服务网格

快速入门

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

文档目录

- 快速入门
  - 概述
  - 部署 Demo 应用
  - 应用配置
    - 配置公网访问
  - 流量控制
    - 多版本路由
    - 灰度发布
  - 服务高可用
    - 故障注入测试
    - 服务超时配置
    - 会话保持
    - 使用连接池限制并发
  - 多集群容灾多活
    - 服务跨集群容灾
    - 地域感知
    - 就近接入
  - 安全加固
    - 认证
    - 授权

版权所有：腾讯云计算（北京）有限责任公司

快速入门

概述

最近更新时间：2023-12-26 10:42:23

快速入门将引导您通过部署和管理一个 demo 应用快速熟悉 TCM 的常见操作，包括服务高可用配置、流量管理、多集群容灾多活、安全加固等，您可以在快速入门中选择相应的章节查看。

前提条件

已创建服务网格实例，如果您还没有网格实例，请参考 创建网格。
已部署 TCM 提供的 demo 应用，如果您还未部署，请参考 部署 Demo 应用。

操作步骤

部署完 demo 应用后，可以参考以下步骤快速了解网格的常见功能。
配置公网访问
多版本路由
灰度发布
故障注入测试
服务超时配置
会话保持
使用连接池限制并发
服务跨集群容灾
地域感知
就近接入
认证
授权
如部署时遇到问题，您可以提交工单 来寻求帮助。

部署 Demo 应用

最近更新时间：2023-12-26 10:42:53

Demo 应用概览

Demo 应用是一个电商网站，基于 Istio 社区的官方样例 bookinfo 改造，由 6 个服务组成：

frontend：网站前端，调用 user、product、cart、order 服务。

product：商品服务，提供商品信息。product 包含两个版本，版本一没有顶部广告 banner；版本二有顶部广告 banner。

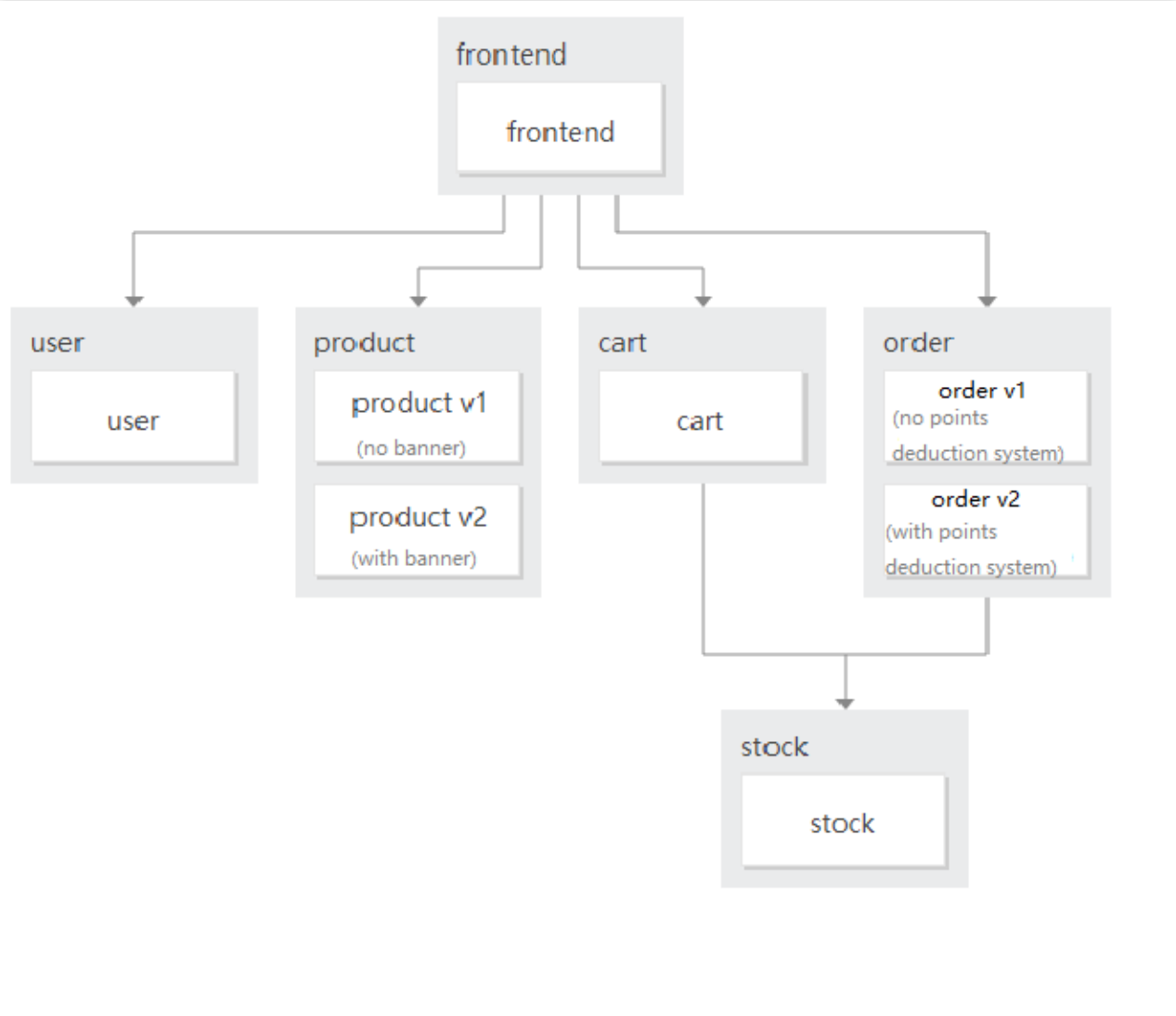
user：用户登录服务，提供登录功能。

cart：购物车服务，提供添加、查看购物车功能，调用库存服务提供库存告警功能，需要登录才可以下单。

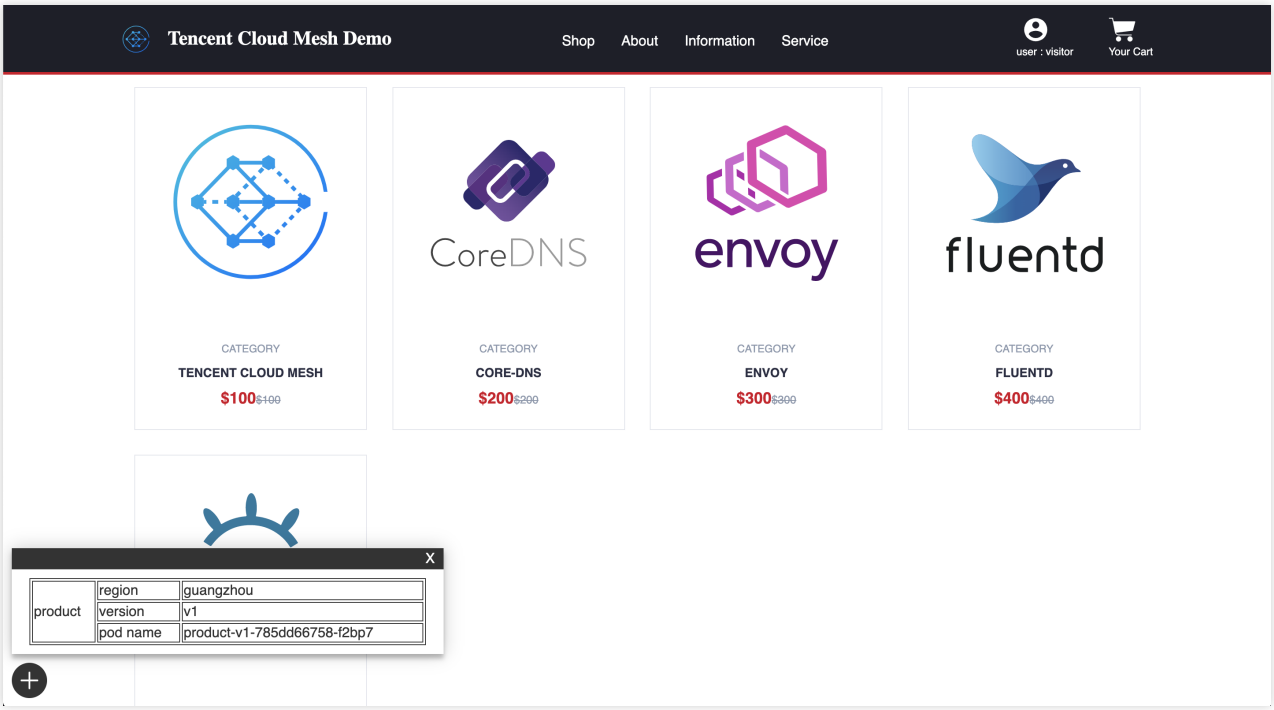
order：订单结算服务，登录后点击 checkout 后可发起结算，结算时需要调用 stock 库存服务查询库存情况，库存不足会下单失败。order 包含两个版本，版本一无积分抵扣运费的功能，版本二有积分抵扣运费的功能。

stock：库存服务，为 order 购物车服务的库存告警功能和 order 订单结算服务的库存查询提供库存信息。

Demo 应用架构

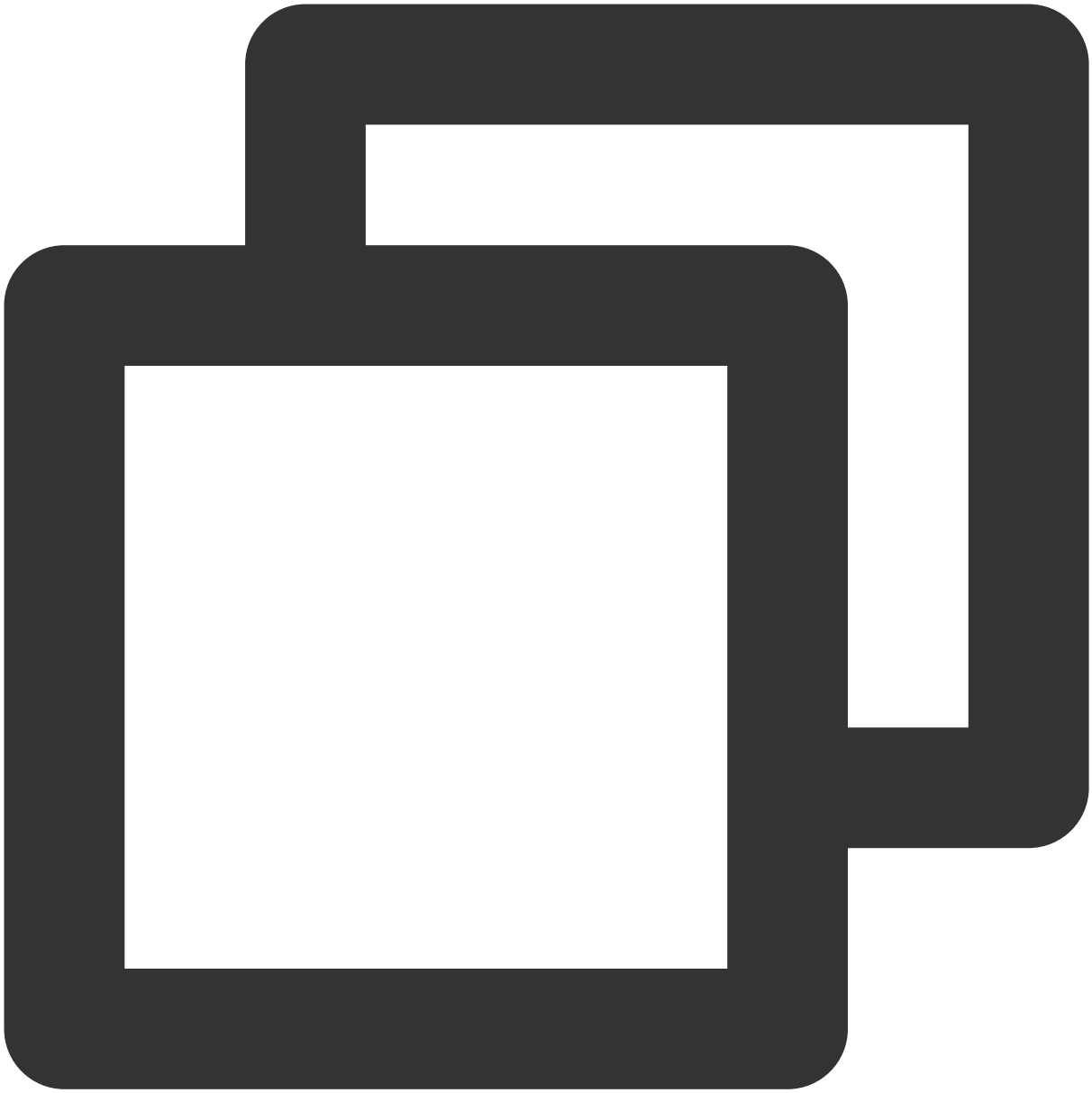

frontend
frontend
user
user
product
product v1
(no banner)
product v2
(with banner)
cart
cart
order
order v1
(no points
deduction system)
order v2
(with points
deduction system)
stock
stock

Demo 应用首页



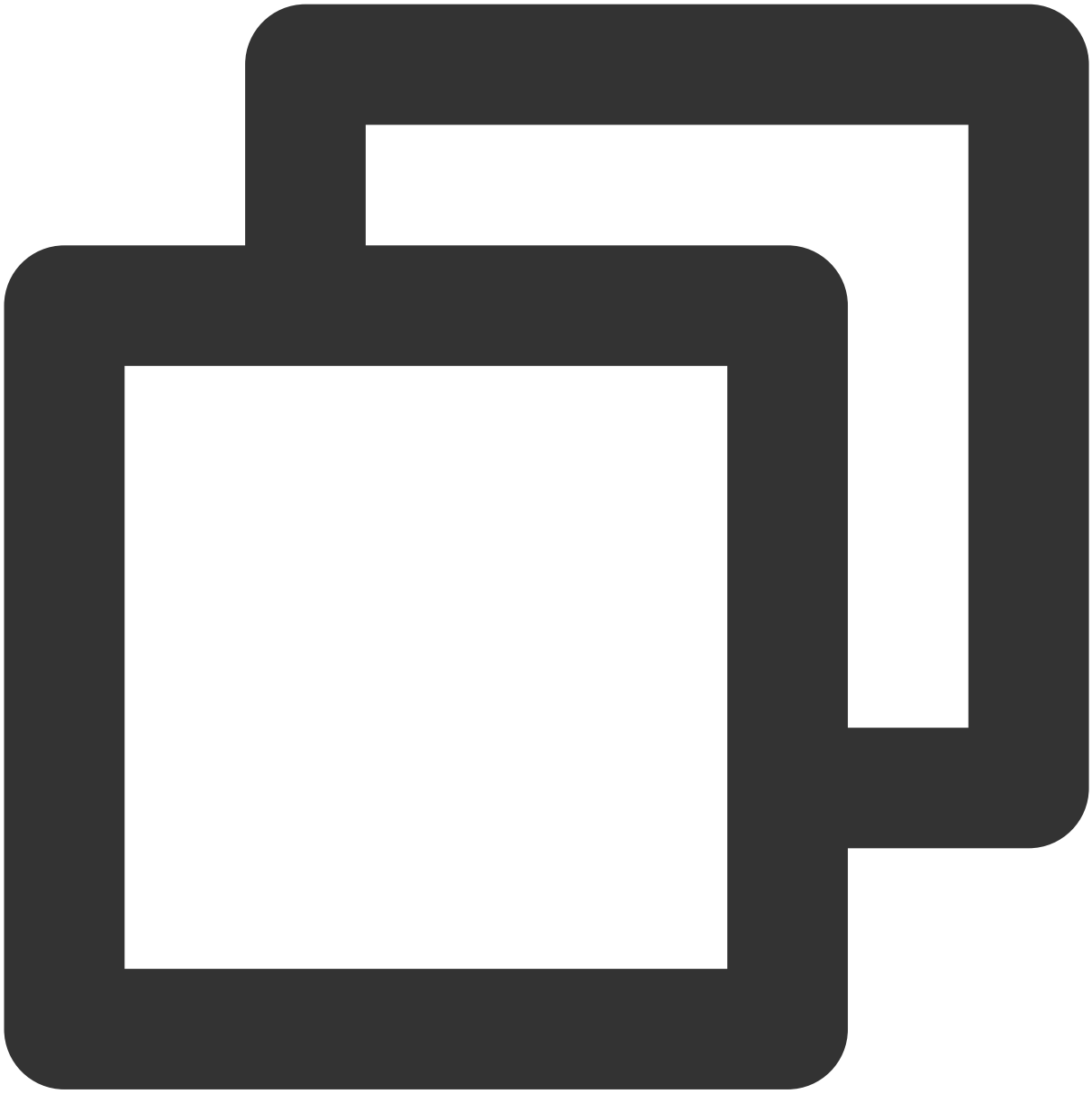
安装 Demo 应用

您可以在 TCM Demo 仓库 中获取 Demo 应用，由于 TCM 的 sidecar 自动注入需要标记 Istio 版本，您需要选择与您 Istio 版本一致的分支，或直接修改 master 分支，路径 mesh-demo/yamls/step01-apps-zone-a.yaml 中 base namespace 中的版本 label：



```
apiVersion: v1
kind: Namespace
metadata:
  name: base
  labels:
    istio.io/rev: 1-10-3
spec:
  finalizers:
    - kubernetes
```

例如，您的 istio 版本为 1.8.1，则需要将 `istio.io/rev: 1-10-3` 修改为 `istio.io/rev: 1-8-1`，否则 sidecar 注入将会失败。
使用如下命令可快速部署 Demo 应用：



```
kubectl apply -f yamls/step01-apps-zone-a.yaml
```

您也可以前往 容器服务控制台，在**集群详情页 > 工作负载 > Deployment** 中，使用 **YAML 创建资源**，复制上述 yaml 内容，一键创建 Demo 应用相关资源。

应用配置
配置公网访问

最近更新时间：2023-12-26 10:43:35

体验环境创建完成后，网站所有服务已经部署至广州的集群（product 服务和 order 服务只部署了 v1 版本），并且已自动注入了 envoy sidecar 接管服务流量，istio-ingressgateway 已经创建，但未配置监听器规则以及路由规则以放通网站 frontend 服务至公网。

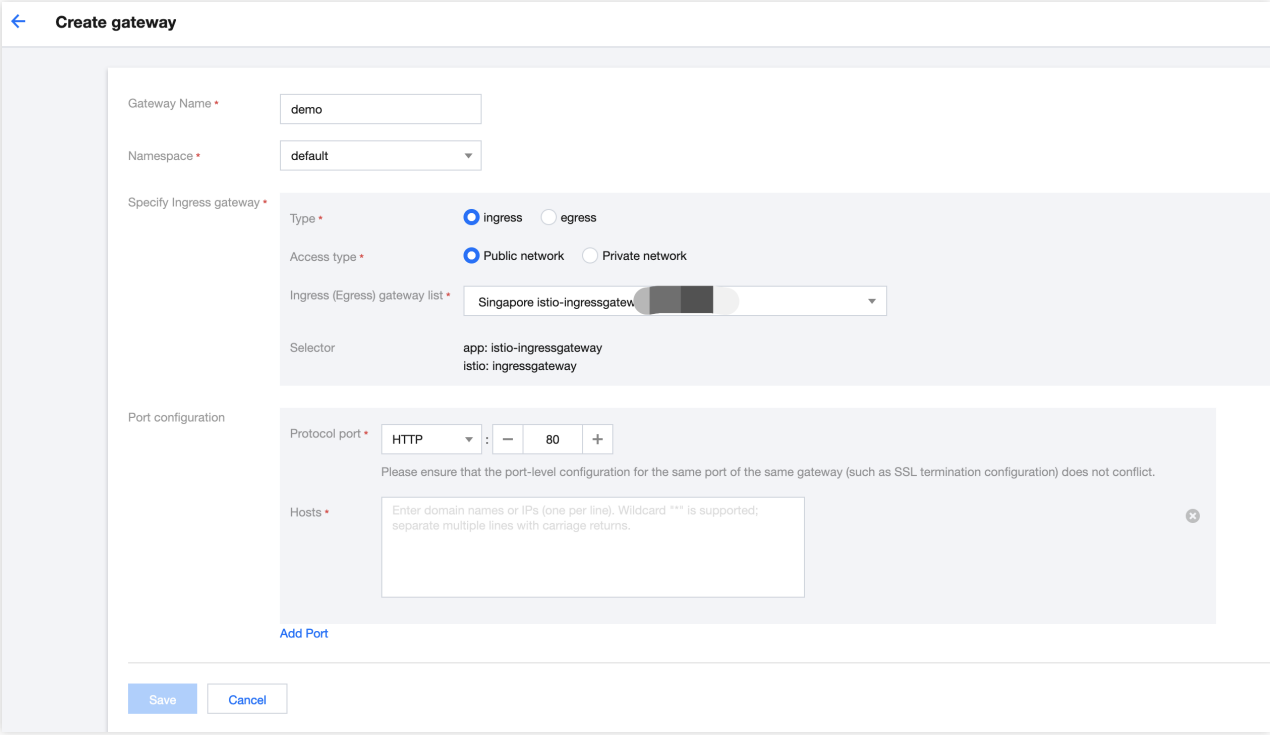
1. 创建 Gateway 配置监听规则

首先需要创建 Gateway 资源，配置 istio-ingressgateway 的监听器规则，端口为 80，协议为 http。用户只需要配置 Gateway 规则，TCM 后台会自动实现 istio-ingressgateway 相关的 pod、service 和绑定的负载均衡器 CLB 的配置同步。

通过控制台

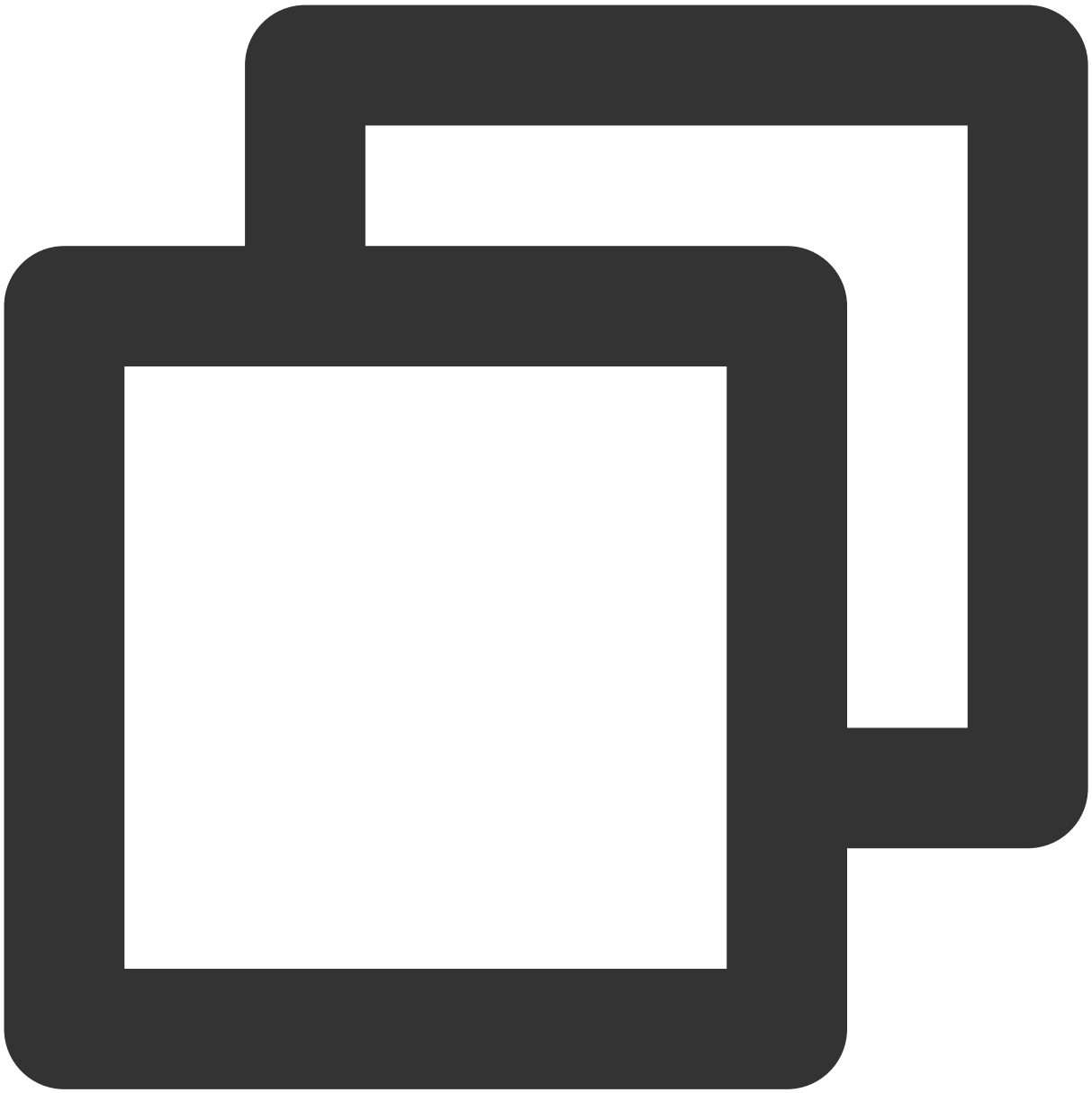
通过 kubectl

1. 登录 服务网格控制台。
2. 单击服务网格 ID，进入已创建的服务网格管理页面。
3. 在 **Gateway** 中单击**新建**。
4. 在**新建Gateway** 中，设置 Gateway 参数。关键参数信息如下图所示：



5. 单击**保存**，完成创建。

通过 kubectl 提交 YAML 至**主集群**完成配置：



```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: frontend-gw
  namespace: base
spec:
  servers:
    - port:
        number: 80
```

```
        name: http
        protocol: HTTP
      hosts:
        - '*'
  selector:
    app: istio-ingressgateway
    istio: ingressgateway
```

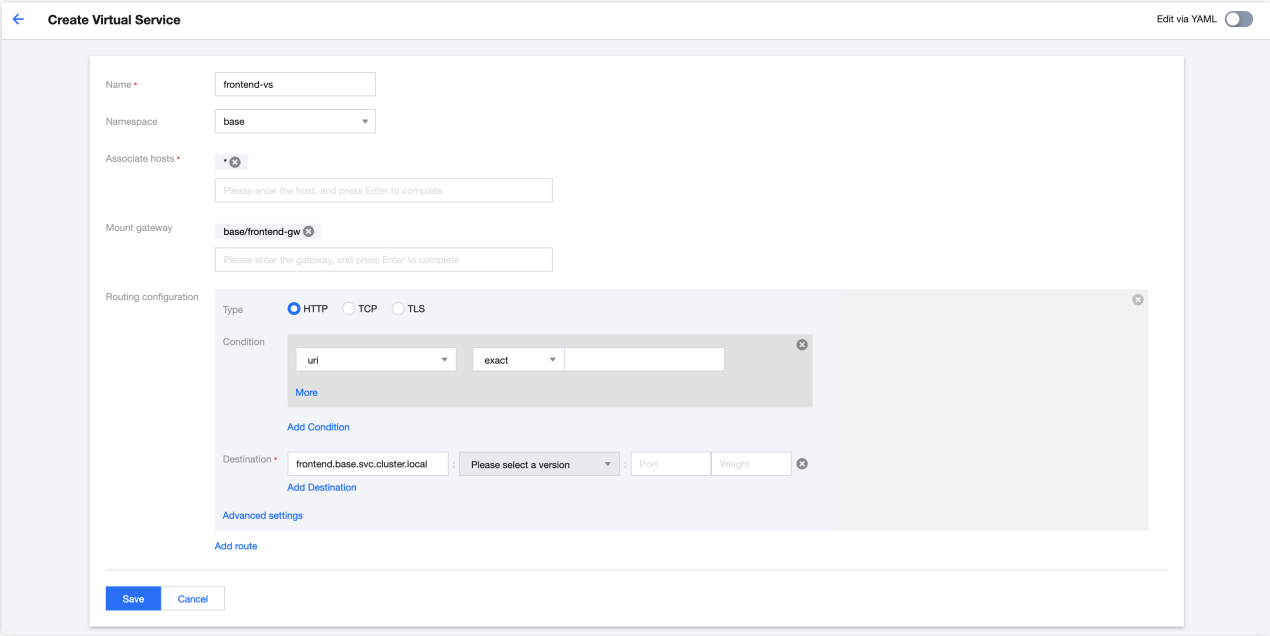
2. 配置路由规则

监听器规则配置完成后，还需要通过 Virtual Service 资源配置路由规则，将来自 istio-ingressgateway 的流量路由至 frontend 服务。

通过控制台

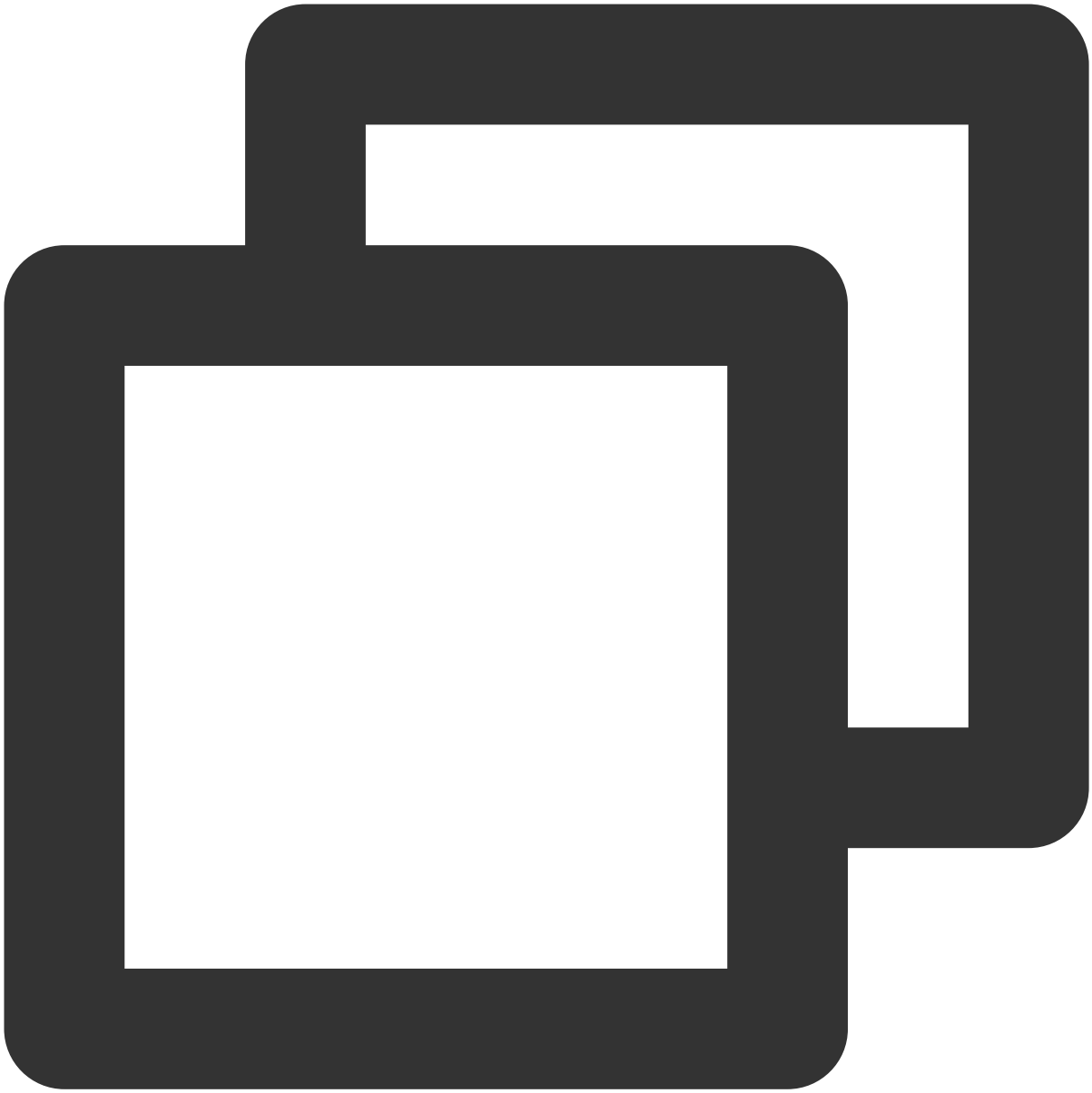
通过 kubectl

1. 登录 服务网格控制台。
2. 单击服务网格 ID，进入已创建的服务网格管理页面。
3. 在 **Virtual Service** 中单击**新建**。
4. 在**新建Virtual Service** 中，设置参数。关键参数信息如下图所示：



5. 单击**保存**，完成创建。

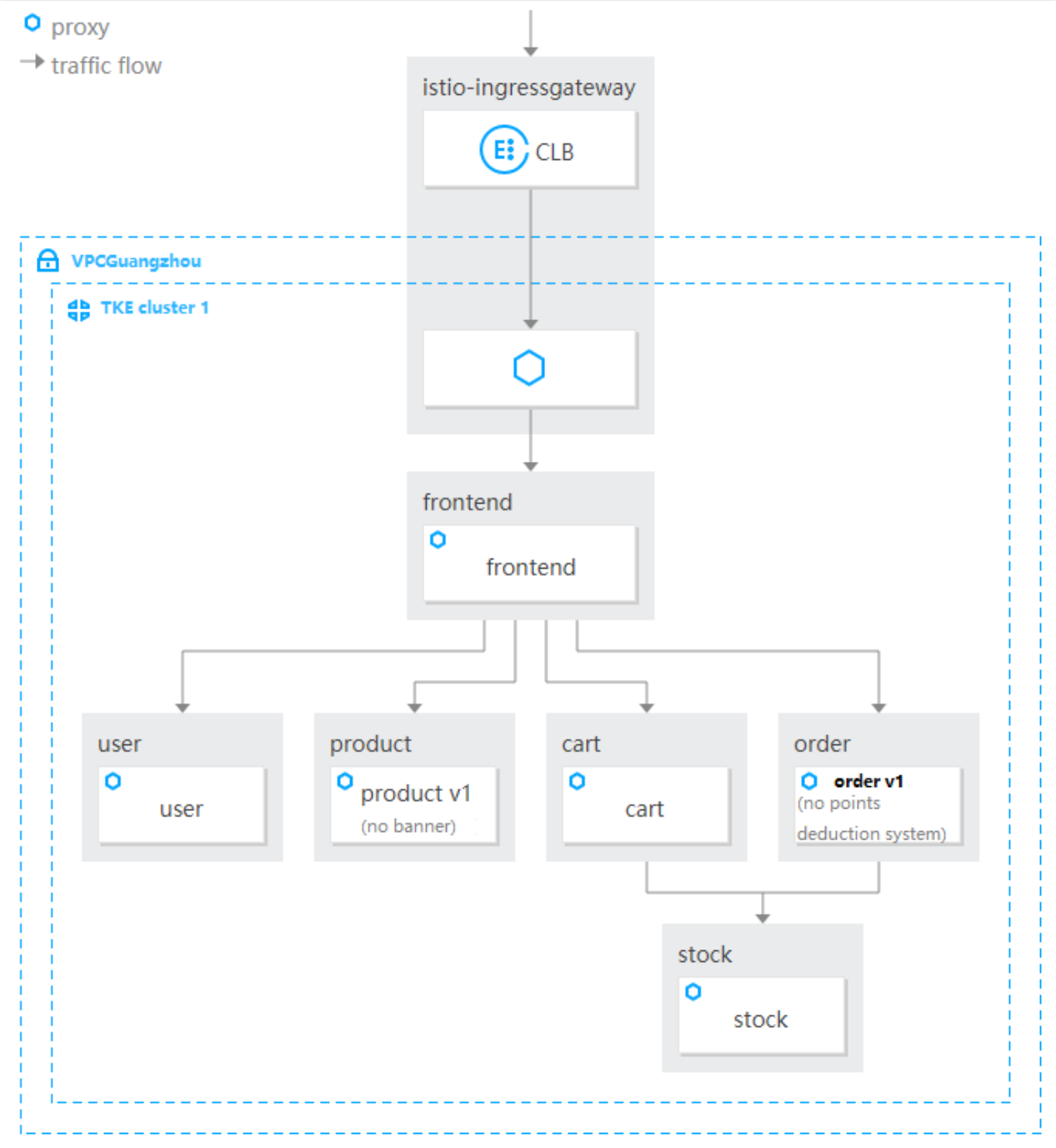
通过 kubectl 提交 YAML 至**主集群**完成配置：



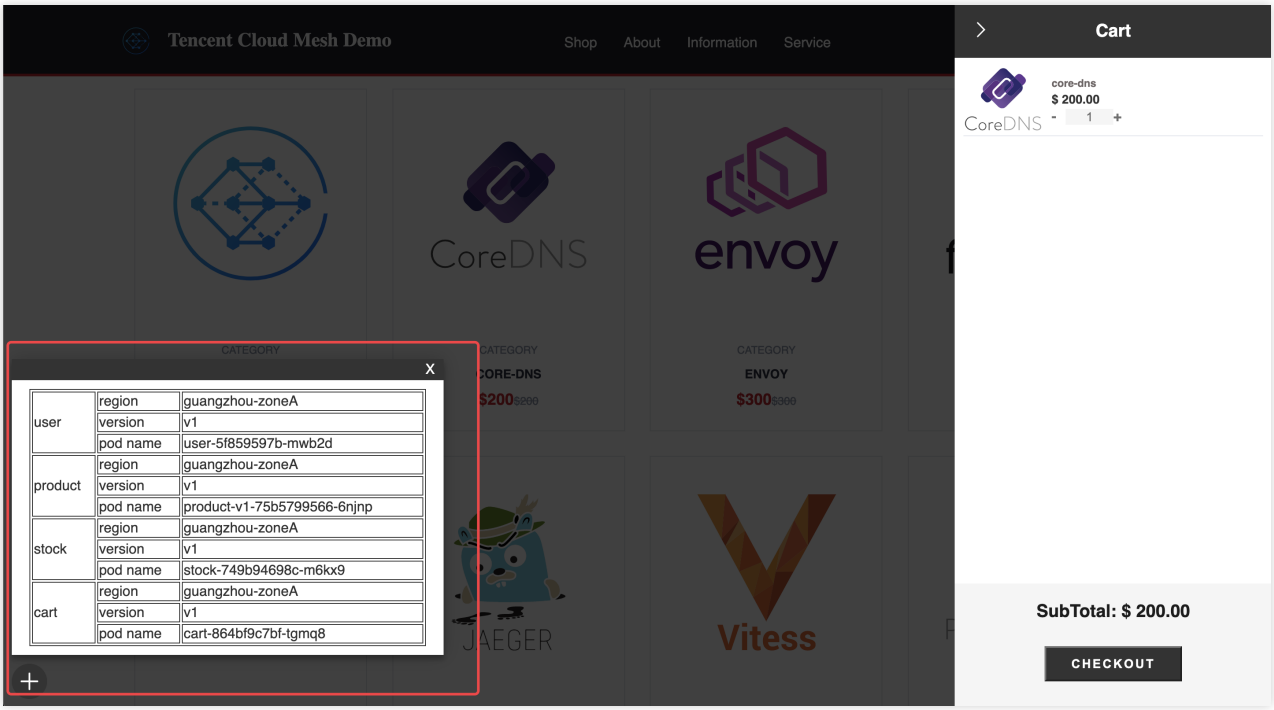
```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: frontend-vs
  namespace: base
spec:
  hosts:
    - '*'
  gateways:
    - base/frontend-gw
  http:
```

```
    - route:
      - destination:
          host: frontend.base.svc.cluster.local
```

配置完成后，通过 istio-ingressgateway 的公网 IP 地址即可访问到 Demo 网站，当前部署的网站的结构如下图所示：



单击链接访问网站后，可登录（1-5 均可登录，其中 1-3 为会员，4-5 为非会员）、添加购物车，下单，以产生调用完所有部署的服务的请求，网站界面右下角的悬浮窗展示了前端服务当前调用服务的名称、地域、版本、pod name 信息。悬浮窗信息展示如下图所示：

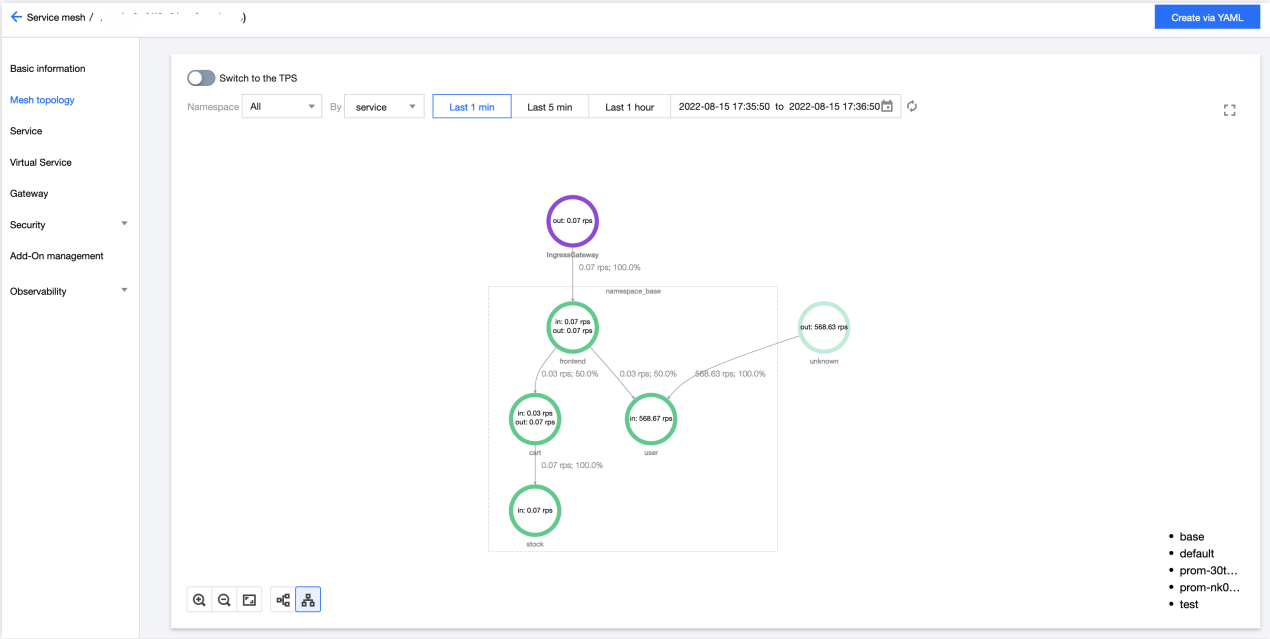


有流量数据后，可单击网络拓扑 tab 查看网格内网络流量的拓扑图。单击服务 tab，服务详情页面可查看到请求的调用链，及可查看调用 stock 服务的完整链路及每层调用的详细信息。

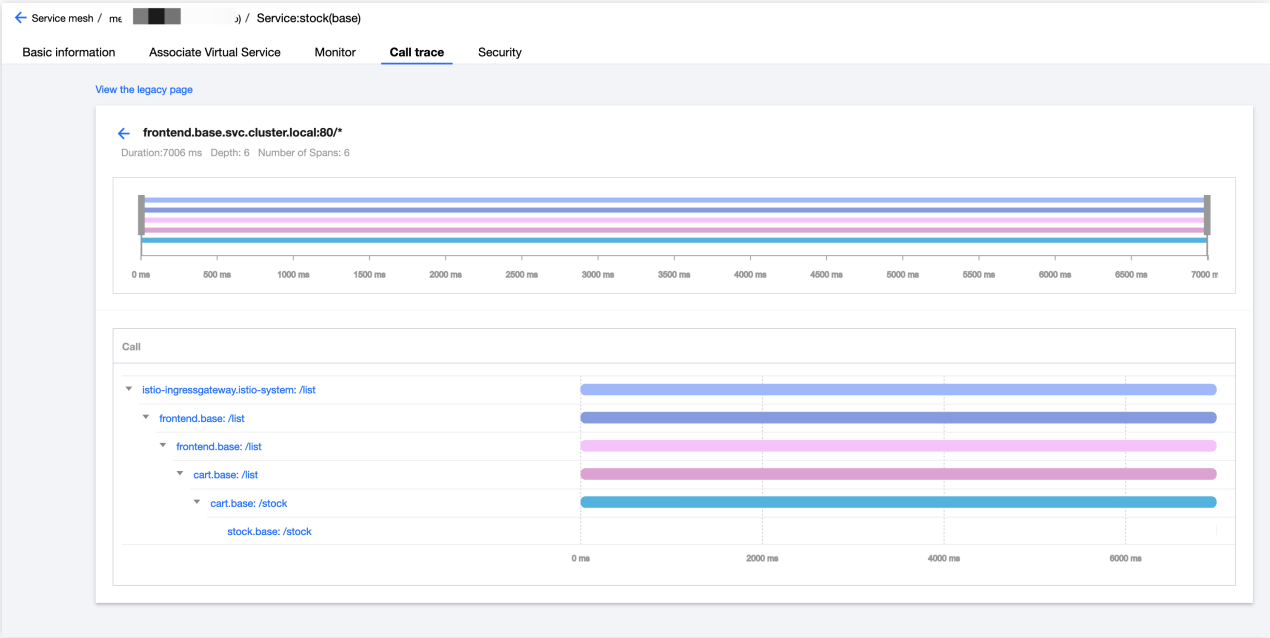
业务规模增加，服务数量较多时，几乎每个前端请求都会形成复杂的调用链路，此时需要能够：在复杂的链路中快速定位和分析问题，判定故障影响范围；或需要梳理服务的调用依赖关系，判断其合理性；或分析链路的请求耗时等性能，做串并行分析的调用逻辑优化等。

通过全链路跟踪系统，可描绘整个网格的流量特征，帮助开发者进行链路分析。

网格拓扑如下图所示：


Service mesh /
Create via YAML
Basic information
Mesh topology
Service
Virtual Service
Gateway
Security
Add-On management
Observability
Switch to the TPS
Namespace
All
By
service
Last 1 min
Last 5 min
Last 1 hour
2022-08-15 17:35:50 to 2022-08-15 17:36:50
out: 0.07 rps
IngressGateway
0.07 rps; 100.0%
namespace_base
in: 0.07 rps
out: 0.07 rps
frontend
out: 568.63 rps
unknown
0.03 rps; 50.0%
0.03 rps; 50.0%
568.63 rps; 100.0%
in: 0.03 rps
out: 0.07 rps
cart
in: 568.67 rps
user
0.07 rps; 100.0%
in: 0.07 rps
stock
base
default
prom-30t...
prom-nk0...
test

链路追踪如下图所示：

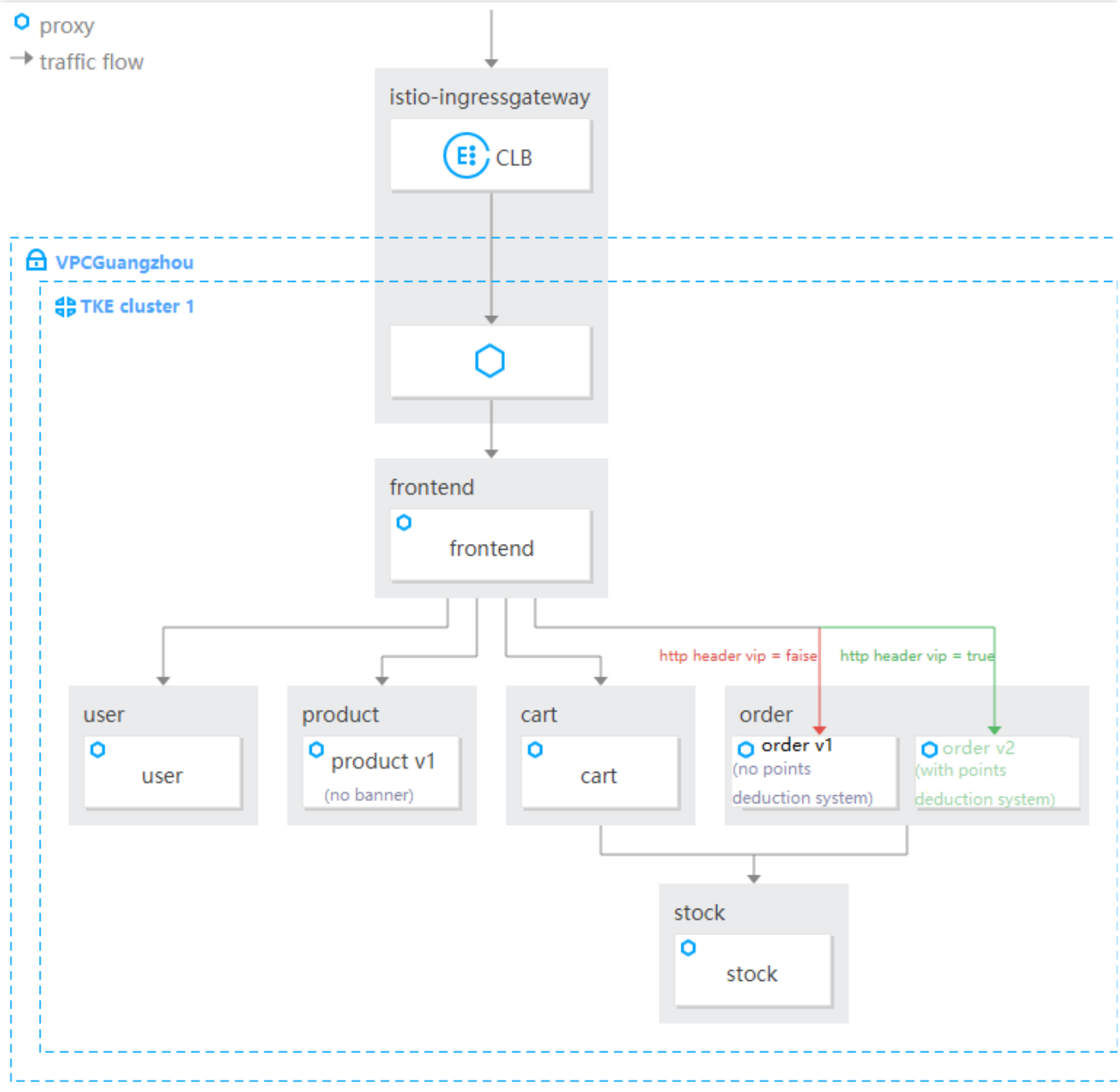

Service mesh /
Service:stock(base)
Basic information
Associate Virtual Service
Monitor
Call trace
Security
View the legacy page
frontend.base.svc.cluster.local:80/*
Duration:7006 ms Depth: 6 Number of Spans: 6
0 ms
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000 ms
4500 ms
5000 ms
5500 ms
6000 ms
6500 ms
Call
istio-ingressgateway.istio-system: /list
frontend.base: /list
frontend.base: /list
cart.base: /list
cart.base: /stock
stock.base: /stock
0 ms
2000 ms
4000 ms
6000 ms

流量控制
多版本路由

最近更新时间：2023-12-26 10:44:06

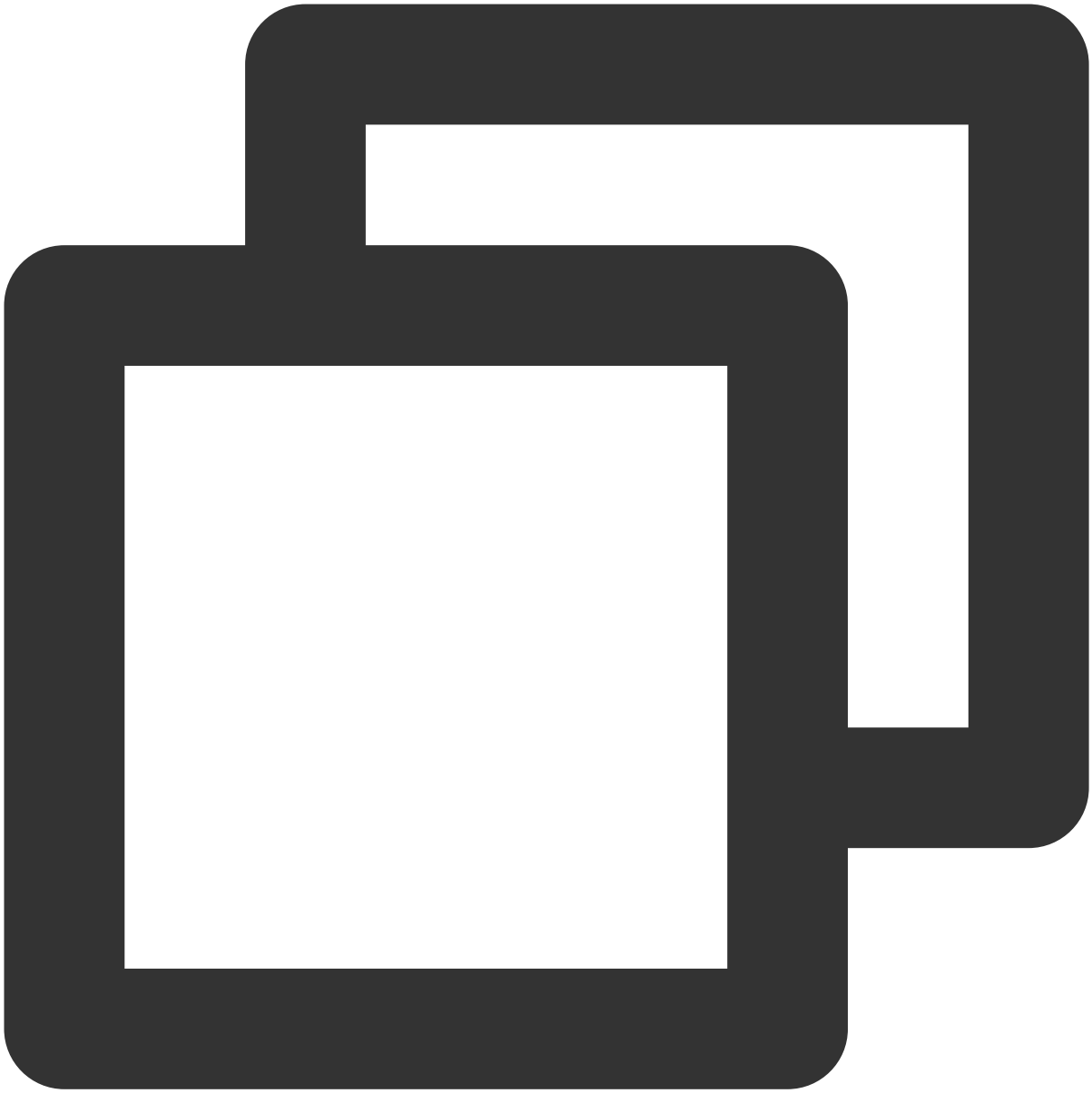
操作场景

网站计划推出会员积分抵扣运费的活动以发展会员。电商网站策划了会员积分抵扣订单金额的新功能，当前部署的 order 服务由 v1 deployment 提供，没有运费抵扣的功能；网站新开发了 order 服务的 v2 版本，有积分抵扣运费的功能。网站希望可以于请求的 header 中是否会员的 cookie 信息进行路由，会员路由至 order v2（有运费抵扣功能），非会员路由至 order v1（无运费抵扣功能）。
服务多版本路由概览图如下所示：



操作步骤

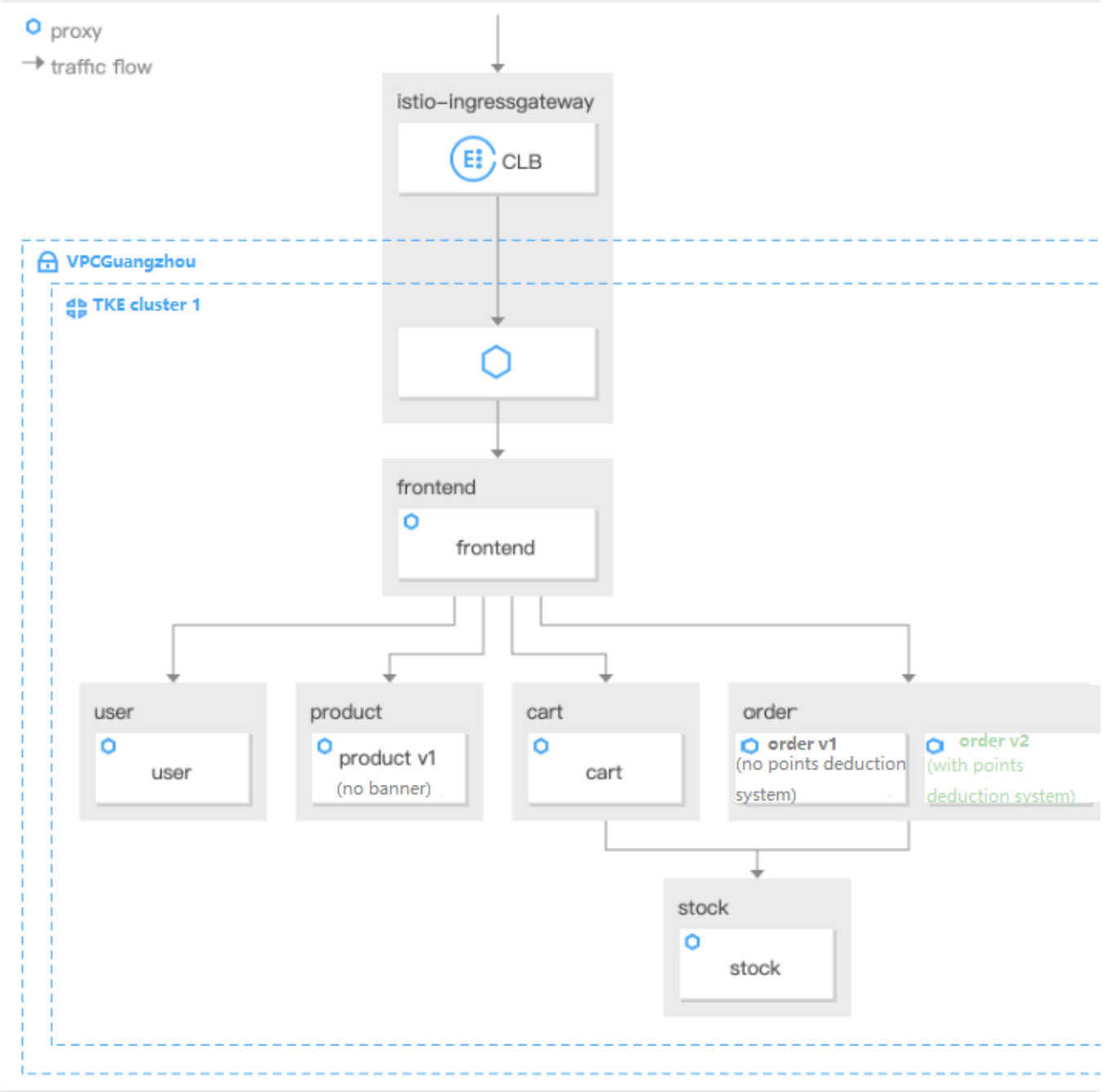
提交以下 yaml 文件至主集群，部署 order v2 至集群。



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: order-v2
  namespace: base
  labels:
    app: order
    version: v2
spec:
  replicas: 1
  selector:
```

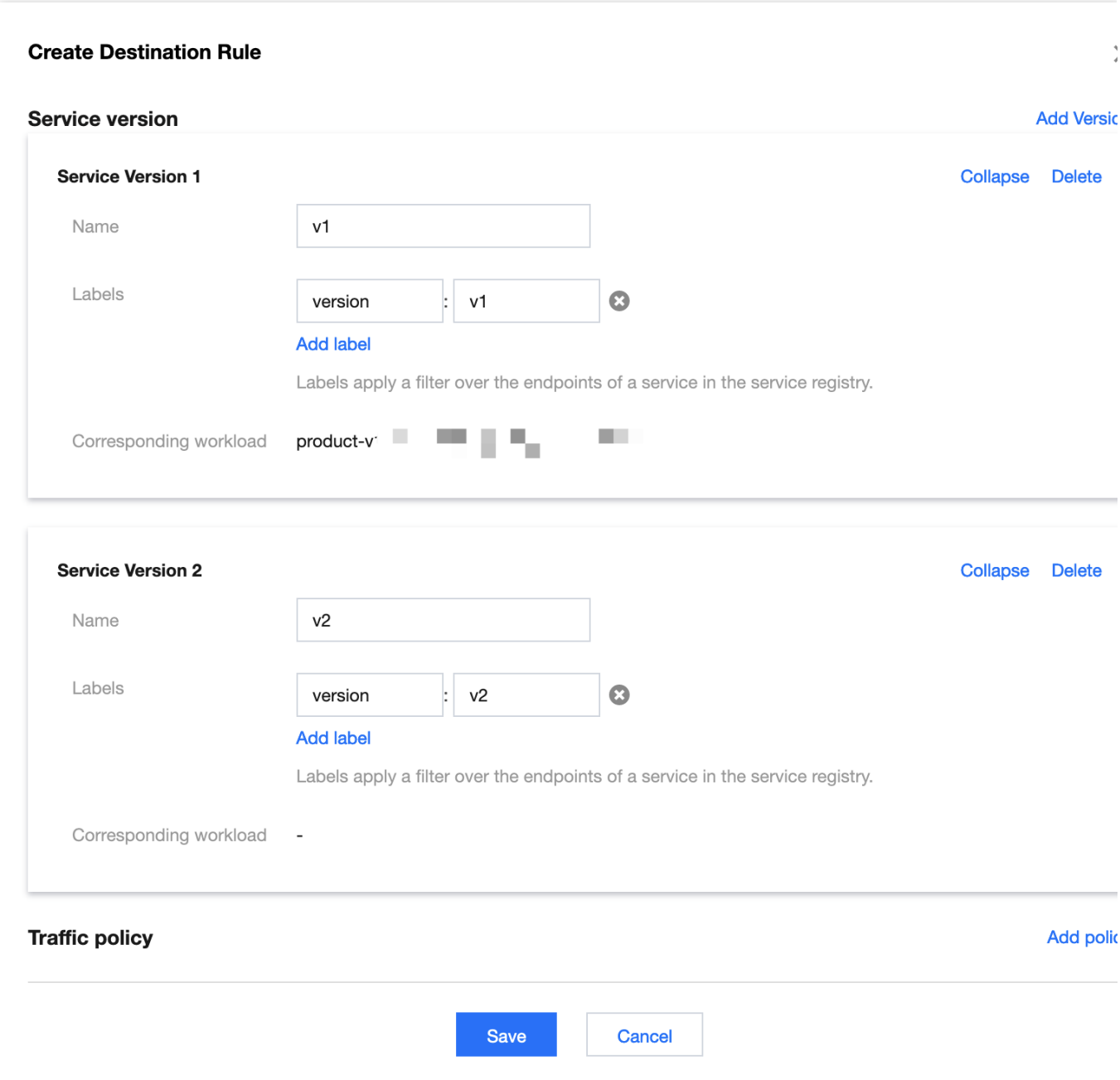
```
    matchLabels:
      app: order
      version: v2
  template:
    metadata:
      labels:
        app: order
        version: v2
    spec:
      containers:
        - name: order
          image: ccr.ccs.tencentyun.com/zhulei/testorder2:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneA"
          ports:
            - containerPort: 7000
              protocol: TCP
```

部署完成后，由于还未配置路由规则，此时访问 order 服务的流量会被随机路由至 v1 版本或 v2 版本。如下图所示：

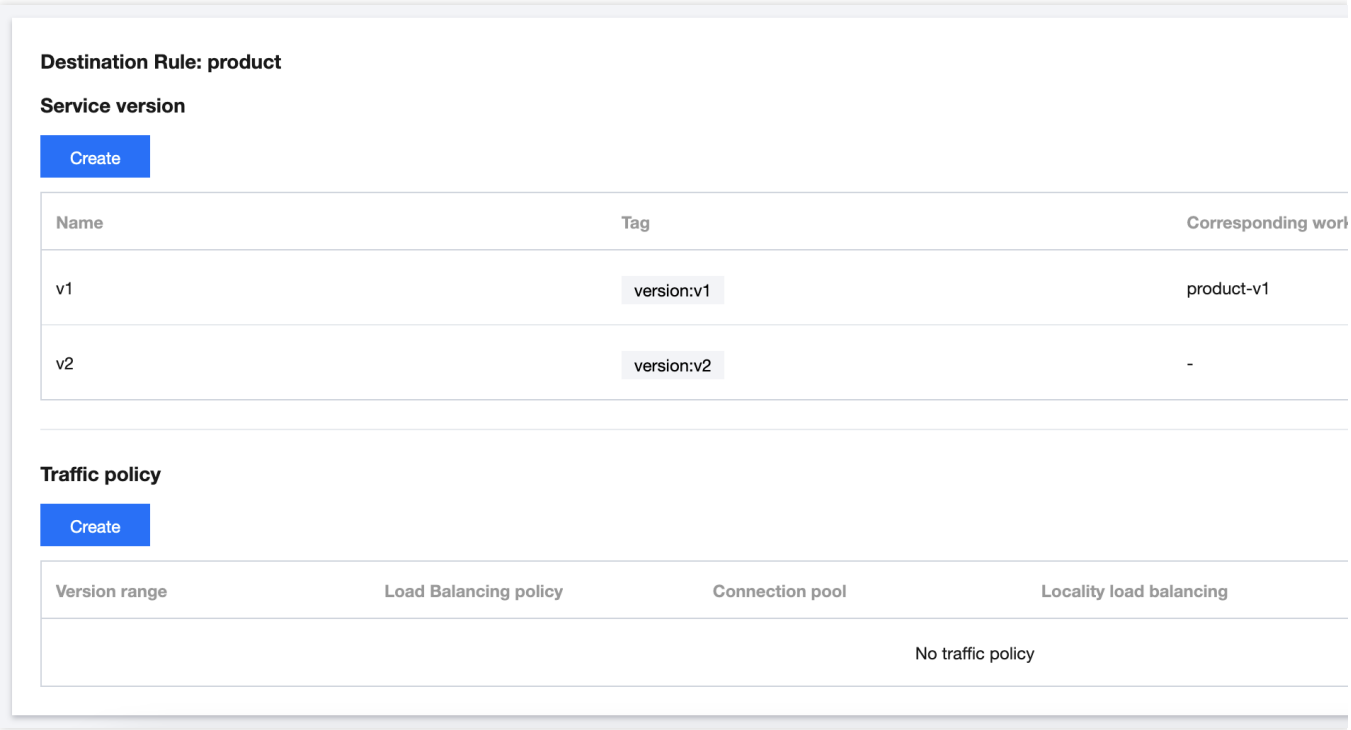


配置基于流量特征内容的路由规则前先需要通过 DestinationRule 定义 order 服务的两个版本。如下图所示：

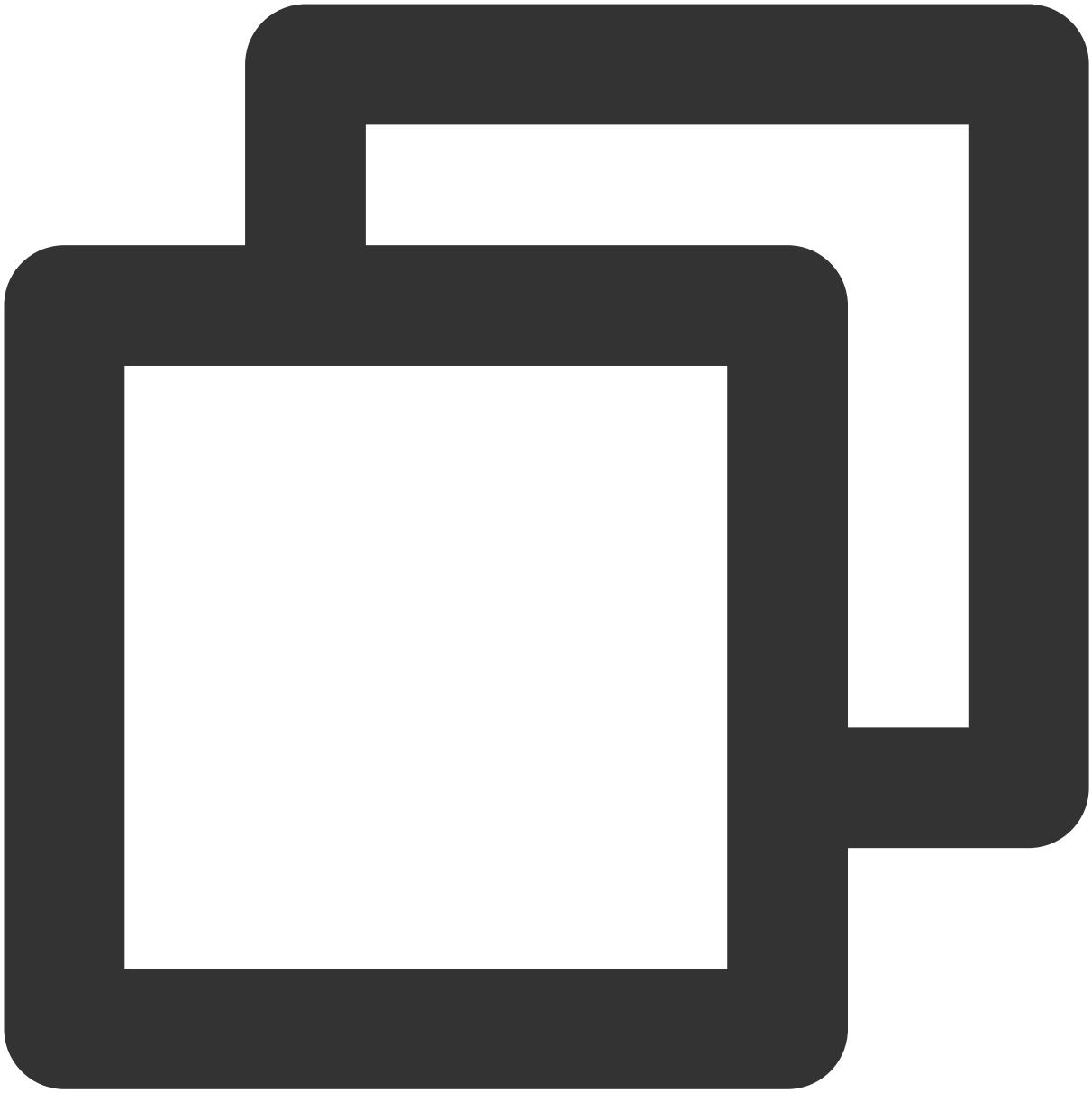
定义 order 服务的版本如下图所示：



order 服务版本定义完成如下图所示：



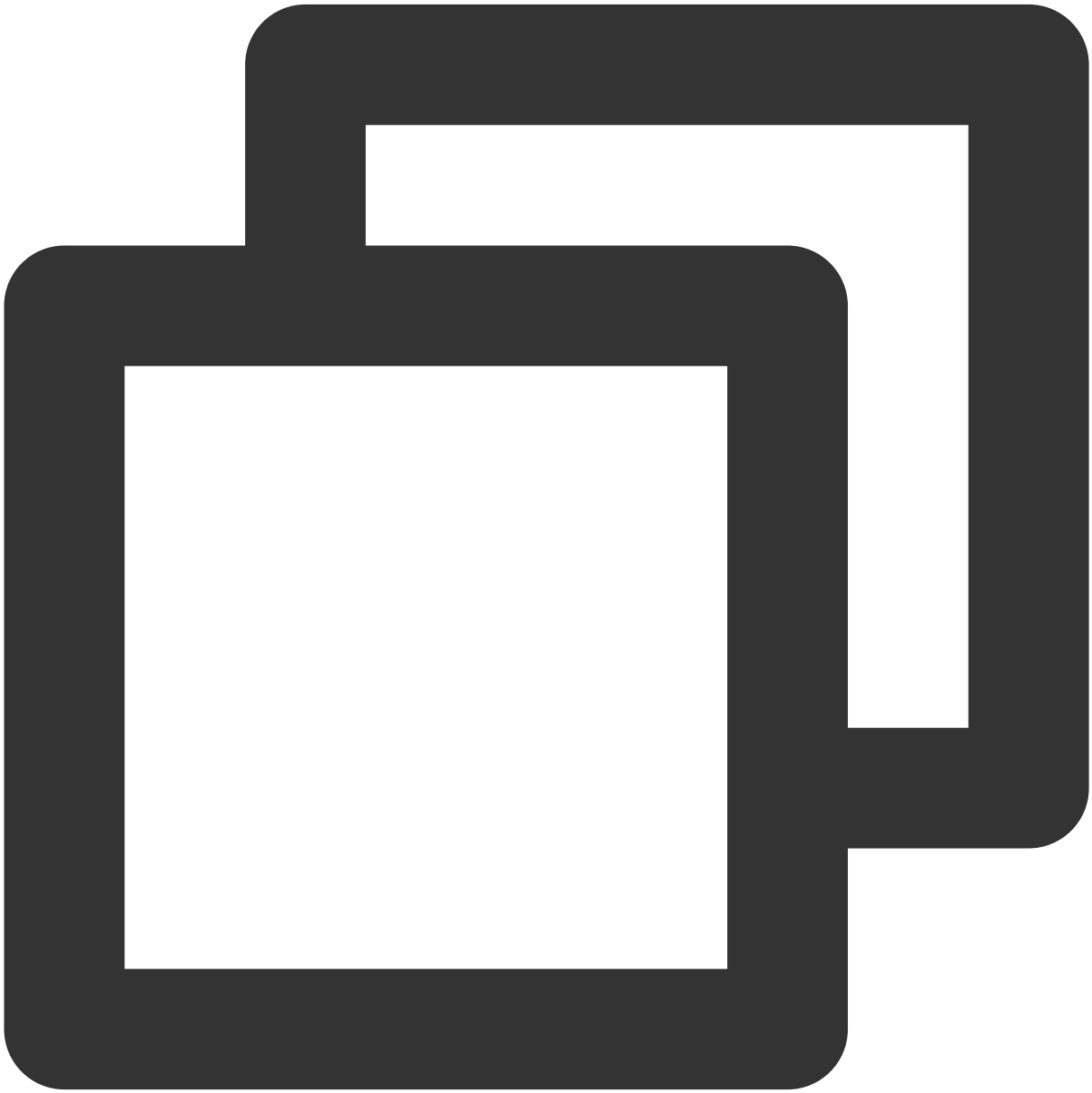
或可通过提交 YAML 文件至主集群完成 Destination Rule 的创建：



```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: order
  namespace: base
spec:
  host: order
  subsets:
    - name: v1
      labels:
        version: v1
```

```
    - name: v2
      labels:
        version: v2
  exportTo:
    - '*'
```

两个版本定义完成后，通过 VirtualService 定义按流量特征进行路由，请求的 header-cookie 中 vip=false 时路由至 order 服务的 v1 subset，vip=true 时路由至 order 服务的 v2 subset。即会员的请求路由至 order v2，非会员的请求路由至 order v1。提交以下 yaml 资源至主集群，即可完成上述配置。

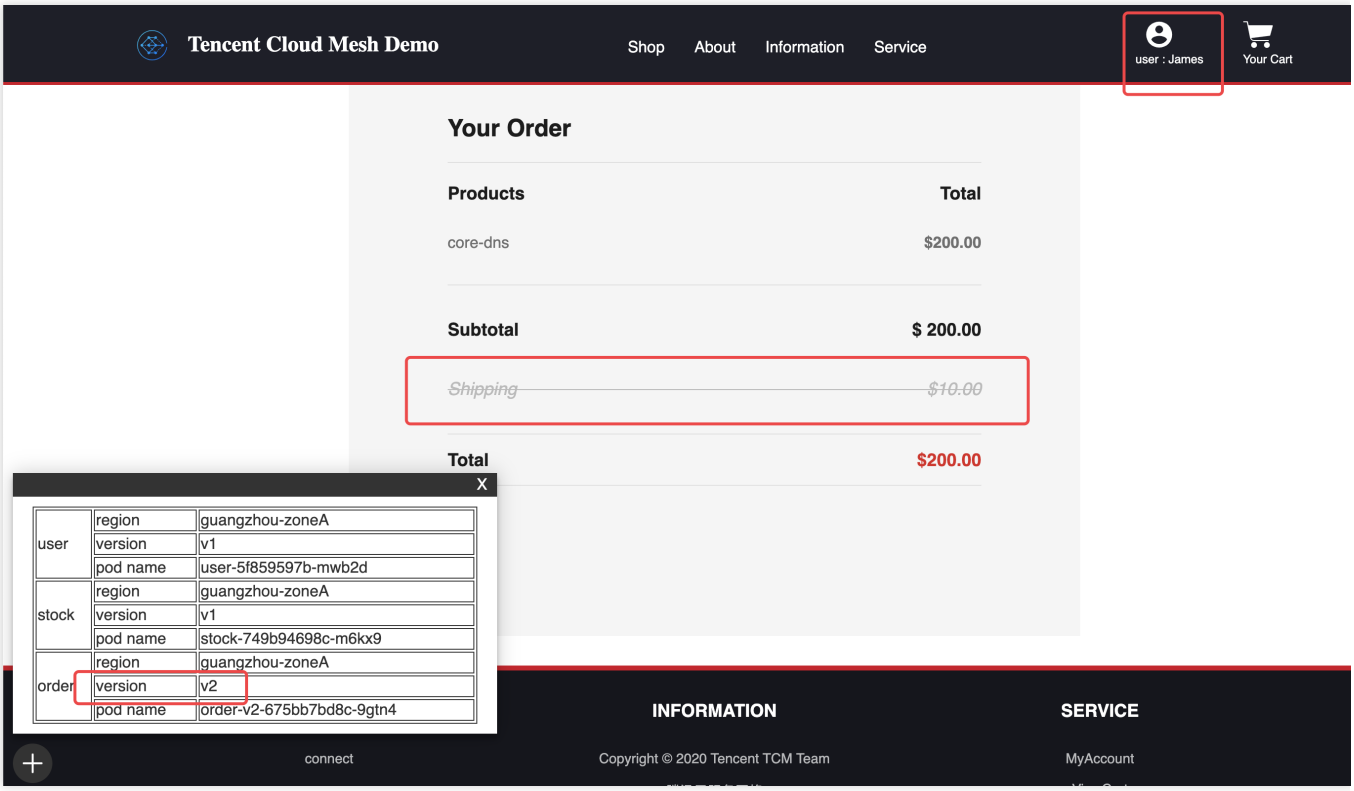


```
apiVersion: networking.istio.io/v1alpha3
```

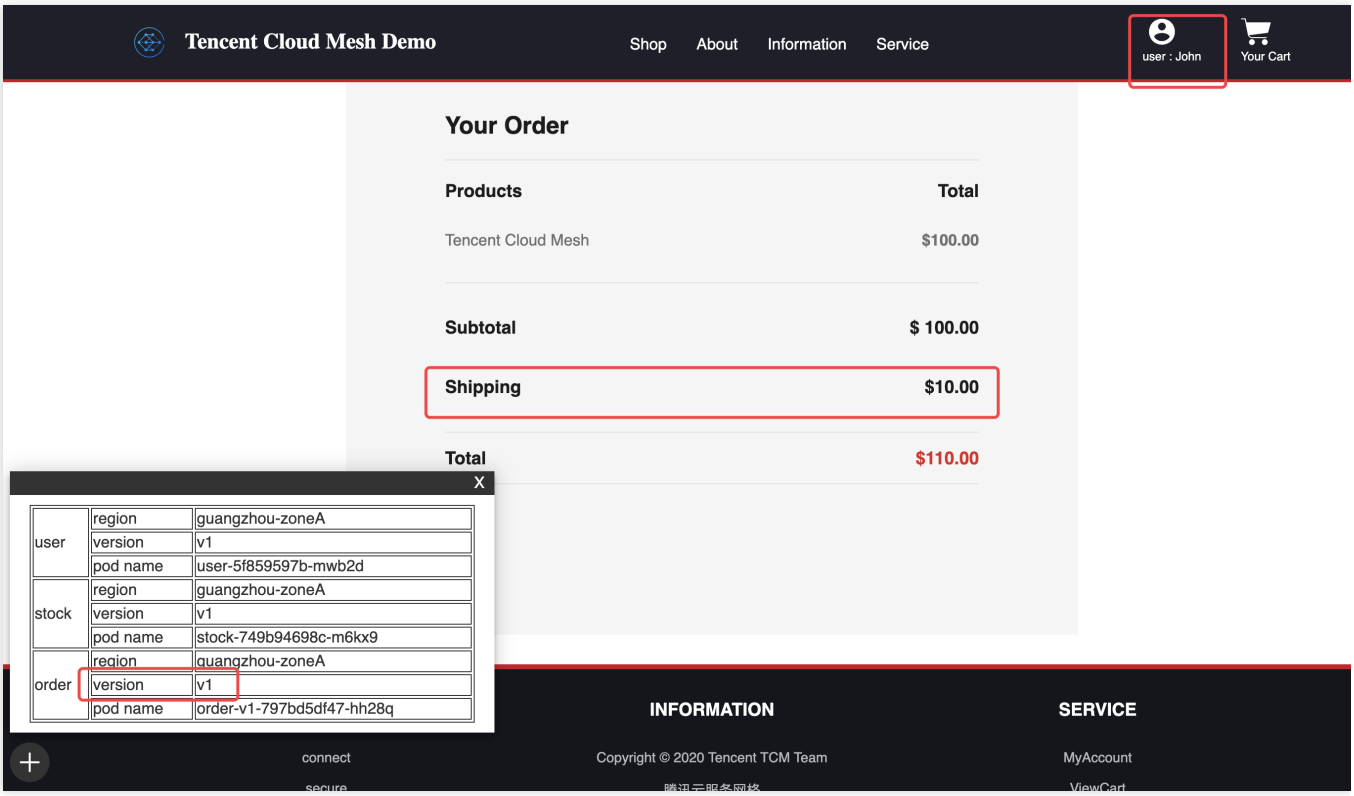
```
kind: VirtualService
metadata:
  name: order-vs
  namespace: base
spec:
  hosts:
    - order.base.svc.cluster.local
  http:
    - match:
        - headers:
            cookie:
              exact: vip=false
      route:
        - destination:
            host: order.base.svc.cluster.local
            subset: v1
    - match:
        - headers:
            cookie:
              exact: vip=true
      route:
        - destination:
            host: order.base.svc.cluster.local
            subset: v2
```

配置完成后，可登录会员账号（ID：1-3）加购和买单，发现有运费抵扣功能，流量被路由到了 order v2 版本；登录非会员账号（ID：4-5）加购和买单，发现无运费抵扣功能，根据 header 中的 VIP 字段信息，请求被路由到了最初部署的 order v1 版本。版本信息也可通过左下角悬浮窗中的信息观察。

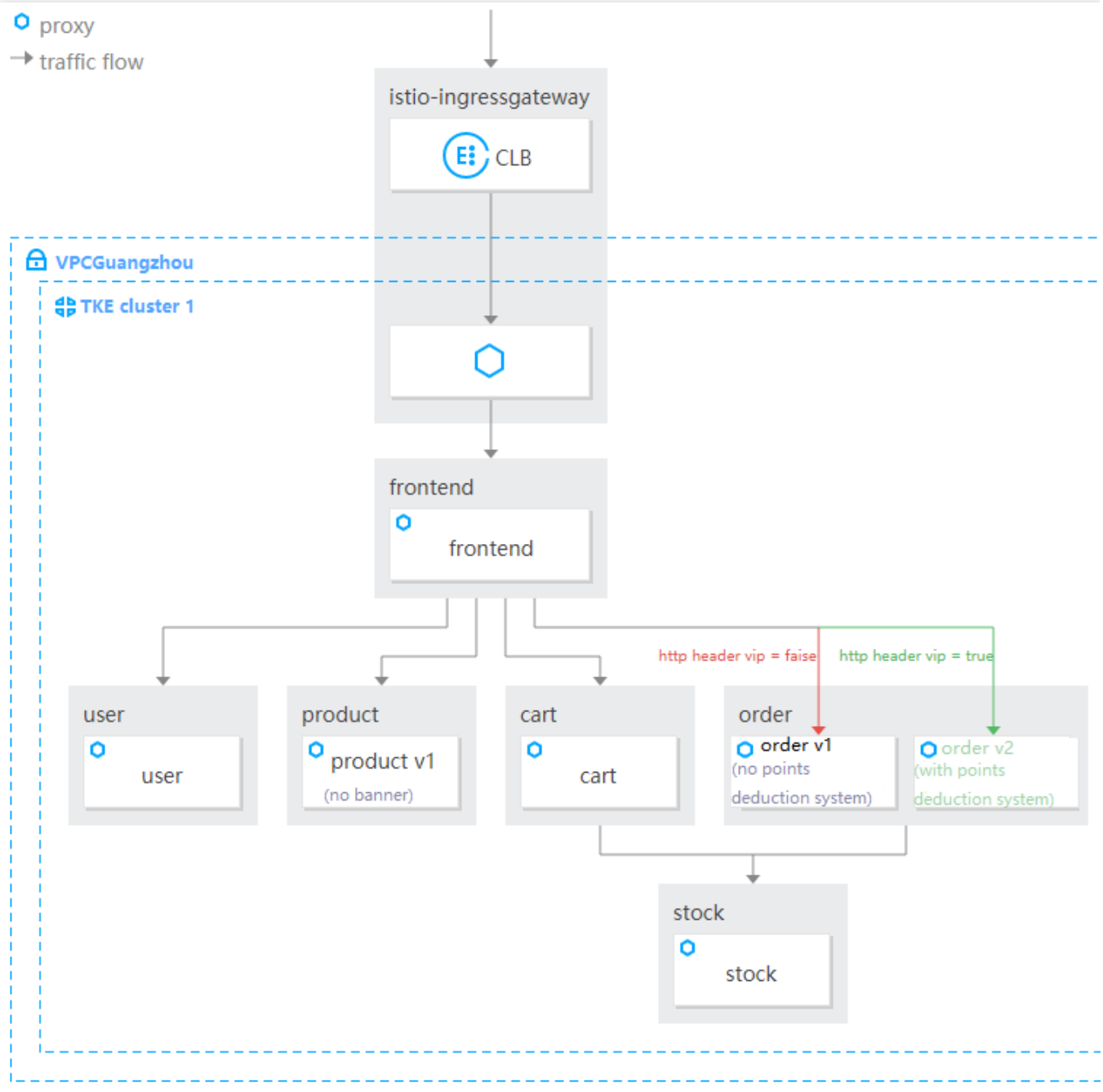
会员用户请求被路由到 v2 版本如下图所示：



非会员用户请求被路由到 v1 版本如下图所示：



基于流量规则的路由如下图所示：

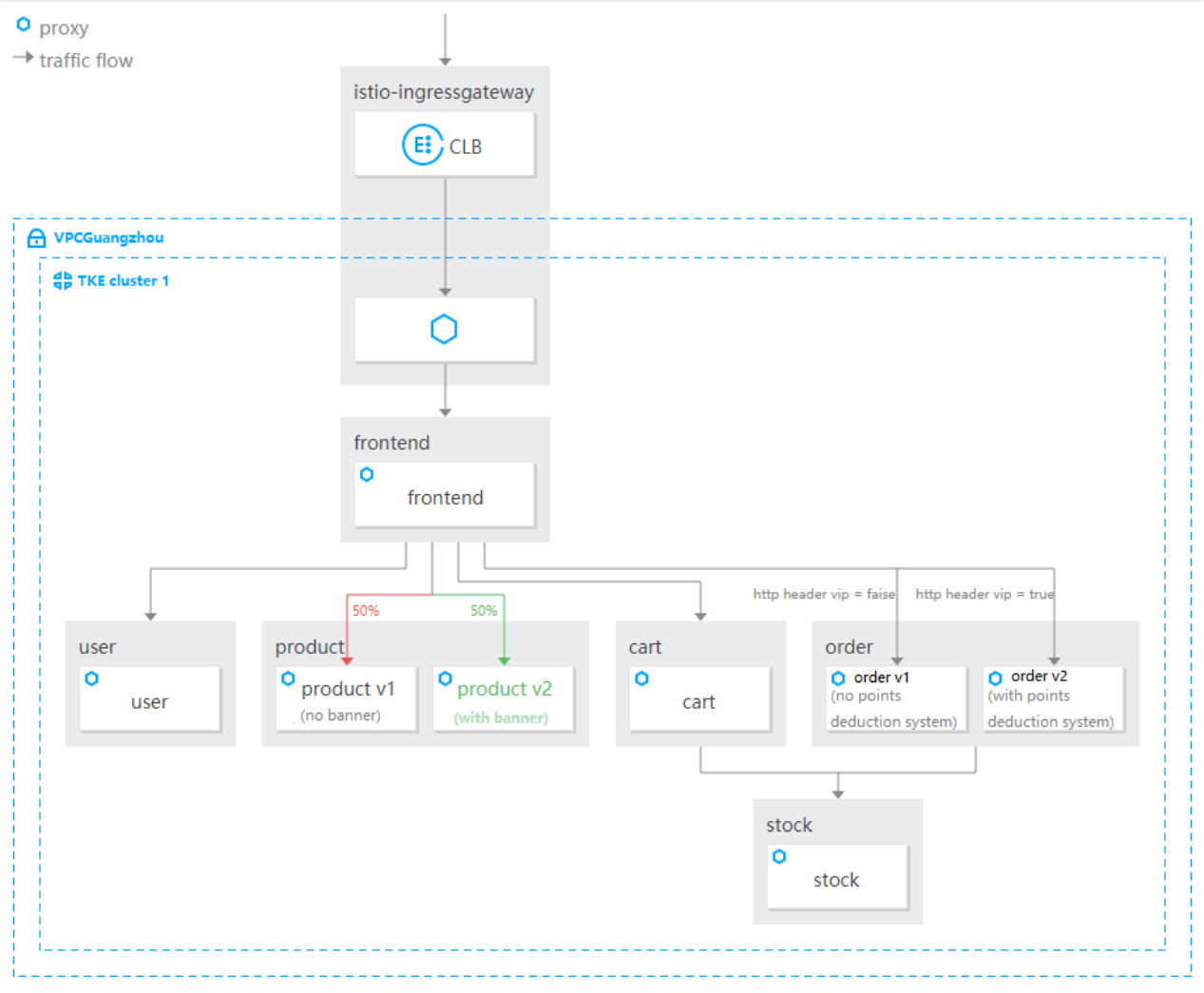

proxy
traffic flow
istio-ingressgateway
CLB
VPCGuangzhou
TKE cluster 1
frontend
frontend
http header vip = faise
http header vip = true
user
user
product
product v1
(no banner)
cart
cart
order
order v1
(no points
deduction system)
order v2
(with points
deduction system)
stock
stock

灰度发布

最近更新时间：2023-12-26 10:45:02

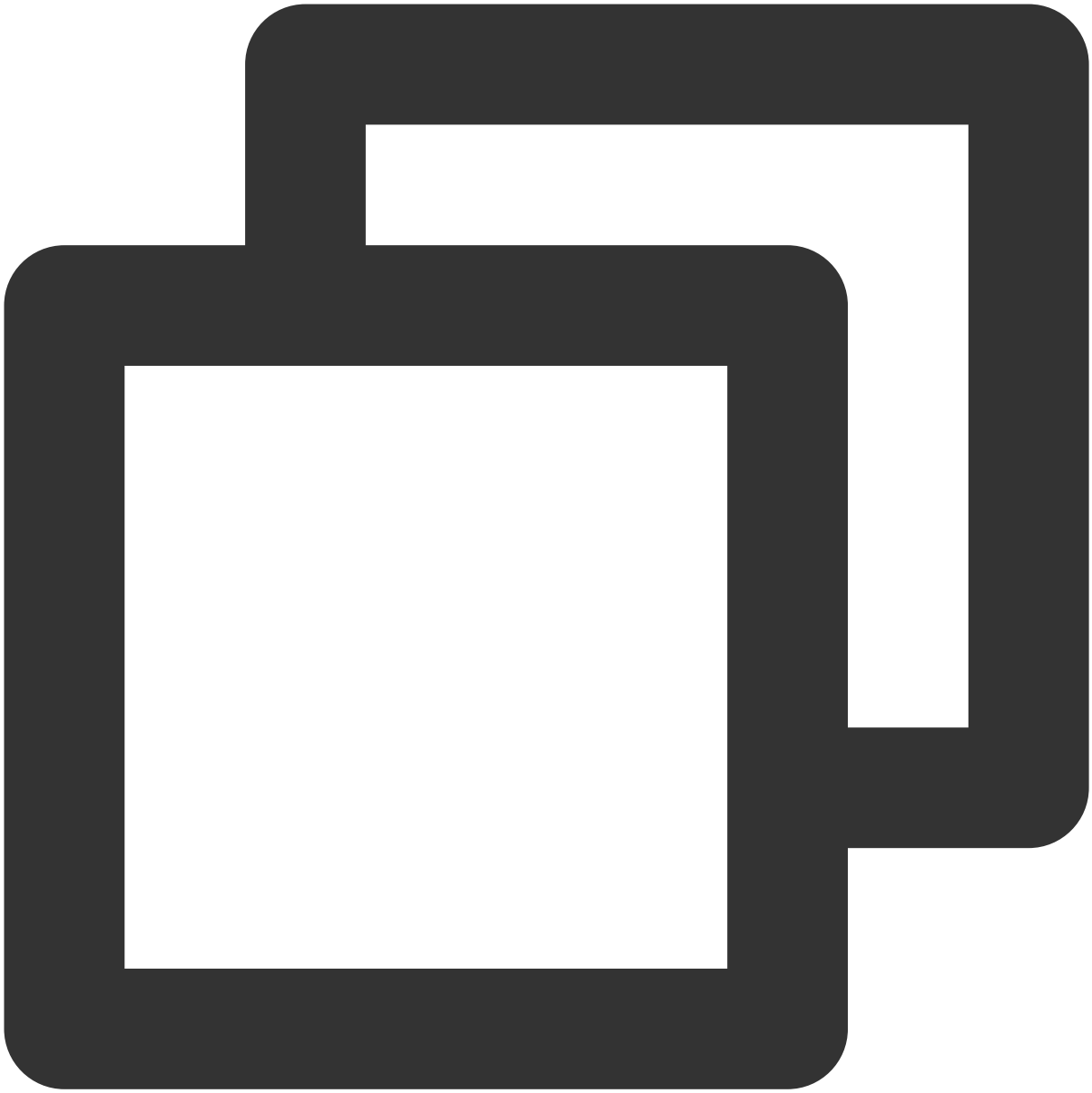
操作场景

随着网站流量的增加，网站开始有了广告投放的需求，广告投放需要在商品页面增加广告位。网站的开发人员新开发了 product 服务的 v2 版本，以 product v2 的 deployment 的形式提供，并希望对 product-v2 版本做灰度发布。灰度发布概览图如下所示：



操作步骤

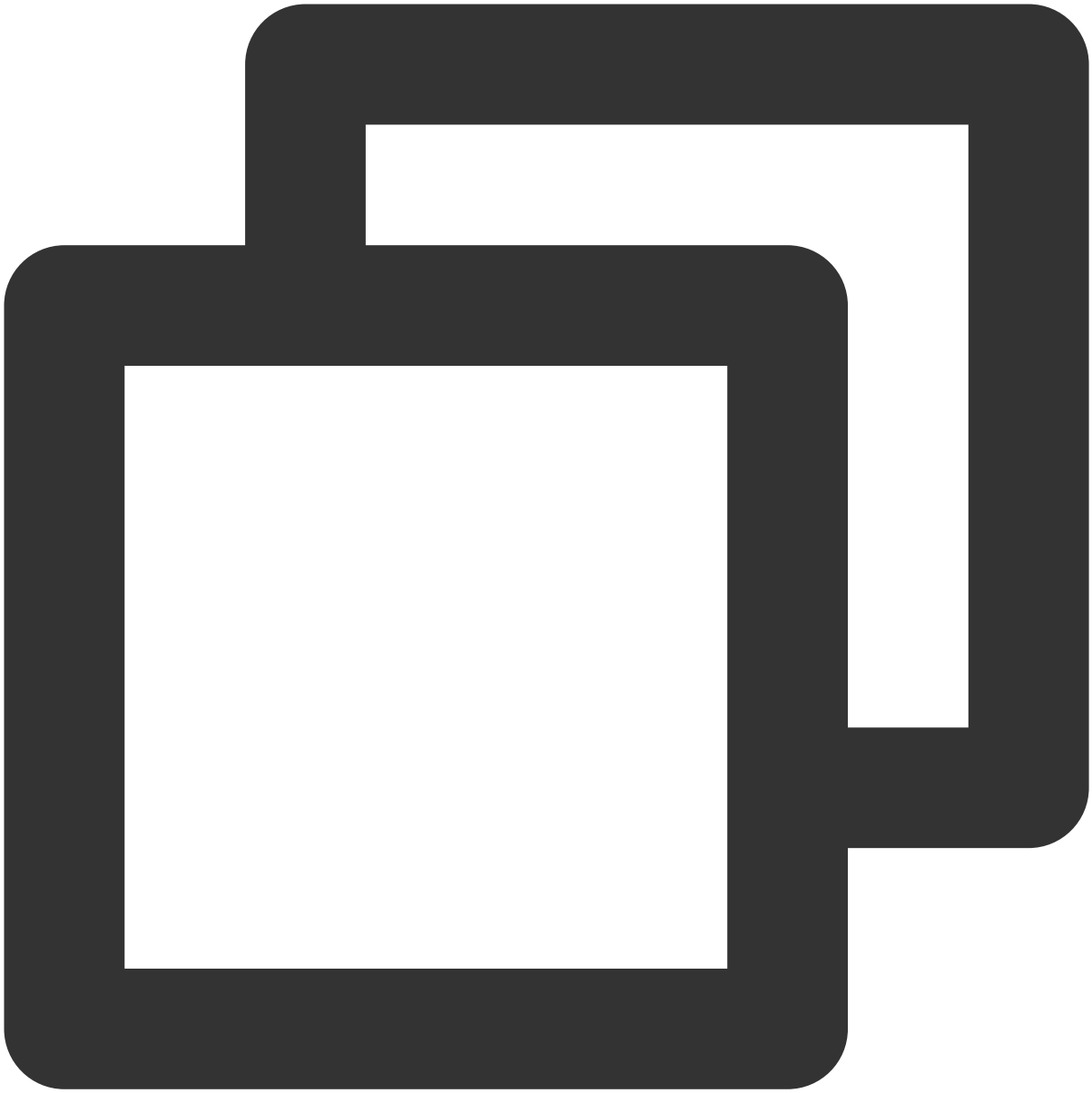
先将 product v2 版本的 deployment 部署至主集群：



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: product-v2
  namespace: base
  labels:
    app: product
    version: v2
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: product
      version: v2
  template:
    metadata:
      labels:
        app: product
        version: v2
    spec:
      containers:
        - name: product
          image: ccr.ccs.tencentyun.com/zhulei/testproduct2:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneA"
          ports:
            - containerPort: 7000
```

通过 DR 定义服务版本 + 通过 VS 定义权重路由来完成灰度发布的第一步，将部分流量（50%）路由至 product v2 subset 以验证新版本，剩余部分（50%）的流量仍然路由至 product v1 版本。将以下 YAML 文件提交至主集群即可完成以上设定。

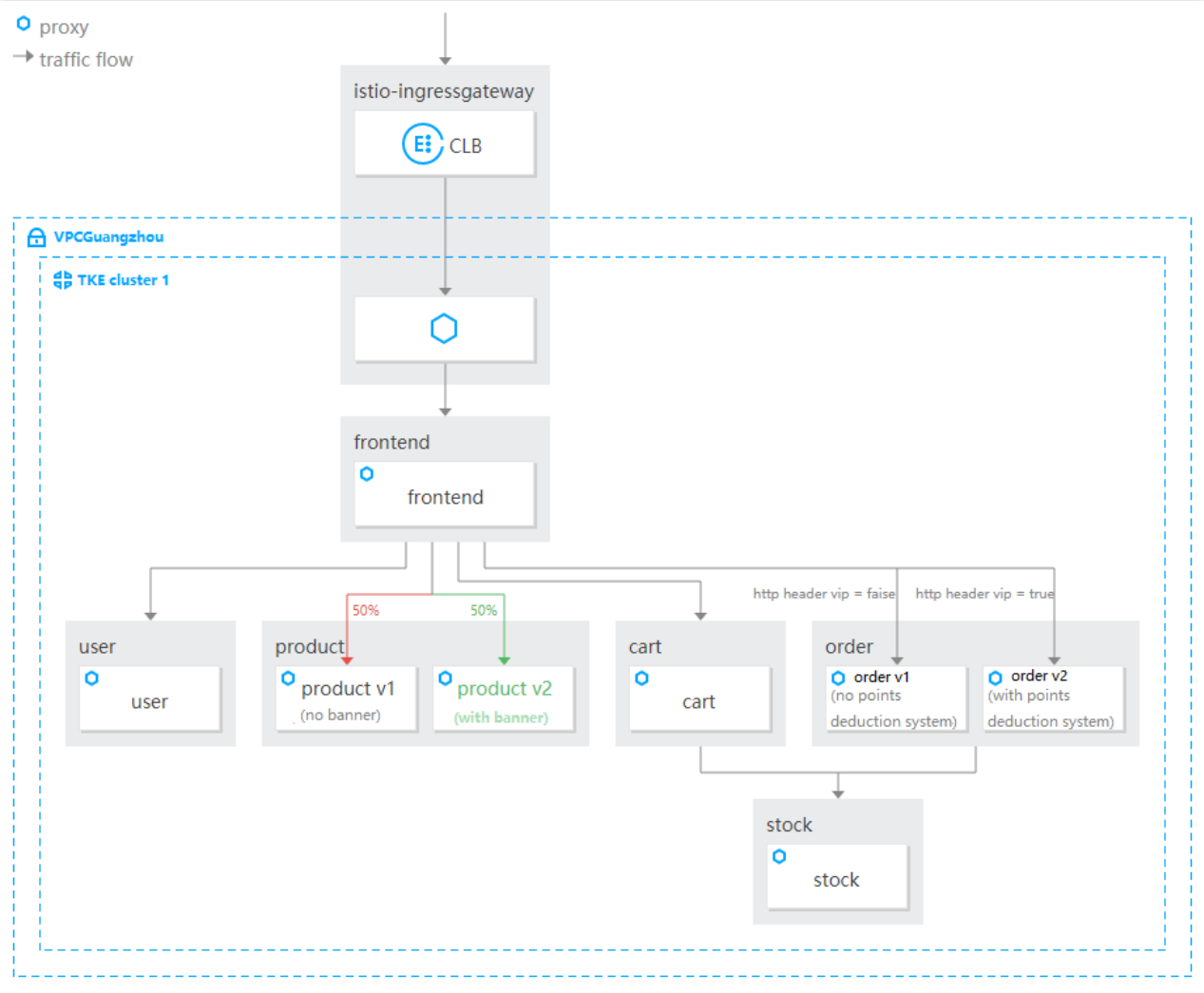


```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: product-vs
  namespace: base
spec:
  hosts:
    - "product.base.svc.cluster.local"
  http:
    - match:
        - uri:
```

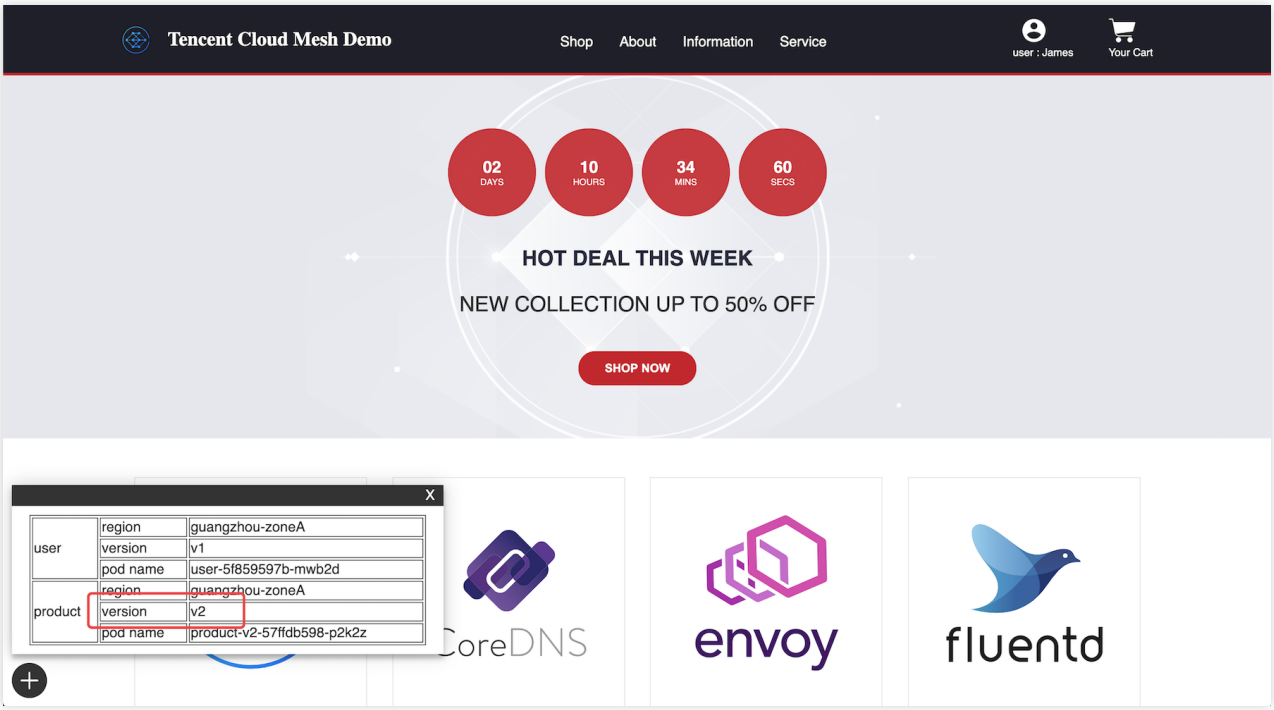
```
            exact: /product
      route:
        - destination:
            host: product.base.svc.cluster.local
            subset: v1
            port:
              number: 7000
          weight: 50
        - destination:
            host: product.base.svc.cluster.local
            subset: v2
            port:
              number: 7000
          weight: 50
---

apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: product
  namespace: base
spec:
  host: product
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2
```

配置完成后，访问 product 服务的流量将有 50%被路由至 v1 版本，50%被路由至 v2 版本，刷新网站商品页面即可验证。

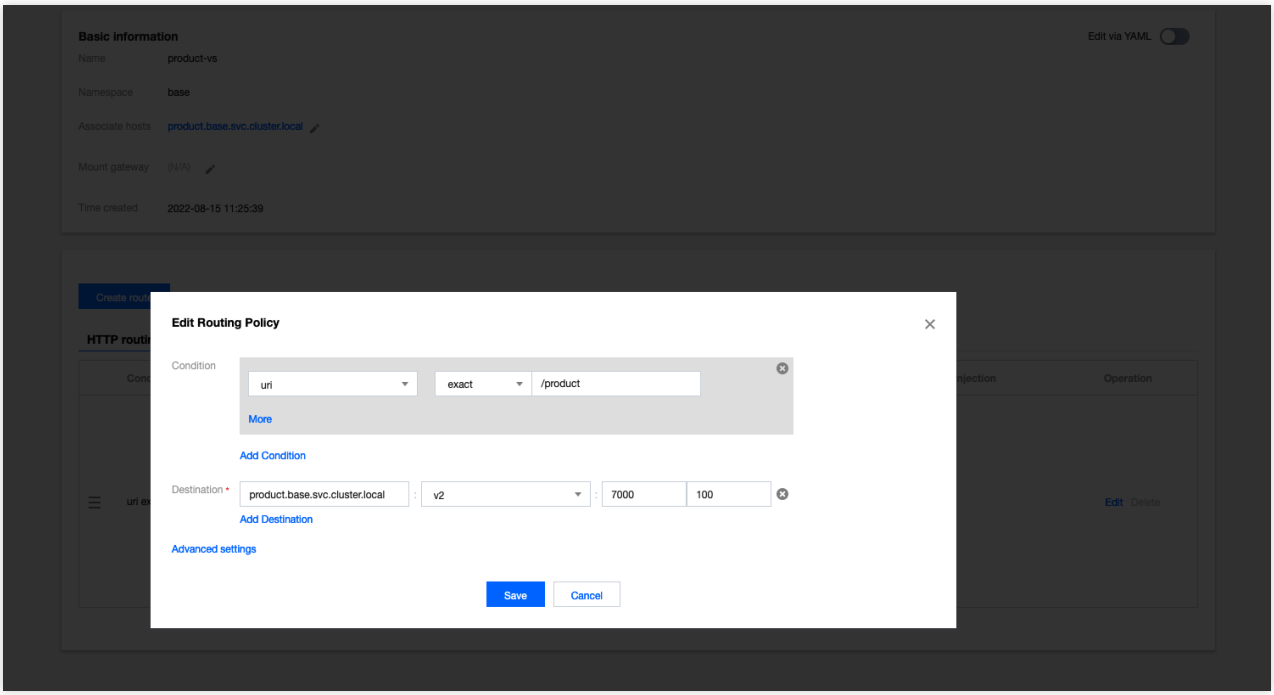
权重路由如下图所示：



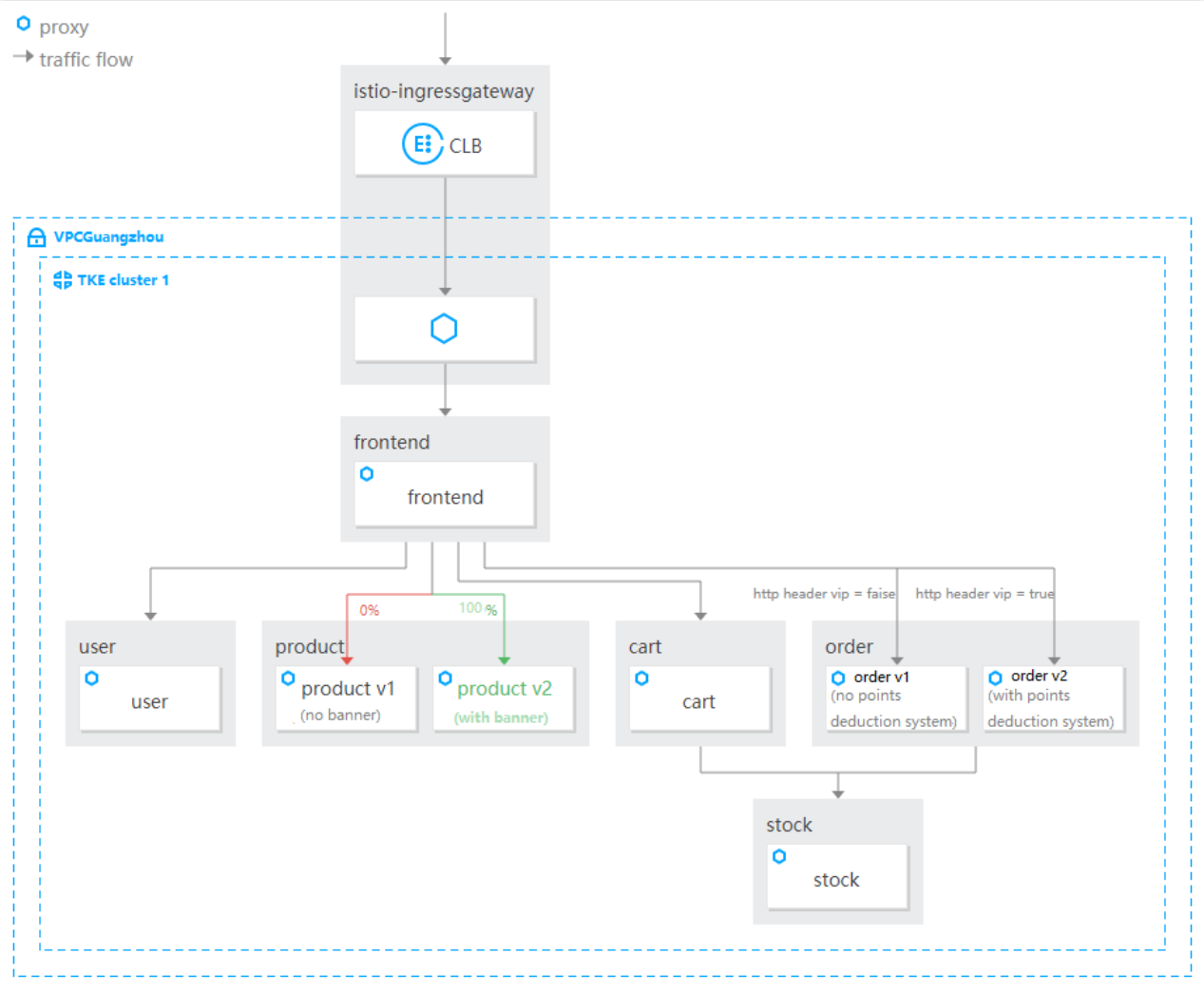
50%的请求路由到 product v2 版本如下图所示：



product v2 版本验证通过后，即可修改关联 product 的 VirtualService 中路由规则目的端的权重，设置访问 product 服务的所有流量（100%）至 v2 版本，设置完成后可刷新商品列表页面验证。基于 virtual Service 更改权重如下图所示：



灰度发布完成如下图所示：


proxy
traffic flow
istio-ingressgateway
CLB
VPCGuangzhou
TKE cluster 1
frontend
frontend
0%
100%
http header vip = faise
http header vip = true
user
user
product
product v1
(no banner)
product v2
(with banner)
cart
cart
order
order v1
(no points
deduction system)
order v2
(with points
deduction system)
stock
stock

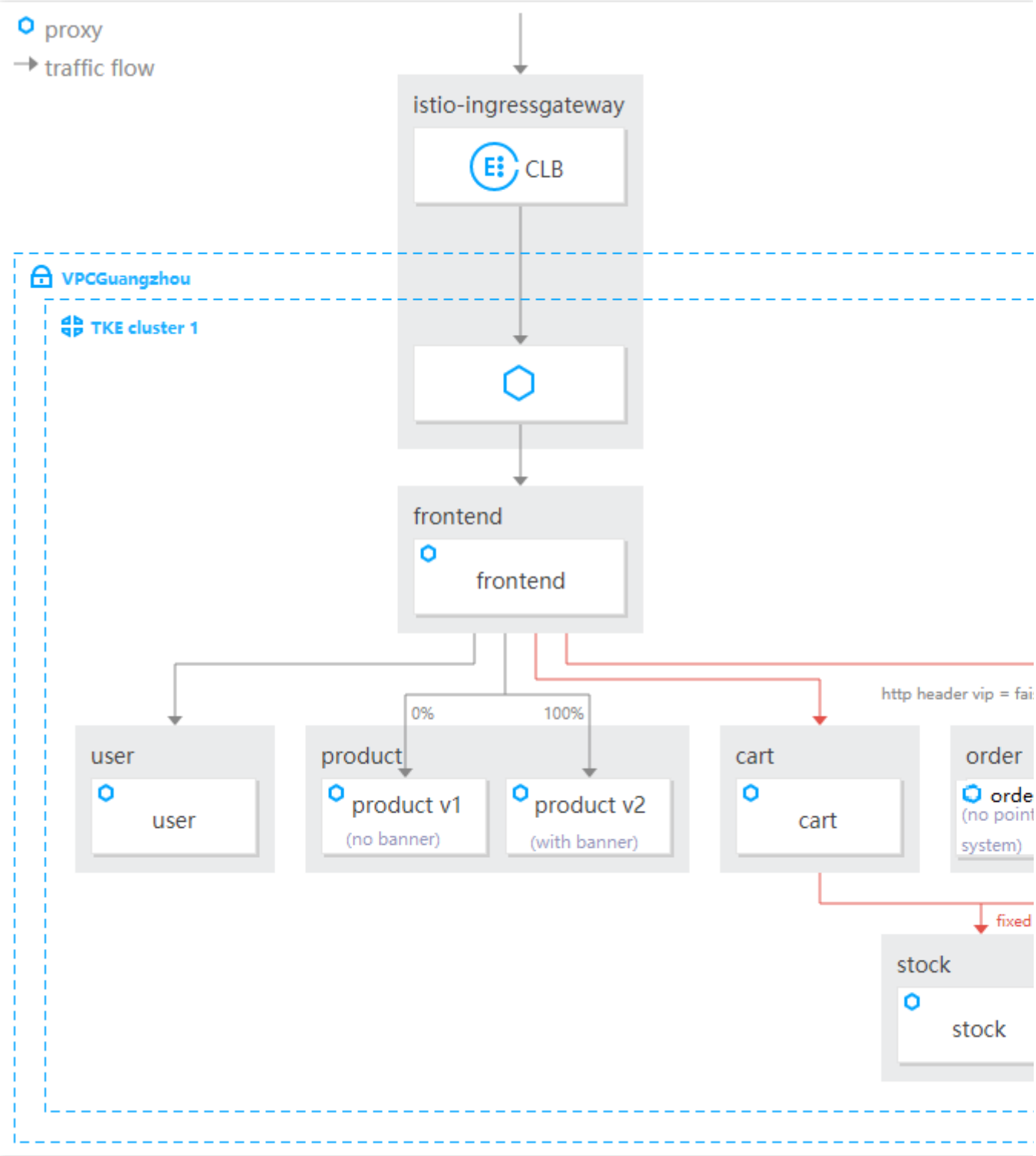
服务高可用
故障注入测试

最近更新时间：2023-12-26 11:30:38

操作场景

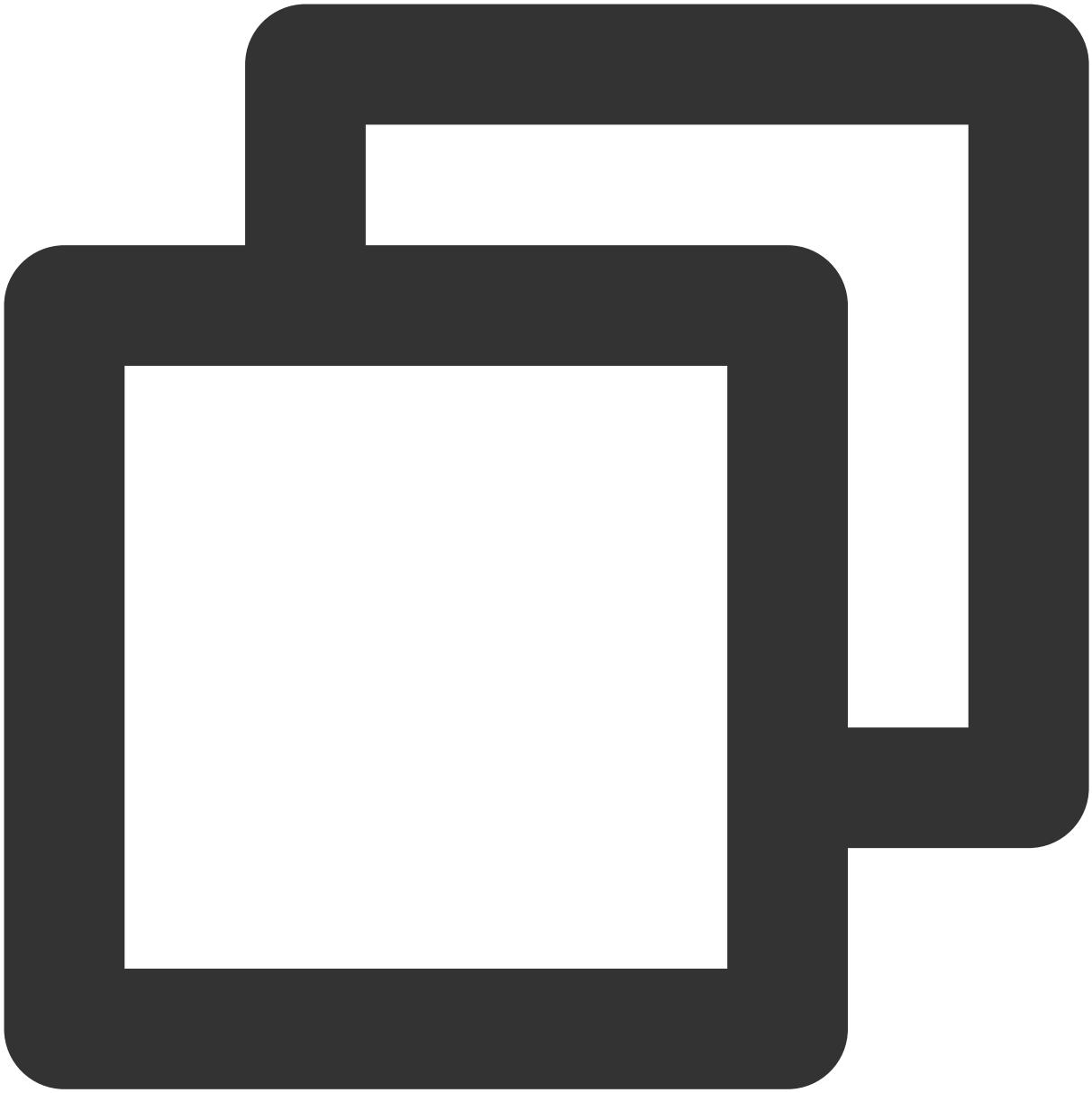
电商网站业务团队需要模拟访问库存服务存在延迟故障时网站系统的行为，以测试服务弹性，网站用户的优化访问体验。

stock 服务 fixed delay 7s 如下图所示：



操作步骤

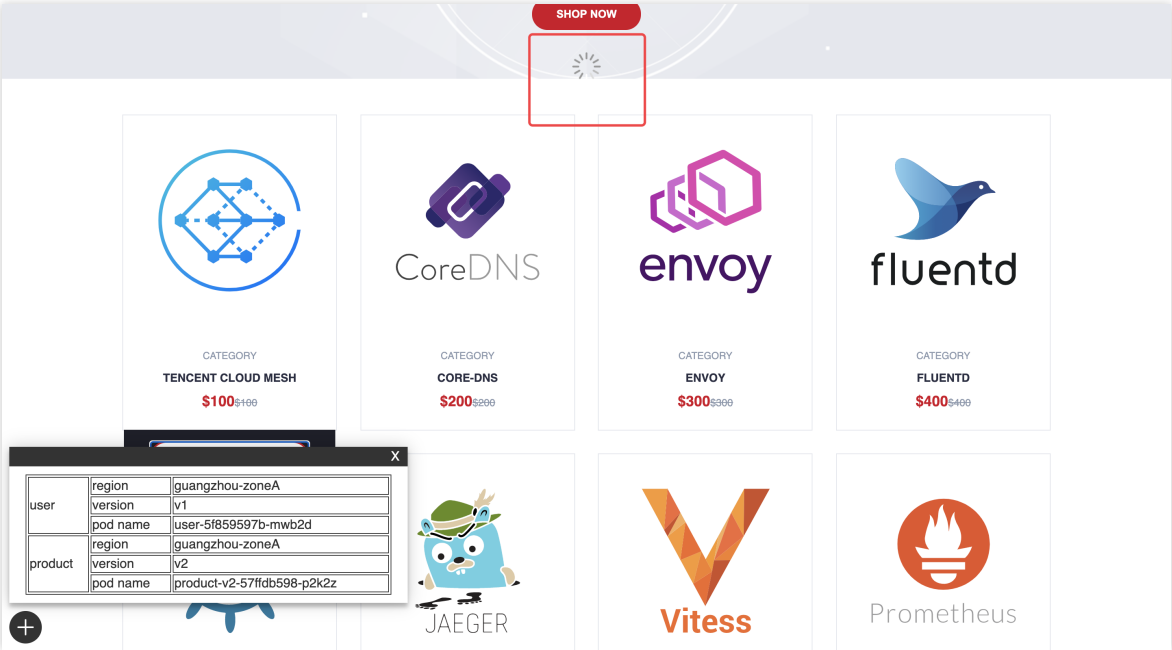
通过配置绑定 stock 服务的 VritualService，设置访问 stock 服务的故障注入策略：100%的请求会有 7 秒的固定延迟。



```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: stock-vs
  namespace: base
spec:
  hosts:
    - stock.base.svc.cluster.local
  http:
    - route:
        - destination:
```

```
            host: stock.base.svc.cluster.local
      fault:
        delay:
          fixedDelay: 7000ms
          percentage:
            value: 100
```

配置完成后，在 Demo 网站页面单击 “ADD TO CART” 加入购物车或者单击 “YOUR CART” 调用购物车服务，购物车服务会调用 stock 服务查询库存，访问 stock 服务有 7 秒的固定延迟故障注入策略，以及在购物车页面单击“CHECKOUT”发起结算会调用 order 服务，order 服务会调用 stock 服务查询库存，访问 stock 服务有 7 秒的固定延迟故障注入策略。此时页面处于加载中的状态会持续 7 秒一直等待故障结束，造成网站用户的不流畅浏览体验。cart 服务调用 stock 服务的等待状态如下图所示：



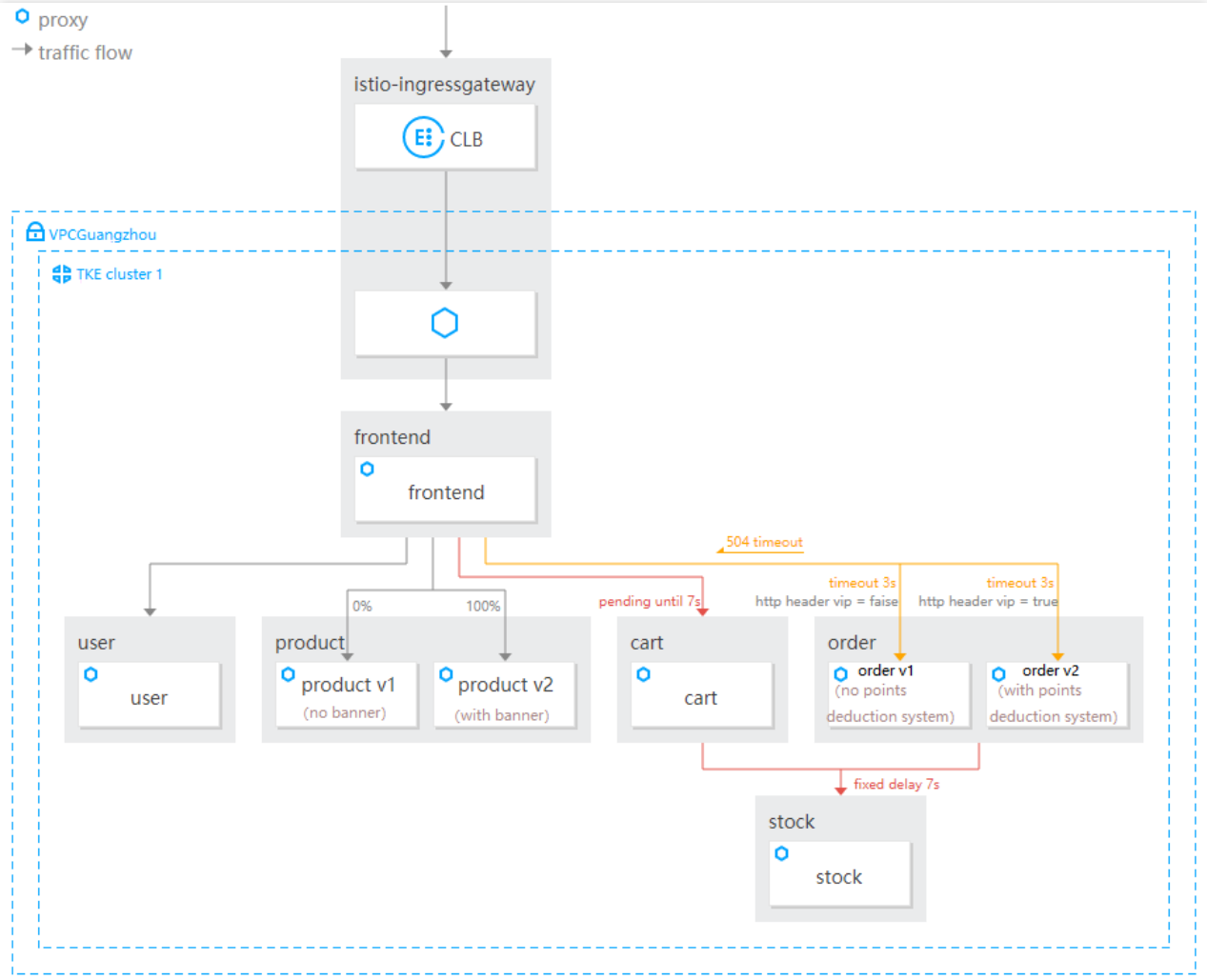
网站用户的浏览体验优化可以通过配置超时 timeout 实现。

服务超时配置

最近更新时间：2023-12-26 10:46:00

操作场景

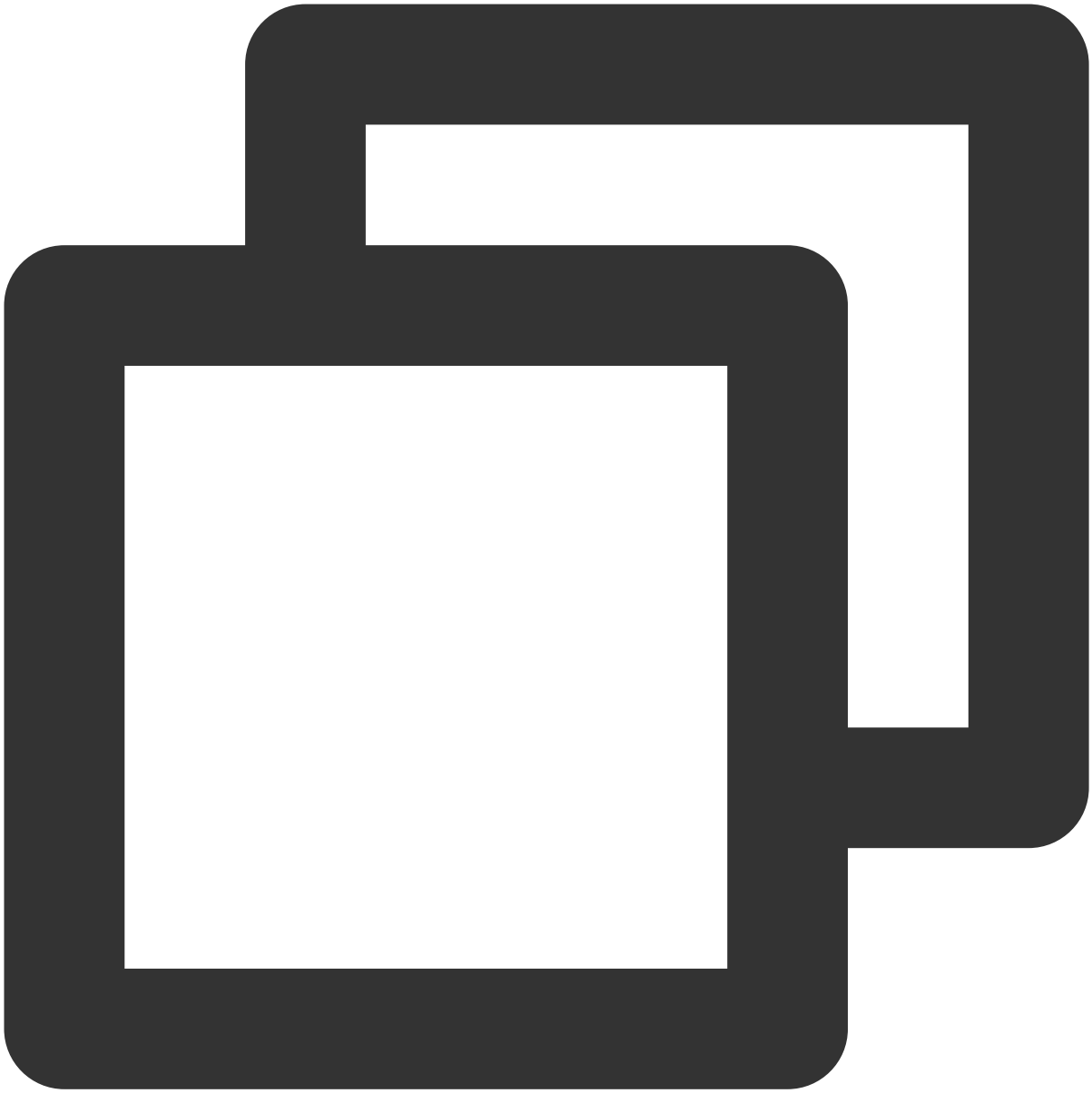
order 服务 timeout 3s 如下图所示：



通过对 stock 服务配置故障注入，发现由于故障会导致网站用户的请求一直处于等待状态，为优化网站用户的浏览体验，需要为服务配置 timeout。

操作步骤

应用以下 VS，为 order 服务配置 3 秒的超时时间，cart 服务不设置超时时间作为参照对比。

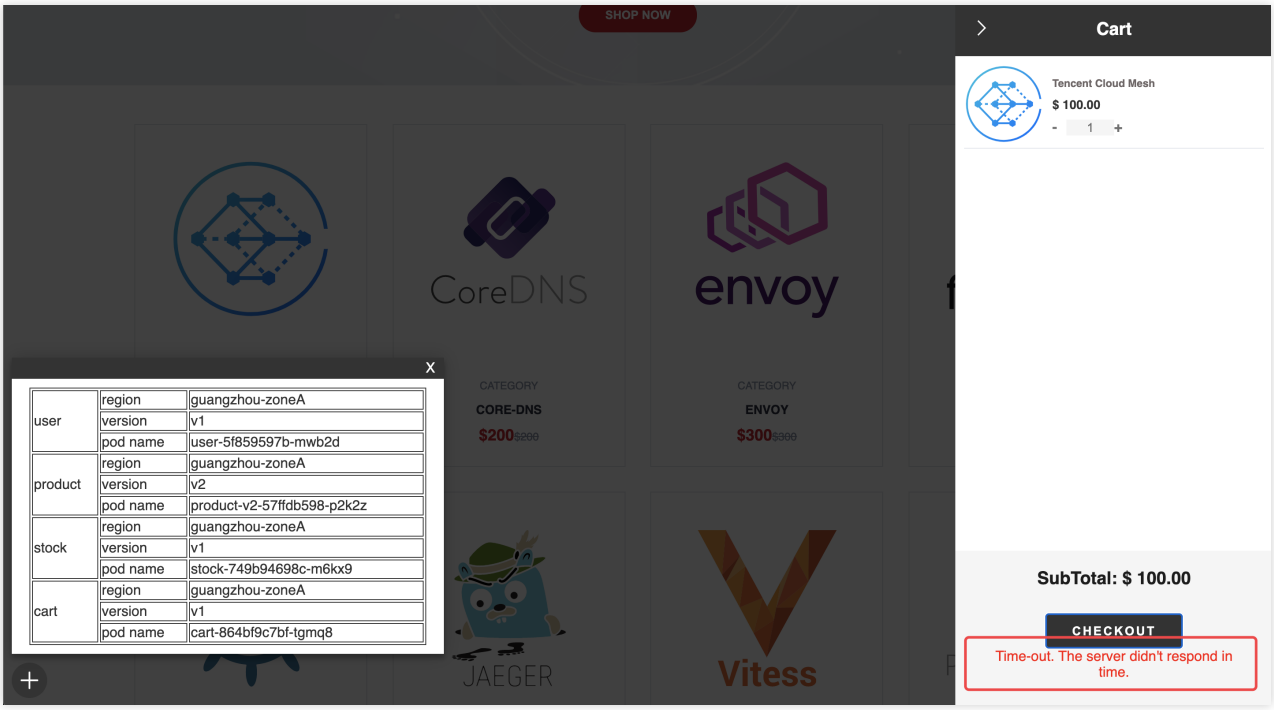


```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: order-vs
  namespace: base
spec:
  hosts:
    - order.base.svc.cluster.local
  http:
    - match:
        - headers:
```

```
            cookie:
              exact: vip=false
      route:
        - destination:
            host: order.base.svc.cluster.local
            subset: v1
      timeout: 3000ms
    - match:
        - headers:
            cookie:
              exact: vip=true
      route:
        - destination:
            host: order.base.svc.cluster.local
            subset: v2
      timeout: 3000ms
```

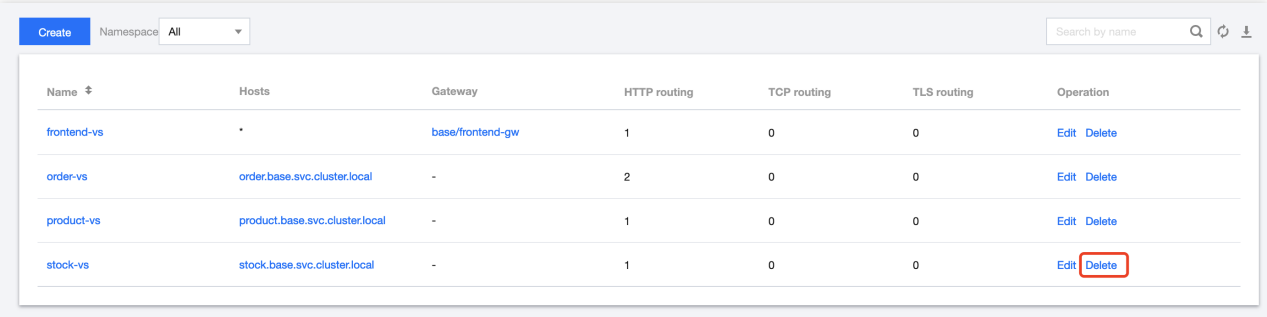
配置完成后，选择商品加入购物车，此时访问 cart 服务会有 7 秒的故障注入访问延时，且没有 timeout 处理，点击“CHECKOUT”发起结算调用 order 服务，此时虽然访问 order 服务也会有 7 秒的故障注入访问延时，但是有 3 秒的 timeout 超时处理，在调用 order 服务 3 秒内没有反应会做超时处理。

cart 服务调用 order 服务 timeout 显示如下图所示：



超时配置已完成，对于服务的故障注入测试已完成，可删除关联 stock 服务的 VirtualService 资源以解除对 stock 服务配置的故障注入策略。

删除 stock 服务关联 Virtual Service 操作如下图所示：

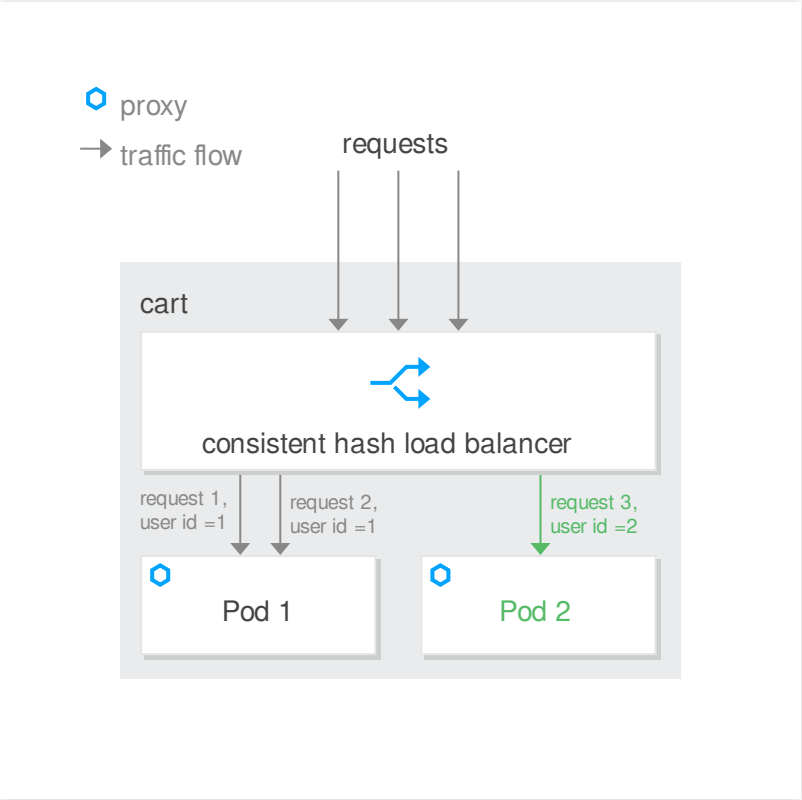

Create
Namespace
All
Search by name
Name
Hosts
Gateway
HTTP routing
TCP routing
TLS routing
Operation
frontend-vs
*
base/frontend-gw
1
0
0
Edit Delete
order-vs
order.base.svc.cluster.local
-
2
0
0
Edit Delete
product-vs
product.base.svc.cluster.local
-
1
0
0
Edit Delete
stock-vs
stock.base.svc.cluster.local
-
1
0
0
Edit Delete

会话保持

最近更新时间：2023-12-26 11:33:14

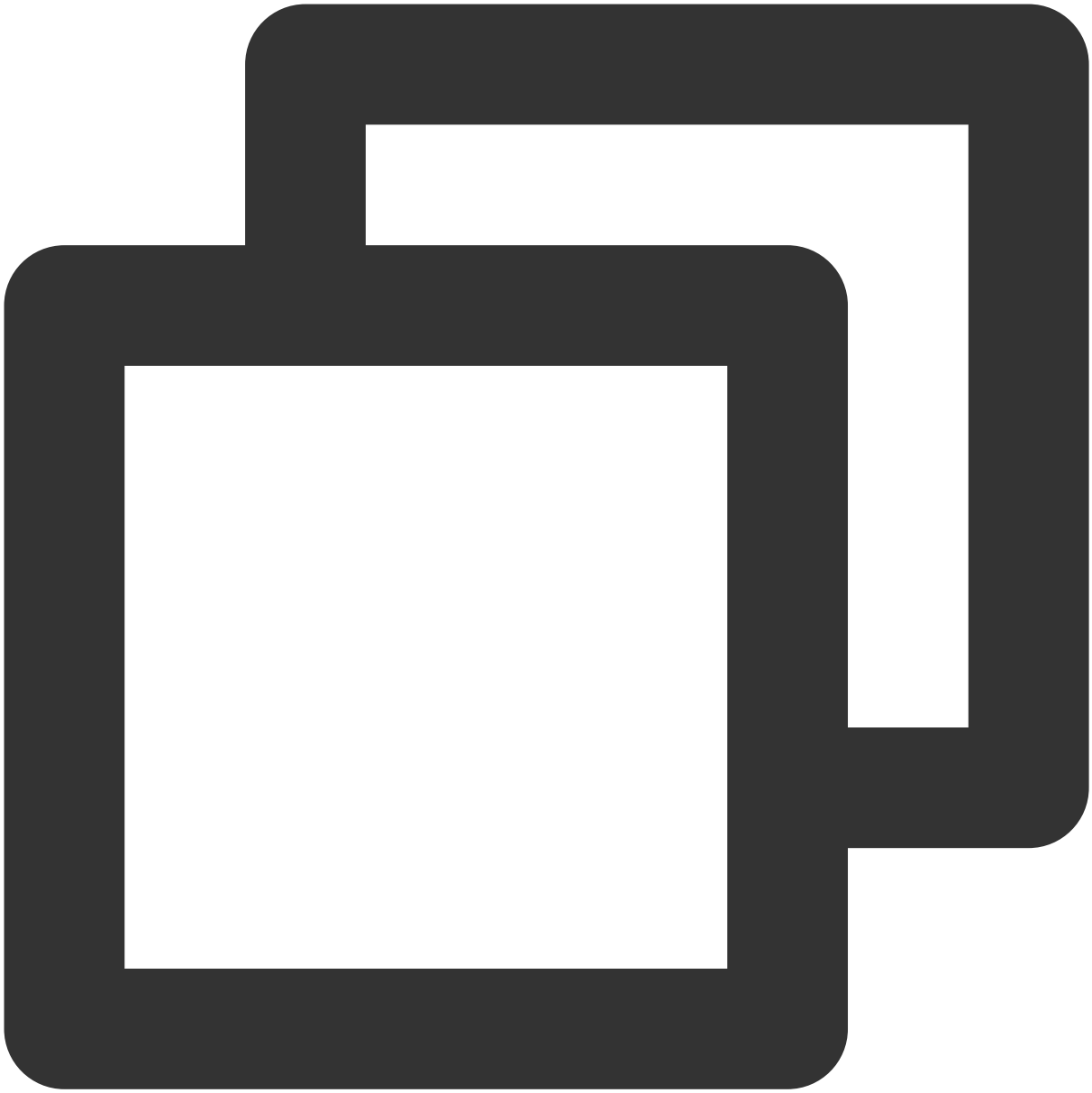
操作场景

购物车服务由多个 pod 副本运行，需要会话保持功能，以保证同一用户请求被路由至同一个 pod，保证同一用户的购物车信息不会丢失。
会话保持如下图所示：



操作步骤

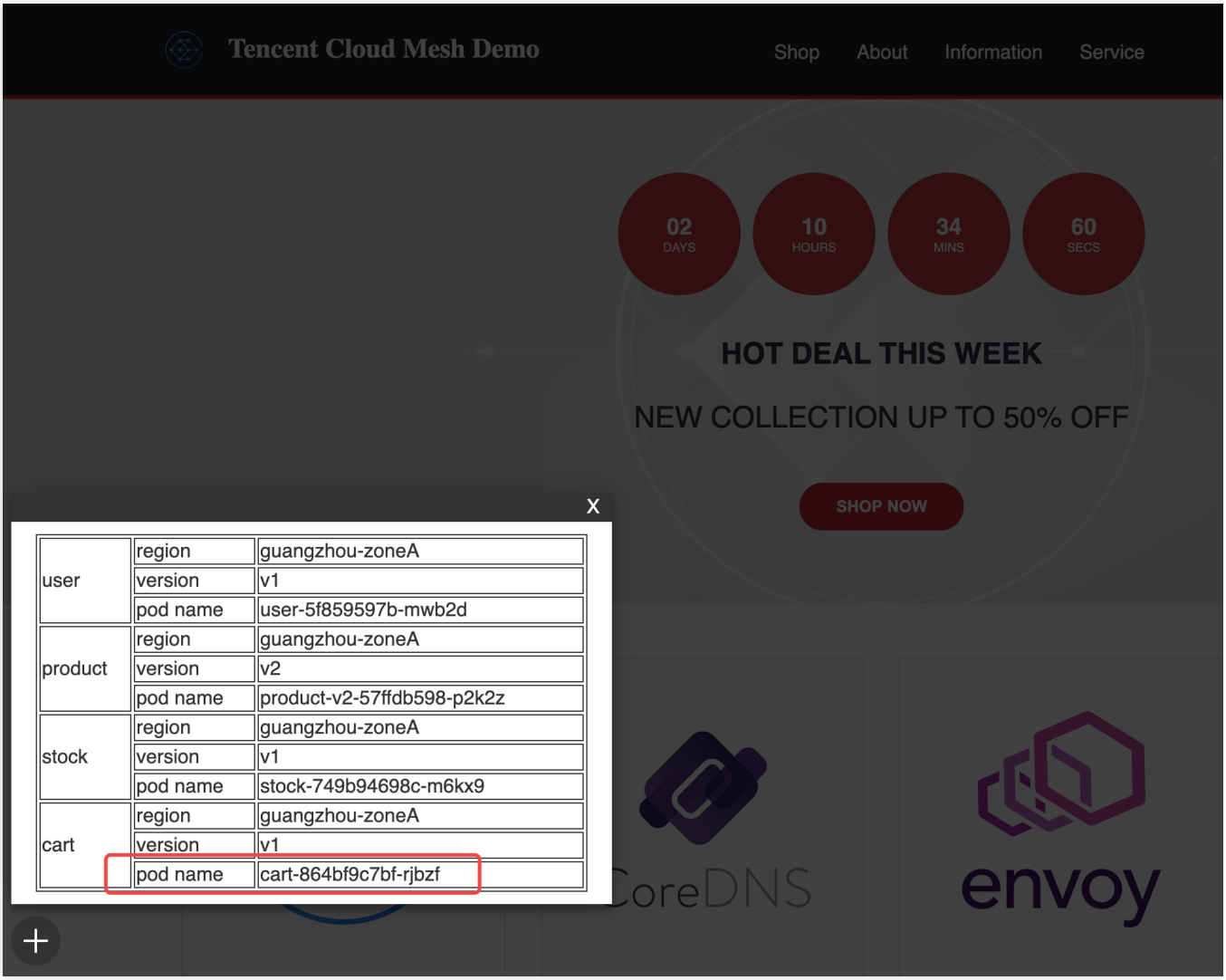
会话保持功能可通过设置 cart 服务 DestinationRule 的负载均衡策略实现，以请求中 header 中的 UserID 做一致性 hash 负载均衡。



```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: cart
  namespace: base
spec:
  host: cart
  trafficPolicy:
    loadBalancer:
      consistentHash:
        httpHeaderName: UserID
```

```
  exportTo:
    - '*'
```

配置完成后，可在登录状态多次点击 “Your Cart” 或点击 “ADD TO CART” 调用 cart 服务验证会话保持功能，同一用户的多次请求会被路由至同一个 pod，左下角悬浮窗可查看提供 cart 服务的 pod name。同一用户多次请求的 pod name 不会变化。来自同一用户的多次请求被负载均衡至相同 pod 如下图所示：



使用连接池限制并发

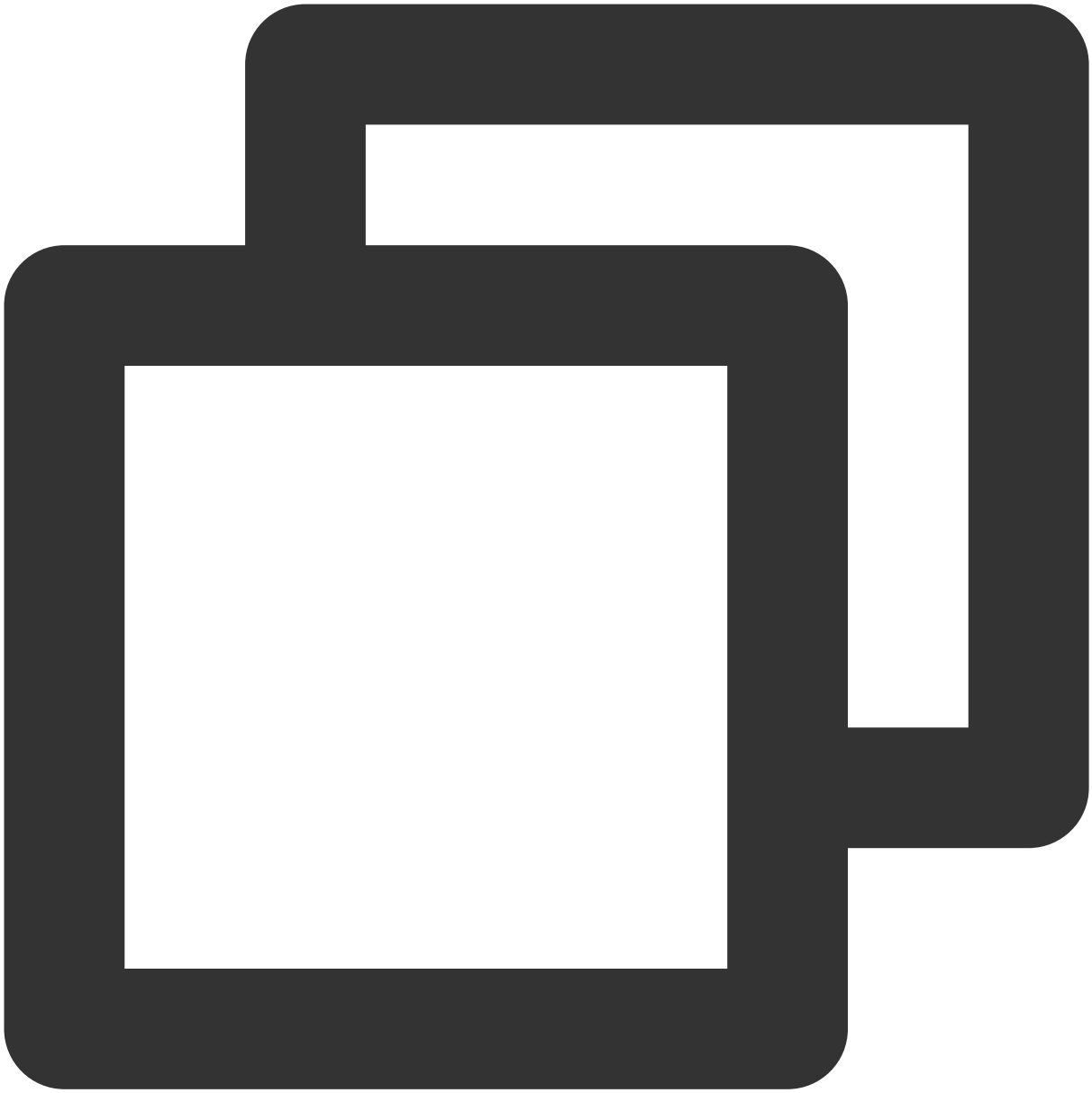
最近更新时间：2023-12-26 11:35:22

操作场景

随着电商网站业务规模的增大，对网站的访问请求并发量开始增加，网站业务人员计划限制服务最大并发数，保证服务运行健壮性。

操作步骤

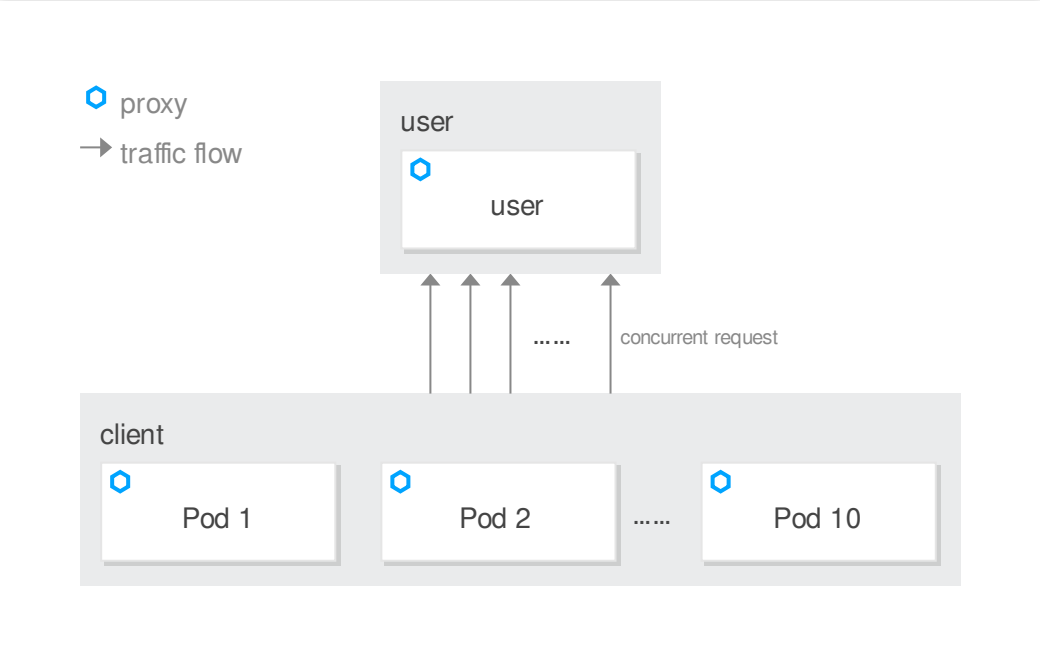
为模拟“高并发”请求场景，首先通过提交以下 YAML 部署 client 服务（10 pods），模拟对 user 服务的高并发请求。



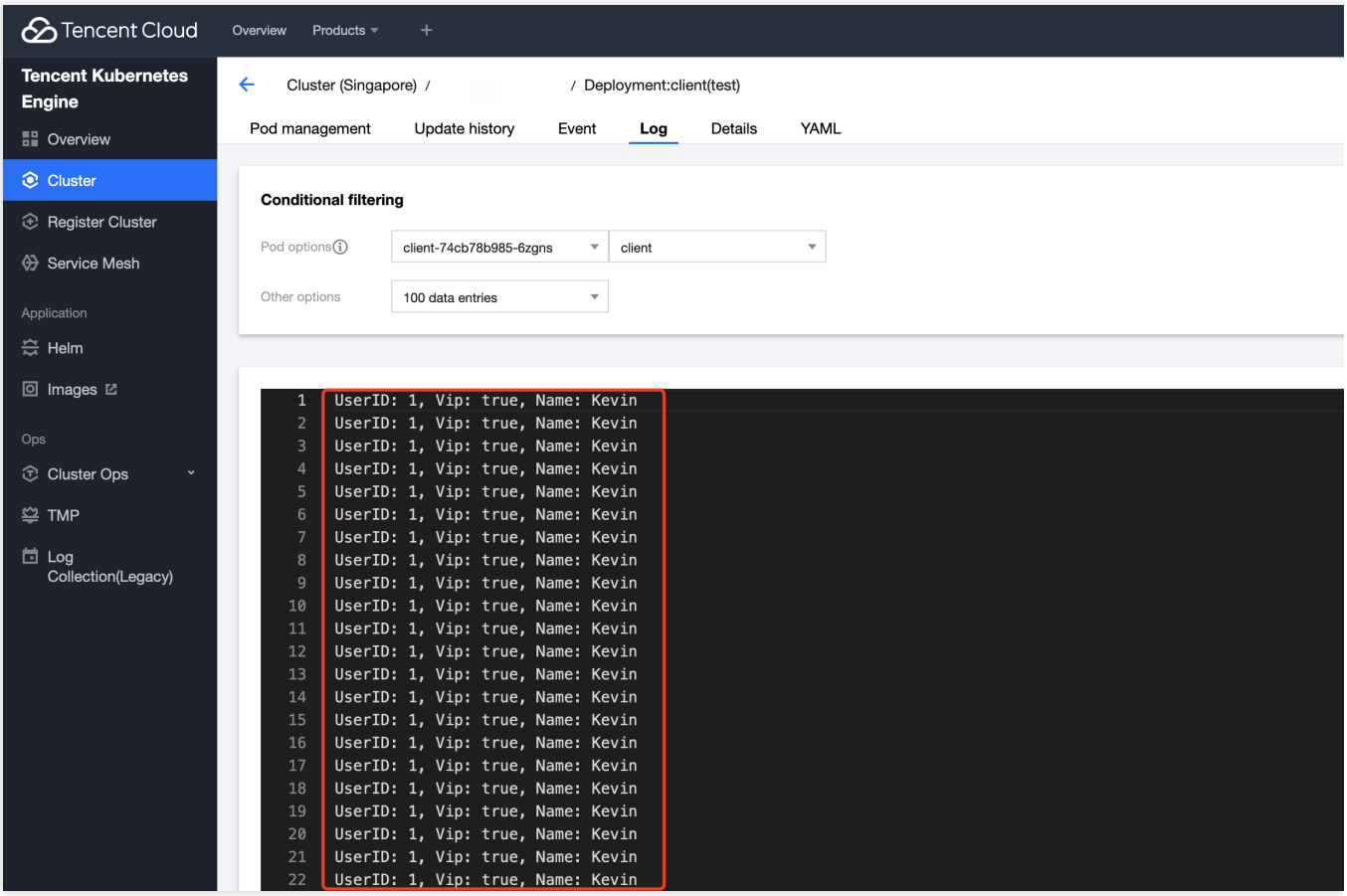
```
apiVersion: v1
kind: Namespace
metadata:
  name: test
  labels:
    istio-injection: enabled
spec:
  finalizers:
    - kubernetes
---
apiVersion: apps/v1
```

```yaml
kind: Deployment
metadata:
  name: client
  namespace: test
  labels:
    app: client
spec:
  replicas: 10
  selector:
    matchLabels:
      app: client
  template:
    metadata:
      labels:
        app: client
    spec:
      containers:
        - name: client
          image: ccr.ccs.tencentyun.com/zhulei/testclient:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneA"
          ports:
            - containerPort: 7000
              protocol: TCP
---

apiVersion: v1
kind: Service
metadata:
  name: client
  namespace: test
  labels:
    app: client
spec:
  ports:
    - name: http
      port: 7000
      protocol: TCP
  selector:
    app: client
  type: ClusterIP
```

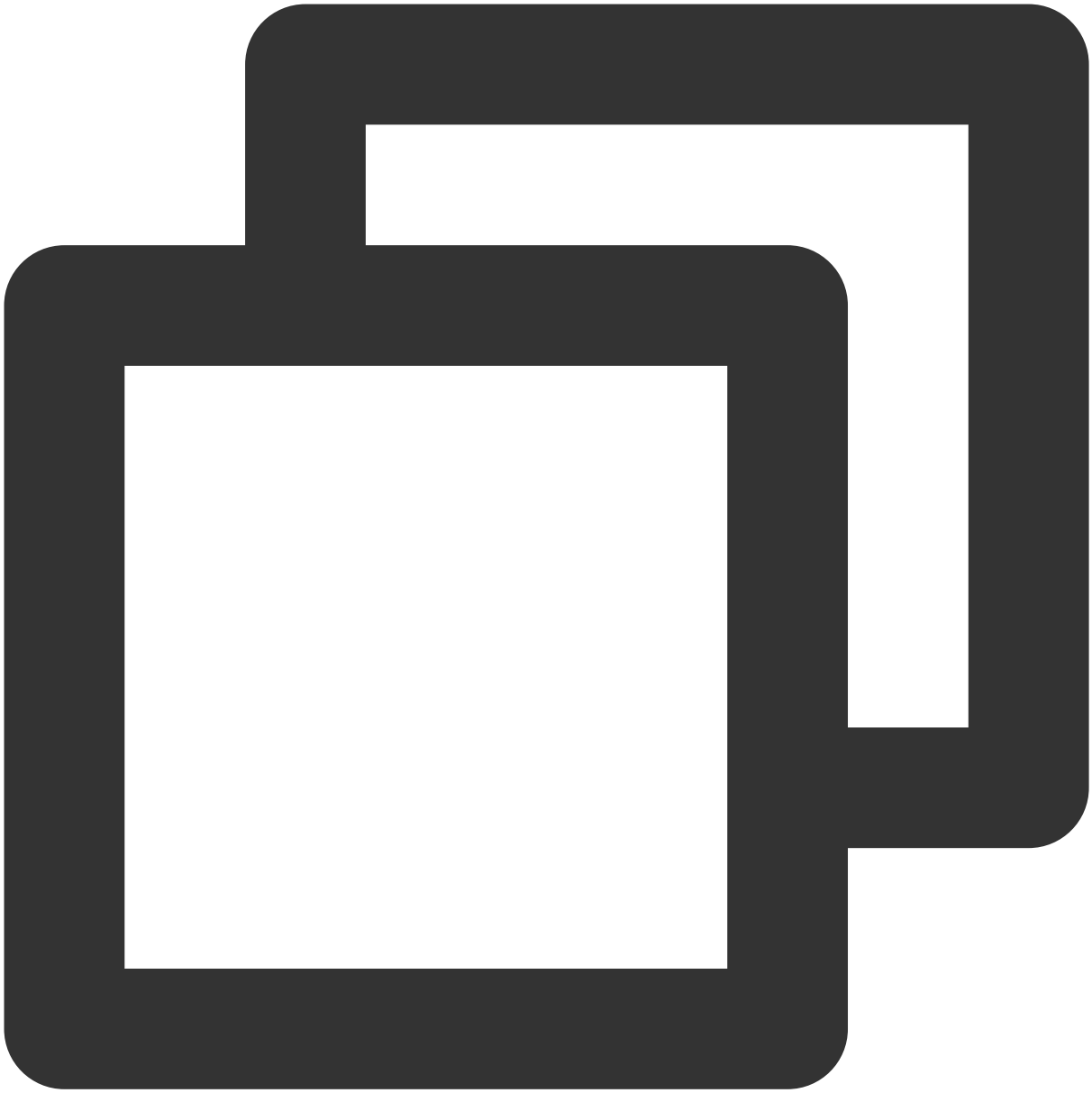
此时对于访问 user 服务没有最大并发数限制，所有请求均可访问成功。通过 TKE 控制台 client deployment 查看 client pod 日志，所有的请求均返回了用户名 Kevin，证明访问请求成功。
高并发请求如下图所示：



所有请求均可访问成功如下图所示：



通过配置 user 服务的 Destination Rule 限制最大并发数为 1：

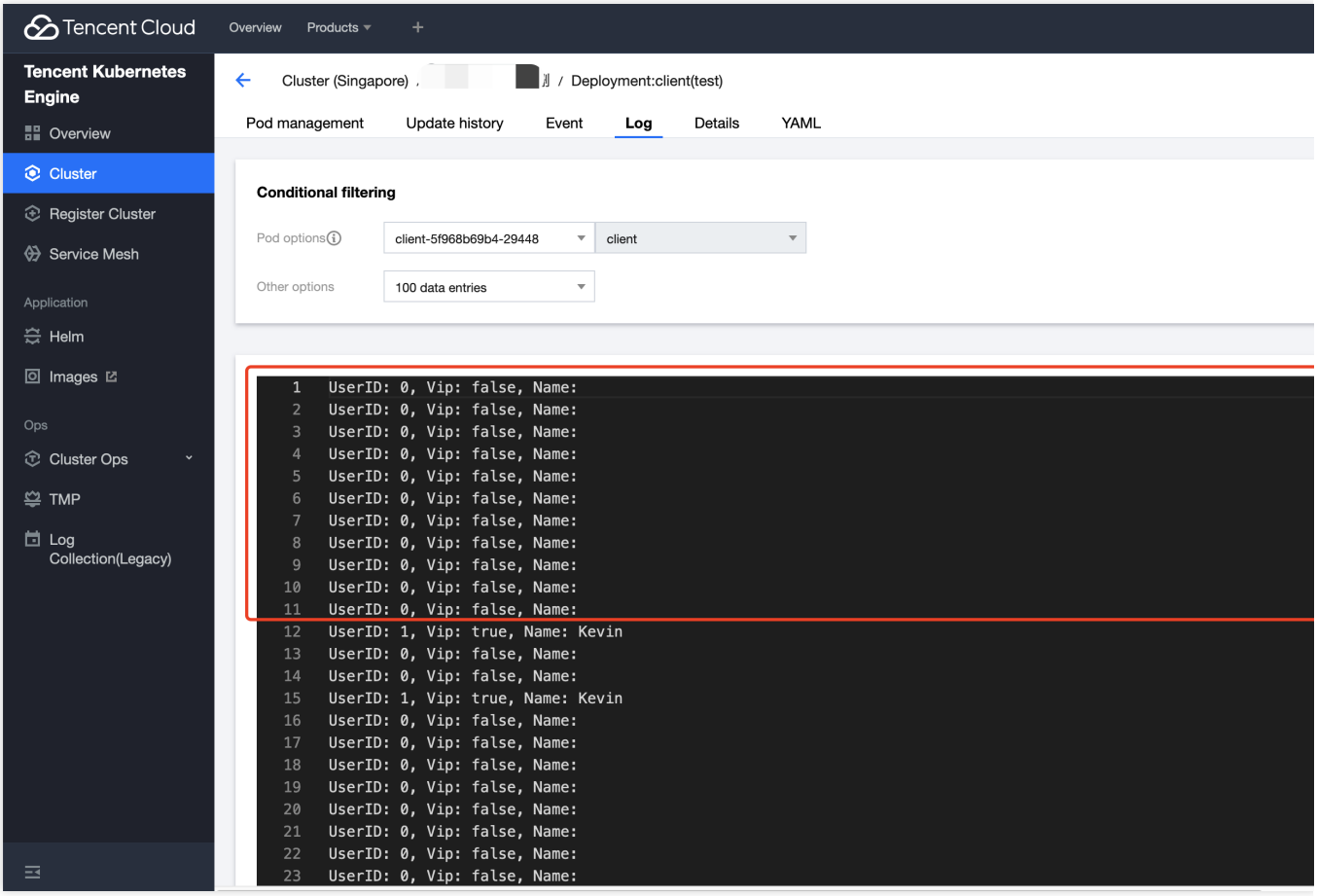


```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: user
  namespace: base
spec:
  host: user
  trafficPolicy:
    connectionPool:
      http:
        http1MaxPendingRequests: 1
```

```
        http2MaxRequests: 1
        maxRequestsPerConnection: 1
  exportTo:
    - '*'
```

此时查看 client pod 日志，部分请求开始出现异常，未返回用户名，请求失败，连接池起到了限制访问服务最大并发数的作用。

部分请求访问失败如下图所示：



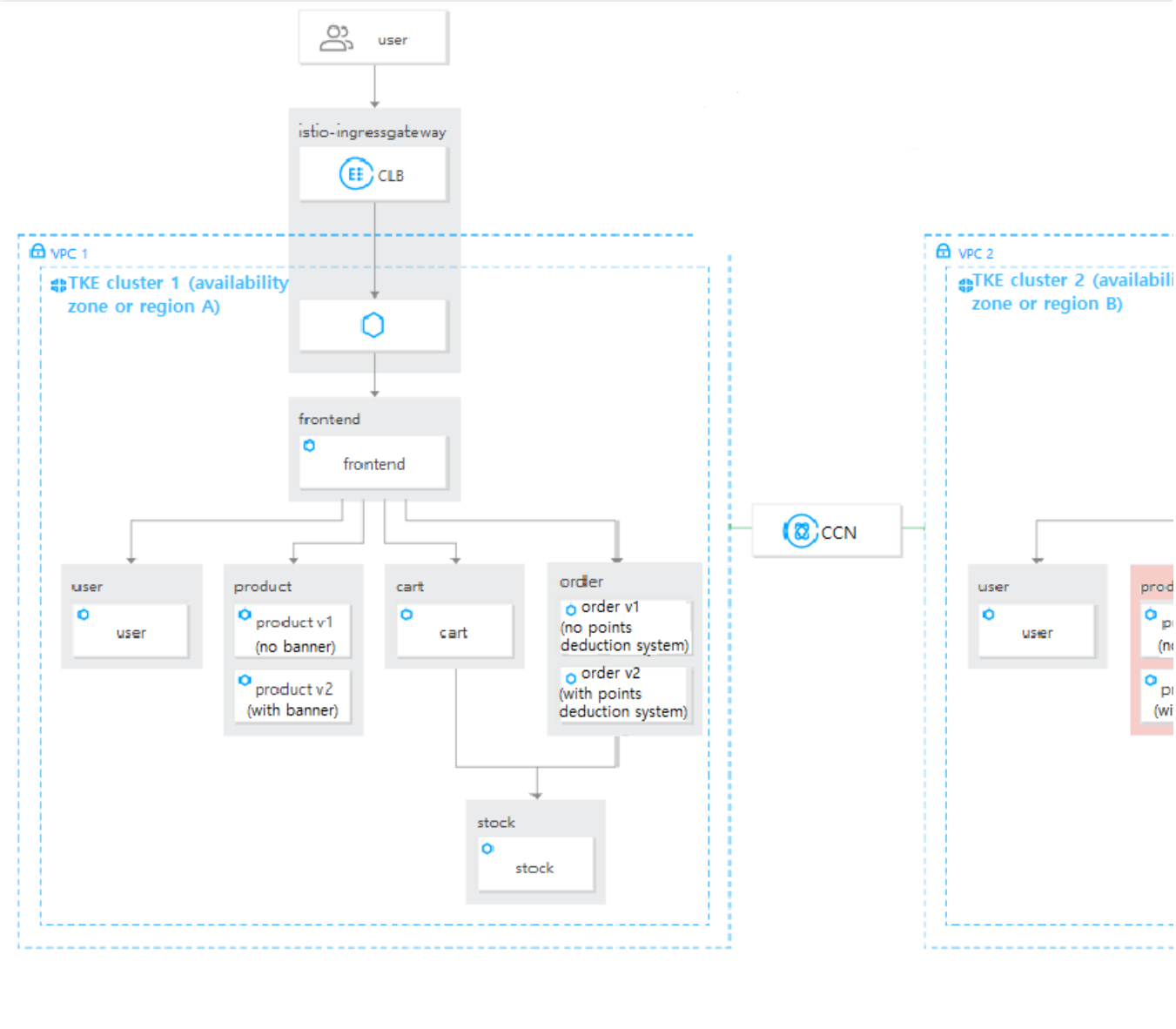
连接池测试完成后，在 user 服务的详情页面删除连接池相关流量策略配置。

多集群容灾多活
服务跨集群容灾

最近更新时间：2023-12-26 11:35:48

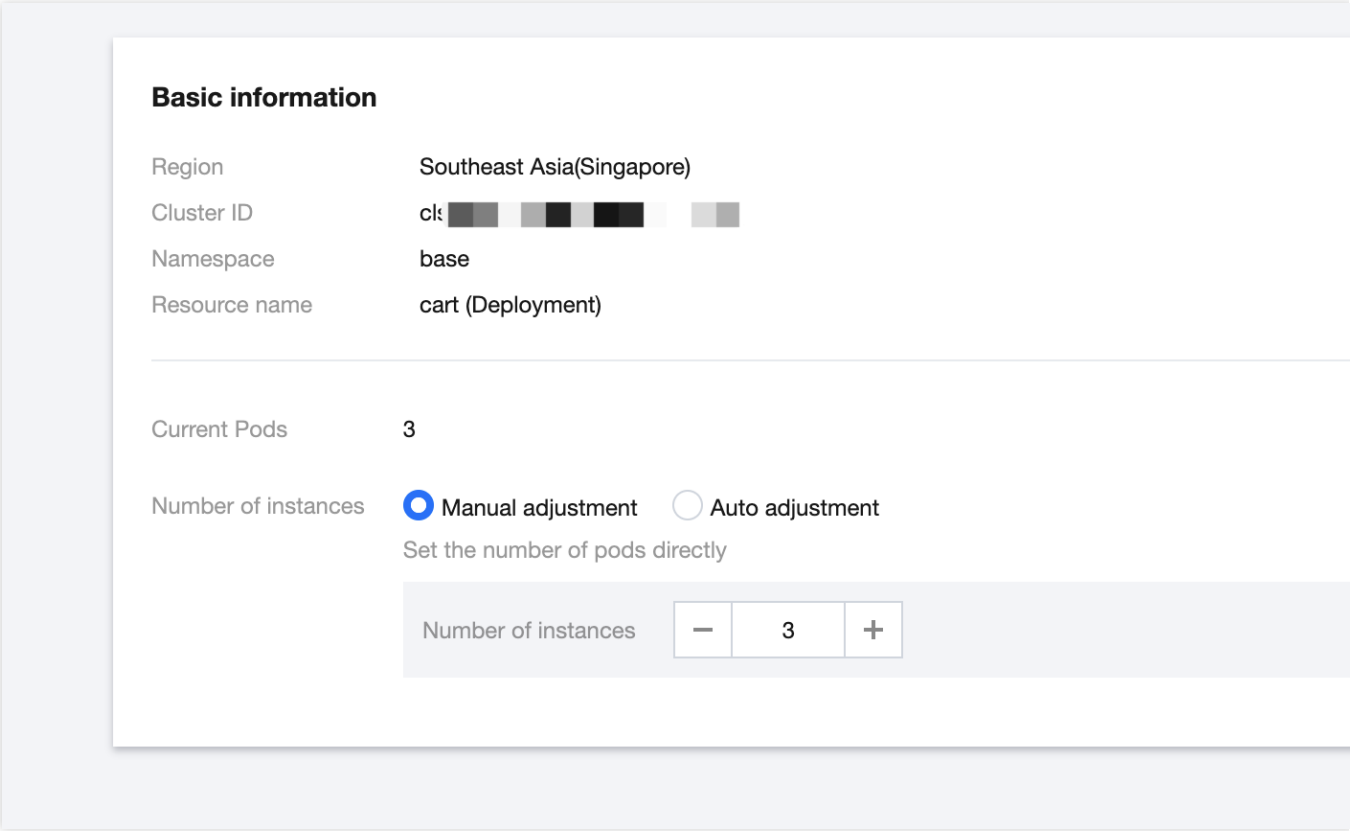
操作场景

网站业务团队希望所有的服务具备跨地域（可用区）容灾的能力，如果其中一个地域（可用区）的服务故障后，访问该服务的流量会被自动切换到另一地域（可用区）。
服务跨地域容灾如下图所示：

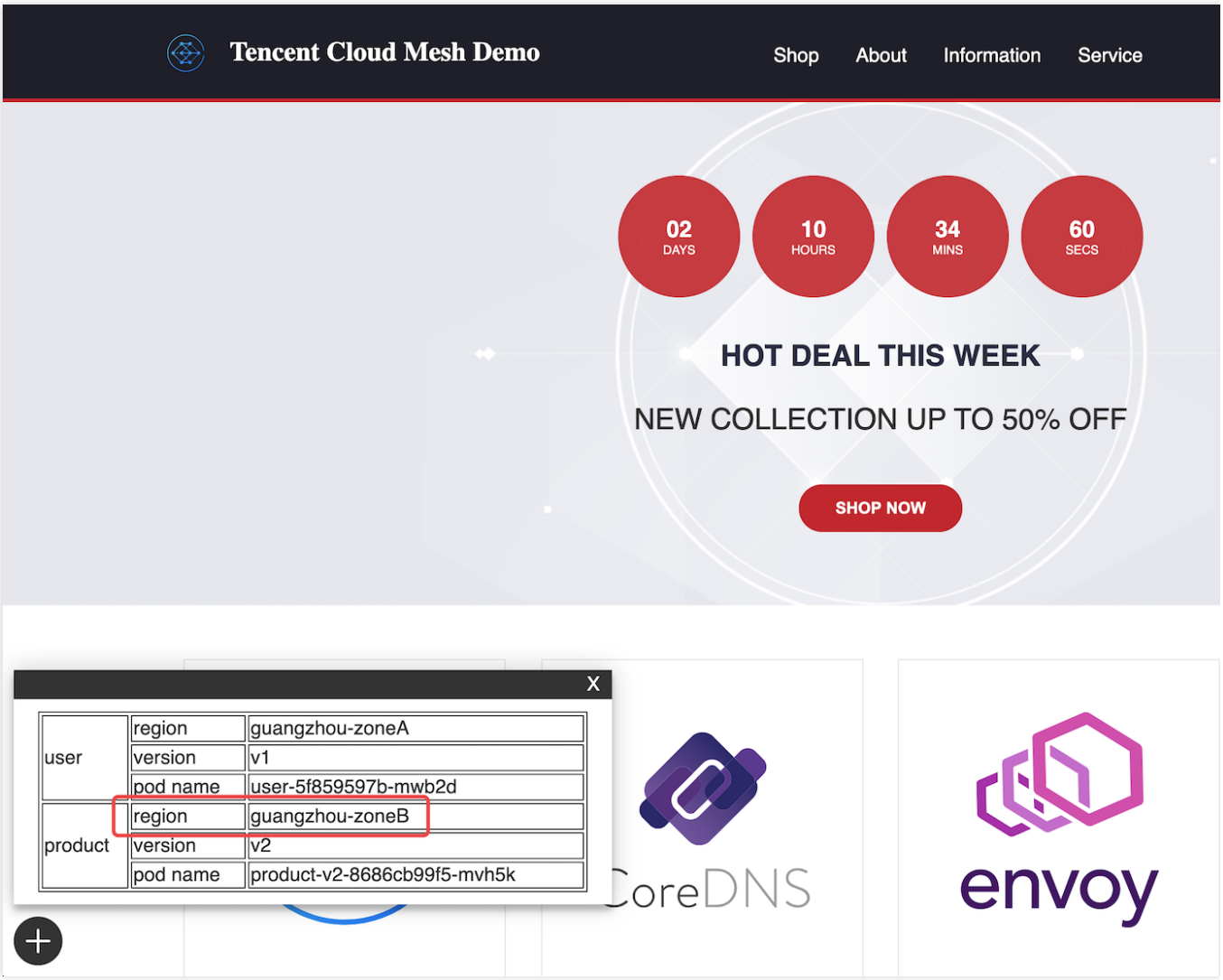


验证操作

TCM 默认提供服务跨地域（可用区）容灾的能力，以下展示的是服务跨可用区容灾的能力验证。
首先，通过 TKE 控制台将主集群的 product-v2 deployment 的 pod 数量配置为 0，模拟主集群所在可用区 product 服务的故障情况。
更新 pod 数量如下图所示：



配置完成后，通过主集群边缘代理网关的 IP 地址访问 Demo 电商网站，即使现在主集群所在的可用区（A）的 product 服务处于故障状态，商品页面还是可以正常访问，通过左下角悬浮窗可查看服务调用情况，当前页面调用的是另一可用区的 product 服务，可用区 A 访问 product 服务的请求被路由到了另一可用区（B）的 product 服务，实现了服务级别的跨可用区容灾。
将两个集群分别部署至不同地域，即可实现服务级别的跨地域容灾。
product 服务跨可用区容灾如下图所示：

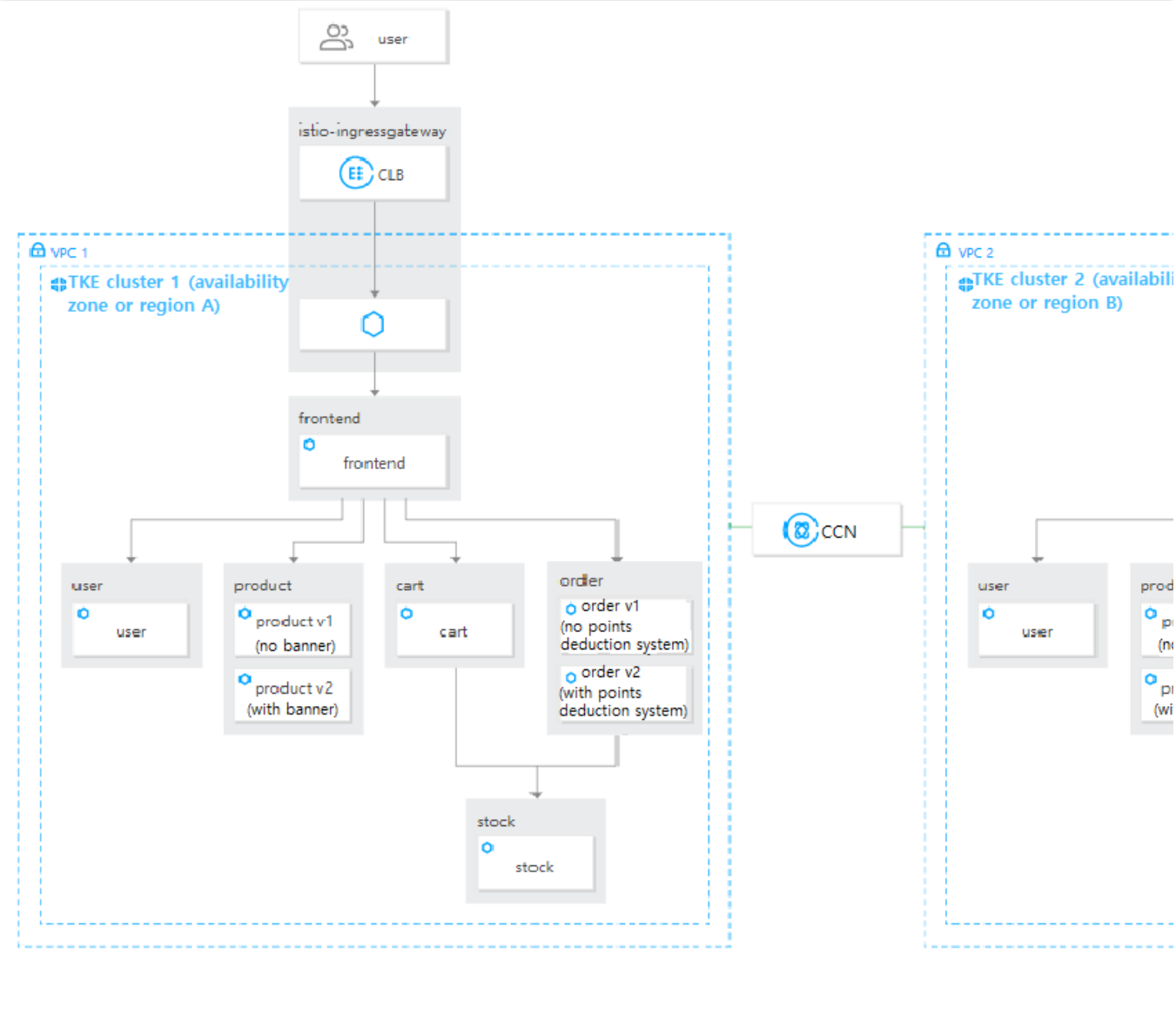

Tencent Cloud Mesh Demo
Shop
About
Information
Service
02 DAYS
10 HOURS
34 MINS
60 SECS
HOT DEAL THIS WEEK
NEW COLLECTION UP TO 50% OFF
SHOP NOW
X
user
region
guangzhou-zoneA
version
v1
pod name
user-5f859597b-mwb2d
product
region
guangzhou-zoneB
version
v2
pod name
product-v2-8686cb99f5-mvh5k
CoreDNS
envoy

地域感知

最近更新时间：2023-12-26 11:36:34

操作场景

业务规模增长，为提高网站可用性，电商网站业务团队计划在另一可用区的集群也部署一套网站业务，两套相同的网站业务在两个不同集群同时为用户提供服务。地域感知如下图所示：



操作步骤

两套网站业务在正常运行的情况下，ingress gateway 会优先将流量路由至本地域或可用区的 frontend 服务，即使另一集群中也有 frontend 服务；frontend 服务会优先就近访问相同可用区 user，product，order，cart 服务；order 和 cart 服务也会优先就近访问相同可用区的 stock 服务。

在 Kubernetes 中，Pod 的地域是通过在已部署的节点上的 Region 和 Zone 的标签决定的，Demo 应用 yaml 中已为工作负载设置了相应的 zone 标签，首先将网站全套服务也部署至另一可用区的集群（子集群）：

```
apiVersion: v1
kind: Namespace
metadata:
  name: base
```

```
  labels:
    istio.io/rev: 1-6-9
spec:
  finalizers:
    - kubernetes
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  namespace: base
  labels:
    app: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
          image: ccr.ccs.tencentyun.com/chloeyhuang/demo:v202007101540
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 80
---

apiVersion: v1
kind: Service
metadata:
  name: frontend
  namespace: base
  labels:
    app: frontend
spec:
  ports:
```

```
    - port: 80
      name: http
  selector:
    app: frontend
---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: product-v1
  namespace: base
  labels:
    app: product
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: product
      version: v1
  template:
    metadata:
      labels:
        app: product
        version: v1
    spec:
      containers:
        - name: product
          image: ccr.ccs.tencentyun.com/zhulei/testproduct1:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: product-v2
  namespace: base
  labels:
```

```
    app: product
    version: v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: product
      version: v2
  template:
    metadata:
      labels:
        app: product
        version: v2
    spec:
      containers:
        - name: product
          image: ccr.ccs.tencentyun.com/zhulei/testproduct2:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
---

apiVersion: v1
kind: Service
metadata:
  name: product
  namespace: base
  labels:
    app: product
spec:
  ports:
    - port: 7000
      name: http
  selector:
    app: product
---

apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: user
  namespace: base
  labels:
    app: user
spec:
  replicas: 1
  selector:
    matchLabels:
      app: user
  template:
    metadata:
      labels:
        app: user
    spec:
      containers:
        - name: user
          image: ccr.ccs.tencentyun.com/zhulei/testuser:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
---

apiVersion: v1
kind: Service
metadata:
  name: user
  namespace: base
  labels:
    app: user
spec:
  ports:
    - port: 7000
      name: http
  selector:
    app: user
---

apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: stock
  namespace: base
  labels:
    app: stock
spec:
  replicas: 1
  selector:
    matchLabels:
      app: stock
  template:
    metadata:
      labels:
        app: stock
    spec:
      containers:
        - name: stock
          image: ccr.ccs.tencentyun.com/zhulei/teststock:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
---

apiVersion: v1
kind: Service
metadata:
  name: stock
  namespace: base
  labels:
    app: stock
spec:
  ports:
    - port: 7000
      name: http
  selector:
    app: stock
---

apiVersion: apps/v1
kind: Deployment
metadata:
```

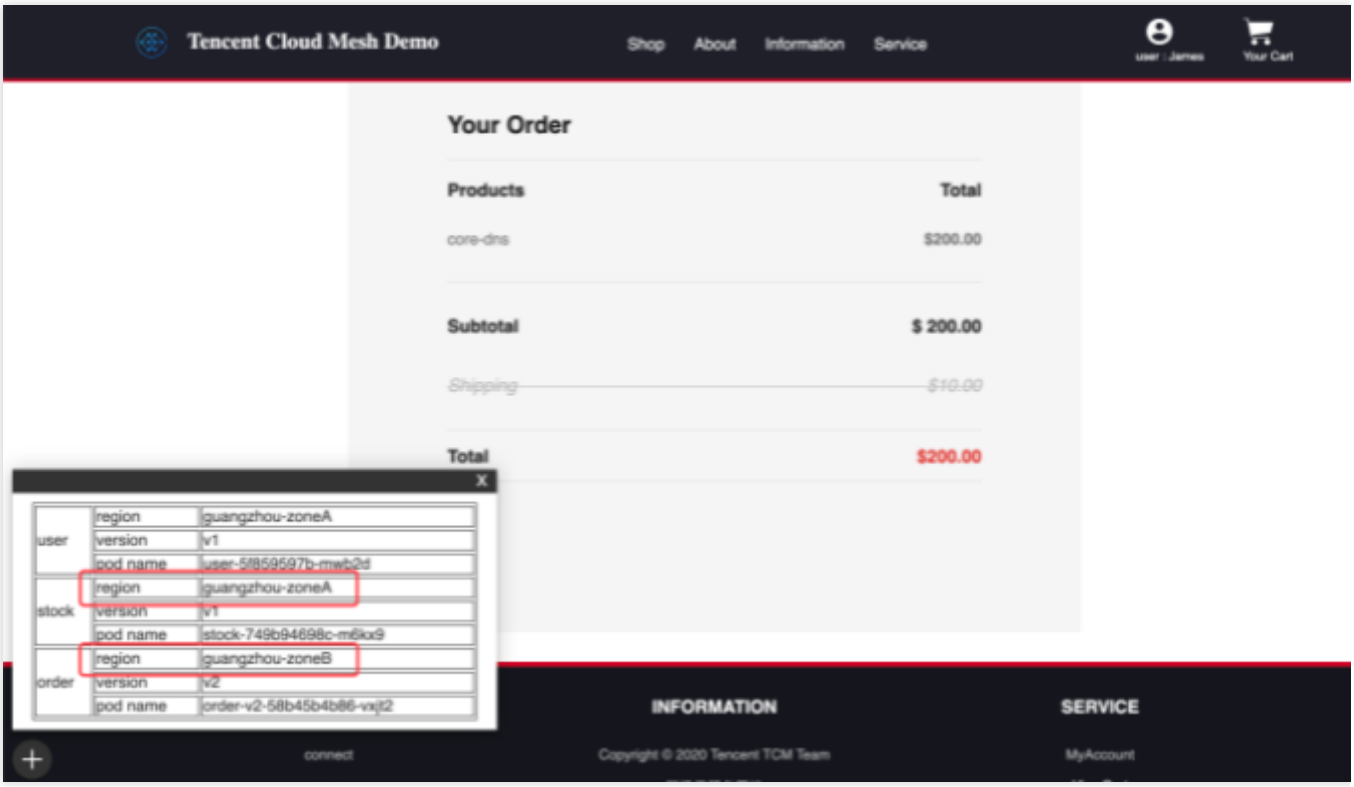
```
  name: cart
  namespace: base
  labels:
    app: cart
spec:
  replicas: 3
  selector:
    matchLabels:
      app: cart
  template:
    metadata:
      labels:
        app: cart
    spec:
      containers:
        - name: cart
          image: ccr.ccs.tencentyun.com/zhulei/testcart:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
              protocol: TCP
---

apiVersion: v1
kind: Service
metadata:
  name: cart
  namespace: base
  labels:
    app: cart
spec:
  ports:
    - name: http
      port: 7000
      protocol: TCP
  selector:
    app: cart
  type: ClusterIP
---
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: order-v1
  namespace: base
  labels:
    app: order
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: order
      version: v1
  template:
    metadata:
      labels:
        app: order
        version: v1
    spec:
      containers:
        - name: order
          image: ccr.ccs.tencentyun.com/zhulei/testorder1:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
              protocol: TCP
---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: order-v2
  namespace: base
  labels:
    app: order
    version: v2
spec:
  replicas: 1
  selector:
```

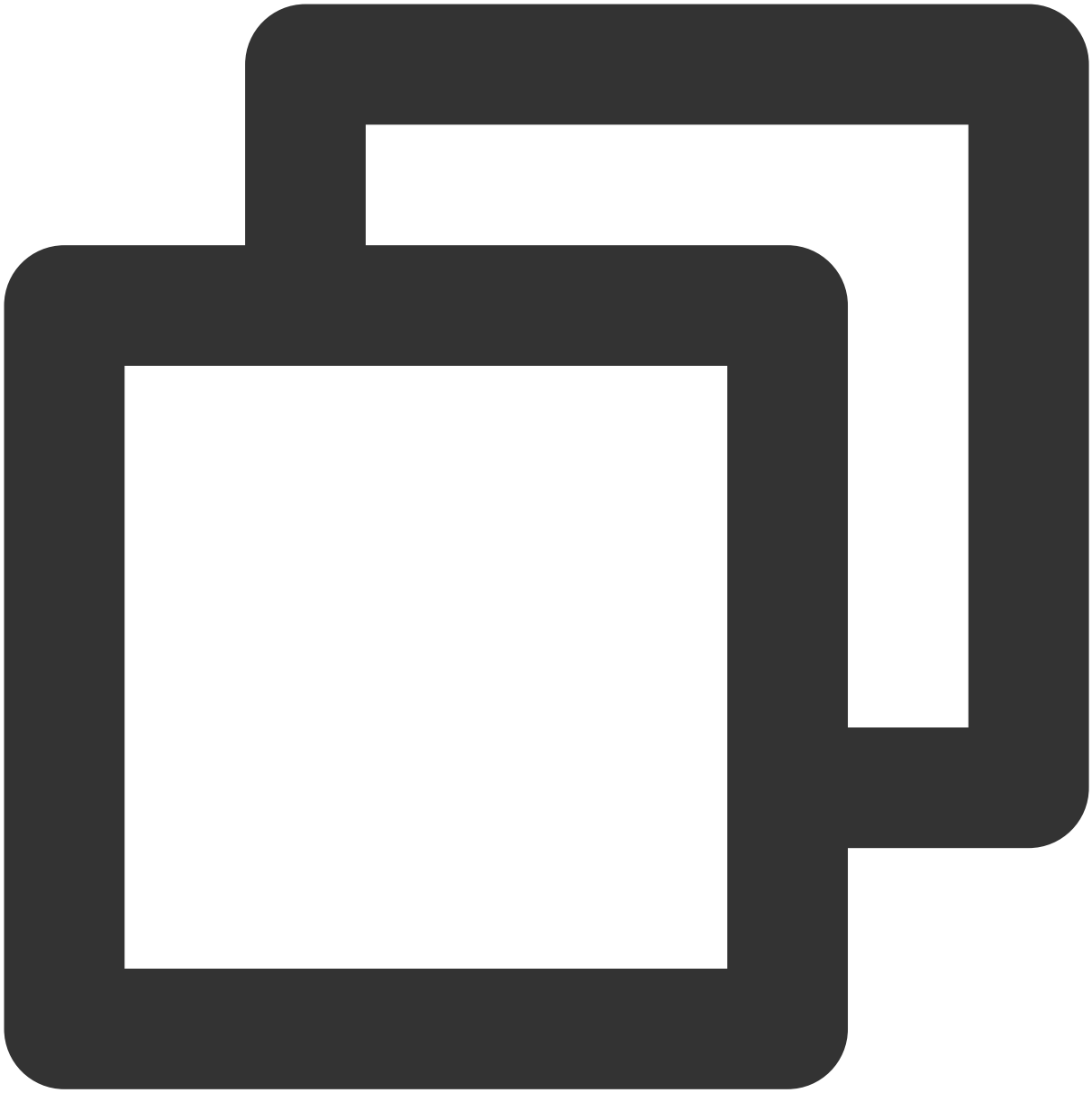
```
    matchLabels:
      app: order
      version: v2
  template:
    metadata:
      labels:
        app: order
        version: v2
    spec:
      containers:
        - name: order
          image: ccr.ccs.tencentyun.com/zhulei/testorder2:v1
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: REGION
              value: "guangzhou-zoneB"
          ports:
            - containerPort: 7000
              protocol: TCP
---

apiVersion: v1
kind: Service
metadata:
  name: order
  namespace: base
  labels:
    app: order
spec:
  ports:
    - name: http
      port: 7000
      protocol: TCP
  selector:
    app: order
  type: ClusterIP
```

部署配置完成后，未配置健康检查时，地域感知不生效，两个可用区的某一服务调用另外的服务是随机访问，不会按照就近访问的原则。

order 服务调用不同可用区的 stock 服务如下图所示：



要开启服务访问的地域感知，需要配置所有服务的健康检查功能，通过将以下 yaml 文件提交至主集群实现：



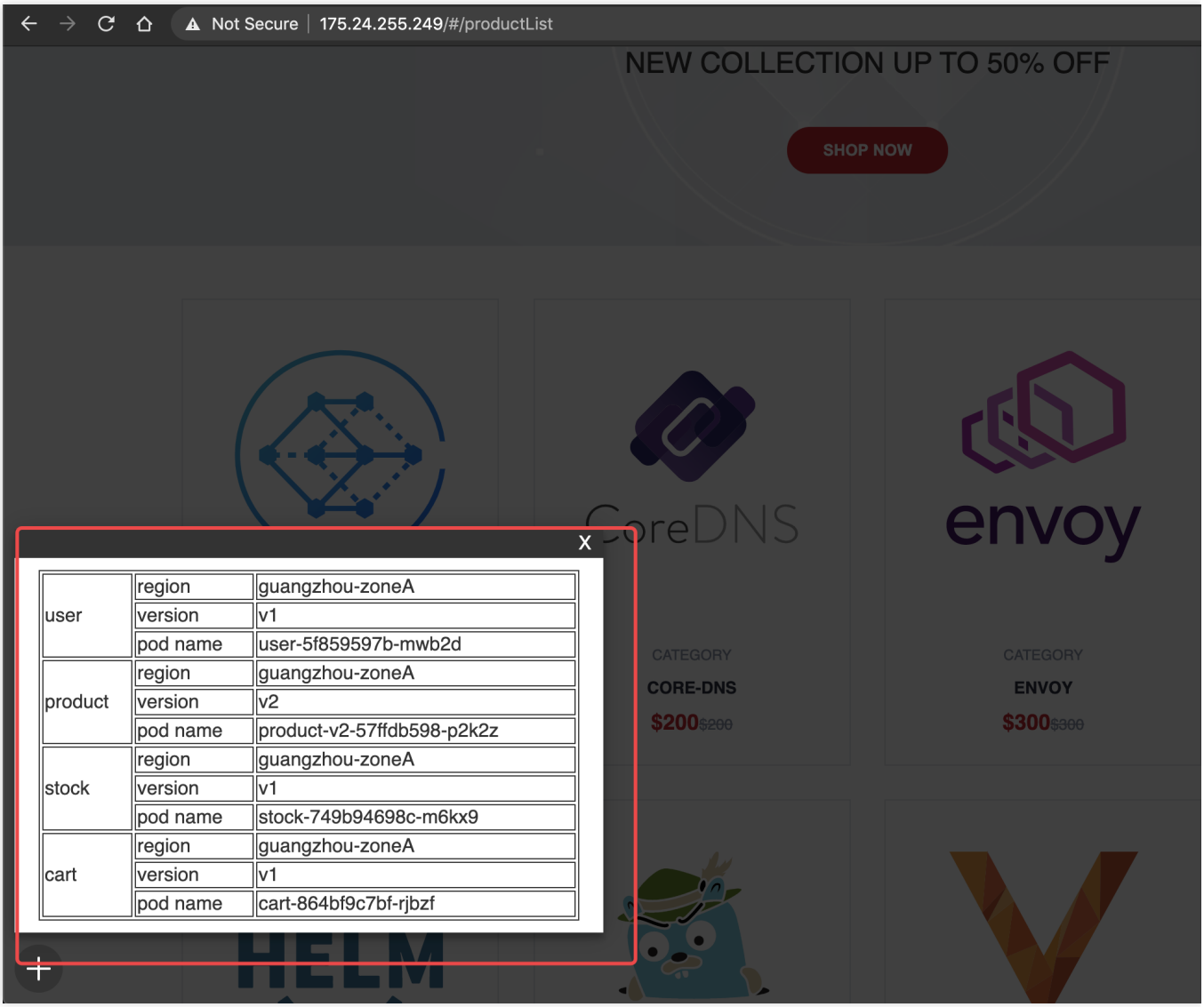
```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: cart
  namespace: base
spec:
  host: cart
  trafficPolicy:
    loadBalancer:
      consistentHash:
        httpHeaderName: UserID
```

```
    outlierDetection:
      consecutiveErrors: 5
      interval: 10000ms
      baseEjectionTime: 30000ms
      maxEjectionPercent: 10
      minHealthPercent: 50
  exportTo:
    - '*'

---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: frontend
  namespace: base
spec:
  host: frontend
  trafficPolicy:
    outlierDetection:
      consecutiveErrors: 5
      interval: 10000ms
      baseEjectionTime: 30000ms
      maxEjectionPercent: 10
      minHealthPercent: 50
  exportTo:
    - '*'

---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: order
  namespace: base
spec:
  host: order
  trafficPolicy:
    outlierDetection:
      consecutiveErrors: 5
      interval: 10000ms
      baseEjectionTime: 30000ms
      maxEjectionPercent: 10
      minHealthPercent: 50
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
```

```
      labels:
        version: v2
  exportTo:
    - '*'

---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: product
  namespace: base
spec:
  host: product
  trafficPolicy:
    outlierDetection:
      consecutiveErrors: 5
      interval: 10000ms
      baseEjectionTime: 30000ms
      maxEjectionPercent: 10
      minHealthPercent: 50
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2

---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: stock
  namespace: base
spec:
  host: stock
  trafficPolicy:
    outlierDetection:
      consecutiveErrors: 5
      interval: 10000ms
      baseEjectionTime: 30000ms
      maxEjectionPercent: 10
      minHealthPercent: 50
  exportTo:
    - '*'

---
```

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: user
  namespace: base
spec:
  host: user
  trafficPolicy:
    outlierDetection:
      consecutiveErrors: 5
      interval: 10000ms
      baseEjectionTime: 30000ms
      maxEjectionPercent: 10
      minHealthPercent: 50
  exportTo:
    - '*'
```

健康检查配置完成后，由可用区 A 集群的边缘代理网关访问网站服务，浏览商品页面，添加购物车，下单等操作，可用区 A 的边缘代理网关会将流量路由至相同可用区的前端 frontend 服务，前端服务也会地域感知就近调用同一可用区的 user、cart、order、stock 服务；通过可用区 B 的边缘代理网关访问网站业务，请求路由至可用区 B 的前端服务，可用区 B 服务也会就近调用相同可用区的服务。通过 Demo 页面左下角悬浮窗可以查看当前调用服务的可用区信息。地域感知如下图所示：



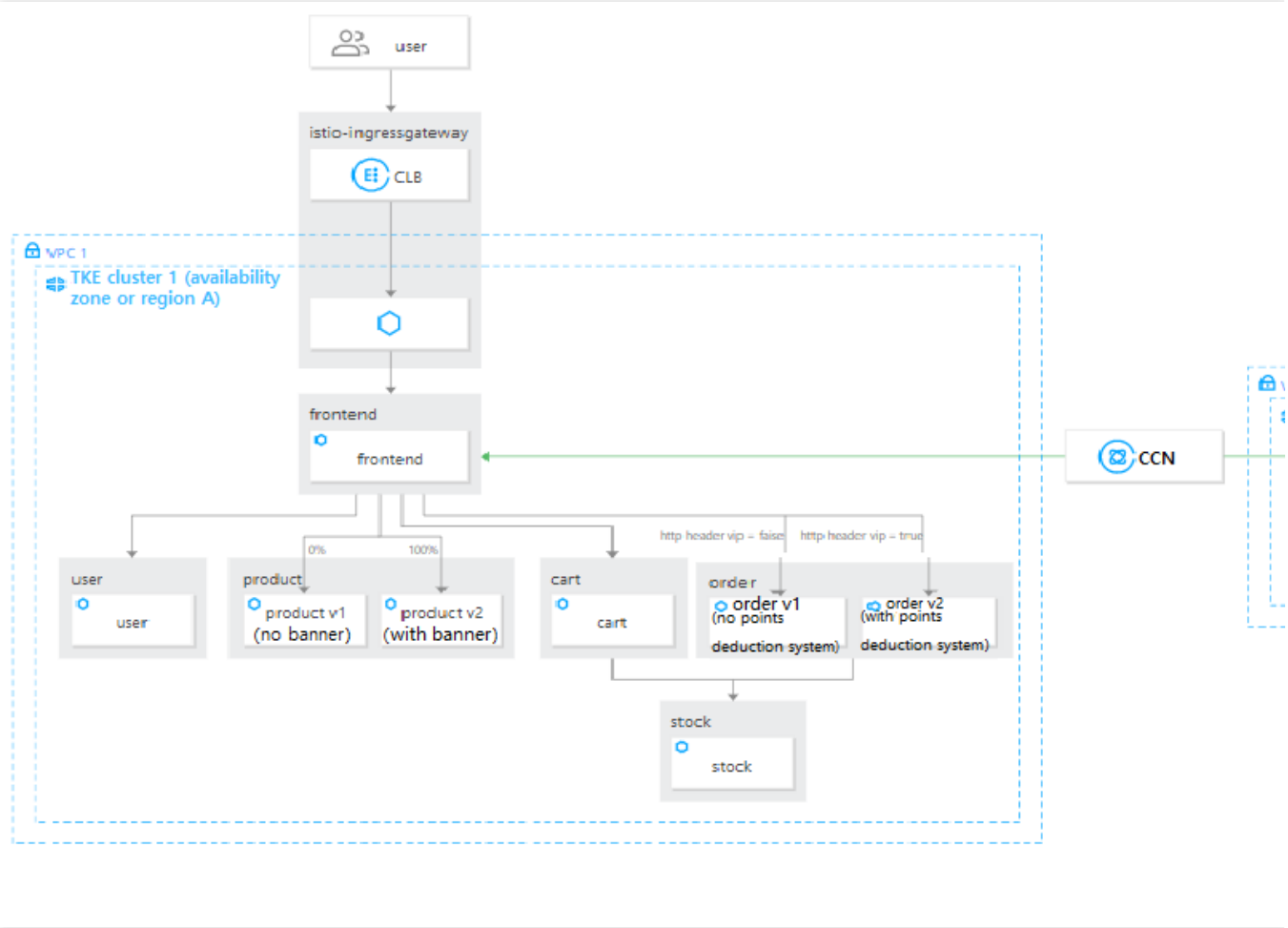
user	region	guangzhou-zoneA
	version	v1
	pod name	user-5f859597b-mwb2d
product	region	guangzhou-zoneA
	version	v2
	pod name	product-v2-57ffdb598-p2k2z
stock	region	guangzhou-zoneA
	version	v1
	pod name	stock-749b94698c-m6kx9
cart	region	guangzhou-zoneA
	version	v1
	pod name	cart-864bf9c7bf-rjbzf

就近接入

最近更新时间：2023-12-26 11:37:25

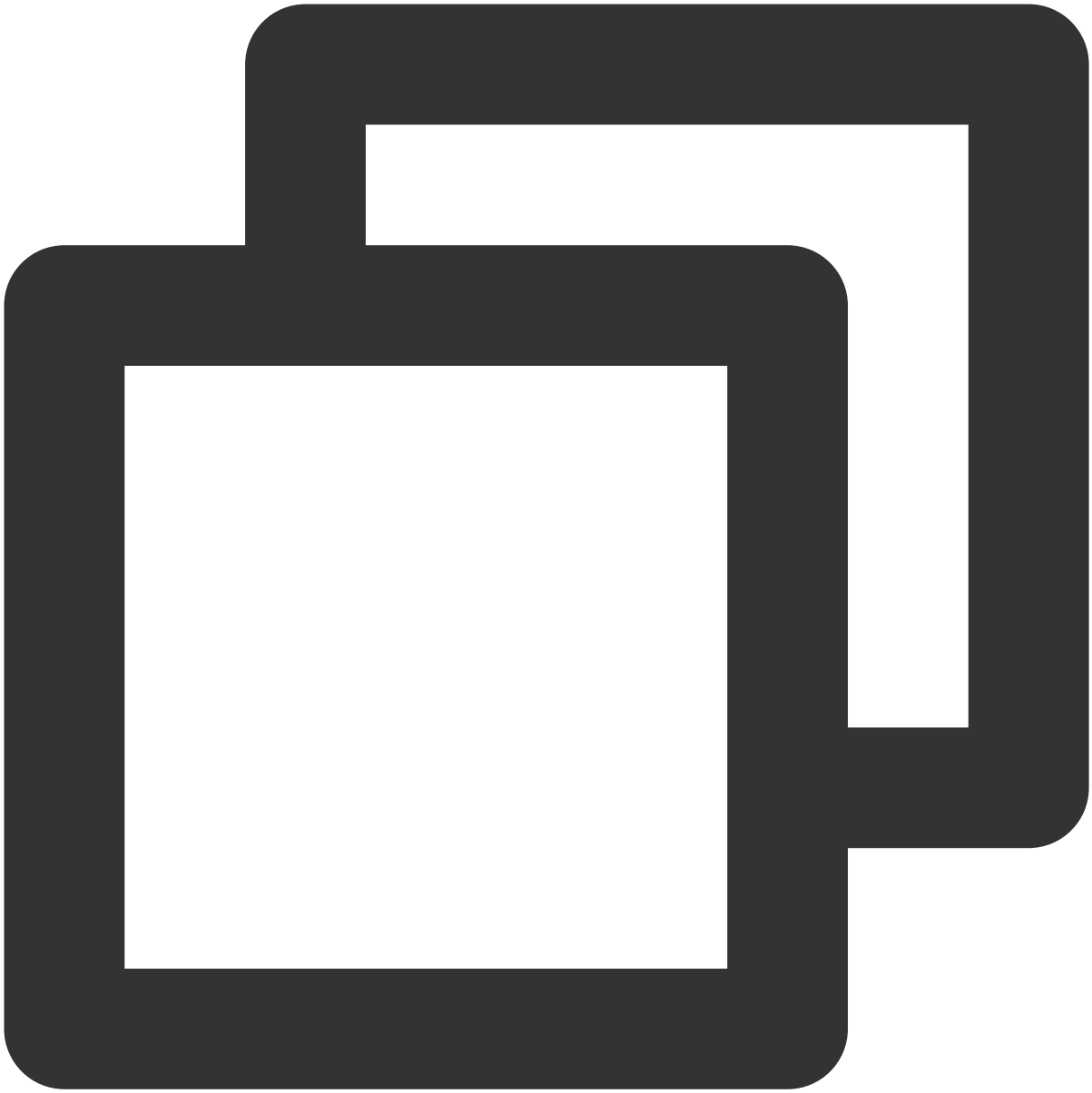
操作场景

随着电商网站业务规模的扩张，网站的业务需要快速部署至另一地域/可用区的集群，从另一集群的网关也能访问电商网站业务。此时无需在另一集群部署整套业务，只需在网格管理的另一个集群中部署边缘代理网关并配置好监听规则，即可以另一集群为入口访问电商网站业务。就近接入如下图所示：



操作步骤

应用以下配置到主集群，gateway 配置集群 2（子集群）的边缘代理网关的监听器规则，放通 80 端口，http 协议，VirtualService 配置将来自边缘代理网关的流量路由至 frontend 服务。

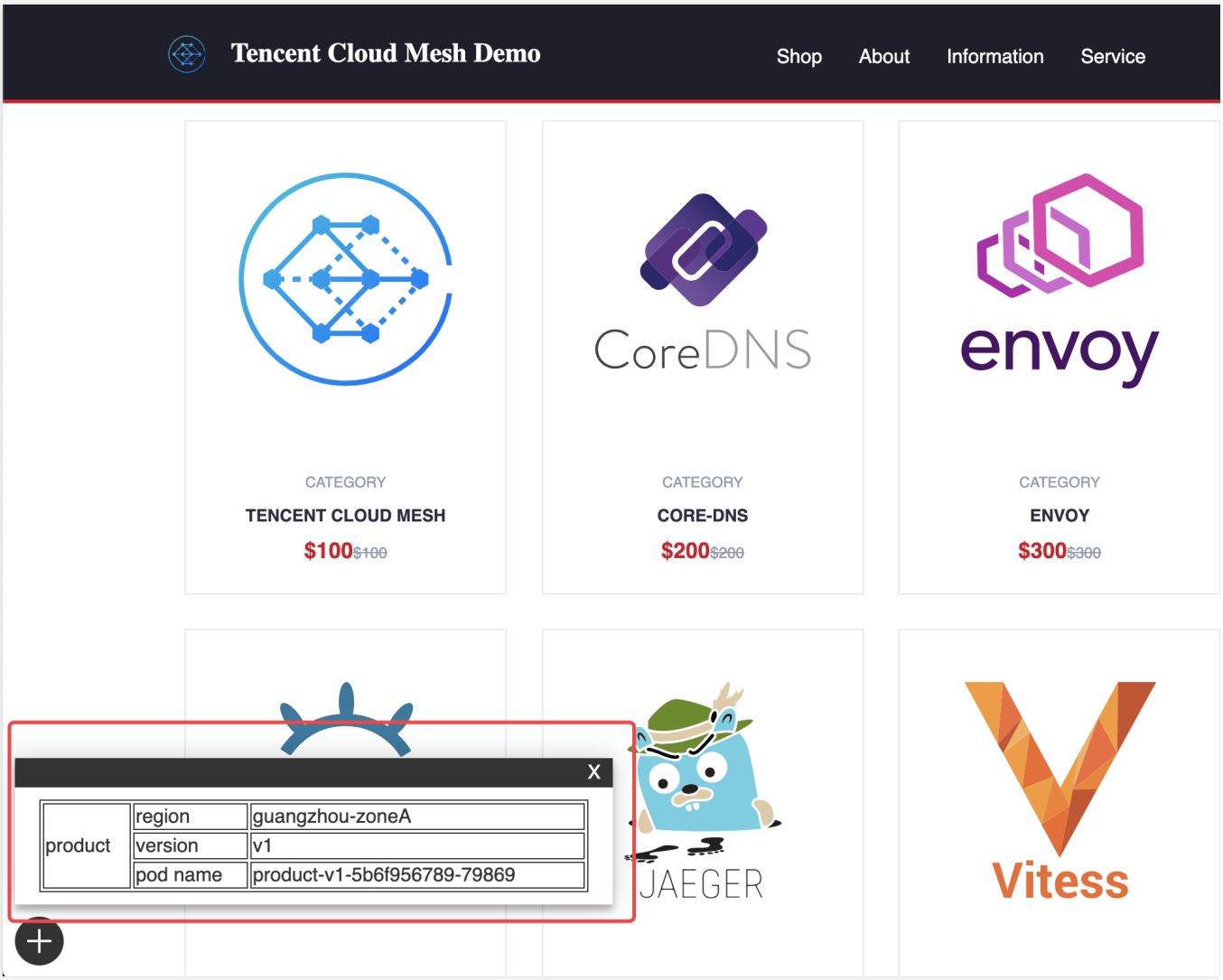


```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: frontend-gw-2
  namespace: base
spec:
  servers:
    - port:
        number: 80
        name: http
        protocol: HTTP
```

```
      hosts:
        - '*'
  selector:
    app: istio-ingressgateway-1
    istio: ingressgateway

---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: frontend-vs
  namespace: base
spec:
  hosts:
    - '*'
  gateways:
    - base/frontend-gw
    - base/frontend-gw-2
  http:
    - route:
        - destination:
            host: frontend.base.svc.cluster.local
```

配置完成后，通过集群 2（子集群）的边缘代理网关 IP 地址即可访问到电商网站，即使上海的集群中没有部署电商网站相关服务，因为流量被路由到了主集群的服务。

来自子集群的请求就近接入后访问部署在主集群的服务如下图所示：


Tencent Cloud Mesh Demo
Shop
About
Information
Service
CoreDNS
envoy
CATEGORY
TENCENT CLOUD MESH
$100$100
CATEGORY
CORE-DNS
$200$200
CATEGORY
ENVOY
$300$300
X
product
region
guangzhou-zoneA
version
v1
pod name
product-v1-5b6f956789-79869
JAEGER
Vitess

安全加固
认证

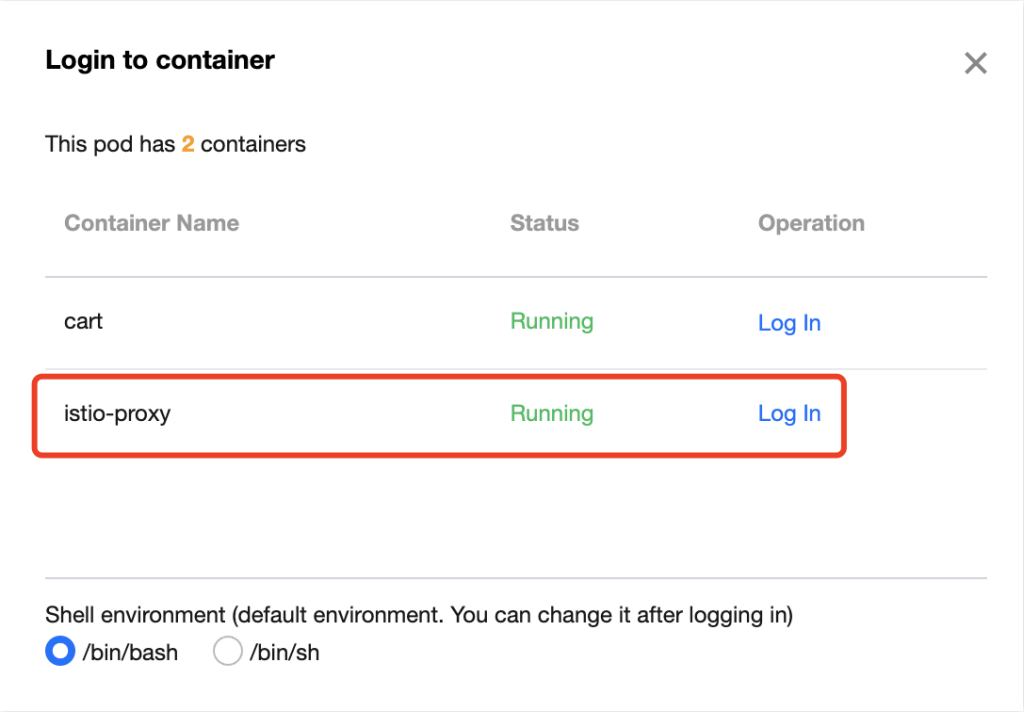
最近更新时间：2023-12-26 11:38:01

操作场景

电商网站业务团队希望限制访问生产环境（base namespace）下所有服务间的访问必须开启双向认证 mTLS，以防御中间人攻击。

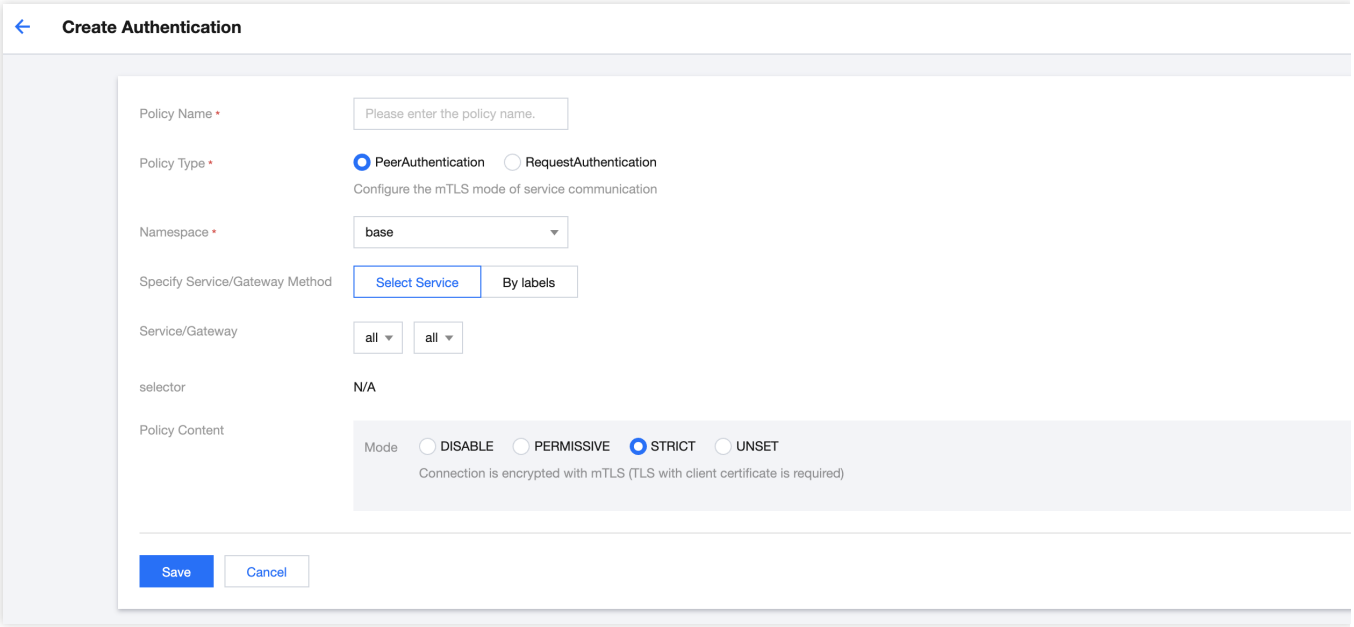
操作步骤

服务间通信的 mTLS 模式默认为 PERMISSIVE 宽容模式，即服务间的通信既可以使用 mTLS 加密，也可以使用 plaintext 明文连接。
此时在 TKE 控制台登录 client 的 istio-proxy 容器，使用明文连接对生产环境（base namespace）product 服务发起请求：`curl http://product.base.svc.cluster.local:7000/product`，此时明文连接也可正常访问 product 服务。如下图所示：

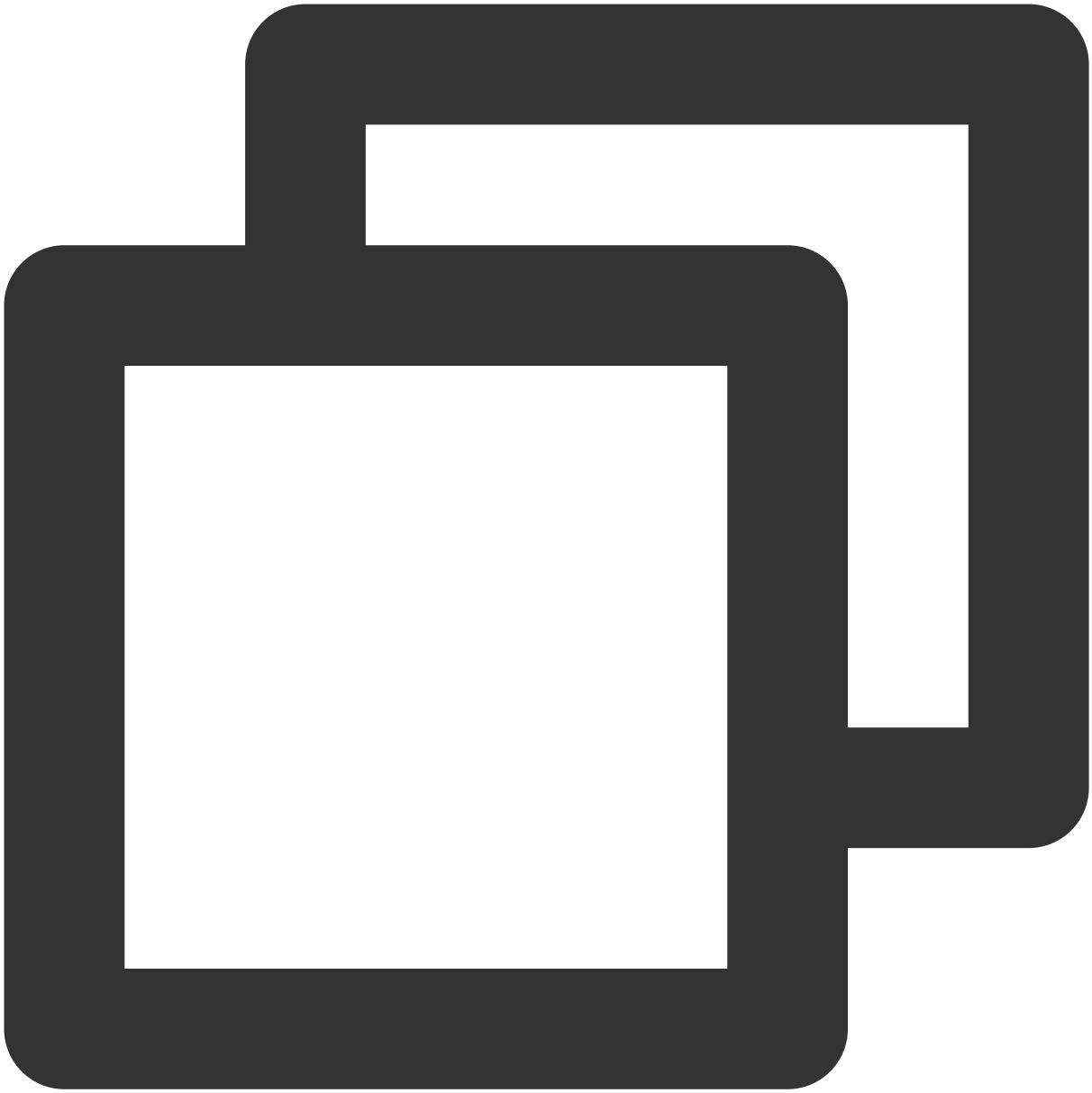


明文连接访问成功如下图所示：

限制 base namespace 下的服务间通信必须采用 mTLS 模式可以通过 PeerAuthentication 策略设置 mTLS 模式为 STRICT 完成。如下图所示：



也可以通过 kubectl 提交 YAML 文件至主集群完成配置：



```
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: base-strict
  namespace: base
spec:
  mtls:
    mode: STRICT
```

配置完成后，在 TKE 控制台登录 client 的 istio-proxy 容器，使用明文连接对生产环境（base namespace）product 服务发起请求： `curl http://product.base.svc.cluster.local:7000/product` ，此时明文连接访问失败。如下图所示：

授权

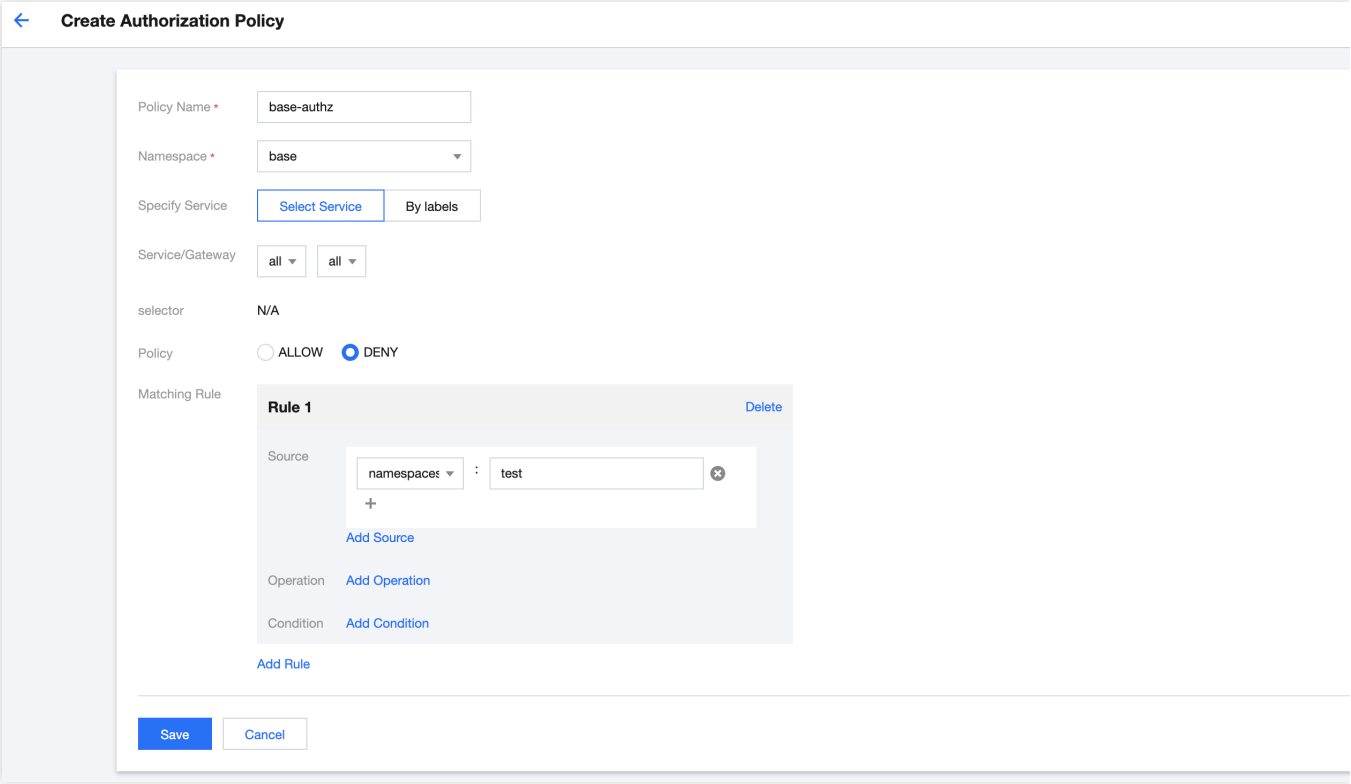
最近更新时间：2023-12-26 11:38:34

操作场景

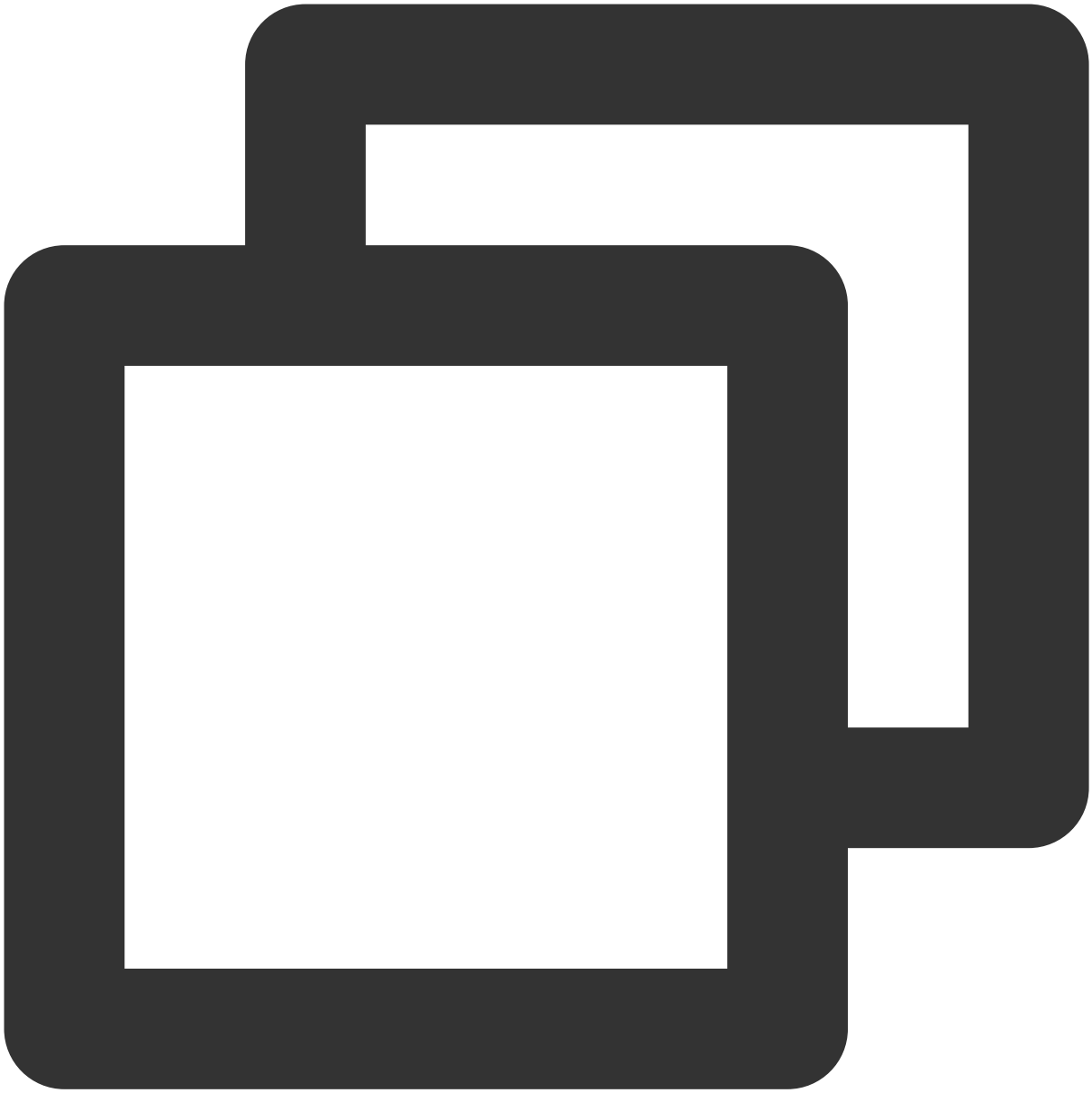
生产环境（base namespace）的服务已经稳定运行，电商网站业务团队希望对网格中的服务做权限控制，限制生产环境（base namespace）下的服务不能被测试环境（test namesapce）下的服务访问。

操作步骤

对网格中的服务进行权限控制可通过配置 AuthorizationPolicy 实现，配置以下 AuthorizationPolicy 策略限制 base namespace 下所有服务不能被 test namespace 下的服务访问。
通过控制台配置授权规则，如下图所示：

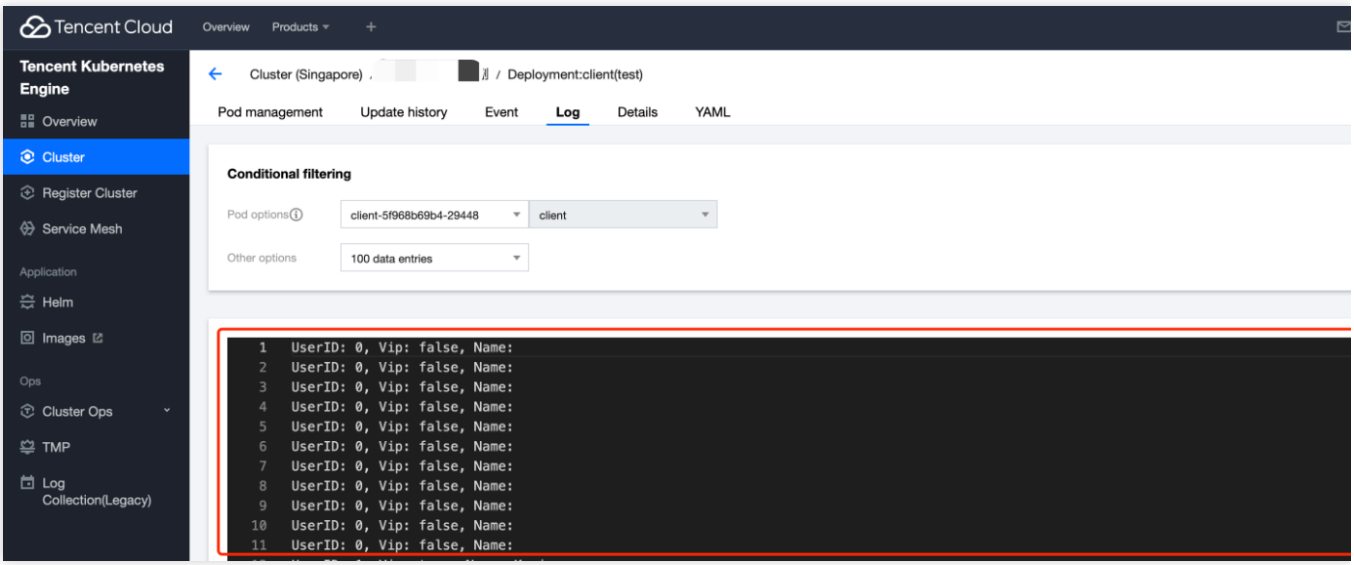


也可以通过提交 YAML 文件至主集群完成授权配置：



```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: base-authz
  namespace: base
spec:
  action: DENY
  rules:
    - from:
        - source:
            namespaces:
```

```
                - test
```

配置完成后通过 TKE 控制台查看 test namespace 下 client 服务的 pod 日志，发现 client 服务访问 base namespace 的 user 服务失败。授权策略生效。

授权规则限制后，访问失败如下图所示：