DaoCloud Enterprise

DaoCloud Enterprise Product Documentation

DaoCloud

Copyright © 2016 - 2023 DaoCloud

Table of contents

1. 常见问题 3
2. 安装服务网格 5
3. Intro 8
   3.1 系统架构 8
   3.2 产品优势 10
   3.3 功能总览 11
   3.4 UI 界面支持的功能 13
   3.5 服务网格 Release Notes 15
   3.6 适用场景 17
   3.7 什么是服务网格 19
4. Test 20
   4.1 OCP 环境接入服务网格配置方法 20
   4.2 兼容性测试报告 - OpenShift 22
5. User guide 30
   5.1 Cluster management 30
   5.2 Gateway instance 35
   5.3 Mesh config 41
   5.4 Multicluster 47
   5.5 Security 107
   5.6 Service list 129
   5.7 Service mesh 143
   5.8 Sidecar management 158
   5.9 Traffic governance 176
   5.10 Traffic monitor 199
   5.11 Upgrade 202

Copyright © 2016 - 2023 DaoCloud

1. 常见问题

1. 服务网格中的 Istio 是否可以单独安装？

   可以，但不建议。用户单独安装的 Istio 控制面组件可通过外接网格的方式接入服务网格，但相比于其他模式，只提供基础的 Istio 具备的功能。

2. 服务网格中的 Istio 组件版本？

   有多个版本，用户可以在创建网格时选择。

3. 服务网格的 Istio 是否可以升级？

   可以，支持用户自主升级。

4. 网格中注入的边车是否可以升级？

   可以，同时支持边车热升级。

5. 第 4 代服务网格（DSM）是否可升级到第 5 代服务网格？

   不可以升级，但可以以外接网格的形式接入服务网格。建议采用重新安装的方式，获得最好的扩展性和后期维护。

6. 如果设置了不同的命名空间边车注入策略和工作负载边车注入策略，实际执行效果是什么样？

   将按以下规则顺序匹配，匹配到有效规则将不再执行之后的规则：

   - 如果二者有一个设置为禁用，则禁用边车注入
   - 如果二者有一个设置为启用，则启用边车注入
   - 如果二者都没有设置，将执行网格全局边车注入策略（values.sidecarInjectorWebhook.enableNamespacesByDefault）

   详细情况可参考 安装 Sidecar

7. 传统微服务加入服务网格有什么不同？

   传统微服务加入服务网格只需要将其对应的工作负载注入边车，以便网格能够对其进行管理。 基于不同的微服务框架以及网格的无侵入的特性，原先框架的治理策略等可能会有所改变，但无需用户自行实现。 当接入网格后，传统微服务即可获得网格的全部能力，包括流量治理、安全、可观测性等等。

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

2. 安装服务网格

请确认您的集群已成功接入 容器管理 平台，然后执行以下步骤安装服务网格。

Copyright © 2016 - 2023 DaoCloud

1. 从左侧导航栏点击 容器管理 ，进入 集群列表 ，点击准备安装服务网格的集群名称。

2. 在 集群概览 页面中点击 控制台 。

3. 自控制台中逐行输入如下命令：

```
sh
export VERSION=0.0.0-xxxx
helm repo add mspider https://release.daocloud.io/chartrepo/mspider
```


Copyright © 2016 - 2023 DaoCloud

```
helm repo update
helm upgrade --install --create-namespace -n mspider-system mspider mspider/mspider --version=${VERSION} --set global.imageRegistry=release.daocloud.io/mspider
```

!!! note

> 请将 `0.0.0-xxx` 替换为计划安装的服务网格版本号。

4. 查看命名空间 `mspider-system` 下 Pod 信息，看到相关 Pod 已创建并运行，表示服务网格安装成功。

Copyright © 2016 - 2023 DaoCloud

3. Intro

3.1 系统架构

服务网格产品具有多网格管理以及多集群服务聚合治理的能力；用户可以在多云的环境中将不同来源的集群接入网格中，进行统一的流量管理与安全治理。

从整体架构上讲，服务网格产品可分为三个层面：全局管理模块、托管网格模块、被托管集群网格

- 全局管理模块

  运行于独立的控制面集群 Global Service Cluster（以下简称 GSC），用于统一对多个服务网格进行管理，处理与终端用户的交互。 整个网格管理平台的资源配置、安全治理等主要业务逻辑都将在这一层进行构建。同时，也是和其它系统对接的主要承担模块，这样，将减少其它模块的适配成本。 因其不参与具体的服务治理事务，因此并不包含 Istio 相关组件。

- 托管网格模块

  运行于 Mesh Control Plane Cluster（以下简称 MCPC），本质上也是用户工作集群，同时具备网格控制面的角色，一个 GSC 可以管理多个 MCPC。 Istio 等核心控制组件会安装在该集群，作为网格的核心控制平面，直接对本网格内的多个集群进行管理，进行统一的策略管控和服务发现，实际执行、下发各项流量策略和安全策略，并确保仅有写入 MCPC 的策略才会对整个网格生效。

- 被托管集群网格

  处于一个网格中的实际工作集群中，包含了 Istio 的基本组件，但不会作为控制面使用，仅是提供一个类似管理代理的模式，主要负责边车注入、证书转发、xDS 转发等业务，同步来自 MCPC 的策略、服务注册信息下发至集群的所在业务 Sidecar。

贯穿整体架构的拓展模块主要有：观测模块、微服务平台/应用工作台

- 观测模块

  可观测性完全由 Insight 处理，服务网格通过接口调用的方式获取流量指标信息绘制拓扑图，并直接调用 Istio 原生 grafana 为用户提供各项指标图表。

- 微服务平台/应用工作台

  微服务平台可以通过网格对其微服务进行网格能力的赋能，方便用户通过单一平台对各类微服务系统做统一管理。

Copyright © 2016 - 2023 DaoCloud

功能架构

如上图所示，服务网格提供了 9 个模块 12 个子模块功能，实现了多样化集群接入、多种模式的网格管理的能力。

Copyright © 2016 - 2023 DaoCloud

3.2 产品优势

此处介绍 DCE 5.0 服务网格相比其他产品所具备的优势。

- 简单易用

  无需修改任何业务代码，也无需手动安装代理，只需开启服务网格功能，即可体验丰富的无侵入服务治理能力。

- 策略化的智能路由与弹性流量管理

  支持为服务配置负载均衡、服务路由、故障注入、熔断等治理规则。结合一站式治理系统；提供实时的、可视化的微服务流量管理；支持无侵入智能流量治理，应用无需任何改造，即可进行动态的智能路由和弹性流量管理。

- 权重、内容、TCP/IP 等路由规则。
- HTTP 会话保持，满足业务处理持续性诉求。
- 限流、熔断，实现服务间链路稳定、可靠。
- 网络长连接管理降低资源损耗，提升网络吞吐量。
- 服务安全认证：认证、鉴权、审计等，提供服务安全保障基石。

- 图形化应用全景拓扑，流量治理可视化

  服务网格提供了可视化的流量监控，包括链路信息、服务异常响应、超长响应时延等信息，通过图表以及拓扑等多样化形式全面展现业务运行情况。

  服务网格可以结合应用运维管理、应用性能管理服务，提供详细的微服务级流量监控、异常响应流量报告以及调用链信息，能够更快速、更准确的定位问题。

- 性能增强，可靠性增强

  服务网格控制面和数据面基于社区版本的基础上进行更可靠的加固和性能优化。

- 多云多集群、多基础设施

  提供免运维的托管控制面，提供多云多集群的全局统一的服务治理、安全和服务运行监控能力，还提供对容器和虚拟机（VM） 等多种基础设施的统一服务发现和管理。

- 协议扩展

  扩展 Dubbo 协议的支持。

- 传统 SDK 集成

  提供 Spring Cloud、Dubbo 等传统微服务 SDK 的集成解决方案，传统微服务 SDK 开发的业务无需大量代码改造即可快速迁移到云原生容网格运行环境上运行。

Copyright © 2016 - 2023 DaoCloud

3.3 功能总览

此处介绍服务网格支持的功能。

流量治理

- 七层连接池管理

  支持配置 HTTP 最大请求数、最大重试次数、最大等待请求数、每次连接最大请求数以及连接最长空闲时间。
- 四层连接池管理

  支持配置 TCP 最大连接数、连接超时时间、最大无响应次数、最短空闲时间以及健康检查间隔。
- 熔断

  支持配置服务熔断规则，包括实例被驱逐前的连续错误数、检查周期、基础隔离时间以及最大隔离实例比例。
- 重试

  支持配置 HTTP 重试次数、重试超时时间以及重试条件。
- 超时

  支持配置 HTTP 请求超时时间。
- 负载均衡

  支持配置随机调度、轮询调度、最少连接和一致性哈希多种负载均衡算法。
- HTTP Header

  可以灵活添加、修改和删除指定 HTTP Header，包括将 HTTP 请求转发到目标服务之前对 Header 的操作，以及将 HTTP 响应回复给客户端前对 Header 的操作。
- 故障注入

  支持配置延时故障和中断故障。

安全

- 透明双向认证

  支持界面配置服务间的双向认证。
- 细粒度访问授权

  支持通过界面配置服务间的访问授权（后台 API 可以配置 Namespace 级别授权，授权将会给一个特定的接口）。

可观测性

- 流量拓扑

  支持查看网格应用流量拓扑，了解服务间依赖关系。
- 服务运行监控

  支持查看服务访问信息，包括服务和服务各个版本的 QPS 和延时等指标。
- 访问日志

  支持收集和检索服务的访问日志。
- 调用链

  支持非侵入调用链埋点，并可以通过检索调用链数据进行问题定界定位。

Copyright © 2016 - 2023 DaoCloud

多集群模式

- 多集群配置统一管理

  支持网格下多集群的网格配置、集群配置管理；支持不同集群不同粒度的边车注入策略，以及支持跨集群流量策略等对数据面配置的统一管理。
- 可扩展性

  支持一键接入、移除集群

网格数据面微服务框架

- Spring Cloud

  支持 Spring Cloud SDK 开发的服务无侵入式的接入网格，并统一管理。
- Dubbo

  支持 Dubbo SDK 开发的服务无侵入式的接入网格，并统一管理。

兼容性和扩展

- 版本兼容

  API 完全兼容通用服务网格。
- 插件支持

  支持 Tracing、Prometheus、Kiali、Grafana。

Copyright © 2016 - 2023 DaoCloud

3.4 UI 界面支持的功能

DCE 服务网格支持三种网格：托管网格、专有网格、外接网格。 在 UI 界面上，对这三种网格支持的功能如下。

网格管理

子类	托管网格	专有网格	外接网格
创建	√	√	√
配置	√	√	√
删除	√	√	√

网格概览

子类	托管网格	专有网格	外接网格
网格概览	√	√	√

集群管理

子类	托管网格	专有网格	外接网格
接入	√	×	×
移除	√	×	×

服务管理

子类	托管网格	专有网格	外接网格
服务列表	√	√	√
服务详细信息	√	√	√
服务编辑	√	√	√
服务条目（增删改查）	√	√	√

流量治理

子类	托管网格	专有网格	外接网格
虚拟服务（增删改查）	√	√	√
目标规则（增删改查）	√	√	√
网关规则（增删改查）	√	√	√

安全治理

子类	托管网格	专有网格	外接网格
对等身份认证（增删改查）	√	√	√
请求身份认证（增删改查）	√	√	√
授权策略（增删改查）	√	√	√

Copyright © 2016 - 2023 DaoCloud

流量监控

子类	托管网格	专有网格	外接网格
拓扑	√	√	√
图表	√	√	√

边车管理

子类	托管网格	专有网格	外接网格
注入	√	√	√
卸载	√	√	√
资源限制	√	√	×

网格网关

子类	托管网格	专有网格	外接网格
创建	√	√	×
删除	√	√	×
设置	√	√	×

Istio 资源管理

子类	托管网格	专有网格	外接网格
创建	√	√	√
编辑	√	√	√
删除	√	√	√

Copyright © 2016 - 2023 DaoCloud

3.5 服务网格 Release Notes

本页列出服务网格各版本的 Release Notes，便于您了解各版本的演进路径和特性变化。

2023-02-28

v0.13.1

新功能

- 多云网络互联管理及相关接口
- 网格全局配置能力（网格系统配置、全局流量治理能力、全局边车注入设置）增强
- 支持 zookeeper 代理

优化

- Namespace 控制器加上 Cache，当集群中 Namespace 过多时会造成太多的重复的 Get 请求
- 减少正常模式下 ckube 的日志输出

修复

- 由容器管理平台移除集群导致的网格相关逻辑错误
- RegProxy 注入 Sidecar 会导致 Nacos 无法注册的问题
- Spring Cloud 服务识别错误的问题
- 没有同步容器管理平台 Cluster 状态信息
- 流量透传策略 - 流量筛选 IP 段配置未生效
- 流量透传策略 - 流量透传策略无法删除
- 获取网格配置时，由于配置路径不存在导致返回错误

2022-12-29

v0.12.0

新功能

- 流量透传功能的接口实现
- 对 istio 1.15.4、1.16.1 的支持

优化

- 命名空间边车管理页面中增加“未设置”边车策略，避免命名空间层面的边车策略对工作负载层面产生连带影响；
- 发布流水线参数优化
- 优化边车注入、边车资源限制逻辑、避免出现不同步现象

修复

- 网格升级后部分组件未更新的问题
- 网格移除后部分资源未清除的问题 - 边车资源同步不正确，改为通过 Pod 中的 istio-proxy 容器的实际边车资源
- 托管集群无法作为工作集群
- 当边车注入实例为 0/0 时注入情况显示不正确
- 取消边车注入后仍然显示边车相关信息

Copyright © 2016 - 2023 DaoCloud

2022-11-30

v0.11.1

新功能

- `密钥管理` 相关 API
- 在 `虚拟服务`、`目标规则`、`网关规则` 列表新增治理策略标签及相关筛选能力
- 网格中集群健康状态检查能力
- otel sdk 接入
- secret 的多个接口实现
- 对 istio 1.15.3 的支持

优化

- 重工作负载边车注入接口，防止出现问题
- 网格服务监控面板默认展开
- 禁用网格全局监控面板子链接跳转，避免出现界面混乱现象
- 优化网格控制面处理流程，避免更新已经更新过的 object 导致冲突
- 优化网格的集群接入、移除逻辑

修复

- 当控制面 MSpider 升级之后，mcpc 等组件未更新的问题
- 获取集群资源错误
- 获取集群命名空间列表接口集群名称不存在时响应码为 200 情况
- 网格升级流程中前置条件判断有误的问题
- 网格升级未执行 k8s 版本限制的问题
- 边车管理接口工作负载边车资源配置失效的问题
- 因网格内集群的监控检测失败导致集群无法移除的问题
- 数据面的镜像没有被打包到 DCE 5.0 的离线包中
- Ckube 无法自动更新配置的问题

Copyright © 2016 - 2023 DaoCloud

3.6 适用场景

此处介绍服务网格适用的具体场景。

服务流量治理

流量治理是一个非常宽泛的话题，例如：

- 动态修改服务间访问的负载均衡策略，如根据某个请求特征做会话保持。
- 同一个服务有两个版本在线，将一部分流量切到某个版本上。
- 对服务进行保护，例如限制并发连接数、限制请求数、隔离故障服务实例等。
- 动态修改服务中的内容，或者模拟服务运行故障等。

服务网格可以提供非侵入的流量治理能力，无需修改任何业务代码就能实现这些服务治理功能。根据服务的协议，提供策略化、场景化的网络连接管理。可以根据需要对特定服务的特定端口配置不同的治理规则。

场景优势

- 重试

  服务访问失败自动重试，从而提高总体访问成功率和质量。支持配置 HTTP 请求重试次数、重试超时时间和重试条件。
- 超时

  服务访问超时自动处理，快速失败，从而避免资源锁定和请求卡顿。支持配置 HTTP 请求超时时间。
- 连接池

  通过连接池管理，可以对四层协议配置 TCP 的最大连接数、连接超时时间、最大无响应次数、最短空闲时间和健康检查间隔； 可以对七层协议配置 HTTP 最大请求数、最大重试次数、最大等待请求数、每连接最大请求数、连接最大空闲时间，从而防止一个服务的失败级联影响到整个应用。
- 熔断

  通过熔断配置实例被驱逐前的连续错误次数、驱逐间隔时长、最小驱逐时间、最大驱逐比例等参数，从而定期考察被访问的服务实例的工作情况。如果连续出现访问异常，则将服务实例标记为异常并进行隔离，在一段时间内不为其分配流量。 过一段时间后，被隔离的服务实例会再次被解除隔离，尝试处理请求。如果还不正常，则被隔离更长的时间。从而实现异常服务实例的故障隔离和自动故障恢复。
- 负载均衡

  配置各种负载均衡策略，如随机、轮询、最少连接，还可以配置一致性哈希将流量转发到特定的服务实例上。
- HTTP Header

  灵活增加、修改和删除指定 HTTP Header，包括将 HTTP 请求转发到目标服务之前对 Header 进行操作。 还可以在将 HTTP 响应回复给客户端前，对 Header 进行操作，以非侵入方式管理请求内容。
- 故障注入

  通过对选定的服务注入中断故障、延时故障来构造故障场景，无需修改代码即可进行故障测试。

端到端的透明安全

众所周知，将传统的单体应用拆分为一个个微服务固然带来了各种好处，例如：更好的灵活性、可扩缩性、重用性，但微服务也同样面临着特殊的安全需求：

- 为了抵御中间人攻击，需要用到流量加密。
- 为了提供灵活的服务访问控制，需要用到 TLS 和细粒度访问策略。
- 为了决定哪些人在哪些时间可以做哪些事，需要用到审计工具。

面对这些需求服务网格提供全面的安全解决方案，包括身份验证策略、透明的 TLS 加密以及授权和审计工具。

Copyright © 2016 - 2023 DaoCloud

场景优势

- 非侵入安全

  服务网格是以一种安全基础设施的方式向用户提供透明的安全能力，让不涉及安全问题的代码安全运行，让不太懂安全的人可以开发和运维安全的服务，不用修改业务代码就能提供服务访问安全。应用服务网格提供了一个透明的分布式安全层，并提供了底层安全的通信通道，管理服务通信的认证、授权和加密，提供 Pod 到 Pod、服务到服务的通信安全。开发人员在这个安全基础设施层上只需专注于应用程序级别的安全性。

- 多集群安全

  在多集群场景下服务网格提供了全局的服务访问安全。多个集群的网格共享一套根证书，给数据面的服务实例分发密钥和证书对，并定期替换密钥证书，根据需要撤销密钥证书。在服务间访问时，网格的数据面代理就会代理本地服务和对端进行双向认证、通道加密。这里的双向认证的服务双方可以来自两个不同的集群，从而做到跨集群的透明的端到端双向认证。

- 细粒度授权

  在认证的基础上，就可以进行服务间的访问授权管理，可以控制某个服务，或者服务的一个特定接口进行授权管理。如只开放给特定的一个 Namespace 下的服务，或者开放给某个特定的服务。源服务和目标服务可以在不同的集群，甚至源服务的不同实例在不同的集群，目标服务的不同实例在不同的集群。

服务运行监控

运营容器化的基础设施带来了一系列新的挑战。我们需要增强容器、评估 API 端点的性能以及识别出基础设施中的有害部分。服务网格可在不修改代码的情况下实现 API 增强，并且不会带来服务延迟。

服务网格可以为网格内的所有服务通信进行遥测，这种遥测技术提供了服务行为的可观察性，有助于运营商对其应用程序进行故障排除、维护和优化，而不会给服务开发人员带来任何额外负担。通过服务网格，运营商可以全面了解被监控的服务如何与其他服务以及组件本身进行交互。

场景优势

- 非侵入监控数据采集

  在复杂应用的场景下，服务间的访问拓扑、调用链、监控等都是服务访问异常时进行定位定界的必要手段。 服务网格技术的一项重要能力就是以应用非侵入的方式提供这些监控数据的采集，用户只需关注自己的业务开发，无需额外关注监控数据的生成。

- 丰富性能监控能力

  基于网格生成服务访问数据，集成各种不同的性能监控服务，提供跨集群智能的服务运行管理。 包括跨集群的服务调用链、服务访问拓扑和服务运行健康状态、通过跨集群的全局视图来关联服务间的访问状况等。

Copyright © 2016 - 2023 DaoCloud

3.7 什么是服务网格

服务网格是基于 Istio 开源技术构建的面向云原生应用的下一代服务网格。

服务网格是一种具备高性能、高易用性的全托管服务网格产品，通过提供完整的非侵入式的微服务治理方案，能够统一治理多云多集群的复杂环境，以基础设施的方式为用户提供服务流量治理、安全性治理、服务流量监控、以及传统微服务（SpringCloud、Dubbo）接入。

DCE 5.0 的服务网格兼容社区原生 Istio 开源服务网格，提供原生 Istio 接入管理能力。在较高的层次上，服务网格有助于降低服务治理的复杂性，减轻开发运维团队的压力。

服务网格作为 DCE 5.0 产品的体系一员，无缝对接容器管理平台，可以为用户提供开箱即用的上手体验， 并作为基础设施为微服务引擎提供容器微服务治理支持，方便用户通过单一平台对各类微服务系统做统一管理。

服务网格的学习路径如下：

```
flowchart TD

    install([安装部署])
    install --> mesh[创建网格]
        subgraph mesh[创建网格]
            managed[托管网格]
            private[专有网格]
            external[外接网格]

        end

    mesh --> cluster[纳管集群]

    cluster --> inject[注入边车]

        subgraph inject[注入边车]
            global[全局注入]
            namespace[命名空间注入]
            workload[工作负载注入]
        end


    inject -.-> service[服务管理]
    inject -.-> traffic[流量治理]
    inject -.-> security[安全治理]
    inject -.-> sidecar[边车管理]
    inject -.-> watch[流量监控]
    inject -.-> gateway[网格网关]
    inject -.-> upgrade[版本升级]


    service -.-> entry[服务条目]
    traffic -.-> virtual[虚拟服务<br>目标规则<br>网关规则]
    security -.-> peer[对等身份认证<br>请求身份认证<br>授权策略]
    sidecar -.-> sidecarm[命名空间边车管理<br>工作负载边车管理<br>全局边车管理<br>边车流量透传]
    watch -.-> watch2[流量监控<br>流量拓扑]

    classDef plain fill:#ddd,stroke:#fff,stroke-width:1px,color:#000;
    classDef k8s fill:#326ce5,stroke:#fff,stroke-width:1px,color:#fff;
    classDef cluster fill:#fff,stroke:#bbb,stroke-width:1px,color:#326ce5;

    class managed,private,external,global,namespace,workload plain
    class install,service,gateway,traffic,watch,upgrade,security,entry,virtual,peer,cluster,sidecar,sidecarm,watch2 cluster

    click install "https://docs.daocloud.io/mspider/install/"
    click managed "https://docs.daocloud.io/mspider/user-guide/service-mesh/create-mesh/"
    click private "https://docs.daocloud.io/mspider/user-guide/service-mesh/create-mesh/"
    click external "https://docs.daocloud.io/mspider/user-guide/service-mesh/integrate-mesh/"
    click cluster "https://docs.daocloud.io/mspider/user-guide/cluster-management/join-clus/"
    click global "https://docs.daocloud.io/mspider/user-guide/sidecar-management/global-sidecar/"
    click namespace "https://docs.daocloud.io/mspider/user-guide/sidecar-management/ns-sidecar/"
    click workload "https://docs.daocloud.io/mspider/user-guide/sidecar-management/workload-sidecar/"
    click gateway "https://docs.daocloud.io/mspider/user-guide/gateway-instance/create/"
    click service "https://docs.daocloud.io/mspider/user-guide/service-list/"
    click traffic "https://docs.daocloud.io/mspider/user-guide/traffic-governance/"
    click security "https://docs.daocloud.io/mspider/user-guide/security/"
    click watch "https://docs.daocloud.io/mspider/user-guide/traffic-monitor/"
    click upgrade "https://docs.daocloud.io/mspider/user-guide/upgrade/istio-update/"
    click entry "https://docs.daocloud.io/mspider/user-guide/service-list/service-entry/"
    click virtual "https://docs.daocloud.io/mspider/user-guide/traffic-governance/virtual-service/"
    click peer "https://docs.daocloud.io/mspider/user-guide/security/peer/"
    click sidecar "https://docs.daocloud.io/mspider/user-guide/sidecar-management/ns-sidecar/"
    click sidecarm "https://docs.daocloud.io/mspider/user-guide/sidecar-management/passthrough/"
    click watch2 "https://docs.daocloud.io/mspider/user-guide/traffic-monitor/conn-topo/"
```

下载 DCE 5.0{ .md-button .md-button--primary } 安装 DCE 5.0{ .md-button .md-button--primary } 申请社区免费体验{ .md-button .md-button--primary }

Copyright © 2016 - 2023 DaoCloud

4. Test

4.1 OCP 环境接入服务网格配置方法

OCP (OpenShift Container Platform) 是 Red Hat 推出的容器平台。

本页说明服务网格接入 OCP 平台的操作步骤。

SCC 安全策略设置

在 Openshift 集群中，为服务网格增加命名空间的 `privileged` 用户权限，以 istio-operator、istio-system 两个命名空间为例：

```
oc adm policy add-scc-to-user privileged  system:serviceaccount:istio-operator:istio-operator
oc adm policy add-scc-to-user privileged  system:serviceaccount:istio-system:istio-system
```

接入 Openshift 集群

创建一个网格，接入 Openshift 集群。返回网格列表，发现 Openshift 集群已接入成功。

但后端会报错：

```
COMMIT
2022-10-27T07:06:50.610621Z info    Running command: iptables-restore --noflush /tmp/iptables-rules-1666854410610268141.txt1105821213
2022-10-27T07:06:50.616716Z error   Command error output: xtables parameter problem: iptables-restore: unable to initialize table 'nat'

Error occurred at line: 1
Try `iptables-restore -h' or 'iptables-restore --help' for more information.
2022-10-27T07:06:50.616746Z error   Failed to execute: iptables-restore --noflush /tmp/iptables-rules-1666854410610268141.txt1105821213, exit status 2
```

通过以下几步消除错误。

OCP 激活 iptables

修改 YAML

参考以下 YAML，根据实际环境修改部署：

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: dsm-init
  namespace: openshift-sdn
spec:
  revisionHistoryLimit: 10
  selector:
```


Copyright © 2016 - 2023 DaoCloud

```
    matchLabels:
      app: dsm-init
  template:
    metadata:
      labels:
        app: dsm-init
        type: infra
    spec:
      containers:
      - command:
        - /bin/sh
        - -c
        - |
          #!/bin/sh
          set -x
          iptables -t nat -A OUTPUT -m tcp -p tcp -m owner ! --gid-owner 1337 -j REDIRECT --to-ports 15006
          iptables -t nat -D OUTPUT -m tcp -p tcp -m owner ! --gid-owner 1337 -j REDIRECT --to-ports 15006
          while true; do sleep 100d; done
        image: release.daocloud.io/mspider/proxyv2:1.15.0    # 修改proxy的镜像地址
        name: dsm-init
        resources:
          requests:
            cpu: 100m
            memory: 20Mi
        securityContext:
          privileged: true
      dnsPolicy: ClusterFirst
      hostNetwork: true
      hostPID: true
      nodeSelector:
        kubernetes.io/os: linux
      priorityClassName: system-node-critical
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: sdn
      serviceAccountName: sdn
```

添加参数

在 globalmesh YAML 中添加以下一行参数：

```
istio.custom_params.components.cni.enabled: "true"
```

!!! note

```
OpenShift 4.1 以上版本不再使用 iptables，转而使用 nftables。
因此需要安装 istio CNI 插件，否则在边车注入时会出现如下错误，即无法执行 iptables-resotre 命令。

```none
istio iptables-restore: unable to initialize table 'nat'
```
```

部署 istio-cni

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  components:
    cni:
      enabled: true
      namespace: istio-system
  values:
    sidecarInjectorWebhook:
      injectedAnnotations:
        k8s.v1.cni.cncf.io/networks: istio-cni
    cni:
      excludeNamespaces:
        - istio-system
      psp_cluster_role: enabled
      cniBinDir: /var/lib/cni/bin
      cniConfDir: /etc/cni/multus/net.d
      cniConfFileName: istio-cni.conf
      chained: false
```


Copyright © 2016 - 2023 DaoCloud

4.2 兼容性测试报告 - OpenShift

基于 OpenShift，测试服务网格在创建专有网格时的兼容性。

本次测试报告以 OpenShift 4.11 为例，详细阐述了服务网格接入 OpenShift 专有网格模式时的兼容性测试过程。

测试目的

验证 DCE 5.0 服务网格上创建基于原生 istio，从控制面/数据面聚合 OpenShift 单集群模式的专有网格。

测试环境

服务网格环境

基于 demo-dev.daocloud.io

OpenShift 环境

准备了 OpenShift 3 个节点：

HostName	IP	Kubernetes 版本	Crictl 版本	OS 内核版本	OS Release
master01.mspider.openshift.io	10.6.157.111	v1.24.0+3882f8f	1.24.2	4.18.0-372.26.1.el8_6.x86_64	rhcos
master02.mspider.openshift.io	10.6.157.112	v1.24.0+3882f8f	1.24.2	4.18.0-372.26.1.el8_6.x86_64	rhcos
master03.mspider.openshift.io	10.6.157.113	v1.24.0+3882f8f	1.24.2	4.18.0-372.26.1.el8_6.x86_64	rhcos

查看 3 个节点的状况：

```
oc get nodes -owide
```

输出类似于：

```
NAME                          STATUS   ROLES           AGE   VERSION           INTERNAL-IP    EXTERNAL-IP OS-
IMAGE                                                        KERNEL-VERSION                CONTAINER-RUNTIME
master01.mspider.openshift.io Ready    master.worker   7d3h  v1.24.0+3882f8f   10.6.157.111   <none>      Red Hat Enterprise Linux CoreOS 411.86.202209211811-0
(Ootpa)  4.18.0-372.26.1.el8_6.x86_64  cri-o://1.24.2-7.rhaos4.11.gitca400e0.el8
master02.mspider.openshift.io Ready    master.worker   7d2h  v1.24.0+3882f8f   10.6.157.112   <none>      Red Hat Enterprise Linux CoreOS 411.86.202209211811-0
(Ootpa)  4.18.0-372.26.1.el8_6.x86_64  cri-o://1.24.2-7.rhaos4.11.gitca400e0.el8
master03.mspider.openshift.io Ready    master.worker   7d2h  v1.24.0+3882f8f   10.6.157.113   <none>      Red Hat Enterprise Linux CoreOS 411.86.202209211811-0
(Ootpa)  4.18.0-372.26.1.el8_6.x86_64  cri-o://1.24.2-7.rhaos4.11.gitca400e0.el8
```

测试过程

通过容器管理接入 OpenShift

获取 OpenShift 集群的 kubeconfig（~/.kube/config），在 demo-dev 环境的容器管理中接入集群。

```
clusters:
- cluster:
    insecure-skip-tls-verify: true
    server: https://10.6.157.130:6443
  name: mspider
contexts:
- context:
    cluster: mspider
    user: admin
  name: admin
current-context: admin
preferences: {}
users:
- name: admin
  user:
    client-certificate-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURaekNDQWsrZ0F3SUJBZ0lJV0wzelp0MFErbGt3RFFZSktvWklodmNOQVFFTEJRQXdOakVTTUJBR0ExVUUKQ3hNSmIzQmxibk5vYVdaME1TQXdIZ1lEVlFRREV
```


Copyright © 2016 - 2023 DaoCloud

```
    client-key-data:
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBUUVBcG9IU2FnZ1MvaGEzWHdYcGNYdXJ3b0pvMk5hNCt6cEZsZ2k3azFIaUdkb29lZUtWClVENjB1WFFFUDVoR2REREt1UUNYZkxUSDF
```

在容器管理中查看接入集群的状态：

创建专有网格

1. 登录 global-cluster 节点，查看服务网格同步容器管理接入集群的状态。

```
[root@g-master1 ~]# kubectl get mc -n mspider-system
NAME                    ROLE          MESHID   PHASE            KUBERNETES
kpanda-global-cluster   FREE_CLUSTER           CLUSTER_RUNNING  https://100.64.0.1:443
kubean-mgmt             FREE_CLUSTER           CLUSTER_RUNNING  https://172.30.120.200:6443
openshift4-mspider      FREE_CLUSTER           CLUSTER_RUNNING  https://10.6.157.131:6443
[root@g-master1 ~]#
```

2. 完成创建网格时的配置后，点击 确定 。

   对于控制面集群，请选择通过容器管理接入的 openshift4-mspider 集群

Copyright © 2016 - 2023 DaoCloud

3. OpenShift 集群成功接入网格

4. 查看各个组件是否健康

```
[root@msic ~]# oc get iop -n istio-system
NAME                        REVISION   STATUS    AGE
openshift-mspider-mcpc                 HEALTHY   73m
[root@msic ~]# oc get po -n istio-system
NAME                                        READY   STATUS    RESTARTS   AGE
istio-ingressgateway-86f8b95ff4-dm4fz       1/1     Running   0          6m33s
istiod-85b75c457b-fpxg5                     1/1     Running   0          6m22s
mspider-mcpc-ckube-remote-77c7fcb6b8-c72zm  1/1     Running   0          73m
mspider-mcpc-mcpc-controller-56498cb99-t5bpk 1/1    Running   0          73m
[root@msic ~]# oc get po -n istio-operator
NAME                              READY   STATUS    RESTARTS   AGE
istio-operator-58df7989b9-g4wc8   1/1     Running   0          2m36s
[root@msic ~]#
```

```
[root@g-master1 ~]# kubectl get mc -n mspider-system
NAME                    ROLE                MESHID      PHASE            KUBERNETES
kpanda-global-cluster   FREE_CLUSTER                    CLUSTER_RUNNING  https://100.64.0.1:443
kubean-mgmt             FREE_CLUSTER                    CLUSTER_RUNNING  https://172.30.120.200:6443
openshift4-mspider      INTEGRATED_CLUSTER  openshift   MANAGED_SUCCEED  https://10.6.157.131:6443
[root@g-master1 ~]# kubectl get gm -n mspider-system
NAME        MODE        OWNERCLUSTER        DEPLOYNAMESPACE   PHASE      MESHVERSION
openshift   DEDICATED   openshift4-mspider  istio-system      SUCCEEDED  1.15.0
[root@g-master1 ~]#
```

接入时遇到的问题

1. 提示错误 `istio-operator RS CreateFailed`

```
  Type            Status  Reason
  ----            ------  ------
  ReplicaFailure  True    FailedCreate
Events:
  Type     Reason        Age                  From                   Message
  ----     ------        ----                 ----                   -------
  Warning  FailedCreate  3m18s (x13 over 3m39s)  replicaset-controller  Error creating: pods "istio-operator-6b98fdb9ff-" is forbidden: unable to validate against any security context constraint: [provider "anyuid": Forbidden: not usable by user or serviceaccount, spec.co
ntainers[0].securityContext.runAsUser: Invalid value: 1337: must be in the ranges: [1000690000, 1000699999], provider "restricted": Forbidden: not usable by user or serviceaccount, provider "nonroot-v2": Forbidden: not usable by user or serviceaccount, provider "nonroot":
 Forbidden: not usable by user or serviceaccount, provider "hostmount-anyuid": Forbidden: not usable by user or serviceaccount, provider "machine-api-termination-handler": Forbidden: not usable by user or serviceaccount, provider "hostnetwork-v2": Forbidden: not usable by
 user or serviceaccount, provider "hostnetwork": Forbidden: not usable by user or serviceaccount, provider "hostaccess": Forbidden: not usable by user or serviceaccount, provider "node-exporter": Forbidden: not usable by user or serviceaccount, provider "privileged": Forb
idden: not usable by user or serviceaccount]
  Warning  FailedCreate  113s (x9 over 2m2s)    replicaset-controller  Error creating: pods "istio-operator-6b98fdb9ff-" is forbidden: unable to validate against any security context constraint: [provider "anyuid": Forbidden: not usable by user or serviceaccount, spec.co
ntainers[0].securityContext.runAsUser: Invalid value: 1337: must be in the ranges: [1000690000, 1000699999], provider "restricted": Forbidden: not usable by user or serviceaccount, provider "nonroot-v2": Forbidden: not usable by user or serviceaccount, provider "nonroot":
 Forbidden: not usable by user or serviceaccount, provider "hostmount-anyuid": Forbidden: not usable by user or serviceaccount, provider "machine-api-termination-handler": Forbidden: not usable by user or serviceaccount, provider "hostnetwork-v2": Forbidden: not usable by
 user or serviceaccount, provider "hostnetwork": Forbidden: not usable by user or serviceaccount, provider "hostaccess": Forbidden: not usable by user or serviceaccount, provider "node-exporter": Forbidden: not usable by user or serviceaccount, provider "privileged": Forb
idden: not usable by user or serviceaccount]
  Warning  FailedCreate  85s (x12 over 96s)     replicaset-controller  Error creating: pods "istio-operator-6b98fdb9ff-" is forbidden: unable to validate against any security context constraint: [provider "anyuid": Forbidden: not usable by user or serviceaccount, spec.co
ntainers[0].securityContext.runAsUser: Invalid value: 1337: must be in the ranges: [1000690000, 1000699999], provider "restricted": Forbidden: not usable by user or serviceaccount, provider "nonroot-v2": Forbidden: not usable by user or serviceaccount, provider "nonroot":
 Forbidden: not usable by user or serviceaccount, provider "hostmount-anyuid": Forbidden: not usable by user or serviceaccount, provider "machine-api-termination-handler": Forbidden: not usable by user or serviceaccount, provider "hostnetwork-v2": Forbidden: not usable by
 user or serviceaccount, provider "hostnetwork": Forbidden: not usable by user or serviceaccount, provider "hostaccess": Forbidden: not usable by user or serviceaccount, provider "node-exporter": Forbidden: not usable by user or serviceaccount, provider "privileged": Forb
idden: not usable by user or serviceaccount]
  Warning  FailedCreate  38s (x13 over 61s)     replicaset-controller  Error creating: pods "istio-operator-6b98fdb9ff-" is forbidden: unable to validate against any security context constraint: [provider "anyuid": Forbidden: not usable by user or serviceaccount, spec.co
ntainers[0].securityContext.runAsUser: Invalid value: 1337: must be in the ranges: [1000690000, 1000699999], provider "restricted": Forbidden: not usable by user or serviceaccount, provider "nonroot-v2": Forbidden: not usable by user or serviceaccount, provider "nonroot":
 Forbidden: not usable by user or serviceaccount, provider "hostmount-anyuid": Forbidden: not usable by user or serviceaccount, provider "machine-api-termination-handler": Forbidden: not usable by user or serviceaccount, provider "hostnetwork-v2": Forbidden: not usable by
 user or serviceaccount, provider "hostnetwork": Forbidden: not usable by user or serviceaccount, provider "hostaccess": Forbidden: not usable by user or serviceaccount, provider "node-exporter": Forbidden: not usable by user or serviceaccount, provider "privileged": Forb
idden: not usable by user or serviceaccount]
  Warning  FailedCreate  7s (x12 over 17s)      replicaset-controller  Error creating: pods "istio-operator-6b98fdb9ff-" is forbidden: unable to validate against any security context constraint: [provider "anyuid": Forbidden: not usable by user or serviceaccount, spec.co
ntainers[0].securityContext.runAsUser: Invalid value: 1337: must be in the ranges: [1000690000, 1000699999], provider "restricted": Forbidden: not usable by user or serviceaccount, provider "nonroot-v2": Forbidden: not usable by user or serviceaccount, provider "nonroot":
 Forbidden: not usable by user or serviceaccount, provider "hostmount-anyuid": Forbidden: not usable by user or serviceaccount, provider "machine-api-termination-handler": Forbidden: not usable by user or serviceaccount, provider "hostnetwork-v2": Forbidden: not usable by
 user or serviceaccount, provider "hostnetwork": Forbidden: not usable by user or serviceaccount, provider "hostaccess": Forbidden: not usable by user or serviceaccount, provider "node-exporter": Forbidden: not usable by user or serviceaccount, provider "privileged": Forb
idden: not usable by user or serviceaccount]
```

原因分析：OpenShift（安全上下文约束）SCC 对创建 Pod 有权限限制

解决办法：增加 SCC 用户命名空间权限 "privileged"

```sh
oc adm policy add-scc-to-user privileged -z istio-operator
oc adm policy add-scc-to-user privileged -z istio-system
oc adm policy add-scc-to-group privileged system:authenticated
```

重启 `istio-operator rs` 解决问题。

2. 提示错误 `message: no running Istio pods in "istio-system"`

Copyright © 2016 - 2023 DaoCloud

原因分析：字面提示是 istio-system 命名空间下缺少 istiod/istio-ingressgateway service pod。 实际是网格全局边车资源限制中的内存资源参数格式错误；正确格式为 500Mi

```
[root@g-master1 ~]# kubectl get gm -n mspider-system openshift -oyaml
apiVersion: discovery.mspider.io/v3alpha1
kind: GlobalMesh
metadata:
  creationTimestamp: "2022-10-27T05:26:07Z"
  finalizers:
  - gsc-controller
  generation: 2
  name: openshift
  namespace: mspider-system
  resourceVersion: "820271"
  uid: a40318a9-5e39-47f2-9fa5-6ac0bfc50205
spec:
  hub: release.daocloud.io/mspider
  mode: DEDICATED
  ownerCluster: openshift4-mspider
  ownerConfig:
    controlPlaneParams:
      global.istio_version: 1.15.0
      global.mesh_capacity: S50P200
      istio.custom_params.values.global.proxy.resources.limits.cpu: "1.5"
      istio.custom_params.values.global.proxy.resources.limits.memory: "500"
      istio.custom_params.values.global.proxy.resources.requests.cpu: "0.6"
      istio.custom_params.values.global.proxy.resources.requests.memory: "500"
      istio.custom_params.values.sidecarInjectorWebhook.enableNamespacesByDefault: "false"
    deployNamespace: istio-system
status:
```

解决办法：修改 yaml 文件，添加 Mi 单位

shell

kubectl edit gm -n mspider-system openshift -oyaml

```
[root@g-master1 ~]# kubectl get gm -n mspider-system openshift -oyaml
apiVersion: discovery.mspider.io/v3alpha1
kind: GlobalMesh
metadata:
  creationTimestamp: "2022-10-27T05:26:07Z"
  finalizers:
  - gsc-controller
  generation: 3
  name: openshift
  namespace: mspider-system
  resourceVersion: "839985"
  uid: a40318a9-5e39-47f2-9fa5-6ac0bfc50205
spec:
  hub: release.daocloud.io/mspider
  mode: DEDICATED
  ownerCluster: openshift4-mspider
  ownerConfig:
    controlPlaneParams:
      global.istio_version: 1.15.0
      global.mesh_capacity: S50P200
      istio.custom_params.values.global.proxy.resources.limits.cpu: "1.5"
      istio.custom_params.values.global.proxy.resources.limits.memory: "500Mi"
      istio.custom_params.values.global.proxy.resources.requests.cpu: "0.6"
      istio.custom_params.values.global.proxy.resources.requests.memory: "500Mi"
      istio.custom_params.values.sidecarInjectorWebhook.enableNamespacesByDefault: "false"
    deployNamespace: istio-system
status:
```

3. 注入边车时提示 pod Init:CrashLoopBackOff

   激活 ocp iptables

yaml

apiVersion: apps/v1

kind: DaemonSet

metadata:

  name: dsm-init

  namespace: openshift-sdn

spec:

  revisionHistoryLimit: 10

Copyright © 2016 - 2023 DaoCloud

```
  selector:
    matchLabels:
      app: dsm-init
  template:
    metadata:
      labels:
        app: dsm-init
        type: infra
    spec:
      containers:
      - command:
        - /bin/sh
        - -c
        - |
          #!/bin/sh
          set -x
          iptables -t nat -A OUTPUT -m tcp -p tcp -m owner ! --gid-owner 1337 -j REDIRECT --to-ports 15006
          iptables -t nat -D OUTPUT -m tcp -p tcp -m owner ! --gid-owner 1337 -j REDIRECT --to-ports 15006
          while true; do sleep 100d; done
        image: release.daocloud.io/mspider/proxyv2:1.15.0
        name: dsm-init
        resources:
          requests:
            cpu: 100m
            memory: 20Mi
        securityContext:
          privileged: true
      dnsPolicy: ClusterFirst
      hostNetwork: true
      hostPID: true
      nodeSelector:
        kubernetes.io/os: linux
      priorityClassName: system-node-critical
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: sdn
      serviceAccountName: sdn
```

运行命令：

```shell
kubectl edit gm -n mspider-system openshift -oyaml
```

添加：`istio.custom_params.components.cni.enabled: "true"`

Copyright © 2016 - 2023 DaoCloud

部署 bookinfo 应用测试

创建 bookinfo namespace，部署 bookinfo 应用时需要对命名空间添加 SCC。

```
oc adm policy add-scc-to-user privileged -z bookinfo
```


Copyright © 2016 - 2023 DaoCloud

1. 启用命名空间边车注入

2. 终端查看 bookinfo 命名空间 Labels

```shell
oc describe ns bookinfo |grep "istio-injection-enabled"
```

屏幕输出类似于：

```none
Labels: istio-injection-enabled
```

3. 部署 bookinfo 应用

```shell
oc apply -f https://raw.githubusercontent.com/istio/istio/release-1.15/samples/bookinfo/platform/kube/bookinfo.yaml -n bookinfo
```


Copyright © 2016 - 2023 DaoCloud

```
[root@msic ~]# oc apply -f https://raw.githubusercontent.com/istio/istio/release-1.15/samples/bookinfo/platform/kube/bookinfo.yaml -n bookinfo
service/details created
serviceaccount/bookinfo-details created
deployment.apps/details-v1 created
service/ratings created
serviceaccount/bookinfo-ratings created
deployment.apps/ratings-v1 created
service/reviews created
serviceaccount/bookinfo-reviews created
deployment.apps/reviews-v1 created
deployment.apps/reviews-v2 created
deployment.apps/reviews-v3 created
service/productpage created
serviceaccount/bookinfo-productpage created
deployment.apps/productpage-v1 created
[root@msic ~]#
```

shell

```
oc get pod -n bookinfo
```


Copyright © 2016 - 2023 DaoCloud

5. User guide

5.1 Cluster management

集群纳管

服务网格的集群纳管页面为用户提供了集群接入、移除功能。

集群接入是一个集群加入网格管理边界的第一步。

完成接入的集群将出现在边车管理的集群列表中，然后用户可以对该集群的命名空间、工作负载启用边车注入，实现对集群的聚合治理管控。

该功能仅在 托管网格 有效，在 专有网格 和 外接网格 模式下未开放该功能。具体参见服务网格分类。 托管网格采用单控制面多集群的方式，实现对多集群统一获取服务注册信息、下发治理策略，各托管集群的控制面组件仅负责接受转发来自唯一管理集群控制面的策略，不执行任何本地创建的策略。

Copyright © 2016 - 2023 DaoCloud

添加集群

当用户创建一个托管网格后，还未接入任何托管的集群，此时网格处于 `未就绪` 状态，用户需要添加一个或多个集群。

Copyright © 2016 - 2023 DaoCloud

具体操作步骤如下：

1. 在左侧导航栏点击 集群纳管 ，点击 添加集群 按钮。或者在网格列表中，点击最右侧的 ... 按钮，在弹出菜单中选择 添加集群 。

2. 在弹出的集群列表中，勾选待加入的集群后，点击 确定 按钮，完成集群的添加操作。

Copyright © 2016 - 2023 DaoCloud

!!! note

```
1. 仅 **托管网格** 可以添加集群，专有网格无法添加集群。

2. 仅可添加状态为`运行中`的集群至网格，其他状态的集群无法选中，也无法添加。

3. 集群添加的过程会持续几秒，请关注网格列表中的`集群统计`信息，了解正常集群的数量变化。
```


Copyright © 2016 - 2023 DaoCloud

移除集群

如果想要从托管网格中移除一个集群，可以按照以下步骤移除集群。

1. 在左侧导航栏点击 `集群纳管`，在集群列表右侧，点击 `移除` 按钮。或者在网格列表中，点击最右侧的 `...` 按钮，在弹出菜单中选择 `移除`。

2. 在弹出窗口中，确认信息无误后，点击 `确定`。

移除集群，需要完成一些前置操作，例如：

- 卸载已注入的边车。
- 清除集群相关的网格网关。
- 其他前置操作，请按屏幕提示操作。

```
!!! warning
```

移除集群，将无法通过网格集中管理集群，另外集群日志等相关信息可能会丢失，请谨慎操作。

Copyright © 2016 - 2023 DaoCloud

5.2 Gateway instance

管理网格网关

网格网关功能提供了网格网关实例的创建、管理功能，用户可以在该页面管理当前网格内的网格网关实例。

网格网关分为 Ingress（入站）和 Egress（出站）两类。 Ingress 网关用于定义服务网格内应用的流量入口，所有进入服务网格内应用的流量应对经过 Ingress 网关。 Egress 网关用于定义了网格内应用的流量出口，可以让外部服务的流量统一经过 Egress 网关，从而实现更精确的流量控制。 网关实例运行的也是 Envoy，但与边车不同，网格以具体的实例单独运行。

Copyright © 2016 - 2023 DaoCloud

创建网格网关的操作步骤如下。

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏中点击 网格网关 ，进入网关列表，点击右上角的 创建 按钮。

2. 在 创建网格网关 窗口中，按提示配置参数，基本都是必填项。检查配置信息无误后点击 确定 。

3. 返回网格网关列表，屏幕右上角提示创建成功。刚创建的网格网关状态为 创建中 。

Copyright © 2016 - 2023 DaoCloud

4. 几秒后刷新页面，状态将变为 `运行中`，表示网关成功配置。点击列表右侧的 `...`，可以执行编辑和删除操作。

!!! info

更直观的操作演示，可参阅[视频教程](../../../videos/mspider.md)。

Copyright © 2016 - 2023 DaoCloud

删除网格网关

如果想要从网关列表中删除一个集群时，可以按照以下步骤删除集群。

删除一个网关

这是推荐操作，可以防止误删除。

1. 在左侧导航栏点击 网格网关 ，在网格列表右侧点击 `...` 按钮，在弹出菜单中选择 删除 。

2. 在弹出窗口中，确认信息无误后，点击 确定 ，该网关将被删除。

!!! warning

删除网关后，该网关相关的信息将会丢失，请谨慎操作。

Copyright © 2016 - 2023 DaoCloud

批量删除

如果确实需要一次性删除多个网关，可以采用此项操作。

1. 在网关列表中勾选多个网关后，点击右上角的垃圾箱图标。

2. 在弹出窗口中，确认信息无误后，点击 `确定` ，选中的网关将全部被删除。

!!! warning

如非必要，请勿使用批量删除功能，删除的网关无法恢复，请谨慎操作。

Copyright © 2016 - 2023 DaoCloud

5.3 Mesh config

Istio 资源管理

`Istio 资源管理` 页面按资源类型列出了 Istio 的所有资源，为用户提供了各类资源的展示、创建、编辑、删除等能力。

本页面提供了以下几种资源类型：

Copyright © 2016 - 2023 DaoCloud

流量治理类

资源类型	描述
DestinationRule（目标规则）	目标规则是服务治理中重要的组成部分，目标规则通过端口、服务版本等方式对请求流量进行划分，并对各请求分流量订制 Envoy 流量策略，应用到流量的策略不仅有负载均衡，还有最小连接数、熔断等。
EnvoyFilter	该资源提供了对 Envoy 配置的能力，可以定义新的过滤器、监听器\集群等。使用该资源时需谨慎，错误配置可能会对整个网格环境的稳定性造成较大影响。 注意： - EnvoyFilter 可以配置在Istio根目录（对所有工作负载生效）或某个工作负载（使用工作负载选择标签）； - 当多个 EnvoyFilter 作用于同一个工作负载时，会按创建顺序优先执行；
Gateway（网关规则）	网关规则用于定义网格边缘的负载均衡器，用于将服务暴露于网格之外，或提供内部服务的对外访问。相较于k8s的ingress对象，istio的网关规则增加了更多的功能： l L4-L6负载均衡 l 对外mTLS l SNI的支持 l 其他istio中已经实现的内部网络功能： Fault Injection，Traffic Shifting， Circuit Breaking， Mirroring 对于L7的支持，网关规则通过与虚拟服务配合实现。
ProxyConfig	用于暴露代理的配置选项，例如：代理的线程数。该资源为可选资源，如不创建，系统将使用内建默认值； 注意： - ProxyConfig 中任何配置变更需要重启相关工作负载才会生效； - 作用于网格或命名空间的 ProxyConfig 不可以包含任何工作负载选择标签，否则将仅作用于选定的工作负载；
WorkloadEntry	该资源提供了对非k8s标准工作负载的描述，例如：运行于虚拟机的工作负载。WorkloadEntry需要与服务条目配合使用，由服务条目通过筛选器建立服务与工作负载之间的对应关系；
ServiceEntry（服务条目）	服务条目允许向Istio的内部服务注册表中添加其他条目，以便网格中自动发现的服务可以访问/路由到这些手动指定的服务。服务条目描述服务的属性包括：DNS名称、VIP、端口、协议、端点等。这些服务可以是网格外部的（例如：web API）或网格内部无法自动注册的服务（例如：一个数据库，几个VM）。
SideCar	Sidecar用于描述边车对工作负载实例之间的流量转发配置。默认情况下，Istio将支持转发网格中所有工作负载实例之间的通信，并接受与工作负载相关的所有端口流量。 注意： - 根命名空间下的 SideCar 对所有没有配置 SideCar 的命名空间及工作负载生效； - 任意命名空间或工作负载如果存在多个 SideCar，将被定义为行为未定义（无生效资源）；
VirtualService（虚拟服务）	在虚拟服务中，提供了 HTTP、TCP、TLS 三种协议的路由支持，可以通过多种匹配方式（端口、host、header 等）实现对不同的地域、用户请求做路由转发，分发至特定的服务版本中，按权重比划分负载，并提供了故障注入、流量镜像等多种治理工具。
WorkloadGroup	WorkloadGroup 描述了工作负载实例的集合，定义了工作负载实例引导代理的规范细节，包括元数据和标识。 WorkloadGroup 仅用于非 Kubernetes 工作负载，模拟 Kubernetes 中边车注入和部署方式，引导 Istio 代理，

安全治理类

资源类型	描述
AuthorizationPolicy（授权策略）	授权策略类似于一种四层到七层的“防火墙”，它会像传统防火墙一样，对数据流进行分析和匹配，然后执行相应的动作。无论是来自外部的请求，或是网格内服务间的请求，都适用授权策略。
PeerAuthentication（对等身份认证）	对等身份认证提供服务间的双向安全认证，同时密钥以及证书的创建、分发、轮转也都由系统自动完成，对用户透明，从而大大降低了安全配置管理的复杂度。
RequestAuthentication（请求身份认证）	请求身份认证用于外部用户对网格内部服务发起请求的认证，用户使用jwt实现请求加密；每个请求身份认证需要配置一个授权策略

代理扩展类

资源类型	描述
WasmPlugin	WasmPlugin 通过WebAssembly过滤器为Istio代理提供扩展功能，其提供的过滤器能力可以和Istio内部过滤器形成复杂交互关系。

Copyright © 2016 - 2023 DaoCloud

系统设置类

资源类型	描述
Telementry	该资源定义了如何为网格内的工作负载生成遥测，为Istio提供了指标、日志、全链路追踪三项可观测性工具的配置。 注意：创建在Istio根目录并且不包含工作负载选择器的遥测资源将对网格全局生效。

Copyright © 2016 - 2023 DaoCloud

UI 操作示例

用户可以用 YAML 形式对资源进行增删改查操作。此处是一个 telementry 遥测资源的创建/删除示例。

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏中点击 `网格配置` -> `Istio 资源管理`，点击右上角的 `YAML 创建` 按钮。

2. 在 YAML 创建页面中，输入正确的 YAML 语句后点击 `确定`。

```yaml
apiVersion: telemetry.istio.io/v1alpha1
kind: Telemetry
metadata:
  name: namespace-metrics
  namespace: default
spec:
  # 未指定选择算符，应用到命名空间中的所有工作负载
  metrics:
  - providers:
```


Copyright © 2016 - 2023 DaoCloud

```
    - name: prometheus
  overrides:
    - tagOverrides:
        request_method:
          value: "request.method"
        request_host:
          value: "request.host"
```

3. 返回资源列表，点击操作一列的 ⋮ 按钮，可以从弹出菜单中选择编辑和删除等更多操作。

Copyright © 2016 - 2023 DaoCloud

5.4 Multicluster

多云网络互联

多云网络互联是在多集群模式下，多集群之间网络互不相通（Pod 无法直接建立通讯）的情况下提供的一套解决方案，可以快速地将多集群的网络连通，实现跨集群的 Pod 互相访问。

使用场景

如果您的环境满足以下条件，可尝试请使用多云网络互联功能：

1. 同一个网格有多个集群
2. 多集群的 Pod 网络无法直接连通（包括 Pod CIDR 冲突等情况）

!!! note

```
多云网络互联仅适用于托管网格。
```

名词解释

- 网络分组：
- 用于辨别网格内集群的网络拓扑关系。
- 如果某些集群的网络能够直接互通（跨集群的 Pod IP 没有冲突且能够直接连通），那么这些集群应该被放在一个网络分组里，否则将属于不同的网络分组。
- 网络分组和集群的网络 CNI 实现类型没有明确的关联关系。
- 东西网关：
- 用于不同网络分组间通信互联，一个网络分组内可以创建多个。
- 用户可以在网络分组下任意集群创建东西网关，当移动集群至其他网络分组时，东西网关也会随之迁移。
- 东西网关所用负载均衡 IP 通常有集群所在的云平台自动分配，也可以由用户配置设定。
- 基本设置：创建用于东西网关的 网关规则 配置，该配置将开放 15443 端口用于网络分组间的通信，用户可以在 网关规则 列表中查看 CRD 文件详细内容，但不建议修改，可能会导致网络通信失败。
- 网络分组列表：
- 展示当前创建的网络分组及分组下集群、东西网关信息；
- 在该列表下用户可以对网络分组、分组下集群及东西网关执行增删操作，并可以在网络分组间迁移集群。
- 互联列表：
- 包含至少一个东西网关的网络分组可以加入互联列表，加入互联的网络分组可以和其他网络分组通信。
- 已加入互联的网络分组可以修改负载均衡 IP，但如需修改其他配置（例如增删集群、东西网关），需要先移除 网络分组互联 列表。
- 加入互联和移除互联会引起网格控制面的重启，建议谨慎操作。

操作流程

建议操作流程如下图所示

Copyright © 2016 - 2023 DaoCloud

操作步骤

1. 进入某个网格，点击启用开关，将自动创建用于东西网关的 `网关规则` 。点击 `创建网络分组` 按钮，为网络分组添加至少一个集群。

2. 为网络分组填入名称，并添加至少一个集群。

Copyright © 2016 - 2023 DaoCloud

!!! note

网络分组名称创建后无法修改。

3. 增删集群。

创建网络分组后，即可为分组内添加更多的集群，同一分组的集群需确保处于同一网络类型并可以互通，否则可能造成互联失败。

添加完成后即可在网络分组下看到多个集群。

Copyright © 2016 - 2023 DaoCloud

!!! note

网络分组中至少要保留一个集群，如需移除所有集群，请删除网络分组。

4. 创建/删除东西网关。

   东西网关用于网络分组间通信，可以在分组内的任一集群上创建一个或多个网关。点击一个集群右侧的 ⋮，选择 编辑东西网关：

Copyright © 2016 - 2023 DaoCloud

创建配置项如下：

- 负载均衡注释 为可选设置，部分云平台会以注释方式提供负载均衡 IP 分配，请参考云平台提供的技术文档；
- 东西网关副本数 默认为 1，如果需要提高网关可用性，可创建多个副本；点击 确定 ；

选择一个东西网关，点击操作中的 删除东西网关 ，可以删除该网关，如图所示：

!!! note

每个网络分组至少需要创建一个东西网关，否则无法加入互联。

Copyright © 2016 - 2023 DaoCloud

5. 网络分组互联

Copyright © 2016 - 2023 DaoCloud

a. 如果要将一个网络分组加入互联，点击 加入互联 按钮。

b. 勾选一个网络分组，点击 下一步 。

c. 选择可用的 东西网关地址 ，点击 确定 。

Copyright © 2016 - 2023 DaoCloud

d. 在互联列表中可以看到新加入的网络分组，列表内的分组之间彼此建立了通信。

!!! note

- 没有东西网关的网络分组无法加入互联
- 加入分组互联操作会引起网格控制面重启，建议加入互联前确保网络分组配置无误

其他操作

1. 更新东西网关地址。

   用户可以对处于互联状态的网络分组增删东西网关地址。在互联列表中勾选一个分组，点击 更新东西网关地址 。

   更新地址后点击 确定 。

Copyright © 2016 - 2023 DaoCloud

2. 移除互联

如果网络分组不再需要和其他网络分组建立互联，或需要调整分组内集群及东西网关设置，可以移除互联。 选择需要移除的网络分组，点击 移除互联 按钮，出现下图对话框。

经过二次确认，网络分组即可移除互联列表。

3. 删除网络分组

在需要删除的网络分组操作下拉框中点击 删除网络分组 ，即可删除所选网络分组。

4. 关闭 多云网络互联 功能

在顶部 基本设置 区域，点击 启用 状态的滑块，将弹出关闭多云互联的前提条件：

Copyright © 2016 - 2023 DaoCloud

先移除网络分组：

在文本框输入 `关闭基本设置` 进行确认后，即可关闭多云互联功能：

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

多集群灰度发布

多集群灰度发布的实现逻辑是，通过在不同集群部署不同的业务应用，然后通过服务网格配置相应的策略实现业务版本之间流量调整，然后根据运行情况下线版本。

前置准备，参考网格多云部署文档，搭建网格基础设施。

创建 demo 应用并且验证流量

集群部署 V1 版本 HELLOWORLD

首先选择一个命名空间（gray-demo），并且在网格页面开启该命名空间边车注入。

在应用工作台部署应用，在这里我们以 istio 的 helloworld 为例。

选择对应集群（mdemo-cluster2）与命名空间，并且配置工作负载基本信息。

Copyright © 2016 - 2023 DaoCloud

- 选择镜像：docker.m.daocloud.io/istio/examples-helloworld-v1
- 版本：latest

配置服务信息：

- 访问类型：集群内访问
- 端口配置
- 协议：TCP
- 名称：http-5000
- 容器端口：5000
- 服务端口：5000

Copyright © 2016 - 2023 DaoCloud

为了区分不同版本的工作负载，需要在 `容器管理平台` 找到对应的工作负载，点击 `标签与注解`，给容器组标签添加键值对： "version"："v1"

Copyright © 2016 - 2023 DaoCloud

集群部署 V2 版本 HELLOWORLD

注意服务命名与命名空间必须相同。

首先选择上面一致命名空间（gray-demo），并且在网格页面开启该命名空间边车注入。

部署流程与上面一样，其主要区别是：

- 镜像发生了变化：`docker.m.daocloud.io/istio/examples-helloworld-v2`
- 在容器平台给相应 容器组标签 加上 label "version"："v2"

Copyright © 2016 - 2023 DaoCloud

部署灰度应用策略

多集群目标规则

首先创建 DestinationRule，通过定义 SubSet 定义不同集群的业务版本。

其标签键值对为上文中添加的容器组标签：`version: <VERSION>`。

策略：必须开启 Istio 双向 TLS

必须开启 Istio 双向 TLS 模式

Copyright © 2016 - 2023 DaoCloud

目标规则 YAML 如下：

```
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: helloworld
  namespace: gray-demo
spec:
  host: helloworld
  subsets:
    - labels:
        version: v1
      name: v1
      trafficPolicy:
        tls:
          mode: ISTIO_MUTUAL
    - labels:
        version: v2
      name: v2
      trafficPolicy:
        tls:
          mode: ISTIO_MUTUAL
  trafficPolicy:
    tls:
      mode: ISTIO_MUTUAL
```

通过 INGRESS 暴露服务

首先需要创建一条网关规则：

Copyright © 2016 - 2023 DaoCloud

网关规则 YAML 如下：

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: gray-demo-gateway
  namespace: gray-demo
spec:
  selector:
    istio: ingressgateway # use istio default controller
  servers:
    - port:
        number: 80
        name: http
        protocol: HTTP
      hosts:
        - "*"
```

然后配置访问服务所需的虚拟服务规则。

Copyright © 2016 - 2023 DaoCloud

虚拟服务 YAML 如下：

```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: gray-demo-vs
  namespace: gray-demo
spec:
  gateways:
    - gray-demo-gateway
```


Copyright © 2016 - 2023 DaoCloud

```
  hosts:
    - "*"
  http:
    - name: http-5000
      route:
        - destination:
            host: helloworld.gray-demo.svc.cluster.local
            port:
              number: 5000
            subset: v1
          weight: 80
        - destination:
            host: helloworld.gray-demo.svc.cluster.local
            port:
              number: 5000
            subset: v2
          weight: 20
```

验证

INGRESS_LB_IP 是指 Ingress 网格负载均衡地址，可以在容器管理平台中查看，如果没有有效的负载均衡 IP 可以通过 NodePort 方式访问。

（由于容器管理平台界面无法直接查看外部 IP，因此使用控制台的能力查看）

在浏览器中访问： http://${INGRESS_LB_IP}/hello

确认其 v1 与 v2 版本的访问比例是否与上面策略的比例为 8:2

Copyright © 2016 - 2023 DaoCloud

多集群网络通讯原理与架构分析

近年来，多集群架构已经成为“老生常谈”。我们喜欢高可用，喜欢异地多可用区，而多集群架构天生就具备了这样的能力。 另一方面我们也希望通过多集群混合云来降低成本，利用到不同集群各自的优势和特性，以便使用不同集群的最新技术与支持现实中越来越复杂的云上业务。 最为本质的问题时单集群模式下存在承载能力上限，并且集群故障时，无法快速故障转移。

就是因为这些种种原因，多集群几乎成为了云计算的新潮流。 随着新架构的诞生，新的难题也接踵而来，尤其是在多集群场景下，其中业务的治理、监控、网络通讯也面临着严重的考验。 这时不得不提到服务网格，该技术从理论上来说对于多云多集群场景下，从设计基因中就自带了解决上面问题的能力。

为什么说服务网格能够支持复杂的多云场景呢？ 服务网格将自身能力下沉到基础设施层面，因此从理论上来说，在多云环境中，它天然能够感知自己云上业务的各种信息，比如说服务地址、服务之间链路关系等网络信息， 网格将感知的信息聚会成一个完整的多云网络拓扑，在一个已知且能控制的多云拓扑中，用户能够从高纬度的观测并且控制业务通讯。

上述都只理论分析，再多的理论分析也不如一个实践论证，因此我们将自身服务网格产品（Mspider）到底如何支持多云环境进行一个由浅入深的整体剖析。

多云服务网格的优势

首先，我们回到最初的起点，我们为什么要选择多云服务网格？ 所有的技术都是为了解决问题而存在，不具有优势与解决问题的技术，是虚假且没有意义的。 因此我们简单介绍一下相对于单集群场景下，多云场景下的服务网格能够给我们带来哪些优势？

故障隔离与故障转移

在单集群场景下，业务服务 A 出现问题时，我们通过将业务流量转移到同集群的另一个业务实例上。 但是如果集群出现故障时，这时候会变成多个业务同时故障，这时候如何将该集群的所有业务转移到另一个集群上，这是已经非常费时且复杂地。 但是实际地生产环境下，每分每秒是非常昂贵。 但是多云场景下，集群维度的故障转移是基本能力，用户能够快速地转移其所有业务的流量。 用户只需要定义集群之间故障转移的策略，例如 A 集群故障时，将业务流量转移到集群 B 中。

地域感知与故障转移

当多云环境下，对云上的地域进行定义；在业务健康时，可以让业务之间通讯以就近原则，发生故障时，可以快速转移至最近的地域环境中。

地域感知对于多云的重要性不言而喻。 因为现实世界、实际地域区域会影响网络通讯，中间有时间空间资源等多种因素。 例如当中国北京的集群都故障时，随机将业务转移到美国，但是实际环境中中国上海的集群空闲在那里，这肯定是不合理的。

多维度团队或项目隔离

虽然随着多云的体量扩大，但是团队的复杂度并没有增加，团队可以将多云划分成不同的区域、不同的集群、不同业务项目。 团队拥有多云能力的情况下，可以只需要管理自己相关的集群与业务。

多云维度的观测能力

单集群模式下，用户只能知道当前集群内的业务运行情况，每个集群之间的业务通讯时不透明且独立的。用户无法感知到集群外的业务通讯链路。

但是多云场景下，只要属于同一个网格，网格能将同一网络下不同集群之间的通讯情况进行聚合分析，将其绘制成一张完整的多云网络拓扑。 好比原先的单集群只是一张 2D 的照片，现在多云场景下，服务网格能够提供一张 3D 立体的拓扑网络。 这不仅仅是范围的扩大，而是维度层次上的增加。

多云下网络通讯原理

在了解了多云场景下巨大的优势，我们回归现实，多云场景有伴随着哪些问题？ 其中哪个问题最为根本？ 首先我们从字面上理解一下这个意思，多个云环境？多个云厂商？ 多个云集群？这里的多个，就意味着存在不同。 而且在多云世界里，网络通讯是一个非常硬核的问题，云厂商之间能够直接通讯吗？ 多集群之间如何通讯无阻？

在单集群的情况，所以的工作负载都在同一网络上（集群内部网络），但是多集群的场景下，每个集群可能存在多种因素导致网络无法互通，这里列举常见的几种情况：

- 地域问题导致集群实际网络之间存在网络隔离
- 初始化集群时，保持默认配置导致集群子网都一样，因此集群与集群之间网络会存在 IP 冲突。 以上几种情况下的多集群之间网络通讯是存在问题的，为了保持集群内部网络的独立性以及集群网络之间网络打通，Istio 提供了两种网络模型，用户可以根据集群状态与用户需求自由选择不同的网络模型。

Copyright © 2016 - 2023 DaoCloud

单一网络模型

顾名思义，在单一网络的情况下，多集群之间的网络实例能给直接通讯，这时候不需要网格进行任何操作，服务之间网络通讯能够直达，其架构图如下：

单一网络模型的优势是非常明显的，用户通讯时，无需复杂的配置和中转，因此网络解析也更加直观，因此也能带来更高的网络性能。

在这时不得不提的是，有的用户会问，如果拥有多个集群，并且当时多集群之间网络存在冲突，但是又想要多集群之间网络选择单一网络模型时如何做？在这里简单提供一些网络改造需要解决的问题：

- 子网冲突问题：需要解决集群实例网络与集群服务网络的子网冲突问题
- 网络隔离问题：集群之间网络可能由于地域、安全等多种因素造成网络隔离的现状，用户可以选择通过隧道或直接路由，解决跨多个集群的 Pod IP 路由。

多网络模型

在多网络模型中，不同网络中的工作负载实例不能够直接通讯，因此服务网格提供了一种解决方案，让业务实例通过一个或多个 Mesh 网关相互访问，从而解决工作实例网络不能互相通讯的问题。

在服务通讯时，网格会根据网络分区进行服务发现，当请求服务与目标服务不属于同一网络分区时，服务请求将会转发给目标分区的东西网关，通过东西网关路由到真实的目标服务。

Copyright © 2016 - 2023 DaoCloud

多云服务网格架构的抉择

在上述网络方面的思考以后，用户可能对服务网格自身架构存在好奇，例如服务网格控制面组件位置在哪里？ 如何运行多集群模式？ 这架构拥有哪些优势或者缺陷？ 让我们具体分析现有服务网格提供的架构，现在主要有如下三种架构：

- 单集群单控制面
- 多集群单控制面
- 多集群多控制面

单集群单控制面

在最简单的情况下，可以在单一集群上使用控制平面运行网格。

Copyright © 2016 - 2023 DaoCloud

多集群单控制面

多集群部署也可以共享控制平面实例。 在这种情况下，在核心控制集群部署核心的共享控制面实例，其网格核心的通讯策略都由 共享控制面实例 控制。 在从集群中，其实也存在一个从控制面实例，该实例其核心能力是 控制本集群的边车生命周期管理 。

Copyright © 2016 - 2023 DaoCloud

思考一个问题，从集群的边车如何与共享控制面建立连接？其回答时共享控制面需要暴露自身控制面服务。

在单一网络模型中主集群的共享控制平面可以通过稳定 IP(例如集群 IP)访问。

但是在多网络模型的多集群中，需要通过 Mesh 网关对外 暴露 共享控制平面。

Copyright © 2016 - 2023 DaoCloud

在实际的生产环境，可以根据实际的云供应商或者内部网络的规划灵活选择； 例如内部负载平衡器，避免将控制面暴露到公共网络中，因为其存在潜在的安全隐患。

共享控制面在上面描述中都是属于某一个网格集群中，其实其远程共享控制面也可以部署在网格外的集群中， 这样能够让网格的控制面与数据面在物理上实现隔离，避免主集群的控制面与数据面同时出现问题，可以分散风险。

多集群多控制面

为了获得高可用性，您可以在多个集群、区或地域之间部署多个控制平面实例。

在具有多个共享控制面的多集群场景下，每个控制面都属于某个集群， 共享控制面接受用户定义的配置（例如 `Service` 、`ServiceEntry`、`DestinationRule` 等）来则自身集群的 Kubernetes API Server。 因此每个主控制面集群都拥有独立的配置来源。

因此这里存在一个问题，多个控制面集群如何同步配置？ 这是一个很复杂的问题，需要增加另外的同步操作。 在大型的生产环境可能需要配合 CI/CD 工具一起自动化该过程，实现配置同步。

Copyright © 2016 - 2023 DaoCloud

在这种多控制面模型下的在部署难度与配置复杂度上需要付出很大代价。 但是也能收获更高的回报，其核心优势在于：

- 高可用 ：如果控制平面不可用，故障范围仅限于由该控制平面管理的群集中的工作负载。
- 配置隔离 ：您可以在一个群集，区域或地区中进行配置更改，而不会影响其他群集。
- 控制面分区域滚动发布 ：能够在不同的控制面区域内，对网格控制面进行滚动发布，包括金丝雀发布网格控制面，其影响范围也只是该控制面所在的区域。

多云环境下的服务发现

了解了服务网格控制面架构以后，这时候我们不得不提到服务网格数据面的通讯与运行原理， 理解这些原理能提高我们在实际运行服务网格遇到问题的解决能力。其中服务网格数据面通讯里面最为核心的技术便是服务发现能力。

在单集群模式下，服务网格控制面会从集群注册中心获取所有注册在集群中的服务，然后将集群注册中心获取到的服务信息聚合成一个服务端点列表，向每个边车提供。

Copyright © 2016 - 2023 DaoCloud

在多群集场景下的服务发现能力更加复杂，因为集群之间存在注册中心隔离，网格为了运行多集群采用的解决方案是服务网格控制平面会观察并且记录每个集群中的注册中心的服务， 然后将集群中的 同命名空间下 的 同名服务 进行聚合，最后梳理成一张完整的 网格服务端点列表 ，然后下发给网格内的边车。因此服务才能具有发现其他集群服务端点的能力。

因此网格控制面为了能够访问其控制面的多集群的注册中心，在部署阶段需要从集群 授权远程密钥 给所属的控制面集群。 授权以后控制面才能对多集群进行服务发现，因此才具有后续的跨集群负载均衡能力。

在多集群网格部署模型下，多集群服务默认的策略是：每个集群均衡负载。 但是在复杂且庞大的生产环境中，其实很多服务只需要在某些区域进行流量通讯，这时候就可以采用本地优先负载均衡策略（具体方式为参考 Istio 官方提到的 Locality Load Balancing）。

在某些场景下我们会发现，跨集群的流量负载能力并不是频繁操作，时时刻刻的跨集群负载也不是我们所期望的。 例如说我们只是需要进行蓝绿部署，其版本位于不同的集群中； 例如当某个集群的某个服务出现故障时，能够切换到备用集群的服务中。 这些场景都是偶发的。发生频率很低，但是又是刚需。

Copyright © 2016 - 2023 DaoCloud

这时候我们怎么样才能通过网格实现呢？在 Istio 服务网格下有两种方式：

1. 我们可以不交换集群的 `API Server` 远程密钥，这样的话集群只能进行自身集群内的服务发现。 如果需要进行跨集群流量负载，可以通过 `ServiceEntry` 配合外部负载器的方式实现。

2. 通过配置 `VirtualService` 与 `DestinationRule` 策略，禁用多集群之间的流量负载。

服务网格技术方案

上述几个章节我们简单围绕服务网格三个核心维度（网络模型、控制面架构、数据面服务发现）分析其相关原理与优势。 但是这时候我们会发现在网络模型和控制面架构部分介绍时都存在多种方案的情形，那么我们如何选择具体那种方案呢？

DaoCloud 深耕在服务网格产品多年的 ，拥有实际落地客户，以及从单集群演变到多集群客户解决方案的经验，我们产品内部也经过了多种方案的验证与实践，采集多位客户的场景，最后将我们的多集群服务网格产品的架构选型方案如下：自由的网络模型 + 多集群单控制面。

网络模型

在选择网格模型落地时，其实并不是非 A 即 B。 比如说用户需要更高效果，其打通部分集群之间网络成本在用户容忍范围内，用户可以选择部分集群网络为单一网络模式，在无法打通集群网络的集群中选择多网络模式。

每种模型都有自己独特的优势与代价，那么我能不能同时拥有呢？ 这个答案时肯定的，我们都要。所以我们提出一个全能网络模型的概念，用户可以根据自己的需要选择网络模型：

- 单一网络
- 多网络
- 单一网络 + 多网络

这样丰富的网络模型方案，让我们能够更加贴合用户需求解决实际问题。

Copyright © 2016 - 2023 DaoCloud

多集群单控制面

到这一部分，很多人会提问为什么选择多集群单控制面模式呢？从上述文章看起来多集群多控制面提供的能力不是更加丰富吗？

这里不得不提到实现多集群多控制面在落地的时候存在一个非常痛苦的点：多控制面从部署到运用到维护的成本非常高。 我们怎么得出这个结论的呢？在早期我们内部选择了多集群多控制面的验证，但是在这个过程中发现多控制面存在一些弊端：

- 策略复杂高隐患：
- 配置复杂：不同的控制面需要不同的策略，虽然换个角度看是独立的控制面策略，但是在多集群场景下，集群肯定是简单的一个两个，应用的总数也会随着集群的扩大变得更加复杂，非常容易发生部分集群忘记配置，或者更改配置时需要重复配置多套相同的策略。
- 策略冲突：多套控制面的策略非常多，用户需要准确掌握每个策略，不然很容易造成策略冲突。
- 控制面部署与升级维护成本高：当存在多套控制面时，我们需要维护多套控制面的部署与升级。
- 污染多个控制面集群：由于网格需要聚会多集群命名空间与服务，会污染控制面集群。让我举例说明：当集群 A 拥有命名空间 N1、N2；集群 B 拥有命名空间 N3 时；且集群 A、B 都控制面集群时，集群 A 需要新增 N3，集群 B 需要新增 N1、N2。当多集群与集群中业务到达一定体量时，这是非常恐怖的。

但是在多集群单控制面模式下能够大部分避免上面存在的问题，尤其是其中治理策略复杂这块，所有的策略将会统一在一个集群中，因此其中配置难度与管理难度将大大降低。

服务网格架构

接下来让我们来了解一下服务网格的实际架构：

在原生的多集群单控制面下其实并不能解决上文提到的污染多个控制面集群问题，但是我们仔细看上面的架构，大家会发现在这部分架构与社区提供的方案有一个很大的不同，在控制面集群中多了一套服务网格组件，这一套组件的最大的特色是提供了一个虚拟集群，该集群与网格控制面连接，并且目的是将网格资源与控制面集群的资源进行隔离，避免产生脏资源，这时候就能完美解决治理资源污染集群的问题，并且避免用户误操作删除控制面策略资源。

Copyright © 2016 - 2023 DaoCloud

总结一下服务网格技术选型方案的的优势：

- 更简单的管理 ： 将整个服务网格的管理集中在一个控制平面中，可以帮助组织更方便、更快速地管理整个服务网格。
- 低复杂度的配置与更好的性能 ： 通过将整个服务网格的所有数据都统一在一个控制平面中，可以更快速地处理服务的路由和治理策略，从而提高整个服务网格的性能。
- 更强的安全性 ：在单控制面多集群模式中，我们可以更好地控制对整个服务网格的访问，并使用更严格的安全策略来保护其数据。
- 支持跨集群负载与容灾能力 ：能够灵活的配置跨集群流量负载与容灾策略。
- 策略配置统一高效 ：不需要配置复杂的控制面配置同步与合并问题，尤其是控制面越多，其业务服务的体量越大，其同步和合并一定会是一个非常复杂且存在风险的存在。
- 灵活的控制面与数据面隔离 ：服务网格允许用户将网格控制面与数据面集群进行隔离，并且在用户资源有限的情况下也支持集群同时兼具控制面与数据面在同一集群下。
- 网络模式灵活切换 ：服务网格默认场景是多集群位于单网络，但是多网络模式也能够支持部署。
- 网格资源隔离 ：在开源模式下，控制面集群需要管理所有集群的网格策略，这时候会存在污染控制面集群的命名空间，或者部分网格资源会受到集群本身操作影响（例如某个命名空间被删除）。

参考

本文部分图片来自 Istio 官网与 DaoCloud 文档网站。

参阅 Istio 部署模型。

Copyright © 2016 - 2023 DaoCloud

网格多云部署

本页说明如何在多云环境下部署服务网格。

前置条件

集群要求

1. 集群类型与版本 ：明确当前集群的类型与版本，确保后续安装的服务网格能够在该集群中正常运行。
2. 提供可靠的 IP ：控制面集群必须提供一个可靠的 IP 给其他数据面集群来访问控制面。
3. 授权 ：加入网格的集群需要提供一个拥有足够权限的远程秘钥，允许 Mspider 对集群进行安装组件以及 Istio 控制面访问其他集群的 API Server。

多集群区域规划

在 Istio 中，Region、Zone 和 SubZone 是用于维护多集群部署中的服务可用性的概念。具体来说：

- Region 表示一个大区域，通常用于表示整个云提供商的数据中心区域；在 Kubernetes 中，标签 `topology.kubernetes.io/region` 确定节点的区域。
- Zone 表示一个小区域，通常用于表示数据中心中的一个子区域；在 Kubernetes 中，标签 `topology.kubernetes.io/zone` 确定节点的区域。
- SubZone 则是一个更小的区域，用于表示 Zone 中的一个更小的部分。Kubernetes 中不存在分区的概念，因此 Istio 引入了自定义节点标签 `topology.istio.io/subzone` 来定义分区。

这些概念的作用主要在于帮助 Istio 管理不同区域间的服务可用性。例如，在多集群部署中，如果一个服务在 Zone A 中出现故障，Istio 可以通过配置自动将服务流量转移到 Zone B，从而保证服务可用性。

配置方式是通过在集群的 每个节点 添加相应的 Label：

区域	Label
Region	topology.kubernetes.io/region
Zone	topology.kubernetes.io/zone
SubZone	topology.istio.io/subzone

添加 Label 可以通过容器管理平台找到对应的集群，给相应节点配置 Label

Copyright © 2016 - 2023 DaoCloud

网格规划

网格基础信息

- 网格 ID
- 网格版本
- 网格集群

网络规划

确认集群之间网络状态，根据不同状态配置 这部分主要在于多网络模式两部分的配置：

- 规划网络 ID
- 东西网格的部署与配置
- 如何将网格控制面暴露给其他工作集群。

网络模式有如下两种：

- 单网络模式 > 明确集群之间 Pod 网络是否能够直接通。 > 如果 Pod 网络能够直接通讯，证明是同网络模式，但是要注意如果其中 Pod 网络出现冲突 ，就需要选择不同网络模式
- 多网络模式 > 如果集群之间网络不通，需要给集群划分 网络 ID ，并且需要不同网络区域的集群安装东西网关， > 并且配置相关配置，具体操作步骤在下面章节不同网络模式的网格组件安装与配置中。

规划表单

将上述所提到的集群与网格相关规划汇聚成一张表单，便于用户更加方便的参考。

Copyright © 2016 - 2023 DaoCloud

集群规划

集群名	集群类型	集群 pod 子网 (podSubnet)	pod 网络互通关系	集群节点 & 网络区域	集群版本	Master IP
mdemo-cluster1	标准 k8s	"10.201.0.0/16"	-	master: region1/zone1/subzone1	1.25	10.64.30.130
mdemo-cluster2	标准 k8s	"10.202.0.0/16"	-	master: region1/zone2/subzone2	1.25	10.6.230.5
mdemo-cluster3	标准 k8s	"10.100.0.0/16"	-	master: region1/zone3/subzone3	1.25	10.64.30.131

网格规划

配置项	值
网格 ID	mdemo-mesh
网格模式	托管模式
网络模式	多网络模式（需要安装东西网关，规划网络 ID）
网格版本	1.15.3-mspider
托管集群	mdemo-cluster1
工作集群	mdemo-cluster1、mdemo-cluster2、mdemo-cluster3

网络规划

由于上面的表单所知，集群之间不存在网络互通的情况，因此网格为多网络模式，需要规划如下配置：

集群名	集群网格角色	集群网络标识（网络 ID）	hosted-istiod LB IP	eastwest LB IP	ingress LB IP
mdemo-cluster1	托管集群，工作集群	network-c1	10.64.30.71	10.64.30.73	10.64.30.72
mdemo-cluster2	工作集群	network-c2	-	10.6.136.29	-
mdemo-cluster3	工作集群	network-c3	-	10.64.30.77	-

接入集群以及组件准备

用户需要准备符合要求的集群，集群可以新创建（创建集群也可以采用容器管理平台的创建能力），也可以是已经存在的集群。

但是后续网格所需的集群都必须将其接入容器管理平台。

接入集群

如果不是通过容器管理平台创建的集群，例如已经存在的集群，或者通过自定义方式（类似 kubeadm 或 Kind 集群）创建的集群都需要将集群接入容器管理平台。

Copyright © 2016 - 2023 DaoCloud

确认可观测组件（可选）

网格的关键能力可观测性，其中需要关键的观测性组件便是 Insight Agent，因此如果需要拥有网格的观测能力，需要安装其组件。

容器管理平台创建集群方式创建的集群将默认安装 Insight Agent 组件。

其他方式需要在容器管理界面中，找到本集群中的 `Helm 应用`，选择 `insight-agent` 安装。

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

网格部署

通过服务网格创建网格，并且将规划的集群加入到对应的网格中。

创建网格

首先在网格管理页面 -> 创建网格：

Copyright © 2016 - 2023 DaoCloud

创建网格的具体参数如图显示：

1. 选择托管网格：多云环境下，只有托管网格模式能够纳管多集群
2. 输入唯一的网格名称
3. 按照前置条件环境，选择预选的符合要求的网格版本
4. 选择托管控制面所在的集群
5. 负载均衡 IP：该参数为暴露控制面的 Istiod 所需要的参数，需要预先准备
6. 镜像仓库：在私有云中，需要将网格所需的镜像上传仓库，公有云建议填入 `release.daocloud.io/mspider`

网格处于创建中，需要等待网格创建完成后，状态由 `创建中` 转变成 `运行中`；

暴露网格托管控制面 HOSTED ISTIOD

确认托管网格控制面服务

确保网格状态正常后，观察控制面集群 `mdemo-cluster1` 的 `istio-system` 下面的 Service 是否都成功绑定了 LoadBalancer IP。

Copyright © 2016 - 2023 DaoCloud

发现托管网格控制面的服务 `istiod-mdemo-cluster-hosted-lb` 的没有分配 LoadBalancer IP 需要额外处理。

分配 EXTERNAL IP

在不同的环境申请或者分配 LoadBalancer IP 的方式不同，尤其是公有云环境，需要根据公有云厂商所提供的方式去创建 LoadBalancer IP。

本文演示 demo 采用了 metallb 的方式，给所属 LoadBalancer Service 分配 IP，相关部署与配置参考Metallb 安装配置部分。

部署完 metallb 以后，再次确认托管网格控制面服务。

验证托管控制面 Istiod `EXTERNAL IP` 是否通畅

在非托管集群环境中验证托管控制面 Istiod，本次实践通过 curl 的方式验证，如果返回 400 错误，基本可以判定网络已经打通：

```
hosted_istiod_ip="10.64.30.71"
curl -I "${hosted_istiod_ip}:443"
# HTTP/1.0 400 Bad Request
```

确认并且配置网格托管控制面 Istiod 参数

1. 获取托管网格控制面服务 `EXTERNAL IP`

在网格 `mdemo-mesh` 控制面集群 `mdemo-cluster1` 中去确认托管网格控制面服务 `istiod-mdemo-mesh-hosted-lb` 已经分配 LoadBalancer IP 以后，并且记录其 IP，示例如下：

Copyright © 2016 - 2023 DaoCloud

确认托管网格控制面服务 `istiod-mdemo-mesh-hosted-lb` `EXTERNAL-IP` 为 `10.64.30.72`。

2. 手动配置网格托管控制面 Istiod 参数

首先，在容器管理平台进入全局控制面集群 `kpanda-global-cluster`（如果无法确认相关集群的位置，可以询问相应负责人或者通过查询全局服务集群）

-> 自定义资源模块 搜索资源 `GlobalMesh` -> 接下来在 `mspider-system` 找到对应的网格 `mdemo-mesh` -> 然后编辑 YAML

- 在 YAML 中 `.spec.ownerConfig.controlPlaneParams` 字段增加 `istio.custom_params.values.global.remotePilotAddress` 参数；
- 其值为上文中记录的 `istiod-mdemo-mesh-hosted-lb` `EXTERNAL-IP` 地址：`10.64.30.72`。

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

添加工作集群

在服务网格的图形界面上添加集群。

Copyright © 2016 - 2023 DaoCloud

1. 等待网格控制面创建成功后，选中对应网格，进入网格管理页面 -> 集群纳管 -> 添加集群：

2. 选中所需的工作集群后，等待集群安装网格组件完成；

3. 在接入过程中，集群状态会由 接入中 转变成 接入成功：

Copyright © 2016 - 2023 DaoCloud

检测多云控制面是否正常

由于当前工作集群与网格控制面集群 Pod 网络不同，需要通过上文暴露网格托管控制面 Hosted Istiod部分将控制面的 Istiod 暴露给公网。

验证工作集群 Istio 相关组件是否能运行正常，需要在工作集群中检查 `istio-system` 命名空间下的 `istio-ingressgateway` 是否能够正常运行：

不同网络模式的网格组件安装与配置

在本部分主要分为两个部分：

1. 给所有工作集群配置 `网络 ID`
2. 在所有网络不互通的集群中安装东西网关

这里先提到一个问题：为什么需要安装东西网关呢？ 由于工作集群之间 Pod 网格无法直达，因此服务跨集群通讯时也会出现网络问题，Istio 提供了一个解决方案就是东西网关。 当目标服务位于不同网络时，其流量将会转发到目标集群的东西网关，东西网关将解析请求，将请求转发到真正的目标服务。

由上面对东西网关的原理理解以后，又有一个新的问题，Istio 如何区分服务在什么网络中呢？ Istio 要求用户每个工作集群安装 Istio 时，显示的定义 `网络 ID`，这也是第一步分存在的原因。

手动为工作集群配置 `网络 ID`

由于工作集群网络的不同，需要手动给每个工作集群配置 `网络 ID`。 如果在实际环境中，集群之间 Pod 网络能够相互直达，就可以配置成同一个 `网络 ID`。

Copyright © 2016 - 2023 DaoCloud

让我们开始配置 `网络 ID` ，其具体流程如下：

Copyright © 2016 - 2023 DaoCloud

1. 首先进入全局控制面集群 `kpanda-global-cluster`（如果无法确认相关集群的位置，可以询问相应负责人或者通过查询全局服务集群）
2. 然后在 `自定义资源模块` -> 搜索资源 `MeshCluster`
3. `mspider-system` 命名空间下找到加入网格的工作集群，本次案例的工作集群有：`mdemo-cluster2` 、 `mdemo-cluster3`
4. 以 `mdemo-cluster2` 为例，编辑 YAML

- 找到字段 `.spec.meshedParams[].params`， 给其中参数列增加 `网络 ID` 字段
- 参数列的注意事项：
- 需要确认 `global.meshID: mdemo-mesh` 是否为同一个网格 ID
- 需要确认集群角色 `global.clusterRole: HOSTED_WORKLOAD_CLUSTER` 是否为工作集群
- 添加参数 `istio.custom_params.values.global.network`，其值按照最初的规划表单中的网络 ID：`network-c2`

Copyright © 2016 - 2023 DaoCloud

重复上述步骤，给所有工作集群（`mdemo-cluster1`、`mdemo-cluster2`、`mdemo-cluster3`）加上 `网络 ID`。

为工作集群的 `ISTIO-SYSTEM` 标识 `网络 ID`

进入容器管理平台，进入对应的工作集群：`mdemo-cluster1`、`mdemo-cluster2`、`mdemo-cluster3` 的命名空间添加网络的标签。

- 标签 Key：`topology.istio.io/network`
- 标签 value：`${CLUSTER_NET}`

下面以 mdemo-cluster3 为例，找到 `命名空间`，选中 `istio-system` -> `修改标签`。

Copyright © 2016 - 2023 DaoCloud

手动安装东西网关

创建网关实例

确认所有工作集群中的 Istio 相关组件都就绪以后，开始安装东西网关。

在工作集群中通过 IstioOperator 资源安装东西网关，东西网关的 YAML 如下：

!!! note

```
一定要根据当前集群的 `网络 ID` 修改参数。
```

```bash linenums="1"
```


Copyright © 2016 - 2023 DaoCloud

cluster1

CLUSTER_NET_ID=network-c1 CLUSTER=mdemo-cluster1 LB_IP=10.64.30.73

cluster2

CLUSTER_NET_ID=network-c2 CLUSTER=mdemo-cluster2 LB_IP=10.6.136.29

cluster3

CLUSTER_NET_ID=network-c3 CLUSTER=mdemo-cluster3 LB_IP=10.64.30.77

MESH_ID=mdemo-mesh HUB=release-ci.daocloud.io/mspider ISTIO_VERSION=1.15.3-mspider cat < eastwest-iop.yaml apiVersion: install.istio.io/v1alpha1 kind: IstioOperator metadata: name: eastwest namespace: istio-system spec: components: ingressGateways: - enabled: true k8s: env: - name: ISTIO_META_REQUESTED_NETWORK_VIEW value: ${CLUSTER_NET_ID} # 修改为当前集群的网络 ID service: loadBalancerIP: ${LB_IP} # 修改为本集群为东西网关规划的 LB IP ports: - name: tls port: 15443 targetPort: 15443 type: LoadBalancer label: app: istio-eastwestgateway istio: eastwestgateway topology.istio.io/network: ${CLUSTER_NET_ID} # 修改为当前集群的网络 ID name: istio-eastwestgateway profile: empty tag: ${ISTIO_VERSION} values: gateways: istio_ingressgateway: injectionTemplate: gateway global: network: ${CLUSTER_NET_ID} # 修改为当前集群的网络 ID hub: ${HUB} # 可选，如果无法翻墙或者私有仓库，请修改 meshID: ${MESH_ID} # 修改为当前集群的 网格 ID（Mesh ID） multiCluster: clusterName: ${CLUSTER} # 修改为当前 EOF

```
其创建方式为：

1. 在容器管理平台进入相应的工作集群
2. `自定义资源`模块搜索 `IstioOperator`
3. 选中 `istio-system` 命名空间
4. 点击`创建 YAML`

![创建网关实例](images/create-ew.png)

![创建网关实例](images/create-ew2.png)

### 创建东西网关 Gateway 资源

在网格的`网关规则`中创建规则：

```yaml
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: cross-network-gateway
  namespace: istio-system
spec:
  selector:
    istio: eastwestgateway
  servers:
    - hosts:
        - "*.local"
      port:
        name: tls
        number: 15443
        protocol: TLS
      tls:
        mode: AUTO_PASSTHROUGH
```

设置网格全局网络配置

在安装完东西网关以及网关的解析规则以后，需要再所有集群中声明网格中的东西网关配置。 在 容器管理平台进入全局控制面集群 `kpanda-global-cluster`（如果无法确认相关集群的位置，可以询问相应负责人或者通过查询全局服务集群）

-> 在 `自定义资源` 部分搜索资源 `GlobalMesh` -> 接下来在 `mspider-system` 找到对应的网格 `mdemo-mesh` -> 然后编辑 YAML

在 YAML 中 `.spec.ownerConfig.controlPlaneParams` 字段增加一系列 `istio.custom_params.values.global.meshNetworks` 参数

```
# ！！该两行配置缺一不可
# 格式：istio.custom_params.values.global.meshNetworks.${CLUSTER_NET_ID}.gateways[0].address
#      istio.custom_params.values.global.meshNetworks.${CLUSTER_NET_ID}.gateways[0].port

istio.custom_params.values.global.meshNetworks.network-c1.gateways[0].address: 10.64.30.73  # cluster1
istio.custom_params.values.global.meshNetworks.network-c1.gateways[0].port: '15443'  # cluster3 东西网关端口
istio.custom_params.values.global.meshNetworks.network-c2.gateways[0].address: 10.6.136.29  # cluster2
istio.custom_params.values.global.meshNetworks.network-c2.gateways[0].port: '15443'  # cluster2  东西网格端口
istio.custom_params.values.global.meshNetworks.network-c3.gateways[0].address: 10.64.30.77  # cluster3
istio.custom_params.values.global.meshNetworks.network-c3.gateways[0].port: '15443'  # cluster3 东西网关端口
```


Copyright © 2016 - 2023 DaoCloud
```

网络连通性 demo 应用与验证

部署 DEMO

主要是两个应用：helloworld 与 sleep（该两个 demo 属于 Istio 提供的测试应用）

集群部署情况：

集群	helloworld 与版本	sleep
mdemo-cluster1	:heart: VERSION=vc1	:heart:
mdemo-cluster1	:heart: VERSION=vc2	:heart:
mdemo-cluster1	:heart: VERSION=vc3	:heart:

容器管理平台部署 demo

推荐使用容器管理平台创建对应工作负载与应用，在容器管理平台找到对应集群，进入【控制台】执行下面操作。

下面以 mdemo-cluster1 部署 helloworld vc1 为例：

Copyright © 2016 - 2023 DaoCloud

其中每个集群需要注意的点：

- 镜像地址：
- helloworld: docker.m.daocloud.io/istio/examples-helloworld-v1
- Sleep: curlimages/curl
- helloworld 工作负载增加对应 label
- app：helloworld
- version：${VERSION}
- helloworld 工作负载增加对应的版本环境变量
- SERVICE_VERSION: ${VERSION}

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

命令行部署 demo

部署过程中需要用到的配置文件分别有：

- gen-helloworld.sh
- sleep.yaml

```bash linenums="1"

## ---------- cluster1 ----------

kubectl create namespace sample kubectl label namespace sample istio-injection=enabled

## deploy helloworld vc1

bash gen-helloworld.sh --version vc1 | sed -e "s/docker.io/docker.m.daocloud.io/" | kubectl apply -f - -n sample

## deploy sleep

kubectl apply -f sleep.yaml -n sample

## ---------- cluster2 ----------

kubectl create namespace sample kubectl label namespace sample istio-injection=enabled

## deploy helloworld vc2

bash gen-helloworld.sh --version vc2 | sed -e "s/docker.io/docker.m.daocloud.io/" | kubectl apply -f - -n sample

## deploy sleep

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


Copyright © 2016 - 2023 DaoCloud

---------- cluster3 ----------

```
kubectl create namespace sample kubectl label namespace sample istio-injection=enabled
```

deploy helloworld vc3

```
bash gen-helloworld.sh --version vc3 | sed -e "s/docker.io/docker.m.daocloud.io/" | kubectl apply -f - -n sample
```

deploy sleep

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

```
### 验证 demo 集群网络

```bash linenums="1"
# 任意选择一个集群执行
while true; do kubectl exec -n sample -c sleep  \
               "$(kubectl get pod -n sample -l app=sleep -o jsonpath='{.items[0].metadata.name}')"    \
              -- curl -sS helloworld.sample:5000/hello; done

# 预期结果会轮询三个集群的不同版本，结果如下：
# Hello version: vc2, instance: helloworld-vc2-5d67bc48d8-2tpxd
# Hello version: vc2, instance: helloworld-vc2-5d67bc48d8-2tpxd
# Hello version: vc2, instance: helloworld-vc2-5d67bc48d8-2tpxd
# Hello version: vc2, instance: helloworld-vc2-5d67bc48d8-2tpxd
# Hello version: vc1, instance: helloworld-vc1-846f79cc4d-2d8kd
# Hello version: vc1, instance: helloworld-vc1-846f79cc4d-2d8kd
# Hello version: vc1, instance: helloworld-vc1-846f79cc4d-2d8kd
# Hello version: vc3, instance: helloworld-vc3-55b7b5869f-trl8v
# Hello version: vc1, instance: helloworld-vc1-846f79cc4d-2d8kd
# Hello version: vc3, instance: helloworld-vc3-55b7b5869f-trl8v
# Hello version: vc2, instance: helloworld-vc2-5d67bc48d8-2tpxd
# Hello version: vc1, instance: helloworld-vc1-846f79cc4d-2d8kd
# Hello version: vc2, instance: helloworld-vc2-5d67bc48d8-2tpxd
```
```

拓展

其他创建集群方式

通过容器管理创建集群

集群创建可以存在多种方式，推荐使用容器管理中的创建集群功能，但是用户可以选择其他创建方式，本文提供的其他方案可以参考拓展章节的其他创建集群方式

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

可以灵活的选择集群需要拓展的组件，网格的可观测能力必须依赖 Insight-agent

Copyright © 2016 - 2023 DaoCloud

如果集群需要定义更多的集群高级配置，可以在本步骤添加。

创建集群需要等待 30 分钟左右。

Copyright © 2016 - 2023 DaoCloud

通过 kubeadm 创建集群

```
kubeadm init --image-repository registry.aliyuncs.com/google_containers \
             --apiserver-advertise-address=10.64.30.131 \
             --service-cidr=10.111.0.0/16 \
             --pod-network-cidr=10.100.0.0/16 \
             --cri-socket /var/run/cri-dockerd.sock
```

创建 kind 集群

```bash linenums="1" cat < kind-cluster1.yaml kind: Cluster apiVersion: kind.x-k8s.io/v1alpha4 networking: podSubnet: "10.102.0.0/16" # 安装网络规划阶段，定义规划的 pod 子网 serviceSubnet: "10.122.0.0/16" # 安装网络规划阶段，定义规划的 service 子网 apiServerAddress: 10.6.136.22 nodes: - role: control-plane image: kindest/node:v1.25.3@sha256:f52781bc0d7a19fb6c405c2af83abfeb311f130707a0e219175677e366cc45d1 extraPortMappings: - containerPort: 35443 # 如果无法申请 LoadBalancer 可以临时采用 nodePort 方式 hostPort: 35443 # set the bind address on the host

EOF cat < kind-cluster2.yaml kind: Cluster apiVersion: kind.x-k8s.io/v1alpha4 networking: podSubnet: "10.103.0.0/16" # 安装网络规划阶段，定义规划的 pod 子网 serviceSubnet: "10.133.0.0/16" # 安装网络规划阶段，定义规划的 service 子网 apiServerAddress: 10.6.136.22 nodes: - role: control-plane image: kindest/node:v1.25.3@sha256:f52781bc0d7a19fb6c405c2af83abfeb311f130707a0e219175677e366cc45d1 extraPortMappings: - containerPort: 35443 # 如果无法申请 LoadBalancer 可以临时采用 nodePort 方式 hostPort: 35444 # set the bind address on the host

EOF

kind create cluster --config kind-cluster1.yaml --name mdemo-kcluster1 kind create cluster --config kind-cluster2.yaml --name mdemo-kcluster2
```

```
### Metallb 安装配置

#### demo 集群 metallb 网络池规划记录

| 集群名          | IP 池                     | IP 分配情况 |
| -------------- | ------------------------- | ----------- |
| mdemo-cluster1 | 10.64.30.71-10.64.30.73 | -           |
| mdemo-cluster2 | 10.6.136.25-10.6.136.29 | -           |
| mdemo-cluster3 | 10.64.30.75-10.64.30.77 | -           |

#### 安装

#### 容器管理平台 Helm 安装

推荐使用容器管理平台中 `Helm 应用` -> `Helm 模板` -> 找到 metallb -> `安装`。
```


Copyright © 2016 - 2023 DaoCloud

```
![安装 metallb](images/install-metallb-from-helm.png)

![安装 metallb](images/install-metallb-from-helm1.png)

##### 手动安装

参阅 [MetalLB 官方文档](https://metallb.org/installation/)。

注意：如果集群的 CNI 使用的是 calico,你需要禁用 calico 的 BGP 模式，否则会影响 MetalLB 的正常工作。

```bash linenums="1"
# 如果 kube-proxy 使用的是 IPVS 模式，你需要启用 staticARP
# see what changes would be made, returns nonzero returncode if different
kubectl get configmap kube-proxy -n kube-system -o yaml | \
sed -e "s/strictARP: false/strictARP: true/" | \
kubectl diff -f - -n kube-system

# actually apply the changes, returns nonzero returncode on errors only
kubectl get configmap kube-proxy -n kube-system -o yaml | \
sed -e "s/strictARP: false/strictARP: true/" | \
kubectl apply -f - -n kube-system

# 部署 metallb
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.13.7/config/manifests/metallb-native.yaml

# 检查 pod 运行状态
kubectl get pods -n metallb-system

# 配置 metallb
cat << EOF | kubectl apply -f -
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: first-pool
  namespace: metallb-system
spec:
  addresses:
  - 10.64.30.75-10.64.30.77  # 根据规划修改
EOF
cat << EOF | kubectl apply -f -
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: example
  namespace: metallb-system
spec:
  ipAddressPools:
  - first-pool
  interfaces:
  - enp1s0
  # - ens192  # 根据不同机器的网卡命名不同，请修改
EOF
```

增加给对应服务增加指定 IP

```
kubectl annotate service -n istio-system istiod-mdemo-mesh-hosted-lb metallb.universe.tf/address-pool='first-pool'
```

验证

```bash linenums="1" kubectl create deploy nginx --image docker.m.daocloud.io/nginx:latest --port 80 -n default kubectl expose deployment nginx --name nginx-lb --port 8080 --target-port 80 --type LoadBalancer -n default

## 获取对应服务 EXTERNAL-IP

kubectl get svc -n default

## NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

## kubernetes ClusterIP 10.212.0.1 443/TCP 25h

## nginx-lb LoadBalancer 10.212.249.64 10.6.230.71 8080:31881/TCP 10s

curl -I 10.6.230.71:8080
```


Copyright © 2016 - 2023 DaoCloud
```

HTTP/1.1 200 OK

Server: nginx/1.21.6

Date: Wed, 02 Mar 2022 15:31:15 GMT

Content-Type: text/html

Content-Length: 615

Last-Modified: Tue, 25 Jan 2022 15:03:52 GMT

Connection: keep-alive

ETag: "61f01158-267"

Accept-Ranges: bytes

```

#### 查询全局服务集群

通过容器管理的集群列表界面，通过搜索 `集群角色：全局服务集群` 。

Copyright © 2016 - 2023 DaoCloud
```

5.5 Security

安全治理

Istio 提供了一种授权机制和两种认证方式（请求身份认证和对等身份认证）， 用户可以在服务网格中通过向导和 YAML 编写两种方式创建、编辑资源文件， 并可以针对网格全局、命名空间、工作负载三个层面创建规则。当资源创建成功后，Istiod 将转换为配置分发至边车代理执行。

Copyright © 2016 - 2023 DaoCloud

授权策略

授权策略类似于一种四层到七层的“防火墙”，它会像传统防火墙一样，对数据流进行分析和匹配，然后执行相应的动作。无论是来自内部还是外部的请求，都适用授权策略。

Copyright © 2016 - 2023 DaoCloud

服务网格提供了两种创建方式：向导和 YAML。通过向导创建的具体操作步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏点击 安全治理 -> 授权策略，点击右上角的 创建 按钮。

2. 在 创建授权策略 界面中，先进行基本配置后点击 下一步 。

3. 按屏幕提示进行策略设置后，点击 确定 。

Copyright © 2016 - 2023 DaoCloud

4. 返回授权列表，屏幕提示创建成功。

5. 在列表右侧，点击操作一列的 ⋮，可通过弹出菜单进行更多操作。

Copyright © 2016 - 2023 DaoCloud

!!! note

```
- 具体参数的配置，请参阅[安全治理参数配置](./params.md)。
- 更直观的操作演示，可参阅[视频教程](../../../videos/mspider.md)。
```


Copyright © 2016 - 2023 DaoCloud

安全治理参数配置

本页介绍有关对等身份认证、请求身份认证、授权策略相关的参数配置。

对等身份认证

采用向导向导模式时，对等身份认证分为基本配置和认证设置两步，各参数说明如下。

基本配置

UI 项	YAML 字段	描述
名称	metadata.name	必填。对等身份认证名称，同一个命名空间下不可重名。
命名空间	metadata.namespace	必选。对等身份认证所属的命名空间。当选择网格的根命名空间时，将创建全局策略。全局策略仅能创建一个，需在界面中做检查，避免用户重复创建。
工作负载标签	spec.selector	可选。应用对等身份认证策略的工作负载选择标签，可添加多个标签，无需排序。
标签名称	spec.selector.matchLabels	必填。由小写字母、数字、连字符（-）、下划线（_）及小数点（.）组成
标签值	spec.selector.matchLabels.{标签名称}	必填。由小写字母、数字、连字符（-）、下划线（_）及小数点（.）组成

认证设置

UI 项	YAML 字段	描述
mTLS 模式	spec.mTLS.mode	必填。用于设定命名空间的 mTLS 模式： - PERMISSIVE：明文和 mTLS 连接 - STRICT：仅 mTLS 连接 - DISABLE：仅明文连接
为指定端口添加 mTLS 模式	spec.portLevelMtls	可选。针对指定端口设置 mTLS 规则，可添加多条规则，无需排序。 - PERMISSIVE：明文和 mTLS 连接 - STRICT：仅 mTLS 连接 - DISABLE：仅明文连接

请求身份认证

采用向导向导模式时，请求身份认证分为基本配置和认证设置两步，各参数说明如下。

基本配置

UI 项	YAML 字段	描述
名称	metadata.name	必填。请求身份认证名称，同一个命名空间下不可重名。
命名空间	metadata.namespace	必选。请求身份认证所属的命名空间。当选择网格的根命名空间时，将创建全局策略。全局策略仅能创建一个，需在界面中做检查，避免用户重复创建。 同一个命名空间内，请求身份认证的名称不能重复。
工作负载标签	spec.selector	可选。应用请求身份认证策略的工作负载选择标签，可添加多条选择标签，无需排序。
标签名称	spec.selector.matchLabels	必填。由小写字母、数字、连字符（-）、下划线（_）及小数点（.）组成
标签值	spec.selector.matchLabels.{标签名称}	必填。由小写字母、数字、连字符（-）、下划线（_）及小数点（.）组成

Copyright © 2016 - 2023 DaoCloud

认证设置

UI 项	YAML 字段	描述
添加 JWT 规则	spec.jwtRules	可选。用于用户请求认证的 JWT 规则，可添加多条规则。
Issuer	spec.jwtRules.issuers	必填。JSON Web Token (JWT) 签发人信息。
Audiences	spec.jwtRules.issuers.Audiences	可选。配置可访问的 audiences 列表，如果为空，将访问 service name。
jwkURI	spec.jwtRules.issuers.jwkUri	可选。JSON Web Key (JWK) 的 JSON 文件路径，格式为：URI。
jwks	spec.jwtRules.issuers.jwks	可选。JSON Web Key Set (JWKS) 文件内容，可以与 jwkURI 二选一提供。

授权策略

采用向导向导模式时，授权策略的创建分为 `基本配置` 和 `策略设置` 两步，各参数说明如下。

基本配置

元素	YAML 字段	描述
名称	metadata.name	必填。授权的策略名称。
命名空间	metadata.namespace	必选。授权策略所属命名空间，当选择网格根命名空间时，将创建全局策略，全局策略仅能创建一个，需在界面做检查，避免用户重复创建。同一个命名空间内，请求身份认证不可重名。
工作负载标签	spec.selector	可选。应用授权策略的工作负载选择标签，可添加多条选择标签，无需排序。
标签名称	spec.selector.matchLabels	可选。由小写字母、数字、连字符（-）、下划线（_）及小数点（.）组成。
标签值	spec.selector.matchLabels.{标签名称}	可选。由小写字母、数字、连字符（-）、下划线（_）及小数点（.）组成。

策略设置

元素	YAML 字段	描述
策略动作	spec.action	可选。包含： - 允许（allow） - 拒绝（deny） - 审计（audit） - 自定义（custom） 选择自定义时，增加 `provider` 输入项。
Provider	spec.provider.name	必填。仅在 `策略动作` 选择为 `自定义` 时，才显示该输入框。
请求策略	spec.rules	可选。包含请求来源、请求操作、策略条件三部分，可添加多条，按顺序执行。
添加请求来源	spec.rules.-from	可选。请求来源可基于命名空间、IP 段等进行定义，可添加多条。各项参数参见下文 请求来源 Source。
添加请求操作	spec.rules.-to	可选。请求操作是对筛选出的请求执行的操作，例如发送至指定端口或主机，可添加多个操作。各项参数参见下文请求操作 Operation。
添加策略条件	spec.rules.-when	必填。策略条件是一个可选设置，可以增加类似黑名单（values）、白名单的限制条件（notValues），可添加多个策略条件。各项参数参见下文策略条件 Condition。

请求来源 Source

Source 指定一个请求的来源身份。对请求来源中的字段执行逻辑与运算。

Copyright © 2016 - 2023 DaoCloud

例如，以下 Source 将匹配：

- 主体为 “admin” 或 “dev”
- 命名空间为 “prod” 或 “test”
- 且 ip 不是 “1.2.3.4”.

```
principals: ["admin", "dev"]
namespaces: ["prod", "test"]
notIpBlocks: ["1.2.3.4"]
```

字段	类型	描述
`principals`	`string[]`	可选。从对等证书衍生的对等身份列表。对等身份的格式为 `"<TRUST_DOMAIN>/ns/<NAMESPACE>/sa/<SERVICE_ACCOUNT>"`，例如 `"cluster.local/ns/default/sa/productpage"`。此字段要求启用 mTLS，且等同于 `source.principal` 属性。如果不设置，则允许所有主体。
`notPrincipals`	`string[]`	可选。对等身份的反向匹配列表。
`requestPrincipals`	`string[]`	可选。从 JWT 派生的请求身份列表。请求身份的格式为 `"<ISS>/<SUB>"`，例如 `"example.com/sub-1"`。此字段要求启用请求身份验证，且等同于 `request.auth.principal` 属性。如果不设置，则允许所有请求主体。
`notRequestPrincipals`	`string[]`	可选。请求身份的反向匹配列表。
`namespaces`	`string[]`	可选。从对等证书衍生的命名空间。此字段要求启用 mTLS，且等同于 `source.namespace` 属性。如果不设置，则允许所有命名空间。
`notNamespaces`	`string[]`	可选。命名空间的反向匹配列表。
`ipBlocks`	`string[]`	可选。根据 IP 数据包的来源地址进行填充的 IP 段列表。支持单个 IP（例如 “1.2.3.4”）和 CIDR（例如 “1.2.3.0/24”）。这等同于 `source.ip` 属性。如果不设置，则允许所有 IP。
`notIpBlocks`	`string[]`	可选。IP 段的反向匹配列表。
`remoteIpBlocks`	`string[]`	可选。根据 X-Forwarded-For 标头或代理协议进行填充的 IP 段列表。要使用此字段，您必须在安装 Istio 或在 ingress 网关使用注解时在 meshConfig 下配置 gatewayTopology 的 numTrustedProxies 字段。 支持单个 IP（例如 “1.2.3.4”）和 CIDR（例如 “1.2.3.0/24”）。这等同于 `remote.ip` 属性。如果不设置，则允许所有 IP。
`notRemoteIpBlocks`	`string[]`	可选。远程 IP 段的反向匹配列表。

请求操作 Operation

Operation 指定请求的操作。对操作中的字段执行逻辑与操作。

Copyright © 2016 - 2023 DaoCloud

例如，以下操作将匹配：

- 主机后缀为 “.example.com”
- 方法为 “GET” 或 “HEAD”
- 补丁没有前缀 “/admin”

```
hosts: ["*.example.com"]
methods: ["GET", "HEAD"]
notPaths: ["/admin*"]
```

字段	类型	描述
`hosts`	`string[]`	可选。在 HTTP 请求中指定的主机列表。不区分大小写。如果不设置，则允许所有主机。仅适用于 HTTP。
`notHosts`	`string[]`	可选。在 HTTP 请求中指定的主机反向匹配列表。不区分大小写。
`ports`	`string[]`	可选。连接中指定的端口列表。如果不设置，则允许所有端口。
`notPorts`	`string[]`	可选。连接中所指定端口的反向匹配列表。
`methods`	`string[]`	可选。HTTP 请求中指定的方法列表。对于 gRPC 服务，这将始终是 “POST”。如果不设置，则允许所有方法。仅适用于 HTTP。
`notMethods`	`string[]`	可选。HTTP 请求中所指定方法的反向匹配列表。
`paths`	`string[]`	可选。HTTP 请求中指定的路径列表。对于 gRPC 服务，这将是 “/package.service/method” 格式的完全限定名称。如果不设置，则允许所有路径。仅适用于 HTTP。
`notPaths`	`string[]`	可选。路径的反向匹配列表。

Copyright © 2016 - 2023 DaoCloud

策略条件 Condition

Condition 指定其他必需的属性。

名称	描述	支持的协议	示例
`request.headers`	`HTTP` 请求头，需要用 `[]` 括起来	HTTP only	`key: request.headers[User-Agent]` `values: ["Mozilla/*"]`
`source.ip`	源 `IP` 地址，支持单个 `IP` 或 `CIDR`	HTTP and TCP	`key: source.ip` `values: ["10.1.2.3"]`
`remote.ip`	由 `X-Forwarded-For` 请求头或代理协议确定的原始客户端 IP 地址，支持单个 IP 或 CIDR	HTTP and TCP	`key: remote.ip` `values: ["10.1.2.3", "10.2.0.0/16"]`
`source.namespace`	源负载实例命名空间，需启用双向 TLS	HTTP and TCP	`key: source.namespace` `values: ["default"]`
`source.principal`	源负载的标识，需启用双向 TLS	HTTP and TCP	`key: source.principal` `values: ["cluster.local/ns/default/sa/productpage"]`
`request.auth.principal`	已认证过 `principal` 的请求	HTTP only	`key: request.auth.principal` `values: ["accounts.my-svc.com/104958560606"]`
`request.auth.audiences`	此身份验证信息的目标主体	HTTP only	`key: request.auth.audiences` `values: ["my-svc.com"]`
`request.auth.presenter`	证书的颁发者	HTTP only	`key: request.auth.presenter` `values: ["123456789012.my-svc.com"]`
`request.auth.claims`	`Claims` 来源于 `JWT`。需要用 `[]` 括起来	HTTP only	`key: request.auth.claims[iss]` `values: ["*@foo.com"]`
`destination.ip`	目标 `IP` 地址，支持单个 `IP` 或 `CIDR`	HTTP and TCP	`key: destination.ip` `values: ["10.1.2.3", "10.2.0.0/16"]`
`destination.port`	目标 `IP` 地址上的端口，必须在 `[0，65535]` 范围内	HTTP and TCP	`key: destination.port` `values: ["80", "443"]`
`connection.sni`	服务器名称指示，需启用双向 TLS	HTTP and TCP	`key: connection.sni` `values: ["www.example.com"]`
`experimental.envoy.filters.*`	用于过滤器的实验性元数据匹配，包装的值 `[]` 作为列表匹配	HTTP and TCP	`key: experimental.envoy.filters.network.mysql_proxy[db.table]` `values: ["[update]"]`

Copyright © 2016 - 2023 DaoCloud

!!! note

```
无法保证 `experimental.*` 密钥向后的兼容性，可以随时将它们删除，但须谨慎操作。
```


Copyright © 2016 - 2023 DaoCloud

对等身份认证

对等身份认证指的是在不对应用源码做侵入式修改的情况下，提供服务间的双向安全认证，同时密钥以及证书的创建、分发、轮转也都由系统自动完成，对用户透明，从而大大降低了安全配置管理的复杂度。

!!! note

```
启用对等身份认证后，相应的目标规则也需要开启 mLTS 模式，否则将无法正常访问。
```

一个对全网格生效的严格 mTLS 策略。生效后，网格内部服务间访问将必须启用 mLTS。

示例：

```
apiVersion: "security.istio.io/v1beta1"
kind: "PeerAuthentication"
metadata:
  name: "default"
  namespace: "istio-system"  #生效的名称空间
spec:
  mtls:
    mode: STRICT      #策略
```


Copyright © 2016 - 2023 DaoCloud

服务网格提供了两种创建方式：向导和 YAML。通过向导创建的具体操作步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏点击 安全治理 -> 对等身份认证，点击右上角的 创建 按钮。

2. 在 创建对等身份认证 界面中，先进行基本配置后点击 下一步 。

3. 按屏幕提示进行认证设置后，点击 确定 。

Copyright © 2016 - 2023 DaoCloud

4. 屏幕提示创建成功。

5. 在列表右侧，点击操作一列的 ⋮，可通过弹出菜单进行更多操作。

Copyright © 2016 - 2023 DaoCloud

!!! note

```
- 具体参数的配置，请参阅[安全治理参数配置](./params.md)。
- 更直观的操作演示，可参阅[视频教程](../../../videos/mspider.md)。
```


Copyright © 2016 - 2023 DaoCloud

请求身份认证

当一个外部用户对网格内部服务发起请求时，可以采用这种认证模式。在这种模式下，使用 JSON Web Token（JWT）实现请求加密。 每个请求身份认证需要配置一个授权策略。

所有标签为 `app: httpbin` 的工作负载需要使用 JWT 认证。示例如下：

```
apiVersion: security.istio.io/v1beta1
kind: RequestAuthentication
metadata:
  name: httpbin
  namespace: foo
spec:
  selector:
    matchLabels:
      app: httpbin
  jwtRules:
  - issuer: "issuer-foo"
    jwksUri: https://example.com/.well-known/jwks.json
```


Copyright © 2016 - 2023 DaoCloud

服务网格提供了两种创建方式：向导向导和 YAML。通过向导创建的具体操作步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏点击 安全治理 -> 请求身份认证，点击右上角的 创建 按钮。

2. 在 创建请求身份认证 界面中，先进行基本配置后点击 下一步 。

3. 按屏幕提示进行认证设置后，点击 确定 ，系统将验证所配置信息。

Copyright © 2016 - 2023 DaoCloud

4. 验证通过后，屏幕提示创建成功。

5. 在列表右侧，点击操作一列的 ⋮，可通过弹出菜单进行更多操作。

Copyright © 2016 - 2023 DaoCloud

!!! note

```
- 具体参数的配置，请参阅[安全治理参数配置](./params.md)。
- 更直观的操作演示，可参阅[视频教程](../../../videos/mspider.md)。
```


Copyright © 2016 - 2023 DaoCloud

5.6 Service list

服务管理

服务管理列出了当前网格下集群中已注入边车的所有服务，您可以基于命名空间筛选服务。

服务网格对各集群的服务做了聚合处理，同一个命名空间下的同名服务将聚合为一个服务，这样有利于对跨集群协同服务进行统一的流量治理。 服务列表中也标注了来自其他微服务注册中心的服务，如下图所示。

您可以点击某个服务名称，进入详情页查看各集群的服务地址、端口等具体信息，还可以在 地址信息 页签中修改通信协议。

Copyright © 2016 - 2023 DaoCloud

请特别关注服务列表中 诊断配置 这一列。当诊断信息为 异常 时，光标悬浮在 ⓘ 上将显示异常原因。异常状态会影响下一阶段的服务流量治理等网格相关能力。

Copyright © 2016 - 2023 DaoCloud

在服务列表右侧，点击 ⋮ 选择相应的菜单项，可以跳转至流量治理和安全治理。

关于如何创建和配置服务，请参阅创建服务。

Copyright © 2016 - 2023 DaoCloud

外部应用对网格内服务访问

本页说明外部应用如何通过配置来访问网格内的服务。

前置条件：

- 服务 `bookinfo.com` 运行于网格 `global-service` 的 `default` 命名空间下
- 网格提供 `ingressgateway` 网关实例

Copyright © 2016 - 2023 DaoCloud

配置目标：实现内部服务 `bookinfo.com` 对外暴露。

Copyright © 2016 - 2023 DaoCloud

1. 通过 URI 匹配方式，实现外部应用对服务 `bookinfo.com` 的指定页面访问路由。

2. 点击 `流量治理` -> `网关规则` -> `创建` 为 istio 网关创建网关规则，对外暴露服务及端口。

完成配置后的 YAML 示例如下：

```yaml
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway # 使用默认控制器
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
```


Copyright © 2016 - 2023 DaoCloud

```
    hosts:
    - bookinfo.com
```

3. 点击 `确定` 回到网关规则列表，可见创建成功提示。
4. 点击 `流量治理` -> `虚拟服务` -> `创建` 来创建路由规则，基于请求中的 URI 路由到指定页面。

完成配置后的 YAML 示例如下：

```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - bookinfo.com
  gateways:
  - bookinfo-gateway
  http:
  - match:
      - uri:
          exact: /productpage
      - uri:
          exact: /login
      - uri:
          exact: /logout
      - uri:
          prefix: /api/v1/products
    route:
    - destination:
        host: productpage
        port:
          number: 9080
```

5. 点击 `确定` 回到虚拟服务列表，可以看到创建成功的提示。

Copyright © 2016 - 2023 DaoCloud

!!! info

```
更直观的操作演示，可参阅[视频教程](../../../videos/mspider.md)。
```


Copyright © 2016 - 2023 DaoCloud

服务条目

服务条目可以将外部服务、Web API 或虚拟机添加到服务网格的内部服务注册表中。 以一个外部服务为例，添加服务条目后，Envoy 代理可以将流量发送到该外部服务，服务网格也可以通过虚拟服务和目标规则对该外部服务进行流量治理，就好像该服务条目是网格中的正常服务一样。

服务网格提供了两种创建方式：YAML 创建和向导创建。

Copyright © 2016 - 2023 DaoCloud

图形向导创建

这种创建方式比较简单直观。

Copyright © 2016 - 2023 DaoCloud

1. 进入所选网格后，在左侧导航栏点击 流量治理 -> 服务条目，点击右上角的 创建 按钮。

2. 在 创建服务条目 页面中，配置各项参数后点击 确定 。具体参数的含义，请参见参数说明

3. 返回服务条目列表，屏幕提示创建成功。

Copyright © 2016 - 2023 DaoCloud

4. 在列表右侧，点击操作一列的 ⋮，可通过弹出菜单进行更多操作。

YAML 创建

1. 进入所选网格后，在左侧导航栏点击 流量治理 -> 服务条目，点击右上角的 YAML 创建 按钮。

2. 选择命名空间，选择模板，修改各个字段参数，或者直接导入现有的 YAML 文件，参数确认无误后点击 确定 。

Copyright © 2016 - 2023 DaoCloud

3. 返回服务条目列表，屏幕提示创建成功。

以下是一个标准的服务条目 YAML 示例：

```
apiVersion: networking.istio.io/v1beta1
kind: ServiceEntry
metadata:
  name: entry01
  namespace: istio-system
spec:
  addresses:
  - 127.10.18.65
  endpoints:
```


Copyright © 2016 - 2023 DaoCloud

```
  - address: 127.10.18.78
    ports:
      test: 9980
  exportTo:
  - istio-system
  hosts:
  - test.service
  location: MESH_INTERNAL
  ports:
  - name: test
    number: 9980
    protocol: HTTP
  workloadSelector: {}
status: {}
```

概念介绍

上述 YAML 文件和创建向导中各项参数的含义简单说明如下。

- Hosts

  服务名。可用于流量治理策略（虚拟服务、目标规则等）中的 hosts 字段匹配。

- 在 HTTP 流量中，服务名将是 HTTP 主机或 Authority 头
- 在具有 SNI 名称的 HTTP、TLS 流量中，服务名将是 SNI 名称
- Addresses

  服务地址。与服务关联的虚拟 IP 地址，也可以是 CIDR 前缀。

- 如果设置了 Addresses 字段，将对请求 HTTP 流量的 服务名和 IP/CIDR 做匹配，确认是否属于该服务。
- 如果 Addresses 字段为空，则仅根据目标端口识别流量。此时网格中的任何其他服务都不能共享使用该端口，边车将会转发所有该端口的传入流量至指定目标 IP/主机。
- Ports

  服务端口。与服务相关联的端口。如果端点是 Unix 域套接字地址，则必须有一个端口。

- Location

  服务地址。需要输入一个有效的 IP 地址。用于标明服务是否处于网格内部。

- Resolution

  解析方式。提供了对服务地址的多种解析方式：

- NONE：直接将流量转发至服务地址或服务端点的地址（如果存在）。
- STATIC：使用服务端点中的静态地址。
- DNS：尝试通过异步查询环境 DNS 来解析 IP 地址。
- 如果未设置服务端点，并且未使用通配符，将解析服务名字段中指定的 DNS 地址。
- 如果指定了服务端点，将解析服务端点中指定的DNS地址。DNS 解析不能与 Unix 域套接字服务端点一起使用。
- DNS_ROUND_ROBIN：尝试通过异步查询环境DNS来解析 IP 地址。与 DNS 方式不同的是，DNS_ROUND_ROBIN 方式仅使用连接建立后返回的第一个 IP 地址，而不依赖于 DNS 解析的完整结果。
- Endpoint

  服务端点。与服务相关的端点信息，包含 IP 地址、端口和服务端口名称等。

- WorkloadSelector

  工作负载选择标签。一个键值对，用于选择网格内部服务的工作负载，该项和服务端点为二选一。

Copyright © 2016 - 2023 DaoCloud

5.7 Service mesh

服务网格

在服务网格主页面，用户可以浏览网格基本信息、集群实例状态以及网格运行状态，还可以对网格执行创建、配置、删除等操作。

网格列表展示了网格的类型、Istio 版本、集群和 Pod 状态等基本信息。 服务网格为用户提供了三种网格类型：托管网格、专有网格、外接网格，用户可以基于实际集群环境选择创建。

- 托管网格：用于多集群管理，网格资源会存储在单独的虚拟集群中，高度的独立性保证用户可以自由接入、移除多个集群，并对多个集群的命名空间、服务做聚合治理；
- 专有网格：用于单集群的网格管理，网格控制面安装在用户的工作集群中，不可外接其他集群；
- 外接网格：接入用户自主安装的原生 Istio，通过服务网格管理平台统一注入边车、下发管理策略等。

Copyright © 2016 - 2023 DaoCloud

网格列表卡片的各项含义如下：

界面文字	含义及变化说明
网格名称	用户创建的网格名称，由字母、数字、中划线（-）组成。
运行状态	网格运行状态，共有以下几种状态： - 运行中，表示网格已纳管集群，并且配置完整 - 未就绪，表示网格未纳管任何集群 - 异常，表示网格无法正常工作；网格创建、删除、设置失败。 - 创建中，表示正在创建网格 - 删除中，表示正在删除网格
网格类型	托管网格/专有网格/外接网格
Istio 版本	当前网格所使用 Istio 版本号
控制面集群	用于运行服务网格控制面的管理集群，该项在网格创建后不可修改
创建时间	网格创建时间
集群统计	网格下状态正常的集群与所有集群数之比
Pod 实例数	网格下各集群中，已注入边车的正常状态 Pod 与所有 Pod 之比
集群列表	光标移动至 `集群统计` 的数字区域，将出现浮动提示，显示当前网格下管理的所有集群名称及其状态

Copyright © 2016 - 2023 DaoCloud

创建托管/专有网格

DCE 5.0 服务网格支持创建托管网格、专有网格、外接网格。创建托管网格/专有网格的步骤如下：

Copyright © 2016 - 2023 DaoCloud

!!! info - 托管网格指的是将网格核心控制面组件从工作集群的概念中分离，可以部署到一个独立的控制面集群中，减轻用户运维负担和资源消耗，并且用户可以对位于同一网格中的多集群服务进行统一治理。 - 专有与托管网格的最大区别是：专有网格是单集群模式。 - 有关如何接入外接网格的信息，请参见创建外接网格。

Copyright © 2016 - 2023 DaoCloud

1. 在服务网格列表页面的右上角，点击 `创建网格` 。

2. 选择 `托管网格` 或 `专有网格` ，填写网格配置信息。

- 网格名称：以小写字母开头，由小写字母、数字、中划线(-)组成，且不能以中划线(-)结尾
- 别名：用于提高易用性和网格识别度，可输入字母、数字、中文及其他符号，60 个字符以内
- Istio 版本：当前系统可以支持的 Istio 版本号，选择一个作为当前创建的网格的 Istio 版本，如果是托管网格，那么所有被纳管的集群的 Istio 都将使用此版本。
- 控制面集群：用于运行网格管理面的集群，列表包含当前网格平台可以访问且状态正常的集群。该项附带一个刷新图标和一个 `创建集群` 按钮。点击 `创建集群` 将跳转至 `容器管理平台` 创建新集群，创建完成后返回本页面，点击刷新图标更新列表。
- 镜像仓库：输入包含数据面组件镜像的镜像仓库地址，如果集群可以访问公网，那么可以填写官方镜像地址：`release.daocloud.io/mspider` ，私有环境请自行上传镜像后填写实际地址。

Copyright © 2016 - 2023 DaoCloud

3. 点击 `确定` 按钮，完成网格创建。系统自动返回网格列表，可以查看和管理新增的网格。

!!! info

```
创建托管网格后，还未接入任何托管的集群，此时网格处于`未就绪`状态。用户可以[添加集群](../cluster-management/README.md)，等待集群接入完成后，选择需要服务治理的集群接入。
```

可选设置

正确填写基本配置后就可以成功创建基本的托管网格。如需使用高级功能，可以点击 `可选设置` 填写可选配置。

- 网格规模：包含四个规模选项，提供了与所选规模相关的资源建议信息
- 边车注入策略：可以启用/禁用网格层面的边车自动注入策略，启用后所有接入集群将自动注入边车
- 边车资源设置：对网格下所有集群的注入边车进行资源限制，可作为网格下的默认边车资源值。各层级的边车资源限制规则执行优先级为：工作负载级资源限制 -> 集群级资源限制 - > 网格资源限制

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

删除网格

当用户不再使用网格治理服务时可以使用删除操作，删除网格需用户完成一系列前置操作，才可以激活对话框的 确定 按钮。

Copyright © 2016 - 2023 DaoCloud

!!! danger

```
执行删除操作后，网格不可恢复，如需再次纳管，需重新创建网格。
```


Copyright © 2016 - 2023 DaoCloud

1. 在网格列表的右侧，点击 `...` 按钮，在弹出菜单中选择 `删除`。

Copyright © 2016 - 2023 DaoCloud

2. 按照网格类型，系统将自动检测是否满足删除条件。

Copyright © 2016 - 2023 DaoCloud

- 删除外接网格。

- 删除专有网格。需要按提示移除边车、清除网关、移除集群。

- 删除托管网格。需要按提示移除边车、清除网关、移除集群。

Copyright © 2016 - 2023 DaoCloud

3. 清除所有依赖项之后，输入屏幕提示的确认码，删除 按钮被激活，点击 删除 按钮删除网格。

Copyright © 2016 - 2023 DaoCloud

创建外接网格

1. 在服务网格列表页面的右上角，点击 `创建网格` 。

Copyright © 2016 - 2023 DaoCloud

2. 选择 外接网格 ，填写网格配置信息。

- 网格名称：以小写字母开头，由小写字母、数字、中划线(-)组成，且不能以中划线(-)结尾
- 控制面集群：用于运行网格管理面的集群，列表包含当前网格平台可以访问且状态正常的集群。 该项附带一个刷新图标和一个 创建集群 按钮。点击 创建集群 将跳转至 容器管理平台 创建新集群，创建完成后返回本页面，点击刷新图标更新列表。
- Istio 根命名空间：网格所在的 Istio 根命名空间。
- 镜像仓库：输入镜像仓库的 URL 地址。

3. 点击 确定 按钮，完成网格创建。系统自动返回网格列表，可以查看和管理新增的网格。

Copyright © 2016 - 2023 DaoCloud

5.8 Sidecar management

全局边车管理

用户可以对网格下所有集群做统一注入设置和边车资源限制设置。

全局边车自动注入

在 边车管理 -> 全局边车管理 页面，打开 全局边车自动注入 开关，该操作将开启网格下所有集群的边车自动注入功能，新接入的集群下 Pod 也将自动执行注入。

!!! note

```
全局注入功能将启用标签 `values.sidecarInjectorWebhook.enableNamespacesByDefault`，该标签仅对未做命名空间和工作负载注入设置的 Pod 实例生效。
```

全局边车资源限制

在集群列表中选择集群，点击 边车资源限制 ，在弹出窗口中为所选集群设置每一个 Pod 实例的请求资源和限制资源，其中请求资源为一定可以获取的资源，限制资源为可用资源上限。输入为 0 表示该项不做限制。

Copyright © 2016 - 2023 DaoCloud

!!! note

在`工作负载边车管理`中为工作负载设置的`边车资源限制`优先级高于`全局边车资源限制`。

Copyright © 2016 - 2023 DaoCloud

命名空间边车管理

用户可以从命名空间的层面启用或禁用边车注入等功能。

!!! note

选中某个边车后，右上角才会显示`注入禁用`和`注入启用`的按钮。

查看边车注入信息

在左侧导航栏中，点击 `边车管理` -> `命名空间边车管理`，点选某个集群（图示为 `centos-50`），可以查看该集群下所有命名空间的边车状态。

当命名空间较多时，可以按命名空间的名称进行排序，并通过搜索功能进行查找。

Copyright © 2016 - 2023 DaoCloud

启用边车注入

用户可以选择一个或多个命名空间，启用边车自动注入功能，具体步骤如下：

1. 选择一个未启用边车注入的命名空间，点击 `注入启用` 按钮；

2. 在弹出的对话框中，确认选择的命名空间数量是否正确，确认无误后，点击 `确定` 。

3. 自动返回命名空间的边车列表，可以看到刚刚所选命名空间的 `边车自动注入` 状态已变更为 `启用` 。在用户完成工作负载的重启后，将完成边车注入，相关注入进度可查看 `边车注入量` 一列。

Copyright © 2016 - 2023 DaoCloud

禁用边车注入

用户可以选择一个或多个命名空间，禁用边车自动注入功能，具体步骤如下：

1. 选择一个已启用边车注入的命名空间，点击 注入禁用 按钮；

2. 在弹出的对话框中，确认选择的命名空间数量是否正确，确认无误后，点击 确定 。

3. 自动返回命名空间的边车列表，可以看到刚刚所选命名空间的 边车自动注入 状态已变更为 禁用 。在用户完成工作负载的重启后，将完成边车禁用，相关卸载进度可查看 边车注入量 一列。

Copyright © 2016 - 2023 DaoCloud

边车流量透传

流量透传（traffic passthrough）指的是工作负载的全部或部分上游、下游流量不经边车转发，直接发送至工作负载本身。

DCE 5.0 服务网格实现了对工作负载出站/入站流量的边车透传可控，可针对特定端口、IP 段实现拦截设置。

- 功能设置对象：工作负载
- 设置参数：端口、IP 段
- 流向：入站、出站

流量透传相关字段：

```
traffic.sidecar.istio.io/excludeInboundPorts
traffic.sidecar.istio.io/excludeOutboundPorts
traffic.sidecar.istio.io/excludeOutboundIPRanges
```


Copyright © 2016 - 2023 DaoCloud

启用流量透传

本节说明如何在 DCE 图形界面上启用/禁用流量透传。

Copyright © 2016 - 2023 DaoCloud

1. 进入某个网格，点击 边车管理 -> 工作负载边车管理 。

2. 点击某个负载右侧的 ⋮ ，在弹出菜单中选择 流量透传设置 。

3. 设置流量透传的参数后，勾选 立即重启工作负载 ，点击 确认变更 。

Copyright © 2016 - 2023 DaoCloud

- 入站：仅支持端口，即从外部访问网格内负载的端口
- 出站：可设置目标的端口或 IP 段

4. 如果设置无误，右上角将出现 流量透传设置成功 的提示消息。您还可以查验流量透传效果。

5. 如果流量透传已启用，上述第 3 步的 流量透传设置 弹窗将显示设置的参数，可点击右侧的 x，勾选 立即重启工作负载，点击 确认变更 来禁用流量透传。

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

查验流量透传效果

在真实的网格集群中，查验流量透传前后的效果。

1. 准备工作

- 准备一个网格集群，例如 10.64.30.130
- 在命名空间中，配置工作负载 `helloworld`，并注入边车
- 启用流量透传，然后比对该负载的流量路由变化

2. 通过控制台或 ssh 登录到网格。

```bash
ssh root@10.64.30.130
```

3. 查看命名空间中的 svc，获取 clusterIP 和 Port：

```bash
$ kubectl get svc -n default
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
helloworld	ClusterIP	10.211.201.221	<none>	5000/TCP	39d
kubernetes	ClusterIP	10.211.0.1	<none>	443/TCP	62d
test-cv	NodePort	10.211.72.8	<none>	2222:30186/TCP	62d

4. 执行 curl 命令查看 helloworld 的流量路由：

=== "启用流量透传前"

```
```bash
$ curl -sSI 10.211.201.221:5000/hello
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 65
server: istio-envoy # (1)
date: Tue, 07 Feb 2023 03:08:33 GMT
x-envoy-upstream-service-time: 100
x-envoy-decorator-operation: helloworld.default.svc.cluster.local:5000/*
```

1. 流量经过 istio-envoy，即边车的代理
```

=== "启用流量透传后"

```
```bash
$ curl -sSI 10.211.201.221:5000/hello
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 65
Server: Werkzeug/0.12.2 Python/2.7.13 # (1)
Date: Tue, 07 Feb 2023 03:08:10 GMT
```

1. 流量直接进入工作负载本身
```

Copyright © 2016 - 2023 DaoCloud

工作负载边车管理

用户可以对工作负载执行边车注入的查看、启用、禁用等操作，还可以为工作负载设置资源限制。

查看边车注入信息

在左侧导航栏中，点击 `边车管理` -> `工作负载边车管理`，选择一个集群后，可以查看该集群下所有工作负载及其边车注入状态、所属命名空间、资源限制等信息。

各列的含义如下：

工作负载	所选集群下所有工作负载（不包含系统占用不可边车注入的工作负载）
状态	三个状态： 已注入 - 已完成边车自动注入 未注入 - 禁用边车自动注入 待重启 - 所述命名空间的 istio-injection 已发生变化，但相关 Pod 还未重启
命名空间	该工作负载所属命名空间；
服务	该工作负载相关服务，该项内容可能有多个，可采用扩缩列表，用户点击后浮动窗显示所有内容；
已注入Pod	该工作负载下 Pod 的注入情况； 格式：已注入 Pod/所有可注入 Pod 如果工作负载的注入状态为 `已注入`，但有部分 Pod 没有注入，例如 3/5，该项将高亮显示，提醒用户有注入失败的 Pod，需及时处理
CPU申请值/限制值	包含 `请求` 资源和 `限制` 资源两个数值，如果用户未进行资源设置，该项目为 `未设置`。 格式：请求 / 限制
内存申请值/限制值	包含 `请求` 资源和 `限制` 资源两个数值，如果用户未进行资源设置，该项目为 `未设置`。 格式：请求 / 限制
操作	包含两个操作链接： 边车资源限制：对当前工作负载做资源设置，该操作对所有状态的工作负载有效； 查看边车状态：仅在状态为 `注入` 的工作负载中显示，调用 `边车运行状态` 对话框；

当工作负载较多时，可以对工作负载名称进行排序，并通过搜索功能查找当前集群下的目标工作负载。

Copyright © 2016 - 2023 DaoCloud

查看边车运行状态

在工作负载边车列表右侧，点击 `操作` 一列的图标 ⋮，在弹出的菜单中选择 `查看边车状态`。可以查看边车当前运行状态及资源限制值。

Copyright © 2016 - 2023 DaoCloud

启用边车注入

选择一个或多个工作负载后，可以启用边车自动注入功能。该操作会重启相关 Pod，因此请在执行该操作前确认 Pod 可以重启。具体操作步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 选择一个或多个未启用边车注入的工作负载，点击右侧的 `注入启用` 按钮。

2. 在弹出对话框中，确认选择的工作负载数量是否正确，确认无误后，勾选 `立即重启` ，点击 `确定` 。

3. 返回工作负载的边车列表，可以看到所选工作负载的 `状态` 已变更为 `已注入` 。在完成工作负载的重启后，将完成边车注入，相关注入进度可查看 `已注入 POD` 这一列。

Copyright © 2016 - 2023 DaoCloud

!!! note

如果工作负载所属命名空间已执行`注入启用/禁用`操作但未重启工作负载，则该工作负载将无法执行新的边车相关操作。
需先完成重启，才能执行新的边车操作。

Copyright © 2016 - 2023 DaoCloud

禁用边车注入

选择一个或多个工作负载后，可以禁用边车自动注入功能。该操作会重启相关 Pod，因此请在执行该操作前确认 Pod 可以重启。具体操作步骤如下：

1. 选择一个或多个已启用边车注入的工作负载，点击右侧的 `注入禁用` 按钮。

2. 在弹出对话框中，确认选择的工作负载数量是否正确，确认无误后，勾选 `立即重启`，点击 `确定`。

3. 返回工作负载的边车列表，可以看到所选工作负载的 `状态` 已变更为 `未注入`。相关卸载进度可查看 `已注入 POD` 这一列。

Copyright © 2016 - 2023 DaoCloud

边车资源限制

为了防止工作负载的资源被过量使用，可以为工作负载设置资源使用限制。该操作会重启相关 Pod，因此请在执行该操作前确认 Pod 可以重启。具体操作步骤如下：

1. 选择一个（或多个）已启用边车注入的工作负载，点击 边车资源限制 按钮。

2. 弹出对话框中，分别设置 CPU/内存的请求值与限制值。选中 立即重启 ，点击 确定 。

3. 在工作负载边车管理列表中，可以看到指定工作负载的 CPU 申请值/限制值 和 内存申请值/限制值 内数据已更新。

Copyright © 2016 - 2023 DaoCloud

5.9 Traffic governance

流量治理

流量治理为用户提供了三种资源配置，虚拟服务、目标规则、网关规则。 通过配置相应规则可以实现路由、重定向、熔断、分流等多项流量治理功能。用户可以通过向导或YAML形式创建、编辑治理策略。

- 虚拟服务主要用于对请求流量的路由定制规则，并可以对数据流做出分流、重定向、超时返回等处理
- 目标规则则更关注流量本身的治理，为请求流量提供更强大的负载均衡、连接存活探寻、熔断等功能
- 网关规则为 Istio 网关提供服务在网关的暴露方式

在实际运用中，需要三类策略配合使用：

- 由虚拟服务来定义路由规则和描述满足条件的请求去哪里
- 目标规则则是定义子集和策略，描述到达目标的请求该怎么处理
- 如果有外部服务通信需求，则需要配置网关规则中的端口映射等细节来实现需求

在服务网格中，为用户提供了向导和 YAML 两种创建/编辑形式，用户可以根据个人习惯自由选择。

- 向导创建方式为用户提供简单直观的互动方式，一定程度上降低了用户的学习成本
- YAML 创建形式更适合资深用户，用户可以直接编写YAML文件创建治理策略，并且创建窗口中也为用户提供了较为常用的治理策略模板，提高用户编写速度

Copyright © 2016 - 2023 DaoCloud

目标规则

目标规则（DestinationRule）同样是服务治理中重要的组成部分，目标规则通过端口、服务版本等方式对请求流量进行划分，并对各请求分流量订制 envoy 流量策略，应用到流量的策略不仅有负载均衡，还有最小连接数、熔断等。

概念介绍

几个重要字段如下：

- Host

  使用 Kubernetes Service 的短名称。含义同 VirtualService 中 `destination` 的 `host` 字段一致。服务一定要存在于对应的服务注册中心中，否则会被忽略。

- LoadBalancer

  默认情况下，Istio 使用轮询的负载均衡策略，实例池中的每个实例依次获取请求。 Istio 同时支持如下的负载均衡模型，可以在 DestinationRule 中为流向某个特定服务或服务子集的流量指定这些模型。

- 随机：请求以随机的方式转到池中的实例。
- 权重：请求根据指定的百分比转到实例。
- 最少请求：请求被转到最少被访问的实例。
- Subsets

  `subsets` 是服务端点的集合，可以用于 A/B 测试或者分版本路由等场景。 可以将一个服务的流量切分成 N 份供客户端分场景使用。 `name` 字段定义后主要供 VirtualService 里 `destination` 使用。 每个子集都是在 `host` 对应服务的基础上基于一个或多个 `labels` 定义的，在 Kubernetes 中它是附加到像 Pod 这种对象上的键/值对。 这些标签应用于 Kubernetes 服务的 Deployment 并作为元数据信息（Metadata）来识别不同的版本。

- OutlierDetection

  熔断是减少服务异常和降低服务延迟的一种设计模式，主要是无感的处理服务异常并保证不会发生级联甚至雪崩。 如果在一定时间内服务累计发生错误的次数超过了预先定义的阈值，就会将该错误的服务从负载均衡池中移除，并持续关注服务的健康状态，当服务回复正常后，又会将服务再移回到负载均衡池。

目标规则列表介绍

目标规则列表展示了网格下的目标规则 CRD 信息，并提供了目标规则生命期管理能力。 用户可以基于规则名称、规则标签做 CRD 筛选，规则标签如下：

- 服务版本
- 负载均衡
- 位置感知负载均衡
- HTTP 连接池
- TCP 连接池
- 客户端 TLS
- 熔断

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

操作步骤

服务网格提供了两种创建方式：表单创建和 YAML 创建。通过表单创建的具体操作步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏点击 流量治理 -> 目标规则，点击右上角的 创建 按钮。

2. 在 创建目标 界面中，先进行基本配置后点击 下一步 。

3. 按屏幕提示选择策略类型，并配置对应的治理策略后，点击 确定 。

Copyright © 2016 - 2023 DaoCloud

4. 返回目标规则列表，屏幕提示创建成功。

5. 在列表右侧，点击操作一列的 ⋮，可通过弹出菜单进行更多操作。

Copyright © 2016 - 2023 DaoCloud

YAML 创建方式与虚拟服务相似，您可以直接借助内置模板创建 YAML 文件，如下图所示。

Copyright © 2016 - 2023 DaoCloud

网关规则

网关规则（Gateway）用于将服务暴露于网格之外，相较于 Kubernetes 的 ingress 对象，istio-gateway 增加了更多的功能：

- L4-L6 负载均衡
- 对外 mTLS
- SNI 的支持
- 其他 Istio 中已经实现的内部网络功能：Fault Injection、Traffic Shifting、Circuit Breaking、Mirroring

概念介绍

对于 L7 的支持，网关规则通过与虚拟服务配合实现。几个重要主字段如下：

- Selector

  选择用于南北流量的istio网关，可以使用多个，也可以与其他规则共用一个。
- Servers

  对外暴露服务的相关信息，包括hosts（服务名称）、监听端口、协议类型等。
- TLS

  提供对外的mTLS协议配置，用户可以启用三种TLS模式，并可以自定义CA证书等操作。

示例：

```
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - istio-grafana.frognew.com
```


Copyright © 2016 - 2023 DaoCloud

操作步骤

服务网格提供了两种创建方式：向导和 YAML。通过向导创建的具体操作步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏点击 流量治理 -> 网关规则，点击右上角的 创建 按钮。

2. 在 创建网关规则 界面中，配置基本信息，并根据需要添加服务端后点击 确定 。

3. 返回网关规则列表，屏幕提示创建成功。

Copyright © 2016 - 2023 DaoCloud

4. 在列表右侧，点击操作一列的 ⋮，可通过弹出菜单进行更多操作。

Copyright © 2016 - 2023 DaoCloud

虚拟服务

在虚拟服务（VirtualService）中，可以通过多种匹配方式（端口、host、header 等）实现对不同的地域、用户请求做路由转发，分发至特定的服务版本中，并按权重比划分负载。

虚拟服务提供了 HTTP、TCP、TLS 三种协议的路由支持。

概念介绍

- Hosts

  流量的目标主机。可以来自服务注册信息，服务条目（service entry），或用户自定义的服务域名。可以是带有通配符前缀的 DNS 名称，也可以是 IP 地址。 根据所在平台情况，还可能使用短名称来代替 FQDN。这种场景下，短名称到 FQDN 的具体转换过程是要靠下层平台完成的。

  一个主机名只能在一个 VirtualService 中定义。同一个 VirtualService 中可以用于控制多个 HTTP 和 TCP 端口的流量属性。

  需要注意的是，当使用服务的短名称时（例如使用 reviews，而不是 `reviews.default.svc.cluster.local` ），服务网格会根据规则所在的命名空间来处理这一名称，而非服务所在的命名空间。 假设 `default` 命名空间的一条规则中包含了一个 reviews 的 host 引用，就会被视为 `reviews.default.svc.cluster.local`，而不会考虑 reviews 服务所在的命名空间。

  为了避免可能的错误配置，建议使用 FQDN 来进行服务引用。 hosts 字段对 HTTP 和 TCP 服务都是有效的。 网格中的服务也就是在服务注册表中注册的服务，必须使用他们的注册名进行引用；只有 Gateway 定义的服务才可以使用 IP 地址。

  示例：

  ```yaml
  spec:
    hosts:
    - ratings.prod.svc.cluster.local
  ```

- Gateways

  通过将 VirtualService 绑定到同一 Host 的网关规则，可向网格外部暴露这些 Host。

  网格使用默认保留字 mesh 指代网格中的所有 Sidecar。 当这一字段被省略时，就会使用默认值（mesh），也就是针对网格中的所有 Sidecar 生效。 如果为 gateways 字段设置了网关规则（可以有多个），就只会应用到声明的网关规则中。 如果想同时对网关规则和所有服务生效，需要显式的将 mesh 加入 gateways 列表。

  示例：

  ```yaml
  gateways:
  - bookinfo-gateway
  - mesh
  ```

- Http

  有序规则列表。该字段包含了针对 http 协议的所有路由配置功能，对名称前缀为 `http-`、`http2-`、`grpc-` 的服务端口，或者协议为 HTTP、HTTP2、GRPC 以及终结的 TLS， 另外还有使用 HTTP、HTTP2 以及 GRPC 协议的 ServiceEntry 都是有效的。 流量会使用匹配到的第一条规则。

Copyright © 2016 - 2023 DaoCloud

http 下主要字段说明：

- Match

  匹配要激活的规则要满足的条件。单个匹配块内的所有条件都具有 AND 语义，而匹配块列表具有 OR 语义。 如果任何一个匹配块成功，则匹配该规则。

- Route

  http 规则可以重定向或转发（默认）流量。

- Redirect

  http 规则可以重定向或转发（默认）流量。 如果在规则中指定了流量通过选项，则将忽略路由/重定向。 重定向原语可用于将 HTTP 301 重定向发送到其他 URI 或 Authority。

- Rewrite

  重写 HTTP URI 和 Authority header，重写不能与重定向原语一起使用。

- Fault

  故障注入策略，适用于客户端的 HTTP 通信。 如果在客户端启用了故障注入策略，则不会启用超时或重试。

- Mirror/MirrorPercent

  将 HTTP 流量镜像到另一个目标，并可以设置镜像比例。

- Tcp

  一个针对透传 TCP 流量的有序路由列表。 TCP 路由对所有 HTTP 和 TLS 之外的端口生效。 进入流量会使用匹配到的第一条规则。

- Tls

  一个有序列表，对应的是透传 TLS 和 HTTPS 流量。路由过程通常利用 ClientHello 消息中的 SNI 来完成。 TLS 路由通常应用在 https-、tls- 前缀的平台服务端口，或者经 Gateway 透传的 HTTPS、TLS 协议端口，以及使用 HTTPS 或者 TLS 协议的 ServiceEntry 端口上。 注意：没有关联 VirtualService 的 https- 或者 tls- 端口流量会被视为透传 TCP 流量。

  Tcp 协议和 tls 的子字段相对简单，仅包含 match 和 route 两部分，并且与 http 相似，不再累述。

虚拟服务列表介绍

虚拟服务列表展示了网格下的虚拟服务CRD信息，用户可以按命名空间查看，也可以基于作用范围、规则标签做CRD筛选，规则标签如下：

- HTTP 路由
- TCP 路由
- TLS 路由
- 重写
- 重定向
- 重试
- 超时
- 故障注入
- 代理服务
- 流量镜像

虚拟服务提供了两种创建方式，表单创建和 YAML 创建，下文将介绍虚拟服务的相关操作。

Copyright © 2016 - 2023 DaoCloud

Copyright © 2016 - 2023 DaoCloud

操作步骤

通过表单创建的具体操作步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 在左侧导航栏点击 流量治理 -> 虚拟服务，点击右上角的 创建 按钮。

2. 在 创建虚拟服务 界面中，先进行基本配置后点击 下一步 。

3. 按屏幕提示选择进行路由配置后，点击 确定 。

Copyright © 2016 - 2023 DaoCloud

4. 返回虚拟服务列表，屏幕提示创建成功。

5. 在列表右侧，点击操作一列的 ⋮，可通过弹出菜单进行更多操作。

Copyright © 2016 - 2023 DaoCloud

通过 YAML 创建的操作相对简单，用户可以点击 `YAML 创建`，进入创建页面直接编写 YAML，也可以使用页面内提供的模板简化编辑操作，编辑窗口会提供基本语法检查功能，帮助用户完成编写工作，如下图所示。

Copyright © 2016 - 2023 DaoCloud

虚拟服务参数配置

本页介绍创建和编辑 虚拟服务 时的参数配置。

基本配置

UI 元素	YAML 字段	描述
名称	metadata.name	必填。虚拟服务名称。 格式要求：小写字母、数字和中划线（-）组成，必须以小写字母开头，以小写字母或数字结尾，最长 63 个字符。
命名空间	metadata.namespace	必填。虚拟服务所属命名空间。 同一个命名空间内，请求身份认证不可重名。
应用范围	spec.gateways	必填。虚拟服务应用的范围，包含两类： - 指定网关规则（可添加多个），可用于对外暴露网格内部服务； - 对所有边车（-mesh）生效。
所属服务	spec.hosts	必填。应用虚拟服务的服务对象，可包含三类： - 来自 Kubernetes 注册中心的注册服务； - 来自服务条目的注册服务； - 服务域名。

路由配置

可添加多条路由规则，执行顺序由上至下，排名在前的路由规则优先生效。

HTTP 路由

多条 HTTP 路由规则之间可以拖动排序，可收起任一条路由，仅显示路由名称。

基本信息

UI 元素	YAML 字段	描述
路由名称	spec.http.-name	必填。http 路由名称。 格式要求：小写字母、数字和中划线（-）组成，必须以小写字母开头，以小写字母或数字结尾，最长 63 个字符。

路由匹配规则

必填。YAML 字段为 `spec.http.-name.match`。

通过 URI 路径、端口等方式对请求做匹配，可添加多条规则，执行顺序由上至下，优先使用排名在前的匹配规则。

UI 元素	YAML 字段	描述
匹配URI	spec.http.-name.match.uri	可选。对请求的URI路径做匹配，有三种匹配方式： - 精确（exact）：字段完全匹配 - 前缀（prefix）：对字段前缀的匹配 - 正则（regex）：基于RE2样式正则表达式的匹配
匹配端口	spec.http.-name.match.port	可选。对请求的端口做匹配。
匹配header	spec.http.-name.match.header	必填。对请求的 http header 做匹配，同样支持三种匹配方式： - 精确（exact）：字段完全匹配 - 前缀（prefix）：对字段前缀的匹配 - 正则（regex）：基于RE2样式正则表达式的匹配

路由目标/重定向规则

`路由目标` 功能和 `重定向` 功能为互斥功能，一条 `HTTP 路由规则` 内仅可以二选一。

Copyright © 2016 - 2023 DaoCloud

!!! note

当用户开启`代理`开关后，该项及相关内容置灰。

路由规则

UI 元素	YAML 字段	描述
路由目标	spec.http.-name.route.-destination	可选。已匹配请求的路由目标，可添加多条，优先执行排名在前的路由目标。
服务名称	spec.http.-name.route.-destination.host	必填。路由目标服务的名称或 IP。
版本服务	spec.http.-name.route.-destination.subset	可选。列表内容来自所选服务的可用 `目标规则`。
权重	spec.http.-name.route.weight	可选。本条 `路由目标` 内各条所占流量的分配权重。所有 `路由目标` 的权重总和应为 100。
端口	spec.http.-name.route.-destination.port.number	可选。路由目标服务的端口。

重定向

UI 元素	YAML 字段	描述
重定向	spec.http.-name.redirect	可选。重定向用于将请求转发至其他路径。
重定向路径	spec.http.-name.redirect.uri	必填。新的访问地址路径（URI）。
authority	spec.http.-name.redirect.authority	可选。URI 路径中认证信息部分，通常 `//` 表示开始，`/` 表示结束。
端口	spec.http.-name.redirect.port.number	可选。重定向服务的端口号。
响应码	spec.http.-name.redirect.redirectCode	可选。指定响应码，当返回指定错误码时，执行重定向操作，默认不填为 301。

可选设置

另外还提供了 6 个可选设置，您可以根据实际需求启用或禁用。

重写

UI 元素	YAML 字段	描述
重写	spec.http.-name.rewrite	可选。默认关闭 l 可以重现完整路径，也可以仅重写 http 前缀。
重写路径	spec.http.-name.rewrite.uri	必填。新的访问地址路径（URI）。
authority	spec.http.-name.redirect.authority	可选。URI 路径中认证信息部分，通常 `//` 表示开始，`/` 表示结束。

超时

UI 元素	YAML 字段	描述
超时	spec.http.-name.timeout	可选。默认关闭。超时功能用于定义向目标服务发起请求的等待时长。
超时时长	spec.http.-name.timeout	必填。可容忍的超时时长。输入格式：数字 + 单位（s，m，h，ms）。

Copyright © 2016 - 2023 DaoCloud

重试

UI 元素	YAML 字段	描述
重试	spec.http.-name.retries	可选。默认关闭。重试功能用于定义当请求反馈为异常时再次发起请求的尝试次数。
重试次数	spec.http.-name.retries.attempts	可选。一个反馈异常的请求的可重试次数，重试间隔默认为 25ms。 当重试和超时功能均开启时，重试次数和重试超时的乘积与超时时长取最短者生效，因此实际的重试次数可能会小于设置值。
重试超时	spec.http.-name.retries.perTryTimeout	可选。每次重试的超时时长，默认与超时功能的设置（http.route.timeout）相同。输入格式：数字 + 单位（s，m，h，ms）。
重试条件	spec.http.-name.retries.retryOn	可选。允许重试的前提条件，该项下包含多个复选内容： - 5xx：当上游服务器返回 5xx 响应码或无响应（断开/重置/读取超时），envoy 将重试。 - refused-stream：当上游服务器用错误码 REFUSED_STREAM 重置数据流，envoy 将重试。 - gateway-error：仅针对 502、503、504 错误进行重试。 - retriable-status-codes：当上游服务器的返回码与响应码或请求头中 x-envoy-retriable-status-codes 定义相同，则重试。 - reset：当上游服务器无响应时（断开/重置/读取超时），将重试。 - connect-failure：当与上游服务器的连接失败（连接超时等）导致请求失败，将重试。 - retriable-headers：当上游服务器响应码与重试策略中定义匹配或与 x-ENVIGET-retriable-header-NAME 头匹配，将尝试重试。 - envoy-ratelimited：当存在报头 x-ENVISENT-ratelimited 时，将重试。 - retriable-4xx：当上游服务器返回 4xx 响应码时（目前仅有 409），envoy 将重试。

故障注入

UI 元素	YAML 字段	描述
故障注入	spec.http.-name.fault	可选。默认关闭。 故障注入功能用于在应用层向目标服务注入故障，并可提供“延迟”和“终止”两种故障类型；使用故障注入时，不能启用超时和重试功能。
延迟时长	spec.http.-name.fault.delay.fixedDelay	必填。请求响应可延迟的时长 输入格式：数字 + 单位（s，m，h，ms）。
故障注入占比	spec.http.-name.fault.delay.percentage	可选。所有请求中故障注入比率，默认 100%。
终止响应码	spec.http.-name.fault.abort.httpStatus	必填。用于终止当前请求的 http 返回码。
故障注入占比	spec.http.-name.fault.abort.percentage	可选。所有请求中故障注入比率，默认 100%。

代理虚拟服务

Copyright © 2016 - 2023 DaoCloud

YAML 字段为 `spec.http.-name.delegate`，默认关闭。

- 虚拟服务代理功能可以将路由配置拆分至主从两个虚拟服务中，由主虚拟服务完成基本设置和匹配规则，由代理虚拟服务完成具体路由规则。
- 开启代理功能后，仅主虚拟服务的路由匹配生效，代理虚拟服务的路由匹配规则不需设置。
- 代理虚拟服务的路由规则会和主虚拟服务路由规则合并执行。
- 代理功能不支持嵌套，仅主虚拟服务可以开启代理功能。
- 当虚拟服务有“路由目标/重定向”配置时，不可配置代理。

UI 元素	YAML 字段	描述
代理虚拟服务	spec.http.-name.delegate.name	必填。用于代理的从虚拟服务 注意： 一个已配置代理项的虚拟服务不可做为代理，即不可嵌套代理； 一个配置了 spec.hosts 字段的虚拟服务不可作为代理。
所属命名空间	spec.http.-name.delegate.namespace	可选。从虚拟服务所属命名空间，默认与主虚拟服务相同。

流量镜像

UI 元素		YAML 字段	描述。
流量镜像	n	spec.http.-name.mirror	可选。默认关闭。用于将请求流量复制至其他目标服务。
镜像至服务	y	spec.http.-name.mirror.host	必填。流量镜像的传输目标服务。
流量镜像占比	n	spec.http.-name.mirror.mirrorPercentage:value	可选。复制的请求流量与原请求流量的比率，默认 100%。
服务版本	n	spec.http.-name.mirror.subset	可选。服务版本列表内容来自当前目标服务的可用“目标规则”。

TLS 路由

- 可添加多条，执行顺序由上至下，排名在前的路由规则优先生效
- 多条 TLS 路由规则之间可以拖动排序
- 每条路由可收起，仅显示路由名称

路由匹配规则

YAML 字段为 `spec.tls.-name.match`。

通过端口（-port）和 SNI（-port.sniHosts）名称方式对请求做匹配，可添加多条规则，执行顺序由上至下，优先使用排名在前的匹配规则。

Copyright © 2016 - 2023 DaoCloud

路由目标

UI 元素	YAML 字段	描述
添加路由目标	spec.tls.-name.route	必填。添加路由目标信息，可添加多条，执行顺序由上至下
服务名称	spec.tls.-name.route.-destination.host	必填。目标服务名称，下拉列表包含当前命名空间下启用 tls 协议的所有服务
端口	spec.tls.-name.route.-destination.port.number	可选。目标服务端口
服务版本	spec.tls.-name.route.-destination.subset	可选。服务版本列表内容来自当前目标服务的可用“目标规则”。
权重	spec.tls.-name.route.-destination.weight	可选。本条“tls 路由”规则内各个“路由目标”所占流量的分配权重，各条规则的权重总和应为 100。

TCP 路由

- 可添加多条，执行顺序由上至下，排名在前的路由规则优先生效
- 多条 TCP 路由规则之间可以拖动排序
- 每条路由可收起，仅显示路由名称

路由匹配规则

UI 元素	YAML 字段	描述
添加路由匹配规则	spec.tcp.-name.match	可选。通过端口（-port）方式对请求做匹配，可添加多条规则，执行顺序由上至下，优先使用排名在前的匹配规则。
端口	spec.tcp.-name.match.-port	必填。tcp 端口号。

路由目标

UI 元素	YAML 字段	描述
添加路由目标	spec.tcp.-name.route	必填。添加路由目标信息，可添加多条，执行顺序由上至下。
服务名称	spec.tcp.-name.route.-destination.host	必填。目标服务名称，下拉列表包含当前命名空间下可用 tcp 协议的所有服务。
端口	spec.tcp.-name.route.-destination.port.number	可选。目标服务端口。
服务版本	spec.tcp.-name.route.-destination.subset	可选。服务版本列表内容来自当前目标服务的可用“目标规则”。
权重	spec.tcp.-name.route.-destination.weight	可选。本条“TCP路由”规则内各个“路由目标”所占流量的分配权重，各条规则的权重总和应为 100。

Copyright © 2016 - 2023 DaoCloud

5.10 Traffic monitor

流量监控

流量监控图表采用了 Istio 的原生 grafana 图表，目前采用了以下 4 个图表板块：

- 网格全局监控：网格各类资源统计
- 网格性能监控：网格控制面和数据面组件性能展示，并有基于组件版本的性能统计信息
- 服务监控：网格内所有注入边车的服务性能统计信息，并提供多种筛选方式，用户可以从命名空间、请求来源、服务版本等多个纬度筛选显示内容
- 工作负载监控：网格内所有注入边车的工作负载性能统计信息，并提供多种筛选方式，用户可以从命名空间、所属服务、请求来源、服务版本等多个纬度筛选显示内容

Copyright © 2016 - 2023 DaoCloud

流量拓扑

在左侧导航栏中点击 流量拓扑 打开页面，该功能展示了网格下所有服务的拓扑关系。用户可以选择 视图方式 及 命名空间 对服务节点进行筛选展示。

显示设置

命名空间边界 设置是否显示命名空间边界线框，选中将把同一命名空间下服务用边框框起，并标明命名空间名称.

Copyright © 2016 - 2023 DaoCloud

服务指标数据信息

点击任意服务，会弹出侧边栏基于协议类型分别展示服务相关指标：

- HTTP 服务：请求速率（RPM）、错误率（%）、平均时延（ms）
- TCP 服务：连接数（个）、接收吞吐量（B/S）、发送吞吐量（B/S）

健康状态

健康状态用于体现服务和连接的状态信息，分为正常、警告、异常和未知，通过比对错误率和延时指标数据判断。

- 正常（灰色）：错误率 = 0 并且 延时不超过 100 ms
- 警告（橙色）：0 < 错误率 <= 5% 或 100 ms < 延时 <= 200 ms
- 异常（红色）：错误率 > 5% 或延时 > 200 ms
- 未知（虚线）：未获得任何指标数据

Copyright © 2016 - 2023 DaoCloud

5.11 Upgrade

网格 Istio 版本升级

网格的 Istio 版本可持续升级，DaoCloud 提供了原生版本和定制版本两类升级版本。

- 原生版本：社区原生 Istio，无任何定制化修改；
- 定制版本：基于社区版本定制化（版本后缀：-mspider），在通信性能、治理能力、热升级能力等各方面做了功能增强及创新。

这两类版本升级过程相同，但不支持不同类型的混合升级。

当系统镜像仓库存在更新的 Istio 版本时，`网格列表` 中可升级的网格卡片会出现叹号提示图标，点击 `立即升级` 按钮即可进入升级向导；

Copyright © 2016 - 2023 DaoCloud

Istio 升级向导包含 `选择目标版本` 、 `环境检测` 、 `执行升级` 三步，升级完成后网格即可立即上线运行。具体步骤如下：

Copyright © 2016 - 2023 DaoCloud

1. 选择目标版本：在列表中选择期望升级的版本，升级后将无法回滚至低版本，建议谨慎选择。

2. 环境监测：系统将基于所选的目标版本，检测网格下各集群（k8s）版本是否符合升级要求，如果符合要求，将激活 `下一步` 按钮，否则，需要用户处理环境问题.

Copyright © 2016 - 2023 DaoCloud

- 如果集群（k8s）版本过低，可以在容器管理平台先升级集群（k8s）版本后，点击 重新检测 按钮；
- 如果集群（k8s）版本过高，建议回退至“选择目标版本”选择其他更高版本的 Istio。

Copyright © 2016 - 2023 DaoCloud

3. 执行升级：环境检测通过后，将进入升级阶段，该过程包含 `升级` 和 `健康检测` 两阶段。

- Istio 升级：Istio镜像拉取和控制面组件升级
- Istio 健康检测：Istio 控制面组件运行状态检查

Copyright © 2016 - 2023 DaoCloud

升级完成后，回到网格列表页面，可以看到网格的 Istio 版本已变更。

```
![执行升级](../../images/IstioUpdate06.png)
```

!!! note

```
- 升级过程一旦开始将无法终止，升级期间建议不要对网格执行任何设置操作。
- 更直观的操作演示，可参阅[视频教程](../../../videos/mspider.md)。
```


Copyright © 2016 - 2023 DaoCloud

边车版本升级

网格的 Istio 版本升级后，将触发边车升级功能。边车升级可分为 热升级 和 重启升级 两种升级方式。

- 热升级：道客订制版本 Istio 下边车可在不重启用户 Pod 的情况下完成边车升级，实现业务无中断；
- 重启升级：社区原生Istio 或不满足热升级环境要求的订制版 Istio 下边车升级方式，需重启用户 Pod。

完成 Istio 版本升级后，进入 工作负载边车管理 界面，具备升级条件的工作负载将出现叹号提示信息，选中期望升级的工作负载，将出现 边车升级 按钮。

点击 边车升级 按钮将进入 边车升级版本 向导，该向导分为 环境检测 、 选择目标版本 、 执行升级 三个步骤，在“热升级”和“重启升级”两种方式下，操作会有一定差别。

热升级

1. 环境检测：该步骤中将检测集群环境是否满足热升级需求，检测项包括以下三项：

- Istio 版本：是否是定制版本（版本后缀：-mspider）
- K8s 版本：是否符合热升级要求范围
- EphemeralContainer：是否已启用

以上三项满足后将在后续步骤中进入热升级流程。

Copyright © 2016 - 2023 DaoCloud

2. 边车升级版本：热升级流程中，可以选择期望升级的边车版本，默认为最新版本，如果选择了其他版本，相关 Pod 重启后也会自动升级至最新版本。

3. 执行升级：升级页面中展示了所选工作负载及相关边车信息，点击 `一键升级` 将启动升级过程。

Copyright © 2016 - 2023 DaoCloud

重启升级

1. 环境检测：在检测阶段如果检测项不满足热升级要求，之后两个步骤将进入重启升级流程。

Copyright © 2016 - 2023 DaoCloud

2. 边车升级版本 在重启升级流程中，无法选择版本，仅支持升级至最新版本。

3. 执行升级：升级页面中展示了所选工作负载的基本信息及边车版本信息，点击 `一键升级` 将立即重启 Pod，请务必谨慎操作。

Copyright © 2016 - 2023 DaoCloud

!!! note

升级过程中关闭升级向导不会中断当前升级任务。

Copyright © 2016 - 2023 DaoCloud