ALIBABA CLOUD

阿里云

服务网格

数据平面

文档版本：20220209

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.安装Sidecar代理

服务网格ASM支持通过Sidecar代理来增强服务调用之间的网络安全性、可靠性以及可观测性。本文介绍如何安装Sidecar代理。

背景信息

Sidecar所支持的功能可以从应用程序的容器中抽象出来，并在作为同一Pod中的独立容器提供的代理中实现。为了充分利用这些功能，应用程序中的每个服务都需要在其Pod中运行一个Envoy Sidecar代理。Envoy代理拦截到该服务的所有入站和出站HTTP通信，并与服务网格ASM提供的控制平面Pilot组件通信。

步骤一：启用Sidecar注入

默认情况下，对所有命名空间禁用Sidecar自动注入。可以通过更新Pod的Kubernetes配置手动注入Envoy代理，也可以使用基于Webhook的机制自动注入。执行以下命令，启用自动注入：

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

> **说明** 其中，*namespace*表示应用程序服务对应的命名空间，如果没有指定则会使用default命名空间。

步骤二：重新启动Pod

由于在创建Pod时会注入Sidecar，因此必须重新启动正在运行的Pod才能使更改生效。

> **注意** 请在测试环境中反复进行重新启动Pod的验证测试，以确保您的服务可以处理任何潜在的流量中断。

1. 运行以下命令重启Pod。

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

2. 检查Pod是否都注入了Sidecar。每个工作负载都有两个容器：主容器和Sidecar代理容器。

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

2.多种方式灵活开启自动注入

为了充分利用服务网格的所有特性，服务网格中ACK集群的应用Pod必须包含一个Sidecar代理。除了手动注入方式外，通常建议启用自动注入的方式来简化部署。本文介绍如何使用多种方式开启自动注入。

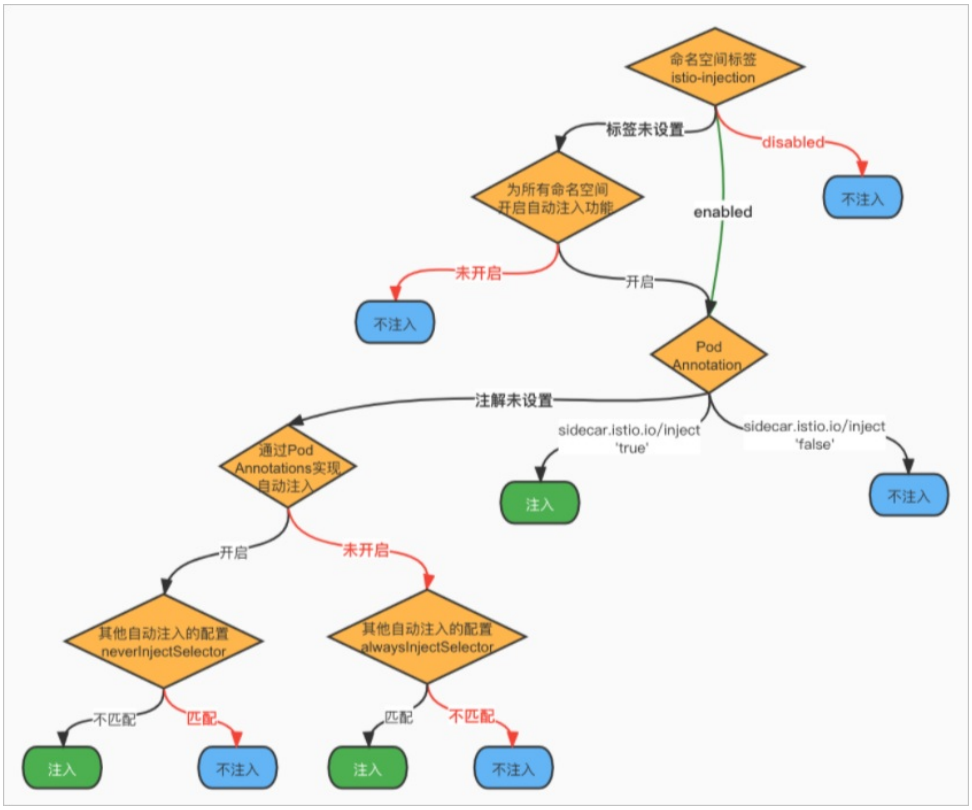
背景信息

ASM默认提供了一个Webhook控制器，可以将Sidecar代理自动添加到可用的Pod中。Sidecar代理的详细介绍请参见设置 Sidecar。

> **说明** 请确保ASM实例的Istio为v1.6.8.17及以上版本。

Sidecar注入优先级

Sidecar注入的设置存在优先级关系，逻辑决策如下图所示：



开启自动注入

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格详情页面左侧导航栏单击**Sidecar注入管理**，在右侧页面设置自动注入功能，然后单击**更新设置**。

   开启自动注入功能有以下选项：

操作项	说明
只选中**为所有命名空间开启自动注入功能**	选中此项后，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 选中此项后，在不具有 `istio-injection:disabled` 标签的命名空间下，在Pod添加 `sidecar.istio.io/inject="true"` 注解，则该Pod会自动注入Sidecar代理。 ○ 关闭自动注入 ■ 选择此项后，为命名空间添加 `istio-injection:disabled` 标签，则该命名空间不会自动注入Sidecar代理。 ■ 选择此项后，命名空间下的Pod没有 `sidecar.istio.io/inject="true"` 注解，则该Pod不会自动注入Sidecar代理。
选中**为所有命名空间开启自动注入功能**和**其他自动注入的配置**	选中此项后，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 选中此项后，在其他自动注入的配置的文本框中设置 `alwaysInjectSelector` 字段，在不具有 `istio-injection:disabled` 标签的命名空间下，在Pod添加 `alwaysInjectSelector` 字段中的 `key` 标签。则该Pod会自动注入Sidecar代理。 ○ 关闭自动注入 ■ 选择此项后，为命名空间添加 `istio-injection:disabled` 标签，则该命名空间不会自动注入。 ■ 选择此项后，命名空间下的Pod没有 `sidecar.istio.io/inject="true"` 注解，则该Pod不会自动注入Sidecar代理。

操作项	说明
只选中**通过Pod Annotations实现自动注入**	选中此项后，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 选中此项后，为命名空间添加 `istio-injection:enabled` 标签，则该命名空间会自动注入Sidecar代理。 ○ 关闭自动注入 ■ 选择此项后，不具有 `istio-injection:enabled` 标签的命名空间不会自动注入Sidecar代理。 ■ 选择此项后，Pod有 `sidecar.istio.io/inject="false"` 注解，则该Pod不会自动注入Sidecar代理。
选中**通过Pod Annotations实现自动注入和其他自动注入的配置**	选中此项后，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 选中此项后，为命名空间添加 `istio-injection:enabled` 标签，则该命名空间会自动注入Sidecar代理。 ○ 关闭自动注入 选择此项后，不具有 `istio-injection:enabled` 标签的命名空间不会自动注入Sidecar代理。 ○ 在具有 `istio-injection:enabled` 标签的命名空间下，在Pod关闭自动注入功能 选中此项后，在其他自动注入的配置的文本框中设置 `neverInjectSelector` 字段，在Pod添加 `neverInjectSelector` 字段中的 `key` 标签后。即使Pod位于具有 `istio-injection:enabled` 标签的命名空间下，具有 `key` 标签的Pod不会自动注入Sidecar代理。

操作项	说明
选中**为所有命名空间开启自动注入功能**和**通过Pod Annotations实现自动注入**	选中此项后，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 选中此项后，只要命名空间没有 `istio-injection:disabled` 标签，则该命名空间会自动注入Sidecar代理。 ○ 关闭自动注入 选择此项后，为命名空间添加 `istio-injection:disabled` 标签，则该命名空间不会自动注入Sidecar代理。
选中**为所有命名空间开启自动注入功能**、**通过Pod Annotations实现自动注入**和**其他自动注入的配置**	选中此项后，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 选中此项后，只要命名空间没有 `istio-injection:disabled` 标签，则该命名空间会自动注入Sidecar代理。 ○ 关闭自动注入 选择此项后，为命名空间添加 `istio-injection:disabled` 标签，则该命名空间不会自动注入Sidecar代理。 ○ 在没有 `istio-injection:disabled` 的标签命名空间下，关闭自动注入 选中此项后，在其他自动注入的配置的文本框中设置 `neverInjectSelector` 字段，在Pod添加 `neverInjectSelector` 字段中的 `key` 标签。即使Pod位于没有 `istio-injection:disabled` 的标签命名空间下，仍然不会自动注入Sidecar代理。

操作项	说明
只选中**其他自动注入配置**	选中此项后，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 选中此项后，为命名空间添加 `istio-injection:enabled` 标签，在其他自动注入的配置的文本框中设置 `alwaysInjectSelector` 字段，在Pod添加 `alwaysInjectSelector` 字段中的 `key` 标签，该Pod会自动注入Sidecar代理。 ○ 关闭自动注入 ■ 选中此项后，不具有 `istio-injection:enabled` 标签的命名空间不会自动注入Sidecar代理。 ■ 选中此项后，具有 `istio-injection:enabled` 标签的命名空间，但是没有 `sidecar.istio.io/inject="true"` 注解的Pod不会自动注入Sidecar代理。
不选择任何项	不选择任何项，您可以根据实际需求开启和关闭自动注入功能： ○ 开启自动注入 为命名空间添加 `istio-injection:enabled` 标签，然后为Pod添加 `sidecar.istio.io/inject="true"` 注解，该Pod会自动注入Sidecar代理。 ○ 关闭自动注入 ■ 不具有 `istio-injection:enabled` 标签的命名空间不会自动注入Sidecar代理。 ■ 具有 `istio-injection:enabled` 标签的命名空间，但是没有 `sidecar.istio.io/inject="true"` 注解的Pod不会自动注入Sidecar代理。

除了开启自动注入功能外，您还可以设置代理资源。

参数	说明
Sidecar代理注入服务资源设置	ASM默认为每个数据面集群提供了一个Webhook控制器，可以将Sidecar代理自动添加到可用的Pod中。此处的资源设置是用于限制Webhook控制器资源的大小。
注入的Istio代理资源设置	istio-proxy为应用提供代理服务，自动注入后，和业务容器运行在同一个Pod中。此处的资源设置是用于限制istio-proxy资源的大小。

其他主动注入的配置

您可以在其他自动注入中配置标签，匹配Pod上的标签，从而控制Pod是否需要中注入Sidecar。

- 使用一个 `alwaysInjectSelector` 的字段，总是将Sidecar注入匹配标签选择器的Pod中，而忽略全局策略。

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

- 使用一个 `neverInjectSelector` 的字段，总是不会将Sidecar注入匹配标签选择器的Pod中，而忽略全局策略。

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

- 其他配置参数。

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

- replicaCount：部署的副本数。
- injectedAnnotations：注入的其他注解。
- nodeSelector：注入部署的节点，本文设置为 `linux` ，表示注入到带有标签 `linux` 的Node节点上。

场景示例一：某命名空间开启自动注入的情况下，该命名空间下某些Pod关闭自动注入

如果您希望命名空间能够开启自动注入功能，但是该命名空间下某些Pod关闭自动注入。您可以进行以下操作：

使用其他自动注入的配置实现命名空间开启自动注入情况下，该命名空间下某些Pod关闭自动注入

1. 在ASM开启自动注入。
   i. 登录ASM控制台。
   ii. 在左侧导航栏，选择**服务网格 > 网格管理**。
   iii. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
   iv. 在网格详情页面左侧导航栏单击**Sidecar注入管理**。
   v. 在**Sidecar注入管理**页面**开启自动注入功能**区域选中**通过Pod Annotations实现自动注入和其他自动注入的配置**，在文本框中添加以下内容，然后单击**更新设置**。

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

2. 创建命名空间。
   i. 在网格详情页面左侧导航栏单击**命名空间**，在右侧页面单击**新建**。
   ii. 在**新建命名空间**面板设置命名空间名称，单击**添加**，设置名称为istio-injection，值为enabled，然后单击**确定**。本文设置命名空间名称为test1。
3. 创建应用。

i. 在ACK集群的test1命名空间下创建应用。具体操作，请参见部署应用到ASM实例，本文以部署details服务为例。

ii. 验证应用的Pod是否注入Sidecar代理。

a. 登录容器服务管理控制台。

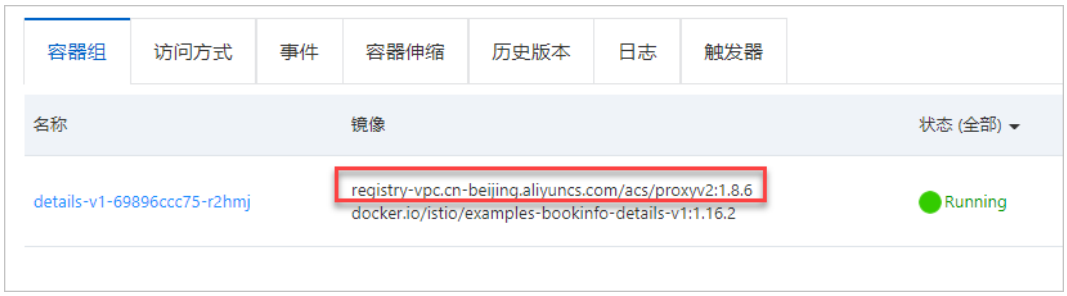
b. 在控制台左侧导航栏中，单击**集群**。

c. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。

d. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。

e. 在**无状态**页面顶部设置**命名空间**为test1，单击details应用名称。

在**容器组**页签下可以看到Pod包含Proxy镜像，说明自动注入Sidecar代理成功。



4. 在Pod中添加标签，使Pod关闭自动注入。

i. 登录容器服务管理控制台。

ii. 在控制台左侧导航栏中，单击**集群**。

iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。

iv. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。

v. 在**无状态**页面设置**命名空间**为test1，选择details服务右侧**操作**下的**更多 > 查看Yaml**。

vi. 在 `labels` 参数下添加 `key` 为 `notinjectapp` 的标签， `value` 可以自定义，然后单击**更新**。

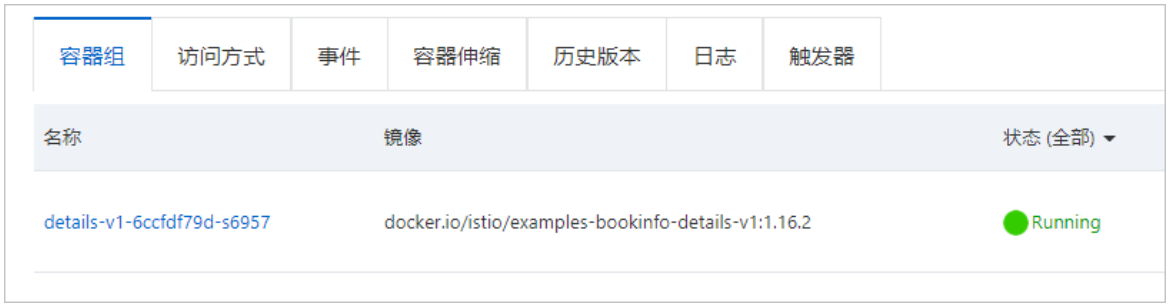
```
135   template:
136     metadata:
137       annotations:
138         redeploy-timestamp: '1629688001960'
139       labels:
140         app: details
141         notinjectapp: details
142         version: v1
143         notinjectapp: details
```

vii. 在**无状态**页面设置**命名空间**为test1，选择details服务右侧**操作**下的**更多 > 重新部署**。

viii. 在弹出对话框单击**确定**。

5. 验证test1命名空间开启自动注入的情况下，details服务的Pod不进行自动注入是否成功。

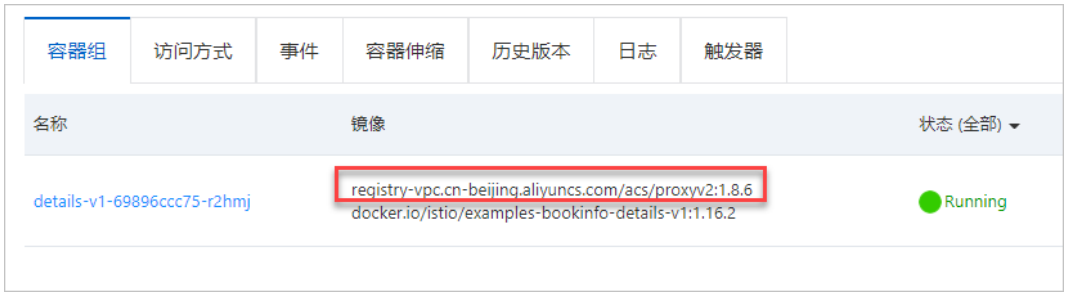
在**无状态**页面单击details服务的名称，在**容器组**页签下可以看到Pod不包含Proxy镜像，说明在test1命名空间开启自动注入的情况下，details服务的Pod不进行自动注入成功。



使用Annotations方式实现命名空间开启自动注入情况下，该命名空间下某些Pod关闭自动注入

1. 在ASM开启自动注入。
   i. 登录ASM控制台。
   ii. 在左侧导航栏，选择**服务网格 > 网格管理**。
   iii. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
   iv. 在网格详情页面左侧导航栏单击**Sidecar注入管理**。
   v. 在**Sidecar注入管理**页面**开启自动注入功能**区域选中**通过Pod Annotations实现自动注入**，然后单击**更新设置**。
2. 创建命名空间。
   i. 在网格详情页面左侧导航栏单击**命名空间**，在右侧页面单击**新建**。
   ii. 在**新建命名空间**面板设置命名空间名称，单击**添加**，设置名称为istio-injection，值为enabled，然后单击**确定**。本文设置命名空间名称为test1。
3. 创建应用。
   i. 在ACK集群的test1命名空间下创建应用。具体操作，请参见部署应用到ASM实例，本文以部署details服务为例。
   ii. 验证应用的Pod是否注入Sidecar 代理。
      a. 登录容器服务管理控制台。
      b. 在控制台左侧导航栏中，单击**集群**。
      c. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
      d. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。
      e. 在**无状态**页面设置**命名空间**为test1，单击details应用名称。

      在**容器组**页签下可以看到Pod包含Proxy镜像，说明自动注入Sidecar代理成功。



4. 在Pod中添加注解，使Pod关闭自动注入。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。

iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。

iv. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。

v. 在**无状态**页面设置**命名空间**为test1，选择details服务右侧**操作**下的**更多 > 查看Yaml**。

vi. 在 `annotations` 参数下添加 `sidecar.istio.io/inject: "false"` 的注解，然后单击**更新**。

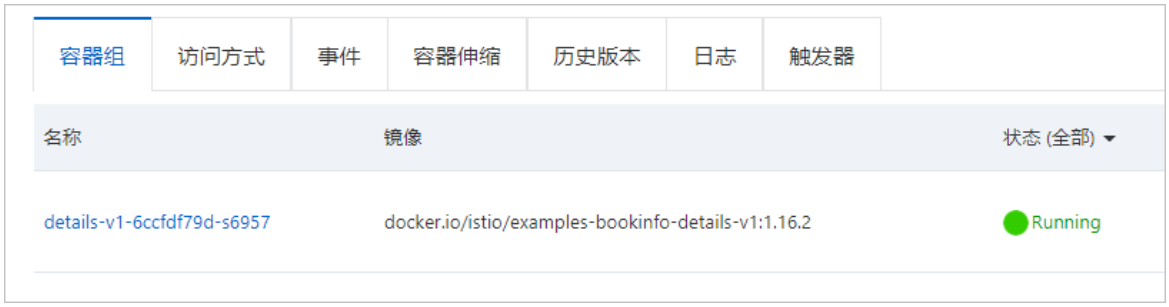
```
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: details
      version: v1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      annotations:
        redeploy-timestamp: '1629711538748'
        sidecar.istio.io/inject: "false"
```

vii. 在**无状态**页面设置**命名空间**为test，选择details服务右侧**操作**下的**更多 > 重新部署**。

viii. 在弹出对话框单击**确定**。

5. 验证test1命名空间开启自动注入的情况下，details服务的Pod不进行自动注入是否成功。

   在**无状态**页面单击details服务的名称，在**容器组**页签下可以看到Pod不包含Proxy镜像，说明在test1命名空间开启自动注入的情况下，details服务的Pod不进行自动注入成功。



场景示例二：单独为Pod设置自动注入策略

如果您不想以命名空间为维度批量设置自动注入策略，您也可以单独为Pod设置自动注入策略。

1. 为命名空间开启自动注入。

   i. 登录ASM控制台。

   ii. 在左侧导航栏，选择**服务网格 > 网格管理**。

   iii. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。

   iv. 在网格详情页面左侧导航栏单击**命名空间**。

v. 单击目标命名空间右侧**自动注入**列下的**启用Sidecar自动注入**，在弹出的对话框单击**确定**。本文以test2命名空间为例。

2. 在ACK集群的test2命名空间下创建应用。具体操作，请参见部署应用到ASM实例，本文以部署reviews服务为例。
3. 为Pod添加注解，使Pod开启自动注入。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。
   iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
   iv. 在集群管理页左侧导航栏中，选择**工作负载 > 无状态**。
   v. 在**无状态**页面设置**命名空间**为test2，选择reviews服务右侧**操作**下的**更多 > 查看Yaml**。
   vi. 在 `annotations` 参数下添加 `sidecar.istio.io/inject: "true"` 注解，然后单击**更新**。

```yaml
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: reviews
      version: v1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      annotations:
        redeploy-timestamp: '1629707593246'
        sidecar.istio.io/inject: "true"
```

   vii. 在**无状态**页面设置**命名空间**为test2，选择reviews服务右侧**操作**下的**更多 > 重新部署**。
   viii. 在弹出对话框单击**确定**。
4. 验证为Pod设置自动注入策略是否成功。

   在**无状态**页面单击reviews服务的名称，在**容器组**页签下可以看到Pod中包含Proxy镜像，说明为Pod设置自动注入策略成功。

3.升级Sidecar代理

控制平面升级后，已经启用Istio的应用程序仍将使用旧版本的Sidecar代理，因此需要进行升级。本文介绍如何通过自动注入Sidecar和手动注入Sidecar这两种方式升级Sidecar代理。

前提条件

kubectl连接至ACK集群，请参见通过kubectl工具连接集群。

背景信息

Sidecar升级的操作是在数据面的Kubernetes集群中进行的，对应的kubeconfig应当是面向数据面的Kubernetes集群，而不是网格实例对应的kubeconfig。因此kubeconfig应当从容器服务控制台获取，而不是从服务网格控制台获取。

自动注入Sidecar

如果使用自动注入Sidecar的方式，可以通过对所有Pod进行滚动升级来升级Sidecar代理，这样新版本的Sidecar将被自动重新注入。建议使用该方式简化升级步骤。

可以使用以下Shell脚本来触发滚动更新。

```
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."
```

例如，将以上脚本存为文件*upgradeproxy.sh*，并赋予可执行权限，例如在Linux命令行下执行 `chmod +x upgradeproxy.sh` 。

该命令需要指定命名空间名称作为参数，例如更新default命名空间下的Pod，则需要执行 `./upgradeproxy.sh default` 。

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

手动注入Sidecar

如果没有使用自动注入Sidecar的方式，则需要执行以下相应的命令手动升级Sidecar。

按照之前手工注入的方式，重新生成一个新的部署YAML文件，并重新执行**kubectl apply**命令。

```
kubectl apply -f <(istioctl kube-inject -f <未注入过Sidecar代理配置的原始应用YAML文件>)
```

4.为Envoy编写WASM Filter并部署到ASM中

WebAssembly（WASM）是一种编程语言，ASM提供了对WASM技术的支持，可以把扩展的WASM Filter通过ASM部署到数据面集群中相应的Envoy代理中。通过这种过滤器扩展机制，可以轻松扩展Envoy的功能并将其在服务网格中的应用推向了新的高度。本文介绍了WASM Filter以及如何为Envoy编写WASM Filter并部署到ASM中。

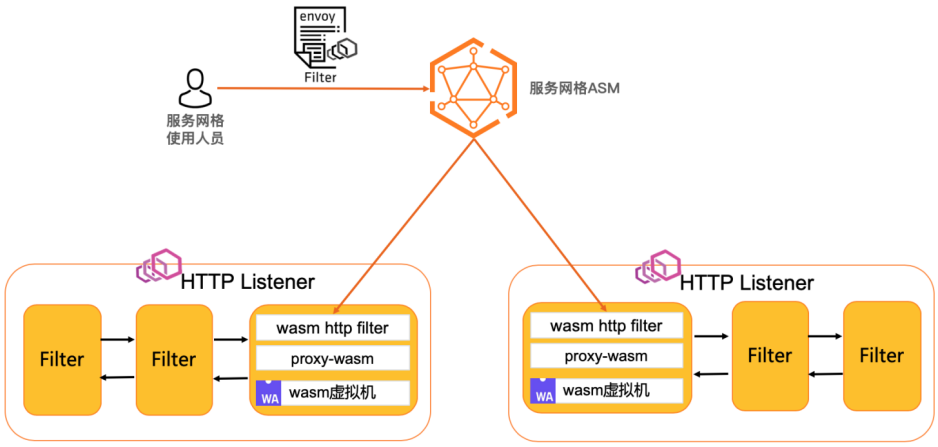
背景信息

Envoy是一个高性能、可编程的L3、L4和L7代理，作为ASM数据面的代理使用。Envoy的连接和流量处理的核心是网络过滤器（Network Filter），该过滤器一旦融合进过滤器链（Filter Chain），就可以实现访问控制、数据或协议转换、数据增强、审计等高级功能。通过添加新的过滤器，可以用来扩展Envoy的已有功能集。当前有两种方法可以添加新的过滤器：

- 静态预编译：将其他过滤器集成到Envoy的源代码中，并编译新的Envoy版本。这种方法的缺点是您需要维护Envoy版本，并不断使其与官方发行版保持同步。此外，由于Envoy是用C++实现的，因此新开发的过滤器也必须用C++实现。
- 动态运行时加载：在运行时将新的过滤器动态加载到Envoy代理中。

显而易见，第二种方法极大地简化了扩展Envoy的过程。这种解决方案依赖于一种称之为WebAssembly（WASM）的新技术，它是一种有效的可移植二进制指令格式，提供了可嵌入和隔离的执行环境。

ASM提供了对WASM技术的支持，如下图所示。



为什么要使用WASM Filter?

使用WASM实现过滤器的扩展，有如下优势：

- 敏捷性：过滤器可以动态加载到正在运行的Envoy进程中，而无需停止或重新编译。
- 可维护性：不必更改Envoy自身基础代码库即可扩展其功能。
- 多样性：可以将流行的编程语言（例如C/C++和Rust）编译为WASM，因此开发人员可以选择实现过滤器的编程语言。
- 可靠性和隔离性：过滤器会被部署到VM沙箱中，因此与Envoy进程本身是隔离的；即使当WASM Filter出现问题导致崩溃时，它也不会影响Envoy进程。
- 安全性：过滤器通过预定义API与Envoy代理进行通信，因此它们可以访问并只能修改有限数量的连接或请求属性。

当前WASM实现过滤器的扩展，也需要考虑以下缺点是否可以容忍：

- 性能约为C++编写的原生静态编译的Filter的70%。
- 由于需要启动一个或多个WASM虚拟机，因此会消耗一定的内存使用量。

使用Proxy-WASM SDK构建过滤器

Envoy Proxy在基于堆栈的虚拟机中运行WASM过滤器，因此过滤器的内存与主机环境是隔离的。Envoy代理与WASM过滤器之间的所有交互都是通过Envoy Proxy-WASM SDK提供的功能实现的。Envoy Proxy-WASM SDK提供了多种编程语言的实现，包括C++、Rust、AssemblyScript以及处于实验中的Golang等。此外，社区也在推动相应的WASMfor Proxies（Proxy-Wasm）应用二进制接口ABI规范，详情请参见spec。

1. 构建WASM Filter的最简单方法是使用Docker，使用C++ Envoy Proxy-WASM SDK创建一个Docker镜像。详情请参见Docker。
2. 创建一个项目并使用上述Docker镜像进行构建。详情请参见Creating a project for use with the Docker build image。
3. 编辑开发此项目。详情请参见WebAssembly for Proxies (C++ SDK)。
4. 切换到项目根目录，执行如下命令构建WASM。

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

在ASM中部署启用WASM Filter

1. 创建一个configmap，用于保存WASM过滤器的二进制文件内容。例如，在命名空间default下，创建一个名称为wasm-example-filter的configmap，并将WASM过滤器的二进制文件example-filter.wasm保存到该configmap中。

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

2. 使用以下两个annotation将WASM过滤器的二进制文件注入到应用程序对应的Kubernetes服务中。

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

3. 执行以下命令更新productpage-v1。

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

4. 执行以下命令更新details-v1。

```
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\":\"/var/local/lib/wasm-filters\",\"name\":\"wasmfilters-dir\"}]"}}}}}'
```

5. 在istio-proxy容器中的路径*/var/local/lib/wasm-filters*下，找到WASM过滤器的二进制文件。

```
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. 执行以下命令，使WASM过滤器在处理针对应用服务 `productpage` 的流量时，能够以DEBUG日志级别记录。

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

7. 执行以下命令，使WASM过滤器在处理针对应用服务 `details-v1` 的流量时，能够以DEBUG日志级别记录。

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

8. 执行以下命令，将WASM过滤器插入到应用服务 `productpage` 的HTTP级别过滤器链中。

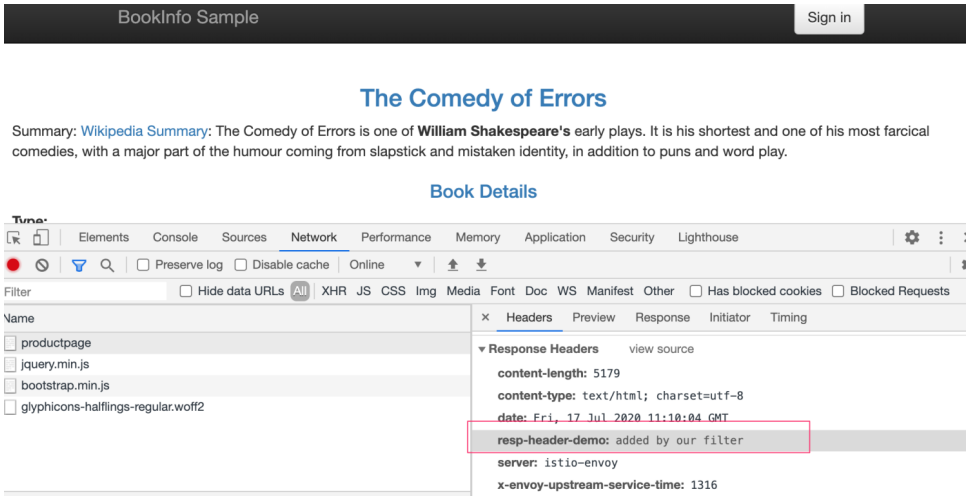
```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: productpage-v1-examplefilter
  labels:
    asm-system: 'true'
    provider: asm
spec:
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      context: SIDECAR_INBOUND
      proxy:
        proxyVersion: '^1\.*.*'
      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. 执行以下命令，将WASM过滤器插入到应用服务details的HTTP级别过滤器链中。

```
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
```

验证结果

1. 通过在浏览器中访问入口网关的地址，将一些流量发送到productpage服务上，在页面响应中，可以看到过滤器的头添加到响应头中，如下图所示。



2. 执行以下命令，将一些流量发送到details服务上。在响应中，可以看到过滤器的头添加到响应头中。

```
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.使用ORAS简化基于WASM的服务网格扩展功能

WebAssembly（WASM）是一种有效的可移植二进制指令格式，可以用于扩展ASM数据平面的功能。然而在服务网格内构建、部署和运行Wasm Filter较为复杂。本文介绍如何使用ORAS简化基于WASM的服务网格扩展功能。

前提条件

- 已创建ASM实例，并添加集群到ASM实例。具体操作，请参见创建ASM实例和添加集群到ASM实例。

  > 说明　ASM实例必须为v1.8.3.17-g1399628c-aliyun或者以上版本。

- 使用kubectl连接到ASM实例。具体操作，请参见通过kubectl连接ASM实例。
- 已部署应用到ASM实例。具体操作，请参见部署应用到ASM实例。
- 已创建Istio虚拟服务和入口网关。具体操作，请参见定义Istio资源。
- 已创建并编译一个Wasm Filter二进制文件。本文以将过滤器的头添加到响应头的Wasm Filter为例。

背景信息

ASM提供了对WASM技术的支持，可以把扩展的Wasm Filter通过ASM部署到数据面集群中相应的Envoy代理中，从而扩展数据平面的功能。ORAS是基于OCI Artifacts规范的OCI注册表存储，可以显著简化OCI注册库中内容的存储过程。基于WASM扩展ASM数据平面功能时，可以使用ORAS对功能扩展过程进行简化。

上传Wasm Filter

使用ORAS CLI工具上传Wasm Filter到镜像仓库，以阿里云容器镜像服务企业版ACR EE为例。

1. 创建容器镜像仓库并获取登录镜像仓库的账号。具体操作，请参见使用企业版实例推送和拉取镜像。
2. 执行以下命令，登录镜像仓库。

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

3. 执行以下命令，将Wasm Filter推送到镜像仓库。

   ```
   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.wasm:application/vnd.module.wasm.content.layer.v1+wasm
   ```

4. 在容器镜像仓库查看推送的Wasm Filter。
   i. 登录容器镜像服务控制台。
   ii. 在顶部菜单栏，选择所需地域。
   iii. 在左侧导航栏，选择**实例列表**。
   iv. 在**实例列表**页面单击目标企业版实例。
   v. 在企业版实例管理页面左侧导航栏选择**仓库管理 > 镜像仓库**。
   vi. 在**镜像仓库**页面单击目标镜像仓库的名称。
   vii. 在镜像仓库管理页面左侧导航栏单击**镜像版本**，在**镜像版本**页面可以看到上传的Wasm Filter。

启用使用WASM的功能

通过控制台启用使用WASM的功能

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格管理详情页面左上角单击**功能设置**。
5. 在**功能设置更新**面板**数据面扩展**区域选中**启用基于WebAssembly的服务网格扩展**，然后单击**确定**。

> **说明** 在**功能设置更新**面板**数据面扩展**区域去掉选中**启用基于WebAssembly的服务网格扩展**，可以关闭使用WASM的功能。

通过命令行启用使用WASM的功能

通过Aliyun CLI可以启用服务网格中的使用WASM的功能。执行以下命令，启用使用WASM的功能。

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

您也可以执行以下命令，关闭使用WASM的功能。

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

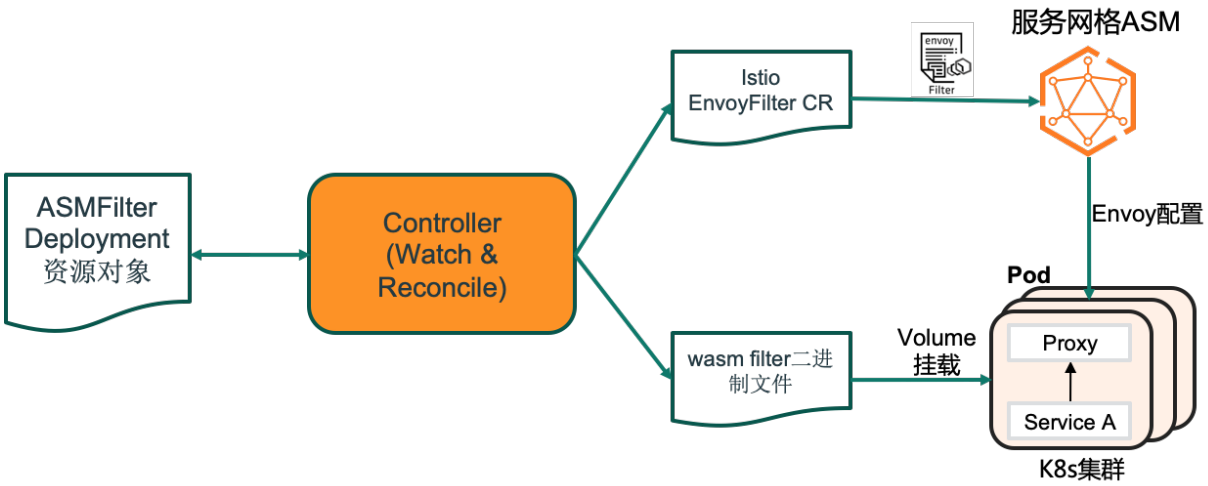
在ASM中使用WASM

服务网格ASM产品提供了一个ASMFilterDeployment资源以及相关的Controller组件。这个Controller组件会监听ASMFilterDeployment资源对象的情况，并进行以下操作：

- 创建用于控制面的Istio EnvoyFilter Custom Resource，并推送到对应的ASM控制面Istio中。
- 从镜像仓库中拉取对应的Wasm Filter镜像， 并挂载到对应的Workload Pod中。

使用WASM流程

1. 启用使用WASM的功能，启用后，自动部署一个DaemonSet类型的asmwasm-controller到ACK集群中。
2. asmwasm-controller监听一个ConfigMap，该configmap存放了拉取的Wasm Filter的镜像仓库地址。
3. 如果需要授权认证，该asmwasm-controller会根据定义的pullSecret值获得相应的Secret值。
4. asmwasm-controller会调用ORAS API从注册库中动态拉取Wasm Filter。
5. asmwasm-controller使用HostPath方式挂载Volume， 所以拉取的Wasm Filter会落盘到对应的节点上。



操作步骤

1. 执行以下命令，启用使用WASM的功能。

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

2. 在ACK集群中创建用来访问镜像仓库的Secret。

   关于Secret的详细信息，请参见Secret。

   i. 使用以下内容，创建名为*myconfig.json*的文件。

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

   - `**********.cn-hangzhou.cr.aliyuncs.com`：镜像仓库地址。
   - `username`：镜像仓库用户名。
   - `password`：镜像仓库密码。

   ii. 执行以下命令，创建Sercet。

> **说明** Secret名字必须为asmwasm-cache，命名空间为istio-system。

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

3. 部署ASMFilterDeployment资源。

i. 使用以下内容，创建一个名为*filter.yaml*的文件。

```
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'
```

- `workload` 下的参数解释：
  a. `kind`：目标工作负载的类型。
  b. `labels`：筛选的条件。
- `filter` 下的参数解释：
  a. `patchContext`：生效的上下文阶段。
  b. `parameters`：运行Wasm Filter所需的配置参数。
  c. `image`：Wasm Fitler对应的镜像仓库地址。
  d. `rootID`：Wasm Filter扩展插件对应的RootID。
  e. `id`：该Wasm Filter的唯一ID。

ii. 执行以下命令，部署ASMFilterDeployment资源。

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

ASMFilterDeployment部署后，会自动生成EnvoyFilter。其中match片段中定义了envoy.router，patch片段中定义了INSERT_BEFORE。

- match片段

- patch片段

4. 查看Workload。

```
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"}]'
```

可以看到Wasm Filter文件以HostPath方式挂载到Proxy容器中。

验证Wasm Filter是否生效

执行以下命令，登录到productpage Pod的istio-proxy容器中并请求details服务。

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

预期输出：

```
*   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
```

返回结果中，可以看到过滤器的头添加到响应头中。

6.启用服务网格中的DNS代理

从Istio 1.8开始，Sidecar上的Istio代理具备缓存DNS代理的功能。当服务网格收到来自应用程序的DNS查询时，Istio代理将进行透明地拦截并提供解析能力。本文介绍如何启用服务网格中的DNS代理。

前提条件

- 创建ASM实例

  > **说明** ASM实例必须为v1.8.3.17-g1399628c-aliyun或者以上版本。

- 添加集群到ASM实例

背景信息

基于Kubernetes服务和定义的服务条目，为应用程序可以访问的所有服务推送主机名到IP地址的映射。来自应用程序的DNS查询被Istio代理透明地拦截并提供解析能力：

- 如果查询的是服务网格中的服务，Istio代理将直接对应用程序返回查询响应。
- 如果查询的是服务网格之外的服务，Istio代理将查询请求转发到 `/etc/resolv.conf` 中定义的上游名称服务器。

启用服务网格中的DNS代理

通过控制台启用服务网格中的DNS代理

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格管理详情页面左上角单击**功能设置**。
5. 在**功能设置更新**面板**流量管理**区域选中**启用DNS代理功能**，然后单击**确定**。

   > **说明** 在**功能设置更新**面板**流量管理**区域去掉选中**启用DNS代理功能**，可以关闭DNS代理功能。

通过命令行启用服务网格中的DNS代理

通过Aliyun CLI可以开启服务网格中的DNS代理。执行以下命令，开启DNS代理功能。

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

您也可以执行以下命令，关闭DNS代理功能。

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

验证DNS代理功能

1. 在服务网格中创建服务条目。

   使用ServiceEntry将https://aliyun.com添加到服务网格内部维护的服务注册表中。

   i. 登录ASM控制台。

ii. 在左侧导航栏，选择**服务网格 > 网格管理**。

iii. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。

iv. 在网格详情页面左侧导航栏选择**流量管理 > 服务条目**，然后在右侧页面单击**新建**。

v. 在**新建**面板，选择**命名空间**，在文本框输入ServiceEntry的配置信息，单击**确定**。

```
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. 在ACK集群中部署Sleep应用。

i. 使用以下内容，创建名为*sleep.yaml*的文件。

```
##################################################################################################
###############
# 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. 执行以下命令，部署Sleep应用。

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

3. 执行以下命令，登录到Sleep容器中并执行curl命令访问https://aliyun.com。

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

预期输出：

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

可以看到返回的地址是240.240.**.**，该地址是服务网格自动分配的虚拟IP，而不是真实的公开的IP地址。因为服务网格中使用iptables劫持了对kube-dns的请求，并将请求路由到了Pod中运行的Sidecar Proxy。当应用程序pod将aliyun.com解析为虚拟IP并发出请求时，虚拟IP将被替换为Sidecar Proxy中解析的实际公共IP地址。

总之，由于创建了hosts地址为aliyun.com的ServiceEntry，因此应用程序Pod将在Istio DNS上查询aliyun.com并获得虚拟IP，并且当通过Sidecar Proxy发送请求时，该虚拟IP将被转换为解析后的实际公共IP地址。

7.ASM数据平面热升级Beta

在ASM管控的集群中，数据平面（Sidecar）代理了应用的全部流量。对数据平面的升级需要重启Sidecar容器，而重启会导致部分请求失败、应用服务中断。ASM提供了对数据平面热升级的能力，在升级数据平面时不会中断服务，使数据平面在应用无感知的情况下完成升级。本文将以ASM的Istio1.6.x版本为演示环境，部署一个Nginx应用，并使用HTTP压力测试工具go-stress-testing对Nginx进行持续访问，在此过程中对数据平面进行热升级。

前提条件

- 创建一个1.6.x版本的ASM实例。更多信息，请参见创建ASM实例。
- 添加集群到ASM示例。更多信息，请参见添加集群到ASM实例。
- 添加入口网关服务。更多信息， 请参见添加入口网关服务。
- 确保ASM默认命名空间未启用自动注入，若已启用，则需要关闭自动注入。更多信息，请参见安装Sidecar代理。

注意事项

ASM数据平面热升级需要借助OpenKruise SidecarSet（简称SidecarSet）完成Sidecar容器替换，所以对于潜在存在热升级需求的Deployment，需要在部署时就使用SidecarSet对Deployment的Pod进行Sidecar注入，以便进行热升级操作。可以使用以下两种方式进行Sidecar注入：

> **说明** 建议您在部署应用时完成Sidecar注入。如果您的应用已启用默认注入，您可以改变注入方式并重建Pod，但是会存在Pod短暂不可用带来的业务风险。

- 将有热升级需求的Deployment或Pod部署于独立的Namespace。

  将有热升级需求的Deployment或Pod部署于独立的Namespace，这样就可以对其他Namespace仍旧启用默认注入方案，对该Namespace则使用SidecarSet注入。

- 禁止特定Pod的默认注入，对这些Pod使用SidecarSet注入。

  如果Pod的命名空间已开启默认自动注入，则可以通过Per-Pod-Annotation实现对该Pod禁用默认注入，再使用SidecarSet的匹配策略匹配该Pod进行注入。

步骤一：在数据面集群部署OpenKruise

ASM当前不会在您的数据面集群安装OpenKruise，您需要手动使用Helm安装OpenKruise。

1. 安装阿里云Helm插件。更多信息，请参见推送和拉取Helm Chart。
2. 添加OpenKruise的Helm仓库。

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

3. 在集群中安装OpenKruise。

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

步骤二：部署ConfigMap

SidecarSet配置中涉及数据面集群ID，您可以通过部署ConfigMap避免在每个SidecarSet中重复配置。

1. 创建*configmap.yaml*。

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

将 `$$$CLUSTER-ID$$$` 替换为您的数据面集群ID。

2. 部署ConfigMap。

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

步骤三：部署SidecarSet

每个应用的注入配置中存在一些特定的字段无法统一配置，您需要为每一个Deployment部署独立的SidecarSet来定义注入配置。

1. 创建*nginx-sidecarset.json*。

   本例中已根据模板文件修改为适用于本例的SidecarSet。关于如何定制SidecarSet请参见相关信息。

```
{
    "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": [
```

```
            "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"
                    ],
                    "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. 将nginx-sidecarset.json应用至数据面集群。

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

步骤四：部署Nginx应用

1. 部署Nginx应用。
    i. 创建*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. 部署Nginx应用。

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

2. 在Istio Ingress Gateway暴露Nginx服务端口以及创建路由规则。

i. 创建*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. 使*nginx-gateway.yaml*生效。

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

3. 验证Nginx应用是否部署成功。

i. 检查Pod是否正常启动。

```
kubectl get pod
```

预期输出：

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

返回结果中 `STATUS` 显示 `Running` ，说明Pod正常启动。

ii. 访问入口网关地址的8080端口，检查Nginx是否可以正常服务。

返回如下页面，说明Nginx已成功部署。

步骤五：使用go-stress-testing启动对Nginx的访问

go-stress-testing是一个使用Go语言编写并且支持多平台的HTTP压力测试工具，本例中使用该工具对Nginx应用持续发起访问，在访问持续期间进行热升级，该工具会统计请求的成功或失败数量。

1. 下载go-stress-testing。关于go-stress-testing的下载地址请参见go-stress-testing。
2. 启动对Nginx的访问。

   将启动4个并发对服务器进行访问，每个并发累计发出10万个请求。

   ```
   go-stress-testing-mac -c 4 -n 100000 -u http://入口网关地址:8080
   ```

   启动后，可以看到命令实行时输出的返回码统计信息。

耗时	并发数	成功数	失败数	qps	最长耗时	最短耗时	平均耗时	错误码
1s	4	200	0	201.52	37.17	15.77	4.96	200:200
2s	4	414	0	208.70	40.62	15.77	4.79	200:414
3s	4	619	0	207.50	40.62	15.46	4.82	200:619
4s	4	826	0	207.91	40.62	15.46	4.81	200:826
5s	4	1040	0	209.22	40.62	15.46	4.78	200:1040
6s	4	1237	0	207.38	41.70	15.46	4.82	200:1237

^C

步骤六：对数据平面进行热升级

1. 编辑SidecarSet。

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

2. 将Sidecar的image字段替换为新的SidecarSet版本Image地址，然后保存并退出。

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

```
        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-faee4bb874d29dabde41481b695718c5b73b6b04-1546
  imagePullPolicy: IfNotPresent
  lifecycle:
    postStart:
      exec:
        command:
```

3. 验证数据平面热升级时，是否会中断服务。

i. 查看热升级状态。

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

预期输出：

```
    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
```

当Pod中的Container变为asm-istio-proxy-empty，asm-isitio-proxy，nginx三个Image时，说明该Pod热升级结束。

ii. 热升级结束后，查看步骤五：使用go-stress-testing启动对Nginx的访问中go-stress-testing-mac的输出结果，从输出中可以看到所有请求的返回值均为200，没有请求在升级过程中失败。

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

相关信息

定制SidecarSet

如果您希望定制自己的SidecarSet注入配置，请使用对应版本的模版文件，以下以Istio 1.6.x为例。您需要按照以下要求替换上述模板中的字段：

```
{
    "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/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/ude/"
```

```
                    "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"
                    ],
                    "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"
            }
        ]
    }
}
```

- 将 `$$$IMAGE$$$` 替换为Sidecar Image镜像地址。
- 配置selector的 `matchExpressions` ，使其可以匹配希望注入的Pod。更多信息，请参见Labels and Selectors。

Istio 1.6.x镜像地址

- 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

8.设置命名空间级别的Sidecar代理

Sidecar代理用于增强服务调用之间的网络安全性、可靠性以及可观测性。ASM对Istio中的Sidecar代理配置进行了细化，提供了命名空间级别的配置能力，帮助您更好的管理Sidecar代理。本文介绍如何按照命名空间设置Sidecar代理的资源、生命周期、终止等待时长等。

前提条件

- 已创建ASM实例，且ASM实例为1.10.5或以上版本。具体操作，请参见创建ASM实例。
- 已创建ACK集群。具体操作，请参见创建Kubernetes托管版集群。
- 添加集群到ASM实例。具体操作，请参见添加集群到ASM实例。
- 已开启自动注入。具体操作，请参见多种方式灵活开启自动注入。

背景信息

ASM按照命名空间设置的Sidecar代理功能的详细描述如下：

功能	描述
设置注入的Istio代理资源	设置注入的Istio代理所需资源和资源限制。
设置istio-init初始化容器资源	设置istio-init初始化容器所需资源和资源限制。
按端口和地址来启用或禁用Sidecar代理	设置拦截对外访问地址、不拦截对外访问地址、按端口使入口流量免于经过Sidecar代理和按端口使出口流量免于经过Sidecar代理。
设置Sidecar代理生命周期	设置Sidecar代理生命周期。更多信息，请参见Container Lifecycle Hooks。
设置Sidecar代理终止等待时长	自定义terminationDrainDuration连接时长，表示Pod停止时，存量的连接最多等待多长时间后强制销毁Sidecar代理。
启用DNS代理功能	启用DNS代理功能，当收到来自应用程序的DNS查询时，Istio代理将进行透明地拦截并提供解析能力。

优先级关系

您可以使用Pod Annotation的方式、按照命名空间级别和全局设置Sidecar代理。这三种方式具有以下优先级关系：

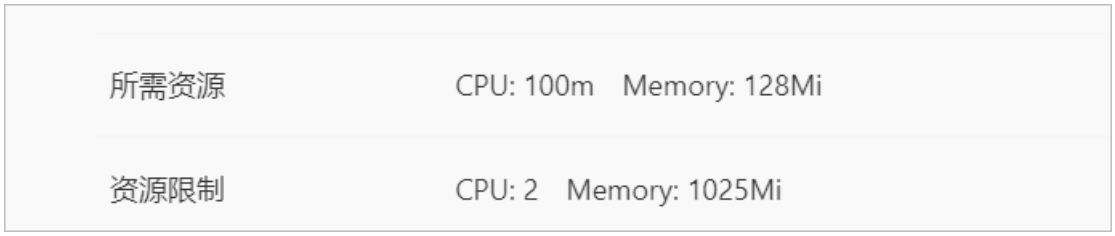
使用Pod Annotation的方式>命名空间级别设置Sidecar代理配置>全局的Sidecar代理设置的优先级

设置注入的Istio代理资源

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格详情页面左侧导航栏选择**Sidecar管理（数据面） > Sidecar代理配置**。
5. 在**Sidecar代理配置**页面单击**命名空间**页签。
6. 选择配置的命名空间，单击**注入的Istio代理资源设置**左侧的>图标，设置Istio资源参数，然后单击**更新设置**。

参数	描述
资源限制	Istio代理资源最大能申请到的CPU和内存，本文设置为CPU为2，内存为1025。
所需资源	Istio代理资源运行时最小使用的CPU和内存，本文设置为CPU为0.1，内存为128。

7. 重启Pod，使Sidecar配置生效。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。
   iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
   iv. 在集群管理页左侧导航栏中，选择**工作负载 > 容器组**。
   v. 在**容器组**页签单击目标Pod右侧**操作**列下的**删除**。
   vi. 在**删除容器组**对话框中单击**确定**。
   稍等一段时间，容器重启后配置生效。
8. 查看设置的Istio代理资源。
   i. 在**容器组**页签单击目标Pod名称。
   ii. 在容器页签下单击**istio-proxy**左侧的 > 图标。
   可以看到Istio代理资源，与所设置的值相同，说明设置Istio代理资源成功。



设置istio-init初始化容器资源

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格详情页面左侧导航栏选择**Sidecar管理（数据面） > Sidecar代理配置**。
5. 在**Sidecar代理配置**页面单击**命名空间**页签。
6. 选择配置的命名空间，单击**istio-init初始化容器资源限制**左侧的 > 图标，设置Istio资源参数，然后单击**更新设置**。

参数	描述
资源限制	Istio-init初始化容器资源最大能申请到的CPU和内存。本文设置COU为0.916 Core，内存为512 MiB。
所需资源	Istio-init初始化容器资源运行时最小使用的CPU和内存。本文设置COU为0.016 Core，内存为18 MiB。

7. 重启Pod，使Sidecar配置生效。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。
   iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
   iv. 在集群管理页左侧导航栏中，选择**工作负载 > 容器组**。
   v. 在**容器组**页签单击目标Pod右侧**操作**列下的**删除**。
   vi. 在**删除容器组**对话框中单击**确定**。

   稍等一段时间，容器重启后配置生效。
8. 执行以下命令，查看设置的istio-init初始化容器资源。

```
kubectl  get pod -n <命名空间名称> <Pod名称>  -o yaml
```

预期输出：

```
- name: DNS_AGENT
image: registry-vpc.cn-hangzhou.aliyuncs.com/acs/proxyv2:1.10.5
imagePullPolicy: IfNotPresent
name: istio-init
resources:
  limits:
    cpu: 916m
    memory: 512Mi
  requests:
    cpu: 16m
    memory: 18Mi
securityContext:
  allowPrivilegeEscalation: false
```

按端口和地址来启用或禁用Sidecar代理

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格详情页面左侧导航栏选择**Sidecar管理（数据面） > Sidecar代理配置**。
5. 在**Sidecar代理配置**页面单击**命名空间**页签。
6. 选择配置的命名空间，单击**按端口和地址来启用/禁用Sidecar代理**左侧的>图标，根据实际情况设置**拦截对外访问的地址访问**、**不拦截对外访问的地址范围**、**设置端口使入口流量经过Sidecar代理**、**设置端口使出口流量经过Sidecar代理**、**设置端口使入口流量免于经过Sidecar代理**、**设置端口使出口流量免于经过Sidecar代理**，然后单击**更新设置**。

   本文设置**拦截对外访问的地址访问**为172.1.1.2/32，**不拦截对外访问的地址范围**172.1.1.1/32，**设置端口使入口流量经过Sidecar代理**为*，**设置端口使出口流量免于经过Sidecar代理**为7015,7016，**设置端口使入口流量免于经过Sidecar代理**为7015,7016，**设置端口使出口流量免于经过Sidecar代理**为7011,7012，
7. 重启Pod，使Sidecar配置生效。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。

iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。

iv. 在集群管理页左侧导航栏中，选择**工作负载 > 容器组**。

v. 在**容器组**页签单击目标Pod右侧**操作**列下的**删除**。

vi. 在**删除容器组**对话框中单击**确定**。

稍等一段时间，容器重启后配置生效。

8. 执行以下命令，查看容器启用或禁用Sidecar代理情况。

```
kubectl  get pod -n <命名空间名称> <Pod名称>  -o yaml
```

预期输出：

```
    - "15001"
    - -z
    - "15006"
    - -u
    - "1337"
    - -m
    - REDIRECT
    - -i
    - 172.1.1.2/32
    - -x
    - 192.168.0.1/32,172.1.1.1/32
    - -b
    - '*'
    - -d
    - 15090,15021,15081,9191,7013,7014
    - -o
    - 7011,7012
```

设置Sidecar代理生命周期

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格详情页面左侧导航栏选择**Sidecar管理（数据面） > Sidecar代理配置**。
5. 在**Sidecar代理配置**页面单击**命名空间**页签。
6. 选择配置的命名空间，单击**Sidecar代理生命周期**左侧的 > 图标，根据实际情况修改以下内容，然后复制到文本框中，然后单击**更新设置**。

```
{
  "postStart": {
    "exec": {
      "command": [
        "pilot-agent",
        "wait"
      ]
    }
  },
  "preStop": {
    "exec": {
      "command": [
        "/bin/sh",
        "-c",
        "sleep 13"
      ]
    }
  }
}
```

- postStart参数下的command：设置启用Sidecar容器后的操作，本文设置为启用Sidecar容器后等待pilot-agent启动。
- preStop参数下的command：设置停止Sidecar容器前的操作，本文设置为停止Sidecar容器前休眠13秒。

7. 重启Pod，使Sidecar配置生效。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。
   iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
   iv. 在集群管理页左侧导航栏中，选择**工作负载 > 容器组**。
   v. 在**容器组**页签单击目标Pod右侧**操作**列下的**删除**。
   vi. 在**删除容器组**对话框中单击**确定**。

   稍等一段时间，容器重启后配置生效。
8. 执行以下命令，查看设置的Sidecar代理生命周期。

```
kubectl  get pod -n <命名空间名称> <Pod名称>  -o yaml
```

预期输出：

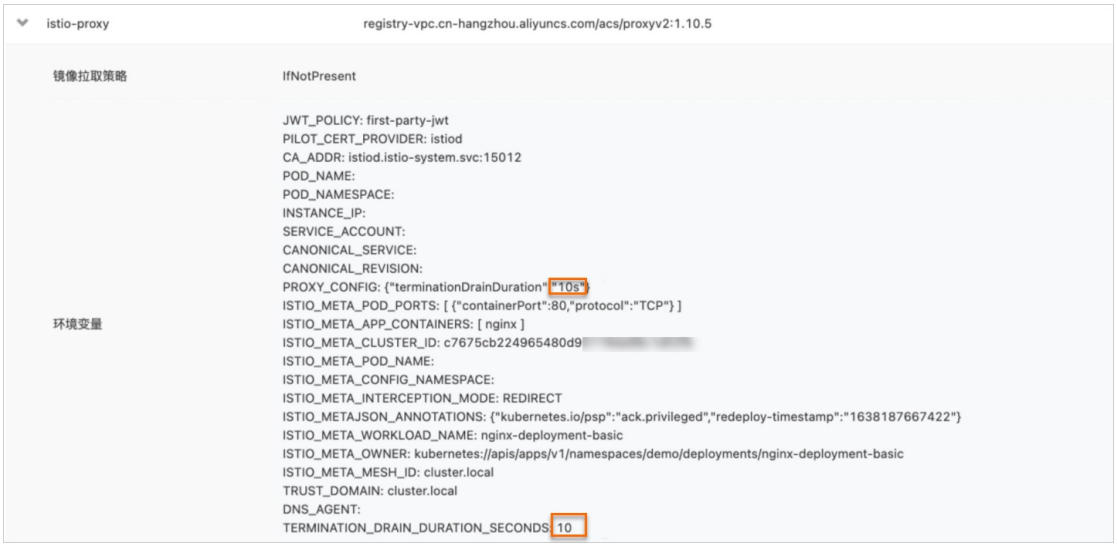
```
    - name: TERMINATION_DRAIN_DURATION_SECONDS
      value: "5"
    image: registry-vpc.cn-hangzhou.aliyuncs.com/acs/proxyv2:1.10.5
    imagePullPolicy: IfNotPresent
    lifecycle:
      postStart:
        exec:
          command:
          - pilot-agent
          - wait
      preStop:
        exec:
          command:
          - /bin/sh
          - -c
          - sleep 13
    name: istio-proxy
```

设置Sidecar代理终止等待时长

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格详情页面左侧导航栏选择**Sidecar管理（数据面） > Sidecar代理配置**。
5. 在**Sidecar代理配置**页面单击**命名空间**页签。
6. 选择配置的命名空间，单击**Sidecar代理终止等待时长**左侧的 > 图标，然后输入等待时长，本文设置**Sidecar代理终止等待时长**为10s，然后单击**更新设置**。
7. 重启Pod，使Sidecar配置生效。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。
   iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
   iv. 在集群管理页左侧导航栏中，选择**工作负载 > 容器组**。
   v. 在**容器组**页签单击目标Pod右侧**操作**列下的**删除**。
   vi. 在**删除容器组**对话框中单击**确定**。
   稍等一段时间，容器重启后配置生效。
8. 查看Sidecar代理终止等待时长。
   i. 在**容器组**页签单击目标Pod名称。

ii. 在容器页签下单击istio-proxy左侧的 > 图标。

可以看到Sidecar代理终止等待时长，与所设置的值相同，说明设置Sidecar代理终止等待时长成功。



启用DNS代理功能

1. 登录ASM控制台。
2. 在左侧导航栏，选择**服务网格 > 网格管理**。
3. 在**网格管理**页面，找到待配置的实例，单击实例的名称或在**操作**列中单击**管理**。
4. 在网格详情页面左侧导航栏选择**Sidecar管理（数据面） > Sidecar代理配置**。
5. 在**Sidecar代理配置**页面单击**命名空间**页签。
6. 选择配置的命名空间，单击**启用DNS代理功能**左侧的 > 图标，选中**启用DNS代理功能**，然后单击**更新设置**。
7. 重启Pod，使DNS配置生效。
   i. 登录容器服务管理控制台。
   ii. 在控制台左侧导航栏中，单击**集群**。
   iii. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
   iv. 在集群管理页左侧导航栏中，选择**工作负载 > 容器组**。
   v. 在**容器组**页签单击目标Pod右侧**操作**列下的**删除**。
   vi. 在**删除容器组**对话框中单击**确定**。
   稍等一段时间，容器重启后配置生效。
8. 执行以下命令，查看设置的DNS代理。

```
kubectl  get pod -n <命名空间名称> <Pod名称>  -o yaml
```

预期输出：

```
    - name: TRUST_DOMAIN
      value: cluster.local
    - name: DNS_AGENT
    - name: TERMINATION_DRAIN_DURATION_SECONDS
      value: "10"
    - name: ISTIO_META_DNS_AUTO_ALLOCATE
      value: "true"
    - name: ISTIO_META_DNS_CAPTURE
      value: "true"
    image: registry-vpc.cn-hangzhou.aliyuncs.com/acs/proxyv2:v1.10.5-33-gba0adb2df7-pro
-aliyun
    imagePullPolicy: IfNotPresent
--
--
    - 7011,7012
    - -q
    - 7015,7016
    env:
    - name: DNS_AGENT
    - name: ISTIO_META_DNS_AUTO_ALLOCATE
      value: "true"
    - name: ISTIO_META_DNS_CAPTURE
      value: "true"
    image: registry-vpc.cn-hangzhou.aliyuncs.com/acs/proxyv2:v1.10.5-33-gba0adb2df7-pro
-aliyun
```