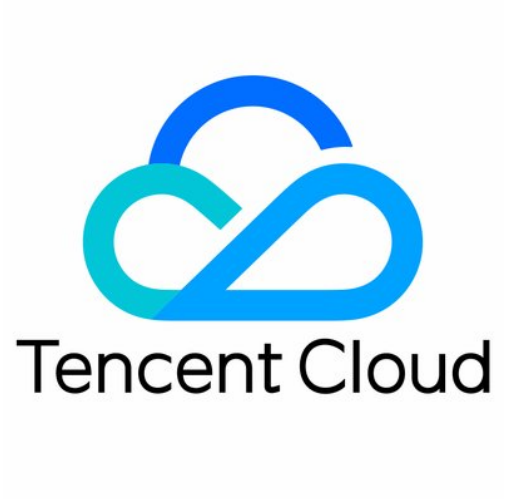
Tencent Kubernetes Engine

Cloud Native Service Guide

Product Documentation



Contents

Cloud Native Service Guide
Cloud Native AI Guide
Ops Console Guide
Managing AI Environment

Last updated：2022-10-12 11:56:32

This document describes AI environments and how to manage them, such as how to create, view, and delete them.

AI Environment Overview

AI environment is an important abstract concept of Cloud Native AI. An AI environment runs in a TKE/EKS container cluster, and the Ops team can manage its lifecycle as needed. For example, for different upper-layer AI businesses, the AI Ops team can combine applicable add-ons to set up diverse AI environments based on various business needs.

Directions

Creating an AI environment

1. Log in to the TKE console and click **Cloud Native AI** on the left sidebar.
2. On the **AI Environment** list page, click **Create** to enter the **Create AI Environment** page and set the parameters.

- **Environment Name**: Custom environment name. You can name the environment based on information such as business needs to facilitate subsequent resource management.
- **Region**: The region of the cluster where the AI environment is to be deployed.
- **Cluster Type**: The type of cluster where the AI environment is to be deployed.
- **Cluster**: The cluster where the AI environment is to be deployed.
- **Deploy Add-On**: Add-ons to be deployed in the AI environment. You can also install and delete add-ons in Add-On Management after environment creation.

5. Click **Create**.

Viewing an AI environment

After creating an AI environment, you can view it on the **AI Environment** list page.

Deleting an AI environment

1. Log in to the TKE console and click **Cloud Native AI** on the left sidebar.
2. On the **AI Environment** list page, click **Delete** on the right of the target environment.
3. In the **Delete AI Environment** pop-up window, read the notes on deletion and click **Confirm**.

AI Add-On Management

Last updated：2022-10-12 11:37:14

Overview

After creating an AI environment, you can combine AI add-ons as needed to set up an AI platform. This document describes how to add, delete, modify, and query AI add-ons.

Note：
The underlying layer of AI add-ons is implemented based on Helm Chart. After you create an AI environment, we recommend you not manage AI add-ons on the relevant page in the application marketplace. Instead, directly manage them **in the AI environment**, so as to avoid data inconsistency.

List of AI Add-Ons

Add-On	Use Case	Description
TF Operator	Model training	After installing it, you can run TF standalone/distributed training jobs.
MPI Operator	Elastic training	You can run elastic training jobs to fully utilize the computing resources.
Fluid	Cache acceleration	Fluid provides data prefetch and acceleration for cloud applications by using a distributed cache engine (GooseFS/Alluxio) with data observability, portability, and horizontal scalability.
Elastic Jupyter Operator	Algorithm debugging	Elastic Jupyter Operator provides an on-demand elastic Jupyter Notebook service to assign computing resources as needed.

AI Add-On Lifecycle Management

Creating an AI add-on

1. Log in to the TKE console and click **Cloud Native AI** on the left sidebar.

2. On the **AI Environment** list page, click the ID of the target AI environment to enter its **Basic Information** page.
3. On the left sidebar, click **Add-On Management**.
4. Click **Create** to enter the **Create AI Add-On** page and set the parameters.

   The main parameters are described as follows:

   - Add-On Name: Custom add-on name.
   - Namespace: The namespace for installing the add-on.
   - Chart: The installation package of the add-on. Only one add-on can be installed at a time.
   - Parameter: Add-on configuration parameters. After the add-on is created, you can still update its parameters as instructed in "Updating an AI add-on".
5. Click **Done**.

Viewing an AI add-on

After creating an AI environment, you can view the list of installed AI add-ons in the AI environment.

Deleting an AI add-on

1. Select the ID of an AI environment to enter its **Basic Information** page.
2. On the left sidebar, click **Add-On Management**.
3. Select **Delete** on the right of the target add-on.
4. In the **Delete Add-On** pop-up window, read the notes on deletion and click **Confirm**.

Updating an AI add-on

1. Select the ID of an AI environment to enter its **Basic Information** page.
2. On the left sidebar, click **Add-On Management**.
3. Select **Update configuration** on the right of the target add-on.
4. On the **Update Add-On** pop-up page, configure add-on parameters as needed and click **Done**.

AI Component List
Fluid

Last updated：2022-06-21 11:14:06

Overview

Fluid is an open-source Kubernetes-native distributed dataset orchestrator and accelerator for data-intensive applications, such as big data and AI. It is hosted by the Cloud Native Computing Foundation (CNCF) as a sandbox project. By defining the abstraction of dataset resources, it features:

- **Native support for dataset abstraction**: Implements the basic capabilities required for data-intensive applications to achieve efficient data access and reduce the cost of multidimensional management.
- **Cloud data prefetch and acceleration**: Fluid provides data prefetch and acceleration for cloud applications by using a distributed cache engine (GooseFS/Alluxio) with data observability, portability, and horizontal scalability.
- **Co-orchestration for data and applications**: During application and data scheduling on the cloud, it takes their characteristics and location into consideration to improve the performance.
- **Multi-namespace management support**: Allows you to create and manage datasets in different namespaces.
- **Heterogeneous data source management**: Unifies the access to underlying data from different sources (COS, HDFS, and Ceph), applicable to hybrid cloud use cases.

Key Concepts

Dataset: A dataset is a set of logically related data that can be used by computing engines, such as Spark for big data and TensorFlow for AI. Smart data applications create core industry values. Managing datasets may require features in different dimensions, such as security, version management, and data acceleration.

Runtime: The execution engine that enforces dataset security and provides version management and data acceleration capabilities. It defines a set of APIs for dataset management and acceleration throughout the lifecycle.

GooseFS Runtime: It is a Java-based implementation of the execution engine developed by Tencent Cloud's COS team, supporting dataset management, caching, and COS. GooseFS is a Tencent Cloud product with dedicated product-level support, but its code is not open-source. Fluid enables dataset visualization, elastic scaling, and data migration by managing and scheduling GooseFS Runtime.

Alluxio Runtime: Based on open-source Alluxio, it is an implementation of the execution engine for dataset management and caching, supporting PVC, Ceph, and CPFS computing, thereby effectively supporting hybrid cloud use cases. Alluxio is an open-source scheme. In spite of the joint efforts of Tencent Cloud and the community to

promote the stability and performance of its data caching, there will be a delay in timeliness and response. Fluid enables dataset visualization, elastic scaling, and data migration by managing and scheduling Alluxio Runtime.

-	Alluxio	GooseFS
Underlying storage types	PVC, Ceph, HDFS, CPFS, NFS	OSS, EMR, PVC, Ceph, HDFS, CPFS, NFS
Support	Open-source community	Tencent Cloud products

Add-on Installation

Prerequisite dependencies

- Kubernetes cluster (v1.14 or later)
- CSI support in the cluster

Parameter configuration

During Helm deployment, all configuration items are included in `values.yaml`.
Some fields may need to be customized, as listed below:

Parameter	Description	Default Value
`workdir`	Backup address of the metadata in the cache engine	`/tmp`
`dataset.controller.image.repository`	Repository where the dataset controller image resides	`ccr.ccs.tencentyun.com/controller`
`dataset.controller.image.tag`	Dataset controller image version	`"v0.6.0-0bfc552"`

Parameter	Description	Default Value
`csi.registrar.image.repository`	Repository where the CSI registrar image resides	`"ccr.ccs.tencentyun.co driver-registrar"`
`csi.registrar.image.tag`	CSI registrar image version	`"v1.2.0"`
`csi.plugins.image.repository`	Repository where the CSI plugins image resides	`"ccr.ccs.tencentyun.co`
`csi.plugins.image.tag`	CSI plugins image version	`"v0.6.0-def5316"`
`csi.kubelet.rootDir`	`kubelet root` folder	`"/var/lib/kubelet"`
`runtime.mountRoot`	Root address of the FUSE mount in the cache engine	`"/var/lib/kubelet"`
`runtime.goosefs.enable`	Enable GooseFS cache engine	`"true"`

Parameter	Description	Default Value
`runtime.goosefs.init.image.repository`	Repository where the initialized image of the GooseFS cache engine resides	`"ccr.ccs.tencentyun.co`
`runtime.goosefs.init.image.tag`	Version of the initialized image of the GooseFS cache engine	`"v0.6.0-0cd802e"`
`runtime.goosefs.controller.image.repository`	Repository where the controller image of the GooseFS cache engine resides	`"ccr.ccs.tencentyun.co` `controller"`
`runtime.goosefs.controller.image.tag`	Version of the controller image of the GooseFS cache engine	`"v0.6.0-bbf4ea0"`

Parameter	Description	Default Value
`runtime.goosefs.runtime.image.repository`	Repository where the GooseFS cache engine image resides	`"ccr.ccs.tencentyun.co`
`runtime.goosefs.runtime.image.tag`	Version of the GooseFS cache engine image	`"v1.1.10"`
`runtime.goosefs.fuse.image.repository`	Repository where the FUSE add-on image of the GooseFS cache engine resides	`"ccr.ccs.tencentyun.co`
`runtime.goosefs.fuse.image.tag`	Version of the FUSE add-on image of the GooseFS cache engine	`"v1.1.10"`
`runtime.alluxio.runtimeWorkers`	Maximum number of the concurrent workers of the Alluxio cache engine controller	`"3"`

Parameter	Description	Default Value
`runtime.alluxio.portRange`	Alluxio cache engine add-on port range	`"20000-26000"`
`runtime.alluxio.enable`	Enable Alluxio cache engine	`"true"`
`runtime.alluxio.init.image.repository`	Repository where the initialization image of the Alluxio cache engine resides	`"ccr.ccs.tencentyun.co`
`runtime.alluxio.init.image.tag`	Version of the initialization image of the Alluxio cache engine	`"v0.6.0-def5316"`
`runtime.alluxio.controller.image.repository`	Repository where the controller image of the Alluxio cache engine resides	`"ccr.ccs.tencentyun.co` `controller"`
`runtime.alluxio.controller.image.tag`	Version of the controller image of the Alluxio cache engine	`"v0.6.0-0cd802e"`

Parameter	Description	Default Value
`runtime.alluxio.runtime.image.repository`	Repository where the Alluxio cache engine image resides	`"ccr.ccs.tencentyun.co`
`runtime.alluxio.runtime.image.tag`	Version of the Alluxio cache engine image	`"release-2.5.0-2-SNAPS`
`runtime.alluxio.fuse.image.repository`	Repository where the FUSE add-on image of the Alluxio cache engine resides	`"ccr.ccs.tencentyun.co` `fuse"`
`runtime.alluxio.fuse.image.tag`	Version of the FUSE add-on image of the Alluxio cache engine	`"release-2.5.0-2-SNAPS`

Best Practices

For more information, see the Fluid documentation.

TF Operator

Last updated：2022-10-12 11:37:14

Overview

Developed by the Kubeflow community, TF-Operator is an add-on used to help deploy and execute TensorFlow distributed training jobs in a Kubernetes cluster.

After deployment, you can create, view, and delete TF jobs.

Prerequisite dependencies

Kubernetes cluster (v1.16 or later)

Deployment

During Helm deployment, all configuration items are included in `values.yaml`.

Some fields may need to be customized, as listed below:

Parameter	Description	Default Value
`image.repository`	The repository where the TF-Operator image resides	`ccr.ccs.tencentyun.com/kubeflow-oteam/tf-operator`
`image.tag`	TF-Operator image version	`"latest"`
`namespace.create`	Whether to create a separate namespace for TF-Operator	`true`
`namespace.name`	The namespace where TF-Operator is to be deployed	`"tf-operator"`

Best practices

See Running TF Training Job.

MPI Operator

Last updated：2022-10-12 11:37:14

Overview

Developed by the Kubeflow community, MPI-Operator is an add-on used to help deploy and execute data-parallel distributed training such as Horovod in a Kubernetes cluster.

After deployment, you can create, view, and delete MPI jobs.

Prerequisite dependencies

Kubernetes cluster (v1.16 or later)

Deployment

During Helm deployment, all configuration items are included in `values.yaml`.

Some fields may need to be customized, as listed below:

Parameter	Description	Default Value
`image.repository`	The repository where the MPI-Operator image resides	`ccr.ccs.tencentyun.com/kubeflow-oteam/mpi-operator`
`image.tag`	MPI-Operator image version	`"latest"`
`namespace.create`	Whether to create a separate namespace for MPI-Operator	`true`
`namespace.name`	The namespace where MPI-Operator is to be deployed	`"mpi-operator"`

Elastic Jupyter Operator

Last updated：2022-10-12 16:05:09

Overview

elastic-jupyter-operator is a native elastic Jupyter service in Kubernetes. It provides an elastic Jupyter Notebook service as needed with the following features:

- It automatically releases resources to the Kubernetes cluster when the GPU is idle.
- It supports delayed resource application, allowing you to apply for CPU, memory, and GPU resources as needed.
- Multiple Jupyter notebooks share a resource pool to increase the resource utilization.

Deployment

During Helm deployment, all configuration items are included in `values.yaml`.

Some fields may need to be customized, as listed below:

Parameter	Description	Default Value
`image.repository`	The repository where the image resides	`ccr.ccs.tencentyun.com/kubeflow-oteam/elastic-jupyter-operator`
`image.tag`	Image version	`"v0.1.1"`
`namespace.name`	Namespace	`"enterprise-gateway"`

How to use

Note：
For more information, see elastic-jupyter-operator.

1. Run the following command to create a Jupyter Gateway CR:

```
kubectl apply -f ./config/samples/kubeflow.tkestack.io_v1alpha1_jupytergateway.yaml
```

Below is the content of the YAML file:

```
apiVersion: kubeflow.tkestack.io/v1alpha1
kind: JupyterGateway
metadata:
name: jupytergateway-sample
spec:
cullIdleTimeout: 3600
```

Here, `cullIdleTimeout` is a configuration item. If a kernel is idle in the time in seconds specified by `cullIdleTimeout` , Gateway will repossess it to release resources.

2. Run the following command to create a Jupyter Notebook CR instance and specify the Gateway CR:

```
kubectl apply -f ./config/samples/kubeflow.tkestack.io_v1alpha1_jupyternotebook.y
aml
```

Below is the content of the YAML file:

```
apiVersion: kubeflow.tkestack.io/v1alpha1
kind: JupyterNotebook
metadata:
name: jupyternotebook-sample
spec:
gateway:
name: jupytergateway-sample
namespace: default
```

3. All resources in the cluster are as listed below:

```
NAME READY STATUS RESTARTS AGE
pod/jupytergateway-sample-6d5d97949c-p8bj6 1/1 Running 2 11d
pod/jupyternotebook-sample-5bf7d9d9fb-nq9b8 1/1 Running 2 11d
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/jupytergateway-sample ClusterIP 10.96.138.111 <none> 8888/TCP 11d
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 31d
NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/jupytergateway-sample 1/1 1 1 11d
deployment.apps/jupyternotebook-sample 1/1 1 1 11d
NAME DESIRED CURRENT READY AGE
replicaset.apps/jupytergateway-sample-6d5d97949c 1 1 1 11d
replicaset.apps/jupyternotebook-sample-5bf7d9d9fb 1 1 1 11d
```

4. Use a method such as NodePort, `kubectl port-forward` , or Ingress to expose Notebook CR to provide the Service. Here, `kubectl port-forward` is used as an example. Run the following command:

```
kubectl port-forward jupyternotebook-sample-5bf7d9d9fb-nq9b8 8888
```

API documentation

See API Reference.

Model Training
Running TF Training Job

Last updated：2023-05-19 17:12:40

This document describes how to run a TF training job.

Prerequisites

- TF Operator has been installed in your AI environment.
- Your AI environment has GPU resources.

Directions

The following steps are based on the official distributed training examples in parameter server/worker mode of `TF-Operator`.

Preparing the training code

The code sample dist_mnist.py at the official website of Kubeflow is used.

Creating a training image

Image creation is easy. You only need to get an official image based on TensorFlow 1.5.0, copy the above code to the image, and configure `entrypoint`.

> Note：
> If `entrypoint` is not configured, you can also configure the container startup command when submitting a `TFJob`.

Submitting the job

1. Prepare a `TFJob` YAML file to define two parameter servers and four workers.

> Note

You need to replace the `<training image="">` placeholder with the address of the uploaded training image.

```
apiVersion: "kubeflow.org/v1"
kind: "TFJob"
metadata:
name: "dist-mnist-for-e2e-test"
spec:
tfReplicaSpecs:
PS:
replicas: 2
restartPolicy: Never
template:
spec:
containers:
- name: tensorflow
image: <training image>
Worker:
replicas: 4
restartPolicy: Never
template:
spec:
containers:
- name: tensorflow
image: <training image>
```

2. Run the following command to use `kubectl` to submit the `TFJob` :

```
kubectl create -f ./tf_job_mnist.yaml
```

3. Run the following command to view the job status:

```
kubectl get tfjob dist-mnist-for-e2e-test -o yaml
kubectl get pods -l pytorch_job_name=pytorch-tcp-dist-mnist
```

Running PyTorch Training Job

Last updated：2023-05-19 17:07:37

This document describes how to run a PyTorch training job.

Prerequisites

- PyTorch Operator has been installed in your AI environment.
- Your AI environment has GPU resources.

Directions

The following steps are based on the official distributed training examples of `PyTorch-Operator` .

Preparing the training code

The code sample mnist.py at the official website of Kubeflow is used.

Creating a training image

Training image creation is easy. You only need to get an official image based on PyTorch 1.0, copy the above code to the image, and configure `entrypoint` (if `entrypoint` is not configured, you can also configure the startup command when submitting a `PyTorchJob`).

> Note：
> The training code is written based on PyTorch 1.0. As APIs of different PyTorch versions may be incompatible, you may need to adjust the above training code in a PyTorch environment on other versions.

Submitting the job

1. Prepare a `PyTorchJob` YAML file to define one master worker and one worker.

> Note
> - You need to replace the `<training image="">` placeholder with the address of the uploaded training image.

- As GPU resources are configured in resource configuration, set `backend` for training to `"nccl"` in `args`; in jobs using no (Nvidia) GPU resources, use another backend such as `gloo`.

```
apiVersion: "kubeflow.org/v1"
kind: "PyTorchJob"
metadata:
name: "pytorch-dist-mnist-nccl"
spec:
pytorchReplicaSpecs:
Master:
replicas: 1
restartPolicy: OnFailure
template:
metadata:
annotations:
sidecar.istio.io/inject: "false"
spec:
containers:
- name: pytorch
image: <training image>
args: ["--backend", "nccl"]
resources:
limits:
nvidia.com/gpu: 1
Worker:
replicas: 1
restartPolicy: OnFailure
template:
metadata:
annotations:
sidecar.istio.io/inject: "false"
spec:
containers:
- name: pytorch
image: <training image>
args: ["--backend", "nccl"]
resources:
limits:
nvidia.com/gpu: 1
```

2. Run the following command to use `kubectl` to submit the `PyTorchJob`:

```
kubectl create -f ./pytorch_job_mnist_nccl.yaml
```

3. Run the following command to view the `PyTorchJob` :

```
kubectl get -o yaml pytorchjobs pytorch-dist-mnist-nccl
```

4. Run the following command to view Pods created by the PyTorch job:

```
kubectl get pods -l pytorch_job_name=pytorch-dist-mnist-nccl
```

TMP
TMP Overview

Last updated：2023-02-02 17:05:22

Overview

Tencent Managed Service for Prometheus (TMP) is a monitoring and alarming solution specially optimized for cloud-native service scenarios. It has the full monitoring capabilities of open-source Prometheus and provides lightweight, stable, and highly available cloud-native monitoring services. It eliminates your need to build a Prometheus monitoring system on your own or care about issues such as data storage, data display, and system Ops, and enables you to enjoy a high-performance multi-cluster Prometheus monitoring service after simple configuration.

Prometheus overview

Prometheus is an open-source system monitoring and alarming framework. It completely disrupts the testing and alarming models of traditional monitoring systems by forming a new model based on centralized rule computing and unified analysis and alarming. As a project in Cloud Native Computing Foundation with a popularity only second to Kubernetes, it has gradually become a core monitoring component in the era of cloud native thanks to its powerful standalone performance, flexible PromQL, and active community ecosystem.

Strengths of Prometheus

- Support for powerful multidimensional data models.
- Built-in flexible query language PromQL.
- Support for all-around monitoring.
- Great openness.
- Support for target discovery and collection through dynamic service or static configuration.

Shortcomings of open-source Prometheus

- The native Prometheus is deployed on a single server and does not provide cluster features, which makes it impossible to monitor large clusters.
- It cannot easily implement dynamic scaling and load balancing.
- It is technically difficult to deploy and get started with.

Comparison between TMP and open-source Prometheus

Comparison Item	TMP	Open-Source Prometheus

Comparison Item	TMP	Open-Source Prometheus
Scenario	Optimized for container cloud-native scenarios and allows you to use the Integration Center to implement the monitoring of non-container scenarios	Oriented to multiple scenarios
Weight	Super lightweight	High memory usage
Stability	Higher than native	Not guaranteed
Availability	High	Low
Data storage capability	Unlimited	Subject to local disk capacity
Monitoring of ultra large cluster	Supported	Not supported
Data visualization	Excellent visualization capabilities based on Grafana and data display of multiple monitoring instances at the same time on Grafana	Limited visualization capabilities based on native Prometheus UI
Open-Source ecosystem	Full compatibility	Native support
Barrier to use	Low	High
Cost	Low	High
Cross-cluster collection	Supported	Not supported
Cross-region and cross-VPC collection	Supports cluster data collection of other regions and VPCs as well as associating with a cluster in TMP	Not supported
Alarming policy configuration	Rich alarm and notification templates	Manual configuration needed

Benefits

Full compatibility with the configurations and core APIs of Prometheus to retain the native features and strengths of Prometheus

TMP supports custom multidimensional data models.

TMP has the built-in flexible query language PromQL.

TMP supports target discovery and collection through dynamic service or static configuration.

TMP is compatible with core Prometheus APIs.

Support for monitoring ultra large clusters

In the performance stress test for a single Prometheus server, when the number of series exceeds 3 million (the length of each label and its value is fixed at 10 characters), the memory usage increases significantly to over 20 GB; therefore, a large-memory server is required for running Prometheus.

TMP can monitor ultra large clusters based on its proprietary sharding technology.

Support for monitoring cross-VPC clusters in one instance

One instance can be associated with multiple clusters. Clusters from other VPCs can be monitored.

Support for template-based management and configuration

TMP allows you to configure templates for monitoring multiple instances and clusters. Then, you can use a template to quickly implement unified multi-cluster monitoring.

Ultra lightweight and non-intrusion monitoring

TMP is lighter than open-source Prometheus, which uses 16–128 GB memory. In contrast, TMP only requires the deployment of a small agent in your cluster, which uses only 20 MB memory to monitor a cluster with 100 nodes. In addition, its memory usage will never exceed 1 GB no matter how large a cluster is.

After you associate your cluster, TMP will automatically deploy the agent in it, so you can start monitoring your businesses without manually installing any add-on. The super lightweight agent has no impact on the businesses and add-ons in your cluster.

Support for real-time dynamic scaling to meet elastic needs

TMP uses Tencent Cloud's proprietary sharding and scheduling technologies to implement real-time dynamic scaling of collection tasks, meeting your elastic needs. It also supports load balancing.

High availability

TPS uses technical methods to avoid data breakpoints and losses, so as to secure high availability of the monitoring service.

Low connection costs

You can write configuration files easily in the TMP console, so you don't need to have an extensive knowledge of Prometheus to use TMP. If you already know how to use Prometheus, TMP also allows you to submit configuration information through a native YAML file, making it easier for you to customize advanced features for personalized monitoring.

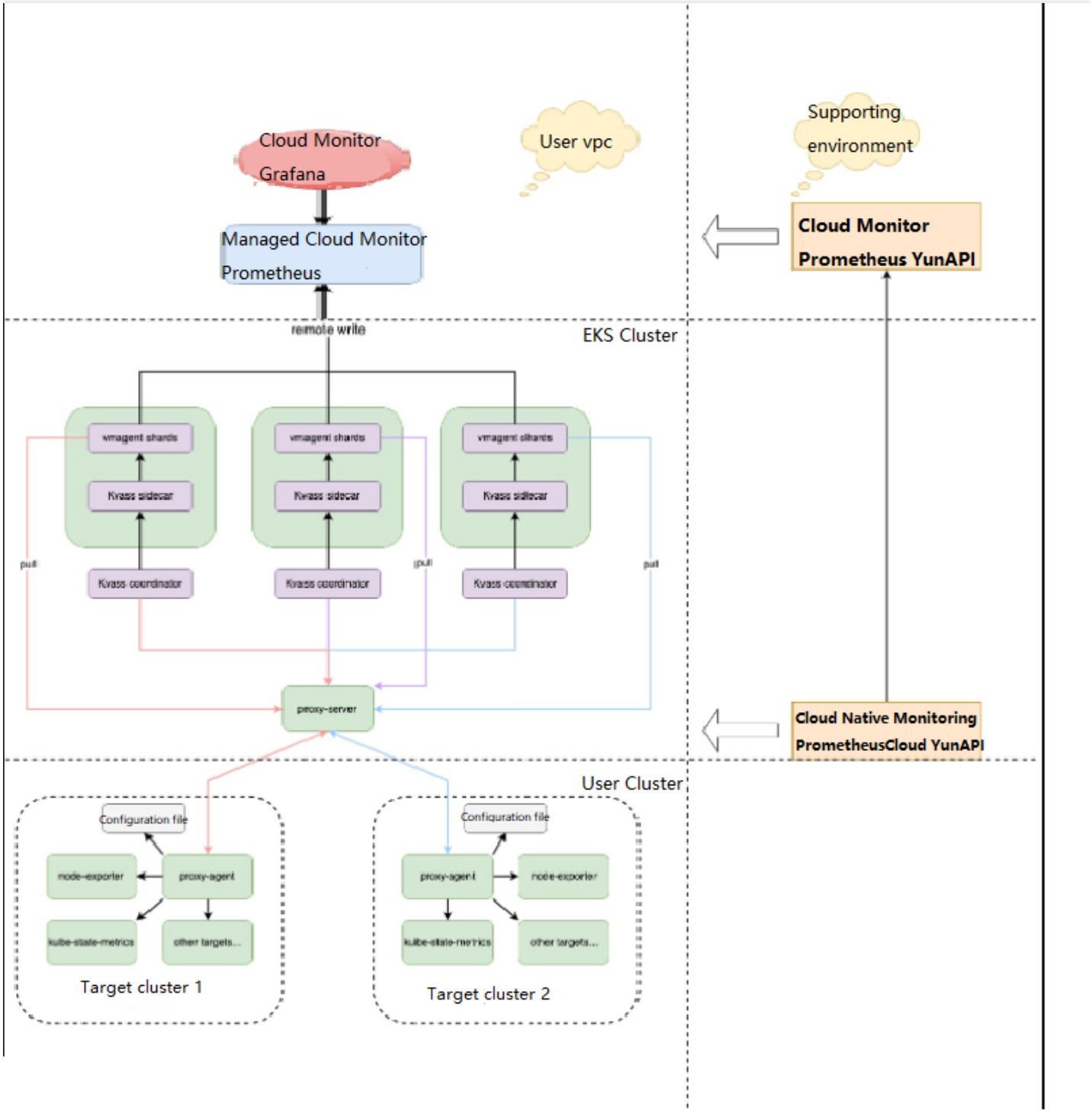
For more information, see Strengths.

Product Architecture

TMP is a super lightweight, highly available, and non-intrusion monitoring system.

- It contains only one lightweight agent in your cluster.
- The collector is a TKE Serverless cluster created under your account and doesn't affect your native clusters.
- Monitoring data storage and display are achieved through separate modules.
- Cloud Monitor Grafana is connected to multiple monitoring instances for a unified view.

The product architecture is as shown below:



TMP can monitor cross-region and cross-VPC clusters, businesses outside clusters in the same VPC, and ultra large clusters. It also supports real-time scaling of the monitoring add-on to secure the high availability of monitoring services.

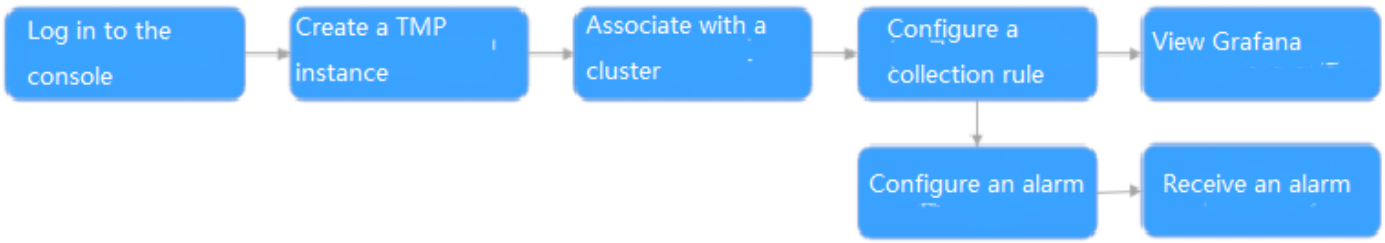
After you associate a cluster, TMP will add the mainstream collection configuration from the community by default, making it available out of the box without any custom configuration required.

In addition, TMP is preset with common Grafana dashboards and alarm rule templates.

Directions

Use your Tencent Cloud account to log in to the TMP console:

1. Create an instance.
2. Associate a cluster with the newly created instance. At this point, the system will automatically deploy the agent in your cluster and monitoring add-on in your newly created TKE Serverless cluster, so you don't need to install any add-ons.
3. Configure a collection rule. After a cluster is successfully associated, you can flexibly configure the data collection and alarm rules as needed. Then, you can open Grafana to view the monitoring data.



Key concepts

- **Instance:** An instance corresponds to a complete set of monitoring services and has an independent GUI. It can be associated with multiple clusters in the same VPC to implement unified multi-cluster monitoring.
- **Cluster:** Generally indicates your TKE or TKE Serverless cluster on Tencent Cloud.
- **Cluster association:** Indicates the operation of associating an instance with a cluster.
- **Collection rule:** Indicates a custom monitoring data collection rule.
- **Job:** In Prometheus, a job is a collection task, which defines the public configurations of all monitoring targets in a job. Multiple jobs can form the configuration file of a collection task.
- **Target:** Indicates a data collection target obtained through static configuration or service discovery. For example, when a Pod is monitored, the target will be each container in the Pod.

- **Metric:** It is used to record the monitoring metric data. All metrics are time series data and identified by name. The sample data collected by each metric contains information in at least three dimensions (metric name, time, and metric value).
- **Series:** It is a metric-label pair displayed as a straight line on a dashboard.

Use Cases

TMP mainly monitors container cloud-native business use cases. In addition to the implementation of mainstream container and Kubernetes monitoring solutions, it also flexibly supports custom monitoring of your businesses, gradually optimizes the preset dashboards in different use cases, and continuously summarizes industry-specific best practices, in order to help you perform multidimensional analysis and personalized display of monitoring data. It is committed to becoming the best monitoring solution in container use cases.

Pricing

For TMP pricing details, see Pay-as-You-Go.
Currently, when you use the TMP service, TKE Serverless clusters and Cloud Load Balancer resources will be created under your account and billed in pay-as-you-go mode. For more information on the created resources and pricing, see Billing Mode and Resource Usage.

Create a monitoring instance

Last updated：2022-12-27 15:28:34

Overview

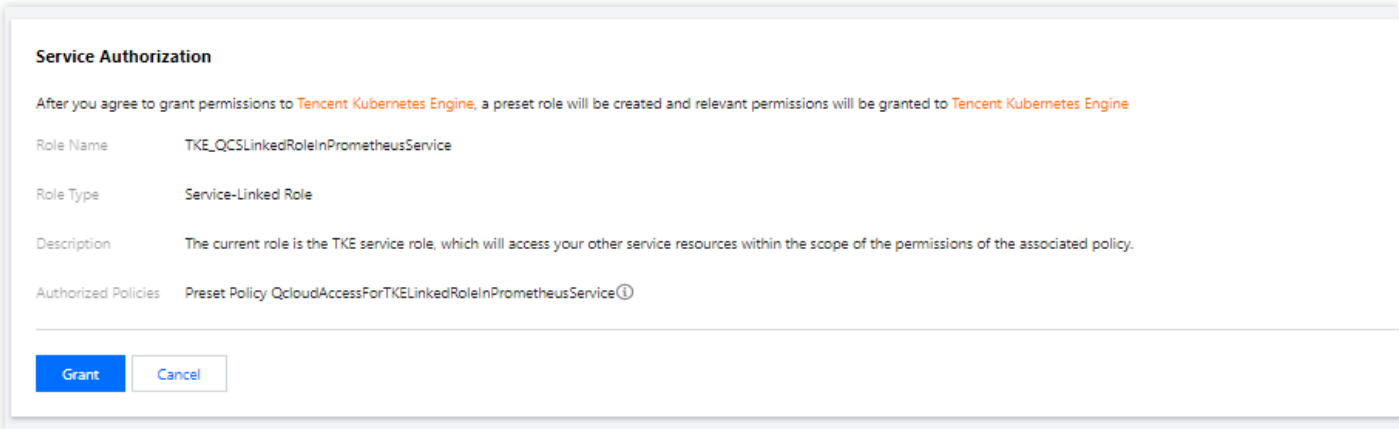
You can easily create a TMP instance and associate it with a cluster in the current region. Clusters associated with the same TMP instance share the same monitoring metrics and alarming policies. Currently, TMP supports managed clusters, self-deployed clusters, serverless clusters and edge clusters. This document describes how to create and manage TMP instances in the TKE console.

Directions

Service authorization

When using TMP for the first time, you need to assign the `TKE_QCSLinkedRoleInPrometheusService` role to the service, which is used to authorize the TMP to access the COS bucket.

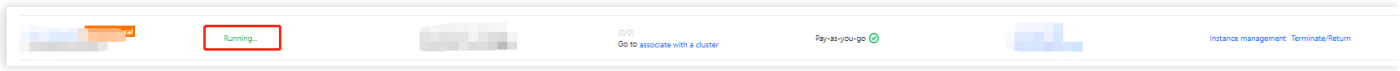
1. Log in to the TKE console and click **TMP** in the left sidebar to pop up the **Service authorization** window.
2. Click **Go to Cloud Access Management** to enter the **Role management** page.
3. Click **Grant** to complete authentication.



Creating TMP instance

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. Click **Create** at the top of the instance list page.
3. You will be redirected to the **Tencent Managed Service for Prometheus** page.
4. Purchase an instance as needed. For parameter details, see Creating Instance.

5. Click **Complete**. Now you can click **Associate with TKE** to see the list of TMP instances.
6. You can check the instance creation progress on the page. If the instance status changes to "Running", the instance was successfully created and is running properly.



Note：
If it takes too long to create an instance or the displayed status is abnormal, submit a ticket.

Associating with Cluster

Last updated：2023-05-19 15:21:46

Overview

This document describes how to associate clusters with TMP instances. When the association is established, you can edit configurations such as data collection rules. Currently, the service supports cross-VPC associations, allowing you to monitor clusters in multiple VPCs in different regions within the same TMP instance.

Prerequisites

- You have logged in to the TKE console and created a cluster.
- You have created a TMP instance.

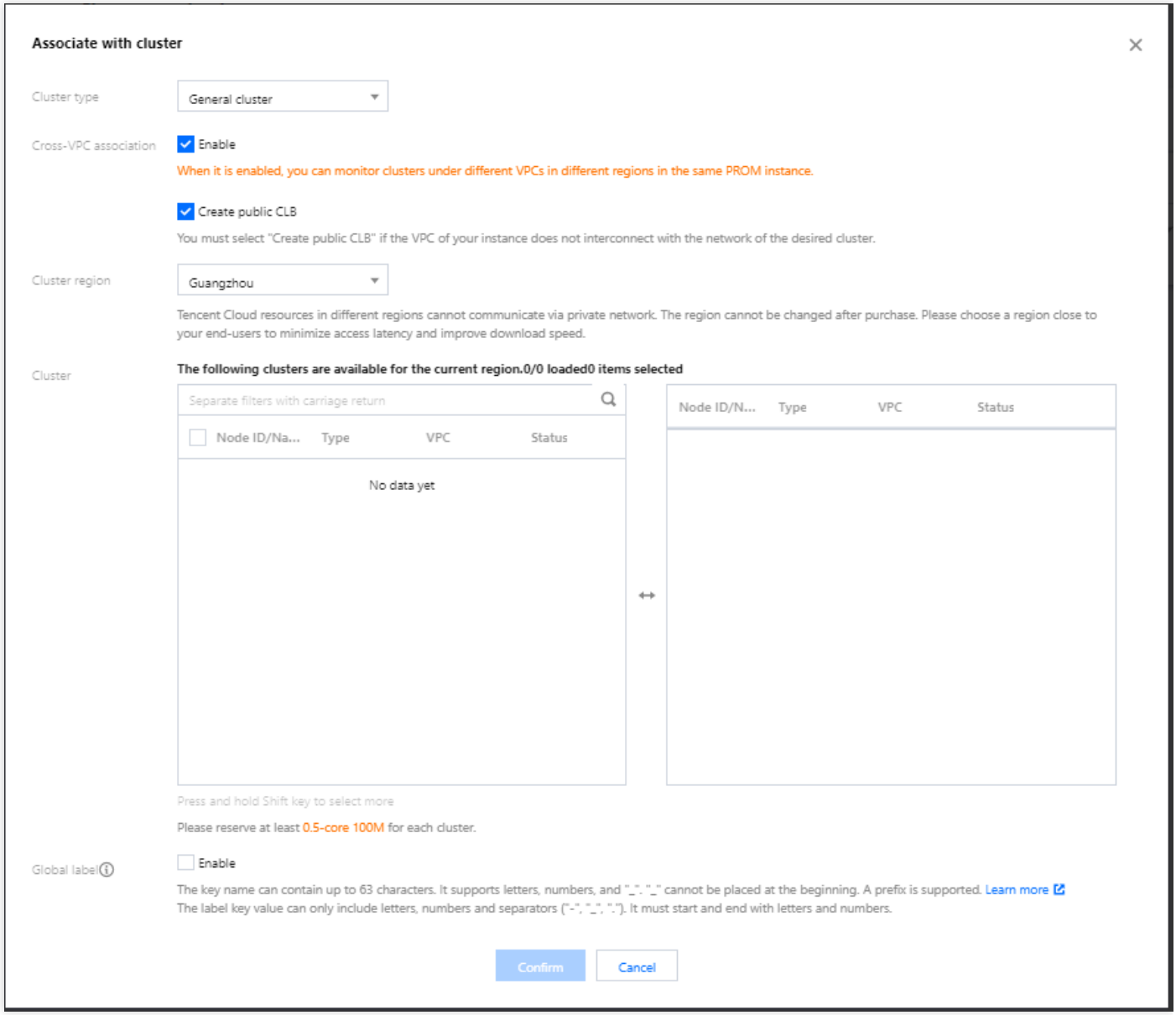
Directions

Associating with cluster

> Note：
> After the cluster is successfully associated, the monitoring data collection add-on will be installed in the cluster, which will be deleted when the cluster is disassociated.

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. On the instance list page, select the target instance to enter its details page.
3. On the **Cluster monitoring** page, click **Associate with Cluster**.

4. In the **Associate with cluster** pop-up window, select the target cluster.



- **Cluster type**: TKE general clusters, serverless clusters, edge clusters, and registered clusters are supported.
- **Cross-VPC association**: When it is enabled, you can monitor clusters in multiple VPCs in different regions within the same TMP instance.
  - Public CLB: You don't need to create a public CLB instance if your monitoring instance's VPC is connected to the VPC of the cluster you want to associate with; if not, you must select **Create public CLB**, otherwise, you cannot collect cluster data across VPCs. For example, if your instance VPC is already connected to the cluster VPC through Cloud Connect Network, you don't need to create a public CLB instance.

3. **Region**: Select the region where the cluster resides.
4. **Cluster**: Select one or multiple clusters to be associated with.

5. **Global tag**: It is used to tag each monitoring metric with the same key-value pair.
6. Click **OK**.

Canceling association

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. On the instance list page, select the target instance to enter its details page.
3. On the **Associate with cluster** page, click **Cancel association** on the right side of the instance.
4. Click **OK** in the **Disassociate cluster** pop-up window.

Data Collection Configurations

Last updated：2023-05-19 15:15:54

Scenarios

This document describes how to configure monitoring collection items for the associated cluster.

Prerequisites

Before configuring monitoring collection items, you need to perform the following operations:

- Create a TMP instance.
- Associate the desired clusters with the TMP instance.

How It Works

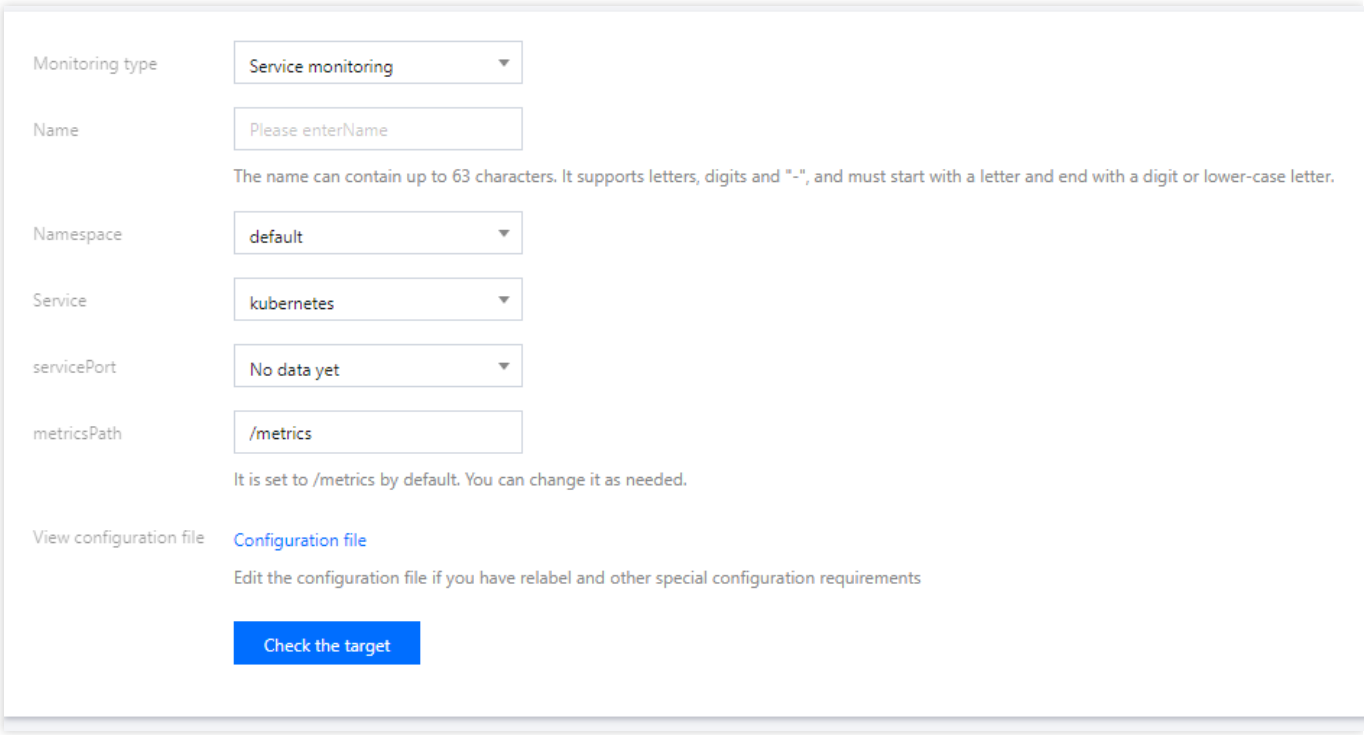
Configuring data collection

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. On the instance list page, select an instance name that needs to configure data collection rules to go to its details page.
3. On the **Cluster monitoring** page, click **Data collection** on the right of the instance to enter the collection configuration list page.
4. On the **Data collection** page, click **Custom monitoring** and add the data collection configuration. TMP has preset some collection configuration files to collect regular monitoring data. You can configure new data collection rules to monitor your business data by using the following two methods:
   - Adding data collection configuration via the console
   - Adding data collection configuration via yaml

   Monitoring a service

   i. Click **Add**.

ii. In the **Create collection policy** pop-up window, enter the configuration information.

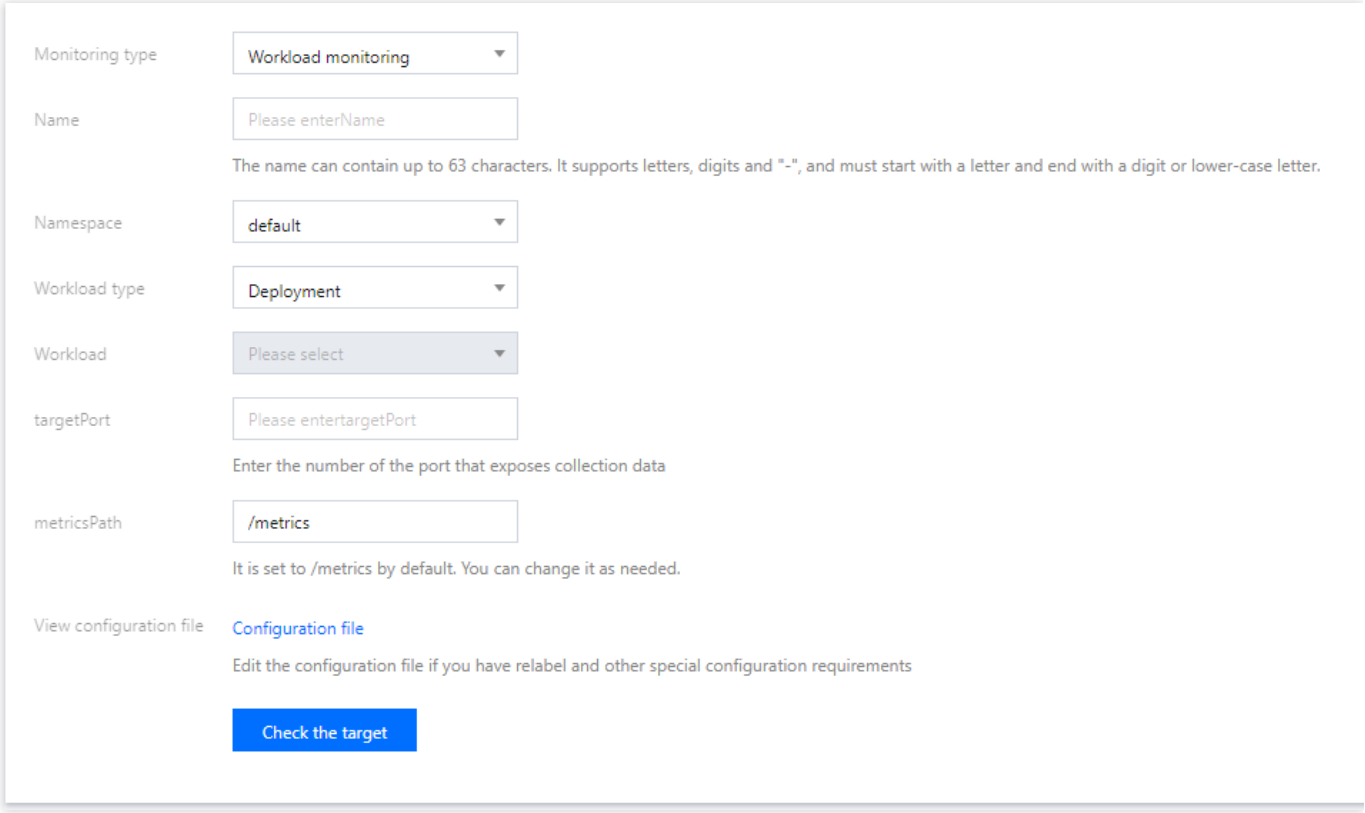


- **Monitoring type**: Select **Service monitoring**.
- **Name**: enter the rule name.
- **Namespace**: select the namespace to which the Service belongs.
- **Service**: select the service to be monitored.
- **ServicePort**: select the corresponding port.
- **MetricsPath**: defaults to `/metrics`. You can directly enter the collection API as needed.
- **View configuration file**: Click **Configuration file** to view the current configuration file. If you have special configuration requirements such as relabel, you can edit them in the configuration file.
- **Check the target**: Click **Check the target** to view a list of all targets that can be collected under the current collection policy, and confirm whether the collection policy meets your expectations.

Monitoring a workload

i. Click **Add**.

ii. In the **Create collection policy** pop-up window, enter the configuration information.

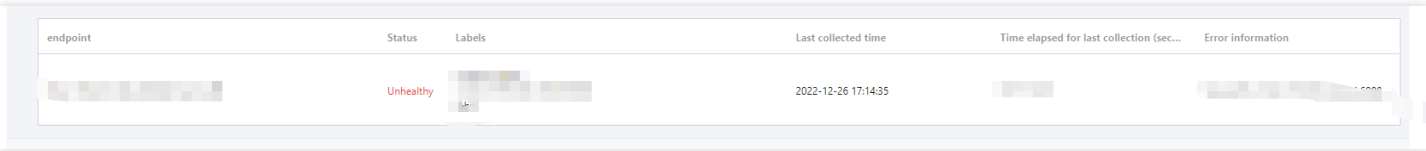


- **Monitoring type**: Select **Workload monitoring**.
- **Name**: enter the rule name.
- **Namespace**: select the namespace to which the workload belongs.
- **Workload type**: Select the workload type to be monitored.
- **Workload**: select the workload to be monitored.
- **targetPort**: enter the target port that exposes the collection metrics through which the collection target can be found. If the port is incorrect, the collection target will not be obtained correctly.
- **MetricsPath**: defaults to `/metrics` . You can directly enter the collection API as needed.
- **View configuration file**: Click **Configuration file** to view the current configuration file. If you have special configuration requirements such as relabel, you can edit them in the configuration file.
- **Check the target**: Click **Check the target** to view a list of all targets that can be collected under the current collection policy, and confirm whether the collection policy meets your expectations.

5. Click **OK**.
6. You can view the status of the collection target on the **Data collection** page of the instance.

targets (3/3) indicates three actually captured targets/three checked collection targets. When the number of actual captured targets equals to the number of checked targets, the status will be "up", which means that the current capture is normal. When the number of actual captured targets is less than the number of checked targets, the status will be "down", which means that some endpoints capture failed.

Click the field value (3/3) to view the details of the collection target. The "down" status is as follows:



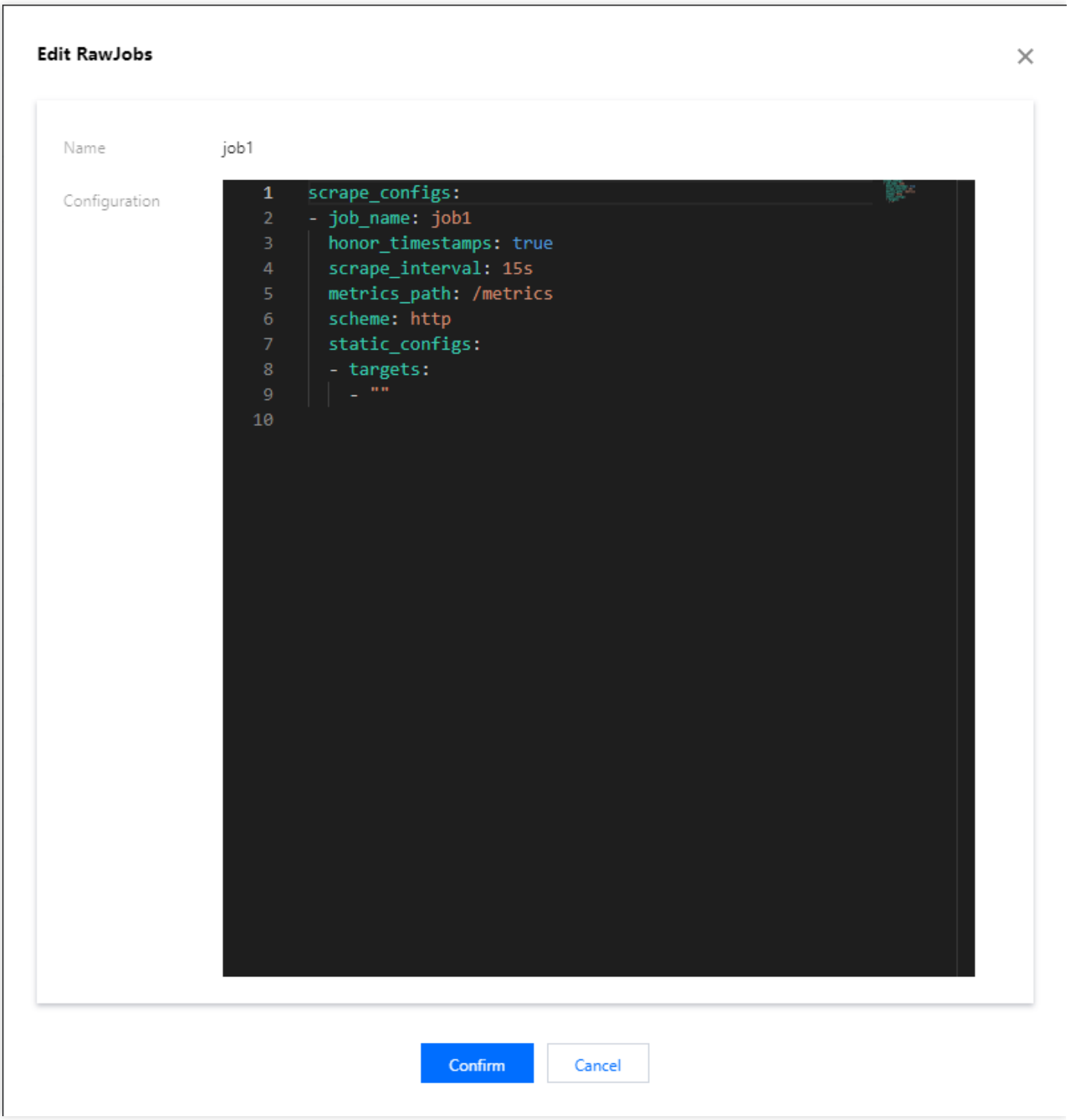
On the **Cluster monitoring** tab of the instance, click **More** > **Target Jobs** on the right of the cluster name to view all the collection targets of this cluster.

Viewing configuration

> Note：
> To view the configured YAML files, you can use only **Custom Monitoring** but not **Basic Monitoring**. All data collection configurations of the basic monitoring are productized, and you only need to **click**/**select** metrics to add/remove them.

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. On the instance list page, select an instance name that needs to configure data collection rules to go to its details page.
3. On the **Cluster Monitoring** page, select the **Associate with Cluster** tab and click **Data Collection** on the right of the instance to enter the data collection page. Select **Custom Monitoring** and click **Edit** on the right.

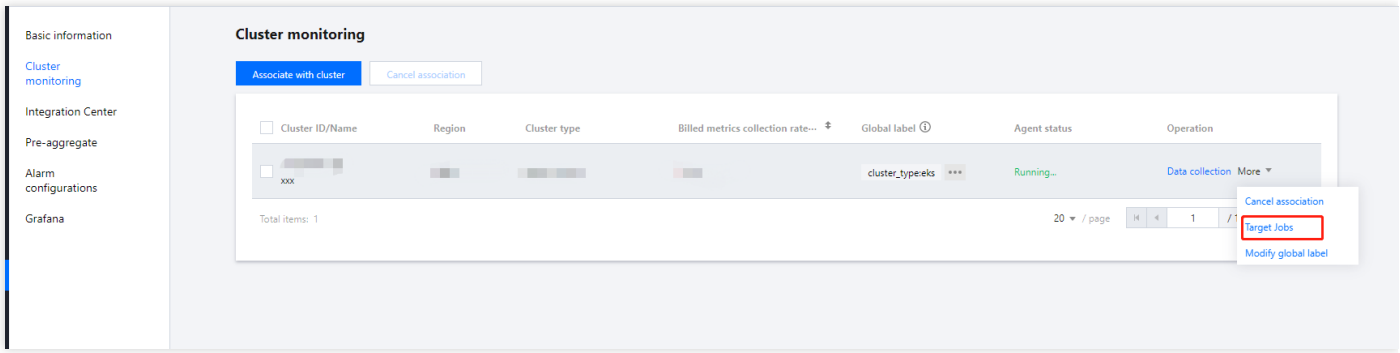
4. In the **Edit RawJobs** pop-up window, view all monitoring metrics configured in the YAML file.



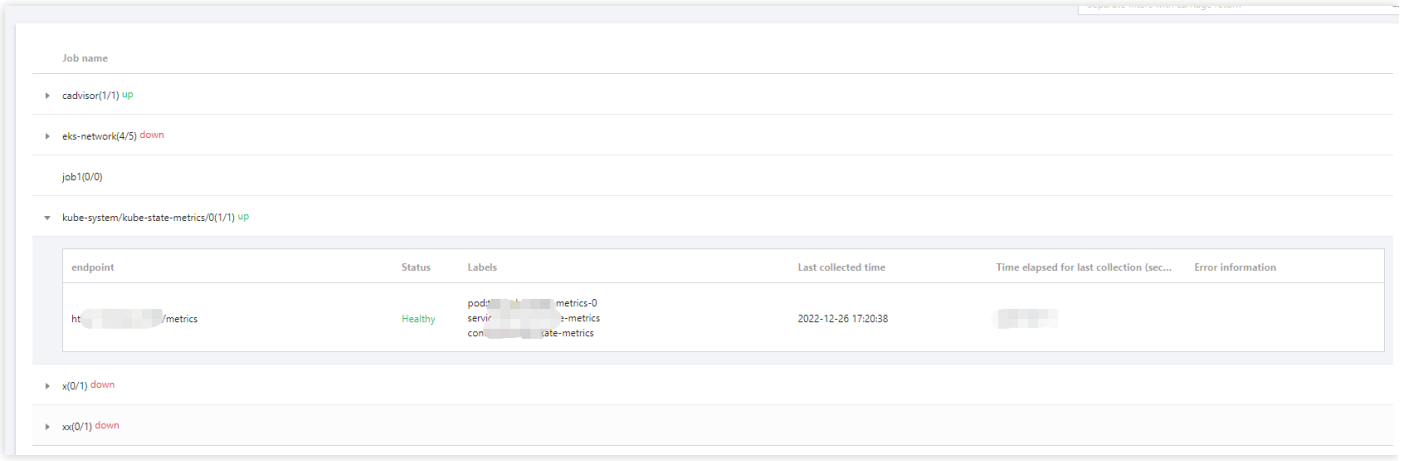
Viewing collection targets

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. On the instance list page, select an instance name that needs to view Targets and go to its details page.

3. On the **Cluster monitoring** page, click **More** > **Target Jobs** in the **Operation** column.



4. On the targets list page, view the collection status of current data.



Note：

- The endpoints in the status of "Unhealthy" are displayed at the top of the list by default.
- You can search the target by the resource attributes in the collection target page.

More

Mounting the file to the collector

When configuring the collection item, if you need to provide some files for the configuration, such as a certificate, you can mount the file to the collector in the following way, and the update of the file will be synchronized to the collector in real time.

- **prometheus.tke.tencent.cloud.com/scrape-mount = "true"**
  Add the above label to the configmap under the `prom-xxx` namespace, and all the keys will be mounted to the collector path `/etc/prometheus/configmaps/[configmap-name]/` .

- **prometheus.tke.tencent.cloud.com/scrape-mount = "true"**
  Add the above label to the secret under the `prom-xxx` namespace, and all the keys will be mounted to the collector path `/etc/prometheus/secrets/[secret-name]/` .

Streamlining Monitoring Metrics

Last updated：2023-05-06 19:41:07

Note

TMP has adjusted the free storage period for free metrics to 15 days on October 27, 2022. For instances with a storage period of more than 15 days, storage fees for their free metrics will be charged based on the excessive storage period. For more information on the billing rules, see Billing Rules for Free Metrics Exceeding Free Storage Period.

This document describes how to streamline the TMP **collection metrics** to avoid unnecessary expenses.

Prerequisites

Before configuring monitoring collection items, you need to perform the following operations:

Create a TMP instance.

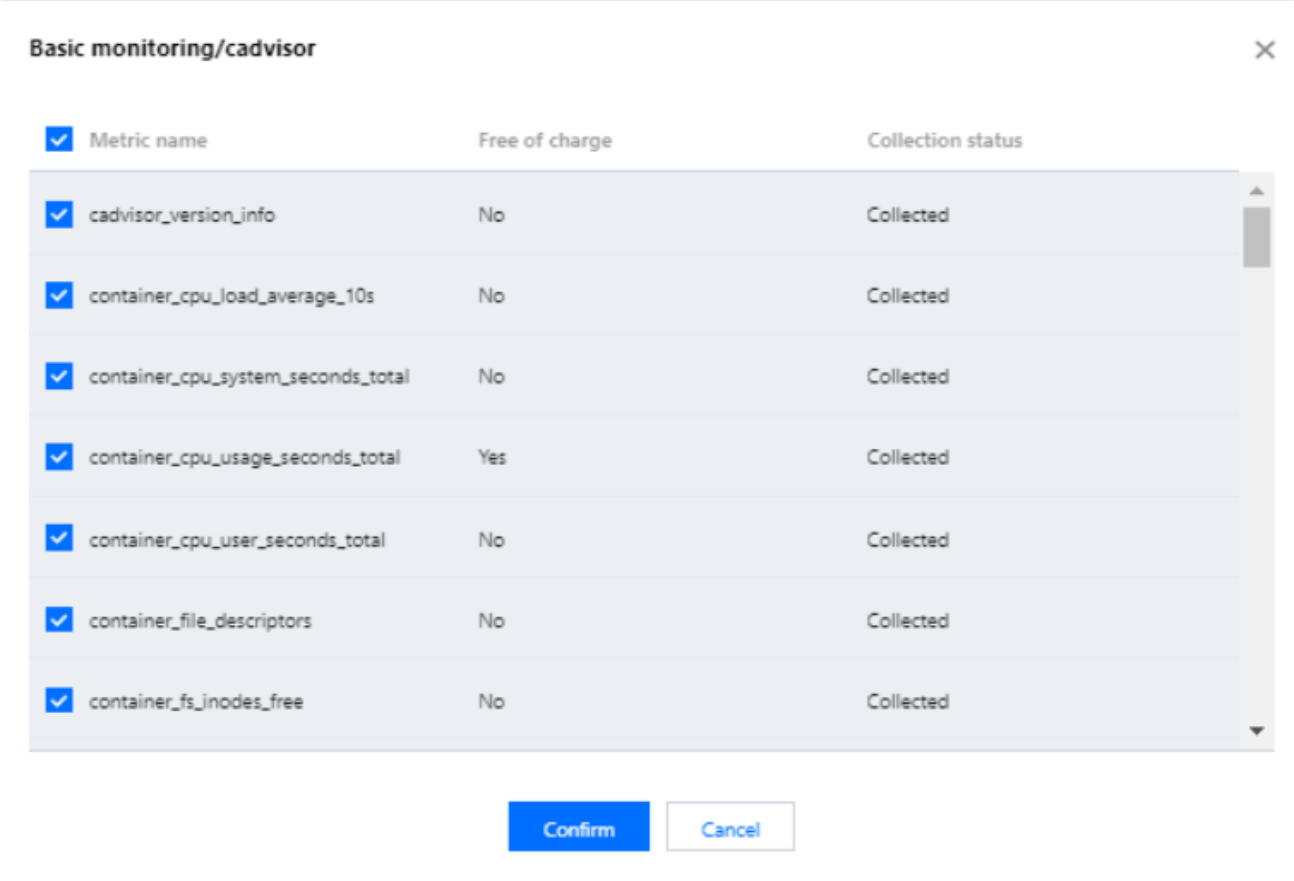
Associate the desired clusters with the TMP instance.

Streamlining Metrics

Streamlining metrics in the console

TMP offers more than 100 free basic monitoring metrics as listed in Free Metrics in Pay-as-You-Go Mode.

1. Log in to the TKE console and click TMP in the left sidebar.
2. On the instance list page, select an instance name that needs to configure data collection rules to go to its details page.
3. On the **Cluster monitoring** page, click **Data collection** on the right of the cluster to enter the collection configuration list page.
4. You can add or remove the basic metrics to be collected by selecting/unselecting the metrics. Click **Metric details** on the right.
5. The following shows whether the metrics are free. If you select a metric, it will be collected. We recommend you deselect paid metrics to avoid additional costs. Only metrics for basic monitoring are free of charge. For more information on free metrics, see Free Metrics in Pay-as-You-Go Mode. For more information on paid metrics, see Pay-as-You-Go.



Streamlining metrics through YAML

Currently, TMP is billed by the number of monitoring data points. We recommend you optimize your collection configuration to collect only required metrics and filter out unnecessary ones. This will save costs and reduce the overall reported data volume. For more information on the billing mode and cloud resource usage, see here.

The following describes how to add filtering configurations to ServiceMonitors, PodMonitors, and RawJobs to streamline custom metrics.

1. Log in to the TKE console and click TMP in the left sidebar.

2. On the instance list page, select an instance name that needs to configure data collection rules to go to its details page.

3. On the **Cluster monitoring** page, click **Data collection** on the right of the cluster to enter the collection configuration list page.

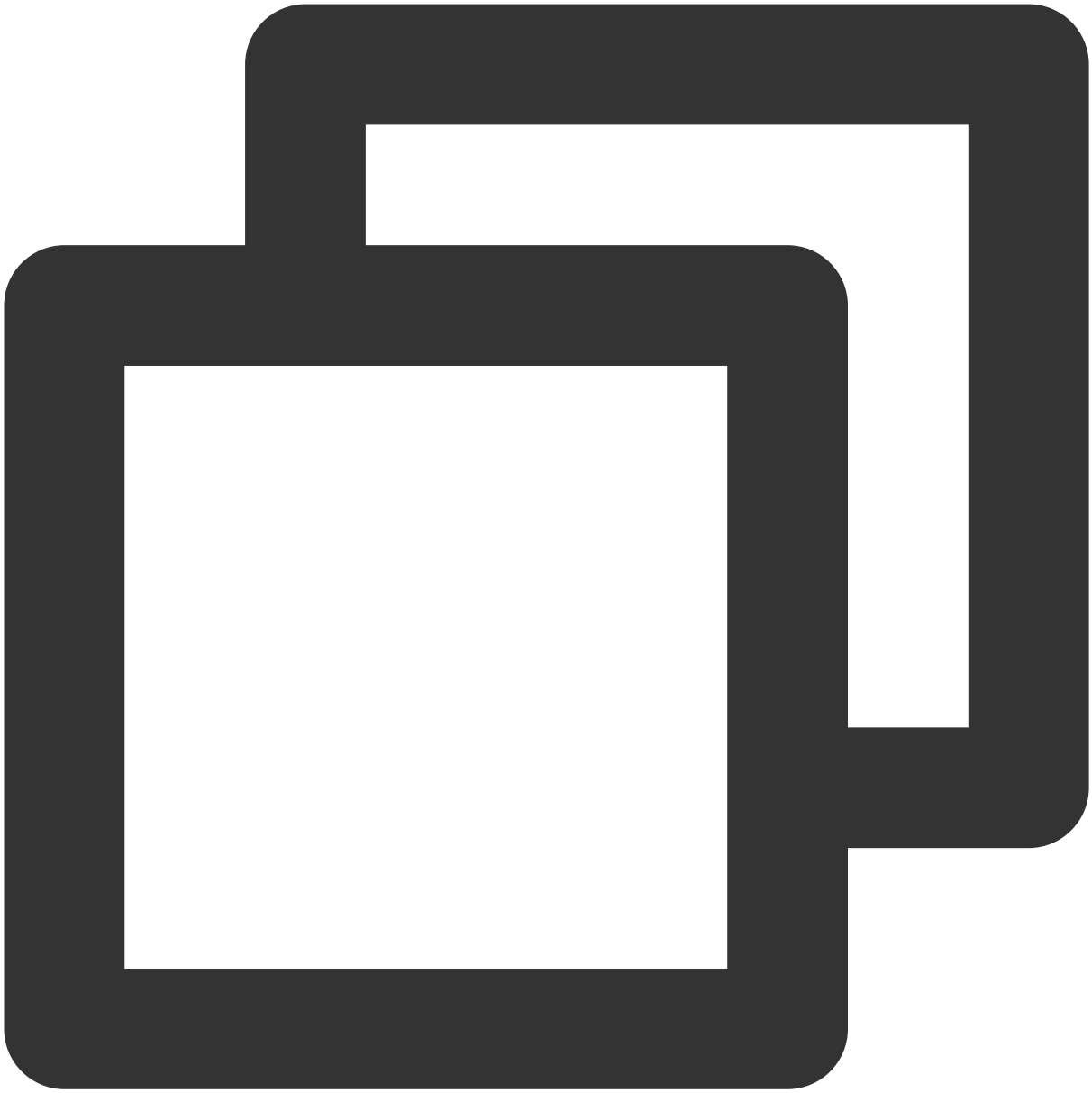
4. Click **Edit** on the right of the instance to view the metric details.

ServiceMonitor and PodMonitor

RawJob

A ServiceMonitor and a PodMonitor use the same filtering fields, and this document uses a ServiceMonitor as an example.

Sample for ServiceMonitor:

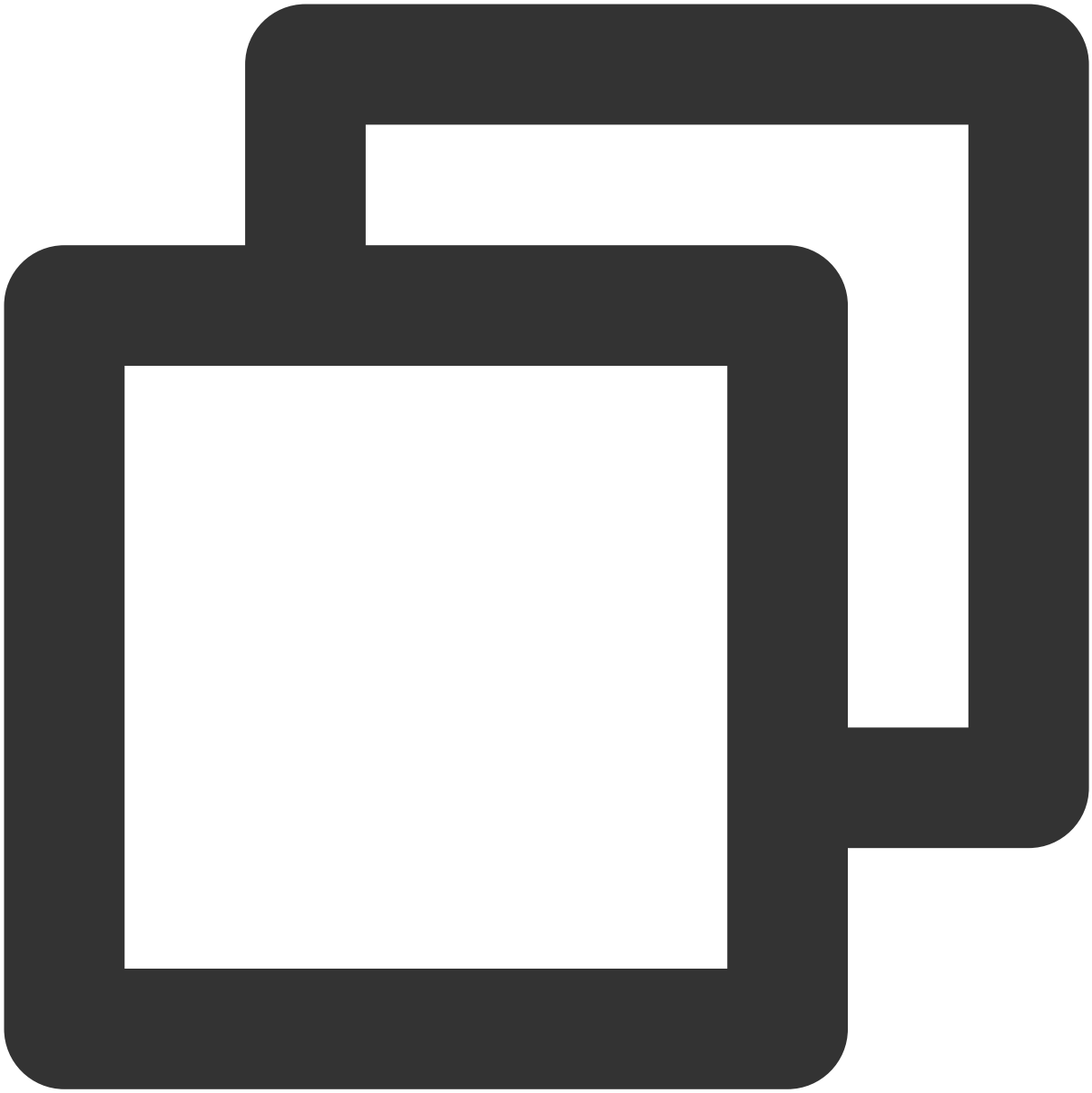


```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
  namespace: kube-system
spec:
  endpoints:
  - bearerTokenSecret:
```

```
      key: ""
    interval: 15s # It indicates the collection frequency. You can increase it to r
    port: http-metrics
    scrapeTimeout: 15s # It indicates the collection timeout period. TMP configurat
  jobLabel: app.kubernetes.io/name
  namespaceSelector: {}
  selector:
    matchLabels:
      app.kubernetes.io/name: kube-state-metrics
```

To collect `kube_node_info` and `kube_node_role` metrics, you need to add the `metricRelabelings` field to the endpoint list of the ServiceMonitor. Note that it is **`metricRelabelings`** but not `relabelings`.

Sample for adding `metricRelabelings`:

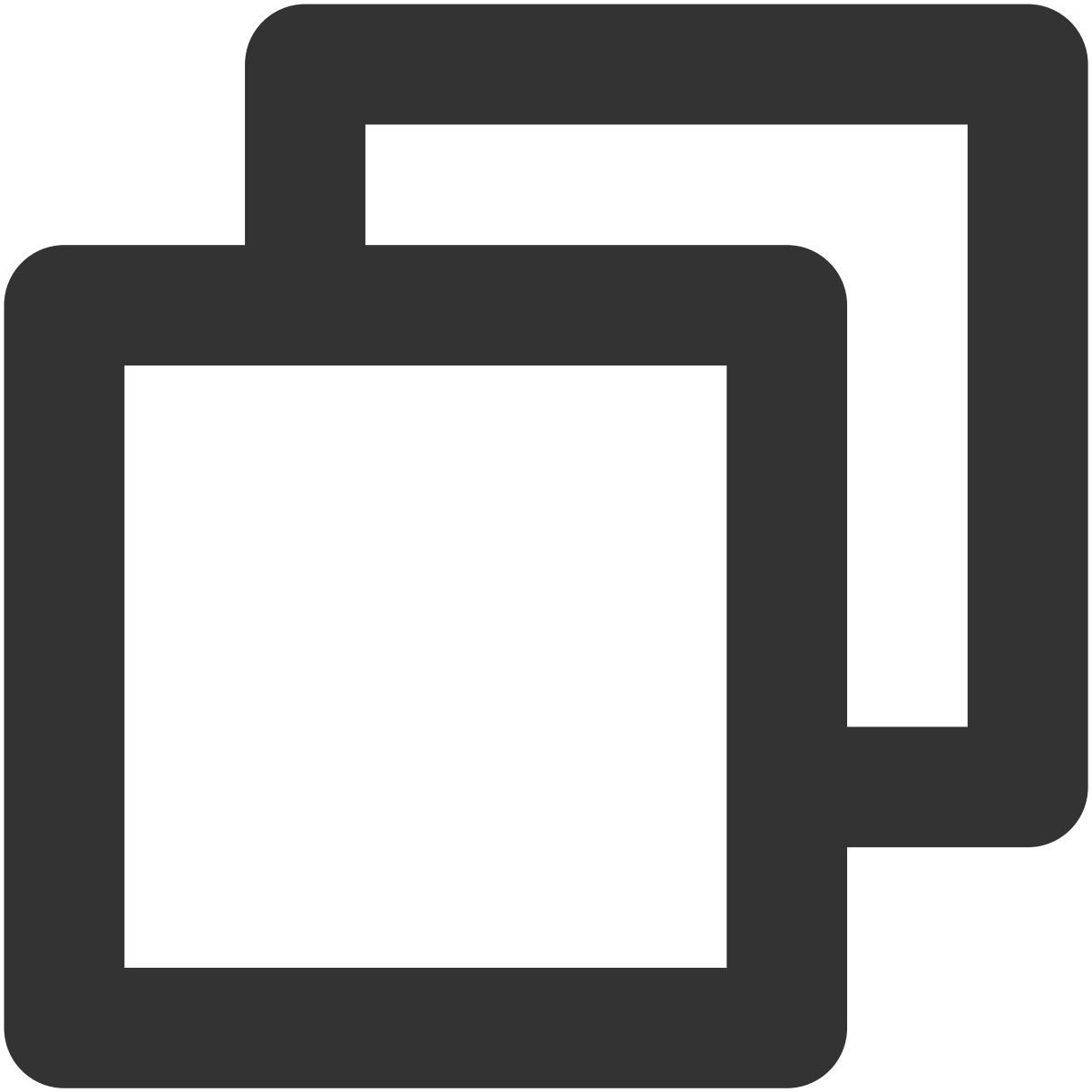


```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
  namespace: kube-system
spec:
  endpoints:
  - bearerTokenSecret:
```

```
      key: ""
    interval: 15s # It indicates the collection frequency. You can increase it to r
    port: http-metrics
    scrapeTimeout: 15s
    # The following four lines are added:
    metricRelabelings: # Each collected item is subject to the following processing
    - sourceLabels: ["__name__"] # The name of the label to be detected. `__name__`
      regex: kube_node_info|kube_node_role # Whether the above label satisfies this
      action:  keep # Keep the item if it meets the above conditions, or drop it ot
  jobLabel: app.kubernetes.io/name
  namespaceSelector: {}
  selector:
```

If Prometheus' RawJob is used, see the following method for metric filtering.

Sample job:

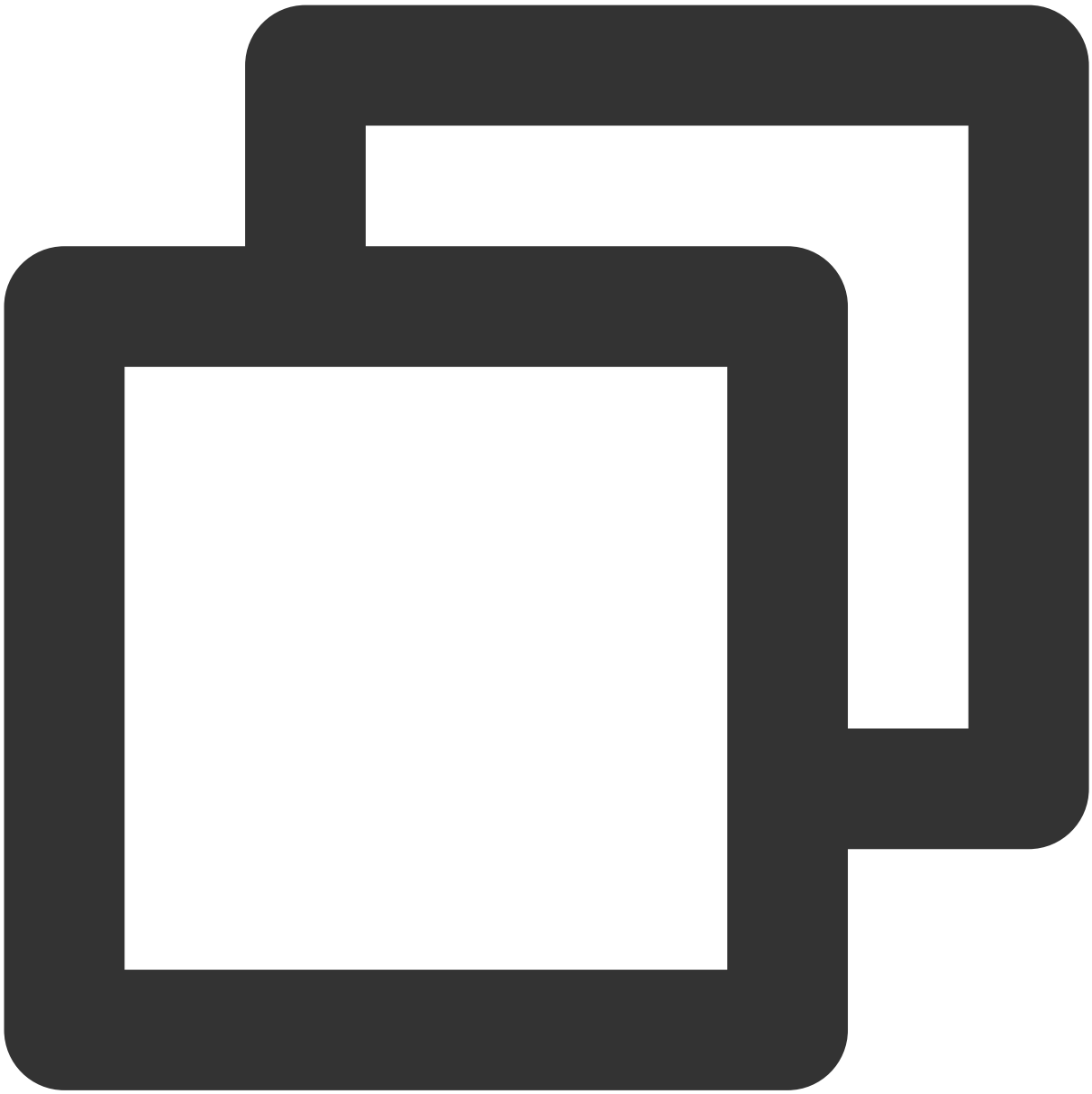


```
scrape_configs:
  - job_name: job1
    scrape_interval: 15s # It indicates the collection frequency. You can increase
    static_configs:
      - targets:
          - '1.1.1.1'
```

If you only need to collect `kube_node_info` and `kube_node_role` metrics, add the `metric_relabel_configs` field. Note that it is **`metric_relabel_configs`** but not

`relabel_configs`.

Sample for adding `metric_relabel_configs`:



```
scrape_configs:
  - job_name: job1
    scrape_interval: 15s # It indicates the collection frequency. You can increase
    static_configs:
    - targets:
      - '1.1.1.1'
    # The following four lines are added:
    metric_relabel_configs: # Each collected item is subject to the following proce
```

```
    - source_labels: ["__name__"] # The name of the label to be detected. `__name__
      regex: kube_node_info|kube_node_role # Whether the above label satisfies this
      action: keep # Keep the item if it meets the above conditions, or drop it oth
```

5. Click **OK**.

Blocking collection targets

Blocking the monitoring of the entire namespace

TMP will monitor all the ServiceMonitors and PodMonitors in a cluster by default after the cluster is associated. If you want to block the monitoring of a namespace, you can add the label of `tps-skip-monitor: "true"` as instructed in Labels and Selectors.

Blocking certain targets

TMP collects monitoring data by creating CRD resources of ServiceMonitor and PodMonitor types in your cluster. If you want to block the collection of the specified ServiceMonitor and PodMonitor resources, you can add the label of `tps-skip-monitor: "true"` to these CRD resources as instructed in Labels and Selectors.

Integration Center

Last updated：2023-02-02 17:05:22

TMP integrates commonly used programming languages, middleware, big data, and infrastructure databases. It supports quick installation and custom installation. You only need to follow the instructions to monitor the corresponding components. It also provides out-of-the-box Grafana monitoring dashboards. The integration center covers three major monitoring scenarios of basic service monitoring, application layer monitoring, and TKE cluster monitoring, making it easier for you to connect and use.

List of Supported Services

Service Type	Service	Monitoring Metric	Quick Installation	Integration Guide
Big data	Elasticsearch	Including cluster/index/node monitoring	Supported	ElasticSearch Exporter Integration
	Flink	Including cluster/job/task monitoring	Not supported	Flink Integration
Development	CVM	The extended `cvm_sd_config` can be used to configure a CVM scrape task and collect Node Exporter or custom business metrics.	Supported	CVM Node Exporter
	Go	Including GC/heap/thread/Goroutine monitoring	Not supported	Go Application Integration
	JVM	Including heap/thread/GC/CPU/file monitoring	Not supported	JVM Integration
	Spring MVC	Including HTTP API/exception/JVM monitoring	Not supported	Spring Boot Integration
Middleware	Kafka	Including broker/topic/consumer group monitoring	Supported	Kafka Exporter Integration
	Consul	Consul monitoring	Supported	Consul Exporter

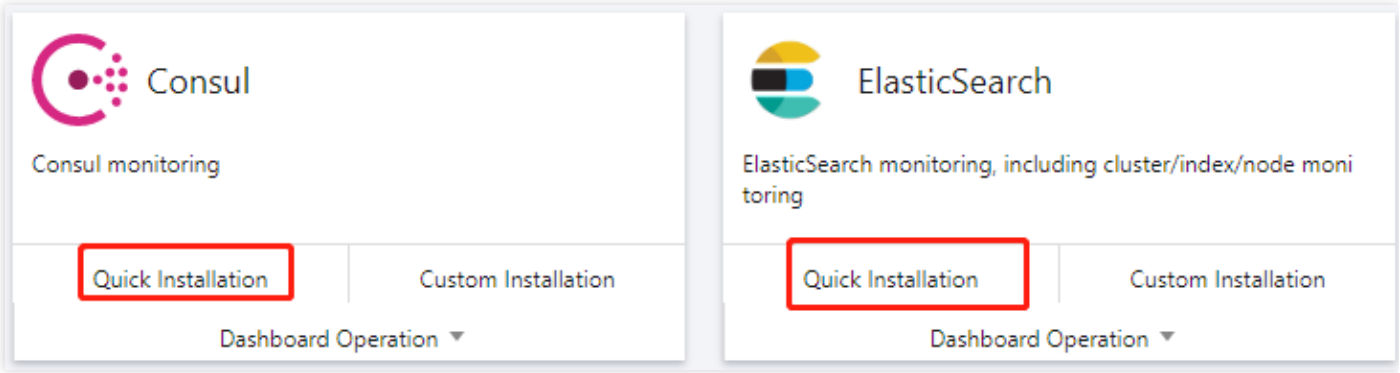
				Integration
	Etcd	Etcd monitoring	Not supported	-
	Istio	Istio monitoring	Not supported	-
Infrastructure	Kubernetes	Including API server/DNS/workload/network monitoring	Supported	Agent Management
Database	TencentDB for MongoDB	Including file count/read and write performance/network traffic monitoring	Supported	MongoDB Exporter Integration
	TencentDB for MySQL	Including network/connection count/slow query monitoring	Supported	MySQL Exporter Integration
	TencentDB for PostgreSQL	Including CPU/memory/transaction/lock/read/write monitoring	Supported	PostgreSQL Exporter Integration
	TencentDB for Redis	Including memory utilization/connection count/command execution status monitoring	Supported	Redis Exporter Integration
	TencentDB for Memcached	Memcached monitoring	Supported	Memcached Exporter Integration
Inspection	Health inspection	Blackbox can be used to regularly test the connectivity of the target service, helping you stay up to date with the service health and discover exceptions in time	Supported	Health Check
CM	Cloud Monitoring	Tencent Cloud service monitoring	Supported	-
Custom	Scrape task	The native `static_config` can be used to configure a scrape task.	Supported	Scrape Configuration Description
	CVM scrape task	The extended `cvm_sd_config` can be used to configure a CVM scrape task.	Supported	Scrape Configuration Description

Directions

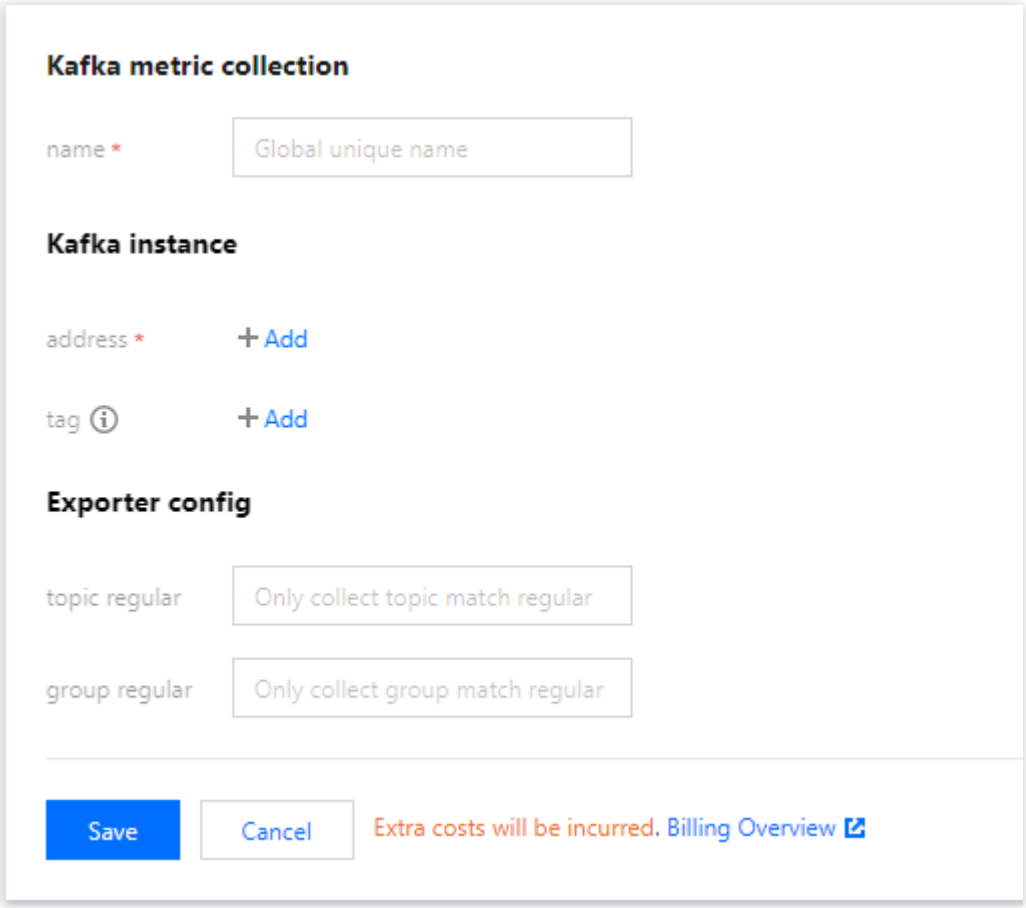
Quick installation

Some services support quick agent installation. For more information, see Integration Center > List of Supported Services.

1. Log in to the TMP console.
2. In the instance list, select the corresponding TMP instance.
3. Enter the instance details page and click **Integration Center**.
4. In the **Integration Center**, select the service that supports quick installation and click **Install** in the bottom-left corner.



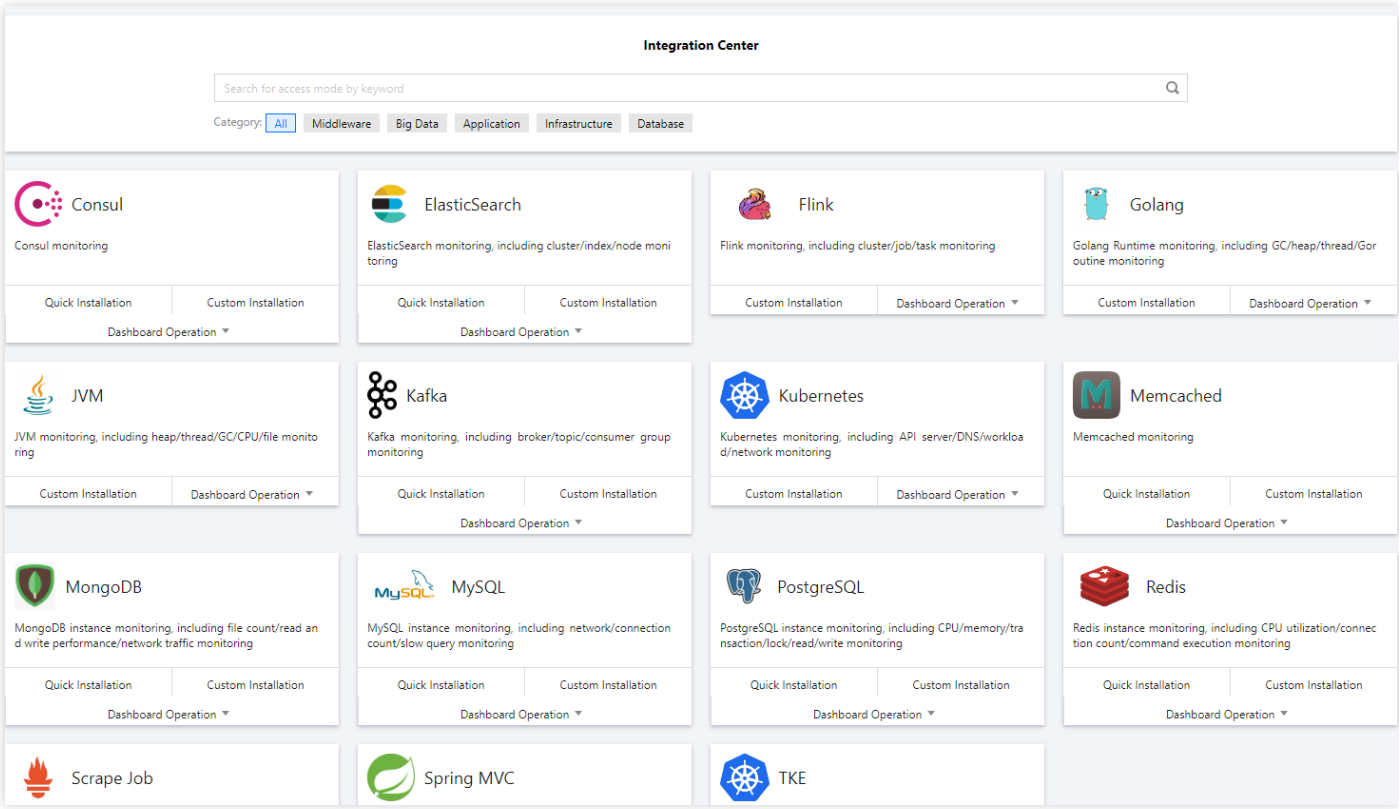
5. On the **Integration List** page, enter the metric collection name and address and click **Save**. Below is a sample for Kafka:



Custom installation

1. Log in to the TMP console.
2. In the instance list, select the corresponding TMP instance.
3. Enter the instance details page and click **Integration Center**.
4. Select the target service in the integration center. You can click **Connection Guide** to view the connection guide. After successful connection, you can monitor the corresponding service in real time. You can also click

Install/Upgrade in **Dashboard Operation** to install or upgrade the Grafana dashboard for the service.



Creating Aggregation Rules

Last updated：2022-08-26 17:44:49

Overview

This document describes how to configure aggregation rules to improve query efficiency when dealing with complex query scenarios.

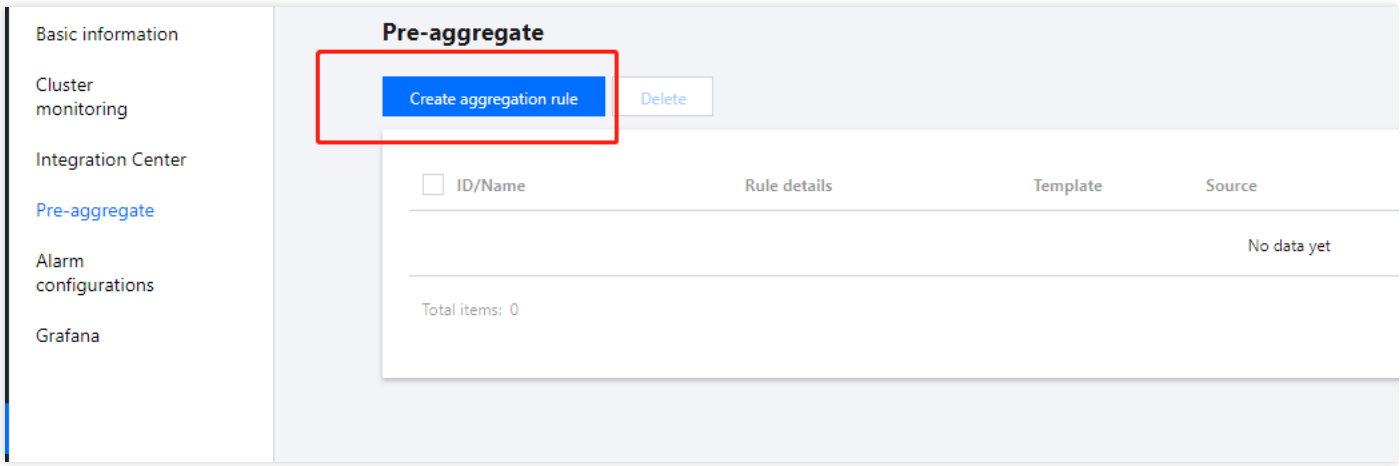
Prerequisites

Before configuring aggregation rules, you need to perform the following operations:

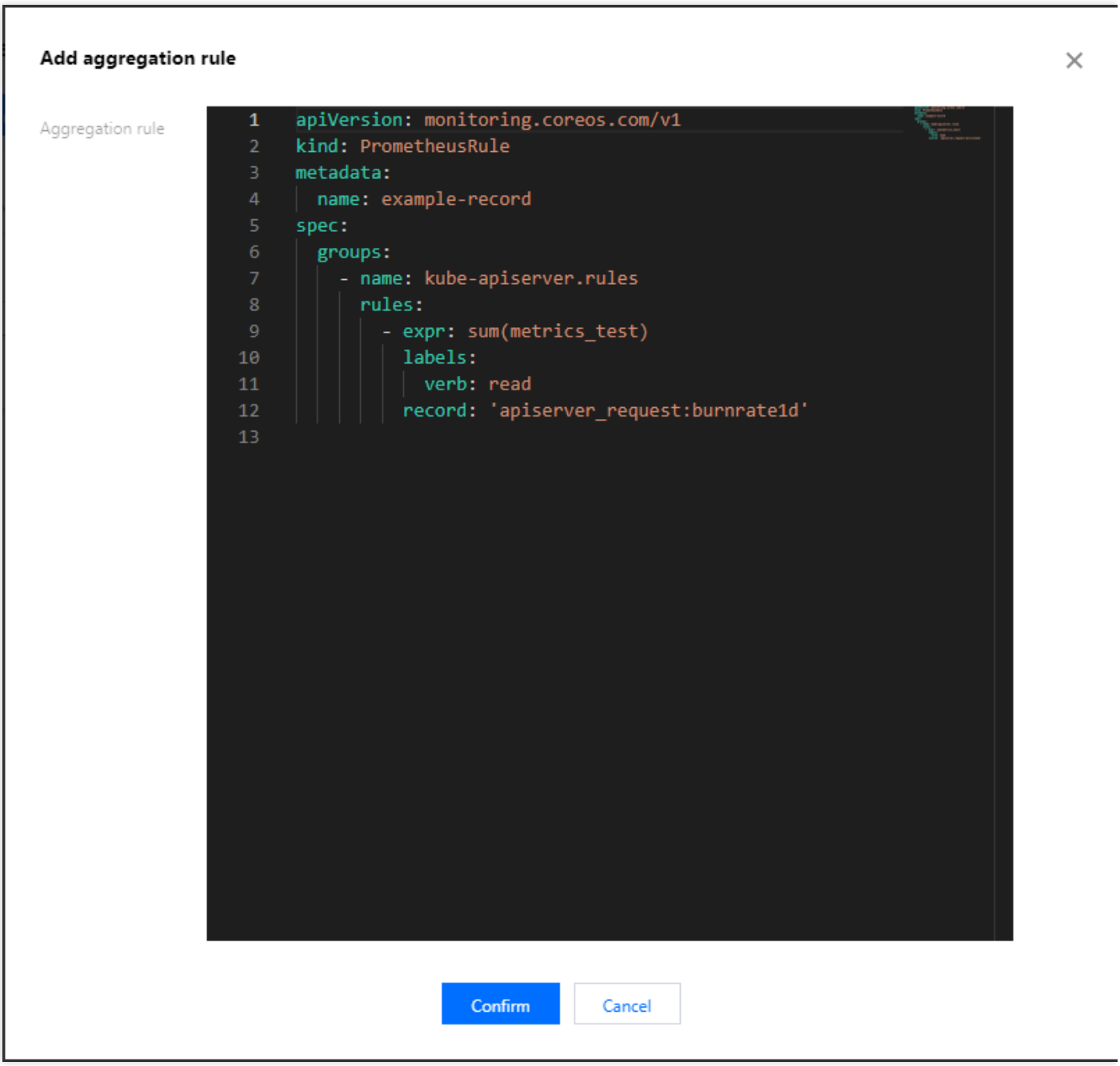
- You have logged in to the TKE console and created a self-deployed cluster.
- You have created a monitoring instance.

Directions

1. Log in to the TKE console and select **TMP** on the left sidebar.
2. On the instance list page, select an instance name that needs to create aggregation rules to go to its details page.
3. On the **Pre-aggregate** page, click **Create aggregation rule**.



4. In the **Add aggregation rule** pop-up window, edit the aggregation rule.



5. Click **OK**.

Alarm Configurations

Last updated：2022-07-13 15:36:51

Overview

This document describes how to configure alarm policies in cloud native monitoring.

Prerequisites

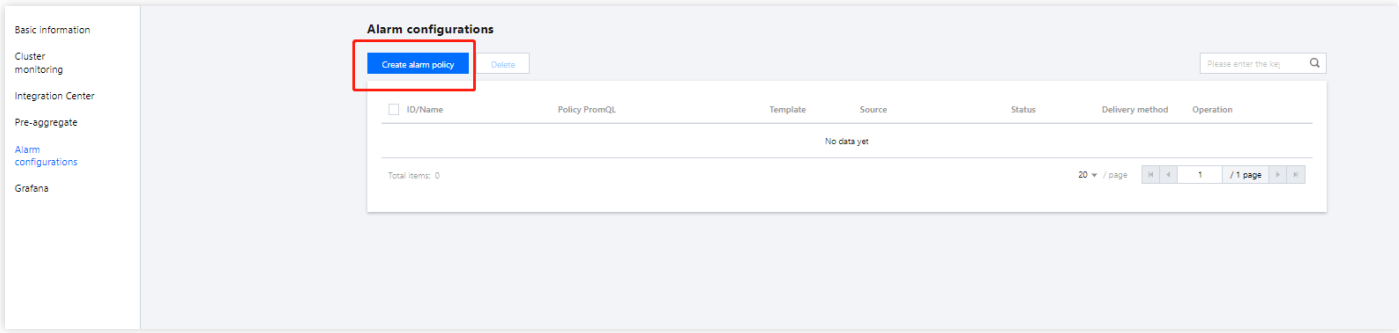
Before configuring alarm policies, you need to perform the following operations:

- Create a TMP instance.
- Associate the desired clusters with the TMP instance.
- Configure the information to be collected.

Directions

Configuring alarm policies

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. On the instance list page, select an instance name that needs to configure alarm policies to go to its details page.
3. On the **Alarm configurations** page, click **Create alarm policy**.



4. On the **Create alarm policy** page, add the details of the alarm policy.

- **Name**: Name of the alarm policy.
- **Policy template**: Select a policy template as needed.
- **Rule**:
  - **Rule name**: Name of alarm rule (up to 63 characters).

- **Rule description**: The description of the alarm rule.
- **PromQL**: The statement of the alarm rule. You can use the default value or customize it. It indicates an alert trigger condition based on a PromQL expression, which is used to calculate whether there is time series data meeting the condition.
- **Label**: Prometheus labels of each rule.
- **Annotation**: It indicates that users are allowed to define additional message for the alarm.
- **Alarm content**: The alarm notifications to be sent to recipients through email and SMS when an alarm is triggered.
- **Duration**: When the condition described in the above statement reaches the duration specified here, an alarm will be triggered.

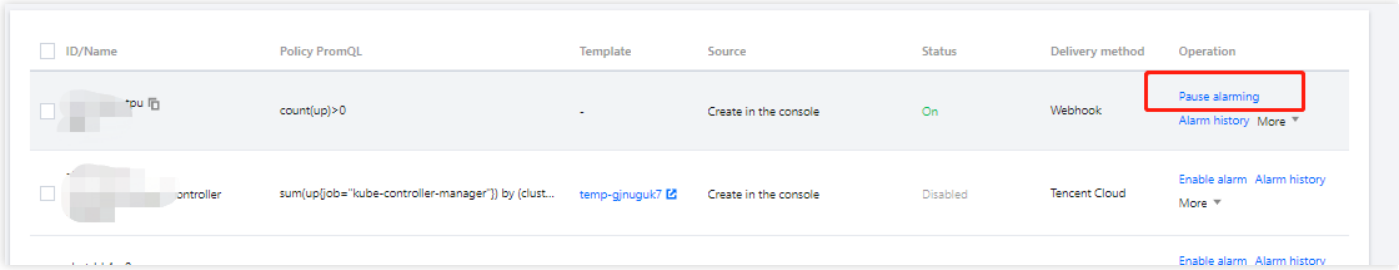
- **Convergence time**: In this specified period, if the alarm condition is met multiple times, only one notification is sent.
- **Delivery method**: The delivery channel of alarm notifications.
- **Alarm notification**: You can customize the alarm notification template, including template name, notification type, target audience and delivery method. For details, see Notification Template.
- **Save the current alarm policy as the template**: The default template name is the alarm policy name. You can edit the template name and template content in the template settings after saving.

5. Click **Complete**.

> Note：
> The alarm policy will take effect by default once created.

Pausing alarming

1. Log in to the TKE console and click **TMP** in the left sidebar.
2. On the instance list page, select an instance name that needs to pause alarming to go to its details page.
3. On the **Alarm configurations** page, click **Pause alarming** on the right side of the instance.



4. In the **Disable alarm policy** pop-up window, click **OK**.

Alarm records

Last updated：2022-08-26 17:44:49

Overview

This document describes how to query the alarm history in cloud native monitoring.

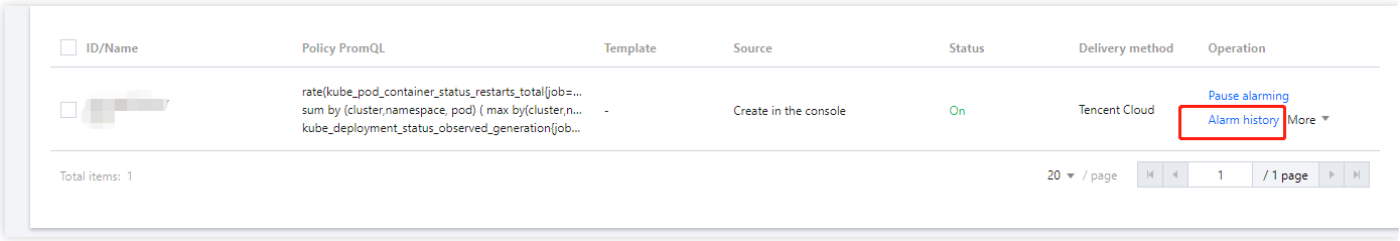
Prerequisites

Before querying alarm history, you need to perform the following operations:

- Create a TMP instance.
- Associate the desired clusters with the TMP instance.
- Configure the information to be collected.
- Configure alarm rules.

Directions

1. Log in to the TKE console and select **TMP** on the left sidebar.
2. On the instance list page, select an instance name that needs to query alarm history to go to its details page.
3. On the **Alarm Configuration** page, select **Alarm history**.



Billing Mode and Resource Usage

Last updated：2022-06-10 19:32:52

Currently, when you use the TMP service, EKS clusters and CLB resources will be created under your account. These resources and TMP are pay-as-you-go. This document describes resource usage details when you use TMP.

Resource List

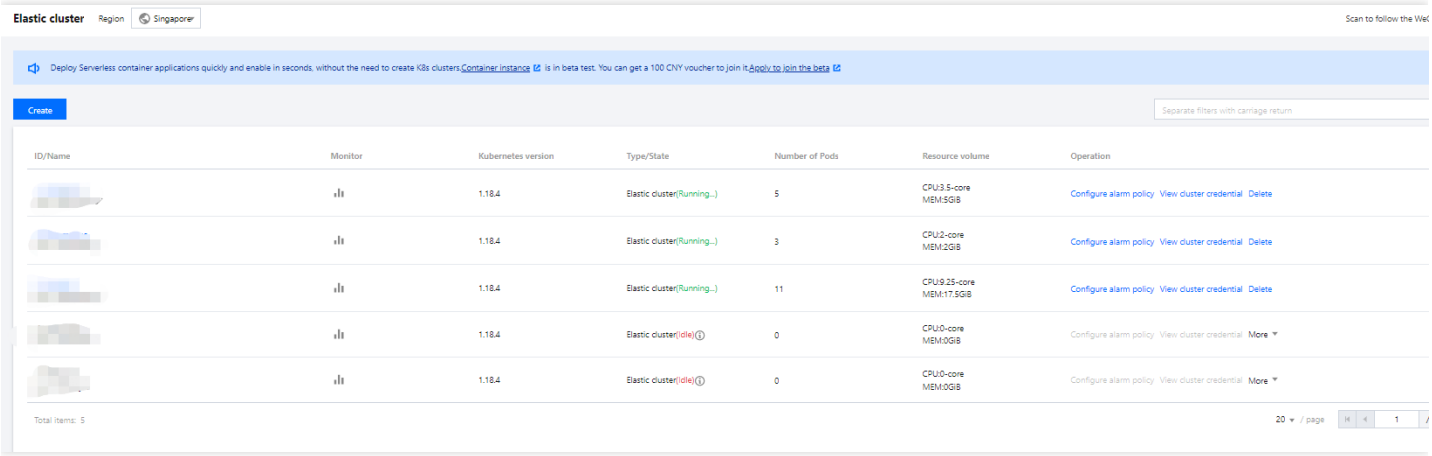
TMP instance

The capability of **Billable Metric Collection Rate** has been launched for TMP, which helps you estimate the cost of monitoring by instance, cluster, target, and metric.

1. Log in to the TKE console and select **TMP** on the left sidebar.
2. In the TMP instance list, view the **Billable Metric Collection Rate**, which indicates the collection rate of billable metrics of a TMP instance and is estimated based on your reported metric data volume and the collection frequency. This value multiplied by 86400 is the number of monitoring data points per day, and you can calculate the estimated published price as instructed in Pay-as-You-Go.
   You can also view the **Billable Metric Collection Rate** under different dimensions on various pages such as **Associate with Cluster**, **Data Collection Configuration**, and **Metric Details**.

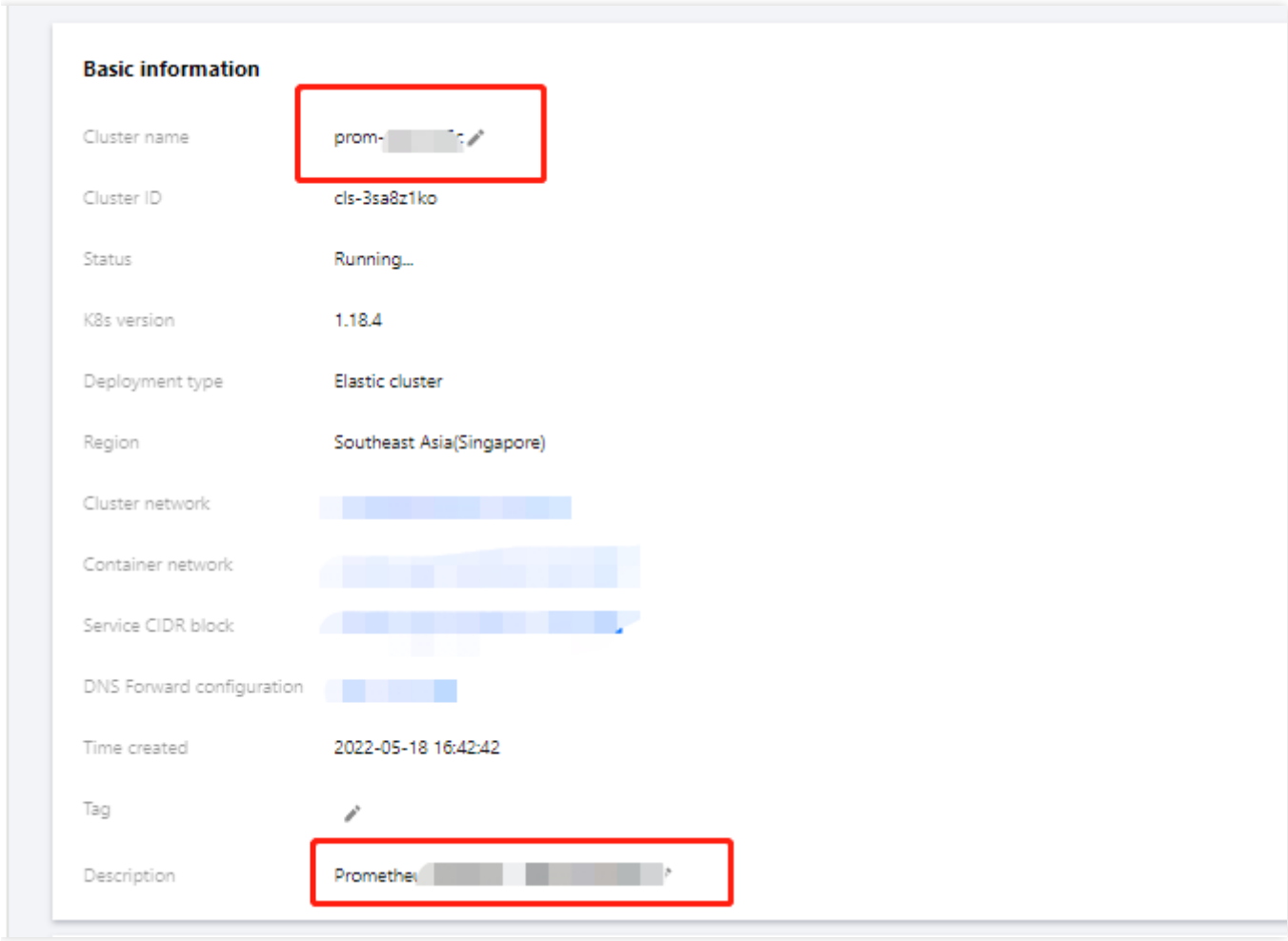
EKS cluster

After each TMP instance is created, a pay-as-you-go EKS cluster will be created under your account for data collection. View the resource information on the elastic cluster list page as shown below:



Notes

The name of the EKS cluster is the TMP **instance ID**, and the cluster description states that "For TMP use only. Do not modify or delete".



Billing

The billing mode is **pay-as-you-go**. For more information, see Product Pricing.

The EKS cluster automatically scales according to the monitoring size. The relationship between the monitoring size and the EKS cluster cost is as shown below:

Reported Instantaneous Series	Estimated EKS Resources Required	Published Price/Day
<500,000	1.25 cores, 1.6 GiB	0.35 USD
1 million	0.5 core, 1.5 GiB*2	1.46 USD
5 million	1 core, 3 GiB*3	2.93 USD
20 million	1 core, 6 GiB*5	7.98 USD
30 million	1 core, 6 GiB*8	12.77 USD

Sample EKS cluster costs are as follows:
The EKS cluster used for a newly initialized TMP instance consumes 1.25 CPU cores and 1.5 GiB memory. The estimated published price per day is 0.0319 x 24 + 0.0132 x 24 = 1.0824 USD.

CLB

When you use TMP to associate the cluster monitoring container service, a private network CLB instance will be created under your account for network connectivity between the collector and the cluster.

If you associate an edge cluster or another cluster that is not connected, a public network CLB will be created for network connectivity.

To access the Grafana service over the public network, you need to create a public network CLB instance.

These CLB resources will be charged. You can view the resource information of the created public network CLB instances in the CLB console.

This resource is pay-as-you-go. For more information, see Billing for Bill-by-IP Accounts.

Resource Termination

Currently, you cannot delete resources in their respective consoles. For example, when you terminate TMP instances in the TMP console, all relevant resources will also be terminated. Tencent Cloud does not repossess TMP instances proactively. If you no longer use TMP, you need to delete the instances promptly to avoid additional charges.

Free Metrics in Pay-as-You-Go Mode

Last updated：2022-05-16 15:39:27

The following metrics are free of charge in the pay-as-you-go mode.

Configuration File	Metric
node-exporter	node_boot_time_seconds
node-exporter	node_context_switches_total
node-exporter	node_cpu_seconds_total
node-exporter	node_disk_io_now
node-exporter	node_disk_io_time_seconds_total
node-exporter	node_disk_io_time_weighted_seconds_total
node-exporter	node_disk_read_bytes_total
node-exporter	node_disk_read_time_seconds_total
node-exporter	node_disk_reads_completed_total
node-exporter	node_disk_write_time_seconds_total
node-exporter	node_disk_writes_completed_total
node-exporter	node_disk_written_bytes_total
node-exporter	node_filefd_allocated
node-exporter	node_filesystem_avail_bytes
node-exporter	node_filesystem_free_bytes
node-exporter	node_filesystem_size_bytes
node-exporter	node_load1
node-exporter	node_load15
node-exporter	node_load5
node-exporter	node_memory_Buffers_bytes

Configuration File	Metric
node-exporter	node_memory_Cached_bytes
node-exporter	node_memory_MemAvailable_bytes
node-exporter	node_memory_MemFree_bytes
node-exporter	node_memory_MemTotal_bytes
node-exporter	node_netstat_TcpExt_ListenDrops
node-exporter	node_netstat_Tcp_ActiveOpens
node-exporter	node_netstat_Tcp_CurrEstab
node-exporter	node_netstat_Tcp_InSegs
node-exporter	node_netstat_Tcp_OutSegs
node-exporter	node_netstat_Tcp_PassiveOpens
node-exporter	node_network_receive_bytes_total
node-exporter	node_network_transmit_bytes_total
node-exporter	node_sockstat_TCP_alloc
node-exporter	node_sockstat_TCP_inuse
node-exporter	node_sockstat_TCP_tw
node-exporter	node_sockstat_UDP_inuse
node-exporter	node_sockstat_sockets_used
node-exporter	node_uname_info
cadvisor	container_cpu_usage_seconds_total
cadvisor	container_fs_limit_bytes
cadvisor	container_fs_reads_bytes_total
cadvisor	container_fs_usage_bytes
cadvisor	container_fs_writes_bytes_total
cadvisor	container_memory_working_set_bytes

Configuration File	Metric
cadvisor	container_network_receive_bytes_total
cadvisor	container_network_receive_packets_dropped_total
cadvisor	container_network_receive_packets_total
cadvisor	container_network_transmit_bytes_total
cadvisor	container_network_transmit_packets_dropped_total
cadvisor	container_network_transmit_packets_total
cadvisor	machine_cpu_cores
cadvisor	machine_memory_bytes
kubelet	kubelet_cgroup_manager_duration_seconds_count
kubelet	kubelet_node_config_error
kubelet	kubelet_node_name
kubelet	kubelet_pleg_relist_duration_seconds_bucket
kubelet	kubelet_pleg_relist_duration_seconds_count
kubelet	kubelet_pleg_relist_interval_seconds_bucket
kubelet	kubelet_pod_start_duration_seconds_count
kubelet	kubelet_pod_worker_duration_seconds_count
kubelet	kubelet_running_containers
kubelet	kubelet_running_pods
kubelet	kubelet_runtime_operations_duration_seconds_bucket
kubelet	kubelet_runtime_operations_errors_total
kubelet	kubelet_runtime_operations_total
kubelet	process_cpu_seconds_total
kubelet	process_resident_memory_bytes
kubelet	rest_client_request_duration_seconds_bucket

Configuration File	Metric
kubelet	rest_client_requests_total
kubelet	storage_operation_duration_seconds_bucket
kubelet	storage_operation_duration_seconds_count
kubelet	storage_operation_errors_total
kubelet	volume_manager_total_volumes
kube-state-metrics	kube_job_status_succeeded
kube-state-metrics	kube_job_status_failed
kube-state-metrics	kube_job_status_active
kube-state-metrics	kube_node_status_capacity_cpu_cores
kube-state-metrics	kube_node_status_capacity_memory_bytes
kube-state-metrics	kube_node_status_allocatable_cpu_cores
kube-state-metrics	kube_node_status_allocatable_memory_bytes
kube-state-metrics	kube_pod_info
kube-state-metrics	kube_pod_owner
kube-state-metrics	kube_pod_status_phase
kube-state-metrics	kube_pod_container_status_waiting
kube-state-metrics	kube_pod_container_status_running
kube-state-metrics	kube_pod_container_status_terminated
kube-state-metrics	kube_pod_container_status_restarts_total
kube-state-metrics	kube_pod_container_resource_requests_cpu_cores
kube-state-metrics	kube_pod_container_resource_requests_memory_bytes
kube-state-metrics	kube_pod_container_resource_limits_cpu_cores
kube-state-metrics	kube_pod_container_resource_limits_memory_bytes
kube-state-metrics	kube_replicaset_owner

Configuration File	Metric
kube-state-metrics	kube_statefulset_status_replicas
kube-controller-manager	rest_client_request_duration_seconds_bucket
kube-controller-manager	rest_client_requests_total
kube-controller-manager	workqueue_adds_total
kube-controller-manager	workqueue_depth
kube-controller-manager	workqueue_queue_duration_seconds_bucket
kube-apiserver	apiserver_current_inflight_requests
kube-apiserver	apiserver_current_inqueue_requests
kube-apiserver	apiserver_init_events_total
kube-apiserver	apiserver_longrunning_gauge
kube-apiserver	apiserver_registered_watchers
kube-apiserver	apiserver_request_duration_seconds_bucket
kube-apiserver	apiserver_request_duration_seconds_sum
kube-apiserver	apiserver_request_duration_seconds_count
kube-apiserver	apiserver_request_filter_duration_seconds_bucket
kube-apiserver	apiserver_request_filter_duration_seconds_sum
kube-apiserver	apiserver_request_filter_duration_seconds_count
kube-apiserver	apiserver_request_total
kube-apiserver	apiserver_requested_deprecated_apis
kube-apiserver	apiserver_response_sizes_bucket
kube-apiserver	apiserver_response_sizes_sum
kube-apiserver	apiserver_response_sizes_count
kube-apiserver	apiserver_selfrequest_total
kube-apiserver	apiserver_tls_handshake_errors_total

Configuration File	Metric
kube-apiserver	apiserver_watch_events_sizes
kube-apiserver	apiserver_watch_events_sizes_bucket
kube-apiserver	apiserver_watch_events_sizes_sum
kube-apiserver	apiserver_watch_events_sizes_count
kube-apiserver	apiserver_watch_events_total

Terminating Instance

Last updated：2023-07-27 09:52:06

Overview

If you no longer need to use TMP to monitor clusters, you can delete all monitoring instances in the TMP console. The system will automatically uninstall the monitoring add-on and terminate relevant resources.

Directions

1. Log in to the **TKE console** and select **TMP** on the left sidebar.
2. On the instance list page, find the target instance and click **Terminate/Return** on the right.
3. In the **Terminate/Return** pop-up window, confirm the instance information and click **OK**.

Note：

- After an instance is deleted, the TMP console will no longer display its information.
- After an instance is deleted, its resources (such as monitoring add-on) and configurations will be deleted, the associated cluster will be automatically disassociated and no longer be monitored, and the associated TKE serverless cluster will be deleted.
- Note that the instance data cannot be recovered after the deletion. Proceed with caution.