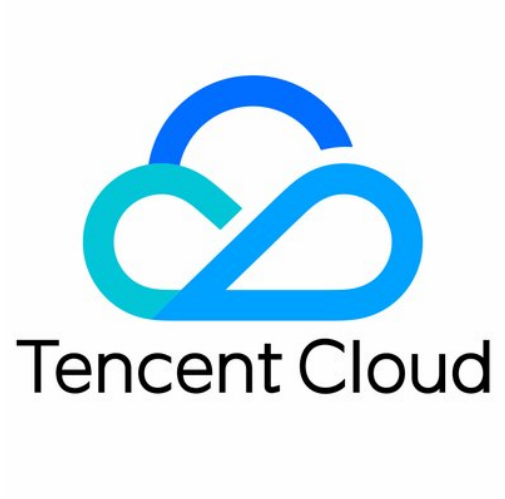
Tencent Cloud Mesh

Operation Guide

Product Documentation



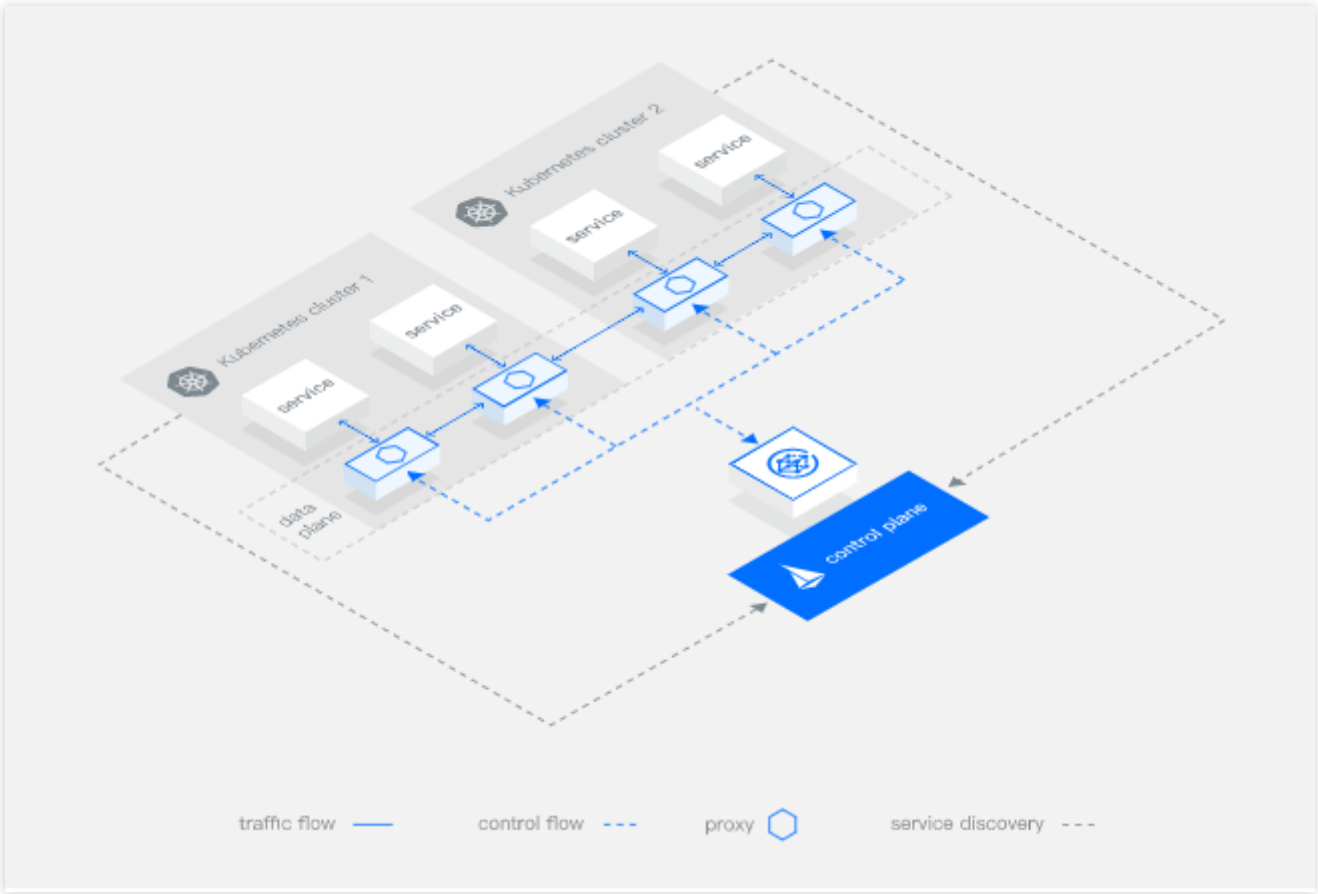
Contents

Operation Guide
- Mesh Instance Management
  - Overview
  - Creating a Mesh
  - Upgrading a Mesh
  - Updating Mesh Configurations
  - Sidecar Injection and Configuration
  - Deleting a Mesh
- Service Discovery Management
  - Overview
  - Automatic Service Discovery
  - Manual Service Registration
- Gateway
  - Gateway Management
  - Gateway Configuration
- Traffic Management
  - Overview
  - Using VirtualService to Configure Routing Rules
  - Using DestinationRule to Configure Service Versions and Traffic Policies
- Observability
  - Overview
  - Monitoring Metrics
  - Call Traces
  - Access Logs
- Security
  - Authentication Policy Configuration
  - Authorization Policy Configuration
- Access Management
  - Overview
  - CAM Service Role Authorization
  - CAM Preset Policy Authorization
  - CAM Custom Policy Authorization
- Extended Features
  - Using a Wasm Filter o Extend the Data Plane

Operation Guide
Mesh Instance Management
Overview

Last updated：2023-12-26 11:39:03

A mesh instance is a logically isolated space for managing services, and services within the same mesh can communicate with each other.



Lifecycle statuses of a mesh are described as follows:

Status	Description
Creating	The mesh is being created, and its details cannot be viewed.
Running	The mesh is running normally.
Upgrading	The mesh is being upgraded, and some features are unavailable.
Idle	When all service discovery clusters managed by the mesh are deleted or disassociated, the mesh will enter an idle state. The mesh in the idle state can be viewed normally, but some

	features are unavailable due to no service entity. You can add a new service discovery cluster for the mesh to restore it to a normal state.
Invalid	When the mesh remains idle for more than 30 days, or the primary cluster of a stand-alone mesh is deleted, the mesh will enter an invalid state and you will no longer be able to perform operations on the mesh other than deletion.
Abnormal	Some components in the mesh are abnormal, which have adverse impact on the mesh features.

The following configurations are required during mesh creation:

Adding a service discovery cluster

This can be implemented by adding a Kubernetes service discovery cluster to automatically discover a service in the cluster or by manually registering a service. The discovered service in the mesh will be displayed in the list on **Mesh details** > **Service** on the Tencent Cloud Mesh console. After the service is discovered, it can be accessed by other services in the mesh. For detailed instructions, see Service Discovery Management.

Creating a gateway

Gateways are divided into two types: ingress and egress, which are the entrance and exit of mesh traffic. Ingress gateways must be created to ensure that traffic can enter the mesh. Egress gateways are optional. For detailed instructions, see Gateway Management.

Injecting sidecars for a service

Sidecar containers are responsible for mesh governance such as data plane traffic management, rule validation, monitoring and reporting. They are the basis for mesh traffic governance and observation. Therefore, for services that require traffic management and observation, sidecars need to be injected into them. For detailed instructions, see Mesh Configuration.

Configuring an observability backend service

Observability includes three parts: monitoring metric viewing, call tracing, and log management. Tencent Cloud Mesh supports integration with Managed Service for Prometheus (TMP), Application Performance Management (APM), and Cloud Log Service (CLS) to provide richer and integrated observability capabilities. In addition, Tencent Cloud Mesh also supports interworking with third-party Prometheus, Jaeger/Zpkin services to provide you with greater component scalability. For detailed instructions, see Observability.

After the mesh is created, you can schedule traffic rules of the mesh, or create traffic governance rules for the mesh through the console or by submitting a YAML file. Currently, Tencent Cloud Mesh is fully compatible with Istio's native syntax. For detailed instructions, see Traffic Management.

Creating a Mesh

Last updated：2023-12-26 11:42:51

Overview

Create a service mesh instance before using the service mesh. Mesh instances have regional attributes, but can manage services in multiple regions.

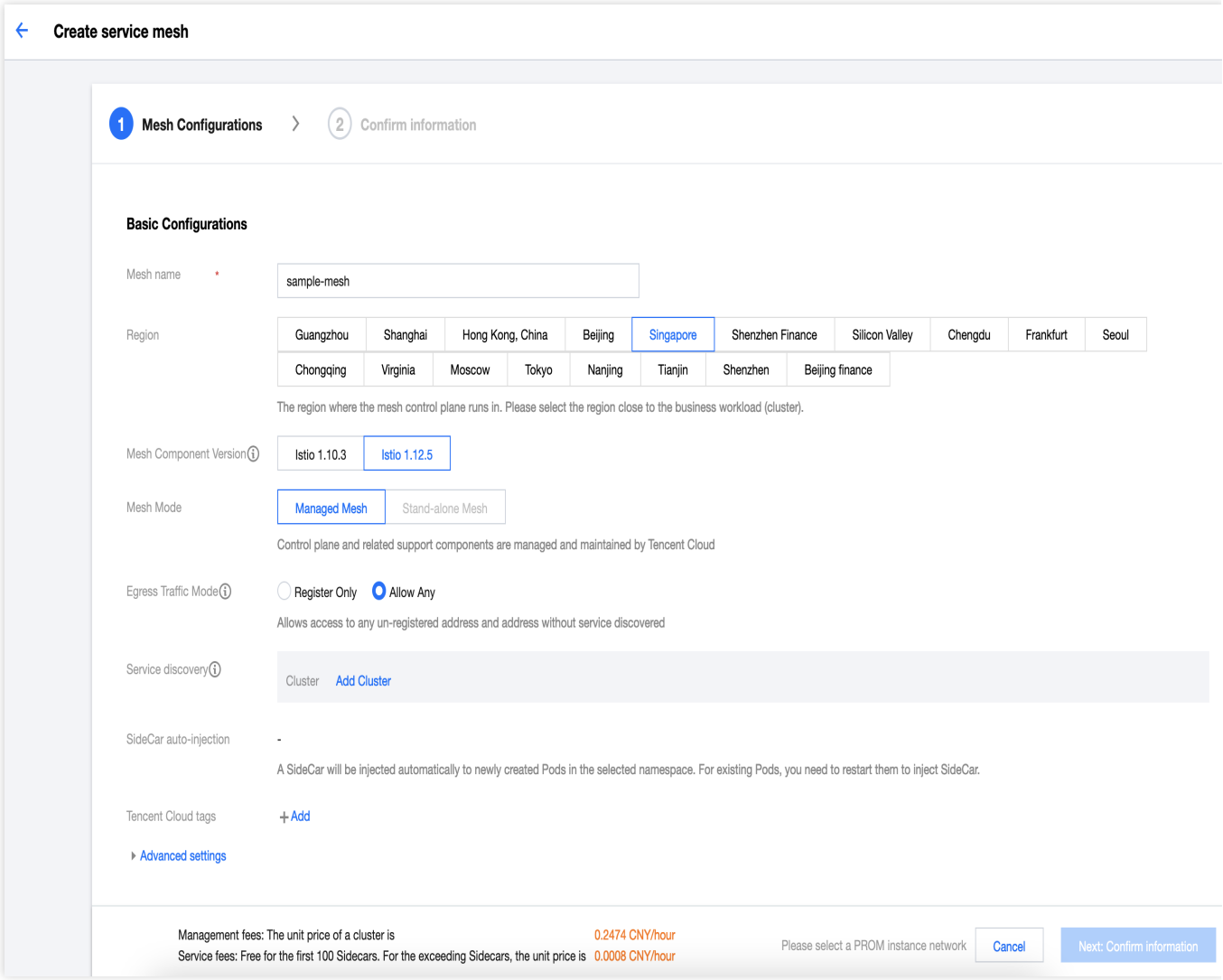
Note:

Each account is allowed to create 20 meshes by default. If more meshes are required, submit a ticket.

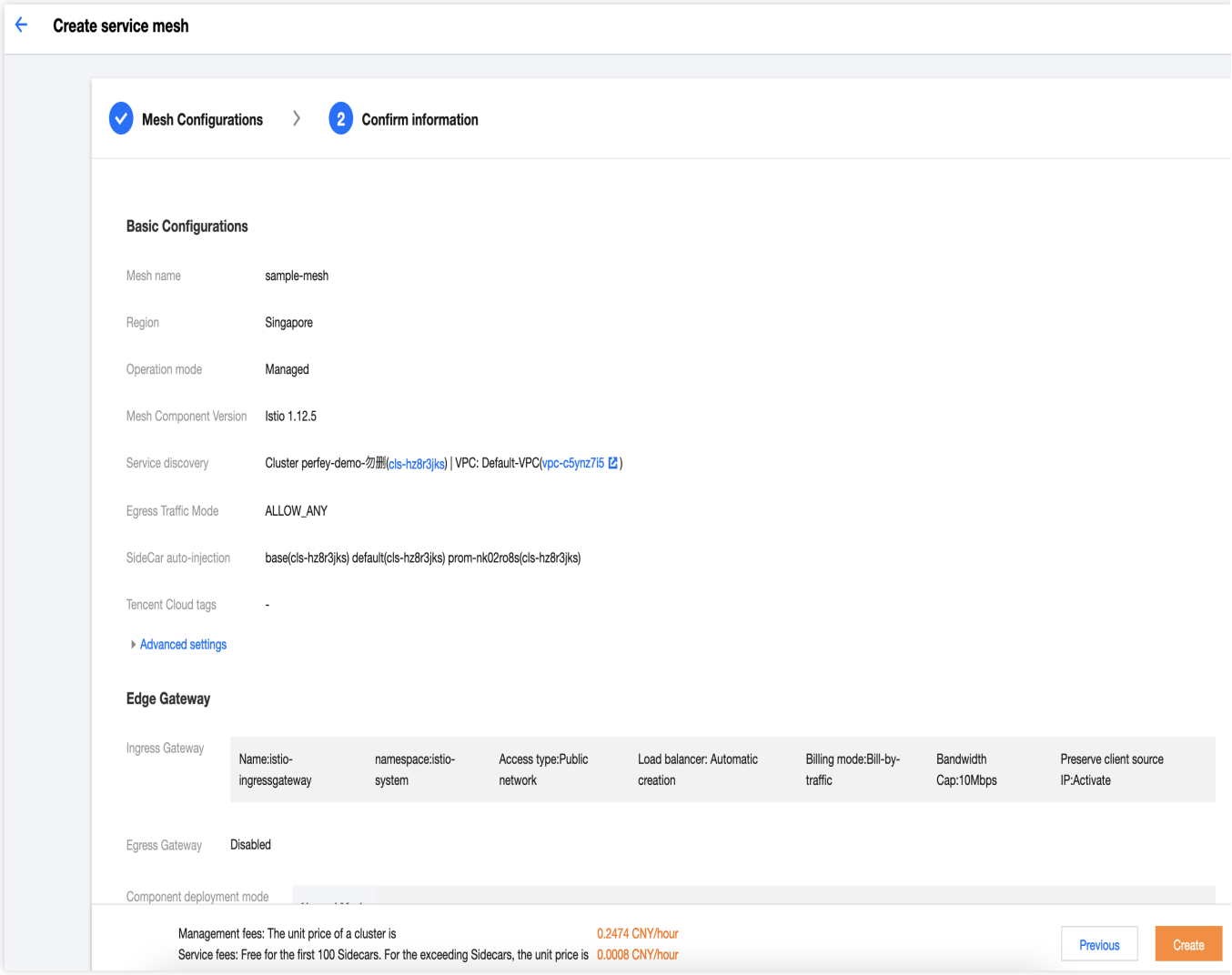
Directions

The procedure of creating a service mesh instance on the console is as follows:

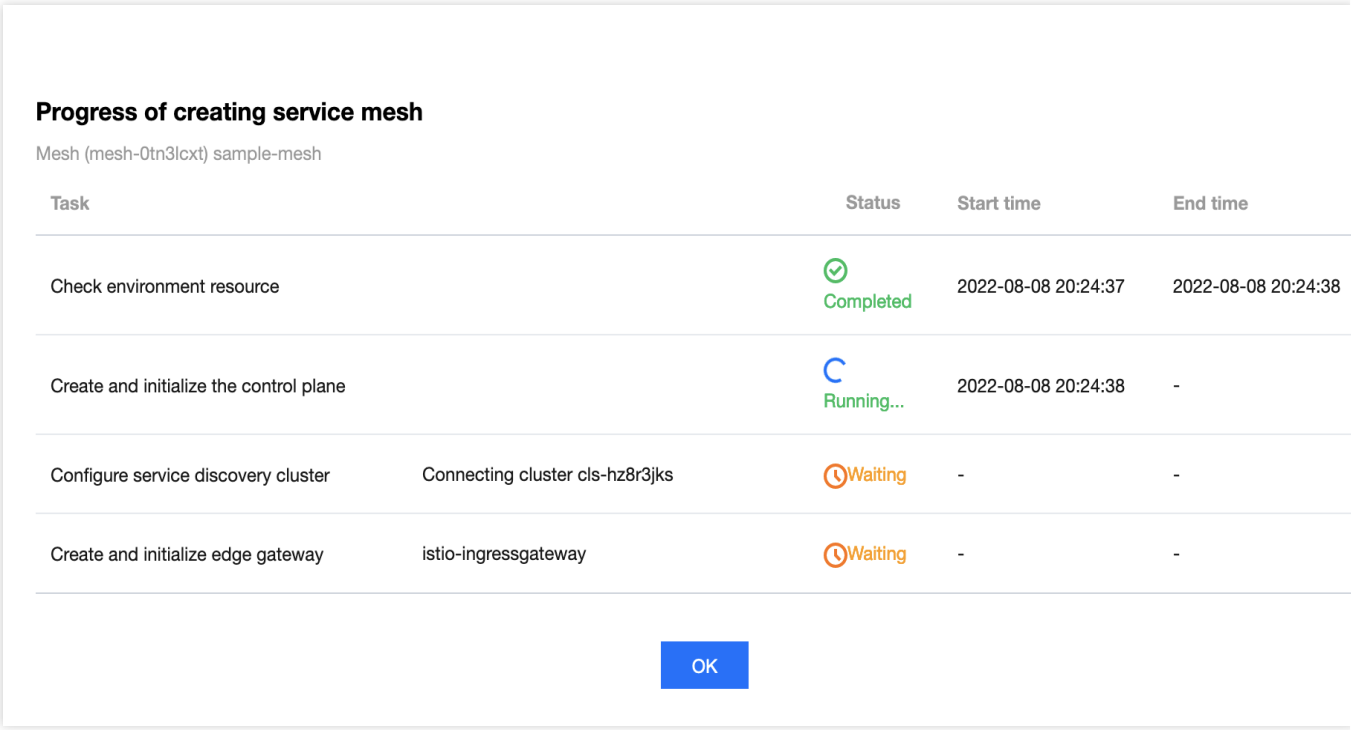
1. Log in to the Tencent Cloud Mesh console.
2. Select a region, and click **Create** in the upper left corner of the page.
3. On the **Create service mesh** page, fill in configurations related to mesh creation as required. For the description of the configuration items, see Configuration Item Description for Mesh Creation. Then, click **Next: Confirm information**.



4. On the **Confirm information** page, confirm that the creation configurations are correct and click **Submit** to start the mesh creation process.



5. After the mesh creation process is complete, view the service mesh instance in the list.



Configuration Item Description for Mesh Creation

Configuration Item	Description	Required
Mesh name	Name of the service mesh to be created.	Yes
Region	Region where the service mesh control plane runs. The region where the control plane runs can be different from the region where the service workload (such as a cluster) is located. It is recommended to select a region close to the region where the service workload (cluster) is located.	Yes
Mesh component version	Control plane and data plane version. Tencent Cloud Mesh is compatible with the latest two major versions of the Istio community.	Yes
Mesh mode	Deployment mode of components related to the service mesh control plane. For a managed mesh, the control plane components are managed and maintained by Tencent Cloud. For a stand-alone mesh, the control plane components are deployed in a cluster you specified, and you need to manage and maintain the control plane components in the cluster. The **Managed mesh** option is available by default. A stand-alone mesh can be used after being added to an allowlist. To apply for a stand-alone mesh, submit a ticket.	Yes

Egress traffic mode	Policy for the external access to services in the mesh. Two options are available: **Registry Only** (access to only services automatically discovered by the mesh and manually registered services is allowed) and **Allow Any**(access to any address is allowed).	Yes
Service discovery	Cluster for implementing automatic service discovery. The cluster must meet constraints such as version, permission, and IP range conflict.	No
Sidecar auto-injection	Namespace into which sidecars are automatically injected. After this field is enabled, sidecars will be automatically injected into all service workloads in the selected namespace. Auto-injection will take effect only for newly created service workloads. Sidecars will be injected into existing service workloads only after the workloads are restarted. If you need to further customize sidecar injection exceptions, see Custom Sidecar Injection.	No
External request bypasses sidecar	Corresponding to excludeIPRanges. By default, sidecars takes over all the traffic in the current pod. If you want the access from a specific IP address not to pass through the sidecar proxy, you can configure this field. After configuration, Istio features such as traffic management and observability will not be performed on the request traffic from the IP range. After the configurations are modified, they take effect only for newly added pods, and for existing pods only after the pods are restarted.	No
Sidecar readiness guarantee	Use the HoldApplicationUntilProxyStarts feature to configure a service container to wait for sidecars to complete the startup before starting. This configuration ensures that a pod in the service container that depends on the sidecars can run normally.	No
Sidecar stop protection	After this field is enabled, a sidecar needs to wait for the process in the service container to be completely terminated before stopping, which increases the pod stop time. It is recommended to enable this field for the service whose service process cannot be shut down at any time. For Istio versions earlier than 1.12, Tencent Cloud Mesh uses the preset container prestop script to check that there is no more service process before allowing the service container to exit. If a user configures other prestop scripts, this feature will be interfered with. For versions later than 1.12, this feature is implemented by the new feature EXIT_ON_ZERO_ACTIVE_CONNECTIONS.	No
Custom sidecar resources	By default, Tencent Cloud Mesh configures a resource limit of up to 2 cores and 1 GB for a sidecar container, which are sufficient in most cases. When the scale of your mesh increases or the logic in the sidecar increases, the default resource limit may be insufficient. You can modify the resource limit based on your service requirements.	No

Ingress gateway	Ingress gateway to be created for the mesh. If the selected cluster is a TKE/TKE Serverless cluster, an ingress gateway of the CLB type is created by default. In this case, CLB-related items need to be configured. If the cluster is a manually registered cluster, only a gateway service of the LoadBalancer type is created because it is not determined whether the cluster supports CLB.	No
Egress gateway	If you need to manage the outgoing traffic of the mesh in a centralized manner, such as unified egress, unified authentication, and rule configurations, you need to create an egress gateway. After this field is enabled, an egress gateway service of the ClusterIP type will be automatically created for you.	No
Gateway deployment mode	Two options are available: **Normal mode** and **Exclusive mode**. For details, see Gateway Deployment Modes.	No
Gateway auto-scale policy	HPA policy for the gateway that is deployed in the specified cluster.	No
Network resource definition	Pod resource limit customized for the ingress/egress gateway.	No
Consumer end	Monitoring metric backend service of the mesh. Currently, interworking with TMP is supported. After configuration, monitoring metrics will be reported to TMP. The Tencent Cloud Mesh console displays metrics based on the TMP data source. You can also view the metrics independently on the TMP console. If a consumer end is not configured for the monitoring metrics, the mesh cannot use monitoring features such as displaying monitoring metrics and topologies.	No
Consumer end	Call tracing backend service of the mesh. Currently, interworking with APM is supported. After configuration, tracing data will be reported to APM from sidecars. The Tencent Cloud Mesh console displays tracing data based on the APM data source. You can also view the data independently on the APM console. If a consumer end is not configured for call tracing, the mesh cannot use features such as viewing traces.	No
Trace sampling rate	Sampling rate at which the mesh collects data and persists in conducting call tracing. The resources consumed by sidecars during data collection and reporting are positively related to the bandwidth and data volume. Set the sampling rate as required. It is recommended to set the sampling rate to 100% for development and test environments, and 1% for production environments.	No
Range	To avoid unnecessary overhead, Tencent Cloud Mesh supports enabling sidecar logs for a specific gateway or namespace.	No

Log format	Tencent Cloud Mesh supports logs in JSON or TXT format.	No
Output template	Field settings for sidecar logs. There are two formats of predefined templates: default and enhanced. Compared with the fields output in the default format, the fields output in the enhanced format are added with **Trace ID**. If you need to further modify the field settings, customize the log fields by referring to Envoy's Standard Specifications.	No
Consumer end	Sidecar log backend service. Currently, interworking with CLS is supported. After this field is enabled, a log collection component will be deployed on cluster nodes to ensure normal use of the feature.	No

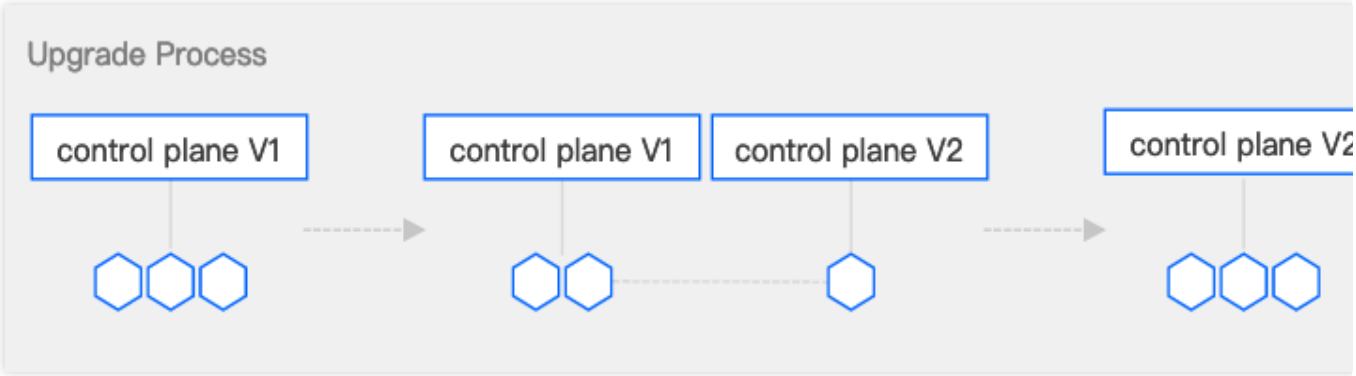
Upgrading a Mesh

Last updated：2023-12-26 11:43:26

Tencent Cloud Mesh provides the mesh upgrade service, which allows you to upgrade a mesh from an earlier version to a later version. The upgrade process follows canary upgrade principles and is divided into the following steps:

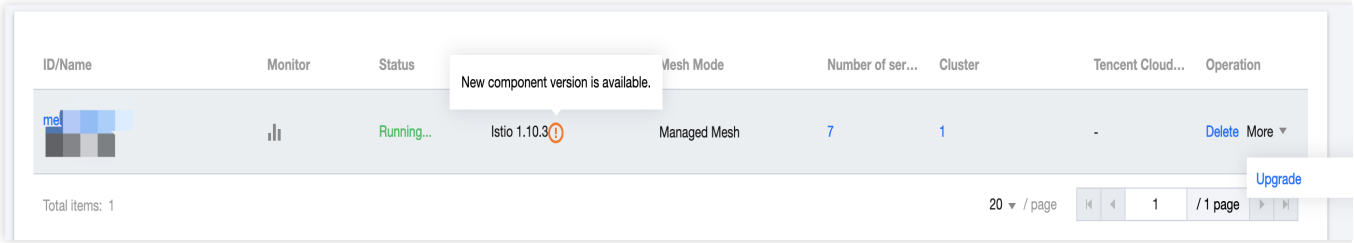
1. Deploy the control plane of a new version to upgrade the control plane of Tencent Cloud Mesh.
2. Conduct a canary upgrade of the data plane, and restart services to update sidecars of existing service pods.
3. Verify the upgrade to check that the services are normal.
4. Take the control plane of the old version offline.

Before the control plane of the old version goes offline, you can roll back the mesh to the state before the upgrade. The upgrade process is shown as follows:



Directions

1. Log in to the Tencent Cloud Mesh console.
2. When the mesh version can be upgraded, there is a prompt indicating that a new version is available.



3. Choose **More** > **Upgrade** and perform the upgrade as prompted.

The upgrade will be performed in three stages: **Control plane upgrade** > **Data plane upgrade** > **Old control**

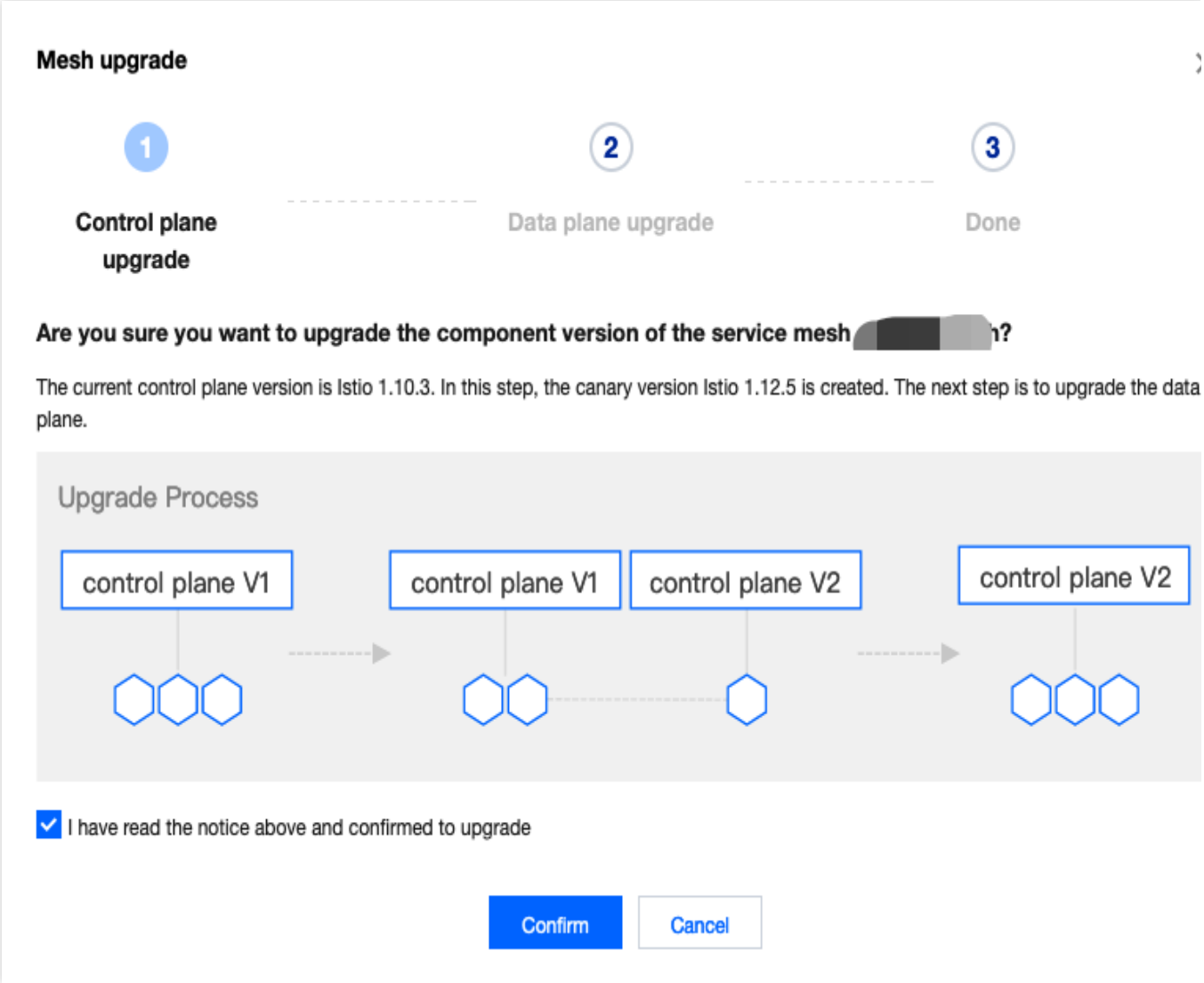
plane offline. Before the control plane of the old version goes offline, you can roll back the mesh to the state before the upgrade.

Control Plane Upgrade

Data Plane Upgrade

Upgrade Verification

During the **Control plane upgrade** stage, Tencent Cloud Mesh deploys control plane components of the new version.

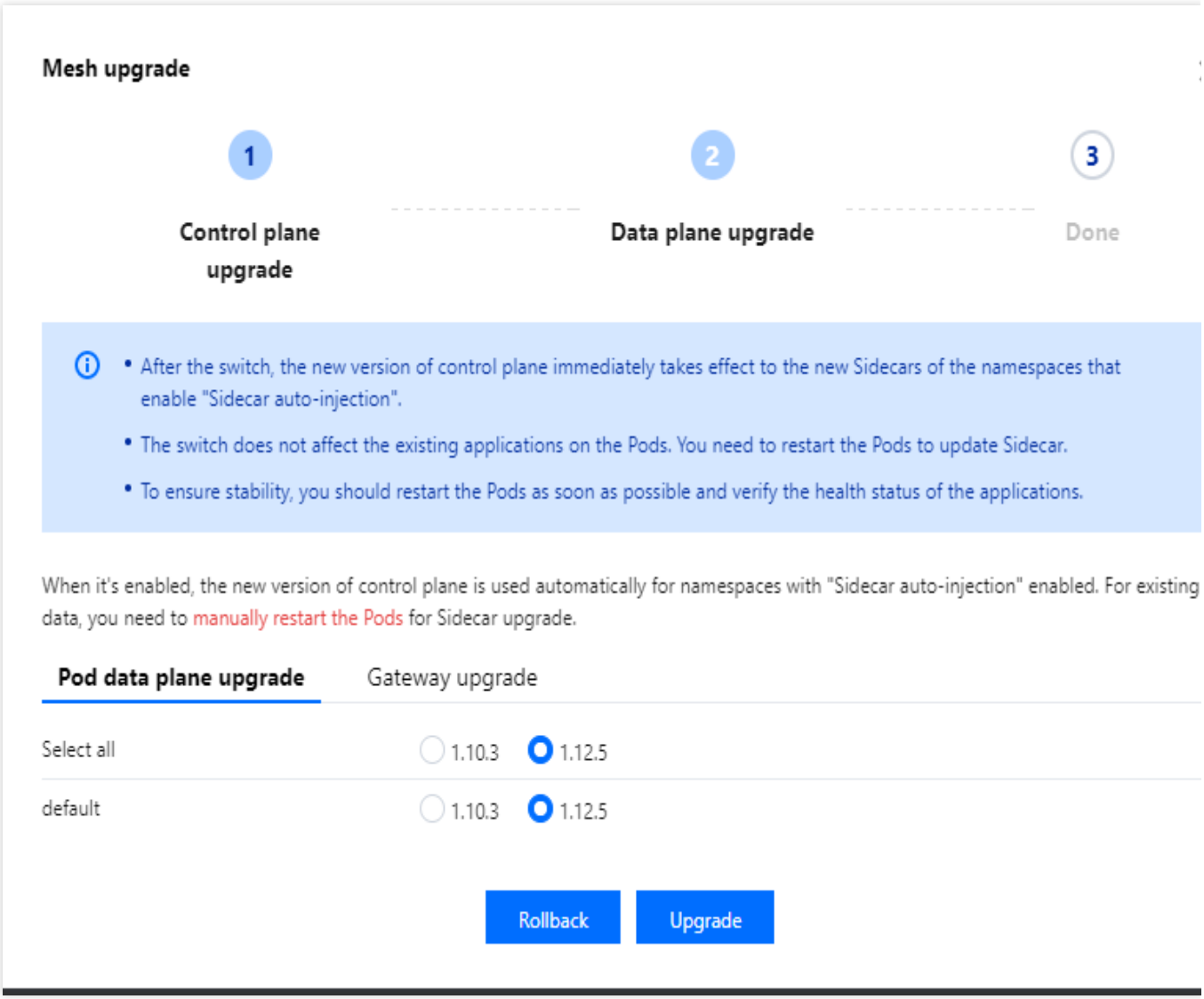


Data plane upgrade consists of service data plane upgrade and gateway upgrade.

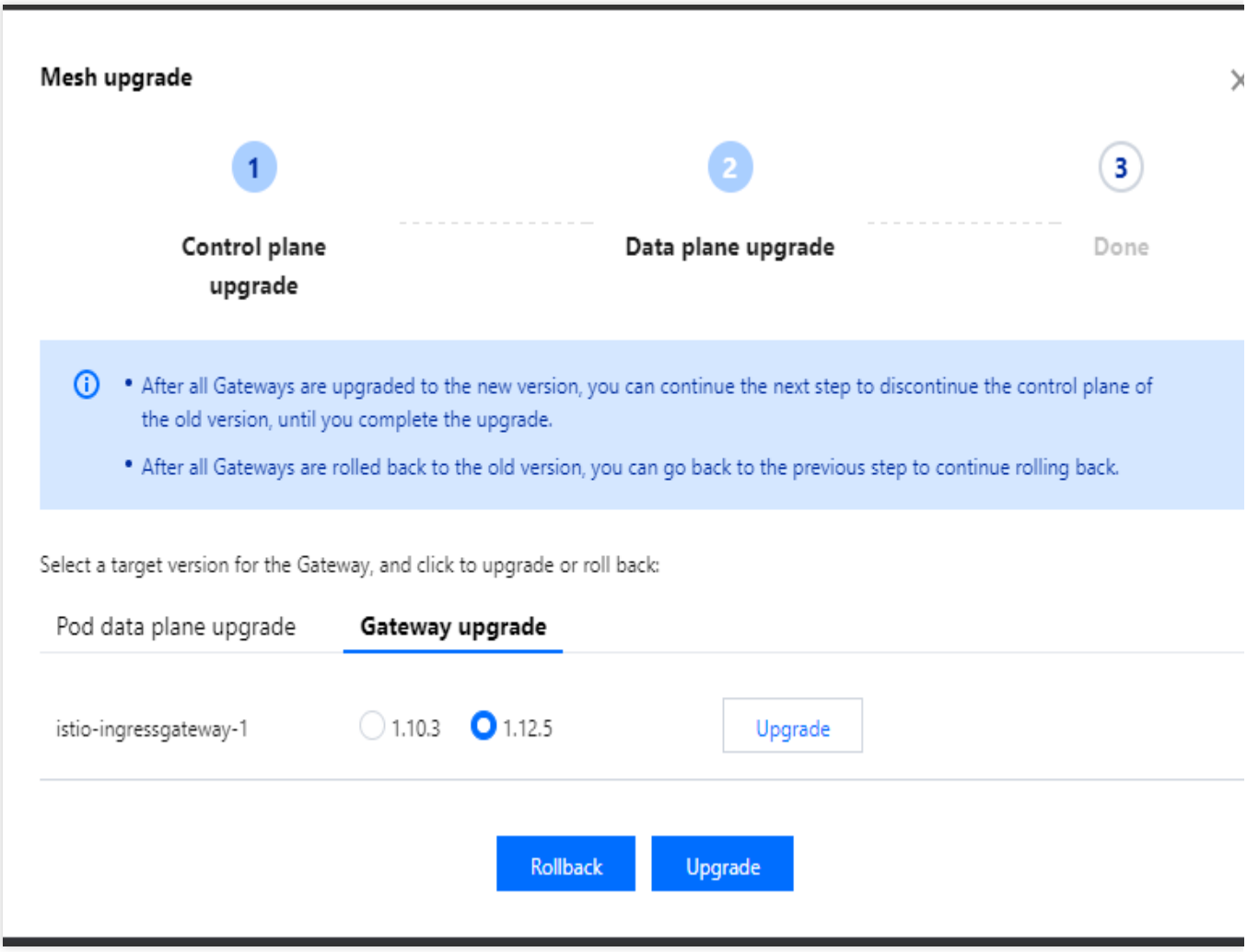
For service data plane upgrade, you need to specify the new version for sidecar auto-injection of the specified namespace. After the new version is selected, sidecars of the new version will be injected into **newly created** service pods under the namespace. **Sidecars in the existing service pods will be updated to the new version only after these pods are rebuilt.** Restart may affect service availability. Therefore, Tencent Cloud Mesh does not automatically rebuild service pods. **Instead, you need to manually rebuild service pods.**

Note:

You can republish a service through a pipeline or manually rebuild a workload by directly using command lines such as kubectl patch and kubectl rollout restart.
In some scenarios, sidecars will be uninstalled instead of being upgraded. For example, assume that a namespace has enabled sidecar injection, sidecars have been successfully injected into some service pods, and then namepsace-level sidecar injection is disabled. After a service pod is restarted, its sidecars will be uninstalled unless a sidecar injection label has been independently set for the pod.



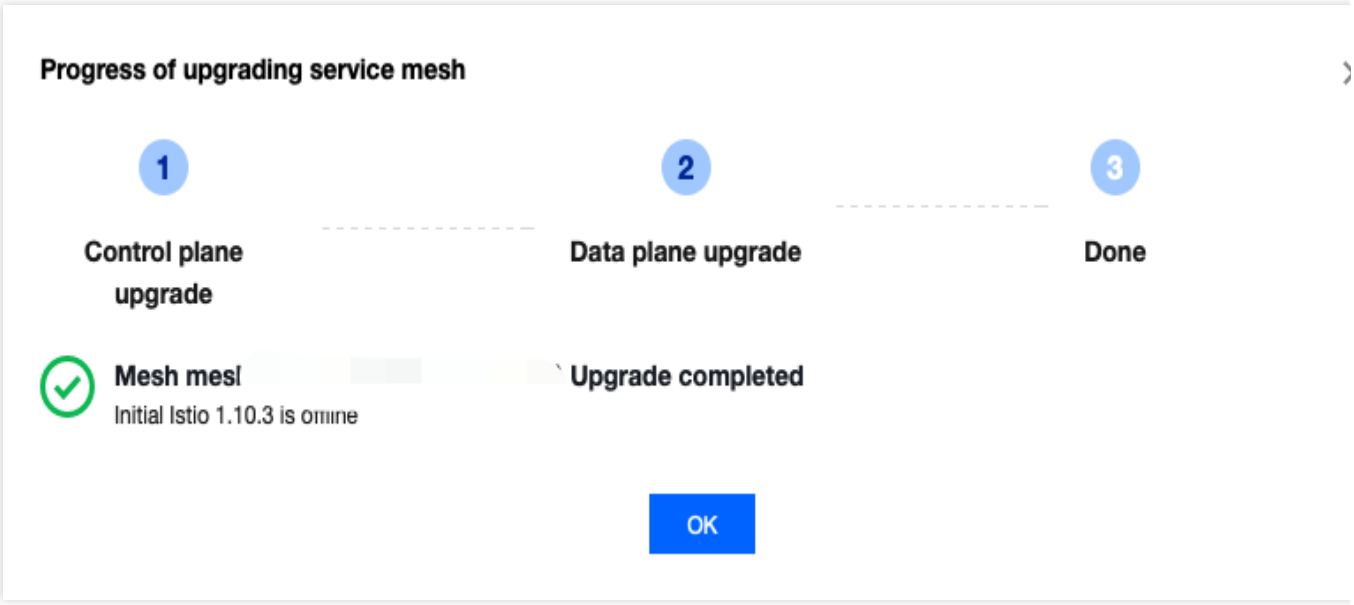
For gateway upgrade, select the new version for all the gateway components that need to be upgraded and click **Upgrade** on the right.



After the data plane upgrade is completed, click **Upgrade** to go to the next step.

Because the version upgrade involves feature changes, there may be compatibility issues. After the service pods are rebuilt, you need to check whether the service is normal. If you find that the upgrade causes service exceptions, you can click **Rollback** on the upgrade page to roll back the data plane sidecars to the source version.

4. Click **Done** or **Cancel upgrade**. During the **Data plane upgrade** stage, you can click **Upgrade** or **Rollback** to check whether the existing pods meet the conditions for entering the next step. When all namespaces are switched to the control plane of the new version, and the sidecars in all the existing service pods have been updated to the new version, you can click **Upgrade** to go to the next step **Old control plane offline** and complete the upgrade. Alternatively, when all namespaces are switched back to the control plane of the old version, and the sidecars in all the existing service pods use the control plane of the old version, you can click **Rollback** to go to the next step to take the control plane of the new version offline and cancel the upgrade.


Progress of upgrading service mesh
1
Control plane upgrade
2
Data plane upgrade
3
Done
Mesh mesl
Initial Istio 1.10.3 is omne
Upgrade completed
OK

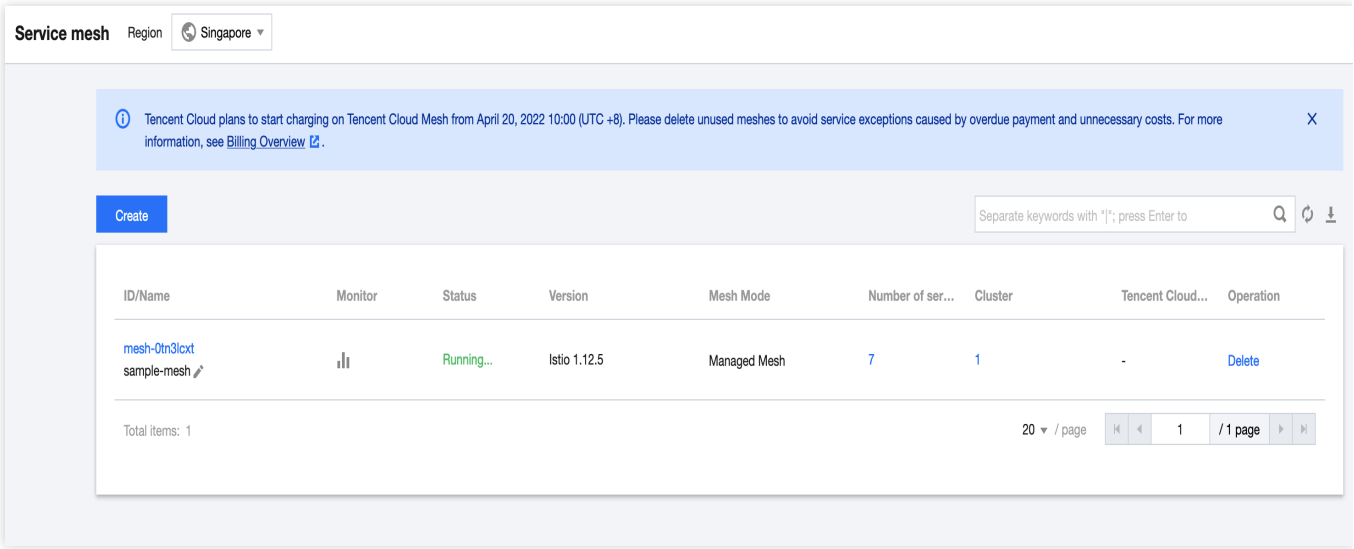
Updating Mesh Configurations

Last updated：2023-12-26 11:44:16

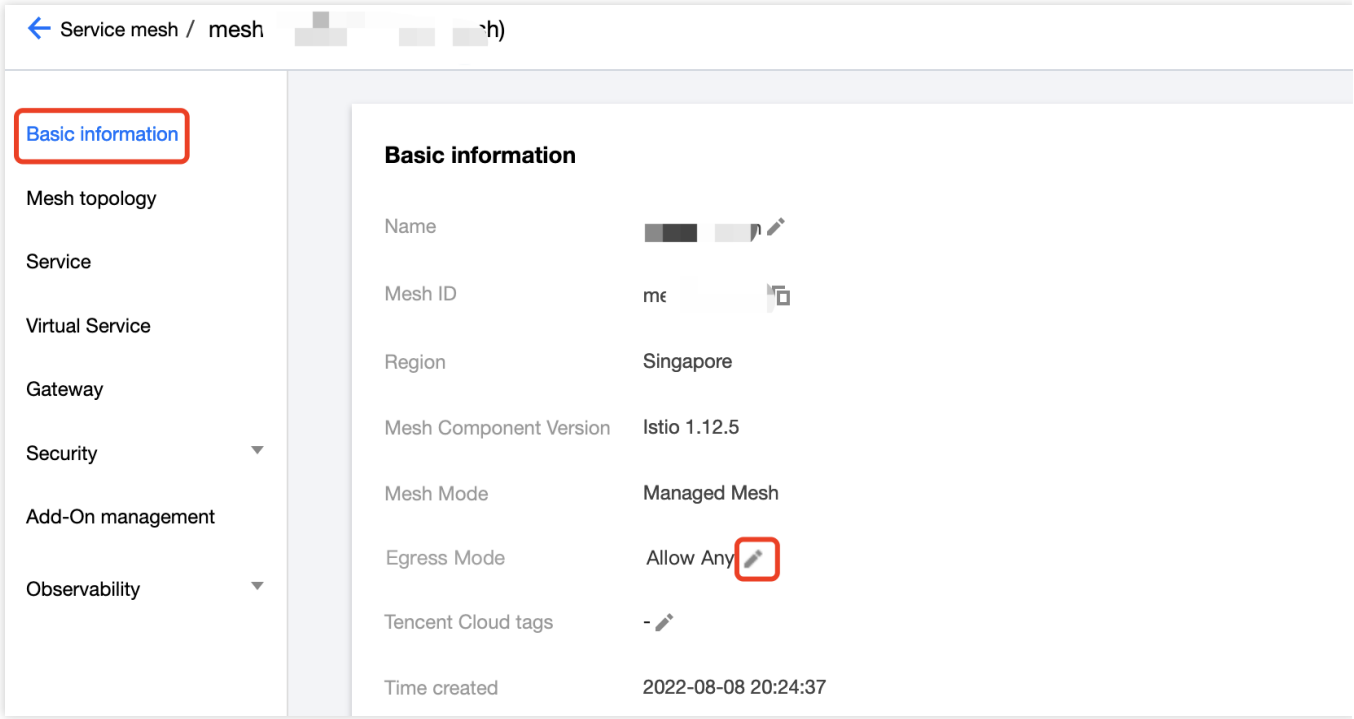
This topic describes how to update the configuration of a running service mesh.

Modifying the Egress Traffic Mode

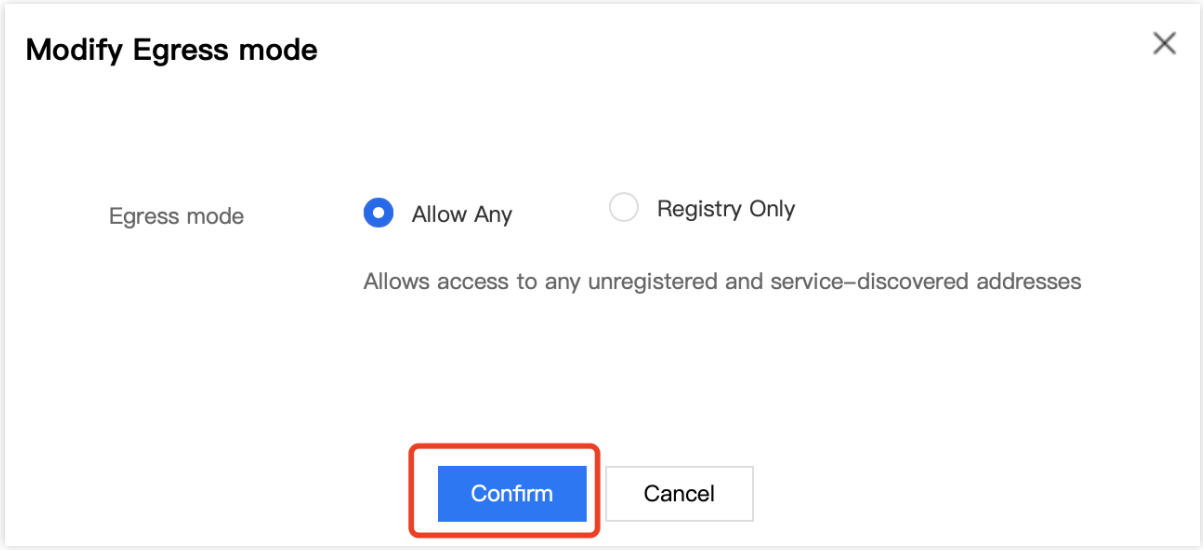
The egress traffic mode defines a policy for the external access to services in the mesh. Two options are available: **Registry Only** (access to only services automatically discovered by the mesh and manually registered services is allowed) and **Allow Any** (access to any address is allowed).
Steps for configuring the egress traffic mode for the mesh are as follows:
1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh management page.



2. On the mesh basic information page, click the **Edit** button of the **Egress traffic mode** field to pop up the **Modify Egress traffic mode** window.

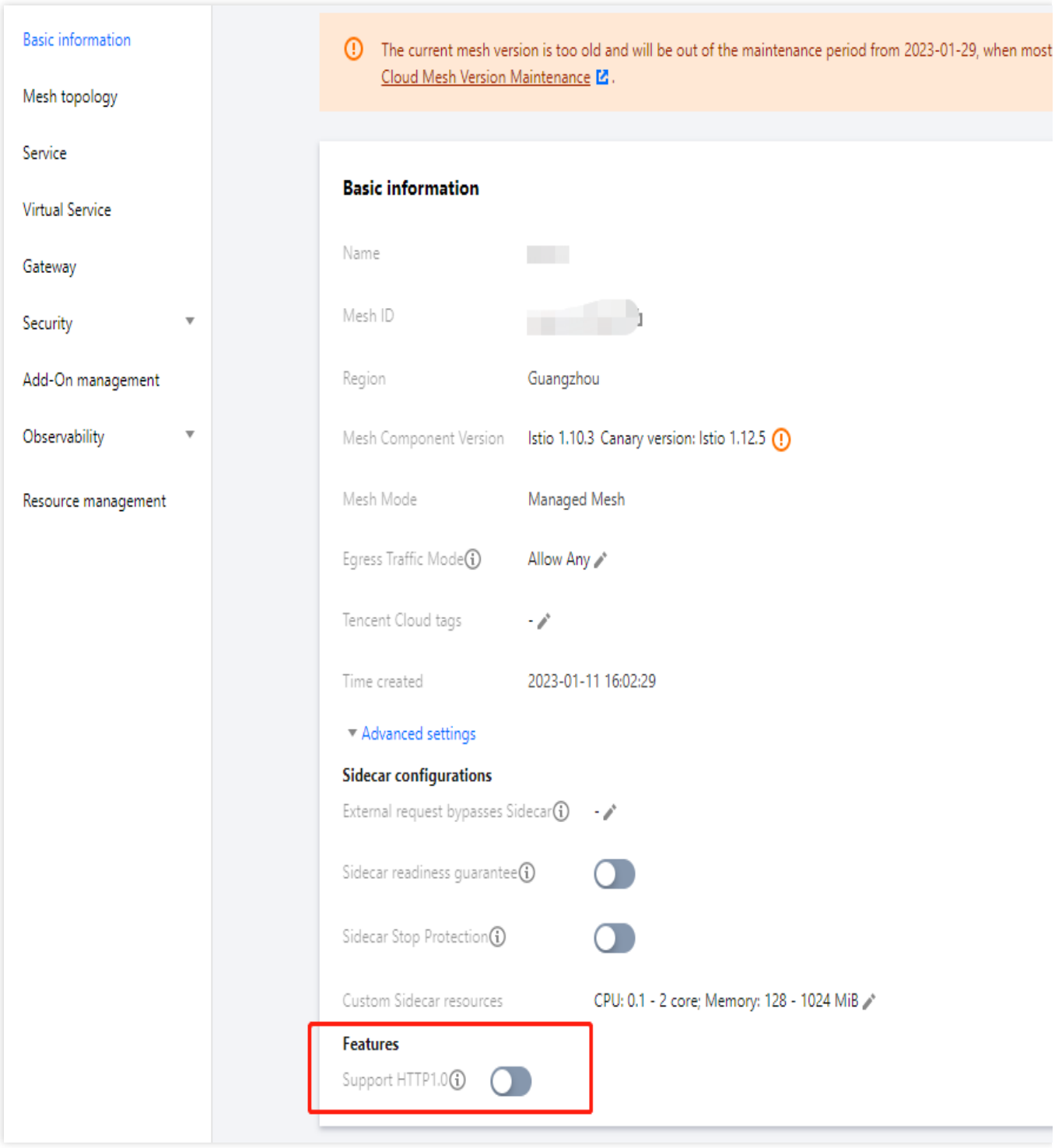


3. Select **Allow Any** or **Registry Only** as required, and click **Confirm** to complete the update of the egress traffic mode.



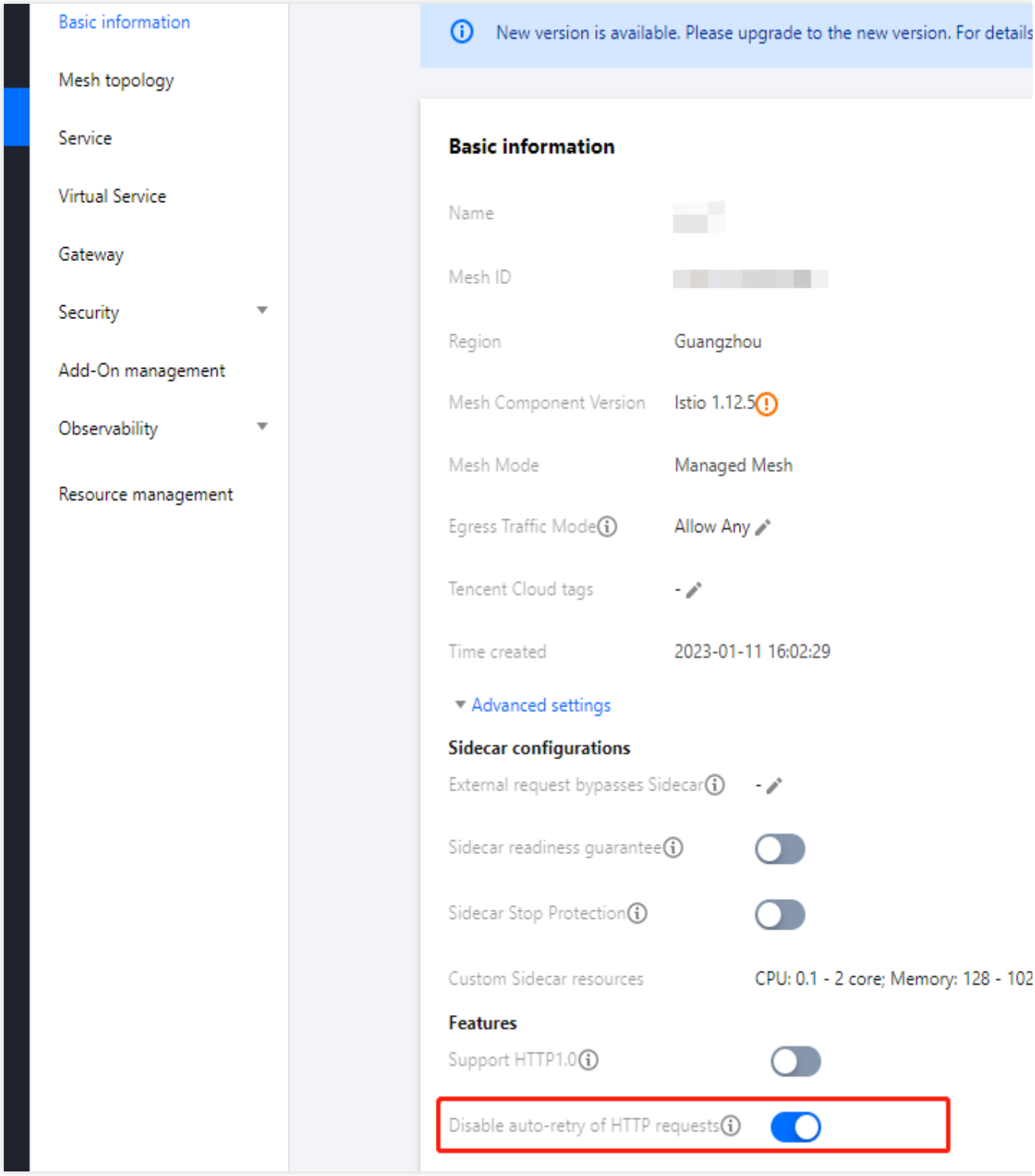
Enabling HTTP 1.0 Support

Istio does not support HTTP 1.0 by default. When necessary, you need to enable HTTP 1.0 support on the mesh basic information page:



Disabling HTTP Auto Retries

Istio automatically retries failed HTTP requests twice by default. If this does not meet your requirements, you can disable auto retries on the mesh basic information page:

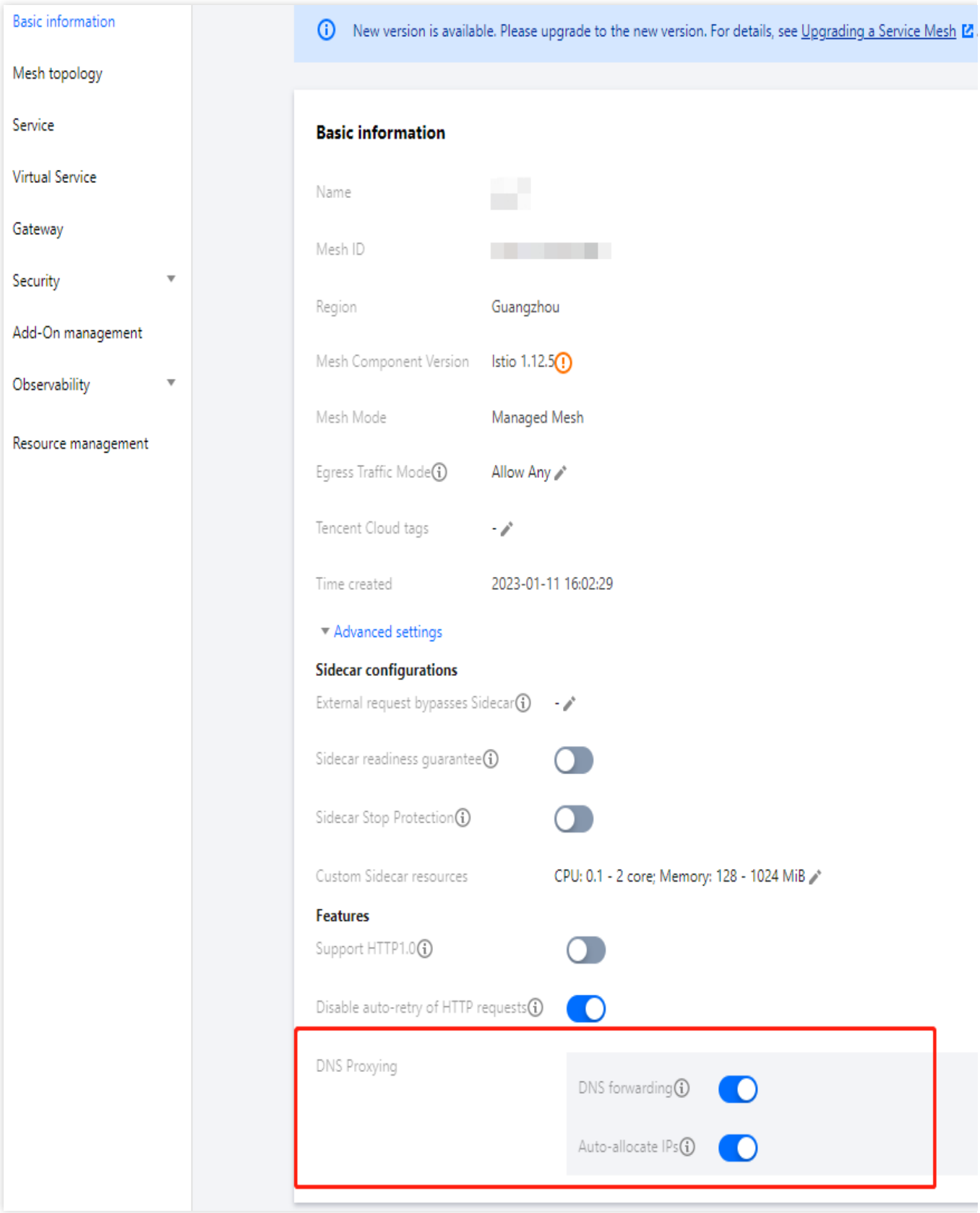


Disabling auto retries applies to the entire mesh. However, you can still set explicit retry policies for specific virtual services.

Enabling DNS Proxy

Istio sidecars support DNS proxy. When DNS proxy is enabled, DNS traffic will also be blocked, and DNS requests will be responded directly by sidecars. On the one hand, sidecars cache DNS resources, which will accelerate DNS responses. On the other hand, in the case of cross-cluster service access in multi-cluster mesh scenarios, the service can still be parsed properly without the need to create a service with the same name in the client cluster. You may follow the steps below to enable DNS forwarding:

1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh management page.
2. On the basic information page, click  on the right of **DNS Proxying** > **DNS forwarding** to enable DNS forwarding. See the figure below:



To automatically allocate IP addresses for ServiceEntrys with no address defined, enable **auto IP allocation**. For more information, see Address auto allocation.

Sidecar Injection and Configuration

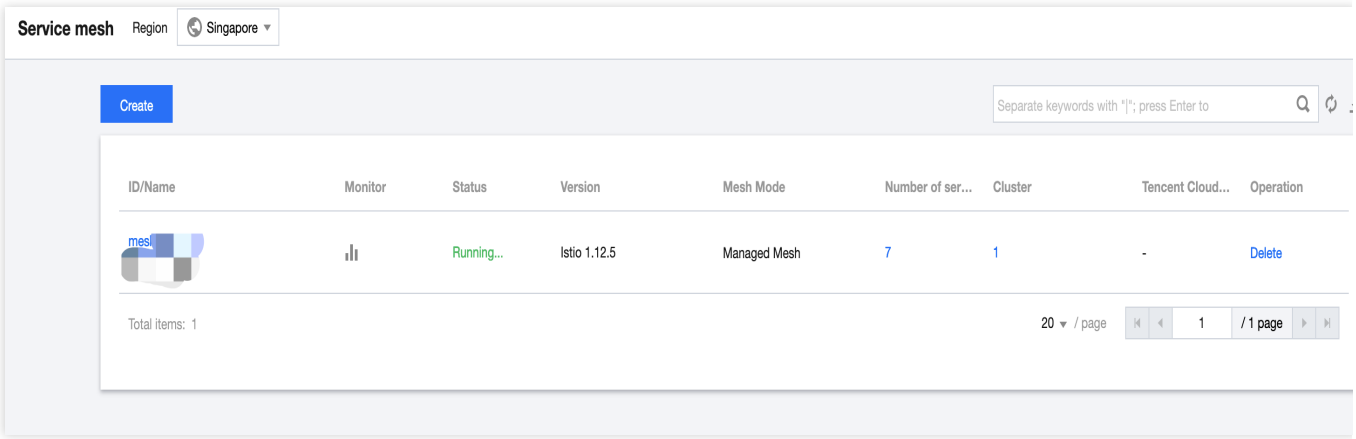
Last updated：2023-12-26 11:45:07

Configuring Sidecar Auto-injection

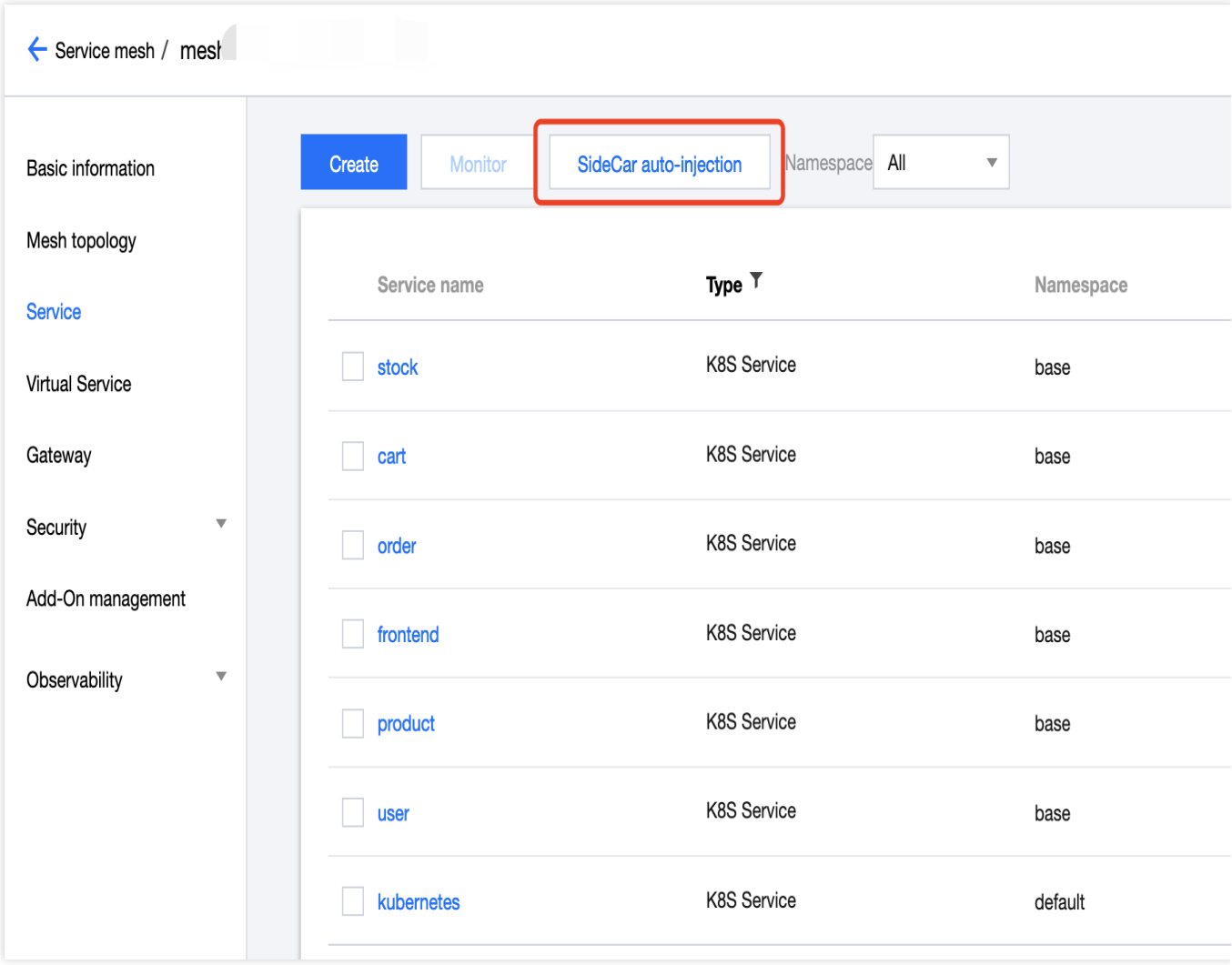
Tencent Cloud Mesh currently supports enabling sidecar auto-injection for a specified namespace on the console. After sidecar auto-injection is enabled, mesh sidecars will be automatically installed on newly created workloads under the namespace. Because injection is completed during the workload creation process, sidecars cannot be automatically installed on existing workloads even if sidecar auto-injection is enabled. You can complete sidecar auto-injection by rebuilding workloads.

Steps for configuring namespace-level sidecar auto-injection are as follows:

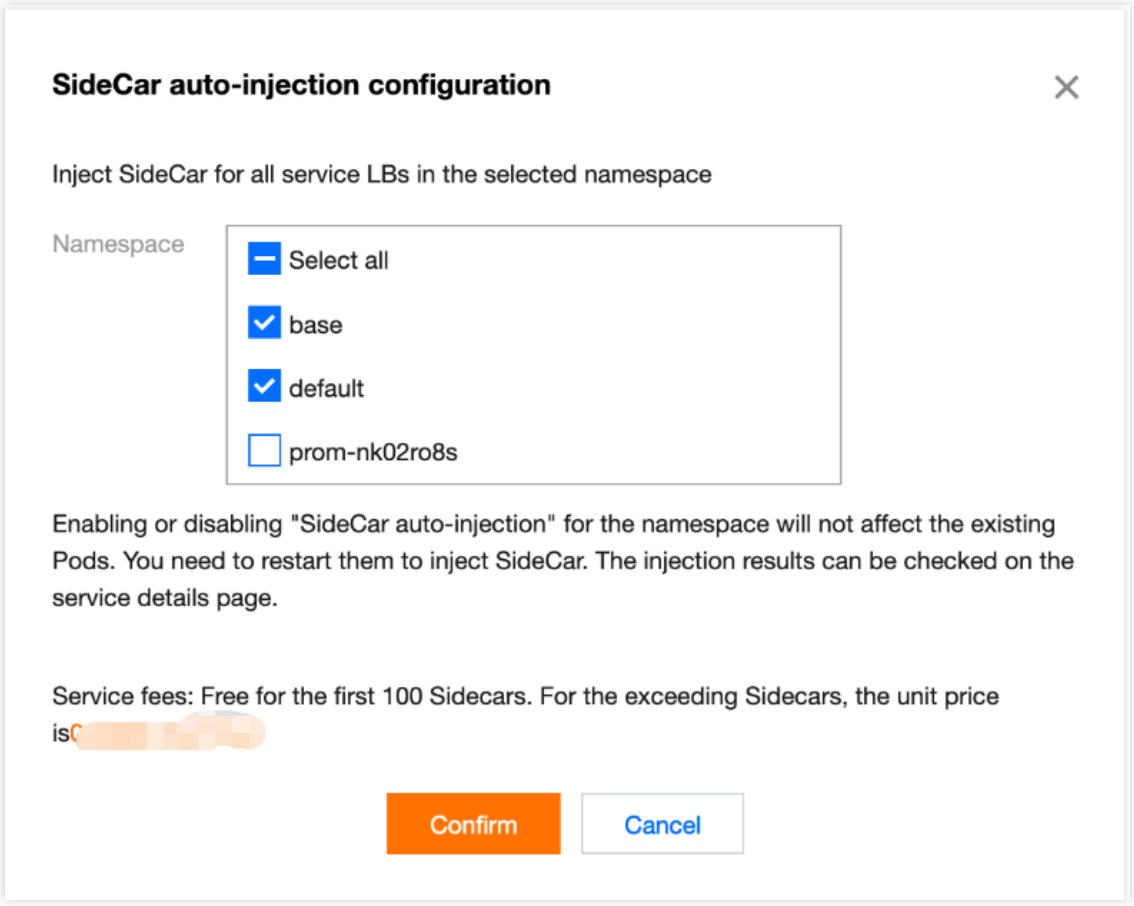
1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh management page.



2. On the service list page, click **Sidecar auto-injection** to pop up the **Sidecar auto-injection configuration** window.



3. Select one or more namespaces for which sidecar auto-injection needs to be enabled, and click **Confirm** to complete sidecar auto-injection configuration.



Customizing Sidecar Injection

Tencent Cloud Mesh also allows you to enable sidecar auto-injection for a specific workload by editing a .yaml file. If necessary, you can add a label `istio.io/rev: {Istio version number}` to a pod. (Note that label settings related to sidecar injection in Tencent Cloud Mesh are slightly different from Istio's default syntax.) An example is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
        istio.io/rev: 1-14-5
    spec:
      containers:
      - name: nginx
        image: nginx:latest
```

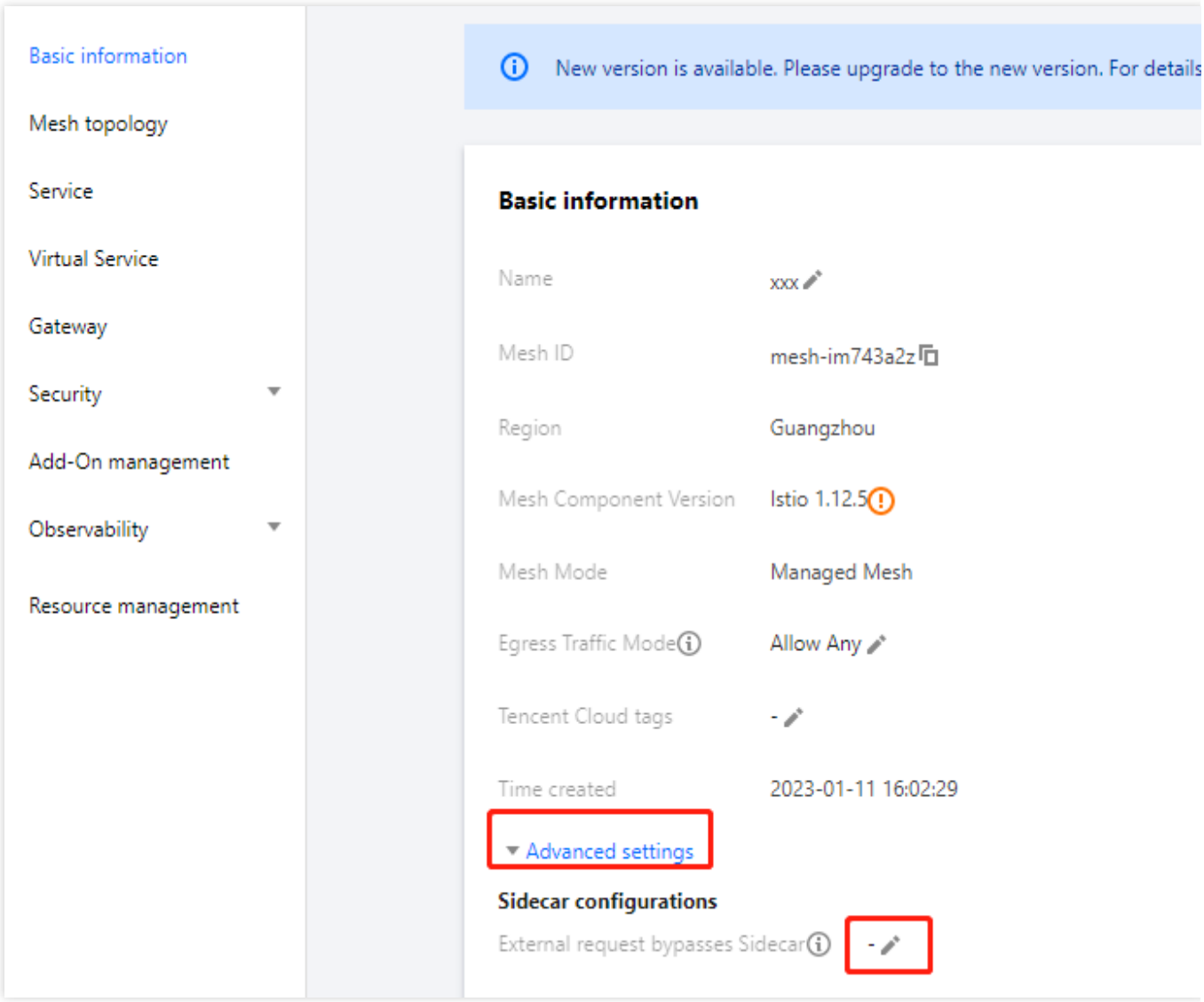
If you need to add a special case for a specific pod under a namespace that has auto-injection enabled to disable sidecar auto-injection, you can add a label `sidecar.istio.io/inject="false"` for the pod. Pod-level injection has a higher priority than namespace-level injection. For more details on sidecar auto-injection, see the Istio documentation Installing the Sidecar.

Allowing Traffic from Specified IP Ranges

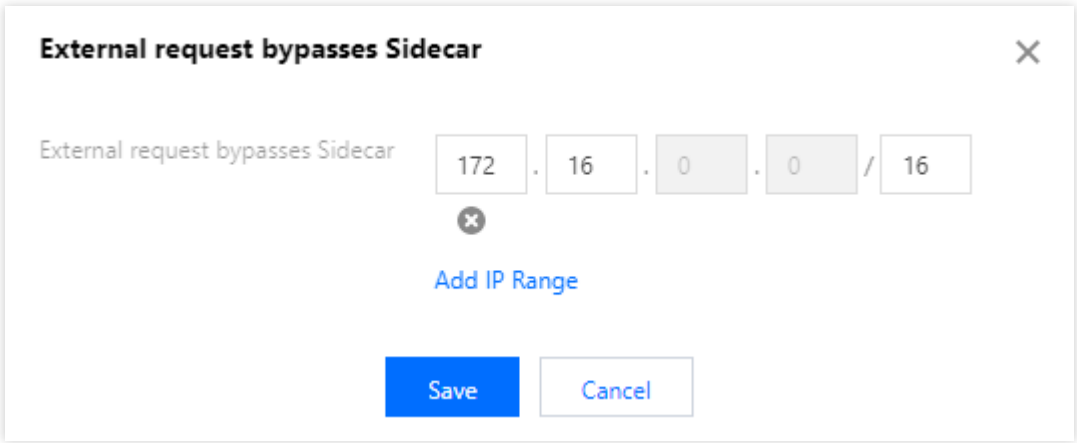
You can configure not to block certain traffic. For example, you may not want to block the traffic of uploading files to Cloud Object Storage (private destination IPs beginning with 169.254). If such traffic is blocked and the downloaded files are large, it may lead to a high memory resource usage of the sidecar. The reason is that the sidecar caches the requested content and the requested content will be reused upon an automatic retry when an error occurs. To allow such traffic, you can go to the **External request bypasses Sidecar** window in the advanced settings area on the mesh basic information page to add the IP ranges that you do not want to block. The steps are as follows:

1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh management page.
2. On the basic information page, click

on the right of **External request bypasses Sidecar**. See the figure below:



3. In the **External request bypasses Sidecar** window that pops up, add the IP ranges that you do not want to block. See the figure below:



4. Click **Save**.

Above is the global configuration method. To make configuration for certain workloads only, add the `traffic.sidecar.istio.io/excludeOutboundIPRanges` annotation to the pod. For more information, see Resource Annotations.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        'traffic.sidecar.istio.io/excludeOutboundIPRanges': '169.254.0.0/16'
      labels:
        app: nginx
```

Controlling the Sidecar Startup Sequence

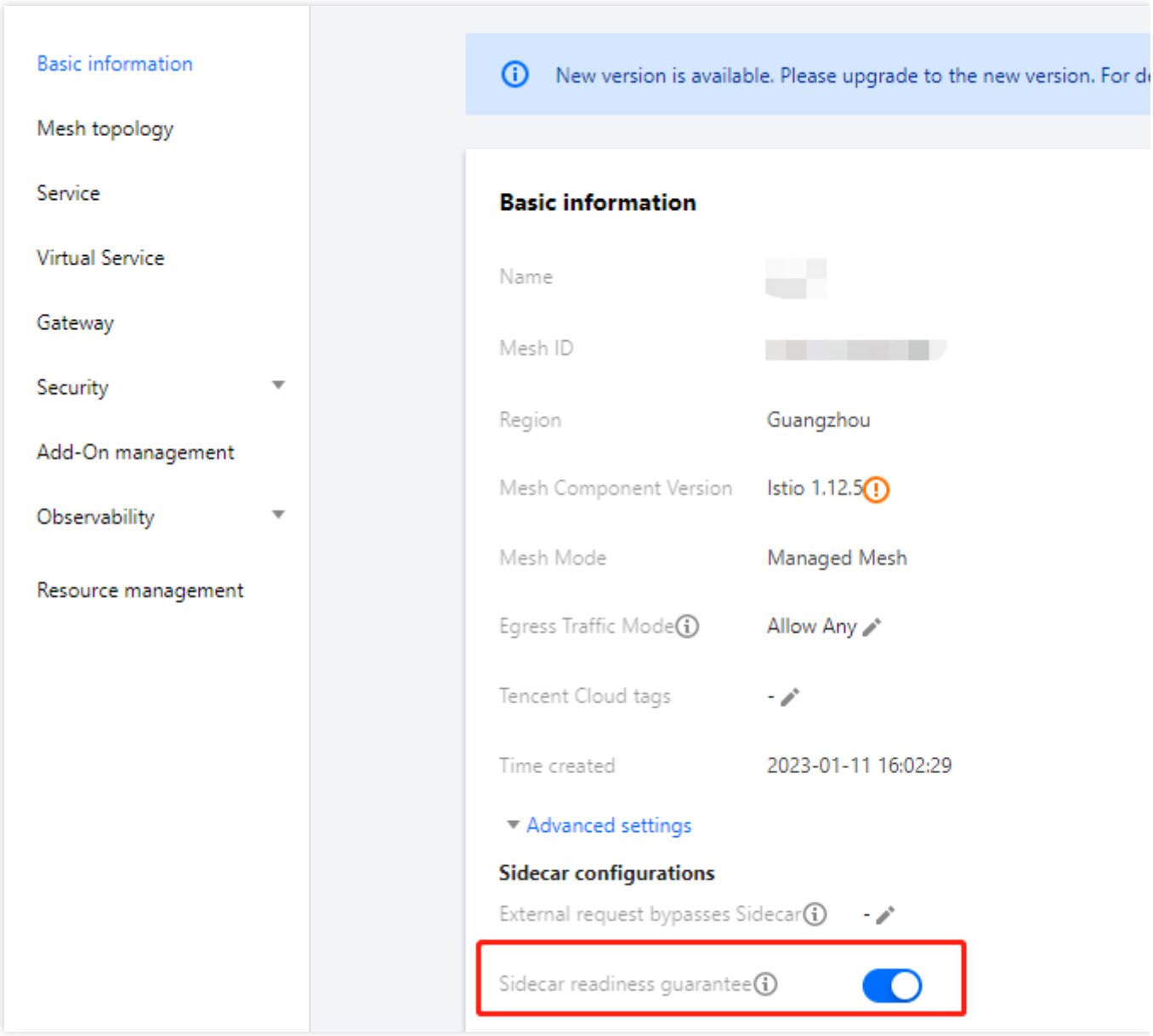
When Kubernetes starts a pod, all containers in the pod will be started simultaneously. In scenarios where Istio is used, because traffic will be blocked by the sidecar, if the sidecar is slower than the service containers in startup, the network requests initiated just after the service containers are started will fail, because the traffic is blocked by the sidecar but the sidecar startup is not completed. A common scenario is that, for a large-scale cluster, the sidecar starts slowly due to the slow pull of XDS when the sidecar is started, and the service process needs to pull configuration from the registry when it starts. The configuration pull fails because the traffic is blocked by the sidecar but the sidecar is not ready to handle the traffic at the time, and then the service process reports an error and exits, and the containers exit as a result.

Two solutions are available: The first is to make the service code more robust by retrying requests that fail during startup until they succeed. The second is to let the sidecar start first, and then start the service containers when the sidecar is ready. You can follow the steps below to enable sidecar readiness guarantee:

1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh management page.

2. On the basic information page, click

on the right of **Sidecar Readiness Guarantee**. See the figure below:



Above is the global configuration method. To make configuration for certain workloads only, add the following annotation to the pod:

```
proxy.istio.io/config: '{ "holdApplicationUntilProxyStarts" : true }'
```

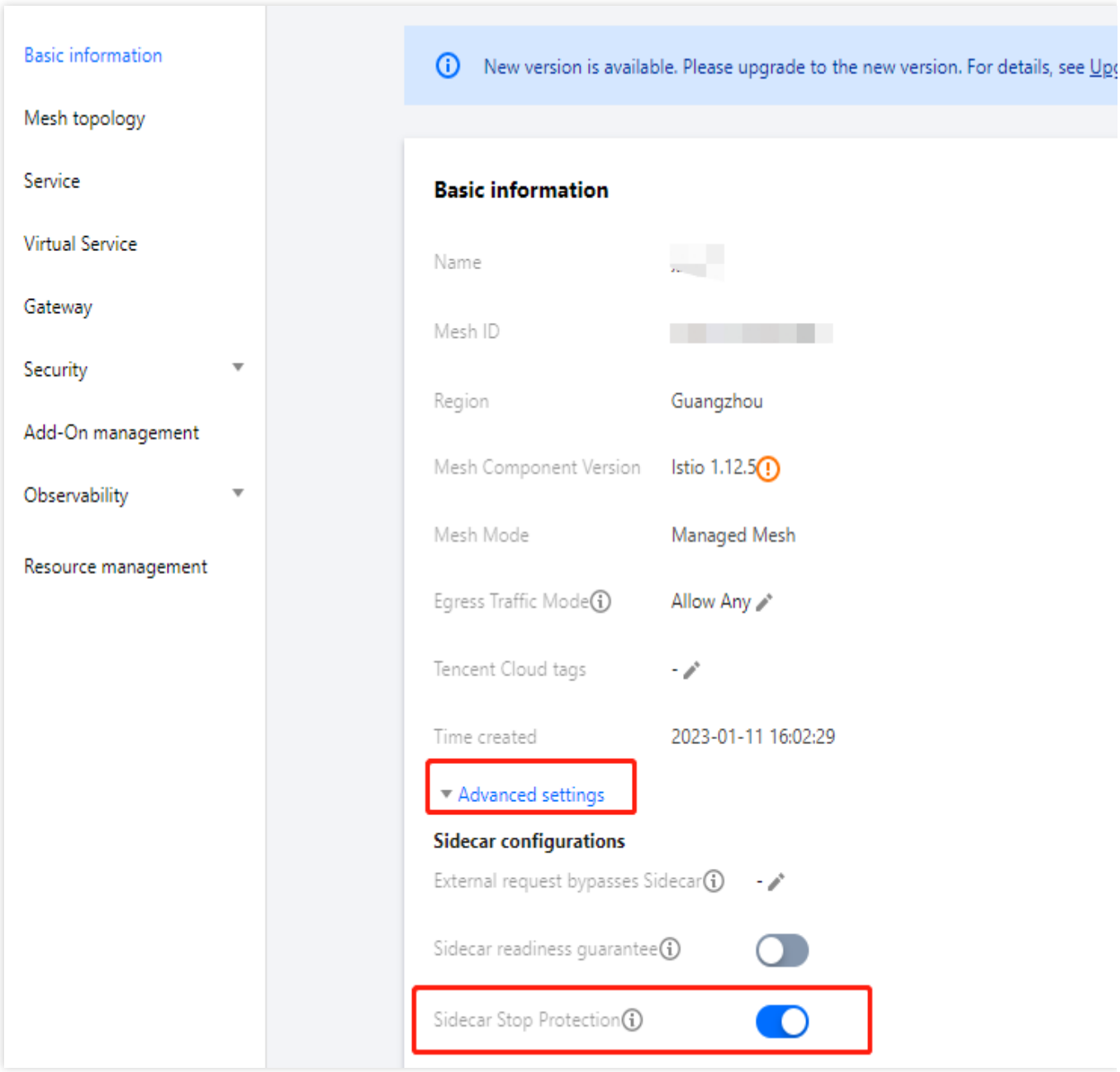
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        proxy.istio.io/config: '{ "holdApplicationUntilProxyStarts" : true }'
      labels:
        app: nginx
```

Graceful Sidecar Termination

When a service is released, the associated workload will be updated on a rolling basis. During the termination of the pod, the sidecar waits only a few seconds by default before being forced to stop. If the service requests themselves take a long time, or if persistent connections are used, some normal service requests and connections may be interrupted. We want the sidecar to wait for the existing service requests and connections to end before exiting for graceful termination. To achieve this, the environment variable `EXIT_ON_ZERO_ACTIVE_CONNECTIONS` is added to sidecars starting from Istio 1.12, and, in responses, the server instructs the client to end persistent connections (adding the `Connection: close` header to HTTP 1 responses and adding the `GOAWAY` frame to HTTP 2 responses). You may follow the steps below to enable sidecar stop protection:

1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh management page.
2. On the basic information page, click

on the right of **Sidecar Stop Protection**. See the figure below:



Above is the global configuration method, which is recommended. To make configuration for certain workloads only, add the following annotation to the pod:

```
proxy.istio.io/config: '{ "proxyMetadata": { "EXIT_ON_ZERO_ACTIVE_CONNECTIONS": "tr
```

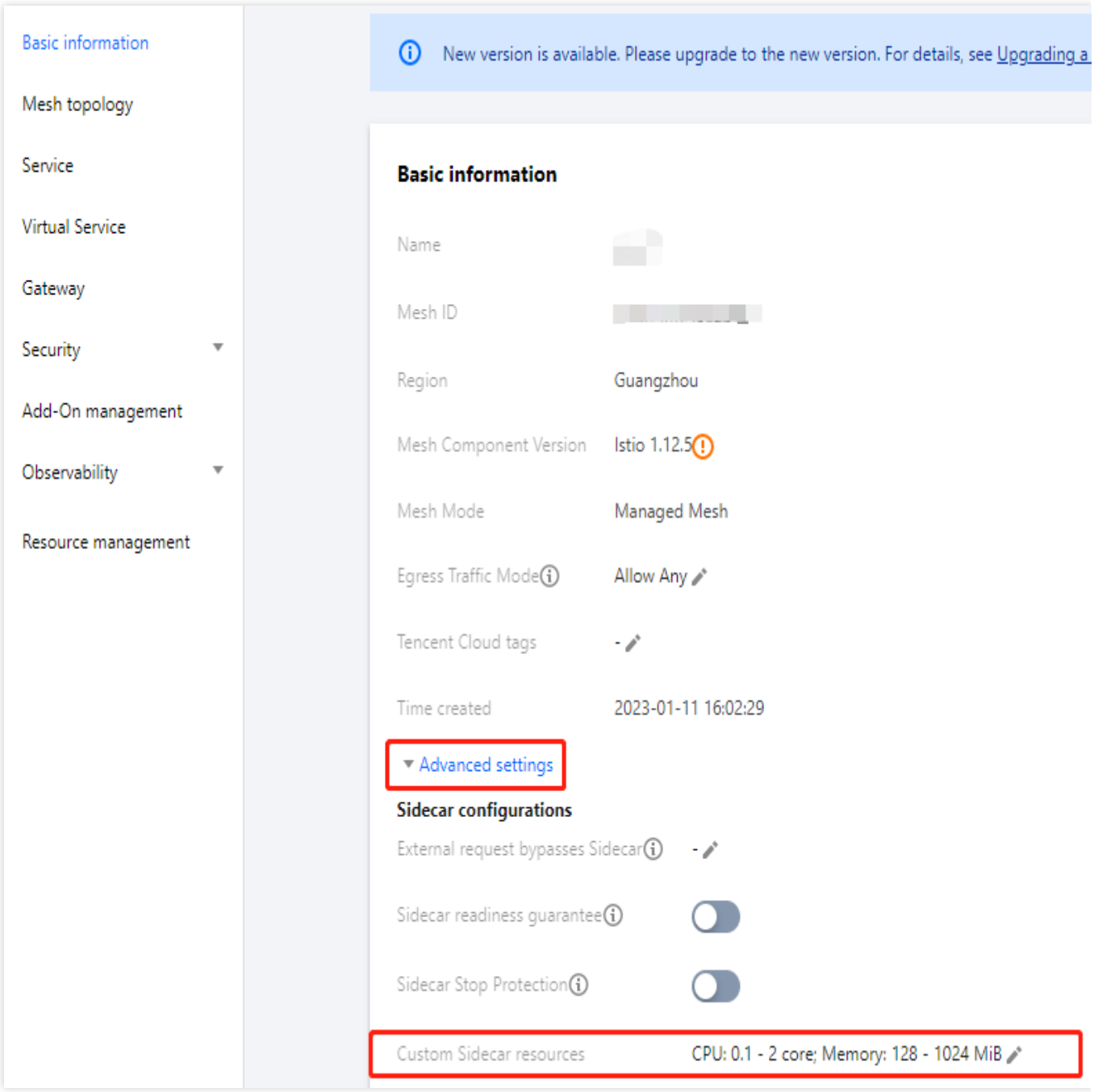
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        proxy.istio.io/config: '{ "proxyMetadata": { "EXIT_ON_ZERO_ACTIVE_CONNECTIONS": "true" }
      labels:
        app: nginx
```

Customizing Sidecar Resources

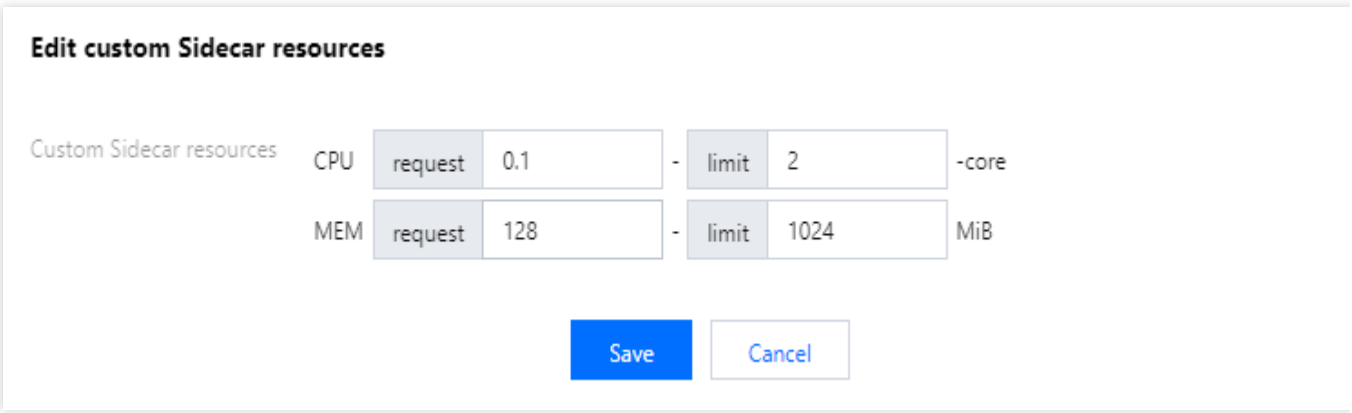
A sidecar is a container under a pod, and `request` and `limit` also need to be set for a sidecar. When necessary, you can make global custom configuration in the advanced settings area on the mesh basic information page. The steps are as follows:

1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh management page.
2. On the basic information page, click

on the right of **Custom Sidecar resources**. See the figure below:



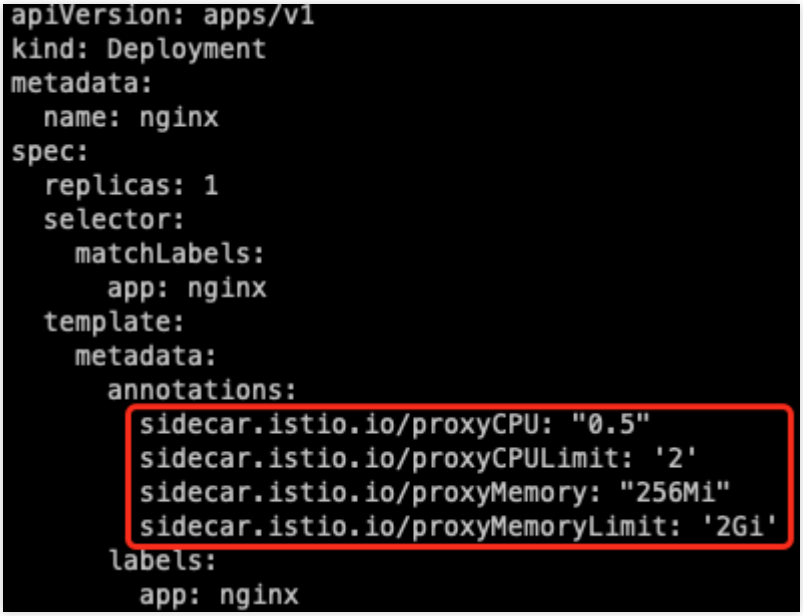
3. In the **Custom Sidecar resources** window that pops up, edit custom resources. See the figure below:



4. Click **Save**.

To apply different custom sidecar resources for different workloads, add annotations similar to the following to the pod:

```
sidecar.istio.io/proxyCPU: "0.5"
sidecar.istio.io/proxyCPULimit: '2'
sidecar.istio.io/proxyMemory: "256Mi"
sidecar.istio.io/proxyMemoryLimit: '2Gi'
```



If TKE Serverless is used and you do not want to increase the pod specifications significantly due to sidecar `request` and `limit` settings, you can use annotations to set `request` but not `limit`. You can set `request` as needed. The value `0` indicates that the pod specifications will not be upgraded due to sidecars.

```
sidecar.istio.io/proxyCPU: "0"
sidecar.istio.io/proxyMemory: "0"
```

Deleting a Mesh

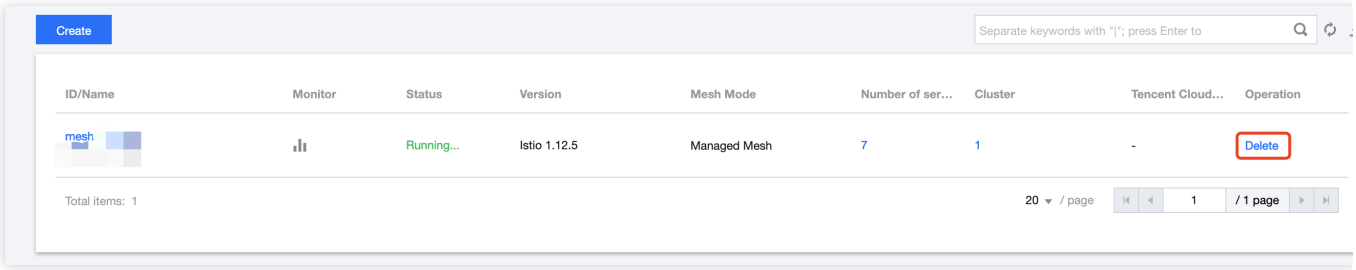
Last updated：2023-12-26 11:45:35

Overview

This section describes how to delete a service mesh instance.

Directions

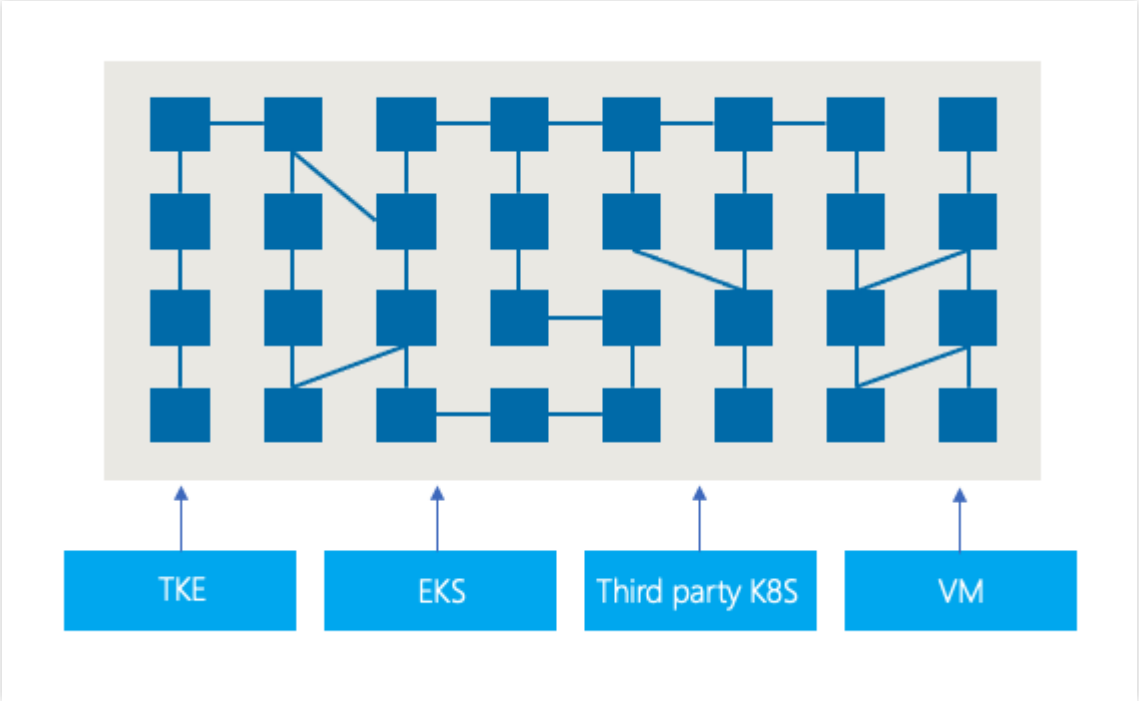
1. Log in to the Tencent Cloud Mesh console to enter the mesh list page.
2. At the top of the page, select the region where the service mesh belongs.
3. Click **Delete** in the **Operation** column at which the mesh to be deleted is located, and confirm the deletion.



Service Discovery Management Overview

Last updated：2023-12-26 11:45:53

Service discovery is to add specific services to a mesh. It is a prerequisite for service governance and observation. Tencent Cloud Mesh supports automatic discovery of services in K8s clusters. You only need to add clusters to the mesh, including TKE and EKS clusters provided by Tencent Cloud, and third-party K8s clusters registered with TKE. For other services other than K8s, such as VM, cloud database, you can manually register them with the mesh by configuring ServiceEntry, WorkloadGroup, and WorkloadEntry.



For details about how to add K8s clusters and heterogeneous services to Tencent Cloud Mesh, see the following sections:

Automatic Service Discovery

Manual Service Discovery

Automatic Service Discovery

Last updated：2023-12-26 11:46:10

Overview

A Tencent Cloud Mesh can associate with multiple TKE clusters and automatically discover K8s services in the clusters. You can associate multiple TKE clusters when creating the mesh or on the mesh basic information page, and Tencent Cloud Mesh will automatically display the services in the clusters on the **Service** page.

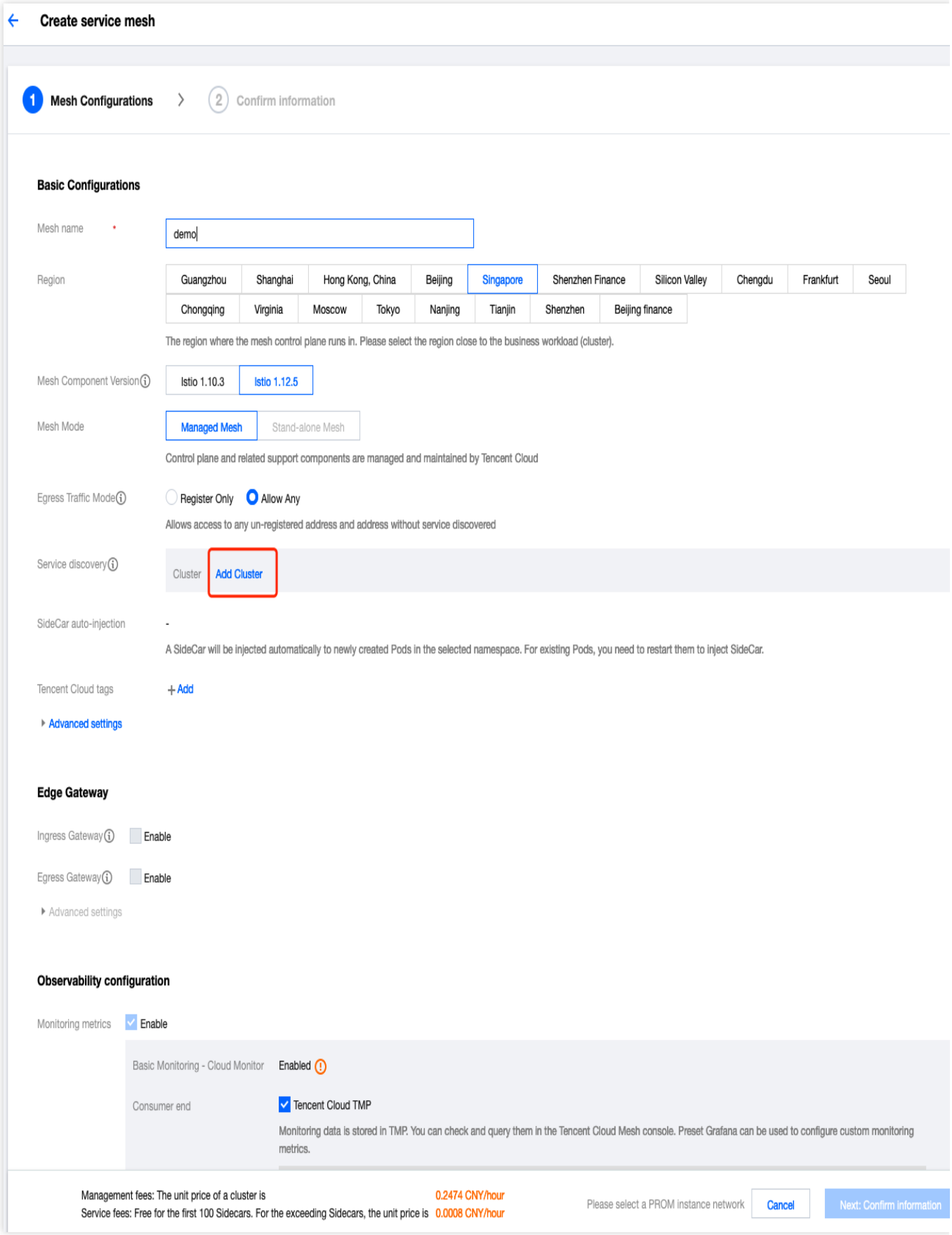
Limits

Cluster quota: A single mesh supports up to 10 K8s clusters by default, and the clusters in the mesh are deployed across up to three regions. After the quota is exceeded, you cannot add more clusters to the mesh.
Cluster version: Tencent Cloud Mesh does not enforce that the cluster versions are exactly the same, but the cluster versions should meet requirements of Istio for the corresponding K8s versions. For details, see Supported Releases.
Cluster permission: You need to have admin permissions for the cluster to be added to the mesh. For details, see Adding Mesh Permissions for a Cluster.
VPC network: To ensure the normal access to pods across multiple clusters that are not in the same VPC, use CCN to connect these clusters. Add the clusters to the same CCN instance. **Ensure that the host CIDR and container CIDR in the VPC at each end of the CCN instance do not conflict.**
Container network: If a TKE cluster uses the Global Router mode, you need to register the container network with CCN, so that newly added container CIDRs can be accessed.

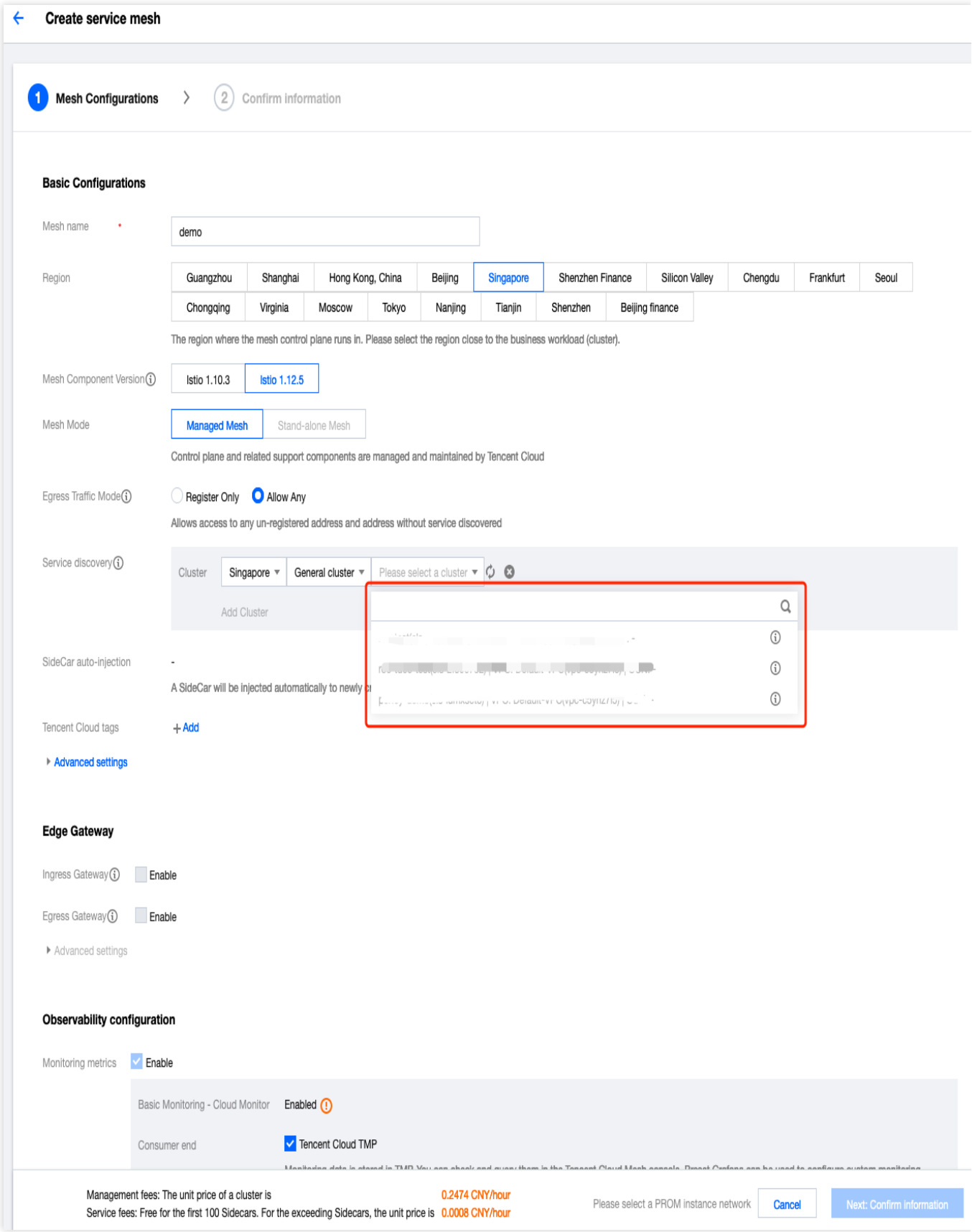
Directions

Mesh creation page

You can add an automatic service discovery cluster when creating a mesh on the mesh creation page.
1. Log in to the Tencent Cloud Mesh console.
2. Click **Create** to create a service mesh.
3. Click **Add cluster** next to **Service discovery** under **Basic information**.


Create service mesh
1 Mesh Configurations
2 Confirm information
Basic Configurations
Mesh name
demo
Region
Guangzhou
Shanghai
Hong Kong, China
Beijing
Singapore
Shenzhen Finance
Silicon Valley
Chengdu
Frankfurt
Seoul
Chongqing
Virginia
Moscow
Tokyo
Nanjing
Tianjin
Shenzhen
Beijing finance
The region where the mesh control plane runs in. Please select the region close to the business workload (cluster).
Mesh Component Version
Istio 1.10.3
Istio 1.12.5
Mesh Mode
Managed Mesh
Stand-alone Mesh
Control plane and related support components are managed and maintained by Tencent Cloud
Egress Traffic Mode
Register Only
Allow Any
Allows access to any un-registered address and address without service discovered
Service discovery
Cluster
Add Cluster
SideCar auto-injection
-
A SideCar will be injected automatically to newly created Pods in the selected namespace. For existing Pods, you need to restart them to inject SideCar.
Tencent Cloud tags
+ Add
Advanced settings
Edge Gateway
Ingress Gateway
Enable
Egress Gateway
Enable
Advanced settings
Observability configuration
Monitoring metrics
Enable
Basic Monitoring - Cloud Monitor
Enabled
Consumer end
Tencent Cloud TMP
Monitoring data is stored in TMP. You can check and query them in the Tencent Cloud Mesh console. Preset Grafana can be used to configure custom monitoring metrics.
Management fees: The unit price of a cluster is 0.2474 CNY/hour
Service fees: Free for the first 100 Sidecars. For the exceeding Sidecars, the unit price is 0.0008 CNY/hour
Please select a PROM instance network
Cancel
Next: Confirm information

4. Select one or more Kubernetes automatic service discovery clusters to be added. After the mesh creation request is submitted, the created mesh instance can automatically discover K8s services in the cluster.

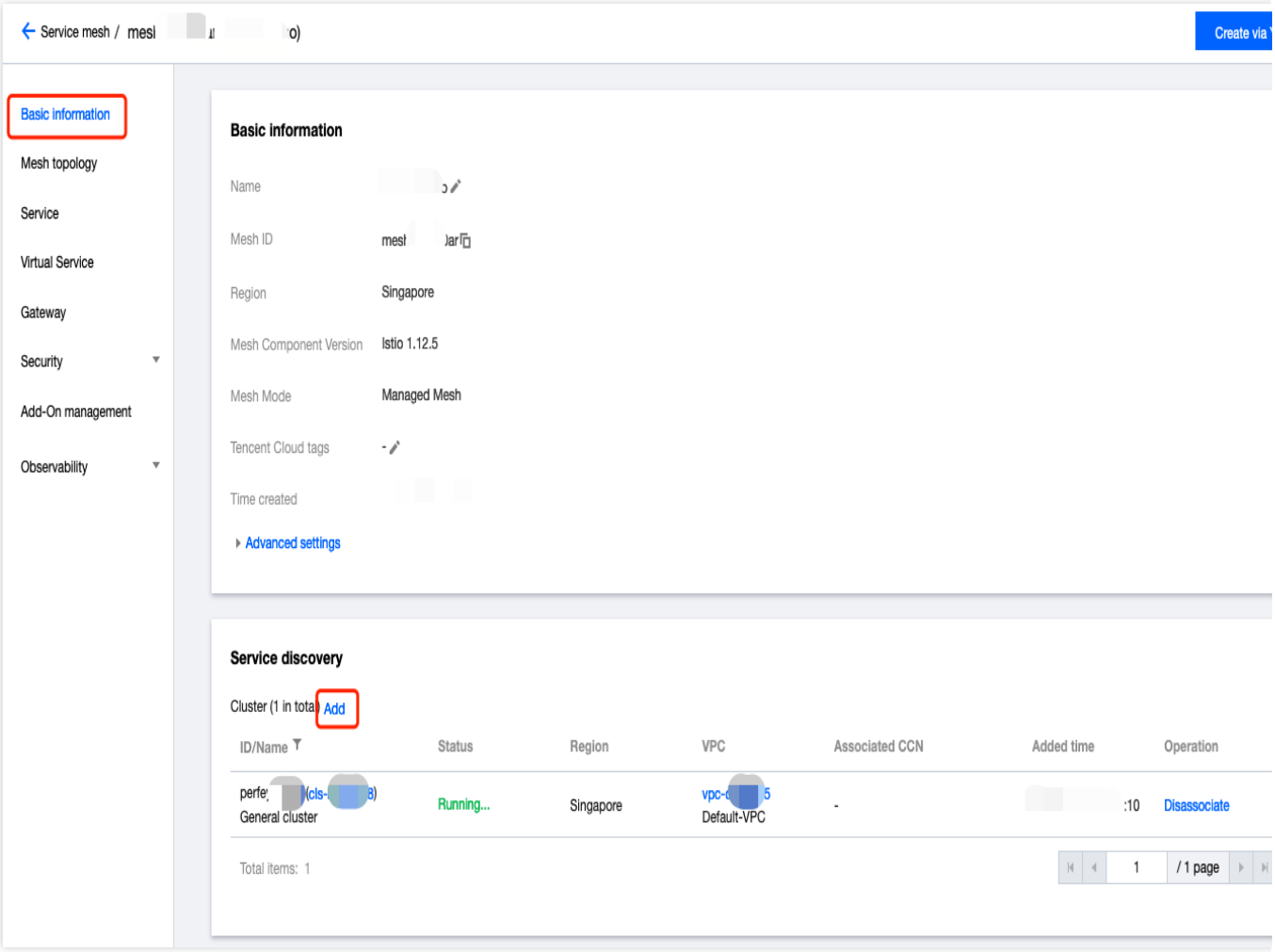


Mesh details page

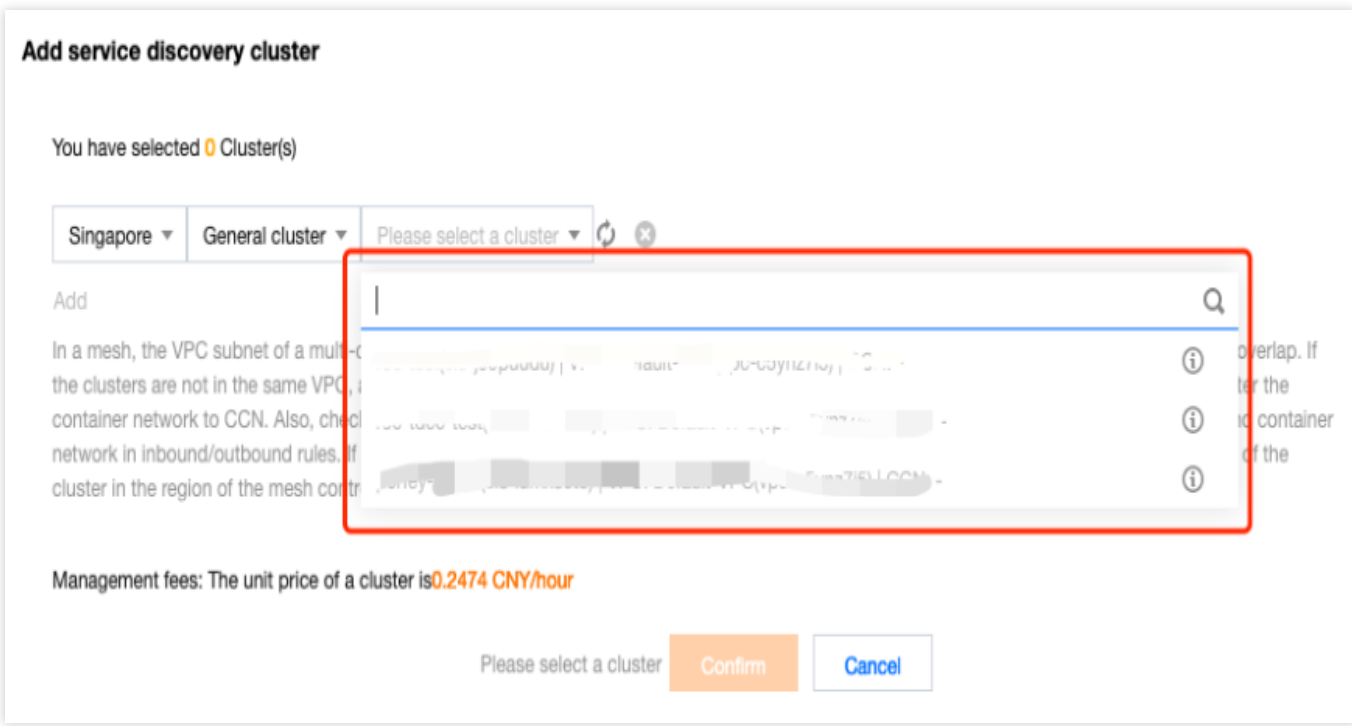
On the mesh details page, you can view the service discovery clusters associated with the current mesh instance, and add or disassociate an automatic service discovery cluster.

Adding a service discovery cluster

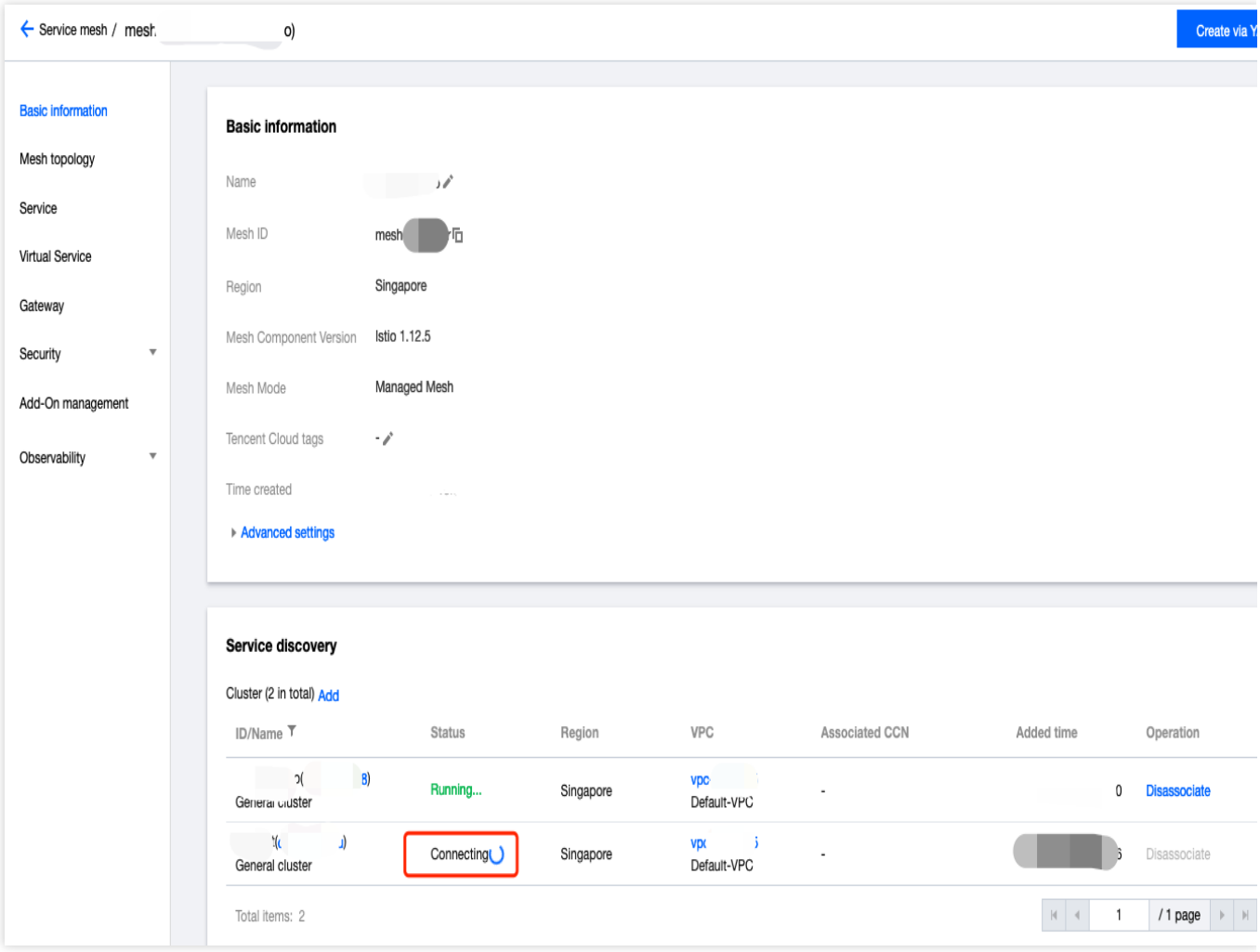
1. Go to the mesh details page, and click **Basic information** in the sidebar. In the **Service discovery** module, you can view the list of service discovery clusters associated with the mesh. Then, click **Add** to pop up the **Add service discovery cluster** window.



2. In the **Add service discovery cluster** window, select one or more Kubernetes service discovery clusters to be added, and click **OK**.



3. After the request for adding a Kubernetes service discovery cluster is submitted, wait for the cluster to be connected. After the cluster is connected, addition of the Kubernetes service discovery cluster is complete.



Note:

After the service is added to the mesh, you need to inject a sidecar into the service and then perform management operations on the service, such as traffic management and visual observation. For related guidelines, see Mesh Configuration.

Disassociating a service discovery cluster

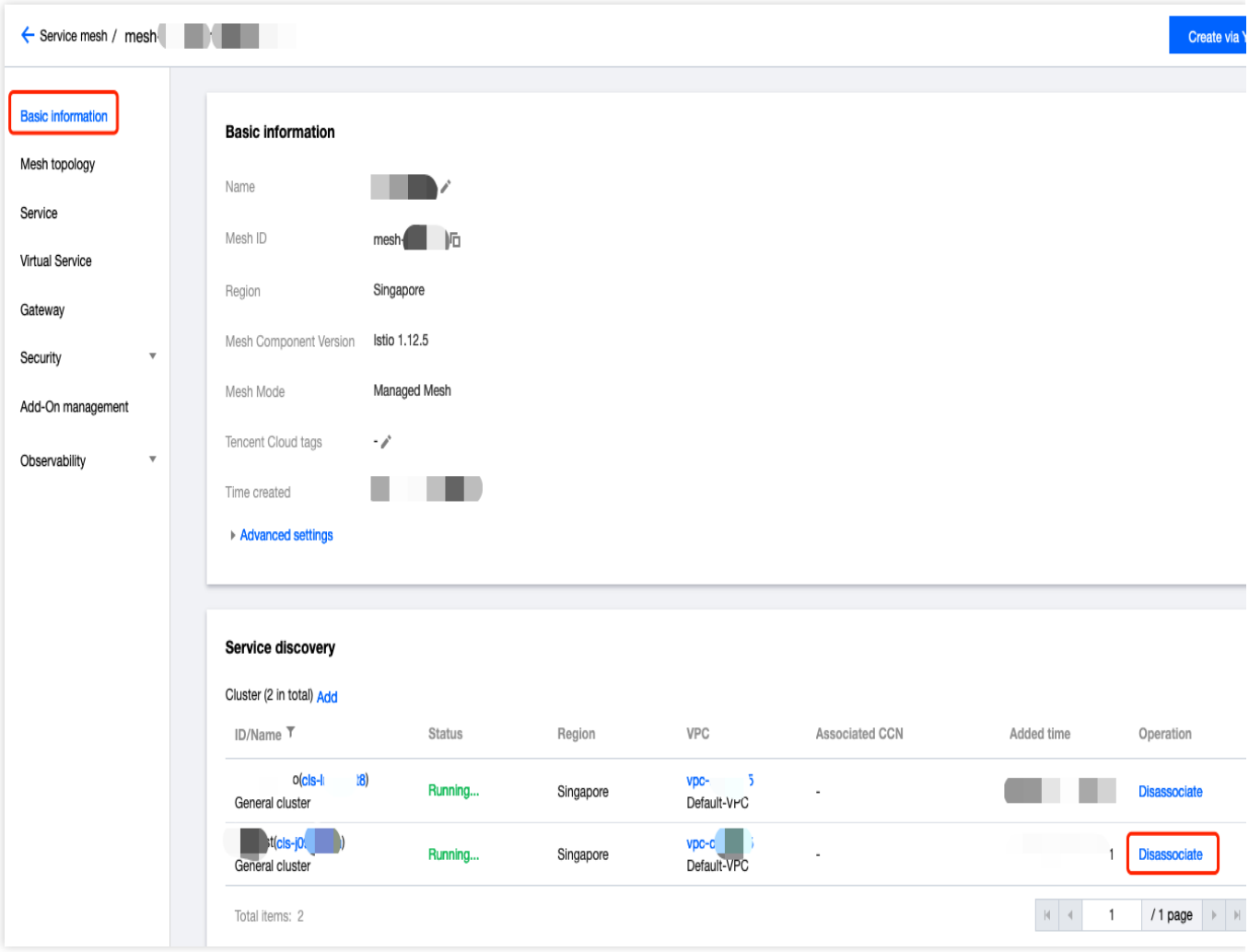
You need to disassociate a service discovery cluster that does not need to participate in mesh management or a deleted cluster to avoid unnecessary fees. You can follow the following steps:

Note：

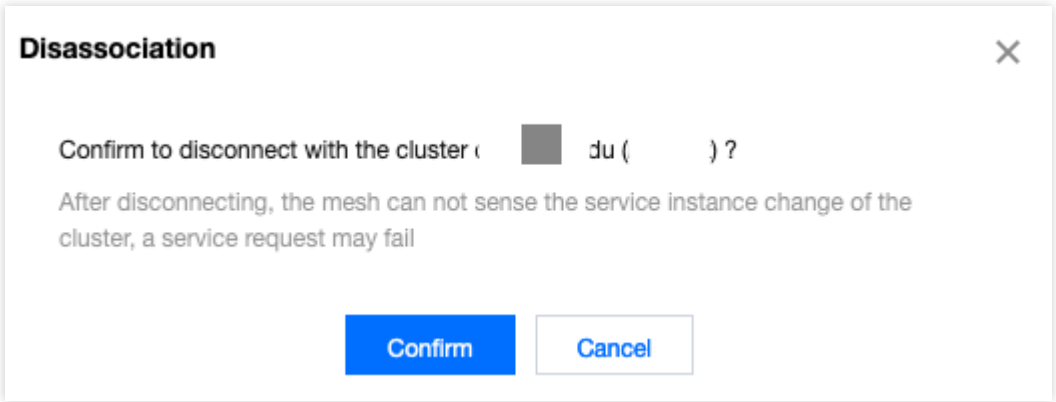
For a deleted cluster, Tencent Cloud Mesh will not automatically disassociate it for you, but will not charge cluster management fees any longer.

If the only cluster in the mesh is deleted, Tencent Cloud will force you to disassociate it to ensure normal mesh experience.

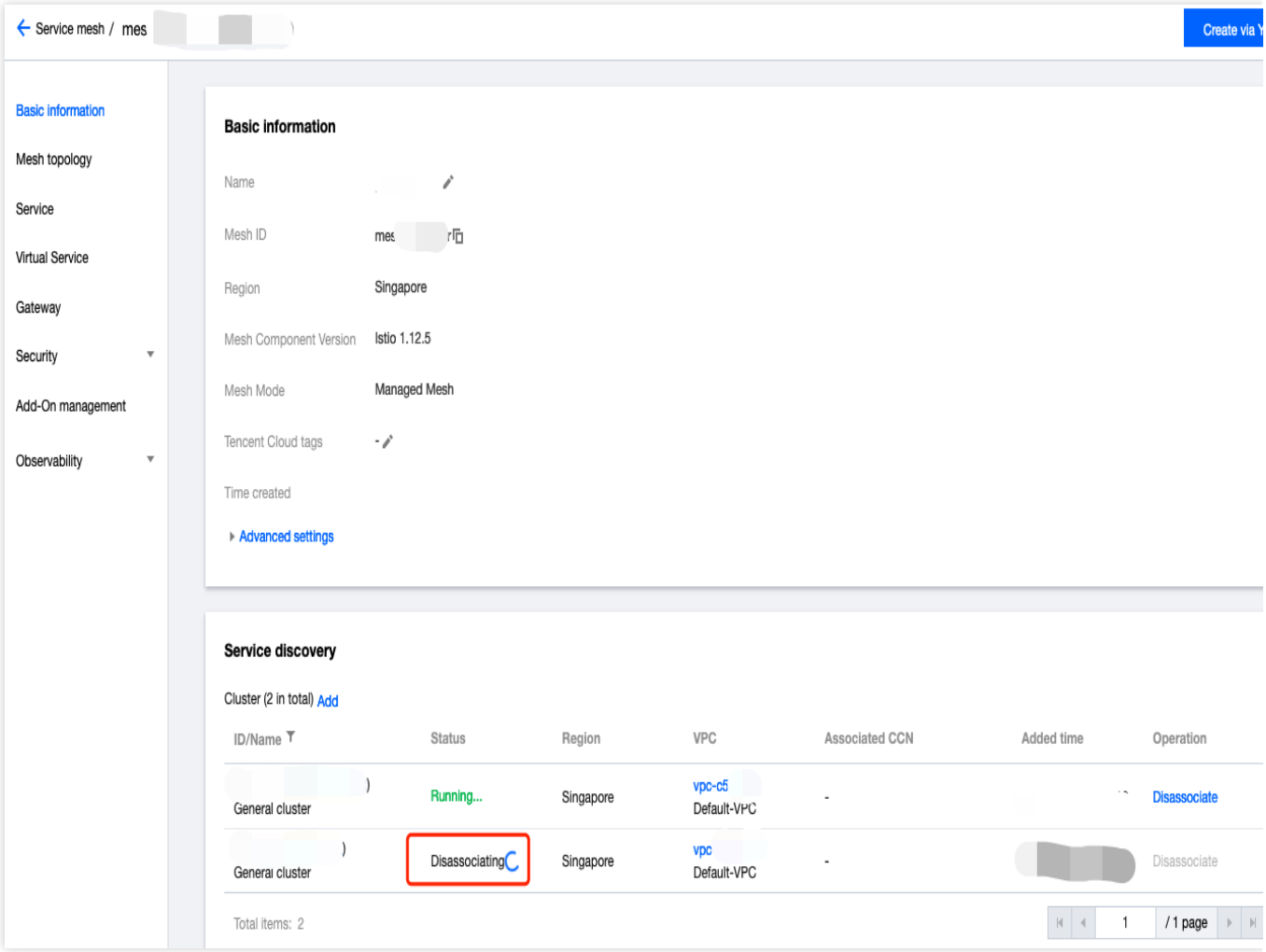
1. Go to the mesh details page, and click **Basic information** in the sidebar. In the **Service discovery** module, you can view the list of service discovery clusters associated with the mesh. Then, in the **Operation** column where the cluster to be disassociated resides, click **Disassociate** to pop up a dialog box for confirming disassociation.



2. In the **Disassociation** dialog box, confirm the information about the service discovery cluster to be deleted, and click **OK** to submit the cluster disassociation request. After the cluster is disassociated, the mesh is no longer aware of service instance changes in the cluster and related service requests may become abnormal.



3. Wait for the disassociation operation to complete.


Service mesh / mes
Create via Y
Basic information
Mesh topology
Service
Virtual Service
Gateway
Security
Add-On management
Observability
Basic information
Name
Mesh ID
Region
Singapore
Mesh Component Version
Istio 1.12.5
Mesh Mode
Managed Mesh
Tencent Cloud tags
Time created
Advanced settings
Service discovery
Cluster (2 in total) Add
ID/Name
Status
Region
VPC
Associated CCN
Added time
Operation
General cluster
Running...
Singapore
vpc-c5
Default-VPC
Disassociate
General cluster
Disassociating
Singapore
vpc
Default-VPC
Disassociate
Total items: 2
1
/ 1 page

Manual Service Registration

Last updated：2023-12-26 11:46:38

Overview

With Istio's ServiceEntry, WorkloadGroup, and WorkloadEntry mechanisms, you can add services in clusters that are not provided by TKE, such as traditional VM services and database services, on Tencent Cloud Mesh. However, if you want to manage and observe external services in the mesh in the same way as other automatically discovered K8s services such as applications deployed in VMs, you further need to install sidecars for applications of the external services through the WorkloadGroup and WorkloadEntry mechanisms. Currently, Tencent Cloud Mesh does not support automatic sidecar installation, you need to install sidecars manually. For detailed instructions, see Virtual Machine Installation.

Note:

Concepts

ServiceEntry is similar to the concept Service in K8s. After a service is added to a mesh through ServiceEntry, it can be accessed by other automatically discovered services in the mesh based on routing rules.

Similar to the concept Deployment in K8s, WorkloadGroup is used to ServiceEntry deployments.

Similar to the concept Pod in K8s, WorkloadEntry is used to map a specific entity application.

Description of Major ServiceEntry Fields

Name	Type	Description
spec.hosts	string	Host name in the URL of a service. Multiple host names are allowed.
spec.ports	Port[]	Port number of the service. Multiple port numbers are allowed.
spec.resolution	string	Static: A static endpoint IP address is used as a service instance. DNS: The endpoint IP address of the service is resolved through DNS, which is mostly used for external services. A declared endpoint needs to use the DNS domain name, and the service is resolved to the host domain name if no endpoint is available. NONE: This option is selected when the service does not require IP resolution.
spec.location	string	Specify whether the service is in the mesh. Some Istio features cannot be used by services outside the mesh. For example, services outside the mesh do not support mTLS. **MESH_EXTERNAL** represents a service outside the mesh, and **MESH_INTERNAL** represents a service in the mesh.
spec.endpoints	String	Endpoints associated with the service. Multiple endpoints can be entered, but only one endpoint is used at a time.

Description of Major WorkloadGroup Fields

Name	Type	Description
spec.metadata.label	string	Label associated with a WorkloadEntry.
spec.template	string	Basic information about generation of the WorkloadEntry.
sepc.probe	string	Parameter settings about health check on the WorkloadEntry.

Description of Major WorkloadEntry Fields

Name	Type	Description
spec.address	string	Address of the current endpoint. It is similar to a pod IP address.
spec.labels	string	Labels of the current endpoint. They are used to associate with the ServiceEntry.
sepc.serviceAccount	string	Permission information about a sidecar. This field must be specified when you need to add a sidecar for the endpoint.

For details about ServiceEntry and WorkloadEntry, see Service Entry and Workload Entry.

Manually Registering a Service

Currently, Tencent Cloud Mesh allows you to add a ServiceEntry on the console or by using yaml.

YAML Configuration Example

Console Configuration Example

ServiceEntry

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: external-svc-https
spec:
  hosts:
  - api.dropboxapi.com
  - www.googleapis.com
  - api.facebook.com
  location: MESH_EXTERNAL
```

```
  ports:
  - number: 443
    name: https
    protocol: TLS
  resolution: DNS
```

WorkloadGroup

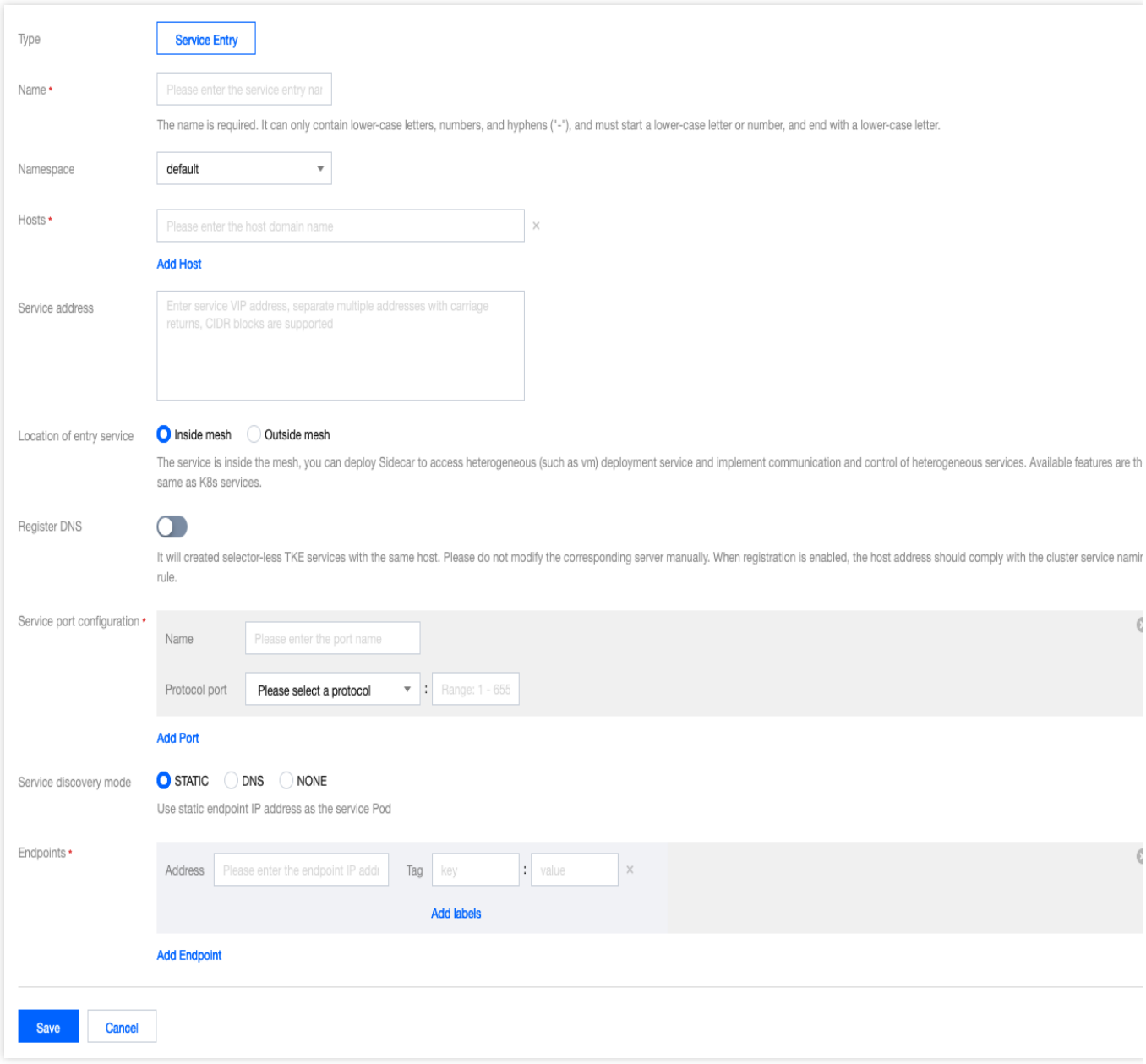
```
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadGroup
metadata:
  name: reviews
  namespace: bookinfo
spec:
  metadata:
    labels:
      app.kubernetes.io/name: reviews
      app.kubernetes.io/version: "1.3.4"
  template:
    ports:
      grpc: 3550
      http: 8080
    serviceAccount: default
  probe:
    initialDelaySeconds: 5
    timeoutSeconds: 3
    periodSeconds: 4
    successThreshold: 3
    failureThreshold: 3
    httpGet:
     path: /foo/bar
     host: 127.0.0.1
     port: 3100
     scheme: HTTPS
     httpHeaders:
     - name: Lit-Header
       value: Im-The-Best
```

WorkloadEntry

```
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadEntry
metadata:
  name: details-svc
```

```
spec:
  serviceAccount: details-legacy
  address: 2.2.2.2
  labels:
    app: details-legacy
    instance-id: vm1
```

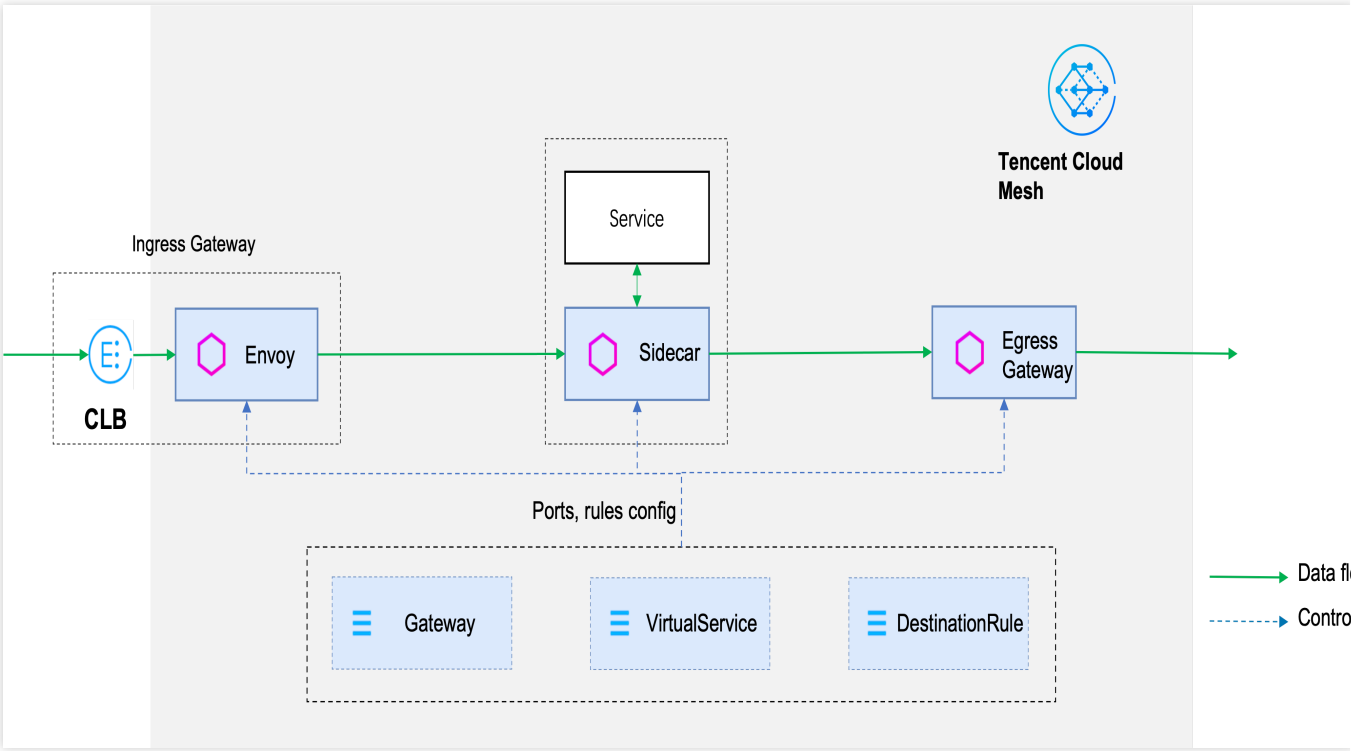
1. Log in to the Tencent Cloud Mesh console.
2. Click **ID/Name** to pop up the mesh details page.
3. Click **Service** > **Create**, and specify service basic information. This operation can register a non-containerized third-party service with Tencent Cloud Mesh.



Gateway
Gateway Management

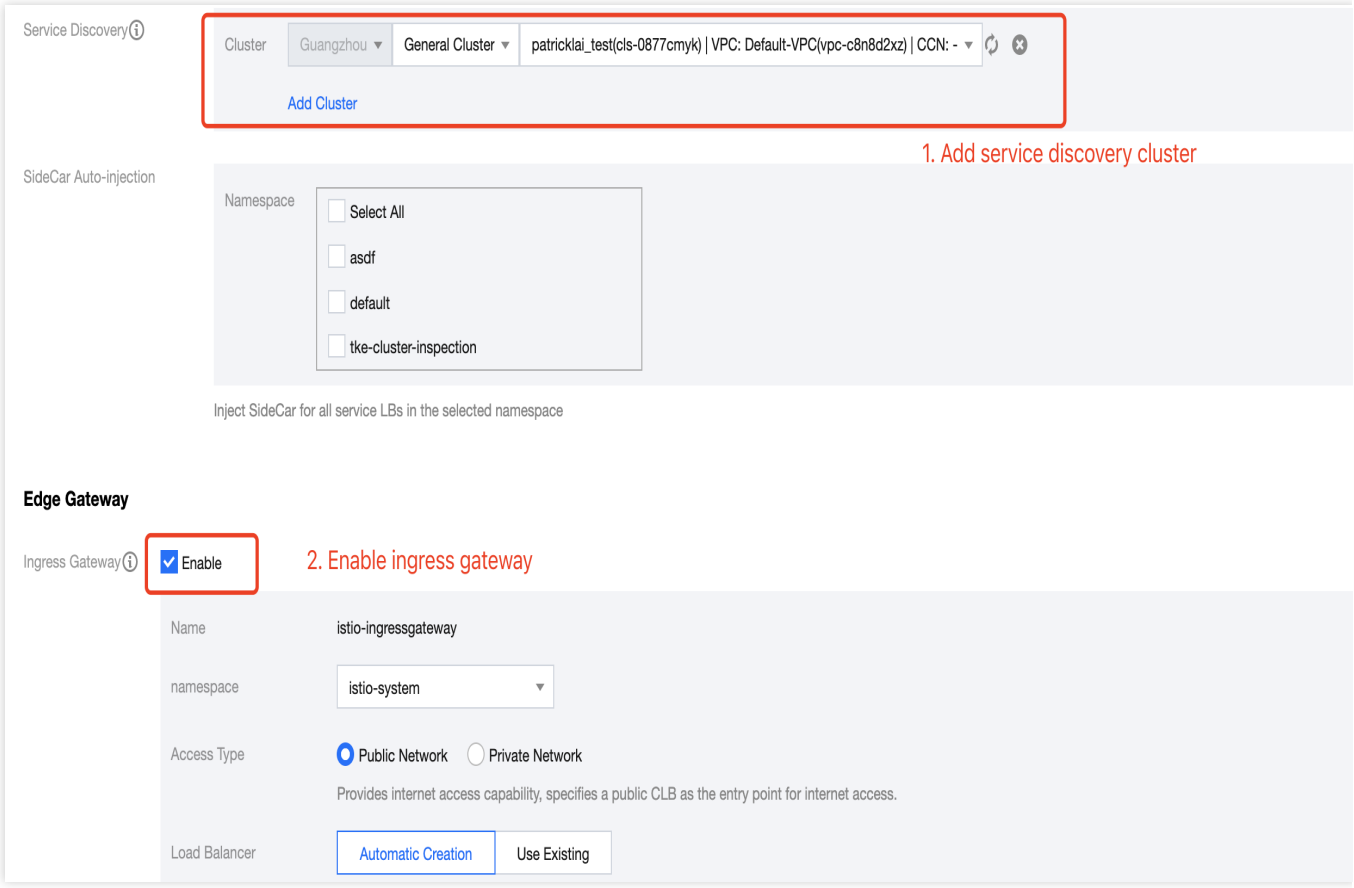
Last updated：2023-12-26 11:46:59

A gateway is a special data plane responsible for load balancing between ingress and egress traffic of a mesh. It is deployed as an independent pod but not a sidecar in your cluster. It is classified into two types: ingress gateway and egress gateway. An ingress gateway instance contains an Envoy pod on the data plane and its associated CLB instance (public network or private network). Tencent Cloud Mesh provides a managed gateway controller, which has implemented automatic integration of ingress gateway configurations and CL. You can configure the ingress gateway by using Istio CRDs. Tencent Cloud Mesh automatically synchronizes the related configurations to the associated CLB instance. The synchronized configurations include port configurations and enhanced port listening rule configurations. In other words, the Envoy container and associated CLB feature are used as a whole to provide you with ingress gateway capabilities.
If you need the capability to balance the ingress and egress traffic of the mesh, you need to create an ingress gateway or egress gateway instance, and then configure listening rules and traffic management (routing) rules of the gateway by using Istio CRDs such as Gateway, VirtualService, and DestinationRule. The listening rules are configured by using the Gateway CRD, and the traffic management rules are configured by using the VirtualService and DestinationRule CRDs (consistent with the east-west traffic management syntax). The following figure is a schematic diagram of the relationship between gateway instances and Istio CRD configurations.

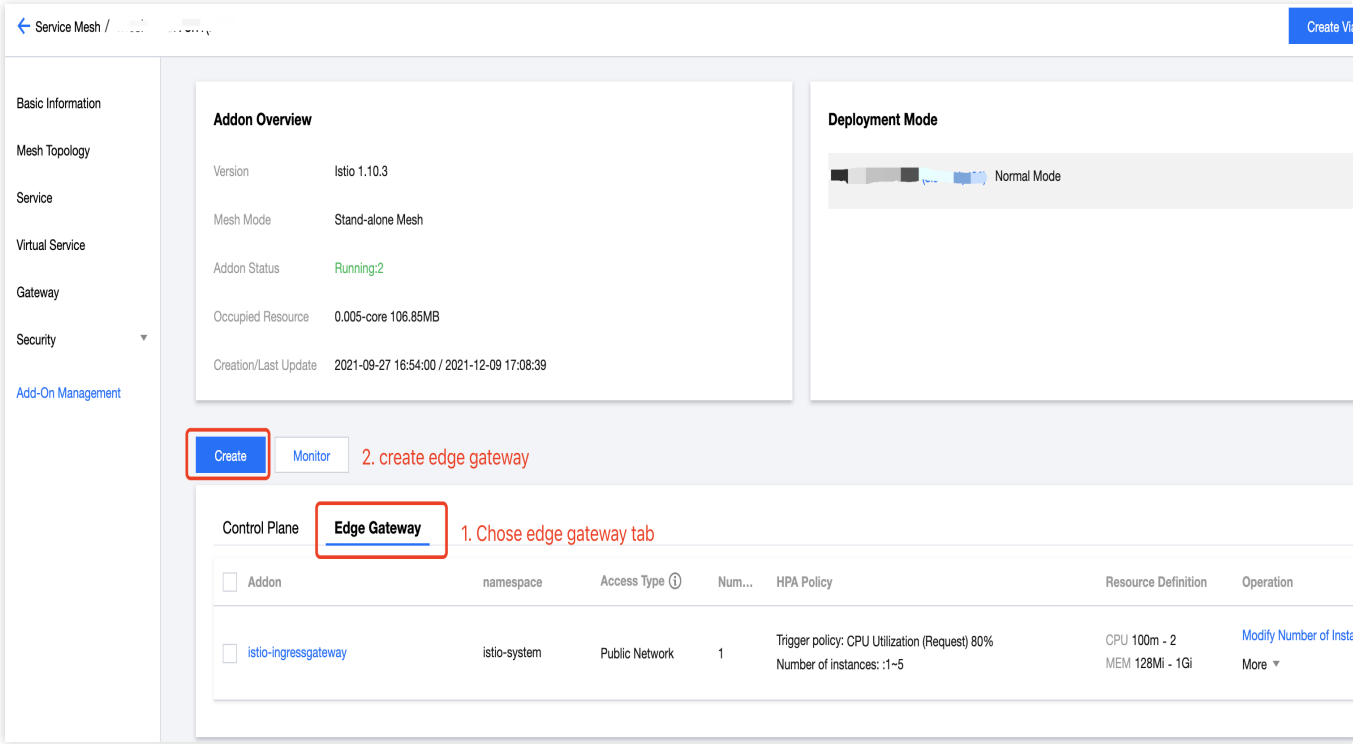


Creating a Gateway

1. Log in to the Tencent Cloud Mesh console.
2. On the mesh creation page, add a service discovery cluster and then create a gateway.



Alternatively, on the **Edge gateway** tab page of the mesh details page, click **Create** to create a gateway.



Major configuration items for creating a gateway are described as follows.

Configuration Item	Description
Type	Whether an ingress gateway or an egress gateway is to be created.
Access Cluster	Kubernetes cluster in which the gateway is to be created.
Namespace	Namespace in which the gateway is to be created.
Access type	Ingress gateway parameter. Select a CLB access type. **Public network** and **Private network** are supported.
Load Balancer	Ingress gateway parameter. Select **Automatic creation** or **Use existing**. For more information about using existing CLBs, see Using Existing CLBs.
Billing mode	Ingress gateway parameter. Select a CLB billing mode. **Bill-by-traffic** and **Bill by bandwidth** are supported. For more information about CLB billing, see Billing Overview.
Bandwidth cap	Ingress gateway parameter. Select a CLB bandwidth cap, which ranges from 0 to 2048 Mbps.
CLB-to-Pod direct access	Ingress gateway parameter. For example, when the network mode for the gateway to access the cluster is VPC-CNI, **CLB-to-Pod direct access** can be enabled. In this case, traffic is not forwarded through NodePort, so as to improve the performance. Preservation of client source IP, and pod-level session persistence and health check are supported. For more details, see Using Services with CLB-to-Pod Direct Access Mode.
Preserve client source IP	Ingress gateway parameter. Set ExternalTrafficPolicy to **Local** in the ingress gateway service to preserve the client source IP, and enable Local binding and Local weighted balancing. This parameter becomes invalid if **CLB-to-Pod direct access** is enabled. For more details, see Service Backend Selection.
Component Configurations	Configurations about CPU and memory resources and HPA policies of the gateway.

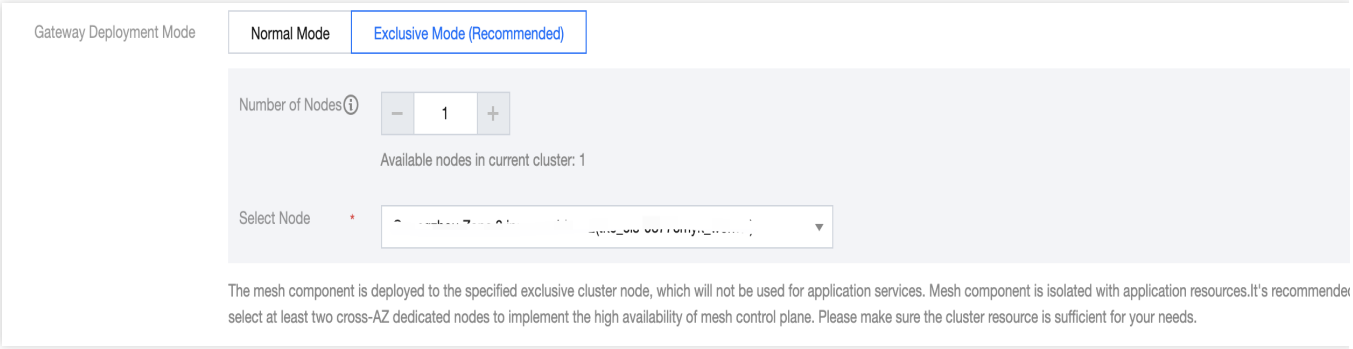
Gateway Deployment Modes

Two gateway deployment modes are available: **Normal mode** and **Exclusive mode**.

Normal mode：A gateway service is deployed in a selected service cluster and is deployed indistinguishably from other service pods.

Exclusive mode：In some scenarios, a gateway is deployed on an exclusive node to improve service stability. In the exclusive mode, you need to add some cluster nodes to an exclusive resource pool, and then the mesh sets taints for

the selected nodes to ensure exclusive use. After settings, all ingress/egress gateways will be deployed only on the selected nodes. You can further specify nodes for a specific gateway in the advanced settings.
You can adjust the gateway deployment mode on the mesh creation page or the component management page.
Note:
Adjusting the deployment mode will trigger gateway service scheduling, which may adversely affect service traffic.

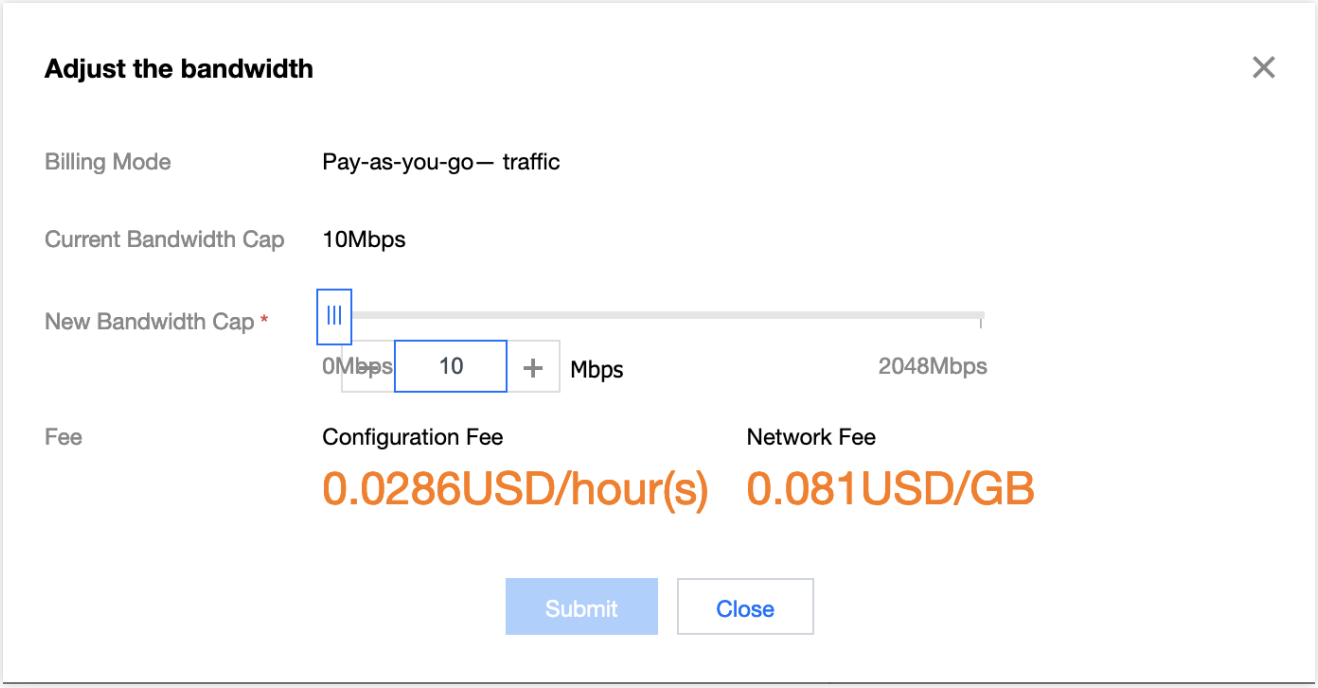


Updating Gateway Configurations

After a gateway is created, you can modify the associated CLB bandwidth (supported only for an ingress gateway), the number of instances, HPA policies, and resource definitions of the gateway.

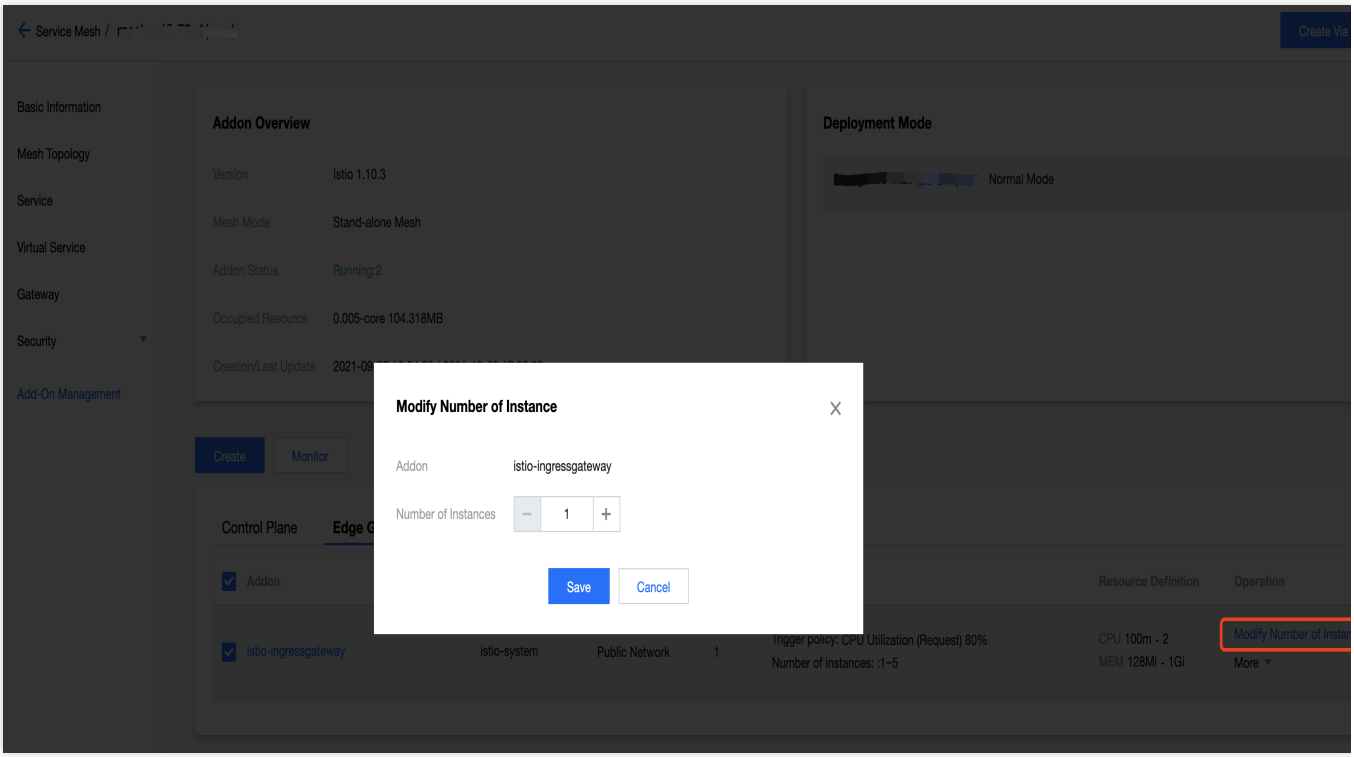
Modifying the CLB Bandwidth

You can modify the bandwidth of the CLB instance associated with an ingress gateway. In the gateway area on the **Basic information** tab page on the mesh details page, you can edit configurations of the CLB associated with the ingress gateway.



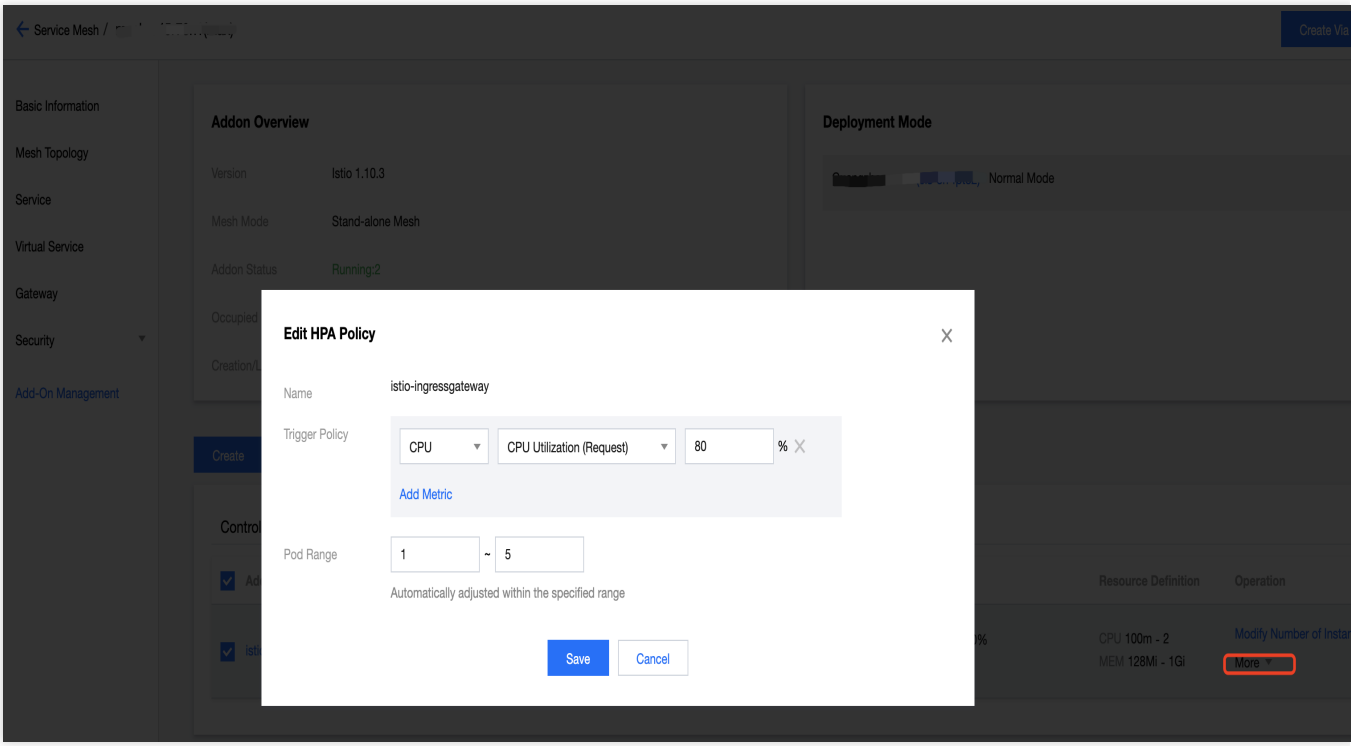
Modifying the Number of Component Instances

You can adjust the number of component instances by choosing **Mesh details** > **Component management**.



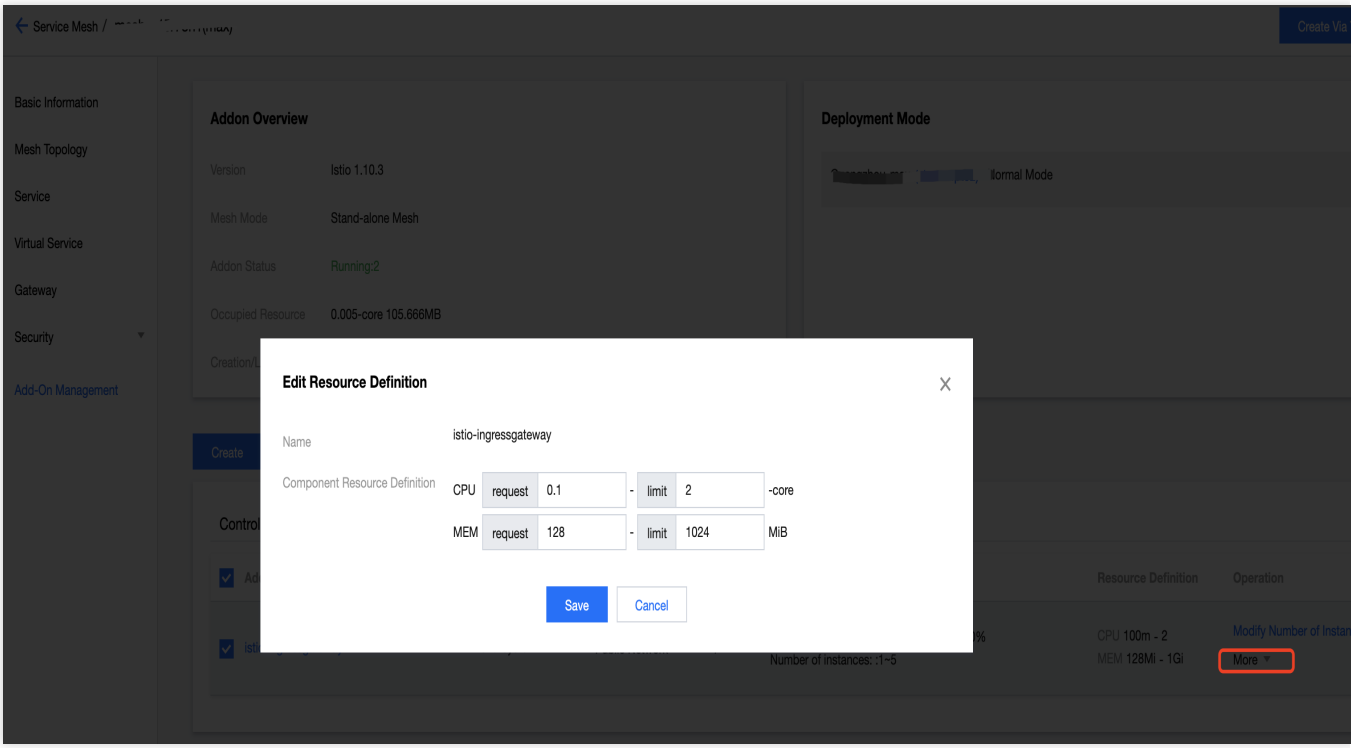
Modifying HPA Policies of Components

You can edit HPA policies of components by choosing **Mesh details** > **Component management**. Scaling policies can be configured based on CPU, memory, mesh, and hard disk metrics.



Modifying Component Resource Definitions

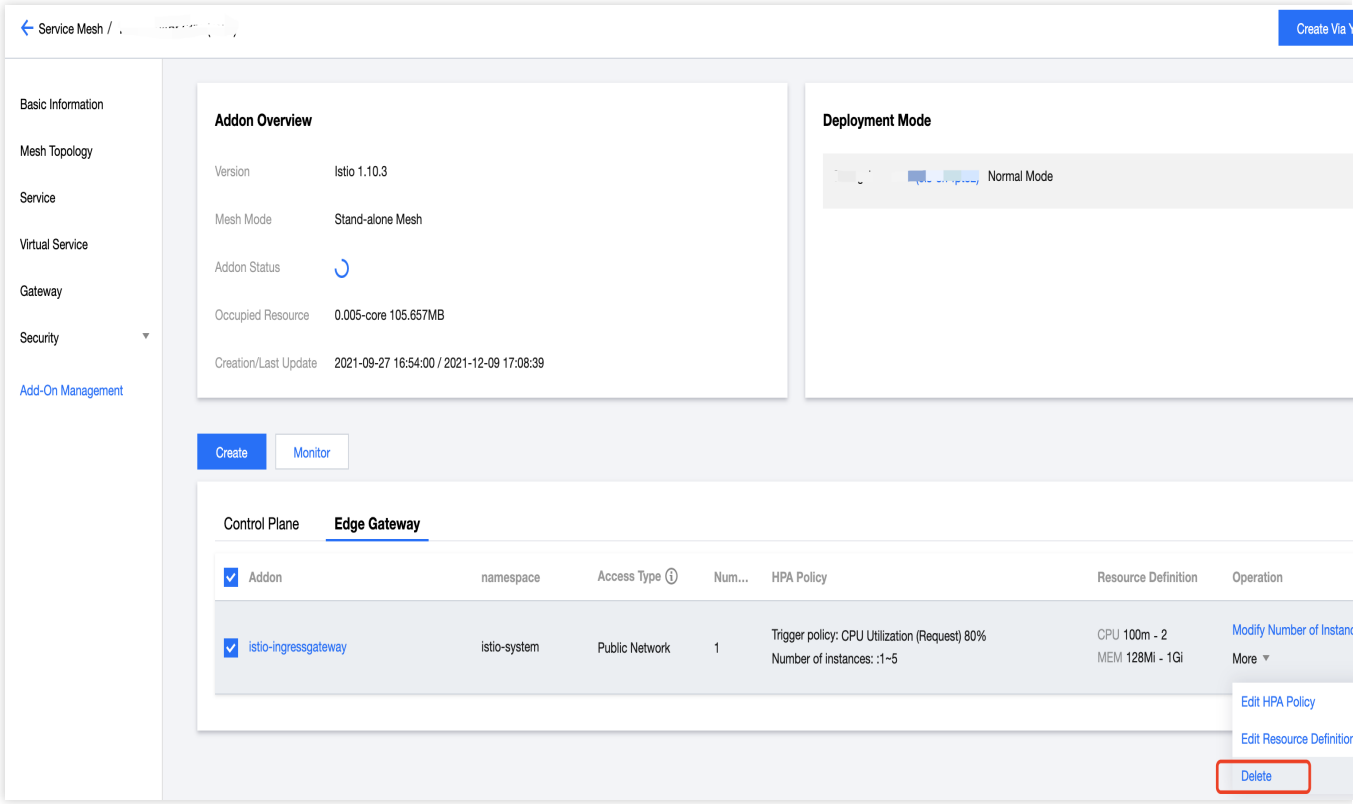
You can edit component resource definitions, including CPU request, CPU limit, memory request, and memory limit, by choosing **Mesh details** > **Component management**.



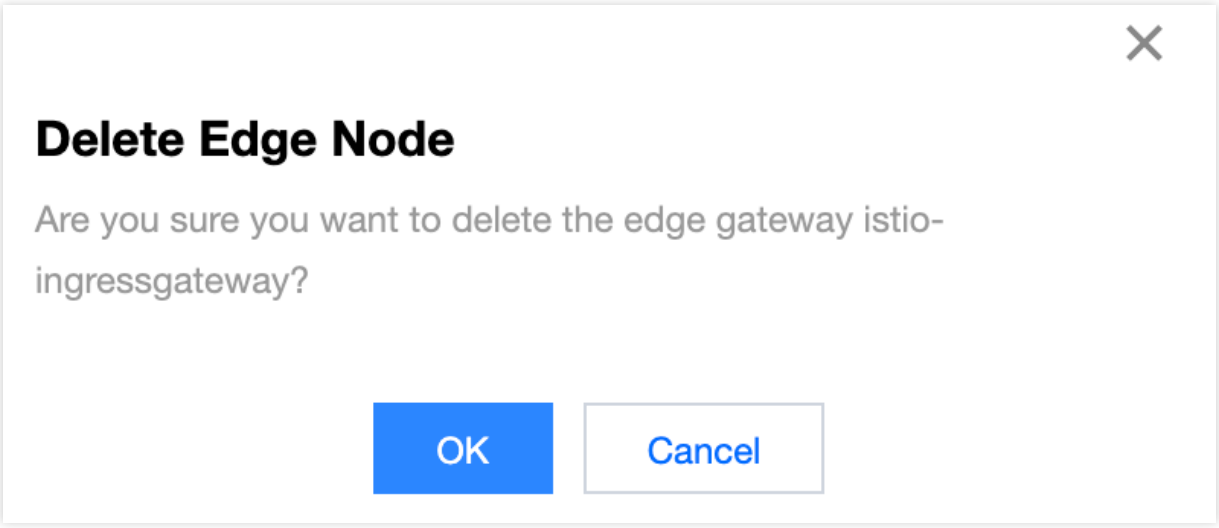
Deleting a Gateway

You can delete a specified gateway by choosing **Mesh details** > **Component management** > **Edge gateway**. The procedure is as follows:

1. Access the mesh details page, click **Component management**, click **Edge gateway**, and choose **More** > **Delete** in the **Operation** column where the gateway to be deleted resides.



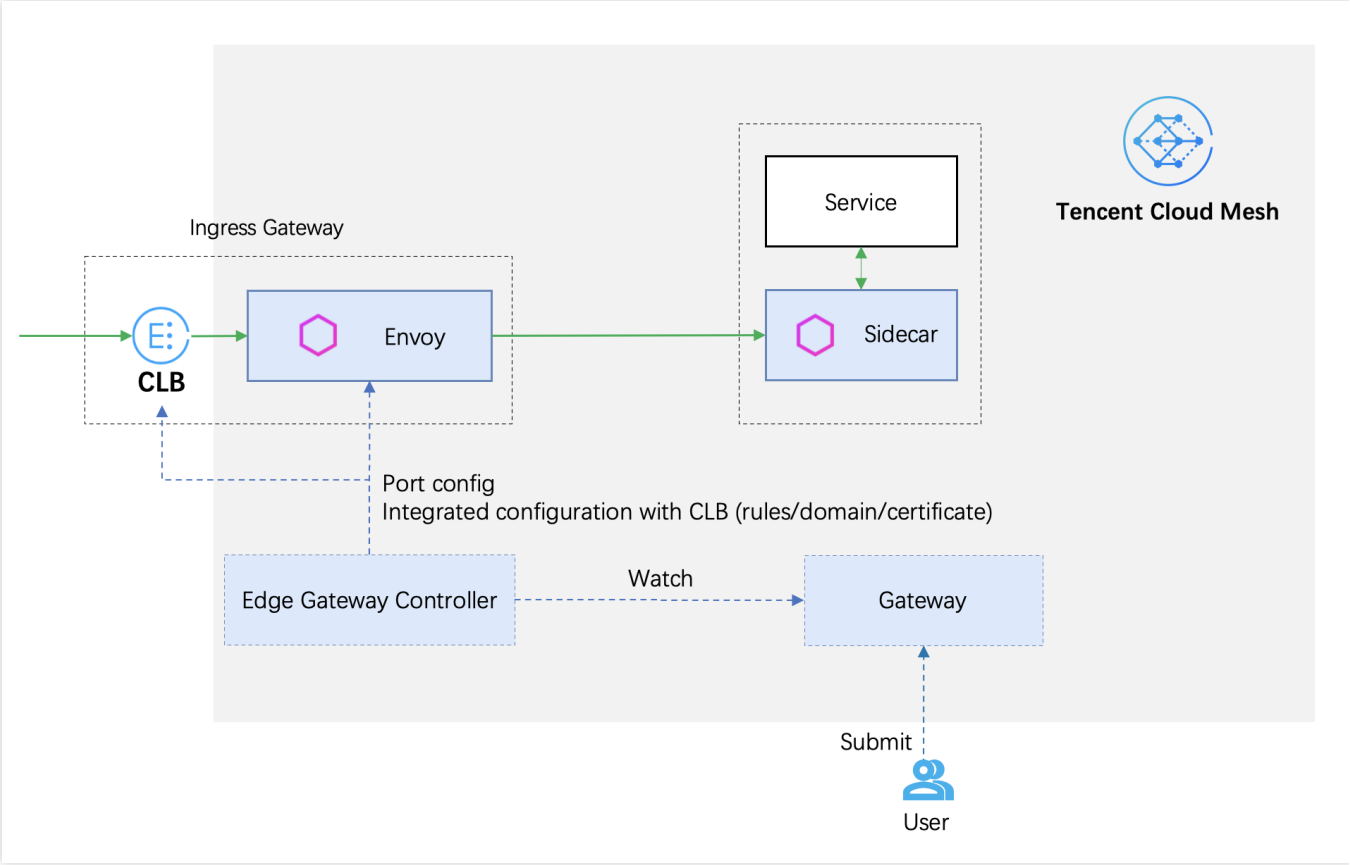
2. In the **Delete edge node** dialog box, confirm the name of the gateway to be deleted and click **OK**.



Automatic Interworking of the Gateway Controller of Tencent Cloud Mesh with CLB

Tencent Cloud Mesh implements the managed gateway controller. The controller monitors the gateway configurations delivered to an ingress gateway in real time, parses the current port configurations, and synchronizes the current port configurations to CLB, so that you no longer need to manually configure CLB ports. CLB ports, ingress gateway service ports, and ingress gateway container ports are in one-to-one mapping. To be specific, if the `80` port is defined in the Gateway CRD, the gateway controller of Tencent Cloud Mesh will configure the container port as `80` and the service port as `80` for the ingress gateway instance and enables the `80` port of the associated CLB synchronously.

The gateway controller of Tencent Cloud Mesh also implements the feature of enabling SSL certificate offloading to take place at CLB. In this way, after certificate offloading takes place at CLB, the ingress gateway provides traffic management capabilities. After this feature is configured on the gateway, the gateway controller will resolve the port, domain name, and certificate that are involved in feature configurations, and synchronize the configurations to the CLB instance bound to the ingress gateway.



Gateway Configuration

Last updated：2023-12-26 11:47:28

Ports and monitoring rules of a gateway are configured by using a gateway CRD. The following is a gateway configuration example, with major fields being explained by comments:

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: gateway-sample
  namespace: default
spec:
  selector: # Match pods delivered by the gateway configurations based on the enter
    istio: ingressgateway
    app: istio-ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - uk.bookinfo.com
    - eu.bookinfo.com
    tls:
      httpsRedirect: true # Send a 301 https redirect.
  - port:
      number: 443
      name: https-443
      protocol: HTTPS # Enable HTTPS ports.
    hosts:
    - uk.bookinfo.com
    - eu.bookinfo.com
    tls:
      mode: SIMPLE # TLS one-way authentication
      serverCertificate: /etc/certs/servercert.pem # Load the certificate in the fi
      privateKey: /etc/certs/privatekey.pem
  - port:
      number: 9443
      name: https-9443
      protocol: HTTPS # Enable HTTPS ports.
    hosts:
    - "bookinfo-namespace/*.bookinfo.com"
    tls:
      mode: SIMPLE # TLS one-way authentication
      credentialName: bookinfo-secret # Load the certificate from the Kubernetes se
```

```
  - port:
      number: 5443
      name: https-ssl
      protocol: HTTPS # Enable HTTPS ports.
    hosts:
    - "*"
    tls:
      mode: SIMPLE # TLS one-way authentication
      credentialName: qcloud-abcdABCD # Load the certificate with the certificate I
  - port:
      number: 6443
      name: clb-https-6443-ABCDabcd # Have certificate offloading on port 6443 to t
      protocol: HTTP
    hosts:
    - "tcm.tencent.com"
```

Gateway Configuration Field Description

Major fields of the gateway CRD are described as follows.

Name	Type	Description
`metadata.name`	`string`	Gateway name.
`metadata.namespace`	`string`	Gateway namespace.
`spec.selector`	`map<string, string>`	Label key-value pair used by the gateway to match the gateway instances delivered by the configurations.
`spec.servers.port.number`	`uint32`	Port number.
`spec.servers.port.protocol`	`string`	Communication protocol. The following protocols are supported: `HTTP, HTTPS, GRPC, HTTP2, MONGO, TCP, TLS` . Note that the protocol configurations of the same port on the same gateway need to be consistent.
`spec.servers.port.name`	`string`	Port name. Currently, Tencent Cloud Mesh implements the feature of enabling SSL certificate offloading to

		take place at CLB based on the port name. If you need to configure this feature, you can set the port name in the format of `clb-https-{port number}-{SSL certificate ID}`. This feature takes effect only when the current port communication protocol is set to **HTTP**. The gateway controller automatically creates a CLB layer-7 listener to implement certificate offloading. After SSL offloading is completed at CLB, the CLB instance and the ingress gateway pod adopt plaintext communication. Note that the certificate offloading configurations of the same port on the same gateway need to be consistent; otherwise, a configuration conflict occurs.
`spec.severs.hosts`	`string[]`	Domain name, which supports wildcard `*`.
`spec.servers.tls.httpsRedirect`	`bool`	When the value is `true`, the gateway returns a 301 redirect to all HTTP requests, requiring the client to initiate an HTTPS request.
`spec.servers.tls.mode`	-	TLS security authentication mode of the current port. Specify this field if you need to enable security authentication of the current port. The following values are supported: `PASSTHROUGH, SIMPLE, MUTUAL, AUTO_PASSTHROUGH, ISTIO_MUTUAL`.
`spec.servers.tls.credentialName`	`string`	Name of the secret from which the TLS certificate key is found. Tencent Cloud Mesh supports loading the certificate and key from the Kubernetes secret in the same namespace of the ingress gateway instance. Ensure that the secret you entered contains the appropriate certificate and key. Tencent Cloud

		Mesh also implements the feature of loading a Tencent Cloud SSL certificate. If you specify this field in the format of `qcloud-{SSL certificate ID}`, the gateway controller of Tencent Cloud Mesh will load the SSL certificate for the gateway. Currently, Tencent Cloud Mesh supports loading only server certificates and private keys in SIMPLE mode (one-way authentication) from the SSL Certificate Service console.
`spec.servers.tls.serverCertificate`	`string`	Certificate path that needs to be entered when the TLS certificate key of the port is mounted in the file mount manner (not recommended; it is recommended that you enter the `credentialName` field to load the certificate private key). By default, Istio uses the istio-ingressgateway-certs secret in the namespace where the gateway locates to load the certificate to the path `/etc/istio/ingressgateway-certs`.
`spec.servers.tls.privateKey`	`string`	Private key path that needs to be entered when the TLS certificate key of the port is mounted in the file mount manner (not recommended; it is recommended that you enter the `credentialName` field to load the certificate private key). By default, Istio uses the istio-ingressgateway-certs secret in the namespace where the gateway locates to load the private key to the path `/etc/istio/ingressgateway-certs`.
`spec.servers.tls.caCertificates`	`string`	Root certificate path that needs to be entered when the TLS certificate key

		of the port is mounted in the file mount manner (not recommended; it is recommended that you enter the `credentialName` field to load the certificate private key). By default, Istio uses the istio-ingressgateway-ca-certs secret in the namespace where the gateway locates to load the root certificate to the path `/etc/istio/ingressgateway-ca-certs`. A root certificate needs to be configured in mutual authentication.

Examples

A configuration example for loading a certificate from a Kubernetes secret to an ingress gateway

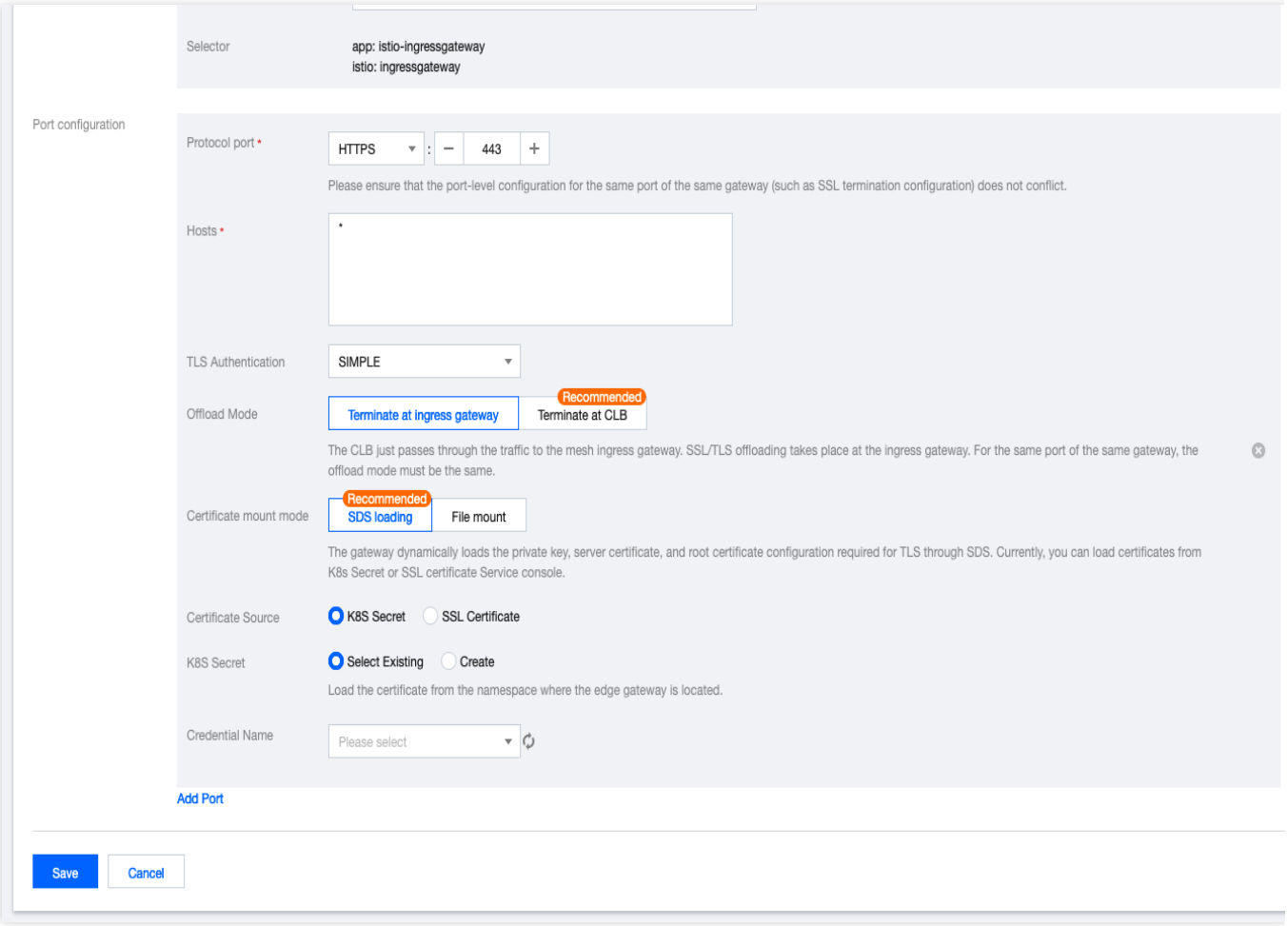
YAML Configuration Example

Console Configuration Example

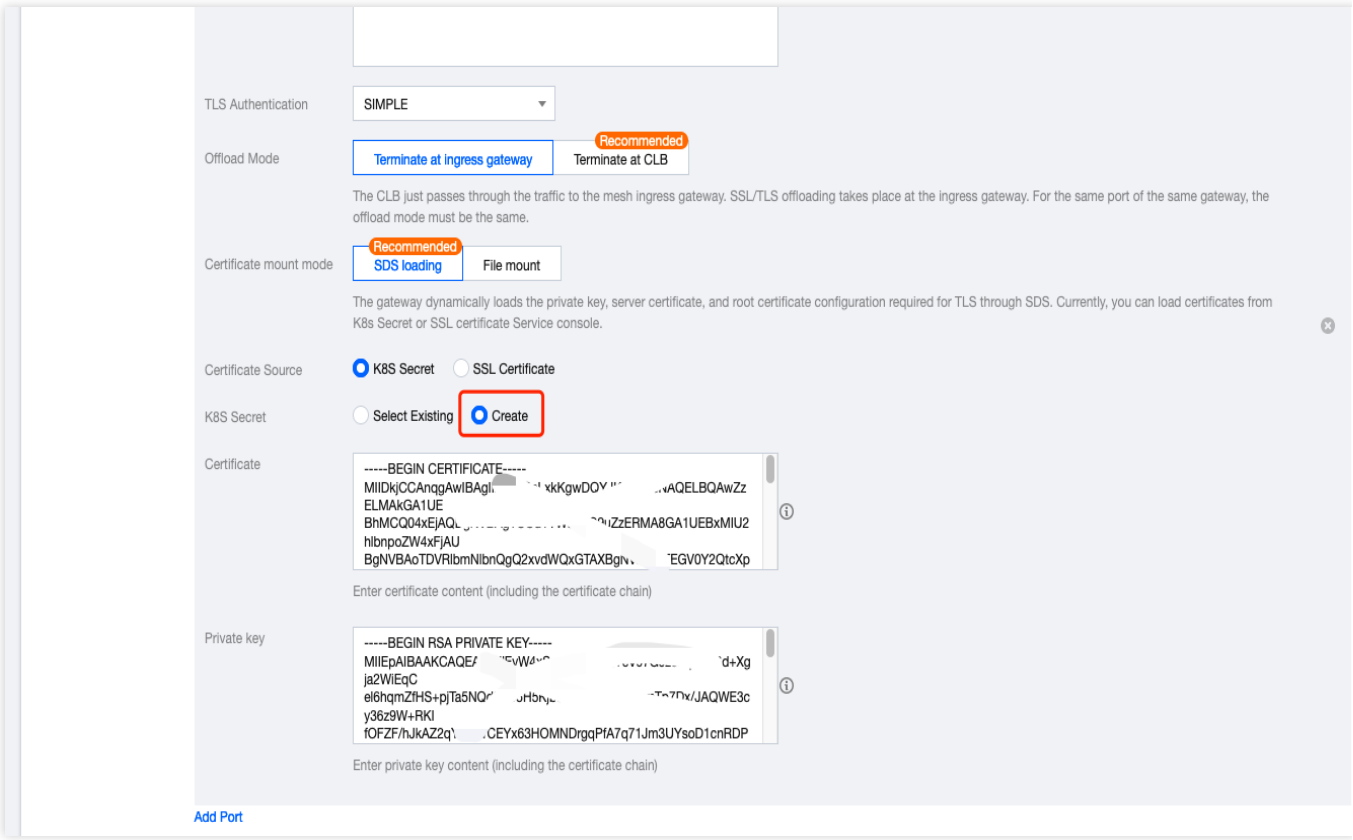
```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: sample-gw
  namespace: default
spec:
  servers:
    - port:
        number: 443
        name: HTTPS-443-6cph
        protocol: HTTPS
      hosts:
        - '*'
      tls:
        mode: SIMPLE
        credentialName: {kubernetes secret name}
  selector:
    app: istio-ingressgateway
    istio: ingressgateway
```

The process of creating gateway configurations in the console to load an HTTPS-based SSL certificate of an ingress gateway from a Kubernetes secret (one-way authentication) is as follows:

1. Select protocol **HTTPS** and **SIMPLE** for **TLS authentication**.
2. Select **Terminate at ingress gateway** for **Offload mode**.
3. Select **SDS loading** for **Certificate mount mode**.
4. Select **K8S secret** for **Certificate source**.
5. Select **Select existing** for **K8S secret**, and select the secret in the namespace where the selected ingress gateway locates. Ensure that the secret contains the appropriate certificate, private key, and root certificate.



6. If the secret does not contain any appropriate certificate, select **Create** for **K8S secret** and copy appropriate certificate, private key, and root certificate content to corresponding input boxes.



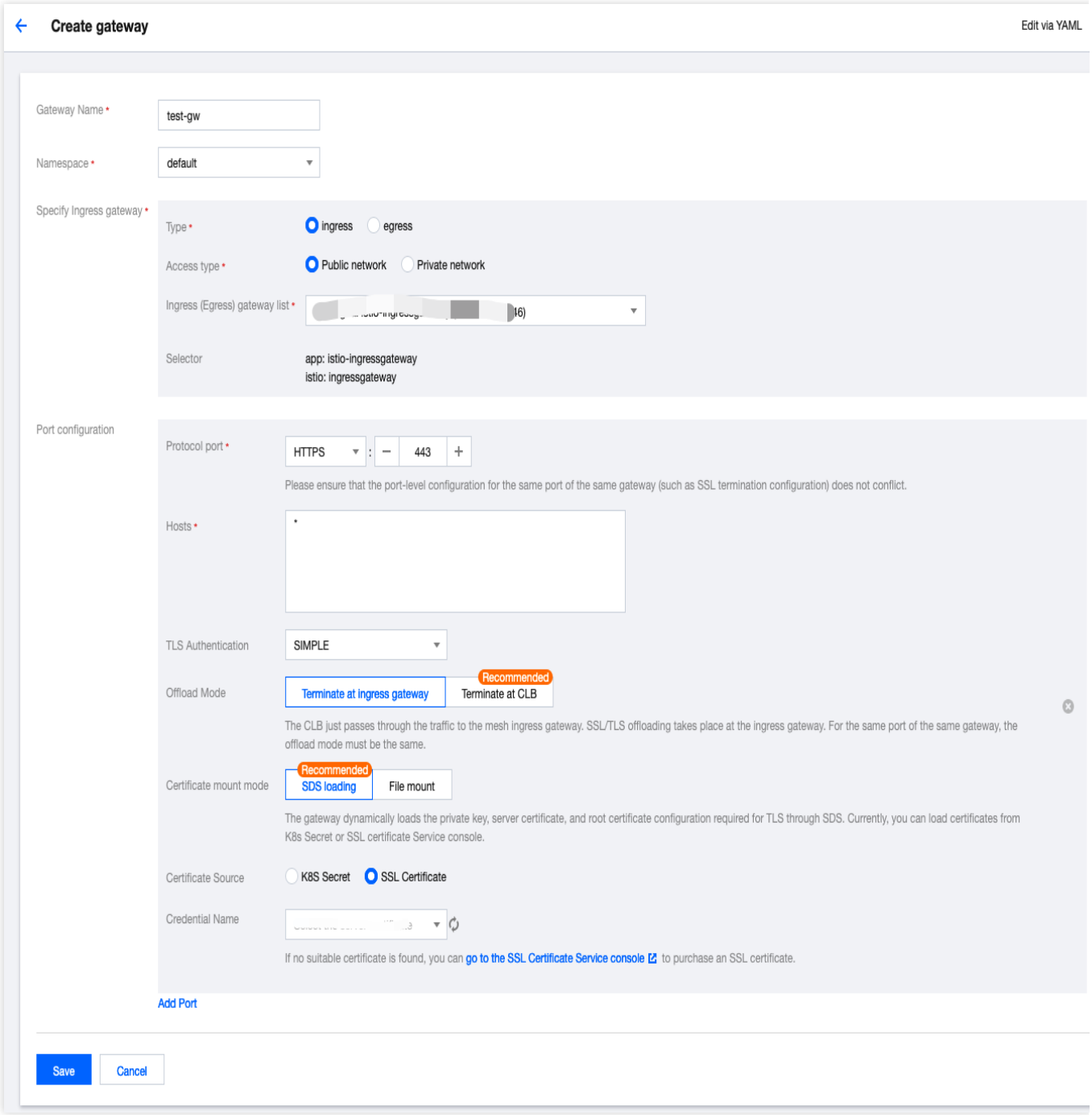
A configuration example for loading a certificate from the SSL Certificate Service console to an ingress gateway

YAML Configuration Example

Console Configuration Example

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: test-gw
spec:
  servers:
    - port:
        number: 443
        name: HTTPS-443-9ufr
        protocol: HTTPS
      hosts:
        - '*'
      tls:
        mode: SIMPLE
        credentialName: qcloud-{Certificate ID}
  selector:
    app: istio-ingressgateway
    istio: ingressgateway
```

In addition to configuring a gateway by using a YAML file, you can also create gateway configurations by using UI in the console. The following is a configuration example for loading a certificate from the SSL Certificate Service console to an ingress gateway. You can select the SSL certificate to be loaded by selecting **SSL certificate** for **Certificate source**.



A configuration example for SSL certificate offloading to take place at CLB

YAML Configuration Example

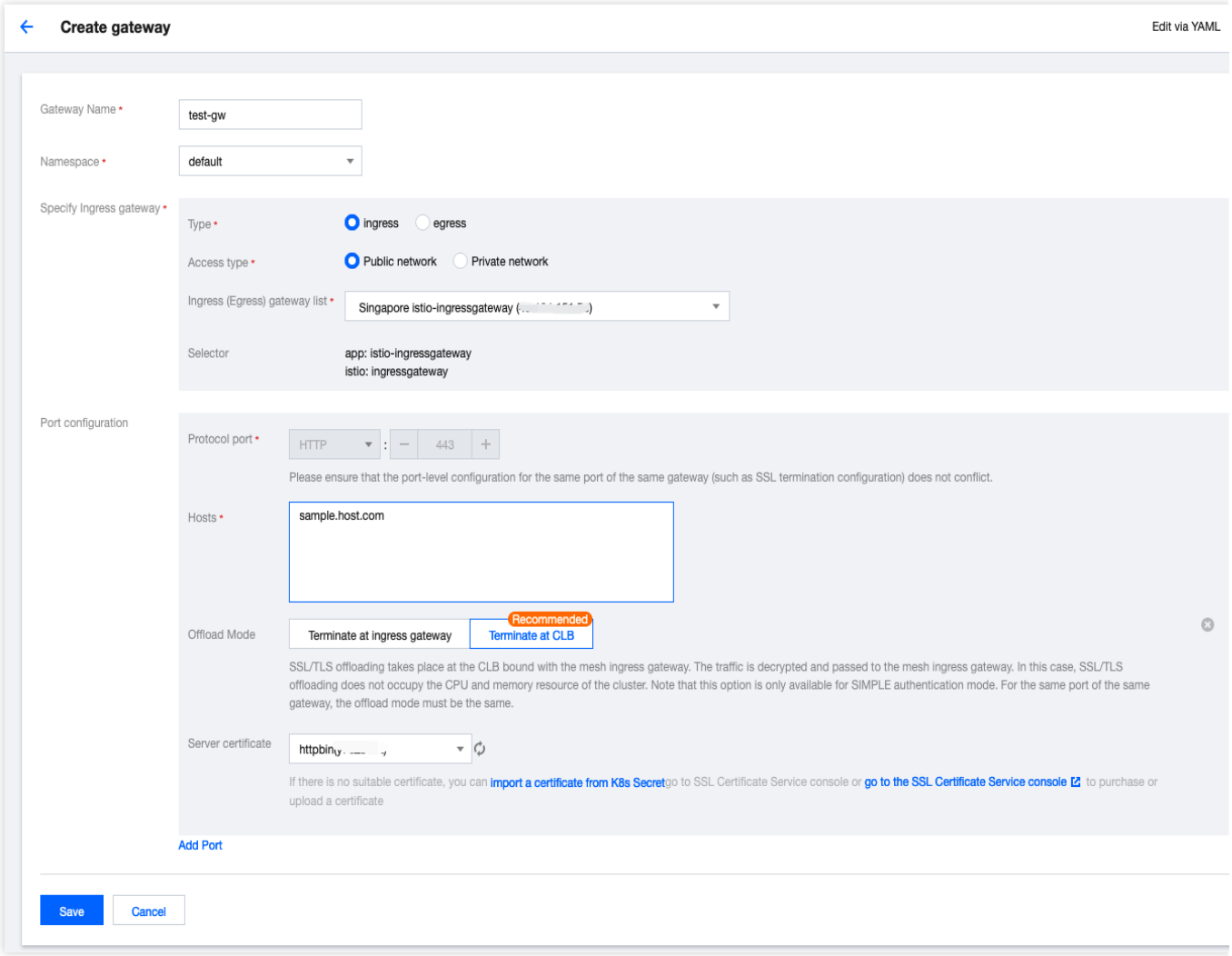
Console Configuration Example

In the following example, certificate offloading on port 443 is configured to take place at CLB, SNI is enabled for this port, the domain name `sample.hosta.org` uses certificate 1, and the domain name `sample.hostb.org` uses certificate 2.

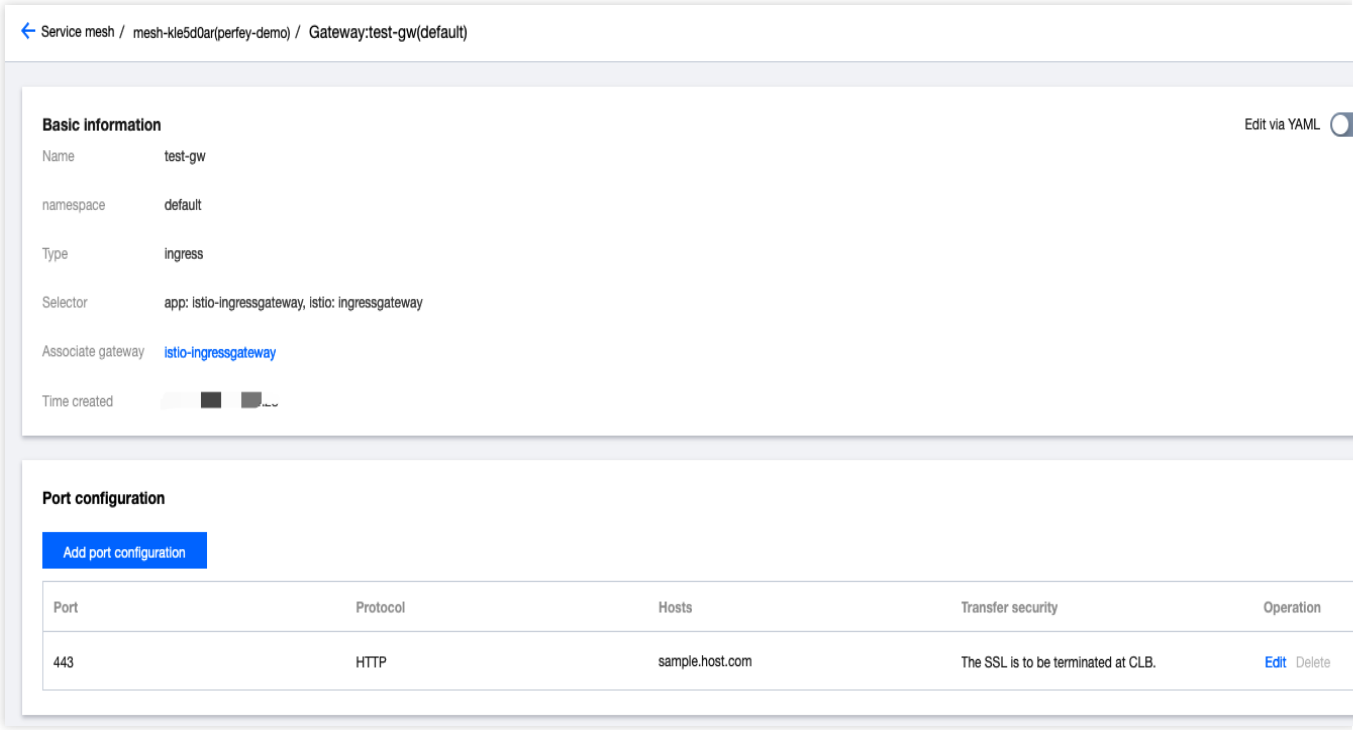
```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: test-gw
spec:
  servers:
    - port:
        number: 443
        name: clb-https-443-{Certificate ID 1}
        protocol: HTTP
      hosts:
        - sample.hosta.org
    - port:
        number: 443
        name: clb-https-443-{Certificate ID 2}
        protocol: HTTP
      hosts:
        - sample.hostb.org
  selector:
    app: istio-ingressgateway
    istio: ingressgateway
```

The process of creating gateway configurations by using UI in the console to implement the feature of enabling certificate offloading to take place at CLB is as follows:

1. Select protocol **HTTPS**. The **TLS authentication** parameter appears.
2. Select **SIMPLE** for **TLS authentication**.
3. Select **Terminate at CLB** for **Offload mode**. The port protocol is automatically changed to **HTTP** (if certificate offloading takes place at CLB, all traffic will be passed to the gateway in plaintext).
4. Select an appropriate server certificate.



After creation is successful, you are redirected to the details page of the created gateway CRD.

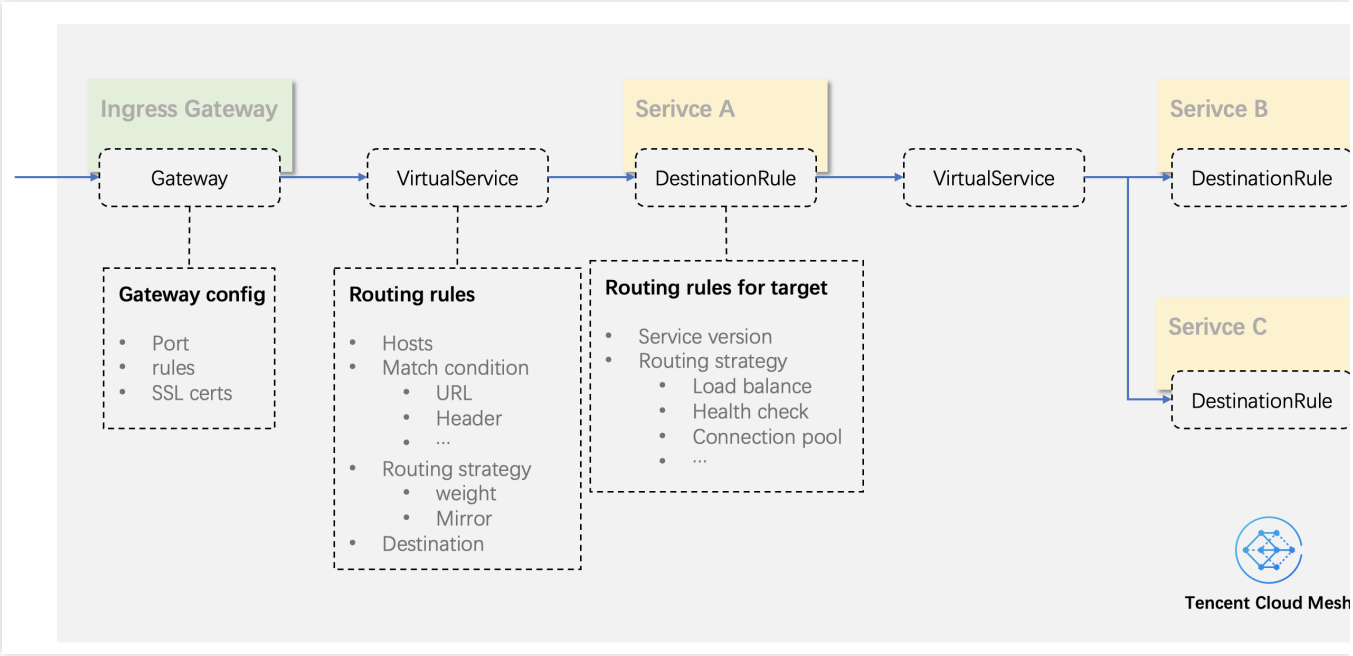

Service mesh / mesh-kle5d0ar(perfey-demo) / Gateway:test-gw(default)
Basic information
Edit via YAML
Name
test-gw
namespace
default
Type
ingress
Selector
app: istio-ingressgateway, istio: ingressgateway
Associate gateway
istio-ingressgateway
Time created
Port configuration
Add port configuration
Port
Protocol
Hosts
Transfer security
Operation
443
HTTP
sample.host.com
The SSL is to be terminated at CLB.
Edit
Delete

Traffic Management Overview

Last updated：2023-12-26 11:48:08

Traffic Management Model of Tencent Cloud Mesh

Tencent Cloud Mesh is fully compatible with Istio's native traffic management CRDs Gateway, VirtualService, and DestinationRule, and presents the native traffic management syntax as a product. The following figure shows the traffic management model of Tencent Cloud Mesh:



Tencent Cloud Mesh uses Gateway, VirtualService, and DestinationRule to manage traffic.
Gateway: defines the port, listening rule, and certificate configurations of a gateway. Gateways and gateway configurations are in a one-to-many relationship. The Gateway specifies a gateway to which the configurations are to be delivered through the selector field.
VirtualService: defines routing rules and traffic operation rules for a specified host. The VirtualService specifies a bound domain name through the hosts field. It can specify that traffic comes from a gateway or an internal component of a mesh.
DestinationRule: defines versions and traffic policies of a service. The traffic policies include load balancing, health check, and connection pools. Services and DestinationRules are in a one-to-one binding relationship.

Traffic Management Configuration Methods

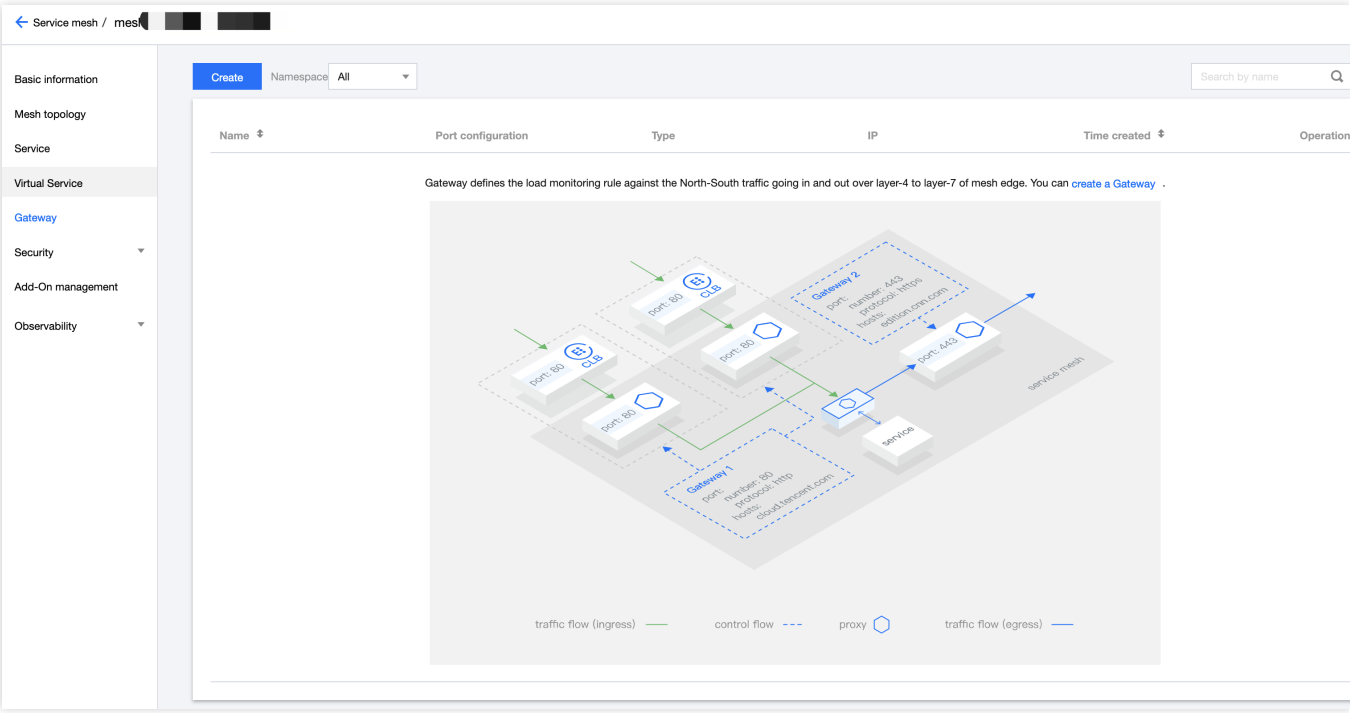
At present, Tencent Cloud Mesh provides the following two methods of configuring Gateways, VirtualServices, and DestinationRules：

Console UI Configuration

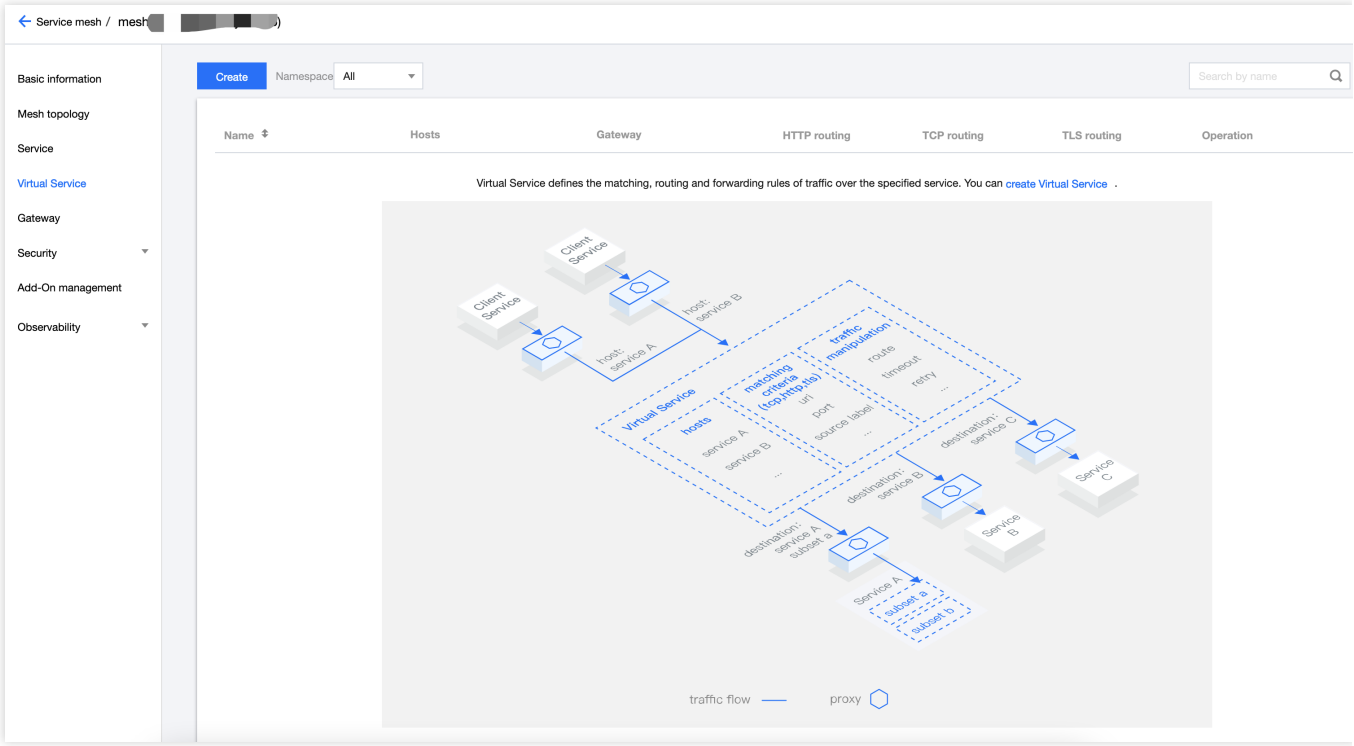
Resource Creation via YAML

You can use the console UI to create, delete, update, and view Gateways, VirtualServices, and DestinationRules.

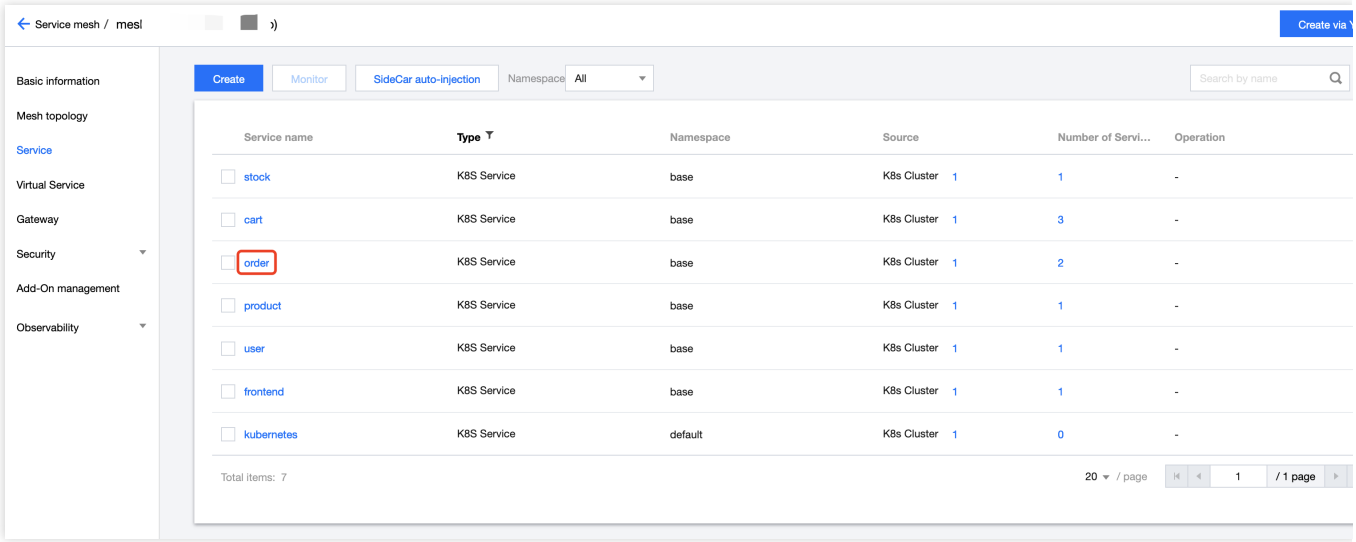
Creating a Gateway

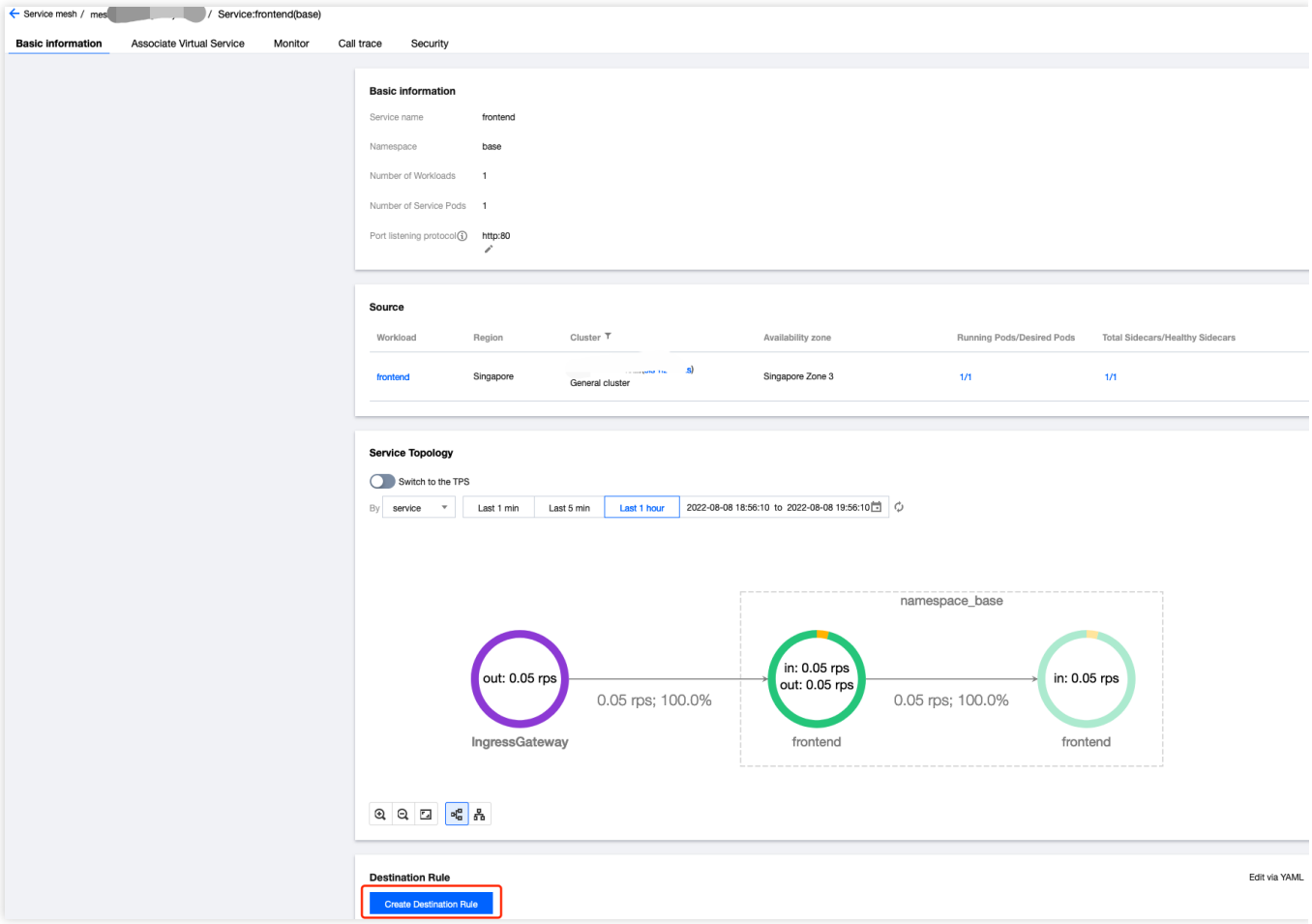


Creating a VirtualService

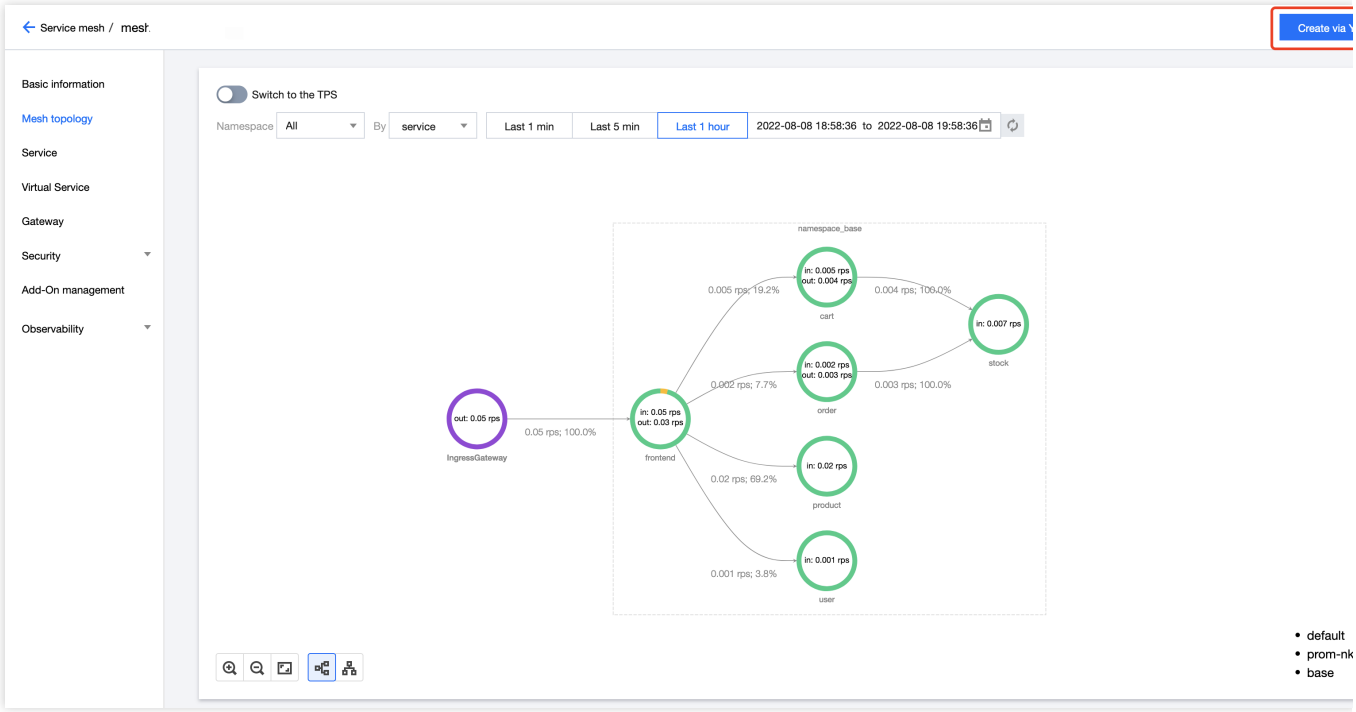


Creating a DestinationRule: As DestinationRules and services are in a one-to-one binding relationship, operations of creating and managing DestinationRules are performed on the service details page.





You can create Istio or Kubernetes resources by clicking **Create via YAML** in the upper right corner of the mesh management window. If the YAML to be submitted contains a Kubernetes resource and the current mesh manages multiple clusters, you need to select a destination cluster to which the YAML-created resource is submitted.


Service mesh / mesh
Create via
Basic information
Mesh topology
Service
Virtual Service
Gateway
Security
Add-On management
Observability
Switch to the TPS
Namespace
All
By
service
Last 1 min
Last 5 min
Last 1 hour
2022-08-08 18:58:36 to 2022-08-08 19:58:36
namespace_base
in: 0.005 rps
out: 0.004 rps
cart
0.005 rps; 19.2%
0.004 rps; 100.0%
in: 0.007 rps
stock
in: 0.002 rps
out: 0.003 rps
order
0.002 rps; 7.7%
0.003 rps; 100.0%
out: 0.05 rps
IngressGateway
0.05 rps; 100.0%
in: 0.05 rps
out: 0.03 rps
frontend
in: 0.02 rps
product
0.02 rps; 69.2%
in: 0.001 rps
user
0.001 rps; 3.8%
default
prom-nk
base

Using VirtualService to Configure Routing Rules

Last updated：2023-12-26 11:49:41

VirtualService defines a set of routing rules and traffic operations (such as weighted routing and fault injection) for a specified host. Each routing rule defines a matching rule for traffic of a specified protocol. If the traffic is matched, it is routed to a specified service or a version of the service. VirtualService configurations mainly include the following parts:

hosts: defines hosts associated with routing rules. The value can be a DNS name with a wildcard or an IP address.

gateways: defines the source of traffic to which routing rules are to be applied. The source can be:

One or more gateways

Sidecars in a mesh

Routing rules: defines detailed routing rules, including routing rules for three protocol types HTTP, TLS/HTTPS, and TCP.

http: defines an ordered list of routing rules for HTTP traffic.

tcp: defines an ordered list of routing rules for TCP traffic.

tls: defines an ordered list of routing rules for non-terminated TLS or HTTPS traffic.

Description of Major VirtualService Fields

Major VirtualService fields are described as follows.

Name	Type	Description
`spec.hosts`	`string[]`	A group of hosts associated with routir The value can be a DNS name with a v an IP address (IP addresses are allow traffic that comes from a gateway.). Th field applies to both HTTP and TCP tra Kubernetes environment, service short can be used. If a short name is used, I interpret the short name based on the namespace where the VirtualService l example, a rule in the default namespa containing a host `reviews` will be i as `reviews.default.svc.cluste` To avoid misconfigurations, it is recom use the full name of the host.

`spec.gateways`	`string[]`	Source of traffic to which routing rules a applied. The source can be one or mul gateways, or sidecars in a mesh. The v specified by `<gateway namespace>/<gateway name>` . T reserved word `mesh` is used to indic sidecars in the mesh. When this field is is set to `mesh` by default, indicating routing rules are applied to all sidecars mesh.
`spec.http`	`HTTPRoute[]`	An ordered list of routing rules for HTT (The first routing rule matching traffic is HTTP routing rules will be applied to tr mesh service ports named `http-` , `http2-` , or `grpc-` and traffic ov ports using protocol `HTTP` , `HTTP2` `GRPC` , or `TLS-Terminated-HTT`
`spec.http.match`	`HTTPMatchRequest[]`	A list of matching rules for a routing rul conditions in a single matching rule ha semantics, while the matching rules in have OR semantics.
`spec.http.route`	`HTTPRouteDestination[]`	A list of forwarding destinations of a ro An HTTP rule can either redirect or for (default) traffic. The forwarding destina be one or multiple services (service ve Behaviors such as configuring weights operations are allowed.
`spec.http.redirect`	`HTTPRedirect`	Route redirection. An HTTP rule can e redirect or forward (default) traffic. If th `passthrough` option is specified ir route and redirect will be ignored. The primitive can be used to send an HTTF redirect to a different URL or Authority.
`spec.http.rewrite`	`HTTPRewrite`	Rewrite HTTP URLs or Authority head Rewrite cannot be configured together redirect primitive. Rewrite will be perfo before forwarding.
`spec.http.timeout`	Duration	Timeout for HTTP requests.
`spec.http.retries`	`HTTPRetry`	Retry policy for HTTP requests.

`spec.http.fault`	`HTTPFaultInjection`	Fault injection policy to be applied on H traffic. Note that the timeout or retry po be enabled when fault injection is enab
`spec.http.mirror`	`Destination`	Mirror HTTP traffic to a another specifi destination. Mirrored traffic is on a "bes basis where the sidecar or gateway wi for a response to traffic mirroring befor the response from the original destinat Statistics will be generated for the mirr destination.
`spec.http.mirrorPercent`	`uint32`	Percentage of the traffic to be mirrored field is absent, all the traffic (100%) wil mirrored. The maximum value is 100.
`spec.http.corsPolicy`	`CorsPolicy`	Cross-Origin Resource Sharing (CORS For more details about CORS, see CO description about Istio CORS policy co syntax, see CorsPolicy.
`spec.http.headers`	`Headers`	Header operation rules, including upda adding, and deleting request and respo headers.
`spec.tcp`	`TCPRoute[]`	An ordered list of routing rules for TCP (The first routing rule matching traffic is TCP rules will be applied to any port th an HTTP or TLS port.
`spec.tcp.match`	`L4MatchAttributes[]`	A list of matching rules for a TCP routi conditions in a single matching rule ha semantics, while the matching rules in have OR semantics.
`spec.tcp.route`	`RouteDestination[]`	Destination to which the TCP connecti forwarded to.
`spec.tls`	`TLSRoute[]`	An ordered list of routing rules for non- TLS or HTTPS traffic (The first routing matching traffic is used.). TLS rules wil applied to traffic over mesh service por `https-` or `tls-`, traffic over unt gateway ports using `HTTPS` or `TLS` service entry ports using `HTTPS` or Note that traffic over `https-` or `t`

		without associated VirtualService will b as TCP traffic.
`spec.tls.match`	`TLSMatchAttributes[]`	A list of matching rules for a TLS routir conditions in a single matching rule ha semantics, while the matching rules in have OR semantics.
`spec.tls.route`	`RouteDestination[]`	Destination to which the connection is to.

Configuring Routing Rules for Traffic (South-North) from a Gateway

VirtualServices can be configured by using the console UI or YAML editing. The following shows VirtualService configurations for routing traffic from a gateway to the service frontend. The relevant gateway configurations are as follows:

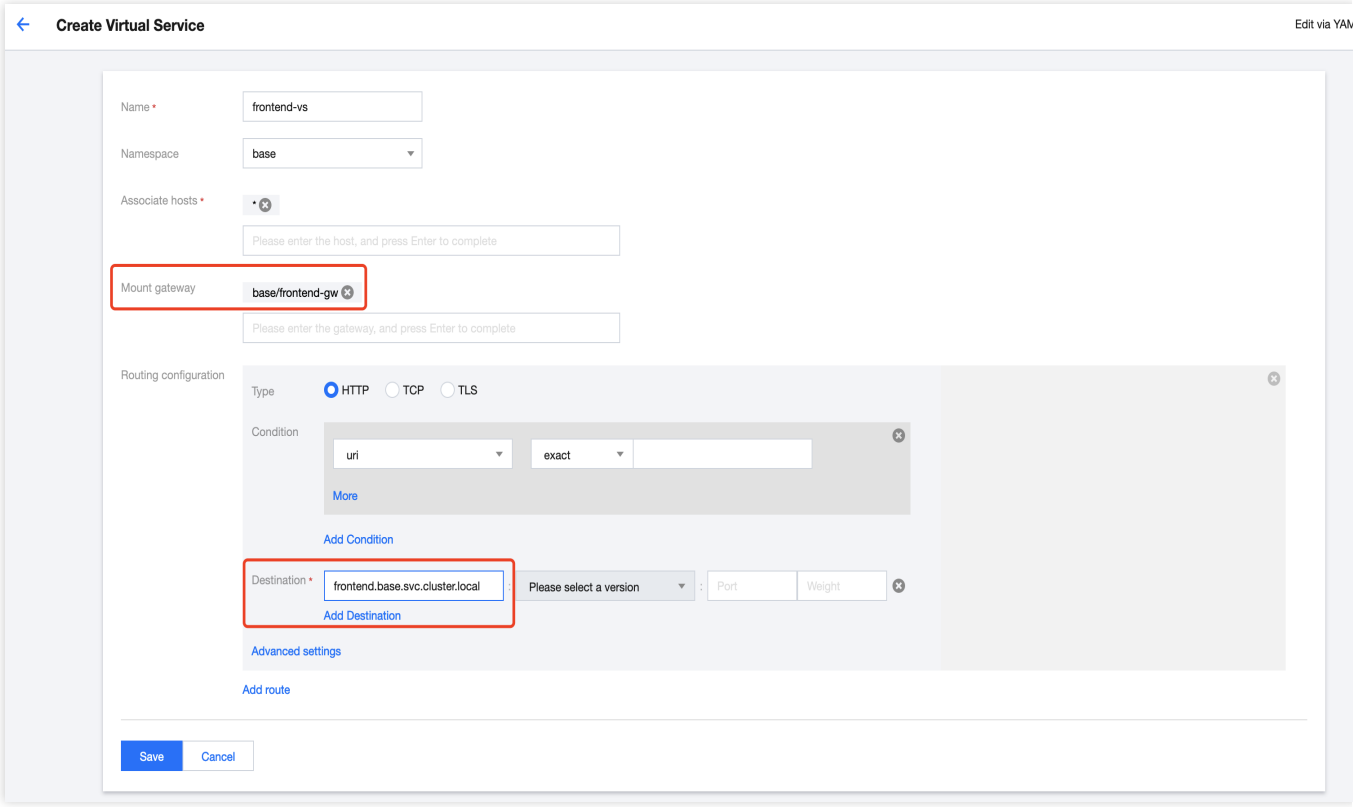
```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: frontend-gw
  namespace: base
spec:
  servers:
    - port:
        number: 80
        name: http
        protocol: HTTP
      hosts:
        - '*'
  selector:
    app: istio-ingressgateway
    istio: ingressgateway
```

YAML Configuration Example

Console Configuration Example

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: frontend-vs
  namespace: base
spec:
  hosts:
```

```
    - '*'
  gateways:
    - base/frontend-gw # Enter the gateway mounted to the VirtualService in the for
  http:
    - route:
        - destination:
            host: frontend.base.svc.cluster.local # Set the routing destination to
```



Configuring Routing Rules for Traffic (East-West) from a Mesh

The following shows VirtualService configurations about routing rules for internal mesh traffic of accessing the product service host: `product.base.svc.cluster.local` : 50% of the traffic is routed to v1 and 50% of the traffic is routed to v2 (a canary release). The service versions of product are defined by the following DestinationRule:

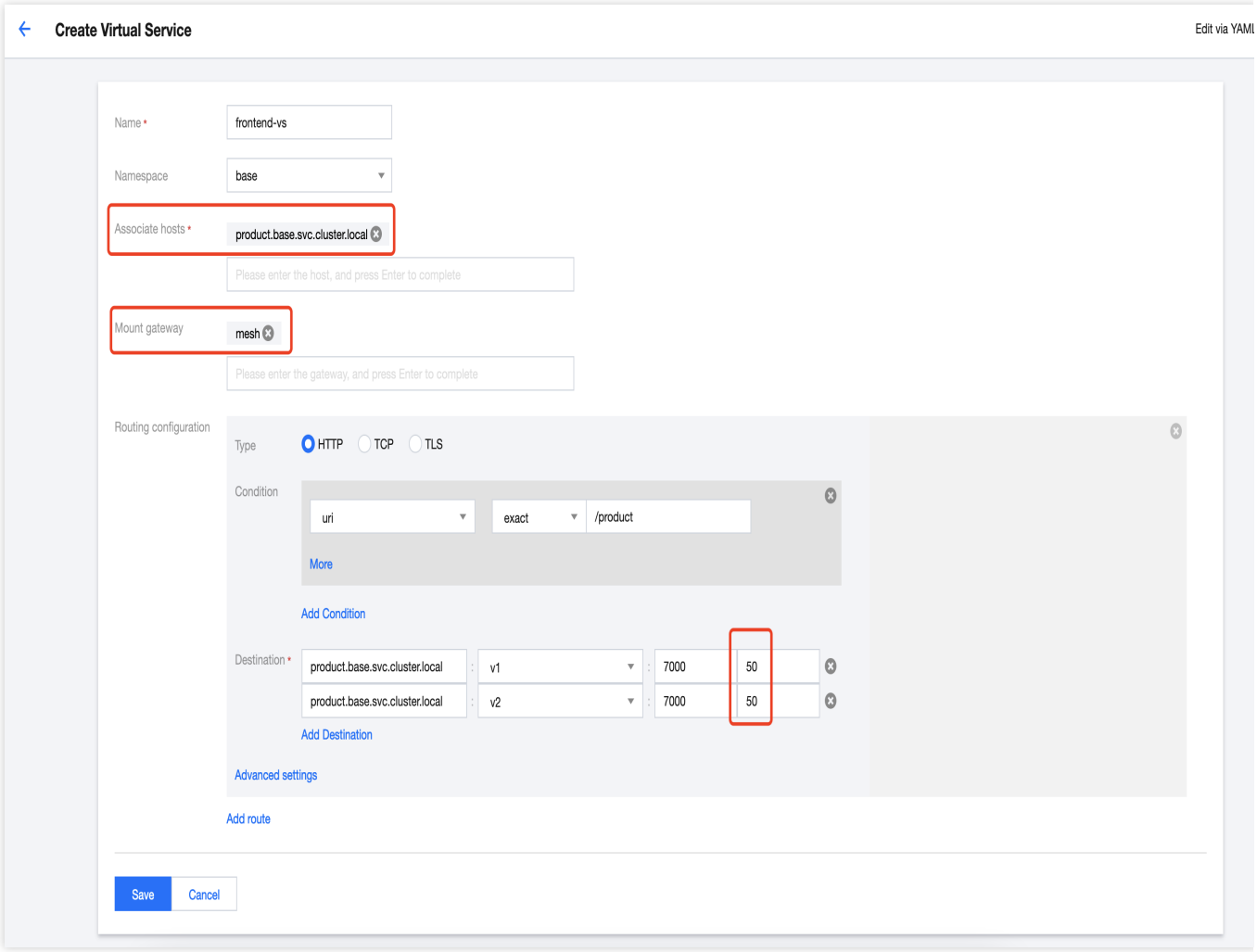
```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: product
  namespace: base
spec:
```

```
  host: product
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2
```

YAML Configuration Example

Console Configuration Example

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product-vs
  namespace: base
spec: # Default gateway parameters, indicating that the routing configurations are
  hosts:
    - "product.base.svc.cluster.local" # The traffic of accessing the host is match
  http:
    - match:
        - uri:
            exact: /product
      route:
        - destination: # Configure the destination and weight.
            host: product.base.svc.cluster.local
            subset: v1
            port:
              number: 7000
          weight: 50
        - destination:
            host: product.base.svc.cluster.local
            subset: v2
            port:
              number: 7000
          weight: 50
```


Create Virtual Service
Edit via YAML
Name *
frontend-vs
Namespace
base
Associate hosts *
product.base.svc.cluster.local
Please enter the host, and press Enter to complete
Mount gateway
mesh
Please enter the gateway, and press Enter to complete
Routing configuration
Type
HTTP
TCP
TLS
Condition
uri
exact
/product
More
Add Condition
Destination *
product.base.svc.cluster.local
v1
7000
50
product.base.svc.cluster.local
v2
7000
50
Add Destination
Advanced settings
Add route
Save
Cancel

Using DestinationRule to Configure Service Versions and Traffic Policies

Last updated：2023-12-26 11:50:46

DestinationRule defines versions of a service and traffic policies for the service after routing has occurred. These rules include load balancing, connection pool size, and health check (to detect and evict unhealthy hosts from the load balancing backend).

Description of Major DestinationRule Fields

Major DestinationRule fields are described as follows.

Name	Type	Description
`spec.host`	`string`	Name of a service associated with DestinationRule configurations. The service can be a service automatically discovered (for example, a Kubernetes service) or a host declared by ServiceEntry. Rules defined in the DestinationRule for the service that does not exist in the preceding source will be ignored.
`spec.subsets`	`Subset[]`	Versions (subnets) of a service. Versions can be matched against endpoints of the service by label key-value pairs. Traffic policies can be overridden at subset level.
`spec.trafficPolicy`	`trafficPolicy`	Traffic policies (load balancing, connection pools, health check, and TLS policy).
`spec.trafficPolicy.loadBalancer`	-	Load balancer algorithms. The following algorithms are available: simple load balancer algorithms (such as

		round robin, least conn, and random), consistent hashing (session persistence, and hashing based on header name, cookie, IP, and query parameters), and locality load balancing
`spec.trafficPolicy.connectionPool`	-	Volume of connections to an upstream service. A TCP or HTTP connection pool can be set.
`spec.trafficPolicy.outlierDetection`	-	Eviction of unhealthy hosts from the load balancing pool.
`spec.trafficPolicy.tls`	-	TLS-related configurations for the client connected to the upstream service. These configurations are used together with PeerAuthentication policies (TLS mode configurations for the server).
`spec.trafficPolicy.portLevelSettings`	-	Port-level traffic policies. Note that port-level policies will override the service-level or subset-level traffic policies.

Defining Service Versions (Subsets)

DestinationRule can define versions (subsets) of a service, and a subset is the smallest traffic management unit of Tencent Cloud Mesh. For example, you can configure traffic to be routed to a specified subset of a specified service. The following is a configuration example of using DestinationRule to define two subsets of the product service.

YAML Configuration Example

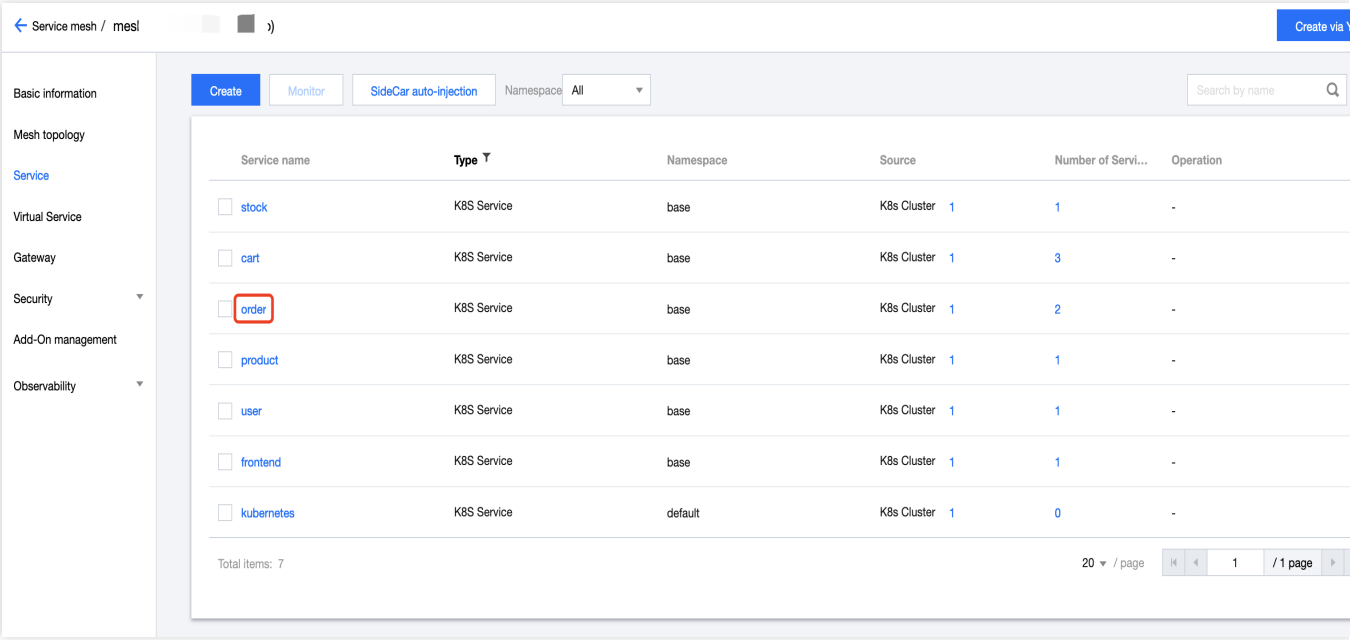
Console Configuration Example

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: product
  namespace: base
spec:
```

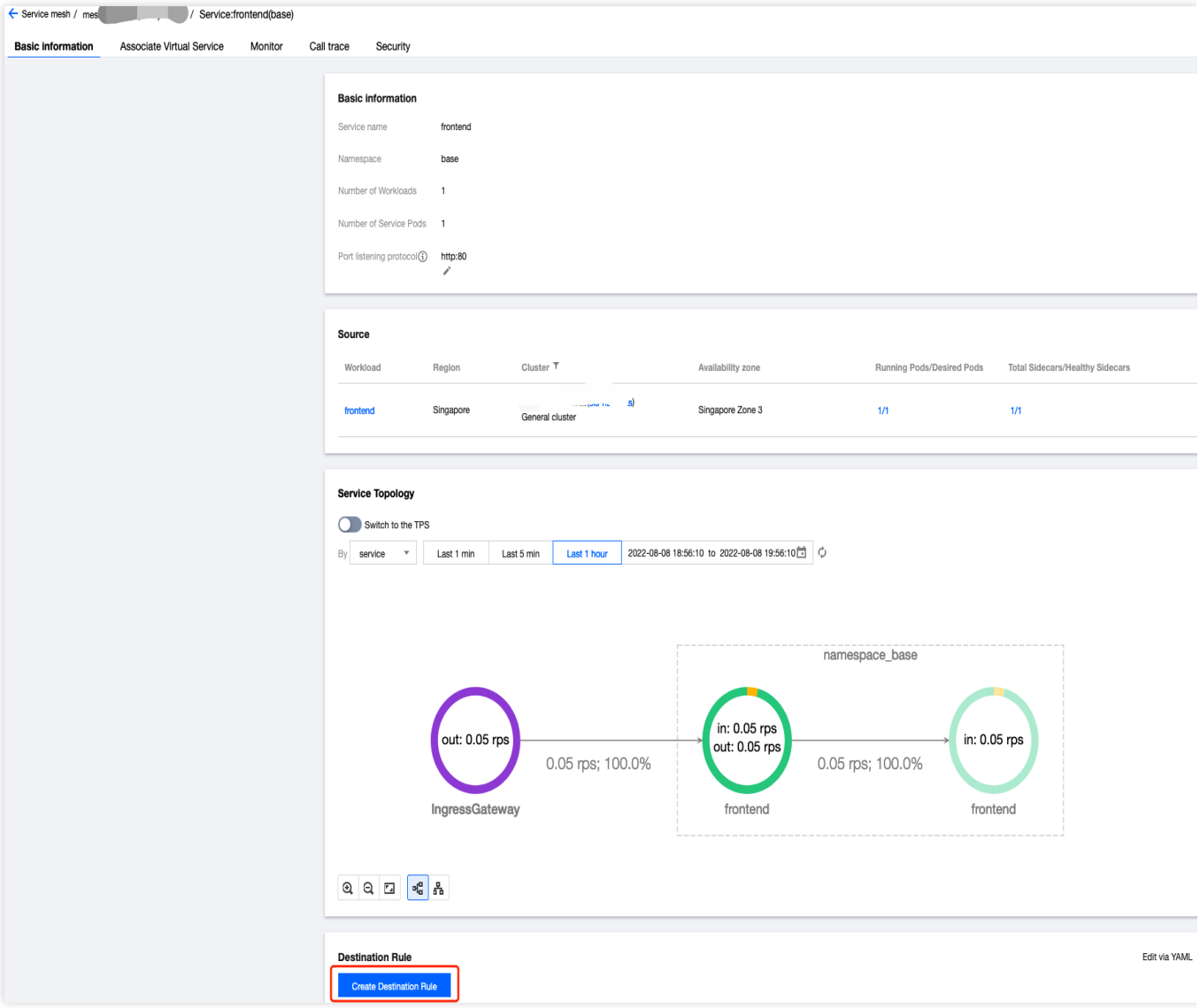
```
  host: product
  subsets:
    - name: v1
      labels:
        version: v1 # Subset v1 is matched against an endpoint of the service by us
    - name: v2
      labels:
        version: v2 # Subset v2 is matched against an endpoint of the service by us
```

DestinationRules and services are in a one-to-one binding relationship. To configure a DestinationRule of the product service, you need to enter the product service details page from the service list page, and configure the DestinationRule on the **Basic information** tab page. The steps to configure two versions of the product service on the console are as follows:

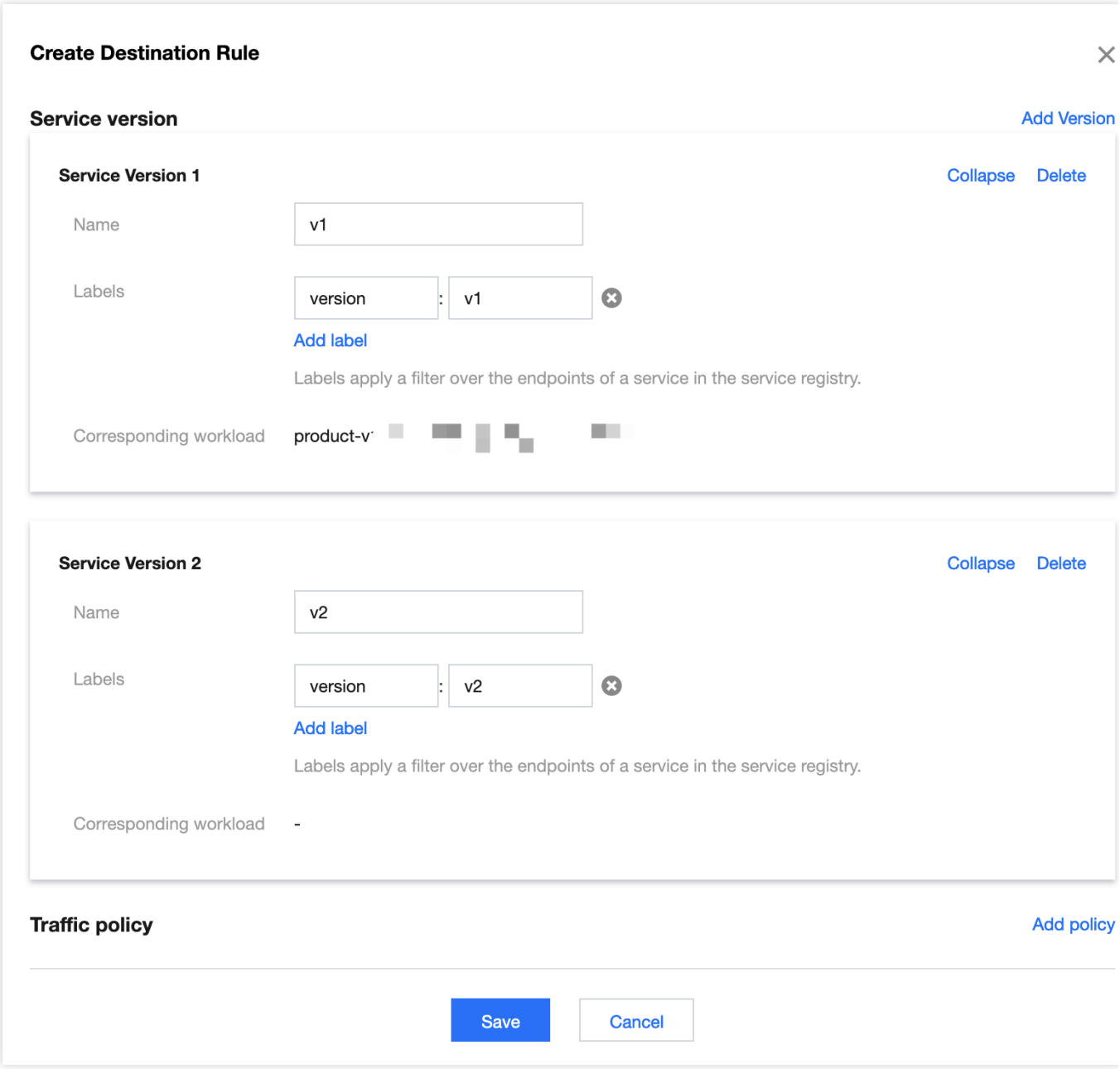
1. On the service list page, click **product** to enter the product service details page.



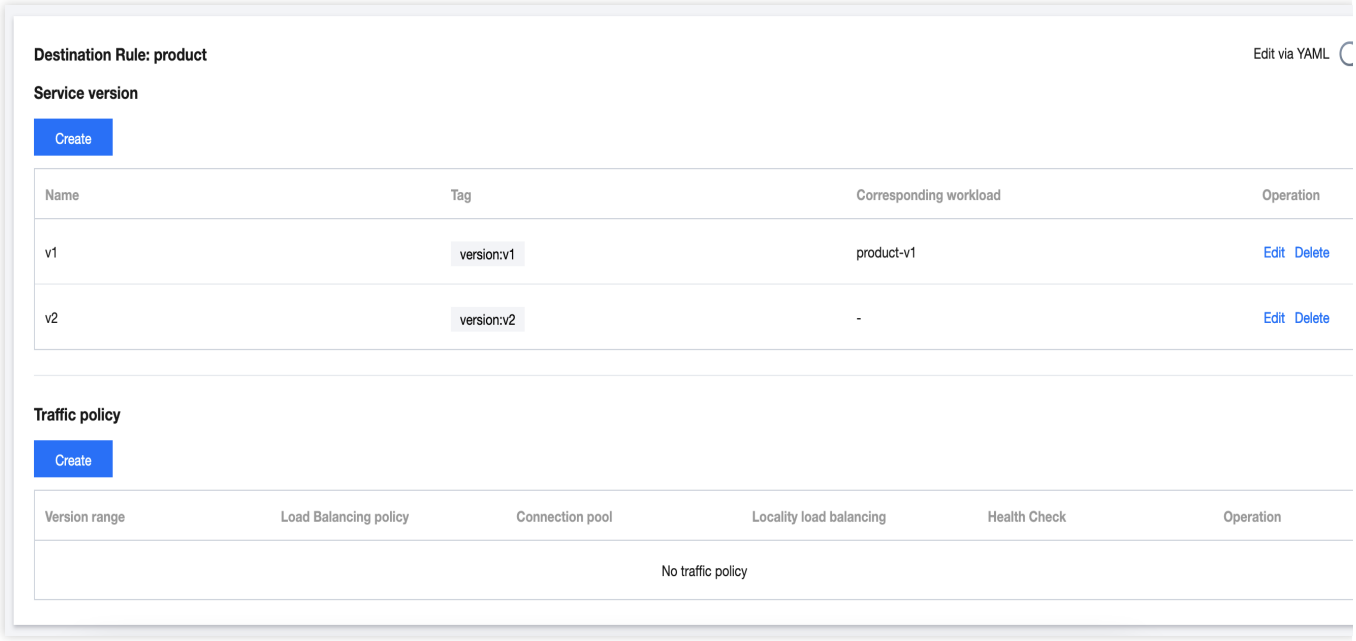
2. In the third card area **DestinationRule** on the **Basic information** tab page of the service details page, click **Create DestinationRule** to enter the creation pop-up window.



3. In the pop-up window, add two versions for the product service and click **Save**.



4. Version configuration of the product service is complete.



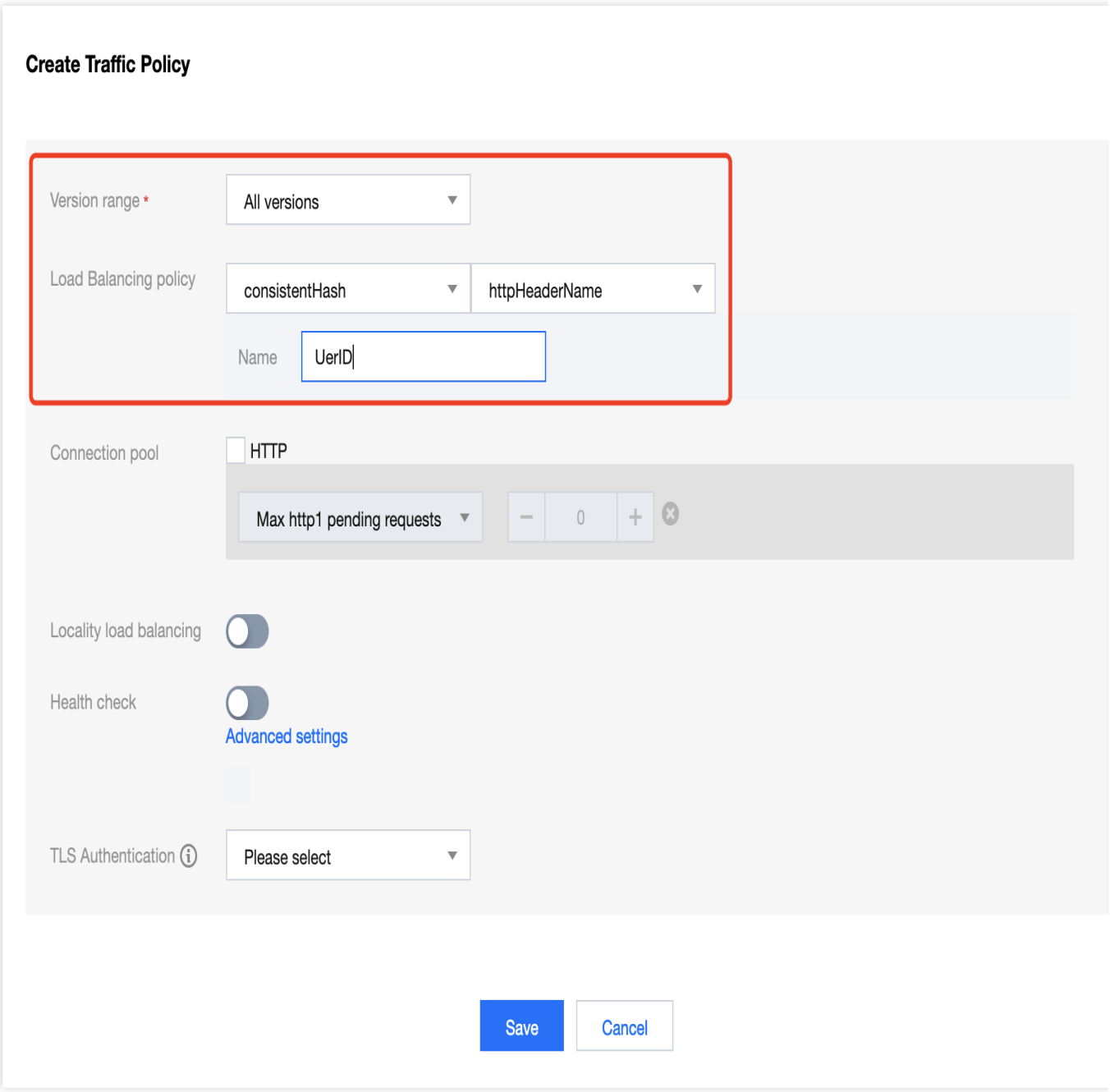
Configuring Consistent Hash-based Load Balancing

The following is a configuration example of using DestinationRule to configure the cart service to perform consistent hash-based load balancing based on the HTTP header name.

YAML Configuration Example

Console Configuration Example

```
kind: DestinationRule
metadata:
  name: cart
  namespace: base
spec:
  host: cart
  trafficPolicy:
    loadBalancer:
      consistentHash:
        httpHeaderName: UserID # Configure hash-based load balancing to be performe
```

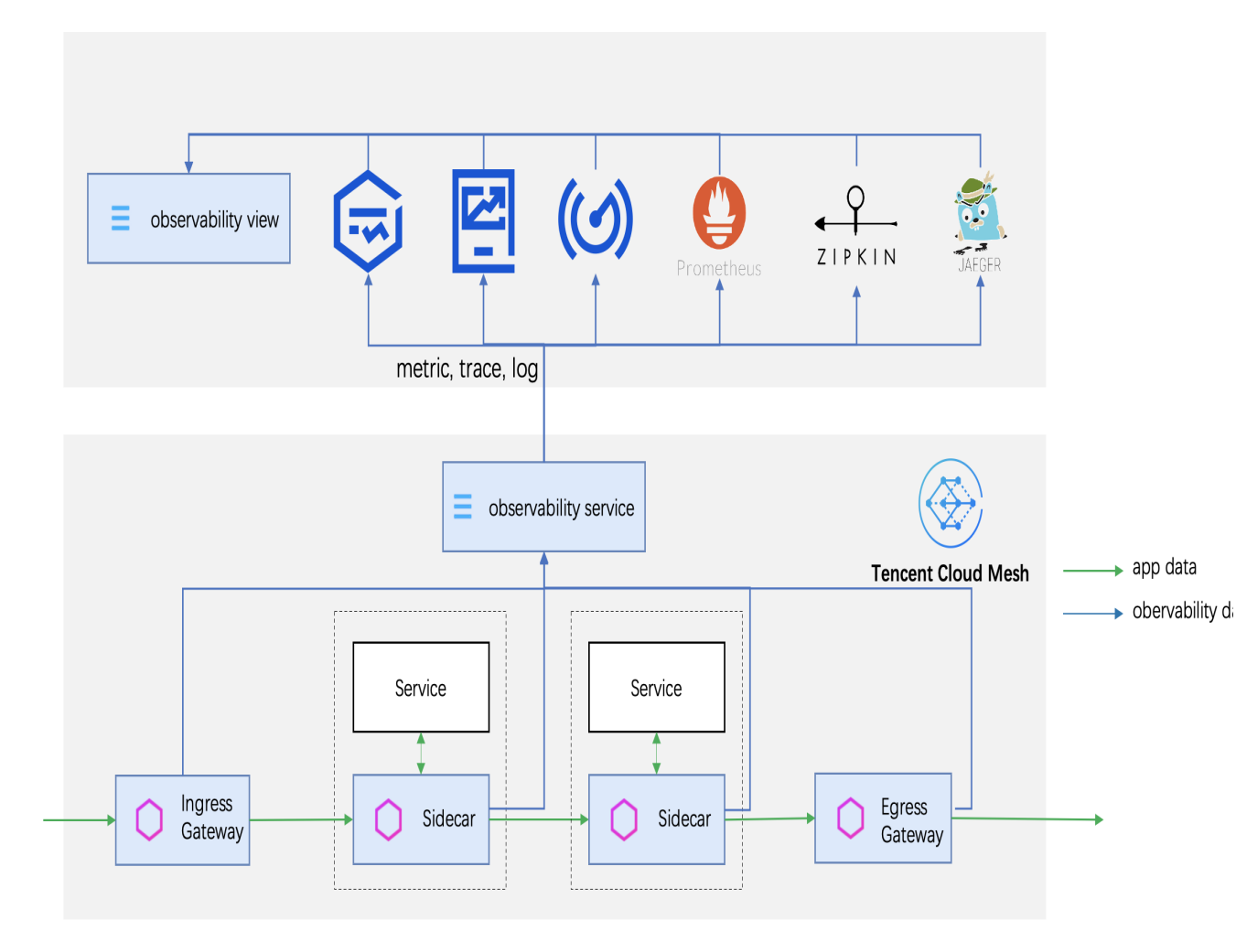

Create Traffic Policy
Version range *
All versions
Load Balancing policy
consistentHash
httpHeaderName
Name
UerID
Connection pool
HTTP
Max http1 pending requests
0
Locality load balancing
Health check
Advanced settings
TLS Authentication
Please select
Save
Cancel

Observability

Overview

Last updated：2023-12-26 11:51:04

Tencent Cloud Mesh provides end-to-end observability between north-south and east-west services.
The collection of observation data depends on reporting of the envoy sidecar proxy (data plane) of a service that has been injected with sidecars. You can flexibly control the production and calculation of observable data on the data plane through Tencent Cloud Mesh. Tencent Cloud Mesh integrates the observation data into suitable monitoring products to provide you with the observability of traffic between services at the edge of a mesh and services inside the mesh.



Tencent Cloud Mesh provides three types of observable data:

Type	Description

Metric	Metrics provide you with traffic observation data of services or gateways, and are suitable for developers of a single service to focus on.
Trace	Call tracing can link multi-layer calls of a service request into a call trace, which is convenient for you to observe the call structure, perform performance analysis, and locate exceptions.
Access log	Access logs completely record each request generation by the Envoy proxy, including information about the request layer and the sidecar proxy layer, which is convenient for operations personnel to conduct access auditing and fault troubleshooting.

The three types of observable data are described as follows:

Observable Data	Recorded Information	Applicable Scenario or Role
Metric	Traffic observation data of a single service or gateway, including but not limited to metrics such as latency, number of requests, and request size. For more metric information, see Istio Standard Metrics.	Developers of a single service monitor the operating status of the service.
Trace	Call dependencies between services. Compared with metric information, trace information further includes URL information. The recorded data is generally sampled.	Overall service developers perform call dependencies and performance analysis of all services.
Access log	Complete information about each request, including rich information output at the sidecar proxy layer. For more information, see Envoy Access Logging.	Mesh operations personnel conduct access auditing and fault troubleshooting.

Monitoring Metrics

Last updated：2023-12-26 11:51:21

Currently, Tencent Cloud Mesh can choose to use Managed Service for Prometheus (TMP) to provide you with the collection, storage, and display of service traffic metric data.

Note:

Tencent Cloud Mesh will support the use of third-party Prometheus services as monitoring backend services in the near future.

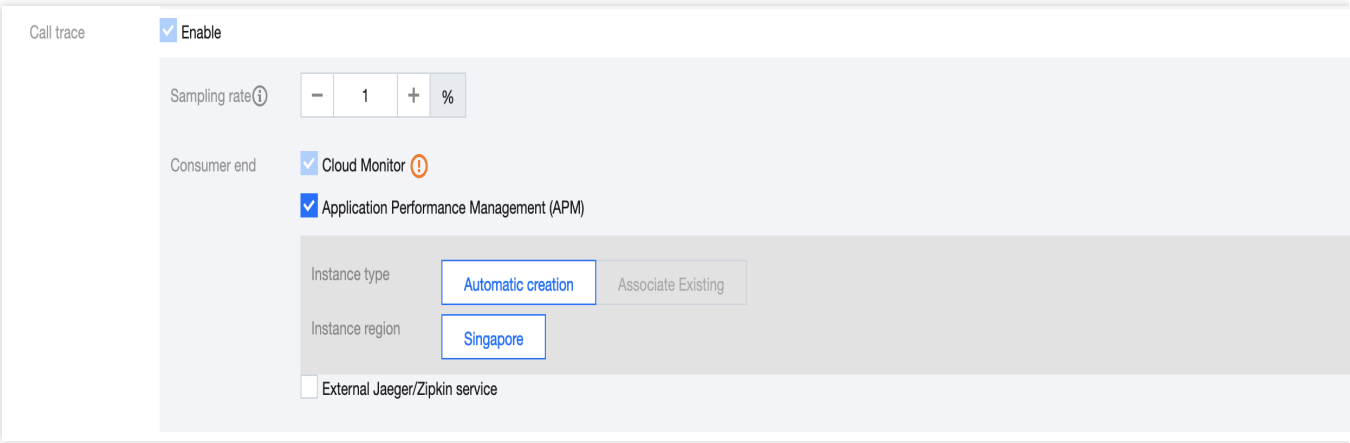
Monitoring charts on the Tencent Cloud Mesh console will be displayed based on the monitoring metrics stored in TMP. If you have custom monitoring requirements, you can set a custom monitoring dashboard through the Grafana dashboard in TMP.

Directions

Based on the metric data reported by sidecars to TMP, the Tencent Cloud Mesh console provides display and analysis of mesh topology, service topology, and service monitoring (number of requests, request status code distribution, request duration, and request size) charts.

Enabling TMP Monitoring

On the **Create mesh** page or the **Basic information** page of the mesh, find **Observability configuration** > **Monitoring metrics**, select **TMP**, and select **Automatic creation** or **Associate existing** for **TMP instance** as needed. After TMP monitoring is enabled, sidecars will report metric data to the corresponding instance, and you can view the instance on the TMP console.

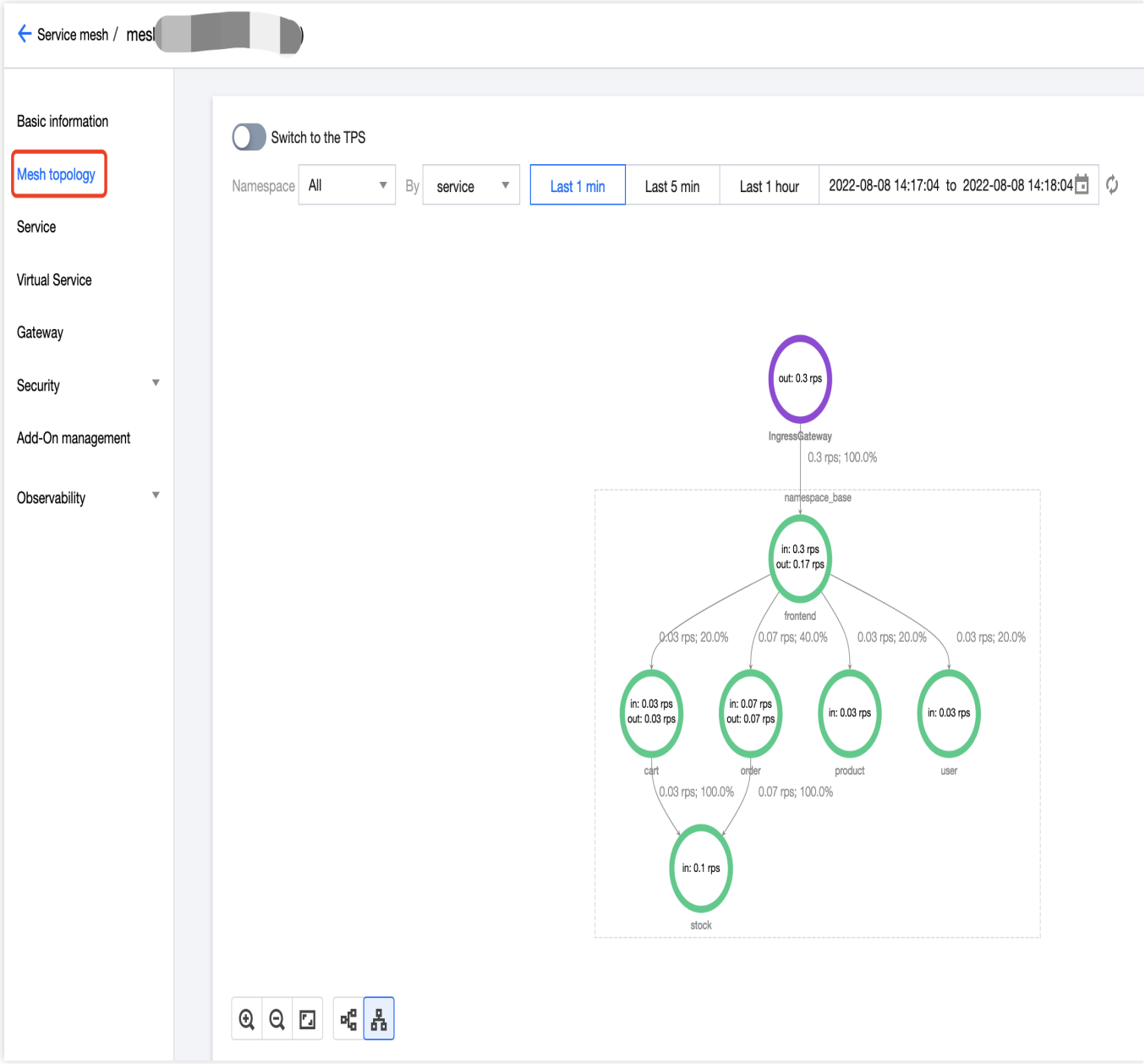


Viewing Monitoring Charts

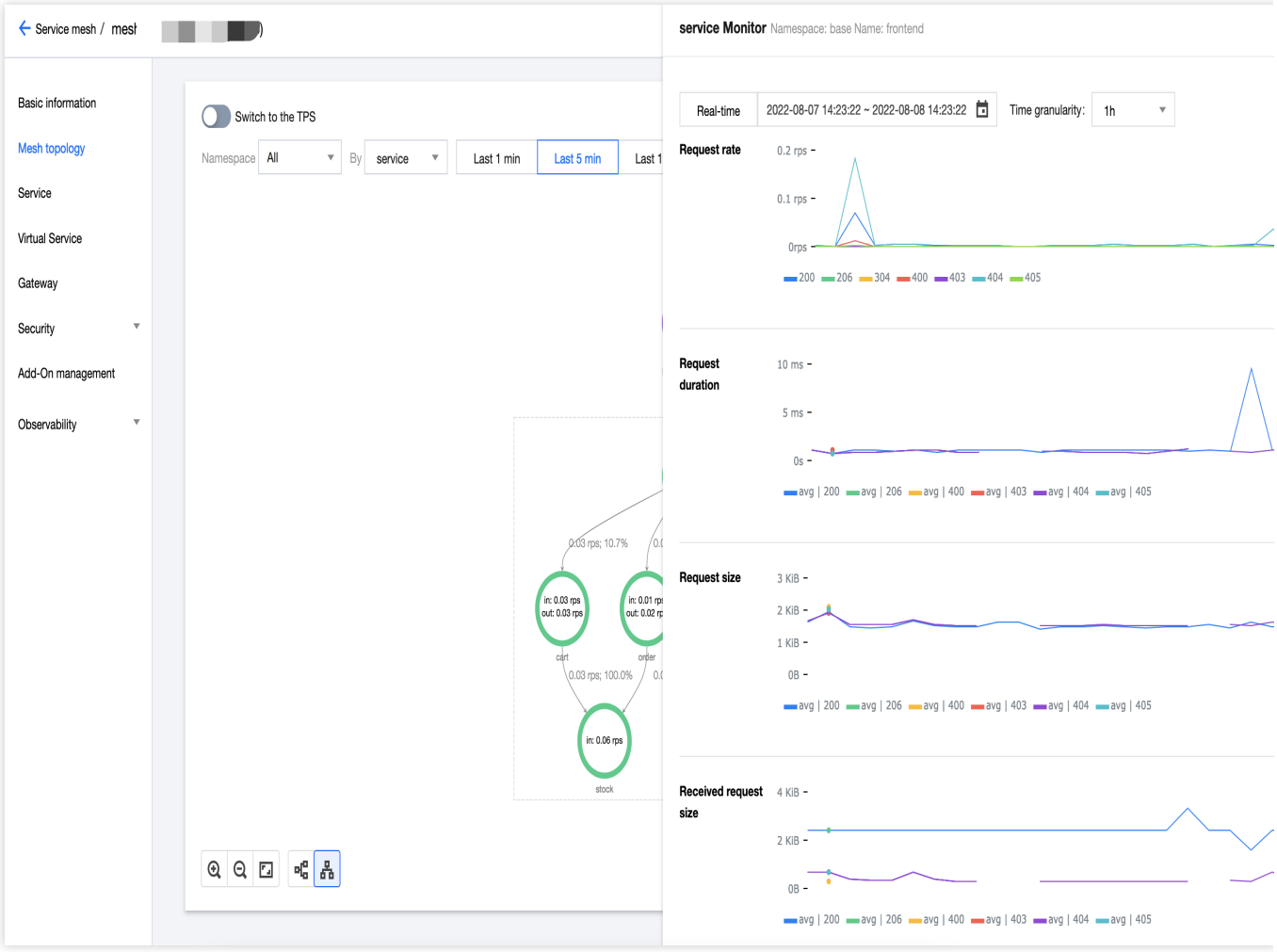
Mesh topology

A mesh topology records call structures of all services in a service mesh. Before viewing the mesh topology, ensure that sidecars have been injected into related services and that there is request traffic. The procedure of viewing the mesh topology of a specified mesh is as follows:

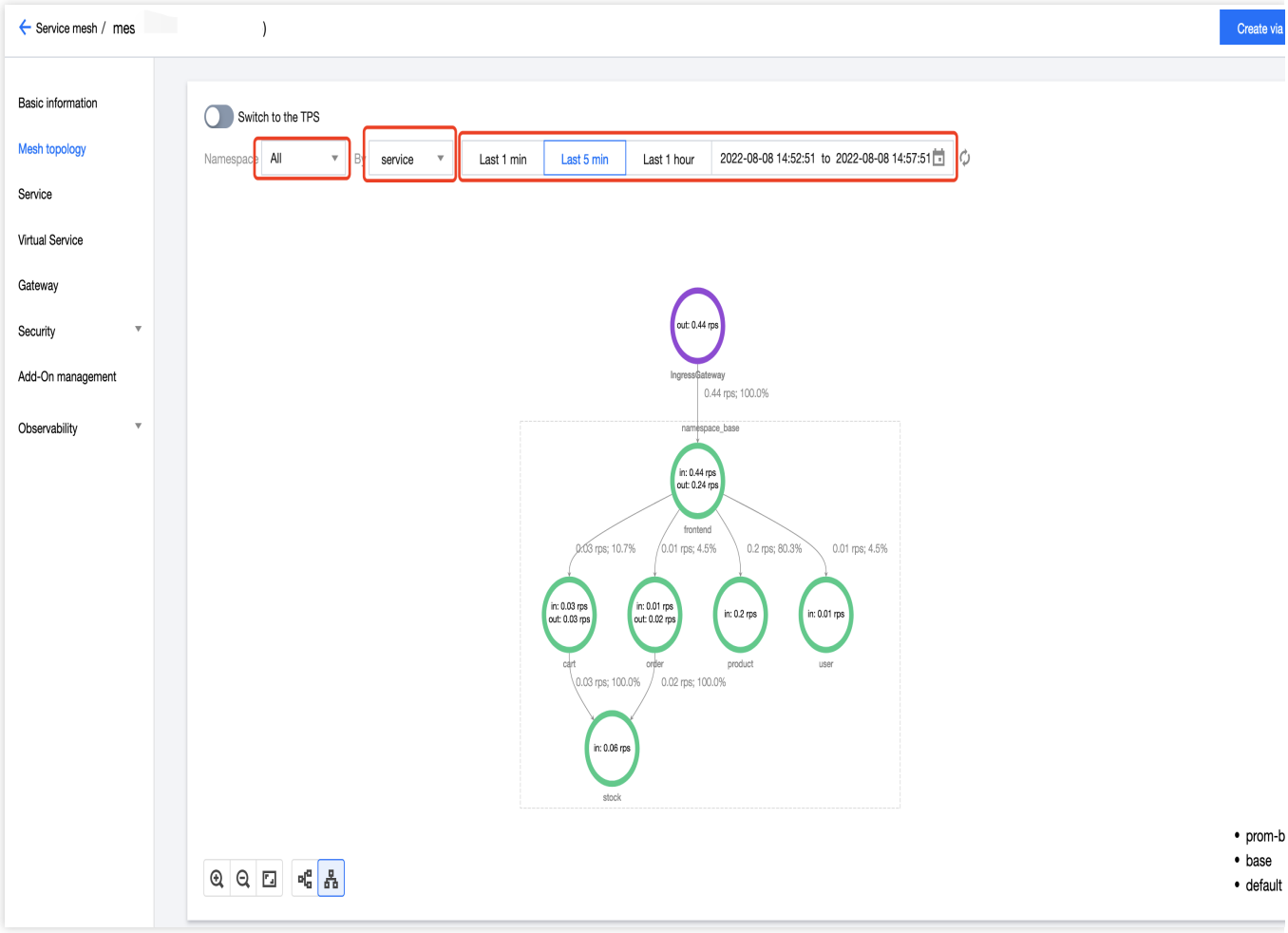
1. Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh details page.
2. Click **Mesh topology** in the left sidebar, and view the mesh topology of the specified mesh.



3. Click a node to display monitoring details related to the node.



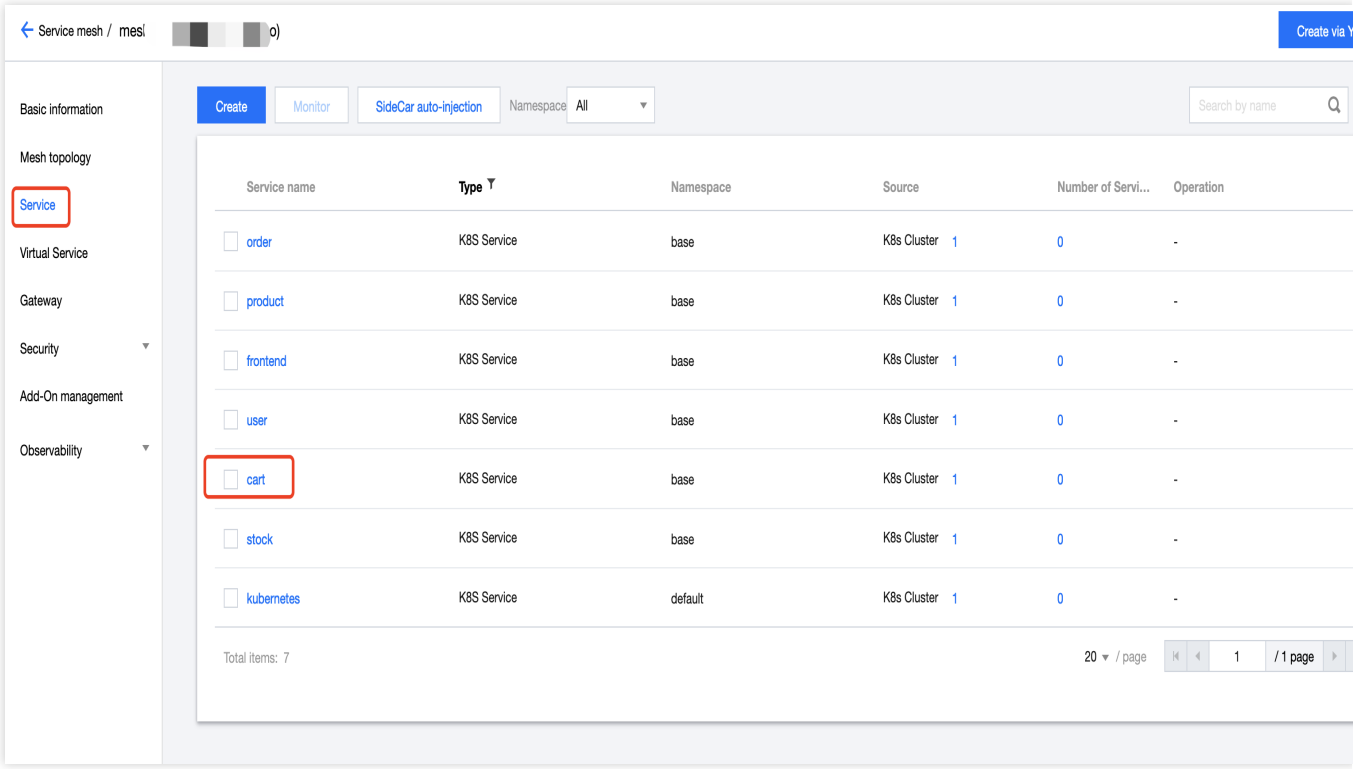
4. At the top of the page, select data filtering conditions (including namespace and time span) and granularity of nodes (service granularity and workload granularity are supported currently).



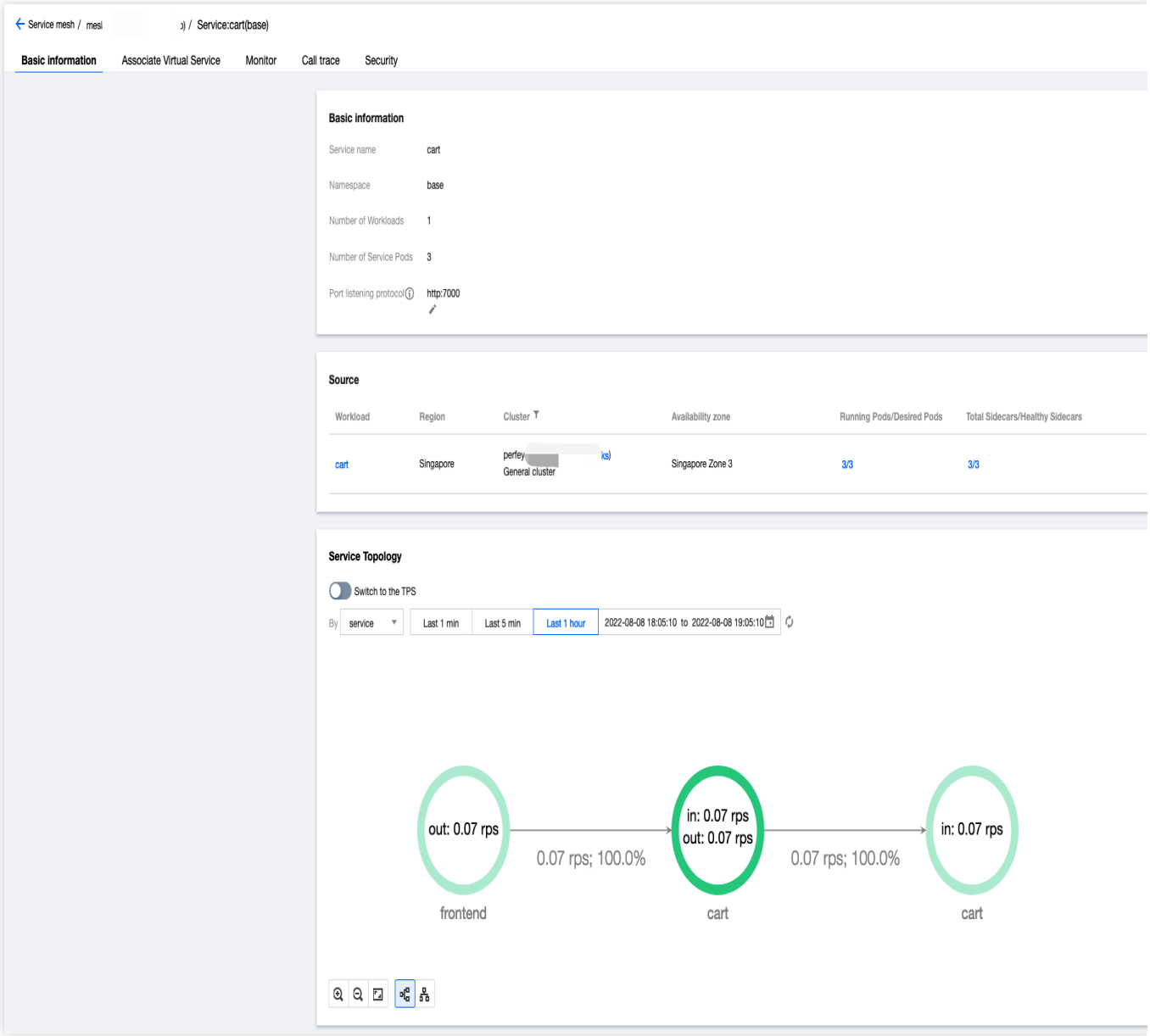
Service topology

A service topology records dependencies between previous and next calls of a service. The procedure of viewing the service topology of a specified service is as follows:

1. On the details page of the specified mesh, click **Service** in the left sidebar to enter the service list page.
2. Click the service to be viewed to enter the service details page.



3. On the **Basic information** tab page of the service details page, view the service topology of the service.



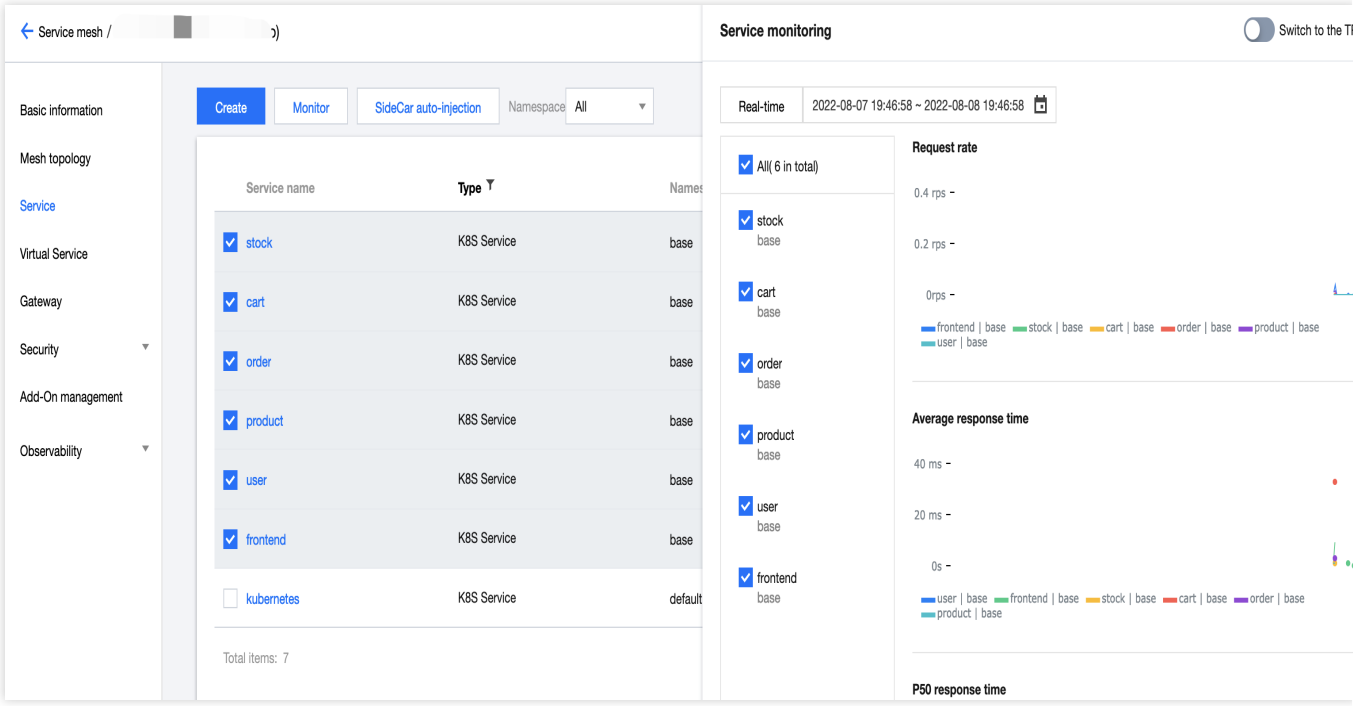
Service monitoring

You can compare the monitoring data (such as the number of requests, request duration, request size) of multiple services on the service list page, or view the monitoring details of a specified service on the service details page.

Viewing the monitoring data of multiple services on the service list page

1.1 Log in to the Tencent Cloud Mesh console, and click a specified mesh ID in the list to enter the mesh details page.

1.2 Choose **Service** > **Monitor**, click the service whose monitoring data is to be viewed, and view the service monitoring data on the right.

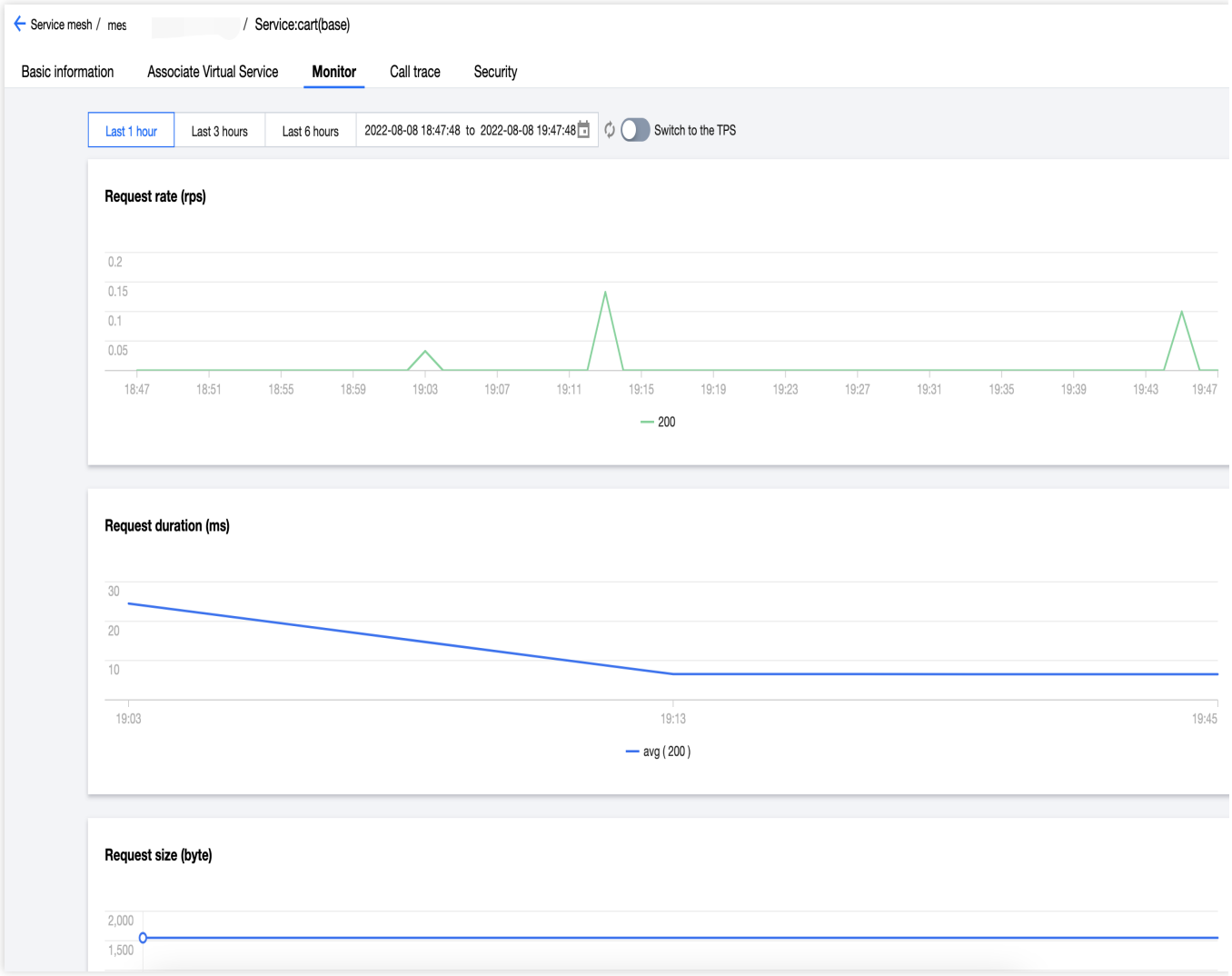


Viewing the detailed monitoring data charts of a specified service on the service details page

1.1 On the details page of the specified mesh, click **Service** in the left sidebar to enter the service list page.

1.2 Click the service to be viewed to enter the service details page.

1.3 View the charts on the **Monitor** tab page of the service details page.



Disabling monitoring

You can choose to edit the observability configuration on the **Basic information** page of the mesh, and deselect **TMP**. After deselection, the TMP instance will not be deleted on the Tencent Cloud Mesh side. If necessary, go to the TMP console to delete the TMP instance.

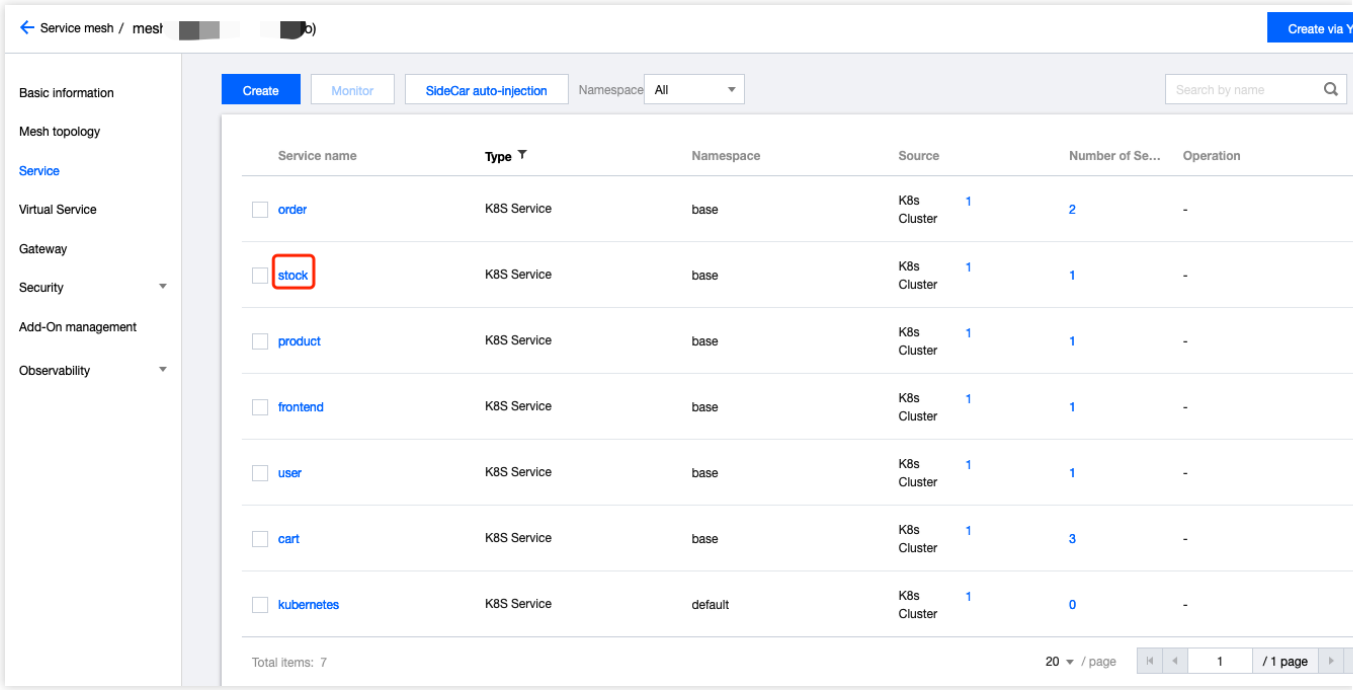
Call Traces

Last updated：2023-12-26 14:16:37

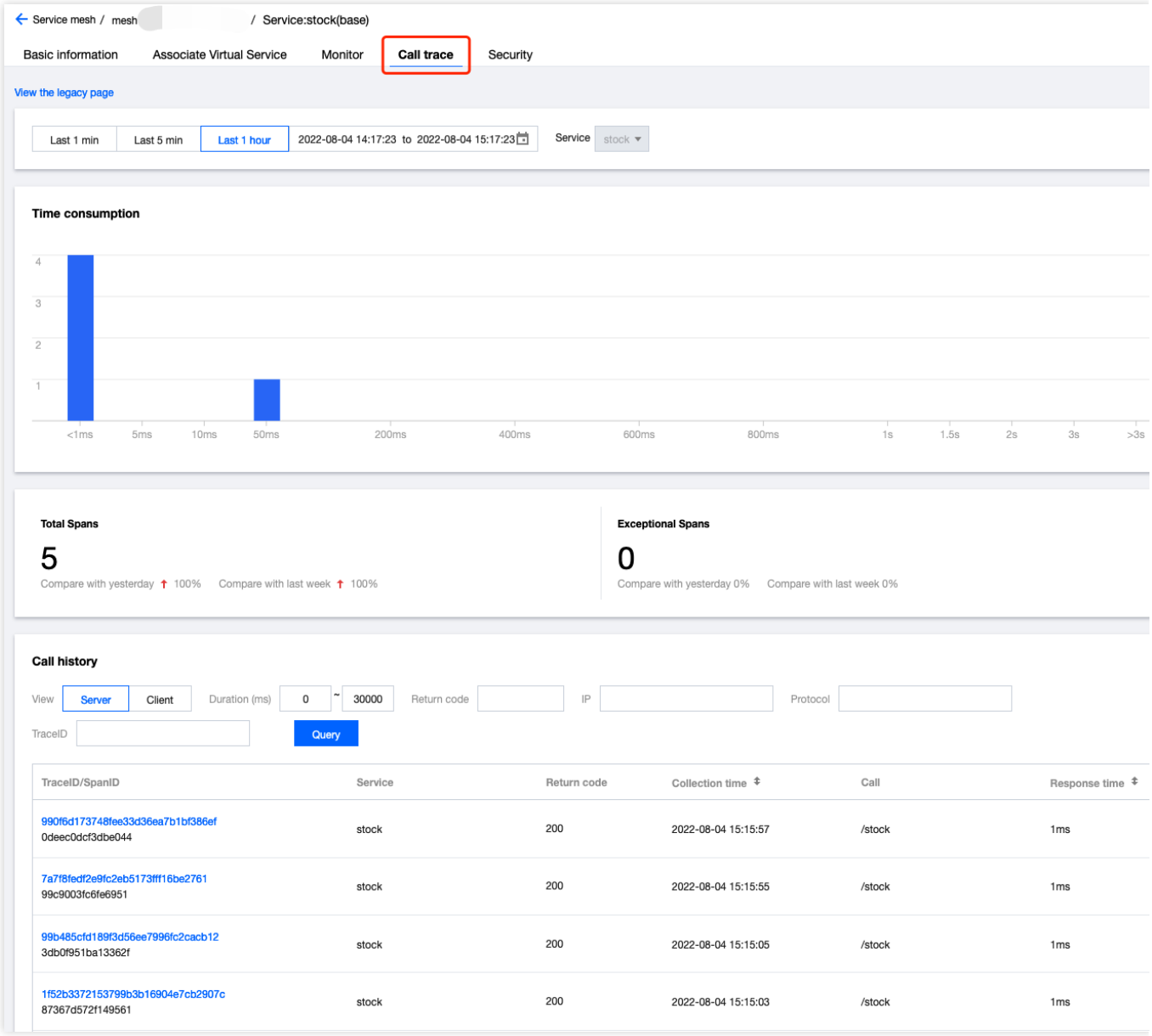
By default, Tencent Cloud Mesh integrates Application Performance Management (APM) as the consumer end for call tracing. After the consumer end is enabled, Tencent Cloud Mesh will create an APM instance for you and report tracing data to the corresponding APM instance. On the Tencent Cloud Mesh console, you can view a complete call waterfall chart of a request in the mesh and tracing log information about calls at each layer, which can help you understand call dependencies of services and conduct latency analysis in the mesh. You can also view call data directly on the APM console.
In addition to APM, the mesh supports reporting the call data to the third-party Jaeger/Zpkin service. If the third-party tracing service is enabled, the Tencent Cloud Mesh console cannot display call tracing information, which needs to be viewed in the third-party service.
Call tracing data is collected and reported by sidecars, and the sidecars automatically generate trace spans. If you need to view the complete call trace information, you need to make few modifications on the service code to deliver the request context, so that Tencent Cloud Mesh can correctly associate the inbound and outbound spans to form a complete call trace. The headers that need to be delivered by the service include:
x-request-id
x-b3-traceid
x-b3-spanid
x-b3-parentspanid
x-b3-sampled
x-b3-flags
x-ot-span-context
For more information about Envoy-based tracing, see Istio Distributed Tracing FAQ.

Viewing Call Tracing

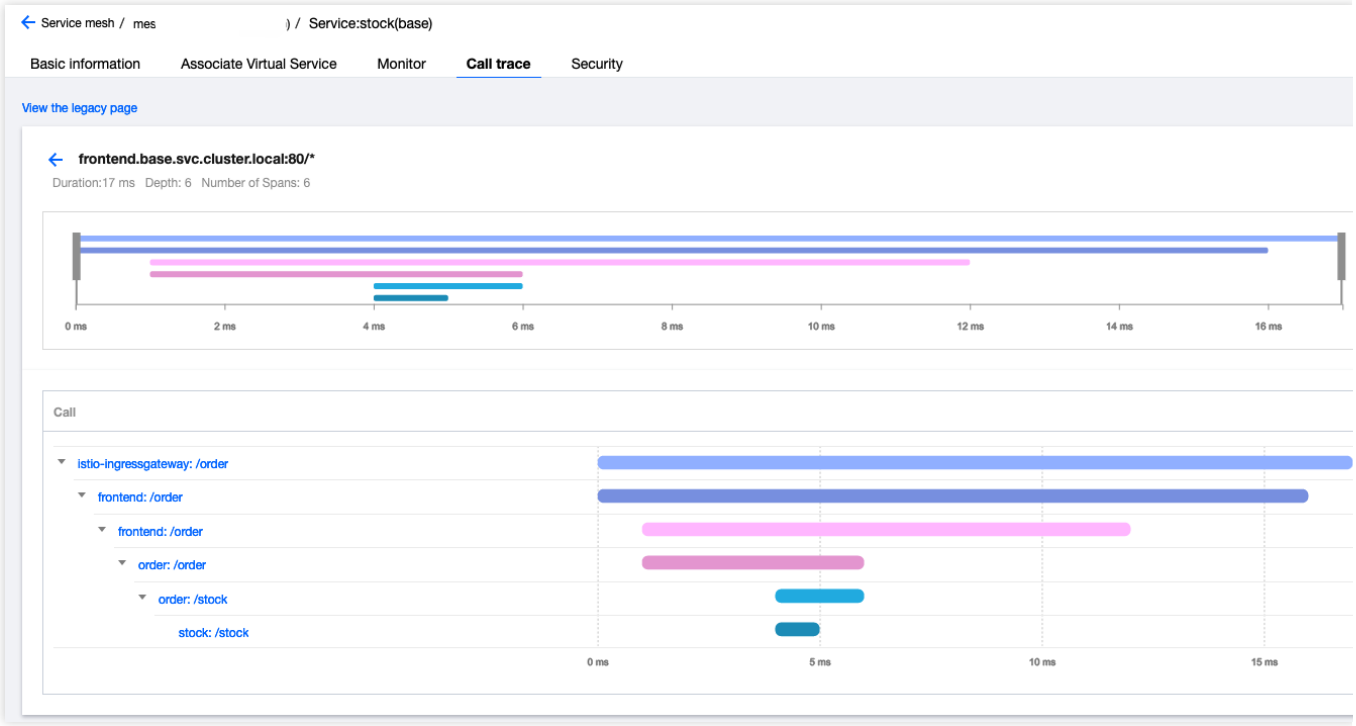
The procedure for viewing call tracing is as follows:
1. In the service list of the mesh, click the service whose call information needs to be focused on to enter the service details page.

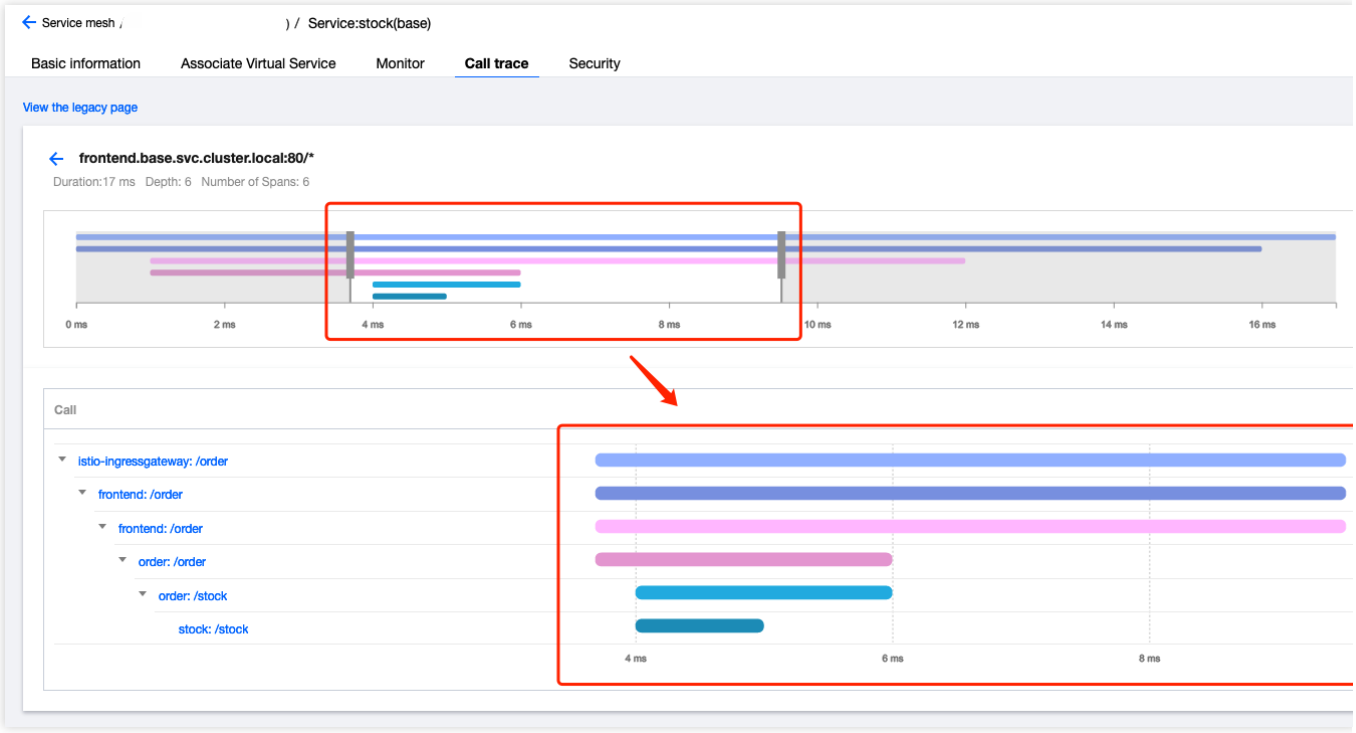


2. On the service details page, click **Call trace**. You can view that the service is a callee, and view a list of called records and a statistical histogram of duration distribution of these records.

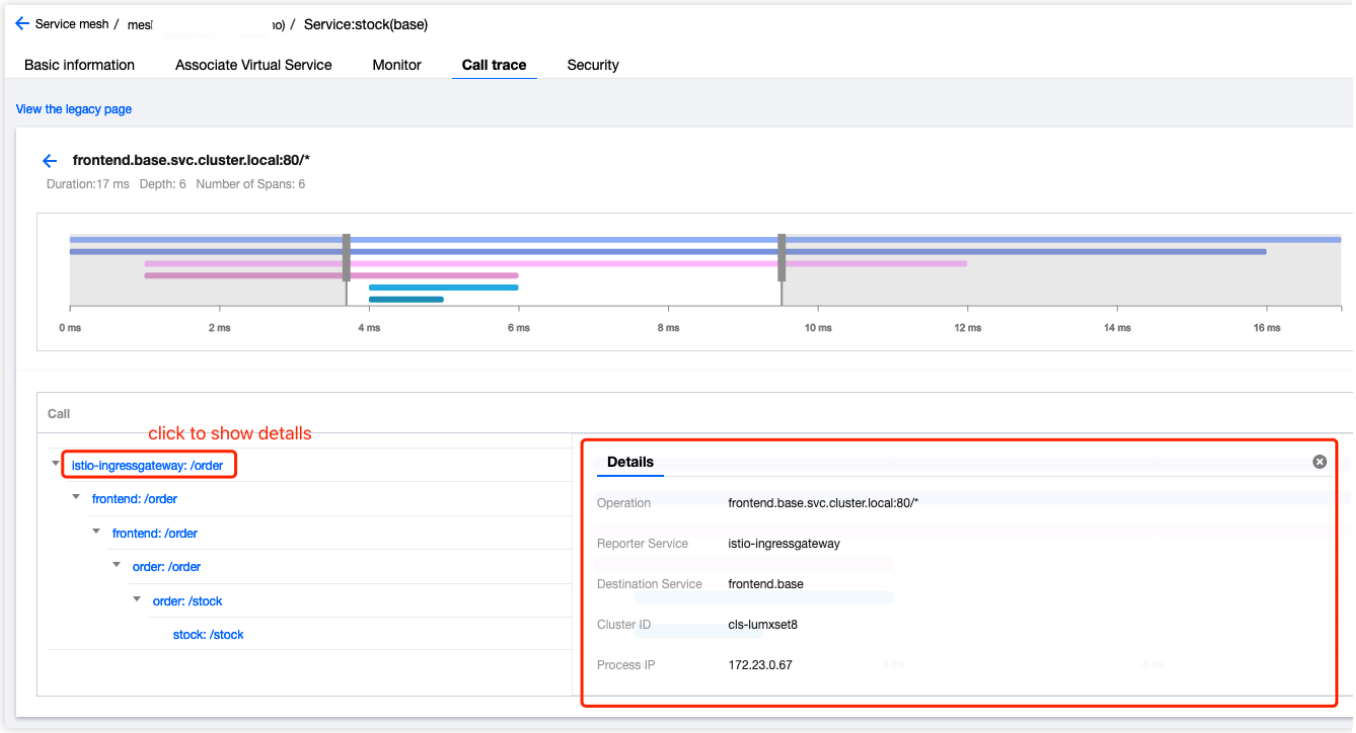


3. Click the first column of the called record list to view a complete call trace waterfall chart related to the call. The first column records the URL of the call. The overview of the waterfall chart above can be zoomed through dragging.

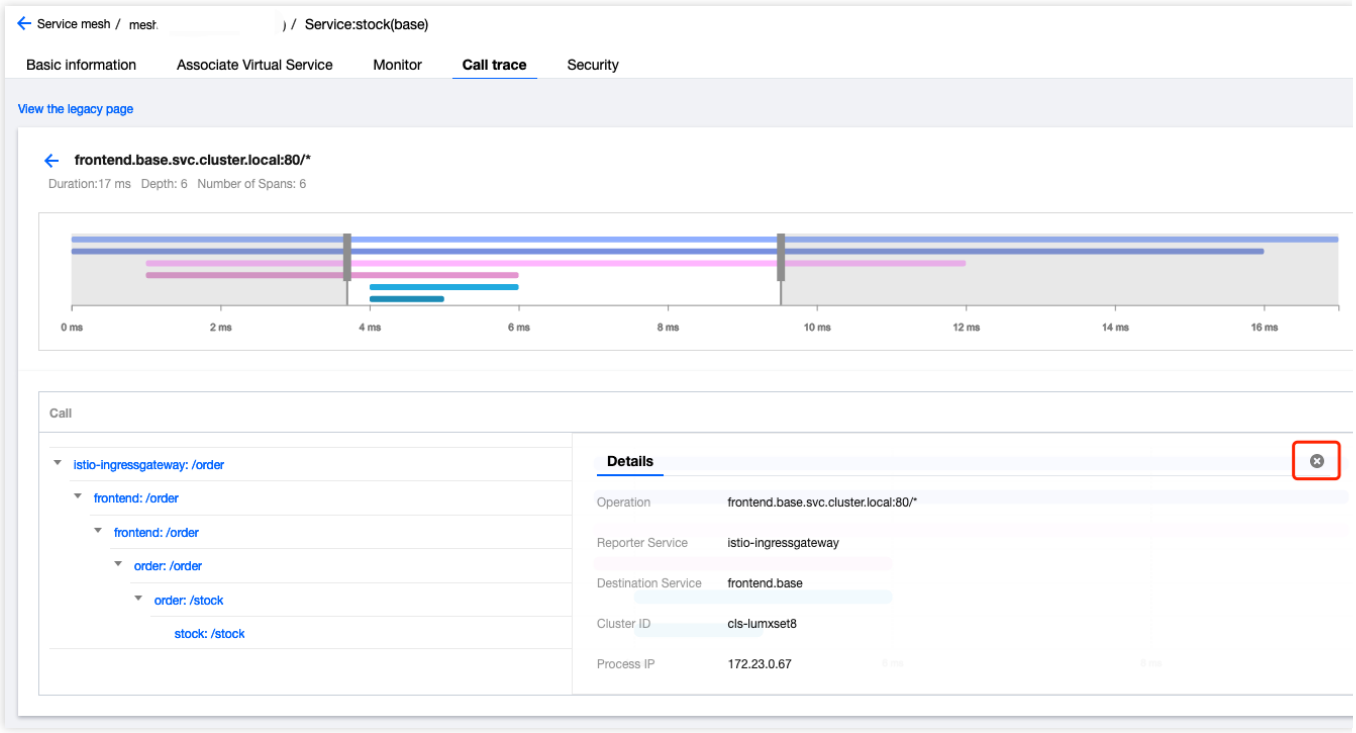




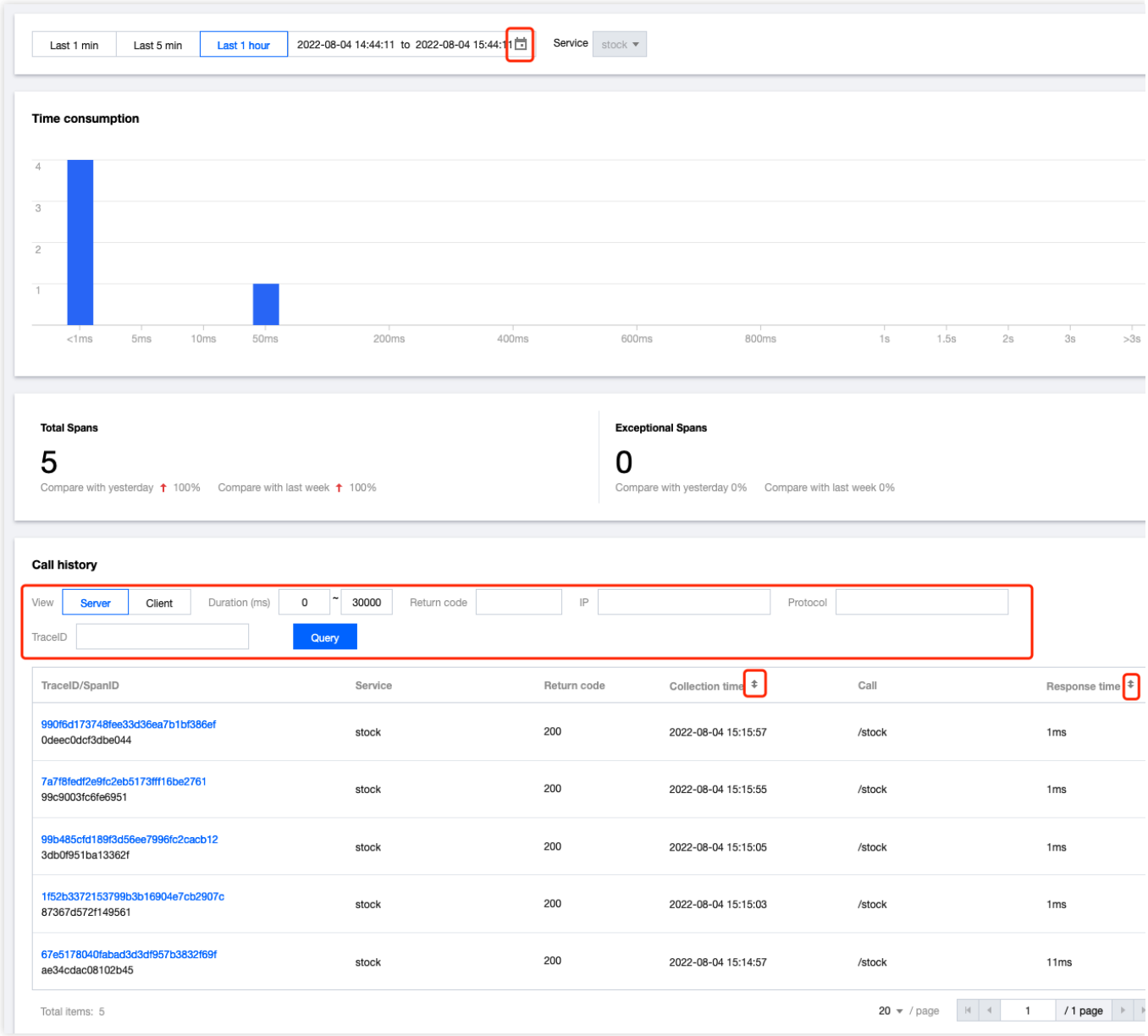
4. Click the call whose details to be viewed. You can view the detailed tracing logs of the call.



5. Click the close button to close the span details page and return to the list of called records.



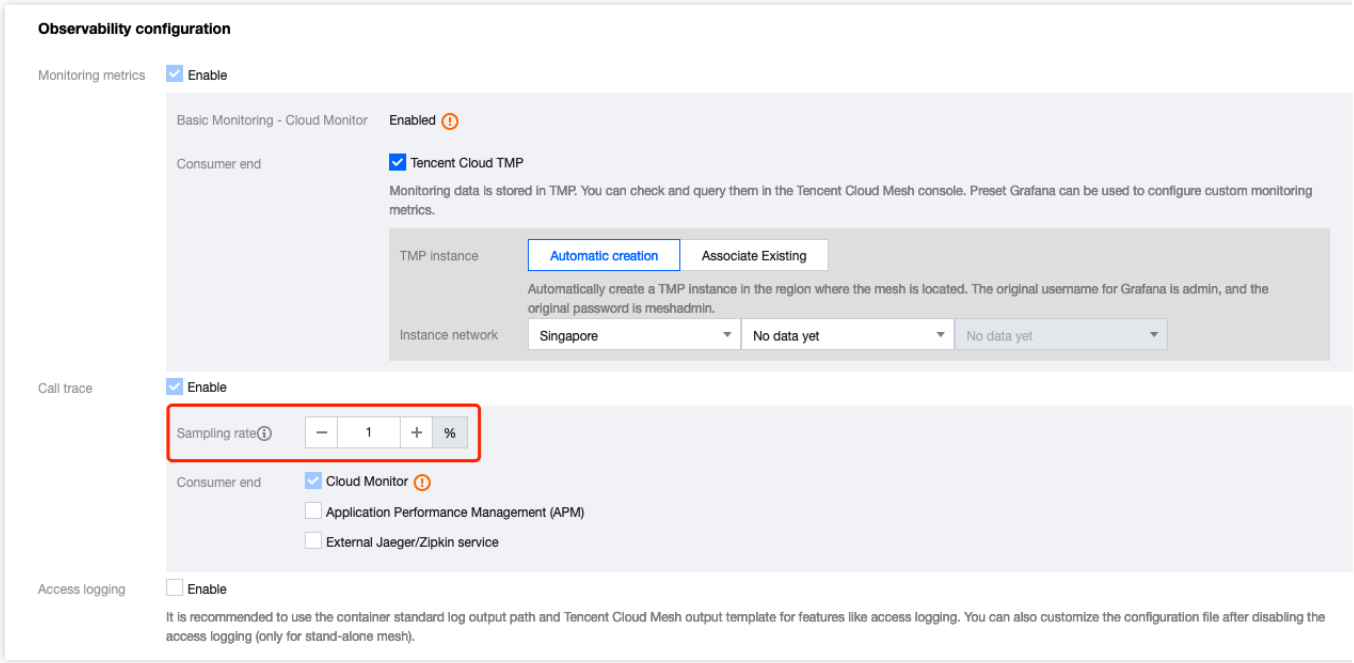
6. Tips for querying service's called records: You can filter the called records by duration, time span, source IP, trace ID, and return code. After filtering, you can sort the call records by **Latency** and **Start time**, so that you can easily choose the call you need to view.



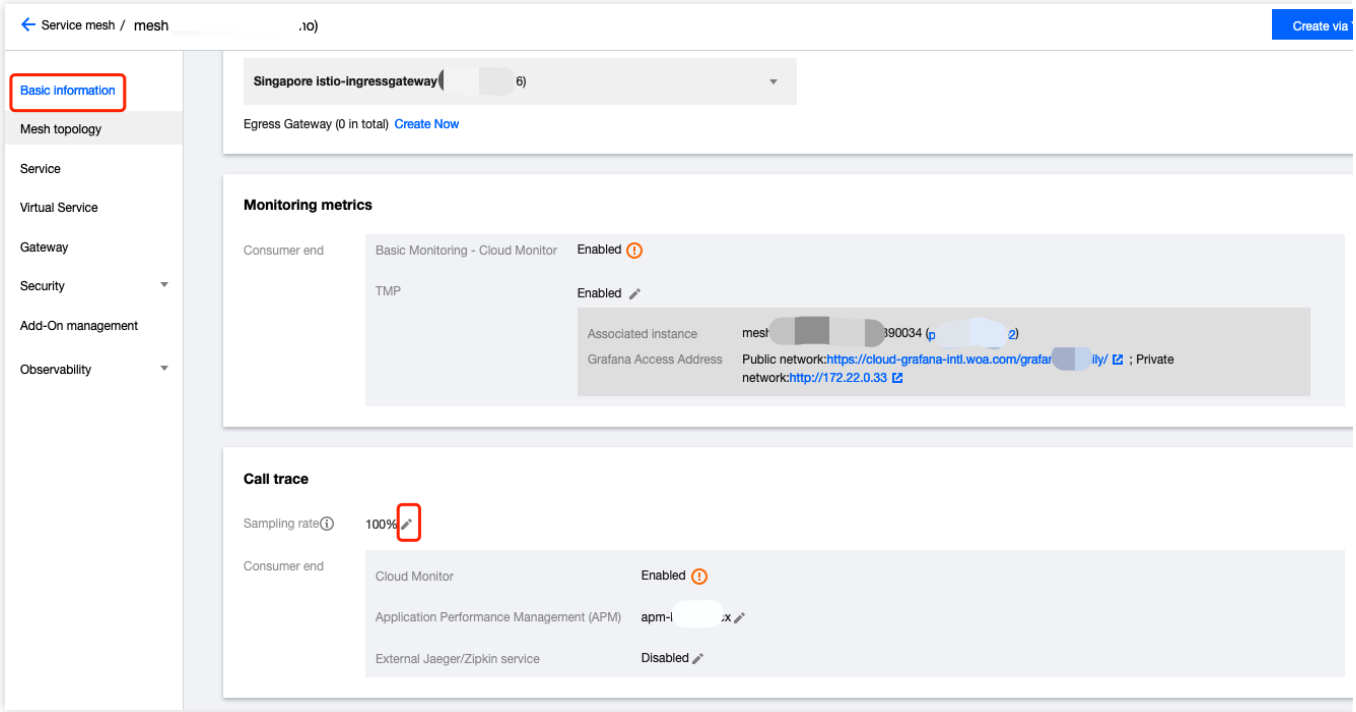
Configuring a Call Tracing Sampling Rate

A call tracing sampling rate is a sampling ratio of tracing data, and the resources consumed by sidecars during data collection and reporting are positively related to the bandwidth and sampling rate. Usually, in a production environment, it is not necessary to generate, collect, or report tracing data for all calls, so as to avoid excessive consumption of computing and bandwidth resources. Instead, only a certain proportion needs to be configured. It is recommended that a 100% sampling rate is configured for a development and test environment and a 1% sampling rate is configured for a production environment.

You can configure a sampling rate when creating a mesh.



Alternatively, you can modify sampling rate configurations on the basic information page of the mesh after the mesh is created.



Access Logs

Last updated：2023-12-26 14:17:14

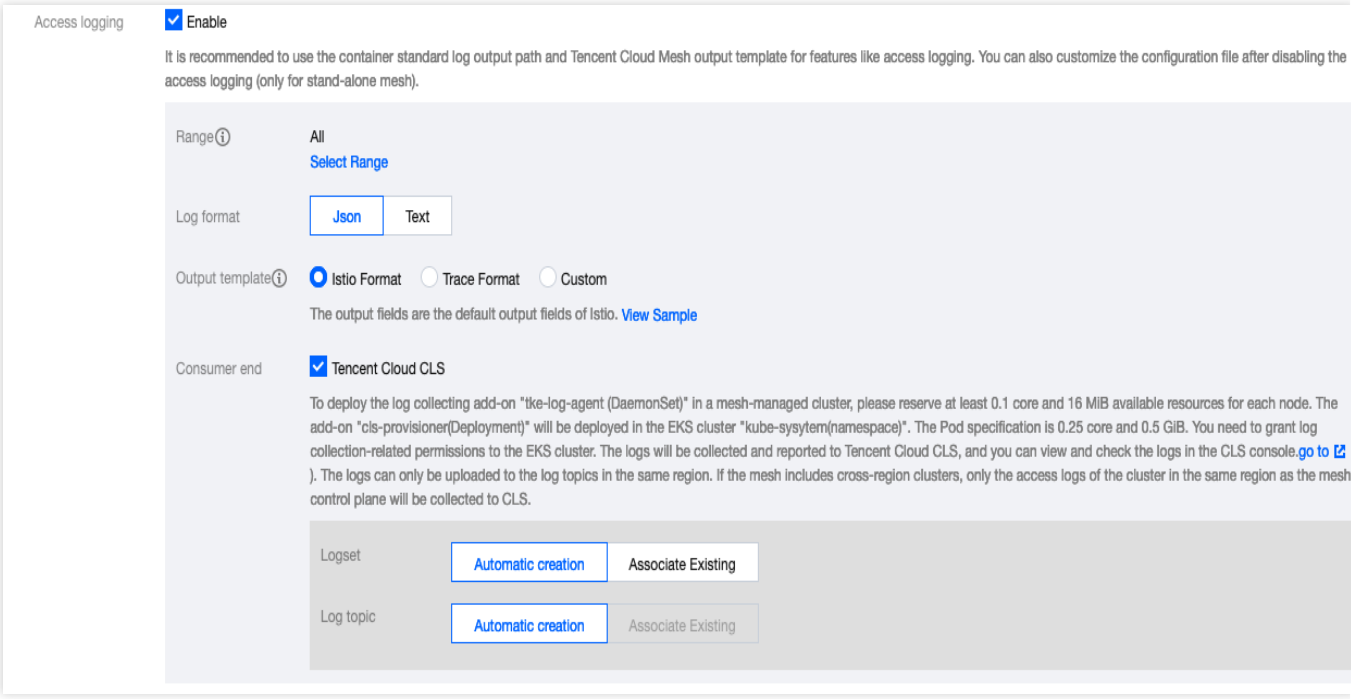
You can configure the output range and format of access logs (standard outputs of containers) of the data plane of a service mesh, and enable automatic collection of access logs to connect to Logset-Log Topic of Cloud Log Service (CLS). You can configure access logs when creating a mesh, and you can also modify access log configurations on the basic information page after the mesh is created.

Configuring Access Logs

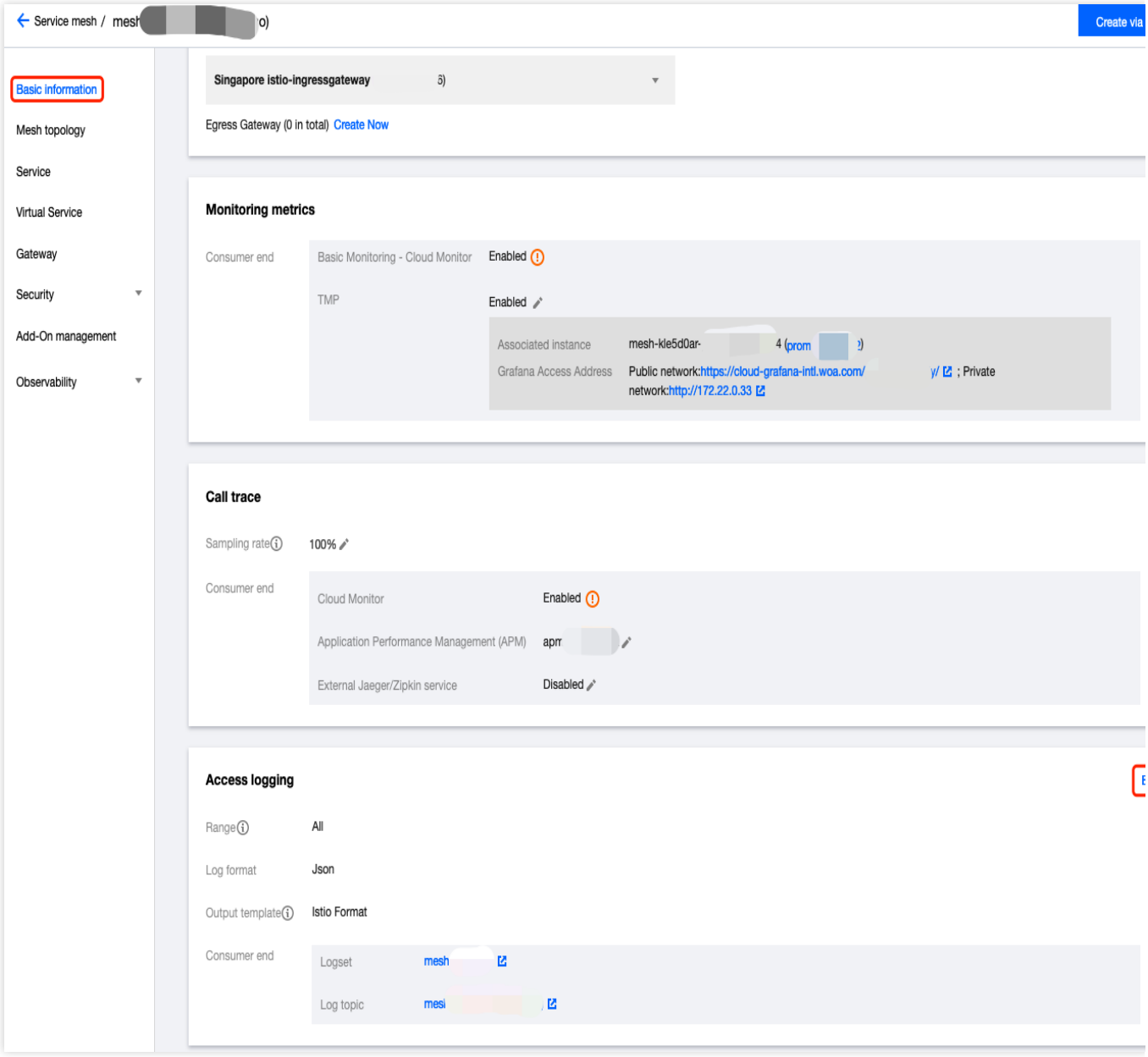
Currently, supported access log configurations are described as follows:

Configuration Item	Description
Range	Data plane (gateway and Istio proxy sidecar) for which access log outputting is enabled. You can enable access logs of all data planes of a specific gateway and namespace or all data planes of the mesh to be outputted to standard outputs of containers.
Output format	Output fields and templates of access logs. The fields output in the default format are the fields output by Istio by default. Compared with the fields output in the default format, the fields output in the enhanced format are added with **Trace ID**.
Consumer end	Configure to collect access logs from the standard outputs of data plane containers to CLS. You need to select a CLS logset and log topic for storing access logs. You can choose to automatically create a logset/topic, or associate an existing logset/topic. An automatically created logset is named in the format of `{mesh ID}` . The name of an automatically created log topic contains a Tencent Cloud Mesh identifier, that is, the log topic is named in the format of `{mesh ID}-accesslog` . After the request for enabling collection of access logs to CLS is submitted, the log collection feature is enabled on clusters managed by the mesh. Then, you need to deploy the log collection component tke-log-agent (DaemonSet) on the clusters managed by the mesh, and configure collection rules and indexes of Tencent Cloud Mesh's access logs. This feature is based on the log collection feature. Ensure that CLS has been activated, and that the service role `TKE_QCSRole` of TKE has been associated with the preset policy `QcloudAccessForTKERoleInOpsManagement` for operations management of CLS. For more information, see Description of Role Permissions Related to Service Authorization

Configuring access logs during mesh creation



Configuring access logs after mesh creation



Viewing Access Logs

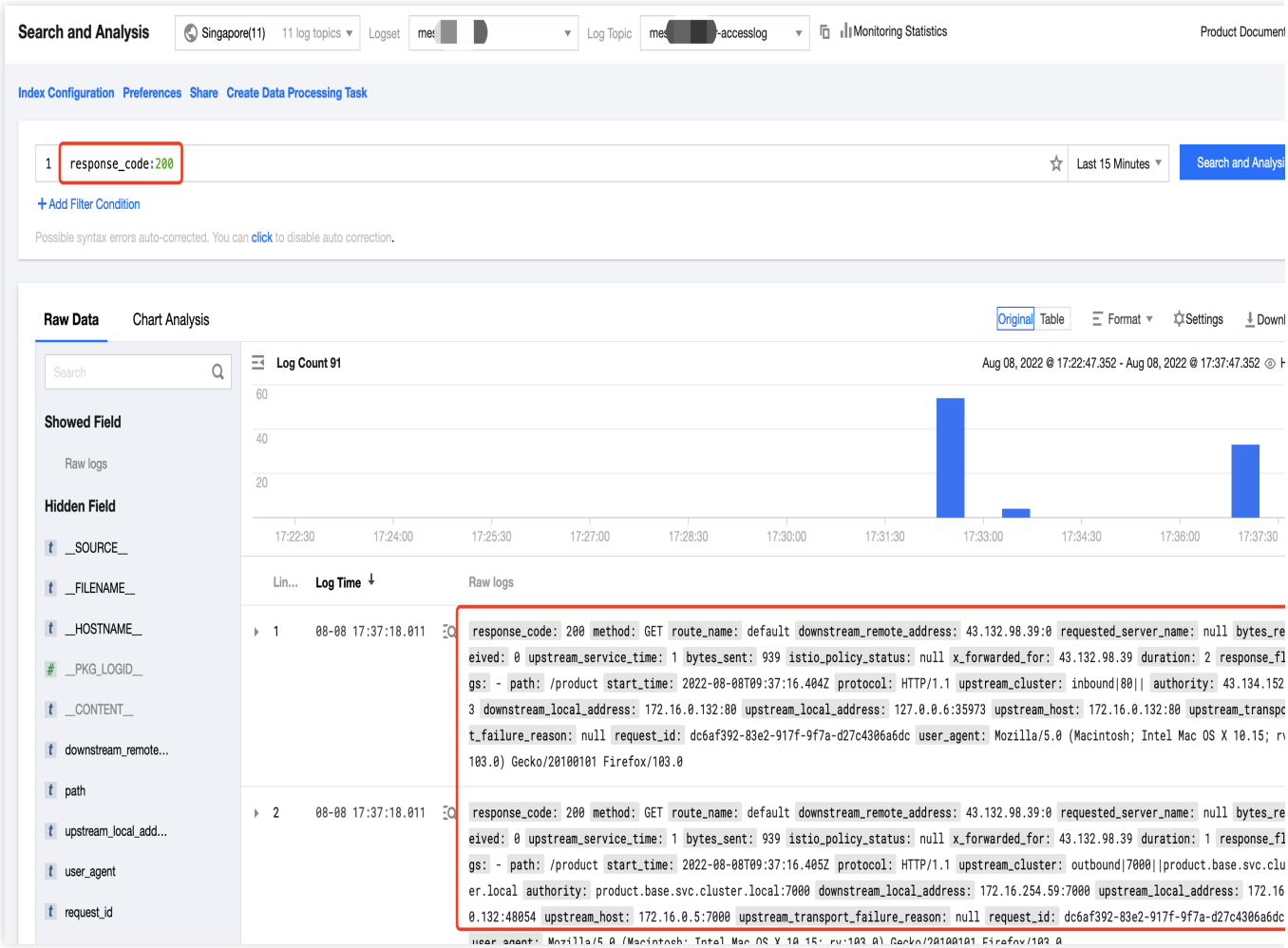
Viewing access logs through standard outputs of containers

Access logs of the Tencent Cloud Mesh data plane are output to the standard outputs of containers. You can view access logs in the standard outputs of the istio-proxy container through your Kubernetes cluster API server.

```
kubectl -n {Namespace} logs {Pod name} -c istio-proxy  --tail 5
```

Viewing access logs through CLS log search

If you have enabled consumer end configurations for access logs to collect the access logs of the Tencent Cloud Mesh data plane to CLS, you can select a corresponding log topic on the search and analysis page on the CLS console to view the access logs of the Tencent Cloud Mesh data plane. For details about CLS log search syntax, see Overview and Syntax Rules.



Security
Authentication Policy Configuration

Last updated：2023-12-26 14:17:48

Authentication policies include PeerAuthentication and RequestAuthentication. The PeerAuthentication policy is used to configure the mTLS mode of service communication, and the RequestAuthentication policy is used to configure a request authentication method of a service.

PeerAuthentication Configuration Field Description

Major PeerAuthentication fields are described as follows.

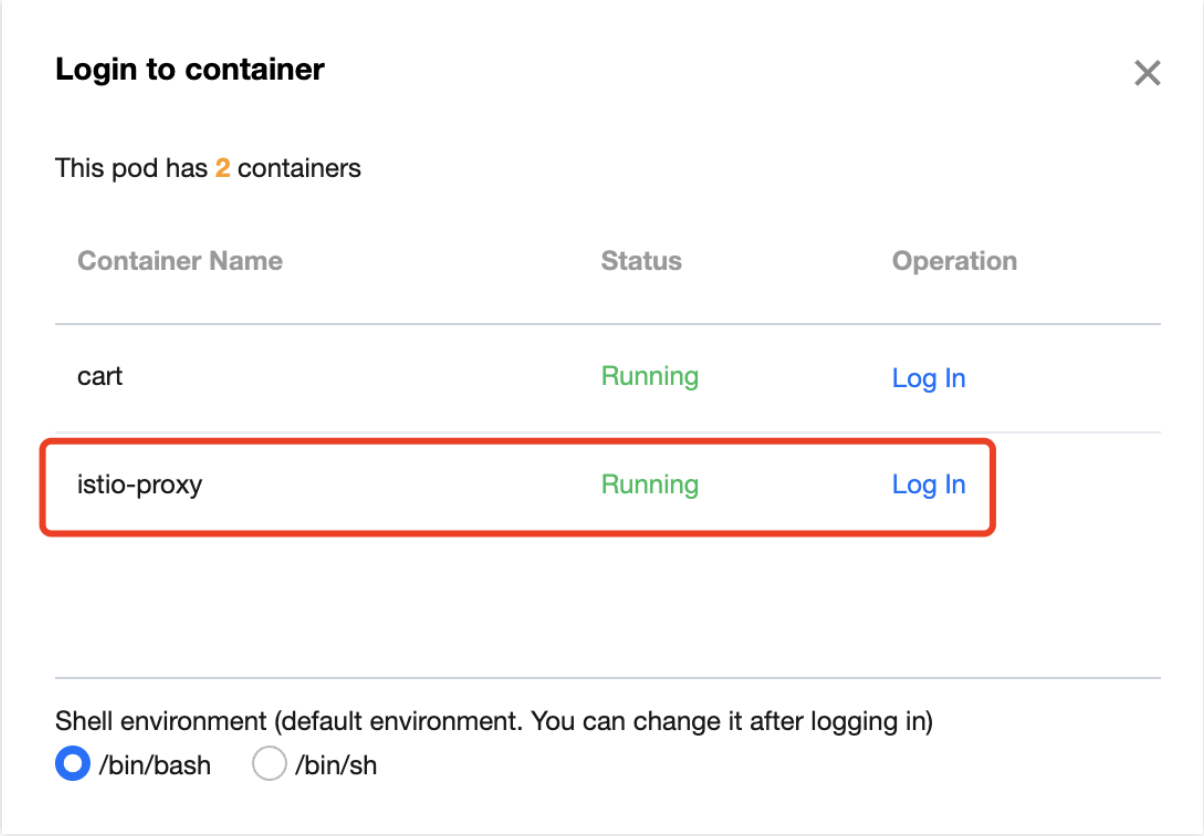
Name	Type	Description
`metadata.name`	`string`	PeerAuthentication name.
`metadata.namespace`	`string`	PeerAuthentication namespace.
`spec.selector`	`map<string, string>`	PeerAuthentication uses an entered label key-value pair and an entered namespace to match a scope of workloads to which configurations are to be delivered. If the entered namespace is istio-system and the selector field is left blank, the policy takes effect for the entire mesh. If the entered namespace is not istio-system and the selector field is left blank, the policy takes effect for the entered namespace. If the entered namespace is not istio-system and the selector field is set to a valid key-value pair, the policy takes effect for the workload that is matched based on the selector in the entered namespace.
`spec.mtls.mode`	-	mTLS mode. Four modes are supported: `UNSET`
`spec.portLevelMtls`	`map<uint32, mTLS mode>`	mTLS mode at the port level.

Note:

The effective priorities of mTLS mode configurations are as follows: port > service/workload > namespace > mesh.

Using PeerAuthentication to Configure the mTLS Mode for Service Communication in a Mesh

The mTLS mode in Tencent Cloud Mesh is PERMISSIVE by default, that is, the communication between services can be encrypted using mTLS or implemented through plaintext connections.
To test the effect of the mTLS mode configurations, you can first initiate a plaintext request to a service in your mesh and test the connectivity of the plaintext request. The following is an example of logging in to the istio-proxy container in the mesh and initiating a plaintext request to another service:
1. In the console of a TKE cluster managed by the mesh, log in to the istio-proxy container.



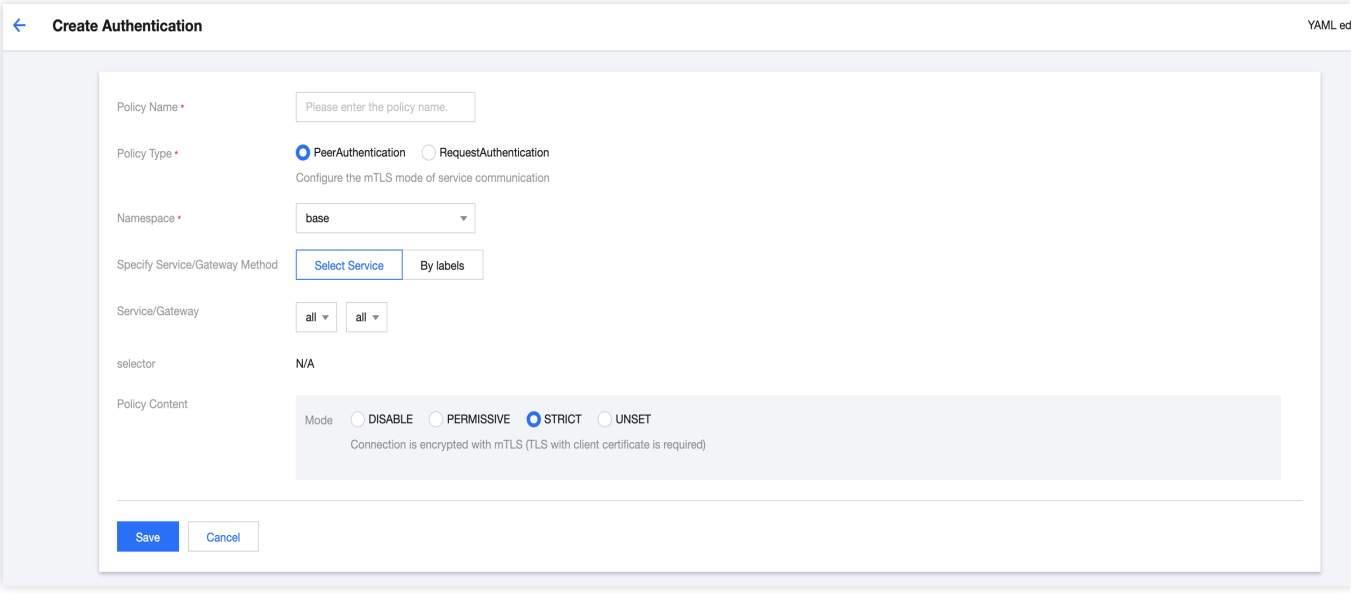
2. Enter the command `curl http://product.base.svc.cluster.local:7000/product` to access the product service in the base namespace in plaintext mode.
3. View the plaintext access result. If the product information is correctly returned, the plaintext access is successful.

Then, set the mTLS mode for the base namespace to **STRICT** and verify whether the configuration takes effect.

YAML Configuration Example

Console Configuration Example

```
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: base-strict
  namespace: base
spec:
  mtls:
    mode: STRICT
```



After the configuration is complete, you are prompted that the access fails when you access the product service in the base namespace in the plaintext mode again. This indicates that the mTLS STRICT mode has taken effect.

RequestAuthentication Configuration Field Description

Major RequestAuthentication configuration fields are described as follows.

Name	Type	Description
`metadata.name`	`string`	RequestAuthentication name.
`metadata.namespace`	`string`	RequestAuthentication names
`spec.selector`	`map<string, string>`	RequestAuthentication uses a value pair and an entered nam scope of workloads to which c be delivered. If the entered namespace is is selector field is left blank, the p the entire mesh. If the entered namespace is nc selector field is left blank, the p the entered namespace. If the entered namespace is nc selector field is set to a valid ke policy takes effect for the work based on the selector in the er
`spec.jwtRules.issuer`	`string`	JWT token issuer. For details,
`spec.jwtRules.audiences`	`string[]`	List of JWT audiences that are The service name will be acce list is empty.
`spec.jwtRules.jwksUri`	`string`	Public key URL for verifying JV details, see OpenID Discovery jwksUri and jwks fields are cor ignored.
`spec.jwtRules.jwks`	`string`	Public key in a JSON Web Ke

		JWT signatures. When both th fields are configured, jwksUri i
`spec.jwtRules.fromHeaders`	`map<string,string>[]`	List of locations in the header f is extracted.
`spec.jwtRules.fromParams`	`string[]`	Parameters in the header from extracted. For example, the JW the parameter mytoken (`/pa`
`spec.jwtRules.outputPayloadToHeader`	`string`	Header name output by a JWT successful verification. The for `base64_encoded(jwt_pa` If this field is left blank, a JWT by default.
`spec.jwtRules.forwardOriginalToken`	`bool`	Whether to forward the raw JW default value is `false` .

Using RequestAuthentication to Configure JWT Request Authentication

To verify the effect of configurations for JWT request authentication, you first need to deploy a test program `httpbin.foo` and then configure this service to be exposed to the public network through an ingress gateway. Create a foo namespace with automatic sidecar injection enabled, and deploy the httpbin service to the foo namespace.

```
apiVersion: v1
kind: Namespace
metadata:
  name: foo
  labels:
    istio.io/rev: 1-6-9 # Enable automatic sidecar injection for the namespace (The
spec:
  finalizers:
    - kubernetes
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: httpbin
  namespace: foo
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  namespace: foo
  labels:
    app: httpbin
    service: httpbin
spec:
  ports:
  - name: http
    port: 8000
    targetPort: 80
  selector:
    app: httpbin
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpbin
  namespace: foo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
      version: v1
  template:
    metadata:
      labels:
        app: httpbin
        version: v1
    spec:
      serviceAccountName: httpbin
      containers:
      - image: docker.io/kennethreitz/httpbin
        imagePullPolicy: IfNotPresent
        name: httpbin
        ports:
        - containerPort: 80
```

Configure the httpbin service to be exposed to the public network for access through the ingress gateway.

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: httpbin-gateway
```

```
  namespace: foo
spec:
  selector:
    app: istio-ingressgateway
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: httpbin
  namespace: foo
spec:
  hosts:
  - "*"
  gateways:
  - httpbin-gateway
  http:
  - route:
    - destination:
        port:
          number: 8000
        host: httpbin.foo.svc.cluster.local
```

Test the connectivity of the service by using the curl statement `curl "$INGRESS_IP:80/headers" -s -o /dev/null -w "%{http_code}\\n"` . Note that you need to replace `$INGRESS_IP` in the statement with the IP address of your ingress gateway. In normal condition, a `200` return code is returned.

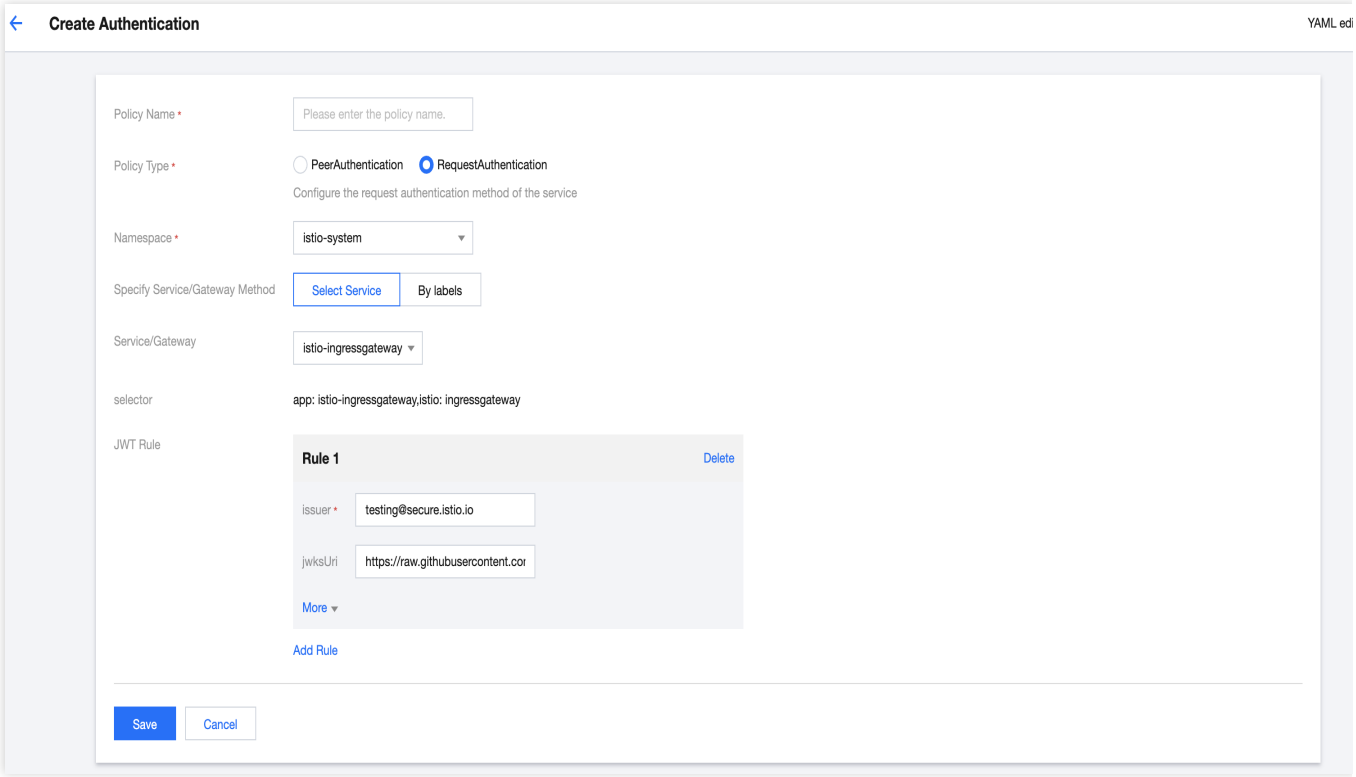
The following configures JWT authentication rules for the ingress gateway to allow requests carrying eligible JWT tokens.

YAML Configuration Example

Console Configuration Example

```
apiVersion: "security.istio.io/v1beta1"
kind: "RequestAuthentication"
metadata:
  name: "jwt-example"
  namespace: istio-system
spec:
  selector:
```

```
    matchLabels:
      istio: ingressgateway
      app: istio-ingressgateway
  jwtRules:
  - issuer: "testing@secure.istio.io"
    jwksUri: "https://raw.githubusercontent.com/istio/istio/release-1.9/security/to
```



After the configuration is complete, verify whether the configured JWT authentication rule takes effect.

Use the following code that carries an invalid JWT token to initiate access. Note that you need to replace `$INGRESS_IP` in the code with the IP address of your ingress gateway. The ingress gateway does not allow the request carrying the invalid JWT token and therefore returns a `401` return code.

```
curl --header "Authorization: Bearer deadbeef" "$INGRESS_IP:80/headers" -s -o
/dev/null -w "%{http_code}\\n"
```

Use the following code that carries a valid JWT token to initiate access. Note that you need to replace `$INGRESS_IP` in the code with the IP address of your ingress gateway. The ingress gateway allows the request carrying the illegal JWT token and therefore returns a `200` return code.

```
TOKEN=$(curl https://raw.githubusercontent.com/istio/istio/release-
1.9/security/tools/jwt/samples/demo.jwt -s)
curl --header "Authorization: Bearer $TOKEN" "$INGRESS_IP:80/headers" -s -o
/dev/null -w "%{http_code}\\n"
```

Through verification, you can find that the JWT request authentication rule that you configured for the ingress gateway has taken effect. Because only the JWT authentication rule is configured at this time, the ingress gateway still allows requests that do not carry a JWT token. To restrict requests that do not carry a JWT token, you need to configure an AuthorizationPolicy. Apply the following YAML file to the service mesh to control the ingress gateway to deny requests that do not carry a JWT token.

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: frontend-ingress
  namespace: istio-system
spec:
  selector:
    matchLabels:
      app: istio-ingressgateway
      istio: ingressgateway
  rules:
    - from:
        - source:
            notRequestPrincipals:
              - '*'
  action: DENY
```

Use the following code that does not carry a JWT token to initiate access again: `curl "$INGRESS_IP:80/headers" -s -o /dev/null -w "%{http_code}\\n"` . It is found that the access fails and a `403` return code is returned, indicating that the AuthorizationPolicy policy has taken effect.

Authorization Policy Configuration

Last updated：2023-12-26 14:18:45

An authorization policy is used to configure access management rules in scopes such as a mesh, namespace, and service/workload. You can configure authorization rules by using an AuthorizationPolicy CRD. AuthorizationPolicy includes the following parts:

selector：specifies the effective scope of the policy.

action: specifies whether the policy is an `ALLOW` policy or a `DENY` policy.

rules: specifies an authorization rule body, consisting of from, to, and where.

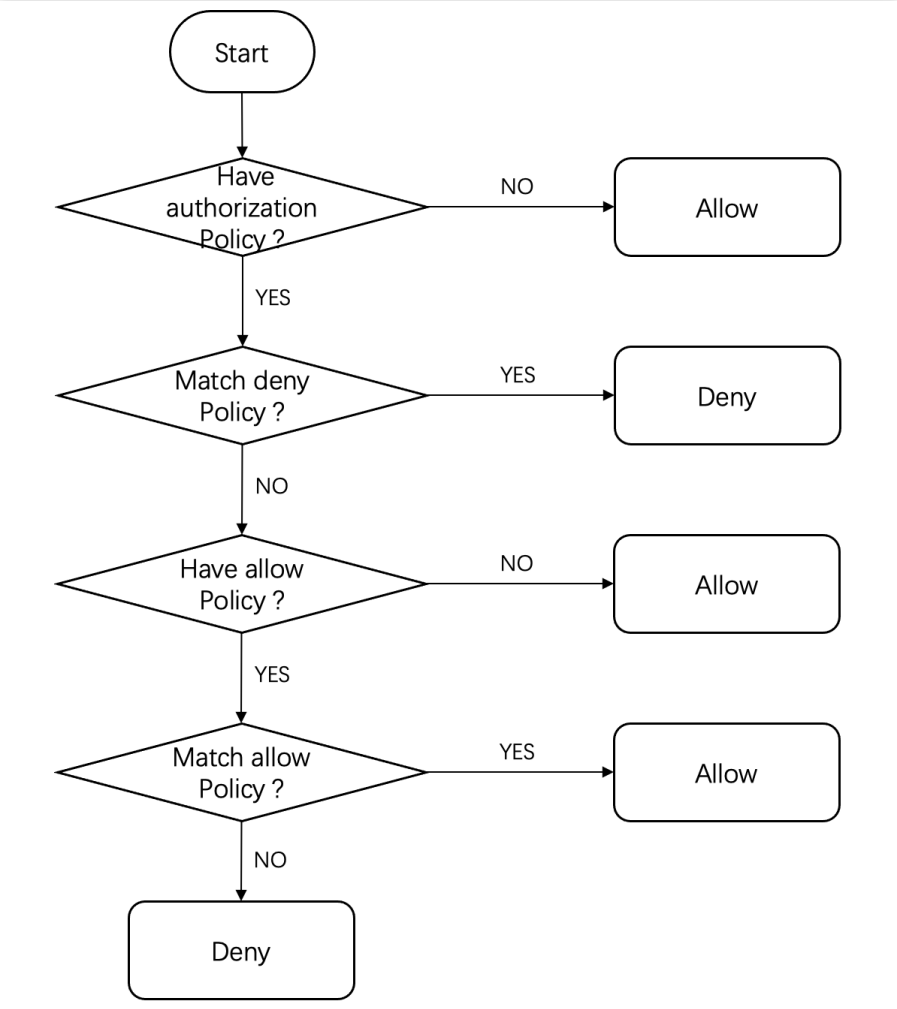
from: specifies the source of a request.

to: specifies the operation of a request.

when: specifies a condition for an authorization rule to take effect.

When `ALLOW` and `DENY` policies of AuthorizationPolicy are applied to a same scope, the `DENY` policy takes precedence over the `ALLOW` policy. The effective rules are as follows:

1. If there are any `DENY` policies that match the request, deny the request.
2. If there are no `ALLOW` policies for the scope, allow the request.
3. If there are any `ALLOW` policies for the scope and any of the `ALLOW` policies matches the request, allow the request.
4. Deny the request.



The following are two special AuthorizationPolicy examples:

Services in the default namespace allow all requests.

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: allow-all
  namespace: default
spec:
  action: ALLOW
  rules:
  - {} # The rule can match any request.
```

Services in the default namespace deny all requests.

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: deny-all
  namespace: default
```

```
spec:
  {} # When the action field is left blank, the value is **ALLOW** by default. In t
```

Description of Major AuthorizationPolicy Fields

Major AuthorizationPolicy fields are described as follows.

Name	Type	Description
`metadata.name`	`string`	AuthorizationPolicy name.
`metadata.namespace`	`string`	AuthorizationPolicy namespace
`spec.selector`	`map<string, string>`	AuthorizationPolicy uses an entered label key-value pair and an entered namespace to match a scope of workloads to which configurations are to be delivered. If the entered namespace is isti system and the selector field is left blank, the policy takes effect for the entire mesh. If the entered namespace is not istio-system and the selector fie is left blank, the policy takes effect for the entered namespac If the entered namespace is not istio-system and the selector fie is set to a valid key-value pair, the policy takes effect for the workload that is matched based on the selector in the entered namespace.
`spec.action`	-	Whether the policy is an `ALLOW` policy or a `DENY` policy.
`spec.rules.from.source.principals`	`string[]`	List of source peer identities (th is, service accounts). This field matches the `source.principal` field and requires mTLS enabled. If

		this field is left blank, any principal is allowed.
`spec.rules.from.source.requestPrincipals`	`string[]`	List of request identities (that is, iss/sub claim). This field matches the `request.auth.principal` field. If this field is left blank, any request principal is allowed.
`spec.rules.from.source.namespaces`	`string[]`	List of namespaces of the request source. This field matches the `source.namespace` field and requires mTLS enabled. If this field is left blank, requests from any namespace are allowed.
`spec.rules.from.source.ipBlocks`	`string[]`	List of IP blocks. This field matches the `source.ip` fiel and supports single IP (for example, `1.2.3.4`) and CID (for example, `1.2.3.4/24`). If this field is left blank, any source IP address is allowed.
`spec.rules.to.operation.hosts`	`string[]`	List of domain names in the request. This field matches the `request.host` field. If this field is left blank, any domain name is allowed. This field can be used only in HTTP requests.
`spec.rules.to.operation.ports`	`string[]`	List of ports in the request. This field matches the `destination.port` field. I this field is left blank, any port is allowed.
`spec.rules.to.operation.methods`	`string[]`	List of methods in the request. This field matches the `request.method` field. If th gRPC protocol is used, this field is always `POST` . If this field is left blank, any method is allowe

		This field can be used only in HTTP requests.
`spec.rules.to.operation.paths`	`string[]`	List of paths in the request. This field matches the `request.url_path` field. I this field is left blank, any path is allowed. This field can be used only in HTTP requests.
`spec.rules.when.condition.key`	`string`	Names of conditions supported by Istio. For details, see Authorization Policy Conditions.
`spec.rules.when.condition.values`	`string[]`	List of values for a correspondir condition.

Using AuthorizationPolicy to Configure Namespace Access Permissions

To check the effect of the configured AuthorizationPolicy policy, first deploy a set of test programs to a cluster managed by the mesh. After the deployment is complete, the client service in the test namespace will automatically initiate access to the user service in the base namespace.

```
apiVersion: v1
kind: Namespace
metadata:
  name: test
  labels:
    istio.io/rev: 1-6-9 # Automatic sidecar injection (Istio 1.6.9)
spec:
  finalizers:
    - kubernetes
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: client
  namespace: test
  labels:
    app: client
spec:
  replicas: 10
  selector:
```

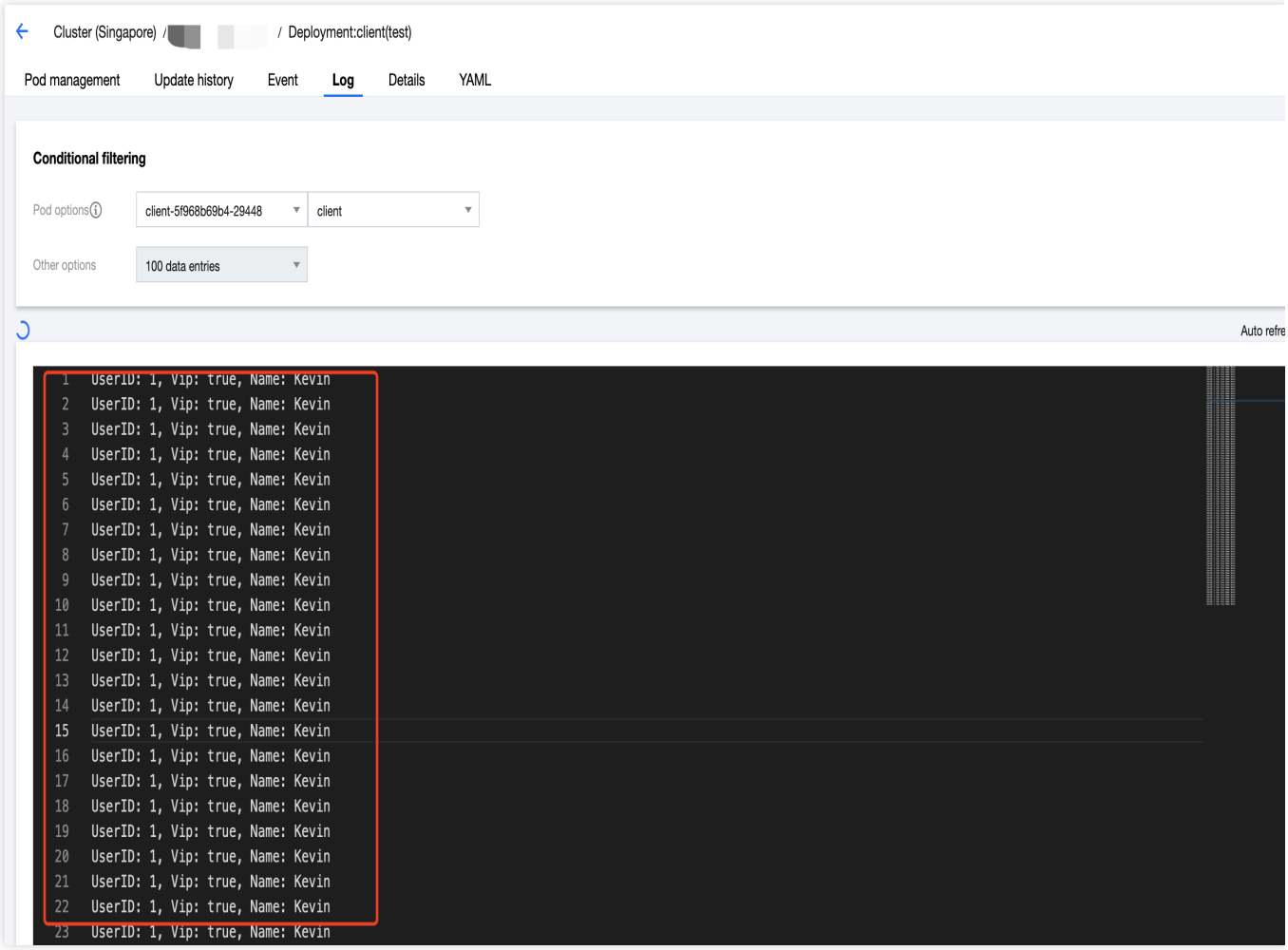
```
    matchLabels:
      app: client
  template:
    metadata:
      labels:
        app: client
    spec:
      containers:
        - name: client
          image: ccr.ccs.tencentyun.com/zhulei/testclient:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneA"
          ports:
            - containerPort: 7000
              protocol: TCP
---

apiVersion: v1
kind: Service
metadata:
  name: client
  namespace: test
  labels:
    app: client
spec:
  ports:
    - name: http
      port: 7000
      protocol: TCP
  selector:
    app: client
  type: ClusterIP
---
apiVersion: v1
kind: Namespace
metadata:
  name: base
  labels:
    istio.io/rev: 1-6-9
spec:
  finalizers:
```

```
    - kubernetes
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: user
  namespace: base
  labels:
    app: user
spec:
  replicas: 1
  selector:
    matchLabels:
      app: user
  template:
    metadata:
      labels:
        app: user
    spec:
      containers:
        - name: user
          image: ccr.ccs.tencentyun.com/zhulei/testuser:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
---

apiVersion: v1
kind: Service
metadata:
  name: user
  namespace: base
  labels:
    app: user
spec:
  ports:
    - port: 7000
      name: http
  selector:
    app: user
```

View logs of the client container. It is found that the access is successful and the user information is correctly returned.



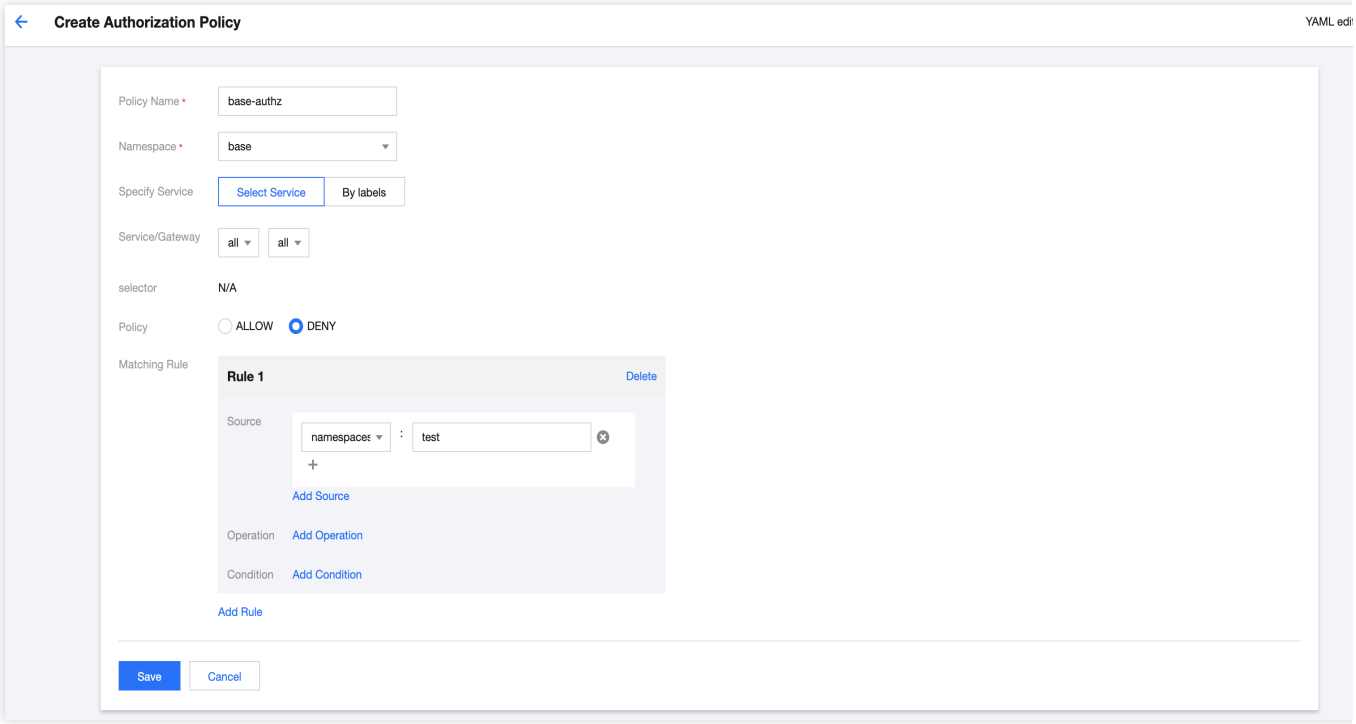
Next, configure AuthorizationPolicy to restrict services in the base namespace from being accessed by services in the test namespace. In this case, mTLS needs to be enabled.

YAML Configuration Example

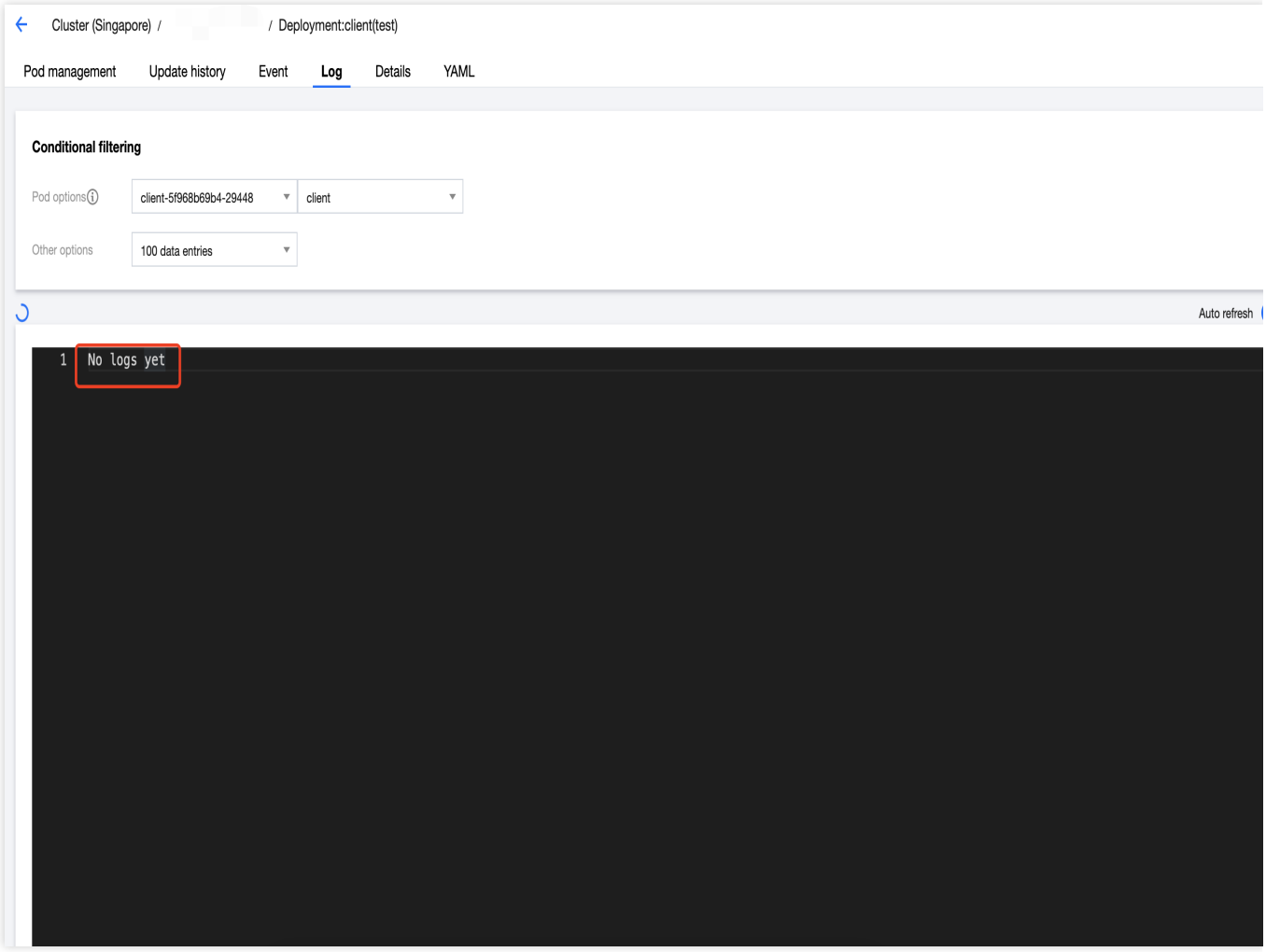
Console Configuration Example

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: base-authz
  namespace: base
spec:
  action: DENY
  rules:
    - from:
        - source:
            namespaces:
```

```
      - test
```



After the configuration is complete, view logs of the client container again. It is found that all access requests fail and no user information is returned, indicating that AuthorizationPolicy has taken effect.



Using AuthorizationPolicy to Configure an IP Blocklist/Allowlist of the Ingress Gateway

You can use AuthorizationPolicy to configure an IP blocklist/allowlist for the ingress gateway.

To verify the effect of blocklist/allowlist configurations, you first need to deploy a test program `httpbin.foo` and then configure this service to be exposed to the public network through the ingress gateway.

Create a foo namespace with automatic sidecar injection enabled, and deploy the httpbin service to the foo namespace.

```
apiVersion: v1
kind: Namespace
metadata:
  name: foo
  labels:
    istio.io/rev: 1-6-9 # Enable automatic sidecar injection for the namespace (The
spec:
```

```
  finalizers:
    - kubernetes
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: httpbin
  namespace: foo
---
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  namespace: foo
  labels:
    app: httpbin
    service: httpbin
spec:
  ports:
  - name: http
    port: 8000
    targetPort: 80
  selector:
    app: httpbin
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpbin
  namespace: foo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
      version: v1
  template:
    metadata:
      labels:
        app: httpbin
        version: v1
    spec:
      serviceAccountName: httpbin
      containers:
      - image: docker.io/kennethreitz/httpbin
        imagePullPolicy: IfNotPresent
        name: httpbin
```

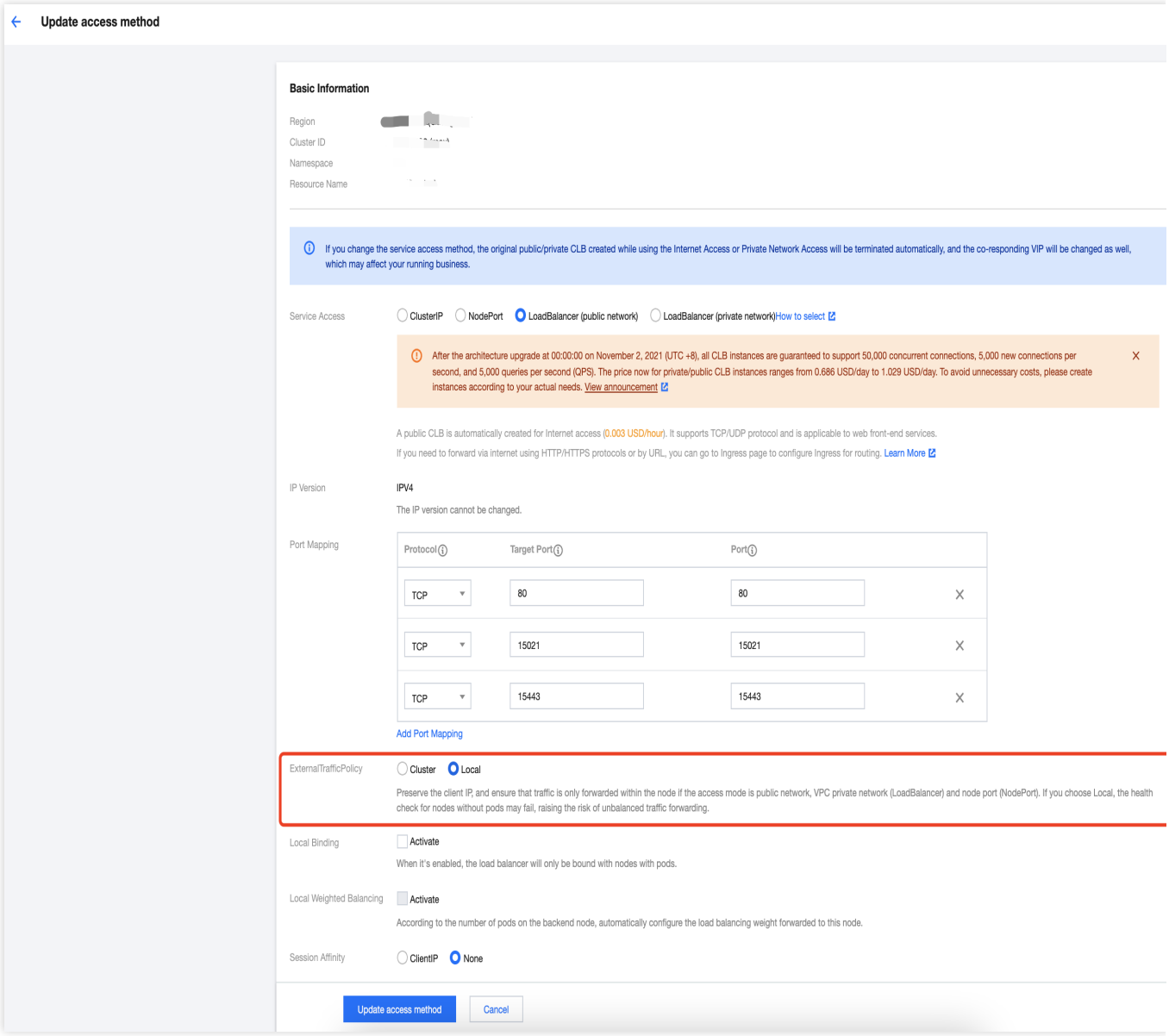
```
        ports:
        - containerPort: 80
```

Configure the httpbin service to be exposed to the public network for access through the ingress gateway.

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: httpbin-gateway
  namespace: foo
spec:
  selector:
    app: istio-ingressgateway
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: httpbin
  namespace: foo
spec:
  hosts:
  - "*"
  gateways:
  - httpbin-gateway
  http:
  - route:
    - destination:
        port:
          number: 8000
        host: httpbin.foo.svc.cluster.local
```

Test the connectivity of the service by using the curl statement `curl "$INGRESS_IP:80/headers" -s -o /dev/null -w "%{http_code}\\n"` . Note that you need to replace `$INGRESS_IP` in the statement with the IP address of your ingress gateway. In normal condition, a `200` return code is returned.

To enable the ingress gateway to correctly obtain the source IP address of the real client, you need to change ExternalTrafficPolicy of the ingress gateway service to **Local**, so that traffic is forwarded only on this node and SNAT is not performed.



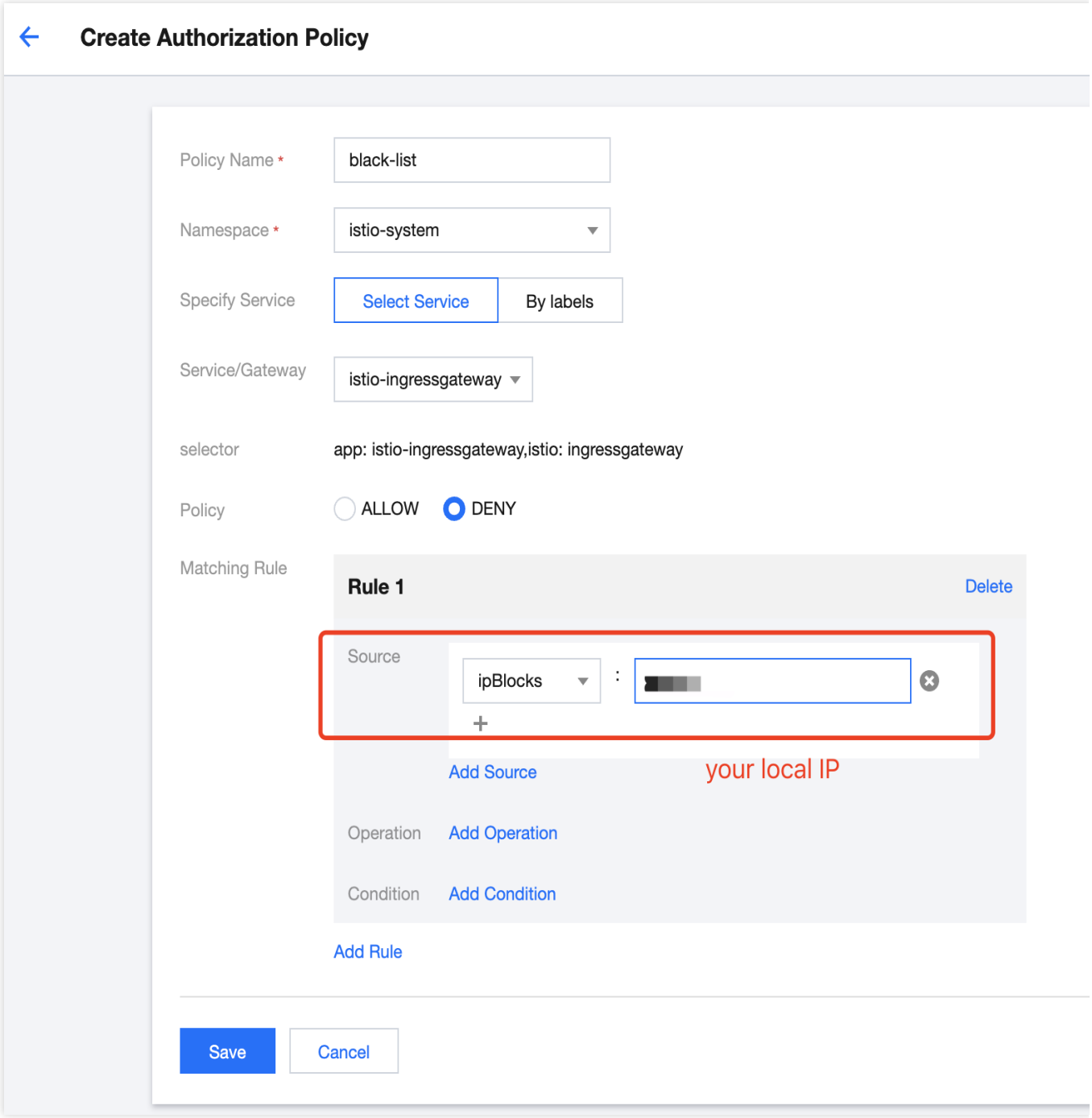
The following uses AuthorizationPolicy to add the IP address of the local host to the blocklist of the ingress gateway, and verify whether the blocklist takes effect.

YAML Configuration Example

Console Configuration Example

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: black-list
  namespace: istio-system
spec:
  selector:
    matchLabels:
      app: istio-ingressgateway
      istio: ingressgateway
```

```
rules:
  - from:
      - source:
          ipBlocks:
            - $ IP address of your local host
action: DENY
```



After the configuration is complete, test the connectivity of the service by using the curl statement `curl "$INGRESS_IP:80/headers" -s -o /dev/null -w "%{http_code}\\n"` again. Note that you need to

replace `$INGRESS_IP` in the statement with the IP address of your ingress gateway. In this case, the access fails and a `403` return code is returned, indicating that the blocklist policy has taken effect.

Access Management Overview

Last updated：2023-12-26 14:20:12

Permission management of a service mesh contains management of Cloud Access Management (CAM) permissions and Tencent Kubernetes Engine (TKE) RBAC permissions.
By default, a sub-account does not have CAM permissions, and a sub-account that is not a cluster creator does not have RBAC permissions for the related cluster. You need to create and associate CAM policies and TKE RBAC authorization policies to allow sub-accounts to access or normally use service mesh resources they need.
CAM permission policies are edited and granted by a CAM administrator (usually a root account or a sub-account with CAM permissions). For more basic information about CAM policies, see CAM policies. RBAC permission policies of a TKE cluster are usually edited and granted by a corresponding cluster administrator (usually a root account or an account that creates the cluster). For information about authorization methods, see TKE RBAC authorization.
Note:
Skip this chapter if you do not need to manage the access permission of sub-accounts for Tencent Cloud Mesh resources. This will not affect your understanding and use of the other sections of the document.

CAM-based Permission Control

Currently, Tencent Cloud Mesh supports CAM-based resource-level permission control. In other words, Tencent Cloud Mesh can allow specified **sub-accounts** to perform specified **operations** on specified **resources**. The sub-accounts do not have Tencent Cloud Mesh-related CAM permissions by default. You need to associate policies with the sub-accounts to complete authorization.
In addition, Tencent Cloud Mesh supports CAM-based resource-level permission control at a granularity of mesh instance. In other words, you can control specified sub-account to perform specified operations on a specified mesh.

RBAC Permission Management of TKE (Tencent Cloud Mesh-related Product)

The use of Tencent Cloud Mesh involves read and write operations on Kubernetes resources in the TKE clusters managed by Tencent Cloud Mesh. These operations require sufficient TKE RBAC permissions are available. By default, a sub-account that is not the cluster creator does not have the RBAC permissions for the cluster. The cluster administrator needs to grant the RBAC permissions for the corresponding cluster to the sub-account before the sub-account can use Tencent Cloud Mesh normally.

The following operations require administrator (tke:admin) permissions for the corresponding cluster: creating/deleting/updating a service mesh in the selected cluster, adding/dissociating a service discovery cluster, and creating/deleting an ingress gateway in the selected cluster. Operations on Istio resources (such as Gateway, VirtualService, DestinationRule, and ServiceEntry) in the mesh do not require RBAC permissions for the cluster.
For more information about TKE Kubernetes object-level permission control, see TKE Kubernetes Object-level Permission Control. For information about TKE RBAC authorization modes, see Comparison of Authorization Modes.

CAM Service Role Authorization

Last updated：2024-12-17 15:12:29

The use of Tencent Cloud Mesh involves service mesh-related cloud resources. To use Tencent Cloud Mesh features normally, you need to authorize the service role `TCM_QCSRole` of Tencent Cloud Mesh. The Tencent Cloud Mesh service can use related cloud resources only after authorization.
Scenarios that require service authorization mainly include Initial Login to the Tencent Cloud Mesh Console and Initial Use of Tencent Cloud Mesh Sample Deployment. The two scenarios correspond to two preset policies `QcloudAccessForTCMRole` and `QcloudAccessForTCMRoleInSampleDeployment`, respectively.

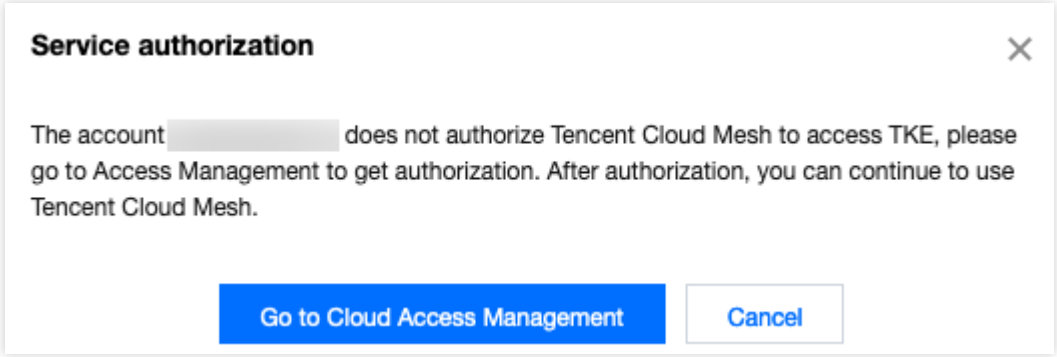
Initial Login to the Tencent Cloud Mesh Console

Authorization Scenario

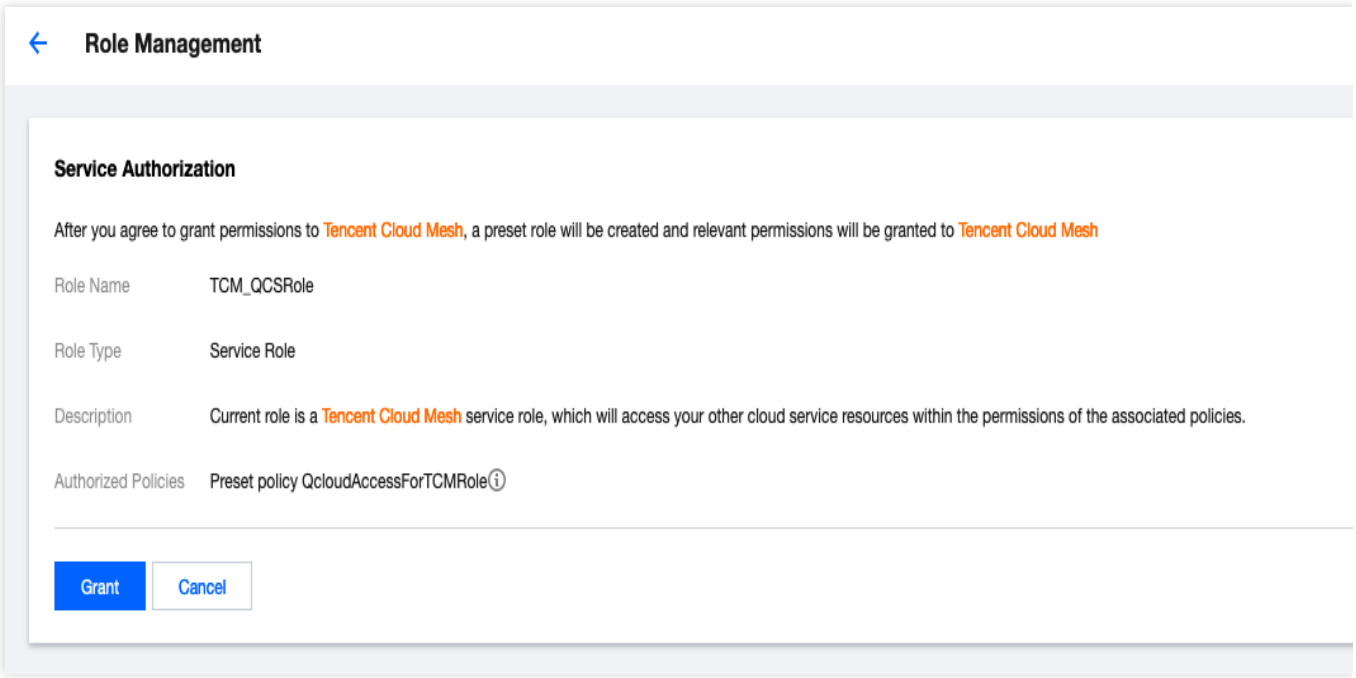
When you log in to the Tencent Cloud Mesh console for the first time after registering and logging in to a Tencent Cloud account, you need to go to the **Cloud access management** page to grant the current account Tencent Cloud Mesh permissions for operating on TKE, SSL certificates, CLS, and other cloud resources. The permissions are granted by associating the preset policy `QcloudAccessForTCMRole` with the service role `TCM_QCSRole` of Tencent Cloud Mesh. This authorization process also involves the creation of a Tencent Cloud Mesh service role if you have not created a Tencent Cloud Mesh service role yet.

Authorization Steps

1. Log in to the Tencent Cloud Mesh console. For the initial login, the **Service authorization** window automatically pops up.



2. Click **Go to cloud access management** to enter the **Service authorization** page.
3. Click **Grant** to complete authentication.



Permission Content

TKE

Permission	Description	Resource
DescribeClusterSecurity	Querying cluster keys	All resources `*`

SSL certificate

Permission	Description	Resource
DescribeCertificateDetail	Obtaining certificate details	All resources `*`

CLS

Permission	Description	Resource
getLogset	Obtaining logset details	All resources `*`
getTopic	Obtaining log topic details	All resources `*`
createLogset	Creating a logset	All resources `*`
createTopic	Creating a log topic	All resources `*`
modifyIndex	Modifying an index	All resources `*`

listLogset	Obtaining a logset list	All resources *
listTopic	Obtaining a log topic list	All resources *

CAM Preset Policy Authorization

Last updated：2023-12-26 14:20:49

You can associate Tencent Cloud Mesh-related preset policies in CAM with sub-accounts to rapidly complete CAM authorization for Tencent Cloud Mesh.

Tencent Cloud Mesh-related Preset Policies

You can grant your sub-account the necessary permissions by using the following preset policies:

Policy	Description
`QcloudTCMFullAccess`	Full access to Tencent Cloud Mesh (All operations such as creation and deletion are allowed.)
`QcloudTCMReadOnlyAccess`	Read-only access to Tencent Cloud Mesh (Viewing all resources in Tencent Cloud Mesh is allowed, but creating, updating, and deleting them are not allowed.)

Preset Policy for Full Access to Tencent Cloud Mesh

Policy name: QcloudTCMFullAccess; policy content:

```
{
    "version": "2.0",
    "statement": [
        {
            "action": [
                "tcm:*"
            ],
            "resource": "*",
            "effect": "allow"
        }
    ]
}
```

Preset Policy for Read-Only Access to Tencent Cloud Mesh

Policy name: QcloudTCMReadOnlyAccess; policy content:

```
{
    "version": "2.0",
    "statement": [
```

```
        {
            "action": [
                "tcm:List*",
                "tcm:Describe*",
                "tcm:ForwardRequestRead"
            ],
            "resource": "*",
            "effect": "allow"
        }
    ]
}
```

CAM Permissions of Tencent Cloud Mesh-related Products

The use of Tencent Cloud Mesh also involves CAM permissions of related products such as VPC, CCN, CLB, and TKE. You can grant appropriate permissions to sub-accounts by referring to the CAM authorization document of the corresponding product.

Tencent Cloud Mesh-related Product	Authorization Guide
VPC	Cloud Access Management Overview
CLB	Overview
TKE	Overview

Associating Sub-accounts with Preset Policies

In the step for setting user permissions when creating a sub-account, you can associate preset policies with the sub-account by direct association or association via group.

Direct Association

You can directly associate your sub-account with a policy to obtain the permissions contained in the policy.

1. Log in to the CAM console and choose **Users** > **User list** on the left sidebar.
2. On the **User list** page, find the target sub-account and click **Grant permission** in the **Operation** column.
3. On the **Associate policies** page, select the policies that you want to associate.
4. Click **OK**.

Association via Group

You can add your sub-account to a user group. Then, the sub-account automatically obtains the permissions that are associated with this user group. To disassociate the sub-account from the policies of the group, you simply need to remove the sub-account from the user group.

1. Log in to the CAM console and choose **Users** > **User list** on the left sidebar.
2. On the **User list** page, find the target sub-account and choose **More** > **Add to group** in the **Operation** column.
3. On the **Add to group** page, select the target user group.
4. Click **OK**.

Logging In to the Sub-account for Verification

Log in to the Tencent Cloud Mesh console to verify that the features corresponding to the associated policies can be used. If they can be used, the sub-account was successfully authorized.

CAM Custom Policy Authorization

Last updated：2023-12-26 14:20:59

If you have custom permission management requirements, you can create a custom CAM policy and associate it with a sub-account to implement custom authorization. You can perform configuration based on actual service requirements by referring to the following description.

CAM Element Reference

Core elements of a CAM custom policy include: action, resource, condition, and effect.

1. Action

This required element describes allowed or denied actions. An action can be an API (described with a name prefix) or a feature set (a set of specific APIs, described with an actionName prefix). You can view CAM APIs accessed to Tencent Cloud Mesh.

2. Resource

This element describes specific data that is to be authorized. A resource is described in six paragraphs. You can view Tencent Cloud Mesh resource description.

3. Condition

This element describes the condition for the policy to take effect. A condition consists of operator, action key, and action value. A condition value may contain information such as time and IP address.

4. Effect

This required element describes whether the statement results in an **allow** or an explicit **deny**.

5. Custom policy sample

This policy defines that it is allowed to obtain details about two mesh instances mesh-abcd1234 and mesh-1234abcd in Guangzhou.

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "resource":[
                "qcs::tcm:gz:uin/1234567:mesh/mesh-abcd1234",
```

```
                "qcs::tcm:gz:uin/1234567:mesh/mesh-1234abcd"
            ],
            "action": [
                "name/tcm:DescribeMesh"
            ]
        }
    ]
}
```

For more information about syntax logic of CAM custom policies, see CAM Syntax Logic.

Tencent Cloud Mesh Resources That Can Be Authorized on CAM

Resource	Resource Description Method in Authorization Policy
Service mesh	`qcs::tcm:$region:$account:mesh/$meshid`

It includes the following fields:

`$region` : describes region information. It is an ID of a region. For example, `gz` is the ID of Guangzhou.

`$account` : describes root account information about a resource owner. It is expressed in the `uin/${uin}` format, for example, `uin/12345678` . If this field is left blank, it indicates the root account to which the CAM user who creates the policy belongs.

`$meshid` : describes mesh instance information. It is an ID of a mesh, or is set to `*` .

For information on how to describe resources in authorization policies, see Resource Description Method.

CAM APIs That Can Authorize Tencent Cloud Mesh

On CAM, you can authorize the following actions for Tencent Cloud Mesh mesh resources:

Mesh Instance

API	Description	Resource
CreateMesh	Creating a service mesh	Mesh resource `qcs::tcm:$region:$account:mesh/*`
DeleteMesh	Deleting a service mesh	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`
DescribeMesh	Obtaining a specified service mesh	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`

ListMeshes	Obtaining a service mesh list	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`
ModifyMesh	Modifying service mesh configurations	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`
UpgradeMesh	Upgrading a service mesh	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`

Istio Resource

API	Description	Resource
ForwardRequestRead	Reading Istio CRD resources	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`
ForwardRequestWrite	Writing Istio CRD resources	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`

Service Discovery

API	Description	Resource
LinkClusterList	Associating a cluster with a service mesh instance	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`
UnlinkCluster	Disassociating a cluster	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`

Gateway

API	Description	Resource
CreateIngressGateway	Creating an ingress gateway	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`
DeleteGatewayInstance	Deleting an ingress gateway	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`
DescribeIngressGatewayList	Querying an ingress	Mesh resource

	gateway list	`qcs::tcm:$region:$account:mesh/$meshid`
ModifyIngressGateway	Modifying an ingress gateway	Mesh resource `qcs::tcm:$region:$account:mesh/$meshid`

Sample Deployment

API	Description	Resource
CreateTrial	Creating Tencent Cloud Mesh sample deployment	Authorizing only interfaces `*`
DeleteTrial	Deleting Tencent Cloud Mesh sample deployment	Authorizing only interfaces `*`
RetryTrialTask	Retrying creating Tencent Cloud Mesh sample deployment	Authorizing only interfaces `*`

Extended Features
Using a Wasm Filter o Extend the Data Plane

Last updated：2023-12-26 14:21:21

Wasm is short for WebAssembly, which can compile binary instructions and load them into the Envoy's filter chain to extend mesh data plane capabilities. In this way, Envoy and extension components are decoupled, and users no longer need to extend capabilities by modifying Envoy code and compiling special Envoy versions. In addition, wasm delivers advantages of dynamic loading and secure isolation.

Since Istio 1.6, the Proxy-Wasm sandbox API has replaced Mixer as a main extension implementation of Istio to implement the interaction between Envoy and wasm virtual machines. Therefore, to extend Envoy through a wasm filter, you need to use Proxy-WASM SDK.

Usually, steps of compiling a wasm file to extend mesh data plane capabilities include the following:

1. Compile a wasm filter by following Examples.

2. Inject the wasm filter into a ConfigMap to mount the wasm filter to any workload through the ConfigMap, thereby preventing the wasm filter from being copied to multiple nodes.

```
kubectl create cm -n foo example-filter --from-file=example-filter.wasm
```

3. Mount the wasm filter to a service workload. You can use Istio Annotations to enable a corresponding file to be automatically mounted when creating a workload.

```
sidecar.istio.io/userVolume: '[{"name":"wasmfilters-dir","configMap": {"name": "exa
sidecar.istio.io/userVolumeMount: '[{"mountPath":"/var/local/lib/wasm-filters","nam
```

Apply the annotation to the corresponding workload.

```
kubectl patch deployment -n foo frontpage-v1 -p '{"spec":{"template":
{"metadata":{"annotations":{"sidecar.istio.io/userVolume":"
[{\\"name\\":\\"wasmfilters-dir\\",\\"configMap\\": {\\"name\\": \\"example-
filter\\"}}]","sidecar.istio.io/userVolumeMount":"
[{\\"mountPath\\":\\"/var/local/lib/wasm-filters\\",\\"name\\":\\"wasmfilters-
dir\\"}]"}}}}}'
```

4. Create an Envoy filter, and add the wasm filter to the Envoy filter chain of the corresponding workload to have it to take effect.

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: frontpage-v1-examplefilter
  namespace: foo
```

```
spec:
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      listener:
        filterChain:
          filter:
            name: envoy.http_connection_manager
            subFilter:
              name: envoy.router
    patch:
      operation: INSERT_BEFORE
      value:
        name: envoy.filters.http.wasm
        typed_config:
          '@type': type.googleapis.com/envoy.extensions.filters.http.wasm.v3.Wasm
          config:
            name: example-filter
            root_id: my_root_id
            vm_config:
              code:
                local:
                  filename: /var/local/lib/wasm-filters/example-filter.wasm
              runtime: envoy.wasm.runtime.v8
              vm_id: example-filter
              allow_precompiled: true
  workloadSelector:
    labels:
      app: frontpage
      version: v1
```

Till now, the wasm filter has been deployed. The wasm filter can also be used as an image. For details, see Build a wasm filter image. For details about how to use the wasme tool to deploy the wasm filter, see Deploying Wasm Filters with Wasme.

It can be seen that the deployment of a wasm filter is cumbersome, especially when large-scale deployment is required. It is difficult to deploy and manage a batch of wasm filters without a tool. Tencent Cloud Mesh provides convenient deployment tools, which can be used to deploy a batch of wasm filters in the binary or image format to services. For details, see Using Tencent Cloud Mesh Tools to Deploy Wasm Filters in Batches.