ALIBABA CLOUD

Alibaba Cloud

Alibaba Cloud Service Mesh

Data Plane

Document Version: 20220209

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Install a sidecar proxy 05
2.Enable automatic sidecar injection by using multiple methods 06
3.Upgrade sidecar proxies 20
4.Write WASM filters for Envoy and deploy them in ASM 22
5.Use ORAS to simplify Wasm-based ASM instance extension 28
6.Enable the DNS proxy feature for an ASM instance 34
7.Upgrade the data plane of an ASM instance without service in... 38

1.Install a sidecar proxy

In Alibaba Cloud Service Mesh (ASM), you can install an Envoy sidecar proxy in the pod of each service in your application to improve the security, reliability, and observability of inter-service communication. This topic describes how to install sidecar proxies.

Context

If you install a sidecar proxy in the pod of an application, an independent container is created in the pod to provide the features of the sidecar proxy. To make full use of these features, each service in your application requires an Envoy sidecar proxy that runs in the pod of that service. The Envoy proxy intercepts all inbound and outbound HTTP traffic to the service and communicates with the Pilot component on the Istio control plane of the corresponding ASM instance.

Step 1: Enable sidecar injection

By default, automatic sidecar injection is disabled for all namespaces. You can manually inject a sidecar proxy by updating the Kubernetes configuration of the pod. Alternatively, you can use the automatic sidecar injection feature of Istio, which is based on webhooks. Run the following command to enable automatic sidecar injection:

```
kubectl label namespace {namespace} istio-injection=enabled --overwrite
```

Note In the preceding command, the *namespace* parameter specifies the namespace of the application. If you do not specify this parameter, the default namespace is used.

Step 2: Restart the pod

Sidecar proxies are injected when pods are created. Therefore, you must restart the pods to make sidecar injection take effect.

Notice We recommend that you restart the pods in the test environment multiple times to ensure that your service will not be affected by any traffic interruptions.

1. Run the following command to restart a pod:

   ```
   kubectl get pod {podname} -n {namespace} -o yaml | kubectl replace --force -f -
   ```

2. Check whether a sidecar proxy is injected to every pod in the namespace. After a sidecar proxy is injected to a pod, each workload is supported by a main container and a sidecar proxy container.

   ```
   kubectl get pod -n {namespace} --all
   ```

2.Enable automatic sidecar injection by using multiple methods

Alibaba Cloud Service Mesh (ASM) allows you to add a Container Service for Kubernetes (ACK) cluster to an ASM instance. To make full use of ASM, you must inject a sidecar proxy into the pod of an application that is deployed in the ACK cluster. ASM supports both manual and automatic sidecar injection. We recommend that you enable automatic sidecar injection because it requires simpler operations than manual sidecar injection. This topic describes the methods that can be used to enable automatic sidecar injection.

Context

By default, ASM provides a webhook controller to automatically inject sidecar proxies into the pods of applications. For more information about sidecar proxies, see Installing the Sidecar.

> **Note** Make sure that the Istio version of the ASM instance for which you want to enable automatic sidecar injection is 1.6.8.17 or later.

Configure automatic sidecar injection

1. Log on to the ASM console.
2. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
3. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
4. On the details page of the ASM instance, choose Sidecar Management(Data Plane) > Sidecar Proxy injection in the left-side navigation pane. Click Injection strategy configuration management, configure automatic injection settings, and then click Update Settings.

   The following table describes the operations that you can perform to configure automatic sidecar injection.

Operation	Description

Operation	Description
Select only **Enable Automatic Sidecar Injection for All Namespaces**.	After you select this option, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection In a namespace that is not labeled with `istio-injection:disabled`, add the `sidecar.istio.io/inject="true"` annotation to a pod. This way, automatic sidecar injection is enabled for the pod. ○ Disable automatic sidecar injection ■ Label a namespace with `istio-injection:disabled`. This way, automatic sidecar injection is disabled for the pods in the namespace. ■ Remove the `sidecar.istio.io/inject="true"` annotation from a pod. This way, automatic sidecar injection is disabled for the pod.
Select **Enable Automatic Sidecar Injection for All Namespaces** and **Other Configurations of Automatic Sidecar Injection**.	After you select this option, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection Set the `alwaysInjectSelector` parameter in the code editor that appears after you select Other Configurations of Automatic Sidecar Injection. In a namespace that is not labeled with `istio-injection:disabled`, add the `key` label in the `alwaysInjectSelector` parameter to a pod. This way, automatic sidecar injection is enabled for the pod. ○ Disable automatic sidecar injection ■ Label a namespace with `istio-injection:disabled`. This way, automatic sidecar injection is disabled for the pods in the namespace. ■ Remove the `sidecar.istio.io/inject="true"` annotation from a pod. This way, automatic sidecar injection is disabled for the pod.

Operation	Description
Select only **Use the Pod Annotation to Enable Automatic Sidecar Injection**.	After you select this option, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection Label a namespace with `istio-injection:enabled`. This way, automatic sidecar injection is enabled for the pods in the namespace. ○ Disable automatic sidecar injection ■ Remove the `istio-injection:enabled` label from a namespace. This way, automatic sidecar injection is disabled for the pods in the namespace. ■ Add the `sidecar.istio.io/inject="false"` annotation to a pod. This way, automatic sidecar injection is disabled for the pod.
Select **Use the Pod Annotation to Enable Automatic Sidecar Injection** and **Other Configurations of Automatic Sidecar Injection**.	After you select this option, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection Label a namespace with `istio-injection:enabled`. This way, automatic sidecar injection is enabled for the pods in the namespace. ○ Disable automatic sidecar injection Remove the `istio-injection:enabled` label from a namespace. This way, automatic sidecar injection is disabled for the pods in the namespace. ○ Disable automatic sidecar injection for a pod in a namespace that is labeled with `istio-injection:enabled` Set the `neverInjectSelector` parameter in the code editor that appears after you select Other Configurations of Automatic Sidecar Injection. Add the `key` label in the `neverInjectSelector` parameter to a pod in a namespace that is labeled with istio-injection:enabled. This way, automatic sidecar injection is disabled for the pod.

Operation	Description
Select **Enable Automatic Sidecar Injection for All Namespaces** and **Use the Pod Annotation to Enable Automatic Sidecar Injection**.	After you select this option, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection Remove the `istio-injection:disabled` label from a namespace. This way, automatic sidecar injection is enabled for the pods in the namespace. ○ Disable automatic sidecar injection Label a namespace with `istio-injection:disabled`. This way, automatic sidecar injection is disabled for the pods in the namespace.
Select **Enable Automatic Sidecar Injection for All Namespaces**, **Use the Pod Annotation to Enable Automatic Sidecar Injection**, and **Other Configurations of Automatic Sidecar Injection**.	After you select this option, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection Remove the `istio-injection:disabled` label from a namespace. This way, automatic sidecar injection is enabled for the pods in the namespace. ○ Disable automatic sidecar injection Label a namespace with `istio-injection:disabled`. This way, automatic sidecar injection is disabled for the pods in the namespace. ○ Disable automatic sidecar injection for a pod in a namespace that is not labeled with `istio-injection:disabled` Set the `neverInjectSelector` parameter in the code editor that appears after you select Other Configurations of Automatic Sidecar Injection. Add the `key` label in the `neverInjectSelector` parameter to a pod in a namespace that is not labeled with istio-injection:disabled. This way, automatic sidecar injection is disabled for the pod.

Operation	Description
Select only **Other Configurations of Automatic Sidecar Injection**.	After you select this option, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection Label a namespace with `istio-injection:enabled`, set the `alwaysInjectSelector` parameter in the code editor that appears after you select Other Configurations of Automatic Sidecar Injection, and then add the `key` label in the `alwaysInjectSelector` parameter to a pod in the namespace. This way, automatic sidecar injection is enabled for the pod. ○ Disable automatic sidecar injection ■ Remove the `istio-injection:enabled` label from a namespace. This way, automatic sidecar injection is disabled for the pods in the namespace. ■ Remove the `sidecar.istio.io/inject="true"` annotation from a pod in a namespace that is labeled with `istio-injection:enabled`. This way, automatic sidecar injection is disabled for the pod.
Select no option.	In this case, you can enable or disable automatic sidecar injection based on your business requirements. ○ Configure automatic sidecar injection Label a namespace with `istio-injection:enabled` and add the `sidecar.istio.io/inject="true"` annotation to a pod in the namespace. This way, automatic sidecar injection is enabled for the pod. ○ Disable automatic sidecar injection ■ Remove the `istio-injection:enabled` label from a namespace. This way, automatic sidecar injection is disabled for the pods in the namespace. ■ Remove the `sidecar.istio.io/inject="true"` annotation from a pod in a namespace that is labeled with `istio-injection:enabled`. This way, automatic sidecar injection is disabled for the pod.

In addition to configuring automatic sidecar injection, you can configure proxy resources.

Parameter	Description
Resource Settings for Sidecar Injector	By default, ASM provides a webhook controller for each cluster on the data plane to automatically inject sidecar proxies into the pods of applications. The specified resource settings are used to limit the size of the webhook controller.
Resource Settings for Injected Proxies	A sidecar proxy provides the proxy service for an application. After a sidecar proxy is automatically injected into the pod of an application, the sidecar proxy runs in the same pod as the container of the application. The specified resource settings are used to limit the size of the sidecar proxy.

Other automatic sidecar injection settings

You can configure labels in other automatic sidecar injection settings to control whether to inject a sidecar proxy into a pod based on label matching.

- Set the `alwaysInjectSelector` parameter to inject sidecar proxies into the pods that are matched by label. This setting takes priority over global settings.

```
{
  "alwaysInjectSelector": [
    {
      "matchExpressions": [
        {
          "key": "key1",
          "operator": "Exists"
        }
      ]
    },
    {
      "matchExpressions": [
        {
          "key": "key2",
          "operator": "Exists"
        }
      ]
    }
  ]
}
```

- Set the `neverInjectSelector` parameter to disable sidecar proxies from being injected into the pods that are matched by label. This setting takes priority over global settings.

```
{
  "neverInjectSelector": [
    {
      "matchExpressions": [
        {
          "key": "key3",
          "operator": "Exists"
        }
      ]
    },
    {
      "matchExpressions": [
        {
          "key": "key4",
          "operator": "Exists"
        }
      ]
    }
  ]
}
```

- Set other parameters.

```
{
  "replicaCount": 2,
  "injectedAnnotations": {
    "test/istio-init": "runtime/default",
    "test/istio-proxy": "runtime/default"
  },
  "nodeSelector": {
     "beta.kubernetes.io/os": "linux"
  }
}
```

  - replicaCount: the number of replicas that are deployed for a sidecar injector.
  - injectedAnnotations: other injected annotations.
  - nodeSelector: the nodes on which sidecar injectors run. In this example, the beta.kubernetes.io/os parameter is set to `linux`, which indicates that sidecar injectors run on the nodes that are labeled with `linux`.

Scenario 1: Disable automatic sidecar injection for specific pods in a namespace for which automatic sidecar injection is enabled

To disable automatic sidecar injection for specific pods in a namespace for which automatic sidecar injection is enabled, perform the following operations:

Use other automatic sidecar injection configurations to disable automatic sidecar injection for specific pods in a namespace for which automatic sidecar injection is enabled

1. Enable automatic injection for an ASM instance.
   i. Log on to the ASM console.
   ii. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.

iii. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.

iv. On the details page of the ASM instance, choose Sidecar Management(Data Plane) > Sidecar Proxy injection in the left-side navigation pane.

v. Click Injection strategy configuration management and select **Use the Pod Annotation to Enable Automatic Sidecar Injection** and **Other Configurations of Automatic Sidecar Injection** in the Enable Automatic Sidecar Injection section. In the code editor that appears, add the following content and click **Update Settings**.

```
{
  "neverInjectSelector": [
    {
      "matchExpressions": [
        {
          "key": "notinjectapp",
          "operator": "Exists"
        }
      ]
    }
  ]
}
```

2. Create a namespace.

   i. On the details page of the ASM instance, click **Namespace** in the left-side navigation pane. On the Namespace page, click **Create**.

   ii. In the **Create Namespace** panel, specify a name for the namespace, click **Add** next to Labels, add a label with the name of istio-injection and the value of enabled, and then click **OK**. In this example, the namespace is named test1.

3. Creates an application.

   i. Create an application in the test1 namespace of the ACK cluster that is added to the ASM instance. For more information, see Deploy an application in an ASM instance. In this example, the details application is deployed.

ii. Check whether automatic sidecar injection is enabled for the pod of the details application.

a. Log on to the ACK console.

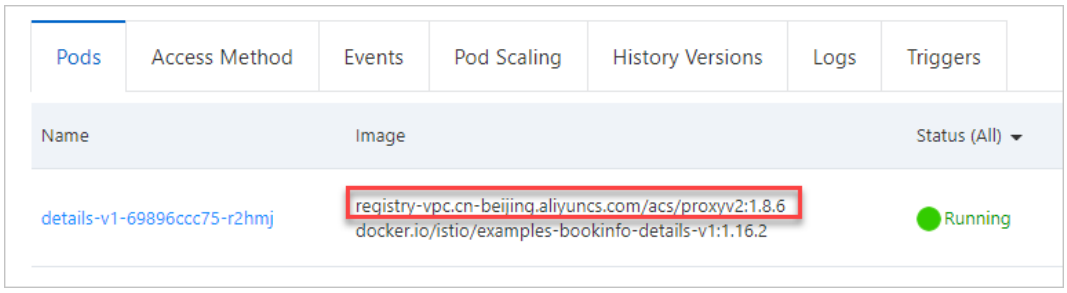
b. In the left-side navigation pane of the ACK console, click **Clusters**.

c. On the **Clusters** page, find the cluster that you want to manage. Then, click the name of the cluster or click **Details** in the **Actions** column.

d. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

e. At the top of the **Deployments** page, select test1 from the **Namespace** drop-down list. Then, click the name of the details application.

The **Pods** tab shows a proxy image. This indicates that automatic sidecar injection is enabled for the pod of the details application.



4. Add a label to the pod to disable automatic sidecar injection.

i. Log on to the ACK console.

ii. In the left-side navigation pane of the ACK console, click **Clusters**.

iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

iv. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

v. At the top of the **Deployments** page, select test1 from the **Namespace** drop-down list. Then, find the details application and choose **More > View in YAML** in the **Actions** column.

vi. In the `labels` parameter, add a label with the `key` of `notinjectapp` and a custom `value`. Then, click **Update**.

```
135    template:
136      metadata:
137        annotations:
138          redeploy-timestamp: '1629688001960'
139        labels:
140          app: details
141          notinjectapp: details
142          version: v1
143          notinjectapp: details
```

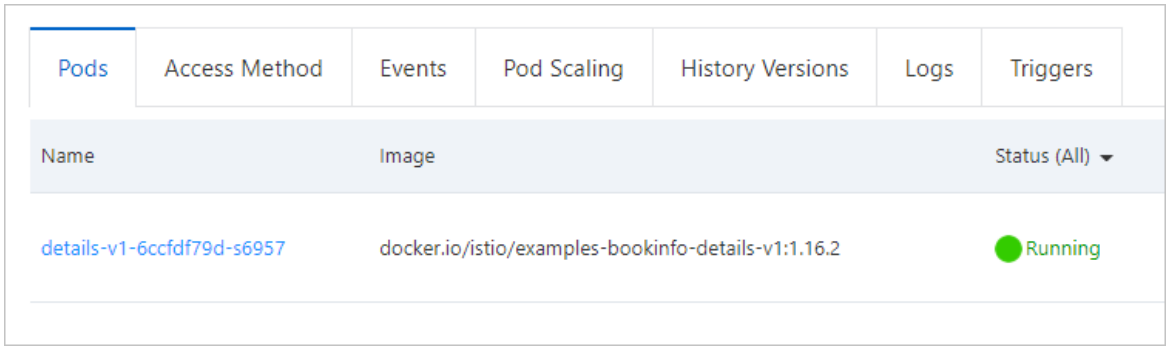
vii. At the top of the **Deployments** page, select test1 from the **Namespace** drop-down list. Then, find the details application and choose **More > Redeploy** in the **Actions** column.

viii. In the dialog box that appears, click **OK**.

5. Check whether automatic sidecar injection is disabled for the pod of the details application even if automatic sidecar injection is enabled for the test1 namespace.

On the **Deployments** page, click the name of the details application. The **Pods** tab shows no

proxy image. This indicates that automatic sidecar injection is disabled for the pod of the details application even if automatic sidecar injection is enabled for the test1 namespace.



Use annotations to disable automatic sidecar injection for specific pods in a namespace for which automatic sidecar injection is enabled

1. Enable automatic injection for an ASM instance.
   i. Log on to the ASM console.
   ii. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
   iii. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
   iv. On the details page of the ASM instance, choose Sidecar Management (Data Plane) > Sidecar Proxy injection in the left-side navigation pane.
   v. Click Injection strategy configuration management, select **Use the Pod Annotation to Enable Automatic Sidecar Injection** in the **Enable Automatic Sidecar Injection** section, and then click **Update Settings**.
2. Create a namespace.
   i. On the details page of the ASM instance, click **Namespace** in the left-side navigation pane. On the Namespace page, click **Create**.
   ii. In the **Create Namespace** panel, specify a name for the namespace, click **Add** next to Labels, add a label with the name of istio-injection and the value of enabled, and then click **OK**. In this example, the namespace is named test1.
3. Creates an application.
   i. Create an application in the test1 namespace of the ACK cluster that is added to the ASM instance. For more information, see Deploy an application in an ASM instance. In this example, the details application is deployed.

ii. Check whether automatic sidecar injection is enabled for the pod of the details application.

a. Log on to the ACK console.

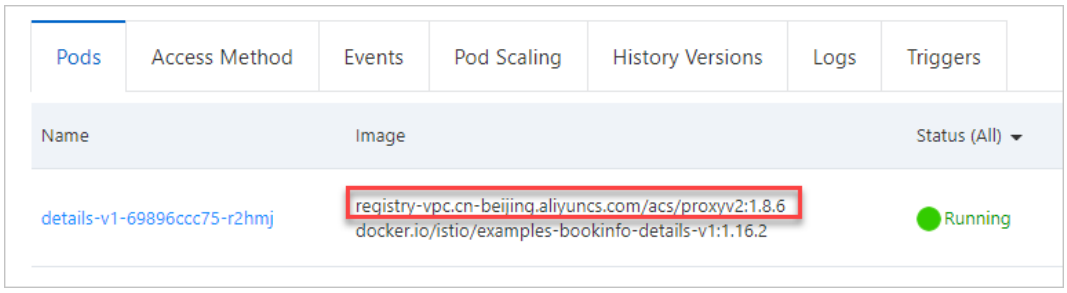
b. In the left-side navigation pane of the ACK console, click **Clusters**.

c. On the **Clusters** page, find the cluster that you want to manage. Then, click the name of the cluster or click **Details** in the **Actions** column.

d. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

e. At the top of the **Deployments** page, select test1 from the **Namespace** drop-down list. Then, click the name of the details application.

The **Pods** tab shows a proxy image. This indicates that automatic sidecar injection is enabled for the pod of the details application.



4. Add an annotation to the pod to disable automatic sidecar injection.

i. Log on to the ACK console.

ii. In the left-side navigation pane of the ACK console, click **Clusters**.

iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

iv. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

v. At the top of the **Deployments** page, select test1 from the **Namespace** drop-down list. Then, find the details application and choose **More > View in YAML** in the **Actions** column.

vi. Add `sidecar.istio.io/inject: "false"` to `annotations`, and then click **Update**.

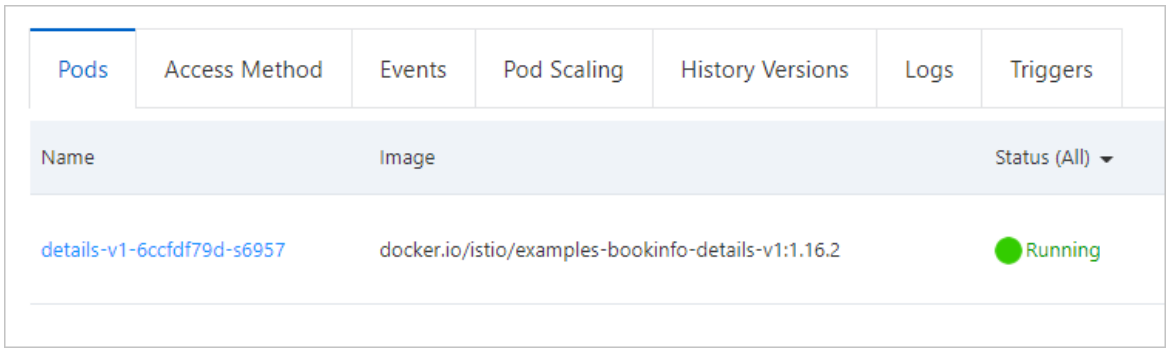
```
121 spec:
122   progressDeadlineSeconds: 600
123   replicas: 1
124   revisionHistoryLimit: 10
125   selector:
126     matchLabels:
127       app: details
128       version: v1
129   strategy:
130     rollingUpdate:
131       maxSurge: 25%
132       maxUnavailable: 25%
133     type: RollingUpdate
134   template:
135     metadata:
136       annotations:
137         redeploy-timestamp: '1629711538748'
138         sidecar.istio.io/inject: "false"
```

vii. At the top of the **Deployments** page, select test from the **Namespace** drop-down list. Then, find the details application and choose **More > Redeploy** in the **Actions** column.

viii. In the dialog box that appears, click **OK**.

5. Check whether automatic sidecar injection is disabled for the pod of the details application even if automatic sidecar injection is enabled for the test1 namespace.

   On the **Deployments** page, click the name of the details application. The **Pods** tab displays no proxy image. This indicates that automatic sidecar injection is disabled for the pod of the details application even if automatic sidecar injection is enabled for the test1 namespace.



Scenario 2: Configure automatic sidecar injection for a pod

If you do not want to configure automatic sidecar injection by namespace, you can configure automatic sidecar injection by pod.

1. Enable automatic sidecar injection for a namespace.

   i. Log on to the ASM console.

   ii. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.

   iii. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.

iv. On the details page of the ASM instance, click **Namespace** in the left-side navigation pane.

v. Find the namespace for which you want to enable automatic sidecar injection and click **Enable Automatic Sidecar Injection** in the **Automatic Sidecar Injection** column. In the Submit message, click **OK**. In this example, the test2 namespace is used.

2. Create an application in the test2 namespace of the ACK cluster that is added to the ASM instance. For more information, see Deploy an application in an ASM instance. In this example, the reviews application is deployed.

3. Add an annotation to the pod of the reviews application to enable automatic sidecar injection for the pod.

   i. Log on to the ACK console.

   ii. In the left-side navigation pane of the ACK console, click **Clusters**.

   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

   iv. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

   v. At the top of the **Deployments** page, select test2 from the **Namespace** drop-down list. Then, find the reviews application and choose **More > View in YAML** in the **Actions** column.

   vi. Add `sidecar.istio.io/inject: "true"` to `annotations`, and then click **Update**.

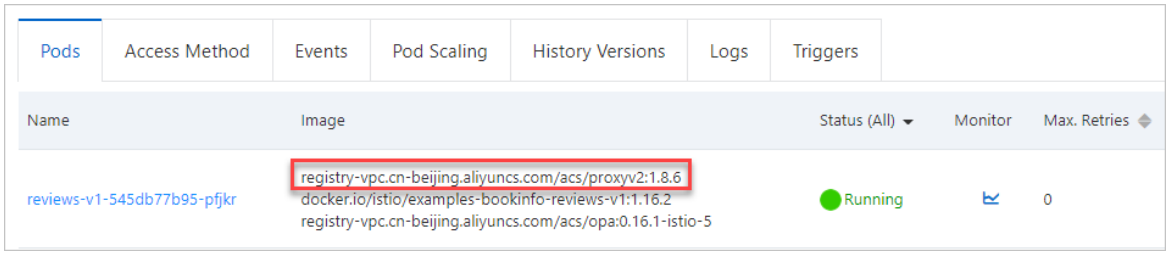
```
146 spec:
147   progressDeadlineSeconds: 600
148   replicas: 1
149   revisionHistoryLimit: 10
150   selector:
151     matchLabels:
152       app: reviews
153       version: v1
154   strategy:
155     rollingUpdate:
156       maxSurge: 25%
157       maxUnavailable: 25%
158     type: RollingUpdate
159   template:
160     metadata:
161       annotations:
162         redeploy-timestamp: '1629707593246'
163         sidecar.istio.io/inject: "true"
```

   vii. At the top of the **Deployments** page, select test2 from the **Namespace** drop-down list. Then, find the reviews application and choose **More > Redeploy** in the **Actions** column.

   viii. In the dialog box that appears, click **OK**.

4. Check whether automatic sidecar injection is enabled for the pod of the reviews application.

   On the **Deployments** page, click the name of the reviews application. The **Pods** tab shows a proxy image. This indicates that automatic sidecar injection is enabled for the pod of the reviews application.


Pods
Access Method
Events
Pod Scaling
History Versions
Logs
Triggers
Name
Image
Status (All)
Monitor
Max. Retries
reviews-v1-545db77b95-pfjkr
registry-vpc.cn-beijing.aliyuncs.com/acs/proxyv2:1.8.6
docker.io/istio/examples-bookinfo-reviews-v1:1.16.2
registry-vpc.cn-beijing.aliyuncs.com/acs/opa:0.16.1-istio-5
Running
0

3.Upgrade sidecar proxies

After the control plane of an Alibaba Cloud Service Mesh (ASM) instance is upgraded, you also need to upgrade the sidecar proxies for Istio-enabled applications on the ASM instance. This topic describes how to upgrade sidecar proxies by automatic sidecar injection and manual sidecar injection.

Prerequisites

The kubectl client is connected to the Container Service for Kubernetes cluster. For more information, see Connect to ACK clusters by using kubectl.

Context

Sidecar proxies are deployed on the data plane. When you upgrade sidecar proxies, you need to upgrade the kubeconfig file of the data plane instead of the ASM instance. Therefore, you need to obtain the kubeconfig file from the Container Service console instead of the ASM console.

Automatic sidecar injection

If automatic sidecar injection is enabled, you can upgrade sidecar proxies in all pods by performing a rolling upgrade for these pods. In this way, sidecar proxies of the new version are injected to the pods. We recommend that you this method because it only requires simple upgrade operations.

You can use the following shell script to trigger a rolling upgrade by patching the grace termination period.

```
NAMESPACE=$1
DEPLOYMENT_LIST=$(kubectl -n $NAMESPACE get deployment -o jsonpath='{.items[*].metadata.nam
e}')
echo "Refreshing pods in all Deployments: $DEPLOYMENT_LIST"
for deployment_name in $DEPLOYMENT_LIST ; do
    #echo "get TERMINATION_GRACE_PERIOD_SECONDS from deployment: $deployment_name"
    TERMINATION_GRACE_PERIOD_SECONDS=$(kubectl -n $NAMESPACE get deployment "$deployment_na
me" -o jsonpath='{.spec.template.spec.terminationGracePeriodSeconds}')
    if [ "$TERMINATION_GRACE_PERIOD_SECONDS" -eq 30 ]; then
        TERMINATION_GRACE_PERIOD_SECONDS='31'
    else
        TERMINATION_GRACE_PERIOD_SECONDS='30'
    fi
    patch_string="{\"spec\":{\"template\":{\"spec\":{\"terminationGracePeriodSeconds\":$TER
MINATION_GRACE_PERIOD_SECONDS}}}}"
    #echo $patch_string
    kubectl -n $NAMESPACE patch deployment $deployment_name -p $patch_string
done
echo "done."
```

Save the preceding shell script in a file named upgradeproxy.sh and grant the executable permission to the file. For example, you can run the chmod +x upgradeproxy.sh command on the Linux command line to grant the executable permission.

You must specify the namespace in the command. For example, if you want to upgrade the pods in the default namespace, you need to run the ./upgradeproxy.sh default command.

```
chmod +x upgradeproxy.sh
./upgradeproxy.sh  default
```

Manual sidecar injection

If automatic sidecar injection is disabled, you need to run the following command to upgrade sidecar proxies.

Create a deployment YAML file and run the kubectl apply command.

```
kubectl apply -f <(istioctl kube-inject -f <A raw application YAML file with no sidecar pro
xy configuration injected>)
```

4.Write WASM filters for Envoy and deploy them in ASM

Alibaba Cloud Service Mesh (ASM) supports the programming language WebAssembly (WASM). You can deploy WASM filters in Envoy proxies that are used to manage clusters on the data plane. Filters help you extend Envoy proxies with new features so that you can use Envoy proxies to implement more features in ASM. This topic introduces WASM filters and describes how to write WASM filters for Envoy proxies and deploy them in ASM.

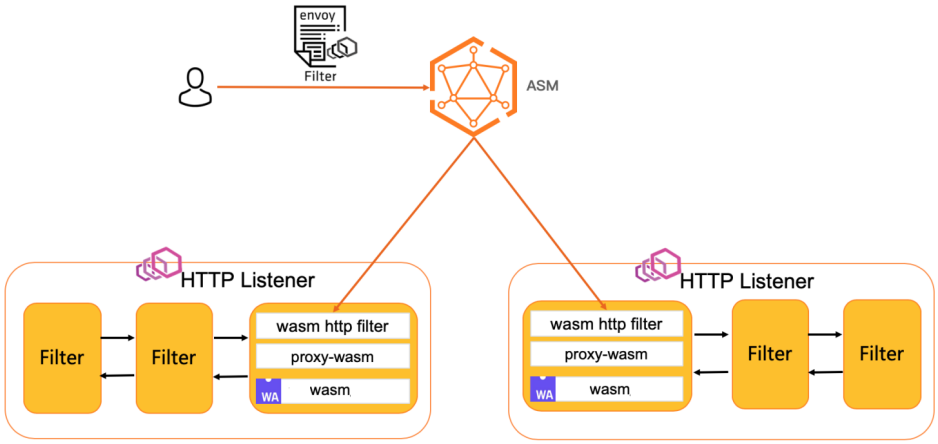
Context

An Envoy proxy is a high-performance programmable Layer 3, Layer 4, and Layer 7 proxy. ASM places Envoy proxies on the data plane. Envoy proxies use network filters to manage collections and handle traffic. Network filters can be mixed into filter chains to implement access control, data and protocol conversion, data enhancement, and auditing. You can add filters to Envoy proxies to expand the feature set of Envoy. Use one of the following methods to add filters:

- Static precompilation: Integrate additional filters into the source code of Envoy proxies and compile a new version for Envoy proxies. The drawback of this method is that you must manually maintain the version of Envoy proxies to keep them in sync with the official version. In addition, Envoy proxies are implemented in C++. Therefore, the filters must be implemented by C++, too.
- Dynamic loading at runtime: Dynamically load new filters into Envoy proxies at runtime.

The second method greatly simplifies the process of extending Envoy proxies with new features. This method relies on WASM, a portable and efficient binary instruction format that provides an embeddable and isolated execution environment.

The following figure shows how ASM works with WASM filters.



Advantages of WASM filters

WASM filters provide the following advantages:

- Agility: Filters can be dynamically loaded into the running Envoy proxies. You do not need to stop or recompile the Envoy proxies.
- Maintainability: You do not need to change the code libraries of Envoy proxies to extend the functionality.
- Diversity: You can use a popular programming language, such as C, C++, or Rust, to compile WASM

filters based on your requirements.

- Reliability and isolation: Filters are deployed into a sandbox virtual machine and are isolated from the Envoy process. If a WASM filter fails, it does not impact the Envoy process.
- Security: Filters communicate with Envoy proxies by using the predefined API. Therefore, filters can access and modify only a limited number of connection or request properties.

Before you use WASM filters, take the following drawbacks into consideration:

- Filters complied by WASM are about only 70% as fast as filters compiled by C++ in a static manner.
- More memories are consumed because WASM filters rely on one or more WASM virtual machines.

Use the Envoy Proxy WASM SDK to build a filter

Envoy proxies run WASM filters in stack-based virtual machines. Memories of filters are isolated from the environment of Envoy proxies. All interactions between Envoy proxies and WASM filters are implemented by the Envoy Proxy WASM SDK. The Envoy Proxy WASM SDK supports many programming languages, including C++, Rust, AssemblyScript, and Go. Note that Go-based implementation is still in experiment. GitHub community is promoting the Application Binary Interface (ABI) specification and conventions to use between proxies and WebAssembly filters. For more information, see WebAssembly for Proxies (ABI specification).

1. The easiest way to build a WASM filter is using Docker. You can use the Envoy Proxy WASM SDK for C++ to create a docker image. For more information, see Docker.
2. Create a project. For more information, see Creating a project for use with the Docker build image.
3. Compile the project in the docker image. For more information, see Compiling with the Docker build image.
4. Go to the root directory of the project and run the following command to build a WASM filter:

```
docker run -v $PWD:/work -w /work  registry.cn-hangzhou.aliyuncs.com/acs/wasmsdk:v0.1 /
build_wasm.sh
```

Deploy the WASM filter in ASM

1. Create a config map to hold the binary file of the WASM filter. For example, create a config map named wasm-example-filter in the default namespace and store the binary file example-filter.wasm of the WASM filter to the config map.

```
kubectl create configmap -n default wasm-example-filter --from-file=example-filter.wasm
```

2. Use the following two annotations to inject the binary file of the WASM filter to the Kubernetes containers of the target application:

```
sidecar.istio.io/userVolume: '[{"name":"wasmfilters-dir","configMap": {"name": "wasm-ex
ample-filter"}}]'
sidecar.istio.io/userVolumeMount: '[{"mountPath":"/var/local/lib/wasm-filters","name":"
wasmfilters-dir"}]'
```

3. Update version 1 of the productpage service.

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

4. Update version 1 of the details service.

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

5. Check whether the binary file of the WASM filter is available under the */var/local/lib/wasm-filters* path in the istio-proxy containers of the application.

```
kubectl exec -it deployment/productpage-v1 -c istio-proxy -- ls /var/local/lib/wasm-fil
ters/
kubectl exec -it deployment/details-v1 -c istio-proxy -- ls /var/local/lib/wasm-filters
/
```

6. Enable the WASM filter to keep logs at the DEBUG level when it processes traffic that targets the `productpage` service.

```
kubectl port-forward deployment/productpage-v1 15000
curl -XPOST "localhost:15000/logging? wasm=debug"
```

7. Enable the WASM filter to keep logs at the DEBUG level when it processes traffic that targets the `details` service.

```
kubectl port-forward deployment/details-v1 15000
curl -XPOST "localhost:15000/logging? wasm=debug"
```

8. Insert the WASM filter into the HTTP-level filter chain of the `productpage` service.

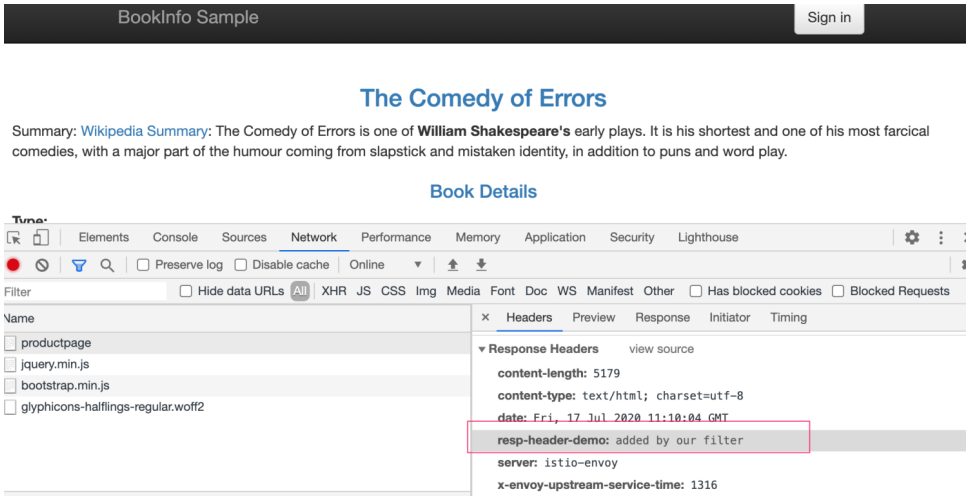
```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: productpage-v1-examplefilter
  labels:
    asm-system: 'true'
    provider: asm
spec:
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      context: SIDECAR_INBOUND
      proxy:
        proxyVersion: '^1\.*.*'
      listener:
        filterChain:
          filter:
            name: envoy.filters.network.http_connection_manager
            subFilter:
              name: envoy.filters.http.router
    patch:
      operation: INSERT_BEFORE
      value:
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.http.wasm.v3.Wasm
          config:
            name: example-filter
            rootId: my_root_id
            vmConfig:
              code:
                local:
                  filename: /var/local/lib/wasm-filters/example-filter.wasm
              runtime: envoy.wasm.runtime.v8
              vmId: example-filter
              allow_precompiled: true
        name: envoy.filters.http.wasm
  workloadSelector:
    labels:
      app: productpage
      version: v1
```

9. Insert the WASM filter into the HTTP-level filter chain of the details service.

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: details-v1-examplefilter
  labels:
    asm-system: 'true'
    provider: asm
spec:
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      context: SIDECAR_INBOUND
      proxy:
        proxyVersion: '^1\.*.*'
      listener:
        filterChain:
          filter:
            name: envoy.filters.network.http_connection_manager
            subFilter:
              name: envoy.filters.http.router
    patch:
      operation: INSERT_BEFORE
      value:
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.http.wasm.v3.Wasm
          config:
            name: example-filter
            rootId: my_root_id
            vmConfig:
              code:
                local:
                  filename: /var/local/lib/wasm-filters/example-filter.wasm
              runtime: envoy.wasm.runtime.v8
              vmId: example-filter
              allow_precompiled: true
        name: envoy.filters.http.wasm
  workloadSelector:
    labels:
      app: details
      version: v1
```

Verify the WASM filter

1. Enter the ingress gateway address of the application in the address bar of your browser and send traffic to the productpage service. The response indicates that the header of the WASM filter is added to the response header, as shown in the following figure.



2. Run the following command to send traffic to the details service. The response indicates that the header of the WASM filter is added to the response header.

```
kubectl exec -ti  deploy/productpage-v1 -c istio-proxy -- curl -v http://details:9080/details/123
```

```
*   Trying 172.31.13.58...
* TCP_NODELAY set
* Connected to details (172.31.13.58) port 9080 (#0)
> GET /details/123 HTTP/1.1
> Host: details:9080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
xxxxxxx
< resp-header-demo: added by our filter
xxxxx
* Connection #0 to host details left intact
xxxxx
```

5.Use ORAS to simplify Wasm-based ASM instance extension

WebAssembly (Wasm) is an effective and portable binary instruction format. You can use Wasm to extend the data plane of an Alibaba Cloud Service Mesh (ASM) instance with new features. However, building, deploying, and running Wasm filters in an ASM instance are complex. This topic shows you how to use OCI Registry as Storage (ORAS) to simplify Wasm-based ASM instance extension.

Prerequisites

- An ASM instance is created, and a Container Service for Kubernetes (ACK) cluster is added to the ASM instance. For more information, see Create an ASM instance and Add a cluster to an ASM instance.

  Note The version of the ASM instance must be v1.8.3.17-g1399628c-aliyun or later.

- The kubectl client is used to connect to the ASM instance. For more information, see Use kubectl to connect to an ASM instance.
- An application is deployed in the ACK cluster that is added to the ASM instance. For more information, see Deploy an application in an ASM instance.
- A virtual service and an ingress gateway are created. For more information, see Define Istio resources.
- A binary Wasm filter file is created and compiled. In this example, the header of a Wasm filter is added to the response header.

Context

ASM supports Wasm. You can deploy Wasm filters in the Envoy that is used to manage clusters on the data plane. This helps you extend the data plane with new features. ORAS provides registry storage based on the Open Container Initiative (OCI) Artifacts project. You can use ORAS to simplify the storage process in OCI registries. When you use Wasm to extend the data plane of an ASM instance with new features, you can use ORAS to simplify the extension process.

Push a Wasm filter

Use ORAS CLI to push a Wasm filter to an image repository. In this example, Container Registry Enterprise Edition is used.

1. Create an image repository in Container Registry Enterprise Edition and obtain the username that is used to log on to the image repository. For more information, see Use a Container Registry Enterprise Edition instance to push and pull images.
2. Run the following command to log on to the image repository:

   ```
   oras login --username=<Username> acree-1-registry.cn-hangzhou.cr.aliyuncs.com
   ```

3. Run the following command to push a Wasm filter to the image repository:

   ```
   oras push acree-1-registry.cn-hangzhou.cr.aliyuncs.com/******/asm-test:v0.1 --manifest-
   config runtime-config.json:application/vnd.module.wasm.config.v1+json  example-filter.w
   asm:application/vnd.module.wasm.content.layer.v1+wasm
   ```

4. View the pushed Wasm filter in the Container Registry console.
   i. Log on to the Container Registry console.

ii. In the top navigation bar, select a region.
iii. In the left-side navigation pane, click **Instances**.
iv. On the **Instances** page, click the required Container Registry Enterprise Edition instance.
v. On the management page of the Container Registry Enterprise Edition instance, choose **Repositories > Repositories** in the left-side navigation pane.
vi. On the **Repositories** page, click the name of the image repository to which you want to log on.
vii. In the left-side navigation pane of the repository information page, click **Tags**. On the page that appears, you can view the pushed Wasm filter.

Enable Wasm-based ASM instance extension

Wasm-based ASM instance extension in the ASM console

1. Log on to the ASM console.
2. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
3. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
4. On the management page of the ASM instance, click **Settings** in the upper-right corner.
5. In the **Settings Update** panel, select **Enable Wasm-based ASM Instance Extension** in the **Data Plane Extension** section and click **OK**.

> **Note** To disable Wasm-based ASM instance extension, clear **Enable Wasm-based ASM Instance Extension**.

Use Alibaba Cloud CLI to enable Wasm-based ASM instance extension

You can use Alibaba Cloud CLI to enable Wasm-based ASM instance extension. Run the following command to enable Wasm-based ASM instance extension:

```
aliyun servicemesh UpdateMeshFeature --ServiceMeshId=xxxx --WebAssemblyFilterEnabled=true
```

Run the following command to disable Wasm-based ASM instance extension:

```
aliyun servicemesh UpdateMeshFeature --ServiceMeshId=xxxx --WebAssemblyFilterEnabled=false
```

Use the Wasm filter in ASM

ASM provides ASMFilterDeployment resources and related controllers. A controller listens to the status of its ASMFilterDeployment resource and performs the following operations:

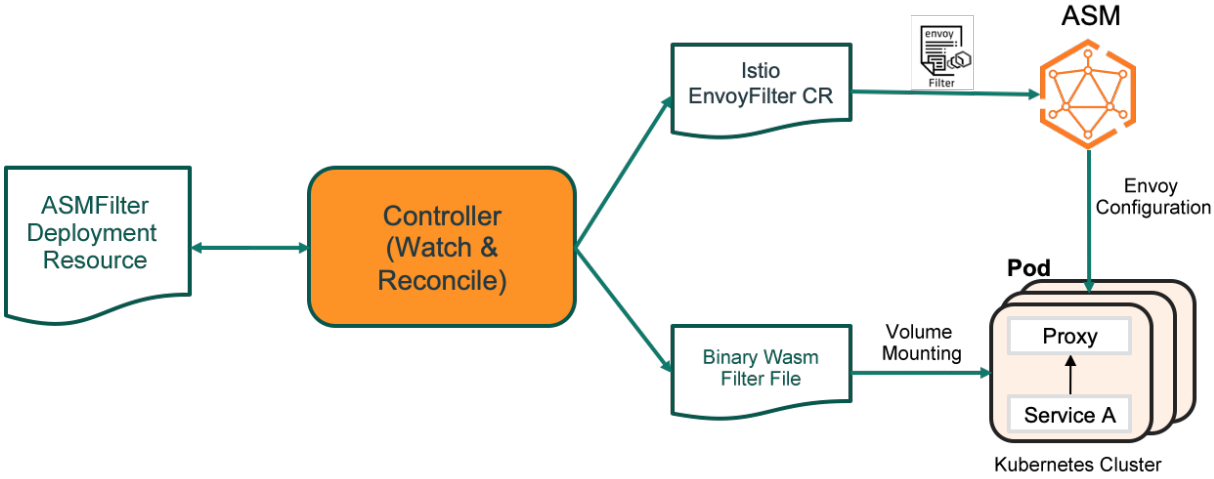
- Creates an Envoy filter and pushes the Envoy filter to the control plane.
- Pulls the Wasm filter image from the image repository and mounts the image to the pod that is controlled by the specified workload.

Process

1. Enable Wasm-based ASM instance extension. After that, the ASM instance automatically deploys an asmwasm-controller DaemonSet to the ACK cluster that is added to the ASM instance.
2. The asmwasm-controller DaemonSet listens to a ConfigMap. The ConfigMap stores the address of

the image repository where the Wasm filter image is to be pulled.

3. If authorization and authentication are required, the asmwasm-controller DaemonSet obtains a secret based on the pullSecret parameter.
4. The asmwasm-controller DaemonSet calls the ORAS operation to dynamically pull the Wasm filter from the OCI registry.
5. The asmwasm-controller DaemonSet uses a hostPath volume to mount the Wasm filter. Therefore, the pulled Wasm filter is stored in the corresponding nodes of the ACK cluster.



Procedure

1. Run the following command to enable Wasm-based ASM instance extension:

```
aliyun servicemesh UpdateMeshFeature  --ServiceMeshId=xxxxxx --WebAssemblyFilterEnabled
=true
```

2. Create a secret for the ACK cluster to access the image repository.

   For more information, see Secret.

   i. Create a *myconfig.json* file that contains the following code:

```
{
    "auths":{
        "**********.cn-hangzhou.cr.aliyuncs.com":{
            "username":"*****username*****",
            "password":"*****password*****"
        }
    }
}
```

   - `**********.cn-hangzhou.cr.aliyuncs.com` : the address of the image repository.
   - `username` : the username of the image repository.
   - `password` : the password of the image repository.

ii. Run the following command to create a secret:

> **Note** The secret must be named asmwasm-cache and belong to the istio-system namespace.

```
kubectl create secret generic asmwasm-cache -n istio-system --from-file=.dockerconfigjson=myconfig.json --type=kubernetes.io/dockerconfigjson
```

3. Deploy an ASMFilterDeployment resource.

   i. Create a *filter.yaml* file that contains the following code:

```
apiVersion: istio.alibabacloud.com/v1beta1
kind: ASMFilterDeployment
metadata:
  name: details-v1-wasmfiltersample
spec:
  workload:
    kind: Deployment
    labels:
      app: details
      version: v1
  filter:
    patchContext: 'SIDECAR_INBOUND'
    parameters: '{"name":"hello","value":"hello details"}'
    image: 'acree-1-registry.cn-hangzhou.cr.aliyuncs.com/asm/asm-test:v0.1'
    rootID: 'my_root_id'
    id: 'details-v1-wasmfiltersample.default'
```

   - Parameters in `workload`:
     a. `kind`: the type of the destination workload.
     b. `labels`: the filter conditions.
   - Parameters in `filter`:
     a. `patchContext`: the context that takes effect.
     b. `parameters`: the parameters that are required to run the Wasm filter.
     c. `image`: the address of the image repository to which the Wasm filter is pushed.
     d. `rootID`: the root ID of the Wasm filter.
     e. `id`: the unique ID of the Wasm filter.

ii. Run the following command to deploy the ASMFilterDeployment resource:

```
kubectl apply -f filter.yaml
```

After the ASMFilterDeployment resource is deployed, ASM automatically generates an Envoy filter. In the Envoy filter, the match parameter defines an envoy.router filter, and the patch parameter defines an INSERT_BEFORE operation.

- match parameter

- patch parameter

4. Check the definition of the workload to which the Wasm filter is mounted.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
....
spec:
   ...
   template:
      metadata:
          annotations:
              sidecar.istio.io/userVolume: '[{"name":"wasmfilters-dir","hostPath":{"pat
h":"/var/local/lib/wasm-filters"}}]'
              sidecar.istio.io/userVolumeMount: '[{"mountPath":"/var/local/lib/wasm-fil
ters","name":"wasmfilters-dir"}]'
```

The preceding definition indicates that the Wasm filter is mounted to the istio-proxy container of the destination application by using a hostPath volume.

Check whether the Wasm filter takes effect

Run the following command to log on to the istio-proxy container of the productpage pod and send a request to access the details service:

```
kubectl exec -ti  deploy/productpage-v1 -c istio-proxy -- curl -v http://details:9080/details/123
```

Expected output:

```
*   Trying 172.21.9.191...
* TCP_NODELAY set
* Connected to details (172.21.9.191) port 9080 (#0)
> GET /details/123 HTTP/1.1
> Host: details:9080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
xxxxxxx
< resp-header-demo: added by our filter
xxxxx
* Connection #0 to host details left intact
xxxxx
```

The response indicates that the header of the Wasm filter is added to the response header.

6.Enable the DNS proxy feature for an ASM instance

Istio 1.8 and later versions enable sidecar proxies to serve as Domain Name System (DNS) proxies. When an Alibaba Cloud Service Mesh (ASM) instance with the DNS proxy feature enabled receives DNS queries from applications, the specified sidecar proxy transparently intercepts the queries and resolves the DNS information in these queries. This topic describes how to enable the DNS proxy feature for an ASM instance.

Prerequisites

- Create an ASM instance

> **Note** The version of the ASM instance must be v1.8.3.17-g1399628c-aliyun or later.

- Add a cluster to an ASM instance

Context

ASM uses Kubernetes services and defined service entries to configure hostname-to-IP-address mappings for all services that an application may access. The specified sidecar proxy transparently intercepts DNS queries that are sent from the application and resolves the DNS information in these queries.

- If the application queries a service that is deployed in an ASM instance, the sidecar proxy directly responds to the application.
- If the application queries a service that is not deployed in an ASM instance, the sidecar proxy forwards the query to the upstream name servers that are defined in `/etc/resolv.conf`.

Enable the DNS proxy feature for an ASM instance

Use the ASM console to enable the DNS proxy feature

1. Log on to the ASM console.
2. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
3. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
4. On the Instance Information page of the ASM instance, click **Settings** in the upper-right corner.
5. In the **Settings Update** panel, select **Enable DNS Proxy** in the **Traffic Management** section. Then, click **OK**.

> **Note** To disable the DNS proxy feature, clear **Enable DNS Proxy** in the **Traffic Management** section in the **Settings Update** panel.

Use Alibaba Cloud CLI to enable the DNS proxy feature

You can enable the DNS proxy feature for an ASM instance by using Alibaba Cloud CLI. Run the following command to enable the DNS proxy feature:

```
aliyun servicemesh UpdateMeshFeature --ServiceMeshId=xxxx --DNSProxyingEnabled=true
```

To disable the DNS proxy feature, run the following command:

```
aliyun servicemesh UpdateMeshFeature --ServiceMeshId=xxxx --DNSProxyingEnabled=false
```

Verify the DNS proxy feature

1. Create a service entry in an ASM instance with the DNS proxy feature enabled.

   Use the service entry to add https://aliyun.com to the service registry that is internally maintained by the ASM instance.

   i. Log on to the ASM console.

   ii. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.

   iii. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.

   iv. On the details page of the ASM instance, choose **Traffic Management > ServiceEntry** in the left-side navigation pane. On the Service entry page, click **Create from YAML**.

   v. In the **Create** panel, select a namespace from the **Namespace** drop-down list, enter code to configure a service entry in the code editor, and then click **OK**.

   ```
   apiVersion: networking.istio.io/v1beta1
   kind: ServiceEntry
   metadata:
    name: mydnsproxying-sample
   spec:
    hosts:
    - aliyun.com
    location: MESH_EXTERNAL
    ports:
    - number: 443
      name: https
      protocol: TLS
    resolution: DNS
   ```

2. Deploy a sleep service in a Container Service for Kubernetes (ACK) cluster that is added to the ASM instance.

   i. Create a *sleep.yaml* file that contains the following code:

```
##################################################################################
###############
# Sleep service
##################################################################################
###############
apiVersion: v1
kind: Service
metadata:
  name: sleep
  labels:
    app: sleep
spec:
  ports:
  - port: 80
    name: http
  selector:
    app: sleep
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sleep
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sleep
  template:
    metadata:
      labels:
        app: sleep
    spec:
      containers:
      - name: sleep
        image: pstauffer/curl
        command: ["/bin/sleep", "3650d"]
        imagePullPolicy: IfNotPresent
---
```

ii. Run the following command to deploy the sleep service:

```
kubectl apply -f sleep.yaml
```

3. Run the following command to log on to the container of the sleep service and use a curl command to access the URL of https://aliyun.com:

```
kubectl --kubeconfig=config.aliyun.worker.k8s -n mytest exec -it deploy/sleep -c sleep
-- sh -c "curl -v https://aliyun.com""
```

The following output is expected:

```
* Rebuilt URL to: https://aliyun.com"
*   Trying 240.240.**.**...
* TCP_NODELAY set
* Connected to aliyun.com (240.240.**.**) port 443 (#0)
```

The output indicates that the IP address 240.240.**.** is returned. The IP address is not an actual public IP address. Instead, it is a virtual IP address that is automatically assigned by the ASM instance. This is because the ASM instance uses iptables to intercept requests that are sent to the kube-dns service and routes the requests to the sidecar proxy that runs in the pod of the sleep service. When the pod of the sleep service resolves aliyun.com to a virtual IP address and sends a request, the virtual IP address is translated into the actual public IP address that is resolved by the sidecar proxy.

In this example, a service entry is created and the hostname of aliyun.com is added to the service entry. When the pod of the sleep service queries aliyun.com from the Istio DNS, the virtual IP address of aliyun.com is returned. If the pod uses the virtual IP address to send a request by using the sidecar proxy, the virtual IP address is translated into the actual public IP address.

7.Upgrade the data plane of an ASM instance without service interruption (Beta)

In an Alibaba Cloud Service Mesh (ASM) instance, the sidecars on the data plane manipulate the traffic of all applications in the clusters that are managed by the ASM instance. To upgrade the data plane, you must restart sidecar containers. This may result in failed requests and application service interruption. ASM allows you to upgrade the data plane without interrupting services or affecting applications. This topic shows you how to upgrade the data plane of an ASM instance without service interruption. In this topic, an ASM instance of which the Istio version is 1.6.x is used as an example. An NGINX application is deployed on the ASM instance. The HTTP stress testing tool go-stress-testing is used to continuously access the NGINX application. During this process, the data plane is upgraded without service interruption.

Prerequisites

- An ASM instance of which the Istio version is 1.6.x is created. For more information, see Create an ASM instance.
- A cluster is added to the ASM instance. For more information, see Add a cluster to an ASM instance.
- An ingress gateway service is deployed on the ASM instance. For more information, see Deploy an ingress gateway service.
- Automatic sidecar injection is disabled for namespaces. If this feature is enabled, you must disable it. For more information, see Install a sidecar proxy.

Precautions

To upgrade the data plane of an ASM instance without service interruption, you must use an OpenKruise SidecarSet (SidecarSet), which can be used to switch sidecar containers. If an application requires no service interruption during data plane upgrade, you must use a SidecarSet to inject sidecars to the pods of the application. In this case, the sidecars must be injected when you create a deployment for the application. You can use the following two methods to inject sidecars:

> **Note** We recommend that you inject sidecars when you deploy your application. If automatic sidecar injection is enabled for your application, you can change the injection mode and recreate the related pods. However, the pods are unavailable for a moment, which may cause service interruption.

- Deploy deployments and pods that require no service interruption during data plane upgrade in an independent namespace.

  Deploy deployments and pods that require no service interruption during data plane upgrade in an independent namespace. This way, you can use a SidecarSet to inject sidecars in this namespace and enable automatic sidecar injection for other namespaces.

- Disable automatic sidecar injection for specific pods and use a SidecarSet to inject sidecars into these pods.

If automatic sidecar injection is enabled for the namespace of a pod, you can disable the feature by using pod annotations. Then, you can use the matching policy of the SidecarSet to match the pod for sidecar injection.

Step 1 Install OpenKruise in a cluster on the data plane

ASM does not automatically install OpenKruise in a cluster on the data plane. You must to manually install OpenKruise by using Helm.

1. Install the Helm plug-in of Alibaba Cloud. For more information, see 推送和拉取Helm Chart.
2. Add the Helm repository address of OpenKruisem to Helm.

```
helm repo add acr-openkruise-asm acr://openkruise-chart.cn-hangzhou.cr.aliyuncs.com/ope
nkruise/kruise-asm
```

3. Install OpenKruise in the cluster.

```
helm install kruise acr-openkruise-asm/kruise-asm --version 0.1.0
```

Step 2: Deploy a ConfigMap

When you configure a SidecarSet, you must specify the ID of the cluster on the data plane. To avoid manually specifying the cluster ID for each SidecarSet, you can deploy a ConfigMap.

1. Create a file named *configmap.yaml*.

```
apiVersion: v1
data:
  clusterid: $$$CLUSTER-ID$$$
kind: ConfigMap
metadata:
  name: ack-cluster-profile
  namespace: default
```

   Replace `$$$CLUSTER-ID$$$` with the ID of the cluster on the data plane.

2. Deploy a ConfigMap.

```
kubectl apply -f configmap.yaml
```

Step 3: Deploy a SidecarSet

The sidecar injection configuration of an application contains parameters that cannot be configured at a time. To resolve this issue, you must deploy an independent SidecarSet for each deployment to configure sidecar injection.

1. Create a file named *nginx-sidecarset.json*.

   In the following code, the template in the References section is modified to apply to the SidecarSet in this example. For more information about how to customize a SidecarSet, see References.

```
{
    "apiVersion": "apps.kruise.io/v1alpha1",
    "kind": "SidecarSet",
    "metadata": {
        "name": "sidecarset-example"
    },
```

```
    },
    "spec": {
        "containers": [
            {
                "args": [
                    "proxy",
                    "sidecar",
                    "--domain",
                    "$(POD_NAMESPACE).svc.cluster.local",
                    "--serviceCluster",
                    "$(ISTIO_META_WORKLOAD_NAME). $(POD_NAMESPACE)",
                    "--drainDuration",
                    "45s",
                    "--parentShutdownDuration",
                    "1m0s",
                    "--discoveryAddress",
                    "istiod.istio-system.svc:15012",
                    "--zipkinAddress",
                    "zipkin.istio-system:9411",
                    "--proxyLogLevel=warning",
                    "--proxyComponentLogLevel=misc:error",
                    "--proxyAdminPort",
                    "15000",
                    "--concurrency",
                    "2",
                    "--controlPlaneAuthPolicy",
                    "NONE",
                    "--dnsRefreshRate",
                    "300s",
                    "--statusPort",
                    "15021",
                    "--trust-domain=cluster.local",
                    "--controlPlaneBootstrap=false"
                ],
                "env": [
                    {
                        "name": "JWT_POLICY",
                        "value": "first-party-jwt"
                    },
                    {
                        "name": "PILOT_CERT_PROVIDER",
                        "value": "istiod"
                    },
                    {
                        "name": "CA_ADDR",
                        "value": "istiod.istio-system.svc:15012"
                    },
                    {
                        "name": "POD_NAME",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.name"
                            }
                        }
```

```
                    },
                    {
                        "name": "POD_NAMESPACE",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.namespace"
                            }
                        }
                    },
                    {
                        "name": "INSTANCE_IP",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "status.podIP"
                            }
                        }
                    },
                    {
                        "name": "SERVICE_ACCOUNT",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "spec.serviceAccountName"
                            }
                        }
                    },
                    {
                        "name": "CANONICAL_SERVICE",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.labels['service.istio.io/canonic
al-name']"
                            }
                        }
                    },
                    {
                        "name": "CANONICAL_REVISION",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.labels['service.istio.io/canonic
al-revision']"
                            }
                        }
                    },
                    {
                        "name": "PROXY_CONFIG",
                        "value": "{\"configPath\":\"/etc/istio/proxy\",\"proxyMetadata\
":{\"DNS_AGENT\":\"\"}}\n"
                    },
                    {
```

```
                "name": "ISTIO_META_POD_PORTS",
                "value": "[\n]"
            },
            {
                "name": "ISTIO_META_CLUSTER_ID",
                "valueFrom": {
                    "configMapKeyRef": {
                        "name": "ack-cluster-profile",
                        "key": "clusterid"
                    }
                }
            },
            {
                "name": "ISTIO_META_POD_NAME",
                "valueFrom": {
                    "fieldRef": {
                        "apiVersion": "v1",
                        "fieldPath": "metadata.name"
                    }
                }
            },
            {
                "name": "ISTIO_META_CONFIG_NAMESPACE",
                "valueFrom": {
                    "fieldRef": {
                        "apiVersion": "v1",
                        "fieldPath": "metadata.namespace"
                    }
                }
            },
            {
                "name": "ISTIO_META_INTERCEPTION_MODE",
                "value": "REDIRECT"
            },
            {
                "name": "ISTIO_METAJSON_ANNOTATIONS",
                "value": "{\"kubernetes.io/psp\":\"ack.privileged\"}\n"
            },
            {
                "name": "ISTIO_META_WORKLOAD_NAME",
                "valueFrom": {
                    "fieldRef": {
                        "apiVersion": "v1",
                        "fieldPath": "metadata.labels['app']"
                    }
                }
            },
            {
                "name": "ISTIO_META_MESH_ID",
                "value": "cluster.local"
            },
            {
                "name": "DNS_AGENT"
            },
```

```
                    {
                        "name": "TERMINATION_DRAIN_DURATION_SECONDS",
                        "value": "5"
                    }
                ],
                "image": "registry.cn-hangzhou.aliyuncs.com/acs/asm-istio-proxy:feature
-1.6.x-faee4bb874d29dabde41481b695718c5b73b6b04-1531",
                "imagePullPolicy": "IfNotPresent",
                "name": "istio-proxy",
                "podInjectPolicy": "BeforeAppContainer",
                "lifecycle": {
                    "postStart": {
                        "exec": {
                            "command": ["/bin/sh", "-c", "/usr/local/bin/pilot-agent wa
it"]
                        }
                    }
                },
                "ports": [
                    {
                        "containerPort": 15090,
                        "name": "http-envoy-prom",
                        "protocol": "TCP"
                    }
                ],
                "resources": {
                    "limits": {
                        "cpu": "2",
                        "memory": "1Gi"
                    },
                    "requests": {
                        "cpu": "100m",
                        "memory": "128Mi"
                    }
                },
                "securityContext": {
                    "allowPrivilegeEscalation": false,
                    "capabilities": {
                        "drop": [
                            "ALL"
                        ]
                    },
                    "privileged": false,
                    "readOnlyRootFilesystem": true,
                    "runAsGroup": 1337,
                    "runAsNonRoot": true,
                    "runAsUser": 1337
                },
                "terminationMessagePath": "/dev/termination-log",
                "terminationMessagePolicy": "File",
                "upgradeStrategy": {
                    "upgradeType": "HotUpgrade",
                    "hotUpgradeEmptyImage": "registry.cn-hangzhou.aliyuncs.com/acs/asm-
istio-proxy-empty:feature-1.6.x-511e4bb6e85be2c753a46d620efb1973251c1778"
```

```
                },
                "volumeMounts": [
                    {
                        "mountPath": "/var/run/secrets/istio",
                        "name": "istiod-ca-cert"
                    },
                    {
                        "mountPath": "/var/lib/istio/data",
                        "name": "istio-data"
                    },
                    {
                        "mountPath": "/etc/istio/proxy",
                        "name": "istio-envoy"
                    },
                    {
                        "mountPath": "/etc/istio/pod",
                        "name": "istio-podinfo"
                    },
                    {
                        "mountPath": "/etc/asm/uds/",
                        "name": "asm-hotupgrade-data"
                    }
                ]
            }
        ],
        "initContainers": [
            {
                "args": [
                    "istio-iptables",
                    "-p",
                    "15001",
                    "-z",
                    "15006",
                    "-u",
                    "1337",
                    "-m",
                    "REDIRECT",
                    "-i",
                    "*",
                    "-x",
                    "172.23.0.1/32",
                    "-b",
                    "*",
                    "-d",
                    "15090,15021,15021"
                ],
                "env": [
                    {
                        "name": "DNS_AGENT"
                    }
                ],
                "image": "registry-vpc.cn-zhangjiakou.aliyuncs.com/acs/proxyv2:1.6.8",
                "imagePullPolicy": "IfNotPresent",
                "name": "istio-init",
                "resources": {
```

```
                "resources": {
                    "limits": {
                        "cpu": "100m",
                        "memory": "50Mi"
                    },
                    "requests": {
                        "cpu": "10m",
                        "memory": "10Mi"
                    }
                },
                "securityContext": {
                    "allowPrivilegeEscalation": false,
                    "capabilities": {
                        "add": [
                            "NET_ADMIN",
                            "NET_RAW"
                        ],
                        "drop": [
                            "ALL"
                        ]
                    },
                    "privileged": false,
                    "readOnlyRootFilesystem": false,
                    "runAsGroup": 0,
                    "runAsNonRoot": false,
                    "runAsUser": 0
                },
                "terminationMessagePath": "/dev/termination-log",
                "terminationMessagePolicy": "File",
                "upgradeStrategy": {}
            }
        ],
        "selector": {
            "matchExpressions": [
                {
                    "key": "app",
                    "operator": "In",
                    "values": [
                        "nginx"
                    ]
                },
                {
                    "key": "sidecarset-injected",
                    "operator": "In",
                    "values": [
                        "true"
                    ]
                }
            ]
        },
        "strategy": {
            "type": "RollingUpdate",
            "partition": 0,
            "maxUnavailable": 1
        },
```

```
        "volumes": [
            {
                "emptyDir": {},
                "name": "asm-hotupgrade-data"
            },
            {
                "emptyDir": {
                    "medium": "Memory"
                },
                "name": "istio-envoy"
            },
            {
                "emptyDir": {},
                "name": "istio-data"
            },
            {
                "downwardAPI": {
                    "defaultMode": 420,
                    "items": [
                        {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.labels"
                            },
                            "path": "labels"
                        },
                        {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.annotations"
                            },
                            "path": "annotations"
                        }
                    ]
                },
                "name": "istio-podinfo"
            },
            {
                "configMap": {
                    "defaultMode": 420,
                    "name": "istio-ca-root-cert"
                },
                "name": "istiod-ca-cert"
            }
        ]
    }
}
```

2. Apply the nginx-sidecarset.json file to the cluster on the data plane.

```
kubectl apply -f nginx-sidecarset.json
```

Step 4: Deploy an NGINX application

1. Deploy an NGINX application.
   i. Create a file named *nginx.yaml*.

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
        sidecarset-injected: "true"
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: ClusterIP
```

   ii. Deploy an NGINX application.

```
kubectl apply -f nginx.yaml
```

2. Expose the service port of the NGINX application to the ingress gateway and create a routing rule.

i. Create a file named *nginx-gateway.yaml*.

```
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: nginx-gateway
  namespace: default
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - '*'
    port:
      name: http
      number: 8080
      protocol: HTTP
---
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: nginx
  namespace: default
spec:
  gateways:
  - nginx-gateway
  hosts:
  - '*'
  http:
  - match:
    - uri:
        exact: /
    route:
    - destination:
        host: nginx
        port:
          number: 80
```

ii. Apply the *nginx-gateway.yaml* file.

```
kubectl apply -f nginx-gateway.yaml
```

3. Verify whether the NGINX application is deployed.

i. Check whether the pod is started.

```
kubectl get pod
```

The following output is expected:

```
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-6c9b9677d4-rlvsn   3/3     Running             0          1m
```

If `Running` is displayed in the `STATUS` column, the pod is started.

ii. Access port 8080 of the IP address of the ingress gateway to check whether NGINX is running as expected.

If the following page is displayed, the NGINX application is deployed.

Step 5: Use go-stress-testing to access the NGINX application

The go-stress-testing tool is an HTTP stress testing tool the is developed in Go. This tool is compatible with multiple platforms. In this example, this tool is used to continuously access the NGINX application. During the continuous access, the data plane is upgraded without service interruption. This tool counts the numbers of successful and failed requests.

1. Download go-stress-testing. To download go-stress-testing, visit go-stress-testing.
2. Start to access the NGINX application.

   Four concurrent processes are started to access the server. Each process sends a total of 100,000 requests.

   ```
   go-stress-testing-mac -c 4 -n 100000 -u http://IP address of ingress gateway:8080
   ```

   After the access starts, the statistics about the return codes of the requests are returned.

Step 6: Upgrade the data plane without service interruption

1. Edit the SidecarSet.

   ```
   kubectl edit sidecarset sidecarset-example
   ```

2. Modify the value of the image parameter of the sidecar to the URL of the image of the new SidecarSet. Then, save the modification and exit.

   ```
   registry.cn-hangzhou.aliyuncs.com/acs/asm-istio-proxy:feature-1.6.x-faee4bb874d29dabde4
   1481b695718c5b73b6b04-1546
   ```

3. Check whether services are interrupted when the data plane is upgraded.

i. Query the upgrade status.

```
kubectl describe pod nginx-deployment-76f4578864-js5hc |grep Image:
```

The following output is expected:

```
    Image:          registry-vpc.cn-zhangjiakou.aliyuncs.com/acs/proxyv2:1.6.8
    Image:          registry.cn-hangzhou.aliyuncs.com/acs/asm-istio-proxy-empty:feat
ure-1.6.x-511e4bb6e85be2c753a46d620efb1973251c1778
    Image:          registry.cn-hangzhou.aliyuncs.com/acs/asm-istio-proxy:feature-1.
6.x-faee4bb874d29dabde41481b695718c5b73b6b04-1546
    Image:           nginx:1.14.2
```

When containers in the pod are changed to the asm-istio-proxy-empty, asm-isitio-proxy, and nginx images, the upgrade is completed.

ii. After the upgrade is completed, view the output of the go-stress-testing-mac command, which is described in Step 5: Use go-stress-testing to access the NGINX application. The following figure shows that the return code of all requests is 200. This indicates that no request failed during the upgrade.

1075s	4	164430	0	153.46	1810.75	14.69	6.52	200:164430
1076s	4	164592	0	153.47	1810.75	14.69	6.52	200:164592
1077s	4	164722	0	153.45	1810.75	14.69	6.52	200:164722
1078s	4	164868	0	153.44	1810.75	14.69	6.52	200:164868
1079s	4	165038	0	153.46	1810.75	14.69	6.52	200:165038
1080s	4	165224	0	153.50	1810.75	14.69	6.51	200:165224
1081s	4	165386	0	153.50	1810.75	14.69	6.51	200:165386
1082s	4	165552	0	153.51	1810.75	14.69	6.51	200:165552
1083s	4	165700	0	153.51	1810.75	14.69	6.51	200:165700
1084s	4	165866	0	153.52	1810.75	14.69	6.51	200:165866

References

Customize a SidecarSet

To customize a SidecarSet to configure sidecar injection, you must use the template file that corresponds to your Istio version. The following code provides an example in which Istio 1.6.x is used. When you use the template file, you must replace the parameters in the template file based on the following requirements:

```
{
    "apiVersion": "apps.kruise.io/v1alpha1",
    "kind": "SidecarSet",
    "metadata": {
        "name": "sidecarset-example"
    },
    "spec": {
        "containers": [
            {
                "args": [
                    "proxy",
                    "sidecar",
                    "--domain",
                    "$(POD_NAMESPACE).svc.cluster.local",
                    "--serviceCluster",
                    "$(ISTIO_META_WORKLOAD_NAME). $(POD_NAMESPACE)",
                    "--drainDuration",
                    "45s",
                    "--parentShutdownDuration"
```

```
                "--parentShutdownDuration",
                "1m0s",
                "--discoveryAddress",
                "istiod.istio-system.svc:15012",
                "--zipkinAddress",
                "zipkin.istio-system:9411",
                "--proxyLogLevel=warning",
                "--proxyComponentLogLevel=misc:error",
                "--proxyAdminPort",
                "15000",
                "--concurrency",
                "2",
                "--controlPlaneAuthPolicy",
                "NONE",
                "--dnsRefreshRate",
                "300s",
                "--statusPort",
                "15021",
                "--trust-domain=cluster.local",
                "--controlPlaneBootstrap=false"
            ],
            "env": [
                {
                    "name": "JWT_POLICY",
                    "value": "first-party-jwt"
                },
                {
                    "name": "PILOT_CERT_PROVIDER",
                    "value": "istiod"
                },
                {
                    "name": "CA_ADDR",
                    "value": "istiod.istio-system.svc:15012"
                },
                {
                    "name": "POD_NAME",
                    "valueFrom": {
                        "fieldRef": {
                            "apiVersion": "v1",
                            "fieldPath": "metadata.name"
                        }
                    }
                },
                {
                    "name": "POD_NAMESPACE",
                    "valueFrom": {
                        "fieldRef": {
                            "apiVersion": "v1",
                            "fieldPath": "metadata.namespace"
                        }
                    }
                },
                {
                    "name": "INSTANCE_IP",
                    "valueFrom": {
```

```
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "status.podIP"
                            }
                        }
                    },
                    {
                        "name": "SERVICE_ACCOUNT",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "spec.serviceAccountName"
                            }
                        }
                    },
                    {
                        "name": "CANONICAL_SERVICE",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.labels['service.istio.io/canonical-n
ame']"
                            }
                        }
                    },
                    {
                        "name": "CANONICAL_REVISION",
                        "valueFrom": {
                            "fieldRef": {
                                "apiVersion": "v1",
                                "fieldPath": "metadata.labels['service.istio.io/canonical-r
evision']"
                            }
                        }
                    },
                    {
                        "name": "PROXY_CONFIG",
                        "value": "{\"configPath\":\"/etc/istio/proxy\",\"proxyMetadata\":{\
"DNS_AGENT\":\"\"}}\n"
                    },
                    {
                        "name": "ISTIO_META_POD_PORTS",
                        "value": "[\n]"
                    },
                    {
                        "name": "ISTIO_META_CLUSTER_ID",
                        "valueFrom": {
                            "configMapKeyRef": {
                                "name": "ack-cluster-profile",
                                "key": "clusterid"
                            }
                        }
                    },
                    {
```

```
                "name": "ISTIO_META_POD_NAME",
                "valueFrom": {
                    "fieldRef": {
                        "apiVersion": "v1",
                        "fieldPath": "metadata.name"
                    }
                }
            },
            {
                "name": "ISTIO_META_CONFIG_NAMESPACE",
                "valueFrom": {
                    "fieldRef": {
                        "apiVersion": "v1",
                        "fieldPath": "metadata.namespace"
                    }
                }
            },
            {
                "name": "ISTIO_META_INTERCEPTION_MODE",
                "value": "REDIRECT"
            },
            {
                "name": "ISTIO_METAJSON_ANNOTATIONS",
                "value": "{\"kubernetes.io/psp\":\"ack.privileged\"}\n"
            },
            {
                "name": "ISTIO_META_WORKLOAD_NAME",
                "valueFrom": {
                    "fieldRef": {
                        "apiVersion": "v1",
                        "fieldPath": "metadata.labels['app']"
                    }
                }
            },
            {
                "name": "ISTIO_META_MESH_ID",
                "value": "cluster.local"
            },
            {
                "name": "DNS_AGENT"
            },
            {
                "name": "TERMINATION_DRAIN_DURATION_SECONDS",
                "value": "5"
            }
        ],
        "image": "$$$IMAGE$$$",
        "imagePullPolicy": "IfNotPresent",
        "name": "istio-proxy",
        "podInjectPolicy": "BeforeAppContainer",
        "lifecycle": {
            "postStart": {
                "exec": {
                    "command": ["/bin/sh", "-c", "/usr/local/bin/pilot-agent wait"]
```

```
                        }
                    }
                },
                "ports": [
                    {
                        "containerPort": 15090,
                        "name": "http-envoy-prom",
                        "protocol": "TCP"
                    }
                ],
                "resources": {
                    "limits": {
                        "cpu": "2",
                        "memory": "1Gi"
                    },
                    "requests": {
                        "cpu": "100m",
                        "memory": "128Mi"
                    }
                },
                "securityContext": {
                    "allowPrivilegeEscalation": false,
                    "capabilities": {
                        "drop": [
                            "ALL"
                        ]
                    },
                    "privileged": false,
                    "readOnlyRootFilesystem": true,
                    "runAsGroup": 1337,
                    "runAsNonRoot": true,
                    "runAsUser": 1337
                },
                "terminationMessagePath": "/dev/termination-log",
                "terminationMessagePolicy": "File",
                "upgradeStrategy": {
                    "upgradeType": "HotUpgrade",
                    "hotUpgradeEmptyImage": "registry.cn-hangzhou.aliyuncs.com/acs/asm-isti
o-proxy-empty:feature-1.6.x-511e4bb6e85be2c753a46d620efb1973251c1778"
                },
                "volumeMounts": [
                    {
                        "mountPath": "/var/run/secrets/istio",
                        "name": "istiod-ca-cert"
                    },
                    {
                        "mountPath": "/var/lib/istio/data",
                        "name": "istio-data"
                    },
                    {
                        "mountPath": "/etc/istio/proxy",
                        "name": "istio-envoy"
                    },
                    {
```

```
                        "mountPath": "/etc/istio/pod",
                        "name": "istio-podinfo"
                    },
                    {
                        "mountPath": "/etc/asm/uds/",
                        "name": "asm-hotupgrade-data"
                    }
                ]
            }
        ],
        "initContainers": [
            {
                "args": [
                    "istio-iptables",
                    "-p",
                    "15001",
                    "-z",
                    "15006",
                    "-u",
                    "1337",
                    "-m",
                    "REDIRECT",
                    "-i",
                    "*",
                    "-x",
                    "172.23.0.1/32",
                    "-b",
                    "*",
                    "-d",
                    "15090,15021,15021"
                ],
                "env": [
                    {
                        "name": "DNS_AGENT"
                    }
                ],
                "image": "registry-vpc.cn-zhangjiakou.aliyuncs.com/acs/proxyv2:1.6.8",
                "imagePullPolicy": "IfNotPresent",
                "name": "istio-init",
                "resources": {
                    "limits": {
                        "cpu": "100m",
                        "memory": "50Mi"
                    },
                    "requests": {
                        "cpu": "10m",
                        "memory": "10Mi"
                    }
                },
                "securityContext": {
                    "allowPrivilegeEscalation": false,
                    "capabilities": {
                        "add": [
                            "NET_ADMIN",
                            "NET_RAW"
```

```
                        "NET_RAW"
                    ],
                    "drop": [
                        "ALL"
                    ]
                },
                "privileged": false,
                "readOnlyRootFilesystem": false,
                "runAsGroup": 0,
                "runAsNonRoot": false,
                "runAsUser": 0
            },
            "terminationMessagePath": "/dev/termination-log",
            "terminationMessagePolicy": "File",
            "upgradeStrategy": {}
        }
    ],
    "selector": {
        "matchExpressions": [
            ...
        ]
    },
    "strategy": {
        "type": "RollingUpdate",
        "partition": 0,
        "maxUnavailable": 1
    },
    "volumes": [
        {
            "emptyDir": {},
            "name": "asm-hotupgrade-data"
        },
        {
            "emptyDir": {
                "medium": "Memory"
            },
            "name": "istio-envoy"
        },
        {
            "emptyDir": {},
            "name": "istio-data"
        },
        {
            "downwardAPI": {
                "defaultMode": 420,
                "items": [
                    {
                        "fieldRef": {
                            "apiVersion": "v1",
                            "fieldPath": "metadata.labels"
                        },
                        "path": "labels"
                    },
                    {
                        "fieldRef": {
```

```
                                "apiVersion": "v1",
                                "fieldPath": "metadata.annotations"
                            },
                            "path": "annotations"
                        }
                    ]
                },
                "name": "istio-podinfo"
            },
            {
                "configMap": {
                    "defaultMode": 420,
                    "name": "istio-ca-root-cert"
                },
                "name": "istiod-ca-cert"
            }
        ]
    }
}
```

- Replace `$$$IMAGE$$$` with the URL of the sidecar image.
- Set the `matchExpressions` parameter for the selector. This way, the selector can be used to match the pod where you want to inject sidecars. For more information, visit Labels and Selectors.

URLs of Istio 1.6.x images

- Istio 1.6.x-1 : registry.cn-hangzhou.aliyuncs.com/acs/asm-istio-proxy:feature-1.6.x-faee4bb874d29dabde41481b695718c5b73b6b04-1531
- Istio 1.6.x-2 : registry.cn-hangzhou.aliyuncs.com/acs/asm-istio-proxy:feature-1.6.x-faee4bb874d29dabde41481b695718c5b73b6b04-1546