

应用服务网格

常见问题

文档版本 02
发布日期 2022-10-09



版权所有 © 华为技术有限公司 2024。保留一切权利。

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

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：

<https://www.huawei.com/cn/psirt/vul-response-process>

如企业客户须获取漏洞信息，请参见如下网址：

<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

目录

1 网格集群	1
1.1 启用服务网格后，状态一直为安装中	1
1.2 卸载服务网格后，状态一直为未就绪	1
1.3 创建网格为什么会自动创建一个 otel-collector 工作负载？	2
2 网格管理	7
2.1 为什么我的集群不能启用网格？	7
2.2 Istio 卸载之后，为什么独享节点还在？	7
2.3 如何升级 ICAgent？	8
2.4 如何为集群开放命名空间注入？	8
2.5 某些工作负载不注入 Sidecar，该如何配置？	9
2.6 Sidecar 未就绪导致 Pod 启动失败	9
2.7 金丝雀升级失败常见场景及解决方案	12
3 添加服务	15
3.1 添加的对外访问方式不能生效，如何排查？	15
3.2 一键创建体验应用为什么启动很慢？	15
3.3 一键创建体验应用部署成功以后，为何不能访问页面？	15
3.4 添加路由时，为什么选不到对应的服务？	16
3.5 如何为普通任务（Job）和定时任务（CronJob）类型负载注入 sidecar	16
4 灰度发布	18
4.1 灰度发布部署版本为什么不能更换镜像？	18
4.2 基于请求内容发布策略对一些服务为什么没有生效？	18
4.3 多端口的服务创建灰度任务时报不合法的请求体	19
5 流量治理	20
5.1 流量治理页面，我创建的集群、命名空间和应用为什么不显示？	20
5.2 如何调整 istio-proxy 容器 resources requests 取值？	20
5.3 ASM 支持 HTTP/1.0 吗？	21
5.4 服务网格如何支持自定义网段或端口拦截规则？	22
5.5 网关如何配置最大并发流 max_concurrent_streams	25
6 流量监控	27
6.1 Jaeger/Zipkin OSC 插件安装指导	27

1 网格集群

1.1 启用服务网格后，状态一直为安装中

问题描述

为CCE集群启用服务网格（即购买网格）后，网格状态一直显示为“安装中”，鼠标放上去提示“正在启用istio服务网格：开通用户安全组规则成功”。

问题定位

登录CCE控制台，在“资源管理 > 命名空间”中查看对应集群的istio-system命名空间是否存在。

原因分析

存在istio-system命名空间残留。

解决方法

删除已有的istio-system命名空间后即可继续安装。

1.2 卸载服务网格后，状态一直为未就绪

问题描述

在ASM控制台卸载服务网格后，网格状态一直显示为“未就绪”。

问题定位

步骤1 登录CCE控制台，进入对应集群详情页，在左侧导航栏选择“运维 > 模板管理”。

步骤2 单击“模板实例”页签，查看模板实例和卸载失败最新事件。

可以看到istio-master模板实例的执行状态为“卸载失败”，并且最新事件提示如下信息：

```
deletion failed with 1 error(s): clusterroles:rbac.authorization.k8s.io "istio-cleanup-secrets-istio-system"
already exists
```

----结束

原因分析

helm对中断状态支持不好，客户异常操作会导致istio的helm模板卡在中间状态，使卸载过程中留下残留资源，从而导致卸载失败。

解决方法

步骤1 通过kubectl连接到CCE集群。

步骤2 执行以下命令，清理istio相关资源。

```
kubectl delete ServiceAccount -n istio-system `kubectl get ServiceAccount -n istio-system | grep istio | awk
'{print $1}'`
kubectl delete ClusterRole -n istio-system `kubectl get ClusterRole -n istio-system | grep istio | awk '{print
$1}'`
kubectl delete ClusterRoleBinding -n istio-system `kubectl get ClusterRoleBinding -n istio-system | grep istio
| awk '{print $1}'`
kubectl delete job -n istio-system `kubectl get job -n istio-system | grep istio | awk '{print $1}'`
kubectl delete crd -n istio-system `kubectl get crd -n istio-system | grep istio | awk '{print $1}'`
kubectl delete mutatingwebhookconfigurations -n istio-system `kubectl get
mutatingwebhookconfigurations -n istio-system | grep istio | awk '{print $1}'`
```

步骤3 登录ASM控制台，重新执行卸载操作。

----结束

1.3 创建网格为什么会自动创建一个 otel-collector 工作负载?

问题描述

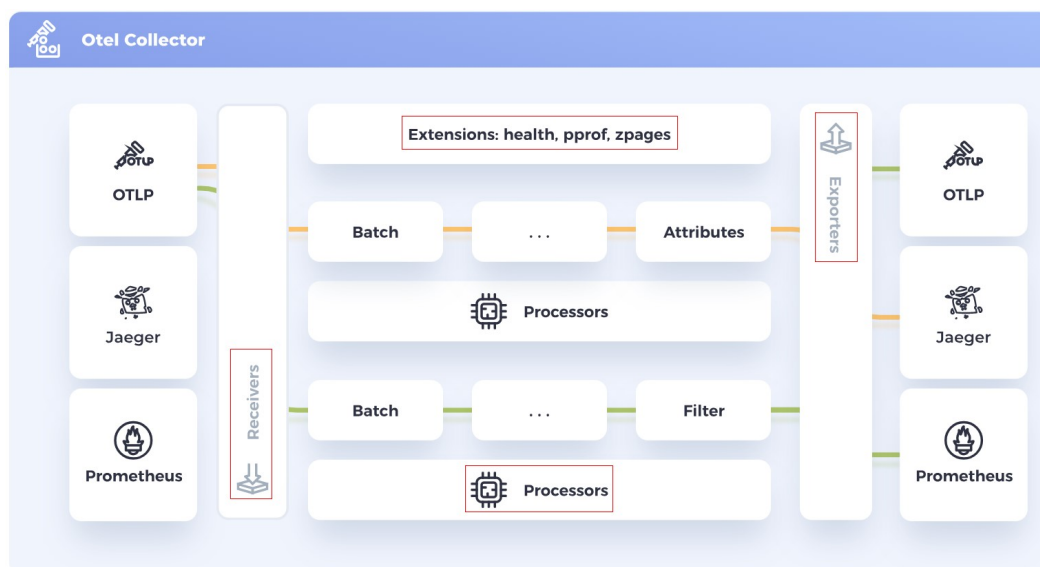
创建网格会自动创建一个otel-collector工作负载。

原因分析

ASM服务网格对接至集群后，会在命名空间monitoring下创建一个otel-collector工作负载。创建这个工作负载的原因是需要利用其对envoy收集遥测数据（trace、log、metric），并进行处理，导出到相应的后端，实现网格的可观测性。

otel-collector架构简介

图 1-1 otel-collector 架构图



如上图的架构图所示，otel-collector包含了四个模块：

- Receivers
接收器Receivers是遥测数据进入otel-collector的方式，可以是推送或拉取。Receivers可以以多种格式接收遥测数据，例如上图中的OTLP、Jaeger、Prometheus格式。
- Processors
处理器Processors用于处理Receivers收集到的数据，例如常用的batch处理器，用于对遥测数据进行批处理。
- Exporters
导出器Exporters是将遥测数据发送到指定后端的方式，它帮助我们更好地可视化和分析遥测数据。
- Extensions
扩展主要适用于不涉及处理遥测数据的任务。扩展是可选的，比如可以增加一个health_check的健康检查功能，获取有关Collector健康状况的信息。

otel-collector在ASM基础版网格中的使用

可通过以下命令获取otel-collector工作负载的配置信息：

```
[root@root ~]# kubectl get cm -n monitoring otel-collector-conf -oyaml
apiVersion: v1
data:
  otel-collector-config: |-
    receivers:
      zipkin: { }
      prometheus:
        config:
          scrape_configs:
            - job_name: 'istio-mesh'
              scrape_interval: 15s
              metrics_path: /stats/prometheus
              kubernetes_sd_configs:
                - role: pod
              relabel_configs:
                - source_labels: [ __meta_kubernetes_pod_container_port_name ]
                  action: keep
                  regex: http-envoy-prom
              metric_relabel_configs:
                - source_labels: [ __name__ ]
                  action: keep
                  regex: istio.*
                - source_labels: [ __name__ ]
                  regex: 'istio_build'
                  action: drop
                - source_labels: [ __name__ ]
                  regex: 'istio_response_bytes.*'
                  action: drop
                - source_labels: [ __name__ ]
                  regex: 'istio_request_bytes.*'
                  action: drop
    processors:
      batch:
      memory_limiter:
```

以在基础版网格获取到的配置文件为例：

- receivers配置项定义了可以选择以zipkin、prometheus两种协议从envoy获取遥测数据，其中prometheus定义了以每15s的间隔从/stats/prometheus路径下抓取数据。

```
otel-collector-config: |-
  receivers:
    zipkin: { }
    prometheus:
      config:
        scrape_configs:
          - job_name: 'istio-mesh'
            scrape_interval: 15s
            metrics_path: /stats/prometheus
            kubernetes_sd_configs:
              - role: pod
            relabel_configs:
              - source_labels: [ __meta_kubernetes_pod_container_port_name ]
                action: keep
                regex: http-envoy-prom
            metric_relabel_configs:
              - source_labels: [ __name__ ]
                action: keep
                regex: istio.*
              - source_labels: [ __name__ ]
                regex: 'istio_build'
                action: drop
              - source_labels: [ __name__ ]
                regex: 'istio_response_bytes.*'
                action: drop
              - source_labels: [ __name__ ]
                regex: 'istio_request_bytes.*'
                action: drop
```


- processors配置项定义了batch、memory_limiter两种对数据处理的方式，分别是批处理和内存限制。

```
processors:  
  batch:  
  memory_limiter:  
    check_interval: 1s  
    limit_percentage: 80  
    spike_limit_percentage: 20
```

- exporters配置项定义了将处理过的遥测数据导出至apm服务器。

```
exporters:  
  apm:  
    address: "100.79.1.215:8923"  
    project_id: 719217bc273743ea8d7ac1ae8bc34480  
    cluster_id: d7491b95-5111-11ee-8779-0255ac100b05
```

- extensions配置项定义了health_check扩展，其用于获取有关otel-collector健康状况的信息。

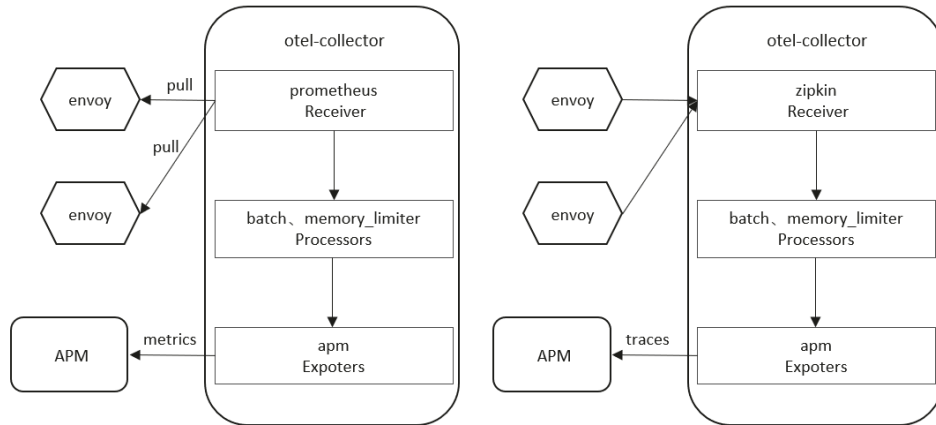
```
extensions:  
  health_check:  
    endpoint: 127.0.0.1:13133
```

- service部分用于配置otel-collector实际会采用哪些上述定义好的配置项。

```
service:  
  telemetry:  
    logs:  
      level: info  
  extensions: [ health_check ]  
  pipelines:  
    metrics/apm:  
      receivers: [ prometheus ]  
      processors: [ memory_limiter, batch ]  
      exporters: [ apm ]  
    traces/apm:  
      receivers: [ zipkin ]  
      processors: [ memory_limiter, batch ]  
      exporters: [ apm ]
```

比如上述配置文件中service项，其配置了两个pipeline分别用于处理metrics数据和traces数据（注：一个pipeline是一组receivers, processors, 和exporters的集合），以及配置了logs输出级别为info及以上。其处理架构如下图所示。

图 1-2 metrics、traces 处理架构图



解决方法

无需处理。

2 网格管理

2.1 为什么我的集群不能启用网格？

问题描述

集群不能启用网格。

原因分析

暂不支持v1.15以下版本集群启用网格。

解决方法

步骤1 检查您的集群版本，目前仅对v1.15、v1.17、v1.19、v1.21或v1.23版本的集群生效。

步骤2 检查您的浏览器，请尽量使用Chrome浏览器访问服务，火狐等浏览器可能因为适配的问题，导致启用网格按钮灰化。

----结束

2.2 Istio 卸载之后，为什么独享节点还在？

问题描述

Istio卸载后独享节点还在。

原因分析

Istio仅会卸载Istio相关控制面组件，不会主动卸载您的节点资源。

解决方法

卸载后的节点，您可以作为普通负载节点使用。如不再需要，请登录CCE控制台，进入对应集群详情页，在“集群 > 节点管理”中删除该节点。

2.3 如何升级 ICAgent?

步骤1 登录应用服务网格ASM控制台，在左侧导航栏选择“监控中心”。

步骤2 跳转至应用性能管理界面后，选择左侧导航栏的“采集管理 > Agent管理”，选择对应的集群后单击“升级ICAgent”。

----结束

2.4 如何为集群开放命名空间注入?

为集群的命名空间注入sidecar时，若集群未开放命名空间注入，请参考如下指导修改集群配置：

步骤1 通过kubectl连接集群。

步骤2 执行**kubectl get iop -nistio-system**，查询iop资源。

- 若回显如下，表示存在iop资源，请执行**步骤3**。

```
user@dts2fot109u4ymb-machine:~$ kubectl get iop -nistio-system
NAME          REVISION  STATUS  AGE
data-plane    1         HEALTHY 69d
```

- 若回显如下，表示不存在iop资源，请执行**步骤4**。

```
web-terminal-7b778fc945-9m2hf:~# kubectl get iop -nistio-system
No resources found in istio-system namespace.
```

步骤3 执行**kubectl edit iop -nistio-system data-plane**，修改autoInject配置项。其中，*data-plane*为上一步查询的iop资源名称，请替换为实际值。

```
global:
  defaultPodDisruptionBudget:
    enabled: true
  hub: *.*.*:20202/asm
  logging:
    level: default:info
  meshID: test-payment
  multiCluster:
    clusterName: test-yy
  network: test-yy-network
  proxy:
    autoInject: enabled
  remotePilotAddress: *.*.*
  tag: 1.8.6-r1-20220512225026
```

步骤4 执行**kubectl edit cm -nistio-system istio-sidecar-injector**，修改istio-sidecar-injector配置项。

```
data:
  config: |-
    policy: enabled
```

----结束

2.5 某些工作负载不注入 Sidecar，该如何配置？

为集群的命名空间开启Sidecar注入后，该命名空间下所有工作负载关联的Pod将自动注入Sidecar。不过有些工作负载因为种种原因不能注入Sidecar，可参考如下指导进行配置：

步骤1 登录CCE控制台，进入对应集群详情页，在左侧导航栏选择“资源 > 工作负载”。

步骤2 单击工作负载所在行的“编辑YAML”。

步骤3 找到spec.template.metadata.annotations字段，添加sidecar.istio.io/inject: 'false'。

```
annotations:
  sidecar.istio.io/inject: 'false'
```

```
107 spec:
108   replicas: 1
109   selector:
110     matchLabels:
111       app: reviews
112       version: v1
113   template:
114     metadata:
115       creationTimestamp: null
116     labels:
117       app: reviews
118       release: istio-bookinfo
119       version: v1
120     annotations:
121       sidecar.istio.io/inject: 'false'
```

您可以单击[Automatic Sidecar Injection](#)了解更多Sidecar注入的知识。

----结束

2.6 Sidecar 未就绪导致 Pod 启动失败

问题背景

加入网格的服务有时可能遇到Pod启动失败，且一直重启。排查原因发现业务容器与外部通信时流量会经过istio-proxy容器，但业务容器比istio-proxy容器先启动，在istio-proxy容器没启动成功时，业务容器已经启动，与外部通信将会失败，Pod一直重启。

规避方案

在Istio 1.7及以后版本，社区通过给istio-injector注入逻辑增加一个叫HoldApplicationUntilProxyStarts的开关来解决该问题，开关打开后，Proxy将会注入到第一个Container，istio-proxy容器先于业务容器启动。

开关配置分为全局和局部两种，下面介绍两种启用方法。

须知

需要注意的是，打开开关后，意味着业务容器需要等Sidecar完全Ready后才能启动，会让Pod启动速度变慢一些。在需要快速扩容应对突发流量场景可能会显得吃力，所以建议您自行评估业务场景，利用局部配置的方法，只给需要的业务打开此开关。

全局配置

- a. 执行以下命令，编辑IOP CR资源。

```
kubectl edit iop private-data-plane -n istio-system
```

在spec.values.global.proxy字段下添加以下配置：

```
holdApplicationUntilProxyStarts: true
```

```
values:
  gateways:
    istio-egressgateway:
      autoscaleEnabled: false
      labels:
        app: istio-egressgateway
      tolerations:
        - effect: NoExecute
          key: istio
          operator: Exists
    istio-ingressgateway:
      autoscaleEnabled: false
      customService: true
      labels:
        app: istio-ingressgateway
      replicaCount: 1
      tolerations:
        - effect: NoExecute
          key: istio
          operator: Exists
  global:
    defaultPodDisruptionBudget:
      enabled: true
    hub: swr.cn-north-7.myhuaweicloud.com/asm
    logging:
      level: default:info
    meshID: envoy-critical
    multiCluster:
      clusterName: test-ysl-multi
    proxy:
      autoInject: enabled
      holdApplicationUntilProxyStarts: true
```

- b. 执行以下命令，确认最新日志无报错。

```
kubectl logs -n istio-operator $(kubectl get po -n istio-operator | awk '{print $1}' | grep -v NAME)
```

- c. 执行以下命令，确认IOP CR是正常状态。

```
kubectl get iop -n istio-system
```

```
[root@lx666-14467 ~]# kubectl get iop -n istio-system
NAME                REVISION    STATUS    AGE
private-data-plane  1           HEALTHY   6d2h
[root@lx666-14467 ~]#
```

- d. 执行以下命令，滚动升级已添加到网格的服务。

```
kubectl rollout restart deployment nginx -n default
```

其中，nginx为示例服务，default为命名空间，请替换为实际取值。

- e. 执行以下命令，确认Pod重启成功。

```
kubectl get pod -n default | grep nginx
```

```
[root@lx666-14467 ~]# kubectl get pod -n default | grep nginx
nginx-6b4959fffb-pr8t8    2/2    Running    0    14s
[root@lx666-14467 ~]#
```

- f. 执行以下命令，确认Pod正常添加了postStart lifecycle，并且istio-proxy容器放在了第一个位置。

```
kubectl edit pod nginx-7bc96f87b9-l4dbl
```

```
- name: ISTIO_META_CLUSTER_ID
  value: test-yy1-multi
image: swr.cn-north-7.myhuaweicloud.com/asm/proxyv2:1.13.9-r1-20221110212800
imagePullPolicy: IfNotPresent
lifecycle:
  postStart:
    exec:
      command:
      - pilot-agent
      - wait
name: istio-proxy
ports:
```

- **局部配置**

如果使用Istio 1.8及其以上的版本，可以为需要打开此开关的Pod加上proxy.istio.io/config注解，将holdApplicationUntilProxyStarts置为true。以default命名空间下nginx服务为例，用户其他服务操作类似。

```
kubectl edit deploy nginx -n default
```

在spec.template.metadata.annotations字段下添加以下配置：

```
proxy.istio.io/config: |
  holdApplicationUntilProxyStarts: true
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "6"
    description: ""
  creationTimestamp: "2022-11-24T07:55:31Z"
  generation: 6
  labels:
    appgroup: ""
    version: v1
  name: tomcat
  namespace: default
  resourceVersion: "55550644"
  uid: cd5dbfe8-83cc-4964-86fc-f657c85e852d
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: tomcat
      version: v1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      annotations:
        kubectl.kubernetes.io/restartedAt: "2022-11-25T10:35:02+08:00"
        proxy.istio.io/config: |
          holdApplicationUntilProxyStarts: true
    creationTimestamp: null
```

2.7 金丝雀升级失败常见场景及解决方案

进行金丝雀升级时，升级失败的常见场景和解决方案：

1. CRD检查失败。
解决办法：新版本Istio 将不支持部分CRD，包括：clusterrbacconfigs、serviceroles、servicerolebindings、policies。若您在当前版本存在即将废弃的资源，则需要删除后再升级。
2. 升级前检查网关配置信息时，Istio 网关标签错误。
解决办法：Istio 网关标签（matchLabels）必须为 {app: istio-ingressgateway, istio: ingressgateway}。
3. 升级前插件检查失败。
解决办法：ASM从1.8版本开始不再支持如下插件（tracing, kiali, grafana, prometheus）部署，升级前需要将上述插件卸载。您可以自行安装开源版本插件，或者使用APM。
4. 升级前集群状态检查任务失败。
解决办法：升级前会检查集群状态，若集群状态异常则无法进行网格升级。
5. 升级前资源检查任务失败。
解决办法：金丝雀升级需要有充足资源。

6. 升级前集群版本检查任务失败。

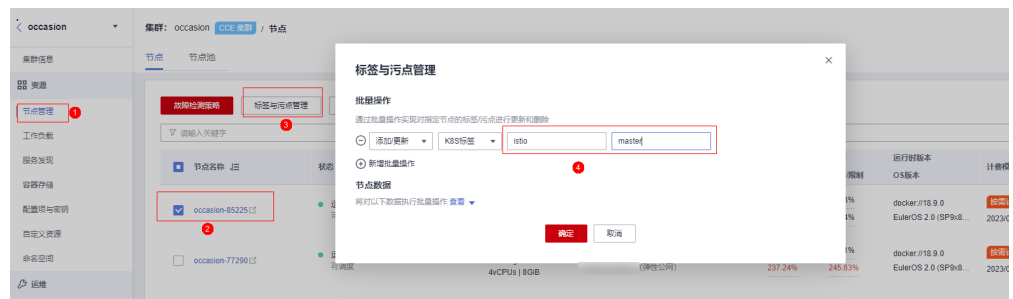
解决办法： 网络支持的版本如下：

网格版本	支持的集群版本
1.3	1.13,1.15,1.17,1.19
1.6	1.15,1.17
1.8	1.15,1.17,1.19,1.21
1.13	1.21,1.23
1.15	1.21,1.23,1.25,1.27
1.18	1.25,1.27,1.28

7. 升级前组件亲和性检查失败。

解决办法： 若您从非金丝雀版本升级到金丝雀版本，打了istio:master labels的节点数量小于两倍的istiod/ingressgateway/egressgateway 数量，则需要将节点数量扩大到两倍或者将istiod、ingressgateway、egressgateway pod反亲和性设置为尽量满足。

- **方法一：** 增加添加istio: master节点，可以从CCE console上进行操作。



- **方法二：** 修改pod反亲和策略，可在CCE界面修改yaml。



preferredDuringSchedulingIgnoredDuringExecution:

```

- weight: 1
  podAffinityTerm:
    labelSelector:
      matchExpressions:
        - key: app
          operator: In
          values:
            - istiod (如果是ingressgateway则为istio-egressgateway、istio-ingressgateway)
    namespaces:
      - istio-system
    topologyKey: kubernetes.io/hostname
    
```

或者在CCE界面升级设置工作负载反亲和性，改为尽量满足。



8. 升级前命名空间自动注入检查失败。

解决办法：若您从专有网格迁移至基础网格，命名空间存在已经注入的pod，但是该命名空间未开启自动注入，则需要开启该命名空间**自动注入**。

3 添加服务

3.1 添加的对外访问方式不能生效，如何排查？

出现上述问题可能是访问相关的资源配置有缺失或错误，请按照如下方法进行排查：

- 通过弹性负载均衡服务界面查看使用的ELB是否成功监听使用的外部端口和弹性云服务器。
- 登录集群，使用`kubectl get gateway -n istio-system`命令查看使用的gateway是否配置好使用的IP/域名和端口。使用`kubectl get svc -n istio-system`命令查看使用的ingressgateway是否有对应的IP和端口，且未处于pending状态。
- 核实加入服务网格的内部访问协议和添加网络配置的外部访问协议一致。
- 如果通过浏览器访问出现“ERR_UNSAFE_PORT”错误，是因为该端口被浏览器识别为危险端口，此时应更换为其他外部端口。

3.2 一键创建体验应用为什么启动很慢？

体验应用包含productpage、details、ratings和reviews 4个服务，需要创建所有相关的工作负载和Istio相关的资源（DestinationRule、VirtualService、Gateway）等，因此创建时间较长。

3.3 一键创建体验应用部署成功以后，为何不能访问页面？

问题描述

一键创建体验应用部署成功后不能访问页面。

原因分析

弹性负载均衡ELB未成功监听端口。

解决方法

请在弹性负载均衡ELB中查看该端口监听器是否创建，后端服务器健康状态是否正常。弹性负载均衡监听器创建方法请参见[监听器](#)。

3.4 添加路由时，为什么选不到对应的服务？

添加路由时，目标服务会根据对应的网关协议进行过滤。过滤规则如下：

- HTTP协议的网关可以选择HTTP协议的服务
- TCP协议的网关可以选择TCP协议的服务
- GRPC协议的网关可以选择GRPC协议的服务
- HTTPS协议的网关可以选择HTTP、GRPC协议的服务
- TLS协议的网关如果打开了TLS终止，只能选择TCP协议的服务；关闭了TLS终止，只能选择TLS协议的服务

3.5 如何为普通任务（Job）和定时任务（CronJob）类型负载注入 sidecar

前置条件

- 确认使用ASM1.15.5-r3及以上版本创建网格。
- 默认场景下，对普通任务（Job）和定时任务（CronJob）类型负载创建的Pod不进行sidecar注入，如果需要注入请在创建工作负载时，设置高级参数“标签与注解> Pod标签” sidecar.istio.io/inject: 'true'。如下图：

高级配置

标签与注解

Pod标签 sidecar.istio.io/inject = true 确认添加

app = x | version = v1 x

Pod注解 键 = 值 确认添加 使用指南

参考CronJob示例：

```
kind: CronJob
apiVersion: batch/v1
metadata:
  name: mycronjob
  namespace: default
spec:
  schedule: */1 * * * *
  jobTemplate:
    spec:
      template:
        metadata:
          creationTimestamp: null
        labels:
          app: mycronjob
          sidecar.istio.io/inject: 'true'
```

- 要使用Job/CronJob类型的话，需要在容器中使用指令退出sidecar。

任务完成后 sidecar 退出

通过调用istio-proxy接口curl -sf -XPOST http://127.0.0.1:15000/quitquitquit，在Job工作完成后退出istio-proxy。

参考CronJob示例:

```
kind: CronJob
apiVersion: batch/v1
metadata:
  name: mycronjob
  namespace: default
spec:
  schedule: */1 * * * *
  concurrencyPolicy: Forbid
  suspend: false
  jobTemplate:
    metadata:
      creationTimestamp: null
    spec:
      template:
        metadata:
          creationTimestamp: null
        labels:
          app: cronjob1
          sidecar.istio.io/inject: 'true'
          version: v1
        spec:
          containers:
            - name: mycronjob-1
              image: 'busybox:latest'
              command:
                - /bin/bash
                - '-c'
              args:
                - |
                  trap "curl --max-time 2 -s -f -XPOST http://127.0.0.1:15000/quitquitquit" EXIT
                  while ! curl -s -f http://127.0.0.1:15020/healthz/ready; do sleep 1;done
                  sleep 2
                  date; echo Hello from the Kubernetes cluster<Your Job command/真实业务运行命令>
...

```

4 灰度发布

4.1 灰度发布部署版本为什么不能更换镜像？

问题描述

灰度发布部署灰度版本时不能更换镜像类型。

原因分析

灰度发布针对服务的同一镜像，只允许选择不同的版本号。

解决方法

将所需镜像打包成同一镜像的不同版本并上传至镜像仓库。

4.2 基于请求内容发布策略对一些服务为什么没有生效？

问题描述

基于请求内容发布策略没有生效。

原因分析

基于请求内容发布策略只对直接访问的入口服务有效。

解决方法

如果希望对所有服务有效，需要业务代码对HEAD信息传播。

4.3 多端口的服务创建灰度任务时报不合法的请求体

问题描述

多端口的服务创建灰度任务时报不合法的请求体，提示“ASM.0002 不合法的请求体”。

排查思路

登录ASM控制台，按“F12”，切换到Network页签查看接口。发现post请求创建release接口全部返回400，查看返回内容提示如下信息：

```
some ports of the service have been configured with routes, ports=[%v]
```

原因分析

配置诊断正常的多端口服务删除了其中的一些端口，如service01存在80和81端口，在CCE界面删除了81端口。

解决方法

恢复删除的service端口。

5 流量治理

5.1 流量治理页面，我创建的集群、命名空间和应用为什么不显示？

1. 请确保您的集群已经成功启用Istio。
2. 确认已经在“服务列表”页面添加了至少一个服务，且服务的状态为“运行中”。
3. 确认完上述几点后，如果还没有数据，请检查您是否自行卸载过集群内的ICAgent系统组件。

5.2 如何调整 istio-proxy 容器 resources requests 取值？

istio-proxy容器资源占用大小的默认配置如下。如果不符合要求，可按照实际需求进行修改。

```
resources:
  limits:
    cpu: "2"
    memory: 512Mi
  requests:
    cpu: "1"
    memory: 512Mi
```

方法一：调整网格中的所有服务

一次配置对所有加入网格的服务的istio-proxy容器资源占用进行调整。

步骤1 执行以下命令修改ConfigMap。

```
kubectl edit cm istio-sidecar-injector -n istio-system
```



```
315     resources:
316     {{ if or (isset .ObjectMeta.Annotations `sidecar.istio.io/proxyCPU`) (isset .ObjectMeta
ar.istio.io/proxyLimitCPU`) (isset .ObjectMeta.Annotations `sidecar.istio.io/proxyLimitMemory`) -
317     requests:
318     {{ if (isset .ObjectMeta.Annotations `sidecar.istio.io/proxyCPU`) -}}
319     cpu: "{{ index .ObjectMeta.Annotations `sidecar.istio.io/proxyCPU` }}"
320     {{ end }}
321     {{ if (isset .ObjectMeta.Annotations `sidecar.istio.io/proxyMemory`) -}}
322     memory: "{{ index .ObjectMeta.Annotations `sidecar.istio.io/proxyMemory` }}"
323     {{ end }}
324     limits:
325     {{ if (isset .ObjectMeta.Annotations `sidecar.istio.io/proxyLimitCPU`) -}}
326     cpu: "{{ index .ObjectMeta.Annotations `sidecar.istio.io/proxyLimitCPU` }}"
327     {{ end }}
328     {{ if (isset .ObjectMeta.Annotations `sidecar.istio.io/proxyLimitMemory`) -}}
329     memory: "{{ index .ObjectMeta.Annotations `sidecar.istio.io/proxyLimitMemory` }}"
330     {{ end }}
331     {{ else -}}
332     {{- if .Values.global.proxy.resources }}
333     #{{ toYaml .Values.global.proxy.resources | indent 4 }}
334     requests:
335     cpu: xxx
336     memory: xxx
337     limits:
338     cpu: xxx
339     memory: xxx
340     {{- end }}
341     {{ end -}}
```

步骤2 重启istio-system命名空间下的istio-sidecar-injector Pod。

步骤3 重启业务服务Pod，多实例滚动升级不会断服。

----结束

方法二：调整网格中的某个服务

步骤1 修改服务的yaml文件。

```
kubectl edit deploy <nginx> -n <namespace>
```

步骤2 在spec.template.metadata.annotations下添加如下配置（大小仅供参考，请自行替换）。

```
sidecar.istio.io/proxyCPU: 500m
sidecar.istio.io/proxyLimitCPU: 500m
sidecar.istio.io/proxyLimitMemory: 1024Mi
sidecar.istio.io/proxyMemory: 1024Mi
```

Istio 1.8网格的配置项有差异，如下所示：

```
sidecar.istio.io/proxyCPU: 500m
sidecar.istio.io/proxyCPULimit: 500m
sidecar.istio.io/proxyMemoryLimit: 1024Mi
sidecar.istio.io/proxyMemory: 1024Mi
```

步骤3 修改后服务滚动升级，确保不会断服。

----结束

5.3 ASM 支持 HTTP/1.0 吗？

问题现象

Istio 默认不支持 HTTP/1.0。

原因分析

Istio中负责流量转发的是Envoy，负责分配规则的是Pilot。Pilot的环境变量PILOT_HTTP10默认为0，即默认不支持HTTP/1.0。

解决方法

编辑iop中的环境变量.values.pilot.env.PILOT_HTTP10设置为1，为pilot传递PILOT_HTTP10环境变量即可。

```
pilot:
  autoscaleEnabled: false
  env:
    DEPLOY_TYPE: cop
    ENABLE_LEGACY_FSGROUP_INJECTION: false
    PILOT_HTTP10: 1
  replicaCount: 1
  resources:
    limits:
      cpu: 2000m
      memory: 4096Mi
    requests:
      cpu: 100m
      memory: 128Mi
```

5.4 服务网格如何支持自定义网段或端口拦截规则？

操作场景

某些场景下，用户希望能够指定拦截的IP网段，只有IP网段内的请求会被代理拦截；某些场景下，需要配置拦截规则仅针对特定端口的请求生效。以下将介绍两种拦截网段的配置方式。

负载级别配置拦截 IP 网段

通过配置业务deployment文件，可以在负载级别配置IP网段拦截：

执行 `kubectl edit deploy -n user_namespace user_deployment`

1. 在deployment.spec.template.metadata.annotations中配置IP网段拦截
traffic.sidecar.istio.io/includeOutboundIPRanges:

```
type: RollingUpdate
template:
  metadata:
    annotations:
      asm/updateTimestamp: "2023-03-23T03:49:21Z"
      sidecar.istio.io/proxyCPU: "0.1"
      sidecar.istio.io/proxyCPULimit: "2"
      sidecar.istio.io/proxyMemory: 128Mi
      sidecar.istio.io/proxyMemoryLimit: 2048Mi
      traffic.sidecar.istio.io/includeOutboundIPRanges: 192.168.0.1/24
    creationTimestamp: null
  labels:
    app: nginx
    version: v1
```

2. 在deployment.spec.template.metadata.annotations中配置IP网段不拦截traffic.sidecar.istio.io/excludeOutboundIPRanges:

```
template:
  metadata:
    annotations:
      asm/updateTimestamp: "2023-03-23T03:49:21Z"
      sidecar.istio.io/proxyCPU: "0.1"
      sidecar.istio.io/proxyCPULimit: "2"
      sidecar.istio.io/proxyMemory: 128Mi
      sidecar.istio.io/proxyMemoryLimit: 2048Mi
      traffic.sidecar.istio.io/excludeOutboundIPRanges: 192.168.0.1/24
    creationTimestamp: null
  labels:
    app: nginx
    version: v1
```

注意：上述操作会导致业务容器滚动升级。

负载级别指定端口配置出入流量拦截

通过修改业务deployment文件，可以在负载级别配置端口上的出入流量拦截规则：

执行 `kubectl edit deploy -n user_namespace user_deployment`

1. 在deployment.spec.template.metadata.annotations中配置入流量指定端口不拦截traffic.sidecar.istio.io/excludeInboundPorts:

```
type: RollingUpdate
template:
  metadata:
    annotations:
      asm/updateTimestamp: "2023-06-01T01:40:56Z"
      traffic.sidecar.istio.io/excludeInboundPorts: 3306,6379
    creationTimestamp: null
  labels:
    app: echo
```

2. 在deployment.spec.template.metadata.annotations中配置入流量指定端口拦截traffic.sidecar.istio.io/includeInboundPorts:

```
type: RollingUpdate
template:
  metadata:
    annotations:
      asm/updateTimestamp: "2023-06-01T01:40:56Z"
      traffic.sidecar.istio.io/includeInboundPorts: 3306,6379
    creationTimestamp: null
  labels:
    app: echo
```

3. 在deployment.spec.template.metadata.annotations中配置出流量指定端口不拦截 traffic.sidecar.istio.io/excludeOutboundPorts:

```
template:
  metadata:
    annotations:
      asm/updateTimestamp: "2023-06-01T01:40:56Z"
      traffic.sidecar.istio.io/excludeOutboundPorts: 3306,6379
    creationTimestamp: null
  labels:
```

4. 在deployment.spec.template.metadata.annotations中配置出流量指定端口拦截 traffic.sidecar.istio.io/includeOutboundPorts:

```
type: RollingUpdate
template:
  metadata:
    annotations:
      asm/updateTimestamp: "2023-06-01T01:40:56Z"
      traffic.sidecar.istio.io/includeOutboundPorts: 3306,6379
    creationTimestamp: null
  labels:
    app: echo
    sidecarVersion: 1.13.9-r1-1685522112
```

注意：上述操作完成后会导致业务容器滚动升级。

验证方式

由于流量拦截配置最终会在容器内iptables中生效，执行下述指令查看配置是否生效：

1. 登录到配置流量拦截策略工作负载所在节点，**docker ps**找到对应的pause容器，查看容器id；
2. 查看容器进程**docker inspect <CONTAINER_ID> | grep -i pid**；
3. 进入对应进程namespace：**nsenter -t <PID> -n bash**；
4. 查询iptables：**iptables -t nat -L -n -v**，检查配置的端口、网段拦截策略是否生效；

```
[root@cluster-zsb-52975 ~]# docker inspect d45ad226406e | grep -i pid
    "Pid": 1673903,
    "PidMode": "",
    "PidsLimit": 0,
[root@cluster-zsb-52975 ~]# nsenter -t 1673903 -n bash
[root@cluster-zsb-52975 ~]# iptables -t nat -l -n -v
Chain PREROUTING (policy ACCEPT 36 packets, 2160 bytes)
  pkts bytes target     prot opt in     out     source            destination
  36 2160 ISTIO_INBOUND tcp  --  *      *      0.0.0.0/0         0.0.0.0/0

Chain INPUT (policy ACCEPT 36 packets, 2160 bytes)
  pkts bytes target     prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 43 packets, 3861 bytes)
  pkts bytes target     prot opt in     out     source            destination
  8 480 ISTIO_OUTPUT tcp  --  *      *      0.0.0.0/0         0.0.0.0/0

Chain POSTROUTING (policy ACCEPT 43 packets, 3861 bytes)
  pkts bytes target     prot opt in     out     source            destination

Chain ISTIO_INBOUND (1 references)
  pkts bytes target     prot opt in     out     source            destination
  0 0 RETURN    tcp  --  *      *      0.0.0.0/0         0.0.0.0/0          tcp dpt:15008
  0 0 ISTIO_IN_REDIRECT tcp -- *      *      0.0.0.0/0         0.0.0.0/0          tcp dpt:22
  0 0 ISTIO_IN_REDIRECT tcp -- *      *      0.0.0.0/0         0.0.0.0/0          tcp dpt:23

Chain ISTIO_IN_REDIRECT (4 references)
  pkts bytes target     prot opt in     out     source            destination
  0 0 REDIRECT  tcp  --  *      *      0.0.0.0/0         0.0.0.0/0          redir ports 15006

Chain ISTIO_OUTPUT (1 references)
  pkts bytes target     prot opt in     out     source            destination
  2 120 RETURN   tcp  --  *      *      0.0.0.0/0         0.0.0.0/0          tcp dpt:80
  0 0 RETURN   tcp  --  *      *      0.0.0.0/0         0.0.0.0/0          tcp dpt:81
  0 0 RETURN   tcp  --  *      *      0.0.0.0/0         0.0.0.0/0          tcp dpt:82
  0 0 RETURN   all  --  *      *      127.0.0.0/8       0.0.0.0/0
  0 0 ISTIO_IN_REDIRECT all -- *      *      127.0.0.0/8       1127.0.0.1         ! owner UID match 1337
  0 0 RETURN   all  --  *      *      0.0.0.0/0         0.0.0.0/0         ! owner UID match 1337
  0 360 RETURN all -- *      *      0.0.0.0/0         0.0.0.0/0         ! owner UID match 1337
  0 0 ISTIO_IN_REDIRECT all -- *      *      127.0.0.1         1127.0.0.1         ! owner UID match 1337
  0 0 RETURN   all  --  *      *      0.0.0.0/0         0.0.0.0/0         ! owner UID match 1337
  0 0 RETURN   all  --  *      *      0.0.0.0/0         0.0.0.0/0
  0 0 RETURN   all  --  *      *      0.0.0.0/0         192.168.0.0/24
  0 0 ISTIO_REDIRECT all -- *      *      0.0.0.0/0         172.17.0.0/24
  0 0 RETURN   all  --  *      *      0.0.0.0/0         0.0.0.0/0

Chain ISTIO_REDIRECT (1 references)
  pkts bytes target     prot opt in     out     source            destination
  0 0 REDIRECT  tcp  --  *      *      0.0.0.0/0         0.0.0.0/0          redir ports 15001

labels:
  app: istio-ingressgateway
  replicaCount: 2
  tolerations:
  - effect: NoExecute
    key: istio
    operator: Exists
global:
  defaultPodDisruptionBudget:
    enabled: true
  hub: sae.cn-north-7.myhuaweicloud.com/asm
  imagePullPolicy: IfNotPresent
  logging:
  level: default:info
  meshID: cluster-zsb
  multiCluster:
    clusterName: cluster-zsb
  proxy:
    autoInject: enabled
    excludeIPRanges: 192.168.0.0/24
    excludeOutboundPorts: 80,81,82
    includeIPRanges: 172.17.0.0/24
    includeInboundPorts: 22,23
  runAsGroup: 10000
  runAsUser: 10000
  tag: 1.15.5-r4-20230725201612
  tracer:
    zipkin:
      address: ""
  hormes:
  base:
  image: hormes
  replicaCount: 2
  clusterID: cluster-zsb
  controllers:
  - lazyxds
  - ratelimit
  excludeNamespaces:
  - kube-system
  - monitoring
  - cce-monitor
  - cce
  outboundTrafficPolicyMode: ALLOW_ANY
  podIPRanges: []
  runAsGroup: 10000
  runAsUser: 10000
  pilot:
    autoscaleEnabled: false
  env:
  DEPLOY_TYPE: hub
```

5.5 网关如何配置最大并发流 max_concurrent_streams

步骤1 登录网关所在的集群任意节点执行以下命令，创建资源。

```
cat>"stream-limit-envoyfilter.yaml"<<EOF
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: http2-stream-limit
  namespace: istio-system
spec:
  workloadSelector:
    labels:
      istio: ingressgateway
  configPatches:
  - applyTo: NETWORK_FILTER # http connection manager is a filter in Envoy
    match:
      context: GATEWAY
      listener:
        filterChain:
          filter:
            name: "envoy.filters.network.http_connection_manager"
    patch:
      operation: MERGE
      value:
        typed_config:
          "@type": "type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager"
          http2_protocol_options:
            max_concurrent_streams: 128
EOF
```

说明

max_concurrent_streams即控制网关最大并发流参数，您可以根据需要进行配置。

步骤2 执行 `kubectl apply -f stream-limit-envoyfilter.yaml` 创建envoyfilter。

```
root@ecs-guobaoqing-0054:~# kubectl get envoyfilter -nistio-system
NAME          AGE
http2-stream-limit 8s
```

----结束

6 流量监控

6.1 Jaeger/Zipkin OSC 插件安装指导

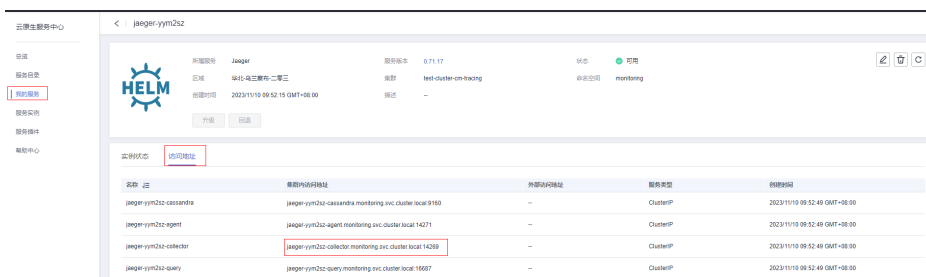
Jaeger/Zipkin OSC插件安装方式一致，下面以Jaeger为例进行安装。

- 步骤1** 进入云原生服务中心OSC控制台，在左侧菜单中选择“服务目录”，搜索“Jaeger”，单击“订阅”。
- 步骤2** 在左侧菜单中选择“我的服务--我的订阅”页面，单击Jaeger服务中的“创建实例”。
- 步骤3** 选择实例的部署场景、区域、容器集群、命名空间，再选择所需要部署的服务实体。勾选左下角“我已知晓”复选框。

说明

命名空间需要选择monitoring命名空间，否则需要自行配置ServiceEntry和WorkloadEntry。

- 步骤4** 单击“下一步：实例参数”，填写实例参数，支持“表单”和“yaml”两种部署模式，建议使用yaml方式进行部署。
- 步骤5** 参数填写完毕后单击“下一步：信息确认”，确认无误，单击“提交”，等待实例创建成功。
- 步骤6** 实例详情中“访问地址”即Jaeger服务接收请求信息的地址和端口。



须知

此处的包含collector关键字的地址和端口即ASM应用服务网格在购买网格的时候“可观测性配置--调用链”选择“第三方Jaeger/Zipkin服务”时需要填入的服务地址和服务端口。

----结束