Application Service Mesh

User Guide

Issue	02
Date	2023-07-03

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Application Service Mesh

Application Service Mesh (ASM) is a service mesh platform developed based on Istio. It seamlessly interconnects with Cloud Container Engine (CCE), an enterprise-level Kubernetes cluster service. With better usability, reliability, and visualization, ASM provides you with out-of-the-box features and enhanced user experience.

ASM is a non-intrusive microservice governance solution that provides full-lifecycle management and traffic management. It is compatible with the Kubernetes and Istio ecosystems and provides a wide range of features such as load balancing, outlier detection, and rate limiting. ASM provides diversified built-in grayscale releases, including canary release and blue-green deployment, enabling one-stop automatic release management. It monitors and collects data in a non-intrusive manner to provide real-time traffic topology and tracing. It provides a panoramic view of service operations, enabling you to easily monitor and diagnose service performance.

For more about ASM, see **Service Overview**.

2 Buying a Service Mesh

2.1 Buying a Service Mesh

ASM allows you to buy a service mesh of the Basic edition, which is standard for commercial use. In a service mesh of the Basic edition, control plane components are deployed in a cluster for non-intrusive governance, grayscale releases, and traffic monitoring of services in the cluster. A service mesh of the Basic version supports Istio 1.8, 1.13, 1.15, and 1.18. It can manage only one cluster and a maximum of 200 pods.

Prerequisites

You have created a CCE cluster. If not, create one by referring to **Buying a CCE Cluster**.

Constraints

- ASM depends on the domain name resolution of CoreDNS. Before creating a service mesh for a cluster, ensure that the cluster has required resources and CoreDNS is running normally.
- Istio components v1.13 and v1.15 cannot run on nodes running CentOS or EulerOS 2.5. When creating a service mesh, do not specify these types of nodes as master nodes.
- When you enable Istio for a cluster, you must enable port 7443 in the inbound direction of the security group to which the worker node belongs, for automatic sidecar injection and callback. If you use the default security group created by CCE, this port is automatically enabled. If you create a security group rule, manually enable port 7443 to ensure that sidecars can be automatically injected.

Procedure

Step 1 Log in to the ASM console.

Step 2 Click Buy Mesh in the upper right corner.

Step 3 Configure the following parameters.

- **Mesh Edition**

  Only service meshes of the Basic edition are supported.

- **Mesh Name**

  Enter a service mesh name, which consists of 4 to 64 characters. It must start with a lowercase letter and cannot end with a hyphen (-). Only lowercase letters, digits, and hyphens (-) are allowed.

  Service mesh names under the same account must be unique and cannot be modified after creation.

- **Istio Version**

  Select the Istio version supported by the service mesh.

- **Enable IPv6**

  Conditions for enabling IPv4/IPv6 dual stack for a service mesh

Service Mesh Edition	Istio Version	Cluster Type	Cluster Network Type	Remarks
Basic edition	1.18 or later	CCE Turbo clusters	Cloud native network 2.0	To enable IPv6, see **Creating an IPv4/IPv6 Dual-Stack Cluster in CCE**.

  NOTE

  - **Enable IPv6** is only available in Istio 1.18 or later.
  - IPv4/IPv6 dual stack cannot be enabled for a service mesh that is upgraded to v1.18 or later.
  - Once IPv4/IPv6 dual stack is enabled for a service mesh, it cannot be disabled. IPv4/IPv6 dual stack cannot be enabled for an existing service mesh.

- **Cluster**

  Select the target cluster from the cluster list or enter the target cluster name in the upper right corner of the list to search for it. You can select only the clusters which versions are supported by the current mesh version.

- **Mesh Control Plane Node**

  The control plane components of the service mesh of the Basic edition are installed in the user cluster. Therefore, you need to select the node for installing the service mesh control plane. If HA is required, you can select two or more nodes from different AZs.

  The selected node is labeled with **istio:master**, and the components are scheduled to this node.

- **Observability Configuration**
  - **Application Metrics**

    If this option is enabled, you can specify service access metrics, application topologies, and service health and SLO definitions in the service mesh.

NOTE

Istio earlier than 1.18 can only work with Application Performance Management (APM) to monitor application metrics. Istio 1.18 or later can only work with Application Operations Management (AOM) to monitor application metrics. For details about how to view application metrics in AOM, see **How Do I Query Application Metrics in AOM?**

- **Access Logging**

  If this option is enabled, you can query inter-service access records in the service mesh to locate exceptions. After enabling this option, you need to select the Log Tank Service (LTS) log group and log stream. Access logs will be transmitted to the log stream. You can view the access logs on the **Monitoring Center** > **Access Logs** page.

  NOTE

  Only Istio 1.18 or later can work with LTS to collect and store access logs. To ensure logs are reported to LTS, install CCE Log-Agent on the **Add-ons** page in advance.

- **Tracing**

  - **Sampling Rate**: Number of requests generated by the tracing service/ Total number of requests

  - **Version**: tracing service. If you select **Third-party Jaeger/Zipkin service**, you need to set **Service Address** and **Service Port**, which indicate the address and port number used by the third-party tracing service to receive requests.

  NOTE

  - Only Istio 1.15 or later support the third-party tracing service.
  - If you want to use the third-party Jaeger or Zipkin service, install it first. You can also install it by referring to **How Do I Connect a Service Mesh to Jaeger or Zipkin for Viewing Traces?** Then, obtain the service address.
  - The default service ports of Jaeger and Zipkin are both 9411. If you customize the service port during Jaeger or Zipkin installation, replace **Service Port** with the actual value.

Step 4 (Optional) Configure advanced settings.

- **Namespace Injection Settings**

  Select a namespace and label it with **istio-injection=enabled**. All pods in the namespace will be injected with an istio-proxy sidecar.

  You can inject a sidecar in **Mesh Configuration** > **Sidecar Management** after the mesh is created. For details, see **Injecting a Sidecar**.

- **Restart Existing Services**

   : Pods of the existing services in the namespace will be restarted, which will temporarily interrupt your services. The **istio-proxy** sidecar is automatically injected into the pods of the existing services.

   : The **istio-proxy** sidecar cannot be automatically injected into the pods of the existing services. You need to manually restart the workloads on the CCE console to inject the sidecar.

- **Traffic Interception Settings**

  NOTE

  By default, sidecars intercept all inbound and outbound traffic of pods. You can modify the default traffic rules in **Traffic Interception Settings**.

  Inbound Ports: Inbound ports separated by commas (,). You can use this field to specify the ports that will be included or excluded for inbound traffic redirection.

  - **Include only specified ports** means that the traffic to services in a service mesh over specified ports will be redirected to the sidecar.
  - **Exclude only specified ports** means that the traffic to services in a service mesh over the ports except the specified ports will be redirected to the sidecar.

  Outbound Ports: Outbound ports separated by commas (,). You can use this field to specify the ports that will be included or excluded for outbound traffic redirection.

  - **Include only specified ports** means that the traffic from services in a service mesh over specified ports will be redirected to the sidecar.
  - **Exclude only specified ports** means that the traffic from services in a service mesh over the ports except the specified ports will be redirected to the sidecar.

  Outbound IP Ranges: IP address ranges separated by commas (,) in CIDR format. You can use this field to specify the IP ranges that will be included or excluded for outbound traffic redirection.

  - **Include only specified IP ranges** means that the traffic from specified IP ranges will be redirected to the sidecar.
  - **Exclude only specified IP ranges** means that the traffic from IP ranges except the specified IP ranges will be redirected to the sidecar.

- **Resource Tags**

  Enter the tag key and tag value. A maximum of 20 tags can be added.

Step 5 Review the service mesh configuration in **Configuration List** on the right of the page and click **Submit**.

It takes about 1 to 3 minutes to create a service mesh. If the service mesh status changes from **Installing** to **Running**, the service mesh is successfully created.

NOTE

An otel-collector workload is automatically created when a service mesh is created. For details, see **otel-collector Functions**.

When the service mesh is enabled, the following operations are performed:

- Helm orchestrates the application into a Release as the resource of the service mesh control plane.
- A security group is enabled for the nodes to allow the inbound traffic for port 7443 to support automatic sidecar injection.

----End

2.2 Comparison Among Basic and Community Editions

Category	Function	Item	Community Edition	Basic Edition
Specification	Management scale	Maximum number of pods a mesh can manage.	-	200
Basic functions	Service discovery and registration	The service center cluster obtains the service list, automatically registers containerized services, and refreshes service pod statuses. You do not need to implement the registration and subscription logic as services can be automatically discovered, and dynamically managed.	√	√
	Multiple service versions	Supports version-based service management and monitoring.	-	√
		Supports version-based service workload management.	√	√
	Multiple service ports	Supports multi-port, multi-protocol service management.	√	√
		Supports multi-port grayscale release.	-	√
	Multiple service types	Supports containerized service backend.	√	√
	Protocols and languages	HTTP and gRPC grayscale release, governance, and monitoring (varying with protocols), language- and framework-agnostic, and non-intrusive	√	√
	Application gateway	Supports layer-4 and layer-7 external access, entry path mapping, TLS termination at gateways, and configuration of external certificates and secrets.	√	√

Category	Function	Item	Community Edition	Basic Edition
	Load balancing	Supports round robin, random, minimum number of connections, and consistent hashing algorithms based on a specific HTTP header or cookie value.	√	√
	Fault injection	Supports injection of faults with a specified delay or error and configuration of fault percentage.	√	√
	Outlier detection	Supports layer-7 request management, including the maximum number of requests, maximum number of requests per connection, maximum number of queued requests, and maximum number of retries; layer-4 connection management, including the maximum number of connections and connection timeout interval; exception check, and automatic isolation and recovery of faulty pods.	√	√
	Traffic types	Supports governance of internal communication between services and external access to services (ingress traffic).	√	√
	Running environments	Supports containerized application management.	√	√
	Authentication	Supports non-intrusive two-way TLS authentication and channel encryption.	√	√
	Authorization	Supports service access authorization management.	√	√
	Grayscale release	Supports grayscale traffic distribution policies based on the browser, operating system, custom HTTP header, cookie content, and URL; and grayscale release based on request parameters and traffic ratio.	√	√

Category	Function	Item	Community Edition	Basic Edition
		Supports canary release template.	-	√
		Supports blue-green deployment template.	-	√
		Supports monitoring of service status during grayscale release to facilitate decision making.	-	√
		Supports monitoring of service requests during grayscale release to facilitate decision making.	-	√
		Supports dynamic configuration of the number of service pods during grayscale release.	-	√
		Supports dynamic traffic ratio monitoring during grayscale release.	-	√
	Application topology	Provides global topology of traces in applications.	-	√
		Provides key metrics such as the number of total requests and abnormal requests in topology.	-	√
		Supports real-time application topology.	-	√
	Tracing	Supports non-intrusive tracing point setting.	√	√
	Monitoring metrics	Supports monitoring of the running data of service pod CPUs, memory, and disks, monitoring of access metrics such as RPS and latency, analyzing statistics of access and abnormal metrics, interconnection with open source metric components such as Prometheus, and interconnection with different metric backends.	√	√
	Access logs	Supports non-intrusive collection of access logs.	√	√

Category	Function	Item	Community Edition	Basic Edition
	Installation	Supports enabling Istio for existing and newly created Kubernetes clusters in a few clicks.	-	√
	Upgrade	Supports smooth upgrade of the control plane without interrupting application services.	√	√
		Supports smooth upgrade of the data plane without interrupting application services.	√	√
	Add-on management	Supports installation of community add-ons, such as Grafana and Prometheus in a few clicks.	-	√
		Supports installation of Tracing in a few clicks.	-	√
		Supports installation of Kiali in a few clicks.	-	√
		Supports installation of ELK in a few clicks.	-	√
	Agent management	Supports transparent and iptables-based traffic interception, as well as automatic proxy injection at namespace and workload levels.	√	√
	Proxy mode	Supports a sidecar in each pod.	√	√
	Command line tool	Supports traffic policy management using command line tools, such as istioctl and kubectl.	-	√

3 Mesh Management

3.1 Service Mesh Events

Scenarios

ASM supports the event center, which allows you to query details about important operations such as service mesh creation and deletion and gateway creation and deletion.

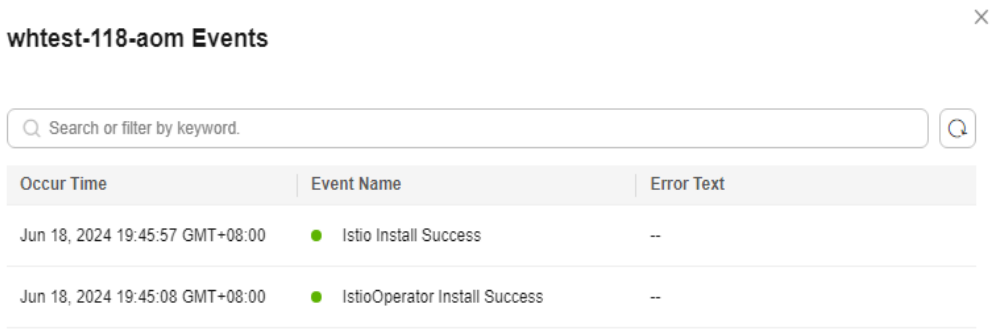
NOTE

You can view events in a service mesh of the Basic edition (1.8 or later) in ASM 1.0.

Procedure

Step 1 Log in to the **ASM console** and search for the service mesh of the Basic edition by edition.

Step 2 Click in the upper right corner. In the window that slides out from the right, view service mesh events.



----End

3.2 Uninstalling a Service Mesh

Scenarios

If you no longer need a service mesh, you can uninstall it.

Constraints

- To uninstall a service mesh in which a grayscale release task is running, you need to complete the grayscale release first.
- You need to ensure available nodes exist in the clusters for running the cleanup task to avoid uninstallation failure.

Procedure

Step 1 Log in to the ASM console.

Step 2 Click  in the service mesh.

Step 3 On the dialogue box displayed, select whether to restart existing services and read the precautions.

By default, existing services are not restarted during the uninstallation. The injected istio-poxy sidecar is removed only after the existing services are restarted. If you want to restart the services, select **Yes**. Restarting the services will interrupt your services temporarily.

NOTE

You are advised to restart existing services to avoid the following exceptions: If the cluster enables the current mesh again after it is uninstalled, gateway access failed.

- Uninstalling a service mesh will uninstall its control plane components and data plane sidecars.
- After the uninstallation, service gateways of applications cannot be used. Configure Services for external access to applications.

  To update the external access mode, log in to the CCE console and choose **Resource Management** > **Network** > **Services**.
- Uninstalling a service mesh will delete the labels of the Istio exclusive nodes, but the Istio-master node will not be automatically deleted. You can delete it on the CCE console.

  To view node information, log in to the CCE console and choose **Resource Management** > **Nodes**.

----End

4 Service Management

4.1 Configuration Diagnosis

ASM diagnoses all services in a managed cluster. Traffic management, traffic monitoring, and grayscale release are available only for normal services.

Constraints

- If multiple services correspond to one deployment, these services cannot be added to the mesh. Otherwise, functions such as grayscale release or gateway access may fail.
- If the workload of a service uses the host network mode (**hostNetwork: true** is configured for the pod), sidecars cannot be injected for the service.

Service Diagnosis

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Service Management**. The diagnosis results of services are displayed in the **Configuration Diagnosis Result** column.

If a service is abnormal, click **Fix** to fix the issues. For details, see **Service Issue Fixing**.

Figure 4-1 Service diagnosis

Step 3 After the issues are fixed, you can click **Diagnose Again** to diagnose the service again.

----End

Service Issue Fixing

If a service is abnormal, you need to manually fix the abnormal items and then perform auto fix for left issues.

Step 1 Click **Fix** in the row of the abnormal service. If there are issues to be fixed manually, click **View Solution** to see how to fix them.

Step 2 Click **Next** to go to the auto fix page, and click **Auto Fix** to automatically fix left issues.

NOTE

- If left issues cannot be fix automatically, click **View Solution** and fix them manually.
- Auto fix does not support Services which have configured gateways or have created grayscale release tasks.
- If the service is not displayed in the service list, check whether the corresponding workload exists.

----End

4.2 Manual Fixing Items

4.2.1 All Pods Have the app and version Labels

Description

All pods of a Service must be labeled with **app** and **version**. **app** traces traffic in traffic monitoring, and **version** distinguishes different versions in grayscale release. If a pod is not labeled with **app** or **version**, this item is abnormal.

Rectification Guide

The labels of pods are configured in **spec.template.metadata.labels** of the Deployment. The recommended configuration is as follows:

```
labels:
  app: {serviceName}
  version: v1
```

CAUTION

Modifying or deleting the Deployment will trigger pod rolling upgrade, which may cause temporary service interruption. Therefore, perform the operation at a proper time.

Step 1 Copy the original workload configuration and save it as a YAML file.

kubectl get deployment {deploymentName} **-n** {namespace} **-o yaml >** {deploymentName}**-deployment.yaml**

For example:

kubectl get deployment productpage -n default -o yaml > productpage-deployment.yaml

Step 2 Modify the **productpage-deployment.yaml** file. If the file does not contain **app** and **version**, add them. You are advised to set **app** to the Service name and the **version** to **v1**.

Step 3 Delete the original workload.

kubectl delete deployment {oldDeploymentName} **-n** {namespace}

Step 4 Apply the new workload configuration.

kubectl apply -f productpage-deployment.yaml

----End

NOTE

For clusters of v1.15 or earlier, you can modify pod labels on the CCE console, but residual ReplicaSets may exist.

To check whether there are residual ReplicaSets, perform the following steps:

1. Query the ReplicaSets of the Deployment.

   kubectl get replicaset | grep {deploymentName}

2. Find the ReplicaSets containing more than one pod. These ReplicaSets may be residual after label modification. You need to delete the old ReplicaSets.

   kubectl delete replicaset {replicaSetName} **-n** {namespace}

4.2.2 All Pods Share the Same app and version Labels

Description

All pods of a Service must share the same **app** and **version** labels. **app** traces traffic in traffic monitoring, and **version** distinguishes different versions in grayscale release. If pods with different **app** or **version** labels exist, this item is abnormal.

Rectification Guide

The labels of pods are configured in **spec.template.metadata.labels** of the Deployment. The recommended configuration is as follows:

```
labels:
  app: {serviceName}
  version: v1
```

To modify the labels of multiple pods to the same value, perform the following steps:

Step 1 View the labels configured for **spec.selector**.

kubectl get svc {serviceName} **-o yaml**

For example, the labels are **app: ratings** and **release: istio-bookinfo**.

Step 2 Search for the pods of a Service by label.

kubectl get pod -n {namespace} **-l app=ratings,release=istio-bookinfo**

NOTE

{namespace} is the same as the namespace of the Service.

Step 3 Find the workload of a pod by the pod name.

kubectl get deployment {deploymentName} **-n** {namespace}

NOTE

- Generally, the pod name is in the format of **{deploymentName}-{random character string}-{random character string}**.
- If no workload of a pod is found by the pod name, you need to delete residual ReplicaSets.

  To check whether there are residual ReplicaSets, perform the following steps:

  1. Query the ReplicaSets of the Deployment.

     kubectl get replicaset | grep {deploymentName}

  2. Find the ReplicaSets containing more than one pod. These ReplicaSets may be residual after label modification. You need to delete the old ReplicaSets.

     kubectl delete replicaset {replicaSetName} **-n** {namespace}

Step 4 For details about how to modify the **app** and **version** labels of a pod, see **Rectification Guide**.

----End

4.2.3 All Pods Have Sidecars Injected

Description

An **istio-proxy** container must exist in all pods of a Service. Otherwise, this item is abnormal.

Rectification Guide

Step 1 Log in to the ASM console and click the name of the service mesh that the Service is added to. Choose **Mesh Configuration** in the navigation pane, click the **Sidecar Management** tab, and check whether a sidecar is injected into the namespace that the Service belongs to.

- If no, go to **Step 2**.
- If yes, go to **Step 3**.

Step 2 Inject a sidecar.

You can inject sidecars for pods of all workloads in the namespace. For details, see **Injecting a Sidecar**. You can also inject sidecars for a workload as follows:

1. Label the namespace where the workload is located with **istio-injection=enabled**.

   kubectl label ns <namespace> **istio-injection=enabled**

2. Add the **annotations** field for the workload on the CCE console.

```
annotations:
  sidecar.istio.io/inject: 'true'
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
      version: v1
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: httpbin
        version: v1
      annotations:
        sidecar.istio.io/inject: 'true'
```

For more details about sidecar injection, see **Installing the Sidecar**.

Step 3 If namespace injection is enabled for the cluster but no sidecar is injected into the pod, you need to manually restart the pod on the CCE console as follows:

On the CCE console, choose **More** > **Redeploy** in the **Operation** column of the target workload.

Step 4 Check whether the host network mode is configured for the workload as follows:

On the CCE console, choose **More** > **Edit YAML** in the **Operation** column of the target workload, and check whether **spec.template.spec.hostNetwork: true** is configured. If yes, check whether this field can be deleted or set to **false**. Otherwise, sidecars cannot be injected.

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
      version: v1
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
        version: v1
    spec:
      hostNetwork: true
      containers:
        - name: container-1
          image: nginx:alpine
```

Step 5 Check whether the number of pods exceeds the service mesh scale.

If the number exceeds 200, the excess pods cannot be injected with sidecars.

----End

4.3 Auto Fixing Items

4.3.1 The Service Port Name Complies with the Istio Specifications

Description

The Service port name must contain the specified protocol and prefix and must be in the following format:

```
name: <protocol>[-<suffix>]
```

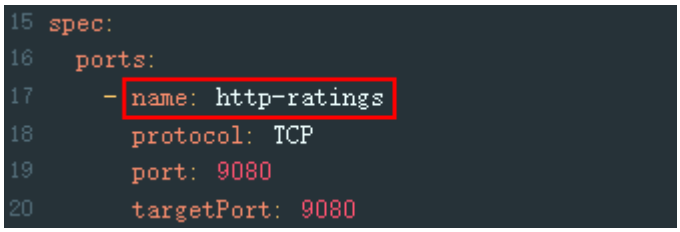
<protocol> can be **http**, **tcp**, or **grpc**. Istio provides routing capabilities based on protocols defined on ports. For example, **name: http-service0** and **name: tcp** are valid port names, while **name: httpforecast** is not.

If the Service port name is invalid, this item is abnormal.

Rectification Guide

Step 1 Log in to the CCE console and click the cluster name to go to the cluster console.

Step 2 In the navigation pane on the left, choose **Resources** > **Networking**, select the target service, click **More** > **Edit YAML**, view its Service protocol, and add a protocol type before the service name as shown in the following figure.



Step 3 Click **OK**.

----End

4.3.2 The Service Selector Cannot Contain version Labels

Description

The **spec.selector** of a Service cannot be labeled with **version**. Otherwise, this item is abnormal.

Rectification Guide

Step 1 Log in to the CCE console and click the cluster name to go to the cluster console.

Step 2 In the navigation pane on the left, choose **Resources** > **Networking**, select the target Service, click **More** > **Edit YAML**, view its **spec.selector**, and delete the **version** label.


```
spec:
  ports:
    - name: http-service0
      protocol: TCP
      port: 8000
      targetPort: 80
  selector:
    app: nginx
    version: v1
```

----End

4.3.3 The Service Is Configured with a Default-version Route and The Route Configuration Is Correct

Description

Istio defines service traffic routing rules in **VirtualService** and **DestinationRule**. Therefore, you need to configure **VirtualService** and **DestinationRule** for each service. The following rules must be met:

- All ports of a Service must be configured in **VirtualService**.
- The protocol type in **VirtualService** must be the same as that of the ports of a Service.
- The default service version must be configured in **VirtualService** and **DestinationRule**.

NOTE

If the check result changes, the port number or port name of a Service may be changed.

Rectification Guide

Step 1 Log in to the ASM console. Select the mesh where the service is located. In the navigation pane on the left, choose **Mesh Configuration**, click **Istio Resource Management**, and select **Istio resources: virtualservices** and the namespace to which the service belongs.

Step 2 Ensure that all ports of the Service are configured in **VirtualService**.


```
spec:
  hosts:
    - reviews
  http:
    - match:
        - gateways:
            - mesh
          port: 9080
      route:
        - destination:
            host: reviews.default.svc.cluster.local
            port:
              number: 9080
            subset: v1
          weight: 50
        - destination:
            host: reviews.default.svc.cluster.local
            port:
              number: 9080
            subset: v2
          weight: 50
```

Step 3 Ensure that the protocol type in **VirtualService** is the same as that of the ports of the Service.

Figure 4-2 Protocol type in VirtualService

```
34 spec:
35   hosts:
36     - ratings
37   http:
38     - route:
39         - destination:
40             host: ratings
41             port:
42               number: 9080
43             subset: v1
```

Figure 4-3 Port protocol type of the Service

```
13 spec:
14   ports:
15     - name: http-ratings
16       protocol: TCP
17       port: 9080
18       targetPort: 9080
19   selector:
20     app: ratings
```

----End

5 Gateway Management

5.1 Adding a Gateway

A gateway enables unified entry, traffic management, security, and service isolation.

Prerequisites

Gateways use load balancers of ELB to provide network access. Before adding a gateway, you need to create a load balancer.

When creating a load balancer, you need to ensure that it belongs to the same VPC as the cluster. For details, see **Creating a Shared Load Balancer**.

Procedure

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Gateway Management** and click **Add Gateway**.

Step 3 Configure the following parameters.

- **Gateway Name**

  Enter a gateway name. Enter 4 to 59 characters starting with a lowercase letter and ending with a lowercase letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.

- **Cluster**

  Select the cluster to which the gateway belongs.

- **Access Mode**
  - **IP Version**: The value can be **IPv4** or **DualStack**. This parameter is available only when IPv6 is enabled.
  - Gateways use shared and dedicated load balancers of ELB for the access over both public and private IPv4 networks.

- **Access Entry**

- **External Protocol**

  Select one to match the protocol type of your service. **HTTP**, **gRPC**, **TCP**, **TLS**, and **HTTPS** are supported.

- **External Port**

  Enter the port number exposed in the Load Balancer Service address. The port number can be specified randomly.

- External Access Address

  The system automatically fills in the IP address of the load balancer as the service access entry. You can also change the IP address to the domain name associated with the load balancer.

- **TLS Termination**

  If **External Protocol** is **HTTPS**, **TLS Termination** is enabled and cannot be disabled.

  If **External Protocol** is **TLS**, you can enable or disable **TLS Termination**. If you enable TLS termination, bind a certificate to support TLS-based data transmission encryption and authentication. If you disable TLS termination, encrypted TLS data will be directly forwarded.

- **Secret Certificate**

  - When configuring a TLS protocol with TLS termination enabled, you need to bind a certificate to support TLS-based data transmission encryption and authentication.
  - When configuring the HTTPS protocol, you need to bind a secret certificate.

- **Earliest TLS Version Supported/Latest TLS Version Supported**

  When configuring a TLS protocol with TLS termination enabled or an HTTPS protocol, you can select the earliest and latest TLS versions.

Step 4 (Optional) Configure routing parameters.

When the access address of a request matches the forwarding policy (which consists of an external access address and URL), the request is forwarded to the corresponding target Service for processing. Click +. The **Add Route** dialog box is displayed.

- **URL Matching Rule**
  - **Prefix**: A URL can be accessed if its prefix is the same as that you configure. For example, **/healthz/v1** and **/healthz/v2**.
  - **Exact**: Only the URL that fully matches the values you set can be accessed. For example, if the URL is set to **/healthz**, only **/healthz** can be accessed.
- **URL**

  Mapping URL supported by the service, for example, **/example**.
- **Namespace**

  Select the namespace to which the gateway belongs.
- **Target Service**

  Service of the gateway. Select a value from the drop-down list box. The target service is filtered based on the corresponding gateway protocol. For details

about the filtering rules, see **Why Cannot I Select the Corresponding Service When Adding a Route?**

The service which configuration diagnosis fails cannot be selected. You need to fix the issues first. For details, see **Manual Fixing Items** or **Auto Fixing Items**.

- **Access Port**

  Only ports that match external protocols are displayed.

- **Rewrite**

  (This parameter is configurable when the external protocol is HTTP.)

  Rewrite the HTTP URI and host/authority header before forwarding. Disabled by default. To enable it, configure the following parameters:

  – URI: This value is used to rewrite the URI or prefix.

  – Host/Authority Header: This value is used to rewrite the HTTP host/authority header.

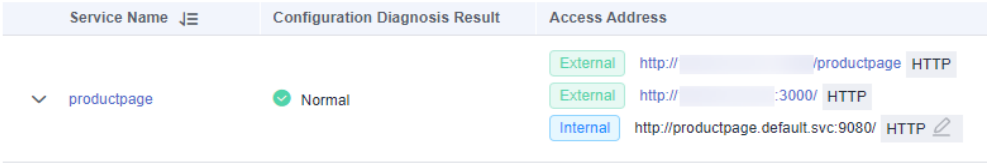
- **Specify Domain Name**

  Configure a routing rule for a domain name of the gateway.

Step 5 Click **OK**.

You can obtain the external network access address of the service in the **Service Management** page.

Figure 5-1 External network access address of the service



----End

5.2 Adding a Route

Scenarios

You can add multiple routes and configure multiple forwarding policies for a created gateway.

Procedure

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Gateway Management**, select the target gateway, click **Add Route** in the **Operation** column, and configure the following parameters:

- **URL Matching Rule**

  – **Prefix**: A URL can be accessed if its prefix is the same as that you configure. For example, **/healthz/v1** and **/healthz/v2**.

  - **Exact**: Only the URL that fully matches the values you set can be accessed. For example, if the URL is set to **/healthz**, only **/healthz** can be accessed.

- **URL**

  Mapping URL supported by the service, for example, **/example**.

  NOTE

  The URLs of the same gateway must be unique.

- **Namespace**

  Select the namespace to which the gateway belongs.

- **Target Service**

  Service of the gateway. Select a value from the drop-down list box. The target service is filtered based on the corresponding gateway protocol. For details about the filtering rules, see .

  The service which configuration diagnosis fails cannot be selected. You need to fix the issues first. For details, see **Manual Fixing Items** or **Auto Fixing Items**.

- **Access Port**

  Only ports that match external protocols are displayed.

- **Rewrite**

  (This parameter is configurable when the external protocol is HTTP.)

  Rewrite the HTTP URI and host/authority header before forwarding. Disabled by default. To enable it, configure the following parameters:

  - URI: This value is used to rewrite the URI or prefix.
  - Host/Authority Header: This value is used to rewrite the HTTP host/authority header.

- **Specify Domain Name**

  Configure a routing rule for a domain name of the gateway.

Step 3 Click **OK**.

----End

6 Grayscale Release

6.1 Grayscale Release Overview

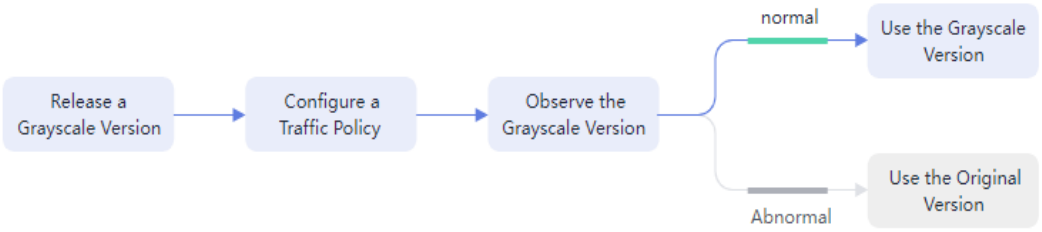
When switching between old and new services, you may be challenged in ensuring the system service continuity. If a new service version is directly released to all users at a time, it can be risky because once an online accident or bug occurs, the impact on users is great. It could take a long time to fix the issue. Sometimes, the version has to be rolled back, which severely affects user experience.

Several release policies are developed for service upgrade: canary release, blue-green deployment, A/B testing, rolling upgrade, and batch suspension of release. Traffic loss or service unavailability caused by releases can be avoided as much as possible. Currently, ASM supports canary release and blue-green deployment.

Canary Release

Canary release is also called grayscale release. It is a smooth iteration mode for version upgrade. During the upgrade, some users use the new version, while other users continue to use the old version. After the new version is stable and ready, it gradually takes over all the live traffic. In this way, service risks brought by the release of the new version can be minimized, the impact of faults can be reduced, and quick rollback is supported.

Figure 6-1 Canary release process

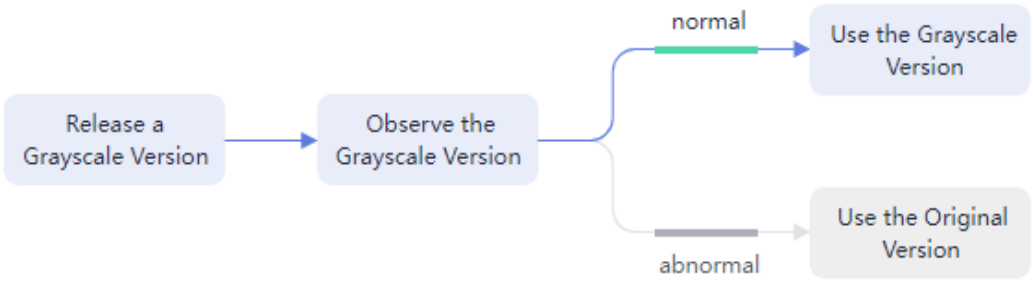


Blue-Green Deployment

Blue-green deployment provides a zero-downtime, predictable manner for releasing applications to reduce service interruption during the release. A new

version is deployed while the old version is retained. The two versions are online at the same time. The new and old versions work in hot backup mode. The route weight is switched (0 or 100) to enable different versions to go online or offline. If a problem occurs, the version can be quickly rolled back.

Figure 6-2 Blue-green deployment process



6.2 Creating a Grayscale Release Task

Basic Concepts

- Grayscale version

  Only one grayscale version can be released for a service. You can configure grayscale policies for the version.

- Grayscale policy

  Before releasing a new service version in the production environment and letting it serve all the live traffic, you can add a grayscale version and configure grayscale policies to serve just a proportion of the traffic. After the grayscale version has run stably for a period, it can serve as the default version to take over all traffic in place of the original version in the production environment.

Creating a Grayscale Release Task

Step 1 Log in to the ASM console and go to the **Create a grayscale release task** page by one of the following ways:

- (Shortcut) In the upper right corner of an Enterprise mesh, click .
- (Shortcut) In the center of the target mesh, click **Create a grayscale release task**.
- Create a grayscale release task on the mesh details page.
  a. Click the target mesh and go to its details page, click **Grayscale Release** in the navigation pane on the left.
  b. If no grayscale task is running, click **Create Release Task** in the **Canary Release** or **Blue-Green Deployment** area. If there is an ongoing grayscale task, click **Grayscale Release** in the upper right corner.

Step 2 Configure basic information of the grayscale release task.

- **Grayscale Release Form**

Select **Canary Release** or **Blue-Green Deployment** as required. For details about the differences between the two forms, see **Grayscale Release Overview**.

- **Task Name**

  Customize a grayscale release task name. Enter 4 to 63 characters, starting with a lowercase letter and ending with a letter or digit. Only lowercase letters, digits, and hyphens (-) are allowed.

- **Namespace**

  Select the namespace to which the service belongs.

- **Service**

  Select the service to be released from the drop-down list box. Services that are running grayscale tasks cannot be selected. They are automatically filtered out from the list.

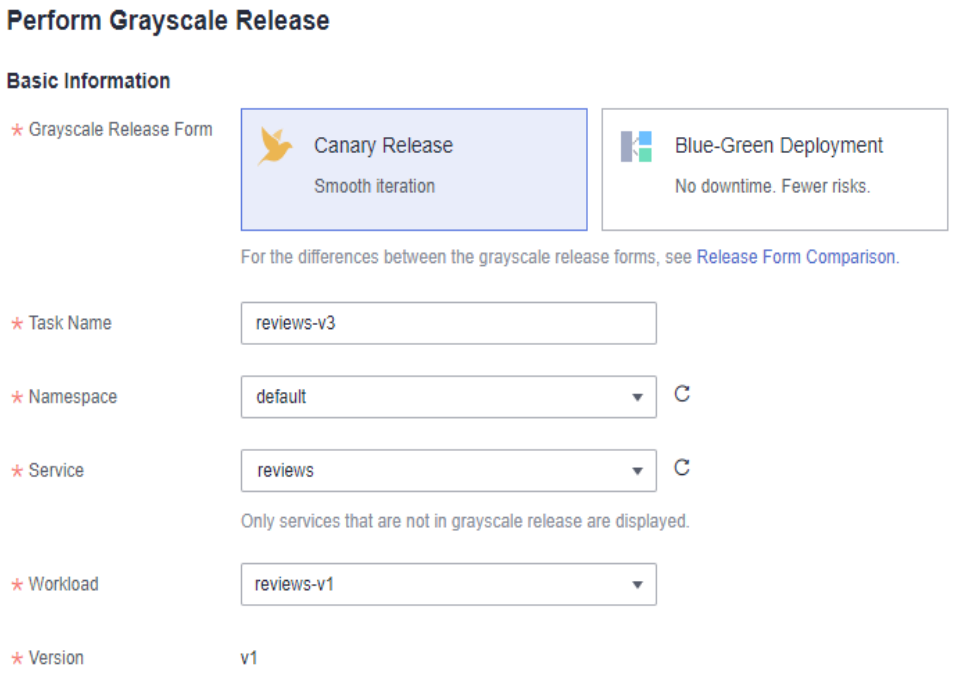
- **Workload**

  Select the workload to which the service belongs.

- **Version**

  Current service version number, which cannot be changed.

Figure 6-3 Basic information



Step 3 Configure grayscale version information.

- **Cluster**

  Select the cluster on which the grayscale version of the service will be deployed.

- **Version**

  Enter the grayscale version number of the service.

- **Pods**

Number of pods of the grayscale version. You can modify the number as required. Each pod of the grayscale version consists of containers deployed with the same image.

- **Image Name**

  The image of the service is selected by default.

- **Image Tag**

  Select the image tag of the grayscale version.

- **Custom Image**

  Configure a local or third-party image path for the grayscale release. The custom path must ensure that the image can be pulled.

Step 4 Click **Release**. Wait for the grayscale version to be created.

Ensure that all pods of the grayscale version are running normally and configure the traffic policy when the startup progress reaches 100%. You can view the pod monitoring information, including **Start Logs** and **Performance Monitoring** on the **View Status** page.

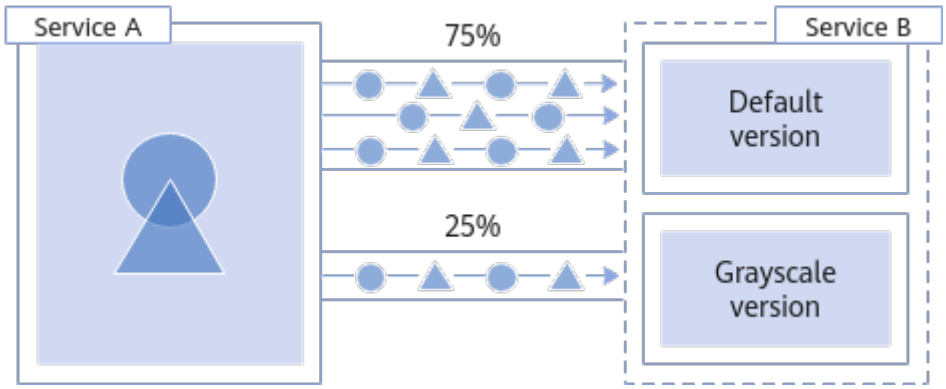
Step 5 (For canary release only) Click **Configure Traffic Policy** to configure a traffic policy.

Policy Type: The value can be **Based on traffic ratio** or **Based on request content**.

- **Based on traffic ratio**

  A specified ratio of traffic will be directed to the grayscale version. For example, 75% of the traffic is directed to the original version, and 25% is directed to the grayscale version. In actual applications, you can gradually increase the traffic ratio of the grayscale version and deliver policies to monitor the performance of the grayscale version.

  Figure 6-4 Based on traffic ratio

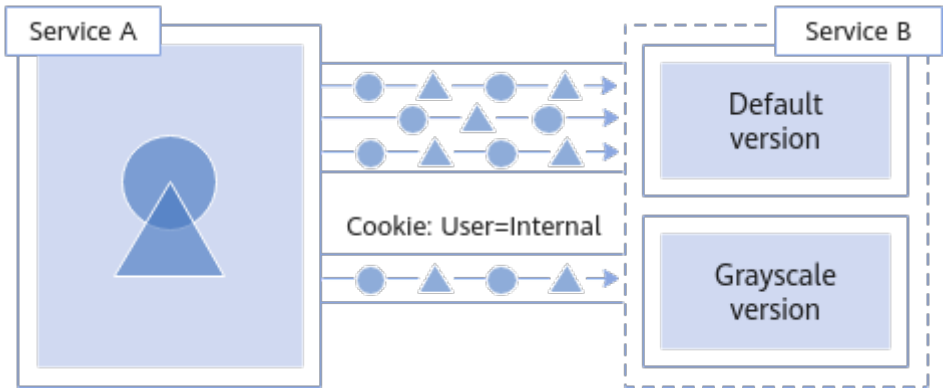
  Traffic ratio: You can set the traffic ratio for the original version and grayscale version. The system distributes traffic to the two versions based on the specific traffic ratio.

- **Based on request content**

  The grayscale version can be accessed only when the traffic meets the rules based on the cookies, custom headers, queries, operating systems, and browsers. For example, only HTTP requests whose cookies meet

User=Internal can be forwarded to the grayscale version. Other requests are still received by the original version.

Figure 6-5 Based on request content



- **Cookie**

  Regular expression: When the cookie of a request matches the configured regular expression, the request will be distributed to the grayscale version.

- **Header**

  - **Full match**: Only the URL that fully matches the values you set can be accessed. For example, if **Key** is set to **User** and **Value** is set to **Internal**, only requests whose headers contain **User** with the value **Internal** are responded by the service of the grayscale version.
  - **Regular expression**: When the header of a request matches the configured regular expression, the request will be distributed to the grayscale version.

    You can customize the key and value for filtering. The value supports the full match and regular expression.

- **Query**

  - **Full match**: Only the URL that fully matches the values you set can be accessed. For example, if **Key** is set to **User** and **Value** is set to **Internal**, only requests whose queries contain **User** with the value **Internal** are responded by the service of the grayscale version.
  - **Regular expression**: When the query of a request matches the configured regular expression, the request will be distributed to the grayscale version.

    You can customize the key and value for filtering. The value supports the full match and regular expression.

- **Allowed OS**: Select OSs that can access the grayscale version, including iOS, Android, Windows, and macOS.
- **Allowed Browser**: Select browsers that can access the grayscale version, including Chrome and Internet Explorer.
- **Traffic management YAML**: The rule YAML is automatically generated based on the configured parameters.

NOTE

A traffic policy based on request content is valid only for the entry service that is directly accessed. If you want the traffic policy to be applied to all services, the header information of HTTP requests needs to be transferred in the service code.

For example, if you configured a grayscale policy based on the request content for service **reviews** and did not transfer the HTTP request header information in the service code, the grayscale policy will not take effect when you send requests to service **productpage**.

The reason is that when the **productpage** service calls the **reviews** service, the header information of the HTTP request you sent to **productpage** will be lost. As a result, the **reviews** service receives a request without the header information. The grayscale policy will not take effect.

Step 6 Click **Deliver Policy**.

It takes several seconds for a grayscale policy to take effect. You can view the traffic monitoring of the service and the health monitoring of the original version and grayscale version.

----End

6.3 Basic Operations on a Grayscale Task

Description

Basic operations on a grayscale version are performed by modifying the configuration of the DestinationRule and VirtualService resources of Istio. After the modification is complete, wait for about 10 seconds for the new policy to take effect.

Modifying the Traffic Policy of a Grayscale Version

Modifying a grayscale policy that is based on traffic ratio

For such a grayscale policy that is based on traffic ratio, you can gradually increase the traffic ratio of the grayscale version to avoid service risks caused by direct traffic switchover. To change the traffic ratio, perform the following steps:

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Grayscale Release**. Then click the target canary release task.

Step 3 On the **Configure Traffic Policy** page, set the traffic ratio of the grayscale version.

If the traffic ratio of the grayscale version is set to **x**, the traffic ratio of the original version is automatically adjusted to **100-x**.

Step 4 Click **Deliver Policy**.

----End

Modifying a grayscale policy that is based on request content

With such a policy, a grayscale version can be accessed only when the traffic meets the rules based on Cookies, Headers, Queries, Allowed Operating Systems,

and Allowed Browsers. In real-world use cases, rules may be modified for multiple times to fully verify the performance of the grayscale version.

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Grayscale Release** and click the target canary release task.

Step 3 On the **Configure Traffic Policy** page, reconfigure **Cookie**, **Header**, **Query**, **Allowed OS**, and **Allowed Browser**.

Step 4 Click **Deliver Policy**.

----End

Switching the Grayscale Policy Type

You can change the type of a grayscale policy from **based on request content** to **based on traffic ratio** and vice versa. After this operation is complete, all configured rules become invalid and all traffic is redistributed based on the new policy.

NOTICE

Grayscale policies can be changed only for running tasks. After a grayscale version is released (that is, the new version completely takes over the traffic and the old version has been brought offline), its grayscale policy cannot be reconfigured.

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Grayscale Release** and click the target canary release task.

Step 3 On the **Configure Traffic Policy** page, change the policy type.

Step 4 Click **Deliver Policy**.

----End

Taking Over All Traffic

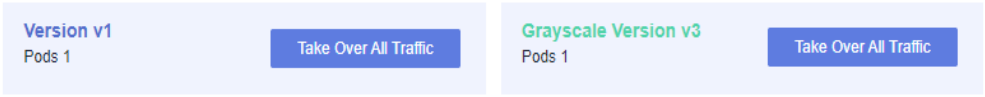
After you click **Take Over All Traffic**, the original version or grayscale version takes over all traffic.

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Grayscale Release** and click the target grayscale release task.

Step 3 On the **Monitor and Manage Traffic** page, click **Take Over All Traffic** next to the target version.

Figure 6-6 Taking over all traffic



Step 4 In the displayed dialog box, click **OK**.

----End

Terminating a Grayscale Release Task

After the grayscale version takes over all traffic, you can terminate the grayscale task. After the grayscale task is canceled, the original version will be brought offline, and all workloads and Istio configuration resources will be deleted.

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Grayscale Release** and click the target grayscale release task.

Step 3 On the **Monitor and Manage Traffic** page, click **Take Over All Traffic** next to the grayscale version.

Step 4 Click **Terminate Task** in the lower right corner.

Step 5 In the displayed dialog box, click **OK**.

You can go to the **Terminated Tasks** tab page to view the finished grayscale task. The **Release Result** is **Released successfully**.

----End

Canceling a Grayscale Release Task

After the original version takes over all traffic, you can cancel the grayscale task.

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Grayscale Release** and click the target grayscale release task.

Step 3 On the **Monitor and Manage Traffic** page, click **Take Over All Traffic** next to the original version.

Step 4 Click **Cancel Task** in the lower right corner. You can also click in the upper right corner of a task in the grayscale task list.

Step 5 In the displayed dialog box, click **OK**.

You can go to the **Terminated Tasks** tab page to view the finished grayscale task. The **Release Result** is **Released canceled**.

----End

Viewing Terminated Grayscale Release Tasks

You can view canceled and finished grayscale tasks on the **Terminated Tasks** tab page.

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane on the left, choose **Grayscale Release** and click the **Terminated Tasks** tab page.

You can view the release task name, release result, service, and release time, and delete a terminated task.

----End

7 Mesh Configuration

7.1 Overview

Mesh configuration provides cluster management, sidecar management, Istio resource management, and upgrade capabilities.

The mesh control plane workloads inject and manage sidecars of data plane pods, deliver policies and configurations, and collect monitoring data. Sidecars work with service containers in data plane pods, and they are in charge of routing and forwarding, traffic policy configuration, and monitoring data collection.

The functions of each tab page in **Mesh Configuration** are as follows:

- **Basic Information**: You can view the mesh name, ID, status, edition, version, billing mode, creation time, and clusters with the mesh enabled.
- **Sidecar Management**: You can view information about all workloads injected with sidecars, perform sidecar injection, and configure sidecar resource limits. For details, see **Sidecar Management**.
- **Istio Resource Management**: You can view all Istio resources (such as VirtualService and DestinationRule), create Istio resources in YAML or JSON format, and modify existing Istio resources. For details, see **Istio Resource Management**.
- **Upgrade**: You can upgrade the version of a service mesh. For details, see **Upgrading a Mesh**.
- **Extensions**: provides the observability configuration. For details, see **Service Mesh Extensions**.

7.2 Sidecar Management

On the **Sidecar Management** page, you can view information about all workloads injected with sidecars, perform sidecar injection, and configure sidecar resource limits.

Injecting a Sidecar

You can view the namespace and cluster to which the injected sidecar belongs. If no sidecar has been injected or you need to inject sidecar for more namespaces, perform the following operations:

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Mesh Configuration**. Then click the **Sidecar Management** tab.

Step 3 Click **Sidecar Management**, select a namespace, determine whether to restart the existing services, and click **OK**.

- **Namespace**: Select one or more namespaces. The system labels the namespaces with **istio-injection=enabled**.
- **Restart Existing Services**

  : Pods of the existing services in the namespace will be restarted, which will temporarily interrupt your services. The **istio-proxy** sidecar is automatically injected into the pods of the existing services.

  : The **istio-proxy** sidecar cannot be automatically injected into the pods of the existing services. You need to manually restart the workloads on the CCE console to inject the sidecar. Whether to restart existing services affects only existing services. If the namespaces are labeled with **istio-injection=enabled**, sidecars will be automatically injected into new pods.
- **Traffic Interception Settings**

  NOTE

  By default, sidecars intercept all inbound and outbound traffic of pods. You can modify the default traffic rules in **Traffic Interception Settings**.

  Inbound Ports: Inbound ports separated by commas (,). You can use this field to specify the ports that will be included or excluded for inbound traffic redirection.

  - **Include only specified ports** means that the traffic to services in a service mesh over specified ports will be redirected to the sidecar.
  - **Exclude only specified ports** means that the traffic to services in a service mesh over the ports except the specified ports will be redirected to the sidecar.

  Outbound Ports: Outbound ports separated by commas (,). You can use this field to specify the ports that will be included or excluded for outbound traffic redirection.

  - **Include only specified ports** means that the traffic from services in a service mesh over specified ports will be redirected to the sidecar.
  - **Exclude only specified ports** means that the traffic from services in a service mesh over the ports except the specified ports will be redirected to the sidecar.

  Outbound IP Ranges: IP address ranges separated by commas (,) in CIDR format. You can use this field to specify the IP ranges that will be excluded from redirection to the sidecar.

  - **Include only specified IP ranges** means that the traffic from specified IP ranges will be redirected to the sidecar.
  - **Exclude only specified IP ranges** means that the traffic from IP ranges except the specified IP ranges will be redirected to the sidecar.

NOTE

- If the system displays a message indicating that modification of namespace injection is not enabled in the following clusters, you need to run the **kubectl** command to enable namespace injection. For details, see **How Do I Enable Namespace Injection for a Cluster**.
- After sidecar injection is enabled for a namespace of a cluster, sidecars are automatically injected for pods of all workloads in the namespace. If you do not want to inject sidecars for some workloads, see **How Do I Disable Sidecar Injection for Workloads**.
- For meshes of 1.8.4-r1 and earlier versions (including all 1.3 and 1.6 versions), you are advised to check whether the workload contains the annotation **sidecar.istio.io/inject: 'true'**. If not, add the annotation to the **spec.template.metadata.annotations** field:

```
annotations:
  sidecar.istio.io/inject: 'true'
```

----End

Viewing Workload Details

The list displays all workloads created in the clusters managed by a mesh. You can view the workload name, cluster to which the workload belongs, service, and sidecar information of the workload, including the sidecar name, version, status, CPU usage, and memory usage. The procedure is as follows:

Step 1 In the drop-down list and search box in the upper right corner of the workload list, select a cluster and namespace, and enter the target workload name.

Step 2 Click ⌄ in front of the workload to view the sidecar information of the workload.

If the system displays a message indicating that there is no sidecar in the workload, no sidecar has been injected into the namespace to which the workload belongs. In this case, you can inject one into the namespace. For details, see **Injecting a Sidecar**.

----End

Configuring Sidecar Resource Limits

You can configure the upper and lower limits of CPU and memory resources for sidecars (istio-proxy container). If the upper and lower resource limits are not set for a workload, a resource leak of this workload will make resources unavailable for other workloads deployed on the same node. In addition, workloads that do not have upper and lower resource limits cannot be accurately monitored.

The default upper and lower limits of sidecar resources are as follows:

- CPU (core): 0.1 to 2 (included)
- MEM (MiB): 128 to 1,024 (included)

To change the value, perform the following operations:

Step 1 Click **Set Resource Limit** in the **Operation** column of the target workload. You can also select multiple workloads and click **Set Resource Limit** in the upper left corner of the workload list to configure sidecar resource limits in batches.

- Minimum CPU: CPU request, the minimum number of CPU cores required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to a node only when the total available CPU on the node is greater than or equal to the number of CPU cores applied for the container.
- Maximum CPU: CPU limit, the maximum number of CPU cores required by a container.
- Minimum memory: memory request, the minimum amount of memory required by a container. Resources are scheduled for the container based on this value. The container can be scheduled to this node only when the total available memory on the node is greater than or equal to the requested container memory.
- Maximum memory: memory limit, the maximum amount of memory required by a container. When the memory usage exceeds the specified memory limit, the pod may be restarted, which affects the normal use of the workload.

----End

Configuring Memory Alarm for Sidecar Resources

When the memory usage exceeds the specified memory limit, the pod may be restarted, which affects the normal use of the workload. Therefore, you are advised to configure memory alarm for all sidecar resources. After the alarm is triggered, you can increase the maximum memory size of the corresponding sidecar resource in a timely manner to reduce the impact on services.

Step 1 Log in to the AOM console.

Step 2 In the navigation pane on the left, choose **Alarm Center** > **Alarm Rules**, and click **Add Alarm** in the upper right corner.

Step 3 Set an alarm rule.

- **Rule Name**: Enter a rule name, for example, istio-proxy.
- **Rule Type**: Select **Threshold rule**.
- **Monitored Object**: Click **Select resource objects**, set **Add by** to **Dimension**, and select **Cloud Service Metrics** > **CCE** > **Container** > **Physical memory usage** for **Metric Name**.

  In the **Dimension** drop-down list, select **Cluster ID**, the ID of the cluster for which the alarm is enabled, **Container name**, and **istio-proxy** by sequence. Click **Confirm**.
- **Alarm Condition**: Set information such as statistical periods, consecutive periods, and threshold condition.
- **Alarm Mode**: Select **Direct Alarm Reporting**.

Step 4 Click **Create Now**.

If the following information is displayed in the rule list, the rule is created successfully.

Step 5 You can view the latest active alarm information and load details in the alarm list.

When the alarm is triggered, one or more records are added to the active alarm list. Click the alarm name. On the **Alarm Object** tab page, view parameters such as **deploymentName** and **nameSpace** to determine which sidecar resource is the alarm source. For details about how to modify the sidecar resource limits on the service plane, see **Configuring Sidecar Resource Limits**. The limits on the sidecar resources on the control plane, such as istio-ingressgateway, istio-egressgateway, and istio-eastwestgateway, need to be modified in upgrade mode on the **Workloads** page of the CCE console.

----End

7.3 Istio Resource Management

7.3.1 Configuring Istio Resources Using YAML

You can modify all Istio resources (such as VirtualService and DestinationRule) of a service in YAML or JSON format on the **Istio Resource Management** page. You can also create new Istio resources.

CAUTION

Istio resource configurations created or modified using YAML may conflict with those you directly add on the console. As a result, console functions will be unavailable. If you want to configure Istio resources only using YAML, edit or create Istio resources by referring to this section. Otherwise, skip this section.

Modifying an Existing Istio Resource

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Mesh Configuration**. Then click the **Istio Resource Management** tab.

Step 3 In the drop-down list, select the Istio resource type (for example, Istio Resources: virtualservices) and the namespace to which the resource belongs.

Step 4 Click **Edit** in the **Operation** column. In the right pane, modify related configurations and click **OK**. By default, the message displayed at the bottom is selected, indicating that related console functions are no longer available.

NOTE

Console functions vary depending on the Istio resource type. For details about unavailable console functions, see **Handling Policy of Resource Configuration Using YAML**.

The configuration file can be displayed in YAML or JSON format and can be downloaded to the local PC.

----End

Creating an Istio Resource

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Mesh Configuration**. Then click the **Istio Resource Management** tab.

Step 3 Click **Create** in the upper left corner of the list.

Step 4 Edit the file in the right pane, or click **Import File** to upload the edited YAML or JSON file.

Step 5 Confirm the file content and click **OK**. By default, the message displayed at the bottom is selected, indicating that related console functions are no longer available.

NOTE

Console functions vary depending on the Istio resource type. For details about unavailable console functions, see **Handling Policy of Resource Configuration Using YAML**.

----End

Istio Resource Description

Table 7-1 Istio resource description

Resource Type	Description
AuthorizationPolicy	Configures authorization policies.
DestinationRule	Defines the target service and traffic policy of a route. VirtualService and DestinationRule are the two most important resources for traffic control. DestinationRule defines the policies and rules for a Service in a mesh to provide external services, including the load balancing policy, exception monitoring, outlier detection control, and connection pool access.
EnvoyFilter	Provides more powerful extension capabilities for the service mesh control plane so that the Filter Chain in Envoy can be customized.
Gateway	Defines the unified ingress and egress for all HTTP/TCP traffic and describes a group of public ports, protocols, load balancing, and SNI configurations.
PeerAuthentication	Configures the mTLS mode for service communication. It is an Istio authentication policy.
RequestAuthentica-tion	Configures the request authentication method of a service. It is an Istio authentication policy.
ServiceEntry	Adds external services to a mesh and manages their traffic.

Resource Type	Description
Sidecar	Sets the sidecar proxies as a whole.
VirtualService	Set routes in a mesh. VirtualService and DestinationRule are the two most important resources for traffic control. VirtualService defines a group of routing rules. When traffic enters the mesh, the traffic is matched with the rules one by one. Once matched, the traffic is forwarded to the specified routing address.
WorkloadEntry	Abstracts VMs or bare metals so that they can be managed by a mesh. They are as important as pods in Kubernetes and provide traffic management, security management, and visualization.

7.3.2 Handling Policy of Resource Configuration Using YAML

You can create Istio resources on the console or using YAML on the **Istio Resource Management** page. To avoid configuration conflicts, you are advised to:

- Use console to maintain resources created on the console.
- Use YAML to maintain resources created by YAML.

If resources created on the console are modified using YAML, the corresponding functions on the console will be unavailable. For example, after VirtualService resources are configured using YAML, the corresponding services on the console cannot perform traffic management and grayscale release.

Handling Policy

When you edit or create Istio resources for a service on the **Mesh Configuration** > **Istio Resource Management** page, **Apply YAML only** at the bottom of the page is selected by default. Retain the selection and click **OK**, the configuration takes effect, however, some functions on the console are no longer available for the service and the resources can be maintained only in YAML mode.

CAUTION

If **Apply YAML only** is not selected, the modified configuration using YAML may conflict with some console functions.

Table 7-2 Impacts on console functions when **Apply YAML only** is selected

Symptom	Possible Cause
On the **Service Management** page, the **Security** button of the target service is unavailable.	The **AuthorizationPolicy** resource is modified using YAML.

Symptom	Possible Cause
On the **Service Management** page, the **Manage Traffic** button of the target service is unavailable.	The **DestinationRule** or **VirtualService** resource is modified using YAML.
On the **Service Management** page, the **Release** button of the target service is unavailable.	The **DestinationRule** or **VirtualService** resource is modified using YAML.
When a release task is created on the **Grayscale Release** page, the target service cannot be selected.	The **DestinationRule** or **VirtualService** resource is modified using YAML.
On the **Gateway Management** page, the **Add Route** button of the target gateway is unavailable.	Gateway resources are modified using YAML.
When a route is added on the **Gateway Management** page, the target service cannot be selected.	The **DestinationRule** or **VirtualService** resource is modified using YAML.

7.3.3 Handling Policy of Resource Configuration Using IstioOperator

When Istio is installed using Istio Operator, the workloads (istiod, istio-ingressgateway, and istio-egressgateway) managed by Istio Operator need to be updated. For example, upgrade the version of the service mesh enabled for the workload, or add more istio-ingressgateway instances. You can update these workloads on the **Workloads** page of the CCE console.

Handling Policy

In ASM 1.8.6, to avoid configuration conflicts and ensure stable running of Istio workloads, you are advised to:

- Define key and non-key running configurations for workloads.

  Table 7-3 Key running configurations of each resource type

Workload	Resource Type	Item	Description	Applicable Version
istiod istio-ingressgateway istio-egressgateway	Deployment	spec.replicas	Number of pods	1.8.6 or later
		spec.strategy	Upgrade policies	
		spec.template.spec.nodeSelector	Scheduling policies	

Workload	Resource Type	Item	Description	Applicable Version
		spec.template.spec.affinity	Scheduling policies	
		spec.template.spec.tolerations	Scheduling policies	
		spec.template.spec.containers.resources	Resource requests and limits	
istiod istio-ingressgateway istio-egressgateway	Deployment	spec.template.spec.containers.env	Container environment variables	1.13.9-r10, 1.15.7-r3, or later 1.18.7-r3 or later
istio-cni-node	DaemonSet	spec.updateStrategy	Upgrade policies	1.18.5-r1 or later
		spec.template.spec.nodeSelector	Scheduling policies	
		spec.template.spec.affinity	Scheduling policies	
		spec.template.spec.tolerations	Scheduling policies	
		spec.template.spec.containers.resources	Resource requests and limits	
		spec.template.spec.containers.env	Container environment variables	1.18.7-r3 or later

- By default, the Istio Operator does not update key running configurations of workloads in the current cluster. Only non-key running configurations can be updated.
- If you need to modify key running configurations, you are advised to modify them on the **Workloads** page of the CCE console. If you have special requirements, submit a service ticket.

7.4 Upgrades

7.4.1 Upgrading a Mesh

Scenarios

You can upgrade a mesh of an earlier version. ASM of the Basic edition supports canary upgrades, and ASM of the Enterprise edition supports in-place patch upgrade.

Upgrade Impact

- During the upgrade, the data plane proxy of the new version is automatically injected. Service pods will be restarted in the rolling mode, which may temporarily interrupt services.
- Do not perform operations including but not limited to creating a grayscale release task or configuring a traffic rule during the upgrade.

Upgrade Path

Mesh Edition	Source Version	Target Version	Upgrade Mode
Basic edition	1.8.4-r1	1.8.6-r2	Patch update
	1.8.4-r2	1.8.6-r2	Patch update
	1.8.4-r3	1.8.6-r2	Patch update
	1.8.4-r4	1.8.6-r2	Patch update
	1.8.4-r5	1.8.6-r2	Patch update

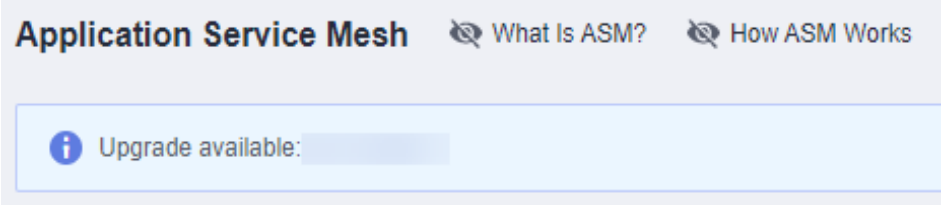
NOTE

For details about the features of each version, see **Features in v1.3**, **Features in v1.6**, and **Features in v1.8**.

Procedure

Step 1 Log in to the ASM console and check whether the meshes need to be upgraded. The specific steps are as follows:

- Check whether a message indicating the mesh to be upgraded is displayed above the list.



- Check whether **Upgrade** is displayed on the right of the mesh name.

If a service mesh can be upgraded, click its name to go to its details page.

Step 2 In the navigation pane, choose **Mesh Configuration**. Then click the **Upgrade** tab.

Step 3 Select a proper upgrade mode to upgrade the service mesh according to **Upgrade Path**.

- **Upgrading the Basic Edition**

  Click **Upgrade Mesh**. The system automatically diagnoses the upgrade. After all items pass the check, click **Upgrade**.

- **Updating the Enterprise Edition Patch**

  Click **Update Patch**. In the dialog box that is displayed, click **OK**.

----End

7.4.2 Features in v1.3

- Mixer-based metrics, access logs, and call chains are supported.
- SDS is supported and the pods do not need to be restarted for certificate rotation.
- Sidecar APIs are supported.
- Control-plane performance optimization is supported.
- The private version of Huawei Virtual Service Delegation goes online, providing decoupling gateway traffic configuration for major customers in advance.
- CCE clusters v1.13, v1.15, v1.17, and v1.19 are supported.

For details, see **https://istio.io/latest/news/releases/1.3.x/announcing-1.3/change-notes/**.

7.4.3 Features in v1.6

- Control plane components are integrated, which simplifies control plane installation and O&M.
- The official community version of Virtual Service Delegation is updated. (This feature is provided by Huawei. The APIs are the same as those of v1.3.)
- Workload Entry enables operators to describe the properties of a single non-Kubernetes workload.
- SDS is enabled by default.
- Data plane–based data collection using Telemetry V2 (non-Mixer-based telemetry) is supported.
- Multi-port service governance based on the port granularity is supported.
- CCE clusters v1.15 and v1.17 are supported.

For details, see **https://istio.io/latest/news/releases/1.6.x/announcing-1.6/change-notes/**.

7.4.4 Features in v1.8

- The Mixer component is officially brought offline. Access logs, traces, and monitoring data are collected based on the data plane.
- **EnvoyFilter** is enhanced to support more flexible **Insert** operations.

- A new AuthorizationPolicy-based authorization policy is enabled.
- CCE clusters v1.15, v1.17, v1.19, and v1.21 are supported.

For details, see **https://istio.io/latest/news/releases/1.8.x/announcing-1.8/change-notes/**.

7.4.5 Features in v1.13

- Istio 1.13.9 is supported.
- CCE Turbo clusters v1.21 and v1.23 are supported.
- CCE clusters v1.21 and v1.23 are supported.
- **Istio sidecar proxy** can be configured through the ProxyConfig API.
- In Istio 1.10 and later versions, Sidecar does not redirect traffic to the loopback interface (lo). Instead, it redirects traffic to the eth0 interface of applications. If your service pod listens the lo, change the listening port to eth0 or all-zero. Otherwise, the service cannot be accessed after the upgrade. **Learn more**.

For details, see **https://istio.io/latest/news/releases/1.13.x/announcing-1.13/change-notes/**.

7.4.6 Features in v1.15

- Istio 1.15.7 is supported.
- CCE Turbo clusters v1.21, v1.23, v1.25, and v1.27 are supported.
- CCE clusters v1.21, v1.23, v1.25, and v1.27 are supported.
- Security vulnerabilities such as CVE-2023-44487, CVE-2023-39325 and CVE-2023-27487 are fixed.

For details, visit **https://istio.io/latest/news/releases/1.15.x/announcing-1.15.7/**.

7.4.7 Features in v1.18

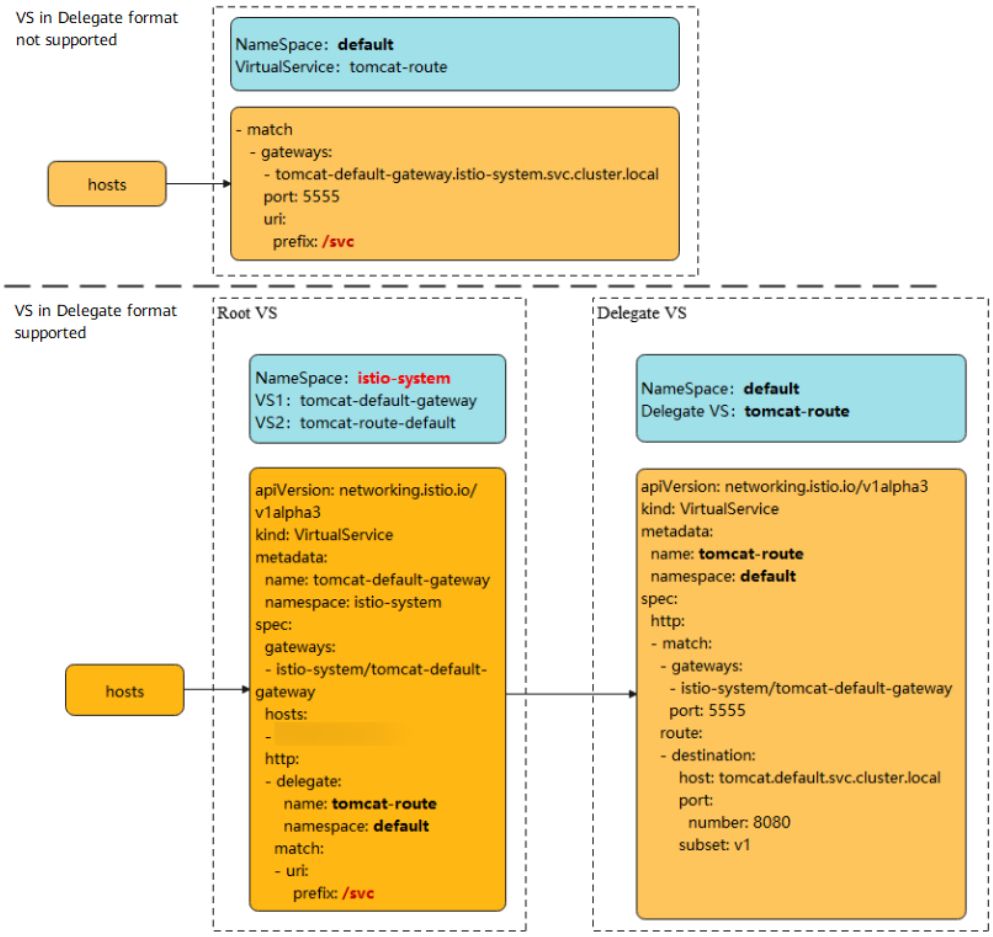
- Istio 1.18.7 is supported.
- CCE Turbo clusters v1.25, v1.27, v1.28, v1.29, and v1.30 are supported.
- CCE clusters v1.25, v1.27, v1.28, v1.29, and v1.30 are supported.
- Kubernetes Gateway API is supported.

For details, visit **https://istio.io/latest/news/releases/1.18.x/**.

7.4.8 Upgrading v1.3 to v1.8 to Allow VirtualService to Support Delegate Switchover

Scenarios

By default, the mesh of v1.8 supports the **Delegate** function of **VirtualService**, and the ASM console supports only the **VirtualService** in **Delegate** format. The upgrade does not modify your **VirtualService**, but you cannot maintain the route on the page after the upgrade. Therefore, you need to modify your **VirtualService** according to this section.



NOTE

For details about **Delegate**, see the description in the Istio community.
https://istio.io/latest/docs/reference/config/networking/virtual-service/#Delegate

Constraints

- **Delegate** can be set only when **route** and **redirect** are left blank.
- ASM supports only 1-level **Delegate**. Multi-level **Delegate** does not take effect.
- **HTTPMatchRequest** of **Delegate VirtualService** must be a subset of **root VirtualService**.
- The **Delegate** feature is valid only for HTTP and gRPC protocols.

Procedure

The modification involves two scenarios. The following uses the **Tomcat** service added to a mesh as an example.

Scenario 1: If no gateway is added to the service before the upgrade, you can skip the following operations after the upgrade.

Scenario 2: If a gateway is added to the service before the upgrade, perform the following operations after the upgrade.

1. Configure the **kubectl** command for the cluster where the mesh is located. For details, see the cluster details on the CCE console.
2. Create two **VirtualService YAML** files in the **istio-system** namespace.

   File name: **tomcat**-default-gateway.yaml

   To be more specific:

   - **tomcat**: name of the service to be modified.
   - **tomcat-default-gateway**: name of the **VirtualService**. The format is **{*Service name*}-default-gateway**.
   - **tomcat-route**: name of the **VirtualService** to be modified.
   - **100.85.219.117**: ELB IP address.

   ```
   apiVersion: networking.istio.io/v1beta1
   kind: VirtualService
   metadata:
     name: tomcat-default-gateway
     namespace: istio-system
   spec:
     gateways:
     - istio-system/tomcat-default-gateway
     hosts:
     - 100.85.219.117
     http:
     - delegate:
         name: tomcat-route
         namespace: default
       match:
       - uri:
           prefix: /test
   ```

   File name: **tomcat**-route-default.yaml

   To be more specific:

   - **tomcat**: name of the service to be modified.
   - **tomcat-route-default**: name of the **VirtualService**. The format is **{*Service name*}-route-default**.
   - **tomcat-route**: name of the **VirtualService** to be modified.

   ```
   apiVersion: networking.istio.io/v1beta1
   kind: VirtualService
   metadata:
     name: tomcat-route-default
     namespace: istio-system
   spec:
     hosts:
     - tomcat.default.svc.cluster.local
     http:
     - delegate:
         name: tomcat-route
         namespace: default
       match:
       - uri:
           prefix: /
   ```

   Run the following commands to create a **VirtualService**:

   kubectl create -f tomcat-route-default.yaml

   kubectl create -f tomcat-default-gateway.yaml
3. Run the **kubectl -n{namespace} get vs** command to obtain the **VirtualService** of the service and run the **kubectl -n{namespace} edit vs tomcat-route** command to modify it as follows:

a. Delete **spec.gateways**, **spec.hosts**, and **spec.http.match.uri**.

b. Replace **tomcat-default-gateway.istio-system.svc.cluster.local** with **istio-system/tomcat-default-gateway**.

c. Change **apiVersion: networking.istio.io/v1alpha3** to **apiVersion: networking.istio.io/v1beta1**.

d. **destination.host**: The format is **{*Service name*}.{namespace}.svc.cluster.local**.

```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: tomcat-route
  namespace: default
spec:
  gateways:
  - tomcat-default-gateway.istio-system.svc.cluster.local
  - mesh
  hosts:
  - tomcat
 - 100.85.219.117 # spec.gateways and spec.hosts need to be deleted.
  http:
  - match:
    - gateways:
      - istio-system/tomcat-default-gateway
      port: 5555
      uri:
        prefix: /test # spec.http.match.uri needs to be deleted.
    route:
    - destination:
        host: tomcat.default.svc.cluster.local
        port:
          number: 8080
        subset: v1
  - match:
    - gateways:
      - mesh
      port: 8080
    route:
    - destination:
        host: tomcat.default.svc.cluster.local
        port:
          number: 8080
        subset: v1
```

4. Click **External Access URL** on the service list page to check whether it can be accessed normally.

5. On the **Service Gateway** page, check whether the service gateway route is displayed properly.

7.5 Extensions

Observability configuration includes applications metrics andaccess logging and tracing of a service mesh. You can enable application metrics and access logging.

NOTE

Tracing can be enabled only when a service mesh is created.

Constraints

- Istio earlier than 1.18 can only work with APM to monitor application metrics. Istio 1.18 or later can only work with AOM to monitor application metrics. Istio earlier than 1.18 can also work with AOM after being upgraded to 1.18. To improve metric monitoring, you are advised to switch APM to AOM.
- Only Istio 1.18 or later can work with LTS to collect and store access logs. To enable access logging, install Cloud Native Logging (log-agent) on the **Add-ons** page in advance.

Enabling Application Metrics

Step 1 Log in to the ASM console.

Step 2 Click the name of the service mesh to go to its details page.

Step 3 In the navigation pane, choose **Mesh Configuration**. Then, click the **Extensions** tab.

Step 4 Enable application metrics, select an AOM instance, and click **OK**.

----End

Enabling Access Logging

Step 1 Log in to the ASM console.

Step 2 Click the name of the service mesh to go to its details page.

Step 3 In the navigation pane, choose **Mesh Configuration**. Then, click the **Extensions** tab.

Step 4 Enable access logging, select the log group and log stream, and click **OK**.

----End

8 Traffic Management

8.1 Overview

Non-intrusive traffic management is a core function of Istio. With traffic management, you only need to focus on your own service logic rather than service access management. Traffic management enables you to:

- Dynamically modify load balancing policies for cross-service access, such as configuring consistent hashing to forward traffic to specific service pods.
- Distribute a certain proportion of traffic to a specific version of a service when the service has two online versions.
- Protect services, for example, limiting the number of concurrent connections and requests, and isolating faulty service pods.
- Dynamically modify the content of a service or simulate a service running fault.

ASM provides retry, timeout, connection pool, outlier detection, load balancing, HTTP header, and fault injection functions to meet traffic management requirements in most service scenarios.

Table 8-1 Common mesh functions and management roles

Mesh Function	Management Role	
	Service Initiator	Service Provider
Route management	Y	N
Load balancing	Y	N
Tracing analysis	Y	Y
Service authentication	Y	Y
Observability data	Y	Y
Retry	Y	N

Rewrite	Y	N
Redirection	Y	N
Authorization	N	Y
Fault injection	Y	N
Timeout	Y	N
Connection pool	Y	N
Outlier detection	Y	N
HTTP header	Y	N

Constraints

Traffic management cannot be performed for the service whose configuration diagnosis fails. For details about rectifying faults, see **Manual Fixing Items** or **Auto Fixing Items**.

8.2 Configuring a Traffic Policy

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Service Management**. In the upper right corner of the list, select the namespace that your services belong to.

Step 3 Locate the target service and click **Manage Traffic** in the **Operation** column. In the window that slides out from the right, configure retry, timeout, connection pool, outlier detection, load balancing, HTTP header, and fault injection policies.

Retry

Auto retries upon service access failures improve the access quality and success rate.

On the **Retry** tab, click **Configure now**. In the displayed dialog box, set the parameters listed in the table below.

Table 8-2 Retry parameters

Parameter	Description	Value Range
Retries	Maximum number of retries allowed for a single request. The default retry interval is 25 ms. The actual number of retries depends on the configured timeout period and retry timeout period.	1-2147483647

Parameter	Description	Value Range
Retry Timeout (s)	Timeout period of an initial or retry request. The default value is the same as the timeout period configured in the **Timeout** area below.	0.001–2592000
Retry Condition	Configure retry conditions. For details, see **Retry Policies** and **gRPC Retry Policies**.	-

Timeout

Auto processing and quickly failure return upon service access timeout eliminate resource locking and request freezing.

On the **Timeout** tab, click **Configure now**. In the displayed dialog box, set the parameters listed in the table below.

Table 8-3 Timeout parameters

Parameter	Description	Value Range
Timeout (s)	Timeout period for HTTP requests	0.001–2592000

Connection Pool

Connections and requests that exceed the thresholds are cut off to protect target services. Connection pool settings are applied to each pod of the upstream service at the TCP and HTTP levels. For details, see **Circuit Breaker**.

On the **Connection Pool** tab, click **Configure now**. In the displayed dialog box, set the parameters listed in the tables below.

Table 8-4 Parameters under TCP Settings

Parameter	Description	Value Range
Maximum Number of Connections	Maximum number of HTTP/TCP connections to the target service. The default value is **4294967295**.	1-2147483647
Maximum Number of Non-responses	Maximum number of keepalive probes to be sent before the connection is determined to be invalid. By default, the OS-level configuration is used. (The default value is **9** for Linux.)	1-2147483647

Parameter	Description	Value Range
Health Check Interval (s)	Time interval between two keepalive probes. By default, the OS-level configuration is used. (The default value is **75** for Linux.)	0.001–2592000
Connection Timeout (s)	TCP connection timeout period. The default value is **10**.	0.001–2592000
Minimum Idle Period (s)	Duration in which a connection remains idle before a keepalive probe is sent. By default, the OS-level configuration is used. (The default value is **7200** for Linux, namely, 2 hours.)	0.001–2592000

Table 8-5 Parameters under HTTP Settings

Parameter	Description	Value Range
Maximum Number of Requests	Maximum number of requests that can be forwarded to a single service pod. The default value is **4294967295**.	1-2147483647
Maximum Number of Pending Requests	Maximum number of HTTP requests that can be forwarded to the target service for processing. The default value is **4294967295**.	1-2147483647
Maximum Connection Idle Period (s)	Timeout period of an idle upstream service connection. If there is no active request within this time period, the connection will be closed. The default value is **3600** (1 hour).	0.001–2592000
Maximum Retries	Maximum number of retries of all service pods within a specified period. The default value is **4294967295**.	1-2147483647
Maximum Number of Requests Per Connection	Maximum number of requests for each connection to the backend. If this parameter is set to **1**, the keepalive function is disabled. The default value is **0**, indicating infinite. The maximum value is **536870912**.	1–536870912

Outlier Detection

Unhealthy pods are automatically isolated to improve the overall access success rate.

The traffic status of service pods is traced to determine whether the pods are healthy. Unhealthy pods will be ejected from the connection pool to improve the overall access success rate. Outlier detection can be configured for HTTP and TCP services. For HTTP services, pods that continuously return 5xx errors are considered unhealthy. For TCP services, pods whose connections time out or fail are considered unhealthy. For details, see **Outlier Detection**.

On the **Outlier Detection** tab, click **Configure now**. In the displayed dialog box, set the parameters listed in the table below.

Table 8-6 Outlier detection parameters

Parameter	Description	Value Range
Consecutive Errors	Number of consecutive errors in a specified time period. If the number of consecutive errors exceeds the parameter value, the pod will be ejected. The default value is **5**.	1-2147483647
Base Ejection Time (s)	Base ejection time of a service pod that meets the outlier detection conditions. The actual ejection time of a service pod = Base ejection time x Number of ejection times. The value must be greater than or equal to 0.001s. The default value is **30**.	0.001–2592000
Inspection Interval (s)	If the number of errors reaches the threshold within this time period, the pod will be ejected. The value must be greater than or equal to 0.001s. The default value is **10**.	0.001–2592000
Maximum Percentage of Ejected Pods (%)	Maximum percentage of ejected service pods. The default value is **10**.	1–100

Load Balancing

You can customize a load balancing policy to target backend service pods.

On the **Load Balancing** tab, click **Configure now**. In the displayed dialog box, select one of the following load balancing algorithms as required:

- **Round robin**: The default load balancing algorithm. Each service pod in the pool gets a request in turn.
- **Least connection**: Requests are forwarded to the pod with fewer connections among two randomly selected healthy pods.

- **Random**: Requests are forwarded to a randomly selected healthy pod.
- **Consistent hashing**: There are four types, as described in **Table 8-7**.

Table 8-7 Consistent hashing algorithm types

Type	Description
Based on HTTP header	The hash value is calculated using the header of the HTTP request. Requests with the same hash value are forwarded to the same pod.
Based on cookie	The hash value is calculated using the cookie key name of the HTTP request. Requests with the same hash value are forwarded to the same pod.
Based on source IP	The hash value is calculated using the IP address of the HTTP request. Requests with the same hash value are forwarded to the same pod.
Based on query parameter	The hash value is calculated using the query parameter key name of the HTTP request. Requests with the same hash value are forwarded to the same pod.

HTTP Header

You can flexibly add, modify, and delete specified HTTP headers to manage request contents in non-intrusive mode.

On the **HTTP Header** tab, click **Configure now**. In the displayed dialog box, set the parameters listed in the tables below.

Table 8-8 Operations on the HTTP headers before the request is forwarded to the destination service

Parameter	Description
Add request headers.	To add a request header, you need to set **key** and **value**. You can also click ⊕ to add more request headers.
Edit request headers.	To edit an existing request header, you need to set **key** and **value**. You can also click ⊕ to edit more request headers.
Remove request headers.	To remove an existing request header, you need to set **key**. You can also click ⊕ to remove more request headers.

Table 8-9 Operations on the HTTP headers before the response is returned to the client

Parameter	Description
Add response headers.	To add a response header, you need to set **key** and **value**. You can also click ⊕ to add more response headers.
Edit response headers.	To edit an existing response header, you need to set **key** and **value**. You can also click ⊕ to edit more response headers.
Remove response headers.	To remove an existing response header, you need to set **key**. You can also click ⊕ to remove more response headers.

Fault Injection

You can inject faults into the system to check whether it can tolerate and recover from faults.

On the **Fault Injection** tab, click **Configure now**. In the displayed dialog box, set the parameters listed in the table below.

Table 8-10 Fault injection parameters

Parameter	Description	Value Range
Fault Type	The options are **Delay** and **Abort**. • **Delay**: Service requests are delayed. • **Abort**: A service is aborted and the preset status code is returned.	**Delay** and **Abort**
Delay (s)	This parameter needs to be set when **Fault Type** is set to **Delay**. A request will be delayed for this period of time before it is forwarded.	0.001–2592000
HTTP Status Code	This parameter needs to be set when **Fault Type** is set to **Abort**. HTTP status code returned when an abort fault occurs. The default value is **500**.	200–599
Fault Percentage (%)	Percentage of requests for which the delay or abort fault is injected.	1–100

----End

8.3 Changing a Traffic Policy

Scenarios

You can change the settings of a configured traffic policy. For example, you can change the load balancing algorithm from **Round robin** to **Random**.

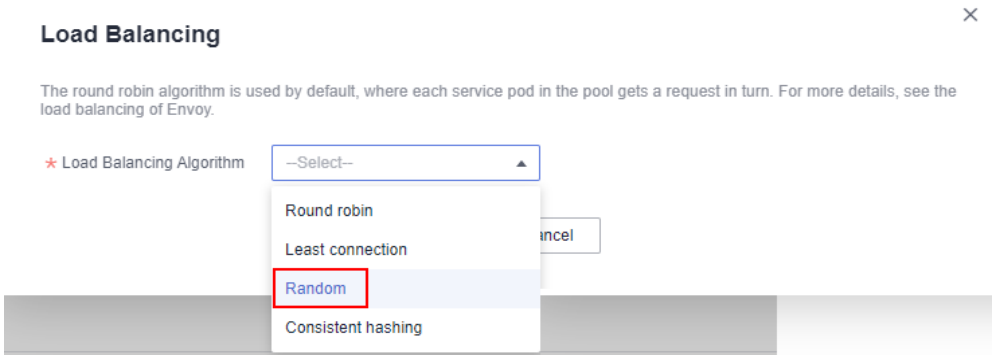
Procedure

Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Service Management**. Locate the service whose traffic policy needs to be modified and click **Manage Traffic** in the **Operation** column. In the window that slides out from the right, modify traffic policies.

Step 3 On the **Load Balancing** tab, click **Configure now**. In the displayed dialog box, change the algorithm to **Random** and click **OK**.

Figure 8-1 Modifying the load balancing algorithm



----End

9 Security

9.1 Configuring a Security Policy

ASM security functions include **Access Authorization**, **Peer Authentication**, **JWT Authentication** to ensure the reliable service communication.

Procedure

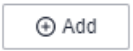
Step 1 Log in to the ASM console and click the name of the target service mesh to go to its details page.

Step 2 In the navigation pane, choose **Service Management**. In the upper right corner of the list, select the namespace that your services belong to.

Step 3 Locate the target service and click **Security** in the **Operation** column. In the window that slides out from the right, configure access authorization and peer authentication.

Access Authorization

Access authorization controls the access to services in the mesh and determines whether a request can be sent to the current service.

On the **Access Authorization** tab, click **Configure now**. In the displayed dialog box, click 

to select one or more services in a specified namespace.

Peer Authentication

Istio enables communication between service pods using the Policy Enforcement Point (PEP) tunnel between clients and servers. Peer authentication defines how traffic reaches the current service pod through the tunnel (or not through the tunnel). By default, service pods that have sidecars injected communicate with each other through tunnels. Traffic is automatically encrypted using TLS.

On the **Peer Authentication** tab, click **Configure now**. In the displayed dialog box, select an authentication policy.

Table 9-1 Authentication policies

Parameter	Description
UNSET	If a peer authentication policy is configured for the parent scope, the service inherits the policy.
PERMISSIVE	Traffic can be transmitted without passing through the tunnel. Workloads accept both mutual TLS and plain text traffic. By default, the mesh is configured with a peer authentication policy in PERMISSIVE mode.
STRICT	Traffic is transmitted only through the tunnel because the request must be encrypted using TLS and must contain the client certificate.

JWT Authentication

You can configure JWT authentication on ASM. With JWT, ASM authenticates whether the access token in a request header is trusted and authorize the valid user requests.

NOTE

JWT authentication can be configured only for HTTP services.

On the **JWT Authentication** tab, click **Configure now**. In the displayed dialog box, set the following parameters:

- **Issuer**: issuer of the JWT
- **Audiences**: audiences who use the JWT token to access the service. Separate audiences by commas (,). A null value indicates that the service can be accessed by any audiences.
- **JWKS**: JWT rule set

For details about the principles and application examples of JWT authentication, see **JWT Authentication Principles** and **Authenticating JWT Requests on the Ingress Gateway Using ASM**.

----End

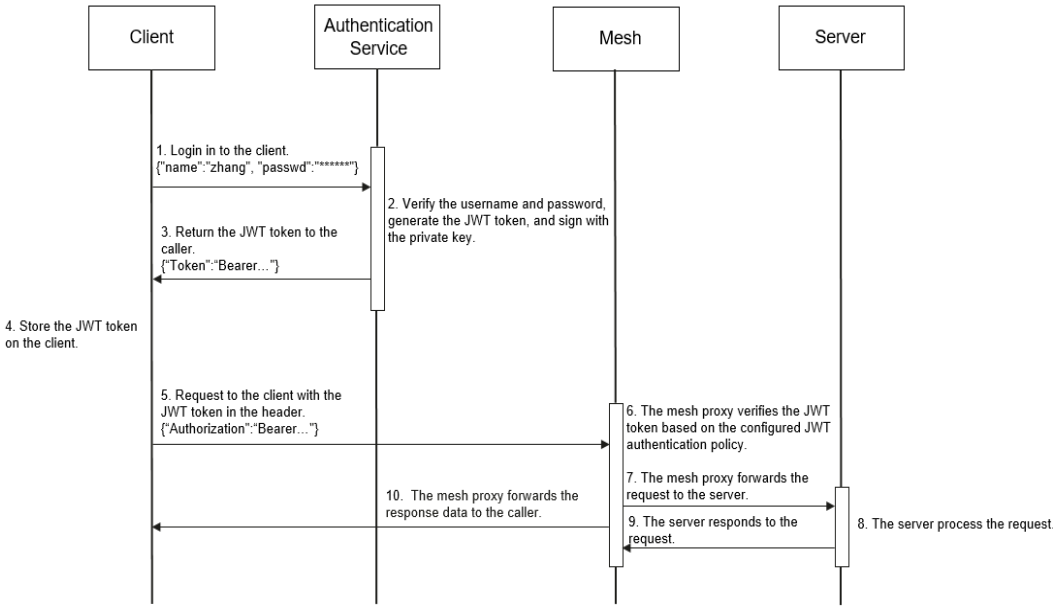
9.2 JWT Authentication Principles

JWT Authentication Principles

JWT is an authentication mode in which the server issues tokens to the client. When a user logs in to the client using the username and password, the server generates and returns a token to the client. The client only needs to carry the token when sending a request to the server. The server verifies the token to determine whether the request is from a valid client and determines whether to return a response to the client. This method, which connects authenticated clients based on tokens in requests, solves various stateful problems of storing sessions on a server at an early stage.

In Istio, the JWT token is generated by a specific authentication service and verified by meshes, completely decoupling the authentication logic in user services and enabling applications to focus on their own services. **Figure 9-1** shows the complete Istio-based JWT mechanism.

Figure 9-1 Istio JWT authentication process



1. The client connects the authentication service by logging with the user name and password.

2. The authentication service verifies the username and password, generates a JWT token (including the user ID and expiration time), and signs the token with the private key of the authentication service.

3. The authentication service returns the JWT token to the client.

4. The client stores the JWT token locally for subsequent requests.

5. When requesting to another service, the client carries the JWT token and does not need to provide information such as the username and password.

6. The mesh data plane proxy intercepts the traffic and verifies the JWT token using the configured public key.

7. Once the JWT token is verified, the mesh proxy forwards the request to the server.

8. The server processes the request.

9. The server returns the response data to the mesh proxy.

10. The mesh data plane proxy forwards the response data to the caller.

The step 6 is important because the JWT authentication function is migrated from the server to the mesh proxy. The mesh data plane obtains, from the authentication policy configured by the control plane, the public key for verifying the JWT token. The public key may be one configured on the JWKS (JSON Web

Key Set) or one obtained from a public key address configured by jwksUri. After obtaining the public key, the mesh proxy uses it to verify the token signed by the private key of the authentication service, decrypts the **iss** in the token, and verifies whether the issuer information in the authentication policy is matched. If the verification is successful, the request is sent to the application. If not, the request is rejected.

JWT Structure

JWT is of the JSON structure that contains specific declarations. Step 6 of the JWT authentication process shows the request identity can be confirmed by verifying the JSON structure. Querying the backend service is not needed. The following shows how the authentication information is carried by parsing the JWT structure.

A JWT token consists of three parts: header, payload, and signature.

- Header

  Describes the JWT metadata, including the algorithm **alg** and type **typ**. **alg** describes the signature algorithm so that the receiver can verify the signature based on the corresponding algorithm. The default algorithm is **HS256** (as follows), indicating **HMAC-SHA256**. **typ** indicates the token type. If **typ** is **JWT**, indicating that the token is of the JWT type.

  ```
  {
    "alg": "HS256",
    "typ": "JWT"
  }
  ```

- Payload

  Stores the main content of the token. The authentication service AuthN generates related information and places the information in the token payload. The attributes of **payload** include:

  – **iss**: token issuer

  – **aud**: token audience

  During JWT verification, the **iss** and **aud** will be verified to check whether they are matched with the token issuer and audience. The JWT content is not encrypted. All services that obtain the token can view the content in the token payload. You are advised not to store private information in the payload.

- Signature

  Signature of the header and payload, ensuring that only the specific legitimate authentication services can issue tokens. Generally, the header and payload are converted into strings using Base64, and then the private key of the authentication service is used to sign the strings. The signature algorithm is the **alg** defined in the header.

The following is a complete JWT example. Signature is obtained by signing the header and payload.

```
# Header:
{
  "alg": "RS512",
  "typ": "JWT"
}

# Payload
{
  "iss": "weather@cloudnative-istio",
  "audience": "weather@cloudnative-istio"
```



```
}

# Signature
RSASHA512(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload)
  )
```

The token output of the preceding structure is as follows. The three strings separated by periods (.) correspond to the header, payload, and signature of the JWT structure, respectively.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjQ2ODU5ODk3MDAsInZlciI6IjIuMCIsImlhdCI6MTUzMjM4OTcwMCwiaXNzIjoid2VhdGhlckBjbG91ZG5hdGl2ZS1pc3Rpby5ib29rIiwic3ViIjoid2VhdGhlckBjbG91ZG5hdGl2ZS1pc3Rpby5ib29rIn0.SEp-8qiMwI45BuBgQPH-wTHvOYxcE_jPI0wqOxEpauw
```

9.3 Authenticating JWT Requests on the Ingress Gateway Using ASM

This section describes how to authenticate JWT requests on the ingress gateway using ASM to ensure that users access services through the ingress gateway with a reliable access token.

Preparations

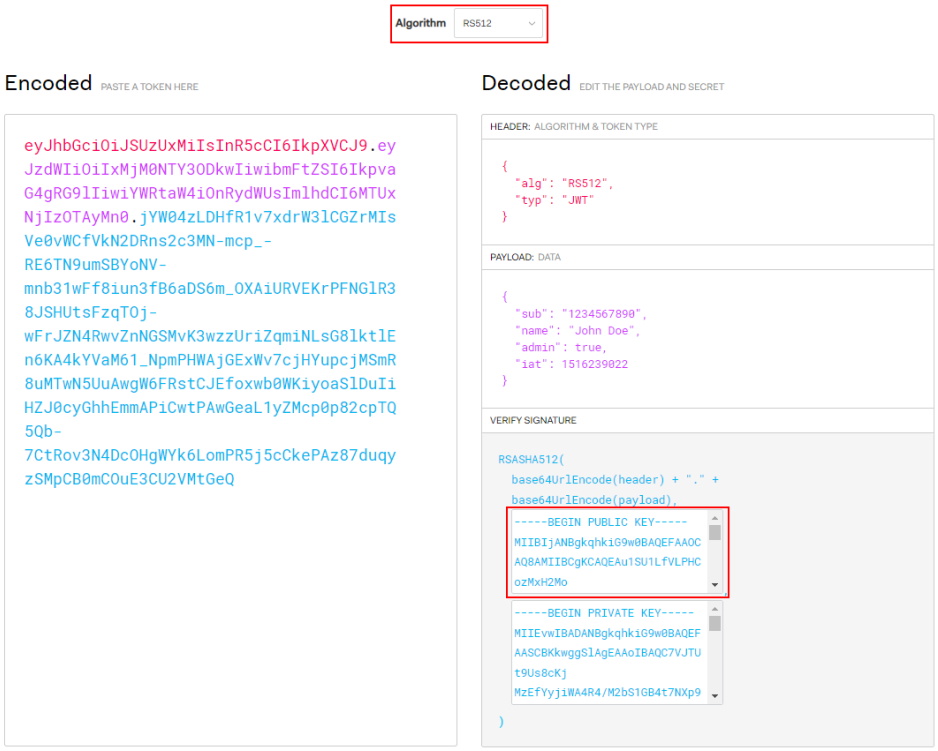
1. A service mesh of version 1.13/1.15 or 1.18 has been created.
2. The **httpbin** service that passes the diagnosis exists in the mesh. The image is **httpbin**, the port protocol is **HTTP**, and the port number is **80**.
3. An accessible gateway has been created for the **httpbin** service in the mesh.

Creating JWT Authentication

Step 1 Create a JWK.

1. Visit **JWT tool website**, set **Algorithm** to **RS512**, and obtain the public key (PUBLIC KEY).

Figure 9-2 Generating a public key



2. Select **PEM-to-JWK (RSA Only)** in the **JWK to PEM Convertor online** tool, enter the public key obtained in the previous step, and click **submit** to convert the public key into a JWK.

Figure 9-3 Converting the public key to a JWK


```
{"kty":"RSA","e":"AQAB","kid":"a78641b9-d81e-4241-b35a-71726c3fa053","n":"u1SU1LfVLPHCozMxH2Mo4lgOEePzNm0tRgeLezV6ffAt0gunVTLw7onLRnrq0
```


_IzW7yWR7QkrmBL7jTKEn5u-qKhbwKfBstIs-bMY2Zkp18gnTxKLxoS2tFczGkPLPgizskuemMghRniWaoLcyehkd3qqGElvW_VDL5AaWTg0nLVkjRo9z-40RQzuVaE8AkAFmxZzow3x-VJYKdjykkJ0iT9wCS0DRTXu269V264Vf_3jvredZiKRkgwlL9xNAwxXFg0x_XFw005UWVRIkdgcKWTjpBP2dPwVZ4WWC-9aGVd-Gyn1o0CLelf4rEjGoXbAAEgAqeGUxrcIlbjXfbcmw"}

Step 2 Create JWT authentication.

1. Log in to the ASM console and click the name of the target service mesh to go to its details page.
2. In the navigation pane, choose **Service Management**. In the upper right corner of the list, select the namespace that your services belong to.
3. Locate the **httpbin** service and click **Security** in the **Operation** column. In the window that slides out from the right, click **JWT Authentication** and then **Configure now**. In the displayed dialog box, set the following parameters:
   - **Issuer**: issuer of the JWT. Set this parameter to **test**.
   - **Audience**: JWT audiences who use the JWT token to access the target service. Set this parameter to **ASM**.
   - **JWKS**: JWT information. Set this parameter to **{"keys": [*JWK created in Step 1*]}**. For example, if the JWK created in **Step 1** is **{"kty":"RSA","e":"AQAB","kid":"a78641b9-d81e-4241-b35a-71726c3****"}**, the value of **JWKS** is **{"keys": [{"kty":"RSA","e":"AQAB","kid":"a78641b9-d81e-4241-b35a-71726c3****"}]}**.

Figure 9-4 Creating JWT authentication

4. Click **OK**.

----End

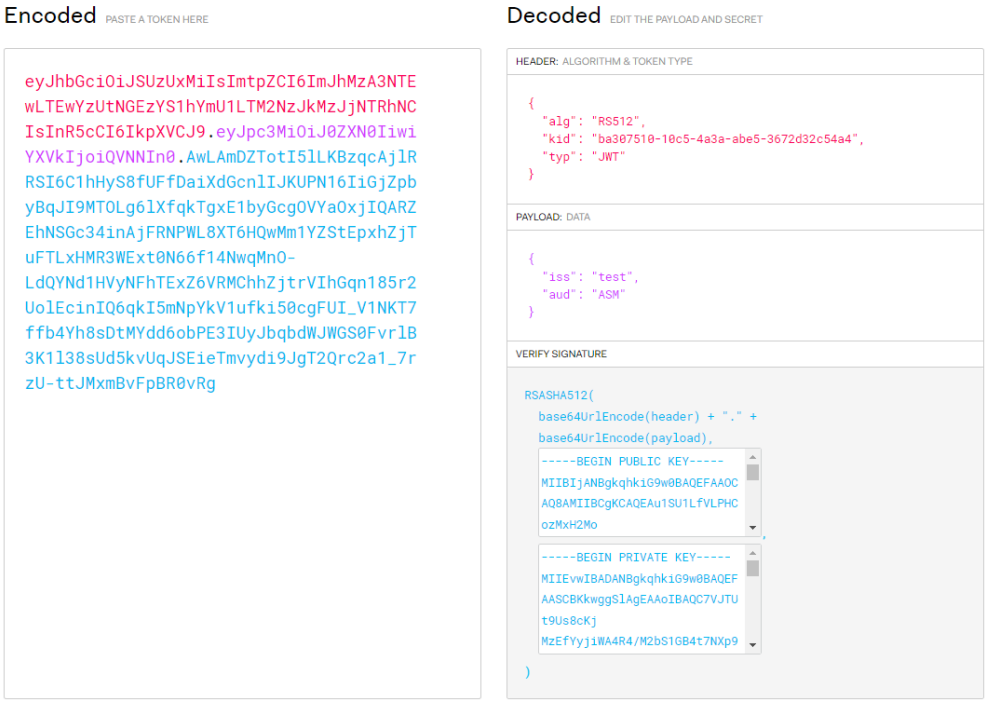
Checking Whether JWT Authentication Takes Effect

Step 1 Use **JWT tool** to encode the JWT request information into a JWT token.

Enter the following JWT request information in the **Decoded** area. The automatically converted JWT token is displayed in the **Encode** area.

- **HEADER**: Set **alg** to **RS512**, enter **kid** in the JWK created in **Step 1**, and set **typ** to **JWT**.
- **PAYLOAD**: Set **iss** to **test** and **aud** to **ASM**. Ensure that the values are the same as the issuer and token audience configured in **Step 2**.
- **VERIFY SIGNATURE**: The value must be the same as the public key in **Step 1.1**.

Figure 9-5 Creating a JWT token



Step 2 Access the **httpbin** service through the ingress gateway.

1. Run the following commands to access the service with the JWT token created in **Step 1**:

   TOKEN=*JWT token created by the **Step 1***.

   curl -I -H "Authorization: Bearer $TOKEN" http:// {*External access address of the **httpbin** service*}/

   Expected outputs:

   ```
   HTTP/1.1 200 OK
   server: istio-envoy
   date: Wed, 21 Sep 2022 03:11:48 GMT
   ```

2. Run the following command to access the service with an invalid JWT token:

   curl -I -H "Authorization: Bearer invalidToken" http:// {*External access address of the **httpbin** service*}/

   Expected outputs:

   ```
   HTTP/1.1 401 Unauthorized
   www-authenticate: Bearer realm="http://***.***.***.***:***/", error="invalid_token"
   content-length: 145
   content-type: text/plain
   date: Wed, 21 Sep 2022 03:12:54 GMT
   server: istio-envoy
   x-envoy-upstream-service-time: 19
   ```

3. Modify the JWT authentication created in **Step 2**, leave the **aud** empty (indicating that the service can be accessed by any services), and run the following command to access the service with the JWT token created in **Step 1**:

   curl -I -H "Authorization: Bearer $TOKEN" http:// {*External access address of the **httpbin** service*}/

   Expected outputs:


```
HTTP/1.1 200 OK
server: istio-envoy
date: Wed, 21 Sep 2022 03:20:07 GMT
```

4. Run the following command to access the service without the JWT token:

   curl -I http:// {*External access address of the* ***httpbin*** *service*}/

   Expected outputs:

```
HTTP/1.1 403 Forbidden
content-length: 85
content-type: text/plain
date: Wed, 21 Sep 2022 03:29:31 GMT
server: istio-envoy
x-envoy-upstream-service-time: 6
```

According to the preceding outputs, the request with the correct JWT token can access the service, and the request with an incorrect JWT token or without a JWT token cannot access the service. This means the request identity authentication takes effect.

----End

