

# 应用服务网格(ASM)

## 最佳实践

文档版本 05

发布日期 2023-05-12



**版权所有 © 华为云计算技术有限公司 2024。保留一切权利。**

未经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

# 目 录

<b>1 数据面 sidecar 升级不中断业务</b>	<b>1</b>
<b>2 面向 Dubbo 协议的服务治理</b>	<b>4</b>
2.1 简介	4
2.2 服务发现模型	4
2.3 SDK 适配方式	5
2.3.1 PASSTHROUGH 方案	5
2.3.2 静态目标服务	6
<b>3 网关访问保留源 IP</b>	<b>7</b>
<b>4 虚拟机治理</b>	<b>14</b>
4.1 方案介绍	14
4.2 约束与限制	16
4.3 部署 ASM-PROXY	17
4.4 验证服务访问	23
4.4.1 虚拟机服务访问容器服务	24
4.4.2 容器服务访问虚拟机服务	26
4.4.3 虚拟机服务访问虚拟机服务	31
4.5 流量治理	32
4.5.1 虚拟机服务添加网关和路由	32
4.5.2 虚拟机服务灰度发布	36
4.5.3 虚拟机服务配置故障注入	43
4.6 卸载 ASM-PROXY	45
<b>5 如何搭建 IPv4/IPv6 双栈网格</b>	<b>47</b>

# 1

## 数据面 sidecar 升级不中断业务

应用服务网格作为集群网络管理的重要工具，为网格内的服务提供流量治理与流量监控的能力。sidecar作为应用服务网格数据面的重要组件，需要依赖服务业务pod的更新来实现sidecar的升级和重新注入。

本章节主要介绍如何实现数据面sidecar升级过程中，不中断服务的业务流量。

### 配置服务实例数

为保证您的服务在sidecar升级的过程中不中断业务流量，首先确保您的服务实例数大于等于2，升级策略为滚动升级（rollingUpdate）。

相关滚动升级策略如下，供参考：

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

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

#### 配置项说明：

- 服务实例数：deployment.spec.replicas >= 2
- 升级策略：deployment.spec.strategy.type == RollingUpdate
- 滚动升级最小存活实例数：deployment.spec.replicas - deployment.spec.strategy.maxUnavailable > 0

## 添加 readiness 探针

添加readiness探针，可以保证您的新实例pod在真正准备就绪时，才开始接管业务流量。这就避免了在新的实例pod未启动时，接管业务流量造成的访问不通问题。

相关配置如下：

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

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

配置项说明：

readiness探针配置项：deployment.spec.template.spec.containers[i].readinessProbe  
其中，包括探针检查初始时间，检查间隔，超时时间等配置。

## 设置服务就绪时间

服务就绪时间，minReadySeconds：用于标识pod的ready时间至少保持多长时间，才会认为服务是运行中。

相关配置如下：

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

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

配置项说明：

服务就绪时间：deployment.spec.minReadySeconds，可根据您业务的实际情况确定。

## 配置优雅关闭时间

terminationGracePeriodSeconds，优雅关闭时间。在滚动升级过程中，首先会移除旧的服务实例pod的endpoint，并将实例pod的状态置为Terminating，这时K8S会发送SIGTERM信号给pod实例，并等待优雅关闭时间后，将pod强制终止。您可以利用这段时间，处理未完成的请求：

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

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

### 配置项说明：

优雅关闭时间： deployment.spec.template.spec.terminationGracePeriodSeconds，  
默认值为30s，可根据业务诉求适当调整。

## 配置 preStop

preStop会在实例pod中止前调用，可以用来实现业务pod的优雅关闭。此处需要根据业务诉求来进行相应的配置，以Nginx为例。

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

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

在“lifecycle”下的“preStop”中，定义了一个命令nginx -s quit; sleep 10，这个命令首先会发送一个优雅关闭信号给Nginx进程，然后暂停10秒。这样，在Pod终止之前，Nginx会有足够的时间完成正在处理的请求并优雅地关闭。

需要注意的是，sleep 10中的10秒是一个示例值，您可以根据实际需要和应用程序的性能来调整这个值，关键是为Nginx提供足够的时间来优雅地关闭。

类似地，您可以选择运行自定义命令或自定义脚本，来优雅关闭您的服务进程。

# 2 面向 Dubbo 协议的服务治理

## 2.1 简介

Dubbo作为一种特有协议，需要有如下支持：

- 网格服务数据面Envoy支持对Dubbo协议的解析和流量管理。
- 网格控制面支持对Dubbo治理规则的配置。支持灰度发布、负载均衡、访问授权等服务管理。

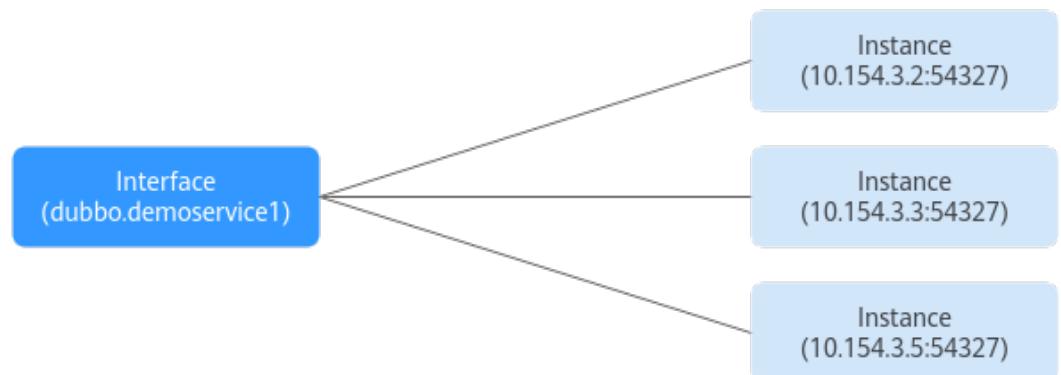
另外，Dubbo的服务发现模型和Kubernetes、Spring Cloud等服务发现模型不一致，需要额外的处理。

## 2.2 服务发现模型

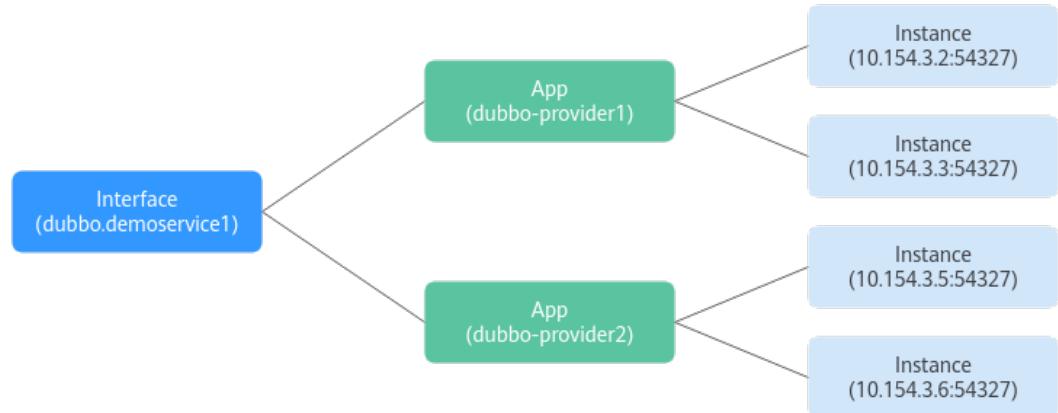
Dubbo现有模型存在的问题（来自Dubbo社区2.7.4总结）：

- 在微服务架构中，注册中心管理的对象是应用（服务），而非对外的服务接口，不过目前Dubbo的注册中心管理的对象是Dubbo服务接口，与Spring Cloud或Cloud Native注册方式背道而驰。
- 一个Dubbo应用（服务）允许注册N个Dubbo服务接口，当N越大时，注册中心的负载越重。

Dubbo现有服务模型：根据Dubbo接口查找服务实例。



Dubbo Cloud Native服务发现模型，将原来Interface一级的服务发现拆分成两级，基于App找实例地址。

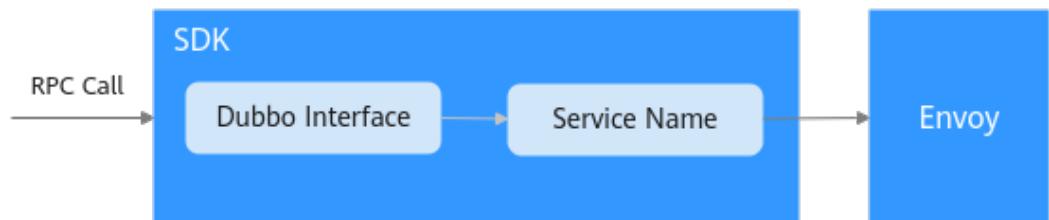


## 2.3 SDK 适配方式

### 2.3.1 PASSTHROUGH 方案

#### 方案介绍

SDK中客户端使用Interface调用目标服务时，修改原有服务发现逻辑。将原有通过Interface查找服务实例，修改为通过接口查找服务名，直接对服务名发起访问。



#### 详细说明

对于Dubbo协议的不同版本会有不同：

- 2.7.4+版本：2.7.4以上的Cloud Native版本中重构过与Kubernetes一致的服务发现模型，直接可以从Interface关联到服务信息。
- 2.7.3及之前版本：Dubbo社区版本未提供Interface到Service的二级关系，需要SDK根据自己的实际使用方式来维护Interface到服务的映射关系。如可以使用在服务注册时通过扩展信息的方式提供服务名等信息。

客户实际使用中可以根据自己的SDK使用情况选择灵活的处理方式。对于老版本的SDK可以基于现有的服务注册和服务发现流程，做如下处理：

1. 在注册信息中扩展Service的定义。在服务部署时会通过环境变量将服务的元数据信息注入到SDK中，包括appname、namespace，分别表示部署的服务名、部署的命名空间。
2. 在服务启动时向注册中心注册Dubbo接口和Kubernetes服务名和命名空间的关系。

3. 在客户端发起访问时，根据原有服务发现的流程根据Interface查询到服务的元数据，并用对应的服务信息组装RPC请求。建议在Dubbo请求中使用Attachment扩展字段存储appname、namespace信息。

## 2.3.2 静态目标服务

### 方案介绍

通过**dubbo:reference**在Dubbo服务的服务消费者中对引用的服务提供者进行配置。  
使用选项url定义点对点直连服务提供者地址，绕过注册中心，直接调用目标服务。

### 详细说明

如果原Dubbo服务中使用的是xml配置文件，则只需要修改配置文件即可。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>
    <!-- 提供哪些接口给消费者调用 -->
    <dubbo:reference id="helloService" interface="com.dubbo.service.HelloService" url="dubbo://
    helloService:20880" />
</beans>
```

如果服务中使用注解的方式来定义引用的目标服务，则需要修改代码中对目标服务的注解。

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

# 3 网关访问保留源 IP

## 操作场景

服务通过网关访问时，默认情况下，目标容器中看到的不是客户端的源IP，如果需要保留源IP，请参考本节指导操作。

## 配置方法

请在CCE控制台“服务发现”页面，istio-system命名空间下，更新服务所关联的网关服务，将服务亲和改成“节点级别”。前提是已开启ELB的获取客户端IP功能（当前为默认开启）。

```
spec:
  ports:
    - name: http-productpage
      protocol: TCP
      port: 82
      targetPort: 1025
      nodePort: 30749
  selector:
    app: istio-ingressgateway
    istio: Ingressgateway
    clusterIP: 10.247.85.97
    clusterIPs:
      - 10.247.85.97
    type: LoadBalancer
    sessionAffinity: None
    loadBalancerIP: 116.63.46.62
    externalTrafficPolicy: Local
    healthCheckNodePort: 30809
  status:
    loadBalancer:
      ingress:
        - ip: 116.63.46.62
          ip: 192.168.0.28
  apiVersion: v1
  kind: Service
```

externalTrafficPolicy：表示此Service是否希望将外部流量路由到节点本地或集群范围的端点。有两个可用选项：Cluster（默认）和Local。Cluster隐藏了客户端IP，可能导致第二跳到另一个节点，但具有良好的整体负载分布。Local保留客户端源IP并避免LoadBalancer和NodePort类型服务的第二跳，但存在潜在的不均衡流量传播风险。

## 验证方式

结合httpbin镜像在“x-forward-for”字段中可以看到源IP，httpbin是一个HTTP Request & Response Service，可以向他发送请求，他将会按照指定的规则将请求返回。httpbin镜像可在SWR中搜索。使用httpbin镜像进行验证时请确保集群已开通网格。

1. 登录ASM应用服务网格控制台，选择一个可用的测试网格并单击进入。
2. 选择左侧“网格配置”查看其关联的集群。

The screenshot shows the basic information page for a mesh network. Key details include:

- Network Name: mesh
- Network Type: 专有网
- Network ID: 67a28b03-aab7-11ed-8108-0255ac1001bb
- Network Status: 运行中 (Running)
- Creation Time: 2023/02/12 17:27:02 GMT+08:00
- Cluster Name: cluster1
- Status: 正常 (Normal)
- Cluster Version: v1.21
- Cluster Service Version: 10.247.0.9/16
- Cluster Network Type: VPC网通
- Cluster Creation Time: 2023/02/12 17:27:02 GMT+08:00
- Cluster Operation: 同步详情

3. 单击集群名称进入集群详情页，单击对应集群右上角第三个图标“工作负载”进入“工作负载”页签。

The screenshot shows the cluster details page for cluster1. The "Workload" tab is selected. Key metrics shown are:

- CPU 使用率: 9.19%
- 内存 使用率: 34.78%
- CPU 总数: 4 Core
- 内存 总数: 8 GiB

单击右上角“创建负载”按钮。

The screenshot shows the "Create Deployment" page. The "Basic Information" section includes:

- Deployment Type: 无状态负载 (Stateless Deployment)
- Deployment Name: httpbin
- Namespace: default
- Replica Count: 2
- Annotations: 请编辑描述信息

4. 配置工作负载的信息。

### 基本信息

- 负载类型：选择无状态工作负载Deployment。
- 负载名称：命名工作负载为httpbin。
- 命名空间：选择工作负载的命名空间，默认为default。
- 其余参数使用默认值。

The screenshot shows the "Container Configuration" section of the deployment creation page. It includes:

- Container Name: httpbin
- Image: httpbin
- Port: 80
- 资源限制: CPU 100m, Memory 200Mi
- 资源请求: CPU 100m, Memory 200Mi
- 启动命令: /bin/sh -c curl -f -s -L -X GET http://\$1
- 环境变量: 无
- 卷挂载: 无

### 容器配置

- 基本信息：

- 容器名称：为容器自定义一个名称
- 镜像名称：单击后方“选择镜像”，在右上角搜索框搜索“httpbin”镜像并选择，单击“确认”。
- 镜像版本：选择一个镜像版本
- 其余参数使用默认值。



## 服务配置

服务 (Service) 是用来解决Pod访问问题的。每个Service有一个固定IP地址，Service将访问流量转发给Pod，而且Service可以给这些Pod做负载均衡。

单击服务配置参数下面的“+”进入创建服务页面。

- Service名称：Service名称为工作负载名称。
- 访问类型：选择集群内访问。
- 端口设置：
  - 协议：TCP
  - 容器端口：80（以实际访问端口为准）
  - 服务端口：80（以实际访问端口为准）

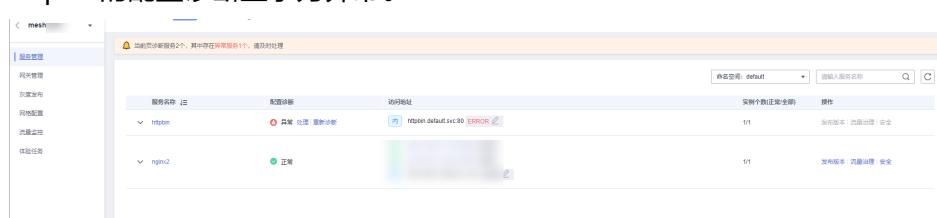
### 创建服务



5. 单击右下角“确定”完成服务创建。
6. 单击右下角“创建工作负载”完成工作负载创建。
7. 在集群详情页选择左侧“服务发现”页签，可在服务列表中查看到所创建的httpbin服务。



8. 返回ASM应用服务网格，选择左侧“服务管理”页签，在服务管理中可查看到httpbin的配置诊断显示为异常。



9. 单击此服务配置诊断中的“处理”按钮，按照弹出“配置诊断”页面对应的修复指导进行修复。

配置诊断

The screenshot shows a 'Configuration Diagnosis' interface with three items listed:

检查项	检查结果	操作
Service 的端口名称是否符合Istio规范	失败 ② 访问端口 80 协议 —	修复指导
Service的选择器中是否配置了version标签	失败 ②	修复指导
服务是否配置了默认版本的服务路由，路由配置是否正确	失败 ②	修复指导

At the bottom are two buttons: '重新诊断' (Re-diagnose) and a prominent red '一键修复' (One-click Repair).

- 以“Service的端口名称是否符合Istio规范”为例，选择协议为“http”，单击“一键修复”。

配置诊断

The screenshot shows the same 'Configuration Diagnosis' interface after repair, with all items now successful:

检查项	检查结果	操作
Service 的端口名称是否符合Istio规范	成功 访问端口 80 协议 http	修复指导
Service的选择器中是否配置了version标签	成功	修复指导
服务是否配置了默认版本的服务路由，路由配置是否正确	成功	修复指导

At the bottom are two buttons: '重新诊断' (Re-diagnose) and a blue '一键修复' (One-click Repair).

10. 选择左侧“网关管理”页签，单击右上角“添加网关”，在弹出“添加网关”页面输入配置信息。

配置信息

- 网关名称：httpbin。
- 集群选择：网格所关联使用的集群。
- 负载均衡：选择公网、选择一个elb公网。
- 对外协议：选择HTTP。
- 对外端口：自定义端口。
- 外部访问地址：负载均衡中所选elb公网地址。

## 添加网关

### 基本信息

\* 网关名称

httpbin

\* 集群选择

cluster1

### 访问方式

\* 负载均衡 [?](#)

公网

elb-1198(



创建负载均衡

仅支持集群所在 VPC vpc-e5b9 下的负载均衡实例，查询结果已自动过滤

### 访问入口

\* 对外协议

HTTP

GRPC

TCP

TLS

HTTPS

\* 对外端口

80

\* 外部访问地址 [?](#)

- 单击“路由配置”下方的“+”在弹出“添加路由”页面添加路由。
  - URL：选择完全匹配并输入映射。
  - 命名空间：服务所在命名空间。
  - 目标服务：默认设置。

## 添加路由

★ URL

完全匹配 /get

★ 命名空间

default

★ 目标服务

httpbin 80

当前服务已根据网关协议自动过滤。了解更多

重写

重写HTTP URI和Host/Authority头，于转发前执行

**确定** **取消**

配置完成后单击“确定”。

11. 单击“确定”完成网关添加。
12. 选择左侧“服务管理”页签，可以在“访问地址”查看到所创建路由的外部访问地址。

The screenshot shows the 'Access Address' table for the 'httpbin' service. It lists two entries: 'http://<IP>/ HTTP' and 'http://<IP>:80 HTTP'. The table includes columns for Service Name, Configuration, Access Address, and Actions.

13. 在映射的外部访问地址后加上“?show\_env=1”，访问添加字段后的外部访问地址，例如：http://xxx.xxx.xxx:80/get?show\_env=1。可以在“x-forward-for”字段中查看网关获取的IP为容器段IP。

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

14. 返回集群详情页，选择左侧导航栏“服务发现”，更改服务所关联的网关服务的配置。方法如下：

下拉上方“命名空间”列表选择“istio-system”。

The screenshot shows the 'Service Discovery' table. It lists services like 'httpbin-svc', 'istio-egressgateway', and 'istiod'. The 'Namespace' column shows 'istio-system' for all listed services. The table includes columns for Service Name, Selector, Namespace, Service Type, IP Address, Port, and Status.

展开服务后方“更多”选项，单击“更新”，在弹出“更新服务”页面将“服务亲和”更改为“节点级别”，勾选“我已阅读《负载均衡使用须知》”，单击“确定”。



15. 返回13中访问的外部地址并刷新，若设置之后“x-forward-for”字段中显示的网关获取IP的结果为本机源IP，则完成验证。

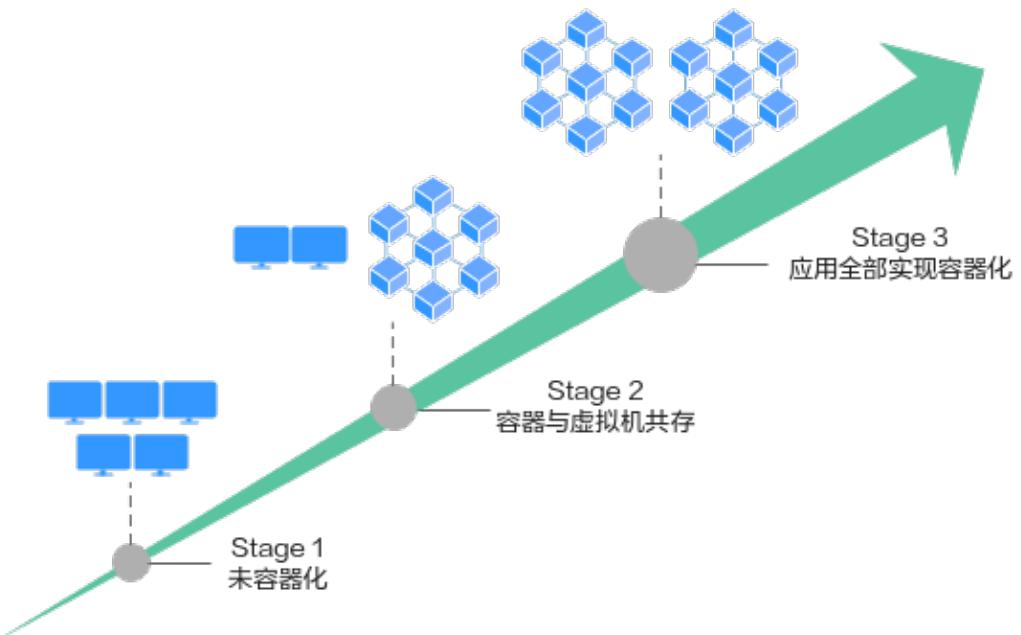
...  
...  
...  
x-forward-for: xxxx

# 4 虚拟机治理

## 4.1 方案介绍

### 应用现状

将应用迁移到云原生架构，不断容器化的过程中，一般会经历三个阶段，如下图所示。



- 阶段一：所有应用都部署在虚拟机上。
- 阶段二：应用既部署在虚拟机上也部署在容器中，正在从虚拟机向容器迁移，并且使用Kubernetes管理容器。
- 阶段三：所有应用都部署在容器中，使用Kubernetes管理容器，并且使用Istio管理应用间的通信。

因为种种原因，容器与虚拟机共存将是一个长期的过程，但容器化的趋势不变。

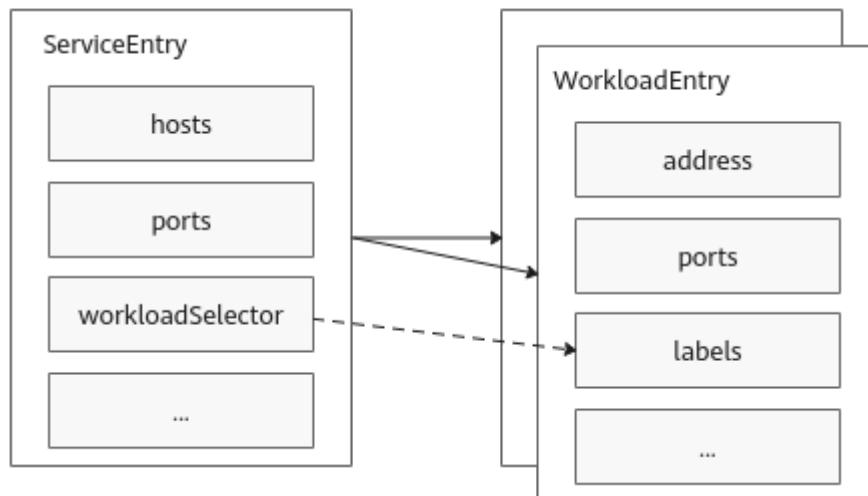
在阶段二中，大量的虚拟机应用并不能很快容器化，人们通常会将新业务和少量应用率先实现容器化，并部署到Kubernetes中。在应用尚未完全实现容器化，处于过渡阶段时会遇到诸多问题，比如虚拟机上的非容器化服务是否可以像容器化服务一样进行流量治理？虚拟机如何访问容器中的服务？是否可以将容器和虚拟机纳入一个统一的控制平面来管理？

本文介绍一种通过为虚拟机安装代理来实现虚拟机治理的方案，可以有效解决以上几个问题。

## 解决方案

原理上网格可以提供对容器和虚拟机同等的治理能力，不同于Kubernetes容器上网格的数据面Proxy可以自动注入，自动流量拦截，基于Kubernetes的服务和实例模型自动进行服务发现，在虚拟机场景下需要手动安装数据面Proxy（ASM-PROXY），在安装过程中初始化流量拦截。具体来说，ASM-PROXY是虚拟机部署工具包，用于连接虚拟机应用所在的数据平面与ASM提供的控制平面。ASM-PROXY部署在虚拟机节点中，负责与ASM通信获取xDS信息，拦截非容器应用流量并执行网格化操作，例如流量管理、请求安全认证、上报链路追踪数据等。

在服务发现上，虚拟机形态的服务通过WorkloadEntry来描述一个虚拟机上安装了Proxy的实例，该虚拟机实例可以和Pod一样通过标签选择器被ServiceEntry选中和管理。

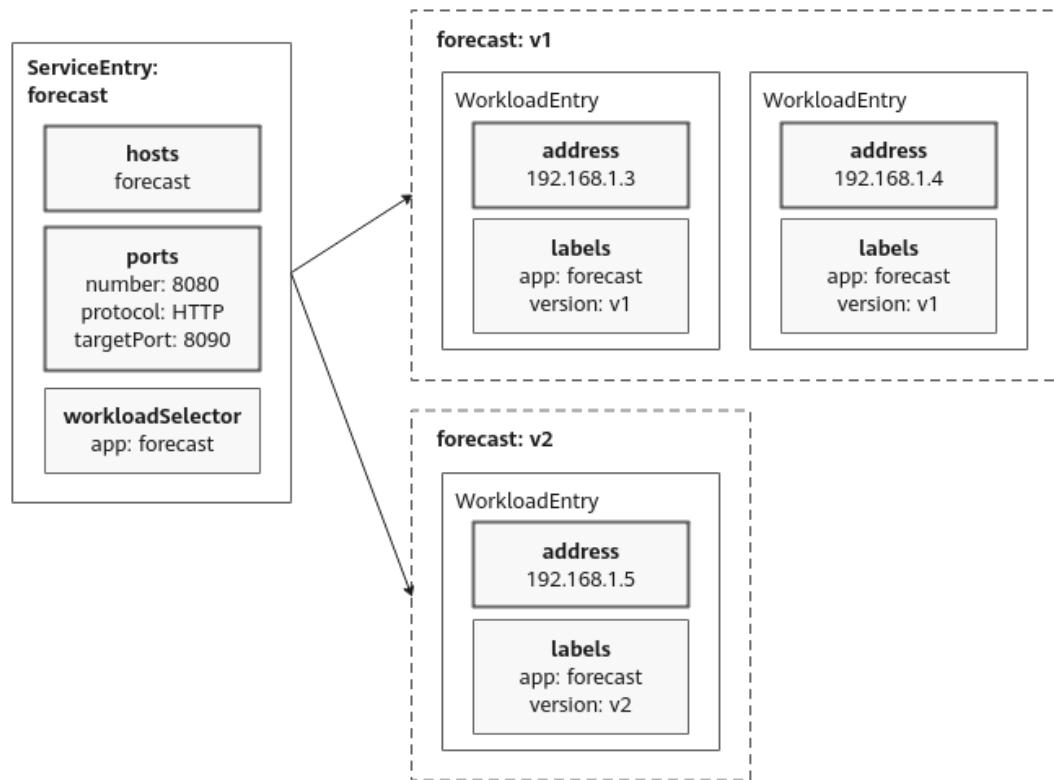


例如，ServiceEntry和三个WorkloadEntry的YAML文件如下：

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: forecast
spec:
  hosts:
  - forecast
  location: MESH_INTERNAL
  ports:
  - number: 8080
    name: http
    protocol: HTTP
    targetPort: 8090
  resolution: STATIC
  workloadSelector:
    labels:
      app: forecast
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadEntry
```

```
metadata:
  name: forecast
spec:
  address: 192.168.1.3
  labels:
    app: forecast
    version: v1
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadEntry
metadata:
  name: forecast
spec:
  address: 192.168.1.4
  labels:
    app: forecast
    version: v1
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadEntry
metadata:
  name: forecast
spec:
  address: 192.168.1.5
  labels:
    app: forecast
    version: v2
```

那么ServiceEntry选择WorkloadEntry的机制可概括如下：



## 4.2 约束与限制

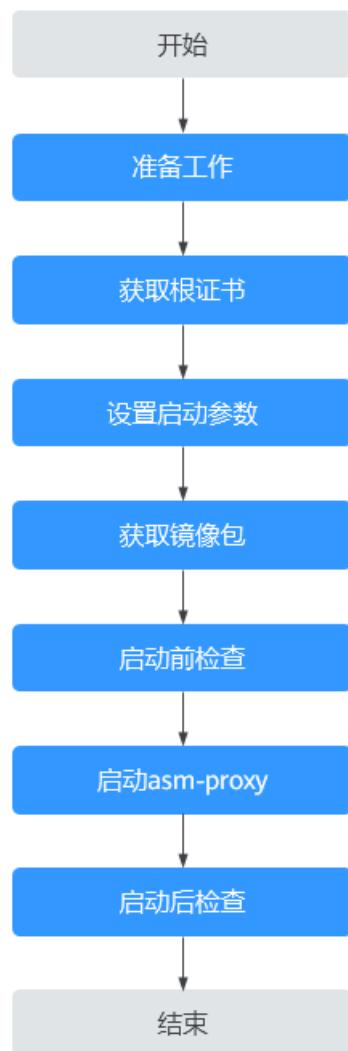
- 仅1.8企业版网格支持虚拟机治理，其他类型不支持，本文档后续章节均以1.8.4网格版本为例。
- 支持v1.19及以上版本集群。

- 支持扁平和非扁平网络架构。
- 如果是多集群，需要集群在同一个VPC，否则虚拟机与其他VPC集群网络不通。
- 不支持容器隧道网络集群。
- 不支持虚拟机和容器实例属于同一个服务。

## 4.3 部署 ASM-PROXY

为了兼容不同虚拟机操作系统的版本差异，ASM-PROXY以Docker镜像的方式进行部署。部署流程如图4-1所示。

图 4-1 ASM-PROXY 部署流程



### 准备工作

- 已购买CCE集群，已在集群中创建虚拟机服务使用的命名空间（假设为vmns），已在该命名空间创建容器服务。具体操作请参见[购买CCE集群](#)。
- 已购买企业版网格，并添加集群到网格，虚拟机服务使用的命名空间已开启sidecar注入。具体操作请参见[购买网格](#)、[添加集群](#)和[sidecar管理](#)。

- 准备两台已安装Docker的ECS虚拟机，并满足以下要求：

要求	操作方法
创建相应目录及文件	在ECS虚拟机上执行以下命令： <code>mkdir -p /opt/asm_proxy/certs/</code> <code>touch /opt/asm_proxy/asm_proxy.env</code>
备份虚拟机iptables规则	在ECS虚拟机上执行以下命令： <code>iptables-save &gt; ~/iptables.backup</code>

### ⚠ 注意

请确保网络三层连通性：

- 为了保证虚拟机与CCE集群的Pod互通，准备的虚拟机需要与CCE集群处于同一VPC内。
- 虚拟机需放通安全组，入方向需添加到<cluster-name>-cce-control安全组的规则，否则ping不通Pod IP。

## 获取根证书

### 步骤1 通过kubectl访问CCE集群。

- 登录云容器引擎控制台，在“集群管理”页面单击集群名称，进入集群详情页。
- 在“连接信息”区域找到kubectl，单击查看后，按照教程为您的客户端机器配置kubectl，并访问CCE集群。

### 📖 说明

假设配置了kubectl的客户端机器称为kubectl节点，下文将一律使用kubectl节点进行描述。

### 步骤2 在kubectl节点执行以下命令，创建临时工作目录。

```
export VM1_DIR=<a working directory for vm1>
mkdir -p ${VM1_DIR}
```

### 步骤3 在kubectl节点执行以下命令，创建服务账号。

```
export NS=<vmns>
export SERVICE_ACCOUNT=<vmsa>
kubectl create serviceaccount "${SERVICE_ACCOUNT}" -n "${NS}"
```

其中，NS为虚拟机要加入的命名空间，SERVICE\_ACCOUNT为虚拟机使用的服务账号。

### 步骤4 获取证书密钥文件。

- 在kubectl节点执行以下命令，确认集群是否已开启TokenRequest API。  
`kubectl get --raw /api/v1 | grep "serviceaccounts/token"`

如果没有返回，则需要参考[Service Account Token Volume Projection](#)开启该API。

2. 在kubectl节点执行以下命令，获取证书密钥文件。

```
export tokenexpiretime=$((3600*24*365))
```

```
1、echo '{"kind":"TokenRequest","apiVersion":"authentication.k8s.io/v1","spec":{"audiences":["istio-ca"],"expirationSeconds":'$tokenexpiretime'} }' | kubectl create --raw /api/v1/namespaces/vmns/serviceaccounts/vmsa/token -f
```

其中， $3600*24*365$ 表示1年有效期， $31536000s$ 。

3. 复制token字段的值，在kubectl节点执行以下命令，将密钥保存在istio-token文件中。

```
echo -n "{token字段的值}" > ${VM1_DIR}/istio-token
```

**步骤5** 在kubectl节点执行以下命令，获取根证书文件。

```
kubectl get secret -n istio-system cacerts -o yaml |grep "root-cert.pem" | head -1 | cut -d ":" -f 2 | sed s/[[:space:]]//g | base64 -d > ${VM1_DIR}/root-cert.pem
```

-----结束

## 设置启动参数

**步骤1** 在kubectl节点执行以下命令，获取相关参数并写入\${VM1\_DIR}目录下的asm\_proxy.env文件中。

- #### 1. 获取ASM控制面通信地址

```
ISTIO_PILOT_ADDR=$(kubectl get pod -n istio-system -o jsonpath='{.items[2].status.podIP}')
```

```
echo "ISTIO_PILOT_A
```

```
ASM CLUSTER_ID=$(kubectl get deploy istio-ingressgateway -n istio-system -o yaml | grep -A1 spec.clusterName)
```

ISTIO\_META\_

```
echo "ISTIO_META_CLUSTER_ID=${ASM_CLUSTER_ID}" >> ${VM1_DIR}/asm_proxy.env
```

ASM\_MESH\_ID

```
"${ISTIO_META_ASM_MESH_ID}" | tail -1 | awk -F\: '{print $NF}' | awk '$1=$1')  
echo "${ISTIO_META_MESH_ID}=${ASM_MESH_ID}" >> ${VM1_DIR}/asm_proxy.env
```

## 获取监控服务端地址

```
ZIPKIN_ADDRESS=$(kubectl get iop -n istio-system -o yaml | grep -A1 "zipkin" | sed -n '2p')  
TRACING_ZIPKIN_ADDRESS=$(echo ${ZIPKIN_ADDRESS#*:})
```

```
echo "TRACING_ZIPKIN_ADDRESS=${TRACING_ZIPKIN_ADDRESS:-$ZIPKIN_HOST:$ZIPKIN_PORT}"
```

#### 获取虚拟机要加入的命名空间

```
echo "NS=${NS}" >> ${VMLINUX_DIR}/a
```

### 获取虚拟机使用的服务账号

**步骤2** 打开kubectl节点的\${VM1\_DIR}/asm\_proxy.env文件，填写以下变量，保存文件。

```
# 集群服务网段，从ASM控制台“网格配置 > 基本信息”页面集群列表中获取  
ISTIO_SERVICE_CIDR=x.x.x.x/x  
  
# 虚拟机使用的Pod名称，自定义  
POD_NAME=<vmapp1>  
# 不指定代理云平台  
CLOUD_PLATFORM=none
```

**步骤3** 执行以下命令检查文件内容。

```
cat ${VM1_DIR}/asm_proxy.env
```

正常回显如下：

```
ISTIO_PILOT_ADDR=10.252.2.36  
ISTIO_META_CLUSTER_ID=cluster-vm  
ISTIO_META_MESH_ID=e256c79c-8b13-11ec-94e9-0255ac10069a  
TRACING_ZIPKIN_ADDRESS=otel-collector.monitoring.svc.cluster.local:9411  
NS=vmns  
SERVICE_ACCOUNT=vmsa  
ISTIO_SERVICE_CIDR=10.233.0.0/16  
POD_NAME=vmapp1
```

**步骤4** 将kubectl节点\${VM1\_DIR}目录下的asm\_proxy.env文件上传到虚拟机/opt/asm\_proxy/目录下。

在root下执行：

```
scp ${VM1_DIR}/asm_proxy.env root@${虚拟机ip}:~/opt/asm_proxy/
```

**步骤5** 将kubectl节点\${VM1\_DIR}目录下的istio-token、root-cert.pem文件上传到虚拟机/opt/asm\_proxy/certs目录下。

在root下执行：

```
scp ${VM1_DIR}/istio-token root@${虚拟机ip}:~/opt/asm_proxy/certs
```

```
scp ${VM1_DIR}/root-cert.pem root@${虚拟机ip}:~/opt/asm_proxy/certs
```

----结束

## 获取镜像包

**步骤1** 以root用户登录虚拟机。

**步骤2** 获取asm-proxy-1.8.4镜像包。

asm-proxy-1.8.4镜像包存放在SWR镜像仓库中，使用如下命令拉取镜像（x86和arm架构的命令相同）。

```
docker pull SWR镜像仓库地址/istio/asm-proxy:1.8.4
```

若网格版本是1.8.6-r3，则拉取镜像的地址为：SWR镜像仓库地址/istio/asm-proxy:1.8.6-r3-20230413150600；若网格版本是1.13.9-r1，则拉取镜像的地址为：SWR镜像仓库地址/istio/asm-proxy:1.13.9-r1-220230412182514。

### 说明

SWR镜像仓库地址的获取方式如下：

在容器镜像服务控制台“总览”页面单击右上角的“登录指令”，弹出“登录指令”对话框，指令末尾的域名即为SWR镜像仓库地址，形式为swr.\*\*\*.com。

----结束

## 启动前检查

**步骤1** 以root用户登录虚拟机。

**步骤2** 检查证书文件信息。

```
ls -l /opt/asm_proxy/certs/
```

以下为预期结果：

```
-rw----- 1 root root 757 Jan 24 10:41 istio-token  
-rw----- 1 root root 1789 Jan 24 10:41 root-cert.pem
```

**步骤3** 检查启动参数文件信息。

```
cat /opt/asm_proxy/asm_proxy.env
```

以下为预期结果（具体值会不同，确保不为空）：

```
STIO_PILOT_ADDR=10.252.2.36  
ISTIO_META_CLUSTER_ID=cluster-vm  
ISTIO_META_MESH_ID=e256c79c-8b13-11ec-94e9-0255ac10069a  
TRACING_ZIPKIN_ADDRESS=otel-collector.monitoring.svc.cluster.local:9411  
NS=vmns  
SERVICE_ACCOUNT=vmsa  
ISTIO_SERVICE_CIDR=10.233.0.0/16  
POD_NAME=vmapp1
```

----结束

## 启动 asm-proxy

**步骤1** 执行以下命令，启动asm-proxy容器。

```
docker run -d \  
  --name=asm-proxy \  
  --network=host \  
  --restart=always \  
  --env-file /opt/asm_proxy/asm_proxy.env \  
  --cap-add=NET_ADMIN \  
  --volume=/opt/asm_proxy/certs:/opt/istio/certs \  
  SWR镜像仓库地址/istio/asm-proxy:1.8.4
```

其中，“SWR镜像仓库地址/istio/asm-proxy:1.8.4”为asm-proxy镜像包地址，请参考[获取镜像包](#)获取。

**步骤2** 若虚拟机为arm架构，还需要执行以下步骤：

1. 执行`docker ps | grep asm-proxy`，查看asm-proxy容器的ID。

返回示例如下：

```
fad1ab9530fc    asm-proxy:1.8.4    "/usr/bin/bash /usr/..."  24 h
```

2. 执行如下命令从容器中拷贝二进制文件。

```
docker cp fad1ab9530fc:/usr/local/bin/pilot-agent /usr/local/bin
```

其中，fad1ab9530fc为上一步获取的容器ID。

3. 为二进制文件赋予可执行权限。

```
chmod 755 /usr/local/bin/pilot-agent
```

4. 定义环境变量并赋值。

```
export ISTIO_SERVICE_CIDR=x.x.x.x/x
```

其中，x.x.x.x/x为集群服务网段，从ASM控制台“网格配置 > 基本信息”页面集群列表中获取。

5. 设置虚拟机的iptables。

```
pilot-agent istio-clean-iptables -m REDIRECT
```

```
pilot-agent istio-iptables -p 15001 -z 15006 -u 1337 -m REDIRECT -i $ {ISTIO_SERVICE_CIDR} -x "" -b "*" -d 15020
```

----结束

## 启动后检查

- 步骤1** 在虚拟机上执行以下命令，检查istio的envoy进程是否正常。

```
ps -ef | grep istio
```

正常回显信息类似如下，envoy进程运行正常。

```
root 27254 27139 0 14:37 ? 00:00:00 su -s /bin/bash -c INSTANCE_IP=10.66.6.88  
POD_NAME=vmapp1 POD_NAMESPACE=vmns exec /usr/local/bin/pilot-agent proxy --serviceCluster  
vmapp1 2> /var/log/istio/istio.err.log > /var/log/istio/istio.log istio-proxy  
systemd+ 27714 27707 0 14:40 ? 00:00:00 /usr/local/bin/envoy -c etc/istio/proxy/envoy-rev0.json --  
restart-epoch 0 --drain-time-s 45 --parent-shutdown-time-s 60 --service-cluster vmapp1 --service-node  
sidecar~10.66.6.88~vmapp1.vmns~vmns.svc.cluster.local --local-address-ip-version v4 --bootstrap-version 3 --  
log-format-prefix-with-location 0 --log-format %Y-%m-%dT%T.%fZ%l?envoy %n?v -l warning --  
component-log-level misc:error  
root 27745 3521 0 14:50 pts/0 00:00:00 grep --color=auto istio
```

- 步骤2** 检查asm-proxy容器状态正常。

```
docker ps | grep asm-proxy
```

如果第5列没有提示Restarting，说明asm-proxy容器启动正常。

```
2bc3ee448c6d  asm-proxy:1.8.4  "/usr/bin/bash /usr/..."  52 seconds ago  Up 51 seconds  asm-proxy
```

启动异常显示如下：

```
root@cluster-  -vm:~# docker ps | grep asm-proxy  
3b45cced1236  asm-proxy:1.8.4  "/usr/bin/bash /usr/..."  44 seconds ago  Restarting (1) 13 seconds ago  asm-proxy  
root@cluster-  -vm:~#
```

- 步骤3** 检查istio启动正常。

1. 执行以下命令进入asm-proxy容器。

```
docker exec -it asm-proxy bash
```

2. 执行以下命令查看istio启动日志。

```
vim /var/log/istio/istio.log
```

正常启动日志如下：

```
2022-02-07T09:23:31.651059Z  info  xdsproxy  Envoy ADS stream established  
2022-02-07T09:23:31.651156Z  info  xdsproxy  connecting to upstream XDS server: istiod.istio-  
system.svc:15012  
2022-02-07T09:23:31.962409Z  info  sds  resource:ROOTCA new connection  
2022-02-07T09:23:31.962484Z  info  sds  Skipping waiting for gateway secret  
2022-02-07T09:23:31.962547Z  info  sds  resource:default new connection
```

```
2022-02-07T09:23:31.962592Z  info  sds  Skipping waiting for gateway secret
2022-02-07T09:23:31.962706Z  info  cache  adding watcher for file ./etc/certs/root-cert.pem
2022-02-07T09:23:31.962785Z  info  cache  GenerateSecret from file ROOTCA
2022-02-07T09:23:31.963252Z  info  sds  resource:ROOTCA pushed root cert to proxy
2022-02-07T09:23:32.041052Z  error  cache  cannot load key pair from /etc/certs: open /etc/certs/
cert-chain.pem: no such file or directory
2022-02-07T09:23:32.055631Z  info  cache  GenerateSecret default
2022-02-07T09:23:32.056570Z  info  sds  resource:default pushed key/cert pair to proxy
```

**步骤4** 在虚拟机上执行以下命令，确认iptables规则正确配置。

**iptables -t nat -L -v**

如果回显信息类似如下，说明iptables规则配置成功。

```
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target  prot opt in  out   source        destination
 106  4240 ISTIO_INBOUND  tcp  --  any    anywhere     anywhere

Chain INPUT (policy ACCEPT 106 packets, 4240 bytes)
pkts bytes target  prot opt in  out   source        destination

Chain OUTPUT (policy ACCEPT 207 packets, 16621 bytes)
pkts bytes target  prot opt in  out   source        destination
  50  3000 ISTIO_OUTPUT  tcp  --  any    any    anywhere     anywhere

Chain POSTROUTING (policy ACCEPT 207 packets, 16621 bytes)
pkts bytes target  prot opt in  out   source        destination

Chain ISTIO_INBOUND (1 references)
pkts bytes target  prot opt in  out   source        destination
  0   0 RETURN  tcp  --  any    any    anywhere     anywhere
tcp dpt:15008
  0   0 RETURN  tcp  --  any    any    anywhere     anywhere      tcp dpt:ssh
  0   0 RETURN  tcp  --  any    any    anywhere     anywhere      tcp dpt:15020
 106  4240 ISTIO_IN_REDIRECT  tcp  --  any    any    anywhere     anywhere

Chain ISTIO_IN_REDIRECT (3 references)
pkts bytes target  prot opt in  out   source        destination
 106  4240 REDIRECT  tcp  --  any    any    anywhere     anywhere      redir ports 15006

Chain ISTIO_OUTPUT (1 references)
pkts bytes target  prot opt in  out   source        destination
 0 ++++++
  0 RETURN  all  --  any    lo    127.0.0.6      anywhere
  0   0 ISTIO_IN_REDIRECT  all  --  any    lo    anywhere      !localhost      owner UID match 1337
  0   0 RETURN  all  --  any    lo    anywhere      anywhere      ! owner UID match 1337
  0   0 RETURN  all  --  any    any    anywhere     anywhere      owner UID match 1337
  0   0 ISTIO_IN_REDIRECT  all  --  any    lo    anywhere      !localhost      owner GID match 1337
  0   0 RETURN  all  --  any    lo    anywhere     anywhere      ! owner GID match 1337
  0   0 RETURN  all  --  any    any    anywhere     anywhere      owner GID match 1337
  0   0 RETURN  all  --  any    any    anywhere     localhost
  0   0 ISTIO_REDIRECT  all  --  any    any    anywhere     10.228.0.0/16
  50  3000 RETURN  all  --  any    any    anywhere     anywhere

Chain ISTIO_REDIRECT (1 references)
pkts bytes target  prot opt in  out   source        destination
  0   0 REDIRECT  tcp  --  any    any    anywhere     anywhere      redir ports 15001
```

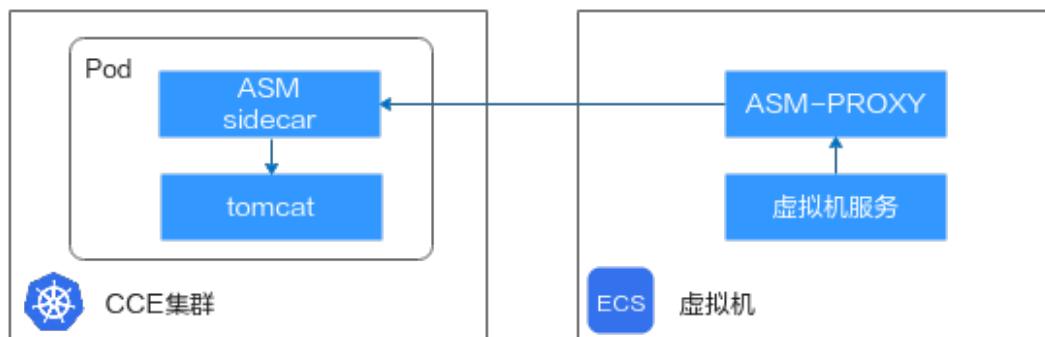
其中，10.228.0.0/16为集群服务网段。

----结束

## 4.4 验证服务访问

## 4.4.1 虚拟机服务访问容器服务

启动ASM-PROXY后，虚拟机服务可以访问容器内的服务，如下图所示。



验证流程如下：

1. **部署容器服务**：在CCE集群中部署容器服务tomcat。
2. **容器服务加入网格**：将tomcat服务加入网格，确保服务诊断状态为正常。
3. **访问容器服务**：编辑虚拟机的/etc/hosts文件，添加域名解析，然后通过curl命令访问容器服务。

## 部署容器服务

**步骤1** 登录云容器引擎控制台，在“集群管理”页面单击集群名称，进入集群详情页。

**步骤2** 在左侧导航栏选择“资源 > 工作负载”，单击右上角“镜像创建”。

**步骤3** 设置工作负载参数。

- 负载类型：选择“无状态负载”。
- 命名空间：选择在**步骤3**中填写的命名空间“vmns”。
- 镜像：选择“tomcat”。
- 服务配置：添加服务，服务端口和容器端口设置为8080。

其余参数按需填写。

### 说明

如果是TCP协议服务，需要服务监听在127.0.0.1或0.0.0.0。

**步骤4** 单击“创建工作负载”。

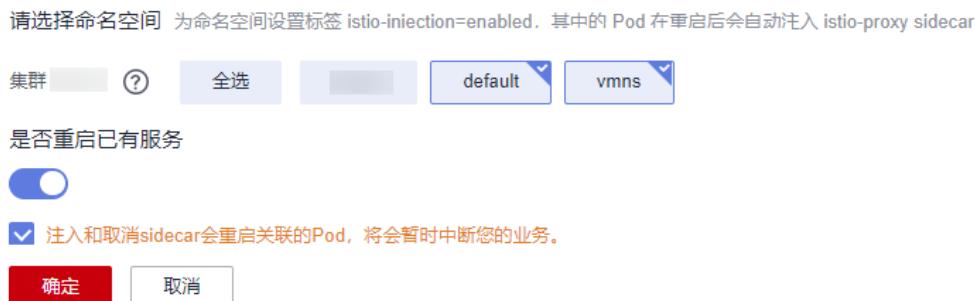
----结束

## 容器服务加入网格

**步骤1** 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。

**步骤2** 在左侧导航栏选择“网格配置”，单击“sidecar管理”页签，为命名空间“vmns”注入sidecar。

图 4-2 注入 sidecar



步骤3 在左侧导航栏选择“服务管理”，选择命名空间“vmns”。

步骤4 按下述指导修复异常状态的tomcat服务。

单击“处理”进入自动修复项页面，单击“一键修复”，自动处理异常状态的检查项，直到配置诊断为正常状态。

图 4-3 服务状态



----结束

## 访问容器服务

步骤1 获取tomcat服务在集群中的IP地址。

1. 登录云容器引擎控制台，在“集群管理”页面单击集群名称，进入集群详情页。
2. 在左侧导航栏选择“资源 > 工作负载”，选择命名空间“vmns”。
3. 单击工作负载名称“tomcat”，进入详情页面。
4. 单击“访问方式”页签，获取服务在集群中的IP地址。

图 4-4 IP 地址



步骤2 登录虚拟机，执行`vim /etc/hosts`打开本地hosts文件，添加域名解析。

<cluster\_ip> <tomcat>.<vmns>.svc

其中，cluster\_ip为步骤1中获取的IP地址，tomcat为容器服务名称，vmns为命名空间。

**步骤3** 执行以下命令，验证访问容器服务是否正常。

```
curl <tomcat>.<vmns>.svc:8080
```

如果回显信息类似如下，说明虚拟机服务访问容器服务正常。

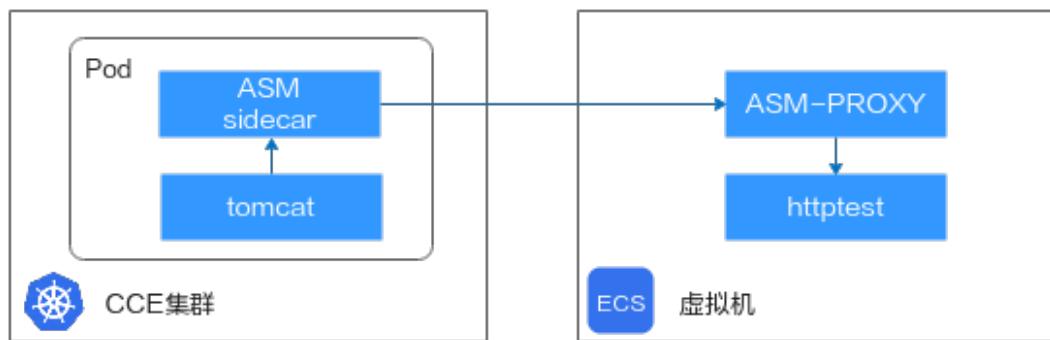
```
root@...:~# curl tomcat.vmns.svc:8080

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Apache Tomcat/8.5.32</title>
    <link href="favicon.ico" rel="icon" type="image/x-icon" />
    <link href="favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <link href="tomcat.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div id="wrapper">
      <div id="navigation" class="curved container">
        <span id="nav-home"><a href="http://tomcat.apache.org/">Home</a></span>
```

----结束

#### 4.4.2 容器服务访问虚拟机服务

启动ASM-PROXY后，容器内的服务可以访问虚拟机上的服务，如下图所示。



验证流程如下：

1. **部署虚拟机服务**：在虚拟机中部署httpptest应用。
2. **部署容器服务**：在CCE集群中部署容器服务tomcat。
3. **添加虚拟机服务到网格**：不同于容器服务有自动注册能力，当把一个虚拟机服务加入网格时，需要用户手动注册服务的基础信息。因此需要手动创建如下资源：Service、WorkloadEntry、ServiceEntry、DestinationRule和VirtualService。
4. **访问虚拟机服务**：容器服务通过服务名访问虚拟机服务。

#### 部署虚拟机服务

此处以httpptest测试程序作为示例。httpptest为go http应用，监听在8080端口，访问正常返回“hello, http protocol server”消息。

**步骤1** 上传httpptest应用到虚拟机。

**步骤2** 修改权限。

```
chmod +x httpptest
```

**步骤3 启动httptest应用。**

**./httptest**

**----结束**

## 部署容器服务

**步骤1** 登录云容器引擎控制台，在“集群管理”页面单击集群名称，进入集群详情页。

**步骤2** 在左侧导航栏选择“资源 > 工作负载”，单击右上角“镜像创建”。

**步骤3** 设置工作负载参数。

- 负载类型：选择“无状态负载”。
- 命名空间：选择在**步骤3**中填写的命名空间“vmns”。
- 镜像：选择“tomcat”。
- 服务配置：添加服务，服务端口和容器端口设置为8080。

其余参数按需填写。

### 说明

如果是TCP协议服务，需要服务监听在127.0.0.1或0.0.0.0。

**步骤4** 单击“创建工作负载”。

**----结束**

## 添加虚拟机服务到网格

**步骤1** 创建Service。

1. 登录云容器引擎控制台，在“集群管理”页面单击集群名称，进入集群详情页。
2. 在左侧导航栏选择“资源 > 服务发现”，选择命名空间“vmns”，单击右上角“YAML创建”。
3. 输入以下内容：

```
apiVersion: v1
kind: Service
metadata:
  name: <vm-server1>
  namespace: "<NS>"
  labels:
    app: <vm-server1>
spec:
  ports:
    - port: 8080
      name: http-<vm-server1>
      targetPort: 8080
  selector:
    app: <vm-server1>
```

其中，<vm-server1>为虚拟机服务名称，<NS>为**步骤3**中填写的命名空间。

示例：

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: vm-server1
5   namespace: "vmns"
6   labels:
7     app: vm-server1
8 spec:
9   ports:
10    - port: 8080
11      name: http-vm-server1
12      targetPort: 8080
13   selector:
14     app: vm-server1|
```

4. 单击“确定”完成创建。

## 步骤2 创建WorkloadEntry。

使用WorkloadEntry定义虚拟机服务的工作负载属性。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源：workloadentries”，命名空间选择**步骤3**中填写的命名空间“vmns”，单击“创建”。

### 说明

这里的“vmns”命名空间中必须存在服务在网格中正常使用，否则选择不到该命名空间。  
你可以将[部署容器服务](#)中的tomcat服务加入网格，并确保配置诊断通过。

4. 输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadEntry
metadata:
  name: "<vm-server1>"
  namespace: "<NS>"
spec:
  address: "<VM_IP>"
  labels:
    app: <vm-server1>
    version: v1
  serviceAccount: "<SERVICE_ACCOUNT>"
```

其中，<vm-server1>为虚拟机服务名称，<NS>为**步骤3**中填写的命名空间，<VM\_IP>为虚拟机的私有IP，<SERVICE\_ACCOUNT>为**步骤3**中填写的服务账号名称。

示例：

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: WorkloadEntry
3 metadata:
4   name: "vm-server1"
5   namespace: "vmns"
6 spec:
7   address: "10.66 [REDACTED]"
8   labels:
9     app: vm-server1
10    version: v1
11   serviceAccount: "vmsa"
```

5. 单击“确定”完成创建。

## 步骤3 创建ServiceEntry。

使用ServiceEntry将虚拟机服务添加到网格内部维护的服务注册表，以便网格中自动发现的服务可以访问或路由到这些手动指定的服务。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源：serviceentries”，命名空间选择**步骤3**中填写的命名空间“vmns”，单击“创建”。
4. 输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: "<vm-server1>"
  namespace: "<NS>"
spec:
  hosts:
    - <vm-server1>.<NS>.svc.cluster.local
  location: MESH_INTERNAL
  ports:
    - number: 8080
      name: http
      protocol: HTTP
      resolution: STATIC
  workloadSelector:
    labels:
      app: <vm-server1>
```

其中，<vm-server1>为虚拟机服务名称，<NS>为**步骤3**中填写的命名空间。

示例：

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: ServiceEntry
3 metadata:
4   name: "vm-server1"
5   namespace: "vmns"
6 spec:
7   hosts:
8     - vm-server1.vmns.svc.cluster.local
9   location: MESH_INTERNAL
10  ports:
11    - number: 8080
12      name: http
13      protocol: HTTP
14      resolution: STATIC
15      workloadSelector:
16        labels:
17          app: vm-server1
```

5. 单击“确定”完成创建。

#### 步骤4 创建DestinationRule。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源：destinationrules”，命名空间选择**步骤3**中填写的命名空间“vmns”，单击“创建”。
4. 输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: <vm-server1>
  namespace: "<NS>"
spec:
  host: <vm-server1>
  subsets:
    - labels:
```

```
version: v1  
name: v1
```

其中，<vm-server1>为虚拟机服务名称，<NS>为**步骤3**中填写的命名空间。

示例：

```
1 apiVersion: networking.istio.io/v1alpha3  
2 kind: DestinationRule  
3 metadata:  
4   name: vm-server1  
5   namespace: "vmns"  
6 spec:  
7   host: vm-server1  
8   subsets:  
9     - labels:  
10       version: v1  
11       name: v1  
12
```

5. 单击“确定”完成创建。

#### 步骤5 创建VirtualService。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源：virtualservices”，命名空间选择**步骤3**中填写的命名空间“vmns”，单击“创建”。
4. 输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3  
kind: VirtualService  
metadata:  
  name: <vm-server1>-8080  
  namespace: "<NS>"  
spec:  
  hosts:  
    - <vm-server1>  
  http:  
    - match:  
      - gateways:  
      - mesh  
      port: 8080  
    route:  
      - destination:  
        host: <vm-server1>.<NS>.svc.cluster.local  
        port:  
          number: 8080  
        subset: v1
```

其中，<vm-server1>为虚拟机服务名称，8080为服务端口，<NS>为**步骤3**中填写的命名空间。

示例：

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: VirtualService
3 metadata:
4   name: vm-server1-8080
5   namespace: "vmns"
6 spec:
7   hosts:
8     - vm-server1
9   http:
10    - match:
11      - gateways:
12        - mesh
13      port: 8080
14    route:
15      - destination:
16        host: vm-server1.vmns.svc.cluster.local
17        port:
18          number: 8080
19        subset: v1
```

5. 单击“确定”完成创建。

----结束

## 访问虚拟机服务

**步骤1** 在kubectl节点执行以下命令，获取tomcat工作负载的Pod名称。

```
kubectl get pods -n <vmns>|grep tomcat
```

回显示例：

```
tomcat-75ccb4b948-nzrfs  2/2  Running  0  19h
```

**步骤2** 执行以下命令，测试访问虚拟机服务。

```
kubectl exec -it <tomcat-xxx> -n <vmns> -- curl <vm-
server1>.<vmns>.svc.cluster.local:8080
```

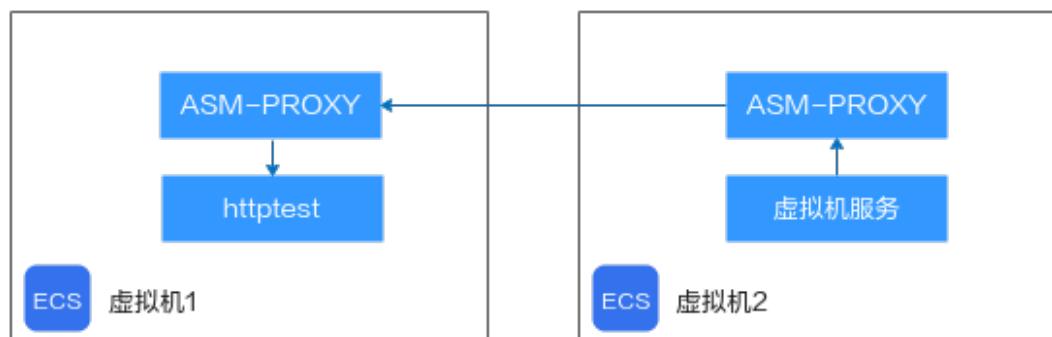
其中，<vm-server1>为虚拟机服务名称，<vmns>为**步骤3**中填写的命名空间。

若回显“hello, http protocol server”，则容器服务访问虚拟机服务成功。

----结束

### 4.4.3 虚拟机服务访问虚拟机服务

启动ASM-PROXY后，虚拟机服务之间可以互相访问，如下图所示。



## 验证流程

**步骤1** 准备两台ECS虚拟机。

虚拟机1请参考[部署ASM-PROXY](#)部署PROXY，虚拟机2可以不用部署。但请确保虚拟机间网络正常，且安全组、防火墙已放通。

**步骤2** 在虚拟机1上部署httptest应用。

具体操作请参见[部署虚拟机服务](#)。

**步骤3** 添加虚拟机1上的服务到网格。

具体操作请参见[添加虚拟机服务到网格](#)。

**步骤4** 虚拟机2访问虚拟机1上的服务。

1. 登录虚拟机2，执行`vim /etc/hosts`添加本地域名解析。

`10.66.xx.xx <vm-server1>.<vmns>.svc.cluster.local`

其中，10.66.xx.xx为虚拟机1的私有IP，<vm-server1>为虚拟机1上的服务名称，<vmns>为**步骤3**中填写的命名空间。

2. 执行以下命令，访问虚拟机1上的服务。

`curl <vm-server1>.<vmns>.svc.cluster.local:8080`

若回显“hello, http protocol server”，则虚拟机2访问虚拟机1上的服务成功。

----结束

## 4.5 流量治理

### 4.5.1 虚拟机服务添加网关和路由

网关（Gateway）定义了在网格出入口操作的负载均衡器，用于接收传入或传出的HTTP/TCP连接。

#### 前提条件

- 已执行[添加虚拟机服务到网格](#)，即已创建v1版本的WorkloadEntry、ServiceEntry，已创建VirtualService、DestinationRule。
- 已创建负载均衡实例，要求所属VPC为集群所在的VPC。

#### 创建 Loadbalancer 服务

**步骤1** 登录云容器引擎控制台，在“集群管理”页面单击集群名称，进入集群详情页。

**步骤2** 在左侧导航栏选择“资源 > 服务发现”，选择命名空间“istio-system”，单击右上角“YAML创建”。

**步骤3** 输入以下内容：

```
apiVersion: v1
kind: Service
metadata:
  name: <vm-server1>-gw-svc
  namespace: istio-system
  annotations:
    kubernetes.io/elb.class: <ELB_TYPE>
```

```
kubernetes.io/elb.id: <ELB_ID>
service.protocol.kubernetes.io/type: LoadBalancer
spec:
ports:
- name: http
  protocol: TCP
  port: <345>
  targetPort: <2050>
selector:
  app: istio-ingressgateway
  istio: ingressgateway
type: LoadBalancer
sessionAffinity: None
loadBalancerIP: <ELB_IP>
externalTrafficPolicy: Cluster
```

其中，

- <vm-server1>为虚拟机服务名称
- <ELB\_TYPE>为负载均衡类型，可设置为union（共享型）或performance（独享型）
- <ELB\_ID>为ELB实例ID
- <345>为服务port
- <2050>为服务targetPort

如果port小于1025，targetPort需要设置为未被占用且范围为1025~65535的值；如果port大于等于1025，则targetPort可直接设置为port值。

#### 说明

判断targetPort是否被占用的方法：

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
  2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
  3. 选择“istio资源：gateways”，命名空间选择“istio-system”，查看Gateway资源的YAML文件中server的port.number值。
- <ELB\_IP>为ELB实例的弹性IP

**步骤4** 单击“确定”完成创建。

----结束

## 创建 Gateway 资源

**步骤1** 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。

**步骤2** 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。

**步骤3** 选择“istio资源：gateways”，命名空间选择“istio-system”，单击“创建”。

**步骤4** 输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  annotations:
    asm/elbPort: '<345>'
    asm/gatewaySvc: <vm-server1>-gw-svc.istio-system
  name: <vm-server1>-gw
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
```

```
servers:  
  - hosts:  
    - <ELB_IP>  
    port:  
      name: http  
      number: <2050>  
      protocol: http
```

其中，<vm-server1>为虚拟机服务名称，<345>为对外端口（同[创建Loadbalancer服务](#)中的服务port），<ELB\_IP>为ELB实例的公网IP或域名，<2050>同[创建Loadbalancer服务](#)中的服务targetPort。

**步骤5** 单击“确定”完成创建。

----结束

## 创建 VirtualService 资源

**步骤1** 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。

**步骤2** 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。

**步骤3** 选择“istio资源：virtualservices”，命名空间选择“istio-system”，单击“创建”，输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3  
kind: VirtualService  
metadata:  
  name: <vm-server1>-root-vs  
  namespace: istio-system  
spec:  
  gateways:  
    - istio-system/<vm-server1>-gw  
  hosts:  
    - <ELB_IP>  
  http:  
    - delegate:  
        name: <vm-server1-8080>  
        namespace: <vmns>  
        match:  
          - uri:  
              prefix: /test  
    - delegate:  
        name: <vm-server1-8080>  
        namespace: <vmns>  
        match:  
          - uri:  
              prefix: /
```

其中，<vm-server1>为虚拟机服务名称，<ELB\_IP>为ELB实例的公网IP或域名，<vm-server1-8080>同[创建VirtualService](#)中的VirtualService name，<vmns>同[创建VirtualService](#)中的VirtualService namespace。

### 说明

上述http.delegate列表配置了“/”和“/test”两个路由，请根据需要添加或者删除。

**步骤4** 选择“istio资源：virtualservices”，命名空间选择“istio-system”，单击“创建”再创建一个VirtualService资源，输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3  
kind: VirtualService  
metadata:  
  name: <vm-server1>-mesh-root-vs  
  namespace: istio-system  
spec:
```

```
hosts:
  - <vm-server1>.<vmns>.svc.cluster.local
http:
  - delegate:
      name: <vm-server1-8080>
      namespace: <vmns>
    match:
      - uri:
          prefix: /test
  - delegate:
      name: <vm-server1-8080>
      namespace: <vmns>
    match:
      - uri:
          prefix: /
```

其中，<vm-server1>为虚拟机服务名称，<vm-server1-8080>同[创建VirtualService](#)中的VirtualService name，<vmns>同[创建VirtualService](#)中的VirtualService namespace。

#### □□ 说明

上述http.delegate列表配置了“/”和“/test”两个路由，请根据需要添加或者删除。

#### ----结束

## 更新 VirtualService 资源

**步骤1** 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。

**步骤2** 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。

**步骤3** 选择“istio资源：virtualservices”，命名空间选择“vmns”，选择[创建VirtualService](#)中创建的VirtualService，单击操作列的“编辑”，进行如下修改。

1. 移除hosts字段配置

2. 复制spec.http[0]后，在下方并列位置添加如下内容：

```
- match:
  - gateways:
    - istio-system/<vm-server1>-gw
    port: <2050>
  route:
    - destination:
        host: <vm-server1>.<vmns>.svc.cluster.local
        port:
          number: 8080
        subset: v1
```

其中，<vm-server1>为虚拟机服务名称，<2050>同[创建Loadbalancer服务](#)中的服务targetPort，<vmns>为**步骤3**中填写的命名空间。

示例：下图红框中为所添加的内容，在结构上需要与spec.http[0]对齐。

```
16   name: vm-server1-8080
17   namespace: vmns
18   resourceVersion: '1488103'
19   selfLink: >-
20     /apis/networking.istio.io/v1alpha3/namespaces/vmns/virtualservices/vm-server1-8080
21   uid: 99c05147-7fec-4876-9401-ab88a2580226
22 spec:
23   http:
24     - match:
25       - gateways:
26         - mesh
27         port: 8080
28       route:
29         - destination:
30           host: vm-server1.vmns.svc.cluster.local
31           port:
32             number: 8080
33             subset: v1
34     - match:
35       - gateways:
36         - istio-system/vm-server1-gw
37         port: 2050
38       route:
39         - destination:
40           host: vm-server1.vmns.svc.cluster.local
41           port:
42             number: 8080
43             subset: v1
```

步骤4 单击“确定”完成修改。

----结束

#### 说明

如果虚拟机服务需要移除网关，请删除以上创建的资源对象，移除添加的字段即可。

## 结果验证

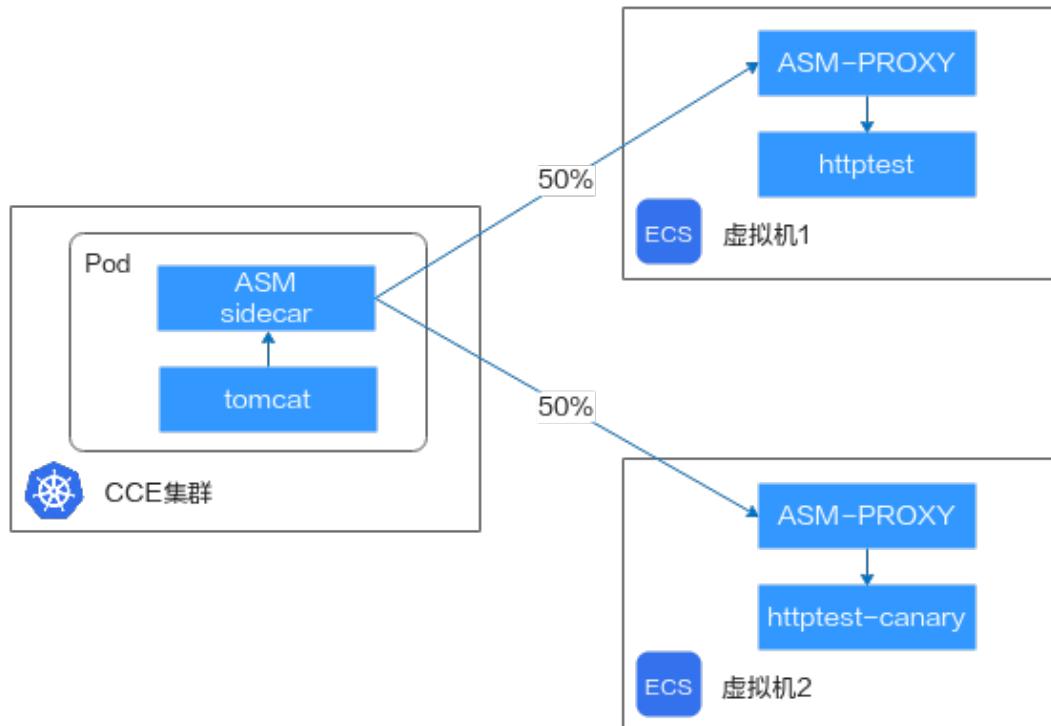
登录ASM管理控制台，在“网关管理”页面查看网关状态为“配置正常”，在浏览器打开访问地址，返回“hello, http protocol server”。

图 4-5 网关管理

网关名称	状态	所属集群	访问地址	对端协议	转发策略个数	操作	
vm-server1-gw	配置正常	test-dl	8080	HTTP	2	添加路由   编辑	
	域名	URL匹配规则	URL	命名空间	目标服务	服务端口	操作
	精确匹配	test		vmns	vm-server1	8080	详情   删除
	前缀匹配	/		vmns	vm-server1	8080	详情   删除

## 4.5.2 虚拟机服务灰度发布

虚拟机上部署ASM-PROXY，且网格化后，可以设置灰度策略实现简单的灰度发布。如下图所示，虚拟机1上部署httptest应用（原版本v1），虚拟机2上部署httptest-canary应用（灰度版本v2），配置v1版本、v2版本分别50%的流量比例。当容器服务访问虚拟机服务时，50%的请求将由v1版本响应，50%的请求由v2版本响应。



实现灰度发布的流程如下：

1. **部署虚拟机服务灰度版本**：在虚拟机2上部署httptest-canary应用，作为虚拟机服务灰度版本v2。
2. **添加虚拟机服务灰度版本到网格**：创建v2版本的WorkloadEntry、ServiceEntry，更新VirtualService和DestinationRule，并在其中配置基于流量比例的灰度策略。
3. **访问虚拟机服务**：容器服务通过服务名访问虚拟机服务，验证灰度策略是否生效。
4. **移除虚拟机服务灰度版本**：完成灰度发布后，更新VirtualService和DestinationRule，移除虚拟机服务灰度版本。

## 前提条件

- 已执行**添加虚拟机服务到网格**，即已创建v1版本的WorkloadEntry、ServiceEntry，已创建VirtualService、DestinationRule。
- 已准备虚拟机2，用来部署虚拟机服务灰度版本。

## 部署虚拟机服务灰度版本

此处以httptest-canary测试程序作为示例。httptest-canary为go http应用，监听在8080端口，访问正常返回“hello, canary http server”消息。

**步骤1** 上传httptest-canary应用到虚拟机2。

**步骤2** 修改权限。

```
chmod +x httptest-canary
```

**步骤3** 启动httptest-canary应用。

```
./httptest-canary
```

----结束

## 添加虚拟机服务灰度版本到网格

### 步骤1 创建v2版本的WorkloadEntry。

使用WorkloadEntry定义虚拟机服务的工作负载属性。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源： workloadentries”，命名空间选择**步骤3**中填写的命名空间“vmns”，单击“创建”。
4. 输入以下内容：

```
apiVersion: networking.istio.io/v1alpha3
kind: WorkloadEntry
metadata:
  name: "<vm-server1>-v2"
  namespace: "<NS>"
spec:
  address: "<VM2_IP>"
  labels:
    app: <vm-server1>
    version: v2
  serviceAccount: "<SERVICE_ACCOUNT>"
```

其中，<vm-server1>为虚拟机服务名称，<NS>为**步骤3**中填写的命名空间，<VM2\_IP>为虚拟机2的私有IP，<SERVICE\_ACCOUNT>为**步骤3**中填写的服务账号名称。

示例：

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: WorkloadEntry
3 metadata:
4   name: "vm-server1-v2"
5   namespace: "vmns"
6 spec:
7   address: "10.66.1.10"
8   labels:
9     app: vm-server1
10    version: v2
11   serviceAccount: "vmsa"
```

5. 单击“确定”完成创建。

### 步骤2 更新DestinationRule。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源： destinationrules”，命名空间选择“vmns”，选择**创建 DestinationRule**中创建的DestinationRule，单击操作列的“编辑”，在spec.subsets下将原有列表复制后并列添加以下内容：

```
- labels:
  version: v2
  name: v2
```

示例：

```
20   selfLink: >-
21     /apis/networking.istio.io/v1alpha3/namespaces/vmns/destinationrules/vm-server1
22   uid: 64ae123b-3268-43de-9f8f-59a53798dc94
23 spec:
24   host: vm-server1
25   subsets:
26     - labels:
27       version: v1
28       name: v1
29     - labels:
30       version: v2
31       name: v2
32
```

4. 单击“确定”完成修改。

**步骤3** 更新VirtualService。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源：virtualservices”，命名空间选择“vmns”，选择**创建VirtualService**中创建的VirtualService，单击操作列的“编辑”，区分两种情况：

- 情况一：虚拟机服务未配置网关

复制spec.http[0].route下的destination后，在下方并列位置添加以下内容：

```
weight: 50
- destination:
  host: <vm-server1>.<vmns>.svc.cluster.local
  port:
    number: 8080
  subset: v2
  weight: 50
```

其中，<vm-server1>为虚拟机服务名称，<vmns>为**步骤3**中填写的命名空间。配置v1、v2版本分别50%的流量比例。

示例：

```
20   selfLink: >-
21     /apis/networking.istio.io/v1alpha3/namespaces/vmns/virtualservices/vm-server1-8080
22   uid: 4d653fb8-62bc-4462-a159-07bd8445138d
23 spec:
24   hosts:
25     - vm-server1
26   http:
27     - match:
28       - gateways:
29         - mesh
30         port: 8080
31     route:
32       - destination:
33         host: vm-server1.vmns.svc.cluster.local
34         port:
35           number: 8080
36         subset: v1
37         weight: 50
38       - destination:
39         host: vm-server1.vmns.svc.cluster.local
40         port:
41           number: 8080
42         subset: v2
43         weight: 50
```

- 情况二：虚拟机服务已配置网关

复制spec.http[0].route下的destination后，在下方并列位置添加以下内容：

```
weight: 50
- destination:
  host: <vm-server1>.<vmns>.svc.cluster.local
  port:
    number: 8080
```

```
subset: v2  
weight: 50
```

复制spec.http[1].route下的destination后，在下方并列位置添加以下内容：

```
weight: 50  
- destination:  
  host: <vm-server1>.<vmns>.svc.cluster.local  
  port:  
    number: 8080  
  subset: v2  
  weight: 50
```

其中，<vm-server1>为虚拟机服务名称，<vmns>为**步骤3**中填写的命名空间。配置v1、v2版本分别50%的流量比例。

示例：

```
16   name: vm-server1-8080  
17   namespace: vmns  
18   resourceVersion: '1511871'  
19   selflink: >-  
20     /apis/networking.istio.io/v1alpha3/namespaces/vmns/virtualservices/vm-server1-8080  
21   uid: 99c05147-7fec-4876-9401-ab88a2580226  
22 spec:  
23   http:  
24     - match:  
25       - gateways:  
26         - mesh  
27         port: 8080  
28       route:  
29         - destination:  
30           host: vm-server1.vmns.svc.cluster.local  
31           port:  
32             number: 8080  
33           subset: v1  
34           weight: 50  
35         - destination:  
36           host: vm-server1.vmns.svc.cluster.local  
37           port:  
38             number: 8080  
39           subset: v2  
40           weight: 50  
41     - match:  
42       - gateways:  
43         - istio-system/vm-server1-gw  
44         port: 2022  
45       route:  
46         - destination:  
47           host: vm-server1.vmns.svc.cluster.local  
48           port:  
49             number: 8080  
50           subset: v1  
51           weight: 50  
52         - destination:  
53           host: vm-server1.vmns.svc.cluster.local  
54           port:  
55             number: 8080  
56           subset: v2  
57           weight: 50
```

4. 单击“确定”完成修改。

----结束

## 访问虚拟机服务

### 说明

如果虚拟机未部署ASM-PROXY，未加入网格，要想正常访问到v2版本，需要配置关闭mTLS认证，即在vmns命名空间下的DestinationRule资源对象的spec下添加如下内容：

```
trafficPolicy:  
  tls:  
    mode: DISABLE  
  
28  spec:  
29    host: vm-server1  
30    subsets:  
31      - labels:  
32        version: v1  
33        name: v1  
34      - labels:  
35        version: v2  
36        name: v2  
37    trafficPolicy:  
38    tls:  
39      mode: DISABLE
```

**步骤1** 在CCE集群中部署容器服务tomcat，具体操作请参见[部署容器服务](#)。

**步骤2** 在kubectl节点执行以下命令，获取tomcat工作负载的Pod名称。

```
kubectl get pods -n <vmns>|grep tomcat
```

回显示例：

```
tomcat-75ccb4b948-nzrfs    2/2    Running  0   19h
```

**步骤3** 多次执行以下命令，对虚拟机服务发起访问。

```
kubectl exec -it <tomcat-xxx> -n <vmns> -- curl <vm-server1>.<vmns>.svc.cluster.local:8080
```

其中，<vm-server1>为虚拟机服务名称，<vmns>为**步骤3**中填写的命名空间。

若交替回显“hello, http protocol server”和“hello, canary http server”，则访问虚拟机服务成功，并且配置的灰度策略生效。

```
[root@cluster] # kubectl exec -it tomcat-75ccb4b948-nzrfs -n vmns -- curl vm-server1.vmns.svc.cluster.local:8080  
Defaulting container name to container-1.  
Use 'kubectl describe pod/tomcat-75ccb4b948-nzrfs -n vmns' to see all of the containers in this pod.  
hello, canary http server  
[root@cluster] #  
[root@cluster] # kubectl exec -it tomcat-75ccb4b948-nzrfs -n vmns -- curl vm-server1.vmns.svc.cluster.local:8080  
Defaulting container name to container-1.  
Use 'kubectl describe pod/tomcat-75ccb4b948-nzrfs -n vmns' to see all of the containers in this pod.  
hello, http protocol server
```

----结束

## 移除虚拟机服务灰度版本

**步骤1** 更新VirtualService。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源：virtualservices”，命名空间选择“vmns”，选择[创建VirtualService](#)中创建的VirtualService，单击操作列的“编辑”，区分两种情况：
  - 情况一：虚拟机服务未配置网关

删除spec.http[0].route下v2版本的destination配置，以及v1版本的weight参数：

```
weight: 50
- destination:
  host: <vm-server1>.<vmns>.svc.cluster.local
  port:
    number: 8080
  subset: v2
  weight: 50
```

删除后配置如下：

```
23 spec:
24   hosts:
25     - vm-server1
26   http:
27     - match:
28       - gateways:
29         - mesh
30       port: 8080
31     route:
32       - destination:
33         host: vm-server1.vmns.svc.cluster.local
34         port:
35           number: 8080
36         subset: v1|
```

- 情况二：虚拟机服务已配置网关

分别删除spec.http[0].route和spec.http[1].route下v2版本的destination配置，以及v1版本的weight参数：

```
weight: 50
- destination:
  host: <vm-server1>.<vmns>.svc.cluster.local
  port:
    number: 8080
  subset: v2
  weight: 50
```

删除后配置如下：

```
23 spec:
24   hosts:
25     - vm-server1
26   http:
27     - match:
28       - gateways:
29         - mesh
30       port: 8080
31     route:
32       - destination:
33         host: vm-server1.vmns.svc.cluster.local
34         port:
35           number: 8080
36         subset: v1
37     - match:
38       - gateways:
39         - istio-system/whgw
40       port: 1026
41     route:
42       - destination:
43         host: vm-server1.vmns.svc.cluster.local
44         port:
45           number: 8080
46         subset: v1|
```

4. 单击“确定”完成修改。

#### 步骤2 更新DestinationRule。

1. 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
2. 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
3. 选择“istio资源：destinationrules”，命名空间选择“vmns”，选择**创建**  
**DestinationRule**中创建的DestinationRule，单击操作列的“编辑”，删除  
spec.subsets下v2版本配置：

```
- labels:  
  version: v2  
  name: v2
```

删除后配置如下：

```
23 spec:  
24   host: vm-server1  
25   subsets:  
26     - labels:  
27       version: v1  
28       name: v1
```

4. 单击“确定”完成修改。

----结束

### 4.5.3 虚拟机服务配置故障注入

故障注入是一种评估系统可靠性的有效方法，故意在待测试的系统中引入故障，从而测试其健壮性和应对故障的能力。有两种类型的故障注入，可以在转发前延迟（delay）请求，模拟缓慢的网络或过载的服务，也可以中断（abort）HTTP请求，并返回一个特定的HTTP状态码给调用者。通过中断，可以模拟一个有故障的上游服务。

如下图所示，注入一个指定的HTTP状态码（如599）后，对于访问的客户端来说，就跟服务端发生异常一样。



本文以配置中断故障为例，其他治理策略的配置方法可参考[Istio官网](#)。

#### 前提条件

已执行[添加虚拟机服务到网格](#)，即已创建v1版本的WorkloadEntry、ServiceEntry，已创建VirtualService、DestinationRule。

#### 更新 VirtualService

步骤1 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。

**步骤2** 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。

**步骤3** 选择“istio资源: virtualservices”，命名空间选择“vmns”，选择**创建VirtualService**中创建的VirtualService，单击操作列的“编辑”，在spec.http[0]下添加以下内容：

```
fault:  
  abort:  
    httpStatus: 599  
    percentage:  
      value: 100
```

```
23 spec:  
24   hosts:  
25     - vm-server1  
26   http:  
27     - match:  
28       - gateways:  
29         - mesh  
30         port: 8080  
31     route:  
32       - destination:  
33         host: vm-server1.vmns.svc.cluster.local  
34         port:  
35           number: 8080  
36         subset: v1  
37         weight: 50  
38       - destination:  
39         host: vm-server1.vmns.svc.cluster.local  
40         port:  
41           number: 8080  
42         subset: v2  
43         weight: 50  
44     fault:  
45       abort:  
46         httpStatus: 599  
47         percentage:  
48           value: 100
```

其中，599为自定义的HTTP状态码，100为故障百分比。

**步骤4** 单击“确定”完成修改。

----结束

## 验证故障注入

**步骤1** 在CCE集群中部署容器服务tomcat，具体操作请参见[部署容器服务](#)。

**步骤2** 在kubectl节点执行以下命令，获取tomcat工作负载的Pod名称。

```
kubectl get pod -n <vmns>|grep tomcat
```

回显示例：

```
tomcat-75ccb4b948-nzrfs    2/2    Running  0    19h
```

**步骤3** 执行以下命令，对虚拟机服务发起访问。

```
kubectl exec -it <tomcat-xxx> -n <vmns> -- curl -I -s -w "%{http_code}\n" -o /dev/null <vm-server1>.<vmns>.svc.cluster.local:8080
```

其中，<vm-server1>为虚拟机服务名称，<vmns>为**步骤3**中填写的命名空间。

若回显“599”，则表示配置的中断故障生效。

```
[root@cluster ~]# kubectl exec -it tomcat-75ccb4b948-nzrfs -n vmns -- curl -I -s -w "%{http_code}\n" -o /dev/null vm-server1.vmns.cluster.local:8080
Defaulting container name to container-1.
Use 'kubectl describe pod/tomcat-75ccb4b948-nzrfs -n vmns' to see all of the containers in this pod.
599
```

----结束

## 移除故障注入

- 步骤1 登录应用服务网格控制台，单击服务网格的名称，进入网格详情页面。
- 步骤2 在左侧导航栏选择“网格配置”，单击“istio资源管理”页签。
- 步骤3 选择“istio资源：virtualservices”，命名空间选择“vmns”，选择[创建VirtualService](#)中创建的VirtualService，单击操作列的“编辑”，删除[更新VirtualService](#)中添加的字段。
- 步骤4 单击“确定”完成修改。

----结束

## 4.6 卸载 ASM-PROXY

如果已完成虚拟机治理任务，可以将其中的ASM-PROXY卸载，释放资源。

### 删除已添加的 iptables 规则

- 步骤1 以root用户登录虚拟机。
- 步骤2 执行以下命令删除添加的istio iptables规则。

```
iptables -t nat -D PREROUTING -p tcp -j ISTIO_INBOUND
iptables -t nat -D OUTPUT -p tcp -j ISTIO_OUTPUT
iptables -t nat -F ISTIO_OUTPUT
iptables -t nat -X ISTIO_OUTPUT
iptables -t nat -F ISTIO_INBOUND
iptables -t nat -X ISTIO_INBOUND
iptables -t nat -F ISTIO_REDIRECT
iptables -t nat -X ISTIO_REDIRECT
iptables -t nat -F ISTIO_IN_REDIRECT
iptables -t nat -X ISTIO_IN_REDIRECT
iptables-save
```

- 步骤3 执行以下命令查看虚拟机当前的iptables规则。

```
iptables -t nat -L -v
```

回显如下，可以看到istio相关规则已经清除。

```
Chain PREROUTING (policy ACCEPT 2 packets, 104 bytes)
pkts bytes target  prot opt in     out     source      destination
```

```
47 2498 DOCKER all -- any any anywhere anywhere ADDRTYPE match dst-type
LOCAL

Chain INPUT (policy ACCEPT 2 packets, 104 bytes)
pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 5 packets, 350 bytes)
pkts bytes target prot opt in out source destination
0 0 DOCKER all -- any any anywhere !localhost/8 ADDRTYPE match dst-type
LOCAL

Chain POSTROUTING (policy ACCEPT 5 packets, 350 bytes)
pkts bytes target prot opt in out source destination
0 0 MASQUERADE all -- any !docker0 172.17.0.0/16 anywhere

Chain DOCKER (2 references)
pkts bytes target prot opt in out source destination
0 0 RETURN all -- docker0 any anywhere anywhere

----结束
```

## 卸载容器

**步骤1** 在虚拟机上执行以下命令，获取asm-proxy容器ID。

```
docker ps | grep asm-proxy
```

回显第一列为asm-proxy容器的ID。

```
46afed024f46 asm-proxy:1.8.4 "/usr/bin/bash /usr/..." 2 days ago Up 2 days asm-proxy
```

**步骤2** 执行以下命令，停止asm-proxy容器。

```
docker rm -f xxx
```

其中，*xxx*为**步骤1**获取的asm-proxy容器ID。

----结束

# 5 如何搭建 IPv4/IPv6 双栈网格

当前CCE支持创建IPv4/IPv6双栈集群，在双栈集群基础上支持开启IPv4/IPv6网格。启用双栈网格后，服务拥有 IPv4地址和IPv6地址，通过这两个地址都可以进行服务间的访问。本教程将指引您搭建一个IPv4/IPv6双栈的网格，使网格内的服务可以通过IPv6地址互访。添加双栈网关后，可以为IPv6终端的客户提供对外访问。

## 使用场景

- 如果您的服务需要使用IPv6地址进行服务间的互访和流量治理，您需要使用IPv6双栈。
- 如果您要为使用IPv6终端的客户提供对外访问，您需要使用IPv6双栈网格来创建网关。

## 约束与限制

- 支持启用双栈网格的约束。

网格类型	Istio版本	支持启用的集群类型	集群网络类型	其他说明
基础版	1.18及以上	CCE Turbo 集群	云原生网络2.0	集群需要已启用IPv6，请参考 <a href="#">通过CCE搭建IPv4/IPv6双栈集群</a>

- 支持创建IPv4/IPv6网关的约束。

网格类型	Istio版本	支持启用的ELB类型	ELB规格	其他说明
基础版	1.18及以上	独享型ELB	4层网络型	ELB需要配置IPv6地址

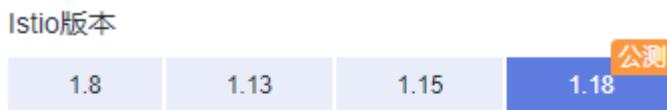
- 您的网格开启IPv4/IPv6双栈后不支持关闭，未开启双栈的网格也不支持创建完成之后开启双栈。

- 您低版本的网格升级到1.18及以上时，不支持升级后开启IPv4/IPv6双栈。

## 创建 IPv6 网格

步骤1 登录ASM控制台，购买一个ASM网格，网格参数填写如下：

- 网格类型：选择“基础版”。
- 网格名称：输入网格名称。
- Istio版本：选择"1.18"或以上版本。
- 启用IPv6：打开启用开关，开启后将过滤满足条件的CCE集群。



其他配置参数根据实际情况填写。

步骤2 创建完成后，单击网格名称进入，在“网格配置-基本信息”中可以看到“IPv6双栈已开启”字样。

网格状态	运行中
计费模式	免费
IPv6 双栈	已开启

----结束

## 添加 IPv4/IPv6 双栈网关

步骤1 登录ASM控制台，在网格列表页面单击已经开启IPv6双栈的网格名称，单击“网关管理-添加网关”。网关参数如下填写

- 访问方式：选择“DualStack”。
- 负载均衡：选择独享型，选中的独享型ELB需要有IPv6地址。

其他配置参数根据实际情况填写。

### 说明

当前IPv4/IPv6双栈网关添加IPv6外部访问地址时，只支持添加域名，通过域名访问，不支持添加IPv6地址访问。

----结束

## 验证方式

1. 客户端通过配置域名解析，解析域名到网关的IPv6地址，可以通过域名访问实现IPv6客户端访问。

```
[root@node1 ~]# curl -v http://test.com:8678
*   Trying 2407:c080:11f0:5fa:4ede:f73d:a9e9:c3c2:8678...
* Connected to test.com (2407:c080:11f0:5fa:4ede:f73d:a9e9:c3c2) port 8678 (#0)
```

2. 查看ingressgateway日志有对应IPv6请求日志信息。

```
{"user_agent":"curl/7.79.1","upstream_transport_failure_reason":null,"cluster_id":"072f35ec-8388-11ee-9bad-0255ac100b06","mesh_id":"59bac675-94d0-11ee-877b-0255ac10069a","downstream_local_address": "[2407:c080:11f0:601:7d2f:d21b:504d:a96c]:1027","upstream_local_address": "[2407:c080:11f0:601:7d2f:d21b:504d:a96c]:51094","requested_server_name": "test.com","upstream_service_ip": "10.12.10.9:6655","request_id": "10ff39ef-3864-4a0a-903b-0aef3864a0a0","path": "/","pod_name": "istio-system","response_flags": "0","duration": "1ms","forwarded_for": "10.12.10.9:6655;de01","authority": "Test.com:8678","bytes_received": 0,"route_name": "tomcat http v6.default.svc.cluster.local:8082","response_code": 200,"protocol": "HTTP/1.1","upstream_cluster": "outbound[8082]||tomcat http v6.default.svc.cluster.local","downstream_remote_address": "[2407:c080:11f0:5fa:4b12:f0c9:6655:de01]:53309","method": "GET","start_time": "2023-12-09T08:24:23.704Z","bytes_sent": 11230,"pod_name": "istio-ingressgateway-1-18-5-rl-5dbfd5d49b-sprl2"}
```