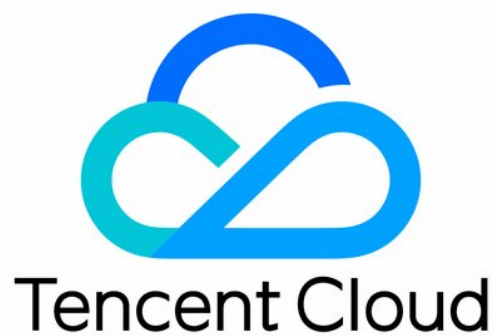


Tencent Kubernetes Engine

FAQs

Product Documentation



Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

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

Contents

FAQs

About OPS

- Basic Monitoring

- Log Collections

TKE General Cluster

- Auto-scaling related

- Cluster FAQ

- How to Choose Containerd and Docker

- External DNS Causes Abnormal Node Initialization

About Services

TKE Serverless Cluster

- TKE Serverless Clusters-related

- Load Balancer FAQs

- Supernodes FAQs

- Image Repository FAQs

Image Repositories

About Remote Terminals

Event FAQs

FAQs

About OPS

Basic Monitoring

Last updated : 2022-10-12 16:05:10

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. A node's specification is 4-core 8 GB MEM, and three Pods are running on it. Below is the resource request usage:

- 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.

The idle resources on the node are $4 - 2 - 1 = 1$ CPU core and $8 - 4 - 2 = 2$ GB memory. Pod 4's requests are 0.8 CPU core and 1.5 GB memory, which meets the scheduler's requirements and is scheduled to the node normally. At this point, the node has four Pods, three normal ones and one failed one, and is assigned 4.3 CPU cores and 8.5 GB memory (as the requests of the failed Pod are not subtracted in the calculation, the node specification is exceeded).

This problem was fixed in the new version in May, that is, requests of failed Pods are subtracted during the calculation of the node resources to be assigned.

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

Error	Cause	Solution
The domain name <code>receiver.barad.tencentyun.com</code> 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 <code>hostAlias</code> to the <code>tke-monitor-agent</code> DaemonSet as follows: <pre>hostAliases: - hostnames: - receiver.barad.tencentyun.com ip: 169.254.0.4</pre>

Log Collections

Last updated : 2023-03-27 11:08:16

Log Collection FAQs

Why can't I view the logs in CLS console after configuring log collection for the cluster?

If the log cannot be viewed or is missing, please check whether the following problems exist:

Check whether you have enabled the index feature for the selected log topic. Index configuration is required for using CLS for log search and analysis. If the index feature is disabled, you cannot view the logs. For more information about index configuration, see [Configuring Index](#).



Check whether the log, audit, and event features use the same topic. If they use the same topic, the logs will be overwritten, resulting in log missing.

Check whether the selected topic uses two extraction modes. The new extraction mode will overwrite the legacy one.

Check whether a soft link is used. If "Container File Path" is selected as the collection type, the corresponding path must not be a soft link. Otherwise, the actual path of the soft link will not exist in the container of the collector, resulting in a log collection failure.

If you use environment variables to enable TKE Serverless log collection and select the role-based authorization method, **you need to select CVM as the role entity when creating a role**. If you select **TKE**, the authorization fails.

If the problem persists, please [submit a ticket](#) to contact us.

Where can I view the logs after configuring the log rules?

1. Log in to the [CLS console](#). In the left sidebar, click **Search and Analysis**.
2. On the **Search and Analysis** page, select the region, log set, and log topic to view logs, and enable full-text index to search for and analyze logs, as shown below:



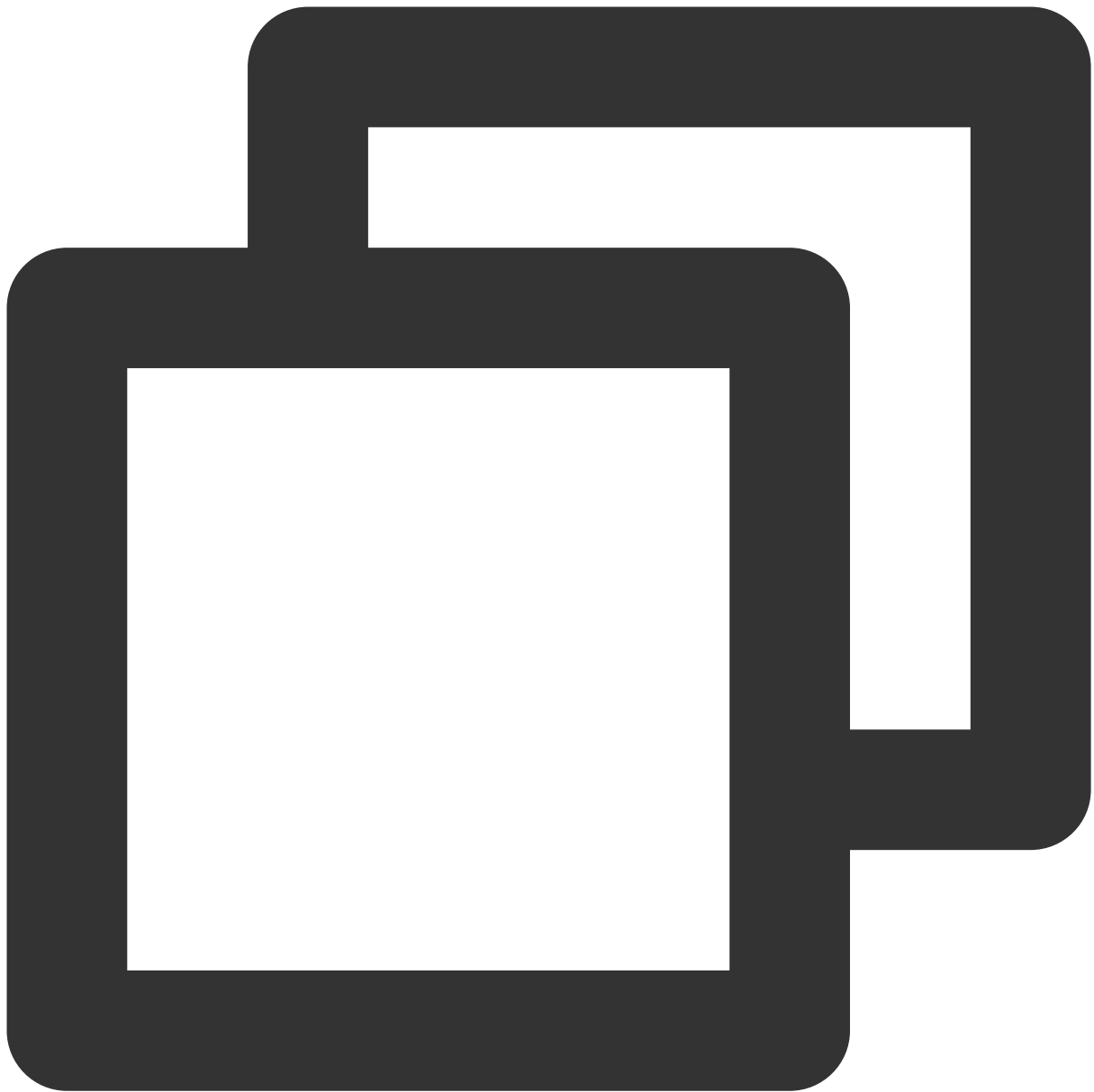
Using Environment Variables to Enable TKE Serverless Log Collection

How do Java applications implement multi-line log merging?

If the log data of an application such as a Java program occupies multiple lines, the line break `\\n` cannot be used to mark the end of a log. To enable CLS to clearly distinguish each log, you need to configure a configmap with the first-line regular expression. When a log in a line matches the preset regular expression, it is considered as the beginning of a log, and the next matching line will be the end mark of the log. For more information, see [Implementing Multi-line Log Merging for TKE Serverless Log Collection](#).

How to adjust the log collection configuration to adapt to different log output rates?

When you use environment variables to enable TKE Serverless log collection, TKE Serverless will start up a log collection component in the Pod sandbox to collect and report logs. Since TKE Serverless limits the memory usage of the log collection component, when the log output rate is too high, the log collection component may be out of memory (OOM). At this time, you can adjust the log collection configuration as needed. The specific method is to manually modify the following Pod annotation to reduce the memory buffer used by the log collection component. This helps reduce the memory usage.



```
internal.eks.tke.cloud.tencent.com/tail-buffer-chunk-size: "2M"  
internal.eks.tke.cloud.tencent.com/tail-buffer-max-size: "2M"
```

The descriptions of the annotations are shown in the following table. For more information, see [Fluent Bit](#).

Parameter	Description	Default Value
Buffer_Chunk_Size	It is used to set the initial buffer size to collect container file logs. It can also be used to increase the buffer size.	2M

Buffer_Max_Size	It is used to set the limit of the buffer size of each monitored file. If the buffer needs to be increased (for example, when long logs are stored), this value can limit the amount of buffer increased. If the buffer size of a file that is read exceeds this limit, the file will be deleted from the monitoring list.	2M
-----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----

What is the standard for outputting the logs of an application in the container of a TKE Serverless cluster?

The logs of an application in the container of a TKE Serverless cluster should be output to stdout. If the application logs are output to a container file, you need to rotate and clean up the log files regularly, or mount a volume for persistent storage. Otherwise, the 20 GB temporary storage will be used up. For more information, see [Pod Temporary Storage](#)

If you do not know how to clean up the log files, we recommend the following methods:

Mount a volume for persistent storage. For more information, see [Storage Management](#).

Enable TKE Serverless log collection. For more information, see [Using Environment Variables to Configure Log Collection](#).

TKE General Cluster

Auto-scaling related

Last updated : 2020-10-10 11:29:25

What is the difference between Cluster Autoscaler and node auto scaling based on monitoring metrics?

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 a CVM instance be added or removed in the AS 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
- Bandwidth billing method
- Whether to assign public IP

- Security group
- VPC
- Subnet

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?

Each Tencent Cloud user is provided with a quota of 30 pay-as-you go CVM instances in each availability zone. You can [submit a ticket](#) to apply for more instances for your auto scaling group.

For more information about the quotas, see [CVM Instance Quantity and Quota](#) for your current availability zone. In addition, there is a maximum limit of 200 instances for Auto Scaling. You can [submit a ticket](#) to apply for a higher quota.

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 you set strict PodDisruptionBudget for a pod, and the PDB is not met, it will not be scaled in.
- Pods under Kube-system.
- Pods on a node that are not created by controllers such as Deployment, ReplicaSet, Job, or StatefulSet.
- 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.
- Whether the account balance is sufficient (scale-out cannot be triggered if the account balance is insufficient) or the quota is insufficient. For more information, see [other reasons](#).

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

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

Where can I find details on the scaling events?

You can query the scaling events of an auto scaling group and view K8s events in the AS Console. Events can be found in the following three resources:

- kube-system/cluster-autoscaler-status config map
 - **ScaledUpGroup** - CA triggers scale-out.
 - **ScaleDownEmpty** - CA deletes a node with no running pod.
 - **ScaleDown** - CA triggers scale-in.
- node
 - **ScaleDown** - CA triggers scale-in.
 - **ScaleDownFailed** - CA failed to trigger scale-in.
- pod
 - **TriggeredScaleUp** - CA triggers scale-out because of this pod.
 - **NotTriggerScaleUp** - CA cannot find an available auto scaling group to schedule this pod.
 - **ScaleDown** - CA tries to drain this pod to scale in the node.

Cluster FAQ

Last updated : 2022-12-08 18:03:06

Cluster Creation

Can I choose not to set a public IP address for the CVM when creating a cluster?

Yes. A CVM instance without a public IP address can only pull images from "My Images" in Image Repository, but cannot pull images from Docker Hub or third-party platforms.

A CVM without a public IP address but with Internet bandwidth can access the Internet by binding an EIP.

Why do I need to choose a network when creating a cluster?

The selected network and subnet are where the CVM resides. You can add different CVMs to subnets in different AZs for cross-AZ disaster recovery.

What CVM models are supported when creating a cluster?

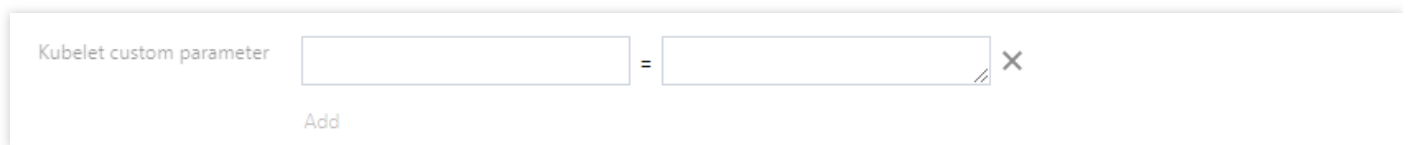
We offer Standard, Computing, High-IO, GPU, and BM CVM instances, as displayed in the TKE console.

What operating systems are supported for TKE hosts?

For more information on the currently operating systems supported by TKE, see [Public Image List](#)

How do I customize Kubelet parameters for a TKE node?

- This feature is made available through an allowlist. To use it, [submit a ticket](#) for application.
- Customize kubelet parameters of a node on the [Add Node](#) page, [Add Existing Node](#) page, or [Create Node Pool](#) page.



The screenshot shows a user interface for adding custom kubelet parameters. It features a text input field with the placeholder text "Kubelet custom parameter". To the right of the input field is an equals sign followed by another text input field, which has a small icon in its bottom right corner. To the right of this second input field is a close button (an 'X' icon). Below the first input field is a button labeled "Add".

How do I customize Kubernetes component parameters when creating a TKE cluster?

For detailed directions, see [Custom Kubernetes Component Launch Parameters](#).

Does TKE support self-deployed clusters?

TKE provides an independent Master deployment mode in which you have full control over your cluster. In this mode, the Master and Etcd of the Kubernetes cluster are deployed on the CVM instances you purchased, and you have all

the management and operation permissions for the Kubernetes cluster. For more information, see [Independent Master Deployment Mode](#).

Can I modify the AZ after a TKE serverless cluster is created?

After a TKE serverless cluster is created successfully, you cannot change or add AZs.

Can I get IPv6 addresses of TKE clusters on the client?

No.

Can I export the TKE cluster configuration?

No.

When a TKE cluster is created, the system prompts me to complete node exception detection plus parameter settings. What should I do?

If you are prompted to complete node exception detection plus parameter settings when creating a cluster, you need to check whether the `NodeProblemDetectorPlus` add-on is selected and configure the parameters if so. For more information, see [NodeProblemDetectorPlus Add-on](#).

Note :

If you are unable to evaluate whether you need this add-on, we recommend you create the cluster directly without selecting it. If you need to use it later, simply install it in add-on management.

How do I enable IPVS for TKE clusters?

For detailed directions, see [Enabling IPVS for a Cluster](#).

How do I view TKE cluster access credentials?

1. Log in to the [TKE console](#) and click **Cluster** in the left sidebar to enter the cluster management page.
2. Click the ID or name of the target cluster to enter the cluster details page.
3. Click **Basic information** in the left sidebar. You can view information such as the access address, public network/private network access status, and kubeconfig access credential of the cluster in the **Cluster APIServer information** section of the **Basic information** page.

How do I add CVM instances under other accounts to a TKE cluster?

Currently, you cannot add CVM instances under other accounts to a cluster. You can only add CVM instances in the same VPC.

What should I do if "certificate verify failed: self signed certificate" is reported when I use the Kubernetes SDK to connect the API server of the cluster?

As the API server in a Kubernetes cluster usually uses a self-signed TLS certificate, which is not trusted by the request library on an earlier Python version or the default settings of Node.js, an error will be reported. This error can be fixed in the following steps:

1. Add the configuration to skip server authentication.

If you use the Node.js SDK, you need to set the environment variable:

```
export NODE_TLS_REJECT_UNAUTHORIZED="0";
```

If you use the Python SDK, you need to set it in `kubeconfig`.

```
clusters:  
- cluster:  
  insecure-skip-tls-verify: true
```

2. Put the public key of the root certificate of the self-signed certificate issuing authority in the list of trusted root certificates of the system.

Cluster Network

What should I do for the cluster network when I develop a custom webhook?

When you develop a custom webhook, do not block Pods under `kube-system namespace`; otherwise, you cannot use the cluster network normally.

Adding CVM Instances

What are the limitations on adding CVMs to a cluster?

You can only choose the CVMs in the same region as the cluster. But you may choose a different AZ to implement cross-AZ deployment of the cluster.

Is there a limit on the number of CVMs?

Yes. The number of pay-as-you-go CVMs cannot exceed the usage quota of the current account. For details, see [CVM Overview](#).

CVM Termination

After a CVM instance is terminated, what will happen to the containers deployed on it?

When a CVM instance is terminated, the resources it contains, such as containers, will also be terminated. If the number of containers for a service drops below the expected number of running containers, the cluster will launch more containers in other CVM instances until the desired number is reached.

How to Choose Containerd and Docker

Last updated : 2023-08-10 17:23:21

How to Choose Containerd and Docker

How do I select a runtime component?

Note :

Select Containerd for the runtime when creating a node in a Kubernetes 1.24 cluster. Images built with Docker can still be used.

The containerd is a more stable runtime component. It supports OCI standard and does not support docker API.

As one of the most important components of Kubernetes (K8s), a container runtime manages the lifecycle of images and containers. Kubelet interacts with the containerd through the `Container Runtime Interface (CRI)` to manage images and containers.

You can choose Containerd and Docker as container runtime components:

- (Recommended) Containerd has a shorter calling chain and fewer components, and features higher stability and lower node resource consumption.
- Docker should be used as the runtime component in the following situations:
 - You need to use docker in docker;
 - You need to use commands such as docker build/push/save/load in the TKE node;
 - You need to call the docker API;
 - You need the docker compose or docker swarm.

How do I modify a runtime component?

1. Log in to the [TKE console](#) and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page, click the target cluster ID to enter the cluster basic information page.
3. Modify the runtime component in the **Basic information** section. See the figure below:

Note :

Modifications on the runtime component and version only take effect for added nodes that are not assigned to any node pool. The existing nodes are not affected.

Runtime components ⓘ



What are the commands commonly used in Containerd and Docker?

Containerd does not support docker API or docker CLI. However, you can get these features with cri-tool commands.

Image related commands

Image Feature	Docker	Containerd
Display the local image list	docker images	crictl images
Download an image	docker pull	crictl pull
Upload an image	docker push	None
Delete a local image	docker rmi	crictl rmi
View image details	docker inspect IMAGE-ID	crictl inspect IMAGE-ID

Container related commands

Container Feature	Docker	containerd
Display the container list	docker ps	crictl ps
Create a container	docker create	crictl create
Start a container	docker start	crictl start
Stop a container	docker stop	crictl stop
Delete a container	docker rm	crictl rm
View container details	docker inspect	crictl inspect
attach	docker attach	crictl attach
exec	docker exec	crictl exec
logs	docker logs	crictl logs

Container Feature	Docker	containerd
stats	docker stats	crictl stats

POD related commands

Pod Feature	Docker	Containerd
Display the pod list	None	crictl pods
View pod Details	None	crictl inspectp
Run a pod	None	crictl runp
Stop a pod	None	crictl stopp

What are the differences between the calling chains of Containerd and Docker?

- When Docker is used as the K8s container runtime, the calling chain is as follows:

```
kubelet --> docker shim (in the kubelet process) --> dockerd --> containerd
```

- When Containerd is used as the K8s container runtime, the calling chain is as follows:

```
kubelet --> cri plugin (in the containerd process) --> containerd
```

Although Docker offers more features such as swarm cluster, docker build, and docker API, it may also introduce some bugs and requires one more calling step than Containerd, including exec, preStop, ipv6, and log stdout formats.

Stream Service Related Differences

Commands such as `kubectl exec` and `kubectl logs` require Kubelet to establish a stream forwarding channel between the apiserver and container runtime.

How does a stream service work?

You can learn how CRI's stream service works by understanding how the 'kubectl exec' command works:

- Dockershim or Containerd listens on a port to run a stream service when it is started.
- When a command such as `kubectl exec` is run, the request finds the node corresponding to Pod through kube-apiserver and forwards it to Kubelet.
- Kubelet requests the GetExec API of CRI-Service (dockershim or containerd-cri). The CR-Server generates a random token for the request, combines the token and the CRI Stream server listening port to form a URL, and returns it to Kubelet.

4. Kubelet upgrades the HTTP request sent by kube-apiserver to WebSocket and works as the proxy between the apiserver and CRI Stream server to forward data.

How do I use and configure stream services in Containerd?

The docker API itself provides a stream service, and the docker-shim inside the Kubelet has its default configuration as `127.0.0.1:0`.

The stream service of Containerd needs to be configured separately:

```
[plugins.cri]
stream_server_address = "127.0.0.1"
stream_server_port = "0"
enable_tls_streaming = false
```

What are the configuration differences between versions before and after K8s v1.11?

The stream service of Containerd has different configurations for different versions of K8s.

- Before K8s v1.11:
Kubelet performs redirection but not stream proxying. That is, Kubelet sends the stream server address opened by containerd to the apiserver which then directly accesses the stream service of containerd. You need to authenticate the stream service forwarder for security purposes.
- After K8s v1.11:
K8s v1.11 introduced [kubelet stream proxy](#), so that the stream service of containerd only needs to listen to the local address.

Container Exec Differences

Docker and Containerd differ slightly in the implementation of Exec, mainly in the implementation of Execsync, which is the execution of a single command.

Therefore, the `kubectl exec` command without specifying the `-it` parameter and the ExecProbe in `pod lifecycle` may behave slightly differently on a node of a different runtime type.

kubectl exec

- Docker exec: **The end of the first process of the current** `exec` marks the end of `exec`.
- CRI exec: **The end of all processes of the current** `exec` marks the end of `exec`.

Take the command `kubectl exec <pod-id> -- bash -c "nohup sleep 10 &"` as an example. On a runtime Docker node, the command will end about two seconds after execution, while on a Containerd node, the

command will not end until the sleep process exits.

pod exec probe

Kubelet uses the ExecSync API of CRI Runtime when implementing `exec probe`. Therefore, `exec probe` works the same as `kubect1 exec # no -t -i :`

- On a Docker node, `exec probe` will still exit normally even if there are residual sub-processes.
- On a Containerd node, `exec probe` will not end until all probe processes exit.

The main impact of this difference will be `postStartHook` and `preStopHook` of `pod lifecycle`. If `exec probe` is used in hooks and residual sub-processes occur, you may encounter the case where Pods are stuck in the `containerCreating` state for a long time on Containerd nodes. The reason is that Kubelet will pull up containers one by one during `syncPod` and execute probe. If a probe is blocked due to the above reason, subsequent containers will fail to start.

Pulling up child processes and exiting the parent process in ExecProbe is a behavior not defined in K8s. The specific actions may vary depending on the runtime version and type. Therefore, it is recommended not to perform overly complex operations on probe.

Container Network Differences

Under normal circumstances, containers in a Pod share the same network namespace, so the Pod needs to have the network ready when the sandbox container is created. To better illustrate the differences, the following briefly introduces the Pod network initialization process:

1. Kubelet calls the CRI runtime (dockershim or containerd) to create a sandbox container for the Pod.
2. The CRI runtime calls the underlying Docker or Containerd to create a pause container (now, the pause process is not started yet but the network namespace has been initialized).
3. The CRI runtime calls CNI to perform network initialization operations such as creating a Veth ENI in the network namespace and adding the cbr0 network bridge.
4. The CRI runtime starts the pause container, and the pause process is started.
5. Kubelet continues to call the CRI runtime to perform subsequent operations such as container creation.

Docker and Containerd act differently in the following two steps: **creating a pause container and initializing the network namespace** and **calling CNI to initialize the Veth ENI**.

Creating a Pause container

Containerd is a CRI runtime designed for Kubernetes and has no separate network module, while Docker is designed with its own network functionalities. When Docker creates a pause container, it implements Docker-specific network

setting. This setting causes the biggest difference between Docker and Containerd: If IPv6 is not to be used, Docker will set the kernel parameter `net.ipv6.conf.all.disable_ipv6` in the container's network namespace to `1` to disable the `ipv6` option in the container.

For the same Pod, only IPv4 is enabled on the Docker node while both IPv4 and IPv6 are enabled on the Containerd node.

If both IPv4 and IPv6 are enabled, DNS may send both IPv4 and IPv6 packets at the same time. In some cases, a business that requires frequent DNS resolution may trigger a bug of the DNS resolution library (depending on the Pod service implementation dependencies). On the Containerd node, you can disable IPv6 setting for a Pod by adding an init container to the Pod. The code is as follows:

```
apiVersion: v1
kind: Pod
...
spec:
  initContainers:
  - image: busybox
    name: sysctl
    command: ["sysctl", "-w", "net.ipv6.conf.all.disable_ipv6=1"]
    securityContext:
      privileged: true
  ...
```

Calling CNI

There is no real difference between Containerd and Docker in calling CNI.

Item	Docker	containerd
Component responsible for calling CNI	docker-shim inside Kubelet	Containerd's built-in cri-plugin
CNI configuration	Kubelet parameters <code>--cni-bin-dir</code> and <code>--cni-conf-dir</code>	containerd configuration file (toml): <pre>[plugins.cri.cni] bin_dir = "/opt/cni/bin" conf_dir = "/etc/cni/net.d"</pre>

Container Log Differences

Item	Docker	containerd
Storage path	<p>If Docker serves as the K8s container runtime, it saves container logs to a directory such as <code>/var/lib/docker/containers/\$CONTAINERID</code>. Kubelet will create a soft link under <code>/var/log/pods</code> and <code>/var/log/containers</code>, pointing to the container log files in the <code>/var/lib/docker/containers/\$CONTAINERID</code> directory.</p>	<p>If Containerd serves as the K8s container runtime, Kubelet saves container logs to the <code>/var/log/pods/\$CONTAINERID</code> directory, and creates a soft link under <code>/var/log/containers</code>, pointing to the log files.</p>
Storage size	<p>For each container in a Pod, Docker retains 1 GB (100 MB x 10 = 1 GB) of logs by default.</p>	<p>For each container in a Pod, Containerd retains 50 MB (10 MB x 5 = 50 MB) by default.</p>
Configuration parameters	<p>Specify in the Docker configuration files:</p> <pre>"log-driver": "json-file", "log-opts": {"max-size": "100m", "max-file": "5"}</pre>	<ul style="list-style-type: none"> Method 1: Specify in the Kubelet parameters: <pre>--container-log-max-file --container-log-max-size="100Mi"</pre> Method 2: Specify in KubeletConfiguration: <pre>"containerLogMaxSize": "100Mi", "containerLogMaxFiles":</pre>
Save container logs to the data disk	<p>Mount the data disk to "data-root" (<code>/var/lib/docker</code> by default).</p>	<p>Create a soft link <code>/var/log/pods</code> point to a directory under the data disk mounting point. Selecting "Store containers and images on the data disk" in TKE will automatically create the soft link <code>/var/log/pods</code>.</p>

External DNS Causes Abnormal Node Initialization

Last updated : 2022-05-09 11:40:29

Background

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 relevant components in the cluster depends on the Tencent Cloud resource library at:

```
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 self-built DNS server and the request source are not in the same region, some Tencent domain names that don't support cross-region access may become invalid.

About Services

Last updated : 2022-06-10 16:48:46

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 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 instance. You can create a CLB instance, configure the security group, and then mount it to the service. For more information, see [Using Existing CLBs](#).
- You can configure the security group through `TkeServiceConfig` in the service. A CLB instance will use a security group according to the configuration at the time of creation. If you need to use this feature, [submit a ticket](#) for application.

Note :

To avoid access failure, we recommend you not access a service in the cluster over the CLB IP.

In general, a layer-4 CLB instance will bind multiple nodes as real servers (RS). In this case, make sure that the client and RS are not on the same CVM instance; otherwise, there will be a certain probability that the packet will fail to loop back.

When a Pod accesses a CLB instance, the Pod is the source IP. When it is transferred to the private network, the CLB instance will not translate the source IP to the Node IP via SNAT. Therefore, the CLB instance cannot identify the source node of the packet. The CLB instance's loopback avoidance policy will not take effect, and the packet may be forwarded to any RS. When the packet is forwarded to the Node where the client is located, the CLB instance will be unable to receive the response, leading to access failure.

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 from the public network to a VPC, from a VPC to the public network, or between different subnets of a VPC. For more information, see [Overview](#).

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 eight hours difference between the container system time and UTC+8 time. You can create a time zone file in `dockerfile` to solve this problem. For more information, see [Solve the inconsistent time zone problem in the container](#).

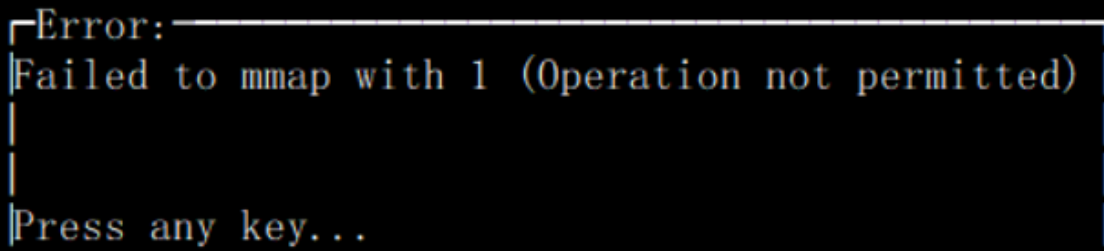
What should I do when some Docker Hub images, such as Ubuntu, PHP, and BusyBox, encounter exceptions in TKE?

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 a container is executing `perf top -p` to check the process CPU status, an error message "Operation not permitted" appears as shown below:



```
Error:
Failed to mmap with 1 (Operation not permitted)

Press any key...
```

The default configuration file of Docker prevents important system calls. `perf_event_open` is disabled because it may leak a large amount of information on the host. If you need to call it, configure a privileged container or change the value of the Pod YAML field `privileged` to `true`. You need to assess the security risks yourself.

TKE Serverless Cluster

TKE Serverless Clusters-related

Last updated : 2023-08-10 18:05:33

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 do I create or modify the container network of a 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**.

Basic information

Region
Cluster ID
Kubernetes version
Cluster network

Super node

Node pool
Node pool name
OS type

The name cannot exceed 25 characters. It only supports Chinese characters, English letters, numbers, underscores, hyphens ("-") and dots.

When Windows OS is selected, Windows containers are supported in this subnet. Linux OS is selected by default.

From January 31, 2023 (UTC +8), discounts on the official site of Tencent Cloud that apply to pay-as-you-go Pods of small sizes on a s

Super node configuration

Availability zone
Billing mode
Container network
Node name

Guangzhou Zone 3

Pay-as-you-go

Subnet ID	Subnet name	Availability zo...	Remaining IPs

The Pod will occupy the IPs of selected subnets. Please select subnets with sufficient IPs and not conflict with other serv suitable, please go to the console to [create subnet](#).

Auto-generated

Confirm Cancel

Add node

Create super node Cancel

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

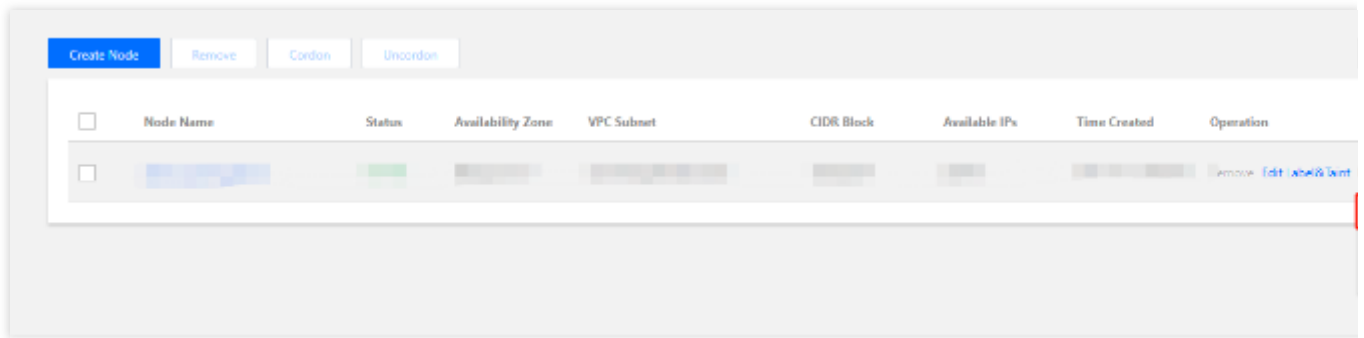
Note

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. On the **Super node** page, click **Remove** on the right of the node name.

6. 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.

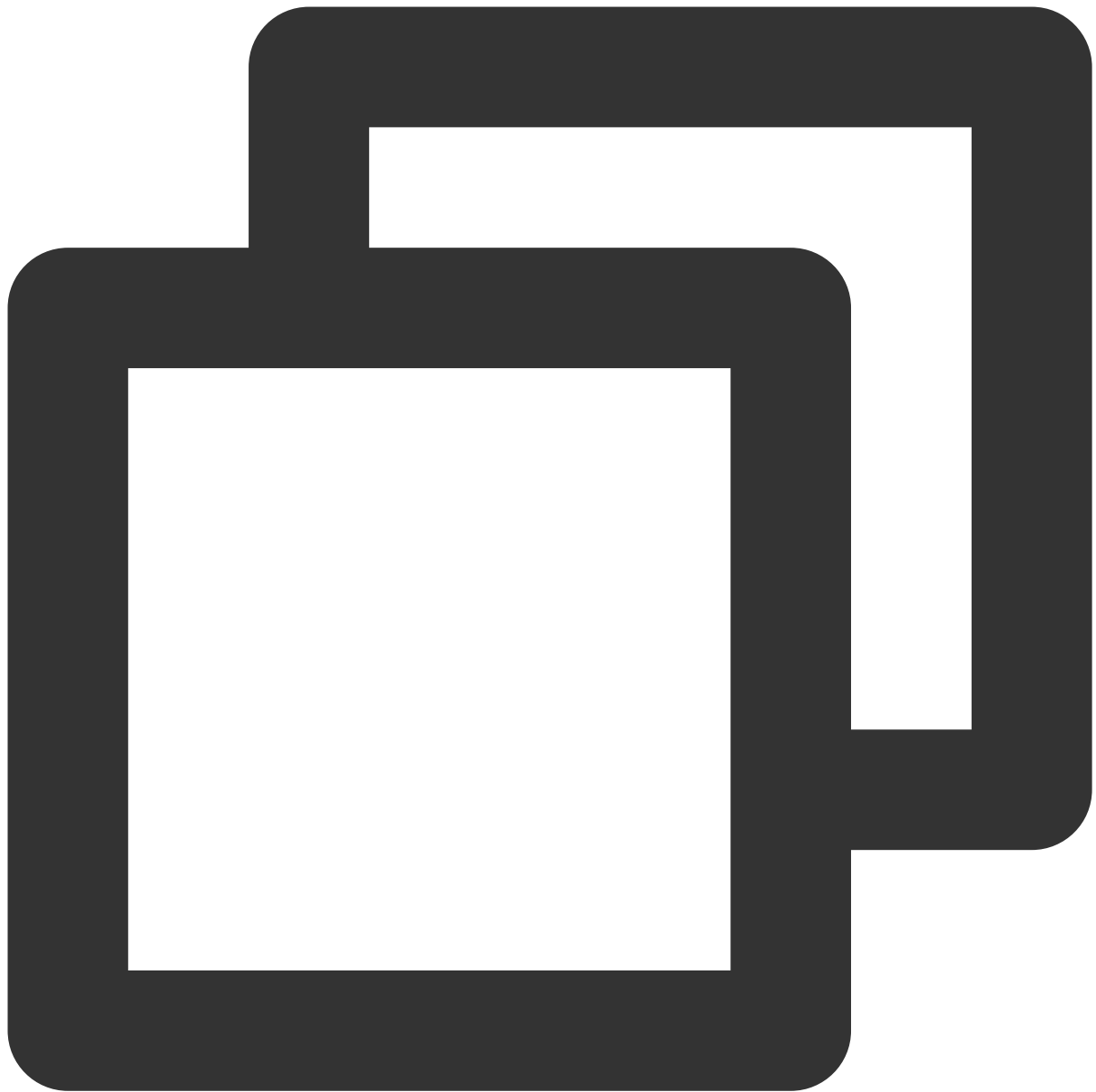
Event 1:

2021-04-20 15:16:01	2021-04-20 15:35:17	Warning	Pod	[redacted]	FailedScheduling
2021-04-20 15:19:11	2021-04-20 15:35:17	Warning	Pod	[redacted]	FailedScheduling

Event 2:

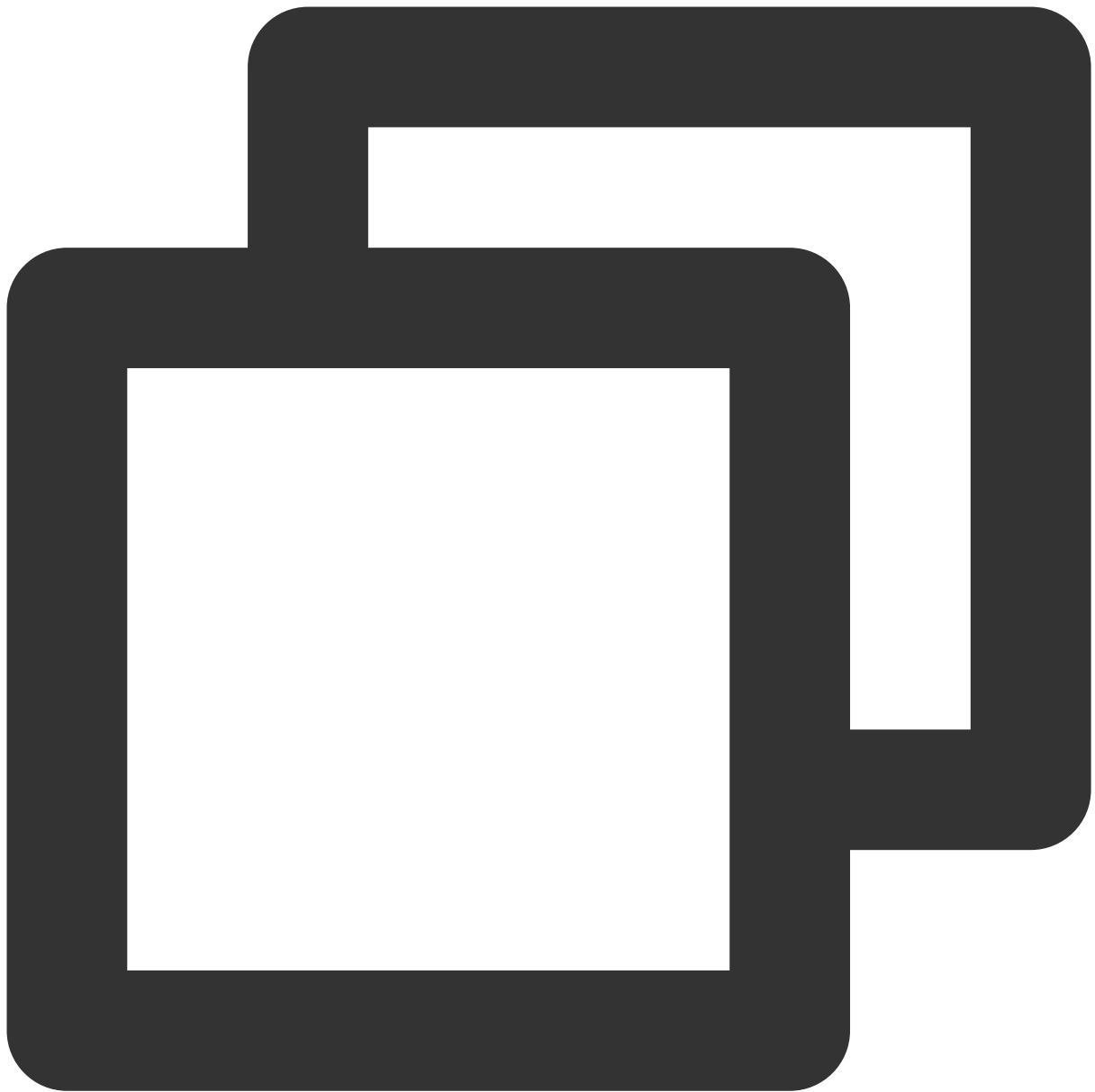
2021-04-20 15:00:10	2021-04-20 15:00:10	Warning	Pod	[redacted]	FailedCreatePodSandB
2021-04-20 15:00:07	2021-04-20 15:00:07	Warning	Pod	[redacted]	FailedCreatePodSanc

You can query the YAML of the super node in the [TKE console]](<https://consoleintl.cloud.tencent.com/tke2/ecluster?rid=1!991ea712f9f39b63beaafb2cdb56ba5f>) or by running the following command in the command line tool:

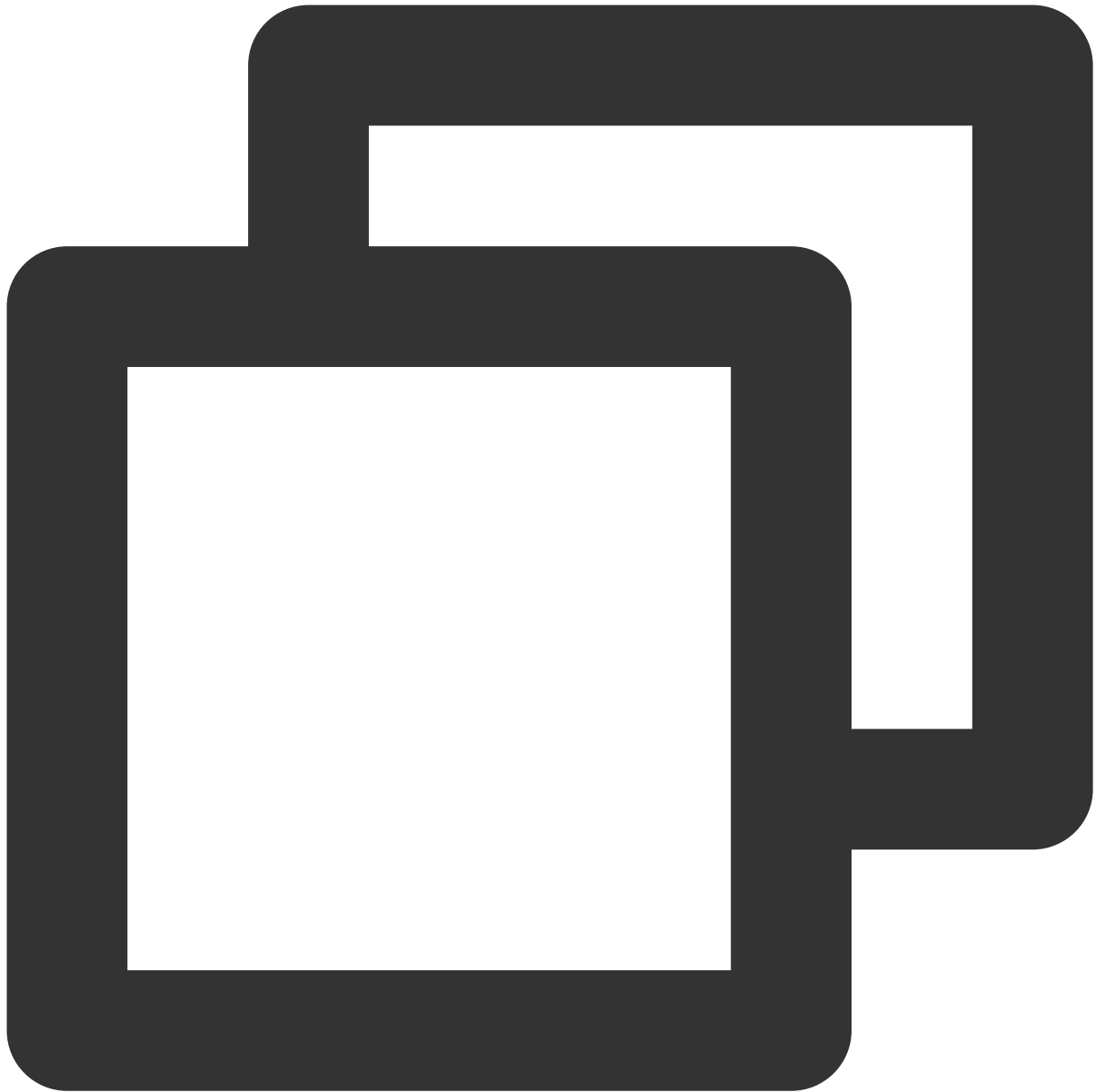


```
kubectl get nodes -oyaml
```

The following information will appear:



```
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 a container termination message?

Kubernetes can set the message source of the container exit through `terminationMessagePath`, that is, when the container exits, Kubernetes will retrieve termination messages from the termination message file specified in the `terminationMessagePath` field of a container, and use this contents from the specified file to populate the container's termination message. The default value of the message is `/dev/termination-log`.

Moreover, you can set the `terminationMessagePolicy` field of a container for further termination message customization. This field defaults to "File", which means the termination messages are retrieved only from the termination message file. You can set the `terminationMessagePolicy` to `FallbackToLogsOnError` as needed, and Kubernetes will use the last chunk of container log output if the termination message file is empty and the container exited with an error.

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 do I use Host parameters?

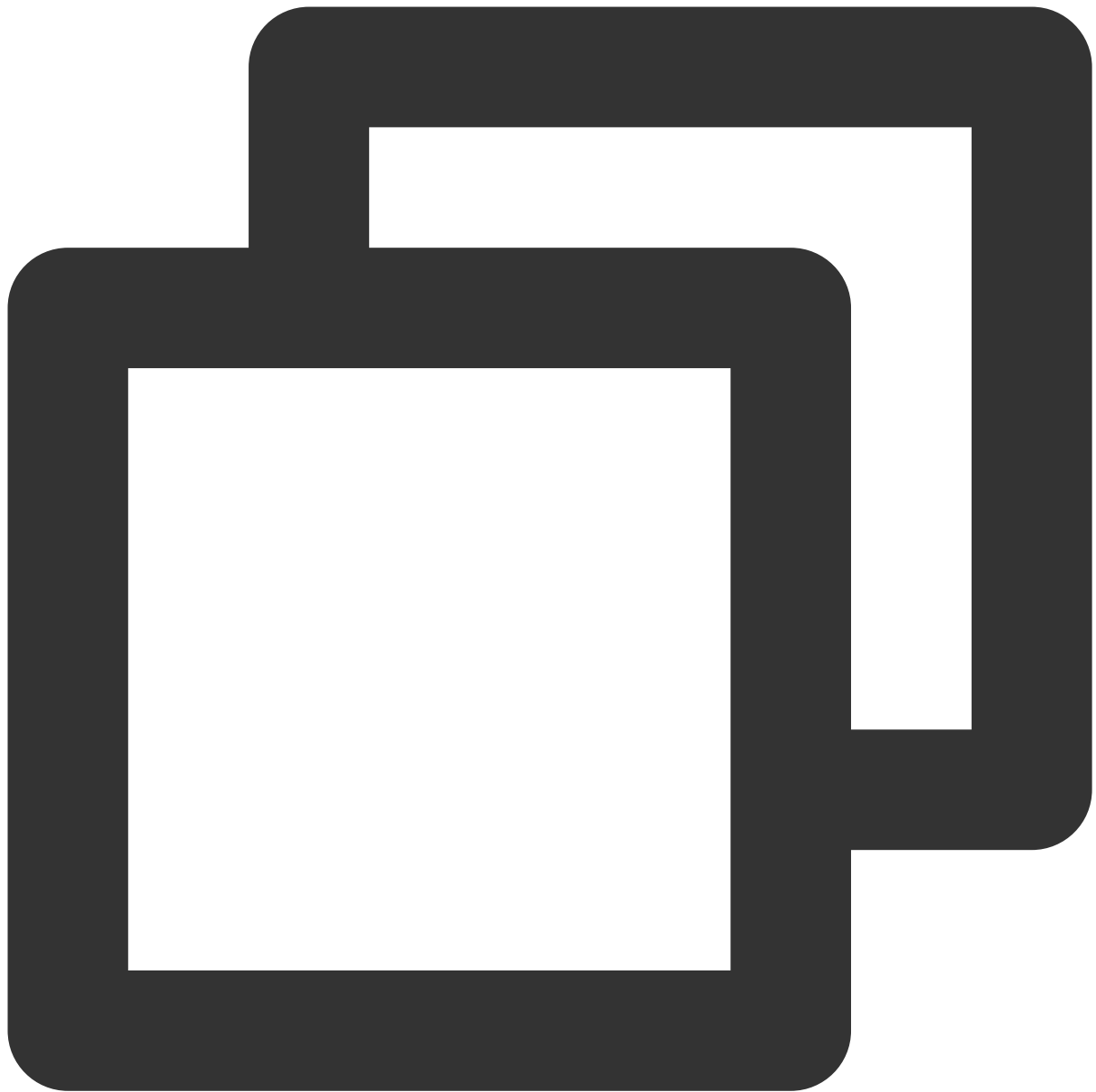
Note the following when using TKE Serverless clusters:

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.

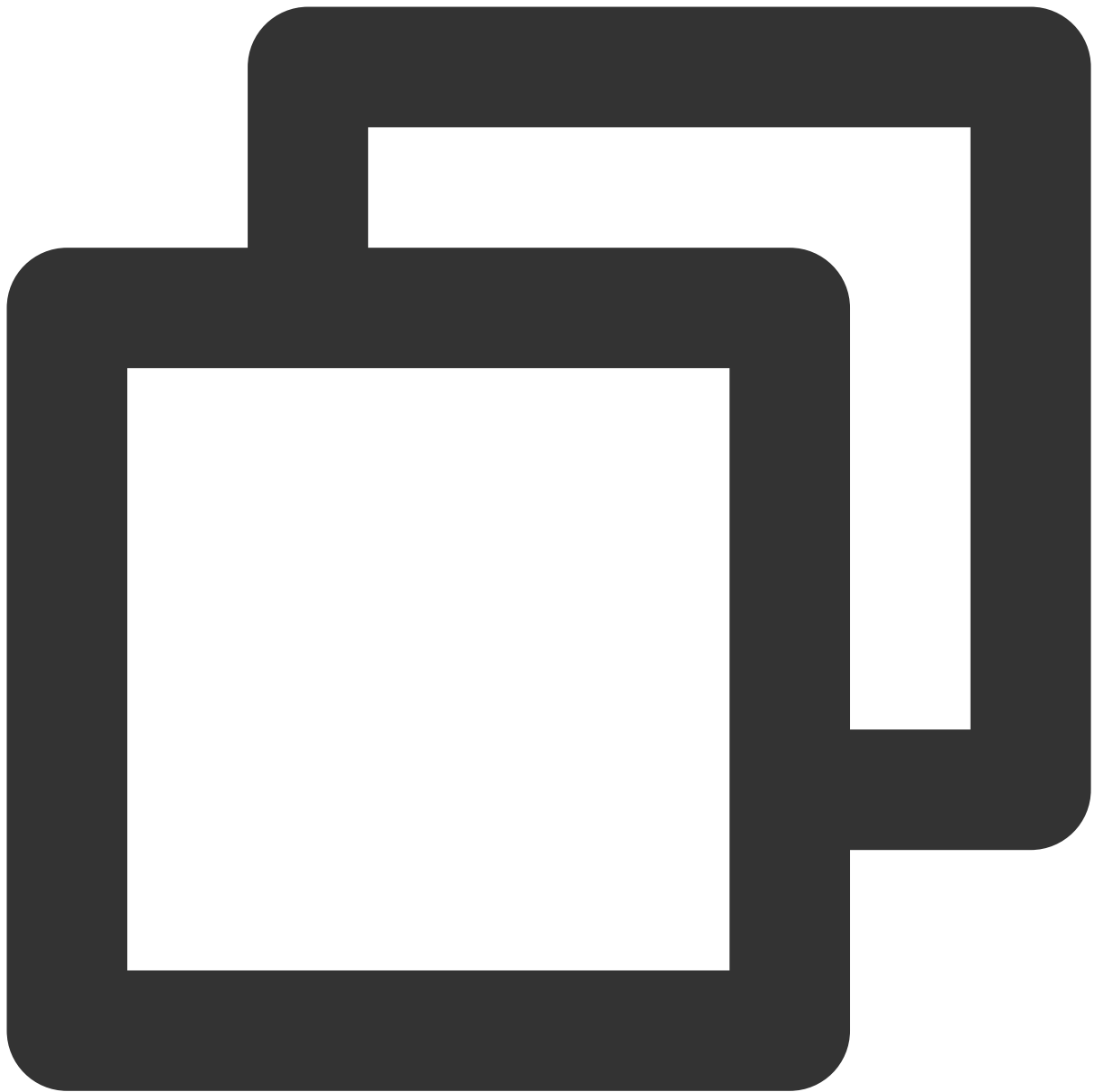
For detailed directions of mounting a volume to a Pod, see [Instructions for Other Storage Volumes](#).

How do I speed up container startup 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 Pods with the same annotation value under the same user account, the start-up image will be reused within the cache time as much as possible. We recommend you 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
```

Methods for Resume:

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

Cause:

```
no space left on device: unknown
```

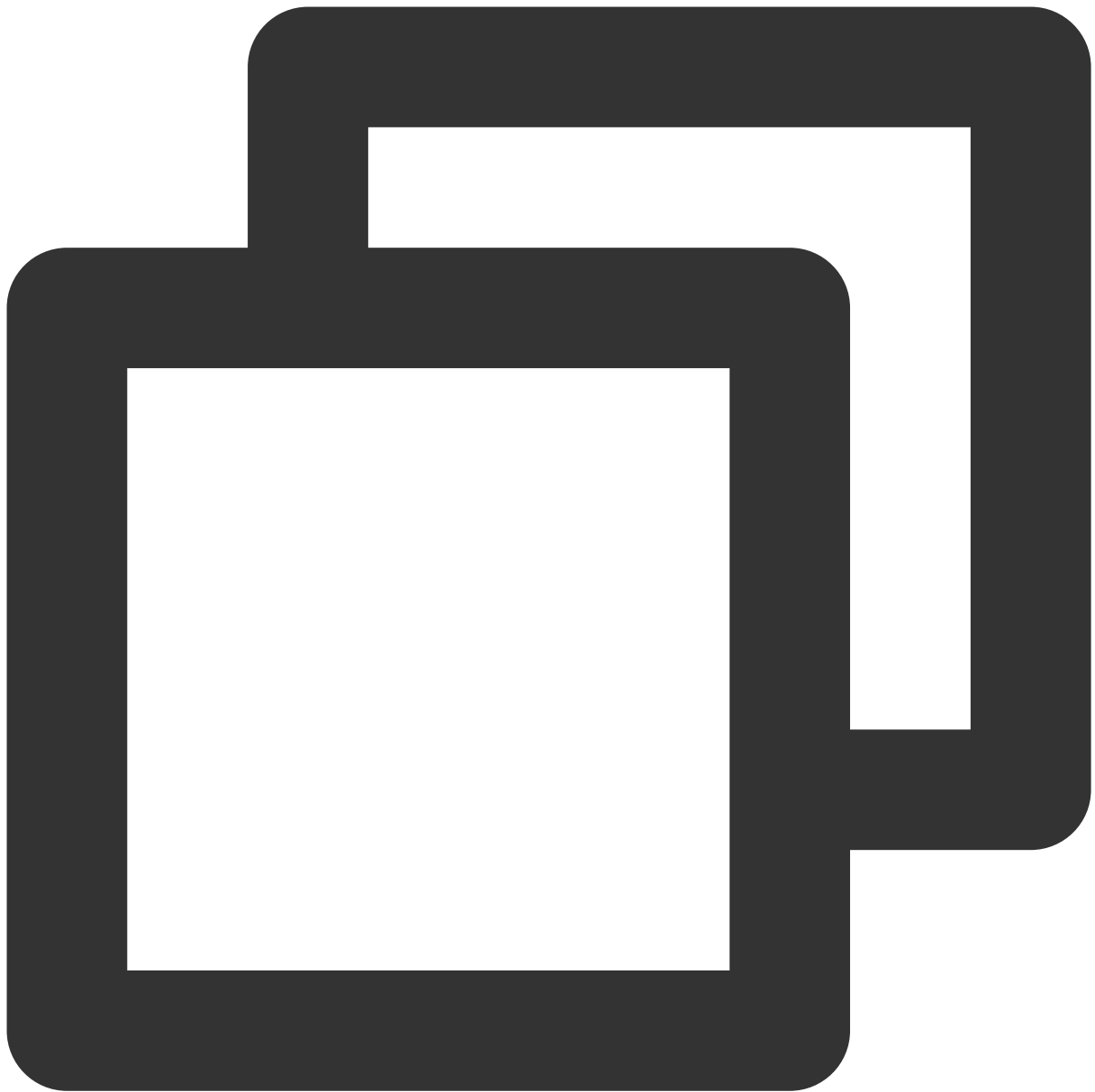
When the Pod reuses the system disk by default, the original image in the system disk occupies all of the space, and the disk does not have enough space to download the new image, so the error "no space left on device: unknown" is reported. TKE Serverless supports the regular image repossession mechanism. When the whole space is occupied, the mechanism will automatically delete the existing redundant images in the system disk to free up the current disk. The process takes 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 shows that the current Pod needs 4220828057 bytes to download the image, but currently only 3889267064 bytes are available. The cause is that there are multiple images on the disk and only some images have been freed up. The regular image repossession mechanism of TKE Serverless will continue to free up the disk until a new image can be successfully pulled.

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

If you use an external NFS for persistent storage, the event `Operation not permitted` will be reported when a connection is made. You need to modify the `/etc/exports` file of your NFS and add the `/<path><ip-range>(rw,insecure)` parameter. See the example 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

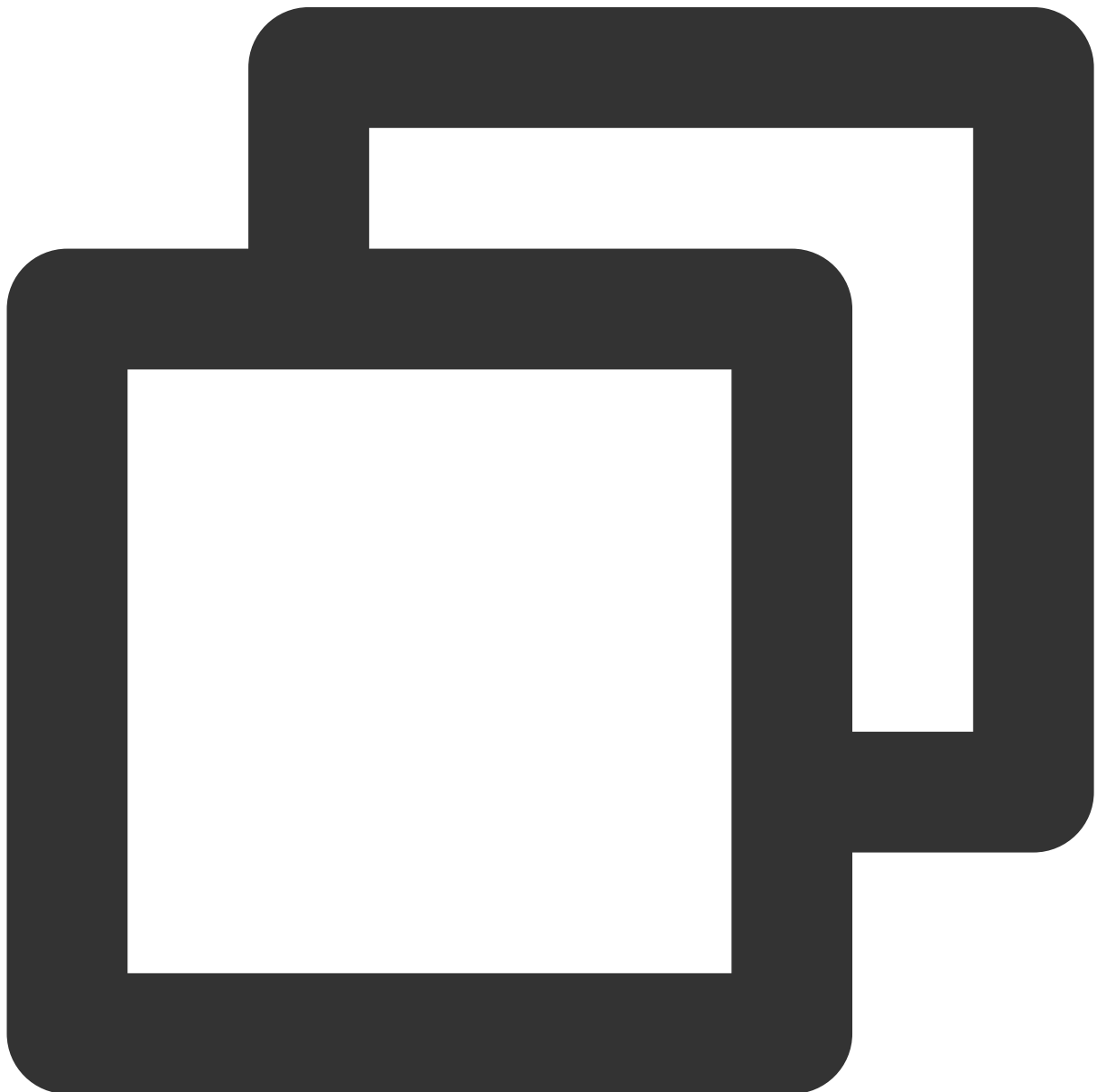
If 80% of the space is used, the TKE Serverless backend will trigger the container image repossession process to recover the unused images and free up the space. If this process fails, the `ImageGCFailed: failed to garbage collect required amount of images` message will be reported to remind you of the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

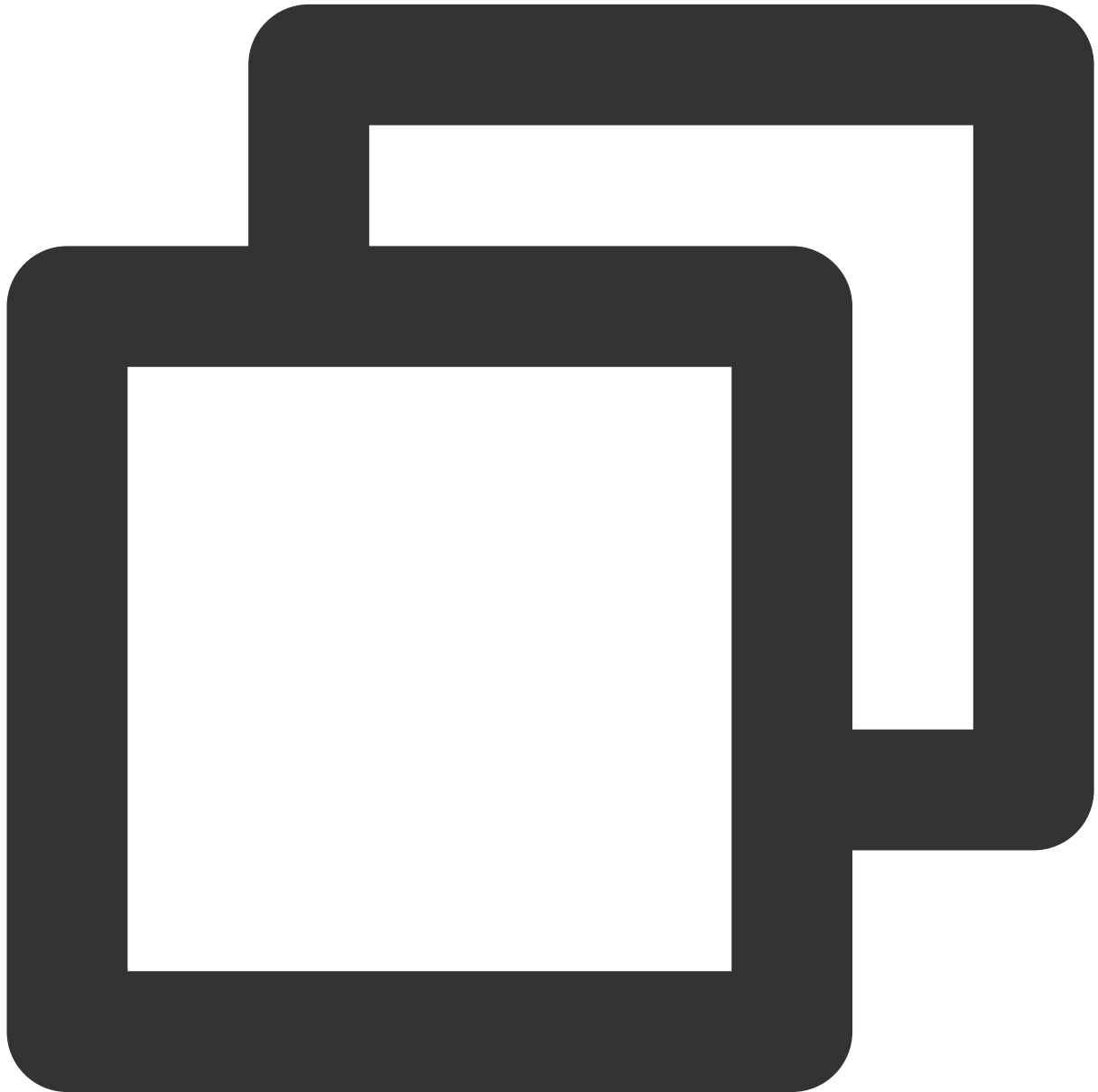
If you want to adjust the threshold for the container image repossession, set the following annotation:



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

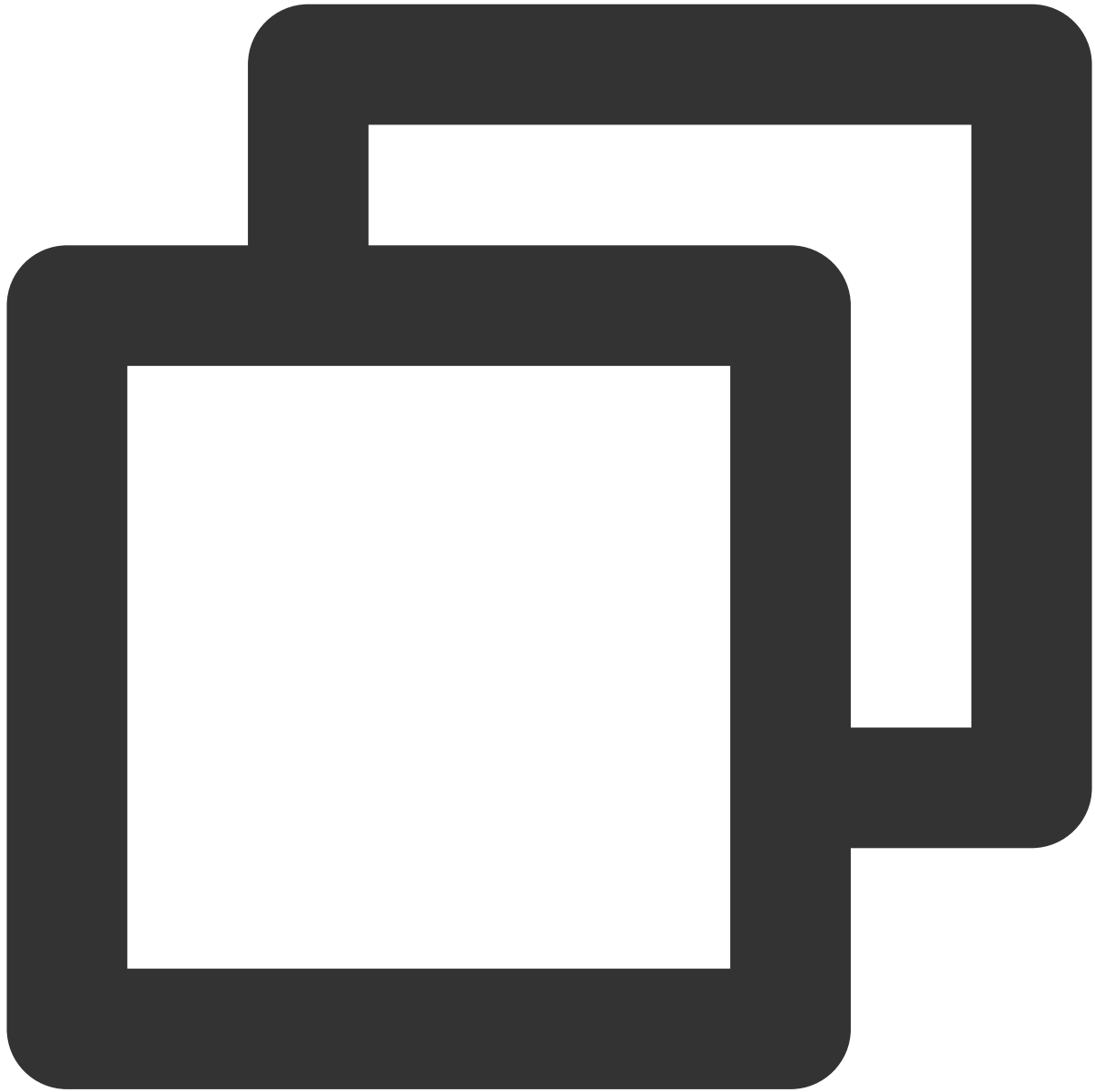
2. Clean up exited containers

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 do not 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.



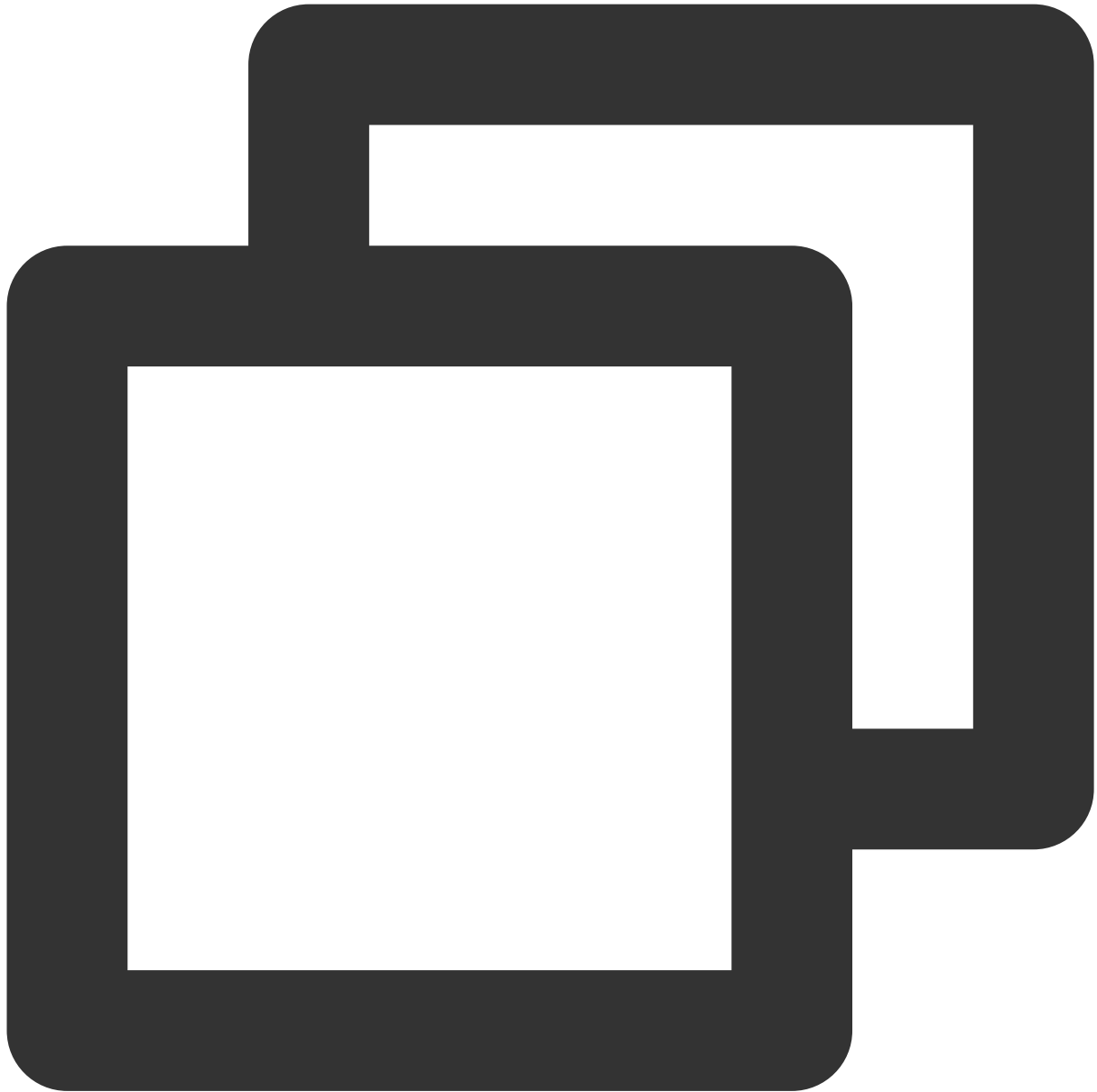
```
eks.tke.cloud.tencent.com/must-keep-last-container: "true"
```

Description

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.

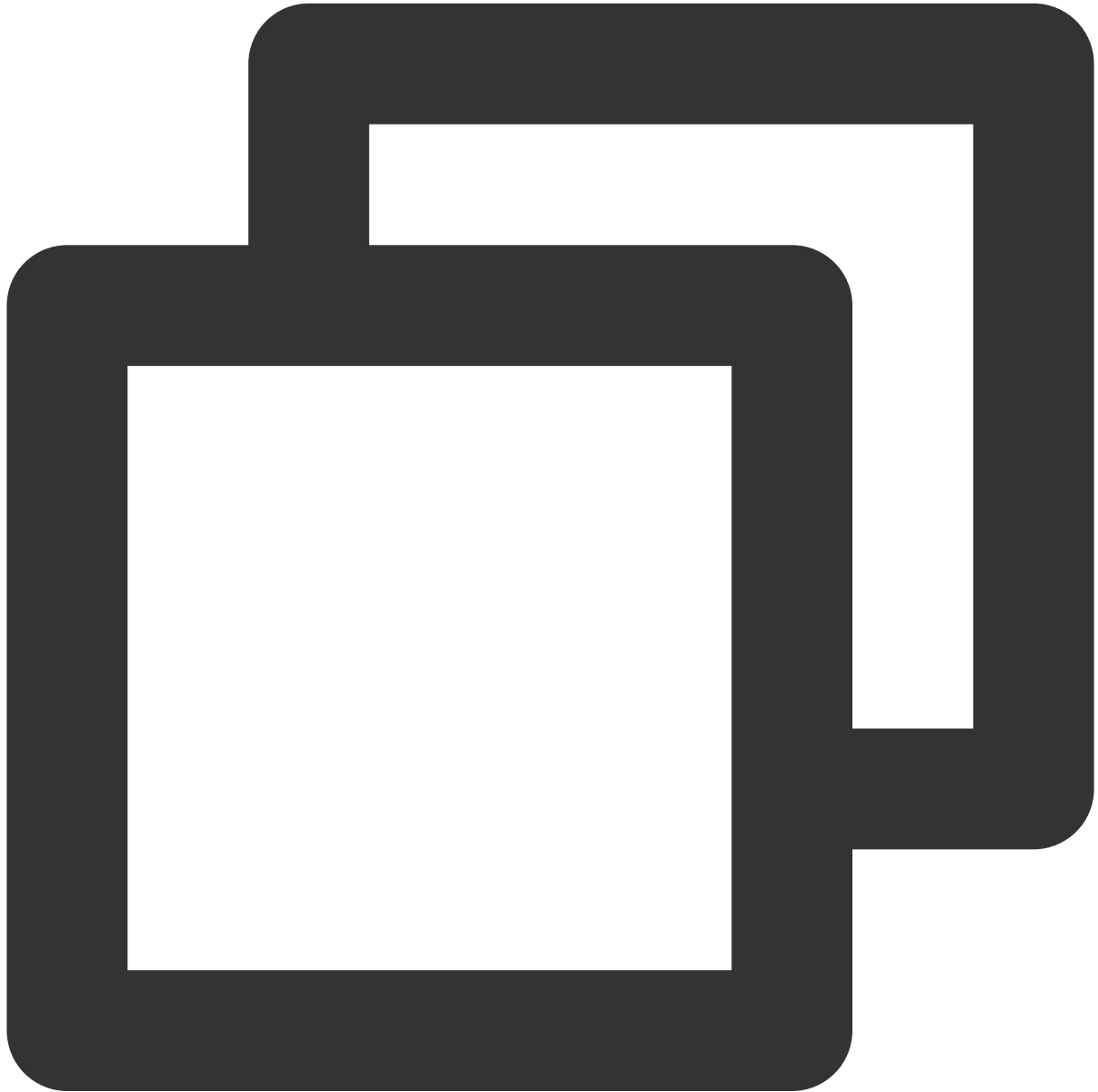
Description

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

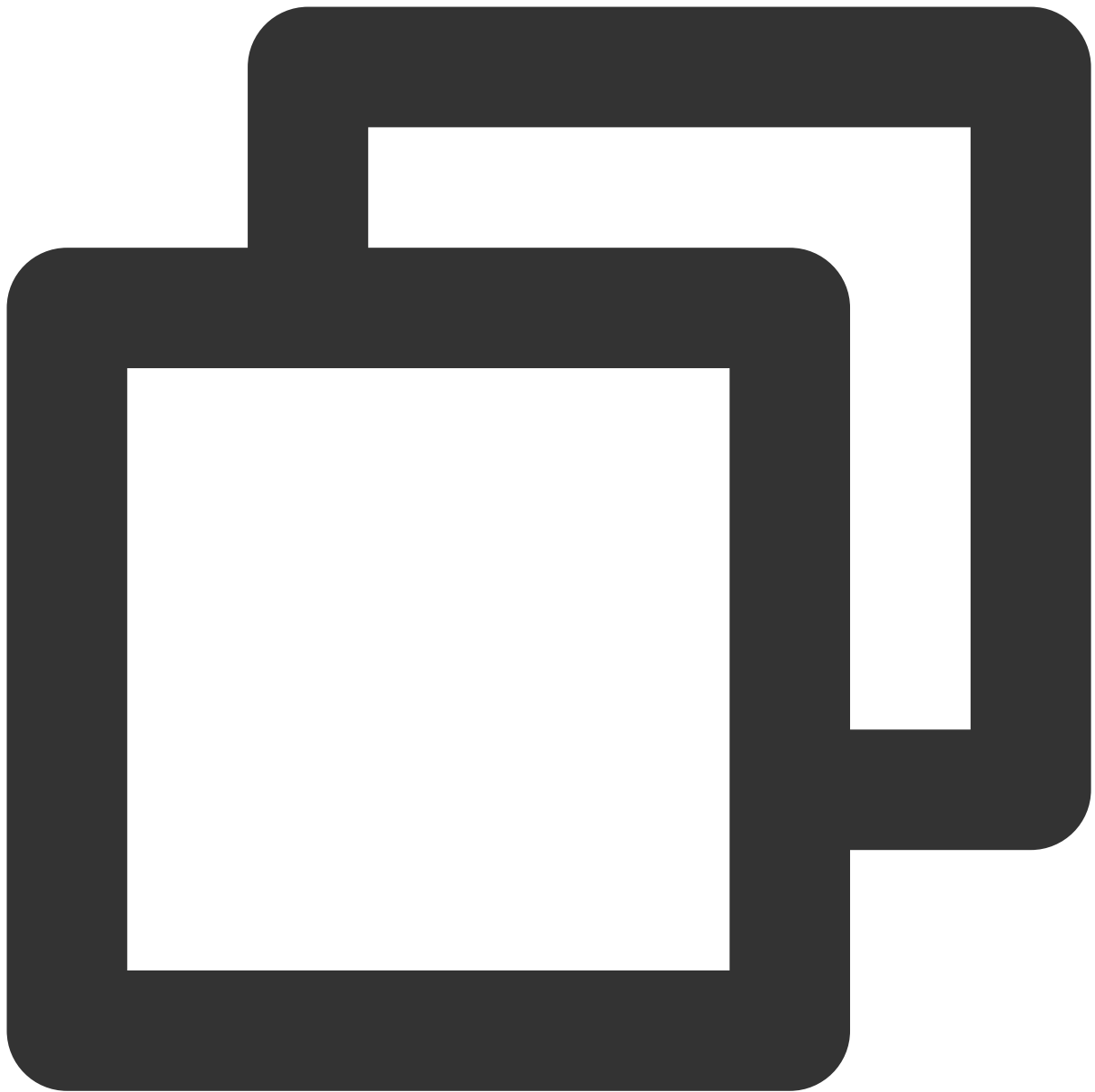
TKE Serverless Pods expose monitoring data via port 9100 by default, and you can access 9100/metrics to get the data by running the following commands:

Get all metrics:



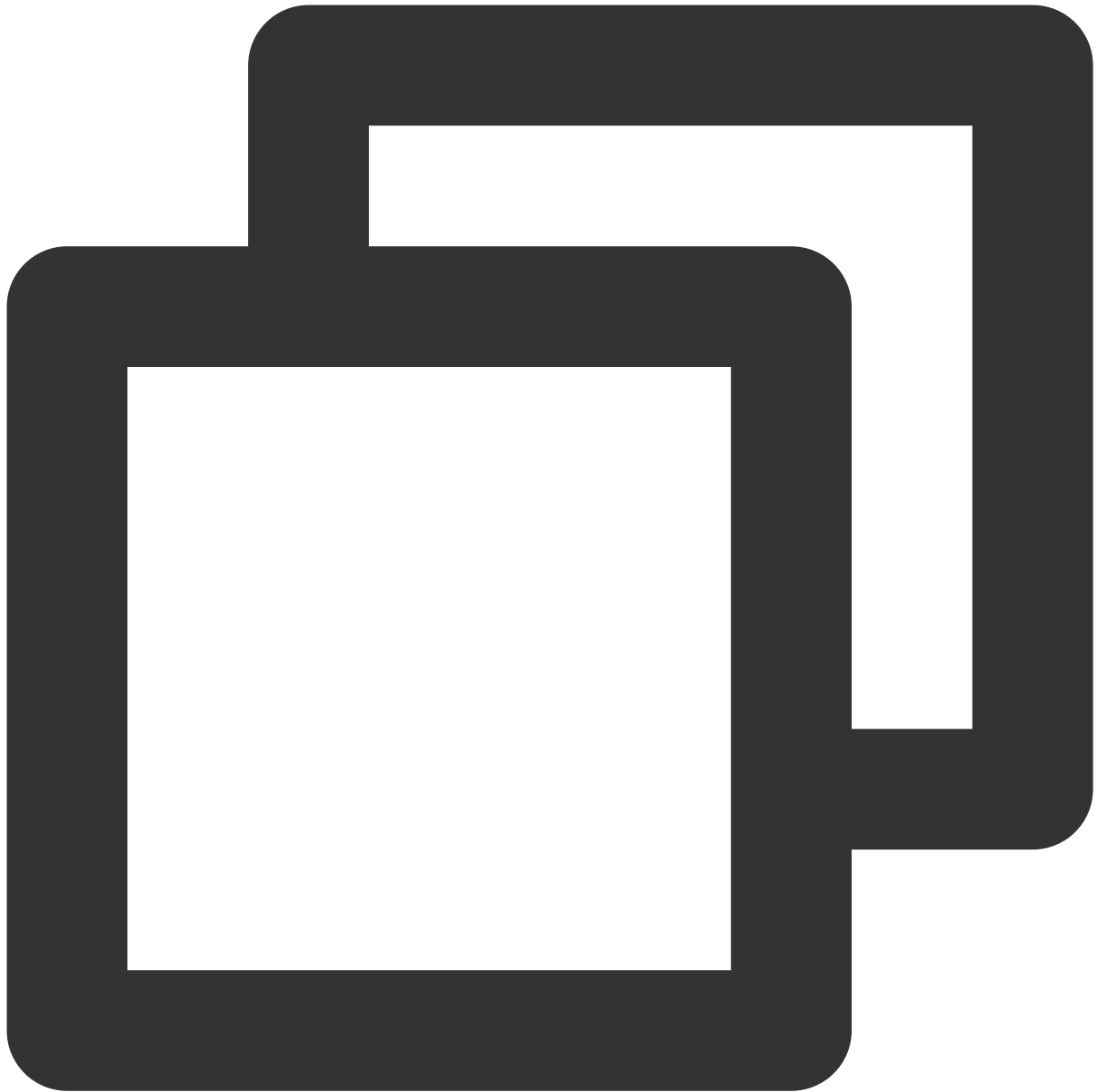
```
curl -g "http://<pod-ip>:9100/metrics"
```

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



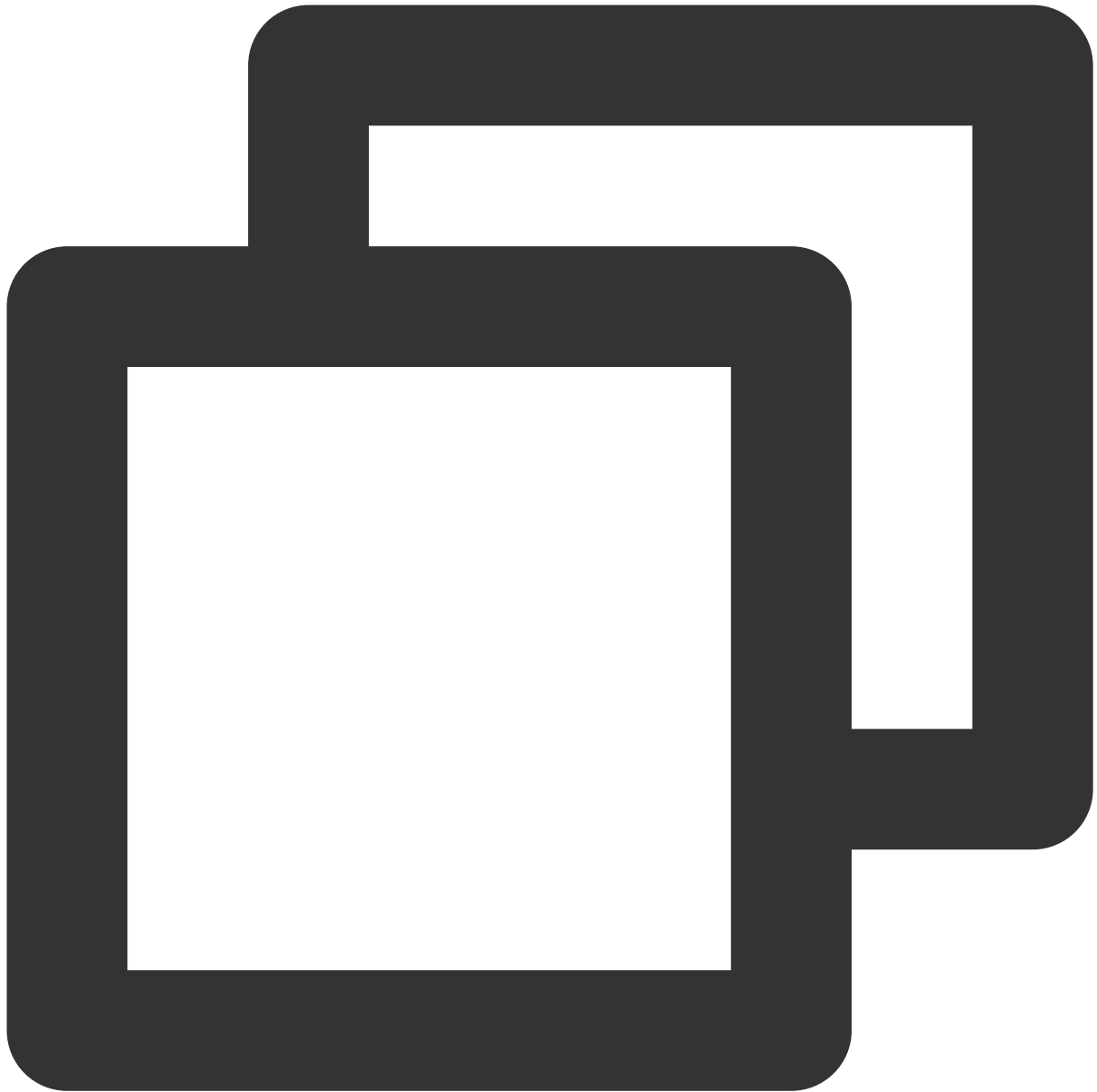
```
curl -g "http://<pod-ip>:9100/metrics?collect[]=ipvs"
```

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 allow traffic over required ports.

Load Balancer FAQs

Last updated : 2023-03-27 11:08:16

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

Prerequisites:

You are familiar with K8s [concepts](#), such as Pod, workload, Service, and Ingress.

You are familiar with the general operations of TKE Serverless clusters in the [TKE console](#).

You know how to use the kubectl command line tool to manage the resources in K8s clusters.

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.

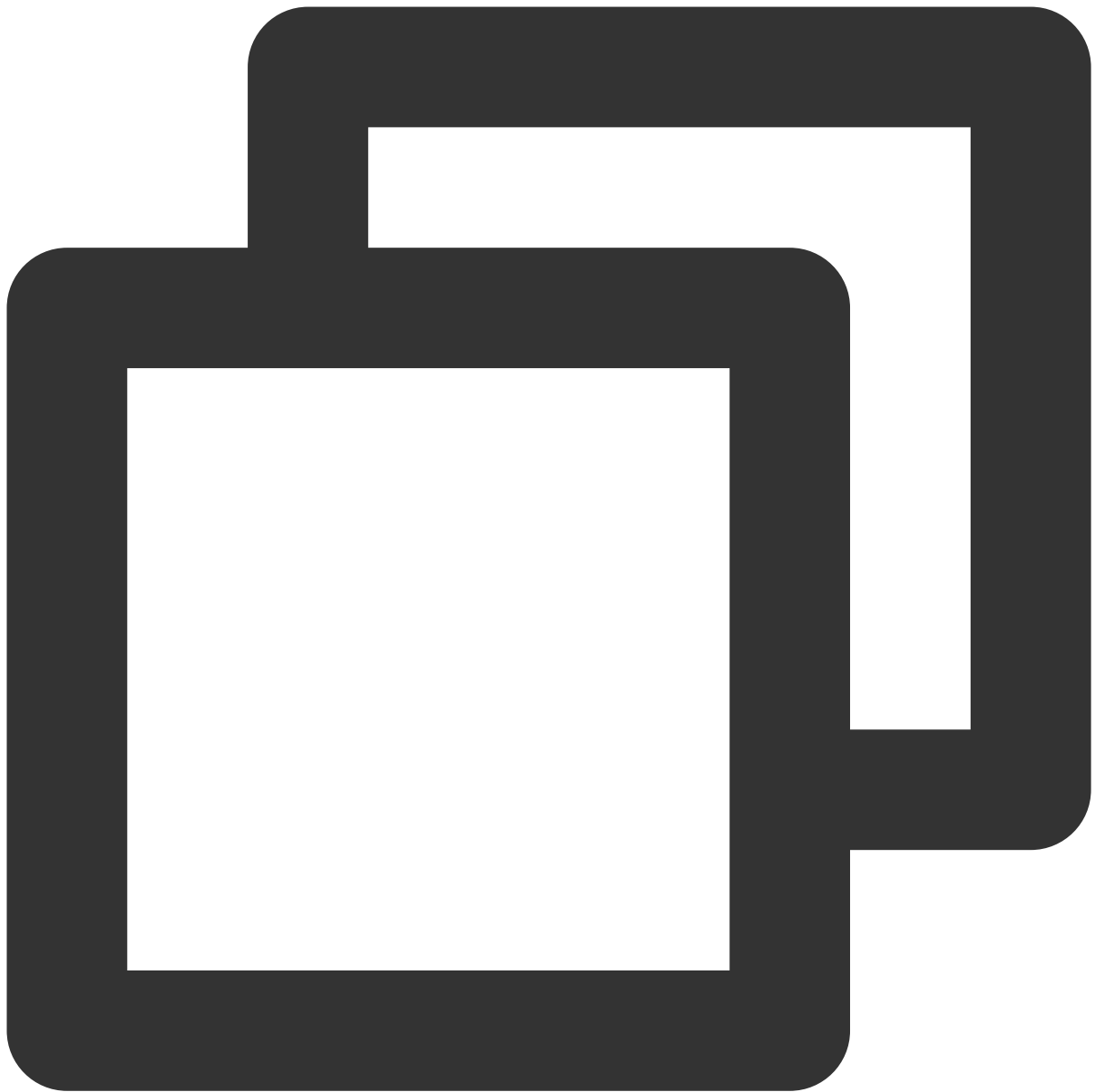
Which Ingress can TKE Serverless create a CLB instance for?

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

Requirements for Ingress Resources	Other notes
Annotations contain the following key-value pair: 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, choose **Service and route** > **Ingress** in the left sidebar.
4. You can find the CLB instance ID and its VIP on the **Ingress** page.



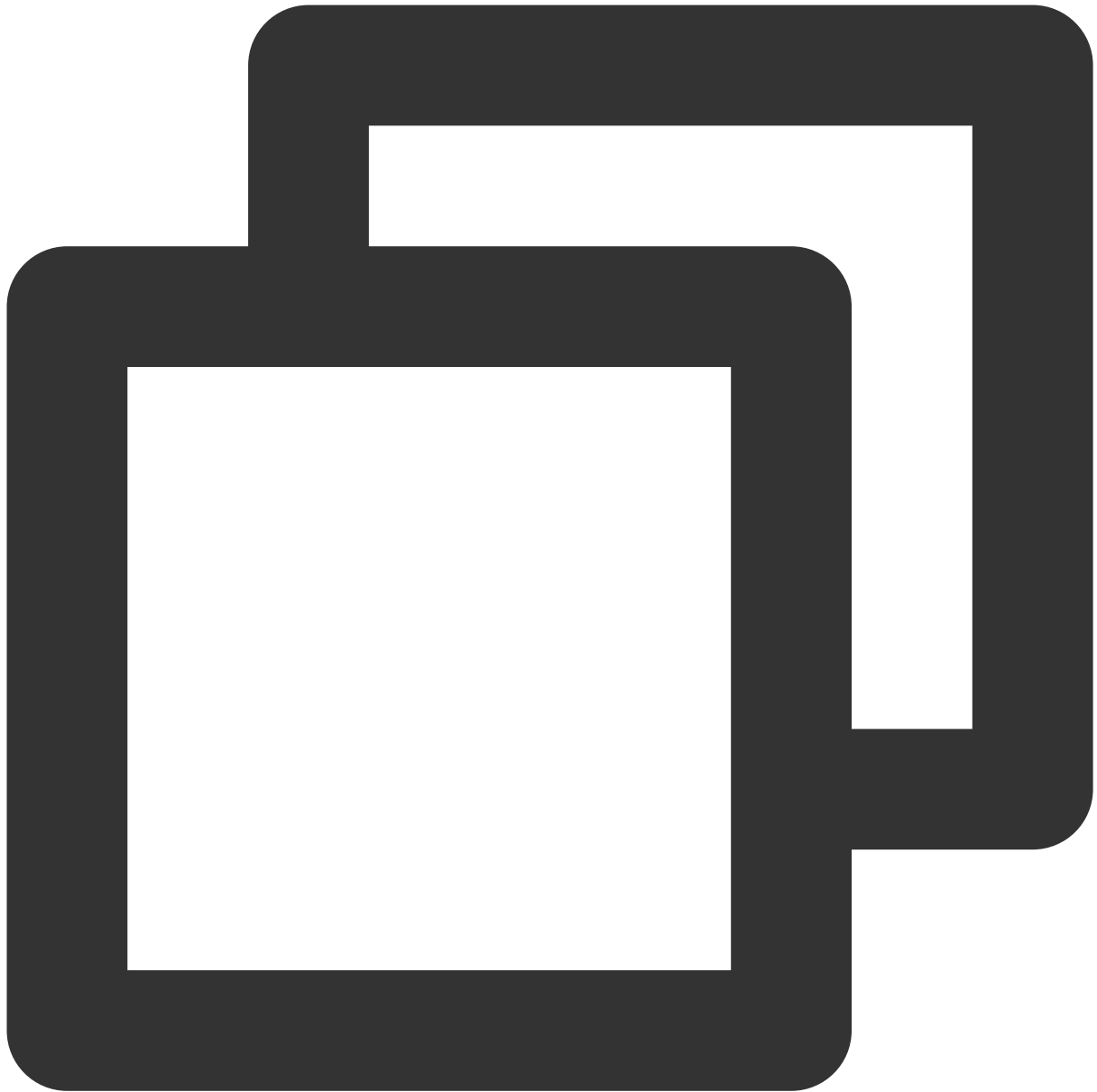
Which Service can TKE Serverless create a CLB instance for?

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	<code>spec.type</code> is <code>LoadBalancer</code> .
The modified version of K8s (Server GitVersion returned by <code>kubectl version</code> has the "eks." or "tke." suffix)	<code>spec.type</code> is <code>ClusterIP</code> , and the value of <code>spec.clusterIP</code> is not <code>None</code> , indicating a non-Headless ClusterIP Service.
The non-modified version of K8s (Server GitVersion returned by <code>kubectl version</code> does not have the "eks." or "tke." suffix)	<code>spec.type</code> is <code>ClusterIP</code> , and <code>spec.clusterIP</code> is specified as an empty string (<code>""</code>).

Note:

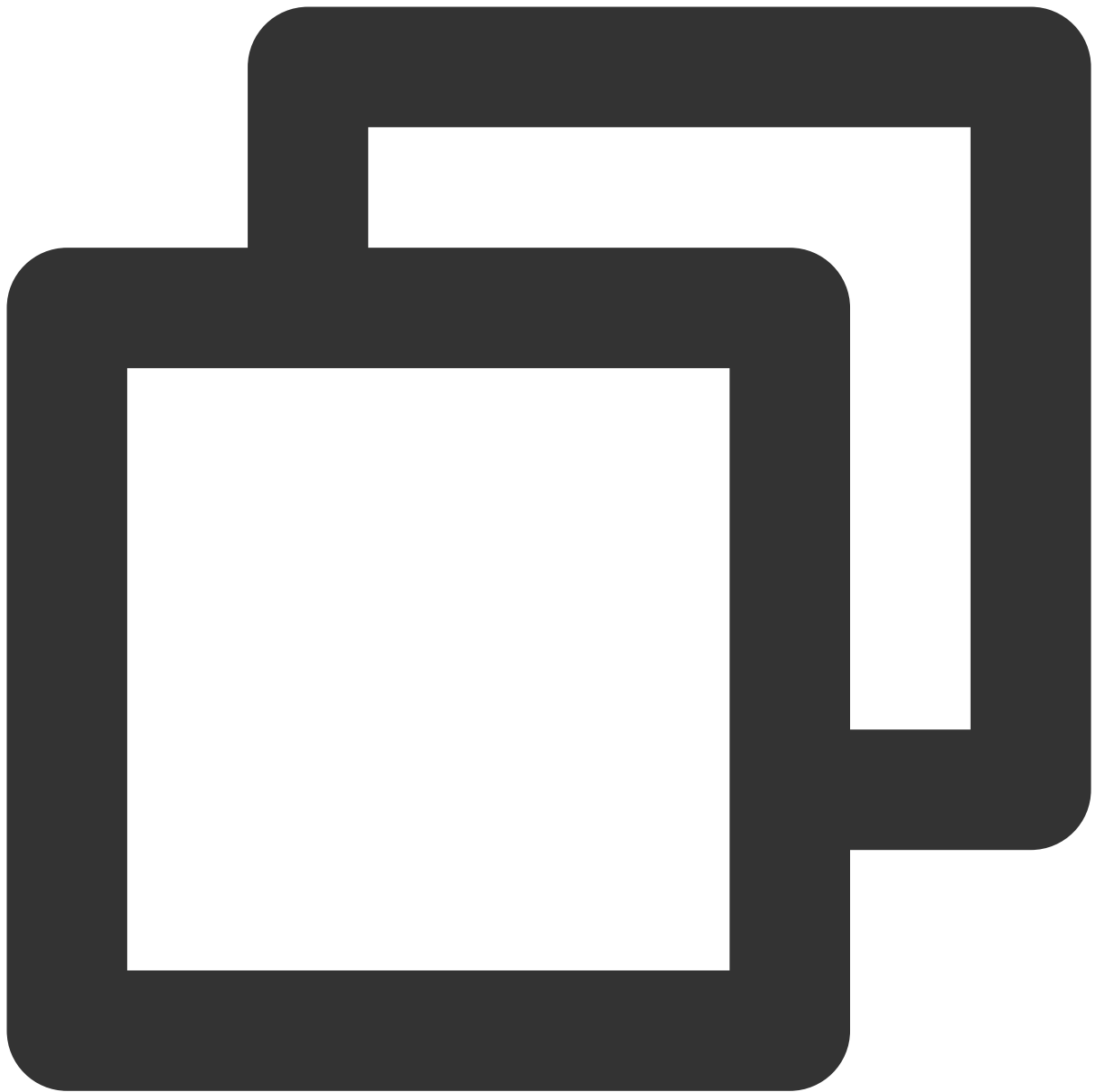
If the CLB instance is successfully created, TKE Serverless will write the following key-value pair to Service annotations:



```
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
```

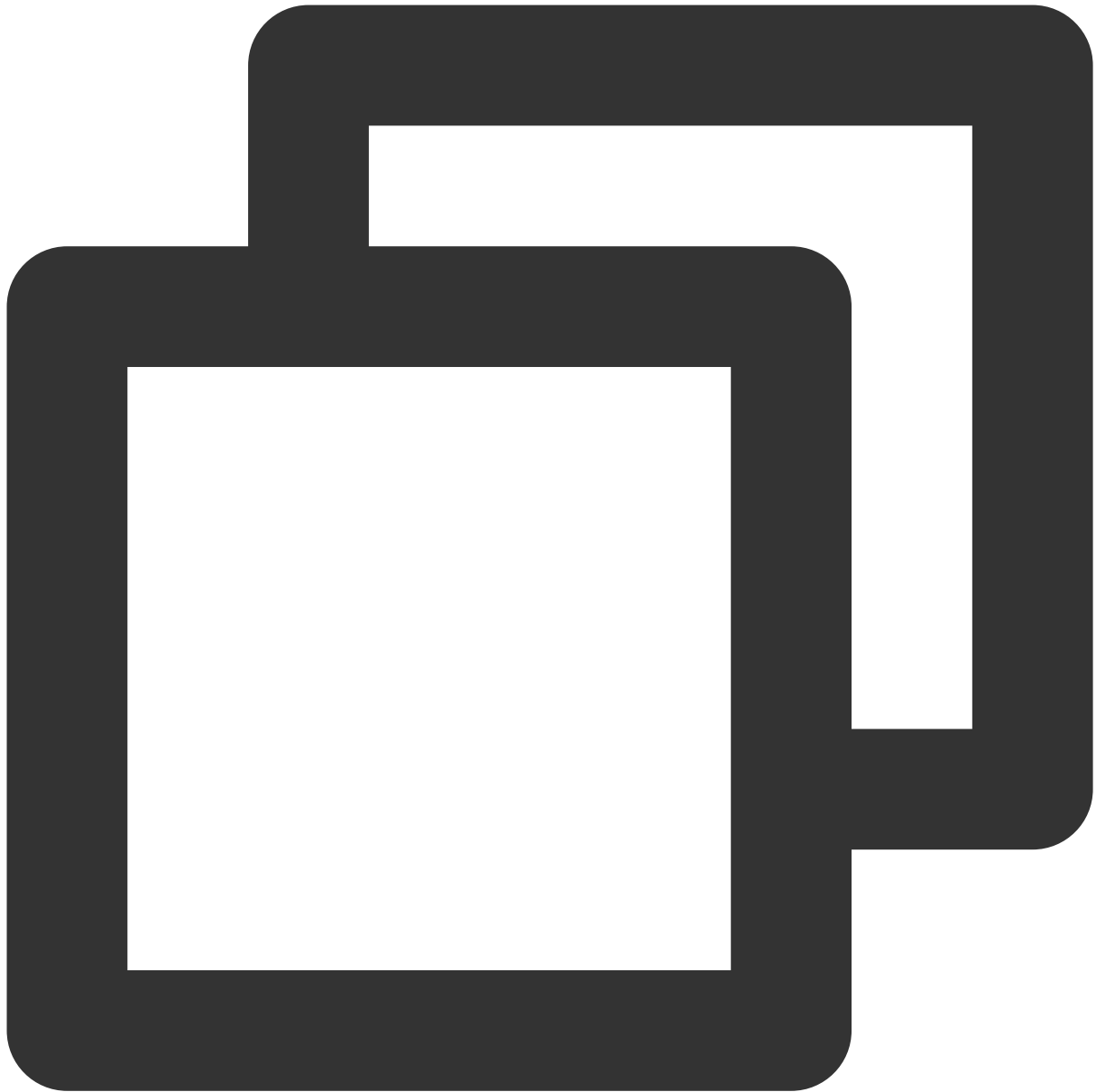
To view the CLB instance created by TKE Serverless for Service, perform the steps below:

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, choose **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 block su
```

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:

Only TKE Serverless clusters that use the modified version of K8s (Server GitVersion returned by `kubectl` version has the "eks." or "tke." suffix) support this feature. For the TKE Serverless clusters created earlier that use 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.

Operations via the TKE console

Operations via the kubectl command line tool

For Ingress, select **Public Network** or **Private Network** for **Network type** to specify the CLB instance type.



For Service, set **Service Access** to specify the CLB instance type. **Via VPC** means the private network CLB instance.



The created CLB instance is of the public network type by default.

If you want to create a CLB instance of the private network type, add the corresponding annotation for the Service or Ingress.

Resource Type	Key-Value Pair to Be Added to Annotations
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 form of `subnet-*****`, and the subnet must be in the VPC specified for the **Cluster network** when creating the cluster. The VPC information can be found in the **Basic information** of the cluster in TKE console.

How do I specify the existing CLB instance?

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

Operations via the TKE console

Operations via the kubectl command line tool

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.

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

Resource Type	Key-Value Pair to Be Added to Annotations
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, choose **Service and route** > **Ingress** in the left sidebar.
4. On the **Ingress** page, click the ID of the target CLB instance to go to the basic information page of the CLB instance.



5. In the **Access log (layer-7)** section of the basic information page of the CLB instance, click



to enable the access log feature for the CLB instance.



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

See [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 conditions for creating a CLB instance. If the conditions are met but the CLB instance is not successfully created, you can use the `kubectl describe` command to view the related events of the resources.

Generally, TKE Serverless will output the related “Warning” events. In the following example, the output event indicates that there are no available IP resources in the subnet, so the CLB instance cannot be successfully created.



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. On the **Ingress** page, click the ID of the CLB instance to go to the basic information page of the CLB instance.



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

Confirming whether the environment for accessing CLB VIP is normal

If the **Instance type** of the CLB instance is the private network, its VIP can only be accessed in the VPC to which it belongs.

Since the IP of the Pods in the TKE Serverless cluster is the ENI IP in the VPC, you can access the VIP of the CLB instance of any Service or Ingress in the cluster in the Pods.

Note

Most LoadBalancer systems have loopback problems (for example, [Troubleshoot Azure Load Balancer](#)). Please do not access the services provided by the workload through the VIP opened by itself (via Service or Ingress) in the Pods to which this workload belongs. That is, Pods should not access the services provided by themselves through the VIP (including "private network" and "public network"). Otherwise, the access delay may increase, or the access will be blocked when there is only one RS/Pod under the rules corresponding to the VIP.

If the **Instance type** of the CLB instance is the public network, its VIP can be accessed in an environment with public network access enabled.

If you want to access the public network VIP in the cluster, please ensure that the public network access has been enabled for the cluster by configuring 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

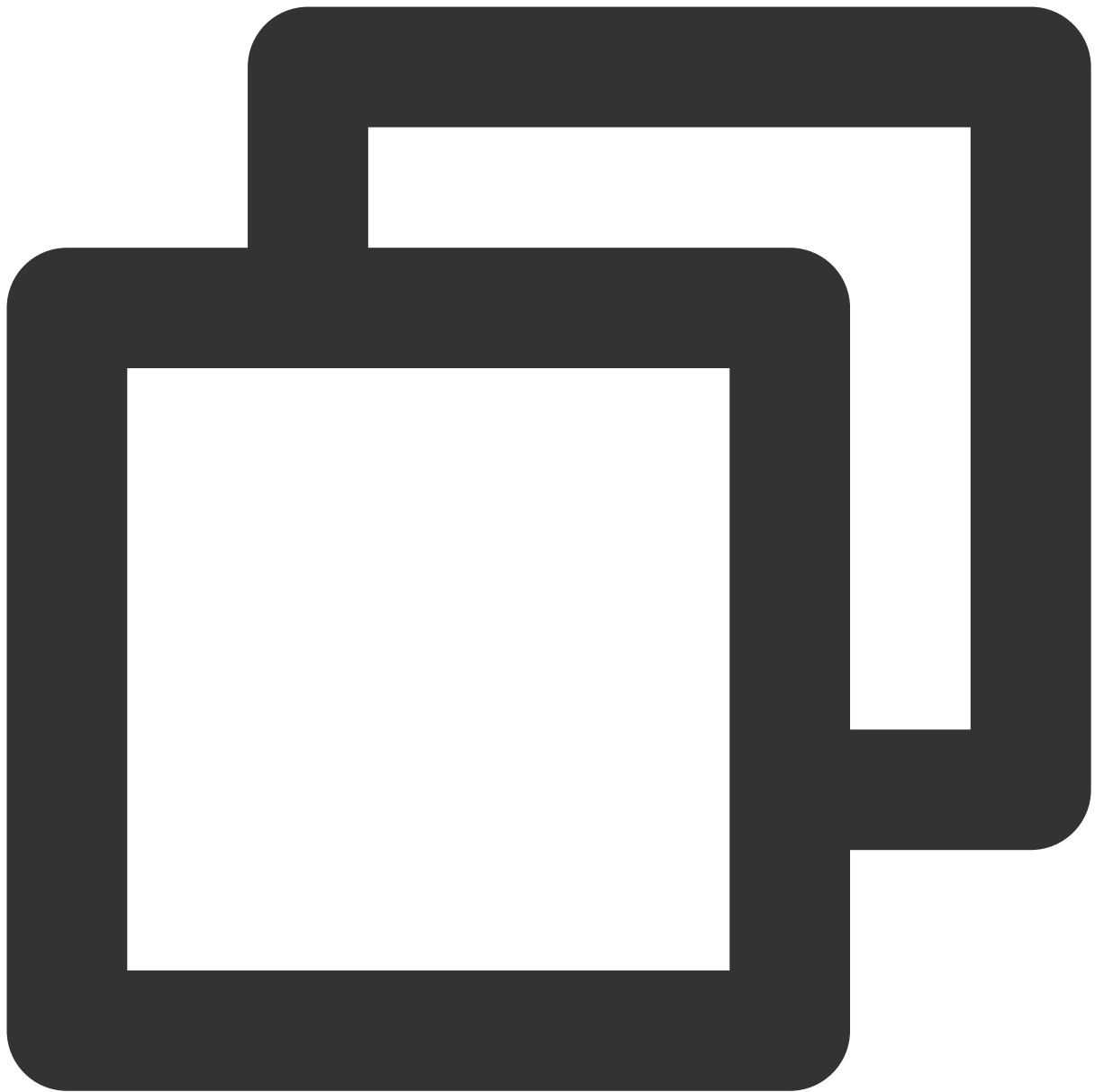
If you have correctly set the labels for the workload and the Selectors for the Service resource, after the Pods of the workload run successfully, you can find that Pods are added by K8s to the ready IP list of the Endpoints corresponding to the Service by running the `kubectl get endpoints` command. The example is as follows:



Pods that are created but in an abnormal state are added by K8s to the unready IP list of the Endpoints corresponding to the Service. The example is as follows:

**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.

Confirming 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 the TKE console

Creating a workload via the kubectl command line tool

The interactive process requires you to specify a security group, and TKE Serverless will use this security group to control the Pods' network access policy. The specified security group will be stored in the

`spec.template.metadata.annotations` of the workload, and finally added to the annotations of the Pods.

An example is as follows:



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 click [Security Group](#) in the left sidebar.
2. Select **Default project** of the same region at the top of the **Security group** page.
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, please [submit a ticket](#) to contact us.

Supernodes FAQs

Last updated : 2023-02-14 14:41:56

How do I prohibit a Pod from being scheduled to a pay-as-you-go supernodes?

By default, an ordinary TKE cluster will automatically schedule a Pod to a pay-as-you-go supernodes after a node pool of the pay-as-you-go supernodes type is added to it when node resources are insufficient. An serverless cluster will automatically schedule Pod randomly at multiple pay-as-you-go supernodes.

If you do not want to schedule a Pod to some pay-as-you-go supernodes (which represents a certain subnet/availability zone), you can cordon the pay-as-you-go supernodes in the following two ways:

- **Cordon** nodes via the [TKE console](#). For more information, see [Cordoning a Node](#).
- Prohibit scheduling by executing the following command through the command line:

```
$kubectl cordon $pay-as-you-go supernodes name
```

How do I prohibit ordinary TKE clusters from automatically scheduling a Pod to a pay-as-you-go supernodes in case of resource inadequacy?

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
```

Specify the value of `AUTO_SCALE_EKS` as `false`, and you can disable the automatic scheduling mechanism to prohibit ordinary TKE clusters from automatically scheduling a Pod to a pay-as-you-go supernodes.

How do I manually schedule a Pod to a pay-as-you-go supernodes?

By default, a pay-as-you-go supernodes automatically adds taints to lower the scheduling priority. If you want to manually schedule a Pod to a (specified) pay-as-you-go supernodes, you need to add corresponding tolerations for the Pod. However, not all the Pods can be scheduled to pay-as-you-go supernodes. For more information, see [Notes for Scheduling Pod to Supernodes](#). For the sake of convenience, you can specify `nodeSelector` in Pod Spec, as shown below ;

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

TKE's control components will judge whether the Pod can be scheduled to a pay-as-you-go supernodes. If not, the Pod will not be scheduled to the pay-as-you-go supernodes.

How do I forcibly schedule a Pod to a pay-as-you-go supernodes, no matter whether the pay-as-you-go supernodes supports the Pod?

Note :

If you forcibly schedule a Pod to a pay-as-you-go supernodes, the Pod may fail to be created. For information on the cause of the failure, see [How do I manually schedule a Pod to a pay-as-you-go supernodes?](#).

If you need to forcibly schedule a Pod to a pay-as-you-go supernodes, you not only need to specify `nodeselector` or `nodename` for the Pod but also add 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 supernodes name
```

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

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

How do I customize DNS configuration for a pay-as-you-go supernodes?

Serverless cluster supports pay-as-you-go supernodes. You can specify annotations in a YAML file to implement capabilities such as custom DNS. For more information, see [Pay-as-you-go Supernodes Annotation Description](#).

Image Repository FAQs

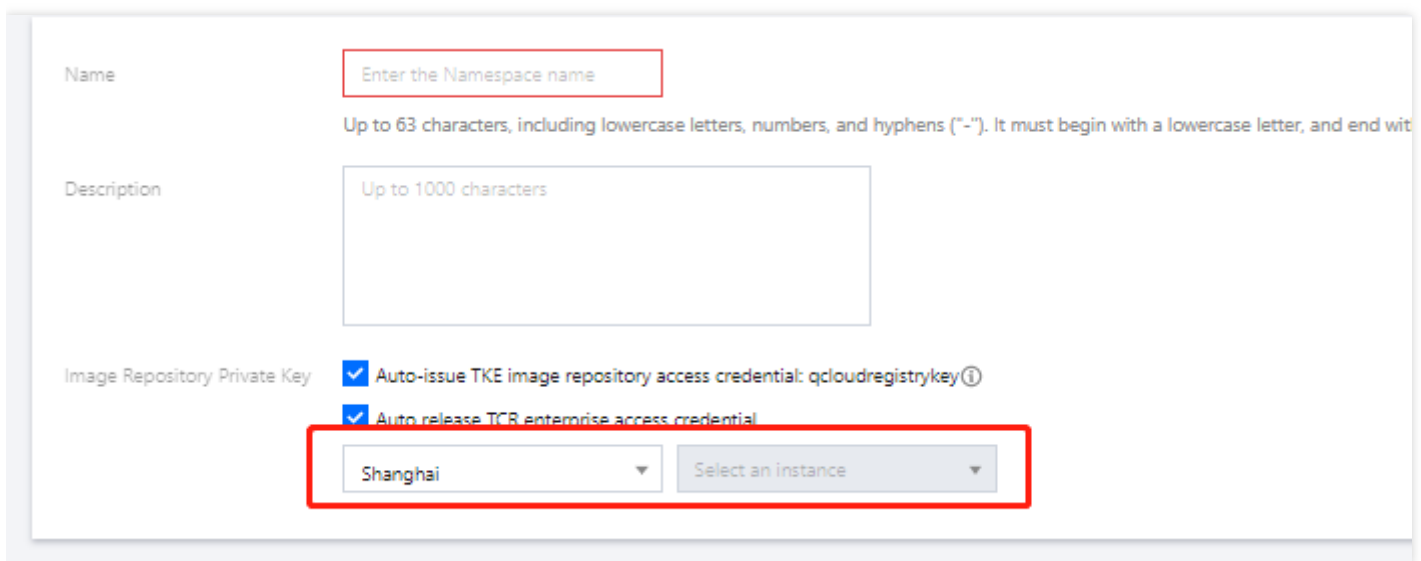
Last updated : 2023-05-23 10:24:27

How do I use the Tencent Container Registry (TCR) service in an serverless cluster?

If you want to use the TCR service in an serverless cluster, ensure that [you have selected the corresponding image access credential](#) and [ensure the network connectivity between the serverless cluster and TCR](#).

Ensuring that you have selected the corresponding image access credential

A container image is private by default. Therefore, you need to select the image access credential for the TCR instance when creating a workload.



The screenshot shows the 'Create Namespace' form in the Tencent Cloud console. The 'Name' field is highlighted with a red box. The 'Image Repository Private Key' section has two checked options: 'Auto-issue TKE image repository access credential: qcloudregistrykey' and 'Auto-release TCR enterprise access credential'. Below these, a dropdown menu is set to 'Shanghai' and is also highlighted with a red box.

You can follow the steps below to create the image access credential:

1. Log in to the [TKE console](#).
2. Click the name of the serverless cluster you need to create the access credential for to go to the serverless cluster details page.
3. Click "Namespace" on the left and click **Create**.
4. On the "Create Namespace" page, select **Auto release TCR enterprise access credential**
5. Click **Create Namespace** to create a namespace, and then you can select the image access credential at the newly created namespace.

Ensuring the network connectivity between the serverless cluster and TCR

The network between the serverless cluster and TCR is not connected by default, so an error indicating network disconnectivity will be reported when you pull the image:

```
dial tcp x.x.x.x:443: i/o timeout
```

Solutions

There are 2 solutions:

Solution	Note
Solution 1: private network access (recommended)	Create a private network access linkage on the TCR console and configure private-network domain name resolution. In this way, the serverless cluster can access TCR via the newly created private network access linkage.
Solution 2: public network access	Enable public network access for the serverless cluster so that it can access TCR via the public network. You also need to make TCR accessible via the public network.

Solution 1: private network access (recommended)

1. Create a private network access linkage
 - i. Log in to the [TCR console](#) and select **Network ACL > Private network** on the left sidebar.
 - ii. Select the region and instance on the "Private network" page and click **Create**.
 - iii. In the "Create a private network access linkage" window that appears, configure the VPC and subnet information.
2. Click **OK** to start creating the private network access linkage.
3. Enable domain name private network resolution.

The domain name of TCR is "<tcr-name>.tencentcloudcr.com", the resolution of which in the VPC needs to be enabled additionally. If the resolution is not enabled, the aforementioned domain name will be resolved into a public IP rather than a private IP, leading to a private network access failure.

Note :

After creating an access linkage, please wait until the backend generates a private IP. After that, the following button can be enabled.

Solution 2: public network access

1. Ensure that TCR has enabled public network access
 - i. Log in to the [TCR console](#) and select **Network ACL > Public network** on the left sidebar.
 - ii. Select the region and instance on the "Public network" page.
 - iii. Enable public network access for the corresponding TCR instance.

In the trial stage, you can set the public IP range as 0.0.0.0/0. In the running stage, you can add the elastic IPs of the NAT gateway egress involved in the following step 2 to the public network allowlist.

2. Enable public network access for the serverless cluster.

Public network access for the serverless cluster is disabled by default, and the serverless cluster needs to access the public network through an NAT gateway. For more information, see [Accessing Internet through NAT Gateway](#). After configuring the NAT gateway, associate the subnet of the serverless cluster with the route table of the NAT gateway, and make sure the elastic IPs of the NAT gateway egress are added to the TCR access allowlist (**for more information, see [step 1](#)**). In this way, the serverless cluster can normally access TCR and pull the image from the public network.

Errors and troubleshooting methods

Error:\sfailed\sto\sdo\srequest:Head\s"xxxx/manifests/late-st":\sdial\stcp\sxxx:443:\si/o\stimeout

An error containing “**443: i/o timeout**” is usually caused by the network disconnectivity between TKE Serverless and TCR.

Please select an access method mentioned in [\[Ensuring the network connectivity between the serverless cluster and TCR\]](#) to realize the network connectivity between TKE Serverless and TCR.

Note

The domain name “<tc<tc>.tencentcloudcr.com” is resolved into a public IP by default. Please figure out the IP address in “dial tcp xxx” indicates a public or private network when the error is reported and solve the problem according to the actual situation.

Error:\scode\s=\sUnknown,\spull\saccess\sdenied,\srepository\sdoes\snot\sexist\sor\smy\srequire\sauthorization:\sserver\smessage:\sinsufficient_scope:\sauthorization\sfailed

An error containing **insufficient_scope: authorization failed**” indicates that the network between TKE Serverless and TCR is interconnected yet you do not have certain permissions. The cause may be that the namespace does not exist, the key is incorrect, or the key is not suitable for the image being pulled, etc.

Error:\scode\s=\sNotFound,\sfailed\sto\sresolve\sreference\s"xxx:xxx":\snot\sfound

An error containing “**not found**” indicates that the image does not exist.

For more information on other common errors, see TCR-related [FAQs](#).

How do I use the external image repository that is created based on the self-signed certificate or HTTP?

Problem description

When you use the image of an external image repository to create a workload in the serverless cluster, you may encounter the error “**ErrImagePull**” and fail to pull the image, as shown in the figure below:

2021-04-02 16:55:20	2021-04-02 16:55:34	Warning	Pod	busybox-764db787c8-4w894.1671fa3df257b8b f	Failed	Error: ErrImagePull	2
2021-04-02 16:55:20	2021-04-02 16:55:34	Warning	Pod	busybox-764db787c8-4w894.1671fa3df252cad f	Failed	Failed to pull image "10.16.100.174:5000/bus..."	2

Cause

Generally speaking, if network connectivity is ensured, the problem may result from the following two causes:

- The external image repository adopts the HTTPS protocol, but the HTTPS protocol certificate is a self-signed certificate.
- The external image repository adopts the HTTPS protocol.

You can solve the aforementioned 2 problems by adding annotations to the PodTemplate in workload Yaml configurations.

Solutions

HTTPS-based self-signed image repository

If the external image repository is an HTTPS-based self-signed image repository, you need to add the following annotation to PodTemplate to make it skip certificate verification.

eks.tke.cloud.tencent.com/registry-insecure-skip-verify: image repository address (for multiple addresses, separate them with “,”, or enter “all”)

Refer to the figure below:

```
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: sampleapp
      qcloud-app: sampleapp
  template:
    metadata:
      annotations:
        eks.tke.cloud.tencent.com/registry-insecure-skip-verify: your-registry-url
      creationTimestamp: null
    labels:
      k8s-app: sampleapp
      qcloud-app: sampleapp
```

Note :

If the images of multiple containers in a Pod are pulled from different repositories, you can enter multiple image repository addresses and separate them with “,”. You can also enter “all”, indicating that all the container image repositories skip certificate verification.

HTTP-based image repository**Note :**

By default, an serverless cluster uses the HTTPS protocol to pull images when running, which means if the image repository supports HTTP, it also needs annotations.

With the command `$kubectl describe pod $podname`, if “**http: server gave HTTP response to HTTPS client**” is exported to report an error, it means the image repository that is accessed uses HTTP, as shown in the figure below:

Events:	Type	Reason	Age	From	Message
	Normal	Scheduled	11m	default-scheduler	Successfully assigned default/busybox-764db787c8-4w894 to eklet-subnet-es398vv5
	Normal	Starting	11m	eklet, eklet	Starting pod sandbox eks-71cpsr8b
	Normal	Starting	10m	eklet, eklet-subnet-es398vv5	Sync endpoints
	Normal	Pulling	9m12s (x4 over 10m)	eklet, eklet-subnet-es398vv5	Pulling image "10.16.100.174:5000/busybox"
	Warning	Failed	9m12s (x4 over 10m)	eklet, eklet-subnet-es398vv5	Failed to pull image "10.16.100.174:5000/busybox": rpc error: code = Unknown desc = failed to pull and unpack image "10.16.100.174:5000/busybox:latest": failed to resolve reference "10.16.100.174:5000/busybox:latest": failed to do request: Head "https://10.16.100.174:5000/v2/busybox/manifests/latest": http: server gave HTTP response to HTTPS client
	Warning	Failed	9m12s (x4 over 10m)	eklet, eklet-subnet-es398vv5	Error: ErrImagePull
	Normal	BackOff	8m58s (x6 over 10m)	eklet, eklet-subnet-es398vv5	Back-off pulling image "10.16.100.174:5000/busybox"
	Warning	Failed	38s (x42 over 10m)	eklet, eklet-subnet-es398vv5	Error: ImagePullBackOff

To solve this problem, you need to add the following annotation to PodTemplate to make it access the image repository through HTTP.

eks.tke.cloud.tencent.com/registry-http-endpoint: image repository address (for multiple addresses, separate them with “,”, or enter “all”)

Refer to the figure below:

```
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: sampleapp
      qcloud-app: sampleapp
  template:
    metadata:
      annotations:
        eks.tke.cloud.tencent.com/registry-http-endpoint: your-registry-url
      creationTimestamp: null
      labels:
        k8s-app: sampleapp
        qcloud-app: sampleapp
```

Note :

If the images of multiple containers in a Pod are pulled from different repositories, you can enter multiple image repository addresses and separate them with “,”. You can also enter “all”, indicating that all the container images are pulled through HTTP.

Notes

Both annotations above involve the entering of image repository addresses, and you can separate multiple repository addresses with “,”.

Note :

If the image repository has a port number, include the port number in the image repository address.

For example, if the image address is `10.16.100.174:5000/busybox:latest` , specify the value of the annotation as `10.16.100.174:5000` , which means the image repository address will be `eks.tke.cloud.tencent.com/registry-insecure-skip-verify: 10.16.100.174:5000` .

Image Repositories

Last updated : 2020-04-21 12:36:34

Image Registry Activation FAQs

What is a namespace used for?

A namespace is an address prefix that identifies a user's private images.

What is an Image Registry account?

By default, this account is your Tencent Cloud account.

What should I do if I forgot the password set during activation?

You can reset the password in the console.

Cluster Creation FAQs

Is there a quota limit on the number of created images?

Yes. The default quota is 500 image repositories per region and 100 image tags per image repository.

If you need a higher quota, [submit a ticket](#).

Can I share my created images with other users?

No, this feature is currently not supported.

How do I use the created images?

First, upload available image tags and then create services by using specific image tags.

Image Deletion FAQs

How do I delete a certain tag of an image?

You can specify and delete a certain tag directly from the console.

When I delete an image from the image list, will all tags of the image be deleted too?

Yes. Be sure to back up your data in advance.

Image Building FAQs

How do I specify the Dockerfile path and building directory when building an image by using source code?

- If you do not enter a path, the system uses the following default values:
 - Default Dockerfile path: Dockerfile under the root directory of the repository (`Dockerfile`).
 - Default building directory: root directory of the repository (`./`).
- To specify a path, **enter a relative path with the project path as the root path**, as shown in the following figure:



 **Build Config**

Image Address

ccr.ccs.tencentyun.com/test-yunx/helloworld

Code source

 Github


 Gitlab

Image Tag Naming Rules

Please enter the naming rule

 - ☐ Branch/label - ☐ Update Time - ☐ Commit No.
Custom prefix, supports variables in the format of \$(Foo)

Overwrite the image tag

Please enter the tag

The generated image also contains the tag

Dockerfile path

qcloud/Dockerfile

Path of the Dockerfile in the code source

Building Directory

qcloud

The working directory for building, should be a relative path

Building Parameters

[Add a variable](#)

Complete

How does the source code building feature use the Dockerfile path and building directory?

Clone the user-specified repository, then switch to the corresponding branch or tag, and lastly run the command `docker build -f $DOCKERFILE_PATH $WORKDIR` in the root directory of the code repository to build a container image.

How do I specify the source path in Dockerfile?

For `COPY` , `ADD` , and other commands related to the source path, specify the source path as a path relative to the building directory.

About Remote Terminals

Last updated : 2020-04-26 16:18:05

What should I do if there is no bash in the container?

If there is no bash in the container, enter the command you want to run in the command line. The result of the command is then displayed on the screen. You can consider the command line as a simplified bash without certain features such as autocomplete. We recommend that you run the bash installation command before proceeding.

Why is software installation using `apt-get` so slow?

When software installation using `apt-get` takes up too much time, this may be due to slow server access to software sources outside China. We provide acceleration solutions for containers with different operating systems. You can select a solution when required.

Precautions

Before selecting the acceleration solution, check the operating system of the container as instructed below:

- Ubuntu: run `cat /etc/lsb-release` to check if the `/etc/lsb-release` file exists.
- CentOS: run `cat /etc/redhat-release` to check if the `/etc/redhat-release` file exists.
- Debian: run `cat /etc/debian_version` to check if the `/etc/debian_version` file exists.

">

Solutions

Ubuntu 16.04

For containers running Ubuntu 16.04, run the following commands on your terminal to set the apt source to a Tencent Cloud source server:

```
cat << EOF > /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
```



```
erse multiverse
ENDOF
```

CentOS 7

For containers running CentOS 7, directly change the source address as instructed below to speed up the installation:

1. Copy and run the following code in the container:

```
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/\${basearch}/
#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}/openstack-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 clear and rebuild the YUM cache.

```

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

```

Debian

For containers running Debian, run the following commands on your terminal to set the apt source to a Tencent Cloud 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
e
deb-src http://mirrors.tencentyun.com/debian-security stretch/updates main
ENDOF

```

Conclusion

Changing the source address in the container directly is an interim solution. When the container is rescheduled, your change becomes invalid. Therefore, we recommend that you solve this problem with the following method when creating the image:

Add the source address provided in [Solutions](#) for corresponding operating systems to the `RUN` field of the Dockerfile for creating the container image. For example, append the following code to the Dockerfile file for an image running the Ubuntu operating system:

```

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 univer
se multiverse
#deb http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted unive
rse multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial main restricted universe mul
tiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-security main restricted uni
verse multiverse
deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-updates main restricted univ
erse multiverse
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-proposed main restricted un
iverse multiverse
#deb-src http://mirrors.tencentyun.com/ubuntu/ xenial-backports main restricted u
niverse multiverse
ENDOF
```

What should I do if I cannot find tools such as vim and netstat after logging in to a container?

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

What should I do if a tool cannot be found when I run the `apt-get install` command?

Install the software program as follows:

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

```
apt-get update
```

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

```
apt-get install
```

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

Go to the remote terminal page, click "File Assistant" in the lower-right corner, and upload your tool.

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

Go to the remote terminal page, click "File Assistant" in the lower-right corner, and upload your file.

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

This issue occurs if the tar program is not included in your container image. To correct it, you can install the tar program by running `apt-get install vim` or `apt-get install net-tools` (run `yum install vim` on CentOS) and try again.

Why can't I find the tools that I installed previously?

When Kubernetes re-schedules your container, it pulls an image to create a new container, and tools that are not in the image will not be installed in the new container. Therefore, we recommend that you install some common troubleshooting tools when creating the image.

How do I copy text in the console?

You can copy text by highlighting and copying it.

How do I paste the copied text?

Press Shift+Insert to paste the copied text.

Why is the connection lost?

This happens when you perform some operations on the container or CVM on another page that modify the container status, or when the current page remains idle for more than 3 minutes. In both cases, the server disconnects the connection.

What should I do if the "TERM environment variable not set" error occurs when I run commands such as `top` ?

Run `export TERM linux` .

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

The bash prompt is set to display "@ " by default. If the current path is too long, bash displays only "<" and the last part of the path by default.

Event FAQs

Last updated : 2022-04-18 15:47:08

Back-off restarting failed docker container

Description: An exceptional Docker container is being restarted.

Solution: Check if the Docker process running in the image has exited due to an exception. If there is no process running in the image, you can add an execution script in the service creation page.

fit failure on node: Insufficient cpu

Description: The cluster CPU is insufficient.

Solution: The node cannot provide enough computing cores. Modify the CPU limit on the service page or scale out the cluster.

no nodes available to schedule pods

Description: Cluster resources are insufficient.

Solution: The reason for this error is that the number of nodes is not sufficient to carry the Pods. From the service page, modify the number of service Pods, the number of Pods or the CPU limit.

pod failed to fit in any node

Description: There is no proper node for the Pod to use.

Solution: This error is caused when the service configures an inappropriate resource limit, which leads to a situation where there are no proper nodes to carry the Pods. Modify the number of service Pods or the CPU limit from the service page.

Liveness probe failed

Description: The container health check failed

Solution: Check if the container processes in the image are normal, and whether the health check port is correctly configured.

Error syncing pod, skipping

Error syncing pod, skipping failed to "StartContainer" for with CrashLoopBackOff: "Back-off 5m0s restarting failed container

Description: The container process crashed or exited.

Solution: Check whether there is a foreground process that is persistently running. If so, check whether it is exceptional. For more information, see [Building a Docker Image Guide](#).

[Submit a ticket](#) if the solutions offered above failed to resolve your issues.