TKE Ingress获取真实源IP Playbook指南（gr模式实现CLB非直连业务Pod）

2025-07-11 20:33

目录

背景

在腾讯云容器服务（TKE）环境中，通过CLB七层负载均衡器获取客户端真实源IP是常见需求，尤其对于非直连业务Pod场景（gr模式）。

本Playbook详细指导如何在TKE集群中配置Ingress，确保后端Pod能正确获取源IP（通过X-Forwarded-For和X-Real-IP头）。

本指南跳过镜像构建等复杂步骤（使用预推送的Flask镜像），聚焦核心操作。

为什么使用此镜像？

开箱即用：

- 预构建的Flask镜像 `test-angel01.tencentcloudcr.com/kestrelli/kestrel-seven-real-ip:v1.0` 已包含完整源IP捕获逻辑：
- 自动捕获并返回所有HTTP请求头（包括 `X-Forwarded-For` 和 `X-Real-Ip`）。
- 响应为标准JSON格式（如下），便于验证：

```
{
  "headers": {
    "X-Forwarded-For": "客户端真实IP",
    "X-Real-Ip": "客户端真实IP",
    ...    // 其他请求头
  },
  "method": "GET/POST"
}
```

- **适用场景**：CLB七层负载均衡器 + TKE Ingress（非直连Pod模式）。
- **跳过构建步骤**：无需自行构建Docker镜像或推送仓库，简化流程。

核心优化

- 直接使用此镜像替换原流程中的自定义镜像，其他步骤完全兼容。
- 已验证镜像与TKE Ingress（gr模式）的兼容性。

为什么选择gr模式？

- gr模式（非直连）通过Service的NodePort转发流量，保留源IP头，适合CLB七层负载均衡器场景。
- 简化版设计：直接使用预构建的Flask镜像（已推送至腾讯云镜像仓库），避免Docker构建和推送过程，减少错误。

前置条件

在开始前，确保完成以下准备：

1. **腾讯云账号**：已开通容器服务（TKE）、负载均衡器（CLB）、容器镜像服务（TCR）。
2. **TKE集群**：版本≥1.14，且已配置好kubectl命令行工具访问凭证（通过TKE控制台获取）。
3. **基础工具**：安装kubectl并配置集群上下文（参考腾讯云文档）。
4. **预构建镜像**：使用我预推送的Flask镜像 `test-angel01.tencentcloudcr.com/kestrelli/kestrel-seven-real-ip:v1.0`。该镜像基于Flask服务，能打印请求头并返回源IP（无需自行构建）。

5. **权限**： 确保账户有拉取容器镜像服务（TCR）的权限（镜像为公开可读）。

⚠️ **注意**：

- 无需Docker环境或镜像构建知识！
- 若需自定义镜像，请参考文档TKE Ingress获取真实源IP Playbook指南 的镜像构建步骤，但本Playbook为简化跳过此部分。

快速开始

跟随以下步骤操作，每个步骤包括命令、YAML文件和截图指导。所有操作在kubectl命令行完成。

步骤1: 创建Deployment（工作负载）

创建Flask应用的工作负载（Deployment），使用预构建镜像。

1.创建命名空间（可选，但推荐隔离环境）：

```
kubectl create ns kestrel-catchip
# 创建命名空间，名称为kestrel-catchip
```

2.创建Deployment YAML文件（保存为 `catchip.yaml`）：

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: real-ip-deployment
  namespace: kestrel-catchip  # 使用步骤1创建的命名空间
  labels:
    app: real-ip-app  # 标签需匹配后续Service
spec:
  replicas: 2  # Pod副本数，推荐2个确保高可用
  selector:
    matchLabels:
      app: real-ip-app
  template:
    metadata:
      labels:
        app: real-ip-app
    spec:
      containers:
      - name: flask-container
        image: test-angel01.tencentcloudcr.com/kestrelli/kestrel-seven-real-ip:v1.0  # 预构建镜像
        ports:
          - containerPort: 5000  # Flask服务端口，与镜像内一致
```

关键配置说明：

- `image`: 使用预构建镜像，直接拉取无需构建。
- `containerPort: 5000`：Flask服务暴露端口，必须匹配。
- `namespace`: 与步骤1创建的命名空间一致。

3.部署Deployment：

```
kubectl apply -f catchip.yaml
```

4.验证Pod状态：

```
kubectl get pods -l app=real-ip-app -n kestrel-catchip
```

预期输出：看到2个Pod状态为 `Running`（例如：`real-ip-deployment-xxxxx-xxxxx Running`）。

```
[root@VM-17-53-tlinux ~]# kubectl get pods -l app=real-ip-app -n kestrel-catchip
NAME                                   READY   STATUS    RESTARTS   AGE
real-ip-deployment-576c795ccd-8kn8f    1/1     Running   0          2m36s
real-ip-deployment-576c795ccd-v7j5m    1/1     Running   0          2m36s
```

步骤2: 创建Service（NodePort类型）

创建NodePort类型的Service，将外部流量转发到Deployment Pod。

1.创建Service YAML文件（保存为 `svc.yaml`）：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: real-ip-service
  namespace: kestrel-catchip  # 匹配Deployment命名空间
spec:
  selector:
    app: real-ip-app  # 匹配Deployment标签
  ports:
    - protocol: TCP
      port: 80         # Service外部访问端口
      targetPort: 5000 # 映射到Deployment的5000端口
  type: NodePort       # 关键：NodePort模式启用源IP透传
```

关键配置说明：

- `type: NodePort`：启用NodePort模式，这是gr模式非直连的核心，确保源IP保留。
- `targetPort: 5000`：必须匹配Deployment的容器端口。

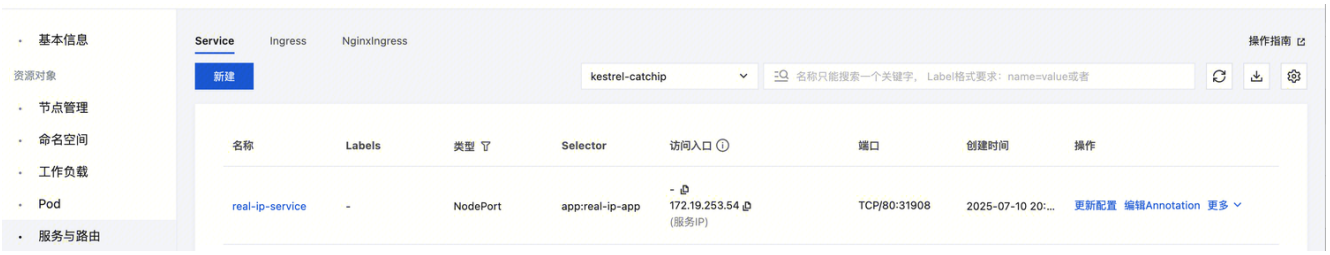
2.部署Service：

```
kubectl apply -f svc.yaml
```

3.验证Service配置：

```
#工作负载指定的命名空间（这里为kestrel-catchip）
kubectl describe svc real-ip-service -n kestrel-catchip
```

```
[root@VM-17-53-tlinux ~]# kubectl describe svc real-ip-service -n kestrel-catchip
Name:                     real-ip-service
Namespace:                kestrel-catchip
Labels:                   <none>
Annotations:              service.cloud.tencent.com/sync-begin-time: 2025-07-10T20:50:23+08:00
                          service.cloud.tencent.com/sync-end-time: 2025-07-10T20:50:23+08:00
Selector:                 app=real-ip-app
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       172.19.253.54
IPs:                      172.19.253.54
Port:                     <unset>  80/TCP
TargetPort:               5000/TCP
NodePort:                 <unset>  31908/TCP
Endpoints:                172.19.0.6:5000,172.19.0.71:5000
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:
  Type    Reason                  Age   From                Message
  ----    ------                  ----  ----                -------
  Normal  EnsureServiceSuccess  2m58s  service-controller  Sync Success. ReturnCode: S2000
```



验证：

```
kubectl get svc real-ip-service -n kestrel-catchip
```

注意 `PORT(S)` 列：`80:31908/TCP`，其中 `31908` 是自动分配的节点端口（NodePort），用于后续Ingress转发。

```
[root@VM-17-53-tlinux ~]# kubectl get svc real-ip-service -n kestrel-catchip
NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
real-ip-service   NodePort   172.19.253.54   <none>        80:31908/TCP   5m28s
```

步骤3: 创建Ingress（核心配置）

创建Ingress资源，配置CLB七层负载均衡器转发规则。

1. 创建Ingress YAML文件（保存为 `ingress.yaml`）：

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: real-ip-ingress
  namespace: kestrel-catchip  # 匹配命名空间
spec:
  ingressClassName: qcloud
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: real-ip-service  # 匹配Service名称
  # 关联上一步的Service
            port:
              number: 80    # 匹配Service端口
```

关键配置说明：

- `ingressClassName: qcloud`：启用腾讯云CLB，确保X-Forwarded-For头透传。
- `service.name`: 必须匹配步骤2的Service名称。

2.部署Ingress：

```bash
kubectl apply -f ingress.yaml
```

3.获取Ingress访问IP：

```bash
kubectl get ingress real-ip-ingress -n kestrel-catchip -o
jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

预期输出：返回一个公网IP（例如 `159.75.190.194`）。这是CLB的入口地址。

```
[root@VM-17-53-tlinux ~]# kubectl get ingress real-ip-ingress -n kestrel-catchip -o jsonpath='{.status.loadBalancer.ingress[0].ip}'
159.75.190.194[root@VM-17-53-tlinux ~]#
```

步骤4: 验证真实源IP

测试配置是否成功，获取客户端真实源IP。

1.执行curl命令（替换为您的Ingress IP）：

```bash
curl http://<上一步获取的IP>  # 例如：curl http://159.75.190.194
```

2.预期输出：

JSON响应中包含 `X-Forwarded-For` 和 `X-Real-Ip` 头，值为您的客户端源IP。

```
{
  "headers": {
    "X-Forwarded-For": "106.55.163.108"",
    "X-Real-Ip": "106.55.163.108"",
    ...  # 其他头信息
  },
  "message": "Here are your request headers"
}
```

```
159.75.190.194[root@VM-17-53-tlinux ~]# curl http://159.75.190.194
{"headers":{"Accept":"*/*","Connection":"keep-alive","Host":"159.75.190.194","User-Agent":"curl/7.61.1","X-Client-Proto":"http","X-Client-Proto-Ver":"HTTP/1.1","X-Forwarded-For":"106.55
.163.108","X-Forwarded-Proto":"http","X-Real-Ip":"106.55.163.108","X-Stgw-Time":"1752152992.769"},"message":"Here are your request headers","method":"GET"}
```

示例：

```
curl 159.75.190.194
# 输出应显示您的客户端源IP在X-Forwarded-For和X-Real-Ip字段中。
```

```
[root@VM-17-53-tlinux ~]# curl 159.75.190.194
{"headers":{"Accept":"*/*","Connection":"keep-alive","Host":"159.75.190.194","User-Agent":"curl/7.61.1","X-Client-Proto":"http","X-Client-Proto-Ver":"HTTP/1.1","X-Forwarded-For":"106.55
.163.108","X-Forwarded-Proto":"http","X-Real-Ip":"106.55.163.108","X-Stgw-Time":"1752153443.934"},"message":"Here are your request headers","method":"GET"}
```

3.验证成功标志：

- 如果输出中包含您的公网IP，表示成功获取真实源IP。
- 失败时参考故障排查表。

故障排查表

基于常见问题整理。如果遇到错误，逐步检查。

问题现象	解决方案
curl命令无响应	1. 检查Ingress IP是否正确：`kubectl get ingress -n kestrel-catchip`。 2. 查看Ingress事件：`kubectl describe ingress real-ip-ingress -n kestrel-catchip`，检查Events是否有错误。 3. 确保集群安全组允许80端口访问。
返回404错误	1. 确认Service名称在Ingress中拼写正确（YAML中的 `service.name` 需匹配）。 2. 验证Deployment标签与Service selector是否一致：`kubectl describe svc real-ip-service -n kestrel-catchip`。
看到Node IP而非公网IP	1. 检查Ingress注解 `ingressClassName: qcloud` 是否配置。 2. 确保Service类型为 `NodePort`（非 `ClusterIP`）。
镜像拉取失败	1. 测试镜像可访问性：在集群VPC内执行 `docker pull test-angel01.tencentclouddcr.com/kestrelli/kestrel-seven-real-ip:v1.0`。 2. 检查TKE集群是否绑定容器镜像服务（TCR）权限。
Pod未运行	1. 查看Pod日志：`kubectl logs <pod-name> -n kestrel-catchip`。 2. 确保Deployment YAML中的 `containerPort` 为5000。
X-Forwarded-For头缺失	1. 确认Ingress配置了 `qcloud` 注解。 2. 确保流量经过CLB七层（直接访问NodePort可能不包含头）。

- **锦囊**：所有YAML文件直接复制即可运行。如果问题持续，在TKE控制台复查资源配置（截图参考步骤中的图片）。

原理解析

理解gr模式如何实现源IP获取，帮助您调试和优化。

流量路径（gr模式非直连）

1. **客户端请求**：用户访问CLB七层负载均衡器（Ingress IP）。
2. **CLB转发**：CLB将请求转发到Ingress Controller，并添加 `X-Forwarded-For` 和 `X-Real-Ip` 头（包含客户端真实IP）。
3. **Ingress到Service**：Ingress根据规则将流量路由到Service（NodePort类型）。
4. **Service到Pod**：Service通过NodePort模式转发到后端Pod（不修改源IP头）。
5. **Pod处理**：Flask应用读取 `X-Forwarded-For` 头，返回真实源IP。

关键设计

- **gr模式优势**：通过NodePort Service非直连Pod，避免了kube-proxy的SNAT操作，确保源IP头保留。
- **镜像作用**：Flask镜像（`kestrel-seven-real-ip:v1.0`）专门处理请求头，打印并响应 `X-Forwarded-For` 和 `X-Real-Ip`。
- **Ingress注解**：`ingressClassName: qcloud` 启用腾讯云CLB七层转发，这是透传源IP的必要条件。
- **端口映射**：Service的 `targetPort:5000` 必须匹配Deployment端口，确保流量正确路由。
- **零构建部署**：直接使用预构建镜像，跳过文档TKE Ingress获取真实源IP Playbook指南中Docker构建（步骤4-6）和推送流程。

为什么能获取真实IP？

CLB七层默认在HTTP头添加源IP，而gr模式NodePort Service不修改这些头，后端Pod直接读取并响应——全链路无IP丢失风险。

总结

本Playbook基于文档TKE Ingress获取真实源IP Playbook指南(简化版)，但网络模式使用GR网络，结构优化为简化版Playbook，跳过镜像构建，使用预推送镜像，适合快速实现。

如果您需要自定义镜像或详细构建步骤，请参考文档TKE Ingress获取真实源IP Playbook指南 的完整流程。部署成功后，真实源IP将稳定返回，适用于日志分析、安全审计等场景。