Tencent Kubernetes Engine

TKE Serverless Cluster Guide

Product Documentation

Contents

TKE Serverless Cluster Guide
TKE Serverless Cluster Overview
TKE Serverless Cluster

Last updated：2022-09-23 11:06:22

What is TKE Serverless cluster?

TKE Serverless cluster is an out-of-the-box TKE service that allows you to deploy workloads without purchasing any nodes. It is fully compatible with native Kubernetes, allowing you to purchase and manage resources natively. This service is billed based on the actual amount of resources used by containers. In addition, it provides extended support for Tencent Cloud products, such as storage and network products, and can ensure the secure isolation of containers.

Concepts

Containers and images

Containers are virtualization tools applied at the process level. With the ability to isolate and control system resources, containers restrict global resources access to processes in selected containers. A container image is a virtual machine snapshot and can be seen as the static form of a container. An image defines all files and dependencies required to run a container, ensuring consistency for running the container.
By containerization, all applications and their dependencies are packaged into an image, and then use the image to generate a resource-isolated environment to run the applications. This allows the applications to run independently in a consistent environment in a simple and efficient manner.

Kubernetes

Kubernetes is an open-source Container Orchestration Engine (COE) inspired by a Google project called Borg. It is one of the most important components of the Cloud Native Computing Foundation (CNCF). Kubernetes provides production-level features such as application orchestration, container scheduling, service discovery, and autoscaling. For more information, see Kubernetes Documentation.

Strengths

Native support

TKE Serverless cluster is a community-driven and out-of-the-box service, which supports the latest version of Kubernetes and native Kubernetes cluster management. It serves as a plug-in to provide extended support for Tencent Cloud products, such as storage, network, load balancing products.

Serverless

TKE Serverless cluster is a fully-managed Kubernetes service, which means that you do not need to manage any computing nodes. It delivers computing resources by using Pods. It allows you to purchase, return, and manage cloud resources as in Kubernetes.

High security and reliability

TKE Serverless cluster achieves 99.95% or higher availability based on the mature virtualization technology and network architecture of Tencent Cloud. Tencent Cloud ensures virtual isolation and network isolation between the TKE Serverless clusters of different users and allows users to configure network policies for a specific service by using services such as security groups and network ACL.

Scaling in seconds

With the lightweight virtual technology developed by Tencent Cloud, you can create or delete a TKE instance in seconds to ensure higher efficiency. TKE Serverless cluster allows you to configure the native Horizontal Pod Autoscaler (HPA) of Kubernetes so that services can be automatically scaled based on actual loads.

Reduced costs

The serverless architecture allows TKE Serverless clusters to provide higher resource utilization and lower Ops costs. The flexible and efficient autoscaling capability ensures that TKE instances only consumes the amount of resources required by the current load.

Service integration

TKE Serverless cluster can be highly integrated with most Tencent Cloud services, including the storage products Cloud Block Storage (CBS), Cloud File Storage (CFS), and Cloud Object Storage (COS), TencentDB product family, and virtual private cloud (VPC) product family. With this capability, TKE Serverless cluster can provide solutions that meet the requirements of a wide range of businesses.

Use Limits

Please see Purchase Limits for purchase limits, and see Resource Specifications for information about resource specifications.

Pricing

Three billing modes are available for TKE Serverless clusters: Reservation, pay-as-you-go, and spot mode. For more information, see Product Pricing.

Comparison with TKE

Feature	TKE General Cluster	TKE Serverless Cluster
Kubernetes	This feature is natively supported.	This feature is natively supported. Some features are not supported due to the lack of computing nodes. For more information, see Notes.
VPC	This feature is supported.	
Computing nodes	You need to purchase and manage computing nodes such as Cloud Virtual Machine (CVM) and Bare Metal (BM) nodes on your own.	You do not need to purchase any computing nodes.
Management method	Native Kubernetes APIs and Kubectl are supported.	
Clusters	Multiple clusters can be created and managed.	
Namespaces	This feature is natively supported.	
Workloads	This feature is natively supported.	Native Kubernetes workloads, except DaemonSet, are supported.
Service	This feature is natively supported. A CLB plug-in is integrated with TKE.	
Storage	This feature is natively supported. Plug-ins such as CBS and CFS can be integrated with TKE.	

Use Cases

Microservices

Running microservices with TKE Serverless clusters can free users from Ops of computing nodes. A service can be automatically scaled based on the actual load and use the necessary amount of resources to run applications, which reduces resource costs.

Offline computing

To run an offline computing task with a TKE Serverless cluster, you simply need to prepare a container image to quickly deploy workloads for the task. In addition, a TKE Serverless cluster bills only the actual amount of computing resources used during the execution of the task and stops billing when Pods are automatically released after the task ends.

Online inference

TKE Serverless cluster can run online inference services by using CPU, GPU, and vGPU resources. The abundant resource specifications and the workloads that support autoscaling improve the operating efficiency and cost-effectiveness of the online inference services.

Additional Services

- Storage: To use a cloud disk or file storage as the persistent storage of a container, you can use CBS and CFS.
- Network:
- To create and manage your VPCs, for example, to create a VPC instance and a subnet, establish a peering connection, use the NAT Gateway, configure a route table, and configure a security policy, please refer to VPC Documentation.
- To manage access configurations for private and public network of services, please refer to CLB Documentation.
- APIs: For information on calling Tencent Cloud APIs to access to Tencent Cloud products and services, see Tencent Cloud API Documentation.

Purchasing a TKE Serverless Cluster
Regions and Availability Zones

Last updated：2023-03-30 16:26:40

Regions

Overview

A region is the physical location of an IDC. In Tencent Cloud, regions are fully isolated from each other, ensuring cross-region stability and fault tolerance. When purchasing Tencent Cloud services, we recommend selecting the region closest to your end users to minimize access latency and improve download speed.
You can see the table below for the regions, availability zones and resource types currently supported by TKE Serverless cluster. Other regions will be opened gradually.

Characteristics

The networks of different regions are fully isolated. Tencent Cloud services in different regions **cannot communicate via a private network by default**.
Tencent Cloud services in different regions can communicate with each other through public IP addresses over the Internet, while those in different VPCs can communicate with each other through CCN, which is faster and more stable.
Cloud Load Balancer currently supports intra-region traffic forwarding, and binding of CVMs in the same region. If the cross-region binding function is enabled, then binding CVMs to the Cloud Load Balancer across regions is supported.

AZs

Overview

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

Characteristics

Tencent Cloud products that are in the same region, different AZs, and the same VPC are interconnected through the private network. They can be accessed directly through private IP addresses.

Note

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

AZs Supported by TKE Serverless Clusters

China

Region(s)	Availability Zone
South China (Guangzhou) ap-guangzhou	Guangzhou Zone 3 ap-guangzhou-3
	Guangzhou Zone 4 ap-guangzhou-4
	Guangzhou Zone 6 ap-guangzhou-6
	Guangzhou Zone 7 ap-guangzhou-7
East China (Shanghai) ap-shanghai	Shanghai Zone 2 ap-shanghai-2
	Shanghai Zone 3 ap-shanghai-3
	Shanghai Zone 4 ap-shanghai-4
	Shanghai Zone 5 ap-shanghai-5
East China (Nanjing) ap-nanjing	Nanjing Zone 1 ap-nanjing-1
	Nanjing Zone 2 ap-nanjing-2
North China (Beijing) ap-beijing	Beijing Zone 3 ap-beijing-3
	Beijing Zone 4

	ap-beijing-4
	Beijing Zone 5 ap-beijing-5
	Beijing Zone 6 ap-beijing-6
	Beijing Zone 7 ap-beijing-7
Southwest (Chengdu) ap-chengdu	Chengdu Zone 1 ap-chengdu-1
	Chengdu Zone 2 ap-chengdu-2
Hong Kong/Macao/Taiwan (China Region) (Hong Kong, China) ap-hongkong	Hong Kong Zone 2 (Hong Kong nodes can cover services in the China regions of Hong Kong, Macao, and Taiwan) ap-hongkong-2

Other countries and regions

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

	ap-mumbai-2
Southeast Asia Pacific (Bangkok) ap-bangkok	Bangkok Zone 1 (Bangkok node users cover services in Southeast Asia) ap-bangkok-1
North America (Toronto) na-toronto	Toronto Zone 1 (Toronto nodes cover services in North America) na-toronto-1
East US (Virginia) na-ashburn	Virginia Zone 1 (Virginia node users cover services in East US) na-ashburn-1
	Virginia Zone 2 (Virginia node users cover services in East US) na-ashburn-2
Europe Zone (Frankfurt) eu-frankfurt	Frankfurt Zone 1 (Frankfurt nodes cover services in Europe) eu-frankfurt-1

Billing Overview

Last updated：2022-12-08 18:03:06

Overview

TKE serverless clusters sustain resources with **pay-as-you-go** super nodes. For more information, see Product Pricing.

Billing Mode

In TKE serverless clusters, pay-as-you-go super nodes are billed based on the specifications and execution duration of Pods scheduled to them. For billing details, see Product Pricing.

Other Fees

If you use TKE serverless clusters with other paid products such as CLB, CBS, and CFS, these products will be billed according to their own billing rules. For more information, see the purchase guide for each product.

Product Pricing

Last updated：2023-05-30 10:44:52

The following pricing takes effect from 00:00 on July 1, 2023 (UTC +8).

Pay-as-You-Go Pricing

TKE Serverless Cluster is billed based on the resource specification and usage. **Fee** = **Billable item specification** × **Resource price per unit time** × **Running time (in seconds)**. For details on specifications of billable items, see Resource Specifications.

Intel Pod Pricing

Region	Billable items			
	CPU (USD/core/second)	Memory (USD/GiB/second)	Premium cloud disk (USD/GB/second) Free of charge of the first 20 GB	SSD cloud disk (USD/GB/second) Free of charge of the first 20 GB
Shanghai, Beijing, Guangzhou, Nanjing, Chongqing, Chengdu, Qingyuan, Wuhan, Changsha, Zhengzhou, Xi'an, Shenyang, Fuzhou, Hefei, Jinan, Hangzhou, Shijiazhuang, Shanghai Finance, Shenzhen Finance, Beijing Finance,	0.000004976	0.000002073	0.00000004	0.000000104

Shanghai ADC, Japan, Thailand, Silicon Valley, India, Virginia, Sao Paulo				
Hong Kong (China), Taipei (China), Singapore	0.000004976	0.000002248	0.00000002	0.000000010
Seoul, Frankfurt	0.000005224	0.000002612	0.00000002	0.000000010
Jakarta	0.000005804	0.000002902	0.00000002	0.000000010

AMD Pod Pricing

Region	Billable items			
	CPU (USD/core/second)	Memory (USD/GiB/second)	Premium cloud disk (USD/GB/second) Free of charge of the first 20 GB	SSD cloud disk (USD/GB/second) Free of charge of the first 20 GB
Shanghai, Beijing	0.00000269	0.00000133	0.00000004	0.000000104
Guangzhou, Qingyuan, Wuhan, Changsha, Zhengzhou, Xi'an, Shenyang, Fuzhou, Hefei, Jinan, Hangzhou, Shijiazhuang	0.00000253	0.00000133	0.00000002	0.000000010
Shanghai Finance, Shenzhen Finance, Beijing	0.00000410	0.00000203	0.00000002	0.000000010

Finance, Shanghai Auto-Driving Cloud				
Hong Kong (China), Taipei (China), Japan, Singapore, Thailand, India, Virginia, Sao Paulo, Frankfurt	0.00000361	0.00000182	0.00000002	0.000000010
Silicon Valley	0.00000299	0.00000149	0.00000002	0.000000010

Running Time

The running time is calculated in seconds, staring when a Pod fetches the first container image until it stops. The Pod is billed based on the resource usage during this period.

Billing Examples

Example 1

Assume that there is a Deployment in Virginia, whose resource specification is 2C4G with 2 replicas. It takes 5 minutes (300 seconds) from the time when the Deployment is started to the time it ends.

In this case, the fee of the Deployment = 2 × (2 × 0.000004976 + 4 × 0.000002073) × 300 = 0.019495 USD.

Example 2

Assume that a CronJob in Silicon Valley needs to start 10 4C8G AMD Pods each time and terminate the Pods 10 minutes (600 seconds) later. If the CronJob executes the job twice a day, the task is billed as follows:

Daily task fee = 2 × 10 × (4 × 0.00000299 + 8 × 0.00000149) × 600 = 0.28656 USD.

Purchase Limits

Last updated：2022-12-08 18:03:06

Use Restrictions

Before using a TKE serverless cluster, you need to sign up for a Tencent Cloud account and complete identity verification.

Supported Regions

For more information on regions where TKE serverless clusters are supported, see Regions and Availability Zones.
For more information on resource specifications, see Resource Specifications.

Resource Quotas

Cluster and Pod limits

Resource	Limit	Description
Clusters in one region	5	Includes clusters that are being created and running.
Pods in one cluster	100	Includes all namespaces, workloads, and stateless and stateful pods.
Pod replicas for one workload	100	Includes all stateless and stateful pods in the workload.
The largest container instance size for the same region	500	Includes all stateless and stateful container instances.

Other limits

All Pods in a TKE serverless cluster are independent computing and network instances in the cloud, which can be regarded as CVM instances and are therefore subject to the following limits:

1. When a workload is created, each Pod will be associated with a security group by default, and all Pod replicas in the same workload will be associated with the same security group. Each Pod is a CVM instance for the security group and is subject to the quota on the number of CVM instances that can be associated with a security group.

2. When CLB Services are used, each Pod bound to the CLB instance is a CVM instance for the CLB instance and is subject to the quota on the number of servers that can be bound to a forwarding rule in a CLB instance.

Applying for a higher quota

If the number of required resources exceeds the quota limit shown in the preceding table, you can submit a ticket to apply for a higher quota. Tencent Cloud will assess your actual needs and increase your quota as appropriate.

1. Submit a ticket. Select **Others** > **Create Now** to enter the ticket creation page.
2. In the **Problem description** field, enter a description such as "I want to apply for a higher quota for the TKE serverless cluster." Specify the target region, object, and quota. Enter your mobile number and other information as instructed.
3. Click **Submit Ticket**.

Resource Specifications

Last updated：2023-05-06 17:36:46

Overview

TKE serverless clusters free you from managing cluster nodes. However, to properly allocate resources and accurately calculate fees, you need to specify resource specifications for Pods when deploying a workload. Tencent Cloud allocates computing resources to the workload and calculates the corresponding fees based on the specified specifications.

When you use the Kubernetes API or kubectl to create a workload for a TKE serverless cluster, you can use annotations to specify resource specifications. If annotations are not used, the TKE serverless cluster will calculate the specifications based on the container parameters set for the workload, such as Request and Limit. For more information, see Specifying Resource Specifications.

Note

The resource specifications indicate the maximum amount of resources available for containers in a Pod.

The following tables list the supported CPU and GPU specifications. Ensure that allocated resources do not exceed the supported specifications.

The total amount of resources specified by Request for all the containers in a Pod cannot exceed the highest Pod specification.

The amount of resources specified by Limit for any container in a Pod cannot exceed the highest Pod specification.

CPU Specifications

The following table lists CPU specifications that TKE serverless clusters provide for Pods in all regions where CPU resources are supported. TKE serverless clusters also provide a set of CPU options. Different CPU sizes correspond to different memory ranges. Select the CPU specification as needed when creating a workload.

Intel

CPU (Cores)	Memory Range (GiB)	Memory Range Granularity (GiB)
0.25	0.5, 1, 2	-
0.5	1, 2, 3, 4	-
1	1–8	1
2	2, 4–16	1

4	8–32	1
8	16–32	1
12	24–48	1
16	32–64	1

Star Lake AMD

Based on Tencent Cloud's self-developed Star Lake servers, EKS provides reliable, secure, and stable high performance. For more information, see Standard SA2.

CPU (Cores)	Memory Range (GiB)	Memory Range Granularity (GiB)
1	1–4	1
2	2–8	1
4	4–16	1
8	8–32	1
16	32–64	1

GPU Specifications

The following table lists the GPU specifications that TKE serverless clusters provide for Pods. Different GPU card models and sizes map to different CPU and memory options. Select the GPU specification as needed when creating a workload.

Note

If you want to create, manage, and use GPU workloads using a YAML file, see Annotation Description for reference.

GPU Model	GPU	CPU (Cores)	Memory (GiB)
NVIDIA Tesla V100 - 1	1	8	40
NVIDIA Tesla V100 - 2	2	18	80
NVIDIA Tesla V100 - 4	4	36	160
NVIDIA Tesla V100 - 8	8	72	320
1/4 NVIDIA T4 - 1/4	1	4	20
1/2 NVIDIA T4 - 1/2	1	10	40

NVIDIA T4 - 1	1	8	32
NVIDIA T4 - 1	1	20	80
NVIDIA T4 - 1	1	32	128
NVIDIA T4 - 2	2	40	160
NVIDIA T4 - 4	4	80	320
NVIDIA A10 - PNV4 - 1	1	28	116
NVIDIA A10 - PNV4 - 2	2	56	232
NVIDIA A10 - PNV4 - 4	4	112	466
NVIDIA A10 - PNV4 - 8	8	224	932
NVIDIA A10 - GNV4 - 1	1	12	44
NVIDIA A10 - GNV4v - 1/4	1	6	24
NVIDIA A10 - GNV4v - 1/2	1	14	58
NVIDIA A10 - GNV4v - 1	1	28	116

Spot Mode

Last updated：2022-09-13 15:01:50

Spot Mode Overview

You can purchase resources with low costs in the spot mode of TKE Serverless Cluster. In some scenarios, you can pay a price lower than that of the pay-as-you-go instances to run Pods until they are interrupted and repossessed by Tencent Cloud, greatly reducing the costs.

When using the spot mode, you can deploy workloads in containers just like you are using other pay-as-you-go resources.

Spot Mode Policy

Price policy

A **fixed discount (80% off)** is used by the spot mode of TKE Serverless Cluster. That means the Pod resources with all specifications are sold at the fixed discount (80% off) of the original prices defined in Product Pricing.

> Note：
> The discount only applies to the billable items (CPU, memory and GPU) of resources in the Pods. It is not applicable to resources such as network bandwidth, network traffic and persistent storage.

Interruption and repossessing mechanism

In the spot mode, when the resources in Tencent Cloud computing resource pool are insufficient, the assigned Pods are interrupted and repossessed randomly. Note that the cache data in the Pods will not be retained.

The following events may occur when the Pods are interrupted and repossessed:

```
EVENT REASON : “SpotPodInterruption”
EVENT MESSAGE : “Spot pod was interrupted, it will be killed and re-created”
```

Use Cases

Short-time emergent and periodic tasks

The spot mode is suitable for workloads that do not run for a long time. For example, video transcoding, video rendering, service stress testing, batch computing and crawlers.

Divisible computing tasks

The spot mode is suitable for systems that can divide long-term tasks into fine-grained tasks for computing based on the objects, such as EMR and other big data suite.

Stateless computing tasks or tasks supporting checkpoint restart

- It is suitable for workloads that put the intermediate computing results on the persistent storage and continue the computing after being restarted when Pods are repossessed.
- It is suitable for stateless workloads that support automatic load balancing and service discovery, and workloads that can be restarted when the Pods are repossessed.

Enabling the Spot Mode

You can enable the spot mode for a workload by adding the following Pod template annotation in the workload YAML.

```
eks.tke.cloud.tencent.com/spot-pod: "true"
```

For more information, see Annotation.

TKE Serverless Cluster Management
Creating a Cluster

Last updated：2023-08-24 14:38:13

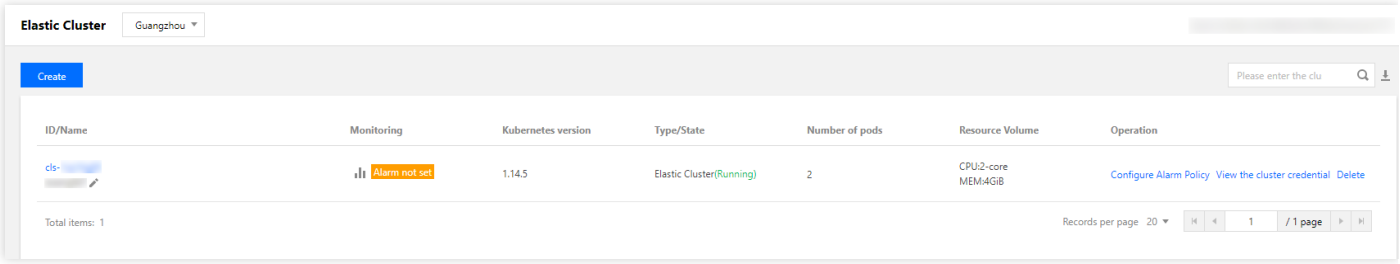
This document describes how to create a TKE Serverless cluster in the TKE console.

Prerequisite

You have activated required permissions for EKS in the CAM console.

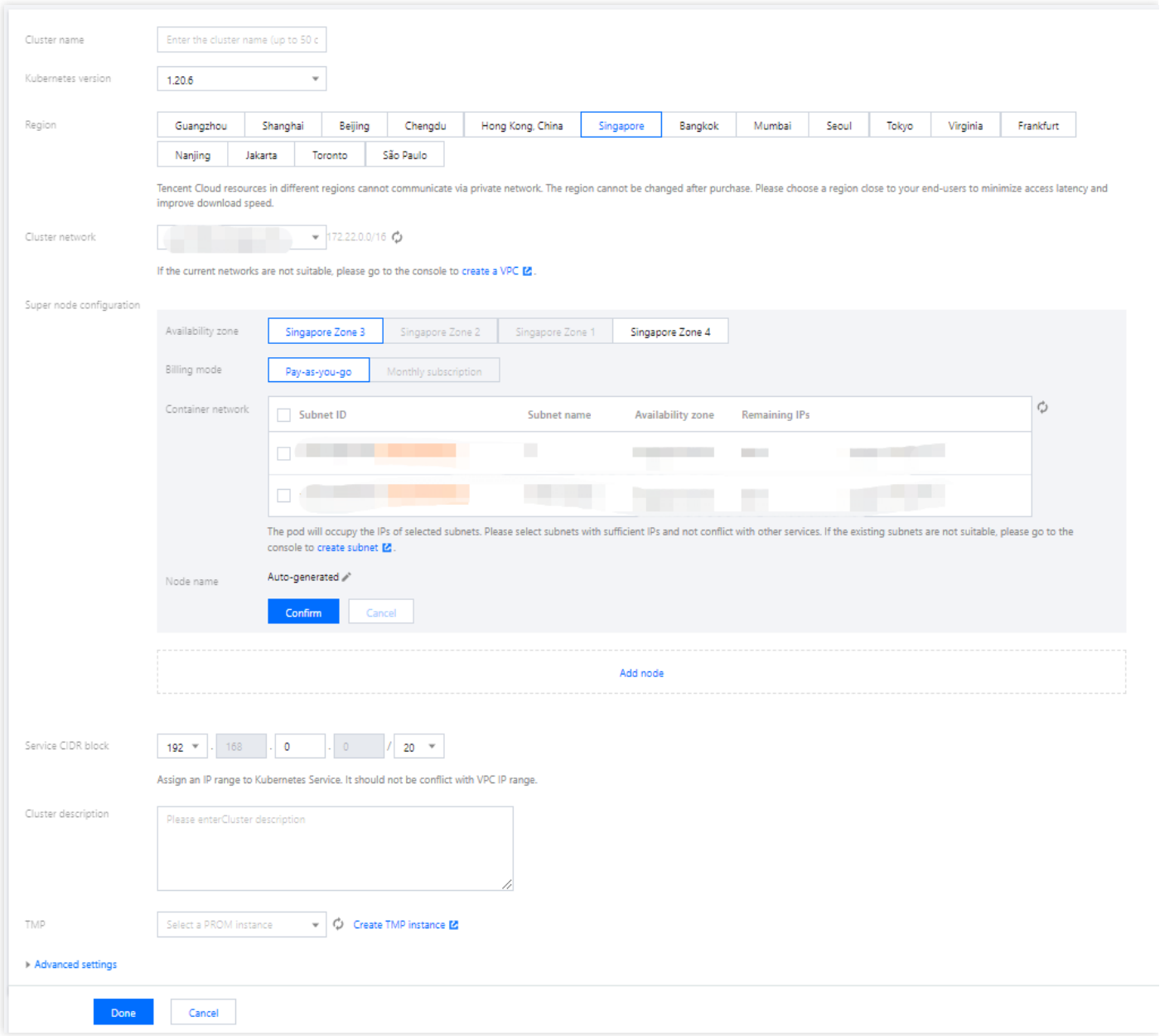
Directions

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the page that appears, select the region where you want to create the cluster, and then click **Create**.



3. In the “Select a cluster type” pop-up window, select **Serverless cluster**, and then click **Create**.

4. On the “Create cluster” page that appears, specify the cluster information as instructed below.



- **Cluster name**: The name of the cluster to be created. Up to 60 characters are allowed.
- **Kubernetes version**: The Kubernetes version of the cluster to be created. It supports V1.16 and later. For a comparison of the features of different versions, see Supported Versions of the Kubernetes Documentation.
- **Region**: The region where the cluster to be created is located. We recommend that you select the region closest to you to help minimize access latency and improve the download speed.

- **Cluster network**: All VPCs you have created in this region are displayed. Please select a suitable VPC as the cluster network. If no network information is available, or the existing VPCs are not suitable, please create a VPC in the console. For details, see Overview. The subnet of the VPC will be used as the container network of the Serverless cluster.

- **Super node configuration**:
  - **Availability zone**: Select the availability zone where the super node is located.
  - **Billing mode**: The pay-as-you-go billing mode is supported in all regions.
  - **Container network**: Assign IPs within the container IP range to containers in the cluster. Pods on super nodes will occupy IPs in the subnet of the VPC. The number of Pods can be scheduled to super nodes are limited by the number of remaining IPs. Please select subnets with sufficient available IPs and without conflict with that of other services. Pods on super nodes will run in the specified VPC. Each Pod is bound to an ENI in the specified VPC. You can check the ENI associated with the Pod in the ENI list.
    Multiple subnets can be selected in the pay-as-you-go billing mode, and a pay-as-you-go super node is created for a subnet. The super nodes are billed based on the Pod specifications and running duration after the workloads are created and the Pods are scheduled to the super nodes.

    Note
    - We recommend that you configure multiple availability zones for the container network so that your workloads can be automatically distributed to multiple availability zones, which improves usability.
    - Ensure that the subnet assigned to the container network has sufficient available IPs, so as to prevent pod creation failure caused by insufficient IPs when creating a large-scale workload.

- **Service CIDR block**: The ClusterIP Service of the cluster defaults to be assigned in the selected VPC subnet. Please select a subnet with sufficient available IPs and without conflict with that of other services.

- **Cluster description**: Specify the information about the cluster to be created. This description will be displayed on the **Cluster information** page.

- **Advanced settings**:
  - CoreDNS: Two replica nodes of `Deployment:coredns` are automatically deployed in the cluster namespace `kube-system`. This service is free of charge, and we recommend that you do not modify it.
  - Tag: Tencent Cloud tags, which are used to manage resources by category from different dimensions.

5. Click **Complete** to start creating the cluster. You can view the creation progress on the “Cluster” page.

Connecting to a Cluster

Last updated：2023-03-10 16:19:09

This document describes how to connect a local Client to an TKE Serverless cluster through kubectl, which is the Kubernetes command-line tool.

Prerequisites

- Install cURL software program.
- Select an appropriate way to obtain kubectl based on the OS type:

> Note
> Replace "v1.18.4" in the command line with the kubectl version required by your business based on actual needs. Generally, the version of kubectl on the Client should be consistent with the latest version of Kubernetes on service end. You can view the K8s version on the **Basic Information** section of the **Basic Information** page.

  - Mac OS
  - Linux
  - Windows

Run the following command to obtain kubectl:

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

Directions

Installing kubectl

1. Install kubectl as instructed in Installing and Setting up kubectl.

> Note
> - If you have already installed kubectl, skip this step.

- This step uses the Linux OS as an example.

2. Run the following command to grant permissions to use kubectl.

```
chmod +x ./kubectl
```

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

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

```
kubectl version
```

If the output is similar to the following version information, the installation was successful.

```
Client Version: version.Info{Major:"1", Minor:"5", GitVersion:"v1.5.2", GitComm
it:"08e099554f3c31f6e6f07b448ab3ed78d0520507", GitTreeState:"clean", BuildDat
e:"2017-01-12T04:57:25Z", GoVersion:"go1.7.4", Compiler:"gc", Platform:"linux/a
md64"}
```

Configuring kubeconfig

1. Log in to the TKE console and select **Cluster** in the left sidebar.
2. On the **Cluster** list page, click the ID of the target cluster that you want to connect to go to the management page of the Serverless cluster.
3. Click **Basic Information** in the left sidebar to go to the **Basic Information** page, as shown in the following figure.
4. In **Cluster APIServer information** section, enable the **Internet Access** or **Private Network Access**, and view information such as the access address, and kubeconfig access credential of the cluster.

- Access entry: configure the access entry as needed.
  - **Internet Access**: this option is disabled by default. Note that enabling public network access will expose the APIServer of the cluster to the public network. In addition, you need to configure source authorization. By default, access from all sources is rejected. You can allow access from a single IP address or CIDR block. It is strongly recommended that you not open the cluster to all sources by configuring 0.0.0.0/0.
  - **Private Network Access**: this option is disabled by default. To enable it, you need to specify the subnet of the APIServer for private network access. IP addresses are assigned from the configured subnet after private network access is successfully enabled.

- **Accessed URL**: APIServer address of the cluster. Note that this address cannot be copied and pasted to a browser for access.
- **Kubeconfig**: access credential of the cluster, which can be copied and downloaded.

5. Configure the cluster credential as needed. For more information, see **Connecting to Kubernetes cluster through Kubectl** in the console.

Accessing the Kubernetes cluster

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

```
kubectl config get-contexts
```

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

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

```
kubectl get pod
```

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

Notes

Introduction to the kubectl CLI

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

Cluster Lifecycle

Last updated：2023-03-27 11:08:16

Cluster Lifecycle Description

Status	Description
Creating	You are creating the TKE Serverless cluster and applying for cloud resources.
Running	The cluster is running properly.
Idle	The cluster is idle.
Activating	The cluster is changing from the idle status to the running status.
Deleting	You are deleting the cluster while terminating and releasing cloud resources.
Abnormal	An exception occurred in the cluster, such as an inaccessible network.

Idle Cluster Status Description

To make the most of idle cluster resources, we have launched the idle cluster repossession feature in TKE Serverless, which displays clusters without legacy Pods and Pod creation or deletion operations for seven consecutive days as "idle" clusters.

A cluster is considered idle if:

It has no legacy Pods created by users.

No Pods are created or deleted for seven consecutive days.

At least three days have passed since it changes from the idle status to the normal status.

Over seven days have passed since it is created.

An "idle" cluster retains all its information but becomes unavailable. You can't view cluster monitoring data and cluster details or perform API server operations. You can activate the cluster by clicking **More** > **Activate** in the **Operation** column or click **Delete** to delete it.

An activated idle cluster has the same features and configuration as it used to have. Note that if it has no Pods and Pod creation or deletion operations for three consecutive days after the activation, it will become idle again.

Notes

Last updated：2022-09-14 10:05:37

Billing Mode

TKE Serverless cluster adopts the pay-as-you-go billing method. For more information about TKE Serverless cluster billing, please see Billing Overview, Product Pricing, and Purchase Limits.

Pod Specification Configuration

Container runtime resources and bills depend on the Pod specification configuration. Please note the Pod specification configuration and specific methods supported by TKE Serverless clusters. For more information, please see Resource Specifications and Specifying Resource Specifications.

Pod Temporary Storage

When created, each Pod is allocated less than 20 GiB of temporary image storage.

Note：

- Temporary image storage will be deleted when the Pod lifecycle ends. Therefore, please do not store important data in it.
- Due to image storage, the actual available storage capacity is less than 20 GiB.
- It is recommended to mount important data and large files to Volume for persistent storage.

Kubernetes Version

Kubernetes v1.14 and earlier versions are not supported.

Notes

TKE Serverless clusters do not have nodes. Therefore, some features, which depend on node components such as Kubelet and Kube-proxy, are not supported.

Node

- Currently, adding or managing a physical node is not supported.
- TKE Serverless clusters nodes will be replaced by "supernodes", and each supernode corresponds to a specified VPC subnet in the container network.
- The supernodes also support scheduling operations such as node affinity, taint, and cordoning.
- The number of Pods that can be scheduled on a supernode is only subject to the number of IPs in the corresponding VPC subnet.

Container network

- The container network of an TKE Serverless cluster is a VPC which is at the same level as Tencent Cloud services such as CVM and TencentDB. Each Pod in the cluster occupies a VPC subnet IP.
- Pods can directly communicate through the VPC with Pods or other Tencent Cloud services in the same VPC, without performance loss.

Pod isolation

Pods in the TKE Serverless cluster and the CVM have the same security isolation. Pods are scheduled and created on the underlying physical server of Tencent Cloud, and resource isolation between Pods is guaranteed by the virtualization technology during the creation.

Kernel

Only the kernel parameters starting with "net" can be defined.

Workload

You cannot deploy workloads of the DaemonSet type through TKE Serverless cluster.

Service

You cannot deploy services of the NodePort type through TKE Serverless cluster.

In addition, for the ordinary Kubernetes cluster, the Service of ClusterIP type relies on nodes to forward traffic. To make the TKE Serverless cluster compatible with the Service of ClusterIP type, you need to specify another VPC subnet as the cluster's Service CIDR block. Each ClusterIP Service will create a private network CLB in this subnet, which will occupy one IP in the subnet, to forward traffic. Please ensure that the subnet has sufficient available IPs.

Volume

TKE Serverless cluster supports the volume of the Hostpath type. For more information, see Instructions for Other Storage Volumes.

TKE Serverless clusters may not necessarily meet your expectations because they don't have nodes, although TKE Serverless cluster is still compatible with Host-related parameters (such as `Hostpath`, `Hostnetwork: true`, and `DnsPolicy: ClusterFirstWithHostNet`, etc.) in the Pod. For example, you want to use Hostpath to share data, but the two Pods scheduled to the same supernode may be the Hostpath of different hosts. Therefore, we recommend that the tasks you run on the TKE Serverless cluster do not strongly depend on Host-related parameters.

Port Limits

Port 9100 are not available.

Super Node Management
Creating a Super Node

Last updated：2023-07-11 18:06:17

Super Node Introduction

Super nodes are created in TKE Serverless clusters for Pod scheduling. A super node is created in each subnet of the container network when you create a TKE Serverless cluster.

In the following scenarios, you can resolve the issues by creating super nodes:

- When you run a large-scale workload in a TKE Serverless cluster, a Pod cannot be created due to exhaustion of IPs in the subnet of availability zone where the service is located. It is recommended that you create super nodes in a new subnet to have more IPs.

- As the scale of services expands, services need to be automatically distributed in multiple availability zones. It is recommended that you create super nodes in a new subnet to expand availability zones of the resources.

- When you create a workload, Pod pending is caused by resource shortage for the availability zone. This may be reflected as the following cluster events:

  ```
  EVENT REASON : “FailedCreatePodSandBox”
  EVENT MESSAGE : “Failed to create pod sandbox in underlay (will retry): insuffi
  cient resource”
  ```

  You can create super nodes in other availability zones to expand resources available in the cluster.

Billing Overview

Fees are not charged for super nodes and the cost will be calculated based on the CPU, GPU, memory value and the running time of the workload. For details, see Billing Overview, Product Pricing, and Purchase Limits.

Overview

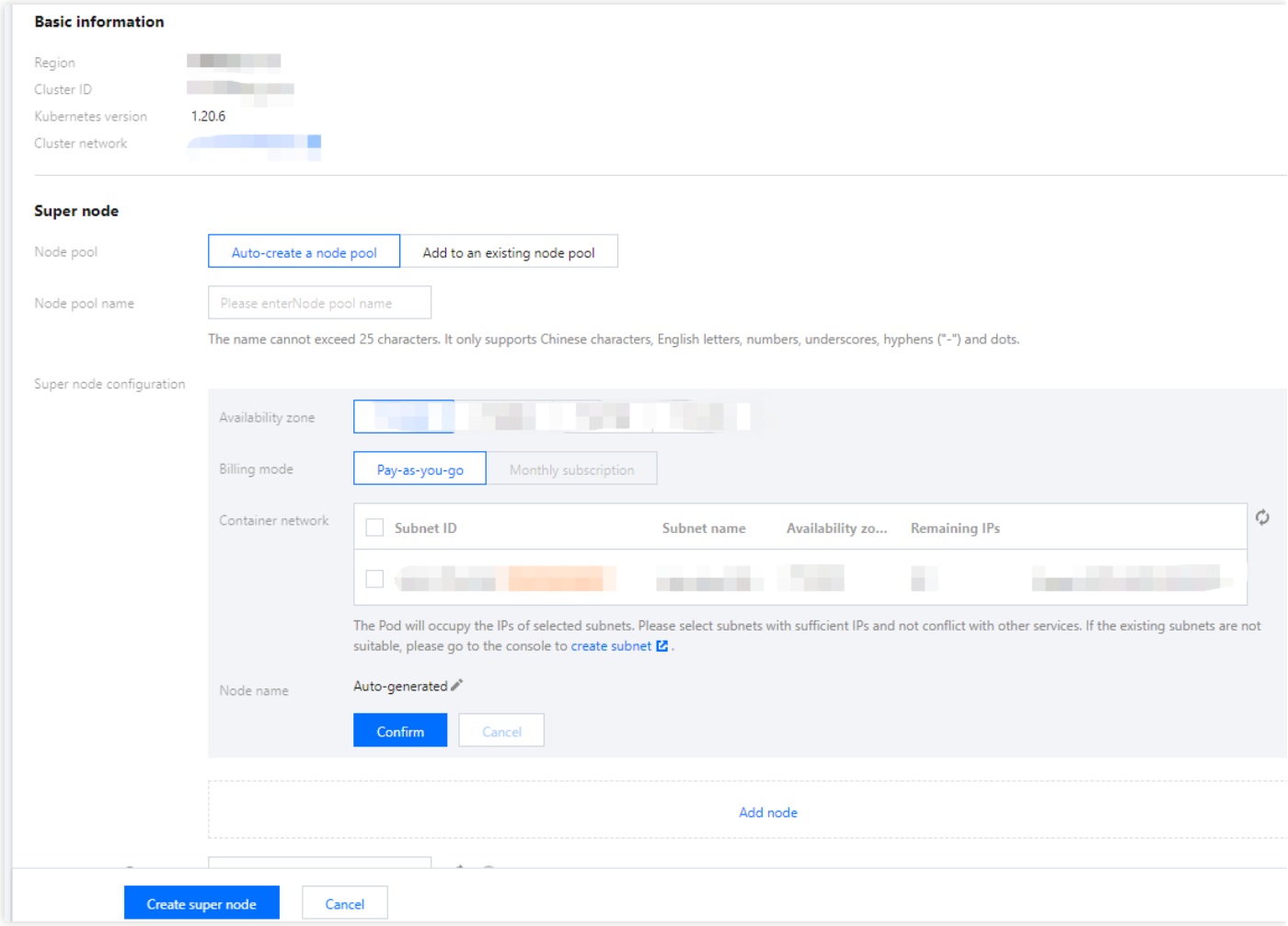
This document describes how to create a super node in a TKE Serverless cluster in the TKE console.

Prerequisites

- You have created a TKE Serverless cluster. For operation details, see Connecting to a Cluster.
- You have known Notes on Pod Scheduled to Super Nodes.

Directions

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the cluster management page, click the ID of the target Serverless cluster to enter the basic information page.
3. Click **Super node** in the left sidebar and click **Create**.
4. On the **Create super node** page, configure the parameters as needed.



- **Billing mode**: Only pay-as-you-go is available.

- **Operating system**: Select the operating system type (**Linux** or **Windows**) supported by the subnet. If you select **Windows**, Windows containers can run on the subnet. By default, **Linux** is selected.
- **Container network**: Specify the network allocated when Pods are scheduled to a super node. The Pods scheduled to the super node are on the same VPC as Tencent Cloud services such as CVM and TencentDB. Each Pod will occupy an IP address of the VPC subnet. You can select any subnet of the VPC where the Serverless cluster is located, as long as it has sufficient IP addresses to meet your needs.
- **Security group**: The security group works as a firewall to control access to the CVM network. For more information, see TKE Security Group Settings.

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

More

After the creation of the super node, please refer to Managing a Super Node for subsequent management.

Managing a Super Node

Last updated：2022-09-14 16:02:05

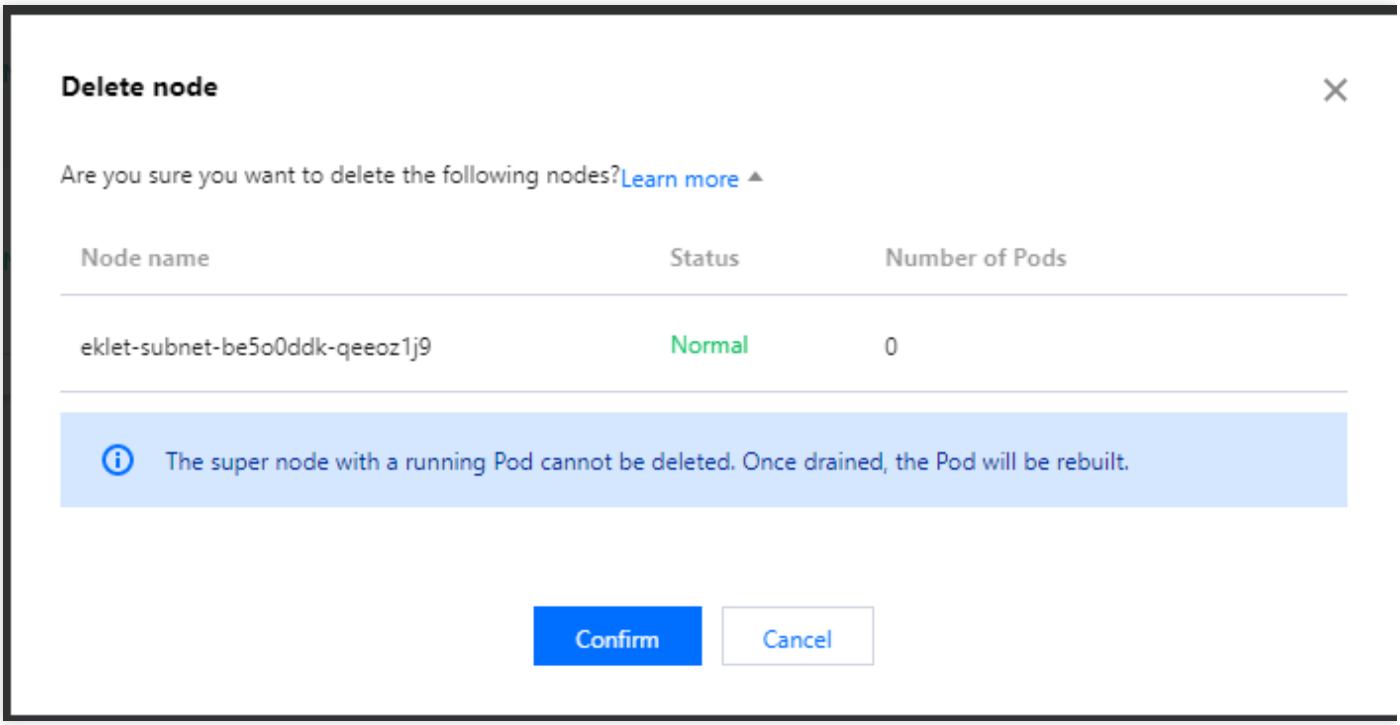
This document describes how to manage super nodes in the serverless cluster on the TKE console.

Prerequisites

- Please ensure the serverless cluster has been created.
- You have known Notes on Scheduling Pod to Super Node.

Deleting a super node

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the management page of clusters, click the serverless cluster ID to open the basic information page.
3. Click **Super Node** in the left sidebar to open **Super Nodes** list page.
4. On the list page, click **Remove** on the right side of the selected super node.
5. Click **OK** to delete the node on the **Delete Node** pop-up window.



Note

- At least one super node is required for the cluster. If there is only one super node, it cannot be deleted.

- The super node with a running Pod cannot be deleted. Please cordon and drain the node before remove it. Once drained, the Pod will be rebuilt.

Managing a super node

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the management page of clusters, click the serverless cluster ID to open the basic information page.
3. Click **Super Node" in the left sidebar to open **Super Nodes** list page.
4. Click the super node name to open its details page. You can manage the Pods on the super node, and view super node events, YAML and other information.

Modifying configuration of a super node

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the management page of clusters, click the serverless cluster ID to open the basic information page.
3. Click **Super Node** in the left sidebar to open **Super Nodes** list page.
4. Select the desired node, and click **Edit Label & Taint** on the right side of the super node.
5. In the **Modify Super Node Label & Taint** pop-up, modify the Label&Taint configuration of the super node.
6. Click **OK** to complete the modification.

Kubernetes Object Management
Workload Management

Last updated：2022-09-23 11:06:23

This document describes how to select a workload to run your applications in a TKE Serverless cluster.

Note：

- Serverless clusters do not have nodes. Workloads allocate resources for each Pod based on parameter settings when they are created. For more information, see Specifying Resource Specifications.
- To create and manage Kubernetes objects using a YAML file, specify annotations. For more information, please see Annotation Description.

Workload Types

Deployment

A Deployment declares the Pod template and Pod running policies. It is used to deploy stateless applications. You can specify the number of replicas, scheduling policy, and update policy for Pods running in the Deployment as needed.

StatefulSet

A StatefulSet is used to manage stateful applications. A Pod created by a StatefulSet has a persistent identifier that is created according to the specifications. The identifier will be retained after the Pod is migrated, terminated, or restarted. When using persistent storage, you can map storage volumes to identifiers. If your application does not require any persistent identifier, we recommend that you use a Deployment to deploy the application.

Job

A Job creates one or more Pods and ensures that these Pods run according to specified rules until they are terminated. Jobs can be used in many scenarios, such as batch computing and data analysis. You can specify the number of repeated running, concurrency, and restart policy as required.
After the Job stops running, it will not create new Pods and only the running records of original Pods exist. However, the real underlying resources have been released. Therefore, you cannot query the logs of the corresponding Pods in "Logs" section of the console.

CronJob

A CronJob object is similar to a line in a crontab (cron table) file. It periodically runs a Job according to the specified schedule. For more information about the format, see the Cron documentation.
The format of a Cron is as follows:

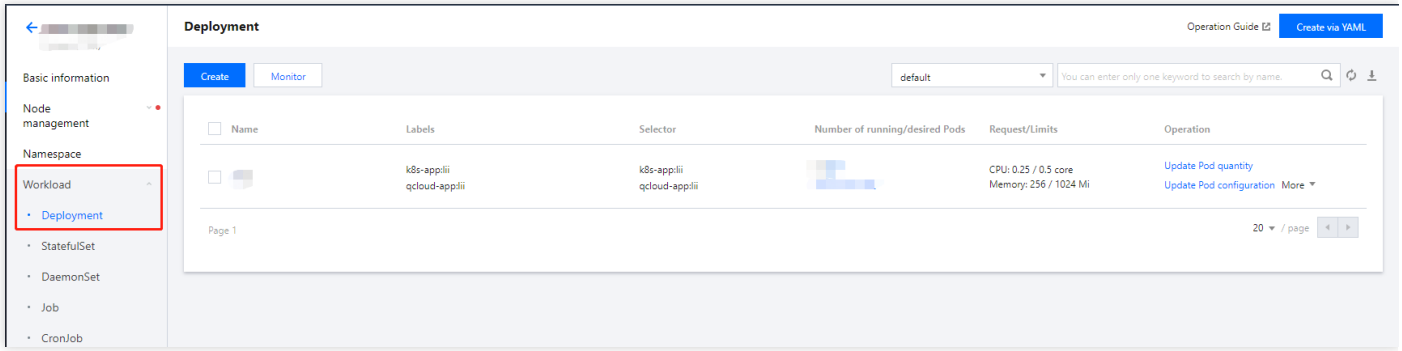
```
# File format description
# ┌──min (0 - 59)
# │ ┌──hour (0 - 23)
# │ │ ┌──day of month (1 - 31)
# │ │ │ ┌──month (1 - 12)
# │ │ │ │ ┌──day of week (0 - 6)
# │ │ │ │ │
# * * * * *
```

Prerequisite

- You have created a Serverless cluster that is in “Running” status. For more information, please see Creating a Cluster.
- The cluster has an appropriate namespace that is in Active status.

Directions

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the cluster management page, click the ID of the target Serverless cluster to enter the basic information page.
3. Select a type of workload in the left sidebar, and click **Create**.



4. On the “Create workload” page, enter a name for the workload and select the workload type.

- For more information about parameter settings for each workload type, see the following documents:
  - Deployment Management

- StatefulSet Management
- CronJob Management
- Job Management

- For the instructions on other operations, see the following documents:
  - Setting the Resource Limit of Workload
  - Setting the Scheduling Rule for a Workload
  - Setting the Health Check for a Workload
  - Setting the Run Command and Parameter for a Workload

Service Management

Last updated：2022-09-14 10:19:25

Introduction

This document describes how to use Service and Ingress as entry points to expose workloads to external sources.

Service Types

Service

Service defines policies for accessing backend Pods and provides a fixed virtual IP address for access. It also provides load balancing for all requests to Pods.

Service can be of the following types:

- Public network access: the public network access Service uses operates in Loadbalance and automatically creates a public network CLB instance. Public IP addresses can access backend Pods.
- Intra-cluster access: the intra-cluster access Service operates in ClusterIP mode and is used for access within the cluster.
- VPC private network access: the VPC private network access Service operates in Loadbalance and automatically creates a private network CLB instance. By using `annotations:service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx` , you can use a private IP address from the VPC private network to access the backend Pod.

> Note：
> When the service type is public network access, `ClusterIP` of this service is disabled by default. You can add the following annotations in yaml to enable `ClusterIP` :

```
service.kubernetes.io/qcloud-clusterip-loadbalancer-subnetid: #Subnet ID of the service CIDR
```

Ingress

Ingress is a collection of rules that allow access to Services of a cluster. You can configure different forwarding rules to allow different URLs to access different Services.

In order for Ingress resources to operate properly, you must run `Ingress-controller`. TKE enables the CLB-based `l7-lb-controller` by default and supports HTTP, HTTPS, and nginx-ingress controllers. You can select Ingress controllers according to your needs.

Prerequisites

- You have created an serverless cluster that is in the Running state. For more information, see Creating a cluster.
- The cluster has an appropriate namespace that is in the Active state.

Directions

For more information and instructions, see Service Management and Ingress Management.

Considerations

- Creating ClusterIP Service in an serverless cluster uses IP addresses from the Service CIDR. Make sure there are enough IP addresses in the subnet.
  In an serverless cluster, a CLB instance created by the Service binds all ENIs of all Pods within the endpoint.
- In an serverless cluster, Services only supports CLB instances.
- To create a Service using an existing CLB, you must ensure that the CLB is not bound to any listener.

Specifying resource specifications

Last updated：2022-09-13 16:50:43

TKE Serverless Cluster specifies the maximum resources allocated to a pod using annotation specification or automatic Request and Limit calculation. You can select either method.

Specifying by Annotation

TKE Serverless Cluster can add `template annotation` in the YAML file of a workload to explicitly specify the pod resource specifications. For more information, see Annotation Description.

Automatically Calculating by Request and Limit

TKE Serverless Cluster can calculate the Request and Limit parameters set for a workload to determine the resources required for running pods. The calculation method varies depending on the pod resource type. For more information on how to automatically calculate specified resource specifications based on the Request and Limit parameters, see CPU specification calculation methods for pods and GPU specification calculation methods for pods.

Note：

- If `template annotation` is specified for a workload, the annotation configuration prevails and the Request and Limit parameters are not calculated.
- For more information about Request and Limit resource allocation, see the supported CPU and GPU specifications in Resource Specifications. If the set value varies greatly from the supported specifications, the allocation of a resource may exceed expectations, resulting in resource waste.
- Regardless of how Request and Limit are set, the final calculation result will match with that in Resource Specifications, and resources allocated to a pod will not exceed the allowed specifications.
- If Request and Limit are not set for a container in a pod, the Request and Limit values of the container are regarded as 0.
- If Request and Limit are not set for all containers in a pod, the default pod specifications are 1 core and 2 GiB.
- Initcontainer and Container are calculated based on the following methods, and the larger value is used.

CPU specifications calculation methods for pods

Step 1. Calculate the total CPU and memory value of a pod.

The total values are the **total Request value of all containers in a pod** and the **maximum Limit value of containers in a pod**, respectively.

Step 2. Match pod resource specifications based on the following table.

Total CPU and Memory Values	Pod Resource Selection Rules
The total CPU and memory values are both 0.	The pod specifications are 1 core and 2 GiB.
Either the total CPU value or total memory value is 0.	Match the minimum value based on the non-0 total value. For example, if the total CPU value is 0 cores, and the total memory value is 8 GiB, match the minimum CPU value in allowed specifications with 8 GiB memory. The selected pod specifications are 1 core and 8 GiB.
Neither the total CPU value nor total memory value is 0.	Match resource specifications in Resource Specifications. First, select a higher specification (specification A) with a CPU value the same as or similar to the total CPU value. Then, select a higher specification with a memory value similar to the total memory value. • If the total memory value is less than the minimum memory value in the memory range of specification A, select the minimum memory value in the memory range of specification A. • If the total memory value is greater than the maximum memory value in the memory range of specification A, select a higher specification (specification B) with a memory value similar to the total memory value and change the total CPU value to that of specification B. • If the total memory value is within the memory range of specification A, select the nearest larger dual-value.
Either the total CPU value or memory value exceeds the maximum specification.	An error occurs, and resource matching fails.

Sample

You can better understand the CPU specification calculation methods for pods based on the following examples.

- Example 1
- Example 2

```
resources:
limits:
cpu: "1"
memory: 2Gi
requests:
cpu: "1"
memory: 2Gi
```

Result: The selected pod specification is 1 core and 2 GiB.

GPU specification calculation methods for pods

Note：

- Typically, GPUs have the same `nvidia.com/gpu` parameter value as vGPUs, and the value must be an integer.
- vGPU can be regarded as an independent GPU type. For example, 1/4*V100 indicates that 1/4 the computing power of a V100 GPU card is virtualized to a complete card. During resource allocation, one GPU card is applied for, that is, nvidia.com/GPU is 1.

Step 1. Calculate the total GPU value of a pod.

The total GPU value is the **total Request value of all containers in a pod**.

Step 2. Match pod resource specifications based on the following table.

Total CPU, Memory, and GPU Values	Pod Resource Matching Rules
The total values must comply with specification requirements, for example, 1, 2, 4, and 8.	First select a higher specification (specification A) with a GPU value the same as or similar to the total GPU value. Then, calculate the CPU and memory values based on the CPU specification calculation methods for pods to obtain the CPU specification (specification B). • If the CPU and memory values of specification A are greater than or equal to those of specification B, select the GPU value of specification A. • If the CPU and memory values of specification A are less than those of specification B, select a higher GPU specification (specification C) with CPU and memory values similar to those of specification B. In this method, the allocated number of GPU cards are more than that needed, which should be avoided. **To prevent waste**, lower the CPU and memory request values.
Any total value	An error occurs, and resource matching fails.

exceeds the maximum specifications.	

Sample

You can better understand the GPU specification calculation methods for pods based on the following examples.

- Example 1
- Example 2

```
## eks.tke.cloud.tencent.com/gpu-type: V100
resources:
limits:
cpu: "8"
memory: 32Gi
nvidia.com/gpu: "1"
requests:
cpu: "4"
memory: 16Gi
nvidia.com/gpu: "1"
```

Note:

Total GPU value: 1

Total CPU value: max(4,8) = 8 cores

Total memory value: max(16,32) = 32 GiB

Result: 8 cores and 32 GiB are less than the CPU and memory values (8 cores and 40 GiB) of the V100 GPU specification (one card) in Resource Specifications. The ultimately selected pod specification is 8 cores, 40 GiB, and 1x V100.

Other Resource Management

Last updated：2022-09-14 10:28:24

This document describes how to manage other Kubernetes resources in the TKE console, including namespaces, configuration, and storage.

Prerequisites

An serverless cluster has been created and is in the Running state. For more information, see Creating a Cluster.

Directions

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the **Cluster Management** page that appears, click the ID of the serverless cluster.
3. On the details page of the cluster, you can manage other resources according to the following documents:

- To manage a namespace, see Namespace.
- To manage Horizontal Pod Autoscaler (HPA), see HPA and HPA Metrics.
- To manage configuration resources, see ConfigMap Management and Secret Management.
- To manage storage resources, see Storage Management Overview.

Annotation

Last updated：2023-02-01 16:10:50

This document describes the annotation that is unique to super nodes and effective for Pods running on super nodes in TKE general and Serverless clusters.

Annotation Usage

Adding a Pod annotation to a workload

Annotations in this document are at the Pod level. Usually, it is the workload not bare Pod that is used by users. This document describes how to add a Pod annotation to a Deployment. Note that a Pod annotation is added in the `.spec.template.metadata.annotations` field of a workload.

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx
spec:
replicas: 1
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
annotations:
eks.tke.cloud.tencent.com/retain-ip: 'true' # A Pod annotation is added in the `.
spec.template.metadata.annotations` field of a workload.
spec:
containers:
- name: nginx
image: nginx
```

Global configuration

You can modify the global configuration to make an annotation effective for all Pods in a cluster by default, that is, the `eks-config` ConfigMap in the `kube-system` namespace. If there are no such configurations, create one.

```
apiVersion: v1
kind: ConfigMap
metadata:
name: eks-config
namespace: kube-system
data:
pod.annotations: |
eks.tke.cloud.tencent.com/resolv-conf: |
nameserver 183.60.83.19
eks.tke.cloud.tencent.com/host-sysctls: '[{"name": "net.core.rmem_max","value":
"26214400"}]'
```

Note：
Annotations added to Pods take priority over the global configuration.

Resources and Specifications

Specifying the CPU and memory

By default, Pods running on super nodes automatically calculate the specifications of underlying resources according to the request and limit values. You can also specify the computing resource specifications required by Pods by adding Pod annotations. For more information, see Specifying resource specifications.

```
eks.tke.cloud.tencent.com/cpu: '8'
eks.tke.cloud.tencent.com/mem: '16Gi' # The memory needs to be measured in Gi. If
G is used, parameter errors will be reported.
```

Specifying system disk size

A Pod running on a super node provides 20 GB of free system disk size by default. The system disk has the same lifecycle as the Pod, meeting the requirements for system disk resources in general scenarios. If you need a larger system disk size, you can use the annotation to declare the desired size. Note that the excessive part will be billed based on the pay-as-you-go published price of a CBS Premium Cloud Storage cloud disk. For billing details, see Pricing | Cloud Block Storage. Below is a sample annotation of the system disk size:

```
eks.tke.cloud.tencent.com/root-cbs-size: '50' # Specify the system disk size. Add
itional charges are applied for the part of the size exceeding 20 Gi
```

Note：
For detailed instructions on how to adjust the disk size when using the image cache, see Annotation.

Automatically upgrading the specification

Pods on super nodes automatically calculate the specifications of underlying resources according to the request and limit values. If no such resources exist, Pod creation will fail and the system will prompt that the resources are insufficient. If you agree to create a Pod with higher-specced resources, add the following annotation to the Pod to enable automatic specification upgrade. Fees will be charged according to the upgraded specifications.

```
eks.tke.cloud.tencent.com/spec-auto-upgrade: 'true' # When resources are insuffic
ient, enable automatic specification upgrade, which is performed only once accord
ing to the CPU specifications.
```

Specifying the GPU

To specify the GPU, add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/gpu-type: 'T4,V100' # Specify the GPU model by priorit
y. If you use the 1/4 T4 vGPU, specify it as 1/4*T4.
```

Note：
You need to set the number of cards and specification of the GPU in `Request` and `Limit` based on your GPU model. For more information, see Resource Specifications.

Specifying the CPU type

To specify the CPU type, add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/cpu-type: 'amd,intel' # It indicates that AMD resource
Pods are created first. If the AMD resources in the AZ of the selected region are
insufficient, Intel resource Pods are created.
```

Enabling the spot mode

To enable the spot mode, add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/spot-pod: 'true'
```

IP Retention and EIP

Fixing an IP in StatefulSet

You can use a fixed IP if the workload is StatefulSet or a bare Pod is used (note that you cannot change the Pod name). You can add the Pod annotation to enable the fixed IP:

```
eks.tke.cloud.tencent.com/retain-ip: 'true' # Set the value to `true` to enable t
he fixed IP.
eks.tke.cloud.tencent.com/retain-ip-hours: '48' # The maximum IP retention period
in hours. If a terminated Pod is not created after this period, the IP will be re
leased.
```

Note：
You don't need to add annotations to use the fixed IP for super nodes in the TKE cluster.

Binding an EIP

To bind an EIP, add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/eip-attributes: '{"InternetMaxBandwidthOut":50, "Intern
etChargeType":"TRAFFIC_POSTPAID_BY_HOUR"}' # The value can be an empty string, in
dicating that the EIP is enabled and the default configuration is used. You can a
lso use the JSON parameter used to create the EIP API. For more information on th
e parameter list, visit https://cloud.tencent.com/document/api/215/16699#2.-.E8.B
E.93.E5.85.A5.E5.8F.82.E6.95.B0. In this example, the parameter indicates that th
e EIP is pay-as-you-go and the bandwidth cap is 50 Mbps.
```

Note：
An EIP cannot be bound for non-bill-by-IP accounts (traditional accounts). If you are using a non-bill-by-IP account, submit a ticket for account upgrade.

Fixing an EIP in StatefulSet

To fix an EIP in StatefulSet, add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/eip-attributes: '{}' # Enable the EIP and use the defau
lt configuration.
eks.tke.cloud.tencent.com/eip-claim-delete-policy: 'Never' # It indicates whether
```

```
to repossess the EIP after the Pod is deleted. By default, it is repossessed. `Ne
ver` indicates not to repossess, which means the same EIP will be bound to the ne
xt Pod created under the same name, thereby fixing the EIP.
```

Note：
When `Never` is set for Deployment workloads, the EIP will not be repossessed after the Pod is deleted or used for the roll-updated Pod.

Using an existing EIP in StatefulSet

To bind an existing EIP to the Pod instead of automatically creating one, specify the ID of the EIP instance to be bound to the Pod. The annotation is as follows:

```
eks.tke.cloud.tencent.com/eip-id-list: 'eip-xx1,eip-xx2' # Specify the list of ex
isting EIP instances and make sure that the number of Pod replicas in StatefulSet
is less than or equal to that of EIP instances.
```

Note：
When specifying the EIP, do not configure the `eip-attributes` annotation.

Image and Registry

Ignoring the certificate verification

If the certificate of the self-built image registry is self-signed, the image pull will fail (ErrImagePull). You can add the following annotation to the Pod to specify not to verify the certificate when images are pulled from the image registry:

```
eks.tke.cloud.tencent.com/registry-insecure-skip-verify: 'harbor.example.com' # Y
ou can write multiple ones separated by comma.
```

Using the HTTP protocol

If HTTP rather than HTTPS is used by the self-built image registry, the image pull will fail (ErrImagePull). You can add the following annotation to the Pod to use HTTP when images are pulled from the image registry:

```
eks.tke.cloud.tencent.com/registry-http-endpoint: 'harbor.example.com' # You can
write multiple ones separated by comma.
```

Reusing an image

By default, the system disk is reused on super nodes to speed up the startup, specifically, the system disk of the Pod within the caching period (two hours after termination) in the same AZ of the same workload. To reuse Pod images in different workloads, add the same `cbs-reuse-key` annotation to all of them:

```
eks.tke.cloud.tencent.com/cbs-reuse-key: 'image-name'
```

Image cache

Super nodes provide image cache capabilities, where an image cache instance is created in advance, the target image is automatically downloaded, and the cloud disk snapshot is created. Then you can enable the image cache when creating a Pod. The snapshot of the image cache instance is automatically matched by image name, allowing you to use the snapshot image content and saving the need to download the image a second time, thereby speeding up the startup.

The annotation to enable the image cache is as follows:

```
eks.tke.cloud.tencent.com/use-image-cache: 'auto'
```

Specify the type of cloud disk created with the image cache:

```
eks.tke.cloud.tencent.com/image-cache-disk-type: 'CLOUD_SSD' # Specify the type o
f cloud disk created with the image cache. Valid values: `CLOUD_BASIC` (HDD cloud
disk), `CLOUD_PREMIUM` (Premium Cloud Disk, it is the default value), `CLOUD_SSD`
(SSD), `CLOUD_HSSD` (Enhanced SSD), `CLOUD_TSSD` (ulTra SSD).
```

Specify the size of cloud disk created with the image cache:

```
eks.tke.cloud.tencent.com/image-cache-disk-size: '50' # Specify the size of cloud
disk created with the image cache. The default size is the one set when the cloud
disk was created. The size can only be increased, but cannot be decreased.
```

You can choose to manually specify the image cache instance instead of having it automatically matched:

```
eks.tke.cloud.tencent.com/use-image-cache: 'imc-xxx'
```

The data disk created by the image cache is terminated upon Pod deletion, but you can specify a period to retain it by using an annotation. When retention is enabled, the retained data disk will be reused when the next Pod is created, saving the time to roll the data from the snapshot to the new cloud disk:

```
eks.tke.cloud.tencent.com/image-cache-disk-retain-minute: '10' # Specify to retai
n the data disk created by the image cache for 10 minutes after Pod termination.
```

Binding a Security Group

By default, super nodes bind Pods to the `default` security group in the default project in the same region. You can also specify a security group by adding the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/security-group-id: 'sg-id1,sg-id2' # Enter the IDs of t
he security groups in the region and separate them by comma. Network policies tak
e effect based on the sequence of security groups. By default, a security group c
an be bound to up to 2,000 Pods. To increase this limit, submit a ticket for appl
ication.
```

Binding a Role

By default, super nodes associate Pods with the `TKE_QCSLinkedRoleInEKSLog` role and grant log collection components in the Pods the permission to report logs. You can associate Pods with other CAM roles by using the annotation to get permissions to manipulate Tencent Cloud resources.

```
eks.tke.cloud.tencent.com/role-name: 'TKE_QCSLinkedRoleInEKSLog'
```

Setting the Host Kernel Parameters

Some kernel parameters are isolated by network namespace and cannot be viewed or set in containers. A Pod on the super node exclusively occupies a VM, but this doesn't mean that you can set host kernel parameters in containers.

You can add Pod annotations to set host kernel parameters:

```
eks.tke.cloud.tencent.com/host-sysctls: '[{"name": "net.core.rmem_max","value":
"26214400"},{"name": "net.core.wmem_max","value": "26214400"},{"name": "net.core.
rmem_default","value": "26214400"},{"name": "net.core.wmem_default","value": "262
14400"}]'
```

Loading the Kernel Module

To load the TOA kernel module, add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/host-modprobe: 'toa'
```

Automatic Recreation and Failover

An agent inside the VM in the cluster of the super node reports heartbeats to the control plane. Generally speaking, if the report times out (five minutes by default), the process in the Pod fails due to a high load. In this case, the cluster migrates the VM by default for failover (the current VM is shut down and a new one is created automatically to take over the Pod).

If you don't want to have another one automatically created (so that the problematic environment can be retained), then add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/recreate-node-lost-pod: "false"
```

To remove the abnormal Pod traffic in less time than the default five minutes, add the following annotation to the Pod to customize the heartbeat timeout period:

```
eks.tke.cloud.tencent.com/heartbeat-lost-period: 1m
```

Disk Cleanup

When the disk usage on the super node becomes too high, a cleanup process is automatically triggered to release the space. You can view the disk usage through `df -h` .

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.

Cleaning up a container image

By default, if the disk usage reaches 80%, the container image is automatically cleaned up to free up the space. If no container images can be released, the following event is reported:

```
failed to garbage collect required amount of images. Wanted to free 7980402688 by
tes, but freed 0 bytes
```

To customize the cleanup threshold, add the following annotation to the Pod:

```
eks.tke.cloud.tencent.com/image-gc-high-threshold: '80' # When the disk usage rea
ches 80%, image cleanup is triggered.
eks.tke.cloud.tencent.com/image-gc-low-threshold: '75' # After triggered, the con
tainer image cleanup stops when 5% (high-threshold - low-threshold) of the space
is released.
eks.tke.cloud.tencent.com/image-gc-period: '3m' # The disk space is checked once
every three minutes by default.
```

Cleaning up an exited container

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

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

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

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

Restarting the 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" # This feature is enabled
after set. It is not enabled by default.
```

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

Note：
This feature was launched on April 27, 2022 (UTC +8). The Pods created before that date must be rebuilt to enable the feature.

Monitoring Metrics

9100 port issue

Pods on super nodes expose monitoring data via port 9100 by default, and you can access 9100/metrics to get the data by running the following command:

- Get all metrics:

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

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

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

If the business listens on port 9100, the following error will be reported, indicating that port 9100 has been used:

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

You can customize the port number to be used for exposing monitoring data to avoid business conflicts. The configuration method is as follows:

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

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

Monitoring data reporting frequency

The `cAdvisor` monitoring data exposed on the super node is refreshed once every 30s by default. You can adjust the refresh frequency with the following annotation:

```
eks.tke.cloud.tencent.com/cri-stats-interval: '30s'
```

Customizing the monitoring metric path

By default, monitoring data is exposed through the `/metrics` path, which doesn't need to be changed. To customize it, use the following annotation:

```
eks.tke.cloud.tencent.com/custom-metrics-url: '/metrics'
```

Naming the Pod ENI

By default, a Pod uses the `eth0` ENI. To rename the ENI, add the following annotation:

```
internal.eks.tke.cloud.tencent.com/pod-eth-idx: '1' # Name the ENI `eth1`.
```

Custom DNS

By default, the host of the Pod uses the DNS of `183.60.83.19` and `183.60.82.98` in the VPC. To modify it, configure the following annotation:

```
eks.tke.cloud.tencent.com/resolv-conf: |
nameserver 4.4.4.4
nameserver 8.8.8.8
```

Note：
A custom DNS on the host prevails.

Collecting Logs to Kafka

Super nodes allow you to use open-source log collection components to collect logs to Kafka. If files in containers are collected or multi-line merging is used during collection, you need to properly configure the line size that defaults to 2 MB. If it exceeds 2 MB, the log in the line is discarded. The line size can be adjusted with the following annotation:

```
internal.eks.tke.cloud.tencent.com/tail-buffer-max-size: '2M' # A single-line log with the maximum size of 2M is supported by default.
```

When logs are collected to Kafka, the field of the log content is `log`. To configure it to `message`, use the following annotation:

```
eks.tke.cloud.tencent.com/log-key-as-message: 'true'
```

When reported, logs contain the Pod metadata information, which is put in the `kubernetes` field as a `string`. To set it to the `json` format, configure the following annotation:

```
eks.tke.cloud.tencent.com/filebeat-metadata-format: 'true'
```

Delaying the Termination

When a job running on the super node ends, the underlying resources along with the temporary output are terminated. If you run kubectl logs , "can not found connection to pod" will be reported. To retain underlying resources for problem locating, delay the termination with the following configuration:

```
eks.tke.cloud.tencent.com/reserve-sandbox-duration: '1m' # Enable the feature to delay the termination for one minute. When the last container of the Pod in the `Failed` status exits, the underlying resources are retained for one minute.
eks.tke.cloud.tencent.com/reserve-succeeded-sandbox: 'false' # Termination delay only applies to Pods in the `Failed` status. You can also change the field to apply it to Pods in the `Succeeded` status.
eks.tke.cloud.tencent.com/reserve-task-shorter-than: '5s' # If you only care about short-running jobs, you can configure this parameter. Then the termination delay will be triggered only when any container in the Pod runs shorter than the specified value. This parameter is not enabled by default.
```

Service Rule Sync

Disabling the service rule sync

The cluster of the super node generates K8s service rules through IPVS. If you don't need such service rules, you can disable them with the following annotation:

```
eks.tke.cloud.tencent.com/cluster-ip-switch: 'disable'
```

Note：
After service rules are disabled, service discovery in all clusters becomes invalid. `dnsPolicy` of `ClusterFirst` cannot be used by the Pod, and you need to change it to another type such as `Default`.

Waiting for the service rule sync

The cluster syncs service rules when the Pod is started. To wait for the sync to be completed before starting the Pod, use the following configuration:

```
eks.tke.cloud.tencent.com/duration-to-wait-service-rules: '30s' # Wait for the service rule sync to be completed before starting the Pod. The maximum value of 30s is set here.
```

Setting the IPVS parameter

IPVS rules can be controlled with the following annotation:

```
eks.tke.cloud.tencent.com/ipvs-scheduler: 'sh' # Scheduling algorithm. `sh` refers to source hash. Forwarding based on the source address hash facilitates distributed global loading balancing. The default value is `rr` (round robin).
eks.tke.cloud.tencent.com/ipvs-sh-port: "true" # Source hash is performed by port, which is valid only when `ipvs-scheduler` is `sh`.
eks.tke.cloud.tencent.com/ipvs-sync-period: '30s' # The maximum interval for refreshing rules, which defaults to 30s.
eks.tke.cloud.tencent.com/ipvs-min-sync-period: '2s' # The minimum interval for refreshing rules. By default, rules are refreshed upon service changes. You can modify this parameter to avoid frequent refreshes.
```

- Set not to guide the traffic through IPVS when the VIP of a CLB instance is accessed within the cluster.

  It applies to guiding traffic through CLB but not IPVS for access within the cluster. When the annotation is configured for the service, no corresponding IPVS rules will be generated:

  ```
  service.cloud.tencent.com/discard-loadbalancer-ip: 'true' # The annotation is configured for the service and takes effect immediately without Pod rebuild required.
  ```

Customizing the Pod Time Zone

By default, UTC time is used for Pods on super nodes. To adjust the time zone to UTC +8, add the following annotation:

```
eks.tke.cloud.tencent.com/host-timezone: 'Asia/Shanghai' # This annotation is used to set the Pod time zone to UTC +8.
```

Serverless Cluster Global Configuration Description

Last updated：2022-10-13 10:09:41

Overview

TKE Serverless cluster supports global configuration through ConfigMaps. In scenarios of automatic TKE super node scaling and a pure TKE Serverless cluster, if you need to batch set annotations for each super node or Pod, configuration at the super node or Pod level will be complicated and is highly intrusive to the business YAML file. Therefore, TKE Serverless cluster offers global configuration capabilities to allow you to perform global configuration through ConfigMaps, so as to add Annotations to each Pod in the cluster.

Directions

1. Create the `eks-config` ConfigMap under `kube-system` .
2. Set the parameters to make it take effect for all TKE Serverless cluster Pods.
   Below is the global configuration:

```
apiVersion: v1
kind: ConfigMap
metadata:
name: eks-config
namespace: kube-system
data:
pod.annotations: |
eks.tke.cloud.tencent.com/resolv-conf: |
nameserver 183.60.83.19
eks.tke.cloud.tencent.com/host-sysctls: "[{"name": "net.core.rmem_max","value":
"26214400"}]"
```

Configuration Priority Description

The global configuration has the lowest priority, next is the configuration at the super node level, and the Pod configuration has the highest priority. If configurations conflict, the one with the higher priority will take effect.

Mirror cache

Last updated：2023-05-19 16:27:16

Image Cache Overview

This document describes how image cache works and its billing rules, creation, and usage method. You can use image cache to accelerate image pull during instance creation so as to expedite instance startup. This capability is applicable to TKE Serverless cluster Pods and super nodes.

How it Works

When image cache accelerates instance startup, it will start a container instance in advance to pull the image, store the image in a data disk with a customizable size, and use the data disk as the cache of cloud disk snapshot, that is, the image data is already stored in the snapshot. You can choose to automatically or manually match the image cache when creating a container instance or Pod, and the specified number of data disks will be created based on the image snapshot and directly mounted to the instance to avoid image layer download, speeding up container instance and Pod creation.
Compared with the image reuse capability, image cache has the following strengths:
It has no time limit and is only subject to the lifecycle of the image cache (snapshot).
You only need to cold start a Pod in advance, which will be terminated immediately after snapshot creation.
It has no limit on AZs, as AZs will be automatically matched during cloud disk creation based on snapshot.
It has no limit on workloads, as workloads in the same region can be matched.

Billing Overview

The following resources are involved when you create an image cache, and their billing rules are as detailed below:

Billable Items	Billing Description	Documentation
Image cache	When you create an image cache, you need to run a 2C4G container instance to pull the image. After image cache creation, the container instance will be automatically released, and its billing will stop.	Product Pricing
CBS data disk	When you create an image cache, you need to bind a Premium Cloud Storage data disk to store the image. The disk size is	Price Overview

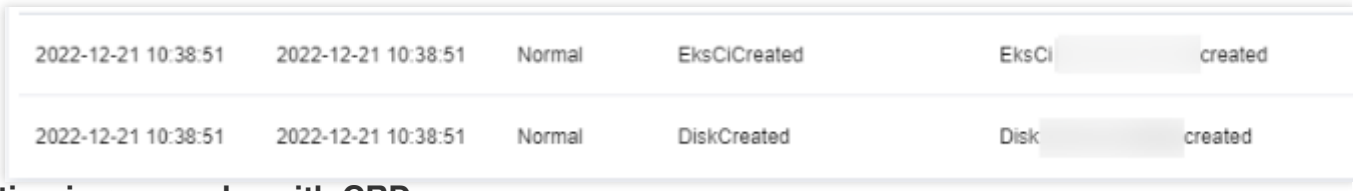
	customizable and 20 GB by default. After image cache creation, the data disk will be automatically released, and its billing will stop.	
Snapshot	A snapshot will be created based on the above-mentioned data disk, and its lifecycle will be the same as that of the image cache. It is billed by usage duration and capacity.	Price Overview

When the image cache is used, a Premium Cloud Storage data disk with the same capacity will be created based on the matched image cache snapshot and bound to the Pod. Therefore, in addition to the Pod creation fees, **data disk fees** will also be incurred.
Data disk fees = Capacity * Unit price * Instance running duration

Directions

Creating image cache in the console

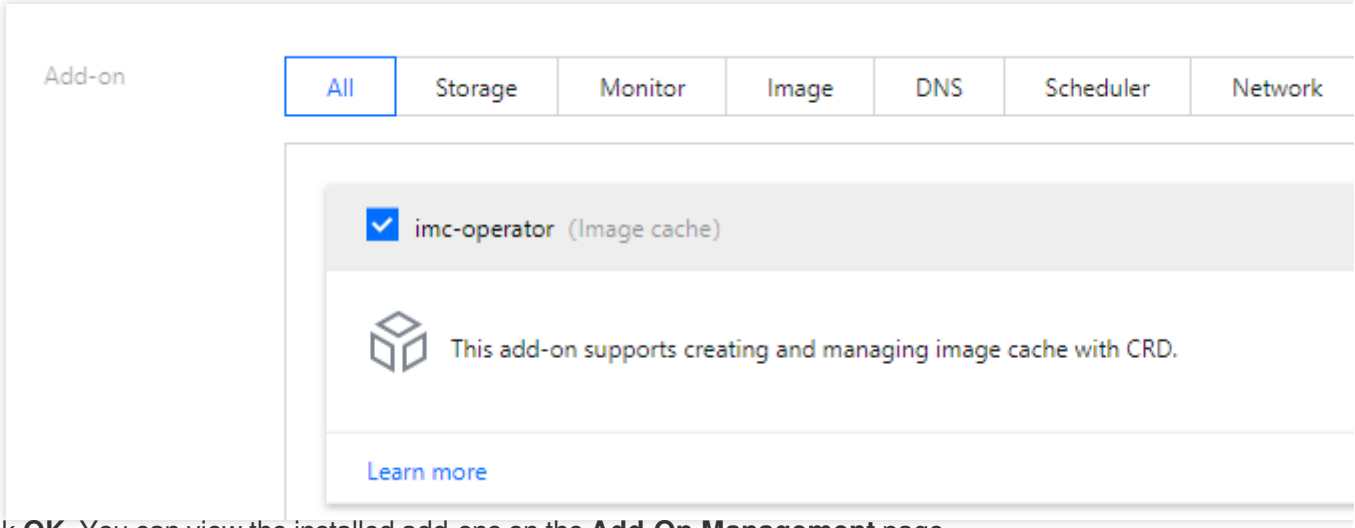
1. Log in to the TKE console.
2. Select **Application** > **Image Cache** in the left sidebar, On the **Image Cache** page that appears, click **Create instance**.
3. On the **Create image cache** page, configure relevant parameters.
Instance name: Custom.
Region: Select as needed.
Container network: Assign IPs within the container IP range to the container instances.
Security group: It has the capability of a firewall and can limit the network communication of the instance. Default value: `default` .
Operating system: Select **Windows** or **Linux**.
Image: Select the image and version to be cached as needed.
Image credentials: When you select Docker Hub or a private image in a third-party image repository, you must enter the image credentials, i.e., access address, username, and password of the repository.
Advanced settings:
Cache size: It determines the size of the snapshot and the data disk bound during instance creation.
Note:
The image size displayed in the image repository is the size of the compressed data. Creating an image cache involves pulling and expanding the image. If the image or image compression ratio is too large, the default 20 GB data disk will be insufficient, so a larger data disk space is recommended.
Expiration policy: Select the retention duration of the image cache, which is "Permanent" by default.
4. Click **Create instance**. After the image cache is created, it will be displayed in the image cache list. You can click the event name to view the creation progress.



Creating image cache with CRD

The image cache add-on needs to be installed in the cluster if you want to create an image cache with CRD. After the add-on is installed, you can use Tencent Cloud image cache with the method of CRD+Controller without the need to call the cloud API. The steps are as follows:

1. Log in to the TKE console.
2. On the cluster list page, click the ID of the target Serverless cluster to enter the cluster details page.
3. Select **Add-On Management** in the left sidebar and click **Create**.
4. On the **Create an Add-On** page, select **imc-operator (Image cache)** as shown in the figure below.

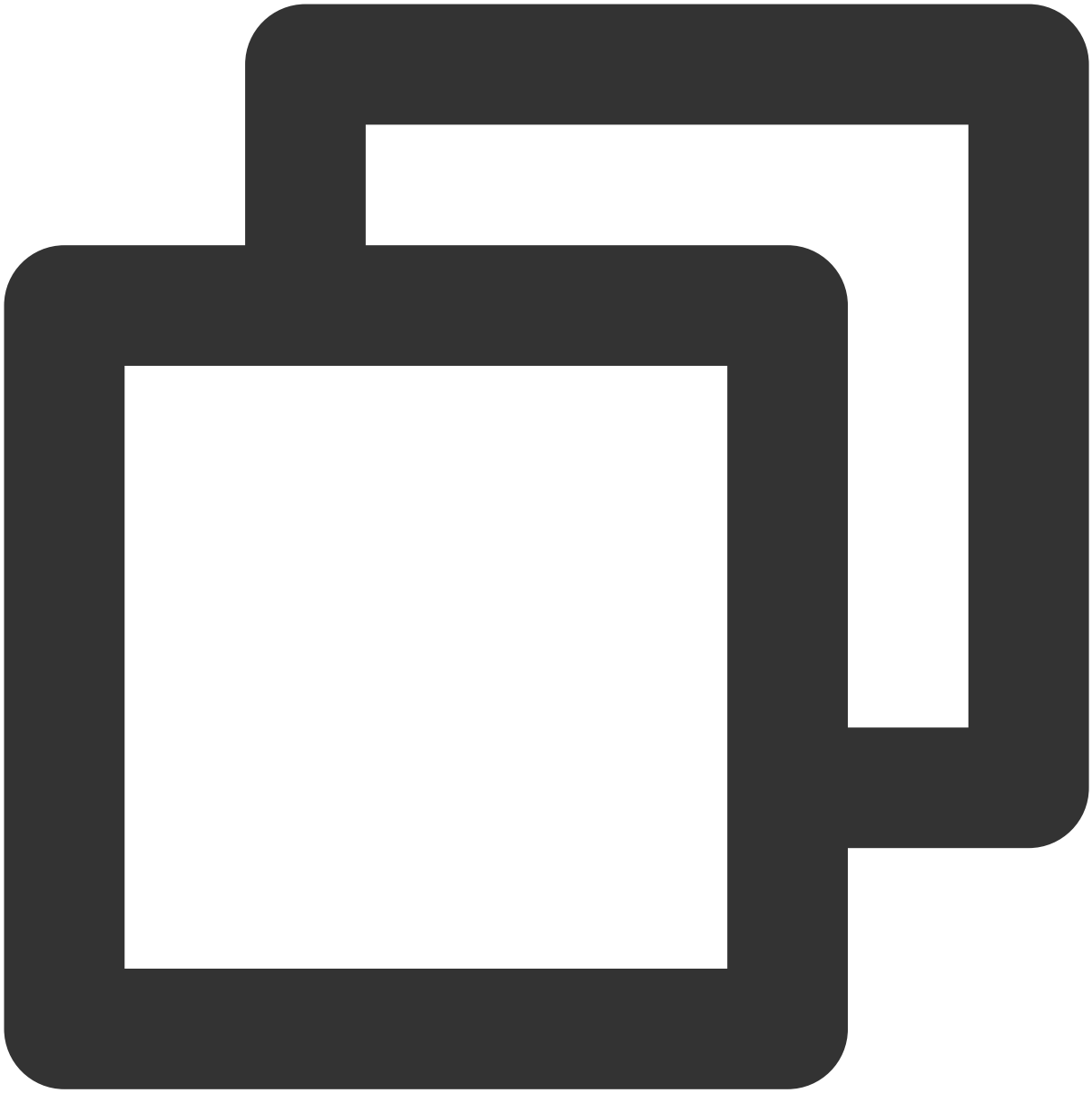


5. Click **OK**. You can view the installed add-ons on the **Add-On Management** page.
6. Edit YAML. Create an image cache as follows:

Creating an Image Cache

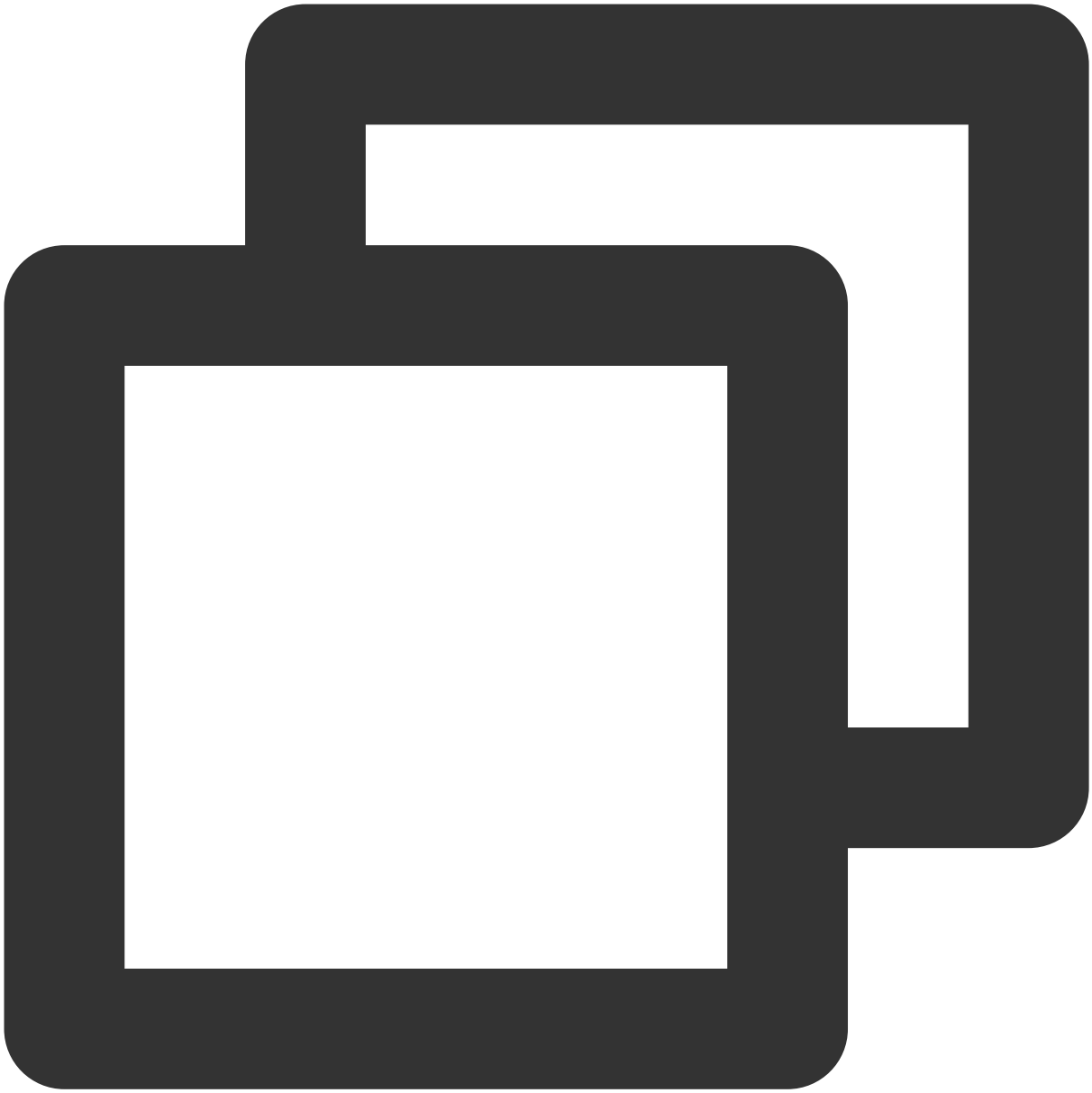
Checking an Image Cache

Example:



```
apiVersion: eks.cloud.tencent.com/v1
kind: ImageCache
metadata:
  name: imagecache-sample
spec:
  images:
    - nginx
#  imageCacheSize: 30
  # TODO(user): Add fields here
```

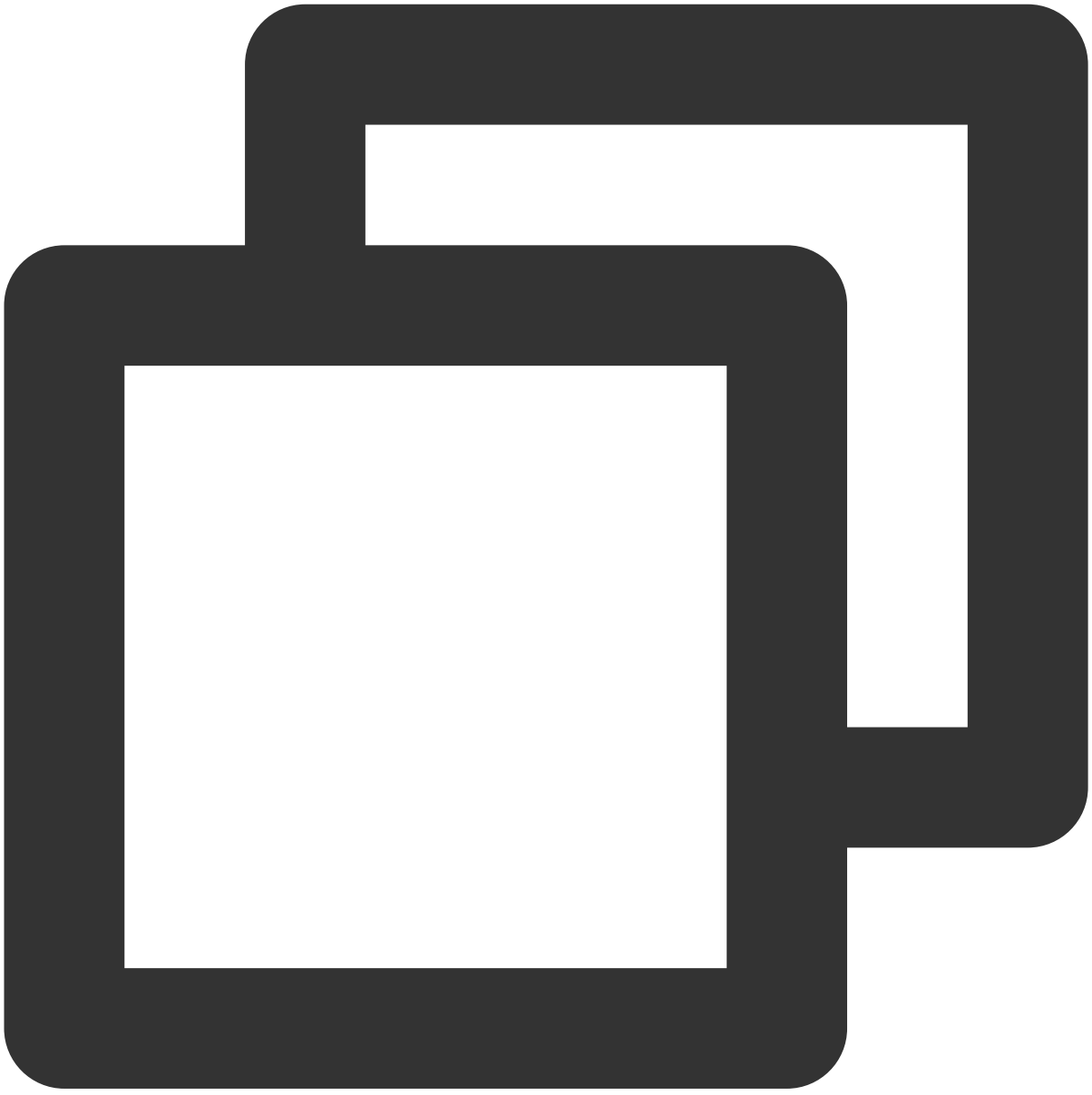
Sample with more parameters:



```
apiVersion: eks.cloud.tencent.com/v1
kind: ImageCache
metadata:
  annotations:
    "eks.tke.cloud.tencent.com/eip-attributes": '{"InternetMaxBandwidthOut":2}' # C
  name: imagecache-sample-more-para
spec:
  images:
    - nginx
    - mysql
  imageCacheSize: 30
```

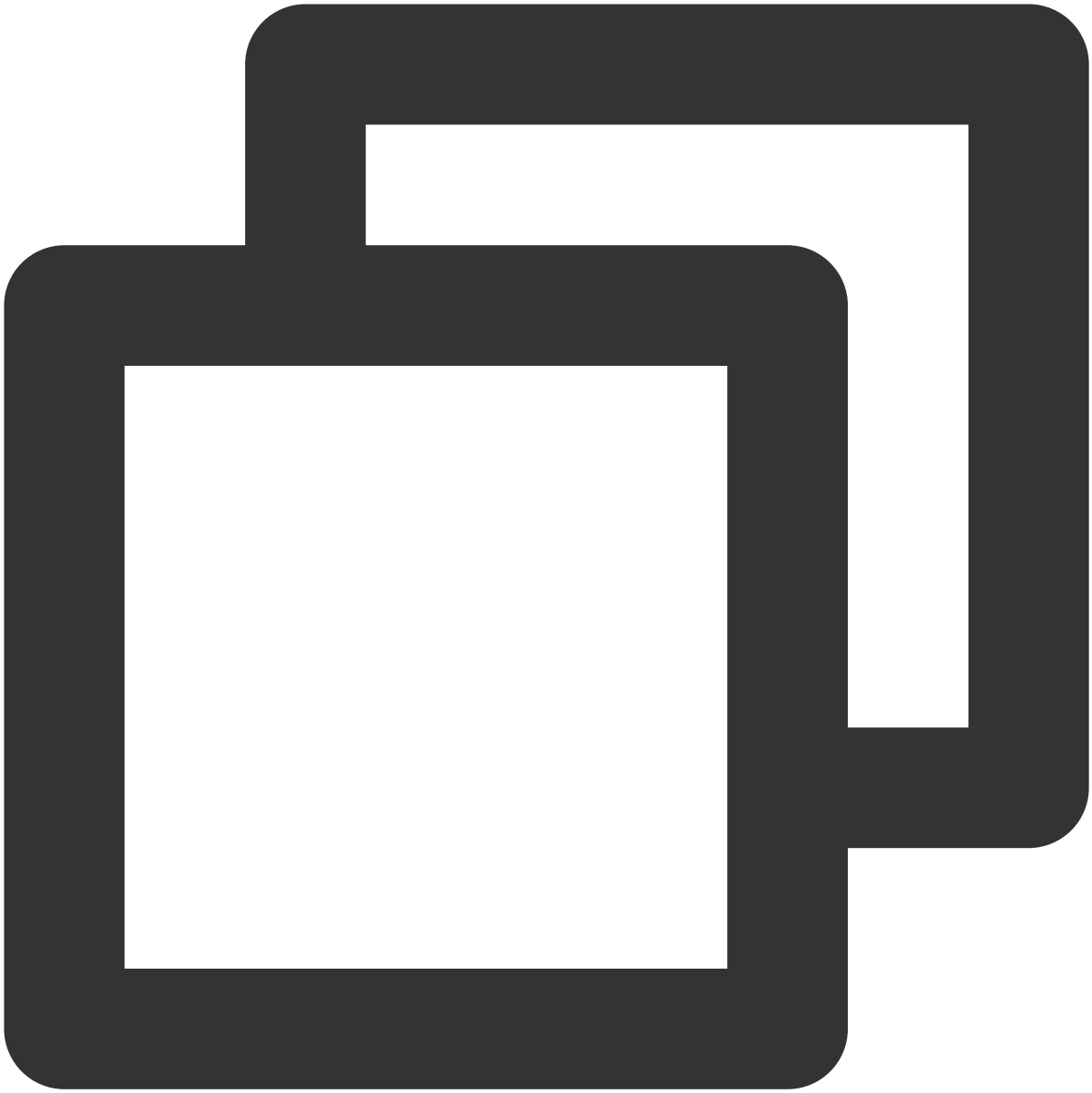
```
  retentionDays: 7
  imagePullSecrets:
    - imc-operator-system/qcloudregistrykey
```

Example:



```
kubectl get imc
```

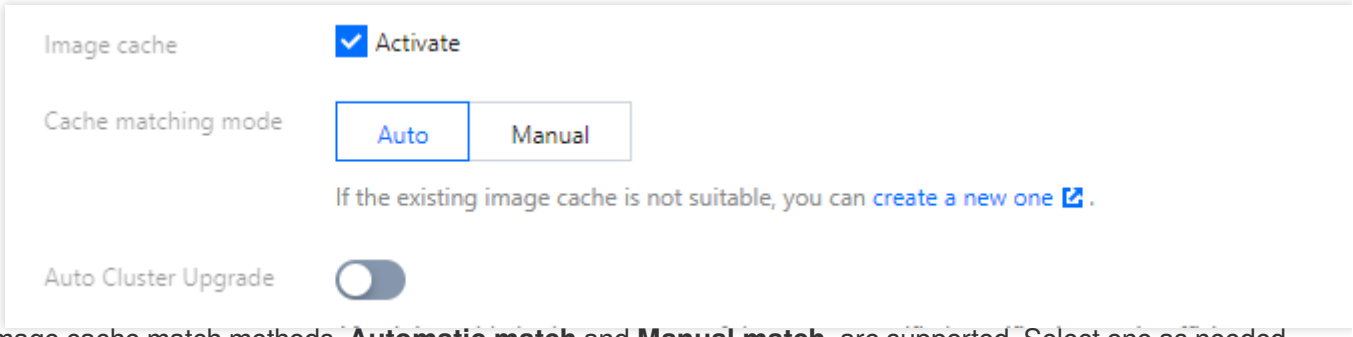
If there are any exception, you can check the events:



```
kubectl describe imc xxx
```

Using an existing image cache

When you create a Pod in a Serverless cluster, you can enable image cache in **Advanced Settings** on the **Create Workload**page.



Two image cache match methods, **Automatic match** and **Manual match**, are supported. Select one as needed.

Automatic match

Manual match

If you select **Automatic match**, the optimal image cache will be matched automatically according to the following match policy:

If the image names and versions are completely the same, the image cache can be matched.

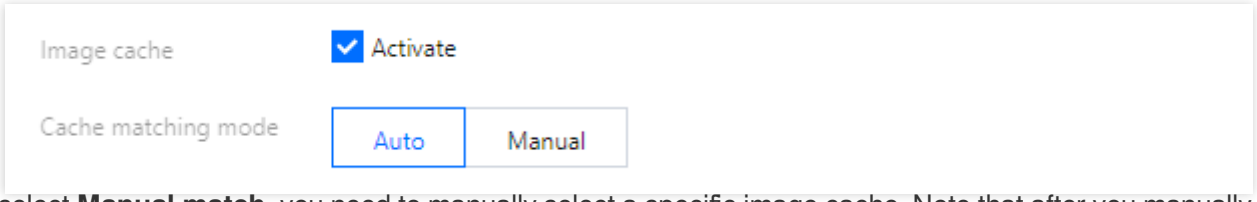
Images with a smaller cache size will be matched first.

Images with a later creation time will be matched first.

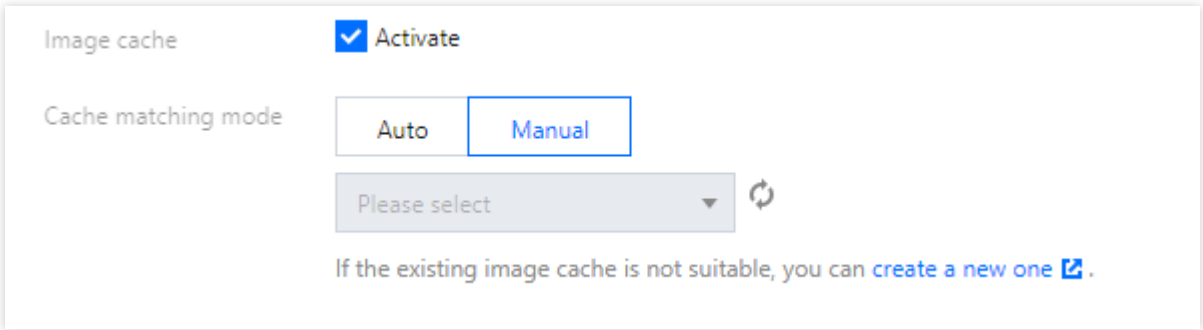
Note

If the versions are both "nginx: latest", the image cache can still be matched; however, as the creation time may be different, the versions may be inconsistent. Therefore, we recommend you specify the version clearly when creating an image cache and instance.

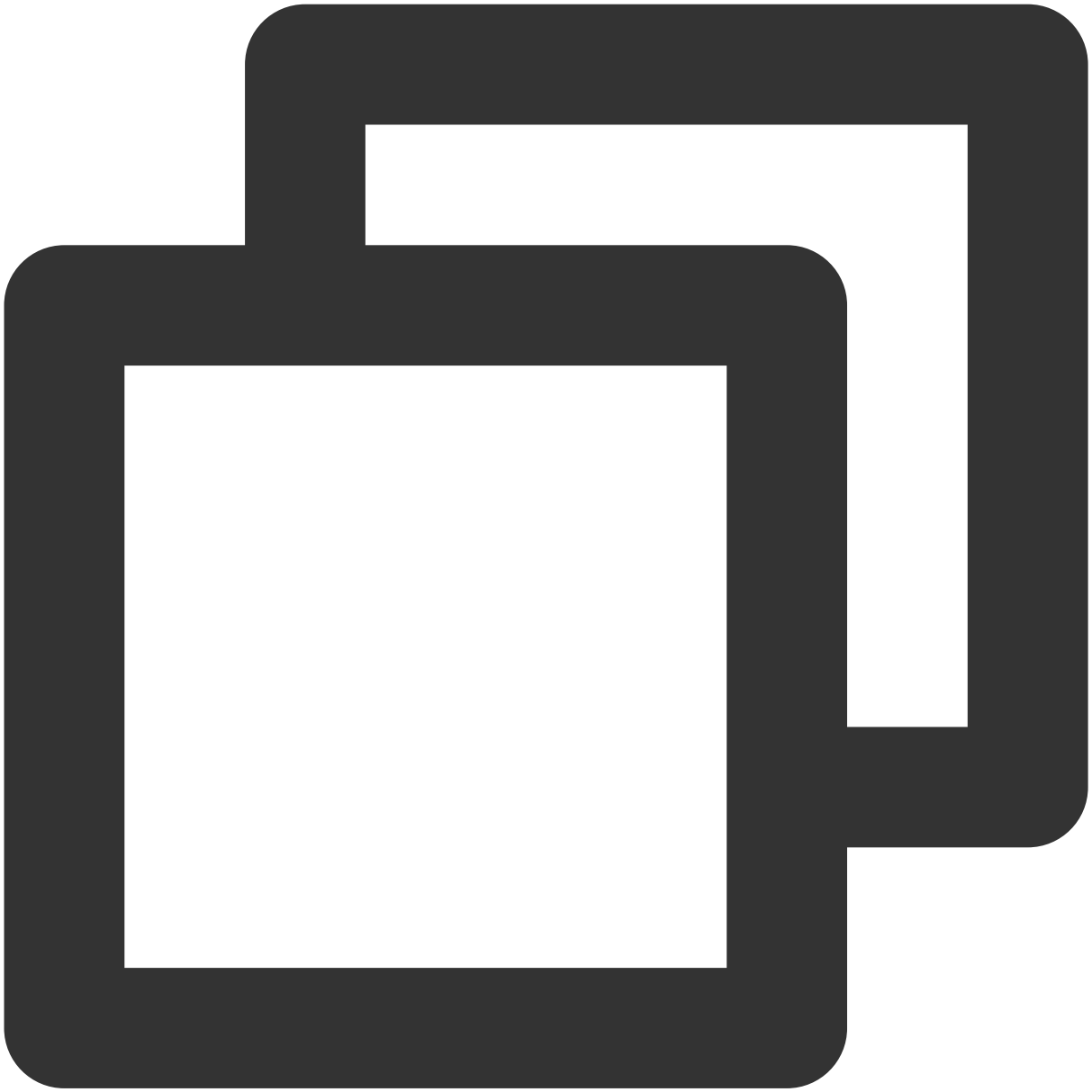
If the corresponding image cache fails to be matched, an image will be pulled normally.



If you select **Manual match**, you need to manually select a specific image cache. Note that after you manually specify an image cache, a data disk will be directly created based on the image cache snapshot and bound to the instance. However, if the data disk does not have the image entered during instance creation (that is, an incorrect image cache is specified manually), an image will be pulled from the newly created data disk.

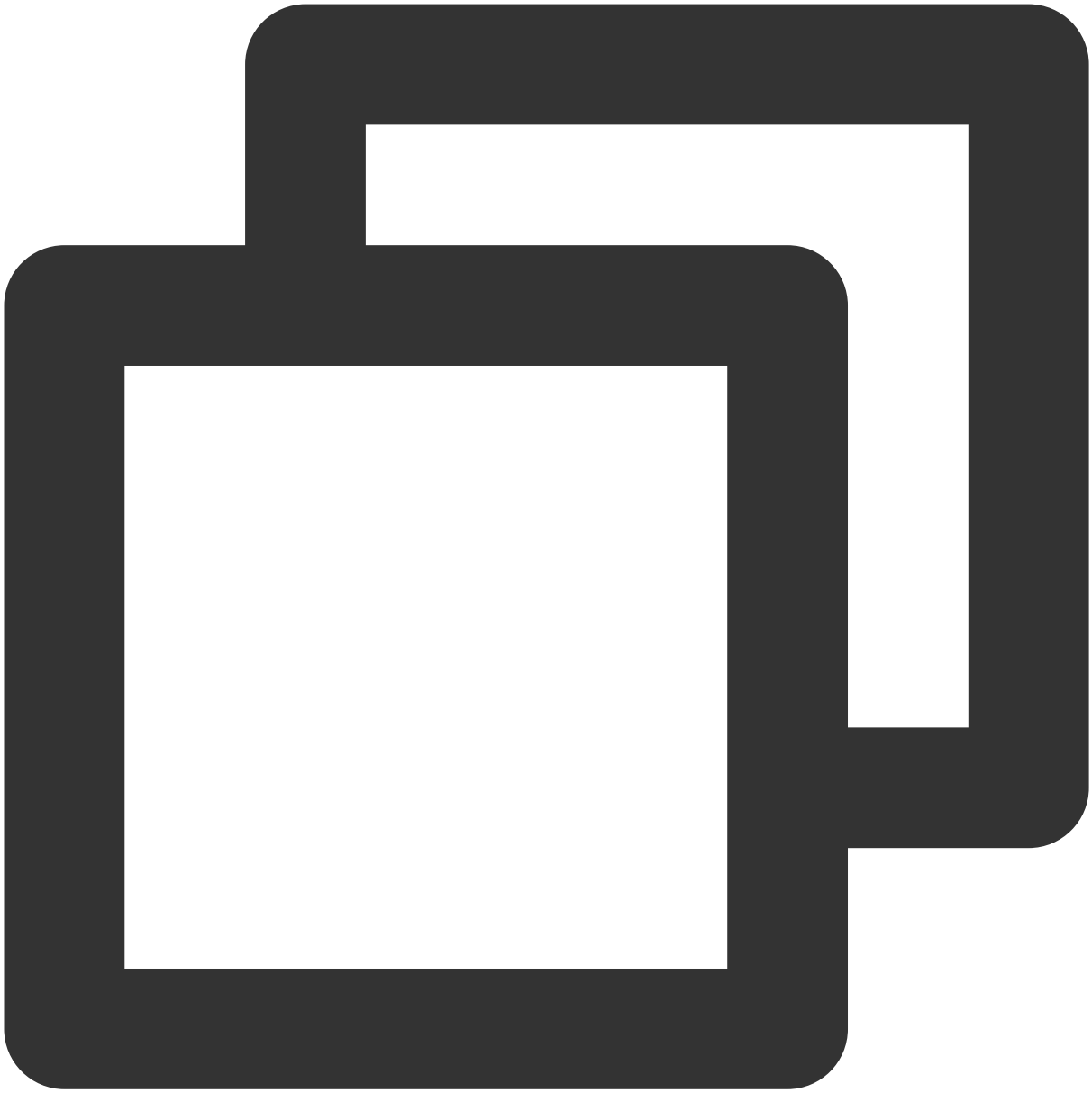


You can specify a Pod annotation to use an image cache on super nodes in TKE clusters. For more information, see super node Annotation.
Automatically match:



```
eks.tke.cloud.tencent.com/use-image-cache: auto
```

Manually specify:

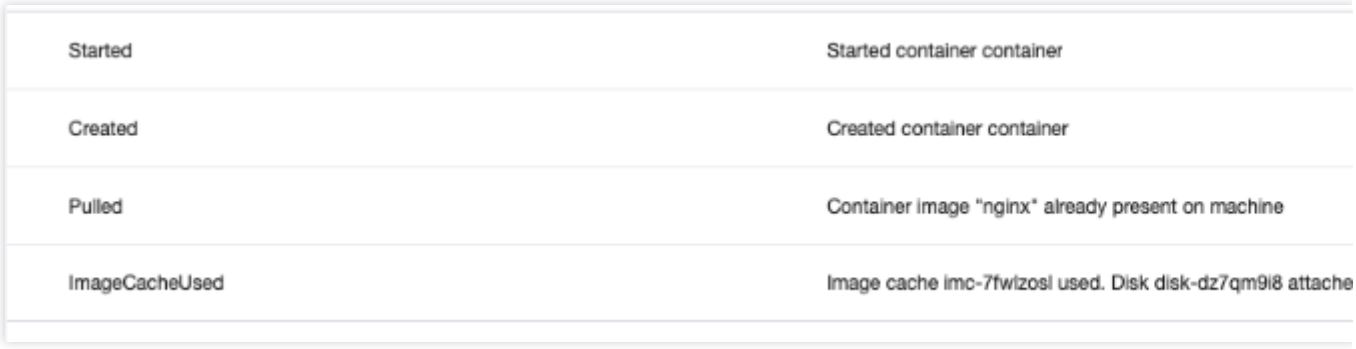


```
eks.tke.cloud.tencent.com/use-image-cache: imc-xxx
```

Matching results

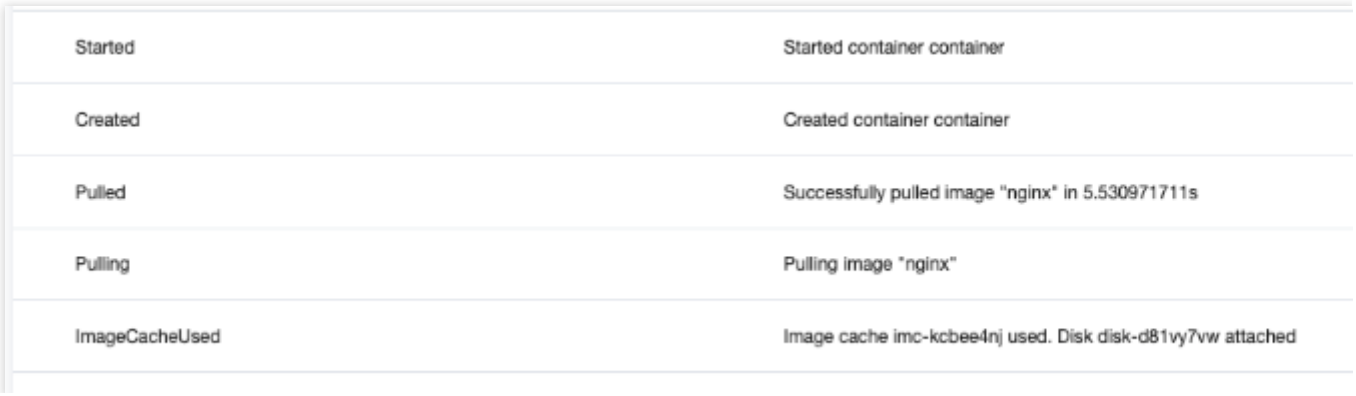
You can check whether the match succeeds in the instance creation event.

If the image cache is matched successfully, the following event will be displayed:



Started	Started container container
Created	Created container container
Pulled	Container image "nginx" already present on machine
ImageCacheUsed	Image cache imc-7fwlzosl used. Disk disk-dz7qm9i8 attache

If this event is not displayed, no appropriate image caches have been matched.
Note that if you choose to manually match an image cache but it fails to be matched, an image will be pulled from the newly created data disk, and the following event will be displayed:



Started	Started container container
Created	Created container container
Pulled	Successfully pulled image "nginx" in 5.530971711s
Pulling	Pulling image "nginx"
ImageCacheUsed	Image cache imc-kcbee4nj used. Disk disk-d81vy7vw attached

OPS Center
Monitoring and alarm

Last updated：2022-09-14 10:35:36

TKE Serverless Cluster collects and displays metrics for clusters, workloads, pods, and containers.

Prerequisites

An TKE Serverless cluster has been created and is in the Running state. For more information, see Creating a Cluster.

Procedure

1. Log in to the TKE console and click **Elastic Cluster** in the left sidebar.
2. On the **Cluster Management** page that appears, click the ID of the TKE Serverless cluster.
3. On the cluster resource management page, view the metrics and configure alarms for them. For more information, see the following documents:

- Viewing Metrics
- Setting Alarms

Monitoring and Alarm Metrics

Monitoring metrics

TKE currently provides monitoring metrics in the following dimensions. All metrics are **average values** of the statistics collected within the statistical period.

Note：

- For monitoring metrics for persistent volumes (PVs) used by a workload, see Block Storage and Cloud File Storage.
- For network monitoring metrics for a CLB associated with a Service, see the Cloud Load Balancer.
- For more information on how to create an alarm policy, see Creating an Alarm Policy.

Monitoring metrics for a cluster

Metric	Unit	Description
CPU Usage	Core	Total number of CPU cores used by the running pods in the cluster
MEM Usage	B	Total amount of memory used by the running pods in the cluster

Monitoring metrics for a workload

Metric	Unit	Description
Restart of Pods	Times	Total number of times the pods in the workload are restarted
CPU Usage	Core	Total number of CPU cores used by the pods in the workload
CPU Utilization (% of Pod Specification)	%	Percentage of the CPU cores allocated to the pods in the workload that is used by the pods
MEM Usage	B	Total amount of memory used by the running pods in the workload
MEM Utilization (% of Pod Specification)	%	Percentage of the memory capacity allocated to the pods in the workload that is used by the pods

Monitoring metrics for a pod

Metric	Unit	Description
Exception	-	Status of the pod, which can be normal or abnormal
CPU Usage	Core	Number of CPU cores used by the pod
CPU Utilization (% of Request)	%	Percentage of the total number of CPU cores specified by Request that is used by the pod
CPU Utilization (% of Limit)	%	Percentage of the total number of CPU cores specified by Limit that is used by the pod
CPU Utilization (% of Pod Specification)	%	Percentage of the CPU cores allocated to the pod that is used by the pod
MEM Usage	B	Amount of memory used by the pod, including the cache usage for the pod
MEM Utilization (% of Request)	%	Percentage of the total amount of memory specified by Request that is used by the pod

Metric	Unit	Description
MEM Utilization (% of Limit)	%	Percentage of the total amount of memory specified by Limit that is used by the pod
MEM Utilization (% of Pod Specification)	%	Percentage of the memory capacity allocated to the pod that is used by the pod

Monitoring metrics for a container

Metric	Unit	Description
CPU Usage	Core	Number of CPU cores used by the container
CPU Utilization (% of Request)	%	Percentage of the total number of CPU cores specified by Request that is used by the container
CPU Utilization (% of Limit)	%	Percentage of the total number of CPU cores specified by Limit that is used by the container
MEM Usage	B	Amount of memory used by the container, including the cache usage for the container
MEM Utilization (% of Request)	%	Percentage of the total amount of memory specified by Request that is used by the container
MEM Utilization (% of Limit)	%	Percentage of the total amount of memory specified by Limit that is used by the container

Alarm Metrics

TKE currently provides alarm metrics in the following dimensions. All metrics are **average values** of the statistics collected within the statistical period.

Alarm metrics for a pod

Metric	Unit	Description
CPU Utilization (% of Pod Specification)	%	Percentage of the CPU cores allocated to the pod that is used by the pod
MEM Utilization (% of Pod Specification)	%	Percentage of the memory capacity allocated to the pod that is used by the pod

Metric	Unit	Description
CPU Utilization (% of Request)	%	Percentage of the total number of CPU cores specified by Request that is used by the pod
MEM Utilization (% of Request)	%	Percentage of the total amount of memory specified by Request that is used by the pod
CPU Utilization (% of Limit)	%	Percentage of the total number of CPU cores specified by Limit that is used by the pod
MEM Utilization (% of Limit)	%	Percentage of the total amount of memory specified by Limit that is used by the pod
Restart of Pods	Times	Number of times the pod is restarted.
Pod Ready	-	Status of the pod. By default, an alarm is generated when the value is False.
CPU Usage	Core	Number of CPU cores used by the pod
MEM Usage	MB	Amount of memory used by the pod, including the cache usage for the pod

Log Collection
Using Environment Variables to Configure Log Collection

Last updated：2022-09-23 14:45:09

Overview

In TKE Serverless clusters, you can use environment variables to configure log collection, and collect logs by line without parsing, or use custom resource definitions (CRD) to configure log collection.

This document describes how to use the environment variables to configure the log collection feature for a TKE Serverless cluster, and to send the logs of services within the cluster to Cloud Log Service (CLS) or external Kafka. This feature is suitable for users who need to store and analyze service logs in TKE Serverless clusters.

To use the log collection feature in a TKE Serverless cluster, you need to manually enable it for the cluster when creating a workload. You can enable this feature by performing the following operations:

- Configuring log collection in the console
- Configuring log collection via yaml
- Updating log collection

Notes

After the log collection feature is enabled for the TKE Serverless cluster, the log collection agent will send the collected logs in JSON format to the specified consumer end based on your configuration. The details of the collection path and consumer end are as follows:

- **Consumer end**: The log collection service allows you to set Kafka or CLS as the log consumer end.
- **Collection path**: The path of the logs to collect. The collection path supports collection standard output (stdout) and absolute path. It also supports wildcard (*). If multiple paths are specified, separate them with “,”.

Prerequisite

- Confirm that the Kubernetes cluster can access the log consumer end.
- The length of a log is limited to 2 M. The log is truncated if this limit is exceeded.

Note

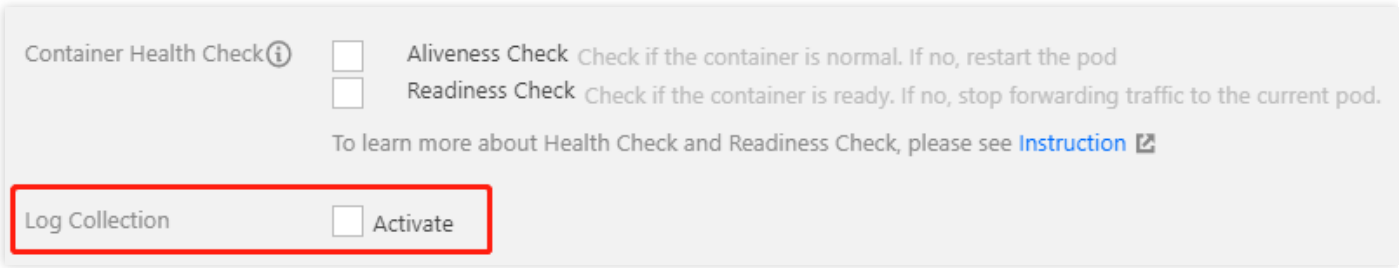
If the log output rate is high, you need to adjust this parameter configuration to avoid OOM. For more information, see How to adjust the log collection configuration to adapt to different log output rates.

Directions

Configuring log collection in the console

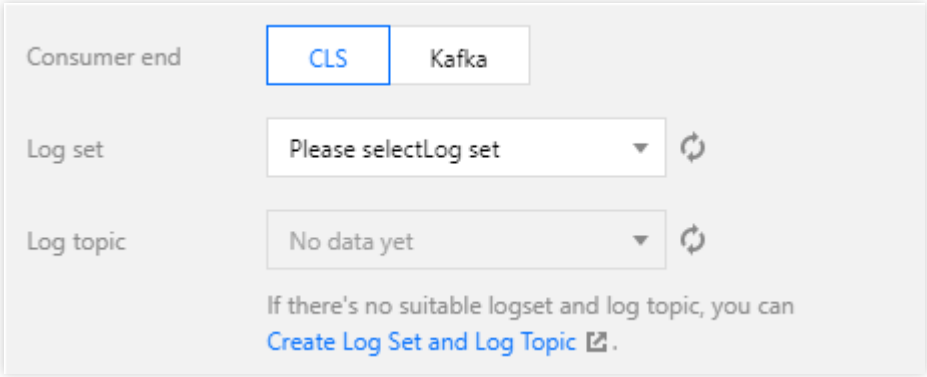
When the log collection is enabled for the TKE Serverless cluster, the log information is collected and output to the specified consumer end in JSON format with Kubernetes metadata attached, including the label of the Pod to which the container belongs, annotation, etc. The specific directions are as follows:

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the cluster management page, click the ID of the target Serverless cluster to enter the cluster details page.
3. Select a workload type in **Workload** on the left to go to the corresponding page, and then click **Create**.
4. In **Containers in the Pod** section, select **Advanced settings**, and check **Activate** to enable log collection.



5. Refer to the following information to configure the log consumer end. You can choose CLS or Kafka as the log consume end.
   - Configuring CLS as the log consumer end
   - Configuring Kafka as the log consumer end

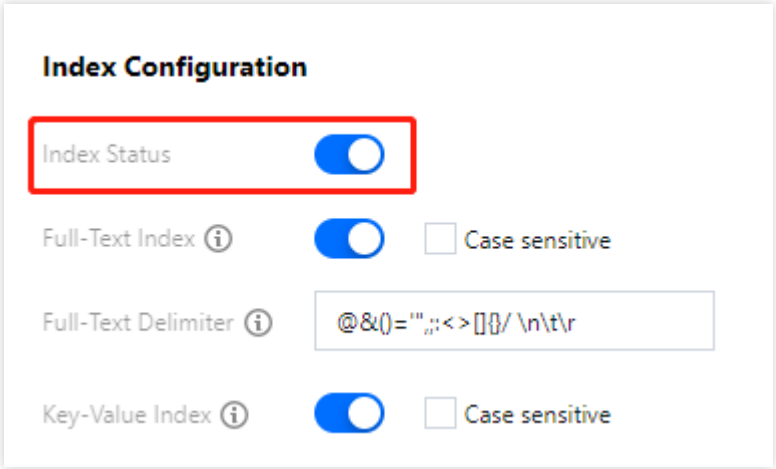
i. Select **CLS** as the **Consumer end**, and select the **Logset** and **Log topic**.



If there is no suitable logset, you can create a logset and a log topic.

> Note
> CLS currently only supports log collection and reporting for intra-region container clusters.

ii. Enable the **log index** for the selected log topic. Index configuration is required for CLS log search and analysis. If it is not enabled, you cannot view the logs. For how to configure the index, see Configuring Index.
You can go to the **CLS console** > **Log topic** page, select a log topic name, and enable the index in the **Index configuration** page.

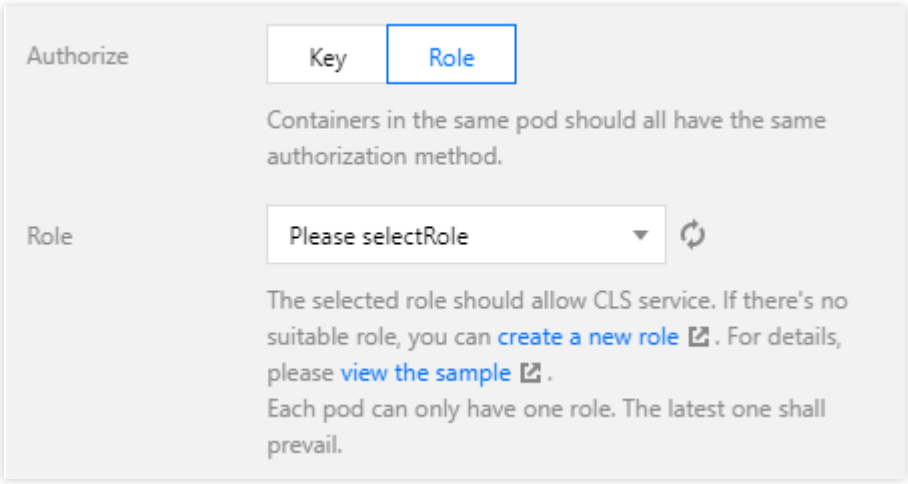


6. Select **Role** or **Key** to authorize.

> Note

- You can only select the same authorization method for the containers in the same Pod. The last modification shall prevail. For example, if you select key authorization for the first container and role authorization for the second container, finally both containers will adopt role authorization.
- You can only select the same role to authorize for the containers in the same Pod.

- Role authorization
- Key authorization
- Select a role that can access CLS, as shown in the figure below:

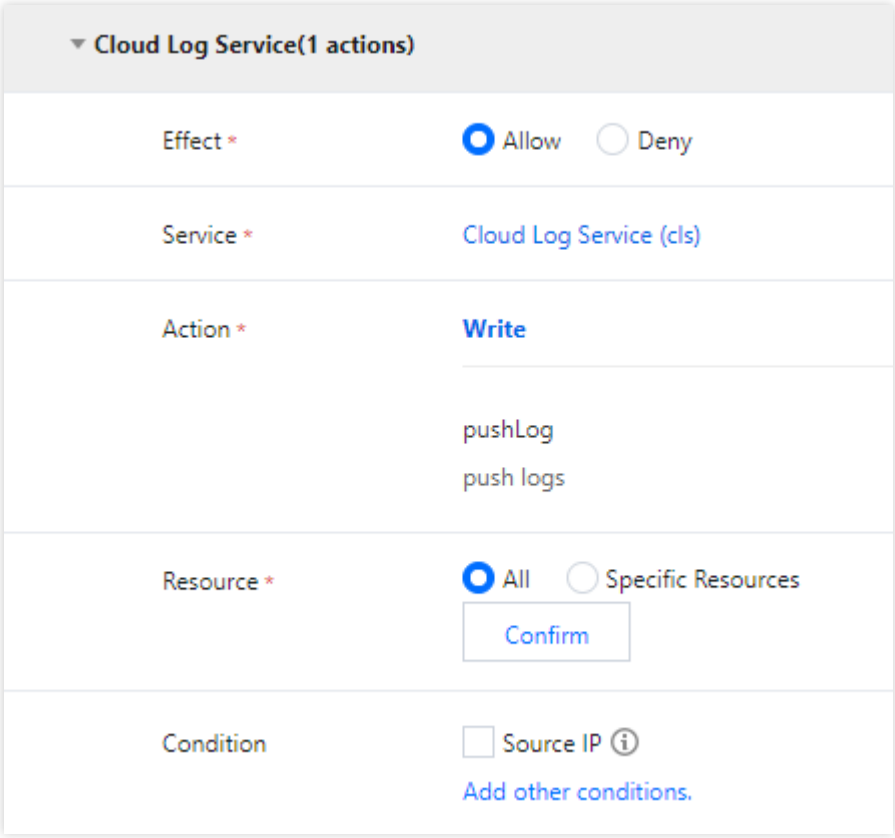


- If there is no suitable role, you can create one as follows:

Creating a policy

You need to create a policy before creating a role. This policy determines the permissions your role can have.

i. Log in to the CAM console and select **Policies** in the left sidebar.
ii. On the “Polices” page, click **Create custom policy**.
iii. Select **Create by policy generator** in “Select policy creation method” pop-up window.
iv. In the “Visual Policy Generator”, select **Cloud Log Service (cls)** for **Service**, and select **Write: pushLog** for **Action**.



v. Click **Next** to go to the “Associate users/user groups” page.

vi. Confirm the policy name and click **Done**.

Creating a role

After creating a policy, you need to bind this policy to a role, so that the role has permissions corresponding to the policy.

i. Log in to the CAM console, and select **Roles** in the left sidebar.

ii. On the “Roles” page, click **Create role**.

iii. In the “Select role entity” window that appears, select **Tencent Cloud Product Service** to go to the “Create custom role” page.

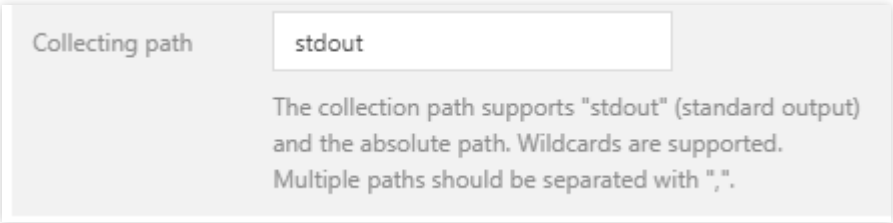
iv. In the “Enter role entity info” step, select **Cloud Virtual Machine (cvm)** and click **Next**.

> Note
> You must select **Cloud Virtual Machine (cvm)** as the role entity, otherwise, you cannot complete the authorization process.

v. In the “Configure role policy” step, select **Created policy**, and click **Next**.

vi. In the “Review” step, enter the role name to review the role information, and then click **Done**. For more information, see Creating a Role.

7. Configure the collection path.



At this point, you have configured the log collection feature. You can set other configurations of the workload as needed.

Configuring log collection via yaml

This document provides three collection methods for your choice: Collecting logs to Kafka, collecting logs to CLS via a secret and collecting logs to CLS via a role.

> Note
> If both key and role authorization are configured in yaml, the Pod actually uses role authorization.

- Collecting logs to Kafka
- Collecting logs to CLS via a secret
- Collecting logs to CLS via a role

Enable log collection by adding environment variables.

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
annotations:
deployment.kubernetes.io/revision: "1"
labels:
k8s-app: kafka
qcloud-app: kafka
name: kafka
namespace: default
spec:
replicas: 1
```

```
selector:
matchLabels:
k8s-app: kafka
qcloud-app: kafka
template:
metadata:
annotations:
eks.tke.cloud.tencent.com/cpu: "0.25"
eks.tke.cloud.tencent.com/mem: "0.5Gi"
labels:
k8s-app: kafka
qcloud-app: kafka
spec:
containers:
- env:
- name: EKS_LOGS_OUTPUT_TYPE
value: kafka
- name: EKS_LOGS_KAFKA_BROKERS
value: 10.0.16.42:9092
- name: EKS_LOGS_KAFKA_TOPIC
value: eks
- name: EKS_LOGS_KAFKA_MESSAGE_KEY # Optional
valueFrom:
fieldRef:
fieldPath: metadata.name
- name: EKS_LOGS_METADATA_ON
value: "true"
- name: EKS_LOGS_LOG_PATHS
value: stdout,/tmp/busy*.log
image: busybox:latest
command: ["/bin/sh"]
args: ["-c", "while true; do echo hello world; date; echo hello >> /tmp/busy.log;
sleep 1; done"]
imagePullPolicy: Always
name: while
resources:
requests:
cpu: 250m
memory: 512Mi
```

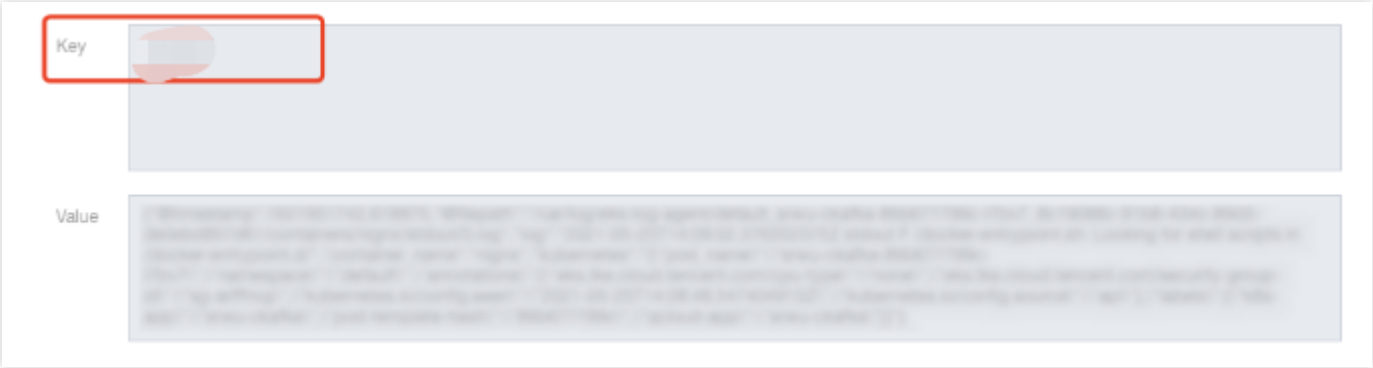
Field description:

Field Name	Description
EKS_LOGS_OUTPUT_TYPE	The consumer end can be kafka or CLS. The key indicates whether log collection is enabled.

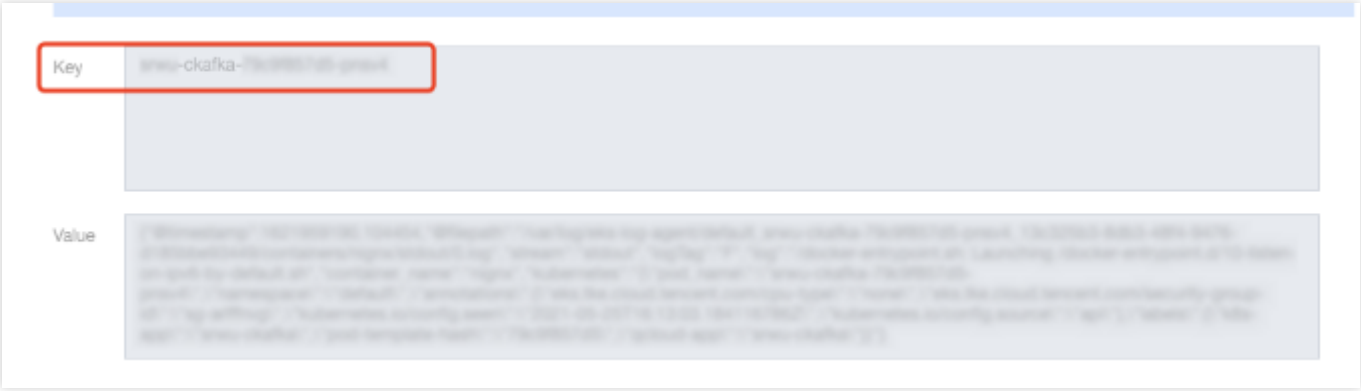
EKS_LOGS_LOG_PATHS	Log path. It supports stdout (collection standard output) and absolute path. It also supports wildcard (*). If multiple paths are specified, separate them with “,”.
EKS_LOGS_METADATA_ON	Valid values: `true` or `false`. Default value: `true`.
EKS_LOGS_KAFKA_TOPIC	Log topic
EKS_LOGS_KAFKA_BROKERS	kafka brokers: ip1:port1, ip1:port2, and ip2:port2 formats, separated by “,”. Use this environmental variable for external application. EKS_LOGS_KAFKA_HOST will no longer be visible to external users.
EKS_LOGS_KAFKA_MESSAGE_KEY	Optional. A key can be specified to deliver the log to the specified partition. • If the delivery by key is not enabled, logs will be randomly delivered to different partitions. • If the delivery by key is enabled, logs with the same key will be delivered to the same partition. **Note**: Here the key gets the variable value from the field of the Pod. The above examples all take the Pod name as an example, and it also supports namespace, PodIP, etc. For more information, see Kubernetes Documentation.

Verify whether the delivery by key is enabled, as shown below:

- If it is not enabled, the key is not displayed in the message details, as shown below:



- If it is enabled, the key is displayed in the message details, as shown below:

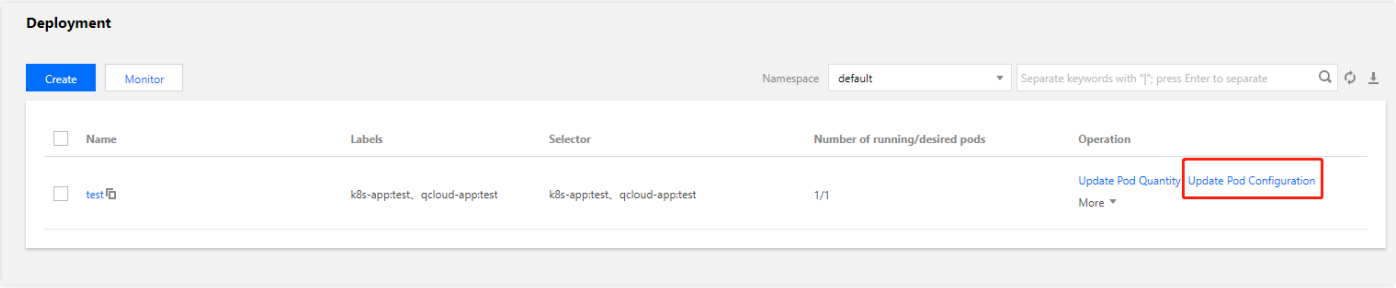


Updating the log collection

You can update log collection in the console or via yaml. Please refer to the following directions:

- Updating log collection in the console
- Updating log collection via yaml

1. Log in to the TKE console and click **Cluster** in the left sidebar.
2. On the cluster management page, click the ID of the target Serverless cluster to enter the cluster details page.
3. Choose **Workload** on the left, find the row of the desired workload for which you want to update log collection, and click **Update Pod configuration** > **Advanced settings** on the right to modify configurations.



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

FAQs

If you have any questions, please refer to Log Collection FAQs. If your question remains unresolved, please Contact Us.

Using a CRD to Configure Log Collection
Enabling Log Collection

Last updated：2022-09-14 18:07:04

Overview

In TKE Serverless cluster, you can use environment variables to configure log collection, or use custom resource definitions (CRD) to configure log collection.

CRD is non-intrusive to Pod and supports single-line, multi-line, separator, full regex, JSON and other log parsing methods. It sends standard output and file logs in the container to Tencent Cloud CLS, which provides various services such as search and analysis, visualization applications, log download and consumption. **It is recommended to use CRD to configure log collection.**

Note
- Using CRD to configure log collection is currently only valid for Pods created after May 25, 2021. If you need to use CRD to configure log collection for the Pods created before, please terminate the Pod and recreate one.
- If the Pods are configured with environment variables and CRD to collect logs at the same time, it will cause repeated collection and repeated billing. Therefore, when using CRD to configure log collection, please delete the relevant environment variables.

Concepts

- **Log rule**: users can use log rules to specify the log collection source, log consumer end, and log parsing method, configure filters, and upload the log that failed to parse etc. The log collector will monitor the changes of the log collection rules, and the changed rules will take effect within 10 seconds.
- **Log source**: includes the standard output of the specified container and the files in the container.
  - When collecting container standard output logs, users can select TKE logs in all containers or specified workloads and specified Pod labels as the log collection source.
  - When collecting container file path logs, users can specify container file path logs in workloads or Pod labels as the collection source.
- **Consumer**: users can select the logsets and log topics of the CLS as the consumer end.
- **Extraction mode**: the log collection agent supports the delivery of collected logs to the user-specified log topic in the format of single-line text, JSON, separator-based text, multi-line text, or full regex.

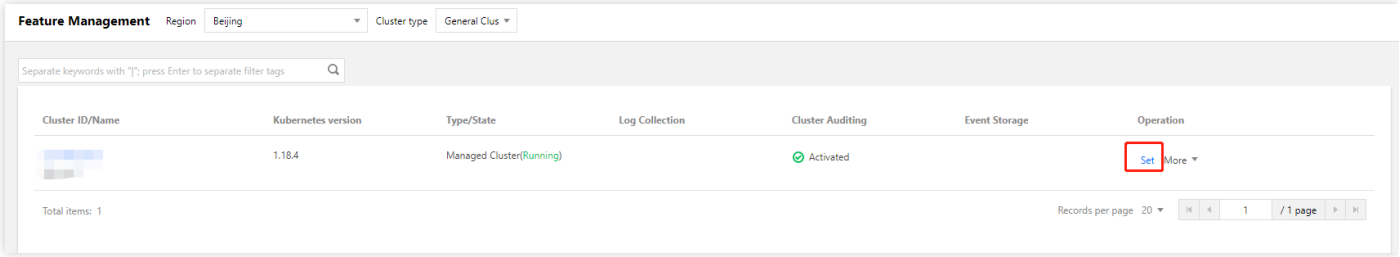
- **Filter**: after filter is enabled, logs will be collected according to the specified rules. "key" supports full matching and the rule supports regex matching. For example, you can set to collect logs containing "ErrorCode = 404".
- **Upload the log that failed to parse**: after this feature is enabled, all logs that failed to parse (as the “Key”), and the original log content (as the “Value”) are uploaded. When this feature is disabled, the logs that failed to parse will be discarded.

Directions

Authorization on first use

When using the log collection feature of the TKE Serverless cluster for the first time, you need to authorize the CLS and other related permissions to ensure that the logs are normally uploaded to the CLS.

1. Log in to the TKE console, and select **OPS Feature Management** in the left sidebar.
2. At the top of the **Feature Management** page, select the region and **Serverless Cluster**. On the right side of the cluster for which you want to enable log collection, click **Set**, as shown in the figure below:



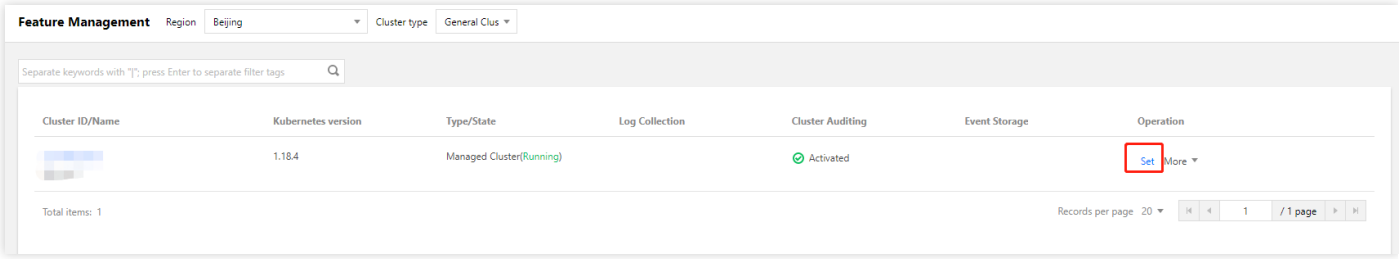
3. On the **Configure Features** page, click **CAM** to complete authorization.
   After authorization is completed, the role TKE_QCSLinkedRoleInEKSLog will be bound to your account by default, and the default policy configured for this role is QcloudAccessForTKELinkedRoleInEKSLog.

> Note：
> You only need to authorize when you use the log collection feature for the first time. If you delete the above roles, you need to authorize again.

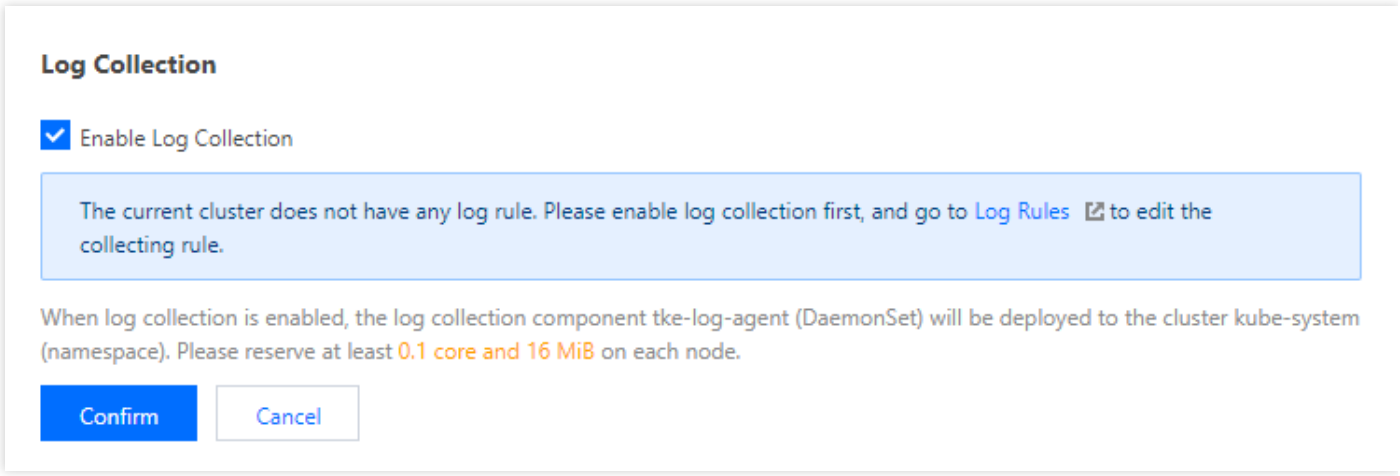
Enabling log collection

After completing the authorization, you can enable the log collection.

1. Log in to the TKE console, and select **OPS Feature Management** in the left sidebar.
2. At the top of the **Feature Management** page, select the region and **Serverless Cluster**. On the right side of the cluster for which you want to enable log collection, click **Set**, as shown in the figure below:



3. On the "Configure Features" page, click **Edit** for log collection, enable log collection, and confirm this operation, as shown in the figure below:



Subsequent Operations

After enabling the log collection feature, you can refer to the following operations to use the CDR to configure the log collection for the TKE Serverless cluster:

- Configuring Log Collection via the Console
- Configuring Log Collection via Yaml

FAQ

If you have problems, please submit a ticket to contact us.

Configuring Log Collection via the Console

Last updated：2023-04-07 15:23:27

This document describes how to use CRD to configure the log collection feature of TKE Serverless cluster.

Prerequisites

Log in to the TKE console, and enable the log collection feature for the serverless cluster. For more information, see Enabling Log Collection.

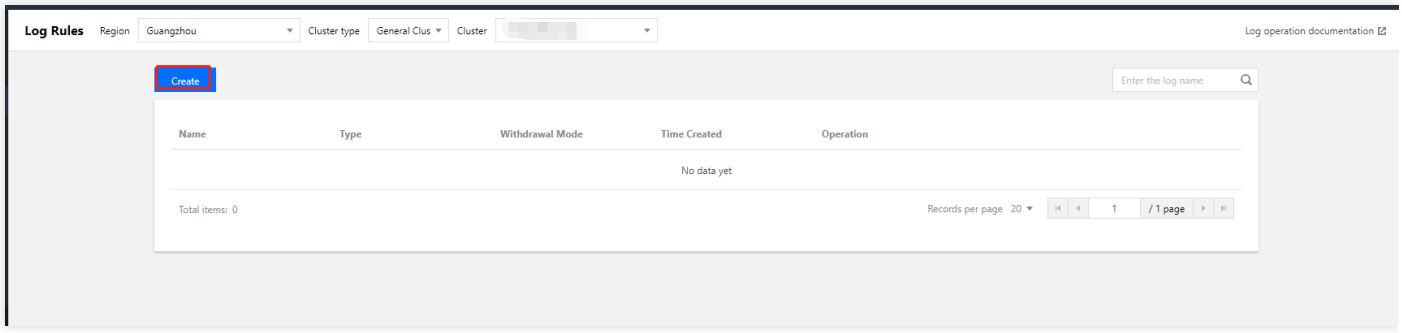
Directions

You can take the following actions to configure after enabling the log collection feature for the cluster:

Configuring the log rule

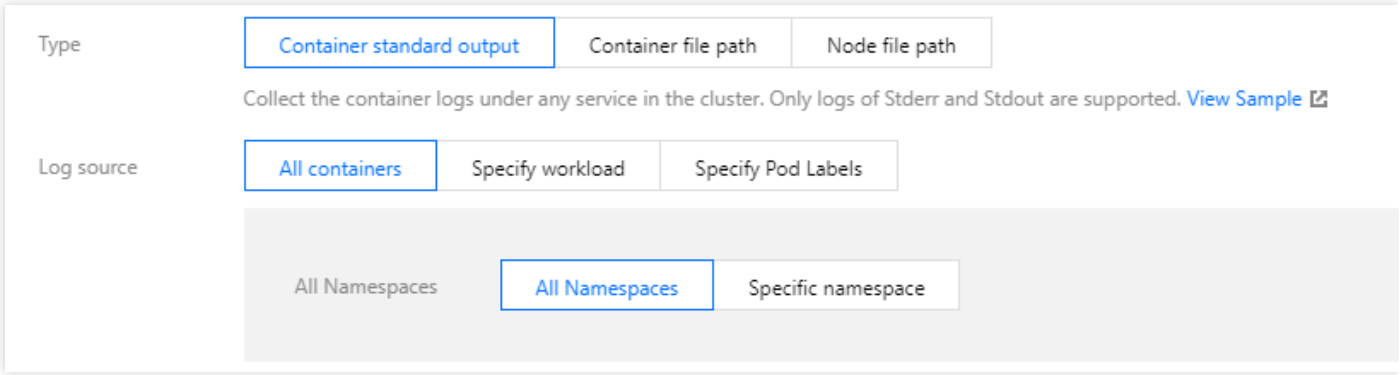
After enabling the log collection, you need to configure the log rules including the log source, consumer end, log parsing method, and so on.

1. Log in to the TKE console, and select **Log Management** > **Log Collection Rules** in the left sidebar.
2. At the top of the **Log Rules** page, select the region and the TKE Serverless cluster where you want to configure the log collection rules and click **Create**, as shown in the figure below:



3. On the "Create Log Collecting Policy" page, select the collection type and configure the log source, consumer end, log parsing method. Currently, the following collection types are supported: container standard output and container file path.
   - Collecting standard output logs of a container
   - Collecting file logs in container

   Select **Container standard output** as the collection type, and configure the log source as needed, as shown below:



This type of log source supports:

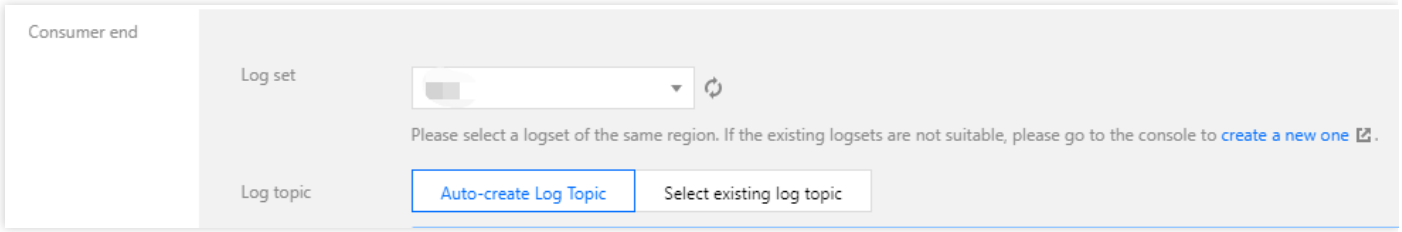
- **All containers**: all namespaces or all containers under a namespace.
- **Specify workload**: the containers of a specified workload under a namespace. You can add multiple namespaces.
- **Specify Pod Labels**: specify multiple Pod Labels under a namespace, and collect all containers that match the Labels.

Note
For container standard output and container files, besides the original log content, the metadata related to the container or Kubernetes (such as the name of the Pod that generated the logs) will also be reported to the CLS. Therefore, when viewing logs, users can trace the log source or search based on the container identifier or characteristics (such as container name and labels).
The metadata related to the container or Kubernetes is shown in the table below:

Field Name	Description
cluster_id	The ID of the cluster to which logs belong
container_name	The name of the container to which logs belong
image_name	The image name IP of the container to which logs belong
namespace	The namespace of the Pod to which logs belong
pod_uid	The UID of the Pod to which logs belong
pod_name	The name of the Pod to which logs belong
pod_ip	The IP of the Pod to which logs belong
pod_lable_{label name}	The labels of the Pod to which logs belong (for example, if a Pod has two labels: app=nginx and env=prod, the reported log will have two metadata entries

attached: pod_label_app:nginx and pod_label_env:prod).

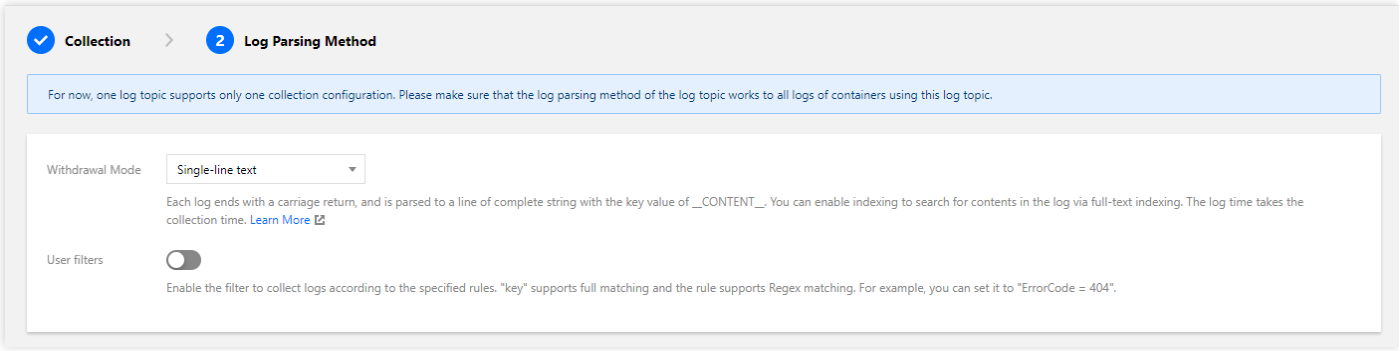
4. Configure the CLS as the consumer end. Select the desired logset and log topic. It is recommended to select **Auto-create Log Topic**, as shown in the figure below:



Note

- CLS currently only supports log collection and reporting for intra-region container clusters.
- The log set and log topic cannot be updated after the log rule is configured.

5. Click **Next** and choose a log extraction mode, as shown below:



Show All

Extraction modes

展开&收起

Parsing Mode	Description	Related Document

Parsing Mode	Description	Related Document
Full text in a single line	A log contains only one line of content, and the line break `\n` to mark the end of a log. Each log will be parsed into a complete string with **CONTENT** as the key value. When log Index is enabled, you can search for log content via full-text search. The time attribute of a log is determined by the collection time.	Full Text in a Single Line
Full text in multi lines	A log with full text in multi lines spans multiple lines and a first-line regular expression is used for match. 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. A default key value, **CONTENT**, will be set as well. The time attribute of a log is determined by the collection time. The regular expression can be generated automatically.	Full Text in Multi Lines
Single line - full regex	The single-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete log. When configuring the single-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	Full Regular Format (Single-Line)
Multiple lines - full regex	The multi-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete piece of log data that spans multiple lines in a log text file (such as Java program logs) based on a regular expression. When configuring the multi-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	Full Regular Format (Multi-Line)
JSON	A JSON log automatically extracts the key at the first layer as the field name and the value at the first layer as the field value to implement structured processing of the entire log. Each complete log ends with a line break `\n`.	JSON Format
Separator	In a separator log, the entire log data can be structured according to the specified separator, and each complete log ends with a line break `\n`. When CLS processes separator logs, you need to define a unique key for each separate field. Invalid fields, which are fields that need not be collected, can be left blank. However, you cannot leave all fields blank.	Separator Format

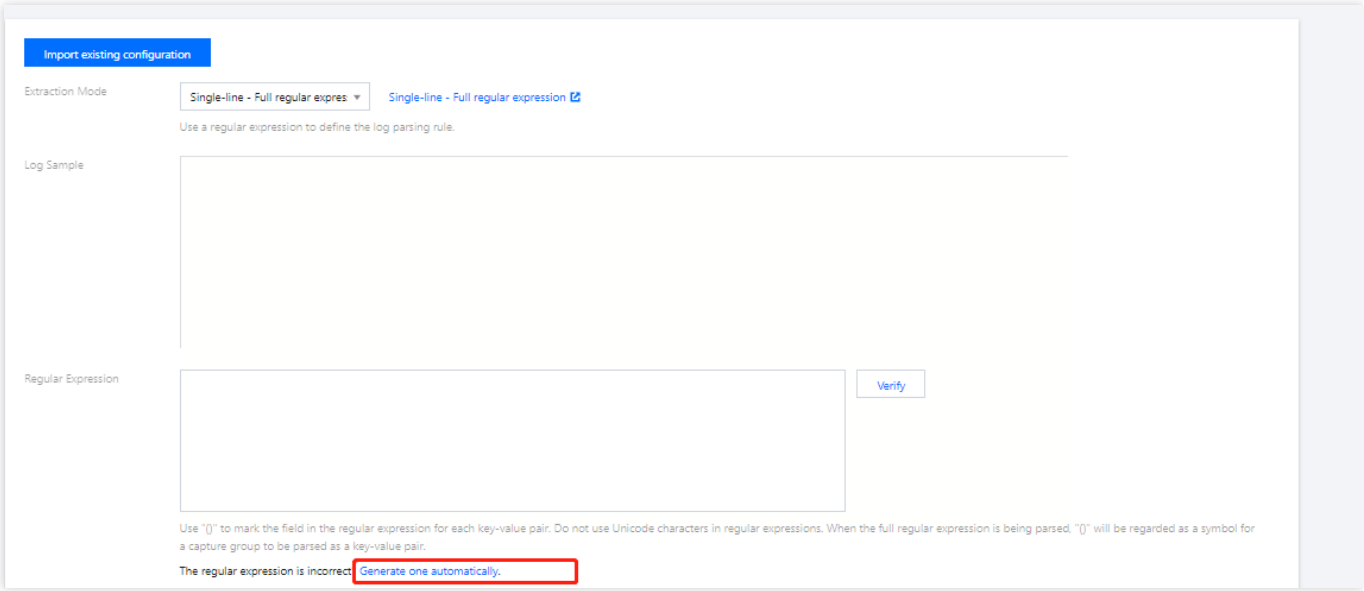
Instructions for automatic regular expression generation

展开&收起

When you select **Multiple lines - full regex**, **Single line - full regex**, or **Multi-line texts**, the **regular expression can automatically generated based on the log sample**.

Here takes the **Single line - full regex** as an example:

i. Click **Auto-Generate Regular Expression**, as shown below:



ii. In the pop-up window, select the field to be extracted in the log sample, and fill in the key.

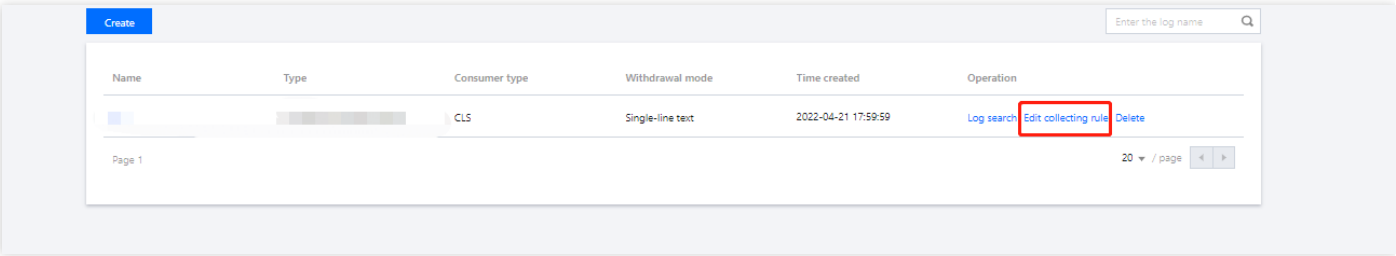
iii. Click **Confirm** to generate the corresponding regular expression of the field, and automatically fill in the extraction result. Repeat this operation until the log is completely extracted.

> Note
> Currently, one log topic supports only one collection configuration. Ensure that all container logs that adopt the log topic can accept the log parsing method that you choose. If you create different collection configurations under the same log topic, the earlier collection configurations will be overwritten.

6. Enable other features as needed.
   - Enable the filter and configure the rules.
     After the filter is enabled, only the logs that meet the filter rules will be collected. Key supports full matching and the rule supports regex matching. For example, you can set to collect logs containing "ErrorCode = 404".
   - Enable the upload of the log that failed to parse
     After this feature is enabled, all logs that failed to parse (as the “Key”), and the original log content (as the “Value”) are uploaded. When this feature is disabled, the logs that failed to parse will be discarded.
7. Click *Done* to complete the process.

Updating the log rules

1. Log in to the TKE console, and select **Log Management** > **Log Collection Rules** in the left sidebar.
2. In the **Log Collection Rules** page, select the log rule to update, and click **Edit Collecting Rule** on the right side of the rule, as shown below:



3. Update the configuration as needed and click **Done**.

Configuring Log Collection via Yaml

Last updated：2023-03-14 18:19:11

This document describes how to use CRD to configure the log collection feature of a TKE Serverless cluster via YAML.

Prerequisites

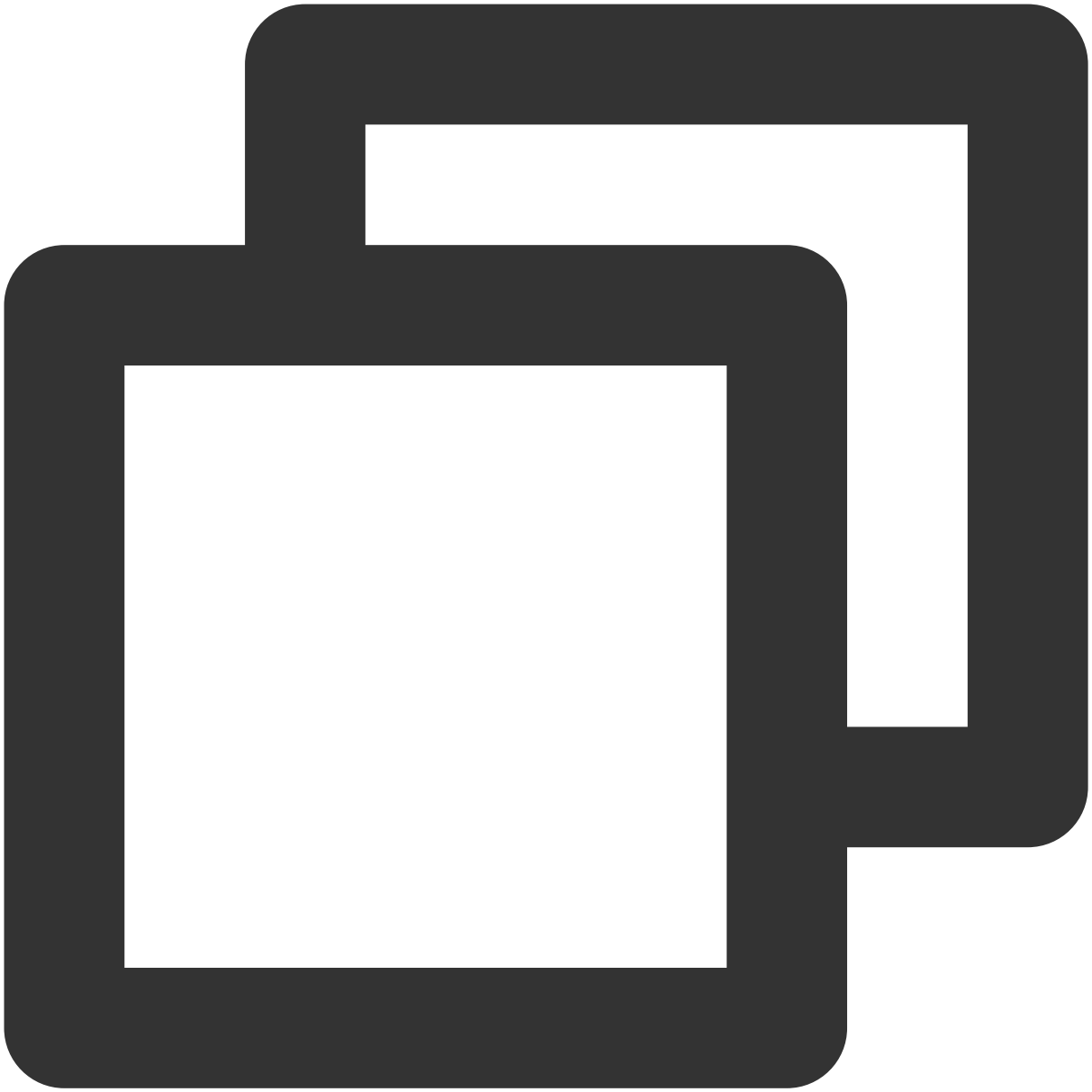
Log in to the TKE console, and enable the log collection feature for the serverless cluster. For more information, see Enabling Log Collection..

Creating the CRD

To create a collection configuration, you only need to define the LogConfig CRD. The collection component will modify the corresponding CLS log topics based on changes to the LogConfig CRD and set the bound server group. The CRD format is as follows:

clsDetail Field Description

Note:
You cannot modify a specified topic.
If the collection type is selected as "Container File Path", the corresponding path cannot be a soft link. Otherwise, the actual path of the soft link will not exist in the collector's container, resulting in log collection failure.



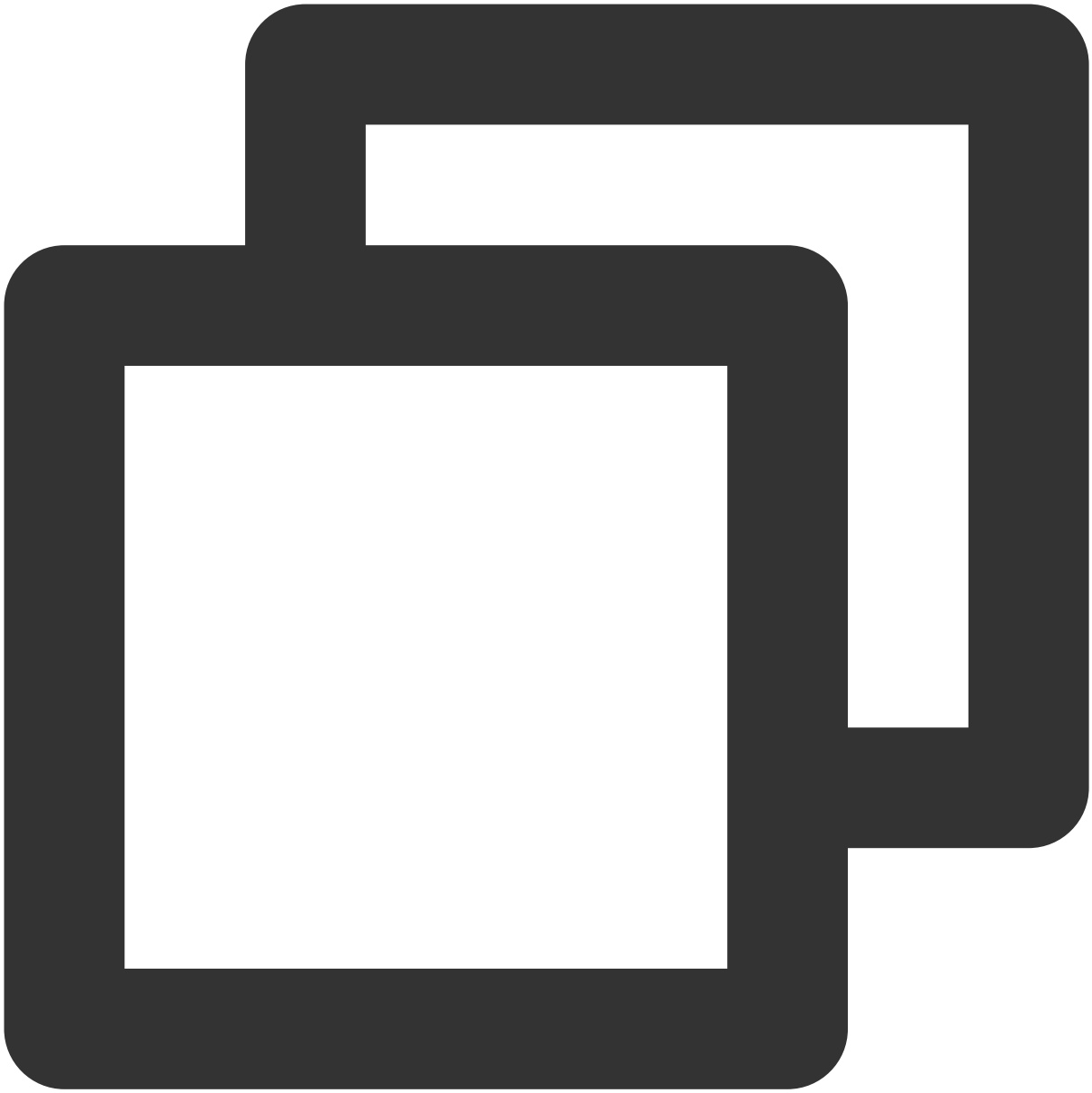
```
clsDetail:
  ## If the log topic is created automatically, the names of logset and topic need
  logsetName: test                    ## CLS logset name. Logset for the name will
  topicName: test                     ## CLS log topic name. Log topic for the name

  # Select an existing logset and log topic. If the logset is specified but the log
  logsetId: xxxxxx-xx-xx-xx-xxxxxxxx  ## The ID of the CLS logset. The logset needs
  topicId: xxxxxx-xx-xx-xx-xxxxxxxx       ## CLS log topic ID. The log topic needs

  logType: json_log  ## Log collection format. json_log: json format. delimiter_log
  logFormat: xxx                      ## Log formatting method
```

```
period: 30                          ## Lifecycle in days. Value range: 1-3600. `3
partitionCount:                     ## The number (an integer) of log topic parti
tags:                              ## Tag description list. This parameter is used
 - key: xxx                         ## Tag key
 value: xxx                        ## Tag value
autoSplit: false                    ## Whether to enable automatic split (Boolean
maxSplitPartitions:
storageType: hot                   ## Log topic storage class. Valid values: `hot`
excludePaths:                       ## Collection path blocklist
  - type: File                      ## Type. Valid values: `File`, `Path`.
      value: /xx/xx/xx/xx.log            ## The value of `type`
indexs:                            ## You can customize the indexing method and f
  - indexName:    ## When a key value or metafield index needs to be configured f
      indexType:    ## Field type. Valid values: `long`, `text`, `double`
      tokenizer:    ## Field delimiter. Each character represents a delimiter. On
      sqlFlag:     ## Whether the analysis feature is enabled for the field (Bool
      containZH:  ## Whether Chinese characters are contained (Boolean)
region: ap-xxx                      ## Topic region for cross-region shipping
userDefineRule: xxxxxx              ## Custom collection rule, which is a seriali
extractRule: {}                     ## Extraction and filter rule. If `ExtractRul
```

inputDetail Field Description



```
inputDetail:
  type: container_stdout  ## Log collection type, including container_stdout (conta

  containerStdout:        ## Container standard output
    namespace: default    ## The Kubernetes namespace of the container to be collec
    excludeNamespace: nm1,nm2  ## The Kubernetes namespace of the container to be e
    nsLabelSelector: environment in (production),tier in (frontend) ## Filter names
    allContainers: false       ## Whether to collect the standard output of all con
    container: xxx             ## Name of the container of which the logs will be c
    excludeLabels:  ## Pods with the specified labels will be excluded. This field
      key2: value2  ## Pods with multiple values of the same key can be matched. Fo
```

```
    includeLabels:  ## Pods with the specified labels will be collected. This field
      key: value1  ## The `metadata` will be carried in the log collected based on

    metadataLabels:           ## Specify the Pod labels to be collected as the meta
    - label1
    customLabels:             ## Custom metadata
      label: l1

    workloads:
    - container: xxx  ## Name of the container to collect. If this parameter is not
      kind: deployment  ## Workload type. Supported values include deployment, daem
      name: sample-app  ## Workload name
      namespace: prod  ## Workload namespace

  containerFile:  ## File in the container
    namespace: default      ## The Kubernetes namespace of the container to be coll
    excludeNamespace: nm1,nm2   ## The Kubernetes namespace of the container to be
    nsLabelSelector: environment in (production),tier in (frontend) ## Filter names
    container: xxx          ## The name of container of which the logs will be coll
    logPath: /var/logs      ## Log folder. Wildcards are not supported.
    filePattern: app_*.log  ## Log file name. It supports the wildcards "*" and "?"
    customLabels:  ## Custom metadata
      key: value
    excludeLabels:  ## Pods with the specified labels will be excluded. This field
      key2: value2  ## Pods with multiple values of the same key can be matched. Fo

    includeLabels:  ## Pods with the specified labels will be collected. This field
      key: value1  ## The `metadata` will be carried in the log collected based on
    metadataLabels:      ## Specify the Pod labels to be collected as the metadata.
    - label1             ## Pod label
    workload:
      container: xxx    ## Name of the container to collect. If this parameter is n
      name: sample-app  ## Workload name
```

Log Parsing Format

Full Text in a Single Line

Full Text in Multi Lines

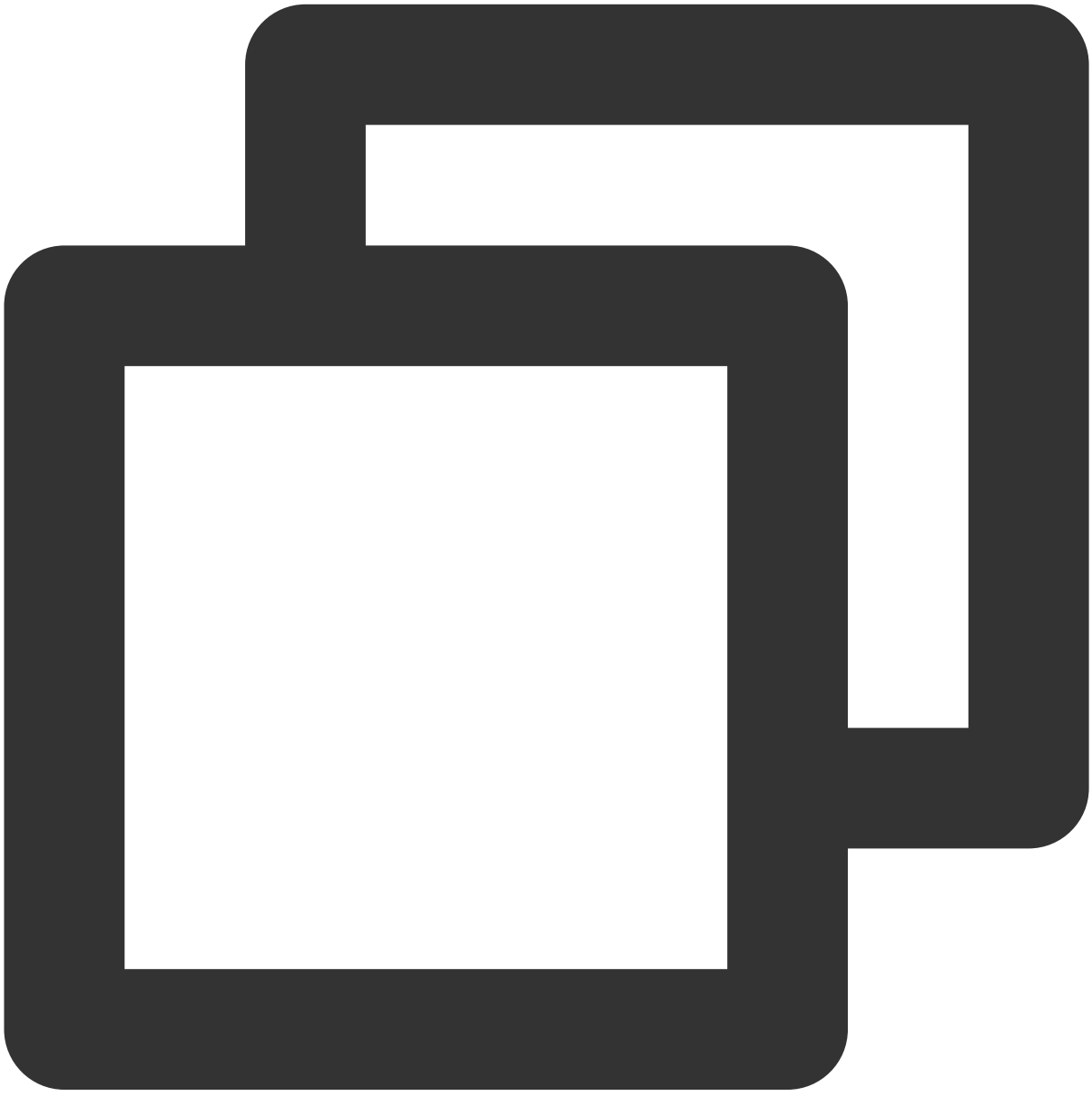
Single line - full regex

Multiple lines - full regex

JSON Format

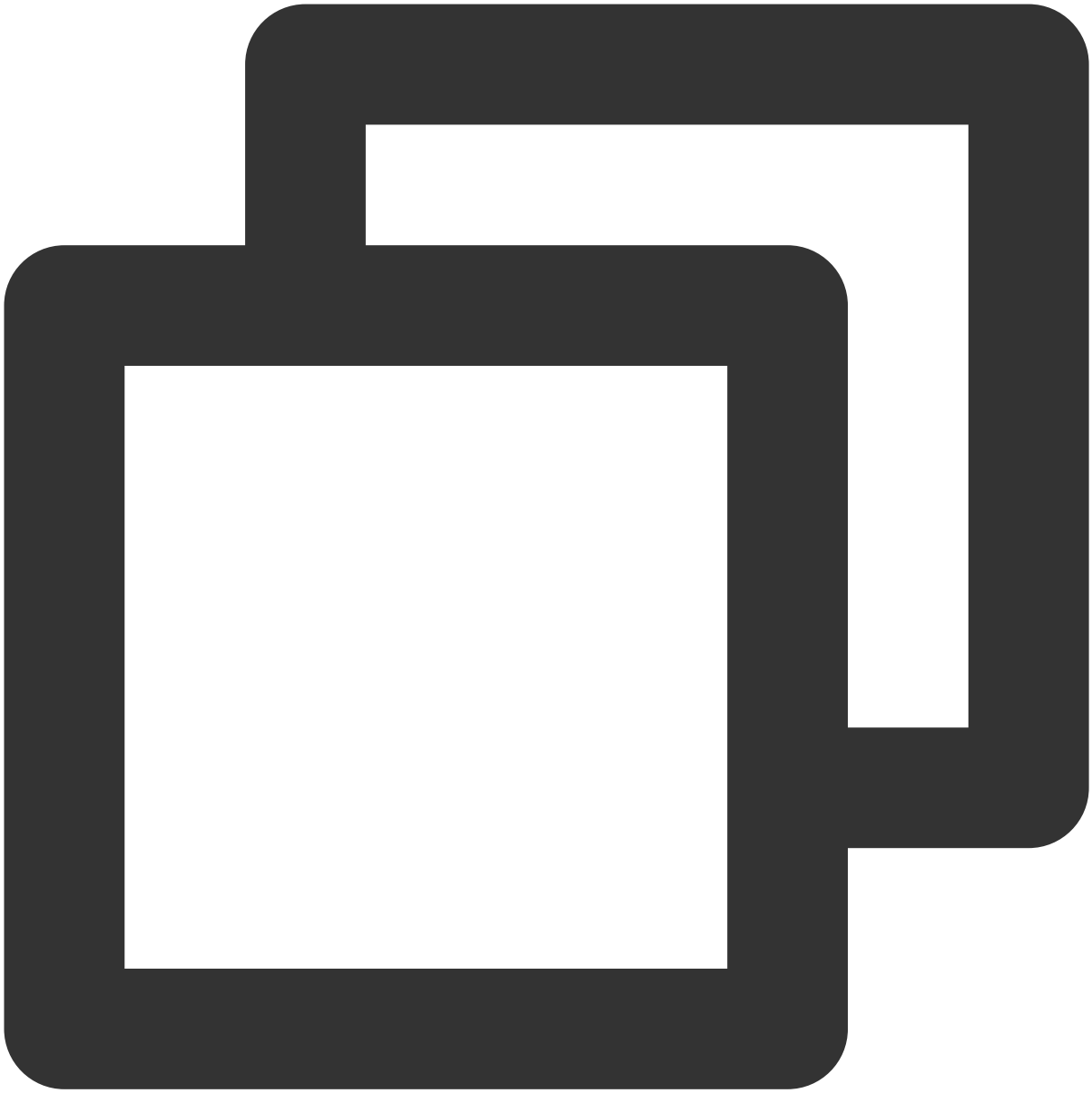
Separator Format

A log with full text in a single line means a full log occupies one line. When CLS collects logs, it uses `\\n` as a line break to end a log. For easier structural management, each log is provided with a default key value `__CONTENT__` . However, the log data itself is no longer structured, and the log fields are not extracted. The `Time` log attribute depends on the time the log is collected. For more information, see Full text in a single line.
Assume that the raw data of a log is as follows:



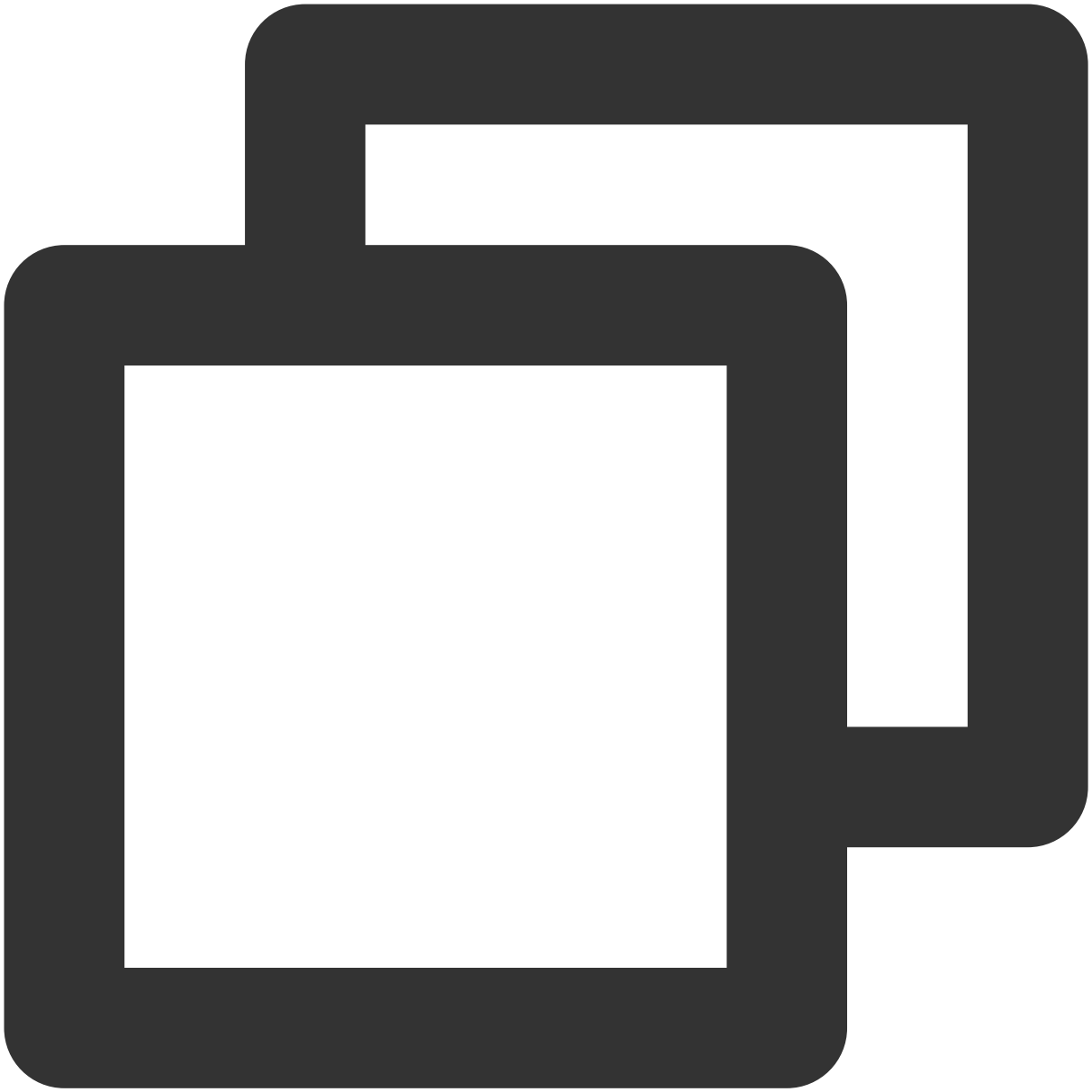
```
Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.el7.x86_64
```

A sample of LogConfig configuration is as follows:



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   clsDetail:
     topicId: xxxxxx-xx-xx-xx-xxxxxxxx
     # Single-line log
     logType: minimalist_log
```

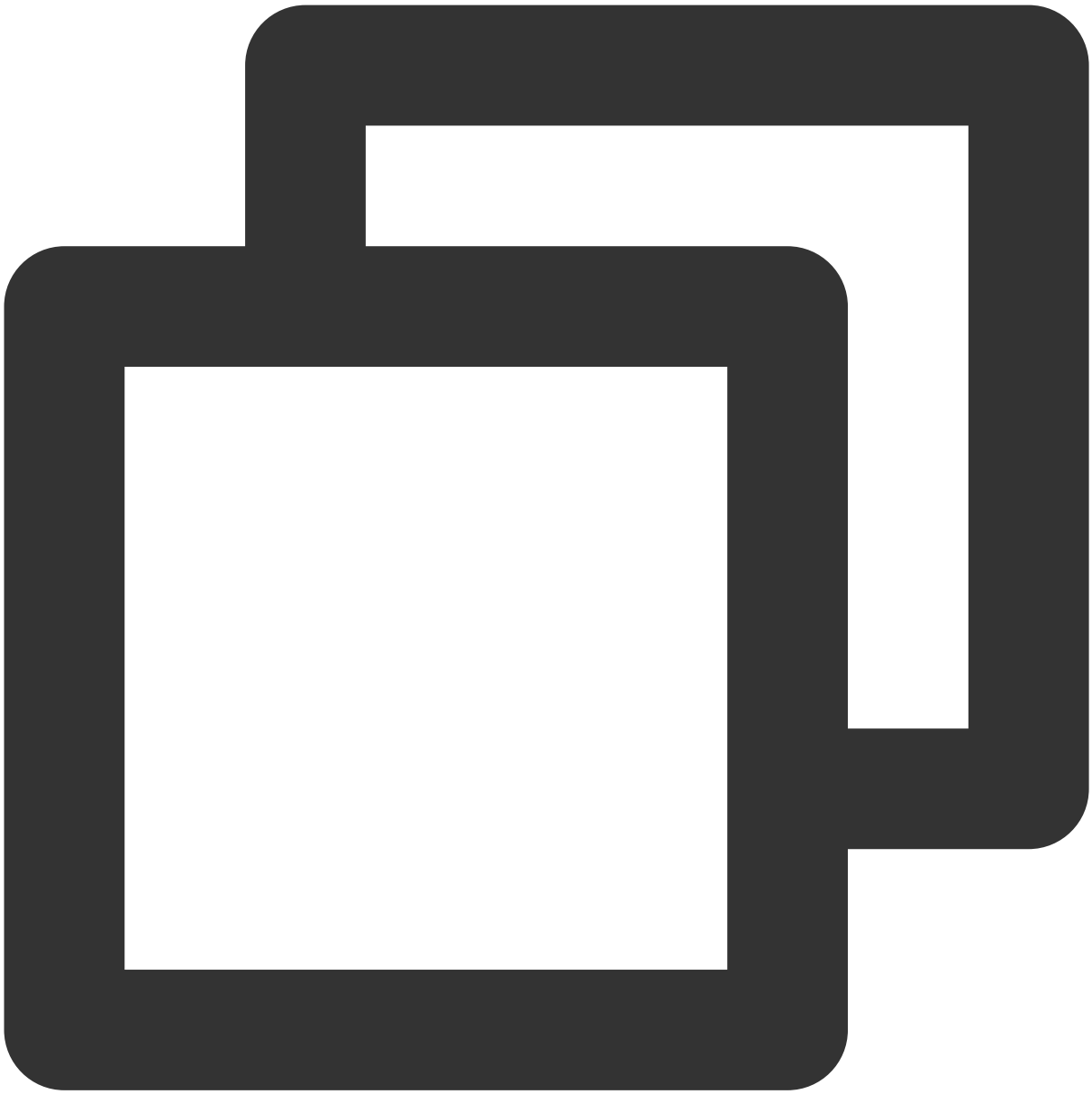
The data collected to CLS is as follows:



```
__CONTENT__:Tue Jan 22 12:08:15 CST 2019 Installed: libjpeg-turbo-static-1.2.90-6.e
```

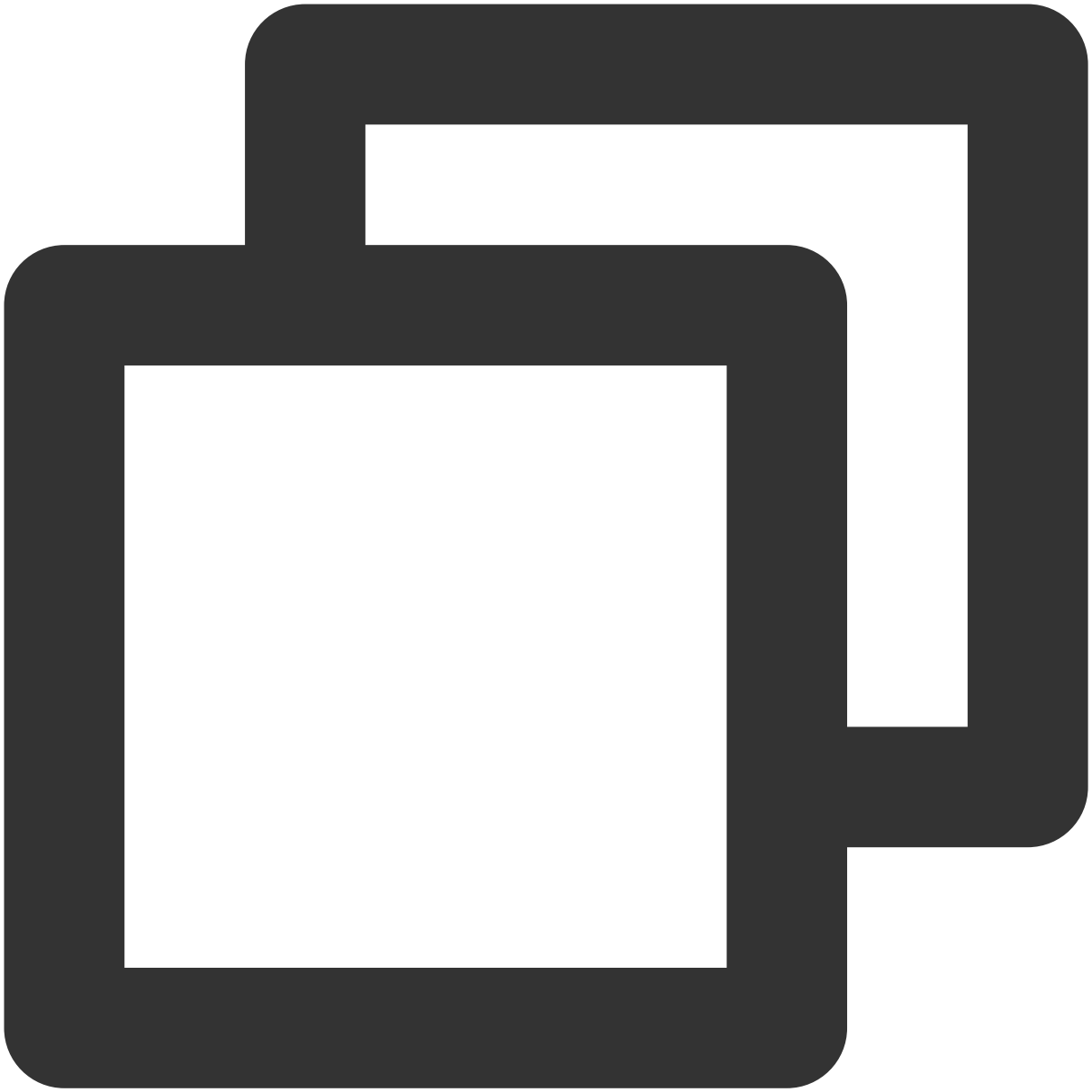
A log with full text in multi lines means that a full log may occupy multiple lines (such as Java stacktrace). In this format, the line break `\\n` cannot be used as the end mark of a log. To help the CLS system distinguish among the logs, "First Line Regular Expression" is used for matching. When a log in a line matches the preset regular expression, it is considered to be the beginning of a log, and the next matching line will be the end mark of the log. In this format, a default key value `__CONTENT__` is also set. However, the log data itself is no longer structured, and the log fields are not extracted. The `Time` log attribute depends on the time the log is collected. For more information, see Full text in multi lines.

Assume that the raw data of a multi-line log is:



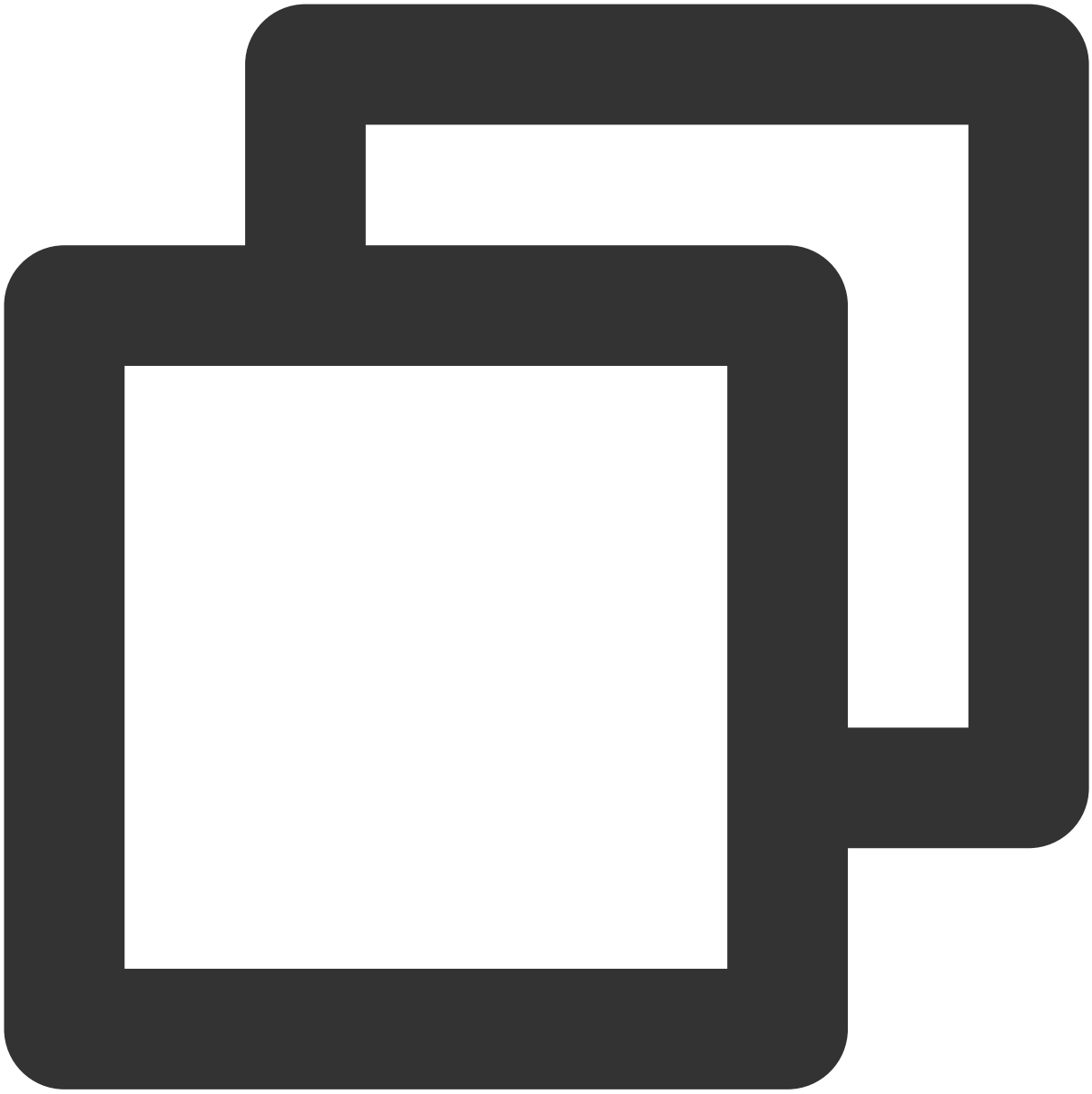
```
2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:
java.lang.NullPointerException
     at com.test.logging.FooFactory.createFoo(FooFactory.java:15)
     at com.test.logging.FooFactoryTest.test(FooFactoryTest.java:11)
```

A sample of LogConfig is as follows:



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   clsDetail:
     topicId: xxxxxx-xx-xx-xx-xxxxxxxx
     # Multi-line log
     logType: multiline_log
     extractRule:
       # Only a line that starts with a date time is considered the beginning of a
       beginningRegex: \\d{4}-\\d{2}-\\d{2}\\s\\d{2}:\\d{2}:\\d{2},\\d{3}\\s.+
```

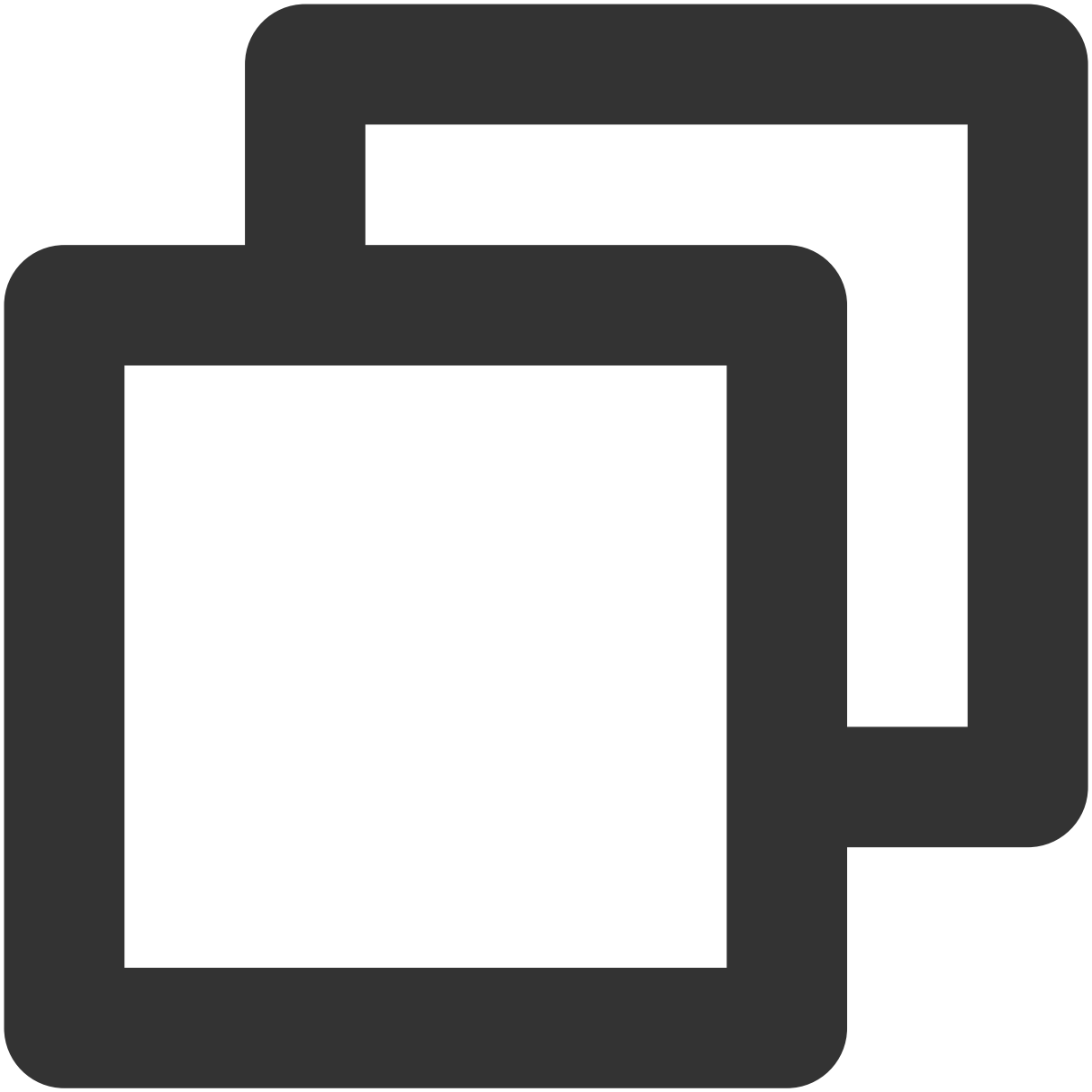
The data collected to CLS is as follows:



```
\\_\\_CONTENT__:2019-12-15 17:13:06,043 [main] ERROR com.test.logging.FooFactory:\\
```

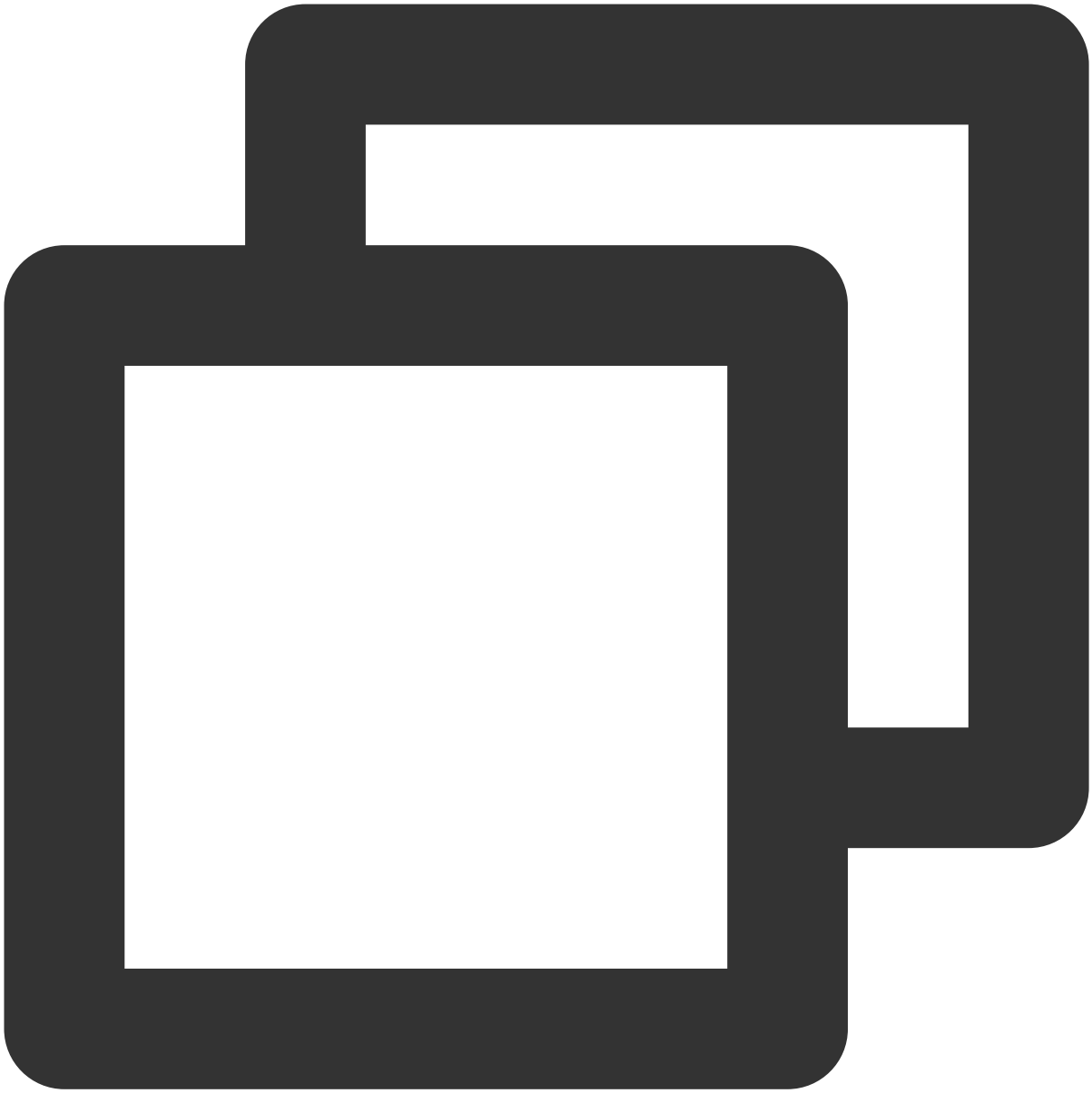
Full RegEx is often used to process structured logs. It parses a full log by extracting multiple key-value pairs based on a regex. For more information, see Full RegEx.

Assume that the raw data of a log is as follows:



```
10.135.46.111 - - [22/Jan/2019:19:19:30 +0800] "GET /my/course/1 HTTP/1.1" 127.0.0.
```

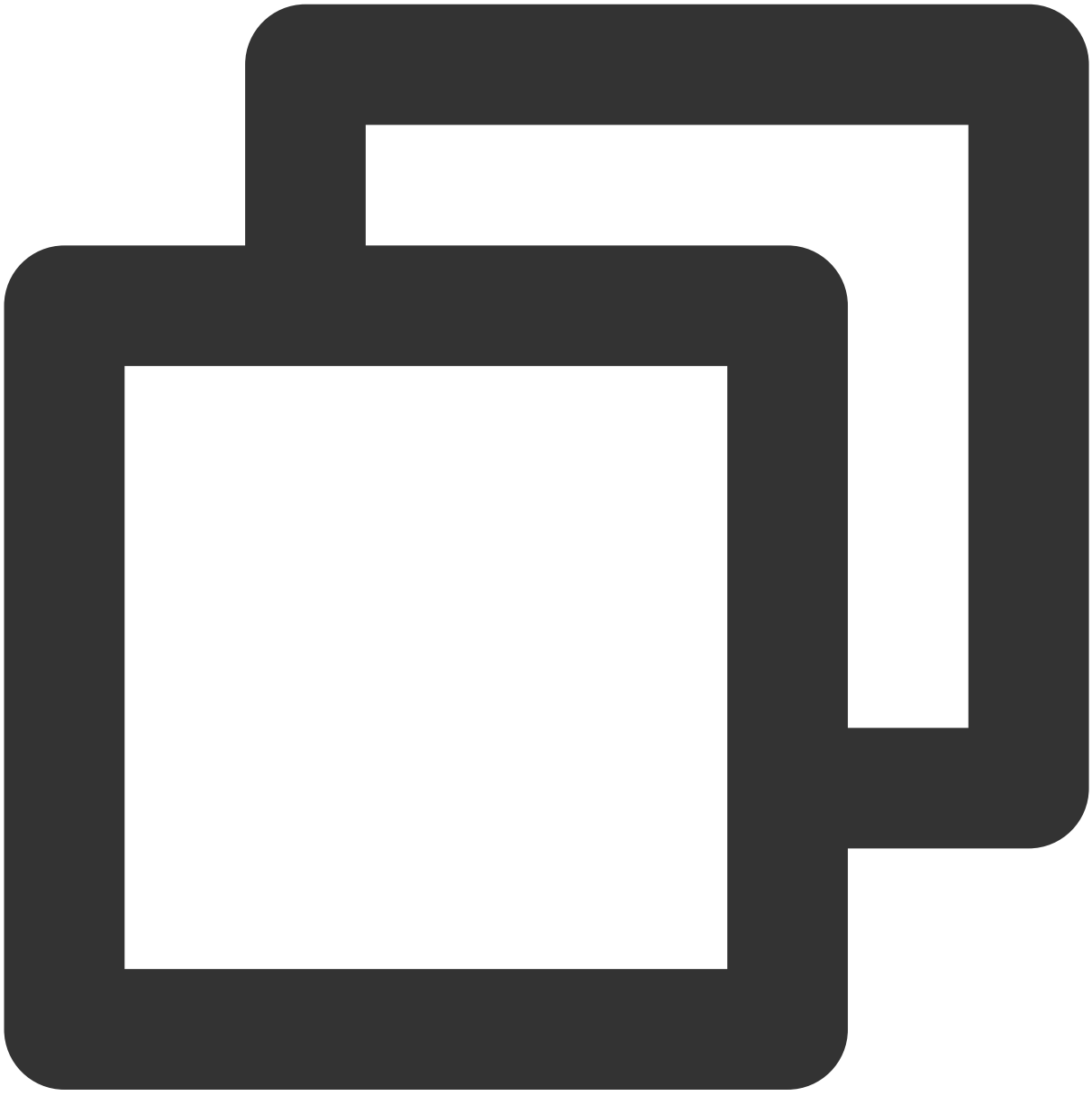
A sample of LogConfig is as follows:



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   clsDetail:
     topicId: xxxxxx-xx-xx-xx-xxxxxxxx
     # Full Regex
     logType: fullregex_log
     extractRule:
       # Regular expression, in which the corresponding values will be extracted ba
       logRegex: (\\S+)[^\\[]+(\\[[^:]+:\\d+:\\d+:\\d+\\s\\S+)\\s"(\\w+)\\s(\\S+)\\
       beginningRegex: (\\S+)[^\\[]+(\\[[^:]+:\\d+:\\d+:\\d+\\s\\S+)\\s"(\\w+)\\s(\
```

```
        # List of extracted keys, which are in one-to-one correspondence with the ex
        keys:   ['remote_addr','time_local','request_method','request_url','http_pro
```

The data collected to CLS is as follows:

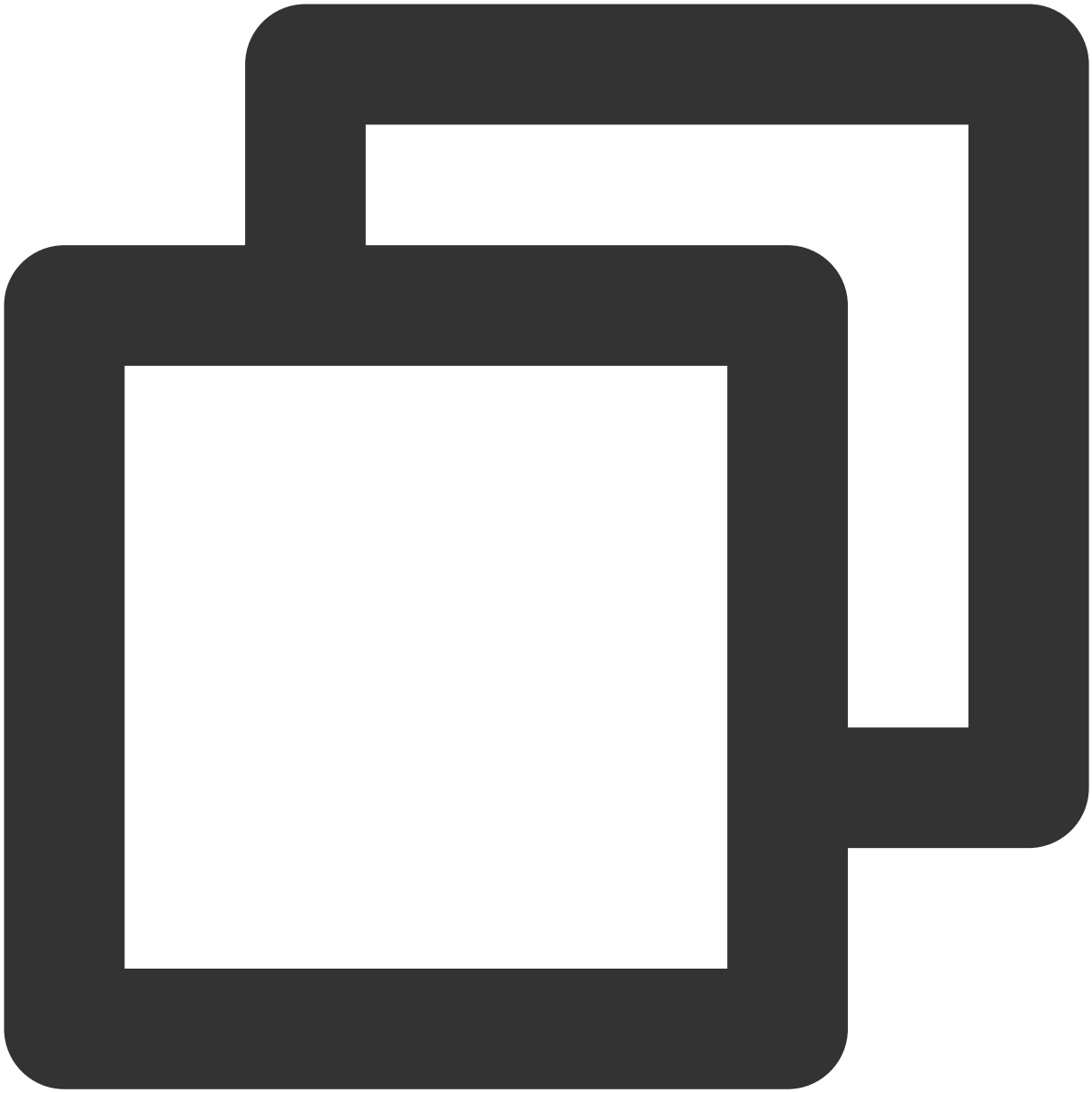


```
body_bytes_sent: 9703
http_host: 127.0.0.1
http_protocol: HTTP/1.1
http_referer: http://127.0.0.1/course/explore?filter%5Btype%5D=all&filter%5Bprice%5
http_user_agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firef
remote_addr: 10.135.46.111
```

```
request_length: 782
request_method: GET
request_time: 0.354
request_url: /my/course/1
status: 200
time_local: [22/Jan/2019:19:19:30 +0800]
upstream_response_time: 0.354
```

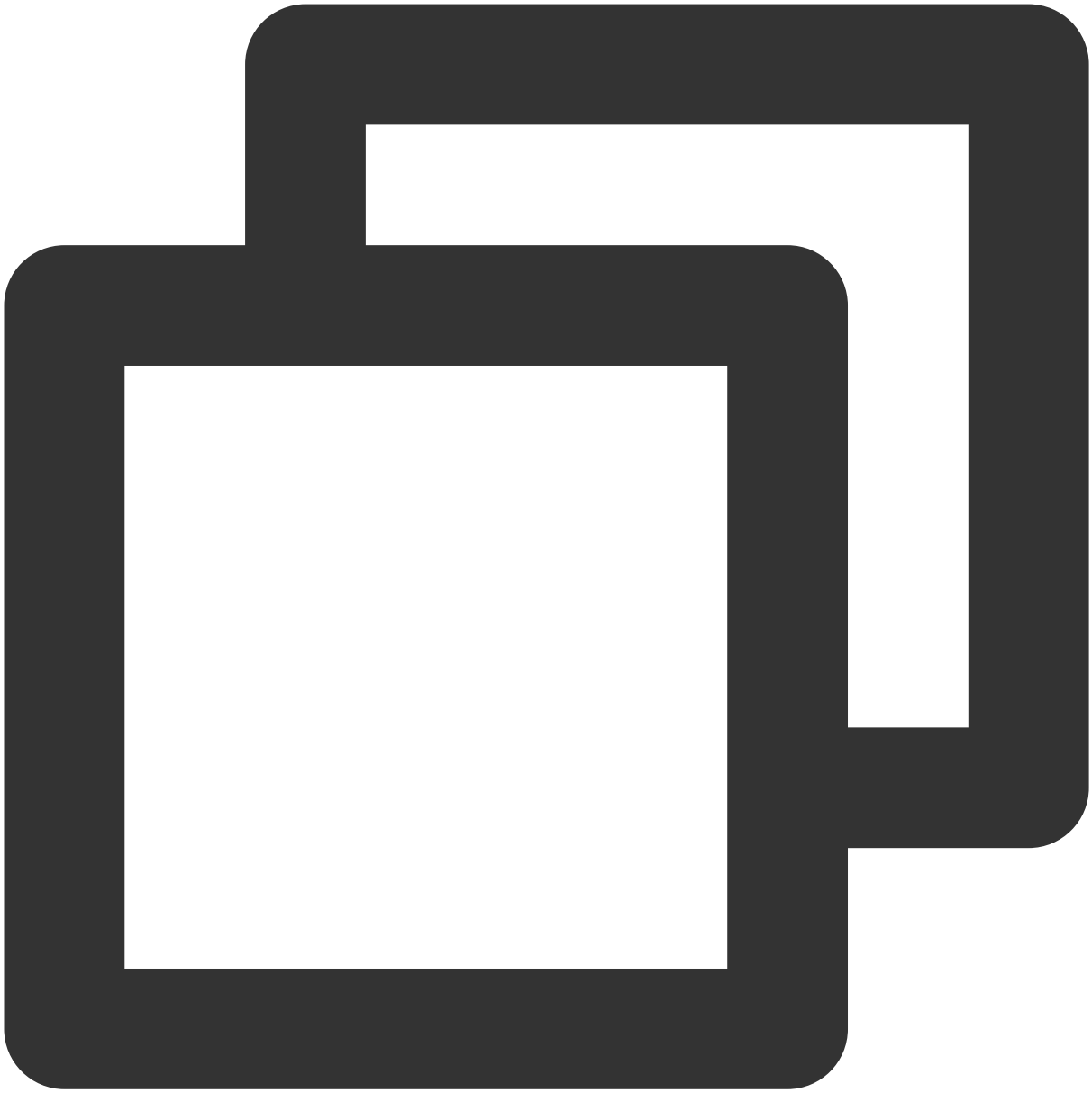
The multi-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete piece of log data that spans multiple lines in a log text file (such as Java program logs) based on a regular expression. If you don't need to extract key-value pairs, please configure it as instructed in full text in multi lines. For more information, see Full Text in Multi Lines.

Assume that the raw data of a log is as follows:



```
[2018-10-01T10:30:01,000] [INFO] java.lang.Exception: exception happened
   at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
   at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
   at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

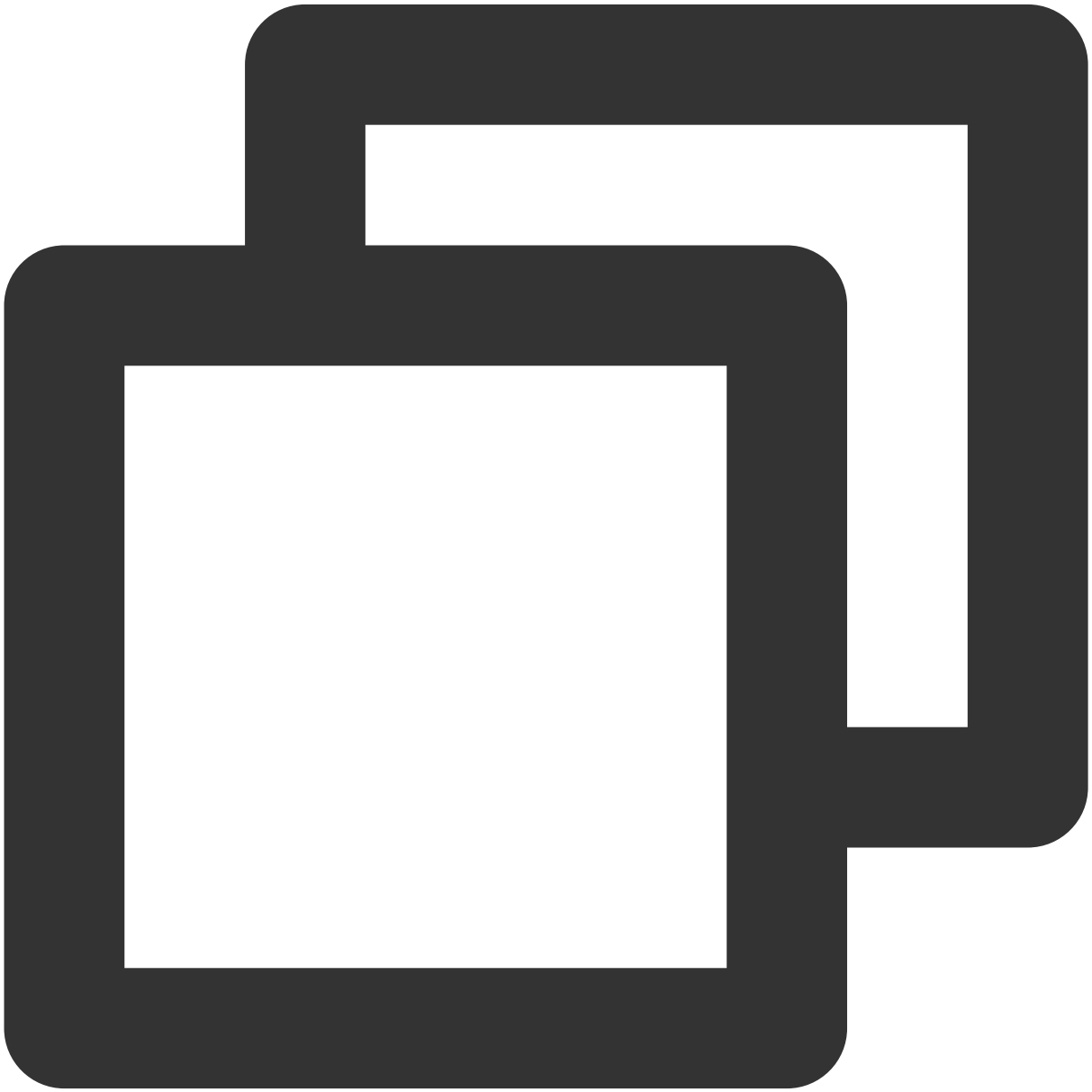
A sample of LogConfig is as follows:



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
  clsDetail:
    topicId: xxxxxx-xx-xx-xx-xxxxxxxx
        # Multiple lines - full regex
        logType: multiline_fullregex_log
        extractRule:
      # The first-line full regular expression: only a line that starts with a date
          beginningRegex: \\[\\d+-\\d+-\\w+:\\d+:\\d+,\\d+\\]\\s\\[\\w+\\]\\s.*
      # Regular expression, in which the corresponding values will be extracted bas
```

```
      logRegex: \\[(\\d+-\\d+-\\w+:\\d+:\\d+,\\d+)\\]\\s\\[(\\w+)\\]\\s(.*)
            # List of extracted keys, which are in one-to-one correspondence with t
            keys:
      - time
      - level
      - msg
```

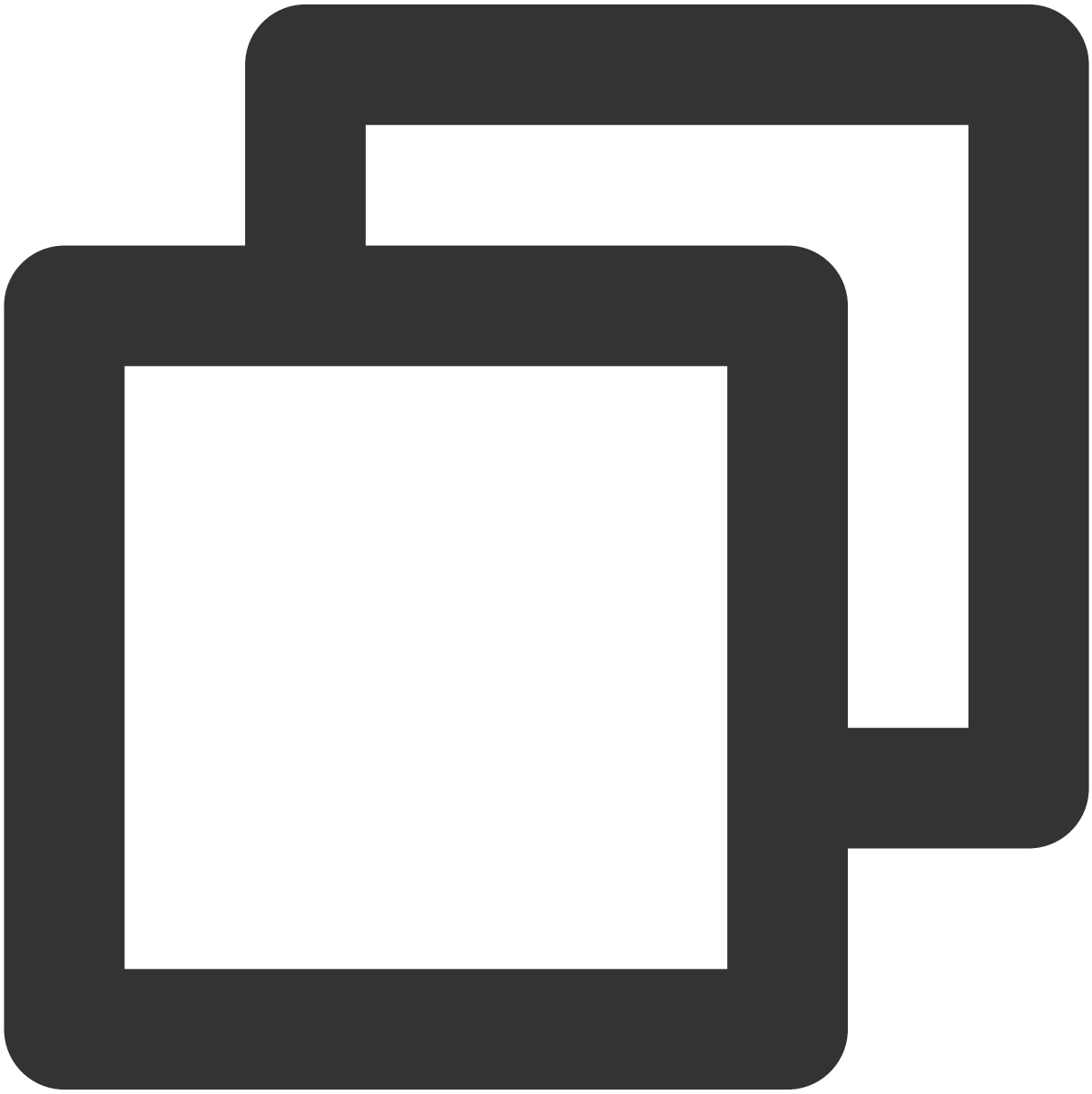
Based on the extracted key, the data collected to CLS is as follows:



```
time: 2018-10-01T10:30:01,000`
level: INFO`
```

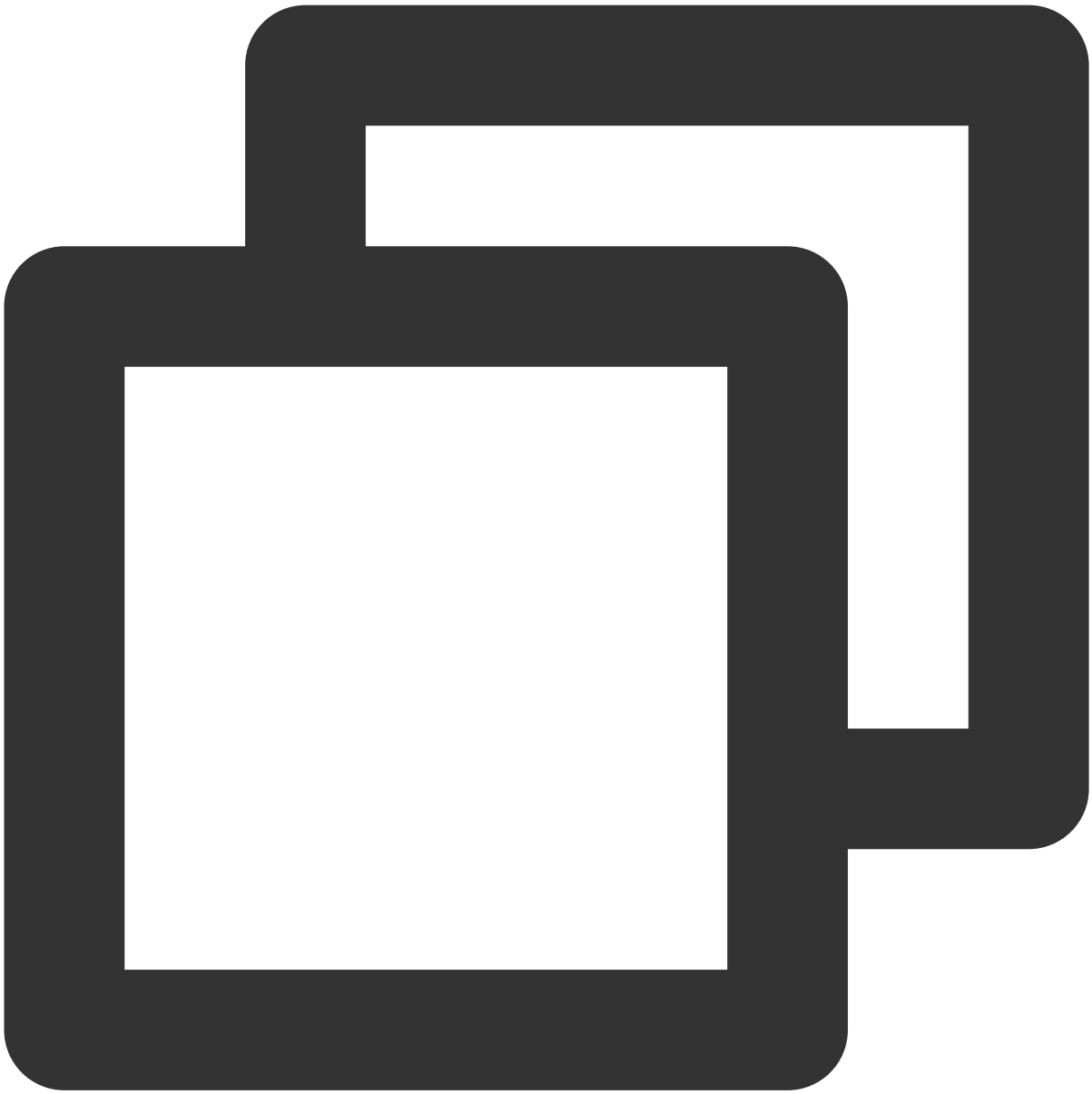
```
msg: java.lang.Exception: exception happened
   at TestPrintStackTrace.f(TestPrintStackTrace.java:3)
   at TestPrintStackTrace.g(TestPrintStackTrace.java:7)
   at TestPrintStackTrace.main(TestPrintStackTrace.java:16)
```

A JSON log automatically extracts the key at the first layer as the field name and the value at the first layer as the field value to structure the entire log. A full log ends with a line break `\\n` . For more information, see JSON Format.
Assume the raw data of a JSON log is as follows:



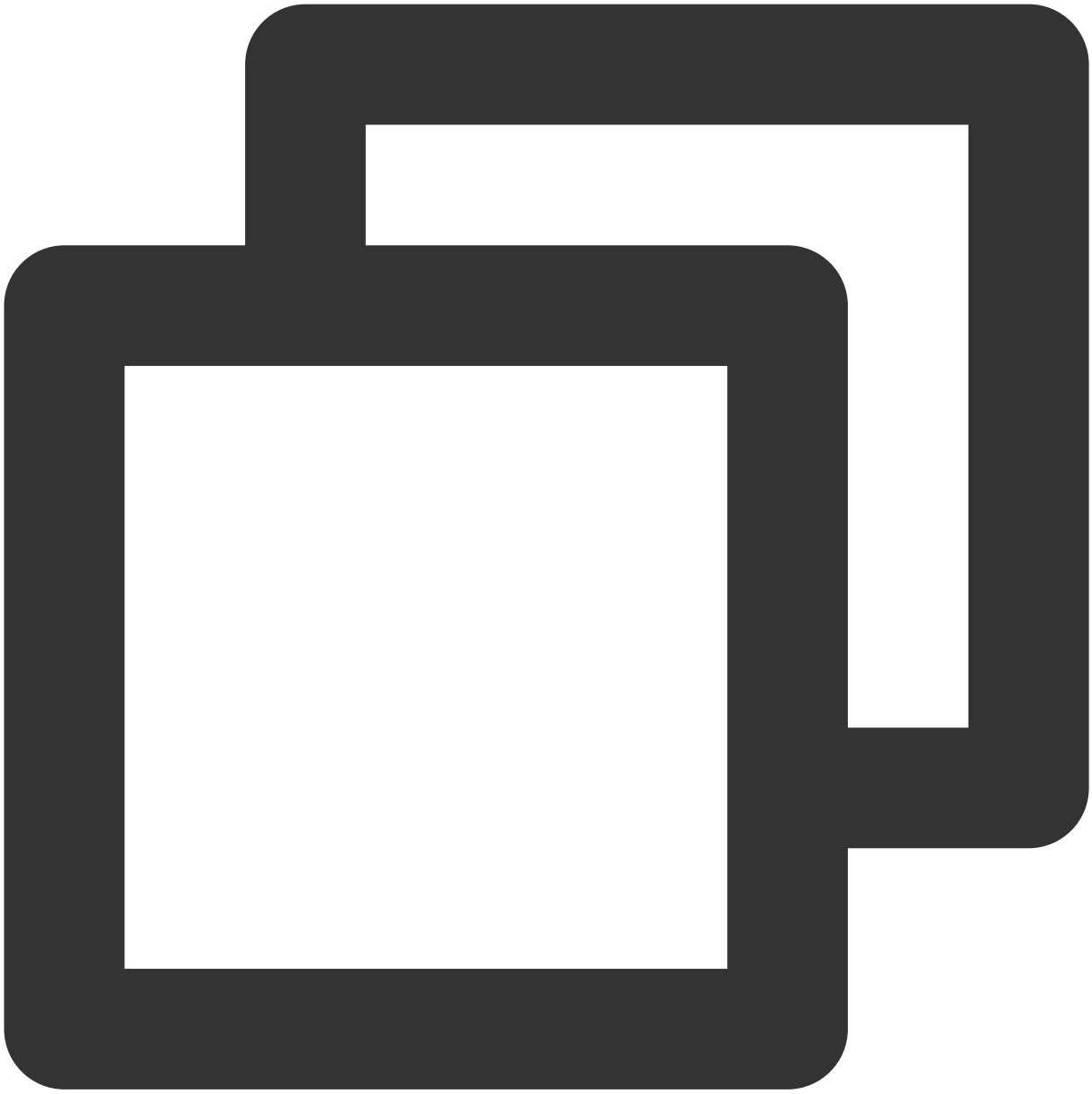
```
{"remote_ip":"10.135.46.111","time_local":"22/Jan/2019:19:19:34 +0800","body_sent":
```

A sample of LogConfig is as follows:



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   clsDetail:
     topicId: xxxxxx-xx-xx-xx-xxxxxxxx
     # JSON log
     logType: json_log
```

The data collected to CLS is as follows:

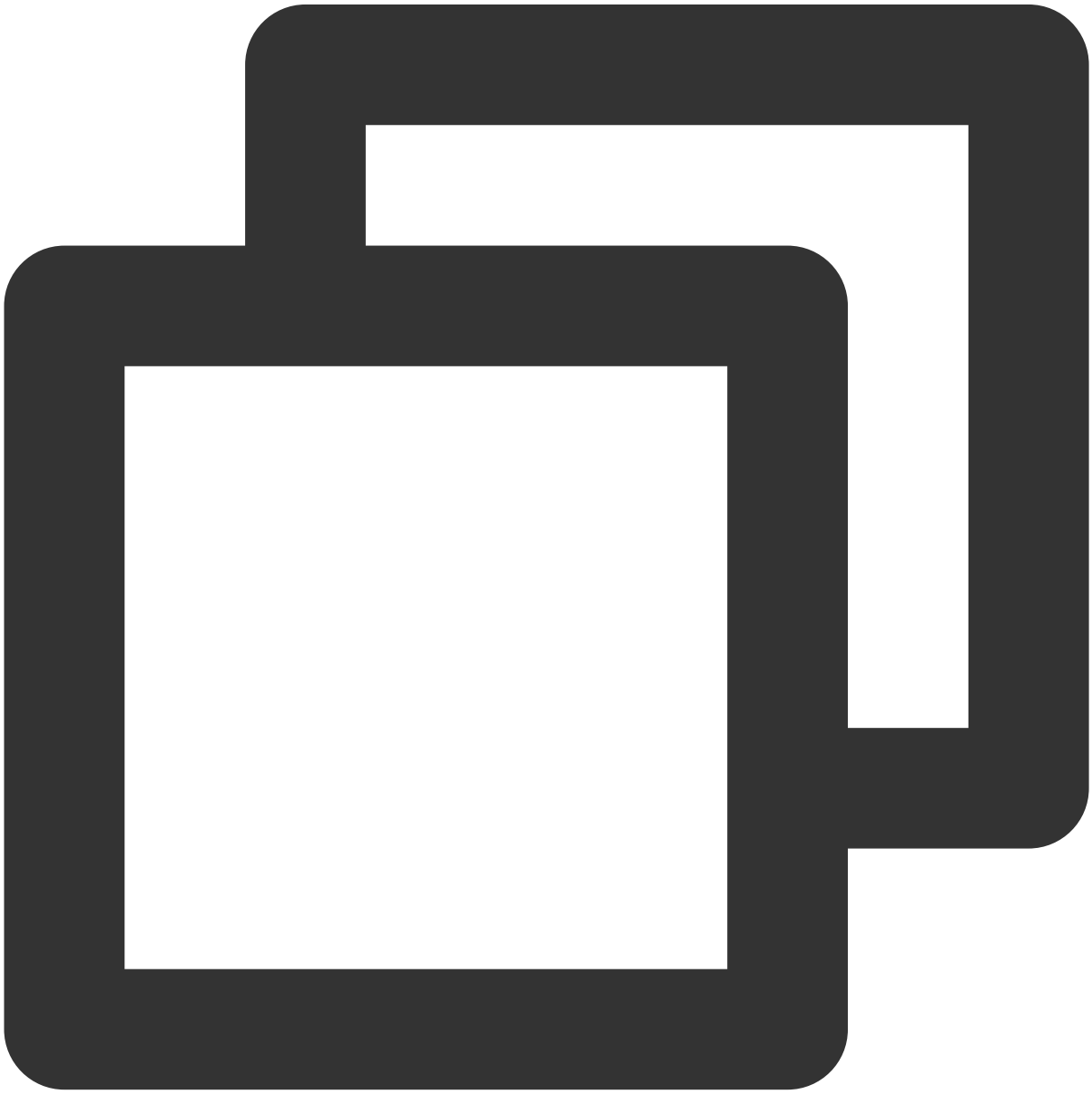


```
agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:64.0) Gecko/20100101 Firefox/64.0
body_sent: 23
http_host: 127.0.0.1
method: POST
referer: http://127.0.0.1/my/course/4
remote_ip: 10.135.46.111
request: POST /event/dispatch HTTP/1.1
response_code: 200
responsetime: 0.232
```

```
time_local: 22/Jan/2019:19:19:34 +0800
upstreamhost: unix:/tmp/php-cgi.sock
upstreamtime: 0.232
url: /event/dispatch
xff: -
```

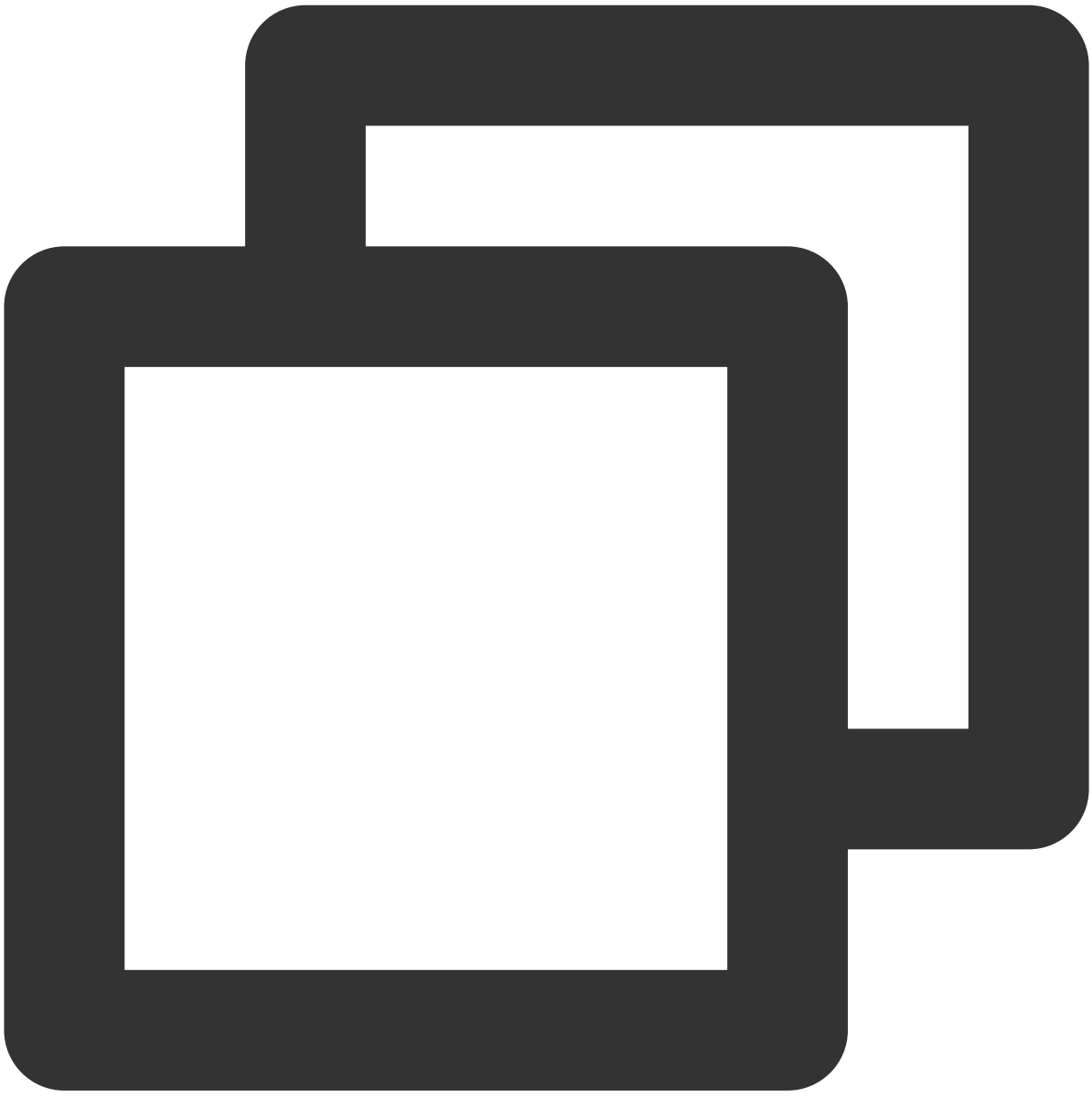
In this format, a log is structured based on the specified separator, and each complete log ends with a line break `\\n`. You need to define a unique key for each separate field for CLS to process separator-based logs. For more information, see Separator-based Format.

Assume the raw log is as follows:



```
10.20.20.10 ::: [Tue Jan 22 14:49:45 CST 2019 +0800] ::: GET /online/sample HTTP/1.
```

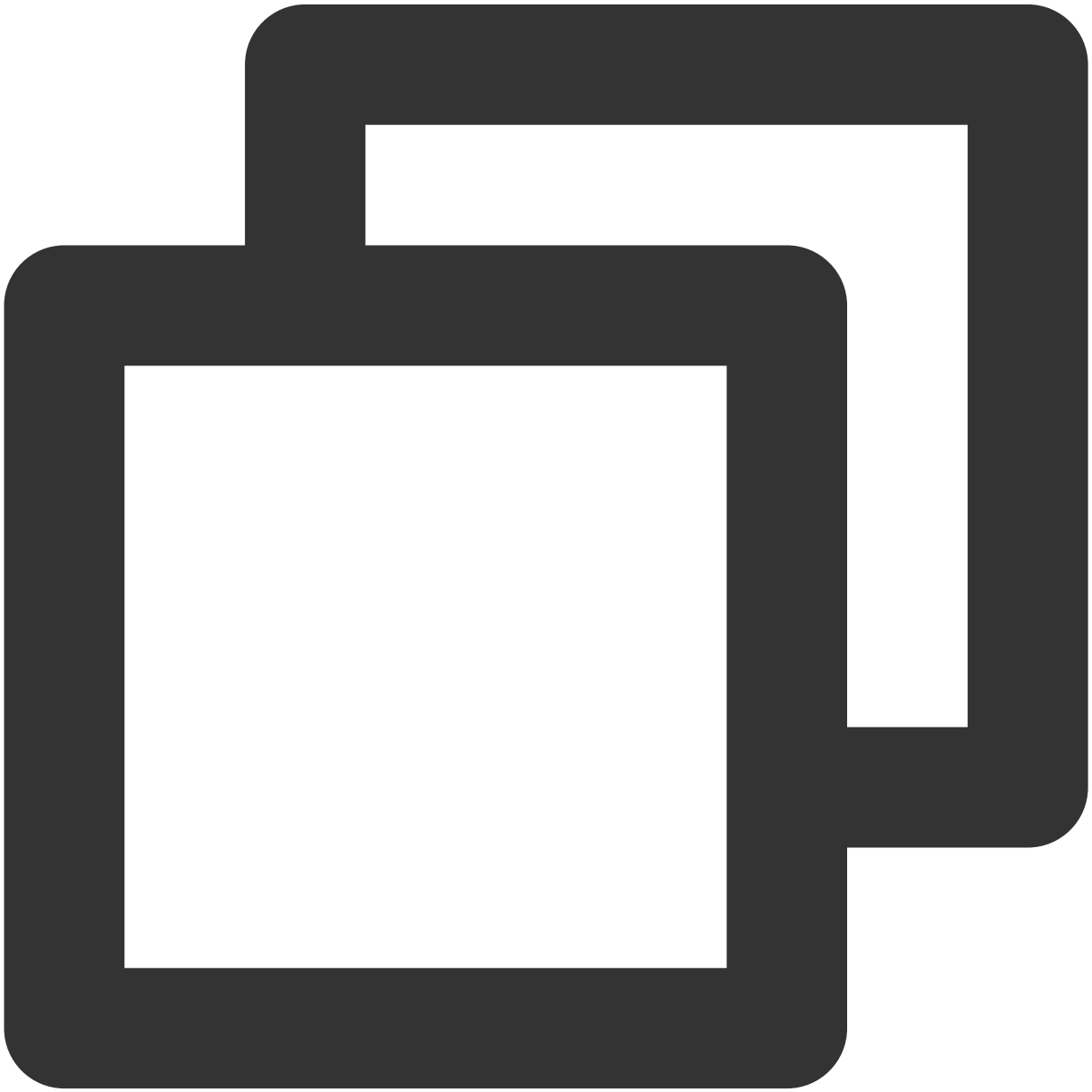
A sample of LogConfig is as follows:



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   clsDetail:
     topicId: xxxxxx-xx-xx-xx-xxxxxxxx
     # Separator log
```

```
      logType: delimiter_log
      extractRule:
        # Separator
        delimiter: ':::'
        # List of extracted keys, which are in one-to-one correspondence to the sepa
        keys: ['IP','time','request','host','status','length','bytes','referer']
```

The data collected to CLS is as follows:



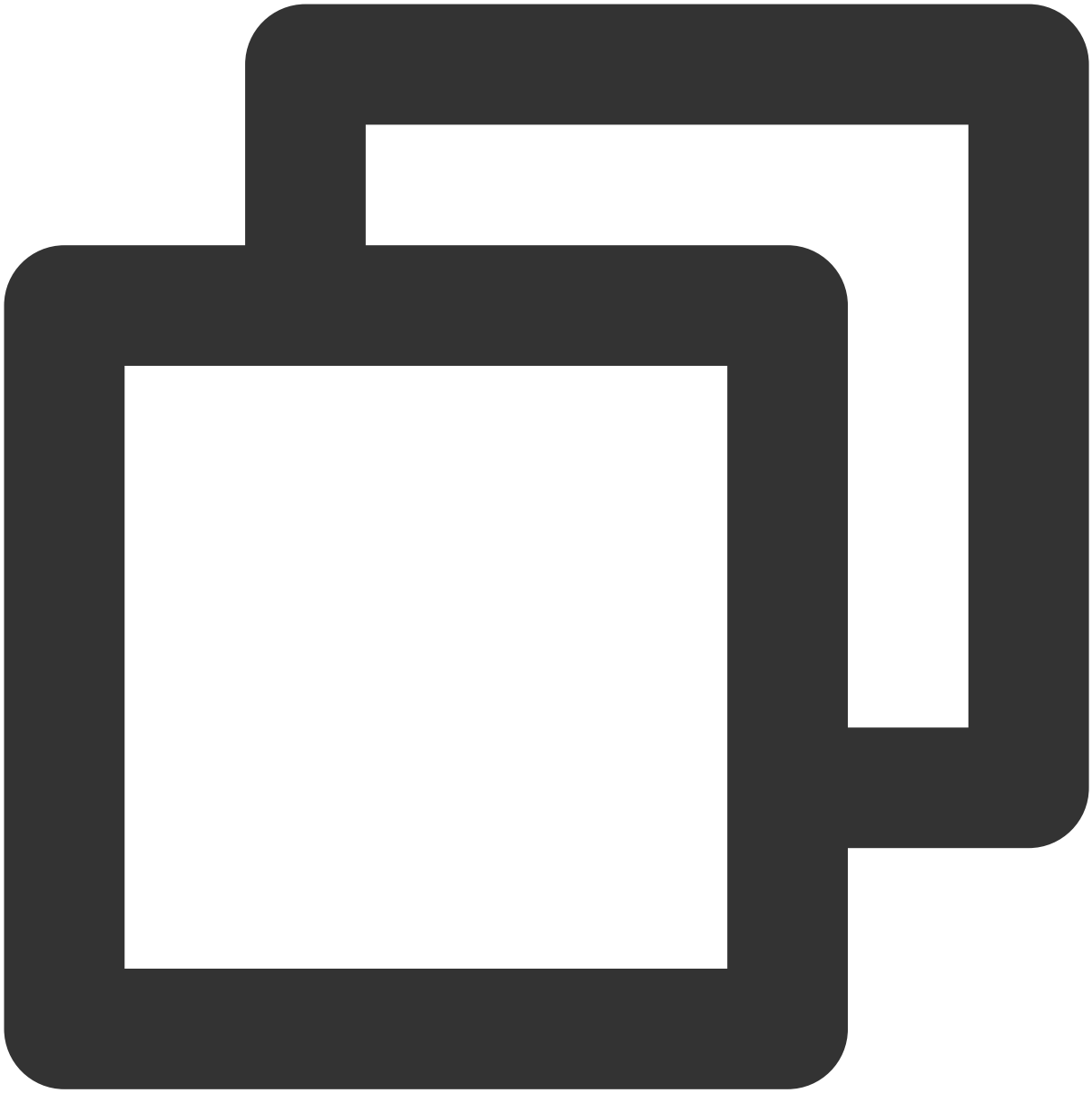
```
IP: 10.20.20.10
bytes: 35
```

```
host: 127.0.0.1
length: 647
referer: http://127.0.0.1/
request: GET /online/sample HTTP/1.1
status: 200
time: [Tue Jan 22 14:49:45 CST 2019 +0800]
```

Log Collection Types

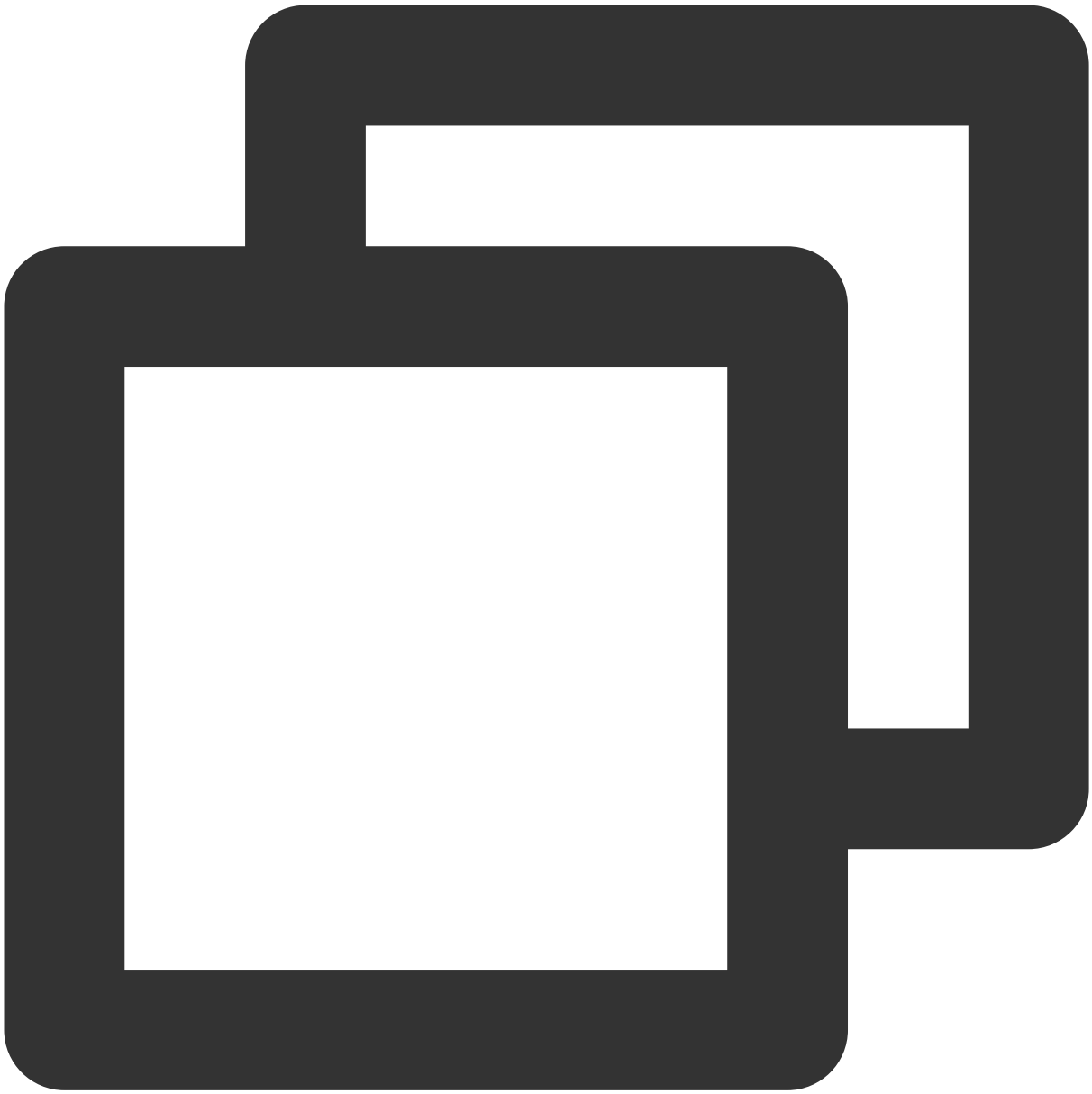
Container standard output

Sample 1: collecting the standard output of all containers in the default namespace



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   inputDetail:
     type: container_stdout
     containerStdout:
       namespace: default
       allContainers: true
 ...
```

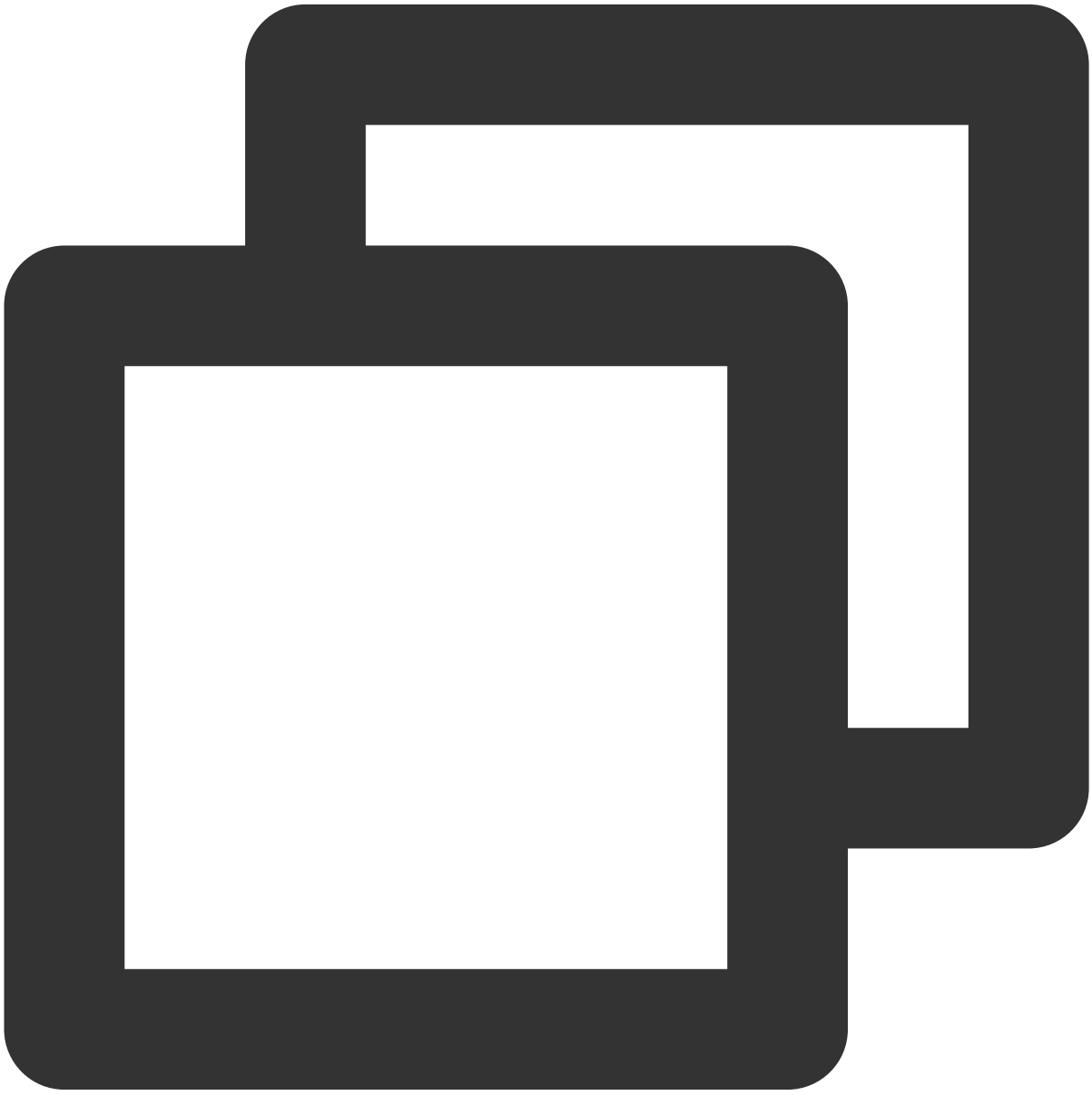
Sample 2: collecting the container standard output in the Pod that belongs to ingress-gateway deployment in the production namespace



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   inputDetail:
     type: container_stdout
     containerStdout:
       allContainers: false
       workloads:
```

```
        - namespace: production
          name: ingress-gateway
          kind: deployment
  ...
```

Sample 3: collecting the container standard output in the Pod whose Pod labels contain "k8s-app=nginx" under the production namespace

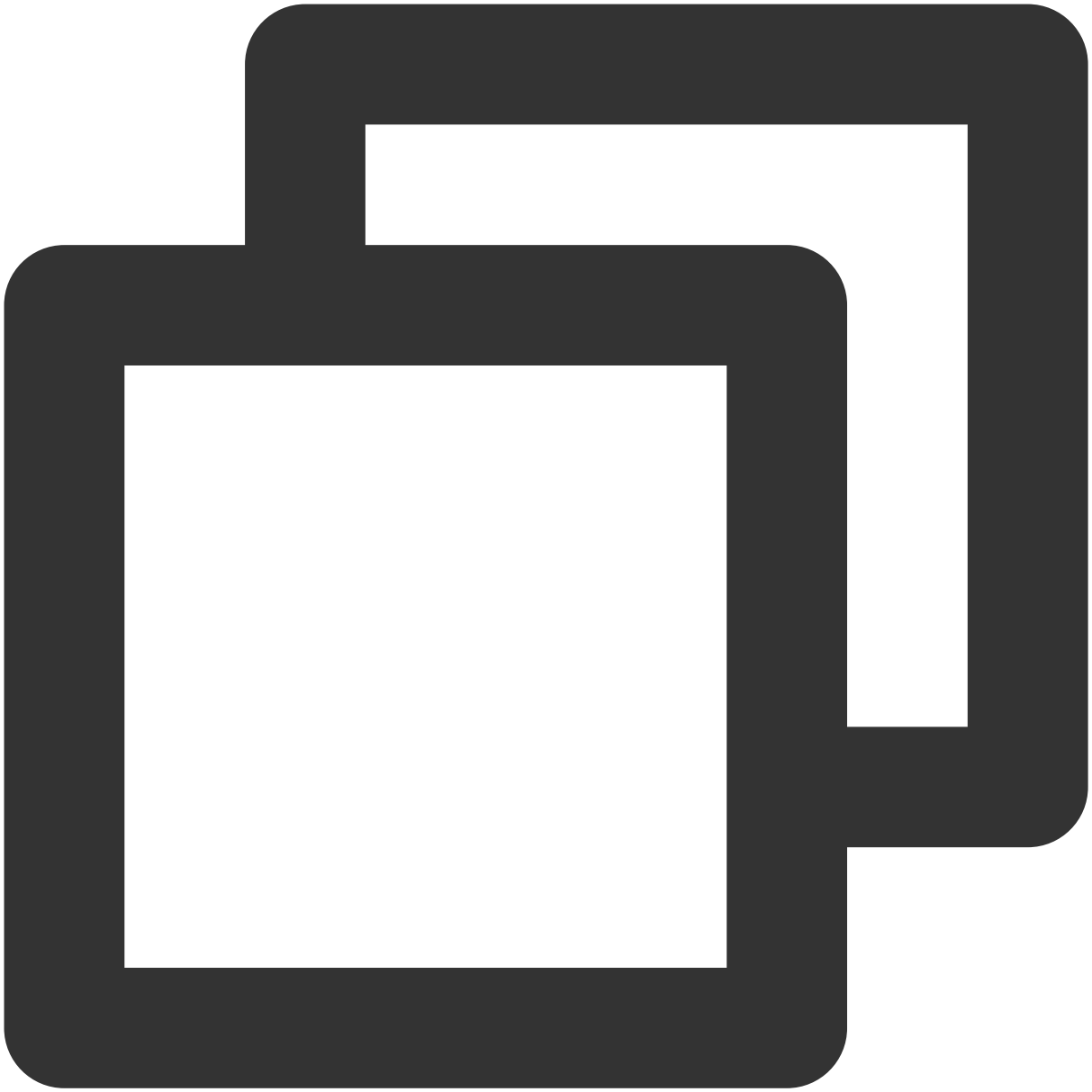


```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
```

```
spec:
   inputDetail:
     type: container_stdout
     containerStdout:
       namespace: production
       allContainers: false
       includeLabels:
         k8s-app: nginx
   ...
```

Container file

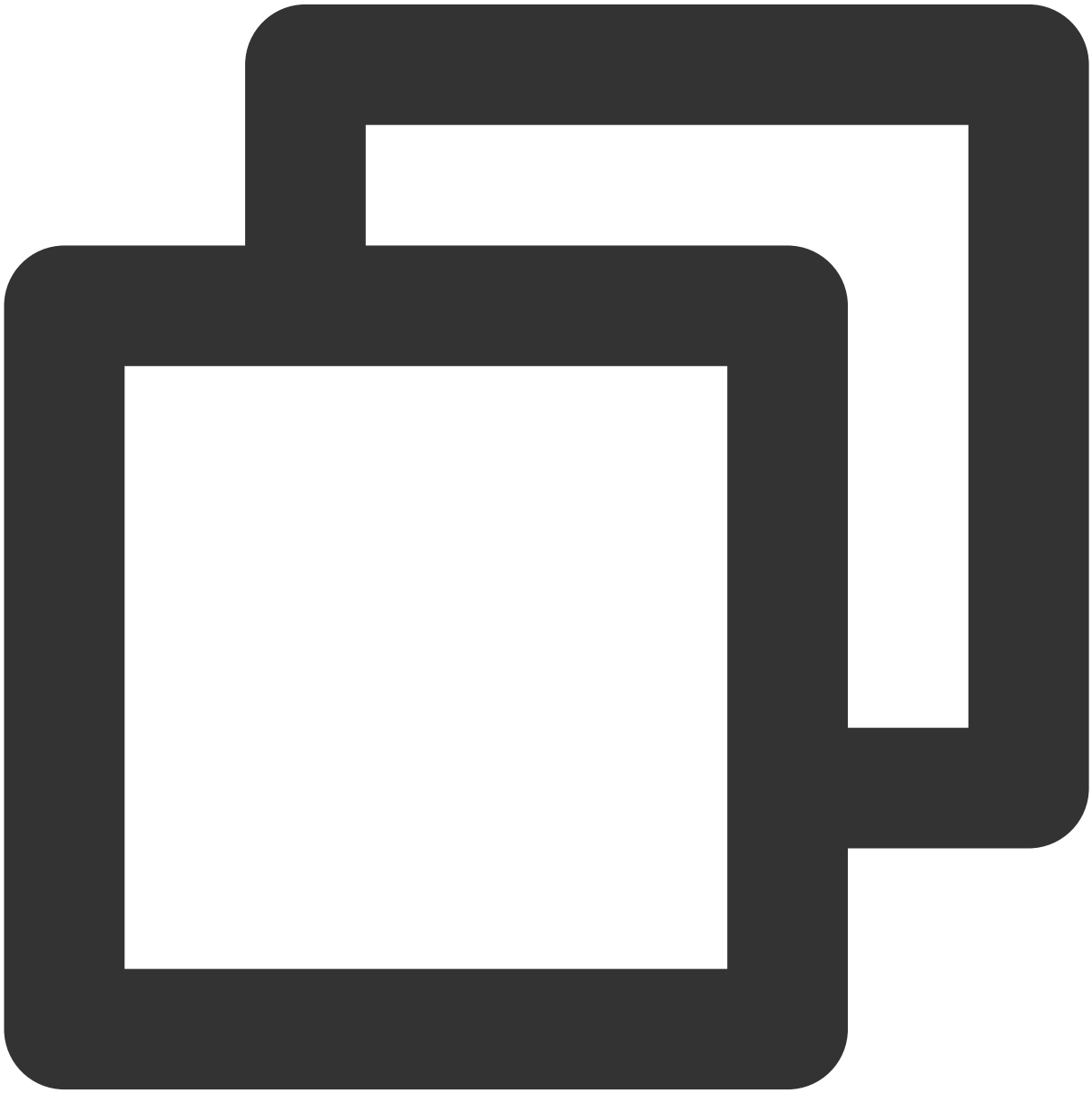
Sample 1: collecting the `access.log` file in the `/data/nginx/log/` path in the nginx container in the Pod that belongs to ingress-gateway deployment under the production namespace



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
spec:
   topicId: xxxxxx-xx-xx-xx-xxxxxxxx
   inputDetail:
     type: container_file
     containerFile:
       namespace: production
       workload:
         name: ingress-gateway
         type: deployment
```

```
        container: nginx
        logPath: /data/nginx/log
        filePattern: access.log
    ...
```

Sample 2: collecting the `access.log` file in the `/data/nginx/log/` path in the nginx container in the Pod whose pod labels contain "k8s-app=ingress-gateway" under the production namespace



```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig
```

```
spec:
   inputDetail:
     type: container_file
     containerFile:
       namespace: production
       includeLabels:
         k8s-app: ingress-gateway
       container: nginx
       logPath: /data/nginx/log
       filePattern: access.log
   ...
```

Metadata

For container standard output (`container_stdout`) and container files (`container_file`), in addition to the raw log content, the container metadata (for example, the ID of the container that generated the logs) also needs to be carried and reported to CLS. In this way, when viewing logs, users can trace the log source or search based on the container identifier or characteristics (such as container name and labels).

The following table lists the metadata:

Field	Description
cluster_id	ID of the cluster to which the log belongs
container_name	Name of the container to which the log belongs
image_name	Image name IP of the container to which the log belongs
namespace	Namespace of the Pod to which the log belongs
pod_uid	UID of the Pod to which the log belongs
pod_name	Name of the Pod to which the log belongs
pod_ip	IP of the Pod to which the log belongs
pod_lable_{label name}	Labels of the Pod to which the log belongs (for example, if a Pod has two labels: `app=nginx` and `env=prod` , the reported log will have two metadata entries attached: `pod_label_app:nginx` and `pod_label_env:prod`)

Using CRD to Collect Logs to Kafka

Last updated：2022-09-13 11:51:18

TKE Serverless not only supports uploading logs to CLS, but also supports collecting logs to self-built Kafka or CKafka.

Creating the CRD

If you want to collect logs to Kafka, you only need to define the CRD. The template is as follows:

```
apiVersion: cls.cloud.tencent.com/v1
kind: LogConfig ## Default value
metadata:
name: test ## CRD resource name, unique in the cluster
spec:
kafkaDetail:
brokers: xxxxxx # A required item, broker address, generally it is domain name:po
rt. If there are more than one address, separate them with ",".
topic: xxxxxx # A required item, topicID
messageKey: # An optional item. You can specify the Pod field as the key to uploa
d to the specified partition.
valueFrom:
fieldRef:
fieldPath: metadata.name
timestampKey: # The key of timestamp. Default value is @timestamp.
timestampFormat: # The format of timestamp. Default value is double.
inputDetail:
type: container_stdout ## Log collection type, including container_stdout (contai
ner standard output) and container_file (container file).
containerStdout: ## Container standard output
namespace: default ## The Kubernetes namespace of the container to be collected.
If this parameter is not specified, it indicates all namespaces.
allContainers: false ## Whether to collect the standard output of all containers
in the specified namespace
container: xxx ## Name of the container to be collected. This item can be left em
pty.
includeLabels: ## Only Pods that contain the specified labels will be collected.
k8s-app: xxx ## Only the logs generated by Pods with the configuration of "k8s-ap
p=xxx" in the Pod labels will be collected. This parameter cannot be specified at
the same time as workloads and allContainers=true.
workloads: ## Kubernetes workload to which the container Pod to be collected belo
ngs
```

```
- namespace: prod ## Workload namespace
name: sample-app ## Workload name
kind: deployment ## Workload type. Supported values include deployment, daemonse
t, statefulset, job, and cronjob.
container: xxx ## Name of the container to be collected. If this item is left emp
ty, it indicates all containers in the workload Pod will be collected.
containerFile: ## File in the container
namespace: default ## The Kubernetes namespace of the container to be collected.
A namespace must be specified.
container: xxx ## Name of the container to be collected. You can enter a * for th
is item.
includeLabels: ## Only Pods that contain the specified labels will be collected.
k8s-app: xxx ## Only the logs generated by Pods with the configuration of "k8s-ap
p=xxx" in the Pod labels are collected. This parameter cannot be specified at the
same time as workload.
workload: ## Kubernetes workload to which the container Pod to be collected belon
gs
name: sample-app ## Workload name
kind: deployment ## Workload type. Supported values include deployment, daemonse
t, statefulset, job, and cronjob.
logPath: /opt/logs ## Log folder. Wildcards are not supported.
filePattern: app_*.log ## Log file name. It supports the wildcards "*" and "?".
"*" matches multiple random characters, and "?" matches a single random characte
r.
```

Note

If you are unable to collect logs, please terminate and recreate the Pod and try it again.

Audit Management
Cluster Audit

Last updated：2023-05-06 17:36:46

Note:
From now to December 31, 2021, users are exempt from CLS service fees incurred by audit log/event data generated by TKE Serverless for auto-created logsets or auto-created log topics in existing logsets.

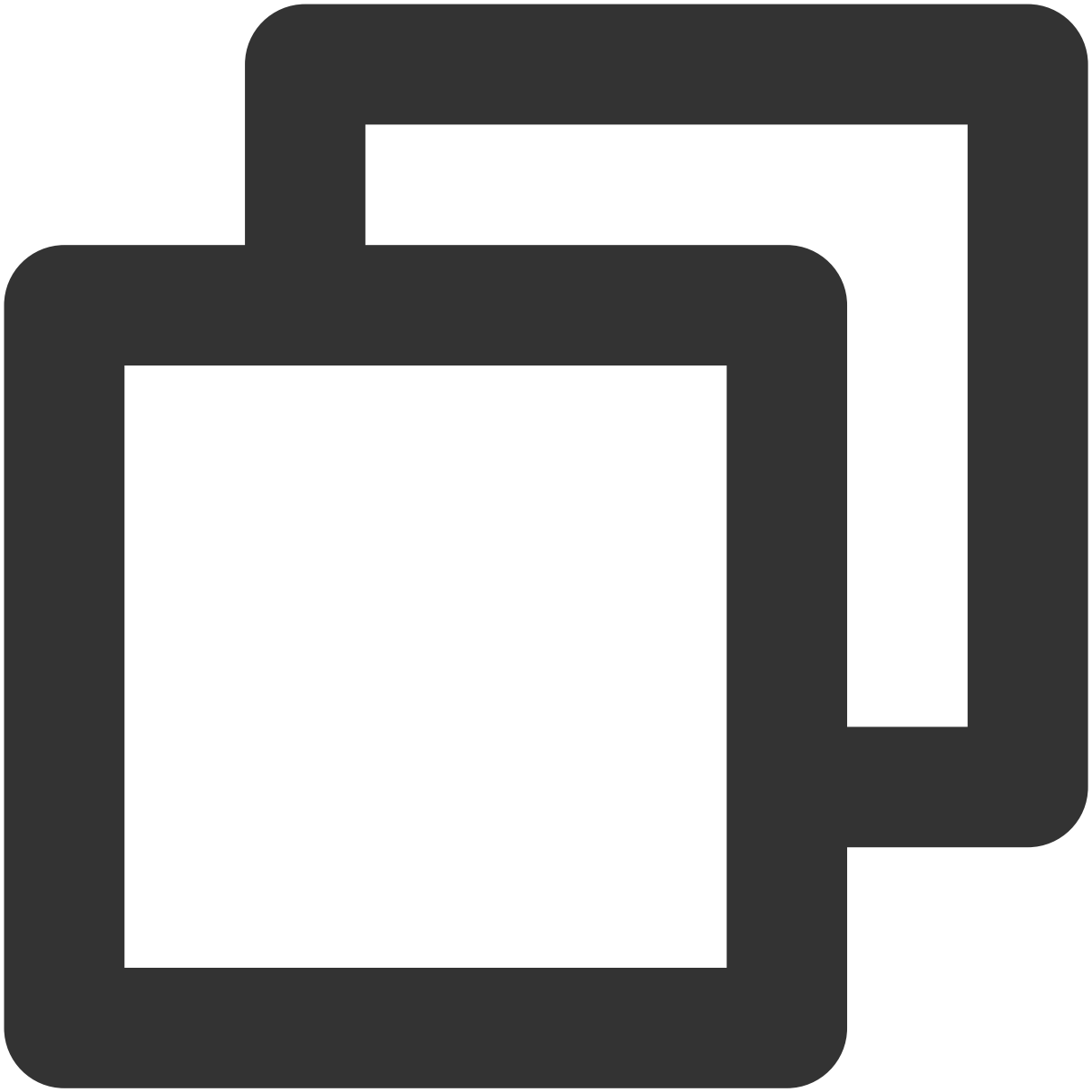
Overview

Cluster audit is a feature based on Kubernetes Auditing that can store and search the records of JSON logs with configurable policies generated by kube-apiserver. This feature records the access events of kube-apiserver and records the activities of each user, admin, or system component that has an impact on the cluster in sequence.

Advantages

The cluster audit feature provides another cluster monitoring dimension different from metrics. After cluster audit is enabled, Kubernetes can record every audit log that operates on the cluster. An audit log is a structured record in JSON format, and includes three parts: metadata, requestObject, and responseObject. The metadata (containing the request context information, such as who initiated the request, where it was initiated, and the accessed URI) is a required part. requestObject and responseObject are optional, depending on the audit level. You can learn the following information from logs:
Activities that occur in the cluster.
Activity occurrence time and objects.
Activity triggering time, triggering positions, and observation points.
Activity results and subsequent processing.

An example of how to read the audit log



```
{
  "kind":"Event",
  "apiVersion":"audit.k8s.io/v1",
  "level":"RequestResponse",
  "auditID":0a4376d5-307a-4e16-a049-24e017******,
  "stage":"ResponseComplete",
  // What happened?
  "requestURI":"/apis/apps/v1/namespaces/default/deployments",
  "verb":"create",
  // Who initiated the request?
  "user":{
```

```
      "username":"admin",
        "uid":"admin",
        "groups":[
          "system:masters",
          "system:authenticated"
        ]
    },
    // Where was it initiated?
    "sourceIPs":[
      "10.0.6.68"
    ],
    "userAgent":"kubectl/v1.16.3 (linux/amd64) kubernetes/ald64d8",
    // What happened?
    "objectRef":{
      "resource":"deployments",
      "namespace":"default",
      "name":"nginx-deployment",
      "apiGroup":"apps",
      "apiVersion":"v1"
    },
    // What's the result?
    "responseStatus":{
      "metadata":{
      },
      "code":201
    },
    // Request and response details
    "requestObject":Object{...},
    "responseObject":Object{...},
    // When did it start/end?
    "requestReceivedTimestamp":"2020-04-10T10:47:34.315746Z",
    "stageTimestamp":"2020-04-10T10:47:34.328942Z",
    // Reason for accepting/rejecting the request
    "annotations":{
      "authorization.k8s.io/decision":"allow",
      "authorization.k8s.io/reason":""
    }
}
```

TKE Serverless Cluster Audit Policy

Audit level (level)

Unlike common logs, the level of Kubernetes audit logs is more like a kind of verbose configuration, which is used to indicate the degree of detail of the recorded information. There are four audit levels, as listed in the following table:

Parameter	Description
None	Nothing is recorded.
Metadata	The metadata of the request (for example, user, time, resources, and operation) is recorded, excluding the request message body and response message body.
Request	The metadata and request message body are recorded, excluding the response message body.
RequestResponse	All the information is recorded, including the metadata, request message body, and response message body.

Audit stage (stage)

Logs can be recorded at different stages, as listed in the following table:

Parameter	Description
RequestReceived	The log is recorded immediately after a request is received.
ResponseStarted	The log is recorded after the message header of the response is sent. This parameter only applies to persistent connection requests, such as WATCH.
ResponseComplete	The log is recorded after the entire response is sent.
Panic	An error occurs to the internal server and the request fails.

Audit policy

By default, TKE serverless clusters record audit logs when receiving requests. For most operations, audit logs at the RequestResponse level are recorded. The following list shows the exceptions:

For GET, LIST, and WATCH requests, logs at the Request level are recorded.

For requests of Secret, ConfigMap, or TokenReview resources, logs at the Metadata level are recorded.

Logs will not be recorded for the following requests:

Requests sent by `system:kube-proxy` for monitoring endpoint, service, or service/status resources.

GET requests sent by `system:unsecured` for ConfigMap resources in the kube-system namespace.

GET requests sent by kubelet for node or node/status resources.

GET and UPDATE requests sent by `system:kube-controller-manager` , `system:kube-scheduler` , or `system:serviceaccount:endpoint-controller` for endpoint resources in the kube-system namespace.

GET requests sent by `system:apiserver` for namespace, namespace/status, or namespace/finalize resources.

Requests sent to URLs that match `/healthz*` , `/version` , or `/swagger*` .

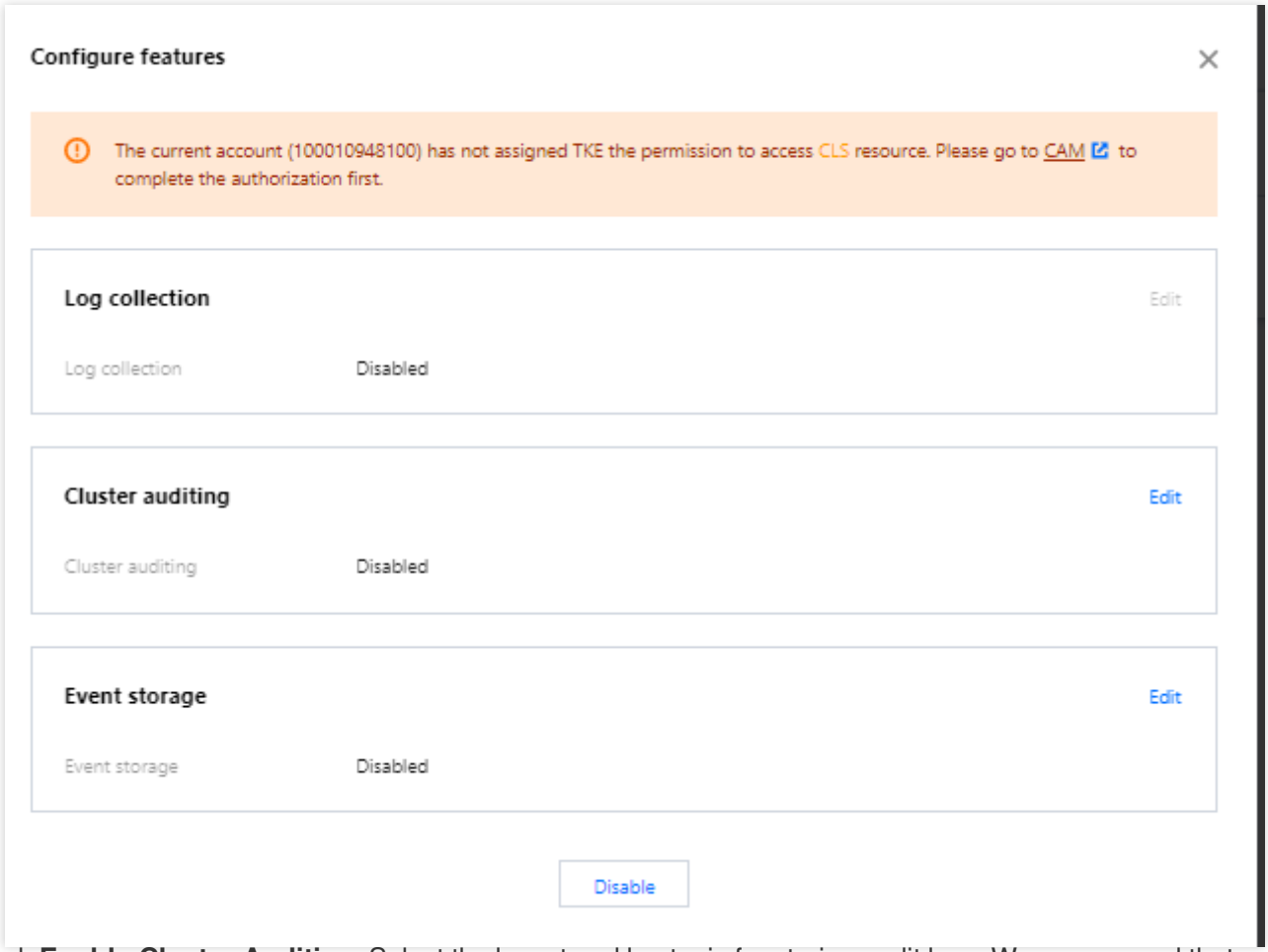
Directions

Enabling cluster audit

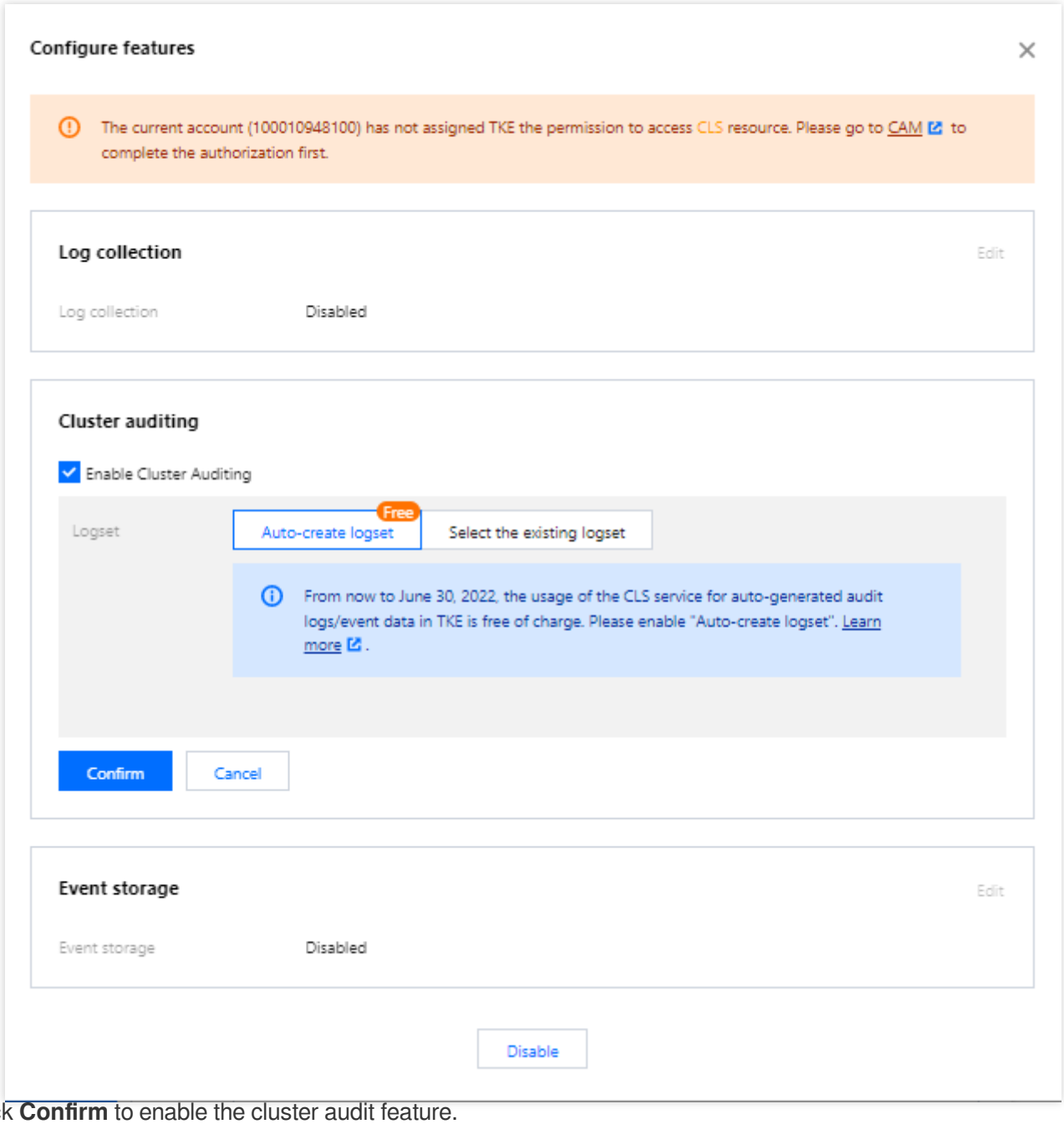
Note

To enable the cluster audit feature, you need to restart kube-apiserver. We recommend that you do not frequently enable and disable the feature.

1. Log in to the TKE console.
2. In the left sidebar, choose **Operation Management** > **Feature Management**.
3. On the **Feature Management** page, select a region and the **Serverless cluster** type.
4. Locate the cluster for which you want to enable the cluster audit feature in the following cluster list. Click **Set** in the **Operation** column on the right.
5. In the **Configure features** pop-up window, click **Edit** for the **Cluster Auditing** feature, as shown below:



6. Check **Enable Cluster Auditing**. Select the logset and log topic for storing audit logs. We recommend that you select **Auto-create Logset**, as shown below:



7. Click **Confirm** to enable the cluster audit feature.

Audit Dashboard

Last updated：2022-09-14 15:30:43

Overview

TKE Serverless cluster provides out-of-the-box audit dashboards and can automatically configure dashboards of audit overview, K8s object operation overview, and aggregated search for the clusters with cluster auditing enabled. With user-defined filters and built-in CLS global search, TKE Serverless cluster makes it convenient for you to observe and search for cluster operations, so as to promptly find and locate problems.

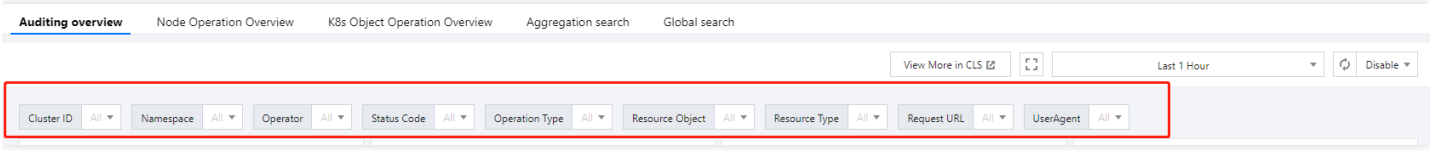
Feature Description

Four dashboards are configured in **Auditing Search**, namely **Audit Overview**, **K8s Object Operation Overview**, **Aggregated Search**, and **Global Search**. Follow the steps below to enter the **Auditing Search** page to use the features:

1. Log in to the TKE console.
2. Enable cluster auditing. For more information, see Cluster Auditing.
3. On the left sidebar, click **Cluster Ops** > **Auditing Search**.

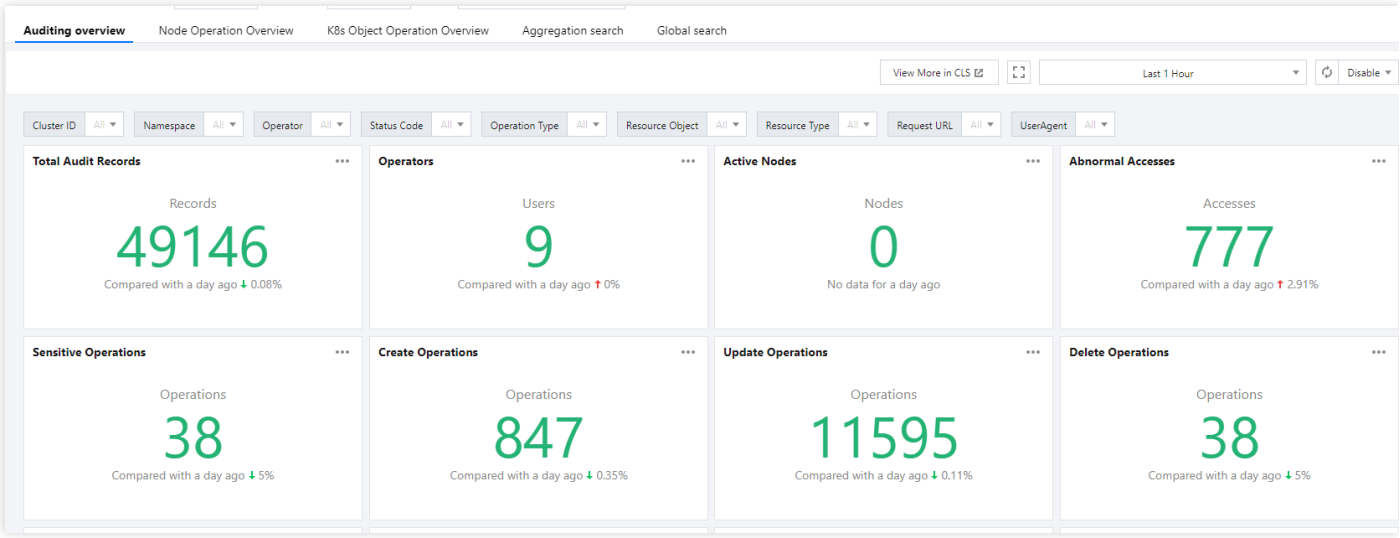
Audit overview

If you want to observe the operations of the API server in the entire cluster, you can set filters on the **Audit Overview** page to view the aggregated statistics of the core audit logs and display the data comparison within a period, for example, core audit log statistics, distribution, important operation trends.
You can customize up to filters as needed.

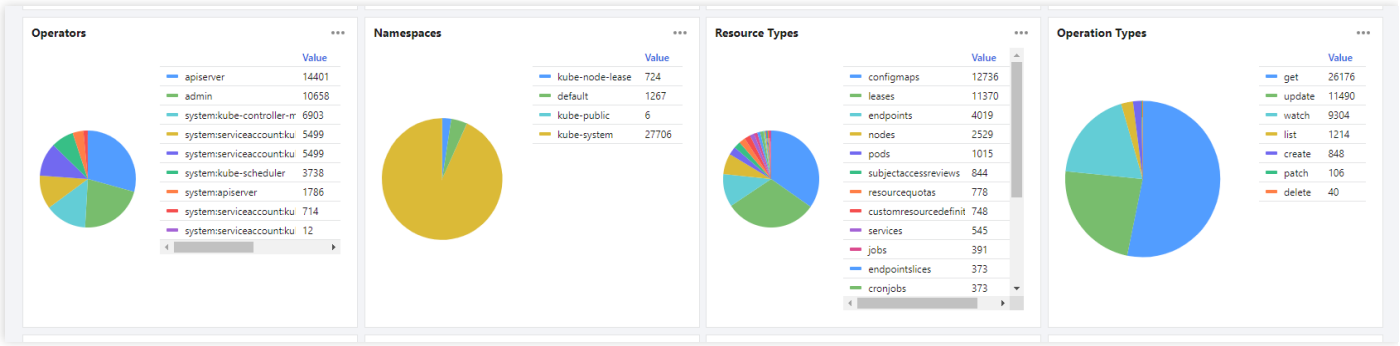


You can view more statistics on this page as shown below:

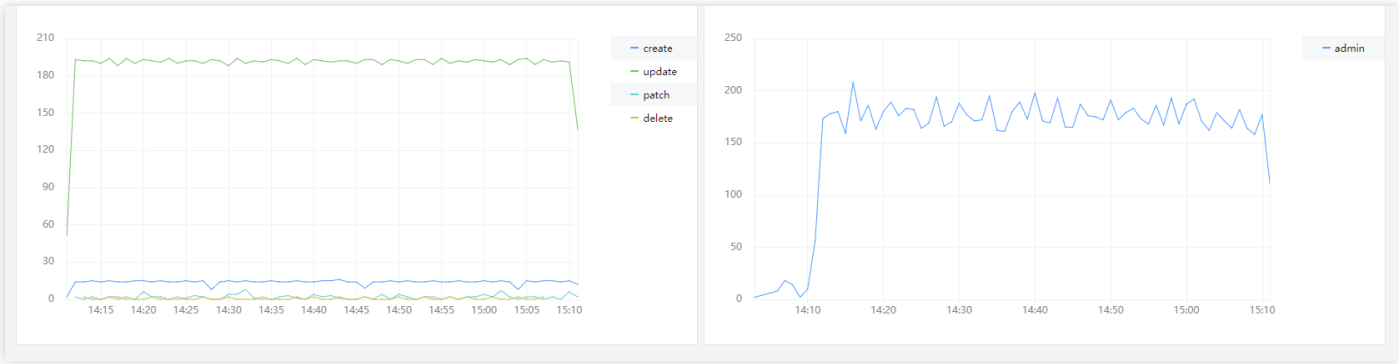
- Core audit log statistics dashboard:



- Distribution dashboard:



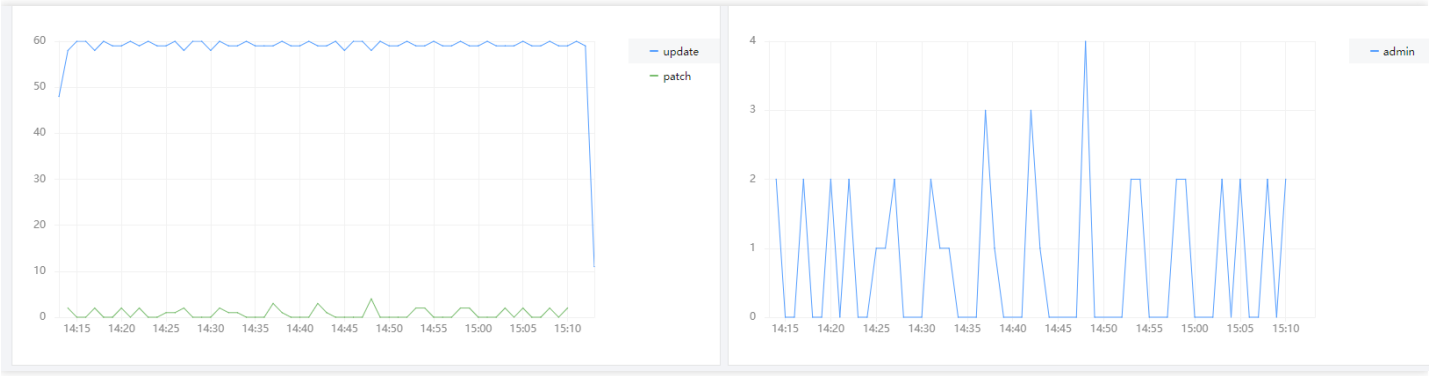
- Important operation trend dashboard:



K8s object operation overview

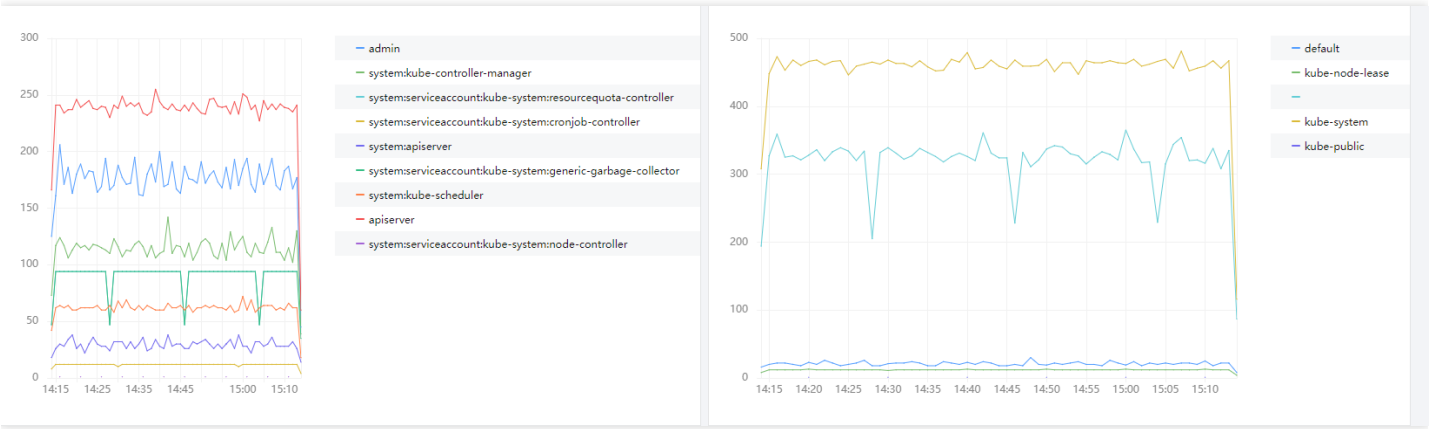
When you need to troubleshoot issues with K8s objects (such as a certain workload), you can switch to the **K8s Object Operation Overview** tab, where you can set filters to view the operation overview, operators, and audit log

lists of various types of K8s objects .



Aggregated search

If you want to observe the distribution trend of audit logs in a certain dimension, you can set filters on the **Aggregated Search** tab to view the sequence diagrams of important operations, including user, namespace, operation type, status code, resource type, and audit log list.



Global search

The global search dashboard with built-in CLS search and analysis makes it convenient for you to quickly search all audit logs in the TKE console .

Configuring alarm by dashboard

You can configure alarms based on the preset dashboards as instructed below, so that alarms will be triggered when the configured conditions are reached:

1. Click ••• on the right of the target dashboard and select **Quickly Add Alarm** from the drop-down list .
2. For more information on how to configure alarm policies, see Configuring Alarm Policies.

Event Management
Event Storage

Last updated：2022-09-14 16:55:25

Note：
From now to September 5, 2022, users are **exempt from CLS service fees** incurred by audit log/event data generated by TKE Serverless cluster for auto-created log topics. For details, see Note on Free Log Storage for TKE Audit and Event Center.

Overview

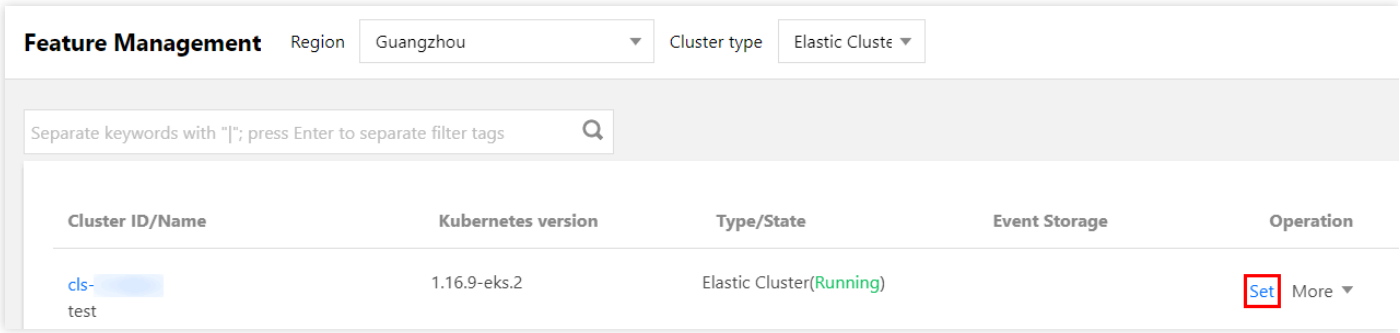
Kubernetes Events contains information about the operations of Kubernetes clusters and the scheduling of various resources, which can help OPS personnel monitor daily changes in resources and locate problems. TKE Serverless cluster supports event persistence for all clusters and also supports the search of event flows by using PAAS services provided by Tencent Cloud or open-source software tools. After enabling this feature, your cluster events will be exported to the configured storage in real time. This document describes how to enable and use persistent storage of cluster events.

Directions

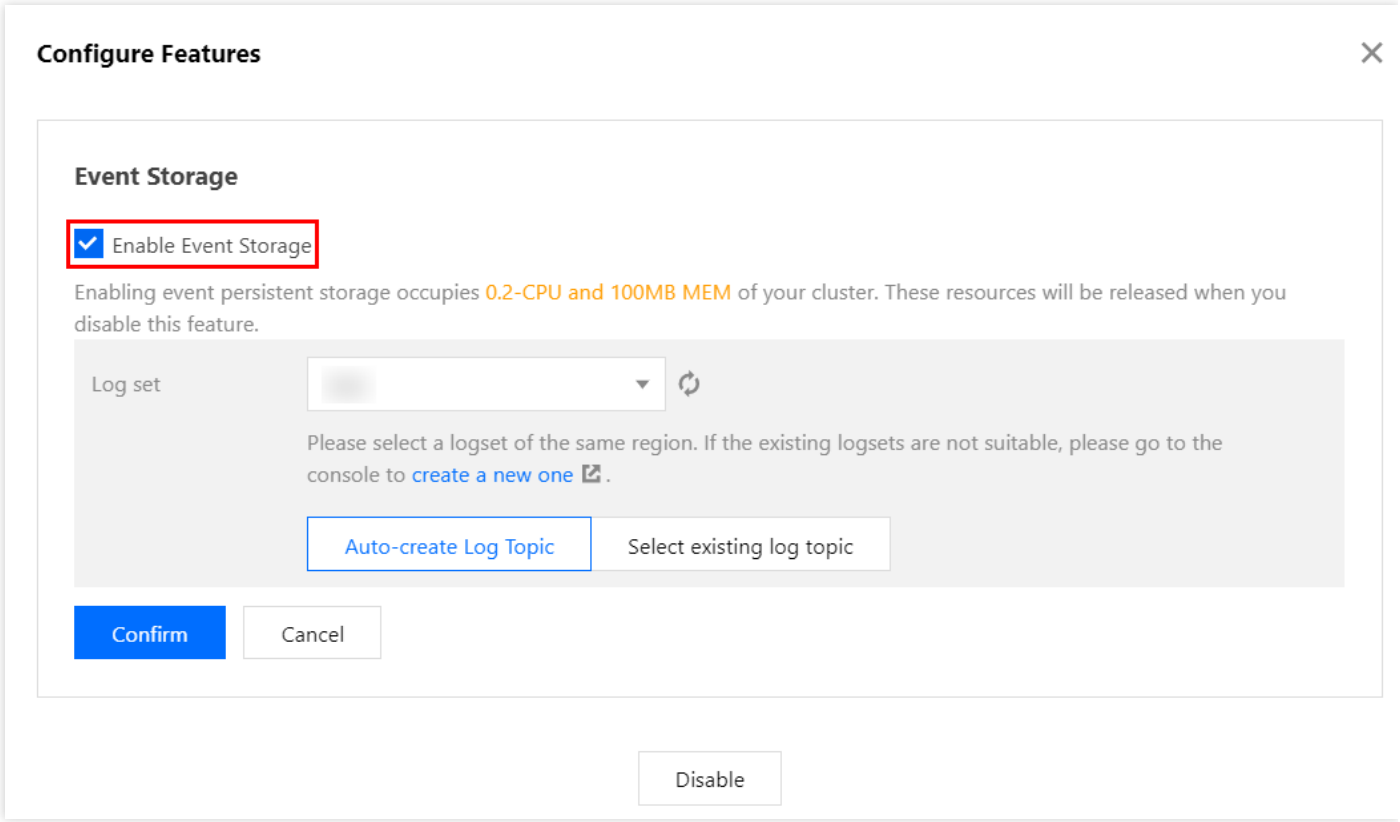
Enabling event storage

1. Log in to the TKE console.
2. Choose **OPS Feature Management** in the left sidebar to go to the **Feature Management** page.
3. At the top of the **Feature Management** page, select the region and select **TKE Serverless Cluster** as **Cluster Type**. On the right side of the cluster for which you want to enable event storage, click **Set**, as shown in the figure

below:



4. On the **Configure Features** page, click **Edit** for event storage.
5. On the **Event Storage** editing page, select **Enable Event Storage** and configure logsets and log topics, as shown in the figure below:

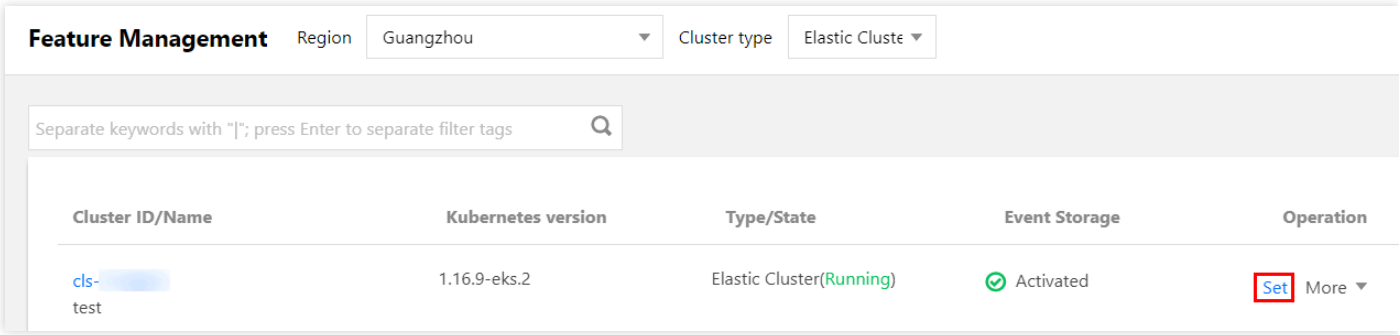


6. Click **Confirm** to enable event storage.

Updating logsets or log topics

1. Log in to the TKE console.
2. Choose **OPS Feature Management** in the left sidebar to go to the **Feature Management** page.
3. At the top of the **Feature Management** page, select the region and select **TKE Serverless Cluster** as **Cluster Type**. On the right side of the cluster for which you want to enable event storage, click **Set**, as shown in the figure

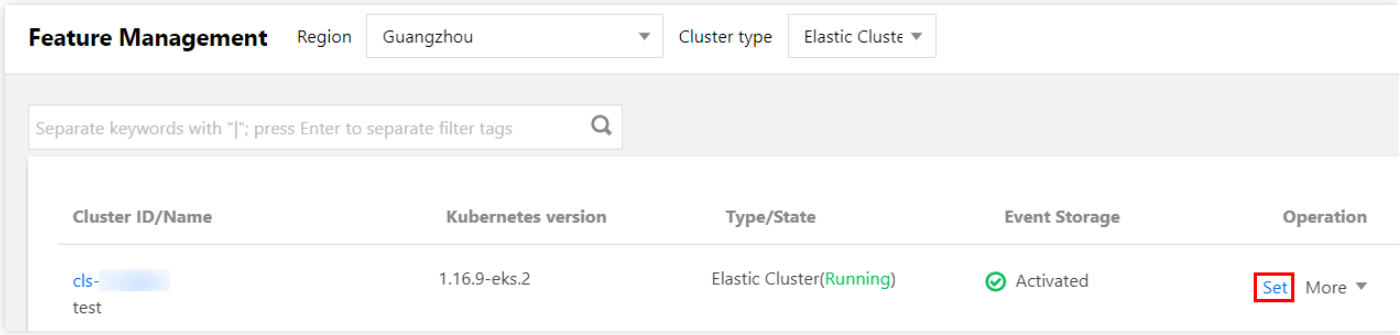
below:



4. On the **Configure Features** page, click **Edit** for event storage.
5. On the **Event Storage** editing page, reselect logsets and log topics. Then, click **OK** to update the logsets and log topics.

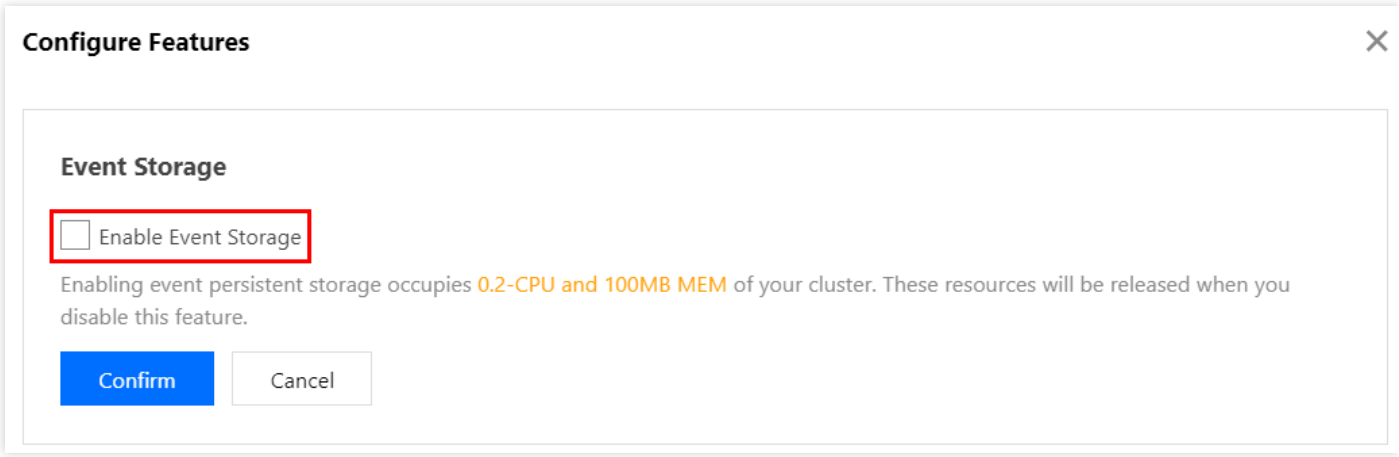
Disabling event storage

1. Log in to the TKE console.
2. Choose **OPS Feature Management** in the left sidebar to go to the **Feature Management** page.
3. At the top of the **Feature Management** page, select the region and select **TKE Serverless Cluster** as **Cluster Type**. On the right side of the cluster for which you want to enable event storage, click **Set**, as shown in the figure below:



4. On the **Configure Features** page, click **Edit** for event storage.

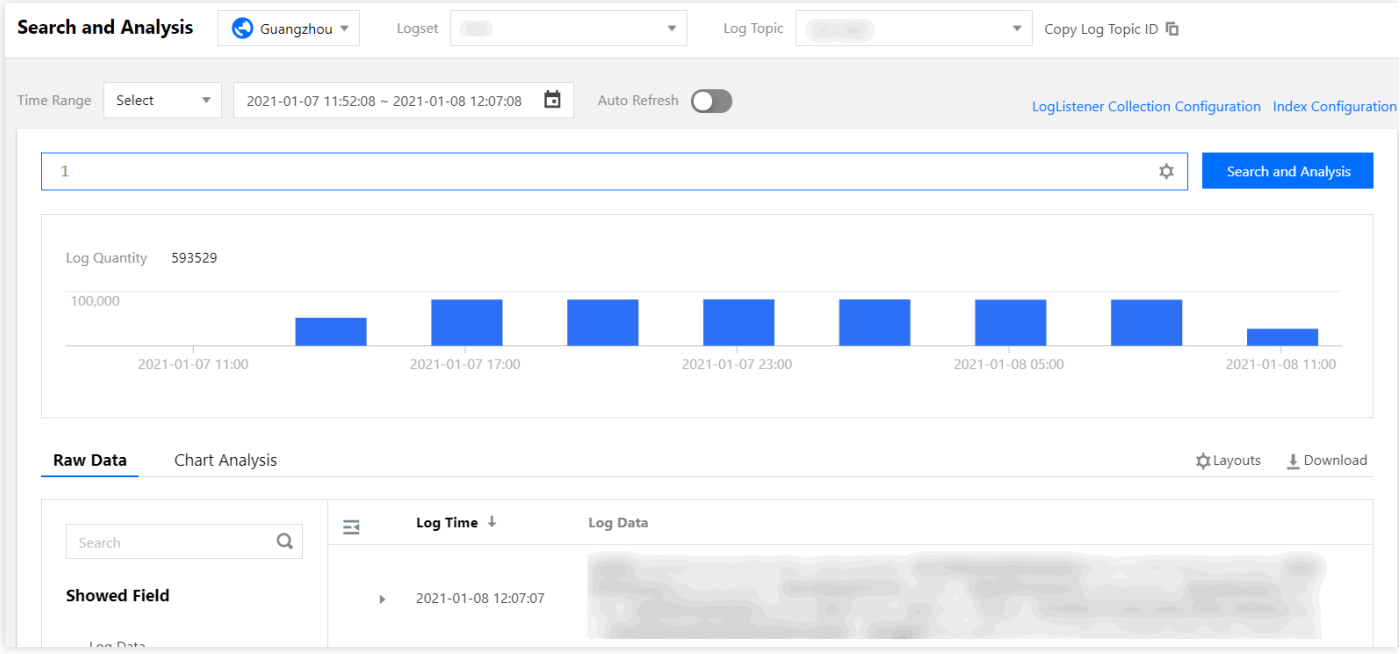
5. On the **Event Storage** editing page, deselect **Enable Event Storage**, as shown in the figure below:



6. Click **Confirm** to disable event storage.

Viewing event in the CLS console

1. Log in to the CLS console.
2. Select **Log Search** on the left side bar to go to the **Search and Analysis** page, as shown in the figure below:



3. Select the logset and log topic configured in Event Storage, set the **Showed Field** as needed, and click **Search and Analysis**. For more information, see Log Analysis.

> Note：
> When you enable event storage, the index will be enabled for your Topic by default.

Event Dashboard

Last updated：2022-09-13 16:38:20

Overview

TKE Serverless provides out-of-the-box event dashboards and can automatically configure the event overview dashboard and aggregated exception search dashboard for the cluster with event storage enabled. With user-defined filters and built-in CLS global event search, TKE Serverless makes it convenient for you to comprehensively observe, find, analyze, and locate problems in the TKE console.

Feature Description

Three dashboards are configured in **Event Search**, namely **Event Overview**, **Aggregated Exception Search**, and **Global Search**. Follow the steps below to enter the **Event Search** page and use the features:

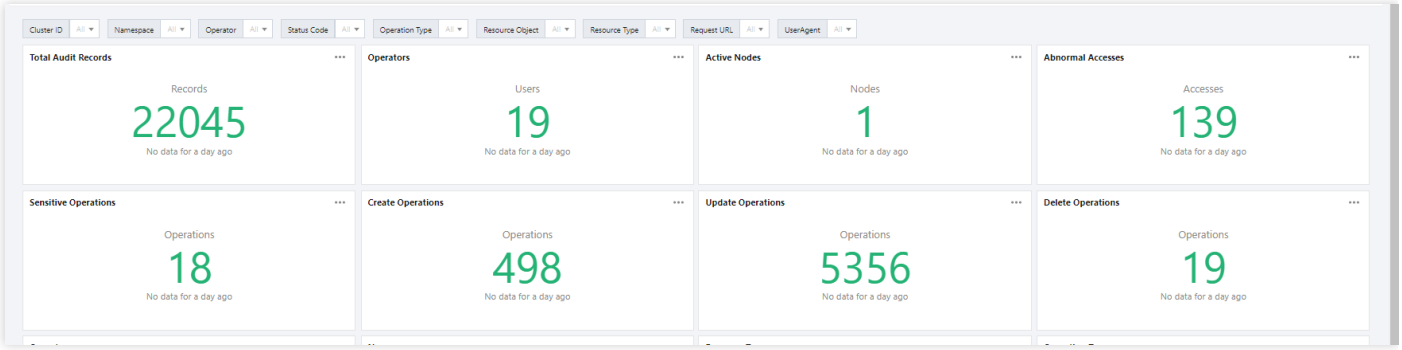
1. Log in to the TKE console.
2. Enable **Event Storage**. For more information, see Event Storage.
3. On the left sidebar, select **Log Management** > **Event Search**.

Event overview

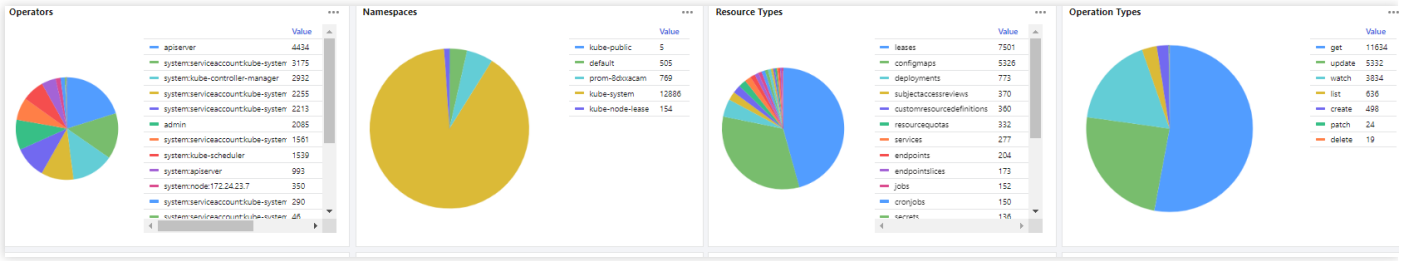
On the **Event Overview** tab, you can filter events based on dimensions such as time, namespace, level, reason, resource type, and resource object, view the aggregated statistics of the core event, and display data comparison within a period. Specifically, you can view dashboards of the total number of events and distribution, node exceptions, Pod OOM errors, and important event trends as well as the top exception list.

You can view more statistics on this page as shown below:

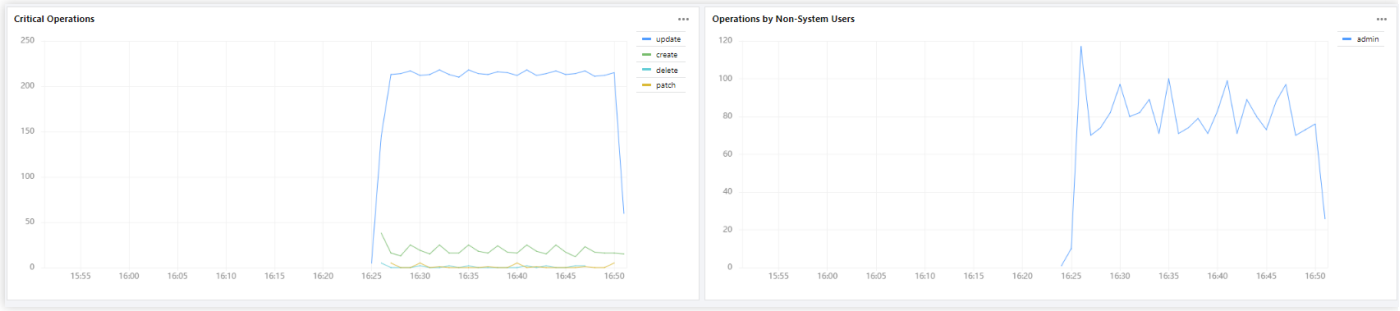
- The total number of events, level distribution, causes of exceptions, and object distribution are as shown below:



- The aggregated information of common events is as shown below:

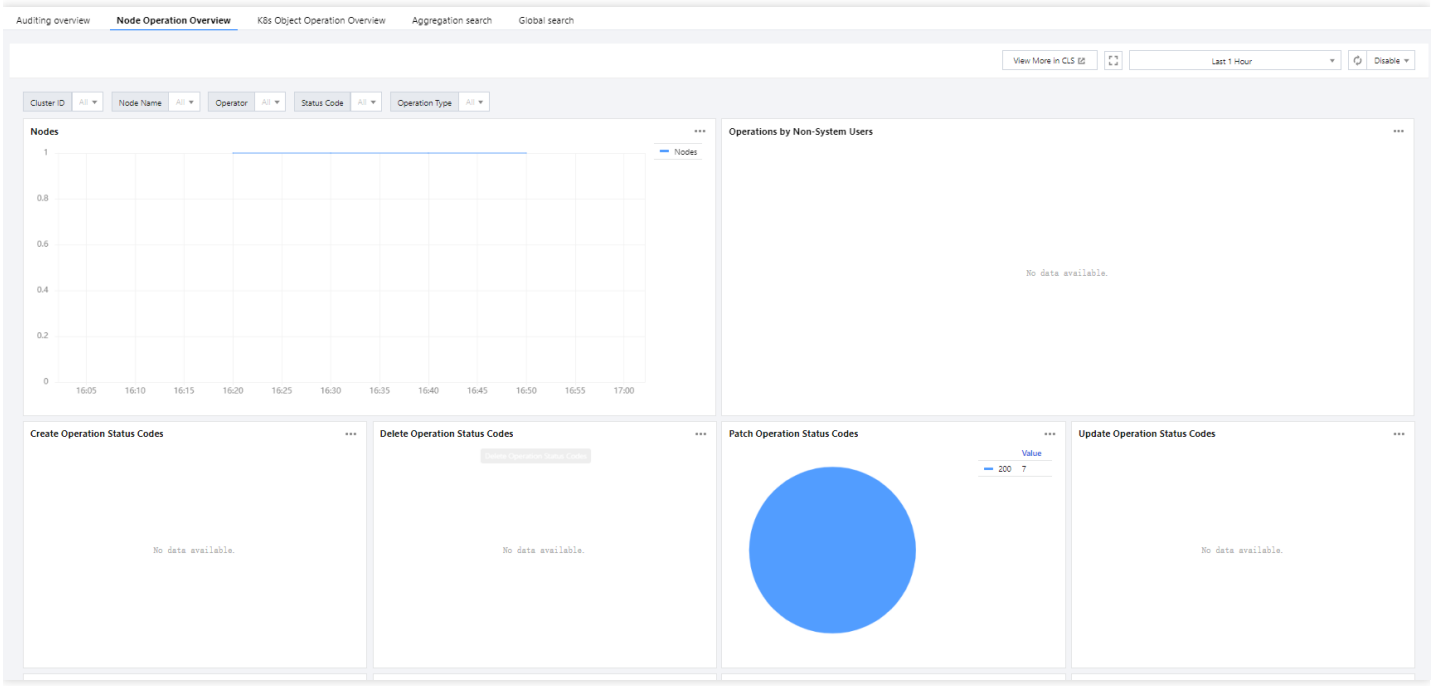


- Event trends and the top exception list are as shown below:



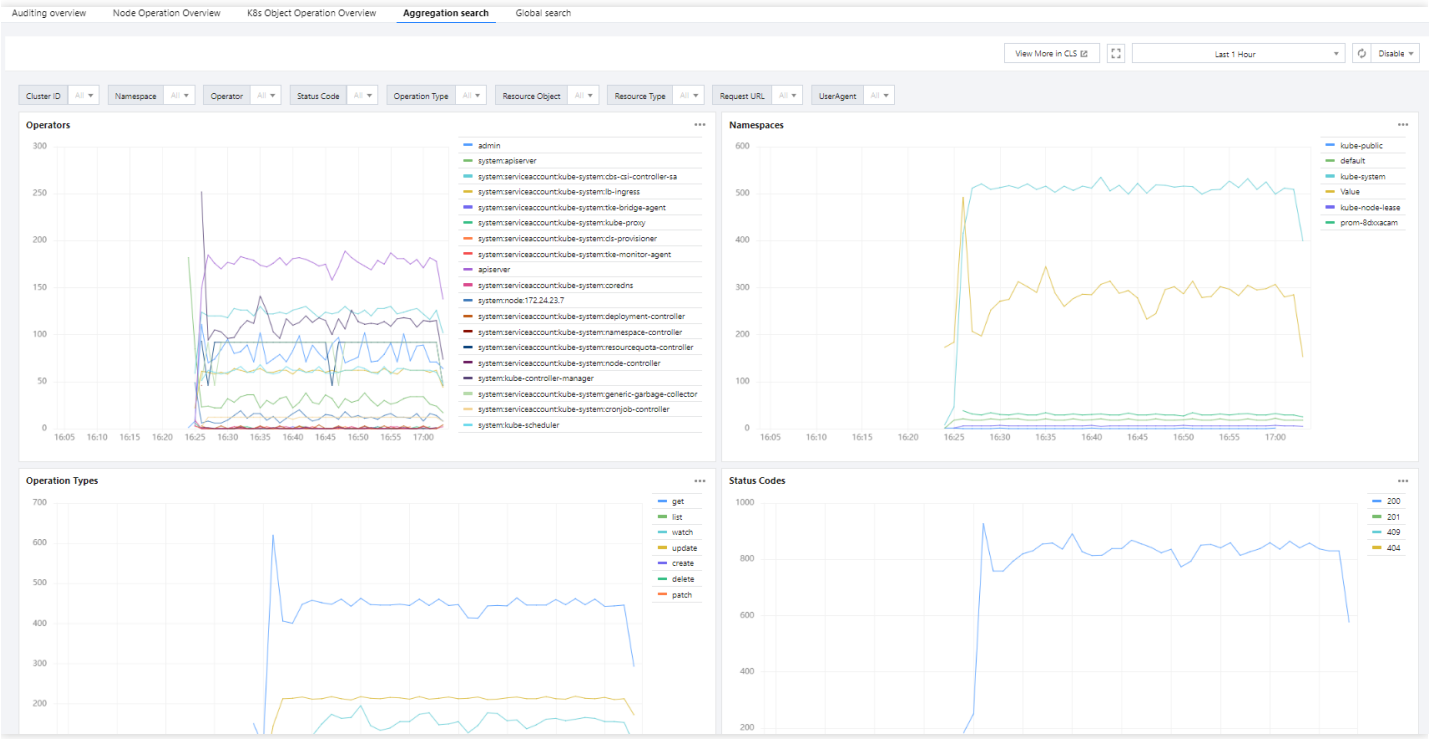
Aggregated exception search

On the **Aggregated Exception Search** tab, you can set filters to view the cause and object distribution trends of exceptions over a specified period of time. You can also search for exceptions in the list below the trend diagrams to quickly locate problems as shown below:



Global search

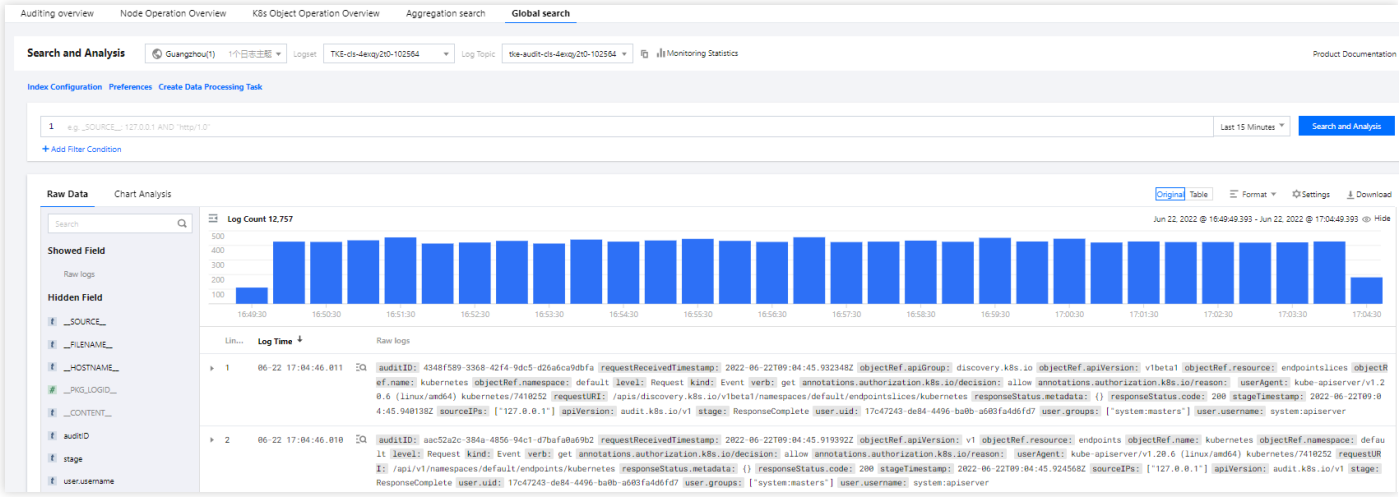
The global search dashboard with the built-in CLS search and analysis page makes it convenient for you to quickly search for all events in the TKE console as shown below:



Configuring alarm by dashboard

You can configure alarms based on the preset dashboards as instructed below, so that alarms will be triggered when the configured conditions are reached:

1. Click ••• on the right of the target dashboard and select **Quickly Add Alarm** from the drop-down list as shown below:



2. For more information on how to configure alarm policies, see Configuring Alarm Policies.

FAQs

Last updated：2023-05-22 17:02:12

TKE Serverless Cluster FAQs

Cloud Load Balancer FAQs

Super Node FAQs

How do I prohibit a Pod from being scheduled to a super node?
How do I prohibit a TKE general cluster from automatically scheduling a Pod to a super node in case of resource inadequacy?
How do I manually schedule a Pod to a super node?
How do I forcibly schedule a Pod to a super node, no matter whether the super node supports the Pod?
How do I customize DNS configuration for a super node?

Image Repository FAQs

How do I use the Tencent Container Registry (TCR) in a TKE Serverless cluster?
How do I use the external image repository that is created based on the self-signed certificate or HTTP protocol?

Log Collection FAQs

Why can't I view the logs in CLS console after configuring log collection for the cluster?
Where can I view the logs after configuring the log collection rules?
How do Java applications implement multi-line log merging?
How to adjust the log collection configuration to adapt to different log output rates?
What is the standard for outputting the logs of the application in containers of a TKE Serverless cluster?

Contact Us

Last updated：2022-04-25 15:37:10

Customer Service

If you have any questions about Tencent Cloud products, please contact our customer service for assistance.

- Hong Kong (China): +852 800-964-163 (toll-free)
- US: +1 888-652-2736 (toll-free)
- Other regions: +86 4009100100

Submitting a Ticket

If you encounter any OPS or technical problems when using our products, you can log in to the Tencent Cloud console and follow the on-screen prompts to submit a ticket. We will get back to you as soon as possible.
Ticket links:

- Submitting a ticket: Submit a ticket
- Querying ticket state: Ticket list

A ticket can have the following status:

- Pending processing: the ticket is just submitted or has been received but not reviewed by the technical support team. You can submit more information for or close the ticket at this stage.
- Processing: the technical support team has received and reviewed the ticket and is taking an action. You can submit more information for or close the ticket at this stage.
- More information required: the technical support team has received and reviewed the ticket, but more information is required for processing it. You can close the ticket at this stage.

> Note：
> The ticket will revert to "pending processing" status after you re-submit the ticket with more information.

- Closed: the ticket has been resolved, or you closed the ticket before it was processed.