ALIBABA CLOUD

阿里云

服务网格

常见问题

文档版本：20220530

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.服务网格控制平面的命名空间与数据平面集群的命名空间有什么区别

通过托管模式，ASM解耦了服务网格控制平面组件与所管理的数据平面（包括ACK集群）的生命周期管理。通过服务网格ASM控制台，可以新建、定义和删除用于定义服务网格CRD的命名空间。本文介绍服务网格控制平面的命名空间与数据平面集群的命名空间的区别以及如何在服务网格ASM控制台启用自动注入功能。

两种命名空间的区别

通过服务网格ASM控制台或者使用ASM Kubeconfig定义的命名空间隶属于ASM实例本身，与该ASM管理的数据平面集群是独立的，因此ASM托管的控制平面的命名空间可以与数据平面集群的命名空间存在不同的情况。即在服务网格ASM控制台新增或者删除命名空间，并不会影响数据平面Kubernetes集群的命名空间。

启用Sidecar代理自动注入

Kubernetes集群中，通过在命名空间上增加istio-injection=enabled标签，可以启用自动注入功能.。在创建Pod的过程中，将Sidecar Proxy容器自动注入到业务容器中。您也可以通过在命名空间上增加istio-injection=disabled标签，禁用自动注入功能。

服务网格ASM控制台提供了便捷的方式为数据平面的Kubernetes集群增加自动注入Sidecar代理的标签，您可以在服务网格ASM控制台启用自动注入功能。

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**网格实例 > 全局命名空间**。
5. 在**命名空间**页面，找到待注入的命名空间，在**自动注入**列中单击**启用Sidecar自动注入**。

   > **说明** 如果您已经在服务网格ASM控制台启用自动注入，您也可以在**自动注入**列中单击**关闭自动注入**，在**确认**对话框中单击**确定**，关闭自动注入。

6. 在**确认**对话框中单击**确定**。

   > **说明** 在服务网格ASM控制台启用或关闭自动注入功能后，将自动同步相关配置到数据平面的Kubernetes集群。其他在控制平面中新增或者删除命名空间的操作不会同步到数据平面的Kubernetes集群，以保证数据平面集群的资源完整性。

2.如何删除处于终止状态的命名空间

尝试删除Kubernetes命名空间后，长时间停留在终止状态。本文介绍如何解决命名空间处于终止状态的问题。

问题现象

尝试删除Kubernetes命名空间后，长时间停留在终止状态。

```
kubectl delete ns <namespacename>
Error from server (Conflict): Operation cannot be fulfilled on namespaces "<namespacename>"
: The system is ensuring all content is removed from this namespace. Upon completion, this
namespace will automatically be purged by the system.
kubectl describe ns <namespacename>
Name: <namespacename>
Labels: <none>
Annotations: kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"v1","kind":"Na
mespace","metadata":{"annotations":{},"name":"<namespacename>","namespace":""}}
Status: Terminating
```

可能原因

通常是因为从集群中删除的这些命名空间下存在资源。

解决方案

删除命名空间的finalizers。

该选项将会快速清除处于终止状态的命名空间，但可能会导致属于该命名空间的资源留在集群中，因为无法自动删除它们。在finalizers数组为空并且状态为终止之后，Kubernetes将删除命名空间。

1. 打开Shell终端，为您的Kubernetes集群创建一个反向代理。

   ```
   kubectl proxy
   ```

   输出示例如下：

   ```
   Starting to serve on 127.0.0.1:8001
   ```

2. 打开一个新的Shell终端，通过定义环境变量来连接到Kubernetes集群，使用curl测试连接性和授权。

   ```
   export TOKEN=$(kubectl describe secret $(kubectl get secrets | grep default | cut -f1
   -d ' ') | grep -E '^token' | cut -f2 -d':' | tr -d '\t')
   curl http://localhost:8001/api/v1/namespaces --header "Authorization: Bearer $TOKEN"
   --insecure
   ```

3. 获取命名空间定义的内容，以命名空间istio-system为例。

   ```
   kubectl get namespace istio-system -o json > istio-system.json
   ```

4. 将finalizers数组置为空，并重新保存文件。

```
    "spec": {
        "finalizers": [
        ]
    },
```

5. 执行以下命令去除finalizers，以命名空间istio-system为例。

```
curl -X PUT --data-binary @istio-system.json http://localhost:8001/api/v1/namespaces/istio-system/finalize -H "Content-Type: application/json" --header "Authorization: Bearer $TOKEN" --insecure
```

3.如何解决Kubernetes集群中的Pod无法访问入口网关的SLB地址的问题？

本文介绍如何解决数据面Kubernetes集群中的Pod无法访问入口网关的SLB地址的问题。

问题现象

服务网格ASM已添加Kubernetes集群，并且使用Local类型的负载均衡SLB部署入口网关。当应用Pod访问入口网关暴露的SLB地址时，出现以下问题：

- 部分节点上的Pod能访问入口网关暴露的SLB地址。
- 部分节点上的Pod不能访问入口网关暴露的SLB地址。

问题原因

如果Kubernetes集群的服务配置了 `externalTrafficPolicy: Local` ，则只有部署了服务的后端Pod才能访问SLB地址。因为SLB地址是集群外使用，如果集群节点和Pod不能直接访问SLB地址，请求不会路由到负载均衡，而是被当作服务的扩展IP地址，被kube-proxy的iptables或IPVS转发。

如果集群节点或者Pod所在的节点上没有相应的后端服务Pod，就会发生网络不通的问题。而如果有相应的后端服务Pod，则可以正常访问SLB地址。更多信息，请参见iptables规则。

解决方案

- 您可以在Kubernetes集群内通过ClusterIP或者服务名访问SLB地址。其中入口网关的服务名为istio-ingressgateway.istio-system。

  > **说明** 推荐使用该方法。

- 如果您不需要源IP，您可以使用以下方法。

  将入口网关服务中的 `externalTrafficPolicy` 修改为 `Cluster` ，但是在应用中会丢失源IP。关于修改入口网关的更多信息，请参见修改入口网关服务。

  ```
  apiVersion: istio.alibabacloud.com/v1beta1
  kind: IstioGateway
  metadata:
    name: ingressgateway
    namespace: istio-system
    ....
  spec:
    externalTrafficPolicy: Cluster
  ....
  ```

- 如果您使用的是Terway的ENI或者ENI多IP集群，您可以使用以下方法。该方法不会丢失源IP，也支持在集群内访问SLB地址。

  将入口网关服务中的 `externalTrafficPolicy` 修改为 `Cluster` ，并且添加ENI直通的Annotation，例如 `serviceAnnotations: service.beta.kubernetes.io/backend-type: "eni"` 。关于修改入口网关的更多信息，请参见修改入口网关服务。

```
apiVersion: istio.alibabacloud.com/v1beta1
kind: IstioGateway
metadata:
  name: ingressgateway
  namespace: istio-system
  ....
spec:
  externalTrafficPolicy: Cluster
  maxReplicas: 5
  minReplicas: 2
  ports:
    - name: status-port
      port: 15020
      targetPort: 15020
    - name: http2
      port: 80
      targetPort: 80
    - name: https
      port: 443
      targetPort: 443
    - name: tls
      port: 15443
      targetPort: 15443
  replicaCount: 2
  resources:
    limits:
      cpu: '2'
      memory: 2G
    requests:
      cpu: 200m
      memory: 256Mi
  runAsRoot: false
  serviceAnnotations:
    service.beta.kubernetes.io/backend-type: eni
  serviceType: LoadBalancer
```

4.如何在删除ASM网关时保留负载均衡实例

本文介绍删除ASM网关时，负载均衡实例也被删除的问题现象、问题原因和解决方案。

问题现象

删除ASM网关实例时，负载均衡实例也被删除。

问题原因

在创建ASM网关时选择**新建负载均衡**，此时会自动创建负载均衡实例。当删除ASM网关时，自动创建的负载均衡实例也会被删除。

> **说明** 创建ASM网关时选择**使用已有负载均衡**，当删除ASM网关时，已有的负载均衡实例不会被删除。

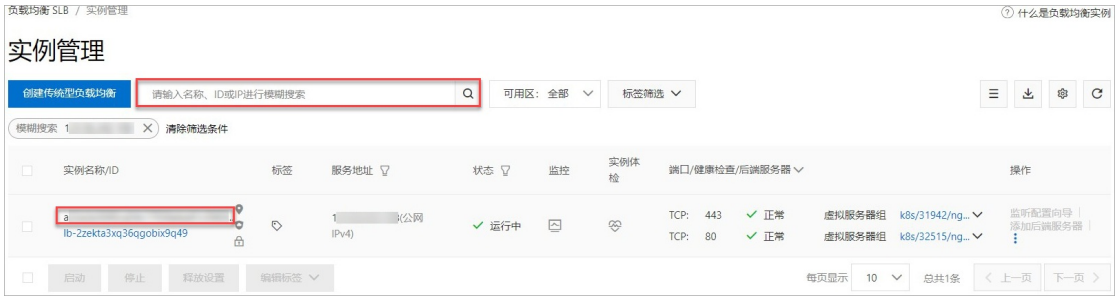
解决方案

您可通过配置ASM网关保留自动创建的负载均衡实例，具体操作步骤如下：

1. 获取ASM网关的IP地址。
   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**ASM网关**。
   v. 在**ASM网关**页面，获取目标ASM网关的IP地址。

2. 获取SLB实例ID。
   i. 登录传统型负载均衡CLB控制台。
   ii. 在负载均衡SLB左侧导航栏，单击**传统型负载均衡 CLB（原SLB） > 实例管理**。

iii. 在**实例管理**页面的搜索框中，输入并搜索步骤1获取的IP地址，然后获取SLB实例ID。



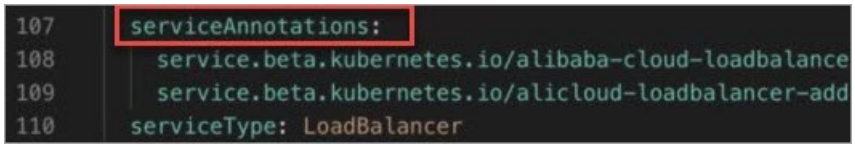
3. 修改IstioGateway YAML文件。

i. 在**ASM网关**页面，单击目标AMS网关右侧**操作**列下方的**查看YAML**。

ii. 添加如下内容至IstioGateway的 `serviceAnnotations` 中，然后单击**确认**。

`{YourSLBId}` 为步骤2获取的SLB实例ID。

```
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: {YourSLBId}
```



修改完毕后，ASM网关会自动重新部署，此时**状态**显示为**创建中**。当**状态**显示为**运行中**，表示ASM网关重建成功。

4. 删除ASM网关并验证SLB实例是否被删除。

i. 在**ASM网关**页面，单击目标AMS网关右侧**操作**列下方的**删除**，然后在弹出的**确认**对话框中，单击**确认**。

ii. 在传统型负载均衡CLB控制台的**实例管理**页面搜索框中，输入并搜索步骤1获取的IP地址。

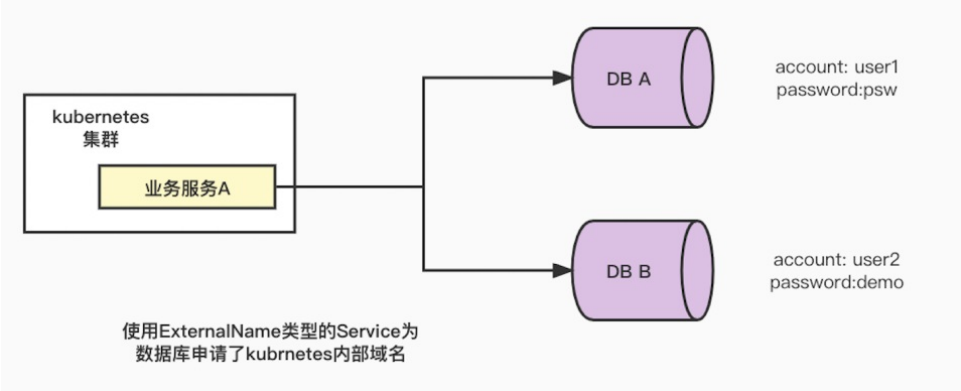
如下图所示，**实例管理**页面中存在目标SLB实例，说明删除AMS网关后，SLB实例未被删除。

5.为什么ASM内的服务能够访问外部数据库，但认证会失败

本文介绍为什么ASM内的服务能够访问外部数据库，但认证会失败的问题现象、问题原因和解决方案。

问题现象

在使用ASM 1.10及之前版本时，如果有多个外部数据库，并且数据库登录的认证信息不同，在业务服务A加入服务网格并注入Sidecar代理后，请求数据库时会提示数据库服务认证失败。



问题原因

由于您对集群外部TCP服务（数据库）使用Kubernetes ExternalName类型的Sevice进行了DNS别名声明，业务服务在加入服务网格后，因为缺少集群IP和外部TCP服务做映射，会按照TCP服务对应的端口进行匹配，这样有可能会将业务服务对数据库A的请求路由到数据库B，而这两个数据库登录信息不一样，导致认证失败。

解决方案

方案一：升级ASM的版本

升级ASM至1.11及以上版本，可以解决该问题。具体操作，请参见升级ASM实例版本。

方案二：使用服务条目

您可以在ASM中创建服务条目，使得网格内的服务通过访问服务条目来访问指定IP的数据库。您也可以将服务条目创建到您指定服务的命名空间下，从而限定服务条目作用范围。

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**集群与工作负载管理 > 服务条目**，然后在右侧页面单击**使用YAML创建**。
5. 在**创建**页面设置**命名空间**为istio-system，选择任意模板，将**YAML**文本框中的内容替换为以下内容，单击**创建**。

```
apiVersion: networking.istio.io/v1beta1
kind: ServiceEntry
metadata:
  name: mysql-demo
spec:
  addresses:
  - 172.1.xx.xx
  endpoints:
  - address: 172.1.xx.xx
  hosts:
  - test-mysql.com
  location: MESH_EXTERNAL
  ports:
  - name: tcp
    number: 3306
    protocol: TCP
  resolution: STATIC
```

- addresses：设置为数据库的IP地址。
- number：设置为数据库的端口。

6.为什么使用kubectl命令列出Istio网关会返回空值或者没有返回Istio网关资源？

本文介绍使用kubectl命令列出Istio网关会返回空值或者没有返回Istio网关资源的问题现象、问题原因和解决方案。

问题现象

已创建Istio网关。执行以下命令，返回 `No resources found` ，或者没有返回stio网关资源。

```
kubectl get gateway --all-namespaces
```

问题原因

ASM v1.8.6及以上版本可能会发生此问题，因为从该版本开始ASM会自动安装自定义资源gateway.networking.x-k8s.io。更多信息，请参见使用Gateway API定义路由规则。

Kubernetes Gateway API和Istio API都有一个名为Gateway的资源。虽然它们功能类似，但不是相同的资源。使用kubectl命令时，Gateway的名称会重叠。执行 `kubectl get gateway` 可能返回的是Kubernetes网关资源，而不是Istio网关资源。如果此时没有定义Kubernetes网关资源，则返回的会是空值。如果定义了Kubernetes网关资源，则返回的是Kubernetes网关资源。

解决方案

- 使用ASM控制台查看Istio网关。
- 在kubectl命令中使用完整的资源名称或易辨认的简称。

  Kubernetes网关的简称为gtw，Istio网关的简称为gw。您可以执行 `kubectl get gw` 或 `kubectl get gateways.networking.istio.io` 以确保返回的是Istio网关。

7.为什么定义的DestinationRule失效

本文介绍定义的DestinationRule失效的问题现象、问题原因和解决方案。

问题现象

为服务部署DestinationRule后，使用客户端调用服务，发现DestinationRule失效，调用服务失败。

问题原因

ASM路由一个请求时，会根据固定查找过程查找DestinationRule去完成路由。如果您的DestinationRule部署在查找过程之外的命名空间，则DestinationRule会失效。查找过程如下所示：

1. 从客户端命名空间查找，即从发起调用的客户端所在的命名空间查找是否存在相应的DestinationRule。
2. 从服务命名空间查找，即从被调用的服务所在的命名空间查找是否存在相应的DestinationRule。
3. 从ASM根命名空间（istio-system）查找，即从istio-system命名空间查找是否存在相应的DestinationRule。

例如，使用以下YAML文件在ns1命名空间中定义了myservice服务的DestinationRule。从 `host` 字段得知，myservice服务是部署在default命名空间。

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: myservice
spec:
  host: myservice.default.svc.cluster.local
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 100
```

- 如果您在ns1命名空间发起对myservice服务的调用，由于myservice的DestinationRule也定义在ns1命名空间中，所以可以查找到可用于完成路由的DestinationRule，最终可以调通服务。
- 如果您在ns2命名空间发起对myservice服务的调用，则对myservice服务的调用将会失败。存在以下DestinationRule查找过程：
  i. 发起调用请求的客户端位于ns2命名空间，因此会从ns2命名空间查找相应的DestinationRule，但是ns2命名空间中并不存在相应的DestinationRule，查找会失败。
  ii. 被调用的myservice服务是位于default命名空间，因此从default命名空间查找相应的DestinationRule，但是default命名空间中并不存在相应的DestinationRule，查找会失败。
  iii. ASM的根命名空间固定为istio-system，因此从istio-system命名空间查找相应的DestinationRule，但是istio-system命名空间中并不存在相应的DestinationRule，查找会失败。

解决方案

部署DestinationRule时，您需要将DestinationRule部署在以下命名空间中：

- ASM根命名空间。
- 服务所在的命名空间。
- 发起调用的客户端所在的命名空间。

> **说明** VirtualService不存在命名空间的限制。VirtualService无论在哪个命名空间定义，默认在所有命名空间都可见，除非在YAML文件中通过 `exportTo` 改变这一默认行为。

8.为什么为Pod注入Sidecar后，Pod处于init crash的状态

本文介绍为Pod注入Sidecar后，Pod处于 `init crash` 状态的问题现象、问题原因和解决方案。

问题现象

为Pod注入Sidecar之后，执行以下命令，查看Pod状态，发现Pod处于 `init crash` 的状态。

```
kubectl get pod
```

预期输出：

```
NAME                READY       STATUS                  RESTARTS        AGE
details-v1-u****    0/2         Init:Error              1               12h
productpage-n****   0/2         Init:CrashLoopBackOff   3               12h
```

然后执行以下命令，查看istio-init容器日志。

```
kubectl --kubeconfig=${USER_KUBECONFIG} -c istio-init logs ${pod}
```

预期输出：

```
......
......
-A ISTIO_OUTPUT -d 127.0.**.**/32 -j RETURN
-A ISTIO_OUTPUT -d 192.168.0.1/32 -j RETURN
-A ISTIO_OUTPUT -j ISTIO_REDIRECT
COMMIT
2022-03-23T06:42:21.179567Z     info    Running command: iptables-restore --noflush /tmp/ipt
ables-rules-1648017741179373856.txt4205119933
2022-03-23T06:42:21.185698Z     error   Command error output: xtables other problem: line 2
failed
2022-03-23T06:42:21.185720Z     error   Failed to execute: iptables-restore --noflush /tmp/
iptables-rules-1648017741179373856.txt4205119933, exit status 1
```

可以看到istio-init容器日志包含 `Failed to execute: iptables-restore` 信息。

问题原因

检查是否曾经手动通过 `docker container rm/docker container prune/docker system prune` 等命令清理了已经退出的istio-init容器，或者是否存在清理容器的相应的定时任务。

清理已退出的istio-init容器，会导致K8s检测到Pod关联的容器不存在，此时K8s会重新启动被删除的容器。由于之前已创建了iptables规则，istio-init容器不能再次执行iptables规则，导致新启动的istio-init容器设置iptables规则失败而崩溃。

解决方案

您需要重建Pod，重建后，服务的Pod将恢复正常状态。

如果您下次仍然要使用命令或定时任务脚本清理数据，需要注意以下事项：

- 如果您需要使用命令清理数据，您需要在批量清理数据命令中过滤istio-init容器，防止istio-init容器被清理。

```
docker system prune --filter "label!=io.kubernetes.container.name=istio-init"
```

- 如果您需要使用定时任务脚本清理istio-init容器，您需要在定时任务脚本中将 `docker system prune` 命令修改为以下命令，过滤istio-init容器，防止istio-init容器被清理。

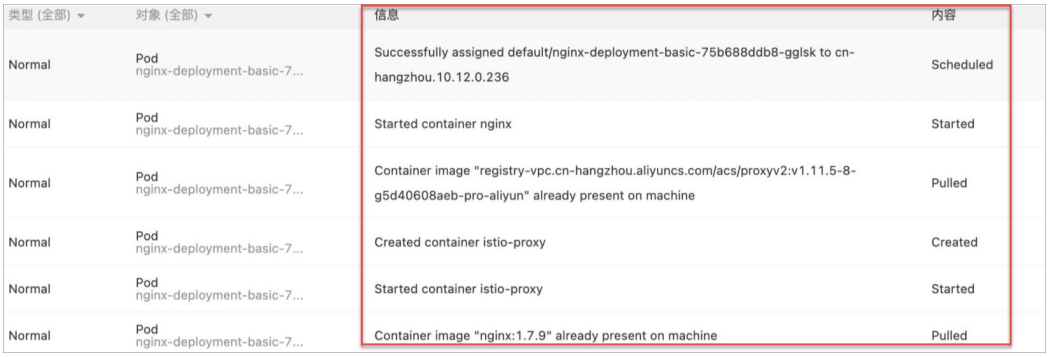
```
docker system prune --filter "label!=io.kubernetes.container.name=istio-init"
```

9.为什么注入Sidecar后，健康检查总失败或者无效

本文介绍注入Sidecar后，健康检查失败或者无效的问题现象、问题原因和解决方案。

问题现象

注入Sidecar后，健康检查失败或者无效。本文以TCP健康检查端口8087为例，启用mTLS后，在容器服务管理控制台的容器组详情页面**事件**页签下，未显示8087端口的健康检查信息。



类型 (全部)	对象 (全部)	信息	内容
Normal	Pod nginx-deployment-basic-7...	Successfully assigned default/nginx-deployment-basic-75b688ddb8-gglsk to cn-hangzhou.10.12.0.236	Scheduled
Normal	Pod nginx-deployment-basic-7...	Started container nginx	Started
Normal	Pod nginx-deployment-basic-7...	Container image "registry-vpc.cn-hangzhou.aliyuncs.com/acs/proxyv2:v1.11.5-8-g5d40608aeb-pro-aliyun" already present on machine	Pulled
Normal	Pod nginx-deployment-basic-7...	Created container istio-proxy	Created
Normal	Pod nginx-deployment-basic-7...	Started container istio-proxy	Started
Normal	Pod nginx-deployment-basic-7...	Container image "nginx:1.7.9" already present on machine	Pulled

问题原因

在服务网格开启mTLS后，kubelet向Pod发送的健康检查请求被Sidecar拦截，而kubelet没有对应的TLS证书，导致健康检查失败。

解决方案

配置端口健康检查流量免于经过Sidecar代理，具体操作步骤如下：

配置端口健康检查流量免于经过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**命名空间**页签下，选中相应的命名空间，单击**按端口或地址来启用/禁用Sidecar代理**页签，配置相应参数。

   参数配置说明如下：

参数	说明
设置端口使入口流量免于经过Sidecar代理	配置入口流量免于经过Sidecar代理的端口，本文配置为*8087*。
设置端口使出口流量免于经过Sidecar代理	配置出口流量免于经过Sidecar代理的端口，本文配置为*8087*。

6. 配置完成后，单击**更新设置**。

查看健康检查结果

1. 登录容器服务管理控制台。
2. 在控制台左侧导航栏中，单击**集群**。
3. 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**。
4. 在集群管理页左侧导航栏中，选择**工作负载 > 容器组**。
5. 单击目标容器组名称或右侧的详情，进入容器组详情页面。
6. 在容器组详情页面，单击**事件**页签。

   如下图所示，端口8087的健康检查生效。

10.为什么注入Sidecar代理后，耗时较长的请求会丢失或失败

本文介绍注入Sidecar代理后，当Pod停止时，耗时较长的请求会丢失或失败的问题现象、问题原因和解决方案。

问题现象

为Pod注入Sidecar代理后，当Pod停止时，存在以下两个问题：

- 调用该Pod的一些耗时较长的请求会丢失。
- 使用该Pod调用其他服务可能会失败。

问题原因

为Pod注入Sidecar代理后，业务Pod的流量会被istio-proxy代理。当Pod开始停止时，对应的Service不再转发流量给Pod。

在Istio中，默认收到退出信号5秒后会强制停止istio-proxy，不再接收新的Inbound连接（入口流量），将会继续处理存量的Inbound连接，Outbound连接（出口流量）不受影响，可以正常发起。如果被停止的服务提供的接口调用的耗时较长，已有的Inbound连接和Outbound连接即使没有处理完成也会被终止。

解决方案

方案一：修改Sidecar代理终止等待时长

您可以延长Sidecar代理终止等待时长，使得Inbound和Outbound连接可以在该时长内处理完成。

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**命名空间**，单击**Sidecar代理终止等待时长**，输入时间，单击**更新设置**。

方案二：配置Sidecar代理生命周期

如果您无法预估请求的最大等待时长，建议配置Sidecar代理生命周期的preStop脚本。使用preStop脚本判断是否还存在请求连接，无请求连接后，将等待默认时长（5秒）再完成退出 。

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**命名空间**，单击**Sidecar代理生命周期**，将以下内容输入到文本框中，单击**更新设置**。

```
{
  "postStart": {
    "exec": {
      "command": [
        "pilot-agent",
        "wait"
      ]
    }
  },
  "preStop": {
    "exec": {
      "command": [
        "/bin/sh",
        "-c",
        "while [ $(netstat -plunt | grep tcp | grep -v envoy | wc -l | xargs) -ne 0 ];
do sleep 1; done"
      ]
    }
  }
}
```