

Tencent Cloud Mesh

FAQs

Product Documentation



Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

FAQs

Overview

Permissions

How to Add Mesh Permissions to a Cluster

Cluster Management by Sub-account Fails

Pod Startup Fails Because the Envoy Sidecar Is Unready

Envoy Status Code 431 "Request Header Fields Too Large" Is Returned

Envoy Converts Headers to Lowercase

Headless Service-related FAQs

Istio-init Crashes

VirtualService Does Not Take Effect

Inappropriate VirtualService Route Matching Order

Failed to Access an Ingress Gateway from the Public Network

Parsing Fails After Smart DNS Is Enabled

Locality Load Balancing Does Not Take Effect

Client Status Code 426 "Upgrade Required" Is Returned

Sidecars Are Not Automatically Injected

Circuit Breaking Does Not Take Effect

404 Is Returned During Access to a StatefulSet's Pod IP Address

Service Is Abnormal Due to a Retry Policy

Incomplete Call Tracing Information

FAQs

Overview

Last updated : 2023-12-26 14:22:13

[Pod Startup Fails Because the Envoy Sidecar Is Unready](#)

[Envoy Status Code 431 "Request Header Fields Too Large" Is Returned](#)

[Envoy Converts Headers to Lowercase](#)

[Headless Service-related FAQs](#)

[Istio-init Crashes](#)

[VirtualService Does Not Take Effect](#)

[Inappropriate VirtualService Route Matching Order](#)

[Failed to Access an Ingress Gateway from the Public Network](#)

[Parsing Fails After Smart DNS Is Enabled](#)

[Locality Load Balancing Does Not Take Effect](#)

[Cluster Management by Sub-account Fails](#)

[Client Status Code 426 "Upgrade Required" Is Returned](#)

[Sidecars Are Not Automatically Injected](#)

[Circuit Breaking Does Not Take Effect](#)

[404 Is Returned During Access to a StatefulSet's Pod IP Address](#)

[Service Is Abnormal Due to a Retry Policy](#)

[Incomplete Call Tracing Information](#)

[How to Add Mesh Permissions for a Cluster](#)

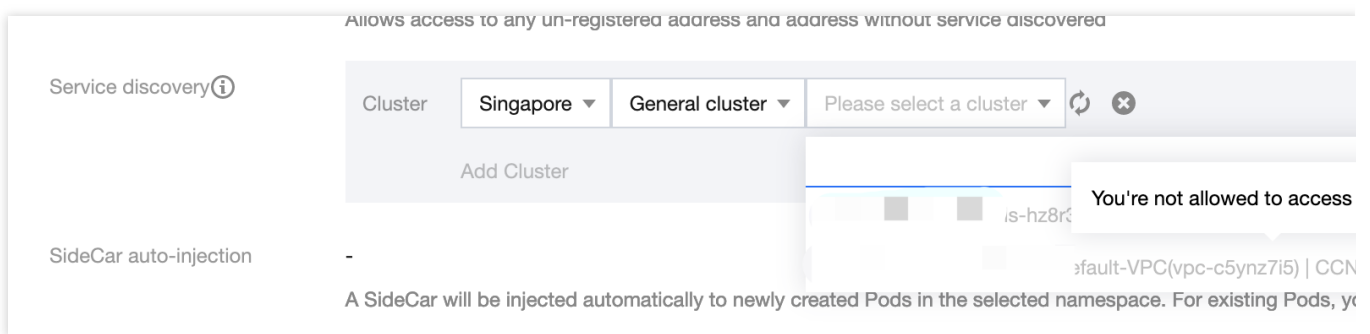
Permissions

How to Add Mesh Permissions to a Cluster

Last updated : 2023-12-26 14:23:40

Problem Description

When you try to operate a service mesh after logging in the Tencent Cloud Mesh console as a sub-account, you are prompted that there is no cluster permission.

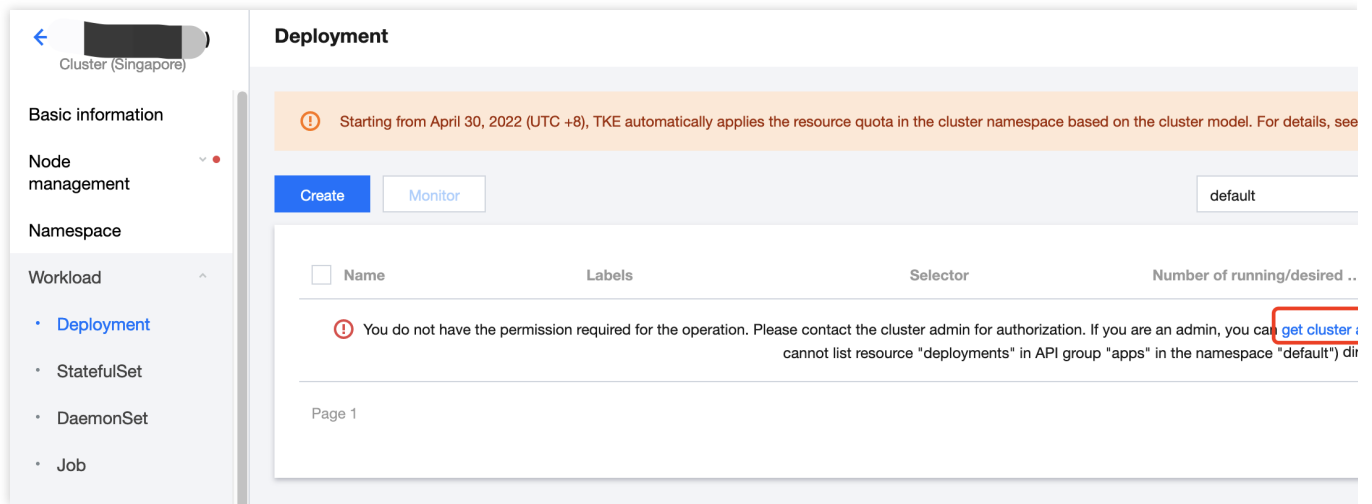


Solution

The currently logged-in sub-account does not have the admin permissions for the cluster. You can use the following method to authorize permissions for the cluster.

Authorization by the Admin

If your root account has admin permissions, you can use the **Get cluster admin role** feature directly on the [Tencent Cloud Mesh console](#) to quickly grant your sub-account admin permissions for the cluster.



Authorizing Other Sub-accounts

1. Log in to the Tencent Cloud console as the Tencent Cloud account that creates the cluster (this account will be used to authorize sub-accounts).
2. On the [TKE console](#), select the cluster.
3. On the cluster details page, choose **Authorization management > RBAC policy Generator**.

Basic information

Node management

Namespace

Workload

HPA

Service and route

Configuration management

Authorization management

- ClusterRole
- ClusterRoleBinding**
- Role
- RoleBinding
- ServiceAccount









Storage

Add-on management

Starting from April 30, 2022 (UTC +8), TKE automatically applies the resource quota in the namespace.

RBAC Policy Generator

Get cluster admin role

Name	Labels
 ClusterRole	cloud.tencent.com/tke-account: [redacted]
 cbs-csi-controller-binding	app.kubernetes.io/managed-by:Helm
 cbs-csi-node-binding	app.kubernetes.io/managed-by:Helm
 lb-ingress-clusterrole-nisa-bin...	-
 system:kube-proxy	-
 tke-bridge-agent	-
 tke-cni-clusterrole-binding	-
 tke-monitor-agent	app.kubernetes.io/managed-by:Helm

4. In the pop-up window, select the sub-account to be authorized.

1 Select an account > 2 Cluster RBAC settings

Account type

Sub-account

Service role

Select an account

Sub-account list18/18 loaded1 item selected

Separate filters with carriage return

Username

Sub-account

☐ Root Account

2l

☐

2l

☐

20l

☒

☐

☐ an

Press and hold Shift key to select more

5. Click **Done** to authorize admin permissions.

✓

Select an account

>

2

Cluster RBAC settings

Selected account

willzgli

Permission settings

Namespace list	Permission
<div>All namespaces</div>	<div>Admin</div>

Add permission

Permission description

Admin

Own the read and write permissions over resources in all namespaces; read and write permissions over cluster nodes, volumes, permissions

Ops team

Own the read and write permissions over resources in all namespaces; read and write permissions over cluster nodes, volumes, Developer

Read-only

Owns the read and write permission for resources visible in the console of all namespaces or selected name spaces

users

Owns the read-only permission for resources visible in the console of all namespaces or selected name spaces

Custom

The permission is subject to the selected ClusterRole. Please make sure that permissions of the selected

Cluster Management by Sub-account Fails

Last updated : 2023-12-26 14:24:14

When you try to operate a service mesh after logging in the Tencent Cloud console as a sub-account, you may encounter permission problems. For example, if you fail to associate a cluster, you will be prompted for insufficient permission. You will also be prompted for insufficient permission when viewing services.

Cause

The core point is that the currently logged-in sub-account needs to have the permission to operate the clusters associated with the mesh. If the sub-account does not have the permission to operate all or some clusters, you need to authorize the sub-account (authorization on each cluster is required). Refer to the following authorization method.

Cluster Authorization Method

1. Log in to the Tencent Cloud console as the Tencent Cloud account that creates the cluster (this account will be used to authorize sub-accounts), enter the [TKE console](#), and then choose **Authorization management > RBAC policy generator**.
2. Select the sub-account to be authorized.
3. Grant the admin permissions.

Pod Startup Fails Because the Envoy Sidecar Is Unready

Last updated : 2023-12-26 14:25:33

In the process of migrating some services to Istio, pods may fail to start and then keep restarting. Through troubleshooting, it is found that other services (for example, pulling configurations from the configuration center) need to be called when such a service starts. In addition, the service will exit upon a call failure because there is no retry logic. The call failure reason is that Envoy is unready (it takes a while for Envoy to pull the configurations from the control plane). As a result, the traffic sent by the service cannot be processed. (For details, see k8s issue [#65502](#).)

Best Practices

At present, the best practice for this problem is to make applications more robust. You can add retry logic, so that the applications do not exit immediately after the call fails. Alternatively, you can add the sleep time before the startup command, so that the applications will wait for a few seconds.

If you do not want to modify your applications, refer to the following workaround solution.

Workaround Solution

The workaround solution is to adjust the sidecar injection sequence.

In Istio 1.7, this problem is resolved by adding a switch named `HoldApplicationUntilProxyStarts` to the istio-injector injection logic. After the switch is enabled, the proxy will be injected into the first container.

Example:

```
// Boolean flag for enabling/disabling the holdApplicationUntilProxyStarts behavior
// This feature adds hooks to delay application startup until the pod proxy
// is ready to accept traffic, mitigating some startup race conditions.
// Default value is 'false'.
HoldApplicationUntilProxyStarts *types.BoolValue `protobuf:"bytes,33,opt,name=holdApplicationUntilProxyStarts" json="holdApplicationUntilProxyStarts"``
```

```
// nolint: staticcheck
holdPod := mc.DefaultConfig.HoldApplicationUntilProxyStarts.GetValue() ||
    valuesStruct.GetGlobal().GetProxy().GetHoldApplicationUntilProxyStarts().Get

proxyLocation := MoveLast
// If HoldApplicationUntilProxyStarts is set, reorder the proxy location
if holdPod {
    proxyLocation = MoveFirst
}
```

View the template used by istio-injector for automatic injection. It is found that if

`HoldApplicationUntilProxyStarts` is enabled, a `postStart` hook is added to the sidecar.

```
{{- if .Values.global.proxy.lifecycle }}
  lifecycle:
    {{ toYaml .Values.global.proxy.lifecycle | indent 4 }}
{{- else if $holdProxy }}
  lifecycle:
    postStart:
      exec:
        command:
          - pilot-agent
          - wait
{{- end }}
```

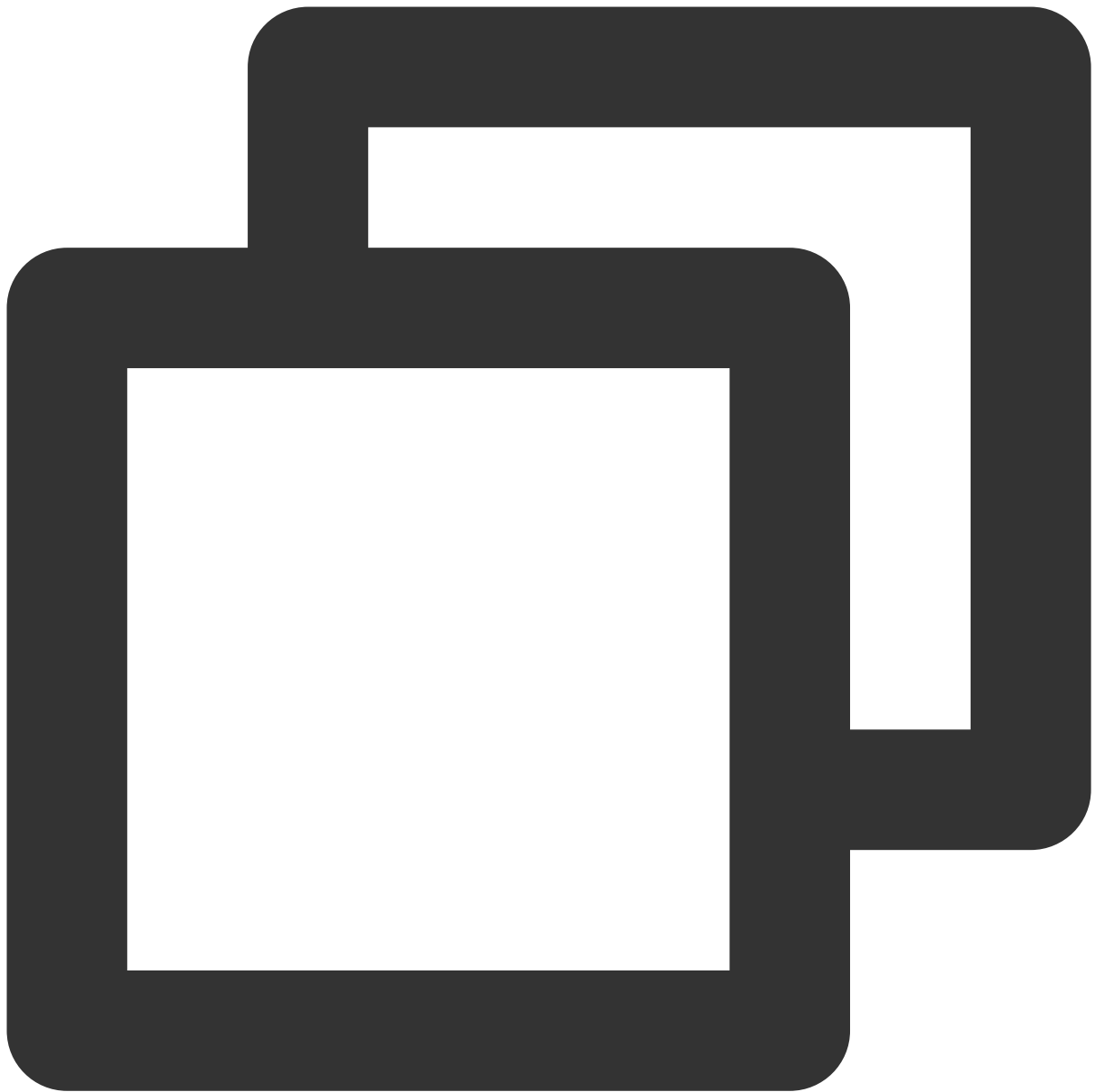
The hook is used to block the startup of the subsequent service containers, so that the subsequent service containers will start only after the sidecar is fully started.

The switch configuration supports global configuration and local configuration, which can be enabled as follows:

Global Configuration

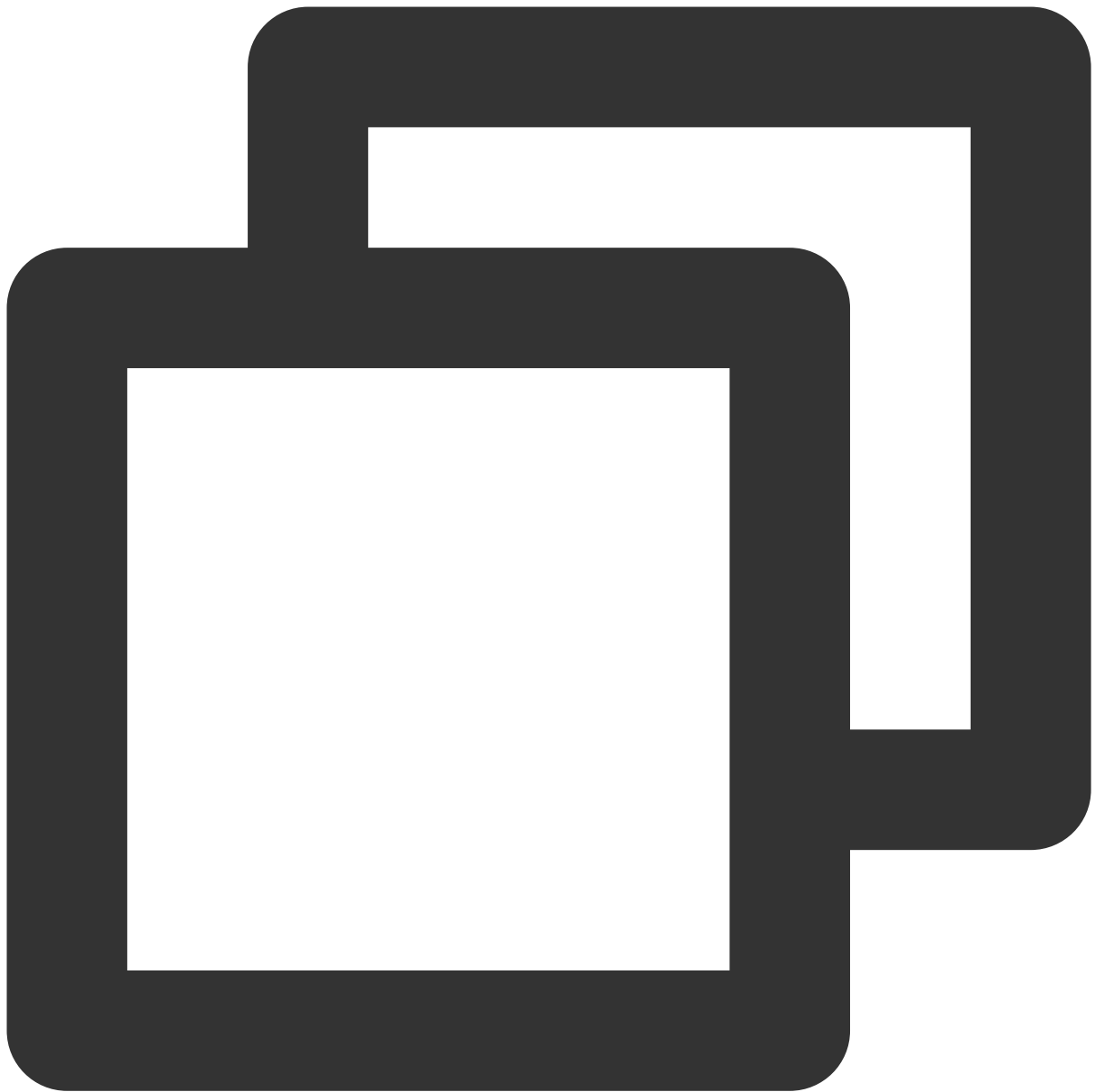
Local Configuration

Modify Istio's global ConfigMap configurations.



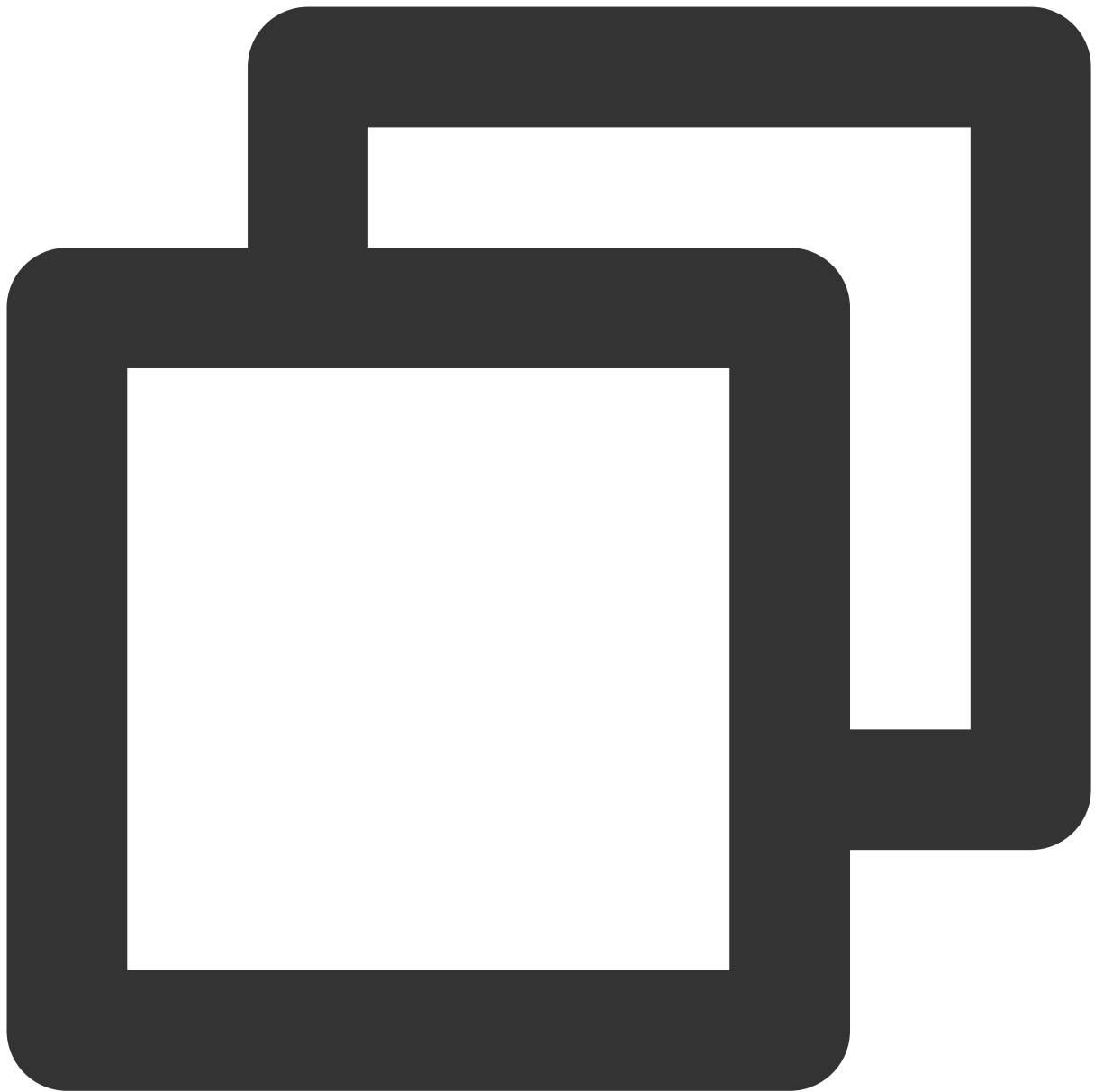
```
kubectl -n istio-system edit cm istio
```

Add `holdApplicationUntilProxyStarts: true` under `defaultConfig` .



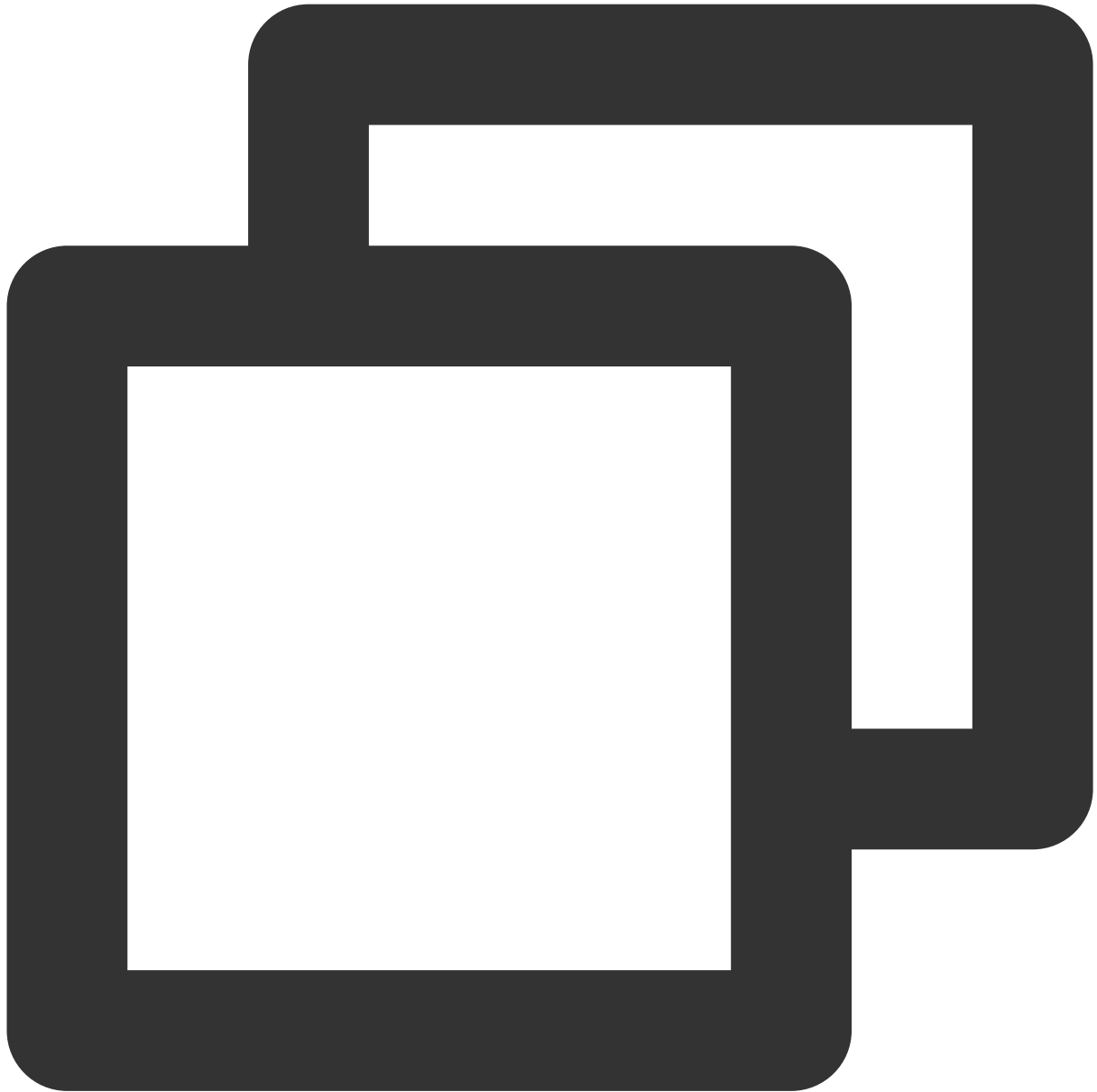
```
apiVersion: v1
data:
  mesh: |-
    defaultConfig:
      holdApplicationUntilProxyStarts: true
    meshNetworks: 'networks: {}'
kind: ConfigMap
```

If IstioOperator is used, modify defaultConfig under the CR field `meshConfig` .



```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
metadata:
  namespace: istio-system
  name: example-istiocontrolplane
spec:
  meshConfig:
    defaultConfig:
      holdApplicationUntilProxyStarts: true
```

If you are using Istio 1.8 or later, you can add the `proxy.istio.io/config` annotation to the pod for which the switch needs to be enabled and set `holdApplicationUntilProxyStarts` to `true`. For example:



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
```



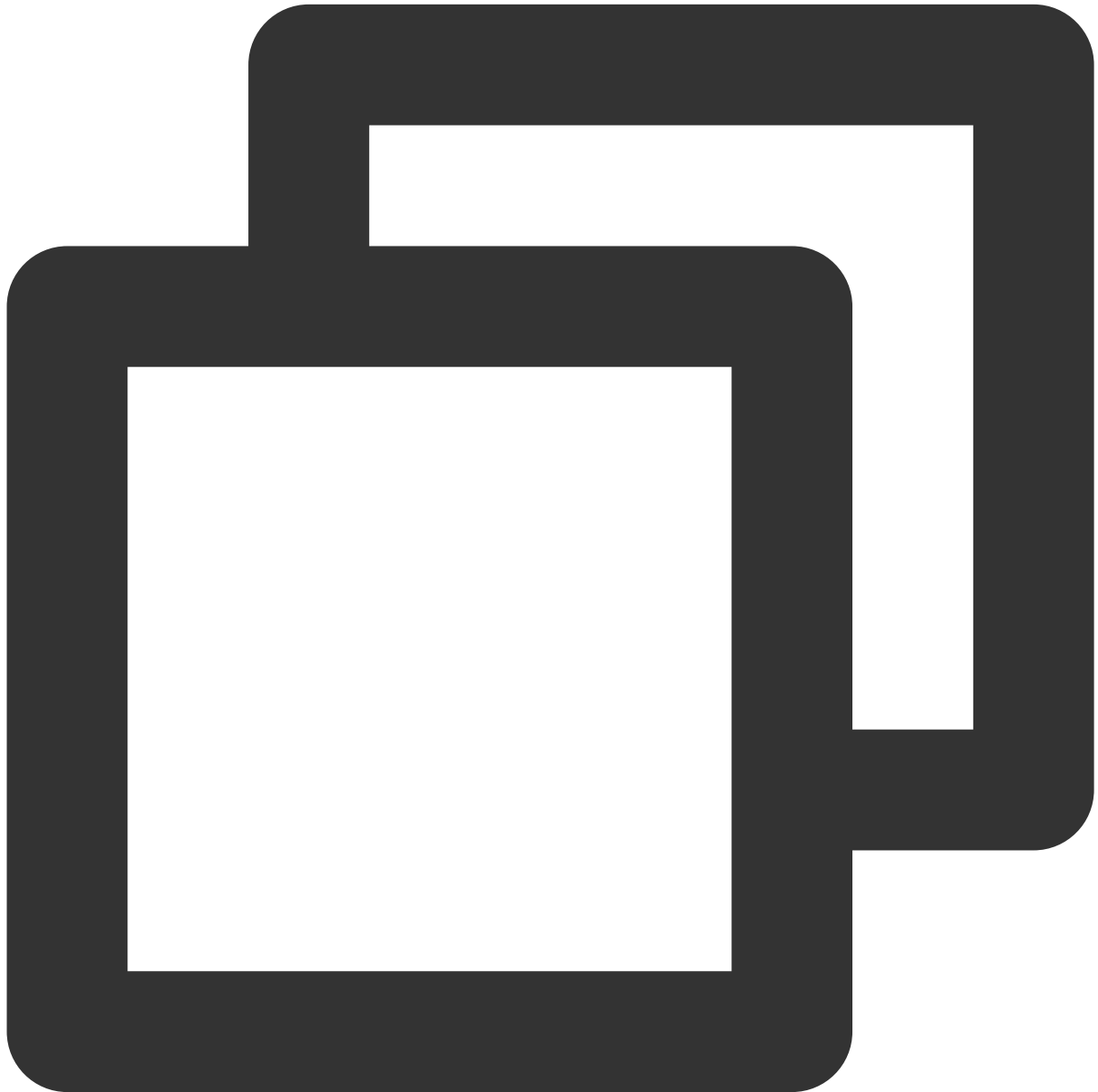
```
    app: nginx
template:
  metadata:
    annotations:
      proxy.istio.io/config: |
        holdApplicationUntilProxyStarts: true
    labels:
      app: nginx
  spec:
    containers:
      - name: nginx
        image: "nginx"
```

Note that after the switch is enabled, the service containers need to wait for the sidecar to be completely ready before they can be started. As a result, pods will start at a slower speed, which may be difficult in scenarios where rapid capacity expansion is required to cope with burst traffic. Therefore, it is recommended to evaluate the service scenarios by yourself and use the local configuration method to enable this switch only for the required service.

Envoy Status Code 431 "Request Header Fields Too Large" Is Returned

Last updated : 2023-12-26 14:32:01

For an HTTP request in Istio, Envoy returns abnormal status code 431.



```
HTTP/1.1 431 Request Header Fields Too Large
```

Cause Analysis

This status code indicates that the size of the HTTP request header exceeds the limit. The default limit is 60 KiB. The limit is determined by the `max_request_headers_kb` field in the `HttpConnectionManager` configuration, and its maximum value can be set to 96 KiB.

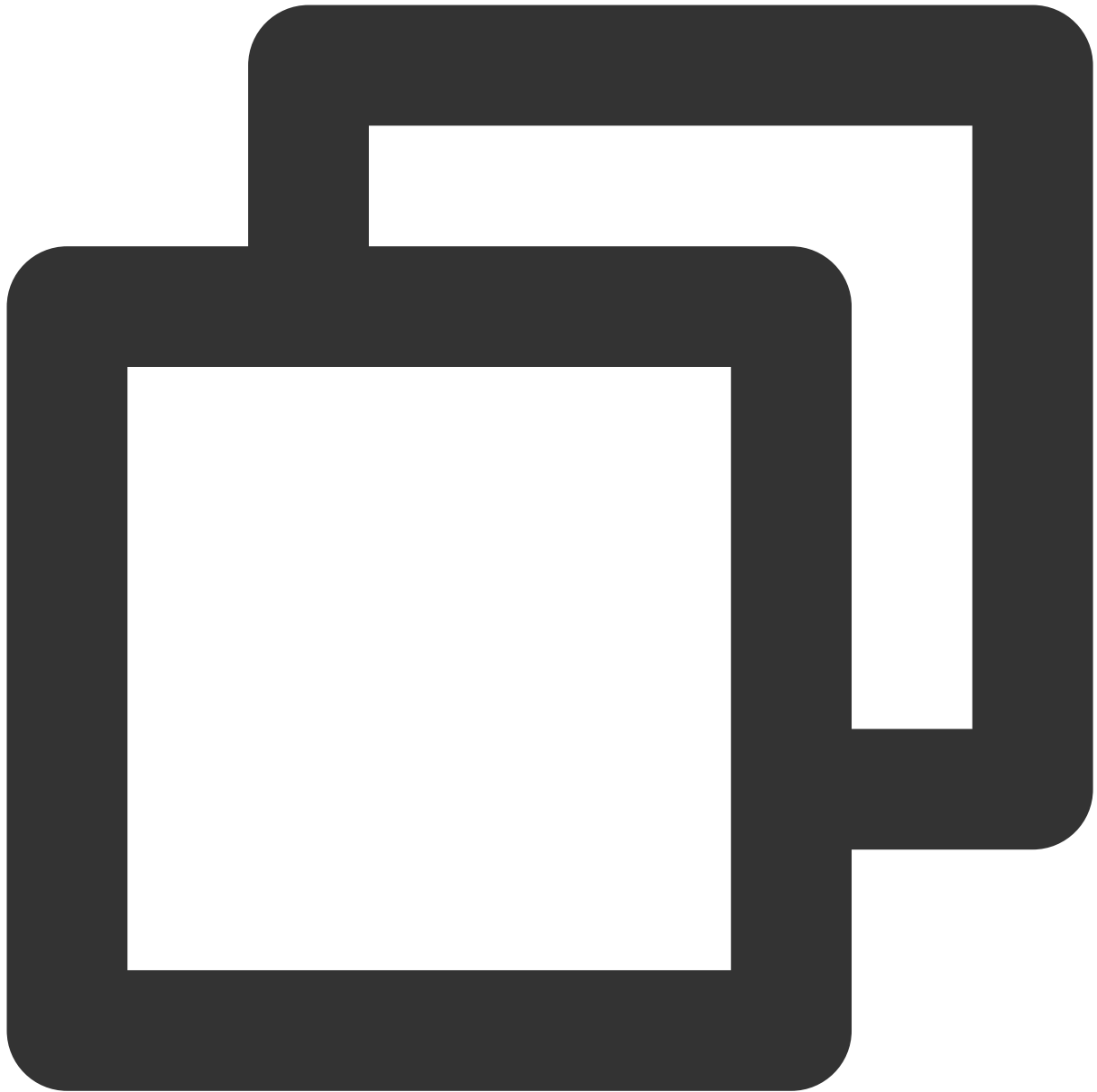
`max_request_headers_kb`

([UInt32Value](#)) The maximum request headers size for incoming connections. If unconfigured, headers allowed is 60 KiB. Requests that exceed this limit will receive a 431 response. The limit is based on current implementation constraints.

Solution

The header size limit can be increased by adjusting the `max_request_headers_kb` field through the `EnvoyFilter`.

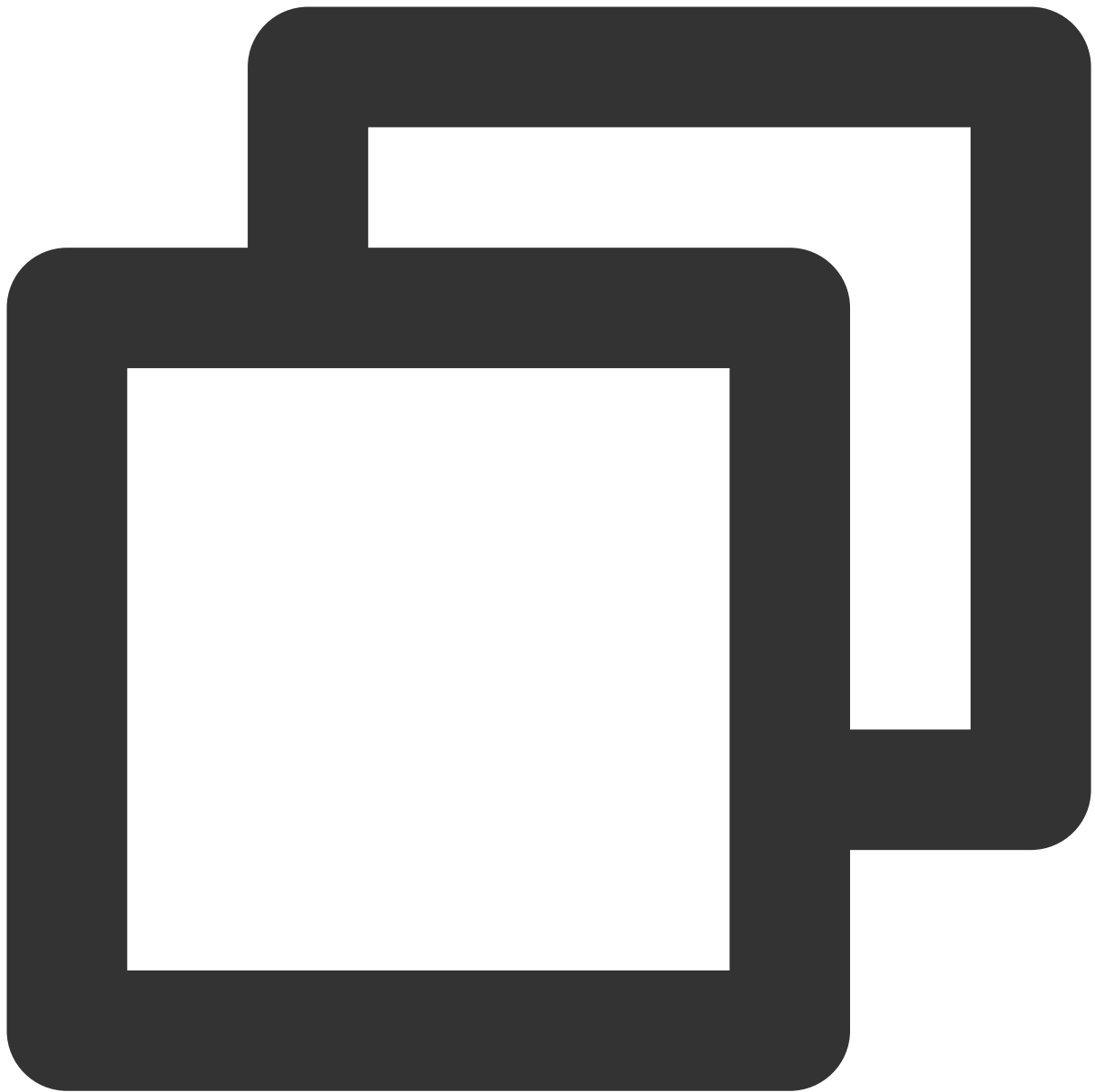
An `EnvoyFilter` example (verification in Istio 1.6 has passed):



```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: max-header
  namespace: istio-system
spec:
  configPatches:
  - applyTo: NETWORK_FILTER
    match:
      context: ANY
      listener:
```

```
    filterChain:
      filter:
        name: "envoy.http_connection_manager"
  patch:
    operation: MERGE
    value:
      typed_config:
        "@type": "type.googleapis.com/envoy.config.filter.network.http_connection
        max_request_headers_kb: 96
```

A later version is compatible with the v2 configurations above, but it is recommended to use v3 configurations (verification in Istio 1.8 has passed).



```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: max-header
  namespace: istio-system
spec:
  configPatches:
  - applyTo: NETWORK_FILTER
    match:
      context: ANY
      listener:
```

```
    filterChain:
      filter:
        name: "envoy.http_connection_manager"
  patch:
    operation: MERGE
    value:
      typed_config:
        "@type": "type.googleapis.com/envoy.extensions.filters.network.http_conne
        max_request_headers_kb: 96
```

If the size of a header exceeds 96 KiB, it is recommended to place this part of data in the body.

Envoy Converts Headers to Lowercase

Last updated : 2023-12-26 14:32:23

Envoy will convert the key of an HTTP header to lowercase by default. For example, the HTTP header `Test-Upper-Case-Header: some-value` will be changed to `test-upper-case-header: some-value` after passing through the Envoy proxy. This is not a problem under normal conditions. [RFC 2616](#) also states that HTTP header processing is case-insensitive.

Scenarios in Which HTTP Headers Are Case-sensitive

Usually, converting a header to lowercase does not bring any specification problem. However, in some cases, header case-sensitivity may cause some problems, for example:

Header parsing is case-sensitive.

The SDK used is sensitive to the letter case of headers. For example, when reading `Content-Length` to determine the length of a response, the SDK depends on the capitalization of first letters.

Rules Supported Envoy

Envoy supports only two rules:

All lowercase (rule used by default), for example, `test-upper-case-header: some-value`

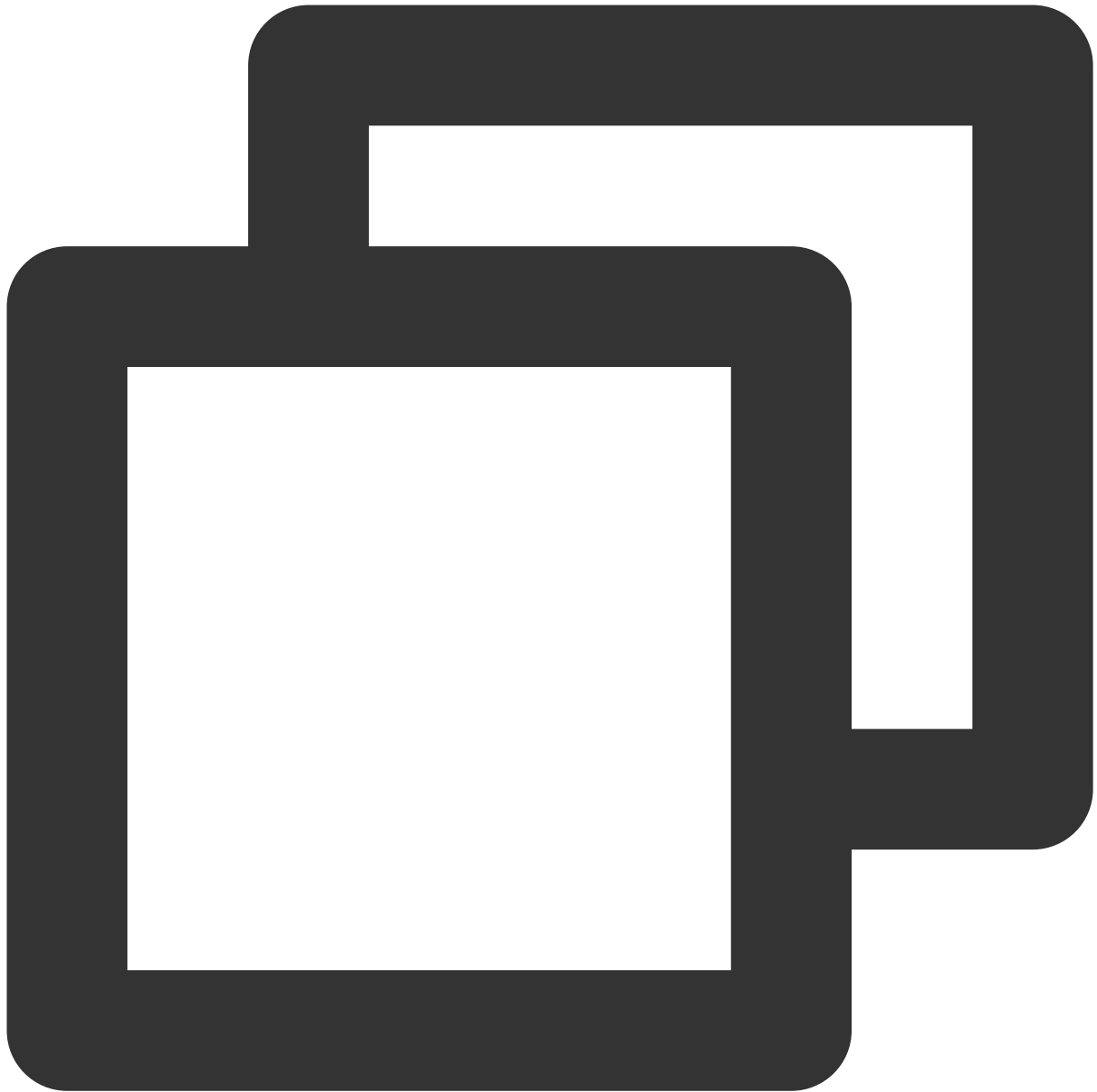
Capitalization of first letters (not enabled by default), for example, `Test-Upper-Case-Header: some-value`

If the letter case of an HTTP header of an application is completely irregular, for example, `Test-UPPER-CASE-Header: some-value`, the header cannot be compatible.

Workaround Solution

To work around this problem, forcibly specify the TCP protocol. To be specific, declare the protocol of the service as TCP to prevent Istio from performing 7-layer processing. In this way, Istio will not change the letter case of the HTTP header. However, it should be noted that the Istio's 7-layer capability will also become invalid.

If the service is deployed in a cluster, add a "tcp" prefix to the name of a port of the service. For example:



```
kind: Service
metadata:
  name: myservice
spec:
  ports:
    - number: 80
      name: tcp-web # Set the protocol used by the port to TCP.
```

If the service is deployed outside a cluster, you can forcibly specify the service as a TCP service through a ServiceEntry similar to the following to prevent Envoy from performing 7-layer processing on the service:



```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: qcloud-cos
spec:
  hosts:
    - "private-1251349835.cos.ap-guangzhou.myqcloud.com"
  location: MESH_INTERNAL
  addresses:
    - 169.254.0.47
  ports:
```

```
- number: 80
  name: tcp
  protocol: TCP
  resolution: DNS
```

Best Practices

If you want Envoy to enable the first letter capitalization rule for headers in some requests, you can use EnvoyFilter to set the header rule to capitalization. For example:



```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: http-header-proper-case-words
  namespace: istio-system
spec:
  configPatches:
    - applyTo: NETWORK_FILTER # http connection manager is a filter in Envoy
      match:
        # context omitted so that this applies to both sidecars and gateways
        listener:
```

```
name: XXX # Listener name used by cos, which can be queried from config_dump
filterChain:
  filter:
    name: "envoy.http_connection_manager"
patch:
  operation: MERGE
  value:
    name: "envoy.http_connection_manager"
    typed_config:
      "@type": "type.googleapis.com/envoy.config.filter.network.http_connection
      http_protocol_options:
        header_key_format:
          proper_case_words: {}
```

Note :

`listener name` must be set based on actual conditions.

Suggestions

Applications should comply with [RFC 2616](#) and follow the principle that HTTP header processing is case-insensitive.

Headless Service-related FAQs

Last updated : 2023-12-26 14:32:44

404 Is Returned for an Inter-Service Call Through the Registry

Symptom

After traditional services (for example, Spring Cloud) are migrated to Istio, 404 is returned for an inter-service call.

Cause

The mesh does not use Kubernetes' service discovery, but obtains service IP addresses from the registry. A call between services is directly issued to the obtained destination IP addresses without domain name resolution. Istio's LDS will intercept a request with PodIP+Port in the headless service, and then match the hosts field in the request. If there is no hosts field or the hosts field does not contain the domain name (for example, the pod IP address) of the service identified by PodIP+Port, the matching will fail and finally 404 is returned.

Solution

1. Register service domain names but not pod IP addresses in the registry.
2. Enable the client to carry the hosts field in requests. (This requires code to be updated.)

Load Balancing Policy Does Not Take Effect

Istio passes through the headless service by default, and forwards it by using `ORIGINAL_DST`, that is, Istio forwards the headless service directly to the original destination IP address without load balancing. Therefore, `trafficPolicy.loadBalancer` configured in `DestinationRule` will not take effect, which will affect the following features:

Session persistence (consistentHash)
Locality load balancing (localityLbSetting)

Solution

Create another non-headless service.

Failed to Access the Headless Service Without a Sidecar

Symptom

The client (with a sidecar) fails to access the server (without a sidecar) through the headless service. It is found that the response_flags field in access logs is `UF, URX` .

Cause

Istio 1.5/1.6 has a bug in headless service support. To be specific, regardless of whether an endpoint has a sidecar or not, mTLS is always enabled, which causes the access to the headless service without a sidecar (such as redis) to be denied (see [#21964](#) for details). For more details, see [Istio Operations Practices \(2\): Troublesome Headless Service - Part 1](#).

Solution

Solution 1: Configure `DestinationRule` in which mTLS is disabled.



```
kind: DestinationRule
metadata:
  name: redis-disable-mtls
spec:
  host: redis.default.svc.cluster.local
  trafficPolicy:
    tls:
      mode: DISABLE
```

Solution 2: Upgrade Istio to v1.7 or later.

Failed to Access a Rebuilt Pod

Symptom

The client can access the server through the headless service. After the pod of the server is rebuilt, the client fails to access the server through the headless service. It is found that the `response_flags` field in access logs is `UF,URX`.

Cause

Istio 1.5 has a bug in headless service support.

The client resolves the headless service through DNS, returns one of pod IP addresses, and initiates a request.

Envoy detects that the service is a headless service and forwards it by using `ORIGINAL_DST`. To be specific,

Envoy does not perform load balancing but forwards it directly to the original destination IP address.

When the pod of the headless service is rebuilt, because the client has a long connection with its sidecar (Envoy), the connection on the client side is not interrupted.

Because of the long connection, the client continues to send requests without performing DNS resolution. The requests are still sent to the old pod IP address that is obtained through previous resolution.

Because the old pod has been destroyed, Envoy returns an error code 503.

The client is not disconnected although the server returns an error, and continues to send subsequent requests to the old pod IP address. This cycle continues, and requests always fail.

For more details, see [Istio Operations Practices \(3\): Troublesome Headless Service - Part 2](#).

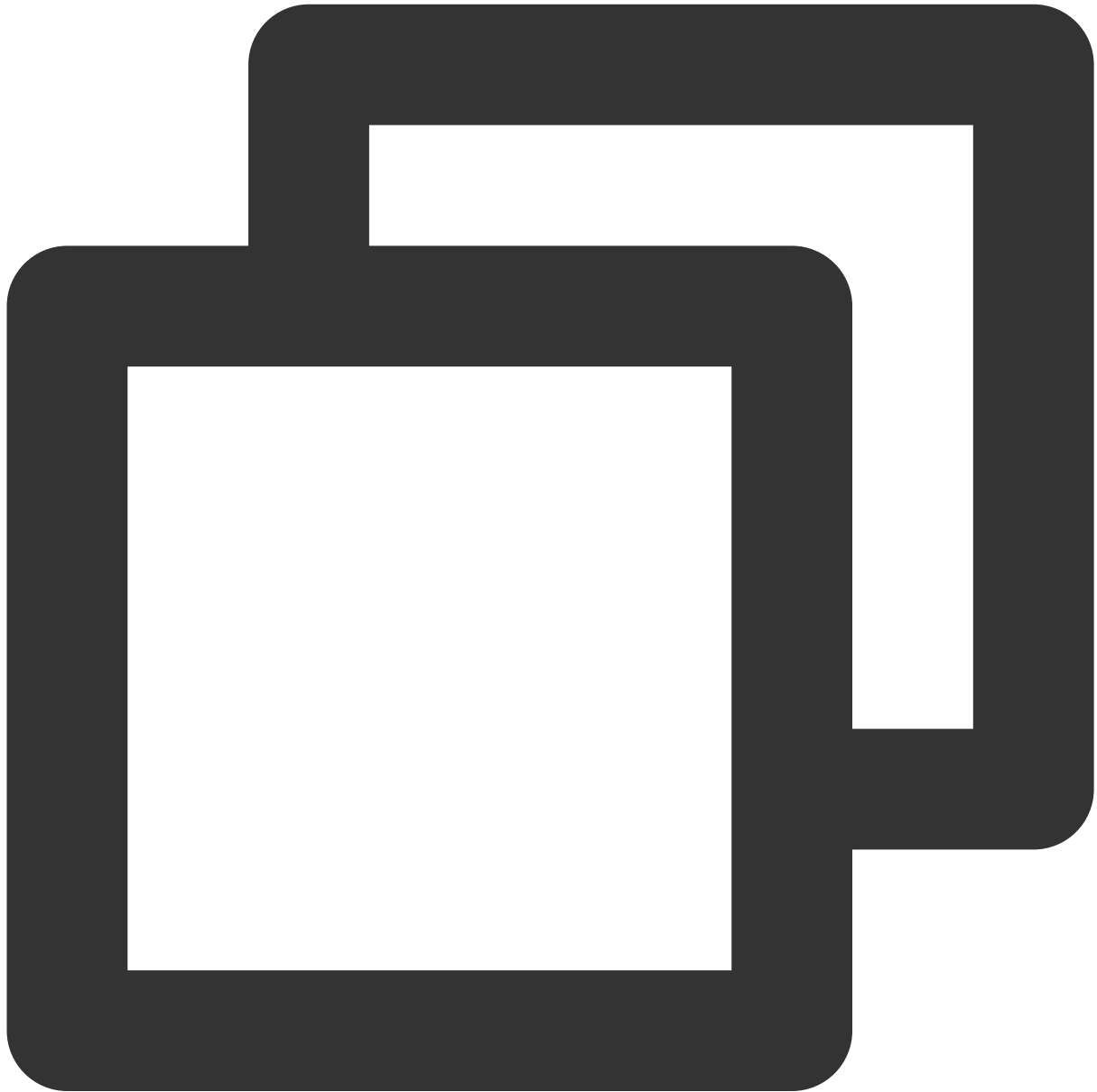
Solution

Upgrade Istio to v1.6 or later. Envoy will actively disconnect the long connection with downstream after the upstream connection is interrupted.

Istio-init Crashes

Last updated : 2023-12-26 15:27:17

In the Istio environment, there is a pod in the `Init:CrashLoopBackOff` state.

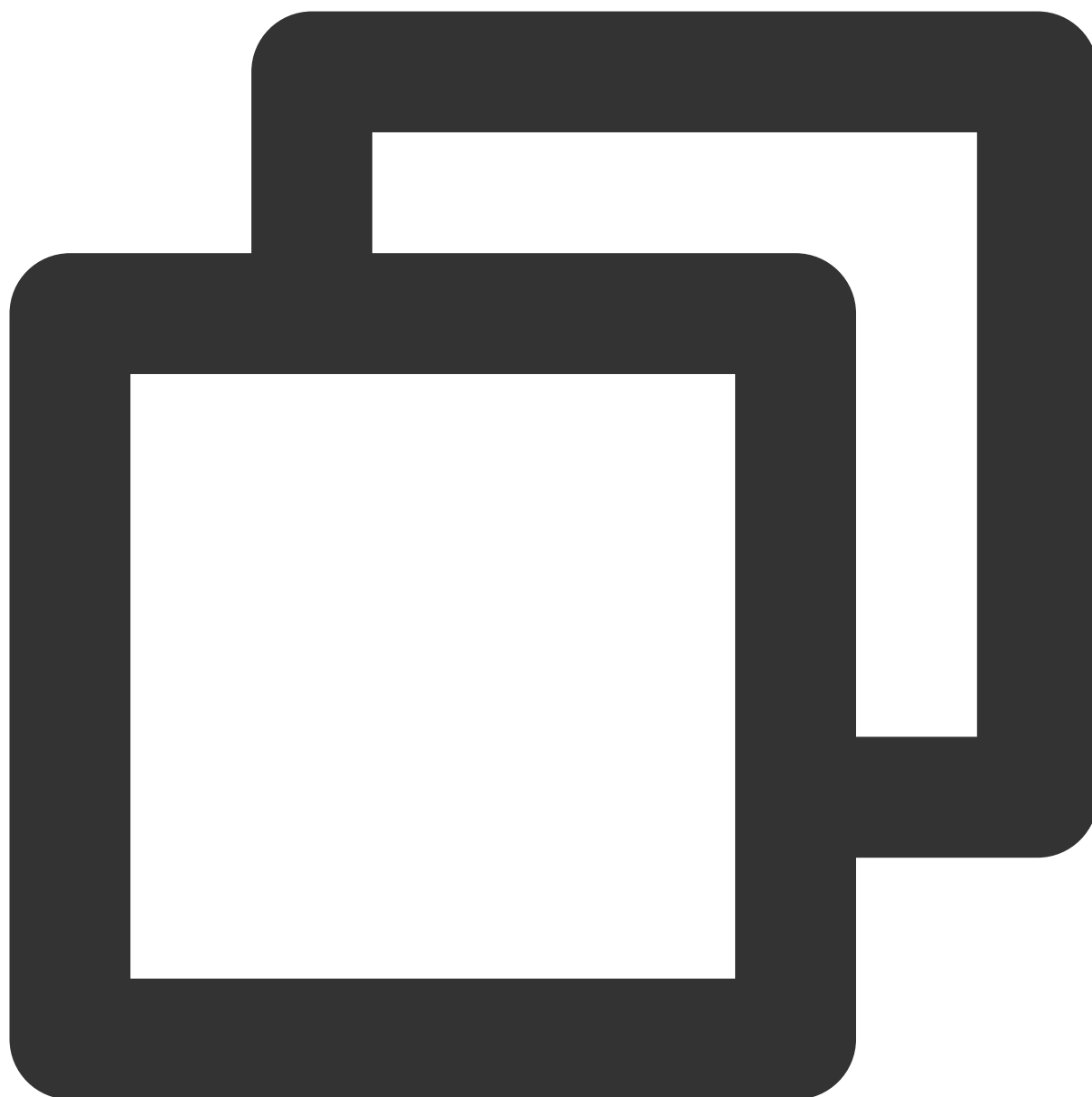


wk-sys-acl-v1-0-5-7cf7f79d6c-d9qcr

0/2

Init:CrashLoo

The queried istio-init logs are as follows:



```
Environment:
-----
ENVOY_PORT=
INBOUND_CAPTURE_PORT=
ISTIO_INBOUND_INTERCEPTION_MODE=
ISTIO_INBOUND_TPROXY_MARK=
ISTIO_INBOUND_TPROXY_ROUTE_TABLE=
ISTIO_INBOUND_PORTS=
ISTIO_LOCAL_EXCLUDE_PORTS=
ISTIO_SERVICE_CIDR=
ISTIO_SERVICE_EXCLUDE_CIDR=
```

Variables:

```
PROXY_PORT=15001
PROXY_INBOUND_CAPTURE_PORT=15006
PROXY_UID=1337
PROXY_GID=1337
INBOUND_INTERCEPTION_MODE=REDIRECT
INBOUND_TPROXY_MARK=1337
INBOUND_TPROXY_ROUTE_TABLE=133
INBOUND_PORTS_INCLUDE=*
INBOUND_PORTS_EXCLUDE=15090,15021,15020
OUTBOUND_IP_RANGES_INCLUDE=*
OUTBOUND_IP_RANGES_EXCLUDE=
OUTBOUND_PORTS_EXCLUDE=
KUBEVIRT_INTERFACES=
ENABLE_INBOUND_IPV6=false
```

Writing following contents to rules file: /tmp/iptables-rules-1618279687646418248.

```
* nat
-N ISTIO_REDIRECT
-N ISTIO_IN_REDIRECT
-N ISTIO_INBOUND
-N ISTIO_OUTPUT
-A ISTIO_REDIRECT -p tcp -j REDIRECT --to-ports 15001
-A ISTIO_IN_REDIRECT -p tcp -j REDIRECT --to-ports 15006
-A PREROUTING -p tcp -j ISTIO_INBOUND
-A ISTIO_INBOUND -p tcp --dport 22 -j RETURN
-A ISTIO_INBOUND -p tcp --dport 15090 -j RETURN
-A ISTIO_INBOUND -p tcp --dport 15021 -j RETURN
-A ISTIO_INBOUND -p tcp --dport 15020 -j RETURN
-A ISTIO_INBOUND -p tcp -j ISTIO_IN_REDIRECT
-A OUTPUT -p tcp -j ISTIO_OUTPUT
-A ISTIO_OUTPUT -o lo -s 127.0.0.6/32 -j RETURN
-A ISTIO_OUTPUT -o lo ! -d 127.0.0.1/32 -m owner --uid-owner 1337 -j ISTIO_IN_REDIRECT
-A ISTIO_OUTPUT -o lo -m owner ! --uid-owner 1337 -j RETURN
-A ISTIO_OUTPUT -m owner --uid-owner 1337 -j RETURN
-A ISTIO_OUTPUT -o lo ! -d 127.0.0.1/32 -m owner --gid-owner 1337 -j ISTIO_IN_REDIRECT
-A ISTIO_OUTPUT -o lo -m owner ! --gid-owner 1337 -j RETURN
-A ISTIO_OUTPUT -m owner --gid-owner 1337 -j RETURN
-A ISTIO_OUTPUT -d 127.0.0.1/32 -j RETURN
-A ISTIO_OUTPUT -j ISTIO_REDIRECT
COMMIT
```

```
iptables-restore --noflush /tmp/iptables-rules-1618279687646418248.txt617375845
```

```
iptables-restore: line 2 failed
```

```
iptables-save
```

```
# Generated by iptables-save v1.6.1 on Tue Apr 13 02:08:07 2021
*nat
:PREROUTING ACCEPT [5214353:312861180]
:INPUT ACCEPT [5214353:312861180]
:OUTPUT ACCEPT [6203044:504329953]
:POSTROUTING ACCEPT [6203087:504332485]
:ISTIO_INBOUND - [0:0]
:ISTIO_IN_REDIRECT - [0:0]
:ISTIO_OUTPUT - [0:0]
:ISTIO_REDIRECT - [0:0]
-A PREROUTING -p tcp -j ISTIO_INBOUND
-A OUTPUT -p tcp -j ISTIO_OUTPUT
-A ISTIO_INBOUND -p tcp -m tcp --dport 22 -j RETURN
-A ISTIO_INBOUND -p tcp -m tcp --dport 15090 -j RETURN
-A ISTIO_INBOUND -p tcp -m tcp --dport 15021 -j RETURN
-A ISTIO_INBOUND -p tcp -m tcp --dport 15020 -j RETURN
-A ISTIO_INBOUND -p tcp -j ISTIO_IN_REDIRECT
-A ISTIO_IN_REDIRECT -p tcp -j REDIRECT --to-ports 15006
-A ISTIO_OUTPUT -s 127.0.0.6/32 -o lo -j RETURN
-A ISTIO_OUTPUT ! -d 127.0.0.1/32 -o lo -m owner --uid-owner 1337 -j ISTIO_IN_REDIRECT
-A ISTIO_OUTPUT -o lo -m owner ! --uid-owner 1337 -j RETURN
-A ISTIO_OUTPUT -m owner --uid-owner 1337 -j RETURN
-A ISTIO_OUTPUT ! -d 127.0.0.1/32 -o lo -m owner --gid-owner 1337 -j ISTIO_IN_REDIRECT
-A ISTIO_OUTPUT -o lo -m owner ! --gid-owner 1337 -j RETURN
-A ISTIO_OUTPUT -m owner --gid-owner 1337 -j RETURN
-A ISTIO_OUTPUT -d 127.0.0.1/32 -j RETURN
-A ISTIO_OUTPUT -j ISTIO_REDIRECT
-A ISTIO_REDIRECT -p tcp -j REDIRECT --to-ports 15001
COMMIT
# Completed on Tue Apr 13 02:08:07 2021
panic: exit status 1
```

```
goroutine 1 [running]:
istio.io/istio/tools/istio-iptables/pkg/dependencies.(*RealDependencies).RunOrFail(
    istio.io/istio/tools/istio-iptables/pkg/dependencies/implementation.go:44 +0x96
istio.io/istio/tools/istio-iptables/pkg/cmd.(*IptablesConfigurator).executeIptables
    istio.io/istio/tools/istio-iptables/pkg/cmd/run.go:493 +0x387
istio.io/istio/tools/istio-iptables/pkg/cmd.(*IptablesConfigurator).executeCommands
    istio.io/istio/tools/istio-iptables/pkg/cmd/run.go:500 +0x45
istio.io/istio/tools/istio-iptables/pkg/cmd.(*IptablesConfigurator).run(0xc0009dfd6
    istio.io/istio/tools/istio-iptables/pkg/cmd/run.go:447 +0x2625
istio.io/istio/tools/istio-iptables/pkg/cmd.glob..func1(0x3b5d680, 0xc0004cce00, 0x
    istio.io/istio/tools/istio-iptables/pkg/cmd/root.go:64 +0x148
github.com/spf13/cobra.(*Command).execute(0x3b5d680, 0xc0004ccd00, 0x10, 0x10, 0x3b
    github.com/spf13/cobra@v1.0.0/command.go:846 +0x29d
github.com/spf13/cobra.(*Command).ExecuteC(0x3b5d920, 0x0, 0x0, 0x0)
    github.com/spf13/cobra@v1.0.0/command.go:950 +0x349
```

```
github.com/spf13/cobra.(*Command).Execute(...)
github.com/spf13/cobra@v1.0.0/command.go:887
main.main()
istio.io/istio/pilot/cmd/pilot-agent/main.go:505 +0x2d
```

Cause and Solution

For details, see [issue](#).

Direct Cause

The cause is that the istio-init container that has exited is cleaned. When K8s detects that the container associated with the pod does not exist, K8s tries to re-pull the deleted container. However, the istio-init container is not reentrant because iptables rules have been created previously. As a result, the istio-init container that is pulled later fails to execute the iptables rules and then crashes.

Root Cause and Solution

The root cause is that a cleanup action is executed by running `docker container rm`, `docker container prune`, or `docker system prune`. Usually, the container is periodically cleaned by the crontab script. To solve this problem, stop cleaning the container.

VirtualService Does Not Take Effect

Last updated : 2023-12-26 15:27:25

In an environment with Istio enabled, `VirtualService` is defined in a cluster, but test results show that the defined rule does not take effect. This problem is usually caused by inappropriate configurations. This section lists common possible reasons.

Intra-Cluster Access: "mesh" Is Not Explicitly Specified in the gateways Field

If the `gateways` field is not specified for `VirtualService`, it actually implies that Istio will add a reserved gateway called "mesh" by default, which indicates all sidecars in the cluster. In other words, this

`VirtualService` rule will take effect for intra-cluster access.

If the `gateways` field is specified, Istio will not add "mesh" by default. For example:



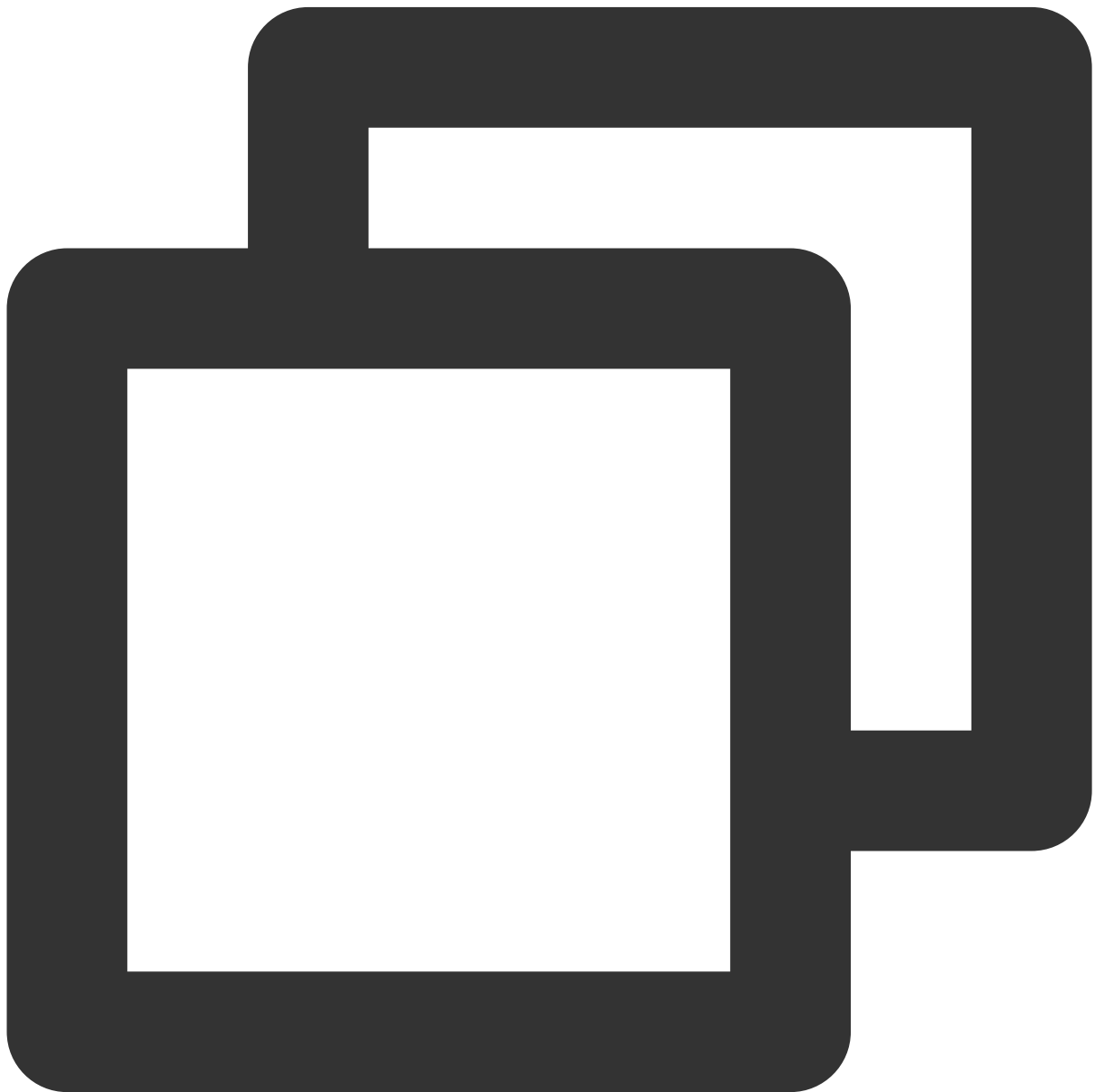
```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: productpage
spec:
  gateways:
  - istio-test/test-gateway
  hosts:
  - bookinfo.example.com
  http:
  - route:
```



```
- destination:
  host: productpage
  port:
    number: 9080
```

The preceding configurations indicate that the `VirtualService` rule will take effect only for the gateway `istio-test/test-gateway`. For intra-cluster access, the traffic will not pass through this gateway, so the rule will not take effect.

If you want the rule to take effect also for the intra-cluster access, explicitly specify "mesh" for `gateways`.



```
gateways:
```

```
- istio-test/test-gateway
- mesh
```

The preceding configurations indicate that this `VirtualService` will take effect for both the gateway `istio-test/test-gateway` and the intra-cluster access.

The following is an explanation of this field in [Istio Documentation](#):

gateways	string[]	The names of gateways and sidecars that should apply these routes. Gateways in other namespaces may be referred to by <gateway namespace>/<gateway name>; specifying a gateway with no namespace qualifier is the same as specifying the VirtualService's namespace. A single VirtualService is used for sidecars inside the mesh as well as for one or more gateways. The selection condition imposed by this field can be overridden using the source field in the match conditions of protocol-specific routes. The reserved word mesh is used to imply all the sidecars in the mesh. When this field is omitted, the default gateway (mesh) will be used, which would apply the rule to all sidecars in the mesh. If a list of gateway names is provided, the rules will apply only to the gateways. To apply the rules to both gateways and sidecars, specify mesh as one of the gateway names.
----------	----------	---

Note that the intra-cluster access usually indicates the direct access to a service name. Therefore, names of services in the cluster to be accessed must be added in the `hosts` field.



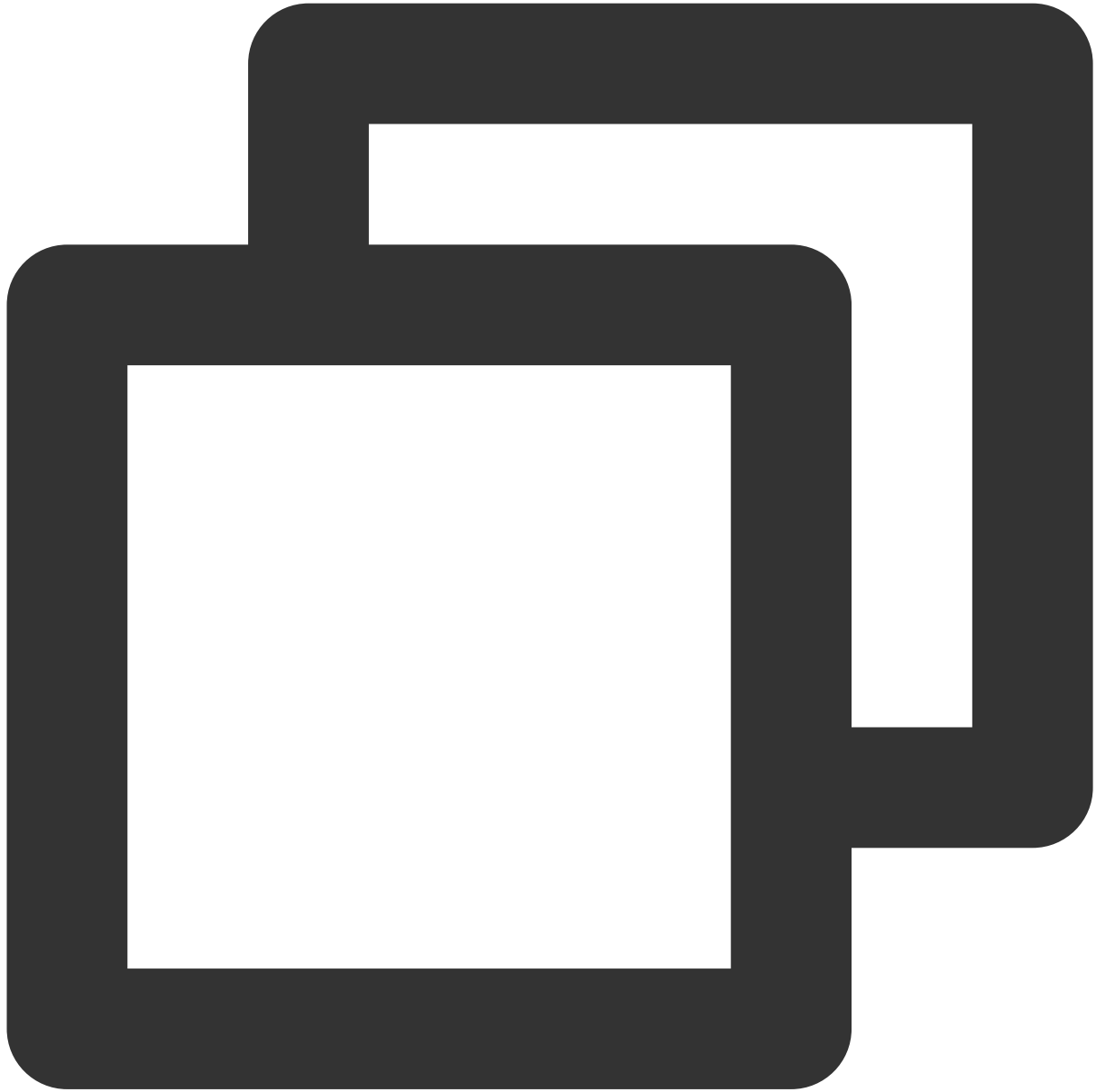
```
hosts:  
- bookinfo.example.com  
- productpage
```

Access Through Ingress Gateway: Incorrect hosts Definition

For the access from an ingress gateway, you need to ensure that the `hosts` fields in `Gateway` and `VirtualService` both contain the actual host used for access or a host name that can be matched by using a

wildcard, usually an external domain name.

As long as the `hosts` field in `Gateway` or `VirtualService` is not defined correctly, `404 Not Found` will be returned. Correct configuration examples are as follows:



```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: test-gateway
  namespace: istio-test
spec:
  selector:
```

```
  app: istio-ingressgateway
  istio: ingressgateway
servers:
- port:
    number: 80
    name: HTTP-80-www
    protocol: HTTP
  hosts:
    - bookinfo.example.com # Define an external access domain name.

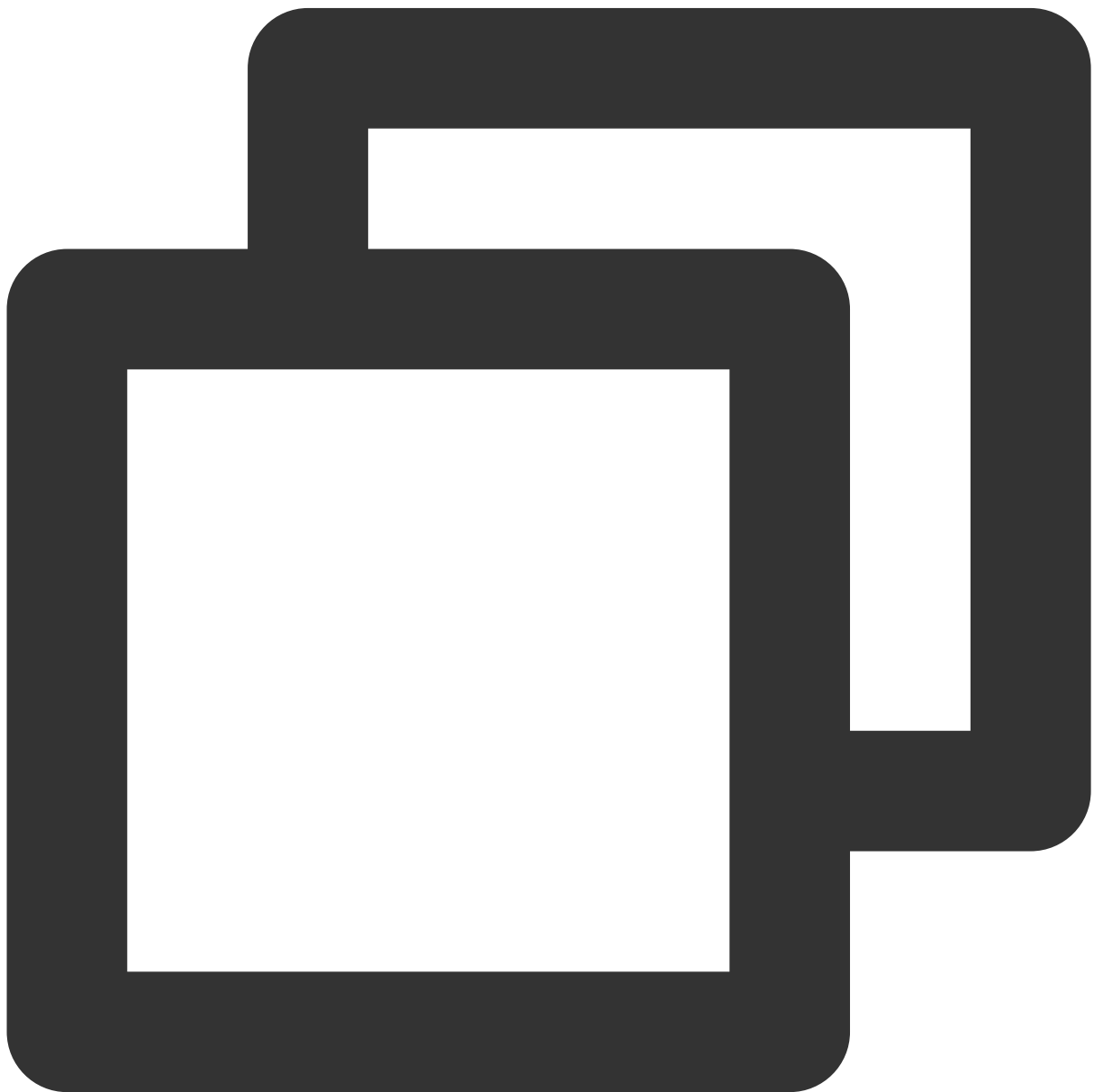
---

apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: productpage
spec:
  gateways:
    - istio-test/test-gateway
  hosts:
    - bookinfo.example.com # Define the external access domain name.
  http:
    - route:
        - destination:
            host: productpage
            port:
              number: 9080
```

Inappropriate VirtualService Route Matching Order

Last updated : 2023-12-26 15:27:43

When writing a VirtualService routing rule, you may set different URIs under the match field to enable requests to be forwarded to different backend services. Sometimes there may be a naming conflict. As a result, only the previous service is always matched. For example:



```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: test
spec:
  gateways:
  - default/example-gw
  hosts:
  - 'test.example.com'
  http:
  - match:
    - uri:
        prefix: /usrv
      rewrite:
        uri: /
      route:
      - destination:
          host: usrv.default.svc.cluster.local
          port:
            number: 80
    - match:
      - uri:
          prefix: /usrv-expand
        rewrite:
          uri: /
        route:
        - destination:
            host: usrv-expand.default.svc.cluster.local
            port:
              number: 80
```

Istio performs matching based on the configuration order, whereas nginx uses longest prefix matching. In this example, prefixes are used for matching. The first prefix is `/usrv`, which indicates that the request will be forwarded to the first service as long as the access URI prefix contains `/usrv`. Because the second matching URI `/usrv-expand` is also a prefix containing `/usrv`, the request will never be forwarded to the service matching the second URI.

Solution

Adjust the matching order. If the prefixes have a containment relationship, the longer prefix should be placed earlier.



```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: test
spec:
  gateways:
  - default/example-gw
  hosts:
  - 'test.example.com'
  http:
  - match:
```



```
- uri:
  prefix: /usrv-expand
rewrite:
  uri: /
route:
- destination:
  host: usrv-expand.default.svc.cluster.local
  port:
    number: 80
- match:
  - uri:
    prefix: /usrv
  rewrite:
    uri: /
  route:
  - destination:
    host: usrv.default.svc.cluster.local
    port:
      number: 80
```

Alternatively, the regular matching can be used.



```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: test
spec:
  gateways:
  - default/gateway
  hosts:
  - 'test.example.com'
  http:
  - match:
```

```
- uri:
  regex: "/usrv(/.*)?"
rewrite:
  uri: /
route:
- destination:
  host: nginx.default.svc.cluster.local
  port:
    number: 80
  subset: v1
- match:
  - uri:
    regex: "/usrv-expand(/.*)?"
  rewrite:
    uri: /
  route:
  - destination:
    host: nginx.default.svc.cluster.local
    port:
      number: 80
    subset: v2
```

Failed to Access an Ingress Gateway from the Public Network

Last updated : 2023-12-26 15:28:00

Accessing an ingress gateway from the public network fails.

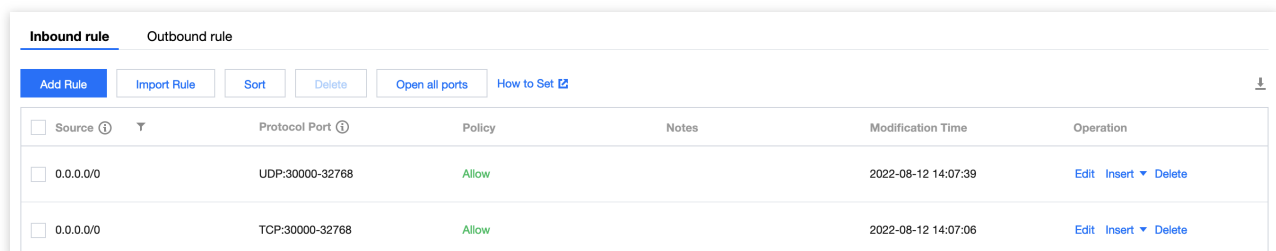
Common Cause

Security Group Does Not Allow NodePort

By default, an ingress gateway installed on Tencent Cloud Mesh is exposed by binding CLB to NodePort. Therefore, the traffic link is: client -> CLB -> NodePort -> ingress gateway.

The key link is CLB -> NodePort. SNAT is not performed on a data packet forwarded by CLB. Therefore, when the packet arrives at a node, the source IP address is the public IP address of the client. If an inbound rule of a security group of the node does not allow the link client -> NodePort, the ingress gateway is inaccessible.

Solution 1: Set a NodePort range (30000-32768) for public access in the inbound rule of the security group of the node.



Inbound rule		Outbound rule				
Add Rule		Import Rule	Sort	Delete	Open all ports	How to Set ?
<input type="checkbox"/>	Source ^①	Protocol Port ^①	Policy	Notes	Modification Time	Operation
<input type="checkbox"/>	0.0.0.0/0	UDP:30000-32768	Allow		2022-08-12 14:07:39	Edit Insert Delete
<input type="checkbox"/>	0.0.0.0/0	TCP:30000-32768	Allow		2022-08-12 14:07:06	Edit Insert Delete

Solution 2: If you are concerned about the security risk of directly allowing all ports in the entire NodePort range, you can expose only the NodePorts used by the ingress gateway service.

Solution 3: If only clients in a fixed IP segment are allowed to access the ingress gateway, all ports in the entire NodePort range can be opened only to this IP segment.

Solution 4: Enable CLB-to-pod direct access for the ingress gateway. In this way, traffic does not pass through the NodePort, and no security group issue occurs. Before enabling CLB-to-pod direct access, ensure that the cluster network supports VPC-CNI. For details, see [How to Enable CLB-to-Pod Direct Access](#).

Parsing Fails After Smart DNS Is Enabled

Last updated : 2023-12-26 15:28:08

After Istio's smart DNS is enabled, DNS resolution fails in some cases. For example:

DNS resolution fails in a container created from an alpine image.

DNS resolution fails in a gRPC service.

Cause

There are some problems in the initial implementation of smart DNS. The format of a responded DNS packet is different from that of ordinary DNS. There is no problem when the underlying library glibc is used for resolution. However, resolution may fail if another DNS client is used.

The alpine image uses musl libc as its underlying library. The resolution behavior of musl libc is different from that of glibc. When the packet format is abnormal, musl libc may fail to resolve the packet. Most applications use the underlying library for resolution, which causes a resolution failure.

For services that use the c/c++ based gRPC framework, the c-ares library is used for DNS resolution by default, and the underlying library is not invoked for resolution. c-ares will fail to resolve the packet in some scenarios when the packet is abnormal.

Bug Fixing

This issue has been fixed in Istio 1.9.2. For details, see the key PR [#31251](#) and one of the [issues](#).

Workaround Solution

If Istio cannot be upgraded to a version later than 1.9.2 temporarily, you can work around the problem by the following methods:

Change the base image from the alpine image to another image (underlying libraries of other base images are basically glibc).

Set the `GRPC_DNS_RESOLVER` environment variable for the c/c++ based gRPC service to `native`. This setting indicates that the underlying library is used for resolution, not the default library c-ares. For description of environment variables, see the [gRPC Documentation](#).

Locality Load Balancing Does Not Take Effect

Last updated : 2023-12-26 15:28:16

It is tested that Istio's locality load balancing does not take effect. This section describes several common reasons.

DestinationRule Is Not Configured with outlierDetection

Locality load balancing is enabled by default. It takes effect only after you configure DestinationRule and specify `outlierDetection`. The configurations enable Istio to perceive whether endpoints are abnormal. When endpoints in the current locality are abnormal, a failover to endpoints in another locality is triggered.

Configuration example:



```
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: nginx
spec:
  host: nginx
  trafficPolicy:
    outlierDetection:
      consecutive5xxErrors: 3
      interval: 30s
      baseEjectionTime: 30s
```

Client Is Not Configured with a Service

The Istio control plane separately delivers EDS to each data plane. Localities of different data plane instances (Envoy) may be different, and the generated EDSs may also be different. Istio will obtain the locality information of the data planes. To be specific, Istio finds the region, zone, and other information stored on the endpoints corresponding to the data planes. If the client does not have any service, no endpoint is available and the control plane cannot obtain the locality information of the client. In this case, it is impossible to implement locality load balancing.

Solution

Configure a service for the client, with the selector field set to the label of the client. If the client does not provide services externally, the ports of the service can be arbitrarily defined.

Headless Service Is Used

For the access to a headless service, locality load balancing is not supported because Istio will directly pass through the headless service request without performing load balancing. Therefore, the client will directly access the pod IP address parsed by DNS.

Solution

Independently create a non-headless service.

Client Status Code 426 "Upgrade Required" Is Returned

Last updated : 2023-12-26 15:28:29

Istio uses Envoy as the data plane to forward HTTP requests, and Envoy requires HTTP/1.1 or HTTP/2 by default.

`426 Upgrade Required` is returned when a client uses HTTP/1.0.

Common nginx Scenario

If nginx is used for `proxy_pass` reverse proxy, it uses HTTP/1.0 by default. You can explicitly set `proxy_http_version` to `1.1`.



```
upstream http_backend {  
    server 127.0.0.1:8080;  
  
    keepalive 16;  
}  
  
server {  
    ...  
  
    location /http/ {  
        proxy_pass http://http_backend;  
    }  
}
```

```
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    ...
}
}
```

Stress Test Scenario

During ab stress test, HTTP/1.0 requests will be sent, and Envoy will always return `426 Upgrade Required` but not forward them. Therefore, the stress test result will not be accurate. You can use other stress test tools, such as [wrk](#).

Enable Istio to Support HTTP/1.0

Some SDKs or frameworks may use the HTTP/1.0 protocol. For example, they use HTTP/1.0 to pull configuration information from the resource center/configuration center. If you want your service to run on Istio without modifying code, you can modify the istiod configuration by adding the `PILOT_HTTP10: 1` environment variable to enable HTTP/1.0.

Sidecars Are Not Automatically Injected

Last updated : 2023-12-26 15:30:07

Sidecar auto-injection is enabled for a namespace through `kubectl label namespace xxx istio-injection=enabled` , but it does not take effect.

Labels in Tencent Cloud Mesh for Automatic Injection

In Tencent Cloud Mesh 1.6 or later, automatic injection for a namespace is enabled by using a label similar to `istio.io/rev=1-8-1` , but not the label `istio-injection=enabled` . The specific label used to enable automatic injection varies depending on versions.

In v1.6, the label `istio.io/rev=1-6-9` is used.

In v1.8, the label `istio.io/rev=1-8-1` is used.

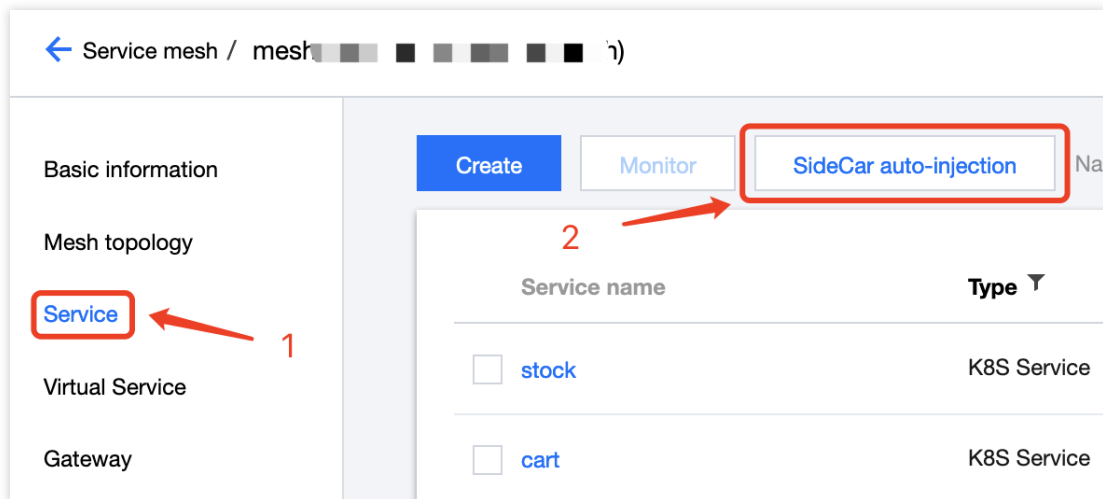
Therefore, the common label `istio-injection=enabled` in the community will not take effect on Tencent Cloud Mesh. The following describes why the common label in the community is not used.

Different Labels Are Required During Mesh Canary Upgrade

To support the canary upgrade of a service mesh, the old and new control planes need to coexist for a period of time, so that the old data plane can be gradually updated and upgraded and then connected to the new control plane. If both control planes use a same label to identify whether automatic injection is required, both control planes will inject sidecars in the rolling update process, which will cause a conflict. To avoid the conflict, Istio officially provides the [canary upgrade scheme](#). This scheme is based on the method of using different labels for different control planes to achieve coexistence of multiple control planes. It is also the scheme adopted by Tencent Cloud Mesh.

How to Enable Automatic Injection on Tencent Cloud Mesh

On the Tencent Cloud Mesh console, choose **Service > Sidecar auto-injection**.



Then, select the namespace for which automatic injection is to be enabled.
You can also use `kubectl` to label the namespace to enable automatic injection.



```
kubectl label namespace xxx istio.io/rev=1-8-1
```

Note :

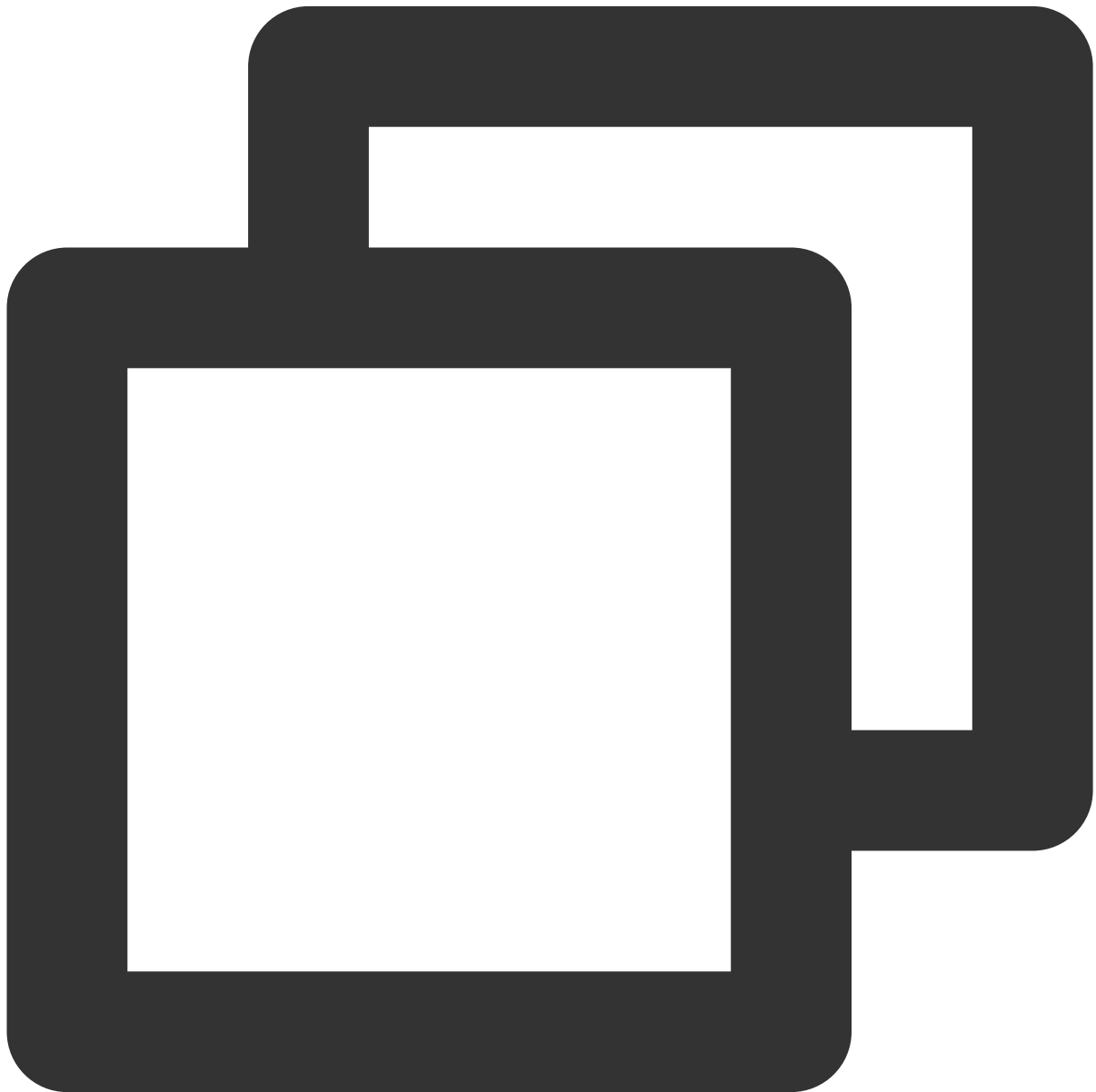
Note that the namespace name must be set based on actual conditions and the label must be modified based on actual Istio version.

Circuit Breaking Does Not Take Effect

Last updated : 2023-12-26 15:30:30

http1MaxPendingRequests Not Defined

A DestinationRule is configured with `maxConnections` .



```
apiVersion: networking.istio.io/v1beta1
```

```
kind: DestinationRule
metadata:
  name: nginx
spec:
  host: nginx
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 1
```

However, the test result shows that when the number of concurrencies exceeds the maximum number of connections defined, circuit breaking is not triggered but the QPS is very low. Usually, this is because the `http1MaxPendingRequests` field is not configured. If the field is not configured, the default value `2^32-1` is used, which is very large. This setting indicates that if the maximum number of connections is exceeded, the request will be queued first (503 will not be returned directly). When the number of connections falls below the maximum value, forwarding will continue.

If you want circuit breaking to be triggered (503 is returned) when the number of connections reaches the upper limit or exceeds the upper limit for a certain amount, you need to explicitly specify `http1MaxPendingRequests`.



```
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: nginx
spec:
  host: nginx
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 1
      http:
```

```
http1MaxPendingRequests: 1
```

404 Is Returned During Access to a StatefulSet's Pod IP Address

Last updated : 2023-12-26 15:30:40

404 is returned when a service container in Istio accesses a pod IP address in the same cluster, but the access in istio-proxy is normal.

Cause

A StatefulSet pod uses headless SVC. In Istio, the support for headless SVC is not the same as that for ordinary SVC. If the pod uses ordinary SVC, the corresponding listener has a passthrough route, that is, the packet will be forwarded to the real destination IP+Port corresponding to the packet. The headless SVC has no such processing. Because the headless SVC does not have a vip, its route is determined, that is, the route points only to the fixed pods in the backend. If the route cannot be matched, a problem occurs. If passthrough routing is also used on the headless SVC, the problem is just masked. Therefore, no passthrough route is created for the headless SVC. For the same service, this problem occurs only when the service is migrated to Istio. It can be considered as a design or implementation issue of Istio.

Sample Scenario

The service uses its own service discovery and therefore directly uses a pod IP address to call the StatefulSet's pod IP address.

Solution

When accessing the StatefulSet's pod IP address in the same cluster, include the host in the request to match the headless SVC's route, thereby avoiding 404 due to a matching failure.

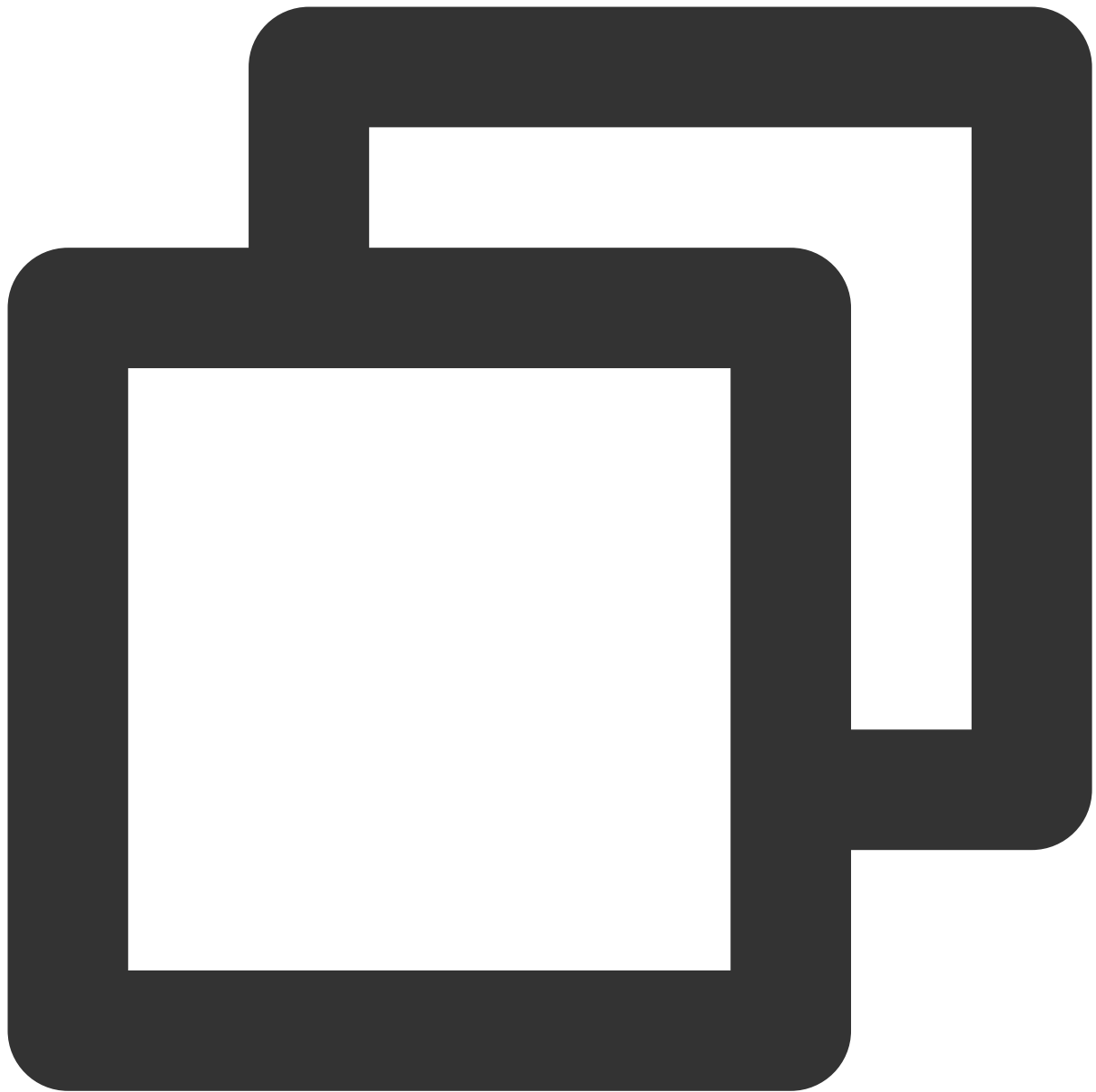
Service Is Abnormal Due to a Retry Policy

Last updated : 2023-12-26 15:30:52

Istio sets a default retry policy for Envoy. Two retries will be performed by default in the case of connect-failure, refused-stream, unavailable, cancelled, or retryable-status-codes. When an error occurs, server logic may have been triggered. An error may occur when the operation is not idempotent (that is, any number of executions will have the same impact as a single execution).

Solution

Configure a VirtualService to disable the retry policy.



```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - retries:
      attempts: 0
```

Incomplete Call Tracing Information

Last updated : 2023-12-26 15:31:01

Call tracing information displayed through the UI is incomplete, for example, information about a previous or next call trace is absent.

Cause

In most cases, it is because the HTTP headers required for tracing are not passed correctly or not passed at the service layer.

Using call tracing in Istio does not mean that services are non-intrusive. There is a basic requirement that a service needs to pass a received tracing-related header to a called service. This step cannot be done by Istio because Istio cannot perceive your service logic and does not know the previous request to which the request for the service to call another service corresponds. Therefore, the service needs to pass the header, and finally the traces can be chained completely.