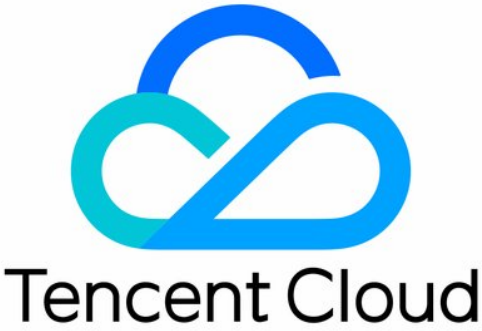
Tencent Kubernetes Engine

TKE Edge Cluster Guide

Product Documentation



Contents

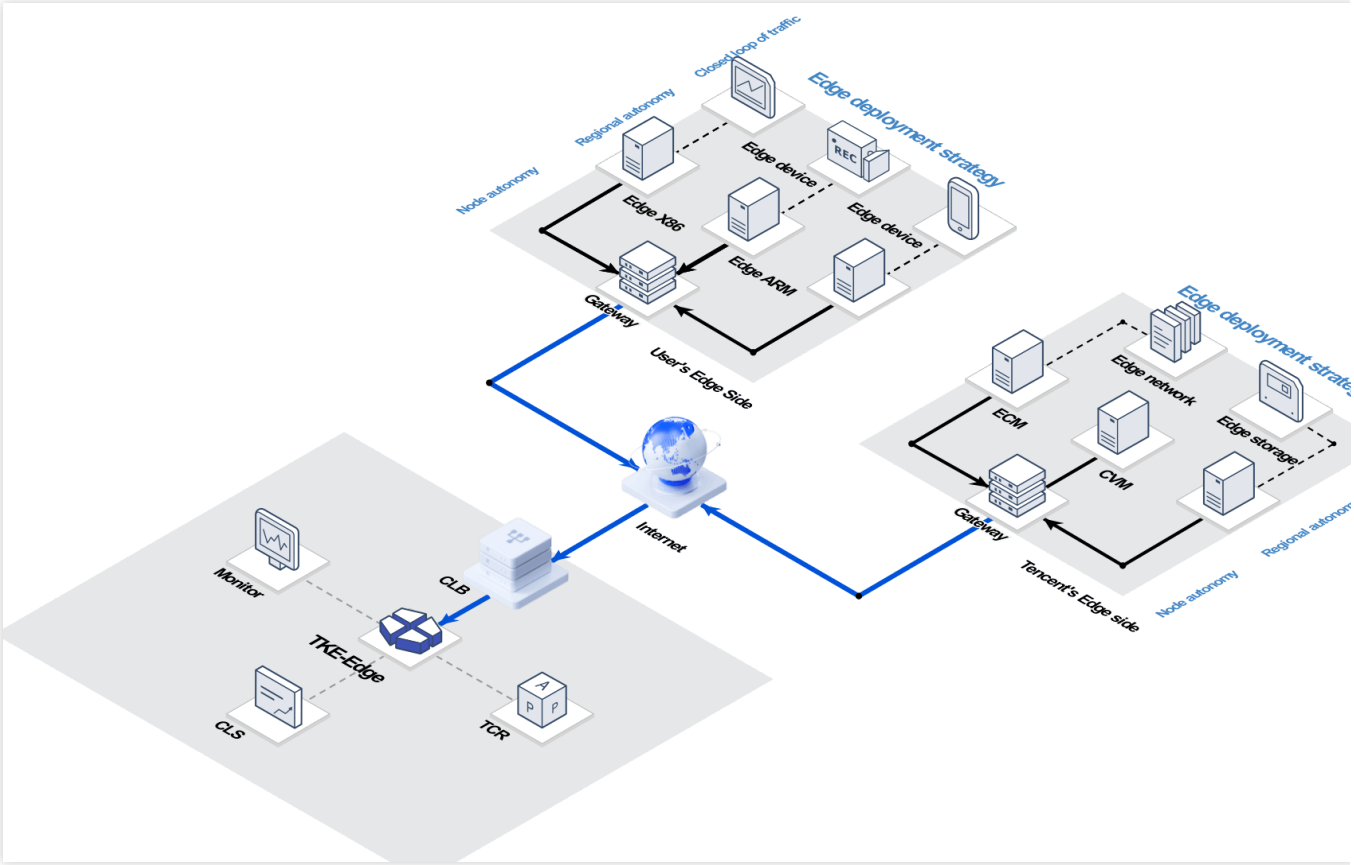
TKE Edge Cluster Guide
Overview

Last updated：2023-06-01 11:22:54

Introduction

Tencent Cloud Engine for Edge (TKE-Edge) is a container platform launched by Tencent Cloud that manages edge resources through a public cloud-hosted control plane, providing cloud-native capability support that integrates cloud and edge computing scenarios. TKE Edge Container is fully compatible with native Kubernetes, supports managing nodes in multiple data centers within the same cluster, allows one-click deployment of applications to all edge nodes, and features edge autonomy and distributed health check capabilities.

Architecture



Capabilities of TKE-Edge

Native support

TKE-Edge is built upon the open-source SuperEdge project, closely following the community. It supports the latest Kubernetes versions and native Kubernetes cluster management methods. It is easy to use out of the box and requires only a minimal learning curve.

High availability of the control plane

TKE-Edge is built upon the core Tencent Cloud TKE product, which hosts the Master control plane. Worker nodes located anywhere can connect to the control plane with a single click to start providing services, significantly improving the availability of the cluster.

Cloud capabilities brought to the edge

TKE-Edge seamlessly reuses most of the supporting capabilities of TKE products, such as monitoring, logging, and app stores, quickly bringing standard public cloud capabilities to the edge side.

Heterogeneous resource support

The connected edge nodes support various types of heterogeneous resources, including Tencent Cloud provided CVM/ECM machines, user-side owned IDC data center servers, PCs, IoT devices, etc. It also supports x86, ARM64, and ARM architecture, allowing for mixed deployment of applications on different types of resources.

Multi-region deployment capabilities

The service supports the division of edge nodes into different node pools according to required features (such as region) and allows for batch application scheduling and deployment across multiple node pools. It also provides closed-loop traffic access capabilities within the node pool.

Secure and reliable

TKE-Edge supports separated internal and external network certificates and minimized node permission control to avoid cluster access control permission leakage. Cloud-edge communication uses TLS encryption to prevent system management data leakage or tampering.

System disaster recovery

In addition to the high availability of the control plane, the TKE-Edge provides reliable edge autonomy capabilities for weak network scenarios in the cloud and features distributed cluster health check capabilities to accurately determine pod migration timing.

Cloud-edge tunnel

Leveraging Tencent's years of tunnel technology accumulation, even if edge devices do not have a public network address, the edge container service can support administrators directly logging into containers running on edge nodes from the cloud.

Pricing

TKE-Edge's International site is temporarily free of charge.
Computing nodes are provided by users and therefore do not generate any additional charges.

Use Cases

Edge computing

TKE-Edge targets demands from customers in industries such as Industrial Internet, Edge IoT, and Edge AI. It supports customers in quickly creating cloud-based control planes, enabling unified management and control of various types of edge devices. Users can manage the application lifecycle in the cloud.

Hybrid Cloud

TKE-Edge addresses the demands of large customers in hybrid cloud scenarios. It enables the rapid deployment of standard K8s clusters and cloud-standard services in user IDCs while seamlessly integrating with cloud-based SaaS services. This lowers deployment and operation costs and provides high availability support at the level of public cloud services.

Additional Services

For more information on how to call TencentCloud APIs to access Tencent Cloud products and services, see TencentCloud APIs.

Edge Cluster Management
Creating a Cluster

Last updated：2023-07-14 16:30:12

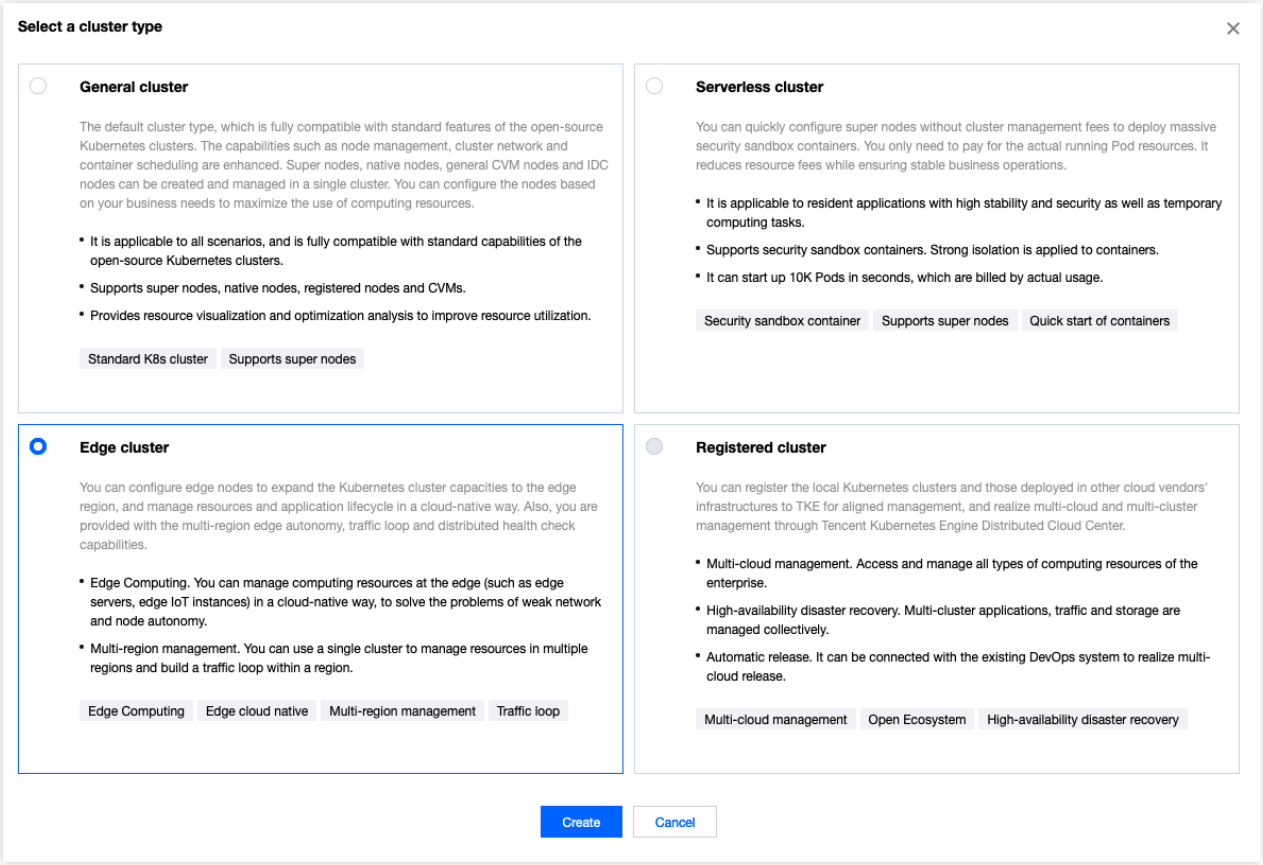
This document describes how to create an edge cluster to use TKE Edge from the Tencent Cloud TKE console.

Container Network Explanation

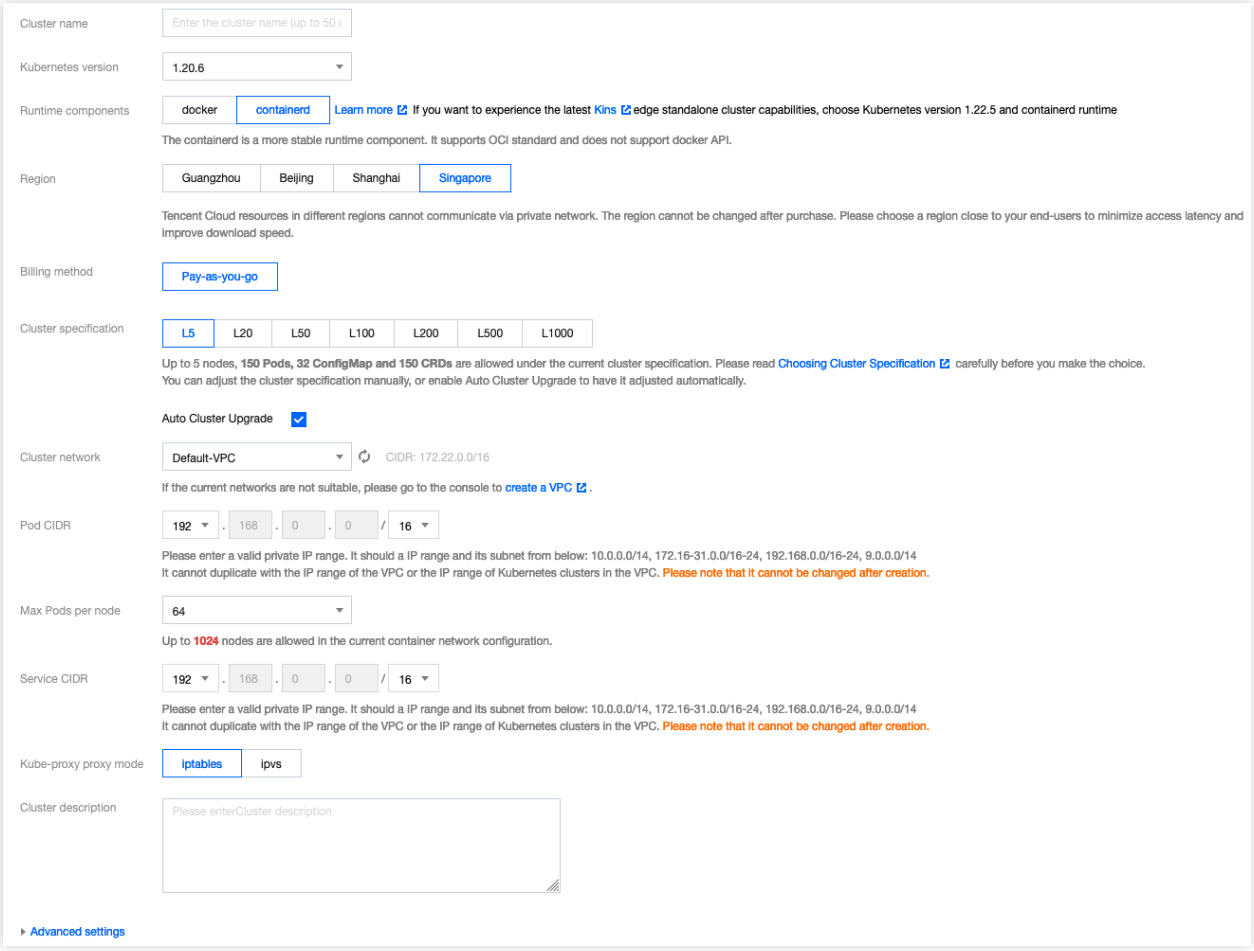
TKE Edge Container Service will utilize the node-side network to build an overlay network. Please ensure that the cluster network (control plane in VPC network) and container network (Pod CIDR/Service CIDR) do not conflict with the internal network of the edge nodes.

Operation Steps

1. Log in to the Tencent Cloud TKE console and click **Edge Clusters** in the left sidebar.
2. On the Cluster Management page, click New.
3. Select Edge Cluster, then click 'Create'.



4. On the Create Edge Cluster page, create an edge cluster according to the following information, as shown in the figure below:



Cluster name: indicates the name of the edge cluster to be created, with a maximum length of 60 characters.

Kubernetes version: Kubernetes version 1.16 is currently supported. This version will be updated when a newer Kubernetes version is published by the Kubernetes community.

Runtime Components: Supports Docker runtime and Containerd runtime. For more information, please refer to How to Choose Between Containerd and Docker.

Region: select the region that is closest to your location to minimize access latency and improve the download speed.

Billing Method: The current Edge Cluster supports pay-as-you-go billing.

Cluster Specifications: You can choose different cluster specifications according to your needs. Different specifications have restrictions on the number of access nodes, as well as the number of pods, configmaps, and CRDs. For more details, please refer to the Purchase Cluster Quota Limitations.

Auto Upgrade Cluster Specifications: If you enable this feature, when the cluster resources exceed the maximum limit of the current specification, the cluster specification will be automatically upgraded to a higher level and billed accordingly. Later on, you can reclaim resources, but the platform will not automatically downgrade. You can manually adjust the specifications to a more appropriate lower level. If this feature is not enabled, when the cluster resources exceed the maximum limit of the current specification, the newly added nodes will be in a "locked" state, not allowing application scheduling to that node. You will have to manually upgrade the cluster specification for the nodes to automatically return to normal.

Cluster network: assign a network for the cluster according to the internal network management of edge servers.
Pod CIDR: you need to assign a container network for the cluster according to the internal network management of edge servers. Therefore, plan the cluster size in advance to assign an IP range with sufficient IP addresses for the container network. **The pod CIDR block cannot overlap with IP ranges used by a VPC instance and existing Kubernetes clusters in the VPC instance. In addition, it cannot be modified once created.**
Service CIDR: you need to assign a service network for the cluster according to the internal network management of edge servers. Therefore, plan the cluster size in advance to assign an IP range with sufficient IP addresses for the service network. **The service CIDR block cannot overlap with IP ranges used by a VPC instance and existing Kubernetes clusters in the VPC instance. In addition, it cannot be modified once created.**
Kube-proxy Mode: You can choose between iptables mode and ipvs mode.
Cluster description: indicates information about the cluster, which is displayed on the **Cluster Information** page.
5. Click **Done** to finish creating the Master components of the cluster. You can check the progress of cluster creation on the "Edge Clusters" page.

Node Management

Note：

After the cluster is created, you need to enable the cluster's internal/external access in order to provide services to the outside world properly. For more details, please refer to Enabling Internal and External Access for the Cluster.

Enable Internet/Intranet Access

Last updated：2023-06-01 11:28:46

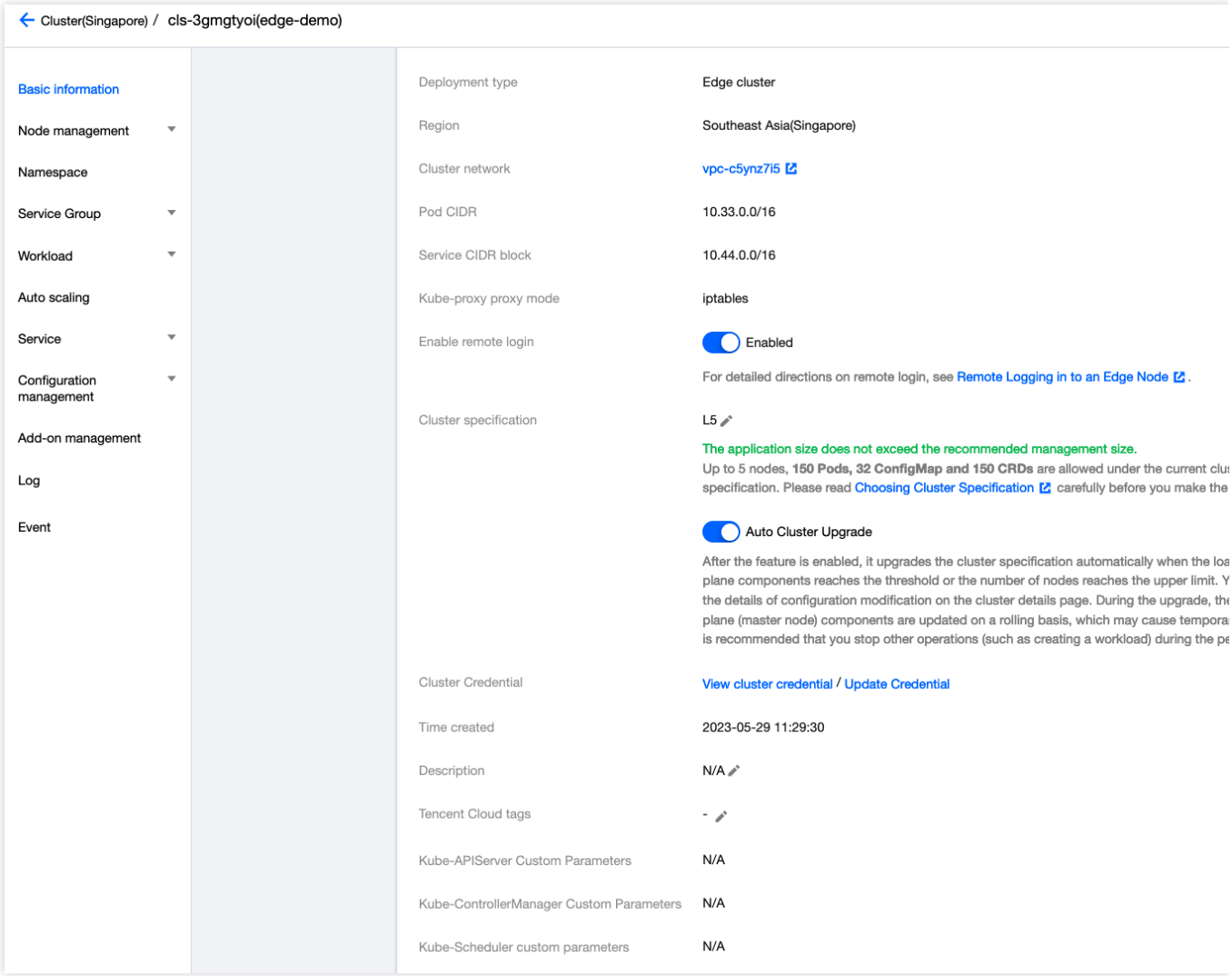
Enable Add Node Capability

If you need to enable the Add Node capability, you first need to open the cluster APIServer service for external access, which includes opening internal network access (within the VPC) and opening external network access (to the Internet) two different types.

Enable External Network Access

External network access means that the user's nodes are outside the Tencent Cloud public cloud system, but can access Tencent Cloud services through various types of Internet access. This situation applies to most edge nodes.

1. Log in to the Tencent Kubernetes Engine, and select the Cluster option in the left-side navigation menu.
2. In the cluster list, click on the Edge Cluster ID to enter the cluster details page.
3. In the cluster "Basic Information", locate the "Cluster APIServer Information" and enable external network access.

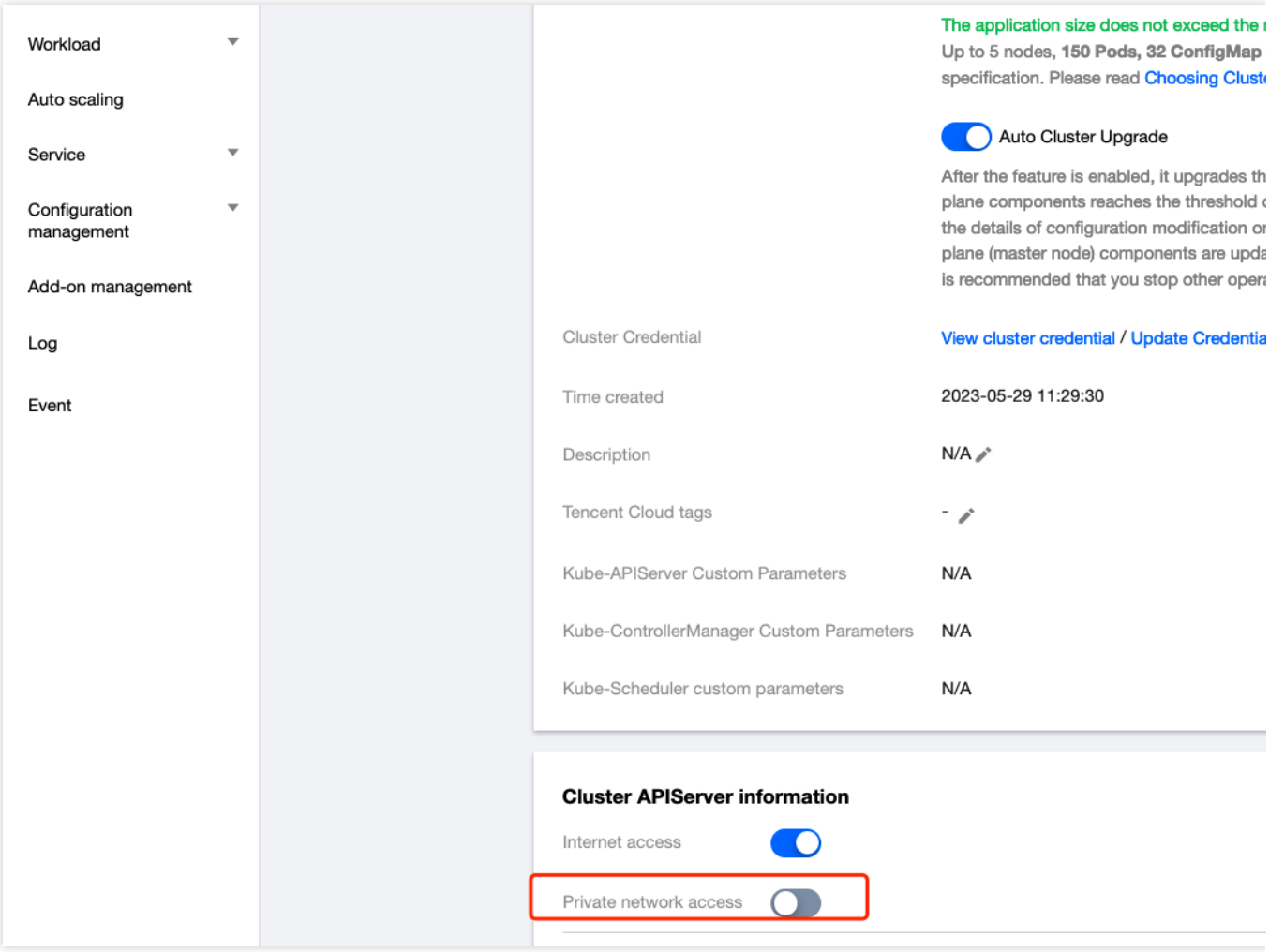


After successfully enabling it, users can add ECM, CVM, and third-party nodes (via script) through the Internet link.

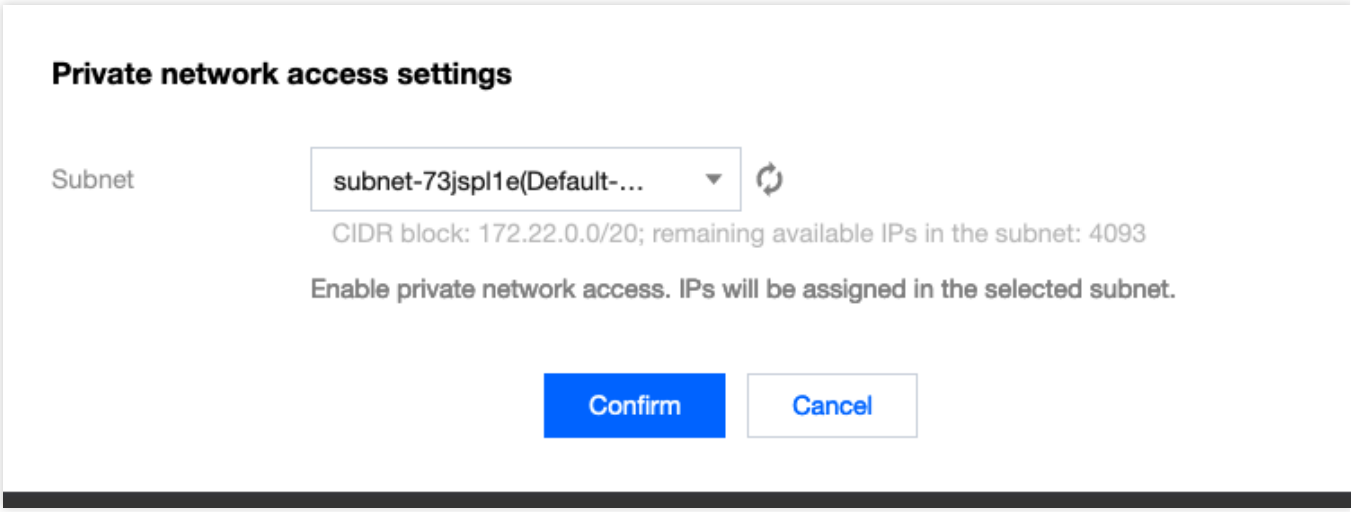
Enable Internal Network Access

Internal network access is provided to facilitate users to add CVM machines under the same VPC to the cluster, and register them to the edge cluster via the VPC intranet, thereby saving public network traffic. Edge Container provides the capability to add nodes through the intranet, and you can use it according to the following steps.

1. Log in to the Tencent Kubernetes Engine, and select the Cluster option in the left-side navigation menu.
2. In the cluster list, click on the Edge Cluster ID to enter the cluster details page.
3. In the cluster "Basic Information", locate the "Cluster APIServer Information" and enable internal network access.



4. In the "Internal Network Access Settings", select a subnet under the VPC where the cluster is located. You need to select a specific subnet (the subnet here refers to the subnet where the subsequently added CVM is located).



5. After clicking Confirm, you can use the script to add CVM nodes under this VPC through the internal network.

Note

After enabling internal network access, you can only add CVM nodes within the VPC through the Add Nodes via Script method on the node page. You cannot use the capabilities to create ECM nodes and create CVM nodes.

Next Steps

After enabling internal/external network access, you can add nodes to the cluster using the following methods:

Create ECM node

Create CVM node

Add third-party nodes via script

Connecting to a Cluster

Last updated：2023-06-01 11:22:54

Scenario

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

Prerequisites

The cURL software program has been installed.

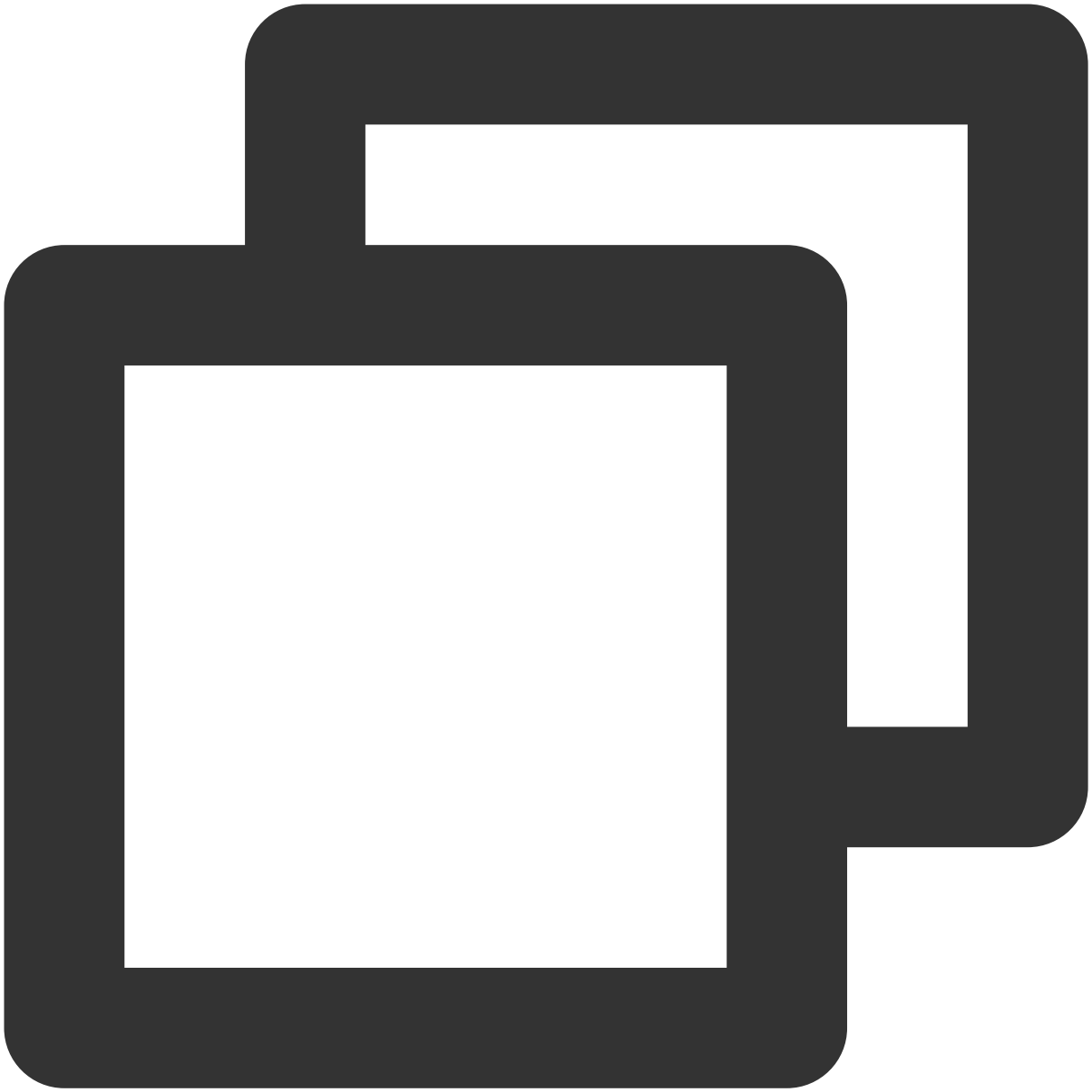
Select the appropriate way to obtain kubectl based on the operating system:

Note：

Replace `v1.18.4` in the command with the kubectl version required by your business.

MacOS X

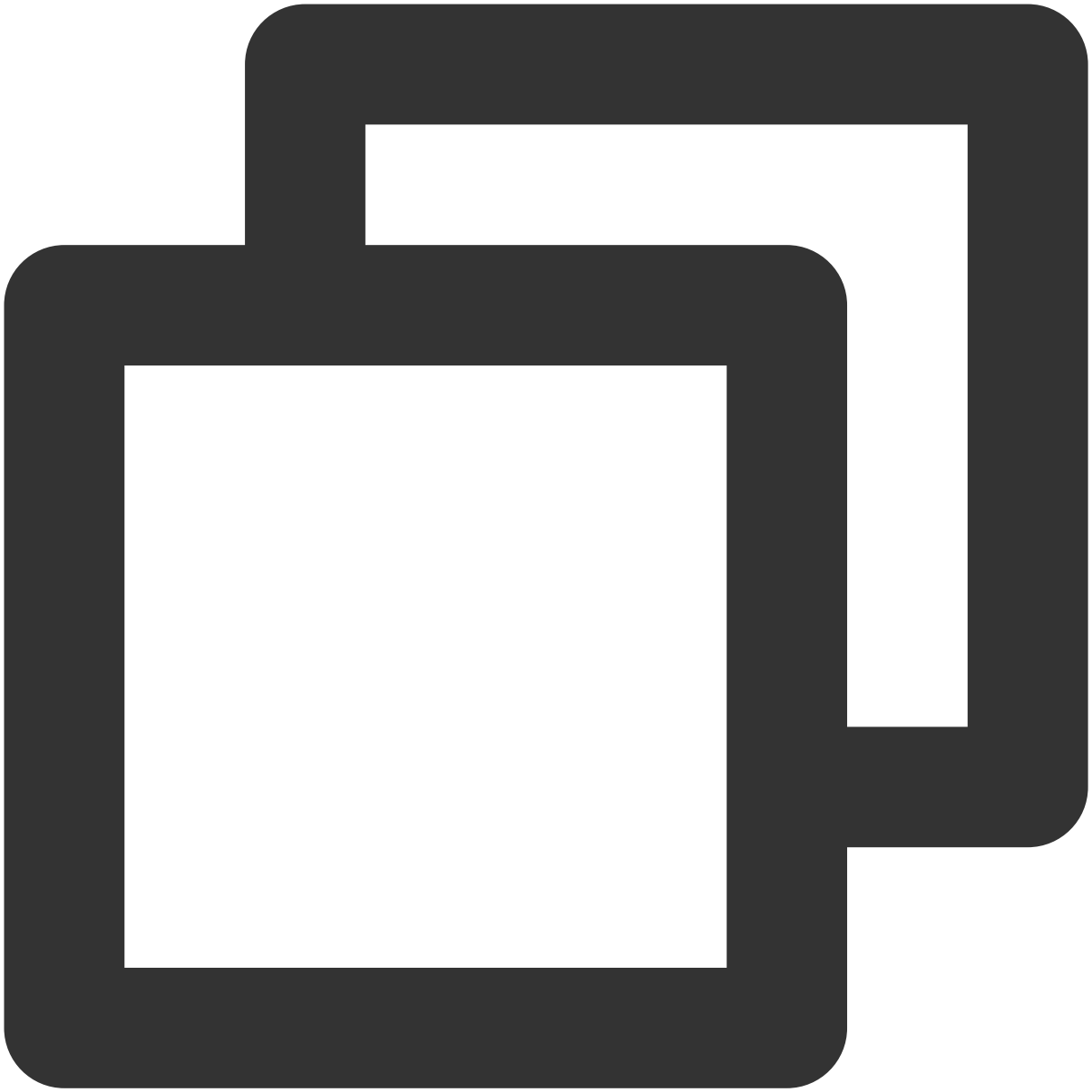
Run the following command to obtain kubectl:



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

Linux

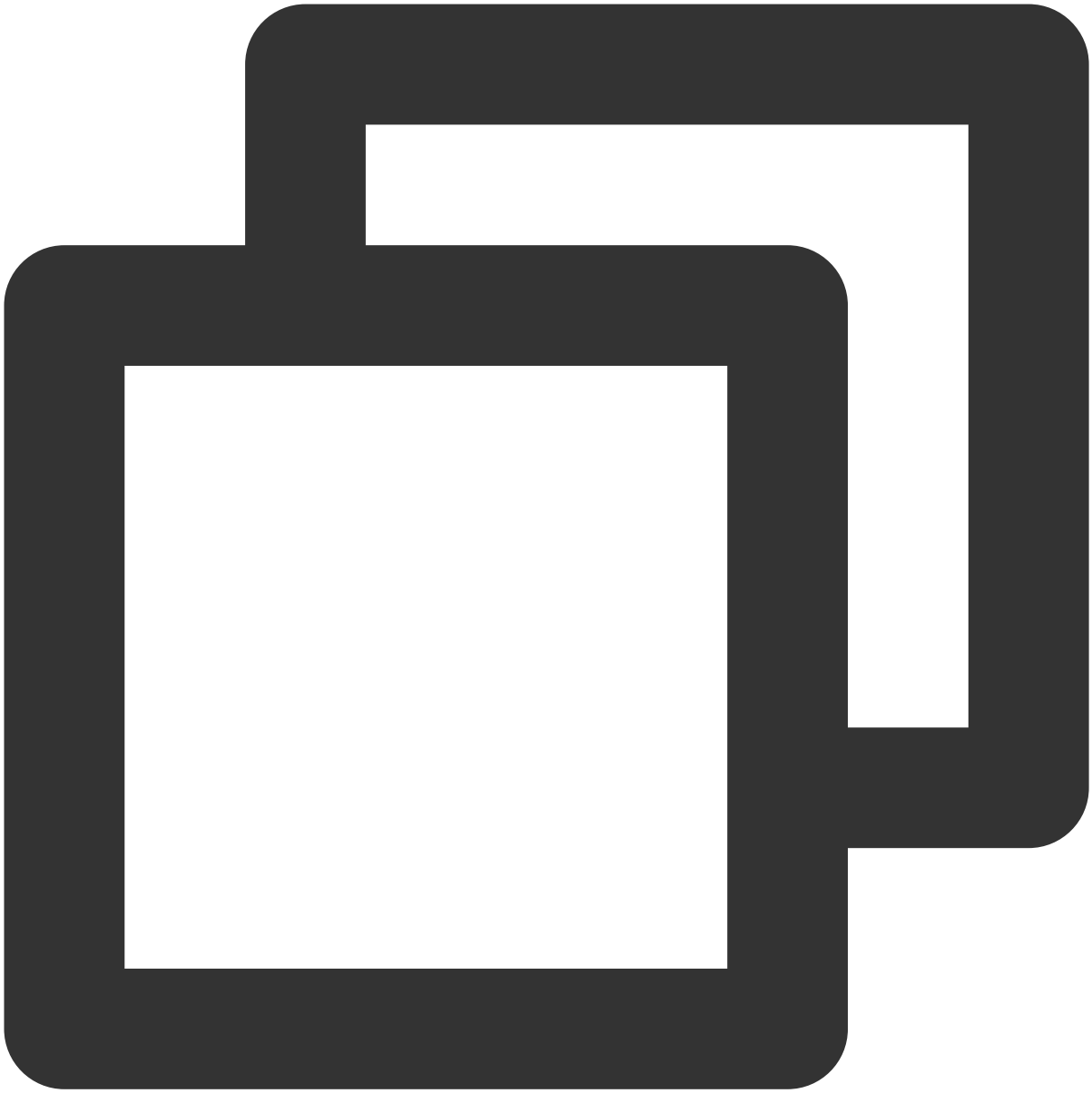
Run the following command to obtain kubectl:



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

Windows

Run the following command to obtain kubectl:

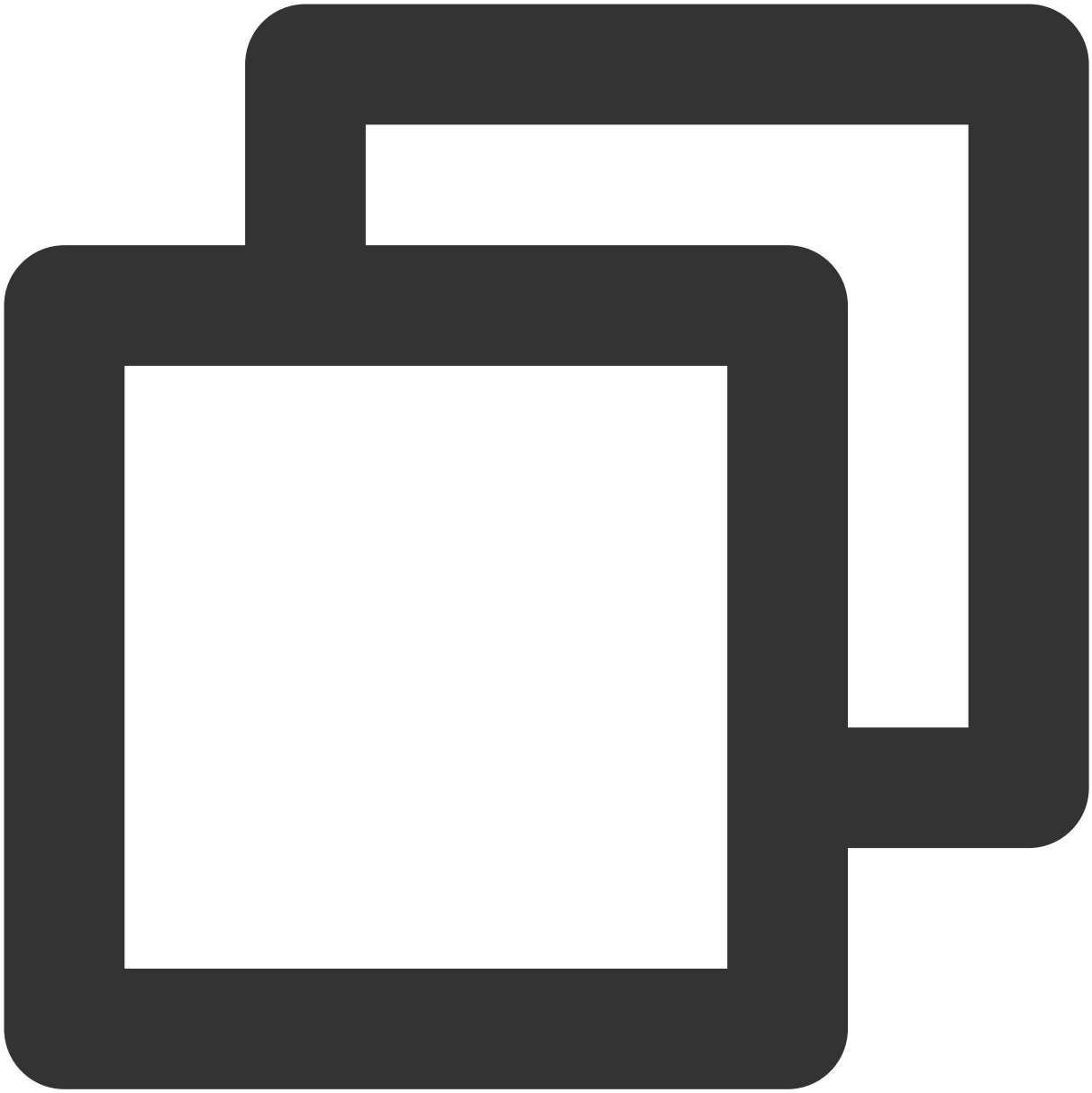


```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.4/bin/wi
```

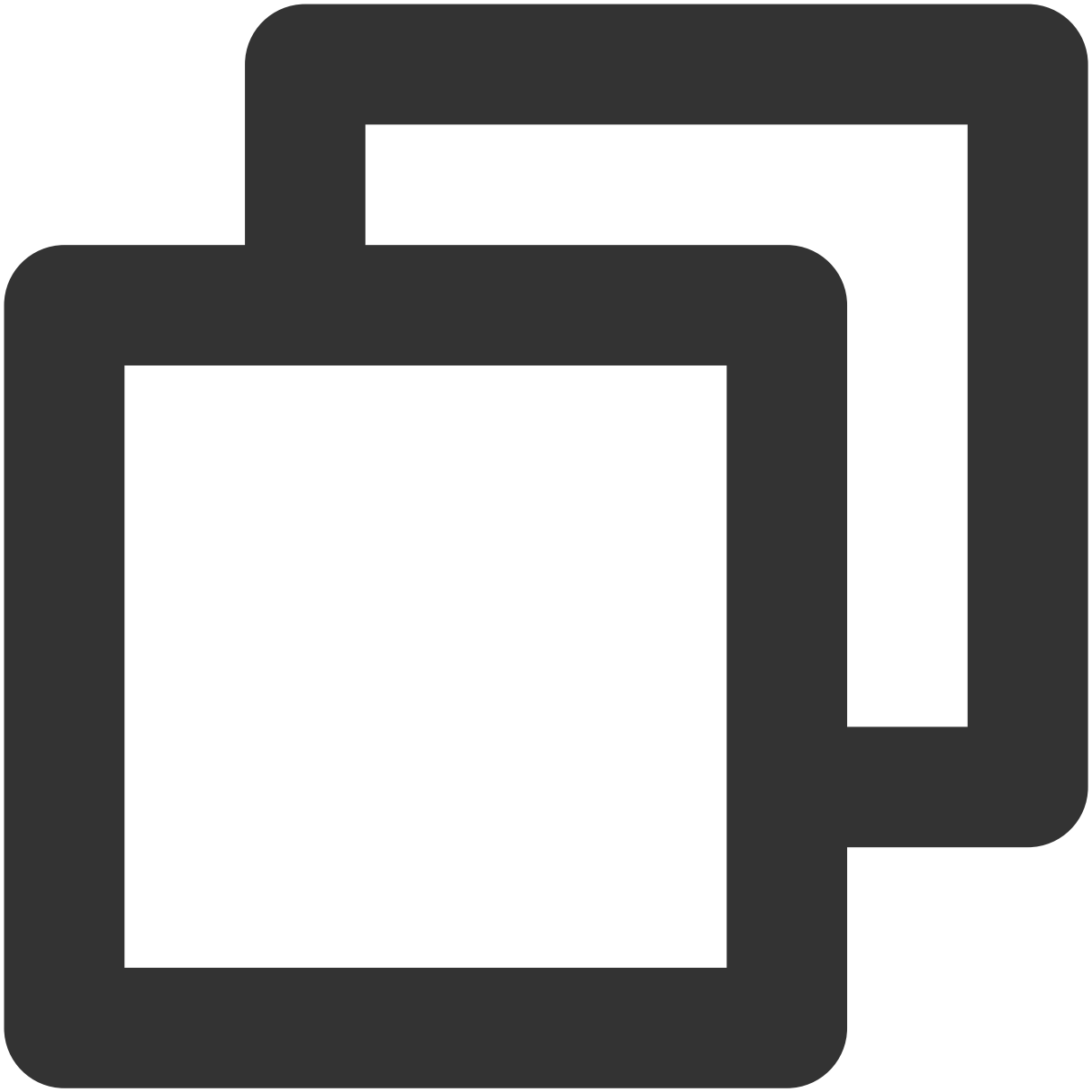
Operation Steps

Installing kubectl

1. Install kubectl as instructions above.
2. Run the following commands to grant permissions to use kubectl.

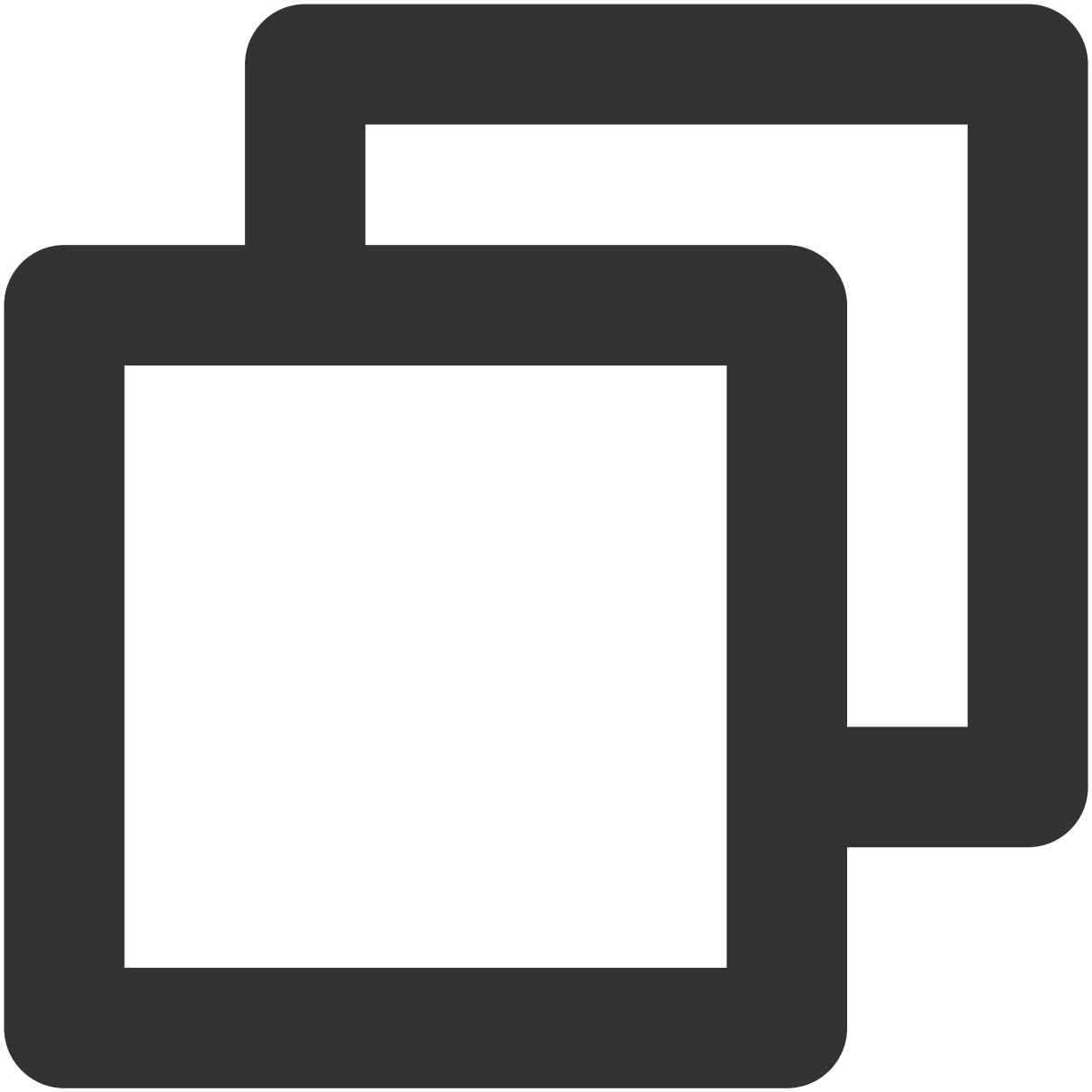


```
chmod +x ./kubectl
```



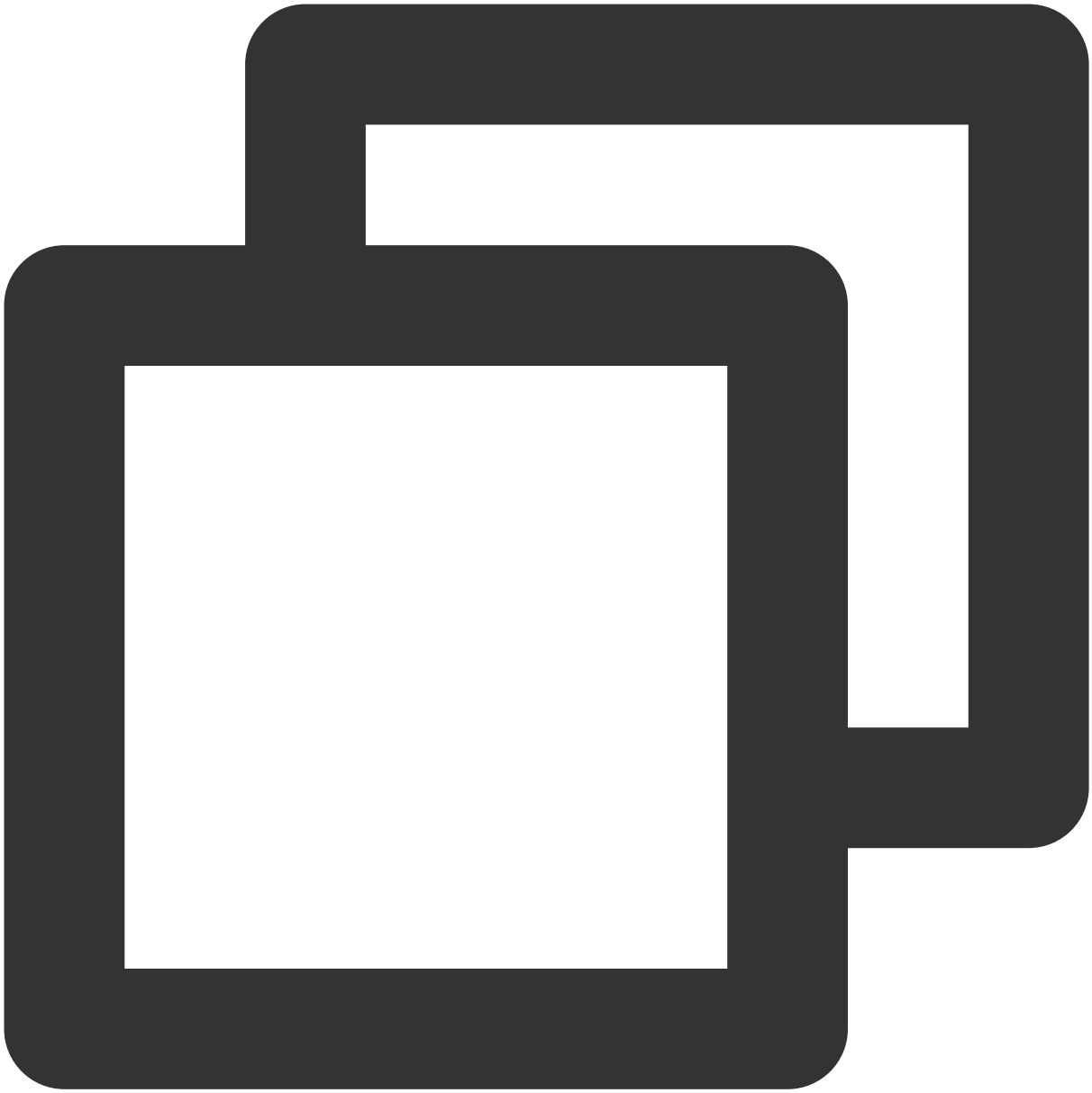
```
sudo mv ./kubectl /usr/local/bin/kubectl
```

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



```
kubectl version
```

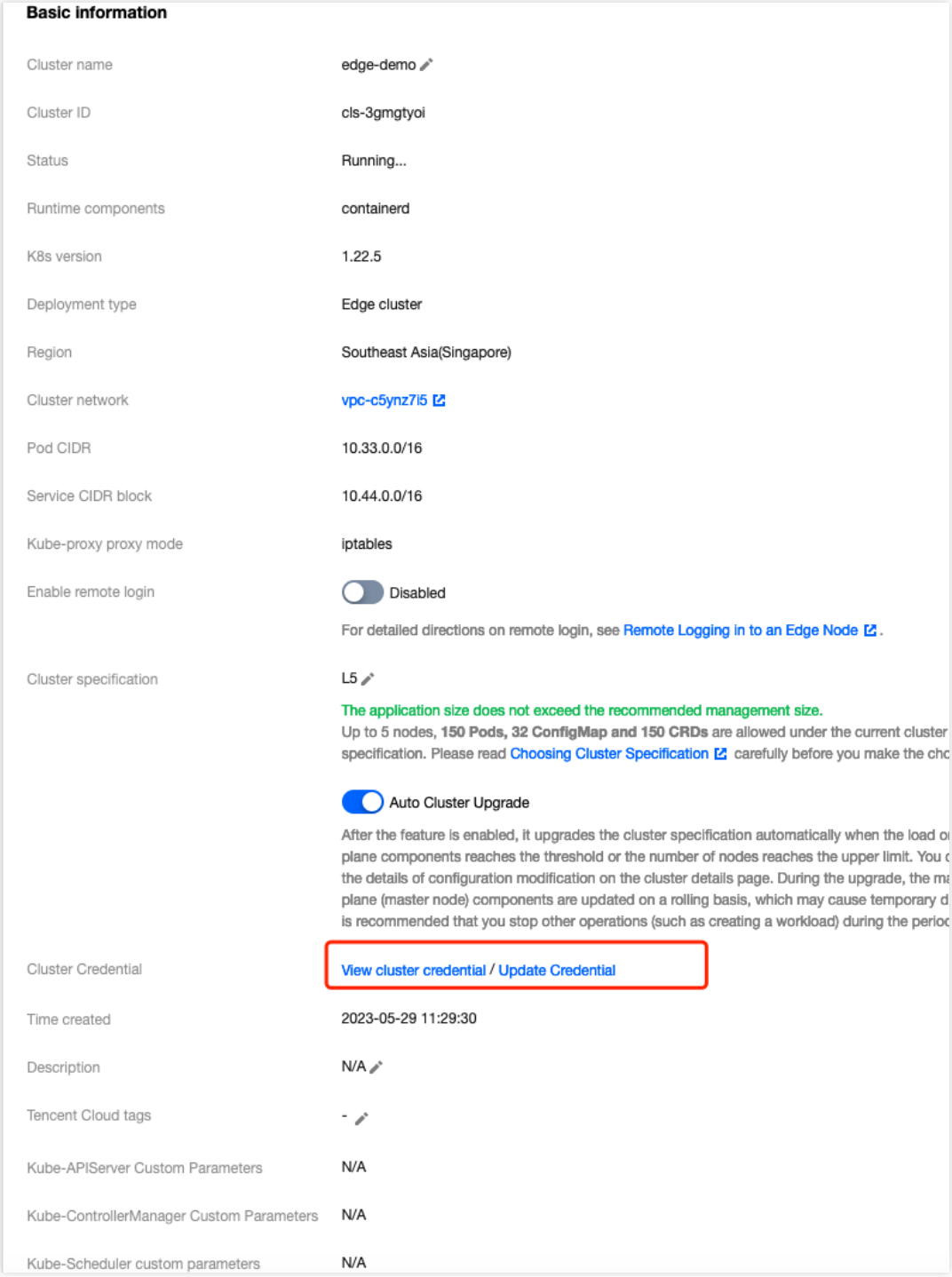
If the output is similar to the following version information, the installation was successful.



```
Client Version: version.Info{Major:"1", Minor:"5", GitVersion:"v1.5.2", GitCommit:"
```

Obtaining cluster certificate information

1. Log in to the Tencent Cloud TKE console and click **Edge Clusters** in the left sidebar.
2. On the **Edge Clusters** page, click **View the cluster credential** for the cluster to be connected.



3. In the **Cluster Credential** window that appears, you can view, copy, and download the credential.

Note：

You can save the cluster access credential locally by clicking **Copy** or **Download** as needed.

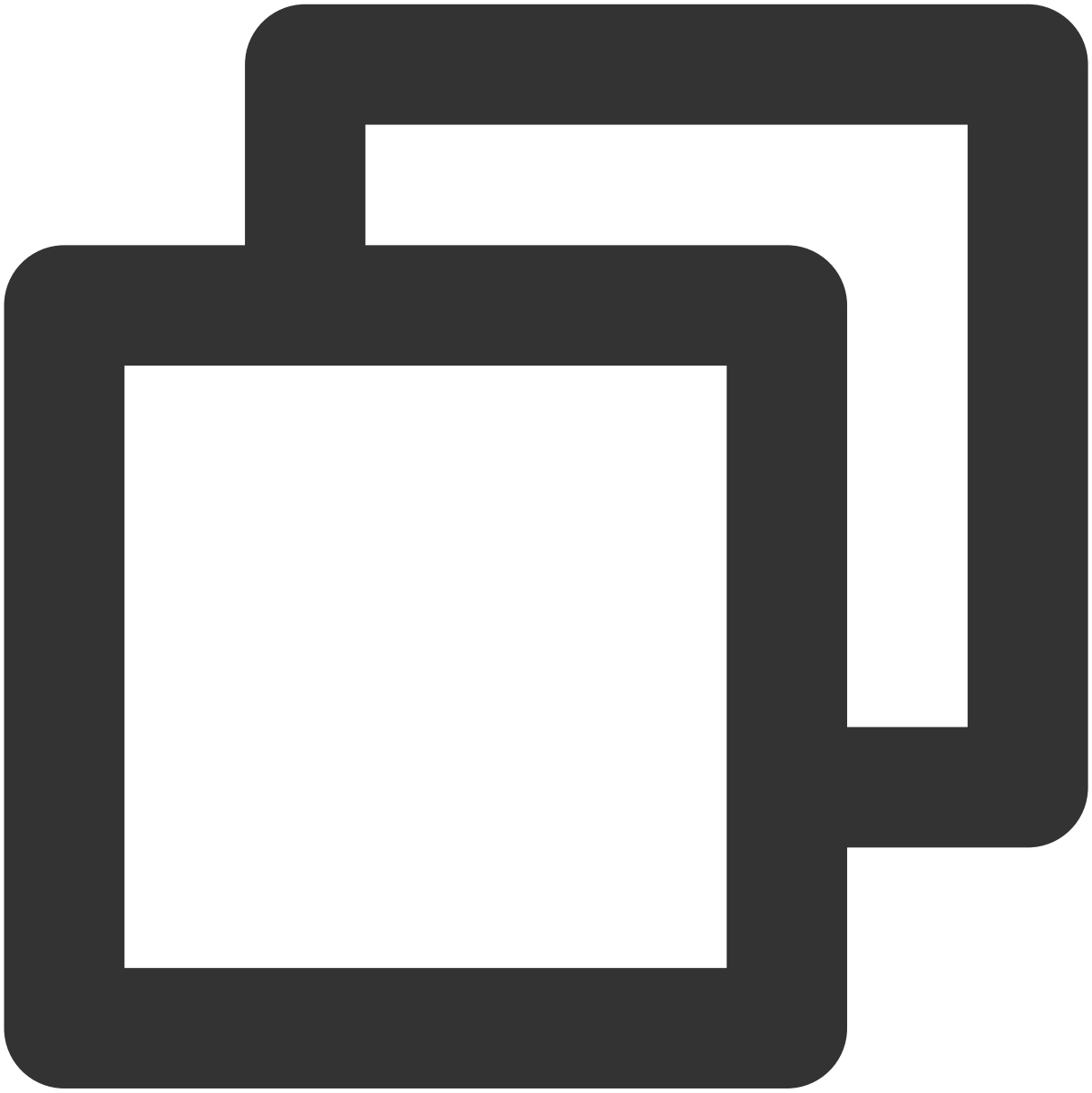
4. In **Internet access**, click

to enable internet access for the cluster. You can also see Configuring kubectl autocomplete for accessing with the cluster access credential.

Using kubectl to manipulate a cluster through certificate information

Request method

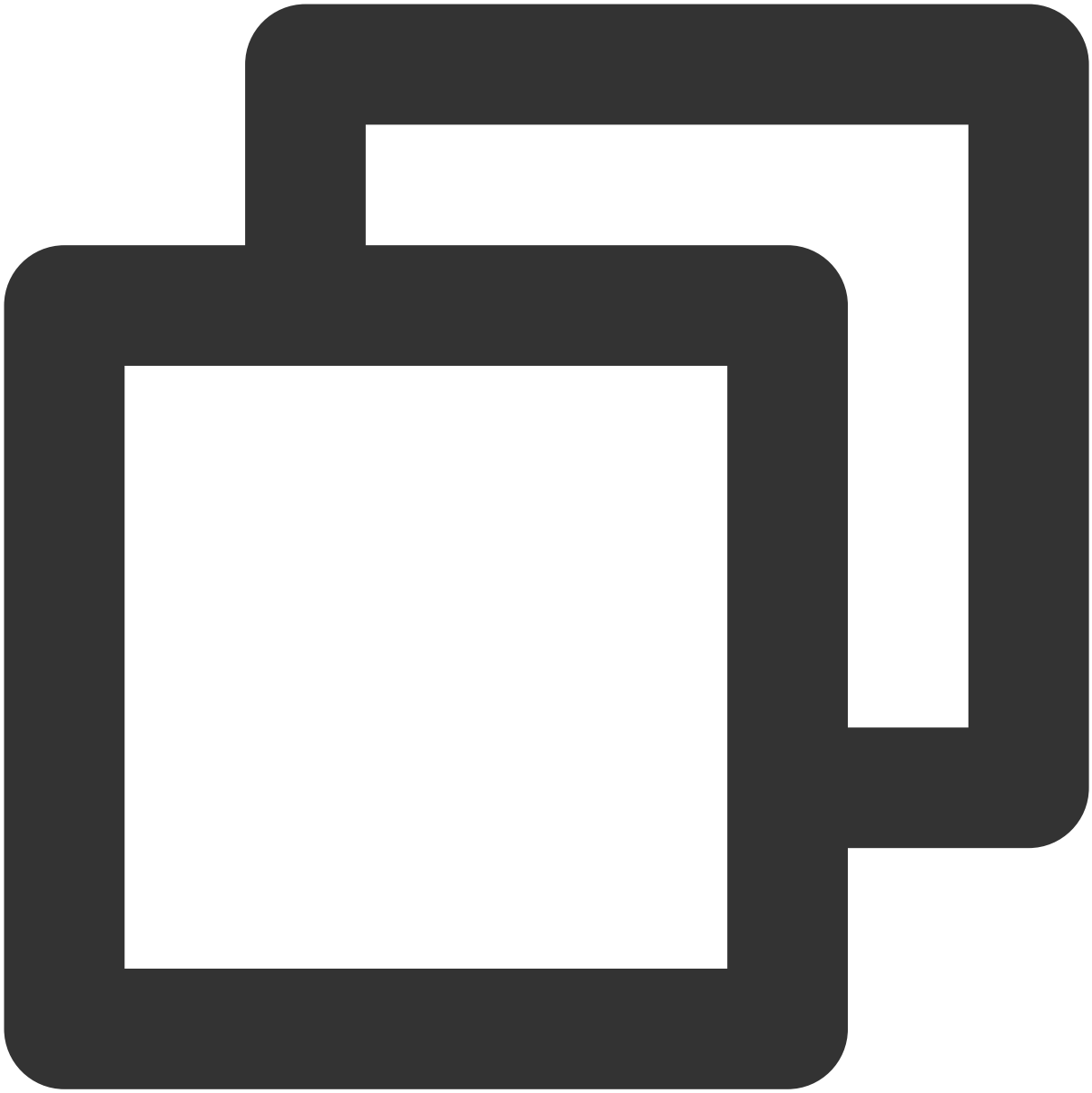
The kubectl command format is as follows:



```
--kubeconfig=<Local cluster access credential>
```

Example

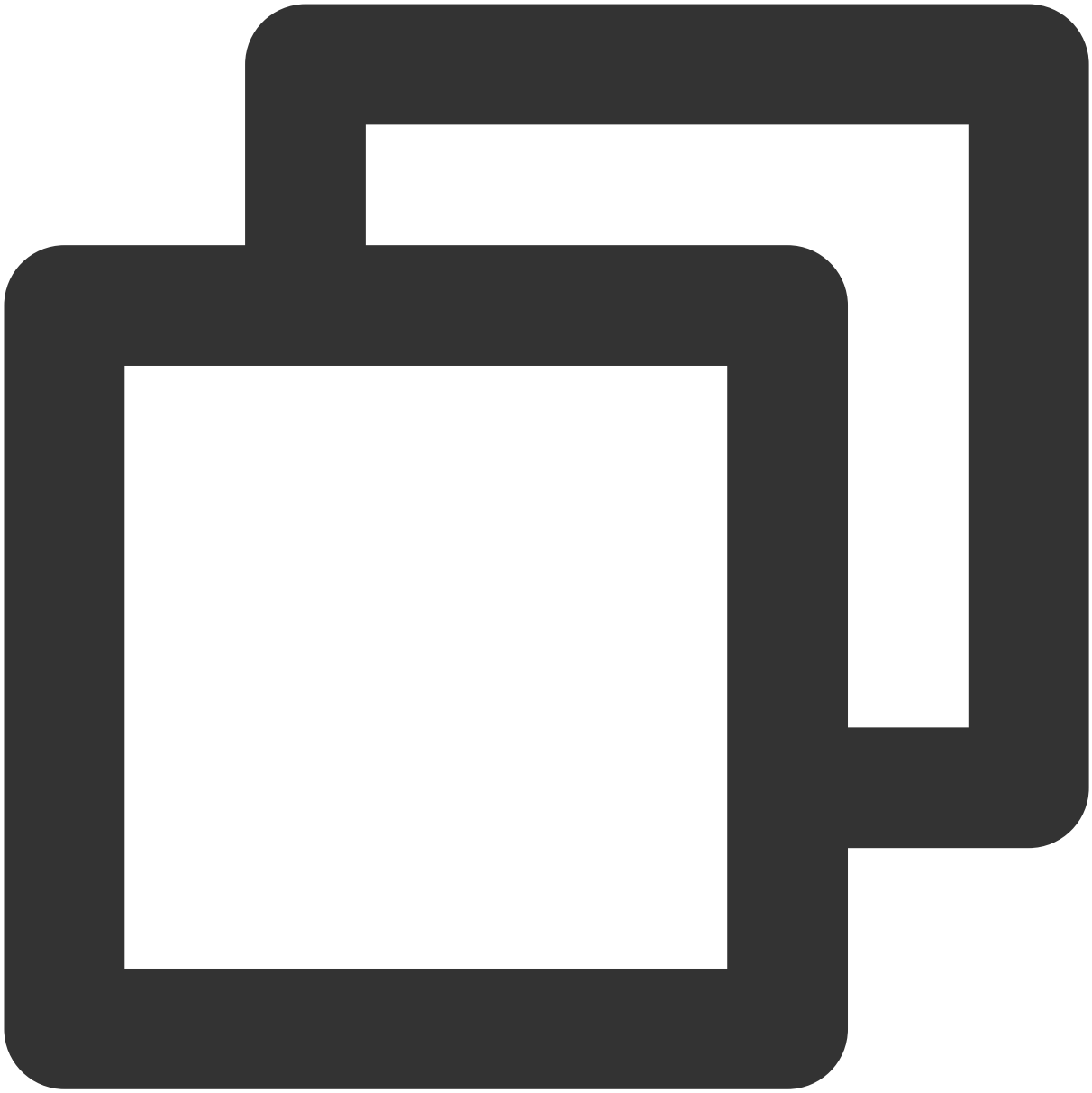
Run the following command to view existing namespaces under the cluster.



```
kubectl get namespace --kubeconfig=cls-8ipgf8u4.kubeconfig
```

The cluster credential used in this example is `cls-8ipgf8u4.kubeconfig`. In real-life cases, replace it with the actual credential.

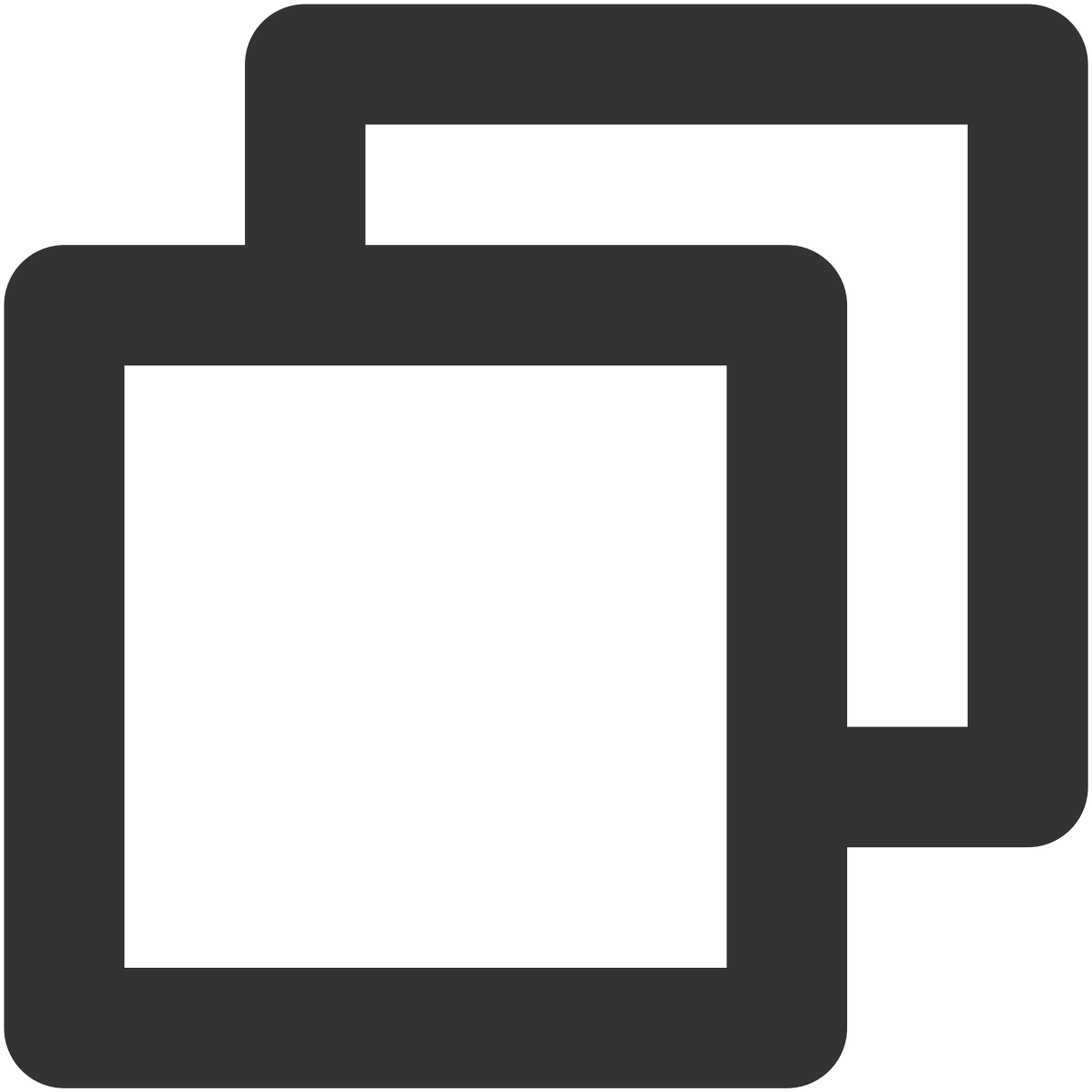
If a message similar to the following is returned, the request was successful.



```
NAME          STATUS    AGE
default       Active    11d
kube-system   Active    11d
```

Configuring kubectl autocomplete

You can configure kubectl autocomplete to improve usability by running the following command.



```
source <(kubectl completion bash)
```

Edge Node Management
Add Node
Add ECM

Last updated：2023-06-01 11:22:54

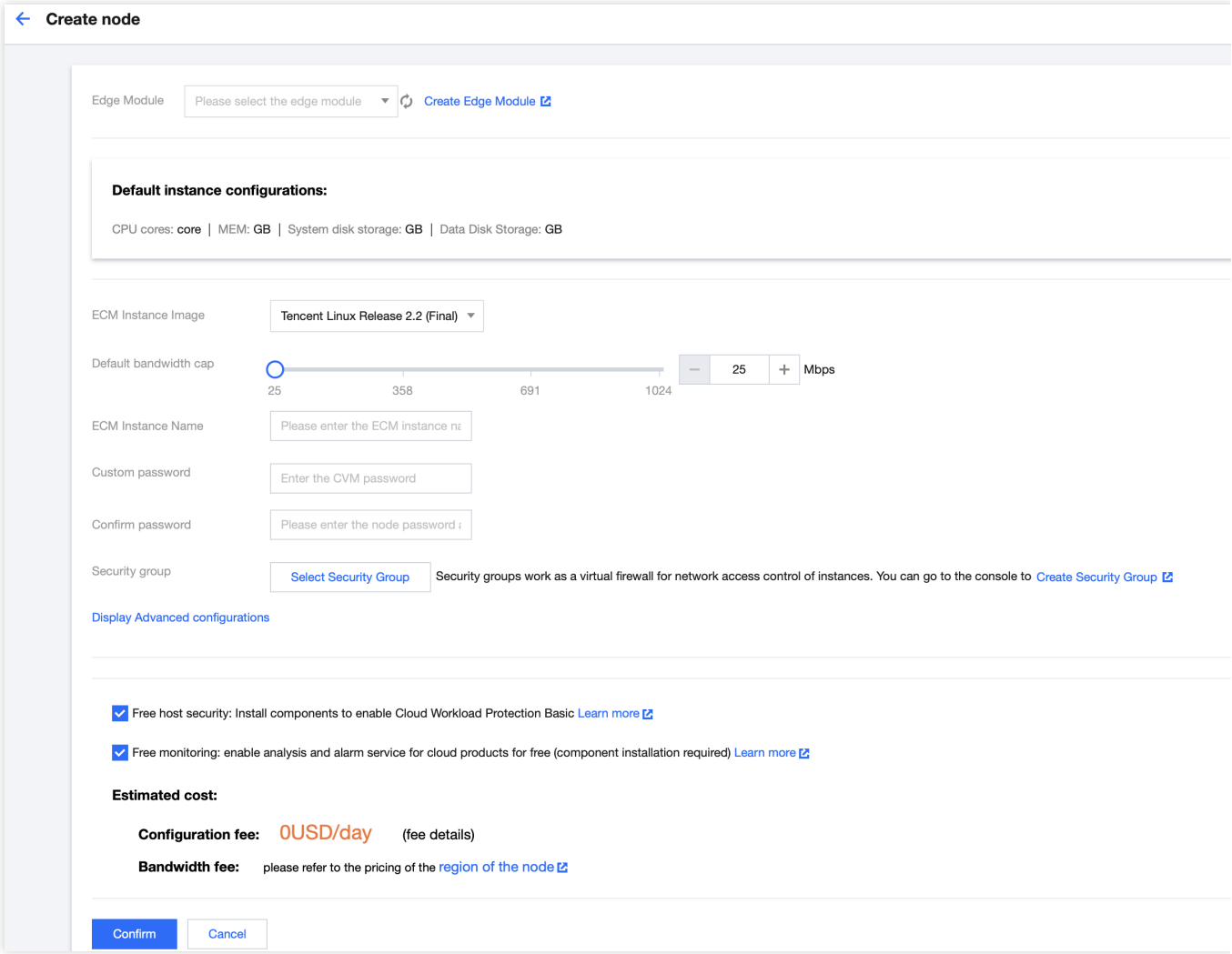
This article describes how to add an ECM node to the edge container cluster. ECM is short for Tencent Cloud Edge Computing Machine. Based on Tencent Cloud's edge nodes distributed in various regions, ECM provides cloud computing and networking services. The edge nodes are widely distributed, closer to users, and have the characteristics of low network latency, high availability, and low cost, which can seamlessly integrate with the edge container service.

Prerequisites

Refer to Cluster Enable Internet/Intranet Access to enable **external network access** capabilities.

Create an ECM node

1. Log in to the Container Service console and select the cluster in the left navigation bar.
2. In the cluster list, click the ID of the edge cluster to enter the cluster details page.
3. Select Node Management > Node on the left side of the page to enter the node list page, and click Create ECM Node.
4. On the "Create Node" page, configure relevant parameters according to actual needs, as shown in the following figure:



The main parameter information is as follows:

Edge Module: Select the edge module to determine the ECM edge node specifications.

ECM Instance Image: Tencent Cloud provides public images and custom images. The default is the same image as the module to which it belongs. Please select it according to actual needs.

Default bandwidth cap: Limits the bandwidth upper limit. If it exceeds this limit, packets will be lost by default. The default is 25Mbps, and the upper limit is 1024Mbps.

ECM Instance Name: Indicates the name of the instance to be created, user-defined.

Set password and confirm password: Define the password for logging in to the instance.

Security group: A security group is a virtual firewall used for network access control of instances. The default selected security group is the security group of the module to which it belongs, and users can modify the security group settings by themselves.

Advanced settings: You can modify the default settings. Please select according to actual needs:

The maximum number of Pods: determines the upper limit of the number of Pods allocated to each Node.

Container directory: Set container and image storage directories, and it is recommended to store them on data disks. For example, /var/lib/docker.

Launch configuration: shell script format.
Province of the Node: It is recommended to select the province closest to your customers, which can reduce access latency and increase access speed.
Node region: Please select according to actual needs.
Network type: Please select public network operators according to actual needs.
Free host security: Default selected, it helps users build a server security protection system to prevent data leakage.
Free monitoring: Default selected, it opens free cloud product monitoring, installs components to obtain host monitoring indicators and displays them in the form of monitoring icons, and supports setting custom alarm thresholds.

5. After clicking OK, wait for the corresponding ECM instance to be created successfully and its status to become Ready.

Add CVM

Last updated：2023-06-01 11:22:54

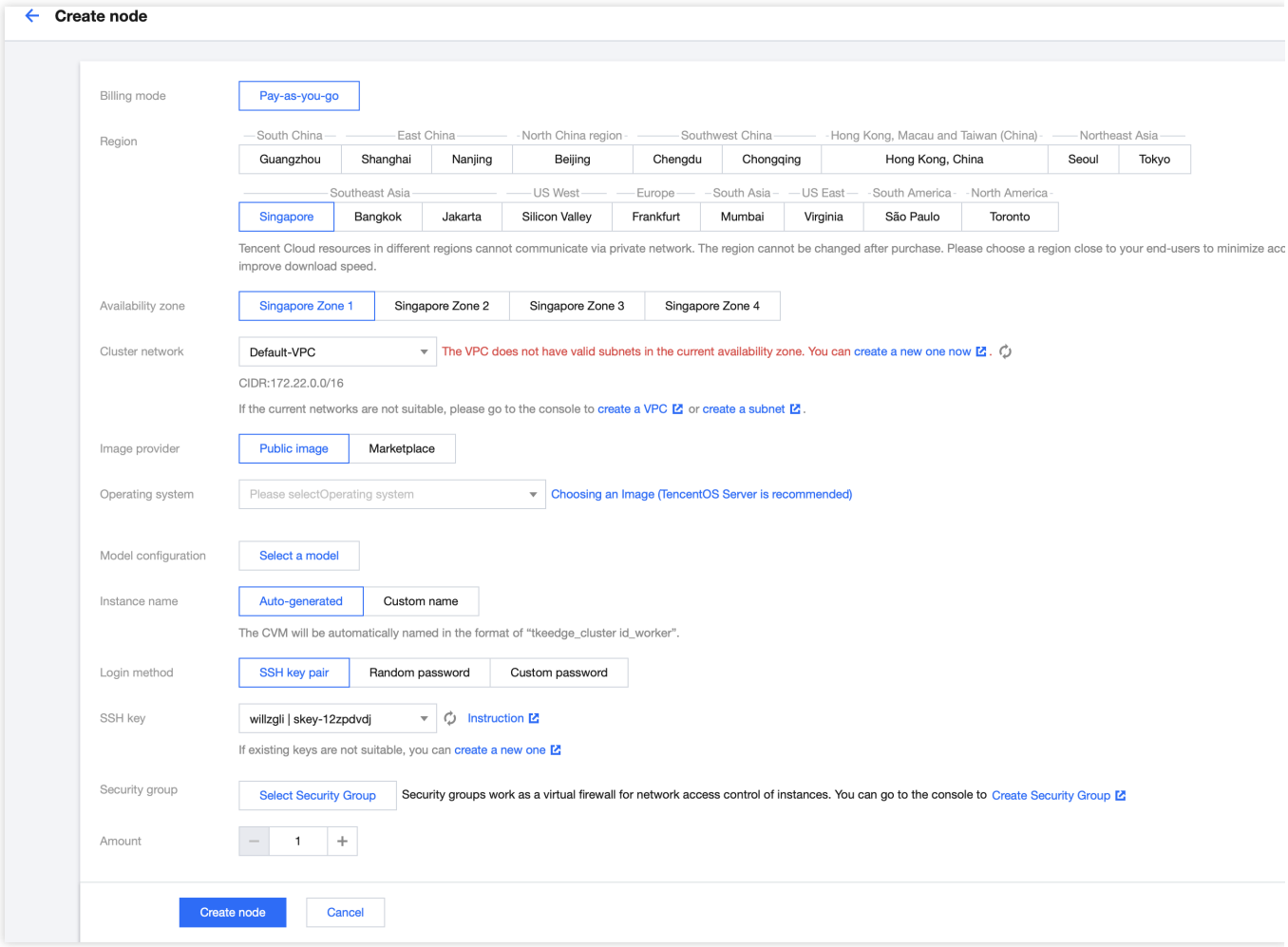
This article describes how to add Tencent Cloud Standard CVM nodes to an existing edge cluster. You can directly purchase Tencent Cloud CVMs in the required region and add them to the edge cluster to provide services. The difference between this and a standard TKE managed cluster is that you can purchase CVM nodes from different regions and VPCs and add them to the same edge cluster, rather than being limited to the VPC selected when creating the edge cluster.

Prerequisites

Please refer to Enable Internet/Intranet Access to enable external network access capability.

Creating a CVM Node

1. Log in to the Container Service console and select "Clusters" from the left navigation bar.
2. On the Cluster Management page, click on the edge cluster ID for which you want to create a CVM node to enter the details page for that cluster.
3. Select "**Node Management** > **Nodes**" from the left-hand side of the page to enter the Node List page. Click on "**Create CVM Node**", as shown in the figure below:



4. On the "New Node" page, configure the relevant parameters based on your actual requirements. The parameters are similar to adding nodes in TKE, please refer to Adding Nodes for details.

Add 3rd Node using script

Last updated：2023-07-14 18:10:08

This article describes how to use the edgectl script to add user's third-party nodes to an edge cluster. Unlike CVMs and ECMs, these nodes belong to the user and may be located in the user's IDC room, office, or various types of edge regions. These machines can access Tencent Cloud services over the Internet. In this scenario, users can download the edgectl script from the cloud and execute it locally to add the node to the edge cluster.

Prerequisites

Please refer to Enable Internet/Intranet Access to enable external network access or internal network access capability.

Note：

After enabling internal network access, if you need to add internal network CVM machines, you can only do so using the script to add nodes for now.

Prerequisites

Prepare the edge node according to the following conditions:

Node source: You can use existing servers from the Tencent Cloud Console or Edge Computing Machine Console, servers from other platforms, or servers from self-built facilities.

Node processors: x86_64, ARM64, ARM.

The following operating systems are supported (tested and adapted for use):

Ubuntu 20.04/18.04/16.04

CentOS 8.0/7.8/7.6/7.2

TencentOS Server 3.2/3.1/2.6/2.4

Tencent Linux Release 2.2 (Final)

Debian 10.2/9.0

SUSE Linux Enterprise Server 12 SP3

Please ensure that the node you want to add has installed wget, systemctl, and iptables.

The node's network needs to have the ability to access the internet actively.

Note：

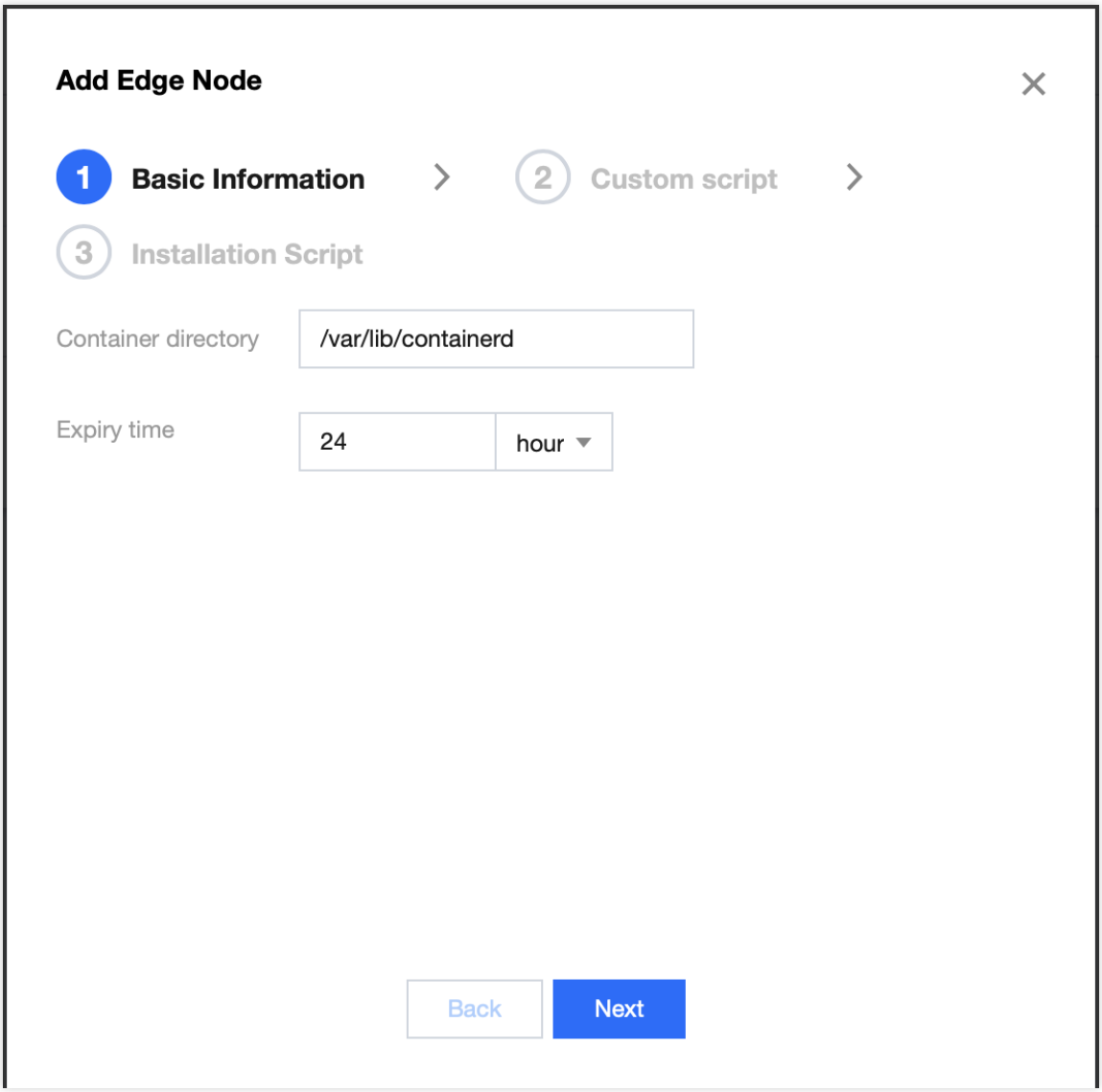
Due to different operating system environments deployed by different users, it is not ruled out that certain configurations may affect the successful addition of nodes. In this case, please check the deployment log of edgectl on

the node (/tmp/tkeedge-install.log) for detailed troubleshooting.

Procedure

1. Log in to the Tencent Cloud Container Service console and select "Clusters" from the left navigation panel.
2. On the Cluster Management page, click on the Edge Cluster ID that needs to add an edge node to enter the details page of that cluster.
3. Select "**Node Management**" > "Nodes" on the left side of the page and enter the node list page. Click "**Script to Add Node**".
4. In the "Add Edge Node" window that pops up, configure according to the following steps to obtain the initialization script for the node.

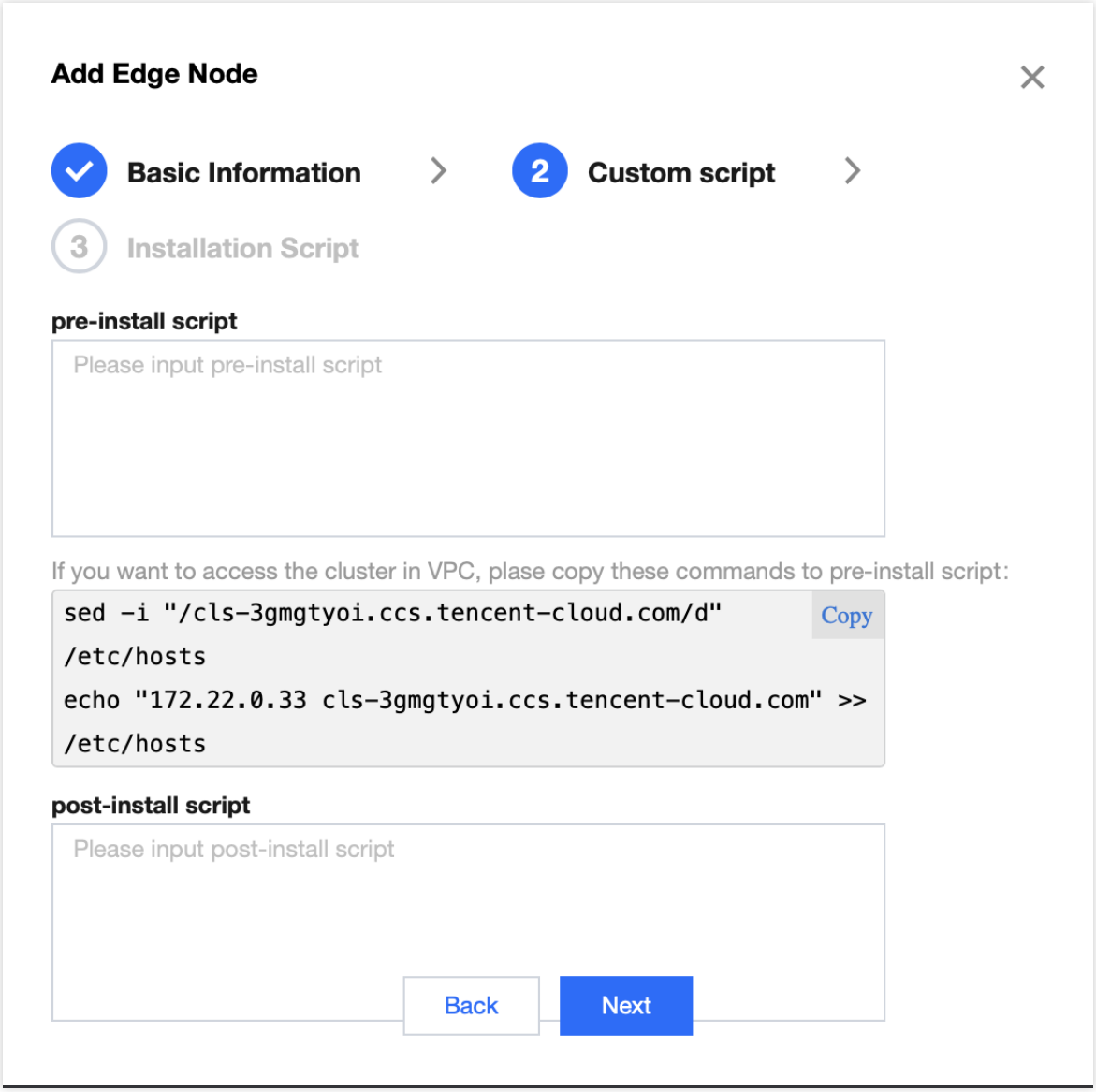
4.1 In the "Configuration" step, obtain the initial configuration and modify parameter values as follows:



Container directory: Set the container and image storage directory, it is recommended to store it in the data disk, for example /var/lib/docker.

Expiration time: Set the expiration time of the "installation script" command, the configurable range is from 1 minute to 24 hours, and the default configuration is 24 hours.

4.2 After clicking "Next", add custom scripts:



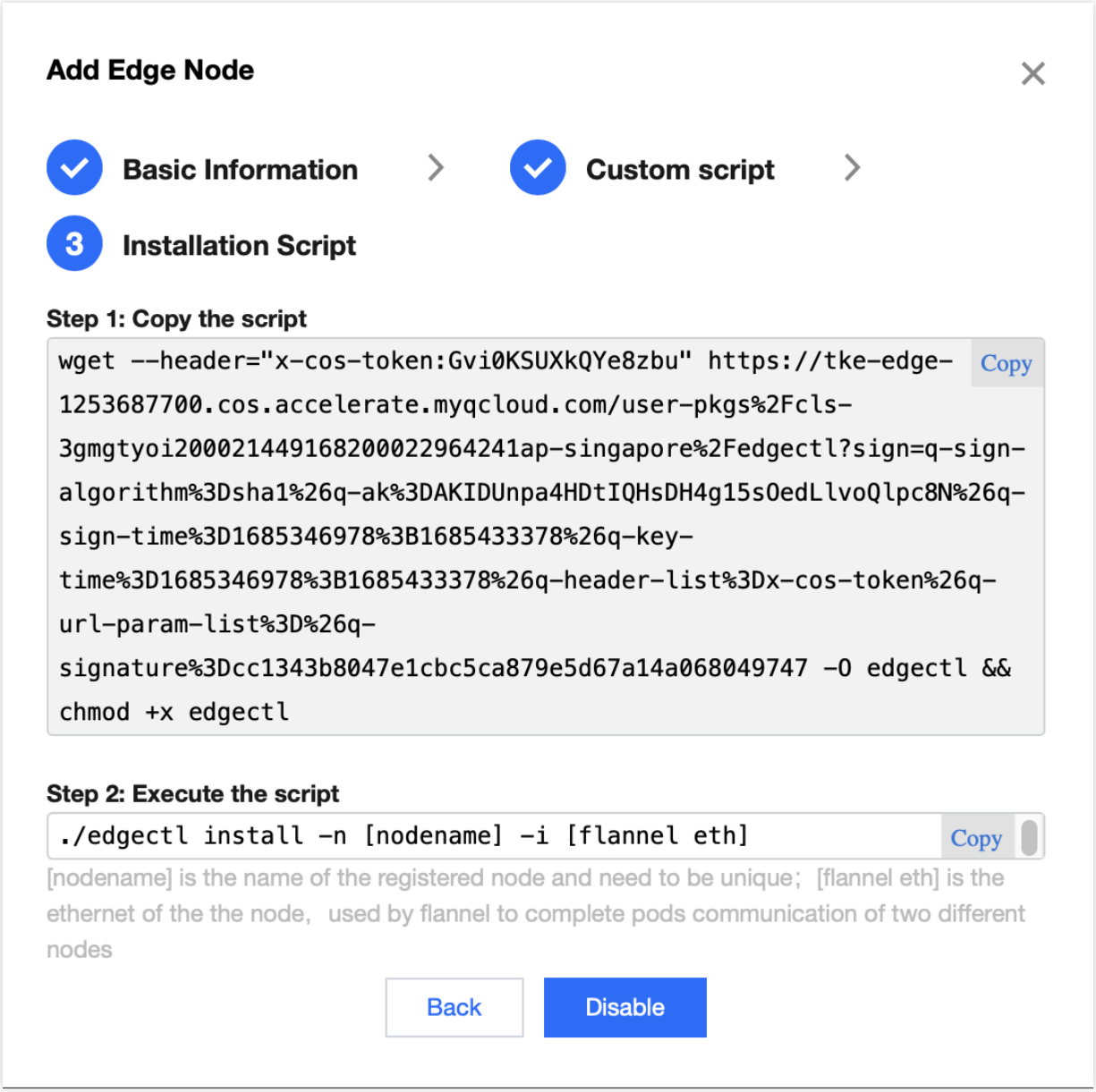
pre-install script: Here, you can set up pre-installation scripts for node access, which can be used to perform some cleanup or pre-configuration work under special conditions to enhance adaptability in different scenarios.

post-install script: Here, you can set up post-installation scripts for node access, which can be used to perform some subsequent cleanup or configuration work after the node is successfully added, enhancing adaptability in different scenarios.

Note：

In the above figure, because "Internal Network Access" is enabled, it will prompt for the execution of commands. If the user wants to add CVM nodes through the internal network, please copy and paste these instructions into the pre-install script. If adding nodes through the external network, these instructions can be ignored.

4.3 In the "Generate Installation Script" step, copy the script command to obtain the corresponding node initialization script.



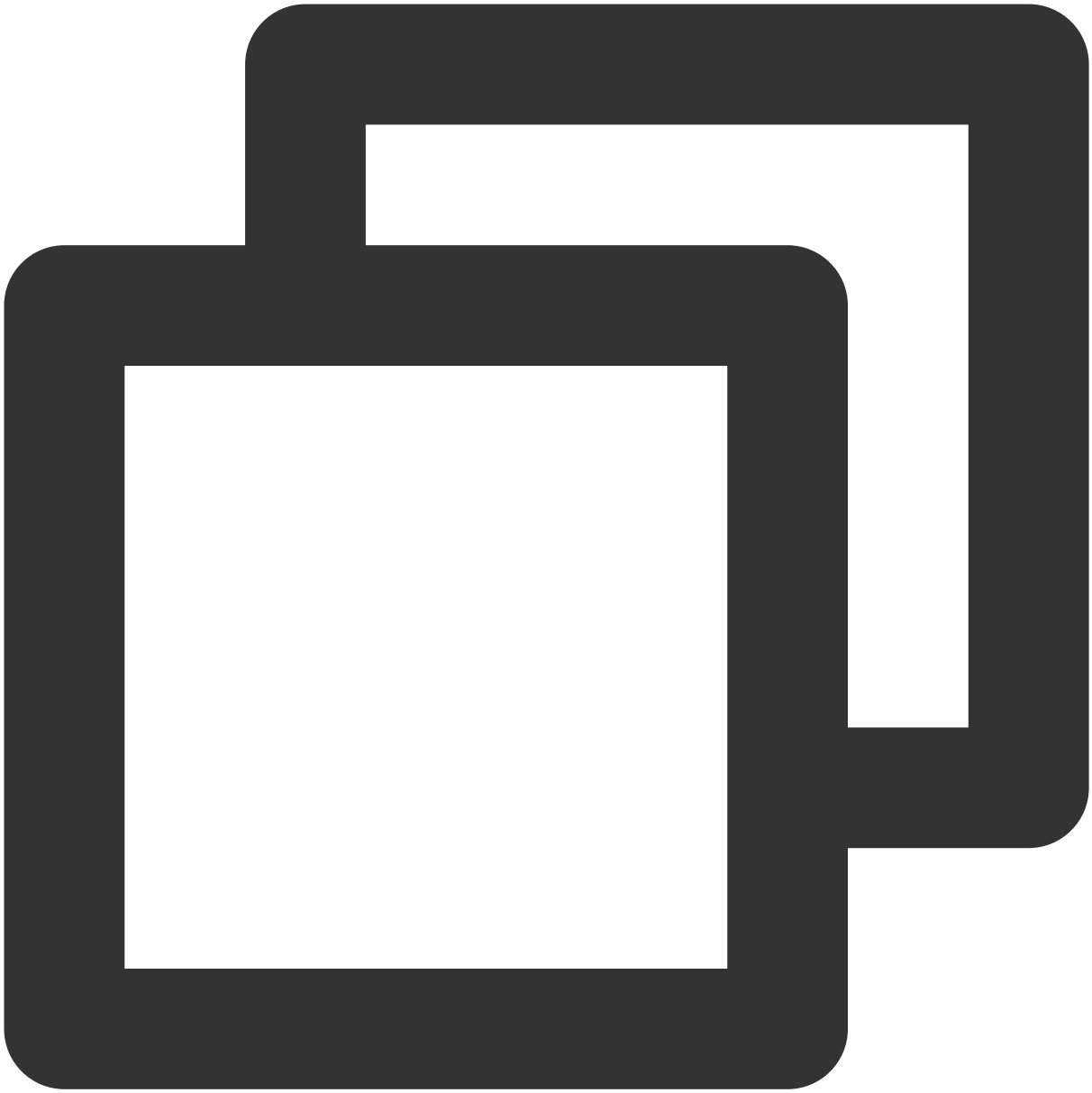
5. Log in to the prepared server and switch to the root account to execute the copied command.

Note：

Executing this command will automatically download the edgectl edge node management tool.

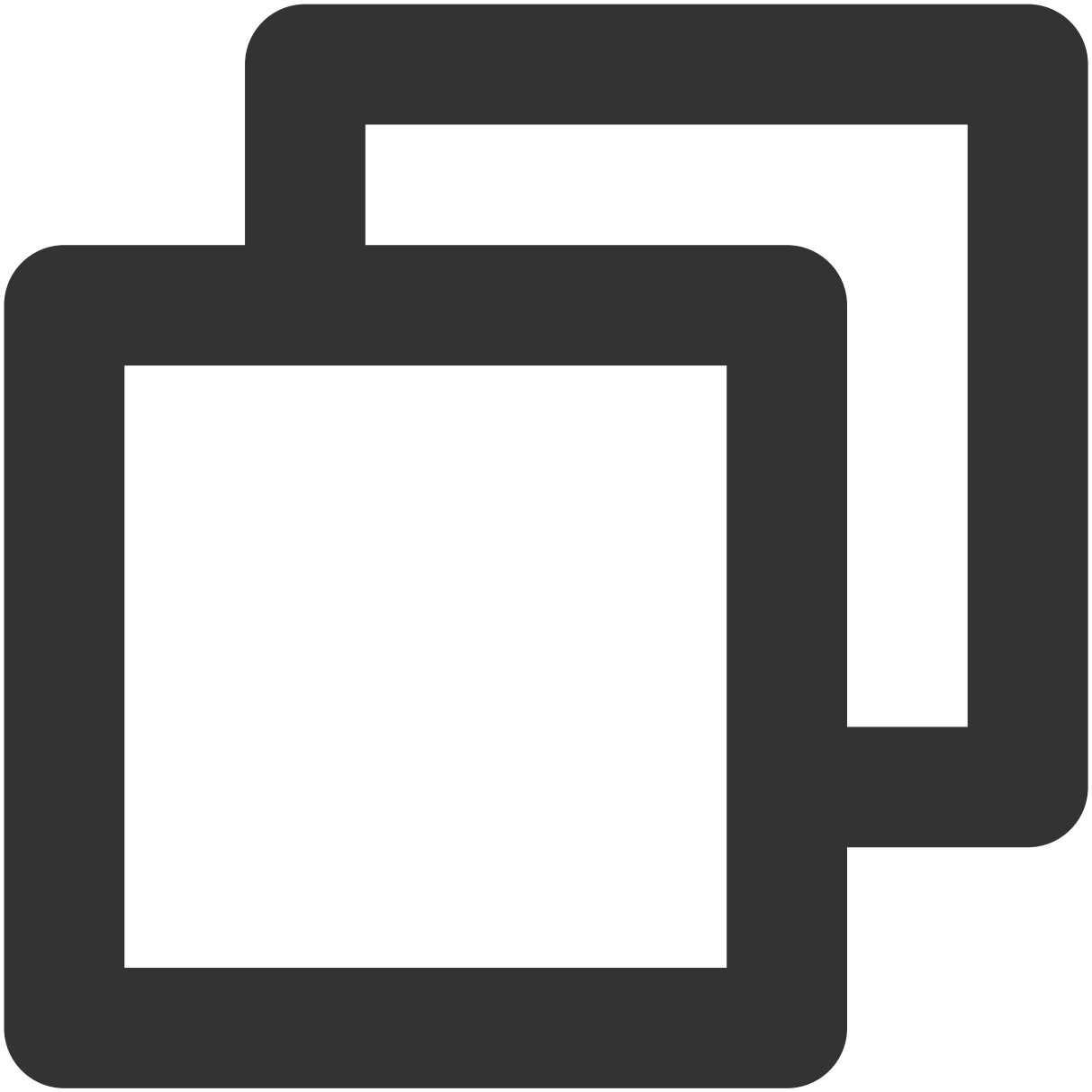
The edgectl obtained in this step is valid for the time period set by the "Expiration Time" parameter. Please complete the node addition within this time period. After the expiration time, please execute the script to add the node operation again to obtain edgectl again.

6. To view the edgectl command, run the "Initialize Node" operation. For more operations with edgectl, please refer to the Edgectl script manual



```
./edgectl -h

Usage:
   edgectl command [flags]

Available Commands:
   check        Check the edge node if to be add to clusters
   install      Install components to edge node
   clear        Clear edge node and recovery as usual

Flags:
```

```
    -h, --help   Help for edgectl
```



```
./edgectl install -n Your-Node-Name -i Your-Interface-Name
```

Note：

Executing this command will check the host environment, install the required components for the edge node and automatically register the node to the edge cluster.

Due to network and installation speed factors, adding nodes may take several minutes.

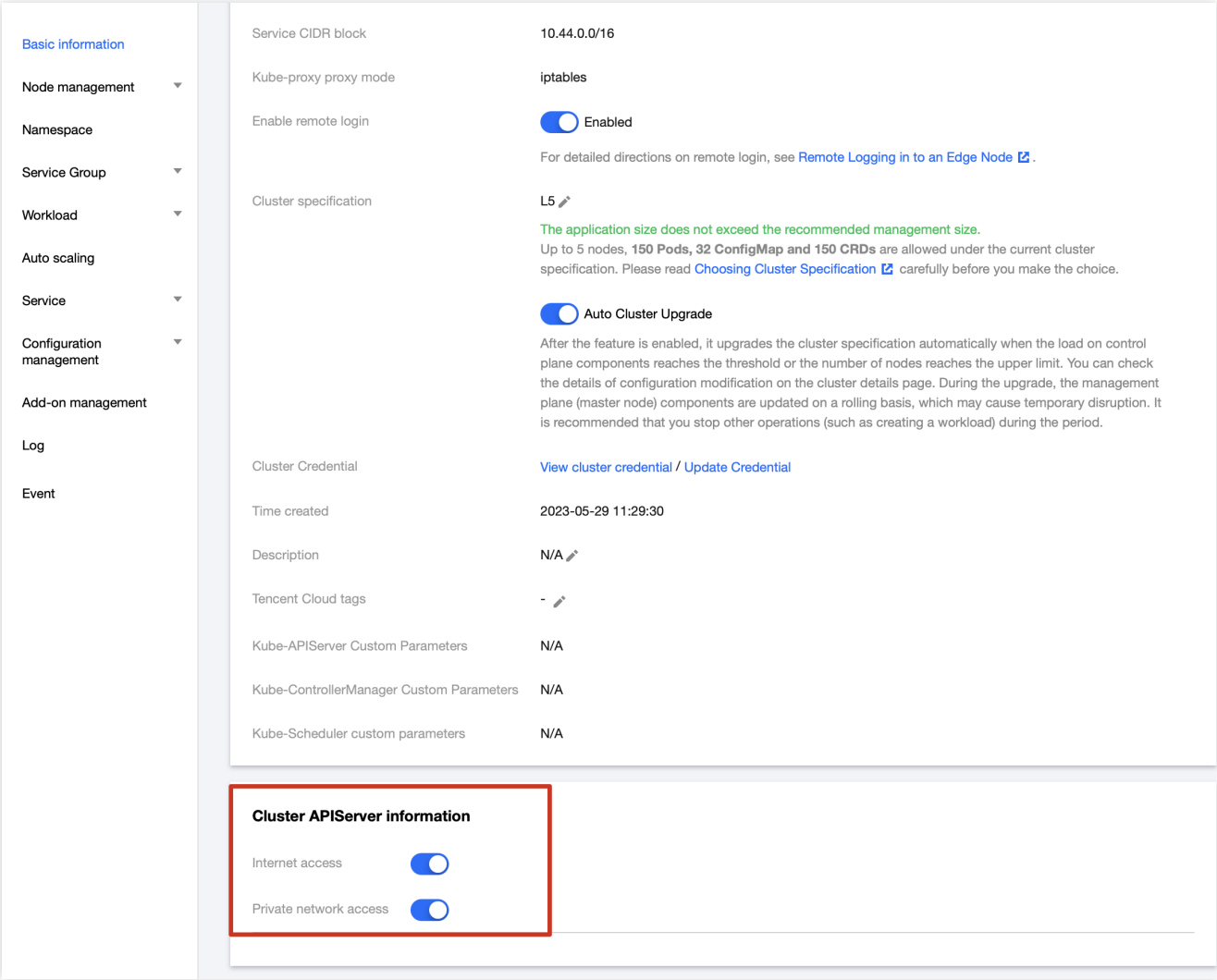
To confirm the installation process information, view the log file at /tmp/tke-edge-install.log.

7. After confirming the command has been executed successfully, go to the "Node List" page and refresh to view the newly added nodes. You can also perform other operations on the node, such as eviction, removal, blocking, editing tags or unblocking.

Related Operations

Close cluster Internet and Private access

To disable both external and internal network access to the cluster, go to the "Basic Information" page of the cluster and turn off "Internet Network Access" and "Private Network Access".



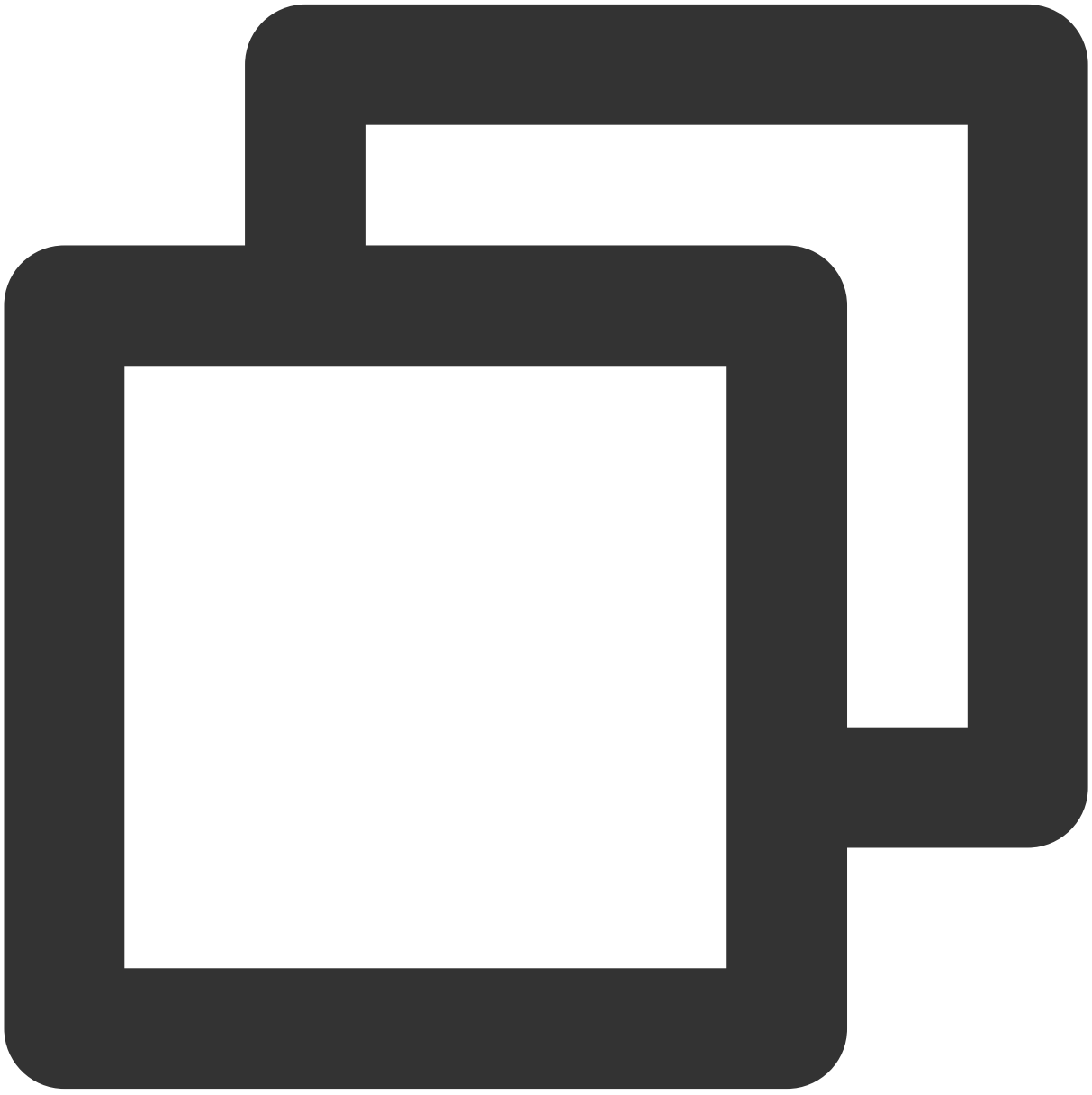
Edge Node Authority

1. The edge nodes managed by the edge cluster K8S have the default permission setting of "system:node". For details on the "system:node" permission, please refer to Using Node Authorization.

2. The kubeconfig file on the edge node will use the "system:node" permission by default. If you need to obtain the cluster certificate for operating the cluster, please refer to Connecting a Cluster.

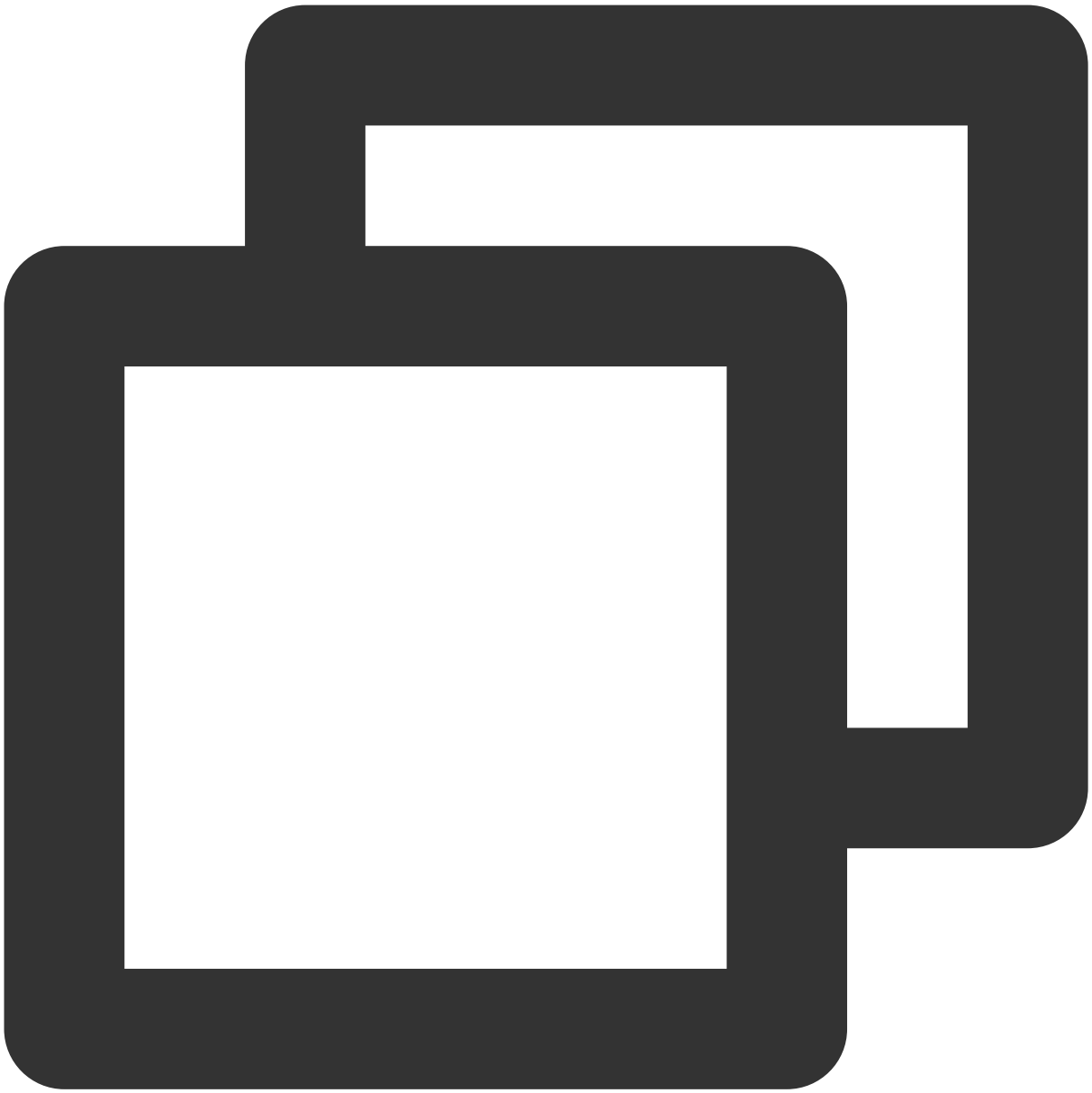
Edge GPU Node

The edge cluster already has a nvidia-device-plugin DaemonSet submitted with image version nvidia-device-plugin:v0.9.0. If an edge node needs to deploy a DaemonSet, execute the following command to automatically deploy it to the corresponding node:



```
kubectl label nodes ${NODE_NAME} nvidia-device-enable=enable
```

To remove the taint added by the previous command, execute the following command:



```
kubectl label nodes ${NODE_NAME} nvidia-device-enable-
```

In the above commands, replace `${NODE_NAME}` with the name of the node where the operation will be performed.

Versions and Operating Systems supported by Tencent Cloud GPU machines

The supported NVIDIA GPU series are:

GN6

GN7

GN8
GN10X, GN10Xp
GI3X

The following Linux systems are supported:

CentOS 7.2 64-bit
CentOS 7.6 64-bit
CentOS 7.7 64-bit
CentOS 7.8 64-bit
CentOS 7.9 64-bit
Ubuntu Server 18.04.1 LTS 64-bit
TencentOS Server 2.4
Tencent Linux Release 2.4
If a corresponding system with an NVIDIA GPU version is used, the edge cluster will automatically install GPU drivers. Users of other systems can install the corresponding GPU drivers themselves.When adding an edge node via script, the edge cluster is not responsible for installing GPU drivers. The user should install the GPU drivers themselves.

To add a CVM node with GPU support via script

For edge nodes added with scripts, the edge cluster is not responsible for installing the GPU driver, which needs to be installed by the user.

Note：

The CVM system version with GPU is a subset of the edge node system. If the system of the edge node does not support this version, adding a CVM node with GPU may fail. Therefore, please follow the requirements of the edge node system carefully.

Node Management

Last updated：2023-06-01 11:22:54

This article introduces how to manage the nodes that have already been added to the cluster.

View node information

1. Log into the Tencent Cloud Container Service console
2. On the cluster management page, click the cluster ID to enter the cluster details page.
3. Select **"Node management" > "Node"** on the left side of the page to enter the node list page.
4. Click on the node name you want to view, and enter the node details page. The detailed information of the node is displayed on this page:

Pod Management: View the Pod list and running status under this node.
Events: Display the Events information of this node within the Kubernetes cluster.
Details: Display the relevant host information and Kubernetes information of this node within the Kubernetes cluster.
YAML: Display the YAML information of this node's resource.

Cordon Node

If you need to block a node and prevent subsequent applications from being scheduled to it, you can evict the node by adding **label:SchedulingDisabled** to it.

1. Log in to the TKE-Edge and select "Clusters" from the left navigation bar.
2. In the Cluster Management page, click on the Edge Cluster ID to enter the cluster details page.
3. Select **"Node Management" > "Nodes"** on the left side of the page to enter the node list page.
4. Choose the node you want to evict, and select **"Operation" > "Cordon"**. The node will enter the "Cordoned" state, and subsequent applications will not be scheduled to this node.
5. If you want to unblock the node, select **"Operation" > "Uncordon"**, and the node will return to the normal state.

Delete Node

If a node is no longer needed within the cluster, you can delete it from the cluster.

1. Log in to the Tencent Cloud Container Service console and select "Clusters" from the left navigation bar.
2. In the Cluster Management page, click on the Edge Cluster ID to enter the cluster details page.
3. Select **"Node Management" > "Nodes"** on the left side of the page to enter the node list page.

4. Choose the node you'd like to delete and select **"Operation" > "Remove"**. The node will then be deleted from the Kubernetes cluster as a Node resource.

Caution

For nodes added using a script: This removal operation will only delete the Node from the Kubernetes cluster, equivalent to `kubectl delete node`, but it will not clean up the node data. Users are required to clean up the node data manually.

For nodes added using ECM/CVM: This removal operation will not only delete the Node from the Kubernetes cluster but will also release the applied ECM/CVM resources; users do not need to manually release these resources.

Node Remote SSH

Last updated：2023-06-01 11:36:31

Scenario

This article describes how to remotely log in to edge nodes via SSH in the cloud. The edge nodes can be public cloud nodes or local intranet nodes.

Enable node remote login

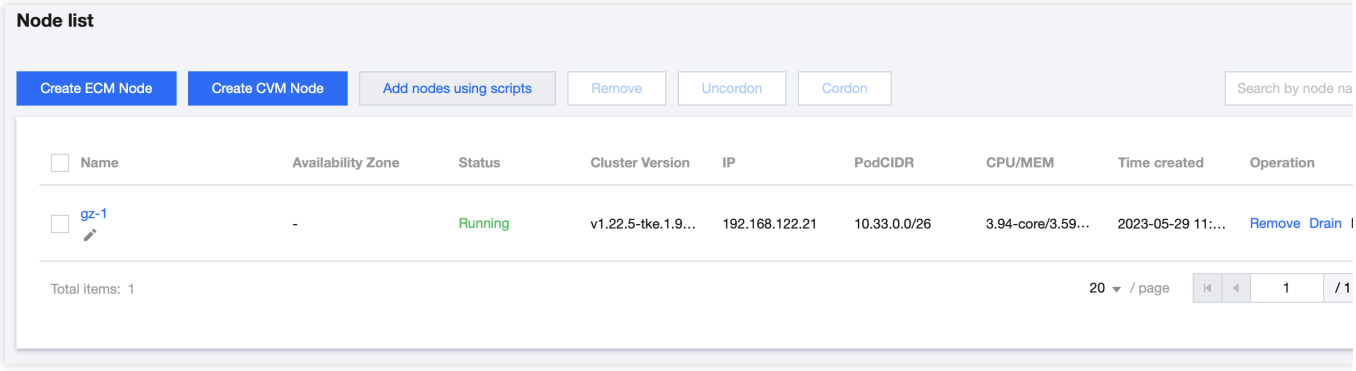
Close node remote login

Log in to the edge node remotely

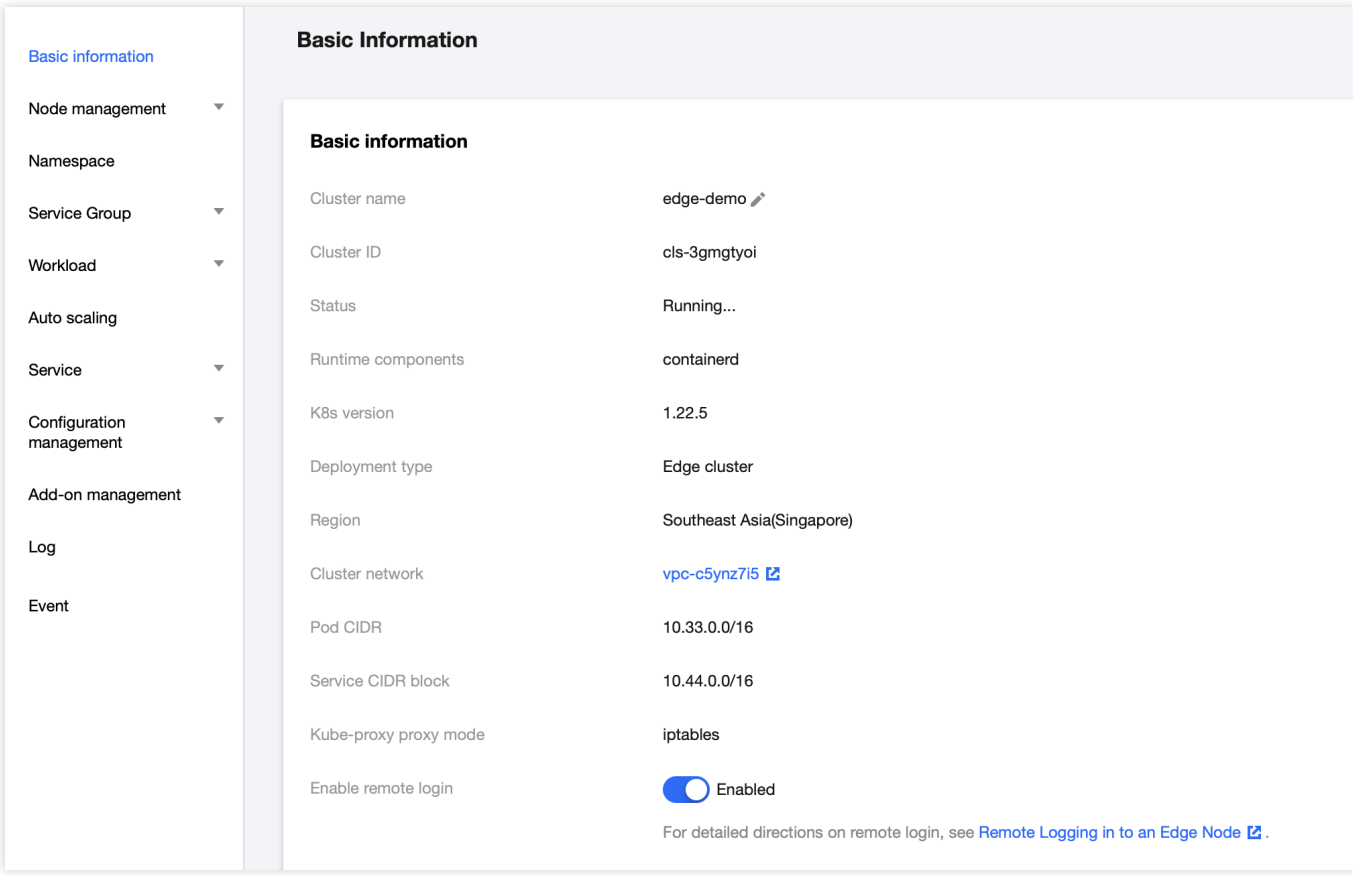
Procedure

Enable node remote login

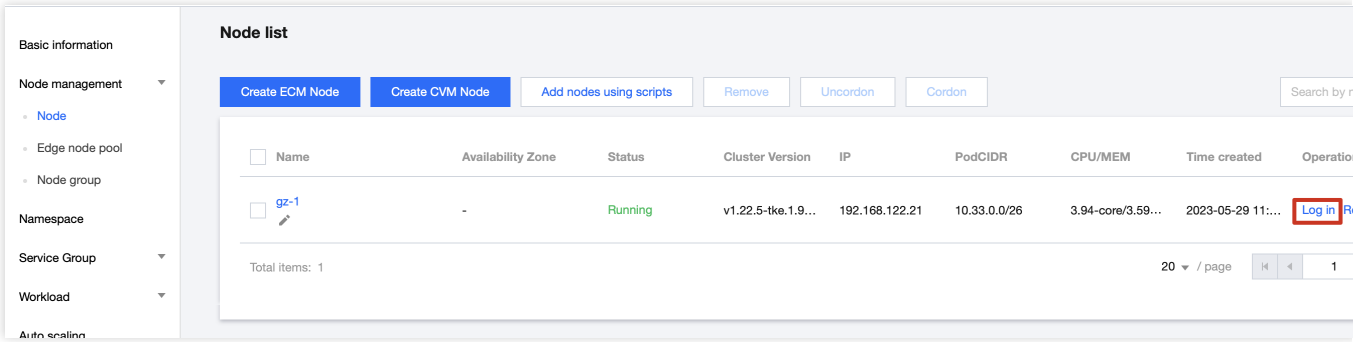
1. Log into the Tencent Cloud Container Service console
2. On the Cluster Management page, click the cluster ID of the node you wish to remotely log into to enter the details page for that cluster.
3. Select **Node Management** > **Nodes** from the left-hand side of the page to enter the Nodes list page, and confirm whether the login hyperlink is displayed on the operation column. By default, the hyperlink is not displayed, as shown in the figure below:



4. Select Basic Information from the left-hand side of the page to enter the Cluster **Basic Information** page, and click on the remote login switch to **enable the remote login** feature, as shown in the figure below:

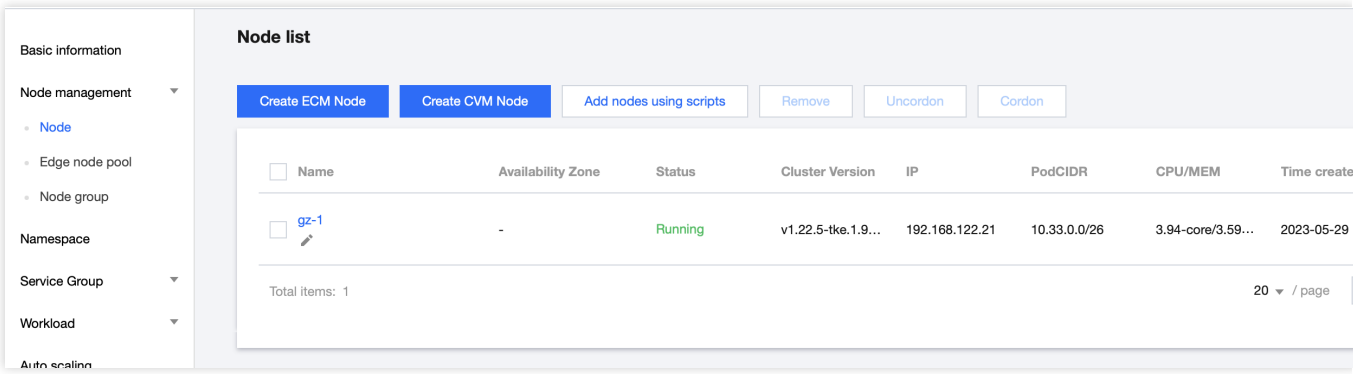


5. Return to **Node Management** > **Nodes** from the left-hand side of the page to access the Node List page again. Confirm that the **login** hyperlink is displayed, as shown in the figure below:

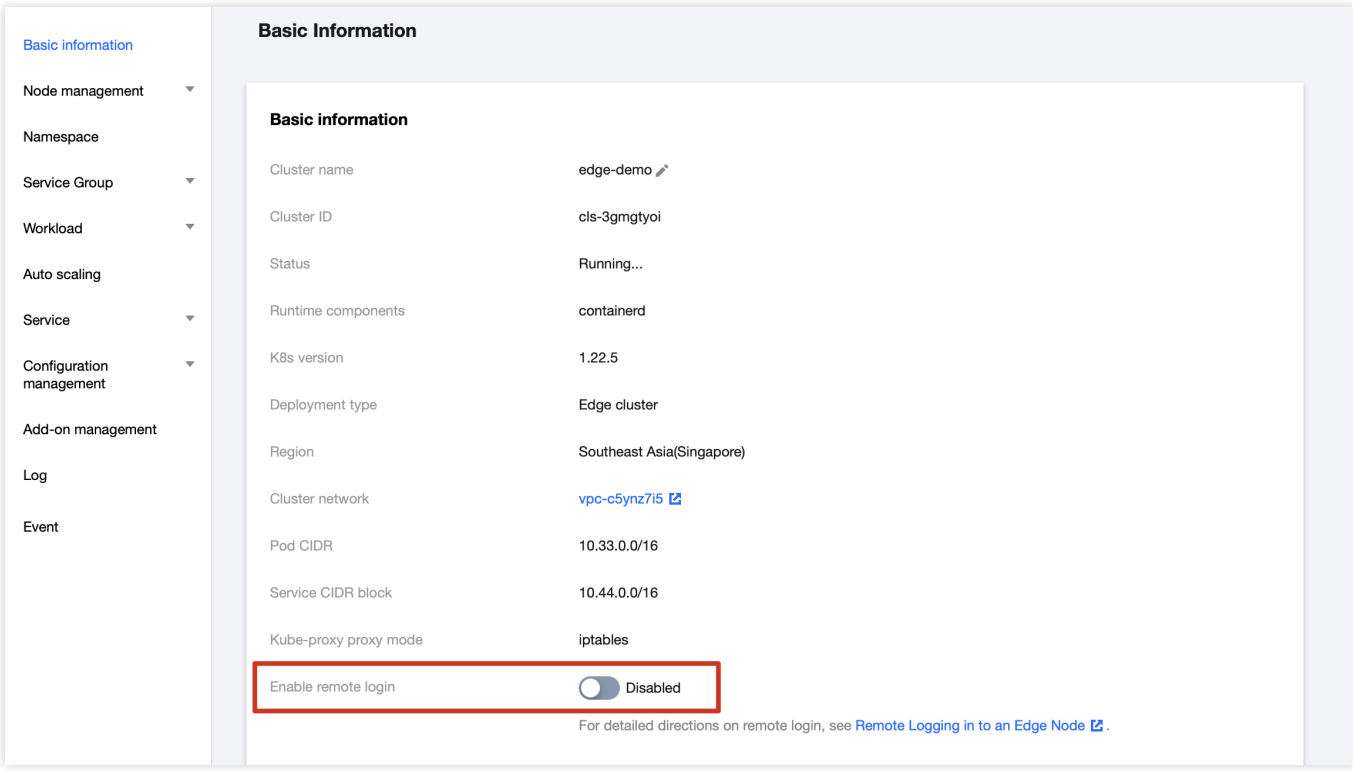


Close node remote login

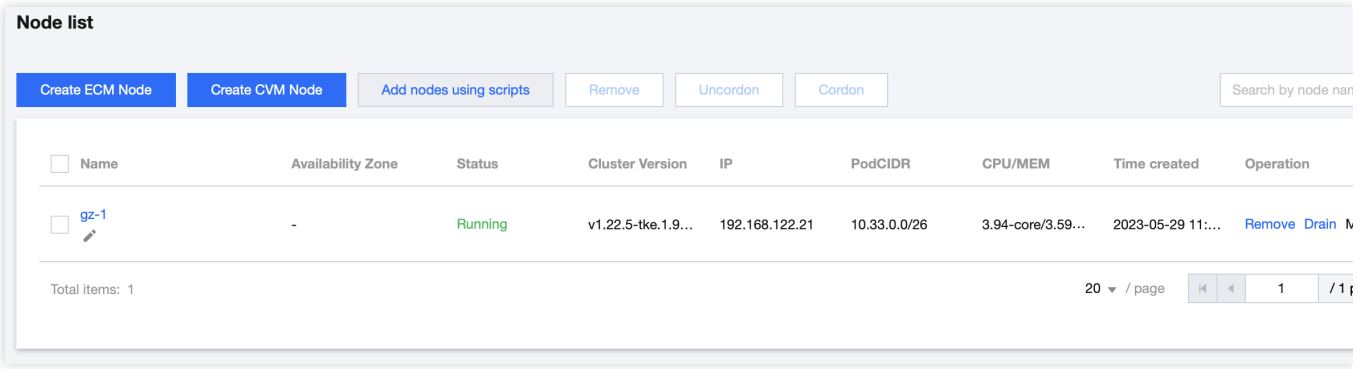
1. Log into the Tencent Cloud Container Service console.
2. On the Cluster Management page, click on the cluster ID of the node you wish to remotely log into to enter the details page for that cluster.
3. Select **Node Management** > **Nodes** from the left-hand side of the page to enter the Node List page, and confirm that the **login** hyperlink is displayed on the operation column. As shown in the figure below:



4. Select Basic Information from the left-hand side of the page to enter the Cluster **Basic Information** page, click on the remote login switch to **disable remote login** feature, as shown in the figure below:



5. Return to **Node Management** > **Nodes** from the left-hand side of the page to access the Node List page again, and confirm that the **login** hyperlink is hidden. As shown in the figure below:

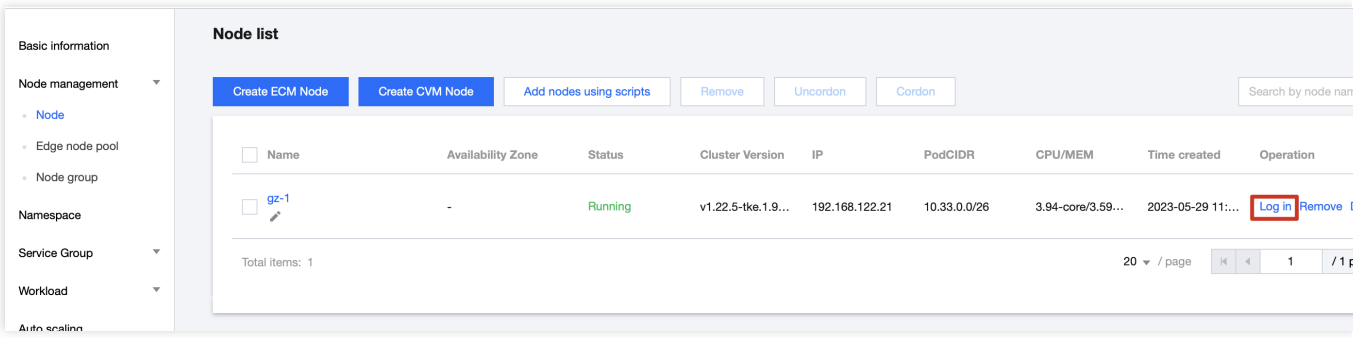


Log in to the edge node remotely

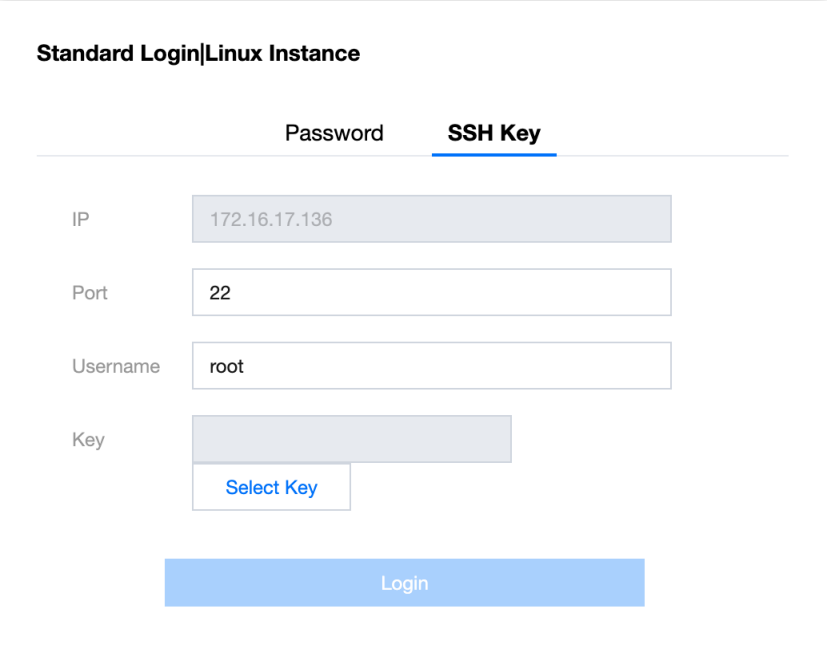
1. Log into the Tencent Cloud Container Service console.

2. On the Cluster Management page, click on the cluster ID of the node you wish to remotely log into to enter the details page for that cluster.

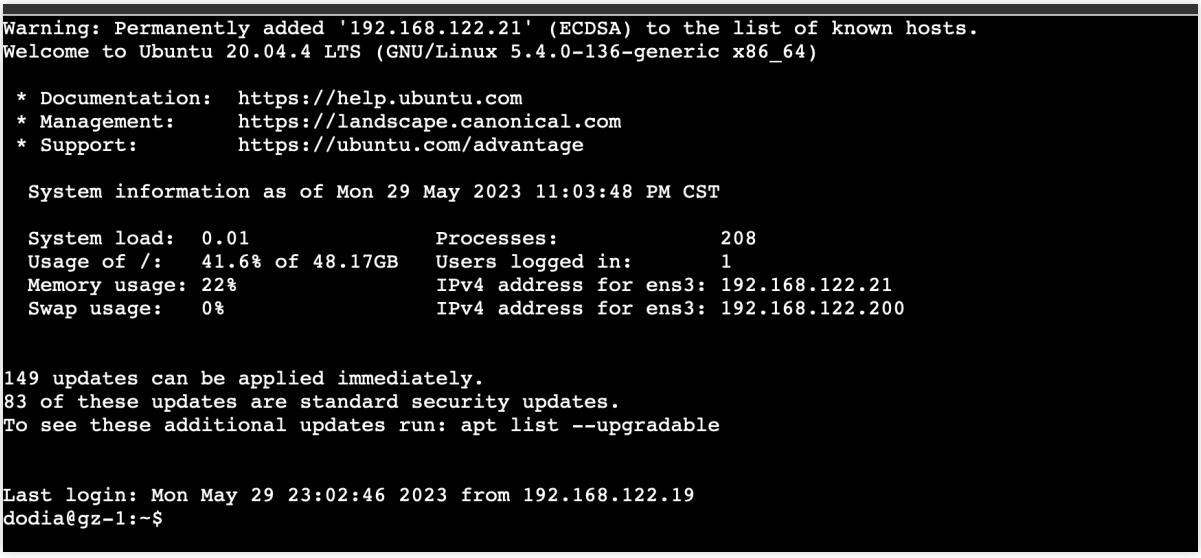
3. Select **Node Management** > **Nodes** from the left-hand side of the page to enter the Node List page, and confirm that the **login** hyperlink is displayed on the operation column. As shown in the figure below:



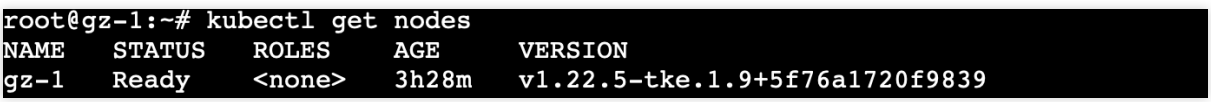
4. On the "Node List" page, select the desired node to log in and click on "**Log In**". You will be redirected to **the remote node login console** page, as shown in the figure below:



If the SSH key login is successful, you will be directed to the console page, as shown in the figure below:



Type in a command, such as 'kubectl get node', in the console and press enter to execute. View the results of the command, as shown in the figure below:



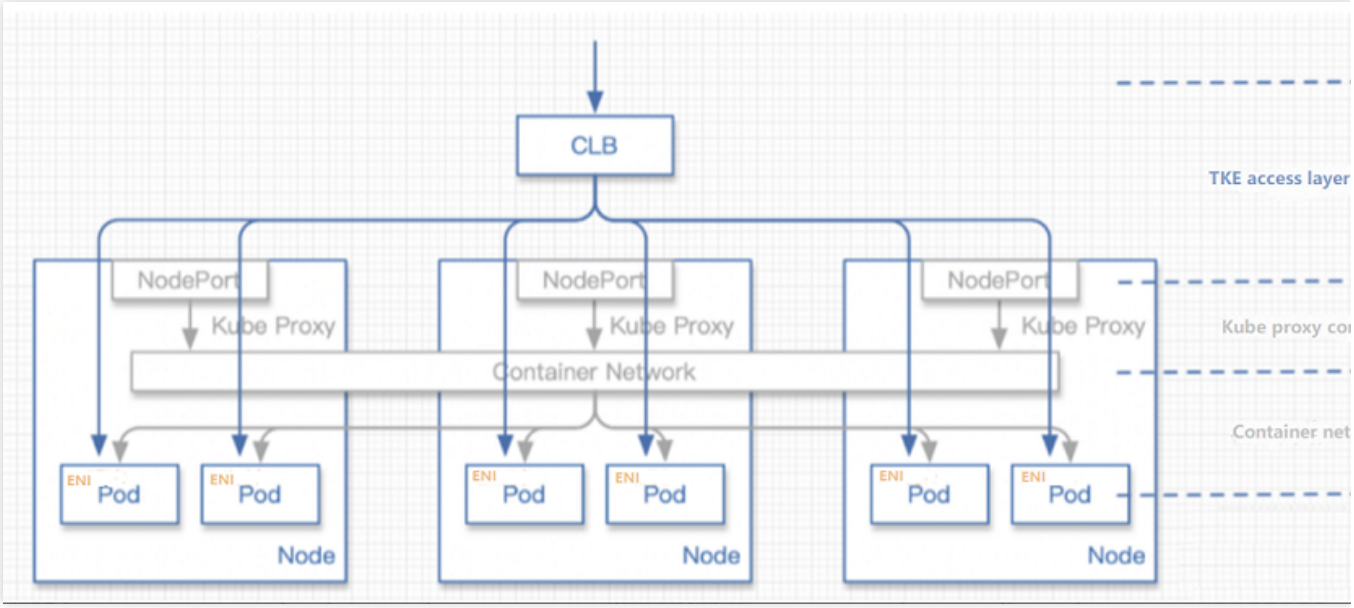
Independent ENI

Last updated：2023-06-01 11:39:14

Overview

If you add Tencent Cloud CVMs to the edge cluster as edge nodes, the platform allows you to enable ENIs and bind Pods on CVM nodes with independent ENIs to implement a high-availability network scheme.

The network architecture is as follows:



You can also use ENIs in the VPC where the CVMs reside to open Pods to external users and then bind different ENIs to the CLB to enable high-performance network forwarding.

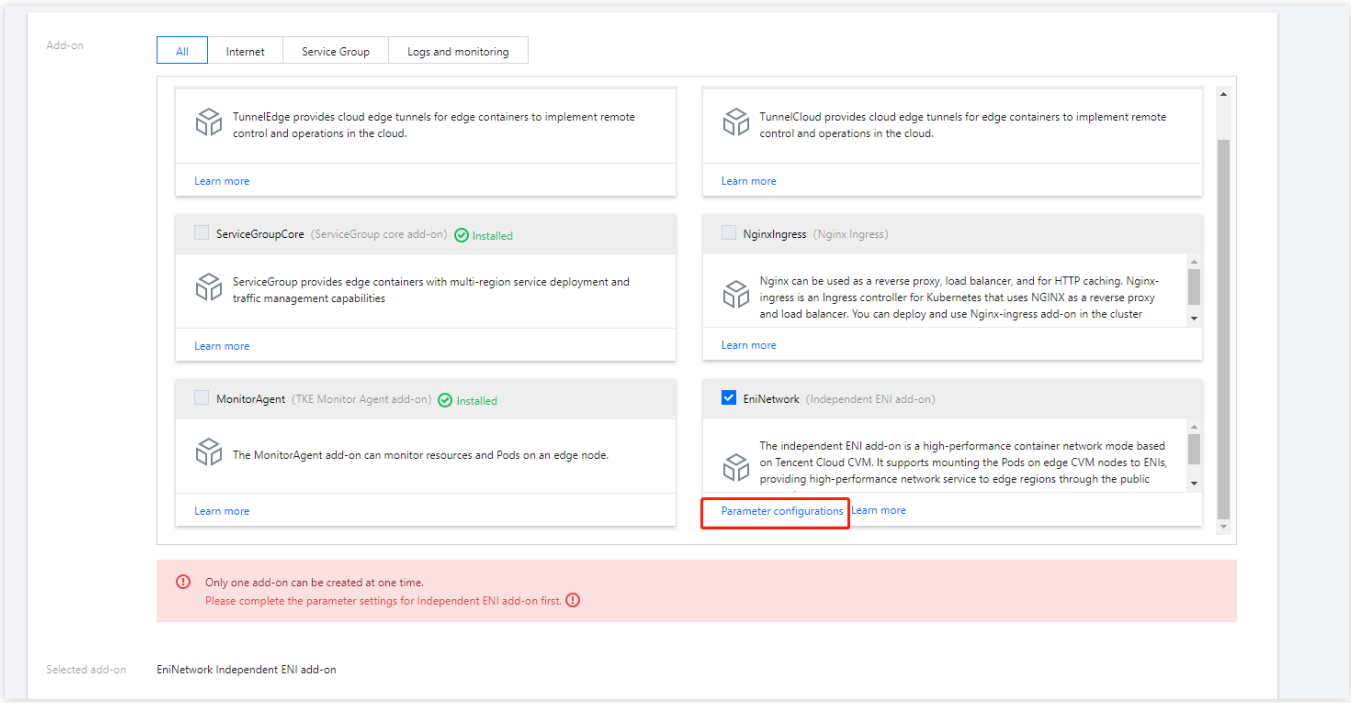
Enabling the independent ENI
Disabling the independent ENI

Directions

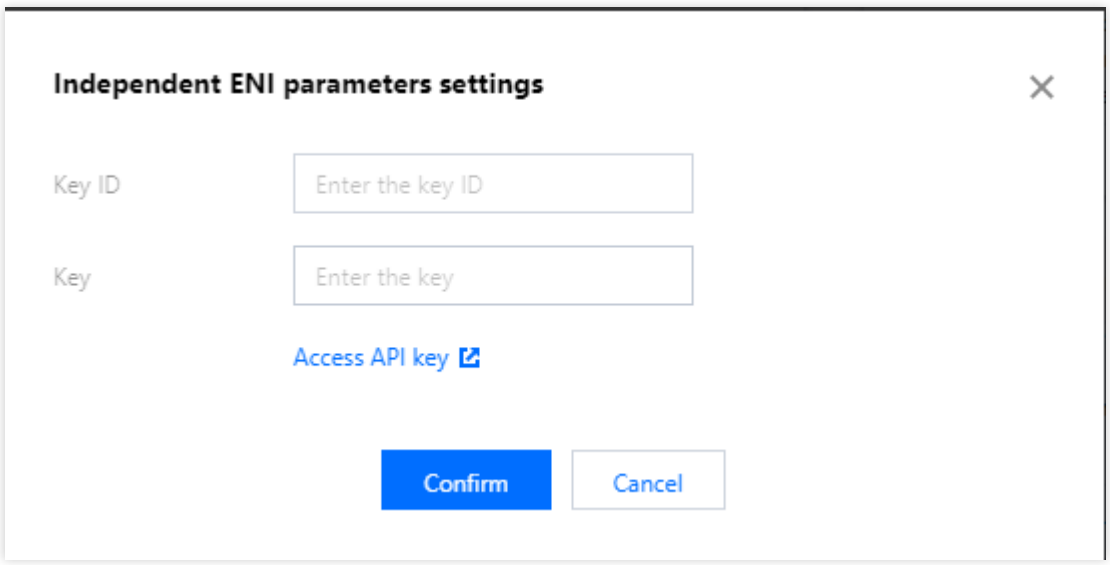
Enabling the independent ENI

1. Log in to the TKE console.

2. On the **Cluster Management** page, click the ID of the target cluster to enter its details page.
3. Select **Add-On Management** on the left of the page and click **Create** on the add-on list page.
4. On the **Create Add-On** page, select **EniNetwork (independent ENI add-on)** and click **Parameter Configurations** as shown below:

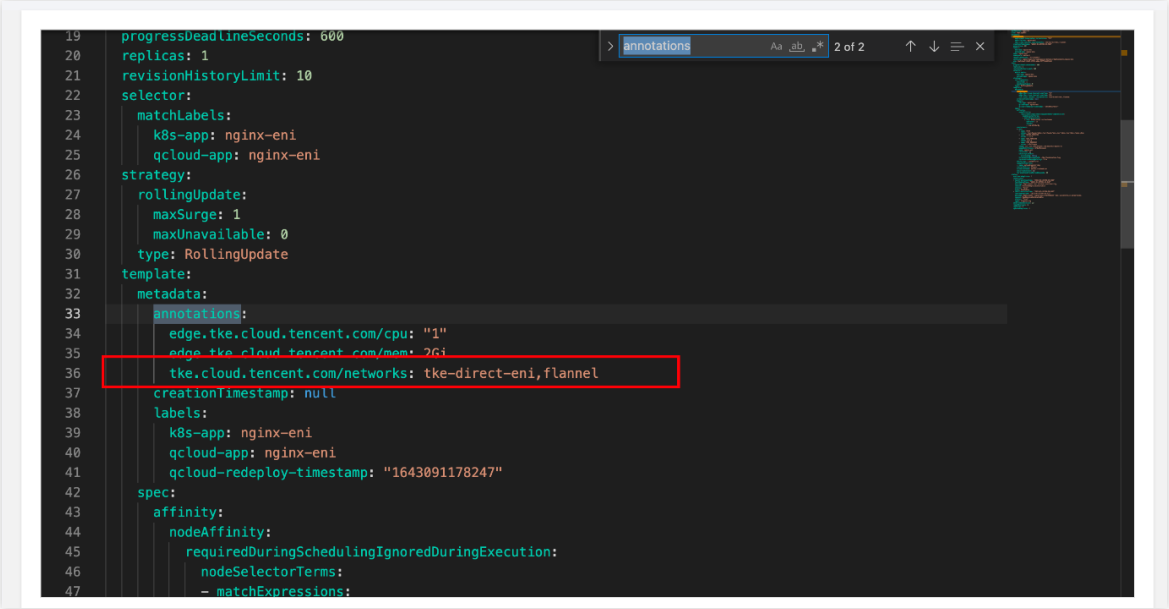


4.1 In the **Independent ENI parameter settings** pop-up window, click **Access API key** to enter the key information page.

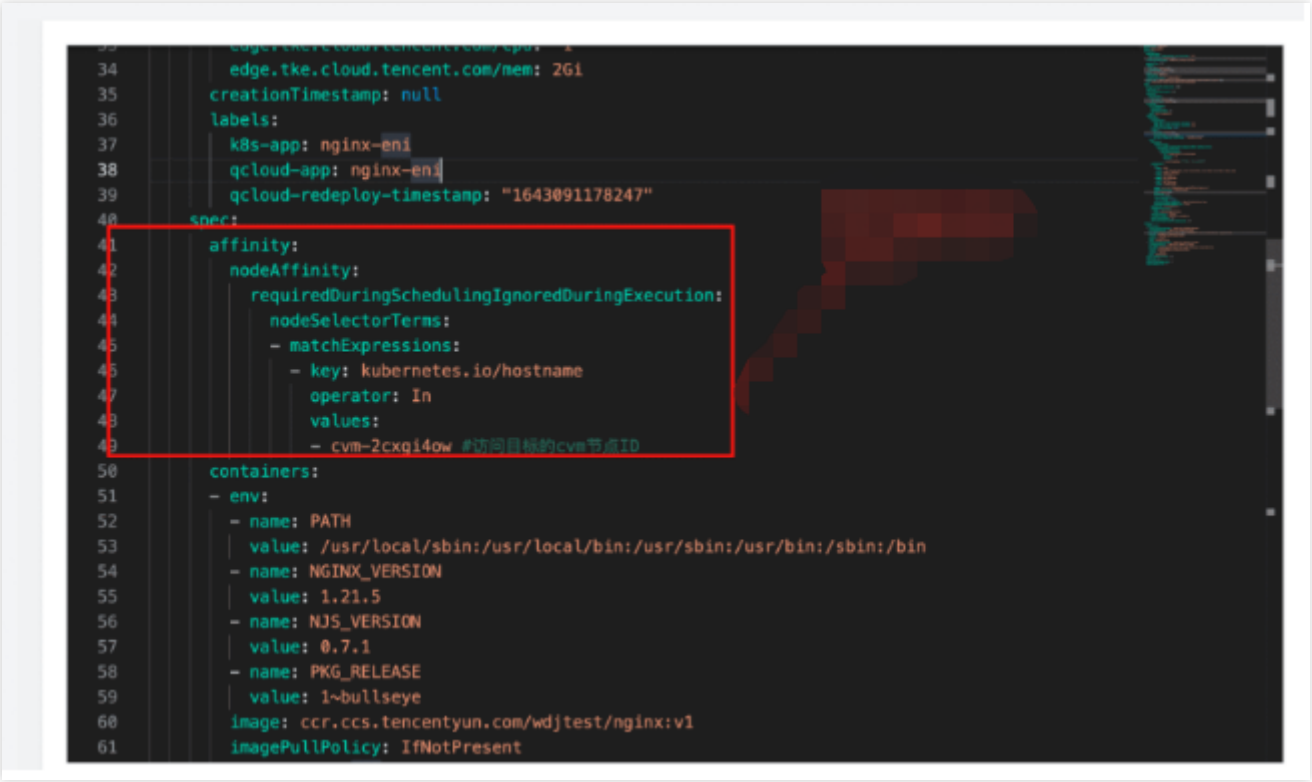


4.2 Create a key and copy the `SecretId` and `SecretKey` in API Key Management.
4.3 In the **Independent ENI parameter settings** pop-up window, enter `SecretId` and `SecretKey` and click **Confirm**.

5. On the **Create Add-On** page, click **Done** to enable the ENI.
6. Select **Workload** > **Deployment** on the left of the page to enter the Deployment list page. If there is already a Deployment in the list, skip this step; otherwise, create one as instructed in Deployment Management.
7. Select **Node Management** > **Node** on the left of the page to enter the node list page. If there is already a CVM node in the list, skip this step; otherwise, create one.
8. Configure the ENI in the Pod of the target edge cluster.
The configuration is as follows:

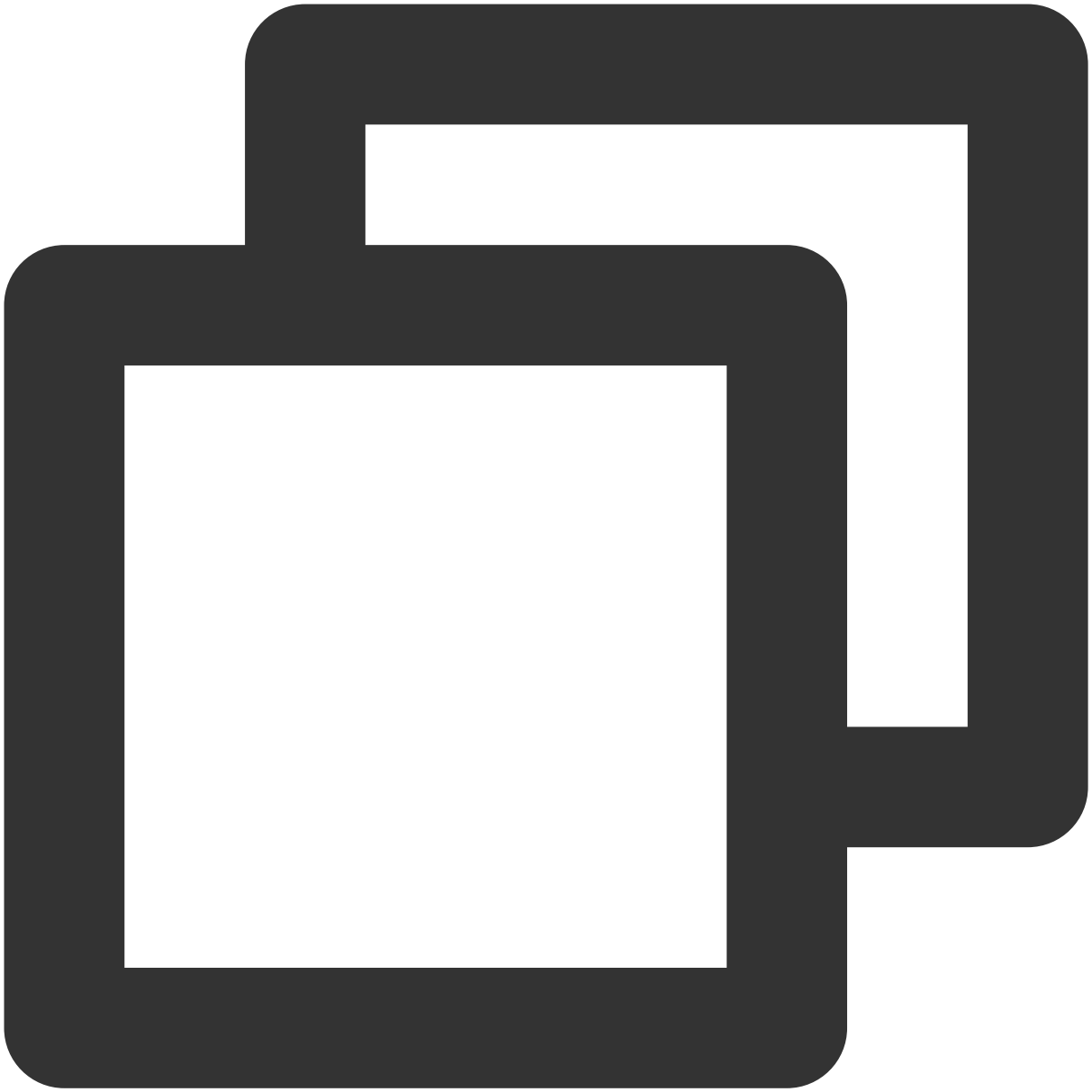


The ENI capability of edge clusters is supported only by Tencent Cloud CVM node resources. Therefore, when deploying an application, you need to use the `nodeAffinity` capability to schedule the Pod mounted with the ENI to the real CVM edge node (you can enter multiple CVM node IDs).

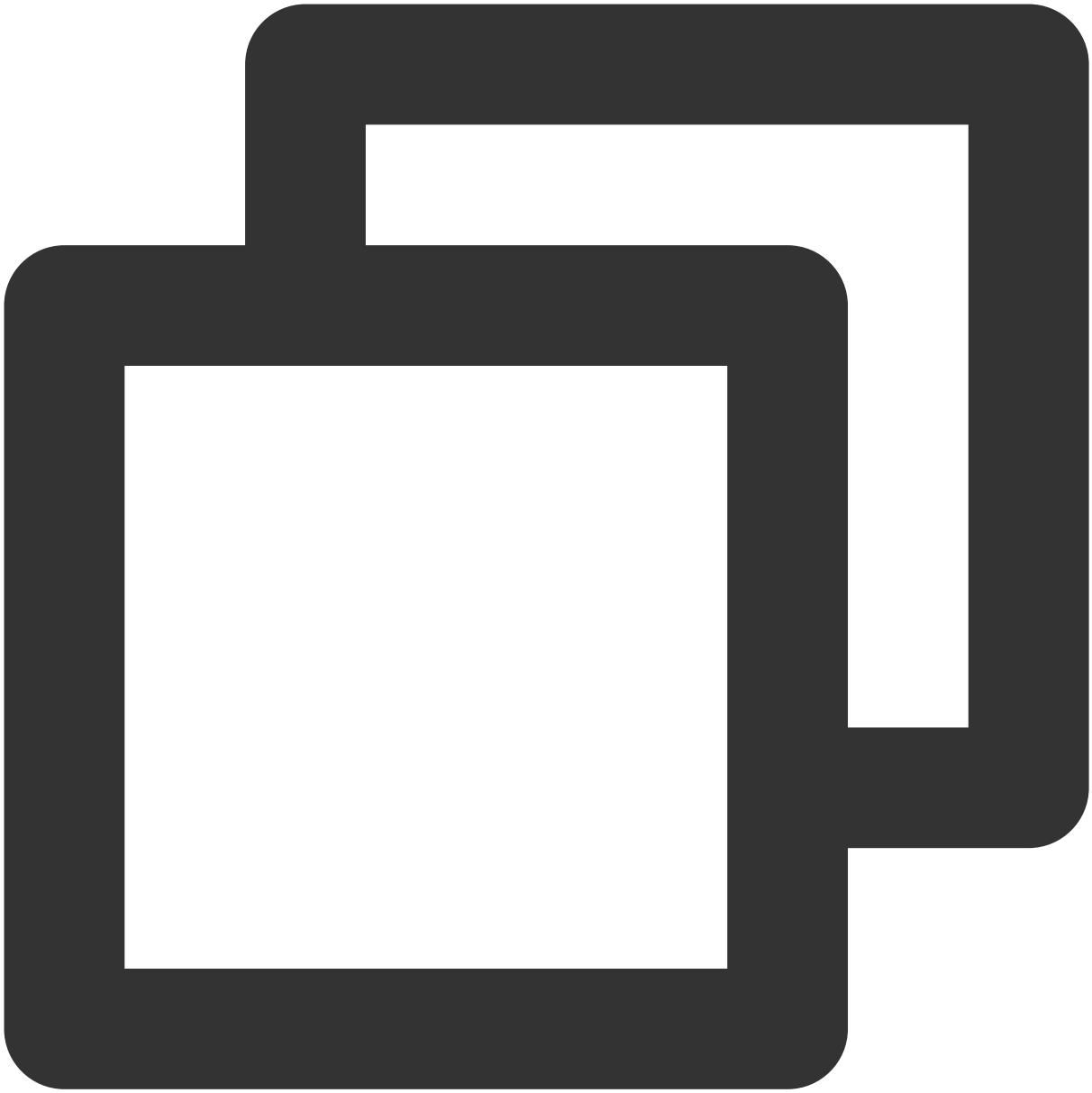


```
      edge.tke.cloud.tencent.com/mem: 2Gi
    creationTimestamp: null
    labels:
      k8s-app: nginx-eni
      qcloud-app: nginx-eni
      qcloud-redeploy-timestamp: "1643091178247"
  spec:
    affinity:
      nodeAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          nodeSelectorTerms:
          - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
              - cvm-2cxqi4ow #访问目标的cvm节点ID
    containers:
    - env:
      - name: PATH
        value: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
      - name: NGINX_VERSION
        value: 1.21.5
      - name: NJS_VERSION
        value: 0.7.1
      - name: PKG_RELEASE
        value: 1~bullseye
      image: ccr.ccs.tencentyun.com/wdjtest/nginx:v1
      imagePullPolicy: IfNotPresent
```

Below is the actual code:



```
template:
  metadata:
    annotations:
      tke.cloud.tencent.com/networks: tke-direct-eni,flannel
```



```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
            - cvm-2cxgi4ow # CVM node ID of the access target
```

Disabling the independent ENI

1. Log in to the TKE console.
2. On the cluster management page, click the target cluster ID to enter its details page.
3. Select **Add-On Management** on the left of the page and click **Delete** on the right of the target add-on on the add-on list page.
4. In the **Delete Resource** pop-up window, click **Confirm**.

Edge and Multi-region Deployment
Node Pool Management
NodeUnit

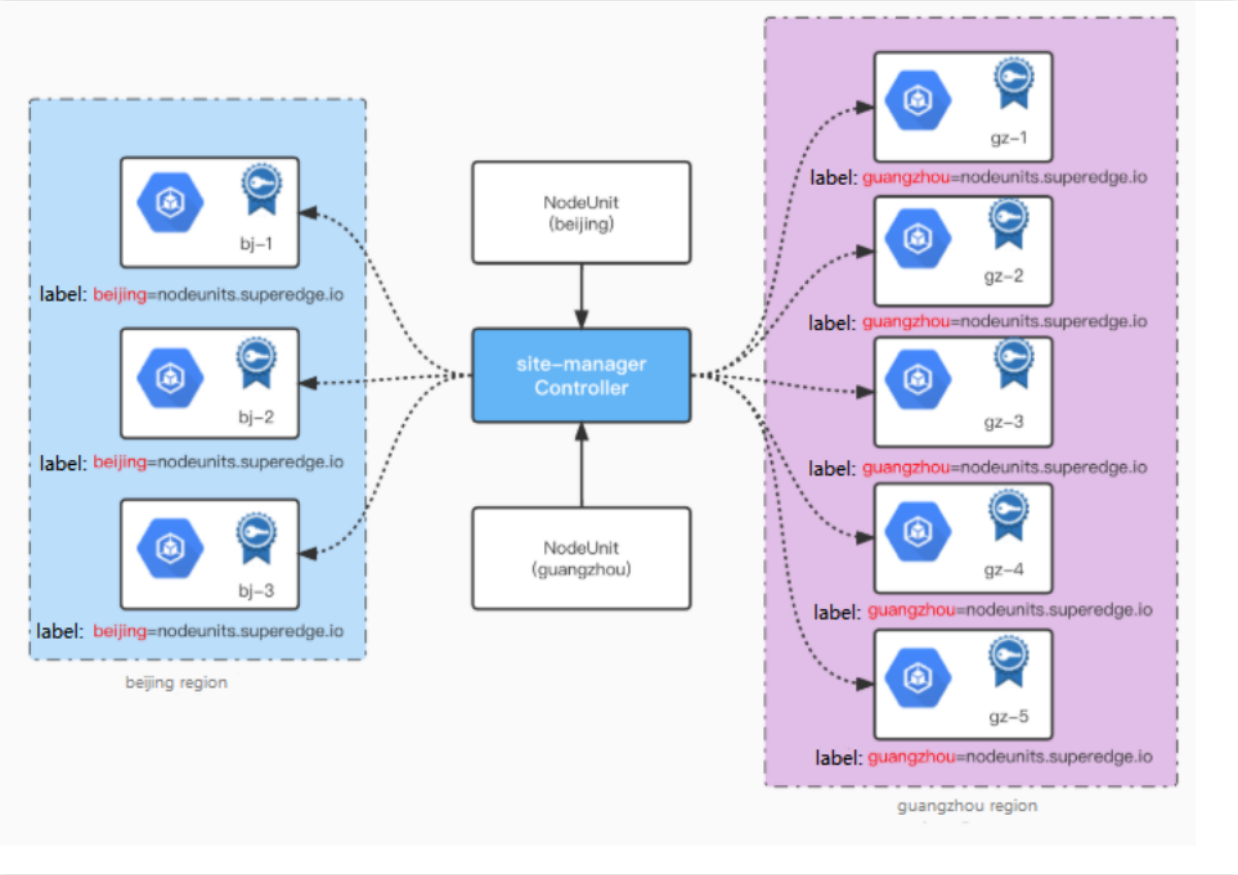
Last updated：2023-06-01 11:22:54

Operation Scenario

This document describes how to manage edge node pools of an edge container in the TKE console.
This update reconstructs the UI interaction between the earlier versions of NodeGroup and NodeUnit. Clusters created after March 29, 2022 will use the new interaction logic, and clusters on earlier versions will not be affected. In this update, the edge node pool corresponds to the design of NodeUnit in SuperEdge, and node pool category corresponds to the design of NodeGroup in SuperEdge.

Overview

In edge scenarios, nodes are assigned more attributes. For example, different nodes can be placed in different network environments, architectures, and cloud services. In some specific scenarios, users can divide nodes with the same characteristics into different groups. For example, if three nodes are located in the Beijing region and the other five nodes are located in the Guangzhou region, we can divide them into two node pools, Beijing and Guangzhou, according to their geographical attributes. Different node pools can schedule and deploy different applications. We name a node pool 'NodeUnit' resource, as a CRD resource implemented for Kubernetes. The platform interacts with users through this CRD and manages nodes in groups by labeling nodes through Kubernetes, as shown in the figure below:



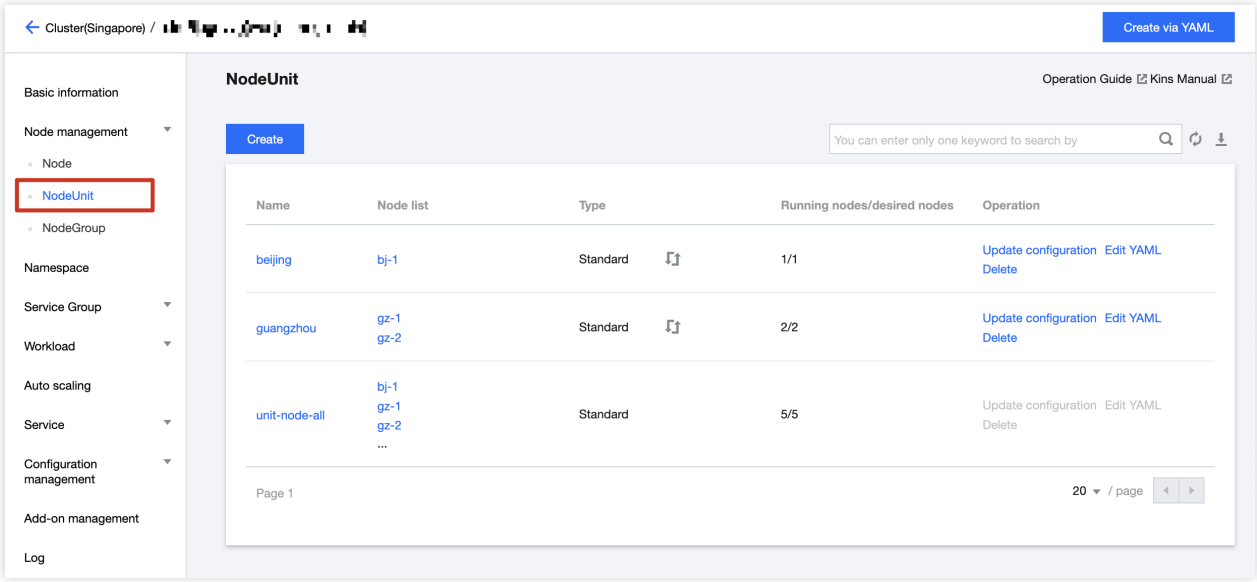
Directions

Note：

As operations on an edge node pool will affect the labels of the nodes, perform such operations with caution.

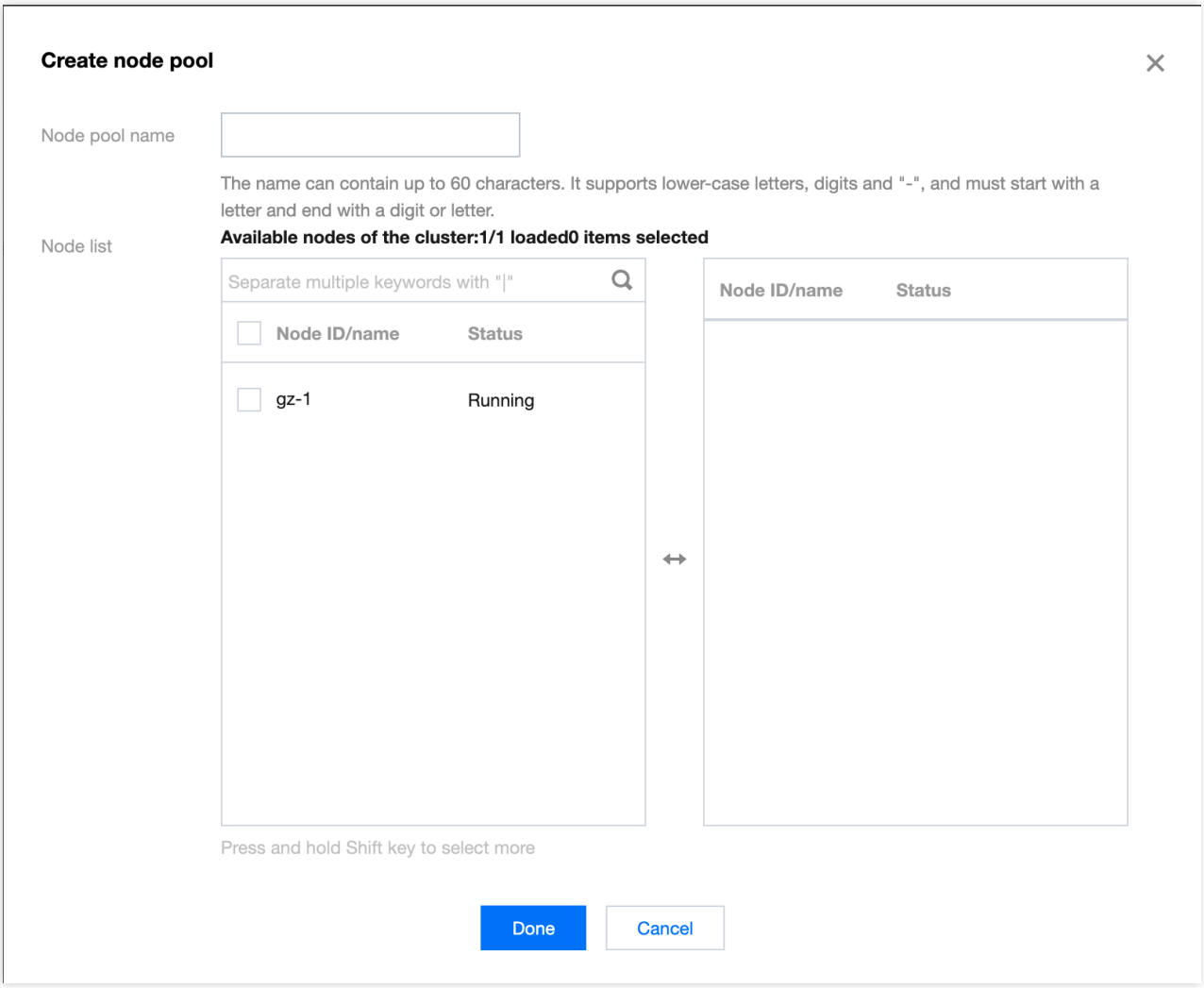
Creating an edge node pool

1. Log in to the TKE console.
2. On the **Cluster Management** page, click the ID of the target cluster to go to its details page.
3. Select **Node management > NodeUnit** in the left sidebar to go to the NodeUnit page.



Each cluster has a default node pool (`unit-node-all`), which contains all edge nodes added to the cluster.

4. Click **Create**. In the **Create edge node pool** pop-up window, enter the node pool name and add available nodes as needed.



5. Click **Done**. You can view added nodes in the edge node pool list.

Managing edge node pools

1. Log in to the TKE console.
2. On the **Cluster Management** page, click the ID of the target cluster to go to its details page.
3. Select **Node management** > **NodeUnit** in the left sidebar to go to the NodeUnit list page.
4. Click **Update configuration** on the right of an edge node pool. On the **Update NodeUnit** page, you can add and delete existing nodes.

Deleting an edge node pool

1. Log in to the TKE console.
2. On the **Cluster Management** page, click the ID of the target cluster to go to its details page.
3. Select **Node management** > **NodeUnit** in the left sidebar to go to the **NodeUnit list** page.
4. Click **Delete** on the right of an NodeUnit to delete the NodeUnit.

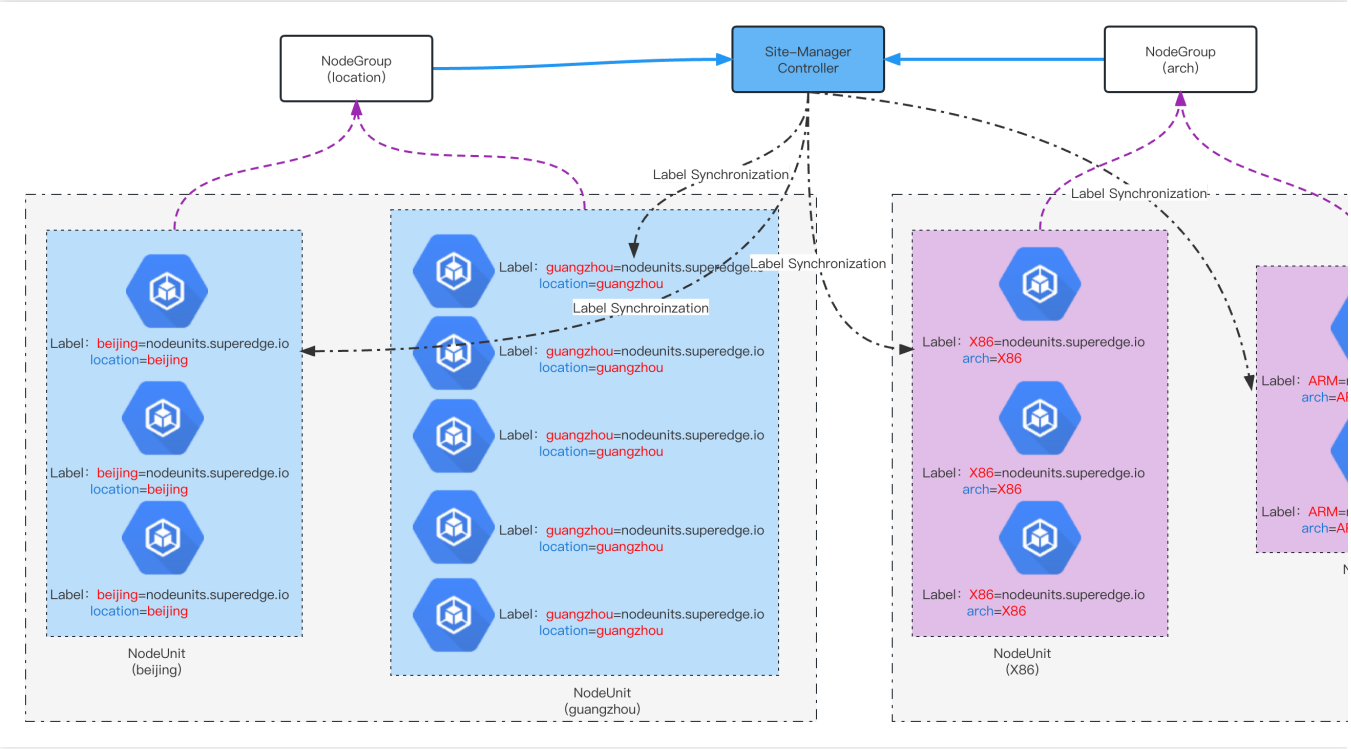
NodeGroup

Last updated：2023-06-01 11:44:35

This article introduces how to manage multiple edge node pools(NodeUnit) in the TKE-Edge through the console UI for group management.

Overview

Node pool classification(NodeGroup) allows further grouping of different node pools(NodeUnit). By adding mutually exclusive node pools to the same group, applications can be deployed in this group, enabling one-click batch deployment capabilities across multiple node pools(NodeUnit). The architecture is as following:



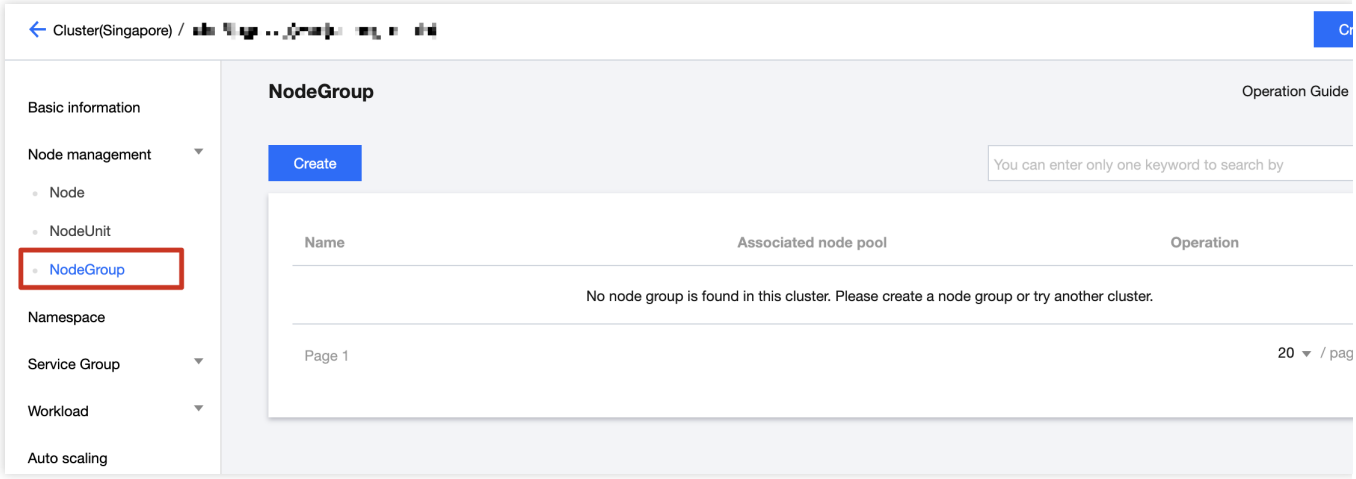
For example, you can classify the nodes in the cluster according to two types of scenarios based on your requirements. One scenario is by location, and the other scenario is by architecture. Each scenario represents a NodeGroup Custom Resource (CR) instance - location and arch. Then, you can add different NodeUnits to the corresponding NodeGroup categories.
After creating different NodeGroups based on this structure, you can use the TKE-Edge's DeploymentGrid/StatefusetGrid/ServiceGrid capabilities for multi-region deployment. For instance, you can create a DeploymentGrid for Nginx instances, bind it to the location-based NodeGroup, and both beijing and guangzhou

NodeUnits will deploy completely identical Nginx Deployment instances, providing access services across both regions simultaneously.

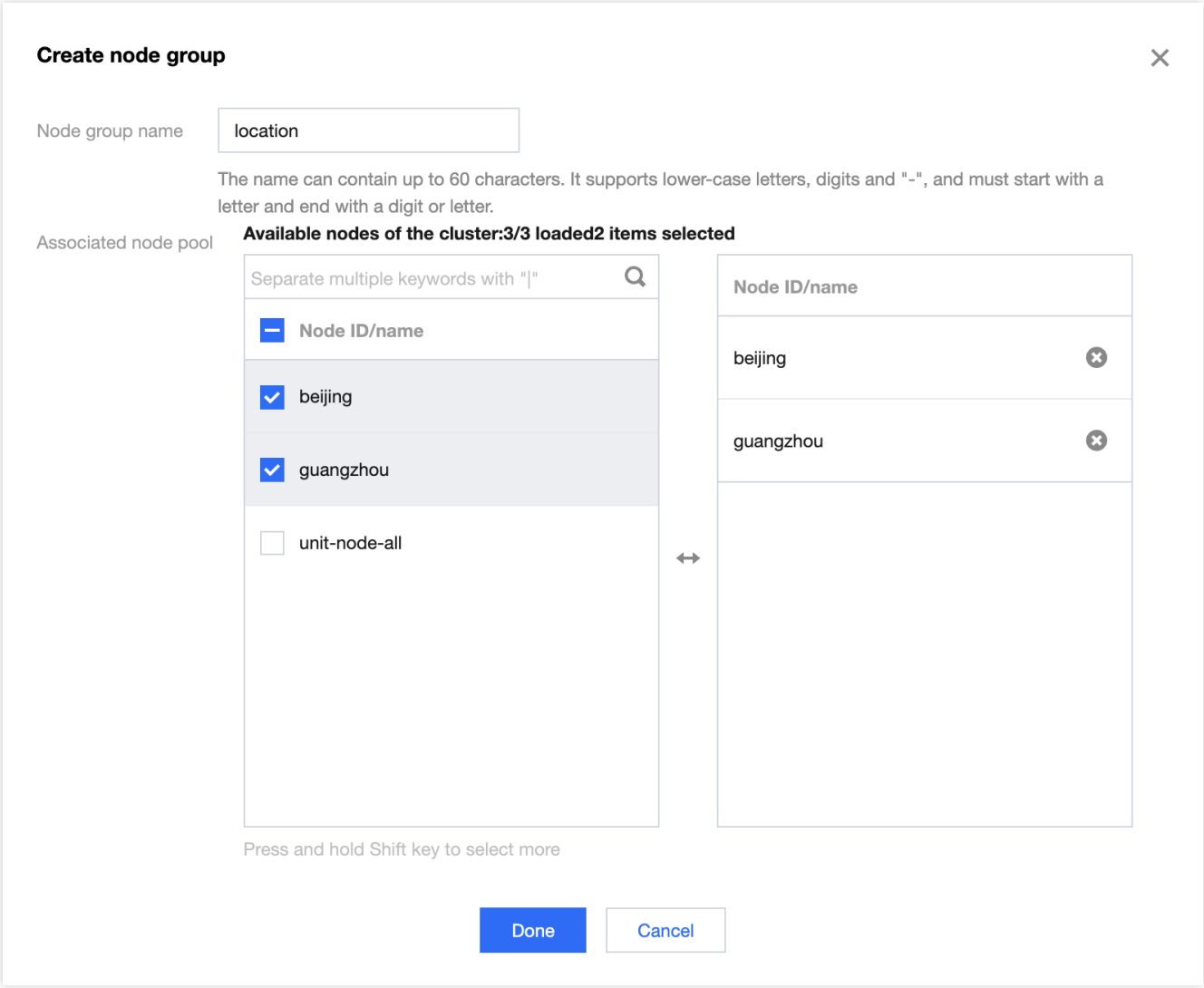
Operation Steps

Create NodeGroup

1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.
3. Select **"Node Management" > "Node group"**, enter the NodeGroup list page, as follow:



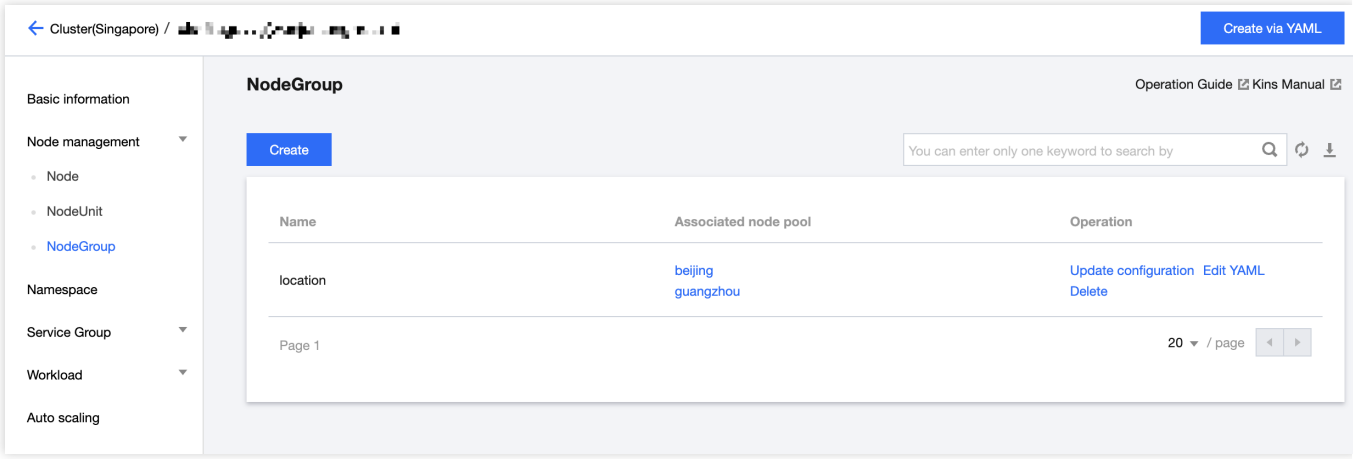
4. Click **"Create"**, select the NodeUnit name on the "Create node pool" page, and associate the NodeUnit as required.



5. Click **"Done"**, and this "location" NodeGroup will be shown in the list page.

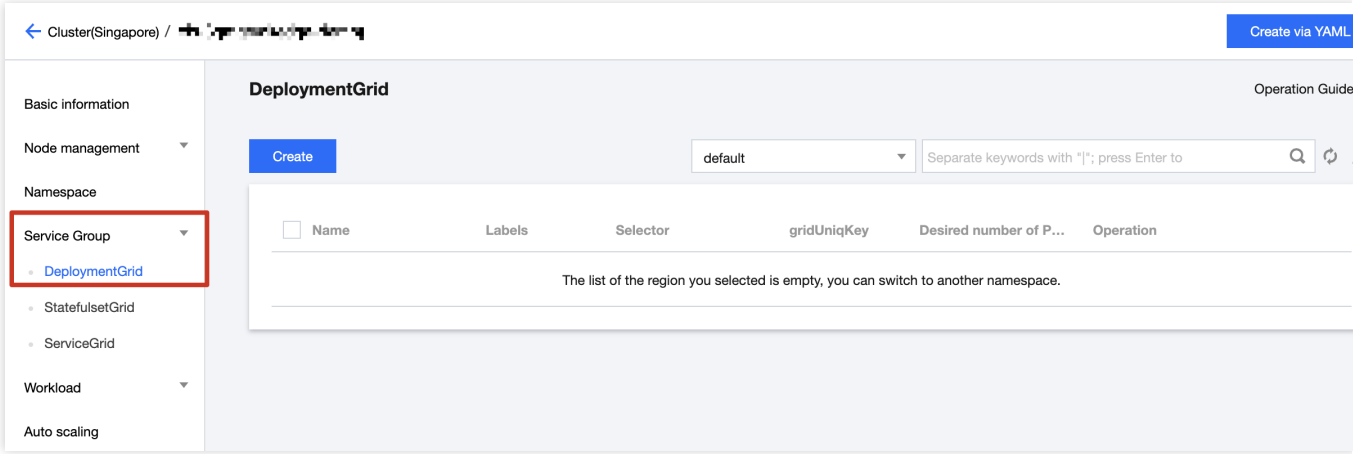
Manage NodeGroup

1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.
3. Select **"Node Management" > "Node group"**, enter the NodeGroup list page.
4. Select the "NodeGroup" you want to update, click "**Update configuration**" on the right side and you can modify the NodeGroup configuration as you want.

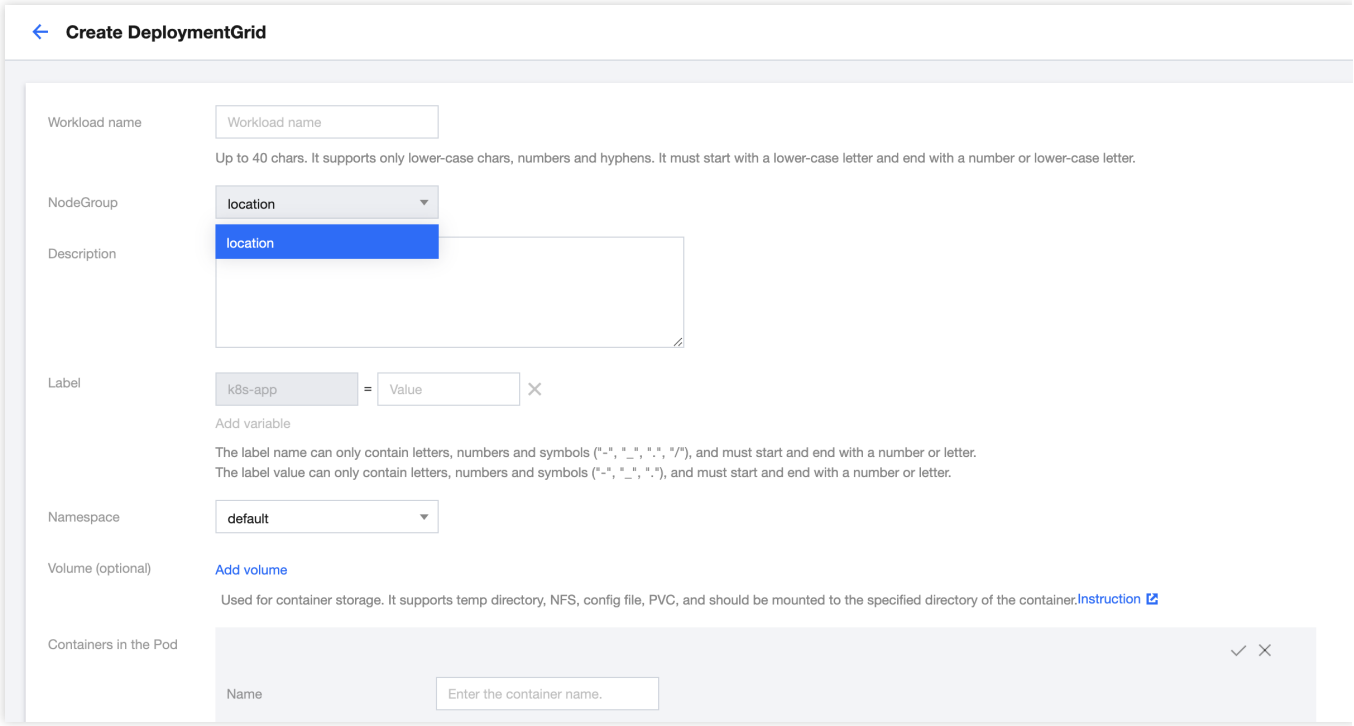


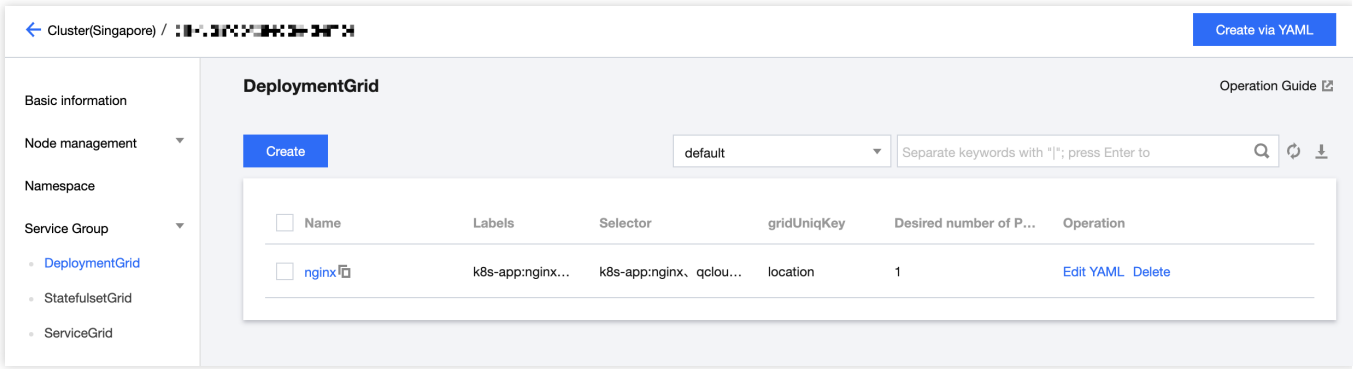
How to use NodeGroup

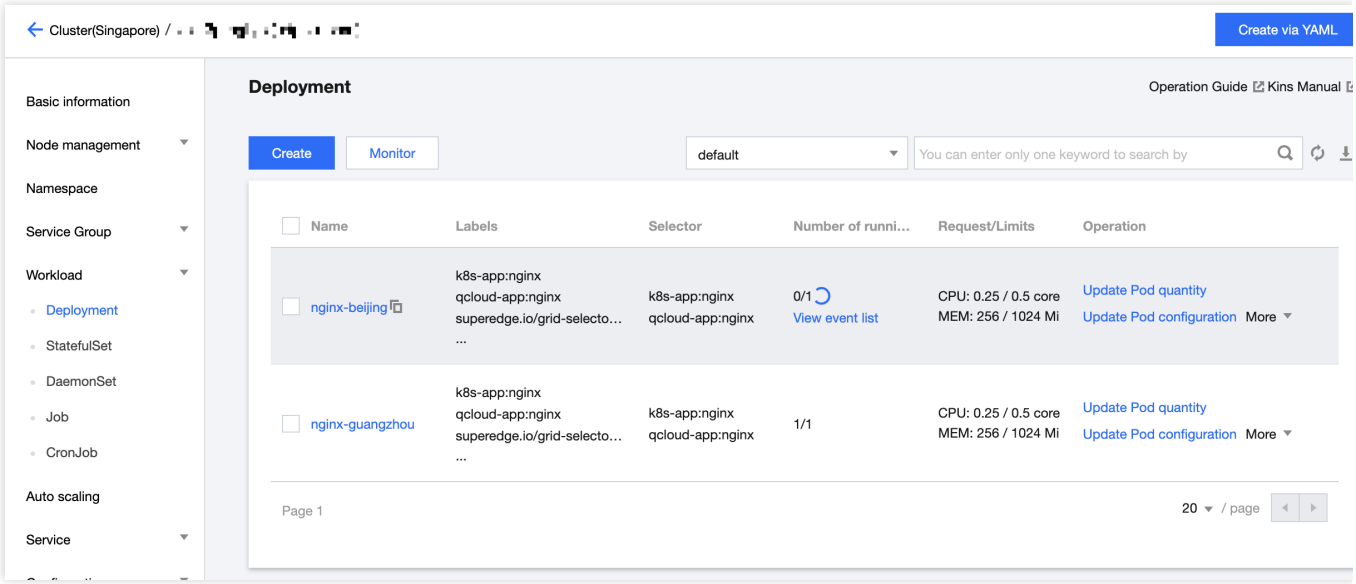
1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.
3. Select **"Service Group"** > **"DeploymentGrid"**, step into the list page.



4. Click **"Create"**, and input the parameters on the "Create DeploymentGrid" page as needed, select the NodeGroup as you want, for example "location". The deployment will be automaticly created on all corresponding NodeUnit:







ServiceGroup Manual

Last updated：2023-07-14 18:37:26

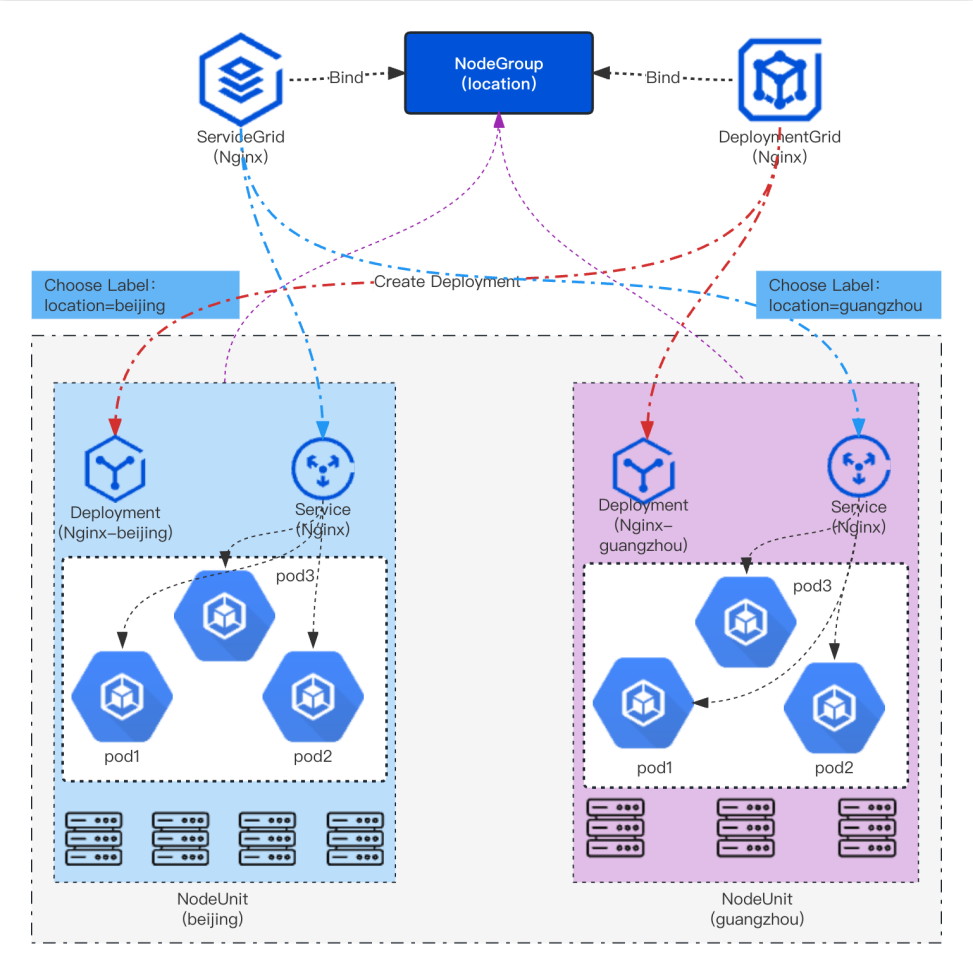
Features

In edge computing scenarios, it is common to manage multiple edge sites within the same cluster, with each edge site containing one or more computing nodes.
It is also desirable to run a set of interconnected services within each site, with the services within each site providing a complete set of features for users.
Due to network constraints, interconnected services may not want or be able to access each other across sites.
Considering these three characteristics of edge computing, TKE-Edge has specifically designed a custom resource logic called **ServiceGroup** to address application distribution and service governance issues encountered in multi-regional scenarios.

Scenarios

ServiceGroup allows for conveniently deploying a set of services within different data centers or regions that belong to the same cluster. It ensures that requests between services can be completed within the same data center or region, avoiding cross-regional access.
Native Kubernetes cannot directly control the specific node locations where Deployment Pods are created. This must be done indirectly by orchestrating node affinity planning. When the number of edge sites and the number of services to be deployed are too high, the management and deployment aspects can become extremely complex or even only theoretically feasible. Furthermore, in order to limit service interactions within a certain range, the business side needs to create dedicated Services for each Deployment, resulting in a huge and error-prone workload that may lead to online business anomalies.
ServiceGroup is designed specifically for this scenario. Users only need to use the two TKE-Edge self-developed Kubernetes resources provided by ServiceGroup, **DeploymentGrid and ServiceGrid**, to conveniently deploy services to these node groups and control service traffic. This approach also ensures the number of services and disaster recovery in each region.

Architecture



Basic Concept

ServiceGroup needs to be used in conjunction with NodeUnit and NodeGroup. The concepts are detailed as follows:

NodeUnit

NodeUnit typically consists of one or more computing resource instances located within the same edge site. It is essential to ensure that nodes within a NodeUnit have an internal network connection.
For more detail information, please refer to: NodeUnit

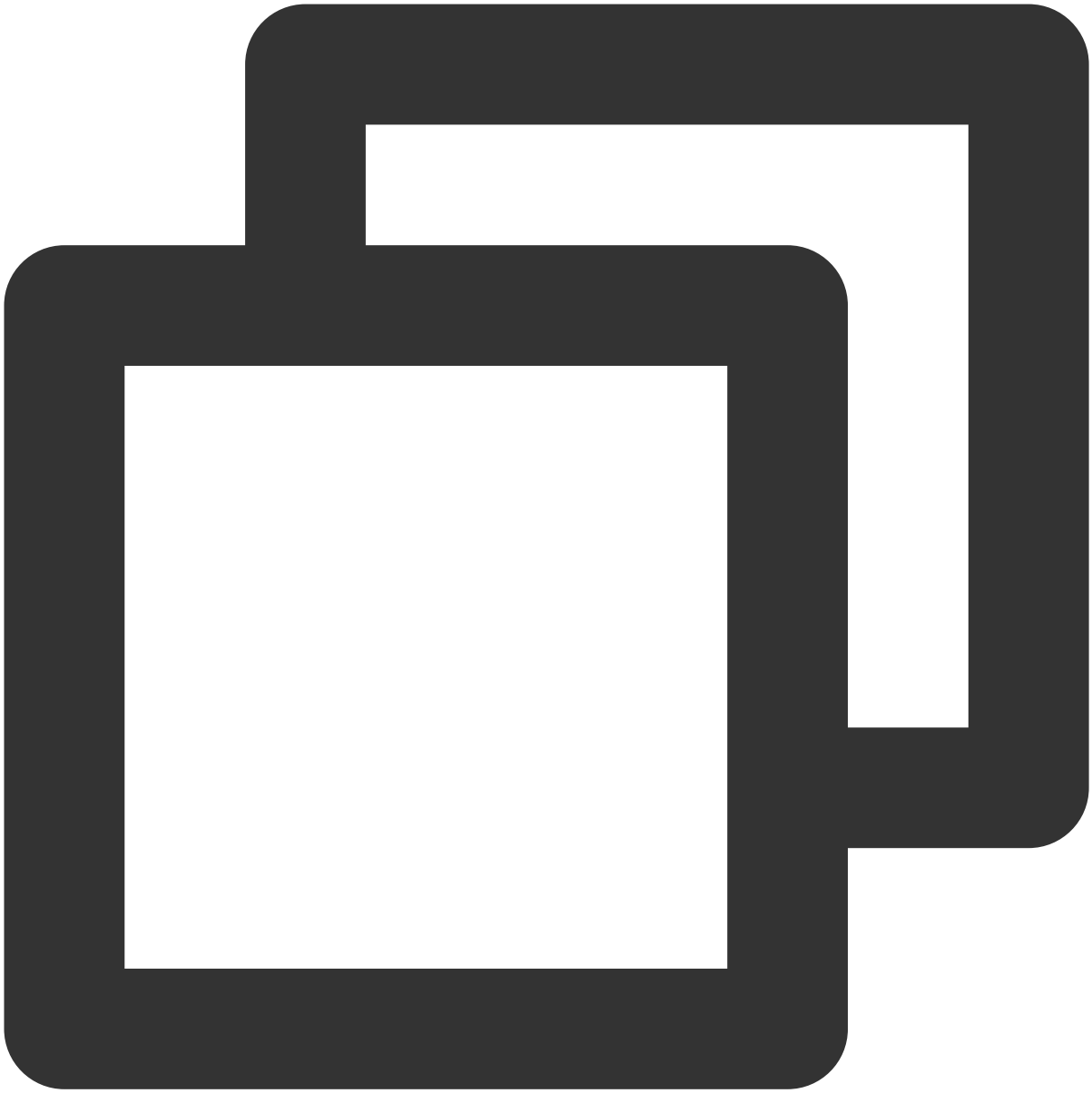
NodeGroup

Multiple NodeUnits: NodeGroup includes one or more NodeUnits.
Deploying services on every NodeUnit: It ensures that the services in the ServiceGroup are deployed on every NodeUnit within the group.
Automatic deployment to new NodeUnits: When a NodeUnit is added to the cluster, the services within the ServiceGroup are automatically deployed to the newly added NodeUnit.

For more detail information, please refer to: NodeGroup

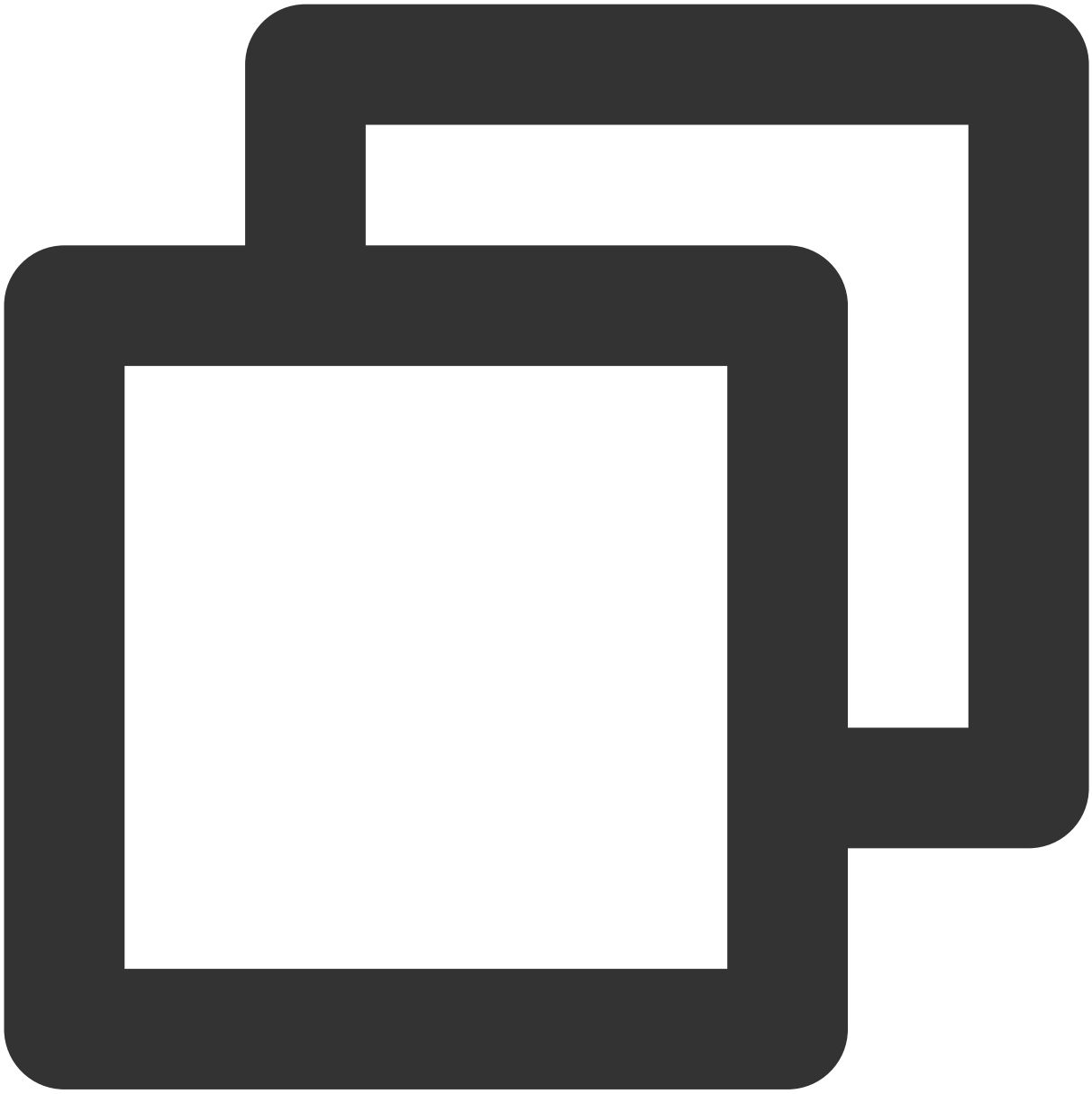
ServiceGroup

ServiceGroup is not a definition of a concrete resource, but rather a collection of Kubernetes custom resources. It is an abstract resource, and multiple ServiceGroups can be created within a cluster.
A ServiceGroup contains one or more business services and is applicable to the following scenarios:
Business services need to be packaged for deployment.
Business services need to run in each NodeUnit and ensure a specific number of pods.
Business services need to control the interactions between services within the same NodeUnit, and cannot forward traffic to other NodeUnits.
The resource types involved in ServiceGroup include the following three categories:
DeploymentGrid
StatefulSetGrid
ServiceGrid
The format of DeploymentGrid is similar to Deployment. The `<deployment-template>` field corresponds to the original `template` field of the deployment. What is special about DeploymentGrid is the `gridUniqKey` field, which specifies the key value of the label for node grouping:



```
apiVersion: superedge.io/v1
kind: DeploymentGrid
metadata:
  name:
  namespace:
spec:
  gridUniqKey: <NodeLabel Key>
  <deployment-template>
```

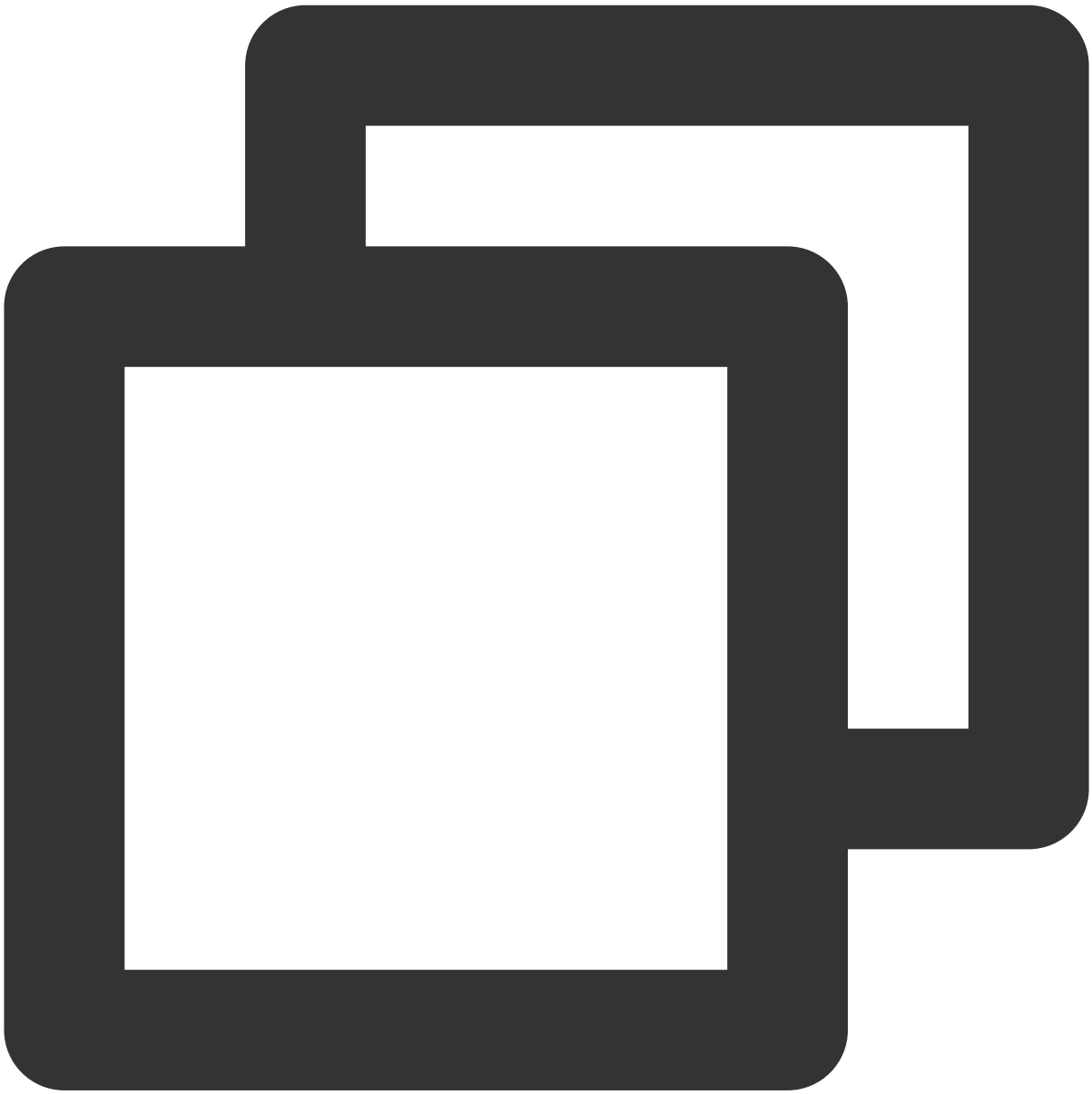
The format of StatefulSetGrid is similar to StatefulSet. The `<statefulset-template>` field corresponds to the original `template` field of the statefulset. What is special about StatefulSetGrid is the `gridUniqKey` field, which specifies the key value of the label for node grouping:



```
apiVersion: superedge.io/v1
kind: StatefulSetGrid
metadata:
  name:
  namespace:
spec:
  gridUniqKey: <NodeLabel Key>
```

```
    <statefulset-template>
```

The format of ServiceGrid is similar to Service. The `<service-template>` field corresponds to the original `template` field of the service. What is special about ServiceGrid is the `gridUniqKey` field, which specifies the key value of the label for node grouping:



```
apiVersion: superedge.io/v1
kind: ServiceGrid
metadata:
  name:
  namespace:
```

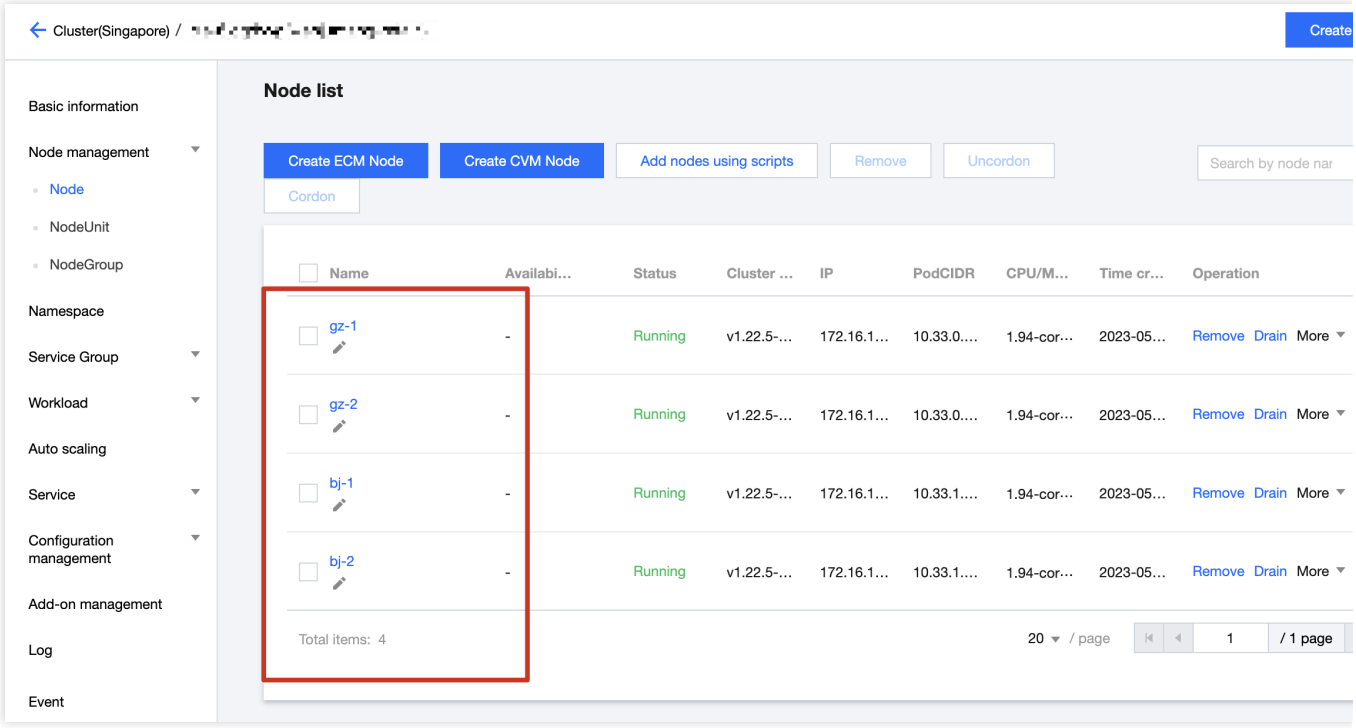
```
spec:
  gridUniqKey: <NodeLabel Key>
  <service-template>
```

Operation Steps

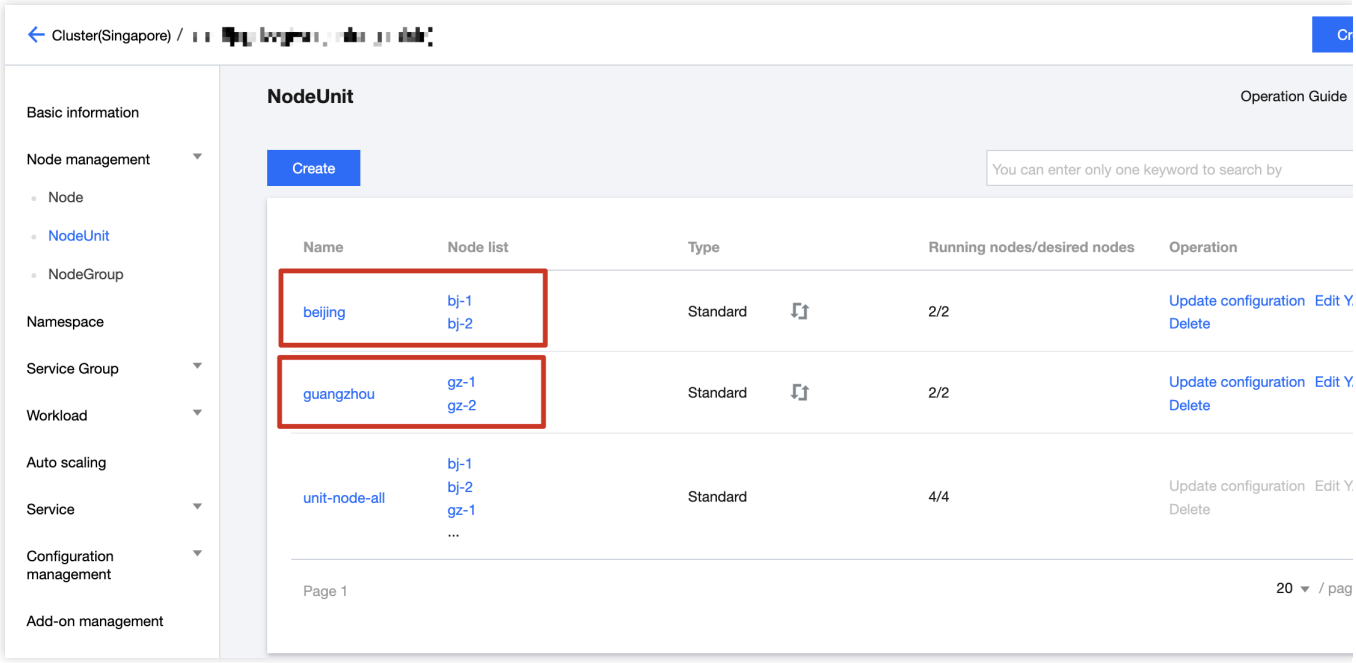
Taking the deployment of Nginx service at the edge as an example, if we want to deploy a complete set of Nginx services separately within multiple node pools, we need to perform the following operations:

Create NodeUnit and NodeGroup

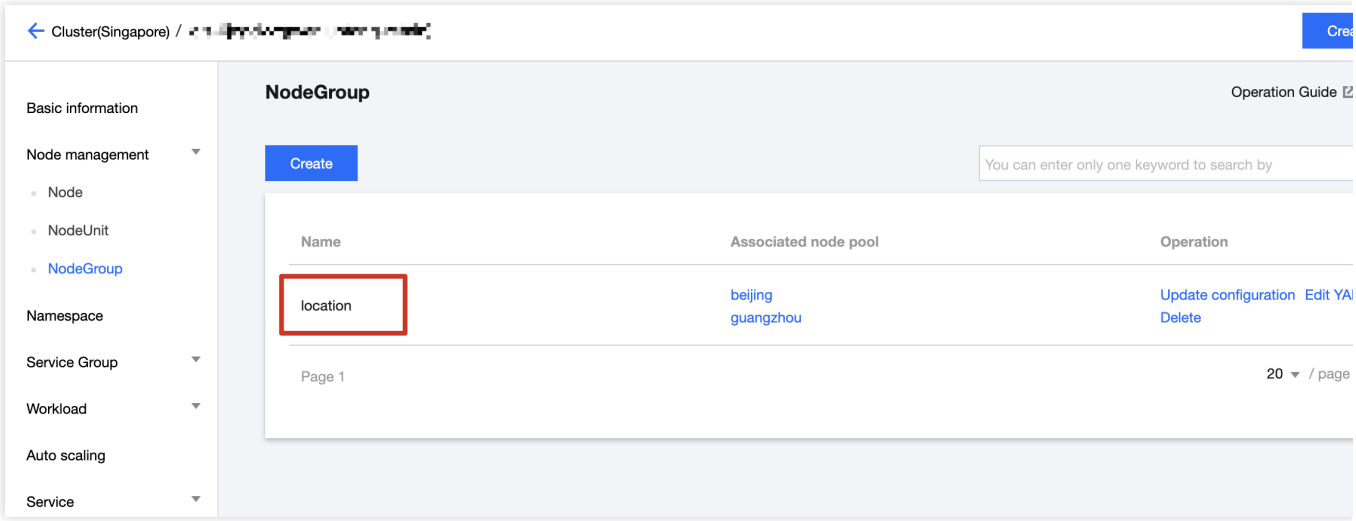
Let's take an edge cluster as an example and add the nodes within the cluster to the edge node pool（NodeUnit）and node pool classification（NodeGroup）. This cluster consists of 4 edge nodes located in 2 regions, Beijing and Guangzhou. The node names are bj-1, bj-2, gz-1, gz-2, as shown in the picture below:



Create 2 NodeUnit: beijing and guangzhou, and add corresponding nodes to the NodeUnit:



Create a NodeGroup (edge node pool classification) named location, and divide the two NodeUnit, beijing and guangzhou, into this classification, as shown in the following image:



After performing the above operations, each node will be labeled accordingly. The labels for node gz-2 are shown in the image below:

```
Name:               gz-2
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    guangzhou=nodeunits.superedge.io
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=gz-2
                    kubernetes.io/os=linux
                    location=guangzhou
                    unit-node-all=nodeunits.superedge.io
```

Description：

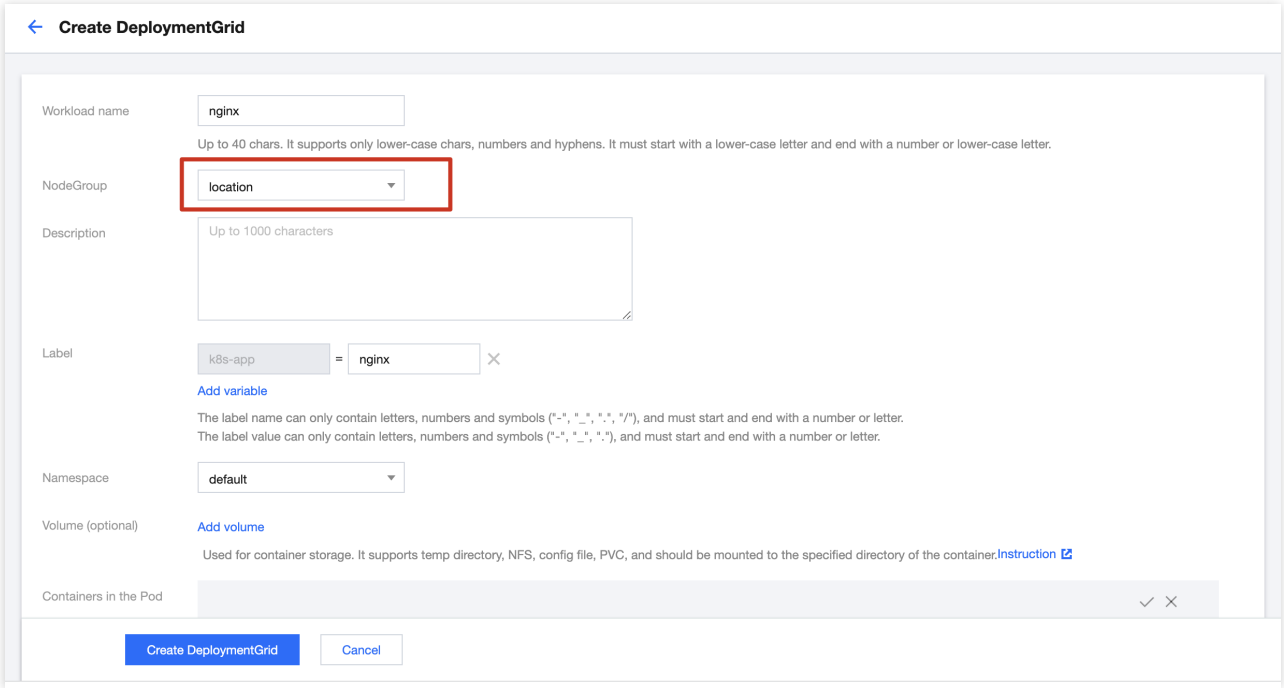
The key of the label is the name of the NodeGroup, and the value is the name of the NodeUnit. Nodes with the same value belong to the same NodeUnit.

If there are multiple NodeGroups within the same cluster, create different NodeGroup names as unique identifiers. When deploying ServiceGroup-related resources, they will be bound to the designated NodeGroup for deployment through the unique identifier of the NodeGroup name.

Stateless ServiceGroup

Create DeploymentGrid

1. Select **"ServiceGroup" > "DeploymentGrid"**
2. Click "Create", create a DeploymentGrid called "nginx"

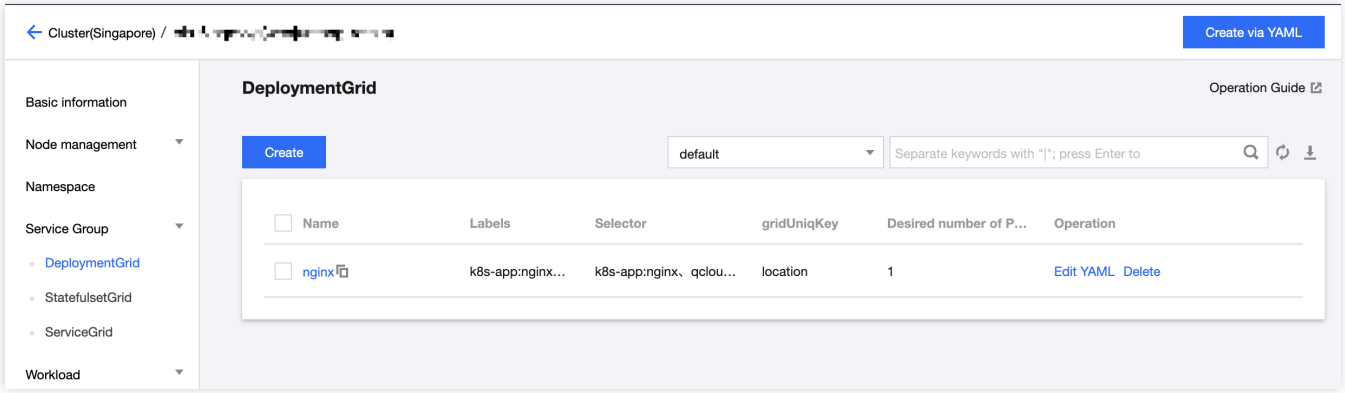


NodeGroup: Here, choose the NodeGroup group where you want to deploy the Nginx service in bulk. By selecting `location` , it means that the corresponding Deployment will be deployed under the two NodeUnits, `beijing`

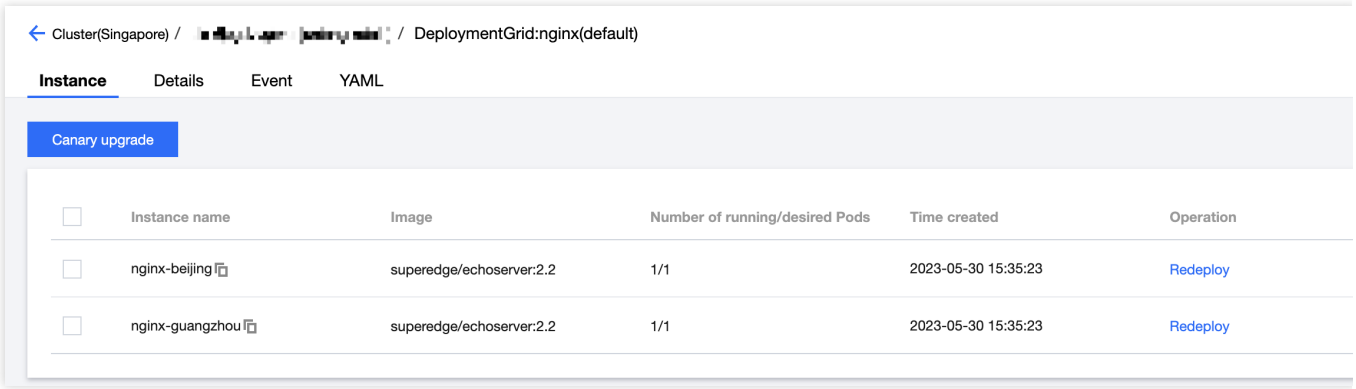
and `guangzhou` .

Other parameters: The remaining parameters are identical to the TKE application deployment method, so no further details are provided here.

3. Click "Create DeploymentGrid" and wait for finish.



4. Click on the "nginx" link to enter the detail page, where you can view the specifics of the created Deployment, as shown in the image below:

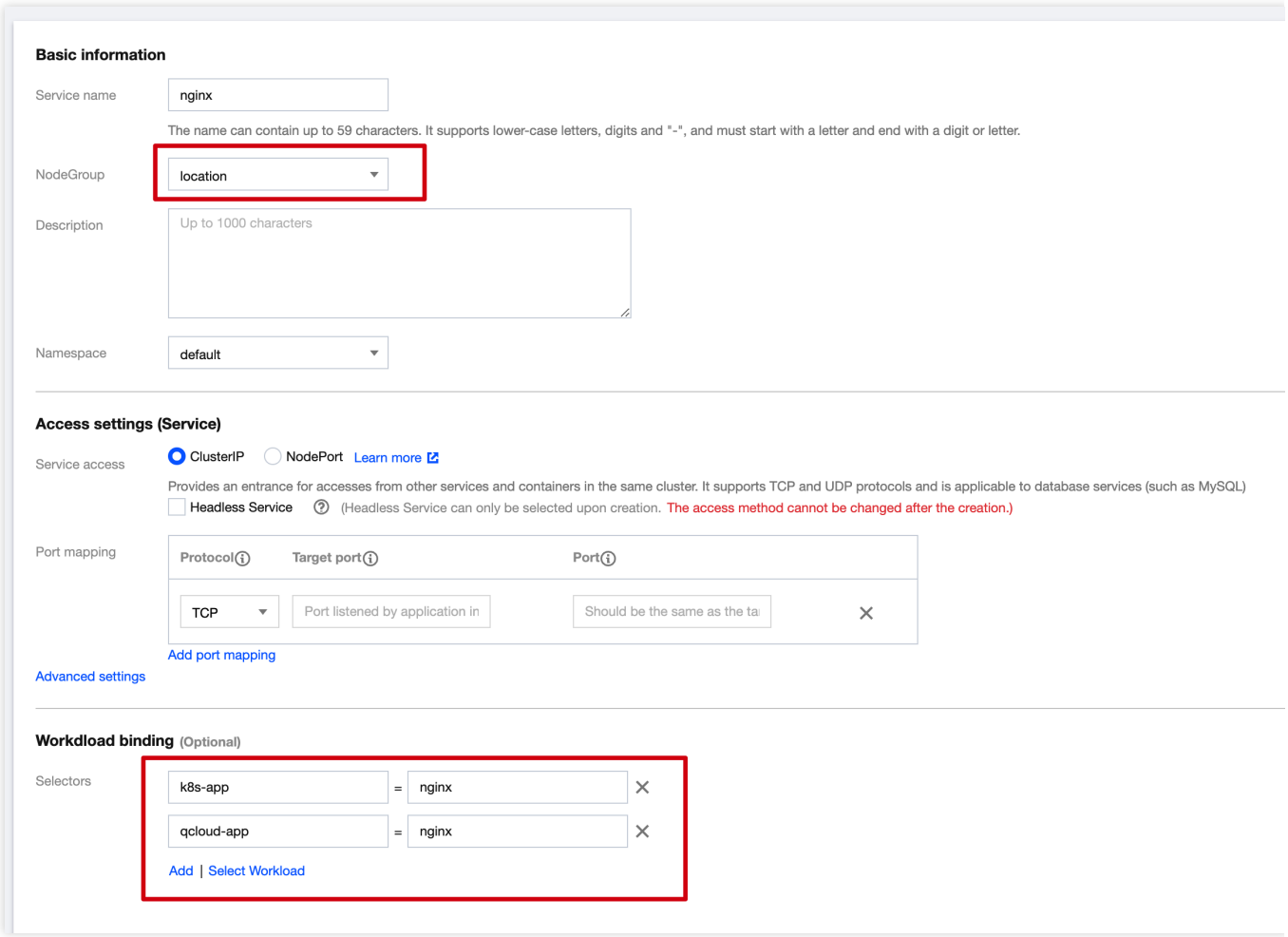


Introduction：

The platform created a standard Kubernetes Deployment under each NodeUnit contained in the NodeGroup, named as DeploymentGrid-NodeUnit. According to the example, the names here are `nginx-beijing` and `nginx-guangzhou` respectively.

Create ServiceGrid

1. Select **"ServiceGroup"** > **"ServiceGrid"**, click **"Create"**, as follow



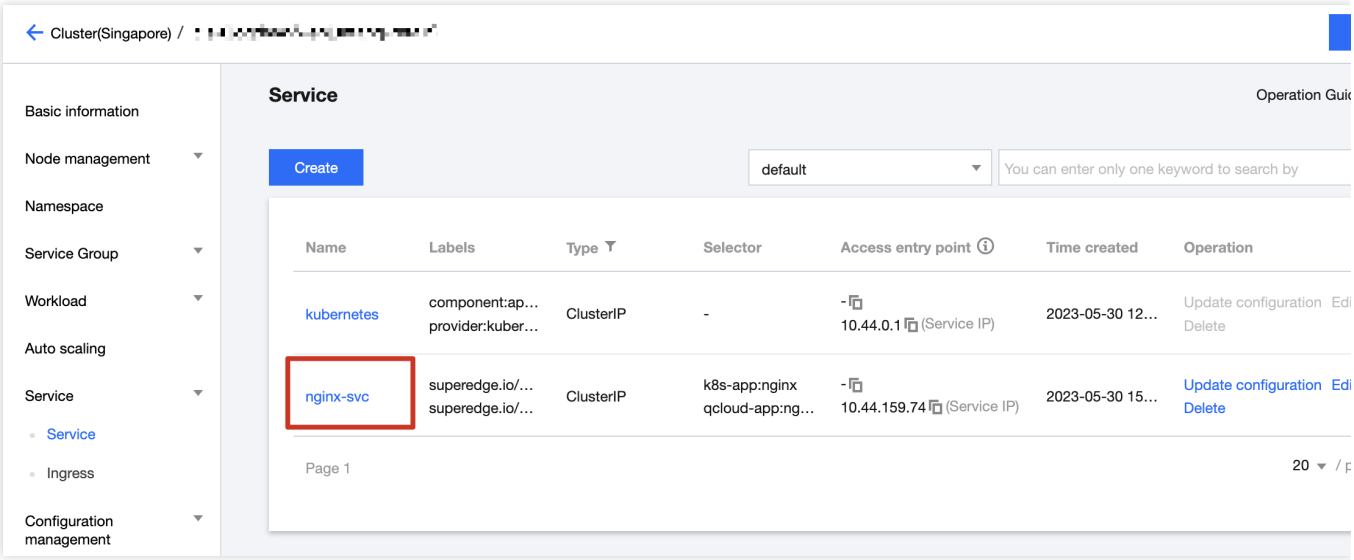
NodeGroup: Choose the required NodeGroup, which should be the same NodeGroup as the one selected for DeploymentGrid previously.

Access Settings: Standard Kubernetes Service configuration information. Choose the required port; in this case, the nginx service is set to port 80 by default.

Workload Binding: Choose the Pods required by the Service using a Selector. You can manually enter and add them, or you can select 'Reference Workload' to add the previously deployed DeploymentGrid, in this case > nginx.

2. Click **"Create ServiceGrid".** After successful creation, the event details page is displayed.

3. You can view the specific created Service information under **Service > Service**, as shown in the following image:



4. By using the **nginx-svc** Service, the following objectives can be achieved:

When accessing this Service from a Pod in the Beijing region, only the Pods within the Beijing region will be accessed by the backend.

When accessing this Service from a Pod in the Guangzhou region, only the Pods within the Guangzhou region will be accessed by the backend.

Access to this Service in each region will be restricted within the local NodeUnit scope.

Health Check

Last updated：2023-06-01 11:22:54

In edge scenarios, weak networks can trigger the Kubernetes eviction mechanism, causing unexpected pod eviction behavior. In edge computing scenarios, the network environment between edge nodes and the cloud is highly complex, the network quality cannot be guaranteed, and issues such as API Server and node connection interruptions can occur easily. Using native Kubernetes without modification may lead to frequent node status anomalies, activating the Kubernetes eviction mechanism, resulting in pod eviction, missing endpoints, and eventually causing service interruptions and fluctuations.
To address this issue, TKE-Edge introduces a distributed node status determination mechanism for the first time. This mechanism better identifies eviction timing, ensuring the system runs smoothly under weak network conditions, and avoiding service interruptions and fluctuations.

Pain Points of the Requirements

Native Kubernetes Approach

Weak cloud-edge networks impact communication between the kubelet running on edge nodes and the cloud API Server. If the cloud API Server cannot receive heartbeat signals from the kubelet or renew leases, it cannot accurately obtain the running status of the node and the pods on the node. If this lasts longer than the set threshold, the API Server will consider the node unavailable and take the following actions:
The disconnected node's status is set to NotReady or Unknown, and taints NoSchedule and NoExecute are applied.
Pods on the disconnected node are evicted and rebuilt on other nodes.
Pods on the disconnected node are removed from the Service's Endpoint list.

Solutions

Design Principles

In edge computing scenarios, relying solely on the connection between the edge node and the API Server to determine whether a node is normal is not reasonable. To make the system more robust, it's necessary to introduce additional judgment mechanisms.
Compared to the cloud and edge nodes, the network between edge nodes is more stable. Thus, a more stable infrastructure can be leveraged to improve accuracy. TKE-Edge pioneered the edge health distributed node status determination mechanism. In addition to considering the connection between nodes and the API Server, it introduces edge nodes as evaluation factors to provide a more comprehensive status assessment of the nodes. Extensive tests and practical demonstrations have proven that this mechanism improves the accuracy of node status determination

under weak cloud-edge network conditions, ensuring stable operation of services. The main principles of this mechanism are as follows:

Each node periodically probes the health status of other nodes.

All nodes within the cluster regularly vote to determine the status of each node.

Both cloud and edge nodes collectively determine the node's status.

Firstly, nodes internally probe and vote for the specific status of a node, and the unanimous judgment of the majority ensures the accurate status. Secondly, although the network status between nodes is typically better than the cloud-edge network, we should consider the complexity of the edge nodes' network environment, which is not 100% reliable. Therefore, it is not possible to fully trust the network between nodes, and the node status cannot be decided solely by the nodes themselves. A joint decision is more reliable. With this in mind, the following design is proposed:

Status Determination	**Cloud Determination Normal**	**Cloud Determination Abnormal**
Intranet Determination Normal	**Normal**	K8s shows **NotReady**, but the behavior is different from the standard K8s process: No longer scheduling new Pods to the node, but not evicting Pods and handling services.
Intranet Determination Abnormal	**Normal**	K8s shows **NotReady**, with behavior consistent with the standard K8s process: evicting existing Pods; removing them from the Endpoint list; no longer scheduling new Pods to the node.

Prerequisite

This feature requires opening port 51005 on the node so that the nodes can perform distributed smart health detection amongst themselves.

Operation Steps

Caution：

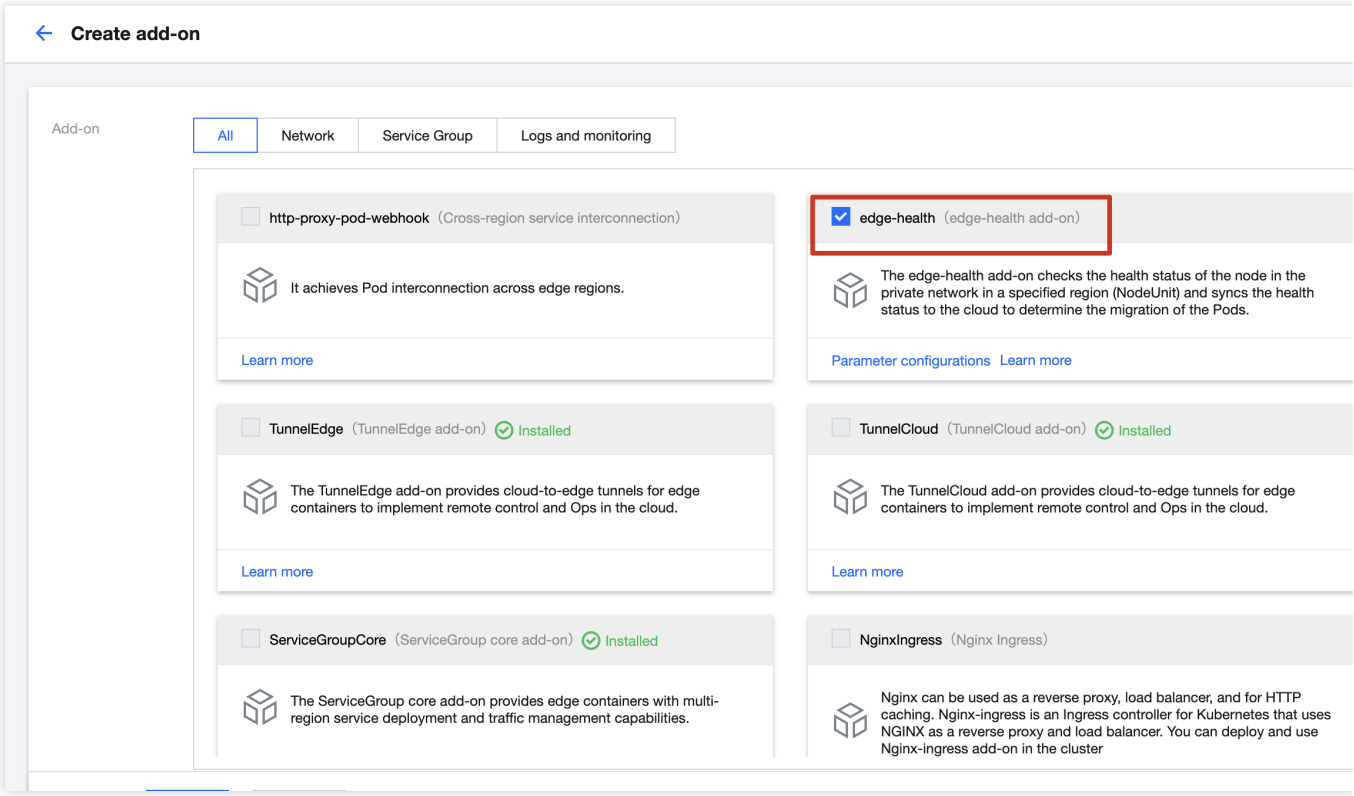
The edge health check and multi-region check features require some deployment and configuration time and will not take effect immediately.

Enable Global Health-Check

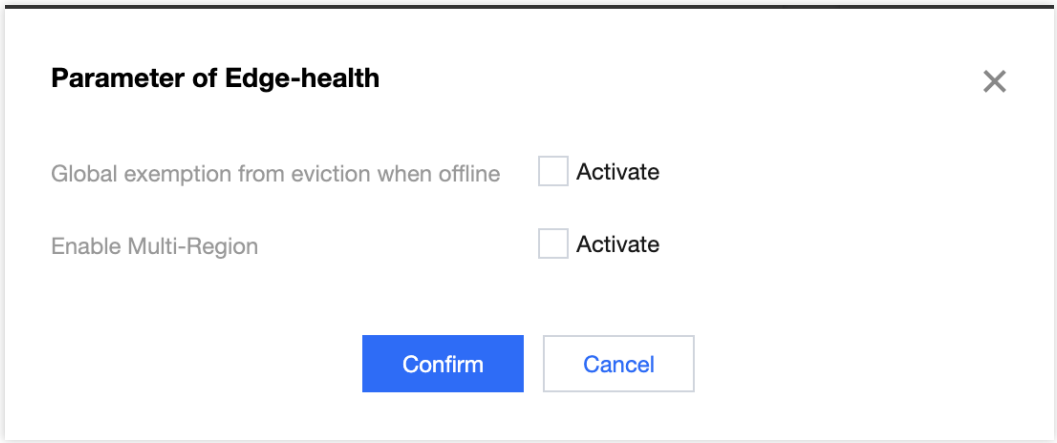
The edge health check feature is turned off by default. Please follow the steps below to enable it manually:

1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.

3. Select **"Add-on management"** and enter the add-on list page.
4. Click **"Create"**, enter the "Create add-on" page.
5. Select **"edge-health"** and click **"Done"**, as follow:



On "Parameter configurations" pages, you can enable "Global exemption from eviction when offline" and "Multi-Region" capabilities, as follow:



6. Click "Done" and "edge-health" add-on will be deployed on the edge cluster.

Global exemption from eviction when offline

After enabling this feature, all eviction flags, specifically the **NoExecute** taints, will be intercepted through the webhook mechanism. This prevents Pods from being evicted and rescheduled due to frequent node disconnections and reconnections. This feature is mutually exclusive with Multi-Region checks. When the eviction prevention feature is enabled, the actual health status of edge nodes will not be detected.

Multi-Region:

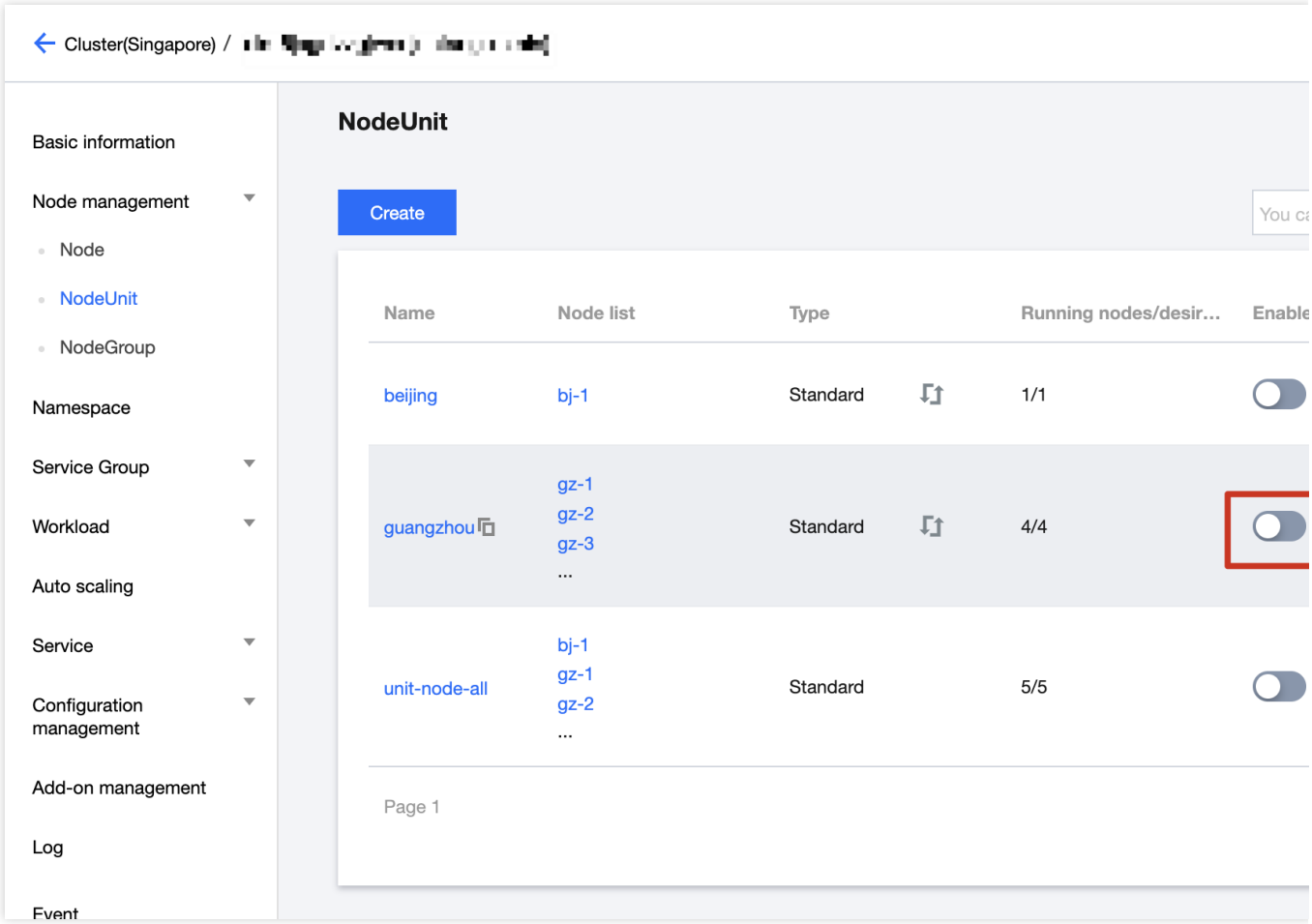
The concept of multi-region here actually refers to NodeUnit in edge-health, which distinguishes the node's regional attributes through NodeUnit. For details, please refer to the NodeUnit. Enabling Edge Health capability deploys edge-health health monitoring Pods on all edge nodes. If the multi-region capability is not enabled at this time, all nodes' statuses will be detected by default, and there is no multi-region concept. All nodes can be considered as nodes within the same region for mutual detection.
If you want nodes within different regions to independently check each other's health status, you need to enable the **Multi-Region** capability. At this point, you'll modify an **edge-health-config** ConfigMap, notifying K8s to handle multi-region health check capability. Once enabled, nodes will be separated into different regions based on the **<nodeunit-name>: nodeunits.superedge.io** label on the node. For example, the label **beijing:nodeunits.superedge.io** indicates that the node belongs to NodeUnit Beijing. Nodes with the same label are considered part of the same NodeUnit. When enabling the multi-region functionality, nodes within the same region will probe and vote on each other's state.

Note：

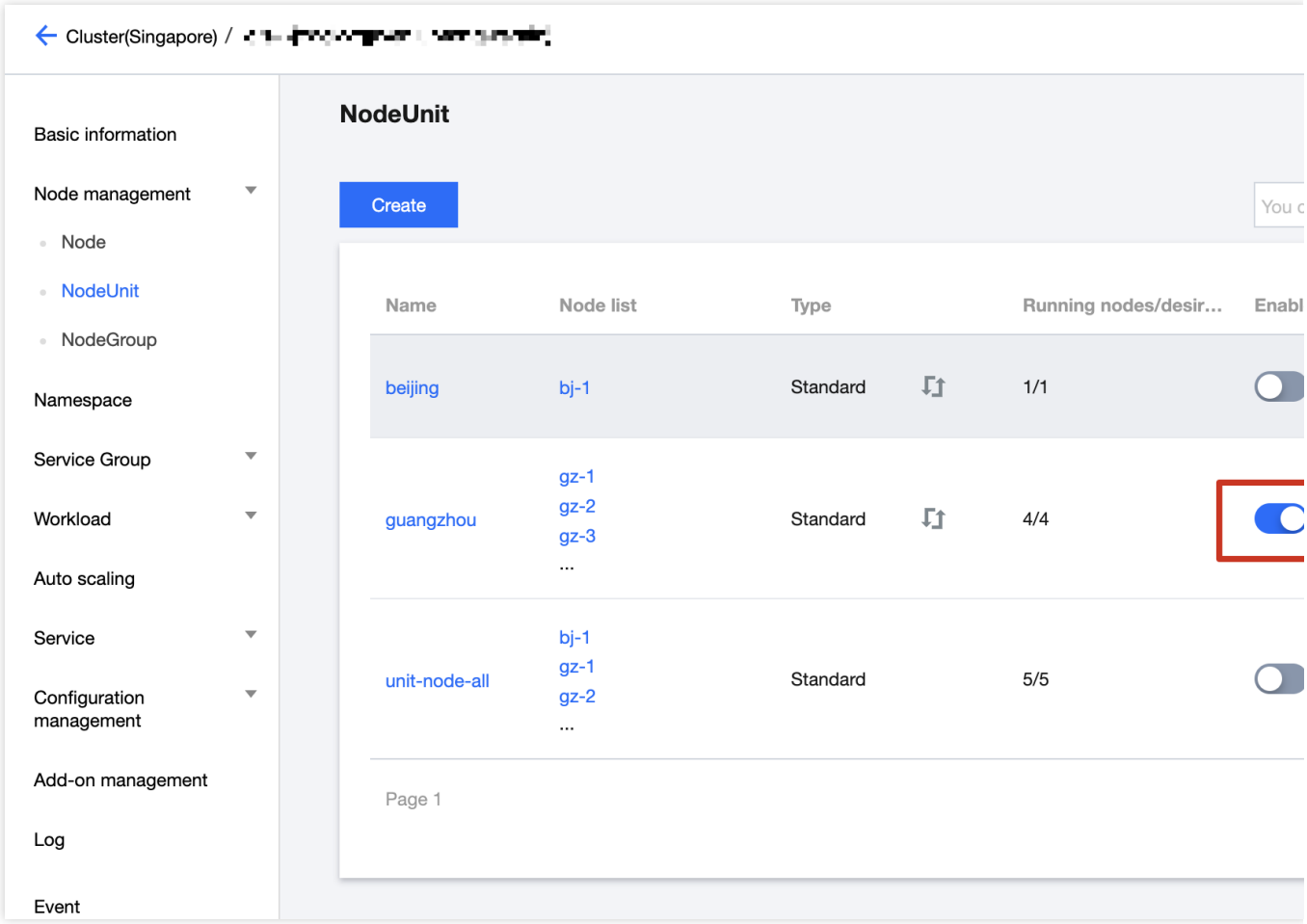
If you enable the Multi-Region capability and the node does not belong to any NodeUnit, the node will not perform health checks.
If the Multi-Region capability is not enabled, all nodes within a cluster will check each other, even if the nodes belong to a NodeUnit.

Enable Health-Check on specified NodeUnit

1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.
3. Select **"Node management" > "NodeUnit"**, and you'll see there is an additional switch called "Enable Edge Health Check", as follow:

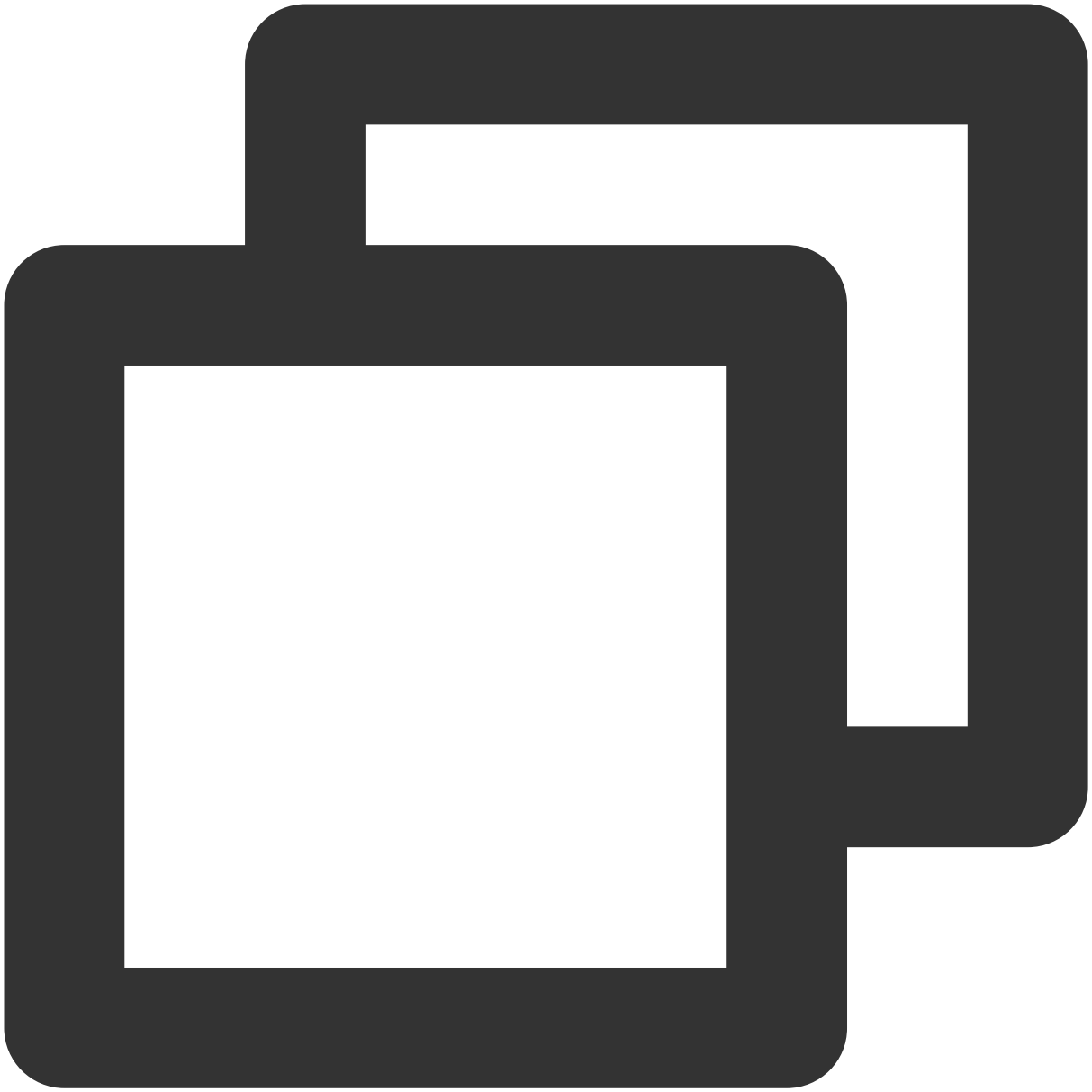


4. You can choose the NodeUnit you want and enable the "Edge Health Check" switch. At that time, you can check on the edge-health of each node that all nodes under the NodeUnit will probe each other.



5. At this point, the multi-regional health status capability is enabled. We can simulate the test using the gz-3 node, as follows:

You can use iptables rules to simulate disconnection from the apiserver:



```
iptables -I INPUT -s xxx.xxx.xxx.xxx -j DROP  # This would be the IP address of the
```

After some time, you can observe that the `gz-3` node is in a `NotReady` state, but the Taint `node.kubernetes.io/unreachable:NoExecute` is not added. Therefore, the Pods on this node will not be evicted, and no new Pods will be scheduled on this node. This behavior demonstrates the proper handling of node unavailability by the cluster and its impact on workload management.

```
Every 2.0s: kubectl get nodes

NAME   STATUS     ROLES    AGE     VERSION
bj-1   Ready      <none>   3h59m   v1.22.5-tke.1.9+5f76a1720f9839
gz-1   Ready      <none>   7h57m   v1.22.5-tke.1.9+5f76a1720f9839
gz-2   Ready      <none>   7h57m   v1.22.5-tke.1.9+5f76a1720f9839
gz-3   NotReady   <none>   4h3m    v1.22.5-tke.1.9+5f76a1720f9839
gz-4   Ready      <none>   4h3m    v1.22.5-tke.1.9+5f76a1720f9839
```

```
Name:               gz-3
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    guangzhou=nodeunits.superedge.io
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=gz-3
                    kubernetes.io/os=linux
                    location=guangzhou
                    unit-node-all=nodeunits.superedge.io
Annotations:        flannel.alpha.coreos.com/backend-data: {"VtepMAC":"2a:44:f6:f2:ce
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 172.16.16.106
                    node.alpha.kubernetes.io/ttl: 0
                    superedge.io/node-quota: false
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 30 May 2023 16:04:42 +0800
Taints:             node.kubernetes.io/unreachable:NoSchedule
Unschedulable:      false
```

To simulate a scenario where both cloud and internal network checks fail, shut down the `gz-3` node and observe its status. In this situation, the node state reflects the standard Kubernetes node failure state, and the Pods on this node will be evicted and rescheduled to other available nodes.

```
Name:               gz-3
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    guangzhou=nodeunits.superedge.io
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=gz-3
                    kubernetes.io/os=linux
                    location=guangzhou
                    unit-node-all=nodeunits.superedge.io
Annotations:        flannel.alpha.coreos.com/backend-data: {"VtepMAC":"2a:44:f6:f2:ce:c0"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 172.16.16.106
                    node.alpha.kubernetes.io/ttl: 0
                    nodeunhealth: yes
                    superedge.io/node-quota: false
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 30 May 2023 16:04:42 +0800
Taints:             node.kubernetes.io/unreachable:NoExecute
                    node.kubernetes.io/unreachable:NoSchedule
Unschedulable:      false
```

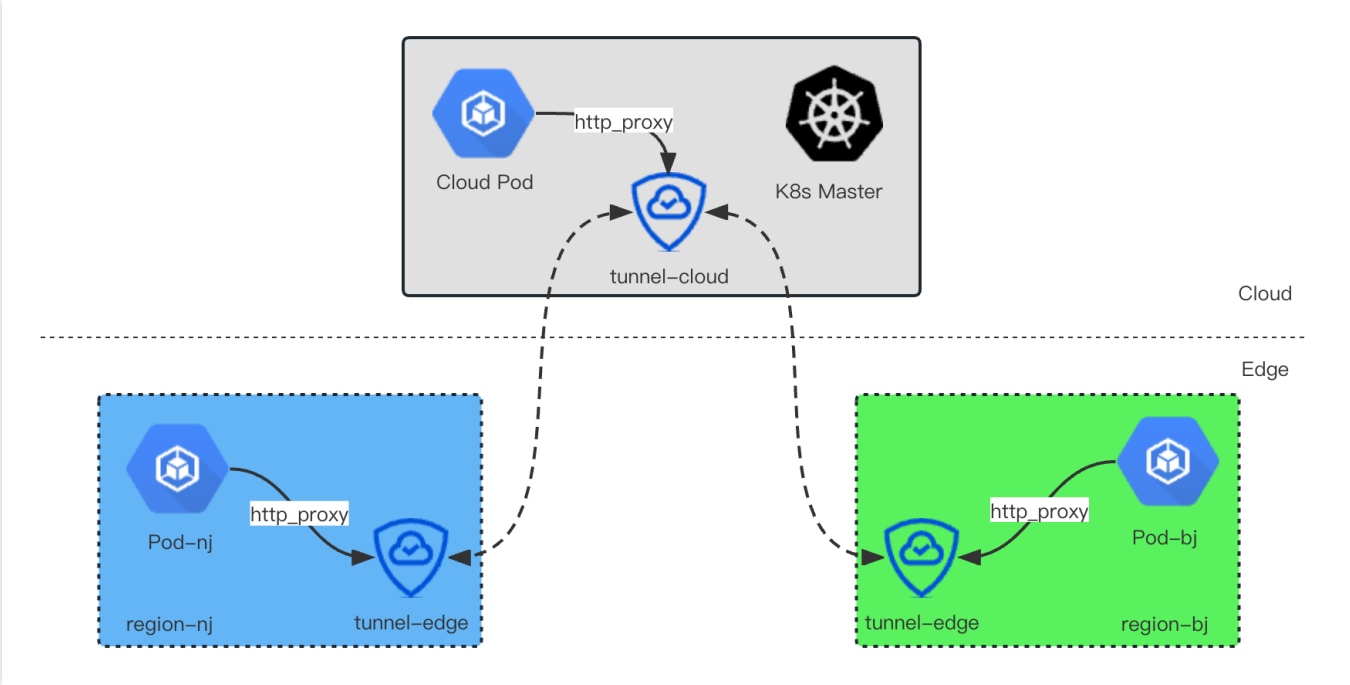
Service Access across Regions

Last updated：2023-06-01 11:22:54

Operation Scenarios

This article introduces how to enable Pods/Svc from different edge regions to access each other through the cloud-edge Tunnel (currently only supporting Layer 7 http/https protocols). For example, a Pod in the Nanjing region can successfully access an Nginx service in the Beijing region, and vice versa.

Architecture



As illustrated in the diagram above, the edge node tunnel-edge will establish a bidirectional tunnel with the cloud tunnel-cloud. The tunnel-edge will open the local 8080 port as the http/https proxy port. If an edge Pod needs to access other regions' Pod IPs or services through a proxy, you can specify the http_proxy or https_proxy environment variable within the Pod. This will forward the http/https traffic through the tunnel to the cloud or other edge nodes in different regions.

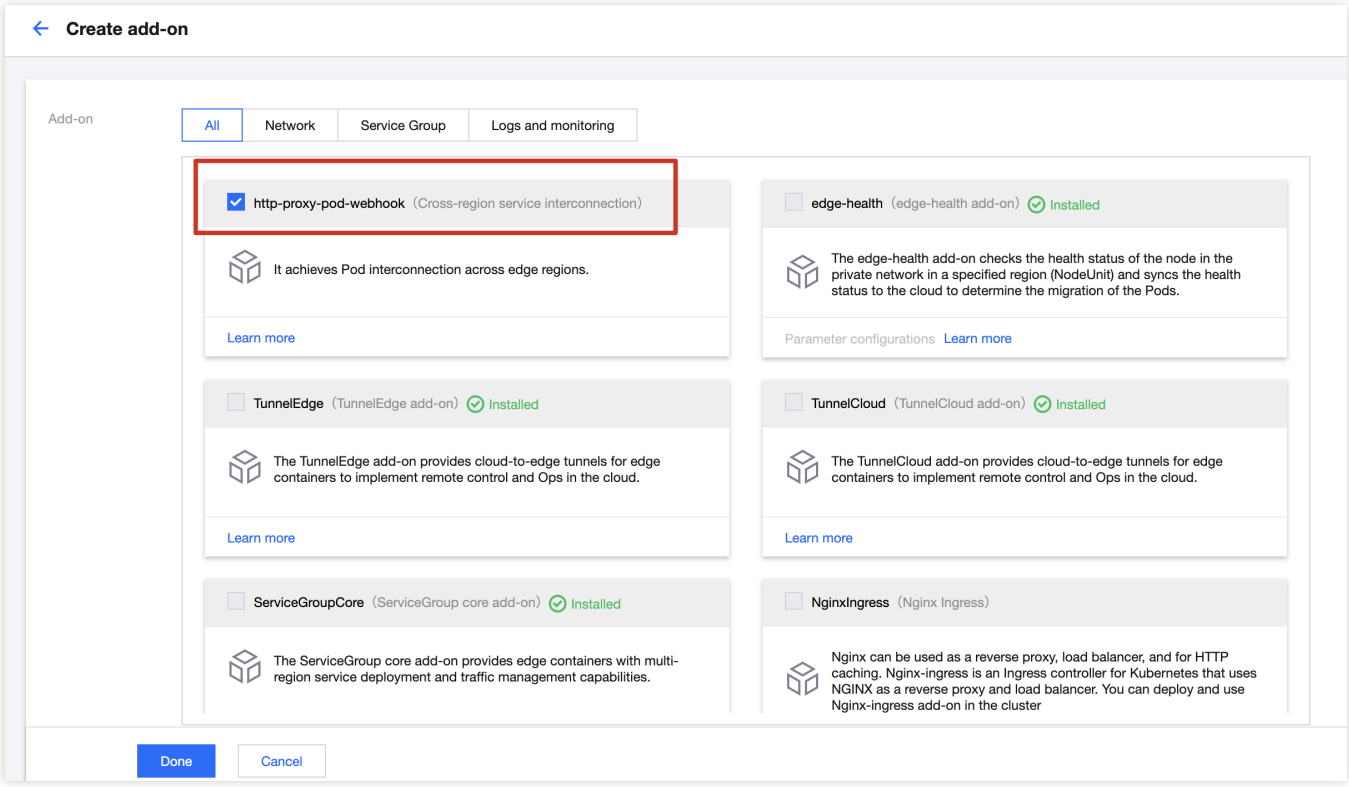
Note：

At the product level, the platform uses the webhook method to conveniently specify proxies for you. You can set the label http-proxy=enable in the workload to enable proxy mode. When this is done, the corresponding Pods of the

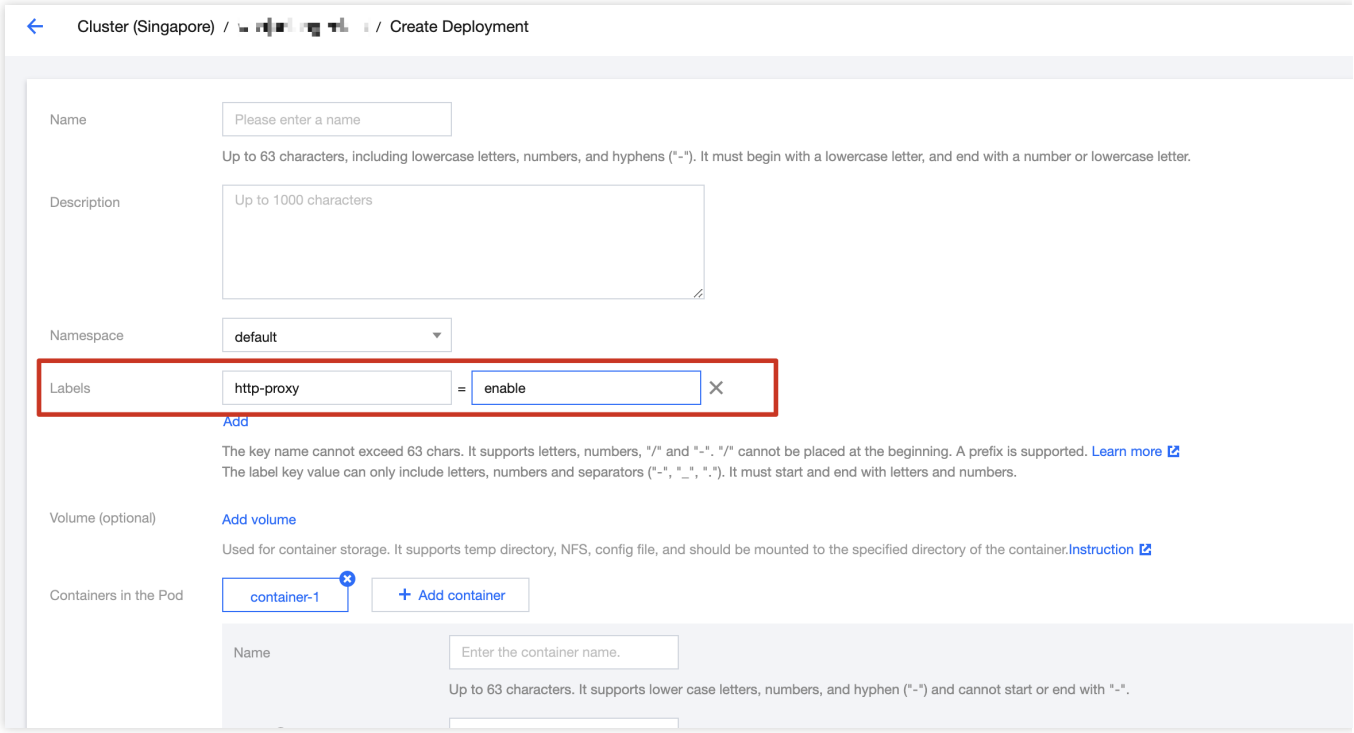
workload will automatically inject the relevant environment variables, such as http_proxy=169.254.20.11:8080

Operation Steps

1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.
3. Select **"Add-on management"** and enter the add-on list page.
4. Click **"Create"**, enter the "Create add-on" page.
5. Select **"http-proxy-pod-webhook"** and click **"Done"**, as follow:



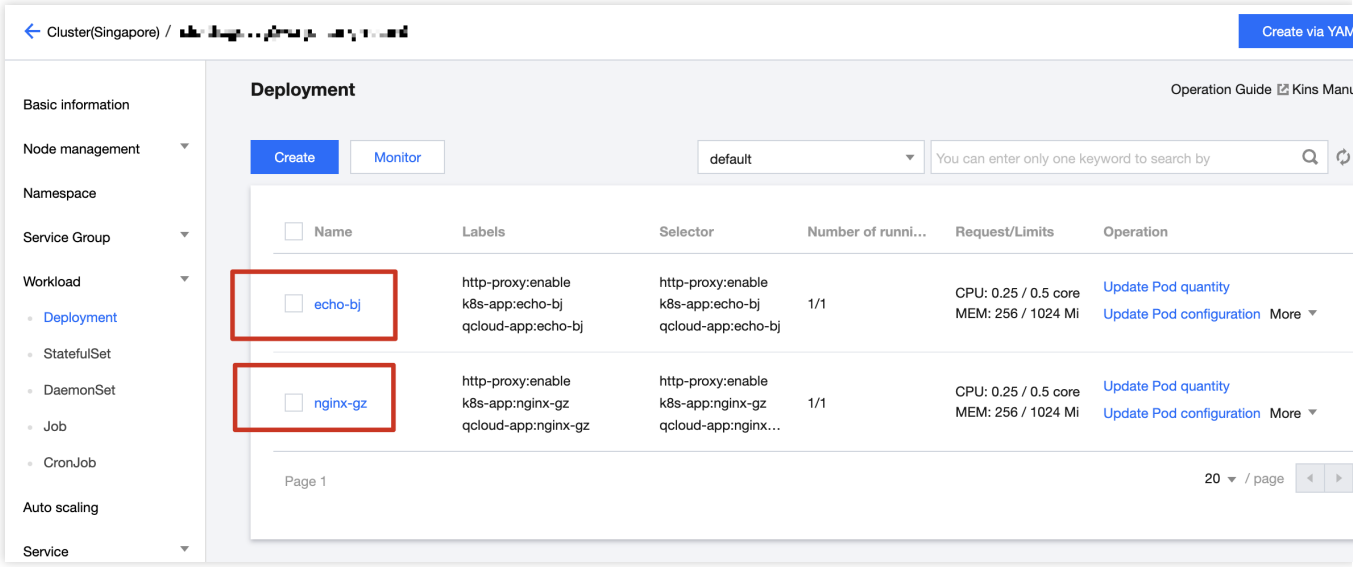
6. To verify the across-region access feature, create deployment in different regions as follows:



Add "http-proxy=enable" label to the deployment, then the webhook of the step 5 will inject http_proxy in the Pod environment, as follow:

```
[root@bj-1 ~]# crictl exec -it d9ba3b2aa2d7a /bin/sh
/ # env
KUBERNETES_PORT=tcp://10.44.0.1:443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=echo-bj-55c4bc47bf-8sq5p
SHLVL=1
HOME=/root
ECHO_BJ_SERVICE_HOST=10.44.224.177
ECHO_BJ_PORT_8080_TCP_ADDR=10.44.224.177
ECHO_BJ_PORT_8080_TCP_PORT=8080
ECHO_BJ_PORT_8080_TCP_PROTO=tcp
http_proxy=http://169.254.20.11:8080
TERM=xterm
ECHO_BJ_SERVICE_PORT=8080
ECHO_BJ_PORT=tcp://10.44.224.177:8080
KUBERNETES_PORT_443_TCP_ADDR=10.44.0.1
NGINX_VERSION=1.15.3
ECHO_BJ_SERVICE_PORT_8080_8080_TCP_3T8IH1V68HM=8080
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
ECHO_BJ_PORT_8080_TCP=tcp://10.44.224.177:8080
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.44.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
PWD=/
KUBERNETES_SERVICE_HOST=10.44.0.1
```

After the 2 deployments deployed, you'll check the status of these deployments:



Caution：

Each deployment must use the scheduler to assign the pod to the specific node. For example, echo-bj is assigned to "bj-1" and nginx-gz is assigned to "gz-2"

7. Login to "bj-1" node add access the pods of "gz-2", as follow:

```
NAME                        READY   STATUS    RESTARTS   AGE     IP           NODE   NOMINATED NODE   REA
echo-bj-55c4bc47bf-8sq5p    1/1     Running   0          9m18s   10.33.2.3    bj-1   <none>           <no
nginx-gz-d94bf4fc7-c492d    1/1     Running   0          7m49s   10.33.0.72   gz-2   <none>           <no
```

When accessing the pod of gz-2, you'll see the request is redirect to **http_proxy "169.254.20.11:8080"**

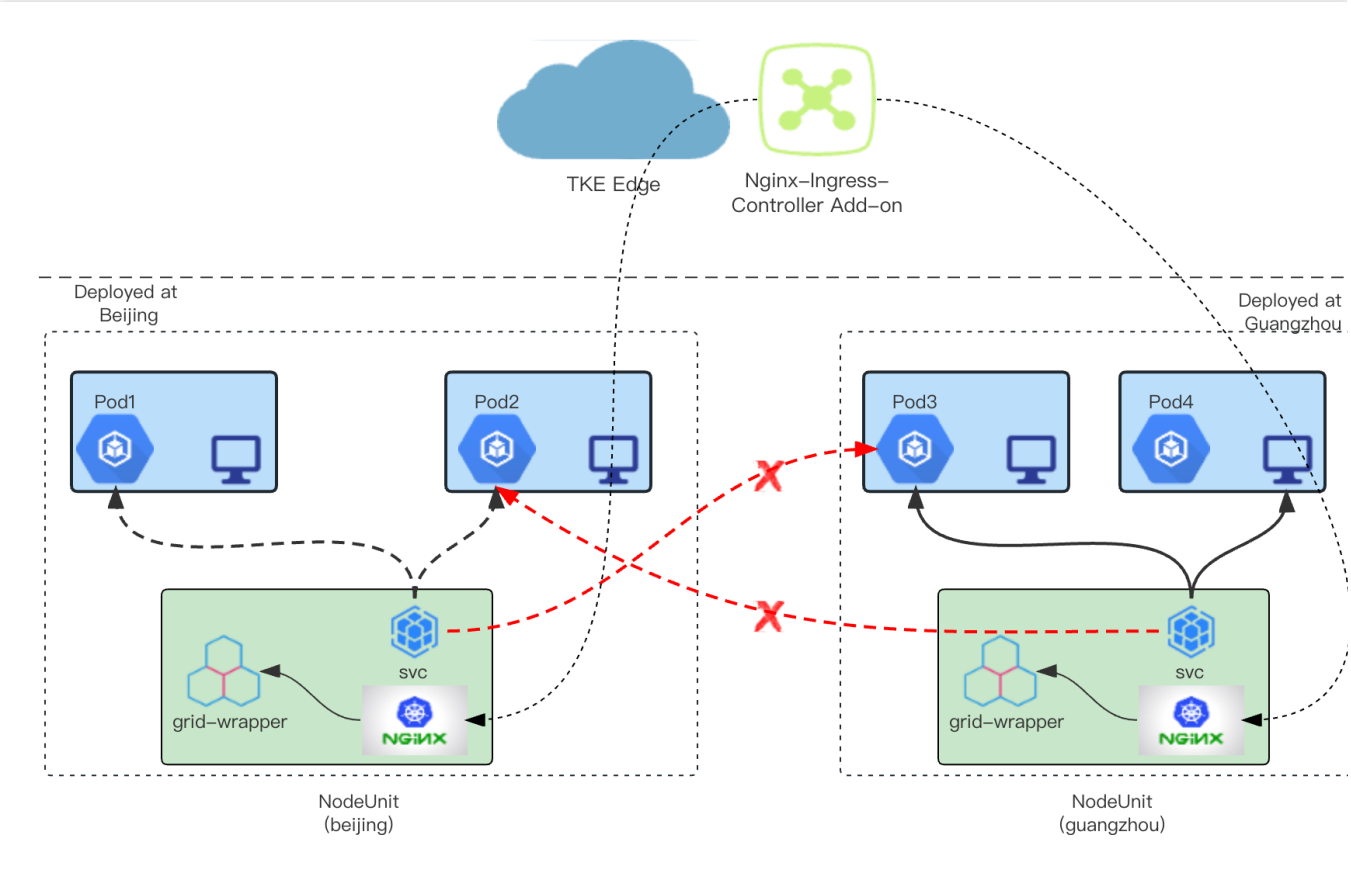
```
[root@bj-1 ~]# crictl exec -it d9ba3b2aa2d7a /bin/sh
/ # wget -O- http://10.33.0.72:8080
Connecting to 169.254.20.11:8080 (169.254.20.11:8080)


Hostname: nginx-gz-d94bf4fc7-c492d

Pod Information:
        -no pod information available-

Server values:
        server_version=nginx: 1.12.2 - lua: 10010

Request Information:
        client_address=10.33.0.65
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://10.33.0.72:8080/

Request Headers:
        connection=close
        host=10.33.0.72:8080
        user-agent=Wget

Request Body:
        -no body in request-

-                    100% |*******************************************
/ #
```

Multi-Region Ingress

Last updated：2023-06-01 11:22:54

Scenarios

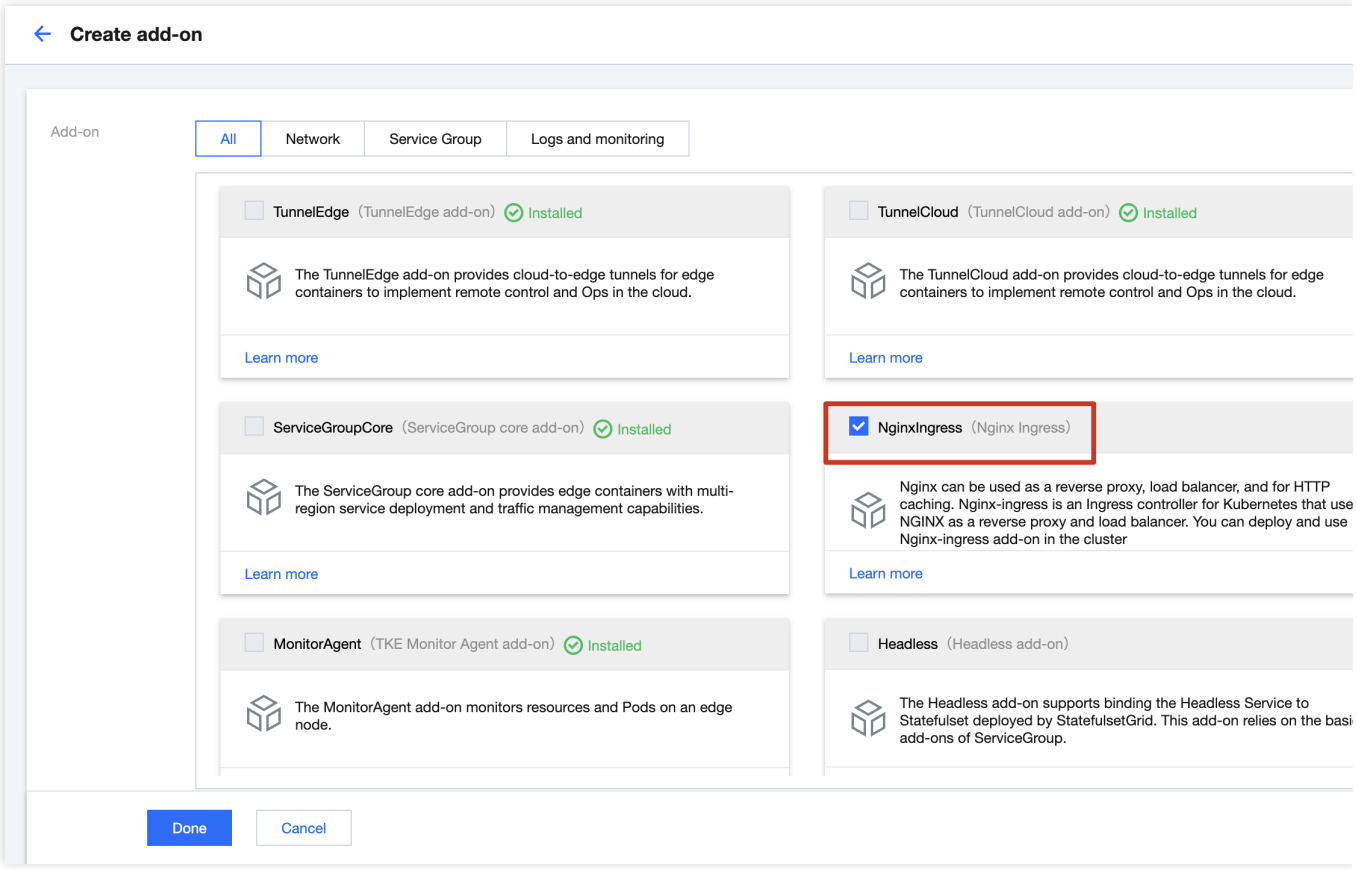
In the case of distinguishing multiple regions in edge nodes, each region has an independent network architecture and needs to provide Ingress service capabilities externally in the local region. Based on the concept of NodeUnit, TKE-Edge supports the ability to create **Nginx-Ingress-Controller** in different regions in the product. At the same time, through the application-grid-wrapper component, the Ingress-Controller's access to pods can be restricted within the local region (NodeUnit range). The specific architecture is shown in the following figure:



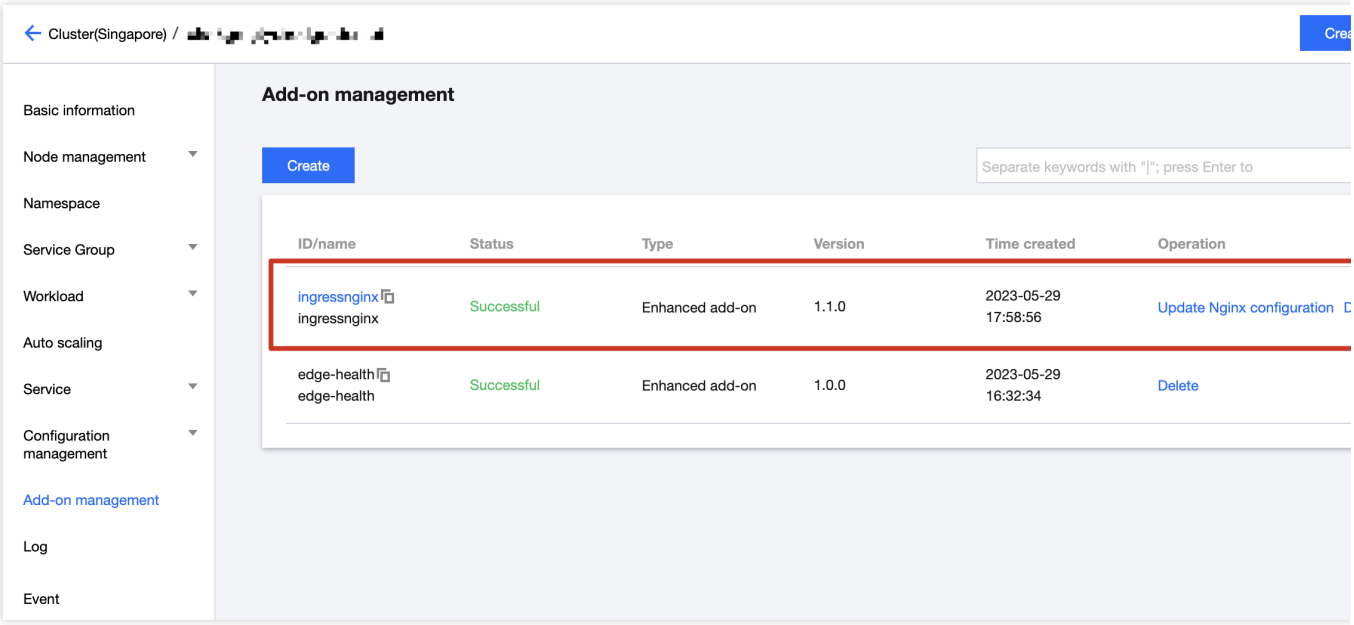
Operation Steps

1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.

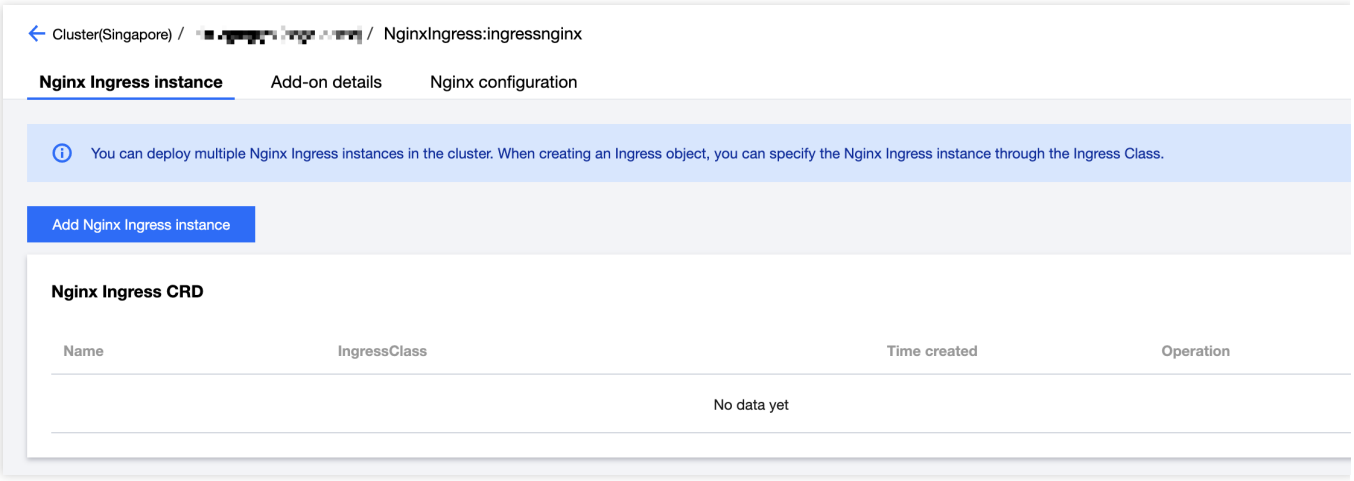
3. Select **"Add-on management"** and enter the add-on list page.
4. Click **"Create"**, enter the "Create add-on" page.
5. Select **"NginxIngress"** and click **"Done"**, as follow:



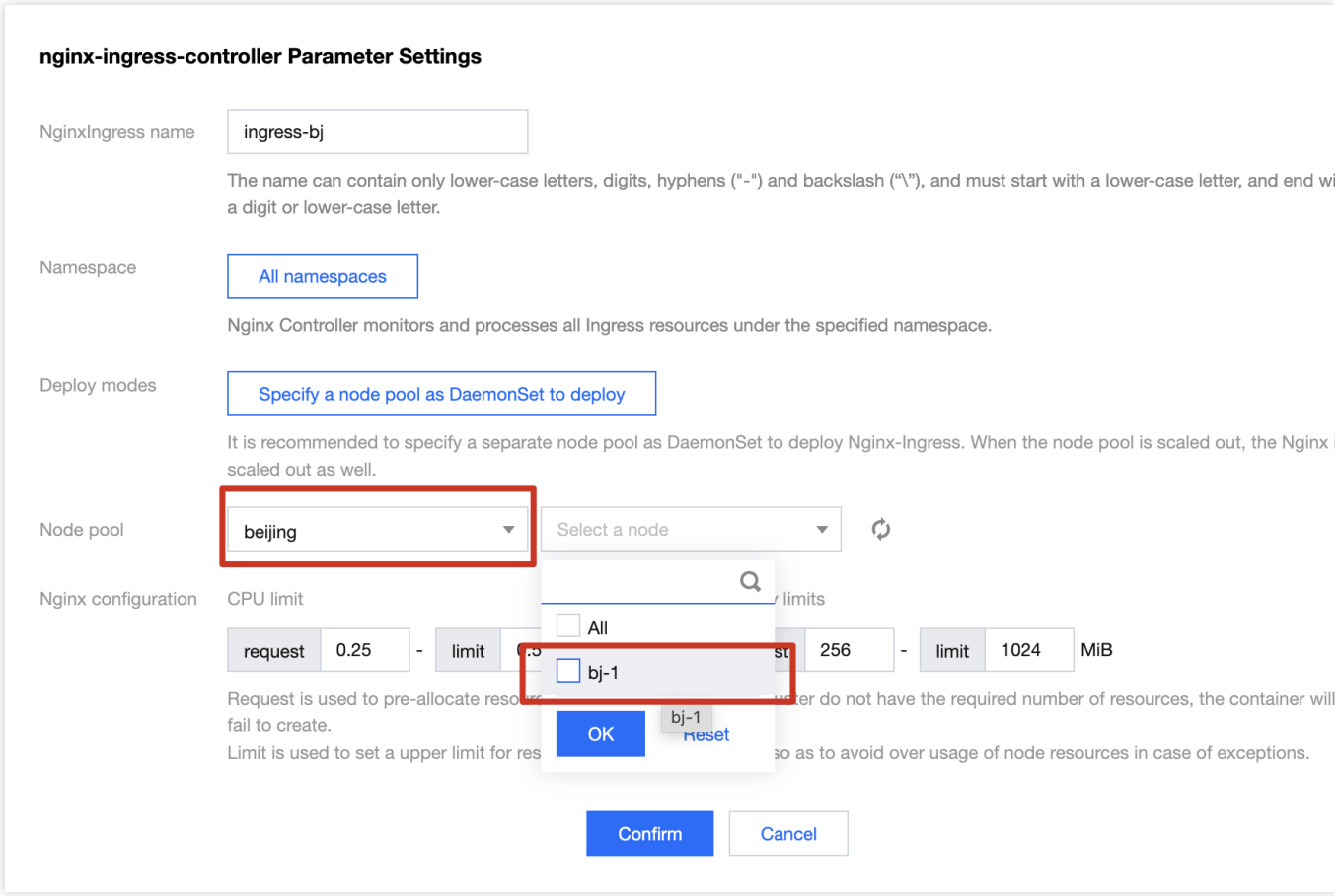
6. Wait for the add-on is ready, as follow:



7. Click **"ingressnginx"** and enter the detailed page. As follow:



8. Click **"Add Nginx Ingress instance"**, and create Nginx-Ingress-Controller at specified NodeUnit, for example in bj NodeUnit:



NginxIngress name: Specifies the name of the deployed Ingress-Ingress-Controller instance, which creates the corresponding daemonset.

Namespace: Currently set to listen to Ingress resources in all namespaces by default.

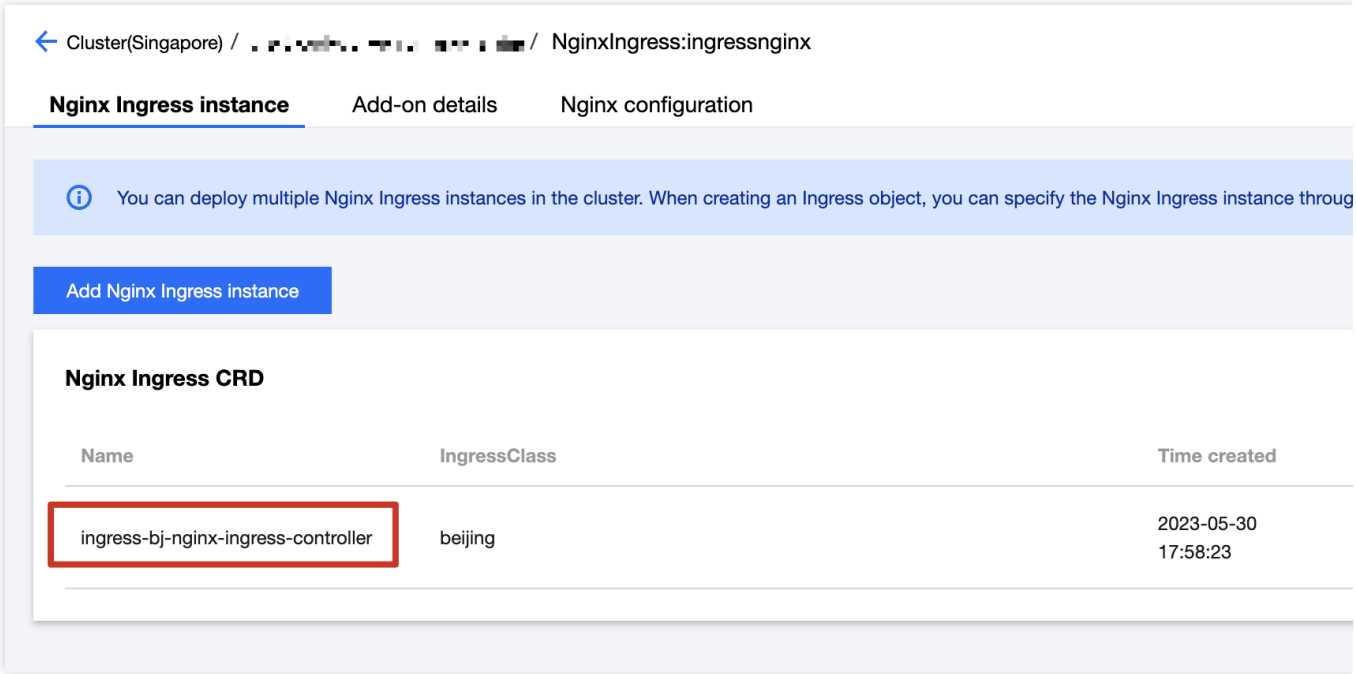
Deployment modes: Currently, the default deployment mode is using DaemonSet. The user must manually choose one or more nodes under the NodeUnit to provide the Ingress Controller service.

Node Pool: Select the NodeUnit where you want to deploy, and then choose the nodes on which you want to deploy the Nginx-Ingress-Controller service.

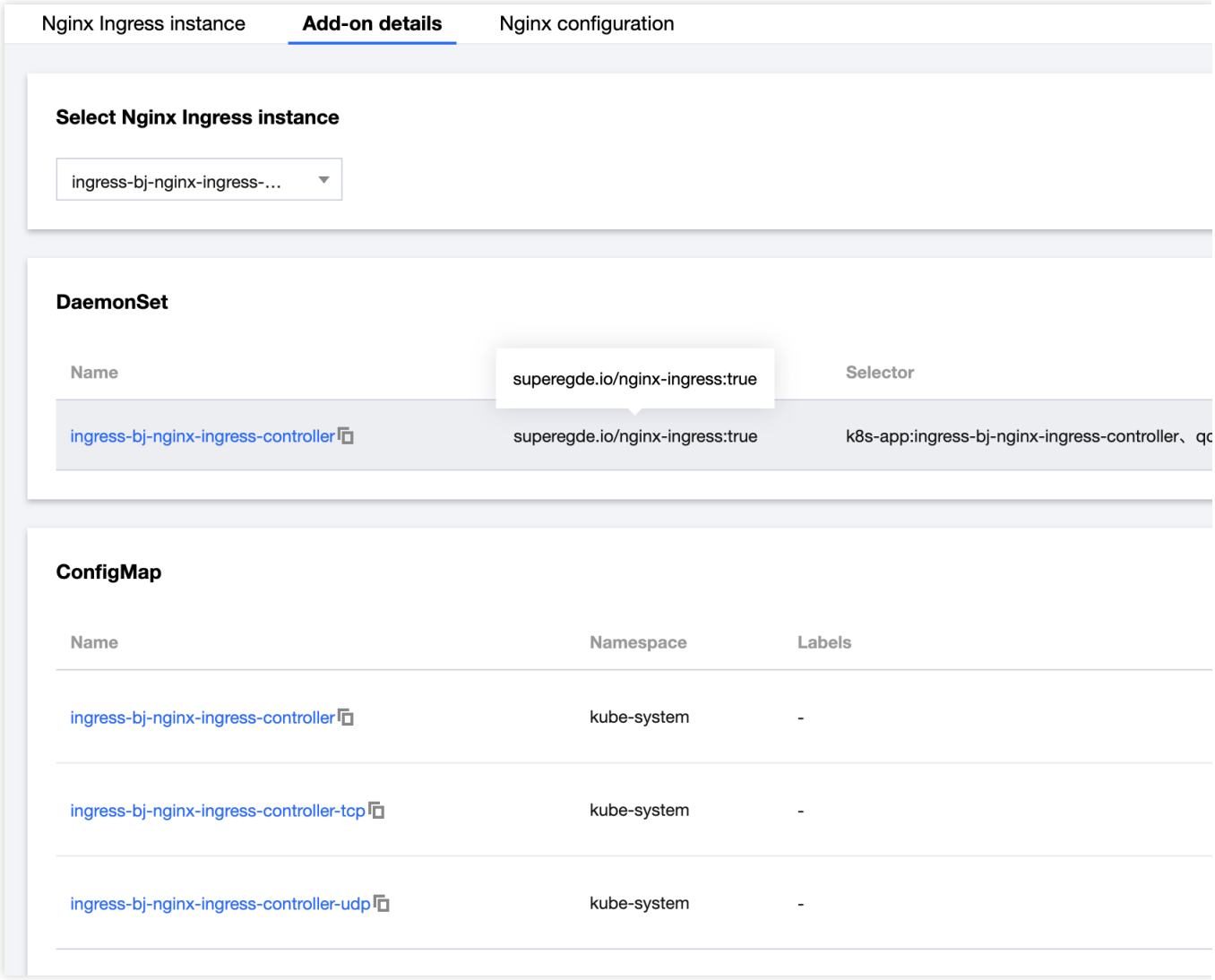
Caution：

Make sure that the selected nodes have ports 80 and 443 available, as they should not be occupied. Otherwise, the Nginx-Ingress-Controller may fail to start.

9. After clicking **"Confirm",** you can view the created Ingress-Controller instance, as shown in the following image:



10. By checking the component details, you can confirm the current deployment status of the controller. Once the number of running Pods reaches the expected count, it indicates that the deployment has succeeded, as shown in the following image:



Deploy Service and Access it Using Ingress:

1. Here we take the Nginx service as an example, creating a Deployment where the Pods will be deployed to the beijing and guangzhou regions, as shown in the image below:

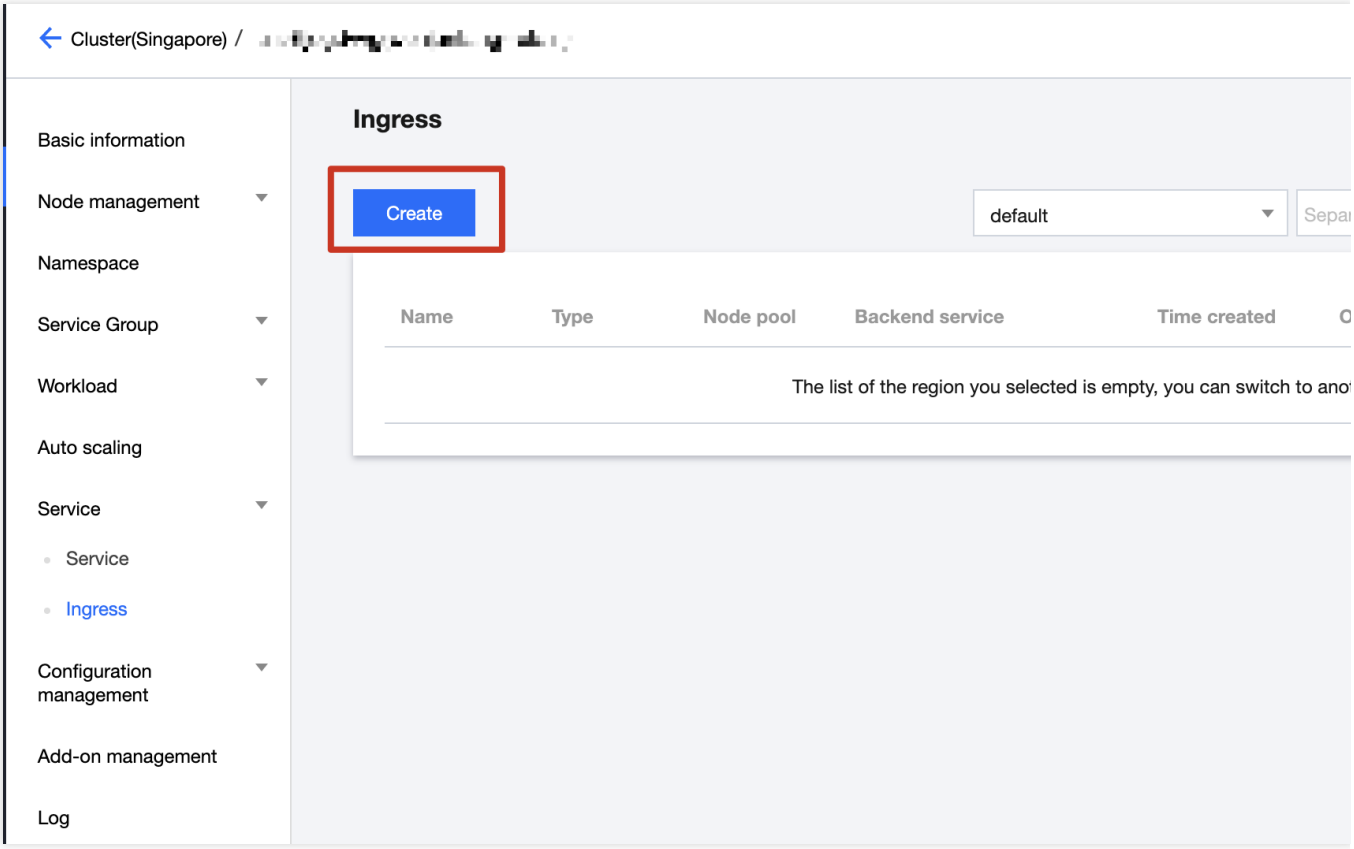
```
nginx-beijing-76cdbbc899-k66c8     1/1     Running   0          2m11s   10.33.2.4
nginx-guangzhou-75f4c49d47-tzkgg   1/1     Running   0          2m11s   10.33.0.6
```

Then use the ServiceGrid to provide internal access to the deployed service:

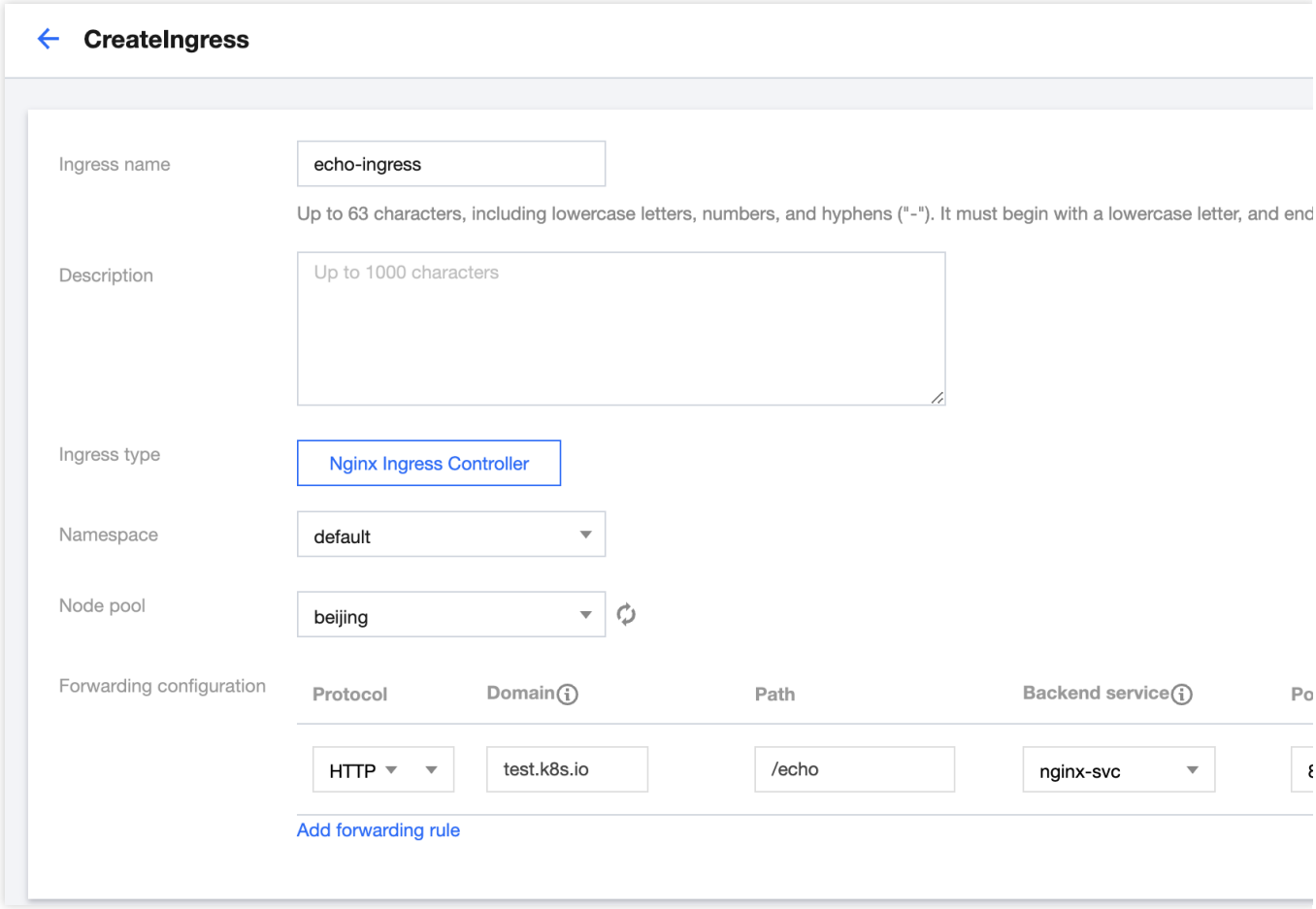
```
Name:              nginx-svc
Namespace:         default
Labels:            superedge.io/grid-selector=nginx
                   superedge.io/grid-uniq-key=location
Annotations:       topologyKeys: ["location"]
Selector:          k8s-app=nginx,qcloud-app=nginx
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.44.6.125
IPs:               10.44.6.125
Port:              tcp-8080-8080  8080/TCP
TargetPort:        8080/TCP
Endpoints:         10.33.0.6:8080,10.33.2.4:8080
Session Affinity:  None
Events:            <none>
```

In this scenario, we expect that when accessing Ingress from the Beijing region, only the Pods in the Beijing region will be accessed, for example, `nginx-beijing-76cdbbc899-k66c8`, and it won't access the Pods in the Guangzhou region.

2. To create an Ingress, navigate to the cluster details page **"Services" > "Ingress"**, and click **"Create"**. As shown in the image below:



3. Enter the required Ingress information, as shown in the following image:



Ingress name: Enter the name for the Ingress you want to create.

Description: Enter a descriptive remark.

Ingress type: Currently, only the Nginx Ingress Controller type is supported by default.

Namespace: Specify the namespace to create this Ingress.

Node Pool: Choose the Ingress-Controller to be bound to this Ingress, which refers to the Nginx-Ingress-Controller instance deployed earlier.

Forwarding Configuration: Input the specific configuration for the service here. For example, entering a custom domain name test.k8s.io, path set to echo, and backend service pointing to the previously created nginx-svc.

4. Access the Ingress service within the corresponding region. For example, in the Beijing region, access http://test.k8s.io/echo, as shown in the image below:

```
[root@bj-1 ~]# curl http://test.k8s.io/echo


Hostname: nginx-beijing-76cdbbc899-k66c8

Pod Information:
        -no pod information available-

Server values:
        server_version=nginx: 1.12.2 - lua: 10010

Request Information:
        client_address=10.33.2.1
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://test.k8s.io:8080/

Request Headers:
        accept=*/*
        host=test.k8s.io
        user-agent=curl/7.61.1
        x-forwarded-for=192.168.20.234
        x-forwarded-host=test.k8s.io
        x-forwarded-port=80
        x-forwarded-proto=http
        x-forwarded-scheme=http
        x-real-ip=192.168.20.234
        x-request-id=7fb10970f188c46323106db51c31d5d6
        x-scheme=http

Request Body:
        -no body in request-
```

You can perform multiple access tests and find that all accesses will be restricted to the Pods within the Beijing region, and will not access the Pods in the Guangzhou region. This demonstrates the effectiveness of the Ingress and ServiceGroup configuration in managing traffic and ensuring that interactions remain within the specified NodeUnit and region.

Kins Manual

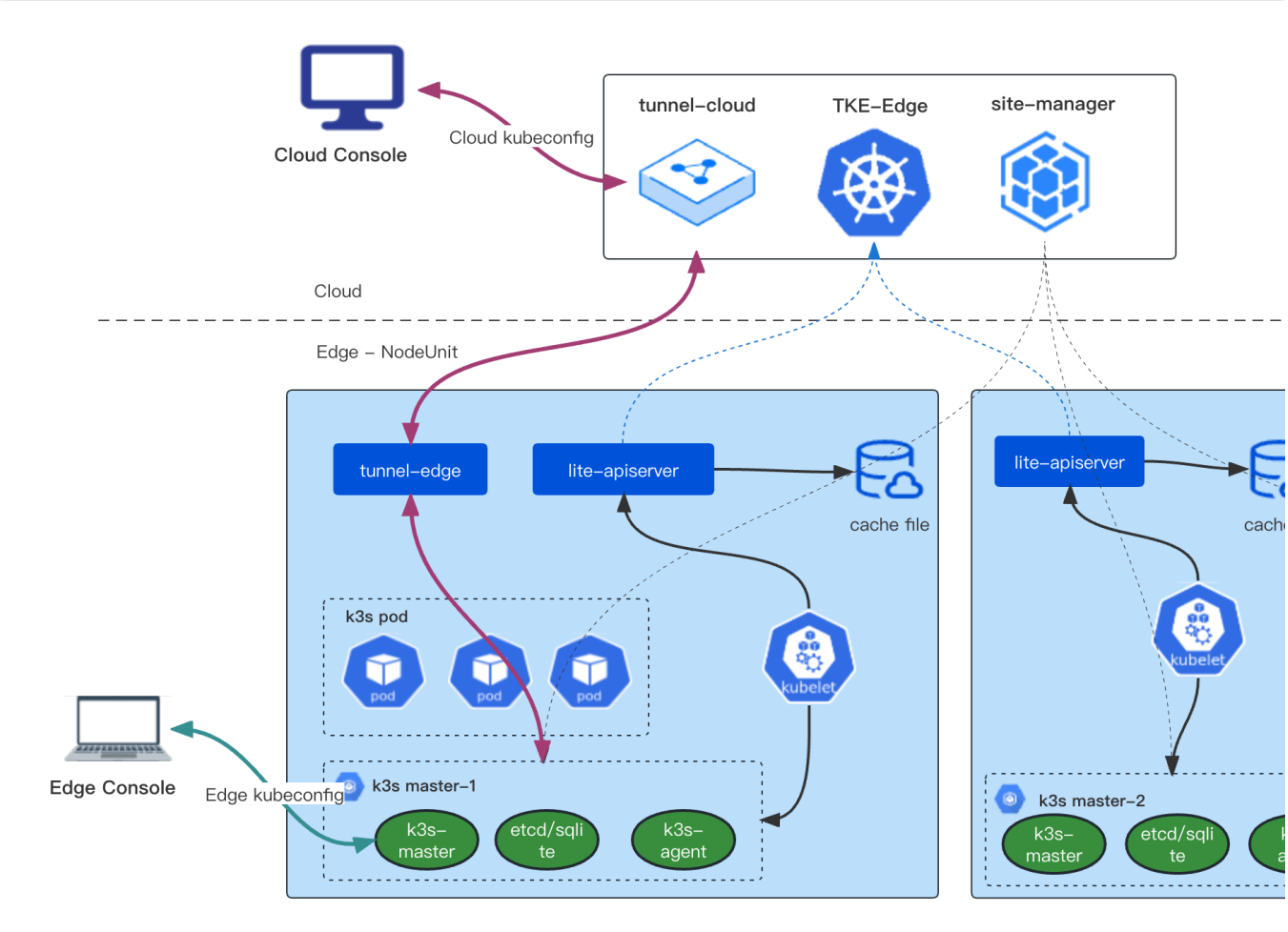
Last updated：2023-06-01 11:22:54

Scenarios

This article introduces a new feature of TKE-Edge - Kins, with the main functions described as follows: This capability can turn a standard NodeUnit edge node pool into an independent K3s cluster with a single click. This edge node pool can then be disconnected from the cloud control plane and used for long-term offline operation. During the offline process, the edge node pool can be independently operated and maintained as a standard K3s cluster. If there is a need for maintenance or upgrades in the future, this node pool can be reconnected to the cloud, enabling remote maintenance operations such as cloud-based synchronization upgrades. This feature comprehensively improves the previous capability deficiency of NodeUnit, which could not operate autonomously after disconnecting from the network.

Architecture

The basic architecture of Kins is as follow:



For example, the three edge nodes can be divided into a NodeUnit. Through edge node pool-related operations, the K3s master and agent components can be deployed on these three nodes to form an independent single-master or 3-master K3s cluster. This edge K3s cluster can be accessed from the cloud via a tunnel, and it can also be accessed directly on the edge-side nodes.

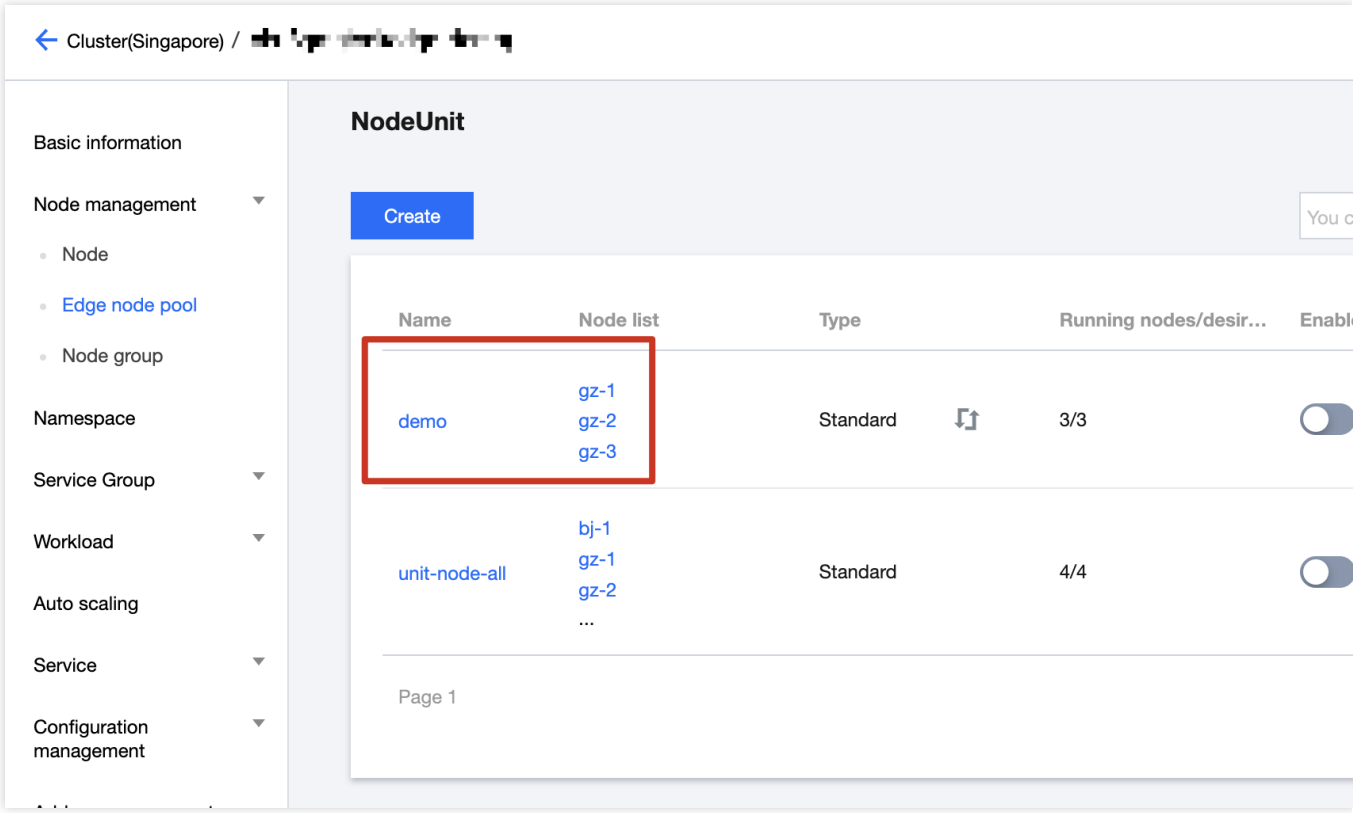
Operation Steps

Caution：

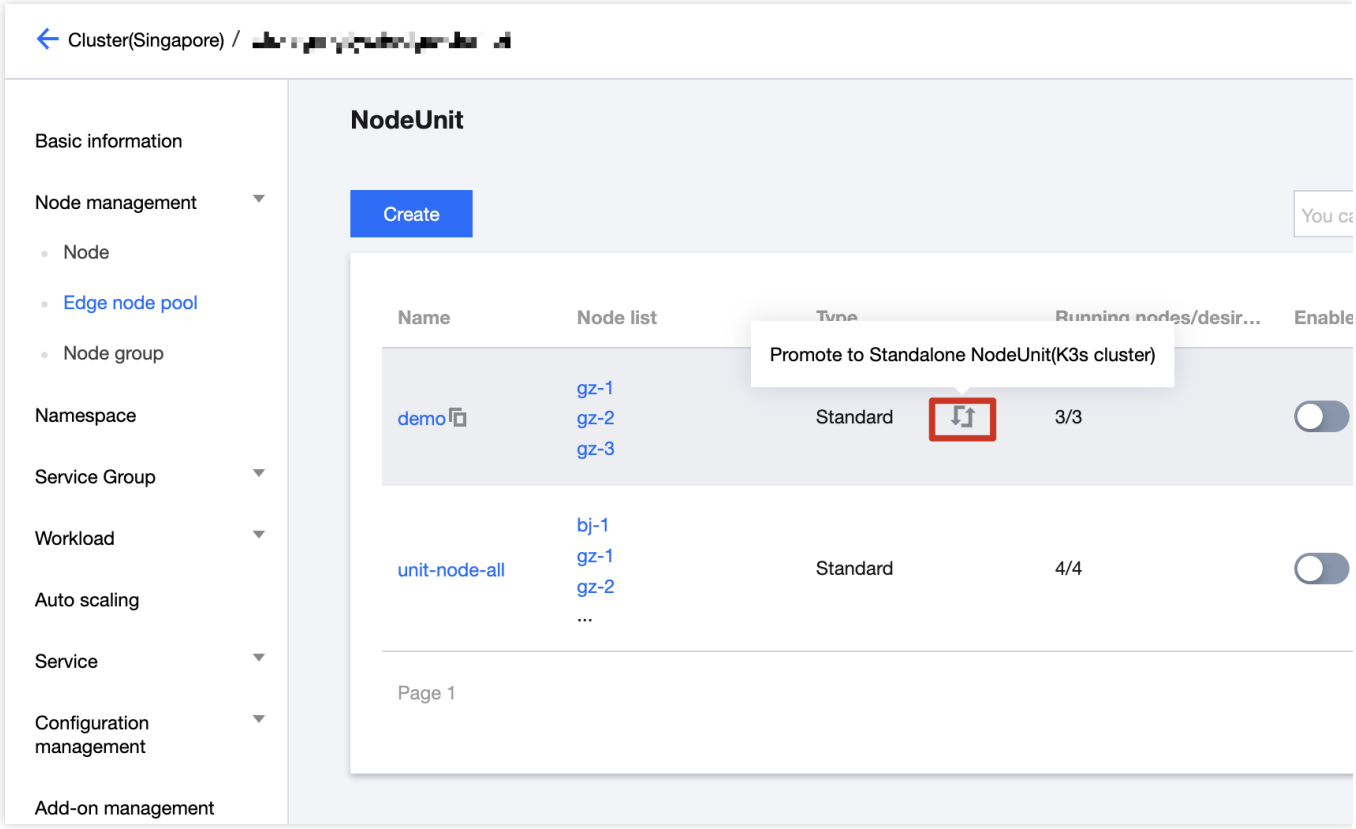
The Kins capability is currently only supported for **Kubernetes version 1.22**, and the runtime must be **containerd**.

Create edge K3s cluster

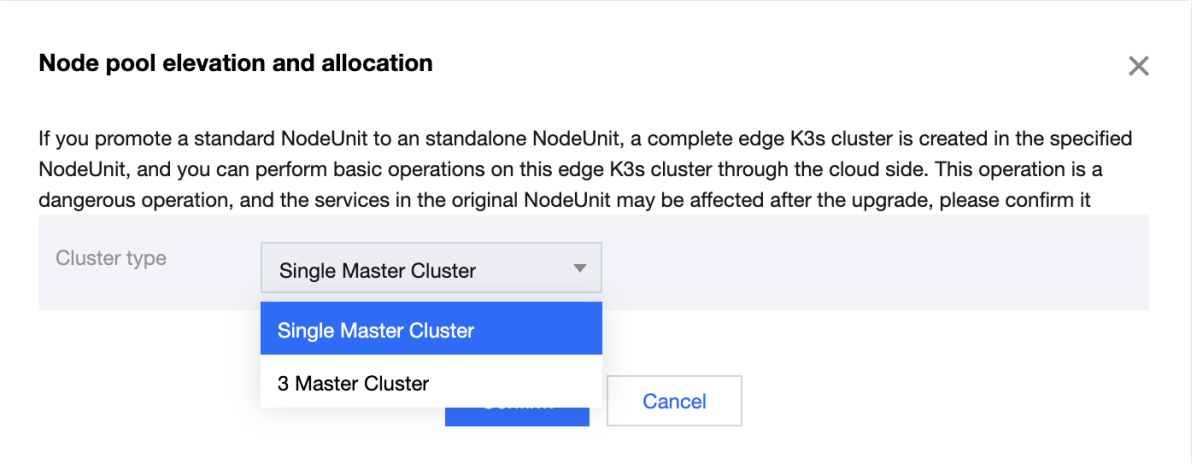
1. Log in to the TKE console.
2. On the cluster management page, click the cluster ID to enter the cluster details page.
3. Add nodes into the cluster.
4. Click **"Node management" > "Edge node pool"**, and create standard NodeUnit called "demo".



5. Select "demo" and click **"Promote to Standalone NodeUnit(K3s cluster)"** to promote K3s cluster, as follow:

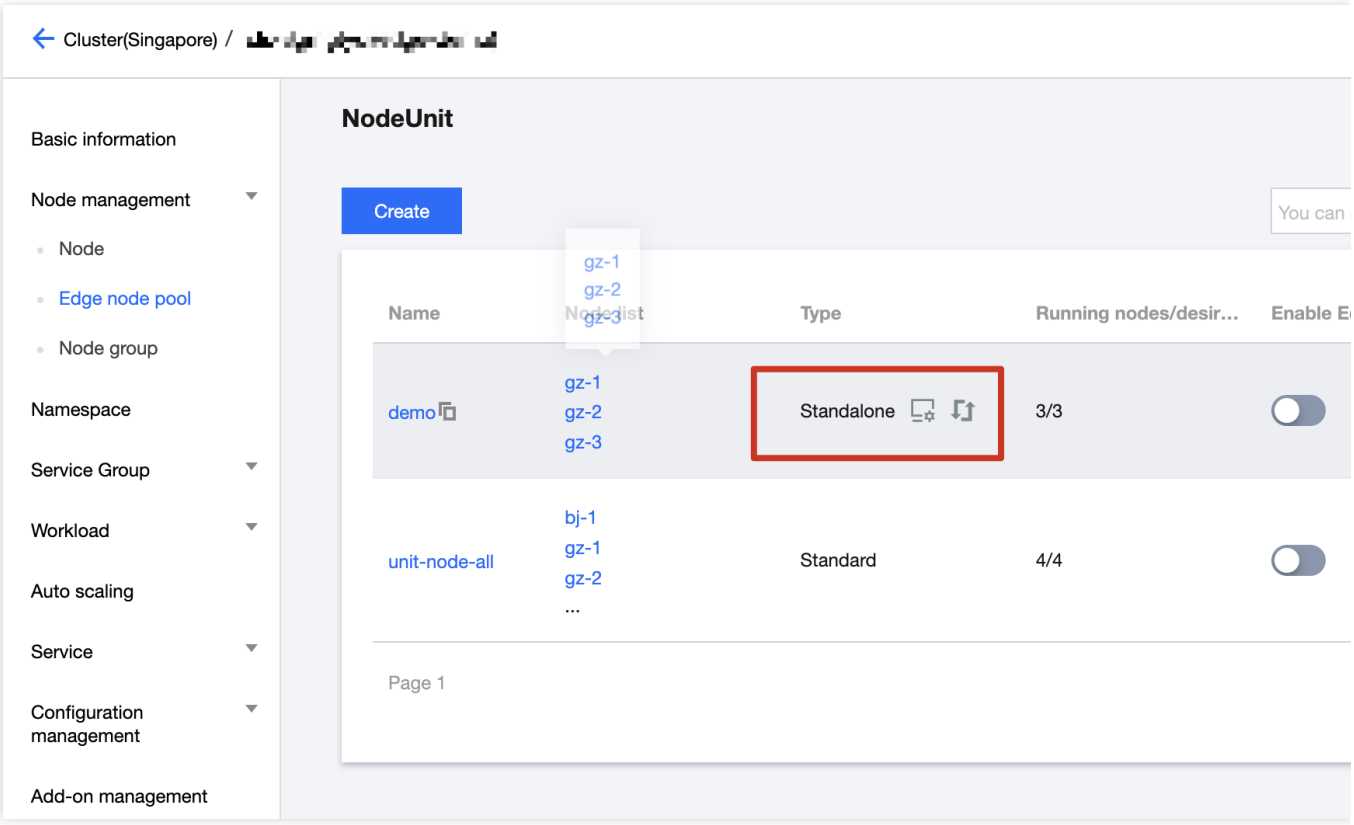


6. At the "Node pool elevation and allocation" page, you can select **"Cluster type"** and Click **"Done"**



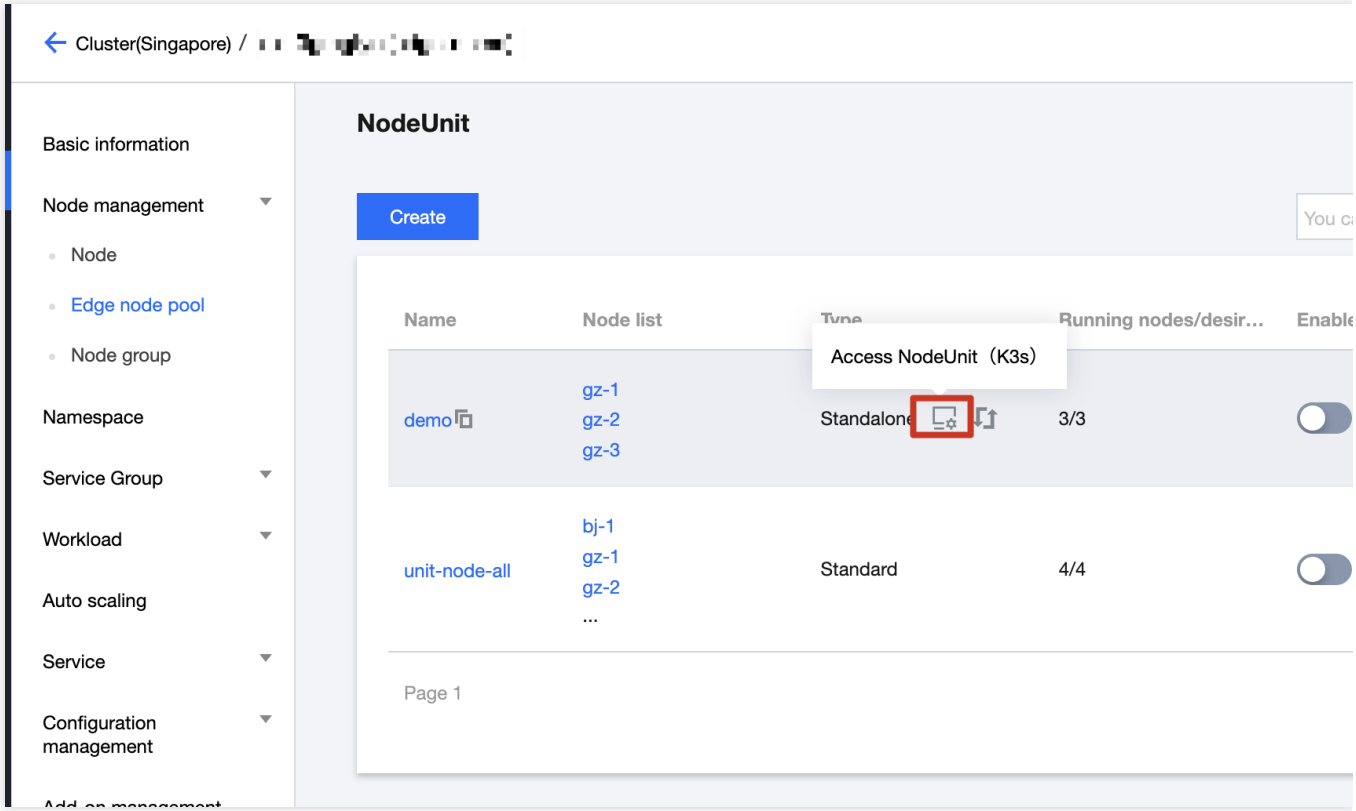
If there are only 1-2 nodes in the NodeUnit, you can only use a "single Master cluster" by default. If the NodeUnit contains 3 or more nodes, you can choose to use a "single Master cluster" or a "3 Master highly available cluster". You can make a choice based on your own needs.

7. After a moment, you'll see the type of "demo" NodeUnit will change to "Standalone":

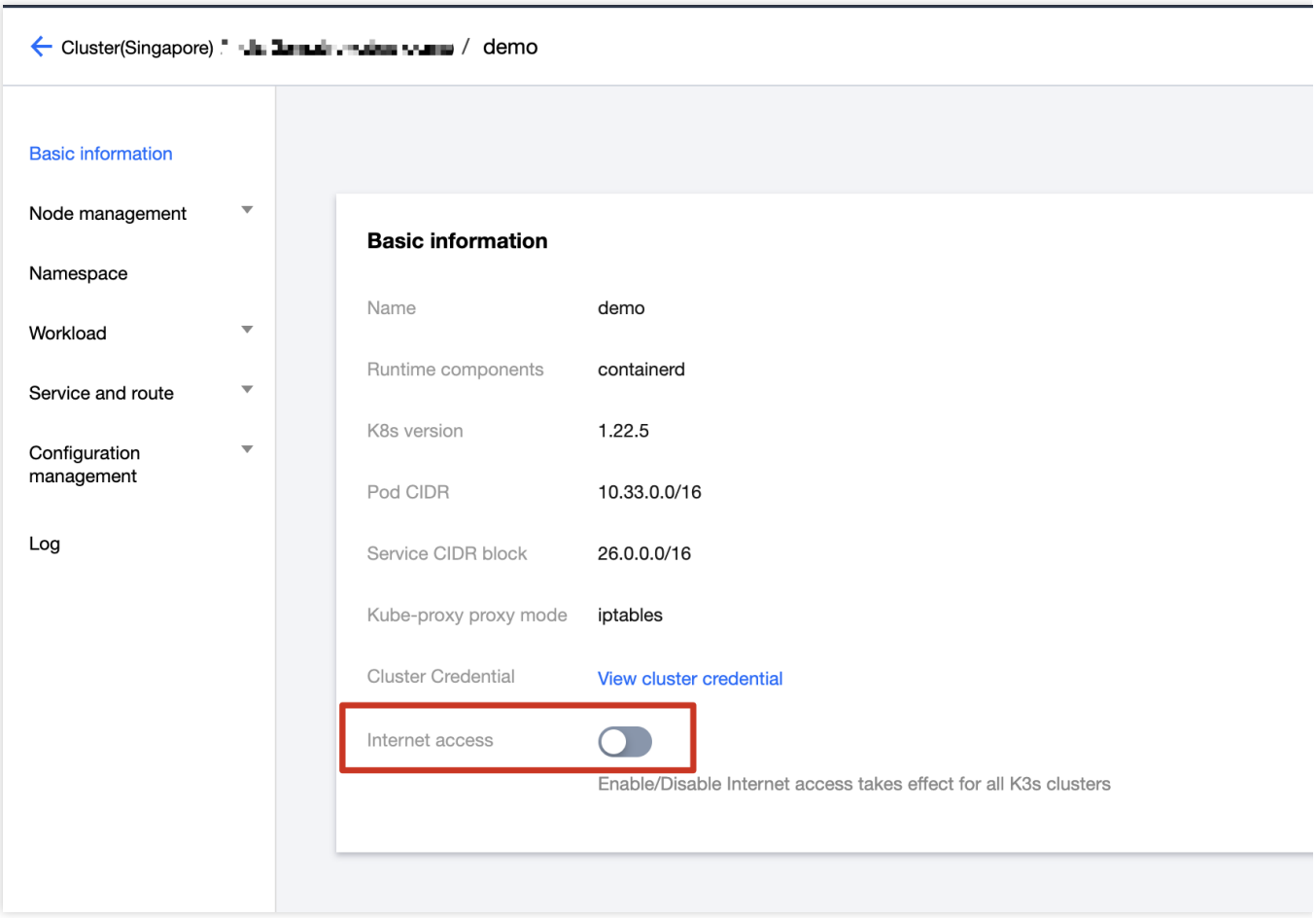


Access the edge K3s cluster

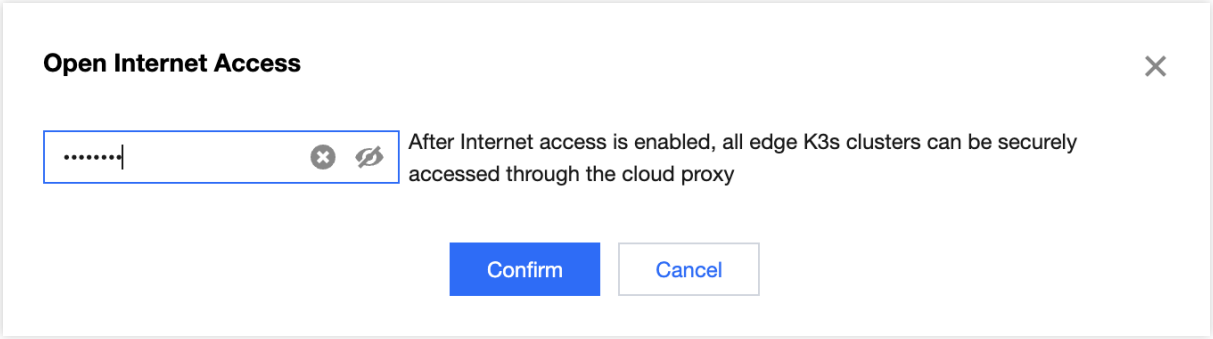
1. At the NodeUnit page list, click **"Access NodeUnit(K3s)"**, and you will be redirected to the console UI of the edge K3s cluster, as shown in the following figure:



2. In the basic information of the edge K3s cluster, click **"Internet access"** to enable the external network access capability of the edge K3s cluster. Once enabled, you can use the `kubectl` command on any client in the public cloud, access the edge K3s cluster through TKE-Edge's external network proxy capability, as shown in the following figure:



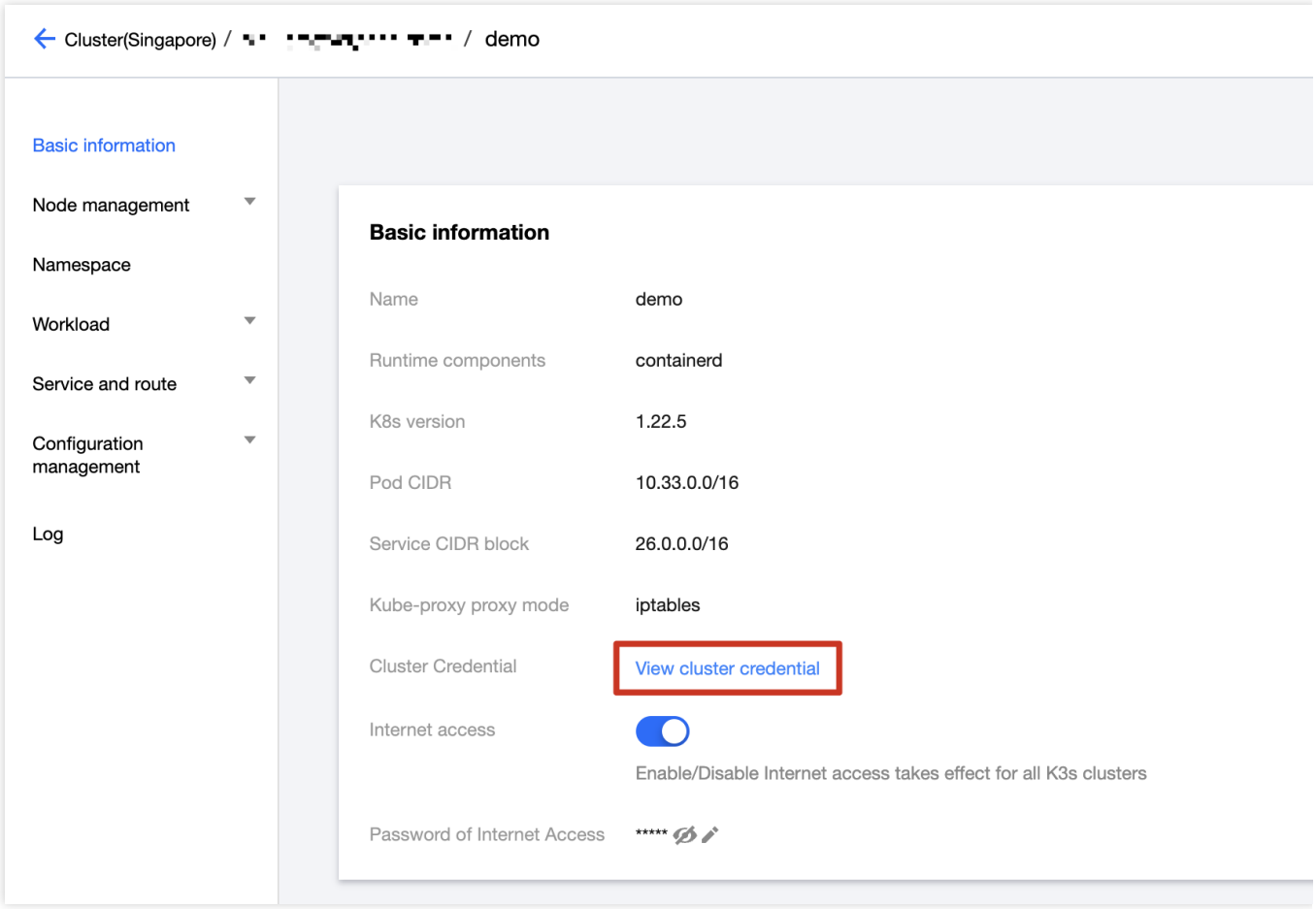
At the "Open Internet Access" page, please enter the password required for proxy access, as shown in the following figure:



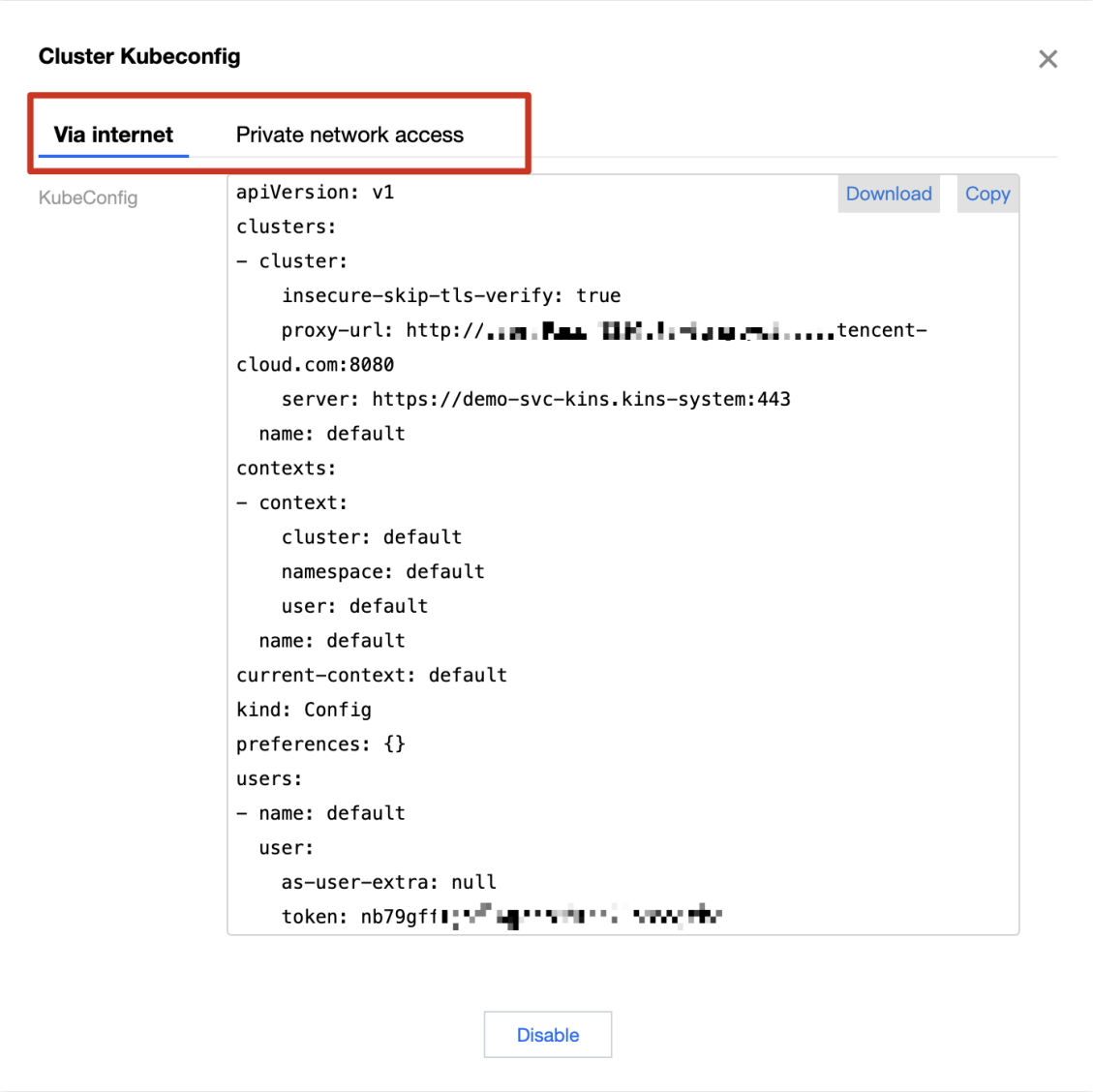
Caution：

When enabling Internet network access, the system will provide http/https proxy access through the cloud CLB capability. Therefore, users are required to set a proxy password for authentication to ensure the security of the proxy.

3. You can log in to the edge K3s cluster using either the cloud kubeconfig or the edge-side kubeconfig. Click to view the cluster credentials, as shown in the following figure:



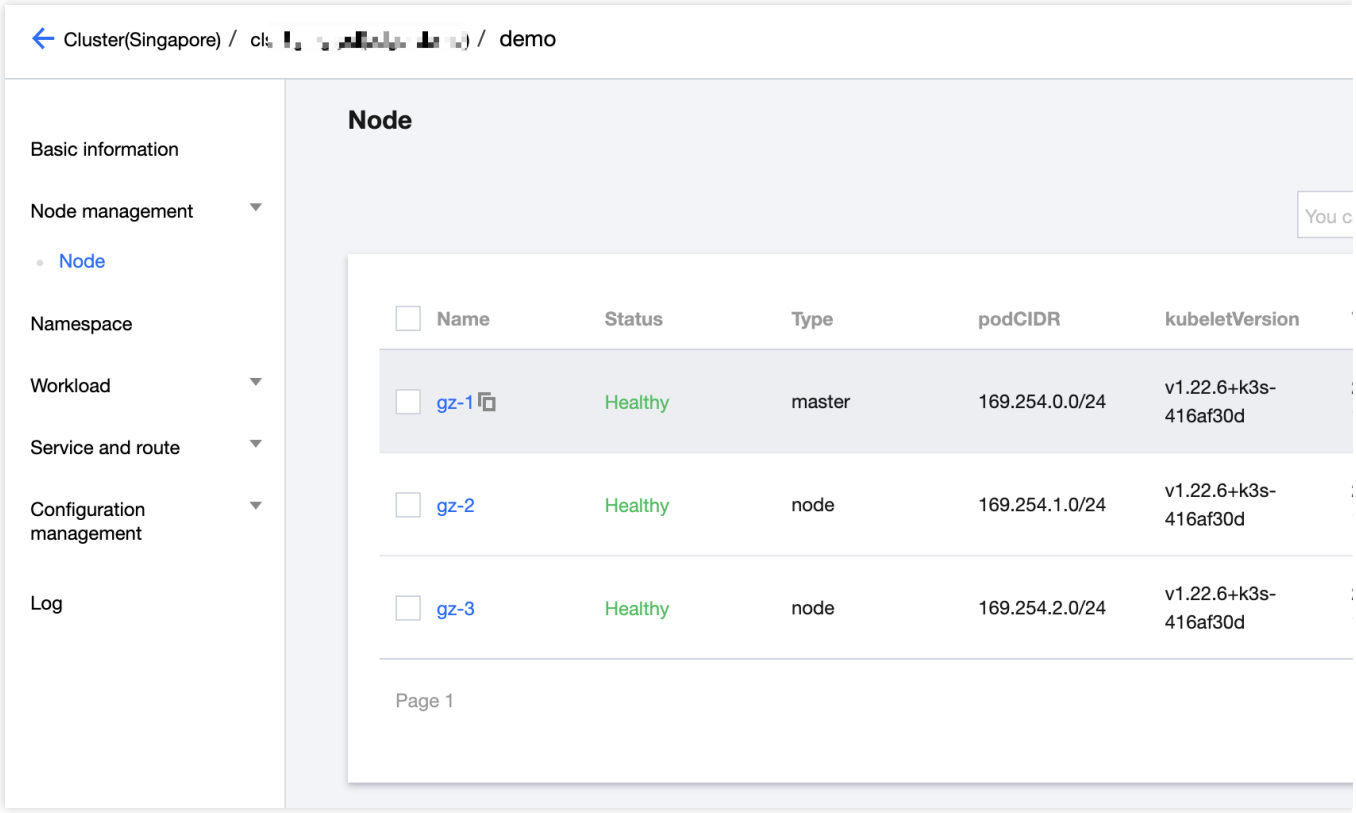
And you can choose the Internet Access or Private Network Access:



If you want to access the edge K3s cluster from the public cloud side, you can download/copy the kubeconfig through "Via internet".

If you want to directly access the local K3s cluster from the edge-side node, you can download/copy the kubeconfig for "Private network access".

4. You can also directly use the console Web UI to perform some basic operations on the edge K3s cluster, as shown in the following figure:

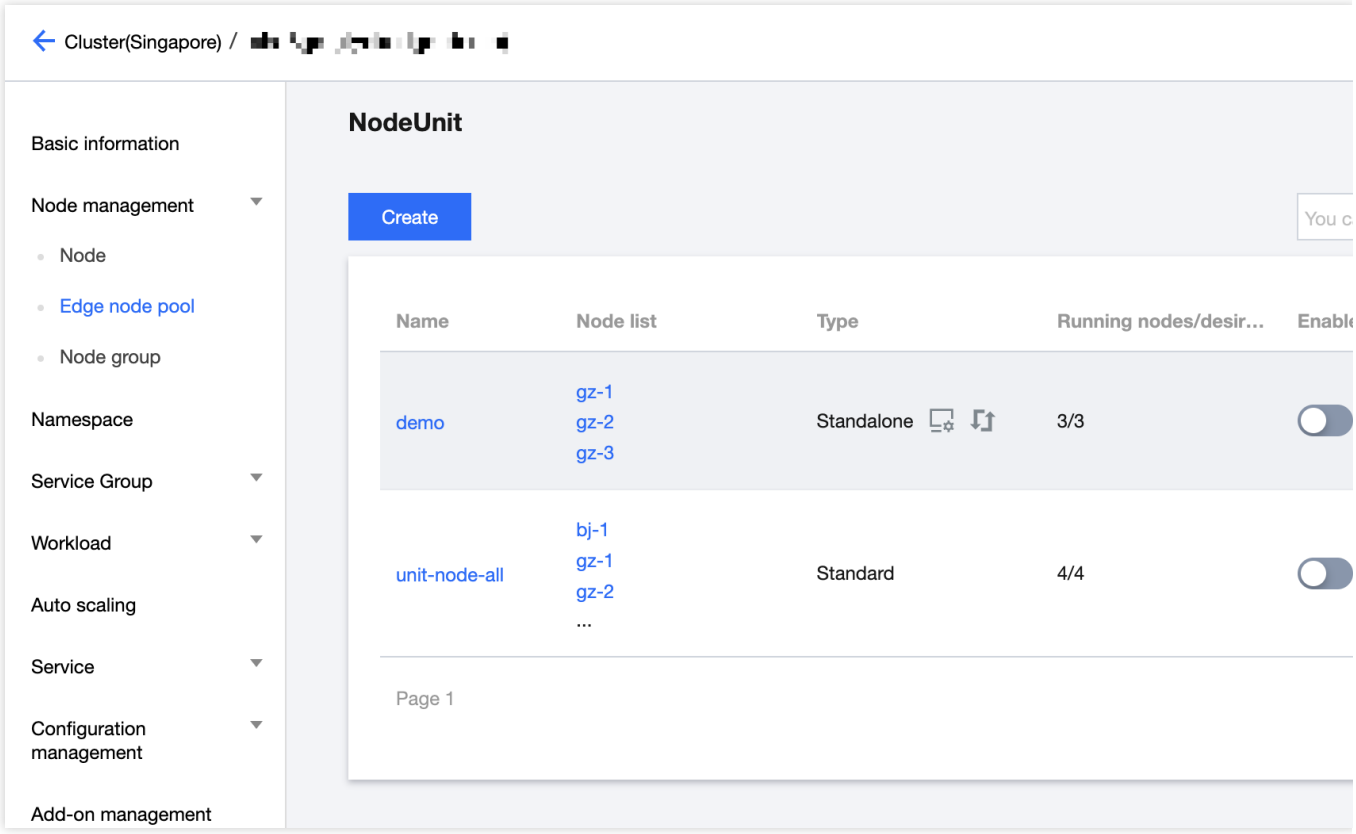


Add node to edge K3s cluster

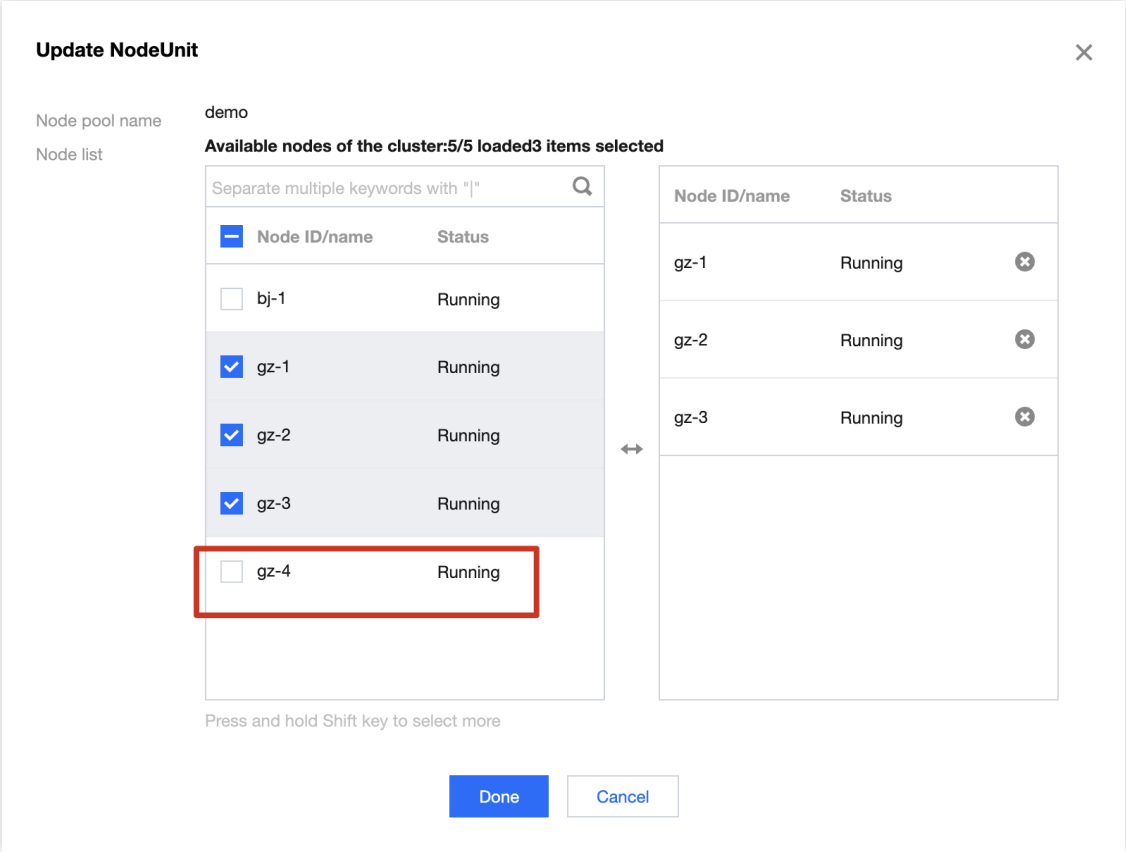
Caution：

In the current product form, users are temporarily only supported to add new nodes to the edge K3s cluster through the Cloud console side, and independent node addition is not supported when the K3s cluster is offline. Users can directly add nodes to the NodeUnit through the cloud, and the new nodes will be automatically joined to the edge K3s cluster.

1. On the NodeUnit list page, select the desired NodeUnit and click Update Configuration, as shown in the following figure:

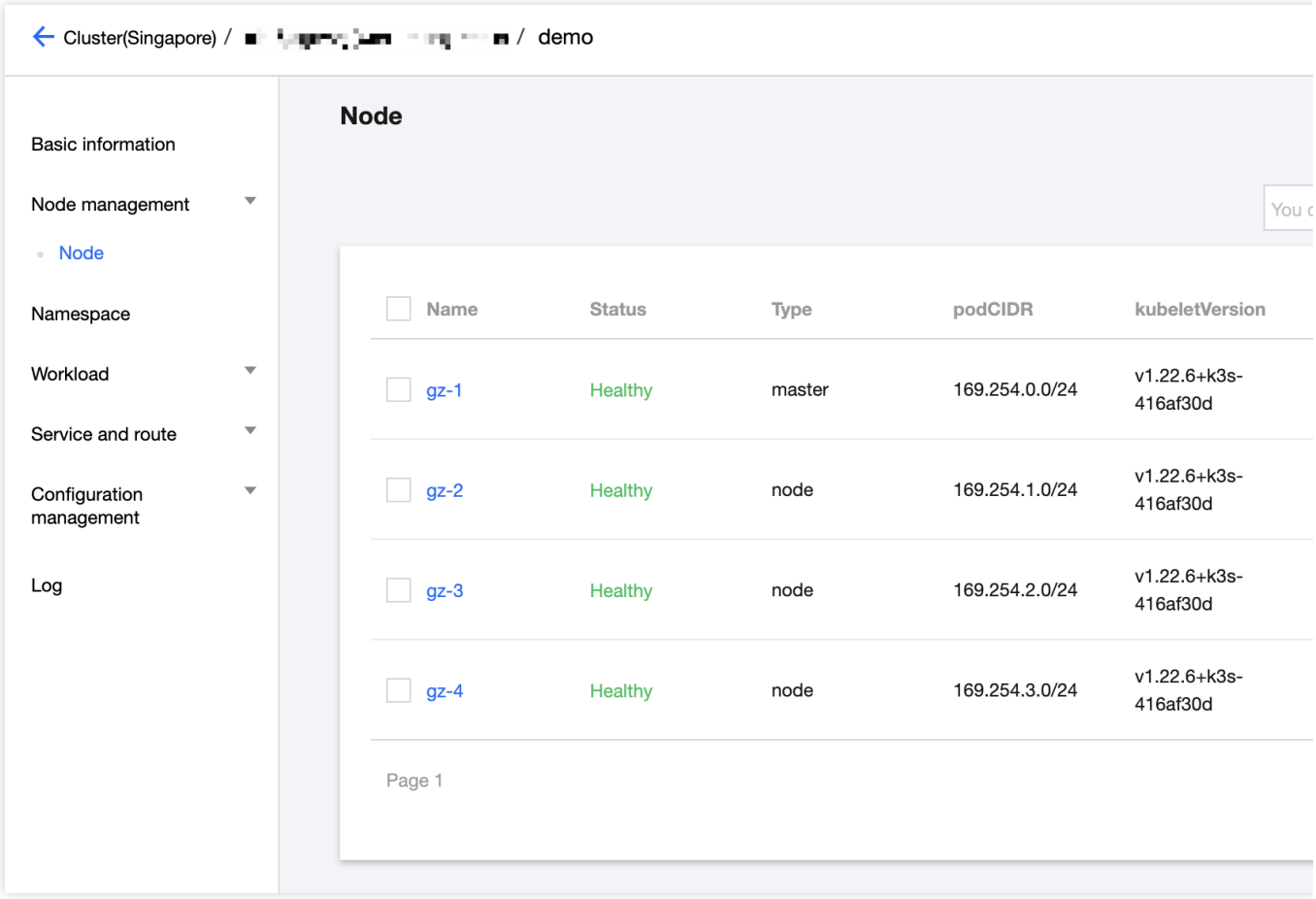


2. At "Update NodeUnit" page, add the new node **"gz-4"**, as follow:



3. Click "Done" and you'll see "gz-4" has already been in the NodeUnit "demo".

4. After waiting for a while, click "Access NodeUnit" button and enter edge K3s console page. Observe the cluster node information, and you will find that the added node has joined the edge K3s cluster, as shown in the following figure:



5. Through interactive operation with NodeUnit on the cloud, you can now freely add computing nodes to the edge K3s cluster.

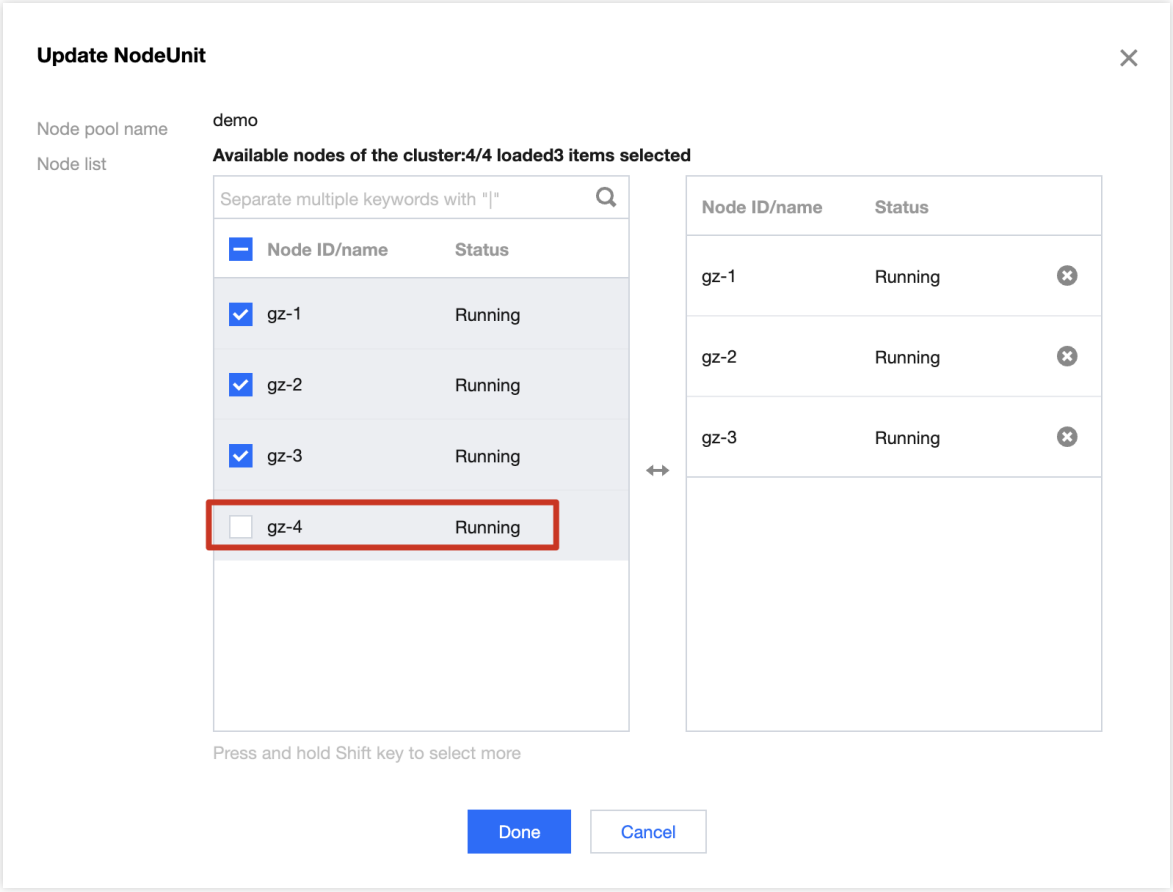
Delete node from edge K3s cluster

At present, this product only temporarily supports users to delete edge nodes in the K3s cluster **via the cloud**. There are two ways for users to remove nodes from the edge K3s cluster.

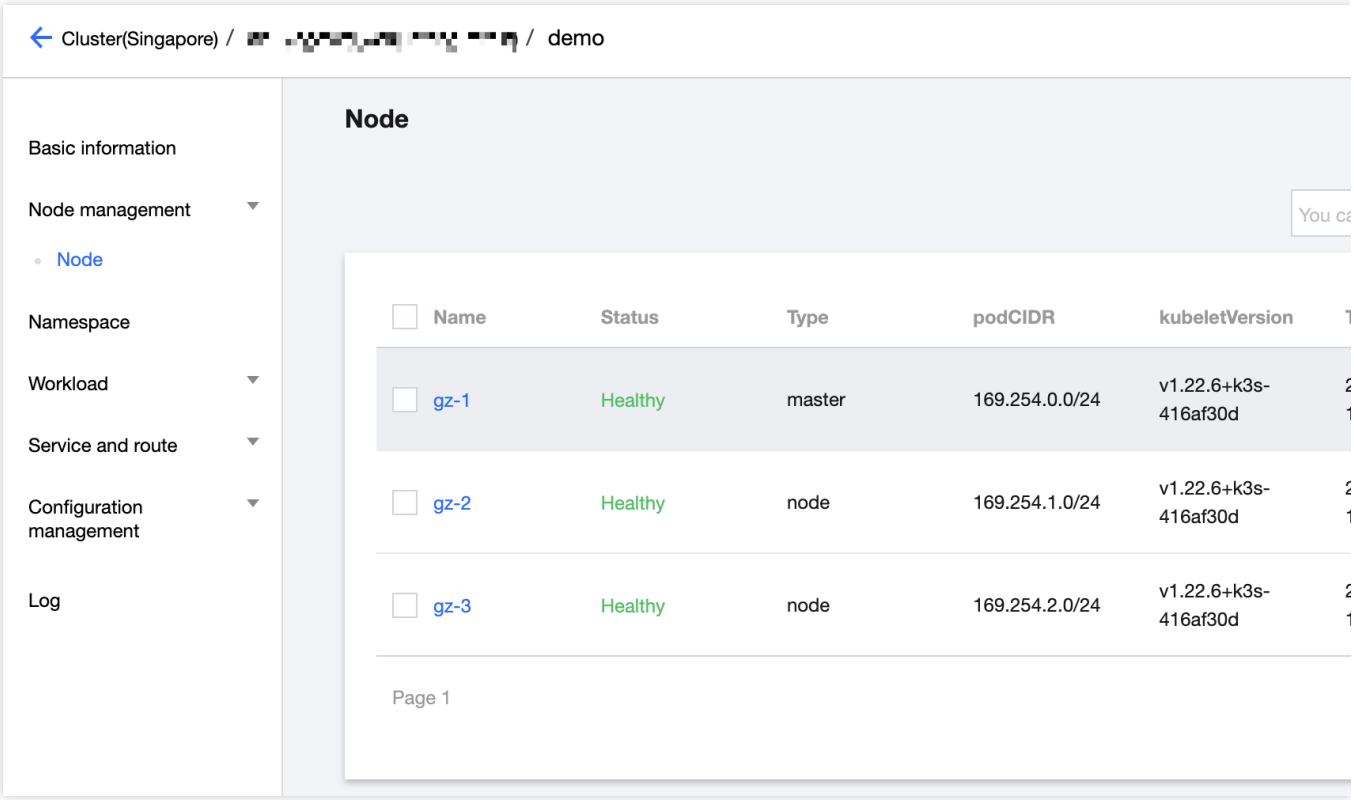
Method 1

If users only want to clear the corresponding node from NodeUnit while not needing to remove it from the TKE-Edge cluster, they just need to modify the NodeUnit's node configuration. As shown below, simply deselect the corresponding node:

1. On the NodeUnit list page, select the specified NodeUnit and click on **"Update configuration"**.
2. On "Update NodeUnit" page, deselect the specified node as shown in the image below:



3. Click on "Done", and you can observe the edge K3s cluster node status. The corresponding node will be cleared, as shown in the following image:



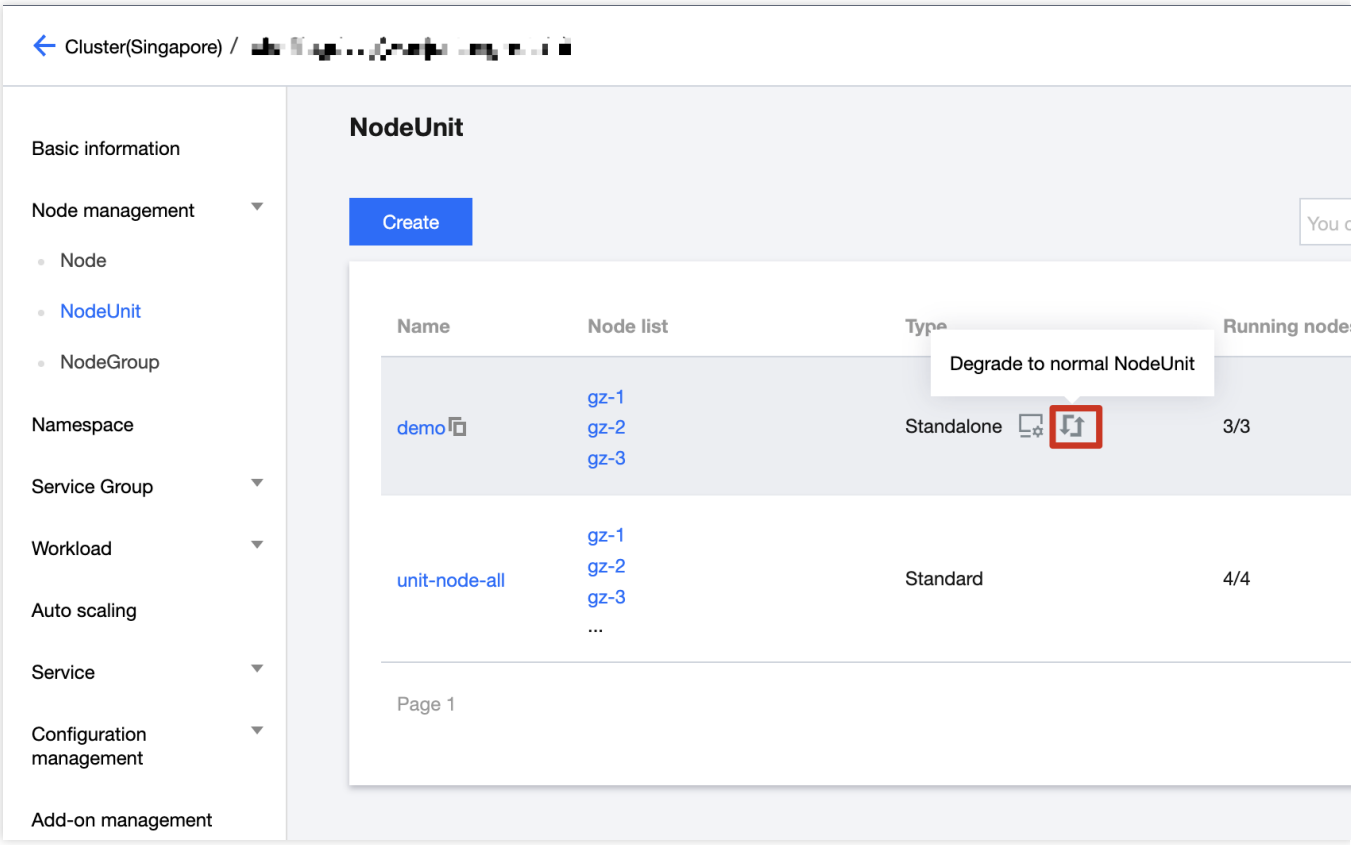
Method 2

If a user wants to delete a node on TKE-Edge and no longer use it, they can directly remove the corresponding node from the TKE-Edge node page. The corresponding node in the edge K3s cluster will also be cleaned up accordingly.

Degrade edge K3s cluster

If a user no longer needs the edge K3s standalone cluster and wishes to revert to using the standard TKE-Edge edge NodeUnit, follow these steps:

1. On the NodeUnit list page, select the NodeUnit that needs to be downgraded, and click on **"Degrade to normal NodeUnit"**. As shown in the image below:



2. After being degraded to a standard node pool, its default behavior will be consistent with that of a standard NodeUnit.

Delete edge K3s cluster

Caution：

By default, the TKE-Edge product only allows users to delete standard NodeUnit from the interface. The delete button for independent node pools is disabled by default, so users are not permitted to directly delete "Standalone" NodeUnit. If you wish to delete an independent node pool, you must first perform the edge K3s cluster degrade operation to downgrade the standalone NodeUnit to a standard NodeUnit. After that, you can delete it from the NodeUnit page.

Edge Platform Capability Monitor

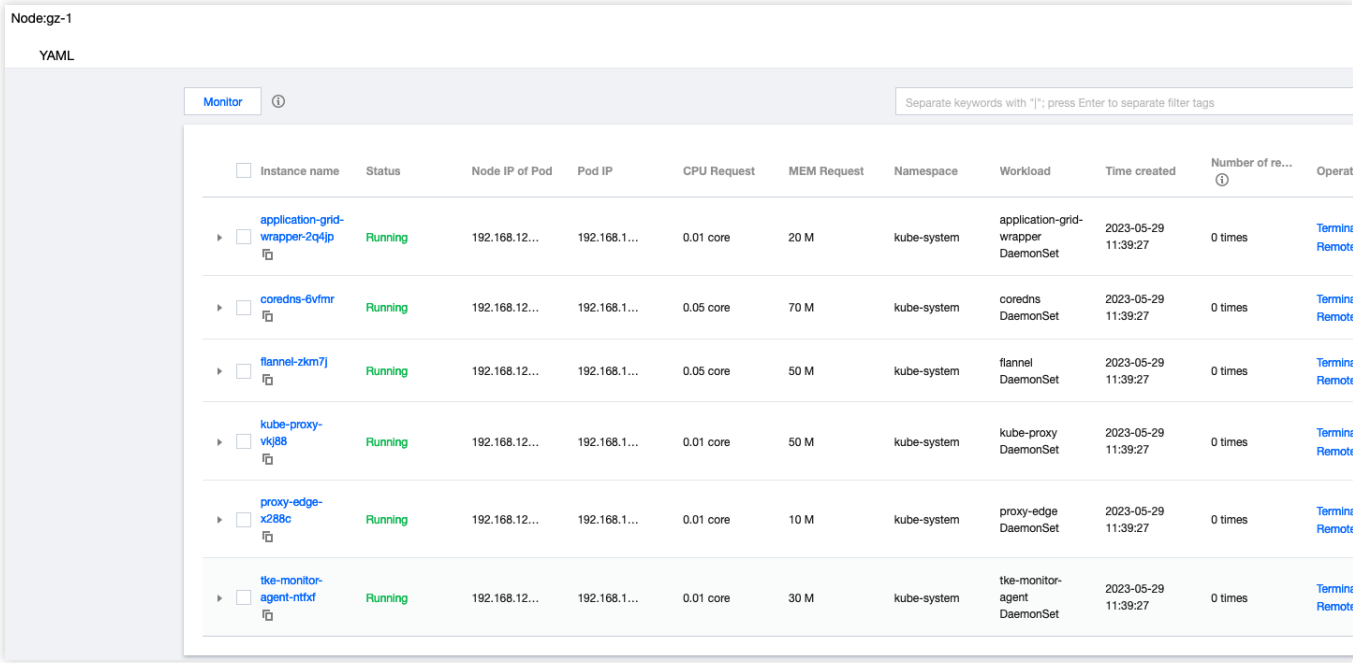
Last updated：2023-06-01 11:22:54

The monitoring component capability is a basic supporting component of a platform system. The edge container reuses Tencent Cloud's cloud monitoring capabilities, which can upload the edge-side monitoring metrics to the cloud through the edge container's "Tunnel Cloud-Edge Channel" providing basic monitoring capabilities. You can view the cluster monitoring information according to the following operations:

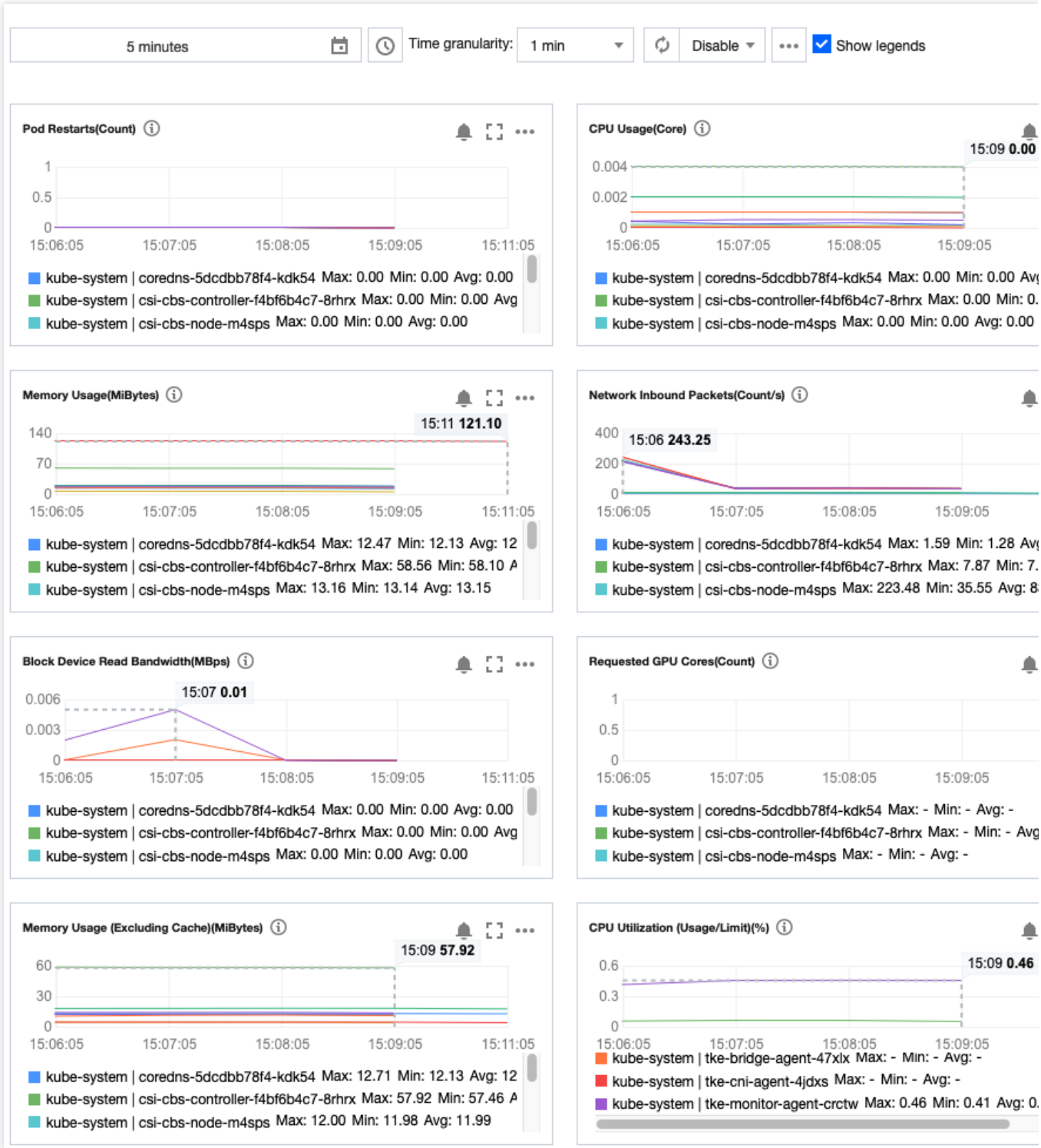
Operation Steps

Basic Monitoring Information

1. Log in to the Tencent Kubernetes Engine, and select the Cluster option in the left-side navigation menu.
2. On the Cluster Management page, click on the Edge Cluster ID to enter the cluster details page.
3. Select Node Management > Nodes on the left side of the page to enter the node list page, and click on Monitoring to view specific monitoring information, as shown in the figure below:



4. View monitoring information, as shown in the figure below:



The specific monitoring Metrics include:

Events: Pod restart count, Pod Ready status.

CPU: CPU usage (cores), CPU utilization (percentage of limit), CPU utilization (percentage of node), CPU utilization (percentage of request).

Memory: Memory usage (MBytes), memory usage (excluding cache), memory utilization (percentage of node, excluding cache), memory utilization (percentage of request), memory utilization (percentage of request, excluding cache), memory utilization (percentage of limit), memory utilization (percentage of limit, excluding cache), memory

utilization (percentage of node), memory usage_working_set, memory utilization_working_set percentage of request, memory utilization_working_set percentage of limit, memory utilization_working_set percentage of node.
Network: Network input packet rate (packets/s), network output traffic (MBytes), network input traffic (MBytes), network output bandwidth (Mbps), network input bandwidth (Mbps), and network output packet rate (packets/s).
GPU (if included): GPU memory usage rate (%), GPU encoding resource usage rate (%), GPU decoding resource usage rate (%), and GPU stream processor usage rate (%).
Storage: Rootfs space size (MiB).

Note
The "Basic Monitoring" capability provided by the edge container is currently available to users for free.

Cloud-native Monitoring

There are some shortcomings in the platform's built-in monitoring information:
Metrics are limited, and custom metrics cannot be defined.
It lacks monitoring capabilities for cloud-native related Metrics.
Therefore, the edge container service also supports the integration of Prometheus cloud-native monitoring capabilities. With this product, the edge container platform can support custom metrics and comes with free node metrics (node-exporter), kubelet metrics, and cluster-level kube-state-metrics information. This provides a complete cloud-native monitoring capability for edge clusters. For more information, refer to Prometheus Monitoring.
Note
This product is a paid product. Please understand the detailed features and purchase it as needed.

Log

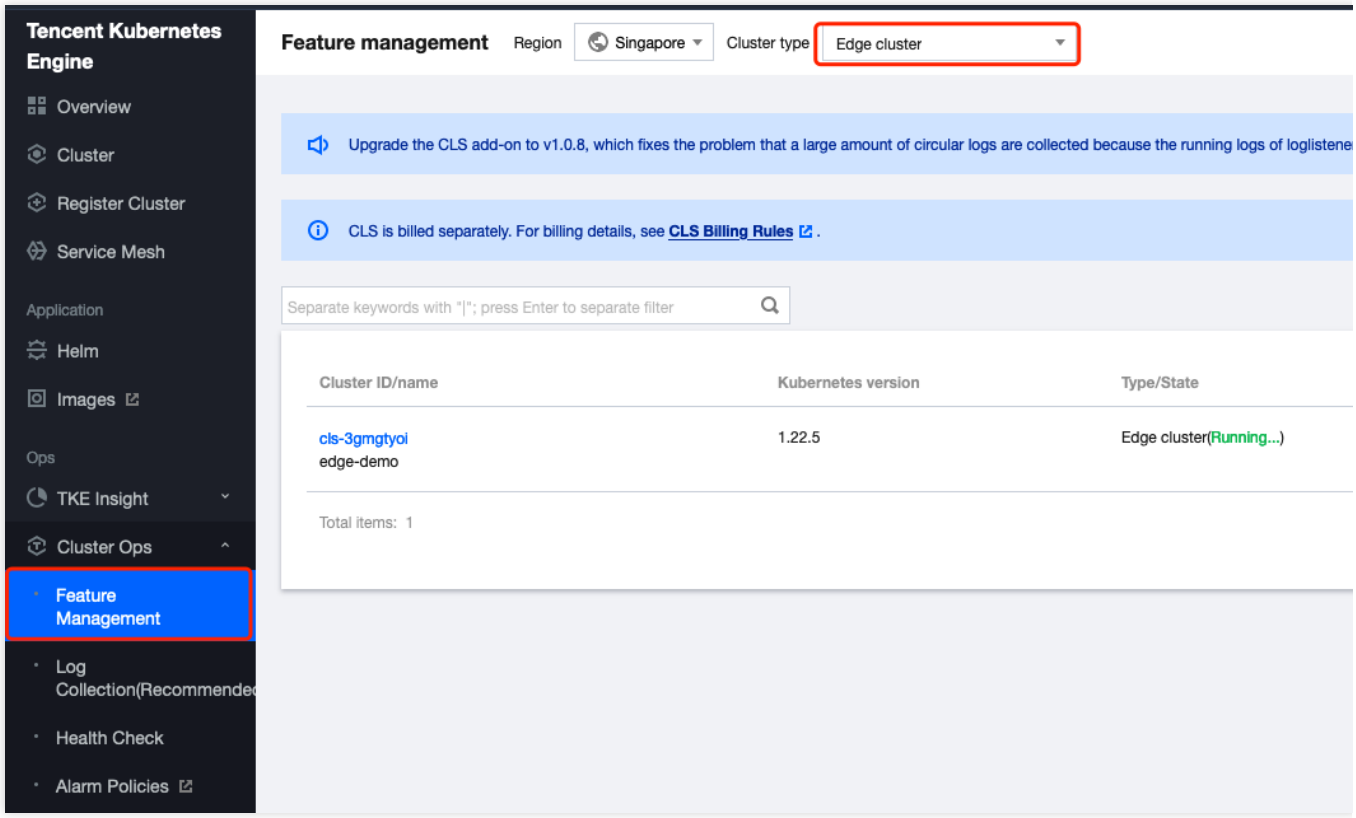
Last updated：2023-06-01 11:22:54

The logging component capability is a basic supporting component of a platform system. The edge container reuses Tencent Cloud's logging service, which can upload the edge-side log data to the cloud, providing a complete log collection and retrieval capability. You can enable the log collection function according to the following operations:

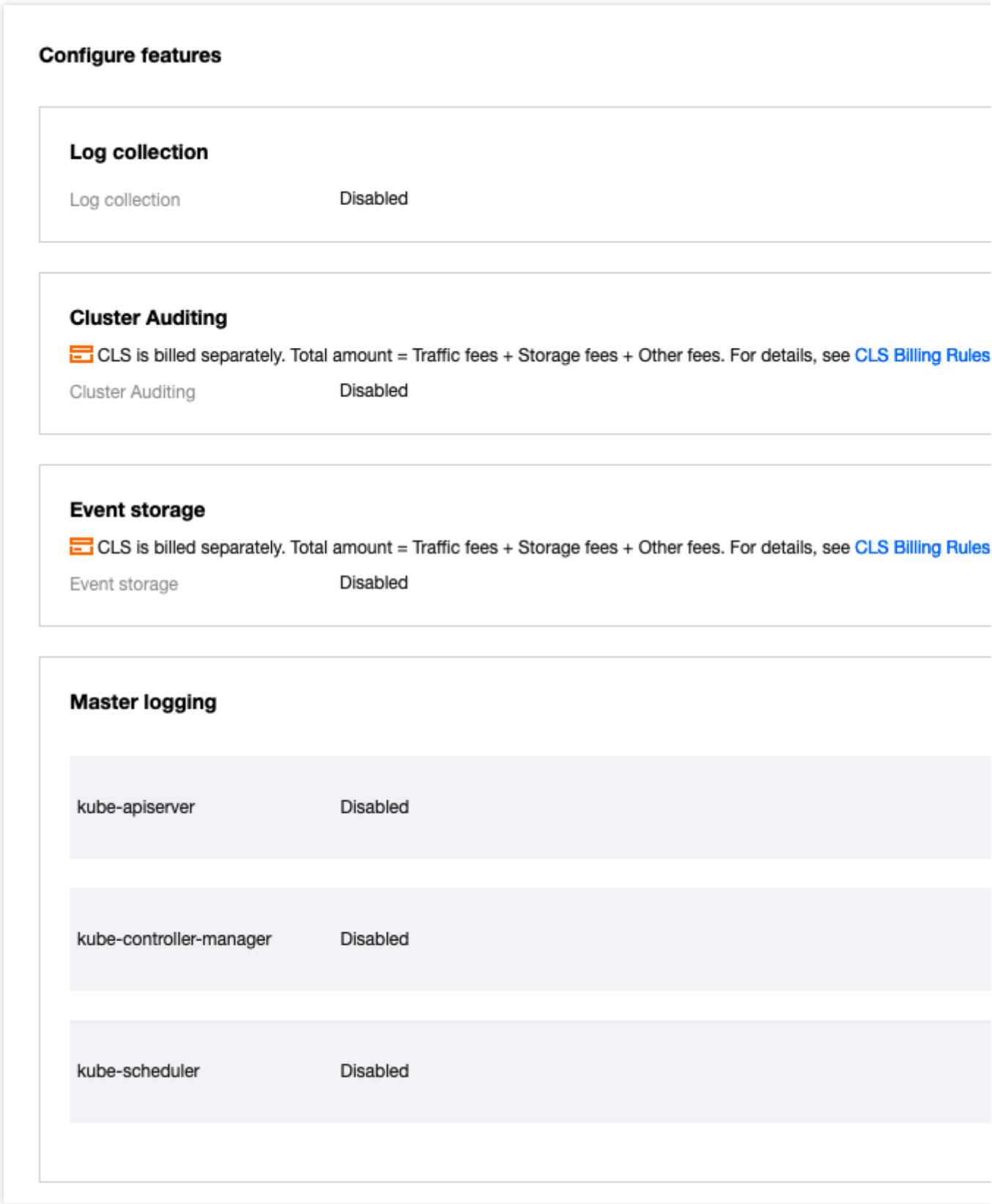
Operation Steps

Enable Edge Container Logging Capability

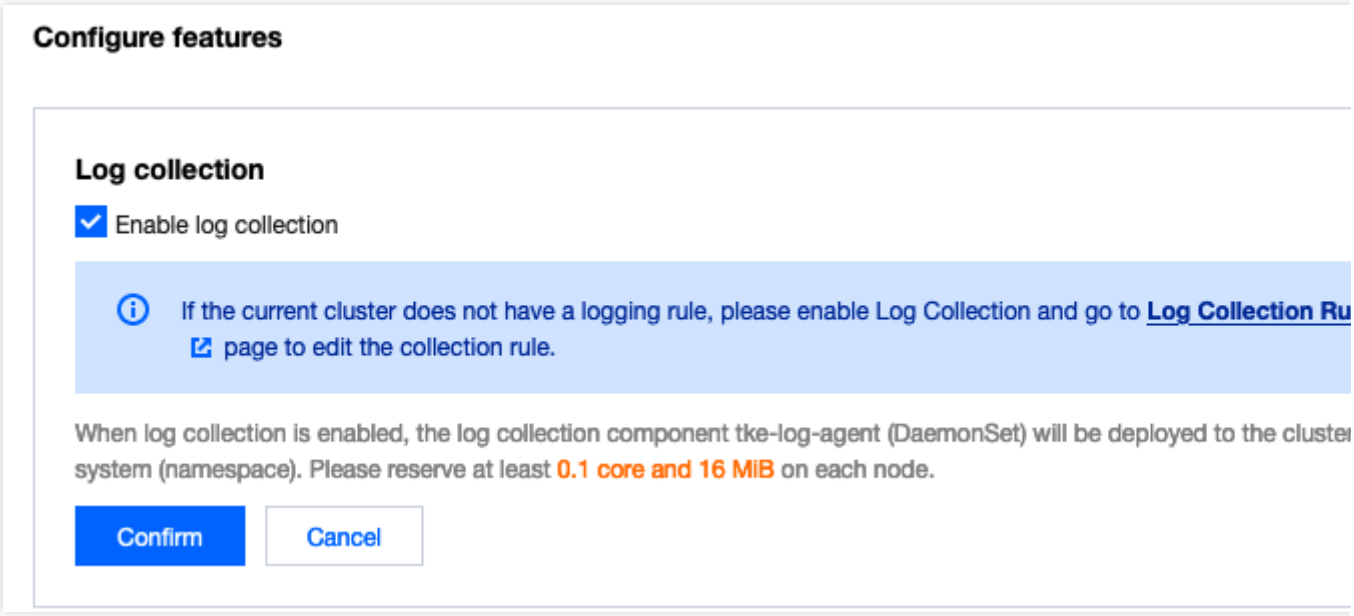
1. Log in to the Tencent Kubernetes Engine, and select Operation Management > Feature Management in the left-side navigation menu.
2. In the "Feature Management" page, select the region and cluster type at the top. As shown in the figure below:



3. Click the settings on the right side of the cluster for which you want to enable log collection.
4. On the "Set Features" page, click Edit on the right side of Log Collection. As shown in the figure below:



Check the "Enable log collection" and then click Confirm. As shown in the figure below:



5. After enabling, you can view or retrieve logs in the Log Service Console. For more details, please refer to the Cloud Log Service.

O&M Management

Edgectl script manual

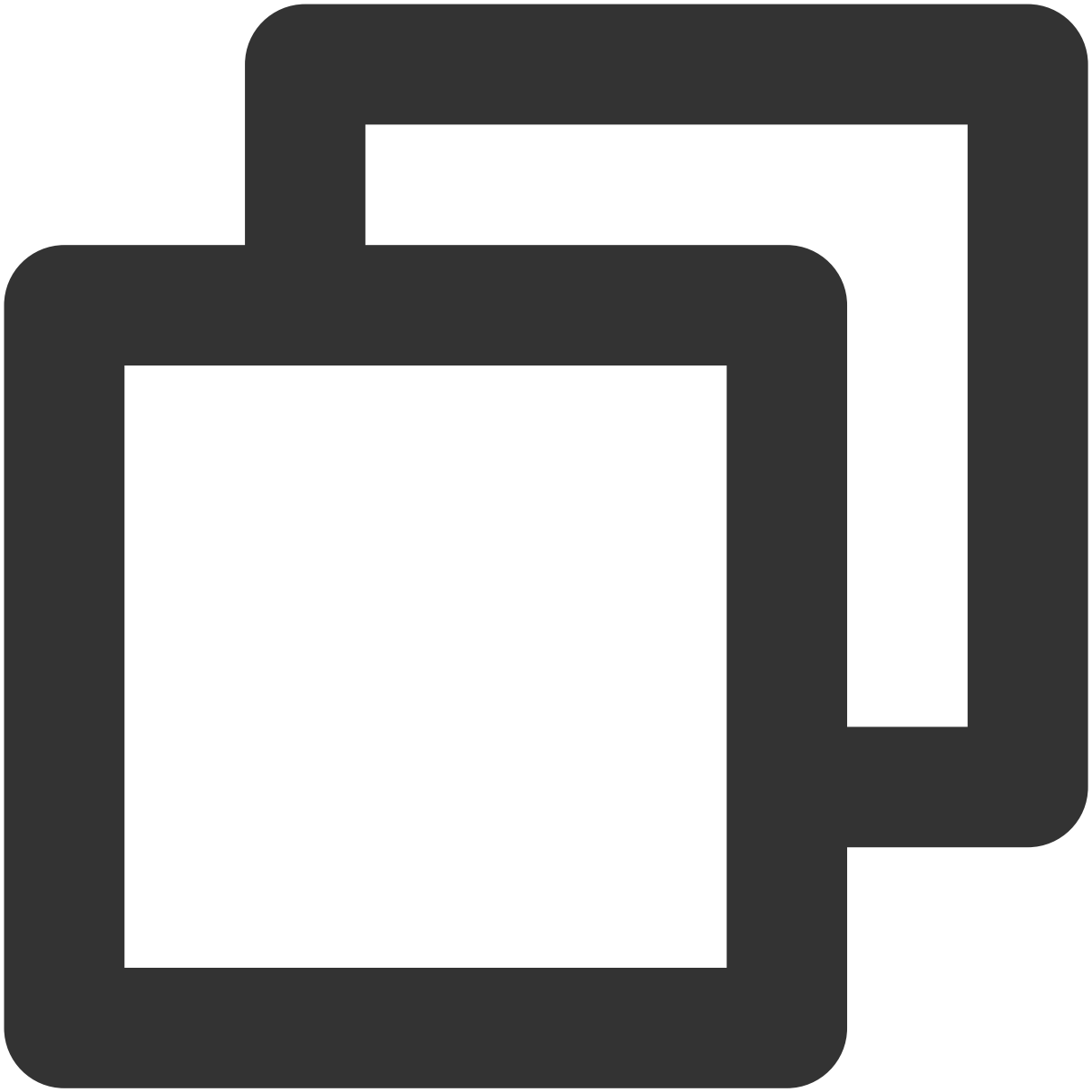
Last updated：2023-06-01 11:53:42

Edgectl is an edge node management tool used for remotely adding edge nodes. You can obtain this tool by going to Node Management > Add Nodes via Script.

Edgectl provides the following three commands:

Command	Description
edgectl check	Check if the node meets the conditions for installing an edge node.
edgectl clear	Clean up the edge node.
edgectl install	Install the edge node.

The sample code is shown below:



```
# ./edgectl -h
Usage:
   edgectl command [flags]
Available Commands:
   check        Check the edge node if to be add to clusters
   install      Install components to edge node
   clear        Clear edge node and recovery as usual

Flags:
   -h, --help   Help for edgectl
```

edgectl check command

Meaning

Check if the node meets the conditions for installing an edge node. The check project includes the following:

Check if the user is the root user.

Check if the system is within the supported range.

Check if the swap area is turned off.

Check if the firewall is turned off.

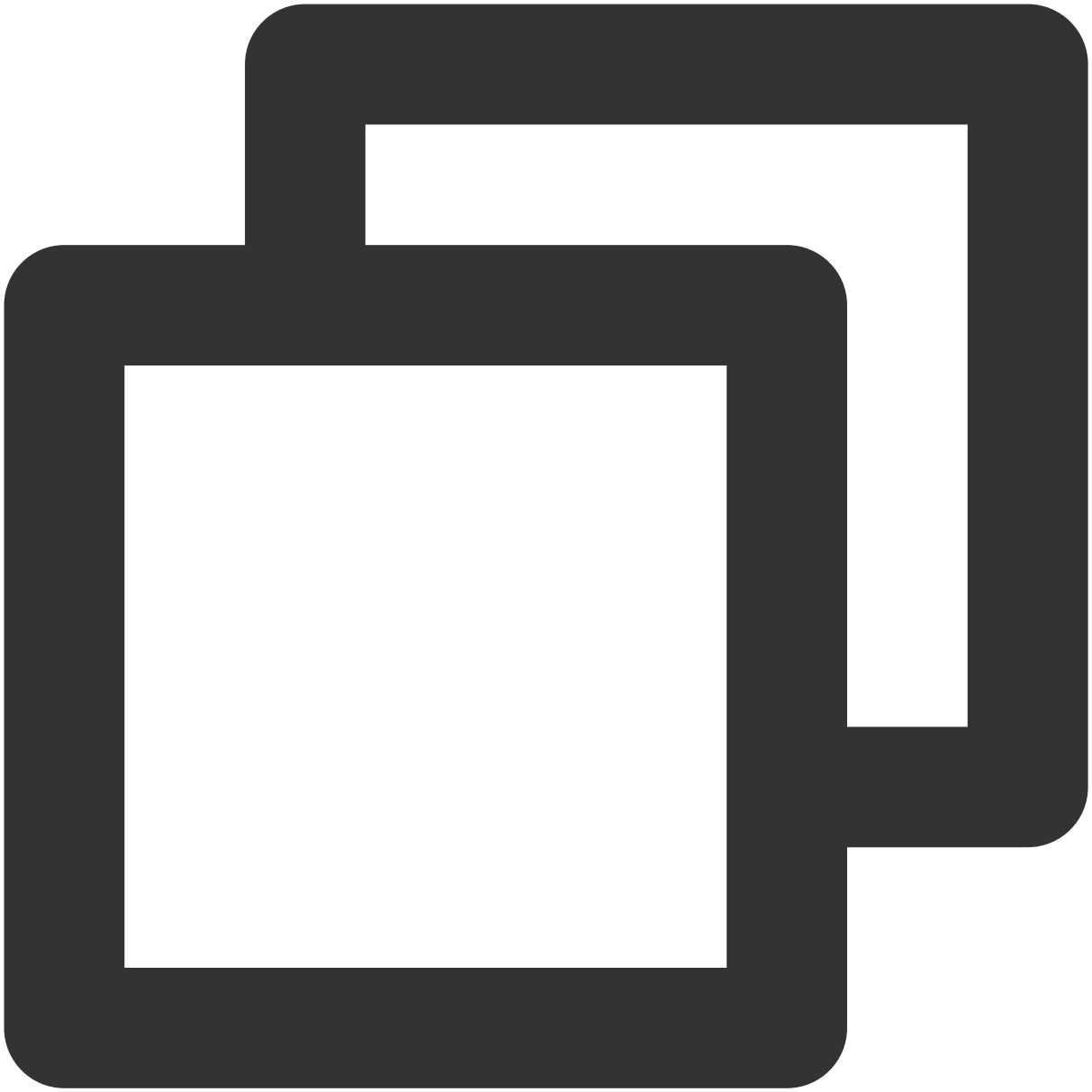
Check if ufw is turned off.

Check if the ports are occupied (the ports checked are 1443 and the ranges 10249-10259 and 51000-51020).

Check if cgroup memory is enabled.

Check if the node has previously installed kubeadm, docker, kubelet, and kubectl.

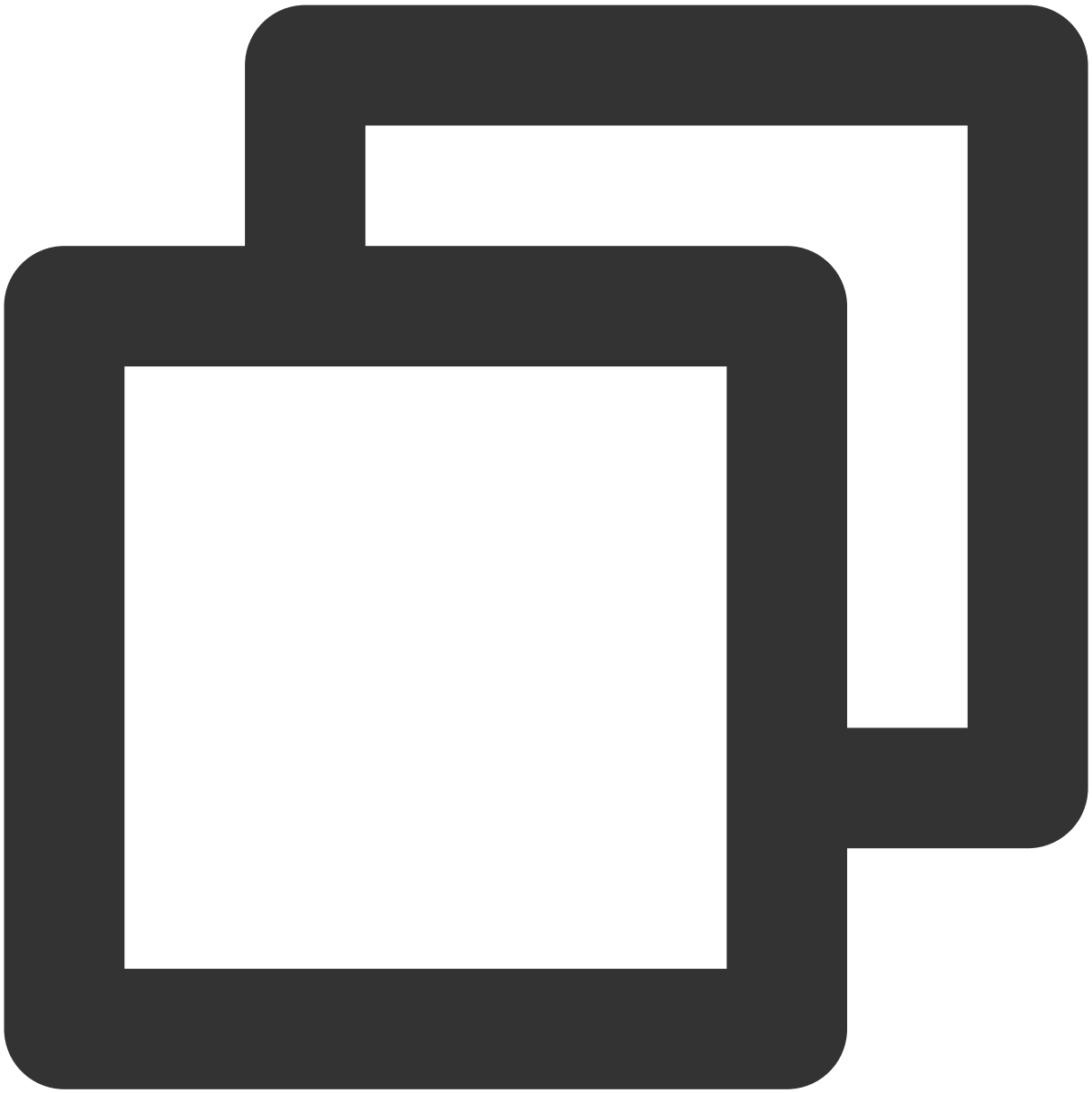
Usage example



```
# ./edgectl check
Unit firewalld.service could not be found.
 WARN >> Port: 1443 occupied. Please turn off port service.
...
 WARN >> The machine is not clean. Please reinstall the system.
/usr/bin/kubelet
...
 >> Check Environment Finish! <<
```

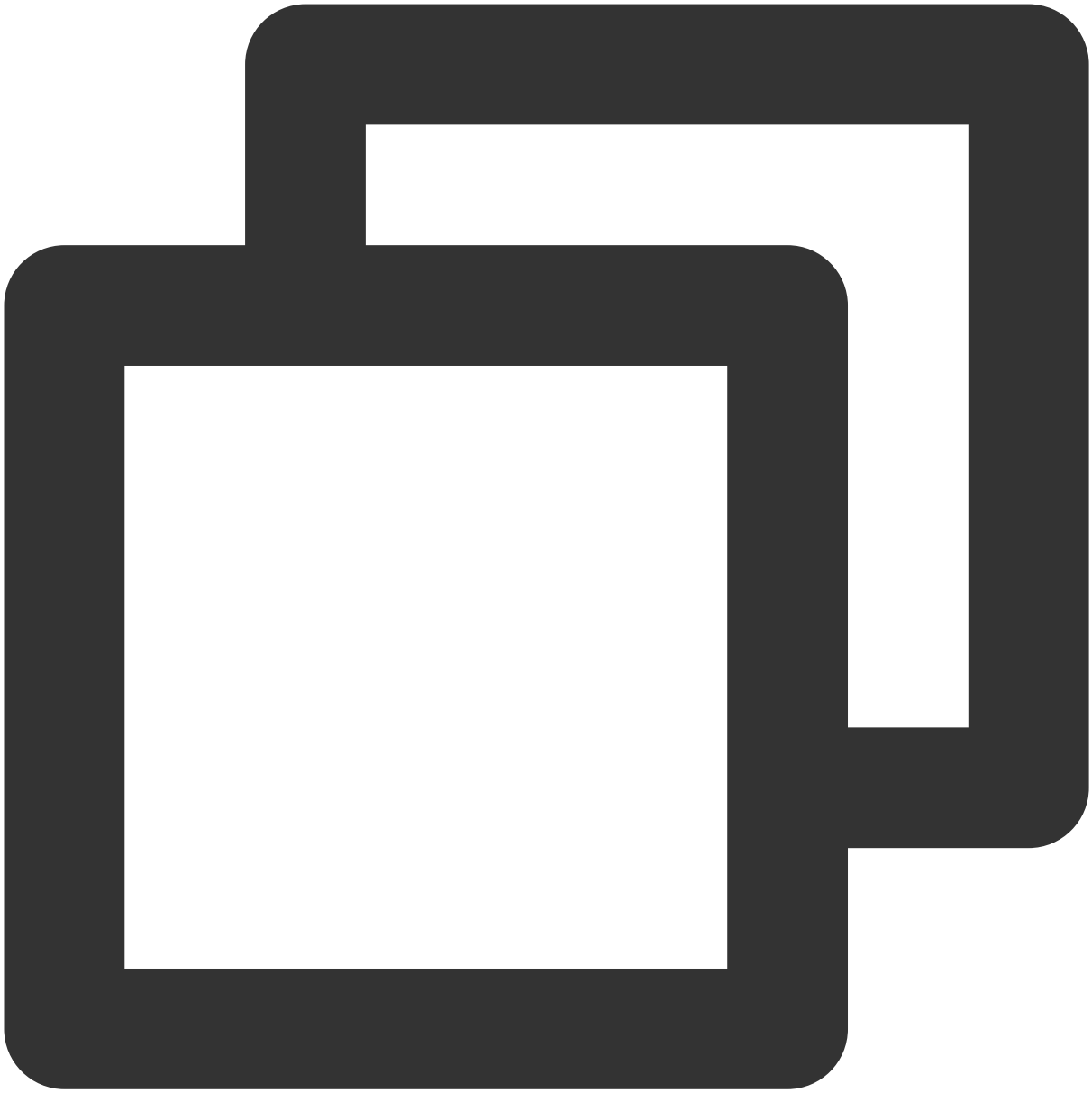
For instance, after executing the sample code this time, the user is prompted with the following 2 risk items:

The edge node's port 1443 is already in use, please shut down its service.



```
WARN >> Port: 1443 occupied. Please turn off port service.
```

The node has kubelet remnants. It is recommended to reinstall the node system, or execute the edgectl clear command to perform the cleanup operation.



```
WARN >> The machine is not clean. Please reinstall the system.
/usr/bin/kubelet ## The kubelet has been installed previously
```

edgectl clear command

Meaning

Clean up the edge node. The clear command will remove the following information:

Remove all containers and Pods running on the edge node.

Stop kubelet, lite-apiserver, and docker.

Remove created network information and routing information.

Delete the following folders or files:

/etc/kubernetes

/etc/docker

/root/.kube/config

/var/lib/kubelet >/dev/null 2>&1

/var/lib/cni

/etc/cni

/etc/sysconfig/kubelet/

/etc/sysconfig/lite-apiserver

/data/lite-apiserver >/dev/null 2>&1

/usr/lib/systemd/system/{kubelet, docker, lite-apiserver}.service

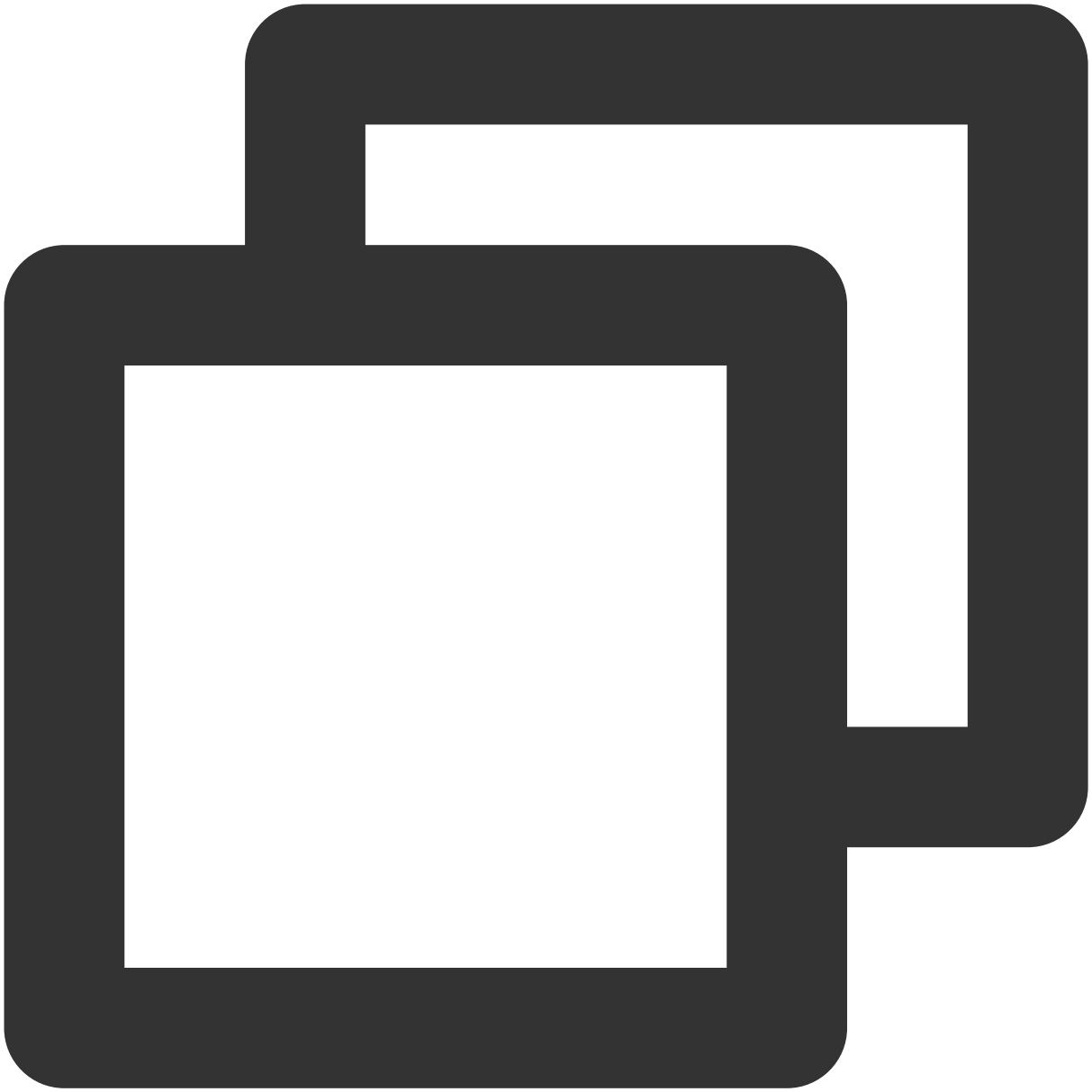
Note：

`edgectl clear` will remove all containers and Pods on the node, please execute with caution.

`edgectl clear` will remove relevant folders or files, please back up important materials in advance.

The `edgectl install` command will execute `edgectl clear` by default. Before executing `edgectl install`, please consider the risk items removed by edgectl clear.

Usage example

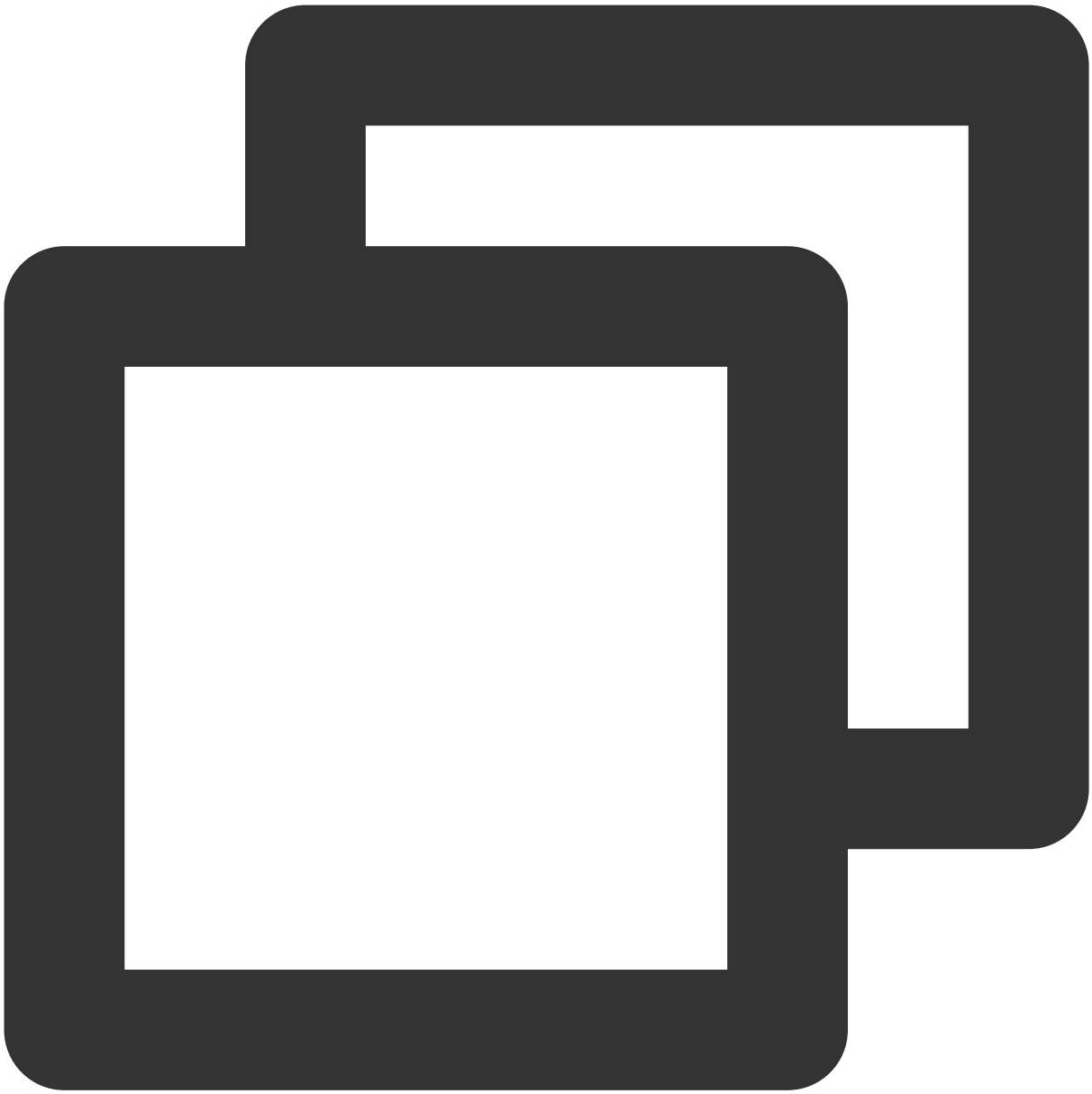


```
# ./edgectl clear
removed '/etc/kubernetes/cluster-ca.crt'
...
  >> Clear Node Complete! <<
```

edgectl install command

Meaning

Install the edge node. Execute the following command to install the edge node.



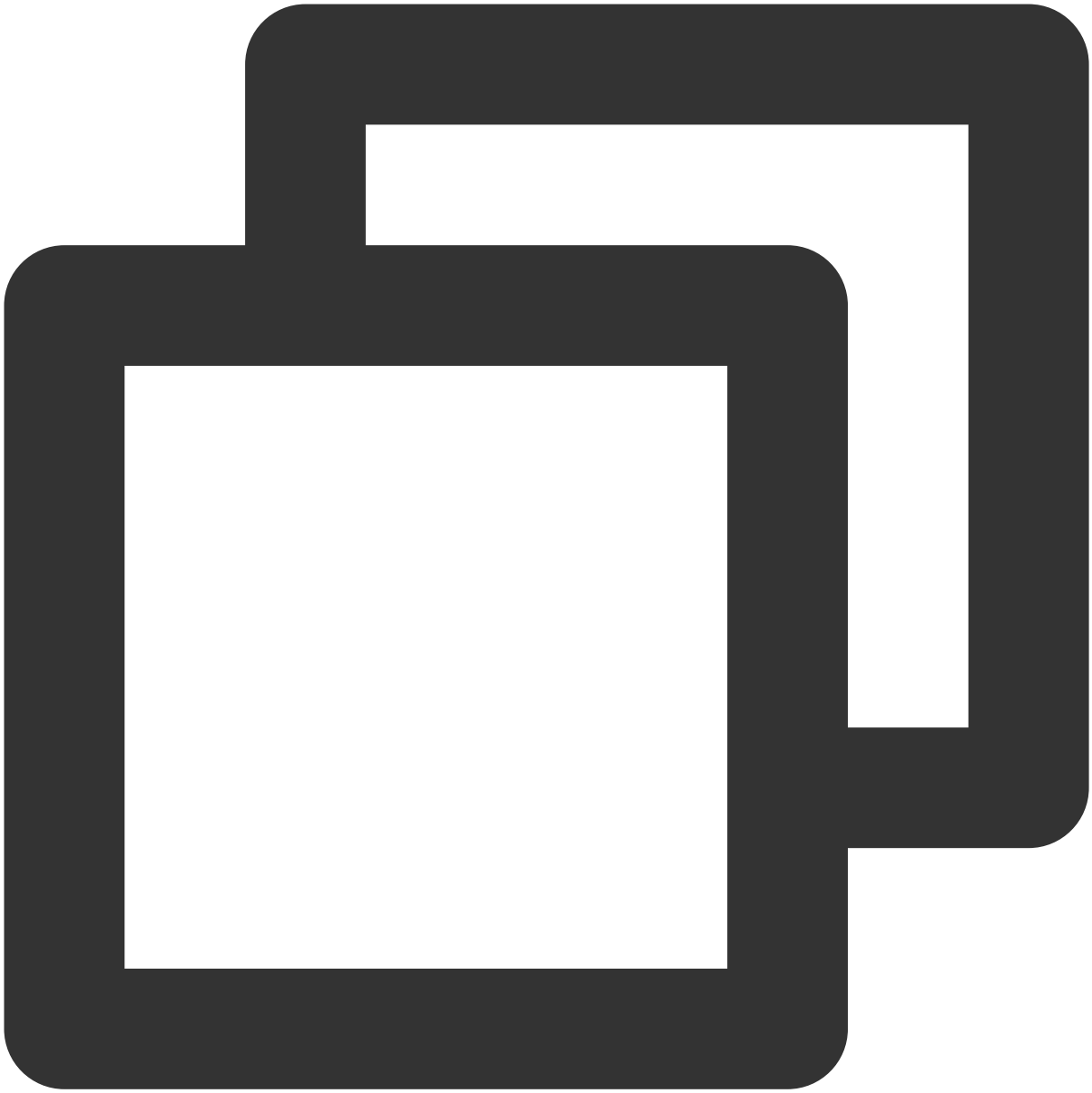
```
# ./edgectl install -h
Usage:
   edgectl install [flags]
Flags:
   -n,  --node-name      Node name in edge cluster. Must！## Node name, required
   -i,  --interface      Default network interface name. ## Node's default network i
```

--node-name：Edge node name, abbreviated as -n.

The name is a required option and must strictly follow the Kubernetes node naming conventions.

Please ensure that the node name is unique within the cluster being added, otherwise, it will cause a problem of continuous switching of nodes with the same node name.
Node names do not support using IP addresses directly, otherwise it will cause the kubectl log and kubectl exec commands to become invalid.
--interface：The default network interface name for the edge node, abbreviated -i.
Optional, the default network interface name of the edge node. Entering it incorrectly will cause the flannel and coredns components to malfunction.
The default value is the specified network interface name entered from the Tencent Cloud page, and specifying `--interface` will override the original default value.

Usage example



```
# ./edgectl install  --node-name node-192.168.67.91 --interface eth0
NOTE:
      input: [ edgectl install --node-name node-192.168.67.91 --interface eth0 ] ##
      logPath: /tmp/tke-edge-install.log ## The log location of this installation
      success-message: Install Edge Node: node-192.168.67.91 Success! ## Message af

 Start Install Edge Node node-192.168.67.91, Please Waiting...
 Waiting Running of the base service
 Dockerd kubelet lite-apiserver has Running! ## Dockerd kubelet lite-apiserver succ
   Install Edge Node: node-192.168.67.91 Success! ## Indicates that the edge node i
```

After the edge node installation is completed, you can execute the command kubectl -n kube-system get pod to check whether all Pods are in the Running state.
The edgectl install command is valid for one hour and can be used to add different nodes multiple times within the valid period. If it expires, please perform the Script Add Nodes operation again to request a new edgectl.
If there is no prompt to exit during the installation process, please refer to the installation log location logPath: /tmp/tke-edge-install.log provided in the NOTE for specific errors.
If there is no "Install Edge Node: node-192.168.67.91 Success!" after execution, the installation is considered failed. Please check the installation log for troubleshooting.