应用服务网格(ASM)

常见问题

文档版本 01

发布日期 2023-05-30

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

目 录

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，进入对应集群详情页，在“资源 > 命名空间”中查看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

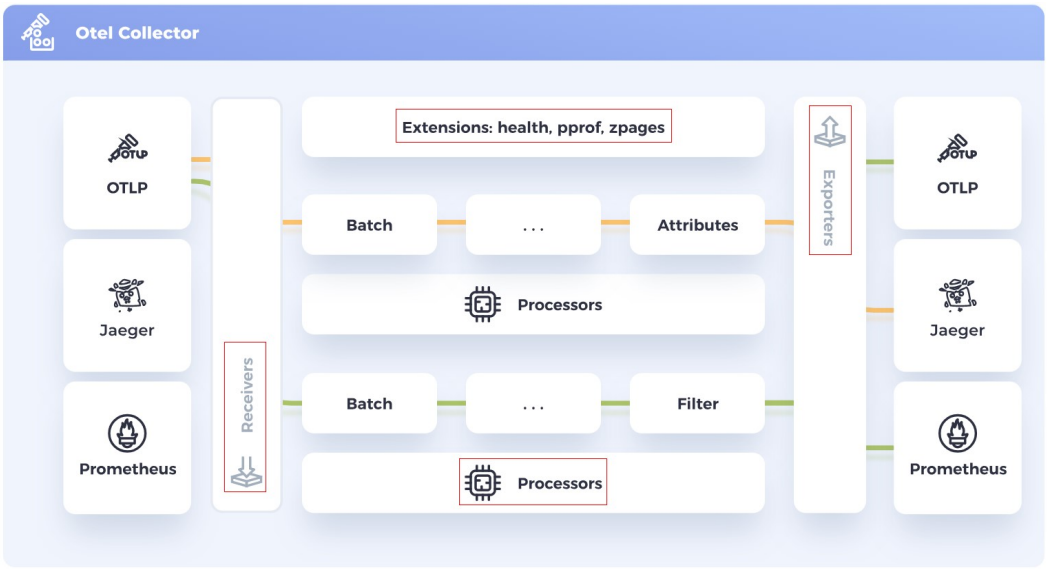
创建网格会自动创建一个otel-collector工作负载。

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@ ~]# 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。

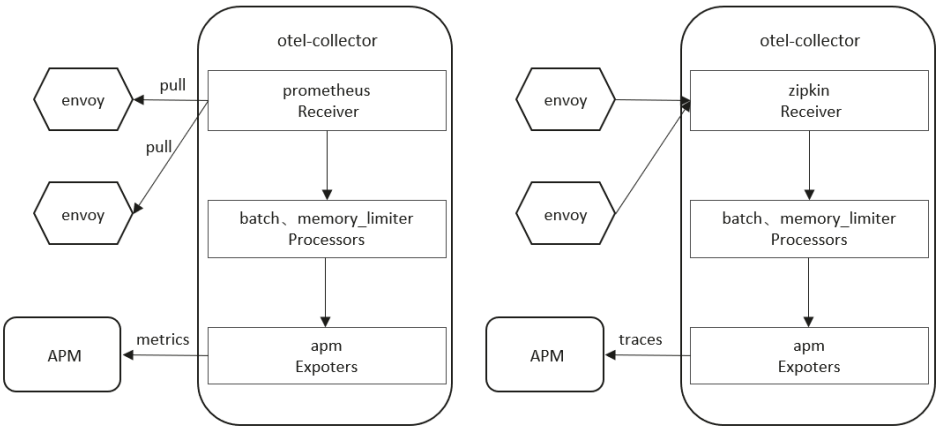
```
extensions:
  health_check:
    endpoint: 127.0.0.1:13133
```

- 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 包周期的独享节点如何退订?

包周期资源退订入口，统一在订单管理页。包周期资源与CCE退订规则相同，需**提交工单**，联系运维人员退订。

2.3 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“资源 > 节点管理”中删除该节点。

2.4 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

2.5 企业版网格添加集群时，选择非扁平网络，为什么查询不到 ELB?

企业版网格添加集群时，如果选择非扁平网络，ASM会为集群创建一个东西向流量网关，需要绑定一个私网负载均衡实例ELB，作为其他集群流量的入口。ASM会查询集群所处VPC下的所有私网ELB（如果ELB绑定了公网IP会被过滤），支持选择共享型和独享型ELB，其中独享型ELB必须包含网络型（TCP/UDP）规格。因此，如果查询不到ELB，可能原因是：

- 未购买ELB实例
- 所购买的ELB实例不在集群所处VPC下
- ELB实例绑定了公网IP
- 独享型ELB未包含网络型（TCP/UDP）规格
- 当前Region暂不支持独享型ELB

说明

ASM支持独享型ELB目前仅在部分Region上线（如“华北-北京四”），其他Region会陆续上线，敬请关注。

关于非扁平网络的详细介绍请参见：**非扁平网络**。

2.6 集群校验报错常见场景及解决方案

为企业版网格添加集群时，系统会自动校验集群是否符合要求，集群校验报错常见场景及解决方案如下所述：

1. 集群需要两个可用资源大于2 vCPUs、4GiB的节点，当前资源不足，无法创建网格

   解决方案：在ECS控制台选择对应的节点进行扩容。

2. 集群容器网段与网格控制面网段冲突

   解决方案：

   - 已购买网格，添加集群场景：重新规划集群容器网段。
   - 购买网格同时添加集群场景：修改网格控制面网段或重新规划集群容器网段。

3. 集群容器网段与集群服务网段冲突

   解决方案：重新规划集群容器网段，确保集群容器网段不与其他待添加集群的服务网段冲突，并且不与网格中已添加集群的服务网段冲突。

4. 集群容器网段与集群VPC网段冲突（containerVPCNetworkOverlapping）

   解决方案：重新规划集群容器网段，确保集群容器网段不与其他待添加集群的VPC网段冲突，并且不与网格中已添加集群的VPC网段冲突。

5. 集群已存在istio-system命名空间

   解决方案：删除已创建的istio-system命名空间。

6. 集群所在VPC已经与其他服务网格建立对等连接

   解决方案：请检查同一个VPC下的集群是否已添加到其他网格中，如果有，需要先移除同一个VPC下已添加到其他网格的所有集群。

7. 集群和网格中已有集群网络类型冲突

   解决方案：请检查待添加集群网络类型，如果是overlay_l2容器隧道网络类型，若网格中已存在集群，则会失败，需要先把网格中已添加集群全部移除；如果是VPC网络，若网格中已有overlay_l2容器隧道网络类型集群，则失败，需要把overlay_l2容器隧道网络类型集群移出网格。

8. 集群服务网段与集群容器网段冲突

   解决方案：重新规划集群服务网段，确保集群服务网段不与其他待添加集群的容器网段冲突，并且不与网格中已添加集群的容器网段冲突。

9. 集群服务网段与网格控制面网段冲突

   解决方案：

   - 已购买网格，添加集群场景：重新规划集群服务网段。
   - 购买网格同时添加集群场景：修改网格控制面网段或重新规划集群服务网段。

10. 集群服务网段与集群服务网段冲突

    解决方案：重新规划集群服务网段，确保集群服务网段不与其他待添加集群的服务网段冲突，并且不与网格中已添加集群的服务网段冲突。

11. 集群服务网段与集群虚拟私有云网段冲突

    解决方案：重新规划集群服务网段，确保集群服务网段不与其他待添加集群的VPC网段冲突，并且不与网格中已添加集群的VPC网段冲突。

12. 集群虚拟私有云网段与集群容器网段冲突

    解决方案：重新规划集群VPC网段，确保集群VPC网段不与其他集群的容器网段冲突。

13. 集群虚拟私有云网段与网格控制面网段冲突

    解决方案：

    - 已购买网格，添加集群场景：重新规划集群VPC网段。
    - 购买网格同时添加集群场景：修改网格控制面网段或重新规划集群VPC网段。

14. 集群虚拟私有云网段与集群VPC网段冲突

    解决方案：重新规划集群VPC网段，确保集群VPC网段不与其他集群的VPC网段冲突。

15. 网格控制面网段与该集群VPC路由表中的路由（xx.xx.x.xxx）冲突，请检查同一个VPC的其他集群是否已添加到其他的网格

    解决方案：

- 已购买网格，添加集群场景：待添加集群的VPC路由表中存在与网格控制面网段冲突的路由，确认是否可以删除该路由，不能删除路由则需要重新规划网格控制面网段购买网格。
- 购买网格同时添加集群场景：修改网格控制面网段。

16. 集群虚拟私有云网段与集群服务网段冲突

    解决方案：重新规划集群VPC网段，确保集群VPC网段不与其他集群的服务网段冲突。

2.7 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              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

2.8 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。

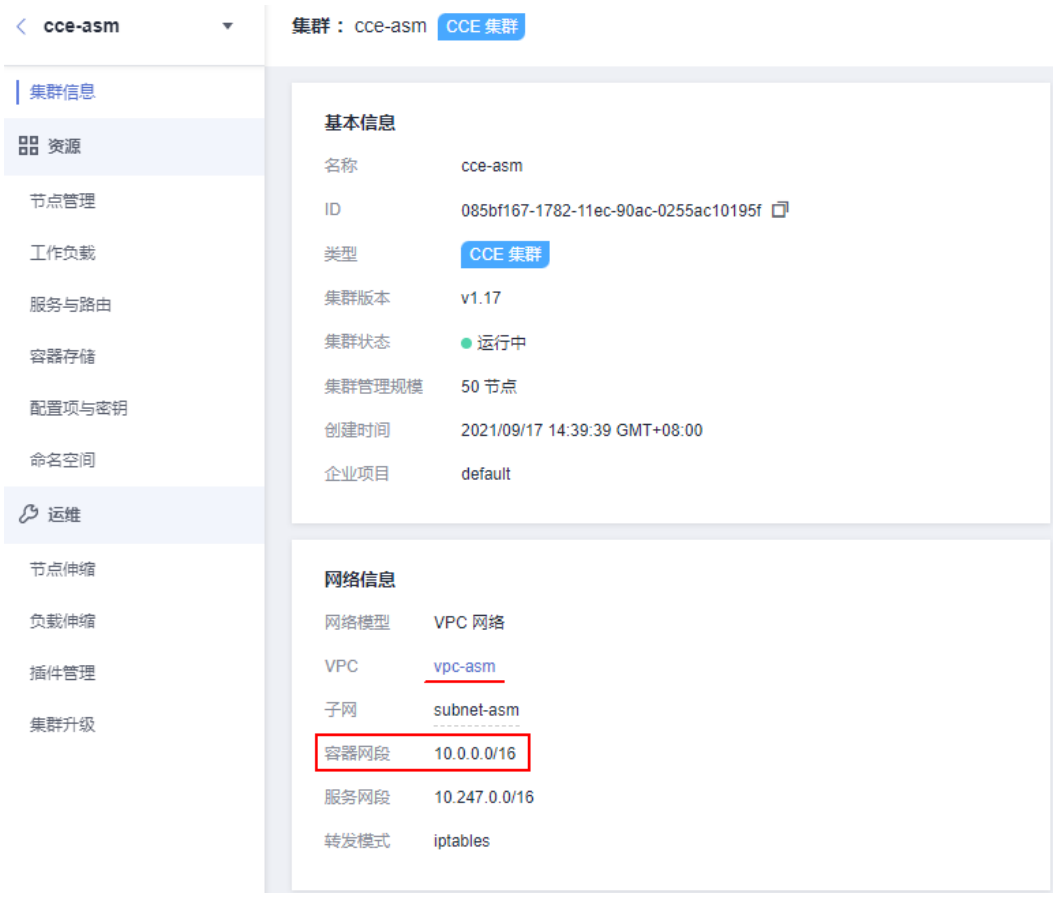
----结束

2.9 如何通过对等连接打通两个集群的 VPC 网络，实现实例跨集群通信?

使用ASM企业版网格多集群特性时，如果两个集群处于不同VPC，可以通过对等连接打通两个集群的网络，从而实现实例的跨集群通信。

准备工作

创建对等连接之前，需要获取两个集群的VPC网段和容器网段，可以在集群详情页查看：



集群的VPC网段在VPC详情页获取：

操作步骤

下文以打通集群A、集群B的网络为例进行介绍。

步骤1 创建对等连接。

1. 登录管理控制台，选择“网络 > 虚拟私有云 VPC”。
2. 在左侧导航栏选择“对等连接”，单击右上角的“创建对等连接”。
3. 填写参数。
   - 名称：自定义
   - 本端VPC：选择集群A的VPC（两个集群的配置顺序不影响功能）
   - 账户：当前账户
   - 对端项目、对端VPC：选择集群B的VPC

图 2-1 创建对等连接

步骤2 添加本端路由。

1. 在对等连接列表，单击**步骤1**中创建的对等连接名称，进入详情页面。
2. 在“本端路由”页签下单击“路由表”，前往路由表页面添加路由。

3. 将对端集群B的VPC网段、容器网段加入本端路由配置中，使得集群A能够访问集群B。下一跳类型选择“对等连接”，下一跳选择**步骤1**中创建的对等连接。

图 2-2 添加本端路由

说明

如果两个集群都是非扁平网络，在路由表中可以只配置VPC网段。关于非扁平网络的介绍，请参见**非扁平网络**。

步骤3 添加对端路由。

1. 在对等连接列表，单击**步骤1**中创建的对等连接名称，进入详情页面。
2. 在“对端路由”页签下单击“路由表”，前往路由表页面添加路由。
3. 将本端集群A的VPC网段、容器网段加入对端路由配置中，使得集群B能够访问集群A。下一跳类型选择“对等连接”，下一跳选择**步骤1**中创建的对等连接。

图 2-3 添加对端路由

说明

如果两个集群都是非扁平网络，在路由表中可以只配置VPC网段。关于非扁平网络的介绍，请参见**非扁平网络**。

步骤4 测试网络是否已连通。

1. 访问对端集群的VPC IP。

   登录集群A的一个节点，访问集群B的内网apiserver地址。

   ```
   curl https://192.168.0.180:5443 -ik
   ```

   如果对端长时间没有回复，说明网络存在问题，需要重新检查配置。

   集群B访问集群A的验证方法相同。

2. 访问对端集群的容器IP（非扁平网络跳过）

   本端和对端集群必须是“容器对接ENI”网络模型。如果是CCE集群，网络模型需要选择“VPC网络”；如果是CCE Turbo集群，网络模型需要是“云原生网络2.0”。

   登录集群A的一个节点，访问集群B的某个Pod：

   ```
   curl http://PodIP:Port
   ```

   如果对端长时间没有回复，说明网络存在问题，需要重新检查配置。

   集群B访问集群A的验证方法相同。

----结束

2.10 服务跨集群通信时网络不通，如何解决?

操作场景

企业版网格可以管理多个集群（CCE集群和CCE Turbo集群均可），服务在跨集群通信时，如果网络访问不通，可能是因为未放通安全组规则导致的，需要按照本文指导配置安全组规则，有如下两种场景：

- CCE集群服务访问CCE集群服务，需要为集群的Node安全组入方向放通对方集群的容器网段。
- CCE集群服务访问CCE Turbo集群服务，需要为Turbo集群的ENI安全组入方向放通CCE集群的容器网段。

CCE Turbo集群服务访问CCE集群服务，以及CCE Turbo集群服务之间互访，均无需额外配置安全组规则。

说明

如果两个集群不在一个VPC内，需要先通过对等连接打通网络，具体操作请参见**如何通过对等连接打通两个集群的VPC网络，实现实例跨集群通信?**。

场景一：CCE 集群服务访问 CCE 集群服务

为方便描述，假设两个集群名称为ccecluster01、ccecluster02。

步骤1 登录CCE控制台，在集群信息页面获取ccecluster01、ccecluster02集群的容器网段。

步骤2 进入VPC控制台，在左侧导航栏选择“访问控制 > 安全组”，搜索框输入集群名称“ccecluster01”。

步骤3 选择**{集群名}-cce-node-{随机ID}**的安全组，单击操作列的“配置规则”，添加入方向规则。

- 协议端口：选择“基本协议/全部协议”。
- 源地址：填写**步骤1**中获取的ccecluster02集群的容器网段。

步骤4 按照上述方法，为ccecluster02集群的Node安全组入方向放通ccecluster01集群的容器网段。

----结束

场景二：CCE 集群服务访问 CCE Turbo 集群服务

步骤1 登录CCE控制台，单击CCE集群名称，在集群信息页面获取CCE集群的容器网段。

步骤2 进入VPC控制台，在左侧导航栏选择“访问控制 > 安全组”，搜索框输入Turbo集群名称。

步骤3 选择**{集群名}-cce-eni-{随机ID}**的安全组，单击操作列的“配置规则”，添加入方向规则。

- 协议端口：选择“基本协议/全部协议”。
- 源地址：填写**步骤1**中获取的CCE集群的容器网段。

----结束

结果验证

以场景二为例，在CCE集群中访问Turbo集群中的服务（假设为nginx-turbo），访问成功的回显示例如下：

图 2-4 访问成功示例

```
root@tomcat-6bf98676ff-xzn6d:/usr/local/tomcat# curl http://nginx-turbo.default.svc:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@tomcat-6bf98676ff-xzn6d:/usr/local/tomcat#
```

2.11 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的开关来解决了该问题，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-crital
    multiCluster:
      clusterName: test-yyl-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               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-yyl-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
```

2.12 设置 fsgroup，导致业务容器挂载文件属组被修改

问题描述

业务pod注入sidecar时设置fsgroup为1337，导致业务容器挂载文件属组被改成1337。

原因分析

因为k8s 版本bug：

https://github.com/kubernetes/kubernetes/issues/57923

https://github.com/istio/istio/pull/27367

在1.8.6-r2之前的版本会在sidecar注入时自动设置fsgroup为1337（此设置会导致挂载进业务容器的文件属组被改为1337）

解决方法

k8s 1.19以上版本解决了该问题，因此网格1.8.6-r2以上版本，如果集群为1.19以及以上版本，ASM会自动设置**EnableLegacyFSGroupInjection** 为**false**， 该配置控制

sidecar注入时不设置fsgroup为1337，此修改会修正业务容器挂载文件属组被设置为1337的错误做法。若业务前期进行了对应适配，则需要改正回来。

2.13 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（ metchLabels ）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。

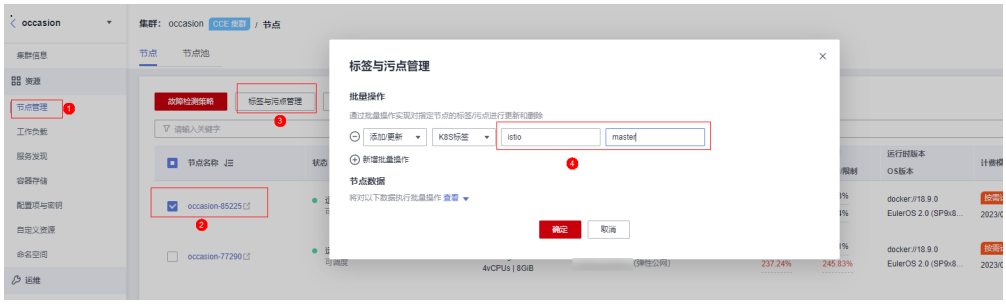
   解决办法：网格支持的版本如下：

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 创建服务网关时，提示 500 错误

问题描述

一键创建体验应用Bookinfo时，提示“创建对外访问方式失败”。

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gateway接口全部返回500，查看返回内容提示如下信息：

```
IP is not the same with LoadBalancerIP
```

原因分析

istio-system命名空间下有gatewayservice残留。残留原因是一键删除模板实例前没有删除已添加的对外访问配置。

解决方法

istio-system命名空间下残留gatewayservice，需要删除该service。

kubectl delete svc *<svc-name>* **-n namespace**

其中，*<svc-name>*为service的名称。

3.5 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

3.6 如何解决应用数据获取失败的问题?

问题描述

服务添加完成后，在“服务列表”中，查看不到已创建的服务，页面提示“应用数据获取失败”。

排查思路

登录ASM控制台，按“F12”，切换到Network页签查看接口，接口全部返回200。查看Console输出存在如下报错：

```
TypeError: Cannot read property 'slice' of undefined
```

原因分析

存在端口为空的服务。

解决方法

步骤1 查看服务端口。

kubectl get svc --all-namespaces

步骤2 给Ports为空的服务添加端口。

----结束

3.7 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

1. 确认使用ASM1.15.5-r3及以上版本创建网格。
2. 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：

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'
...
```

3. 了解Job/CronJob类型使用的约束，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。方法可参考**如何使用Istio调用链埋点**。

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。

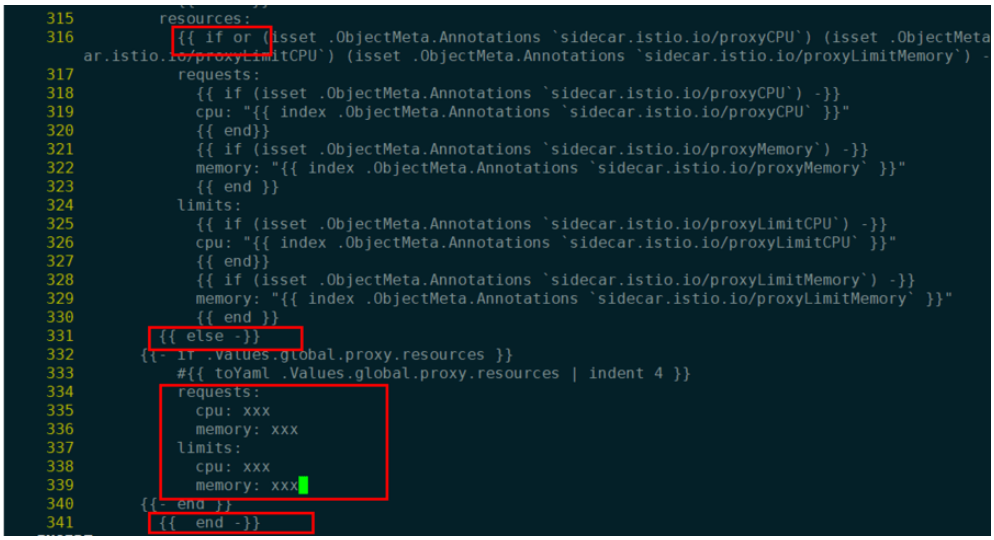
```
resources:
  limits:
    cpu: "2"
    memory: 512Mi
  requests:
    cpu: "1"
    memory: 512Mi
```

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



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

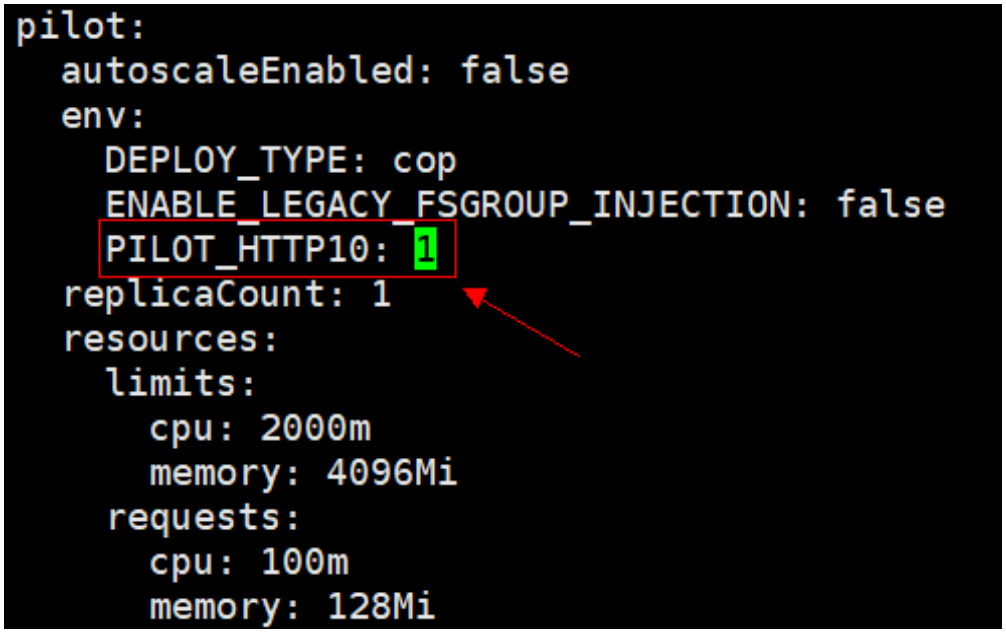
Istio 默认不支持 HTTP/1.0。

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。



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：

```
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：

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

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：

```
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;

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 Pod 刚刚启动后，为什么不能立即看到流量监控数据?

1. 请确保集群已开通APM。
2. 流量监控对采集到的数据进行了聚合处理，需积累一分钟才能看到数据。

6.2 总览页面上的时延数据为什么不准确?

总览页面上的时延数据是显示您账户下全部集群的全部组件的topN数据，且是近一分钟的数据。所以请确保您的组件在近1分钟内，有访问流量产生。

6.3 流量监控拓扑图中为何找不到我的组件?

1. 请选择网格、集群及命名空间后进行观察。
2. 请检查集群中是否正确安装ICAgent采集器。
3. 请检查该组件是否已加入服务网格。

6.4 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。

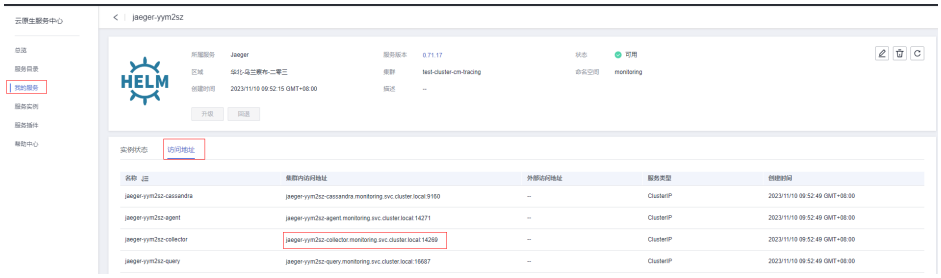
说明

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