Application Service Mesh

Best Practices

Issue	01
Date	2022-07-01

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 Upgrading Data Plane Sidecars Without Service Interruption

ASM enables you to manage and monitor the traffic of services added into a service mesh. Sidecars are important components in ASM data plane. The upgrade of sidecars involves the re-injection of sidecars into data plane service pods, which requires the pods to be updated.

This section describes how to avoid service interruption during sidecar upgrade.

Configuring the Number of Service Pods

Ensure that the number of service pods is greater than or equal to **2** and the upgrade policy is set to **RollingUpdate**.

Sample configurations:

kubectl get deploy nginx -n *namespace_name* **-oyaml | grep strategy -a10**

```
spec:
  minReadySeconds: 2
  progressDeadlineSeconds: 600
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
      version: v1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
```

Configuration description:

- Number of service pods: deployment.spec.replicas >= 2
- Upgrade policy: deployment.spec.strategy.type == RollingUpdate
- Minimum number of alive pods in rolling upgrade: deployment.spec.replicas - deployment.spec.strategy.maxUnavailable > 0

Adding a Readiness Probe

Readiness probes help you ensure that new service pods take over traffic only when they are ready. This prevents access failures caused by unready new pods.

The configurations are as follows:

kubectl get deploy nginx -n *namespace_name* **-oyaml | grep readinessProbe -a10**

```
      readinessProbe:
        failureThreshold: 3
        httpGet:
          path: /
          port: 80
          scheme: HTTP
        initialDelaySeconds: 1
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 1
```

Configuration description:

Configuring a readiness probe: deployment.spec.template.spec.containers[i].readinessProbe

The configuration includes the initial check time, check interval, and timeout duration.

Setting the Service Ready Time

minReadySeconds is used to specify the minimum number of seconds for which a newly created pod should be ready without any of its containers crashing, for it to be considered available.

The configurations are as follows:

kubectl get deploy nginx -n *namespace_name* **-oyaml | grep minReadySeconds -a1**

```
spec:
  minReadySeconds: 2
  progressDeadlineSeconds: 600
```

Configuration description:

The service ready time: deployment.spec.minReadySeconds. Configure this parameter based on the live environment.

Configuring a Graceful Shutdown Time

terminationGracePeriodSeconds is used to configure a graceful shutdown time. During a rolling upgrade, the endpoint of an old service pod is removed and the pod status is set to **Terminating**. A SIGTERM signal is then sent to the pod. After the graceful shutdown time you configured, the pod will be forcibly terminated. The graceful deletion time allows the pod to keep processing unfinished requests, if any, to avoid hard termination.

kubectl get deploy nginx -n *namespace_name* **-oyaml | grep terminationGracePeriodSeconds -a1**

```
securityContext: {}
terminationGracePeriodSeconds: 30
tolerations:
```

Configuration description:

The graceful shutdown time: deployment.spec.template.spec.terminationGracePeriodSeconds. The default value is 30s. Configure this parameter based on the live environment.

Configuring the preStop

preStop enabled you to perform certain execution before a service pod is stopped. In this way, it helps you gracefully shut down a service pod. Configure this parameter based on service requirements. Nginx is used as an example here.

kubectl get deploy nginx -n *namespace_name* **-oyaml | grep lifec -a10**

```
lifecycle:
  preStop:
    exec:
      command: ["/bin/sh", "-c", "nginx -s quit; sleep 10"]
```

In the **lifecycle.preStop** field, the **nginx -s quit; sleep 10** command is defined. This command first sends a graceful shutdown signal to the Nginx process and then the pod termination pauses for 10 seconds. In this way, Nginx has enough time to complete the ongoing requests and gracefully close the pod before it terminates.

10 in **sleep 10** is an example value. You can change it based on the actual requirements and application performance. The key should be a proper value so that Nginx has enough time to gracefully shut down the process.

Alternatively, you can run custom commands or scripts to gracefully shut down your service process.

2 Service Governance for Dubbo-based Applications

2.1 Introduction

Dubbo is a special protocol which needs the following supports:

- Envoy on the service mesh data plane supports the parsing and traffic management of the Dubbo protocol.
- The mesh control plane supports the configuration of Dubbo governance rules to manage services such as grayscale release, load balancing, and access authorization.

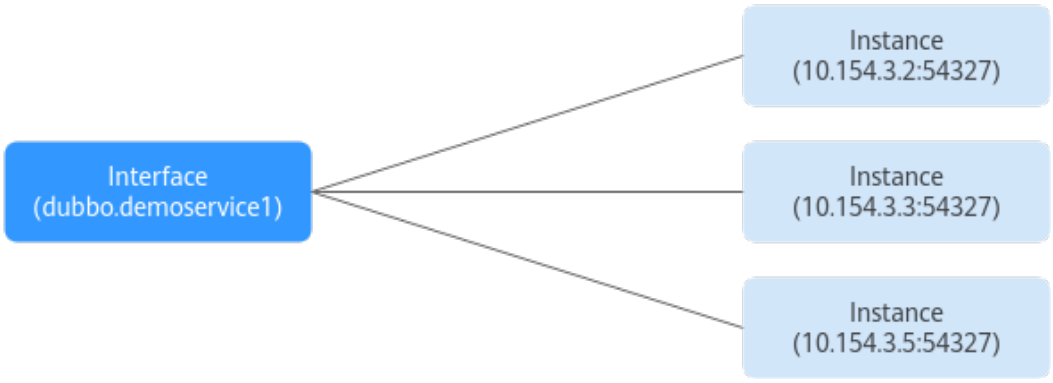
In addition, the service discovery model of Dubbo is different from that of Kubernetes and Spring Cloud. Therefore, additional processing is required.

2.2 Service Discovery Model

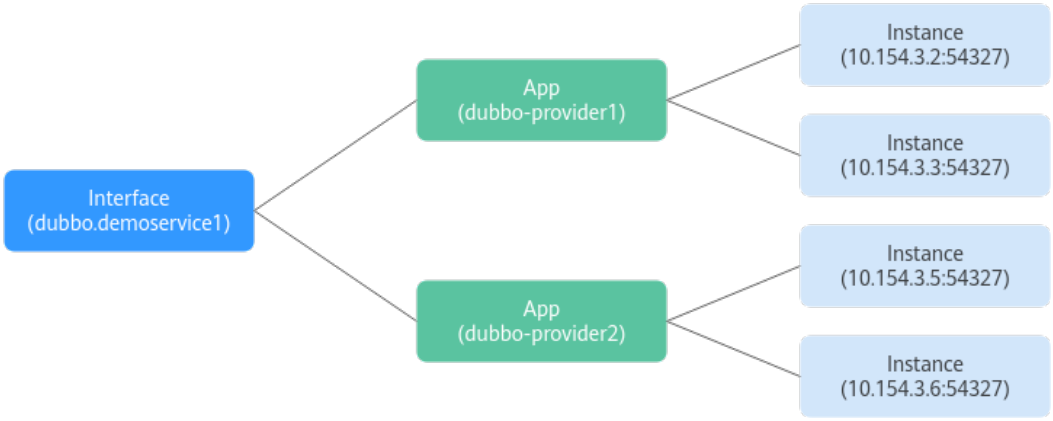
Problems in the existing Dubbo model (summarized from the Dubbo community version 2.7.4):

- In the microservice architecture, the Registry manages applications (services) instead of external service interfaces. However, the Dubbo Registry manages Dubbo service interfaces, contrary to the Spring Cloud or Cloud Native registration mode.
- A Dubbo application (service) allows N Dubbo service interfaces to be registered. The more interfaces, the heavier the load of the Registry.

The existing Dubbo service model searches for service instances based on the Dubbo interface.



The Dubbo Cloud Native service discovery model adds an app layer to search for instances.

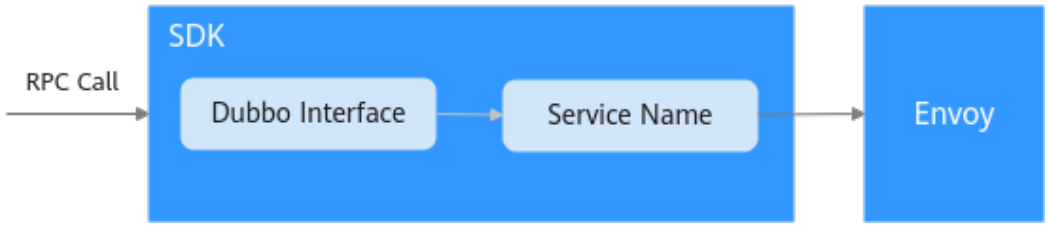


2.3 SDK Adaptation Mode

2.3.1 PASSTHROUGH Solution

Introduction

When the client in the SDK calls the target service by an interface, the client accesses the service name, instead of the service instance.



Description

Cases are different based on the Dubbo protocol versions:

- 2.7.4 and later versions: Cloud Native 2.7.4 and later versions have reconstructed the service discovery model which is consistent with that of Kubernetes. Service information can be directly obtained via interfaces.

- 2.7.3 and earlier versions: The Dubbo community versions do not provide the level-2 relationship between interfaces and services. The SDK needs to maintain the mapping from interfaces to services based on the actual usage mode. For example, information such as a service name may be provided in extended information during service registration.

You can select a processing mode based on your SDK. The SDK of an earlier version can perform the following operations in the existing service registration and discovery processes:

1. Extend the definition of **Service** in the registration information. During service deployment, service metadata can be injected into the SDK as environment variables, including **appname** and **namespace**, which indicate the name and namespace of the deployed service, respectively.
2. When the service is started, the relationship between the Dubbo interface and Kubernetes service name and namespace is registered in the Registry.
3. When a client initiates an access request, the service metadata is queried by the interface according to the original service discovery process, and the corresponding service information is used to assemble an RPC request. The extended field **Attachment** is advised to be used to store the **appname** and **namespace** information in the Dubbo request.

2.3.2 Static Target Service

Introduction

Use **dubbo:reference** to configure the referenced service provider in the service consumer of the Dubbo service. Use the **url** option to define the address of the point-to-point direct connection service provider to bypass the Registry and directly call the target service.

Description

If the original Dubbo service uses the **.xml** configuration file, only the configuration file needs to be modified.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>
    <!-- Interfaces that can be called -->
    <dubbo:reference id="helloService " interface="com.dubbo.service.HelloService " url = "dubbo://
helloService:20880" />
</beans>
```

If an annotation is used to define the referenced target service, only the annotation of the target service in the code needs to be modified.

```
@Reference(url = "dubbo://helloService:20880")
HelloService helloService;
```

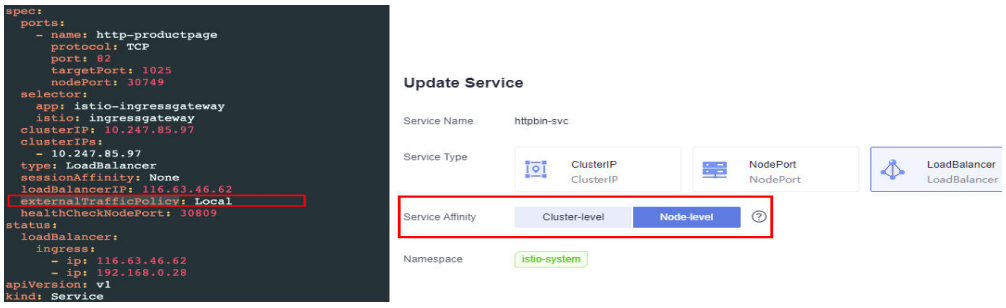

3 Reserving Source IP Address for Gateway Access

Scenario

When a service is accessed through a gateway, the source IP address displayed in the target container is not the source IP address of the client by default. To retain the source IP address, perform the operations described in this section.

Configuration

Log in to the CCE console. On the **Networking** page, select the **istio-system** namespace, update the gateway service associated with the service, and change **Service Affinity** to **Node-level**. The prerequisite is that the function of obtaining the client IP address of the ELB has been enabled (this function is enabled by default).



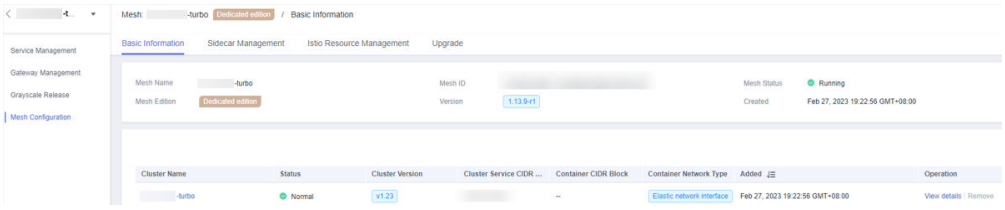
externalTrafficPolicy: indicates whether the Service wants to route external traffic to the local nodes or cluster-wide endpoints. Two options are available: Cluster (default) and Local. **Cluster** hides the client IP address, which may cause the second hop to another node, but has a good overall load distribution. **Local** retains the source IP address of the client and avoids the second hop of **LoadBalancer** and **NodePort** services. However, there is a potential risk of unbalanced traffic transmission.

Authentication Method

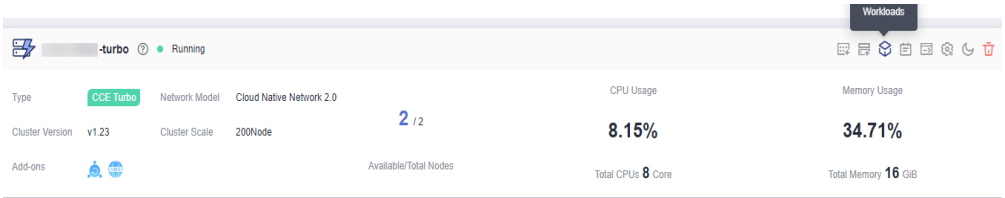
The **x-forward-for** field of the **httpbin** image displays the source IP address. The **httpbin** service is an HTTP Request & Response Service that can send requests to

the **httpbin** image. The **httpbin** image will return the requests based on the specified rules. You can search for the **httpbin** image in SWR. Before using the **httpbin** image for verification, ensure that the mesh function has been enabled for the cluster.

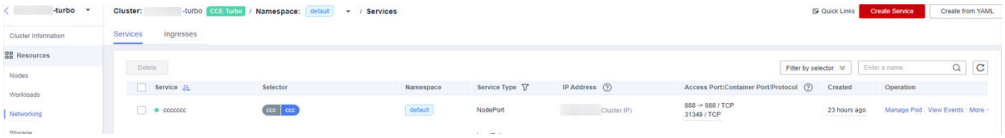
1. Log in to the ASM console, and click an available test mesh to go to its details page.
2. Choose **Mesh Configuration** on the left to view the associated cluster.



3. Click the cluster name to go to its details page. Click the third icon in the upper right corner of the cluster to go to the **Workloads** tab page.



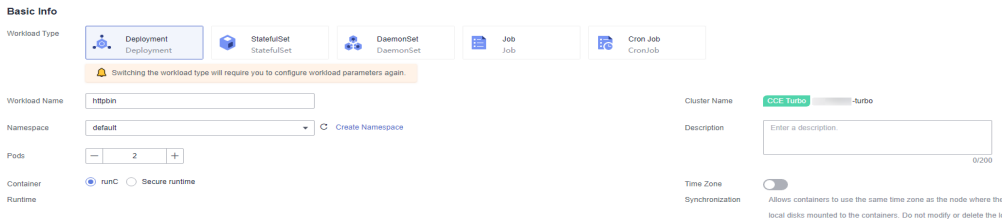
Click **Create Workload** in the upper right corner.



4. Configure workload information.

   Basic Information

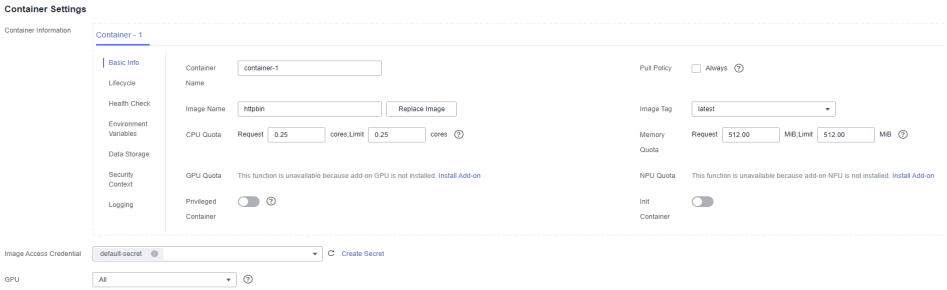
   - **Workload Type**: Select **Deployment**.
   - **Workload Name**: Enter **httpbin**.
   - **Namespace**: Select the namespace of the workload. The default value is **default**.
   - Retain the default values of other parameters.



   Container Configuration

   - **Basic Information**
     - **Container Name**: Customize a name.
     - **Image Name**: Click **Select Image**, search for the **httpbin** image, select the image, and click **OK**.

- **Image Tag**: Select an image tag.
- Retain the default values of other parameters.

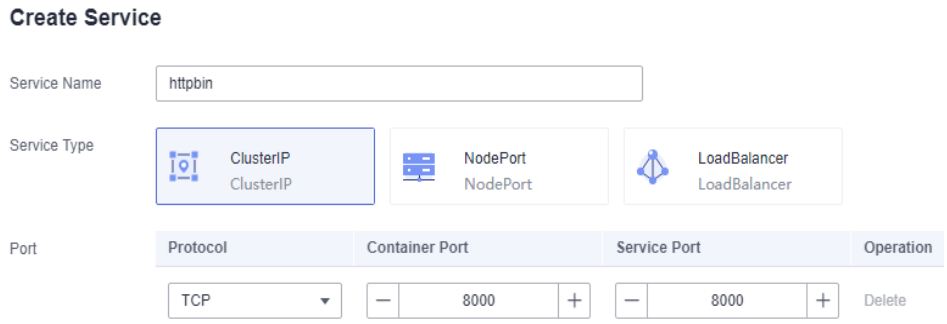


Service Settings

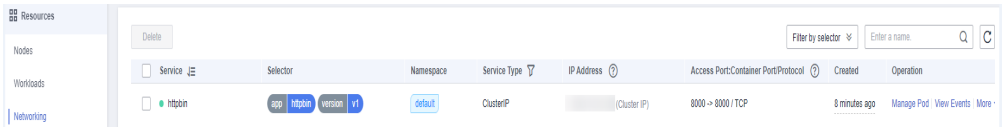
A Service solves the pod access problems. With a fixed IP address, a Service forwards access traffic to pods and performs load balancing for these pods.

Click **+** under the service configuration parameter to go to the **Create Service** page.

- **Service Name**: Enter the workload name.
- **Service Type**: Select **ClusterIP**.
- Port parameters:
  - **Protocol**: select **TCP**.
  - **Container Port**: Set it to **8000**.
  - **Service Port**: Set it to **8000**.

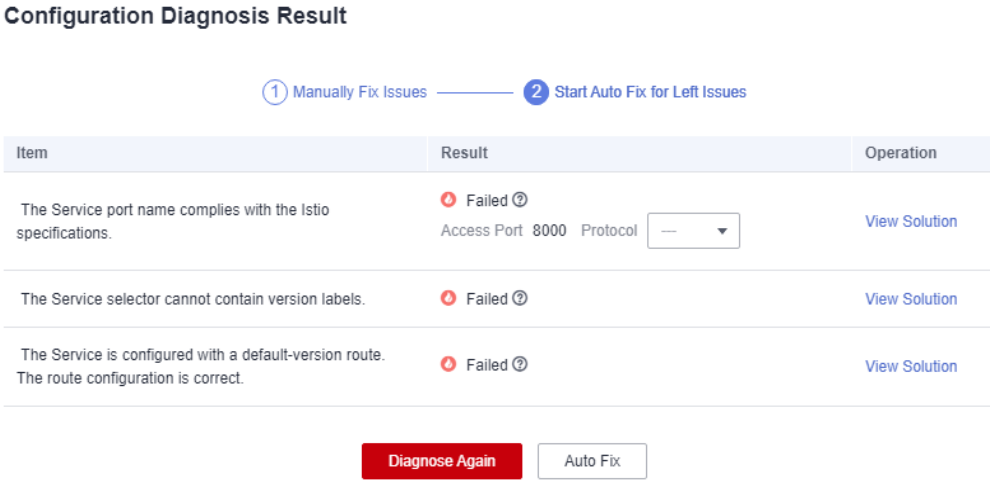


5. Click **OK** in the lower right corner.
6. Click **Create Workload** in the lower right corner.
7. On the cluster details page, click **Networking** on the left. The created **httpbin** service is displayed in the service list.

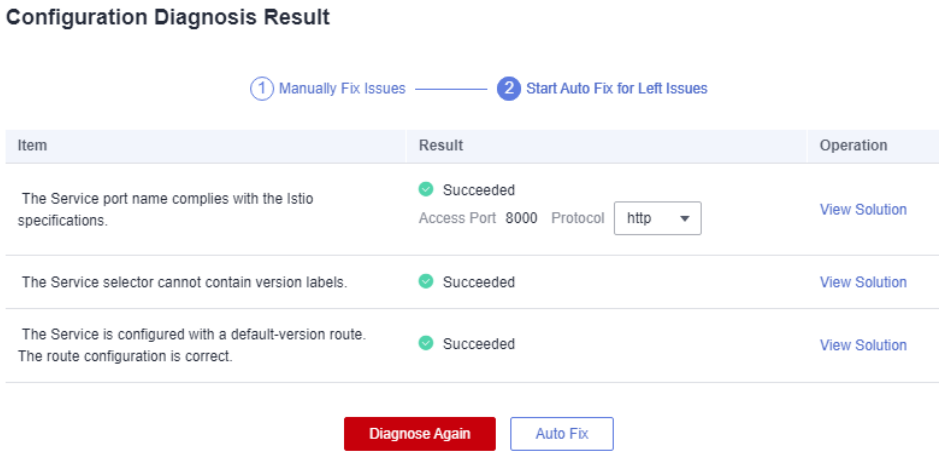


8. Return to the ASM console and click **Service Management**. The **Configuration Diagnosis Result** of the **httpbin** service is **Abnormal**.

9. Click **Fix** and rectify the faults by following the instructions provided in the displayed **Configuration Diagnosis Result** page.



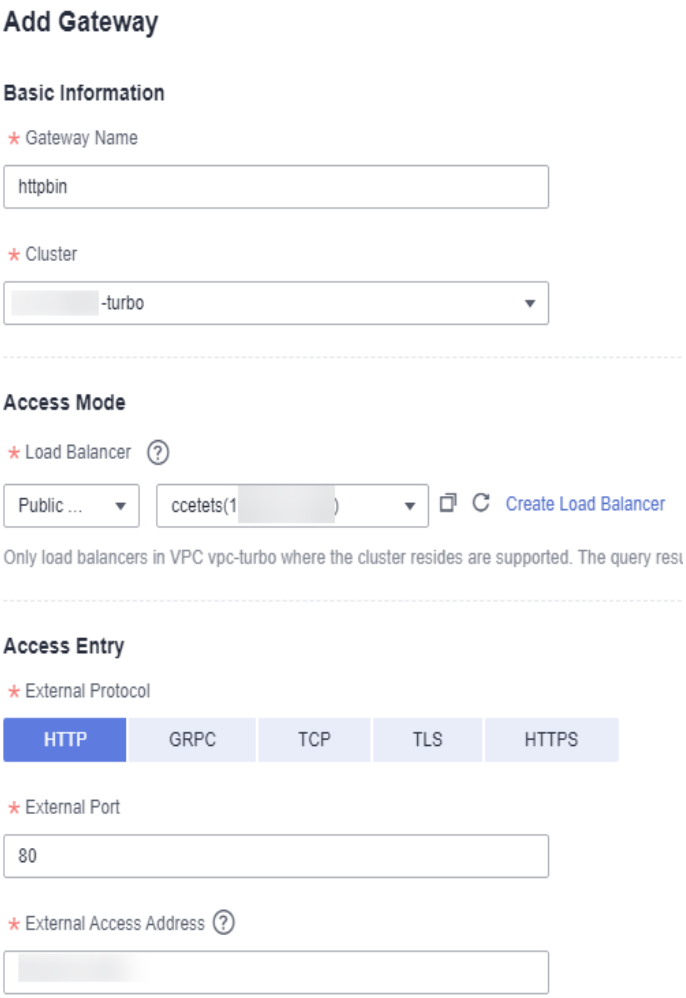
   - Take the "The Service port name complies with the Istio specifications" item as an example. Select **http** and click **Auto Fix**.



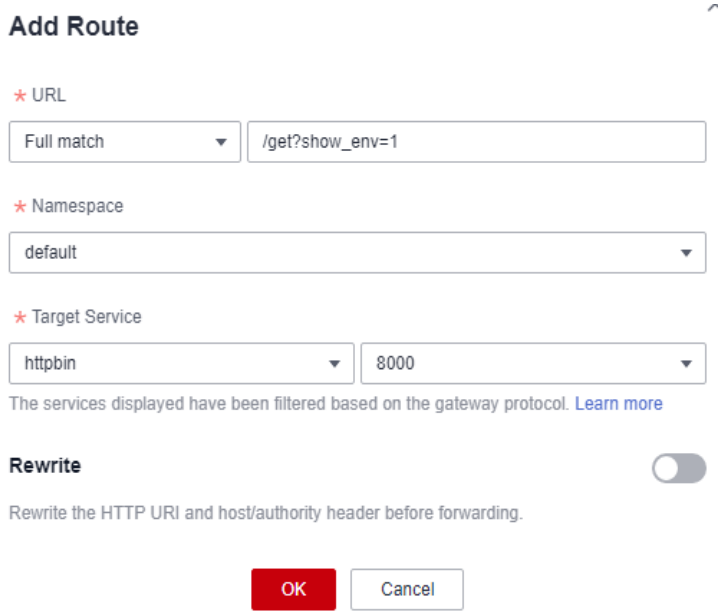
10. Click **Gateway Management** > **Add Gateway**. On the **Add Gateway** page displayed, configure the parameters.

    Configuration Information

    - **Gateway Name**: Enter **httpbin**.
    - **Cluster**: Select the cluster associated with the mesh.
    - **Load Balancer**: Select Public network and select a load balancer.
    - **External Protocol**: Select **HTTP**.
    - **External Port**: Specify a port.
    - **External Access Address**: Enter the public IP address of the load balancer selected for the **Load Balancer**.



- Click **+** under **Routing**. In the **Add Route** dialog box displayed, add a route.
  - **URL**: Select **Full match** and enter a mapping.
  - **Namespace**: Select the namespace to which the service belongs.
  - **Target Service**: Retain the default value.



When the configuration is complete, click **OK**.

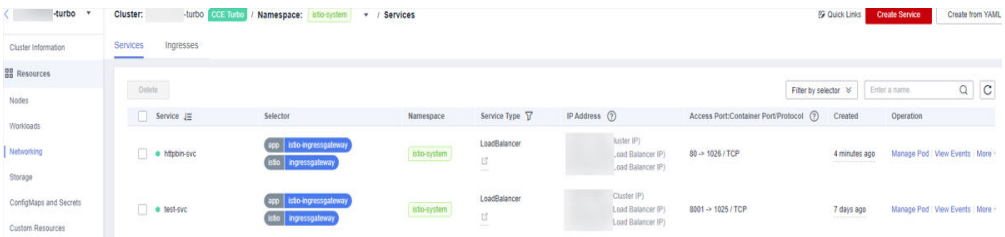
11. Click **OK**.
12. Click **Service Management** on the left. You can view the external access address of the created route in the **Access Address** column.

13. Click the external access address of the mapping you set during route configuration. You can view the IP address obtained by the gateway in the **x-forward-for** field, which is the IP address of the container CIDR block.

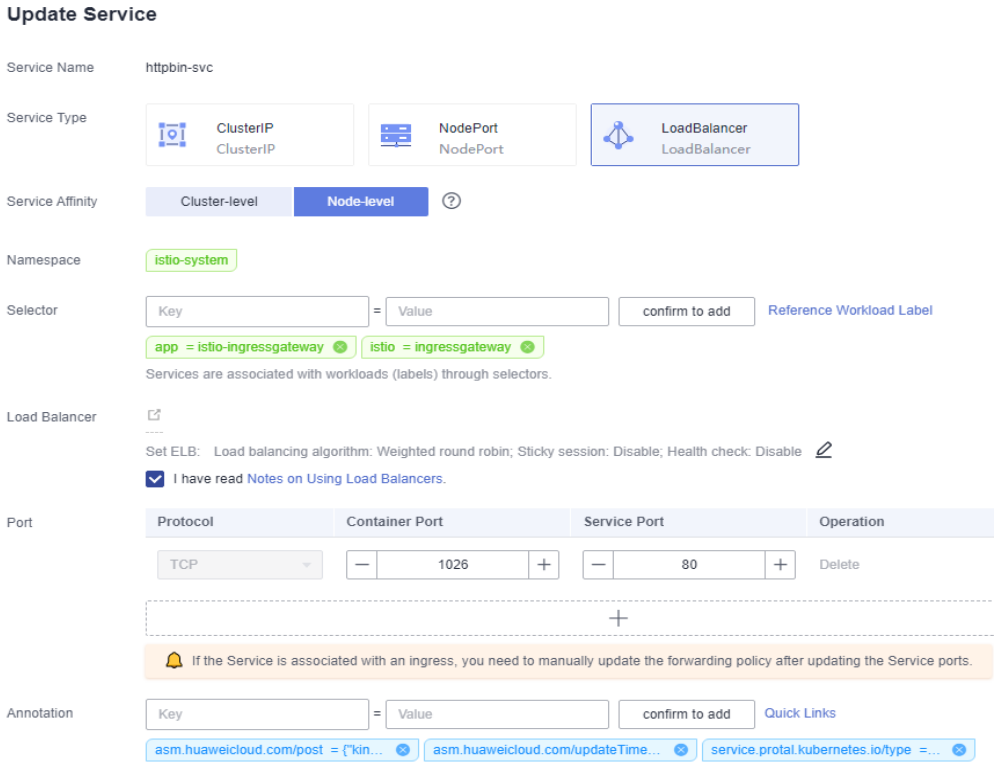
```
...
...
...
x-forward-for: xxxx
```

14. Return to the cluster details page, click **Networking** in the navigation tree on the left, and modify the configuration of the gateway service associated with the service as follows:

    Select the **istio-system** namespace.



    Click **More** > **Update** in the **Operation** column. On the **Update Service** page displayed, change **Service Affinity** to **Node-level**, select **I have read Notes on Using Load Balancers**, and click **OK**.



15. Return and refresh the external IP address accessed in **13**. If the IP address obtained by the gateway in the **x-forward-for** field is the source IP address of the local host, the verification is complete.

```
...
...
...
x-forward-for: xxxx
```

