
      name: gpu-manager-ds
      app: nano-gpu-exporter
  template:
    metadata:
      name: elastic-gpu-exporter
      labels:
        name: gpu-manager-ds
        app: nano-gpu-exporter
    spec:
      nodeSelector:
        qgpu-device-enable: enable
      serviceAccount: elastic-gpu-exporter
      hostNetwork: true
      hostPID: true
      hostIPC: true
      containers:
        - image: ccr.ccs.tencentyun.com/tkeimages/elastic-gpu-exporter:v1.0.8
          imagePullPolicy: Always
          args:
            - --node=$(NODE_NAME)
          env:
            - name: "PORT"
              value: "5678"
            - name: "NODE_NAME"
              valueFrom:
                fieldRef:
                  fieldPath: spec.nodeName
          name: elastic-gpu-exporter
          securityContext:
            capabilities:
              add: ["SYS_ADMIN"]
          volumeMounts:
            - name: cgroup
              readOnly: true
              mountPath: "/host/sys"
      volumes:
        - name: cgroup
```

```
        hostPath:
          type: Directory
          path: "/sys"
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: elastic-gpu-exporter
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: elastic-gpu-exporter
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: elastic-gpu-exporter
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

```
  name: elastic-gpu-exporter
subjects:
  - kind: ServiceAccount
    name: elastic-gpu-exporter
    namespace: kube-system
---
apiVersion: v1
kind: Service
metadata:
  name: elastic-gpu-exporter
  namespace: kube-system
  annotations:
    prometheus.io/scrape: "true"
  labels:
    kubernetes.io/cluster-service: "true"
spec:
  clusterIP: None
  ports:
    - name: elastic-gpu-exporter
      port: 5678
      protocol: TCP
      targetPort: 5678
  selector:
    app: nano-gpu-exporter
```

View running status

After deployment, an elastic-gpu-exporter DaemonSet is created within the cluster:

```
NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
elastic-gpu-exporter 1 1 1 1 1 <none> 3m36s
```

A running elastic-gpu-exporter Pod will be present on eligible nodes:

```
NAME READY STATUS RESTARTS AGE
elastic-gpu-exporter-dblqm 1/1 Running 0 6s
```

Getting Monitoring Metrics

The elastic-gpu-exporter service running on the node will output to the `/metrics` path, so you can also obtain monitoring metrics using the following command:

```
$ curl NodeIP:5678/metrics
```

GPU Card-related Metrics

gpu_xxx	GPU Metrics
gpu_core_usage	Actual GPU computing power utilized
gpu_mem_usage	GPU vRAM actually used
gpu_core_utilization_percentage	GPU Computing Power Utilization
gpu_mem_utilization_percentage	GPU vRAM Utilization

The metrics for GPU cards are presented as follows:

```
gpu_core_usage{card="0",node="10.0.66.4"} 0
```

Note:
"card" represents the GPU's index, and "node" represents the corresponding node.

Pod-related Metrics

pod_xxx	Pod Metrics
pod_core_usage	Computing power actually used by the Pod
pod_mem_usage	Actual GPU memory used by the pod
pod_core_utilization_percentage	Percentage of actual computing power used by the Pod relative to the requested computing power
pod_mem_utilization_percentage	Percentage of actual video memory used by the pod compared to the requested video memory
pod_core_occupy_node_percentage	Percentage of computing power actually used by the pod relative to the total computing power of the node
pod_mem_utilization_percentage	Percentage of total node memory used by the Pod's actual GPU memory usage
pod_core_request	Computing power requested by the Pod
pod_mem_request	GPU memory requested by the Pod

The metrics for Pod are in the following format:

```
pod_core_usage{namespace="default",node="10.0.66.4",pod="7a2fa737-eef1-4801-8937-493d7efb16b7"} 0
```

Note:
"namespace" represents the namespace where the Pod is located, "node" represents the node where the Pod is located, and "pod" represents the name of the Pod.

Container-related Metrics

container_xxx	Container Metrics
container_gpu_utilization	Actual computing power used by the container
container_gpu_memory_total	Actual GPU memory used by the container
container_core_utilization_percentage	Percentage of the actual computing power used by the container compared to the requested computing power
container_mem_utilization_percentage	The percentage of the actual video memory used by the container compared to the requested video memory
container_request_gpu_memory	GPU Memory Requested by Containers
container_request_gpu_utilization	Computing power requested by the container

The metrics for containers are presented in the following format:

```
container_gpu_utilization{container="cuda",namespace="default",node="10.0.66.4",pod="cuda"} 0
```

Note:

"container" refers to the container name, "namespace" refers to the namespace where the container is located, "node" refers to the node where the container is located, and "pod" refers to the name of the Pod where the container is located.

Log Management

Collect container logs to CLS

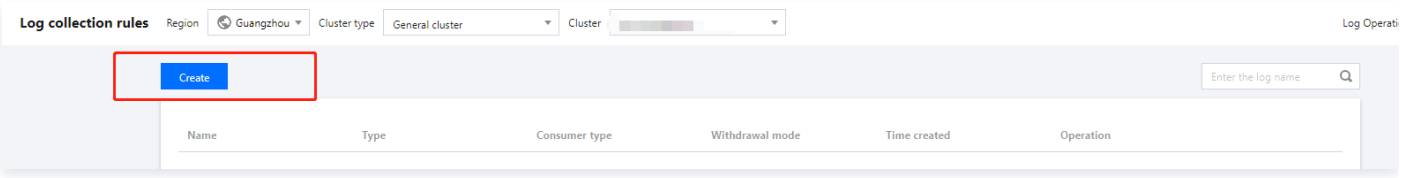
Last updated: 2023-09-26 21:10:34

This document introduces how to configure log collection rules in the TKE console and ship logs to Tencent Cloud Log Service (CLS).

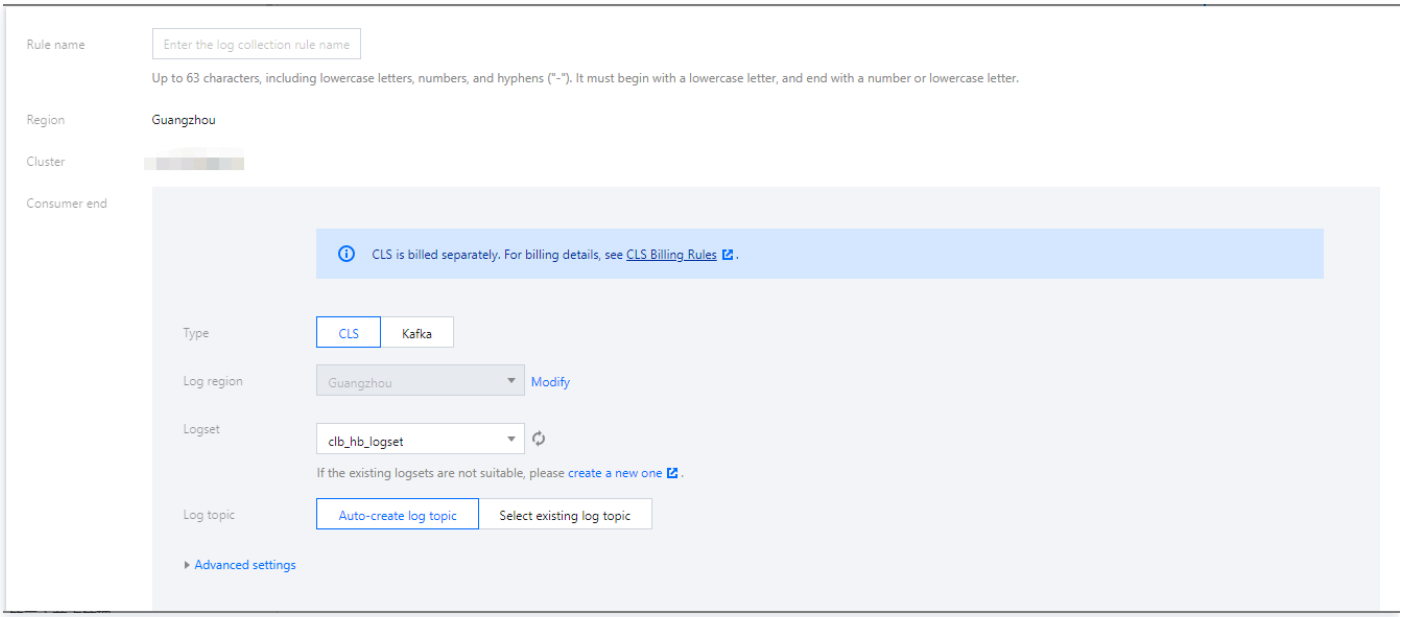
Instructions

Creating a log collection rule

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 cluster for which you want to configure a log collection rule, then click **Create**. As shown in the image below:



3. On the "Create Log Collection Rule" page, configure the log service consumer by selecting **CLS** in the **Consumer** > **Type** section. As shown in the image below:



- 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 search by default.
  - Custom metadata: You can customize metadata and log indexes.

> **Note**
> - CLS does not support cross-region log shipping (from 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

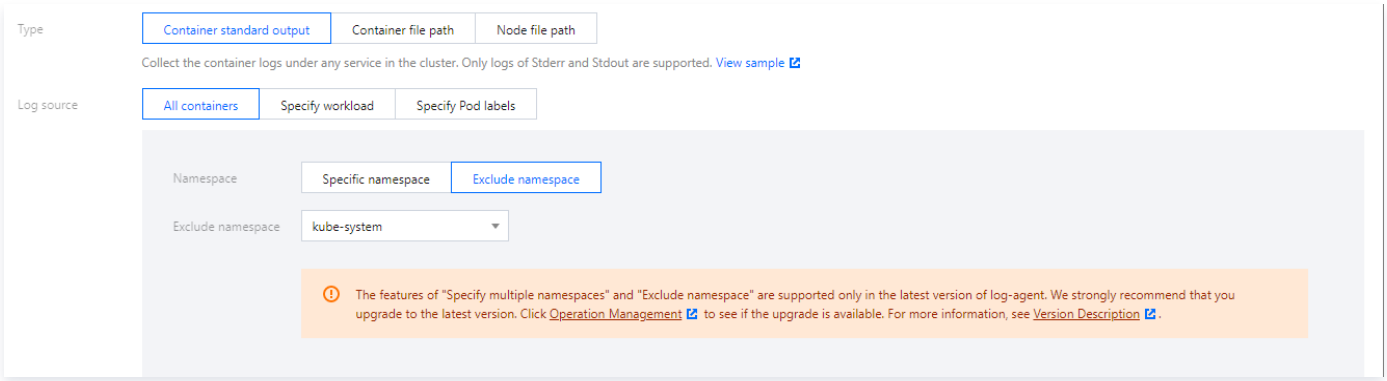
container logs collected from a Shenzhen cluster can be shipped only to Guangzhou, and the containers logs collected from a Tianjin cluster can be shipped only to Beijing. You can find more information in the console.

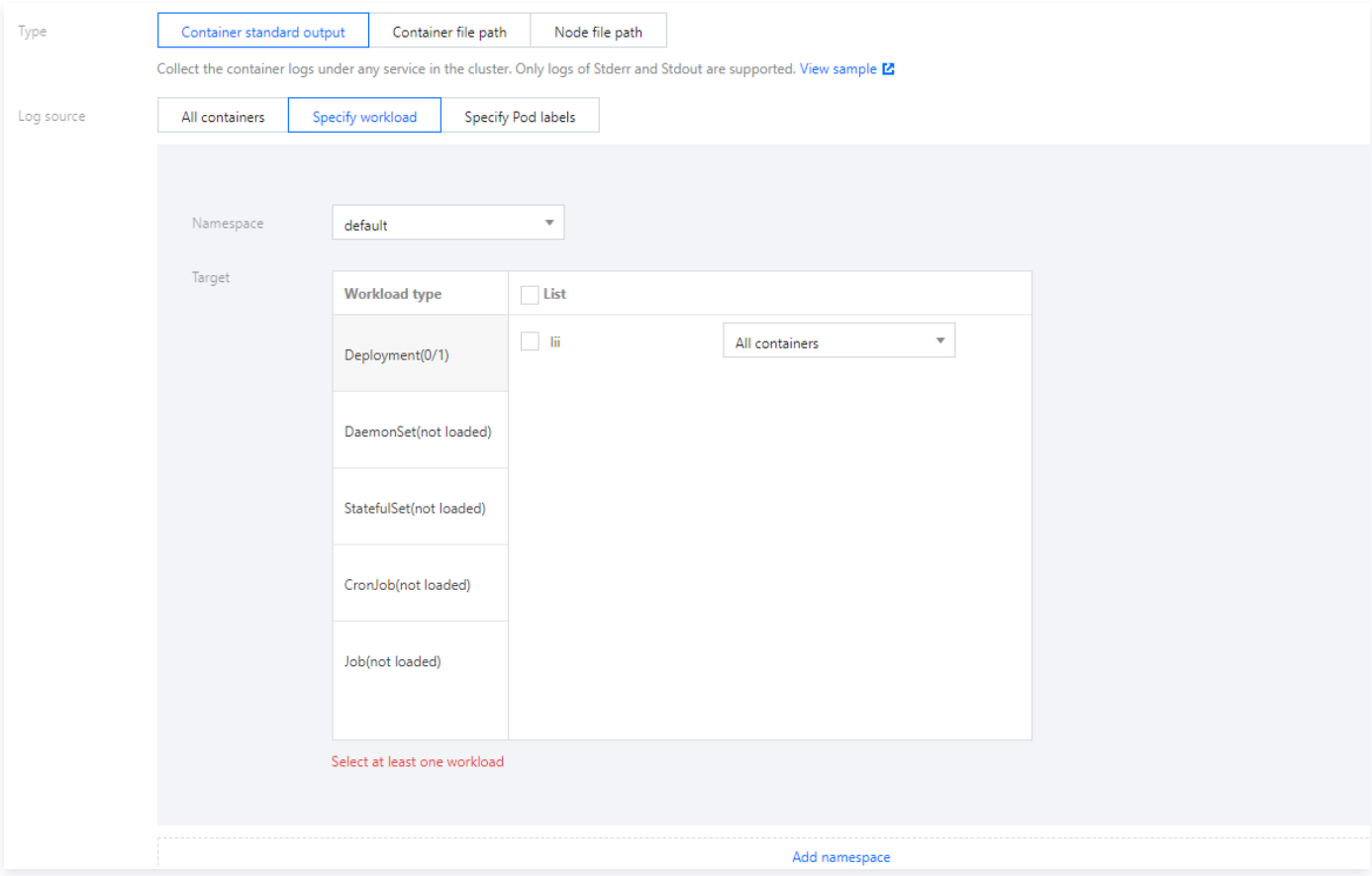
- Currently, a log topic only supports the collection configuration of one type of logs. That is, the log, audit, and event types cannot use the same topic. If they use the same topic, logs will be overwritten. Please ensure that the selected log topic is not occupied by other collection configurations. 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

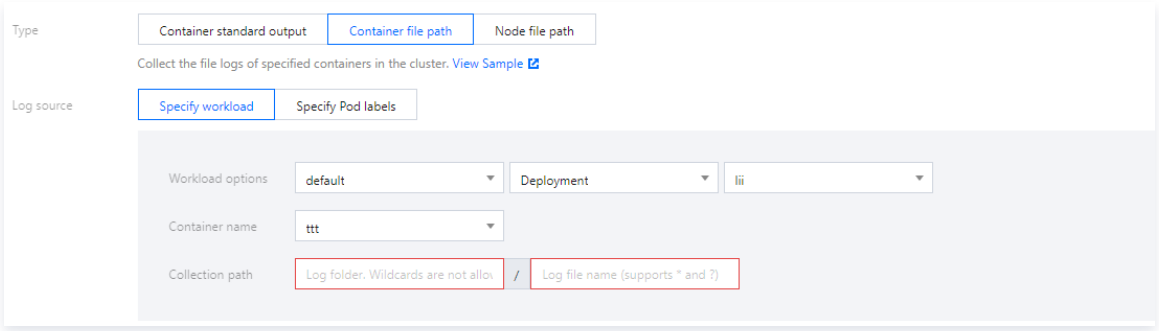
Log sources support three types: **All Containers**, **Specific Workloads**, and **Specific Pod Labels**. As shown in the image below:

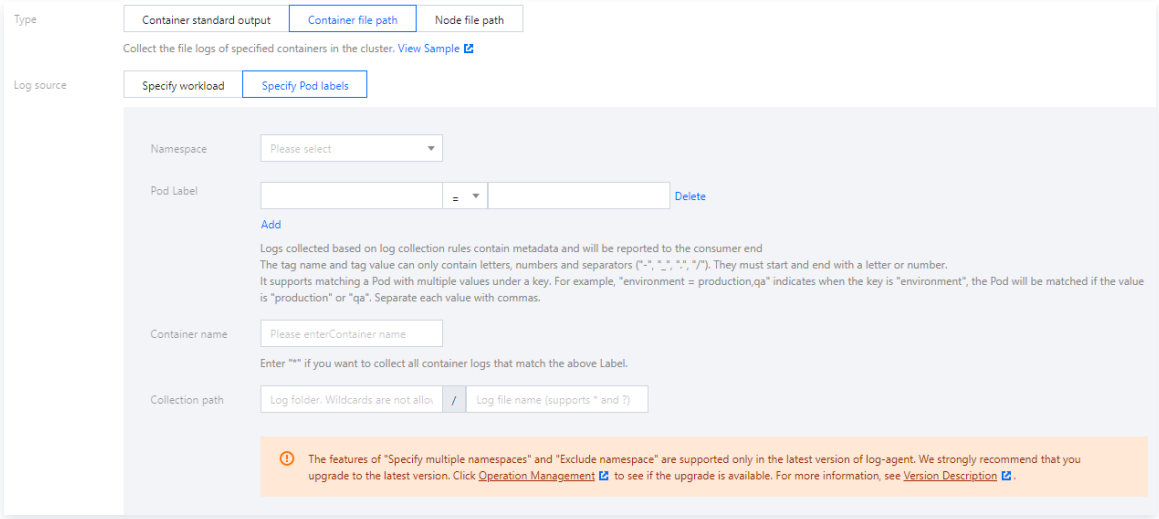




Container File Logs

- Log sources support two types: **Specified Workloads** and **Specified 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`. As shown in the image below:

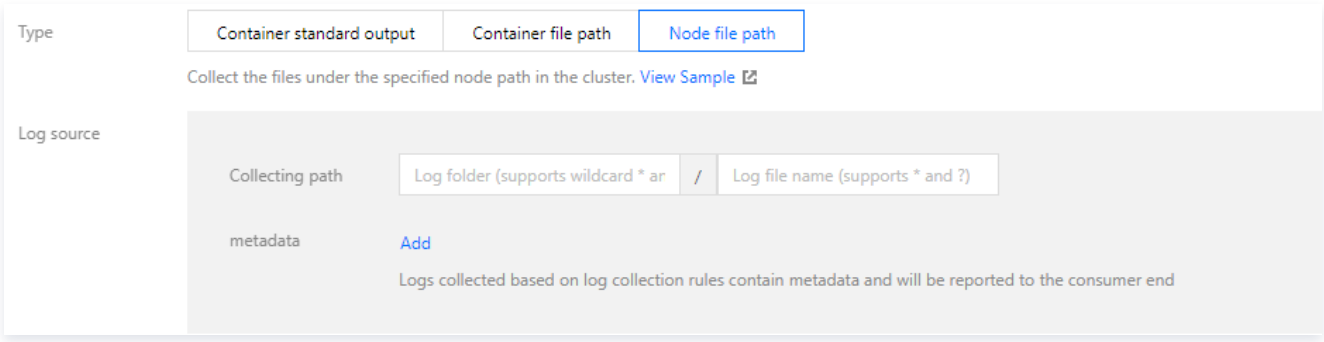




> **Note**
> The "Container File Path" **must not be a soft or hard link**, as this would cause the actual path of the soft link to be nonexistent within the collector's container, resulting in log collection failure.

Node File Logs

- The collection path can be specified using a file path or wildcard rules. For example, if you need to collect all file paths in the format `/opt/logs/service1/*.log` and `/opt/logs/service2/*.log`, you can specify the collection folder as `/opt/logs/service*` and the file name as `*.log`.
- You can customize and add key-value pairs of metadata according to your needs, and the metadata will be added to the log records.



Note:

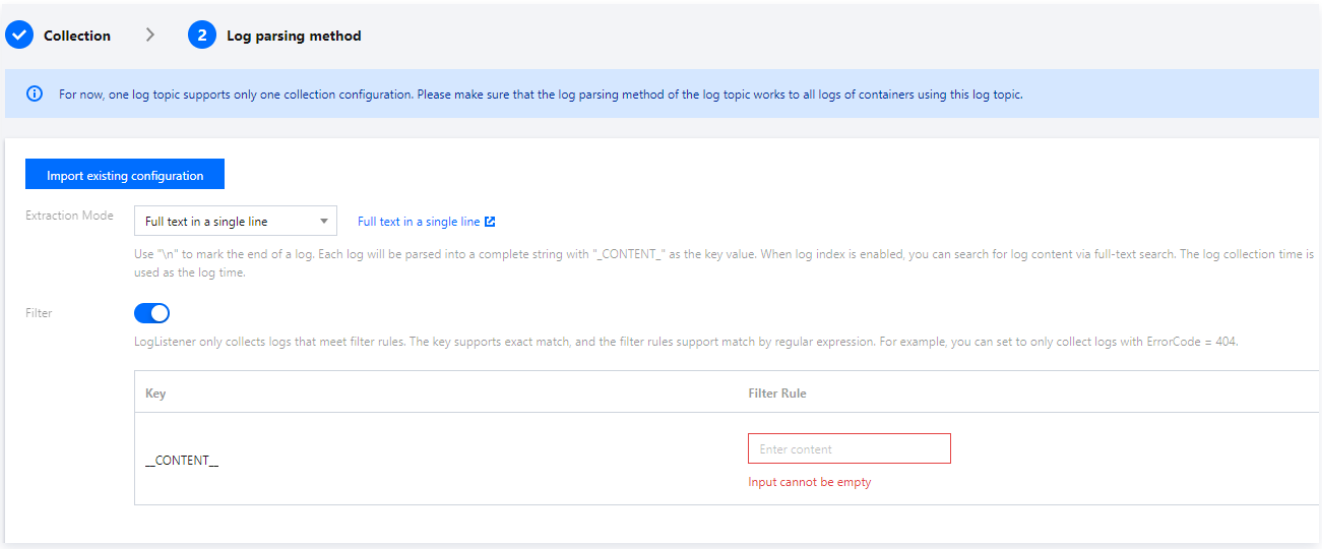
- The "Node File Path" **must not be a soft or hard link,** otherwise, the actual path of the soft link will not exist in the collector, resulting in log collection failure.
- Each node log file can be collected to only one log topic.

Note

For both "container standard output" and "container files" (excluding "node file paths" or hostPath mounts), in addition to the raw log content, container or Kubernetes-related metadata (e.g., the container ID that generated the logs) will also be reported to CLS. This allows users to trace the log source or search based on container identifiers or characteristics (e.g., container name, labels) when viewing logs.
Please refer to the table below for container or Kubernetes-related metadata:

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}	The labels of the pod to which the log belongs (for example, a pod with two labels: app=nginx, env=prod, will have two metadata attached in the uploaded logs: pod_label_app:nginx, 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 in the image below:



- Encoding Mode: Supports **UTF-8** and **GBK**.
- Extraction Mode: Supports multiple extraction modes, as described below:

Parsing mode	Note	Documentation
Full text in a single line	A log contains only one line of content, ending with a newline character (\n). Each log is parsed into a complete string with the key value of CONTENT. After enabling indexing, you can search for log content through full-text retrieval. The log time is based on the collection time.	Single Line Full Text Format
Multi-line Full Text	A complete log entry spans multiple lines, using the first line regex method for matching. When a log line matches the predefined regular expression, it is considered the beginning of a log entry, and the appearance of the next line's start serves as the end identifier for that entry. A default key value CONTENT is also set, with the log time based on the collection time. Automatic generation of regular expressions is supported.	Multi-line Full Text Format
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 piece of log data 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 key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	Full Regular Expression (Multi-Line)
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	In a separator log, the entire log data can be structured according to the specified separator, and each complete log ends with a line break `\n`. When CLS processes separator logs, you need to define a unique key for each separate field. Invalid fields, which are fields that need not be collected, can be left blank. However, you cannot leave all fields blank.	Separator Format

Combined parsing	When your log structure is too complex and involves multiple parsing patterns, and a single parsing pattern (such as Nginx, full regex, JSON, etc.) cannot meet the log parsing requirements, you can use Loglistener's combined parsing format to analyze logs. This mode allows users to input code (in JSON format) in the console to define the log parsing pipeline logic. You can add one or more Loglistener plugin processing configurations, and Loglistener will execute them in the order of the configurations.	

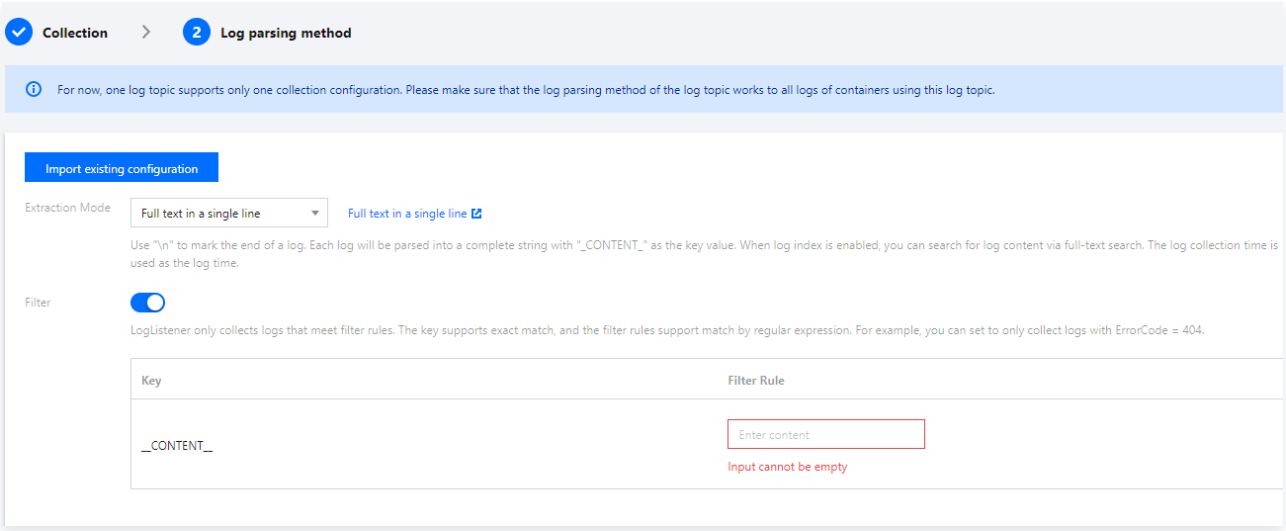
- Filter: LogListener collects only logs that match the filter rules. "Key" supports full matching, and the filtering rule supports regex matching, such as collecting logs with `ErrorCode = 404` only. You can enable the filter and configure rules according to your needs.

> **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 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 cluster for which you want to update the log collection rule, then click **Edit Collection Rule** on the right. As shown in the image below:



3. Update the configuration as needed and click **Done**.

> **Note**
> The logset and log topic cannot be modified later.

Documentation

Besides using the TKE console, you can also configure log collection by using the Custom Resource Definitions (CRD). For more information, see Using CRD to Configure Log Collection.

Using CRD to Configure Log Collection

Last updated: 2023-09-26 16:05:32

Scenario

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.

Preparations

You have enabled Log Collection in the Operation Management section of the TKE console. For more information, see Enabling Log Collection.

CRD Overview

Structure overview

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig                 ## Default value
metadata:
  name: test      ## CRD resource name, unique within the cluster
spec:
  clsDetail:                ## Configuration for shipping to CLS
    ...
  inputDetail:                ## Data source configuration for collection
    ...
  kafkaDetail:                ## The configuration for shipping to a CKafka or self-built Kafka cluster
    ...
status:                     ## CRD resource status
  status: ""
  code: ""                  ## The error code returned by the called API
  reason: ""                ## Error cause
```

clsDetail description

> **Note**
> You cannot modify a specified topic.

```
  clsDetail:
    To automatically create a log topic, both the logset and topic names must be specified simultaneously.
    logsetName: test                ## CLS logset name. Logset for the name will be created automatically if there is not any. If
there is the logset, log topic will be created under it.
    topicName: test                 ## CLS log topic name. Log topic for the name will be created automatically if there is not any.

    Select an existing logset and log topic. If the logset is specified but the log topic is not, a log topic will be created
automatically.
    logsetId: xxxxxx-xx-xx-xx-xxxxxxxx  ## The ID of the CLS logset. The logset needs to be created in advance in CLS.
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx   ## CLS log topic ID. The log topic needs to be created in CLS in advance and should
not be occupied by other collection configurations.

    logType: json_log  ## Log collection format. json_log represents JSON format, delimiter_log represents separator-based
format, minimalist_log represents single-line full text format, multiline_log represents multi-line full text format, and
fullregex_log represents full regex format. The default is minimalist_log.
  logFormat: xxx                ## Log formatting method
  period: 30                    ## Lifecycle in days. Value range: 1–3600. 3640 indicates permanent storage.
  partitionCount:               ## The number (an integer) of log topic partitions. Default value: 1. Maximum value: 10.
  tags:                     ## Tag description list. This parameter is used to bind a tag to a log topic. Up to nine tag key-value
pairs are supported, and a resource can be bound to only one tag key.
```

```
  - key: xxx                    ## Tag key
    value: xxx                  ## Tag value
  autoSplit: false## Whether to enable automatic split (Boolean type). Default value: true.
  maxSplitPartitions:
  storageType: hot              ## Log topic storage class. Valid values: hot (STANDARD); cold (STANDARD_IA). Default value:
hot.
  excludePaths:                 ## Collection path blocklist
    - type: File                ##  Type, choose either File or Path (optional)
      value: /xx/xx/xx/xx.log       ## The value of type
  indexs:                       ## You can customize the indexing method and field when creating a topic.
    - indexName:   ## When configuring a key value or metafield index for a field, there is no need to add the __TAG__. prefix
to the metafield Key. It should be consistent with the field Key when uploading logs. The Tencent Cloud console will
automatically add the __TAG__. prefix when displaying.
      indexType:  ## Field type. Valid values: long, text, double
      tokenizer:  ## Field delimiter. Each character represents a delimiter. Only English symbols and \n\t\r are supported. For
long and double fields, leave it empty. For text fields, we recommend you use @&?|#()='",;:<>[]{}/ \n\t\r\ as the delimiter.
      sqlFlag:    ## Whether the analysis feature is enabled for the field (Boolean)
      containZH: ## Whether Chinese characters are contained (Boolean)
  region: ap-xxx                ## Topic region for cross-region shipping
    userDefineRule: xxxxxx          ## Custom collection rule, which is a serialized JSON string
    extractRule: {}             ## Extraction and filter rule. If ExtractRule is set, LogType must be set.
```

inputDetail description

```
  inputDetail:
    type: container_stdout   ## Log collection type, including container_stdout (container standard output), container_file
(container file), and host_file (host file)

    containerStdout:       ## Container standard output
      namespace: default    ## The Kubernetes namespace of the container to be collected. Supports multiple namespaces,
separated by commas, e.g., default,namespace. If not specified, it represents all namespaces. Note: Cannot be specified
simultaneously with excludeNamespace.
      excludeNamespace: nm1,nm2   ## The Kubernetes namespace of the container to be excluded. Separate multiple
namespaces by comma, for example, nm1,nm2. If this field is not specified, it indicates all namespaces. Note that this field
cannot be specified if namespace is specified.
    nsLabelSelector: environment in (production),tier in (frontend) ## Filter namespaces based on namespace labels
      allContainers: false      ## Whether to collect the standard output of all containers in the specified namespace. Note that
if allContainers=true, you cannot specify workload, includeLabels, and excludeLabels at the same time.
      container: xxx            ## Name of the container of which the logs will be collected. If the name is empty, it indicates the
log names of all matching containers will be collected. Note: In conjunction with
      excludeLabels:  ## Pods with the specified labels will be excluded. This field cannot be specified if workload, namespace,
and excludeNamespace are specified.
        key2: value2  ## Supports matching Pods with multiple values under the same key. For example, specifying environment
= production,qa means that when the key is environment, Pods with values production or qa will be excluded. Please use a
comma to separate multiple values. If includeLabels is also specified, Pods that match the intersection of includeLabels will
be considered.

      includeLabels:  ## Pods with the specified labels will be collected. This field cannot be specified if workload, namespace,
and excludeNamespace are specified.
        key: value1   ## Logs collected based on log collection rules contain metadata and will be reported to the consumer end.
Supports matching pods with multiple values under the same key. For example, specifying environment = production,qa
means that when the key is environment and the value is either production or qa, both will be matched. Please use a comma
to separate multiple values. If excludeLabels is also specified, pods that match the intersection of both will be considered.

      metadataLabels:       ## Specify the Pod labels to be collected as metadata. If not specified, all Pod labels will be collected
as metadata.
      - label1
      customLabels:         ## Custom metadata
        label: l1
```

```
    workloads:
    container: xxx      ## Name of the container to collect. If this parameter is not specified, it indicates all containers in the workload Pod will be collected.
      kind: deployment  ## Workload type. Supported values include deployment, daemonset, statefulset, job, and cronjob.
      name: sample-app    ## Workload name
      namespace: prod   ## Workload namespace

  containerFile:  ## File in the container
    namespace: default     ## The Kubernetes namespace of the container to be collected. A namespace must be specified.
    excludeNamespace: nm1,nm2   ## The Kubernetes namespace of the container to be excluded. Separate multiple namespaces by comma, for example, nm1,nm2. If this field is not specified, it indicates all namespaces. Note that this field cannot be specified if namespace is specified.
    nsLabelSelector: environment in (production),tier in (frontend) ## Filter namespaces based on namespace labels
    container: xxx         ## The name of the container of which the logs will be collected. The * indicates the log names of all matching containers will be collected.
    logPath: /var/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.
    customLabels:          ## Custom metadata
      key: value
    excludeLabels:  ## Pods with the specified labels will be excluded. This field cannot be specified if workload is specified.
      key2: value2  ## Supports matching Pods with multiple values under the same key. For example, specifying environment = production,qa means that when the key is environment, Pods with values production or qa will be excluded. Please use a comma to separate multiple values. If includeLabels is also specified, Pods that match the intersection of includeLabels will be considered.

    includeLabels:  ## Pods with the specified labels will be collected. This field cannot be specified if workload is specified.
      key: value1   ## Logs collected based on log collection rules contain metadata and will be reported to the consumer end. Supports matching pods with multiple values under the same key. For example, specifying environment = production,qa means that when the key is environment and the value is either production or qa, both will be matched. Please use a comma to separate multiple values. If excludeLabels is also specified, pods that match the intersection of both will be considered.
    metadataLabels:        ## Specify the Pod labels to be collected as metadata. If not specified, all Pod labels will be collected as metadata.
    - label1               ## pod label
    workload:
      container: xxx      ## Name of the container to collect. If this parameter is not specified, it indicates all containers in the workload Pod will be collected.
      name: sample-app    ## Workload name

  hostFile:               ## Host file path
    filePattern: '.log'   ## Log file name, supports wildcards and ?, where * matches multiple arbitrary characters and ? matches a single arbitrary character.
    logPath: /tmp/logs    ## Log file folder. Wildcards are not supported.
    customLabels:         ## Custom metadata
      label1: v1
```

extractRule description

Name	Local Disk Types	Required	Description
timeKey	String	Not required	Time field key name. `time_key` and `time_format` must appear in pairs.
timeFormat	String	Not req	Time field format. For more information, see the output parameters of the time format description of the `strftime` function in C language.

		uired	
delimiter	String	Not required	Delimiter for delimited log, which is valid only if `log_type` is `delimiter_log`.
logRegex	String	Not required	Full log matching rule, which is valid only if `log_type` is `fullregex_log`.
beginningRegex	String	Not required	First-Line matching rule, which is valid only if `log_type` is `multiline_log` or `fullregex_log`.
unMatchUpload	String	Not required	Whether to upload the logs failed to be parsed. Valid values: `true` (yes); `false` (no).
unMatchedKey	String	Not required	Unmatched log key.
backtracking	String	Not required	The size of the data to be rewound in incremental collection mode. Valid values: `-1` (full collection); `0` (incremental collection). Default value: `-1`.
keys	Array of String	Not required	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	Not required	Log keys to be filtered, which correspond to `FilterRegex` by subscript.
filterRegex	Array of String	Not required	The `regex` of the log keys to be filtered, which corresponds to `FilterKeys` by subscript.
isGBK	String	Not required	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	Not req	Whether it's standard JSON. Values: `0` (No), `1` (Yes). **Note:** This field may return null, indicating that no valid value was found.

		uir ed	

kafkaDetail description

```
kafkaDetail:
  brokers: x.x.x.x:p    ## (Required) The broker address. Generally, it is domain name:port. If there are more than one address, separate them with ",".
  topic: test           ##
  kafkaType: CKafka     ## Kafka type. Valid values: CKafka (CKafka); SelfBuildKafka (self-built Kafka).
  instanceId: xxxx      ## When kafkaType = CKafka, set the CKafka instance ID
  logType: minimalist_log ## The type of the parsed Kafka log. Valid values: minimalist_log or "" (full text in a single line); multiline_log (full text in multiple lines); json (JSON).
  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, the same as the EKS Kafka collector); filebeat (Filebeat); fluent-bit (Fluent Bit).
  messageKey:           ## Supports specifying a key to ship logs to a designated partition. By default, it is disabled, and logs are randomly distributed by date. When enabled, logs with the same key will be shipped to the same partition. You can choose a Pod field as the key. For example, to use the Pod name, select Field>metadata.name.
    value: Field        ## Required, topicID
    valueFrom:
     fieldRef:
       fieldPath: metadata.name ## If the key is Field, you can select metadata.name, metadata.namespace, spec.nodeName, and spec.serviceAccountName.
```

status description

status	Note
Empty	Initial status
Synced	Configured the collection successfully
Stale	Failed to configure the collection

Sample CRD

Sample CRD for the configuration of the container standard output

All containers

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
```

Specifying a workload

```
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
```

Specifying Pod labels

```
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

```
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
```

Specifying Pod labels

```
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
```

Backup Center Overview

Last updated: 2023-09-26 16:05:56

Tencent Kubernetes Engine (TKE) Backup Center provides an integrated solution for backup, recovery, and migration of containerized applications. This article mainly introduces the use cases and core components of the Backup Center.

> **Note:**
> The Backup Center feature is currently in beta testing. You can submit a ticket to request a trial.

Use Cases

- **Backup and Recovery**: In case all resources in a cluster or namespace are accidentally deleted, you can quickly restore the business using backup data.
- **Cross-cloud Migration**: Flexibly deploy businesses and migrate application data between different public clouds and hybrid clouds.
- **Business Compliance**: Collaborate with the security department to regularly retrieve backup data for business auditing.

Core Components

Component Name	Description
tke-backup	The backup component, deployed in the user's cluster, is based on the open-source tool Velero and supports scheduled backup and restoration of Kubernetes cluster resources through CRD.

Kubernetes objects deployed within a cluster

Kubernetes Object Name	Local Disk Types	Resource Amount	Namespaces
tke-backup	Deployment	At least 0.1 cores CPU and 256 MB memory are required.	tke-backup
tke-backup	Service	-	tke-backup
tke-backup	backupstoragelocation	-	-
tke-backup	backup	-	-
tke-backup	restores	-	-

ResourceType

The TKE-customized backup-related CRD resources are described as follows:

Resource Name	Description
Backup	Specify the backup policy for designated resource objects. Creating a Backup resource initiates the backup process, while deleting a Backup resource does not delete the underlying data stored in the COS backup repository.
BackupSchedule	Specify the backup policy for specific resource objects at a particular point in time, responsible for generating Backup resource objects on a scheduled basis.
Restore	Restore the backup information to the target TKE cluster. Creating a Restore resource initiates the recovery process. Deleting the Restore resource will not have any other impact, it will only remove the recovery operation record from the recovery list.

Instructions

1. Log in to the TKE console and create a backup repository. For more information, see Create a Backup Repository.
2. Create a backup or scheduled backup policy for the target cluster. For more information, see Backup Management.
3. Restore specific resource objects in the cluster based on backup data. For more information, see Recovery Management.

Remote Terminals

Remote Terminal Overview

Last updated: 2023-09-26 16:06:34

Remote Terminal Overview

Remote terminals assist in swiftly debugging containers, connecting to containers for problem inspection, and supporting copy-paste, file upload, and download functions. This resolves issues related to lengthy container login paths and challenging debugging processes.

Help

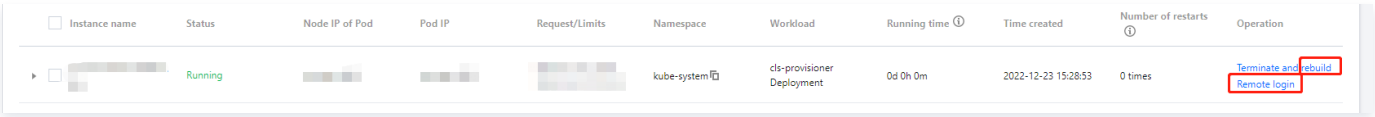
- Basic Operations of Remote Terminal
- Alternative Container Login Methods

Basic Remote Terminal Operations

Last updated: 2023-09-26 16:06:46

Remote terminal connects to the container

1. Log in to the Tencent Cloud Container Service Console and select Cluster from the left navigation bar.
2. On the **Cluster Management** page, click the cluster ID (cls-xxx) to go to the cluster details page.
3. Choose **Node Management > Nodes** in the left sidebar. On the **Node List** page, click the node ID to enter the Pod management page.
4. In the instance list, click **Remote Login** on the right side of the instance, as shown in the figure below.

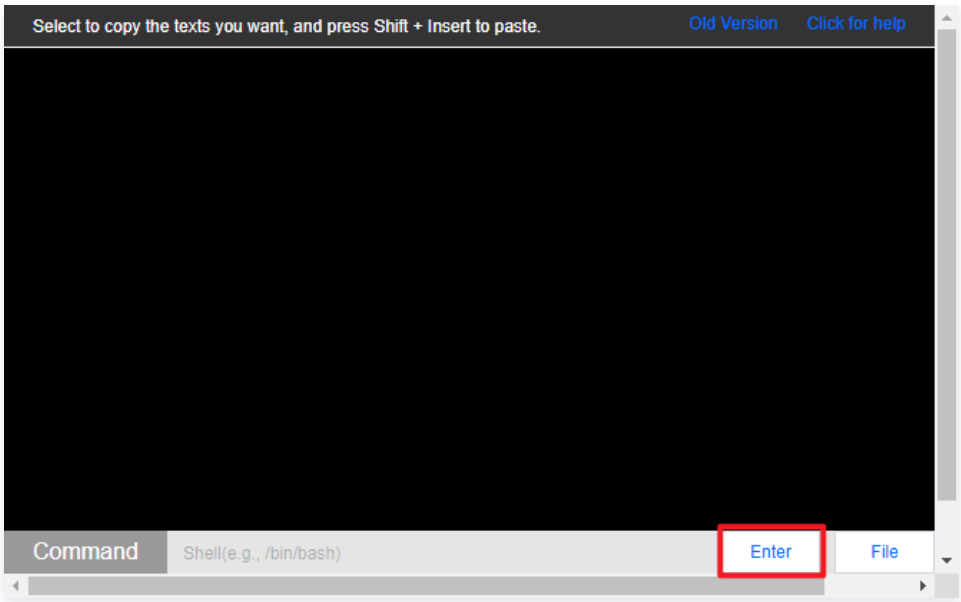


> **Note**
> Containers meeting any of the following conditions do not support remote login:
> - The namespace is kube-system.
> - A bash is not built in the container image.
>
> For FAQs about the remote terminal, see here.

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.

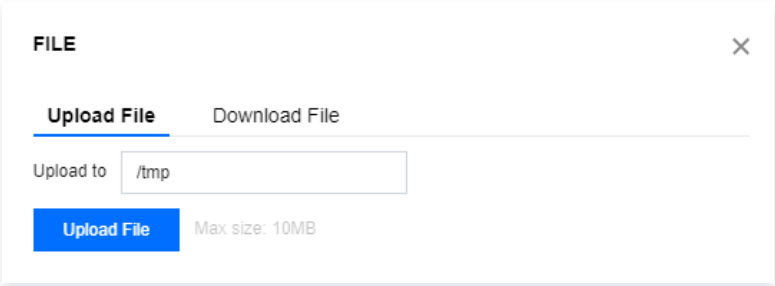
Running commands in a container without an installed shell

1. Enter the remote terminal page.
2. Enter the command you want to execute in the input box below, and click **Enter**, as shown in the figure below:



File upload and download

1. Enter the remote terminal page.
2. Click **File** below and choose to upload or download files, as shown in the figure below:



- **Upload File**: Specify the directory to which the files are to be uploaded.
- **Download File**: Specify the path of the files to be downloaded.

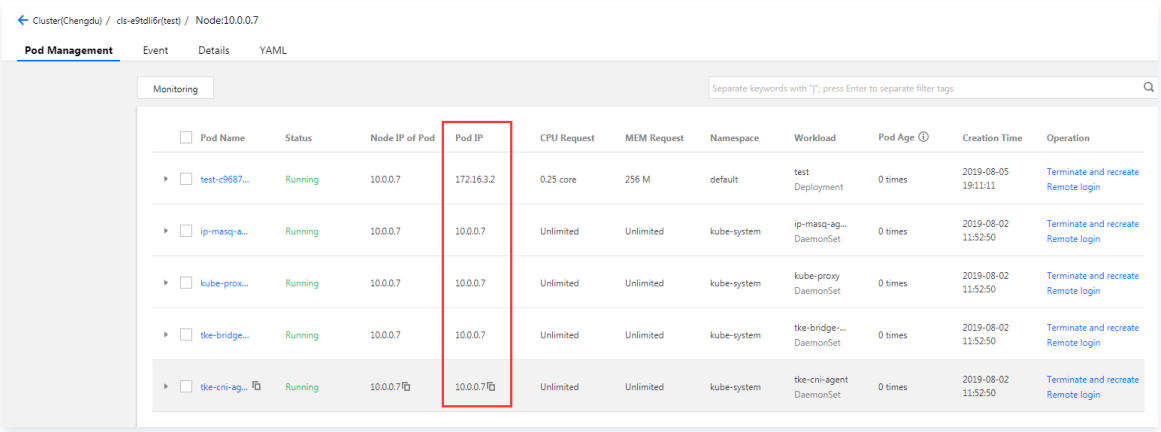
Other Login Methods

Last updated: 2023-09-26 16:07:02

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.

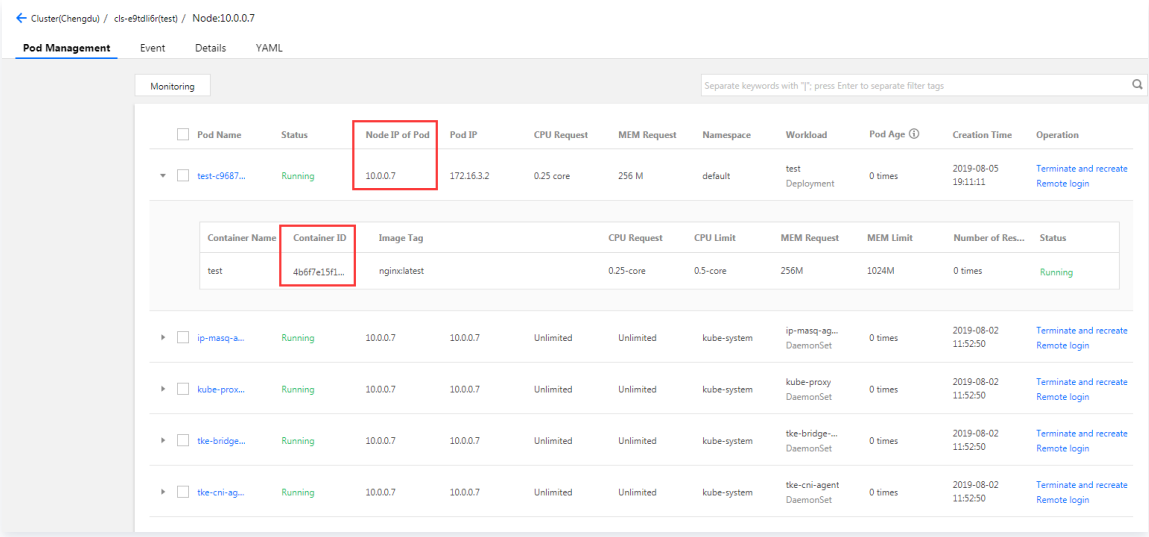
1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the cluster ID (cls-xxx) to go to the cluster details page.
3. On the cluster details page, select **Node Management** > **Node** in the left sidebar.
4. On the **Node List** page, click the node name to go to the Pod management page.
5. In the instance list, obtain the instance IP address, as shown in the following figure:



6. Log in to any node in the cluster. For more information, see **here**.
7. Log in to the container using SSH.

Logging In to a Container Through the Container's Node

1. Log in to the TKE console and click **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 node IP address and container ID where the container is located, as shown in the following image:



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-aa14-11e6-8a18-52540093c40b_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_6b389dd2
[root@VM_0_60_centos ~]# docker exec -it 75b3b15af61a /bin/bash
root@worid-24bn2:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt proc  root  run  sbin  srv  sys  tmp  usr  var
```