Tencent Kubernetes Engine FAQs

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

FAQs
TKE General Cluster
Auto-scaling related

Last updated: 2023-09-26 16:21:27

What are the differences between Cluster Autoscaler (CA) and node scaling based on monitoring metrics in elastic scaling?

Cluster Autoscaler (CA) ensures that all pods in the cluster can be scheduled regardless of the actual load, while node auto scaling based on monitoring metrics do not take pods into consideration during auto scaling. Therefore, nodes without pods might be added, or nodes with system-critical pods such as kube-dns might be deleted during auto scaling. Kubernetes discourages the latter auto scaling mechanism. In conclusion, these two modes conflict and should not be enabled at the same time.

How does CA work with auto scaling groups?

A CA-enabled cluster will, according to the configuration of the selected node, create a launch configuration and bind an auto scaling group to it. The cluster will then perform scale-in/out in this bound auto scaling group. CVM instances scaled out are automatically added to the cluster. Nodes that are automatically scaled in/out are billed on a pay-as-you-go basis. For more information about auto scaling group, see Auto Scaling (AS).

Can a node manually added in the TKE Console be scaled in by CA?

No. CA only scales in the nodes within the auto scaling group. Nodes that are added on the TKE Console are not added to the auto scaling group.

Can I add or remove CVM instances in the Auto Scaling Console?

No. We do not recommend making any modifications on the AS Console.

What configurations of the selected node will be inherited during scaling?

When creating an auto scaling group, you need to select a node in the cluster as a reference to create a launch configuration. The node configuration for reference includes:

- vCPU
- Memory
- System disk size
- Data disk size

- Disk type
- Bandwidth
- Bill-by-bandwidth mode
- Whether to assign public IP
- Security Group
- VPCs
- Subnets

How do I use multiple auto scaling groups?

Based on the level and type of the service, you can create multiple auto scaling groups, set different labels for them, and specify the label for the nodes scaled out in the auto scaling groups to classify the service.

What is the maximum quota for scaling?

Currently, Tencent Cloud users have a quota of 30 pay-as-you-go CVM instances per availability zone. If you wish to have more than 30 pay-as-you-go CVM instances in an auto scaling group, please contact support to apply. For specific quotas, refer to the Instance Count and Quota for your current availability zone. Additionally, there is a maximum limit for elastic scaling, which is set at 200. If you wish to scale beyond this limit, please contact support to apply.

Is scale-in safe for the cluster?

Since pods will be rescheduled when a node is scaled in, scale-in can be performed only if the service can tolerate rescheduling and short-term interruption. We recommend using PDB. PDB can specify the minimum number/percentage of replicas for a pod set that remains available at all times. With PodDisruptionBudget, application deployers can make sure that the cluster operations that actively remove pods will not terminate too many pods at a time, which helps prevent data loss, service interruption, or unacceptable service degradation.

What types of pods in a node will not be scaled in?

- If the Pod for which you set strict PDB does not satisfy the PDB, it will not be scaled down.
- There are Pods under the Kube-system.
- On the node there are Pods that are not created by controllers such as Deployment, ReplicaSet, Job, or StatefulSet.
- There are Pods with local storage.
- Pods that cannot be scheduled to another node.

How long does it take to trigger scale-in after the condition is met?

10 minutes.

How long does it take to trigger scale-in when the node is marked as Not Ready?

20 minutes.

How often is a node scanned for scaling?

Every 10 seconds.

How long does it take to scale out a CVM instance?

It generally takes less than 10 minutes. For more information, see Auto Scaling.

Why is a node with an unschedulable pod not scaled out?

Please check the following:

- Whether the requested resource of the pod is too large.
- Whether a node selector is set.
- Whether the maximum number of nodes in the auto scaling group has been reached.
- Please ensure that your account balance is sufficient (insufficient balance will prevent elastic scaling from expanding) and check for other reasons such as insufficient quota. For more information, see Troubleshooting Elastic Scaling Issues.

How can I prevent CA from scaling in a specific node?

```
You can set the following information in the node's annotations:
kubectl annotate node <nodename> cluster-autoscaler.kubernetes.io/scale-down-disabled=true
```

How can I view the TKE node scaling events?

You can view the scaling activities of the auto scaling group in the Elastic Scaling Console and check the k8s events. Corresponding events will be available on the following three resources:

- kube-system/cluster-autoscaler-status config map

Resources	Event
ScaledUpGroup	CA initiates scaling out.
ScaleDownEmpty	CA removed a node that had no running Pods.
ScaleDown	CA scale-in.

- node

Resources	Event
ScaleDown	CA scale-in.
ScaleDownFailed	Unable to scale in by CA.

- pod

Resources	Event
TriggeredScaleUp	CA scales out due to this Pod.
NotTriggerScaleUp	CA cannot find a suitable auto scaling group to scale out, making the Pod unschedulable.
ScaleDown	CA attempts to drain the Pod in order to scale in the node.

Can the kube-proxy mode of a container cluster be changed?

Kube-proxy proxy mode supports iptables and ipvs, which can be selected during cluster creation. However, once the cluster is created, the mode **cannot be changed**. For more information, see Enabling IPVS for a Cluster.

External DNS Causes Abnormal Node Initialization

Last updated: 2023-09-26 16:23:10

Background Information

In order to resolve to related services in the business when you use a TKE custom image, you may have modified the DNS resolution order in the custom image or completely replace Tencent's DNS service with your self-built DNS.

Operation Impact

The above operation may cause a failure to resolve to Tencent Cloud's official resource library when registering a node into the cluster, resulting in a high probability of a node initialization failure or network/storage component exception.

- Node initialization: The error "Failed to resolve address, dns may be changed" may be reported during node initialization.
- Network components: Network components such as IPAMD depend on Tencent Cloud's private network DNS to work properly. The failure to resolve to the Tencent Cloud resource library may result in the unavailability of network components.
- Storage components: Mount/Unmount of storage components such as CBS-CSI fail.

Note

The installation of related components within the cluster depends on Tencent Cloud's official resource library, with the resolution address as follows:

```
nameserver 183.60.83.19
nameserver 183.60.82.98
```

Solution

Configuring Tencent Cloud DNS nameserver as the upstream of self-built DNS

We recommend you add the nameserver in the `/etc/resolv.conf` to the upstream of your self-built DNS server. Because some services rely on Tencent Cloud DNS resolution, if the nameserver is not set as the upstream of your self-built DNS, some services may not work

properly. This document takes BIND 9 as an example to modify the configuration file and write the upstream DNS address into forwarders as shown below:

```
options {
      forwarders {
            183.60.83.19;
            183.60.82.98;
      };
      ...
```

Within 24 hours after the above operation is completed, the automatic retry policy of the node initialization process will keep trying to perform the node initialization until it is resolved to the Tencent Cloud resource library. After 24 hours, you need to delete the nodes and recreate them. If the above solution does not work, submit a ticket for assistance.

> **Note**
> If the external DNS Server and the request source are not in the same Region, some Tencent domain names that do not support cross-region access may become invalid.

TKE Serverless Cluster
TKE Serverless Clusters-related

Last updated: 2023-09-27 14:21:00

This document describes the causes and solutions of various FAQs of TKE Serverless clusters.

Why is the Pod specification inconsistent with the set Request/Limit?

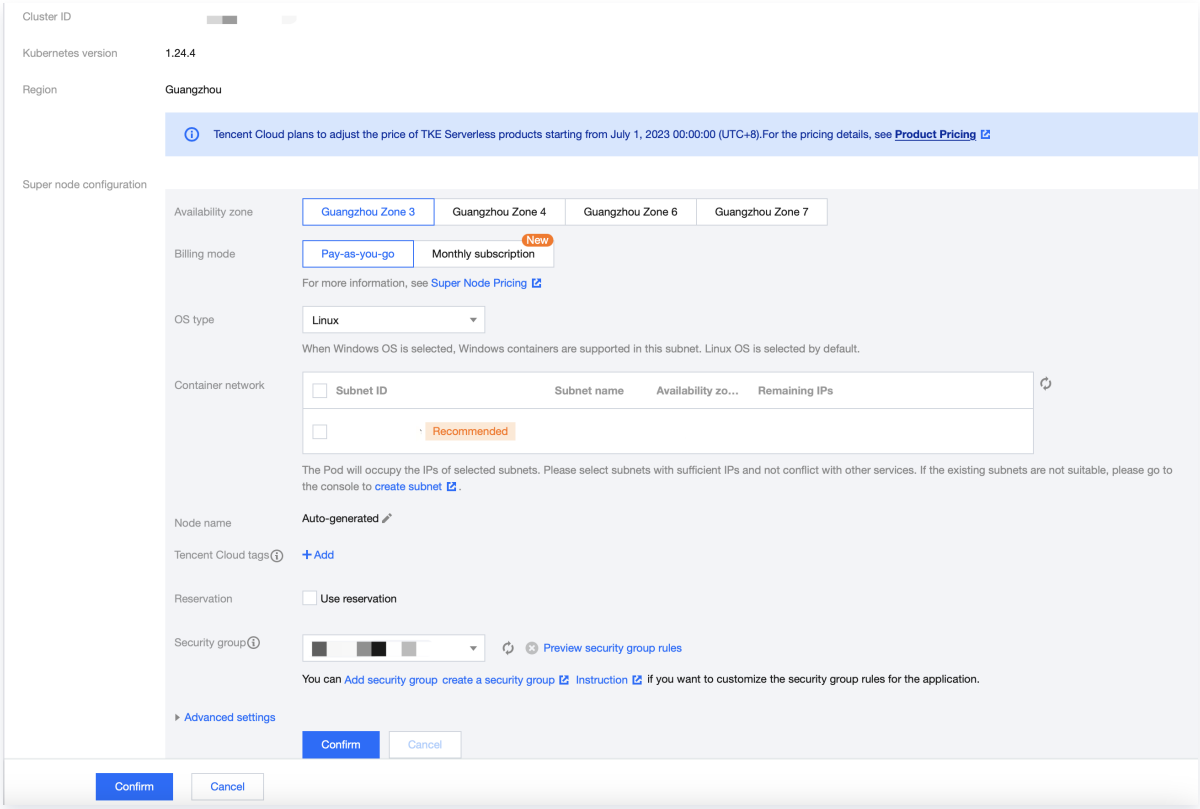
When allocating resources for Pods, TKE Serverless will calculate the Request and Limit values set by the workload, and automatically determine the amount of resources required for running the Pods, instead of allocating resources according to the set Request and Limit values. For more information, see CPU specifications calculation methods for pods and GPU specification calculation methods for pods.

How to create or modify the container network of the TKE Serverless cluster?

When creating a cluster, you need to select a VPC as the cluster network and specify a subnet as the container network. For more information, see Notes on the Container Network. The Pod of the TKE Serverless cluster directly occupies an IP address of the container network subnet. When using the cluster, you can create or modify the container network through creating or removing the super node. The detailed instructions are shown below.

Step 1. Create a super node to add a container network

1. Log in to the TKE console and click Cluster in the left sidebar.
2. Click the ID of the cluster for which you need to modify the container network to go to the cluster details page.
3. Click **Super node** in the left sidebar. On the **Super node** page, click **Create**.
4. On the **Create super node** page, select the container network with sufficient IP addresses and click **OK**.

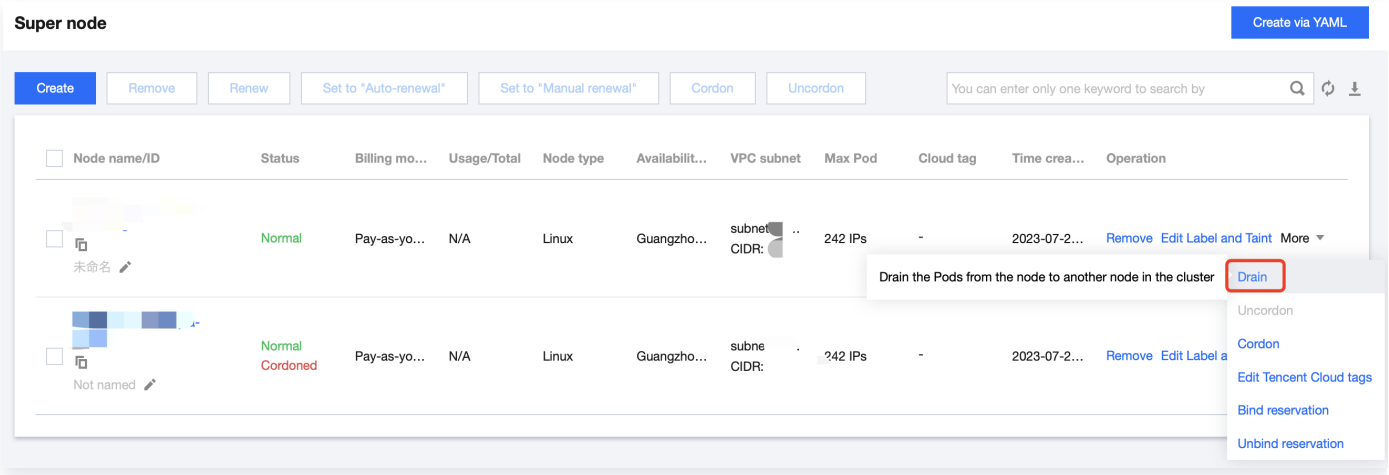


Step 2. Remove the super node to delete the container network

> **Note**
> Make sure that at least one super node remains in the TKE Serverless cluster after the removal. If there is only one super node, you cannot remove it.

Before removing a super node, you need to drain all Pods on it (excluding those managed by DaemonSet) to other super nodes. After the draining is completed, you can remove the super node; otherwise, the removal will fail. The detailed directions are as shown below.

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. Click the ID of the cluster for which you need to modify the container network to go to the cluster details page.
3. Click **Super node** in the left sidebar. On the **Super node** page, choose **More** > **Drain** on the right of the node name.



4. On the **Drain node** page, check the node information and click **OK**. After the node is drained, it will enter the "Blocked" status, and no more Pods can be scheduled to it.

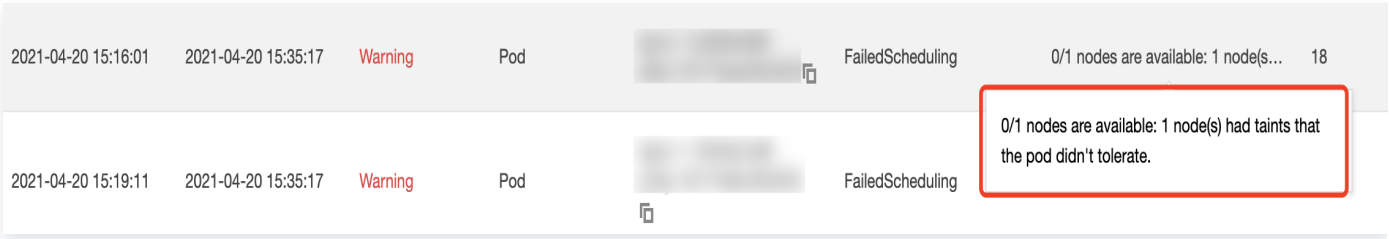
> **Note**
> Note that Pods will be rebuilt once the node is drained.

5. 4. On the **Super node** page, click **Remove** on the right of the node name.
6. 5. On the **Delete node** page, click **OK**.

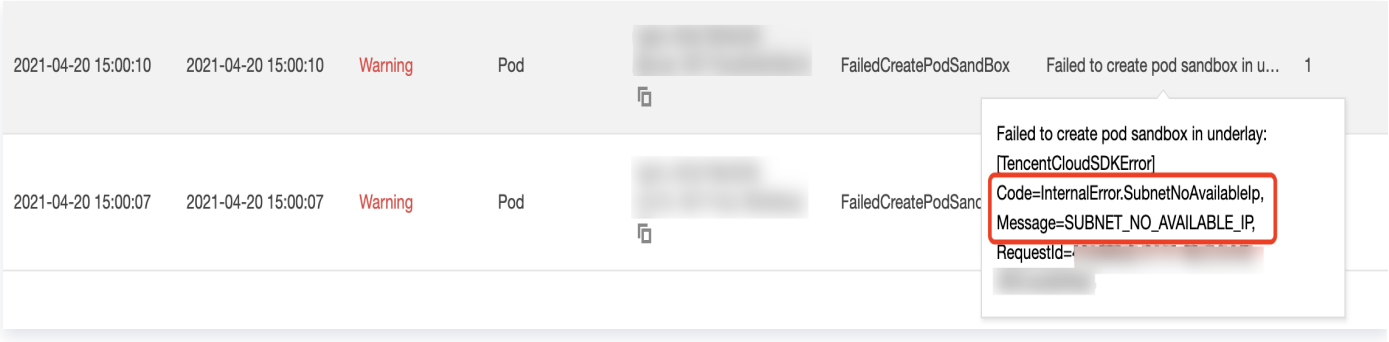
What should I do if the Pod fails to schedule because of insufficient subnet IPs?

When a Pod fails to be scheduled due to insufficient subnet IP addresses, you can find two events in the node logs.

- Event 1:



- Event 2:



You can view the YAML of the super node by accessing the TKE console or executing the following command in the command line.

```
kubectl get nodes -oyaml
```

The returned result is as follows:

```
spec:
  taints:
  - effect: NoSchedule
    key: node.kubernetes.io/network-unavailable
    timeAdded: "2021-04-20T07:00:16Z"
```

```
- lastHeartbeatTime: "2021-04-20T07:55:28Z"
    lastTransitionTime: "2021-04-20T07:00:16Z"
    message: eklet node has insufficient IP available of subnet subnet-bok73g4c
    reason: EKLetHasInsufficientSubnetIP
    status: "True"
    type: NetworkUnavailable
```

It shows that the Pod fails to be scheduled due to insufficient subnet IP addresses of the container network. In this case, you need to create super nodes to add subnets and available IP ranges. For how to create a super node, see Creating Super Node.

What are the instructions for using the TKE Serverless cluster security group?

When creating the TKE Serverless cluster Pod, if you do not specify a security group, the default security group will be used. You can also specify a security group for the Pod through `Annotation eks.tke.cloud.tencent.com/security-group-id: security group ID`. Make sure that the security group ID already exists in the region where the workload resides. For more information about this annotation, see Annotation.

How do I set the container termination message?

Kubernetes can set the source of container termination messages through the terminationMessagePath. When a container exits, Kubernetes retrieves the termination message from the specified termination message file in the container's terminationMessagePath field and uses this content to populate the container's termination message. The default value for the message is: `/dev/termination-log`.
Additionally, you can set the terminationMessagePolicy field for the container to further customize the container termination message. The default value for this field is `File`, which retrieves the termination message only from the termination message file. You can set it to FallbackToLogsOnError according to your needs, which means that if the container exits with an error and the termination message file is empty, the last part of the container log output will be used as the termination message.
Sample code:

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    resources:
      limits:
        cpu: 500m
        memory: 1Gi
      requests:
        cpu: 250m
        memory: 256Mi
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: FallbackToLogsOnError
```

With the above configuration, when the container exits with an error and the termination message file is empty, Get Pod will find that the output of stderr is displayed in containerStatuses.

How to use Host parameters?

Note the following when using TKE Serverless clusters:
TKE Serverless clusters do not have nodes but are compatible with Host parameters, such as Hostpath, Hostnetwork: true, and DnsPolicy: ClusterFirstWithHostNet. Note that these

parameters cannot deliver the full capabilities of K8s, as there is no node.
For example, you may want to use Hostpath to share data, but the two Pods scheduled to the same super node will see the Hostpath of different hosts. In addition, if the Pod is rebuilt, Hostpath files will be deleted at the same time.

How do I mount CFS/NFS?

In TKE Serverless clusters, you can use Tencent Cloud's Cloud File Storage (CFS) or mount an external NFS as a volume to a Pod for persistent data storage. A sample YAML to mount CFS/NFS to a Pod is as shown below:

```
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: nfs
      nfs:
        path: /dir
        server: 127.0.0.1
```

- spec.volumes: Set the name, type, and parameters of the volume.
- spec.volumes.nfs: Set the NFS/CFS disk.
- spec.containers.volumeMounts: Set the mount point of the volume in the Pod.

For detailed directions of mounting a volume to a Pod, see Instructions for Other Storage Volumes.

How to speed up container start-up by image reuse?

TKE Serverless supports caching container images to speed up the next startup of the container with the same images.
Conditions for reuse:

1. For Pods with the same Workload, if a Pod is created and terminated at the same AZ within the cache time, the new Pod will not pull the same image by default.
2. If you want to reuse images for Pods with different Workloads (including Deployment,

Statefulset, and Job), use the following annotation:

```
eks.tke.cloud.tencent.com/cbs-reuse-key
```

For the Pods with the same annotation value under the same user account, the start-up image will be reused within cache time as much as possible. It is recommended to enter the image name of the annotation value: `eks.tke.cloud.tencent.com/cbs-reuse-key: "image-name"` .

Cache time: 2 hours.

How do I solve image reuse exceptions?

When image reuse function is enabled, if a Pod is created, `$kubectl describe pod` may see the following errors:

- `no space left on device: unknown`
- `Warning FreeDiskSpaceFailed 26m eklet, eklet-subnet-xxx failed to garbage collect required amount of images. Wanted to free 4220828057 bytes, but freed 3889267064 bytes`

Recovery Method:

No action is required. Wait for a few minutes and the Pod will run automatically.

Reason:

- `no space left on device: unknown`

  When Pods reuse the system disk by default, the existing images on the system disk occupy all the disk space, causing insufficient space for downloading new images, resulting in the error "no space left on device: unknown". TKE Serverless supports a scheduled image recycling mechanism. When the disk space is full, it will automatically delete the redundant images in the system disk to free up available space. (This process may take several minutes.)

- `Warning FreeDiskSpaceFailed 26m eklet, eklet-subnet-xxx failed to garbage collect required amount of images. Wanted to free 4220828057 bytes, but freed 3889267064 bytes`

  This log indicates that the current Pod requires 4220828057 bytes of space to download the image, but only 3889267064 bytes of data have been cleared. The event is generated because there are multiple images on the disk, and only some of them have been cleaned up. TKE Serverless's scheduled image recycling mechanism will continue to clean up images until the new image can be successfully pulled.

What should I do if `Operation not permitted` is reported when I mount an external NFS?

If you encounter an "Operation not permitted" error when using a self-built NFS for persistent storage, you need to modify the /etc/exports file of your self-built NFS by adding the /<path>

<ip-range>(rw,insecure) parameter. An example is shown below:

```
/data/  10.0.0.0/16(rw,insecure)
```

How do I free up a full Pod disk (ImageGCFailed)?

TKE Serverless Pods provide 20 GB of free system disk space by default. If the disk is full, you can free it up in the following ways.

1. Free up unused container images

If 80% of the space is used, the TKE Serverless backend will trigger the container image repossession process to recover the unused images and free up the space. If this process fails, the `ImageGCFailed: failed to garbage collect required amount of images` message will be reported to remind you of the insufficient disk space.
Common causes of insufficient disk space include:

- The business has a lot of temporary outputs. You can confirm this with the `du` command.
- The business holds deleted file descriptors, so disk space is not freed up. You can confirm this with the `lsof` command.

If you want to adjust the threshold for the container image repossession, set the following annotation:

```
eks.tke.cloud.tencent.com/image-gc-high-threshold: "80"
```

2. Clean up exited containers

If your business has been upgraded in-place or a container has abnormally exited, the exited container will be retained until the disk utilization reaches 85%. The cleanup threshold can be adjusted with the following annotation:

```
eks.tke.cloud.tencent.com/container-gc-threshold: "85"
```

If you don't want to have the exited container automatically cleaned up (for example, you need the exit information for further troubleshooting), you can disable the automatic cleanup with the following annotation; however, the disk space cannot be automatically freed up in this case.

```
eks.tke.cloud.tencent.com/must-keep-last-container: "true"
```

> **Note**
> This feature was launched on September 15, 2021 (UTC +8) and is not available on Pods created earlier.

3. Restart Pods with high disk usage

You need to restart a Pod with the following annotation after the container's system disk usage exceeds a certain percentage:

```
eks.tke.cloud.tencent.com/pod-eviction-threshold: "85"
```

Only the Pod is restarted, but the host will not be rebuilt. Normal gracestop, prestop, and health checks are performed for the exit and startup.

> **Note**
> This feature was launched on April 27, 2022 (UTC +8) and can be enabled on Pods created earlier only after they are rebuilt.

9100 port issue

TKE Serverless Pods expose monitoring data via port 9100 by default, and you can access 9100/metrics to get the data by running the following commands:

- Get all metrics:

```
curl -g "http://<pod-ip>:9100/metrics"
```

- We recommend you remove the `ipvs` metric for large clusters:

```
curl -g "http://<pod-ip>:9100/metrics?collect[]=ipvs"
```

If your business requires listening on port 9100, you can avoid conflicts by using other ports to collect monitoring data when creating a Pod. The configuration is as shown below:

```
eks.tke.cloud.tencent.com/metrics-port: "9110"
```

If the port for monitoring data exposure is not changed and the business listens on port 9100 directly, an error will be reported in the new TKE Serverless network scheme, indicating that port 9100 is already in use:

```
listen() to 0.0.0.0:9100, backlog 511 failed (1: Operation not permitted)
```

When this error is reported, you need to add the annotation `metrics-port` to the Pod to change the monitoring port and then rebuild the Pod.

> **Note**
> If the Pod has a public EIP, you need to set up a security group. Pay attention to port 9100 and open required ports.

Load Balancer FAQs

Last updated: 2023-09-27 15:06:33

This document describes the FAQs of CLB, and introduces the causes and solutions of various FAQs of Service/Ingress CLB.

Prerequisites:

- You are familiar with K8s Concepts, such as Pod, workload, Service, Ingress, etc.
- You are familiar with the general operations of TKE Serverless clusters in the TKE console.
- You can use kubectl command line tool to manage the K8s cluster resources.

> **Note:**
> You can manage the K8s cluster resources in various ways. This document describes how to manage K8s cluster resources in Tencent Cloud console and through the kubectl command line tool.

Which Ingress can TKE Serverless create a CLB instance for?

TKE Serverless will create CLB instances for Ingress that meets the following conditions:

Requirements for Ingress resources	Unsupported Features
annotations contains the following key-value pairs: kubernetes.io/ingress.class: qcloud	If you do not want TKE Serverless to create a CLB instance for Ingress (if, for example, you want to use Nginx-Ingress), you only need to make sure the key-value pair is not contained in the annotations.

How do I view the CLB instance created by a TKE Serverless cluster for Ingress?

If TKE Serverless has successfully created a CLB instance for Ingress, it will write the VIP of the CLB instance to the `status.loadBalancer.ingress` of the Ingress resource, and write the following key-value pair to annotations:

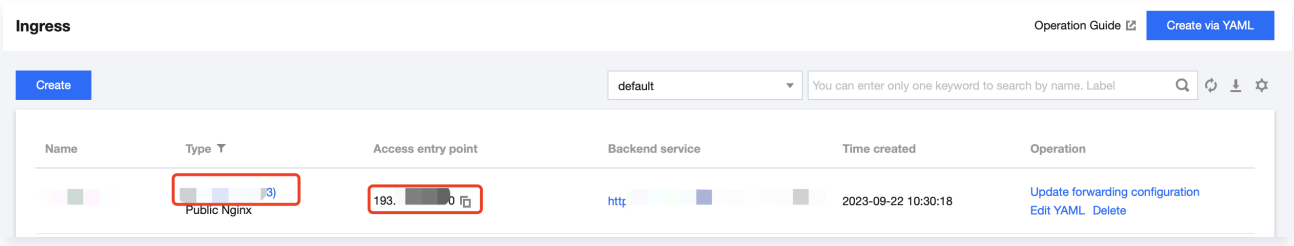
```
kubernetes.io/ingress.qcloud-loadbalance-id: CLB instance ID
```

To view the CLB instance created by TKE Serverless for Ingress, perform the steps below:

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the cluster management

page.

3. On the cluster management page, select **Service and route** > **Ingress** in the left sidebar.
4. On the "Ingress" page, view the CLB instance ID and its VIP, as shown in the image below:



Which Service can TKE Serverless create a CLB instance for?

TKE Serverless will create CLB instances for Service that meets the following conditions:

K8s Version	Requirements for Service resources
All K8s versions supported by TKE Serverless	spec.type is LoadBalancer.
The modified version of K8s (Server GitVersion returned by kubectl version has the "eks." *or "tke."* suffix)	spec.type is ClusterIP, and the value of spec.clusterIP is not None (that is, a non-Headless ClusterIP Service).
The non-modified version of K8s (Server GitVersion returned by kubectl version does not have the "eks." *or "tke."* suffix)	spec.type is ClusterIP, and spec.clusterIP is specified as an empty string ("").

Note:
If the CLB instance is successfully created, TKE Serverless will write the following key-value pair to Service annotations:

```
service.kubernetes.io/loadbalance-id: CLB instance ID
```

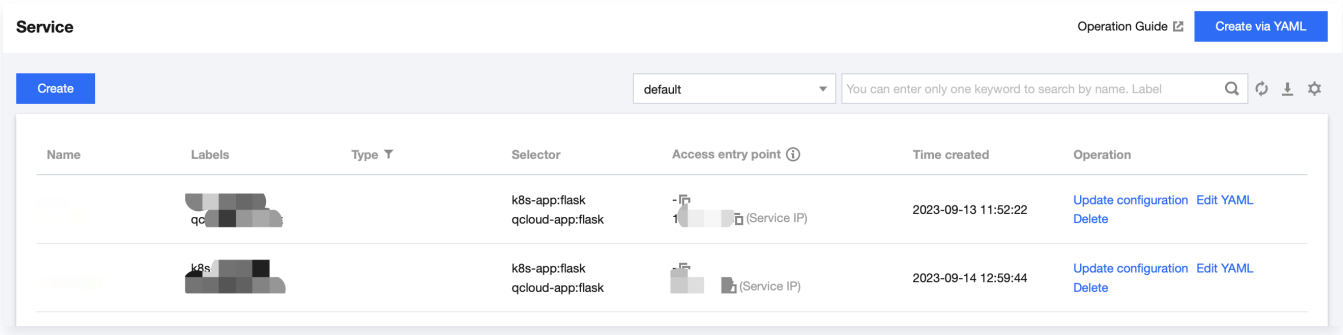
How do I view the CLB instance created by a TKE Serverless cluster for Service?

If TKE Serverless has successfully created a CLB instance for Service, it will write the VIP of the CLB instance to the `status.loadBalancer.ingress` of the Service resource, and write the following key-value pair to annotations:

```
kubernetes.io/ingress.qcloud-loadbalance-id: CLB instance ID
```

To view the CLB instance created by TKE Serverless for Service, perform the steps below:

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the cluster management page.
3. On the cluster management page, select **Service and route** > **Service** in the left sidebar.
4. You can find the CLB instance ID and its VIP on the **Service** page.



Why is the ClusterIP of Service invalid (cannot be accessed normally) or why is there no ClusterIP?

For the Service whose `spec.type` is `LoadBalancer`, TKE Serverless currently does not allocate a ClusterIP by default, or the allocated ClusterIP is invalid (cannot be accessed normally). If you want to use a ClusterIP to access the Service, you can add the following key-value pair to annotations to indicate that TKE Serverless implements a ClusterIP based on a private network CLB instance:

```
service.kubernetes.io/qcloud-clusterip-loadbalancer-subnetid: Service CIDR subnet ID
```

The Service CIDR block subnet ID, which is specified when you create the cluster, is a string in `subnet-*` format. You can view the subnet ID on the basic information page of the CLB instance.

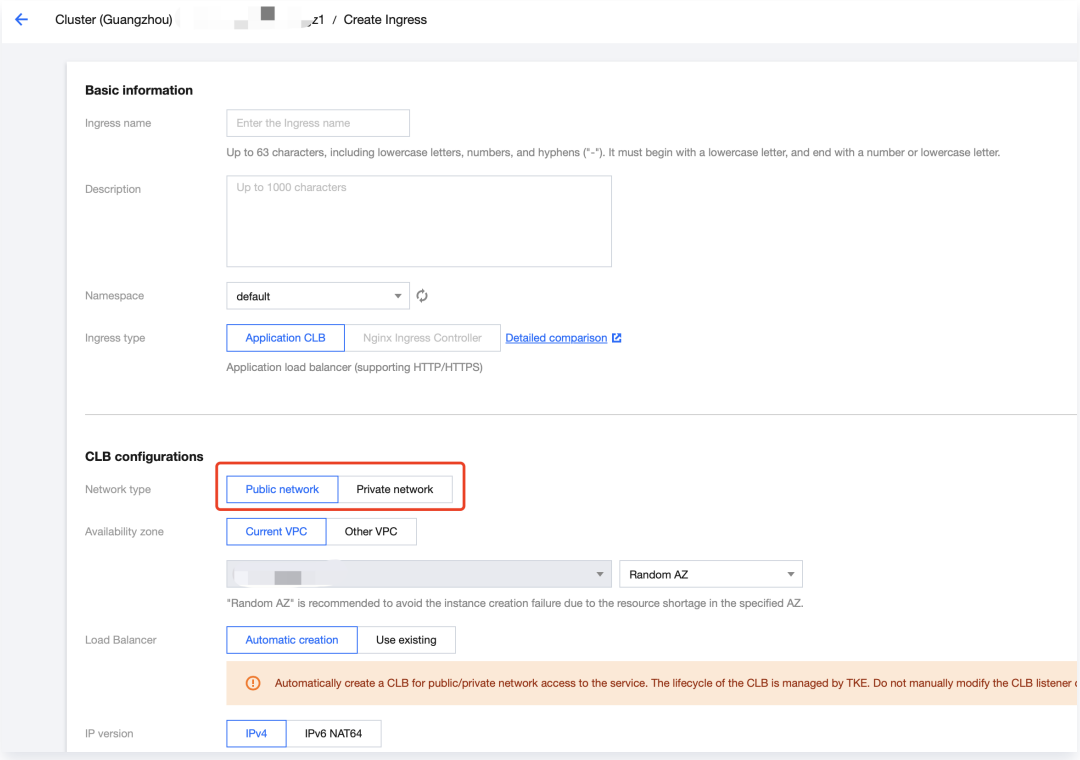
> **Note:**
> This feature is only supported by TKE Serverless clusters using the modified version of K8s (Server GitVersion returned by kubectl version has the "eks.*" or "tke.*" suffix). For early-created TKE Serverless clusters using the non-modified version of K8s (Server GitVersion returned by kubectl version does not have the "eks.*" or "tke.*" suffix), you need to upgrade the K8s version to use this feature.

How do I specify the CLB instance type (public or private network)?

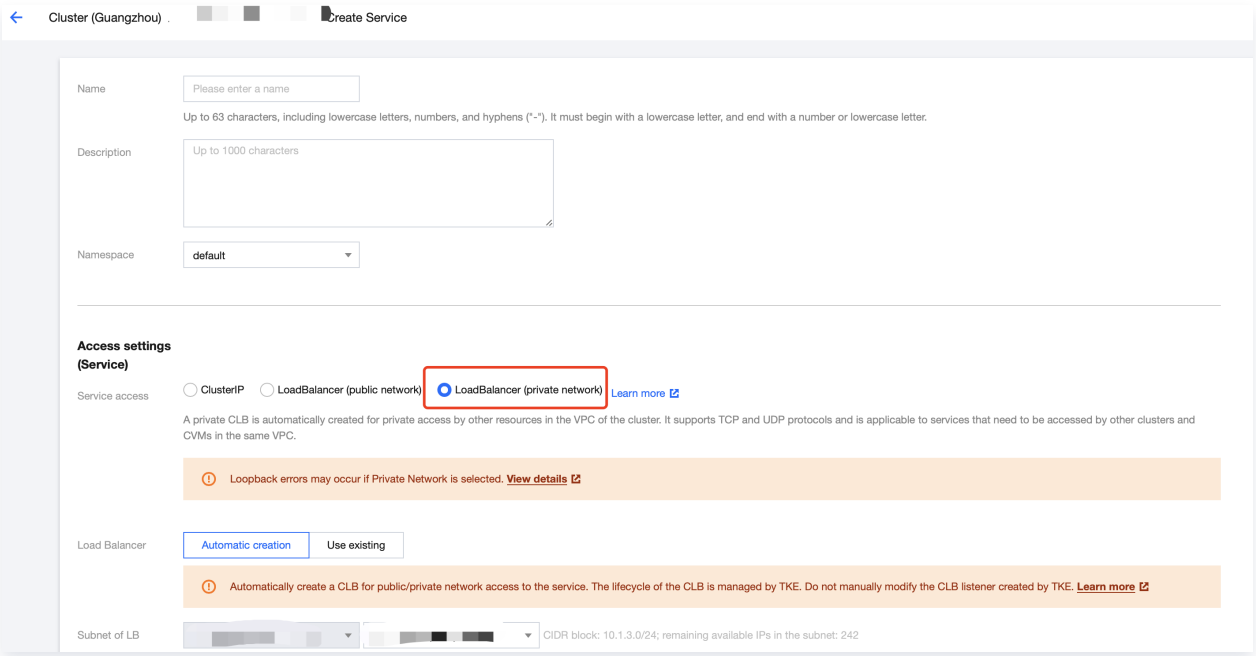
You can specify the CLB instance type via TKE console or Kubectl command line tool.

Specifying via TKE console

- For Ingress, choose "Public Network" or "Private Network" by selecting the "Network Type":



- For Service, the access method is controlled by "Service Access Mode", where "VPC Internal Access" corresponds to the internal CLB instance:



Specifying via the Kubectl command line tool

- The created CLB instance is of the “public network” type by default.
- If you need to create a CLB instance of the "private network" type, you need to add the corresponding annotation for the Service or Ingress.

ResourceType	Add the following key-value pairs in the annotation
Service	service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet ID
Ingress	kubernetes.io/ingress.subnetId: subnet ID

Note:
The subnet ID is a string in the format of subnet-**, and the subnet must be within the VPC specified as the "Cluster Network" during cluster creation. The VPC information can be found in the "Basic Information" section of the Tencent Cloud Console cluster.

How do I specify the existing CLB instance?

You can specify the existing CLB instance via TKE console or Kubectl command line tool.

Specifying via TKE console

When creating a Service or Ingress, you can select **Use existing** to use the existing CLB instance. For Service, you can switch to “Use existing” to use the existing CLB instance through “Update access method” after Service is created.

Specifying via the Kubectl command line tool

When creating a Service/Ingress or modifying a Service, you need to add the corresponding annotation for the Service or Ingress.

ResourceType	Add the following key-value pairs in the annotation

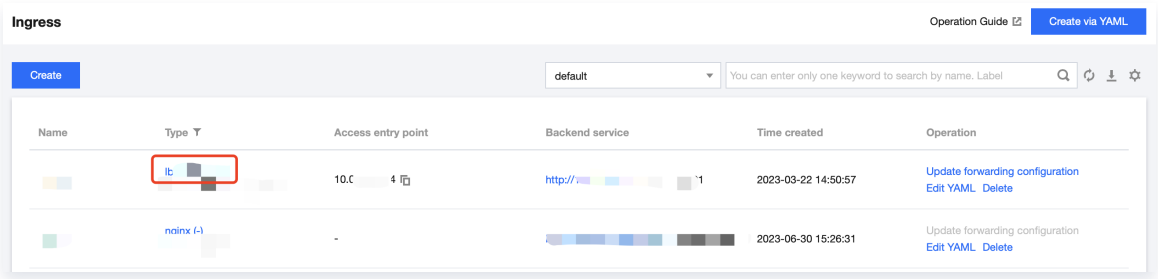
Service	service.kubernetes.io/tke-existed-lbid: CLB instance ID
Ingress	kubernetes.io/ingress.existLbId: CLB instance ID

Note:
The existing CLB instance cannot be the CLB instance created by TKE Serverless for Service or Ingress, and TKE Serverless does not support multiple Services/Ingresses to share the same existing CLB instance.

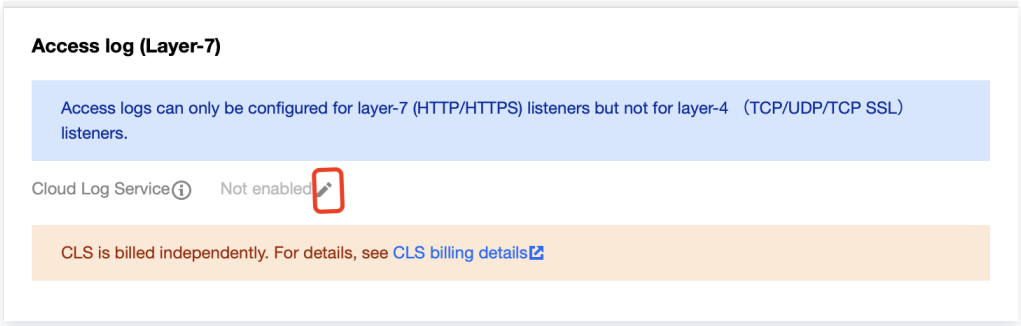
How do I view the access log of a CLB instance?

Only layer-7 CLB instances support configuring access logs, but the access log feature of layer-7 CLB instances created by TKE Serverless for Ingress is disabled by default. You can enable the access log feature for a CLB instance on the details page of the CLB instance, as shown below:

1. Log in to the TKE console and click Cluster in the left sidebar.
2. On the cluster list page, click the ID of the target cluster to go to the cluster management page.
3. On the cluster management page, select **Service and route** > **Ingress** in the left sidebar.
4. Navigate to the "Ingress" page, select the CLB instance ID to access the CLB basic information page, as shown below:



5. In the "Access Logs (Layer 7)" section of the CLB Basic Information page, click to enable it in the pop-up window, as shown below:



Why does TKE Serverless not create a CLB instance for Ingress or Service?

Please refer to the answers for Which Ingress can TKE Serverless create a CLB instance for and Which Service can TKE Serverless create a CLB instance for to confirm whether the corresponding resources meet the requirements. If the requirements are met but the CLB instance is not created successfully, you can use the `kubectl describe` command to view the related events of the "resource".

TKE Serverless typically outputs relevant Warning type events. An example is shown below, where the output event indicates that there are no available IP resources in the subnet, so the CLB instance cannot be created successfully.

```
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl describe svc -n demo normal-clusterip-nginx
Name:              normal-clusterip-nginx
Namespace:         demo
Labels:            <none>
Annotations:       service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-j55z6ay7
Selector:          k8s-app=nginx,qcloud-app=nginx
Type:              ClusterIP
IP:
Port:              tcp-80-80  80/TCP
TargetPort:        80/TCP
Endpoints:         10.0.24.70:80,10.0.24.81:80,10.1.0.174:80
Session Affinity:  None
Events:
  Type     Reason                  Age                From                  Message
  ----     ------                  ----               ----                  -------
  Warning  CreatingLoadBalancerFailed  46m                clb-service-controller  Failed to syncLbWithService (will retry): failed to ensure internal load balan
cer for 'service:demo/normal-clusterip-nginx', [TencentCloudSDKError] Code=ResourceInsufficient, Message=The number of IP in subnet subnet-j55z6ay7 is not enough.,
 RequestId=3c6f9d95-650d-4e98-8171-006fbb0d0f88
  Warning  CreatingLoadBalancerFailed  29m                clb-service-controller  Failed to syncLbWithService (will retry): failed to ensure internal load balan
cer for 'service:demo/normal-clusterip-nginx', [TencentCloudSDKError] Code=ResourceInsufficient, Message=The number of IP in subnet subnet-j55z6ay7 is not enough.,
 RequestId=8567bab4-3f57-4f3d-9e99-f34d8c28d5e8
```

How do I use the same CLB in multiple Services?

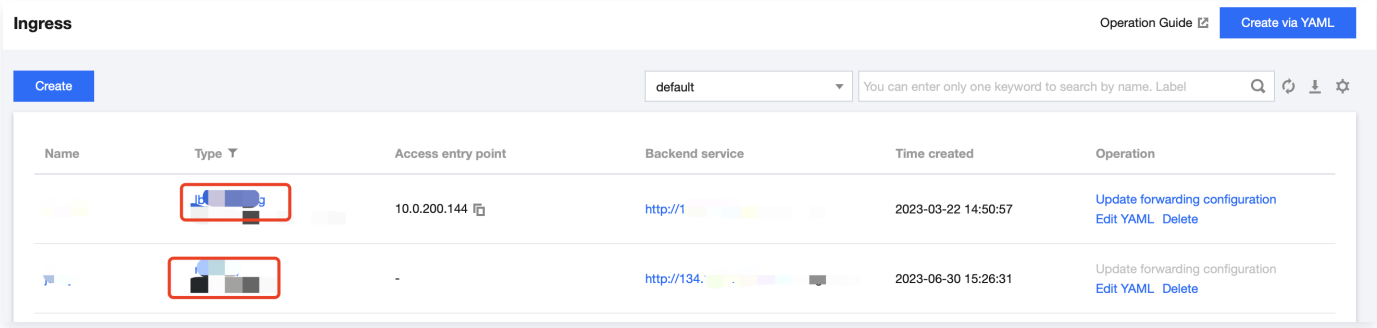
For TKE Serverless clusters, multiple Services cannot share the same CLB instance by default. If you hope that a Service uses the CLB instance occupied by other Services, add this annotation and specify the value as "true": `service.kubernetes.io/qcloud-share-existed-lb: true`. For more information about this annotation, see Annotation.

Why do I fail to access CLB VIP?

Please follow the steps below to analyze:

Viewing the CLB instance type

1. Navigate to the CLB basic information page by selecting the CLB instance ID on the "Service" or "Ingress" page, as shown below:



2. You can view the **Instance type** of the above CLB instance on the CLB basic information page.

Confirming whether the environment for accessing CLB VIP is normal

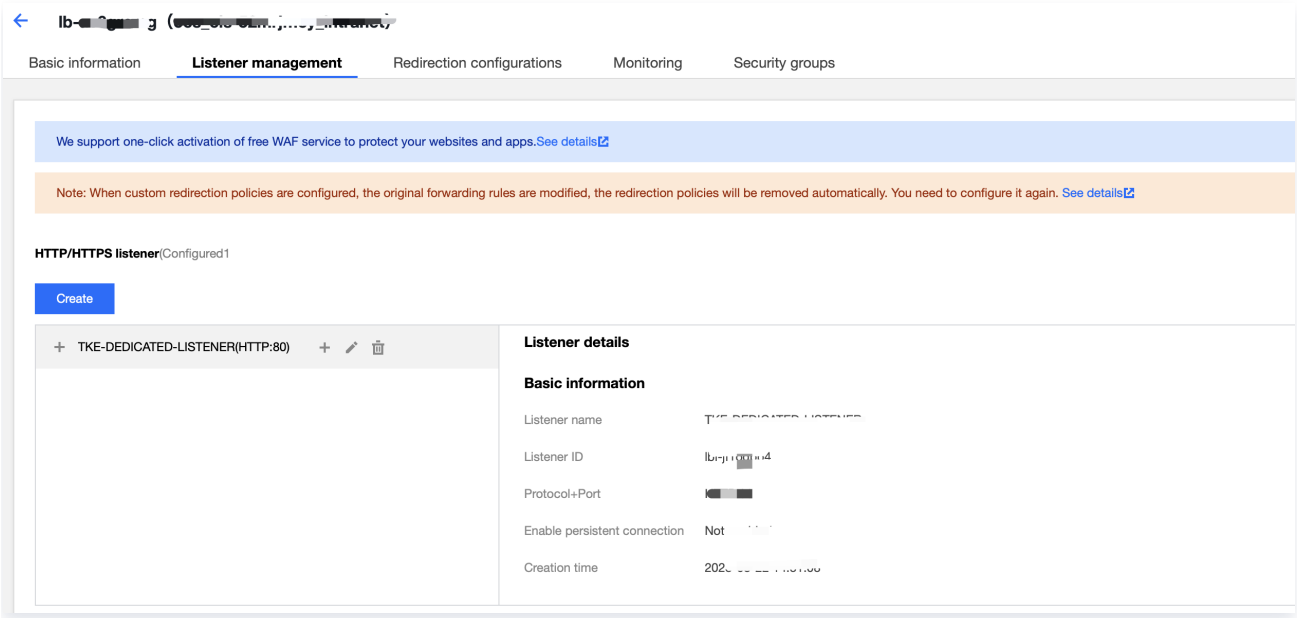
- If the CLB instance "Instance Type" is set to private, its VIP can only be accessed within the associated VPC. Since the IP addresses of Pods in TKE Serverless clusters are the elastic network interface IPs within the VPC, they can access the VIP of any Service or Ingress CLB instance within the cluster.

> **Note**
> LoadBalancer systems typically have loopback issues (e.g., Azure Load Balancer Troubleshooting Guide). Avoid accessing the services provided by the workload from the Pods belonging to the workload through the VIP exposed by the workload (via Service or Ingress). In other words, Pods should not access their own services through VIPs (including "private network type" and "public network type"). Otherwise, it may result in increased latency or inaccessibility (when there is only one RS/Pod under the corresponding VIP rule).

- If the CLB instance "Instance Type" is public, its VIP can be accessed in environments with public network access capabilities. To access the public VIP within the cluster, ensure that public network access has been enabled for the cluster through the configuration of a NAT gateway or other methods.

Viewing the RS in CLB including (and only including) the IP and port of the expected Pods

On the CLB management page, click the **Listener management** tab to view the forwarding rules (layer-7 protocol) and the bound backend services (layer-4 protocol). The IP address is expected to be the IP address of each Pod. An example is as follows:



Confirming whether the corresponding Endpoints are normal

If you have correctly set labels for the workload and selectors for the Service resource, you can view the ready address list of Pods in the corresponding Endpoints of the Service using the `kubectl get endpoints` command after the Pods are running successfully. The example is shown below:

```
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get pod -n demo -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE                                                NOMINATED NODE   READINESS GATES
nginx-7c7c647ff7-4b8n5   1/1     Running   0          18h   10.0.24.70   cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0   <none>           <none>
nginx-7c7c647ff7-b95zc   1/1     Running   0          18h   10.0.24.81   cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0   <none>           <none>
nginx-7c7c647ff7-ckhb9   1/1     Running   0          18h   10.1.0.174   cls-4plj5ho0-virtual-kubelet-subnet-l9dkwcyx-0   <none>           <none>
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get deployment -n demo nginx
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   3/3     3            3           18h
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get svc -n demo internal-clb-nginx
NAME                TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
internal-clb-nginx   LoadBalancer   10.0.15.28   10.0.15.28    80:30082/TCP   18h
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get ep -n demo internal-clb-nginx
NAME                ENDPOINTS                                        AGE
internal-clb-nginx   10.0.24.70:80,10.0.24.81:80,10.1.0.174:80   18h
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$
```

Pods that are created but in an abnormal state will be listed in the not-ready address list of

the corresponding Endpoints of the Service. The example is shown below:

```
[flyma@VM_227_154_centos ~]$ kubectl get pod -n demo -o wide
NAME                     READY   STATUS             RESTARTS   AGE   IP           NODE                                                NOMINATED NODE   READINESS GATES
nginx-7c7c647ff7-4b8n5   0/1     ImagePullBackOff   0          18h   10.0.24.70   cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0      <none>           <none>
nginx-7c7c647ff7-b95zc   1/1     Running            0          18h   10.0.24.81   cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0      <none>           <none>
nginx-7c7c647ff7-ckhb9   1/1     Running            0          18h   10.1.0.174   cls-4plj5ho0-virtual-kubelet-subnet-l9dkwcyx-0      <none>           <none>
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$
[flyma@VM_227_154_centos ~]$ kubectl get ep -n demo internal-clb-nginx -o yaml
apiVersion: v1
kind: Endpoints
metadata:
  annotations:
    endpoints.kubernetes.io/last-change-trigger-time: "2020-12-22T11:41:25+08:00"
  creationTimestamp: "2020-12-21T08:58:22Z"
  name: internal-clb-nginx
  namespace: demo
  resourceVersion: "2482053203"
  selfLink: /api/v1/namespaces/demo/endpoints/internal-clb-nginx
  uid: 7893ab4c-f8b4-45d9-a4df-21daabb1c42e
subsets:
- addresses:
  - ip: 10.0.24.81
    nodeName: cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0
    targetRef:
      kind: Pod
      name: nginx-7c7c647ff7-b95zc
      namespace: demo
      resourceVersion: "2475611832"
      uid: 38c40d5f-9ab6-4647-9e75-642b8203268e
  - ip: 10.1.0.174
    nodeName: cls-4plj5ho0-virtual-kubelet-subnet-l9dkwcyx-0
    targetRef:
      kind: Pod
      name: nginx-7c7c647ff7-ckhb9
      namespace: demo
      resourceVersion: "2475610295"
      uid: c839e449-aa5b-4cf7-a5ff-a35df7262811
  notReadyAddresses:
  - ip: 10.0.24.70
    nodeName: cls-4plj5ho0-virtual-kubelet-subnet-e714bobz-0
    targetRef:
      kind: Pod
      name: nginx-7c7c647ff7-4b8n5
      namespace: demo
      resourceVersion: "2482053202"
      uid: f5d2a2d3-676d-4323-98dd-4aaf168e6e22
  ports:
  - name: tcp-80-80
    port: 80
    protocol: TCP
[flyma@VM_227_154_centos ~]$
```

Note:

You can run the `kubectl describe` command to view the cause of the abnormal Pods. The command is as follows:

```
kubectl describe pod nginx-7c7c647ff7-4b8n5 -n demo
```

Confirming whether Pods can provide services normally

Even Pods in the Running state may not be able to provide services normally due to some exceptions. For example, the specified protocol + port are not listened to, the internal logic of Pods is incorrect, the process is blocked, etc. You can run the `kubectl exec` command to log in to the Pod, and run the `telnet/wget/curl` command or use a custom client tool to directly access the Pod IP+ port. If the direct access fails in the Pod, you need to further analyze the reasons why the Pod cannot provide services normally.

Confirm whether the security group bound to the Pod allows the protocol and port of the service provided by the Pod

The security group controls the network access policy of Pods, just like the IPTables rules in the Linux server. Please check based on the actual situation:

Creating a workload via TKE console

The interactive process requires specifying a security group, which TKE Serverless will use to control the network access policies of Pods. The selected security group will be stored in the workload's `spec.template.metadata.annotations` and ultimately added to the Pods' annotations. An example is shown below:

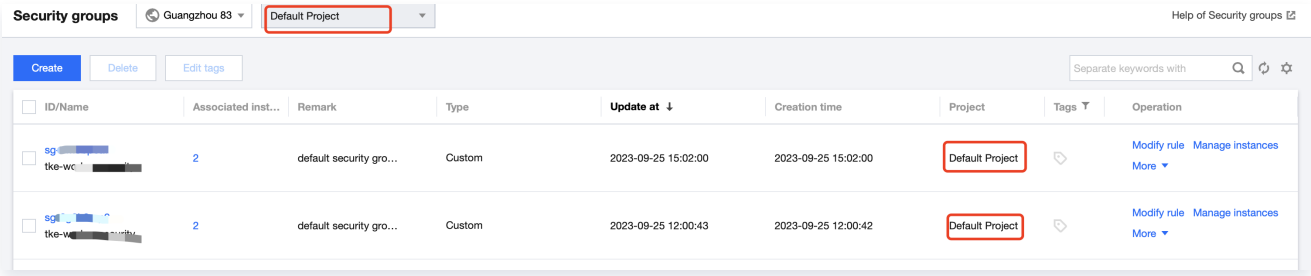
```
[flyma@VM_227_154_centos ~]$ kubectl get pod -n demo nginx-7c7c647ff7-ckhb9 -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    eks.tke.cloud.tencent.com/cpu-type: intel
    eks.tke.cloud.tencent.com/security-group-id: sg-7wsmhh5d
  creationTimestamp: "2020-12-21T08:55:17Z"
  generateName: nginx-7c7c647ff7-
  labels:
    k8s-app: nginx
    pod-template-hash: 7c7c647ff7
    qcloud-app: nginx
  name: nginx-7c7c647ff7-ckhb9
  namespace: demo
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: nginx-7c7c647ff7
    uid: dc06824c-425c-4229-929d-46d9afe393b6
  resourceVersion: "2475610295"
  selfLink: /api/v1/namespaces/demo/pods/nginx-7c7c647ff7-ckhb9
  uid: c839e449-aa5b-4cf7-a5ff-a35df7262811
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    resources:
```

Creating a workload through the kubectl command

If you create a workload via the kubectl command line tool and do not specify a security group for Pods (by adding annotations), TKE Serverless will use the default security group of the default project in the same region under the account. The directions are as follows:

1. Log in to the VPC console and select Security Group in the left sidebar.
2. Select **Default project** of the same region at the top of the **Security group** page.As

shown in the image below:



3. You can view the default security group in the list, and click **Modify rule** to view the details.

Contact Us

If the problem persists, you can contact the TKE team through online customer service or submit a ticket for assistance.

Supernodes FAQs

Last updated：2023-09-26 16:25:01

How can I prevent a Pod from being scheduled on a specific pay-as-you-go super node?

By default, when a pay-as-you-go super node node pool is added to a regular TKE cluster, Pods will be automatically scheduled on the super node if there are insufficient Node resources. In a Serverless cluster, Pods will be automatically scheduled on multiple pay-as-you-go super nodes at random.
If you do not want to schedule Pods on a specific pay-as-you-go super node (representing a subnet/availability zone), you can block the super node and prevent scheduling using the following two methods:

- You can **block** the node through the Tencent Cloud Container Service Console. For more information, please refer to Blocking Nodes.
- Prohibit scheduling by executing the following command through the command line:

```
$kubectl cordon $pay-as-you-go_super_node_name
```

How can I prevent a regular TKE cluster from automatically scheduling Pods on pay-as-you-go super nodes when resources are insufficient?

You need to create a configmap called “eks-config” in the kube-system namespace by running the following command:

```
$kubectl create configmap eks-config --from-literal=AUTO_SCALE_EKS=false
```

Set the value of AUTO_SCALE_EKS to false to disable the automatic scheduling mechanism of the regular TKE cluster to pay-as-you-go super nodes.

How can I manually schedule a Pod to a pay-as-you-go super node?

By default, pay-as-you-go super nodes automatically add Taints to lower their scheduling priority. To manually schedule a Pod on a pay-as-you-go super node or specify a super node, you usually need to add the corresponding Tolerations to the Pod. However, not all Pods can be scheduled on pay-as-you-go 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 as follows:

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

TKE's management component will determine whether the Pod can be scheduled on a pay-as-you-go super node. If it is not supported, the Pod will not be scheduled on the pay-as-you-go super node.

How can I force a Pod to be scheduled on a pay-as-you-go super node, regardless of whether the super node supports the Pod?

> **Note**
> Forcing scheduling to a pay-as-you-go super node may cause the Pod creation to fail. For possible reasons, please refer to How to Manually Schedule a Pod on a Pay-as-you-go Super Node.

To force the scheduling of a Pod onto a pay-as-you-go super node, in addition to specifying a nodeselector or nodename for the Pod, you must also add the corresponding Tolerations.

1. Specify `nodeselector` in Pod Spec, as shown below:

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

   Or specify `nodename` in Pod Spec, as shown below:

```
spec:
  nodeName: $pay-as-you-go_super_node_name
```

2. Add tolerations for the Pod, as shown below:

```
spec:
  tolerations:
  - effect: NoSchedule
    key: eks.tke.cloud.tencent.com/eklet
    operator: Exists
```

How can I customize the DNS for pay-as-you-go super nodes?

TKE Serverless clusters support the pay-as-you-go super node feature. You can achieve capabilities such as custom DNS by defining annotations in the YAML file. For more information, please refer to Pay-as-you-go Super Node Annotation.

About OPS
Basic Monitoring

Last updated: 2023-09-26 16:25:53

Basic Monitoring

Why is a node assigned more CPU cores and memory than the node resource specification?

Reason: CPU cores and memory assigned to a node are calculated based on the CPU and memory requests in each Pod on the node, but the requests of failed Pods are not subtracted in the calculation.
Example: The node specification is 4 cores and 8 GB of memory, and there are currently 3 Pods running on the node (resource request usage is as follows):

- Pod 1's requests are two CPU cores and 4 GB memory during normal operations.
- Pod 2's requests are one CPU core and 2 GB memory during normal operations.
- Pod 3 is in **Failed** status, and its requests are 0.5 CPU core and 1 GB memory.

At this point, the remaining schedulable resources on the node are 4-2-1=1 core and 8-4-2=2 GB. Pod 4's requests are 0.8 cores and 1.5 GB, which meet the scheduler's requirements and are normally scheduled to this node. Now, there are a total of 4 Pods on the node, with 3 running normally and 1 in an abnormal state. The node-level allocation is 4.3 cores and 8.5 GB (since the calculation does not exclude the failed Pod, it exceeds the node specification).
This issue has been fixed in the May release, meaning that the calculation of node resource allocation now excludes abnormal Pods.

Why the Pod status is normal, but the `k8s_workload_abnormal` monitoring metric is abnormal?

Reason: The metric status is subject to whether Pods of the workload are normal, and whether a Pod is normal is subject to the four types in `pod.status.condition`. `k8s_workload_abnormal` will be considered normal only when the four metrics are all `True` at the same time; otherwise, it will be considered abnormal.

- PodScheduled: The Pod has been scheduled to a node.
- ContainersReady: All containers in the Pod are ready.
- Initialized: All init containers have completed successfully.
- Ready: The Pod can provide the Service to requests and should be added to the load balancer pool of the corresponding Service.

Causes of `tke-monitor-agent` DaemonSet errors

Symptom	Cause	Solution

The domain name `receiver.barad.tencentyun.com` failed to be resolved, and the metric failed to be reported, so the cluster didn't have the monitoring data.	The node DNS was modified.	Add `hostAlias` to the `tke-monitor-agent` DaemonSet as follows:
		`hostAliases:`
		`- hostnames:`
		`  - receiver.barad.tencentyun.com`
		`  ip: 169.254.0.4`

About Services

Last updated: 2023-09-26 16:26:22

FAQs About Service Creation

Why must a service name be unique?

A service name is a unique identifier of a service in the current cluster. Services access each other through the service name and access port.

Can I use a third-party image instead of a Tencent Cloud or Docker Hub image when creating services?

Yes. You can log in to your CVM instance and run the `docker login` command to log in to the third-party image repository and pull the image.

What are the prerequisites for using public network services?

Make sure that the CVM instances in the cluster have public bandwidth; otherwise, public network services may fail to be created.

How do I configure memory and CPU limits?

For more information, see Setting the Resource Limit of Workload.

What does the "Privileged" option mean when I am creating a service?

If this option is enabled, applications in the container will have true root permission. We recommend you enable it when you need to perform higher-level system operations on applications in the container, such as building an NFS server.

Can I create a load balancer with a monthly subscription billing mode for my service?

Load balancers with a monthly subscription billing mode cannot be deleted synchronously when a service is deleted, resulting in wasted resources and user load balancer waste. Load balancers automatically created by services use the pay-as-you-go billing mode. Users can utilize existing monthly-subscribed load balancers by using the existing CLB option. For more information, please refer to Using Existing CLB with Services and Service Working Principles.

Can I specify the security group for a CLB instance when creating it?

Yes. Currently, you can use the following two options to specify the security group for a CLB instance when a service uses it:

- Use an existing CLB. You can create a CLB and configure the security group, and then mount it to the service. For more information, see Using Existing CLBs.
- You can configure a security group in the service using `TkeServiceConfig`. The corresponding security group will be used when creating the load balancer based on the configuration. To use this feature, please contact us online to apply.

> **Note**
> When accessing services within a cluster, it is recommended not to use the load balancer IP to avoid connectivity issues. Generally, a layer-4 load balancer binds multiple nodes as real servers (RS). To ensure proper functionality, clients and RS should not be on the same CVM instance, as this may cause packet loopback failure.
>
> When a Pod accesses the load balancer, the Pod's IP is the source IP. The load balancer does not perform SNAT to convert the source IP to the Node IP when transmitting within the private network. As a result, the load balancer cannot determine which Node sent the packet, rendering the loopback avoidance policy ineffective, and all RS may be forwarded. If the packet is forwarded to the Node where the client is located, the load balancer will not receive the response, leading to connectivity issues.

FAQs About Updating the Number of Service Containers

What should I pay attention to when I update the number of containers?

Confirm whether CPU and memory resources are sufficient. If the resources are insufficient, a container may fail to be created.

Can I set the number of containers to 0?

Yes. You can set the number of containers to 0 to release the resources while retaining service configurations.

FAQs About Service Configuration Update

Is rolling update supported?

Both rolling update and quick update are supported.

Can I switch from a public network CLB instance to a private network CLB instance?

Yes, you can switch public network to VPC private network, or switch VPC private network to public network, and switch between different subnets of a VPC. For more information, see

Service lifecycle management.

> **Note**
> - If the service is responsible for lifecycle management of the CLB instance, the CLB instance and its public network IP will be released.
> - The process of switching from the public network to the private network is not instantaneous. It takes a certain amount of time to deactivate the public network CLB instance and activate the private network CLB instance. We recommend you configure a private network service resource in the cluster, conduct a test, and delete the original public network service resource after the traffic switch is completed.

FAQs About Service Deletion

Will the CLB instance auto-created by a service be terminated after I delete the service?

When a service is deleted, the CLB instance auto-created at the time of the service creation will be deleted simultaneously. If an existing CLB instance is selected at the time of service creation, the CLB instance will not be affected at all.

Is business data affected by deleting a service?

The business container will not be deleted and business data will not be affected if the service is deleted. No need to back up data in this regard.

FAQs About Service Running

How do I set the container system time to UTC+8 time?

The container uses UTC time by default. Users often encounter the problem of 8 hours difference between the container system time and UTC+8 time. You can create a time zone file in dockerfile to solve this issue. For more information, see solve the inconsistent time zone problem in the container.

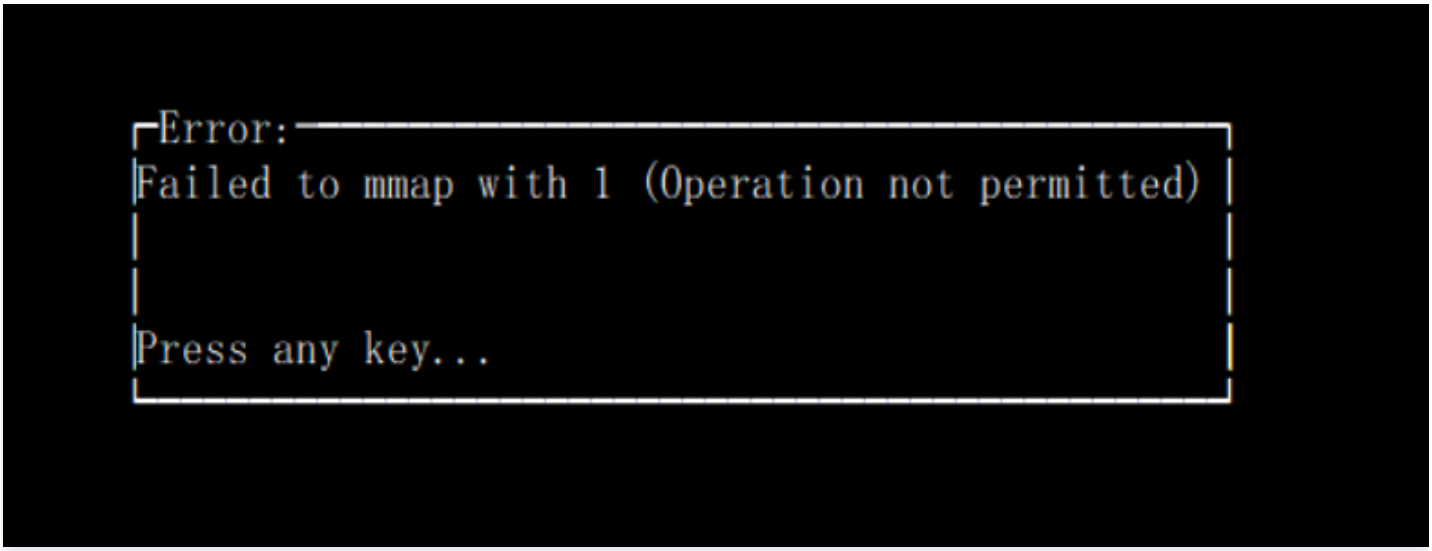
What should I do when some Docker Hub images, such as Ubuntu, PHP, and BusyBox, encounter exceptions in TKE?

If no start command is set or the default start command is `bash`, the container will exit after start. To keep the container running, make sure that the process whose PID is `1` in the container is a resident process; otherwise, the container will exit when this process ends. For some images such as CentOS, you can create services by using `/bin/bash` as the running

command and `-c sleep 800000` as the running parameter. `-c` and `sleep 800000` must be entered in different rows in the console.
Currently, images that cannot be started when default parameters are used include Clear Linux, ROS, Mageia, Amazon Linux, Ubuntu, Clojure, CRUX, GCC, Photon, Java, Debian, Oracle Linux, Mono, Bash, buildpack-deps, Go, Source Mage, Swift, OpenJDK, CentOS, BusyBox, Docker, Alpine, IBM Java, PHP, and Python.

What should I do if an error message "Operation not permitted" appears when a container is executing `perf top -p` to check the process CPU status?

When running `perf top -p` in a container to view the CPU usage of a process, an "Operation not permitted" message appears, as shown below:



Docker's default configuration blocks critical system calls, as `perf_event_open` may leak a significant amount of information on the host and is therefore prohibited. If you need to use this call, please configure a privileged container or modify the Pod YAML field `privileged` to `true`. Be sure to assess the security risks involved.

About Remote Terminals

Last updated: 2023-09-26 16:28:32

I don't have bash in my container.

In this case, you can enter the command you want to run in the command line, and the result of the command will be displayed on the screen. You can regard the command line as a lite edition of bash without functions such as autocomplete. We recommend running the bash installation command before proceeding.

Why is the software installation so slow when I run "apt-get"?

If you experience slow software installation speeds when using `apt-get`, it may be due to slow access to foreign software repositories. We provide acceleration solutions for different operating system containers. You can follow the appropriate steps based on your situation:

Supports and Limits

Before selecting an acceleration solution, please accurately identify the container's operating system to avoid issues during the process. You can use the following methods for identification:

- For Ubuntu systems: Run the command `cat /etc/lsb-release` to verify if the `/etc/lsb-release` file exists.
- For CentOS systems: Run the command `cat /etc/redhat-release` to verify the existence of the `/etc/redhat-release` file.
- Debian system: Run the command `cat /etc/debian_version` to verify the existence of the `/etc/debian_version` file.

Solution

Ubuntu 16.04

For containers with Ubuntu 16.04 as the operating system, you can run the following command in the terminal to set the apt source to Tencent Cloud's source server:

```
cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe
multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe
multiverse
```

```
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe
multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe
multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted
universe multiverse
ENDOF
```

CentOS 7

For containers running CentOS 7, you can execute the following steps in the terminal to directly modify the repository address and improve the installation speed:

1. Copy and run the following code:

```
cat << ENDOF > /etc/yum.repos.d/CentOS-Base.repo
[os]
name=Qcloud centos os - \$basearch
baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/os/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
[updates]
name=Qcloud centos updates - \$basearch
baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/updates/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[centosplus]
#name=Qcloud centosplus - \$basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/centosplus/\$basear
ch/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[cloud]
#name=Qcloud centos contrib - \$basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/cloud/$basearch/op
enstack-kilo/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[cr]
#name=Qcloud centos cr - \$basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/cr/\$basearch/
#enabled=1
```

```
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
[extras]
name=Qcloud centos extras - \$basearch
baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/extras/\$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
#[fasttrack]
#name=Qcloud centos fasttrack - \basearch
#baseurl=http://mirrors.tencentyun.com/centos1/\$releasever/fasttrack/\$basearch
/
#enabled=1
#gpgcheck=1
#gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
ENDOF
```

2. Run the following command to empty and rebuild the YUM cache.

```
yum clean all && yum clean metadata && yum clean dbcache && yum makecache
```

Debian System

For containers with Debian-based systems, you can run the following command in the terminal to set the apt source to Tencent Cloud's source server:

```
cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/debian stretch main contrib non-free
deb http://mirrors.tencentyun.com/debian stretch-updates main contrib non-free
deb http://mirrors.tencentyun.com/debian-security stretch/updates main

deb-src http://mirrors.tencentyun.com/debian stretch main contrib non-free
deb-src http://mirrors.tencentyun.com/debian stretch-updates main contrib non-free
deb-src http://mirrors.tencentyun.com/debian-security stretch/updates main
ENDOF
```

Issue Summary

Modifying the repository address directly in the container is a temporary solution. When the container is rescheduled, your changes will be lost. It is recommended to address this issue when creating the image. The specific steps are as follows:
To modify the Dockerfile for creating a container image, add the information provided in the corresponding operating system's solution to the RUN field of the Dockerfile. For example, in

a Dockerfile for an Ubuntu-based image, include the following content:

```
RUN cat << ENDOF > /etc/apt/sources.list
deb http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe
multiverse
deb http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted universe
multiverse
#deb http://mirrors.tencentyun.com/ubuntu/ xenial-proposed main restricted universe
multiverse
#deb http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted universe
multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe
multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted universe
multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted
universe multiverse
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-proposed main restricted
universe multiverse
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted
universe multiverse
ENDOF
```

I didn't find tools such as vim or netstat after I logged in to a container.

You can download the required tools by running commands such as `apt-get install vim` and `net-tools` (`yum install vim` on CentOS).

The system prompted that no tool was found when I run the apt-get install command.

You can install the software as follows:

1. Run the following command to update the software list.

```
apt-get update
```

2. Run the following command to install the software (run `yum updateinfo` on CentOS).

```
apt-get install
```

How do I use an in-house tool in a container?

Enter the remote terminal page, click “File Assistant” and download files.

How do I copy a live file such as dump or log for local analysis?

Enter the remote terminal page, click “File Assistant” and download files.

Why can't I upload a file to a container or download a file to the local system?

That is probably because the tar program is not installed in your container image. You can install it by running `apt-get install vim` or `net-tools` (`yum install vim` on CentOS) and try again.

Why can’t I find tools that I installed previously?

When Kubernetes re-schedules your container, it will pull an image to create a new container. Tools that do not exist in the image will not be in the new container. We recommend installing some common troubleshooting tools when making images.

How do I copy text in the console?

Simply select and copy the text.

How do I paste the copied text?

Press Shift + Insert to paste the copied text.

Why is the connection broken?

This might have happened if you modified the container status by performing actions on the container/CVM in other related products or services. This might also have happened if the page has been idle for 3 minutes or longer. In both cases the server will terminate the connection.

What if I receive the error "TERM environment variable not set" when I run commands such as "top"?

Run "export TERM linux".

Why does the bash prompt only display "<" and a part of the path when I enter a directory with a long absolute path?

That is because the bash prompt is set to display "username@hostnamecurrentdirectory" by default. If the current path is too long, bash will only display "<" and the last part of the path by default.

Event FAQs

Last updated: 2023-09-26 16:28:50

Error: Back-off while attempting to restart failed Docker container.

Description: If the error message contains "Back-off while attempting to restart failed Docker container," it indicates that an abnormal Docker container is being restarted.
Solution: Check if the Docker process running in the image has exited abnormally. If there is no continuously running process within the image, you can add an execution script on the service creation page.

Error: Fit failure on node due to insufficient CPU resources.

Description: If the error message contains "Insufficient CPU," it indicates that the cluster has insufficient CPU resources.
Solution: The cause is that the node cannot provide enough computing cores. Please modify the CPU limit on the service page or scale out the cluster.

Error: No nodes available to schedule Pods.

Description: If the error message contains "No nodes available to schedule Pods," it indicates that the cluster resources are insufficient.
Solution: The cause is that there are not enough nodes to accommodate instances. Please modify the number of instances or CPU limit on the service page.

Error: Pod failed to fit in any node.

Description: If the error message contains "Pod failed to fit in any node," it indicates that there are no suitable nodes available for instance usage.
Solution: The cause is that the service has configured inappropriate resource limits, resulting in no suitable nodes to host instances. Please modify the instance count or CPU limit on the service page.

Error: Liveness probe failed.

Description: If the error message contains "Liveness probe failed," it indicates that the container health check has failed.
Solution: Please verify if the image container process is functioning properly and if the detection port is configured correctly.

Error: Issues syncing Pod, skipping.

Error: Error syncing Pod, skipping due to failed "StartContainer" with CrashLoopBackOff: "Back-off 5m0s restarting failed container."
Description: If the error message contains "Error syncing Pod, skipping," it indicates that the container process has crashed or exited.
Solution: Please check if there is a continuously running foreground process within the container, and if so, inspect it for any abnormal behavior.

If the above solutions do not resolve your issue, you can consult online customer service or submit a ticket for assistance.

Resource management

Last updated: 2023-09-26 16:29:04

Why should containers set Request/Limit?

During runtime, containers typically consume CPU and memory resources. Without any configuration, the maximum amount of resources a container can use is determined by the allocatable resources of the node it resides on.

- A single node typically runs multiple containers. If there is only one container on a node, the idle resources on the node are wasted when the container does not fully consume them. A modern personal computer can usually run hundreds of processes, and similarly, a node typically runs many containers. However, this raises another issue: containers may compete for resources, while the node's resources are fixed.
- It is necessary to use Limit to control the maximum resource usage of containers. In the world of Kubernetes, Limit is used to define the maximum amount of resources a container can consume. If a container requests resources exceeding its Limit, its usage will be throttled or even evicted to another node.
- Is it sufficient to only set Limit to control the maximum resource usage of containers? Consider this scenario: If a 10-core node runs 100 containers, and each container requires at least 1 core to start and operate normally, none of the containers on the node can function properly. Therefore, Kubernetes uses Request to ensure the minimum resource allocation for containers.

In summary, Kubernetes uses Request and Limit to both ensure and restrict the amount of CPU and memory resources consumed by a container.

What are the units for CPU and memory?

CPU

The default unit for CPU is cores. You can also use decimal values for CPU resources. When you define a container's CPU Request as 0.5, the requested CPU is half of what would be requested for 1.0 core. Additionally, 0.5 is equivalent to 500m, which can be considered as "500 millicpu" and read as "five hundred milli-cores."

Memory

The default memory unit is bytes. You can also use plain integers or integers with a quantity unit suffix to represent memory. For example, the following expressions represent approximately the same value:

```
128974848、129e6、129M、128974848000m、123Mi
```

> **Note**
> Please pay attention to the case of the suffix. For example, if you request 400m of temporary storage, the actual requested value is 0.4 bytes. Similarly, if you request 400Mi bytes (400Mi) or 400M bytes, the actual requested value is 0.4 bytes.

How should one understand Request/Limit?

In Kubernetes, Request/Limit is implemented using CPU Share and CPU Quota technologies.

CPU Share

Suppose multiple containers are running on a machine, how are resources allocated among them? You need to understand the concept of CPU Shares.
CPU Shares is a feature of Linux Control Groups (cgroup) that controls the amount of CPU time available for processes within a container. CPU time refers to the amount of time the CPU spends processing instructions for computer programs or operating systems, rather than the concept of time in everyday life. For example, when a process enters an interrupt, suspension, or sleep state, CPU time does not increase. However, when the process resumes operation, CPU time continues to increase from the point before the interruption.

Characteristics of CPU Shares

1. CPU Shares is a relative concept, not an absolute one. The CPU Shares of a container are used to schedule CPU time among different containers in a relative manner. The numerical value of CPU Shares alone has no inherent meaning. For example, setting the CPU Shares of container A to 512 does not provide information about how much CPU time the container will receive. If the CPU Shares of another container B are set to 1024, it means that container B will receive twice the CPU time of container A. In other words, it still does not provide information about the actual CPU time each container will receive, only their relative amounts. If A and B run simultaneously on a 3-core device, they will theoretically receive 1 core and 2 cores of CPU time, respectively. If they run on a 6-core device, they will receive 2 cores and 4 cores, respectively. If they run on a 0.3-core device, it becomes 0.1 core and 0.2 core.
2. CPU Shares come into play only when there is resource contention.
   - CPU Shares values you set will only be used to allocate CPU cores when both containers A and B are expected to run at the same time.
   - If only one container is running, it can utilize all available CPU resources.
   - When multiple containers are running concurrently, the allocatable CPU cores on the

node are distributed based on the configured CPU Shares values.
    - For non-running containers, even if CPU Shares are configured, it will not affect the allocation of CPU time for running containers.
3. The purpose of CPU Shares design is to maximize CPU resource utilization.
    - Regardless of the CPU Shares value, any container has the potential to utilize all CPU resources on a node.
    - In the event of CPU contention, CPU Shares can be used to determine the amount of CPU time allocated to each container.

CPU Quota

CPU Quota is used to limit the maximum resource usage of a container. Even if a node has remaining resources, the container cannot use more than the specified CPU Quota value.

CPU Share in Kubernetes

In Kubernetes, Request/Limit is implemented through CPU Share and CPU Quota. However, Request has additional implications:

1. Request is an absolute value, not a relative one, ensuring the minimum available resources for a container.
2. Request is used by the scheduler to make decisions, finding the optimal node with available resources greater than the current Pod's Request among the cluster nodes, and scheduling the current Pod on that node.
3. During resource contention, the concept of relative CPU Share values is utilized to allocate CPU resources.

Pod without Request

Pods without a Request can be scheduled to any node, as the remaining allocatable resources on any node meet the Pod's requirements. However, in the event of resource contention, the Pod will not receive any resources, potentially leading to an indefinite lack of resources for the Pod.

Practical Application

Create an interactive busybox Pod in the terminal:

```
kubectl run -i --tty --rm busybox \
  --image=busybox \
  --restart=Never \
  --requests='cpu=50m,memory=50Mi' -- sh
```

Use the following command in the interactive terminal to allow the Pod to fully utilize the available CPU and memory on the current node:

```
while true; do true; done
dd if=/dev/zero of=/dev/shm/fill bs=1k count=1024k
```

In another terminal, view the current resource usage of the Pod.

```
kubectl top pods
NAME      CPU(cores)   MEMORY(bytes)
busybox   460m         65Mi
```

It can be observed that the current busybox Pod's CPU and memory usage are both greater than the Request. However, theoretically, since busybox runs an infinite loop program, it should consume all available CPU resources. Why does it only consume 460m? This is because there are other Pods and processes in the cluster, and they compete for CPU resources through CPU Share.

Summary

- Kubernetes uses Request to guarantee the minimum resource consumption for containers.
- Kubernetes uses Limit to restrict the maximum resource consumption of containers.
- When entering values, be aware of the default units: CPU default unit is cores; memory default unit is bytes.
- In the event of resource contention, Kubernetes allocates resources based on the proportion of Request values specified by different containers.