ALIBABA CLOUD

阿里云

服务网格

ASM 服务网格公共云合集

文档版本：20210301

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.产品公告

1.1. CVE-2020-11080漏洞公告

Nghttp2是一个用于实现HTTP/2的C库。Nghttp2 1.41.0之前版本中存在安全漏洞。攻击者可借助恶意的客户端构建14400字节长度的SETTINGS帧利用该漏洞造成拒绝服务。Istio中的Sidecar代理Envoy使用了该HTTP2库，并存在该漏洞。本文介绍CVE-2020-11080漏洞的影响范围和防范措施。

通过发送特制的数据包，攻击者可能导致CPU尖峰达到100%。该漏洞涉及到入口网关或sidecar代理。具体操作，请参见ISTIO 。

影响范围

以下版本在此漏洞影响范围内：

- Istio1.5.x版本：1.5.0、1.5.1、1.5.2、1.5.3以及1.5.4。
- Istio1.6.x版本：1.6.0、1.6.1。

防范措施

- 对于Istio1.5.4及之前的版本，需要升级到1.5.5版本。
- ASM支持1.6.x版本，您需要将ASM升级到1.6.2以上版本，从而规避该漏洞。

1.2. CVE-2020-10739漏洞公告

攻击者向启用telemetry v2的服务网格服务发送特制的数据包，触发空指针异常，从而造成拒绝服务。本文介绍CVE-2020-10739漏洞的影响范围及防范措施。

发送途径包括Ingress gateway或者Sidecar代理，详细描述请参见ISTIO。

影响范围

以下版本在此漏洞影响范围内：

- Istio1.4.x版本：1.4.0到1.4.8版本。
- Istio1.5.x版本：1.5.0到1.5.3版本。

防范措施

- 对于Istio1.5.4以及之前的版本，需要将版本升级到1.5.5。
- ASM支持1.6.x版本，您需要将ASM升级到1.6.2以上版本， 从而规避该漏洞。
- 创建服务网格时，通过控制台禁用**开启采集Prometheus 监控指标**。

1.3. CVE-2020-12603漏洞公告

通过向服务网格服务发送特制的数据包，攻击者可以触发代理HTTP 2请求和响应的Envoy，从而消耗大量的内存。本文介绍CVE-2020-12603漏洞的影响范围和防范措施。

更多信息，请参见Isito官网。

影响范围

以下版本在此漏洞影响范围内：

- Istio1.5.x版本：1.5.0到1.5.6。
- Istio1.6.x版本：1.6.0到1.6.3。

防范措施

- 对于Istio 1.5.x版本：升级到1.5.7。
- 对于Istio 1.6.x版本：升级到1.6.4。

1.4. CVE-2020-12605漏洞公告

通过向服务网格服务发送特制的数据包，攻击者可以触发代理HTTP 1.1请求和响应的Envoy，从而消耗大量的内存。本文介绍CVE-2020-12605漏洞的影响范围和防范措施。

更多信息，请参见Isito官网。

影响范围

以下版本在此漏洞影响范围内：

- Istio1.5.x版本：1.5.0到1.5.6。
- Istio1.6.x版本：1.6.0到1.6.3。

防范措施

- 对于Istio 1.5.x版本：升级到1.5.7。
- 对于Istio 1.6.x版本：升级到1.6.4。

1.5. CVE-2020-8663漏洞公告

攻击者发起DoS攻击，Envoy接收过多请求连接，导致系统文件描述符FD耗尽。本文介绍CVE-2020-8863漏洞的影响范围和防范措施。

详情请参见Isito官网。

影响范围

以下版本在此漏洞影响范围内：

- Istio1.5.x版本：1.5.0到1.5.6。
- Istio1.6.x版本：1.6.0到1.6.3。

防范措施

- 对于Istio 1.5.x版本：升级到1.5.7。
- 对于Istio 1.6.x版本：升级到1.6.4。

1.6. CVE-2020-12604漏洞公告

攻击者向服务网格发送特制的数据包，导致内存占用持续增加。本文介绍CVE-2020-12604漏洞的影响范围和防范措施。

详情请参见Isito官网。

影响范围

以下版本在此漏洞影响范围内：

- Istio1.5.x版本：1.5.0到1.5.6。
- Istio1.6.x版本：1.6.0到1.6.3。

防范措施

- 对于Istio 1.5.x版本：升级到1.5.7。
- 对于Istio 1.6.x版本：升级到1.6.4。

2.产品发布记录

2.1. 新功能发布记录

本文为您介绍了服务网格ASM相关内容的最新动态。

2021年01月

功能	功能描述	发布地域	相关文档
支持中国站成都地域和国际站弗吉尼亚地域。	ASM相继在中国站成都地域、国际站弗吉尼亚地域开服。	全部	无
支持一键开启访问日志、Prometheus监控以及Kiali的能力。	为提升可观测能力的用户体验，ASM在新版本中支持了一键开启访问日志、Prometheus监控以及Kiali的能力。	全部	• 使用日志服务采集数据平面入口网关日志 • 使用日志服务采集数据平面的AccessLog • 通过ASM控制台开启Kiali的可观测性
支持一键开启HTTP/1.0协议。	支持一键开启HTTP/1.0协议，解决Envoy默认需要上游服务使用HTTP/1.1或HTTP/2.0的问题，更好地兼容使用HTTP/1.0协议的遗留系统。	全部	无
完善入口网关定义， 优化更新与升级。	• 完善入口网关定义，支持NodeSelector能力， 规范化入口网关使用负载均衡SLB的Annotation注解定义。 • 对网格实例配置更新、新版本升级进行优化，减少等待时间，提升用户体验。 • 增强EnvoyFilter自定义资源的校验功能。	全部	无

2020年11月

功能	功能描述	发布地域	相关文档
支持Istio 1.7.5版本，并开放支持国际站。	ASM支持Istio 1.7.5版本，并开放支持国际站。	全部	无

功能	功能描述	发布地域	相关文档
支持Istio CNI插件。	ASM从1.7版本开始支持Istio CNI插件，该插件可以去掉Istio-init容器的特权诉求，从而符合更好的安全要求。	全部	启用CNI插件
支持Kiali for ASM。	Kiali for ASM提供了Web图形用户界面，用于查看服务网格的运行状况。	全部	通过应用目录开启Kiali的可观测性
发布数据平面热升级Beta版本。	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。	全部	ASM数据平面热升级Beta

2020年10月

功能	功能描述	发布地域	相关文档
支持多种方式灵活开启自动注入。	ASM提供了多种方式灵活开启自动注入，包含以下功能： 说明 ASM须升级到v1.6.8.19或者以上版本。 • 支持一键启用为所有命名空间开启自动注入功能。 • 支持通过Pod Annotations实现自动注入Sidecar的策略。 • 支持特定情况下的alwaysInjectSelector、neverInjectSelector的自动注入策略。	全部	多种方式灵活开启自动注入
支持数据面Kubernetes集群1.18版本。	ASM支持数据面Kubernetes集群1.18版本，覆盖当前ACK集群的所有支持版本。 说明 ASM须升级到v1.6.8.19或者以上版本。	全部	无

2020年09月

功能	功能描述	发布地域	相关文档
ASM Istio版本升级到1.6.8。	ASM现已支持Istio版本1.6.8，并增加了对ACK-on-ECI以及ASK集群的全面支持，至此ASM已完成全面统一支持多种类型的计算基础设施，包括ACK专有集群、ACK托管集群、ASK集群、ACK-on-ECI、ECI、ECS虚拟机、外部注册集群。	全部	无
增强Mixerless Telemetry V2能力。	ASM增强了Mixerless Telemetry V2能力，为网格内的服务通信提供了一种非侵入式的生成遥测数据的能力，并使用这些ASM指标实现工作负载的自动弹性伸缩。	全部	使用ASM指标实现工作负载的自动弹性伸缩
支持网格诊断功能。	ASM支持对网格实例进行网格诊断，诊断项包括数据平面版本检查、服务端口检查、服务关联检查、App及Version标签检查、目标地址检查和虚拟服务冲突检查，以帮助您更方便地使用和运维服务网格。	全部	使用ASM网格诊断
支持AHAS流量控制能力。	服务网格ASM配合阿里云应用高可用服务AHAS可以对部署在服务网格内的应用进行流量控制。	全部	通过AHAS对网格实例进行流量控制

2020年08月

功能	功能描述	发布地域	相关文档
支持集群本地域名设置。	提供网格实例创建时集群本地域名的定义，默认为cluster.local，例如接入的ACK集群使用了自定义域名，则在此需要设置与ACK集群相同的集群本地域名。	全部	无
支持非容器化的虚拟机应用。	ASM现已支持虚拟机接入到网格实例中，实现非容器化应用和容器应用的统一流量管理。	全部	通过ASM管理VM非容器应用Bookinfo

功能	功能描述	发布地域	相关文档
支持ASK集群接入。	服务网格已支持容器服务Serverless Kubernetes（ASK）集群接入，可对ECI容器实例上的工作负载做统一流量管理。	全部	无

2020年07月

功能	功能描述	发布地域	相关文档
服务网格产品商用化。	阿里云服务网格（ASM）提供一个全托管式的服务网格平台，兼容社区Istio开源服务网格，用于简化服务的治理，包括服务调用之间的流量路由与拆分管理、服务间通信的认证安全以及网格可观测性能力，从而极大地减轻开发与运维的工作负担。ASM定位于混合云、多云、多集群、非容器应用迁移等核心场景中，构建托管式统一的服务网格能力，能够为阿里云用户提供以下功能： • 一致的管理方式。 • 统一的流量管理。 • 控制平面核心组件托管化。 产品当前为免费，用户只需承担关联产品如ACK、SLB、SLS等费用即可。	开放北京、杭州、张家口、上海、深圳、雅加达、法兰克福区域，其他区域陆续开放中。	无
支持阿里云自建链路追踪。	ASM支持向阿里云链路追踪服务（xtrace）导出链路追踪数据，也同样支持向自建的兼容Zipkin协议的系统导出追踪数据。	全部	向自建系统导出ASM链路追踪数据
支持注册集群接入。	您可以通过容器服务控制台注册外部Kubernetes集群，并通过服务网格ASM进行应用管理。	全部	通过ASM管理外部注册Kubernetes集群应用

3.网格诊断

3.1. 使用ASM网格诊断

服务网格ASM支持对实例进行网格诊断，诊断项包括数据平面版本检查、服务端口检查、服务关联检查、App及Version标签检查、目标地址检查和虚拟服务冲突检查。本文介绍如何使用ASM进行网格诊断。

操作步骤

1. 登录ASM控制台。
2. 在左侧导航栏选择**服务网格 > 网格诊断**。
3. 在**网格诊断**页面左上角选择网格实例，单击**运行**。

   > **说明** 如果已诊断该网格实例，**网格诊断**页面将显示上一次的诊断结果。您可以单击**运行**，对该网格实例进行再次诊断。

   在**网格诊断**页面下方显示诊断项结果：

   - 诊断项**结果**列显示，表示该诊断项正常。
   - 诊断项**结果**列显示，表示该诊断项异常并显示异常原因。鼠标移至**详情**，您可以查看该诊断项异常的详细信息。

4.入口网关

4.1. 添加入口网关服务

如果部署的应用需要对公网提供访问，需要部署一个入口网关服务到集群中。本文介绍如何为ASM实例中的ACK集群添加入口网关服务。

前提条件

已创建至少一个ASM实例，并已添加至少一个ACK集群到该实例中。

背景信息

入口网关服务（Ingress Gateway）为Kubernetes集群提供了七层网关功能，对外提供一个统一的七层服务入口，根据HTTP请求的内容将来自同一个TCP端口的请求分发到不同的Kubernetes服务。

操作步骤

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**数据平面**区域，单击**入口网关服务**页签。
5. 在**入口网关服务**页签，单击**部署默认入口网关**。

   > **说明** 您也可以通过单击**部署自定义入口网关**来自定义入口网关服务，详情请参见自定义入口网关服务。

6. 在**部署入口网关**页面，为集群添加入口网关服务。
   i. 从**部署集群**列表中选择要部署入口网关服务的集群。
   ii. 指定负载均衡的类型，**公网访问**或**内网访问**。
   iii. 选择负载均衡。
   - 使用已有负载均衡：从已有负载均衡列表中选择。
   - 新建负载均衡：单击**新建负载均衡**，从下拉列表中选择所需的负载均衡规格。

   > **说明** 建议您为每个Kubernetes服务分配一个SLB。如果多个Kubernetes服务复用同一个SLB，存在以下风险和限制：
   > - 使用已有的SLB会强制覆盖已有监听，可能会导致您的应用不可访问。
   > - Kubernetes通过Service创建的SLB不能复用，只能复用您手动在控制台（或调用OpenAPI）创建的SLB。
   > - 复用同一个SLB的多个Service不能有相同的前端监听端口，否则会造成端口冲突。
   > - 复用SLB时，监听的名字以及虚拟服务器组的名字被Kubernetes作为唯一标识符。请勿修改监听和虚拟服务器组的名字。
   > - 不支持跨集群、跨地域复用SLB。

7. 配置端口映射。

i. 单击**添加端口**。

ii. 在新增端口行中，输入服务端口和容器端口。

> 说明
>
> - 建议容器端口与服务端口一致，并在Istio网关资源定义中启用了该端口。
> - 控制台默认提供了4个Istio常用的端口，但并不意味着必须从中选择，您可以根据需要自行添加或删除端口。

8. 单击**确定**。

执行结果

添加入口网关服务之后，可登录容器服务控制台查看详情。

- 查看新添加的入口网关服务的基本信息。
  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单击**服务**。
  v. 在**服务**页面，从**命名空间**下拉列表中选择istio-system。
  vi. 单击目标服务**操作**列的**详情**，查看入口网关服务的详细信息。
- 查看新添加入口网关服务的Pod信息。
  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**应用管理**。
  iv. 在工作负载页面单击**容器组**页签。
  v. 在**容器组**页面，从**命名空间**下拉列表中选择istio-system。
  vi. 单击目标Pod操作列的**详情**，查看入口网关服务的Pod详细信息。

4.2. 修改入口网关服务

服务网格ASM支持修改入口网关服务的配置，本文介绍如何在服务网格ASM修改入口网关服务。

前提条件

已添加入口网关，详情请参见添加入口网关服务。

操作步骤

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**数据平面**区域单击**入口网关服务**页签。
5. 在**入口网关服务**页签单击目标入口网关**操作**列的YAML。
6. 在**编辑**对话框修改参数，单击**确定**。

4.3. 删除入口网关服务

入口网关是服务网格的一种资源对象，用于管理服务网格的流量。您可以通过入口网关访问网格内的服务。本文介绍如何删除入口网关服务。

操作步骤

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**数据平面**区域单击**入口网关服务**页签。
5. 单击目标入口网关**操作**列的**删除**。
6. 在**确认**对话框中单击**确定**。

4.4. 自定义入口网关服务

阿里云服务网格ASM（Alibaba Cloud Service Mesh）除了可以通过控制台创建默认的入口网关服务之外，还支持通过CRD方式管理自定义网关。

前提条件

- 创建ASM实例，请参见创建ASM实例。
- 部署应用到ASM实例的集群中，请参见部署应用到ASM实例。
- 新增入口网关必须创建在命名空间istio-system中，以获取相关的配置信息。如果部署到其他命名空间，在Istio 1.6及以后的版本中，将因为不能获取相关配置而导致入口网关无法正常启动。

部署自定义网关

1. 创建并拷贝以下内容到*myexample-customingressgateway.yaml*文件中。

```
apiVersion: istio.alibabacloud.com/v1beta1
kind: IstioGateway
metadata:
 name: "myexample-customingressgateway"
 namespace: "istio-system"
spec:
 clusterIds:
  - "cluster1Id"
  - "cluster2Id"
 cpu:
  targetAverageUtilization: 80
 env:
```

```
  - name: "envname1"
    value: "envvalue1"
  externalTrafficPolicy: Local
  maxReplicas: 2
  minReplicas: 1
  ports:
  - name: status-port
    port: 15020
    targetPort: 15020
  - name: http2
    port: 80
    targetPort: 80
  - name: https
    port: 443
    targetPort: 0
  - name: tls
    port: 15443
    targetPort: 15443
  replicaCount: 1
  resources:
    limits:
      cpu: '2'
      memory: 2G
    requests:
      cpu: 200m
      memory: 256Mi
  sds:
    enabled: false
    resources:
      requests:
        cpu: 100m
        memory: 128Mi
      limits:
        cpu: 2000m
        memory: 1024Mi
# secretVolumes:
# - name: myexample-customingressgateway-certs
#   secretName: istio-myexample-customingressgateway-certs
#   mountPath: /etc/istio/myexample-customingressgateway-certs
  serviceType: LoadBalancer
  serviceAnnotations:
```

```
  service.beta.kubernetes.io/alicloud-loadbalancer-address-type: internet
serviceLabels:
 serviceLabelKey1: "serviceLabelValue1"
podAnnotations:
 podAnnotationsKey1: "podAnnotationsValue1"
rollingMaxSurge: "100%"
rollingMaxUnavailable: "25%"
overrides:
 cluster1Id:
  replicaCount: 1
  resources:
    limits:
     cpu: '2'
     memory: 2G
   requests:
    cpu: 200m
    memory: 256Mi
  serviceAnnotations:
   service.beta.kubernetes.io/alicloud-loadbalancer-address-type: internet
   service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: "slb.s1.small"
 cluster2Id:
  replicaCount: 2
  resources:
    limits:
     cpu: '4'
     memory: 4G
   requests:
    cpu: 400m
    memory: 512Mi
  serviceAnnotations:
   service.beta.kubernetes.io/alicloud-loadbalancer-address-type: internet
   service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: "slb.s2.small"
```

参数说明

字段	说明	默认值
metadata.name	名称，生成的Kubernetes Service和Deployment名称为istio-{该值}。	无

字段	说明	默认值
metadata.namespace	命名空间，生成的Kubernetes Service和Deployment所在的命名空间。 **注意** 为兼容Istio 1.6及以后的版本，该命名空间必须为istio-system。	istio-system
clusterIds	数组类型。将部署入口网关的集群Id，这些集群隶属于当前网格实例所管理。	无
cpu.targetAverageUtilization	HPA支持cpu的阈值。	80
env	数组类型。入口网关Pod的环境变量。	无
externalTrafficPolicy	表示此服务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和Local。	Local
maxReplicas	弹性伸缩的最大副本数。	5
minReplicas	弹性伸缩的最小副本数。	1
ports	数组类型。入口网关Pod定义的端口列表。例如： - name: status-port port: 15020 targetPort: 15020 - name: http2 port: 80 targetPort: 80 - name: https port: 443 targetPort: 0 - name: tls port: 15443 targetPort: 15443	无
replicaCount	副本数。	1

resources	入口网关Pod的资源配置。	- limits:
		  - cpu: '2'
		  - memory: 2G
		- requests:
		  - cpu: 200m
		  - memory: 256Mi

字段	说明	默认值
sds.enabled	是否启用SDS。	false
sds.resources	如果启用SDS，对应的Pod的资源配置。	◦ requests: ▪ cpu: 100m ▪ memory: 128Mi ◦ requests: ▪ cpu: 2000m ▪ memory: 1024Mi
secretVolumes	入口网关Pod所使用到的secret挂载卷，例如： ◦ name: myexample-customingressgateway-certs ◦ secretName: istio-myexample-customingressgateway-certs ◦ mountPath: /etc/istio/myexample-customingressgateway-certs	无
serviceType	入口网关的服务类型，可以是LoadBalancer、Nodeport或者ClusterIP。	LoadBalancer
serviceAnnotations	入口网关服务的Annotation定义。例如：service.beta.kubernetes.io/alicloud-loadbalancer-address-type: internet	无
serviceLabels	入口网关服务的Label定义。	无
podAnnotations	入口网关Pod的Annotation定义。	无
rollingMaxSurge	滚动更新过程中运行操作期望副本数的最大Pod数，可以为绝对数值，也可以为百分数。	"100%"
rollingMaxUnavailable	滚动更新过程中不可用的最大Pod数，可以为绝对数值，也可以为百分数。	"25%"

字段	说明	默认值
overrides	当clusterIds指定了2个或以上的集群时，可以针对特定的集群指定不同于上述参数定义的配置值，配置值为Map类型。 **说明** ○ key：本次定义的clusterIds中某一个集群Id。 ○ value：支持serviceAnnotations、resources、replicaCount参数的赋值。	无

2. 使用kubectl切换到服务网格实例对应的kubeconfig环境下，请参见通过kubectl连接ASM实例。
3. 创建命名空间myexample，请参见新建命名空间。
4. 在kubectl中执行 `kubectl apply -f myexample-customingressgateway.yaml` 命令，创建自定义入口网关。

执行结果

添加入口网关之后，可登录容器服务控制台查看详情。

查看新添加入口网关的服务信息。

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**服务与路由 > 服务**。
5. 在**服务**页面，从**命名空间**下拉列表中选择**myexample**。
6. 单击目标服务**操作**列的**详情**，查看新添加入口网关的服务信息。

查看新添加入口网关的Pod信息。

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**应用管理**。
4. 在工作负载页面单击**容器组**页签。
5. 在**容器组**页面，从**命名空间**下拉列表中选择**myexample**。
6. 单击目标Pod**操作**列的**详情**，查看新添加入口网关的Pod信息。

4.5. 通过服务网关启用HTTPS安全服务

本文介绍如何通过服务网关以文件挂载证书的方式启用HTTPS安全服务，即通过secret卷挂载的方式，将证书以文件方式挂载到Sidecar容器中。

前提条件

- 创建ACK集群，请参见快速创建Kubernetes托管版集群。
- 创建ASM实例，请参见创建ASM实例。
- 添加集群到ASM实例，请参见添加集群到ASM实例。
- 部署应用到ASM实例的集群中，请参见部署应用到ASM实例。
- 在加入到ASM实例的ACK集群中部署入口网关，请参见添加入口网关服务。
- 使用域名时需要备案才能正常访问，本示例中使用*aliyun.com*。

注意事项

建议使用SDS方式替代这种文件挂载，从而提升服务网关的安全性，详情请参见使用SDS为服务网关提升安全性。

步骤一：生成服务器证书和私钥

如果您已经拥有针对 `aliyun.com` 可用的证书和私钥，需要将密钥命名为 `aliyun.com.key` ，证书命名为 `aliyun.com.crt` 。如果没有，可以使用openssl通过如下操作来生成证书和密钥，操作步骤如下。

1. 执行以下命令，创建根证书和私钥。

```
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -subj '/O=myexample Inc./CN=aliyun.com' -keyout aliyun.root.key -out aliyun.root.crt
```

2. 执行以下命令，为 `aliyun.com` 服务器生成证书和私钥。

```
openssl req -out aliyun.com.csr -newkey rsa:2048 -nodes -keyout aliyun.com.key -subj "/CN=aliyun.com/O=myexample organization"
openssl x509 -req -days 365 -CA aliyun.root.crt -CAkey aliyun.root.key -set_serial 0 -in aliyun.com.csr -out aliyun.com.crt
```

步骤二：保存证书和私钥为Secret

通过以下命令在istio-system命名空间中创建包含证书和私钥的Secret：

```
kubectl create -n istio-system secret tls istio-ingressgateway-certs --key aliyun.com.key --cert aliyun.com.crt
```

> **说明** 注意是在入口网关Pod所在的集群对应的kubeconfig环境下。

步骤三：添加证书和私钥挂载卷到入口网关

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**数据平面**区域，单击**入口网关服务**页签。
5. 在对应的已部署的入口网关的操作列，单击**YAML**。
6. 在入口网关服务对应的YAML定义文件中，添加如下信息到spec：

```
secretVolumes:
 - mountPath: /etc/istio/ingressgateway-certs
   name: ingressgateway-certs
   secretName: istio-ingressgateway-certs
```

添加之后的YAML文件内容类似如下：

```
apiVersion: istio.alibabacloud.com/v1beta1
kind: IstioGateway
metadata:
 name: ingressgateway
 namespace: istio-system
spec:
 clusterIds:
  - c58638b491a5248669b59609e0a17****
 cpu: {}
 externalTrafficPolicy: Local
 maxReplicas: 1
 minReplicas: 1
 ports:
  - name: status-port
    port: 15020
    targetPort: 15020
  - name: http2
    nodePort: 31380
    port: 80
    targetPort: 80
  - name: https
    nodePort: 31390
    port: 443
    targetPort: 443
  - name: tls
    port: 15443
    targetPort: 15443
 replicaCount: 1
 secretVolumes:
  - mountPath: /etc/istio/ingressgateway-certs
    name: ingressgateway-certs
    secretName: istio-ingressgateway-certs
 serviceAnnotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: slb.s1.small
 serviceType: LoadBalancer
```

步骤四：检查入口网关配置

添加上述内容之后，入口网关Pod会加载证书与密钥。

通过以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），验证tls.crt与tls.key已经被挂载到入口网关Pod中。

```
kubectl exec -it -n istio-system $(kubectl -n istio-system get pods -l istio=ingressgateway -o jsonpath='{.items[0].metadata.name}') -- ls -al /etc/istio/ingressgateway-certs
```

运行上述命令后，可以得到类似以下结果。

```
lrwxrwxrwx 1 root root   14 May  9 01:14 tls.crt -> ..data/tls.crt
lrwxrwxrwx 1 root root   14 May  9 01:14 tls.key -> ..data/tls.key
```

执行以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），检查Ingress gateway证书中的Subject字段的正确性。

```
kubectl exec -it -n istio-system $(kubectl -n istio-system get pods -l istio=ingressgateway -o jsonpath='{.items[0].metadata.name}') -- cat /etc/istio/ingressgateway-certs/tls.crt | openssl x509 -text -noout | grep 'Subject:'
```

执行以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），检查Ingress gateway的代理能够正确访问证书。

```
kubectl exec -it -n istio-system $(kubectl -n istio-system get pods -l istio=ingressgateway -o jsonpath='{.items[0].metadata.name}') -- curl 127.0.0.1:15000/certs
```

步骤五：定义内部服务

示例中的内部服务是基于Nginx实现的，首先为Nginx服务器创建配置文件。以域名aliyun.com的内部服务为例，定义请求根路径直接返回字样Welcome to aliyun.com! 及状态码200。*myexample-nginx.conf*的具体内容如下。

```
events {
}
http {
 log_format main '$remote_addr - $remote_user [$time_local]  $status '
 '"$request" $body_bytes_sent "$http_referer" '
 '"$http_user_agent" "$http_x_forwarded_for"';
 access_log /var/log/nginx/access.log main;
 error_log  /var/log/nginx/error.log;
 server {
  listen 80;
  location / {
     return 200 'Welcome to aliyun.com!';
     add_header Content-Type text/plain;
  }
 }
}
```

1. 执行以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），创建Kubernetes ConfigMap存储Nginx服务器的配置。

   ```
   kubectl create configmap myexample-nginx-configmap --from-file=nginx.conf=./myexample-nginx.conf
   ```

2. 执行以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），设置命名空间default，启用sidecar自动注入。

   ```
   kubectl label namespace default istio-injection=enabled
   ```

3. 创建并拷贝以下内容到myexampleapp.yaml文件中，并执行 `kubectl apply -f myexampleapp.yaml` 命令，创建域名aliyun.com的内部服务。

```
apiVersion: v1
kind: Service
metadata:
 name: myexampleapp
 labels:
  app: myexampleapp
spec:
 ports:
 - port: 80
  protocol: TCP
 selector:
  app: myexampleapp
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: myexampleapp
spec:
 selector:
  matchLabels:
   app: myexampleapp
 replicas: 1
 template:
  metadata:
   labels:
    app: myexampleapp
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - name: nginx-config
      mountPath: /etc/nginx
      readOnly: true
   volumes:
   - name: nginx-config
    configMap:
     name: myexample-nginx-configmap
```

步骤六：创建自定义网关配置对象

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**控制平面**区域，单击**Gateway**页签，然后单击**新建**。
5. 在**新建**页面，按以下步骤定义服务网关，然后单击**确定**。
   i. 选择相应的命名空间。本文以选择**default**命名空间为例。
   ii. 在文本框中，定义服务网关。可参考以下YAML定义：

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
 name: istio-myexample-customingressgateway
spec:
 selector:
  istio: ingressgateway
 servers:
 - hosts:
  - '*.aliyun.com'
  port:
   name: http
   number: 80
   protocol: HTTP
  tls:
   httpsRedirect: true
 - hosts:
  - '*.aliyun.com'
  port:
   name: https
   number: 443
   protocol: HTTPS
  tls:
   mode: SIMPLE
   privateKey: /etc/istio/ingressgateway-certs/tls.key
   serverCertificate: /etc/istio/ingressgateway-certs/tls.crt
```

在**Gateway**页签可以看到新建的网关服务。

步骤七：创建虚拟服务

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**控制平面**区域，单击**VirtualService**页签，然后单击**新建**。
5. 在**新建**页面，按以下步骤定义虚拟服务，然后单击**确定**。
   i. 选择相应的命名空间。本文以选择**default**命名空间为例。
   ii. 在文本框中，定义Istio虚拟服务。可参考以下YAML定义。

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
 name: istio-myexample-customvirtualservice
spec:
 hosts:
 - "www.aliyun.com"
 gateways:
 - istio-myexample-customingressgateway
 http:
 - route:
  - destination:
     host: myexampleapp.default.svc.cluster.local
     port:
      number: 80
```

在**VirtualService**页签可以看到新建的虚拟服务。

执行结果

可以通过以下任意一种方法获取入口网关服务的地址：

- 通过控制台查看，请参见查看入口网关的服务信息。
- 在入口网关Pod所在的集群对应的kubeconfig环境下，执行以下命令，获取入口网关服务的地址。

```
kubectl get svc -n istio-system -l istio=ingressgateway
```

- 执行以下命令，通过HTTPS协议访问aliyun.com服务。

```
curl -k -H Host:www.aliyun.com --resolve www.aliyun.com:443:{替换成真实的入口网关IP地址} https://www.aliyun.com
```

```
Welcome to aliyun.com!
```

4.6. 通过服务网关启用TLS透传

本文介绍如何通过服务网关启用TLS透传，以实现对集群内HTTPS服务的安全入口访问。

前提条件

- 创建ACK集群，请参见快速创建Kubernetes托管版集群。
- 创建ASM实例，请参见创建ASM实例。
- 添加集群到ASM实例，请参见添加集群到ASM实例。
- 部署应用到ASM实例的集群中，请参见部署应用到ASM实例。
- 在加入到ASM实例的ACK集群中部署入口网关，请参见添加入口网关服务。
- 使用域名需要备案才能正常访问，本示例中使用aliyun.com。

步骤一：生成服务器证书和私钥

如果您已经拥有针对 `sample.aliyun.com` 可用的证书和私钥，需要将密钥命名为 `sample.aliyun.com.key` ，证书命名为 `sample.aliyun.com.crt` 。如果没有，可以通过openssl执行以下操作来生成证书和密钥。

1. 执行以下命令，创建根证书和私钥。

```
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -subj '/O=mynginx Inc./CN=aliyun.com' -keyout aliyun.root.key -out aliyun.root.crt
```

2. 执行以下命令，为 `sample.aliyun.com` 服务器生成证书和私钥。

```
openssl req -out sample.aliyun.com.csr -newkey rsa:2048 -nodes -keyout sample.aliyun.com.key -subj "/CN=sample.aliyun.com/O=mynginx sample organization"
openssl x509 -req -days 365 -CA aliyun.root.crt -CAkey aliyun.root.key -set_serial 0 -in sample.aliyun.com.csr -out sample.aliyun.com.crt
```

步骤二：定义内部服务

示例中的内部服务是基于Nginx实现的，首先为Nginx服务器创建配置文件。以域名**aliyun.com**的内部服务为例，定义请求根路径直接返回字样Welcome to aliyun.com without TLS Termination! 及状态码200。*mynginx.conf*的具体内容如下。

```
events {
}
http {
 log_format main '$remote_addr - $remote_user [$time_local]  $status '
 '"$request" $body_bytes_sent "$http_referer" '
 '"$http_user_agent" "$http_x_forwarded_for"';
 access_log /var/log/nginx/access.log main;
 error_log  /var/log/nginx/error.log;
 server {
  listen 443 ssl;
  location / {
    return 200 'Welcome to aliyun.com without TLS Termination!';
    add_header Content-Type text/plain;
  }
  server_name www.aliyun.com;
  ssl_certificate /etc/nginx-server-certs/tls.crt;
  ssl_certificate_key /etc/nginx-server-certs/tls.key;
 }
}
```

1. 在入口网关Pod所在的集群对应的kubeconfig环境下，执行以下命令，创建Kubernetes ConfigMap存储Nginx服务器的配置。

   ```
   kubectl create configmap mynginx-configmap --from-file=nginx.conf=./mynginx.conf
   ```

2. 在入口网关Pod所在的集群对应的kubeconfig环境下，执行以下命令，将在default命名空间中创建包含证书和私钥的Secret。

   ```
   kubectl create secret tls nginx-server-certs --key sample.aliyun.com.key --cert sample.aliyun.com.crt
   ```

3. 在入口网关Pod所在的集群对应的kubeconfig环境下，执行以下命令，设置命名空间default，启用sidecar自动注入。

   ```
   kubectl label namespace default istio-injection=enabled
   ```

4. 创建并拷贝以下内容到mynginxapp.yaml文件中，并执行 `kubectl apply -f mynginxapp.yaml` 命令，创建域名aliyun.com的内部服务。

   ```
   apiVersion: v1
   kind: Service
   metadata:
    name: mynginxapp
    labels:
     app: mynginxapp
   spec:
   ```

```
  ports:
  - port: 443
    protocol: TCP
  selector:
    app: mynginxapp
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mynginxapp
spec:
  selector:
    matchLabels:
      app: mynginxapp
  replicas: 1
  template:
    metadata:
      labels:
        app: mynginxapp
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 443
        volumeMounts:
        - name: nginx-config
          mountPath: /etc/nginx
          readOnly: true
        - name: nginx-server-certs
          mountPath: /etc/nginx-server-certs
          readOnly: true
      volumes:
      - name: nginx-config
        configMap:
          name: mynginx-configmap
      - name: nginx-server-certs
        secret:
          secretName: nginx-server-certs
```

5. 确认Nginx服务器已成功部署，执行以下命令从其sidecar代理向服务器发送请求。

```
kubectl exec -it $(kubectl get pod  -l app=mynginxapp -o jsonpath={.items..metadata.name}) -c istio-proxy -- curl -v -k --resolve sample.aliyun.com:443:127.0.0.1 https://sample.aliyun.com
```

步骤三：创建自定义网关配置对象

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**控制平面**区域，单击**Gateway**页签，然后单击**新建**。
5. 在**新建**页面，按以下步骤定义服务网关，然后单击**确定**。
   i. 选择相应的命名空间。本文以选择**default**命名空间为例。
   ii. 在弹出窗口的文本框中，定义服务网关。可参考以下YAML定义。

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
 name: istio-mynginx-customingressgateway
spec:
 selector:
  istio: ingressgateway
 servers:
 - hosts:
  - 'sample.aliyun.com'
  port:
   name: https
   number: 443
   protocol: HTTPS
  tls:
   mode: PASSTHROUGH
```

在**Gateway**页签可以看到新建的网关。

步骤四：创建虚拟服务

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**控制平面**区域，单击**VirtualService**页签，然后单击**新建**。
5. 在**新建**页面，按以下步骤定义虚拟服务，然后单击**确定**。
   i. 选择相应的命名空间。本文以选择**default**命名空间为例。

ii. 在文本框中，定义Istio虚拟服务。可参考以下YAML定义。

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
 name: istio-mynginx-customvirtualservice
spec:
 hosts:
 - "sample.aliyun.com"
 gateways:
 - istio-mynginx-customingressgateway
 tls:
 - match:
  - port: 443
   sniHosts:
   - sample.aliyun.com
  route:
  - destination:
     host: mynginxapp.default.svc.cluster.local
     port:
      number: 443
```

在VirtualService页签可以看到新建的虚拟服务。

执行结果

可以通过以下任意一种方法获取入口网关服务的地址：

- 通过控制台查看，请参见查看入口网关的服务信息。
- 执行以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），获取入口网关服务的地址。

```
kubectl get svc -n istio-system -l istio=ingressgateway
```

- 执行以下命令，通过HTTPS协议访问aliyun.com服务。

```
curl -v --cacert aliyun.root.crt --resolve sample.aliyun.com:443:xx.xx.xx.xx  https://sample.aliyun.com
```

```
Welcome to aliyun.com without TLS Termination!%
```

4.7. 使用SDS为服务网关提升安全性

通过使用SDS（Secret Discovery Service）为服务网关提供HTTPS安全支持、证书动态加载，从而提升服务网关安全性。本文介绍如何通过SDS配置TLS入口网关。

前提条件

- 创建ACK集群，请参见快速创建Kubernetes托管版集群。
- 创建ASM实例，请参见创建ASM实例。
- 添加集群到ASM实例，请参见添加集群到ASM实例。
- 部署应用到ASM实例的集群中，请参见部署应用到ASM实例。
- 在加入到ASM实例的ACK集群中部署入口网关，请参见添加入口网关服务。
- 使用域名需要备案才能正常访问，本示例中使用aliyun.com。

背景信息

在文档通过服务网关启用HTTPS安全服务中介绍了如何通过secret卷挂载的方式，将证书以文件方式挂载到Sidecar容器中。这种情况下，当证书发生轮转时需要重启服务让Sidecar代理重新加载证书。SDS（Secrete Discovery Service）是Istio提供的另外一种动态加载证书的方式，TLS （Transport Layer Security）所需的私钥、服务器证书以及根证书都可以由SDS完成配置。

入口网关代理与入口网关在同一Pod中运行，并监视与入口网关相同的命名空间中创建的Secret。在入口网关上启用SDS具有以下好处：

- 入口网关可以在不需要重启的情况下，动态添加、删除或更新所需要的证书、私钥或者对应的根证书。
- 不需要secret卷挂载。创建Kubernetes secret后，网关代理会捕获该secret，并将其包含的证书、私钥或根证书发送到入口网关。
- 网关代理可以监视多个证书、私钥对，只需要为多个主机创建secret并更新网关定义。

步骤一：在入口网关中启用SDS

> **说明** 在Istio1.6版本中，默认启用SDS，无需执行以下步骤。

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**数据平面**区域，单击**入口网关服务**页签。
5. 在对应的已部署的入口网关的**操作**列，单击**YAML**。
6. 在入口网关服务对应的YAML定义文件中，在spec下添加以下信息。

```
sds:
 enabled: true
 resources:
  requests:
   cpu: 100m
   memory: 128Mi
  limits:
   cpu: 2000m
   memory: 1024Mi
```

> **说明** 其中enabled必须设置为true, resources部分可以根据实际情况设置不同于默认值的资源配置。

添加之后的YAML文件内容类似如下。

```
apiVersion: istio.alibabacloud.com/v1beta1
kind: IstioGateway
metadata:
 name: ingressgateway
 namespace: istio-system
spec:
 clusterIds:
  - c58638b491a5248669b59609e0a17****
 cpu: {}
 externalTrafficPolicy: Local
 maxReplicas: 1
 minReplicas: 1
 ports:
  - name: status-port
    port: 15020
    targetPort: 15020
  - name: http2
    nodePort: 31380
    port: 80
    targetPort: 80
  - name: https
    nodePort: 31390
    port: 443
    targetPort: 443
  - name: tls
    port: 15443
    targetPort: 15443
 replicaCount: 1
 sds:
  enabled: true
  resources:
   requests:
    cpu: 100m
    memory: 128Mi
   limits:
    cpu: 2000m
    memory: 1024Mi
 serviceAnnotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: slb.s1.small
 serviceType: LoadBalancer
```

步骤二：为多个主机准备服务器证书和私钥

为 `aliyun.com` 生成证书和私钥，并保存为Secret。

如果您已经拥有针对 `aliyun.com` 可用的证书和私钥，需要将密钥命名为 `aliyun.com.key` ，证书命名为 `aliyun.com.crt` 。如果没有，可以通过openssl执行以下步骤来生成证书和密钥。

1. 执行以下命令，创建根证书和私钥。

```
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -subj '/O=myexample Inc./CN=aliyun.com' -keyout aliyun.root.key -out aliyun.root.crt
```

2. 执行以下命令，为 `aliyun.com` 服务器生成证书和私钥。

```
openssl req -out aliyun.com.csr -newkey rsa:2048 -nodes -keyout aliyun.com.key -subj "/CN=aliyun.com/O=myexample organization"
openssl x509 -req -days 365 -CA aliyun.root.crt -CAkey aliyun.root.key -set_serial 0 -in aliyun.com.csr -out aliyun.com.crt
```

3. 通过以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下）将在istio-system命名空间中创建包含证书和私钥的Secret。

```
kubectl create -n istio-system secret generic myexample-credential --from-file=key=aliyun.com.key --from-file=cert=aliyun.com.crt
```

> 注意 secret名称不应以istio或prometheus开头，且不应包含token字段。

步骤三：为a.aliyun.com定义内部服务

示例中的内部服务是基于Nginx实现的，首先为Nginx服务器创建配置文件。以域名 `a.aliyun.com` 的内部服务为例，定义请求根路径直接返回字样Welcome to a.aliyun.com!及状态码200。*myexample-nginx.conf*的具体内容如下。

```
events {
}
http {
 log_format main '$remote_addr - $remote_user [$time_local]  $status '
 '"$request" $body_bytes_sent "$http_referer" '
 '"$http_user_agent" "$http_x_forwarded_for"';
 access_log /var/log/nginx/access.log main;
 error_log  /var/log/nginx/error.log;
 server {
  listen 80;
  location /hello {
     return 200 'Welcome to a.aliyun.com!';
     add_header Content-Type text/plain;
  }
 }
}
```

1. 在入口网关Pod所在的集群对应的kubeconfig环境下，执行以下命令，创建Kubernetes ConfigMap，即存储Nginx服务器的配置。

   ```
   kubectl create configmap myexample-nginx-configmap --from-file=nginx.conf=./myexample-nginx.conf
   ```

2. 在入口网关Pod所在的集群对应的kubeconfig环境下，执行以下命令，设置命名空间default，启用sidecar自动注入。

   ```
   kubectl label namespace default istio-injection=enabled
   ```

3. 创建并拷贝以下内容到myexampleapp.yaml文件中，并执行 `kubectl apply -f myexampleapp.yaml` 命令，创建域名a.aliyun.com的内部服务。

```
apiVersion: v1
kind: Service
metadata:
 name: myexampleapp
 labels:
  app: myexampleapp
spec:
 ports:
 - port: 80
  protocol: TCP
 selector:
  app: myexampleapp
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: myexampleapp
spec:
 selector:
  matchLabels:
   app: myexampleapp
 replicas: 1
 template:
  metadata:
   labels:
    app: myexampleapp
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - name: nginx-config
      mountPath: /etc/nginx
      readOnly: true
   volumes:
   - name: nginx-config
    configMap:
      name: myexample-nginx-configmap
```

步骤四：为b.aliyun.com定义内部服务

本示例中的内部服务是基于httpbin实现的，创建并拷贝如下内容到httpbin.example.yaml文件中，并执行 `kubectl apply -f httpbin.example.yaml` 命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），创建域名 `b.aliyun.com` 的内部服务。

```
apiVersion: v1
kind: Service
metadata:
 name: httpbin
 labels:
  app: httpbin
spec:
 ports:
 - name: http
  port: 8000
 selector:
  app: httpbin
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: httpbin
spec:
 replicas: 1
 selector:
  matchLabels:
   app: httpbin
   version: v1
 template:
  metadata:
   labels:
    app: httpbin
    version: v1
  spec:
   containers:
   - image: docker.io/citizenstig/httpbin
    imagePullPolicy: IfNotPresent
    name: httpbin
    ports:
    - containerPort: 8000
```

步骤五：创建自定义网关配置对象

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**控制平面**区域，单击**服务网关**页签，然后单击**新建**。
5. 在**新建**页面，按以下步骤定义服务网关，然后单击**确定**。
   i. 选择相应的命名空间。本文以选择**default**命名空间为例。
   ii. 在文本框中，定义服务网关。可参考以下YAML定义：

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
 name: mysdsgateway
spec:
 selector:
  istio: ingressgateway # use istio default ingress gateway
 servers:
 - port:
    number: 443
    name: https
    protocol: HTTPS
  tls:
    mode: SIMPLE
    credentialName: myexample-credential # must be the same as secret
  hosts:
  - '*.aliyun.com'
```

在**服务网关**页签可以看到新建的虚拟服务。

步骤六：创建虚拟服务

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**控制平面**区域，单击**虚拟服务**页签，然后单击**新建**。
5. 在**新建**页面，按以下步骤定义虚拟服务，然后单击**确定**。
   i. 选择相应的命名空间。本文以选择**default**命名空间为例。
   ii. 在文本框中，定义Istio虚拟服务。可参考以下YAML定义。

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
 name: mysdsgateway-myexamplevs
spec:
 hosts:
 - "a.aliyun.com"
 gateways:
 - mysdsgateway
 http:
 - match:
  - uri:
     exact: /hello
  route:
  - destination:
     host: myexampleapp.default.svc.cluster.local
     port:
      number: 80
```

同样地，为httpbin.example.com提供定义相应的虚拟服务。

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
 name: mysdsgateway-httpbinvs
spec:
 hosts:
 - "b.aliyun.com"
 gateways:
 - mysdsgateway
 http:
 - match:
  - uri:
     prefix: /status
  - uri:
     prefix: /delay
  route:
  - destination:
     port:
      number: 8000
     host: httpbin.default.svc.cluster.local
```

在**虚拟服务**页签可以看到新建的虚拟服务。

执行结果

可以通过以下任意一种方法获取入口网关服务的地址：

- 通过控制台查看，请参见查看入口网关的服务信息。
- 执行以下命令（注意是在入口网关Pod所在的集群对应的kubeconfig环境下），获取入口网关服务的地址。

```
kubectl get svc -n istio-system -l istio=ingressgateway
```

- 执行以下命令，通过HTTPS协议访问aliyun.com服务。

```
curl -k -H Host:a.aliyun.com --resolve a.aliyun.com:443:{替换成真实的入口网关IP地址} https://a.aliyun.com/hello
```

```
Welcome to aliyun.com!
```

- 执行以下命令，通过HTTPS协议访问httpbin.example.com服务。

```
curl -k -H Host:b.aliyun.com --resolve b.aliyun.com:443:{替换成真实的入口网关IP地址} https://b.aliyun.com/status/418
```

```
  -=[ teapot ]=-

     _...._
   .' _  _ `.
  | ."` ^ `". _,
  \_;`"---"`|//
    |       ;/
    \_     _/
      `"""`
```

5.对接服务注册中心

5.1. 对接Consul注册中心

服务网格ASM提供了对接Consul注册中心功能，便于您将微服务迁移至服务网格ASM。本文介绍如何在服务网格ASM中对接Consul注册中心。

前提条件

- 服务网格ASM版本需要升级到v1.7.5.31-g28ec7490-aliyun或者以上版本。
- 已部署Consul作为服务注册中心，具体操作，请参见Installing Consul on Kubernetes。
- 请确保加入服务网格中的Kubernetes集群的Pod可以访问Consul Server的访问地址。例如Consul Server是安装在相同的Kubernetes集群，Consul Server暴露了公网或者Consul Server提供了可访问的内网地址。
- Consul中已注册示例服务，分别为Web，Web2以及内置的Consul。具体操作，请参见Services。

通过Aliyun CLI对接Consul Server

1. 在浏览器中输入Cloud Shell，或者在OpenAPI Explorer中打开命令行操作界面。关于OpenAPIExplorer，请参见OpenAPI。

   > **说明** 您可以根据实际需要打开多个命令行窗口，但最多可同时打开5个云命令行窗口。

2. 使用以下内容，创建*config.json*。

```
[
 {
  "name": "consul-test",
  "prefix": "consul-",
  "type": "consul",
  "endpoint": "http://consul-server.consul:8500",
  "toNamespace": "default"
 }
]
```

参数	说明
name	注册中心的名称，保证唯一。
prefix	生成的ServiceEntry的名称前缀。
type	注册中心的类型，当前支持值为consul。
endpoint	注册中心的访问端点地址。
toNamespace	生成的ServiceEntry所在的命名空间，如果该命名空间不存在，将会自动创建。

3. 执行以下命令，对接Consull注册中心。

```
aliyun servicemesh SetServiceRegistrySource --ServiceMeshId cf9e58cf8743748b3bd13867d6d87e30b --Config "$(cat config.json)"
```

ASM对接Consull注册中心后，会自动在ACK集群安装asm-serviceregistry组件，并将在Consul中的服务同步到服务网格中。

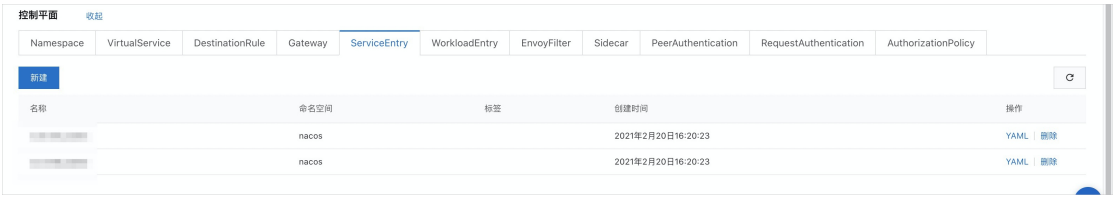
查看Consull注册中心对接结果

1. 查看ASM组件安装情况。
   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**页面查看到名为asm-serviceregistry的组件。

2. 查看Service Entry同步情况。
   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**控制平面**区域单击**ServiceEntry**页签。

v. 在**ServiceEntry**页签可以看到3个注册在Consul中的服务已经同步到服务网格中。

> 说明
>
> - 在**ServiceEntry**页签下的Consul服务的名称命名规则为：[步骤定义的prefix值]-[在Consul中注册的服务名]。
> - 在**ServiceEntry**页签下的Consul服务的命名空间为步骤定义的 `toNamespace` 值。



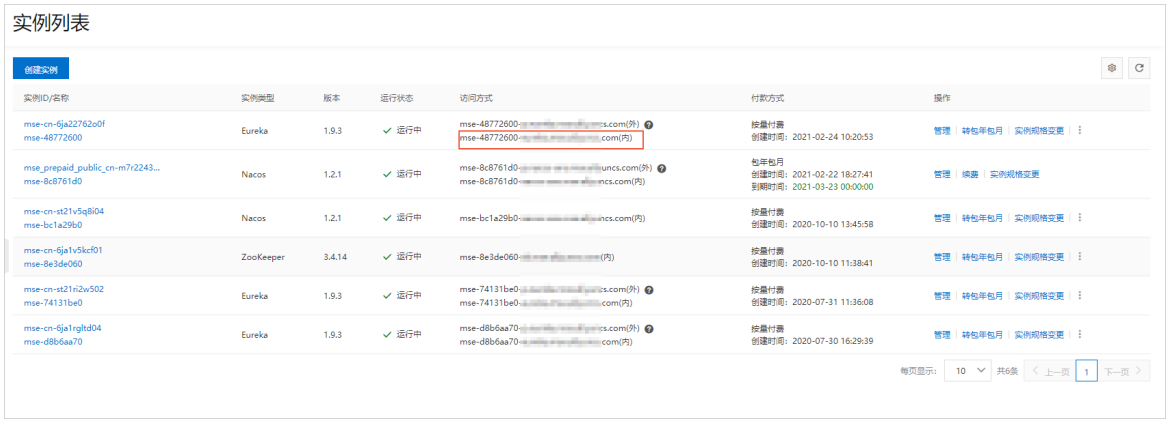
5.2. 对接Nacos注册中心

服务网格ASM提供了对接Nacos注册中心功能，便于将Nacos上的微服务与服务网格ASM进行互通。本文介绍如何对接Nacos注册中心。

操作步骤

1. 开启MCP功能。
   - 如果您已有1.2.1版本的Nacos实例，您需要开启MCPEnabled。
     a. 登录MSE管理控制台。
     b. 在控制台左侧导航栏选择**注册配置中心 > 实例列表**。
     c. 在**实例列表**页面单击目标实例右侧**操作**列的**管理**。
     d. 在实例详情页面左侧导航栏单击**参数设置**。
     e. 在**参数设置**页面单击**编辑**，在**MCPEnabled**参数值下方选择**是**，然后单击**保存并重启实例**。
   - 如果您没有1.2.1版本的Nacos实例，您需要创建1.2.1版本的Nacos实例并开启MCPEnabled。
     a. 创建Nacos实例。
        a. 登录MSE管理控制台。
        b. 在控制台左侧导航栏选择**注册配置中心 > 实例列表**。
        c. 在**实例列表**页面单击**创建实例**。
        d. 设置Nacos实例参数，然后单击**立即购买**。

           设置引擎类型为Nacos，引擎版本为1.2.1。其他参数设置请参见购买并构建Nacos引擎。
        e. 在确认订单页面阅读并选中MSE服务协议，并依据提示支付。
        f. 支付完成后，单击**管理控制台**，返回**实例列表**页面。
     b. 开启MCPEnabled。
        a. 在**实例列表**页面单击**操作**列的**管理**。
        b. 在实例详情页面左侧导航栏单击**参数设置**。
        c. 在**参数设置**页面单击**编辑**，在**MCPEnabled**参数值下方选择**是**，然后单击**保存并重启实例**。
2. 在**实例列表**页面查看目标实例**访问方式**列下内部访问地址。



3. 在浏览器中输入Cloud Shell，或者在OpenAPI Explorer中打开命令行操作界面。关于OpenAPIExplorer，请参见OpenAPI。

> **说明** 您可以根据实际需要打开多个命令行窗口，但最多可同时打开5个云命令行窗口。

4. 使用以下内容，创建名为*config.json*的文件。

```
[
 {
  "name": "nacos-test",
  "type": "nacos",
  "endpoint": "nacos-server.nacos:18848",
  "toNamespace": "nacos"
 }
]
```

参数	说明
name	注册中心的名称，保证唯一。
type	注册中心的类型，本例为 `nacos` 。
endpoint	注册中心接入点。格式为步骤获取的访问端点地址:18848。
toNamespace	生成的ServiceEntry所在的命名空间，本例为 `nacos` 。

5. 执行以下命令，对接Nacos注册中心。

```
aliyun servicemesh SetServiceRegistrySource --ServiceMeshId cf9e58cf8743748b3bd13867d6d87**** --Config "$(cat config.json)"
```

ASM对接Nacos注册中心后，会自动在ACK集群安装asm-serviceregistry组件，并将在Nacos中的服务同步到服务网格中。

查看Nacos注册中心对接结果

1. 查看ASM组件安装情况。
   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**页面查看到名为asm-serviceregistry的组件。

2. 查看Service Entry同步情况。
   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**控制平面**区域单击**ServiceEntry**页签。
   v. 在**ServiceEntry**页签可以看到注册在Nacos中的服务已经同步到服务网格中。