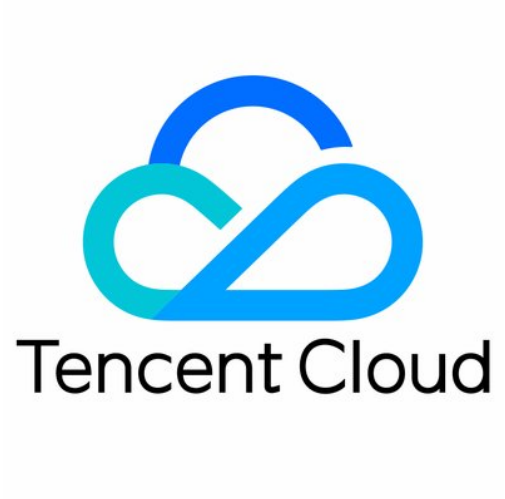
Tencent Kubernetes Engine

TKE Registered Cluster Guide

Product Documentation



Contents

TKE Registered Cluster Guide
Registered Cluster Management
Creating a Registered Cluster

Last updated：2024-05-09 15:46:41

The registered cluster is a new type of cluster in Tencent Kubernetes Engine (TKE), allowing users to register their Kubernetes clusters from local infrastructure or other cloud providers with TKE for unified management. This document will introduce how to register third-party Kubernetes clusters with TKE.

Prerequisites

The feature of registering clusters has been enabled. Currently, the capability of registering clusters is in a free beta phase. Please contact us to apply.
The supported version range of the registered Kubernetes clusters is from 1.18.x to 1.24.5. Versions outside this range have not been validated, and support for those versions is not guaranteed.

Directions

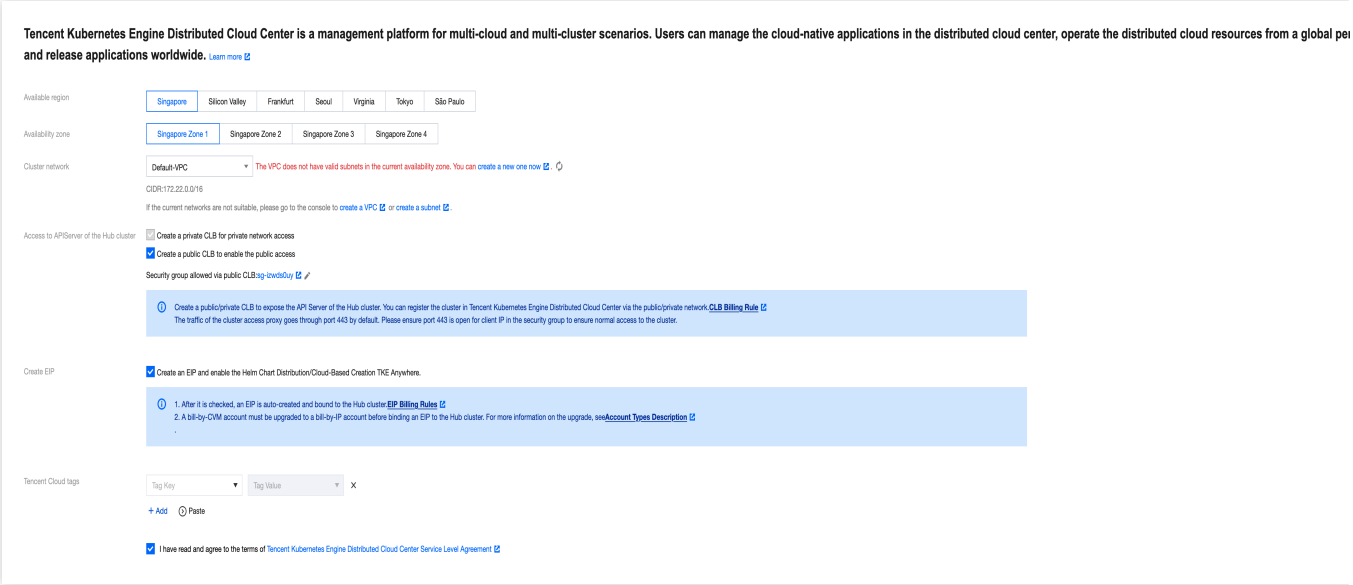
Creating a Hub Cluster

Note:
The registered cluster is an important part of the resource management capabilities of the Tencent Kubernetes Engine Distributed Cloud Center (TDCC), implemented based on the open-source Clusternet multi-cluster application governance project.
Before performing the cluster registration operation, it is necessary to first create a hub cluster, which can then be used to manage other registered child clusters.
1. In the Tencent Cloud console, choose **Cloud Products** > **TDCC** to enter the TDCC console, and follow the on-screen prompts to activate the TDCC service and authorize it. (If you have already authorized the service, skip this step.)
2. Follow the prompts on the page to set the basic information of the hub cluster:
Available region: Select a region for the hub cluster. Currently, only Guangzhou, Beijing, and Singapore are supported, but more regions will be supported in the future.
Availability zone: Select an availability zone for the hub cluster.
Cluster network: Select a subnet. Access to the hub cluster's kube-apiserver requires the use of an Elastic Network Interface (ENI), so you need to provide a VPC subnet. TKE will automatically create a proxy ENI within the selected

subnet.

Note:

Once the hub cluster is created, the access region and availability zone cannot be changed.



Creating a Registered Cluster

1. Log in to the TKE console, and choose **Register an existing cluster** from the left navigation bar.

2. On the registered cluster management page, click **Register an existing cluster** above the cluster list.

3. Set the basic information of the registered cluster:

Cluster name: The name of the registered cluster, up to 60 characters.

Access region: Select an access region for the registered cluster. Currently, only Guangzhou, Beijing, and Singapore are supported, but more regions will be supported in the future.

Note:

The access region is unrelated to the actual operating region of the cluster to be registered. It refers to the region where the hub cluster that manages this registered cluster is located.

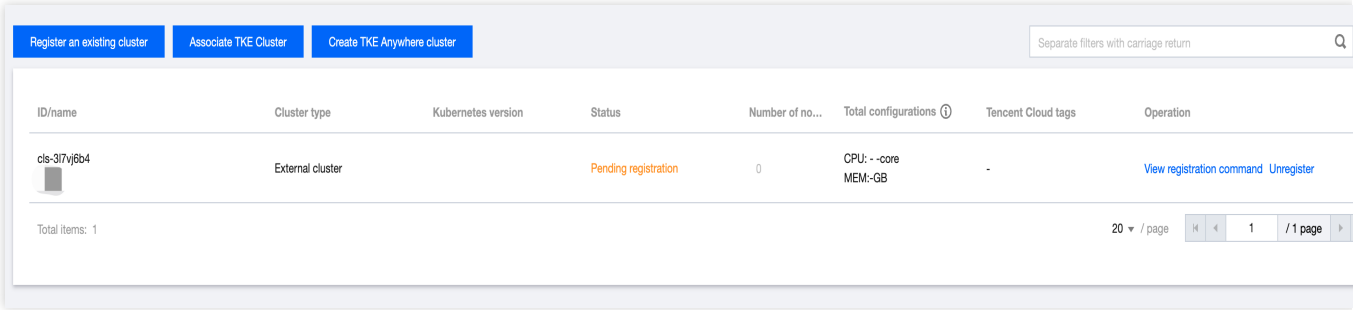
Tencent Cloud tags: Binding tags to a cluster enables classified management of resources. For details, see Querying Resources by Tag.

Cluster description: Fill in the relevant information of the cluster. This information will be displayed on the **Cluster Info** page.

4. Click **Done** to create a registered cluster. You can see the cluster you created in the registered cluster list, with the state **Pending registration**, as shown below:

Note:

You can **View registration command** or **Unregister** a cluster waiting for registration in the cluster list.



Executing the Registration Command

1. On the cluster management page, locate the created registered cluster and select **View registration command** on the right side of the registered cluster to see the corresponding registration command.

2. Users can choose to register the cluster via **public network** or **private network**, copy or download the registration command, and execute the kubectl command in a third-party cluster to complete the registration.

Note:

The validity period of the registration command is 24 hours. Please complete the registration within this period. If it exceeds the validity period, you will need to regenerate the registration command on the page.

3. Execute the following command to check the agent running status, as shown below:

```
# kubectl get pod -n clusternet-system
NAME                                  READY   STATUS    RESTARTS   AGE
clusternet-agent-78444974d7-f6fsc     1/1     Running   0          7m32s
clusternet-agent-78444974d7-qjp2q     1/1     Running   0          7m32s
clusternet-agent-78444974d7-r575w     1/1     Running   0          7m32s
```

After successful registration, the status of the registered cluster changes to **Running**, indicating that the cluster has been successfully registered.

Connecting to a Registered Cluster

Last updated：2024-05-09 15:47:11

Operation Scenarios

This document describes how to connect a local client to a registered cluster using kubectl, the Kubernetes command-line tool.

Prerequisites

The cURL software has been installed.

Select an appropriate way to obtain kubectl based on the type of the operating system:

Note:

Replace the version `v1.8.13` in the command line with the kubectl version required for your business, according to the version you are actually using.

macOS

Linux

Windows

Execute the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-
release/release/v1.8.13/bin/darwin/amd64/kubectl
```

Execute the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-
release/release/v1.8.13/bin/linux/amd64/kubectl
```

Execute the following command to obtain kubectl:

```
curl -LO https://storage.googleapis.com/kubernetes-
release/release/v1.8.13/bin/windows/amd64/kubectl.exe
```

Directions

Installing kubectl

1. Refer to Install and Set Up kubectl to install kubectl.

Note:

If you have already installed kubectl, skip this step.

This step uses the Linux operating system as an example.

2. Execute the following commands in sequence to grant execution permissions.

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

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

3. Execute the following command to check the installation result.

```
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", GitCommit:"
```

Configuring kubeconfig

1. Log in to the TKE console, and choose Cluster from the left navigation bar.
2. On the cluster list page, click the ID of the registered cluster you want to connect to, to enter the management page of the cluster.
3. Choose **Basic Information** from the left navigation bar to enter the basic information page of the cluster.
4. In the **Cluster APIServer Information**, obtain the kubeconfig for either **public network access** or **private network access**, which you can copy or download.
5. Configure the cluster credential as needed. For details, see **Connecting to the Kubernetes Cluster Through kubectl** in the console.

Accessing the Kubernetes Cluster

1. After completing the kubeconfig configuration, execute the following commands in sequence to view contexts and switch the contexts to access the cluster.

```
kubectl config get-contexts
```

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

2. Execute 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.

Relevant Descriptions

Introduction to the kubectl CLI

kubectl is a command-line tool for operating Kubernetes clusters. This document covers syntax and common command operations of kubectl, and provides common examples. For detailed information about each command (including all main commands and subcommands), refer to the kubectl Reference Documentation or use the `kubectl help` command for more detailed assistance. For kubectl installation instructions, refer to the earlier section Installing kubectl.

Unregistering a Cluster

Last updated：2024-05-09 15:47:27

Operation Scenarios

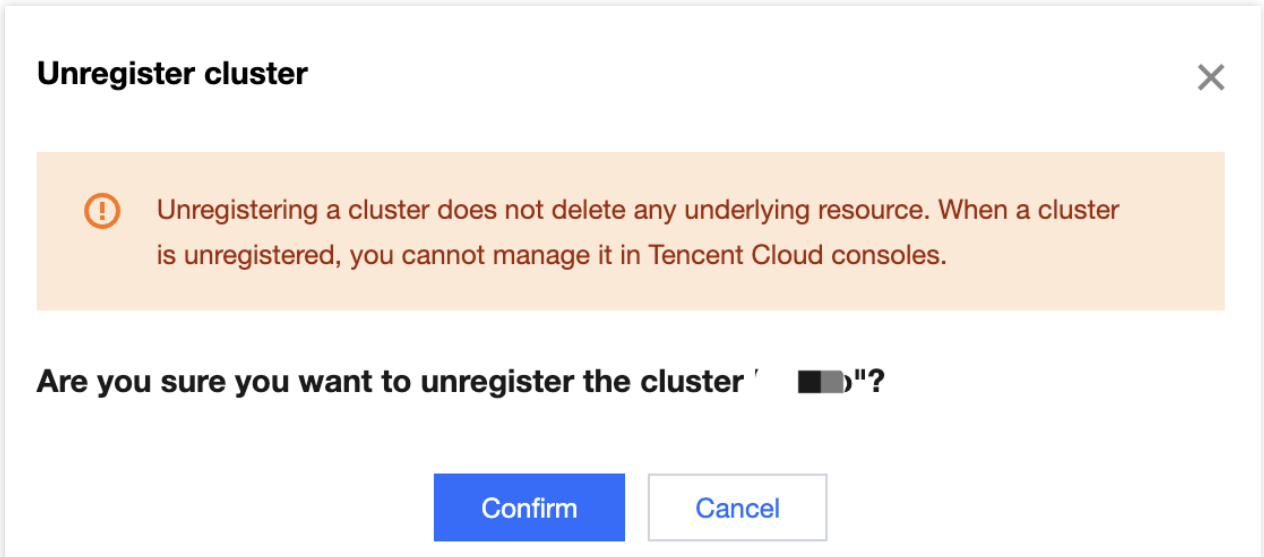
You can unregister an already registered cluster. Once a cluster is unregistered, you will not be able to manage it through the TKE console.

Directions

1. Log in to the TKE console.
2. In the cluster list, click **Unregister** on the right side of the row containing the cluster you need to unbind.
3. In the **Unregister cluster** window, click **Confirm**.

Note:

After a cluster is unregistered, the proxy software installed in the cluster will be deleted, but the cluster itself and other resources within the cluster will not be affected.



Ops Guide
Log Collection

Last updated：2023-06-08 15:28:12

This document describes how to ship logs of a registered cluster to CLS in the console.

Scenario

TKE's log collection feature allows you to collect logs in a cluster and send logs in specific paths of cluster services or nodes to Tencent Cloud Log Service (CLS). Log collection applies to users who need to store and analyze service logs in Kubernetes clusters.
You need to manually enable log collection for each cluster, and configure the collection rules. After log collection is enabled for a cluster, the log collection agent runs as a DaemonSet in the cluster, collects logs from the collection source based on the collection source, CLS log topic, and log parsing method configured by users in the log collection rules, and sends the collected logs to the consumer.

Notes

You have created a registered cluster, and it is in **Running** status.
Currently, logs of a registered cluster can be shipped to only CLS but not other log consumers.
Before enabling log collection, ensure that there are sufficient resources on cluster nodes.
0.11 to 1.1 cores are required. You can increase the CPU resources on your own as needed.
24 to 560 MB memory is required. You can increase the memory resources on your own as needed.
The maximum size of a log is 512 K. The log is truncated if this limit is exceeded.
To use the log collection feature, check whether nodes in the Kubernetes cluster can access the log consumer. Here, TKE ships logs over the public and private networks. You can select one option based on your business needs.
Shipping over public network: Cluster logs will be shipped to CLS over the public network. This requires that the cluster nodes can access the public network.
Shipping over private network: Cluster logs will be shipped to CLS over the private network. This requires that the cluster nodes are interconnected with CLS over the private network. Before choosing this option, contact us for confirmation.

Concept

Log Collection Agent: The agent that TKE uses to collect logs. It adopts Loglistener and runs within the cluster as a DaemonSet.
Log Rules: Configures rules to specify the log collection source, log topic, and log parsing method and configure the filter.
The log collection agent monitors changes in the log collection rules, and rule changes take effect within 10 seconds. Multiple log collection rules do not create multiple DaemonSets, but too many log collection rules cause the log collection agent to occupy more resources.
Log Source: It includes the specified container standard output, files in containers, and node files.
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.
When collecting node file path logs, users can set the node file path as the log collection source.
Consumer: It can be a logset or a log topic.
Extraction mode: The log collection agent can ship the collected logs to the specified log topic in the format of single-line text, JSON, separator-based text, multi-line text, or full regex.
Filter: Sets filters to collect only logs match the rules. "key" supports full matching and the rule supports regex matching. For example, you can set to collect logs containing "ErrorCode = 404".

Directions

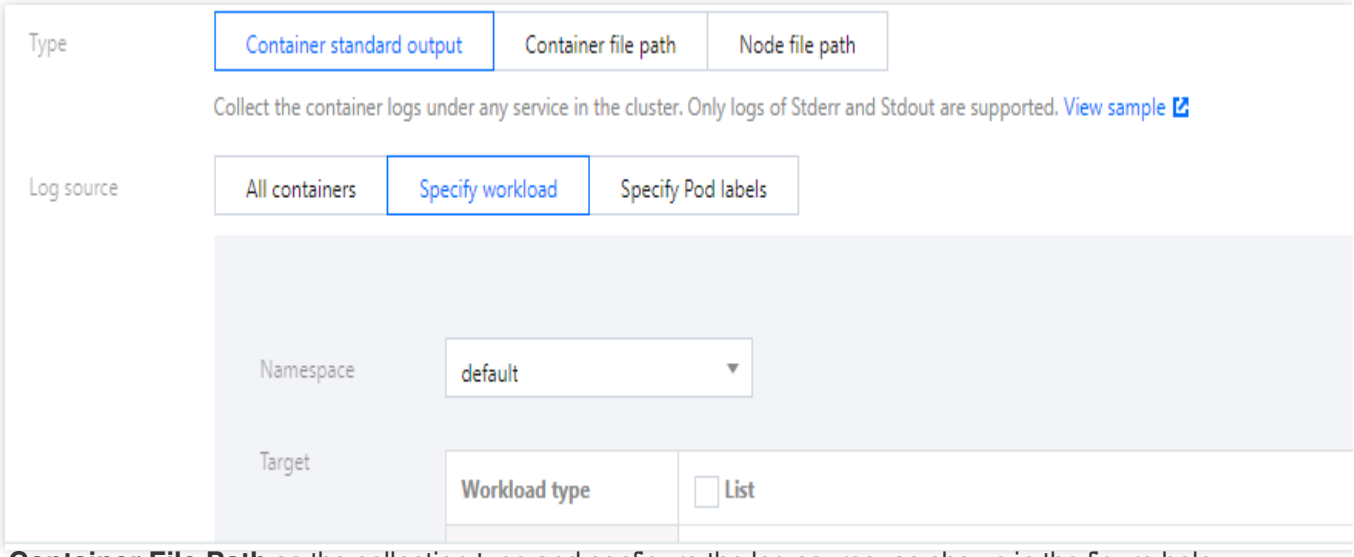
Enabling log collection

1. Log in to the TKE console and select **Operation Management** in the left sidebar.
2. At the top of the **Feature Management** page, select a desired region and **Registered cluster** to filter out the cluster for which you want to enable log collection, and click **Set** on the right.
3. On the **Configure Features** page, click **Edit** for log collection, select **Enable Log Collection**, select the **Shipping Method**, and click **Confirm**.

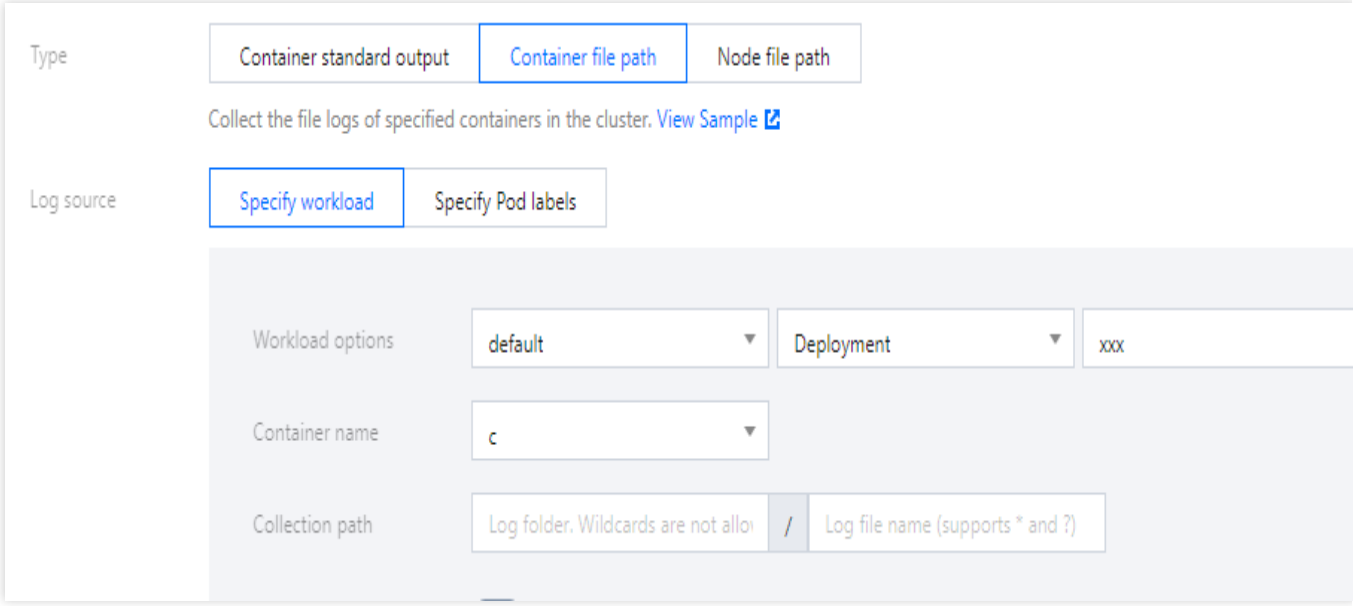
Configuring the log rules

1. Log in to the TKE console and select **Log Management** > **Log Rules** in the left sidebar.
2. At the top of the **Feature Management** page, select a desired region and **Registered cluster** to filter out the cluster for which you want to configure the log collection rules, and click **Create**.
3. On the **Create Log Collecting Policy** page, select the collection type and configure the log source. Currently, the following collection types are supported: **Container Standard Output**, **Container File Path**, and **Node File Path**.
Collecting standard output logs of a container
Collecting file logs in containers
Collecting file logs on nodes

Select **Container Standard Output** as the collection type and configure the log source as needed. This type of log source allows you to select the workloads of multiple namespaces at a time, as shown in the figure below:



Select **Container File Path** as the collection type and configure the log source, as shown in the figure below:



You can specify a file path or use wildcards for the collection path. For example, when the container file path is `/opt/logs/*.log`, you can specify the collection path as `/opt/logs` and the file name as `*.log`.

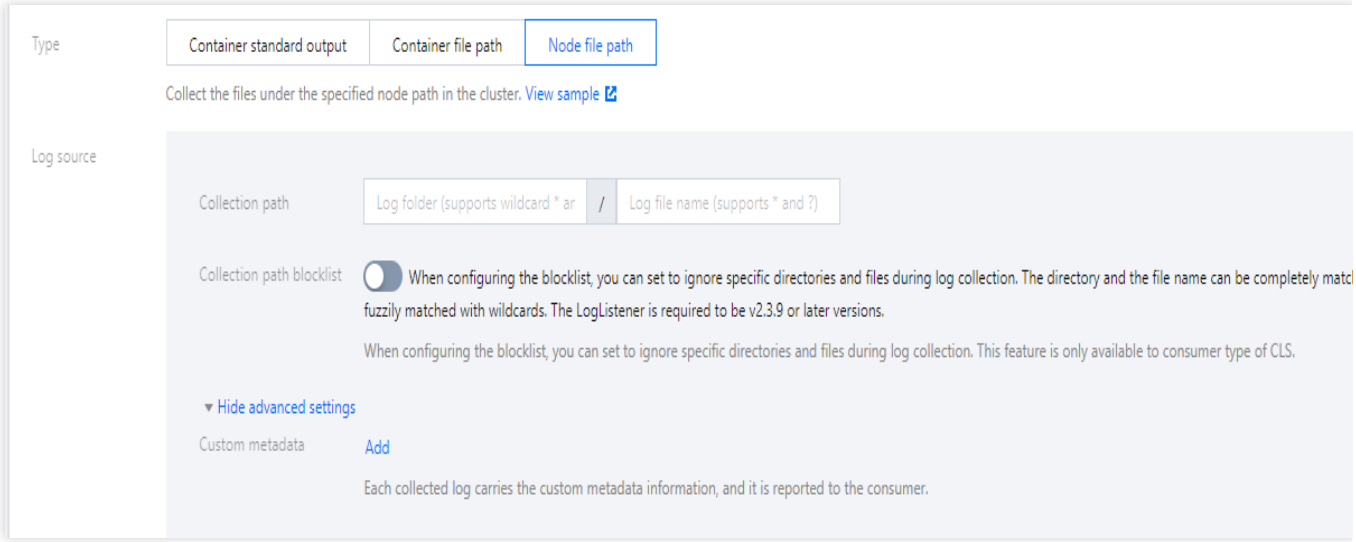
Note:

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.

Select **Node File Path** as the collection type. You can add custom `metadata` as needed. Attach `metadata` with a specified key-value pair to the collected log information to add the attached metadata to log records, as shown in the figure below:

Note

Each node log file can be collected to only one log topic.



You can specify a file path or use wildcards. For example, when the container file paths for collection are `/opt/logs/service1/*.log` and `/opt/logs/service2/*.log`, you can specify the folder of the collection path as `/opt/logs/service*` and the file name as `*.log`.

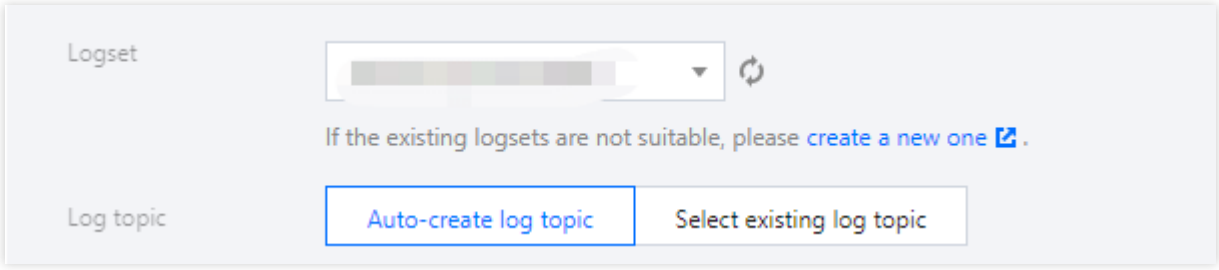
Note:

For container standard output and container files (not mounted in hostPath), besides the original log content, the metadata related to the container or Kubernetes (such as the ID of the container that generated the logs) will also be reported to the CLS. Therefore, when viewing logs, users can trace the log source or search based on the container identifier or characteristics (such as container name and labels).

The metadata related to the container or Kubernetes is shown in the table below:

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

4. Configure CLS as the consumer end. Select the desired logset and log topic. You can create a log topic or select an existing one.

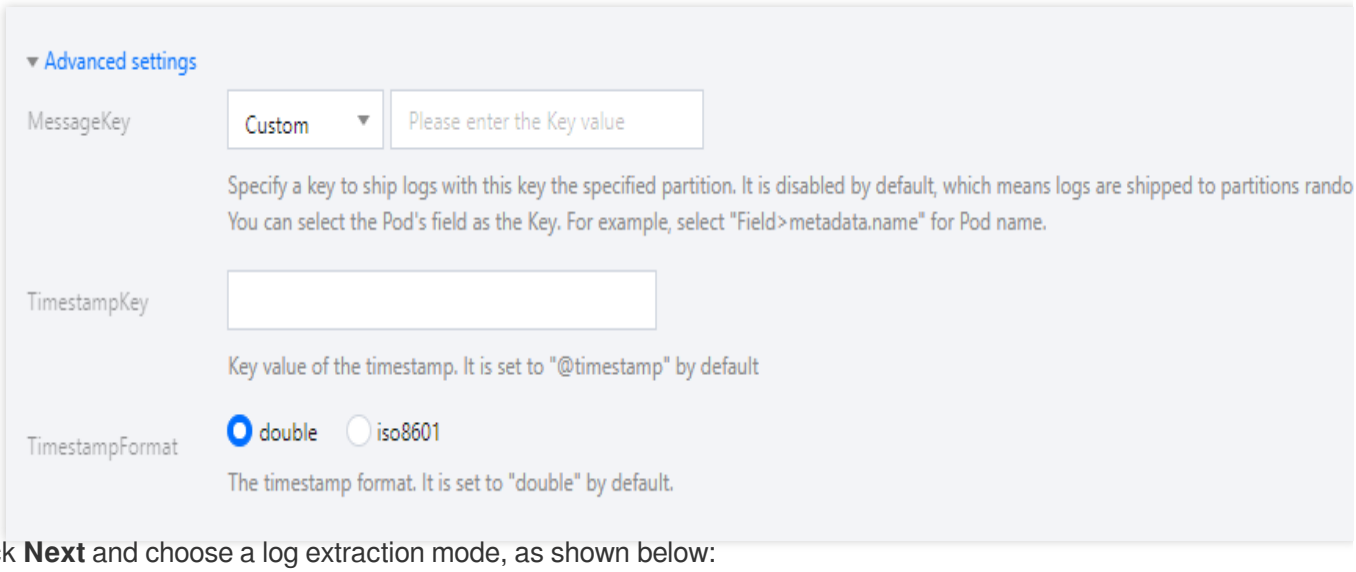


Note

Currently, CLS only supports log collection and reporting for TKE clusters in the same region.

If there are already 500 log topics in the logset, no more log topic can be created.

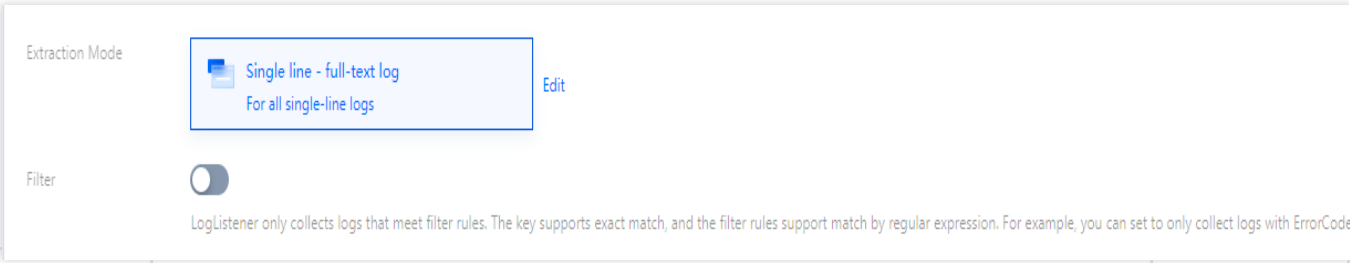
5. You can ship the logs to a specified partition by specifying a key in advanced settings. This feature is disabled by default and the logs are shipped randomly. When it is enabled, logs with the same key are shipped to the same partition. You can enter the TimestampKey (@timestamp by default) and specify the timestamp format. See the figure below:



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

Note

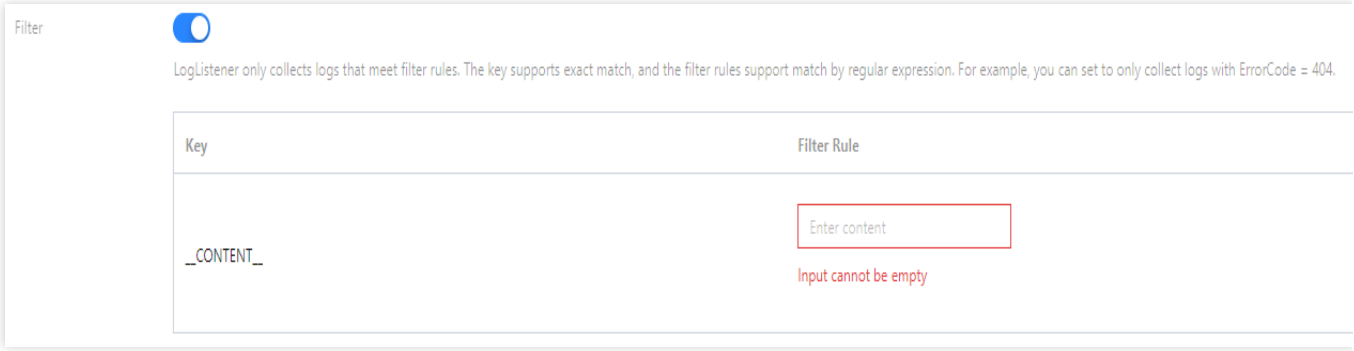
Configuring log parsing method is only supported when you select shipping logs to CLS.



Parsing mode	Description	Reference
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 Expression (Single-Line)
Multiple lines - full regex	The multi-line - full regular expression mode is a log parsing mode where multiple key-value pairs can be extracted from a complete piece of log data that spans multiple lines in a log text file (such as Java program logs) based on a regular expression. When configuring the multi-line - full regular expression mode, you need to enter a sample log first and then customize your regular expression. After the configuration is completed, the system will extract the corresponding key-value pairs according to the capture group in the regular expression. The regular expression can be generated automatically.	Full Regular Expression (Multi-Line)
JSON	A JSON log automatically extracts the key at the first layer as the field name and the value at the first layer as the field value to implement structured processing of the entire log. Each complete log ends with a line break `\\n` .	JSON Format
Separator	Structure the data in a log with the specified separator, and each complete log ends with a line break `\\n` . Define a unique key for each separate field.	Separator Format

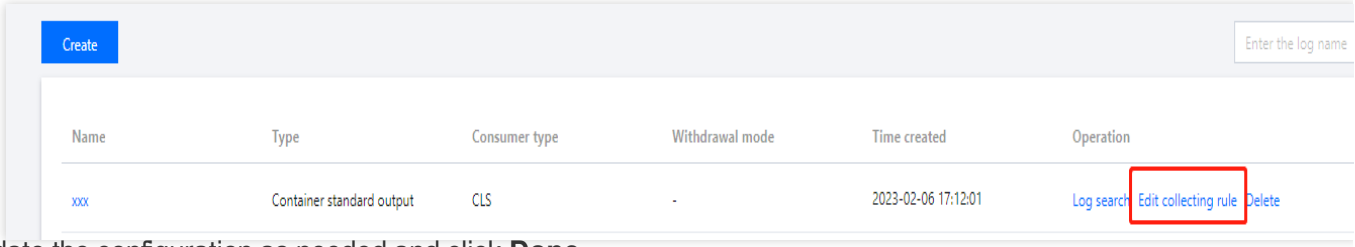
	Leave the field blank if you don't need to collect it. At least one field is required.	

7. Enable the filter and configure rules as needed and then click **Done**.



Updating the log rules

1. Log in to the TKE console and select **Log Management** > **Log Rules** in the left sidebar.
2. At the top of the **Log Collection** page, select a desired region and **Registered cluster** to filter out the cluster for which you want to configure the log collection rules, and click **Edit Collecting Rule**.



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

Note

The logset and log topic cannot be modified later.

References

Using CRD to Configure Log Collection

Cluster Auditing

Last updated：2024-12-23 15:06:31

This document describes how to ship audit logs of a registered cluster to CLS.

Overview

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

Notes

You have created a registered cluster, and it is in **Running** status.
Currently, audit logs of a registered cluster can be shipped to only CLS but not other log consumers.
To enable the auditing feature of a registered cluster, you need to log in to all master nodes of the cluster to configure relevant audit policies and API server parameters.
If the cluster auditing feature is enabled, cluster log collection will also be enabled automatically at the same time by default.
To use the cluster auditing feature, check whether nodes in the Kubernetes cluster can access the log consumer. Here, logs can be shipped over the public and private networks. You can select one option based on your business needs.
Shipping over public network: Cluster auditing logs will be shipped to CLS over the public network. This requires that the cluster nodes can access the public network.
Shipping over private network: Cluster auditing logs will be shipped to CLS over the private network. This requires that the cluster nodes are interconnected with CLS over the private network. Before choosing this option, submit a ticket for confirmation.

Directions

Configuring audit policies on master nodes in the cluster

Log in to all master nodes in the cluster one by one and configure the audit policy file `/etc/kubernetes/audit-policy.yaml` based on your actual business conditions.

```
apiVersion: audit.k8s.io/v1beta1
```

```
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
      - group: ""
        resources: ["endpoints", "services"]
  - level: None
    users: ["system:unsecured"]
    namespaces: ["kube-system"]
    verbs: ["get"]
    resources:
      - group: ""
        resources: ["configmaps"]
  - level: None
    users: ["kubelet"]
    verbs: ["get"]
    resources:
      - group: ""
        resources: ["nodes"]
  - level: None
    userGroups: ["system:nodes"]
    verbs: ["get"]
    resources:
      - group: ""
        resources: ["nodes"]
  - level: None
    users:
      - system:kube-controller-manager
      - system:kube-scheduler
      - system:serviceaccount:kube-system:endpoint-controller
    verbs: ["get", "update"]
    namespaces: ["kube-system"]
    resources:
      - group: ""
        resources: ["endpoints"]
  - level: None
    users: ["system:apiserver"]
    verbs: ["get"]
    resources:
      - group: ""
        resources: ["namespaces"]
  - level: None
    nonResourceURLs:
```

```
      - /healthz*
      - /version
      - /swagger*
  - level: None
    resources:
      - group: ""
        resources: ["events"]
  - level: Metadata
    resources:
      - group: "" # core
        resources: ["secrets", "configmaps"]
      - group: authentication.k8s.io
        resources: ["tokenreviews"]
  - level: Request
    verbs: ["get", "list", "watch"]
    resources:
      - group: ""
      - group: "admissionregistration.k8s.io"
      - group: "apps"
      - group: "authentication.k8s.io"
      - group: "authorization.k8s.io"
      - group: "autoscaling"
      - group: "batch"
      - group: "certificates.k8s.io"
      - group: "extensions"
      - group: "networking.k8s.io"
      - group: "policy"
      - group: "rbac.authorization.k8s.io"
      - group: "settings.k8s.io"
      - group: "storage.k8s.io"
  - level: RequestResponse
    resources:
      - group: ""
      - group: "admissionregistration.k8s.io"
      - group: "apps"
      - group: "authentication.k8s.io"
      - group: "authorization.k8s.io"
      - group: "autoscaling"
      - group: "batch"
      - group: "certificates.k8s.io"
      - group: "extensions"
      - group: "networking.k8s.io"
      - group: "policy"
      - group: "rbac.authorization.k8s.io"
      - group: "settings.k8s.io"
      - group: "storage.k8s.io"
  - level: Metadata
```

Configuring API server parameters on master nodes

Log in to all master nodes in the cluster one by one and modify the `/etc/kubernetes/manifests/kube-apiserver.yaml` file.

1. Add the following command parameters:

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --audit-log-maxbackup=10
    - --audit-log-maxsize=100
    - --audit-log-path=/var/log/kubernetes/kubernetes.audit
    - --audit-log-maxage=30
    - --audit-policy-file=/etc/kubernetes/audit-policy.yaml
```

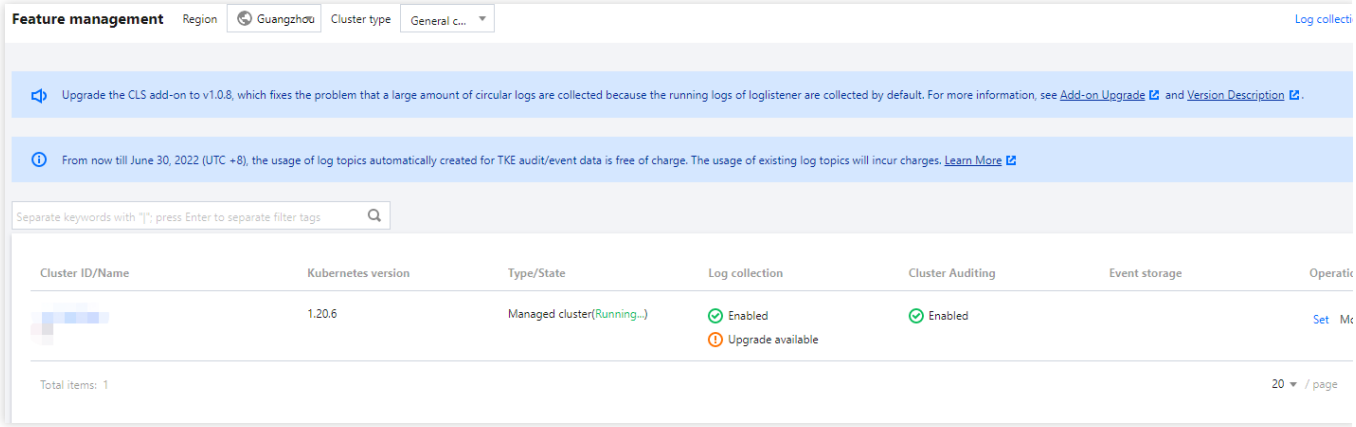
2. Add volume parameters to mount `/etc/kubernetes/audit-policy.yaml` to the API server Pod.

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --audit-log-maxbackup=10
    - --audit-log-maxsize=100
    - --audit-log-path=/var/log/kubernetes/kubernetes.audit
    - --audit-log-maxage=30
    - --audit-policy-file=/etc/kubernetes/audit-policy.yaml
    ...
    ...
    volumeMounts:
    - mountPath: /var/log/kubernetes
      name: k8s-audit
    - mountPath: /etc/kubernetes/audit-policy.yaml
      name: audit-policy
      readOnly: true
    ...
    ...
  volumes:
  - hostPath:
      path: /var/log/kubernetes
      type: DirectoryOrCreate
    name: k8s-audit
  - hostPath:
      path: /etc/kubernetes/audit-policy.yaml
      type: FileOrCreate
    name: audit-policy
```

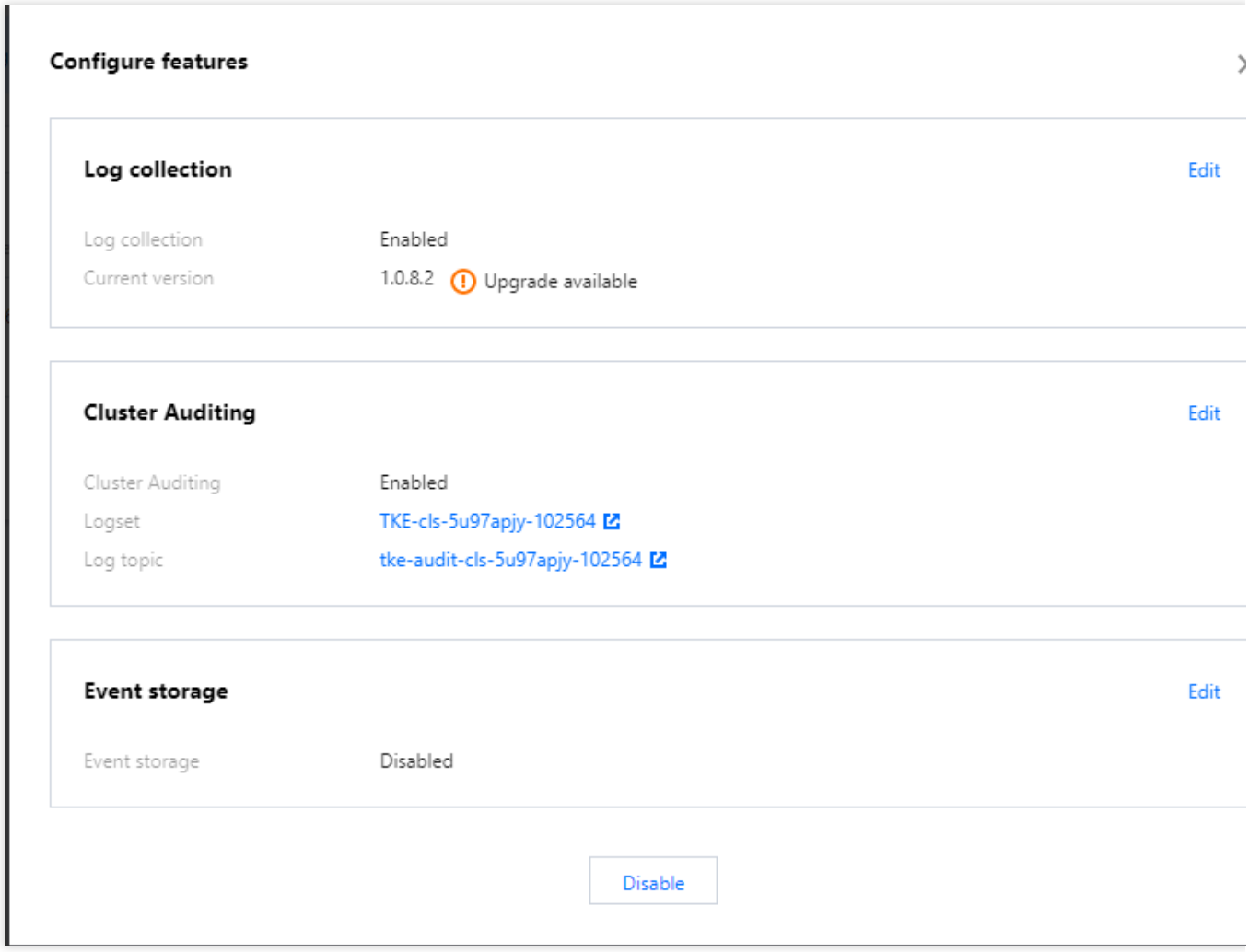
```
...
```

Enabling cluster auditing

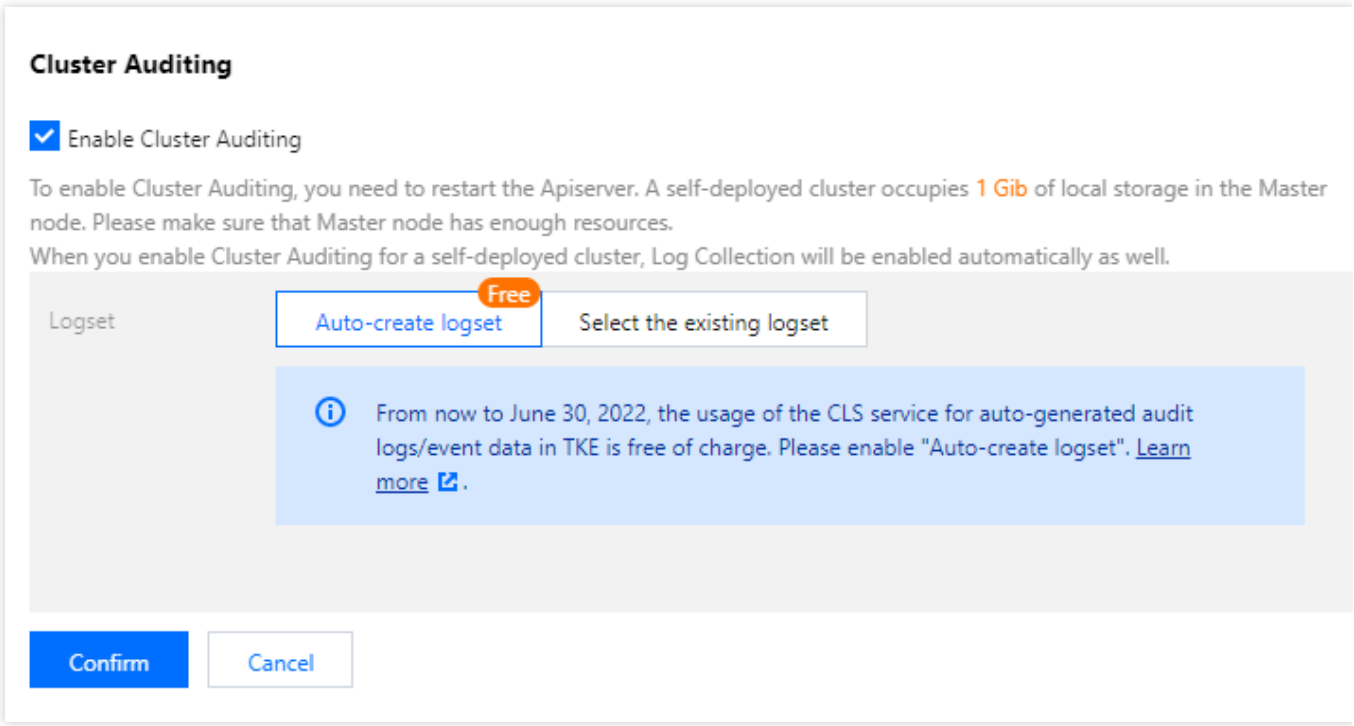
1. Log in to the TKE console and select **Ops Feature Management** on the left sidebar.
2. At the top of the **Feature Management** page, select the **Region** and **Registered Cluster**. Then, click **Set** on the right of the target cluster.



3. In the **Configure features** pop-up window, click **Edit** on the right of the **Cluster Auditing** feature.



4. Select **Enable Cluster Auditing** and select the shipping method and the logset and log topic for audit log storage. We recommend you select **Auto-create Log Topic**.



5. Click **Confirm**.

Audit Dashboard

TKE provides out-of-the-box audit dashboards and can automatically configure dashboards of audit overview, node operation 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 makes it convenient for you to observe and search for cluster operations, so as to promptly find and locate problems. For more information, see Auditing Dashboard.